AI前线

2019年06月刊

AI-FRONT

关注落地技术，探寻AI应用场景

2019极客邦科技会议推荐

5

QCon 北京

全球软件开发大会

大会：5月6–8日
培训：5月9–10日

QCon 广州

全球软件开发大会

培训：5月25–26日
大会：5月27–28日

6

GTLC GLOBAL TECH LEADERSHIP CONFERENCE 上海

技术领导力峰会

时间：6月14–15日

GMTC 北京

全球大前端技术大会

大会：6月20–21日
培训：6月22–23日

7

ArchSummit 深圳

全球架构师峰会

大会：7月12–13日
培训：7月14–15日

10

QCon 上海

全球软件开发大会

大会：10月17–19日
培训：10月20–21日

11

GMTC 深圳

全球大前端技术大会

大会：11月8–9日
培训：11月10–11日

AiCon 北京

全球人工智能与机器学习大会

大会：11月21–22日
培训：11月23–24日

12 (月份)

ArchSummit 北京

全球架构师峰会

大会：12月6–7日
培训：12月8–9日

查看会议详情

Geekbang. 极客邦科技 InfoQ

目录

卷首语

2019，请冷静看待自动驾驶

作者：陈思

起：无人车“杀”人了

“我们预计在2020年实现自动驾驶全面落地！”

“2020是自动驾驶的时代！”

“自动驾驶会在2020年迎来全面爆发！”

……

2017年到2018年初，这样的豪言壮语不止一次地出现在各种与自动驾驶研究相关的发布会上。

然而2018年3月的一场突如其来的车祸，将不少豪言壮语撞得粉碎。

2018年3月25日，一名行人在美国亚利桑那州坦佩被 Uber 的自动驾驶汽车撞击身亡。

经过调查，这起事故是由于自动驾驶车辆的测试安全员失职，才导致了悲剧的发生，根据车载摄影机拍摄的视频，由于受害者是从阴暗处突然冲上道路，毫无疑问在任何驾驶模式（自动或手动）下，这种碰撞都非常难以避免。

同年5月，谷歌旗下自动驾驶公司Waymo也出了一次车祸，事故导致一名测试人员受伤。

后续的调查显示，这起车祸仍旧是人为因素导致的，测试员走神打盹，才没能及时处理突发情况，进而引发了这起悲剧。

但是这样的调查结果并没有打消公众对于自动驾驶的担忧，无人车的安全性成

为了公众关注的焦点。

一组调查数据显示：约 71% 的美国人不信任无人驾驶车辆。甚至有案例显示：Waymo 无人车上路以后，遭到了路人持枪的威胁，扎轮胎、故意阻拦行驶等等…

就这样，自动驾驶变成了“洪水猛兽”，因为大部分人在心里都坚信：真正的无人驾驶系统理论上应该能挽救更多生命，因为相比人类驾驶员，能够全面控制整车的程序更有警惕性，操作也更精准。

但事实真的如此吗？

问：自动驾驶离全面落地到底还差什么？

先来看一组数据。

以 Waymo 的无人车为例：平均每行驶 1.1 亿英里会出现一次人工干预，行驶两千万英里还没有一次死亡事件发生。而美国人类驾驶员的平均水平是这样的：平均每 25 万英里会出一次险，每 50 万英里警方就会收到一次事故的通知，每 150 万英里会出现一次致伤的事故，每 94000 万英里会发生一次致命的事故。

可以看到，即使是 Waymo 的技术，离现在人类的驾驶水平还是有差距的。

所以，理论归理论，现实却总有些残酷。

2018年的两起事故同样波及到了国内的自动驾驶行业，虽然表面上大部分公司仍然强调不必过分担心自动驾驶的安全性，但实际上，已经有一些公司侧面表达过这样的想法：

“2020年全面落地还是有些太着急了。”

当然，也有一些公司默认全面落地的自动驾驶级别是L2-L3，虽然有“文字游戏”之嫌，但是不失为一个宣传的由头，这里就暂且不表。

> 注：L2 级别，系统能够完成某些驾驶任务，但驾驶员需要监控驾驶环境并准备随时接管；

> L3 级别，驾驶员将不再需要手脚待命，机器可以独立完成几乎全部的驾驶操作，但驾驶员仍需要保持注意力集中，以便随应对可能出现的人工智能应对不了的情况;
>
> L4 级别，指车辆能够完全接管驾驶功能，可以不需要驾驶人员，车上人员也无需专注路况，但仅限于特定路段;
>
> L5 级别，指车辆完全自动驾驶，可以不需要驾驶员，车上人员也无需关注路况，且不限路段。

2019年第二季度，InfoQ做了一期名为《中国自动驾驶行业观察》的季度专题，分别对驭势科技、图森未来、小鹏汽车等国内知名科技企业的技术专家们进行了采访，在此过程中，我们总结出了目前国内自动驾驶在全面落地之前还需要努力的一些关键因素：

技术短板仍然存在

从自动驾驶的级别来看：L2级别的自动驾驶是目前市面上的主流，一些较高级配置的车辆都已能够做到类似的功能，比如自动泊车等。

L3级别的自动驾驶在一些车厂也已经实现，最有名的案例莫过于特斯拉一直以来宣传的自动驾驶功能。有不少用户已经拍摄了相关的自动驾驶视频上传到了社交网络，虽然是对自动驾驶技术的最佳宣传，但是特斯拉官方仍然表示：并不支持用户完全放开手脚进行自动驾驶，目前该功能仍然需要驾驶员随时集中注意力观察路况。

而L4级别的自动驾驶则是目前大部分公司正在攻克的难关。

以前文两起事故为例，突然出现的行人或者车辆仍然是自动驾驶的一大技术难题，系统的计算速度还没有快到能够实时处理这类突发情况，不过有专家表示：5G的出现或许能对这一情况有所改善。

另外，极端天气下的自动驾驶也同样困扰着研发人员，激光雷达质量的良莠不

齐、系统对环境的识别等等也都影响着无人车的安全，一个极端的例子是：美国一辆无人车与大货车追尾，原因是将大货车尾部的反光识别成了蓝天白云，于是事故就这样发生了。

正如一位受访的专家所言：商业化落地的前提是要先把技术做好。

政策支持有待跟进

目前来看，已有不少公司在封闭测试路段取得了一定的成就，但是本着更好的宣传效果，企业都会宣称自己的无人车已经可以“安全上路”了，于是问题也跟着来了：能不能上路谁说了算？

所以，政策的支持很重要。

综合几位专家的观点，有这样几类政策是亟需的：

1. 权威第三方检测。不仅是国内，全球都在这一点上有些滞后，缺少权威的第三方检测机构进行评测，企业再怎么宣传也都是纸上谈兵。
2. 运营政策。现在政策主要开放的还仅限于城市，并且重心都在设计测试上，而没有去面向运营方面，所以如果能有一个这样的政策，国内的自动驾驶或许能有更大的进步。
3. 问责政策。自动驾驶车辆上路，势必会引发人们对安全的担忧，那么一旦有事故发生，事故的责任该按照怎样的法律法规来进行划分？
4. 保险政策。事故一旦发生，保险、理赔等条款又该如何规定？

盼：全自动驾驶遥遥无期？

德国计算机科学家Matthias Niessner曾在推特上这样说：

> 参加近几年CVPR的感悟如下：
> - CVPR'17：实现全自动驾驶还要5年
> - CVPR'18：实现全自动驾驶还要10年
> - CVPR'19：全自动驾驶遥遥无期

对于这段话，AI领域的不少专家都深以为然。

不论从技术角度还是宣传角度来说，自动驾驶不再是实现一个demo就敢说落地的时代了，谨慎是第一位的，所谓小心驶得万年船，尤其在自动驾驶出现不少事故之后，谨慎一些总是没错的。

另外，不止一个专家谈到过对全自动驾驶落地的不确定性，毕竟自动驾驶是被称为“集AI大成于一身的专业领域”，有专家认为：突破技术难关可能会花一些时间，但应该不会太远，至于到底是5年、10年、20年甚至更远，谁也说不好。

自动驾驶“最后一公里”真的到来了吗？

采访嘉宾 | 黄波，周鑫，彭进展

作者 & 编辑 | Vincent

AI 前线导读：当“全面量产无人车”成为热门话题的时候，有这么一群人在默默地观察，没有紧锁眉头也没有喜笑颜开，他们能够看到别人看不见的问题，他们是无人车落地的关键，他们要保持冷静。

这群深耕自动驾驶领域的技术专家们，似乎更能淡定地对待外界对无人车的狂热。自动驾驶技术真的成熟了吗？无人车离商业化落地到底还有多远？带着这些问题，AI 前线对驭势科技的多位技术专家进行了独家专访。在热度爆表的无人车话题下，我们想要冷静地谈谈技术。

一辆成熟的无人车应该...

虽然自动驾驶的话题炒得火热，但是相信有不少人对无人车的结构并不了解。

驭势科技的黄波博士告诉 AI 前线：一台成熟的具备自动驾驶功能的汽车需要具备三大核心子系统：感知子系统、规划决策子系统、执行器子系统。三个核心子系统互相紧密配合、缺一不可，如此才能实现可靠、安全的自动驾驶功能。

详细来说，执行器子系统是汽车执行自动驾驶动作的基础，更贴近传统汽车产业，包括线控转向、线控驱动、线控刹车、线控车身控制等等，行业重点是升级成安全可靠新一代线控系统。

以驭势科技为例，驭势 U-Drive™智能驾驶系统集成了规划决策子系统的全部软硬件载体，包括多层面的规划算法及其 AI 硬件、多层面的控制算法及其实时计算软硬件。它还集成了传感器接入、感知处理算法和 AI 算法计算平台，能够接入并且分析感知多摄像头、多激光雷达、多毫米波雷达、多超声波。

软件系统

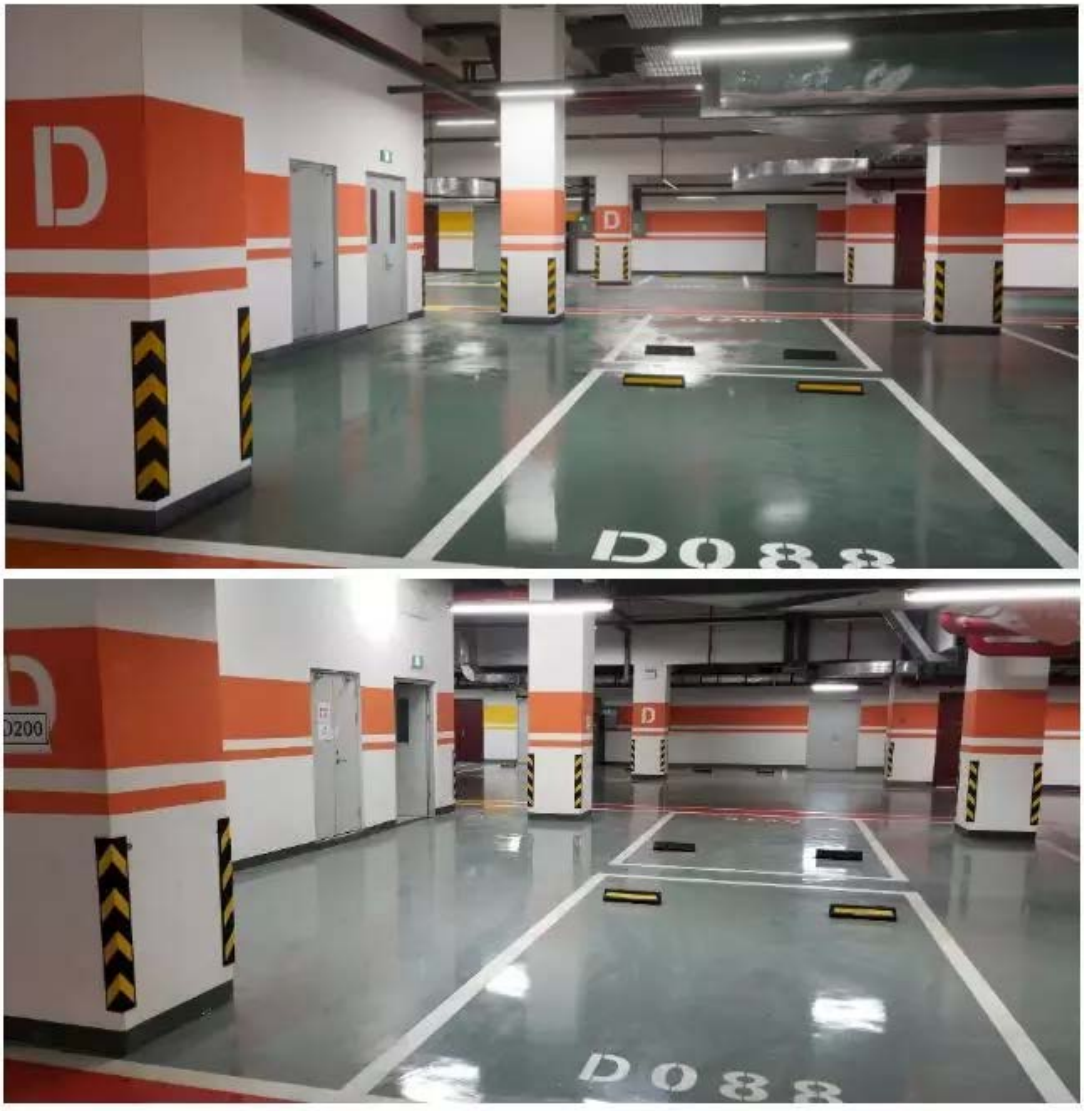
驭势科技 CEO 吴甘沙曾经在北大 AI 公开课上进行过自动驾驶话题的演讲，在谈到技术与成本的话题时，他提到过这样一种思路：

我们都知道，在真实世界里跑一亿英里谈何容易，要耗掉多少油，要消耗多少请司机的人工费用。但在仿真世界里面，只要云计算的算力足够，就可以无限测试，而且这里要求仿真要足够的真，开发者对这种场景是可控的，可以设计任意的复杂的环境。此外，这种环境是能够产生无穷的新场景的，比如模拟建筑、地面、植物、天气、光照条件等等，这是仿真世界要解决的。

为此，驭势科技开发出了一套无人驾驶全功能仿真系统，可以对各类传感器（比如：相机、激光雷达、毫米波雷达、GPS、IMU 等）进行仿真模拟，并支持多个无人驾驶系统同时接入仿真系统，互为障碍物。

正如吴甘沙在演讲中提到的那样，驭势科技构造了一个丰富的仿真场景素材库，

覆盖道路、交通、动态障碍物、路边环境、天气等各个方面，并能通过这些素材库的组合生成更加复杂的仿真场景。这套无人驾驶全功能仿真系统提供的各类仿真场景具有非常高的逼真度，为基于仿真器的 AI 训练和学习提供了一个坚实的基础。



你能识别出哪张图是仿真的吗？

通过仿真环境内进行的大规模测试，研究人员可以对驭势智能驾驶算法及智能驾驶系统进行充分的验证和检测，从而提高驭势智能驾驶系统的质量。黄波博士提到：“在智能驾驶车的实际场景商业化落地前，我们会先确保驭势智能驾驶系统在

仿真环境中测试通过落地场景可能涉及的各类工况，从而极大地提高了驭势智能驾驶系统的场景适应性及安全性，进而提高我们项目落地的信心并缩短项目部署的时间。”

在仿真平台上除了能够验证智能驾驶算法和系统，还可以利用仿真场景来进行算法 / 系统的学习和训练。比如在自主代客泊车（AVP）仿真场景中，训练出来可行驶区域检测网络跟用真实停车场数据训练出来的可行驶区域检测网络达到了几乎一致的检测效果。

硬件装备

一辆成熟的无人车不仅要有强大的核心操作系统，在硬件装备上更是不能含糊。

无人车就像是一台移动的计算机，所以在不少自动驾驶方案里，一个首要的核心部件就是芯片。

驭势科技首席产品官周鑫说：车规级 AI 芯片有嵌入式 GPU、FPGA、AI ASIC 三个发展阶段。

根据他的观察，从行业发展来看，过去两年主要自动驾驶企业以嵌入式 GPU 搭建主动驾驶系统，有实力的企业比如驭势采用了嵌入式 GPU+FPGA 的深度优化方案。未来可能会慢慢过渡到 FPGA+AI ASIC。

这个过程中，全栈系统开发和车规 AI 芯片开发是两个行业层面的工作，财力雄厚的大公司可以承担风险并发进行。但并不意味着当下阶段自研芯片对自动驾驶公司是必须的。一个类比就是手机行业，国内知名手机品牌在成长过程说始终保持着自研芯片和外购芯片同时应用、并不互相冲突。

周鑫告诉记者：驭势科技选择和半导体行业合作，也是看到半导体行业已经大幅度增加车规级 AI 芯片投入开始到了收获季节，与行业合作能够快速获得行业成果；同时驭势科技也认识到现有车规级 AI 芯片必定不能完全满足快速演进的算法，FPGA 研发成果正好能够在 AI ASIC 和算法之间完美契合。

同时，他表示：自动驾驶技术行业前景依赖于快速发展的高性能、低功耗、可

量产、低成本的 AI 芯片的可获得性。

但是，他认为在行业发展不同阶段，行业对芯片依赖不一样，自动驾驶企业对自研芯片的必要性也不一样。当下，自动驾驶行业还处在快速研发阶段，正在积极探索各种行业应用、扩展无人驾驶应用边界，正在 2B 应用落地、2C 应用方案验证的阶段。

对应的无人驾驶感知（深度学习）算法和规划控制算法都在飞速发展、不停推陈出新。这个时候，无人驾驶初创公司的首要任务是充分利用现有车规级 AI 芯片，在新算法和新应用之间构建桥梁、搭建端到端的完整嵌入式解决方案，快速帮助新算法体系和新应用潜力进入车辆服务领域和车辆主机厂技术体系，构建应用闭环。

作为无人车的“眼睛”，摄像头与感知系统尤其重要。

目前业内比较普遍方案一般是基于深度学习的激光雷达和视觉的检测、识别方案，驭势科技采用基于深度学习的检测、识别、跟踪、多传感器融合方案。

驭势尤其深度投入了摄像头感知子系统垂直开发，和多家供应商在摄像头 HDR、光学参数定制、镜头设计方面展开深度协同设计优化，形成了特有竞争力。基于 U-Drive™，驭势科技不仅实现了从传感模块到感知子系统到规划控制子系统的全栈式垂直优化整合、能够对具体 L4 场景做出最优匹配，还能够基于嵌入式硬件和系统软件、对各个算法模块做到深度剪裁和深度优化、提升可靠性提升算力功耗比。

自动驾驶的商业化落地与“最后一公里”

说起自动驾驶，商业化落地是行业内外近期最关注的方向。

作为从业者，驭势科技首席架构师彭进展首先谈了谈自动驾驶领域近些年取得的成果。

根据他的观察，自动驾驶领域这几年进入快速发展期，相关产业链上的核心技术和设备已经进入成熟阶段，比如融合多种传感器的感知和定位技术，高性能高可靠多功能的集成域控制器，增强网络和环境感知的 V2X 技术等。这些核心技术和

设备是自动驾驶能够商业化落地的基础。

就驭势科技来说，目前已经有了以上这些核心技术和设备的产品级解决方案，且已经应用到了非常多的自动驾驶应用中，包括全自主泊车（AVP）、无人机场\工厂物流、无人公交和无人微循环等。

他认为，自动驾驶的商业化落地，**首要考虑的是应用的安全性和可靠性，其次是商业上的效率和成本接受度**。

他进一步解释道：运行在高可靠的域控制器上的融合感知系统，提供了冗余的环境感知和多重定位能力，再辅以增强环境感知的 V2X 技术，让自动驾驶应用有了足够的安全性和可靠性。

