

# 架构师

ARCHITECT



## 热点 | Hot

Dubbo正式进入Apache孵化器

## 推荐文章 | Article

Apache Kylin在客户画像系统中的实践

## 观点 | Opinion

批处理ETL已经消亡，

Apache Kafka才是数据处理的未来吗

## 专题 | Topic

微服务架构技术栈选型手册



# CONTENTS / 目录

## 热点 | Hot

Dubbo 正式进入 Apache 孵化器，开启开源新时代

## 观点 | Opinion

批处理 ETL 已经消亡，Apache Kafka 才是数据处理的未来吗？

## 推荐文章 | Article

Apache Kylin 在绿城客户画像系统中的实践

## 专题 | Topic

微服务架构技术栈选型手册

## 特别专栏 | Column

云数据库 UDB 的三重境界



## 架构师 2018 年 3 月刊

本期主编 陈思

提供反馈 [feedback@cn.infoq.com](mailto:feedback@cn.infoq.com)

流程编辑 丁晓昀

商务合作 [hezuo@geekbang.org](mailto:hezuo@geekbang.org)

发行人 霍泰稳

内容合作 [editors@cn.infoq.com](mailto:editors@cn.infoq.com)

# 卷首语

## 人工智能时代什么最贵？人才！

作者 陈思

还记得电影《天下无贼》中葛优扮演的黎叔那句经典台词吗：21世纪什么最贵？人才！

中国的 AI 市场发展速度相当惊人。根据美中经济与安全评估委员会在其 2017 年的年度报告当中所指出，“中国政府已经承诺在人工智能领域投入超过 70 亿美元资金，而以深圳为代表的多个城市亦为人工智能初创企业提供 100 万美元启动支持。相比之下，美国联邦政府于 2015 年投入 11 亿美元用于未明确分类的人工智能研究，且主要以竞争性补助形式发放。”

人工智能领域专家吴恩达在采访当中所提到，“中国对英语世界所发生的一切都拥有着相当深刻的认识，但西方对中国的了解则恰恰相反”。虽然中国研究人员能够用英语进行表达并涉足西方的研究领域，但由于语言障碍的存在，英语社区与中国的研究方向仍然存在着巨大差异。正因为如此，中国在西方世界的视野之外取得了巨大的学术飞跃。

先来看一组数据：

根据 InfoQ 旗下人工智能公众账号 AI 前线的调查：2000~2016 年期间，

中国的人工智能企业累计增长约为 1470 余家，仅北京市，到 2016 年就有 450 余家人工智能初创企业出现。人工智能大火的同时，相关技术的人才需求也在持续增长着，但是据全球知名职场社交平台领英发布的《全球 AI 领域人才报告》显示，基于领英平台的中国 AI 领域技术人才数量位居全球第七，相关人才总数却只有 5 万多。也就意味着：全国将近 1500 家 AI 智能企业在疯狂争抢着 5 万多的人工智能领域人才。

进入 2018 年，前文所述的各种数字将会有更大的变化，人工智能企业不断增加，职位与机会也会更多，但是人才缺口却一直是个巨大的问题。国家在 2017 年接连发布了各种有关人工智能的发展计划，人才也将是发展战略中相当重要一环。

在发表于《The Aleph》上的一篇文章中指出中国之所以近年来快速实现教育水平提升，主要是源自其拥有北京大学与清华大学这两所全球前三十强一流学府（根据泰晤士高校排名）。该文章还认为这种趋势还将继续持续下去，“虽然像斯坦福大学这类顶尖学术机构仍然在全球范围内保持领先地位，但像北京大学这样的高校正在缩小差距。斯坦福大学在特定领域拥有更高的评价分数，但在技术转化等方面事实上却落后于其它大学。”

尽管中国的人工智能教育一直在快速发展，但人工智能教学人员仍存在严重短缺。中国的众多人工智能从业人员主要来自电气工程或计算机科学等其它分支领域。总而言之，尽管中国人才库的增长势头仍将继续保持下去，但中国显然还需要一段时间才能建立起真正能够与美国正面对抗的完善市场体系。

相信看到这里你已经明白了：现在的中国 AI 发展现状如此，职位多，人才少，虽然不可避免地造成职场高薪满天飞地现象，但是我们要说的是，对于你，现在是一个重要的机会。

或许你还未走出校园，仍然在踌躇是否该步入这瞬息万变的社会；或许你已经有了多年经验，但为了生活还在犹豫。不论怎样，希望你能慎重考虑再做出选择，套用《双城记》的开头：这是最好的时代，也是最坏的时代。希望每个人都能抓住机会，让每一个时代都成为你的时代！

# 百家互联网名企新实践

[北京站 · 2018] | 北京·国际会议中心



会议 2018年04月20-22日

培训 2018年04月23-24日

Google	Google Translate助力自然语言理解	田野 Google研究院 机器学习工程师
Confluent	Apache Kafka的过去, 现在, 和未来	Jun Rao Confluent 联合创始人, Kafka作者之一
Envisage AR	Future Directions for Augmented Reality	Mark Billinghurst CEO of Envisage AR Limited, Christchurch, New Zealand
LinkedIn	Linkedin Derived data platform	严岩 LinkedIn Staff Engineer
Alluxio	使用开源分布式存储系统Alluxio来有效的分离计算与存储	富羽鹏 Alluxio 创始成员&资深架构师
Oracle Labs	GraalVM及其生态系统	郑雨迪 Oracle Labs 高级研究员
广发证券	面向未来的原生化Web开发	郭力恒 广发证券 前端架构师
阿里巴巴	Dubbo开源现状与未来规划	罗毅 阿里巴巴 高级技术专家
中国金融认证中心	人工智能技术在金融行业应用探索	李闯 中国金融认证中心 机器学习实验室高级研究员
华为	SkyWalking的发展之路——从无名小卒到拥抱全球	吴景 华为 软件开发云分布式追踪技术专家
360	人工智能系统中的安全风险	李康 360网络安全北美研究院负责人, IoT安全研究院院长
Airbnb	Airbnb的闪订功能和增长策略	范力 Airbnb 技术经理
阿里巴巴	万物皆向量——双十一淘宝首页个性化推荐背后的秘密	黄丕培 (灵培) 阿里巴巴 高级算法专家

**8折** 优惠倒计时, 3月11日前立减1360元

团购享受更多优惠

如有任何问题, 欢迎咨询。电话: 15110019061, 微信: qcon-0410

访问官网获取更多前沿技术趋势





# 聚焦

- 从云架构到边缘计算
- 人工智能业务架构
- 大数据平台架构实践
- 智能物联网
- 架构和产品研发
- 不可阻挡的AIOps
- 数据库架构
- 广告系统、精准推荐
- 能征善战的工程师文化和团队
- 微服务架构
- 系统架构研究
- 业务基础架构进化
- 大型分布式系统架构
- 视频和游戏架构
- 移动端研发
- 金融核心技术
- 业务出海，架构先行
- 深度学习平台和机器学习应用

# 分享

**Facebook | Machine Learning in Security and Integrity**  
PB数据下，Facebook如何保证实时机器学习平台的安全与完整？

**阿里巴巴 | Flink SQL: 使用标准的ANSI SQL驱动大数据流计算**  
阿里几乎所有的Blink作业都是由Flink SQL编写的，究竟有哪些场景和经验？

**贝聊 | 微服务架构实战历程**

用户规模迅速达到千万，如何平滑从单体应用架构演进到微服务架构？

**更多内容，敬请观看官网..**

Microsoft、eBay、Pinterest、Netflix、阿里、腾讯、百度、小米、有道 ...



知名互联网公司系统架构图

关注ArchSummit公众号，  
快速获取历届架构师的结晶与创意

# 7折优惠

## — 联席主席



方国伟  
平安科技  
CTO兼总架构师



褚霸（余锋）  
阿里巴巴  
研究员

## — 出品人与分享嘉宾



丁宇（叔同）  
阿里巴巴  
2017天猫  
双11技术大队长



金晓军  
阿里巴巴  
高级专家



肖世广  
腾讯  
QQ技术运营总监



陈功  
腾讯  
微信广告引擎负责人



张贺  
南京大学  
软件学院教授



Yunong Xiao  
Netflix  
Principal Software Engineer



杨钦民  
贝聊  
研发总监



孟晓桥  
Pinterest  
监控组经理



Bin Xu  
Facebook  
Software Engineer Manager



大沙  
阿里巴巴  
计算平台事业部  
高级技术专家



段亦韬  
网易有道  
首席科学家



扫码  
快速了解  
大会嘉宾



目前7折报名中，欢迎以目前**最优惠价格**报名ArchSummit

如果在报名过程中遇到任何问题，可联系大会主办方，欢迎咨询！

微信：aschina666 电话：010-84780850

ArchSummit全球架构师峰会深圳站 | 大会演讲：7月6-7日，深度培训：7月8-9日

# Dubbo 正式进入 Apache 孵化器，开启开源新时代

作者 郭蕾



2月15日，大年三十，经过一系列紧张的投票，来自阿里巴巴的广受社区欢迎的RPC开源框架Dubbo宣布正式进入Apache孵化器。说起Dubbo框架，可能很多后端开发者都有所了解，它是国内比较早的、影响较大的开源项目，包括阿里巴巴、京东、当当网、去哪儿网、网易考拉、微店等电商平台都有其成功应用案例。

Dubbo于2011年开源，之后就迅速成为了国内该类开源项目的佼佼者。可以想象，2011年时，优秀的、可在生产环境使用的RPC框架很少，Dubbo的出现迅速给人眼前一亮的感觉，而同时它又有阿里巴巴背书，所以也迅速收到了开发者的亲睐。Dubbo目前在GitHub上有超过16000个

star 和超过 12000 的 fork 数，绝对是国内影响力最大的开源项目之一。

但奇怪的是，在 2014 年 10 月 30 日发布 2.4.11 版本后，Dubbo 突然停止更新，当时社区一片哗然（其实是在 2012 年 10 月之后就基本停止了重要升级，改为阶段性维护）。具体原因现在也不得而知，知乎上也有一些讨论，包括团队调整、内部主推 HSF 等。不过可以确认的是，在 4 年前，国内企业对于开源的重视程度都远远没有今天高。

而在官方停止更新 Dubbo 之后，当当网 (Dubbox)、网易考拉 (Dubbok) 都有维护自己单独的分支，这也从另外一个侧面证明 Dubbo 确实应用到了这些企业的重点业务，并且规模不小。

随着阿里巴巴对于开源的逐步重视，2017 年 9 月 7 日，Dubbo 悄悄的在 GitHub 发布了 2.5.4 版本。随后，没过多久，又迅速发布了 2.5.5、2.5.6、2.5.7 等版本。在 10 月举行的云栖大会上，阿里宣布 Dubbo 被列入集团重点维护开源项目，这也就意味着 Dubbo 起死回生，开始重新进入快车道。

而对于为什么要重新启动维护 Dubbo，以及 Dubbo 和 HSF 的关系，Dubbo 未来的计划，当时聊聊架构也采访了 Dubbo 负责人、阿里巴巴中间件高级技术专家罗毅，感兴趣的读者可以[点击阅读](#)。

这次采访中，令我印象深刻的是罗毅提到了 Dubbo 的愿景，他说开源就阿里巴巴集团在技术层面赋能的重要领域，阿里巴巴中间件团队今后不仅要聆听社区的声音，及时修复问题，及时合并优秀的 pull request，还会力争将 Dubbo 打造成有国际影响力的 RPC 框架。国际影响力，让人一下子沸腾。

而对于 Dubbo 和 Spring Cloud 的区别，罗毅也做了总结，一针见血：

需要强调的是 Dubbo 未来的定位并不是要成为一个微服务的全面解决方案 (Spring Cloud 是)，而是专注在 RPC 领域，成为微服务生态体系中的一个重要组件。至于大家关注的微服务化衍生出的服务治理需求，我们会在 Dubbo 积极适配开源解决方案，甚至启动独立的开源项目予以支持。

这一次，Dubbo 进入 Apache 孵化器。也就是说，Dubbo 将不再是

阿里巴巴的 Dubbo，而是社区的，它未来的走向以及规则将会像其他的 Apache 项目一样。

Today is the Chinese New Year's Eve, thanks everyone for their effort to bring Dubbo into Apache Incubator, and happy new year to everyone in the community!

