

架构师

4月 ARCHITECT



特别专题

构建高效能团队

全功能团队 - 数据篇

Facebook王淮谈科技公司工具文化

自动化构建：一致性关键之道

微软Kinect for Windows

Kinect for Windows SDK v1.发布

Kinect for Windows培训北京站

Kinect for Windows样本代码开放

开发者需要了解的WebKit

聊聊并发：JAVA线程池的分析和使用

TeamToy2的开发故事

深入理解Java内存模型（七）





促进软件开发领域知识与创新的传播



中文 | 英文 | 日文 | 葡文 |

本期主编

贾国清

总编辑

贾国清

美术/流程编辑

水羽哲

发行人

霍泰稳

读者反馈

editors@cn.infoq.com

商务合作

Sales@cn.infoq.com

15810407783

卷首语

If you are trying to decide among a few people to fill a position, hire the best writer.

- via Rework, By 37Signals

3月，InfoQ 在 Mozilla 办公室组织了第一期 [TWERC](#)(Tech Writers, Editors and Reporters in China)训练营线下活动。 TWERC 训练营是以培养中文界技术作者为目的的线下聚会活动。训练内容包括但不限于：国内各个技术媒体的背景与稿件需求介绍、技术文章的写作规范与写作技巧、如何提高高质量的写作、如何做采访以及如何做内容的推广。每期活动，会邀请包含技术媒体的总编、编辑，已经在从事写作、翻译的技术作者，对中文技术作者感兴趣以及希望参与技术写作的工程师在内的嘉宾到场交流和分享经验。

InfoQ 希望通过这一开放的组织来发展、扩大国内技术作者人群。此外，在技术媒体之间、技术作者之间分享经验，帮助作者写出更好的内容，让“技术传播”这个职业更加专业化。我们也希望能够将 TWERC 作为中文界的 [Society of Technical Communications](#)。如果您对技术写作或 TWERC 感兴趣，欢迎加入我们的讨论组，[点击加入](#)。

本期架构师以【构建高效能团队】为主题，《打造 Facebook》作者王淮曾在其博客分享过关于软件开发原则、Facebook 新兵训练营和导师系统等内容，受到了读者广泛关注，本次摘录 InfoQ 的一篇有关[《科技公司应有的工具文化》](#)的内容，希望能对大家有所启发。继胡凯[建设全功能团队](#)和[《建设全功能团队——实践篇》](#)之后，该系列又添新篇[《全功能团队——数据篇》](#)，此文也被收录于本期架构师。对这个系列感兴趣的读者，欢迎观看胡凯在 QCon 上的视频分享[《建设全功能团队——故事两个半》](#)。

自去年 10 月 Kinect for Windows 在中国发布，微软随后启动“Kinect for Windows 加速器计划”之后，3月又启动了名为“码”上 Kinect 的全国培训活动，InfoQ 有幸参与了这次活动的报道。除本期选编的[《Kinect for Windows 培训北京站：了解 K4W 软硬件原理》](#)等三篇外，还可通过链接访问 InfoQ 网站的[《Kinect for Windows 培训营：自平衡机器人项目作者访谈》](#)。

目录

[人物专访]

Etsy 工程师 Sam Haskins 谈代码部署，监控与故障处理 7

[热点新闻]

从小型网站到超大规模网站的 MySQL 参考架构 16

58 同城开源其轻量级 Web 框架 Argo 20

QQ 安全中心的两小时 Bug：快速响应+快速部署的经典案例 22

Iron.io 从 Ruby 迁移到 Go：减少了 28 台服务器并避免了连锁故障 26

LinkedIn 实时低延迟数据抓取系统 Databus 开源 28

[特别专题]

全功能团队 - 数据篇 33

Facebook 元老王淮谈科技公司应有的工具文化 40

自动化构建：一致性关键之道 45

[本期专栏]

微软发布 Kinect for Windows SDK v1.7 51

微软开放 Kinect for Windows 样本代码 53

Kinect for Windows 培训北京站：了解 K4W 软硬件原理 54

[推荐文章]

深入理解 Java 内存模型（七）——总结 57

| | |
|---|----|
| 开发者需要了解的 WebKit | 65 |
| 聊聊并发（三）——JAVA 线程池的分析和使用 | 74 |
| TeamToy2 的开发故事：面向移动的 API 设计，代码重用，快速迭代 | 81 |

[新品推荐]

| | |
|--|----|
| 详解 Java 7 中新的文件 API | 85 |
| Spring for Apache Hadoop 1.0 发布 | 85 |
| Visual Studio 2012 Update 2 最终预览版发布 | 85 |
| 基于 GitHub 的 jQuery 插件资源库业已发布 | 85 |
| 微软和亚马逊在 Android 方面的最新消息 | 86 |
| Node.js v0.10 版本发布 | 86 |
| Lienzo 1.0：HTML5 Canvas 元素的 Java 版本场景图 API | 86 |
| Eclipse RAP 2.0 发布——首词相同，含义不同 | 87 |
| jQuery Mobile 1.3.0 发布 | 87 |
| Red Gate 发布 ASP.NET MVC 学习门户网站——Simple Web Dev | 87 |
| Google 即将发布 Go 语言 1.1 版，含多项重大更新 | 87 |
| Scrum Primer 网站发布卡通版本的 Scrum 总览图 | 88 |
| Visual Studio2012 代码审阅特性的对比、注释、意见及状态更新功能 | 88 |
| Eclipse Foundation 首次发布 Hudson | 88 |
| 基于 Canvas 的图表类库 Chart.js 0.1 发布 | 88 |
| Grunt 0.4.0 发布：更强调模块化 | 89 |
| Concurrent 发布 Lingual——一种用于 Hadoop 的领域专用语言 | 89 |

GCC 4.8 发布 , 完成向 C++的迁移 89

推荐编辑 | 翻译团队编辑张卫滨 90

[封面植物]

人物专访 | Interview

Etsy 工程师 Sam Haskins 谈代码部署，监控与故障处理

作者 Sai Yang 译者 乔梁

在本次访谈中，Sam Haskins 分享了他在 Etsy 做 DevOps 的经验。Etsy 使用 Git 做代码管理，平均每天提交的更新数量在 30 次左右，有时甚至达到 70 个。在此如此高频率提交代码的情况下，他们如何进行代码审查？如何在每次更新之后监控系统性能？如何处理故障？下面，Sam 将一一解答。

Sam Haskins 目前属于 Etsy 的核心平台团队，该团队一共有 10~15 人，整体负责服务器及系统层与应用层的接口，为上层开发特性的工程师们提供服务。Sam 于 2010 年中加入 Etsy，他现在主要负责数据库接入层，也在 web 框架上做一些东西。核心平台团队的其他成员还负责开发图片存储系统、异步任务管理等功能。Sam 毕业于卡内基梅隆大学的数学系。

代码部署和审计

InfoQ：第一个问题是关于部署系统的。你们使用哪些工具？

Sam：我们使用 Git 做代码版本管理，使用 Deployinator（我们为自己开发的一个工具）从 Git 中拿东西出来，并把它放到所有的服务器上。基本上就是文件复制工作，没有什么特别的。另外，我们使用 Chef 做配置管理。就这些了。实际上我们用 GitHub。我们有一个内部的 GitHub 来管理代码。

关于代码审计流程，实际上我们没有很多正式的检查。只要你的代码能编译，能运行，你就可以放到代码库上，不需要审计。但是，我们鼓励大家做代码复查，而且大家经常这么做。代码复查通常是这样的——你只要把你的补丁发给别人看，他们看过后给你一些点评。大多数人在推送代码到代码库前都会这么做。

InfoQ：有 QA 吗？

Sam：没有。我们没有传统意义上的 QA。因为我们每次的提交都很小，所以发现哪些代码破坏了系统通常是相当容易的。当然，也有一些人的工作有点儿像 QA，他们的工作就是试着破坏我们的系统，然后让我们知道哪些东西坏了。我们部署做得很多，一天部署 50 次挺平常的，有时候会更多。在部署前后做全面

的测试是不可能的，根本没有足够的时间。但是我们有自动化测试，和代码审查等等，而且我们每次的变更都很小，所以，我们认为我们并不需要有一个大规模的正式的 QA 过程。

InfoQ：那么，当你做更新时，你如何做环境控制，保证代码安全？

Sam：对于我们的工具 Deployinator 来说，只要你点一下按钮，它所做的就是从 Git 中拿出东西，并把它分发下去，仅此而已。

InfoQ：你们部署的是源代码，而不是打包后的二进制文件？

Sam：是的，我们不打包。它就是把它们放到 web 服务器的某个目录下之类的，没有什么太复杂的。我们不做太复杂的事儿。

非常简单，只要几分钟就运行完了。正是因为很容易，所以我们才能部署这么多次。

InfoQ：那么一天 50 次是比较典型的部署频率吗？

Sam：差不多。今年的平均值可能是 30 或 33 次，算上星期六和星期日。但通常星期六和星期日我们什么也不做。我想，最多的时候，我们可能在一天里我们最多部署 60 或者 70 次，少的时候也有二、三十次。的确很多，但这也是我们改变了写代码的方式，让它变成合理的事情了。假如我们每次都做很大的变更，这么做可能就不行啦，一定会变成一种灾难。正是因为我们每次只做很小的更改，所以这种方式才能行。

部署的流程与健康监控

InfoQ：由于你们每天部署这么多次，那么你们通常在一天中的什么时间部署？你是部署后，观察几分钟，然后再部署另一次吗？

Sam：是的，我们有很多度量项，可能成千上万吧。所以 每次部署后，除了要监控你的变更是否起作用了，我们还有一个标准的部署指示仪表盘，你需要关注。那上面有一堆图表，你要看着它，并确保仪表盘上的指标没有异常。

我们还有一个工具，用来读实时日志，你也要看看那个东西。假如有些东西不正常，那你就需要修复它。但只需要几分钟就行，通常是一两分钟。但我要看更多一些的图表数据。只要没有什么问题出现，并且你认为你的修改正常工作了，那就轮到下一组啦。

InfoQ : 当部署时，你是先把它部署到测试环境上吗？

Sam : 有一个前提，在你部署之前，你已经自行测试过它了，所以我们是直接部署的。不过，的确也可以说我们已经在测试环境上部署过了，因为我们的试运行环境也是我们的生产环境，是同一个环境，只是版本高一点儿，而且用户不可能访问到这些服务器，只有我们才能访问。

因此，当部署时，首先会放到试运行环境，然后 CI (持续集成服务器) 会运行测试，是单元测试。顺便说一下，我们使用的是 Jenkins。同时，你可以访问试运行环境，Jenkins 运行一系列的测试。之后，一旦推送代码的人认为试运行环境上的功能没有问题，他们已经测试过他们的变更，并认为这些变更运行正确的话，那么只是 Jenkins 运行完 (通常总共需要 5 分钟)，就可以推送到生产环境了。

InfoQ : 你在这里提到的生产环境，是指所有的机器吗？

Sam : 是的，是所有的机器。

这个工具只有两个按钮，一个指向试运行环境，另一个就指向生产环境。

InfoQ : 那是否根据服务的不同，分成了不同的集群呢？比如一种服务，使用一类集群？

Sam : 算是吧。所有的部署都是同样的软件栈。同样的 web 软件栈。但我们的确有搜索专用的软件栈和图片存储专用的软件栈，但那些软件栈也与它相关。我们并没有太多的 Service。我们只有一个代码库，所以它也非常大，大约 2GB 吧。但绝大部分都用同一个软件栈。

因些，当做部署时，先是 web 服务器，然后是 API 服务器。我们有内部的支持系统，异步任务，Gearman，以及类似于运行后台任务(Cron)的机器和相关设施，所有这些都是一次完成的。

InfoQ : 那么你会根据它们的重要程度做分别推送吗？你要知道，有些部署可能会对用户产生很大的影响。

Sam : 我们非常喜欢现在这种工作方式的原因，就是因为这种方式决定了你说的这种情况不会经常发生。如果真有什么事情非常重要的，那么你可以自己做部署，并确保更顺利。但是，通常来说，我们更喜欢鼓励大家以一种好的方式工作，以避免这种事情的发生。

InfoQ：也就是说，避免一次性做很大的变更，是吗？

Sam：是这样的。

InfoQ：所以，你们总是提交很小的变更？

Sam：当我们发布新功能时，我们会使用配置文件对功能进行控制，比如关闭这个功能，打开那个功能，因为很难一点儿一点儿的发布一个新功能。当需要发布某个功能时，我们再把开关打开。这种方法使得发布并不需要太多的精力，并且比较安全，我们可以做到只在公司内部发布，我们也可以做到只发布给属于 beta 群组中的用户，也能做到每次向百分之多少的用户发布。

所以当做了一点儿改动时，我们常常只发布给 1% 的用户，以验证没有破坏其它功能，然后再逐步发布给所有的用户，而不是一次性发布。如果发布的是一个新特性，我们更多的时候只是把开关打开。但有时我们只会将特性发布给一部分用户，以便观察用户在接下来的几天里是如何使用它的。我们会持续关注。

InfoQ：是不是说，beta 用户会被导向某个特定的集群？

Sam：节点都是一样的。我们只是在运行时检测，比如通过用户的 user ID。

InfoQ：你们是使用自己构建的系统做这件事呢？还是用其它的什么工具？

Sam：我们自己写的代码。它原来的版本是开源的，而这个新版本要比原来那个版本好得太多了，但是还没有发布，不过即将发布了。

这个项目的名字不怎么好，好像是叫做 Feature Flags 之类的。它在我们 Github 的主页里。

这就引出了另一个话题，也是我们开发中使用的方法。我们并不使用 Git 中的分支。大家并不会在一个分支上工作很长时间。通常一个分支的生命周期很短，很快就会把代码部署了。

有关故障

InfoQ：你们是如何处理故障的？

Sam：因为大家都被培训过如何看图表，如何查日志。通常我们发现故障是非常非常快的，因为一直有人在查看我们的那些度量项，这些度量项的确非常多。大家都非常了解这些度量项正常是应该是什么样子的。

InfoQ：那是一分钟查看一次，还是一秒钟？

Sam：我们使用 Nagios 做自动化的检查。每个检查的时间周期不同。有一些非常非常频繁，而有一些则不是。我们不但有很多这种 Nagios 做的自动化检查，还有一些人来看图表，以及对应的模式。这种图表的更新通常是一秒钟。

因为一直在不停地部署代码，所以不得不时刻查看那些图表。既然经常看着那些图表，那么对我们来说，快速地发现问题也就不是什么难事啦，因为一直不停地在做这件事，所以也就做得好啦。

InfoQ：听上去感觉好像很依赖人肉啊？

Sam：是这样的。但我们也像 Nagios 这样的工具来检查那些可能出错的地方。不能及时捕获故障的情况很少，几乎总是能马上发现问题。那些无法用 Nagios 无法检查的东西在半夜也不会出问题，因为没人会在大家都睡觉时去部署代码。那些出错的地方通常都是在系统级别上，而通常都是由 Nagios 来监控的。在白天我们工作时，因为有新的部署，这些地方才有可能出错。但这时会有人看着，因为他们正在部署代码。所以那些很难用计算机去监控的东西在晚上不会出错。

InfoQ：那么，当出现了一个问题时，你们会采取什么样的措施？

Sam：如果某次部署严重地影响了网站，我们所做的就是要求大家别再推送代码，并让整个生产线停下来，然后无论是谁发现的问题，他都要马上开始分析，尝试找出是哪里出的错。我们都可以访问那些日志和图表数据。当我们确定哪个特性出了问题时，我们会联系那些做这个特性的人或可能知道这个特性的人。通常情况下，如果与部署有关的话，相关的部署人员都已经就位了，因为我们时刻使用 IRC Chat。

大家都在这个聊天室中，所以如果是部署问题，正在做部署和观察监控的人就知道哪里出了错，很可能也已经知道如何修复它了，因为每次部署只做了很小的改动，所以他们可以将代码回滚，部署原来的那个版本。但对于比较小的缺陷，我们更倾向于推送新的修复代码，而不是回滚。就是直接修复好缺陷，再次部署。

InfoQ：那会花的时间较长吧。

Sam：是的。所以如果只是小问题，能很快修复，那就提交并部署新的代码。因为即使回滚的话，你也要再一次推送代码，总共也需要大约十分钟的时间。因为先要上试运行环境，然后才是生产环境。所以无论你是提交修复，还是回滚，花

的时间差不多。如果你多花五分钟 能够正确地修复问题，其实并不赖，甚至可能更好。

所以，通常我们会这么做。如果问题很严重，显然我们会先回滚，然后再找问题的根本原因。然而经常遇到的情况是你并不能完全确定回滚就是你想要的结果，所以与盲目回滚相比，我们更多的是试图找到真正的问题所在。

InfoQ：向前修复通常是以什么样的方式进行的呢？

Sam : 大多数在部署时遇到的问题并不会影响整个网站，它们只是会影响其中的一小部分。如果是向前修复的话，当你在修复这个问题时，其他人还可以继续推送他们的代码，继续做他们原来做的事情。但是，显然我们做这种决定要基于故障的严重程度。

InfoQ：修复不好，怎么办？

Sam : 如果不立即修复的话，就要开始做回滚操作。

InfoQ：比如说，花了十分钟以后，你也不能修复故障怎么办呢？

Sam : 我们并没有那种严格的用于快速决策的规则。但是如果大家不得不在那里等待你修复你的问题，那通常说明什么地方出了问题，其他人就不会部署代码。如果你耽误其他人部署代码太长时间的话，那么我们的做法是采取最快的方式，无论哪种方式都行，让大家可以部署代码。所以十分钟过去了你还没有找到问题，但你认为回滚会让工作回到正轨上的话，当然你就要先回滚，然后在你自己的机器上继续找问题。

InfoQ：能讲一下最近的一次故障吗？

Sam : 好。比较典型的情况是，我们只会遇到一些很小的问题。比如，有人正在修改一个页面，而此时我们网站上的一个卖家正在修改他有多少东西可以卖。我并不能说这是一个现在正在发生的案例，但是，它的确与我们看到的小故障类似。这个开发人员修改的恰好是这个页面，那么卖家可能就会看到一个错误，但这并不会耽误大家在网站上卖东西。这时我们会认为，问题并不大。这个开发人员去查看一下，然后修复它就行了。通常由于变更很小，所以直接修复，然后推送部署就行了。但要强调的是，如果问题比较严重，那就会回滚。

最近就发生了一个非常严重的问题 我们在数据库中使用完整的 IDs ,而这些 IDs 不是自增的。我们没有让数据库来做这种 ID 自增，因为我们所有的数据库都是

双主方式(two masters)的，即 master-master。所以无法使用自增，因为你不知道哪个 master 是最近一次做自增操作的。所以我们有另外一个服务器，由它来专门提供这些自增数字。这个服务器也有两个。其中一个只产生奇数，而另一个只产生偶数。这就是用户 ID 帐户的来源。

几个月前，这个服务器的数字超过了 2^{31} ，所以溢出了 32 位的整数列，但这也不是太大的问题，因为你可以把它保存在一个 64 位的列中。然而，在代码中的一些地方并没有这么做。但是，大家并没有意识到为什么 ID 突然出现了 64 位。因此，当 ID 太大时，一些相关的特性就无法工作了。

这些地方的代码无法得到合理的 ID，但仍旧试图把它保存到数据库上，但数据库无法保存它。当这个问题发生时，我们立刻要求大家不要再推送代码了。我们只花了大约一分钟就找到了这个故障的原因，因为在日志里有很多错误信息，这些信息中能看到那些数字。如果之前你看到过这样的数字，你会就发现这个问题。你能想像得到当看到这样的数字时，那种不祥的感觉。

我们看到后，我们就立刻停止推送，是我们的团队（core platform team）发现它的。尽管我们知道这是一个严重的问题，但我们并不知道有多少数据库表使用了它，因为没有地方可以看到这样的信息。我们召集大家，包括一些运维工程师和我们团队的一些人，到会议里来讨论如何处理这个问题。最后决定查看所有的表，看看到底哪些数据库表的列宽不足。

在数据库里，代码中，以及写的 schemas 中都可能有相关的信息。所以我们只能查看所有的代码，来判断有多少 schemas 中存在这个问题。有些只是代码中有问题，有些只是在数据库中有问题。有些地方则是两方面都有问题。同时，其他人来判断哪些特性受到了牵连。我们从错误日志中有可能发现这些受牵连的特性，也可能从图表上发现。只要发现了，就可以找以相应的配置文件，把这些特性关掉。

一旦我们修改好所有的地方，我们就可以在数据库上运行这些变更，将那些字段改成 64 位，再把关掉的特性开关打开，然后马上监控，看哪些数据还会出异常。整个过程只持续了几个小时，而且只有几个比较小的特性受到了影响。但直到把全部问题都修复了，我们才能知道到底有多少特性受到了影响。因为很多影响不是显而易见的。

InfoQ：这是一个相当快的修复了。

Sam : 是的，这是最糟糕的故障之一。我想，在过去的几年里我们并没有出现太多的问题。在过去的几年只，可能停机的时间不超过三小时，停机事件也很少发生。如果真发生了，那绝对是很糟糕的事情。

监控

InfoQ：你们如何做服务器监控？我是指网络性能监控和应用程序的监控。

Sam : 对于网络性能，我们用 Cacti，以及我们自己写的一个工具，叫做 FITB，在 Github 上开源了。他会监控所有的 switch，并保持那些度量项数据比较少。它有点儿像 Cacti。但我不记得到底区别在哪里了，因为我很少处理网络度量数据。我们用这两个工具监控不同端口上 switch 的性能。我们在这方面很少遇到问题。网络几乎没有给我们带来任何麻烦。我们的容量目前是足够的。

InfoQ：那么，系统常见的瓶颈在哪里呢？

Sam : 更多的是数据库和 memcache。最近，我们在 memcache 上遇到了问题，而且与网络相关。有很多特别关键的数据保存在 memcache 中，很多人都需要用。最近，在美国，我们的流量陡增。好象是在 Cyber Monday (译者注：感恩节假期之后的第一个上班日的网购促销活动)，流量非常高，有一个关键数据被访问了很多次，其中的一个 memcache 服务器占满了网络连接。而对这个问题的修复方法只是不再把这个数据放在 memcache 中了。因为根本不需要放在那里，所以我们就把它从那里删除了。但这个修复并不合理，因为直到网卡饱和了，我们才发现。

我们用 Cacti，但我并不知道它的监控频度是多少，也不知道其他人隔多长时间来看一下数据。当然，我们用 Nagios 来监控这个数据，判断事情是否出了问题。

对于客户端数据，我们有自己的一套信号器，记录用户在网站上的操作，这些信息会保存到我们的大数据栈上。对于用户行为的监控，我们使用 Hadoop 集群，来处理分析这些信号量，比如我加载了主页，点击了一个物品，并买下了它。我们会观察分析这类的操作模式。我们也会将前端 JavaScript 的错误记录到后台。我们既有服务器端的性能测试，也有 web 页面的测试，并经常运行它们。我们有一个 web 页面测试集群，用它来判断性能是否达到许可水平。

我们也会将应用程序的度量数据与用户的行为做关联。所以我们会度量有多少人在网站上查询买品列表，以便来衡量用户是如何使用我们的网站的。如果这样的

操作不再发生了，那么也许我们应该在用户体验方面要改点儿什么东西了。我们也会保存我们帮助论坛的那些图片。这样的话，如果越来越多的人在帮助论坛中请求帮助的话，那么很可能我们破坏了什么功能。当然，还有来自客服的信息，我们常常和他们沟通来尝试判断用户遇到的问题。

原文链接：<http://www.infoq.com/cn/articles/interview-sam-haskins>

相关内容：

- [在生产中使用金丝雀部署来进行测试](#)
- [2013 纽约 DevOpsDays](#)
- [最新的技术雷达趋势](#)
- [测试自动化和持续交付](#)
- [使用云计算和虚拟化技术实现持续交付](#)

热点新闻 | News

从小型网站到超大规模网站的 MySQL 参考架构

作者 [Abel Avram](#) 译者 [臧秀涛](#)

