

时刻关注企业软件开发领域的变化与创新

架构师

1月 ARCHITECT



程立谈大规模
SOA系统

ASP.NET的未来

Eclipse插件Top30

开发者如何
保鲜自己的技能

被高估的云计算

构建自动化
驶向何方

继续，高质量的内容创造

新的一年开始了，首先祝《架构师》的所有读者朋友新年快乐，抖擞精神，再踏新程！

过去的 2009 年对 InfoQ 中文站来说是值得在“史书”上记下一笔的一年，在近 40 名编辑的努力下，网站的内容得到越来越多国内中高端技术人员的认可，突出表现即是访问量的逐日升高，去年 12 月份的月独立访问人数已经超过 10 万人。《架构师》的单期下载量也超过 15,000 人，不包括其他技术论坛社区提供的下载。在 2009 年 4 月份、9 月份分别举行的 QCon 全球企业开发大会（北京站）和敏捷中国大会也得到参会者的首肯，尤其是 QCon 北京大会被誉为目前“国内最高技术水平的大会”。等等这一切在很多人看来，InfoQ 中文站正在创造一个不大不小的奇迹。

为什么一个专注于企业软件开发这么一个窄众领域，而且内容全是原创的网媒技术媒体，竟然能够在国内生存下来，而且生存的还算不错？在我看来，这件事情本身并没有什么秘密可言，它只是很好地遵循了一个媒体生存的基本法则——内容为王。在开始筹备 InfoQ 中文站的时候，很多朋友说，不去转载，没有大的内容量，很难支撑一个网络媒体的发展，也不会引起客户的注意。现在来看，这个观点有待商榷，海量的内容固然可以吸引读者的注意力，但是高质量的精品内容可能更是读者所关注的，估计这也是为什么《南方周末》要比许多日报的发行量要大的主要原因。“读者是客户，他们用时间购买了我们的作品。因此，作为回报，他们希望提供的信息是尽可能精确和高效的。考虑到这个原则会引导并确保我们所写的新闻精确而且富有价值，所以我们应该仔细检查每一个句子，保证它们的必要性。”是 InfoQ 中文站的新闻价值观里的一句话，自建站伊始，编辑在类似这些原则的指导下，抵制住转载的诱惑，致力于高质量原创内容的创造。事实证明，这样做我们先是取悦了自己——在做一件有价值的事情；然后取悦了读者——一定的时间内获得了更多的信息；而且合作的客户好像也高兴了——找到了适合宣传的平台。

新的一年开始了，InfoQ 中文站也抖擞精神，再次上路！尽管国内非法转载依然猖獗，互联网环境依然糟糕，但我们创造高质量内容的理想没变，而且与亲爱的读者您同行，靠谱！

本期主编：霍泰稳

目 录

[篇首语]

继续，高质量的内容创造	I
-------------------	---

[人物专访]

程立谈大规模SOA系统.....	1
------------------	---

[热点新闻]

GOOGLE已停止GEARS开发.....	8
AMAZON RDS: 作为云服务的MYSQL数据库.....	9
ASP.NET的未来：简化开发，HTML 5 及性能提升	11
案例分析：移植大型VB6 应用程序到.NET	13
RUBY ON RAILS登陆MICROSOFT AZURE	15
ECLIPSE插件TOP30，而你会喜欢哪个IDE？	16
开发者如何保鲜自己的技能？	17
CLOJURE近况：分布式、数学运算与构建的新动向	19
JBPM4 与BPMN2 将于 2010 年 1 月联姻	21
面向数据是面向服务的基石	23

[推荐文章]

OOPSLA辩论：被高估的云计算.....	26
构建自动化驶向何方？	29
使用JAVA构建高伸缩性组件.....	35

为网站和智能手机构建FLIGHTCASTER前台应用..... 43

SOA与服务识别..... 47

[新品推荐]

8.8.8.8——用于快速浏览的DNS服务器..... 51

TORQUEBOX：JVM上的RAILS企业级解决方案..... 51

IBM WEBSphere拥抱REST..... 51

GWT 2.0：新性能工具——SPEED TRACER..... 52

一种新的可视化处理数据的语言——VEDEA 52

ECMAScript 5 正式发布 52

INTELLIJ IDEA 9：JAVA EE 6、OSGI、FLEX及更多 53

SPRING 3.0 发布：基于JAVA5，添加了新的表达式语言和对REST的支持..... 53

MONO迈上新台阶：MONO 2.6、MONODEVELOP 2.2 和MOONLIGHT 2 发布..... 53

DOJO 1.4 发布：性能改进、稳定性提升..... 54

[封面植物]

桂滇桐..... 55

架构师

www.infoq.com/cn/architect

每月8号出版



程立谈大规模SOA系统

如何让整个组织充分理解这个复杂的动态系统？如何控制变更带来的未知影响，防范风险发生？如何驱动变更，使这个系统能够朝着期望的方向进化？这些问题挑战着SOA实践者的治理能力。在[QConBeijing 2009](#)期间，InfoQ中文站有幸采访了程立，探讨了相关话题。



