TOP TECH

中国顶尖技术团队访谈录

[第一季]

卷首语

优秀的产品背后，必定有优秀的团队做支撑。《中国顶尖技术团队访谈录》系列采访以各个国内知名公司的IT技术团队为线索，展示他们的文化、思想与经验。

互联网蓬勃发展之后，IT的力量从企业进入民间，为更多普通人服务；同时，软件越来越快的吞噬世界，企业级开发与消费级开发互相牵引而爆发。开源文化渗透到所有IT企业的各个角落，来自开源世界的开发者涌入企业，企业内部的开发者也纷纷学习开源。不同的架构思路、不同的组织形态、不同的管理体系、不同的文化哲学随着人员在企业间快速流动而四散，形成一个个与众不同的团队。

本次的《中国顶尖技术团队访谈录》·第一季挑选的九个团队虽然都来自互联网企业，却是风格各异。规模上，有的只有几个人，有的有成百上千人；产品上，有的是社交类服务，有的是数据类服务；技术上，有的关注前端，有的关注底层。在一些通用的层面，他们会遇到类似的问题，有一些通用的思路指导他们去解决这类问题，但具体的实现却各有各的特点；在业务层面，则是大大不同。

希望通过这样的记录，能够让一家家品牌背后的技术人员形象更加鲜活，让更多人感受到他们的可爱与坚持。

本刊主编 杨赛

目录

1

工程篇

腾讯 QZONE 系统架构设计选型与变迁

本文是InfoQ中文站编辑水羽哲跟腾讯SNG平台开发中心总监孙超在2014年1月的一次采访实录。在深圳腾讯大厦的休息室里，孙超分享了Qzone在SET化方面的经验。

受访者简介

孙超，腾讯SNG平台开发中心总监，西安交通大学硕士，2006年毕业加入互联网业务系统，一直从事Qzone平台的后台开发和设计工作，先后负责Qzone的SET分布、IDC分布和SOA的系统架构的建设，见证了Qzone从百万到亿级业务的多次产品和技术变革。

InfoQ: **我是InfoQ主持人，我们现在在腾讯大厦，很高兴腾讯大讲堂帮我们邀请了孙超。孙超请您先做一下自我介绍?**

孙超：大家好，我是来自于Qzone的孙超。我从2006年毕业就加入了腾讯，一直从事和Qzone平台建设相关的一些工作，现在也正在负责整个Qzone品牌这块的一些业务。其实这里也面临着比较大的一些挑战，包括当时Qzone从一个比较小量级的几百万，然后到现在亿级的一个平台，过程中整个平台存在几次大的架构调整和优化。我非常荣幸正好参与了这样一个产品的快速发展阶段。

InfoQ: **我们在Qzone中经常了解到的一个名词就是SET，能够为我们介绍一下SET是怎样一个形式?**

孙超：说到SET，我先说一个比较简单的比喻：你可以把它理解成一个集装箱，一个SET是一个壳子，它里面装着很多货物；拿到Qzone这来说，其实SET就是把Qzone的某些服务放到一个小的盒子里面，这样来进行分布部署。

当时为什么要有这个SET，当时存在一个背景，就是Qzone从05年诞生，后面到06年、07年快速的发展，机器的整个规模从原来的一二百台快速的膨胀到了上千台，机房的建设其实对于业务的发展是有点滞后的，所以当时存在这种情况，比如说我们一个服务，从逻辑层到数据层可能分布在深圳的几个机房，这样的话，我们就会存在大量的穿越。当网络之间的专线，不管是因为变更也好，还有意外也好——比如挖断的专线等等这种情况，导致了我们整个的服务受到非常大的波动。

另外像这种用户的行为，尤其是SNS平台，它存在着很多的一些，我要拉取我好友的信息，其实有很大的扩散存在里面，所以一个简单的请求可能到下面就扩散成了几十倍、上百倍，这样对专线形成一个相当大的依赖。所以在06年、07年的时候，如果出现了一些专线的，比如交换机一些问题，或者是专线的抖动，整个Qzone就打不开。所以当时可以看到，在百度上，空间打不开的抱怨是排名非常靠前。

所以我们经过分析这些原因之后，就针对于这样的一个背景，来做了一次大的部署优化。其实SET可以认为是一个部署优化，我们的目的就是减少对专线的各种依赖，来保证在我们的核心服务当进入这样的一个SET之后，可以完整的来启用整个的功能。可以认为这个集装箱，就在这样的一个部署之后，所有的请求不需要去集装箱外面来访问了。

我刚才介绍了整个的一个背景，和当时的一个目的，SET其实我们在当时建设的时候，其实要把它建设好，要考虑比较多的因素。比如说对一个SET，我们要覆盖多大的范围，因为Qzone整个业务是非常大的，包括里面的一些基础功能，日志相册，然后还有一些平台级的服务。最终，我们经过这种圈定，当时把整个的Qzone SET定义成空间的首屏。因为空间的首屏——就是进入空间之后第一屏看到的一些业务，它这里的访问量是非常大的。而且，为了当时做整个Qzone的平台化，我们还有一个用户必须要先进入这样一个首屏才能去玩其他的一些业务的设定。既然有这样一个非常关键的点，所以我们必须要保证用户的第一跳是成功的，所以我们把所有的精力都聚焦在一定要把首屏的可用性和质量提上去。所以我们当时的SET，圈定的就是空间的整个首屏业务，它包括，比如说你有没有QQ空间，你的关系链，你的权限，你的一些底层数据等等。还有，首屏当时在07年的时候，我们的那个空间做了一个大的改版，从原来一个主要以装扮为主的、面向客人的模式，转向了一个以好友动态为主的、个人中心的模式，所以里面又会把好友的feeds当成了数据，核心数据，统一的都放到了这个SET里面去。

所以我们当时就是，SET解决空间首屏的速度和质量，圈定了平台级的核心服务在里面，这样的话范围就出来了，第二个，我们还有一些SET的标准，就是这个SET也不是

无限大就好，除了满足刚才的那一个，就是这么多的核心业务在里面，还有这个SET要提供多少用户在使用，因为你提供用户越多，扩容的频率就会越少，但是要扩一次的规模就很大，所以当时我们经过推算，发现其实在一个交换机下，当时因为交换机的一些限制，下面大概有四百多台的服务器，我们不希望一个访问跑出这个交换机，所以我们就在一个交换机内的一些机器把它充分使用，大概建立了一个当时的标准，是五百万在线的情况，这样的一个SET标准。

而后面的话，就是考虑一些容错容灾，部署的情况来建立这样一个SET。

所以刚才提到那个SET的概念，主要是当时的一个思考过程，包括上述这几点。

InfoQ: **我听说还有一些去SET化的过程？**

孙超：是这样，去SET化，首先是SET比较多了，就有一个去SET化的问题。随着我们当时从08年开始做了SET化，这样的方式不管是扩容还是运营成本都大幅下降，而且服务质量也提升非常快，所以在整个的业务中开始大量的采用这种模式，最多的时候我们的SET有30多个。

30多个SET存在一个问题：当时这个30多个SET包括后面的IDC分布之后，存在着三地，每个地方都有几个SET，合起来有30多个，这30多个里面就存在着一些配置管理，因为相当于一份数据我在三十多个SET里边都同时存在，而我访问的路由肯定是不一样的，这样路由管理的成本就变得非常的大。为了解决路由管理的问题，我们希望用去SET化的方式，让前端在使用服务的时候，不需要区分它到底是在哪里，他只需要调一个统一名字的服务，我们会根据它就近的部署的情况，自动的给他去路由到那里。

所以，这就是后面的去SET化，相当于前面使用方不用关心SET这个概念了。

InfoQ: **能不能介绍一下，承载这些SET的IDC这些是怎么来做的？因为这些SET全都是在IDC中来做承载的，来部署在IDC中，我们了解到现在是一点写多点读的方式，全国有多个IDC机房，能不能介绍一下这块的详细情况？**

孙超：这个问题很好，是这样，SET承载方式其实跟我们部署的考虑关系非常大的，首先，在一个SET中，其实核心是解决容灾问题，而容灾的话，比如说像现在的机房的故障非常的，就是多种多样，包括机架交电、交换机故障，所以我们一个SET情况下，其实是在交换机是双倍的，在机架上是尽量的在，因为我们要解决SET内的容灾，就是一

个数据总共有两份数据分别存在两个机架上，往再大里来看，它还是在一个机房，所以我们像深圳的话，之前Qzone是从深圳的这里发展出去的，所以它很多的服务都是在深圳，而且它跟QQ的依赖非常的多，所以QQ上也会调Qzone的一些服务，而QQ当时主机也是在深圳，所以我们在深圳这边，为了解决这样的一个单机房的风险，在深圳我们部署多个机房，多个机房就形成了一个相互之间，既使有一个故障，我们把它去来调度走就好，但是随着后面的发展， Qzone又要走出去，因为要解决北方用户的使用体验，包括华东地区，所以我们也是采用了三点分布，像北方、华东，还有那个华南用深圳来覆盖，每个地区也都有这样的一些情况，所以我们后面就产生了多个SET多个IDC。

InfoQ: **然后因为你是多个SET跟IDC这样的分布最主要面临的一个问题，就是数据同步的问题，然后现在我们的Qzone是怎么来解决这个问题的?**

孙超：不管是SET也好，还是IDC分布也好，其实都是把一个数据分布到多地，它必然存在的数据一致性、效率和最终一致性这样的一个问题，其实用那个CAP理论来说，的确想满足三个是做不到的。

所以刚才说的挺好的，就是我们采用的一点写多点读，但是这样的话，用户会有一些滞后感，所以我们在过程中是通过这种方式来解决：

第一是我们在所有的定义好的主写点，因为要分布的数据，我们是知道的，所以把他所有的写点集中在一个SET，我们称之为数据源SET，然后当产生数据源之后，他又会把这个数据的操作指令，或者最终的数据，通过我们做的分发系统（我们叫做同步中心），把它同步到全国的各个机房，然后重新再执行一次，就相当于把用户的行为重录了一次，当时的这个系统，这样就把那多个拷贝的数据变成最新的。

但这里的确存在一个延时和网络抖动的情况，所以我们这个同步中心里面做了很多的一些工作，比如说我们虽然公司里面有很多专线，但是这个专线的确有一些流量限制，比如说他的整个带宽因为公司全在用，也不可能一个业务全占掉，所以我们还是采用了大量的宽带业务，比如说像Feeds，这种富媒体的是采用外网传输，紧急情况下可能会切到内网，像一些底层的基础数据数据量非常小，又非常重要，比如说我加了一个好友，这样的情况下，就采用了公司的专线来做，当异常情况下，就会调度到外网去，这样就形成多个的传输通道。

然后对于时延，的确在互联网上来说，我看我好友的Feeds，虽然说他对时间性的要求不是那么高，比如说我的好友发了一个Feed，我可能不会马上就能收到，但是随着我们对用户的触达的方式越来越多，的确希望用户能快速的得到，这样跟我们的分布，其实是有一定的冲突的，但是过程中，我们做了很多的一些工作，比如说，我们刚完成了一个操作之后，会把用户重新定向他的数据源，来读取数据，这样就避免了他的数据是不完整的，而且在传输的过程中，我们保证了传输的时间都是在毫秒级的，这样用户就不会有太多的一些感知，当网络传输比如出现拥塞的情况，我们也会帮用户再重新去执行，保证数据不会丢失，等等的一些策略来，比说流量控制，SOA的一些服务质量的保证，包括网屏对我们的一些支持来保证了我们整个传输过程中的质量，还是现在做的是非常可控的。

InfoQ: **那么现在Qzone它自己是有一个云计算的平台，还是使用腾讯云的底层的架构?**

孙超：这个问题是这样的，就是刚才提到的这个云，其实这个概念的确现在比较火，腾讯可以说有两朵云，一朵是给外部的，像第三方应用、托管，这种是叫外部云，而且正在很多的一些第三方应用正在建设，而我们内部呢，因为业务的复杂度也会大大变大，所以我们是在原来外部云的一些建设基础上，比如说存储，我们的存储也是接的公司的统一的云存储，然后像逻辑层，这种我们也是统一的使用的外部云的组件，但是管理的方式是不一样的，因为外部的话，会有一些虚拟化的一些考虑，而内部的，都是访问量非常大的业务，所以应该说是在原来外部云的一些基础积累之上，针对云内部的这种用户专门做了更多的一些定制，然后把一些对内部需要不到的一些情况把它去掉，所以其实我们，空间来说，应该是90%的功能，其实都是在内部的这种存储云、计算云之上，还有一些少部分呢，是因为那个是老的一些系统，还没有往上来去搬迁，所以应该是一个往内部来转换的一个过程，因为对于我们的运营效率和质量其实帮助是非常大的。

InfoQ: **Qzone经过好几次改版，每次改版我相信后台的架构也会有相应的变化，您也负责整个SOA的整个系统的一个架构建设，能不能和我们分享一下，整个Qzone、SOA它是怎么来做得，有没有哪些新的体验，因为SOA它这个历史也比较久了吗。**

孙超：那是这样的，这个词其实最大的一个感触是在09年，当时和电商，就是腾讯内部的拍拍，我们做了一次技术交流，其实SOA做的最好的国外的亚马逊，在国内的话应该是淘宝，当然那个拍拍也是在参考类似的模式，因为他们能把这个整个的一个事务的管理，包括一些整个的事务流程做成这种服务化，而那个对于Qzone这么大的一个

服务，其实当初我们并没有提出过概念，虽然我们在做的时候，也是一个模块、一个功能，都采用一些网络传输这种低耦合的方式，但是它还的确没有做成服务化，当时遇到的，经过一次交流之后呢，感受到就是其实我们的功能复杂度，其实比那个电商来说要更加复杂一些，因为模块非常的多，这样我们的运营成本和管理成本其实非常高的。

所以在这个一个背景下，当时想就是我们一定要把空间的平台的这个服务，因为我们云做成平台了，外面支撑了非常多的一些业务，我们一定要这块一些质量和管理做好，所以当时启动了一个Qzone，SOA的一个优化项目。其实这个优化项目呢，我们当时考虑，第一，我们对外提供了哪些核心的服务，第二是我们这些服务有没有一些共性，还有呢，就是当前我们遇到了哪些一些问题，经过这几方面呢，我们梳理完之后发现其实我们整个的服务还是有很多的共性的。

第一是我们提供出去的很多的服务都是平台级的，像关系链资料、权限、这种底层的Qzone的基础数据，包括APP，还有Feeds，这些呢，其实我们可以用统一的一个网关，然后来管理。这样的，比如包括它的频率控制也好，容错，容灾也好，包括调度，完全不需要让外部的业务能了解我们内部的部署情况，用这么之间的一个统一的叫Qzone的接入网管这一层，把所有的外部的服务给它屏蔽起来，这是第一，做了第一步对外统一的屏蔽，其实一个初级化的一个大的服务，基本上是建立起来了。

第二，我们发现我们的协议其实也是非常多样的，这样的话呢，使用起来也非常的方便，而且其实在SOA里面，协议的一个标准是非常重要的一点，所以呢，我们也针对了我们对外的协议，统一改成了，当时因为那个腾讯内部有一个叫WP的协议，从无限延伸过来的，类似于Google的protobuf，也是一种字节性的协议，就发现外面在增加字段，减少字段的时候非常方便，所以我们把对外的所有的协议，统一的换了这样一个协议。

第三，我们又整理了一下，虽然我们做了第一层网络，但是后面发现内部的模块非常非常的多，像分布就是一个大的一个公共服务，因为支撑着那个几十个业务，上百个业务的分布情况，所以我们就把这种非常重要的，像分布服务，存储服务，还有这种抽象出来的，比如说评论系统，然后还有关系链的服务，因为有正向关系、反向关系，等等这些的，都是可以抽象成一些服务化的形式，通过这种抽象，我们就把原来的上百个模块减少到了几十个模块，这样就大大减少了我们整个一个维护成本，其实我们的SOA的建设呢，虽然名字叫SOA，其实不完全等同于拍拍，类似跟亚马逊一样的，这是比较有空间自己特色的，就是把我们的更多的服务、更多的一些功能抽象服务化，在上一个新功能的时候，能通过快速搭建，像搭建积木一样，而不是再多的，从原始的组建上，

比如说我们有网络框架、有存储服务，如果从那上面再重新建设，其实成本还是蛮高的，所以经过这种服务化，建设之后，我们再做一个功能，时间就大大减少了，所以呢，我们整个的SOA，其实是为了解决效率和对外的平台服务的问题来产生的。

InfoQ: **最后一个问题就是下一步会做哪些工作来做优化?**

孙超：这个问题其实也是我们最近一直在思索，也是想到了几个点，现在已经有一部分在实施了。

第一是刚才提到了像我们已经做完的工作，像我们Qzone平台级的一些基础数据的一些服务化，而现在，像Qzone的核心的动态像Feeds其实它跟业务之间，像多个业务都在接入空间，他很多的方式采用了Qzone的Feeds这种暴光展示的方式来介入的，而不仅仅是接入一个入口，而是更多的参与到用户内容里去，而Feeds这个平台涉及到的服务非常多，所以第一步，我们会把这个更核心的业务Feeds做成一个服务化的一个模式。

第二，像用户，我们之前，空间可能针对于不同的用户，他看到东西其实一样的，现在呢，我们也正在做这种精细化的运营，尤其是像不同的用户，他可能有不同的一些喜好特点，然后他可能进来之后我们给他推荐不同的一些内容，像热点的一些Feeds，就是挖掘出来的，然后呢，像那个Feeds的一些排序，Facebook很早就已经做了，然后我们也正在做这方面的事情，第二点，就是我们正在做这种智能化Feeds，这种精细化运营的一些事情。

第三，我觉得还是回到我们Qzone一定要解决用户的基本的整个服务质量的问题，像Qzone的盘子也是非常大，功能也非常多，我们要压缩成本，解决用户的那个速度，因为在不同的环境里面，访问空间的速度虽然有很多优化，但是还是有用户访问慢，所以我们还重点解决这个慢的问题，还有这个如何能把这个内部的所有的这种情况更好的能完全监控到，等等的一些情况，能做到，尽量的能发现用户使用服务中的存在的一些体验的一些问题，而不等用户来投诉，我们及时通过这种问题发现之后快速去解决它，所以大概也就这三部分的事情。

