

架构师

10月 ARCHITECT



特别专题

海量存储的使用与运维

大数据存取的选择:行存储还是列存储

云存储服务的可用性

Hadoop管理员的十个最佳实践

.....

分布式开发

Node.js之网游服务器实践

采访及书评--回顾手册

SoundCloud研发团队谈架构演变

阅读者(二十二) :从重构到模式

卷首语 | Editor's Letter

海量存储的使用与运维

当今的世界瞬息万变，反映在 IT 上则更是一日千里。移动、云、大数据等早已深入人心，本期《架构师》也在这个时代背景下应运而生。海量存储对于不少公司来说还处在摸索阶段，期间必定会遇到种种问题，而这些问题又在推动着更好的解决方案的涌现。对于海量存储来说，可用性如何保证、可伸缩性如何实现、访问速度又如何能做到更快、灾备实现是否完善等等，这都是大家不得不面对的现实问题。现在有越来越多的企业与公司在使用海量存储，那么在使用过程中会遇到什么问题、出现了哪些新的挑战、对运维人员提出了哪些新的要求、又有哪些最佳实践，这些问题都将在本期《架构师》中得到答案。作为一名技术人员，我们都深信技术改变世界的理念；时代在发展、技术在演变、人们的使用需求也随之发生着深刻变化，我们唯有适应这种变化、调整自己的步伐、增强自身实力方能在种种变化中立于不败之地。当你选择了技术这条路，注定需要我们付出艰辛的努力与汗水来浇灌，正所谓“活到老，学到老”，这句话我认为是对我们这些技术人的真实写照。但当你真正做到乐在其中时，那么在外人看来所谓的痛苦是不是本来就是一种快乐呢？

本期主编：张龙



促进软件开发领域知识与创新的传播



中文 | 英文 | 日文 | 葡文 |

目录

[卷首语]

海量存储的使用与运维.....	2
-----------------	---

[人物专访]

专访 SegmentFault 开发团队：垂直问答社区的架构升级.....	7
---------------------------------------	---

[热点新闻]

Eclipse Juno 版本的性能问题.....	14
---------------------------	----

进度与交付的那些事儿——持续沟通、持续反馈	17
-----------------------------	----

商业软件工程——产出会计和约束理论.....	21
------------------------	----

EMC 颜开分析 Google 全球级分布式数据库 Spanner.....	25
--	----

社区热议淘宝开源的优化定制 JVM 版本：Taobao JVM	31
---------------------------------------	----

[特别专题]

大数据存取的选择：行存储还是列存储？	37
--------------------------	----

云存储服务的可用性——从又拍网看云存储服务	42
-----------------------------	----

Hadoop 管理员的十个最佳实践.....	47
------------------------	----

[推荐文章]

分布式开发.....	57
------------	----

Node.js 之网游服务器实践.....	67
阅读者(二十二):从重构到模式.....	75
采访及书评--回顾手册	79
SoundCloud 研发团队 Sean Treadway 谈 SoundCloud 架构演变	85

[新品推荐]

WiX——强大的.NET 部署工具	100
即将来临的 Rails 4.0 将放弃 Ruby 1.8 支持，改进后台任务、缓存等多项内容.....	100
微软随.NET 4.5 发布新 REST API 框架.....	100
基于 Web 的代码编辑器 ACE，发布 1.0 版本.....	101
Embarcadero 更新 Delphi 和 C++ Builder，发布 HTML5 Builder.....	101
用 PostSharp 对.NET 做死锁检测	102
OpenXava 4.5 支持 JPA 继承映射和自动化业务逻辑	102
Google Cloud Messaging for Android (GCM)已推出，将取代 C2DM 框架.....	102
前 Sun 开发人员为 Android , iOS 等其他移动平台提供 JAVA 的 WORA 支持.....	103
Yeoman : 构建漂亮 Web 应用的工具和框架.....	103
HTML5 Boilerplate 4 : 改进了 Apache 配置和图片替换技术，并采用 MIT 许可证.....	104

[封面植物]

桃儿七.....	105
----------	-----

人物专访 | Interview

专访 SegmentFault 开发团队：垂直问答社区的架构升级

作者 [水羽哲](#)

[SegmentFault](#) 是一个面向国内 IT 行业开发者的问答社区，经过两个月的设计和开发 SegmentFault 团队于 8 月对网站进行了升级，不仅 UI 整体改版，交互流程和网站功能也有了很大的[不同](#)。InfoQ 就 SegmentFault 的这次升级对团队成员进行了采访。

InfoQ：SegmentFault 于近日进行了升级，能介绍一下升级的背景吗？

祁宁：如果说 SegmentFault 以前只是一个心血来潮的结果，那么现在已经算我们比较正式的事业了，毕竟各位参与者都放弃了手中现成的工作来到了杭州，一起共谋这个在他人眼里注定艰难的事业。这次升级其实也是这么一个背景，以前的网站只能算是一个个人作品，但是现在我们要把它变成一个产品了，这两者的差距我相信做过项目的人心里都清楚。相比较来说，技术架构上的升级都是顺其自然的，因为一切都已经有了这个前提。

InfoQ：当前的网站和之前在架构上有哪些不同？

祁宁：网站的框架已经全部重写了，层次更加分明，不过重构这东西对技术人员来说就跟家常便饭一样。要说真正的不同其实有几个方面。在框架上来说我们在 PHP 里引入了 Java 中注入变量的概念，虽然不是完全相同，但也已经到了神似。实现起来其实很简单，只是几个 PHP 的小技巧，但是却解决了 PHP 项目中一个让人头疼的问题，就是模块的自由引用。

另外一点升级就是我们对 [Redis](#) 使用更加成熟，如果说以前我们看 Redis 只是一个见猎心喜的小孩，什么数据都要一股脑的往里面放，现在我们就更加理智了，因为我们已经知道 Redis 什么能做什么不能做，使用的时候也更加有的放矢。

在数据层次上，可以看到我们已经统一了所有对象的 id，也就是说在 SegmentFault 中所有资源都有一个各自对应的完全独立的 id。这为以后功能的扩展提供了极大的方便。

高嘉峻：祁宁描述了技术架构上的变化，我说一下这次升级在产品架构上的变化。之前 SegmentFault 的产品和设计很大程度上借鉴了 [StackOverflow](#)，在过去一年左右的时间里，我们也在不断思考，什么样的产品模式才适合国内的技术环境。

这次改版，我们更强调了产品的社交属性，提供更多个性化功能，增强用户发现答案的可能性；另一个重要的变化，就是对搜索框的强调，我们希望搜索框可以成为 SegmentFault 最重要的入口，通过减少入口简化解决问题的流程。

可以说这次改版，我们摆脱了 StackOverflow 的影子，这是一个属于我们自己的 SegmentFault。

董锋：产品设计之声望与等级：声望单纯从数字上看，很难看出用户的声望值与提问或回答的价值关系，所以我们引入了等级概念。将用户声望按高低排名分成了 11 个等级，参见[这里](#)，这样问题和答案的价值就能通过作者的等级很直观的区分。

InfoQ：SegmentFault 希望能够为国内的程序员提供最有用的问答，那么在搜索上我们是如何做优化的？

祁宁：首先我们提供了实时搜索功能，当然这是 redis 实现的，实时搜索能让用户更快更准确的定位资源。

然后我们的主力搜索引擎使用了国人开发的 [xunsearch](#)，虽然它是基于 [Xapian](#) 的二次开发，但是封装后的 API 非常好用，而且也满足了我们的基本需求，它对中文的良好支持也是我们选择它的重要原因。

高嘉峻：之前说过我们希望强化搜索框的入口功能，所以这个产品对搜索要求会比较高。现阶段我们对搜索引擎的设计，更多考虑的是高扩展性，尤其是产品和功能上的扩展性。

早期 SegmentFault 的数据量还非常小，所以我们没有在搜索性能上做太多工作。功能上，我们希望随着网站的发展和运营，通过对用户行为的深入挖掘再扩展，我们使用 [xunsearch](#) 为问题、标签和用户建立了三份索引，也是出于易扩展方面的考虑。

高阳：新版网站搜索的改进很大程度上归功于 Redis 的使用，也可以通过这个[文档](#)来了解相关内容。

InfoQ：SegmentFault 的问题 URL 是无语义的，这一点和 Stackoverflow 不同，这是基于哪些考虑？

祁宁：是的，现在的 url 已经是由纯 id 组成。其实这也不难理解，Stackoverflow 或者国外网站用标题切分作为 URL 的一部分无疑就是为了 SEO，但是这一点英语上有天然优势，首先它每个单词是切分好的，再用中横线连接就可以自然形成一个个关键字，但汉字却不行，大多数标题连一个标点符号都没有，在 SEO 中的效果不会很好，其次，英文的字节长度要小于中文，可不要小看了这一点体积，因为我们的中文字节码都是要经过 urlencode，所以一个汉字的实际长度就很可观了。

因此对我们来说，与其在 url 这种细微末节上想伎俩，不如扎实的做好内功，做一个真正高质量的社区相信对搜索引擎的影响总比 URL 来的高吧。

InfoQ：我看到 SegmentFault 的邮件系统采用的 Amazon Simple Email Service(Amazon SES)，那在 SegmentFault 的整个系统中是否还采用了其他的 SaaS 服务？在集成这些服务的时候您觉得需要注意哪些问题？

祁宁：在选择 SaaS 服务上来说我们显得很谨慎，主要是国内这一点还太欠缺，没有什么可靠的选择，我们还不想当试验品，另一点就是选择过多的 SaaS 服务，特别是这些服务还来自不同的提供商，会增加系统本身的复杂度，让它难以维护，可靠性也会降低。

但是选择国外的服务也会有风险，比如有段时间 [SES](#) 就会因为众所周知的原因而导致网络无法访问，这时候就需要你用另外的服务器做中转了，这也是我们没有贸然选择的原因，这其中的坑实在比较多。

InfoQ：在整个 SegmentFault 的开发过程中您能分别列举最让人激动和纠结的两个技术实现吗？

高嘉峻：我个人比较兴奋的一个技术实现是我们这个 PHP 框架中一个 Data 模式，Data 模式加上以上 Joyqi 提到的注入。

我们解决这个问题，多场景要求不同的数据模型进行不同组合，而我们却不需要为这些场景分别定义数据结构，只需要在使用时直接调用，框架会在运行时注入数据。无论是开发效率还是运行效率都有提升。

董峰：在使用 [LESS](#) 还是 [Sass](#) 语法编写 CSS 纠结了很久。先用 LESS 写了一段时间，发现代码量控制不太好，与 HTML 结合过于紧密，有不少局限性，于是转投更成熟的 Sass，很好的将样式表模块化，提高复用率和效率。

我们还大胆了引入了响应式 web 设计（ responsive design ），使网页在不同分辨率的终端下都能良好的展示。

InfoQ : SegmentFault 社区当前讨论比较热烈的话题有哪些？SegmentFault 如何跟踪这些热点话题？

高嘉峻：首先我想解释一个问题，SegmentFault 提供的不是一个讨论平台，SegmentFault 是一个发现答案，解决问题的地方。

现在在 SegmentFault 平台提及最多的技术还是 Web 开发这个领域，如果说具体的技术上，PHP 和 MySQL 是最活跃的。我想这个现象与当下 SegmentFault 的发展阶段比较初级，会员还集中在 Web 开发领域有关。我认为随着 SegmentFault 的发展，用户组成会逐渐扩展到各个技术领域，趋于平衡。移动开发技术的问题会逐渐多起来，移动技术发展阶段比较初级，而且初入这个领域的人也会越来越多，需要 SegmentFault 的人在整个领域更集中。

我们正在构思一些数据挖掘类的产品，随着我们数据量的上升，对这些热点问题的跟踪和挖掘才会有意义。

InfoQ : SegmentFault 使用的 PHP 语言做开发，那么作为 PHP 语言的资深用户，据你的观察最近整个 PHP 社区都主要关注哪些热点问题？

祁宁：要说 PHP 的发展，确实还是比较让人振奋的。首先是 [Laruence](#) 成为了为数不多的 PHP 核心开发组一员，作为一个中国的 PHP 开发者，我感到由衷的高兴。另外一个就是 PHP 的开发者越来越务实了，最近看到开发语言排行榜，PHP 的排名下降了一些，其实这也是件好事，肯定很多人明白了 PHP 能干什么，不能干什么，这可以让这门语言的社区少一点浮躁。我希望的是 PHP 在擅长的领域做到极致，而不是什么都做个大概。

InfoQ：能和大家分享一下垂直型问答社区发展的探索、感悟，以及 SegmentFault 接下来 来的开发计划？

高嘉峻：其实不管做什么社区，最重要的是要有一个属于自己的性格。社区有性格，才能吸引志趣相投的人，留住这些人。我们最垂直的问答社区，就是要找到这个性格，踏踏实实做技术，做一个勤奋，聪明的技术宅，坚持展现这个性格，是我要做的最重要的事。

说到接下来的计划，我们仍然是坚持帮助用户发现答案，解决问题的原则，围绕用户个性开发一些产品。

另外，我们还发现很多朋友对 SegmentFault 的私有技术很感兴趣，我们会逐渐把这些技术完善，在合适的时候开源。

高阳：对于 SegmentFault 接下来的发展，我想在保证更好产品设计的前提下，接下来的重点会在社区运营方面加强。对于创业公司而言面临的是综合的竞争，除了你熟悉的产品和技术，运营能力在国内互联网环境中显得极其重要。国内的竞争环境更加复杂甚至恶劣，好的产品自动脱颖而出的可能性小，所以我们会转变思路在运营以及对外合作上面下一些功夫。我相信通过不断的运营可以发掘迭代出真正满足用户需求的产品。

原文链接：<http://www.infoq.com/cn/news/2012/09/interview-segmentfault-team>

QClub

我们影响有影响力的人

北京 上海 广州 大连 西安 太原 成都 杭州 武汉 南京 深圳...

QClub

邀请
业内知名专家

自由开放的
讨论氛围

定期举办的线下活动

结识
圈内技术好友

InfoQ



中文 | 英文 | 日文 | 葡文 |

热点新闻 | News

Eclipse Juno 版本的性能问题

作者 [Victor Grazi](#) 译者 [张卫滨](#)

在一场关于 Eclipse Juno 版本性能问题的[邮件讨论](#) (email thread) 中，Eclipse 的白银赞助商同时也是 Cloudsmith 的共同创始人 Thomas Hallgren 开启了一波对话。作为 [Eclipse b3 项目](#) 的活跃提交者，Hallgren 说将 4.2 版本切换回 3.8 版本后，“震惊于切换后的性能提升。3.8 版本的平台要快得多得多 (much MUCH faster)”。

Eclipse 缺陷管理工具 [Bugzilla 上的 385272 缺陷](#) (升级到 Juno 发布版后，响应非常慢) 条目上，充满了评论。按照一些回复的说法，当 4.2 刚刚启动的时候，一切安好。但是，随着它的运行，性能逐渐下降。一些用户报告说，重新启动 Eclipse 能够临时恢复到令人满意的性能水平。

Eclipse 的执行董事 Mike Milinkovich 在其名为 [“Eclipse 生活” \(Life at Eclipse \) 的博客中提到](#)，这并不是什么新鲜事，并将其归因为缺乏资源。“因为 Eclipse 有严重的资源问题，性能测试就被停掉了。这个问题的实际情况是 Eclipse 平台团队的扩张超出了他们按期正常交付的能力。至少在最近的三四年中，这个问题已经在很多论坛中讨论过了。遗憾的是，很少有人或组织来对此做出有意义的贡献。”

InfoQ 和 Milinkovich 了解到了更多的信息。

InfoQ：会有一个可预计的日期来修正 Juno 版本，还是我们要等待 Kepler 版本 (Eclipse 平台 4.3) 呢？

Milinkovich : Juno 团队的关注点在稳定性、功能以及兼容性上。在发布前，并没有关于性能方面的抱怨。

既然我们已经知道了问题，那团队会尽快解决。在未来几周的要发布的 SR1 中，会修正一种内存泄露的问题。在二月份发布的 SR2 版本中肯定会包含其它问题的修正。所以我们希望在 Kepler 版本之前及 Kepler 版本中会有明显的提升。

以下事实为我们的观点提供了支持，当我们 2004 年推出 Eclipse 3.0 的时候，实际上社区的反应比现在大得多。当你对像 Eclipse 这样广泛使用的平台进行大规模改造的时候，有些问题会很常见。

我们查看了所有发现的问题，并会尽可能多地处理它们。发现并解决这些问题的唯一途径就是实际使用 Juno、报告问题并对其进行审查。我们不会发布 3.9 版本，所有新的功能和性能提升都会在 4.2 和 4.3 的代码流中进行。

InfoQ : 你能介绍一下 Eclipse 开发中，Eclipse 基金会所扮演的角色吗？

Milinkovich : 我们的角色从来就不是领导研发，而是托管这些工程并保证 Eclipse 研发的顺利进行和工业生产过程（IP processes）。在 Eclipse 基金会中我们没有雇佣开发人员或架构师。满足用户和相关采纳者的需求是各个项目的责任。社区能够提供极大的帮助来提供良好的反馈以及可重用的测试用例和补丁。

一直以来，为 Eclipse 平台项目贡献功能相对来讲比较复杂。我们希望这能很快得到改观。我们的一些改进诸如切换到 Git、开始使用通用的构建设施（基于 Maven 和 Tycho）以及更易访问的 Eclipse 4 代码都是希望能够明显增强社区贡献的能力。

原文链接：<http://www.infoq.com/cn/news/2012/09/eclipse-juno-performance>

热点新闻 | News

进度与交付的那些事儿——持续沟通、持续反馈

作者 [王伟肠](#)

说起进度和交付，我想起一个和客户开发人员共同完成的项目。ThoughtWorks 为客户提供敏捷教练的服务，而带领大家完成一个项目，才是检验大家各种敏捷实践的试金石。在项目中，这个团队需按客户要求，在三个迭代的时间内交付一个符合用户期望的数据转换插件。参加项目的是已有 6 个月以上敏捷实践经验的团队。这样的“敏捷”团队在项目中的表现又如何呢？作为敏捷团队，每个迭代是不是都能按时交付用户想要的特性呢？

迭代一，铩羽而归

团队在第一个迭代中，埋头苦干，计划完成尽可能多的需求。把各种脏活累活抗在自己肩上，甚至出现了加班。然而，最终客户却并不满意。痛定思痛，大家总结了如下问题：

- 团队开始开发的时候，没有与客户确认需求，很多需求不清晰。这导致开发人员根据自己的假设做事。为了满足后期各种可能发生的需求变化，插件被做到可以配置，以致于最终的插件无比复杂。
- 团队在开发结束之后才给用户看成品，修改成本非常高。而从用户角度看，团队只有在项目开始的时候询问需求，在迭代结束时给用户看成品，其间没有任何交流。最后插件的功能和界面对用户来说，是莫大的“惊悚”，而并非“惊喜”。

有两个需求是互相冲突的。团队没有和用户确认，就任选一个做了，用户并不认可。

由上面的问题，我们发现，这个典型项目中发生的问题，并不仅仅该项目的问题，各个项目都存在。然而要避免或者减轻这些问题带来的后果，团队要做的就是沟通，找用户要到真实的需求，要到反馈，对工作方式和产品做出改进。

迭代二，硕果累累

团队在第二个迭代中，从做估算到做计划、再到开发、客户验收，全程和用户保持沟通。任何一个决定都要客户给出反馈。由于任一阶段，任一决策和设计都经过客户把关，最终插件自然也就和用户想要的八九不离十。团队成员们非常开心，他们又做了如下改进：