Oracle 发布《[面向大规模可伸缩网站基础设施的 MySQL 参考架构](#)》白皮书，针对将 MySQL 用作数据存储的不同类型和不同规模的网站给出了推荐的拓扑结构。

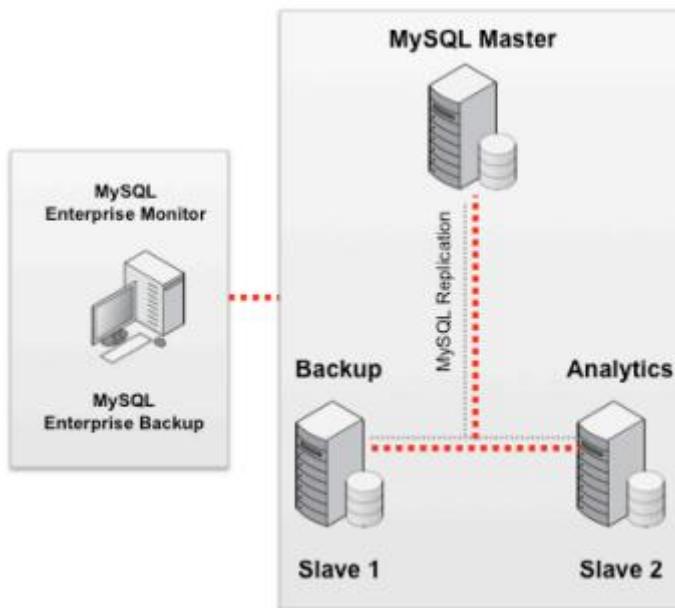
根据分别提供 4 类服务——用户和会话管理、电子商务、分析类应用（多结构数据）和 CMS（元数据）——的网站的规模和可用性要求（如下表所示），这份白皮书给出了 4 个参考架构。

| | Small | Medium | Large | Social Network |
|------------------------|--------------|---------------|--------------|-----------------------|
| Queries/sec | <500 | <5,000 | 10K+ | 25K+ |
| Transactions/sec | <100 | <1,000 | 10K+ | 25K+ |
| Concurrent Read Users | <100 | <5,000 | 10K+ | 25K+ |
| Concurrent Write Users | <10 | <100 | 1K+ | 2.5K+ |
| Database | | | | |
| Sessions | <2GB | <10GB | 20+GB | 40+GB |
| eCommerce | <2GB | <50GB | 50+GB | 200+GB |
| Analytics | <10GB | <1TB | 10+TB | 100+TB |
| CMS | <10GB | <500GB | 1+TB | 2+TB |

请注意，这里给出的指导方针只是基本建议，实际应用中需要根据读写模式、负载平衡和所用的缓存机制等因素进行调整。

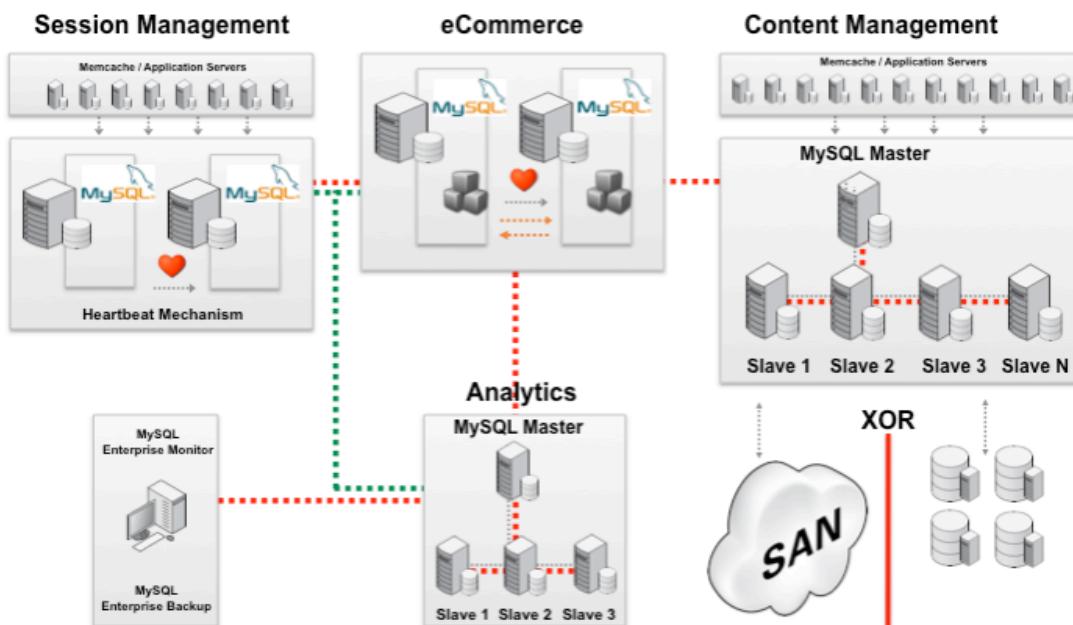
小型 (Small) 网站参考架构

这一参考架构可用于上述 4 类网站的所有小型实现。可以使用 MySQL Replication 来制作数据的副本以支持备份和分析。



中型 (Medium) 网站参考架构

在这种情况下，推荐针对不同类型的活动选择独立的基础设施，考虑每个 MySQL 服务器最多支持 8 个应用服务器，如果因伸缩性需求应用服务器数量增加，则添加更多的 MySQL 从服务器。



为满足会话管理网站和电子商务网站的高可用性要求，可以使用 [Linux 心跳 \(Heartbeat\)](#) 和半同步复制。CMS 网站通常对读操作的向外扩展有更高要求，假定每个 MySQL 从服务器最多可以处理 3000 个并发用户，白皮书建议为每个

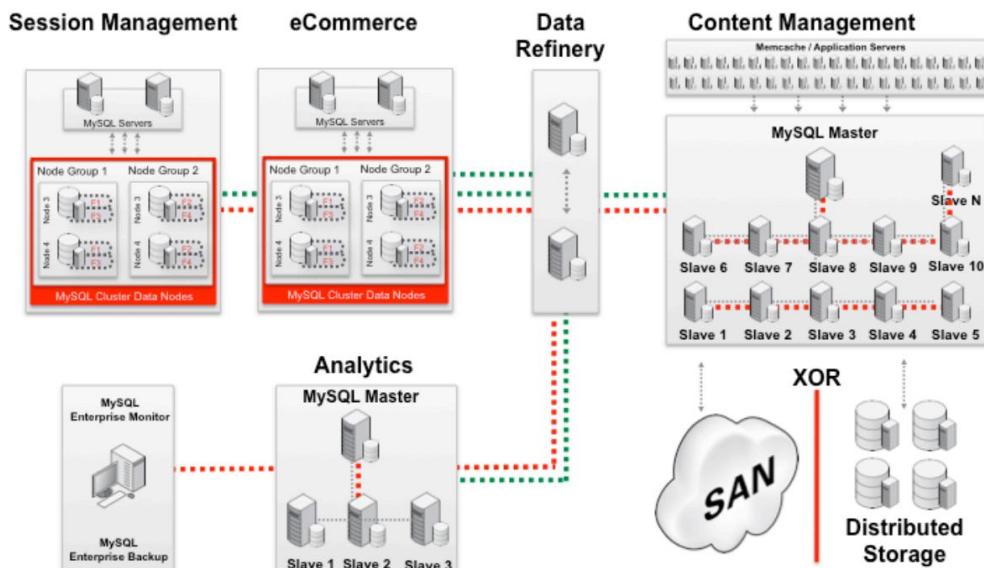
MySQL 主服务器添加 20-30 个从服务器。CMS 系统可将数据保存在一个 SAN 中，或者保存在连接到该服务器的分布式设备中。

会话管理网站和 CMS 网站推荐使用 Memcached，这有助于减轻应用服务器和 MySQL 服务器的负担。

分析类网站的拓扑结构简单一些，1 个主服务器加 3 个从服务器就能解决问题。

大规模 (Large) 网站参考架构

针对大规模网站，白皮书推荐使用 MySQL Geographic Replication 来进行跨数据中心的数据库复制，这种方式支持跨越地理上分离的集群进行异步复制。

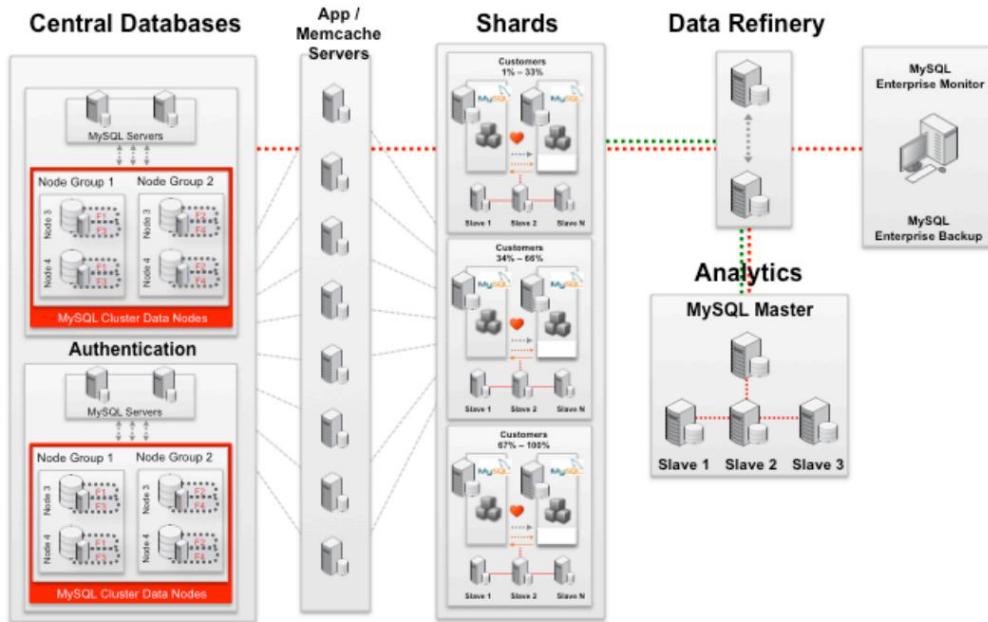


会话管理网站和电子商务网站应该使用集群，白皮书声称“4 个数据节点，1 秒可以支持 6000 个会话(页面点击)，其中每次页面点击生成 8–12 个数据库操作”。
 大规模 CMS 网站使用的配置与中型网站类似，只是必要时需要多添加一些从服务器。针对分析类应用，这里引入了一个数据提炼 (Data Refinery) 单元，用于数据的清理和组织。

超大规模 (Extra Large) 网站参考架构

针对社交网站，白皮书也给出了相应建议。它声称“网络上流量最大的 10 个网站有 9 个部署了 MySQL，其中包括 Google、Facebook 和 YouTube”，但是没有说明这些网站用 MySQL 干什么，不过众所周知的是，LinkedIn 成功应用了 MySQL。

社交网站的拓扑结构利用了中型和大规模网站中实现的概念，包括专用应用服务器、Memcached 和数据提炼单元，但为支持写操作的向外扩展引入了分片（Shard）。MySQL 集群被用于用户的认证和查找，当“用于查找的键（key）不止 1 个”时，直接读写相应的分片。



MySQL 主服务器和从服务器的推荐规格如下：

- 8–16 个 x86-64 位 CPU 核心 (MySQL 5.5 及以上)。
- 4–8 个 x86 -64 位 CPU 核心 (MySQL 5.1 及更早版本)。
- 比活动数据多 3–10 倍的内存。
- Linux、Solaris 或 Windows 操作系统。
- 最少 4 块磁盘，8–16 块磁盘能增加 I/O 密集型应用的性能。
- 支持电池供电高速缓存的硬件 RAID。
- 推荐使用 RAID 10。如果负载为读密集型，RAID 5 也是合适的。
- 2 个网卡和 2 个供电单元用作冗余。

另外，白皮书还有一些针对 MySQL 集群和数据存储设备的建议，再就是用于监控、备份和集群管理的解决方案。

原文链接：

<http://www.infoq.com/cn/news/2013/03/MySQL-Reference-Architectures>

热点新闻

58 同城开源其轻量级 Web 框架 Argo

作者 贾国清

近日，[@58 同城开源](#)微博称，58 同城轻量级 Web 框架 Argo 正式开源。目前 Argo 支撑着 58 同城几乎所有的 Web 站点，包括 Wap 和手机端的访问等，现在 Argo 每天处理 10 亿级的请求。经过长时间的运作与运行，证明 Argo 是一个可靠、高效的 Web 框架。

Argo 在 GitHub 上的地址：<https://github.com/58code/Argo>，Argo 是希腊神话中的一艘船，58 所有开源项目都将采用希腊神话系列命名，这也包括早先开源的服务通信框架 [Gaea](#)。

Argo 起源于 58 同城的内容 Web 框架 WF(Web Framework)。目前 WF 支撑着 58 同城几乎所有的 Web 站点，包括 Wap 和手机端的访问等，现在 WF 每天处理 10 亿级的请求。经过长时间的运作与运行，证明 WF 是一个可靠的、高效的 Web 框架。Argo 在 WF 做了大量优化和重构，以适应各组织软件开发的个性化需求，提升了系统性能，具有更好的可扩展性。Argo 的开源反过来也促进 WF2.0 的开发。

Argo 不是一个通用的 Web 框架，主要工作环境是：

- Servlet 3.0 环境，主要针对 Tomcat 7.X；
- 基于 [Guice](#) 的 IoC，组织和项目可以各提供一个 module 注入模块，而且 module 的命名必须符合约定；
- Maven 依赖，项目的代码体系和 Maven 默认代码体系一致，Maven 以插件提供开发过程中所需要的开发运行环境

Argo 的设计遵循“约定优于配置”、简单和纪律严明的哲学观，既可以减少软件开发人员做决定的数量，又不失灵活性。同时，Argo 项目代码结构简单，可以不需要任何配置文件。

关于 The Gaea Project

[Gaea](#)是服务通讯框架(Service Communication Framework)支持跨平台具有高并发、高性能、高可靠性，并提供异步、多协议、事件驱动的中间层服务框架。

关于 58 同城开源

微博账号：[@58 同城开源](#)

联系邮箱：code@58.com

原文链接：<http://www.infoq.com/cn/news/2013/03/58com-opensourced-argo>

相关内容

- [58 同城开源其轻量级 Web 框架 Argo](#)
- [腾讯前端 Alloy 团队访谈：HTML5 开源图像处理框架 AlloyImage](#)
- [GitHub 引起开源“平民化”变革](#)
- [腾讯 AlloyTeam 再次发力：开源 HTML5 图像处理引擎 AlloyImage](#)
- [Redis 开源文档《Redis 设计与实现》](#)
- [Netflix 继续开源，更多猴子进入视野](#)

热点新闻

QQ 安全中心的两小时 Bug：快速响应+快速部署 的经典案例

作者 [杨赛](#)

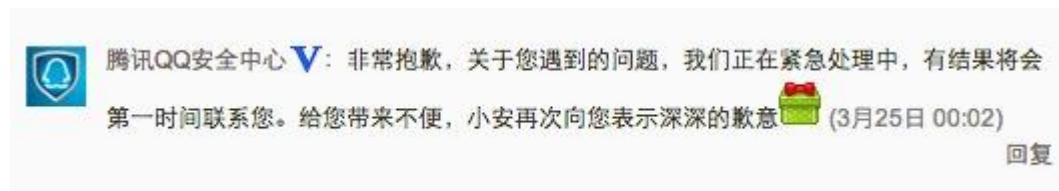
3月24日晚11点22分，[@左耳朵耗子](#)想改QQ密码，触发了QQ安全中心的一个bug：



于是[发微博](#)吐槽之。大家初步分析这是一个比较低级的错误，[@翁雨键](#)在评论中认为：

目测是因为#是非法字符。程序员偷懒跟密码长度不对用了同一个错误提示，QA不知道有这样的规定设计测试用例没覆盖到，PM/UE太junior只关心正常的path没给错误用例设计提示。

然后过了半个多小时，[@腾讯 QQ 安全中心](#)的运营人员发现了这条吐槽，并进行了反馈：



又过了一个多小时，[@腾讯 QQ 安全中心](#)又回来这条微博下评论，说 bug 已经改好了：



腾讯QQ安全中心 **V** 您好，您反馈的问题我们已经紧急修复上线，给您带来不便小安非常非常抱歉。感谢您对QQ安全的关注和支持，如有其他问题请随时私信小安批评指正，我们会全力改进。 (3月25日 01:18)

回复

此消息一出，腾讯 QQ 安全中心背后的运维、响应、部署流程立刻引起了大家的关注。当事人陈皓对腾讯这次的响应速度[进行了高度评价](#)：

只有高度自动化测试+程序员完全自主才会有这样高效的流程。

在看到这条消息的时候，InfoQ 编辑正好在跟[腾讯互联网产品运维副总监 Coati](#)沟通 [QCon 北京](#)分享的事情，便问他腾讯在快速响应、部署这方面是怎么做的。Coati 表示，虽然 QQ 安全中心属于基础运维部，而自己负责的范围属于应用运维，但根据他跟同事的沟通，他们在运维响应方面的很多处理流程是差不多的，比如：



一般我们推送新版本，大体分为三级：一种是大版本的更新，涉及到特性增减改动等方面的，一般每周会做 2-3 次，而且会选择周一到周四之间做推送，一般不在周五做推送；另外一种是运营性的版本更新，这个更新频率不一定。这两种都是需要 QA 参与的。第三种是 bug 修复，这里也分级别，比如外网对 bug 进行快速修复，大的 bug 是需要走灰度的，而小的 bug，比如只是涉及到界面、体验相关的，是可以免测试的。

这次密码功能 bug 的修复，因为涉及到程序逻辑，所以上线之前还是要经过测试的。不过这并不是说测试人员就一定要在公司，我们可以 VPN 登陆，从家里也可以随时响应的。

Coati 表示会在 [QCon 北京](#)上的分享介绍更多的细节，包括：

- 腾讯的互联网产品运维这块是如何划分领域的？
- 运维和开发有明确的界限么？平时如何协作？
- 代码部署和代码审查都使用什么工具？发布流程如何？发布的频率如何？
- 应用运维的监控分为几个层面？信息源来自哪里，监控的力度、频率如何？

想了解腾讯运维体系的同学们，可以重点关注 Coati 的分享。

InfoQ 在前不久刚刚介绍过 [Etsy 的运维体系](#)。根据 Sam Haskins 的介绍，Etsy 的核心平台团队平均每天的部署次数在 20-30 次，最高时甚至能达到 70 次。他们的监控手段也比较立体，在自动化监控工具之外，有大量的工作是靠人肉看图表来完成的，另外帮助论坛和客服那里也是重要的反馈来源（他倒是没提到有没有在用 Twitter 做这方面的监控）。

阿里系方面，应用运维的刘勇（[@淘宝仲明](#)）也[公开介绍](#)过他们的运维流程。阿里应用运维的发布观察期一般在一个小时左右，涉及到交易等重要的变更则走灰度，需要 24 小时的观察期；监控也分为很多维度，针对底层有常规指标的监控，针对业务层面有交易量等业务指标的监控，也有对用户投诉的监控，根据投诉出现的频率和修复 bug 需要的时间来确定 bug 修复的优先级。仲明也是今年 QCon 北京自动化运维专场的讲师。

乔梁也对[百度的 DevOps 运动进行过介绍](#)，提到了从 2010 年到 2011 年之间，百度内部的提测耗时和上线部署耗时大大减少的情况。对于不了解 DevOps 文化的同学，这篇文章也是一个很好的入门资料。

最近几年在运维界流行的 [DevOps 文化](#)，可以说是开发领域的敏捷运动的一个延伸，提倡开发和运维协同工作，以实现快速部署、快速响应。这个理念尤其适用于互联网产品和移动互联网产品，因为此类产品的迭代速度是核心竞争力，不仅影响产品本身的质量、功能发展，还很大的影响到产品的市场口碑和传播效应。像本次 QQ 安全中心的例子，用户反馈的信息源出现在新浪微博这个第三方平台上，而众所周知，官方微博的运营人员往往是不懂技术的。

微博运营人员如何能够判定这个 bug 反馈的重要程度（无论是在技术层面还是营销层面）？运营人员把此类 bug 反馈到哪个渠道（一般的情况下，腾讯这样大的公司，你不可能很快知道负责 QQ 安全中心这个产品的技术经理是哪位的）？谁来决定这个 bug 值不值得在凌晨 12 点的时候电话把同事叫起来进行修复（当然，也有可能是这一块的负责人那个时候还没下班）？

这些都是很有意思的话题，而且已经不完全是技术的范畴。腾讯这样规模的公司能做到这样快速的反馈，的确很厉害；同时这也意味着，即使你是一个小公司，在产品迭代、响应速度上要赶超腾讯，是难度很大的。

现在就开始建立这样一个快速响应的开发+运维+测试的体系吧！

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

Iron.io 从 Ruby 迁移到 Go : 减少了 28 台服务器并避免了连锁故障

作者 [Todd Hoff](#) 译者 [臧秀涛](#)

过去几个月，我在用 Go 语言编写系统，所以一直密切关注能够证实我的选择是正确的”的那些消息。当 Iron.io 记录下[使用 Go 重写 IronWorker 的经验](#)时，机会出现了。Iron.io 中非常忙碌的作业执行系统最初是用 Ruby 编写的。

重写的结果是：

- 服务器数量从 30 台减少到 2 台，而且第 2 台仅用于实现冗余。
- CPU 利用率下降至 5% 以下。
- 所用内存也下降了很多。Rails 应用在启动时需要接近 50MB 内存，而 Go 版本在启动时只需要几百 KB 内存。
- 连锁故障成为历史。
- 运行于成百上千台服务器上的新服务完全用 Go 编写。
- 他们认为，Go 的使用使他们得以“构建伟大的产品，得以成长和扩展，同时还能吸引一流人才”。他们的博客中写道：“我们认为，在可预见的未来，它将继续帮助我们成长。”一般建议根据人才库的规模来选择编程语言，他们发现 Go 语言的选择帮助他们吸引了顶级人才。
- 容易部署，因为 Go 程序会编译为一个单一静态映像。
- Go 存在的小问题：需要学习一种新语言，库还有限。
- 如果服务器流量很高，或者你想应对突发的增长，Go 是很好的选择。

当然，如果没有第二系统效应(Second System Effect)的影响，重写会快得多，不过你可能会回想起 LinkedIn 的类似经验：[LinkedIn 从 Rails 迁移到 Node : 服务器减少 27 台，速度提升多达 20 倍。](#)

下面说明一下 Go 解决的问题：

- 在使用 Ruby 时，服务器的 CPU 利用率维持在 50% 到 60% 之间。为将 CPU 利用率保持在 50% 左右，可以增加服务器，这样就可以优雅

地处理流量峰值。但这种方式有个缺点：需要昂贵的服务器来进行水平扩展。

- 他们有一个非常有趣的故障模式。当流量出现峰值时，Rails 服务器的 CPU 利用率将达到 100%。这就致使该服务器看上去是失效了，进而引发负载均衡器把流量路由到其余服务器上，这样更多服务器的 CPU 利用率会飙升到 100%。最终导致连锁故障。

使用 Ruby 有助于产品快速上市，这个理由十分在理。虽说性能并非一切，但这里我们看到了性能的价值，尤其是在 Web 层之外，性能更为重要。表现良好的服务在健壮性和成本方面都有巨大优势。

以往，系统的弱点被服务器的量所掩盖，但大量服务器要花很多钱，而且未必总能解决问题。

性能起到了缓冲作用，使系统既能承载流量而不至崩溃，又能容忍一次次的突发冲击。即时（Just-in-time）分配和启动新实例都需要时间，而长时间的流量峰值可能引发连锁故障。针对性能编码，而不是针对产品上市时间编码，即可防患于未然。

Go 并不完美

如果看一下 Go 的 Google 网上论坛，你会发现 Go 并不是没有性能问题。不过这些问题往往可以编码绕过。比如用 `bufio.ReadSlice` 代替 `bufio.ReadString`，能够去掉一次数据复制，代码魔法般地快了若干倍。