查看原文：[腾讯 Qzone 系统架构设计选型与变迁]

ArchSummit

全球架构师峰会2014

2014.12.19-20 北京国际会议中心

8折优惠
11月2日前

秉承“侧重业务场景，引领技术趋势；开放竞合，不忘初心”ArchSummit全球架构师峰会首次来到北京，作为2014年国内IT领域的年末收官之作：

- 互联网金融
- 研发体系构建
- 云计算解决方案专场
- 电商，不是搭个平台就能赢
- 转型中的SNS
- 智能硬件，更懂你
- 移动互联网，随时随地
- 云计算与大数据，从技术选型说起

参会咨询：010-89880682　邮箱：arch@cn.infoq.com　大会官网：www.archsummit.com

美团云的技术演变：先把云主机做稳定了再说别的

本文根据InfoQ中文站编辑杨赛、崔康跟美团云业务部的三位工程师唐君毅、邱剑、朱晏在2014年8月的一次沟通整理而成。在美团云的办公室里，双方沟通了美团云的状态、过去的演变历程和下一步发展计划。

受访者简介

唐君毅，美团云产品工程师。哈工大毕业，曾任积木恒硕产品总监。现负责美团云的产品研发和运营工作。

邱剑，美团云架构师。清华毕业，曾任安和创新副总经理。现负责美团云的平台研发和架构工作。

朱晏，美团网高级技术经理。清华、中科院计算所毕业，曾任百货网联合创始人。现负责美团网的系统运维、云计算工作。

背景

2013年上半年，美团发布了其公有云服务美团云。该产品一开始的背后支撑团队是美团系统运维组，到2014年6月，美团云业务部从系统运维组独立了出来，专门负责云计算方面的产品研发与运营。

独立出来的美团云业务部目前有十几位工程师。虽然部门独立，但工作中仍然跟系统运维组有紧密的配合。美团系统运维组原本就有三分之二的开发工程师，而运维工程师也全部具备编写代码的能力，因此开发与运维工程师能够进行紧密的配合。

美团云的最初版本起步于2012年7月，一开始是作为私有云计算平台来构建。第一版开发了大约2个月的时间，之后用了大概10个月的时间将美团的所有业务迁移到云平台上。现在除了Hadoop、数据库仍跑在物理机上之外，美团网的所有业务都已经运行在美团云上，内部的研发、测试平台也运行在美团内部的办公云平台上。

2013年5月，美团云开始对外提供公有云服务，截止到目前仅对外提供云主机产品。对象存储、Redis、MySQL、负载均衡、监控、VPC等服务也在研发，部分产品已经在内部以及部分测试客户使用，未来会根据产品的成熟度和客户的需求程度逐步对外开放。

稳定是公有云服务的核心价值。自从公有云服务开放以来，美团云主要的工作都放在提升云主机的稳定性，完善云主机模板、备份、监控、安全等工作上。预计在2014年9月，美团云将发布其第三个版本的更新，继续打磨其产品细节。

技术架构

美团云最初选型的时候对OpenStack、CloudStack、Eucalyptus都做过调研，调研的结果是架构设计使用OpenStack的框架，网络架构参考CloudStack，主要组件由自己开发，部分组件在OpenStack原生组件上进行了二次开发。

- 核心的云主机管理系统是自己研发，没有使用Nova。采用Region-Zone-Cluster三层架构，支持跨地域、多数据中心的大规模集群部署。采用了基于KVM的主机虚拟化和基于OpenVSwitch+OpenFlow的网络虚拟化技术。
- 镜像管理用了Glance。有一定修改，例如，多数据中心分布式支持，以及镜像替换。
- 身份管理用了Keystone。有一定修改，例如，高并发性能改进，与美团帐户系统的集成。
- 对象存储用了Swift，但Swift在写延迟方面存在性能问题，同时在OAM方面的功能比较薄弱，所以也做了一些修改和研发。Swift现在已经在为美团网内部存储量几十TB量级的业务提供服务。

之所以没有整个引入OpenStack，是因为当时调研时，感觉OpenStack的设计比较脱离美团的实际情况。例如，网络架构需要有较大的调整，同时需要有共享存储的支持。当时对美团来说，现有业务的基础设施已经基本固化，为适应OpenStack而做这样的调整基本不可接受。另外，OpenStack在很多细节方面达不到需要的级别。比如，OpenStack对跨机房多Zones的设计，它假设你机房之间的网络是完备的，这也不太符合我们的网络现状。因此，我们基于美团现有的主机使用模式和网络架构，重新设计

了网络、存储和主机管理模型，自主研发了虚拟化管理平台。同时，我们在从单机房做到多机房的时候，在Zones之间做了解耦，比如在每个机房里都放置Glance服务节点，减少对跨机房网络的依赖。正是由于这些研发工作，使得我们经过不到一年时间的磨练，就实现了美团整个基础设施完全运行在私有云上的目标。

当然，由于我们不使用Nova，就意味着OpenStack社区的很多依赖Nova的组件和功能我们基本无法直接使用。不过，相对于OpenStack大而全/兼容并包的架构，我们坚持在每一方面都集中精力用好一种技术，如主机虚拟化使用KVM，网络虚拟化使用OpenVSwitch+OpenFlow，使得整个系统的开发和维护成本相对较低，同时能深挖这些技术方案的功能特性，最大限度地压榨硬件性能。同时，由于我们掌握了基础系统的代码，使得我们可以较高的效率添加一些新的业务功能（例如，对虚拟IP，USB KEY的支持等），以及实现系统架构的升级改造（例如，对多机房架构支持等）。另外，我们对使用的OpenStack组件做的一些修改，比如对Swift的优化，目前技术委员会也在提议如何回馈给上游社区。当然了，这个还需要看社区对我们的patch接受与否，而我们也还是以满足业务需求优先。

其他方面，块存储落在本地的SAS盘上并在本地做RAID。目前我们对美团网自己的业务做RAID5，对公有云用户做RAID10。这是考虑到美团网自己的业务在应用层已经做了较完备的高可用设计的，即使掉了单个节点也不会影响到业务；但对于公有云用户而言，他们用的那一台云主机就是一个单点，所以要对他们的云主机做更好的保护。使用RAID10当然成本会比较高。我们也在考虑共享存储，当然前提是先解决了上面的稳定性和性能问题。以后会使用SSD，使块存储的性能有更大的提升。

网络方面做了分布式设计，主机上用了OpenFlow，通过OpenFlow修改二层协议，让每个用户拥有一个独立的扁平网络，跟其他用户的网络隔离。通过DNS虚拟化技术，使得不同的用户可以在各自的私有网络上使用相同的主机名字，并在每个宿主机上部署分布式DNS和DHCP以实现基础网络服务的去中心化。

运维

美团云的运维思路跟整个美团的运维思路是一致的。下面介绍的运维思路既适用于美团云，也适用于整个美团网。

运维框架可以概括为五横三纵。从横向来看，自底向上分为五个层次：

- 物理层，包括机房网络、硬件设施。我们已在开展多机房和城域网建设，从最底层保证基础设施的稳定性。为了应对大规模机房建设带来的运维成本，我们实现了Baremetal自动安装部署的Web化管理，从服务器上架之后，其他工作均由自动化完成，并可以和虚拟机一样管理物理机。
- 系统层，包括操作系统、虚拟化。我们在虚拟化基础之上采用了模板化（镜像）的方式进行管理，也对Linux内核做了一部分定制开发，例如针对OVS的兼容性做了优化。
- 服务层，包括Webserver、缓存、数据库等基础服务。我们基于Puppet工具做了统一配置管理，有自己的软件仓库，并对一部分软件包做了定制。统一配置管理的好处，一方面是避免不一致的修改，保证集群的稳定性，另一方面是提高运维效率。
- 逻辑层，包括业务逻辑、数据流。这一层的主要工作是发布和变更。在很多其他公司，业务的发布上线、数据库的变更管理都是由运维来做，我们认为这样对开发、运维的协作成本较高，所以一直往开发人员自助的方向做，通过代码发布平台、数据库变更平台实现开发和运维工作的轻耦合。在发布平台中，每个应用对应独立的集群，有一位开发作为应用owner有最高权限，有多位开发作为应用的成员可以自助发布代码。数据库变更平台也有类似的权限控制机制，并在任务执行层面有特殊的稳定性考虑，例如将大的变更任务自动调度到夜间执行，对删除数据表的任务在后台先做备份。
- 应用层，包括用户可见部分。除了跟逻辑层有类似的发布和变更之外，我们有统一前端平台，实现访问流量的进出分离、行为监测和访问控制，这对于整体的安全性有很大的好处。

从纵向来看，有三部分工作，对上述五个层次是通用的：

- 监控。从物理层到服务层的监控和报警都是运维来跟进、响应的。对于逻辑层和应用层，也是开发人员自助的思路，运维提供监控API的规范，开发可以自己创建监控项、设定报警规则、进行增删改查。监控报警之后的处理，现在有些做到了自动化，有些还没有。尤其是有些基础架构和业务之间的纵向链条还没有打通，包括建立业务容量模型，某种特定的业务形态在多少用户的情况下最高负载多少，不同负载等级下的SLA应该是多少，等等，这些模型都建立起来之后就能够进行自动化的处理。
- 安全。我们很早就部署了统一的安全接入平台，所有线上的人工操作都需要登陆relay跳板机，每个人有独立的登陆帐号，所有线上操作都有审计日志。更多的安全工作由专门的信息安全组负责。

- 流程。早期基于Jira做了一些简单的流程，但仍需要改进。现在正在针对比较集中的需求，开发相应的流程控制系统，方向也是自动化、自助化。从业务部门申请VM资源，到业务扩容的整个流程，我们正在进行上下游的打通，未来可以在Web界面上通过很简单的操作实现，也提供服务化的API，方便其他业务平台进行集成。虚拟化覆盖全业务线之后，这些事情做起来都变得很方便。

总之，美团网整体的运维思路就是：保证业务稳定运行，同时推动全面自动化、自助化。涉及开发、运维沟通协作的部分，尽可能通过自动化平台的方式，由开发人员自助完成。运维人员除了基础环境、平台建设之外，帮助业务进行高可用架构的梳理，提高代码的可运维性，以及定位和解决业务中的各类问题。

改进与演变

美团云从对内服务开始到现在两年以来，最大的一次改进就是从单机房到多机房的建设，这是跟美团网的城域网建设同步开展的。

单机房的时候，美团网业务早期曾遇到过运营商网络中断几小时的情况，期间业务不可用，非常痛苦。多机房冗余做到最理想的情况下是，即使一个机房整个断电了，业务也不受影响，当然这就意味着需要100%的冗余量，成本是比较高的。不过对于美团网来说，冗余的成本是很愿意承担的，因为业务不可用造成的损失要大于做这些冗余的成本，所以我们现在物理资源都留有50%的冗余，带宽一般会预留30%的冗余。

因为美团网的发展速度很快，去年我们一度遇到资源不够用的情况，在这上面踩了很多坑之后，开始做一些长远规划。现在美团网业务的双机房冗余已经实施了一部分，美团云也有两个机房，如果公有云客户的业务支持横向扩展，那么也可以做跨机房部署。这种机房级高可用做好了，对稳定性又是一个很大的提升，大大减少网络抖动对业务的影响，可用性SLA可以从现在的4个9做到更高。有些规模比较大的客户对服务质量会有比较高的需求，所以美团的城域网、以及未来的广域网，也会共享给我们的公有云客户。

另外上面说到我们数据库跑在物理机上，这一块现在用的是SSD，读写性能顶得上早期的三台15000转SAS，瓶颈在千兆网卡上，所以我们现在也在做万兆网络的升级改造。数据库服务以后也会开放给公有云用户使用，基础设施跟美团自身业务一致。

未来的计划

由于使用本地存储，所以现在虚拟机迁移需要在夜间进行，以减少对用户服务的影响。为了提高服务的可用性，在确保稳定性和性能的前提下，共享存储是一个不错的选择，所以我们正在测试万兆网络下的共享存储方案。另外，我们底层虚拟化机制用的KVM，本身是没有热插拔的功能，这也是我们计划要做的一件事。

现在很多客户问我们，什么时候出Redis，什么时候出云数据库，一些客户对Redis和MongoDB会有需求，Web服务想要MySQL。我们的计划是由DBA团队提供一些模板，相当于是一些专门针对Redis/MySQL做好优化的系统镜像，让客户可以直接拿来用。这可能会在下一个版本release的时候推出。

我们还会提供一些基础架构的咨询服务，这个咨询服务一方面是工程师提供的人工服务，另一方面是以工具+文档的形式，以互联网的方式将我们的最佳实践共享出去。美团网做到现在的几百亿规模，内部有很多经验积累，如果能把这些积累传递给我们的客户，能够帮助客户少走很多弯路。

查看原文：美团云的技术演变：先把云主机做稳定了再说别的

豌豆荚质量总监分享：从自建机房到云计算的演进之路

本文根据InfoQ中文站编辑杨赛跟豌豆荚质量总监高磊在2014年7月的一次交流整理而成。在豌豆荚的办公室里，双方讨论了豌豆荚在IT基础架构、工具、流程方面做过的一些事，在不同需求之间如何平衡，团队职责的划分，以及遇到的一些挑战。

受访者简介

高磊，2012年4月加入豌豆荚，现任豌豆荚质量总监，负责豌豆荚的工程生产力部门（Engineering Productivity，EP）。长期关注QA这个话题，对流程、开发技术、自动化测试、敏捷、持续集成、运维、代码库、生产力工具等方向均有所涉猎。

背景

豌豆荚作为创新工场的首批孵化项目之一，从2009年12月发展至今，用户量已经增长至4.1亿。豌豆荚的主要业务在国内，帮助用户在手机上发现、获取和消费应用、游戏、视频、电子书、壁纸等娱乐内容，在东南亚地区等海外市场也做了类似的业务探索。这样一个快速增长的系统，对IT的底层支持也是一个相当大的挑战。

基础设施的建设与增长

豌豆荚诞生于2009年12月，机房部署是从2010年年初开始。那时候因为还没有成熟的云服务可用，所以选择了自建机房的方案。到目前为止，我们已经在全国各地尤其是北京、天津地区建立了多个节点。

从对基础设施资源使用的情况来看，我们的主要业务对带宽和 CDN 资源用量会比较高；而从单一业务来看，各类数据挖掘和分析对服务器资源的占用是最大的。豌豆荚从创建一开始就是数据驱动的业务，有很强的用户行为导向，因此数据挖掘的工作量非常多。

数据挖掘主要是基于Hadoop集群。豌豆荚有一个数据挖掘团队专门做产品研发（主要是面向内部），而我们这个团队则提供硬件资源和底层的Hive、HBase等基础设施的支撑和维护。整体的数据量、计算量一直都在增长，一开始的几年增长极快，最近几年稍微慢一些，也有每年几倍的增长。

差不多在2011年左右，我们开始尝试做海外版的豌豆荚Snappea。当时评估过在海外自建机房的可行性，在考察过各个地方不同位置、不同IDC、不同运营商的选项之后，我们发现即使在进展顺利的情况下，也至少需要两三个月才能建成，这个时间成本太高。如果不自建，那就只有公有云这一个选择，正好当时我们很多工程师都自己用过亚马逊的AWS，出于时间、知识门槛、成本的考量，就决定在海外使用AWS作为我们的基础支撑。

团队

EP团队的目标很明确：在主要产品的完整生命周期内，实现一流的效率、质量和服务稳定性；至于具体用什么技术或者方法，则并不做限制。一开始我们团队比较关注流程、开发工具等方面，现在我们对CI、代码库、自动化测试、运维、基础设施建设等各个方面都做了很多工作，有时候工程师要引入一些新的基础设施相关的技术或框架，我们也会进行review它们是不是靠谱，总的目标就是让产品从开始开发到线上生产环境运行这整条路径下，其稳定性和质量都有所保证。

现在整个团队的全职工程师有不到三十人，其中运维团队有十个人，而且他们也都会承担开发任务（我们叫做SRE，网站可靠性工程师），运维过程中需要什么工具、支持系统，都是由他们自己开发。运维团队的主要工作都在维护我们自建的机房系统上，AWS上面现在平均投入的维护人力差不多只有三分之一个人。这一方面是因为AWS的维护成本确实低，另一方面也是因为我们在AWS上面的规模还不是太大。

从代码库到生产环境

我们的产品发布流程还是相对成形的。不同的产品线有不同的发布频率，比较稳定的在一周两次，有些比较早期的项目可能一天一次，没有太大的压力。

产品下一个release要发布哪些feature、发布周期设置成多久间隔，主要是由产品经理和设计师来决定，工程师实现需求。到了发布日期截止之前，从代码库的主干拉一支发布分支出来做feature freeze和最后的验收测试，到发布分支上只能做bug修复，不再接受新的feature。

有的产品线有统一的测试机制，有的产品线则主要靠工程师自己做测试。无论是哪种测试模式，在进入CI做集成之前和之后都会统一进行静态检查和已有的单元测试用例，然后才上到staging环境。从staging环境开始就属于运维负责的领域了，我们的staging没有真实的流量，但是环境跟线上是一模一样的，可以说是一直处在最新版本的服务，然后staging再跟线上环境同步代码。

这一套自动发布、部署的流程虽然也不是很完善，比如持续集成的检查点还不够多，单元测试率还比较低，不过还算跑的不错。现在AWS上也是同样的一套部署过程，当时适配起来也很快，大概做了一个星期就跑上去了。

监控

我们的监控系统要实现的目的无非是两个：实时的报警，以及可以追溯的历史数据，其他都是衍生的功能。跟大部分互联网公司一样，我们一开始做监控也都是用开源软件搭起来的，不过开源的监控软件现在越来越不能满足我们的需求。