--  
Best Regards!  
Huxing

不过，从孵化项目到正式的开源项目，Dubbo 其实还有一段路要走。知乎上，昵称为二货的用户对这一流程做了详细解释，以下为摘录：

Apache 项目有多个阶段，第一个阶段是进入孵化器。在进入孵化器前会有诸多审核流程，通过后进入 Apache Incubator。此时成员需要签一个协议，完成后获赠 Apache 账户（Apache 邮箱可以免费使用 intellij 哦，这也是 jetbrains 对开源贡献者的鼓励呐~）。

在这个阶段会有 mentor 进行社区化指导，包括 PR 流程，包括 license 检查，包括 mail list 的回复，等等等。除了项目保持活跃外，还需要有外部 commiter。当项目在孵化器中持续一段时间满足毕业条件后便可以走正式毕业流程了。

毕业后，项目移出 incubator，成为正式开源项目。项目更新流程不会有变化。另一种情况是项目失活，缺少社区支持与维护。那么就会被移出（不多见）。这里需要注意的是，社区活跃度是一个培养的过程。并不是说你一来就社区全是人的，这也正是孵化阶段的目的。

最后，祝 Dubbo 能有一个更好的未来，就像其使命一样，成为有国际视野的顶级开源项目。同时，也祝各位开发者新年快乐，狗年旺旺！

# 批处理 ETL 已经消亡，Apache Kafka 才是数据处理的未来吗？

作者 Daniel Bryant 译者 张卫滨



在 [QCon 旧金山 2016](#) 会议上，Neha Narkhede 做了“ETL 已死，而实时流长存”的演讲，并讨论了企业级数据处理领域所面临的挑战。该演讲的核心前提是开源的 Apache Kafka 流处理平台能够提供灵活且统一的框架，支持数据转换和处理的现代需求。

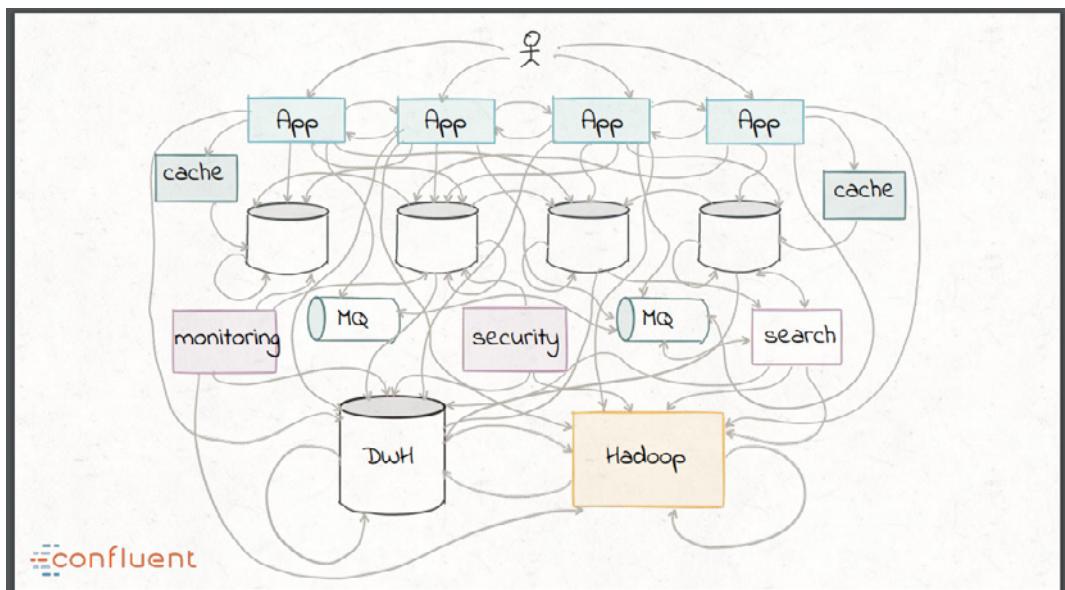
[Narkhede](#) 是 Confluent 的联合创始人和 CTO，在演讲中，他首先阐述了在过去的十年间，数据和数据系统的重要变化。该领域的传统功能包括提供联机事务处理（online transaction processing, OLTP）的操作性数据库以及提供在线分析处理（online analytical processing, OLAP）的关系型数据仓库。来自各种操作性数据库的数据会以批处理的方式加载到数据仓库的主模式中，批处理运行的周期可能是每天一次或两次。这种数据集成过

程通常称为抽取-转换-加载（extract-transform-load，ETL）。

最近的一些数据发展趋势推动传统的 ETL 架构发生了巨大的变化：

- 单服务器的数据库正在被各种分布式数据平台所取代，这种平台在整个公司范围内运行；
- 除了事务性数据之外，现在有了类型更多的数据源，比如日志、传感器、指标数据等；
- 流数据得到了普遍性增长，在速度方面比每日的批处理有了更快的业务需求。

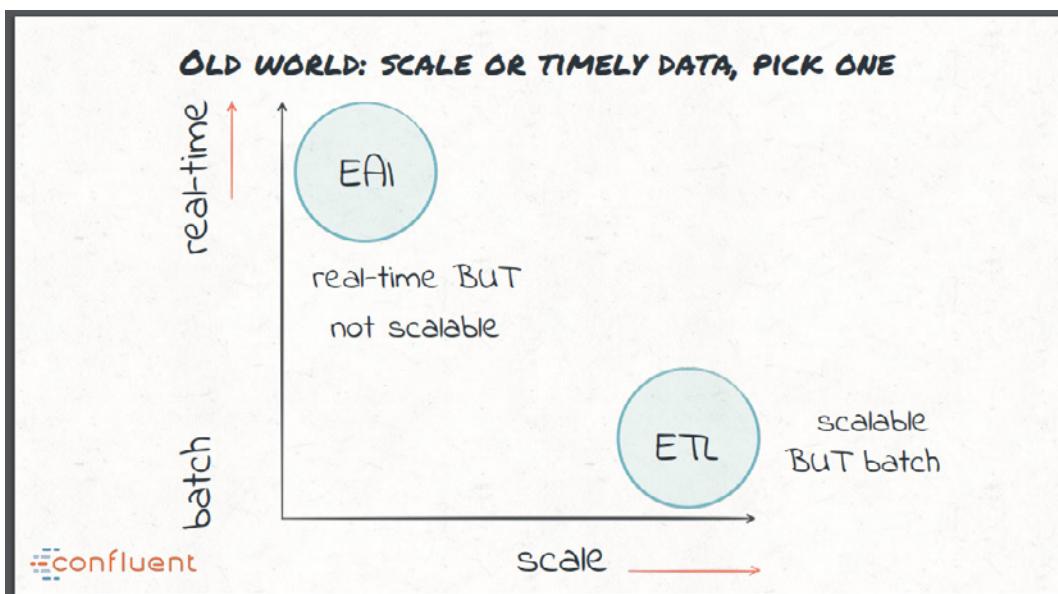
这些趋势所造成的后果就是传统的数据集成方式最终看起来像一团乱麻，比如组合自定义的转换脚本、使用企业级中间件如企业服务总线（ESB）和消息队列（MQ）以及像 Hadoop 这样的批处理技术。



在探讨现代流处理技术如何缓解这些问题之前，Narkhede 简要回顾了一下数据集成的历史。在上世纪 90 年代的零售行业中，业务得到了一些新形式的数据，所以对购买者行为趋势进行分析的需求迫切增长。存储在 OLTP 数据库中的操作性数据必须要进行抽取、转换为目标仓库模式，然后加载到中心数据仓库中。这项技术在过去二十年间不断成熟，但是数据仓库中的数据覆盖度依然相对非常低，这主要归因于 ETL 的如下缺点：

- 需要一个全局的模式；
- 数据的清洗和管理需要手工操作并且易于出错；
- ETL的操作成本很高：它通常很慢，并且是时间和资源密集型的；
- ETL所构建的范围非常有限，只关注于以批处理的方式连接数据库和数据仓库。

在实时 ETL 方面，早期采用的方式是[企业应用集成](#) (Enterprise application integration, EAI)，并使用 ESB 和 MQ 实现数据集成。尽管这可以说是有效的实时处理，但这些技术通常很难广泛扩展。这给传统的数据集成带来了两难的选择：实时但不可扩展，或者可扩展但采用的是批处理方案。



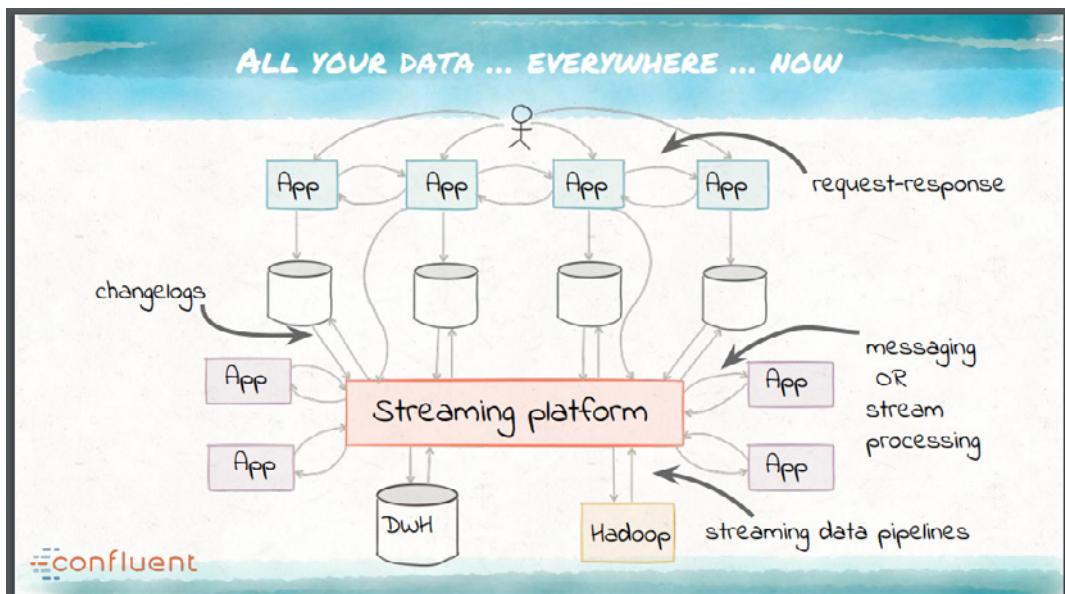
Narkhede 指出现代流处理对数据集成有了新的需求：

- 能够处理大量且多样性的数据；
- 平台必须要从底层就支持实时处理，这会促进向以事件为中心的根本转变；
- 必须使用向前兼容的数据架构，必须能够支持添加新的应用，这些新的应用可能会以不同的方式来处理相同的数据。

这些需求推动一个统一数据集成平台的出现，而不是一系列专门定制的工具。这个平台必须拥抱现代架构和基础设施的基本理念、能够容错、能够并行、支持多种投递语义、提供有效的运维和监控并且允许进行模式管理。Apache Kafka 是七年前由 LinkedIn 开发的，它就是这样一个开源流平台，能够作为组织中数据的中枢神经系统来运行，方式如下：

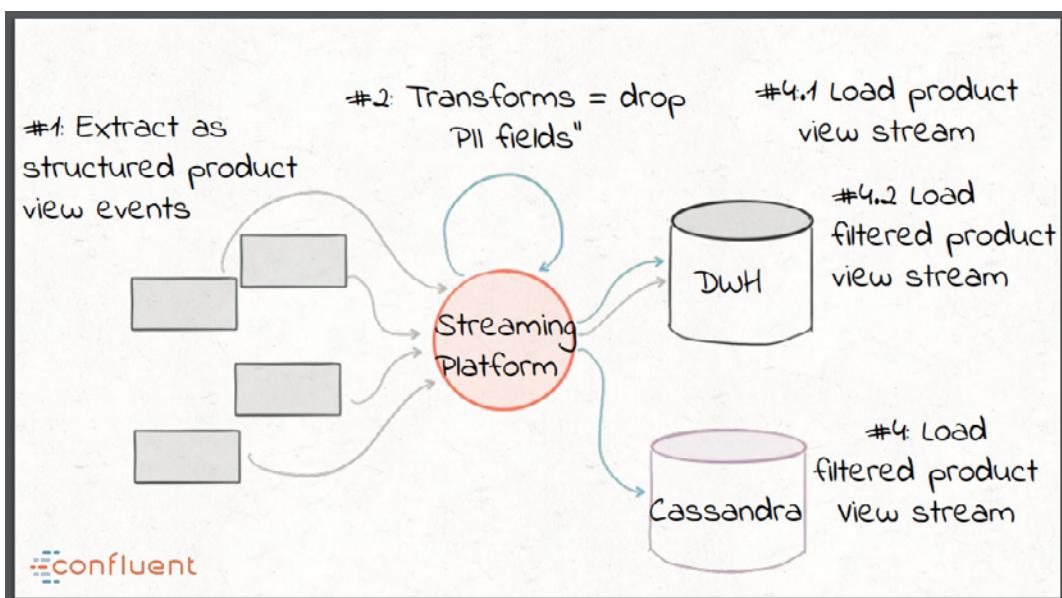
- 作为应用的实时、可扩展消息总线，不需要EAI；
- 为所有的消息处理目的地提供现实状况来源的管道；
- 作为有状态流处理微服务的基础构建块。

Apache Kafka 在 LinkedIn 目前每天处理 14 万亿条的消息，并且已经部署到了世界范围内成千上万的组织之中，包括财富 500 强的公司，如 Cisco、Netflix、PayPal 和 Verizon。Kafka 已经快速成为流数据的存储方案，并且为应用集成提供了一个可扩展的消息支撑（backbone），能够跨多个数据中心。



Kafka 的基础是 log 的理念，log 是只能往上追加（append），完全有序的数据结构。log 本身采用了发布 - 订阅（publish-subscribe, pubsub）的语义，发布者能够非常容易地以不可变的形式往 log 上追加数据，订阅者可以维护自己的指针，以便指示当前正在处理的消息。

Kafka 能够通过 [Kafka Connect API](#) 实现流数据管道的构建，也就是 ETL 中的 E 和 L。Connect API 利用了 Kafka 的可扩展性，基于 Kafka 的容错模型进行构建并且提供了一种统一的方式监控所有的连接器。流处理和转换可以通过 [Kafka Streams API](#) 来实现，这提供了 ETL 中的 T。使用 Kafka 作为流处理平台能够消除为每个目标 sink、数据存储或系统创建定制化（很可能重复的）抽取、转换和加载组件的需求。来自 source 的数据经过抽取后可以作为结构化的事件放到平台中，然后可以通过流处理进行任意的转换。



