架构师
ARCHITECT

ArchSummit
全球架构师技术峰会
特刊

ArchSummit 全球架构师峰会 | Geekbang 极客邦科技 InfoQ

卷首语

如何培养
架构设计思维

拍拍贷总监 杨波

良好的架构设计思维的培养，离不开工作中大量高质量项目的实战锻炼，然后是平时的学习、思考和提炼总结。

另外，基本的架构设计思维，其实在大学计算机课程（比如数据结构和算法）中可以找到影子，基本的架构设计思维在那个时候就已经埋下了种子，后面工程实践中进一步消化和应用。

一个架构师的成长高度和他大学期间的思维习惯的养成关系密切。如 Google 等世界一流技术公司在招聘工程师新人时，对数据结构和算法的要求可以用苛刻来形容，这个可以理解，谷歌级别公司要解决的问题都是超级复杂的，基本思维功底薄弱根本无法应对。

对于演化设计思维，当前大学教育大都采用脱离现实场景的简化理想模型，不利于培养演化式设计思维。我个人的体会，演化式设计思维更多在实际工作中通过实战锻炼和培养。

这里总结一些思想经验。

1. 架构的本质是管理复杂性，抽象、分层、分治和演化思维是架构师征服复杂性的四种根本性武器。
2. 掌握了抽象、分层、分治和演化这四种基本的武器，你可以设计小到一个类，一个模块，一个子系统，或者一个中型的系统，也可以大到一个公司的基础平台架构，微服务架构，技术体系架构，甚至是组织架构，业务架构等等。
3. 架构设计不是静态的，而是动态演化的。只有能够不断应对环境变化的系统，才是有生命力的系统。所以即使你掌握了抽象、分层和分治这三种基本思维，仍然需要演化式思维，在设计的同时，借助反馈和进化的力量推动架构的持续演进。
4. 架构师在关注技术，开发应用的同时，需要定期梳理自己的架构设计思维，积累时间长了，你看待世界事物的方式会发生根本性变化，你会发现我们生活的世界也是在抽象、分层、分治和演化的基础上构建起来的。另外架构设计思维的形成，会对你的系统架构设计能力产生重大影响。可以说对抽象、分层、分治和演化掌握的深度和灵活应用的水平，直接决定架构师所能解决问题域的复杂性和规模大小，是区分普通应用型架构师和平台型/系统型架构师的一个分水岭。

Geekbang. 极客邦科技 | 极客时间

从0开始学架构

资深技术专家的实战架构心法 50讲

你将获得

1. 理解架构设计的本质和目的
2. 掌握高性能和高可用架构模式
3. 走进 BAT 标准技术架构实战
4. 从编程到架构，实现思维跃迁

扫码了解专栏

目录

架构师 ArchSummit 特刊

本期主编　徐川

流程编辑　丁晓昀

发行人　霍泰稳

联系我们

提供反馈　editors@cn.infoq.com

商务合作　hezuo@geekbang.org

内容合作　editors@cn.infoq.com

社交软件中轨迹追踪、定向圈人背后有什么数据难题？

作者 宾理涵

从 2004 年地图应用开始流行之时，很多人并不会了解到地图会成为未来各大互联网企业尤其 020 企业必备的基础能力之一，即使是大数据技术如此成熟的今天，诸如“导航”、“附近的人”等基础功能的背后大规模时空处理也并不是容易的事情。

以 Facebook 为例，2017 年 Facebook 已经覆盖全球 130 多个国家和地区，直逼 20 亿用户，在地理位置数据处理方面的技术（包括轨迹追踪、定向圈人、GEO 相关研究）Facebook 在全球名列前茅。

万亿级数据处理背后，Facebook 是如何设计数据库索引结构、查询优化、数仓建模等难题的，InfoQ 采访了 Facebook 的 GEO 组 Tech Lead 的宾理涵。

InfoQ：您曾在 Qualcomm（下译为高通）工作过，在你看来高通与 Facebook 的工程师文化有哪些异同？Facebook 的哪些特性吸引您加入？参与 OpenCL 标准制定有哪些不一样的体验？

宾理涵：目前我在 Facebook 担任 GeoAPI 组的 Tech Lead，以前曾在标准制定组织 Khronos Group 任高通的代表。

Facebook 最吸引我的文化是开放：大家共享同一代码库。公司鼓励员工自由地选择内部调动，做你最感兴趣的项目。一般工作年限满一年的员工会收到内部邮件鼓励参加 hackamonth，即去别的组工作一个月，期满后工程师可自行决

定是否留在新的团队。

另外，Facebook 的软件工程师没有等级 title，所有工程师统一叫做 Software Engineer，这极大地降低了对话的门槛，你可以毫无压力地和业界最顶尖的程序员交流技术问题。

至于高通，它的工程师文化相对而言更像传统的计算机公司，有严格的开发流程，毕竟硬件不像软件那样可以轻易的升级换代。

参与 OpenCL 标准制定的经历很有意思，你面临的不是公司内部的资深员工，而是业界巨头例如 Intel，Google，NVIDIA 等的资深架构师。他们是这个行业最顶尖的专家，对 GPU 软件和硬件的发展趋势都非常的熟悉。

作为公司代表，你的任务是代表公司的利益，根据公司长期的 roadmap，提出新的软件、硬件功能的 proposal，并试图让别的公司的代表通过你的提案。我们每周有电话会议，每季度有面对面会议。从草案到发布周期可以长达一年，甚至更久。

以 OpenCL 举例，整个过程简单来说分三步：

- 第一步是 working group 内部讨论
- 第二步是各公司内部审核
- 第三步是投票，如果投票通过，那么该方案将加入下一代的 OpenCL 标准

除了 OpenCL，著名的 OpenGL 及其后续版本 Vulkan 都是 Khronos Group 旗下的标准，该组织非盈利，主席由各个公司轮番担任。

InfoQ：Facebook 大规模时空数据处理中，有哪些 quirks 问题？为什么需要解决这些问题？

宾理涵：我目前在带领团队开发 Geospatial Indexing 平台，该平台能将地理大数据和实时数据流检索导入以提供实时的查询和计算。

处理大规模空间数据除了有在规模上的挑战以外，还有一些独特的问题，举个例子：

作为一个没有做过空间数据的人，第一个常犯的错误就是两个坐标点的距离，如果你简单地计算了欧几里得距离，那么你最后在产品中看到计算出来的点肯定是错的。同样的，相同经纬度差在赤道和两级在距离上有很大区别。

第二个是空间数据的分布问题，大城市密度比小城市密度高，这在多个层面影响你的算法。从分布式的角度，你的数据怎么分布延时最小。从数据结构设计上，怎么处理地图 zoom level 问题。从产品的角度，计算多少米的半径才是用户愿意去的。

最后就是工具的缺失，传统大数据分析不是为空间数据优化的，通常要 JOIN 两个 HIVE 表格会花很长的时间而且答案不一定正确。

InfoQ：Facebook 大规模时空数据处理的技术栈是怎样的？用了哪些编程语言、算法、开源软件项目、工具等？其中有哪些有趣的经历？

宾理涵：我会在 ArchSummit 深圳站分享《Facebook 万亿级混合复杂时空数据的处理决策》中详细介绍我们 Location Infrastructure 团队的技术栈，这里简单介绍概括一下：

数据来源于 HIVE 和 Scribe；

- 离线查询使用 Presto 加上专门的 spatial join 的支持，在线查询通过 indexer 建立索引提供低延时实时查询；
- 在线系统基于 Facebook 的 Thrift 框架提供高性能高并发来实现海量查询；
- 实时流数据我们采用 Facebook 开发的 Stylus 平台。

我们大多数系统采用 C++ 开发。算法上

RTree，QuadTree 用的比较多。我们使用开源的 Google S2 做坐标编码。我们的在线 index 系统经历了一系列的架构升级，从最简单的静态 In-memory 到现在的分布式，多存储形式，支持实时数据更新。

在整套体系的建设过程中，我们在内部技术和开源技术之间采取了折中而务实的办法。Facebook 推崇 hacker 文化，我们的项目来源于 hackathon，即几个程序员聚在一起 hack 好几天，来验证一个项目构想。

我们第一个项目的产生完全是因为一个产品组需要高性能的 circle index，而现成的系统虽然功能非常强大但是对几何图形支持却非常有限。所有我们用 boost rtree 写了一个原始版本没想到性能居然超过了别的所有方案。

一般程序设计初期，我们先制定 MVP，不会过度设计。随着我们 index 平台的用户越来越多，我们开始针对一个又一个的产品需求进行优化，最终演变到了现在产品的样子。

InfoQ：时空数据相比其他类型的数据，在处理方法上是否有异同？在任职 GeoAPI leader 期间有什么技术挑战和非技术挑战？

宾理涵：任职 Tech Lead 期间，遇到最大的挑战是帮助 Facebook Marketplace 的全球发布。当时正值我们一个版本的架构遇到了 scale 的瓶颈，加之产品组的 Business Logic 嵌入我们的代码太深导致我们不能很快的迭代，影响别组的进程。

我们有两个选择，要么继续打补丁，让友组在我们的 codebase 里面继续开发，要么直接开发下一代架构，当时我们从几个方面去研究了这个问题：

- 第一，由于我们初期版本架构瓶颈，虽然一系列代码优化减少了内存消耗，最终（一年以后）无法满足像 Marketplace 这样高速发展的产品的需求。
- 第二，我们组初期成员数量和 Marketplace 这样的大型部门成员数量相差悬殊，如果不开发新系统做平台化，把 Business Logic 尽快从我们的系统剥离，是无法满足多倍于我们组人数的各个 Marketplace 产品组的需求的。
- 第三，平台化有利于接受更多的产品组基于地理平台做产品开发。事实证明平台化后我们的平台用户增长数量比以前大一个数量级。

虽然诸多好处，最大的问题在于新系统开发周期比旧系统打补丁时间长很多。所以我们和 Marketplace 部门合作把内存占用最高的模块 ML model 临时移到了另外一个简单的 KV index。以延时 (higher latency) 换取我们的开发时间。

考虑上述诸多因素后，我们最后直接开发新系统。新系统在 PHP 层提供了类似 SQL 的语法来替代以前类似于插件模式（客户代码和我们系统一起编译发布）。这个决定从根本上定义了清晰的架构界线，从而让两个组都可以各自高效的向前推进。

InfoQ：关于大规模空间数据，你们是如何考虑降维或特征选取的？

宾理涵：通常空间数据只是高位数据的其中一维。怎么整合空间数据到多维数据处理是我们常遇到的问题。例如我们的图搜索引擎把多个根据 social graph 连接的文档用 inverted index 来保存，怎么把空间信息也加入到这个图中就是一个有趣的挑战。

从大数据分析上，Spatial join 也是我们处理高纬度数据需要解决的问题，例如多个 HIVE 表格要根据空间数据进行连接，怎么通过优化建立空间索引让 JOIN 更高效。

从 ML 层面上，时空数据怎么表达，怎么用现成的 Caffe2 library 来挖掘时空数据的价值都是我们遇到的问题。

InfoQ：万亿级别的数据是如何进行存储和传输的？谈到开源，如何看待 Oracle 与 Google 的 Java 诉讼？

宾理涵：Facebook 的 data warehouse 使用了很多开源系统，例如 Hadoop，HIVE，presto，spark，有些是我们自己开发，然后开源的给社区的，有些是直接来自于开源社区，然后我们自己进行修改。

大部分 Facebook 的数据产生于我们的图数据库 TAO，以及一些实时的 log，TAO 数据存储于 MySQL，然后定期发布到 HIVE，log 数据通过我们自己的流平台 Scribe 做数据发行，下游系统例如 PUMA 可以做实时计算，FBETL 可以将其导入 HIVE。

数据存储层 Facebook 有使用 ORC 格式对 HIVE 表格进行压缩优化。

传输层使用最多的是 thrift 系统，该系统提供序列化和反序列化。Thrift 可以很好的支持数据 schema 的变化。

我们不便做出评论 Oracle 与 Google 的 Java 诉讼，但是这不会影响我们选择开源产品。Facebook 也会继续拥抱开源社区，把更多优秀的内部孵化的平台系统开源。

InfoQ：技术更新迭代速度之快，你们是如何获取与工作相关的前沿技术的呢？如何看待近期的人工智能？是否有考虑引入相关的技术来解决问题？

宾理涵：我很幸运 Facebook 遇到的挑战都是独一无二，同事们研究的话题也是业界最前沿的。作为 Location Infra，我们有跟各个组整合的机会，例如 location 在 AR/VR 中的应用，location 在 ML 中的应用等等。

关于近期火热的人工智能技术，我们公司内部的 FBLearner ML 系统被高达一半的员工使用，ML 已经在我们工作的方方面面了，从 GBDT，到 Logistic Regression 到 CNN，这些算法帮助 Facebook 的员工开发更智能的产品。目前时空数据和 ML 的整合是我们组现在正在研究的课题。

宾理涵，毕业于美国佛罗里达大学硕士，就职于 Facebook 的 GeoAPI 组担任技术领导。带领团队开发了 Geospatial Indexing 平台，将地理大数据和实时数据流检索导入以提供实时的查询和计算，该平台已经用于多个面向用户的产品。他还参与了 Facebook 搜索引擎中的地理查询的设计和开发。此前就职于 Qualcomm，任 Qualcomm 在标准制定组织 Khronos Group 的代表，参与 OpenCL 标准制定。

在架构师眼里，一份美团外卖是如何做出来的？

作者 方建平

写在前面

2018 年 4 月，中国外卖市场迎来巨变，外卖从无人问津开始，到现在已经培育成互联网巨头必争之地。作为为数不多能够达到日千万订单级别的业务，其后端服务是怎么支撑的？ InfoQ 采访 ArchSummit 出品人、美团点评技术总监方建平，请他回顾及展望美团外卖的后端架构史，本文根据采访整理而成。

美团外卖后端架构迭代各阶段

美团外卖发展到今天差不多有 4 年多的时间，按照外卖业务发展的几个特征，可以相应地把外卖技术分成三个主要阶段：

第一阶段：业务初探期

大约截止到 2015 年初，持续差不多一年左右的时间。这个阶段的主要特征就是美团对于外卖的业务还处于市场摸索期，研发人员相对也比较少，差不多 10 来个同学，产品上需要快速迭代、试错。

所以这个阶段的系统架构比较简单，就是典型的单系统 Web 应用服务，主要是需要满足产品需求上的快速上线，验证业务模型的市场可行性。

第二阶段：业务爆发期

外卖业务在 2015 年初开始了爆发式增长。基于当前外卖的业务特性，90% 以上的交易都是

在午高峰和晚高峰这个期间完成的，对业务系统来说高峰期负载重，压力大。这个阶段，我们主要是从最早期的基于单系统的 Web 应用架构，向分布式服务架构的迁移改造。期间主要优化工作如下：

一、做架构的拆分，应对高并发、保证高性能

对系统的拆分，主要体现在系统服务层、以及数据存储层上。

通过对线上业务流程的分解，将外卖系统分成数据浏览体系、用户订单交易体系、商户接单配送体系、用户信息 UGC 服务等，同时也针对大的业务服务体系内的流量分布、以及功能差异性，再做进一步的拆解。比如浏览体系中会有门店服务、商品服务、搜索推荐服务等等。

针对并发的读写数据压力，我们也针对性地搭建了相应的分布式缓存服务、针对特定数据库表，例如订单表，也进行了基于订单 ID、门店 ID、用户 ID 等多个维度的拆库、拆表操作。

二、构建系统化运维体系，保障服务高可用

分布式系统拆分，提升了外卖服务的并发处理能力，同时也给线上运维带来了负担。需要对线上近百个服务应用、几千台机器资源进行运维管理，才能保证整个业务链路服务的稳定体验。这个期间，我们在业务系统的可运维方面，做了大量的工作。例如：

- 通过完整业务链路的梳理，形成关联各服务接口的 SLA，保障核心链路的稳定 ；
- 通过定期的全链路压测，验证各子系统的处理极限，合理规划系统容量；
- 通过链路故障模拟演练，验证各服务系统在功能层面、系统层面的故障降级处理能力。

第三阶段：业务扩展期

这个期间是伴随着业务爆发期逐步开始的。外卖的业务中，除了传统的餐饮品类外，还逐步提供了鲜花、生鲜、果蔬、商超、以及跑腿服务等业务的扩展。7 月 6-9 日，我除了在 ArchSummit 深圳站 上担任《大型分布式系统架构》出品人，也会现场分享美团外卖这个阶段在服务端的架构迭代，包括数据处理上的技巧经验，这里简单总结下：

业务扩展期阶段外卖业务和产品，我们期望以最小的代价，支持不同品类的差异性服务，能够在当前的系统中快速上线。对技术来说，就需要将系统由业务的功能化向业务的平台化方向发展。