这里面有两个挑战：

- 性能问题
- 数据采集的定制化问题

数据采集的定制化主要是涉及到一些业务数据的采集，通用的开源软件也还是要做适配，需要自己去写实现，这个其实还好。性能问题是一个更加严重的问题，这个问题来自于三个方面：越来越多的机器、越来越多的采集项、越来越高的采集频率。

以前我们监控，可能5分钟抓一次数据就行；现在我们希望做到秒级的采集。监控系统需要有实时分析日志的能力，而到机器数量增长到千台以上之后，要做秒级的采集和分析，无论是数据收集的速度还是数据分析的速度都会遇到瓶颈。

所以现在我们正在自己重写一套监控的系统，专门针对我们自建的机房体系，包括对多机房架构的支持、与资产系统的对接等等。而AWS上我们直接使用了其CloudWatch监控功能，目前来讲完全够用了。

一些挑战

因为业务跟数据密切相关，而我们部门承担了给数据分析团队提供基础设施的责任。

业务对数据报告的需求一般有两类：

1、定制化的、定期的数据指标报告

此类报告有按天的、按周的、按月的或者按小时的，一般都是比较常规的监测指标，持续监控、持续分析，中间数据保留完整，需要的计算量和存储量容易预测。这种报告需求比较容易满足。

2、按需出报告

此类需求经常是针对之前没有中间数据的监测值，之前并不知道需要针对此类数值做分析，现在忽然发现需要了，业务部门会要求一次性分析过去半年到一年跟该数据相关的趋势。此类报告往往很耗时，有时候我们做估算，一年的数据分析完毕需要用多长时间，结果可能是用我们现在的计算资源，可能要分析一个月才能产出他想要的报告，但这是无法满足业务需求的。

要提高分析速度，最直接的做法就是投入更多的计算资源——放在我们自建的机房就是扩容，如果用公有云就是起更多的实例。一方面扩容是要做的，另一方面现在AWS进入国内，我们也在考察使用AWS来做这种任务的可能性。

实际上我们用了AWS以来，也逐渐发现我们之前系统设计的并不是那么好。比如我们在海外的数据分析，原本是想用EMR的，但是发现我们现在这套东西搬过去不好直接用，只好基于EC2来做这个事情。为什么呢？因为AWS的理念是让不同的组件做不同的事，比如EC2只做计算，数据持久化存储最好都放在S3；但是我们的系统一开始设计并没有考虑这些，数据都存在本地计算节点上，如果要重构，需要投入的时间还是挺多的。

包括scaling这件事也是，现在我们基本没有用到scaling，因为我们的应用上下游之间的依赖关系太重，所以对scaling机制支持的不好。这些都是需要努力的方向。一个比较好的事情是，我们豌豆荚的工程师都是比较有情怀的，对重构这件事情比较支持。当然，这里面有投入成本和产出的考量，我们首先要满足的还是业务需求，解决业务问题，至于重构的工作，会随着我们在AWS上的业务规模更大而变得优先级更高。

最后想分享的是，EC2的reserved instance如果用好了，能够比on demand模式节省很多。我们一开始不知道AWS除了on demand之外还有reserved instance、spot instance这些玩法，最近才刚知道。Reserved instance非常适合Web Service使用，临时性数据分析用spot instance则比较合适。

查看原文：豌豆荚质量总监分享：从自建机房到云计算的演进之路

对话百度前端工程师：F.I.S与前端工业化

本文是InfoQ中文站编辑李彬对百度前端工程师、F.I.S项目技术负责人张云龙的一次采访实录，采访是在2013年6月于车库咖啡举办的百度技术沙龙之后通过邮件完成的。

受访者简介

在本次采访的时候，张云龙是百度公司Web前端研发部前端集成解决方案小组的技术负责人，负责F.I.S项目。后来，他在2014年初离开了百度，不过仍然在分享他的前端工程经验。本次采访之后，张云龙将涉及到的内容又重新整理为两篇深度文章分享在InfoQ中文站，分别是《前端工程精粹（一）：静态资源版本更新与缓存》与《前端工程精粹（二）：静态资源管理与模板框架》。读者可以关注他的微博以及博客。

InfoQ：请为我们介绍一下F.I.S，它究竟是什么？

张云龙：F.I.S，全称Front-end Integrated Solution，即前端集成解决方案。

我们发现，不管哪个前端团队，不管他们的产品有多大差别，在发展的过程中总会渐渐形成一套配合自己产品开发的【项目规范】+【前端框架】+【模板框架】+【自动化工具】+【辅助开发工具】，这些技术需求的总和就是F.I.S。

经过一年半的努力，我们和众多产品线前端团队共同探索出一套前端集成解决方案。我们相信，在支持了百度30多条产品线，覆盖从PC到移动端的众多项目之后，我们所总结的F.I.S系统是具有普适性的，它能够快速应用到绝大多数公司的前端团队中，并有效

的提升其生产力水平。F.I.S在百度孕育的过程中的经历和思考，以及F.I.S系统演变等“思想产物”对其他公司的前端团队来说，在寻找提升前端生产力水平解决方案的道路上也有许多可借鉴之处。

InfoQ：F.I.S项目是如何诞生的？是什么原因促使百度成立F.I.S团队，开发并着手进行推广？

张云龙：在F.I.S出现之前，百度的每个前端团队都或多或少有这样一套东西，然而产品线与产品线之间由于采用的技术基础不同，没办法互通有无。而且每个新人加入团队后，只有需要熟悉自己门派的规范、工具和流程后才能上手项目。另外，每个团队都需要投入一定的人力来维护自己的系统。这些工作的成本相当高昂。

而随着团队规模增大，很多产品线之前设计的系统在大团队下都会出现不同程度的问题。尤其是当性能优化摆上项目议程的时候，面对多人团队，想要较平滑的对页面性能进行优化几乎是不可能的！

为了解决这些问题，前端研发部顺势成立了F.I.S团队，希望可以找到满足这些技术需求的通用解决方案，能够整合前端开发资源，对系统进行持续的平滑的性能优化，并提升前端团队生产力水平。

当时，我们面对的是大型互联网公司内部五花八门的前端开发模式，以及各种神秘莫测的自动化工具。直到现在，我都觉得能在这样大规模的互联网公司内统一前端开发是一件非常了不起的成就。当我们用了一年半的时间，几乎完成百度前端团队统一使用F.I.S之后，才将过去所做的一切沉淀下来，希望能够为业界贡献一些新的思路和想法。而F.I.S这个产品也作为前端工业化的最佳实践开源出来。

InfoQ：为何选择Node.js作为F.I.S的基础？

张云龙：事实上，最开始在百度内部使用的F.I.S是用PHP实现的。由于当时设计思路的局限，我们认为，既然后端模板是Smarty，那就必然需要PHP环境，也就理所当然应该使用PHP作为自动化和辅助开发工具的技术选型。这样一个想当然的决定，使得我们经历了非常漫长且痛苦的开发过程，毕竟PHP不是为了做这样的事而设计的。

直到今年年初我们才意识到，虽然模板需要PHP环境运行，但并不意味着自动化工具必须用PHP开发！多么浅显的道理，但是，又是多么痛苦的领悟。很快我们把注意力转向

了Node.js。它对前端工程师有着非常强的亲和力，有各种基于Node.js的压缩、优化、校验工具，有着极高的运行性能，有npm这样强大的包管理工具……简直就是为自动化和辅助开发工具量身定做的平台嘛！因此我们毫不犹豫的选择用Node.js重写F.I.S。而且在这次重构中，我们认清了F.I.S的本质、理顺了F.I.S要专注的事情和F.I.S系统的层次关系。最终，F.I.S变成了现在大家看到的样子。

InfoQ：F.I.S如何帮助开发者提升工作效率？

张云龙：首先强调一点，F.I.S系统包含四个非常重要的部分：【前端组件化框架】+【后端模板框架】+【自动化工具】+【辅助开发工具】。诚然，自动化工具是F.I.S的重要组成部分，但如果把F.I.S单纯当作类似Grunt的自动化工具就太片面了。因此，对于F.I.S如何提升前端开发效率，可以从这几个方面讲起：

- F.I.S提供了一套高效的前端项目编译系统：该系统可以很方便地组织前端编译工具，对项目进行优化、测试、校验、打包等处理。

- F.I.S的自动化工具扩展了前端语言的三种能力：资源定位、内容嵌入、依赖声明。资源定位能力可以帮助系统隔离开发环境和部署环境之间的变化；内容嵌入功能可以帮助工程师解决资源的初等拆分合并问题；依赖声明可以构建大型的组件化系统。这些工作量都会因为接入F.I.S而省掉。

- F.I.S的自动化工具会扫描整个项目的资源生成一张资源表。这张表可以与前后端框架配合，精确地按需加载资源，平滑地优化网站性能，甚至可以利用监控数据来自动优化网站性能，从而实现自适应的网站系统。

- F.I.S的资源表支持命名空间特性：可以将一个大系统拆成几个子系统来维护，子系统之间没有依赖关系，开发、提测互不影响，从而提升团队的并行能力。

- F.I.S的辅助开发工具可以解决工程师的本地调试、数据模拟等问题，解耦前端代码对后端程序的依赖，提升前后端团队的并行开发能力。

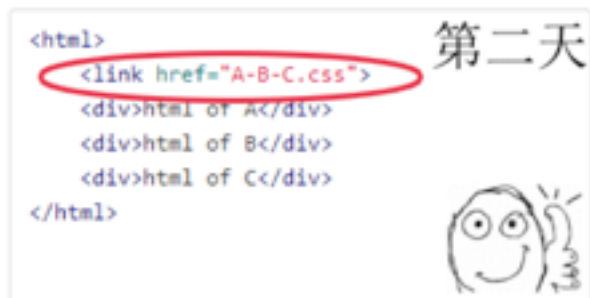
- F.I.S的辅助开发工具还提供了自动部署多台联调或测试机的功能，节省了工程师联调和提测的时间。

- 百度内部使用的F.I.S系统是有固定技术选型搭配的，因此内部项目初始团队几乎没有选型的成本，F.I.S把开发中的点点滴滴就想进去了。

InfoQ：F.I.S中带有文件编译，以及若干优化手段，那么如何评价通过F.I.S产出项目的性能？

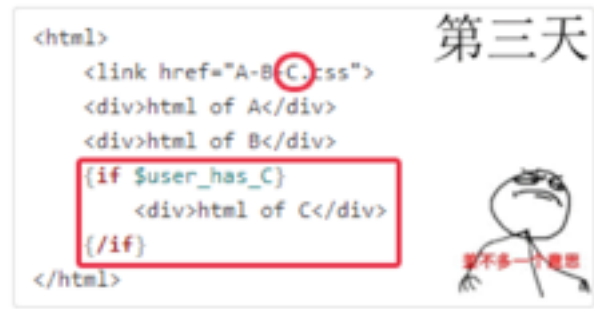
张云龙：百度内部使用F.I.S的产品线都有自己的前后端性能统计数据，百度的Web前端研发部也有性能小组来跟踪各产品线的性能指标，这些都会成为衡量F.I.S收益的重要手段。

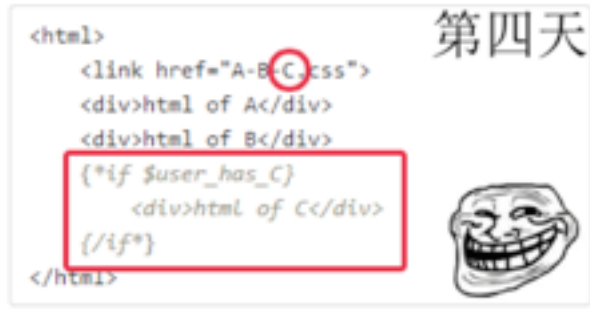
性能收益是比较容易衡量的，但F.I.S的另一项收益——生产力提升——比较难以量化。目前还只是从产品线工程师的口碑，以及一些直观的工作量减少上得到反馈。相信在各大互联网公司前端团队的类似项目中，都有遇到这类情况。虽然目前还不能以数据的形式反映F.I.S在生产力提升上的成效，但从迄今为止F.I.S在百度内部的普及程度以及产品线使用之后项目的迭代速度来看，它确实带来了很多收益。

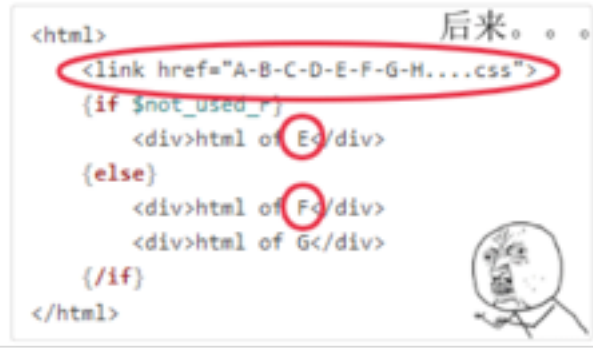


InfoQ：请为我们介绍一下F.I.S中的静态资源优化，与传统打包的差异吧。

张云龙：“雅虎14条优化原则”教导我们要减少HTTP请求，传统的静态资源打包策略采用的是简单直观的“文件合并”方式。这种方式在小团队、实验性或内部系统项目上运作的还好，因为它毕竟非常直观。但随着团队规模的壮大，这种策略非但不能优化前端性能，反倒会带来前端性能的恶化。比如Facebook在Velocity China 2010的讲座议题《静态网页资源的管理和优化》中提到的一个传统资源打包都会遇到的尴尬问题，见右图。







出现这一问题的另一典型场景是换肤和国际化。传统资源合并策略难以满足这样的需求。

最开始，F.I.S的实现思路是针对不同产品线提供不同的打包工具。很快我们就发现，F.I.S团队陷入了不断创造各种版本的打包工具的泥潭，维护成本非常高。

这迫使我们寻求通用的打包解决方案。基于资源表的静态资源管理系统就是这样诞生的。

使用资源表有很多好处：首先，打包成了资源的备份，可以非常方便地控制页面是否输出打包后的结果，线上页面很容易通过query将页面切换为输出零散的资源，以便工程师定位线上问题。其次，基于资源表的静态资源管理系统，在产品线的模板框架层实现了静态资源的调度，使得F.I.S团队不用针对每个产品线写一套独特的打包工具。而资源调度策略可以非常灵活，我们后来还实现了资源的异步加载等功能，极大提升了页面首次渲染的速度。近期，我们还在几个产品线做了静态资源调度统计的实验，通过统计数据来生成最优的打包算法，让网站性能随用户访问而自适应优化。

InfoQ：在F.I.S中，如何管理框架、插件等资源的，开发者如何添加插件？

张云龙：F.I.S提供了install命令，可以用来获取各种前端开发资源，比如示例、配置、组件、基础库、框架、甚至前端开发素材等。

我们后续会努力经营好这个资源和F.I.S的install这个入口，为用户提供优秀的、逐版本的前端开发资源。

F.I.S的自动化工具/辅助开发系统完全插件化，我们利用npm来扩展F.I.S。而F.I.S的所有插件与F.I.S核心是分离的。如果用户想在项目中使用Coffee Script、LESS、Markdown等语言来开发页面，那么只要在F.I.S安装目录的同级安装这些插件，F.I.S就能自动加载它们。F.I.S提供了11个扩展点，包括编译扩展（6个）、打包扩展（4个）和命令行扩展（1个）。所以大家看到F.I.S的GitHub项目上只有一个fis.js文件，请不要以为F.I.S项目是未完成的，因为那是我们整个系统插件化的结果啊。

InfoQ：你认为F.I.S最主要的亮点是什么？

张云龙：【语言能力扩展】、【基于表的静态资源管理系统】与【前端资源聚合】。我觉得这是F.I.S系统最大的三个亮点。

我们做了这么久的前端集成解决方案，才总结到这样一个结论：前端领域语言只要扩展了资源定位、内容嵌入和依赖声明三种能力，就可以实现绝大多数前端开发需求。有些前端自动化工具采用目录规范来替代资源定位能力，采用开发规范来替代依赖声明能力。这样做固然可行，但大大限制了其解决方案的适用范围。毕竟每个公司、每个团队都有着自己不同的开发理念，我们曾经在“开发规范”这条路上走了很久，回头看看才发现，规范都是浮云。依靠比规范更小的原子规则就可以组合形成规范，它们才是前端自动化工具的“尺规”。

基于表的静态资源管理系统设计是我们发现的另一块瑰宝。有了它，我们才能实现平滑的性能优化，自适应的网站系统，方便的线上问题定位手段，根据不同浏览器和国家地区投送不同的静态资源。而且资源表只是一种数据结构，与平台无关，很容易让fis平滑地接入到不同类型后端的系统中。

F.I.S解决了以上开发问题之后，开始向着前端资源聚合迈进。我们希望能将业界优秀的工具、代码、技术等资源聚合起来形成生态系统，让前端工程师最平等快捷的获取前端开发资源，找到所求。所以我们在插件系统设计、资源获取上下了很多功夫，虽然最后的结果看起来很简单，但那是为了让大家能用最低的学习和使用成本获得最大的收益。

InfoQ：F.I.S起自2011年，那么在过去一年多的时间里，它是如何演进的？

张云龙：F.I.S系统至今已演化了三代。

第一代（2011年末~2012年中）： F.I.S团队刚组建，我们还没有认清F.I.S的本质，以为编译就是一切。所以在参考了公司内几个比较大型产品线的前端构建工具之后，我们把其中几个运行的比较好的系统杂糅了一下，得到了第一代F.I.S——项目代号Gaea——并交付百度云相册团队试用。说真的，第一代F.I.S系统实现的很糟糕，给人一种摇摇欲坠的感觉，好像一不小心就会导致崩溃一样。它大量使用了目录规范，尽管这些规范都是来自大产品线经验的总结，但还是很难被新人接受。就这样，一代F.I.S在百度呱呱坠地，或许不那么闪耀，但起码有了一个开始。