学习这些技巧是需要时间的，尤其对这样一种新语言而言。

让 Go 真正处于不利地位的是，JVM 和 V8 JavaScript 引擎在垃圾回收优化和代码生成方面已经有了许多年的经验积累。Go 要赶上尚需时日。

性能永远不会是免费的。你必须巧妙地编码，将状态共享最小化，不要老翻腾内存，要尽力剖析，理解自己的语言并且能够用这种语言来做正确的事。

原文链接：<http://www.infoq.com/cn/news/2013/03/ruby-to-go>

热点新闻

LinkedIn 实时低延迟数据抓取系统 Databus 开源

作者 [郑柯](#)

去年的架构师峰会上，来自 LinkedIn 的高级软件工程师 [Lei Gao](#) 做了一场名为 [《LinkedIn 的数据处理架构》](#) 的演讲，着重介绍 LinkedIn 内部的数据基础设施的演变，其中提到 Databus 数据总线项目，当时就引起大家诸多好奇。前不久，LinkedIn 工程团队官方博客[发布消息](#)：Databus 项目开源。

在互联网架构中，数据系统通常分为真实数据（source-of-truth）系统，作为基础数据库，存储用户产生的写操作；以及衍生数据库或索引，提供读取和其他复杂查询操作。后者常常衍生自主数据存储，会对其中的数据做转换，有时还要包括复杂的业务逻辑处理。缓存中的数据也来自自主数据存储，当主数据存储发生变化，缓存中的数据就需要刷新，或是转为无效。

LinkedIn 内部有很多专用的数据存储和服务系统，构成了一个多种多样的生态系统。基础的 OLTP 数据存储用来处理面向用户的写操作和部分读操作。其他专用系统提供负责查询，或者通过缓存用来加速查询结果。因此，整个生态系统中就需要一个可靠的、支持事务的、保持一致性的数据变更抓取系统。

Databus 就是一个实时的低延迟数据抓取系统。从 2005 年就已经开始开发，2011 年在 LinkedIn 正式进入生产系统。

在 Databus 的 [Github 页面](#) 上，介绍了他们选择目前解决方案的决策过程。

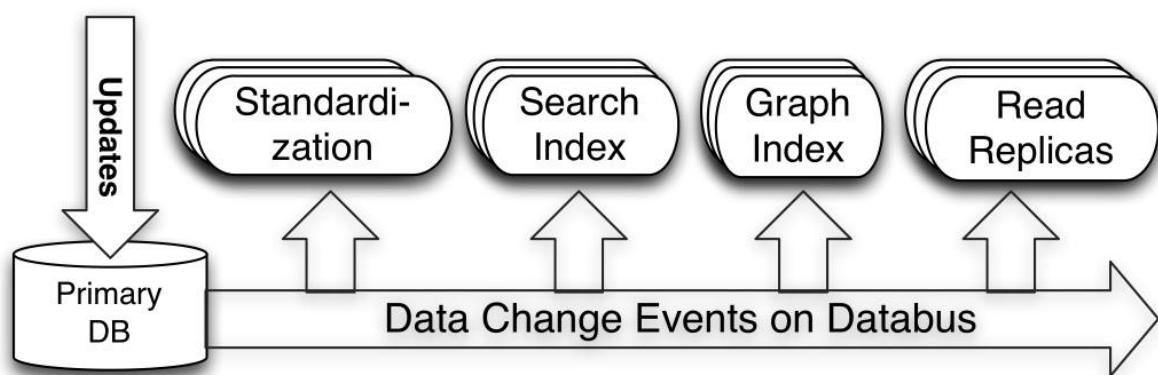
“ 处理这种需求有两种常用方式：

应用驱动双向写：这种模式下，应用层同时向数据库和另一个消息系统发起写操作。这种实现看起来简单，因为可以控制向数据库写的应用代码。但是，它会引入一致性问题，因为没有复杂的协调协议（比如两阶段提交协议或者 paxos 算法），所以当出现问题时，很难保证数据库和消息系统完全处于相同的锁定状态。两个系统需要精确完成同样的写操作，并以同样的顺序完成序列化。如果写操作是有条件的或是有部分更新的语义，那么事情就会变得更麻烦。

数据库日志挖掘：将数据库作为唯一真实数据来源，并将变更从事务或提交日志中提取出来。这可以解决一致性问题，但是很难实现，因为 Oracle 和 MySQL 这样的数据库有私有的交易日志格式和复制冗余解决方案，难以保证版本升级之后的可用性。由于要解决的是处理应用代码发起的数据变更，然后写入到另一个数据库中，冗余系统就得是用户层面的，而且要与来源无关。对于快速变化的技术公司，这种与数据来源的独立性非常重要，可以避免应用栈的技术锁定，或是绑死在二进制格式上。

在评估了两种方式的优劣之后，我们决定选择日志挖掘，将一致性和单一真实数据来源作为最高优先级，而不是易于实现。

Databus 的传输层端到端延迟是微秒级的，每台服务器每秒可以处理数千次数据吞吐变更事件，同时还支持无限回溯能力和丰富的变更订阅功能。概要结构如下图。

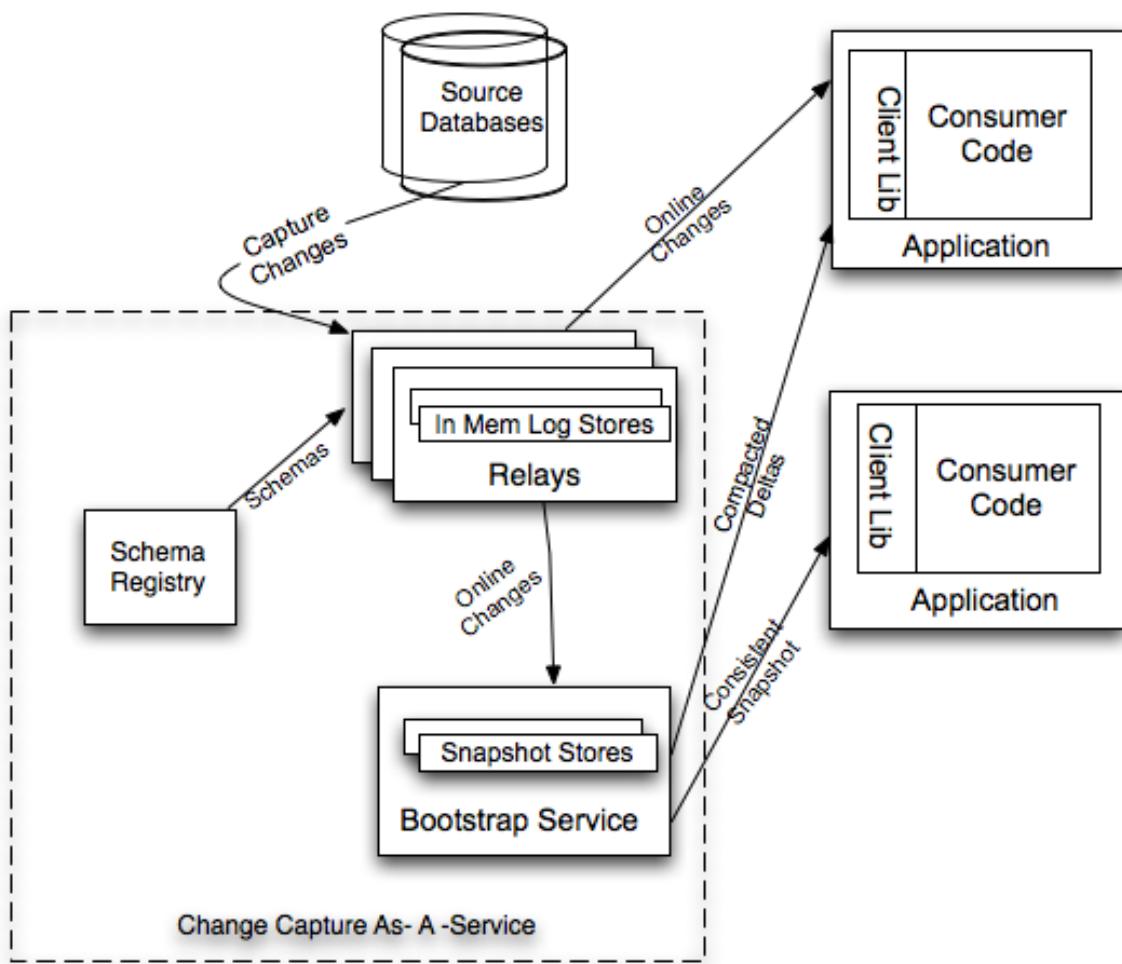


图中显示：Search Index 和 Read Replicas 等系统是 Databus 的消费者。当主 OLTP 数据库发生写操作时，连接其上的中继系统会将数据拉到中继中。签入在 Search Index 或是缓存中的 Databus 消费者客户端，就会从中继中拉出数据，并更新索引或缓存。

Databus 提供如下功能：

- 来源独立：Databus 支持多种数据来源的变更抓取，包括 Oracle 和 MySQL。Oracle 适配器在开源版本中有提供，MySQL 适配器将在以后提供。
- 可扩展、高度可用：Databus 能扩展到支持数千消费者和事务数据来源，同时保持高度可用性。

- 事务按序提交：Databus 能保持来源数据库中的事务完整性，并按照事务分组和来源的提交顺序交付变更事件。
- 低延迟、支持多种订阅机制：数据源变更完成后，Databus 能在微秒级内将事务提交给消费者。同时，消费者使用 Databus 中的服务器端过滤功能，可以只获取自己需要的特定数据。
- 无限回溯：这是 Databus 最具创新性的组件之一，对消费者支持无限回溯能力。当消费者需要产生数据的完整拷贝时(比如新的搜索索引)，它不会对主 OLTP 数据库产生任何额外负担，就可以达成目的。当消费者的 data 大大落后于来源数据库时，也可以使用该功能。



上图中介绍了 Databus 系统的构成，包括中继 Relay、bootstrap 服务和客户端库。Bootstrap 服务中包括 Bootstrap Producer 和 Bootstrap Server。快速变化的消费者直接从 Relay 中取事件。如果一个消费者的 data 更新大幅落后，它要的数据就不在 Relay 的日志中，而是在 Bootstrap Producer 里面，提交给它的，将会是自消费者上次处理变更之后的所有 data 变更快照。

Databus Relay 中继的功能主要包括：

1. 从 Databus 来源读取变更行，并在内存缓存内将其序列化为 Databus 变更事件
2. 监听来自 Databus 客户端（包括 Bootstrap Producer）的请求，并传输新的 Databus 数据变更事件

Databus 客户端的功能主要包括：

1. 检查 Relay 上新的数据变更事件，并执行特定业务逻辑的回调
2. 如果落后 Relay 太多，向 Bootstrap Server 发起查询
3. 新 Databus 客户端会向 Bootstrap Server 发起 bootstrap 启动查询，然后切换到向中继发起查询，以完成最新的数据变更事件
4. 单一客户端可以处理整个 Databus 数据流，或者可以成为消费者集群的一部分，其中每个消费者只处理一部分流数据

Databus Bootstrap Producer 只是一种特定的 Databus 客户端，它的功能有：

1. 检查中继上的新数据变更事件
2. 将变更存储在 MySQL 数据库中
3. MySQL 数据库供 Bootstrap 和客户端使用

Databus Bootstrap Server 的主要功能，就是监听来自 Databus 客户端的请求，并返回长期回溯数据变更事件。

在 LinkedIn，Databus 支持的系统有：

1. 社会化图谱索引（Social Graph Index），服务 LinkedIn 所有图谱查询
2. 人员搜索索引（People Search Index），支持搜索所有 LinkedIn 用户
3. 用户档案数据（Member Profile）多个冗余的读取查询

对 Databus 项目感兴趣的同学，可以去 [Github](#) 上查看更多信息和相关源码。

原文链接：<http://www.infoq.com/cn/news/2013/03/linkedin-databus>

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
沐泽宁谈主观决策架构
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转型构
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深度进阶十：Java对集群划分与Hadoop
将数据打散之一：关于横向的数据设计
分布式平台的组件化PaaS
社区驱动的开源计划
来自Padmanab的真言

InfoQ 每月8号出版

特别专题

在高效团队中工作的经历令人难忘 ,这样的团队是如何建立起来的 ?日常工作中又有哪些实践可以维持一个高效团队的存在 ?一项 2007 年科学杂志的研究表明 ,科学研究、工程甚至一些艺术创作越来越多的是由团队而不是个人完成。但是哪怕是把一群高智商的人聚在一起 ,也不一定能够组成高智商的团队 ,而弗吉尼亚理工大学的研究表明 ,团队中很多的聪明人甚至会变得弱智。导致团队作出愚蠢的决定。

高效的团队协作和卓越的团队文化是打造高效能团队核心要素 ,在本专题中我们会从不同背景的组织邀请一些高效团队的领导者 ,分享他们在打造高效团队过程中的挑战、成功以及失败经验。

本期为您精选的内容有 :继 [《建设全功能团队》](#) 和 [《建设全功能团队——实践篇》](#) 之后的新作 [《全功能团队 - 数据篇》](#) ,以帮助读者从数据的维度来解读团队。Facebook 的工程师文化和 Hacker 文化令人印象深刻 ,本期选编了《打造 Facebook》作者王淮的一篇关于 [《以 Facebook 为案例剖析科技公司应有的工具文化》](#) 的报道。第三篇是 [《自动化构建:一致性关键之道》](#) ,您将了解到组建自动化流程的一些动机 ,以及你将需要接触的一些概念。本系列的第二部分将描述针对.NET 解决方案的具体实现 ,但这些技术在任何环境下都适用。

特别专题 | Topic

全功能团队 - 数据篇

作者 吴少博

在《[建设全功能团队](#)》和《[建设全功能团队——实践篇](#)》两篇文章中，我的同事[胡凯](#)曾介绍过建设全功能团队的必要性和良好实践，此后在围绕这一话题的讨论中，很多人都分享了自己的理解，或看好，或看淡。在[ThoughtWorks](#)有许多团队一直在建设全功能团队方面实践着，在这篇文章中我希望与大家分享我从这些团队收集到的过去一年来的数据，以及更切身的理解。

简短回顾

全功能团队

它不仅是由一专多能的多面手成员组成的软件开发团队，而且是所有成员共同分担职责的团队。团队中的各项职责不再与具体的人员耦合，每一个人都有可能做并且有能力做超过一种传统角色所做的事：例如在某个时刻，开发人员在做测试，测试人员在做业务分析，业务分析人员在做部署；前端开发、后端开发、数据库维护被开发人员一视同仁；所有的人都能与客户沟通，也承担着只有传统的项目经理才闹心的项目风险。

来自软件开发业界的质疑

在关于全功能团队的讨论中，最激烈的质疑集中在能力和效率两个方面：

- 从效率上看，除非团队小得可怜，分工是必然的：团队成员频繁地转换工作职能，就意味着要不时地做点不是特别熟练的事，这是否降低了效率呢？比如，重复性高的测试工作虽然入门简单，但一个传统开发人员也需要一些时间的经验积累，才能替代专职 QA，做到快速且高效地测试；又是不是应该找专人来做部署呢？
- 从能力上看，分工似乎是必需的：让业务分析人员明白代码实现是不是有必要，会不会强人所难？开发人员有较强的代码和业务阅读能力，但是否同样具有同样水平的测试用例设计能力？即便是多面手团队具备的技能深度也是有限的。

带着这样的问题，我们在过去一年里用实践里证明了全功能团队的可行性，并在团队成员培养和项目可持续进展上都受益不少。

我们怎么做到的

针对效率问题

对效率的通常考虑方式源于工业化生产，认为分工后的重复性工作能提高劳动熟练度，从而提高生产效率。但是，知识工作不是流水线上拧螺丝，它的核心问题是效果 (Effectiveness) 而不是效率 (Efficient) 1。通俗地说，打字最快的不一定是个程序员，100 行代码也不一定比一行代码更有价值。所以，对有价值的软件开发者而言，真正重要的是知道要做什么、为什么、怎么正确地做到。

全功能团队中，我们让成员做不同角色的职责，就是要打破知识壁垒，让大家都站在不同的角度看我们的软件，传递知识、扩展认知；等再回过头看自己原来做的“本职”工作时，从其他角色的角度得到的知识会帮助我们用更正确方式地做对的事。同时，培养多面手团队成员的益处也在于，可以按需调整做某件事的人数或安排人员的替补，这对团队的人员利用率提升是很有帮助的。

我们这样做：

- 我们不为前端开发、后端开发和运维工作划分岗位，要求所有开发人员接触到所有层次的代码和环境。有了全面的了解之后，再没有人“因为没做过所以不敢碰”，所以接下来就是提升各自能力来把事情做得更好了。
- 我们每两周轮换做测试工作的成员。做测试人员期间，他会测试开发完成的功能以及回归测试各个组件，这有助于他了解系统的整体行为；同时他带去了自己的开发经验，不仅能在外部功能层次测试，还能深入代码挖掘，比专职的测试人员更能找到潜在的缺陷。
- 我们的业务分析人员要计划每次部署和安排部署前的回归测试，对待部署的功能须十分清晰；他们要为每个待开发功能准备全面的验收条件，甚至写出验收测试描述(Given/When/Then)2，这样的用户故事可读性非常好，因为验收测试可直接拿来与客户或开发人员交流。
- 我们没有固定的人与客户做接口沟通，所有的轮值测试人员都需要向客户展示完成的功能以确认验收，所有的人都有义务向客户询问疑难的需求，所有的人轮流做迭代报告或主持回顾会议。

- 所有项目上的信息都在团队内共享，结对编程也 2、3 天一轮换，这减少了团队成员的上下文盲点，便于各人迅速定位并正确处理问题。

针对能力问题

对能力的担心是可以理解的：对不熟练的事甚至头一次做的事，谁都不能很有把握独立做到；每个人的技术能力是有限的，职责的切换意味着挑战来临。然而这是一个团队，没有人孤军奋战，也不会安排成员做登天的事。

全功能团队在能力问题上重视的是团队成员的成长和项目的可持续进展。培养成员逐渐胜任某项新职责是他能力的拉伸，只要方法得当，团队就会平稳地在几个月后收获一批一专多能的多面手成员。而这种成长也不是以暂时牺牲项目进展为代价的，项目和人员成长不站在对立面。

我们这样做：

- 将职责和个人去耦合之后，提炼出若干职责，如业务分析、测试、前端开发、数据库维护、部署等。
- 我们为每项职责找出领导者，称为教练。教练是某一职责上的专家，如测试教练，他是测试工作上能力最强又所知最多的人，由他来辅导测试技能尚需锻炼的人，通过结对的方式授机宜，帮助他适应“新角色”的工作。
- “新人”成长后，教练会跟踪其工作的进展，并只在复杂情况下才伸手帮忙，如某些紧急应对。教练也担负者第一时间响应客户疑问的责任，他们是客户与开发团队关于某方面工作的接口，客户知道想要跟踪某项工作的进展情况，只要联系他即可。
- 某职责上的教练也常是其他职责上的“新人”，他也需要被帮助，需要同样努力胜任新职责。
- 从项目的可持续进展上看，全功能团队能够轻易地克服人员短板，并保持很高的团队凝聚力。因为团队里都是多面手成员，且大家保持了非常频繁交流和信息共享，各个人即便相互替换位置来做事情也很容易。

我所在的团队里，有很多事都是其他非全功能团队无法想象的：

- 没有专职的测试人员和部署人员，所有人都有能力做开发、测试和向产品环境部署，即便是才加入团队半年的大学毕业生，也能独挑这副担子了。项目的缺陷和交付进度依旧保持平均数目，并未出现由于没有“专职”人员而导致的不“专业”。
- 从不因为某人的突然请假而阻碍某件工作。这得益于多面手成员们每天充分的信息共享和结对工作，任何一个人请假了立即有人能顶替他做好职责。
- 从项目中移出成员比较容易，不会因关键人物的离开导致项目遇险。四个月前，当时做业务分析兼项目经理的女同事突然得知怀孕需要休假，我们也只做了简单的交接，就完成了这次人员变更，并保持了平稳的项目交付能力；如今这项职责早已向另一成员成功交接。

用数据说话

问题 1：没有了专职测试人员之后，系统新增功能的缺陷数目是否显著增加呢？

答案：没有。图 1 显示的是我所在的项目过去一年半内的缺陷数曲线。竖线标示的是自 2012 年 7 月 1 日起，团队取消了专职测试人员，以两周一轮换的频率让所有开发人员轮流做测试。由图可见，这个实践开始之后的缺陷并未较之前显著增加，7 月初和 10 月初两次重大发布仍然是影响缺陷曲线的最重要因素。

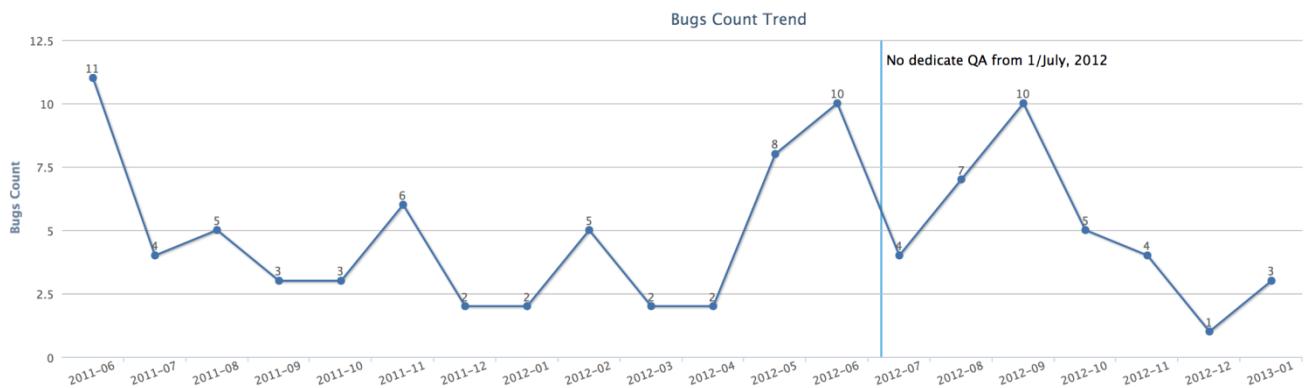


图 1 取消专职测试人员前后的缺陷数曲线

问题 2：全功能团队中的成员角色切换甚至人员变更较频繁，有没有对项目的交付进度产生严重影响？

答案：没有。图 2 是我所在的项目近一年来的团队人员变更情况与交付速率曲线的对比，这里的交付速率数值仅包含具有业务价值的用户故事卡的点数，而其他

的如技术卡和缺陷卡的工作量是单另进行统计的。在 2012 年 7 月到 8 月这段时间内，团队的成员调整的较频繁，同时伴随成员调整，各人的角色也在调整，参照交付速率曲线来看，在这段变更时期以及之后的适应期里，团队仍然保持了如以往的交付节奏和速度；交付速率曲线上在 1-4 月、4-7 月、7-10 月体现出的冲高而后渐落的变化模式，也与项目在 4 月、7 月、10 月的三次重大新功能发布相契合。前文图 1 也参证了在 7 月到 8 月这段时间里，项目的缺陷数目并未显著爆发。所以我们观察到的是项目交付进度未受影响。