我们通过对通用电商逻辑的抽取，形成公共服务平台，将差异性的品类业务特性，通过系统插件或者流程配置的方式，集成到公共服务平台上，以达到整个链路的整合复用。例如：

将订单交易流程通过类似工作流引擎的方式，针对不同的品类，配置不同的流程模板，使其能够灵活的运行在整个外卖的核心交易体系之中；

商家促销活动平台，支持从门店、商品、区域等等不同品类进行各种活动配置，满足不同品类的营销需求。

美团外卖后端新项目

当前美团外卖业务依然处于持续增长期，年前我们的日订单刚突破了 1800 多万。伴随着业务的发展，基于业务背后的技术系统也日趋复杂。无论是对于机器资源规模，还是系统服务之间的调度关联，都对线上服务的运维带来很大的挑战。

为保证外卖线上服务规模的可扩展和可运维，近期我们正在开展两大项目：一个是外卖的 Set 化部署项目、另一个是智能化业务运维系统建设。

外卖 Set 化部署项目

这个项目的目标是为了解决外卖当前大数据、高并发场景下的机房集群资源规模限制问题。我们希望将用户与商家的大规模请求流量，按照特定的规则，拆分到不同 Set 内，同时在 Set 内做到整个用户交易流程的闭环。这样我们并可以通过可控容量大小的 Set 划分，达到分散系统压力，快速支持扩容的能力。

Set 化不是一项新技术，当前业界也有部分公司已经在线上实施了，我们计划 2018 年下半年进入部署状态。这块涉及到的基础服务改造，应用服务改造非常多，我想等我们上线后，可以总结下经验，再来给大家详细分享下。

外卖智能化业务运维建设项目

在美团外卖的稳定性建设过程中，我们希望通过一套智能化运维系统，帮助快速、准确地识别各链路子系统服务异常，发现问题根因，并自动执行对应的异常解决预案，从而避免或减少线上事故影响。

当前业界关于 AIOps 这块的探索也有很多，我们也在同步的跟进摸索中，目前主要还处于基础性建设期间，包括核心链路的故障演练系统、全链路压测系统、告警分析诊断系统，服务自保护系统等等。

高并发应对策略

这里简单介绍下我们的高峰低谷策略、动态调度设计、目前的瓶颈，以及我们的优先级策略。

高峰低谷策略

针对当前外卖的周期性峰值波动特性，我们的主要采取如下几个策略：

系统基础运行能力的适度冗余

外卖系统在系统架构的设计上，无论是在服务层面系统部署，还是数据层面的存储资源都会做适当的冗余。比如我们大部分的核心系统服务，会保持线上高峰时 1.5 倍左右的冗余，以保证高峰期间的业务异常流量情况下的系统可用。

系统资源的动态调配

美团外卖的服务系统都是部署在美团私有云上的，最近一年来，我们也在很多的服务上使用了美团内部的弹性计算容器，以应对外卖系统的突增压力时，可以做到近实时的系统快速扩容。同时在低峰期也可以释放资源，提升机器资源利用率。

系统在极限压力下的自我保护

除了从基本的资源部署上来保证以外，在外卖的系统设计中，也会采取大量的自保护策略，其中包括各系统服务的限流、降级、熔断等自我保护策略，以保证系统在异常流量压力下核心链路的可用性。

动态调度设计

外卖业务链路较长，其中关联角色有三方：用户、商家、骑手，简单的业务流程是：用户下单 -> 商家接单 / 发配送 -> 骑手接单配送。

外卖服务是一个强履约服务，需要尽最大可能，保障用户下单之后，以最快的时间内送达商品，保障用户的体验。外卖当前主要基于餐饮的商品，属于非标品服务，同时也是基于人员配送

的即时物流服务，所以一定时间内的服务规模是有限的。一旦商家商品供给能力不足、或者配送资源不足的时候，都需要通过系统来调节。

例如，当运力充足但商家订单过多，出餐瓶颈出现的时候，为保证用户的体验，我们会从产品上提示用户商家忙，继续下单需要能够让用户有延迟送达的预期；同时针对订单较多，出餐较慢的商家，也会在门店列表排序上给予一定的关联因子降级。

对于运力不足的情况下，也会通过适当的实时策略调整，来保证现有运力的供给平衡，比如在高峰期之外降低配送费，高峰期间提升配送费来调节需求的平衡分布。在极端天气，爆单的情况下，可以通过动态调整特定区域商家的配送范围，来保证用户体验。

总之在外卖的整个交易链路中，用户、商家、配送等系统之间的实时数据是互通的，各业务系统通过关注不同的实时数据变化，采用不同的策略应对，从而调节整个业务的运作模式，保证外卖链路三方的平衡。

当前瓶颈

随着业务规模的进一步扩展，当前外卖分布式架构体系中，系统各层的水平扩展性是我们可能会面临的瓶颈。 具体来说，我们面临的问题包括如下两点：

服务层单个集群大小受限

服务层虽然是无状态服务，理论上可以无限增加，但是单个集群规模越大，运维成本就会越高，比如基础服务治理对于服务状态的维护、更新、同步等。

数据存储单个集群大小受限

单个集群的从库个数是受限的，从库越多，主库消耗越大，导致从库个数是受限的。

所以总的来说，系统集群规模是存在上限的，导致实时线上服务的系统处理能力受限。近期我们启动了外卖服务 Set 化升级，结合外卖的强地域特征，将流量按照地域拆分开，每部分流量由单独集群提供服务，从将整个大的分布式集群拆分为多个小集群，从而可以通过从业务逻辑层的设置，来达到控制集群规模大小的目的。

在当前我们的非 Set 化架构体系中，受限于上面提到的两点，大约可以满足两倍于当前业务的数据量，可以支撑近一年左右的业务增长。随着我们的 Set 化项目 2018 年完成部署，理论上通过增加新的 Set 集群，进一步扩展系统能力，可很好地支持未来 3 年到 5 年的业务增长。

优先级策略

作为一个线上即时交易服务系统，在线上资源不足，或者故障发生时，我们首先会优先保护和订单交易相关的核心业务链路。这条链路从用户能感知到的服务就是：商家列表的浏览、商家内的商品展示、用户提单支付、商品骑手配送。

这就需要我们梳理最小化的核心服务链路，除此之外的其他链路分支所关联的所有服务，都可以在故障时刻被降级。比如针对商家列表的排序、推荐、广告服务、评论服务、搜索服务等等。

而实现最小化交易链路保证的前提是：各系统具备能够降级非核心服务接口依赖的能力。外卖服务经过这几年的不断业务梳理与积累，逐步开发实现了各业务链路的限流开关、功能服务开关、依赖服务降级开关百余个。这些开关通过我们的事故紧急预案，被分类组织到我们的业务运维系统中，一旦对应的事故发生，我们可以快速的通过系统，来完成批量的服务降级处理，从而

避免事故的发生、或减少事故造成的损失。

外卖高峰期的入口请求 QPS 达到 20W 左右，这其中包含了部分恶意攻击或抓取流量。针对恶意爬虫抓取这块，我们会与公司内部的反爬团队一起做这块的防范工作。而针对突发的攻击性流量，除了公司上层流量入口层的控制以外，外卖自身的服务也会根据用户或地域维度做一定的限流策略，避免恶意攻击造成对正常流量的影响。

架构优化

扩容还是性能优化

为达到目标系统的容量，一般有扩容和性能优化两种手段，这两种手段会根据服务当前的具体情况配合使用。大的原则是：长期保持系统的优化、紧急状况扩容或按照长期业务趋势规划扩容。具体我们在扩容和性能优化上做的事情如下：

关于扩容能力

外卖的业务技术系统，构建在美团内部分布式基础中间件之上（例如分布式缓存、消息队列、服务治理中间件等），这些通用的分布式组件，为上层应用的水平扩容，提供了基础能力保证。

对外卖业务系统，我们也进行了对应的系统拆分（例如交易服务、营销服务、评论服务、商品门店服务等），拆分中保证各服务均采用无状态设计，可以快速支持水平扩容。另外类似营销这样存在突发流量的服务，也接入到美团弹性计算容器中，可以支持根据流量变化，来进行自动化的扩缩容。

关于性能优化

针对外卖高并发场景，合并分散的调用接口，降低各服务系统间访问频次，防止一次调用链中对依赖服务的请求次数放大；提升数据获取性能，优化存储结构设计（分库、分表、建索引），采用适当的存储方式（DB、分布式缓存、文件索引、本地内存等）；将部分请求处理异步化，非实时处理离线化等。

对于如何避免过度依赖扩容，大的方面还是从系统的性能优化入手，当机器资源达到服务瓶颈的时候，需要深入分析具体瓶颈点在哪里，是存储、CPU、带宽还是其他的原因。然后针对具体的点，再考虑是否必须扩容。在外卖定期进行的全链路压测场景中，我们会根据压测的数据，结合各服务的 SLA 要求，来评估各服务的处理能力，再结合未来业务的增长趋势，做较长期的容量扩容规划。

系统架构拆分原则

架构的拆分需要匹配业务规模的发展程度，并具备一定的超前性。过早的拆分会造成不必要的资源浪费，而过晚的拆分，则又会累积现有系统的风险，增加拆分业务的梳理难度。

在外卖的技术发展过程中，由于外卖早期的业务爆发太快，相对技术资源储备不足，拆分的时机是落后于业务发展的，所以早期也出现过很多的线上事故。后来我们通过快速的系统架构拆分与调整，逐步完善起外卖的后端体系架构，减少了线上事故的发生。总结起来，我们针对系统架构的拆分遵循以下三个原则：

流量拆分原则

外卖服务流量按类型，我们可以简单的分为 To C 流量和 To B 流量，也就是面向外卖用户和面向商家、运营的流量。针对不同流量的服务系统会拆分，典型的比如：针对商家或 BD 运营的门店管理、商品管理、营销活动配置管理等，就

会和线上的门店、商品、营销等服务分开。除了服务系统上，存储 DB 上也会拆分，中间我们会采用一些实时的数据同步机制，保证数据的一致性。

另外一个维度的流量拆分是按照同步与异步的请求分类。 就是从用户流量的发起方来看，在请求的处理链条中，那些可以异步处理的逻辑服务，会从实时链路处理系统中分拆。比如我们的用户提单系统、与商家接单处理系统、发配送系统等之间都是异步拆分的。

功能闭合原则

在软件设计领域有一个著名的康威（Conway）定律，其中提到组织架构与软件架构相互影响的辩证关系。我们在按照业务功能垂直拆分的过程中，也会有同样的考虑，尽量在一个组织架构中，将功能性业务系统闭合，减少跨团队之间的沟通成本，提升产品开发迭代效率。比如我们的的搜索服务、推荐服务、订单交易服务、UGC 服务、营销服务、门店商品服务等等。

平台服务原则

对不同业务系统公用逻辑或服务的抽取，形成基础性中间件平台，服务于多个业务系统，从而降低维护与开发的成本。

比如我们的数据算法工程平台，我们发现在很多的业务系统中，都会有些算法策略迭代，它们都需要线上特征数据、策略 A/B 实验、效果分析、工程系统支撑等。所以我们将此抽取成一个公共的工程平台，算法只需要提供策略逻辑代码，上传到平台上，平台提供针对业务系统对接的标准化服务接口、特征的统一管理服务、算法的版本管理、A/B 实验、以及效果分析等等。这样的业务中间平台，可以很好的提升了各业务系统的开发效率。

如何梳理核心链路思路

作为线上即时 O2O 电商业务，外卖业务核心是为用户提供订单交易履约，因此我们的核心链路是根据订单交易的业务场景来梳理的。主要包括门店浏览、商品选购、交易支付、订单配送环节，这四大业务环节强依赖的服务接口组合起来的链路，就是核心链路。而不在此关键链路中的服务或非强依赖服务链路，我们将此归集为非核心链路。

对于如何合理地对业务链路进行分级，需要看业务规模的发展阶段，早期可以只划分核心链路和非核心链路两级即可，后期可以再针对非核心链路，进一步向下划分，用以确定更细粒度的降级容错控制。

比如我们在外卖入口 API 层，调度外卖红包服务的链路中，会按照红包的使用、红包的浏览、红包的获取等进行进一步的分级，当红包的系统压力过大时，按照核心与非核心链路的分级方式，我们就只能直接降级红包系统服务了，这样对整个下单交易不会有太大问题，但不能使用红包，会影响用户的下单体验。所以按照进一步的链路梳理降级，我们会优先保证红包的使用接口服务，而降级其他的红包调用链路。

当然，如果针对非核心链路分级太细，也会同时带来服务维护成本的提升，这块需要针对业务做相应的平衡。

人工智能探索

美团外卖自成立以来，就开始做一些 AI 方面的研究和尝试。比如：

- OCR方面，针对商家的证件照识别、营业执照识别等，这块的识别召回达到了85%以

上，准确率我们达到 99% 以上；

- 机器人方面，我们在和业界的一些智能音箱和家庭服务类机器人公司对接外卖服务，通过语音对话交流的方式，完成整个外卖订单的交易；
- 用户方面，我们不久前上线了外卖智能助手服务，目前大家可以在微信钱包的美团外卖入口看到，也有部分 App 灰度用户可以看到。通过与智能助手小袋对话，她可以帮你快速的完成商家、商品的自动选择和推荐，体验还不错，大家可以去试试；
- 骑手配送方面，美团外卖也在积极开发配送机器人，前不久刚刚发布了一个测试版本，大家可以在网络上搜到。

总之，正如王兴之前在“中国发展高层论坛 2018 年会”中说的，美团将会在科技创新、人工智能方面积极发展，普惠所有大众。外卖作为一个重要的连接线上线下的服务体，也会不断在 AI 方面做更多的尝试。

外卖的业务目标是能够在几年内达到每天上亿次的用户服务，那么针对外卖后端架构的发展，我想后续会更多的关注在多品类支撑的平台化改造方面，以及在越来越复杂的系统关系调度与智能化运维方向。

方建平，美团点评技术总监，ArchSummit 出品人，致力于大型互联网分布式后台技术架构建设相关工作近十年。于 2014 年加入美团，目前主要负责美团外卖的后端技术。在美团外卖时期，从无到有，设计并主导构建了高效的分布式外卖后端技术架构体系，支撑了外卖业务背后亿级用户日访问处理请求。

个人对大型分布式服务建设过程中，不同阶段所需要解决的关键问题，以及对应的解决方案，有较多的认知与实践。在加入美团之前，任职于百度，主要参与负责百度开放平台、移动云平台等后端服务技术架构与团队管理工作。

专访平安科技CTO方国伟：云计算十年，最需要突破的不仅是技术

作者 蔡芳芳

中国平安保险（集团）股份有限公司于1988 年诞生于深圳蛇口，是中国第一家股份制保险企业，至今已经发展成为金融保险、银行、投资等金融业务为一体的整合、紧密、多元的综合金融服务集团。

平安科技成立于 2008 年，负责平安集团 IT 基础设施建设、系统运维、业务应用开发、管理等工作，是国内最早成立的金融科技公司之一。自 2013 年平安集团推进互联网战略，平安科技就加大了在移动互联网、云计算、大数据和人工智能等领域的研发和应用。平安云是平安科技打造的面向金融、医疗和智慧城市三大垂直领域的云平台，是平安产品和服务输出的总平台。

我们采访了平安科技 CTO 兼总架构师、ArchSummit 联席主席及 Keynote 演讲嘉宾方国伟，共同探讨了云计算产业的发展和平安云的演进历程，结合平安科技的发展经验分析 AI 产业在过去一年的发展情况。

从私有云到公有云

您先后在微软、AWS、平安这几个科技巨头都工作过一段时间，能否请您简单介绍一下自己的这几段职业经历？

方国伟：2007 年我初到微软的时候主要负责大企业的创新技术支持。2008 年微软对外推出了 Window Azure 这个服务，当时还只是技术预览版（Technology Preview），从那个时候开始我就正式投入云计算相关的工作。我在微软前后

工作了 6 年多，其中有一段时间在总部雷德蒙参与了 Windows Azure 和 Windows Server 的相关工作。我负责设计了微软中国第一个私有云解决方案，构建了第一个云计算演示中心，并实现了第一个私有云的企业客户案例。我还组织牵头写了微软中国第一个云计算白皮书和第一本关于微软云计算的图书。

2013 年亚马逊 AWS 进入中国，虽然当时的 AWS 在中国其实有点像一个创业公司，只有几个员工，但我看好亚马逊的云技术，因此正式加入成为 AWS 中国的第一个技术人员。我在亚马逊 AWS 主要负责技术市场相关的工作，同时也负责一些重要客户的服务咨询工作。14 年平安科技开始大规模投入做云计算，对于我来说，能自己和团队一起去设计一个完整的、全新的云平台产品是一个更有挑战也更有成就感的事，所以我和家人来到深圳，加入平安。

