TOP TECH TEAM

中国卓越技术团队访·谈·录

CHINA

COVER STORY

封面故事

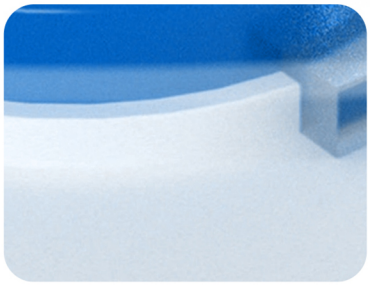
国内最大的 C++ 软件项目之一，WPS 的“自守”之道

2021年

第五季

InfoQ

扫码了解更多



目录

封面故事

国内最大的 C++软件项目之一，WPS 的“自守”之道.....i

重磅访谈

从 NoSQL 到 Lakehouse，Apache Doris 的 13 年技术演进之路.....1

同程旅行云原生改造：半年完成全部核心业务改造、抗住爆品300%流量冲击.....14

存储正在经历新一轮架构革命：戴尔科技集团在数据湖的探索和思考.....24

打破国外垄断，出门问问主导研发的端到端语音识别开源框架 WeNet 实践之路.....38

为了让你搞定数据库选型，这些工程师重写了 26 万行代码.....49

封面故事

国内最大的 C++软件项目之一，WPS 的“自守”之道

作者：罗燕珊

一个历经了三十多年发展的软件，这话听起来就让人感到沉甸甸的。但这款产品经历无数风雨后不仅活得好好的，至今还在不断焕发新的生机。

本期《技术先锋团队访谈录》封面故事的主角是金山办公旗下的 WPS 研发团队。WPS 是通用型国产办公软件的代表，可能是目前国内最大的 C++单体项目。有人用这样一种比喻去形容 WPS 的源代码：“就像翻动沉积岩，一层一层，这层是框架、往下是更老的核心框架、再翻则是更底层的数据处理等等，和考古一样（有趣）。”虽然底层的东西一直都处于比较稳定的状态，不过在底层之上，也有另一番热闹景象。

> **采访嘉宾：**
>
> 庄湧，金山办公高级副总裁、研发中台事业部总经理
> 晁云瞳（Yeppy），金山办公助理总裁、金山文档事业部副总经理，兼 WebOffice 负责人
> 柳杨，金山办公总监，负责 Linux 方向研发

重新演绎 WPS

1989 年，WPS 1.0 问世后，在中文文字处理软件市场上获得了消费者的广泛追捧。一时间，风头无两。

上个世纪 90 年代末，微软公司的办公软件跟随着操作系统进入中国市场，装载在中国消费者电脑上，MS Office 成为“事实标准”的时候，也是 WPS 推翻重来之时。这是一段不得不提的历史。当时 WPS 不是要改写，不是重构，而是需要一次彻底的重写。

“我们要让用户能够在 Windows 端重新使用 WPS，首先就得承认微软 Office 的市场地位以及事实，而且要想让用户能够无缝切换过来，就必须得做到兼容。”2003 年进入 WPS 的庄湧表示，**当时 V6 的主要目标是“三大兼容”，包括：文件格式兼容、操作习惯兼容、二次开发接口兼容**。

2002 年～2005 年，历经三年时间，一个涵盖了数百万行代码，内部研发代号为“V6”的 WPS 2005 版本，在一个全新的架构下“重生”了，并且能很好地兼容微软办公软件文件格式。

对于当时的 WPS 来说，只有这三大兼容做好了，才能让 WPS 有机会跟微软 Office 同台竞技。但也因为“完美兼容”，一些质疑抄袭的声音开始出现。

同样在 2003 年加入 WPS 的 Yeppy 还记得，当时有一些用户认为 WPS 推翻重写的决策是错误的，“说我们‘放弃了传统’。重写之后，WPS 原先很多心思巧妙的设计和功能来不及做，所以都砍掉了。我们是为了重新回到同台竞技的状态，但有些用户不理解，为什么不做自己的特色而是跟着人家跑了。”

当时 WPS 2002 的产品在政企客户中的口碑并不好。Yeppy 表示，问题主要体现在几个方面：首先微软 Office 的文档格式逐渐成为主流，WPS 在打开微软文档的时候，容易出现各种错误和异常。其次产品交互和处于市场领先地位的竞品有很大不同，站在客户的角度，其实是给他们带来了不小的学习成本。

第三，还是兼容问题，微软 Office 不再兼容 WPS。客户的系统（比如 ERP、OA 系统）都已经跟 Office 打通，需要导出、导入 Office 文件并在这些系统里处理，但这些系统接入 WPS 后发现功能都失效，不能用了。

“其实说来说去就是一个问题，企业已经有的这些文件是 IT 资产，员工生产力也是资产，各种系统是 IT 基础建设的资产，从做 B 端市场的角度来说，我们至少要把这些资产给保护好，人家才愿意用我们的产品。"Yeppy 说道。

于是，兼容成了 WPS 突围的唯一出路。文档格式兼容，保证不管用户从哪个端打开，文档都不会跑版、数据表现一致；操作习惯兼容，则要让用户在使用的过程中每个动作都不受影响，不另外产生新的学习成本；二次开发接口兼容主要是面向政府企业客户，这些客户更多是跟系统对接，很多时候需要用到丰富的二次开发接口，所以这个接口也要跟微软的保持一致。

那时候，像庄湧和 Yeppy 这批新人进 WPS 后就投入到了做兼容这件事上。

四大引擎的演进

以文件格式兼容、操作习惯兼容、二次开发接口兼容为目标，WPS 2005 版本构建

了全新的四大引擎，分别是文字的排版引擎、表格的计算引擎、演示的动画播放引擎以及公共的图形对象渲染引擎。

庄湧表示，本质上，支撑 WPS 后续发展的基础就是当年的四大引擎，演进过程可以分三个阶段去看。

2005 年 9 月 3 日，WPS Office 2005 封盘纪念。

第一个阶段是 PC 时代，彼时四大引擎的重心是打磨自身，最基本的目标是保证 WPS 在 Windows 端的稳定性以及性能提升。

文字排版引擎方面，不仅要让 WPS 打开微软 Office 文档的时候一页不差、一行不差、一字不差，同时也要提升性能，在打开几百页、上千页文档的时候，无论是在哪个位置编辑，甚至在多人协作时（当时还不是现在普遍的基于网络端的协作协同，而是需要通过文件传递和发送的方式去做批复），都要保障过程的顺畅。

包括后来进入以银行为代表的金融企业，打磨表格的计算引擎的重点之一也是性能。“要做到在百万行数据中，无论在哪个地方操作都稳定且顺畅，背后其实要花好多功夫去优化。”庄湧说。

第二个阶段是移动时代。2011 年，雷军重回金山软件担任董事长，并一脚把 WPS 踹进了移动互联网。那会移动手机起步不久，受限于内存和屏幕尺寸，对于像 Office 这样的产品是否适合在手机上使用，该怎么用，大家心里都没底。但也正是因为这个机遇，WPS 的团队才真正开始用产品思维去思考问题。此前，为了与微软 Office 更好地兼容，团队更多是思考怎么和微软做得更像。

团队面临的第一个问题就是，移动端的办公软件应该采用什么引擎？直接移植 C++开发的 PC 端引擎到安卓平台，会遇到很多兼容性问题，拖慢研发进度。因此研发团队选择了用 Java 语言开发新引擎，以赶上移动互联网时代的快速发展。而在做 iOS 版本的时候，庄湧既借鉴了安卓团队的移动开发经验，同时由于 iOS 平台支持 C++语言的开发，又复用了 PC 端的内核代码进行移植，节省了时间。

然而，落地执行中团队遇到了更大困难：移动场景下办公需求发生了剧烈变化，功能

不能照搬照抄，而是要围绕小内存、小屏幕等特性做取舍，这是移动时代引擎迭代过程中最大的挑战。“但是当时我们内部流行一句话，理解要执行，不理解也要在执行的过程中去加深理解。”庄湧说。

当时移动端仍然是新鲜事物，要把电脑里的办公软件搬到小小的手机屏幕里，这件事从没有哪家公司或者是哪个人尝试过，没有经验可借鉴，短时间甚至找不到合适的人。为了支持移动版开发，WPS 安卓团队的人不得不从 PC 团队中抽调，且均是骨干。

随着人均持有的办公设备越来越多（比如手机、iPad、电脑等），为了解决文档在不同设备间流转所面临的问题，借助云技术，金山办公研发了 WPS 云文档。而随着对 Linux、Mac 等平台的不断适配，四大引擎在移动时代也充分验证了跨平台能力。

第三阶段，随着 Web 端需求的增长，尤其是疫情下在线协作办公需求爆发，WPS 团队进一步在引擎中加入了协同特性。

新的不稳定状态

在日益月新的互联网行业里，新产品新事物层出不穷，某种意义上，发展了许多年的 WPS 已经是个相对稳定和成熟的产品，成长空间是否已经接近天花板？

对于这个问题，Yeppy 说：“得看怎么去理解‘稳定’这两个字。如果是说内核，内核能不能保证不出错呢？”

他以 WPS 的在线协作文档产品「金山文档」举例道，“为了能帮用户处理 Office 文档，我们在服务器上也跑了一个自己的内核。假如我们处理同一份文件，你在编辑，

我也在编辑，那这两个编辑动作，该怎么样合到一个文件去，这里面有一个 OT（Operational Transformation）引擎，并且这个 OT 引擎一定要和我们原来的引擎相结合，保证计算、排版、调度、绘制、动画这些功能都是正常的。"

“加了上面这些东西以后，原来的功能还算不算稳定呢？”Yeppy 进一步表示，从最基础的功能来说，WPS 看起来没有大的变化，但能力一直在横向和纵向扩展，比如协作是通过云端的引擎来做，比如多平台适配把内核从单一环境“搬”到了不同的设备环境，但它并不是直接搬就能跑起来，里面还有大量的开发和适配工作，这其实是一种“新的不稳定状态”。

WPS 的客户端基本覆盖全平台，在服务器、云存储、浏览器前端等方向都有人在研究、在折腾各种新技术。

Yeppy 介绍道，在线协作文档的服务器上跑的是核心引擎，怎么去做多人之间的操作、协作变换则是另外一个模块，用户连到服务器后怎么做并发承载，则是用的具备并发特性的语言去处理。为了能在各种浏览器运行，前端用的是 JavaScript，甚至在 10 年前团队还曾经试过用 Adobe 的 Flex 技术去做在浏览器里运行的小办公平台。诸如此类，不胜枚举。“**折腾新技术其实是为了新的场景做准备**。”Yeppy 说道。

庄湧亦表示，从 PC 到移动再到云端，从工具转向云服务，为了解决用户的跨设备以及多人之间的协同协作问题，WPS 一直在不断吸收新技术，包括 AI。**但技术只是手段，目的是帮助用户解决问题和提升工作效率，本质还是“用户第一”的理念，基于用户的场景和真实需求去应用技术**。

Linux 版 WPS：从“用爱发电”开始

即使是“小部分”人群的需求，WPS 也没有忽视。如今，WPS Office 基本做到了全平台覆盖，包括在早期并不能带来商业价值的 Linux 平台。

今年是 Linux 系统诞生三十周年，若从 2011 年立项算起，WPS 陪伴 Linux 其实也有十年了。柳杨从 2013 年开始参与 Linux 版 WPS 开发，他坦言，从操作系统的角度看，当时的 Linux 相比起 Windows，市占率非常低，而大家决定做 Linux 版的出发点其实很简单，因为有用户需要。

“那时候用户给我们发了不少邮件，说想要 Linux 版，因为觉得 Linux 平台没有好用的办公软件。”柳杨补充说，决定开发 Linux 版也跟 WPS 全平台覆盖的愿景有关，“**我们就想让全平台用户都能有好用的办公软件**。Linux 用户虽然不像 Windows 用户那么庞大，但是从 Linux 的用户画像来看，其中包括大量技术人员，有一些是真正的系统爱好者，有一些是在 Linux 平台上工作的，这些用户其实也是我们需要考虑到的一批优质用户。”

“每一次我们发新版本的时候，尤其在 Linux 平台，很多用户就会给我们好评，说我们是一家良心企业，因为做 Linux 社区版不赚钱，但是我们一直在维护，默默去做这个产品，用户也常给我们一些正向的反馈。”谈及怎么坚持做这么多年 Linux 社区版的时候，柳杨表示来自用户的支持也是一种动力。

不久前，Linux 版 WPS 收到来自用户的暖心评价

不过在发展过程中，Linux 版也有过一个“小插曲”。2017 年上半年，一条关于“WPS Office for Linux 项目中止”的消息不胫而走，在圈内引起过一阵热议。随后金山办公官方对此作出否认，称“WPS Office for Linux 并未停止更新，也从未有过停止更新的任何计划。”