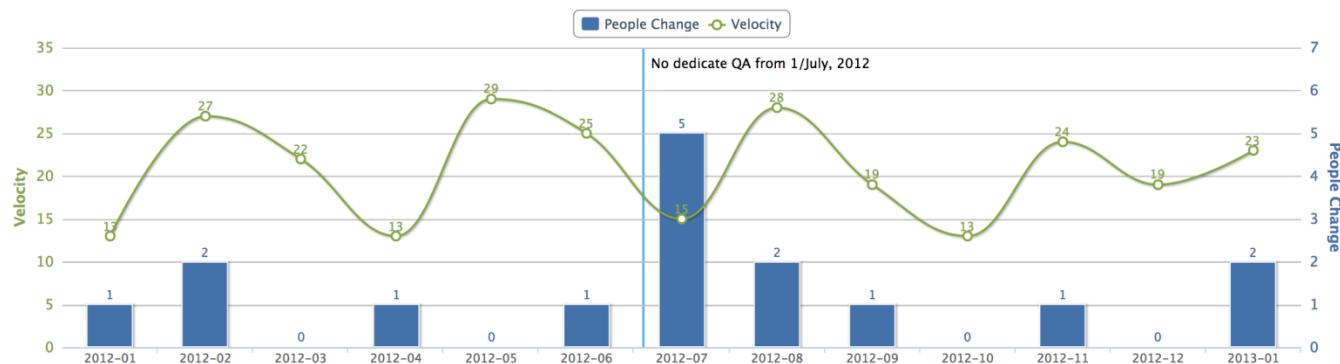


图 2 团队人员变更与交付速率按月对比

问题 3：全功能团队强调一人多用，那它真的比普通团队的人才利用率高吗？

答案：是的。图 3-1, 3-2 是在 [ThoughtWorks](#) 西安办公室做的一次关于各人所担任过的团队职责的调查结果。来自包括我所在的团队在内的几个团队共 38 人参与调查，他们自认为的本职角色大多为开发人员（78.9%），具体的本职角色统计如图 3-1 所示，角色比例与业界数据 3 相去甚远。但他们在团队中做过的职责都不止一个，如图 3-2 所示，以人数所占比例最高的开发人员为例，他们中 77% 的人做过测试工作，23% 的人做过项目管理，30% 的人做过业务分析，40% 的人做过运维。从数据可见，全功能团队中不易出现因为某角色人员的缺席导致的交付阻塞，因为其他角色的人可以转换职责来代替缺席者。建设这样的团队对多个团队之间共享人才、提高公司整体人员利用率会有帮助。

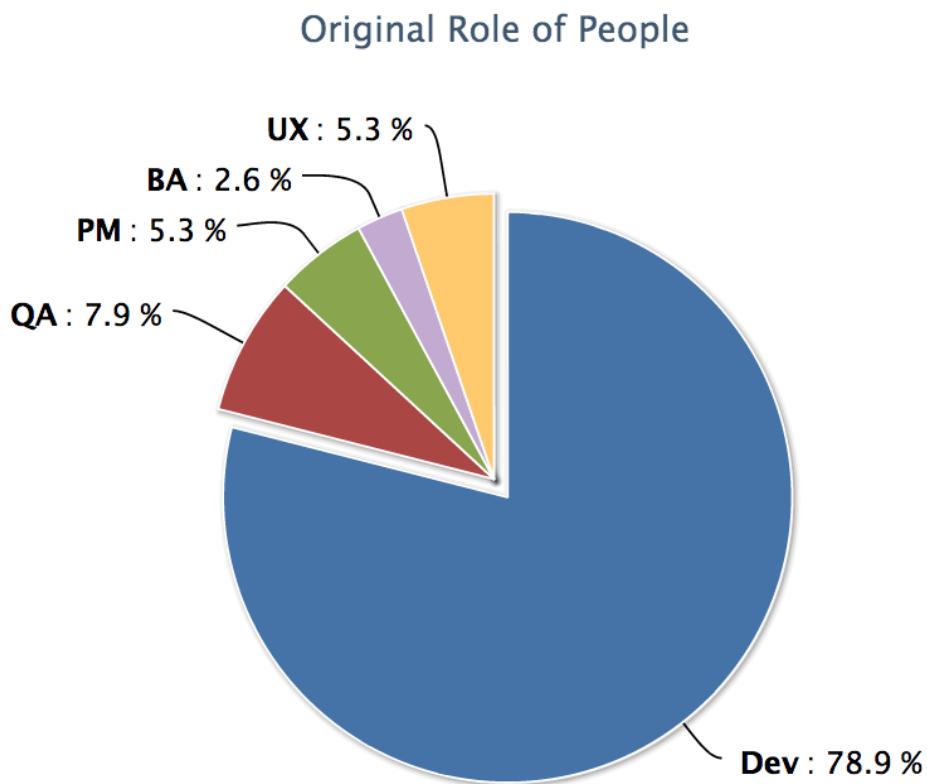


图 3-1 所有参与调查者职位的分布比例

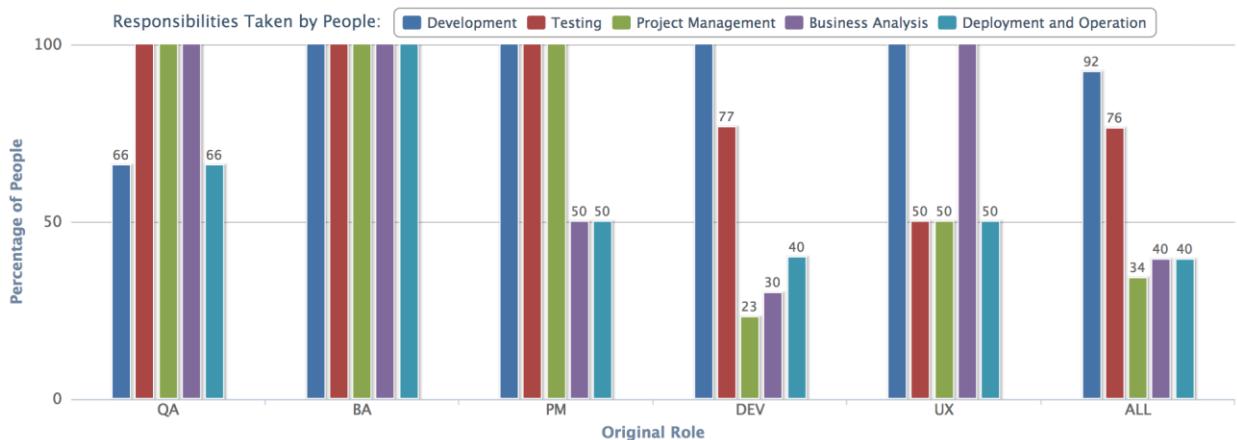


图 3-2 参与调查者所担任职责的调查结果

结论

从我观察到的 ThoughtWorks 全功能团队的实践以及收集到的数据来看，建设全功能团队在中小型项目里能顺利进行。我们按照良好实践所做的尝试和努力，让项目、个人以及公司都受益了。那些来自软件开发业界的忧虑，从本文谈及的实践以及数据来看也应该释然了。

注解与参考

1. Peter Drucker,
2. [Uncle Bob's post](#), Nov 2008
3. 通过招聘网站估算出的数字，如果需求是基本平衡的，市场所提供的职位数量比例与相关从业者的比例应当基本一致。对某主流招聘网站 IT 板块进行搜索得到：35000 开发职位，14000 测试职位，12000 分析职位。

原文链接：<http://www.infoq.com/cn/articles/full-function-team-about-data>

相关内容

- [全功能团队 - 数据篇](#)
- [物理墙和虚拟墙之争](#)
- [实战：持续交付中的业务分析](#)
- [持续集成理论和实践的新进展](#)
- [建设 DevOps 能力，实现业务敏捷](#)
- [运用系统思考，走上改善之路](#)

特别专题

Facebook 元老王淮谈科技公司应有的工具文化

作者 郑柯

王淮是 Facebook 的早期员工，也是 Facebook 内部第二位中国籍工程师和第一位研发经理，曾经负责支付后台和安全系统，担任反欺诈部门的技术经理。现在，他是一位以兴趣为导向的顾问型天使投资人。

前不久，他发表了一篇文章——《[以 Facebook 为案例剖析科技公司应有的工具文化](#)》。其中，他开宗明义提出：

不断发展、改进公司的内部工具，可以极大提高每个员工的工作效率，可以减少运营人员的数目；这样既改善了整体协调，又减少了整体开支。

王淮提到：当时招聘他进 Facebook 的总监黄易山，是对内部工具的最有力倡导者：

他极度建议，公司要把最好的人才放到工具开发那一块，因为工具做好了，可以达到事半功倍的效果，所有人的效率都可以得到提高，而不仅仅是工程师。

王淮接下来介绍道：Facebook 有两个工具组——研发工具组、网站支持和工具组。

研发工具组的工具包括：

- 管理分配所有的开发专用服务器

在一个页面上你可以很清晰的看到所有的开发服务器，包括哪些人是现在的使用者，什么时候申请分配的，服务器的操作系统版本，配置信息等等；对于空余的服务器，你可以一键申请，并自动初始化该服务器。这让刚加入的菜鸟们可以迅速的获得自己的研发活动空间。

- Git 集成工具

在提交代码之前自动的检测所修改的代码是否符合公司代码规范，如果不符合，该工具会自动警告，但把决定权交给工程师。

- 现已开源的代码审查工具 Phabricator

工程师可以在页面上非常方便的针对每一段(单行或者多行)代码进行交互讨论；负责审查的工程师可以接受代码改变，可以提出疑问要求原作者继续修改，可以提出自己不适合以推出该代码审查，等等。只有代码被明确接受之后才能被工程师提交到服务器端的代码库，这一点集成到提交工具中强制执行。

- 灰度发布工具

在这个工具中，工程师和产品经理（也可以授权给其他非研发人员）可以设计新产品发布的目人群特点（比如年龄，性别，地域，教育，等方面做出限制）及发布的人群比例（在0-100%之间自由调整），所有的改变不需要代码的改变，只需要在工具页面上点击鼠标即可，让灰度发布变得很轻松。

- 发布过程监控工具

发布的过程由一个利用点对点（BitTorrent）算法实现的工具进行多线程同时发布，对于更新几十万台机器只需要几十分钟。由于是不间断的发布，对公众的服务不可以停，所以Facebook会将一部分的机器从公众服务状态中拿下来，更新之后再放回，然后继续下一批，直到所有机器都被更新。这样就可以保证在任意状态都有足够多的机器来支持用户访问。……工具检测发布过程并且将其进度可视化，你可以很方便的看到哪些服务器更新了，现在正在更新哪些服务器，整个网站的进度是百分之几，等等。

- 数据检测工具

数据收集只要1-2行代码即可完成，数据的整理分类存储皆在后台的上万台服务器上自动完成，数据的可视化报表只需要通过一个页面工具点点鼠标设置即可即时生成，而不需要任何代码；数据波动的自动警报也可以设置，可以自动发邮件发短信。

王淮说到了这些工具背后的基本理念：

“基本理念就是凡是被很多人不断重复的好习惯，要将其自动化，绑定到工具之中。以“Don't make me think”的方式来推广好的 practice。”

当然，他们当时仍使用白板作为最重要的人为工具之一：

“ 把最最重要的目标及相关的任务，目标日期，负责人等信息写到白板上挂到我们最近的墙。每天一抬头就可以看到，每次开会都会路过，时刻提醒我们最最重要的事情是什么。这种工具对我们组非常有效。

在网站支持和内部的通讯工具方面，王淮提到的工具包括：

- 用户问题自动分配工具

Facebook 内部的处理工具做得最重要的事情就是把相关问题自动分配到最相关的运营组 (routing)，比如和支付欺诈相关的问题应当自动分配到反欺诈运维组的那十几个人那边。然后工具会提供常见的通用解决方案，绝大多数的回信内容自动产生 (用户姓名，原问题等个体信息都会自动嵌入)。

如果针对某一个功能的问题突然多起来，工具会自动发现，并提醒运维人员手动查看

所有用户问题的回复率，回复满意度，交互次数等等都会被统计或抽样统计，以保证客服服务的质量。

- 招聘工具

Facebook 有一套专门的做题系统 (Puzzle System) 尝试去筛选可靠的工程师。

所有的内部推荐都是通过专门的人才提交工具来上传简历，这个工具和整个招人系统结合。

当然还有必不可少的权限控制 – 只有参加面试的人员才能够看到关于应聘者整个流程的所有资料。

最后，该工具允许打印所有的相关资料以帮助决策委员会讨论该应聘者时拥有所有相关数据。

- 业绩评价工具

这个工具要允许员工自己对自己评价，员工互相评价，员工和老板之间互评，等等；还要考虑权限的管理。并不是一个很容易开发的工具。这个工具一开始是内部开发，但后来还是决定使用 Rypple 来提供专业的业绩评价工具。

接下来，王淮提供了一些对于工具的思考：

- 工具开发是手段，而不是目的。……如果某个需要的工具有其他更专业的人做得更好的话，Facebook 非常乐意付费买他们的服务，而把自己的精力集中在核心产品上。
- Facebook 希望通过工具的方式来解决所有可能想到的问题……能够想到的地方就尽可能用工具。与物理工具不同，计算机工具可以实现“杠杆效应”的反复累积，通过组合这些“杠杆效应”可以达到更高的层级。
- 工具团队不应该是一个由二线员工组成的“事后诸葛亮”的后勤部门，公司里最有才华的工程师应该用公司自己的工具来工作，并且企业文化里要优先反映这些。当公司过了最一开始开发原型证明概念的萌芽阶段之后，开发出优秀的工具并继续加以改善、更新，这比寻找下一个伟大的创意更重要。

王淮也谈到了工具文化面临的最大挑战：

“ 现实的最大挑战是，工具团队要招聘新员工有一定的难度。Facebook 的用户已经达到数亿，而且还在不断增长，如果你开发的是直接面向用户的产品，每天有那么多人在使用，那带来的成就感非常棒。而你要说服新员工去开发内部工具，说这样可以带给工程师以及其他同事更高的效率、最终帮助公司做出更好的产品，相对是间接并缺乏吸引力的。

他的解决方法是：

- 用一些具体的工具提高效率的案例和数据来做理性说服；这需要在开发工具的同时要检测工具使用前后的效率变化。
- 为了吸引内部最好的人才愿意到工具团队，企业文化中也一定要着重反映出这一点：在不同的公开场合私下会面都不断的强调其重要性，让所有人都清楚，公司将内部工具视为持续的重要投资。
- 集中精力努力说服几位整个工程部门认同的顶尖工程师加入工具组，具有很好的示范效果和磁铁效应。如果真正做到如此重视，最优秀的

工程师是愿意加入工具团队的，可以大大提升同事们的效率，从而更好地服务于用户，这也是一种外部用户所感受不到的成就感。

InfoQ 的读者，你们所在的公司有工具文化么？

原文链接：<http://www.infoq.com/cn/news/2012/08/facebook-tools-culture>

相关内容

- [Facebook 已将 HHVM/JIT 用于其开发和产品中](#)
- [Sean Lynch 谈 Facebook Claspin 监控工具的由来](#)
- [Facebook 如何提高软件质量？](#)
- [Facebook 的海量数据架构演变过程](#)
- [Serkan Piantino 谈 Facebook 的扩展](#)

特别专题

自动化构建：一致性关键之道

作者 Joe Enos 译者 雷慈祥

介绍

如果有那么一件事软件开发人员很在行（并非引述电影《黑客》），那一定是将通常需要人工完成的任务自动化。让计算机处理重复乏味的任务将使得大家生活得更轻松，这里我们讨论的是如何让大家专注于他们所关心的事情。然而，研发团队时常会忽略那个最有帮助的受众—他们自己。

在为数众多的中小型软件作坊中，不存在自动化构建和发布工具。构建、交付准备环境、代码发布全由手工完成，同样还有运行测试、备份旧版本、新版本打标签以及许多其他重复的事情。毕竟你可能认为这全是非常简单的工作，集成开发环境通过按钮或快捷键就可构建项目，你开启两个窗口拖放少许文件或文件夹即可完成网站发布。但当你在维护代码库和应用时所有这些事情加在一起，这里几分钟，那里几分钟，最终会浪费几个小时。

庆幸的是，很容易解决这个问题。基本的自动化构建方案易部署，可高度定制化，成本低廉。本文描述了组建自动化流程的一些动机，以及你将需要接触的一些概念。本系列的第二部分将描述针对.NET 解决方案的具体实现，但这些技术在任何环境下都适用。

我们试图解决什么问题？

在我们深入了解之前，我们先看下一些我们试图解决的问题。并非所有这些问题在你的组织中都存在，但是如果你仔细观察你们团队，你将发现可能存在其中的一些问题，很可能你的构建流程中其余部分可能会耗费一些工作。

不一致的构建：用集成开发环境编写代码是件美妙的事情。然而这是有代价，诸如框架/运行时版本、输出结构、编译/调试版本、配置设置、环境变量以及编译选项等通常是由集成开发环境或操作系统处理。看似好事，但除非大家使用完全

一样的开发机器，如果开发人员不注意这些细节，同一个代码库，不同开发人员会得到不同的输出。

不完整的构建：我们中的大部分人都曾经都对糟糕的源码控制实践感到内疚。我们可能忘了提交一个缺陷修复，或是编译前忘了更新最新代码以获取其他人的改动。当这些发生在人工构建环境下时，即将发布的代码便不是最新的。我们需要一个解决方案，它保证你发布的总是源码控制系统中存在的。

失败的单元测试：单元测试是任何优秀应用程序不可或缺的一部分，如果使用得当，它有助于避免在修复 bug 的同时引入新 bug。然而，光写测试远远不够。你必须定期批量执行所有测试，当有些事需要你自己亲自处理时很容易会被遗忘。实际上没人执行这些测试，所以可能有些单元测试没通过而你却从来不知道。

人为错误：即使是精心策划的简单流程也容易出现人为错误。我们都曾有过手指在键盘上徘徊却意外敲错按键，或者意外删除了操作系统内核（这比你想象地要经常发生）。或是在凌晨 1 点钟，昏昏欲睡时，我们意外地把正式服务器当作测试机开启。这是不可避免的，因为人无完人，所以任何事情一旦需要人工交互都有可能出错。

安全：服务器和网络的安全总是由单独的软件团队处理。通常有两个极端，要么服务器连接有限制，达到这个限制任何人都访问不了，培养新员工的繁文缛节需要一个月之久；要么服务器对所有人开放，那么你们团队任何一个成员的恶意点击都可能搞垮系统。身为一名开发人员，我通常偏向于第二种方案，因为实际上我可以把事情做好，但我看到了这种做事方式的危险性。无论你的组织在该领域处于什么位置，自动化只会改进你的流程。

我们首先需要做什么？

一旦你确定了团队的主要痛点，你可以针对需求设计解决方案。没有放之四海而皆准的方案。只要你有自动构建，它能就减少人为操作并使事情趋于一致，你前进的方向就是对的。

然而，你真正还需要很多东西。

你必须开始接触优秀的源码控制实践。不管你是使用 SVN、Mercurial、GIT 或是 TFS（请不要使用 SourceSafe），你需要定义诸如分支策略，如何处理第三

方以及内部库，如何在仓库中组织项目。当然了，你的团队已经启航。当有人在小项目里做事不遵守规范，他们能搞砸整个流程。

必须有一名开发人员担任自动构建工程师。这个人将负责编写构建脚本，搭建持续集成开发环境，很可能还负责源码控制系统的搭建和部署。除非你有个很大而且复杂的环境，否则这些工作应该只占用他一小部分的时间，所以他一周中大部分时间还能从事常规的开发工作。

即使你希望永远不要用到，流程中还是应该有个应急计划，就像组织中其他的任务关键型程序。编译服务器很可能成为一个单点故障，它很可能不具备负载均衡的能力，而且也不会有热备份以防服务器宕机。像这种情况，你想要确保能够快速搭建新的服务器，完成配置和权限的设定，还应该有个不使用构建服务器的 B 计划。尽管该实现的目标是再也不要由人工完成任何事情，却总应保证它是可行的。

构建脚本

构建流程的自动化依赖于简单的重复性任务。第一步是编写构建脚本。构建脚本可以是以任何形式：批处理文件/shell 脚本、基于 xml 的任务集合、自己写的可配置程序、或是他们中的任意组合。在.NET 世界中，MSBuild 是由微软提供的命令行功能，它使用基于 xml 的项目文件构建 Visual Studio 解决方案。NAnt 是另一个常见.NET 构建脚本工具，类似于流行的 Java 工具 Ant。其他的包含开源社区中常见的 Make，Ruby 中的 Rake 等。

无论你选择如何编写构建脚本，你应该寻找适合你的方法并坚持下去。例如，你一旦找到构建 web 程序项目的最佳方式，为新的 web 应用程序创建构建脚本应该就很简单了，只要从其他项目中拷贝脚本、修改部分名称和路径即可。

操作系统和编程框架之间的实现显然有很大的区别，但理念基本一致。脚本应该完成你在构建/编译/交付准备环境时通常要做的所有事情。通常，这意味着编译代码，使用特定的编译选项，将输出文件和原始代码库分开保存，做部署准备。即便在无须编译的项目中，例如静态网站，项目中可能有不可动态更新的内容，例如测试页面或调试脚本，你想在源码控制系统中统一管理这些内容，但是在你确实想要部署时，构建脚本将文件交付准备环境时不会发布这些内容，所以你无须每一次都要考虑这些事情。

除了基本的构建/编译/交付准备环境等步骤，你其实可以做任何想做的。想压缩 JavaScript 文件？为它创建一个任务。编译之前代码中需要唯一时间戳？创建一个任务。为了不让网站落入对手囊中，需要分型数据加密算法？添加一个任务（请个人帮忙）。在构建流程中，你能在命令行做的都能在代码编译前后执行。

大多数软件项目都不止一个模块。例如，你可能有个 web 应用程序，但你还有个独立的数据库，它是该整体解决方案的一部分。对于这种场景，单个控制脚本是解决的方法。该脚本是控制器，每次调用一个单独脚本。在每个 Visual Studio 项目、Java 包、抑或是自定义的代码结构中都有脚本。每个单独的脚本都有该项目特有的任务，而控制脚本包含了所有共享功能。

尽可能使脚本通用并且可重用。使用相对路径，不要使用绝对路径，具体项目的信息定义在一处，可重用的信息定义在控制脚本中。这么做便于维护，也有助于今后构建新项目。

持续集成

一旦编写了脚本，项目就可以通过一个命令完成编译和交付准备环境。这是个良好的开始，但还需要有人执行命令。我们的目标是去除人工介入，所以持续集成将为我们考虑这些。

至于构建脚本，供选择的有许多种不同的技术，项目的组织方式也多种多样。但再次强调，你将想要找到适合于你的解决方案，并坚持下去，以便在项目中保持一致性。

一些流行的选择是 TeamCity、Jenkins、CruiseControl（或 CruiseControl.NET），如果你喜欢多用途的应用程序，Microsoft Team Foundation Server 除了源码控制和构建之外还能够完成持续集成。每个产品都有各自的目标受众，但是他们全都是设计用来监视源码并自动运行构建脚本，所以不需要手工完成这些。持续集成服务器的传统策略是监视源码控制仓库的变动，当有变动时自动下载最新代码，然后执行脚本，它依次编译应用程序并为发布做准备。然而，在构建脚本中，你能够添加任何你需要的任务或行为。

你可以设置单元测试或任何其他你编写的自动化测试作为该流程的一部分执行。一旦有人提交代码，测试将立即运行。至于测试不通过，更有甚者编译不通过，你会被告知出错了，所以这些问题能够被快速处理。就本人而言，我建议用飞镖射击责任人（戴高帽是另一个不错的选择），但是如何实现当然是由你决定。

发布

到目前为止所有的一切都是为了这一刻。此刻我们只是提交了代码，自动编译了，运行了测试，交付到准备环境和做好了发布的准备。最后一步是将交付准备环境的代码发布到任何需要的地方。

发布自己托管的 web 应用程序通常是将文件从准备环境拷贝到 web 服务器。这可能意味着人工拷贝文件，或整理到一个简单的批处理/shell 脚本中。因为我们仍在努力使生活变得简单并且自动化，你将寻求更佳的解决方案。取决于你们单位的安全策略，你也许能够将持续集成流程中的任务组织在一起，以便通过文件系统或 FTP 拷贝文件。至于一个纯 HTTP 的解决方案，你可以看看 DubDubDeploy 等产品，它不受域安全或文件系统访问的限制，能够服务器之间拷贝文件。

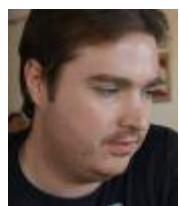
如果你有个包装产品，最后一步只要简单地将产品打包。构建脚本可能已经处理了创建安装包，组织文档，以及其他与发布文件相关的东西。现在有了可部署的产品，剩下的就是将产品拷贝到 3.5 英寸软盘、打包、发布。