从您 2008 年开始正式进入云计算领域已经十年了，在不同的职业阶段，就职于不同的公司，您对于云平台的认知是怎么迭代和变化的？

方国伟：这十年我做过公有云，也做过私有云，微软的 Window Azure 是公有云，外企的云落地到中国是很有挑战的，我当时提出的微软云战略是，一方面通过 Window Azure 帮助微软建立云计算的领导力，另一方面是通过私有云解决方案帮助微软获得云方面的收入。从供应商的角度，私有云的解决方案就是如何以云的概念去销售 IT 产品和服务。我在微软也曾帮助多个中国客户构建了私有云。我在亚马逊的时候做的是公有云，到了平安则是逐步把私有云构建成了公有云。

在云计算刚刚开始发展的时候，大家认识也不统一。我记得曾经有个笑话讲在一个云计算大会上，主持人问了 10 个人什么是云计算，结果得到了 12 个答案。我做了大量的学习和阅读，印象比较深的有两篇论文，一个是 09 年伯克利的一篇论文（Above the Clouds: A Berkeley view of Cloud Computing），还有一篇是美国 NIST 的论文（The NIST Definition of Cloud Computing），这个组织经常给行业做标准定义的。云计算的标准，包括 IaaS、SaaS、PaaS 三种服务模式，公有云、私有云、混合云部署模式等都是 NIST 提出来的。我通过前几年的学习和实践，我逐渐完善了对云的认知。后面几年更多的是将所学所想在实际场景应用起来，我自己对于云的认知本身没有太大变化，但可以看到现在市场对于云的接受程度越来越快。

你觉得在这十年里，云计算的应用情况可以分成哪几个阶段？

方国伟：我认为云的发展或者说企业 IT 的发展可以分为几个阶段：第一个阶段是传统 IT 方式，这个阶段不是很动态，采用技术也比较固定，都是传统 IT 厂商提供的技术。第二阶段是采用虚拟化阶段，2000 年后虚拟化技术慢慢成熟，越来越多的企业开始使用虚拟化技术，但也面临着很多挑战，比如服务器虚拟之后数量变多、灵活性增加的同时管理复杂度也增加了，因此需要更好的方式来使用这些新技术。然后过渡到了私有云，私有云是在已有的 IT 虚拟化技术上加上动态管理用以提高效率的方式。我认为云计算发展的最终的方向还是公有云，但往公有云发展可能中间有一个阶段，有些企业既用公有云也用私有云，属于混合阶段。随着云技术的不断发展，我认为公有云的比例肯定会越来越高。

我之所以一直看好公有云（含行业专有云）的发展，是因为关于云计算的三个基本因素一直

没有变化：

- 经济学上的规模效应；
- 自动化提升效率；
- 专业分工，专业的人做专业的事。

只要这三个基本因素没有变化，公有云、专有云的发展方向就不会变。

平安云的四年升级之路

刚才您介绍平安是 13 年底开始构建云计算平台，平安云的架构设计经历了哪几个阶段的演进？能否介绍几个关键节点？

方国伟：在云平台搭建之初，确定一个合理的架构是非常重要的，因为架构确定之后再做调整是非常困难的。在平安云的架构上，我们当时做了两个非常明智的决定。第一个是决定用公有云的服务设计和架构方式来做私有云，比如我们一开始就做了多租户，平安内部有很多的专业公司，专业公司下面还有不同部门和项目组，他们在我们的云上就是一个个租户。还有我们使用了 VPC（虚拟私有云）的方式，这种方式在于可以把租户和租户之间通过网络很好地隔离开，这个决策使得我们后续在向公有云方向发展的时候，技术架构上的调整比较简单，没有太大的阻碍。

第二个决定是平安云的整体架构设计由我们团队自己控制，云平台中一个的功能模块使用什么技术可以由团队自己来决定，有些是自主研发，有些是开源技术，有些是选用商业产品。这个决策我们认为非常重要，有些公司在搭建私有云的过程中，采用与某个供应商整体合作的方式，这样做不仅内部团队对技术把控不住，灵活性也欠缺，因为供应商产品往往是一个整体，定制化困难。如果整个架构是由我们自己设计，每个模块采用的具体技术方案由我们自己选择，这样得出的整体方案灵活性很强，可以做到整体自主可控。云的发展越来越快，规模越来越大，业务越来越多，打造一个自身能力很强的技术团队，对以后的高速发展也非常有利。

您刚才谈到的两个明智决定保证了平安云不管是随着公司战略发展而调整，还是跟随整个云平台技术而不断发展，平安云都能游刃有余。那么，能否介绍下，在平安云这四年多的发展历程里，有哪些关键的变化？

方国伟：这是一个高速发展的时代，每个公司的业务发展规划都是动态调整的。2014 年初到平安时，公司高层的想法是搭建平安集团的私有云。到 2015 年底，这个任务基本完成，平安私有云的规模、架构都是比较完备的，云平台上的应用也越来越多。2016 年初我们就在构想，既然平安云在平安内部推广和应用的相当成功，为什么我们不能将这个平台介绍给更多的客户，对外部客户也开放呢？因此 2016 年公司提出平安云要从对内服务转变为对外服务，我们专注的是金融行业，这是我们的巨大优势，所以我们要做最专业的金融公有云。这个是第一个比较大的业务发展规划的调整。

2016 年我们推出的金融云成功的为很多银行业务提供了技术支撑，2017 年年中的时候我们对外发布了平安公有云。17 年底平安集团的又进行了新的业务战略调整，进一步加大科技的投入。另外不但做金融，还在医疗和智慧城市两个方向发力，所以 2018 年云平台的重点就是为金融、医疗、智慧城市这三大行业做支撑，这是第二个比较大的发展方向的变化。由于这三个行业都属于强监管行业，因此我们面临了很大的挑战。我们在云平台构建里面做了很多隔离技术，使得整

个云平台可以支撑不同的行业。为了应对政府对数据安全性的高要求，我们使用了不同的管理方式，有托管也有专区，以此来满足不同行业对数据安全性上的各种特殊要求。

平安云目前的业务主要分为哪几部分？

方国伟：从产品角度划分，有计算、存储、网络、数据库等。这个角度上平安云跟一般的云平台很像。但平安云跟其他云平台最大的区别是我们有很强的业务属性，如果从业务角度来划分，主要分为三大类，金融、医疗和智慧城市。目前金融行业的业务量最大，例如我们跟平安壹帐通公司合作，在云平台上提供很多针对银行的应用和服务。

您谈到目前平安云上金融行业的业务量是最大的，这能否代表金融行业对云计算的需求是最大的呢？

方国伟：这应该不是普遍的规律。平安云是行业云，我们专门针对上述行业来提供服务，因此这些行业的业务量相对较大。目前市场上其他的云平台各有侧重，有的云平台主要是做游戏，有的是做互联网创业公司，每个云平台都有自己的个性特点，专注不同的市场。所以我们在打造平安云时也重点打造平安云的差异化和竞争优势，和其他的云平台区别开来并为我们的目标客户提供更好的服务。

现在，越来越多的客户需要获得人工智能的支持，这个能力可以通过云端获得——无论是公有云、私有云还是混合云，也可以通过终端获得。平安云现在主要是通过哪种方式提供这种能力的接入接出？能否介绍一下现在通过平安云平台使用这些人工智能能力的案例？

方国伟：平安云的人工智能能力分为三种方式来输出，一是直接提供了人工智能的服务能力，例如平安云已经上线的人脸识别 API，用户可以直接调用 API 来使用这个服务。第二，我们提供了机器学习的框架服务，比如我们支持 TensorFlow，在这个服务的基础上，用户需要自己设计算法，然后将算法放在机器学习服务上面运行。第三种，更加底层一点，我们在云平台上面也会提供 GPU 计算服务，客户可以使用这些计算力，构建自己的机器学习框架并进行训练。云平台上一个典型的人工智能案例是车险定损例子，我们现在可以通过手机拍照并上传受损的车辆照片，后台系统就可以自动辨别车型、受损部位、损伤程度等并自动计算需要赔偿的金额。

您感觉市场对 AI 能力的需求现在处于一个怎样的量级？从平安云的角度，涉及 AI 的业务在你们平台上有怎样的增长趋势？

方国伟：整个社会对 AI 的需求从 2017 年开始增长的非常快，可以说是爆炸式增长，我们的平台也反映了这一变化。目前做 AI 的重点大多是通过机器学习和深度学习，这两方面对数据计算能力要求很高，无论是什么样的模型，都需要大量的计算能力，所以在云平台上我们看到对 GPU 的需求量非常非常大。现在最大的挑战在于 GPU 的供不应求。目前在市场上采购 GPU 服务器，周期还是比较长的，我们经常遇到的一个挑战就是我们预测的需求增长赶不上用户实际的需求增长。这从另一方面也反映出 AI 的需求量确实非常大，一个模型就需要很多资源去训练。我们认为 AI 正在成为云平台上的一个 Killer App（杀手级应用场景）。

云平台如何做技术选型？

刚才我们提到金融是非常重要的云计算应用

的行业，相比其他传统行业，你们为金融行业打造的云服务有哪些不同的业务需求和技术挑战？

方国伟：金融行业最重要的是安全，平安金融云的安全性首先得益于平安科技的团队，大部分团队成员的背景一直从事金融 IT，所以对金融行业对于安全、运维的相关要求都非常了解，由这个团队构建的金融云平台，在满足金融监管要求和提供高安全性服务方面是有重要优势的。第二，银监会对银行应用云服务是有特殊要求的，如果某个银行关键应用使用了云平台服务，那么云平台的隔壁租户也必须是银行，也就是要求云平台必须做一个银行专区。因此我们做的金融云会通过银行专区的方式提供给银行客户，也就是底层的计算资源、存储资源等专门给这些专业用户用，通过物理上的隔离来满足行业监管的要求。最后，我们在架构和资源上都充分考虑的金融行业的高可靠性和冗余性要求。

请问对于医疗行业打造的云服务有哪些不同的业务需求和技术挑战呢？

方国伟：对于医疗行业也是和金融行业一样，首先要满足行业监管的要求，然后是考虑如何让我们的服务更符合医疗行业对底层技术的要求。我们在平台层 PaaS 层，应用层 SaaS 层，都在努力提供更多的能实际解决医疗行业业务的服务，使我们的服务更符合行业云的特点。

平安云采用的是哪种开源云平台？为什么做出了这样的选择？

方国伟：很多人认为开发云平台，需要选择某一个框架，比如 CloudStack、OpenStack 等等。但是经过几年的实际耕耘之后，我认为选哪一个框架并不关键，云的关键在于底层的网络、存储，包括一些自动化的流程设计。框架上我们最开始用的是 CloudStack，后面也融入了一些 OpenStack 的模块，所以我们的做法相当于把 CloudStack 和 OpenStack 的相应模块整合在一起，加上我们四年多的大量定制构建起我们自己的 PAStack（也叫做平安 Stack），形成了平安自己的一套框架，使得我们平台的定制能力和适应能力更强。但是平安云的真正关键不在这个框架，而是底下的网络、存储等核心服务，以及如何将这些部分做得更加安全可靠，扩展性更好，跟 PAStack 能有机的整合在一起，提供卓越的云平台服务。

您刚才提到的网络、存储和自动化流程上进行过哪些选型工作？

方国伟：我们尝试过很多不同的技术，以存储为例，云存储服务可以分为三大类，对象存储、块存储和文件系统。我们的对象存储是基于 Ceph 做的，块存储采用了混合的方式，使用了 Ceph 还有传统的 SAN，甚至也有用到本地盘。我们通过不断尝试来研究哪一种存储方式能更好地适应云平台的需求。我们现在的云平台有 11 个数据中心，每一个数据中心的构建时间都不同，比如说最开始构建的数据中心，当时没有 Ceph，就只能是用传统的 SAN 来做块存储。当技术慢慢越来越可靠，团队把握能力越来越强以后，新的数据中心构建就会选择 Ceph。技术始终是在不断发展的，我们主要关注什么样的技术更能解决当前场景的问题。平安云从 14 年底 15 年初开始做容器的尝试，我们尝试过很多不同的技术，目前最新的容器技术发展方式就是按照 Kubernetes 来构建的，虽然其他的容器技术也还在用，但重点是放在 Kubernetes 上。

可否介绍下平安科技的整体技术研发团队和容器研发团队的规模？

方国伟：由于平安集团在做科技方向的业务

转型，所以今年的策略跟之前不太一样，今年我们平安科技的转型目标就是做云公司。所以平安云不仅仅包括刚才讨论到的计算、存储、网络、容器，我们也包括很多平台层和业务层的服务，都会整合到云平台里面来。我们对外的服务都会通过云平台统一展示和服务，所以从这个角度来讲的话，大部分平安科技的研发人员都会参与到平安云的建设工作中。另外，平安云上的许多SaaS 服务也是平安云的一部分，许多针对各个行业的应用服务都是由平安不同专业公司参与共建的，所以整体研发团队规模是比较大的。

您一开始也是架构师，慢慢走到现在，在整个团队中，架构师所承担的是什么样的角色？您所理解的架构师职责包括哪些？

方国伟：架构师还是挺关键的，因为架构师是从需求到最后实际做产品研发落地的重要桥梁，他需要对很多需求有比较深刻的理解。这个需求包括两方面，一个是功能性需求，到底这个产品做什么的；另一个是非功能性需求，比如说安全性、可靠性、性能等等，跟直接功能没关系，但是会间接地影响功能本身的使用好坏，这些需求架构师都要有很深的理解。因此对架构师的职责要求也非常高，技术要过硬，一方面要有自己的技术特长，另一方面是技术面要广，比如你要设计云相关的某个架构，关联的系统都要有所了解，否则设计出来的的系统是孤立的。如果说这个架构师是做软件相关的架构，我们希望这个架构师能够开发核心模块。如果是系统架构师，那他需要对相关系统软件有一线的经验。不能光说不练，还是要有很强动手能力才行。

如何成为一名优秀的架构师，您是否能跟我们分享一些自己的经验？

方国伟：仅供参考，我认为一个架构师有几点比较重要，一个是一定要有自己的技术特点，某方面的技术必须了解得很深，架构师不能光是了解面，好像什么都懂一点，没有自己的专长的话，很难建立起自己的技术口碑，没有技术口碑就没有技术上的公信力。第二，架构师眼界要广，知识面要广，敏感度要比较强，你要了解到哪些技术是有发展潜力的，架构师很多时候会涉及到技术的选型，需要帮团队做技术决策，敏感度不行的话就可能选错。第三点其实跟很多技术工作都有关联，只是对架构师来说可能更重要一点，就是学习的能力，要能够不停地学习新的东西，快速了解新的东西，一旦有很新的技术出来架构师要能快速地了解并上手使用。

您刚才说到的敏感度，感觉比较抽象，怎么培养敏感度呢？

方国伟：敏感度有两种方式来培养，一个是你要关注业界发展的情况，要知道现在业界大概有哪些新的技术出来，这些新的技术跟老技术有什么转变关系和替代关系。第二是架构师要动手，我一直鼓励团队的人要把你的手弄脏（Keep your hands dirty）。如果你做系统设计，就要去装一些系统，配置一些系统，如果是应用代码设计，自己要写代码，自己动手做了之后，才能增强技术敏感度。举个例子，16 年的时候，我负责云平台团队，当时团队想做一些自动化运维，我们取了个名字叫 AlphaOps，因为 2016 年 AlphaGo 下围棋赢了，我们觉得在运维上面也可以应用人工智能。为了了解如何在运维上应用人工智能，我也去学习了 Python 语言，学习了之后就更能了解为什么 Python 对运维很重要。因为 Python 语法简洁，还有不同的运行模式，支持交互模式。Python 的交互运行在运维里非常有效，而且 Python 在机器学习里面也非常流行。

我希望团队的人都是这样的，自己动手去做，才能对技术更了解，敏感度就会更好。

云计算回顾和展望

有人认为 2017 年是 Kubernetes 的胜利之年，您怎么看？

方国伟：Kubernetes 在 2017 年确实发展得非常好，Kubernetes 其实从 2014、2015 年开始发展势头慢慢起来，为什么能发展得这么火？因为 Kubernetes 背后是容器，背后最重要的推手是谷歌。谷歌为什么要花这么大精力去推一个开源的产品？我的看法是，主要还是因为谷歌在云计算领域是后来者，他们在想办法改变游戏规则，他们想改变原来基于虚拟机的运作方式，因为在原来的游戏规则里，谷歌是不可能赶上其他厂商了，比如 AWS。但实际对更多的云从业者来讲，这是一个很好的机会，既然谷歌可以利用这个趋势，其他云厂商也可以利用这个趋势，因为容器的好处是让云平台上面的应用和云平台本身的耦合度降低，大家不会绑定在某一个特定的云平台上，所以迁移能力会比以前更强一些。2017 年发展很快，我相信一方面是谷歌背后在推，另外是更多的从业者、云服务商觉得这是很好的机会，可以改变游戏规则，能够让自己有更大发展空间。