程立，支付宝公司的首席架构师。他从 2004 年起参与支付宝与淘宝网的建设，2005 年正式成为支付宝人，随着支付宝的业务与技术的成长，从开发工程师、架构师到首席架构师一路走来，全身心投入并见证了支付宝业务与系统发展的完整过程。在加入支付宝之前，他在上海交通大学攻读计算机专业博士，同时作为 Sun 公司的兼职顾问，参加过电信、互联网、教育等各个行业 Java 企业系统的咨询与建设。

InfoQ：大家好，这里是首届 QCon Beijing 的现场，现在坐在我的旁边的是支付宝的首席架构师程立。先给大家介绍一下，支付宝架构发展到今天，经历哪些时期，都有哪些里程碑？

程立：我回忆一下，支付宝系统架构发展大概有这么几点。我本人大概是 2004 年下半年参与支付宝系统建设的。当时的目标，支付宝系统是面向整个互联网，而不是淘宝网内部的一个产品。那应该说是支付宝系统的一个起点，那当时非常的简单，就是一个应用程序，提供了我们所有的功能。功能也不多，有我们基本的支付功能，还有清算的功能，基本的会员管理功能，包括后台管理功能，这是支付宝的第一个里程碑，从无到有的过程。

第二个阶段是我正式加入支付宝，2005 年 2 月份，进来之后，我们做了一件事情，当时作为支付宝三期，这期项目实际上是一个真正意义上的支付宝，因为支付宝不仅仅是一个交易系统，支持各种各样的业务。我们希望它能够支持各种各样的交易流程，包括担保型的支付宝交易、即时到帐的交易等等都会在里面。但是当时支付宝主体的系统还是在一个应用程序里面——我们面向前端用户的系统，包括我们最后交易、支付的、会员管理的，都在这么一个系统里，这是第二阶段。

在这之后呢，我印象很深刻，就是在 2005 年上半年，这个系统里面有 20 个子工程，到 2005 年的下半年，不到半年时间翻了一倍。当然我们也尽可能把一些业务做成独立的系统，但是

发现支付宝大量的业务它的耦合度还是挺高的，所以我们不能把它做到一个系统里面。那这应该是支付宝的第二个阶段。就是我们开始在一个核心的主体应用里面，不断去堆积我们的业务功能，发展很快。

那在 2006 年初的时候，我们觉得这个不是长久之计，于是我们开始探索怎么样用 SOA 的思路去解决这样的问题，找一个切入点的话，我们找的是用 ESB，把一些可以异步处理的，耦合度不高的业务拆开，做成单独的业务服务。那这个阶段，大概持续了有大半年左右。这个时候我们发现，虽然我们把一些相对来说比较边缘的业务拆开，但是对我们核心业务，我们的交易、支付、处理、会员，他们之间耦合的非常紧，基于 ESB，基于消息，我们很难把他们拆开。去年的时候，我们在讨论这样一个问题，我们不能在持续一个应用中维护这么多业务，当时我们支付宝，开发规模已经上百人，十几个并行项目，每周一到两次发布，工作在这么一个 codebase 上面，肯定是不可接受的。那我们当时选择交易作为一个点，前面所说的原因，交易系统当时响应业务的变化已经很吃力了，一个应用程序里面，这么复杂的商业逻辑。我们既要响应前端的快速处理，又要保证交易逻辑不出任何错误，那么选择交易服务，针对这个服务我们也从技术上做一些重要的构思，我们建立了自己的服务框架。我们之前也做了很多调研，把一些我们不需要的功能拿掉，再把我们特别关注的一些功能，放上去，那这个项目大概持续了有四个月左右。

这是我们支付宝的第一组服务，它是可以支持我们核心业务的。在这个基础上，我们就去拿支付宝最重要的功能去改造去提升了——支付宝的帐务会计体系，还有支付宝的客户体系。关于这两个体系的建设，我们的思考，从几年前就开始了。那么支付宝最初上线的时候，业务的范围非常小，但是随着支付宝整个业务领域的拓展，原有的帐务会计体系不论是灵活性还有专业化的程度都不能满足需要。客户的模型，也不能满足很多业务的需要。首先是在这两个业务模型的基础上，做一个很大的提升。然后我们 SOA 整个一套业务的思想，把它分成真正可以支持业务发展的两套服务——会计帐务服务以及客户信息服务。这两个服务无论从业务和技术上都有挑战。业务上挑战是什么？就是我们业务服务模型是不是能够支撑支付宝未来数年里持续的发展。还有一个挑战就是技术上的挑战。帐务处理一旦拿出来后，那我们所有的业务，和我们的帐务处理之间全部都是分布的，很显然这个事务就是很关键的问题，还有它未来的伸缩性，当时我们的会员数可能接近 1 个亿。我们的交易笔数也是每年翻 3 番。那这个我们也做了考虑，解决了一些技术上的问题，同时也对交易服务框架做了一个提升。那么这样的服务全部做完了的话，大概大半年的时间，这个时间是比较慢的，慢的主要瓶颈是，我觉得前面我们讲的业务服务，怎么去识别服务，怎么去搭建一个可以支撑业务持续发展的模型，这块我们觉得难度非常大，需要有非常强的业务架构思维的人来做这样的事情，那么在支付宝的话，我们这样的人相对比较稀缺，所以说这样的事情我们只能一件