柳杨回忆道，“官方说明是我这边推动市场部去做的。”为什么会出现这个传闻？他进一步解释称，原因主要有两方面，一是 Linux 面向个人的社区版分为海外版和国内版，当时有人提到海外版近一年没更新，很快网上就开始流传 Linux 版停止更新的猜测；二是跟当时产品刚好碰上大版本更迭有关。

“17 年属于我们的一个转折点，WPS 要从 WPS 2016 升级到 WPS 2019，当时内部都在重点准备这个大版本，在推出 2016 加强版的同时，我们也在准备研究和开

发 WPS 2019 版。大版本开发的工作量非常大，因此时间跨度也比较长。"柳杨指出，大版本更新不仅做了新图表、云文档、SmartArt 等大量新功能，同时做了各种优化工作以及 bug 修复。而在这些工作没做完之前，发小版本其实对用户来说意义不大。

此外，柳杨还提及了当下一个重要的行业变化。近几年，在 IT 国产化的浪潮下，强化产业链供应链自主可控，WPS 在 Linux 上的商业价值迅速显现，因此 Linux 团队经常需要闭关去做新的需求。

据最新数据，金山办公的业务覆盖中国 30 多个省市自治区政府、400 多个市县级政府，连续多年为包括党政机关、金融、能源、航空、医疗、教育等领域在内众多行业提供定制化的办公产品和服务，帮助政府和企业加速实现数字化、智能化办公。

程序员文化的传承

2019 年 11 月，金山办公上市。现在人们往往只会看到其"光鲜亮丽"的一面：金山办公的市值挺高，WPS 活得挺滋润。但其实过去有很长一段时间，WPS 的日子并不好过，其发展起起伏伏。金山办公名誉董事长雷军亦曾说过："曾经有很多人很多次劝我放弃 WPS，但是我们能够坚持下来，还是跟我们的使命息息相关，而不是一个纯粹的商业决定。"

"技术立业"的程序员文化是金山从创办之初就传承下来的。如今金山办公数千名员工，60%以上是程序员，总数超过 2188 人，研发费用占营收比例超过 30%，位居科创板前列。Yeppy 认为，WPS 能坚持这么多年，一个很重要的原因是跟公司里的研发人员多有关，研发人很多时候会比较理想化、有使命感。公司历代领军人物细数

下来，其实都是研发出身。

最早可以追溯到 1988 年，当年求伯君把自己关在一个旅馆的小房间，用汇编语言敲出了 WPS 的第一个版本，这十多万行代码揭开了中文排版、中文办公时代的帷幕，这是个振奋人心的起点。“**我认为这算是程序员个人英雄时代巅峰的辉煌。**”Yeppy 感慨道，**因为在那之后，软件开发就更多是群体作战模式**。现在公司这么多号人，就是为了把几条产品线做到极致。

“来 WPS 十年，我一直就是做 WPS Linux 版这款产品，从最开始只有几个人做，到现在我们也有非常大的团队去做，就一直围绕一个产品，把一件事做到极致。”柳杨的语气中不乏坚定，在他看来，整个公司的氛围和基因里具备专注、热爱和理想的东西。“在我的认知范围里，我的职场生涯不用做太多花花绿绿的东西，也许把某一件事做好了，做到极致也就够了。”

两个月前，在金山技术开放日的活动上，庄湧曾介绍说，金山办公的技术人员具备一些独特特质，比如喜爱不断“折腾”、精益求精等。具体来说，金山办公一代代传承下来的优秀程序员在写代码之前都会有多个维度的思考，从深度性到通用性，再到可维护性以及可扩展性。

2002 年那一次 WPS 代码推翻重写，同时进行重塑的还有 WPS 的内核引擎。这么多年过去，当时重塑的四大办公软件引擎作为底座能支撑到今天，或许也恰恰印证了金山办公技术人的实力与底蕴。

重磅访谈

从 NoSQL 到 Lakehouse，Apache Doris 的 13 年技术演进之路

整理：蔡芳芳、Tina

采访嘉宾：

百度 Apache Doris 主创团队

马如悦、张志强、陈明雨、武云峰、杨政国、缪翎、鲁志敬等

从 2008 年第一个版本开始到今天，Apache Doris 已经走过了 13 个年头。从推出之初为了满足百度商业系统的业务专用需求，到后来为解决通用报表与数据分析需求进一步改造，并在 2017 年改名 Palo 开源（详见 InfoQ 当时报道），再到 2018 年用回 Doris 这个名字并进入 Apache 软件基金会孵化，Apache Doris 的愿景一直是成为世界顶级的分析型数据库产品。但与此同时，进入云原生时代，Apache Doris 也已经有了它新的定位和目标。

早在 Apache Doris 开源之初，InfoQ 就曾采访过项目负责人马如悦，而今年正好是这个项目开源的第四个年头，我们再一次找到百度 Apache Doris 主创团队，跟大家聊聊 Apache Doris 的过去、现在和未来。据透露，目前 Apache Doris 的毕业筹备工作已经启动，团队接下来的工作重心之一就是推动 Apache Doris 尽快从 Apache 基金会毕业成为顶级项目。

以下内容整理自访谈实录。

Apache Doris 的新目标

InfoQ：Apache Doris 发展至今，已经 13 年了，如果要将发展历史划分成几个阶段，您们认为是怎样的？

Apache Doris 团队：Doris 的十多年历史，走到今天，我们重新去审视，去掉细枝末节，大体可以分为三个阶段：

- **“NoSQL”阶段（2008-2011 年）**

这个阶段主要是满足百度商业系统几个大业务的专用需求。这几个业务，需要给几十万到几百万的客户或者用户提供实时的报表分析与可视化能力。而传统的分析数据库，基本上主要支撑公司内部自己的 BI 需求，而这些 BI 需求，对数据入库的时效性、查询的并发性、查询的延迟性要求都不是很高。所以使用传统的分析数据库根本无法支撑互联网公司全新的分析需求。当时，我们采用了那时候市场上比较火的 NoSQL KV 数据库来存取数据，并且自己实现了一个专用的分布式查询引擎，这个查询引擎不是 SQL 接口，而是类似 REST API，提供了一些聚合函数调用给业务使用来解决需求。

- **“NewSQL”阶段（2012-2020 年）**

这一阶段的研发重点主要是满足以下新的需求：1）通用的报表与数据分析需求开始增多，大家需要 SQL 接口；2）原来的 KV 存储引擎无法提供足够的性能支撑越来越多的需求。所以，我们开始研发新的 Doris 系统。首先，我们研发了全新的单机列式存储引擎 olapengine，先是使用单机 MySQL 来做 SQL 查询引擎，通过分库分表方式来解决分布式大规模问题；后来又将单机的列式存储引擎改造为全分布式列式存储

引擎，把单机 MySQL 查询引擎改造为 MPP 的 SQL 查询引擎。分布式存储和分布式 SQL 查询引擎的改进，大大提升了性能和应用场景满足度，Doris 在百度被大规模采用。2017 年，Doris 也正式对外开源。

- **“LakeHouse”阶段 （2021 年开始）**

随着用户需求不断进化和云计算技术的广泛推进，Doris 需要考虑离线在线一体化、存算分离、实时更新、半结构化数据分析支持等需求。这些需求总结下来，简单地说就是用户希望拥有传统 MPP 数仓和基于数据湖的湖分析融合能力。目前 Doris 就处在这一阶段，正在全力研发这些新的功能。

InfoQ：Apache Doris 的设计目标是为了解决什么问题？

Apache Doris 团队：因为技术和需求会随着时间发生变化，Doris 也会跟着每个阶段去制定不同的目标。

第一阶段 Doris 主要还是满足专用系统的统计分析需求，第二阶段主要是满足通用的报表与数据分析可视化需求。

到今天，我们发现用户或者客户对数据的分析需求，逐渐收敛为三大块：

1. 50%的需求依旧是各类报表和数据分析可视化需求，就是我们经常提的 BI 的需求；
2. 20-30%的需求，是对日志等半结构化数据的搜索分析需求；
3. 20-30%的需求，是对数据科学与机器学习的需求；

而新的 Doris 将会针对这三类场景，进行重点功能和性能设计，以便支撑这三类需求。

InfoQ：Apache Doris 最初的定位是什么？10 多年过去后，这个目标定位是否有了变化？

Apache Doris 团队：Doris 最初的定位是**新式数仓**，满足在线的数据分析场景，主要以高并发小查询的性能最为出色。但是发展到今天，它的定位正在发生变化，这个主要变化可以用一个 **T 形（一纵两横）**来说明。一纵就是指把原来 Doris 最擅长的在线结构化 MPP 数据分析性能优化到最快，而导入实时化、存储读写性能优化、计算性能优化，这些会学习和借鉴 ClickHouse 的一些设计。两横之一是**支持半结构化数据**，当前全球很多对日志等半结构化数据分析都使用 Elasticsearch，Doris 后续会加强对 ES 所支持场景的满足能力；另一横，就是**拥抱云原生技术**，支持存算分离，支持较大的查询，满足对数据科学与机器学习场景的支持，这一块需要多去借鉴 Snowflake 和 Databricks 的一些设计。

当前 Doris 的新目标，就是主攻这个类似 T 形的一纵两横。

只关注性能过于片面

InfoQ：现在业内出现了越来越多的各种 OLAP 软件，相比较起来，您认为 Doris 具有什么样的优缺点？适合什么样的使用场景？

Apache Doris 团队：Doris 和很多其它竞品不大相同的，主要是源于产业实践。数据库技术不同于应用层软件，数据库技术的研发需要积累多年，并且还要经历大规模

的实践检验。在实践中发现问题、发现需求，然后解决，这样整个系统才会比较实用。

Doris 运维非常友好：很多数据库公司研发数据库，但是自己又没有大规模使用，所以对运维友好性支持欠缺。Doris 来自于实践，所以在多年的发展中增加了大量方便运维的特性，比如高可用、方便的扩缩容等。

比如为了节省成本，Doris 支持分层存储，即一个表的一个 Partition 分区，可以设置为过了多久以后自动从 SSD 磁盘转移到 SATA 硬盘上。

比如 Doris 的后端节点，需要管理员在前端主节点手动添加，好多人可能不理解，为什么不是后端节点自动汇报？问出这个问题，就可以发现其没有一线工程经验，自动汇报会带来很多潜在的运维风险，都是我们曾经有过的血泪教训，比如一个很久以前死掉的节点，突然重新启动，那么很可能就会误加入进来，造成查询不可控。

比如 Doris 支持物化视图和基础表的数据一致性，这都是源自一线业务对数据一致性的强烈要求，业务无法接受物化视图表和基础表的不一致，因为对终端用户来讲，不一致会带来很多的理解问题。

综上，Doris 里面有大量的这种设计，这些功能对于不是一线运维的同学，或者运维经验不丰富的同学，可能不会了解到其好处，反而还会认为是坏处。

Doris 主要做的不好的我认为有两处，一个是对传统数仓的兼容性，毕竟它来自互联网公司，在推广到传统数仓领域时，在一些 SQL 兼容性上遇到了一些问题，当前正在优化解决；另一个是对云原生技术的全面拥抱，Doris 最初设计时，主要还是考虑私有化部署，那时云计算还不火。但当前云技术的采用正在加速，所以 Doris 后续也会加强对云原生的深度融合适配。

InfoQ：2017 年，您在 InfoQ 的采访中说过“性能不该是唯一关注点”，现在您们对 Apache Doris 的要求是否有变化？

Apache Doris 团队：我们的观点还是没有变化，虽然市场上依旧是看性能为主。我们认为一个生产级别的数据库，要综合考虑各个方面，稳定性、易用性等，都需要考虑在内。比如，很多人一直抱怨 Doris 没有 ClickHouse 快，这个我是认为比较片面的。

就拿性能来说，一个在线系统，尤其针对高并发的在线分析系统，需要关注整个系统对众多并发查询都能提供稳定的响应，还要充分考虑预留足够的资源给可能突发的一些查询。如果一个查询就把所有磁盘和 CPU 全部用满，那么其它查询如何保证得到足够的资源进行响应？多并发来了，如何保证系统内存不崩？所以，**有些设计不是能不能做到的问题，而是要考虑应该不应该这样做的问题**。比如 Doris 的每个查询，就会控制内存和 IO 线程的使用，并不是全量将系统的算力资源耗尽，而是在尽量满足性能响应需求的情况下，理性控制其使用量。