您觉得过去这一年，云计算领域最重要的新事物是什么？

方国伟：云计算行业有两个事情是我们重点关注的，一个是应用方面，两大类应用是比较有潜力，一个是人工智能，当前人工智能大多基于深度学习或者机器学习，需要大量数据，这些数据放在云平台上比较合适，同时对计算的需求量也很大，既有存储需求又有计算的需求，因此人工智能对云平台会有比较大的驱动作用。另外一个是正在兴起来的 IoT，也就是物联网，以后的智能应用会无处不在，物联网本身会产生大量的数据，有很多边缘计算方面的需求，这跟云平台的关系也非常密切，因此从需求角度来看，我们认为云平台在 IoT 方向上的发展应该也是很明显的。

另外从计算模式上来讲，现在云平台有一个新的计算模式，平安云也在做，就是 Serverless 的计算模式。我们内部也有一个项目也在做 Function as a Serverless（FaaS），目的是希望降低用户对资源、环境管理的需求，只要把代码一提交，云平台可以帮他自动运行，而且真的能做到用多少付多少钱，如果没有人调用这个服务，这个服务本身是没运行的，也就没有计费。这个是云在近两年慢慢发展

起来的一种计算模式。

您觉得云计算领域现在最有待突破的地方在哪里？

方国伟：云计算最有待突破的不是技术，而是另外两个。一个是心理，我并不认为公有云的安全性不如私有云，但这是很多人认为的。这是一个误解，实际上公有云的安全性并不会比私有云差，我认为没有本质区别，所以这是云计算行业需要更多用户突破的一个心理障碍。第二要突破的是利益问题，很多企业都有自己的数据中心，有自己的团队去做基础设施运维，如果采用公有云的方式，或者行业云（行业云也是公有云的一种，是针对某个业的公有云），这个团队本身的利益会受到损失，可能这个团队就不需要了，或者团队工作量会减少，这是利益方面的分配问题。我觉得这才是云计算行业目前面临两个需要突破的问题。技术的演进本身我觉得是没问题的，我们并不期待一个特别大的突变来改变云计算领域，因为在过去的发展中，绝大部分应用已经可以在云平台上完成和实现，至少可以实现得跟本地一样好，所以关键不在技术层面，而是心理或习惯上，在利益上。我们需要意识到，整个行业在转变的时候，肯定有些人的工作方式会发生变化。

您认为下一阶段云计算的主流发展方向是什么？ IaaS、PaaS、SaaS 和平安提出的 CaaS 各自将如何发展？

方国伟：云计算发展最终是要解决业务问题，我们认为 IaaS 这一层慢慢会趋向于同质化。很多公司都在做云平台，云平台里面很大一块是做 IaaS，但最终 IaaS 这一层慢慢会同质化，差别会有，但是不会有本质区别，就像我们买服务器或买笔记本，最开始笔记本出来有许多不同的品牌，慢慢的最后剩下的品牌就很少了，因为大家觉得选哪个品牌都一样，服务器也是一样的。所以云平台的 IaaS 一层未来也不会有太多不同的厂商品牌，而对用户来说选哪个 IaaS 也都能满足它的需求。

平安云一方面是要继续夯实 IaaS 这一层的技术，使得我们的产品跟其他的产品在通用基础功能上一样，但是在针对行业上具备一些独有的特点。另外我们会在业务属性上加强突破，希望我们的云平台可以针对智慧城市、医疗和金融提供特殊服务，尤其是在整个 P 层和 S 层，后续有很多发挥空间。

对于平安云的下一步您有哪些规划？平安云未来有哪些改进计划？还将探索哪些新的技术或发展方向？

方国伟：从技术方面来讲，我们始终在跟踪云计算业界的发展，

希望在技术发展上保持跟业界同步。另一方面，从业务拓展方面，整个平安云的业务紧紧围绕平安集团战略来做的，平安云有自己的云战略，但从整个集团来讲我们是支撑集团的业务战略。例如，集团如果要拓展海外市场，平安云也会拓展海外市场，比如今年我们在香港已经建立两个数据中心，会提供平安云的服务。第三，我们会继续加强业务能力构建，例如我们今年构建了一个 API 市场，其中包括技术的 API 和业务能力的 API，业务 API 是围绕着刚才说的三大行业来构建的。

当前整个云计算市场厂商非常多，竞争激烈，您希望平安云未来在其中扮演什么样的角色？

方国伟：我希望平安云是与众不同的云平台，目前大部分云平台都是技术属性，没有业务属性，我希望平安云是最有自己的业务属性的，而且是比较强烈的业务属性，这也是我们构建行业云的一个很重要的特点，比如说云主机可能跟其他云平台类似，但是还有一些不一样的，带有很强业务属性的服务，这需要把平安整个集团各个业务公司的专业特点发挥出来。另外平安云会比较开放，比如我们平台上的 API 市场，对于平安内部提供的 API 和合作伙伴提供的 API 认为都是一等公民，而且技术的、业务的 API 都会放到我们的云平台上面。

方国伟（William Fang），平安科技 CTO 兼总架构师，ArchSummit 联席主席及 Keynote 演讲嘉宾。毕业于清华大学计算机系，获硕士学位。曾担任华为中央软件院架设部部长，亚马逊 AWS 中国的首席云技术专家，微软全球服务部门云计算“卓越中心”（COE）团队资深架构师。他曾主编《让云触手可及》和《详解微软 Windows Azure 云计算平台》，并在国内外的技术大会上多次发表过关于云计算的主题演讲。

专访死马：为什么说Egg.js是企业级Node框架

作者 徐川、薛梁

自从 Node.js 问世以来，阿里就是国内跟进 Node.js 最激进的公司之一。阿里不仅在公司内大范围使用 Node.js 用作服务端开发，2016 年还将他们使用 Node 做 Web 开发的经验沉淀下来，推出 Egg.js 开源项目，号称为企业级框架和应用而生。企业级开发对于框架有着诸多要求，那么 Egg.js 又是怎么满足这些要求的？

InfoQ：目前 Egg.js 在阿里内部使用的程度如何，有哪些使用场景？

死马：阿里和蚂蚁内部在开源的 Egg.js 基础上封装了一个内部的框架，不同的子公司在此内部框架的基础上二次封装符合自身业务的框架。内部已经有超过 20 个 Egg.js 的上层框架，服务于十多个子公司，包括蚂蚁、天猫、阿里游戏、河马、阿里云等。线上运行了数百个基于 Egg.js 的应用。

不同的业务会将 Egg.js 用在不同的场景，分类下来看主要是以下几类：

- BFF层：做业务的前后端之间的中间层，用于更清晰的划分前后端工作职责，提升研发效率。
- 全栈：许多创新性业务和内部系统会通过 Egg.js完成全栈应用研发，蚂蚁近期推出的语雀就是基于Egg.js全栈开发的。
- SSR：部分业务需要通过服务端渲染来提升用户体验，有Egg.js的上层框架专注于服务端渲染同构解决方案。

InfoQ：为什么说 Egg.js 是阿里和蚂蚁合作

维护的，双方是如何协作的？

死马：Egg.js 最开始脱胎于蚂蚁金服的内部 Node.js 框架 Chair，为了更好的服务于蚂蚁和阿里全集团的业务线，双方合作将 Chair 底层的核心抽离出来后开源，所以它是阿里和蚂蚁以虚拟工作组的方式合作推出的。

而到现在，Egg.js 已经是一个很成熟的开源项目了，不仅仅有来自阿里和蚂蚁内部的开发者，还有许多的外部开发者一起帮忙维护。可以看到 Egg.js 主仓库已经有超过 100 个贡献者了。整个的协作流程以异步交流为主，所有的大的新功能都会在 Github 上开 issue 进行讨论，而代码变更都会由多方 review 后再合并。

InfoQ：之前 Node.js 在做企业级 Web 开发时有哪些缺陷，Egg.js 做了哪些工作？

死马：不能说 node 在做企业级 Web 开发时有缺陷，而是没有规范。企业项目和个人项目有许多不同之处：参与的人员会很多，项目的维护者也经常变更有非常多内部系统需要对接，特别像阿里、蚂蚁这个规模的公司，内部都维护了许多自己的中间件服务和定制化的 RPC 通信框架对稳定性、性能、Bug 追踪等方面的要求会更高 Egg.js 为了解决这个问题，主要是从定制规范和给团队架构师更大的灵活性两方面入手，同时也从我们内部的日常开发中提取了许多经验集成到框架中。

相较于 koa/express 来说，Egg.js 首先约定了一套代码目录结构，清晰的定义了从配置、路由、扩展、中间件到控制器、定时任务等各个 Web 应用研发过程中一些最基础的概念，这样不同团队的开发者使用框架写出来的代码风格会更一致，接手老项目的上手成本也会更低。

同时 Egg.js 通过插件机制，让团队架构师可以更轻松的整合企业内部服务，定制一个企业内部的框架，通过定制化的框架来统一输出一个技术栈，业务开发人员可以不去了解细节，即可接入企业的研发流程、内部服务、监控体系。

Egg.js 不仅仅提供运行时的框架，同时也在测试方面提供了很多支持。包括测试执行框架、请求库、mock 方式等都有提供，可以从 Egg.js 的单元测试文档可以看出来我们是很认真的对待这件事情的，提升应用稳定性没有捷径。

InfoQ：既然 Egg.js 以企业级开发为目标，有没有计划推出 LTS 版本？

死马：Egg.js 最近发布了 2.0，而其实我们内部还有大量的应用都还是运行在 1.0 的版本上的，因此我们仍然会同时维护 1.0 和 2.0 两个版本。后续的计划中，我们起码会保证最近两个大版本的维护。

InfoQ：由于 JS 社区活跃，Node 框架的依赖一般又很多，版本更迭带来的代码腐化速度特别快，阿里内部是如何面对这一问题的？

死马：随着 yarn 的诞生和 npm5 的跟进，其实整个 js 社区对待依赖这件事情的看法都在慢慢在向着锁版本倾斜的。但是锁版本在我们看来是一个治标不治本的方案，虽然看起来更稳定，但是业务繁忙起来没有人会愿意去升级依赖，导致一段时间后应用的依赖无法升级，许多新特性和隐藏的 Bug 无法被修复。

而在阿里和蚂蚁内部的Node实践方式来看，绝大多数复杂的功能都是由底层框架提供的，底层框架经常会有大量的新特性引入和 Bug 修复，特别是一些安全性问题非常紧急，一旦版本被应用开发者锁定，就会导致线上服务出现安全风险。所以阿里和蚂蚁内部的实践是遵循 Semver 风格的依赖引用，不锁定版本。

当然不锁版本也有它的问题，总会有模块不小心发布出了问题，所以我们内部针对这个场景有不少的优化：

- 阿里和蚂蚁内部使用自建的npmregistry和自研的npm安装模块机制，可以完全掌控所有模块，当一个底层模块出现问题时，可以快速进行版本回退。
- 通过控制研发流程，保证线上发布的代码包和测试用的代码包是同一个，避免线上运行的代码使用的依赖和测试时不一致。
- Node服务绝大部分复杂的逻辑都封装在底层框架中，由框架研发团队来确保兼容性。

InfoQ：TypeScript 被认为是企业级 JS 开发的标准，最近 Egg.js 支持了 TypeScript，本身有没有使用 TypeScript 重写的计划？

死马：Egg.js 本身不会使用 TypeScript 重写，对于框架本身而言，JS 的灵活性可以让它更容易实现一些特性，同时它也并没有那么复杂的业务逻辑，TypeScript 并无法给框架研发带来更多的帮助。

InfoQ：Egg.js 团队有没有计划推出一个官方的应用项目作为模板展示？

死马：近期朴灵发起了一个 egg-cnode 项目，使用 Egg.js 重写了 cnode 社区。

InfoQ：在国内做企业级框架有没有考虑过商业化的问题？

死马：Egg.js 暂时并没有考虑商业化的问题，毕竟框架的维护成员仍然要花大量精力在公司业务上。当然这也不意味着 Egg.js 维护力度会下降，它是阿里和蚂蚁内部数百个应用的基石。

InfoQ：Egg.js 的发版频率和 roadmap 是怎样的？

死马：Egg.js 的核心其实是很精简的，而且由于其直接服务于内部应用，所以对它的兼容性要求是非常高的，从 Egg.js 开源以来，只发布了两个大版本，Egg.js1 基于 Koa1，底层使用 co 来解决异步问题，Egg.js2 基于 Koa2，底层切换到 AsyncAwait，即便是这种升级，应用层也是可以无感知直接切换的。

Egg.js 自身的迭代采取插件化的开发机制，功能分散在不同的模块中，可能开发者在使用 Egg.js 的时候感知不到它的版本变更，但其实它一直都在进化。每周都可能有新的特性和 Bugfix 发布，Egg.js 更像是一个『改良派』而不是『改革派』，它会在兼容的前提下不断进化。

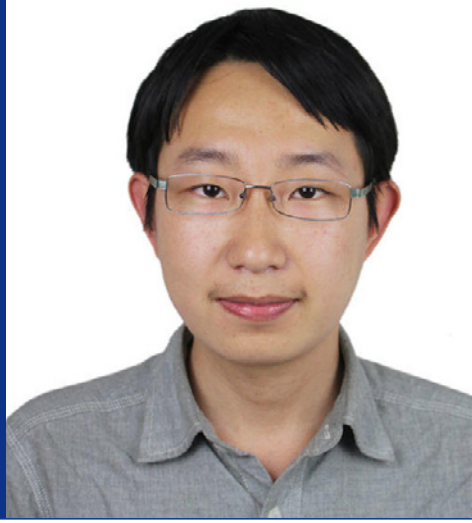
InfoQ：如何保证 Egg.js 的持续维护？

死马：首先，Egg.js 的核心维护者都是 node 社区的重度用户，除非这群人都不再当程序员，否则都会投入精力在其中继续维护它。其次，阿里和蚂蚁内部有数百个应用依赖于 Egg.js，从公司层面上来说也是愿意持续投入人力来维护它。现在 Egg.js 的运作也是完全社区化方式，讨论和代码变更都是基于 Github，有超过 100 位来自社区的贡献者，活跃度还是很高的。

不四，现就职于蚂蚁金服体验技术部，专注在 Node.js 领域 6 年，搭建了阿里巴巴和蚂蚁金服内部 Node.js 模块管理系统、中间件服务等基础设施，负责内部 Node.js Web 框架的研发和开源。同时也是开源爱好者，Node.js Web 框架 Koa.js 和 Egg.js 的核心开发者，cnpm 中国镜像维护者。

Docker？Kubernetes？容器生态圈现状如何？

作者 张磊



Docker 前路如何？

去年 Kubernetes 已经赢得了容器编排战，越来越多的提供商在拥抱 Kubernetes。而今年 3 月份，在 Docker 公司刚满五周岁的时候，创始人 Solomon 宣布离职，更多的人开始质疑 Docker。但是不要忘了，Docker 公司才是这些“拥抱 Kubernetes 的提供商”最具有分量的那位。更何况早在一年前，Solomon 就已经开始不负责 Docker 公司的具体业务了，所以他的离任在公司层面不会产生太多影响。而 Solomon 此前在 Docker 公司所做的一系列举措，实际上就是一次赌博。一家技术创业公司身处以小博大的处境，做出这样的决策其实并没有太大的问题。相信彼时的 CoreOS 等竞争对手，也曾一度笼罩在 Docker 公司赌赢的恐慌中。即使到了最后，做出牺牲的只是 Swarm 等几个具体项目，Docker 公司还是完成了预期的战略转型。

而另一方面，Docker 公司一向擅长的开发者关系工作，依然保持着强大的战斗力，这是云计算尤其是公有云提供商的核心竞争力之一。尽管迫于形势 Docker 公司目前主要专注于企业服务等更加现实的业务上，一旦有某些公有云厂商回过神来，Docker 公司的这部分价值会立即凸显出来，我们不妨拭目以待。

在技术层面上，containerd 项目距离成为 Kubernetes 下一代默认容器运行时只差了一层窗户纸。LinuxKit 项目则成功牵制

了 Unikernel 的崛起势头，还进一步完成了 Docker on Mac 和宿主操作系统封装的重要任务。如果说 Docker 公司的势头在 2015 年到 2016 年期间达到了巅峰，那么现在充其量也只是有所衰减，并且依然是容器技术圈的执牛耳者。更何况，Docker 公司的巅峰，已经超越了现有任何基础设施领域开源公司所能达到的高度。