一件去做。可能这两个基础的服务体系搭建完了之后,支付宝的基础服务便形成了一个三角:帐务会计、会员,还有交易,在这上面,我们发现,有了他们这个基础,我们又产生了很多服务。

这时,支付宝应该是一个全面铺开的阶段。SOA 对像支付宝这样的企业,既要响应互联网快速变化,又要保证核心业务处理绝对安全可靠,可持续稳定服务。这是必须的。我们不能想象,支付宝现在数百位开发人员,在一个单体应用程序上,怎么去做。但现在,我们不但知道如何去做,而且做的更好,我们现在支付宝上海也有研发中心,都能够一起协作,这是 SOA 给我们带来的价值。

InfoQ: 您基本上把支付宝发展的历程完整地说了一下。您刚才也谈到了业务,我觉得在支付中业务需求的变化应当是非常快的,支付宝对于这个业务变化是做出了哪些改变,这个是不是支付宝引入 SOA 的一个主要的原因?

程立:之前提到了,引入 SOA,首先要让所有人达成一个共识。不光是技术人员,不光是业务人员,还有我们的高层,要达成一个共识。实际上当时支付宝所有人都看到了,存在一个挑战,就是前面提到的前端快速响应用户需求、后端持续稳定服务,这个快和稳的矛盾。我们觉得首先 SOA 会帮我们把快和稳切开,这样的话,才能够让前端真正快起来,SOA 在这方面确实给我们提供了价值。

但是 SOA 的话,让我们实现真正的业务敏捷的话,它有更深的意义,这是我们当时没有看到的,那就是 SOA 是真正能够让我们的系统和我们的业务架构对齐。因为它统一了业务和技术,通过服务的概念统一起来了。这样我就能让业务不但符合当前的需要,而且能够符合长远发展的需要,能够符合整个架构,甚至和公司的整个使命、甚至是愿景对齐。这些方面,其实现在我们也在探索。

InfoQ: 现在支付宝的这个架构是非常优秀的,您觉得对于一个企业来说,一个优秀的架构意味着什么呢?

程立:架构的话,我个人觉得,可以从很多层面去理解。我刚到支付宝的话,我看到的架构很狭隘,我就看到了,比如说我写一个框架或者一个公共基础类的,这就是架构,这是我的理解。后来我发现这个架构仅仅局限于技术的话,真的是太小了,我们会把业务考虑进去。我们希望整个架构,既包含业务又包含技术。双方的基础是统一的,只不过大家面向的客户是不一样的。大家可能采用的技术手段是不一样的。但是从概念层上里看是统一的。

再后来,我发现,光有业务、技术架构,以用户的需求去驱动,也不够。当我们一次、两次、三次满足用户需求之后,突然发现用户的某个需求可能跟我们原来的设计模型和基础设施是

不匹配的，这样如果要快速响应，我们很难去做到。所以说我们这时候，认识到架构它不是直接面对业务的需求，还有一个输入是企业的战略，架构要与企业的战略对齐，这样的话，才能支持企业长期的发展。战略会规划企业未来几年的方向，可能的规模，重点的任务。如果架构能够跟它对齐，那架构将会有更强的生命力。

InfoQ：那，我的理解就是说一个企业要想看这个架构适不适合它，就是看它是否与企业的战略对齐呢？

程立：是的。因为说战略，可能有些虚。如果是实了，就是说架构不但要支持当前的需求，对业务需求要有一个持续稳定的支持能力。而且这个持续稳定，我们很难去估计未来的需求，那这个时候就只能靠战略，如果是和战略对齐的，我们会发现战略里面考虑的未来重点方向，都在架构中有落实，无论是业务上，还是技术上，这样我们就更加信心，架构是能够支持企业长期发展的。

InfoQ：下面问一些技术方面的问题。支付宝的系统在 SOA 化以后，觉得面对的一个很大的挑战应该是分布式事务的问题，因为事务对于一个交易平台来说应该是非常重要的，那么在这个方面有什么经验给大家讲一下？

程立：我其实第一次遇到事务问题，是在建设交易系统的时候，其实交易一旦拿出来，我们就已经遇到事务的问题了。交易系统作为一个单独的系统，帐务系统在另外一个系统里面，那怎么保证他们之间的事务一致性。当时，由于交易系统本身业务的特殊性，我们当时采用的就是重置的方式，在交易的过程中，可能请求某些帐务，这时帐务处理可能会失败，这个时候我们保证通过恰当安排交易系统的处理顺序，如果是由于业务原因造成帐务处理失败，那交易系统是能够正常地把业务回滚的。那如果由于是系统的原因，那交易系统会处于一个等待系统重置处理的状态，这个系统会安全地通过重置的方式，让最后的交易与帐务处理达成一致。交易处理是幂等的，帐务处理也是幂等的，这是当时的一个处理方案。