第二代（2012年中~2013年初）：虽然一开始比较艰难，但好在靠着产品线同事的大力支持和团队每个人骨子里的那种倔强精神，我们挺过了F.I.S最艰难的时期。后来，随着F.I.S的不断铺开，我们开始面对大团队、大规模产品线的开发需求，同时也增加了对移动开发的支持。这一代F.I.S项目的内部代号是Oak，其功能处于快速增长期。我们很善于使用编译工具来解决前端开发中遇到的各种问题，与此同时也衍生出了很多针对特定终端的开发模式，包括PC、Mobile、WebApp甚至RIA等。也正是在这个时期，我们设计出了基于表的静态资源管理系统，而且F.I.S的后台界面也变得相当华丽，整个F.I.S系统的代码已增至8万行！但是很快我们意识到，肆意的使用编译能力，会导致系统产生巨大的黑盒效应。虽然F.I.S系统产出的结果非常高效，但普通工程师对其原理不甚了解，代码难于调试。而F.I.S团队在维护编译工具的路上也越陷越深，自顾不暇。

第三代（2013年3月至今）：由于2013年初所暴露的种种问题，F.I.S项目的问题开始进入不收敛时期，团队的几个负责人也认识到了问题的严重性，因此组织了几次闭门会议。这几次会议中，我们很严肃的思考了F.I.S的过去与未来，找到了问题的症结，并对关于F.I.S是什么、F.I.S的本质和理念是什么的问题做了很深入的思考。我们重新检查了F.I.S的代码，在将近10万行代码中找到解决前端开发问题的最基本的三条规则，并确定了以资源表为核心的静态资源管理系统设计理念，放弃华丽的后台界面改为命令行交互。最终我们决定，用Node.js重构F.I.S，完成对F.I.S的减法升级！这就是大家现在看到的F.I.S了。

F.I.S不是某个风和日丽的下午我们一拍脑门想象出来的，所有现在大家看到的结果，都经历了大规模大团队实战的考验。开源以后，希望业界可以看到我们曾经为解决这些棘手问题而沉淀下来的结果，相信这些经验是很值得借鉴的。

InfoQ：在下一阶段，F.I.S将会向什么方向前进？

张云龙：后续F.I.S会朝着【前端开发资源聚合】和【高性能前端架构设计】两个方向迈进，我们会不断的开源在公司内部孵化出来的前后端框架、组件、交互体验示例等资源，并对前端性能优化、开发体验、自动化工具、前端项目测试等领域做进一步的探索和研究。我们希望借助F.I.S平台可以不断整合公司内外前端资源，输出优雅高性能的解决方案，以此来提升前端工业生产力水平。

InfoQ：F.I.S目前的推广情况如何？下一步F.I.S团队对于它的推广有什么打算？

张云龙：推广才刚刚开始，比起毫无意义的广告和论战，我们团队更注重实际的产出。后续我们还会继续开放出我们在前端工业化领域所做的探索，并不断的将这些经验化作程序可描述的结果通过F.I.S平台输出给大家，我相信这就是最好的推广方式。

InfoQ：对于行业中，前端工业化整体情况做一些点评和展望吧。

张云龙：也许是我比较关注这一领域吧，我发现这两年各大互联网公司纷纷开始打造自己的集成解决方案，甚至有专门针对这一领域的岗位招聘，相信大家都已对这个方向的未来心照不宣了。五年前，前端团队能用上或者实现一个前端库来解决常见DOM操作问题已经非常不错了；三年前，前端团队能拥有一套工具来自动压缩代码就已经能解决很多开发问题了；而如今，我们要考虑的更多：框架，工具，库，辅助开发，组件化，性能优化、多终端、国际化、团队协作……新时代的前端集成解决方案是严肃思考的产物， 它能成为前端团队开发的重要利器之一。从过去到现在，从JS库到工具再到集成解决方案，相信前端人在提升生产力水平的道路上会披荆斩棘，不断前进！

查看原文：对话百度前端工程师张云龙：F.I.S与前端工业化

相关内容

- FIS2.0全新的百度前端解决方案
- 百度技术沙龙第39期回顾：前端快速开发实践（含资料下载）
- 对话豆瓣张克军：前端，工程与人

时刻关注软件开发领域的变化与创新

架构师

www.infoq.com/cn/architect

每月8号出版

InfoQ news

商务合作：sales@cn.infoq.com

读者反馈/投稿：editors@cn.infoq.com

阿里云ODPS的愿景、技术实现与难点

本文是InfoQ中文站编辑杨赛跟阿里云ODPS平台的技术负责人徐常亮在2014年3月的一次沟通的采访实录，当时，ODPS产品线刚刚在1月对外发布，而在4月即将举办的天猫天池算法大赛中，ODPS是大赛指定的数据平台。在杭州阿里巴巴的会议室中，徐常亮介绍了ODPS的愿景、技术实现和实现难点。

受访者简介

徐常亮（@常亮姓徐），北京大学双学士（主修化学，转入IT行业纯属兴趣），普林斯顿大学博士（计算化学方向），曾在纽约时报网络部任职搜索组组长，开发、维护自主开发 的搜索引擎，最早期的Amazon ec2、s3和Hadoop用户。2009年加入阿里云，曾负责阿里云分布式平台--飞天--底层基础维护，现在主要负责ODPS平台的架构和开发，产品主要满足数据仓库、分布式编程框架、数据交互等各种场景。

InfoQ：先介绍一下ODPS现在的情况吧。这个产品能做什么？

徐常亮：ODPS是2011年正式有的名称，全称叫做Open Data Processing Service，简单来说就是数据处理的服务。它的定位是在飞天之上，提供数据仓库、数据挖掘和其他数据应用等功能。

2011年的时候我们尝试对外提供ODPS，当时有一些小试点，但是后来发现各方面条件没有完全成熟，不管是外部对云的了解还是内部对ODPS未来的预期都不是很清晰，所以一直到2012、2013年，它发展的节奏都比较慢。去年大概6、7月的时候有一些

变化，因为飞天到了5K的里程碑，在技术能力方面的顾虑已经小了很多。因为飞天是分布式操作系统，它提供最基本的存储、CPU调度能力、内存使用、网络等功能，是最基本的资源包装整合，相当于是一台计算机，而我们是在它上面开发的应用，相当于是一个分布式的数据仓库，让用户可以在上面做基本的ETL处理、SQL查询、数据导入导出等，还有一些MATLAB、统计软件的功能。

除了这些基本功能之外，我们还提供了一整套数据挖掘算法Xlib，让用户可以建模、做高级的数据分析。另外，我们还可能提供一些编程框架，让用户自己可以编写程序进行数据处理，比如单机上有Python、Java，我们就提供MapReduce编程框架、或者专门为了解决迭代问题的Graph编程框架（也叫做BSP，跟Google Pregel模型很类似）。我们会逐渐加入各种内容，凡是涉及数据处理的工具和编程框架我们都会想办法加进去，让开发者和用户可以对数据进行各方面的操作。

总而言之，ODPS就是基于飞天分布式系统提供的一套关于数据处理各方面工具和框架的服务。

InfoQ：对应AWS的话，相当于是RedShift和EMR吧？

徐常亮：可以这么去对应。纯粹从功能来讲，我们会提供类似EMR和RedShift的功能。但我们不仅于此，我们还有建模的库、机器学习的库，从编程框架的丰富性上面也不仅仅是MapReduce，还有迭代框架。暂时来看我们做的可能更多，当然AWS也在逐步提供更多的功能。

另外有一个很大的不同是，ODPS是作为一个有机的整体来提供这些服务的，不同的功能是服务的不同层面，而不是单卖的功能。比如在我同一个体系里面，我数据仓库类型的一个SQL处理好了，我紧接着一个MapReduce的作业就可以很好地关联起来，他们的物理存储数据，以及描述这些数据的元数据，都是在同一个体系里面。不像RedShift和EMR，它们是在一边处理完了之后，要把数据导出到另一套系统里面去处理，它们的元数据描述不是互相共享的，要有一个第三方来做对应，比如RedShift表结构是怎样的，EMR的结构要怎样相应的去设计。ODPS希望让对象都在一起，让要处理的对象和元数据都在一个ODPS体系里。在此之上，你要做授权也好，管理维护也好，都是同一个界面，对用户而言就是在一套系统里面做不同的处理，用户觉得我就是在一台机器里面，只不过在不同的文件夹。AWS的话，用户会感知这是两台计算机。

另一个区别是，ODPS希望做成服务：open data processing service，我们希望看到用户把数据往里灌，相当于是公有云的用法，总之数据都放在同一个系统里面。如果以后用户之间希望他们的数据之间发生一些作用，则能够非常容易的做到，只需做一些互相授权就可以。而AWS的RedShift和EMR对各自用户而言都相当于是私有云，它自己处理的东西只在它自己的空间里，如果要跟外部交互，可能必须要借助S3等外界方式。当然了，可能它原本的设计目标就是这样，这个也谈不上优劣，只是目标不同。

在我们这个体系里，因为用户的东西都在一个平台上，所以我们其实也可以像苹果那样开一个应用市场，用户把数据挖掘算法或者清理流程当做一个应用来发布，别人如果想用可以来买。当然这个可能之后有一系列的如何算钱之类的问题要处理，但平台在这儿，如果商业、产品愿意多考虑，这个事情也是水到渠成的。这个和整个阿里巴巴的愿景是相关的：阿里巴巴想成为数据分享的第一平台。这就真的要有这么大的一个地方做存储，要有那么大的计算能力，让用户有能力来处理大数据，还要保证安全。其实安全也是这次大赛我们比较紧张的地方：我们既要允许用户的代码跑上来，又要保障用户的数据安全，这是个非常大的挑战。

InfoQ：你们组是怎样分工的？

徐常亮：大致有三个方向：数据仓库场景，数据挖掘场景，编程框架场景。其中编程框架不仅是SDK，还会有一些重新定义，会引入一些新的框架。比如Hadoop上有Hive这样用SQL做的，还有Yahoo的Pig——完全是另外一种语言，还有现在很火的Spark，虽说是基于Scala，但数据处理那一层又是抽象了一层出来，提供了groupby、filter这些算子。我们也会提供类似的东西，或者让用户根据我们提供的基础编程能力来定义自己的框架。也可以说我们今后可能会自己再造一套分布式系统的处理语言，或者让用户来创造也有可能。

面向数据仓库的SQL，和数据挖掘有一个Xlab让用户能像写R或者Madlib、MATLAB一样建模，这些是基本算法的包装，都是用户可见的。还有很多模块，大家不一定看得见比如SQL的执行引擎怎么做，数据的存储怎么做。去年我们做了一个比较大的事情，我觉得跟飞天5K可以媲美：飞天5K是单集群5000台，但今天5000台当然也是不够的，你需要有多个5000台。ODPS就有一套系统能够管理多个集群，同时让用户觉得自己只是面对一个集群。这里涉及很多策略，决定你的计算到底在哪个集群上跑，数据在哪个集群的哪台机器上存放，是否在多个集群上都有存放，多个集群间数据的平衡复制怎么做等等。这个东西管的事情是比较多的，我们对外希望做到比较透明化。

InfoQ：你个人主要关注的方向是什么？

徐常亮：和计算相关的我都关注。比如数据仓库SQL这块，从解析到执行计划到执行引擎、存储，我都会看。这块是我直接负责。另外还有编程框架这块也是我这边会看的。这两块的同学直接汇报给我。另外，整体ODPS的架构怎么做、上面的控制集群怎么做的，我也会参与。

InfoQ：你们做过的这些东西当中，你觉得最值得跟我们分享的一件事是什么？

徐常亮：可能有一点是比较有难度的，就是怎么做开放。今天我们看Hadoop社区，因为它是开源的，大家对它各方面都有所了解，所以可以基于它的架构出很多新的东西。新东西都是有迹可循的，不是突然就冒出来的，Hadoop上的很多新东西都是基于单机时代的理论，比如数据库上的理论，这些东西都是有一定基础的，可能今天是有人把它们应用到了分布式环境下就成了新东西。

ODPS是开放数据处理服务，而开放不一定是开源，目前飞天和ODPS的代码都还没有公开的计划，即使现在公开出去你也用不了，因为要依赖很多配套设施。所以，在不开源的情况下做开放，这里面需要很好的平衡。

开放，意味着让用户自由、方便的使用我们的计算能力，充分挖掘数据的价值。对ODPS而言，要做到开放，让用户的想象力充分激发，取决于我们能把编程框架做得多漂亮。编程框架很重要。SQL、算法库这些可能更多面向BI的人员，他们可以拿相对现成的东西来用；开放数据处理服务在编程框架上做的事情更多是面向开发者，让他们根据我们开放的引擎、构造通过接口暴露出去，让他们能够用，又不至于把下面的运作模式都暴露出来，既要让用户有很高的定制权，又不违背我们的安全原则和我们对分布式和单机的平衡选择。

MapReduce就是一个很好的方式，因为有人给我们领过这条路，大家觉得这个方法比多线程处理锁的关系要容易很多，仿佛在写一个单机程序，只不过步骤不同。所以，我们提供的MapReduce可以照抄已有的东西。但是今后很多东西可能不是两个步骤就能处理完的，我们想用DAG——就是有向无环图，用比如现在的YARN或者MapReduce 2.0来支持这样的理念，像Hortonworks那个Tez框架就能支持一连串的、若干个task相继的关系，只要你不要成环，能描述依赖关系为有向无环图，我们都能把它分解出来，让用户在各个阶段做什么操作，这样来定制。这个东西我们会拿出来给用户用。当然对开发者来说，DAG就比MapReduce复杂一些，但它的处理能力和自由度更高。

我们还在想一些能帮用户做包装的东西，比如写一个wordcount：可能用户写一个MapReduce也很简单，但如果在SQL里面写就只要一个select和groupby就完成了，一句话就覆盖了wordcount的东西。所以，我们能不能给用户再包装一些语义？我们提供一个groupby的算子，用户就可以用。SQL虽然也被称为一门编程语言，但是它毕竟不像我们一般语言的逻辑，你可以写for循环，if之类的，控制能力很强，而SQL就感觉你只能表述一下自己想干什么，后面的细节很难控制，所以开发人员会感觉受局限。提供类似于SQL的基本算子——groupby、filter这种想法，在Spark里面也有类似的体现，我们可能也会做类似的事情。我们会考虑是否有一些东西能沉的更底层一些，或者有些东西可以拔高一些，以此来做一些设计或权衡。

当然这个思路可能有很多，我只是提出几个点，如MapReduce、DAG、结合SQL算子来提供高层功能，让用户跟写程序一样。我觉得写一个SQL可能还不是写程序，写程序还是有变量赋值、关系等更多操作。今后我也不知道会不会有别的，但这几个地方我们会下很大的力气，希望在整个大体系下做到安全并提供关键功能，在里面能做迭代、广播等MapReduce不提供的东西，让这些都能通过编程框架放出去，外面的人就能更好的控制分布式系统所具有的能力。如果真能做到的话，我觉得就能把开放做的很好。

InfoQ：所以从某种程度上而言，Hadoop下面出来这么多子项目，也是因为MapReduce的局限性？

徐常亮：某些方面是这样。你看Hadoop 2.0，或者说YARN调度器的出现，很大的原因就是Hadoop 1.0的job tracker只支持MapReduce和map only这两种简单的调度模型。在YARN上你就可以做MPI或者迭代等各方面事情，Spark也可以在YARN上跑，各方面的事情都相对容易。对ODPS而言，因为基于飞天，而飞天的调度——伏羲——从第一天开始就支持YARN今天能支持的模式。从这点也可以看到飞天的发展历程，一开始很多想法还是比Hadoop好的。

InfoQ：如果想提供这些比较丰富的模式，也是可以直接复制现成的子项目的吧？

徐常亮：这是一个做法。比如SQL，因为有标准的定义，我们就可以很容易的复制，只要你写出这个SQL，我的解析器就能按你想的那样解析它，你也想不出别的花招来。这方面已有的理论和体系都比较成熟了。但是Spark你拿出来，虽然这套东西我觉得也很不错，但是毕竟还没有像数据库理论那么定型，或者说自成体系，它有一些缺点。

我们是拿来主义，它好的地方我们就拿过来。Spark基于Scala写的，对于很多同学还是比较陌生的，如果我们把它移植成Java或者Python，这两个语言的社区更大，可能会更容易做。其实Spark这个东西在今天的ODPS上也可以跑起来，但我们这上面跑起来的Spark可能执行体系是完全不一样的。这块也是开放编程框架未来的一个方向，以后比如你可以把Pig也搬上来，都是有可能的。Spark有十几二十个算子，现在已经差不多都能在我们上面跑起来了。

今天我们做飞天也好，ODPS也好，我们做这些自主研发的东西并不意味着我们在闭门造车，我们一定会看外面好的东西，有些东西我们会结合我们自己的场景做整合或者微创新、创新。

InfoQ：对于ODPS，目前业务部门来提需求的多吗？

徐常亮：有一些业务部门的需求很明确，比如业务部门可能做一些数据分析，说我想更快，或者要处理更大的数据，以前支持TB级，现在可能要PB级。有些需求很明确，这些我们就想办法去解决，而且这些在分布式系统下，数据量变大本身就是线性扩展必须解决的问题，否则分布式系统就没有意义了。而处理速度更快这方面，我们也在做一些自己的探索，比如刚才我提到我们在里面做迭代很容易，有一些数据不落地，在实时化处理上，今天我们内部的SQL跑的速度非常快了，比Hive这些都要快。今后感兴趣的话我们可以公布一些benchmark的数据。

另外一些方面，比如编程接口，这些用户都是开发人员，他们的品味都会不一样，所以这就是为什么我们希望把底层包装好、放出来，让开发人员可以自己定制。这样每个人都会高兴。当然今天可能就是只能有一部分人高兴，毕竟把Java、C、PHP、Python的同学放在一起肯定是会意见不同的，我们希望还是把底层的算子、我们定义的一些东西拿出来，这样以后定制能力更高。如果每个人的需求都一个一个去搞，我觉得很难实现。

InfoQ：你觉得实现过的最有挑战的东西是什么？

徐常亮：我觉得那些学术性的、理论性的东西其实都解掉了，也看得到别人已经做好的产品，这方面没什么特别的难题。一路走来，我觉得还是工程问题居多。比如分布式系统里面本来是小概率的事件变成常态，而且因为不断地交互会放大，解决这些小概率事件变成了挺难的事情，因为这些问题往往在你的防范之外，你要怎么定位、解决，是非常有挑战的事情。