- 不熟悉插件的宿主软件的同事，要多学习，多去写一写。例如在开发之余，可以做些练习。等到正式开发时，就可以很快上手了。
- 除了和用户沟通功能上的需求，还需要和客户沟通质量上的需求（项目中的非功能性需求，例如性能。该项目需转换 Gigabytes 级数据，性能有一定的要求）在这些问题中，我们发现：
- 同事不熟悉插件的宿主软件，但是在第一迭代中，并没有人提出这个问题。然而到了第二个迭代，当团队交付顺利的时候，他们勇敢地提出要求抽时间学习，以提高效率。我们可以想到，任何项目，都会有同事对现有技术和框架不熟悉。那么我们是不是也可以为他们提供一些时间去上手呢？
- 团队对自己和客户之间的沟通有进一步的要求——非功能需求。这在项目的分析阶段是经常被忽略的。大多数情况是：对该满足非功能性需求的时候，没有满足。没有要求更高质量或者性能的地方，反而被做得非常强大。为了改进，团队觉得在平时的交流中加入此类问题的沟通。

迭代三，成员突变

迭代三中，由于部门经理希望团队间的知识可以分享，所以对不同组的成员进行了替换以达成知识传递。具体做法是：团队成员中只能留下一半“老成员”。在这个迭代中，我们

预期插件的交付速度会下降。而结果确实如此，参与替换的三个团队中，两个团队的速度下降非常明显。而另一个团队的速度却不降反升，这是为什么呢？

团队针对突来的“替换”，总结了如下问题：

- 有的团队成员之前没有参与太多交付工作，对技术和框架不熟悉。而有的团队成员非常强，一直在自己做开发，和用户交流，别的同事只有很少的机会参与开发，而长此以往，他们了解的项目知识越来越少，开发能力越来越差。

我们特别注意到这个差异化现象：有的团队成员对技术和框架不熟悉。这不应该出现在当前团队中，因为他们每人都至少有两年以上的开发经验。然而当我们了解到，有一个同事是项目组中的“超人”时，我们明白了其中的缘由。这个“超人”和其他“不熟悉技术”的同事以前是同一个团队的，而这个“超人”在项目中是项目经理兼技术经理的角色。这次替换暴露了这个问题。随后，他和他们团队的成员也意识到这个问题的严重性，回去制定了知识分享的计划。

到此为止，插件开发的项目就讲完了。我们再来看看其中的问题：

- 和客户沟通少，客户反馈少。（功能和非功能需求）
- 和客户沟通晚，反馈太晚导致修改成本高。
- 团队成员有时也不会对项目提出反馈。例如在压力状态下，没有去熟悉工具和框架。
- 团队成员之间有时也不会对问题提出反馈。例如替换后项目经理的知识传递问题。

而这一切都是关于沟通和反馈。我们肯定技术在进度和交付中的重要性，同时我们也应认识到：进度和交付不是一个独立的问题，而是一个系统问题。系统中的非技术因素对交付的影响可能是致命的，而沟通和反馈是保证交付的重要方法。

热点新闻 | News

商业软件工程——产出会计和约束理论

作者 [Michael Stal](#) 译者 [余纬邦](#)

Steve Tendon 在最近一篇博文“约束理论和软件工程”里解释了为什么在软件开发公司中产出会计要优于成本会计，同时他还给出了一个适用于软件工程的产出会计简单模型。

软件架构师除了负责架构设计之外，还需要关注商业因素。架构师在做架构决策时一定要考虑到商业因素，否则构造出来的系统可能没有很好的经济价值。Steve Tendon 是一名专注于软件工程管理的顾问，在这篇文章里他提议用产出来评估商业价值，而不是用成本。

为了阐述观点，Steve Tendon 在博文中引用了约束理论（Theory of Constraints，简称 TOC）里的“链子的强度是由最弱的连接点决定的”的观点：

TOC 认为所有业务都是将输入转变为输出的系统，输入经过一系列工作步骤处理最终转化为输出。输出是商业客户定价和付款的产品或服务。

TOC 的核心原则是：总会存在一个工作步骤限制了系统的输出能力。这一步骤被称为“能力约束资源”（Capacity Constrained Resource，简称 CCR）。Steve Tendon 认为 CCR 往往可以通过将过程流水化和细分任务来避免。在软件工程中，最小的任务单元可

以是 RUP 里的用户用例，XP 里的故事点或者是 Scrum 里的 backlog。优化 CCR 的方法是称之为“5 步聚焦法”的模型，如在维基百科中所提到的：

1. 指出系统约束（阻碍组织在单位时间内达成更多目标）
2. 决定如何利用系统约束（如何从系统约束中获益最多）
3. 所有一切配合上述的决定（整个组织要支持上述决定）
4. 升级该系统约束（为打破约束做必要的改变）
5. 注意！如果前面的步骤已打破了某个约束，直接回到第一步。不要让惯性产生新的系统约束。

介绍完 TOC 模型后，Tendon 用三个公式解释了产出会计的概念：

- 产出 $T = \text{收益} - \text{变动费用总和}$
- 净利润 $NP = \text{产出} - \text{运营费用}$
- 投入回报 $ROI = \text{净利润} / \text{投资额}$

如把这些概念用于软件工程的话，对应如下：

- 产出：指将工作中的代码发布到产品后的现金变现率，它通过将销售价格减去直接成本得到，直接成本包括：打包、发布、安装、培训、支持，以及网络系统。
- 投入：包括软件使用环境以及获取对客户有价值功能所消耗的金钱。包括软件开发工具以及需求采集。
- 营业费用：将概念需求转变为实际工作代码所涉及花费到的金钱。主要是软件工程师的人力成本，但也包括销售、综合成本和管理成本。

这个简单模型使得产出会计也能够被那些非会计专业出身的人（如 软件架构师）所理解。很多公司在提高商业绩效过程中常常会把重点放在减少成本上，但产出会计提出了另外一个方案，正如 Tendon 所提到的：

减少成本是有限度的，而提高产出则有可能是没有限度的。过度地减少成本会危害到交付的能力，从而反过来会引起产出降低等更严重的后果。

Tendon 在文章里列举了 3 个运用了产出会计方法的例子：

- 减少产出会计里的营业费用的一种办法是：在实现前砍掉需求。减少每个故事点都能够降低营业费用，从而提高净利润。
- 利用开源软件减少投入，虽然这样做有可能提高营业费用，但通常会比新构造一个同样系统所耗费的费用少。
- 当处于小众市场环境时，软件公司可以通过满足这个市场特有需求来提高报价。

就像 Tendon 在文章里解释的，产出会计不仅仅是可用于软件工程的会计模型，而且是可以彻底替代传统的专注于减少成本（如减少营业成本）成本会计。在文章的最后，Tendon 解释了产出会计在其他商业通用流程中（如周转、招聘、项目计划方面的约束、预算、资源和范围）所起的作用。最后是他对产出会计的总结：

产出会计可对所有商业流程进行管理决策，包括周转，招聘，外包，选择方法等。方法很简单，做任何决策都要考虑到产出、营业费用和投入三因素，这样的决策才是明智且财政稳健的。

一部分读者对这个博文发表了自己的看法，Robert McGinley 是一名支持 TOC 方法论的读者，他说：

“自从我读了《The Goal》一书后，我就成为了 TOC 的坚定拥护者。我认为它对系统架构师是非常有意义的，这篇文章作了很多很好的解释。

Dave Nicolette 喜欢这篇文章，但不同意对故事点的处理：

“我不同意将故事点与预估收入价值联系在一起。IME 故事点是基于成果的而预估价值完全不是，我更倾向于用“挣值”来跟踪顾客价值交付。它适用于传统和非传统开发模式。我见过有人用同样的方法来追踪范围和价值，其实是有问题的。你的看法可能不同。

Tendon 回复 Dave Nicolette 说道：

“是的，将故事点与收入价值联系在一起并不是最好的方法，用净值来衡量会更好，但从我所处理过的案例里来看，这样做已经足够接近真实值了，而且它可以为开发人员估计与管理决策所需要数字间的鸿沟铺平道路。也就是说：它是个可以达到理想结果的实用性的方法。它是基于平均来考虑的。从平均来看，一个故事点可以被平均地认为一个收益价值。而且如果你对工作清单进行严格分类的话，这个平均还能很好反映清单里剩余的工作量。

Christian Beck 喜欢这篇文章但认为约束并不一定总是坏的：

“约束并不一定是总是坏的。在 CDS（信用违约掉期）中，约束很大程度地决定了交付的范围（这在软件领域里是很难定义和衡量），而且还涉及了其他重要的交付指标如质量，结果是否新颖。甚至，约束还是系统设计的一部分（如在制品数量限制，时间限制等），变化的约束是影响（而不是控制）结果的重要工具。

Tendon 回复 Christian Beck 道：

“是的，约束必须结合着你自己的目标来看。一些约束实际上会帮你往正确的好的方向发展；而有些却相反。最好是能够分辨它们，而且也要知道怎样利用这些好或不好的约束来让你自己的目标受益。

原文链接：<http://www.infoq.com/cn/news/2012/09/tendon-toc-ta>

相关内容：

- [Oracle 和苹果遭受 Java 安全问题困扰](#)
- [Windows Identity Foundation 已包含在.NET 4.5 中](#)

热点新闻 | News

EMC 颜开分析 Google 全球级分布式数据库 Spanner

作者 [郑柯](#)

完成对 [Google Dremel 原理的分析](#)后，EMC 中国研究院的研究员 [颜开](#) 又在自己博客上 [分析了 Google 的全球级分布式数据库 Spanner](#)，他重点分析了 Spanner 的背景、设计和并发控制。

在简介中，颜开指出：

“Spanner 的扩展性达到了令人咋舌的全球级，可以扩展到数百万的机器，数以百计的数据中心，上万亿的行。更给力的是，除了夸张的扩展性之外，他还能同时通过同步复制和多版本来满足外部一致性，可用性也是很好的。冲破 CAP 的枷锁，在三者之间完美平衡。”

接下来，他提到：

“Spanner 能做到这些，离不开一个用 GPS 和原子钟实现的时间 API。这个 API 能将数据中心之间的时间同步精确到 10ms 以内。因此有几个给力的功能：无锁读事务，原子 schema 修改，读历史数据无 block。”

在背景分析部分，颜开认为：Spanner 在 Google 的定位，处于 F1 和 GFS 之间。

颜开这样介绍 F1：

一个可容错可扩展的 RDBMS——F1。和一般的分布式数据库不同，F1 对应 RDMS 应有的功能，毫不妥协。起初 F1 是基于 Mysql 的，不过会逐渐迁移到 Spannerr。

F1 有如下特点：

- 7×24 高可用。哪怕某一个数据中心停止运转，仍然可用。
- 可以同时提供强一致性和弱一致。
- 可扩展
- 支持 SQL
- 事务提交延迟 50-100ms，读延迟 5-10ms，高吞吐

至于为什么 Google 不用 BigTable，颜开的分析是：

因为 BigTable 提供的最终一致性，一些需要事务级别的应用无法使用。同时 BigTable 还是 NoSql，而大量的应用场景需要有关系模型。就像现在大量的互联网企业都使用 Mysql 而不愿意使用 HBase，因此 Google 才有这个可扩展数据库的 F1。而 Spanner 就是 F1 的至关重要的底层存储技术。

颜开又介绍了第二代 GFS——Colossus。

初代 GFS 是为批处理设计的。对于大文件很友好，吞吐量很大，但是延迟较高。所以使用的系统不得不对 GFS 做各种优化，才能获得良好的性能。

Colossus 是第二代 GFS。Colossus 是 Google 重要的基础设施，因为他可以满足主流应用对 FS 的要求。Colossus 的重要改进有：

1. 优雅 Master 容错处理 (不再有 2s 的停止服务时间)
2. Chunk 大小只有 1MB (对小文件很友好)
3. Master 可以存储更多的 Metadata(当 Chunk 从 64MB 变为 1MB 后，Metadata 会扩大 64 倍，但是 Google 也解决了)

Colossus 可以自动分区 Metadata。使用 Reed-Solomon 算法来复制，可以将原先的 3 份减小到 1.5 份，提高写的性能，降低延迟。

至于 Spanner 相对于 BigTable 和 Megastore 的优势，颜开认为：

“BigTable 在 Google 得到了广泛的使用，但是它不能提供较为复杂的 Schema，还有在跨数据中心环境下的强一致性。Megastore 有类 RDBMS 的数据模型，同时也支持同步复制，但是它的吞吐量太差，不能适应应用要求。Spanner 不再是类似 BigTable 的版本化 key-value 存储，而是一个“临时多版本”的数据库。何为“临时多版本”，数据是存储在一个版本化的关系表里面，存储的时间数据会根据其提交的时间打上时间戳，应用可以访问到较老的版本，另外老的版本也会被垃圾回收掉。”

Google 官方认为 Spanner 是下一代 BigTable，也是 Megastore 的继任者。

颜开提到 Spanner 作为一个全球化分布式系统的有趣特性：

- 应用可以细粒度的指定数据分布的位置。精确的指定数据离用户有多远，可以有效的控制读延迟(读延迟取决于最近的拷贝)。指定数据拷贝之间有多远，可以控制写的延迟(写延迟取决于最远的拷贝)。还要数据的复制份数，可以控制数据的可靠性和读性能。(多写几份，可以抵御更大的事故)
- Spanner 还有两个一般分布式数据库不具备的特性：读写的外部一致性，基于时间戳的全局的读一致。这两个特性可以让 Spanner 支持一致的备份，一致的 MapReduce，还有原子的 Schema 修改。

这些特性都得益于 Spanner 的全球时间同步机制：

全球时间同步机制，可以在数据提交的时候给出一个时间戳。因为时间是系列化的，所以才有外部一致性。这个很容易理解，如果有两个提交，一个在 T1, 一个在 T2。那有更晚的时间戳那个提交是正确的。

这个全球时间同步机制是用一个具有 GPS 和原子钟的 TrueTime API 提供了。这个 TrueTime API 能够将不同数据中心的时间偏差缩短在 10ms 内。这个 API 可以提供一个精确的时间，同时给出误差范围。

颜开认为：

“这个 TrueTime API 非常有意义，如果能单独开源这部分的话，很多数据库如 MongoDB 都可以从中受益。”

颜开接下来分析了 Spanner 的体系结构：

Spanner 由于是全球化的，所以有两个其他分布式数据库没有的概念。

- Universe。一个 Spanner 部署实例称之为一个 Universe。目前全世界有 3 个。一个开发，一个测试，一个线上。因为一个 Universe 就能覆盖全球，不需要多个。

Zones. 每个 Zone 相当于一个数据中心，一个 Zone 内部物理上必须在一起。而一个数据中心可能有多个 Zone。可以在运行时添加移除 Zone。一个 Zone 可以理解为一个 BigTable 部署实例

其中还包括如下组件：

- Universemaster: 监控这个 universe 里 zone 级别的状态信息
- Placement driver : 提供跨区数据迁移时管理功能
- Zonemaster : 相当于 BigTable 的 Master。管理 Spanserver 上的数据。
- Location proxy : 存储数据的 Location 信息。客户端要先访问他才知道数据在那个 Spanserver 上。
- Spanserver : 相当于 BigTable 的 ThunkServer。用于存储数据。

颜开分析了 Spanner 中的抽象概念 directory , 它让 Spanner 比 Bigtable 具备更强扩展性 :

Directory 是一些 key-value 的集合 , 一个 directory 里面的 key 有一样的前缀。更妥当的叫法是 bucketing 。 Directory 是应用控制数据位置的最小单元 , 可以通过谨慎的选择 Key 的前缀来控制。 Directory 还是记录地理位置的最小单元。

对于 Spanner 的数据模型 , 颜开的分析是 :

在设计之初 , Spanner 就决心有以下的特性 :

- 支持类似关系数据库的 schema
- Query 语句
- 支持广义上的事务

.....

据模型是建立在 directory 和 key-value 模型的抽象之上的。一个应用可以在一个 universe 中建立一个或多个 database , 在每个 database 中建立任意的 table 。 Table 看起来就像关系型数据库的表。有行 , 有列 , 还有版本。 Query 语句看起来是多了一些扩展的 SQL 语句。

Spanner 的数据模型也不是纯正的关系模型 , 每一行都必须有一列或多列组件。看起来还是 Key-value 。主键组成 Key, 其他的列是 Value 。但这样的设计对应用也是很有裨益的 , 应用可以通过主键来定位到某一行。

对于 Spanner 的并发控制 , 颜开提到 :

Spanner 使用 TrueTime 来控制并发 , 实现外部一致性。支持以下几种事务。

- 读写事务
- 只读事务
- 快照读，客户端提供时间戳
- 快照读，客户端提供时间范围

那么何时能出现开源的 Spanner 和 F1 产品呢？颜开的分析是：

GFS 出现于 2001 年，Hadoop 出生是在 2007 年。如果 Hadoop 是世界领先水平的话，GFS 比世界领先水平还领先了 6 年。同样的 Spanner 出生大概是 2009 年，现在我们看到了论文，估计 Spanner 在 Google 已经很完善，同时 Google 内部已经有更先进的替代技术在酝酿了。笔者预测，最早在 2015 年才会出现 Spanner 和 F1 的山寨开源产品。

颜开博客上的原文对 Spanner 有更多深入的细节分析，如果读者想了解更多，请移步[颜开的这篇博客](#)

原文链接：<http://www.infoq.com/cn/news/2012/09/emc-yankay-analyzes-spanner>

相关内容：

- [JS Pattern 及 HTML5 Boilerplate 开发实践](#)
- [Embarcadero 更新 Delphi 和 C++ Builder，发布 HTML5 Builder](#)

热点新闻 | News

社区热议淘宝开源的优化定制 JVM 版本：Taobao JVM

作者 郑柯

9月18日，淘宝核心系统部专用计算组的王峥（花名：长仁）在微博上宣布：

jvm.taobao.org 上线，开源基于 OpenJDK vm 的优化定制 JVM 版本：TaobaoJVM

在 jvm.taobao.org 上，介绍了项目的背景：

淘宝有几万台 Java 应用服务器，上千名 Java 工程师、及上百个 Java 应用。为此，核心系统研发部专用计算组的工作之一是专注于 OpenJDK 的优化及定制，根据业务、应用特点及开发者需要，提供稳定，高效和深度定制的 JVM 版本：TaobaoJVM。

TaobaoJVM 基于 OpenJDK HotSpot VM，是国内第一个优化、定制且开源的服务器版 Java 虚拟机。目前已经在淘宝、天猫上线，全部替换了 Oracle 官方 JVM 版本，在性能，功能上都初步体现了它的价值。

专用计算组在淘宝的职责是：

- 针对特定领域问题，以计算性能、效能为导向的优化。
- 异构计算推广及实践。
- JVM 优化、定制及相关工具开发。JVM 相关故障，问题排查及解决。
- 协助优化特定应用。

目前他们在这个项目上正在做的工作包括：

- JVM 优化及定制

- 持续为阿里集团提供优化、定制 JVM 版本。
- 线上 JVM 相关故障排查，问题解决。
- 反向图像搜索引擎 iflake 优化
 - CPU 算法优化。
 - GPU 应用实践。
- ETao 淘一淘系统优化
 - 逻辑回归最优解过程 CPU 算法优化。
 - GPU 应用实践。
- Hadoop 优化
 - 性能角度优化 Hadoop。
 - 压缩卡应用预研。
 - Namenode 性能优化。

现在，他们已经提交的 JVM Patch 主要分为一下三大类：

- 性能优化（针对淘宝 x86 平台的专用优化）
- 定制（根据淘宝业务需求）
- Bug 修复（已贡献给 OpenJDK 社区，官方版也有）

该计算组的成员包括：王琤（花名：长仁）、莫枢（花名：撒迦）、莫简豪（花名：坤谷）、孙宇（花名：洪熙）、费辉（花名：成滔）、孔建钢（花名：群旋）、梅路峣（花名：云达）、高洋（花名：望舒）、李临川（花名：谦正）。