相比 Kubernetes 项目的强势崛起等因素，CoreOS 公司提前被收购才是目前 Docker 公司面临的现实障碍。作为 Kubernetes 社区的标杆玩家，CoreOS 的收购案几乎划定了这一领域并购的天花板。在这样的背景下，Docker 公司要突破这个天花板倒不算困难，但要重现当年微软 40 亿美金报价的无限风光，希望就非常渺茫了。

Kubernetes 落地的技术障碍

作为基础设施领域的系统软件，工作层级太低是目前 Kubernetes 在更多场景中落地的主要技术障碍。

这就好比在 Google 内部，我们所熟知的基础软件如 MapReduce，GFS，Dapper，BigTable，Spanner 等都依赖于 Borg 的存在。Kubernetes 的工作层次使得它在落地的过程中表现出来的侵入性非常强。很难去真正接管用户的全套基础设施体系来发挥出它的“容器化“能力。

另一方面，很多团队（包括很多技术能力很强的团队）在落地 Kubernetes 项目的过程中也

存在一些误区，比如依然试图按照“试点测试”、“定制开发”、“内部推广”的思路来在运作这个开源项目。而实际上，Kubernetes 项目的核心乃在于它鼓励的并不是单纯的“用”，而是深入的“玩”。它是在从代码层面保证参与者成为整个生态的一部分，这跟之前大部分基础设施开源项目的设计都是不太一样。

这也意味着，只有能“玩”好 Kubernetes，才能真正保证团队在这次容器技术的浪潮中站稳脚跟。事实上，无论 Microsoft、RedHat、CoreOS，还是 Google，都不是 Kubernetes 的典型用户，但这并不妨碍他们成为这个领域数一数二的“玩家”。这其中的微妙差异，正体现了所谓的基础设施项目的“民主化”设计思路，也是我们在 ArchSummit2018 上的一个重要议题。

至于其他的功能或者性能问题，以目前 Kubernetes 生态的体量来说，基本上不存在非常困难的状况。但是如何能够鼓励用户以“白盒”化的方式去“玩好”Kubernetes 项目，改变目前用户落地 Kubernetes 项目的思路，则是一个非常值得深入探讨的话题。

混合云环境不是目前 Kubernetes 项目的主旋律

虽然 Kubernetes 创始人 Craig McLuckie 说 Kubernetes 促成了多云，VMware 也已经和 Pivotal 还有 Google Cloud 合作推出了 PKS (Pivotal Container Service)，可以监控 Kubernetes 节点，适合多云环境，但是“跨混合云和多云环境”本质上是与公有云巨头，也就是 Kubernetes 和 CNCF 背后几位重要的支持者的现阶段利益（最大程度的争取最大规模的开发者）是相悖的。

所以不难理解，混合云的支持还不是当前 Kubernetes 项目在技术上需要关心的重点。当然，作为一个完全中立的 Linux 基金会项目，生态参与者们如何将 Kubernetes 应用在混合云的环境中是一件完全自主的事情。这当然也是 Craig 为之背书的主要原因：Heptio 的解决方案必然是没有厂商锁定的。

所以，在确保用户只 vendor lock 在 Kubernetes 本身而不是某朵云的前提下，混合云的市场还是要交给 vendor 去打，比如这里提到的 Heptio 和 VMware。Kubernetes 社区中也有多集群联邦管理的能力在推进（注意：这个项目目前已经被 RedHat 基于 Kubernetes 的插件特性改造成了 Multi-Cluster 项目）。但这些并不是 Kubernetes 社区的主旋律，就目前情况而言，这个社区里还有很多更重要也更令人兴奋的事情要做。但需要指出的是，随着 Kubernetes 项目和社区的进一步成熟，以及网络、存储和多租户特性的完善，多集群管理的优先级必然会在未来得到提升。

在 Kubernetes 基础上的技术二次创新才是容器生态的焦点

我在总结 2017 年的容器生态圈的文章里说到在新的一年“曾经在国内外普遍开花的、以直接售卖 Kubernetes 集群为核心的各种‘CaaS’，将会沉寂许多”，但市面上从微软 Azure 到国内的腾讯云华为云都在提供“容器+Kubernetes”服务。对此我仍然坚持自己的看法：

Kubernetes 已经成为了所有公有云提供商的标配，但这并不意味着 2018 年依然是“CaaS”

们的春天。事实上，对市场变化更加敏感的容器创业公司们基本上都已经关停了公有“CaaS”服务，转而专注于企业落地或者咨询业务。而现今的容器技术圈子里真正“风声水起”的，早已不是 Kubernetes 业务本身，而是以 Kubernetes 为基础所构建出来的整个容器技术二次创新的生态。

我们不妨先来看 Microsoft。2018 年，Azure 产品线上着重发力的明星业务是 ACI（Serverless Container 服务），重点推广的开源项目是 virtual-kubelet（ACI 的 Kubernetes 插件）和 Brendan Burns 自己的 Metaparticle（Kubernetes 的 SDK）。尽管覆盖面并不相同，但这些产品的本质都是围绕着 Azure 本身的 Kubernetes 服务（ACS）逐步构建起来一系列 Serverless 业务，最终目的则是覆盖从开发者到运维人员的完整业务流程，对用户屏蔽学习曲线陡峭的 Kubernetes API。

然后来看 Google。Google Cloud 在 2018 年专门创建了一个名为 GoogleContainerTools 的 GitHub Organization，在短短几个月内密集发布了一系列基于 Kubernetes 的容器技术工具，这包括了 Kubernetes 集群专属的镜像制作项目 kaniko 和持续集成项目 skaffold。而在 Kubernetes 社区，Google 在重点发力的项目则是 Metacontroller，这是一个帮助用户快速编写符合 Kubernetes 编程范式的 API 插件的工具。

此外，Google Cloud 基础设施团队还在最近开源了基于用户态操作系统内核的 gVisor 容器项目，开始进入容器运行时安全这一新兴领域。作为 Kubernetes 的发起者，Google 这一系列举措的目的不言而喻，我们不妨小小预言一下：Google Cloud 的 Serverless 服务很快就会与大家见面了。

我们再来看 Heptio。Heptio 从创立一开始就推出的一系列开源项目，统统指向了 Kubernetes 集群运维这一传统痛点：Ark，Contour，Gimbal，Sonobuoy，分别对应 Kubernetes 集群备份与恢复，负载均衡，Ingress 和配置管理，堪称 Kubernetes 项目的“运维全家桶”。而 Heptio 也于近日放弃了原本在 sig-scheduling 的日常管理权限，转而专注于 sig-cluster-lifecycle（集群部署）的工作。不难看出，同样作为 Kubernetes 创始人之一，Joe Beda 的思路清晰而明朗。

而 CoreOS 在被 RedHat 收购之后在社区的首次发声，则是高调宣布了 Operator Framework 计划。把已经在 Kubernetes 社区取得了显著成绩的 Operator 推到了一个全新的高度，大有一统作业管理领域的野心。这个举措看似微不足道，实际上却意义重大，要知道，Kubernetes 作业管理领域的领导权，近乎等同于把持了 Kubernetes 项目的用户入口。这并非痴人说梦：同样的故事已经在前 Deis 的 Helm 项目上上演过一次，而 CoreOS 只不过把这个故事复制在了使用场景更加的广泛的有状态作业上而已。

上述四位不同体量的 Kubernetes 玩家，正是当前整个容器技术圈的一个缩影。事实上，我们已经强调过很多次，Kubernetes 项目不会成为一个新的 PaaS，它的设计初衷，不是直接在公有云上抢夺用户入口；它的发展过程，也不需要迎合诸如“Build Ship Run”、CI/CD、“不可变基础设施”等容器领域的热词。

Kubernetes 的现状和未来，是成为云计算

领域基础设施的核心依赖，然后进一步催生出构建于其上的“容器化革命”。在云计算平台竞争日趋严酷的大环境下，单纯基于 Kubernetes 的容器化 PaaS（或者“CaaS”），是完全不足以拉开不同服务商之间的差距的。而构建于 Kubernetes 之上的工具链、Serverless Container Instance、Secure Container Runtime、ACI、Fargate、FaaS 们才是接下来云平台上争抢用户的主角。

虚拟化容器技术很可能在 Serverless 场景成为主流

2017 年年底，OpenStack 基金会于 KubeCon 峰会正式发布基于 Apache 2.0 协议的容器技术 KataContainers 项目，主要目标是使用户能同时拥有虚拟机的安全和容器技术的迅速和易管理性。需要明确的是，KataContainers 并不是 OpenStack 基金会自己推出的项目，而是 Intel ClearContainer 和 Hyper runV 两个独立项目合并之后的产物。这个项目本身目前托管于 OpenStack 基金会并接受 OCI 的指导。以 KataContainers 为代表的 Secure Container Runtime，目前是 Kubernetes 社区最高优先级的特性之一，并会以 API 的方式固化在 Kubernetes 项目中，其重要性可见一斑。

这其实很容易理解，前面已经提到过，公有云上的容器产品之争，将会以各种形态的 Serverless 为主旋律。而多租户场景下高效的容器管理，Secure Container Runtime 几乎是唯一的选择。毕竟，传统玩法先创建虚拟机再创建容器的执行效率，在 Serverless 场景下可谓捉肘见襟。而 Secure Container Runtime 无需虚拟机做宿主的关键特性，几乎就是为这种业务而生。无需多做猜测，未来世界上主流的公有云提供商都会上线 Serverless Container 服务，其中将有很大一部分会基于 KataContainers 或者类似技术来提供安全的容器环境。

另一方面，基于虚拟化的容器技术为多租户下的容器化业务提供了全新的可能，不同业务运行在不同的 Guest Kernel 之上、甚至 Linux 和 Windows 业务在同一宿主上混部，都是虚拟化容器的拿手好戏。更有甚者，譬如 Google gVisor 和 Hyper linuxd 项目，会选择直接在用户态运行定制后的 Guest Kernel，从而在保证虚拟化级别的隔离能力的同时进一步降低 Sandbox 带来的性能损耗，在诸如 Serverless 等特定场景里，这样的安全容器技术将很有可能取代 Docker 成为主流，甚至间接促使诸如 GAE 等传统 PaaS 产品的”重生“。

方国伟（William Fang），平安科技 CTO 兼总架构师，ArchSummit 联席主席及 Keynote 演讲嘉宾。毕业于清华大学计算机系，获硕士学位。曾担任华为中央软件院架设部部长，亚马逊 AWS 中国的首席云技术专家，微软全球服务部门云计算“卓越中心”（COE）团队资深架构师。他曾主编《让云触手可及》和《详解微软 Windows Azure 云计算平台》，并在国内外的技术大会上多次发表过关于云计算的主题演讲。

微众银行张开翔：开源联盟链的挑战与应对

作者 金缕春

《经济学人》2015 年 10 月刊的封面文章《信任的机器》如是说——“比特币背后的技术有可能改变经济运行的方式”。

后来有着“数字经济之父”誉称的唐塔普斯科特（Don Tapscott）在他的《区块链革命》书中这样评论《信任的机器》，“银行家们喜欢安全性、零摩擦及实时交易的概念。世界各地的银行纷纷组织一流的团队去调查这其中可能存在的机会，这些团队里面还有不少的杰出技术人员。”

张开翔正是那个时候开始他在区块链技术领域的探索之旅，2015 年他加入微众银行，担任微众银行的区块链首席架构师。

而此前，张开翔在分布式系统、网络安全、海量服务等技术领域已有丰富的经验。张开翔从南京大学毕业后，从事软件开发多年；2007 年张开翔加入腾讯，一直致力于互联网业务系统架构设计和开发。

关于区块链技术、区块链技术在金融领域的运用、以及联盟链的架构和挑战，张开翔都有着最直接而深刻的理解。

关于区块链

您是如何接触并投身区块链领域的？

张开翔：腾讯这样的大型互联网公司对分布式的海量服务能力，系统和业务安全投入非常大。我个人也一贯很关注创新的技术形态，比特币出现后，出于对其“点对点电子现金系统”的兴趣，我通过阅读代码和文档，以及进行体验和实践，

了解了其相关算法和运作原理。

在 2015 加入微众银行之后，适逢区块链技术被金融业所关注，被认为是一项非常有潜力的金融科技技术。我把重点工作方向转向了区块链相关的领域，至今已经全力投入了近三年，团队在区块链底层平台建设和应用落地方面都取得了一些进展。

投身区块链这几年，您的关注点主要是什么？

张开翔：区块链被誉为“信任的机器”，其巨大的潜力和复杂的技术，以及行业里此起彼伏的各种声音，使这项技术既引人注目，又充满了神秘感，堪称“黑科技”。在这几年中，我的任务总结起来，就是“让黑科技变成明明白白”。

这一方面需要在技术上深入研究，梳理澄清这项技术的架构、难点、算法内涵，明确研究方向；同时建设一个稳定、易用的底层技术平台，让对区块链应用有兴趣的开发人员可以不需要陷入到技术细节，安全、稳妥的进行业务开发，以使这项技术在业务层面快速落地，体现其潜力和价值。另一方面，在金融业使用一种前沿的技术或者说新的运作模式来开展业务，是需要兼顾创新和稳定的，如何平衡技术的“颠覆性”和金融业务“安全合规”之间的关系，也是需要思考和把握的课题。

区块链研发最大的壁垒是什么？

张开翔：区块链技术本身是一系列成熟技术的组合，与在互联网公司时接触的分布式系统技术、信息安全技术大体上有很多共通之处。

比如互联网公司会大量的采用分布式架构，包括分布式计算、分布式存储，面向全世界用户提供服务。技术人员主要的工作除了实现业务之外，重点是处理分布式系统架构对容量、性能的影响，所用到的各种服务模型，网络模型和一致性算法如 Paxos、Raft、pBFT，在区块链体系里都有类似的涉及。同时，分布式系统里的著名的 CAP 原理，也影响着区块链系统的发展，需要思考在一致性、可用性、分区容错性等不同维度的权衡。

所以，只要是面向分布式系统的开发，同样都需要深度理解分布式系统架构原理，把技术做细和持续的优化。另一方面，相对普通的分布式系统，区块链体系和跨机构和人群的社会协作更密切相关，需要研究人员更多的去理解与博弈论、概率论、经济模型有关的知识。

区块链到底带来了哪些改变？区块链最有价值的地方在哪里？

张开翔：在分布式商业模式下，参与各方具有平等地位，专注各自领域且足够成熟。在此基础上，采用基于对等架构的技术平台实时交换和共享数据，同时接入多家合作方，可以有效提升商业上的容错性，从而避免商业上的“大而不能倒”的情况发生。

为了实现分布式商业的对等、共享与透明规则，以开源为主要特征的分布式技术得以发挥优势，区块链技术、分布式账本技术等渐渐成为了前沿科技的核心代表，因此，以区块链为代表的分布式账本技术的价值逐渐凸显。

关于微众银行

微众银行在区块链技术探索和应用落地方面有哪些经验？

张开翔：我们走的是技术研究，应用落地探索，社区运营互相结合的路线。在底层平台方面，微众银行推出了开源的企业级联盟链底层平

台——BCOS（BlockChain OpenSource），及金融版区块链底层平台——BCOS 的金融分支版本 FISCO BCOS。BCOS 开源平台，通过集成身份认证、非对称加密算法、引入技术治理功能、支持全面监管审计功能等举措，可支持多个行业的应用需求。FISCO BCOS 是基于 BCOS 平台加以模块升级、最终完成深度定制的金融版区块链底层平台，以达到促进区块链技术在金融业务的落地的目标。

在场景落地方面，2016 年 8 月，微众银行联合上海华瑞银行推出微粒贷机构间对账平台，这也是国内首个在生产环境中运行的银行业联盟链应用场景，目的是为解决金融机构间对账成本高的问题。此外，微众银行推出了区块链电子证据存管平台，并联合广州仲裁委、杭州亦笔科技三方基于 FISCO BCOS 区块链底层平台搭建了应用于司法领域的“仲裁链”，目的是为了解决金融领域的取证难、诉讼难问题。

信息安全对银行业很重要，展开区块链业务和开发时你们做了哪些安全准备工作，在实践中有哪些重要的地方？

张开翔：区块链先天的和密码学算法有紧密的结合，在数据防篡改抗抵赖，多方鉴证方面有很强的保障。我们也看到现有区块链方案在网络接入，数据读写，智能合约引擎，业务安全方面还是有一些风险。

所以，从一开始我们就在底层技术和业务架构设计上，采用立体安全策略，多层面，全面的安全防护，满足高要求的安全标准，总结一下包括以下维度：

- 对通信层，用户数据等模块进行高强度的加密保护。
- 在系统治理方面，实现准入控制，权限控制策略，保护系统安全
- 支持多种密码学算法，包括基于密码学的隐私保护算法。
- 提供安全方面的最佳实践，协助合作伙伴共同实施安全措施，保障全网安全。