在演讲的最后一部分，Narkhede 详细讨论了流处理的概念，也就是流数据的转换，并且提出了两个相互对立的愿景：实时的 MapReduce 和事件驱动的微服务。实时的 MapReduce 适用于分析用例并且需要中心化的集群和自定义的打包、部署和监控。[Apache Storm](#)、[Spark Streaming](#) 和 [Apache Flink](#) 实现了这种模式。Narkhede 认为事件驱动微服务的方式（通过 Kafka Streams API 来实现）让任何用例都能访问流处理，这只需添加一个嵌入式库到 Java 应用中并搭建一个 Kafka 集群即可。

Kafka Streams API 提供了一个便利的 fluent DSL，具有像 join、map、filter 和 window 这样的操作符。

### WORD COUNT PROGRAM USING KAFKA'S STREAMS API

```
KStreamBuilder builder = new KStreamBuilder();
KStream<String, String> textLines = builder.stream(stringDeserializer, stringDeserializer, "TextLinesTopic");

KStream<String, Long> wordCounts = textLines
    .flatMapValues(value -> Arrays.asList(value.toLowerCase().split("\\W+")))
    .map((key, value) -> new KeyValue<>(value, value))
    .countByKey(stringSerializer, longSerializer, stringDeserializer, longDeserializer, "Counts")
    .toStream();
wordCounts.to("WordsWithCountsTopic", stringSerializer, longSerializer);

KafkaStreams streams = new KafkaStreams(builder, config);
streams.start();
```



这是真正的每次一个事件 (event-at-a-time) 的流处理，没有任何微小的批处理，它使用数据流 (dataflow) 风格的窗口 (window) 方式，基于事件的时间来处理后续到达的数据。Kafka Streams 内置了对本地状态的支持，并且支持快速的状态化和容错处理。它还支持流的重新处理，在更新应用、迁移数据或执行 A/B 测试的时候，这是非常有用的。

Narkhede 总结说，log 统一了批处理和流处理，log 可以通过批处理的窗口方式进行消费，也能在每个元素抵达的时候进行检查以实现实时处理，Apache Kafka 能够提供“ETL 的崭新未来”。

Narkhede 在旧金山 QCon 的完整视频可以在 InfoQ 上“[ETL Is Dead; Long Live Streams](#)”查看。

## 作者简介

Daniel Bryant 一直在组织内和技术方面引领变化。他目前的工作包括通过引入更好的需求收集和计划技术推进企业内部的敏捷性，关注于敏捷开发中的架构关联性，并搭建持续集成 / 交付环境。Daniel 现在的技术专长是“DevOps”工具、云 / 容器平台和微服务实现。

# Apache Kylin 在绿城客户画像系统中的实践

作者 秦海龙



## 前言

作为国内知名的房地产开发商，绿城经过 24 年的发展，已为全国 25 万户、80 万人营造了美丽家园，并将以“理想生活综合服务提供商”为目标，持续为客户营造高品质的房产品和生活服务。

2017 年，绿城理想生活集团成立，围绕客户全生活链、房屋全生命周期，为客户提供从买房子到房屋的保养维护，再到业主全方位的生活服务。为此构建了绿城 +App 生活服务平台、房产营销数字化平台及房屋 4S 服务平台，这些系统的构建为业主购房及生活服务提供了极大的便利，部分系统不仅开放给绿城客户、业主使用，同时也服务于非绿城的客户。通过一

整套垂直行业的用户画像系统构建并使用 Apache Kylin 加速主要数据服务，有效提升了互联网广告推广、营销服务的效率。

## 一、绿城客户画像系统的背景

房产品的营造和线下销售是当前绿城的主营业务，为有效提升服务质量、管理效能，降低营销费用，实现客户服务智能化、销售行为自动化、成本管理合理化，绿城积极拥抱互联网，于 2015 年开始了数字化营销（Digital Marketing）的探索和研究，通过 + 互联网创新营销业务。

经过 2 年的探索和模式验证之后，2017 年绿城成立了专门的大数据团队，围绕营销全过程和客户全生命周期，构建了房地产行业首个全闭环的“房产营销数字化平台”，服务于营销找客到成交回款全过程，如下图所示：

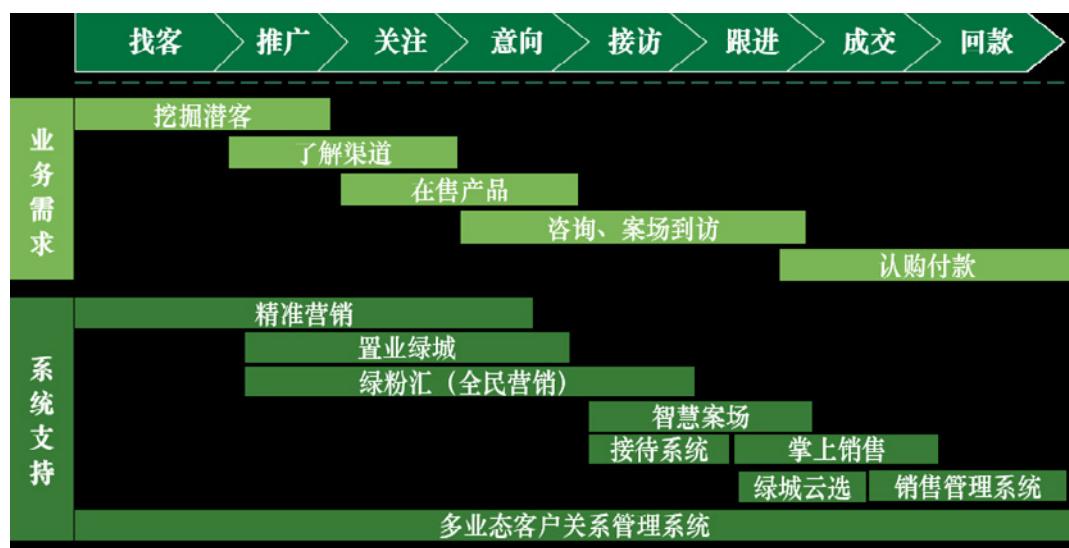


图 1 绿城房产营销数字化平台

在“房产营销数字化平台”中，精准营销和智慧案场为营销线最核心的两个系统，它们以广告投放、客户数据资产管理、经营指标分析为基础，延展出集合营销知识分享与学习、营销与转化工具、第三方供应商为一体的互联网平台，服务于房地产市场营销产业链生态圈，为 Marketing 阶段的客户获取提供了一站式程序化解决方案。另外置业绿城、掌上销售等系统则为后续的 Sales 环节提供数字化服务。

精准营销系统和智慧案场系统，基于 DMP (Data Management Platform，数据管理平台) 的数据分析和处理能力支撑和流转起所有业务逻辑，一方面，绿城 DMP 系统通过积累营销投放过程中的回流数据，另外一方面又采集置业绿城、全民营销系统（绿粉汇）、掌上销售系统中的埋点行为数据及数据库数据。通过上述种种方式为数字化营销建立更为准确优化的策略，从而真正做到“数据驱动营销”。绿城 DMP 的数据包含第一、第二和第三方数据：

第一方数据，即完全自有的数据。企业自身的 CRM 系统数据、网站和 APP 等运营活动的应用数据；

第二方数据，主要包括程序化广告投放过程中的交易数据；

第三方数据，主要为 BAT 数据、运营商数据等。

绿城 DMP 整体的业务架构图如下：

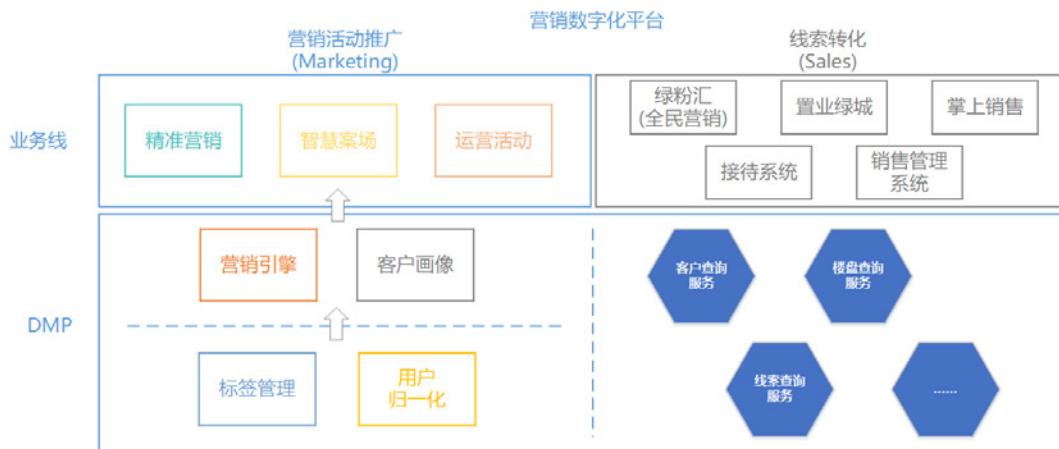


图 2 绿城 DMP 与系统间的逻辑架构

DMP 作为服务于 Marketing 的核心工具，客户画像发挥着极其重要的作用。客户画像依赖于 DMP 的标签管理、用户归一化以及营销相关的客户数据，它为房子的营销推广提供决策支持和依据。

另外一方面，营销相关运营活动也需要画像系统支持。营销引擎基于用户画像系统，为精准营销、智慧案场系统提供统一的广告投放服务。

## 二、客户画像与 Apache Kylin 的结合

如前所述，客户画像服务于 Marketing，其核心的业务流程可以用下图表示：

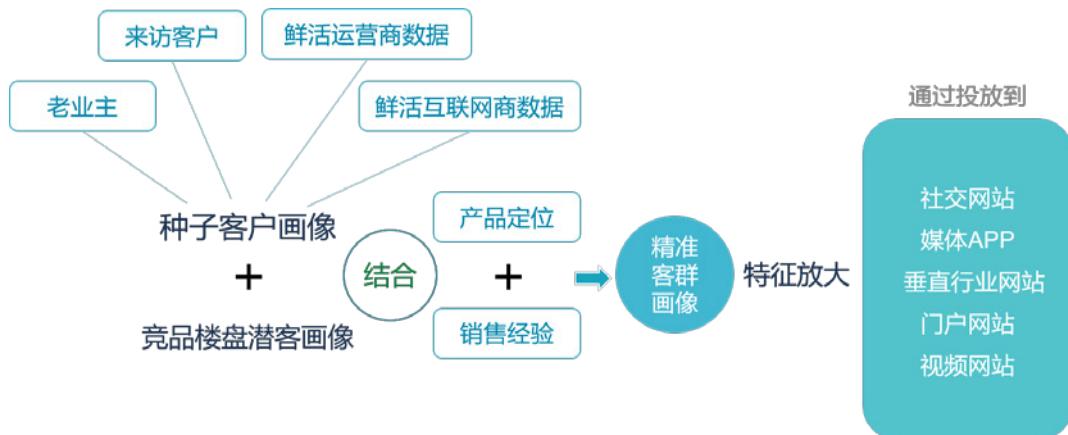


图 3 客户画像的核心逻辑

通过 DMP 进行数据的采集、融合分析、归一化处理，再基于行业标签，为精准营销系统提供精准的人群画像，并投放到各类媒体及网站，实现对于受众的精准触达。

2015~2016 年，绿城大数据平台中的数据主要通过 Hive + HBase 进行存储以及分析计算，后台的数据服务尤其是画像服务，均是基于 HBase 的 Java API 开发，那时基本能满足业务秒级的响应需求。但经历 2017 年的业务高速发展之后，随着渠道及合作方的增多，数据的体量和维度的增加了数十倍，画像等数据服务的响应速度逐渐降至 5 秒甚至 30 秒，部分业务查询甚至超过了 1min，而且数据源头繁杂、维度众多，需要体系化地管理。为解决这个问题，绿城大数据团队于 17 年上半年进行标签体系建设形成共 13 大类、8000+ 细类的多维度标签，客户画像的构建，便依赖于这个丰富成熟的标签体系。

日均 300G 以上数据会沉淀在大数据平台中，数据体量的增加导致性能瓶颈明显，经过多轮测试、综合对比分析 Apache Kudu，Presto，Druid

以及 Apache Kylin 之后，最终选择 Apache Kylin 作为 OLAP 工具，最终优化并解决了数据服务查询的性能问题。选择 Apache Kylin 的主要原因有以下几点：

- 成熟度来讲：Apache Kylin 和 Druid 更为成熟（参照稳定性、性能、社区活跃度等因素）
- 查询效率来讲：Druid  $\approx$  Apache Kylin，优于其他（主要业务场景）
- 实用和便捷性：Apache Kylin 搭建和使用均较为便捷（同时也是华人的顶级开源项目）

另外，Apache Kylin 还有以下优点：

- Apache Kylin 进行预算算，空间换时间，通过预定义、计算 Cube 的方式提升查询的速度和性能，同时，查询的性能随业务的增长也不会受到影响；
- 数据管理及同步方便。预算算、构建 Cube、数据管理都可基于 Apache Kylin 自行管理；有开放的 API 可以方便、快速地对接内部数据处理流程、与调度系统打通。

绿城大数据平台每日增量构建数百 GB 的 Cube，构建的时间从几小时到十几小时不等，之前后台较慢的查询时间范围是从十几秒到几十秒，使用 Apache Kylin 后则基本都在 1-2 秒内即可予以响应。最终优化之后的客户画像构建流程如下：