这条微博发出后，引起技术社区强烈兴趣，到目前位置，已经有 256 条转发，88 条评论。

莫简豪在评论中提到：这个项目是

内部日常沉淀的结果，非为开源而做，非为核高基而做。解决实际问题中慢慢沉淀下来的
在技术上，elathen 表示：

不错，根据自己的需求定制化了。GC-Invisible Heap/GC-Invisible Heap Shared Memory 对那些长期缓存和不想被 GC、也不希望 GC 在这一部分 Heap 消耗资源的人很有用

[雪中飞](#) 的提醒是：

jvm.taobao.org 是基于 OpenJDK (GPL-licensed , sun 发布) 上做的改进和整合！推广程度要看这个 jvm 能给用户带来差异化的体验。这么底层的技术，选择一定要慎重！

[@bluedavy](#) 说：

强烈的顶，做 Java 在全球淘宝应该都是难得的环境，这里能碰到太多在其他环境中无法碰到的问题，战斗力绝对是疯狂增长，Taobao JVM 在排查故障上提供给很多帮助，例如程序中有分配大数组时自动输出警告日志和堆栈、PrintGCReason 等，值得大家使用

[@Fennng](#) 也表示赞扬：

非常赞！感谢淘宝技术人对开源社区的贡献！

来自思科的[张毅](#) [WeiBo](#) 在微博中指出：淘宝的贡献已经得到了国际社区的认可。

最近跟国外几个资深工程师交流，惊讶于他们对 taobao 和 taobao 贡献开源社区的了解程度，每每谈起无不是赞许有加，开源世界的根本价值观--talk is cheap, show me the code. taobao 真的做到了。

[丐别](#) 更是认为：

淘宝从技术上应该超越 redhat 了，开源贡献步伐也很快

不过对开源的贡献并不一定仅限于技术，[mulder](#) 就说：

其实对开源社区的贡献不一定是那种有一定技术含量的，中国的 ruby 程序员最感谢淘宝的是它为 ruby gems 提供了一个镜像 ruby.taobao.org

当然，也有人产生了一点小误会，[莫枢](#)在微博中提到：

请大家不要把这个组和做阿里云 OS 的 VM 的组弄混了。

淘宝确实对开源社区贡献良多，大家可以查看 InfoQ 上更多关于“[淘宝开源](#)”的内容。同时，我们也呼吁国内其他技术公司更加重视开源，重视开发者，重视开发者社区。

原文链接：<http://www.infoq.com/cn/news/2012/09/taobao-jvm>

开发者大赛 首届

—— 云 计 算 成 就 代 码 之 美 ——

如果你是 **开发者**，你热爱 云计算；

如果你是 **创业者**，你热爱 **云计算**；

那么请与阿里云一起漫步云端，共创云端的精彩！

只要用你手中的 “**代码**” 谱写出令人 “心动”的 **工具**

你就有机会 **获得** 总额 **百万大奖**

有可能 **获得** 宝贵的 **风投机会**

阿里云成就你的梦想！

活动时间

官方网站

官方微博

2012.7-2012.10

2012.aliyun.com

@阿里云开发者社区

特别专题 | Topic

目前有越来越多的企业与公司在使用海量存储，但海量存储对于众多公司来说还是一个新的话题，也许并没有现成的资料供参考，很多时候都是靠我们自己不断摸索、试错、解决、完善。在这样一种大背景下，对于海量存储来说，在使用过程中会遇到什么问题、出现了哪些新的挑战、对运维人员提出了哪些新的要求、又有哪些最佳实践，这些问题都将在本期《架构师》中得到答案。其中来自又拍网的云存储实践、Hadoop 管理员的十个最佳实践以及大数据存取的选择：行存储还是列存储？从不同角度深刻剖析了云存储的使用方式、运维等，对于广大读者来说是不可多得的学习资源，希望你能喜欢本期《架构师》。

特别专题 | Topic

大数据存取的选择：行存储还是列存储？

作者 [袁萌](#)

上个月参加了一个云存储的技术讨论会。这一个月里，陆续收到几位同学讨论大数据保存和处理的邮件。今天是周末，索性把这个月的交流内容整理写下来，供各位参考。

目前大数据存储有两种方案可供选择：行存储和列存储。业界对两种存储方案有很多争持，集中焦点是：谁能够更有效地处理海量数据，且兼顾安全、可靠、完整性。从目前发展情况看，关系数据库已经不适应这种巨大的存储量和计算要求，基本是淘汰出局。在已知的几种大数据处理软件中，Hadoop 的 HBase 采用列存储，MongoDB 是文档型的行存储，Lexst 是二进制型的行存储。在这里，我不讨论这些软件的技术和优缺点，只围绕机械磁盘的物理特质，分析行存储和列存储的存储特点，以及由此产生的一些问题和解决办法。

一 . 结构布局

行存储数据排列

	Column 1	Column 2	Column 3	Column 4	Column 5
Row 1	Data 1-1	Data 1-2	Data 1-3	Data 1-4	Data 1-5
Row 2	Data 2-1	Data 2-2	Data 2-3	Data 2-4	Data 2-5
Row 3	Data 3-1	Data 3-2	Data 3-3	Data 3-4	Data 3-5

列存储数据排列

	Row 1	Row 2	Row 3
Column 1	Data 1-1	Data 2-1	Data 3-1
Column 2	Data 1-2	Data 2-2	Data 3-2
Column 3	Data 1-3	Data 2-3	Data 3-3
Column 4	Data 1-4	Data 2-4	Data 3-4
Column 5	Data 1-5	Data 2-5	Data 3-5

表格的灰色背景部分表示行列结构，白色背景部分表示数据的物理分布，两种存储的数据都是从上至下，从左向右的排列。行是列的组合，行存储以一行记录为单位，列存储以列数据集合单位，或称列族（column family）。行存储的读写过程是一致的，都是从第一列开始，到最后一列结束。列存储的读取是列数据集中的一段或者全部数据，写入时，一行记录被拆分为多列，每一列数据追加到对应列的末尾处。

二. 对比

从上面表格可以看出，行存储的写入是一次完成。如果这种写入建立在操作系统的文件系统上，可以保证写入过程的成功或者失败，数据的完整性因此可以确定。列存储由于需要把一行记录拆分成单列保存，写入次数明显比行存储多，再加上磁头需要在盘片上移动和定位花费的时间，实际时间消耗会更大。所以，行存储在写入上占有很大的优势。

还有数据修改，这实际也是一次写入过程。不同的是，数据修改是对磁盘上的记录做删除标记。行存储是在指定位置写入一次，列存储是将磁盘定位到多个列上分别写入，这个过程仍是行存储的列数倍。所以，数据修改也是以行存储占优。数据读取时，行存储通常将一行数据完全读出，如果只需要其中几列数据的情况，就会存在冗余列，出于缩短处理时间

的考量，消除冗余列的过程通常是在内存中进行的。列存储每次读取的数据是集合的一段或者全部，如果读取多列时，就需要移动磁头，再次定位到下一列的位置继续读取。再谈两种存储的数据分布。由于列存储的每一列数据类型是同质的，不存在二义性问题。比如说某列数据类型为整型（int），那么它的数据集合一定是整型数据。这种情况使数据解析变得十分容易。相比之下，行存储则要复杂得多，因为在一行记录中保存了多种类型的数据，数据解析需要在多种数据类型之间频繁转换，这个操作很消耗CPU，增加了解析的时间。所以，列存储的解析过程更有利于分析大数据。

三. 优化

显而易见，两种存储格式都有各自的优缺点：行存储的写入是一次性完成，消耗的时间比列存储少，并且能够保证数据的完整性，缺点是数据读取过程中会产生冗余数据，如果只有少量数据，此影响可以忽略；数量大可能会影响到数据的处理效率。列存储在写入效率、保证数据完整性上都不如行存储，它的优势是在读取过程，不会产生冗余数据，这对数据完整性要求不高的大数据处理领域，比如互联网，犹为重要。

改进集中在两方面：行存储读取过程中避免产生冗余数据，列存储提高读写效率。

如何改进它们的缺点，并保证优点呢？

行存储的改进：减少冗余数据首先是用户在定义数据时避免冗余列的产生；其次是优化数据存储记录结构，保证从磁盘读出的数据进入内存后，能够被快速分解，消除冗余列。要知道，目前市场上即使最低端CPU和内存的速度也比机械磁盘快上100-1000倍。如果用上高端的硬件配置，这个处理过程还要更快。

列存储的两点改进：1.在计算机上安装多块硬盘，以多线程并行的方式读写它们。多块硬盘并行工作可以减少磁盘读写竞用，这种方式对提高处理效率优势十分明显。缺点是需要更多的硬盘，这会增加投入成本，在大规模数据处理应用中是不小的数目，运营商需要认真考虑这个问题。2.对写过程中的数据完整性问题，可考虑在写入过程中加入类似关系数据库的“回滚”机制，当某一列发生写入失败时，此前写入的数据全部失效，同时加入散列码校验，进一步保证数据完整性。

这两种存储方案还有一个共同改进的地方：频繁的小量的数据写入对磁盘影响很大，更好的解决办法是将数据在内存中暂时保存并整理，达到一定数量后，一次性写入磁盘，这样消耗时间更少一些。目前机械磁盘的写入速度在 20M-50M/秒之间，能够以批量的方式写入磁盘，效果也是不错的。

四. 总结

两种存储格式各自的特性都决定了它们不可能是完美的解决方案。如果首要考虑是数据的完整性和可靠性，那么行存储是不二选择，列存储只有在增加磁盘并改进软件设计后才能接近这样的目标。如果以保存数据为主，行存储的写入性能比列存储高很多。在需要频繁读取单列集合数据的应用中，列存储是最合适的。如果每次读取多列，两个方案可酌情选择：采用行存储时，设计中应考虑减少或避免冗余列；若采用列存储方案，为保证读写入效率，每列数据尽可能分别保存到不同的磁盘上，多个线程并行读写各自的数据，这样避免了磁盘竞用的同时也提高了处理效率。无论选择哪种方案，将同内容数据聚凑在一起都是必须的，这是减少磁头在磁盘上的移动，提高数据读取时间的有效办法。

	行存储	列存储
优点	写入效率高，提供数据完整性保证	读取过程没有冗余，适合数据定长的大数据计算
缺点	数据读取有冗余现象，影响计算速度	缺乏数据完整性保证，写入效率低
改进	优化的存储格式，保证能够在内存快速删除冗余数据	多磁盘多线程并行写入/读（需要增加运营成本和修改软件）
应用环境	商业领域、互联网	互联网

关于作者：

袁萌，现就职于国际商用机器（IBM）中国有限公司，主要从事大规模数据产品的设计/开发工作,存储数据高可用性以及数据生命周期管理，积攒了大量的设计及工作经验。专注于金融、电信、制造等大型数据中心存储架构设计。对业界主流的云储存产品以及技术有着深刻的认识。

原文链接：<http://www.infoq.com/cn/articles/bigdata-store-choose>

特别专题 | Topic

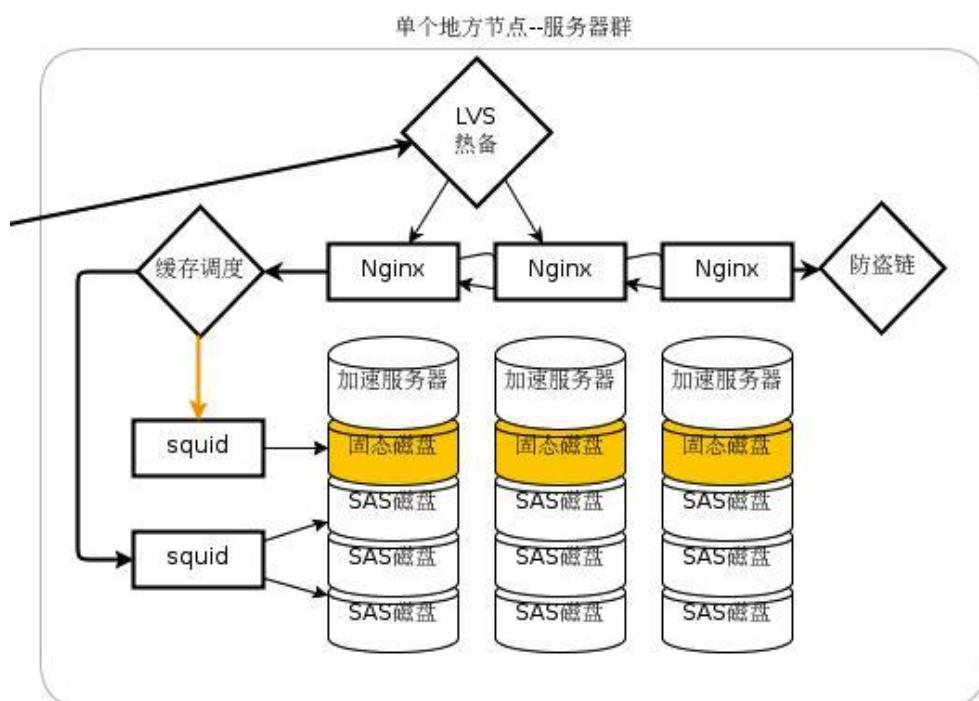
云存储服务的可用性——从又拍网看云存储服务

作者 沈志华

“云”这个概念在今年非常的火热，2年前国内的云存储服务还只有又拍云存储一家，如今国内已不下十家，面对如此多的云服务商，选择云服务的标准成了大家比较关注的问题。我们很荣幸在 InfoQ 与大家交流一些心得。

我们在六年的云服务经验基础上沉淀了三个词：安全稳定、快速、易用。

一、安全稳定



云服务的安全隐患大致会出现在两个方面：第一是服务的持续可用；第二是数据的丢失和泄漏。

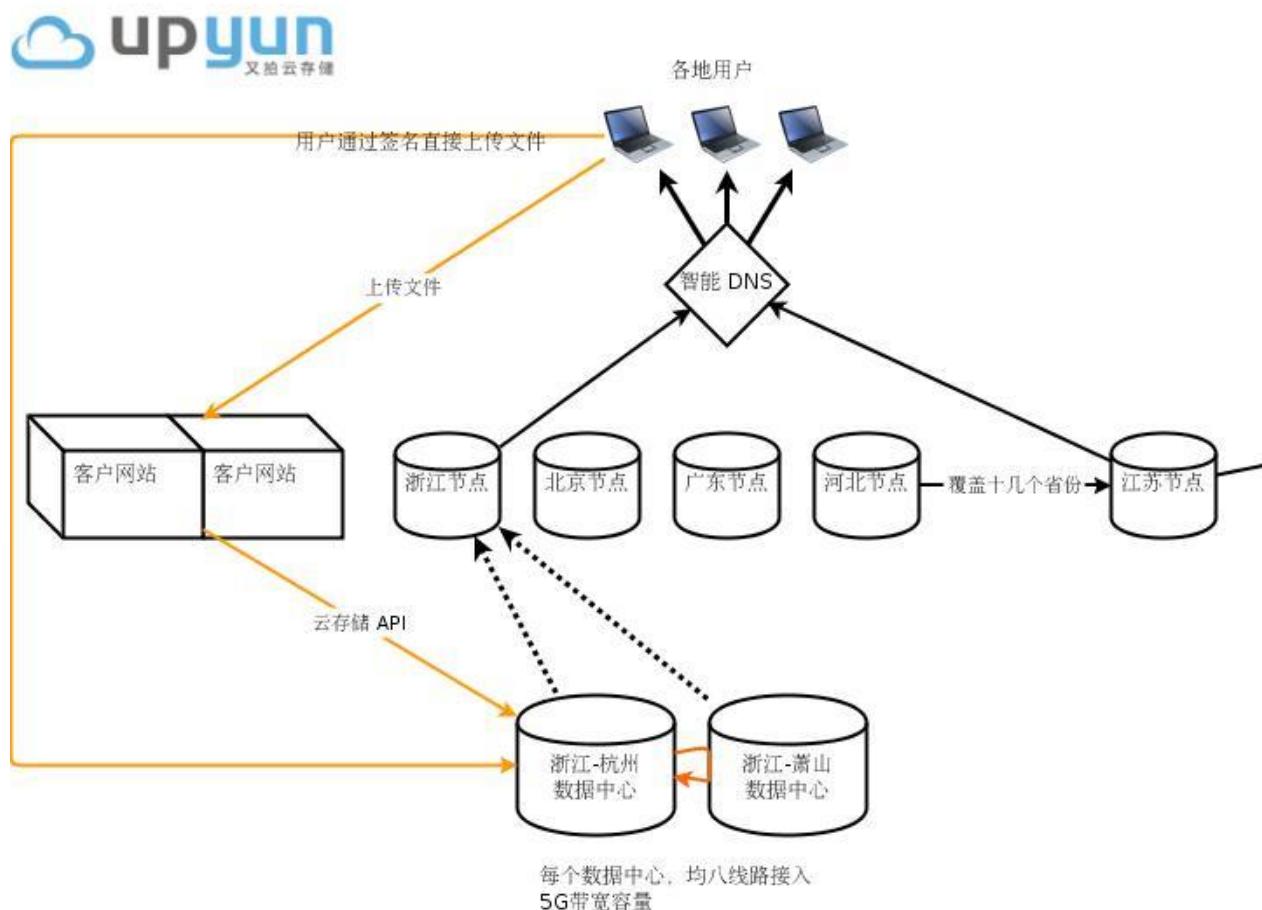
今年很多云服务平台屡屡爆出服务宕机或丢失数据的问题，这让大家对国内云服务更加的不放心，其实云计算并不应该存在这类严重问题，云计算的主要使命之一恰恰是解决稳定和安全隐患问题。如 SAE 类 PaaS 云计算平台是保障我们网站应用的正常服务，高度容错且可扩展，而又拍云存储则属 IaaS 类云计算平台，存储数据的稳定和安全保障是云存储最主要的工作。

先说持续可用性的保证问题。无单点是一个云服务的基础，而目前很多云服务是单点的，所以致使故障频发。一个真正的云计算平台至少应该保证有两个互相热备的数据中心，三分以上分布不同机柜和机房的数据，在机房引入的线路上也应该保证至少电信、联通有两根以上的线路。只有这样才能保证不论是机房断电、硬盘故障、还是断线，都能保持持续的访问。另外就是对于服务器集群的部署上要实现负载的均衡，可采用服务器 HA 互备，lvs 进行 4 层负载，7 层 nginx 进行一致性 hash 及冷热文件调度，一旦有服务器出现硬件故障，前端调度会自动识别并剥离出集群，确保不影响用户每一次的实际访问。

为充分发挥 Nginx 的 7 层代理的优势，我们在此基础上加入了较多的业务模块，如：**一致性 HASH 模块**，根据业务需求通过请求信息进行计算，把请求统一发到后端缓存服务器，避免使用普通负载均衡方式而导致缓存命中率降低，可大大提高缓存集群的业务处理能力；**缓存调度模块**，基于 LRU 和 MRU 算法对全局的所有访问 URL 进行热度分类，从客户端发起的请求到达 Nginx 就能快速确定该请求是否属于热门缓存，而直接到 SSD 磁盘获取资源；**统计模块**，在 Nginx 内部对所有访问 URL 进行统计并汇总，定期向后端业务系统发送统计报告，使得我们可以对客户提供实时的流量统计查询服务，这也是服务计费的标准；