而易用性、运维友好这个可以追求极致，你会看到 Doris 为了不额外引入 ZooKeeper 这种系统造成运维复杂，自己研发了一套内置的多 FE 系统。

当然，我们在面向 To B 推广 Doris 时，很多人经常会通过单一 SQL 的查询性能来衡量这个系统优还是劣，POC 测试对性能非常看重。针对这些情况，Doris 后面会采用类似汽车中的驾驶模式那种形式，提供 Normal 和 Sport 模式。当你将 Doris 设定为 Sport 模式时，Doris 将会以性能最快方式运行，榨取系统每一滴算力。而 Normal 模式，我们更建议在线上使用，以保持系统的稳定性和应对突发请求的能力，

不要让系统始终运行在崩溃边缘。

InfoQ：您们团队在这几年的维护过程中，投入了多少人力，解决了哪些比较关键的技术问题？做了哪些功能优化？

Apache Doris 团队：这几年团队成员有过变化，但团队规模一直在稳步增加，目前好几个方向的人员数量加起来有 40 多人，既包含了 Doris Core 核心数据库的研发，也包含了百度智能云上产品和外围生态组件的前后端开发人员，还有一支实力强大的产品和运营团队。

从开源至今，在社区的共同努力下，Doris 得到了前所未有的飞速发展，做了非常多的功能迭代和更新。主要包括以下几方面：

- 流式导入功能帮助 Doris 从分钟甚至小时级别的导入延迟推进到了秒级，更好地支撑了准实时的业务需求；
- 完全重构了存储引擎，提升扩展性的同时，支持了包括二级索引、字典压缩编码在内的多项实用功能；
- 进行了大量的大数据生态打通工作，包括 Spark、Flink、ES、Hive、Kafka 的直接连通，使得 Doris 不再成为数据孤岛；
- 在明细数据上扩展了预聚合模型，完成了明细、聚合模型的数据统一访问；
- 全新的向量化执行引擎和资源隔离方案也即将发布，将进一步提升 Doris 的数据分析性能和业务应用场景；
- 还有其他非常多的稳定性和易用性的提升，也是得益于开源后社区用户的不断打磨和反馈。

InfoQ：Apache Doris 和数据湖架构之间有哪些区别和联系？

Apache Doris 团队：Doris 最初设计是存算一体化的 MPP 数据仓库，偏在线分析。而数据湖架构的分析，主要是存算分离，偏离线或者交互式分析，存储引擎一般是 HDFS 或者对象存储，而分析引擎类似 Spark/Hive/Presto。

从去年开始，大家已经开始广泛地推进 Data Warehouse 和 Data Lake 架构的融合，即是所谓的湖仓一体，Lakehouse 的架构。Doris 也正在从数仓架构向 Lakehouse 演进。

InfoQ：在周边生态上，最近几年有了一个什么样的变化？

Apache Doris 团队：最大的变化就是 SQL 的取胜，实时的取胜，云原生的取胜。

1. SQL 的取胜：从使用 Java 写 MapReduce、Pig，用 Scala 写 Spark 程序到 PySpark，最终还是 SQL 笑到了最后，SQL 占据了数据分析的 80%；
2. 实时的取胜：人们对于速度的追求是无止境的，一个事情不能做，希望可以做到，这个事情可以做到了，希望能越快越好。数据分析领域正在全面拥抱实时化的需求，希望实时的数据导入，希望实时的数据产出。从离线做起的 Hive、Spark 正在不断优化查询性能，而那些直接从实时性能切入的 MPP 数仓和实时湖分析，比如 Presto，正在全面攻占在线实时市场；
3. 云原生的取胜：云原生已经不再是噱头，而是正在成为关键赋能技术，Snowflake 的大卖，让云原生成为每个数据分析产品都绕不开的领域。

基础设施软件必然要开源

InfoQ：您们当初是如何选择开源的时机的？Doris 加入 Apache 经过了一个什么样的流程？

Apache Doris 团队：Doris 从 13 年设计新版时，就考虑到了未来会开源出去，所以，我们在 13 年设计时，就没有依赖百度内部任何一个库，并且整个系统也不依赖百度任何服务就可以独自运作。

百度很多系统难以开源，主要是开始设计时，对百度内部闭源库和内部系统的依赖较多，导致开源的时候需要大量重写，最终使得开源难度非常大。Doris 没有这个问题。

Doris 从 13 年就坚信未来基础设施软件必然是开源的，只有开源才能保持活力和持续迭代。并且像 Doris 这种基础软件，需要较大投入，如果不开源，不寻找其它价值点，是很难让一个大公司持续投入资源来维持其不断发展的。

Apache 是对开源极其友好的基金会，在大数据领域，Apache 软件基金会的项目都极具影响力，比如 Hadoop 和 Spark 都是 Apache 软件基金会的项目，所以 Doris 开源时也选择了 Apache 软件基金会。

InfoQ：您们认为什么样的开源软件可以称之为是开源成功的？

Apache Doris 团队：我们认为衡量开源的成功与否关键在于以下三点：

- **被广泛认可的产品价值**

- **繁荣、自治、良性发展的社区生态**
- **开源与商业化的平衡与共存**

InfoQ：您们怎么看开源文化？您们团队是如何构建开源文化的？

Apache Doris 团队：作为任何一个技术人员，开源已经成为了一种信仰，一方面是解决更多人的问题所带来的成就感，另一方面就是社区的广泛参与必定为项目带来更好的活力，所以我们非常鼓励团队成员参与开源。

InfoQ：在参与开源的过程中，您们有什么样的经验可以和大家分享？

Apache Doris 团队：开源社区不是只有维护团队，每一个开源产品的使用者其实都是开源社区的一份子。在使用开源产品的同时，也可以多多回馈社区，这样开源产品才能有更旺盛的生命力。

这里引用我们社区里一些用户的话“在开源过程中，你会结识志同道合的朋友，获得朋友的认可与支持，甚至能够与自己崇拜的业界大佬共同交流。”、“我们每个人都有能力让社区变得更好，在社区帮助我们成功支持业务的同时，我们也应该尽自己所能，去回馈社区、帮助社区，哪怕只是一个文档的修复，也是帮助。”

上面其实也是我们想传达的理念，参与开源其实没有什么门槛，我们希望能有更多的小伙伴参与到社区建设中来。不论是提交 Issue 或参与讨论、帮助我们打磨产品和丰富功能，或者是修改和完善系统文档，或者是贡献应用案例、让我们知道 Apache

Doris 在真实业务场景中还能发挥出超出我们想象的能力，亦或是口碑相传、让 Apache Doris 被更多人知晓，都是帮助 Apache Doris 在成长道路上更进一步！

InfoQ：您们如何看待开源项目社区之间的竞争与合作？面对中国开源市场，您有什么好的建议、寄语与大家分享么？

Apache Doris 团队：开源社区之间其实不存在竞争一说，倒是有非常大的合作空间。

代码和社区其实不用一概而论，代码是代码，社区是社区。使用代码的人是用户，这些用户是完全自由的，如何选择一款开源产品及其代码是由用户自己的技术认知和业务需求来决定的，这里的竞争是存在于代码层面的。而开源社区其实在代码之上，也就是 Apache 理念的 Community Over Code，每个人都可以参与到社区，不管是不是用户，不管有没有需求，都可以作为独立的身份加入到社区里来。

社区的发展有先后之分，社区间的合作可以帮助社区在更大范围的人群中得到传播，也能帮助新兴社区更快成长，还可以让开源代码汲取到更丰富的养分。

对于中国开源市场，希望能有更多的开源项目可以蓬勃发展，这也会让每一个人从中受益。

开源与商业化协同

InfoQ：您们如何理解开源和商业化之间的关系？

Apache Doris 团队：当前大量底层技术产品都采用开源模式，客户也愿意采用开源产品，所以大环境也会逼着你去开源；另外，在商业市场中存在着 2/8 原则，即 80% 的收入来自 20%的付费用户，而另外 80%的用户贡献收入并不高，然而前者无论开源与否，都可能付费；而后者则更喜欢开源产品；但是，其中最重要的一条规律是，前面 20%付费用户的选择会参考后面 80%用户的选择。因此从商业上来看，让产品开源，让 80%的用户免费使用你的产品，必然会带来很好的口碑，这直接会影响到那 20%的高付费用户，20%的这群高付费用户更多地关注服务。

所以，**对于未来的技术产品，开源可能成为必须，这个“必须”不一定损害商业模式，反而会促进商业上的成功**。最近一两年我们也跟很多面向开源软件领域的投资人有过多次沟通，开源和商业化的之间必定是相互成就的。

但开源与商业化如何协同是当前和未来开源面临的问题。**开源与商业化需要找到一个良性并存的方式，才能将开源推向另一个高度。**

当前开源与商业化如何协同，业内都在探索，还在苦苦寻求中。付费技术支持、Open Core、SaaS 模式仍然是三个主要的商业化模式，但是在实际操作中都有其大的问题。

但是，我相信，随着各类基于开源的商业化公司的不断探索，成功与失败，最终一定会探索出比较好的商业模式。

InfoQ：Doris 的商业化路径是怎么规划的？目前已经有哪些商业客户？

Apache Doris 团队：商业化路径方面，我们认为云上才是未来，因此我们数年前就在百度智能云上推出了基于 Apache Doris 的企业版产品 Palo 并提供了云端托管服务，通过云服务的优势（比如按需取用和更加可控的海量资源、从繁琐的运维工作中解放人力等）去满足更多企业上云的需求。我们**云上 Palo 的核心代码与开源版完全一致**，避免用户可能担心被公有云厂商强绑定。我们公有云托管服务的价格，比用户购买物理机甚至云上虚机自行搭建的费用还要低。我们还基于 Palo 提供了管控运维平台等一系列云上组件，通过丰富的外围组件给用户带来体验更加的云上服务，目前我们的自助分析平台 Studio 和可视化运维监控 Manager 已经逐步成熟起来。

目前我们已经拿下的商业化客户大概有接近 50 家，包括银联商务、知乎、四川航空等，更具体的数字就不进一步展开了。

InfoQ：Doris 所在的市场或所覆盖的应用场景，市场潜力还有多大？

Apache Doris 团队：数据分析场景主要是三大块：数据仓库与商业智能、日志检索与分析、数据科学与机器学习场景，这三大场景占据了客户 80%的数据分析需求。这三大场景的不断发展，未来一定会将数据分析的需求推为企业 No.1 的需求。从各大咨询调研报告来看，数据分析产品的增长依旧位列各种软件产品的第一位。

同程旅行云原生改造：半年完成全部核心业务改造、抗住爆品 300%流量冲击

作者：褚杏娟

采访嘉宾：彭涛

2018 年 3 月，同程集团旗下同程网络与艺龙旅行网合并为同程旅行，同年登陆香港联交所主板挂牌上市，成为港股“OTA 第一股”。财报显示，2021 年上半年，同程艺龙 MAU 约为 2.56 亿，其中在第二季度，MAU 达到 2.8 亿，同比增长 58.3%，创下了历史新高。上半年，同程旅行的各项核心业务增长均远超行业增速，并超越疫情前水平。

业务量的增长让同程旅行的技术团队感到欣喜，但另一方面这也意味着团队需要直面高流量带来的新挑战，云原生改造成了解决问题的关键。虽然同程旅行从 2019 年年底才开始进行云原生改造，至今还不到两年，但成果显著，成功帮助同程旅行经受住了业务流量暴增的考验。近日，InfoQ 专访了同程旅行研发中心云原生平台负责人彭涛，一探同程旅行云原生的改造之路。（本文节选自《中国技术先锋团队访谈录》2021 年第五季）

半年完成全部核心应用改造

2019 年，同程旅行主要面临两个问题。首先，由于刚和艺龙网完成公司主体合并不久，两个前身公司各自存在着不同技术体系的构建、发布等系统，这些系统随着公司业务的逐步整合，也必须在技术层面做进一步的收敛，以达到平台统一的目的。同时，在线旅行业务具有较明显的业务波动特性，在季度、节假日、每日时段上都有比较突出的波峰波谷特性。这样的业务特性对技术资源的整体利用率波动影响较大。

当时，企业服务云化成为业界技术发展较为主流的新趋势。基础服务的云化支持和业务应用的云原生改造成为技术团队较为紧迫的工作。2019 年底，同程旅行基于原有两个前身公司容器化基础上的云原生的二次改造工作正式启动。

2019 年底到 2020 年中的这半年里，技术团队定下的总体目标是提升集群资源利用率，降低资源使用成本。围绕这个目标，团队计划利用云原生思维重构部分技术体系，将多套旧有系统合并、收拢到一套以云原生应用为核心的私有云平台上，同时将 IDC、物理网络、虚拟网络、计算资源、存储资源等通过 IaaS 、PaaS 等，实现虚拟化封装、切割、再投产的自动化流程。