关于金链盟

在联盟链的开发中面临的挑战主要有哪些？

张开翔：联盟链的应用场景通常运作在机构之间的商业合作领域，主要的挑战体现在高安全性，性能表现，系统稳定性，业务功能周全性（友好性），监管合规等五方面。

安全性不言而喻，安全是金融的生命线，各种商业场景里的财产安全，业务和数据安全是非常关键的，必须在网络通信、数据存储、身份认证，隐私保护等方面进行周全的保护。在性能方面，开发者需要评估所用技术是否能满足所要求的并发交易数、交易时延、数据容量等，如达不到，则需要进行持续的优化。

商业机构要求系统 7x24 小时的运行，提供高质量服务，对系统的稳定性要求很高，在完善系统本身的稳定性的同时，需要配套周全的监控运维系统。业务功能周全性和开发友好程度会直接影响开发效率和业务开展，开发者需要评估智能合约开发方案是否可行，业务端 SDK 是否周全，平台是否提供丰富的功能，或者是否便于进行二次开发等。

最后，开展业务应遵守有关法律法规，尤其在金融业，必须符合监管要求，包括对科技方面的要求如两地三中心部署，规避技术风险和操作风险，以及业务合规，重视和监管的协作，兼顾创新和金融稳定。

金链盟的共识的研发用了一年，共识开发的难点在哪里？

张开翔：共识算法做为区块链系统的核心组件，用于组织多个参与方协同工作，共同记账，保证数据的一致性可信性，维持系统稳定运行，其算法本身具备很高的复杂性，对性能，稳定性，安全性等各方面影响较大，且容易受区块链各节点运行质量，网络波动等多种因素影响，实现一个稳定高效安全的共识算法，需要反复验证，迭代优化。

基础技术的研究和实施没有任何捷径可走，我们从读基础理论的论文开始，完成足够的理论储备，然后进行细致的工程实现，这一年的时间里，我们的工作包括对共识算法的技术研究，方案选型，代码开发和原型测试，性能调优，生产验证等，最终的效果是实现了插件化的共识算法框架以及多种稳定高效的共识算法。

金链盟 FISCO BCSOS 的性能优化到什么程度？各方面有具体的数字指标吗？

张开翔：区块链的交易通常被认为并发较低、确认时延较长，比如 tps 在每秒 7 笔到 10 多笔这样，需要 10 分钟以上的确认时间，且交易确认是一种概率性的存在，追求的是最终一致性。联盟链里的商业场景需要高并发，短时间确认，且具备高度的确定性，这是我们在提升安全之外重点要解决的问题。

我们在 FISCO BCOS 的整体架构和交易处理全流程都进行了大量的优化，包括采用高效的共识算法，把能并行的计算并行化，减少重复计算，对关键计算单元进行升级等，目前 FISCO BCOS 单链性能满足金融场景需求，达到千级 TPS，交易被秒级确认，一旦确认就达到了最终确定性。这样的性能表现能满足大多数金融业务的要求。

对于区块链底层平台的性能，我们一直以来的观点是，单链不管性能能到达多少，总是受限的，无法真正从根本上解决问题。性能核心点不仅仅在于单链，更在于基于不错的单链性能，在区块链底层平台的架构上设计并实现灵活、高效、可靠、安全的并行计算和可平行扩展的能力。这让开发者能够灵活地根据自己业务场景的实际需要，通过简单增加机器，达到自己需要的性能。这是对性能问题从理论到实践上的一个比较彻底的解决之道。FISCO BCOS 在这方面已经做了大量的工作，支持了并行计算，包括多链、跨链、热点账户等一整套完整的解决方案。

张开翔，微众银行区块链首席架构师，FISCO BCOS 平台架构师。曾在腾讯工作多年，在分布式系统，网络安全，海量服务等技术领域有丰富的经验。目前致力于区块链平台系统建设、以及推动基于区块链的业务落地。

产品经理和研发工程师的关系经常被大家调侃，可偏偏就有同时受到研发和设计都喜欢的“别人家的产品经理”，沟通协调、对接需求、把控方向面面俱到还有好人缘。有没有人天生就是产品经理？产品经理的工作就是写需求写需求和写需求么？顶级公司的产品经理都是如何锻炼技能，提升思考能力的？带着这些问题 InfoQ 编辑采访了 Facebook 产品经理曲晓音，希望能给成为或者即将成为产品经理的同学一些帮助。

被选入 Facebook 轮换制产品经理项目的经历和担任期间的收获

我（曲晓音，以下内容以作者第一人称整理）在美国的私立文理学院 Pomona College 读的本科，学计算机和经济双专业。之前在 Atlassian 还是小公司的时候我给他们做过产品营销，还有曾在微软做过产品经理的实习的经验。因为这些经历，很自然的在大四的时候收到 Facebook 从领英上发的邀请，邀我去参加他们轮换制产品经理的活动，我也从那时开始了解轮换制产品经理的项目。经过了三轮电话面试和三轮终轮面试，很幸运的在两天之后就拿到了他们的 offer，也是第二个进入这个项目的中国人。当年我也被麦肯锡和波士顿咨询公司的美国办公室录取做咨询师，经过非常艰难的决定，最终选择了去 Facebook 做产品经理，现在看来这个决定是很正确的。

我们那届一共有 12 个人入选这个项目， 其中的 10 个应届本科毕业生大多毕业于斯坦福、麻省理工、伯克利这种工科名校，还有 2 个是哈佛和斯坦福的 MBA 毕业生。进入 Facebook 之后，我首先在 Instagram，Facebook Page 和视频广告三个产品进行轮换，每个时间是半年。从这个项目毕业之后我就去了视频产品工作，到现在已经是三年的时间了。

轮换制产品经理项目非常有意思，Facebook 从 2012 年开始这个项目，当年的很多人现在已经成了高级产品主管的公司中流砥柱，负责很多重要项目。其实我们也是借鉴了谷歌的产品经理项目的模式，Yahoo 的前 CEO 玛丽莎·梅耶尔曾经在谷歌担任产品经理，她建立了谷歌后来闻名遐迩的产品经理项目，每年在全球挑选和培养三四十个刚毕业最聪明的年轻人，让他们一上来就直接开始带团队管产品。这些年轻人会得到对公司成功非常重要的顶级项目，并且会有专人对他们进行各方面的培训，还会组织一次环游世界的游学，玛丽莎亲自带领他们和世界各地的用户交流，拓展他们的视野。

红杉资本的首个女性高级合伙人 Jess Lee 之前担任 Polyvore 公司 CEO（被雅虎 2 亿美元收购），就出身于玛丽莎开创的这个谷歌产品经理训练营，在谷歌担任了 4 年的产品经理。这个项目还出了 Dropbox 的产品总监 Jeff Bartelma，融资上亿美元的任务管理创业公司 Asana 的创始人 Justin Rosenstein，被 Salesforce 以 7.5 亿美元收购的 Quip 创始人，曾任 Facebook 首席技术官的 Bret Taylor，A/B 测试公司 Optimizely 的两位创始人 Dan Siroker 和 Pete Kooman 等等，堪称人才辈出。

我个人在参与 Facebook 的产品经理项目一年半的时间中，快速跟进各种类型的产品，需要很快适应不同的团队。比如 Instagram 当时就是有亿万用户的产品，Facebook Page 则是针对中小企业的商业产品，而视频广告是针对广告商的诉求，这些产品目标群体，成熟阶段和团队文化都差得非常远。在这个过程中我看到了进展缓

慢、杂乱无章的团队，与有效率有激情的团队之间运营和管理模式的区别，这帮助我在现在管理团队的时候能够趋利避害，绕过很多坑。

从进入这个项目一开始，我就担任产品经理的工作，也就是说团队的人不论多高级的工程师都自然地把我视为团队的领导，虽然在那个时候，我还是一个没有多少工作经历的黄毛丫头。所以一开始真得特别有挑战，亚历山大。开始负责的第一个产品是一个比较小的功能，后来我的项目一个比一个大，做的决定一个比一个多，工程团队也一个比一个大，这也是 Facebook 有意设计的一个循序渐进的过程。

公司非常注重这个项目里的年轻人，所以特意安排我们参加一个为期两周的世界旅行。我们去了南美洲和非洲的三个国家，去和不同文化环境下的 Facebook 用户沟通，了解他们的不同诉求，去感受网络状况截然不同的、经济条件千差万别的人们怎么使用 Facebook。这是公司的传统，每一届的轮换制产品经理都会和公司高管一起去三个不同的国家，增强他们的国际视野。而通过这个过程，我们也能很好地和其他的年轻产品经理们增进友谊，是一笔宝贵的财富。

不同公司内产品经理文化的异同

在 Facebook，非常注重工程导向，喜欢用数据说话。很多产品功能的设计和发布与否的决定，都取决于成功数据指标。这包括短期功能指标也包括长期产品线成功指标。我们有非常强大的数据分析和 A/B 测试系统，能够非常清晰地了解新产品在世界各地的使用情况，也有非常聪明的数据科学家帮助我们发现数据中有意思的地方，从而改进产品策略。

Instagram 相对而言，是一个很注重产品体验跟设计感的公司。产品经理们喜欢从用户的内心体验和诉求出发，去思考产品的功能如何满足这样的诉求。产品设计师在公司具有很大的影响力，尽管数据很好看，但是如果体验本身不符合 Instagram 的产品形象，可能也不会发布。

Microsoft 更流程导向。公司毕竟已经非常大了，各方面都有成熟的流程和要求。这可能导致项目进度没有 Facebook 这么快，但是在文件信息存档方面优于 Facebook，即使一个产品换了产品经理，也能很快了解产品进度和相关信息。这方面，Facebook 更喜欢口头做决定，很多决定没有落实在纸上，进度虽然快，但是一旦调换产品经理，新来的人需要花很久时间才能跟得上。

面试不同等级产品经理时的出题思路

我比较在乎面试者的产品感和分析问题的能力，其实在这两个方面，不同级别的产品经理我问的问题很相似，因为我觉得级别并不一定代表这两方面能力的高低。我会让面试者从头到尾设计一个产品，看看他思考的过程，如何从用户需求出发，如何权衡取舍，如何设计用户体验。我也会让面试者分析某个数据突然下滑，有什么方式能够细化问题，如何找到问题的要害。这些问题，都不需要面试者用某个领域的专业经验或者以前的工作经验来解决，我在乎的是他们如何思考一个新的问题的能力。

如何获得面试官的青睐？

我认为非常优秀的面试者是一个非常好的思考者。比如一个简单的产品指标怎么定的问题，

优秀的 PM 面试者会告诉我短期的小目标是什么， 长期的大目标是什么。比如一个帮年轻人找导师的产品，一个特别好的面试者告诉我，这个产品最重要的是帮助年轻人找到好的职业导师，从而帮助年轻人更好地提高自己。长期来看，我们可以看这个年轻人的薪水有没有涨，和这个导师的联系频率；短期来看，我们可以看看每个用这个产品的年轻人找到了几个导师。产品刚发布的时候，之前的这两个指标都不重要，因为年轻人必须有导师搭理他们才会继续用这个产品，所以更重要的是年轻人给导师发消息后导师的回复率。

产品经理是天生的么？

我认为产品经理最重要的能力第一是沟通能力，能够和不同性格风格的人沟通；第二是影响力，能够让团队充满激情，鼓舞团队；第三个是方向感，能够给大家描绘一个成功的产品体验长什么样，帮助大家编织一个可以付诸行动的梦想。这三个能力，我倒不觉得都是天生的。第一个沟通能力，来自于产品经理积极和不同的人聊天，对他们的需求和性格的敏感捕捉的能力；第二个其实有点像销售的能力，能够鼓舞大家，我觉得多做其实也是可以提高的；第三个，我觉得需要对其他学科知识的积极学习，对新生事物的热情，和不断学习的能力。这三个在我看来都不是所谓天生的，而是后天可以培养的。

从事技术的人一般有比较强的逻辑思考能力， 有比较强大的分析思考能力的人会更擅长把控方向，这是一个可发扬的优点。可能会欠缺销售的能力和沟通的能力，这一点，我认为可以通过私下多和其他人沟通，积极参加一些组织活动来锻炼。之前运营的人销售能力和沟通能力比较强，但是产品感和方向感稍微略逊，还有就是可能对技术完全不了解，这也是可以通过后天的学习来提升的。

对于缺乏经验的选手，最重要的是突出自己已经有的优势。我们在乎他们的潜力，而潜力就是展示自己已有的优点，让面试官觉得是可以培养其他技能的。所以对于技术也好还是运营也好，一定要先思考自己的优势和特点，如果把你放在人堆里，你的什么特点是可以脱颖而出的，然后在面试中淋漓尽致地体现这些特点。毕竟对于这类经验缺乏的选手， 最重要的是让大家记住你，让面试官有这个人不一般的感觉。

产品经理与产品运营的协作

我觉得产品运营对于产品的成功至关重要。很多人说产品运营的门槛低，什么人都能干。其实找到一个优秀的运营人才，非常难，有了优秀的运营人才，作为产品经理才能把自己的愿景付诸实践。我以前认识一位非常优秀的运营人才，总是能另辟蹊径，把我产品功能的一些天马行空的想法付诸实践，制定合理的运营流程让这些想法能够投入实际。一个优秀的运营人才，让产品用最好的体验满足用户的需求。

在和运营团队的合作之中， 我认为有以下几点需要注意。

第一，工作边界。一般我会和运营团队提早沟通，向他们介绍产品的计划，提前商量好合作的模式和负责范围，以及产品开发和运营团队各自的成功指标。尽早和运营团队进行合作，而不是等到产品发布的那个时候才开始合作，能够避免很多问题。

第二， 给其他人机会。我个人的理念是，如果其他人有好的产品想法，即使看上去提的问题是产品经理会做的，我也会让这些人有机会展示。毕竟，我的团队有 40 多个各个职能的同事们，指望我一个人把所有东西都想到是根本不可能的，我觉得一个好的产品经理是能让团队里的每一个人都有能动性，思考产品经理的问题。我并不觉得他们提出产品经理会做的事情，就说明我这个产品经理当得不好，相反，我觉得能够激发团队能动性的产品经理才是好的团队领袖。

第三，我一直觉得内容运营和用户运营本身就是产品体验的一部分。比如当年 Airbnb 要是没有运营团队拍摄高质量的房屋照片，根本没有后来的客户增长和成功，我觉得运营团队本来就应该是产品体验的一部分，作为产品经理，我一直把内容运营或者用户运营想成是产品的一个功能，和增加一个按钮，设计一个留言板一样，都应该是产品体验的功能。

组织高效率的需求评审会

我觉得需求评审会的目的有这么几个。

第一，让全组人都了解产品需求解决了哪些问题，并且让大家相信这些需问题确实存在，而且确实需要现在来解决。我们会提前通过用户调研、数据分析等方式展示已有的信息，鼓励大家积极提问题和提出质疑，并且确保每个人都了解我们到底解决的问题是什么，为什么要现在解决。这是非常重要的，一定要在讲述需求之前先明确解决的问题。

曲晓音 ，现任 Facebook 产品经理，管理一个 40 人的产品研发团队，负责视频产品，包括网红社区、视频体验、音乐产品等产品线，在科技和娱乐产业的交叉领域领导发布过多个千万级用户产品。曾在 Instagram，微软担任产品经理，在上市之前的 Atlassian 公司负责市场营销和商业拓展。

微服务架构为什么需要配置中心？

作者 杨波

一、介绍

在系统架构中，和安全、日志、监控等非功能需求一样，配置管理也是一种非功能需求。配置中心是整个微服务基础架构体系中的一个组件，如图 1，它的功能看上去并不起眼，无非就是简单配置的管理和存取，但它是整个微服务架构中不可或缺的一环。另外，配置中心如果真得用好了，它还能推动技术组织持续交付和 DevOps 文化转型。

本文介绍在分布式微服务环境下，应用配置管理背后的业务需求，配置的各种分类和一些高级应用场景。

二、配置定义和形态

配置其实是独立于程序的可配变量，同一份程序在不同配置下会有不同的行为，常见的配置有连接字符串，应用配置和业务配置等。

配置有多种形态，下面是一些常见的（见图 2)：

- 程序内部hardcode，这种做法是反模式，一般我们不建议！
- 配置文件，比如Spring应用程序的配置一般放在application.properties文件中。
- 环境变量，配置可以预置在操作系统的环境变量里头，程序运行时读取，这是很多PaaS平台，比如Heroku推荐的做法，参考12