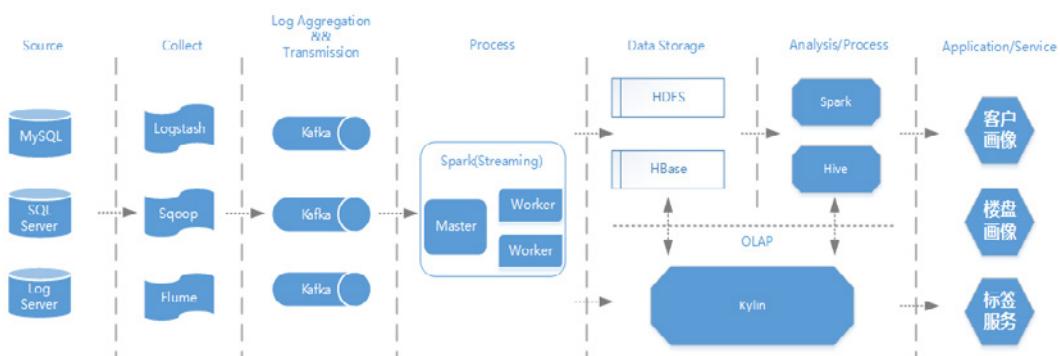


图 4 客户画像构建流程

其中，业务系统数据和 Log 数据通过采集、传输后，基于 Spark 进行初步处理，之后包含埋点、运营活动等的结果数据会写入 HDFS 以及 HBase 中。一部分客户、楼盘的数据报告和分析服务通过 Hive 及 Spark 进行支撑和输出，而主要的数据服务则通过 Apache Kylin 进行构建。

在 Kylin 中，对于小数据量的 Cube，或者经常需要全表更新的 Cube，使用全量构建需要更少的运维精力，以少量的重复计算降低生产环境中的维护复杂度。而对于大数据量的 Cube，例如对于一个包含两年历史数据的 Cube，如果需要每天更新，那么每天为了新数据而去重复计算过去两年的数据就会变得非常浪费，而在这种情况下需要考虑使用增量构建。

因为绿城大数据平台的数据每天按日更新，并且日均数据量都会在百 G 以上，所以我们用到了 Apache Kylin 的增量构建 Cube。Kylin 在 Web 界面上提供了手动构建 Cube 的操作，此外，Apache Kylin 也提供了 Rest API 进行增量构建。在绿城客户画像系统中，70% 的自动化触发增量构建都基于 Rest API 完成。

Name	Status	Cube Size	Source Records	Last Build Time	Owner	Create Time	Actions	Admins	Streaming
Cube_00000_00000	READY	611.06 GB	6,328,965,668	2017-01-01 00:00:00 GMT+8	admin	2017-01-01 00:00:00 GMT+8	Action	Action	Action
Cube_00000_00001	READY	120.83 GB	346,361	2017-01-01 00:00:00 GMT+8	admin	2017-01-01 00:00:00 GMT+8	Action	Action	Action
Cube_00000_00002	READY	120.83 GB	346,361	2017-01-01 00:00:00 GMT+8	admin	2017-01-01 00:00:00 GMT+8	Action	Action	Action

图 5 Apache Kylin 构建 Cube 的 Web 页面

我们基于 Apache Kylin 构建好的数据服务，又通过开源工具 Superset 进行客户画像中标签数据的可视化分析展示，如图 6 所示。

大数据可视化工具的选择非常丰富。在对比了开源工具 Superset、Zeppelin 以及商业工具 FineBI 后，最终采用 Airbnb 开源的 Superset（曾用名 Caravel）的主要原因如下：

1. 数据安全性、权限控制，仅 Superset 有表检索的权限控制
2. 图表多样性，Superset 拥有多达 30 张以上的图表，多表的联动

## 性-filter支持多表联动

3. 数据库多元性，Superset既支持关系型数据库，也支持像Apache Kylin这样的大数据框架
4. 社区活跃度相对更高
5. Superset作为一款开源的BI工具，能够满足我们对于标签画像联动分析的需求，节省了前端、UI的开发资源

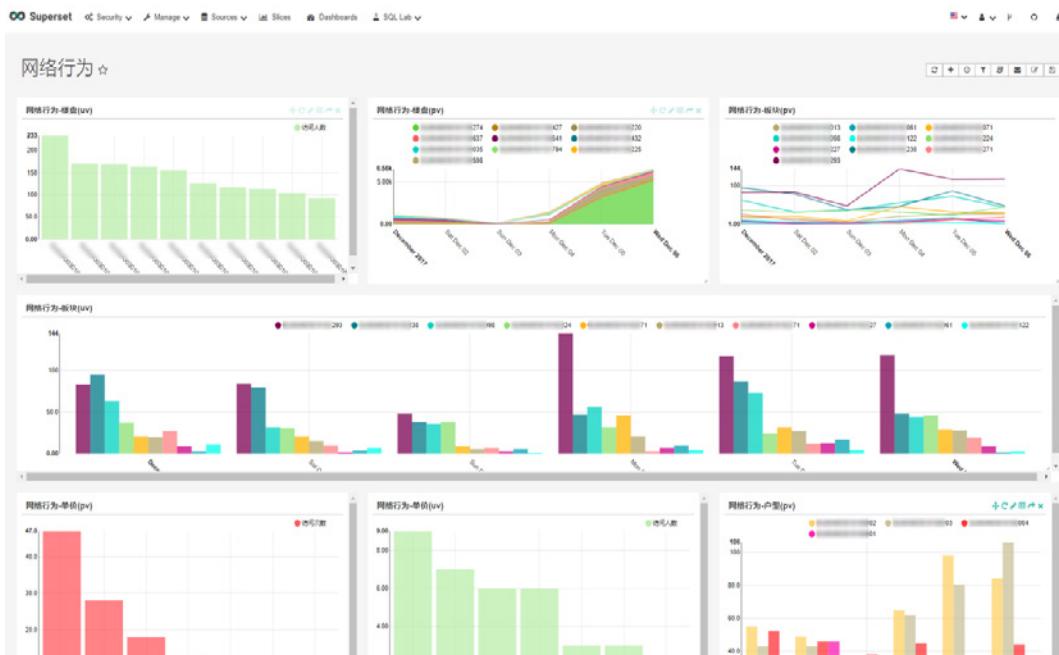


图 6 基于 Superset 的标签画像展示

客户画像依赖的数据、后台计算引擎以及标签都构建完成后，绿城客户画像的一瞥如图 7 所示。

## 三、未来客户画像系统的展望

绿城客户画像系统目前只服务于房产营销，随着房屋 4S、园区商业、绿城 +App 生活服务平台的日益成熟，画像系统将融合各业务系统数据，完成客户全生活链用户画像的建设，同时客户画像会融入知识图谱，建立业主与业主、业主与房子之间的连接，从而形成一套更加全面、可视化的用户画像系统。绿城大数据团队将积极拥抱开源、拥抱互联网，拥抱变化，

用户画像管理系统 欢迎您, [用户名] | 退出

按楼盘筛选  行政区域  城市公司  项目名称  
选:

按单套筛选:

业态:	<input type="checkbox"/> 高层公寓	<input type="checkbox"/> 多层公寓	<input type="checkbox"/> 排屋	<input type="checkbox"/> 储藏室	<input type="checkbox"/> 小高层公寓	<input type="checkbox"/> 写字楼	<input type="checkbox"/> 商铺	更多 <input type="button" value="▼"/>
单套总价:	<input type="checkbox"/> 100万以下	<input type="checkbox"/> 100万-200万	<input type="checkbox"/> 200万-300万	<input type="checkbox"/> 300万-500万	<input type="checkbox"/> 500万-1000万	<input type="checkbox"/> 1000万以上		
单套单价:	<input type="checkbox"/> 5千以下	<input type="checkbox"/> 5千-1万	<input type="checkbox"/> 1-1.5万	<input type="checkbox"/> 1.5-2万	<input type="checkbox"/> 2-3万	<input type="checkbox"/> 3-4万	<input type="checkbox"/> 4-5万	更多 <input type="button" value="▼"/>
单套面积:	<input type="checkbox"/> 90m <sup>2</sup> 以下	<input type="checkbox"/> 90-110m <sup>2</sup>	<input type="checkbox"/> 110-130m <sup>2</sup>	<input type="checkbox"/> 130-150m <sup>2</sup>	<input type="checkbox"/> 150-170m <sup>2</sup>	<input type="checkbox"/> 170-190m <sup>2</sup>	<input type="checkbox"/> 190-220m <sup>2</sup>	更多 <input type="button" value="▼"/>
付款方式:	<input type="checkbox"/> 一次性付款	<input type="checkbox"/> 商业贷款	<input type="checkbox"/> 公积金贷款	<input type="checkbox"/> 分期付款	<input type="checkbox"/> 组合贷款			
购房时间:	<input type="checkbox"/> 1年以内	<input type="checkbox"/> 1-3年	<input type="checkbox"/> 3-5年	<input type="checkbox"/> 5-8年	<input type="checkbox"/> 8年以上			

[展开高级筛选](#)

[开始分析](#)

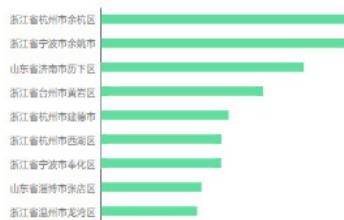
## 群体分析

共找到 人

## 楼盘分布



## 工作区域分布



## 基本信息



## 性别分布



行业分布(未知: 人 120000)

年龄分布(未知: 人 280000)

持续用技术和数据驱动绿城各条线的业务发展。

## 作者简介

秦海龙，绿城理想生活科技有限公司大数据平台负责人。Java 语言、Scala 语言，Hadoop 生态、Spark 大数据处理技术爱好者。

# 微服务架构技术栈选型手册

作者 杨波



## 一、前言

2014 年可以认为是微服务 1.0 的元年，当年有几个标志性事件，一是 Martin Fowler 在其博客上发表了”Microservices”一文，正式提出微服务架构风格；二是 Netflix 微服务架构经过多年大规模生产验证，最终抽象落地形成一整套开源的微服务基础组件，统称 NetflixOSS，Netflix 的成功经验开始被业界认可并推崇；三是 Pivotal 将 NetflixOSS 开源微服务组件集成到其 Spring 体系，推出 Spring Cloud 微服务开发技术栈。

一晃三年过去，微服务技术生态又发生了巨大变化，容器，PaaS，Cloud Native，gRPC，ServiceMesh，Serverless 等新技术新理念你方唱罢我

登场，不知不觉我们又来到了微服务 2.0 时代。

基于近年在微服务基础架构方面的实战经验和平时的学习积累，我想总结并提出一些构建微服务 2.0 技术栈的选型思路，供各位在一一线实战的架构师、工程师参考借鉴。对于一些暂时还没有成熟开源产品的微服务支撑模块，我也会给出一些定制自研的设计思路。

## 二、选型准则

对于技术选型，我个人有很多标准，其中下面三项是最重要的：

### 1. 生产级

我们选择的技术栈是要解决实际业务问题和上生产抗流量的（选择不慎可能造成生产级事故），而不是简单做个 POC 或者 Demo 展示，所以生产级（Production Ready），可运维（Ops Ready），可治理，成熟稳定的技术才是我们的首选；

### 2. 一线互联网公司落地产品

我们会尽量采用在一一线互联网公司落地并且开源的，且在社区内形成良好口碑的产品，它们已经在这些公司经过流量冲击，坑已经基本被填平，且被社区接受形成一个良好的社区生态（本文附录部分会给出所有推荐使用或参考的开源项目的 GitHub 链接）。

### 3. 开源社区活跃度

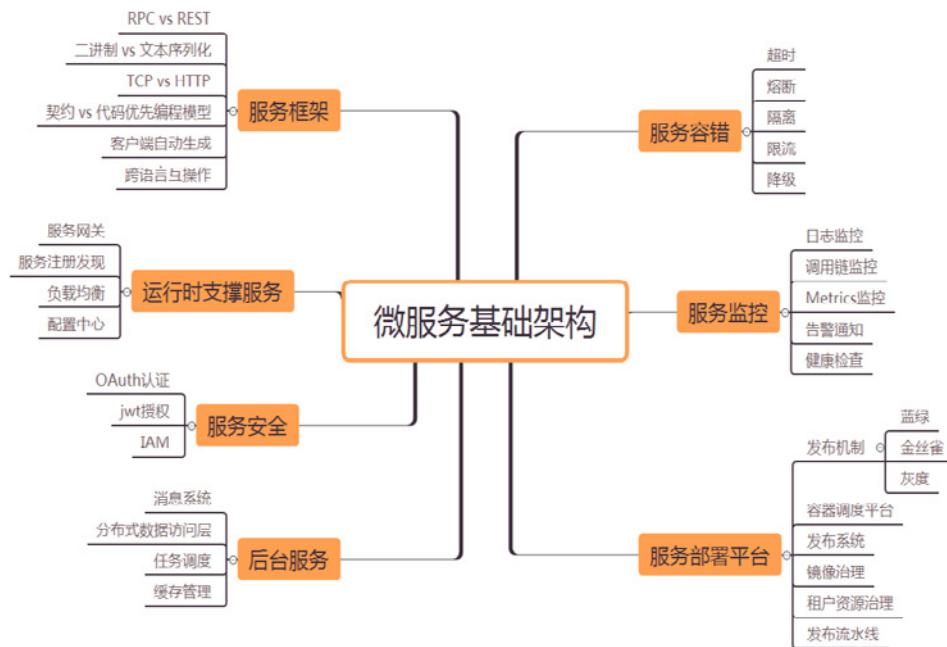
GitHub 上的 stars 的数量是一个重要指标，同时会参考其代码和文档更新频率（尤其是近年），这些指标直接反应开源产品的社区活跃度或者说生命力。