彭进展提到：自动驾驶可以减轻甚至消除对人类司机的依赖，能大大提升应用运行效率。

自动驾驶进入商业化应用，会加速各种关键技术和设备的成熟和进入量产的速度，同时大幅度降低自动驾驶应用的整体成本。我们驭势科技的自动驾驶解决方案已经跨过了应用落地的门槛，进入到了技术和商业化落地相互促进和加速发展的阶段。

采访的最后，话题不可避免地又回到“自动驾驶的最后一公里”上。

现在已经是 2019 年了，离“2020 年全面实现自动驾驶”的说法仅有一年的距离，也有一些人把 2019 年称为“实现自动驾驶的最后一公里”，当然在这期间有不少反对的声音，认为 2020 年实现自动驾驶并不现实，无人车这种东西还有至少五到十年才能完全实现。

在彭进展看来，自动驾驶的商业化应用，需要从两个维度来分析，一个是应用的广度，另一个是应用的深度。上面的两种声音实际上是代表着着从这两个不同维度看到的结果。

从应用的广度来看，不同级别的自动驾驶都开始进入实际的应用和运营，包括应用于乘用车市场的 L1/L2/L3 的 ADAS 和 L4 级的 AVP/HZP，以及应用于机场、

港口、矿山和园区的 L4 级的无人物流车和微循环车等。比如驭势科技已经和上汽通用五菱于 2018 年 11 月交付给普通消费者业内首款具备无人驾驶 AVP 功能的量产车宝骏 E200，在世界顶级的国际机场运行了可全天候工作的无人电动物流拖车。“从这个角度来分析，完全可以说 2019 年已经进入‘实现自动驾驶的最后一公里’，2020 年将进入自动驾驶应用全面爆发的新时代。”他说道。

但从应用的深度来讲，他认为可以在公开道路运营的 L4 级自动驾驶应用，依然面临着非常多的技术和商业化挑战。比如复杂场景下的安全决策训练，多重冗余设备的高成本，以及适合完全自动驾驶的道路和网络等基础设施的建立等，都表明类似于 Robotaxi 这样的无人车离实际应用还依然遥远，还需要至少五年以上整个产业甚至整个社会的加入和努力。

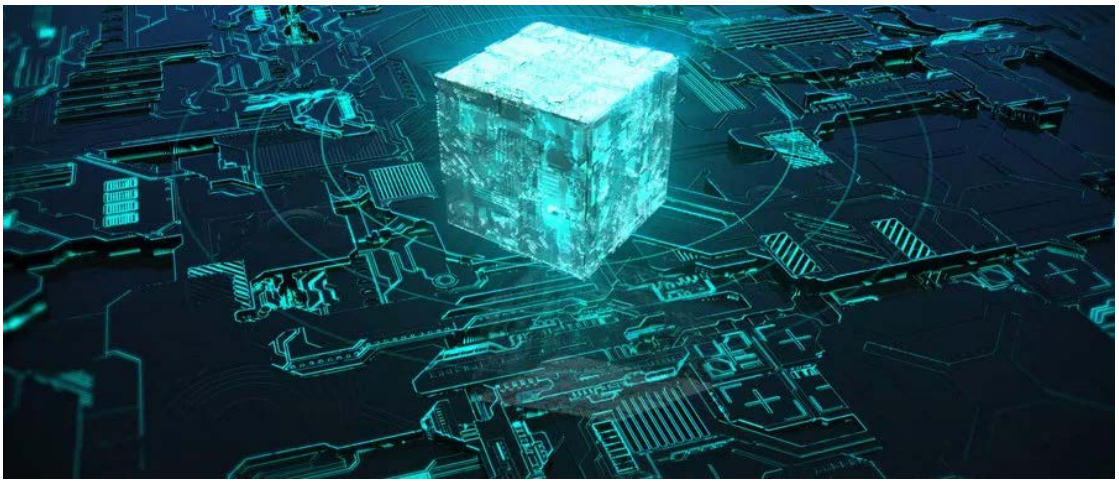
彭进展认为：L3 以下的自动驾驶方案已经逐步在各款中高档量产车型中得到部署，业界也普遍认为 2-3 年后 L3 自动驾驶方案可以被逐步量产。同时，特定场景的 L4 在物流、微循环、短途接驳等各类应用场景也已经开始试点。

在技术层面，更大规模自动驾驶解决方案的落地依赖于下面几个方面的突破：

- 汽车控制结构上经济的安全冗余
- 智能驾驶系统基于数据积累的自动智能增强
- 高性能、低功耗、符合车规且低成本智能域控制器的量产
- 可以量产的低成本且符合车规的高冗余自动驾驶传感器配置
- 具有规范定位接口的高精地图的商业化服务运营
- 高带宽、低延时通讯设备和基础设施的普及

“当然，”彭进展补充说：“非技术层面（比如自动驾驶相关政策、法规、标准、保险等）的进展也对自动驾驶的普及起着至关重要的作用。”

图灵奖得主David Patterson：RISC-V的未来在中国！



采访嘉宾 | 谭章熹

作者 | 陈思

AI 前线导读：当地时间 6 月 12 日， 图灵奖得主、计算机体系结构领域享誉世界的顶级科学家 David Patterson（大卫·帕特森）在瑞士宣布，将依托清华 - 伯克利深圳学院（TBSI），建设 RISC-V 国际开源实验室（RISC-V International Open Source Laboratory），又称 David Patterson RIOS 图灵奖实验室（以下简称 RIOS 实验室）。

AI 前线第一时间对这一事件进行了跟踪报道，并有幸就此事采访到了 David Patterson 的高徒、同时也是主研 RISC-V 技术深圳睿思芯科的创始人谭章熹博士。

David Patterson 是 2017 年图灵奖得主，美国科学院、工程院、艺术与科学学院三院院士，伯克利加州大学电子工程与计算机科学学院 Pardee 荣誉教授。他与清华有多年深厚的友谊和密切的合作，并于 2018 年获颁清华大学名誉博士学位。

据介绍，RIOS 实验室将瞄准世界 CPU 产业战略发展新方向和粤港澳大湾区产业创新需求，聚焦于 RISC-V 开源指令集 CPU 研究领域开展研究，建设以深圳为根节点的 RISC-V 全球创新网络。研究将极大地推动全球 RISC-V 技术的工业化进程和软硬件生态建设。RIOS 实验室由 David Patterson 院士担任实验室主任，将依托清华 - 伯克利深圳学院开展工作。

未来依托该实验室将开展硕士、博士培养项目，并以此作为清华大学深圳国际研究生院核心学科建设的一部分，在教师和研究人员的选聘方面建立配套措施。

对标 ARM，RISC-V 是何方神圣？

作为计算机体系结构领域享誉世界的顶级科学家，David Patterson 最早提出“精简指令集”(RISC) 体系。**第五代精简指令集（RISC-V）是目前最新一代伯克利 RISC 处理器指令集，由 Patterson 教授带领的伯克利加州大学团队于 2011 年首次发布。RISC-V 的硬件和软件技术发展吸引了世界各国的关注**。

特别值得一提的是，该指令集完全开源并免费。David Patterson 院士一直坚持

原创知识成果开源，希望以非商业性的开源运动带动 RISC-V 全球化，形成新的开源 CPU 生态体系。

作为新一代的芯片指令集，RISC-V 不仅与 ARM 形成了竞争，对于 x86 来说，也是一种替代选择，最终可能会给英特尔和 AMD 造成威胁。

ARM 的芯片设计则是基于早期 RISC 架构。ARM 将这种设计授权给多家芯片厂商，而此类芯片几乎被用于所有智能手机。在桌面和服务器市场，既有芯片架构的地位非常牢固，作为一个“新品”，RISC-V 为何被认为能够挑战到 ARM 等架构的位置呢？

作为 David Patterson 院士的门徒，谭章熹不止一次在公开场合谈到 ARM 与 RISC-V 的区别。

他告诉 AI 前线：“ARM 是一个完全私有化、封闭式的指令机器；而 RISC-V 更像是 Linux 这样的，由完全中立的国际组织拥有的，完全开放的架构。”

谭章熹补充说，RISC-V 比 ARM 更晚出现，但 RISC-V 的架构设计比 ARM 更为精简，同时 RISC-V 还有模块化的特点，非常适合物联网等等新的应用，他表示：“目前 RISC-V 很多的创新都是通过用户在现有的指令架构上面增加指令子集的方式来实现的。”

在谭章熹看来，RISC-V 最大的好处就是：不存在任何的法律纠纷，因为它是完全开源的，他告诉 AI 前线：“也正因如此，包括印度、以色列甚至欧洲一些国家，都相继的把 RISC-V 定为所谓的‘国家指令集’，而一些新的处理器设计和学术研究，也都纷纷把它作为指令架构。”

“RISC-V 的未来在中国”

David Patterson 教授曾经说过：“**RISC-V 的未来在中国，而中国的半导体芯片技术和市场发展也离不开 RISC-V**。”

对于这句话，谭章熹深以为然，而他本人也致力于将 RISC-V 的未来发展重心

放在中国。

2017 年 2 月，谭章熹作为联合创始人在硅谷成立了一家名为“OURS”的公司，主要开发基于 RISC-V 技术的 SoC，应用于 IoT、传感器融合和 AI 加速领域。

2018 年下半年，谭章熹团队开始在中国组建团队，并在深圳成立一家名为“睿思芯科”（RiVAI）的公司，将 RISC-V 的发展重点逐渐转向中国。

此次 RIOS 实验室的合作，在谭章熹看来也是“天时、地利、人和”的结果。

首先，清华与伯克利加州大学的合作基础深厚。

早在 1979 年，两校便签署了改革开放以来清华大学与国外大学的第一个学术交流协议。40 年来，两校共同致力于解决全球面临的重大挑战，在能源、环境、健康、经济、社会发展等众多领域开展了合作研究。

基于多年的合作互信关系， 2014 年，清华大学与伯克利加州大学在深圳市政府的支持下联合建立“清华 - 伯克利深圳学院”，凭借两校的综合学科优势和雄厚工科基础，吸引世界一流的生源与顶尖的教授和研究者，致力于培养产业科学家、解决中国的世界级问题。

其次，RIOS 实验室在深圳的建设也有着深远的意义。

实验室的研究方向契合深圳市战略性新兴产业布局，将极大推动 RISC-V 的工业化进程和在全球的广泛产业应用，进一步建立与全球各大公司的联系与合作。

RIOS 实验室落户深圳，还将为深圳聚集和培养面向处理器和开源硬件设计的高端急需人才，有助于深圳国际化科技生态圈建设及智能硬件产业链布局，提升粤港澳大湾区在 RISC-V 开源处理器全球生态圈中的话语权与影响力。

此外，谭章熹表示：RISC-V 作为一种颠覆性的开源技术，它真正能够成长起来的土壤并不是在已经被高度开发，并且非常成熟的环境下。恰恰相反，这种新的技术接受度更高的地方会是中国。中国有一个巨大的市场，拥有各种各样的应用，RISC-V 本身是开放的，它没有法律专利问题，并且支持很多定制化的需求，种种的

RISC-V 的特点表明，它非常适合在中国市场发展，RISC-V 对中国的市场也会有非常正面的影响。

合作规划与展望

对于未来的合作发展，谭章熹本人还是非常期待的。

他告诉 AI 前线：“我们（RIOS）会有一个为期 5 年的项目，会有很多来自中外不同规模的公司一起，在非盈利的开源模式下进行合作，对整个 RISC-V 生态进行完善，与业界的合作是 RIOS 工作中非常重要的一块内容。”

另外谭章熹表示，他们也意识到，CPU 的发展需要各种各样的人才，因为合作方是学校，而学校的意义不仅是做科研，更需要培养人才。因此，他希望通过 RIOS 和 TBSI 培养出 100 个跟 RISC-V 相关的硕士毕业生，“我想这不光是对整个 RISC-V 的生态有积极的作用，对深圳，包括中国在芯片领域的影响力都是非常正向的，而且还增进了清华和伯克利两所名校之间的合作。”谭章熹补充道。

而对于 RISC-V 生态的打造，谭章熹却表现得更为谨慎。

他认为：“生态并不是一个公司来打造或者运营的，睿思芯科也只是这个生态里面其中一个公司，我们也是非常愿意与任何做 RISC-V 的国内外公司进行合作。”

在他看来，现在 RISC-V 生态的最大竞争者其实是像 ARM 这样的对手，而不是 RISC-V 本身这个行业里面的这些公司。他进而说道：“在 RISC-V 行业里的这些企业要积极合作，因为比如在编译器、操作系统等等很多的地方目前并不完善，所以，我们公司现在做的一些东西，将来慢慢的也会贡献给生态，我们认为这种贡献从长远来讲，对生态、对企业的商业利益都是有好处的。”

最快开源OLAP引擎！ClickHouse在头条的技术演进

演讲嘉宾 | 陈星

整理 | Natalie

编辑 | Vincent

AI 前线导读：ClickHouse 是由号称“俄罗斯 Google”的 Yandex 公司开源的面向 OLAP 的分布式列式数据库，能够使用 SQL 查询生成实时数据报告。本文整理自字节跳动高级研发工程师陈星在 QCon 全球软件开发大会（北京站）2019 上的演讲，他介绍了 ClickHouse 的关键技术点、在字节跳动的应用场景以及主要的技术改进。

ClickHouse 简介

ClickHouse 是由号称“俄罗斯 Google”的 Yandex 开发而来，在 2016 年开源，在计算引擎里算是一个后起之秀，在内存数据库领域号称是最快的。大家从网上也

能够看到，它有几倍于 GreenPlum 等引擎的性能优势。

如果大家研究过它的源码，会发现其实它采用的技术并不新。ClickHouse 是一个列导向数据库，是原生的向量化执行引擎。它在大数据领域没有走 Hadoop 生态，而是采用 Local attached storage 作为存储，这样整个 IO 可能就没有 Hadoop 那一套的局限。它的系统在生产环境中可以应用到比较大的规模，因为它的线性扩展能力和可靠性保障能够原生支持 shard + replication 这种解决方案。它还提供了一些 SQL 直接接口，有比较丰富的原生 client。另外就是它比较快。

大家选择 ClickHouse 的首要原因是它比较快，但其实它的技术没有什么新的地方，为什么会快？我认为主要有三个方面的因素：

1. 它的数据剪枝能力比较强，分区剪枝在执行层，而存储格式用局部数据表示，就可以更细粒度地做一些数据的剪枝。它的引擎在实际使用中应用了一种现在比较流行的 LSM 方式。
2. 它对整个资源的垂直整合能力做得比较好，并发 MPP+ SMP 这种执行方式可以很充分地利用机器的集成资源。它的实现又做了很多性能相关的优化，它的一个简单的汇聚操作有很多不同的版本，会根据不同 Key 的组合方式有不同的实现。对于高级的计算指令，数据解压时，它也有少量使用。
3. 我当时选择它的一个原因，ClickHouse 是一套完全由 C++ 模板 Code 写出来的实现，代码还是比较优雅的。

字节跳动如何使用 ClickHouse

头条做技术选型的时候为什么会选用 ClickHouse？这可能跟我们的应用场景有关，下面简单介绍一下 ClickHouse 在头条的使用场景。

头条内部第一个使用 ClickHouse 的是用户行为分析系统。该系统在使用 ClickHouse 之前，engine 层已经有两个迭代。他们尝试过 Spark 全内存方案还有一些其他的方案，都存在很多问题。主要因为产品需要比较强的交互能力，页面拖拽的方式能够给分析师展示不同的指标，查询模式比较多变，并且有一些查询的 DSL 描述，也不好用现成的 SQL 去表示，这就需要 engine 有比较好的定制能力。

行为分析系统的表可以打成一个大的宽表形式，join 的形式相对少一点。系统的数据量比较大，因为产品要支持头条所有 APP 的用户行为分析，包含头条全量和抖音全量数据，用户的上报日志分析，面临不少技术挑战。大家做了一些调研之后，在用 ClickHouse 做一些简单的 POC 工作，我就拿着 ClickHouse 按需求开始定制了。

综合来看，从 ClickHouse 的性能、功能和产品质量来说，效果还不错，因为开发 ClickHouse 的公司使用的场景实际上跟头条用户分析是比较类似的，因此有一定的借鉴意义。

目前头条 ClickHouse 集群的规模大概有几千个节点，最大的集群规模可能有 1200 个节点，这是一个单集群的最大集群节点数。数据总量大概是几十个 PB，日增数据 100TB，落地到 ClickHouse，日增数据总量大概是它的 3 倍，原始数据也就 300T 左右，大多数查询的响应时间是在几秒钟。从交互式的用户体验来说，一般希望把所有的响应控制在 30 秒之内返回，ClickHouse 基本上能够满足大部分要求。覆盖的用户场景包括产品分析师做精细化运营，开发人员定位问题，也有少量的广告类客户。

图 1 是一个 API 的框架图，相当于一个统一的指标出口，也提供服务。围绕着 ClickHouse 集群，它可以支撑不同的数据源，包括离线的数据、实时的消息中间件数据，也有些业务的数据，还有少量高级用户会直接从 Flink 上消费一些 Databus 数据，然后批量写入，之后在它外围提供一个数据 ETL 的 Service，定期把数据迁移到 ClickHouse local storage 上，之后他们在这之上架了一个用户使用分析系统，也有自研的 BI 系统做一些多维分析和数据可视化的工作，也提供 SQL 的网关，做一些统一指标出口之类的工作，上面的用户可能是多样的。

综合来说，我们希望在头条内部把 ClickHouse 打造成为支持数据中台的查询引擎，满足交互式行为的需求分析，能够支持多种数据源，整个数据链路对业务做到透明。在工作过程中，我们也碰到了很多的问题。

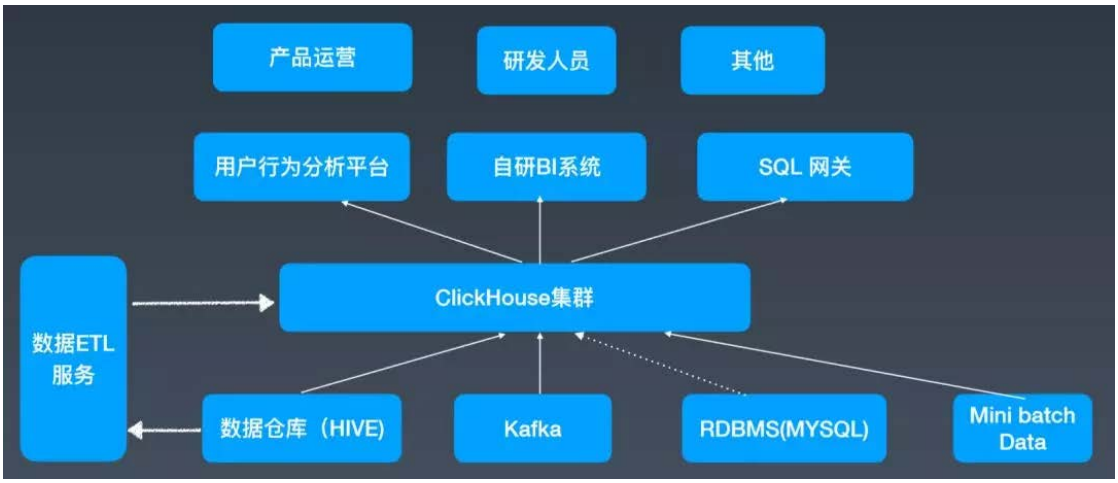


图 1 Bytedance 如何使用 ClickHouse

问题与解决方案

接下来我会详细介绍我们在使用 ClickHouse 的过程中碰到过什么问题，希望对大家有一些借鉴意义。

数据源到 ClickHouse 服务化

我们在做 ClickHouse 服务化的过程中，第一步就是如何把数据落到 ClickHouse 集群中。原生的 ClickHouse 没有 HDFS 访问能力，我们同时还需要保证对用户透明，就可能存在几个问题：