再一方面就是早期，不管飞天还是ODPS在人员配备上，人数和工作进度的压力都很大，有一些工程、项目管理上的问题。当然这个不是技术上的挑战了。挑战都是有的，但是一定会解决。

最常见的小概率事件就是设备坏掉。硬盘坏掉大家听到很多，另外网卡也会坏掉。虽然理论上盘古团队会处理硬盘坏掉的问题，但早期不管是调度还是存储都是坐在一起的，所以大家一起处理，更何况我们这儿有真实的场景，有大数据量，可以发现很多问题。

我们之前碰到一个网卡的问题：一台机器大概有千分之一的几率网卡坏了，它坏了又不是全坏，大概是万分之五的机会会把一个数据传错，一个bit会翻转——比如1变成0。总的来说是将近亿分之一的机会出一个错误。但是因为交互的数据量大，就给撞上了。

这个问题怎么发现的呢，刚才提到ODPS的几个特点，其实有一点很重要但是我没说，就是正确性。我们对正确性的要求很高，因为我们的第一个正式的商业客户是小微金服，就是阿里小贷。他们的业务关系到钱，直接关系到你能否把这个钱贷出去，所以我们要对他们的坏账率负责。在这个层面上，我们对准确性要求很高，所以每一次发布之前，我们都会做全批量的验证。这个过程我们需要比对各版本的数据，确保他们都是对的。这个过程，因为我们有数据做对比，所以发现有这么一个问题。这个用户都不一定能发现，他可能某一次跑发现某个数据不能解释，但是跑下一次又ok了，这个事情可能就过去了，因为亿分之一的几率几乎肯定不会再次发生在他头上，可能就换一个人。

发现问题后跟飞天的同学沟通，飞天网络层的同学可能会觉得是不是你们上层逻辑写错了，造成这种随机性，我们就要想办法证明我们上层逻辑没错。后来我们专门做了一个端到端的数据校验checksum。之前我们可能就是像HDFS那样对存储的数据做一个checksum，网络传输过程中做因为会带来一些额外的开销所以以前是没有全做的，但因为发生了这件事就不得不做了。所以我们必须对自己每一次的版本发布做一个很严谨的回归，有任何错误都不能放它过去。这也是我们的一大特色。

InfoQ：最后谈谈你对天池算法竞赛的期待吧？

徐常亮：ODPS是今年1月24日开始商用，开始邀请一些用户进来。我们希望在这次大赛开始的时候，也就是4月底，将整个ODPS正式对外商用。到时候我觉得外面的用户也会反馈很多，借助天池大赛也是看一看我们的竞赛选手对ODPS的反馈。

首先，我们毕竟是做平台出身，在用户体验方面可能做的不太好。我们在平台底层投入很大，但是对交互式的使用、API可能并不是定义的很好。这方面用户如果有反馈，对

我们来说是很大的帮助。商业化以后，我们要对外部的方方面面要投入更多。所以希望借着大赛做出相应的改进。

另外，我们这次提供给用户的东西还是比较多。1月我们只有SQL，4月我们会开放Xlib机器算法平台帮助用户建模，这个我们觉得还是很有威力的。去年我们内部做了一次大赛，类似这次的，得奖的前几名基本都用了这个超级武器，这个在今天同类产品里面基本上是没有的。我们也是希望借这次大赛把招牌打响，当然也是看看用户的反馈，让它不仅是威力很大，也要让用户的整个建模流程比较流畅。

另外，我们会把一些用户可自定义编程的东西放出来。当然我们也不希望一次开放太多，前期有MapReduce框架和结合SQL的udf，让用户可以自定义一些函数。这块我们也希望看一下用户的体验。这一块4月不会商用，但是会开放出来做一个测试，可能就是以大赛的用户为主。

最后，我们也在探索这个“数据分享第一平台”该怎么做。今天天猫把数据分享出来让大家建模，如果能达到很好的推荐效果，我们阿里巴巴也会受益很大。因为有几千支队伍，大家会有不同的想法，也许也会有新的东西。在我看来我们要做数据分享，就是要让大家能看到数据的价值。这就要看大家的想象力了。

查看原文：[阿里云ODPS的愿景、技术实现与难点]

京东的云化：相比IAAS，应用系统的SAAS化才是重点

本文是InfoQ中文站编辑杨赛、包研对京东集团技术副总裁何刚在2013年底的一次邮件采访内容，采访中对京东云的技术完成度、产品思路和发展过程中遇到的挑战进行了探讨。

受访者简介

何刚，现任京东集团技术副总裁，负责集团云计算和开放平台工作。曾任盛大集团副总裁，盛大云计算公司CEO，美国亚马逊公司AWS S3技术负责人，美国微软公司网上广告部门和SQL Server数据库部门首席研发经理等职务。何刚是中国电子学会云计算专家委员会委员，中国通信学会云计算和SAAS专家委员会委员，中国通信学会大数据专家委员会委员，中国数据中心产业发展联盟云计算服务专家委员会副主任委员。

背景

截止到2013年底，京东已陆续推出京东宙斯、京东云鼎、京东云擎、京东云汇、京东云峰等云计算解决方案，分别覆盖了API开放平台、IaaS基础设施、PaaS应用托管、代码托管/众包服务、移动应用后端服务等层面。

京东集团技术副总裁何刚在2013年8月的一次公开分享中表示，京东电商云的发展已经实现了第一阶段目标——完成京东IT资源的内部云化。在此基础上，京东电商云正走入第二阶段，将努力在2015年向产业链开放京东电商业务全部资源和能力。

InfoQ：之前提到京东IT资源的内部云化已经完成。这意思是不是说现在我们看到的京东，包括背后的交易、营销、物流、库存管理之类的系统，都已经运行在京东云上了？您觉得这第一阶段是最难的一步吗？能否介绍一下京东内部云化的实现流程，包括先改哪部分，后做哪部分，怎么做旧系统到新系统的切换，切换是否采用灰度部署等等。

何刚：我们内部的云化并不仅仅是简单的基础资源的云化，应用都跑在IaaS上。

京东的云化是电商能力和资源的云化，从信息系统角度来看就是京东七大业务信息系统的SOA化，应用系统的SaaS化。为了支撑这个，我们内部搭建了私有云平台，从PaaS和IaaS层面全面支撑应用系统的开发、运行和运维。

业务系统的SOA化我们已经搞了几年了，应该说已经基本实现了SOA化。但是，这是项长期工程，将持续进行下去。至于底层技术平台的云化，我们也是根据业务需求来进行的，不是为了云化而云化。

目前，我们云存储、中间件系统、自动化运维平台已经在内部广泛使用，云数据、虚拟化平台已经在支撑我们营销、交易、物流、金融等核心业务，但这个不是完全迁移上来，而是有需要的和新开发的系统才运行在云上。

InfoQ：各个京东云的服务是否共享底层基础架构？这个底层系统现在哪些基于OpenStack，哪些基于自研？

何刚：京东云主要包括公有云、私有云、金融集团云三大部分。物理上金融集团云完全独立；技术上三大部分共享IaaS和PaaS层。

现在底层系统，在IaaS层部分有使用OpenStack，但是基于京东优化版，做了大量的稳定性和规模扩展方面的改进，其中大部分系统是自研。

InfoQ：云擎是用CloudFoundry实现的，使用到现在也做了很多的改良。您对目前云擎在功能和性能上的表现是否满意？目前在云擎上的主要工作是什么？

何刚：基于CloudFoundry搭建的京东应用引擎（JAE），我们对其整体架构和解决方案比较满意。其原理是基于消息订阅/发布的消息通信的分布式系统，在支持组件横向扩展方面具有优势，使系统具有较强的伸缩性和较高的资源利用率。

目前JAE主要面向个人开发者和中小开发团队，支持多种Web应用开发语言和丰富的底层服务支持（云数据库、云存储、缓存等）。JAE从任何一个方向延伸出去都有较大的优化和扩展空间，可以做精做强，不过我们近期将力量集中在以下方面：

1. 更智能的组件弹性伸缩功能
2. 更友好的用户操作介面和流程
3. 更合理的资源分配策略

另外，考虑到组件的大并发处理和效率，我们规划将来对原Ruby语言实现的关键组件进行改写，用Go语言实现。

InfoQ：云鼎和云擎有很多服务是重合的，类似于Google的GCE和GAE之间的关系。我们是否允许开发者通过云擎对云鼎上的实例进行管理？

何刚：目前是不可以的。云擎提供的是应用开发PaaS平台，通过核心的应用引擎将应用代码进行编译、部署、运行和运维，用户看不到自己的应用具体部署在哪个实例上，所以不能像在云鼎里那样对实例进行全面自助的管理。但是云擎也提供了部分对实例资源管理的功能，如弹性扩展、云监控等。

InfoQ：我们看到目前京东云的几块服务，宙斯已经对外开放注册，云擎、云峰需要人工审核，云鼎仅针对企业开放，云汇、服务市场、设计师门户则要求提供详细的企业认证信息或个人认证信息及客服电话。为什么进行这样的差异设定？

何刚：从用户范围上来看确实存在一个漏斗关系。

最大的是云擎、云峰、云汇：所有的个人开发者、开发团队，甚至软件公司都可以在上面开发应用，而且这些应用是没有限制的，可以是to B的应用，也可以是to C的应用，应用类型也不限于电商相关，各种应用类型都可以。

第二个是宙斯，客户是需要调用JOS API接口的开发者、站长和ISV，其应用肯定是想要调用JOS接口，从事电商相关的业务。

再下来是云鼎，面向需要稳定、高级的应用运行环境的JOS用户。云鼎为需要更多京东开放数据的电商应用提供商提供服务，可以说是给JOS客户提供的VIP服务。

InfoQ：将投入产出比低的云鼎单独拆分出来，提高准入门槛，这个比较容易理解；将相对可控、不容易造成性能问题和安全问题的宙斯单独拆分出来，也比较容易理解；把云擎和云峰拆分运作，这个不太容易理解。当时决定从运营层面将这几块产品拆分，都考虑过哪些利弊？

何刚：我们将云擎、云峰单独作为两个云解决方案，分别提供了Web应用和移动客户端应用全流程的云服务，从代码，到编译打包，到部署测试，到运维管理，并配套基础、成熟的中间件服务。

之所以这样，是考虑到中国云计算尚处于起步阶段，大部分的开发者还不能像美国那样，用最基础的云产品就能搭建一个完整的应用解决方案。我们打包成云解决方案就是为了培养广大开发者使用云服务开发应用的能力和习惯，引导中国的开发者进入下一代的云端开发模式。

InfoQ：一个基础架构系统做起来，在规模上会有几个特别难过的坎儿，过去之后就有很长一段时间可以平稳发展，直到遇上下一个坎儿。京东云是否已经遇到过规模上的门槛？是如何解决的？

何刚：过规模的坎，网络流量显著变化和控制节点的挑战增加，是主要的。我们在解决这个问题时，60%的部分主要还是靠事前的充分预估和设计；20%的是靠各种从整体到单个子系统的规模上量的测试来保证；余下的部分靠上线后有问题处理问题的方式来面对。总体来说还算比较平稳。

InfoQ：云化带来的问题是如何对用户和数据进行隔离，业界的趋势是通过SDN进行更好的网络隔离。是否有计划引入OpenStack的Neutron，或者自研一套方案？

何刚：SDN已经在我们试验和开发集群，完成从核心到接入交换机的全方位测试和试验，能够较好的融合。

在私有云的生产环境中，我们刚开始上了一个小集群，提供离线计算类的服务。现在线上的EC2部分出于稳定考虑还是使用的传统网络。

查看原文：京东的云化：相比IaaS，应用系统的SaaS化才是重点

相关内容

1. 深度剖析京东云引擎核心技术
2. 京东李大学：技术、人与云生态
3. 京东电商云提供7大能力
4. Ruby在京东云擎中的使用
5. 京东电商云三步走战略：开发者是当前重点

2

文化篇

豆瓣的研发管理

本文基于InfoQ中文站社区编辑庄表伟跟豆瓣工程副总裁段念在2014年3月的一次沟通整理而成。在豆瓣的办公室里，双方探讨了如何给团队设置规则、如何传输价值观、如何恰到好处的设置激励策略、如何考核工程师等话题。本次拜访时，庄表伟正在华为2012实验室的研发能力中心工作，当时关注的重点是如何将开源社区的经验应用于企业内部实践。

受访者简介

段念，现任豆瓣网工程副总裁。80年代开始接触编程（BASIC），在个人兴趣的支持下进入软件行业，先后在华为、新太科技等企业任职，后加入Google中国。在软件开发、测试、软件团队管理等多个领域都有所涉及，近期关注的重点是团队建设，工程师文化推进。对于能够提升团队生产率和个人技能的方法感兴趣。

概况

豆瓣的研发管理理念建立在一系列约束与规则之上，其中约束是根本，规则基于约束去制定。我们的基本约束有三条：

- 最终的评价标准取决于对产品线的贡献
- 以正确的方式做事
- 要有技术目标

第一点也可以说是绩效认可原则，也就是什么样的绩效能够被认可。1000行代码不是绩效，带来了什么才是绩效。或者如果你没有对业务直接贡献，但提高了生产力，这个也算。

第二点主要是不接受低质量的代码。品味是好的工程师天然就有的。

第三点也可以说是我们要求成员有对代码的追求。如果我们招聘一个人，他如果只是把工作当成一项任务，对提升自己的技术这件事本身缺乏热情，那么哪怕他技术再好，我们也会拒绝。这样的人可能会仅仅关注绩效，而不关注如何更聪明、更有效率的做事。

基于此三点约束，我们制定了一些规则，这些规则又会衍生出其他规则，同时各个团队自己内部也会产生自己的规则。

比如，所有的代码都可以被review，这是一个总体的规则。这条规则需要工具的支持，这也是我们开发CODE的初衷。我们的代码review是一个相对自治的过程，不是从上到下，也不能算是从下到上。基本上，每个团队内部会形成一两个有代码洁癖的人，他们就成了发现低质量代码的人这样一个角色。

代码review在形成CODE这个平台之后，又衍生出了其他的规则，比如在merge代码前执行CI，还有创建分支的规则，提交代码的规则等等。当然，这也需要工具的支持。CODE作为平台，可以很好地容纳这些实现规则。

各个团队内部，如PM与工程师如何做分工，他们可以有各自不同的规则。有些团队则建立了20%的时间不做功能开发的规则，专门用这20%的时间做产生长期价值的开发，比如补技术债。有些团队则采取每次协商的方式来分配不可见的工作量。这些都来源于团队的技术追求。

总体而言，约束是不变的，规则是可以调整的。

规则如何制定

《管理3.0》这本书里说：好的管理者绝不是游戏规则制定者。我们倾向于让团队成员自己以民主的方式形成自己团队的规则，我尽量不插手。当然，在团队发展早期，你也可以尝试为它设置一些规则，但你会发现这些规则很快就会被团队内部产生的新规则所替代。

作为管理者，我们会忍不住像游戏一样去设计规则，但这是不可能的，我们也不应该这样做。我们应该去强调约束，定义规则的边界，确保规则跟约束没有冲突。

有些公司比较看重员工的一致性，尤其是思想上的一致性，统一的价值观，这点我是不认可的。我觉得统一思想这件事毫无意义。所谓共识，大致有三个层次：

- 愿景。就是“什么该做什么不该做”。比如往页面上放广告，这事儿该不该做，大家会有自己的看法。
- 价值观。就是“应该怎样做事”，在愿景之下，通过规则和约束体现出来。我觉得价值观不是贴在墙上的东西——越是贴在墙上反复念叨的，一般都是越没有的东西。
- 规则与约束。这是在行为层面。规则是一个很容易被复制的东西，比如开发流程，你看到大家都用pull request提交，你也很容易跟着这样做。在这个过程中，价值观得到了强化。

对于我来说，我更愿意大家在行为上一致，而不去追求思想上的一致。规则创建的过程应该是民主的，这个过程需要有冲突，有碰撞，有妥协——因为大家的思想并不一致；而规则一旦建立，则人人遵守规则。

如何激励跨团队协作与分享

最早豆瓣只有20多个人的时候，当时还没有分产品线，所有的任务都在一个池子里，公开列在Trac上，大家选择感兴趣的自己去做。当然，那时候也有一些约束，例如一个惯例是“自己维护自己的代码”。09年以后为了解决工程师的归属感，以及保持小团队快速响应，改成了产品线的方式。在产品线方式下，工程师的工作范围相对固定，而不像以前那样走一个公共的池子。

这样一来就弱化了工程师之间的横向联系，好的经验、体系、原则无法跨产品线共享；同时，工程师自己也被限制在了一个产品线之内，时间长了会觉得没意思。

所以在去年，我们用很大精力去推动跨团队的协作。一开始的做法是建立公共池，给个人成果更多展示的机会，但是没有特别好的效果。现在我们把注意力放在绩效规则上，让跨团队的贡献能够被豆瓣绩效体系识别，虽然也不能说就好了很多，但相比去年的尝试更加适合一些。

激励机制

我们对激励机制的使用非常谨慎。一方面你要表示自己看到了员工做的贡献、在意他的贡献，另一方面你又要避免激励过度而导致事情变质，把自发的行为变成了为激励去做一件事情。

去年，我们有个员工自发地清理了数据库中的无用数据，投入了很多业余时间，让数据库访问速度大大提升。那么，该不该给他发奖金？显然，这是一个很有价值的，应该鼓励的贡献，但立即的现金奖励并非是最好的方式。因为这种直接的现金奖励很可能会导致事情的动机发生变化。还有分享也是，我们也考虑过把分享做成一个积分体系，但我们非常在意这样会不会把一个内部驱动的事情变质成了外部驱动——难道没有积分奖励大家就不分享了吗？而且你设置了积分，有的任务积分多，有的任务积分少，又会导致“挑活儿”的问题。