再说数据的安全和泄漏问题。安全性的解决主要是通过多样的备份机制，像云存储主要依托在不同服务器上实现动态的实时三备份，也就是说会自动搜寻用户的数据是否存在 3 份，如没有自动选取服务器生成，这种机制可以完全的保证数据的安全。数据泄漏是使用第三方云计算的最大忧虑，因为云计算的 API 开放性，决定了云计算服务在安全性上的隐患更大。目前通用的解决办法是采用 128 位 AES 加密码保护，以及权限控制，但是其实目前还没有绝对的办法可以杜绝数据部署在云上的泄漏问题。云存储目前主要是托管用户的公开数据，及网站上本身提供给用户访问的数据。

二、快速



快速是互联网平台发展的基石，优秀的速度才能创造有利于增长的用户体验。但是传统的 IDC 部署方式下，受限于硬件规模和存储架构的影响，通常速度很难发挥。这时候云存储就能发挥作用了，其集群服务器部署的方式，能最大的发挥数据运算的效率。开发者在评估云存储服务的速度时，应该看看他们有没有全国分布的 CDN 加速网络，如果没有通常速度都无法保证，严格来说，云服务是需要具备 CDN 节点的。

再就是看这个云服务的 CDN 部署架构是否优良，这个对速度的影响非常大。云存储 CDN 架构采用各地方缓存节点、核心缓存层、中心数据机房，3 层结构部署，前端智能 DNS 调度用户到该用户访问最快的节点，地方缓存节点会保持连接 2 个核心缓存机房做负载均衡及相互备用，避免单路网络问题。核心缓存机房通过多条线路互备到数据机房读取文件。

三、易用

云服务因为其弹性扩容的特点，大幅度降低了互联网平台的运维规划压力。但同时他也可能需要做一些额外的对接开发，因此易用就非常重要。好的云服务会开放高度可用的 API，让用户系统极容易与云平台对接。如果云平台的 API 不够优秀，会让开发者的对接成本以及后续维护成本都非常的高。最好的云服务，应该有一些基于云的处理功能，去帮助用户节省一些工作时间和成本。比如又拍云存储，我们做了 10 种缩略图自定义、文件防盗链、以及与各种第三方平台系统的对接插件，以使得用户易用性更高。

最后给大家一个建议，如何去选择云服务。我们知道亚马逊的云服务划分为 EC2 和 S3 两块，EC2 专用于网站的计算，而 S3 专用于静态文件的存储。在国内目前还没有公司具备亚马逊这样的云服务能力，因此建议大家可以考虑把网站托管到云主机，而静态文件托管

到云存储。而对于数据库这类有高要求的数据应用，还是建议大家使用托管的物理服务器，毕竟目前云主机的性能和稳定性方面仍有待观察。

原文链接：<http://www.infoq.com/cn/articles/szh-cloud-storage-services>

相关内容：

- [Android 中的单元测试](#)
- [Yeoman：构建漂亮 Web 应用的工具和框架](#)
- [如何控制单元测试的粒度？](#)

特别专题 | Topic

Hadoop 管理员的十个最佳实践

作者 [张月](#)

接触 Hadoop 有两年的时间了，期间遇到很多的问题，既有经典的 NameNode 和 JobTracker 内存溢出故障，也有 HDFS 存储小文件问题，既有任务调度问题，也有 MapReduce 性能问题。遇到的这些问题有些是 Hadoop 自身的缺陷（短板），有些则是使用的不当。

在解决问题的过程中，有时需要翻源码，有时会向同事、网友请教，遇到复杂问题则会通过 mail list 向全球各地 Hadoop 使用者，包括 Hadoop Committer (Hadoop 开发者) 求助。在获得很多人帮助后，自己将遇到问题和心得整理成文，希望本文可以对那些焦头烂额的 Hadoop 新手们有所帮助，少走笔者的弯路。

PS. 本文基于 Cloudera CDH 3u4 (同 Apache Hadoop 1.0) 编写。相关推荐配置为官方推荐值或者笔者经验值，它不是绝对的，可能会因为不同的应用场景和硬件环境有所出入。

1. 选择 Cloudera CDH 部署你的 Cluster

动机

大多数管理员都是从 Apache Hadoop 开始学习。笔者最开始也使用 Apache 版本 Hadoop 进行开发和部署工作，但接触到 Cloudera CDH 后，我发现它可以使管理员的工作更简单，不仅可以获得最新的特性和 Bug 修复，有时也会带来令人惊喜的性能改善。

CDH 为什么更好？笔者罗列了以下几点：

1. CDH 基于稳定版 Apache Hadoop，并应用了最新 Bug 修复或者 Feature 的 Patch。Cloudera 常年坚持季度发行 Update 版本，年度发行 Release 版本，更新速度比 Apache 官方快，而且在实际使用过程中 CDH 表现无比稳定，并没有引入新的问题。
2. Cloudera 官方网站上安装、升级文档详细，省去 Google 时间。
3. CDH 支持 Yum/Apt 包，Tar 包，RPM 包，Cloudera Manager 四种方式安装，总有一款适合您。官方网站推荐 Yum/Apt 方式安装，笔者体会其好处如下：
 1. 联网安装、升级，非常方便。当然你也可以下载 rpm 包到本地，使用 Local Yum 方式安装。
 2. 自动下载依赖软件包，比如要安装 Hive，则会级联下载、安装 Hadoop。
 3. Hadoop 生态系统包自动匹配，不需要你寻找与当前 Hadoop 匹配的 Hbase，Flume，Hive 等软件，Yum/Apt 会根据当前安装 Hadoop 版本自动寻找匹配版本的软件包，并保证兼容性。
 4. 自动创建相关目录并软链到合适的地方（如 conf 和 logs 等目录）；自动创建 hdfs, mapred 用户，hdfs 用户是 HDFS 的最高权限用户，mapred 用户则负责 mapreduce 执行过程中相关目录的权限。

推荐指数：★★★

推荐理由：获取最新特性和最新 Bug 修复；安装维护方便，节省运维时间。

2. Hadoop 集群配置与管理

安装和维护 Hadoop 集群涉及大量的管理工作，包括软件安装，设备管理（crontab、iptables 等）、配置分发等。

对于小型集群软件分发和节点管理可以使用 PDSH 这款软件，它可以通过免密钥的 SSH 将文件分发到目标服务器，以及为一组目标设备发送命令并获得反馈。如果是大型集群或者硬件配置差别很大的集群，推荐使用 puppet 这样的工具帮助你维护配置文件，或者通过 Cloudera Manager 以 GUI 的方式的管理集群（注意：Cloudera Manager 不是开源软件，免费版最多支持 50 个节点）。

推荐指数：★★★

推荐理由：提高运维效率

3. 开启 SecondaryNameNode

SecondaryNameNode（下称 SNN）的主要功能是工作是帮助 NameNode（下称 NN）合并编辑日志，然后将合并后的镜像文件 copy 回 NN，以减少 NN 重启时合并编辑日志所需的时间。SNN 不是 NN 的热备，但是通过以下步骤可以实现将 SNN 切换为 NN 的目的。首先，SNN 节点上导入从 NN Copy 过来的镜像文件，然后修改 SNN 机器名和 IP 与 NN 一致，最后重启集群。

特别注意的是 SNN 的内存配置要与 NN 一致，因为合并编辑日志的工作需要将 metadata 加载到内存完成。另外，不仅仅是 SNN，任何保存 NN 镜像的节点都可以通过上面步骤变为 NN，只是 SNN 更适合罢了。

推荐指数：★★★

推荐理由：减少 NN 重启导致集群服务中断时间；NN 节点故障后，SNN 充当 NN 角色

4. 使用 Ganglia 和 Nagios 监控你的集群

当运行一个大型 mapreduce 作业时，我们通常非常关心该作业对 TaskTracker (下称 TT) CPU、内存、磁盘，以及整个网络的带宽情况，这时候就需要 Ganglia 这个工具为我们生成相关图表来诊断、分析问题。

Ganglia 可以监控集群状态，但当你的服务器 down 机或者某个 TT 挂掉，它却无法通知到你，这时我们可以使用 Nagios 这款告警软件，它可以配置邮件告警和短息告警。通过编写 plugins，可以实现自己的监控功能。我们的集群目前做了如下监控：

NameNode、JobTracker 内存

DataNode 和 TaskTracker 运行状态

NFS 服务状态

磁盘使用情况

服务器负载状态

推荐指数：★★★

推荐理由：Ganglia 可以帮你记录集群状态，方便诊断问题；Nagios 可以再遇到问题时第一时间通知你。

5. 设置好内存至关重要

Hadoop 集群安装完毕后，第一件事就是修改 bin/hadoop-env.sh 文件设置内存。主流节点内存配置为 32GB，典型场景内存设置如下

NN: 15-25 GB

JT : 2-4GB

DN : 1-4 GB

TT : 1-2 GB , Child VM 1-2 GB

集群的使用场景不同相关设置也有不同，如果集群有大量小文件，则要求 NN 内存至少要 20GB , DN 内存至少 2GB。

推荐指数 : 00000

推荐理由：几个组件中 NN 对内存最为敏感，它有单点问题，直接影响到集群的可用性； JT 同样是单点，如果 JT 内存溢出则所有 MapReduce Job 都无法正常执行。

6. 管理员玩转 MapReduce

Hadoop 原生 MapReduce 需要 Java 语言编写，但是不会 Java 也没问题，通过 Hadoop streaming 框架管理员可以使用 Python , Shell , Perl 等语言进行 MapReduce 开发，但更简单的办法是安装和使用 Hive 或者 Pig。

推荐指数 : 000

推荐理由：减少运维时间，快速响应各种 ad-hoc 需求和故障诊断。

7. NameNode HA

前面已经说过，NN 是整个集群可能出现的单点故障。

Hadoop 通过在 hdfs.site.xml 文件的 dfs.name.dir 属性指定保持的 metadata 路径，如果希望保持到多个路径，可以使用逗号分割配置多个路径。

```
<property>    <name>dfs.name.dir</name>  
<value>/data/cache1/dfs/nn,/data/cache2/dfs/nn</value> </property>
```

Hadoop 官方推荐配置为 metadata 配置多个 path , 其中包含一个 NFS 的路径。但根据笔者一次集群严重故障经验 , 即使这样 , 还是导致了所有镜像文件损坏 , 包括 SNN 上的镜像文件 , 所以定期备份一个可用的副本还是很有必要的。

推荐指数 : ⚡⚡⚡⚡

推荐理由 : Cloudera3uX 和 Apache1.0 的 NN 单点问题是大家最头痛问题之一 , 多些准备 , 少许痛苦。

8. 使用 firewall 阻止坏人进入

Hadoop 的安全控制非常简单 , 只包含简单的权限 , 即只根据客户端用户名 , 决定使用权。它的设计原则是 : “避免好人做错事 , 但不阻止坏人做坏事”。

如果你知道某台 NN 的 IP 和端口 , 则可以很轻松获取 HDFS 目录结构 , 并通过修改本机机器用户名伪装成 HDFS 文件所属 owner , 对该文件进行删除操作。

通过配置 kerberos , 可以实现身份验证。但很多管理员使用更简单有效的办法——通过防火墙对访问 IP 进行控制。

推荐指数 : ⚡⚡⚡⚡

推荐理由 : 安全无小事 , 防范于未然。

9. 开启垃圾箱(trash)功能

动机

我曾经犯下一个错误，在我加班非常累，大脑稍有混乱的时候，不小心删除执行了一个命令 “hadoop fs -rmr /xxx/xxx” ，没有删除提示，几 TB 的数据，一下子就没有了。简直让我崩溃，后悔莫及。这时你多希望有个时间机器可以让 HDFS 恢复到删除前的状态。

trash 功能就是这个时间机器，它默认是关闭的，开启后，被你删除的数据将会 mv 到操作用户目录的”.Trash”文件夹，可以配置超过多长时间，系统自动删除过期数据。这样一来，当操作失误的时候，可以把数据 mv 回来。开启垃圾箱步骤如下：

vi core-site.xml，添加下面配置，value 单位为分钟。

```
<property>  
    <name>fs.trash.interval</name>  
    <value>1440</value>  
</property>
```

笔者在 CDH3u4 下不用重启 Namenode 就可以生效。开启垃圾箱后，如果希望文件直接被删除，可以在使用删除命令时添加 “–skipTrash” 参数，如下：

```
hadoop fs –rm –skipTrash /xxxx
```

推荐指数：★★★★★

推荐理由：想要时间机器吗？

10. 去社区寻找帮助

Hadoop 是一个非常优秀的开源项目，但它仍存有很多尚未解决的问题，诸如，NN,JT 单点问题，JT 挂死问题，Block 在小文件下汇报效率低下等问题。此时可以通过如下渠道找到可以帮助你的人，笔者几次集群严重故障都是通过 Cloudera 公司的 google user group 直接获得几位 committer 的帮助。通常前一天提问，第二天就会有反馈。下面是两个能够帮助的你的社区，当然你也可以帮助其他人：

Apache hadoop 的 mail list :

http://hadoop.apache.org/mailing_lists.html

Cloudera CDH google group:

<https://groups.google.com/a/cloudera.org/forum/#!forum/cdh-user>

推荐指数：★★★★★

推荐理由：没有人比软件作者更熟悉 Hadoop 本身，去社区求助，帮你解决很多自己无法跨越的问题。

Cloudera 简介：

公司是一家 Hadoop 软件服务公司，提供免费软件 CDH 和 Cloudera Manager Free Edition，同时提供 Hadoop 相关资讯、培训、技术支持等服务。Hadoop 创始人 Dong Cutting 在该公司任架构师，同时该公司拥有多名 Apache Committer。

作者介绍：

张月，Java程序员，7年工作经验，2007年加入蓝汛 chinacache 至今，目前从事 Hadoop 相关工作，关注敏捷和海量数据领域，关注软件开发过程。他拥有 Cloudera Certified Administrator for Apache Hadoop (CCAH) 和 Cloudera Certified Developer for Apache Hadoop (CCDH) 证书，博客：heipark.iteye.com

原文链接：<http://www.infoq.com/cn/articles/hadoop-ten-best-practice>

相关内容：

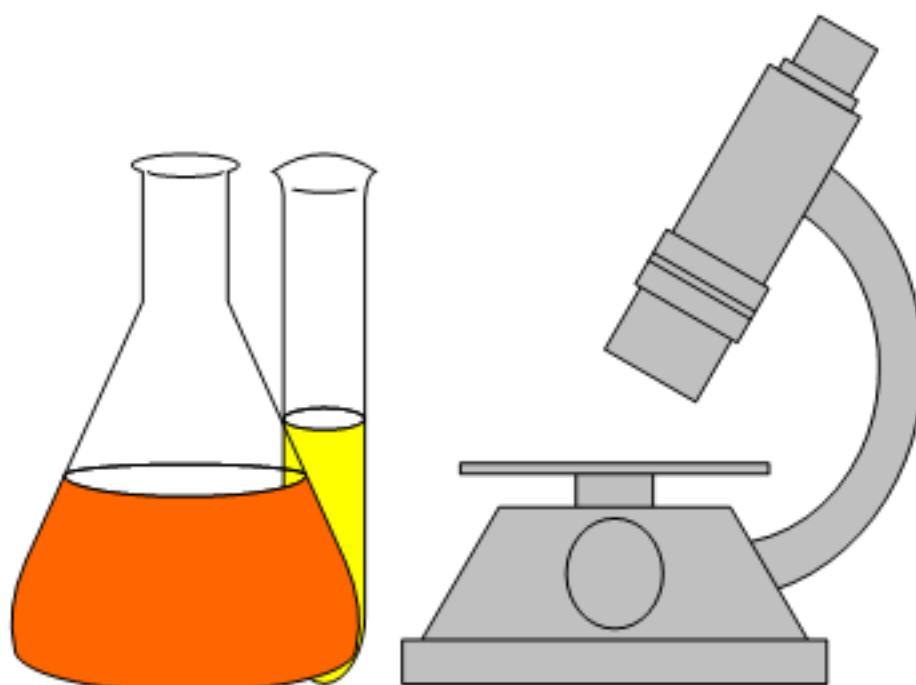
- [Android 中的单元测试](#)
- [编写综合的单元测试](#)
- [大型搜索引擎的系统测试方法及案例分享](#)

推荐文章 | Article

分布式开发

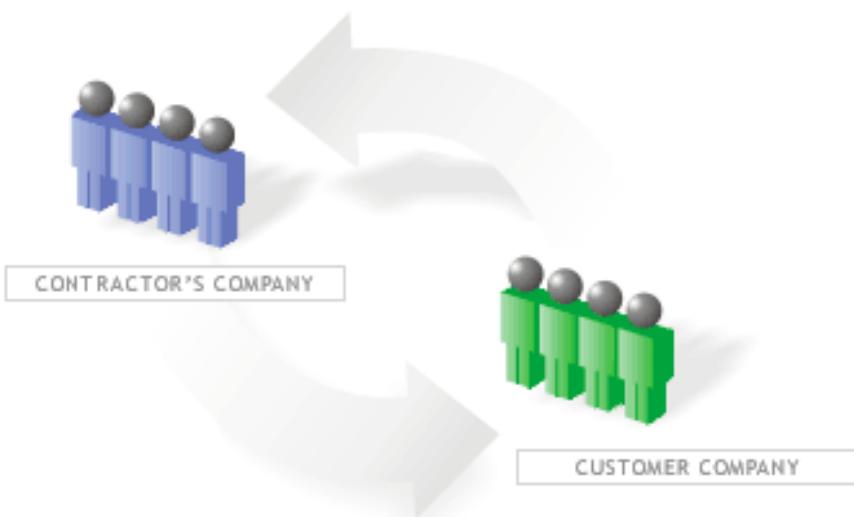
作者 李响

作为 ThoughtWorks 的一名咨询师，我曾不止一次的被问到 ThoughtWorks 的交付项目和一般意义上的外包到底有何区别。要区分差别，首先要对外包加以定义，外包从最传统的 IT 外包到业务流程的外包，以及最近几年新兴的知识流程外包，其本身的定义也在不断的演化。每种外包有其不同的诉求，传统的 IT 外包和业务流程外包追求成本的降低，而知识流程的外包则更着眼于客户 知识能力的提升以及团队的成长。



Knowledge Process Outsourcing

ThoughtWorks 的交付项目更多的是一种知识流程外包的高端服务。交付项目的成功不仅是交付卓越的软件，还需要在交付卓越的软件的过程中，深刻理解客户的市场需要和业务模式，并通过自己的努力去影响客户，最终和客户以一个团队的模式一同交付高质量的软件。而怎样去影响客户的行为，进而鼓励他们去尝试更多最佳实践呢？是否只能通过咨询项目达到这个目的呢？一般来说，咨询是需要长期在客户现场进行持续的影响，才能产生效果。但交付项目如同很多外包项目一样，通常是离岸的。那么隔着十万八千里的距离，再加上几个小时的时差，如何才能真的像一个团队一样紧密合作，并且持续的影响客户尝试更多最佳实践，一步步实现交付卓越软件这一目标呢？沟通，就是基于知识流程外包的分布式开发中，最大的挑战。



正如前面所说的，我们有着地理上的距离，这就意味着我们会有巨大的文化差异。客户说“还可以”，可能意味着其实有些问题。如果说东西方的文化差异可以通过了解当地的风俗民情去理解的话，那么企业的文化差异就需要一些同理心了。作为一家“不创新就会死（Innovate or Die）”的先锋企业，我们的咨询师通常是勇于尝试，热衷创新的；而我们

的客户却很有可能是拥有庞大的组织结构的传统企业，做任何一点改变都需要考虑所有部门利益，甚至很长时间都难以做出一个决定，这和我们的快速反馈原则往往都是不相容的。



所以沟通毫无疑问会成为分布式开发最大的挑战。经过多个分布式开发项目，我们总结出以下几个最佳实践：

快速启动 (Inception)