总结

组建自动化构建环境可能会花些时间，让它按你想要的方式工作至少需要几天时间，也可能是几个礼拜，可能还要按照你自己的意愿做调整。最后，你需要权衡后续每天将节省的时间，以及那些平时大家很容易掉入而在采用一致的过程后可以有效避免的陷阱。

当每个人都做他们最擅长事情的时候，你的组织会是最高效的。开发人员编写代码，构建工程师处理构建和部署配置，构建服务器完成所有重复性任务，它的确做得很出色。运转流畅的软件过程通常会直接提升产品质量和缩短发布周期，两者都将极大地影响着你公司的财务状况。

关于作者



Joe Enos 是一位软件工程师，同时还是位企业家，在.NET 软件领域拥有近十年的工作经验。他主要关注自动化和过程改进，涉及软件领域的方方面面。他在全美各类软件大会上给小型软件团队做过关于自动化构建的演讲，介绍构建脚本以及持续集

成等话题。

他公司的首款软件产品 [DubDubDeploy](#) 已于近期发布，是帮助改善软件团队管理构建和部署流程的系列产品中的第一款。他的团队目前致力于.NET 构建流程的全自动化。

原文链接：<http://www.infoq.com/cn/articles/Automated-Builds>

相关内容

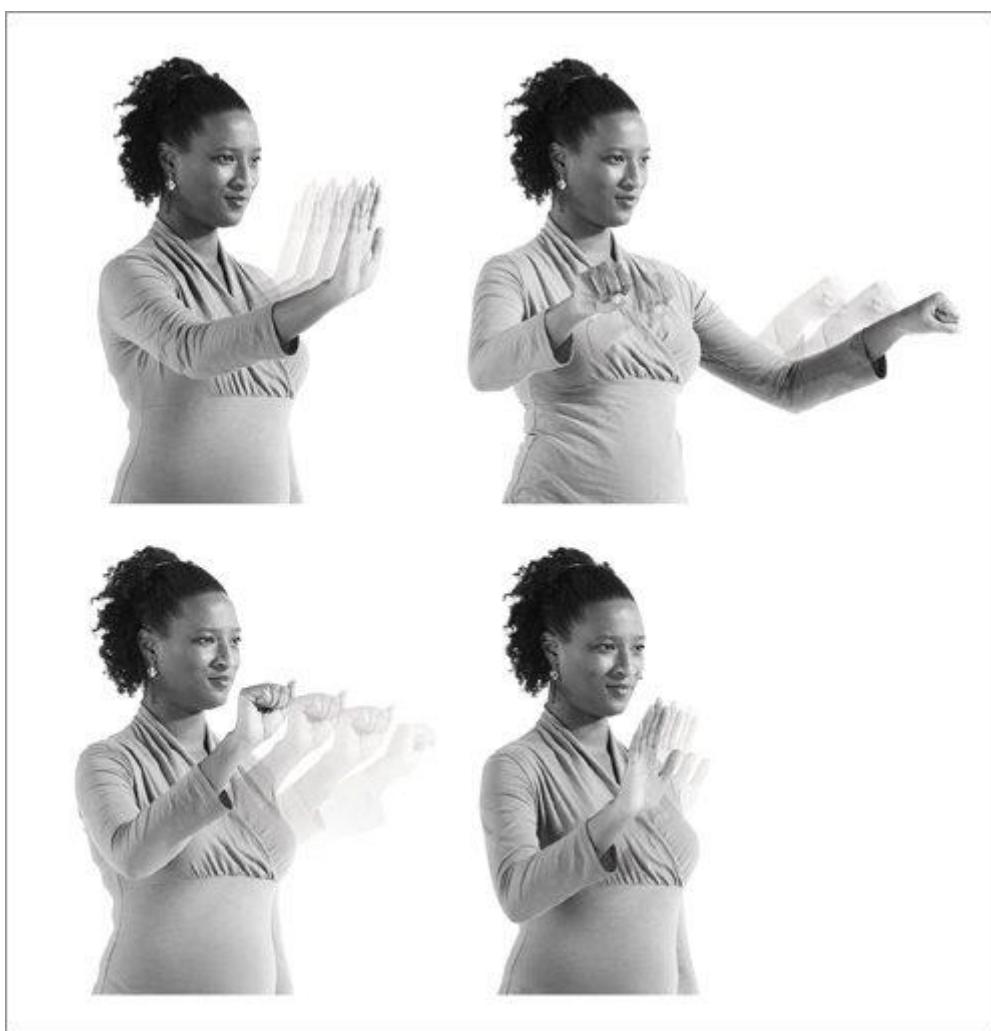
- [用 MSBuild 和 Jenkins 搭建持续集成环境 \(1\)](#)
- [用 MSBuild 和 Jenkins 搭建持续集成环境 \(2\)](#)
- [Grunt 0.4.0 发布：更强调模块化](#)
- [了解 Travis CI：开源的持续集成](#)

本期专栏 | Column

微软发布 Kinect for Windows SDK v1.7

作者 贾国清

微软于近日在其[博客](#)中宣布了最新的 SDK 及开发者工具 (Developer Toolkit) v1.7 , 本次发布了大家期待已久的 Kinect 交互功能 , 主要包含紧握 (Grip) 和按压 (Press) 、 Kinect Fusion (视觉增强) 、更多的 MATLAB 示例以及 OpenCV 等。



从左上开始 , 逆时针方向 : “推按”表示选择 (push to select) , “抓住 (握拳) ”来实现滚动和移动 , 通过“挥手”来识别玩家。双手缩放 (右上) 也是可以实现的。

Kinect 交互 (Kinect Interactions) 可以帮助玩家很直观地进行一些操作，比如手掌向前推几英寸来表示按下屏幕中的按钮，又或者通过握紧手来“抓和移动 (Grip and Pan)”某个区域。对于商务用户来说，进行会议时，可以不限于坐在座位上，可以完全释放自己，在会议室很自然的走动。

本次发布的 Kinect for Windows SDK v1.7 还包含了 Kinect 视觉增强 (Kinect Fusion) 工具，此工具可以实时创建高精度的 3-D 人物和物体效果图。微软也曾经在博客中写到：

“ Kinect Fusion 通过整合来自 Windows 版 Kinect 传感器的连续数据流，重建一个物体或环境的 3D 模型。它允许你以任何人都不能看到的透视角度，采集扫描到的物体或环境的信息。

前段时间，InfoQ 也对[微软开放 Kinect for Windows 样本代码](#)进行了报道，文章提出：

“ 微软在 [CodePlex](#) 上基于 [Apache 2.0](#) 协议开放了 22 个代码样本。这些代码样本也包含在 [Kinect for Windows Toolkit](#) 中，展示了如何利用 Kinect 的各种特性，包括音频、基本交互、颜色、深度、人脸追踪、红外、连续手势、语音、WPF、XNA 等。

Kinect for Windows 相关文档

- [人机界面准则](#)
- [编程指南\(英\)](#)
- [Kinect for Windows 一般讨论\(英\)\(BBS\)](#)
- [下载 Kinect for Windows SDK v1.7](#)

号外：Kinect for Windows 人机交互新体验，免费培训报名中，限额 30 人（成都：3 月 27 日，深圳：3 月 29 日），报名或了解详情请点击：[“码”上 Kinect 吧](#)。InfoQ 也会对其他几站的活动进行报道，欢迎关注[@InfoQ](#) 官方微博账号。

原文链接：<http://www.infoq.com/cn/news/2013/03/k4w-sdk1.7-release>

相关内容

- [Kinect for Windows 培训北京站：了解 K4W 软硬件原理](#)
- [Kinect for Windows 培训营：自平衡机器人项目作者访谈](#)

本期专栏

微软开放 Kinect for Windows 样本代码

作者 [Abel Avram](#) 译者 [廖煜嵘](#)

微软最近在 [CodePlex](#) 上基于 [Apache 2.0 协议](#) 开放了 22 个代码样本。这些代码样本也包含在 [Kinect for Windows Toolkit](#) 中，展示了如何利用 Kinect 的各种特性，包括音频、基本交互、颜色、深度、人脸追踪、红外、连续手势、语音、WPF、XNA 等。

这些样本大部分用 C# 和 C++ 编写，还有一部分用 VB 编写。它们 利用了 WPF 或 DirectX 技术。

虽然可以从 CodePlex 的 [Git 仓库](#) 创建出代码分支，但微软尚>不接受任何开发者贡献代码。他们提到将来如何还在研究。

Kinect 的开发需要使用 Visual Studio 2010 或者 2012、.NET 4.0 或 4.5、[Kinect for Windows SDK](#) 和 [Kinect for Windows Toolkit](#)。

尽管 Kinect 最初是种游戏设备，但目前在[很多项目](#) 中都有应用，或尚在研究，比如 [Autodesk 建模工具](#)、[控制手套](#)(controlling gloves)、3D 手势跟踪、[Google TV 控制器](#)、缓解病人情绪和服装导购等。

原文链接：<http://www.infoq.com/cn/news/2013/03/Microsoft-Kinect-Samples>

相关内容

- [开发者谈微软 Windows Phone 8 之变](#)
- [微软发布 Kinect for Windows SDK v1.7](#)
- [Microsoft 和 AWS 推出免费的云优化服务](#)
- [微软为 Windows Azure 创建基于 Linux 的虚拟机的目录](#)
- [微软 Internet Explorer 10 浏览器 指尖触碰盛放网络之美](#)

本期专题

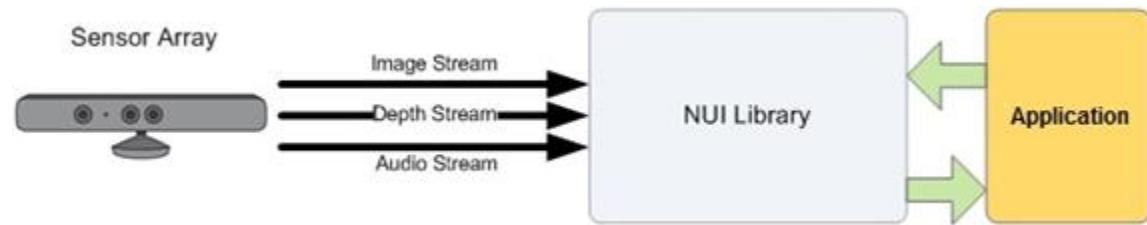
Kinect for Windows 培训北京站：了解 K4W 软硬件原理

作者 贾国清

自去年10月Kinect for Windows在中国发布，微软随后启动“Kinect for Windows加速器计划”之后，上周又启动了名为“码”上Kinect的全国培训活动，本次活动共设北京、南京、上海、成都以及深圳五站，意在普及Kinect for Windows开发，鼓励开发者通过这一新的人机交互方式，开发出更具创新、生产率更高的行业应用。

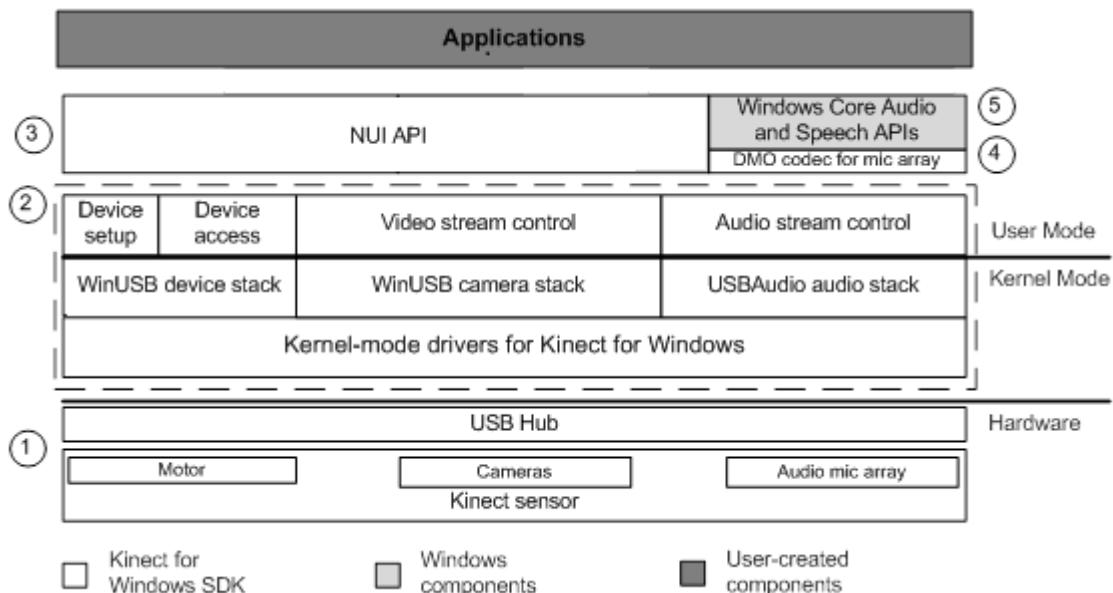
北京站活动于3月20日举行，培训内容主要涵盖Kinect关键技术及交互设计、Kinect Scenarios(NUI、NEO、NUO)、Kinect for Windows SDK v1.7的新特性以及Kinect技术的更多应用及其商业价值探讨等。

SDK主要提供了软件类库和开发工具，从而帮助开发者可以捕获丰富的基于Kinect的自然数据，下图展示了Kinect硬件同软件类库的交互过程：



硬件方面，Kinect for Windows Sensor 主要包含红外线发射器、RGB 摄像头、红外接收器、倾角控制马达、麦克风阵列以及加速计。

在 API 方面，通过添加新 API 增加了开发的易用性和连贯性。例如利用更快的方法来在深度图像和颜色图像之间进行映射。这将带来新的场景，如绿屏侦测和 3-D 点云。此外，Kinect for Windows SDK 可以捕获超过 4 米的扩展深度数据，可针对应用环境优化色彩摄像头。白平衡，对比度，色度，饱和度和其他设定均可微调，添加了一些新的 API 来转换坐标空间的数据：颜色，深度和框架。共有两组 API：一组为转换单个像素，一组为转换整幅图像。更支持红外，Raw Bayer，扩展深度数据和加速器等功能。SDK 架构图展示如下：



主要包含以下组件：

- Kinect 硬件：主要包含 Kinect Sensor 以及与电脑相连的 USB Hub；
- Kinect 驱动：Kinect Windows 驱动，主要包含麦克风阵列驱动，音频和视频驱动（用来处理颜色、深度和骨骼数据）；
- 音频和视频组件：帮助 Kinect 捕捉骨骼移动以及深度图像；
- DirectX Media Object (DMO)相关的数据；
- Windows 7 标准 API，主要包含音频、视频以及多媒体等 API，目前也已支持 Windows 8
- 此外，近距离模式（Near Mode）使深度传感器能够探测到最近 40 厘米范围内的物体，并在此前可实现的范围之外与之通信，获得更多与深度值相关的信息。在色彩和深度的同步，深度到色彩映射和完整 API 等方面也进行了改进。Kinect 可识别人体内的 20 个骨骼，摄像头帧数 30FPS（普通电影 24FPS）。

开发者关注问题

最新的 Kinect for Windows SDK 更新是否提供面部识别支持？

自 1.5 版本开始，Kinect for Windows SDK 已通过面部追踪 SDK 组件支持实时面部追踪。面部识别 SDK 并非专门针对面部识别和辨认。面部识别实时探测和追踪面部的位置和方向，并提供三维网格动画，这些动画结合 Kinect 的深度数据以 3-D 的方式呈现。面部识别能够实时绘制眉毛的位置和嘴型。传统的面部

识别算法能都用于 Kinect 的 RGB 流；我们仍在持续研究 Kinect 传感器的其他可用功能。了解更多关于面部识别的内容

针对 Kinect 密集处理的推荐硬件配置？

针对需要大量使用 Kinect 框架追踪的应用以及定制程序处理，我们建议采用如下的计算机参考配置。我们发现其能够提供大多数高强度追踪场景所需的强大性能，同时也能为其他的程序运行提供合理的帧数。处理器：类型：英特尔酷睿 i5 桌面处理器系列或以上 时钟频率：3.0 GHz 或以上 核心数：4 或以上 内存：4 GB DDR3 1333 或以上 操作系统：64 位版本 Windows 7 或 64 位版本 Windows 8。

原文链接：<http://www.infoq.com/cn/news/2013/03/k4w-roadshow-beijing>

相关内容

- [微软发布 Kinect for Windows SDK v1.7](#)
- [Kinect for Windows 培训营：自平衡机器人项目作者访谈](#)
- [有关京东商城采用.NET 架构的社区讨论](#)
- [微软发布 Windows 管理框架 3.0 测试版](#)
- [WCF 4.5：配置文件更小，对 ASP.NET 的支持更好](#)

推荐文章 | Article

深入理解 Java 内存模型（七）——总结

作者 程晓明

处理器内存模型

顺序一致性内存模型是一个理论参考模型，JMM 和处理器内存模型在设计时通常会把顺序一致性内存模型作为参照。JMM 和处理器内存模型在设计时会对顺序一致性模型做一些放松，因为如果完全按照顺序一致性模型来实现处理器和 JMM，那么很多的处理器和编译器优化都要被禁止，这对执行性能将会有很大的影响。

根据对不同类型读/写操作组合的执行顺序的放松，可以把常见处理器的内存模型划分为下面几种类型：

1. 放松程序中写-读操作的顺序，由此产生了 total store ordering 内存模型（简称为 TSO）。
2. 在前面 1 的基础上，继续放松程序中写-写操作的顺序，由此产生了 partial store order 内存模型（简称为 PSO）。
3. 在前面 1 和 2 的基础上，继续放松程序中读-写和读-读操作的顺序，由此产生了 relaxed memory order 内存模型（简称为 RMO）和 PowerPC 内存模型。

注意，这里处理器对读/写操作的放松，是以两个操作之间不存在数据依赖性为前提的（因为处理器要遵守 as-if-serial 语义，处理器不会对存在数据依赖性的两个内存操作做重排序）。

下面的表格展示了常见处理器内存模型的细节特征：

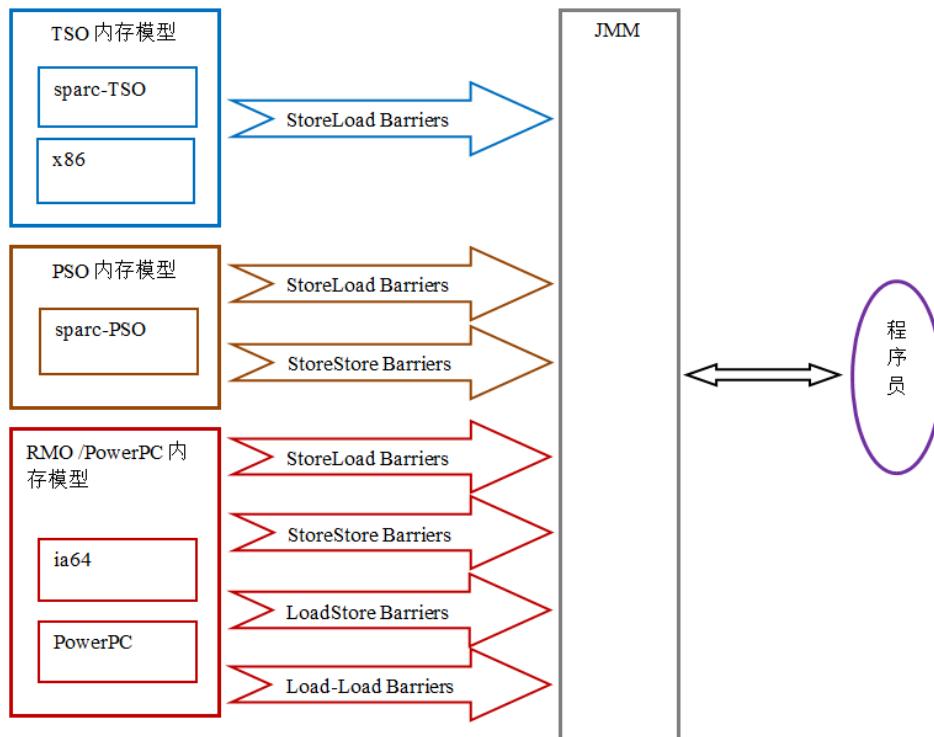
| 内存模型名称 | 对应的处理器 | Store-Load 重排序 | Store-Store 重排序 | Load-Load 和 Load-Store 重排序 | 可以更早读取到其它处理器的写 | 可以更早读取到当前处理器的写 |
|--------|--------|----------------|-----------------|----------------------------|----------------|----------------|
| | | | | | | |

| | | | | | | | |
|---------|-----------|---|---|---|---|--|---|
| TSO | sparc-TSO | Y | | | | | Y |
| | X64 | | | | | | |
| PSO | sparc-PSO | Y | Y | | | | Y |
| RMO | ia64 | Y | Y | Y | | | Y |
| PowerPC | PowerPC | Y | Y | Y | Y | | Y |

在这个表格中，我们可以看到所有处理器内存模型都允许写-读重排序，原因在第一章以说明过：它们都使用了写缓存区，写缓存区可能导致写-读操作重排序。同时，我们可以看到这些处理器内存模型都允许更早读到当前处理器的写，原因同样是因为写缓存区：由于写缓存区仅对当前处理器可见，这个特性导致当前处理器可以比其他处理器先看到临时保存在自己的写缓存区中的写。

上面表格中的各种处理器内存模型，从上到下，模型由强变弱。越是追求性能的处理器，内存模型设计的会越弱。因为这些处理器希望内存模型对它们的束缚越少越好，这样它们就可以做尽可能多的优化来提高性能。

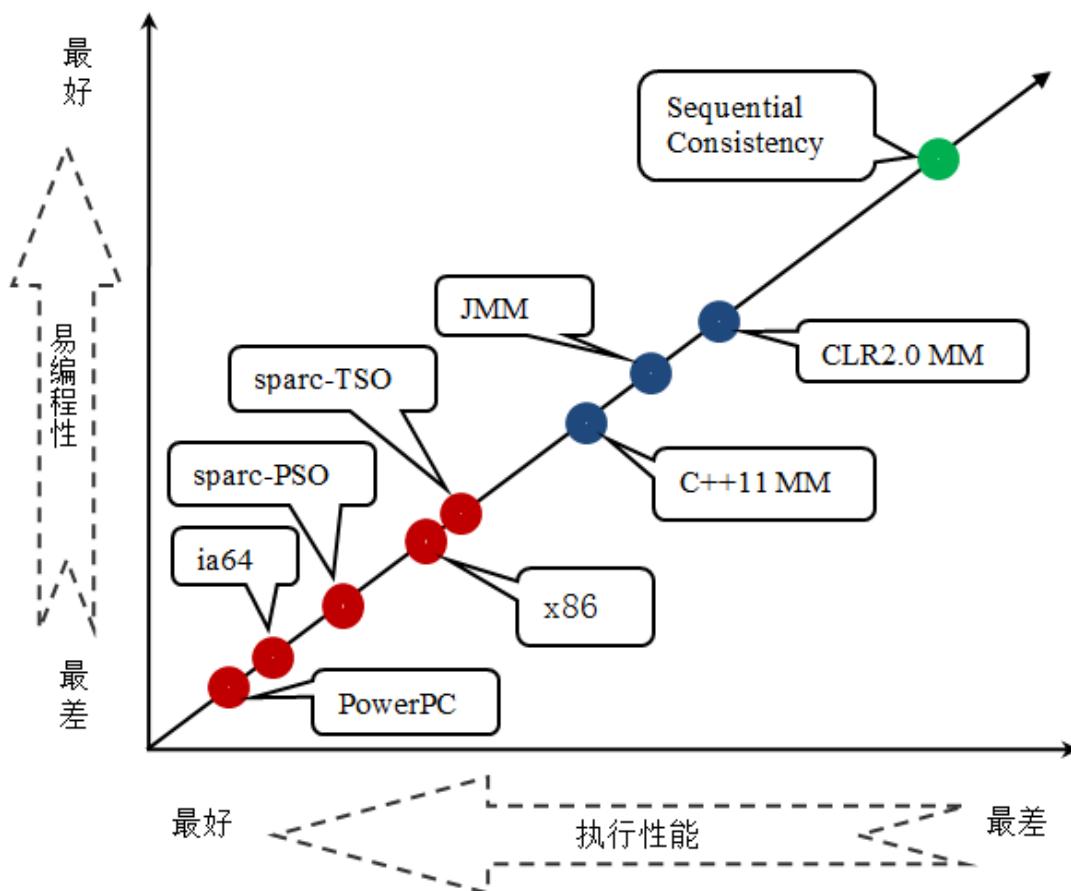
由于常见的处理器内存模型比 JMM 要弱，java 编译器在生成字节码时，会在执行指令序列的适当位置插入内存屏障来限制处理器的重排序。同时，由于各种处理器内存模型的强弱并不相同，为了在不同的处理器平台向程序员展示一个一致的内存模型，JMM 在不同的处理器中需要插入的内存屏障的数量和种类也不相同。下图展示了 JMM 在不同处理器内存模型中需要插入的内存屏障的示意图：