第一，怎么访问离线数据？

第二，ClickHouse 没有事务支持，如果在数据导入过程中发生了 Fail，如何做 Fail over？

第三，ClickHouse 数据就绪速度。我们整个数据就绪的压力很大，上游就绪的时间比较晚，每天早上就会有一些分析师在 ClickHouse 上看指标，整个数据落到 ClickHouse 留给我们的空挡可能不是太长。

我们针对这些问题做了一些改动。第一，从 HAWQ 上移植过来 HDFS client，

让 ClickHouse 能够直接访问数据，我们的 ETL 服务实际上维护了一套外部事务的逻辑，然后做数据一致性的保证；为了保证就绪时间，我们充分利用各个节点的计算能力和数据的分布式能力，实际上最终都会在外围服务把数据作一些 Repartition，直接写入各个节点本地表。另外，我们还有一些国际化的场景，像 TikTok、Musical.ly 等，数据就绪和分析师分析的时间是有重叠的，数据写和查询交互的影响还是有一些。我们最近也在尝试把数据构建和查询分离出来，并开发相应的 Feature，但是还没有上线，从 Demo 来看，这条路是行得通的。

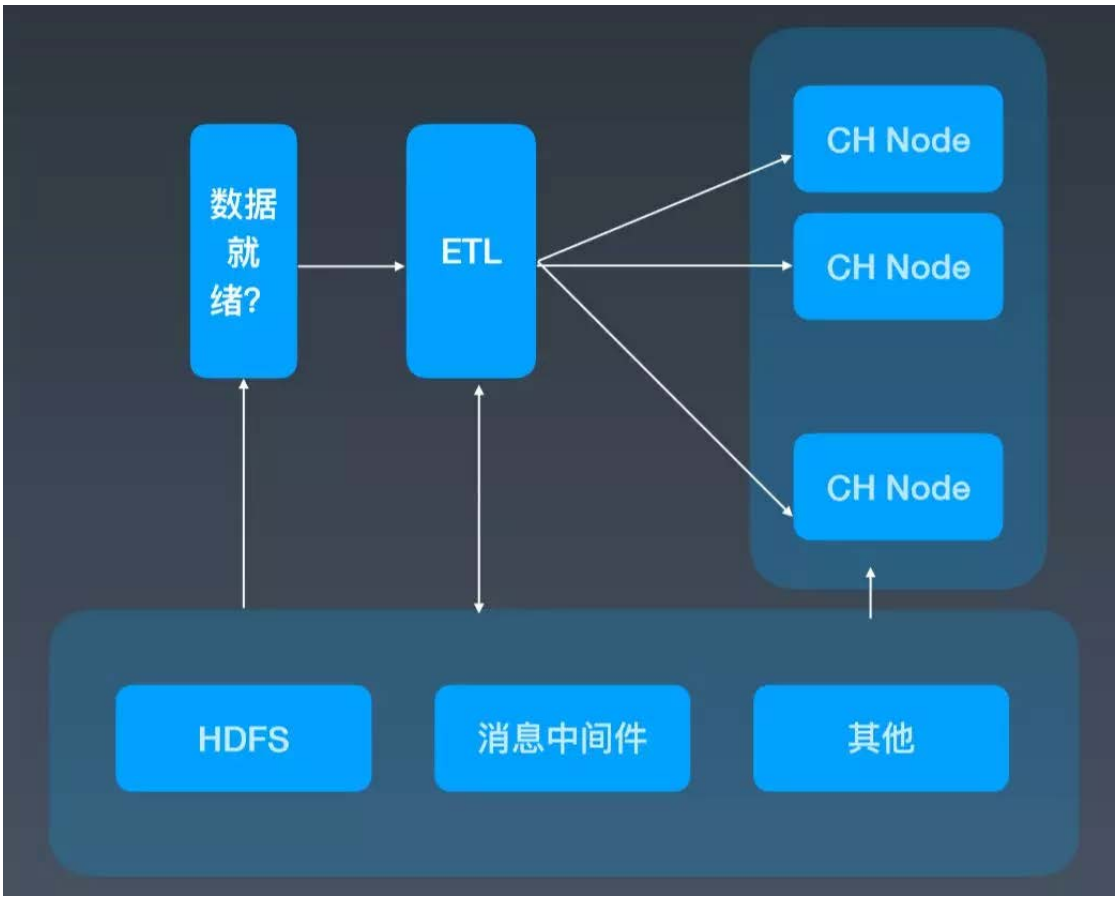


图 2 ClickHouse 服务化与自动化中的问题

Map 数据类型：动态 Schema

我们在做整个框架的过程中发现，有时候产品存在动态 Schema 的需求。我们当时增加了 Map 的数据类型，主要解决的问题是产品支持的 APP 很多，上报的

Model 也是多变的，它跟用户的日志定义有关，有很多用户自定义参数，就相当于动态的 Schema。从数据产品设计的角度来看又需要相对固定的 Schema，二者之间就会存在一定的鸿沟。最终我们是通过 Map 类型来解决的。

实现 Map 的方式比较多，最简单的就是像 LOB 的方式，或者像 Two-implicit column 的方式。当时产品要求访问 Map 单键的速度与普通的 column 速度保持一致，那么比较通用的解决方案不一定能够满足我们的要求。当时做的时候，从数据的特征来看，我们发现虽然叫 Map，但是它的 keys 总量是有限的，因为依赖于用户自定义的参数不会特别多，在一定的时间范围内，Keys 数量会是比较固定的。而 ClickHouse 有一个好处：它的数据在局部是自描述的，Part 之间的数据差异自动能够 Cover 住。

最后我们采用了一个比较简单的展平模型，在我们数据写入过程中，它会做一个局部打平。以图 3 为例，表格中两行总共只有三个 key，我们就会在存储层展开这三列。这三列的描述是在局部描述的，有值的用值填充，没有值就直接用 N 填充。现在 Map 类型在头条 ClickHouse 集群的各种服务上都在使用，基本能满足大多数的需求。

{'a': 1, 'b':2 }　{'a':1, 'c':3} 会在存储层表示为column_a, column_b, column_c 三个列，对应的值如下图所示：

column_a	column_b	column_c
1	2	N
1	N	3

图 3 局部 PART level 展平模型（自描述）

另外，为了满足访问 key 的高效性，我们在执行层做自动改写，key 的访问会直接改写成对隐私列的访问。这样架构会有一个比较大的问题，它对于 Map 列的全值访问代价比较大，需要从隐式列反构建出全值列。对于这个问题，我们也没有很好地解决，因为实际上在很多时候我们只关心 key 的访问效率。

另外一个问题，这是 LSM 架构，存在一个数据合并的过程，合并时可能需要

重构 Map。我们为了提高合并的速度，做了一些相应的优化，可以做到无序重构。这些做完后，收益还是比较大的。首先，Table 的 schema 能够简化，理论上现在 Table 的定义只需要做几种技术类型的组合就可以；然后 ETL 构建的逻辑不再需要关注用户的隐私列参数，可以简化 ETL 的构建逻辑；最后，对数据的自动化接入帮助也很大。图 4 是我们优化之后的语法，大家可以看到相对比较简单。

```
Create table t(c1 UInt64, c2 Map(String, UInt8)) ENGINE=MergeTree....

insert into t values(1, {'abc':1, 'bcd':2})

Select c2{'abc'} from t
```

图 4 Map 数据类型 - 动态 Schema 相关语法

大数据量和高可用

不知道大家在使用 ClickHouse 的过程中有没有一个体会，它的高可用方案在大的数据量下可能会有点问题。主要是 zookeeper 的使用方式可能不是很合理，也就是说它原生的 Replication 方案有太多的信息存在 ZK 上，而为了保证服务，一般会有一个或者几个副本，在头条内部主要是两个副本的方案。

我们当时有一个 400 个节点的集群，还只有半年的数据。突然有一天我们发现服务特别不稳定，ZK 的响应经常超时，table 可能变成只读模式，发现它 znode 的太多。而且 ZK 并不是 Scalable 的框架，按照当时的数据预估，整个服务很快就会不可用了。

我们分析后得出结论，实际上 ClickHouse 把 ZK 当成了三种服务的结合，而不仅把它当作一个 Coordinate service，可能这也是大家使用 ZK 的常用用法。ClickHouse 还会把它当作 Log Service，很多行为日志等数字的信息也会存在 ZK 上；还会作为表的 catalog service，像表的一些 schema 信息也会在 ZK 上做校验，这就会导致 ZK 上接入的数量与数据总量会成线性关系。按照这样的数据增长预估，ClickHouse 可能就根本无法支撑头条抖音的全量需求。

社区肯定也意识到了这个问题，他们提出了一个 mini checksum 方案，但是这

并没有彻底解决 znode 与数据量成线性关系的问题。所以我们就基于 MergeTree 存储引擎开发了一套自己的高可用方案。我们的想法很简单，就是把更多 ZK 上的信息卸载下来，ZK 只作为 coordinate Service。只让它做三件简单的事情：行为日志的 Sequence Number 分配、Block ID 的分配和数据的元信息，这样就能保证数据和行为在全局内是唯一的。

关于节点，它维护自身的数据信息和行为日志信息，Log 和数据的信息在一个 shard 内部的副本之间，通过 Gossip 协议进行交互。我们保留了原生的 multi-master 写入特性，这样多个副本都是可以写的，好处就是能够简化数据导入。图 6 是一个简单的框架图。

以这个图为例，如果往 Replica 1 上写，它会从 ZK 上获得一个 ID，就是 Log ID，然后把这些行为和 Log Push 到集群内部 shard 内部活着的副本上去，然后当其他副本收到这些信息之后，它会主动去 Pull 数据，实现数据的最终一致性。我们现在所有集群加起来 znode 数不超过三百万，服务的高可用基本上得到了保障，压力也不会随着数据增加而增加。

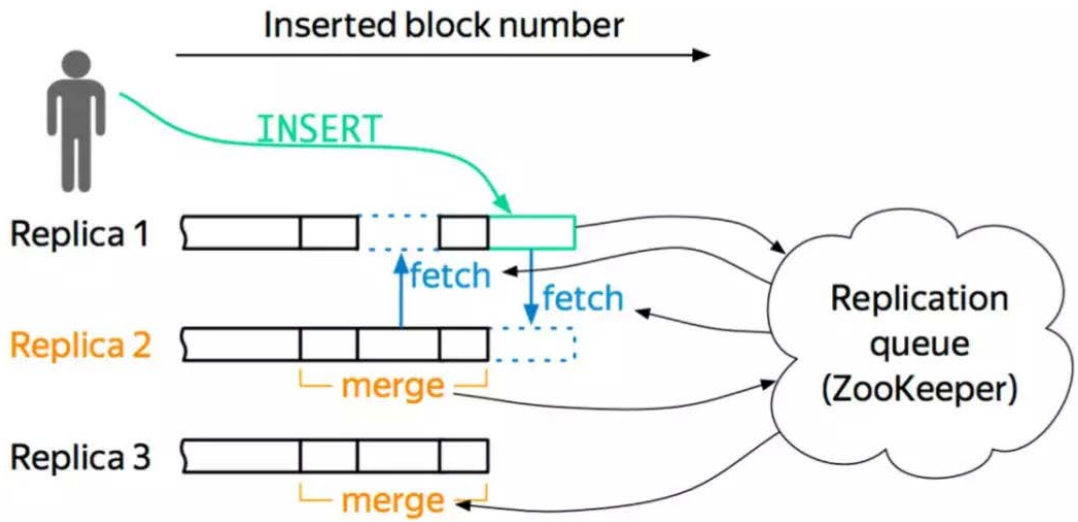


图 5 zookeeper 使用问题

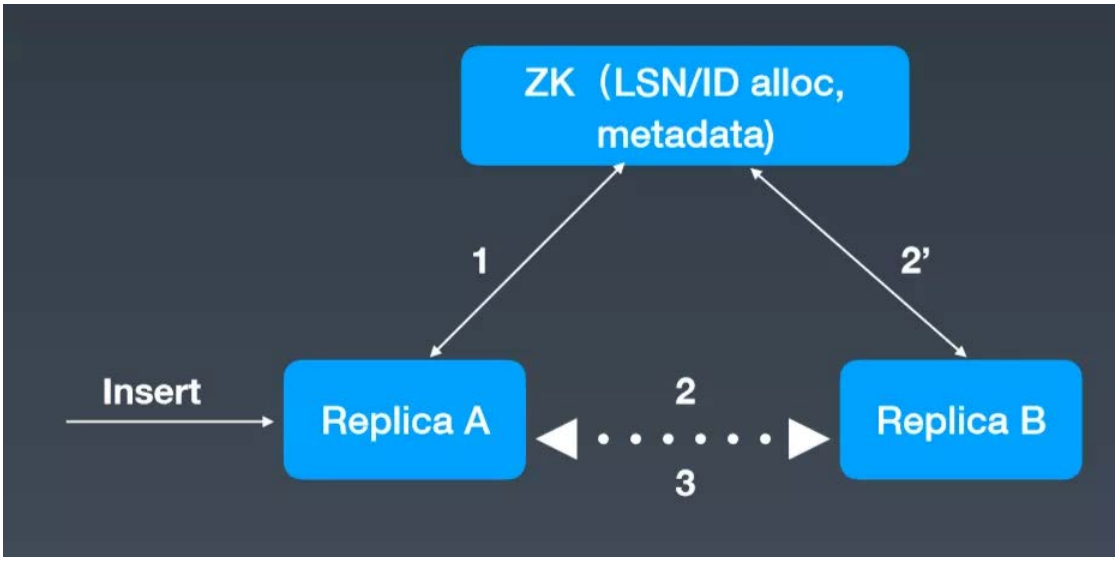


图 6 HaMergeTree 简单框架

解决了以上几个问题之后，我们还在对 ClickHouse 做持续改进。我们最近也碰到了一些 Log 调度之类的问题，当时我们对 Log 调度并没有做特别的优化，实际上还是用 ClickHouse 的原生调度，在有些集群上可能会碰到一些问题，比如有些表的 Log 调度延迟会比较高一点，我们现在也正在尝试解决。

String 类型处理效率：Global Dictionary

另外，为了满足交互式的需求，在相当长的一段时间我们都在思考怎么提高数据执行的性能。大家在做数仓或者做大数据场景的时候会发现，用户特别喜欢字符串类型，但是你如果做执行引擎执行层，就特别不喜欢处理这类 String 类型的数据，因为它是变长的，存在执行上有较高代价。String 类型的处理效率，跟数字类型的处理效率有 10 倍的差距，所以我们做了一个全局字典压缩的解决方案，目的并不是为了节省存储空间，而是为了提高执行的效率，这是相当重要一个出发点。我们希望把一些常见的算子尽量在压缩域上执行，不需要做数据的解压。

目前我们只做了一个 pure dictionary compression，支持的算子也比较少，比如 predication 支持等值比较或者 in 等类似的比较能够在压缩域上直接执行，这已经能够覆盖我们很多的场景，像 group by 操作也能够在压缩域上做。

说到 Global Dictionary，其实也并不是完全的 Global ，每个节点有自己的 Dictionary，但是在一个集群内部，各个节点之前的字典可能是不一样的。为什么没有做全局在集群内部做一个字典？

第一，全局字典会把 coordinate 协议搞得特别复杂，我以前做数据库的时候有个项目，采用了集群级别 Global Dictionary，碰到了比较多的挑战。字典压缩只支持了 MergeTree 相关的存储引擎。压缩的行为发生主要有三种操作，像数据的插入或者数据的后台合并，都会触发 compression，还有很多数据的批量 roll in 或 roll out，也会做一些字典的异步构建。

刚才也提到，我们的主要出发点就是想在执行层去做非解压的计算，主要是做 Select query，每一个 Select 来的时候，我们都会在分析阶段做一些合法性的校验，评估其在压缩域上直接执行是否可行，如果满足标准，就会改写语法树。如果压缩的 column 会出现在输出的列表中，会显式地加一个 Decompress Stream 这样可选的算子，然后后续执行就不太需要改动，而是可以直接支持。当 equality 的比较以及 group by 操作直接在压缩上执行，最后整体的收益大概提高 20% 到 30%。

刚才提到，我们的字典不是一个集群水平的，那大家可能会有所疑问，比如对分布式表的 query 怎么在压缩域上做评估？我们稍微做了一些限制，很多时候使用压缩场景的是用户行为分析系统，它是按用户 ID 去做 shard，然后节点之间基本做到没有交互。我们也引入了一个执行模式，稍微在它的现有计算上改了一下，我们叫做完美分布加智能合并的模式。在这个模式下，分布式表的 query 也是能够在字典上做评估。收益也还可以，满足当时设计时候的要求。

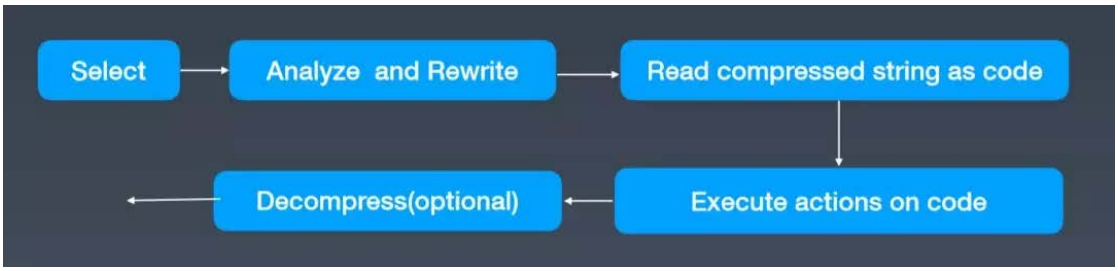


图 7 压缩域执行

特定场景内存 OOM

有时候做一个系统，内存使用的问题也会比较严重。尤其当做数据量大的系统时，经常发生内存受限的问题，或者说 OOM 最后被系统杀掉。ClickHouse 因为有很多数据的加速，比如 Index & mark 文件，信息会在实例启动的时候加载，整个加载过程非常慢，有时候一个集群起来可能得要半个小时。

虽然我们对这个问题做了一些优化，能够做到并行加载，但是也得好几分钟。如果实例被系统 Kill 了之后，对服务还会有影响，我们的系统经常要回答一些用户这样的查询，例如需要查 60 天内用户的转化率或者整个用户的行为路径对应的每天转化率。这种 Block 的操作需要把很多数据从底层捞出来，在时间纬度上进行排序，找出对应的模式。

如果不进行优化，基本上一个 Query 需要使用的内存会超过一百 G，如果稍微并发一下，内存那可能就支撑不了。并且，由于其使用的内存分配器的原因，也很难把内存的实际使用量限制得很准，这就偶尔会发生被系统 Kill 的场景。

我们想从 engine 优化的角度去解决问题，本质上就是 Blocked Aggregator 的操作，它没有感知到底层的数据分布。这个 Feature 有点意思，也是我们从数据分布到执行共同优化的一个尝试，实现相对来说比较粗糙，但是现在线上也已经开始用了。

它的思路是这样的，我们的 Aggregator 执行路径可以由 HINT 来控制，HINT 的生成是由上面的产品生成的，因为产品能够感知数据分布，也能够知道这些指标的语义。HINT 最关键的一个作用是把 Blocked Aggregator 局部做到流水线化，比如计算 60 天的指标，它可以生成一个 read planner 控制底层的 reader，每一批处理的是那一部分数据。上层的指标输出可以把这些信息 aggregate 到对应的地方，做从下向上的执行输出。最上层的 schedule 流输出指标可以把每天的计算结果汇聚起来，然后做一个总体的整理，最终就形成一个输出。

这些优化工作完成以后有了很明显的收益，与默认没有开启的时候相比，系统的内存使用可能会下降 5 倍左右。现在应用场景主要在两个指标的计算上，像漏斗之类的和计算用户行为路径会使用。