快速启动中，所有的项目干系人通常会聚集到一起，在两周左右的时间中进行一系列的愿景分享、业务探索、产品设计、需求收集以及计划发布等活动。

快速启动的目的首先是让大家聚到一起。作为可能要合作上几个月甚至几年的队友，虽然之后很长一段时间大家天各一方的工作，但在启动阶段能够认识彼此，通过面对面的沟通对对方的工作有感性的认知，无疑对于后面工作的展开会起到重要作用。

快速启动通过两周密度较大的活动，让每个团队成员，在尽可能短时间的内，达成共识。这其中包括了解战略愿景，进而理解项目的机遇与挑战；通过和市场人员的沟通去理解客户需要；和产品部门一起设计出符合客户体验需要的产品原型，进而拆分出可开发的需求；根据优先级排列并发布计划。

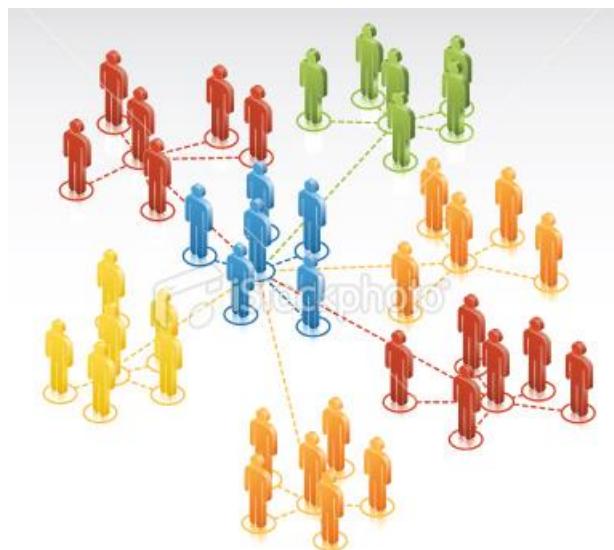
在两周的时间里，团队将一个商业概念转换为形象的原型，并制定可供开发使用的发布计划，快速启动的交付物无疑对于后面的开发阶段非常重要，而更重要的是快速启动的交付物是整个团队一起工作产出的。每个人都参与了产生的过程，分享了相同的上下文，这对于后面的开发阶段的有效沟通会非常有帮助。

在快速启动之后，团队一般就可以开始进入迭代 0 的开发了。在迭代 0 中，所有团队成员能继续一起工作，在这个较短的迭代搭建好环境，对架构都形成共识，试着在迭代 0 里去完成一个基本的需求。

快速启动和迭代 0 这段时间中，所有团队成员紧密合作，不光对于交付软件有所帮助，也能更好地让咨询师理解客户团队的工作习惯、技术水平和代码风格，进而评估哪些最佳实践是适合客户团队的。比如我曾经参与过的一个项目中，在快速启动阶段，我们发现客户团队的很多基本实践，如 TDD，已经做的很好，代码水平也很娴熟，但部署的方式却很落后。同时，业务部门的需求明显需要持续交付来支撑，于是持续交付就成了重点尝试的实

践。每个项目都具有自己的独特性，所以通过快速启动的方式，用最快的速度了解客户的知识流程上需要改进的地方，这样跟客户沟通的时候，才会让客户更加认同知识流程外包的高附加值。

站会：



相信了解敏捷的人对站会这一实践都不会陌生。在分布式开发过程中，我们不仅开展了团队的站会，还进行了基于角色的站会。一般而言，团队的站会更多的关注故事卡的状态流动，检查路障，而不会关注每个不同角色的具体工作；而在基于角色的站会上，例如开发站会上，开发人员之间可以讨论技术上的某个难点。同样的，业务分析人员也要通过每日的业务分析站会来了解产品经理的思路变化和业务部门的更新。我曾经遇到过一个项目有来自三地的测试人员，那么测试人员之间的站会就非常重要，测试人员利用角色站会这一机会讨论测试策略。随着项目的演进，测试的重点也不断转移，从一开始关注功能测试到逐渐关注集成测试，角色站会给他们提供了一个契机及时讨论项目碰到的问题和机会。

迭代计划会议：



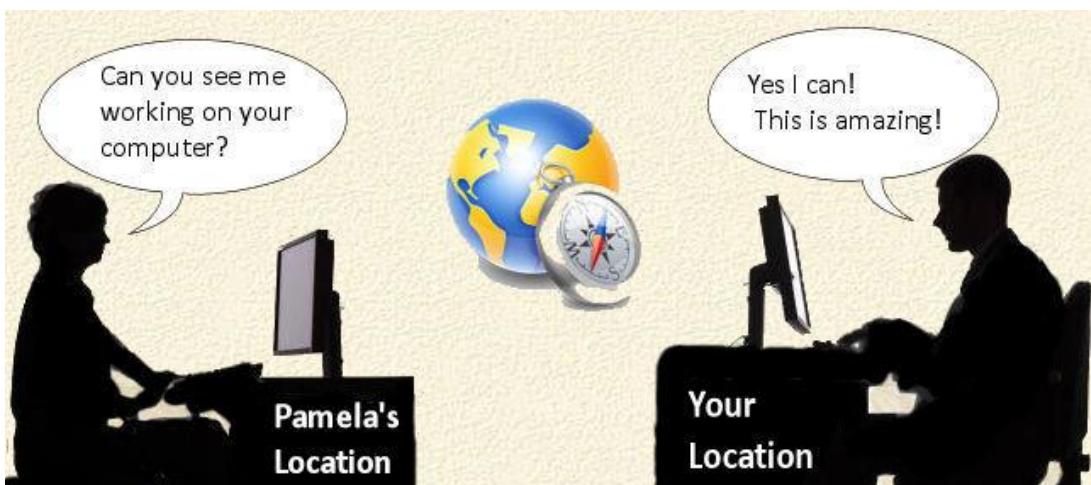
迭代计划会议也是常见的实践。在分布式的场景下，迭代计划会议需要更多的准备，要求各方团队成员提前理解下一迭代需要完成的需求，这样大家可以在迭代计划会议的时候着重讨论有疑惑的需求，而不是浪费很多时间在解释需求本身是什么上。在答疑解惑之后，大家对要完成的迭代目标必须形成一致的意见。同时，在分布式的开发中，不同地域方的团队成员会更加关注本地团队所能完成的需求，却忽略了其他地域团队的开发速度。在其他地域团队遇到困难时，更重要的是帮助对方一起完成对方做的需求，而不是只关注自己做的需求。通过一起达成迭代目标，让大家分享共同的责任感。

在实际项目过程中，通常业务分析师会提前将计划的需求故事发给团队成员。运用合适的分布式项目管理工具会让这一过程更加容易，如 Mingle 或者 Jira。在工具中整理出下一个迭代的故事墙，让团队成员提前看到虚拟墙。在迭代计划会议上，主持方将虚拟墙打开并屏幕共享给其他方，这样各方就可以关注在同一个需求故事卡的讨论上。

回顾会议：

由于各种限制原因，分布式团队很容易产生沟通不足现象。而回顾会议创造了必要的沟通桥梁，因而非常重要，一定要保证所有方都能参与进来。远程回顾会议通常利用在线白板，让每个成员提前在在线白板上提交上个迭代里做得好的和值得改进的地方。这样，当回顾会议开始后，每个人都是有所思考的，提出的意见更有深度，而且也可以更好地利用回顾会议的时间一起探讨所有人都关注的问题。通常，通过回顾会议可以发现，处于不同地理位置的多方成员往往关注的事情也不同，而会议上各方成员又可以了解对方在担忧什么，遇到了哪些困惑，并将这些担忧困惑分享，形成大家都认同的改进行动。

结对编程：



结对编程这一实践对于知识的传递非常重要，即便在分布式团队中，结对编程依然非常重要。合理地利用时差，在各方重叠的工作时间里通过屏幕共享工具远程结对，是保证代码质量的重要手段。

每天合理地利用远程结对不光可以传递知识，同时也是一种有效地影响其他团队成员的方式。远程结对可以让咨询师清楚了解处于不同地域位置的客户团队的代码水平、代码风格和思维方式，从而可以通过每天频繁的结对编程言传身教最佳实践。这种影响行为改变的方式效果显著，而且对于增强团队成员互相了解信任，并形成有统一责任感的“同一个团队”，也非常有帮助。

但不可否认，远程结对会影响开发效率。并且，由于各方工作时间安排以及公共假期等原因，很难保证远程结对的频率。所以，远程的代码评审（Code Review）就显得格外重要。每天各方开发人员在一个固定时段去评审所有提交的代码，能够让团队成员不光关注自己，也了解别人做了哪些代码改动。同时，代码评审对于形成统一的代码风格也很重要。当处于远端的咨询师想去影响客户的代码风格时，如果无法让所有的人都理解为什么要用某种风格，那么之后的一致性也就无从谈起。而远程代码评审就能在沟通不足的条件下，充分展示好的风格带来的改进（Lead By Example），从而达到促成远程提升客户技能的目的。

在分布式开发中，我们依然遵循基本的实践，同时为了克服远程开发沟通不足的缺陷，并达到整个团队能力提升的目的，我们更关注建立沟通渠道和及时的分享。所有的实践在满足基本的目的之后，都要考虑是否能通过这一实践分享知识以及获取知识的正确方式，客户成员是否能得到能力的提升。



除了沟通这一挑战之外，分布式开发环境下当然还有其他各种挑战，如如何远程的让团队保持足够的凝聚力，让大家不因为距离而产生陌生感？所以，经常在工作之余进行些适合远程的团队的游戏，定期邀请各方成员去对方工作的地方工作几周，体会对方的工作环境，能够更好地帮助团队保持一致的目标。此外，分布式团队模式下的项目也有需要关注的特有的风险，如各自国家的成员有不同的公共假期，最后的部署上线如何安排等细节都需要去关注。

由于基于知识流程外包的分布式开发追求交付的成功以及团队的成长，那么面对不同的客户团队就会有不同的挑战。作为咨询师，在关注软件交付本身的同时，更要关注如何提升团队的能力。通过以上这些实践不难发现，虽然依然采用敏捷的基本实践，但在分布式开发的场景下，适当地改进基本实践，才能真正实现高附加值的卓越软件交付。

原文链接：<http://www.infoq.com/cn/articles/distributed-develop>

推荐文章 | Article

Node.js 之网游服务器实践

作者 尧飘海

随着 [Node.js](#) 的不断发展与壮大，应用范围也越来越广泛，从传统的企业应用，到互联网使用，再到云计算的发展，它的身影也是随处可见。当然，它的受欢迎程度能在短时间内得到这么快的发展，除却与其本身的事件模型及 [V8](#) 的性能优化等一系列特性有关之外，还和国内外很多互联网公司的攻城师的大量应用和参与到开源项目中有密切关系，如网易的游戏开发，淘宝的数据之美等等。随着 HTML5 应用和移动互联网平台的指数增长，越来越多的用户使用了移动平台的休闲服务，采用 Node.js 实现高性能和可扩展性的游戏服务将是一件有意义的工作。

在互联网上，目前有一些采用 Node.js 实现的开源游戏服务框架，如 Mozilla 的 [Browser Quest](#)，Google 的 [Grits](#)，[Chilly](#) 等。但是无一例外，这些框架不但与游戏逻辑联系紧密，而且几乎没有可扩展性和性能数据，同时也不提供任何游戏开发的管理工具，除了采用 JavaScript 编写外，很难体现出采用 Node.js 实现游戏开发的优越性。

概念

通常游戏分为角色扮演类和策略类及混合类等几种游戏类型。那么在网页游戏类型中，根据游戏的类型，开发者可能采用不同的架构实现方式，如策略类游戏可能更偏重于游戏的策略性和逻辑性，也就是考验游戏玩家的各种组合或搭配之类的游戏，对实时性的要求不

会很高，在用户的可接受范围之内即可，因此也可以采用常用的 Web 应用开发模式来实现，而客户端采用轮询或长连接等方式来实现，这些应用模式也有很多的相关经验可以参考，因此能做到具有较好的可扩展性及伸缩性；此外，应用服务器和服务框架等也可以采用现存的技术实现，但是这只能满足对游戏实时性要求不高的场景服务，由于此类的应用方案非常成熟，在下文的讨论中，将不再详述。

角色扮演类游戏根据策划的要求可以实现各种各样的功能，如聊天，打斗，自动刷怪等各种复杂的功能，但是开发者基本可以把这些功能抽象地划分为二大类，即服务端的网络传输与逻辑运算能力，这样游戏服务的开发和实现会稍微简单些。现在，网络的实时传输基本上是采用 Socket 服务器来实现的，逻辑运算这个就不需要再述了，因此，游戏服务器就可以粗略的归纳为 Socket 的服务器综合游戏业务逻辑的实现。

与 Web 应用区别

从上面游戏服务的概念简单介绍中可以得知，实时游戏服务器的开发与传统的 Web 开发还是具有一些不同点，这些不同点着重体现在以下几个方面：

1. 大部分的 Web 应用还是短连接的方式去实现，而游戏的实时性需要采用长连接的方式进行。
2. Web 应用的场景一般是读多写少的应用，热点数据基本可以缓存；游戏服务的数据经常发生变动，如移动中的路径，打怪掉血等，缓存基本很快就失效，属于写多读少的应用场景。
3. 广播特性，即 Web 应用行为上基本只需要用户与服务器交互，其他用户要实现共享他人信息的时候，会采用类似拉的方式如长轮询来实现数据交互，即使在没有数据更新的状态时也需要消耗一定的服务端资源，造成一定的浪费；游戏里面的交互方式需要实时进行，对影响三方的数据，必须采用广播的方法进行消息推送，即推的模式。

4. 目前的 Web 应用开发一般采用无状态性的方式来实现，因此可以实现较好的可扩展性，很多 Web 服务会采用绑定会话或集中会话等方式工作，当后面的某台服务器出现宕机时，服务也可以很快的切换；但是游戏服务器和用户之间的连接是有状态的，在断开时需要进行相关的通知和数据持久化，在重连时需要进行状态恢复等手段来维护游戏世界的运行。

除以上几点不同之外，还有负载均衡的策略，服务驱动方式，系统判定，系统安全如外挂作弊等相关问题，文章中不再对他们进行对比说明。一般来讲，根据游戏的规模与需求，通常游戏服务器的实现的模式会使用如下的三种架构方式来实现。

单进程服务架构



图 1 单进程服务架构

图 1 是目前互联网上的很多游戏服务器使用的架构，比如商用的 [SmartFoxServer](#) 游戏服务器，开源的有 Google 的 [Grits](#), Mozilla 的 [Browser Quest](#) 等框架。所有的程序在同一个进程里面运行，架构简单，开发人员上手快，开发和调试非常方便又快速，成本较低，后期的维护成本可能随着游戏项目的大小而不同，但是整体应该不会太高。很明显，游戏的世界是在同一个进程里运行，在单个场景都有不少在线用户的时候，如果采用单进程的

Node.js 实现的话，每个用户的操作可能会有一定的延迟，特别是在大量用户同时做大量操作与 AI 运算时，如服务端寻路和自动杀怪，延迟会更加严重；同时单进程对计算机资源使用有限。因此只适合游戏的原型制作或小型的游戏开发。另外，由于游戏的状态与复杂性，也无法较好的实现游戏的可扩展性，基本只能通过添加游戏服务器的方式来实现，即所谓的开新服，而这些用户之间是无法在同一个世界里通信会话和交互的。尽管 Node.js 支持原生的 TCP 服务，使用简单；但是目前大部分网络应用中，尤其是支持 HTML5 的应用，[socket.io](#) 由于性能较好并且使用简单，成为大部分应用采用的网络库来实现高性能 websocket 服务。在移动互联网应用中，由于协议较复杂与通信数据等问题，需要一定的裁减。下面示例中，就是最简单的单进程服务的原型：

```
var socketio = require('socket.io');

var io = socketio.listen(8080);

io.sockets.on('connection',function(socket){

    socket.on('disconnect',function(){

        //user leave });

    socket.on('message',function(msg){

        if (getLoginStatus(socket)) {

            game[data.action].apply(null,[socket,msg]);

        } else {

            game['login'].apply(null,[socket,msg]);
```

```
} }));});
```

多进程服务架构

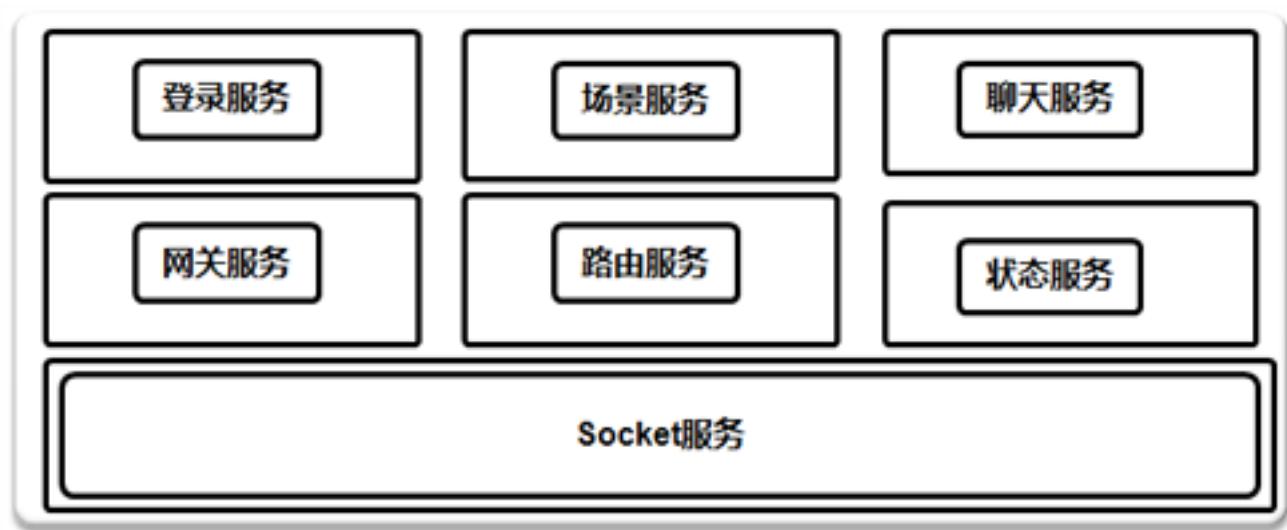


图 2 多进程服务架构

在上图 2 中，我们可以看出，每个进程负责采用单一职责，各进程间通过一定的规则（如 RPC 远程过程调用）来进行通信，游戏中的各场景服务使用一个进程来服务，实现各场景游戏运行的分离。但是在开发过程中，不需要考虑每个场景具体的信息，全部采用统一的代码来实现，开发方便，通过添加适当的参数来实现较易的调试，每个进程分别运行在服务器的多核上，即可以充分使用计算机的资源，也由于职责单一，由于每个进程都专职做自己的事，因此，在压力大的时候，可以很明显的查出问题所在，并可以较好的实现对相应的服务进行调优，可以达到较好的性能和一定的可伸缩性。但是，在各游戏的世界里，无法准确的知道游戏的状态如在线人数之类的，会存在热点数据和服务过载等一些问题，造成部分服务堵死等现象，也无法达到在不停服务的情况下进行服务器扩容与自动切换等。

问题，同时会给游戏运营带来一定的问题，不能充分的对玩家数据进行分析与挖掘，因此，这种实现方式可以满足中小型的游戏开发。这种架构目前很多的游戏服务器中还在使用，技术发展也较成熟，如传奇服务端架构等。