这个方案比较简单，但是能满足帐户和交易之间是业务的需要。后来提到支付宝把整个帐户的体系进行 SOA 化的时候，那帐户系统面对的不只是交易系统，和所有系统都有这种分布式处理的交易。我们需要提升这个框架，满足多系统，多层次的服务之间做分布式事务的需要，所以当时我们想了很多的方案，我们也是看了一些标准，那个时候正好是 2007 年的上半年左右，正好 WS Transaction 作为标准推出来了，我们看了看 WS-AtomicTransaction 等，而且我们实际做了相应测试的环境，去做一个概念验证，最后发现实在是成本太高了。完成一个事务，20 多条的消息的交互，其中只有 1 条是业务消息，其他都是系统之间的协议消息。那对于支付宝互联网应用，必须在很短时间做出响应的交互，我们无法支付这样的成本，那我们只能去考虑一些山寨的解决方案。

当时对我们最有启发的一个是 Ebay，关于系统设计，他又一些最佳实践。其中有一条，就是关于一致性的。我们在满足业务需求的情况下，允许有不一致的情况出现。这对业务是允许的，这是一个很重要的思路，虽然他没告诉你怎么去做。另外呢，亚马逊关于这些方面也有一些很好的阐述，比如说亚马逊有一位架构师，他关于这方面写了一篇文章，当然他也是面向他的场景，比如他们的分布式存储如何保持一致性，这些对我们也有一些启发，那关于事务我们需要的 ACID 到底哪些是必须保证的，哪些是可以放宽的，那 atomic 原子性这个是必须要保证的，还有 isolation，对事物的处理过程中，对事物所涉及到的资源中，我们一定要控制起来，否则如果你再用这个资源做事务中的某些事情的时候，就可能被人用掉了。如果要是一笔钱的话，用户可能把钱用到了其他的场景，没法在做业务上的处理，那这样的话，就把一些没有隔离性的补偿方案排除掉了。那还 Consistency 有一致性我们是必须需要的，但我们要的是最终一致性，我不一定在某一个瞬间完全一致。那我们就认为一致性可以放宽。那最后决定，在 ACID 四个属性中，我们放宽一致性的需求，同时我们也是在考虑到未来系统可伸缩的需要，我们一定不能用分布式的事务，标准的基于 XA 的分布式事务。

在这两个基础上，我们就设计了一套支付宝分布式事务方案，在数据库操作这个层次我们建立 transaction，我们的帐务处理是一种 TCC 的模式，任何一个帐务处理，你可以 Try 它，最后你可以 Comfirm 它，也可以 Cancel 它。那在这个过程中，有 TCC 的基础支持的话，结合中央事务协调器，我们就可以做分布式服务，而且可以多层次的，任何一个服务，都可以做在事务里面。这个也会有成本，像设计一个支持 TCC 操作的业务服务。当然这方面，我们也探索出一些模式，可以做得比较好。

InfoQ：现在支付宝每天的交易量是非常惊人的，在这么大交易量的情况上，在网站的伸缩性设计上，有哪些考量？

程立：我们觉得做 SOA 很大的好处就是他把很多复杂操作的伸缩性，简化成单个服务的伸缩性。比如对支付宝来说，我们有这么多的业务，对伸缩性最有挑战的，还是帐务处理和会员业务。比如帐务处理，那帐户处理就是相对来说比较单一的一个应用，那对这样的应用，我们有一些比较好的伸缩性的方案，那比如说应用处理，它对 SOA 来说是比较简单的，因为每个服务都是无状态的，是自治的，是可以很方便的通过水平扩展的方式去处理。其次是数据，那数据的话，取决于你的使用场景，如果你的场景支持数据很好的水平分割，那它的伸缩性也不会有什么问题的，那从目前来看，因为我们把服务拆开了，对于每一种服务我们都可以选择最适合它的一种方案。比如会员数据，我们可以很方便的根据会员把他们分开。

我们帐务处理其实也是一样的模式。那交易可能会复杂一些，但它也有它相应的模式，但我们并没有立刻去做这样的事情，因为目前来看，底层的基础还能够做相应的支持，支持我们

现在事务的处理，但是我们相应的方案已经准备好了。如果当业务量达到去做分割的时候，我们就会去做。毕竟分拆之后，对运维，对成本都会有一些相应的要求。我们可以在适当的时候支付这样的成本。

InfoQ：还有一个问题，其实也是大家都想问的问题。如果一个公司，它想在系统里面引入 SOA 的话，您作为一个在这方面的实施有非常丰富经验的人，能给他们提一些建议吗？

程立：实施方面，关于 SOA 怎么实施，其实有很多指导。那我们支付宝实施 SOA 的话，前面也说到，就是大概先去学习，对这个概念的认识，基本知识的掌握，获得所有人对走上这条路的共识，因为只有这个共识，大家才愿意投入这样的资源。因为 SOA 既有风险的，又要投入很多的资源，而且它不一定会马上看到效果，所以这肯定是第一步。