激励这件事情要如何做的平衡？我觉得奖金、工资最好跟着绩效考核走，而不能针对具体某个事件来发奖金——会导致自发行为变质是一方面，另一方面也很容易不公平。对于员工自发做的好事，即时激励是应该的，但未必要用金钱。CODE的徽章系统就是一个不错的即时激励机制——徽章代表我看到了他的努力，同时也可以展示给别人看，可以有很好的传播，又不会对内部驱动力造成不良干扰。

评估制度主要解决如何评价工程师的问题。我们基本上没有设置特别具体的量化指标，主要是两个基本点：一个是面对面，跟tech lead面对面评估每一个工程师。另一个是公开，就是所有工程师的评估依据都是公开的。我们每半年做一次评估，每次3天时间，5、6个tech lead，大家一起讨论，甚至PK，各种理由一条一条的过。现在所有的评估我自己都需要参与，整个开发团队现在130多人，我基本上需要每个人都过，今后长期发展，我希望评估的流程能转变成委员会的形式。

不过，我认为绩效评估并不是考评最重要的部分。考评最重要的部分应该是目标设定与定期检查，这里面内容就多了，比如如何授权、如何进行目标选择等等。管理者管理的对象不是人，而是系统：对于团队这一个系统，你如何让团队认可大的目标和约束，如何让团队有能力形成自己的规则，让这些规则与目标调和，这是团队管理者应该关注的事情。对于人的管理，管理者最多只应该做到激励机制这个层面，再往下给人分配事情就不合适了。团队自己只要有了目标，就会自发的往前走，你不需要去关注团队具体是怎么去做的。

现在市面上很多管理学的理论，都有各自强调的点，比如KPI或者奖金，其实你会发现很多理论之间是冲突的，因为他们没有把公司整体纳入考量，而是只看到某一个层面，

一个部分，就着这一个部分衍生出一套管理理论，看起来很有道理，但真要实践起来，其中不少都仅仅是”看上去很美“而已。

最后推荐两本书：一本是我刚才提到的《管理3.0》，里面提供了很好的框架和管理者需要考量的维度。虽然这本书的作者为了贴合“复杂理论下的管理”这个“颠覆性”的主题，在提到不少经典管理理论中也存在的概念时，故意用一些不同的名字或是描述方式——从这一点上说作者有点“故弄玄虚”，但书仍然是好书。《管理3.0》提到的复杂性理论的知识可以从梅拉妮的《复杂》一书中找到，《复杂》这本书介绍了很多复杂领域的基本理论，比如细胞自动机、蚁群、混沌理论、非图灵机、生物信息工程等，对理解复杂系统有很大帮助。当然，要对复杂理论有更多了解的话，侯世达的书是不能错过的。

查看原文：豆瓣的研发管理

相关内容

- 对话豆瓣张克军：前端，工程与人
- 豆瓣网工程副总裁段念：文化建设是个“系统工程”
- 精于心，简于形——豆瓣FM产品设计
- 按需搭建灵活多变的 Web 应用 —— 以豆瓣阅读计划为例
- 豆瓣 CODE 两年历程回顾：git 不是万能的，没有 review 是万万不能的

360谭晓生谈技术管理与安全趋势

本文是InfoQ中文站编辑包研跟360副总裁、首席隐私官谭晓生在2013年底的一次采访实录。

受访者简介

谭晓生，男，1992年毕业于西安交通大学。曾任3721技术开发总监、雅虎中国技术开发总监、雅虎中国CTO、阿里巴巴-雅虎中国技术研发部总监、聚友网CTO兼任COO。现为北京奇虎科技有限公司任副总裁兼首席隐私官。

InfoQ: **欢迎360首席隐私官谭晓生先生接受InfoQ的采访。谭总，您可以先跟我们的读者打个招呼。**

谭晓生：InfoQ的朋友们，大家好，我是360的副总裁、首席隐私官谭晓生。

InfoQ: **好，谢谢谭总。我们今天的话题肯定离不开安全这个事。因为360在安全市场做了很多很多工作。但是我们看到，国内的安全市场和国外的成熟的环境还有很大的差距，包括用户对于隐私的认识，对于安全的投入，以及购买安全产品的意识，还远远没有达到国外的水平。那您觉得国内我们安全产业的这些企业，接下来这段时间应该主要做哪些方面的努力？**

谭晓生：国内的安全企业也蛮多的，有几千家传统的安全公司。但是，过去卖产品可能会倾向于企业市场，倾向于卖给一些合规用户，因为这些客户有等保和分保的要求。但这些设备买回去了之后是不是真正用起来了，很难讲。因为这种方式是以合规为目的销

售，产品本身做得是不是好用是存在比较大问题的。过去安全产品在易用性和有效性方面，我觉得国内的产品和国外的存在比较大的差距。但是今天，我们越来越多的信息是在互联网上，对企业来说，他的资产越来越多的数字化了，在这种面临着新型的安全威胁的情况下，用户对安全防范的敏感性提升，对安全防范的有效性提出更高的要求，易用性的问题也就暴露出来。用户如果真的想用你的软件或者设备，能不能真正的用起来？能不能做有效的防卫？比如，如果产品规则特别复杂，甚至在企业内部都找不到人进行规则配置，或者没有办法从告警中有效的找出真正的威胁是什么，安全防范就很难开展起来。我觉得这是现在传统的网络安全设备或技术所面临的比较大的挑战。

InfoQ: **传统的安全市场现在更加的开放，由于互联网的发展，用户的需求和对安全这个产品的功能挑战也比以前要多许多？**

谭晓生：我觉得安全挑战有几个方面。

第一，新型的威胁。来自于互联网的新型威胁，比如以信息窃取为目的的，或潜伏下来长期偷情报为目的的，他们有新的名字叫APT。因为APT攻击技术的进步，对于传统的安全防范的手段提出了很大的挑战：过去的单一、单点的防范，比如防火墙用IPS、IDS或UTM，是不是能有效的抵御这种新型的攻击，结果大家其实已经很清楚，传统的防范手段的有效性是比较成问题的。

第二，针对网络攻击的威胁，虽然也开始有新型的防范产品推出，但产品的成熟度还是有待提高的。网络安全是一个攻防互动的过程，一般来说首先是攻击的技术先领先，防范的技术跟得上，再去解决一定的问题，这是一种攻防的互动演进过程。在这个过程中，网络的安全专家毕竟是少数的，必然有一段时间，不管从政府还是企业，有很多机构暴露在网络攻击的危险之下。

第三，在今天这个时间点上，这些年我们培养出来的网络安全的人才还是远远不够的，我们现有的防范手段还是相对比较落后的。比如去年，我们的网站检测产品，检测了一百多万的网站，有65.5%的网站有安全漏洞，这个数字很能说明问题。首先，这个漏洞为什么会存在呢？扫描出来的漏洞是不是有能力去修补呢？这里面有很多问题，比如这个网站是用什么工具开发的，这个工具是不是有漏洞？这个网站是由哪个企业开发的，他的开发人员是不是具备了安全知识？面临的问题可能是找不到人去修复漏洞。

InfoQ: **刚才谭总提到，360已经发现了很多现有的企业或者网站中存在漏洞，至少我们比不知道存在漏洞已经前进了一步，下一步不仅仅是安全公司要做的事，因为需要企业内部开发、运维和安全公司要协同把这个事做好。我注意到360在去年已经推出了天机、天眼、天擎这样的安全引擎，结合这三个产品，能不能谈谈360接下来在企业，以及在个人和BYOD市场的具体的动作？**

谭晓生：企业市场的安全防范的防线是很长的。360只是在现在我们所擅长的几个领域做了几个点：

第一，在Windows终端安全管理方面，就是360天擎终端安全管理系统这个产品，过去在个人安全市场，我们有云查杀扫描病毒和木马的技术，有在用户本机的主动防御技术，有针对木马的检出技术等。我们给客户做了在企业应用环境下的适配，企业可能要求更强的可管理性，或者要求在隔离网络里有效的工作，在这些特性方面我们做了增强，于是就做了天擎终端安全管理系统，这个目的是解决Windows操作系统用户的防病毒、防木马、打补丁，管理员对所有终端的安全进行统一集中管控，软件分发，甚至包括一些基本的固定资产管理能力，这是天擎，是解决用户的Windows终端的安全问题的。

第二，天机产品的出发点是基于过去的这两年里，越来越多的企业员工开始用手机或者Pad办公，出现了比较严重的公私混用的问题，这些混用的机器可能已经Root了，可能装了一些游戏，游戏里不排除可能有木马等等，企业的数据假如也出现在这台机器里，是不是容易被盗，会不会成为一个攻击跳板等等。

很多人可能认为天机是360手机卫士的延伸，其实不完全是。BYOD的解决方案里有设备管理（MDM）、内容管理（MCM）和应用管理（MAM）三层，360手机卫士解决了设备管理的问题，而BYOD解决方案又解决了内容管理和应用管理问题，比如在企业应用里所有的数据在这台手机上存储的时候都是加密的，通讯链路也是用SLL加密的。对于应用来说，他有企业级的私有应用商店，只有经过企业的管理员认证，并且重新打包加固过后，这个应用才能够安装，并且还有能力侦测这台机器过去都安装了什么，提供这一系列的管理能力，其实天机是一个涵盖MDM、MCM和MAM这三个领域的产品，他和360手机卫士是一个互动的关系，天机可以调手机卫士的一些功能来实现设备管理的一些功能。除此之外，天机是面向位置威胁发现的，针对于APT攻击检测的，这个技术在全世界范围内还处于发展阶段，没有哪家公司敢说自己的技术已经很成熟了。天机、天眼包含了大数据的方法来分析企业网络是不是遇到了安全威胁，是不是有针对

于我的网站可注入攻击，我的企业网络里是不是有机器中了木马，是不是有远控，是不是有对外操作，通讯流量有什么异常。此外，在天眼中还集成了针对各种0 day攻击的检测，或者n day攻击的检测，是不是有一处攻击了，或者用户送过来的这个是不是非可执行性文件，是不是一个畸形构造的文件，是不是嵌了攻击代码等等一系列复杂的技术，天眼是面向于有APT攻击风险的机构的一种防卫手段。

天擎、天眼和天机这三个产品是一条防线，我们还给用户提供了针对网站的领域的云端防护：360网站安全检测和360网站卫士。目前在企业产品线，我们只做了这几个点。其他的安全防线产品，包括传统的防火墙，IPS、IDS这些依然有他存在的价值，比如SOC、下一代防火墙，在安全防线中也是不可或缺的，包括数据防泄露产品的市场也很大。安全需要纵深的联合防御，360只是在自己最擅长的领域做了几个点。

InfoQ: **谭总你刚才谈到国内目前安全市场的技术人才其实是比较欠缺的，包括您也提起过，360在校招时候，找到高水平的毕业生还是蛮难的，那你怎么看未来国内安全市场的人才的培养，因为其实就像您说的，各个安全公司都在招募人才，但这个市场的确是供不应求的，360接下来有没有一些其他的方法来更多的获取这样的人才？**

谭晓生：我们在做几个努力。

第一，包括与启明、绿盟的一些朋友一起，我们在共同在呼吁。从学界看，比如在新型人才的培养上面能不能做更多的事情，在课程的设置上是不是能够有一些更实用的课程出现，这些是在努力推动的。并且，我们这几家已经出手去主动的在学校里面去做宣传和推广，毕竟能够有比较好的天资来做网络安全的人是少数的，不是所有的人都能吃得了这碗饭的，需要有很强的天赋，我们现在做得就是让有天赋的人能够知道还有这么一个领域，在这个领域可能有比较好的发展，我们给他打开这么一扇窗子，这是我们首先第一步要做的事情。一旦有人走上这条路之后，我们在下面给大家提供一个更好的能够顺利发展的空间，但是最终我觉得既使这样做，我们也没办法在短时间之内解决安全人才匮乏的现状。针对于企业将来的信息安全的防卫手段，可能需要一些革新：我们没有办法指望每个需要防护的企业，都有足够多的网络安全人员帮他来做防卫，就得有一种商业模式，是不是有一些安全公司，真正的把SaaS，就是Security as a Service，把安全即服务这个概念真正的落到实处。在这种情况下，对于安全产品的要求也会变：过去可能期望一个产品我买回去了，装上了，加上电，规格配好，OK就起作用了。将来呢，这种情况可能不复存在了，安全产品可能仅仅是一个平台，最终还是会有更专业的

安全人员通过这个平台或云，不管是私有云还是公有云，对企业实施更加有效的安全防卫，这可能会是整个安全产业商业模式的大变化。

InfoQ: **你提到了安全跟云的结合，包括Security as a Service这种安全即服务的模式，其实360现在已经有类似的产品，只不过他还是一个点，没有形成一个面，或者说还没有形成一个平台，比如硬件防火墙，我们就没有。企业在安全投入是一个全面的投入，不可能只做一个点，未来360是不是有可能会打造这样的平台，或者加入这样的平台，来帮助用户能够通过一个点覆盖所有从软件到硬件，到云，有没有这种可能?**

谭晓生：这是我有多次演讲里面提到的一个概念，我觉得将来是一个立体防御，云加端加边界的立体防御，那么这个立体防御，这会是一个很长的产业链条。这个产业链条，我们认为最后应该是多家安全企业合做的，很难会有一家企业把中间的每个链条都做出来并且做好，所以这是不太现实的事情。

InfoQ: **刚才我们进到这个办公区的时候，我们看到墙上有一些法务的提示，一些安全的提示，作为一家安全公司，我们相信360肯定是在内部的安全审核、安全的培训做得比一般的公司要更加严格，更加合规，谭总能不能介绍一下在360内部，自身的安全是怎么做的? 比如一些规章、培训?**

谭晓生：在员工入职的时候，首先会有一个标准的入职培训，入职培训里面会有信息安全方面的一些培训，比如有哪些事情不能做，密码的强度，信息不许外传等等，但实际大家对此不要有太高的期望，人都是有弱点的，他会懒惰，各种各样的人性的弱点最后在信息安全上都会暴露出来的。我们的做法就是除了培训之外，要有很多的监督手段，比如密码，如果你设的弱密码被我们内部暴力破解，那二话不说你立马就得改，不改就要怎么处分你。那么对于信息的不当转存，转存到了不应该的介质或者网络存储上，我们也会有些探测，一旦发现了这种情况，就会立即给这个同事警告，要求他立即纠正。比如公司的网络资源的滥用，如果被监测到，也会有相应的警告。当然对于一个安全公司来说，本身安全防卫本身就会是一个很大的挑战。大家可以想像，如果把360的网站黑了，或者是360办公网突破了，那在行业里面恐怕是很容易让人扬名立万的事情，这方面我们比较谨慎，一方面360自己的安全产品，比如360天擎、天眼我们都会装，360肯定是第一个用户。除此之外，还是做了很多的防线，既使这样，也不是说万无一失，现在的网络世界的确是危机四伏，既使做安全的专业公司，在安全防卫上也是面临着很多的挑战和不可知的因素，只是说我们在这方面尽自己的能力，把安全安防的门槛垒得足够高，把防卫的墙尽可能垒得高。

InfoQ: **在国外有另外一种安全的这种体系，他们认为这种传统的按边界隔离的安全机制已经不能够适应现在互联网比较复杂的环境了，您怎么看待去边界化（de-perimeterization）？**

谭晓生：我觉得这不是去边界化的问题，而是有了一个边界之后，人们往往容易产生错觉，认为边界里面就是安全了，这就是很多隔离网络其实并不安全的原因。在隔离网络，他认为内部和外面的网络根本就不连，所以病毒之类的进不来，但实际真的是吗？肯定不是，我从来就不相信伊朗的核离心机是挂在互联网上，他一定是在隔离网络里的，但是最后他为什么会被攻击？其实，不管是U盘还是光盘，还是单向光闸，只要涉及到介质交换，涉及到文件交换，攻击代码都有可能藏在这些交换的文件里面，不管它是一个可执行程序，还是一个看似无害的不可执行的程序，一个文本文件，或者一个PDF文件，一个Excel文件，可能会利用了之前某个阅读器的0 day漏洞，恶意代码就进去了，那在实际的实践中，我们也看到，在很多的隔离网络里面是病毒木马横行的。首先各种安全软件安装本身就不是那么完全，这些软件更新更差，甚至可能是两三年前的软件还在那跑，不用最新的木马，老木马在里面都能活的很好，老漏洞，nday了几年的漏洞照样可以在里面这个去攻城略地，所以只要有一个边界，你就认为这个边界是安全的，那就大错特错了。真正的防卫，比如360网络对外肯定是围边界，但在这个边界之内，我并不认为边界之内是安全的，所以边界之内的探测一点也不会少。去边界化，我认为应该是这么理解，不是认为边界不重要，边界是重要的，但是边界内和边界外都需要有很高的安全防范级别，有一层边界，是会比没有边界好的，但是前提条件是你的边界之内，也要进行充分的安全防卫。

InfoQ: **刚才您已经谈到了，未来的安全市场应该是多家公司共同来打造更坚固的产业链，就您的观察，目前在一些成熟的国家和地区，比如说像美国，安全产品的渗透率大概有多少，主流的安全服务形式，比如Security as a Service这种形式，是不是已经成为了主流，能不能给我们介绍一下？**

谭晓生：是这样的，我手边没有美国具体的数字，只有些零星的数字，美国这些主流的安全市场碎片化比较厉害的，不是说有哪一家安全公司的产品一下子占了百分之好几十的市场。赛门铁克的市场占有率只有百分之十几，这已经是第一大的安全公司了。安全即服务这个概念，其实已经提了好几年了，但既使在美国我觉得它也没有进入主流市场，还是有那么多公司在卖各种各样的安全产品，不管是软的还是硬的，那么它里面会带有服务，只是说在西方，他对服务的重视程度比中国要更高一些，在中国靠安全服务可能养活自己的团队都会成问题。在现在这个时代来而言，我认为安全即服务可以解决