分布式服务架构



图 3 分布式的服务架构

同样，分布式的服务架构与多进程的架构具有很多相同点，如进程服务单一职责，较好的性能调优，同时采用集中式的方式对游戏服务器进行管理，各游戏服务器可以通过自定义的方式即 DSL 进行游戏业务的路由与分发，通过主备服务器的状态可以很清楚的知道每个服务的状态及运行情况，这样可通过动态添加服务来到达负载均衡，具备较好的性能和可扩展性，能满足大型游戏的开发。当然，采用这种方式，会具有一定的复杂性，同时各服务之间可能是不在同一台服务器上的，因此会带来一定的网络开销，在大量广播的场景中，网络 IO 压力大，需要通过定时批量或 AOI 服务等方式来进行广播通信，同时，游戏的通信协议需要经过特别的设计，尽量避免数据序列化上的 CPU 重复开销，采用胖客户端的方

式去分担游戏服务器的序列化与反序列化；另外还有可能引入分布式事务等相关问题，需要在一个集中点进行处理。当然，在游戏设计中，策划开发人员需要尽量避免这种跨场景跨进程的游戏逻辑需求。

在分布式架构实现中，目前商业上较成功的案例是 [BigWorld](#) 游戏服务框架，国内外很多大型的游戏开发商都在使用。而在开源部分，目前还没有较稳定成熟的方案，我们团队采用 Node.js 正在开发的 Pomelo 游戏服务框架正在努力实现这一目标，现已进入最后的文档整理阶段，预计将在十月份进行开源与线上示例演示。

总结

采用 Node.js 来实现网络游戏服务器的开发，除了能利用 JavaScript 的动态语言的优势特性、强大的开源社区和 V8 引擎的优越性能外，在网络数据传输的异步化方面具有快速、实时等事件编程模型；特别在 HTML5 的应用开发中，还具有前后端代码共享、DSL 灵活定义等特点。但是也由于 JavaScript 的动态脚本语言的性质，同时 V8 在内存使用上的限制等问题。尽管 V8 引擎对脚本语言通过动态编译进行了静态化，但如果开发工程师在大规模开发时，没有注意到一些代码的细节，比如代码的额外异常检查和类属性的动态变化等方面，导致未经过调优的代码整体上还是无法和 C、Java 等静态语言相比的，然而在经过调优过后的代码是可以达到静态语言的性能的。在云计算应用中，也得到很多公司的推广与使用，如国外的 SmartOS, Vmware 以及国内的阿里云等，相信其性能与应用层面都会发展的越来越好；

此外，采用 Node.js 来实现游戏服务框架并开源也是一件非常有价值的事情，在追求高性能与低成本同时，希望能给开发者提供即简单又快速的开发方式，游戏开发者通过使用游

戏框架，可以完全把精力放在业务代码的实现，不需要再把精力放在框架的实现。有机会我们将在后面的一系列讨论中介绍采用 Node.js 来开发游戏服务架构遇到的很多性能问题及优化过程。最后，本文并不是建议大家一定要采用 Node.js 去开发游戏服务器，只是给新老游戏开发者提供一种解决方案，欢迎大家关注和探讨我们团队近期将开源的 Node.js 游戏服务解决框架。

个人介绍

尧飘海，网易杭州开发工程师，系统分析员，正致力于 Node.js 的移动游戏引擎开发和性能优化等方面的工作，同时对 JVM 性能和服务端的应用架构设计也很有兴趣，欢迎交流。
个人 Github 地址：<http://github.com/piaohai>。

原文链接：<http://www.infoq.com/cn/articles/nodeJs-online-game>

相关内容：

- [微软随.NET 4.5 发布新 REST API 框架](#)
- [IG GROUP 开源 RESTdoclet 项目](#)
- [IT 领域对技术的重视超过了思考](#)

推荐文章 | Article

阅读者(二十二):从重构到模式

作者 [张岳](#)

在谈论对《重构与模式》(Refactoring to Pattern)一书的认识之前，我想谈两点作为一个有近四年工作经验的软件从业人员在面向对象领域的一些困惑：

- 重构的困惑：随着重构技巧的不断实践，诸如重命名、提取函数、搬移函数、内联、改变函数参数等技巧已经掌握得很熟练，然而一些困惑也开始出现。经过长期实践发现，我能使用重构所做的改善基本停留在类的内部，很容易地处理一些常见的坏味道，但是很难上升到类与类之间这个层次。一旦上升到这个层次，思路变得不清晰，把控能力明显下降。我也曾跟一些同事交流，与我有差不多经验的同事都有类似的困惑。
- 模式式的困惑：读完《设计模式》以及《深入浅出设计模式》之后，了解了很多模式框架，进而知道了很多解决一些典型问题的最佳方案。但是应用起来只会照搬照套，很难做到恰到好处。甚至某些时候适得其反，引入一些不必要的复杂性。更糟糕的是，一旦学会一些模式之后，非常容易滥用。有些时候为了显示自己对模式的把握技高一筹，在代码中恨不得“无处不模式”。后来逐渐发现模式这个工具不是万能的，如果用错只会让代码变得奇怪，甚至不可理解。因而，如何恰到好处地运用模式一直困扰着我。

当我读完《重构与模式》的前言时，立即意识到此书是我寻觅良久的作品。因为作者开门见山地道明了本书的主旨，即将重构与模式结合，使用模式来改善既有设计。如果你也认为模式是对面向对象设计在特定问题域的最佳实践，那么使用它来构建类结构岂不是高屋建瓴？当然，在前言的“此书目的”中也清晰明了地指出了如何才能将重构与模式很好的结合，那就是使用“模式导向的重构改善既有代码设计”。

纵观整本书，结构十分清晰。首先回顾了重构和模式的基本概念，当然它们是本书的基石，同时给那些对于重构和模式缺乏了解的读者一个熟悉它们的机会。接着谈到了代码的坏味道，这也是很有必要的。如果都不清楚什么样的代码是不好的，改善就无从谈起，因为代码的坏味道是重构的起点。最后是本书的精髓，所谓的“干货”部分，即授予读者如何将模式与重构结合的利器。通过代码实例，结合前面提到的坏味道，讲解每种重构技法的具体操作步骤。

任何一本书的结构设计与章节组织，其作者往往基于少量因素考虑，无法兼顾广大读者的需求。鉴于本书的实际操作性极强，作者很贴心地从读者学习的角度出发，考虑实例间对同一个项目的引用关系，在第五章最后部分给出了一个学习这些重构技法的推荐顺序列表（表 5-1）。具体到每个重构技巧的介绍模式跟 Martin Folower《重构》一书基本类似，所以对于熟悉 Martin 著作的读者来说，读这本书有种似曾相识的感觉，很容易上手。从第六章开始都采用了如下组织结构：

- 名称
- 概要
- 动机
- 做法
- 示例
- 变体

我认为最有价值的是动机和做法这两部分，因为每个重构技法的理论指导都在这里了。特别是在动机部分，作者给出了此重构技法的优缺点。哲学的基本原理告诉我们任何事物都有两面性。有利有弊，给读者决定采用此重构方法时提供了具体的考量标准。

此外，在第四章《代码坏味》中，作者给出了一个表（表 4-1）。此表列出了本书处理的 12 种代码坏味的 27 种重构手法。我想它的最大作用在于，当你日后遇到某种代码坏味而忘记重构方法时，可以充当一个快速查阅工具。同时，本章还针对每种坏味言简意赅地说明了它的定义以及带来的危害，并且给出了解决它的重构方法。

至于每个具体的重构技法，我认为最佳学习实践是根据表 5-1 建立工程项目，把每个重构技法按照作者的指示实际动手实践。强烈推荐使用 IntelliJ Idea 来操作，因为它提供了很好的重构功能。例如，作者在第十一章提到的几个重构方法，就可以选中需要重构的代码块，使用几个快捷键立马可以搞定。具体的说，比如提取参数，只需要选中需要提取为参数的变量，`Ctrl+Alt+P`，重构结束，所有调用此方法的地方都自动将参数值传入。当然，工具虽然能节约敲代码的时间，并保证正确性，但还是需要认真阅读作者所写的每个步骤，深入理解背后的原理。

本书的好我不需要再画蛇添足地说太多，因为推荐此书那些大牛们的华丽言辞已经无以复加。如果还是太抽象，我将给本书作序的软件行业泰斗，ThoughtWorks 首席科学家—— Martin Folower 的赞誉摘抄如下：

正因为如此，如果说有什么人最合适写模式与重构之间的联系，那应该非 Joshua 莫属了。

个人觉得本书略显不足之处在于其中文译名，有点错失原作者的主要意图了。Refactoring to Patterns 的目的在于使用重构工具，以模式为导向来改善既有代码设计。如果按照中文译名理解，感觉只是介绍重构和模式而已，没能体现上述意图。

无论如何，本书是一本难得的理论与实践兼顾的作品，对于掌握和深入理解模式导向的重构有非常实用的指导意义。

个人介绍

张岳，ThoughtWorks 咨询师。擅长面向对象设计与编程，对敏捷开发与实践有丰富经验，关注互联网及移动平台新技术。主要项目经验在于开发、部署和运维企业级 Web 系统，数据整合与分析，以及企业级系统集成。主要熟悉的平台有 Java、.NET；经常使用的语言有 Java、C#、Ruby、JavaScript 等等。

原文链接：<http://www.infoq.com/cn/articles/refactor-pattern>

推荐文章 | Article

采访及书评--回顾手册

作者 [Anand Vishwanath](#) 译者 [李剑](#)

[Patrick Kua](#) 最近写了一本书，名叫《[回顾手册：敏捷项目向导](#)》。在这本书中，Pat 总结了他过去几年在真正的敏捷团队中做回顾的经验。电子版在 [LeanPub](#) 出售。LeanPub 同时还提供了[免费样章](#)。

作者在书中介绍了一些如何准备回顾会议的宝贵建议，他详实描述了为了保证参与者交流顺畅，应该如何有效控制时间、使用合适的材料、布置会场。

书中有一章专门讲怎样组织分布式回顾，这部分内容对离岸/分布式团队很有借鉴意义。它推荐在每个地点都使用视频接入，并通过多个引导者（facilitator）和各种在线工具在各地之间共享信息。书中还提到了在多文化背景下会遭遇到的一些社交方面的挑战，也针对这些问题提供了建议。

敏捷团队或许会因为一再使用同一种回顾方式而感到厌倦。书中的一些小贴士可以保证回顾会议常开常新：人们可以每次问一些不同的问题，或者改变一下环境，让大家保持新鲜感，积极参与。

全书共分 10 章

- 回顾的基石
- 准备回顾
- 引导回顾

- 首次引导小贴士
- 分布式回顾
- 其他回顾形式
- 回顾之后
- 常见的回顾坏味道
- 回顾保鲜
- 尾声

Patrick Kua 最近接受了 InfoQ 编辑 Anand Vishwanath 的采访

InfoQ：你可否跟我们的读者简单介绍下自己？

Pat：没问题。我在日常工作中扮演的是技术领导的角色，我的团队都用了很多敏捷实践。我在敏捷环境中工作大概有十年了，经历了各种环境，各种不同的“敏捷”风格。我也辅导培训过很多团队和组织，对学习和持续改进背后的理念特别感兴趣，所以把回顾当作一项核心敏捷实践。

InfoQ：是什么促使你写一本回顾方面的书的？

Pat：在我做敏捷开发，做讲师，做教练的那些年里，我参加或者引导过不计其数的回顾。有些人引导的时候，会一再重复某种模式，或好或坏。我也跟很多人聊过，他们觉得从回顾中看不到任何好处。究其原因，常常是或者准备不充分，或者引导者经验不足，或者是形式重复枯燥。虽然很多敏捷实践都容易出现类似的“初学者行为”，我还是对回顾的话题兴趣最大。

我觉得回顾是敏捷实践中极重要的一项，因为它体现出了敏捷宣言中列出的“响应变化”和“探寻更好的软件开发方法”的原则。而且，如果回顾做得好的话，它可以创造出立足点给一些站得住脚的改变。

InfoQ：这本书和其他有关回顾的书有什么不同？

Pat：讲回顾的书只有另外两本，我现在还会把它们推荐给别人看，因为它们做入门指导是非常棒的，而且书里面还提供了丰富的练习，团队可以吸收借鉴到自己的环境。

《回顾手册》这本书的重点是帮助团队让回顾变得更加高效。我见过许多团队在同样的问题上陷足泥潭，这本书可以帮助解决这些问题。我会把引导者的经验分享出来，让读者可以深入了解其他团队准备、运行、引导回顾的不同方式，书中列举的小技巧也有助于让回顾变得焕然一新。

我知道，很多团队所处的环境离敏捷方法所推崇的小团队、同地开发相距甚远，他们很希望了解怎么调整回顾形式，让它能在分布式团队或者大型团队中运作。这类问题在 Retrospectives 邮件组中经常出现，在我们公司内部的邮件组中出现得更加频繁。我希望能够给大家提供一个资源，在这里能够读到所有的推荐方案、技巧、工具，帮助各种回顾“形式”的顺利开展。

InfoQ：在你引导回顾的时候，遇到过什么普遍性的问题么？

Pat：问题有很多啊。我遇到过的一个最大的问题是团队找不到独立的引导者，只能从团队里面找一个人出来引导回顾。这就会有潜在的利益冲突，因为引导者不再保持中立了。一方面来看，从团队中选人出来引导也是有好处的，因为引导者会对话题有足够的了解，能够问出比较尖锐的问题；但从另一方面看，大多数人在引导的时候都无法把自己的观点分离出来，所以谈话总是带有偏见的——这是我的亲身体会。

我也曾见过团队中的引导者大多数时候都是由比较权威的角色担任（如团队领导，项目经理，Scrum Master 等等）。因为这种关系的存在，团队有时候就不能畅所欲言，或者有意回避某个有争议的话题，避免挑战处于权威地位的人。

引导回顾时的另一个常见问题是引导者没有接受过良好的训练：谈话会被中途切断；问题可能没有得到回答；人们会说“做了再看”吧，可做的时候就脱离了当时的上下文，于是便得到了一个不一样的结果。幸运的是，引导的技巧比比皆是，相关的书籍也是随处可见。从团队中选引导者带来的问题，也可以通过各成员轮换的方式得到解决。但我还要强调一下，保证引导者具有良好的引导技巧，这件事情还是很重要的。

InfoQ：你在回顾中观察到的常见的坏味道有哪些？

Pat：常见的坏味道包括准备不足、一种活动一再重复、在重要话题上讨论不充分。

当我们看到引导者在回顾现场花时间布置会场，分发材料——笔和贴纸，我们就是知道他/她犯了准备不足的毛病了。很多人都没意识到，回顾会议跟其他会议一样，都是需要提前做准备的——比如整理材料和布置房间，在会议开始以后就要把精力从流程本身转移到谈话上。开始之前的准备工作会花些精力，但它会有所回报的。我自己倾向于在回顾之前花至少半个小时做准备工作。

如果一种活动一再重复，回顾就会变得枯燥乏味。稍稍改变点形式，或者改几个问题，就会出现意料之外的效果，因为这时候团队就会从另一个角度来看待当前的上下文了。另外两本回顾的书提供了大量的可选方案和资料，[敏捷回顾维基百科](#)上也有许多可以参考的点子。

我也见到过有的团队常常没怎么花时间讨论问题根源，就匆匆得出解决方案。没有完整了解事情真相之前，行动往往与事无益。所以大家还是应该在讨论不同方案的不同影响之前，先多花点时间收集事实，然后再去讨论孰优孰劣。

InfoQ：你对那些刚开始引导回顾的人有什么建议么？

Pat: 书里有整整一章是为这部分读者写的。我觉得从个人角度来讲有很多事情可以做，首先，如果对引导这件事情没什么概念的话，先去参加一下培训，或者至少读一本引导技巧相关的书。在工具箱里面多放几件引导工具，为应对各种场景做好准备。另外还可以去当一个旁观者，仔细观察其他人怎么做，问他们为什么这么做。旁观其他引导者跟结对编程很类似，双方都可以从角色互换中收益。你可以看到怎样做有效果，怎样做没效果，可以深入了解其他人在不同情况下的处理方式。当你问其他引导者为什么这样做的时候，也可以帮助对方把思考过程清晰地整理出来，这一点往往是他们从前没意识到的，在讨论过程中对方也会有很大收获。

作者简介

作者简介



Patrick Kua 在 ThoughtWorks 工作，富有激情与活力，是一名通才型专家，不喜欢被束缚。他一直在带领技术团队，常常培训团队和组织实施敏捷和精益方法，偶尔也会引导人们走出逆境。Patrick 热衷于学习和持续改进，也会帮助他人燃起学习和持续改进的激情。

评论者简介



Anand Vishwanath (Twitter 帐号为 @anand003) 是一名敏捷教练和项目经理。他 2002 年以 Java 和 .NET 程序员的身份，在 ThoughtWorks 开始了职业生涯。Anand 为各个国家的客户提供敏捷咨询和实施服务。他也在 [个人博客](#) 上记录实战经验。

推荐文章 | Article

SoundCloud 研发团队 Sean Treadway 谈 SoundCloud 架构演变

作者 郑柯

[SoundCloud](#) 是一个新兴的社会化音乐创建和分享平台，前不久，他们研发团队的 Sean Treadway 在 SoundCloud 的博客上谈到了 [SoundCloud 的架构演变](#)。

Sean 开门见山指出：

扩展是一个奢侈的问题，它与组织架构的关系远超与具体技术实现的关系。在每个变化阶段，我们都会预测用户的下一个数量级，从数千开始，我们的设计现在支持数亿用户。我们识别出瓶颈，解决它们的方法也很简单：在基础设施中加入明确的集成点，并以分而治之的方式处理各个问题。

识别扩展点，将其转为一系列更小的问题；有良好定义的集成点；这些方法让我们能够以有机而系统的方式增长。

产品初期

SoundCloud 最开始的架构简单而直接：

互联网->Web 层 (Apache) ->应用层 (Rails) ->数据层 (MySQL)

Apache 支持图片、风格和行为资源，由 MySQL 支持的 Rails 提供一个环境，几乎所有的产品都在其中有 model，可以快速完成路由和呈现。大多数团队成员都可以理解这种模型，交付的产品与我们现在的产品很类似。

我们有意没有在这时处理高可用问题，也知道到时可能面临哪些问题。此时我们版本脱离 private beta，面向公众发布了 SoundCloud。

我们的主要成本是机会成本，只要是阻碍了我们开发 SoundCloud 产品理念的东西，都被规避了。

在早期，我们有意确保构建的不仅是一个产品，而是一个平台。从一开始，我们的[公开 API](#)与网站同步开发。现在，我们与第三方集成者使用的 API 完全相同，并以之[推动网站开发](#)。

后来，SoundCloud 用 Nginx 替换了 Apache，主要原因有两个：

Rails 应用服务器运行在多个主机之上，而 Apache 处理多个虚拟主机的配置和路由很繁琐，特别是要保持开发和生产环境之间的同步时；

为了更好地提供连接池和基于内容的路由配置，以便管理、分配接收到的 web 请求、缓存向外的响应，并可以空出一个应用服务器，尽快处理后续请求。

负载分布与排队理论

接下来，SoundCloud 发现有些负载耗去的时间比其他要多几百个毫秒，一些较快的请求必须要等较慢的请求处理完成。2008 年，他们研发架构时，Rails 和 ActiveRecord 中的

并行请求处理还不够成熟。他们也开发了一些并行请求处理的代码，但是为了不占用更多时间去检查依赖，他们使用的方法是：

在每个应用服务器进程上运行一个并行进程，并在每个主机上运行多个进程。

Sean 接下来用到了排队理论中的肯德尔记录法（Kendall's notation）。

我们从 web 服务器向应用服务器发送一个请求，这个请求过程可以建模为一个 M/M/1 队列。该队列的响应时间由之前所有的请求决定，因此，如果大幅提升一个请求的平均处理时间，那么平均响应时间也会大幅提升。