另外，对于不同业务体量和团队规模的公司，技术选型标准往往是不同的，创业公司的技术选型和 BAT 级别公司的技术选型标准可能完全不同。本文主要针对日流量千万以上，研发团队规模不少于 50 人的公司，如果小于这个规模我建议认真评估是否真的需要采用微服务架构。考虑到 Java 语言在国内的流行度和个人的背景经验，本文主要针对采用 Java

技术栈的企业。本文也假定自建微服务基础架构，有些产品其实有对应的云服务可以直接使用，自建和采用云服务各有利弊，架构师需要根据场景上下文综合权衡。

### 三、微服务基础架构关键点

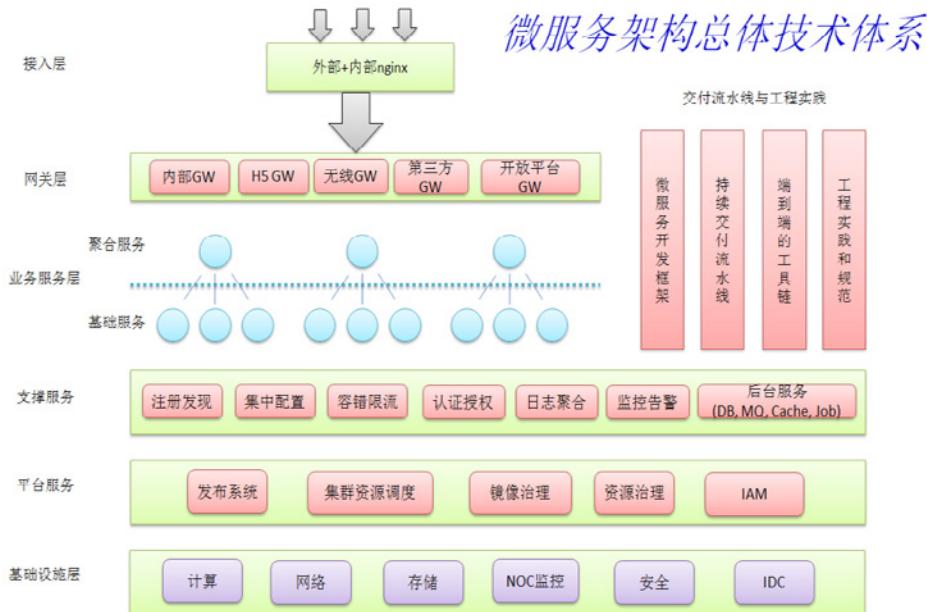
下面脑图中芒果色标注的七个模块，我认为是构建微服务 2.0 技术栈的核心模块，本文后面的选型会分别基于这些模块展开。对于每个模块我也列出一些核心架构关注点，在选择具体产品时，需要尽可能覆盖到这些关注点。



下图是我近期工作总结和参考的一个微服务技术体系，我想同时分享给一线架构师或者工程师参考，其中粉红色标注的模块是和微服务关系最密切的模块，大家在做技术选型时，可以同时对照这个体系。

### 四、服务框架选型

服务框架是一个比较成熟的领域，有太多可选项。Spring Boot/Cloud[附录 12.1] 由于 Spring 社区的影响力和 Netflix 的背书，目前可以认



为是构建 Java 微服务的一个社区标准，Spring Boot 目前在 GitHub 上有超过 20k 星。

基于 Spring 的框架本质上可以认为是一种 RESTful 框架（不是 RPC 框架），序列化协议主要采用基于文本的 JSON，通讯协议一般基于 HTTP。RESTful 框架天然支持跨语言，任何语言只要有 HTTP 客户端都可以接入调用，但是客户端一般需要自己解析 payload。目前 Spring 框架也支持 Swagger 契约编程模型，能够基于契约生成各种语言的强类型客户端，极大方便不同语言栈的应用接入，但是因为 RESTful 框架和 Swagger 规范的弱契约特性，生成的各种语言客户端的互操作性还是有不少坑的。

Dubbo[ 附录 12.2] 是阿里多年构建生产级分布式微服务的技术结晶，服务能力非常丰富，在国内技术社区具有很大影响力，目前 github 上有超过 16k 星。Dubbo 本质上是一套基于 Java 的 RPC 框架，当当 Dubbox 扩展了 Dubbo 支持 RESTful 接口暴露能力。

Dubbo 主要面向 Java 技术栈，跨语言支持不足是它的一个弱项，另外因为治理能力太丰富，以至于这个框架比较重，完全用好这个框架的门槛比较高，但是如果你的企业基本上投资在 Java 技术栈上，选 Dubbo

可以让你在服务框架一块站在较高的起点上，不管是性能还是企业级的服务治理能力，Dubbo 都做的很出色。新浪微博开源的 Motan (GitHub 4k stars) 也不错，功能和 Dubbo 类似，可以认为是一个轻量裁剪版的 Dubbo。

gRPC[ 附录 12.3] 是谷歌近年新推的一套 RPC 框架，基于 protobuf 的强契约编程模型，能自动生成各种语言客户端，且保证互操作。支持 HTTP2 是 gRPC 的一大亮点，通讯层性能比 HTTP 有很大改进。Protobuf 是在社区具有悠久历史和良好口碑的高性能序列化协议，加上 Google 公司的背书和社区影响力，目前 gRPC 也比较火，GitHub 上有超过 13.4k 星。

目前看 gRPC 更适合内部服务相互调用场景，对外暴露 RESTful 接口可以实现，但是比较麻烦（需要 gRPC Gateway 配合），所以对于对外暴露 API 场景可能还需要引入第二套 RESTful 框架作为补充。总体上 gRPC 这个东西还比较新，社区对于 HTTP2 带来的好处还未形成一致认同，建议谨慎投入，可以做一些试点。

## 五、运行时支撑服务选型

运行时支撑服务主要包括服务注册中心，服务路由网关和集中式配置中心三个产品。

服务注册中心，如果采用 Spring Cloud 体系，则选择 Eureka[ 附录 12.4] 是最佳搭配，Eureka 在 Netflix 经过大规模生产验证，支持跨数据中心，客户端配合 Ribbon 可以实现灵活的客户端软负载，Eureka 目前在 GitHub 上有超过 4.7k 星；Consul[ 附录 12.5] 也是不错选择，天然支持跨数据中心，还支持 KV 模型存储和灵活健康检查能力，目前在 GitHub 上有超过 11k 星。

服务网关也是一个比较成熟的领域，有很多可选项。如果采用 Spring Cloud 体系，则选择 Zuul[ 附录 12.6] 是最佳搭配，Zuul 在 Netflix 经过大规模生产验证，支持灵活的动态过滤器脚本机制，异步性能不足（基于 Netty 的异步 Zuul 迟迟未能推出正式版）。Zuul 网关目前在 github 上有

超过 3.7k 星。基于 Nginx/OpenResty 的 API 网关 Kong[附录 12.7] 目前在 github 上比较火，有超过 14.1k 星。因为采用 Nginx 内核，Kong 的异步性能较强，另外基于 lua 的插件机制比较灵活，社区插件也比较丰富，从安全到限流熔断都有，还有不少开源的管理界面，能够集中管理 Kong 集群。

配置中心，Spring Cloud 自带 Spring Cloud Config[附录 12.8]（GitHub 0.75k stars），个人认为算不上生产级，很多治理能力缺失，小规模场景可以试用。个人比较推荐携程的 Apollo[附录 12.9] 配置中心，在携程经过生产级验证，具备高可用，配置实时生效（推拉结合），配置审计和版本化，多环境多集群支持等生产级特性，建议中大规模需要对配置集中进行治理的企业采用。Apollo 目前在 github 上有超过 3.4k 星。

## 六、服务监控选型

主要包括日志监控，调用链监控，Metrics 监控，健康检查和告警通知等产品。

ELK 目前可以认为是日志监控的标配，功能完善开箱即用，ElasticSearch[附录 12.10] 目前在 GitHub 上有超过 28.4k 星。Elastalert[附录 12.11]（GitHub 4k stars）是 Yelp 开源的针对 ELK 的告警通知模块。

调用链监控目前社区主流是点评 CAT[附录 12.12]（GitHub 4.3k stars），Twitter 之前开源现在由 OpenZipkin 社区维护的 Zipkin[附录 12.13]（GitHub 7.5k stars）和 Naver 开源的 Pinpoint[附录 12.14]（GitHub 5.3k stars）。个人比较推荐点评开源的 CAT，在点评和国内多家互联网公司有落地案例，生产级特性和治理能力较完善，另外 CAT 自带告警模块。下面是我之前对三款产品的评估表，供参考。

Metrics 监控主要依赖于时间序列数据库 (TSDB)，目前较成熟的产品是 StumbleUpon 公司开源的基于 HBase 的 OpenTSDB[附录 12.15]（基于 Cassandra 的 KariosDB[附录 12.16] 也是一个选择，GitHub 1.1k stars，它基本上是 OpenTSDB 针对 Cassandra 的一个改造版），OpenTSDB 具有分布式能力可以横向扩展，但是相对较重，适用于中大规模企业，

	CAT	Zipkin	Pinpoint
调用链可视化	有	有	有
报表	非常丰富	少	中
ServerMap	简单依赖图	简单	好
埋点方式	侵入	侵入	不侵入字节码增强
Heartbeat支持	有	无	有
Metric支持	有	无	无
Java/Net客户端支持	有	有	只有Java
Dashboard中文支持	好	无	无
社区支持	好, 文档较丰富, 作者在携程点评	好, 文档一般, 暂无中文社区	一般, 文档缺, 无中文社区
国内案例	携程、点评、陆金所	京东、阿里不开源	暂无
源头祖先	eBay CAL - Centralized Application Logging	Google Dapper	Google Dapper

OpenTSDB 目前在 GitHub 上有近 2.9k 星。

OpenTSDB 本身不提供告警模块, Argus[附录 12.17] (GitHub 0.29k 星) 是 Salesforce 开源的基于 OpenTSDB 的统一监控告警平台, 支持丰富的告警函数和灵活的告警配置, 可以作为 OpenTSDB 的告警补充。近年也出现一些轻量级的 TSDB, 如 InfluxDB[附录 12.18] (GitHub 12.4k stars) 和 Prometheus[附录 12.19] (GitHub 14.3k stars), 这些产品函数报表能力丰富, 自带告警模块, 但是分布式能力不足, 适用于中小规模企业。Grafana[附录 12.20] (GitHub 19.9k stars) 是 Metrics 报表展示的社区标配。

社区还有一些通用的健康检查和告警产品, 例如 Sensu[附录 12.21] (GitHub 2.7k stars), 能够对各种服务 (例如 Spring Boot 暴露的健康检查端点, 时间序列数据库中的 metrics, ELK 中的错误日志等) 定制灵活的健康检查 (check), 然后用户可以针对 check 结果设置灵活的告警通知策略。Sensu 在 Yelp 等公司有落地案例。其它类似产品还有 Esty 开源的 411[附录 12.22] (GitHub 0.74k 星) 和 Zalando 的 ZMon[附录 12.23] (GitHub 0.15k 星), 它们是分别在 Esty 和 Zalando 落地的产品, 但是定制 check 和 告警配置的使用门槛比较高, 社区不热, 建议有定制自研能力的团队试用。ZMon 后台采用 KairosDB 存储, 如果企业已经采用 KairosDB 作为时间序列数据库, 则可以考虑 ZMon 作为告警通知模块。

## 七、服务容错选型

针对 Java 技术栈，Netflix 的 Hystrix[ 附录 12.24] (github 12.4k stars) 把熔断、隔离、限流和降级等能力封装成组件，任何依赖调用（数据库，服务，缓存）都可以封装在 Hystrix Command 之内，封装后自动具备容错能力。Hystrix 起源于 Netflix 的弹性工程项目，经过 Netflix 大规模生产验证，目前是容错组件的社区标准，GitHub 上有超 12k 星。其它语言栈也有类似 Hystrix 的简化版本组件。

Hystrix 一般需要在应用端或者框架内埋点，有一定的使用门槛。对于采用集中式反向代理（边界和内部）做服务路由的公司，则可以集中在反向代理上做熔断限流，例如采用 Nginx[ 附录 12.25] (GitHub 5.1k stars) 或者 Kong[ 附录 12.7] (GitHub 11.4k stars) 这类反向代理，它们都插件支持灵活的限流容错配置。Zuul 网关也可以集成 Hystrix 实现网关层集中式限流容错。集中式反向代理需要有一定的研发和运维能力，但是可以对限流容错进行集中治理，可以简化客户端。

## 八、后台服务选型

后台服务主要包括消息系统，分布式缓存，分布式数据访问层和任务调度系统。后台服务是一个相对比较成熟的领域，很多开源产品基本可以开箱即用。

消息系统，对于日志等可靠性要求不高的场景，则 Apache 顶级项目 Kafka[ 附录 12.26] (GitHub 7.2k stars) 是社区标配。对于可靠性要求较高的业务场景，Kafka 其实也是可以胜任，但企业需要根据具体场景，对 Kafka 的监控和治理能力进行适当定制完善，Allegro 公司开源的 hermes[ 附录 12.27] (GitHub 0.3k stars) 是一个可参考项目，它在 Kafka 基础上封装了适合业务场景的企业级治理能力。阿里开源的 RocketMQ[ 附录 12.28] (GitHub 3.5k 星) 也是一个不错选择，具备更多适用于业务场景的特性，目前也是 Apache 顶级项目。RabbitMQ[ 附录 12.29] (GitHub 3.6k