人才不够的问题，因为太依赖安全专家的技术技能，没有足够多的人的时候，又想要服务更多的人，那么用服务的方式，用云的方式作为一个平台，这是一个解决之道。当所有这些安全威胁来的时候，人们总是会和他的损失进行对价的，如果造成损失足够大，大家一定会要找到解决方法，市场最后会给一个解。其中一个解法我觉得就是安全即服务，不排除将来有人会发明出来新的解法，当有足够大的市场需求，没有足够多的供给，这时候就是共享，服务是最容易共享的方式，一个安全工程师可以同时为多家企业服务，每家企业的安全攻击的威胁被捕获到之后，即可被当作一种知识，其他企业都可以去进行这样的防御，这样就会出现一个基于云端的知识共享的防御体系。

InfoQ: **如果对云计算稍微关心一点，都知道美国有一个最大的云计算服务商亚马逊，很多创业公司包括一些大公司，很多已经开始在亚马逊上部署一些业务，不管是简单的存储一些数据，还是把一些相对来说比较重要业务放上去，这件事情在美国已经非常的流行，当然在国内可能这个国内的网络环境，国内各种这种法律环境还有各种的欠缺。但谭总您提到了，既使在美国这样成熟的互联网市场，成熟的云计算市场，大家已经非常明确的意识到，我要把这个非核心的业务要扔出去，交给第三方来做，交给像亚马逊这样的公司来帮我节省成本，既使在这样的环境下，那个安全市场仍然碎片化，这是不是因为安全服务或者安全产业，有这种特别的特殊地方：对信息的保护，包括用户的信任程度，获得用户的难度要比其他服务要获得用户难度要高的多？**

谭晓生：你是从这个角度来看。其实碎片化，我的理解可能会有几个原因： 第一，产品的适用性。很难在安全里面找到了一个产品一用就灵，一用90%的用户都灵，不是这样的。现在已有安全产品，包括防火墙，IPS，IDS，UTM，NGFW，SOAP，其实用户真正要的不是一个产品，一拿去这个企业安全就高枕无忧了，他需要自己的组合。某个公司可能就做好了其中的一两个，这方面他很擅长，比如做防火墙，谁家很好，那UTM又是另外一家很好，这是市场碎片化的一个因素。在产品的具体的类目上，包括类目可能有那么几家做得好的，还会有人在上面有点创新，比如他在UTM上扩展了一点，一招鲜，就能拿下来一部分市场。我觉得一个产品一旦成熟的时候，就意味着这个产品它的有效性开始受到置疑的时候，因为攻击方法总是有，这种新的方法出来了，又有人做出来针对新的攻击新的有效的产品，那产品在不断在演进。

第二，和本地服务还是有关系。中国的安全公司，除了北京有若干个公司之外，哪怕在成都，重庆，包括这些二三线城市，也会有当地的安全公司，他能够给客户提供某一种客户化的安全方案，并且能够提供本地的服务，这也是他们生存的空间，一个产品一出来适合所有人，这种模式在现在的安全防卫市场上可能还不太成立，它跟个人市场不一

样。个人市场杀毒做了这么多年，360的杀毒安全卫士做得好，百分之九十几的人就用了，但在企业的个性化的东西比较多，而且因为企业安全的环境变化，环境差异比较大，他对于去配置产品的人，对产品维护的人要求比较高，有点类似于专家服务，所以很难在这个世界上，见到一个做资讯顾问的公司把市场给垄断了。

InfoQ: **有安全圈子里的朋友曾经预测，移动的安全市场还远远没有成熟，还远远没有到来，换句话说，市场上还有很大的安全提升空间，或者说会因为一次或几次安全的漏洞的大爆发来引爆这个市场，您怎么看这种观点？**

谭晓生：我是赞同这个观点的，大家知道智能手机就是一台小电脑，而且是随时随地联网的小电脑，而且他们的漏洞是很多的，尤其安卓操作系统漏洞挺多。他安全防范的水平可能还未必比得上Windows下的安全防范水平，过去在上面病毒木马少，仅仅是说他原来的用户量少，作为一个攻击目标的经济价值不是足够大。但是现在的话，一个人甚至都不只一部手机了，攻击价值已经非常大了，而且这个是离个人信息、离钱更近，所以我觉得未来有可能会出现一两个，或者更多的大的事件会让移动终端安全引起更高的重视，移动终端的安全防卫手段目前比Windows上还是要弱的。

InfoQ: **我们看到360已经在移动安全方面做了很多工作，但要向普通用户收取服务费，让普通用户每个月在手机上花钱买安全的服务，目前来看还不是很现实的事情。如果说有这种大的安全事件爆发，会不会引爆这个市场，用户觉得我每个月去花一些钱去买些服务，获得一些安全的保证是值得的，有没有这种可能？**

谭晓生：我觉得可能性不大，在PC市场的杀毒被360免费之后，在手机上将来也没人能收的起来钱，一旦真的发生安全事件，我相信会有无数的安全厂商，希望立马帮用户解决问题，而且是免费的，现在求着你赶快用我的解决方案，那么他就是获得海量用户之后的商业价值，360通过自己的创造，几乎让所有互联网公司都能看得懂，这个模式之下，我觉得将来移动终端上的安全，个人市场也没有可能有收费的。

InfoQ: **换句话说，个人市场和企业市场是两个完全不同的市场。刚才已经提到，就是像国内有一些做公有云的公司，包括做私有云的公司，他们会通过SDN，或者VPN的手段，来帮助用户打造属于自己的相对封闭的云和安全的环境，那360有没有打算推出这样的服务？**

谭晓生：这个可能不会当作一个独立的服务推出，比如360天机的BYOD的解决方案里，我们内嵌了SSVP通讯，让用户的移动终端和服务器通讯的时候是在一个虚拟网络

内，但是没有专门给用户提供一个私有的VPN网络，这可能涉及到国内的一些管制，你这么做了，你是不是虚拟运营商，是不是有资质等。但是在一个产品里面，为了解决产品的特性，用到了这样的一些VPN的技术，这个是可以的。

InfoQ: **从您的观察来看，天机、天擎用户的反馈最满意的地方是什么？现在有一些创业公司在做类似BYOD相关的安全的平台，从用户的声音来看，和其他产品对比，用户认为360做得比较好的地方是什么？**

谭晓生：比如天擎，这个是用户量比较大的，我们这里已经有几十万的企业了，用户的反馈主要就是好用，其实就是遇到病毒能够检测得出来，能够杀的掉，遇到攻击能够拦得掉，就是这个特别简单的评判标准。还有一点是用户他有一个使用习惯，过去在家里面用360安全卫士，现在在企业里面用，他也会觉得很适宜，网管也省心。这是对天擎的反馈，主要是有效性和管理的方便性，给网管提供软件分发，提供固定资产管理，他们还是觉得这些特性是比较喜欢的。

对于天机，这个市场的启动速度比我们预期的要慢。在2013年的下半年，这个市场并没有很大的启动。目前能够看到的，像保险公司和航空公司，这些企业他把Pad当作生产工具，这些企业对BYOD很重视。在其他的商业公司，让员工通过手机来收文件，处理公文，还没有能引起很好的共鸣，这个市场的启动还有待观察。

InfoQ: **在去年360启动了“库带计划”，吸引社会上一些白帽子，一些安全高手，让他们贡献一些安全漏洞，现在这个计划进行的怎么样，2014年在这个项目上投入多少？**

谭晓生：在2013年，我觉得这是我们做得非常成功的一件事，“库带计划”在2013年大概收了两千多个漏洞，经过验证并明确是0 day，可以给白帽子奖励的，到12月份的数据已经到八百多了，给出钱的漏洞就这么多。在2014年，360的预算管理有自己的特点，比如先给一百万人民币，基本的策略就是不够了再加，就像去年在微博上一个口水战，有人说漏洞会多到我们的钱给不了，我就和赵武（“库带计划”负责人）说，你只管给，一百万不够，我再给你批一百万。我们对经费管理有这种灵活性，比如花钱买第三方建站软件的漏洞，我们给白帽子报酬是不会吝惜的，而且如果说这个价格随行就市，价格涨了，我们也会提高价格，这件事我们认为还是非常有价值的。

InfoQ: **换句话说，我们调动了社会各个方面的资源来帮助360来发现漏洞，帮助360发现漏洞也就是帮助360的客户去发现漏洞。**

谭晓生：其实最主要的是帮助第三方建站软件的开发者发现漏洞，帮他们发现漏洞很重要，因为他的一个建站软件可能服务几十万的网站，它的一个升级就意味着这几十万网站那几十万个漏洞都被补住了，所以这是从根上去解决问题的方法之一。虽然还不是最根源，但是已经比较接近问题的解决，我们会推动建站软件的开发商去修补他的漏洞。

InfoQ: **现在聊一聊360公司内部的文化和管理的话题。众所周知360和其他的互联网公司都是工作压力都比较大，但互联网圈子又是工程师文化，工程师是一个比较自由、自我团体，较大的工作压力与自由的文化会不会有些冲突，你怎么看？**

谭晓生：首先这两个不是对立的，但肯定会有冲突。比如说，有人我就是现在想玩儿，但是突然就有病毒木马上来了，必须得立马处理，可能要逼着他要去干事。那可能是家里孩子在等着他，但是又来了木马安全威胁，这就是你的职责，我们作为安全公司，你是一个安全从业者，遇到这种威胁的时候，你肯定要处理，就像去打仗，士兵上到战场，后方的有很多事，你不得不去做这个权衡，但这两个并不是绝对矛盾的。其实你想想，优秀的工程师有什么特点？我这么多年见过优秀的开发工程师，从来没见过哪个说不加班，或者不超时工作，他就能干的很好的，一个都没有。这个行业这种特点就是，他要在技术上能够做得好，首先得聪明，要有悟性。

第二，还要有足够的时间作为代价，这个世界上聪明人很多，适合思考计算机类的问题的人也很多，但优秀的程序员很少，就是因为有那么聪明的人他还得加上勤奋才能把这件事情做好，而这个勤奋的时候，有时候甚至和是不是公司安排工作没有关系。我以前在雅虎工作的时候，手下有一个架构师，他每天晚上干完正常的工作，可能都已经10点、11点了，但他还会再坐下来，差不多一直干到11点40、50的样子，去读各种各样的资料，因为大学毕业的时候，因为没有过四级导致没有拿到学位证。就是这位同学，最后他和我们美国的同事用英文交流做得很流利，这期间他就是利用业余时间学很多的东西，读英文资料等等，这种人最后才能会成为一个真正的技术上的高手。所以在360很多人并不是说他的老板逼着他去努力去干，他闲着没事的时候他也会去找点事去干，去做技术上的这个研究精进。有时候外部人可能觉得360很苦，但是在内部大家的感觉可能真的不见得是这样的，很多时候是有兴趣就做。我以前做程序员的时候，调试程序调到第二天早上太阳升起来是挺常见的，第二天早上太阳升起来那时候，其实身体非常难受的，但是当天晚上非要让我去睡觉，我那时候真睡不下去，就是这种特性。

InfoQ: **我们在招聘的时候，像你刚说这种员工，通过人力资源部门招聘有局限，人力资源有各种条条框框，刚才说的员工自发的这种学习的意愿和这种动力，往往不容易通过条条框框去框出来，您也曾经说过现在这种招人挺难的，这方面有什么心得吗？**

谭晓生：我觉得这是我比较擅长的。

第一，看兴趣。这是面试的重要性，因为用各种条条框框去框人的时候，你是没有办法知道这个人的个性是什么样的，他的特性是什么样的。但是和人面对面谈的时候就很清楚，比如我谈这个计算机问题的时候，有时候会谈一些很枯燥的原理性的问题，那么你看这个人他到底了解多少，因为如果是他对计算机没有兴趣，那些枯燥的东西他基本学不进去，但是这种东西是需要花时间去一点一点抠里面到底是怎么回事，当他把一个问题想明白的时候，你在里边讨论的时候，他的观点就会是不一样，这种就是对于计算机基础类问题的考察，就是我们判断这个人对计算机有没有兴趣的，没有兴趣，这个人其他条件再怎么满足都没戏。可能他今天干这个活，隔几天有一个别的什么兴趣就开始转了，但是这种对计算机是怎么工作的，计算机里面的一些比如算法感兴趣的人，你跟他聊下去，我能判断他有没有兴趣，有了兴趣，这是一点。

第二，神经兴奋程度。他是不是思维足够活跃，做事是不是足够快，这也是他能不能成为一个优秀程序员重要条件。有的人行动特别慢，如果像我们这种要求行动特别快的公司，那两边在一起不match也是很痛苦的。有人就是出慢活的，但是我们会找神经兴奋程度比较高的，比如什么呢？说话语速，你在和他谈一个问题的时候，他是不是身体会前倾，来和你争论，是不是脸红脖子粗。神经兴奋程度比较高的人，往往他的精力也会比较好，他一天可能保持比较长时间的神经兴奋，这些是在面试的过程中都能够筛选出来。

最后，性格的稳定性。因为有的人性格是飘忽不定的，这种人风险也很大，他有可能会成为一个很优秀的人，但是也有可能过两天他这个兴奋点一转。我们挑人时候，就是要怎么去找出来，有计算机悟性的，聪明的，对计算机还感兴趣的，神经兴奋程度还高的，定性还好的。如果找到了，恭喜，这个人很有可能就能成为比较优秀的程序员。

InfoQ: **我理解神经兴奋比较高和定性，就是情商，或者说跟情商有些关系？**

谭晓生：定性和情商有关，神经兴奋程度比较高更多还是生理特性。

InfoQ: **基本上筛掉90%以上的人了。不管怎么样，看了这段视频的工程师，如果想来360，基本上可以给自己画个框了，看看自己是不是在10%的这个目标人群里面。下来一个问题是关于这个工程师在研发过程当中，他们可能会用到哪些工具，比如在Google、Facebook这样的公司，工程师可能30%甚至更多的时间，首先要打造工具，通过工具来帮助自己这个研发和调试中提升效率，那在360内部有没有这样一些大家普遍应用的一些工具，能否介绍一下？**

谭晓生：这个还是蛮多的，但是不是广泛应用是另外一回事。比如我们做病毒木马分析的这些工程师，就会有他自己的工具，把可疑文件往这个窗口里面一拖，然后几个结果在一个平面展现了，工作人员就扫一眼就知道这个程序是干什么的，这个是病毒木马分析的工具。比如说像运维人员，他要管几万台服务器，他可能打开了Web的界面，又打开了什么地方有报警，我需要报修还是怎么样的，一点，这台机器重启，那台机器重装操作系统，这都是在过去的这几年时间里面逐渐形成的，这个都是为了改善自己的生活素质而做出来的东西，对于一流的工程师来说的确是，他们是有意识的来可以自己造工具的，能够让自己的工作从日常重复的这些工作里面一步一步的摆脱出来。在360除了自己造工具之外呢，我们最主要的贡献，还是用开源的工具，开源的系统，我们是大量的使用。从Hadoop，Cassandra，Storm到LVS，我们对LVS进行二次开发，大量采用OpenStack做的虚拟化的管理。

InfoQ: **有关于项目管理的问题，比如这个工程师可能比较活跃，很多创新，很多想法，但是他同时要面对项目要交付，代码提交的压力，我们内部有没有一种比较好的一种方式来管理，满足工程师创意的发挥，同时又能够保证这个项目按时的交付？**

谭晓生：我的观点可能有点不太一样，我个人过去是做过很重的项目管理，实际在360，我们用的非常灵活的开发方法。我们首先觉得一个优质的代码是工程师写出来的，不是管出来的。那么首先你的第一任务就是要找到足够优秀的工程师来干这件事情，然后干这件事情的时候，让他能够发挥他的长处，最快的速度把这件事情做出来。这里最重要的一点，就是要让这个工程师自己有ownership，好的工程师基本上会有特性，他是很好面子的，就是他自己如果是承诺了一个工期，他会不惜一切代价的去按期提交，所以这个任务首先是要他自己，认为是自己的任务，而不是老板安排给它的任务，这不是说一种管理方法。比如我们在做一个项目时候，经常就会把这个任务讲清楚，做好分解，然后就说兄弟们，觉得这件事多长时间能干完？谁愿意做那块？你要领了这块，你觉得这块你多长时间能做，为什么？为什么7天，不是3天，又为什么不是

10天？这个过程在这个任务在层层分解的过程中，大致的技术难度，每个人的实际的工作能力，大体的能够量化。

比如他说7天能做出来，那对于Leader来说就是评估7天是不是真的合理，是不是工作量大到7天做不出来，有可能要到10天，这时候你要提醒他，你为什么觉得7天能做完，你说说你第一天做什么，第二天为什么这样。分解下来之后，可能7天真做不完，有可能要到10天。也有可能他想到了一个很奇怪的做法，能很快的把这个东西搞定，这有很可能，这是个互动，这个互动过程之后，ownership就会形成，你可以认为，这时候领这个任务的工程师，他形成了他的一个承诺，他对外的一个承诺，对项目组的一个承诺。这时候去做，不是靠项目管理方法保证的，这是靠这个人的素质，你找的这个人的素质和他的责任心，这个团队最后形成的文化，如果大家都是我承诺了，我必须要做到这个文化，这时候不管是加班加点他都会做，而且人有压力的情况下，有时候会想出来很好的方法去解决问题。还有比如对文档和注释的问题，我大概在7年前，已经不对我的团队提写文档和写注释的要求，甚至我不反对我的程序员去重写别人以前的代码。只是条件是，对不起，这段我不记入你的工作内容，你要重写，愿意熬夜加班，把别人代码重写了，你看了就不舒服你就重写，绝大多数的情况下，你得到的新代码既不会比老代码要好很多，也不会比它要差很多，但最有可能的结果是这个人很happy的把这段代码写了，而且今后他会继续维护这段代码。

其实，程序是和每个人的思路有很大关系的，你非要让一个人A去遵循B的思路，他真的会很痛苦，但是如果他愿意，有时间，质量差不多的，把这个东西给写出来，用A和用B的代码有什么本质区别？想明白了这些事情之后，我当时的做法就是，第一不要求写文档，第二不要求写注释，第三，你爱重新写别人代码，你写，你加班写，没关系，我不管，就是这个就做到最后让大家很happy去干活。