完成这一步之后呢，需要选择一个切入点，每家企业都有自己最痛/痒的地方，第一个投入这个项目的时候，要找准切入点，在找的过程中，要选择一个既不要太难，又不要太简单，然后做之后能产生效果的项目，因为太难的话，可能 SOA 这个项目就会做失败，那对 SOA 迁移就会产生非常大的打击。如果太简单的话，又不能起到锻炼技术、锻炼队伍、积累经验的目的。最后它要看到结果，如果看不到结果的话，那后续的资源投入都会遇到问题。然后下一步，我感觉支付宝是比较大胆的，我们立刻拿我们最核心的业务去动刀了。这一块我觉得要动，早动会比晚动好。我们在 2007 年去动这块业务，当时我们敢动。现在如果动的话，我相信会有更多的挑战、更大的难度。那整个基础打好之后，这个时候我想，包括你的基础设施，包括你的主要的 SOA 团队的建设，包括前面讲到基础，不光是技术基础，也是业务基础，包括整个一套的方法论，无论是已经正式成形的，还是大家口头认识上的，都应当有一定基础了。

这个时候，可以让 SOA 交付它的价值了。但是我们支付宝，有一方面我们觉得做得还不够，还需要改进。就是治理引入太晚了，对于治理的问题，我们并没有非常早的意识到，一路的高歌猛进，但是后来发现，SOA 会引入一些问题。使系统在某些方面变得更加复杂，更加难以控制等等这样一些问题，包括怎么让 SOA 真正交付价值，这些都需要治理来支持。

今天我才看到，有一篇应用 SOA 治理的报道，非常有意思，它里面就提到，如果你的系统服务个数达到 50 个以上，治理的话，不但要有，而且是正式的，得到了充分资源的保证的流程，实际上治理的话，不能说没有，一开始我们是一种人治的方式，通过这么多 SOA 系统建设的话，我们有一个很好团队，通过他们的口头的方式治理，但是随着整个服务体系全面铺开，靠人是很不靠谱的。我们希望如果有其他的企业想要走在 SOA 这条路的话，希望能够把治理尽量提前。

InfoQ：非常感谢程立接受我们的采访，谢谢大家。

程立：谢谢。

观看完整视频：

<http://infoq.com/cn/interviews/soa-chengli>

相关内容：

- [调查表明SOA势头正劲](#)
- [遗留系统要想加入SOA需要服务么？](#)
- [将SOA经验应用于Web 2.0 实现](#)
- [SOA=集成？](#)
- [服务监管人](#)

Google已停止Gears开发

作者 [Abel Avram](#) 译者 [张龙](#)

Google 对 Gears 的继续开发已经失去了兴趣，转而投向了 HTML 5。

近日有人发现在Mac OS Chromium的一个bug（[Issue 13161](#)）描述上写到“不支持Gears”，鉴于此Google Chrome for Mac的技术经理Mike Pinkerton说到“我们不打算在Mac上支持Gears了，而是全面拥抱HTML 5”。该bug的状态是“不再修复”。因此，即将到来的Chrome for Mac将不会使用Gears实现某些操作，如离线浏览或是拖拽。

Pinkerton 提到的原因就是 HTML 5，它将提供离线浏览和当前 Gears 所具有的其他特性，但问题在于 HTML 5 距离标准化还遥遥无期，甚至连 IE 都还不支持。Google 的这个举动表明了其想借助于 HTML 5 这个机会取胜的信心，微软也打算在 IE 9 中加入 HTML 5 的某些特性。

与此同时，[L.A. Times报道说](#)Google一位发言人澄清了Google对Gears的立场：

“我们会继续对 Gears 提供支持以便使用其的站点不会因此受到影响。但我们希望开发者采取 HTML 5 实现这些特性，因为这是基于标准的方法，可以跨越多个浏览器。

综上所述，Google 会继续修复现有 Gears 代码中的 bug，但却不会再添加任何新特性了。开发者应该认识到：迁移到 HTML 5 才是明智之举。

原文链接：<http://infoq.com/cn/news/2009/12/Google-Stops-Gears>..

相关内容：

- [IE中使用Google Chrome Frame运行HTML 5](#)
- [SproutCore：一个HTML 5 应用框架](#)
- [各方未就HTML 5 Video Codec达成一致](#)

Amazon RDS: 作为云服务的MySQL数据库

作者 [Carlos Armas](#) 译者 [晁晓娟](#)

Amazon最近给他们的[Amazon Web Services](#) (AWS) 平台增加了一个新的[MySQL](#) 数据库,叫做 Amazon [关系数据库服务](#)(RDS), 它能和传统的MySQL系统一样工作。在RDS之前, 客户在AWS的数据库服务上有几种选择:

- 运行在[Amazon Machine Image](#) (AMI) 的客户自提供数据库服务
- Amazon Web服务所拥有的[SimpleDB service](#)