说起来似乎不难，但整个过程需要多个技术工种的支持和配合。

基础层面，为了支持 IaaS 层的网络虚拟化，运维人员选择了 Vxlan、大二层技术，并用 KVM 作为计算资源的切割。在容器网络虚拟化这部分，考虑到要降低损耗，采用了 BGP、Host 网络模式等技术，同时开发了绑核、NUMA 等相关技术。容器存储方面，远端存储选择了 Ceph，本地层使用块存储设备、NUMA 设备等。异构资源侧则采用了 GPU 改 CUDA library 的方式来完成虚拟化的切分和分时复用。技术团

队将资源调度变成了利用时序数据预测应用规模的方式，提升了资源利用率。

这一阶段的最大成果就是完成了有显著潮汐特征的订单服务的改造。通过改造，订单业务从原先独享机器集群切换到了共享机器集群，仅使用之前独享机器集群 40%的机器就完成了对全线服务业务的支撑，同时由于调度算法加入了自研的服务画像技术作为默认调度属性，资源调度的稳定性不降反升。

改造进行半年之后，同程旅行已实现纳入到该平台部分单机资源利用率提升了 20%，并通过云原生化的旧应用改造，下掉了当时集群内一半的服务器和相应的机房水电资源。

虽然看起来顺利，但是技术团队其实也面临着很多问题。比如在改造完成后服务部署时，团队发现大批量的物理机都出现负载上升的情况，经过排查发现是低版本的 Java 程序无法准确识别容器里的规格，导致 GC 时频繁发生资源争抢。

“当时正在做业务迁移，这个问题导致部分业务的稳定性感受不太好。”彭涛说道。由于无法确定其他语言是否会出现同样的问题，研发团队紧急开发了垂直扩缩容，确保 GC 可以使用更多的计算资源。另一方面，研发团队与业务团队一起进行了 JVM 版本升级，两项措施双管齐下，才没有耽误迁移进度。事后，研发团队还引入了隔离性较强的 Kata Container 来彻底解决该问题。

同程旅行云原生平台架构图，来源：同程旅行

流量突增，团队迎来大考

从 2020 年中到年底这段时间，可以认为是同程旅行云原生改造的第二个阶段。

第一阶段改造完成后，平台开始服务同程旅行的大部分在线业务。随着服务器集群规模的扩大，部分机器开始频繁出现故障。此时，保障服务稳定性成了技术团队的首要任务。

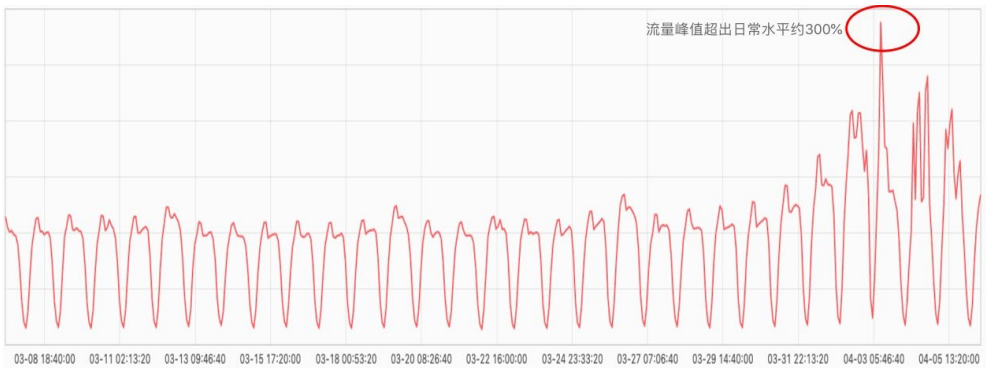
基于公有云、私有云和离线专属云集群等新型动态计算环境，同程旅行的技术团队帮助业务构建和运行具有弹性的云原生应用，促进业务团队开始使用声明式 API ，同时通过不可变基础设施、服务网格和容器服务，来构建容错性好、易于管理和观察的应

用系统，并结合平台可靠的自动化恢复、弹性计算来完成整个服务稳定性的提升。

具体工作上，技术团队将公有云的镜像预热、分发，专线直连内网机房，解决了内网集群需要镜像快速分发等问题，依赖的缓存资源和持久化数据实现了常驻云上，离线资源所在的专有云集群也同步被打通。同时，依托弹性计算能力，团队将集群间资源使用成本降到最低，并将最高服务稳定性的智能化调度平台的服务动态部署在多个集群上。针对业务专有需求和特殊，平台可以输出基础设施 API 和基础能力 API，供业务构建自己的云服务。

上述工作完成之后，原本用来应对季节性流量高峰期而采购的机器资源开始减少。通过判断服务当前冗余度来缩容线上服务的实例数，平台可以用最小的实例数量提供线上服务，而节省下来的资源可以提供给离线业务混合部署使用。

“在不额外新增机器的情况下额外获得的算力，成功支持了屡次创纪录的峰值流量。”彭涛说道。同时，团队开发的 Service Balance 系统可以在服务性能受损时自动尝试修复该节点性能，使得平台能够以较低的成本稳定运行。



同程旅行流量变化趋势图，来源：同程旅行

在 2020 年，同程旅行的部分业务跟进了直播方式。由于业务转变较快，应用架构尚未来得及调整，导致部分应用出现了明显的卡顿，影响到用户体验。为解决该问题，技术团队首先通过弹性计算改造为业务快速提供支持，之后又尝试了 Scale Zero 等方式，最终将该业务的资源使用量降到了之前常备资源的 20%。这也是在第二阶段中取得的重要成果。

得益于这次经历，弹性计算项目在同程旅行内部快速推广开来，企业核心业务的主要链路在三个月内就完成了云原生弹性计算接入，并借用弹性计算成功撑住爆款应用带来的日常流量 300%的峰值流量，也顶住了 2021 年上半年的屡次刷新公司峰值流量，为公司同类业务场景提供了坚实的技术支撑。

2021 年上半年，同程旅行进入到云原生改造的第三个阶段。团队通过基础组件、服务的云原生改造、服务依赖梳理和定义等方式，使应用不再需要考虑底层资源、机房、运行时间和供应商等因素。此外，同程旅行还利用标准的云原生应用模型，实现了服务的跨地域、跨云自动化灾备、自动部署，并向云原生场景下的 DevOps 演进。

考虑到五一出行和爆款产品带来的叠加流量，加上全球 GPU 资源供应紧张带来的影响，同程旅行将混合云改造计划提前。通过打通公有云的弹性容器集群，平台实现了 GPU 资源按需申请。通过使用混合云服务，同程旅行的基础资源使用成本和维护成本都进一步大幅降低。

目前，同程旅行核心业务均已完成了第二阶段的改造，部分小业务线还在持续推进中。改造完成的项目现在由平台控制在不同的机器、集群、云上进行着符合预期的流转和动态的部署。团队还完成了对机器数量较多的两个离线集群云原生改造。

同程旅行容器化专项组获年度最佳收益项目奖，来源：同程旅行

面对云原生改造过程中可能存在的新旧架构并存情况，彭涛给出了以下五条建议：

1. 所有的迁移改造方案都要有对应的回滚方案，这样业务才可以大胆尝试；
2. 新老架构转换要努力做到一键化和无感化，产品、功能等提前培训；
3. 研发团队要通过直接面对平台用户、自己先用自己的平台等方式，尽快发现问题。提需求、开发、测试、上线交付的各个时间点都要明确。事后要回访，响应每个用户的意见；
4. 研发团队需要积极参与平台资源使用的整个生命周期，降低平台的使用复杂度；
5. 提前做好沟通。各业务的服务观察周期不同，可以利用时间空隙进行服务的新老架构切换。

目前，同程旅行的云原生改造计划仍在持续深入进行。技术团队计划在明年实施存储类基础服务的云原生化改造，借助云平台提供的声明式接口加速软件交付流程，同时将业务代码下沉至函数级，运维能力由集成平台统一提供。

“这个过程的侧重点在于人效的提升。我们感觉比较好的落地方向是前端和机器学习。结合之前的经验，技术团队可以构建一个拖拽式的特定业务场景组件和界面编程平台，通过简单的编码就可以构建出整体服务的 pipeline，相关组件的运维能够对接到统一的平台运行。”彭涛解释道。

除了推进内部改造外，同程旅行也计划提供一些行业专属的云原生服务，推动整个上下游链路完成技术升级，未来也计划将一些项目开源，回馈给社区。

“积极参与到这次技术变革中”

在彭涛看来，云原生改造给团队带来的最大影响就是，原来权责清晰的研发、测试、运维的工作定位，开始变得模糊起来。

“三个部门都开始面向平台工作，三者之间的沟通协作也借由平台变得更加稳定和高效。”彭涛说道。从运维侧看，自动化处理占比开始增加，集群资源利用率提升的同时，稳定性并没有下降。从研发侧看，迭代交付速度提升，基础组件和基础服务可以通过 API 的方式调用，单体应用很方便就能拆解成小的微服务。团队开始借助云原生实现 DevOps，从而促使开发部门和运维部门建立密切协作，应用代码开发完成后能够非常快速、顺畅地转入生产。

云原生是一种构建和运行应用程序的整套技术体系和方法论。彭涛认为，云原生服务是希望研发人员能够关注自己的业务。从定义来看，云原生应用是通过标准化应用和服务构建容错性好、易于管理和便于观察的松耦合系统。结合可靠的自动化手段，云原生技术使研发能够轻松地对系统作出频繁和可预测的重大变更。如果云原生改造之后，研发人员需要关注的东西变多或者需要配置的东西更多了，那么就有点本末倒置了，因为复杂配置的背后不只是代码不过是另一套代码。

“以后的程序员也许真的要面向云原生应用编程了。”彭涛感叹道。

“早期，大家自己写代码部署到虚拟机上，容器服务带来了更轻量的操作系统打包和分发方式，提升了软件的交付效率，再结合大量的自动化组件，使 DevOps、线上部署和运维效率倍增，应用移植性变得更强。目前流行的无服务器计算更是让用户无需管理和运维底层的计算资源。这种趋势下，程序员都需要将自己的思维转变成面向云原生应用编程，继而改变自己负责的服务。”

云原生技术带来的技术跨代，相比之前单纯满足业务需求开发工作，能够更加成系统化地解决用户痛点。不过彭涛也指出，云原生作为一种新的思维方式和编程理念，目前缺乏比较好的引导方式。如果一家公司希望进行云原生改造，首先就需要上下层达成思想上的共识。

当前时代的发展带来了越来越多的变化和机会，很多企业业务的更新速度已经从以“周”为单位提升至按“小时”计。在这种情况下，云原生带来的极致交付体验是值得大家花费时间和精力去学习、去实践的。云原生作为新生事物，虽然学习成本不是很高，但带来的变化却是巨大的。

“大家可以积极地参与到这次的技术变革中。”彭涛说道。

嘉宾简介

彭涛，同程艺龙架构师，2019 年加入艺龙，目前在研发中心负责资源调度相关工作，包括容器化平台建设、弹性扩缩容、离在线混部、GPU 隔离调度、弹性扩所容、虚拟网络等，尤其在推动业务落地有丰富的实战经验，主导设计的容器平台 Furt 已成为公司推动的主流产品，各项技术指标达到业内领先，已经受过相当规模的业务落地检验。在容器化方向有丰富的理论和实战经验，过去曾就职于百度基础架构部、新浪微博研发中心，参与开发和设计了百度公有云虚拟化网络、以及微博峰值流量与热点应对。Kubernetes 代码贡献者，Flink 代码贡献者。

存储正在经历新一轮架构革命：戴尔科技集团在数据湖的探索和思考

采访嘉宾：孙骜、滕昱、雷璐

编辑：Tina

数据湖仓赛道上，正在批量跑出黑马。大数据初创公司 Databricks 于今年宣布获得 16 亿美元的融资，资金将主要用于加速湖仓一体的产品创新和市场开拓。2012 年才成立的 Snowflake，去年一上市就来个惊喜，IPO 筹资 38 亿美元，创近 12 年来最大 IPO 金额……

自从 Snowflake 上市以来，我们就被各种讨论所淹没，无论是投资者还是技术人，大家都希望能更好地了解这个赛道上的机会。但从技术上说，无论是数据仓库、数据湖还是 S3 存储，他们都是数据存储的一种方式，所以从某种程度上来说，存储行业正在迅速演变为数据业务行业。