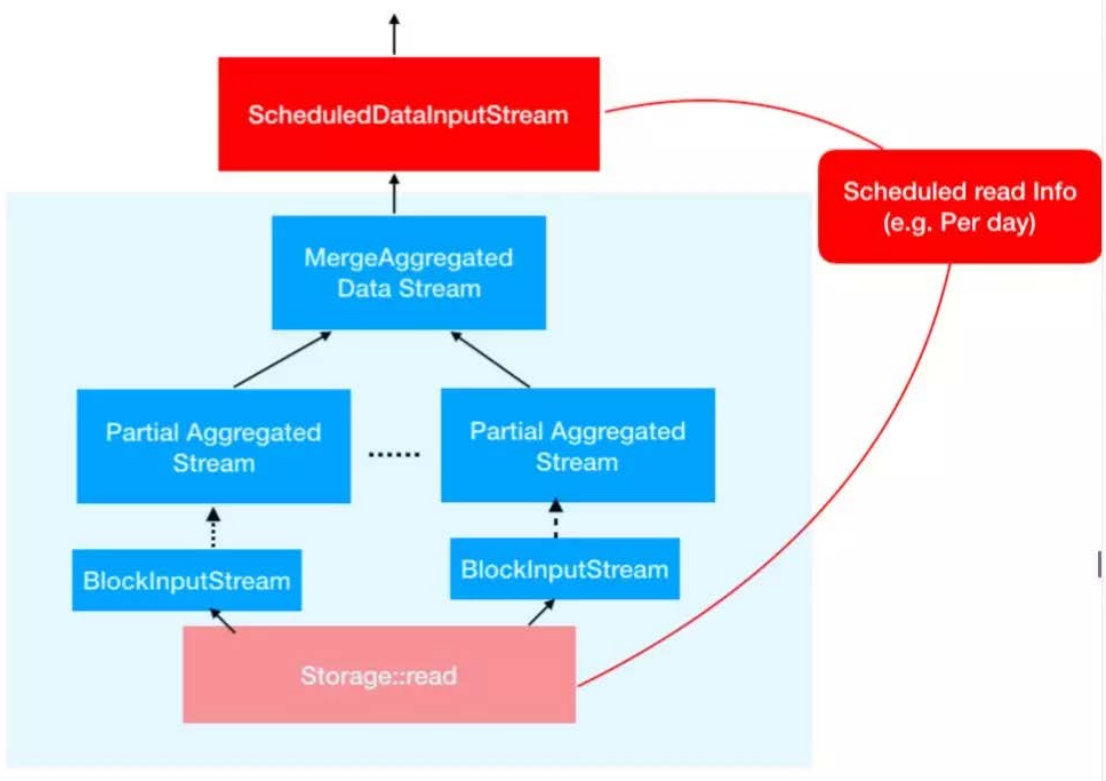


图 8 特定场景内存 OOM - Step-ed Aggregation

Array 类型处理

下面介绍一下我们怎么处理 Array 类型，并将它做得更高效。、

Array 类型处理的需求主要来自于 AB 实验的需求。当前我们的系统也会做一些实时 AB 指标的输出，实验 ID 在我们系统中以数组的形式存储。头条内部的 AB 实验也比较多，对于一个单条记录，它可能命中的实验数会有几百上千个。有时候我们需要查询命中实验的用户行为是什么样的，也就是要做一些 Array hasAny 语义的评估。

从 Array 执行来看，因为它的数据比较长，所以说从数据的反序列化代价以及整个 Array 在执行层的 Block 表示来说不是特别高效，要花相当大的精力去做 Array column 的解压。而且会在执行层消耗特别大的内存去表示，如果中间发生了 Filter 的话，要做 Block column 过滤，permutation 会带上 Array，执行起来会比较慢。那我们就需要一个比较好的方式去优化从读取到执行的流程。

做大数据，可能最有效的优化方式就是怎么样做到底层数据的剪枝，**数据少是提高数据处理速度的终极法宝**。我们提出了现在的剪枝方法，一个是 Part level，一个是 MRK range level。那有没有一种针对于 Array column 的剪枝方式？我们做了下面两个尝试：

首先做了一个双尺度的 Bloom Filter，记录 Array 里面 Key 的运动情况，实现了 Part level 和细粒度的 MRK range level，做完后在一些小的产品上效果还挺好的，但最后真正在大产品上，像抖音、头条全量，我们发现 Fill factor 太高，实际上没太大帮助。之后我们开发了一个 BitMap 索引，基本的想法是把 Array 的表示转化成 Value 和 Bit 的结合。在执行层改写，把 has 的评估直接转换成 get BitMap。

做完之后，我们上线了一两个产品，在一些推荐的场景上使用。这个方案主要问题就是 BitMap 数据膨胀问题稍微严重了一点，最近我们也做了一些优化，可能整体的数据占用是原始数据的 50% 左右，比如 Array 如果是 1G，可能 Bit map 也会有 500M。我们整个集群的副本策略是一个 1：N 的策略，副本存储空间比较有压力，我们现在也没有大范围的使用，但效果是很好的，对于评估基本上也会有一、二十倍的提升效果。

其他问题和改进

以上是我今天分享的主要内容，后面的内容相对比较弹性。字节跳动自身的数据源是比较多样的，我们对其他数据源也做了一些特定的优化。比如我们有少量业务会消费 Kafka，而现在的 Kafka engine 没有做主备容错，我们也在上面做了一些高可用的 HaKafka engine 的实现，来做到主备容错以及一些指定分区消费功能，以满足一些特定领域的需求。

另外，我们发现它的 update/delete 功能比较弱，而我们有一部分业务场景想覆盖业务数据库上面的数据，像 MySQL 上也是会有一些增删操作的。我们就基于它 Collapse 的功能做了一些设计，去支持轻量级的 update/delete，目前产品还属于刚起步的阶段，但是从测试结果来看，能够支撑从 MySQL 到 ClickHouse 的迁移，基于 delta 表的方案也是可行的。

我们还做了一些像小文件读取的问题，提供了一个多尺度分区的方案，但由于

各种原因没有在线上使用。

说到底，我们的需求还有很多，现在也还有很多工作正在做，比如控制面的开发，简化整体运维，还有 Query cache 以及整个数据指标的正确性还不能达到百分之百的保障，特别是像实时数据流的数据，我们也想做更深层次的优化。我们还希望增强物化视图，也准备提高分布式 Join 能力，因为我们自研 BI 对此还有比较强的需求，未来我们会在这一块做一些投入。

以上就是去年一年我们在 ClickHouse 这块主要做的一些工作。总体来说 ClickHouse 是一个比较短小精干的引擎，也比较容易上手和定制，大家都可以去尝试一下。

演讲嘉宾介绍

陈星，字节跳动高级研发工程师，主要负责 ClickHouse 查询引擎相关的技术规划、改进等工作，在字节跳动之前就职于 IBM，从事过多年的数据库研发工作，在 OLAP、OLTP 等领域有深厚的技术积累。

下载完整版 PPT：

https://2019.qconbeijing.com/schedule?utm_source=infoq&utm_campaign=full&utm_term=0605

单机训练速度提升640倍！独家解读快手商业广告模型GPU训练平台Persia

作者 | 快手 FeDA 智能决策实验室

编辑 | Natalie

AI 前线导读：2018 年是快手的“商业化元年”。为了联结用户体验和商业价值，快手开始推行个性化的广告推荐。截止 5 月底，快手的 DAU 已经突破 2 亿，且将在明年 1 月达到 3 亿 DAU 的目标。随着快手用户和使用时长的迅速增长，为了更好地挖掘海量用户和实时数据的核心价值，推荐模型需要快速迭代，从而对用户兴趣迁移的做出迅捷的反应。因此，模型训练效率成为至关重要的一环。

基于历史原因，行业内推荐模型的训练大都通过 CPU 来实现。然而随着模型从 Logistic Regression 到深度神经网络的演化以及硬件的发展，**基于 CPU 的训练**

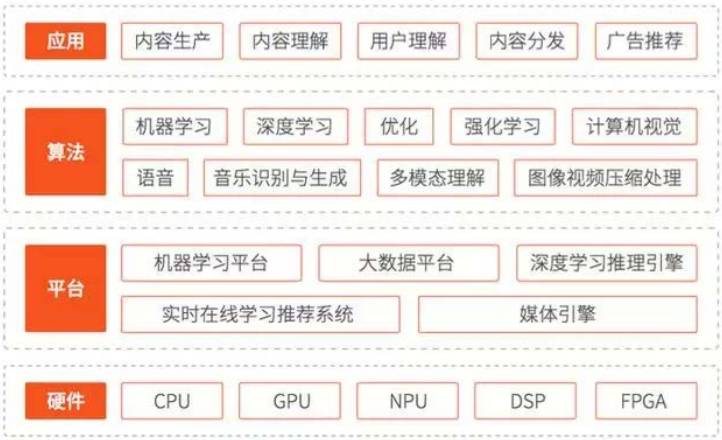
系统或许已经不再是最合适的解决方案了。本着不盲从、不抄袭、坚持原创技术路线的原则，快手西雅图 FeDA 智能决策实验室推出了名为"Persia"的基于 GPU 的广告推荐训练系统。以往需要 50 台 CPU 机器训练 20 小时的系统，如今只需要一台普通的 GPU 机器在一到两小时完成，单机效率提升高达 640 倍。

这意味着：

- 以往使用五十台计算机，一天只能尝试一个新想法，新系统只需一台计算机，一两个小时就能尝试一个新想法。
- 以往同时只能有一两个同学尝试新模型，新系统可以让很多同学同时尝试各自的新想法。

这套系统已经在快手商业化内部迅速推广使用，让大家可以快速试错和测试新模型以及特征。项目发起者是一位来自罗切斯特大学的实习生。他提出的 GPU 解决方案得到他在罗切斯特大学的导师、FeDA 智能决策实验室负责人刘霁和公司内很多算法策略专家的肯定。

FeDA 实验室随即成立了项目组，并决定以项目发起人最喜爱的漫画角色 Persia（“佩尔西亚”）命名，展开了紧锣密鼓的开发。团队首先以 PyTorch 为基础平台着手解决各种技术难题，然后实现并优化 TensorFlow 版本。经过 4 个月的开发和通力合作，Persia GPU 广告训练系统初步成型。系统同时支持 PyTorch 和 TensorFlow 两套方案，以方便模型开发同学的不同偏好。目前，Persia 已支持多个业务项目，每位研发人员只需要一台机器便可以迅速地迭代试错。



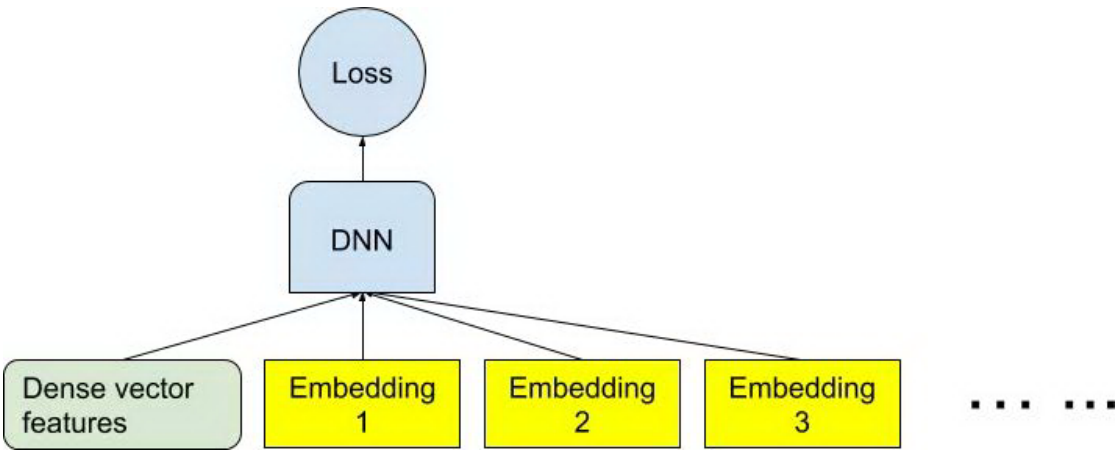
Persia 背后的技术

Persia 实现高效训练背后的技术包含 GPU 分布式训练、高速数据读取等多个方面。

GPU 分布式运算加速模型训练效率

近年来，GPU 训练已在图像识别、文字处理等应用上取得巨大成功。GPU 训练以其在卷积等数学运算上的独特效率优势，极大地提升了训练机器学习模型，尤其是深度神经网络的速度。然而，在广告模型中，由于大量的稀疏样本存在（比如用户 id），每个 id 在模型中都会有对应的 Embedding 向量，因此广告模型常常体积十分巨大，以至于单 GPU 无法存下模型。目前往往将模型存在内存中，由 CPU 进行这部分巨大的 Embedding 层的运算操作。这既限制了训练的速度，又导致实际生产中无法使用比较复杂的模型——因为使用复杂模型会导致 CPU 对给定输入计算时间过长，无法及时响应请求。

广告模型的构成：在广告模型中，模型往往由下图中的三部分构成：



- 用户 id、广告 id 等构成的 Embedding 层。每个 id 对应一个预设大小的向量，由于 id 数量往往十分巨大，这些向量常常会占据整个模型体积的 99% 以上。假设我们有 m1（注：因排版原因，此处 1 为下标，下同）种这样的 id：$\{id_i\}_{i=1}^{m1}$，它们对应的 Embedding 层$\{E_i\}_{i=1}^{m1}$将会输出 m1 个向量。

- 图像信息、LDA 等实数向量特征。这部分将会与 id 对应的 Embedding vector 组合在一起，输入到 DNN 中预测点击率等。假设我们有 m2 种这样的向量：$\{dense_i\}_{i=1}^{m2}$。
- DNN。这部分是一个传统神经网络，接受 Embedding vector 和实数向量特征，输出点击率等希望预测的量：prediction = $DNN([E_1(id_i), E_2(id_2), ..., E_{m_1}(id_{m_1}), dense_1, dense_2, ..., dense_{m_2}])$。

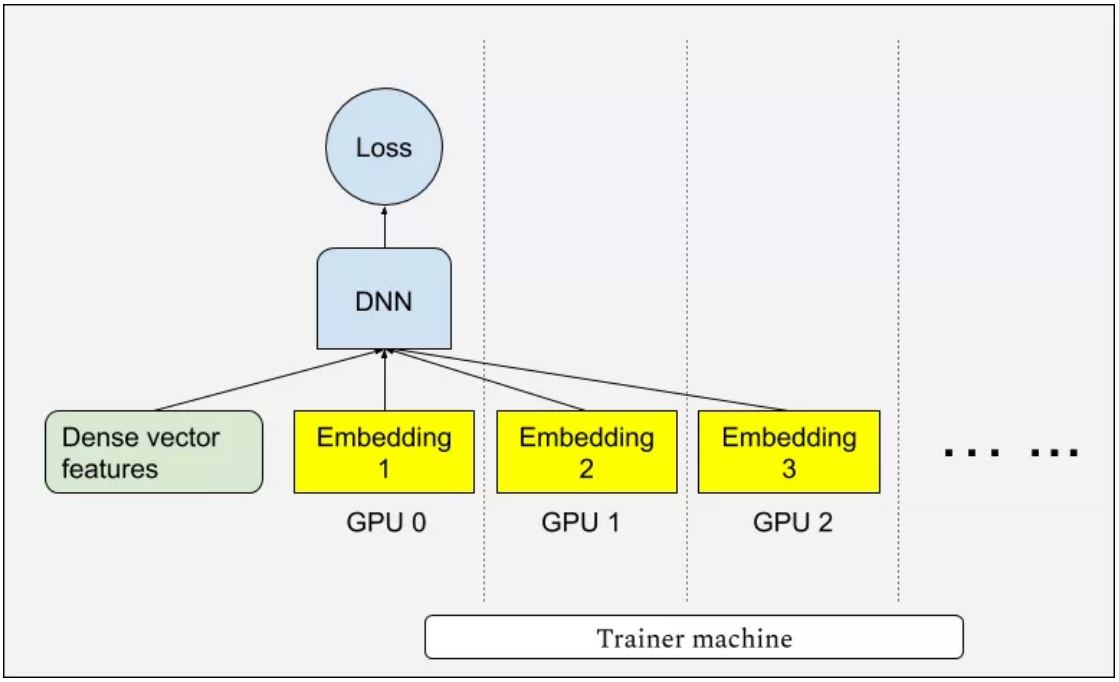
Persia 使用多种技术训练广告模型，我们将在接下来几节依次介绍。

1. 大模型 Embedding 分片训练

广告模型的 Embedding 部分占模型体积和计算量的大部分。很有可能无法放入单个 GPU 的显存中。为了使用 GPU 运算以解决 CPU 运算速度过慢的问题，但又不受制于单 GPU 显存对模型大小的限制，Persia 系统使用多 GPU 分散存储模型，每个 GPU 只存储模型一部分，并进行多卡协作查找 Embedding 向量训练模型的模式。

Persia 将第 i 个 Embedding 层 Ei 放入第 (i% 总显卡数) 个显卡中，从而使每个显卡只存放部分 Embedding。与此同时，实数向量特征和 DNN 部分则置于第 0 个显卡中。在使用 Persia 时，它将自动在各个显卡中计算出 $\{E_i\}_{i=1}^{m1}$ 的值（如果对于一个 Embedding 输入了多个 id，则计算其中每个值对应的 Embedding vector 的平均），并传送给第 0 个显卡。第 0 个显卡会合并这些 Embedding vector 和实数向量特征，输入 DNN 中进行预测。

当求解梯度时，第 0 个显卡会将各个 Embedding 层输出处的导数传回各个显卡，各个显卡各自负责各自 Embedding 的反向传播算法求梯度。大致结构如下图所示：



GPU 分配的负载均衡：由于将 Embedding 依次分配在每个 GPU 上，可能导致部分 GPU 负载显著高于其他 GPU，为了让每个 GPU 都能充分发挥性能，Persia 训练系统还支持对 Embedding 运算在 GPU 上进行负载均衡。

给定 k 个 GPU，当模型的 m1 个 Embedding 层对应 GPU 负载分别为 l1, l2, …,lm1，Persia 将会尝试将 Embedding 分为 k 组 S1, S2, …, Sk，并分别存放在对应 GPU 上，使得每组 $\sum_{i \in S_j} l_i, \forall j$ 大致相等。这等价于如下优化问题：

$$\min_{S_1, \ldots, S_k} \text{Variance}_j[\sum_{i \in S_j} l_i],$$

$$\text{s.t.} \sum_{i \in S_j} V_i \leq C,$$

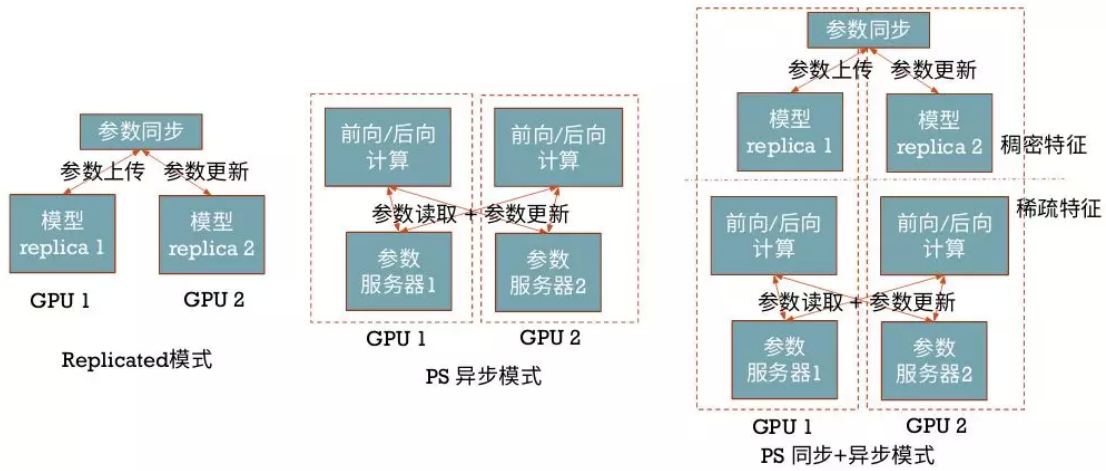
其中 Vi 是第 i 个模型的大小，C 是单个 GPU 的显存大小。Persia 使用贪心算法得到该问题的一个近似解，并依此将不同 Embedding 均匀分散在不同 GPU 上，以达到充分利用 GPU 的目的。当需要精确求解最优的 Embedding 放置位置时，Persia 还可以通过 integer optimization 给出精确解。