SimpleDB 是一个简单的数据存储, 它缺乏一个完全成熟的关系数据库管理系统(RDBMS) 所拥有的完善的功能, 但是提供了一种可伸缩的键值存储。客户自提供数据库服务和传统的数据中心环境差不多, 由客户自己的员工负责管理数据库应用程序, 包括配置, 性能调优, 容量管理, 版本升级, 打补丁和数据备份等。你可以使用和传统 MySQL 数据库连接的交互工具来以同样的方式控制它。

Amazon RDS 使得客户员工减少了很多MySQL的运维任务, 有了它, 数据库计算资源的可扩展性和性能监测都无需人为的干涉。而数据库软件通常都由服务提供商来打补丁和备份, 并且是由客户定义的保留时间段来做。可扩展性来自AWS 所谓的“实例类”, 总共有五个。你可以从一个普通的虚拟CPU 内核以及 1.7G的内存 (被叫做“小的数据库实例”) 逐步增大到 “超大型的数据库实例”, 也就是 68G内存和 8 个虚拟CPU内核, 而备份存储被活动状态的数据库数据 100%占满后, 额外的存储空间是要收费的。而且数据存在另一个不同的[可用区](#)而不是该实例所在的地方。这个和传统数据安全模型的[异地数据保护](#)的概念是类似的。

这个服务得益于灵活性, AWS定义了一个[每周 4 小时维护窗口](#)。这个维护窗口可以被用来为应用软件打补丁和数据备份。客户[不能选择退出打补丁的过程](#)。但是他们可以指定维护窗口在一周内何时发生。在维护窗口中, 数据库实例会在特定时间段内被离线。Amazon 声明“只有很少情况下, 打补丁需要超过你的维护窗口的部分时间, 即使发生也只是为了安全或

者持久性相关的补丁。”

这意味着客户必须预期和计划这样一个每周发生的实例离线事件。即使服务商表示不太可能用完四个小时的时间，但客户也会预期最差的情况，每周要有四个小时的实例离线时间。对于能够接受一个相对短时间的数据库实例不可用的客户，按计划关闭时间而只有最小可能的影响的方案也许能够被接受。但有一些客户没有这样选择的自由。他们必须保证服务 24x7 可用，即使在每周的维护窗口运行的时候也一样。在传统的数据库部署中[数据库复制](#)技术常常被用来达到高可用性。复制技术能不能也用到RDS中，从而让客户能够[为不同的数据库实例指定不同的维护时机](#)呢？比如，如下几种情况可能吗？

- 2 个或更多的实例运行在 master-slave 模式？
- 2 个实例运行在 master-master 模式？
- 2 个或更多的实例运行在 cluster 模式？

现在还没有很明确的答案。在[RDS 服务细节页面](#)的“即将推出的新特性”一节中，Amazon 预期数据复制可用性的选择将会是：

“提供高可用性 --对于想要超出 Amazon RDS 自动备份之外灵活性的那些开发者和商业人士，将不需要对此额外付费。有了高可用性的支持，他们能够很容易并且在成本有效的情况下在多个可用区之间同步复制数据库实例，来防止出现单一存储导致的失败。

看起来这将会通过多个可用区为代价来解决可用性问题。而解决可用性的传统技术如 master-slave 和 master-master 模型在这一点上并不能起到作用。

原文链接：<http://infoq.com/cn/news/2009/12/amazon-rds-cloud-db-cn>

相关内容：

- [亚马逊协助.NET开发人员使用其云计算平台](#)
- [亚马逊开始提供MySQL服务](#)
- [Flex：Engine Yard的全新云服务](#)
- [云计算的虚拟研讨会](#)
- [InfoQ案例研究：纳斯达克市场回放](#)

ASP.NET的未来 :简化开发 ,HTML 5 及性能提升

作者 [赵劼](#)

在上月举办的 PDC 09 大会中,微软 ASP.NET 团队的 Jonathan Carter 和 Scott Hunter 演示了为 ASP.NET 4 以后版本设计的一些功能,其主要方向是简化应用程序的开发,支持 Web 标准,以及提高性能提升。

在简化应用程序开发方面,ASP.NET 团队正在考虑以下几个功能:

- 可用于 ASP.NET MVC 和 WebForms 的 Action Record 模式支持,基于 Entity Framework,方便快速建模,快速开发。
- 更易于使用的 Route 规则:能结合各种信息(如硬盘上的文件路径)自动判断路由目标及相关参数。
- 可扩展的,基于常见任务/场景的辅助方法,例如:
 - 图片处理,如缩放,水印等常用操作。
 - OpenID 支持,这样开发人员可以轻松将 ASP.NET 认证与 OpenID 集成。
 - 后台计划任务,如“每 10 分钟”或“每天凌晨 2 点”执行某个任务。
 - Email 发送,以及使用 Email 进行验证的注册流程。
 - 真实的文件上传进度提示,目前实现这个功能需要使用某些危险的技巧,而今后 ASP.NET 可能会释放更多接口来进行支持。