由于当时仍然处于机会成本最高的阶段，所以 Sean 和他的团队决定：在继续开发产品的同时，用更好的请求分发方法来解决这个问题。

我们研究了 Phusion 旅客方法，也就是在每个主机上使用多个子进程，可是认为这样可能很快会在每个子进程上填满长时间运行的请求。这就像有多个队列，每个队列上有几个工作者，模拟在单个监听端口上的并发请求处理。

这就把 M/M/1 队列模型变成了 M/M/c 队列模型，c 是子进程数目。

该模型类似于银行中使用的排号系统。

该模型降低的响应时间由 c 决定，如果有 5 个子进程，对缓慢请求的处理速度能快 5 倍。但是，我们当时已经预计未来几个月用户会有 10 倍增长，而且每个主机上的处理能力有限，因此，只加入 5 到 10 个工作者并不足以解决排队阻塞问题。

我们希望系统中不要有等待队列，如果有，队列中的等待时间也要降到最低。如果将 M/M/c 用到极致，我们自问：“如何才能让 c 尽可能大？”

为了达到该目的，我们需要确保单个 Rails 应用服务器每次绝不接收超过 1 个请求，TCP 方式的负载均衡就此出局，因为 TCP 无法区分 HTTP 请求和响应。我们也要保证：如果所有的应用服务器都是忙碌状态，请求将会被排队到下一个可用的应用服务器中。这就意味着我们必须保证所有的服务器做到完全无状态。当时，我们做到了后者，但未实现前者。我们在基础设施中加入了 HAProxy，将每个后端配置的最大连接数为 1，并在所有主机中加入了后端进程，保证 [M/M/c](#) 在等待时间上的减少，生成 HTTP 请求队列，当任何主机上的后端进程可以处理时再发送过去。

将 HAProxy 作为队列负载均衡器，Sean 他们就可以把其他组件中复杂的队列设计推到请求管道中处理。此时架构如下图：



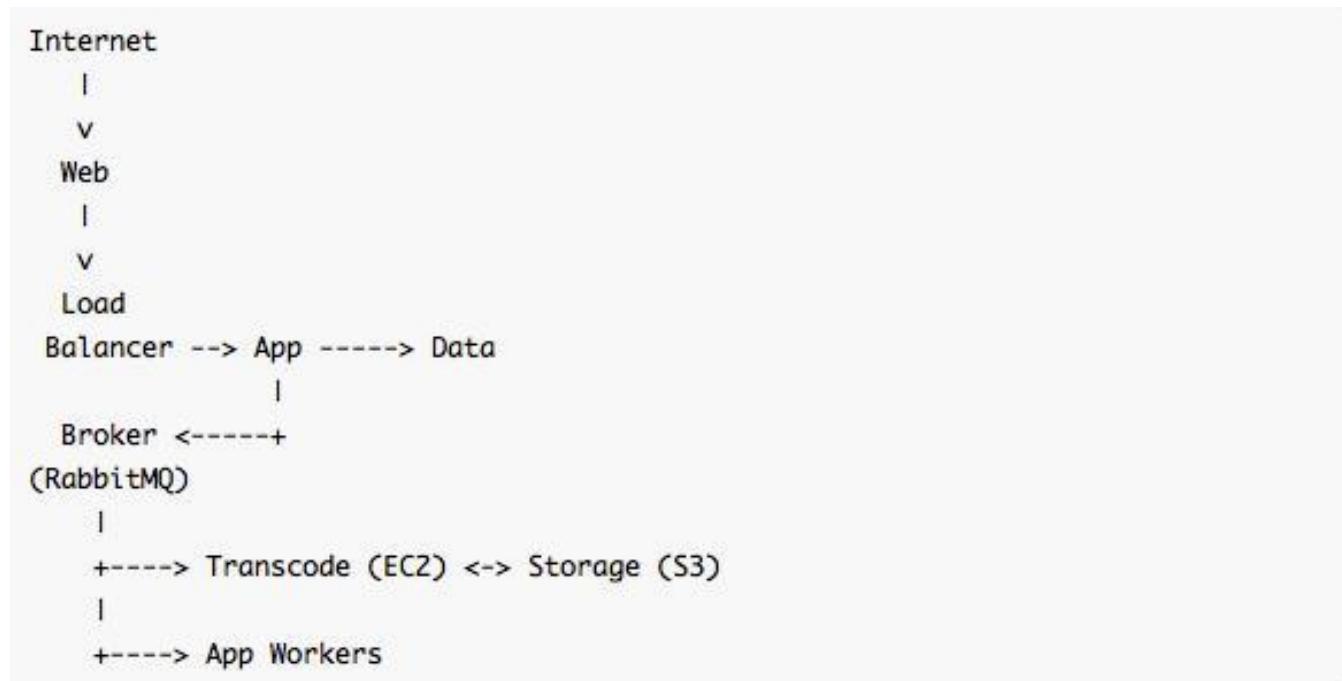
Sean 全力推荐 [Neil J. Gunther](#) 的书籍《[使用 Perl::PDQ 分析计算机系统性能](#)》，帮助大家复习排队理论，更多了解如何针对 HTTP 请求队列系统进行建模和度量，并可以深入到磁盘控制器的层面。

走向异步

为了解决用户通知和存储增长方面的问题，SoundCloud 决定加入中间层，以有效地解决工作队列的失败处理问题。最后他们选择了 AMQP，因为它提供可编程的拓扑，由 RabbitMQ 实现。

为了不修改网站中的业务逻辑，我们调整了 Rails 环境，并为每个队列构建了一个轻量级的分发器。队列的命名空间描述了预估的工作次数，这在异步工作者中创建起一个优先级系统，同时不需要向 broker 中加入信息优先级的处理复杂性，因为每一类工作的分发器进程只处理该类工作中的多个队列。应用服务器中的绝大多数异步工作队列，其命名空间中要么有 “interactive”（工作时间小于 250ms），要么是 “batch”（任何工作时间）。其他命名空间被用于特定应用。

此时他们的架构图如下：



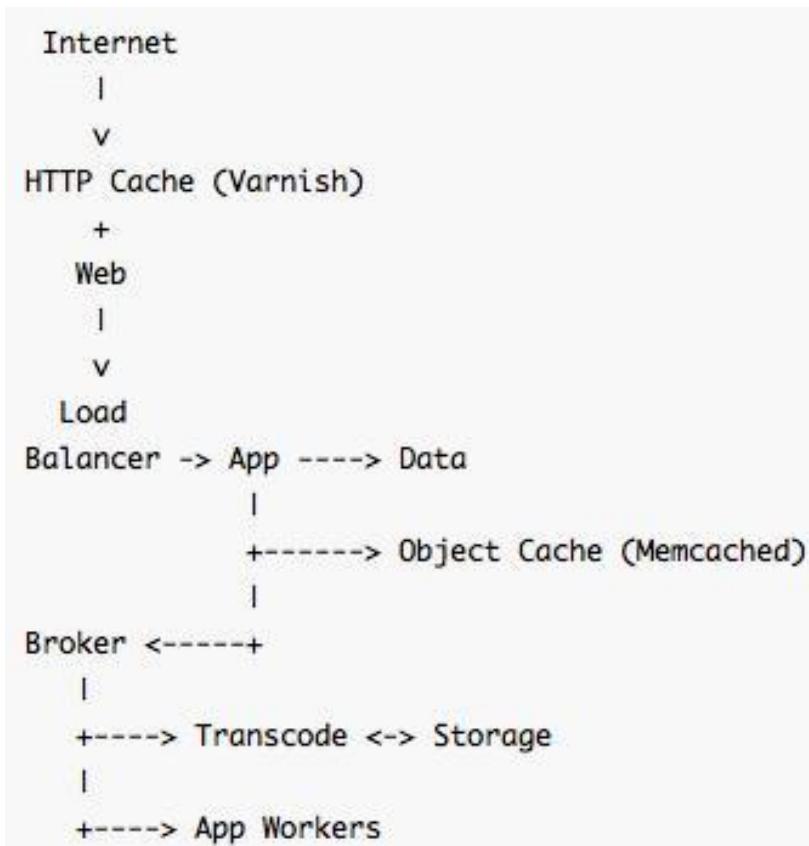
缓存

当用户达到十万数量级时，Sean 发现应用层占用了过多 CPU，主要用在呈现引擎和 Ruby 运行时上。

不过，他们没有用 Memcached，而是缓存了大量 DOM 片段和完整页面。由此引发的失效问题，他们通过维护缓存主键的反向索引解决，使用 Memcached，当应用中的 model 发生改变从而导致失效时，同样需要该方法解决。

我们最高的海量请求，来自于某个特定的服务，它向页面微件（widget）交付数据。我们在 Nginx 中为该服务创建了特定路由，并为其加入代理缓存。不过我们希望缓存功能能够通用化，做到任何服务都能创建正确的 HTTP/1.1 缓存控制头，并可以由我们控制的某个中介正确处理。现在，我们的[微件内容](#)完全由公开 API 提供。

此后，为了处理后端部分呈现模板缓存和大多数只读 API 响应，他们加入了 Memcached，并在很晚之后加入了[Varnish](#)。此时，他们的架构是：



通用化

SoundCloud 后来的处理模型变成：针对某个领域 model，为其状态设定连续处理方式，以便处理后续状态。

为了通用化这个模式，他们利用了 ActiveRecord 的 after-save 钩子，加入了 ModelBroadcast 方式。原则是：当业务领域变化时，事件会丢入 AMQP 总线中，任何对此变化感兴趣的异步客户端会得到该变化。

把写路径从阅读者中解耦出来，这种技术容纳了我们从未想到过的集成点，为未来的增长和演化提供了更大空间。

以下是示例代码：

```
after_create do |r| broker.publish("models", "create.#{r.class.name}", r.attributes.to_json) end after_save do |r| broker.publish("models", "save.#{r.class.name}", r.changes.to_json) end after_destroy do |r| broker.publish("models", "destroy.#{r.class.name}", r.attributes.to_json) end
```

Dashboard 功能

数据的快速增长引出了 Dashboard 功能。在 SoundCloud 中，用户可以在 Dashboard 中看到个人和的社会化活动索引，并可以个人化来自自己关注的人制作的音乐片段。 SoundCloud 一直受困于 Dashboard 组件带来的存储和访问问题。

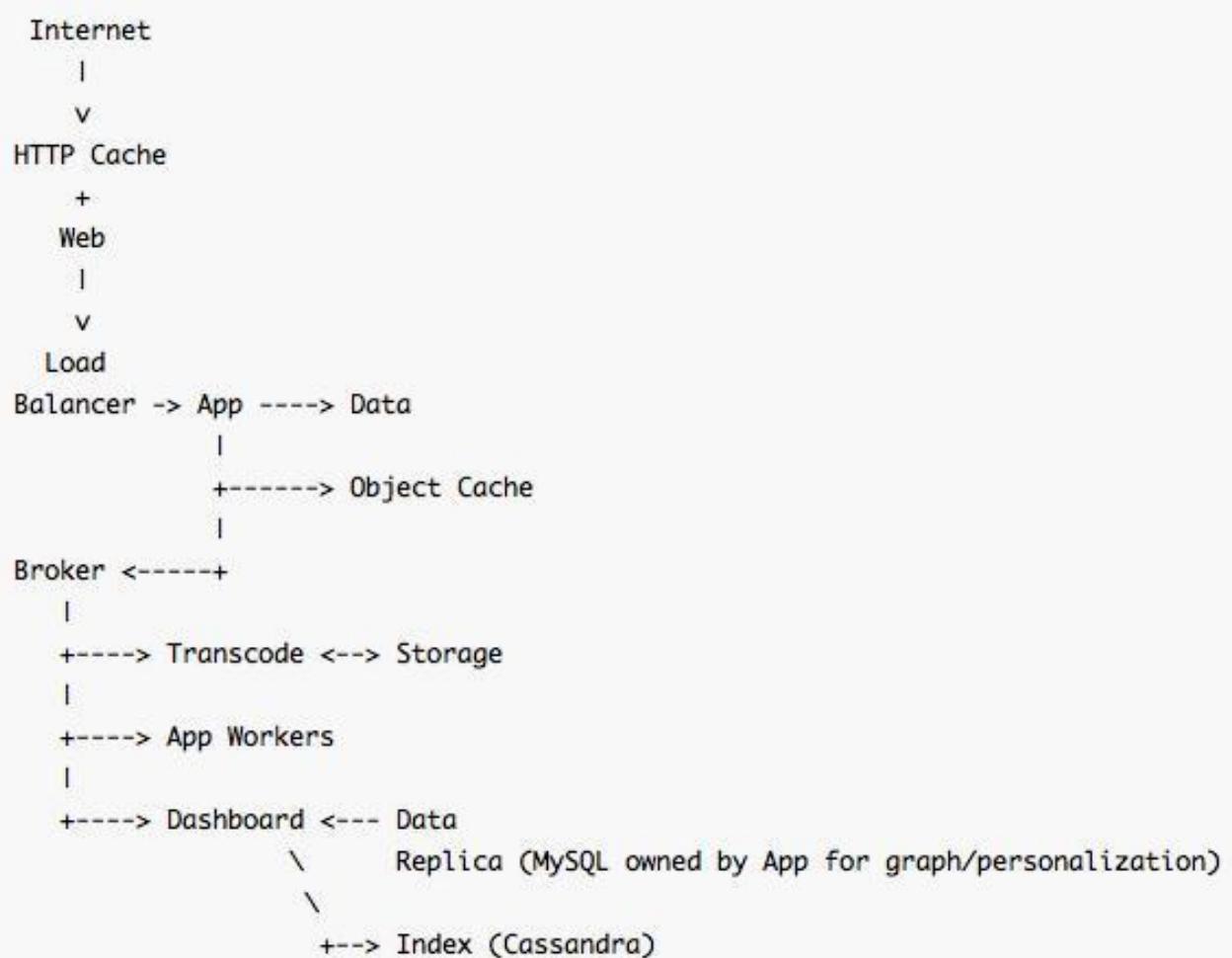
读写的路径各自不同，读操作路径需要针对每个用户提供一定时间范围内的顺序读并做优化，写操作路径需要对任意访问做优化，而且一个事件就有可能影响几百万用户的索引。

解决方法是重新排序任意读操作，将其变为顺序方式，并以顺序格式存储供未来的读取使用，这可能会扩展到多个主机上。排序字符串表非常适合用持久化格式，考虑到需要自由分区和扩展，我们选择了 Cassandra 存储 Dashboard 的索引。

我们从 model 广播开始，然后用 RabbitMQ 做队列完成步骤处理，主要包括三步：扇出 (fan-out)、个性化、指向领域模型的外键引用串行化。

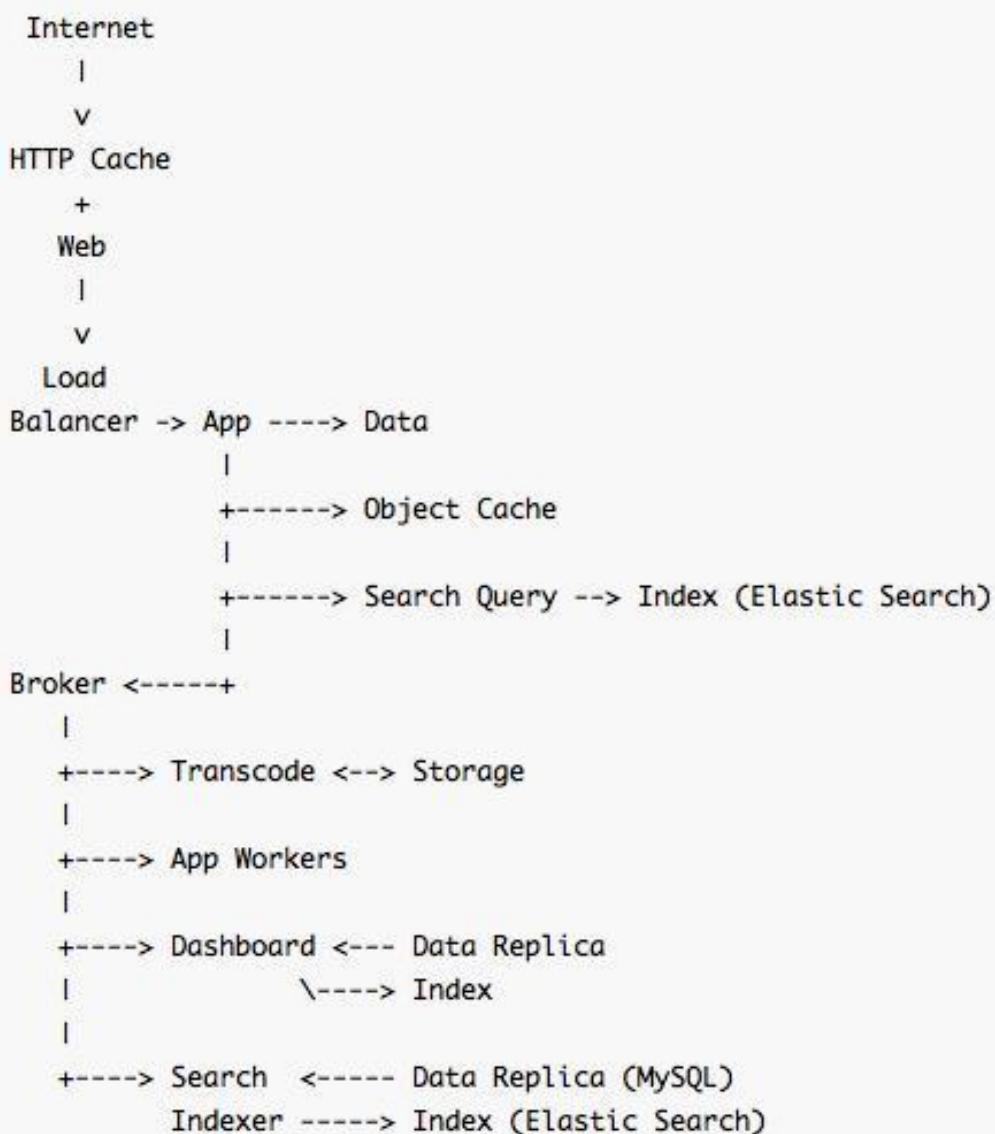
- 扇出会找出一个活动应该传播到的社会化图谱中的区域。
- 个性化查看发起者和目的用户之间的关系，以及其他注解或过滤索引项目的信号。
- 串行化把 Cassandra 中的索引条目持久化，供以后查找使用，并与领域模型做联接，以供显示或 API 展示。