2. 简化小模型多 GPU 分布训练

当模型大小可以放入单个 GPU 时，Persia 也支持切换为目前在图像识别等任务中流行的 AllReduce 分布训练模式。这样不仅可以使训练算法更加简单，在某些情景下还可以加快训练速度。

使用这种训练模式时，每个 GPU 都会拥有一个同样的模型，各自获取样本进行梯度计算。在梯度计算后，每个 GPU 只更新自己显存中的模型。需要注意的是即使模型可以置于一个 GPU 的显存中，往往 Embedding 部分也比较大，如果每次更新都同步所有 GPU 上的模型，会大大拖慢运算速度。因此 Persia 在 AllReduce 模式下，每次更新模型后，所有 GPU 使用 AllReduce 同步 DNN 部分，而 Embedding 部分每隔几个更新才同步一次。这样，即不会损失太多信息，又保持了训练速度。

此外，在 TensorFlow 上，Persia 还支持 TensorFlow 的"Replicated", "PS", "PS" + "Asynchronous" 模式多卡训练，它们的主要区别如下图：



模式	同步方式	精度	速度	支持模型大小
Replicated模式	同步	高	较快	较小
PS 同步模式	同步	高	快	较大
PS 同步+异步模式	sparse 异步/dense 同步	较高	快	大

模型准确度提升

同步更新：由于普遍使用的传统异步 SGD 有梯度的延迟问题，若有 n 台计算机参与计算，每台计算机的梯度的计算实际上基于 n 个梯度更新之前的模型。在数学上，对于第 t 步的模型 xt，传统异步 SGD 的更新为：

$$x_{t+1} \leftarrow x_t - \text{learning rate} \times g(x_{t-\tau_t}),$$

其中 $g(x_{t-\tau_t})$ 是训练样本的损失函数在 τ t 个更新之前的模型上的梯度。而 τ t 的大小一般与计算机数量成正比，当计算机数量增多，$x_{t-\tau_t}$ 与 xt 相差就越大，不可避免地导致模型质量的降低。Persia 的训练模式在 Embedding 分片存储时没有这种延迟问题，而在 AllReduce 模式下也仅在 Embedding 层有常数量级的延迟，因此模型质量也有所提升。

优化算法：与此同时，Persia 还可以使用 Adam 等 momentum optimizer，并为其实现了 sparse 版本的更新方式，比 PyTorch/TensorFlow 内置的 dense 版本更新在广告任务上快 3x-5x。这些算法在很多时候可以在同样时间内得到比使用 SGD 或 Adagrad 更好的模型。

训练数据分布式实时处理

快手 Persia 的高速 GPU 训练，需要大量数据实时输入到训练机中，由于不同模型对样本的需求不同，对于每个新实验需要的数据格式可能也不同。因此 Persia 需要：

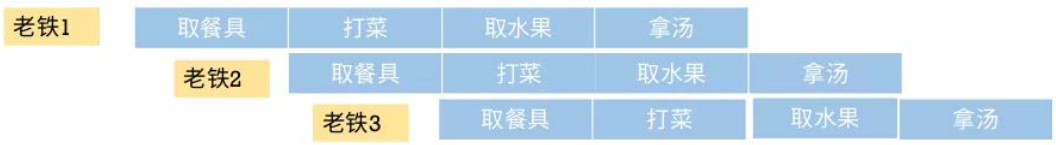
- 简单灵活便于修改的数据处理流程，
- 可以轻易并行的程序架构，
- 节约带宽的数据传输方式。

为此，Persia 系统实现了基于 Hadoop 集群的实时数据处理系统，可以应不同实验需求从 HDFS 中使用任意多计算机分布式读取数据进行多级个性化处理传送到训练机。传输使用高效消息队列，并设置多级缓存。传输过程实时进行压缩以节约

带宽资源。

1. 并行数据处理

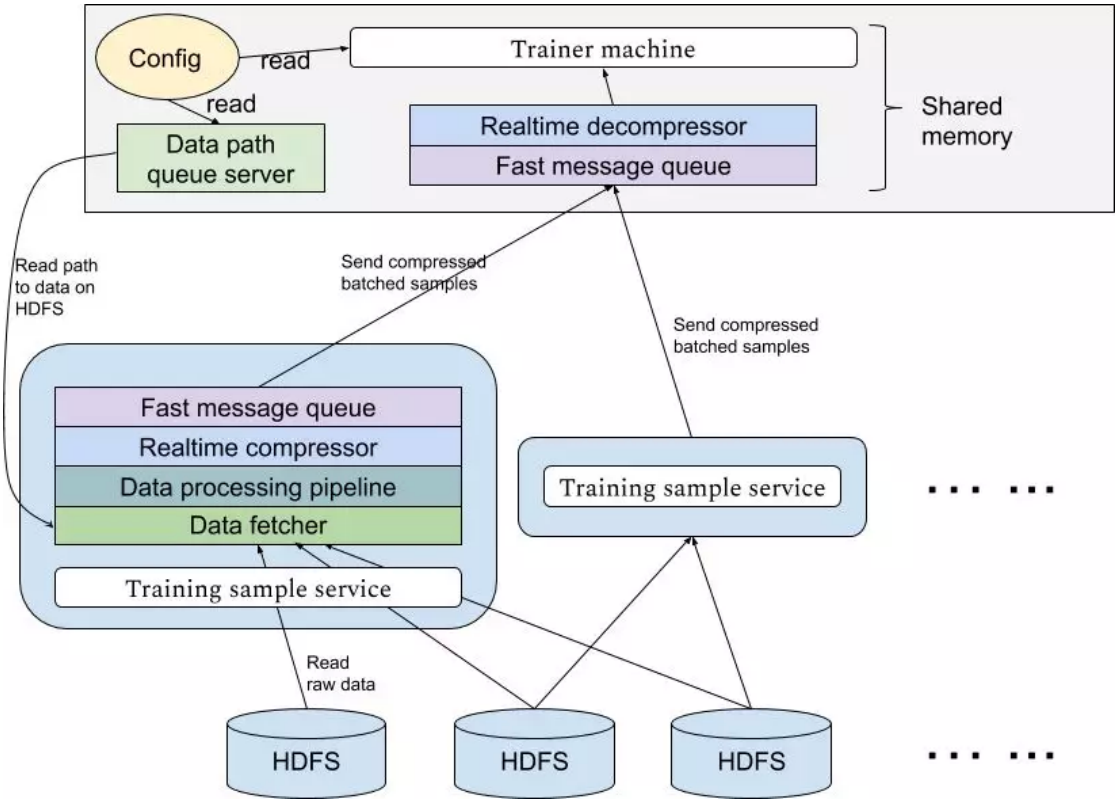
数据处理 pipeline：为了使 Persia 获取数据的方式更灵活，Persia 使用 dataflow 构建数据处理 pipeline。在 Persia 中可以定义每一步处理，相当于一个函数，输入为上一个处理步骤的输出，输出提供给下一个处理步骤。我们定义这些函数为 $\{f_i\}_{i=1}^{p}$。在 Persia 中，这些函数可以单独定义修改。在每个函数的入口和出口，Persia 有数据队列缓存，以减少每个函数获取下一个输入的时间。这些函数的运行可以完全并行起来，这也是 pipeline 的主要目的。以在食堂就餐为例，pipeline 的运行就像这样：



数据压缩和传输：全部处理之后，数据处理任务会将数据组成 mini-batch 并使用 zstandard 高速压缩每个 batch，通过 ZeroMQ 将压缩数据传输给训练机进行训练。定义 batching 操作为函数 B，压缩操作为函数 C，则每个数据处理任务相当于一个函数$C(B(f_p(f_{p-1}(\cdots f_1(\text{rawdata from HDFS})))))$。

Queue server：在 Hadoop 集群中 Persia 将启动多个数据处理任务，每个数据处理任务之间完全独立。数据处理任务本身并不知道处理哪些数据，而是通过请求训练机得知训练数据的位置。这样的好处是，在 Persia 中训练机可以应自己需求动态控制使用什么样的训练数据，而数据处理任务相当于一个无状态的服务，即使训练机更换了新的训练任务也不需要重启数据处理任务。具体来说，在 Persia 中训练机会启动一个 queue server 进程，该 queue server 将会应数据处理任务的请求返回下一个需要读取的数据文件。Persia 的每个数据处理任务会同时从 queue server 请求多个文件，并行从 HDFS 读取这些文件。

整个系统的构造如下图：



2. 实时训练

由于 Persia 的数据处理任务在获取数据时完全依赖于训练机的指示，Persia 支持对刚刚生成的数据进行在线训练的场景，只需要使 queue server 返回最近生成的数据文件即可。因此，Persia 在训练时的数据读取模式上非常灵活，对 queue server 非常简单的修改即可支持任意数据读取的顺序，甚至可以一边训练一边决定下一步使用什么数据。

3. 更快的数据读取速度：训练机共享内存读取数据

由于训练机要同时接收从不同数据处理任务发送来的大量数据，并进行解压缩和传输给训练进程进行实际训练的操作，接收端必须能够进行并行解压和高速数据传输。为此，Persia 使用 ZeroMQ device 接收多个任务传输而来的压缩数据，并使用多个解压进程读取该 device。每个解压进程独立进行解压，并与训练进程共享内存。当结束解压后，解压进程会将可以直接使用的 batch 样本放入共享内存中，

训练任务即可直接使用该 batch 进行训练，而无需进一步的序列化反序列化操作。

训练效果

Persia 系统在单机上目前实现了如下训练效果：

- 数据大小：百 T 数据。
- 样本数量：25 亿训练样本。
- 8 卡 V100 计算机，25Gb 带宽：总共 1 小时训练时间，每秒 64 万样本。
- 8 卡 1080Ti 计算机，10Gb 带宽：总共不到 2 小时训练时间，每秒 40 万样本。
- 4 卡 1080Ti 达 30 万样本 / 秒，2 卡 1080Ti 达 20 万样本 / 秒。
- Persia 同样数据上 Test AUC 高于原 ASGD CPU 平台。
- Persia 支持很大 batch size，例如 25k。

综上，Persia 不仅训练速度上远远超过 CPU 平台，并且大量节省了计算资源，使得同时尝试多种实验变得非常方便。

展望：分布式多机训练

未来，Persia 系统将展开分布式多 GPU 计算机训练。有别于成熟的计算机视觉等任务，由于在广告任务中模型大小大为增加，传统分布式训练方式面临计算机之间的同步瓶颈会使训练效率大为降低。Persia 系统将支持通讯代价更小、系统容灾能力更强的去中心化梯度压缩训练算法。据快手 FeDA 智能决策实验室负责人刘霁介绍，该算法结合新兴的异步去中心化训练 (Asynchronous decentralized parallel stochastic gradient descent, ICML 2018) 和梯度压缩补偿算法 (Doublesqueeze: parallel stochastic gradient descent with double-pass error-compensated compression, ICML 2019），并有严格理论保证，快手 Persia 系统在多机情景下预计还将在单机基础上做到数倍到数十倍效率提升。

超越TensorFlow！未来我们需要基于图的全新计算模式

作者 | 蔡芳芳

编辑 | Linda

AI 前线导读：图神经网络是当前 AI 领域最为火爆的研究热点之一，学术界与工业界各大公司纷纷投入大量资源研究。它在因果推理上拥有巨大潜力，有望解决深度学习无法处理的关系推理、可解释性等一系列问题，而这些问题被业界认为是能够推动 AI 出现实质性进展的关键。

目前图神经网络尚处于发展早期，虽然研究成果和论文频出，但真正在业界落地并对外分享的案例仍然偏少。为了更好地了解图神经网络在实际业务落地中可能遇到的挑战和解决方案，InfoQ 采访了极验 Geetest 的算法负责人刘忠雨，对 GCN 的落地场景和实际效果、图计算框架和模型的选型经验、算法落地部署的难点等话题进行逐一探讨。

“图数据本身非常复杂，深度学习数据样本之间可以随意切割，但图数据不行，因为节点之间存在关联。深度学习时代，我们通常基于 TensorFlow 框架，将数据 Tensor 放到一个计算图里在显卡中做计算。未来，我们的计算模式可能是完全基于一张图，通过节点之间的信息传递与聚合这样一个视角去组织新的计算模式。那可能就称之为 GraphFlow，而不是 TensorFlow 了。”

GCN 天然适合关联模式识别

极验 Geetest 深耕交互安全领域，业务方向包括行为验证、身份认证、风控等。这些业务场景无一不需要收集大量用户行为数据或异常数据，从中挖掘出异常行为的模式，进而提供实时拦截、标记和风险控制。而这正是图神经网络擅长处理的：挖掘海量数据之间的关联模式。

极验在 2014 左右开始使用机器学习和深度学习解决行为验证过程中的人机判别问题，主要是利用 CNN 对用户行为轨迹数据进行自动学习和建模。在这个过程中，研发团队逐渐发现了隐藏其中的数据风险关联模式。黑产在进行攻击的时候，往往是调用某些打码平台的 API 接口或者不断复用自己构建的一套破解代码，这都导致黑产的数据之间存在一些或明或暗的关联模式，风险的传递性很强。最初，研发团队只能通过手工规则来定义这些关联模式，但这种规则定义方式过于简单，很容易被黑产绕过。

后来极验开始使用图聚类的方法挖掘异常，比如最早的社群检测，再到后来的高密子图等算法。这些算法是无监督算法，优点是不需要人工添加标签，但灵活性比较差。如果数据有了更多信息，或有更好的标签质量，这些信息很难融合到原来的算法模型中。如此一来，信息融合和关系建模成为了突破的关键。

2017 年，图神经网络研究开始在业界崭露头角，也进入了极验研发团队的视线。图蕴含了大量多元化信息，能够建立起更全的数据维度，进而比较全面地识别黑产数据。极验从 2018 年开始引入 GCN（图卷积神经网络），利用 GCN 技术实现防御模型升级，进而提升数据挖掘与安全分析的能力。

对于极验来说，GCN 是一套行之有效且拥有巨大潜力的建模方法，只要场景

中存在数据和标签，就可以尝试用 GCN 学习其中的模式。以极验深知这款产品为例，GCN 主要用来提升其人机判别能力。极验拥有大量的行为数据户，包括用户操作行为的序列数据、IP 及设备的关联数据等，这些数据涵盖了几乎所有行业的用户，数据之间存在或明或暗的关联性。借助 GCN，研发团队可以对这些海量数据进行集成建模，达到更好的模式识别效果。从模型的 AOC 指标和稳定性指标这两个考察维度来看，相比原来的模型，引入 GCN 之后的算法模型在这两个维度上分别有 1.2-1.5 倍的提升，不论从最终效果还是模型鲁棒性来说都十分有益。据介绍，即使将网络中的节点属性特征全部抹去，只提取节点之间的关联特征，当前的模型依然能够得到非常好的效果。这些都离不开 GCN 模型强大的关系建模能力和极验海量数据的支持。

GCN 落地实践经验

GCN 落地应用难在哪？

从考虑引入 GCN 到真正落地，极验用了一年多的时间，到现在为止，相关计算模型已经经过了两次大版本的迭代。对于 GCN 落地实际业务过程中遇到的问题，刘忠雨总结为三个方面：

1. 如何解决大规模图的训练问题。比如 GraphSage 通过图采样操作把大图切分成很多小图再进行训练，并且这个操作一定是要有 online 的效率保障，这样才能使 GPU 的计算资源不会被 CPU 上的数据采样操作所拖累，因此如何保证图采样操作的高效性是至关重要的。而现在的图数据库对这类图采样算子并没有原生的支持。
2. GCN 模型怎么做实时预测。我们的业务场景不同于推荐营销，数据的表征可以定时更新，然后把数据推到业务线去做实时的推荐；在风控领域，一般我们需要不断地去判断每一次新的业务事件是否带有风险，这需要我们把模型推到业务线上去做实时的预测。图神经网络是一个协同式的训练与预测过程，需要把数据累计到一定量之后再进行预测，这是 GCN 的优势所在，但同时也大大提升了模型实时预测的难度。如何在线上预测输出接下来的数据的表现，需要对算法模型本身做一些改造工作，并且配套大量工程化的工作。

3. 图数据建模 Pipleline 的各个环节设计。其中最为核心的是图数据的预处理和图模式（Schema）的设计。图模式反映的是对于业务问题的理解，需要基于数据本身的路径和上层业务建模的特性来设计图的模式。刘忠雨认为，图数据预处理和图模式设计比后面的算法模型更重要，做好前两项对于算法模型的效果提升大有裨益。

因此，从算法落地层面考虑，极验花费了很大精力解决包括图采样效率、实时数据预测、图数据建模流水线、图数据预处理等问题。

针对大规模图的分布式训练问题，极验重点完成了图采样操作所需要的存储层重构工作，数据库选用了非常成熟的 HBase 将数据组织管理起来，同时在索引上做了大量的开发，尽量保证图上相近的数据在一个数据分片上，同时优化了数据的遍历性能。这一套配合下来，基本满足了团队对大规模图数据的分布式训练要求。

同时团队也在稳步推进基于 Graph Partitioning 的数据存储方案，这样不仅能更进一步地提高模型的训练效率，同时也能将分布式训练拓展到超大规模（如百亿边量级以上）的图数据上。这一套方案的开源也在计划中，等方案打磨得更加稳定成熟后，极验希望提供给工业界一套扎实可用的图数据分布式训练方案。

另外，极验针对数据建模的整体流程，如数据导入、数据预处理、图模式设计、模型配置与训练、模型评估等环节，开发了一套 Web 平台来辅助高效便捷的建模工作。磨刀不误砍柴工，事实证明，这是一本万利的工程性投资，加快了模型调研的同时，也大大降低了模型的后期维护成本。

图计算框架选择

图神经网络当前已经有不少专门的框架，比如亚马逊的 DGL、阿里的 Euler、基于 PyTorch 的 PyG，同时主流的深度学习框架也会支持图神经网络和图计算。但在 2017 年极验开始调研图神经网络的时候，还没有这么多开源框架可供选择，为了解决前面提到的三个落地问题，极验自己开发了一个内部的图计算引擎，做了一些图数据存储底层组件的定制化开发和优化工作，上层再与 TensorFlow 做整合。

在刘忠雨看来，**图神经网络计算框架是未来深度学习计算框架一个新的主流**，因此今年以来各类框架层出不穷。对于目前市面上主流的几个图神经网络计算框架，

刘忠雨认为各个框架都有自己的鲜明的特点。比如，基于 PyTorch 的 PyG 非常适合用于学术研究，其内部已经汇聚了很多任务模型的实现。如果图数据规模比较小，用户希望做一些学术研究或开发新算法，DGL 也非常合适，它提供了一个非常通用的图神经网络的编程模式，并且最新的 0.3 版本也对性能做了比较大的优化。刘忠雨认为 DGL 未来的开发方向会朝着学术和工业两条线并行发展。阿里开源的 Euler 定位更加精准，就是如何在一个工业级别的大图上高效地训练算法。