HTML 5 带来了许多新特性,例如新的 HTML 标记,原生的视频和音频支持,以及拖放操作

等等。未来的 ASP.NET 首先会支持 HTML 5 中更符合语义的标记。如在 ASP.NET 2.0 中，`<asp:Menu />`控件会生成复杂的 table 标记，在 ASP.NET 4 中则会变成符合目前语义的 ul/li 嵌套，而在未来的 ASP.NET 中，便可能会生成`<menu />`标记。此外，HTML 5 的 Web Storage 功能允许将数据储存在浏览器上，未来的 Microsoft AJAX 库中将会提供一个可选的 `IntermediateDataContext` 用于替换目前的 `AdoNetDataContext`，后者将数据通过 WCF 接口存放在服务器端，而前者则将数据保存在本地。

在性能提高方面，ASP.NET 团队会在在微软的分布式缓存 Velocity 发布之后，为 ASP.NET 提供相应的各类 provider。这样 ASP.NET 便可以将数据缓存，会话状态等各种信息存放在进程外的分布式缓存中，以此得到更好的性能和健壮性。这些 provider 实现可以与 ASP.NET 现有的扩展方式良好集成，对开发人员的使用保持透明。

由于 Web 应用程序的显示效果越来越丰富，网页前端性能优化的重要性也随之提高。未来的 ASP.NET 将会内置 CSS 或 JavaScript 文件的压缩及 合并，并对 CSS Sprites 等复杂优化方式提供支持。CSS Sprite 的优化原理是将页面上大量的小图片合并成一个文件，然后使用 CSS 定位机制来显示其中的一部分，这么做的好处是大大减少了浏览器与服务器端的 通信次数，往往可以使页面加载速度有明显提高。ASP.NET 在未来可以根据开发人员的需求，自动将一组图片进行合并，并通过一些接口将单独某幅图片的信息（如位置，尺寸）暴露出来，甚至直接在页面上生成包含特定属性的 HTML 标签。

你可以在[PDC 2009 的网站](#)上浏览或下载[本次演讲的完整录像及幻灯片](#)等资源。

原文链接：<http://www.infoq.com/cn/news/2009/12/aspnet-futures>

相关内容：

- [PDC 09：ASP.NET 4 运行时的改进](#)
- [微软发布Microsoft Ajax脚本库和脚本缩小器](#)
- [ASP.NET MVC 2 预览版更新](#)
- [ASP.NET vs. PHP，哪个更快？](#)
- [FireAtlas：ASP.NET AJAX查看器](#)

案例分析：移植大型VB6 应用程序到.NET

作者 [Abel Avram](#) 译者 [王波](#)

一个 IT 服务提供商花了 9 个月时间把一个由 950000 行 VB6 代码组成的 ERP 应用程序移植到.NET。

处理旧版 VB6 应用程序有以下 5 个选择：

- **不作修改**——如果应用程序甚少更新或更改的话
- **用新程序取代**——如果有类似的商业或开源应用程序或服务的话
- **移植到.NET**——使用升级工具把所有 VB6 代码转换成相应的.NET 代码
- **使用.NET扩展**——使用.NET添加新功能。可以给VB6 应用程序添加.NET窗体和控件，详情请查看：[Interop Form Toolkit 2.0 \(PDF\)](#)帮助文件
- **重构**——用.NET 重写所有代码。在旧版 VB6 应用程序的代码不健全或者有需要重写应用程序的时候，即应用程序需要大量修改以满足新的需求或源代码不可用的时候，这是一种可行的方法

若移植是必由之路，以下则有三个备选方案：

- 微软Visual Basic升级向导。VS 2008 包含该向导，它可以很好地处理中小型的项目。对于大型企业级应用程序，[微软则有两个推荐解决方案](#)
- [ArtinSoft公司的Visual Basic升级伴侣](#)
- [Code Architects公司出品的Visual Basic移植搭档](#)

微软最近公布了一个把 950000 行VB6 代码移植为.NET的[成功案例](#)。该过程由澳洲的IT公司[SiS](#)主持使用Code Architect的VB移植搭档耗时 9 个月完成。该项目是一个十年前构建的ERP项目，由 33 个应用程序组成。这个应用的最佳解决方案是采取移植的方式 而不是定制ERP应

用或重写整个项目。不建议定制商业的ERP项目，因为它需要花费 3 至 5 百万欧元，且需耗时 2 年并没有任何供应商保证实现所有原始功能。

sis 从中选取 25000 行代码并尝试用多种工具进行移植。VB 移植搭档是最好的工具，代码块的移植、编译和运行总共只需 25 小时。他们还声称 VB 移植搭档不需要过多的手动干预也不会影响用户界面的外观。

整个 ERP 应用程序由 3 个开发人员花了 9 个月的时间来完成，“3650 个工时用于移植代码，3400 个工时用于检查和重构代码，1300 个工时进行测试”。检查代码是必要的环节，因为将来代码有可能会需要进一步完善，而初次参与的开发人员未必能再次参与其中。该项目总共耗费 75000 欧元，远远低于重新订做 ERP。过程按 ([a href="#">方式进行：完成一个模块之后，它就会与余下的 VB 应用程序集成直到整个应用程序移植到.NET 为止。