如上图所示，JMM 屏蔽了不同处理器内存模型的差异，它在不同的处理器平台之上为 java 程序员呈现了一个一致的内存模型。

JMM，处理器内存模型与顺序一致性内存模型之间的关系

JMM 是一个语言级的内存模型，处理器内存模型是硬件级的内存模型，顺序一致性内存模型是一个理论参考模型。下面是语言内存模型，处理器内存模型和顺序一致性内存模型的强弱对比示意图：



从上图我们可以看出：常见的 4 种处理器内存模型比常用的 3 中语言内存模型要弱，处理器内存模型和语言内存模型都比顺序一致性内存模型要弱。同处理器内存模型一样，越是追求执行性能的语言，内存模型设计的会越弱。

JMM 的设计

从 JMM 设计者的角度来说，在设计 JMM 时，需要考虑两个关键因素：

- 程序员对内存模型的使用。程序员希望内存模型易于理解，易于编程。
程序员希望基于一个强内存模型来编写代码。
- 编译器和处理器对内存模型的实现。编译器和处理器希望内存模型对它们的束缚越少越好，这样它们就可以做尽可能多的优化来提高性能。
编译器和处理器希望实现一个弱内存模型。

由于这两个因素互相矛盾，所以 JSR-133 专家组在设计 JMM 时的核心目标就是找到一个好的平衡点：一方面要为程序员提供足够强的内存可见性保证；另一方

面，对编译器和处理器的限制要尽可能的放松。下面让我们看看 JSR-133 是如何实现这一目标的。

为了具体说明，请看前面提到过的计算圆面积的示例代码：

```
double pi = 3.14; //A
double r = 1.0; //B
double area = pi * r * r; //C
```

上面计算圆的面积的示例代码存在三个 happens-before 关系：

1. A happens-before B ;
2. B happens-before C ;
3. A happens-before C ;

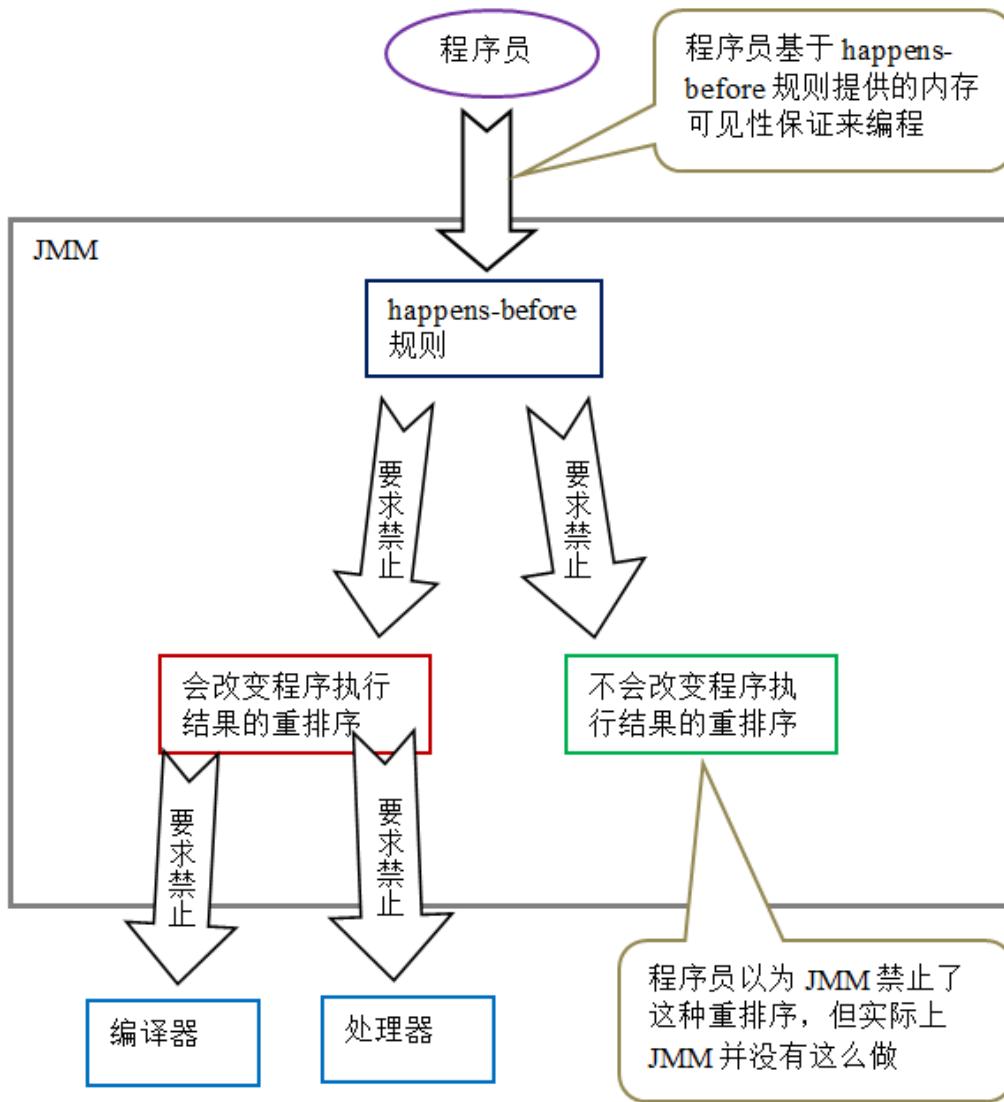
由于 A happens-before B , happens-before 的定义会要求：A 操作执行的结果要对 B 可见，且 A 操作的执行顺序排在 B 操作之前。但是从程序语义的角度来说，对 A 和 B 做重排序即不会改变程序的执行结果，也还能提高程序的执行性能（允许这种重排序减少了对编译器和处理器优化的束缚）。也就是说，上面这 3 个 happens-before 关系中，虽然 2 和 3 是必需要的，但 1 是不必要的。因此，JMM 把 happens-before 要求禁止的重排序分为了下面两类：

- 会改变程序执行结果的重排序。
- 不会改变程序执行结果的重排序。

JMM 对这两种不同性质的重排序，采取了不同的策略：

- 对于会改变程序执行结果的重排序，JMM 要求编译器和处理器必须禁止这种重排序。
- 对于不会改变程序执行结果的重排序，JMM 对编译器和处理器不作要求（JMM 允许这种重排序）。

下面是 JMM 的设计示意图：



从上图可以看出两点：

- JMM 向程序员提供的 happens- before 规则能满足程序员的需求。
JMM 的 happens- before 规则不但简单易懂，而且也向程序员提供了足够强的内存可见性保证（有些内存可见性保证其实并不一定真实存在，比如上面的 A happens- before B）。
- JMM 对编译器和处理器的束缚已经尽可能的少。从上面的分析我们可以看出，JMM 其实是在遵循一个基本原则：只要不改变程序的执行结果（指的是单线程程序和正确同步的多线程程序），编译器和处理器怎么优化都行。比如，如果编译器经过细致的分析后，认定一个锁只会被单个线程访问，那么这个锁可以被消除。再比如，如果编译器经过细致的分析后，认定一个 volatile 变量仅仅只会被单个线程访问，

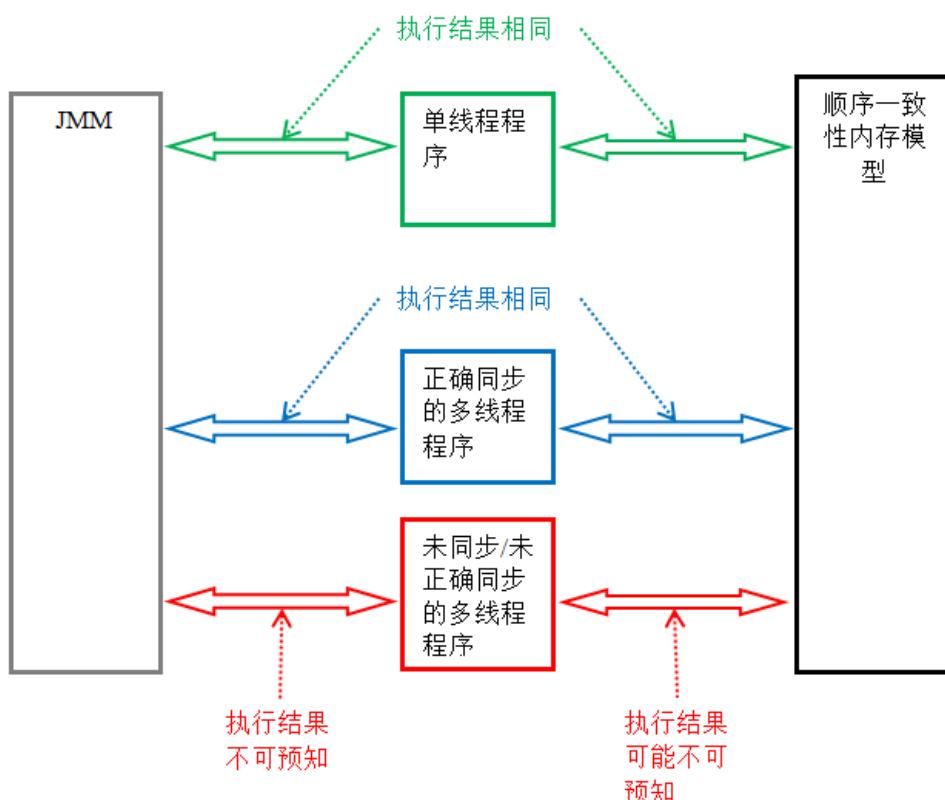
那么编译器可以把这个 volatile 变量当作一个普通变量来对待。这些优化既不会改变程序的执行结果，又能提高程序的执行效率。

JMM 的内存可见性保证

Java 程序的内存可见性保证按程序类型可以分为下列三类：

1. 单线程程序。单线程程序不会出现内存可见性问题。编译器，runtime 和处理器会共同确保单线程程序的执行结果与该程序在顺序一致性模型中的执行结果相同。
2. 正确同步的多线程程序。正确同步的多线程程序的执行将具有顺序一致性(程序的执行结果与该程序在顺序一致性内存模型中的执行结果相同)。这是 JMM 关注的重点，JMM 通过限制编译器和处理器的重排序来为程序员提供内存可见性保证。
3. 未同步/未正确同步的多线程程序。JMM 为它们提供了最小安全性保障：线程执行时读取到的值，要么是之前某个线程写入的值，要么是默认值 (0 , null , false) 。

下图展示了这三类程序在 JMM 中与在顺序一致性内存模型中的执行结果的异同：



只要多线程程序是正确同步的，JMM 保证该程序在任意的处理器平台上的执行结果，与该程序在顺序一致性内存模型中的执行结果一致。

JSR-133 对旧内存模型的修补

JSR-133 对 JDK5 之前的旧内存模型的修补主要有两个：

- 增强 volatile 的内存语义。旧内存模型允许 volatile 变量与普通变量重排序。JSR-133 严格限制 volatile 变量与普通变量的重排序，使 volatile 的写-读和锁的释放-获取具有相同的内存语义。
- 增强 final 的内存语义。在旧内存模型中，多次读取同一个 final 变量的值可能会不相同。为此，JSR-133 为 final 增加了两个重排序规则。现在，final 具有了初始化安全性。

参考文献

1. [Computer Architecture: A Quantitative Approach, 4th Edition](#)
2. [Shared memory consistency models: A tutorial](#)
3. [Intel® Itanium® Architecture Software Developer's Manual Volume 2: System Architecture](#)
4. [Concurrent Programming on Windows](#)
5. [JSR 133 \(Java Memory Model\) FAQ](#)
6. [The JSR-133 Cookbook for Compiler Writers](#)
7. [Java theory and practice: Fixing the Java Memory Model, Part 2](#)

关于作者

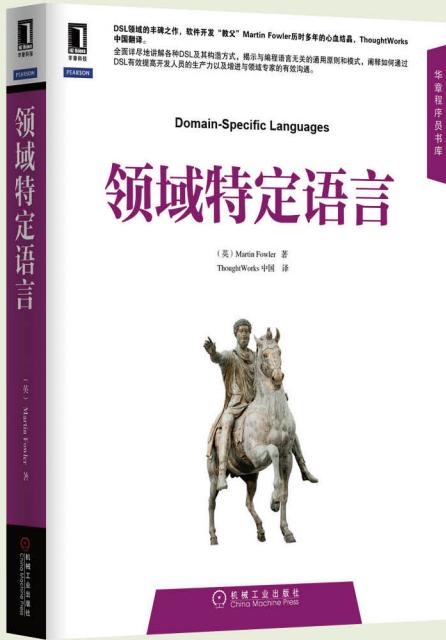
程晓明，Java 软件工程师，国家认证的系统分析师、信息项目管理师。专注于并发编程，就职于富士通南大。个人邮箱：asst2003@163.com。

原文链接：<http://www.infoq.com/cn/articles/java-memory-model-7>

相关内容

- [深入理解 Java 内存模型（六）——final](#)
- [深入理解 Java 内存模型（五）——锁](#)
- [深入理解 Java 内存模型（四）——volatile](#)
- [深入理解 Java 内存模型（三）——顺序一致性](#)
- [深入理解 Java 内存模型（二）——重排序](#)

四|月|新|书|推|荐



《领域特定语言》

作者：Martin Fowler
译者：Thought Works中国
出版时间：2013年3月

DSL领域丰碑之作！软件开发“教父”
Martin Fowler历时多年心血力著！

ThoughtWorks中国译

内容简介

《领域特定语言》是DSL领域的丰碑之作，由世界级软件开发大师和软件开发“教父”Martin Fowler历时多年写作而成，ThoughtWorks中国翻译。全面详尽地讲解了各种DSL及其构造方式，揭示了与编程语言无关的通用原则和模式，阐释了如何通过DSL有效提高开发人员的生产力以及增进与领域专家的有效沟通，能为开发人员选择和使用DSL提供有效的决策依据和指导方法。

作者简介

Martin Fowler，世界级软件开发大师，软件开发“教父”，敏捷开发方法的创始人之一，在面向对象分析与设计、UML、模式、极限编程、重构和DSL等领域都有非常深入的研究并为软件开发行业做出了卓越贡献。他乐于分享，撰写了《企业应用架构模式》（荣获第13届Jolt生产力大奖）、《重构：改善既有代码的设计》、《分析模式：可复用的对象模型》、《UML精粹：标准对象建模语言简明指南》等在软件开发领域颇负盛名的著作。

系统讲解Windows应用商店应用开发的方方面面，涵盖开发环境的搭建，开发工具的使用和应用的发布。

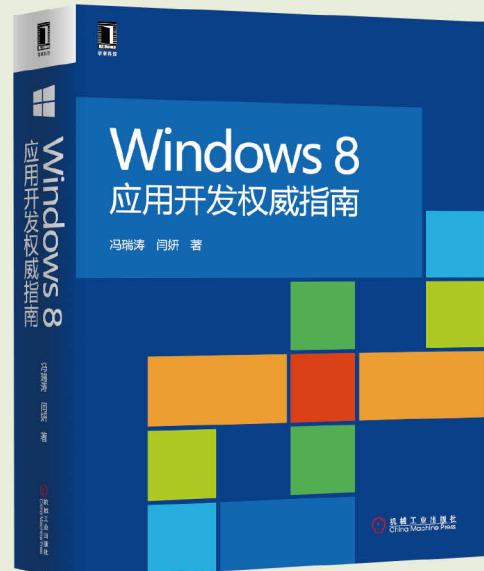
内容简介

《Windows 8应用开发权威指南》应该是目前最为系统、全面、详尽和极具实战性的一本关于Windows应用商店应用开发的著作。书中深刻地阐述了Windows应用商店应用简洁、直观、优雅的界面设计理念和思想，以及开发中的重点和难点。技术内容涵盖了Visual C#、JavaScript和Visual C++开发语言，以满足不同读者群体的需求。为方便读者学习，书中在讲解技术重点的同时辅以完整的示例演示，使读者能够更快地将所学知识运用到实践开发当中，最后还针对不同的开发语言精心设计了3个综合性案例，可操作性极强。

作者简介

冯瑞涛，来自黑龙江省东南部小城鸡西市。在北京生活的8年时间里一直从事软件研发及项目管理工作，对Windows系统相关的开发技术有着浓厚的兴趣，特别是对移动互联网及服务器相关的技术，并为微软强大且易用的开发工具和服务器软件而着迷，业余时间喜欢与社区的伙伴们分享和交流技术心得。多年忙碌的工作没有让他放弃学习，也没有忘记上学时曾许下写一本图书的愿望。

闫妍，软件开发工程师，专注于移动互联网和云计算，对移动终端设备应用的界面设计及自动化测试有深入研究，熟悉WindowsPhone、Android、iOS平台应用开发技术。



《Windows 8 应用开发权威指南》

作者：冯瑞涛，闫妍
出版时间：2013年3月

推荐文章

开发者需要了解的 WebKit

作者 彭超

[Paul Irish](#)是著名的前端开发工程师，同时他也是Chrome开发者关系团队成员，jQuery 团队成员，Modernizr、Yeoman、CSS3 Please 和 HTML5 Boilerplate 的 lead developer。针对大家对 WebKit 的种种误解，他在自己的博客发表了[《WebKit for Developers》](#)一文，试图为大家解惑。

对许多开发者来说，WebKit 就像一个黑盒。我们把 HTML、CSS、JS 和其他一大堆东西丢进去，然后 WebKit 魔法般的以某种方式把一个看起来不错的网页展现给我们。但事实上，Paul 的同事 Ilya Grigorik 说：

 WebKit 才不是个黑盒。它是个白盒。并且，它是个打开的白盒。

所以让我们来花些时间了解这些事儿：

- 什么是 WebKit？
- 什么不是 WebKit？
- 基于 WebKit 的浏览器是如何使用 WebKit 的？
- 为什么又有不同的 WebKit？

现在，特别是 Opera 宣布将浏览器引擎转换为 WebKit 之后，我们有很多使用 WebKit 的浏览器，但是我们很难去界定它们有哪些相同与不同。下面我争取为这个谜团做些解读。而你也将会更懂得判断浏览器的不同，了解如何在正确的地方报告 bug，还会了解如何在特定浏览器下高效开发。

标准 Web 浏览器组件

让我们列举一些现代浏览器的组件：

- HTML、XML、CSS、JavaScript 解析器
- Layout
- 文字和图形渲染
- 图像解码

- GPU 交互
- 网络访问
- 硬件加速

这里面哪些是 WebKit 浏览器共享的？差不多只有前两个。其他部分每个 WebKit 都有各自的实现，所谓的“port”。现在让我们了解一下这是什么意思……

WebKit Ports 是什么？

在 WebKit 中有不同的“port”，但是这里允许我来让 WebKit hacker , Sencha 的工程主管 Ariya Hidayat 来解释：

“WebKit 最常见的参考实现是 Apple 在 Mac OS X 上的实现（这也是[最早和最原始的 WebKit 库](#)）。但是你也能猜到，在 Mac OS X 下，许多不同的接口在很多不同的原生库下被实现，大部分集中在[CoreFoundation](#)。举例来说，如果你定义了一个纯色圆角的按钮，WebKit 知道要去哪里，也知道要如何去绘制这个按钮。但是，绘制按钮的工作最终还是会落到[CoreGraphics](#)去。

上面已经提到，CoreGraphics 只是 Mac port 的实现。不过 Mac Chrome 用的是[Skia](#)。

“随时间推移，WebKit 被“port”（移植）到了各个不同的平台，包括桌面端和移动端。这种做法被称作“WebKit port”。对 Windows 版 Safari 来说，Apple 通过[\(有限实现的\) Windows 版本 CoreFoundation](#) 来 port WebKit。

.....不过 Windows 版本的 Safari [现在已经死掉了](#)。

除此之外，还有很多很多其它的“port”（[参见列表](#)）。Google 创建并维护着它的 Chromium port。这其实也是一个基于 Gtk+ 的 WebKitGtk。诺基亚通过收购 Trolltech，维护着以[QtWebKit module](#) 而闻名的 WebKit Qt port。

让我们看看其中一些 WebKit ports：

- Safari
 - OS X Safari 和 Windows Safari 使用的是不同的 port
 - 用于 OS X Safari 的 WebKit Nightly 以后会渐渐成为一个边缘版本
- Mobile Safari

- 在一个私有代码分支上维护，不过代码现在正在[合并到主干](#)
- iOS Chrome(使用了 Apple 的 WebView , 不过后面的部分有很多不同)
- Chrome (Chromium)
 - 安卓 Chrome (直接使用 Chromium port)
 - Chromium 也驱动了 [Yandex Browser](#)、[360 Browser](#)、[Sogou Browser](#)，很快，还会有 Opera。
- [还有很多](#)： Amazon Silk、Dolphin、Blackberry、QtWebKit、WebKitGTK+、The EFL port (Tizen)、wxWebKit、WebKitWinCE.....

不同的 port 专注于不同的领域。 Mac 的 port 注意力集中在浏览器和操作系统的分割上，允许把 ObjectC 和 C++ 绑定并嵌入原生应用的渲染。 Chromium 专注在浏览器上。 QtWebKit 的 port 在他的跨平台 GUI 应用架构上给 apps 提供运行时环境或者渲染引擎。

WebKit 浏览器共享了那些东西？

首先，让我们来看看这些 WebKit ports 的共同之处：

(作者注 : 很有意思 , 这些内容我写了很多次 , 每次 Chrome 团队成员都给我纠正错误 , 正如你看到的.....)