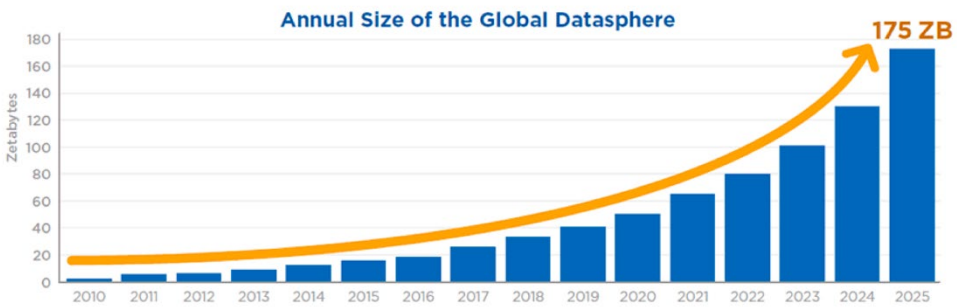
那么从存储角度来说，目前的这些数据平台公司还有着哪些短板？从技术方面来讲，大家还面临什么样的挑战？戴尔科技 OSA（Object, Stream and Analytics）研发团队讲述了他们的思考和探索。

InfoQ：国内外众多企业均在探索数据湖，在您看来，为什么现在数据湖会成为新的热点？

滕昱：随着企业数字化转型带来的数据量的爆炸性增长，数据在企业运营中的地位也

在日渐上升，很多人提出了数据资本（data capital）和数据资产（data assets）的概念来形容数据对企业的重要性。

Figure 1 – Annual Size of the Global Datasphere



Source: Data Age 2025, sponsored by Seagate with data from IDC Global DataSphere, Nov 2018

伴随而来的问题是，数据体量的增长远远超过了数据处理能力的提升。如何处理这堆不断增加的数据呢？这些数据在未来可能会有用，因此与其因为无法足够快地处理它而将其扔掉，不如先把它们存储下来，留待以后使用。因为你永远不能预测现在看似无用的内容是否在未来可以挖掘出新的价值，而数据湖恰恰提供了这样一种可能。

数据湖的概念兴起于 2006 年，它所应对的挑战是随着数据规模的爆炸性增长，传统的中心化存储系统已是不堪重负，而且过于昂贵。数据湖有以下几个特点。第一是便宜，相对于传统的专用存储系统，数据湖一般是部署在通用型服务器之上的分布式系统，在价格上有很大优势。第二是先存储，后消费。数据以原始格式存储于数据湖之中，而只有在被访问到的时候才进行组织和规整化处理（read on schema）。第三是容量大、扩展性强。数据湖提供了统一的数据存储系统。和各个部门单独运营各自的存储系统，从而在一个企业内形成若干大大小小的数据孤岛不同，数据湖提供了这

样一种解决方案，能够使得企业各个部门的数据都统一存放于一套系统之中，为之后的跨部门数据访问和数据分析业务提供了便利。

InfoQ：数据湖和数据仓库都是有着十年以上历史的概念，在云计算和开源的推动下（新的云计算的背景下），数据湖和数据仓库获得了哪些新的含义？有哪些突破性技术？

孙骜：在云计算的背景下，计算存储相分离的设计概念逐渐清晰，促进了现代数据湖和数据仓库的架构在数据存储和数据消费端的进一步解耦以及业界标准接口的规范化，这使得开源社区通过这些标准接口贡献新技术的发展成为可能。同时，公有云计算平台的出现，某种程度上加速了数据的垄断和计算需求的集中，推动业界对于数据以及数据处理做出更明确的需求定义，针对性地投入开源项目，以社区这种更灵活开放的方式促进技术发展，再反哺公有平台的进化和发展。由云计算和开源合作推动相辅相成，才孵化出一系列有利于数据湖和数据仓库发展的突破性技术。

列式存储相较于传统数据库的行式存储能加速 OLAP 工作负载的性能，这已经是众所周知的事实。但在我们看来，更加革命性的变化是列式存储格式的标准化。Parquet 和 ORC 的列式存储格式都是 2013 年发明的，随着时间的推移，它们已经被接受为业界通用的列式存储格式。数据是有惯性的，要对数据进行迁移和格式转换都需要算力来克服惯性。而数据的标准化格式意味着用户不再被某一特定的 OLAP 系统所绑定（locked in），而是可以根据需要，选择最合适的引擎来处理自己的数据。

第二大突破性技术是分布式查询引擎的出现，如 SparkSQL、Presto 等。随着数据存储由中心式向分布式演进，如何在分布式系统之上提供快速高效的查询功能成为一大挑战，而众多 MPP 架构的查询引擎的出现很好地解决了这个问题。SQL 查询不再

是传统数据库或者数据仓库的独门秘籍。

在解决了分布式查询的问题之后，下一个问题是，对于存储于数据湖中的数据，很多是非结构化的和半结构化的，如何对它们进行有效地组织和查询呢？在 2016 到 2017 年之间，Delta Lake、Iceberg、Hudi 相继诞生。这些类似的产品在相近的时间同时出现，表明它们都解决了业界所亟需解决的问题。这个问题就是，传统数据湖是为大数据、大数据集而构建的，它不擅长进行真正快速的 SQL 查询，并没有提供有效的方法将数据组织成表的结构。由此，在缺乏有效的数据组织和查询能力的情况下，数据湖就很容易变成数据沼泽（data swamp）。

InfoQ：您怎么评价 Databricks 和 Snowflake 这两家头部公司？他们为什么能够获得成功？

滕昱：Databricks 和 Snowflake 都抓住了 OLAP 的数据分析场景，基于兴起的云技术在数据存储和数据消费之间构建了新的中间数据抽象层（Data Virtualization），即屏蔽了底层系统的异构性，又提供了远超 Hadoop 生态系统的用户体验。这是他们能够成功的根本原因。或者用一句话描述，就是利用公有云的 infrastructure，来做自己的“cloud on cloud”商业模式。

如果仔细了解一下 Databricks 和 Snowflake 的发展历程，可以发现两者的出发点有所不同。Databricks 是立足于数据湖，进行了向数据仓库方向的演化，提出了湖仓一体的理念；而 Snowflake 在创建之初就是为了提供现代版的数据仓库，近些年来也开始引入数据湖的概念，但本质上说它提供的还是一个数据仓库。

我们先说说 Snowflake。数据仓库的概念最早出现在上世纪 80 年代。随着数据在企

业运营中变得日益重要，企业开始愈加依赖数据来做出关键业务决策。结构化的数据被存放在一个中心化的存储系统中并且对外提供查询功能，这就是传统的数据仓库模式，一般以定制化软硬件的方式提供给客户。Snowflake 利用云技术革新了传统数据仓库。它提供了一个基于公有云的、完全托管的数据仓库，把传统的软硬件一体的消费模式改造为了软件服务的模式（Software as a Service）。无论是存储还是计算，Snowflake 都利用了公有云提供的基础设施，从而使任何人都可以在云端使用数据仓库服务。

而另一方面，Databricks 从数据湖方向发力。传统的数据湖在数据分析上存在不足，像我们之前说到的，不能很好地提供 OLAP 场景的支持。因此，Databricks 通过 Delta Lake 提供的表结构和 Spark 提供的计算引擎，构建了一套完整的基于数据湖的 OLAP 解决方案。Databricks 的远景是基于数据湖提供包括 AI 和 BI 在内的企业数据分析业务的一站式解决方案。与 Snowflake 相似的是，Databricks 也充分利用了云基础架构提供的存储和计算服务，在其上构建了入门成本低、定价随使用而弹性扩展的软件服务方案。

InfoQ：为什么越来越多的数据平台类 startup 公司选择采用对象存储作为存储的核心？对象存储本身在这些年经历了什么样的发展？

孙骜：首先是实际上的考虑，从头开始搭建一个分布式存储是困难的，其中的坑只有踩过的人才知道。所谓“计算出了问题大不了 retry，而数据出了问题则是真出了问题”。

所以很多数据平台类创业公司如 Databricks、Snowflake 等都会借着计算存储分离的趋势，选择公有云提供的存储服务作为它们的数据和元数据存储，而公有云上最通用的分布式存储就是对象存储。例如 Databricks 的论文标题就是“Delta Lake:

High Performance ACID Table Storage over Cloud Object Stores”。

从技术角度来说，首先，对象存储即为非结构化存储，数据以原始对象的形式存在。这点贴合数据湖对于先存储原始数据，再读取完整数据信息后续分析的要求。

其次，对象存储拥有更先进的分布式系统架构，在可扩展性和跨站点部署上，比传统存储更具优势。由于对象存储简化了文件系统中的一些特性，没有原生的层级目录树结构，对象之间几乎没有关联性，因此对象存储的元数据设计能更为简单，能够提供更好的扩展性。此外，数据湖业务往往也需要底层存储提供多站点备份和访问的功能，而绝大部分对象存储原生支持多站点部署。通常用户只要配置数据的复制规则，对象存储就会建立起互联的通道，将增量和/或存量数据进行同步。对于配置了规则的数据，你可以在其中任何一个站点进行访问，由于跨站点的数据具备最终一致性，在有限可预期的时间内，用户会获取到最新的数据。AWS 在 Storage Day 上刚刚宣布的“S3 Multi-Region Access Points”就是一例。

第三，在协议层面，由 AWS 提出的 S3 协议已经是对象存储事实上的通用协议，这个协议在设计之初就考虑到了云存储的场景，可以说对象存储在协议层就是云原生的协议。对象存储还能兼容已有文件存储接口（Hadoop 生态常用的 s3a），在数据接口的选择和使用上更具灵活性。

第四，对象存储本身就是应云存储而生，一开始起家的用户场景即为二级存储备份场景，本身就具备了低价的特性。

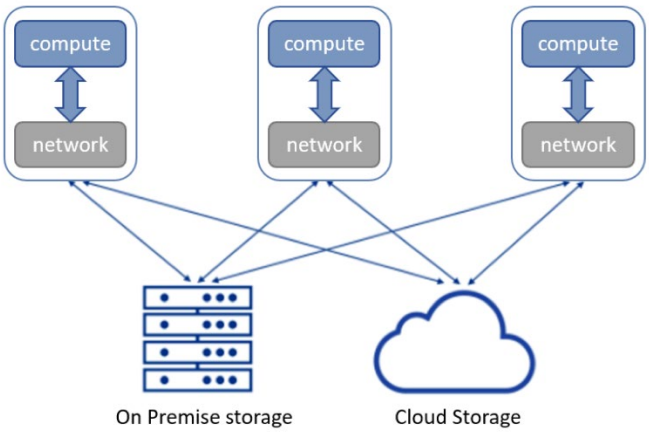
因此，对象存储是云时代的产物，支持原始数据存储、分布式可扩展、高灵活性、低价，都是对象存储之所以被选择的原因。可以预计在未来会有更多的数据业务完全基

于对象存储而构建。

InfoQ：从存储角度来说，目前的这些数据平台公司还有着哪些短板？对象存储还面临哪些挑战？

孙骜：我们发现市场上的很多数据平台公司还是利用 s3a 协议而非原生的 s3 协议来访问对象存储的服务。例如 Databricks 虽然在他们的论文中提到他们在对象存储之上提供了一层表结构的服务，即 Delta Lake。但实际上 Delta Lake 与底层对象存储交互的协议仍然用的是 s3a，即对象存储版本的 HDFS 协议。但是，从对象存储的角度来看，s3a 协议由于多了一层协议转换，并不能完全发挥高性能对象存储的延时和吞吐量的优势。因此，我们认为基于原生 s3 协议构建数据平台的存储访问层应该能大大提升存储层的访问能力。

同时，对象存储的部署方式也需要从单一的公有云变成混合云/多云的混合架构。“Latency matters”，如何在尽可能接近数据产生端进行实时、近实时处理是下一代数据湖需要直面的问题。更灵活的架构也会帮助企业级用户打消对于数据平台锁定的担忧（locked in/lock down）。



新一代数据平台的基本架构都是存算分离，即计算层和存储层是松耦合的。计算层无状态，所有的数据、元数据以及计算产生的中间数据都会存储于存储层之中。这一架构的优势包括更好的扩展性（计算、存储独立扩展），更好的可用性（计算层的失效不影响存储，因此能够很快恢复），以及更低的成本。为了适应存算分离的架构，对象存储本身也需要进一步发展。

在这个发展趋势中我们的产品研发也经历了从软件定义（Software Defined）到云原生（Cloud Native）的改造，来适应数据平台的需要。尽管我们从一开始就使用了 Scale-out 架构，基于标准 x86 服务器构建，所以在一开始我们认为自己已经做到了软件定义。但是在实践中，我们发现部署在第三方服务器上的存储产品还有复杂的运维问题、动态扩展问题等等，带来的用户体验并不完美。与此同时，以容器、Kubernetes 为代表的技术的出现，颠覆了原来的软件定义存储，可以让存储做到云原生，从提供存储产品转型为提供存储服务（Storage as a Service）。Kubernetes 提供了强大的集群管理功能，而容器提供了低开销的资源隔离和控制方