算法模型选择

当前图神经网络已经有非常多的算法模型，包括大量变体，比较出名的有 GraphSage、Graph Attention Network、RGCNN，在这之上还有一个比较通用的 GNN 编程模式叫做 message passing 机制，稍微改造一下就可以适配属性图的学习，而属性图可以代表最广泛的图数据格式。这些功能基本上已经能够覆盖极验实际业务中非常多的场景。因此，对于模型选择，刘忠雨认为一个 Message Passing 方法就够了，其他变体算法其实没有什么本质上的区别。真正在实际业务落地中，重心应该放在数据本身和业务特点上，根据特定的业务数据和相关任务进行落地模型的考量。变体虽多，但在实际业务中使用的可能性或者性价比比较低，比如 Graph Attention Network 对于显卡计算负载过大，相比较而言，给业务带来的收益却并不明显。

模型更新问题

图蕴含的数据量往往特别大，全量更新的成本比较高，但是新产生的数据（如用户行为数据）时效性更强，价值也更大，那么如何做好 GNN 模型更新的工作？刘忠雨认为可以分三个层面来做。

首先可以直接用 GNN 模型去预测，现在的图模型大多是支持 inductive learning 的，基本可以直接对新数据进行预测，这本身就是在衡量模型算法的泛化能力。其次，如果数据变化比较快或者新出现的数据比较多，模型需要快速跟进并适配这些数据，就可以采用一些小技巧。比如将变化比较大的节点以及部分采样出来的就数据结合到一起，对这部分数据进行模型的 FineTune，从而做到模型的快速更新。第三步，还是需要选择一个适应业务指标需求建模的窗口数据来进行模型的训练，从而得到更好的校正效果。总体来说，极验的思路就是上述三个步骤的循环，

优先保证模型本身的泛化性，再通过对新数据的微调来补充模型的快速响应能力。

GCN 未来展望

图神经网络在工业界尚未成为现象级技术

刘忠雨告诉 InfoQ，目前图神经网络在工业界并没有成为现象级的技术，但在大公司已有尝试，比如阿里将图神经网络用在了搜索排序和欺诈用户检测，国外的图片社交网站 Pinterest 发布了第一篇大规模图神经网络落地应用的论文等。虽然整体来看图神经网络的落地应用尚处于早期，还没有大范围推广开，但刘忠雨非常看好图神经网络的落地前景。图神经网络天然适用于大规模业务数据挖掘的需求，而前一波深度学习浪潮积累下来的软硬件设施也为图神经网络的落地打下了基础。

首先，很多公司的业务数据本身就存在大量的关系模式，关系建模可以帮助解决很多业务问题，典型场景比如反欺诈、营销推荐等，这给 GNN 的发展奠定了很好的数据与业务的基础。其次，很多企业已经在这波 AI 浪潮中搭建了自己的建模团队和机器集群，GNN 和 CNN、RNN 是一脉相承的，都是深度学习在新领域的推进，这又给 GNN 的发展奠定了很好的技术基础。基于这两点，刘忠雨认为 GNN 后续的研发和应用都会十分迅速，**2 年之内应该就能真正落地**。

图神经网络的典型应用场景主要包括以下 5 类：

1. 反欺诈。通过 GNN 对数据进行更深层的挖掘，识别恶意用户的模式，从而进行风险控制。
2. 营销推荐。原来我们可能只是关注用户与商品之间的交互矩阵，但其实用户可能与商品还有更多层次的交互，比如用户与商家、商品跟商品之间的交互，而且用户与商品本身也有很多属性信息。这些信息以及前面提到的交互性的关联信息怎么做端到端的学习？图神经网络就提供了对应的解决方案，这可能是未来推荐系统非常普适的一个方向。
3. 推理。推理现在可以说是学术界研究 GNN 非常火的一个方向，它指的是把信息或概念融合之后得到推断。很多信息和概念之间是存在一定关联的，因此可以协同考虑，这一过程与 GNN 的计算模式不谋而合。常见的应用包括视觉问答、视觉推理、语义图等。

4. 3D 视觉。前几年业界主要使用 CNN 来解决 2D 图形的分割检测等问题，而现在 CVPR 接收论文有相当大的一部分都在做 3D 视觉，比如 3D 点云识别分割等，其中有大量工作都是基于图神经网络的思想。
5. 学术场景，如生物的蛋白质活性检测、RNA 分类、物理领域的研究等。

图神经网络落地的瓶颈

刘忠雨告诉 InfoQ，图神经网络本身是一个产学研结合非常紧密的方向，但当前还存在几大瓶颈问题待解决。

首先是算法的可扩展性。虽然以采样操作为核心的 GraphSage 算法可以适应数十亿边规模的图，但它不是最优的解决方案。未来业界还需要考虑更高效的针对图建模的通用优化算法，这对于 GCN 在工业界全面落地能够提供非常大的帮助。目前来看，学术界已经逐渐有一些关注图学习优化的论文出来，这在未来会成为一个比较重要的学术研究方向。

另一个瓶颈是 GCN 的过平滑问题。目前图神经网络不能设计成特别深的层，否则节点的表达就趋于一致了，导致这个问题的一个主要原因就是过平滑，这个问题现在也已经有了一些解决方向。（注：参考论文 https://arxiv.org/abs/1806.03536，https://arxiv.org/abs/1810.05997）

未来极验在 GCN 的工作重点会继续放在图建模流水线的优化上，以加快业务产出的效率；另外也会把部分精力放在更高效的图的优化方法上，包括如何更快速地进行图算法模型的迭代等。

采访嘉宾介绍

刘忠雨，毕业于华中科技大学，极验算法负责人。负责极验各项目的算法研究以及技术路线制定，在图神经网络方面有比较深入的研究。

谷歌技术面试终极通关指南

作者 | Vincent Russo

译者 | 王强

编辑 | Natalie

AI 前线导读：通过谷歌面试是所有工程师的梦想。技术面试谷歌招聘流程必不可少的一环，它是对应聘者技术能力的终极测试，以评价你是否具备开发最优秀软件所需的编程与算法能力。谷歌面试是评价技术能力的关键衡量标准，可以说是决定是否聘用候选人的最重要决定因素。

想要顺利通过谷歌面试的话，你需要为面试中的重点环节提前做好准备。这篇文章就会教你相应的策略。此外，有大批企业都在模仿谷歌的面试风格，所以这篇文章也能帮你应付其他大型科技公司的面试。

本文中你将学到以下内容：

1. 什么是谷歌面试？
2. 谷歌面试与其他公司有何不同？

3. 如何准备谷歌电话面试和谷歌现场（Onsite）面试
4. 最常见的谷歌面试问题
5. 其它与谷歌面试相关的资源

什么是谷歌面试？

谷歌面试与常见的面试区别很大。如果之前你经历过编程面试的话还好，否则谷歌面试和你习惯的那些流程是完全不同的。

你在谷歌面试中可能会遇到三大类型的问题：

1. 编程面试问题
   - 问题：一项依赖数据结构和算法知识的编程问题。
   - 要求：在面试的时间限制内为问题提供有效且优化的解决方案。
2. 系统设计问题
   - 问题：涉及复杂系统设计的高层次宽泛问题。例如，面试官可能会让你设计 Gmail。
   - 要求：能够与面试官一起确定系统关键组件的内容，并设计一个可扩展的解决方案。
3. 一般分析问题
   - 问题：一个数学、设计或基于观点的问题，面试官想要考察你的思维逻辑，考察你作为员工的行事风格。
   - 要求：提供许多不同的解决方案，并能够讲出每个解决方案的各种利弊。这个过程犹如试金石，会让面试官了解你在团队中的作用和角色。

谷歌面试的大部分内容都是编写代码，这也是本文重点关注的内容。关于应对系统设计面试的策略，请参阅这篇文章。

谷歌面试与其他公司有何不同？

谷歌很乐于分享他们的招聘流程，还专门写了一整页介绍。

面试流程第一步是电话面试，如果通过就会进行一系列的现场面试。谷歌电面和现场面试在技术面试中是很常见的策略，但他们的流程有一些独一无二的细节特征：

谷歌电话面试

电话面试阶段会有至少一位谷歌员工与你沟通，给你发送编程试题。你会同面试官共享一份谷歌文档，并用它来编写面试官给出试题的代码。

高级技巧：在谷歌文档中写代码是很难受的，但如果你提前设置好自己的的偏好，那这个过程就能轻松很多很多了。这里是具体的做法。

这次面试将重点考察你在没有 IDE（集成开发环境）的情况下编写代码的能力。面试官提出的问题一般来说会有一个暴力解决方案，然后可以逐渐改进优化解决思路。

电话面试大约需要 30-45 分钟。如果你表现良好，谷歌招聘人员准备对你继续下一阶段考察的话，他们会再联络你并告诉你之后的流程。

如果他们要求进行第二次电话面试也不要失望。可能只是因为他们觉得一次电面得到的信息还不够下结论，这也很正常，不要因此在之后的面试中有任何心理负担。

谷歌现场面试

谷歌现场面试阶段，应聘者会同多名谷歌员工进行交流。这一阶段一般会有四到六次单独的访谈，包括一次“午餐面谈”。

一般来说他们问你的问题以编程为主，可能还有一两个涉及系统设计的问题。

你的工作经验越丰富，越可能被问到系统设计和特定领域的问题。谷歌很少向

经验不足 5 年的工程师询问任何系统设计问题。

每位面试官都会记录一对一面谈中你的表现并汇报上去。几位面试官的报告是各自独立的，以避免从众效应和偏见。所以就算你觉得其中一次面试时自己表现不好，这次的记录也不会影响下次面试。

面试结束后

面试结束后谷歌会收集面试官们的报告并汇总。谷歌会对你的编码经验、分析能力等一系列表现分类打出 1-4 分，然后将汇总报告发给招聘委员会，以做出最终决定。一切顺利的话，你就可以开始同公司商讨薪水等事项了。

如果你没通过，也可以在半年到一年后重新申请。设定这个时限是考虑到你可以用这段时间重新学习，提升你的技能和经验，这样下次你的机会就会更多一些了。

尽管很多谷歌员工都曾在现场面试阶段失利，但投入了大量时间和精力之后还没能成功真的很打击人。最重要的是，要等待半年再次申请的规定可能会让人迷失方向，甚至干脆把最开始的雄心和计划抛在脑后了。

谋事在人，成事在天，但起码你可以做好准备，把握好自己可以做到的事情。接下来，我就会教你该如何为谷歌面试做准备。

准备谷歌面试

怎样为面试做好准备，争取一次就通过？像 HackerRank、LeetCode、ProjectEuler、TopCoder 这类网站都提供了技术面试的题库以便练习。你当然可以花费大量时间把 LeetCode 上所有习题都过一遍，但这真的值得吗？有没有更有效率的路子可走？

人的时间和精力都是有限的，因此我们要尽量高效地使用这两种资源。你需要专注、持续并有针对性的训练，这是通向谷歌面试成功之路的必要步骤。

模拟真实场景做训练

贴近现实的练习才有效率。

面试时，你会希望自己全神贯注，将大部分精力都投入到面试官提出的问题上。你不应该受无关因素，比如说不习惯在非 IDE 环境中写代码这种事情干扰。

以下列出了在制定练习方案时应注意的因素。

时间限制

电话面试时间为 30-45 分钟，每次一对一的现场面试大约持续 45 分钟到一小时。每次面试中你至少会被问到一个问题，多数情况下会是一个编程问题。

为了在这种时间限制下高效练习，一种方法是选一个特定问题入手。

一开始你可以从本文的附注资源部分下的书籍或视频中选择一个问题。然后开始计时，并尝试在不用 IDE 的情况下解决它。

这样针对几个不同类别的编程问题练习过后，你就差不多知道自己在时限内处理问题有哪些不足了。你的最终目标是在 30-45 分钟的时限内，针对各种难度的问题做出最佳解决方案。如果你正在努力实现这一目标，那么就应该为此做针对性的训练。

白板

无论是谷歌电面还是现场面试中，你都要在没有 IDE 的情况下写出语法正确的代码来。实话说这真的很难。

电话面试时代码要写在谷歌文档上；现场面试时你要在白板上手写代码。如果你从未练习过，那么就很难在白板上手写出正确的代码了。因此你要在练习时就习惯时间限制、习惯摆脱对 IDE 的依赖，这是非常重要的。

每次练习后可以把代码完整抄到编辑器中运行一下，你可能会惊讶地发现自己在白板上手写的代码竟然会有这么多错误。为了避免在面试中出这种问题，你应该练习在纸上手写代码。

你还必须非常熟悉面试中使用的语言。虽然你可以请面试官帮忙提供库功能来完成某项任务，但这会浪费时间，并且反映出你准备不足、缺乏领域相关知识的缺

陷。

压力

作为应聘者，面试时你是被关注的焦点，大家的目光都聚焦在你身上。

面试时你要在限定时间内以不常见的方法解决一项颇具挑战性的难题，这时还要有人死死盯着你，这是非常让人不舒服的事情，很容易影响人们的表现。

对抗焦虑的一个好办法是训练自己在时间限制内用白板手写答案，同时找一位伙伴在旁边盯着你看。但你可能准备一个人去应聘谷歌，所以这种伙伴也不好找，怎么办？

这种情况下，Pramp 就可以帮助你解决问题。Pramp 会随机匹配两位程序员，你问同伴一个问题并扮演半小时的面试官角色，然后角色反转，你再扮演半小时的面试者角色。

高级技巧：你还可以通过 Pramp 了解成为面试官的感受，这是一个很好的角度。

测试和罕见情况

当你针对给定问题开发解决方案时，你需要考虑到可能由意外输入导致的罕见情况。这是一个很好的习惯，这样你的面试官就能知道你会考虑代码会出现哪些问题。

此外，你还要为这些罕见状况提出解决方案。

如果你发现自己能在设定的时间限制内解决指定的编程问题，那么可以编写一些单元测试，并好好考虑开发初始解决方案时可能遗漏的罕见状况。

养成写代码时考虑到将要或者应该做的测试的习惯，这样你就能预测面试官会提出哪些问题。经过充分练习后你就会形成条件反射，面试官提问时你就不会猝不及防了。

关键在于训练素材

在练习时选择覆盖所有种类题型的题库是很好的，但关键在于专注正确的训练素材。

虽然你可以在 LeetCode 上随机练题，但这样效率很低，多练一道题也不会大幅提升你通过谷歌面试的机会。

相反，何不专门练习过去谷歌面试中出现过的题库呢？这样会让你的时间和精力更加集中，提升效率。

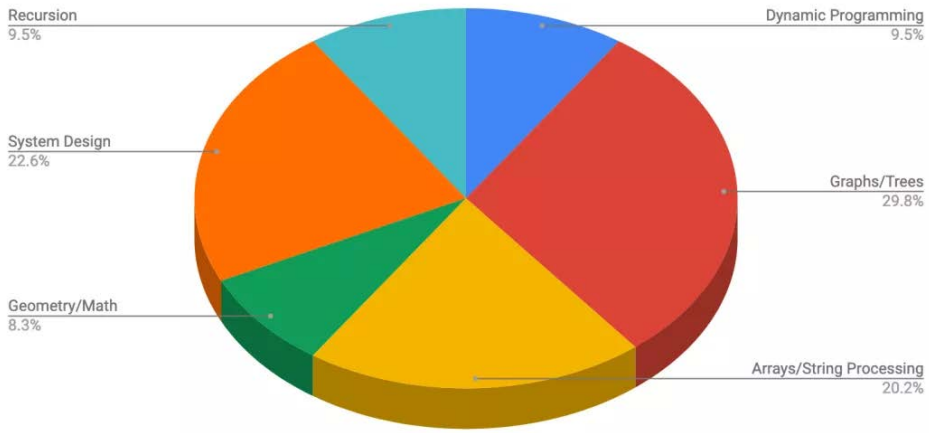
下一节中，我们将介绍谷歌喜欢考察的具体问题和类型。

最常见的谷歌面试问题类型

我们在 Glassdoor 网站中收集了一些谷歌面试中出现的问题类型。

Glassdoor 有一个谷歌专页，里面都是面试过谷歌的应聘者写的经历。他们一般会提到面试中被问到哪些问题，或者起码提及问题的类型。

我们分析了 Glassdoor 中的几百条面试经历，总结出了谷歌面试中最常问到的问题类型分布。



不算“系统设计”，只看“编程”大类，下面排行前三的分类分别是：

1. 图 和 树
2. 数组和字符串处理
3. 递归 和 动态规划

高级技巧：谷歌通常只向至少有 3 到 5 年软件工程经验的应聘者提出系统设计类问题。

谷歌看起来很喜欢问这几类问题，所以应该把大部分时间和精力投入到与它们相关的练习上。接下来我们来看如何针对这几大类问题做针对训练。

图和树

注意：这部分内容应该配合这份关于二叉树的 Youtube 视频列表学习。我们在本节中介绍的二叉树代码的完整实现都放到了 GitHub 上。

树和图是谷歌面试问题中经常涉及到的数据结构类型。你应该熟练设计、引导和操控这些结构来解决问题，这样就能在面试中得到高分。

为了检测自己对这些结构的了解程度，首先你可以自己编写一种树或图的实现。

比如说我们想要研究二叉树。了解二叉树概念的话，你肯定知道它是由节点对象组成的。

每个节点都有一些属性，除了指向左右子节点（如果有的话）的左右指针外，还包括存储在该节点中内容的特定数据字段。在 Python 中我们可以定义一个 Node 类，如下所示：

```
class Node(object):
def __init__(self, value):
self.value = value
self.left = None
self.right = None
```

我们还可以创建一个将由 Node 对象构造的 BinaryTree 组合类

```
class BinaryTree(object):
def __init__(self, root):
self.root = Node(root)
```

有了这个基本结构后，我们现在可以构造一个简单的 BinaryTree 对象：

```
# Set up binary tree:
tree = BinaryTree(1)
tree.root.left = Node(2)
tree.root.right = Node(3)
```

这个二叉树的根节点值为 1，左子节点值为 2，右子节点值为 3。

这个视频可以帮助你复习二叉树知识。

现在你对二叉树对象有了基本概念，知道了想要用好它，很重要的一点就是遍历它的结构。

接下来复习二叉树遍历算法，分为前序、中序和后序三类。

看完上面的内容后我们想要测试一下自己对二叉树数据结构的理解有多深刻。我建议尝试去解决一些二叉树问题来测试这方面的能力。

假设你已经复习到这一步了，那么你可以更上一个台阶，尝试实现一个水平遍历，也就是从上到下，从左到右遍历二叉树的全部节点。

做这个测试时试着给自己创造前文提到的压力环境，也就是给定时间、找伙伴模拟面试官，并在纸上或白板上手写代码。

之后你可以试着总结自己对二叉树结构的理解，并考察自己利用这些知识解决其它问题的能力，例如计算二叉树的大小和计算二叉树的高度。

通过这种渐进学习的过程，你就能逐步构建或增强对特定数据结构的知识框架。虽说并不是所有数据结构都需要这么复杂的学习过程，但是采用这种渐进深入的方法可以对特定数据结构建立非常深刻的理解，帮助你更好地利用它解决相关问题。

数组和字符串

数组和字符串算一类问题，因为字符串处理一般可以看作是执行某种类型的数组操作。

在谷歌面试中，你将遇到的几乎所有问题，其基础元素都包括数组数据结构（在 Python 这类语言中是列表）。

你应该深入理解数组结构的基础知识，理解从数组中存储、检索等操作的复杂性，才能快速将用这些知识来解决现实问题。

我们先从 50 个面试问题指南中挑一个问题来分析。来看第四道题，“查重”：

给定一个整数数组，其中每个值 1 <= x <= len(array)，编写一个函数来查找数组中的所有重复项。

例如：

```
dups([1, 2, 3]) = []
dups([1, 2, 2]) = [2]
dups([3, 3, 3]) = [3]
dups([2, 1, 2, 1]) = [1, 2]
```