星) 是老牌经典的 MQ，队列特性和文档都很丰富，性能和分布式能力稍弱，中小规模场景可选。

对于缓存治理，如果倾向于采用客户端直连模式(个人认为缓存直连更简单轻量)，则 SohuTv 开源的 cachecloud[附录 12.30] (GitHub 2.5k stars) 是一款不错的 Redis 缓存治理平台，提供诸如监控统计，一键开启，自动故障转移，在线伸缩，自动化运维等生产级治理能力，另外其文档也比较丰富。如果倾向采用中间层 Proxy 模式，则 Twitter 开源的 twemproxy[附录 12.31] (GitHub 7.5k stars) 和 CodisLab 开源的 codis[附录 12.32] (GitHub 6.9k stars) 是社区比较热的选项。

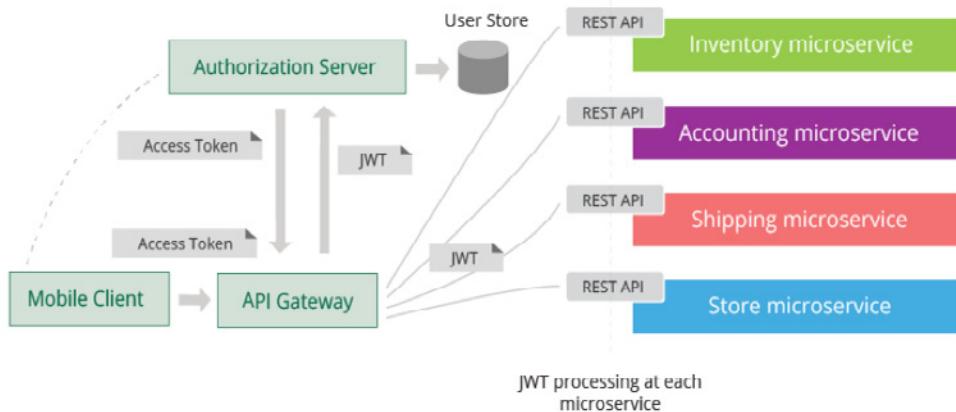
对于分布式数据访问层，如果采用 Java 技术栈，则当当开源的 shardingjdbc[附录 12.33] (GitHub 3.5k stars) 是一个不错的选项，分库分表逻辑做在客户端 jdbc driver 中，客户端直连数据库比较简单轻量，建议中小规模场景采用。如果倾向采用数据库访问中间层 proxy 模式，则从阿里 Cobar 演化出来的社区开源分库分表中间件 MyCAT[附录 12.34] (GitHub 3.6k stars) 是一个不错选择。proxy 模式运维成本较高，建议中大规模场景，有一定框架自研和运维能力的团队采用。

任务调度系统，个人推荐徐雪里开源的 xxl-job[附录 12.35] (GitHub 3.4k stars)，部署简单轻量，大部分场景够用。当当开源的 elastic-job[附录 12.36] (GitHub 3.2k stars) 也是一个不错选择，相比 xxl-job 功能更强一些也更复杂。

## 九、服务安全选型

对于微服务安全认证授权机制一块，目前业界虽然有 OAuth 和 OpenID connect 等标准协议，但是各家具体实现的做法都不太一样，企业一般有很多特殊的定制需求，整个社区还没有形成通用生产级开箱即用的产品。有一些开源授权服务器产品，比较知名的如 Apereo CAS[附录 12.37] (GitHub 3.6k stars)，JBoss 开源的 keycloak[附录 12.38] (GitHub 1.9 stars)，spring cloud security[附录 12.39] 等，大都是 opinionated (一家观

点和做法) 的产品，同时因支持太多协议造成产品复杂，也缺乏足够灵活性。个人建议基于 OAuth 和 OpenID connect 标准，在参考一些开源产品的基础上（例如 Mitre 开源的 OpenID-Connect-Java-Spring-Server[附录 12.40]，GitHub 0.62k stars），定制自研轻量级授权服务器。Wso2 提出了一种微服务安全的参考方案 [附录 12.45]，建议参考，该方案的关键步骤如下：



1. 使用支持 OAuth 2.0 和 OpenID Connect 标准协议的授权服务器（个人建议定制自研）；
2. 使用 API 网关作为单一访问入口，统一实现安全管理；
3. 客户在访问微服务之前，先通过授权服务器登录获取 access token，然后将 access token 和请求一起发送到网关；
4. 网关获取 access token，通过授权服务器校验 token，同时做 token 转换获取 JWT token；
5. 网关将 JWT Token 和请求一起转发到后台微服务；
6. JWT 中可以存储用户会话信息，该信息可以传递给后台的微服务，也可以在微服务之间传递，用作认证授权等用途；
7. 每个微服务包含 JWT 客户端，能够解密 JWT 并获取其中的用户会话信息。
8. 整个方案中，access token 是一种 by reference token，不包含用户信息可以直接暴露在公网上；JWT token 是一种 by value token，

可以包含用户信息但不暴露在公网上。

## 十、服务部署平台选型

容器已经被社区接受为交付微服务的一种理想手段，可以实现不可变（immutable）发布模式。一个轻量级的基于容器的服务部署平台主要包括容器资源调度，发布系统，镜像治理，资源治理和 IAM 等模块。

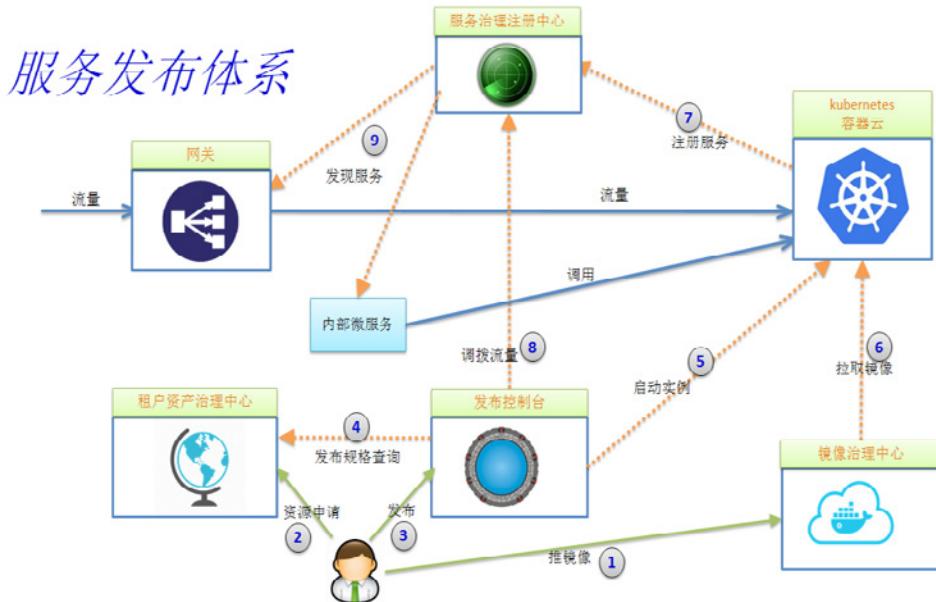
**集群资源调度系统：**屏蔽容器细节，将整个集群抽象成容器资源池，支持按需申请和释放容器资源，物理机发生故障时能够实现自动故障迁移（fail over）。目前 Google 开源的 Kubernetes[附录 12.41]，在 Google 背书和社区的强力推动下，基本已经形成市场领导者地位，GitHub 上有 31.8k 星，社区的活跃度已经远远超过了 mesos[附录 12.42]（GitHub 3.5k stars）和 swarm 等竞争产品，所以容器资源调度建议首选 K8s。当然如果你的团队有足够的定制自研能力，想深度把控底层调度算法，也可以基于 Mesos 做定制自研。

**镜像治理：**基于 Docker Registry，封装一些轻量级的治理功能。VMware 开源的 harbor[附录 12.43]（GitHub 3.5k stars）是目前社区比较成熟的企业级产品，在 Docker Registry 基础上扩展了权限控制，审计，镜像同步，管理界面等治理能力，可以考虑采用。

**资源治理：**类似于 CMDB 思路，在容器云环境中，企业仍然需要对应用 app，组织 org，容器配额和数量等相关信息进行轻量级的治理。目前这块还没有生产级的开源产品，一般企业需要根据自己的场景定制自研。

**发布平台：**面向用户的发布管理控制台，支持发布流程编排。它和其它子系统对接交互，实现基本的应用发布能力，也实现如蓝绿，金丝雀和灰度等高级发布机制。目前这块生产级的开源产品很少，Netflix 开源的 spinnaker[附录 12.44]（github 4.2k stars）是一个，但是这个产品比较复杂重量（因为它既要支持适配对接各种 CI 系统，同时还要适配对接各种公有云和容器云，使得整个系统异常复杂），一般企业建议根据自己的场景定制自研轻量级的解决方案。

IAM：是 identity & access management 的简称，对发布平台各个组件进行身份认证和安全访问控制。社区有不少开源的 IAM 产品，比较知名的有 Apereo CAS (GitHub 3.6k stars) , JBoss 开源的 keycloak (GitHub 1.9 stars) 等。但是这些产品一般都比较复杂重量，很多企业考虑到内部各种系统灵活对接的需求，都会考虑定制自研轻量级的解决方案。



考虑到服务部署平台目前还没有端到端生产级解决方案，企业一般需要定制集成，下面给出一个可以参考的具备轻量级治理能力的发布体系：

简化发布流程如下：

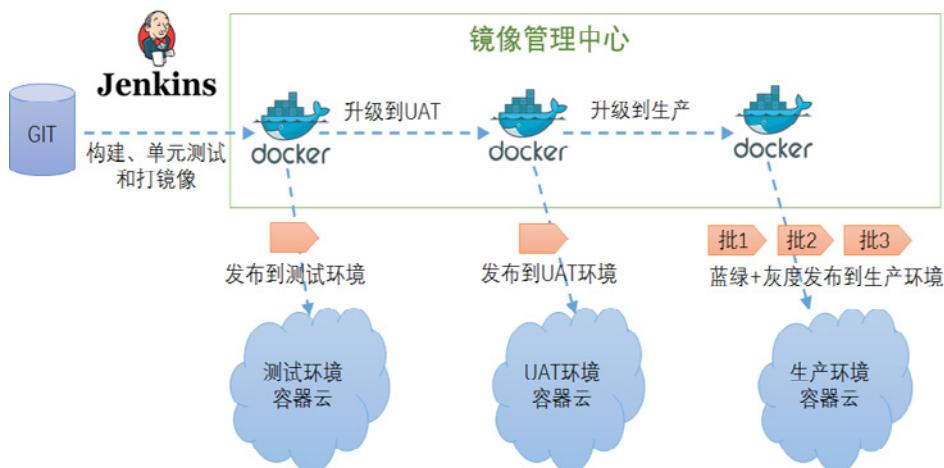
1. 应用通过 CI 集成后生成镜像，用户将镜像推到镜像治理中心；
2. 用户在资产治理中心申请发布，填报应用，发布和配额相关信息，然后等待审批通过；
3. 发布审批通过，开发人员通过发布控制台发布应用；
4. 发布系统通过查询资产治理中心获取发布规格信息；
5. 发布系统向容器云发出启动容器实例指令；
6. 容器云从镜像治理中心拉取镜像并启动容器；
7. 容器内服务启动后自注册到服务注册中心，并保持定期心跳；
8. 用户通过发布系统调用服务注册中心调拨流量，实现蓝绿，金丝。

雀或灰度发布等机制；

- 网关和内部微服务客户端定期同步服务注册中心上的服务路由表，将流量按负载均衡策略分发到新的服务实例上。

另外，持续交付流水线（CD Pipeline）也是微服务发布重要环节，这块主要和研发流程相关，一般需要企业定制，下面是一个可供参考的流水线模型，在镜像治理中心上封装一些轻量级的治理流程，例如只有通过测试环境测试的镜像才能升级发布到 UAT 环境，只有通过 UAT 环境测试的镜像才能升级发布到生产环境，通过在流水线上设置一些质量门，保障应用高质量交付到生产。

## 持续交付流水线



## 十一、写在最后

注意，本文限于篇幅，对测试和 CI 等环节没有涉及，但它们同样是构建微服务架构的重要环节，也有众多成熟的开源产品可选。

技术选型虽然重要，但还只是微服务建设的一小部分工作，选型后的产品要在企业内部真正落地，形成完整的微服务技术栈体系，则后续还有大量集成、定制、治理、运维和推广等工作。

本文仅限个人经验视角，选型思路仅供参考借鉴。每个企业的具体上

下文（业务场景，团队组织，技术架构等）各不相同，每个架构师的背景经验也各不相同，大家要结合实际自己做出选型，没有最好的技术栈，只有相对较合适的技术栈。另外，好的技术选型是相互借鉴甚至 PK 出来的，欢迎大家讨论，给出自己的微服务 2.0 技术栈选型意见。

## 十二、附录链接

[Spring Boot](#)

[Alibaba Dubbo](#)

[Google gRPC](#)

[NetflixOSS Eureka](#)

[Hashicorp Consul](#)

[NetflixOSS Zuul](#)

[Kong](#)

[Spring Cloud Config](#)

[CTrip Apollo](#)

[ElasticSearch](#)

[Yelp Elastalert](#)

[Dianping CAT](#)