案。借助于这两大技术，我们彻底革新了对象存储的技术栈，在对象存储内部也实现了存算分离，用微服务的方式拆分巨型单体服务，例如元数据服务可以根据计算负载单独扩容，而不受限于底层存储。改造之后，对象存储本身也是 Kubernetes 所托管的诸多服务之一，只要集群中有足够的计算和存储资源，对象存储就能根据 IO 负载来动态地 Scale up 或者 Scale down，达到完全的弹性，而这也是云原生存储的最终目标之一。

所以我们认为，**想要适应存算分离的大趋势，不是简单地把现有存储对接到计算层就可以完成的，存储本身要经历新一轮架构革命才能更好地服务于计算层**。

在架构之外，数据平台型业务也给对象存储的特性提出了若干新的挑战。我们试举一二。

第一个挑战是数据分析型业务所需要的性能要远高于数据备份的场景，对象存储需要能够提供与计算需求相匹配的大带宽与低延时。NVMe 特别是 NVMe over Fabrics 技术的发展使得高性能对象存储成为可能。在选择 Fabrics 的时候，我们建议选择基于以太网的 RoCE 或者 TCP 协议作为网络传输层，能同时兼顾性能、成本和部署的灵活性。同时，要完全发挥 NVMe 的高性能，也需要对数据链路做彻底的异步化和无锁化改造。我们倾向于选择用户态编程模型，因为它对内核的依赖小，对云原生环境下的复杂部署更友好。另一方面，对象存储还需要根据业务场景来优化性能。例如，高带宽指的不仅仅是性能测试时能达到的带宽，因为传统性能测试往往使用 GB 级别甚至更大的对象，达到高带宽相对容易。然而，在数据分析的场景中，所要处理的是中等或者较大的对象（32Kb -1024Kb 级别），这就需要依据数据分析的业务特征来同时优化对象存储的带宽和延迟。

第二个挑战来自于数据分析所包含的众多元数据操作。因此对象存储不仅要能够提供大带宽，还要在处理小对象和元数据操作如 list 时提供足够的性能。这就比较考验对象存储的元数据管理能力。根据我们的经验，基于 B-tree 的某些高效变种来存储元数据在写放大、读延时等方面依然要优于 LSM tree。

第三个挑战是对象存储如何兼顾性能和成本。数据湖中存储了庞大的企业数据，但在任一时间点，可能只有一小部分数据是被数据分析业务所需要的。如果所有数据都放在性能最优的物理介质上（比如非易失性内存），那么成本将变得过高，失去了云存储的经济性，而如果在对象存储的前端再加一层 cache 层，无疑也会增加整个系统的复杂度。因此如何有效识别冷热数据，并将它们分区放置是对象存储需要解决的问题。

InfoQ：您们团队在构建数据湖产品时，有一个什么样的决策流程？（为什么选择 Iceberg）

滕昱：现阶段比较成熟的在数据湖之上提供表结构的开源产品是 Delta Lake、Iceberg 和 Hudi。对这三款产品我们都做了一些预研和实验。Hudi 基于 Hadoop 生态系统，特别是和 Hive 有深度绑定。Delta Lake 的设计非常优秀，不过由于是 Databricks 的产品，它还有一个不开源的商业版，许多高级特性只有在商业版上才提供。同样由于 Datatricks 的关系，在计算层上，Delta Lake 和 Apache Spark 深度绑定。与前两者相比，Iceberg 的接口抽象特别清晰，其绝佳的灵活性使得我们在进行对象存储产品研发时可以对其进行完美的适配，并且在计算层也可以跟 Apache Flink、Apache Spark 等计算层解决方案无缝集成。

因此，作为长期企业级对象存储解决方案的提供者，我们在对这些开源方案进行对比研究后，选择了 Apache Iceberg 作为我们的解决方案组成部分。当然，不排除未

来我们将和更多的开源产品集成，为客户提供完整的解决方案。

InfoQ：您们对 Iceberg 做了哪些改进?

孙骜：为了更好地适配底层的对象存储，我们为 Iceberg 做了一个通用的 S3 表管理组件（S3 Catalog）。

在 S3 的标准 API 中，上传数据需要预先知道对象的大小，因此在追加上传的场景下，其调用方法无法像 HDFS 那样简洁。所以在具体实现中，追加写的操作需要在本地预先处理，并以整体上传。第二，我们利用第三方锁的实现，解决了对象存储中不支持原子 Rename 操作的挑战。

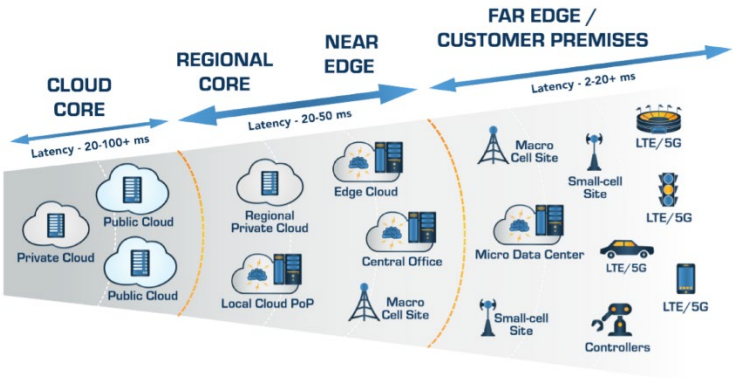
除此之外，我们还提供了一个基于 Dell EMC 商业对象存储 ECS 的专属表管理组件（ECS Catalog）。ECS 支持 Append 语义，使用 Append 的操作可以完美应对顺序写入未知长度文件的场景。ECS 还支持类 CAS 语义。在并发提交的场景下，ECS 支持使用 If-Match 和 If-None-Match 对对象进行 CAS 操作，来实现原子化 rename 的操作。

更多细节可以参考我们之前发表的一篇文章：《HDFS 廉颇老矣？基于对象存储的数据湖构建新思路》。这个改动已提交给 Iceberg community review，希望近期 merge 回下一个主版本。

InfoQ：您们如何展望未来的下一代数据平台?

滕昱：在我们看来，下一代数据平台有更广阔的范畴和视角，超越当前人们对数据湖的理解。它将不光涵盖传统的数据中心或者云端部署的范畴，结合日益发展的边缘计

算，将来的数据平台是会涵盖从传感器到边缘节点到数据中心到云端一整套生态系统。



Source: VMware 调研白皮书《基于全栈云基础架构，构建企业级业务创新平台》

在这套生态系统上，数据是主角。平台的使用者会慢慢弱化对底层系统的依赖，而更多地关注于数据处理管道和基于业务逻辑的数据处理逻辑需求的定义。基于此，平台对于使用者的专业性要求也会逐渐降低，越来越多的人能够利用数据平台从数据中获取信息价值。

存算分离将是下一代数据平台的标准架构。存储层将更统一，而计算层根据负载也将拥有更多的灵活性。和传统数据湖相似，数据依旧会统一存储在数据湖中，先存储后消费。但是，对于数据的结构化定义要求会越来越不重要，数据清洗或结构化的转化将更多地由平台智能化或者根据更人性化的配置定义完成。至于数据在各个环节中的存储，也将隐藏在平台内部。数据孤岛将逐渐消亡，原因很简单，如果数据湖本身能够提供足够好的 OLAP 支持，为什么还要把数据拷贝一份再放到数据仓库中呢?

在计算层面，数据平台应该是完全开放的。根据具体的业务需要，用户可以自行选择

灵活度更高的 Spark 或者 Flink 计算框架，或者是集成度更好的一体化方案。下一代数据平台也应该提供强大的跨表查询能力。无论数据是直接存储在对象存储中、存储在 Iceberg 等表结构中、还是存储在外部的数据库中，数据平台都支持对这些表进行联合查询。

存储和计算之间会进化出新的数据抽象层（现在正在发生）。更多统一的业界标准如 Iceberg 等，会越来越多地出现。数据产生者和数据消费者基于这层抽象层协同合作。通过这层数据抽象层，数据平台会慢慢将各个角色的数据消费者从系统部署与理解的细节中解脱出来，以关注业务逻辑本身。正如 Iceberg 创始人之一 Ryan Blue 所说，数据平台的使用者应该更多关心自己的数据和数据产品，把剩下的问题交给数据平台基础架构去解决。

同时从应用场景上来说，在传统的离线数据分析场景之外，实时数据分析的业务场景正在增加。以 Spark micro batch 为代表的近实时框架可以解决一部分的业务需求，但对于延时要求更高的场景，实时分析的框架还有待发展。以流式存储为依托的实时分析框架依然是个尚待开发的领域。另外，目前的数据平台对结构化和半结构化数据已经有了比较好的支持，但对于非结构化数据，如何为它们建立高效的索引来加速查询和分析，依然是业界悬而未决的难题。

总结一下，在我们看来，未来的数据平台应该是涵盖传感器、边缘节点、数据中心和云端，存储计算相分离，依赖数据抽象层，为多种多样的数据产生和消费者，提供集成了离线分析、近实时分析和实时分析数据处理链路的统一平台，提供更简单和标准化的接口供用户使用，让用户聚焦于从数据中获取价值。

嘉宾简介

滕昱，Pravega 中国社区创始人，戴尔科技集团 OSA（Object, stream and analysis）软件开发总监。自 2007 年加入戴尔科技集团中国研发集团以来，一直专注于分布式系统领域，先后参与了前后 2 代 Dell EMC 对象存储产品的研发及商业化工作，正在积极拥抱新的混合云/多云和实时流处理时代。 从 2016 年开始，参与 Pravega 开源项目，活跃在多个开源社区。

孙骜，技术团队杰出成员（Distinguished Member of Technical Staff），负责戴尔科技集团分布式对象存储系统架构设计和开发。

雷璐，高级主管工程师（Senior Principal Engineer），专注于戴尔科技集团流处理平台 SDP（Streaming Data Platform）系统架构的设计和开发。

打破国外垄断，出门问问主导研发的端到端语音识别开源框架 WeNet 实践之路

作者：刘燕

采访嘉宾：张彬彬，出门问问 WeNet 项目负责人

> 端到端语音识别技术，如何更好的落地？

出门问问开源端到端语音识别框架 WeNet，star 数已超 1200

今年 2 月，中国人工智能公司出门问问联合西北工业大学推出了全球首个面向产品和工业界的端到端语音识别开源工具 —— WeNet。

在正式发布后短短 7 个月的时间里，WeNet 在 Github 上的 star 数已超过了 1200 个。

近日，出门问问 WeNet 项目负责人张彬彬接受了 InfoQ 专访，他详细介绍了 WeNet 研发创新的思考与实践。

据了解，2020 年 10 月，WeNet 在出门问问内部正式立项。当时，公司内部多位研发人员同时展开对端到端语音识别技术的研发探索。

在探索过程中发现，主流的端到端语音识别工具 ESPnet 并不能完全满足需求，ESPnet 在工程上难以产品化，也难以支持流式语音识别、语言模型等语音产品中的核心特性。

因此，出门问问决定走自研之路，打造出一款以产品化为核心的端到端语音识别工具。与此同时，也想借此机会将公司内端到端的研究工作加以整合，将技术成果沉淀下来，这也正是“WeNet”名字的来源，“We”寓意“共创”。

“WeNet 的研发过程其实也是‘摸着石头过河’，边实践，边总结，边提高”，最开始，只有张彬彬一个人在开发核心代码，他用一个月时间完成了框架设计，很快又有 2 位成员加入进来，组成‘三人小分队’。随后，他们联合了西北工业大学的音频语音与语言处理研究组来开发这款 WeNet。

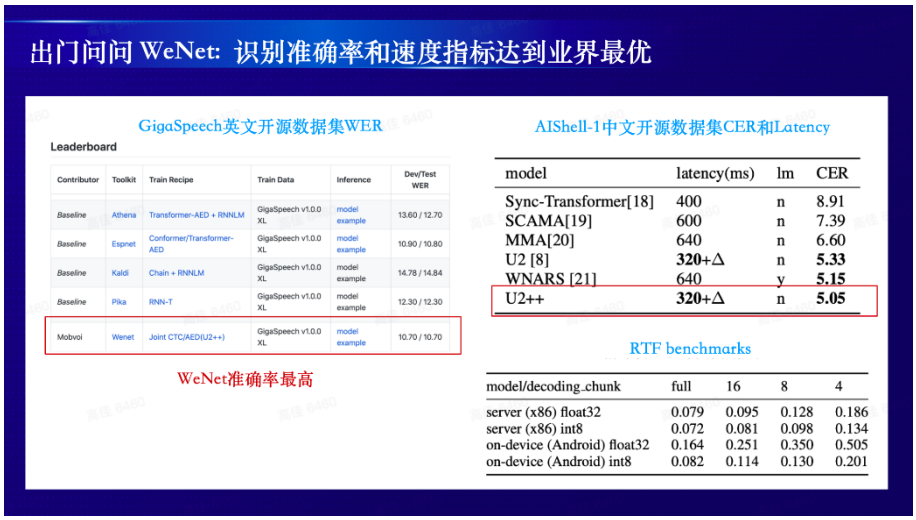
怎样做好流式的端到端语音识别是 WeNet 团队在研发过程中遇到的一个棘手问题。流式端到端语音识别是指在处理音频流的过程中，实时返回识别结果，延迟极低，对实时率要求高。为攻克这一难点，WeNet 团队首创了 U2 算法，经过反复试错、调优、实验，最终实现了良好的模型和识别效果。