欲了解本问题的细节点击此处。

如前所述，先从问题的暴力解决方案入手是不错的思路。虽说这种方案不是最好的，但起码能解决问题。当你盯着白板手足无措的时候，想出这样一个方案总比发呆要强。

那么这里该用什么暴力方案呢？ 想一下，你可以一次处理一个元素，同时检查剩下的元素，看看数组中的其它元素是否等于你正在处理的元素。

下面的代码就是一种思路：

```
def dups(A):
dups = {}
for i in range(len(A)):
```

```
num = A[i]
for j in range(i+1, len(A)):
if num == A[j]:
dups[num] = num
return list(dups.keys())
```

这段代码中，我们让外部循环遍历数组，再让内部循环在我们遍历数组时遍历剩余元素。这就会生成一个 O(n^2) 算法。

方法很笨，但起码开了个头。下面我们可以利用 Python 内置的 set() 函数来改进这个方案。其它语言的算法也是类似的。

这里的改进思路是，我们可以创建一个 set 对象并遍历输入列表。如果我们迭代的元素不在集合中，则将其添加进来。否则就继续处理列表。

代码如下：

```
def dups(A):
s = set()
for i in A:
if i not in s:
s.add(i)
return list(s)
```

这样我们的 O(n^2) 算法就改进成了 O(n) 算法。现在它还需要 O(n) 空间来在集合中存储元素。那么还能继续改进吗？

接下来我们可以改进空间需求，让它变成恒定值。

如果我们对输入数组排序，我们就可以遍历数组并检查我们正在处理的元素与数组中的下一个元素是否不同。如果两个元素是一样的就代表有重复项，我们就能对其做处理了。反之，我们就继续处理数组。

代码如下：

```
def dups(A):
```

```
A = sorted(A)
s = set()
for i in range(len(A)-1):
if A[i] == A[i+1]:
s.add(A[i])
return list(s)
```

这样我们就把空间需求优化到了 O(1)，但因为排序带来了 O(n log n) 的时间复杂度，所以花费的时间就会增加了。

到这一步我们找出了几个方法，它们在速度和存储需求方面各有优势。现在我们应该再看一下这个问题，看看它有什么特点可以用来做方案优化。

再看一遍后我们注意到了前提条件，那就是每个值都会 1 <= x <= len(array)。这里的“技巧”在于，既然已经有了一个不用额外数据结构的定值，那么能不能在空间复杂度不变的同时优化方案呢。

鉴于数组中所有数字的值域都介于 1 到数组长度之间，我们就可以把它编码到已有的数组数据结构中。这样，我们就能利用上给定的输入中已经分配给我们的空间了。

来看一个具体的输入示例，看看这个新方法的实现原理。假设输入的是列表 [2, 1, 2, 1]。重申一下，我们知道数组中的每个值都介于 1 和数组的总长度之间。

为了简化问题，我们从列表的值中减去一，这样条件就变成了每个值都是 0 <= x <= len(array) - 1。

另一个重要发现是，根据前提我们知道数组中的每个值都是正整数。

当我们遍历这个数组时，每个数字如果见过的话就把值对应的索引值正负号调转。这样如果发现了一个带有负号的值，就知道之前遇到过它了，并将该值添加到我们的重复集中。

具体来看 [2, 1, 2, 1] 这个例子。我们遍历这个列表的所有元素，遇到的第一个

值是 2。我们现在想要记录下我们在数组中遇到 2 这件事，所以我们需要从值中减去一来计算索引，亦即：

index = 2 - 1 = 1

然后我们将对应索引 1 的索引值取负值，列表变为：

[2, -1, 2, 1]

继续处理列表，我们遇到了 -1，这里取绝对值来获取它的原始值并像之前一样计算索引：

index = 1 - 1 = 0

现在我们转到数组中的索引 0 并取其负值：

[-2, -1, 2, 1]

这个列表告诉我们，我们已经遇到了一个 1，因为我们取了索引 0 的负值；我们还遇到了一个 2，因为我们取了索引 1 的负值。继续这个算法，我们遇到了第二个 2，索引计算为

index = 2 - 1 = 1

现在检查数组中索引 1 处的值，但该值为负，表示我们已经在数组中遇到过这个值了。所以我们将此重复值添加到输出集中。

按照这套方法，我们的算法需要 O(n) 的时间并使用 O(1) 的空间。最终代码如下：

```
def dups(A):
result_set = set()
for i in range(len(A)):
# Translate the value into an index (1 <= x <= len(A))
index = abs(A[i] - 1)
# If the value at that index is negative, then we've already seen
# that value so it's a duplicate. Otherwise, negate the value at
```

```
# that index so we know we've seen it.
if A[index] < 0:
result_set.add(abs(A[i]))
else:
A[index] = -A[index]
# Return the array to its original state.
for i in range(len(A)):
A[i] = abs(A[i])
return list(result_set)
```

这篇文章有关于常见字符串和数组模式的更多内容。

递归和动态编程

递归 和 动态编程 是相辅相成的。用动态编程解决问题时通常要先找到问题的递归解决方案，然后找到存储和参考之前计算结果的方法来避免之后不必要的计算。

要掌握这这两块内容肯定要对递归有充分的了解。仅了解递归的概念是不够的，关键在于用简洁的方式将递归思维快速应用到解决方案上。

显然，前面提到的常见问题类型，包括树 / 图和数组 / 字符串问题通常也会用到递归。

例如，在字符串中查找大写字母、计算字符串长度或计算字符串中辅音数量等问题都可以归类为“字符串”的类型，但它们也都有非常简洁的递归方案。

解决需要递归的问题，关键在于制定识别、分析和解决递归问题的策略。

我们还探讨了如何使用 FAST 方法解决动态编程问题。

其它资源

应对谷歌面试所涉及的资源和材料是很多的，本文不可能一一详解。

决定你在谷歌面试中能否成功的关键因素在于你的编程熟练度，为此我们会介

绍一些我们最喜欢的书籍和视频资源。

图书：

破解编程面试（https://www.byte-by-byte.com/aff/crackingthecodinginterview）

作为练习材料，Gayle Laakmann McDowell 的破解编程面试，（Cracking the Coding Interview，CTCI）是最受欢迎的资源之一。除了提供一系列实践问题外，本书的引言还提供了有关谷歌如何做招聘的具体情报。

这部分内容只有简短的几页，但作者 Gayle 之前曾参与谷歌的面试流程，所以她的建议很有价值。这段视频是对 CTCI 的一个详解。

编程面试的要素（https://www.byte-by-byte.com/aff/elementsofprogramminginterviews）

如果你想开个书单，这个视频提到了他为技术面试准备工作推荐的 5 本书，其中就有 CTCI。

我最喜欢的一本是 Adnan Aziz、Tsung-Hsien Lee 和 Amit Prakash 编写的 Python 编程面试要素（EPI）。

其中一位作者 Tsung-Hsien 是谷歌的现任员工，本书中提到的许多问题与 Glassdoor 数据总结出来的问题很接近，这可能并非巧合。CTCI 的习题太常见了，所以面试官不太喜欢问这类问题。

EPI 的解决方案非常简洁，写得很好，会逐步引导你的思维。每次看一个解决方案时，你可以看到一半开始自己动手，无需看完全文。

这种安排就很棒了，你自己动手时遇到问题了，还可以模拟从面试官获得提示的过程。虽然最好在没有任何明确提示的情况下独立解决问题，但这本书能有这个选项当然更好。

它的方案代码也很有 Python 风格。我发现就算我找出了正确的思路，再看一遍作者解决问题的简洁思路也有助于训练编写格式优雅的 Python 代码。

如果你认为 Python 不是最强的面试语言，那么本书也有 Java 和 C++ 版本，可能更适合你。这些版本的写作风格和内容也同样出色。

Byte-Byte 的 50 道编程面试问题（https://www.byte-by-byte.com/50-questions/）

我们自己也制作了一组问题列表可以作为上述资源的良好补充。你可以在此下载免费指南。

如果你正在练习一个特定问题，并需要一些高级指导，那么这些补充材料非常有助于理解和解决问题。

视频资源：

根据你的时间表和学习习惯，观看视频内容可能是准备谷歌面试的首选方式。假设你的知识基础薄弱，那么可以看像 Coursera和 Udacity这样的网站提供的内容，这些网站专注于计算机科学、数据结构、算法等基础知识。

如果你在数据结构和算法方面有扎实的基础，那么观看谷歌面试中遇到的特定类型问题相关的视频内容做训练更好。

可以看看我们的 Youtube 频道。我们已经针对某些问题类型制作了视频列表，其中包括适合初学者的动态编程、图形、递归和数组教程。这些视频都放在了我们的编程面试问题播放列表里。

作者介绍

Vincent Russo是 Byte by Byte 的撰稿人，也是一名全职开发者。他还运营着 LucidProgramming这个 YouTube 频道，可以帮助开发者提升编程技能和个人价值。LucidProgramming 的内容以 Python 开发为主，其中涉及的主题包括数据结构、算法、Web 抓取、自然语言处理等等。

查看英文原文：

https://www.byte-by-byte.com/google-interview/

解读PingSage：图卷积神经网络在数十亿数据网络级别推荐系统的应用

论文作者 | Rex Ying, Ruining He 等

编译 | Maglish

编辑 | Vincent

AI 前线导读：深度神经网络在图结构数据方面的进步大大提高了推荐系统基准的最佳性能。然而，对于具有数十亿数据项和数亿用户网络级别的推荐任务来说，这些方法的实际应用和可扩展性仍然是一个挑战。Pinterest 的研究人员提出了一种新的图卷积网络 PinSage，能够生成包含图结构和节点特征信息的有效嵌入表示，并且设计了新的训练策略，以提高模型的鲁棒性和收敛性。作者在 Pinterest 网站上部署了 PinSage，并利用 75 亿个样本对其进行训练。与深度学习基线方法相比，PinSage 能够生成更高质量的推荐。对于推荐任务，PinSage 比最佳基线方法的点击率提高了 150%，MRR 提高了 60%。**这是迄今为止深度图嵌入表示的最大应用，为基于图卷积结构的新一代网络级别的推荐系统铺平了道路**。本文是 AI 前线第 80 篇论文导读，为你详细解读 PinSage 背后的技术细节。

介 绍

深度学习方法在推荐系统应用中有着越来越重要的应用，利用深度模型学习到的特征表示，可以补充，甚至取代传统的推荐算法。近年来，随着能够在图结构数据上进行学习的深度学习方法的出现，这一领域取得了重大进展，因为图结构数据一直是推荐应用的基础（例如，开发用户到项目的交互图以及社交图）。

其中最突出的是图卷积网络（GCN）这一深度学习体系结构的成功。GCN 背后的核心思想是学习如何利用神经网络迭代地聚合来自局部图邻域的特征信息（如图 1 所示）。一个“卷积”操作从一个节点的单跳图邻域转换并聚集特征信息，并且通过叠加多个这样的卷积操作，信息可以传播到图的远端。与基于内容的深度模型（如递归神经网络）不同的是，GCN 既利用了内容信息，也利用了图结构。基于 GCN 的方法在无数推荐系统的基准上建立了新的标准。然而，相对于基准任务的提升还需要进一步转化才能在现实生产环境中应用。

现实环境所面临的主要挑战是如何将基于 GCN 的节点嵌入表示的训练和推理阶段扩展到具有数十亿个节点和数百亿个边的图中。扩展 GCN 是很难的，因为在大数据环境下工作，会违反它们的设计中潜在的许多核心假设。例如，现有的基于 GCN 的推荐系统在训练期间都需要在全图拉普拉斯上运行，但当基础图具有数十亿个节点并且结构不断变化时，这种假设是不成立的。

我们提出了一个在 Pinterest 中开发并已应用于生产的高度可扩展的 GCN 框架，PinSage。它在具有 30 亿个节点以及 180 亿个边的图结构上运行，比经典的 GCN 应用中的图结构大了 10000 倍。PinSage 充分利用了以下几个关键的思想，大大提高了 GCN 的可扩展性：

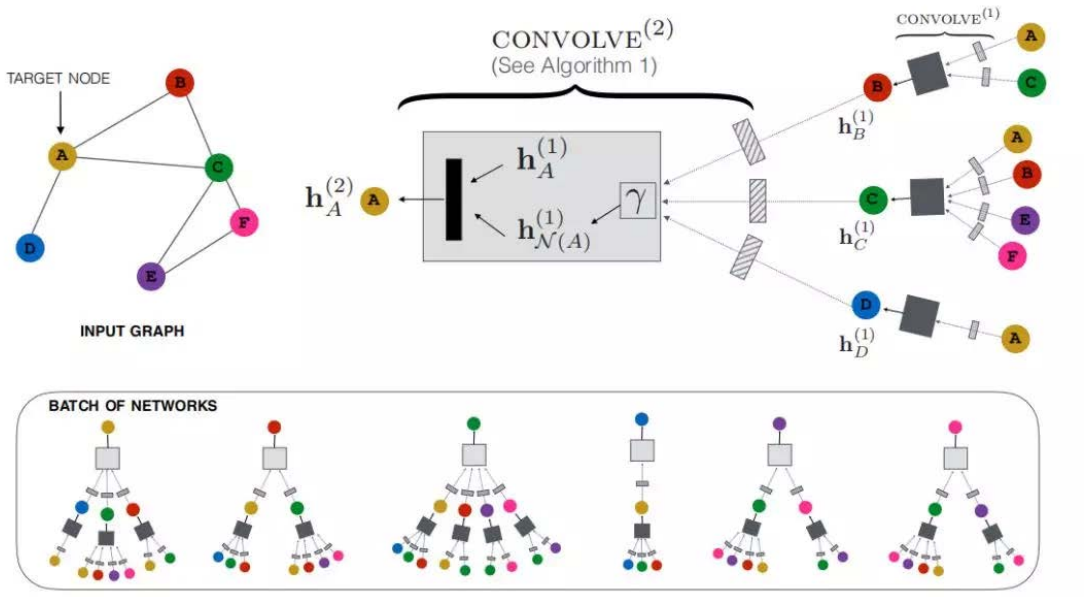


图 1：使用 2 层卷积的模型架构概览。左：一个小的输入图示例。右图：2 层神经网络，用上一层的表示计算 A 节点及其邻域 N(A)（节点 B、C、D）的嵌入表示 h(2)A。底部：计算输入图每个节点嵌入表示的神经网络。虽然每个节点的神经网络都不同，但它们共享相同的参数集（即卷积 (1) 和卷积 (2) 函数的参数）。具有相同阴影的框表示共享参数；γ 表示重要性池化函数；而长矩形框表示稠密连接的多层神经网络。

在线卷积：传统的 GCN 算法通过将特征矩阵乘以全图拉普拉斯的幂进行图卷积。相反，PinSage 算法通过对一个节点周围的邻域进行采样，并从该采样邻域动态构造一个计算图，来实现高效的局部卷积。这些动态构造的计算图（图 1）指定了如何围绕特定节点进行局部卷积，并降低了在训练期间对整个图进行操作的需要。

通过随机行走构造卷积：取节点的整个邻域进行卷积（图 1），会产生巨大的计算图，因此我们利用采样的方法。我们开发了一种利用短随机游走对计算图进行采样的技术。该方法的另一个好处是每个节点都有一个重要性评分，我们在池化 / 聚合步骤中可以利用该评分。

重要性池化：图卷积的一个核心组成部分是图中来自局部邻域的特征信息的聚合。我们引入了一种基于随机游走相似性度量的方法来衡量节点特征在这个聚合中的重要性，从而在离线评估度量中获得了 46% 的性能提升。

除了这些在可扩展性方面的改进之外，我们还引入了新的训练技巧和算法创新。

这些创新提高了 PinSage 学习的特征表示质量，从而显著提升了下游推荐系统任务的性能：

生产者 - 消费者 minibatch 构造：我们设计了一个生产者 - 消费者架构，用于构造 minibatch 运算，以确保在模型训练期间最大限度地利用 GPU。一个大内存、CPU 绑定的生产者有效地对节点网络邻域进行采样并获取定义局部卷积所需的特征，而一个 GPU 绑定的 TensorFlow 模型则消费这些预定义的计算图，实现高效地 SGD 计算。

有效的 MapReduce 推理：给定一个完全训练的 GCN 模型，我们设计了一个高效的 MapReduce 管道，可以将训练后的模型分布到数十亿个节点上，同时最小化重复计算。

课程训练：我们设计了一个课程训练方案，逐步提升训练样本的难度，从而获得了 12% 的性能提升。

我们在 Pinterest 的各项推荐任务中部署了 PinSage，Pinterest 是一个流行的内容发现和管理的应用程序，用户可以通过图钉（Pins）进行交互，Pins 是在线内容的可视书签（例如，他们想烹饪的菜谱，或者他们想购买的衣服）。Pinterest 是世界上最大的用户管理的图像库，有超过 20 亿个独特的图钉被收集到超过 10 亿块钉板上。

通过广泛的离线评价、受控用户研究和 A/B 测试，我们发现，PinSage 在项目 - 项目推荐任务（pin 相关的推荐）和家庭订阅推荐任务中都达到了最先进的性能。在离线排序指标中，我们比最佳表现基线提高了 40% 以上，在用户调查中，我们的推荐中有 60% 左右被首选，而 A/B 测试显示，用户参与度提高了 30% 至 100%。

据我们所知，这是有史以来最大的深度图嵌入应用，为基于图卷积结构的新一代推荐系统铺平了道路。

方 法

PinSage 中最重要的思想是局部图卷积。为了产生一个节点的嵌入表示，我们应用多个卷积模块，从一个节点的局部图邻域累积特征信息（视觉特征、文本特征）。每个模块都从一个小的图邻域中学习如何累积信息，通过堆叠多个这样的模块，我们的方法可以从局部网络拓扑中学习到有用的信息。更重要的是，这些局部卷积模块的参数在多个节点之间共享，使算法的参数复杂度独立于输入图的大小。

问题描述

Pinterest 是一个内容发现应用，用户可以通过图钉（Pins）来互动。我们的任务是为图钉生成高质量的内嵌表示。为了学习这些内嵌表示，我们将 Pinterest 环境建模为一张由两套不相关的节点组成的二分图，I（图钉）和 C（钉板）。

除了图结构，我们也假设图钉 u 与实值属性 xu 相关。这些属性指定了一个条目的元数据或内容信息。在 Pinterest 中，我们提取图钉的文本以及图像特征。我们的目标是利用这两个输入属性，以及二分图的结构，生成高质量的内嵌表示。这些内嵌表示随后被用于推荐系统，通过最近邻查找得到候选条目（例如给定一个图钉，找到相关的图钉），或者在机器学习系统中用于对候选项排序。

模型结构

我们利用局部卷积模块为节点产生嵌入表示。首先输入节点的特征，然后学习神经网络，将图的特征转换并累积，计算节点内嵌表示。

局部卷积

Algorithm 1: CONVOLVE

Input : Current embedding $\mathbf{z}_u$ for node u; set of neighbor embeddings $\{\mathbf{z}_v | v \in \mathcal{N}(u)\}$, set of neighbor weights $\boldsymbol{\alpha}$; symmetric vector function $\gamma(\cdot)$

Output: New embedding $\mathbf{z}_u^{\text{NEW}}$ for node u

$\mathbf{n}_u \leftarrow \gamma\left(\{\text{ReLU}\left(\mathbf{Q}\mathbf{h}_v + \mathbf{q}\right) \mid v \in \mathcal{N}(u)\}, \boldsymbol{\alpha}\right)$;