1. “WebKit 在使用相同的方式解析 WebKit。”——实际上 , Chrome 是唯一支持多线程 HTML 解析的 port。
2. “一旦解析完成 , DOM 树也会构建成相同的样子。”——实际上 Shadow DOM 只有在 Chromium 才被开启。所以 DOM 的构造也是不同的。自定义元素也是如此。
3. “WebKit 为每个人创建了‘window’对象和‘document’对象。”——是的 , 尽管它暴露出的属性和构造函数可以通过 [feature flags](#) 来控制。
4. “CSS 解析都是相同的。将 CSS 解析为对象模型是个相当标准的过程。”——不过 , Chrome 只支持-webkit-前缀 , 而 Apple 和其他的 ports 支持遗留的-khtml-和-apple-前缀。
5. “布局定位 ? 这些是基本设计问题啊”—— 尽管 Sub-pixel layout 和 saturated layout 算法是 WebKit 的一部分 , 不过各个 port 的实现效果还是有很多不同。

所以，情况很复杂。

就像 Flickr 和 GitHub 通过 flag 标识来实现自己的功能一样，WebKit 也有相同处理。这允许各个 port 自行决定是否启用 [WebKit 编译特性标签](#) 的各种功能。通过 [命令行开关](#)，或者通过 [about:flags](#) 还可以控制是否通过运行时标识来展示功能特性。

好，现在让我们再尝试一次搞清楚 WebKit 究竟有哪些相同…

每个 WebKit port 有哪些共同之处

- DOM、window、document
- CSS 对象模型
- CSS 解析，键盘事件处理
- HTML 解析和 DOM 构建
- 所有的布局和定位
- Chrome 开发工具和 WebKit 检查器的 UI 与检查器
- contenteditable、pushState、文件 API、大多数 SVG、CSS Transform math、Web Audio API、localStorage 等功能
- [很多其他功能与特性](#)

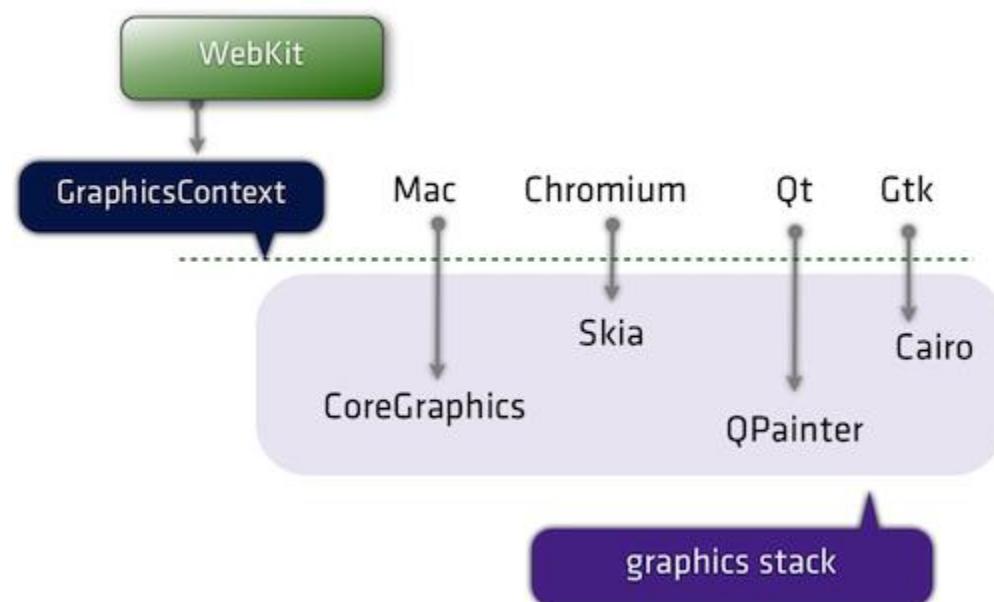
这些领域现在有点儿模糊，让我们尝试把事情弄得更清楚一点。

什么是 WebKit port 们并没有共享的：

- GPU 相关技术
 - 3D 转换
 - WebGL
 - 视频解码
- 将 2D 图像绘制到屏幕
 - 解析方式
 - SVG 和 CSS 渐变绘制
- 文字绘制和断字
- 网络层（SPDY、预渲染、WebSocket 传输）
- JavaScript 引擎

- JavaScriptCore 在 WebKit repo 中。V8 和 JavaScriptCore 被绑定在 WebKit 中。
- 表单控制器的渲染
- <video>和<audio>的元素表现和解码实现
- 图像解码
- 页面导航 前进/后退
 - pushState()的导航部分
- SSL 功能，比如 Strict Transport Security 和 Public Key Pins

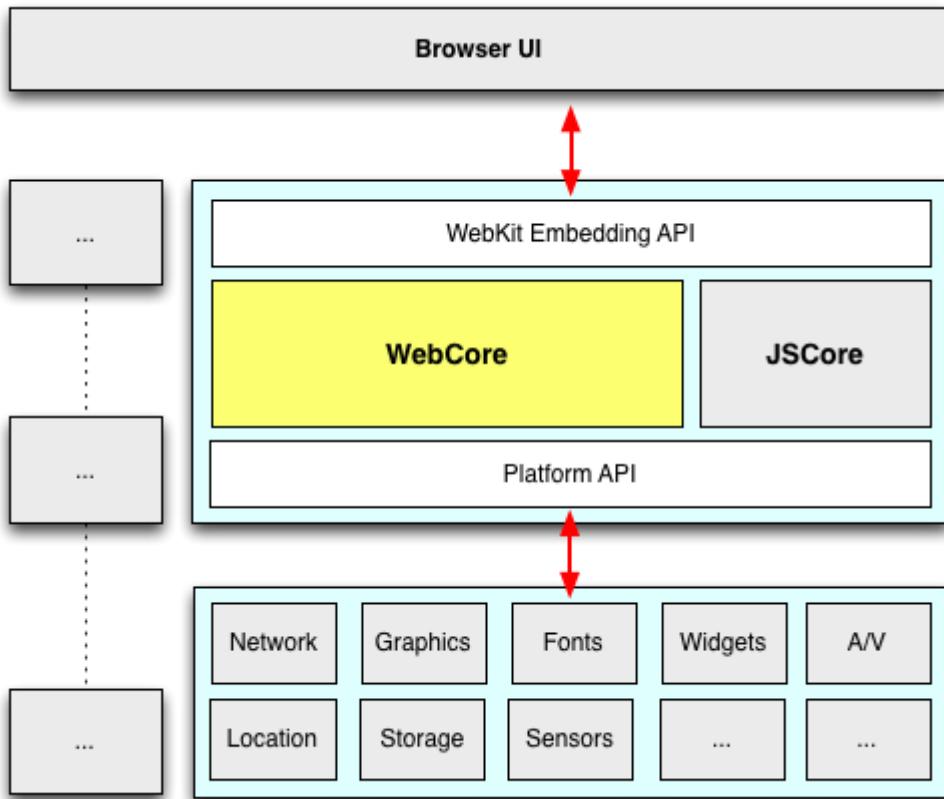
让我们谈谈其中的 2D 图像部分：根据 port 的不同，我们使用完全不同的库来处理图像到屏幕的绘制过程：



更宏观一点来看，一个最近刚添加的功能：CSS.supports()在除了没有 css3 特性检测功能的 win 和 wincairo 这两个 port 之外，在其它所有 port 中都可用。

现在到了卖弄学问的技术时间。上面讲的内容其实并不正确。事实上那是 WebCore 被共享的东西。而 WebCore 其实是当大家讨论 HTML 和 SVG 的布局、渲染和 DOM 处理时提到的 WebKit。技术上讲，WebKit 是 WebCore 和各种 ports 之间的绑定层，尽管通常来说这个差别并不那么重要。

一个图表应该可以帮助大家理解：



WebKit 中的许多组件都是可以更换的（图中标灰色的部分）。

举个例子来说，Webkit 的 JavaScript 引擎，JavaScriptCore，是 WebKit 的默认组件。（它最初是当 WebKit 从 KHTML 分支时从 KJS 演变来的）。同时，Chromium port 用 V8 引擎做了替换，还使用了独特的 DOM 绑定来映射上面的组件。

字体和文字渲染是平台上的重要部分。在 WebKit 中有两个独立的文字路径：Fast 和 Complex。这两者都需要平台特性的支持，但是 Fast 只需要知道如何传输字型，而 Complex 实际上需要掌握平台上所有的字符串，并说“请绘制这个吧”。

"WebKit 就像一个三明治。尽管 Chromium 的包装更像是一个墨西哥卷。一个美味的 Web 平台墨西哥卷。"

—— Dimitri Glazkov, Chrome WebKit hacker, Web Components 和 Shadow DOM 拥护者。

现在，让我们放宽镜头看看一些 port 和一些子系统。下面是 WebKit 的 5 个 port；尽管它们共享了 WebCore 的大部分，但考虑一下它们的 stack 有哪些不同。

| | | | | | |
|--|---------------|---------------|----------|-----------------|----------------|
| | Chrome (OS X) | Safari (OS X) | QtWebKit | Android Browser | Chrome for iOS |
|--|---------------|---------------|----------|-----------------|----------------|

| | | | | | |
|------------|------------------------|----------------|----------------------------------|----------------------------------|------------------------------------|
| Rendering | Skia | CoreGraphics | QtGui | Android stack/Skia | CoreGraphics |
| Networking | Chromium network stack | CFNetwork | QtNetwork | Fork of Chromium's network stack | Chromium stack |
| Fonts | CoreText via Skia | CoreText | Qt internals | Android stack | CoreText |
| JavaScript | V8 | JavaScriptCore | JSC (V8 is used elsewhere in Qt) | V8 | JavaScriptCore (without JITting) * |

*iOS Chrome 注：你可能知道它使用 UIWebView。由于 UIWebView 的能力限制。它只能使用移动版 Safari 的渲染层，JavaScriptCore（而不是 V8）和单进程模式。然而，大量的 Chromium 代码还是起到了调节作用，比如网络层、同步、书签架构、地址栏、度量工具和崩溃报告。（同时，由于 JavaScript 很少成为移动端的瓶颈，缺少 JIT 编译器只有很小的影响。）

好吧，那么我们该怎么办？

现在所有 WebKit 完全不同了，我好怕。

别这样！[WebKit 的 layoutTests 覆盖面](#)非常广（据最新统计，有 28,000 个 layoutTests），这些 test 不仅针对已存在的特性，而且针对任何发现的回归。实际上，每当你探索一些新的或难懂的 DOM/CSS/HTML5 特性时，在整个 web 平台上，layoutTests 经常已经有了一些奇妙的小 demo。

另外，W3C 正在努力研究一致性测试套件。这意味着我们可以期待使用同一个测试套件来测试不同的 WebKit port 和浏览器，以此来获得更少的怪异模式，和一个带来更少的怪癖模式和更具互操作性的 web。对所有参加过 [Test The Web Forward](#) 活动的人们……致谢！

Opera 刚刚迁移到了 WebKit 了。会怎样？

Robert Nyman 和 Rob Hawkes 也[谈到了这个](#)，但是我会再补充一些：Opera 在公告中明显指出 Opera 将采用 Chromium。这意味着 WebGL, Canvas, HTML5 表单，2D 图像实现——Chrome 和 Opera 将在所有这些功能上保持一致。API

和后端实现也会完全相同。由于 Opera 是基于 Chromium , 你可以有足够的信心去相信你的尖端工作将会在 Chrome 和 Opera 上获得兼容。

我还应该指出 ,所有的 Opera 浏览器都将采用 Chromium :包括他的 Windows , Mac、Linux 版本 ,和 Opera Mobile(完全成熟的移动浏览器)。甚至 Opera Mini 都将使用基于 Chromium 的服务器渲染集群来替换当前的基于 Presto 的服务器端渲染。

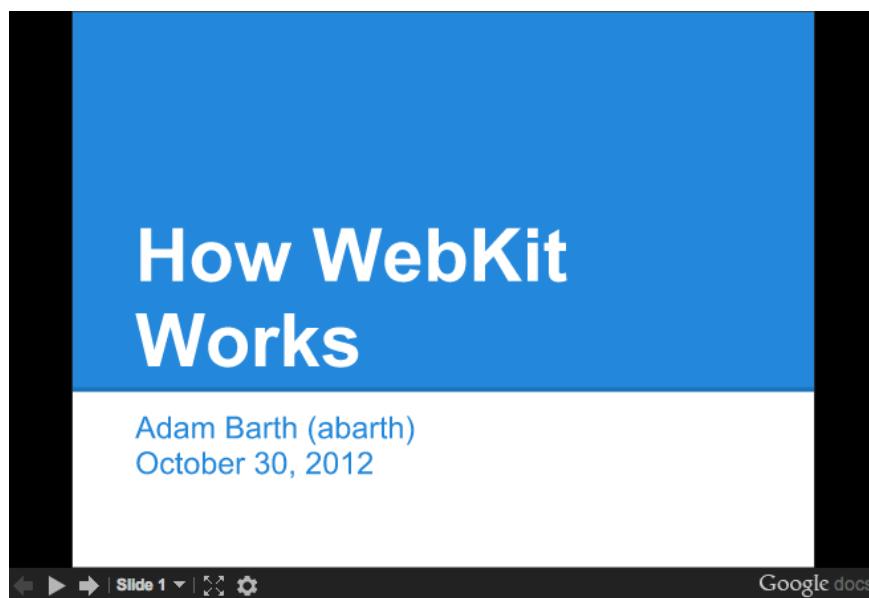
.....那 WebKit Nightly 是什么 ?

它是 WebKit 的 [mac_port](#) , 和 Safari 运行的二进制文件一样 (尽管会替换一些底层库) 。因为苹果在项目中起主导地位 , 所以它的表现和功能与 Safari 的总是那么一致。在很多情况下 , 当其它 port 可能会试验新功能的时候 , Apple 却显得相对保守。不管怎样 , 如果你想我用中学一样的类比 , 想想这个好了 : WebKit Nightly 对于 Safari 就像 Chromium 对于 Chrome.

同样的 , [Chrome Canary](#) 有着最新的 WebKit 资源。

告诉我更多的 WebKit 内幕吧。

就在这儿了 , 小伙子 :



扩展阅读 :

- [WebKit internals technical articles | webkit.org](#)

- [WebKit: An Objective View](#) | Robert Nyman & Rob Hawkes (译者注：本文在 InfoQ 有授权发布的中文译文)
- [your webkit port is special \(just like every other port\)](#) | Ariya Hidayat
- [Getting Started With the WebKit Layout Code](#) | Adobe Web Platform Blog
- [WebKit Documentation Overview](#) | Arun Patole
- [Rendering in WebKit](#), by Eric Seidel | YouTube
- [web performance for the curious](#) | Ilya Grigorik

原文链接：<http://www.infoq.com/cn/articles/webkit-for-developers>

相关内容：

- [都用 WebKit 也并不意味 Web 的统一：WebKit 的前世今生](#)
- [玉伯、寒冬、老赵和大城小胖谈 WebKit](#)
- [Web 趋向统一？Opera 宣布浏览器引擎将切换至 WebKit](#)
- [jQuery 作者 John Resig：WebKit 就是浏览器引擎中的 jQuery](#)
- [傲游勾三股四解析 HTML5 中的 Web Storage 使用](#)

推荐文章

聊聊并发（三）——JAVA 线程池的分析和使用

作者 方腾飞

1. 引言

合理利用线程池能够带来三个好处。第一：降低资源消耗。通过重复利用已创建的线程降低线程创建和销毁造成的消耗。第二：提高响应速度。当任务到达时，任务可以不需要等到线程创建就能立即执行。第三：提高线程的可管理性。线程是稀缺资源，如果无限制的创建，不仅会消耗系统资源，还会降低系统的稳定性，使用线程池可以进行统一的分配，调优和监控。但是要做到合理的利用线程池，必须对其原理了如指掌。

2. 线程池的使用

线程池的创建

我们可以通过 ThreadPoolExecutor 来创建一个线程池。

```
new ThreadPoolExecutor(corePoolSize, maximumPoolSize, keepAliveTime, milliseconds, runnableTaskQueue, handler)
```

创建一个线程池需要输入几个参数：

- corePoolSize（线程池的基本大小）：当提交一个任务到线程池时，线程池会创建一个线程来执行任务，即使其他空闲的基本线程能够执行新任务也会创建线程，等到需要执行的任务数大于线程池基本大小时就不再创建。如果调用了线程池的 prestartAllCoreThreads 方法，线程池会提前创建并启动所有基本线程。
- runnableTaskQueue（任务队列）：用于保存等待执行的任务的阻塞队列。可以选择以下几个阻塞队列。
 - ArrayBlockingQueue：是一个基于数组结构的有界阻塞队列，此队列按 FIFO（先进先出）原则对元素进行排序。
 - LinkedBlockingQueue：一个基于链表结构的阻塞队列，此队列按 FIFO（先进先出）排序元素，吞吐量通常要高于

ArrayBlockingQueue。静态工厂方法

Executors.newFixedThreadPool()使用了这个队列。

- SynchronousQueue :一个不存储元素的阻塞队列。每个插入操作必须等到另一个线程调用移除操作 ,否则插入操作一直处于阻塞状态 ,吞吐量通常要高于 LinkedBlockingQueue ,静态工厂方法 Executors.newCachedThreadPool 使用了这个队列。
- PriorityBlockingQueue :一个具有优先级的无限阻塞队列。
- maximumPoolSize (线程池最大大小) :线程池允许创建的最大线程数。如果队列满了 ,并且已创建的线程数小于最大线程数 ,则线程池会再创建新的线程执行任务。值得注意的是如果使用了无界的任务队列这个参数就没什么效果。
- ThreadFactory :用于设置创建线程的工厂 ,可以通过线程工厂给每个创建出来的线程设置更有意义的名字。
- RejectedExecutionHandler (饱和策略) :当队列和线程池都满了 ,说明线程池处于饱和状态 ,那么必须采取一种策略处理提交的新任务。这个策略默认情况下是 AbortPolicy ,表示无法处理新任务时抛出异常。以下是 JDK1.5 提供的四种策略。
 - AbortPolicy :直接抛出异常。
 - CallerRunsPolicy :只用调用者所在线程来运行任务。
 - DiscardOldestPolicy :丢弃队列里最近的一个任务 ,并执行当前任务。
 - DiscardPolicy :不处理 ,丢弃掉。
 - 当然也可以根据应用场景需要来实现 RejectedExecutionHandler 接口自定义策略。如记录日志或持久化不能处理的任务。
- keepAliveTime (线程活动保持时间) :线程池的工作线程空闲后 ,保持存活的时间。所以如果任务很多 ,并且每个任务执行的时间比较短 ,可以调大这个时间 ,提高线程的利用率。
- TimeUnit (线程活动保持时间的单位) :可选的单位有天 (DAYS) ,小时 (HOURS) ,分钟 (MINUTES) ,毫秒(MILLISECONDS) ,微秒(MICROSECONDS, 千分之一毫秒)和毫微秒(NANOSECONDS, 千分之一微秒)。

向线程池提交任务

我们可以使用 execute 提交的任务，但是 execute 方法没有返回值，所以无法判断任务是否被线程池执行成功。通过以下代码可知 execute 方法输入的任务是一个 Runnable 类的实例。

```
threadsPool.execute(new Runnable() {
    @Override
    public void run() {
        // TODO Auto-generated method stub
    }
});
```

我们也可以使用 submit 方法来提交任务，它会返回一个 future，那么我们可以通过这个 future 来判断任务是否执行成功，通过 future 的 get 方法来获取返回值，get 方法会阻塞住直到任务完成，而使用 get(long timeout, TimeUnit unit)方法则会阻塞一段时间后立即返回，这时有可能任务没有执行完。

```
Future<Object> future = executor.submit(harReturnValuetask);
try {
    Object s = future.get();
} catch (InterruptedException e) {
    // 处理中断异常
} catch (ExecutionException e) {
    // 处理无法执行任务异常
} finally {
    // 关闭线程池
    executor.shutdown();
}
```

线程池的关闭

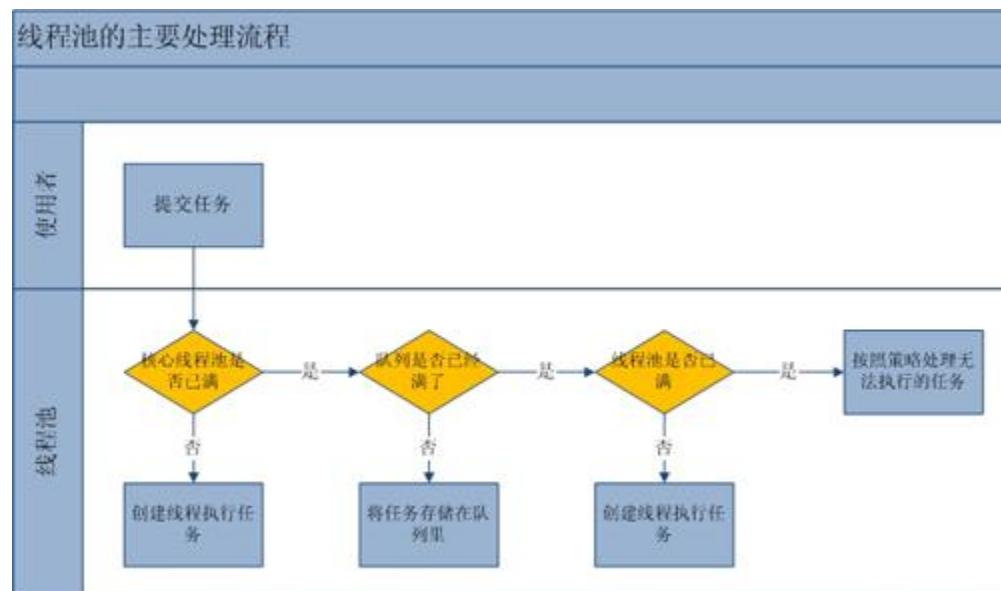
我们可以通过调用线程池的 shutdown 或 shutdownNow 方法来关闭线程池，它们的原理是遍历线程池中的工作线程，然后逐个调用线程的 interrupt 方法来中断线程，所以无法响应中断的任务可能永远无法终止。但是它们存在一定的区别，shutdownNow 首先将线程池的状态设置成 STOP，然后尝试停止所有的正在执行或暂停任务的线程，并返回等待执行任务的列表，而 shutdown 只是将线程池的状态设置成 SHUTDOWN 状态，然后中断所有没有正在执行任务的线程。

只要调用了这两个关闭方法的其中一个，isShutdown 方法就会返回 true。当所有的任务都已关闭后，才表示线程池关闭成功，这时调用 isTerminated 方法会返回 true。至于我们应该调用哪一种方法来关闭线程池，应该由提交到线程池的任

务特性决定，通常调用 shutdown 来关闭线程池，如果任务不一定要执行完，则可以调用 shutdownNow。

3. 线程池的分析

流程分析：线程池的主要工作流程如下图：



从上图我们可以看出，当提交一个新任务到线程池时，线程池的处理流程如下：

- 首先线程池判断基本线程池是否已满？没满，创建一个工作线程来执行任务。满了，则进入下个流程。
- 其次线程池判断工作队列是否已满？没满，则将新提交的任务存储在工作队列里。满了，则进入下个流程。
- 最后线程池判断整个线程池是否已满？没满，则创建一个新的工作线程来执行任务，满了，则交给饱和策略来处理这个任务。

源码分析。上面的流程分析让我们很直观的了解了线程池的工作原理，让我们再通过源代码来看看是如何实现的。线程池执行任务的方法如下：

```

public void execute(Runnable command) {
    if (command == null)
        throw new NullPointerException();
    //如果线程数小于基本线程数，则创建线程并执行当前任务
    if (poolSize >= corePoolSize || !addIfUnderCorePoolSize(command)) {
        //如线程数大于等于基本线程数或线程创建失败，则将当前任务放到工作队列中。
        if (runState == RUNNING && workQueue.offer(command)) {
            if (runState != RUNNING || poolSize == 0)
                ensureQueuedTaskHandled(command);
        }
        //如果线程池不处于运行中或任务无法放入队列，并且当前线程数量小于最大允许的线程数量，则创建一个线程执行任务。
        else if (!addIfUnderMaximumPoolSize(command))
            //抛出RejectedExecutionException异常
            reject(command); // is shutdown or saturated
    }
}
    
```

工作线程。线程池创建线程时，会将线程封装成工作线程 Worker，Worker 在执行完任务后，还会无限循环获取工作队列里的任务来执行。我们可以从 Worker 的 run 方法里看到这点：

```

public void run() {
    try {
        Runnable task = firstTask;
        firstTask = null;
        while (task != null || (task = getTask()) != null) {
            runTask(task);
            task = null;
        }
    } finally {
        workerDone(this);
    }
}
    
```

4. 合理的配置线程池

要想合理的配置线程池，就必须首先分析任务特性，可以从以下几个角度来进行分析：

1. 任务的性质：CPU 密集型任务，IO 密集型任务和混合型任务。
2. 任务的优先级：高，中和低。
3. 任务的执行时间：长，中和短。
4. 任务的依赖性：是否依赖其他系统资源，如数据库连接。

任务性质不同的任务可以用不同规模的线程池分开处理。CPU 密集型任务配置尽可能小的线程，如配置 Ncpu+1 个线程的线程池。IO 密集型任务则由于线程并不是一直在执行任务，则配置尽可能多的线程，如 2*Ncpu。混合型的任务，

如果可以拆分，则将其拆分成一个 CPU 密集型任务和一个 IO 密集型任务，只要这两个任务执行的时间相差不是太大，那么分解后执行的吞吐率要高于串行执行的吞吐率，如果这两个任务执行时间相差太大，则没必要进行分解。我们可以通过 Runtime.getRuntime().availableProcessors()方法获得当前设备的 CPU 个数。

优先级不同的任务可以使用优先级队列 PriorityBlockingQueue 来处理。它可以让优先级高的任务先得到执行，需要注意的是如果一直有优先级高的任务提交到队列里，那么优先级低的任务可能永远不能执行。