[Zipkin](#)

[Naver Pinpoint](#)

[OpenTSDB](#)

[KairosDB](#)

[Argus](#)

[InfluxDB](#)

[Prometheus](#)

[Grafana](#)

[Sensu](#)

[Esty 411](#)

[Zalando ZMon](#)

[NetflixOSS Hystrix](#)

[Nginx](#)

[Apache Kafka](#)

[Allegro Hermes](#)

[Apache Rocketmq](#)

[Rabbitmq](#)

[Sohutv CacheCloud](#)

[Twitter twemproxy](#)

[CodisLab codis](#)

[Dangdang Sharding-jdbc](#)

[MyCAT](#)

[Xxl-job](#)

[Dangdang elastic-job](#)

[Apereo CAS](#)

[JBoss keycloak](#)

[Spring cloud security](#)

[OpenID-Connect-Java-Spring-Server](#)

[Google Kubernetes](#)

[Apache Mesos](#)

[Vmware Harbor](#)

[Netflix Spinnaker](#)

[Microservices in Practice – Key Architecture Concepts of an MSA](#)

# 云数据库 UDB 的三重境界

作者 Robert



## 前言

公有云服务本质上是用户和 IT 基础设施的连接器，通过打碎传统 IT 繁重的流程、低效的工作方式、不透明的价格以及糟糕的用户体验，重构出云主机、云数据库、云对象存储等产品，让用户更便捷地获取计算和存储能力，并保持使用习惯不变。

经过近十年的发展，一个越来越明显的趋势是公有云服务正从基于传统 IT 基础设施的包装和组合式创新，演进为围绕公有云场景、计算和存储能力的重新进化和升级。诸如容器云和 Serverless 架构、AWS Aurora 云数据库、UCloud 安全屋等，便是这一趋势的典型代表。

由此，我们可以对公有云的发展进程做一个两阶段的概括。云计算 1.0 的关键词是连接，通过互联网和公有云来连接用户和计算存储能力；而云计算 2.0 的关键词是进化，围绕公有云场景，重新看待全社会使用计算和存储资源的问题，对现有 IT 基础设施、模式做进一步的升级和进化。

站在云计算 1.0 向 2.0 进化和升级的档口，UCloud 云数据库团队希望通过这篇文章梳理过去、剖析当下、想象未来，以此来全面展现 UCloud 云数据库服务（UCloud DataBase Service，简称 UDB）能力，分享我们过去的经验和对未来的思考。

## 基因

考察一个云计算服务的发展犹如观察一颗种子落地后的生长。传统 IT 设施向云端变迁的趋势是云服务生长所需的阳光和雨露，但一颗种子能否长成参天大树，除了足够的阳光雨露，还要考察这颗种子的基因和成色。

在 UCloud 公司的四大价值观里，“客户为先”是放在首位的价值观。这体现了 UClouders 一以贯之的理念：只有为客户创造出真正的价值，企业才能够生存和发展。

创造真正的用户价值是 UCloud 所有产品的基因，也是 UDB 产品和云数据库团队的基因。对于 UDB 产品而言，创造真正的用户价值体现在两个方面：

### 需求驱动的产品研发和运营

需求驱动产品设计，技术评估实现可行性，必要时非标快速定制，定制逐渐沉淀为标准产品，整个过程循序渐进。小步快走，是互联网研发和运营的要领，也是公有云服务的要领。

以 UDB 跨地域跨可用区容灾为例，从单机版 UDB 开始，不断有用户因跨可用区容灾场景提出建跨机房从库的需求，中大型互联网客户尤为强烈。起初，以一种非标形式来提供能力的支持。后期因 VPC 2.0 上线，

技术也愈加成熟，现已将这种非标能力转化为标准能力，即多可用区高可用 UDB 产品，同时也将 UDB 由可用区级提升为地域级，产品形态得到一次质的提升，传统模式下需要付出极高成本才能构建的异地容灾方案，通过 UDB 产品可以轻松获得，用户价值进一步被创造。

## 一切以客户价值为归依，匠心铸造真正价值

云计算产品是 IT 基础设施类产品，技术人员在云服务的研发中起主导作用。但技术并不直接等同于用户价值。即使再先进的技术，离真正的用户价值还是会有一段距离。这段距离则需要用做产品的匠心来弥补。

所谓的产品匠心，非常重要的两点是对需求的洞察和对技术的取舍。技术人员常见的一个毛病是先入为主，将自己觉得酷的、牛的技术点等同于用户价值。但事实往往证明不一定。真正的用户价值创造，要打破技术人员思维的藩篱，洞察到用户需求的本质，从需求角度出发做技术选型，必要时敢于放下自己的喜好甚至利益，成就真正的用户价值。

以 UDB 产品的硬件架构选型为例。2013 年 UDB 立项之初，并没有选择云主机方案，而是选择了物理机 + Docker 的方案。如果基于云主机来构建 UDB，能够充分复用云主机成熟的能力，UDB 团队只需要关心硬件层面之上问题，同时降低研发成本，快速推出产品。但在 2013 年我们判断，当时的云主机对 IO 的优化还存在不足。具体体现为 IO 路径过长，管理层次太多，这些都将影响 IO 性能和 IO 稳定性。而 IO 性能和稳定性，恰好又是云数据库最重要的两个技术指标。因此，UDB 从一开始就选择了物理机 + CGroup 的架构，在 2014 年全面转向 Docker。事实证明，这是一个明智的选择。5 年以来，在各公有云厂商的云数据库产品性能对比上，UDB 每次都是完胜。

## 三重境界

王国维在《人间词话》二六节写到：古今之成大事业、大学问者，必经过三种之境界。“昨夜西风凋碧树，独上高楼，望尽天涯路”，此第一境

也。“衣带渐宽终不悔，为伊消得人憔悴”，此第二境也。“众里寻他千百度，回头蓦见，那人正在灯火阑珊处”，此第三境也。此等语皆非大词人不能道。然遽以此意解释诸词，恐晏、欧诸公所不许也。

UDB 的成长之路，也经历三个阶段，细分为三重境界。这三个阶段互相独立，又存在一个内在的逻辑，将它们牢靠地连接在一起。这个内在逻辑就是 UDB 的基因：创造真正用户价值。UDB 在每一个阶段的萌芽、发展、跃迁，无一不是这个基因和理念在发挥作用。

### 1. 做透一个点：取代自建数据库

UDB 产品第一阶段要比拼的是能否比用户自建数据库（基于云主机或者自建 IDC），具备更大的用户价值。只有创造出更大价值，形成更高的价值势能，才能吸引用户将业务迁移到云数据库。所以 UDB 的第一个目标就是把“取代自建数据库”这一个点给做透。

### 2. 构建功能网：全方位覆盖用户需求

过去几十年来，围绕 DBMS 出现了从容灾、迁移、安全到读写分离、数据拆分等解决方案和软件，对应用业务的各种需求。这些解决方案和软件同样需要云化，并且需要利用公有云的优势产生比自建更大的价值。如此，才能不断强化云数据库的价值势能，服务好已有用户并吸引更多用户向公有云转化。

因此，UDB 产品第二阶段要做的是构建一张云数据库功能网。在第一阶段的基础上，继续将用户需要的各个功能点做透。众多功能点以及功能点的组合，最终构成一张大网，全方位地覆盖用户的各种需求。

### 3. 三位一体融合平台：云计算 2.0 下的内生进化

第一阶段和第二阶段，对新价值的创造都是基于成熟的软件或解决方案，利用公有云来实现功能的随手可得、快速部署和弹性扩展。这种模式清晰明确，但并不意味着云数据库价值创造的终点。

云计算 2.0 时代，公有云开始摆脱传统 IT 基础设施和软件的藩篱。在

产品和技术上，围绕自身业务场景开启独立进化。其中，如何解决全社会大规模用云时的成本、效率和智能问题，是这场进化的核心。而 UCloud 云数据库团队也需要进一步去思考，是否能提供更加廉价优质、高效智能的云数据库产品。带着问题和思考，UCloud 云数据库团队内部做了多次探讨，最终达成这样一个认知：云计算 2.0 下的云数据库服务，会是数据库 PAAS 化，运维智能化，以及结构化数据处理生态体系这样三位一体的组合。

下文我们将对做透一个点、构建功能网、三位一体融合平台展开详细介绍，用具体的例子来勾勒 UDB 发展的三重境界。

## 做透一个点：取代自建数据库

取代自建数据库，看来似乎简单，但逐一罗列并剖析需要考虑的五个价值点：

- a. 可靠性
- b. 稳定性
- c. 高性能
- d. 零维护
- e. 性价比

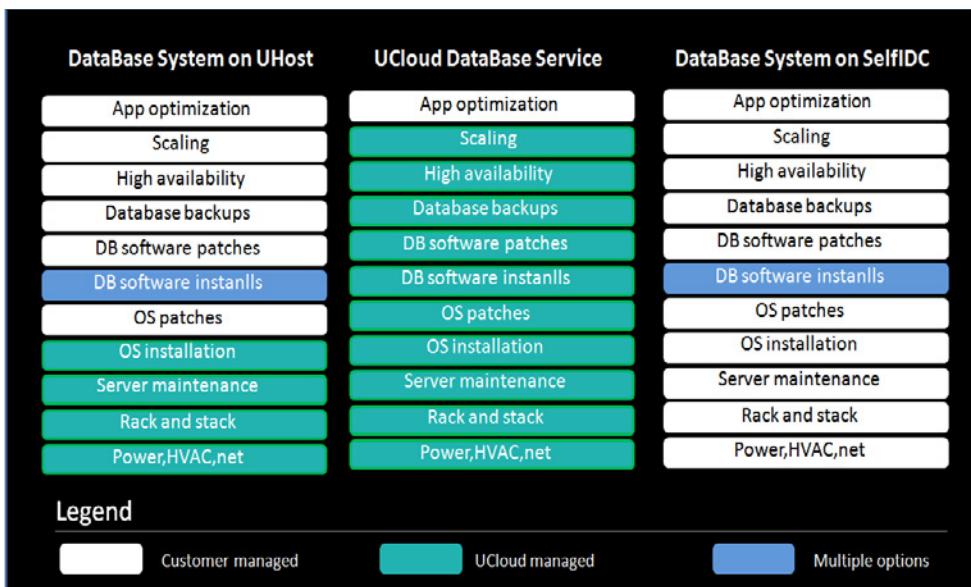
你会发现要做好绝非易事。UDB 产品经过几年的努力，完美地实现了做透一个点：取代自建数据库 这一目标。

### 可靠性

云数据库的可靠性强调数据安全性包括两方面：一是 DB 数据；二是备份数据。DB 数据落盘的持久性通常要求 99.9999% 及以上，表明数据保持存储状态不丢失的概率。此类数据主要是指用户存储在数据库中的数据，不包括缓存和临时存储。DB 数据本地盘采用 RAID10 或者 RAID50 做好冗余，若是高可用机型，则再有实例级冗余。备份数据要求异地存储，多副本存储。

## 稳定性

这里强调的是单机稳定性。我们可以看下如何自建一套数据库，在数据中心的电力、物理网络、机架、物理服务器等基础设施之上，部署操作系统和补丁，安装数据库软件和补丁，运行数据库软件，启用数据库服务。如果是采用虚拟化部署，则额外涉及计算、网络、存储虚拟化。这是一套庞大的系统，各个环节都存在不可预知的故障风险。UDB 经过多年的运营积累了诸多经验，在多方面多层次保障其足够稳定。



## 高性能

如何通过软硬件结合使单机数据库的性能发挥到极致？高性能 UDB 机型底层采用 PCI-E/NVMe SSD 存储硬件，定制化宿主机 Linux 内核专门适配最新型硬件。采用自研 IO 调度算法，可良好保障实例级的 IO 隔离。数据库层面通过参数调优、内核定制优化，使数据库发挥出最优性能。通常情况下，数据库的性能瓶颈会出现在磁盘 IO。采用虚拟机自建存在诸多弊端，例如 IO 路径过长、IO 稳定性较差、IO 竞争等。UDB 采用高性能物理机 + Docker 架构 + 自研 IO 调度算法，打造出强劲的 IO 性能，持续保障稳定性和隔离性。

## 零维护

通常情况下，数据库是后台服务框架里最为核心的组件，重要性不言而喻，日常运维工作慎之又慎。在第一个阶段，UDB 提供的是数据库的全托管运维能力，包括一键部署、保活、容灾、备份、恢复、迁移、配置、漏洞修复、升级、监控与告警、巡检等等一系列的后台运维类操作，解放了客户的 DBA 人力 / 精力。本身在 UDB 产品上集成了上述多数的控制台操作，使客户对数据库基本可控。

## 性价比

客户对云数据库买不买账，性价比成为非常重要的因素。可靠、稳定、高性能、高可用、零维护等基础能力作为 UDB 的价值基础，在 UDB 产品上更是提供丰富的配置组合，自定义存储（普通盘 or SSD 盘）、内存大小、VPC 网络、可用区等，灵活多配，按需付费，一键交付。按业务增长，弹性扩容。客户完全省去自建数据库的一切环节，规划 IDC，规划资源、采购、上架、交付部署以及后期一切运维工作。对于一些商业数据库（如 SQL Server）尤其划算，省去自购官方 License 费用。

## 构建功能网：全方位覆盖用户需求

云数据库发展第一阶段的目标是把“取代自建数据库”这个价值点做透，创造出区别于传统方式的全新价值。在第二阶段则需紧扣用户使用数据库的痛点，有针对性地推出公有云解决方案，构成一张功能网，全方位覆盖用户需求。我们把第二阶段用户的痛点需求归结为三类。