项目启动 2 个月后，WeNet 即在 GitHub 上放出了部分代码。今年 2 月，WeNet 发布第一个正式版本 — WeNet 1.0 版本，WeNet 1.0 版本支持流式和非流式语音识别，支持云端 x86 和设备端 android 端的推理。

这时，WeNet 框架已相对完备，初步达成产品化目标，也收获了不少来自社区的正向反馈，于是团队决定将 WeNet 正式对外开源。

开源地址：https://github.com/wenet-e2e/wenet/

今年 6 月，WeNet 推出了 1.0.0 版本，该版本支持更多的数据集，解决了目前主流语音开源工具的痛点，且各项性能指标表现优异。



WeNet 使用业内前沿的深度学习模型结构 U2++，支持语言模型、endpoint、n-best、时间戳、提供数据量最大的中文和英文预训练模型等，在 Aishell-1，Aishell-2 和 GigaSpeech 上准确率达到 SOTA；推理方案支持 Android 平台和 x86 平台，支持基于 gRPC 和 WebSocket 的服务端推理和端侧推理。

接下来，WeNet 将按照“边开发、边开放”的节奏逐步开源。

目前，WeNet 团队正在规划下一个版本，张彬彬介绍，新功能主要会围绕三个核心点研发：

- 支持更多产品级、工业级特性。如超大规模数据 IO（10 万小时以上）、热词、关键

词检测、ITN，标点、标注错误检测等；

- 完善生态建设。包括文档和教程建设，进行国际化推广、开发者社群维护、获得更多公司、高校的支持，支持更多更广泛的数据集，目前正在进行数个对中文、英文、日文等标准数据集的支持；
- 更前沿模型的探索：在技术上，将探索更好的端到端模型、预训练模型、无监督训练等技术。

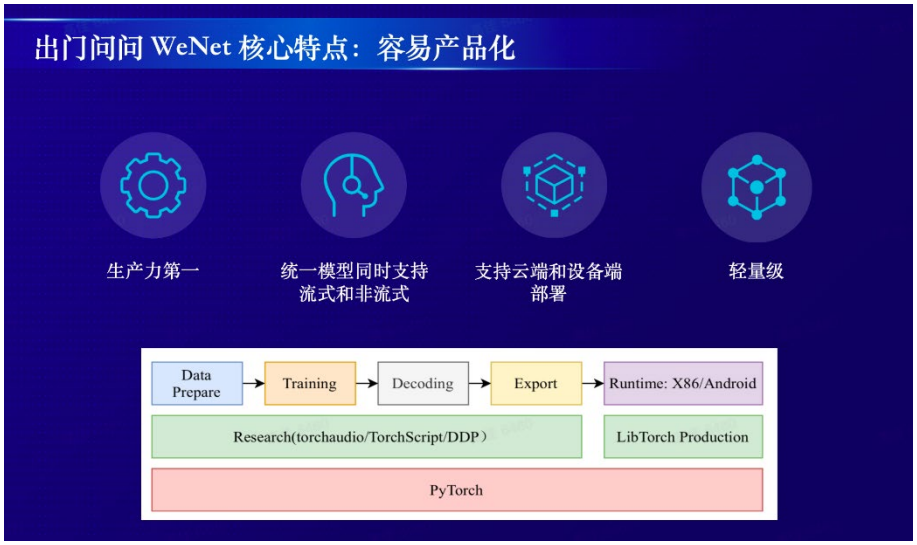
WeNet 核心特性：生产力第一

“我们希望 WeNet 成为国内和国际上最流行、最有生产力的智能语音工具”，出门问问对 WeNet 的未来充满期待。

如果用简单的几个词总结 WeNet 的特性，那就是“**更快、更高、更强、更有生产力**”。目前，WeNet 1.0 以其“小而精”的鲜明特色，已构建了一个完整完善的语音识别所需的方方面面的能力，且具有工业界应用的典型案例。

“生产力第一”是 WeNet 自诞生之初就确立的核心原则，其定位是面向产品和工业界。“WeNet 在设计之初、开源之前，就是为了落地端到端语音识别，解决语音识别在实际产品和应用中的实际问题”，张彬彬表示，“端到端语音识别产品在落地过程中存在的痛点和难点，多数是产品化的问题，把产品化做好才是关键”。

WeNet 的架构和特性也主要遵循“生产力至上”的原则而设计。



据介绍，WeNet 的核心算法是 U2 和 U2++，WeNet 1.0 中将 U2 算法升级到 U2++，U2++是当前全球最前沿的深度学习模型结构。使用 Conformer 网络结构和 CTC/attention loss 联合优化方法，先用 CTC 进行流式解码，再用 Attention Decoder 去重打分，进一步提高了识别准确率。

与企业相比，高校和研究机构做语音识别相关的模型和算法，无需过多考虑模型是否有应用场景以及能不能落地。在设计公开数据集上的性能时，也基本不用考虑是否是流式的，模型的参数量如何，是否能采用低成本的方式等。

但在企业不同。出门问问认为，一个算法、一个模型、一个产品和项目，如果不能在公司里面落地，可能毫无价值可言。

生产级语音识别系统的建设，对技术团队的能力提出了更高的要求，需要更深入的理解语音识别的场景和实际的产品诉求，以及更高标准的工程能力的要求，还要更专注

产品的特性，更克制。

不过，WeNet 虽是一款面向工业级产品的端到端语音识别解决方案，但因其简单、高效的特性，也被很多高校用来作为学习和科研工具。WeNet 在整体结构设计上属于轻量级的框架，安装、使用方便，这对于高校的研究者来说，便于快速上手。

WeNet 所具有的生产力第一、轻量级、准确率高等性能，对开发者群体十分友好，即便是在开发者已经使用其他语音识别框架的情况下，也能快速、安全、低成本的迁移到 WeNet 上来。

在迁移成本方面，WeNet 提供了模型训练、推理、预训练模型，如果用户想搭建语音系统，在出门问问的平台上下载预训练模型，再用推理的流程把模型给构思起来，整个过程约 10 分钟内就可以操作完，轻松获得专业级别、可以应用的语音能力。

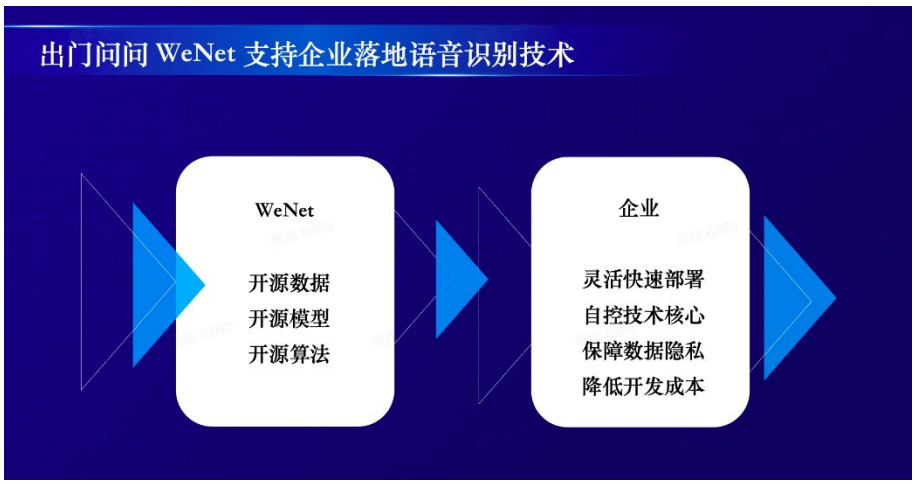
WeNet 还提供一站式的服务，传统的语音识别模型的研发分为模型研发、模型训练、模型部署三个阶段，每一步都有很多复杂冗长的工作要做，而通过 WeNet ，原来需要三步，三个人甚至三个团队去做的工作，现在靠这一个平台就解决了。

现在，WeNet 已经广泛应用到到出门问问内部的各个产品线，如车载、魔音工坊等 toB 项目。

出门问问也为企业使用 WeNet 部署语音识别提供商业化和技术支持。企业皆可基于自己的数据和服务体系，拥有私有化系统。

目前，WeNet 应用到了喜马拉雅、作业帮、京东、腾讯等数百家公司，他们采用 WeNet 构建自己语音服务，覆盖智能车载、智能家居、智能客服、音频内容生产、

直播、会议等语音识别应用场景。



在上述场景中，WeNet 都做到了更高水平的准确率。WeNet 被用户赞为“产品化集成度最好的框架”。WeNet 通过打造社区支撑、提供行业解决方案、落地私有部署等方面，为 AI 行业创建共享机制、建立生态。

端到端语音识别时代来临

“生产力第一”也是 WeNet 有别于其他语音识别工具的核心优势。

在 WeNet 早期，出门问问内部曾将其和主流的端到端语音识别框架 ESPnet 做过性能上的对比，WeNet 在易安装、易用性、产品化等方面表现要好于 ESPnet，其中在易用性上，可提供一键式训练脚本、预训练模型和多平台运行时工具。

相比 ESPnet，Wenet 没有对各类序列转化任务进行统一抽象，完全聚焦于语音识别任务，同时对常用的语音识别应用场景提出了一套端到端解决方案，而不是提供各类模型方案的大而全的集合。

自去年 12 月开始，出门问问内部全部迁移到了 WeNet 上，经验证开发效率得以大幅提升。如今，在 Github 的 Star 数量上，WeNet 也远超国内其他开源语音框架。

张彬彬也坦言，相比于其他的端到端语音识别工具，WeNet 现阶段还是一名“年轻选手”，WeNet 正式发布也就半年时间，在生态和数据集的建设方面还相对欠缺，“现在学术界有 50 个场景的数据集，我们只做了 5 个，数量还远远不够，未来会逐步补上来，目前正在借助社区的数量展开工作”。

近些年，随着深度学习技术的发展，语音识别技术经历了深刻的变革，从基于 DNN-HMM 的语音识别，到基于 CTC 的端对端语音识别再到基于基于 Attention 的端对端语音识别。

语音识别进入到“全民”端到端时代，已是毋庸置疑的趋势。学术界最早在 2014 年开始研究端到端语音识别技术，经过 7 年发展，该技术现在已经逐步趋向成熟。

端到端语音识别技术具有明显的优势，它大大简化了语音识别的流程，流程简化后，上手学习、应用的门槛都大大降低，同时还能实现非常好的语音识别效果。特别是近两三年，该技术已经在业界广泛的应用，落地，未来会有更多的行业应用涌现。

任何一项技术发展到一定阶段后都会面临瓶颈期，端到端语音技术现在面临的一个问题在于，端到端语音识别依赖平行语料数据，平行语料在低资源语言下的应用还需进一步探索，低资源的学习将是接下来的研究重点。

目前的深度学习依赖大量的语料及标注数据，预训练技术和无监督学习技术是当前语音领域研究的热点和难点。张彬彬判断，接下来 3-5 年，预训练技术和无监督学习技术将是发展趋势。如何使用海量的无监督数据学习，并进行模型的预训练，然后以低成本进行迁移到其他任务上，值得投入更多探索。在预训练技术上，出门问问很早就开始了语音预训练和自然语言处理 GPT-3 的研究，未来会持续在这块发力。

拥抱开源生态，打破依赖国外语音框架的长期垄断

WeNet 发展的每一步离不开开源社区的助力。WeNet 研发借鉴了 Espnet 、Pytorch 、Kaldi 等优秀的开源项目。如今发展渐至成熟的 WeNet 也选择开源开放来回馈社区，为语音开源生态出一分力。

张彬彬觉得，Wenet 开源最大的意义是降低了语音识别的门槛。“传统的语音识别技术，门槛高，需要专业的背景知识。门槛降低后，越来越多的人能够接触、从事、开发和应用语音识别，只有越来越多的人从事这个行业，这个行业才会发展的更快，也能更好、更快的赋能和产品化”。