执行时间不同的任务可以交给不同规模的线程池来处理，或者也可以使用优先级队列，让执行时间短的任务先执行。

依赖数据库连接池的任务，因为线程提交 SQL 后需要等待数据库返回结果，如果等待的时间越长 CPU 空闲时间就越长，那么线程数应该设置越大，这样才能更好的利用 CPU。

建议使用有界队列，有界队列能增加系统的稳定性和预警能力，可以根据需要设大一点，比如几千。有一次我们组使用的后台任务线程池的队列和线程池全满了，不断的抛出抛弃任务的异常，通过排查发现是数据库出现了问题，导致执行 SQL 变得非常缓慢，因为后台任务线程池里的任务全是需要向数据库查询和插入数据的，所以导致线程池里的工作线程全部阻塞住，任务积压在线程池里。如果当时我们设置成无界队列，线程池的队列就会越来越多，有可能会撑满内存，导致整个系统不可用，而不只是后台任务出现问题。当然我们的系统所有的任务是用的单独的服务器部署的，而我们使用不同规模的线程池跑不同类型的任务，但是出现这样问题时也会影响到其他任务。

5. 线程池的监控

通过线程池提供的参数进行监控。线程池里有一些属性在监控线程池的时候可以使用

- taskCount：线程池需要执行的任务数量。
- completedTaskCount：线程池在运行过程中已完成的任务数量。小于或等于 taskCount。

- `largestPoolSize` : 线程池曾经创建过的最大线程数量。通过这个数据可以知道线程池是否满过。如等于线程池的最大大小，则表示线程池曾经满了。
- `getPoolSize`: 线程池的线程数量。如果线程池不销毁的话，池里的线程不会自动销毁，所以这个大小只增不减。`getActiveCount` : 获取活动的线程数。

通过扩展线程池进行监控。通过继承线程池并重写线程池的 `beforeExecute`, `afterExecute` 和 `terminated` 方法，我们可以在任务执行前，执行后和线程池关闭前干一些事情。如监控任务的平均执行时间，最大执行时间和最小执行时间等。这几个方法在线程池里是空方法。如：

```
protected void beforeExecute(Thread t, Runnable r) { }
```

6. 参考资料

- Java 并发编程实战。
- JDK1.6 源码

作者介绍

方腾飞，花名清英，淘宝资深开发工程师，关注并发编程，目前在广告技术部从事无线广告联盟的开发和设计工作。个人博客：<http://ifeve.com>

原文链接：<http://www.infoq.com/cn/articles/java-threadPool>

相关内容

- [聊聊并发（三）——JAVA 线程池的分析和使用](#)
- [深入理解 Java 内存模型（六）——final](#)
- [深入理解 Java 内存模型（五）——锁](#)
- [聊聊并发（六）——ConcurrentLinkedQueue 的实现原理分析](#)
- [聊聊并发（五）——原子操作的实现原理](#)

推荐文章

TeamToy2 的开发故事：面向移动的 API 设计，代码重用，快速迭代

作者 杨赛

2013 年的元旦后，一个名叫 [TeamToy](#) 的项目再次进入了人们的视野。跟 BaseCamp、TeamBox、Asana 等服务类似，TeamToy 是一个面向团队的 Todo 类产品，目的在于促进团队沟通、提升协同做事的效率。

TeamToy 项目的启动在 2008 年前后。早期的 TeamToy 在产品设计层面走过很多弯路，也积累了很多经验。TeamToy 的作者陈理捷专门为此写过一篇[产品备忘录](#)，透露了很多 TeamToy 产品诞生与发展的故事。对于 TeamToy 项目本身，2013 年是个极大的转折点：

1. 整体架构进行了重写，重写后的项目发布为 TeamToy2。
2. 有了一个[靓丽的产品主页](#)，不仅分别提供了自架设版本的源码下载和[新浪云商店版本的一键安装入口](#)，而且还加入了大量产品介绍和 FAQs 等说明。
3. 整个项目的代码[转移到了 GitHub 上](#)，从代码本身到整个开发流程，都彻底开源。
4. 版本迭代速度较之前大大加快，采用了“小步快跑”的模式。
5. 专门针对智能手机发布了移动版本：[TeamToyPocket](#)，并且也完全开源。

目前，TeamToy2 仍处于快速更新、完善产品的阶段，用户基数也在不断增长。其移动版本 TeamToyPocket 刚刚[在昨天宣布开源](#)，为此，InfoQ 中文站联系到了 TeamToy 的作者陈理捷，请他谈谈整个 TeamToy2 开发中的一些想法和经验，以及移动版本的选型、整个项目开源这一决策的考量。

陈理捷（[@Easy](#)），80 后，宅，重度萌物控。现任新浪云平台产品主管，负责过 SAE、新浪移动云、新浪云商店、微盘等产品。个人博客：《[方糖气球](#)》。

InfoQ：从 TeamToy v1 到 v2，整体进行了重写。做出这个决定，都考虑到哪些因素？

Easy :

最主要的是考虑到移动端的强烈需求，这要求提供客户端可用的 API，Web 页面需要进行自适应设计，而 TeamToy V1 写于 08 年，并没有面向 API 设计，使用的一些周边库也比较陈旧。另外就是产品设计上，TeamToy v1 更多的是去追求功能，而 V2 更多的是希望简捷，整个前端界面的风格都需要重做。

考虑这些因素后，我觉得重写成本虽然不低，但更利于项目的长期发展。实际上在重写过程中 我还顺便把它所用的框架 [LazyPHP](#) 升级到 3，换上了 JQuery、Bootstrap 等主流的前端库和框架，最后节省了大量的时间。

InfoQ : 在重写 TeamToy 的过程中，有没有哪些地方觉得自己跟写第一版的时候相比的处理更好，或者眼界更高的情况？

Easy :

重写开始时我就定下了几个规矩，这是之前在第一个版本上后悔莫及的：(1) 完全面向 API 设计，即使是 Web 版，也调用标准 API 接口实现。(2) 坚持写单元测试，这样在项目中后期，上线新功能后才能监测对原有功能是否有影响。(3) 让项目支持一键升级，这样才能将 BUG 的危害降到最低。

第一点保证了系统的扩展性，收获了不少的第三方插件；第二和第三则保证了项目的快速迭代，现在从一个 bug 修复到用户代码更新通常只需要 1~2 个小时，这点让 TeamToy 更具竞争力。

其实写第二版时，我并没有比第一版聪明多少，但血泪教训让我保持了更好的习惯，所以开发起来轻松了很多。

InfoQ : 从 TeamToy v2 发布以来，开发的迭代速度很快（Github 上的 commit 频率从 1 月开始就很高）。目前在这个项目上的时间规划是如何安排的？

Easy :

因为 TeamToy 主要是用业余时间来开发，所以我在功能上一直坚持“小步快跑”和“需求拉动”的原则。可能你已经注意到了，TeamToy2 直接使用 reversion 当版本号，即使 3~5 行的代码的一个更新，也会作为一个版本发布出来。一方面这样产品的质量会更高，另一方面也更容易获得成就感，Bug 这种东西，即使是很小的几个，清理掉以后会感觉世界都变美好了。在需求管理上，我一般是只有

用户多次遇到的小需求才会慢慢迭代进去，而大块的需求一般会考虑很久，最后放到长假时去做 (T__T)

InfoQ：TeamToy Pocket 选用了 PhoneGap 进行打包。在做这个选型时，都考虑过哪些因素，PhoneGap 的优势在哪里？

Easy :

选择用 PhoneGap 进行打包，最主要还是因为采用 HTML5 技术可以重用大量的代码；比如在 TeamToyPocket 的 TODO 页面，TODO 的样式和事件绑定的 JS 其实重用了 Web 版的。当然也有其他的打包工具，但显然 PhoneGap 是其中发展最快的一个，实际上借助 PhoneGap，TeamToyPocket 不但可以支持 Android 和 iOS 平台，新的 Tizen 平台、甚至 Mac 桌面都是支持的。对于一个以跨平台为卖点的客户端来讲，这是非常重要的。

InfoQ：移动版的开发有没有遇到什么难点？如何解决的？

Easy :

最开始使用了 JQueryMobile 做框架，但实测时发现整体表现相当差，于是后来我自己重写了一个简单的框架，主要思想是通过 div 切换来避免 ajax 载入，实际使用起来流畅度好了不少。

另外因为 PhoneGap 实际上是使用移动设备默认的浏览器内核，所以要处理各种兼容性问题。比如 Android2.3 和 4.1 以后对 touch 事件不同处理方式。

不过整体上来讲，采用 HTML5 开发还是很自由的，可以很容易的实现自己想要的效果。

InfoQ：对于开源，有些人觉得是一种文化信仰，有些人觉得就是一种开发协作的模式。你如何定义开源？你认为开源对 TeamToy 这个项目最大的好处是什么？

Easy :

开源对我来讲，更多的是一种推动我们积累知识向前进步的方式。

为什么这么说呢？现在的很多协作是基于开放平台的，Facebook 和新浪微博上有很多的应用，让我们的工作和生活都更加方便了。但如果 Facebook 倒闭了呢？

如果新浪微博把你的 APPKey 收回了呢？我们就瞬间回到了原始时代，而第三方在这些平台上花费的精力就完全报废了。

但 TeamToy 不一样，因为通过开源，我释放了控制权，使得任何人都可以自行架设它。只要你能找到服务器，你就能继续你已经习惯了的高效工作环境。不会因为开发商的问题而无法继续；而你为 TeamToy 贡献的所有插件，都会成为整个 TeamToy 用户群的财富。

这就是开源对 TeamToy 的意义，它保证了这个项目不会退步，保证了用户得到的不会失去。

原文链接：<http://www.infoq.com/cn/articles/teamtoy2>

相关内容：

- [HTML5 在移动平台上比原生应用更具优势](#)
 - [朱坤谈移动应用开发的优秀法则](#)
 - [jQuery Mobile 1.3.0 发布](#)
 - [移动开发那些事儿：半数开发者已经在项目中使用了 HTML5](#)
 - [在移动应用程序上少有建树的企业正计划有所行动](#)
-

新品推荐 | Product

详解 Java 7 中新的文件 API

作者 [Dmitriy Rogatkin](#) 译者 [臧秀涛](#)

Java 7 向语言中引入了一些有用的特性 ,其中包括一个新的 I/O 文件包。相对于老的 java.io 包 ,这个包针对文件系统——特别是基于 POSIX 的系统——提供了粒度更细的控制功能。本文首先介绍一下新的 API ,之后通过一个基于 Web 的文件管理器项目 WebFolder 来详细探索这些 API。

原文链接 : <http://www.infoq.com/cn/articles/java7-nio2>

Spring for Apache Hadoop 1.0 发布

作者 [Bienvenido David III](#) 译者 [孙镜涛](#)



发布了 Spring for Apache Hadoop 1.0。开发者能够通过它编写基于 Spring Framework 的 Hadoop 应用 ,还能很容易地与 Spring Batch 和 Spring Integration 集成。Spring for Apache Hadoop 是 Spring Data 大型项目的一个子项目 ,它基于开源的 Apache 2.0 许可发布。

原文链接 :

<http://www.infoq.com/cn/news/2013/03/spring-for-apache-hadoop-1.0>

Visual Studio 2012 Update 2 最终预览版发布

作者 [Jeff Martin](#) 译者 [廖煜嵘](#)



期待已久的 VS2012/TFS2012 Update 2 最终预览版本已经发布。该版本带来了更多的更新 ,其中包括非常重要的新 LightSwitch HTML 客户端。

原文链接 : http://www.infoq.com/cn/news/2013/03/VS2012_CTP4

基于 GitHub 的 jQuery 插件资源库业已发布

作者 [Tim Heckel](#) 译者 [邵思华](#)



1月16日，jQuery Foundation发布了新版插件资源库，以期能够为jQuery核心代码库的第三方开发带来更好的支持与促进。

原文链接：<http://www.infoq.com/cn/news/2013/03/jquery-github-plugin-repo>

微软和亚马逊在Android方面的最新消息

作者 [Abel Avram](#) 译者 [李彬](#)

微软Azure平台发布Android SDK，支持对Android设备推送通知。亚马逊发布移动广告的API，可在任意Android设备上展示亚马逊的广告，并可与Google AdMob的API共存。

原文链接：

<http://www.infoq.com/cn/news/2013/03/Microsoft-Azure-Amazon-Android>

Node.js v0.10 版本发布

作者 [Zef Hemel](#) 译者 [雷慈祥](#)



Node.js研发团队发布了node.js v0.10版本，它是个基于Javscript、用于构建高性能异步服务器的平台。该版本主要更新如下：更易于使用的数据流处理模块，通过域更好地处理错误，此外还带来了性能方面的提升。该团队还宣布在v0.10之后、v1.0之前还会发布一个更稳定版本v0.12。

原文链接：<http://www.infoq.com/cn/news/2013/03/node.js-0.10-released>

Lienzo 1.0：HTML5 Canvas 元素的 Java 版本场景图 API

作者 [Kostis Kapelonis](#) 译者 [雷慈祥](#)

Lienzo 1.0是个全新的GWT库，它基于HTML5 Canvas提供了一套高级API。除了GWT中Canvas已提供的低级操作之外，Lienzo添加了一套丰富的GUI元素，例如图形、缩放、动画、拖放、事件处理等等。

原文链接：<http://www.infoq.com/cn/news/2013/03/lienzo>

Eclipse RAP 2.0 发布——首词相同，含义不同

作者 [Fabian Lange](#) 译者 [廖煜嵘](#)

StartFragment 2013 年 1 月 11 日，在经过 6 年的开发后，Eclipse RAP 2.0 终于发布。InfoQ 采访了项目负责人以了解关于新版本的情况。

原文链接：<http://www.infoq.com/cn/news/2013/03/eclipse-rap-2-released>

jQuery Mobile 1.3.0 发布

作者 [Ralph Winzinger](#) 译者 [廖煜嵘](#)



jQuery 基金会发布了旗下的 JavaScript 和 HTML5/CSS 框架 jQuery Mobile 1.3.0。更新主要集中在响应式的 web 设计并新增了多个移动应用的 widget。

原文链接：<http://www.infoq.com/cn/news/2013/03/jquery-mobile-1.3.0>

Red Gate 发布 ASP.NET MVC 学习门户网站——Simple Web Dev

作者 [Anand Narayanaswamy](#) 译者 [廖煜嵘](#)

StartFragment 位于英国的 Red Gate 软件公司发布了新的门户网站，允许学生和开发者通过门户网站提供的大量资源学习 ASP.NET MVC。

原文链接：<http://www.infoq.com/cn/news/2013/03/red-gate-simply-web-dev>

Google 即将发布 Go 语言 1.1 版，含多项重大更新



作者 [孙镜涛](#)

继 2012 年 3 月 Google 发布 Go 语言的第一个正式版本 Go 1 之后，时隔一年，Google 将于近期发布 Go 1.1。通过从 Google 网站内收集到的信息，我们了解到新版本保持了对旧版本的兼容性，同时还添加了一些重大的语言特性，修改了大量类库，

并且改善了编译器、类库和运行时的实现。

原文链接：<http://www.infoq.com/cn/news/2013/03/google-go-1-1>

Scrum Primer 网站发布卡通版本的 Scrum 总览图

作者 [Shane Hastie](#) 译者 [李彬](#)

《Scrum Primer》的作者们发布了该书中一幅 Scrum 流程图解的卡通版本，该版本及原始图解均可遵循知识共享协议（CC）下载和使用。

原文链接：<http://www.infoq.com/cn/news/2013/03/anime-scrum-primer>

Visual Studio2012 代码审阅特性的对比、注释、意见及状态更新功能

作者 [Anand Narayanaswamy](#) 译者 [陈菲](#)



Visual Studio2012 中增加了代码文件对比、注释添加、审阅意见添加以及状态更新等新功能，这些功能的增加简化了代码审阅的工作。

原文链接：

<http://www.infoq.com/cn/news/2013/03/code-review-visual-studio-2012>

Eclipse Foundation 首次发布 Hudson

作者 [Alex Blewitt](#) 译者 [邵思华](#)



在分支出 Jenkins 之后，Eclipse Foundation 首次发布了 Hudson。新版本带来了哪些变化，为何间隔了这么长时间？InfoQ 采访了 Eclipse Hudson 项目的领导人 Winston Prakash，期待得出这些问题的解答。

原文链接：<http://www.infoq.com/cn/news/2013/03/hudson-eclipse>

基于 Canvas 的图表类库 Chart.js 0.1 发布

作者 [Tim Heckel](#) 译者 [孙镜涛](#)

Nick Downie 于 3 月 17 日发布了一款基于 canvas 的 JavaScript 图表类库 Chart.js，该类库基于 MIT 开源许可发布，是 SVG 图表类库的一种替代方案。

原文链接：<http://www.infoq.com/cn/news/2013/03/chartjs-v.0.1-released>

Grunt 0.4.0 发布：更强调模块化

作者 [Tim Heckel](#) 译者 [邵思华](#)



2 月 18 日，Grunt 团队将他们的 JavaScript 任务运行器（task runner）升级至 0.4.0 版，继续为了实现将整个库解耦为多个模块的目标而努力。

原文链接：

<http://www.infoq.com/cn/news/2013/03/Grunt-0.4.0-Released>

Concurrent 发布 Lingual——一种用于 Hadoop 的领域专用语言

作者 [Boris Lublinsky](#) 译者 [夏雪](#)

Concurrent 股份有限公司是一家企业级大数据应用平台公司，该公司近期发布了 Lingual，它是一个开源项目，它能够使 Apache Hadoop 上的大数据应用开发可以快速、简单地使用 SQL。

原文链接：<http://www.infoq.com/cn/news/2013/03/Lingual>

GCC 4.8 发布，完成向 C++的迁移

作者 [Jeff Martin](#) 译者 [臧秀涛](#)

GCC 4.8 已经完全用 C++ 实现，除了内部改进，还增加了对 C++11 和 Go 的支持，并引入了新的编译器优化。

原文链接：http://www.infoq.com/cn/news/2013/03/gcc48_released

推荐编辑 | 翻译团队编辑张卫滨



大家好，我是张卫滨，很荣幸获邀成为本期《架构师》的推荐编辑，非常开心能够通过这个媒介与大家相识。

刚刚入行之时，我就经常浏览 InfoQ 站点，在这里获取了很多的业内前沿资讯和知识，很多编辑更是我入行时非常仰慕的人，像 SpringSide 开源项目的作者肖桦、技术译者丁雪

丰等等，尤其是 InfoQ 著名的“劳模”张龙，在我未出校门之时就因求实 BBS 的一篇文章而得知这位优秀的校友了。但真正与 InfoQ 的朋友结识还是在 2011 年 5 月，当时 InfoQ 来大连组织一项技术活动，从而有幸与泰稳、伯薇结识。后来，伯薇着手推进大连的技术社区工作，于是更加熟悉了起来。随后，在伯薇的引荐下，发表了一些原创的文章，在 2012 年的下半年开始参与 InfoQ 的翻译工作至今。

参与 InfoQ 的工作之后，深感这是一个严谨务实的团队。如果说在这样一个社会大环境中，“靠谱”是一种稀缺品质的话，那么这个团队正在通过自己的努力，为业界传递着“靠谱”的知识和资讯，推动这个行业的进步。进入团队之后，我才知道，为了一个单词或一句话的翻译，众多的编辑和译者可以在邮件列表或 QQ 群中展开热烈的讨论，从而保证译文能够准确表达原文的含义。原创团队则时刻把握着国内外最新的行业技术动态，为读者奉献及时而深刻的报道。

在严谨务实的同时，这又是一个轻松而开放的团队，每个人都有自己的爱好和个性，都可以按照自己的特长来领取任务，当有疑惑和困难时，随时可以找到热心提供帮助的战友。大家都是凭借兴趣和热情参与到 InfoQ 中文站的建设中来，为了一个好点子能够付诸实施而群策群力，你难道不觉得这是一件很美好的事情吗？如果心动了的话，你也可以加入进来！只要你有热情，这真的不难！

崔健在一期访谈中，曾经这样说：“中国现在需要特别好的质量。各方面的质量，高质量的牛奶，高质量的房子，高质量的桥，高质量的路，高质量的文化。”这句话对我的震撼很大，希望我们能够享用到更多高质量的产品，也希望我们每个人都有高质量的产出。借这句话，自勉，共勉！

非常荣幸能够加入这个团队和大家一起学习成长，以上就是我的一些个人感悟，
大家可以通过新浪微博：@张卫滨 1895 或邮件：levinzhang1981@gmail.com
联系到我。如果您有意给 InfoQ 中文站投稿或是加入我们，请邮件至
editors@cn.infoq.com。

封面植物 | 卫矛

卫矛，又叫鬼箭羽、鬼箭、六月凌、四面锋、蓖箕柴、四棱树、山鸡条子、四面



戟、见肿消、麻药。为卫矛科植物卫矛的具翅状物的枝条或翅状附属物。全年可采，割取枝条后，除去嫩枝及叶，晒干。或收集其翅状物，晒干。有破血通经；解毒消肿；杀虫之功效。产于我国东北、华北、西北至长江流域各地；日本、朝鲜也有分布。适应性强，耐寒，耐荫，耐修剪，耐

干旱、瘠薄。对二氧化硫有较强抗性。生长较慢。嫩叶及霜叶均紫红色，在阳光充足处秋叶鲜艳可爱，，蒴果宿存很久，也颇美观；常植于庭院观赏。

灌木，高约 2—3 米。小枝四棱形，有 2—4 排木栓质的阔翅。叶对生，叶片倒卵形至椭圆形，

长 2—5 厘米，宽 1—2 . 5 厘米，两头尖，很少钝圆，边缘有细尖锯齿；早春初发时及初秋霜后变紫红色。花黄绿色，径约 5—7 毫米，常 3 朵集成聚伞花序。蒴果棕紫色，深裂成 4 裂片，有时为 1—3 裂片；种子褐色，有桔红色的假种皮。花期 4—6 月，果熟期 9—10 月。

木翅入药，称“鬼箭羽”，有破血、止痛、通经、泻下、杀虫等功效；种子油做工业用油。全株含卫矛醇、糖类等成分。

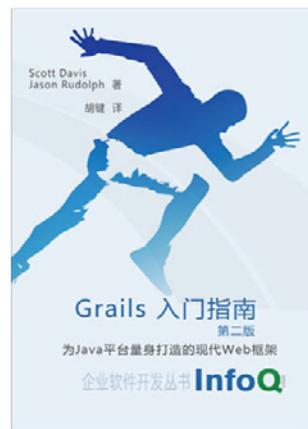
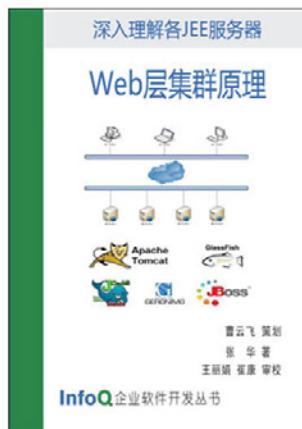
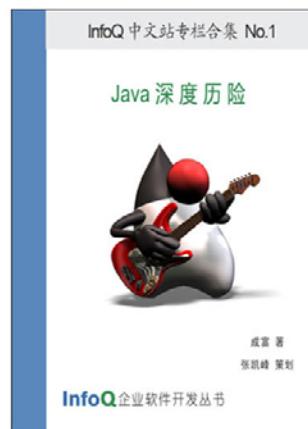
1kg.org 多背一公斤

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 4 月刊

每月 8 日出版

本期主编：贾国清

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>

商务合作：sales@cn.infoq.com 15810407783



本期主编：贾国清，InfoQ 中文站总编

是 InfoQ 中文站主编，毕业于北京工业大学，土木工程本科，硕士转行投入软件工程与系统设计方向。热爱生活，喜欢旅游和体育运动，热衷于关注苹果 (Apple) 产品的动向和使用，现主要负责 InfoQ 中文站的内容和活动策划以及商务项目执行等工作，内容上尤其专注 HTML5、移动开发等方面。