InfoQ: **我明白了，管理就是让大家很happy。另外一个潜台词，还是回到刚才招人的问题上，关键还是找到合适的人，足够牛的人，这个是给领导省心，给管理省心，一切都省心。谭总，您刚才说到，在360做管理，包括之前在一些公司已经做了很长时间了，这么多年下来，你觉得自己最深的领悟是什么？有没有什么让自己印象深刻的一两件事情？**

谭晓生：这个就涉及的太多了，其实我觉得就是作为一个技术管理者来说，可能体会就是这几点：

第一，你要想把这事做好，你看你自己是不是对技术和对管理这两个跨界是不是你的乐趣。对我来说这是个乐趣，纯粹管理和纯粹技术对我来说都不会让我现在这样的干起来兴奋，我恰恰是就把技术和管理两个东西揉在一块的时候，在跨界的时候，我是特别兴奋。我觉得我自己的技术路线还不错，其实我已经有12年没有大量的写过代码，但这些年就是对技术的兴趣会让我保持不断的，对新的发展跟进，新出来一种东西，它的特性是什么，我可能会去看不少东西去比较。出来一些新的玩意，我也会去买来玩儿，比如一些硬件的东西之类，我也会去琢磨琢磨它是怎么做得，遇到一件事的时候有时候我会想它是怎么实现的，它给我的思想是什么，其实这22年下来之后，计算机的工作方法没产生什么变化，很多系统用最原始的理论推依然能推导出今天的结果，这些东西我觉得是把这个事给理顺了，理顺了之后，你有了这种思维方式就会很轻松。

第二，在技术管理其实永远不要忘了，最后你管的是人，这个观点可能既使在360也有人不一定会赞同，他会觉得技术怎么样，但其实我觉得所有最后做技术的是掌握了这个技术的人。那首先，就是你要找到足够聪明的人，筛选人比培养人重要的多，人不见得是都能教化的，教化成功的可能性很小，而且是要选到合适的人，选到合适人，大家一起来做事的时候，你就会发现这个选对了人，所有事情都可以简化的。而人都是有他自己的诉求的，他不管是个人发展的诉求，还是说是这个其他经济方面的，就是你想好，大家一起来做这件事情。然后就是投入产出之类合适，做的事是他感兴趣的事情，他为什么不做？他会很开心的去做这件事情，那在这里面大家就作为一个这个技术管理者的一个职责，我觉得就是把大家的利益最后能统一找好，兴趣点找好，然后给大家创造一个更宽松开心的这么一个工作环境，让大家最后做完这件事，即有成就感，经济上收益又能好，个人在这个行业里面我不管要名誉之类这些东西都有，你如果能够构建出来一个共赢的局面，这件事做起来就会很简单，团队也可以很开心的去做。

查看原文：[谭晓生谈技术管理与安全趋势]

小米首席架构师谈开源文化的建立

本文是InfoQ中文站社区编辑刘宇跟小米首席架构师崔宝秋在2014年QCon北京大会上的采访记录。在采访之前，崔宝秋在大会上就《拥抱开源：小米的经验分享》这个话题进行了公开分享。

受访者简介

崔宝秋是小米科技有限责任公司的首席架构师，小米服务器和云平台团队负责人，有十多年传统软件和互联网产品的开发经验。2000年获得美国纽约州立大学石溪分校计算机系博士学位。2000年至2006年历任IBM高级工程师和高级研发经理，从事数据库优化和内核总控等核心模块的工作。2006年至2010年任雅虎搜索技术（YST）核心团队主任工程师，参与了雅虎搜索引擎的热门搜索、查询优化和新一代查询缓存等重要项目的研发。2010年至2012年任LinkedIn主任工程师，开始接触社交网络，并负责LinkedIn搜索产品的研发，期间作为5个创始成员之一开源了SenseiDB，一个分布式实时搜索系统。2012加入小米科技有限责任公司，现负责小米服务器和云平台团队的工作。

InfoQ: **大家好，我是InfoQ的社区编辑刘宇，今天非常欢迎崔宝秋来到QCon，也非常荣幸能够请到崔宝秋来给我们做一个专访，现在请宝秋先简单做个自我介绍。**

崔宝秋：大家好，我叫崔宝秋，是小米公司的首席架构师，也是我们小米云平台团队的负责人，非常高兴跟大家交流。

InfoQ: **今天宝秋和我们分享了小米在开源路线上面的分享，我们觉得这个主题非常的不错，也受到了大家一致的好评。**

崔宝秋：谢谢，希望对大家有用。

InfoQ: **我们就想跟您就这个方面的问题再深入的聊一下，比方说现在目前大大小小的一些公司，在开源软件的选型上面是比较困惑的，我们应该怎么样去做一些快速的选型，因为毕竟小米公司在这方面的投入比我们要更多一些。**

崔宝秋：这是一个很好的问题，大家怎么选型，选什么样的软件，是决定自主开发呢，还是使用开源的，这个我觉得首先应该根据自己的业务需求，看有没有软件能够真正满足你的百分之百的需求，或者最起码百分之九十几的需求。你的需求，包括性能上的，规模上的，可水平扩展的，还是说可以掌控的程度等等，有很种因素，但主要是根据你的业务。第二就是你不管选什么，你要拿得下，要能驾驭，然后能真正掌控，不能说我随便搭个Demo、工作了就行了，将来线上有问题，压力上来了，有千万同时在线，或者几百万、上千万的日活跃，这种压力下，这个软件能不能够撑得起，有了事故以后你有没有能力、这个团队有没有能力能够解决问题，这些都是一个要考虑的因素。

InfoQ: **据刚才您分享的时候，有提到，在最开始在做HBase的时候，你下面的工程师团队大概投入了有多少人？**

崔宝秋：刚开始我们这个真正投入HBase也就三、四个人，现在专注HBase的也就五个人，当然HDFS、Hadoop这个生态系统里面我们人稍微多一些了。

InfoQ: **我记得最开始的时候，时间是非常的长的，我记得是有九个月的时间投入，对吧？**

崔宝秋：我现在回头看，应该是这么一个时间点，我们在2012年10月份成立了这么一个团队，我当时给了这个团队的一定的时间和空间来爬坡，学习、熟悉HBase，真正把它弄通，这个需要花一定时间。

InfoQ: **那对于我们其他的一些公司而言，可能没有像你开源的倡导者，或者提倡者，因此我们可能会面临着没有这么长的一个周期的一个学习期间，有可能老板会给我们的一定的时间的限制，就说我现在业务需求必须满足这个业务需求，我的时间会比较短，有可能是一个月，或两个月，并且人数可能也没有三个人到五个人，有可能就一个人，这个时候我们应该如何去面对？**

崔宝秋：不是说每个选型都需要花很长的时间的，有的人力不够，确实有的团队可能就一两个人，一个人要把这个项目搞定，这种情况下，有的项目不像HBase这么大，不会说一个项目在整个公司里面的战略地位有像HBase、Hadoop生态系统这么重要，有些只是一个小的KV系统，或小的工具型的选型，那么一个人花一个星期，快速搭建一个Demo，稍微比较一下性能，你要关心的是性能能否满足你的需求，比如说latency达到20毫秒这么一个，或者几毫秒这么一个的需求，如果能达到，这个小团队，或者一个人两个人，可以是老板本人，就快速搭一个Demo，用一些数据来说话，你可以举其他公司的例子，别人已经用过的，他们的性能怎么样的，在各种QPS下，两百QPS，或者几万QPS下，性能怎么样，他们使用场景是不是跟你们公司的使用场景接近，他们公司最坏情况下会达到一个什么样的性能，最重要的是，你要根据你自己的实际使用场景做些自己的测试，压力测试也好，各种配置，维护，等等的测试，自己拿一套数据来，列一张表，来比较，这个软件跟另外一个软件，两个开源的性能怎么样的，然后如果自己开发，投入是什么样的，有这么一张表的话，你就可拿这个表，和其他一些数据，来跟你的老板讲，为什么你认为选择这个开源软件是对的。

InfoQ: **或者更好的。**

崔宝秋：就是这么说服你老板，当然我希望每个公司都有这么一个有大的愿景，或者有长期计划的老板做决定是最理想的，如果从下往上推动，我希望小的公司团队或者小的管理者，你可以以自己的数据来说话，做足功课。

InfoQ: **其实我们从下往上去推动的时候，我们前期的这个准备工作是非常重要的?**

崔宝秋：有时候这种做功课可能会花点时间，但是长期来看，会为公司节省成本。

InfoQ: **那这点是非常好的，我们一定要放长远的去看。**

崔宝秋：应该有个长远的计划的。

InfoQ: **那么小米公司除了在追随开源脚步之外，自己也开放了很多开源软件?**

崔宝秋：一些，我们刚开始。

InfoQ: **在我看来有很多，就是从我个人角度来看，运维工具是很不错的，包括这些新的功能，其实是很赞的，那么这些其中会有您提到的，毫无保留的commit，因为我以前就是说比较了解雅虎的Apache Traffic Server，这是一个Cache软件，从它的**

功能性上来讲，他是有很多代码是在开源之前有所保留的，尽管他现在已经捐给了Apache社区，但直到现在，以前在雅虎内部使用的这些非常好的功能现在也是没有公布，没有被开源，已经在陆陆续续的被阿里的工程师在逐步的去重构去写。

崔宝秋：重新实现。

InfoQ: **重新实现，重新开源，这个对于整个开源社区来讲，或者生态链来讲是不是有些违背了初衷？**

崔宝秋：是有一些损失，有些可惜。

InfoQ: **如果在最开始开源出来，那么有可能打败了所有的这些Cache软件，因为当时真的，我使用至少有三年，那就是很早了，在那个时候Squid也好Varnish也好，等等，这还是非常的弱的，我是觉得挺可惜的，那么因此，现在小米正在走这样的一条路，那么因此就刚您提到这些毫无保留的commit，但是作为我们这些中型公司，或者小型公司来讲，我们有可能也会去开发这些工具。中国人有另外一个特性就是留一手，就说的不好听一点。我们应该怎么样去，用什么样的心态也好，或者用什么样的方式也好，去权衡这个关系或者说利弊？**

崔宝秋：这个也是大家都常问的问题，首先我讲很多公司开源，把一些自主研发的软件开源出去，很多情况下，他没有百分之百地开源，也就是所谓的留一手。这个是非常可以理解的，就像我们不能要求每家公司把自己的代码全公开一样的，因为每家公司都有自己的核心技术和一些核心代码，或者有些代码它就是没有必要去公开的，没有公共的属性，没有通用性，这种情况下，首先我们不能要求别人把自己的代码，家里一些什么东西都拿出去，也没必要。第二，他们肯定要有所保留，留一些有竞争优势的东西，比如我可能要先用起来了，稳定之后，我取得了领先地位，或者过了半年以后，我已经取得市场领先地位了，我再把之前的老版本开源出去。回到这个雅虎这个Traffic Server这个例子呢，当时他们决定Open Source，我也在雅虎知道一些，肯定有这方面原因，这是我的猜想，就是一些代码没有开源出去，一个原因应该是因为有这个核心代码了，或者说他们的一些代码根深蒂固，牵涉到其他的那些代码库，很多，也可能有这些原因。

InfoQ: **就是不太好梳理。**

崔宝秋：那些团队我觉得开源还是比较匆忙的，他们可能就会把一些公用的，或者比较好用一些东西先开源出去，这个我们应该去赞的，而不应该说你为什么不全部开源。

InfoQ: **是很赞的，因为给我们真的是带来很多用处，只是觉得有点惋惜，或者可惜。**

崔宝秋：回到我们小米的态度呢，我认为我们小米是希望能够更大尺度的开放，就是小米会更加并非常鼓励这个共享技术，这么一种态度，一种精神，我们有好的东西，我们不会藏着掩着，更愿跟人分享，比如HBase我们有一些新的功能，对我们非常有用，解决了我们实际的问题，性能提高了三点五倍，比如写性能提高三四倍，这种情况下，我们完全可以藏着不拿出去，但是我认为这有点不太合适，因为什么？我坚信小米在某种领域领先，如手机、路由器，小米电视、MIUI、云服务等等，我们不能全凭这个技术上的一点小的先进性来领先。技术领先，技术更加先进，会保证一个公司的领先地位，但是如果一个公司纯靠这么一些技术上的一点小创新，靠这个来领先，不敢跟人分享，这个公司长远看，我觉得是没有大的前途的。所以在这点上，我们有足够的自信，我们愿意把好东西跟大家分享，也就互联网服务这些，我可以做到很多情况下毫无保留，愿意跟人分享。

InfoQ: **小米真的是走在我们的前沿了。**

崔宝秋：我希望小米能走在前沿，靠其他的模式来创新，所谓小米模式，靠这些来创新，技术上我们是可以造福于自己，也造福于大家的，兄弟的创业公司。

InfoQ: **还有一个，可能也是有人比较关注的一个问题，作为一个开发者也好，平常在公司也好，那么我们有可能去使用某一个开源软件，但是在由于业务驱动的情况下，这时我们也有自己的粉丝了，因此我们会开发有一些新的Feature，但是在merge 回社区的时候，第一个可能就是不容易被采纳，当然就是你也提到了小米在HBase那块，总共提交了228个，那么其实总体上来讲，也是有一百多个被merge。**

崔宝秋：有些改动是需要时间的，不是说你每提交一个改动，马上给你merge，有些是在处理过程中。

InfoQ: **在处理过程中，持续性的，但是其实对于个人而言，我这里是提到一个个人，个人而言，他的兴趣和爱好，很多人还是热爱开源的，那么他有些时候是提交了，但是很可能很难被社区这样去被认可，因为毕竟个人的力量和公司这种团队的力量，包括领导的支持，它的力度是限的。**

崔宝秋：不可比的，很难比。

InfoQ: **我跟有一些这种开源社区的爱好者，包括开发者去沟通的时候，大家会觉得有点小小的失落感，毕竟这种开源软件是越来越多了，我们去Follow开源软件以后，还是我们自己去开放一个工具也好，很多人可能慢慢的会逐渐的与其不被他认可，我不如自己写个小工具，我可以放在Github上，我可以放在开源中国上，这样大家很多人评论去讨论，这样至少，我能得到一个安慰，鼓励着我，我继续向前进。**

崔宝秋：也可以，如果你的贡献没被一个大的开源社区接受，你可以Github给大家，你可以创建自己的一个代码库，或者自己创建一个一个新的分支，就把它放进去，如果有人对你的代码感兴趣，觉得有价值，自然有人，自然而然地向你要，来从你的分支上面再创立更多的分支，你的分支将来变成一个主分支了，变成trunk都有可能，只要你的贡献有价值。

InfoQ: **那么对于开发者而言，可能后续您更加推荐走这种方向？**

崔宝秋：这就是我刚才在分享中我也提到过了，确实，你讲的一个人以个人的力量打入社区，融入这个开源社区，让开源社区的人接纳你的Patch刚开始可能会是一个比较艰难的过程，首先别人为什么相信你？有点像单个的这些贡献者，很可能有成百上千，几十万，或者几百万的在这个社区里。他们怎么知道你不是一个散兵游勇，打一枪换一个地方，只是三分钟热度，最后你提交了代码也不维护，有问题你也不回答，别人改进的意见你也不理，你从此消失了？这种情况都有的。所以作为这个开源项目的负责人，他会非常谨慎地来评估这个人靠不靠谱。除非他觉得，一看改动非常大，性能非常好，一看就是一个高手、一个经过很多深思熟虑的一个技术牛人进来的提交的Patch。

大部分情况下，他还看不出来，他只能通过长时间的来跟你一些交流，直到你在圈里面混的，时间长了，混得脸熟了，这句话我老讲，混个脸熟，这个大家才取得相互信任，你才能进去，个人需要长时间的involvement，长时间的融入才可以达到你的目的，你想提交，你想贡献，你想把自己的代码让更多的人用，你的成就感，你想让这些代码跑在比如多个机器上，多个服务器上，你这种所谓热爱开源的成就感得以实现，最好的情况下是你背后有一个大的团队去支持你，这个团队往往来自一个比较有规模的公司，像硅谷那些大的互联网公司，他们在推动这上面比较有成效，也主要是因为这个原因，Facebook，雅虎，LinkedIn，Google这些大的公司，他们推动起来就比较快，像Cloudera，Hortonworks这些公司，他们有一帮committer进去，大家拧成一股绳，互相讨论一下，每一个Patch都有含金量的，他们永远站在技术的最前沿，方便一些，所以在国内我希望小米公司也能这样。当然我们每个个人，每个工程师要能力很强，另外一方面我从公司层面，我也希望把公司的长期的投入展示给开源社区，让他们知道我

们这个团队不仅力量强，而且有这么一个长期的规划，长期的投入，有与这个开源社区一起成长的一个愿景，这样会有帮助。

InfoQ: **以后小米有没有可能也会去推动国内这些开源爱好者这种个人，一起去构建这个生态链的方式?**

崔宝秋：这也是一个很好的建议，我们确实也讨论过，当然小米现在发展特别快，也刚起步，还是比较年轻的一个创业公司，很多方面还不是很完善，我希望在不久的将来，我们会有自己的小米的开发者大会等等之类的，或者说跟其他兄弟公司一起组建开源社区，让中国的开源爱好者一起有一个交流的平台，合作的平台，如果一个公司不容易打入国际的开源社区的话，那么几家公司联合起来是不是就可以达到？我认为我们中国工程师已经到了一个地步，我们有大量的优秀的工程师，在中国大陆绝对有能力组团，或者捆绑起来一些力量影响开源软件，为开源软件这个巨人指些方向，尤其在移动互联网领域，我们中国还是有希望。

InfoQ：我们也非常期望这天的到来，非常感谢宝秋能接受今天的采访。

查看原文：小米首席架构师谈开源文化的建立

中国顶尖技术团队访谈录

责任编辑 杨赛 | 发行人 霍泰稳 | 读者反馈 editors@cn.infoq.com

InfoQ 中文站新浪微博:http://weibo.com/infoqchina

商务合作:sales@cn.infoq.com 15810407783