此时他们的架构如下：



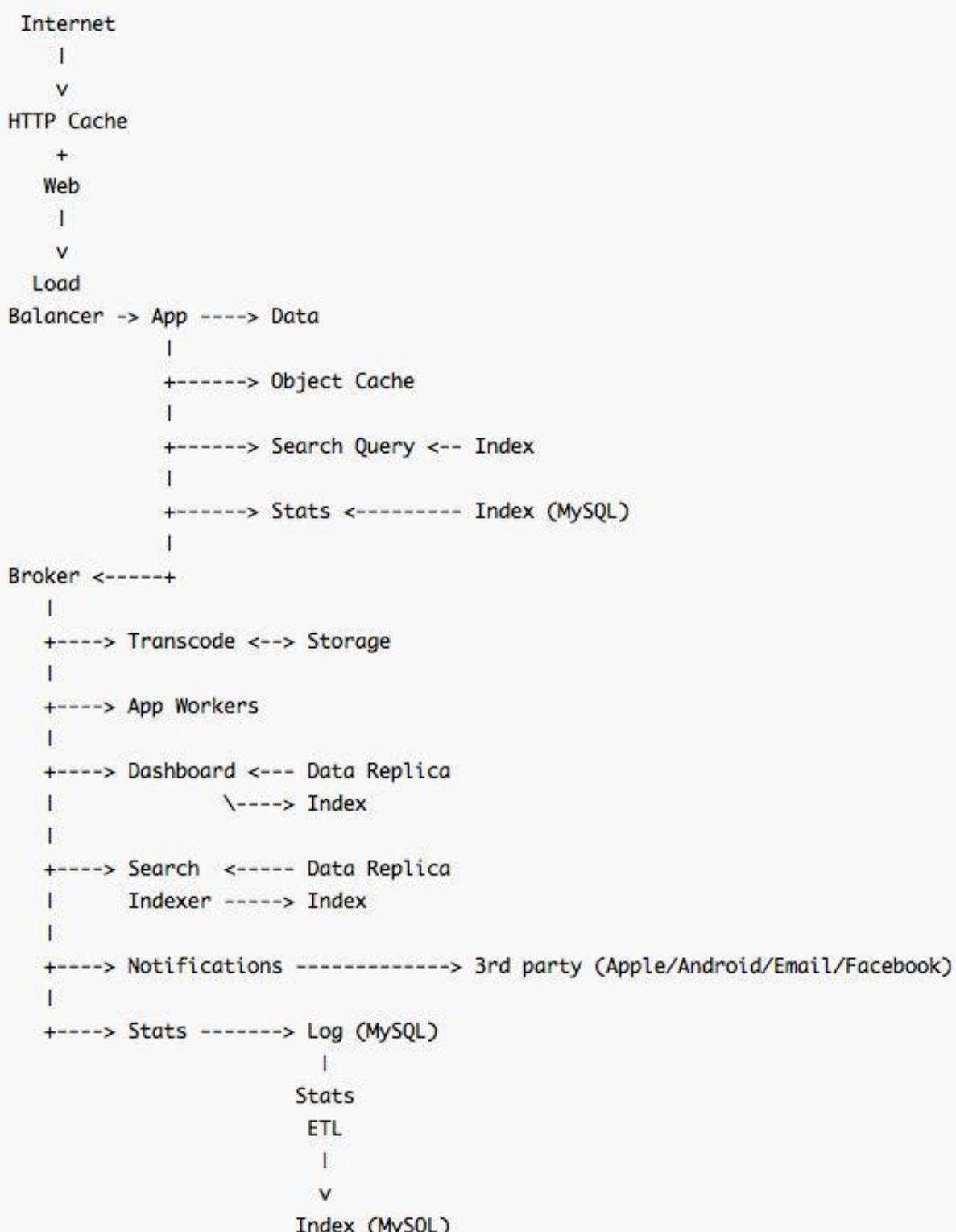
搜索

SoundCloud 的搜索通过 HTTP 接口暴露数据集操作子集，供查询使用。索引的更新与 Dashboard 类似，通过 ModelBroadcast 完成，并使用了由 [Elastic Search](#) 管理的索引存储，在复制数据库方面有提升。



通知和状态

为确保用户得到 Dashboard 的通知，SoundCloud 在 Dashboard 的工作流中加入了一个阶段，用来接收 Dashboard 索引更新的消息。Agent 可以通过消息总线得到路由到它们自己的 AMQP 队列中的事件完成通知。他们的状态和统计通过 broker 中介集成，但是没有 ModelBroadcast，他们会发出在日志的队列中的特定领域事件，然后保存在隔离的数据库集群中，以满足不同时间段快速访问需要。



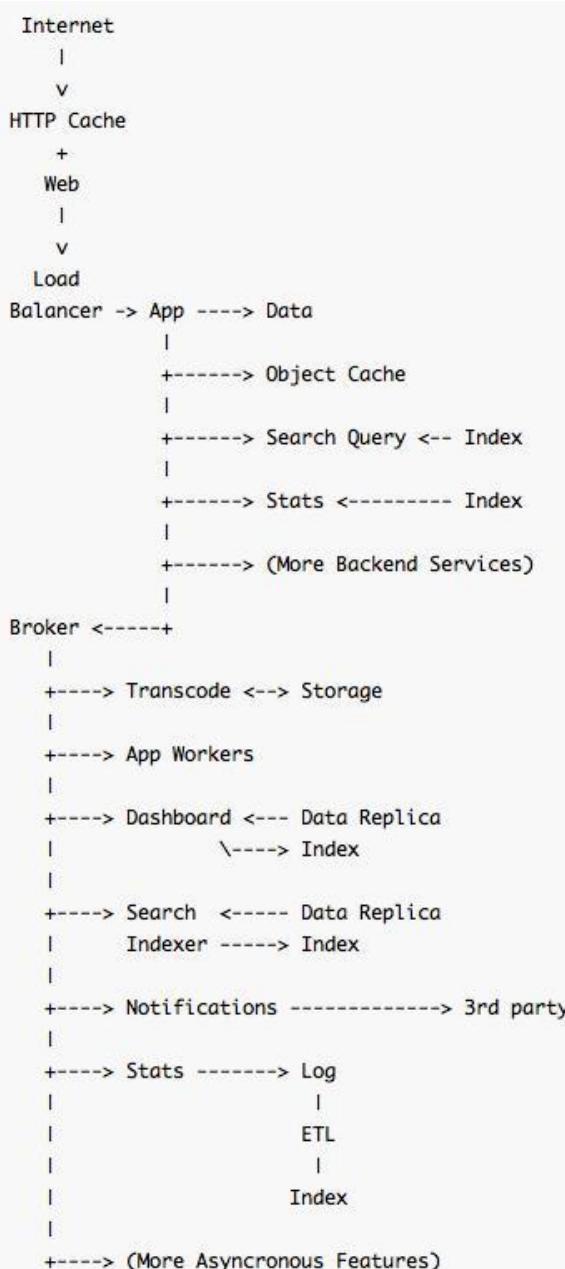
未来预期

Sean 这样描述他们对未来的规划：

我们已经建立起了明确的集成点，包括供异步写路径的 broker 中介中，包括向后端服务完成同步读和写的路径的应用中。

随着时间演变，应用服务器的数据库已经承担了集成和功能两方面的职责。产品开发基本尘埃落定，我们现在有信心将功能与集成分开，转移到后端服务中，供应用层还有其他后端服务使用，各自都可以在持久化层中有自己的命名空间。

我们在 SoundCloud 的研发方式，是识别扩展点，然后分别隔离、优化读路径和写路径，并预估下一个成长阶段的数量级。



最后，Sean 指出了 SoundCloud 以前和现在在架构上的约束：

在产品开发开始阶段，我们读写扩展的限制来自消费者的关注和开发人员的时间。现在，我们的工程方向是 I/O、网络和 CPU 的限制。

原文链接：<http://www.infoq.com/cn/articles/soundcloud-Architecture-Evolution>

架构师

www.infoq.com/cn/architect

每月8号出版

时刻关注软件开发领域的变化与创新

架构师

11月 ARCHITECT

特别专题
光棍节狂欢的背后——
电商系统深探
1号店B2C电商系统深造之路
百万点推荐引擎——从需求到架构
麦当劳购物系统浅谈分享
REST的远程API设计案例
大型Rails与VoIP系统架构
与部署实践
什么是Node.js
扩展Oozie
浅谈dojox中的一些小工具

InfoQ 每月8号出版

时刻关注企业软件开发领域的变化与创新

架构师

10月 ARCHITECT

特别专题
大数据时代
大数据
大数据时代的数据管理
阿里巴巴数据架构设计经验与挑战
大数据时代的创新者们
关系数据库还是NoSQL数据库
向Java开发者介绍Scala
HTML 5 or Silverlight?
解析JDK 7的Garbage-First收集器
了解云计算的基础

Steve Jobs
1955-2011

InfoQ 每月8号出版

时刻关注企业软件开发领域的变化与创新

架构师

9月 ARCHITECT

特别专题
QCon全球企业大会精华点滴
QCon在中国的三年回顾
麦肯锡对阿里巴巴国际站架构演进
精讲Apache和大数据流
新浪微博团队建设的虚与实
沐泽宁谈主数据架构
跟着李树学Oozie
如何查看我的订单—
REST的远程API设计案例
通用系统思考，走上改善之路
Redis内存使用优化与存储

InfoQ 每月8号出版

时刻关注企业软件开发领域的变化与创新

架构师

8月 ARCHITECT

特别专题
云计算的安全风险
圆桌会议：云计算的安全风险
设计一种云级别的身份认证结构
云应用和平台的现状：
云采纳者如是说...

Java虚拟机家族考
专家视角看IT转型
为什么使用 Redis及高产品定位
架构演化之谜

InfoQ 每月8号出版

时刻关注企业软件开发领域的变化与创新

架构师

7月 ARCHITECT

特别专题
深入理解Node.js
为什么要用后端工程语言Node.js
虚拟研讨会：Node.js生态系统之
概览、棒、最佳实践
使用JavaScript和Node.js构建Web应用
Node.js的起源和实践应用—
专访Node.js创始人Ryan Dahl

Java深度剖析十：Java对集群划分与Hadoop
将数据打散之一：关于横向的数据设计
分布式平台的前世今生
构建复杂的消息机制
来自Padmanabha的真实声音

InfoQ 每月8号出版

新品推荐 | News

WiX——强大的.NET 部署工具

作者 [Roopesh Shenoy](#) 译者 [滕云](#)

 Visual Studio Visual Studio 2012 去除了“VS Setup”，取而代之以开源的 WiX 工具包来创建安装包。

原文链接：<http://www.infoq.com/cn/news/2012/09/wix-net>

即将来临的 Rails 4.0 将放弃 Ruby 1.8 支持，改进后台任务、缓存等多项内容

作者 [Mirko Stocker](#) 译者 [丁雪丰](#)



即将来临的 Rails 4.0 将放弃 Ruby 1.8 支持，提供了多项新特性。其中最重要的是支持针对 mass-assignment 的强参数、针对后台任务的新队列以及缓存方面的改进。

原文链接：<http://www.infoq.com/cn/news/2012/09/rails-40>

微软随.NET 4.5 发布新 REST API 框架

作者 [Richard Seroter](#) 译者 [区志为](#)

在最近发布的 Visual Studio 2012 及.NET 4.5 中，微软正式推出新的网络服务框架 ASP.NET Web API。作为 ASP.NET MVC 4 的一部分，ASP.NET Web API 这套开源框架的设计目的是简化 RESTful 服务的开发和使用。

原文链接：<http://www.infoq.com/cn/news/2012/09/rest-web-api>

基于 Web 的代码编辑器 ACE，发布 1.0 版本

作者 Abel Avram 译者 雷慈祥

基于 Web 的嵌入式开源代码编辑器 ACE 1.0 版本已发布，该版本支持超大文件的编辑，45 种语言的高亮语法，TextMate 主题，Emacs 和 Vi 风格的按键设置以及其他特性。

原文链接：<http://www.infoq.com/cn/news/2012/09/ACE>

Embarcadero 更新 Delphi 和 C++ Builder，发布 HTML5 Builder

作者 Michael Floyd 译者 巨泽建



Embarcadero Technologies 推出其开发工具产品线的重大更新，包括 RAD Studio XE3，Delphi XE3 和 C++Builder XE3。该公司还推出了面向移动和 web 应用开发者的 HTML5 Builder。

InfoQ 采访了 Embarcadero 的产品管理主管 John Thomas。

原文链接：<http://www.infoq.com/cn/news/2012/09/Delphi>

用 PostSharp 对.NET 做死锁检测

作者 [Jonathan Allen](#) 译者 [区志为](#)

AOP 框架 PostSharp 的开发公司 SharpCrafters 开发了一款即插即用的死锁检测工具包。只要在项目中增加一行代码，这个工具包就可以对 Mutex、Monitor、
ReaderWriterLock 等大部分标准的基本锁机制进行死锁检测。

原文链接：<http://www.infoq.com/cn/news/2012/09/PostSharp-Deadlock>

OpenXava 4.5 支持 JPA 继承映射和自动化业务逻辑

作者 [Sriini Penchikala](#) 译者 [臧秀涛](#)

OpenXava 是一个支持快速企业级应用开发的 Java 框架，其最新版本支持所有的 JPA 继承映射策略和自动化业务逻辑（Automated Business Logic，ABL）库。OpenXava 4.5 版本已于 7 月份发布。

原文链接：<http://www.infoq.com/cn/news/2012/09/openxava-4.5>

Google Cloud Messaging for Android (GCM)已推出，将取代 C2DM 框架

作者 [Daniel Rubio](#) 译者 [雷慈祥](#)

 Google 已经发布了 Google Cloud Messaging for Android，该

服务对已被废弃的云到端消息框架(C2DM)做出改进，取而代之的服务无配额限制、无需注册，并提供了一套更丰富的全新接口。

原文链接：<http://www.infoq.com/cn/news/2012/09/GoogleCMReplacesC2Dm>

前 Sun 开发人员为 Android , iOS 等其他移动平台提供 JAVA 的 WORA 支持

作者 [Victor Grazi](#) 译者 [方盛](#)



2012 年，在以色列出现了一个名为 Codename one 的公司，该公司旨在生成一种新的 Java SDK，该 SDK 将允许 Java 开发人员通过一个单一的代码库就能为包括 iOS , Android , BlackBerry 和 Windows Phone 等一系列的移动设备编写本地应用。对于 iOS , Codename one 通过自己的云服务器先将 Java 代码转换成 C 或者 Objective C 代码，然后再将转换后的源代码编译成本地应用程序。这使得利用 Java 来编写 iTunes 兼容应用成为可能。

原文链接：<http://www.infoq.com/cn/news/2012/09/codenamene>

Yeoman : 构建漂亮 Web 应用的工具和框架

作者 [侯伯薇](#)



Yeoman 是由 Paul Irish、Addy Osmani、Sindre Sorhus、Mickael Daniel、Eric Bidelman 和 Yeoman 社区共同开发的一个项目。它旨

在为开发者提供一系列健壮的工具、程序库和工作流，帮助他们快速构建出漂亮、引人注目的 Web 应用。

原文链接：<http://www.infoq.com/cn/news/2012/09/yeoman>

HTML5 Boilerplate 4：改进了 Apache 配置和图片替换技术，并采用 MIT 许可证

作者 [Bienvenido David III](#) 译者 [雷慈祥](#)

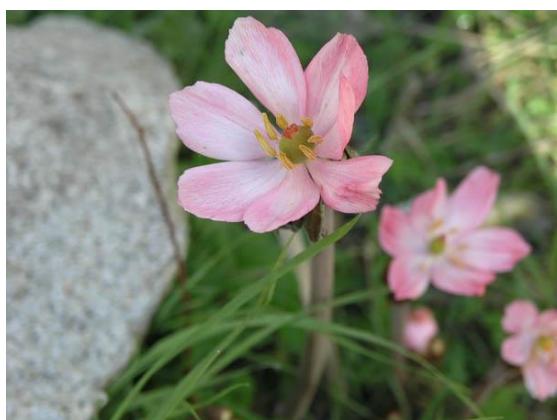


HTML5 Boilerplate (H5BP) 为 HTML5 和 CSS 开发提供了原始模板。它包含了一套有关 Web 前端开发的最佳实践，其中包括 Web 服务器的推荐设置，在新版本中的改变有：改进了 Apache 的压缩配置、HiDPI 设备检测、随机文档、MIT 许可、console.log 保护、中性色作为选中颜色以及图片替换 (IR , image replacement) 新技术等。

原文链接：<http://www.infoq.com/cn/news/2012/09/html5-boilerplate-4>

封面植物

桃儿七



“桃儿七”属于“太白七药”之一，具有神奇的抗癌作用。以“桃儿七”为主药制成的“天福星”Ⅲ号抗癌药，对于乳腺癌的治疗效果尤为明显，临床有效率达88.5%。“桃儿七”为小檗科植物桃儿七（小叶莲）。*Sinopodophyllum extnedium*(Wall.)Ying 的根及根状茎。多年生草本植物，生于海拔2800~3300米的山林下或草丛中。

其性辛、温，味苦，有毒。具有祛风除湿、活血化淤、化痰止咳、解毒的功效。

形状特征：多年生草本，高40—80厘米；根状茎粗壮，横走，通常多少结节状；不定根多数，长达30厘米以上，直径2—3毫米，红褐色或淡褐色；茎直立，基部具抱茎的鳞片，上部有2（—3）叶。叶具长达30厘米的柄；叶片轮廓心脏形，长13—20厘米，宽16—30厘米，3或5深裂几达基部，顶生裂片3浅裂，侧生裂片2中裂，小裂片先端渐尖，边缘具尖锯齿，下面被柔毛。花两性，6基数，单生茎端，先叶开放；萼片早落；花瓣粉红色，倒卵状长圆形，外轮大，内轮较小；雄蕊长约9毫米，花药线形，长约4毫米，具四合花粉；子房1室，生多数胚珠。浆果卵圆形，熟时红色，长5.5—6.5厘米，直径约3.5厘米；种子多数。

保护价值：桃儿七的根茎与果实均有较高的药用价值。同时也是东亚和北美植物区系中的一个洲际间断分布的物种，对研究东亚、北美植物区系有一定的科学价值。

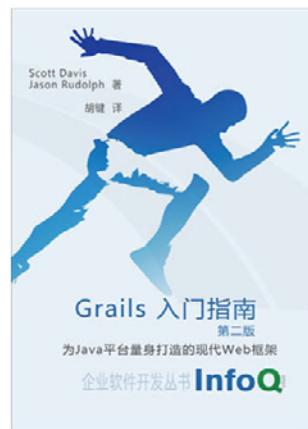
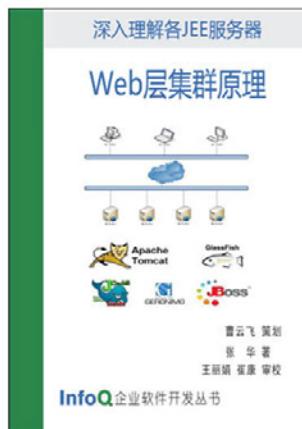
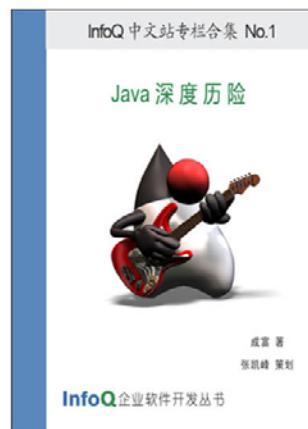
1kg.org 多背一公斤

爱自然 | 更爱孩子



InfoQ 软件开发丛书

欢迎免费下载



商务合作: sales@cn.infoq.com

读者反馈/内容提供: editors@cn.infoq.com



架构师 10 月刊

每月 8 日出版

本期主编：张龙

美术、流程编辑：水羽哲

总编辑：贾国清 发行人：霍泰稳

读者反馈：editors@cn.infoq.com

投稿：editors@cn.infoq.com

InfoQ 中文站新浪微博：

<http://weibo.com/infoqchina>



本期主编：InfoQ 中文站翻译团队编辑，张龙

张龙，天津大学工学学士，同济大学软件工程硕士，混迹于外企多年。热衷于编程，乐于分享，对新技术有强烈的探索欲，对 Java 轻量级框架有一定研究，工作流领域研究者，苹果产品重度痴迷者。目前对 Mac、iPhone/iPad、Android、Windows Phone RIA 开发、动态语言及产生了浓厚兴趣。翻译出版了《设计原本：计算机科学巨匠 Frederick P. Brooks 的思考》等书籍，资深培训讲师。