$\mathbf{z}_u^{\text{NEW}} \leftarrow \text{ReLU}\left(\mathbf{W} \cdot \text{CONCAT}(\mathbf{z}_u, \mathbf{n}_u) + \mathbf{w}\right)$;

$\mathbf{z}_u^{\text{NEW}} \leftarrow \mathbf{z}_u^{\text{NEW}} / \|\mathbf{z}_u^{\text{NEW}}\|_2$

算法 1 局部卷积操作

我们将 u 的邻域 v 的内嵌表示 zv 通过一个稠密神经网络进行转换，然后对得到的矢量集应用累积或池化函数（元素级别的平均或加权和，用 γ 表示）（步骤 1）。累积步骤得到了 u 的局部邻域 N(u) 的矢量表示 nu。然后我们将累积邻域矢量 nu 和 u 当前的表示 hu 相连接，并通过另一个稠密神经网络层进行转换（步骤 2）。步骤 3 中对 zu 进行归一化，能够稳定训练过程。算法的输出即为节点 u 的嵌入了自身信息和局部图邻域信息的特征表示 zu。

基于重要性的邻域

算法的一项重要创新是选择领域 N(u) 的方法。在 PinSage 中，我们定义了基于重要性的邻域，其中节点 u 的邻域定义为对 u 最具有影响力的 T 个节点。我们模拟随机游走，从节点 u 开始，计算随机游走访问节点的 L1 正则化访问次数。u 的邻域定义为正则化访问次数最高的前 T 个节点。这一定义让算法 1 在累积邻域的矢量表示时考虑其重要性。因此，算法 1 中的 γ 函数为加权平均，权重即为节点的 L1 正则化访问次数。我们将这一方法命名为重要性池化。

堆叠卷积

每次应用卷积运算（算法 1），我们都会得到一个新的节点表示。我们可以将多个这样的卷积叠加在一起，以便获得围绕节点 u 的局部图结构的更多信息。我们使用多个卷积层，在这里，第 k 层卷积的输入取决于第 k-1 层输出的表示，而初始（即“第 0 层”）表示等于输入的节点特征。算法 1 中的模型参数（Q，q，W 和 w）对于所有节点共享，但是不同层之间不共享。

Algorithm 2: MINIBATCH

Input : Set of nodes $\mathcal{M} \subset \mathcal{V}$; depth parameter K;
neighborhood function $\mathcal{N} : \mathcal{V} \rightarrow 2^{\mathcal{V}}$
Output: Embeddings $\mathbf{z}_u, \forall u \in \mathcal{M}$

/* Sampling neighborhoods of minibatch nodes. */
$\mathcal{S}^{(K)} \leftarrow \mathcal{M}$;
for $k = K, \ldots, 1$ **do**
  $\mathcal{S}^{(k-1)} \leftarrow \mathcal{S}^{(k)}$;
  for $u \in S^{(k)}$ **do**
    $\mathcal{S}^{(k-1)} \leftarrow \mathcal{S}^{(k-1)} \cup \mathcal{N}(u)$;
  end
end
/* Generating embeddings */
$\mathbf{h}_u^{(0)} \leftarrow \mathbf{x}_u, \forall u \in \mathcal{S}^{(0)}$;
for $k = 1, \ldots, K$ **do**
  for $u \in \mathcal{S}^{(k)}$ **do**
    $\mathcal{H} \leftarrow \left\{\mathbf{h}_v^{(k-1)}, \forall v \in \mathcal{N}(u)\right\}$;
    $\mathbf{h}_u^{(k)} \leftarrow \text{CONVOLVE}^{(k)}\left(\mathbf{h}_u^{(k-1)}, \mathcal{H}\right)$
  end
end
for $u \in \mathcal{M}$ **do**
  $\mathbf{z}_u \leftarrow \mathbf{G}_2 \cdot \text{ReLU}\left(\mathbf{G}_1 \mathbf{h}_u^{(K)} + \mathbf{g}\right)$
end

算法 2 详细描述了堆叠卷积如何为一个小批次的节点集 M 生成嵌入表示。我们首先计算每个节点的邻域，然后应用 K 次卷积迭代生成目标节点的第 K 层表示。最后卷积层的输出通过一个全连接网络生成最终的嵌入表示 zu。

损失函数

我们使用基于最大边界的损失函数。基本思想是我们想最大化正样本的内积，即查询条目和对应的相关条目的内嵌表示。同时我们想确保负样本（查询条目和无关条目的内嵌表示）的内积比正样本的内积小，并且小的程度超过一个提前定义的边界。因此一对节点内嵌表示 (zq, zi) 的损失函数定义如下：

$$J_{\mathcal{G}}(\mathbf{z}_q \mathbf{z}_i) = \mathbb{E}_{n_k \sim P_n(q)} \max\{0, \mathbf{z}_q \cdot \mathbf{z}_{n_k} - \mathbf{z}_q \cdot \mathbf{z}_i + \Delta\},$$

较大批尺寸在多 GPU 上训练

为了在一台机器上充分利用多个 GPU 进行训练，我们以多塔的方式进行正向和反向传播。对于多个 GPU，我们首先将每个小批次（图 1 底部）划分为大小相等的部分。每个 GPU 接受小批次的一部分，并使用相同的参数集执行计算。反向传播之后，所有 GPU 上每个参数的梯度聚合在一起，然后执行一次同步 SGD。

生产者 - 消费者 minibatch 构造

在训练过程中，数十亿节点的邻接表和特征矩阵因其尺寸较大而被放置在 CPU 内存中。然而，在 PinSage 的卷积步骤中，每个 GPU 进程都需要访问邻域和邻域中节点的特征信息。从 GPU 访问 CPU 内存中的数据是不高效的。为了解决这个问题，我们使用重新索引技术创建子图 G’=(V’, E’)，其中包含节点及其邻域信息。在每个 minibatch 迭代开始时，G’的邻接表和小特征矩阵被送入 GPU，这样在卷积过程中不需要 GPU 和 CPU 之间的通信，大大提高了 GPU 的利用率。

采样负样本

为了提高大批次训练的效率，我们采样 500 个负样本，供每个 minibatch 中的所有训练样本共享。与单独为每个节点进行负采样相比，这大大节省了在每个训练步骤中需要计算的嵌入表示的数量。

在最简单的情况下，我们可以从整个样本集中均匀采样负样本，但这样的负样本构成对系统的约束太过“简单”，因为系统只需要确保正样本对（q, i）的内积比 q 和 500 个负样本的内积大即可，系统无法学习到足够细粒度的特征表示。为了解决这一问题，我们为每个正训练样本增加“更难”的负样本，即与查询条目 q 相关，但是不如正样本 i 相关程度高的负样本，我们称之为“难负样本”。它们是通过对图中条目根据个性化的 PageRank 分数进行排序而生成的。排位在 2000-5000 的条目被随机采样为“难负样本”。如图 2 所示，“难负样本”与其他随机的负样本相比，与查询样本更相似，因此对于模型来说挑战性也更强，使模型从更细的粒度上区分条目。

图 2：随机负样本和“难负样本”示意图。“难负样本”与其他随机负样本相比，与查询样本更相似，但是不如正样本相似。

在整个训练过程中使用“难负样本”将使训练所需的时间加倍。为了帮助收敛，我们制定了课程训练计划。在训练的第一个阶段，不使用“难负样本”，使算法在参数空间中快速找到损失相对较小的区域。然后，我们在随后的训练中添加“难负样本”，让模型学习如何区分高度相关的图钉和仅轻微相关的图钉。在训练的第 n 个阶段，我们将 n-1 个“难负样本”添加到每个样本的负样本集合中。

通过 MapReduce 的节点内嵌表示

由于模型训练过程节点的邻域会有重叠，因此许多节点在不同层被重复计算。为了保证推理阶段的效率，我们采用 MapReduce 方法实现无重复计算的模型推理。MapReduce 管道包括两个关键部分：

（1）一个 MapReduce 将所有的图钉映射到低维的隐空间，进行堆叠操作。

（2）另一个 MapReduce 将得到的图钉表示和他们所出现的钉板的标号相关联，然后通过池化其邻域的特征得到钉板的内嵌表示。

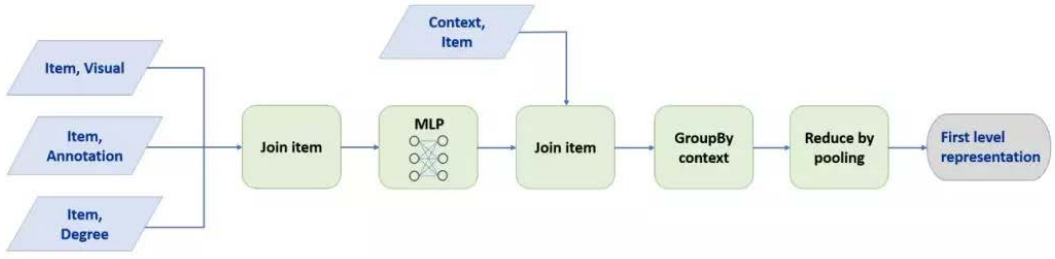


图 3：使用 MapReduce 计算第一层表示。第二层计算遵循相同的管道，只是输入是第一层表示，而不是原始条目特征。

高效最近邻查找

PinSage 产生的内嵌表示可以用于许多下游的推荐任务，我们可以通过在学习到的内嵌空间进行最近邻查找来完成推荐任务。即给定一个查询条目 q，我们可以通过查找查询条目 q 的内嵌表示的 K 个最近邻内嵌表示，来进行推荐。PinSage 模型是离线训练的，所有的节点内嵌表示都是通过 MapReduce 计算并保存在数据库中，高效的近邻查找操作使系统能够以在线方式提供推荐。

实　验

我们通过两个任务来评价 PinSage 生成的内嵌表示：推荐相关图钉和在用户的家庭或新闻订阅中推荐图钉。对于推荐相关图钉任务，我们选择查询图钉在内嵌空间的 K 个最近邻。我们使用离线排序衡量和受控用户研究来评估推荐相关图钉任务的表现。对于家庭订阅推荐任务，我们选择与用户最近钉在钉板上的条目在内嵌空间最接近的图钉。我们使用 A/B 测试来衡量对用户参与度的整体影响。

对比基线实验

（1）视觉内嵌表示（Visual）：利用深度视觉内嵌表示的最近邻进行推荐。视觉特征为 VGG-16 中提取的特征。

（2）注释内嵌表示（Annotation）：利用注释内嵌表示的最近邻进行推荐。文本注释内嵌表示使用 Word2Vec 模型训练得到。

（3）结合内嵌表示（Combined）：结合视觉内嵌表示和注释内嵌表示，利用

一个 2 层感知器计算结合内嵌表示。

（4）基于图的方法（Pixie）：Pixie: A System for Recommending 3+ Billion Items to 200+Million Users in Real-Time

消融研究

- max-pooling（ γ =max）
- mean-pooling（ γ =mean）
- mean-pooling-xent
- mean-pooling-hard
- PinSage

离线评价

Method	Hit-rate	MRR
Visual	17%	0.23
Annotation	14%	0.19
Combined	27%	0.37
max-pooling	39%	0.37
mean-pooling	41%	0.51
mean-pooling-xent	29%	0.35
mean-pooling-hard	46%	0.56
PinSage	**67%**	**0.59**

表 1：PinSage 与基于内容的深度学习基线方法的点击率和 MRR（Mean Reciprocal Rank）对比。总的来说，相比于最佳基线，PinSage 在点击率上提升了 150%，在 MRR 上提升了 60%。

内嵌表示相似性分布

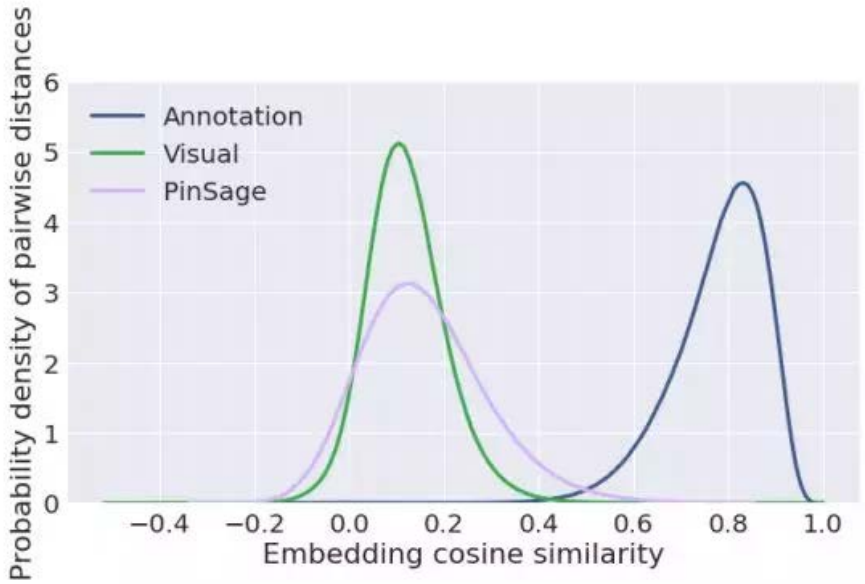


图 4：视觉内嵌表示、标注内嵌表示和 PinSage 内嵌表示的成对余弦相似度的概率密度。其中 PinSage 的分布范围最广，证明了学习到的内嵌表示具有足够的分辨率。

用户调查

Methods	Win	Lose	Draw	Fraction of wins
PinSage vs. Visual	28.4%	21.9%	49.7%	56.5%
PinSage vs. Annot.	36.9%	14.0%	49.1%	72.5%
PinSage vs. Combined	22.6%	15.1%	57.5%	60.0%
PinSage vs. Pixie	32.5%	19.6%	46.4%	62.4%

表 2：用户调查结果：哪种方法推荐的图像与查询图像更相关。大约有 60% 的用户更喜欢的条目是由 PinSage 推荐的。

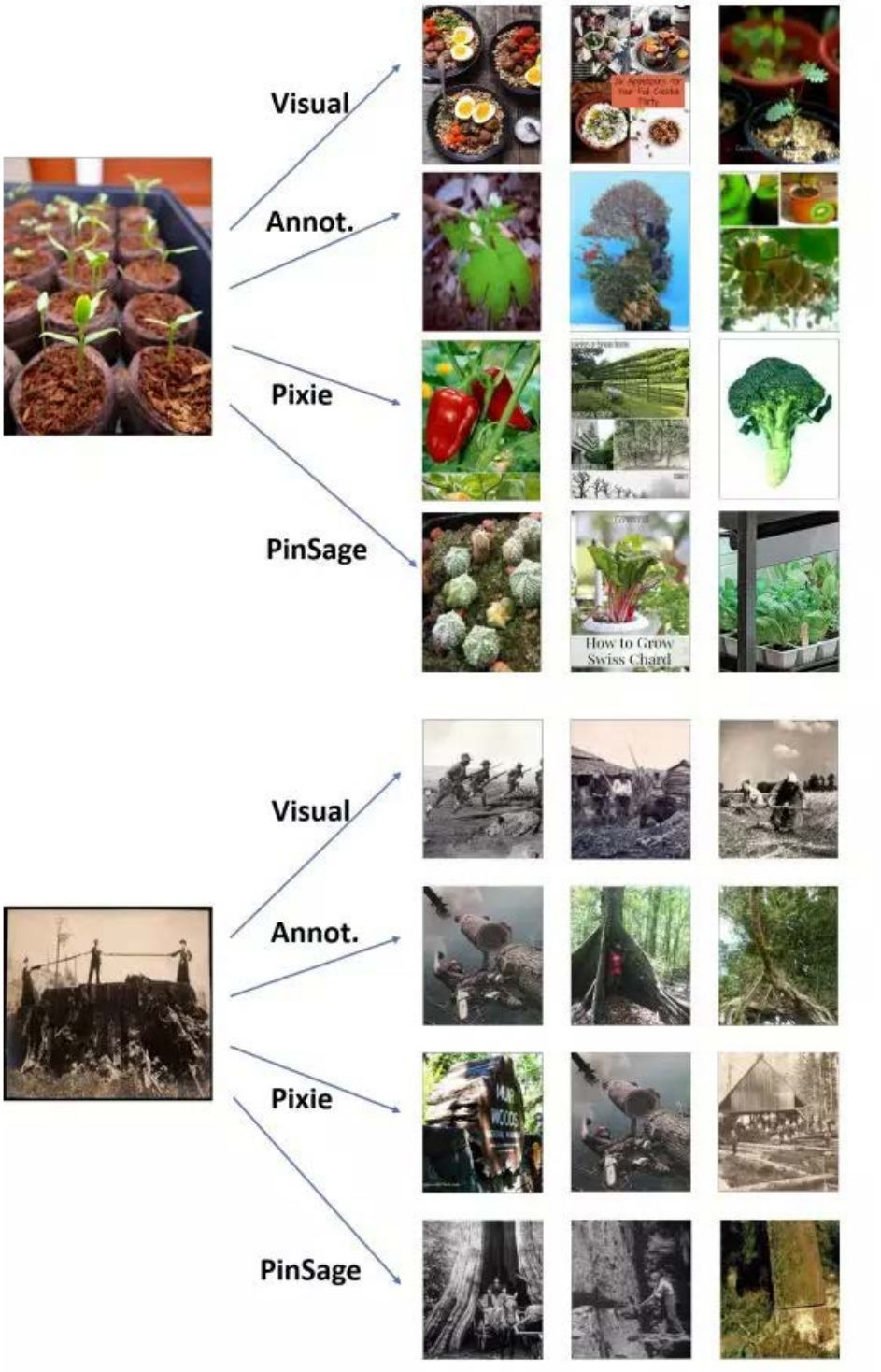


图 5：不同算法推荐的 Pinterest 图钉结果。左边的是查询图钉，右边的是分别利用 Visual、Annotation、Pixie 和 PinSage 计算得到的推荐结果。

图 6：PinSage 内嵌空间的可视化图。我们发现到条目内嵌表示的接近性与内容的相似性很好地对应，并且同一类别的条目位于内嵌空间的同一位置。

产品 A/B 测试

A/B 测试对比了 PinSage 和其他基于内容的深度学习推荐系统在 Pinterest 上对于家庭订阅推荐的表现。我们通过观察用户参与度来衡量算法的表现，衡量指标为用户保存推荐的比例——repin 率。我们发现 PinSage 推荐的图钉更容易被用户采纳，其 repin 率大概超过 Visual 和 Annotation 内嵌表示方法的 10%-30%。

训练和推理运行时间分析

Batch size	Per iteration (ms)	# iterations	Total time (h)
512	590	390k	63.9
1024	870	220k	53.2
2048	1350	130k	48.8
4096	2240	100k	68.4

表 3：不同批尺寸的运行时间对比，当批尺寸 =2048 时，训练效率最高。

# neighbors	Hit-rate	MRR	Training time (h)
10	60%	0.51	20
20	63%	0.54	33
50	67%	0.59	78

表 4：邻域 T 的选择对表现的影响。我们发现随 T 的增加，收益逐渐减少，并且两层的 GCN，邻域尺寸为 50 的时候能最好的捕捉到节点的邻域信息，同时保证计算效率。

总 结

这篇论文提出了 PinSage，一个具有高度可扩展性的图卷积网络，能够学习网络规模下包含数十亿数据节点的图结构，为节点生成内嵌表示。通过局部卷积、重要性池化等方法，大大提高了 GCN 的可扩展性。作者介绍了重要性池化和课程训练两个训练技巧，显著提升了算法的性能。作者将 PinSage 部署在 Pinterest 上，并且在多个推荐任务上全面地评估了学习到的内嵌表示的质量。广泛的离线评价、受控用户研究和 A/B 测试结果显示，PinSage 在项目 - 项目推荐任务（pin 相关的推荐）和家庭订阅推荐任务中都取到了最先进的性能。

查看论文原文：

https://arxiv.org/pdf/1806.01973.pdf