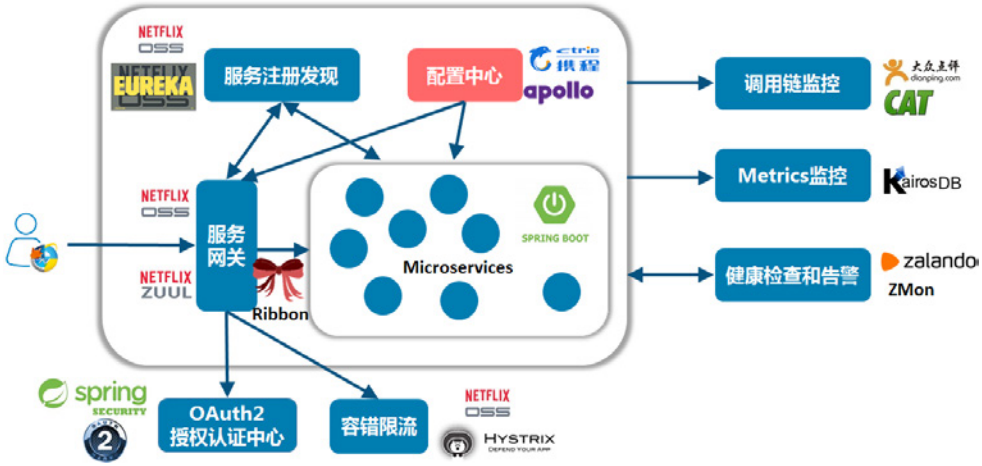


图 1

factor app[附录9.1]。

- 启动参数，可以在程序启动时一次性提供参数，例如Java程序启动时可以通过java -D方式配启动参数。
- 基于数据库，有经验的开发人员会把易变配置放在数据库中，这样可以在运行期灵活调整配置，这个做法和配置中心的思路已经有点接近了。

图 2

三、传统应用配置的痛点

在没有引入配置中心之前，一般企业研发都会面临如下痛点：

1. 配置散乱格式不标准

有的用 properties 格式，有的用 xml 格式，还有的存 DB，团队倾向自造轮子，做法五花八门。

2. 主要采用本地静态配置，配置修改麻烦

配置修改一般需要经过一个较长的测试发布周期。在分布式微服务环境下，当服务实例很多时，修改配置费时费力。

3. 易引发生产事故

这个是我亲身经历，之前在一家互联网公司，有团队在发布的时候将测试环境的配置带到生产上，引发百万级资损事故。

4. 配置缺乏安全审计和版本控制功能

谁改的配置？改了什么？什么时候改的？无

图 3

从追溯，出了问题也无法及时回滚。

四、现代应用配置核心需求

近年，持续交付和 DevOps 理念开始逐步被一线企业接受，微服务架构和容器云也逐渐在一线企业落地，这些都对应用配置管理提出了更高的要求。（见图 3）

1. 交付件和配置分离

传统做法应用在打包部署时，会为不同环境打出不同配置的包，例如为开发 / 测试 /UAT/ 生产环境分别制作发布包，每个包里头包含环境特定配置。

现代微服务提倡云原生 (Cloud Native) 和不可变基础设施（Immutable Infrastructure）的理念，推荐采用如容器镜像这种方式打包和交付微服务，应用镜像一般只打一份，可以部署到不同环境。这就要求交付件（比如容器镜像）和配置进行分离，交付件只制作一份，并且是不可变的，可以部署到任意环境，而配置由配置中心集中管理，所有环境的配置都可以在配置中心集中配，运行期应用根据自身环境到配置中心动态拉取相应的配置。

2. 抽象标准化

企业应该由框架或者中间件团队提供标准化的配置中心服务 (Configuration as a Service)，封装屏蔽配置管理的细节和配置的不同格式，方便用户进行自助式的配置管理。一般用户只需要关注两个抽象和标准化的接口： 1. 配置管理界面 UI，方便应用开发人员管理和发布配置， 2. 封装好的客户端 API，方便应用集成和获取配置。

3. 多环境多集群

现代微服务应用大都采用多环境部署，一般标准化的环境有开发 / 测试 /UAT/ 生产等，有些应用还需要多集群部署，例如支持跨机房或者多版本部署。配置中心需要支持对多环境和多集群应用配置的集中式管理。

4. 高可用

配置中心必须保证高可用，不能随便挂，否则可能大面积影响微服务。在极端的情况下，如果配置中心不可用，客户端也需要有降级策略，保证应用可以不受影响。

5. 实时性

配置更新需要尽快通知到客户端，这个周期不能太长，理想应该是实时的。有些配置的实时性要求很高，比方说主备切换配置或者蓝绿部署配置，需要秒级切换配置的能力。

6. 治理

配置需要治理，具体包括：＊配置审计，谁、在什么时间、修改了什么配置，需要详细的审计，方便出现问题时能够追溯。＊配置版本控制，每次变更需要版本化，出现问题时候能够及时回滚到上一版本。＊配置权限控制，配置变更发布需要认证授权，不是所有人都能修改和发布配置。＊灰度发布，高级的配置治理支持灰度发布，配置发布时可以先让少数实例生效，确保没有问题再逐步放量。

五、配置分类

配置目前还没有特别标准的分类方法，我简单把配置分为静态和动态两大类，每一类再分为若干子类，如图 4。

1. 静态配置

所谓静态配置，就是在程序启动前一次性配好，启动时一次性生效，在程序运行期一般不会变化的配置。具体包括：

1.1 环境相关配置

有些配置是和环境相关的，每个环境的配置不一样，例如数据库、中间件和其它服务的连接字符串配置。这些配置一次性配好，运行期一般不变。

1.2 安全配置

有些配置和安全相关，例如用户名，密码，

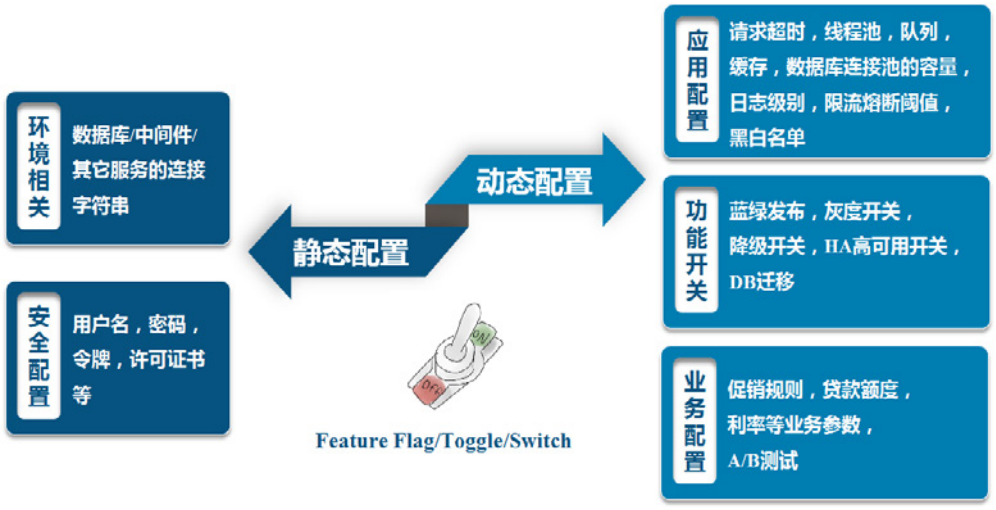


图 4

访问令牌，许可证书等，这些配置也是一次性配好，运行期一般不变。因为涉及安全，相关信息一般需要加密存储，对配置访问需要权限控制。

2. 动态配置

所谓动态配置，就是在程序的运行期可以根据需要动态调整的配置。动态配置让应用行为和功能的调整变得更加灵活，是持续交付和 DevOps 的最佳实践。具体包括：

2.1 应用配置

和应用相关的配置，例如服务请求超时，线程池和队列的大小，缓存过期时间，数据库连接池的容量，日志输出级别，限流熔断阀值，服务安全黑白名单等。一般开发或者运维会根据应用的实际运行情况调整这些配置。

2.2 业务配置

和业务相关的一些配置，例如促销规则，贷款额度，利率等业务参数，A/B 测试参数等。一般产品运营或开发人员会根据实际的业务需求，动态调整这些参数。

2.3 功能开关

在英文中也称 Feature Flag/Toggle/Switch，简单的只有真假两个值，复杂的可以是多值参数。功能开关是 DevOps 的一种最佳实践，在运维中有很多应用场景，比如蓝绿部署，灰度开关，降级开关，主备切换开关，数据库迁移开关等。功能开关在国外互联网公司用得比较多，国内还没有普及开，所以我在下一节会给出一些功能开关的高级应用场景。

六、配置中心高级应用场景

场景一、蓝绿部署

蓝绿部署的传统做法是通过负载均衡器切流量来实现，如图 5 左边所示。这种做法一般研发人员无法自助操作，需要提交工单由运维介入操作，操作和反馈周期比较长，出了问题回退还需运维人员介入，所以回退也比较慢，总体风险比较高。

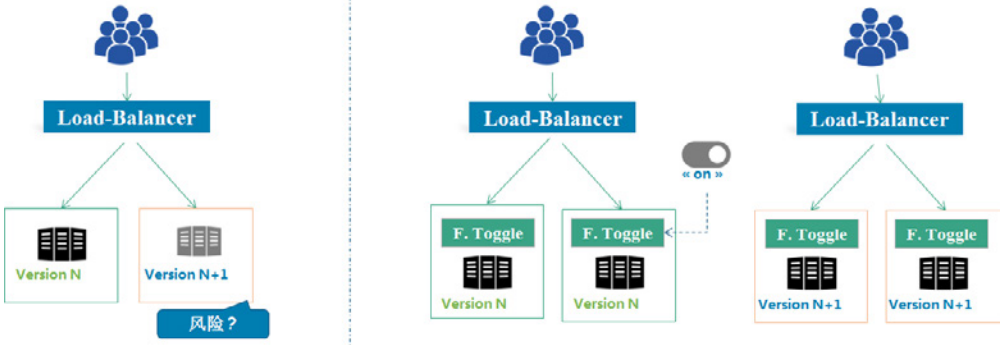


图 5

蓝绿部署也可以通过配置中心 + 功能开关的方式来实现，如上图右边所示。开发人员在上线新功能时先将新功能隐藏在动态开关后面，开关的值在配置中心里头配。刚上线时新功能暂不启用，走老功能逻辑，然后开发人员通过配置中心打开开关，这个时候新功能就启用了。一旦发现新功能有问题，可以随时把开关关掉切回老功能。这种做法开发人员可以全程自助实现蓝绿部署，不需要运维人员介入，反馈周期短效率高。

场景二、限流降级

当业务团队在搞促销，或者是系统受 DDOS 攻击的时候，如果没有好的限流降级机制，则系统很容易被洪峰流量冲垮，这个时候所有用户无法访问，体验糟糕，如图 6 左边所示。

所以我们需要限流降级机制来应对流量洪峰。常见做法，我们一般会在应用的过滤器层或者是网关代理层添加限流降级逻辑，并且和配置中心配合，实现限流降级开关和参数的动态调整。如果促销出现流量洪峰，我们可以通过配置中心启动限流降级策略，比如对于普通用户，我们可以先给出“网络不给力，请稍后再试”的友好提示，对于高级 VIP 用户，我们仍然保证他们的正常访问。

国内电商巨头阿里，它内部的系统大量采用限流降级机制，实现方式基于其内部的 diamond+sentinel 配置管理系统。如果没有限流降级机制的保护，则阿里的系统也无法抵御双十一带来的洪峰流量冲击。

场景三、数据库迁移

LaunchDarkly 是一家提供配置既服务 (Configuration as a Service) 的 SAAS 服务公司，它在其博客上给出了一片关于使用功能开关实现数据库迁移的案例文章，该案例基于其内部一次成功的数据库迁移实践，从 MongdoDB 迁移到 DynamoDB[参考附录 9.2]，图 7 展示了一个简化的迁移流程。

简化迁移腾挪流程如下： 1. 开发人员先在应用端的 DAO 层埋好数据双写双读、以及数据

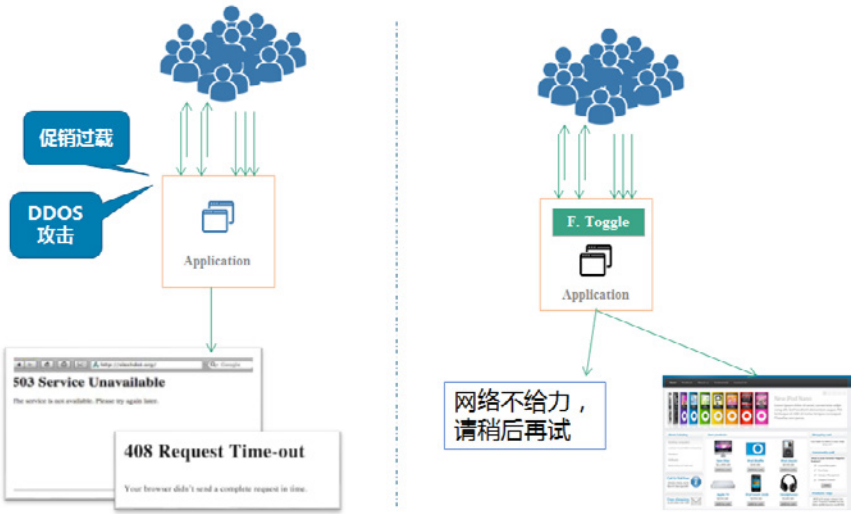


图 6

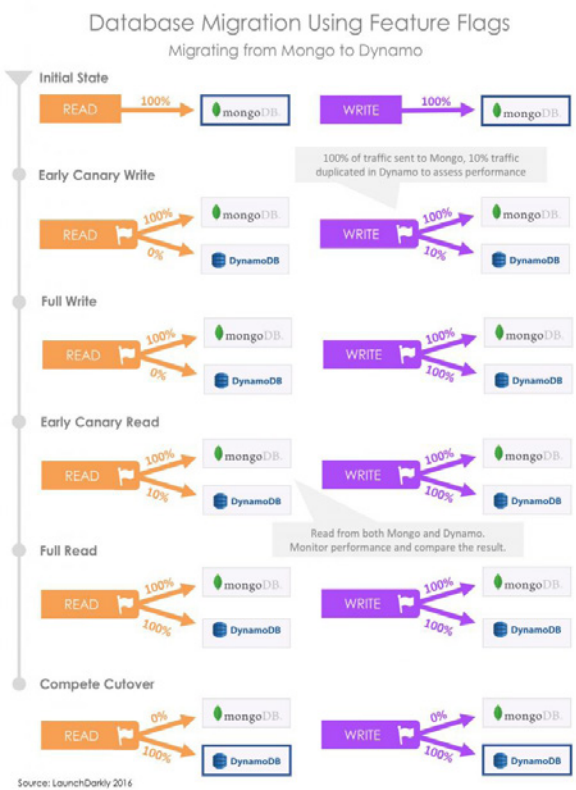


图 7

比对逻辑。双写双读逻辑由开关控制，开关的值可在配置中心配。 2. 先保证应用 100% 读写 mongoDB，然后先放开 10% 的 DynamoDB 双写，也称金丝雀写 (Canary Write)，确保金丝雀写没有功能和性能问题。 3. 逐步放量 DyanamoDB 写到 100%，确保全量双写没有功能和性能问题。 4. 放开 10% 的 DynamoDB 双读，也称金丝雀读 (Canary Read)，通过比对逻辑确保金丝雀读没有逻辑和性能问题。 5. 逐步放量 DynamoDB 读到 100%，通过比对逻辑确保全量双读没有逻辑和性能问题。 6. 关闭对 mongoDB 的读写，迁移完成。

整个迁移流程受配置中心的开关控制，可以灵活调整开关和参数，有问题可以随时回滚，大大降低迁移风险。

场景四、A/B测试

如果我们需要对电商平台的结账 (checkout) 功能进行改版，考虑到结账功能业务影响面大，一下子上线风险大，为了减低风险，我们可以在配置中心配合下，对结账功能进行 A/B 测试，简化逻辑如下图 8。

我们在配置中心中增加一个 ab_test_flag 开关，控制 A/B 测试逻辑：

1. 如果 A/B 测试开关是关闭的 (ab_test_flag==false)，那么就走老的结账逻辑。

2. 如果 A/B 测试开关是打开的 (ab_test_flag==true，并且是普通用户 (user==regular，

```
if (ab_test_flag==true
   && user==beta)
```


```
if ((ab_test_flag==true
   && user==regular)
   || ab_test_flag==false)
```

图 8

可以检查数据库中用户类型），那么就走老的结账逻辑。

3. 如果 A/B 测试开关是打开的（ab_test_flag==true），并且是 beta 用户（user=beta），那么就走改版后的新结账逻辑。

通过配置中心，我们可以灵活调整开关，先对新功能进行充分的beta试验，再考虑全量上线，大大降低关键业务新功能的上线风险。

七、公司案例和产品

在一线前沿的互联网公司，配置中心都是其技术体系中的关键基础服务，图 9 给出一些公司案例产品。

阿里巴巴中间件部门很早就自研了配置中心 Diamond，并且是开源的。Diamond 对阿里系统的灵活稳定性发挥了至关重要的作用。开源版本的 Diamond 由于研发时间比较早，使用的技术比较老，功能也不够完善，目前社区不热已经不维护了。