他表示，Wenet 开源后也收到了一些圈内人士的关心，Wenet 通过开源的方式把语音识别的门槛降低了，但这是否对出门问问公司本身带来不好的影响？

思考许久后，他更坚定的觉得这是一个正确无比的决定。

“如果用发展和长远的眼光来看，只有语音行业的快速发展、进步和应用才能带来更

多的成长空间，我们更多的是追求全面的、发展的、生态的共赢，而不是片面的、孤立的、垄断的零和博弈。此外，WeNet 的成熟也能促使大家把重心放到打造真正的产品和应用上，而不是重复造轮子，只是处理各种琐碎和边边角角。”

如果从更宏观的维度看，WeNet 开源后，国产原创语音识别工具阵营又添一员“大将”。

我国产业界对开源语音框架依赖性较高。在很长一段时间里，我国语音识别领域所使用的工具和框架，多出自西方国家的企业或高校研发，被国外垄断开发。

最早期，在上世纪 90 年代到 2010 年代，传统的语音识别系统 HTK 是英国高校开发的。近 10 年里，最为流行的语音识别工具 Kaldi，是美国公司和高校开发的。最近三年，最主流的端到端语音识别工具 ESPnet 是美日高校和公司开发的。

现阶段，国内原创的语音识别工具还比较少，一些科技大公司虽开发了自己的工具，

但因涉及核心资产等因素，这些工具通常不会被开放出来，也未能得到广泛应用。

在这样的背景下，打造自主可控的语音开源工具已是箭在弦上。

出门问问认为，WeNet 是真正意义上的第一个国产并广泛流行和应用的语音识别框架，打破了西方国家在该领域的长期垄断，对实现真正的自主可控具有重要意义。出门问问希望通过探索和构建开源开放协作的共享机制，创建自主可控的语音开源工具，寻求国产 AI 技术的进一步突破。

采访嘉宾介绍

张彬彬，出门问问 WeNet 项目负责人，2018 年加入出门问问负责端到端语音识别系统的研发和落地，包括 WeNet 的开源推进，车载和 toB 项目等。2017 年硕士毕业于西北工业大学音频语音与语言处理研究组，曾在微软、百度、地平线等公司工作。

为了让你搞定数据库选型，这些工程师重写了 26 万行代码

作者：王一鹏

无论多么有主见的架构师，在做数据库选型的时候，也可能会犯难。

传统 SQL、NoSQL 还是 NewSQL？架构风格是以 久经考验的关系型数据库为主，还是偏向所谓原生的分布式架构？如果提及具体产品，那选择就更多了，TiDB、OceanBase、PolarDB、TDSQL、GaussDB、MongoDB…… 现在还有许多服务于新场景的产品，比如处理时序数据的 TDengine，处理图数据的 Nebula

Graph……以及最老派又最完善的 Oracle。

如果从业务场景或即将面临的迁移成本来看，问题会更加复杂。牵扯到底层数据的选型和架构设计，有时更像一锤子买卖，一旦定了某种方案，再想替换代价可不是一般的大。

差不多在 10 年前，这事儿还没有那么棘手。Oracle、IBM Db2，二选一而已。但今时不同往日，这是一个数据量急速膨胀、业务高度复杂的时代，真正让人焦虑的不是单纯的选型问题，而是将“降本提效”推向极致的数字化转型问题。

那么，除了咬牙硬选一个数据库，或者基于现有数据库的基础上自研一套存储方案，真的没有其他路可走了吗？其实也有，只不过在分布式数据库热度越来越高的当下，显得有些“透明”，那就是中间件 + 多商业数据库的解决方案。

看到这个答案，你可能会有些失望。中间件方案出现的时间，尚在分布式数据库成熟之前。因此在业内很多架构师看来，这种方案在技术上不够超前，只能算是某种“过渡策略”，本质上是 NewSQL 数据库成熟前的无奈之举。

但来自金融等行业的诸多落地实践证明，事实可能并非如此。

我们为此特别采访了 全球顶级开源项目、数据库中间件产品 Apache ShardingSphere 的作者，以及背后商业公司 SphereEx 的 CEO 张亮，他一直对分布式架构设计保持关注，曾是京东科技架构专家、当当架构部总监。**关于整个数据库行业的选型和架构设计问题，张亮有着特别的思考**。

可插拔架构，或许是答案

“需求是多元化的，一部分用户适合分布式数据库，一部分用户适合用数据库中间件，甚至还有一部分适合两种都用，没有太绝对的答案”，张亮说。

这是个无错的答案，但也可以得出一个推论：**如果场景是多元化的，数据库是多元化的，那么架构师应该尽量规避扩展性、兼容性不好的数据库解决方案**

无独有偶，Oracle ACE、数据库专家韩锋也在其个人公众号里发表过类似的观点：“（关于数据库选型）为了规避路线选择、厂商绑定的风险，比较现实的方法是选择一款兼容通用性协议的产品，并且在应用中仅使用标准数据库的用法。”

二者结合起来，我们能抽离出一些关键词：中立、兼容、标准，对于很多在选型问题上难以抉择的架构师来说，这让中间件路线看起来更加实际。

与数据库行业不同，以 ShardingSphere 为代表的中间件层由于不涉及存储引擎，如今已将目光从单纯的水平扩展问题转向业务支持和灵活性问题，也因此得以实现对异构数据库的统一管理。

灵活性、兼容性，是数据库中间件产品的核心，也是解决“数据库选型”问题的关键。据张亮透露，近两年的主要研发重点一直是可插拔架构，以将灵活性和兼容性推向极致。好消息是，ShardingSphere 的可插拔架构预计将在未来一段时间内正式上线。

所谓的可插拔架构，是指在架构层面，将整个系统分为基座和插件两部分，插件部分互相隔离、互不影响，基座可以自由接入多个插件。可插拔架构多见于相对轻量级的前端领域，属于微前端体系的一部分，但在基础软件部分则相当少见。

张亮认为，可插拔架构形式是未来数据库中间件的主要趋势之一，一则产品需要高度的灵活性，二则还有大量的能力需要被构建，比如数据安全、异构数据网关等功能，可插拔架构自然成为了产品核心。

话虽如此，但可插拔架构的设计难度却很大，让人望而生畏。这种设计难度，大致分可为两部分来谈：

第一，可插拔架构是对 OCP（Open-Closed Principle）原则的一次彻底执行，力图仅通过增加新模块来满足新需求，旧有模块完全保持 0 修改。这意味着，可插拔架构要清晰地定义出，什么是基座，什么是插件。它对上层、下层都无感知，一切面向接口。用张亮的话说，就是："完全面向一个抽象的、虚无的东西，不涉及任何的业务细节"。

比如，ShardingSphere 转向可插拔架构后，其核心流程里已经没有分片功能了，分片会作为可插拔能力的一部分接入到服务中。对于数据库中间件来说，几乎属于产品重定义。与许多人对数据库中间件的固有认知相悖，因为在许多人的理解中，数据库中间件不就是为了分库分表而存在的吗？

但实际情况是，**单体数据库的覆盖场景依然很多，分库分表并不是 0 级功能。这是在架构层面，必须具备的关键洞察**。

第二，与微服务相似，只要涉及服务拆分，就会涉及颗粒度问题。对于可插拔架构来说，需要插件化的不一定只是产品功能，比如两阶段强一致事务和柔性事务，也是能够实现可插拔的。

基于这些拆分问题，ShardingSphere 把可插拔架构分为三层，分别是内核层、功

能层、生态层，分别面向数据库内核、企业功能、数据库生态进行可插拔设计。其中，查询优化器、分布式事务引擎、调度引擎等是内核层的可插拔模块；数据分片、读写分离、数据库高可用、数据加密、影子库都是功能层的可插拔模块；数据库协议、SQL 方言等则是生态层的可插拔模块。

要实现可插拔架构，除了设计难度和颗粒度拆分，其工作量也令人叹为观止。ShardingSphere 有 190 多个模块，近 43 万行代码，核心 Java 代码 29 万行，张亮回忆道："为了做可插拔架构，老代码留了不到 1/10。"这意味着，有近 26 万行的核心代码被重写了。

可插拔架构是 ShardingSphere 追求灵活性最重要的标志之一，但它对灵活性的追求又不仅限于可插拔架构。

比如，ShardingSphere 还额外提供了两种部署形态，分别为 JDBC 和 Proxy。JDBC 是 Java 访问数据库的标准接口，Proxy 是中间件最常见的服务形式，且两者经常能够在同一环境下进行混用，以满足多用户下多类型访问的需求。

这样多种类型的服务接口，一方面服务了不同类型的开发人员，另一方面也实现了性能层面的可定制化，工程师可以结合场景调整数据库分片的键值，实现不同场景下，性能的最大化提升；反之，"全包式"数据库方案，则往往需要放弃部分灵活性，以相对中庸的方案来换取无感知、低侵入的使用体验，体现了与数据库中间件方案的差异性。

真正的开源项目，是社区说了算

如果我们要做技术选型，需要注意的另外一点是，备选产品的维护主体是谁，备选产品的基因是什么，是开源，还是闭源？这与搞清楚产品的技术方案、技术理念同样重要。

当下，无论开源的热度如何，大部分分布式中间件、分布式存储、数据库都是闭源的，这是不争的事实。

看到的是，大量的开源创业公司正在出现，资本也在快速进入，比如 SphereEx、欧若数网、Neo4j，以及大家熟知的 PingCAP。同时也有许多数据库宣布开放源代码，比如 OceanBase、Tendis、openGauss。

为什么在数据存储领域，开源这么引人关注?

一个可能的答案是，开源在技术层面的想象空间更大，对开发者更友好。就像 ShardingSphere 的可插拔架构，架构设计完成只是第一步，后续还有海量的不同模块的开发工作。对于创业公司来说，如果不借助社区的力量，美好的可插拔架构也可能成为公司的研发黑洞。

产品的中立性，也是导致开源项目集中迎来爆发的另一个要素。尤其是在数据存储领域，最美好的答案可能是无依赖、跨多云，最差的答案才是被单一产品强绑定。

当然，开源再好，也抵不过现实的骨感。开源两年，Star 几百，花钱不少，效果为零，这恐怕是众多开源项目的常态。社区的健康程度，往往直接定义了开源项目的生死，这导致即便架构师想做选型，也没有太多的好选择。

在张亮看来，一个开源项目能不能成功，大致可以分为三个维度来考察：

第一，能否耐的住寂寞，团队是真的相信开源，还是拿开源当做商业上的捷径。一个最简单的考核指标便是运营时间，“开源项目一定要度过‘静默期’才会迎来爆发，只做了半年、一年，是没法预估项目未来的”，张亮说。

第二，一些必备的运营技巧。张亮一方面把 ShardingSphere 捐献给 Apache 基金会，另一方面也带着项目参与了许多活动，比如谷歌举办的黑客马拉松、编程夏令营，除国内用户以外，也吸引了大批的海外开发者、学生参与到社区建设中来。

第三，观念的转变，也是最关键的部分。从小处着眼，是从“自己开发”到“社区开发”；从大处着眼，就是在真正意义上拥抱开源，而不只是嘴上说说。

“ShardingSphere 的项目发展是受社区的引导。比如说，社区认为 ShardingSphere 该做基于影子库的压测和可观察性，ShardingSphere 就真的做了。这些都不是项目自上而下的设计，只要需求爆发，且在项目的 Scope 内，就可以实现。但如果一个公司在运营开源项目时，遇见所谓的偏离主线设计的社区诉求，就拒掉它，那么大概率也会影响这个项目的成长，因为它不算真正扎根社区的开源项目。”

未来的发展方向

目前相关的中间件产品，还是把核心聚焦在水平分片、弹性迁移和 MySQL 实例管理上。

但某种程度上，ShardingSphere 可能代表了未来数据库中间件发展的核心方向之一，即 0 级功能是可插拔，1 级功能才是数据分片。开源和可插拔架构结合在一起，等于打开了数据库中间件在技术和产品维度的想象空间。张亮透露，SQL 审计、基于数据的权限引擎、多租户、TTL（Time To Live）都会被提上开发日程。

除此之外，ShardingSphere 还有一个正在开发中的构想，叫做 Database Mesh，力争实现数据库上云的原生体验，但还需要一定的开发周期。

Database Mesh 会在数据库集群之上，封装一层代理，做智能的负载均衡。传统的负载层无法识别 SQL 特征，只能用轮询或权重的方式透传。但 Database Mesh 会根据不同的 SQL，匹配计算实例的标签，更加智能选择要访问的数据库计算或存储节点。

对于架构师而言，最重要的是打开技术选型的眼界与想象力。分布式数据库对业务的侵入性更低，但中间件方案规避了对厂商的依赖问题，究竟如何选择，要以实际场景为判断依据。但这并不妨碍我们给出阶段性的推论：

很可能，在未来的 5 – 10 年间，数据库中间件都是底层架构最重要的解决方案之一，值得每一个架构师认真调研。

扫码关注InfoQ公众号

Geekbang. 极客邦科技 | InfoQ