### 高可用和容灾

数据库的高可用是大型 IT 系统的必备机制，但不少系统在建设时，受限于自身基础设施和能力，无法实现优质可靠、高等级的数据库高可用。

而这一点恰好是公有云的优势。首先是完善的基础设施。公有云厂商拥有大量的数据中心，数据中心之间通过跨可用区，乃至跨地域的光纤专线进行连通，构成一张覆盖全国乃至全球的高性能网络；第二，公有云的

规模效应将带来成本优势，能够显著降低基础设施的建设成本；最后，规模效应也让技术团队被超大力度地锤炼，技术和运营水平不断提升。基于以上三点原因，公有云是实施数据库高可用的优良环境。

## 容量和性能

大数据时代，全社会数据量在高速增长。原因之一是生产和生活日益互联网化，产生大量数据；原因之二是数据的价值越发得到重视，企业普遍希望能够沉淀并分析数据、从数据中挖掘出价值。

而传统的单机数据库系统，必然随着大数据时代的到来，在容量和性能上遭遇挑战。如何应对这两大挑战，且总体保持 IT 成本可控，是令客户越来越头痛的问题。

## 运维和安全

运维和安全是云数据库研发和运营永远的课题。需要围绕用户需求构建运维闭环，从数据导入导出、数据库运行期管理 / 问题排查，到性能分析、数据库优化，让用户不使用任何第三方运维工具，即可将云数据库运维工作全部搞定。在安全上，除了 DBMS 自身的用户访问控制机制之外，还应该围绕审计、加密、防恶意攻击等方面为用户提供优质的产品或功能，构建数据库安全运营闭环。

下面将列出 UDB 围绕高可用和容灾、容量和性能、运维和安全三个方面，构建的众多功能点。这些功能点和功能点的组合构造出一张大网，全方位覆盖用户需求。

## 高可用和容灾

功能点	说明（以 MySQL 产品为例）
同可用区高可用	<p>1. 主备 UDB 节点部署在同一区域的不同可用区（光纤专线联通），节点间采用自研增强型半同步复制保持数据一致性。</p> <p>2. 主节点异常后，由上层 Proxy 将备节点升级为主节点。Proxy 亦做跨可用区部署，Proxy 的容灾和同可用区高可用保持一致。</p>

跨可用区高可用	在可用区高可用的容灾能力基础上，将UDB备节点和上层备Proxy部署到同区域不同可用区，做到跨可用区高可用
三节点强一致高可用集群（内测中）	1.基于分布式一致性算法的三节点高可用 MySQL 集群，数据可靠性16个 9，系统可用性和数据可靠性兼具，有效满足金融行业高可用和数据高可靠的需求。 2.搭配异地灾备节点，即可实现两地三中心的金融数据库解决方案。
跨可用区从库	从节点和主节点部署在同一区域的不同可用区，节点间采用异步复制保持数据一致性。
跨区域从库	从节点和主节点部署在不同区域，节点间采用异步复制保持数据一致性。

## 容量和性能

功能点	说明
UDB 读写分离中间件	1.业务透明访问，对MySQL接近100%兼容； 2.极致性能，SQL转发能力比ProxySQL高出25%； 3.中间件节点双活部署高可用； 4.提供指定SQL路由、用户权限控制等管理类SQL； 5.永久免费。
分布式数据库UDDB	1.高性能分库分表，分库分表场景下 Sysbench 测试性能相比竞品高出2 - 3倍； 2.基于二级分区的灵活强大的水平分表机制轻松应对各种业务的数据库水平划分难题； 3.强大的单表查询和基本的多表 Join 功能，以及支持TokuDB压缩引擎，有效应对物联网大数据分析场景下的海量数据分析问题； 4.兼容主流语言的 SQL API 和数据库访问组件，支持Navicat 等图形化客户端管理工具；
数据库压缩引擎TokuDB	在性能不损失的前提下，数据压缩率在 3 倍以上，甚至可以高达 10倍。

## 运维和安全

功能点	说明
在线迁移 DTS	数据库在线迁移工具 DTS 提供自建数据库到 UDB、UDB 到自建数据库、UDB 到 UDB 的迁移功能，运行稳定、使用方便。
数据库审计	对数据库操作进行细粒度分析和审计，提供实时监控、违规发现、历史行为回溯等操作分析功能
DBAMaster（自研中）	该系统沉淀了 UDB 团队 DBA 在数据库系统运维上沉淀的经验和知识，将为开发团队提供数据库的自助化诊断优化，帮助开发团队实时掌控数据库系统运行情况，及时甚至提前发现数据库异常，协助对数据库的优化，持续跟踪优化效果
存储加密（自研中）	在存储引擎层对数据库文件、RedoLog 文件进行加密，加密操作对业务透明。

## UDB 产品全景图

经过一、二阶段的发展，UDB 产品已经成为基础扎实、品类完善、功能全面的一个云数据库产品体系。UDB 产品的全景图如下：



## 三位一体融合平台：云计算 2.0 下的内生进化

UDB 产品在第一、二阶段的成长和发展是基于公有云场景，围绕成熟的数据库软件和解决方案来为用户创造使用价值。这种模式清晰明确，但也存在不少短板。

1. 最大的问题在于传统的数据库软件的架构和代码已不适应公有云发展的需要。传统数据库在容量和性能提升、容灾、最新硬件的利用上都存在不足。

2. 用户对云数据库提出的新需求，传统数据库已不能满足。比如按需付费、大数据量的高速备份、更短的灾难恢复时间、OLTP 和 OLAP 融合、异构数据库等问题，传统数据库没有很好的解决方案。

3. 数据库运维的方法仍然传统。当数据库实例和客户量达到一定的规模后，传统的人工运维的方法已变得不切实际。

我们不可能用产生问题的方式去解决问题。上述问题来源于传统的数据库架构和运维方式，因此不再可能用传统的方式去解决。唯一解决之道在于依托云计算 2.0，实现云数据库的内生进化。摆脱传统模式的窠臼，开创云数据库的新境界。

在云计算从 1.0 向 2.0 跃迁的关键时间节点上，UDB 团队提出未来发展的三位一体战略来刻画未来云数据库的技术和产品形态，满足未来客户的普遍需求：

- a. 数据库 PAAS 化
- b. 运维智能化
- c. 结构化数据处理生态体系

上述三个支点构成一个云数据库 2.0 体系，有力支撑 UDB 未来的发展。

### 数据库 PAAS 化

从用户使用的角度，IAAS 和 PAAS 的区别在于，IAAS 需要用户为租用的资源实例付费（即使资源没有被 100% 使用），而 PAAS 只需要为资

源实际使用量付费。PAAS 在 IAAS 的基础上，进一步降低了全社会 IT 使用成本，符合公有云发展的终极理想：让 IT 像水和电一样被人类社会使用。

像使用水和电一样使用数据库，这个理想看上去很美好，但很难实现。要做到像水和电一样使用数据库，必须解决两个关键问题：

1. 容量和性能根据业务需求弹性扩展
2. 存储和计算能力按实际使用量计费

第一个问题既目前热门的分布式数据库问题，而第二个问题长期以来只是一个模糊的目标，可望而不可及。但近几年来，计算和存储分离的架构理念，结合新型硬件或创新的数据库 IO 优化方法，为解决这两个问题带来希望。

计算和存储分离，就是将数据存储下放到分布式存储系统，数据库实例只保留 SQL 执行和事务处理等计算类操作，计算层和存储层通过网络交互彻底解耦。历史上，计算和存储分离架构的产品有存在（如腾讯云 CDB 第二代产品），但并未成为主流，其原因在于将计算和存储分离后，数据库的性能上不去，导致实际价值不大。但近几年来，业内分别从两个不同角度，突破了这一瓶颈：

1. AWS Aurora 创造性地裁剪数据库内核，只落地 RedoLog 而不落地 Page，从而减少了持久化数据的写入量，IO 开销大为减少。

2. 阿里 PolarDB 利用高速网络和新型硬件，优化了数据库内核的 IO 路径，实现了通过网络的分布式 IO，和本地 IO 同样的延迟时间和更高的吞吐量。

由此，云数据库正式进入 2.0 时代，既计算和存储分离时代，向 PAAS 化开启大步演进。以 AWS 2017 年 11 月份推出的 Aurora Serverless 为例，目前已经做到了根据业务压力实现动态伸缩调配系统资源，存储能力和读写性能得以动态扩展；做到了根据业务实际数据量和读写 QPS 进行计费，从而实现像使用水和电一样使用数据库。

而 UDB 从 2017 年下半年开始，已经启动计算和存储分离架构的新一代数据库的开发，迈出构建 UDB 2.0 的关键一步，后续产品敬请期待。

## 运维智能化

一般企业的 DBA 只需要为本公司数据库系统服务，而公有云 DBA 需要为上万家企业服务，工作量不言而喻。随着客户和数据库实例的增多，必须要通过自动化、智能化的方式来帮助 DBA 进行在线实例的维护，让 DBA 从繁重琐碎的日维护工作中脱离出来，把精力放在更高阶、更重要的事情上。

DBA 工作智能化的过程总的来说分为三个阶段： 1. 原始手工运维； 2. 重复性工作自动化； 3. 结合机器学习的运维智能化。

在手工运维阶段，DBA 凭借自己的经验借助一些半自动化的工具，完成云端数据库实例的日常运维、故障分析和解决、性能调优工作；

在重复性工作自动化阶段，云数据库团队将建设体系化和自动化的 DBA 运维系统，收集数据库实例各层面的系统状态和性能数据，构建数据分析平台进行数据分析，判断或预判有问题或有潜在问题的数据库实例，以及通过预设规则进行数据库故障的处理、防范或告警。

在 DBA 工作智能化阶段，将利用大数据分析和机器学习的一些技术去进一步增加 DBA 自动化运维系统。比如，细时间粒度的数据库实例异常预警、数据库故障自动诊断和处理等。

## 结构化数据处理生态体系

数据库 PAAS 化 + 运维智能化，这两个目标的实现，可以说实现了公有云数据库从业者一直以来追求的圣杯：一个根据业务压力自动弹性伸缩、按需付费且运维自治，仅需少数人工干预的云端结构化数据存储和处理平台，让用户真正得以向水和电一样使用数据库。但这还不够。

事实上，用户对结构化数据的存储和处理时复杂的，有时候还是非常个性化的。回顾对象存储 PAAS 平台的发展历程，一开始是解决文件的存取问题，随后开始提供对文件的加工处理能力（比如转码、压缩、鉴黄等），最后基于该平台结合其他技术（比如 AI），进一步延伸诸如机器视觉、

图像知识提取等功能，不断满足用户的需求。

因此，对于一个结构化数据处理 PAAS 平台而言，必然也需要随用户需求而动，不断向上生长，添加更多能力。更有必要的是打造一个开放的、共赢的结构化数据处理生态体系，引入全行业和全社会的力量，来建设一个拥有涵盖各行业用户需求，有着勃勃生机同时稳定可靠的结构化数据存储和计算生态系统，进一步解放全社会的生产力。

## 结语

通过公有云为用户创造比传统 IT 更大的价值，这是 UCloud 云数据库团队开展工作的出发点。云计算和公有云的高速发展最根本的原因不在于有多前沿的技术、多便宜的价格，而是在于通过一个个产品和功能的创新和创造，不断产生新的用户价值，在真实用户需求的助推下发展壮大。UDB 推出 5 年以来，UCloud 云数据库团队一直秉持这一理念，伴随云计算发展大潮经历三重境界，任凭岁月更迭而初心不改。

## 作者简介

Robert，UCloud 资深数据库开发工程师

2007 年毕业于华中科技大学数据库和多媒体技术研究所，曾先后在达梦数据库、腾讯从事过多年数据库内核和分布式后台服务的研发工作，目前专注于分布式数据库服务的研发和运营，UCloud UDDB 产品和技术负责人。



# 微服务架构

---

## 核心20讲

从理论到实践  
理解微服务关键问题



扫一扫，试看课程



# GMTC 2018

## 全球大前端技术大会

—— 大 前 端 的 下 一 站 ——

聚集12大热门专题

UI与动画

PWA

终端AI

工程化

语言专场

Weex

性能优化

WebRTC

Android专场

iOS专场

Node专场

Web框架

时间：6月21-22日 地点：北京·国际会议中心

票务咨询

微信：18514549229  
Q Q：209463896

4月4日前购票，**6折**优惠  
(团购享更多优惠)

扫码了解更多信息





## 架构师 月刊 2018年2月

本期主要内容：英特尔、谷歌和 AWS 回应 CPU 安全事件：AMD 和 ARM 也有问题，影响巨大；开发者需要知道的有关软件架构的五件事；中小型研发团队架构实践：如何规范公司所有应用分层？阿里盒马领域驱动设计实践。



## 深度学习器： TensorFlow程序设计

本书详细介绍了 TensorFlow 程序设计中的几个关键技术。



## AI前线特刊： AI领域2017进展总结

AI 前线在 2018 年之初为各位读者奉上这样一本迷你书，涵盖了来自全球 AI 和大数据领域技术专家的年终总结与趋势解读，同时还有世界知名技术大厂的年终技术总结与趋势预测。



## 架构师特刊 范式大学

构建商业 AI 能力的五大要素；判别 AI 改造企业的 70 个指标；用最小成本 验证 AI 可行性；企业技术人员如何向人工智能靠拢？人工智能的下一个技术风口与商业风口。