Facebook 内部也有一整套完善的配置管理体系［可参考其论文，附录 9.3］，其中一个产品叫 Gatekeeper，目前没有开源。

Netflix 内部有大量的微服务，它的服务的稳定灵活性也重度依赖于配置中心。Netflix 开源了它的配置中心的客户端，叫变色龙 Archaius［参考附录 9.4］，比较可惜的是，Netflix 没有开源它的配置中心的服务器端。

Apollo［参考附录 9.5］是携程框架部研发并开源的一款配置中心产品，企业级治理功能完善，目前社区比较火，在 github 上有超过 5k 星，在国内众多互联网公司有落地案例。如果企业打算引入开源的配置中心，那么 Apollo 是我推荐的首选。

百度之前也开源过一个叫 Disconf［参考附录 9.6］的配置中心产品，作者是前百度资深工程师廖绮绮。在 Apollo 没有出来之前，Disconf 在社区是比较火的，但是自从廖琦琦离开百度之后，他好像没有足够精力投入维护这个项目，目前社区活跃度已经大不如前。

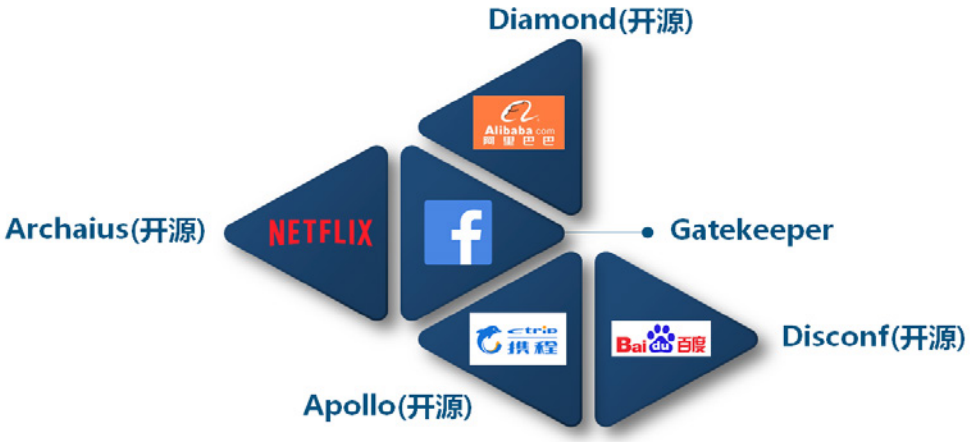


图 9

八、结论

配置中心是微服务基础架构中不可或缺的核心组件，现代微服务架构和云原生环境，对应用配置管理提出了更高的要求。

配置中心有众多的应用场景，配置中心 + 功能开关是 DevOps 最佳实践。用好配置中心，它能帮助技术组织实现持续交付和 DevOps 文化转型。

携程开源的 Apollo 配置中心，企业级功能完善，经过大规模生产验证，社区活跃度高，是开源配置中心产品的首选。

我近期和极客时间合作，推出《微服务架构实践 160 讲》视频课程，其中第二模块会对微服务配置中心Apollo的架构和实践进行深度剖析，欢迎大家关注。

九、附录

- 12 Factor App https://12factor.net/config
- 使用功能开关实现数据库迁移 https://blog.launchdarkly.com/feature-flagging-to-mitigate-risk-in-database-migration/
- Facebook的配置管理体系论文 http://sigops.org/sosp/sosp15/current/2015-Monterey/printable/008-tang.pdf
- Netflix开源的Archaius配置库 https://github.com/Netflix/archaius
- 携程开源的Apollo配置中心 https://github.com/ctripcorp/apollo
- Disconf配置中心 https://github.com/knightliao/disconf

因为AI，Blued成为垂直社交产品里“不一样的烟火”

作者 王英杰

《越人歌》

今夕何夕兮，搴舟中流。

今日何日兮，得与王子同舟。

蒙羞被好兮，不訾诟耻。

心几烦而不绝兮，得知王子。

山有木兮木有枝，心悦君兮君不知。

这是一首春秋时期有名的同性爱情诗歌。自古以来，描写同性爱情的作品数不胜数，但由于传统道德理念上的限制，同性恋情在中国长期以来成为不为大众所知的行为。近来相关管理部门对网络中有关同性恋的内容和信息的一些处理方式似乎让外界觉得，这个群体的处境仍比较艰难。而男同社交应用 Blued 的成功，让这个群体有了某种程度上的归属感，并证明了这个市场巨大的潜力。很少有人知道，它的成功和人工智能的进步有着密不可分的关系。

相比一些欧美国家，中国对同性恋的包容度似乎还是更低一些，但这并不能阻挡这个群体的生存发展，以及消费能力的增长。Blued 就诞生在这样的环境下。

和其他互联网企业一样，Blued 也开始通过时下最热门的技术——人工智能，应用于产品日常运营中，以应对越来越多涌入的新用户，以此改善体验。

面对数量巨大的用户和社交网络数据信息，如何为每个人找到身边的好友并根据用户兴趣进行匹配，成为 Blued 算法工程师面临的极大挑战。图像、视频、动态图片对于社交网站的重要性不言而喻，算法工程师很大一部分工作就是处理与视觉信息相关的数据，AI 成为他们解决问题的最佳利器。

图像社交业务

据 Blued AI 算法部数据科学家王英杰的介绍，Blued 在平台的图像社交业务中已经广泛采用 AI 技术。

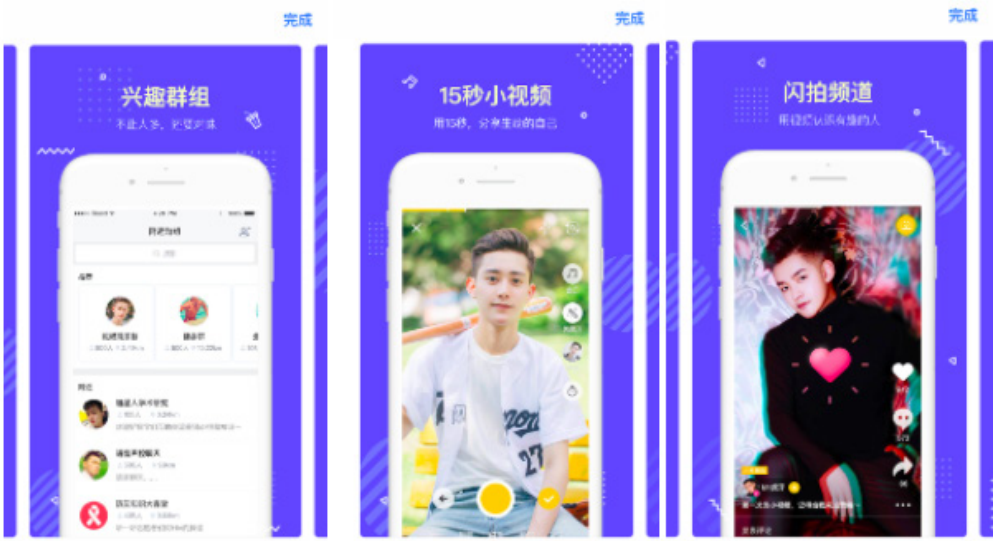


图 1

他们从 LGBT 人群的陌生人社交切入市场，逐渐转向兴趣社交和泛娱乐化平台，拓展出了很多使用场景。在这些使用场景下，用户可以在 Blued 上发布人脸头像、相册、图片动态、小视频、直播等。鉴于视觉信息在社交产品的重要地位，用户期望浏览兴趣标签下的高颜值照片，用小视频快速了解真实的对方，观看推荐的主播直播。Blued 上社交、内容、商业化模块都已有落地的应用在深度使用 AI 图像技术（见图 1）。

在社交产品上，使用人脸检测的技术筛选头像照片中含人脸的照片，并对得到的人脸特征做相似性分析；对含人图片进行体型胖瘦的分类，通过检测模型提取了一些身材和服饰上的标签，这些特征对于构建社交推荐产品的模型是非常重要的特征；使用图像分类的技术剔除掉不含人的小视频；以及使用图像检测结合图像分类的技术提取主播推荐的特征值等。

在变现业务方面，Blued 基于图像和短视频的内容 feed 流推荐产品，已经推出了商业化广告模块；图像算法在头像认证、隐私保护上的应用也是会员和增值服务等变现业务的重要组成部分。

具体到 AI 图像技术解决方案和应用算法的内部机制，王英杰解释道，Blued 的 AI 图像技术方案根据产品需求，首先拆解出几个核心的图像任务，选取适合的网络模型，如人脸检测模型、人脸识别模型、图像标签检测模型、图像分类模型等；之后利用平台上生产的大量图片做训练和微调，不断迭代完些基础模型；最后在不同的业务场景上，组合使用这些模型，并在使用的过程中根据测试结果随时调整模型输出阈值参数。从算法机制上讲，模型的迭代，数据的累积，参数的调整，形成三个并行的演进过程。同时，数据的累积促成模型的迭代，模型迭代后参数不断优化调整，参数优化调整后获得质量更好的数据积累，从而推动 () 整个系统进化。

通过这套在 Blued 内部运行了半年的技术解决方案，Blued 解决了以前靠人工审核、人工运营、产品规则解决不了的问题和实现不了的效果。比如在一些应用了 AI 技术的产品模块上，有超过 30% 的 UV 增长，人均 PV 有超过 60% 的增长，推荐成功率比人工精选提高 2 倍以上。现在，Blued 的算法模型基本上每个月都有大的迭代更新，但在与内容生产环节的配合上，和内容消费的社交转化倾向性上，还有很多需要不断完善算法、优化目标的地方。

为了体验这款产品的性能如何，AI 前线对该产品进行了体验测试。在注册 Blued 账号后，系统会通过用户选择的兴趣标签进行推荐。那么，Blued 的推荐排序机制是怎样运作的呢？

AI 前线了解到，Blued 数据平台会收集用户注册填写的基本资料信息，并结合用户在平台的内容浏览行为产生兴趣标签，Blued 会进一步探索用户的社交关系链，并将这些数据导入推荐系统。另外，在推荐算法的基础上，Blued 还会考虑用户定义的过滤和筛选条件进行排序，但主要还是以登陆时间和距离远近为原则。

作为一家LGBT社交应用，面临不寻常的技术挑战

Blued 并不是一家普通的网站，它的用户是一个特殊的群体，因此具有一些不同于普通网站的特点，并因此让工程师们面临“不同寻常”的挑战。Blued 的 AI 之路走的并非一帆风顺，很多时候，工程师们面临着应接不暇的挑战。

王英杰向 AI 前线坦承，目前，Blued 最大的技术瓶颈，是云端大规模数据并行运算，以及移动端模型运算效率问题。前者的难点在于模型计算平台和数据存储平台目前还没有打通，这一问题云计算服务已经在着手解决了。后者的难点在于目前的方案在效率和性能上还没有达到很好的平衡点，因为在移动端对算力和功耗要求较高。但王英杰相信，随着移动端技术的快速发展，这个瓶颈很快就能突破。

Blued 用户也有不同的特点，包括兴趣标签细分程度更大，用户资料真实性的甄别难度更高，用户反馈行为的分布上更不均衡，用户的频繁访问次数更多等。这些都给算法的数据和算力提出更多挑战。

而这些难题并非无解。在数据问题上的挑战，Blued 通过提取更多特征，尝试各种聚类和分类算法，特别是对数据缺失不敏感的模型，以及不依赖用户反馈行为的模型等来解决。在算力问题上，则把计算压力分配在离线计算、近线计算和在线计算上，根据数据随时调整各个部分的计算频次和计算量。

另外，社交网站往往是色情信息的“重灾区”，作为主要为 LGBT 人群提供服务的应用，Blued 还承担着向用户科普、宣传艾滋病等疾病防治方面的任务。Blued 同样在面临着这样的挑战，具体体现在色情图像、文字、低俗内容识别等任务上。

对此，Blued 在社区管理中通过人工审核团队制定严格规范的识别标准，在模型的训练和推理过程中考虑到不同分类检测类别在准确率和召回率上的不同要求，比如色情内容的检测需要更高的准确率，性感内容的检测需要更高的召回率，这反过来提高了人工审核团队的复审效率。Blued 告诉 AI 前线，他们在低俗内容的识别上面临的挑战更大一些，具体体现在 1. 判断标准随时间会发生较大的变化，而且变化较快，需要不断增减需要检测的类别；2. 样本准确标记难度

大，模型的准确率和召回率也都比较难保证。目前，Blued 还在采取诸如尝试不断完善这个模型动态更新的流程，加大人工审核的力度，增加用户举报反馈的入口等措施来解决这个问题。

未来的技术规划与探索

利用 AI 技术在产品和服务中的布局已经铺展开来，未来在技术上还会进行更多的探索。

Blued 的技术规划是 AI 优先，强调对于细分人群的个性化运营，把兴趣社交知识数据化、模型化。基于不同类型细分人群的社交需求，设计合理的产品场景，找到合适的特征，选择匹配的模型，设计如何选取正负样本和细化的优化目标函数。在这个过程中，新的产品想法成为可能，产品和运营的经验知识也在模型的训练过程中被数据化。

未来 AI 技术一定会在 Blued 产品上越来越多的体现出来，不只是兴趣社交领域，Blued 还表示将探索新的商业化机会，比如新社交和新电商的结合等。

王英杰，Blued 数据科学家，目前在 Blued（北京蓝城兄弟信息技术有限公司）AI 算法部，负责图像和推荐相关工作，包括社交、内容、直播、风控等的 AI 技术方案和实施。2007 年博士毕业于北京邮电大学，拥有多项国内和国外专利，有丰富的图像深度学习和图像处理技术经验。

TGO 鲲鹏會

关于 TGO 鲲鹏会 / ABOUT TGO

TGO 鲲鹏会是极客邦旗下高端技术人聚集和交流的组织

旨在组建全球最具影响力的高端技术人社交网络，线上线下相结合，为会员提供专享服务。

TGO 鲲鹏会采用实名付费会员制

每一位申请加入的会员都必须经过 TGO 鲲鹏会组织的严格审核，保证会员信息的真实性，让每一位会员在平等和相互信任的环境中分享交流，大家共同学习，共同成长。

TGO 鲲鹏会采用以点带面的组织架构形式

目前已经建立 TGO 鲲鹏会北京、上海、杭州、广州、深圳、成都、硅谷等分会，厦门、南京、台北等分会也将近期成立。

欲了解更多详情扫描二维码
关注TGO微信公众账号

扫描二维码
填写TGO会员申请表

01

怎么加入 TGO 鲲鹏会 /

HOW TO JOIN TGO /

在线申请

资质审核

正式加入

大咖推荐

支付费用

02

TGO 鲲鹏会入会基本资格 /

TGO MEMBERSHIP QUALIFICATION /

- 每位申请人需要至少一位 TGO 鲲鹏会会员或者极客邦科技高管推荐。
- 申请人需具备至少五年技术工作经验、两年以上的管理经验、带领过30人以上技术团队。
- 通过提交资料初审及严格的面试筛选，综合考察个人素质，最终确认是否符合入会标准。
- 9900元/年即可享有 TGO 鲲鹏会全年活动的参与权及更多会员专享权益及福利。

Geekbang. 极客邦科技

极客邦企业培训与咨询

帮助企业和技术人成长

业务简介

极客邦科技超过10年技术领域深耕，汇集超过2000位国内外一线技术专家资源，实践驱动，并具备专业完善的企业培训咨询服务流程，服务超过3000家企业研发团队，帮助企业提升技术竞争壁垒，真正让技术驱动业务发展。

- 海量技术讲师/专家咨询
- 技术领域专业洞察和知识积累
- 专业完善的企业培训咨询服务流程

产品与服务类型

公开课 | 企业内训 | 技术咨询 | 在线学习

提供9大技术领域的培训与咨询

云计算 | 大数据 | 软件架构 | 移动与前端 | 语言开发 | 运维与容器 | 人工智能 | 区块链 | 技术管理

近期深度培训公开课一览

ArchSummit架构师训练营
时间：2018.7.8–9
地点：深圳

BCCon区块链技术深度培训
时间：2018.8.20–21
地点：北京

QCon软件开发深度培训
时间：2018.10.21–22
地点：上海

CNUTCon运维技术深度培训
时间：2018.11.18–19
地点：上海

ArchSummit架构师训练营
时间：2018.12.9–10
地点：北京

AICon人工智能与机器学习深度培训
时间：2018.12.22–23
地点：北京

企业服务顾问：Rodin（罗丹） 咨询电话：15002200534 邮箱rodin@geekbang.org

扫描二维码
了解AS大会更多信息