对 VB6 投入大量资源的公司现在也可以通过移植到.NET 来利用他们的旧资产。微软及其合作伙伴似乎已为他们准备好了所需的工具。

原文链接：<http://www.infoq.com/cn/news/2009/12/Migrating-VB6-to-.NET>

相关内容：

- [VB 10 中集合与数组的初始值设定项](#)
- [VB.NET路在何方？](#)
- [VB和C#的自动实现属性](#)
- [在.NET语言中封装存储过程](#)
- [.NET 4 中的模式匹配](#)

Ruby on Rails登陆Microsoft Azure

作者 [Abel Avram](#) 译者 [张龙](#)

微软已向众多的非微软技术开放了 Windows Azure，旨在让这些公司和开发者将其应用部署到 Azure cloud 上而不是其竞争者那里。Ruby on Rails 就是其中之一。

微软在PDC 2009 上演示了如何在Azure上运行MySQL/PHP/Apache，同时宣布[WordPress登录 Azure](#)。此外，[Java应用也可以部署在Azure cloud上的Tomcat](#)。实际上，微软已经向 .NET、PHP、Python和Ruby语言开放了Azure，并为那些没有使用Visual Studio的开发者以及使用[PHP](#)、[Java](#)和[Ruby SDK](#)的开发者提供了[Eclipse插件](#)以加速开发。

Microsoft Azure架构师Simon Davies[说到](#)Azure SDK最新的改进（通过HTTP、HTTPS和TCP与 worker roles进行通信的功能）为很多新应用场景提供了可能，包括运行“各种应用和技术，比如MySQL、Mediawiki、Memcached和 Tomcat等”。他使用Ruby on Rails构建了一个[示例性的Web站点](#)并部署到Azure上。虽然该示例非常简单，仅仅使用了一个简单的[SQLite数据库](#)，但足以证明Ruby on Rails兼容于Azure。

既然很多非微软技术可以部署到 Azure 上，我们倒是想看看这个世界是否会走进 Microsoft cloud，联姻开源与私有代码，还是选择跟随 Amazon 和 Google 的脚步。

原文链接：<http://www.infoq.com/cn/news/2009/12/Ruby-on-Rails-Azure>

相关内容：

- [Brad Abrams终于完成了.NET RIA Services的开发系列文章](#)
- [微软发布工具帮助用户从SQL Server快速迁移到SQL Azure](#)
- [SQL Azure万事俱备，用户需要在 12 月前迁移到新的CTP上](#)
- [即将到来的.NET访问控制服务](#)
- [PHP与微软云计算](#)

Eclipse插件Top30，而你会喜欢哪个IDE？

作者 [张凯峰](#)

近日，AjaxLine统计发布了 30 个最受欢迎的Eclipse插件，详文请见[这里](#)。众所周知，Eclipse是最初作为Java IDE发布的一个开放平台，因其具备完善的插件扩展机制，并逐渐加入对于C++和其他语言平台的支持，而受广大开发群体的欢迎，一度成为IDE市场事实上的标准。

很软件开发厂商，包括IBM和Oracle这样软件巨鳄在内，也都基于Eclipse平台推出各自的能贯穿软件生命周期的产品平台，包括有IBM的[Rational产品序列](#)，Oracle的[JDeveloper](#)，还有ThoughtWorks的自动化测试平台[Twist](#)。

这次遴选出的 30 款最受开发社区欢迎的插件，范围涉及 Python、LAMP、Web、Emac、SQL、Log、UML 和 IDE 增强等诸多方面，充分体现 Eclipse 插件的繁荣事实和 Eclipse 做为 IDE 在开发者心目中的地位，钟爱 Eclipse 的开发者也必定能从中选出得心应手的兵器。

值得注意的是[Sun（现在应该是Oracle）](#)公司推出的开源IDE [NetBeans](#)，也以其开放的姿态（[对动态语言Ruby、Python和Groovy的支持](#)）和优雅的开发体验，[开始重新俘获一部分开发者的芳心](#)。加上在开发社区中具备良好口碑的IntelliJ IDEA前段时间开始[提供免费的社区版本](#)，IDE市场重现良性竞争的形势指日可待，而最终受益的无疑会是我们开发者自己。

原文链接：<http://www.infoq.com/cn/news/2009/12/plugins-top30>

相关内容：

- [IBM developerWorks十周年：精彩内容回顾](#)
- [OSGi 4.2 发布了](#)
- [Eclipse社区调查结果发布](#)
- [伦敦Eclipse DemoCamp活动上的新技术展示](#)
- [Google发布Eclipse插件](#)

开发者如何保鲜自己的技能？

作者 [张凯峰](#)

开发者如何保护好自己在技能上所作的投资，让自己的技能保质保鲜，是一直以来头疼但又必须面临的问题。William Jordan为大家总结出了[六点](#)：

1. **阅读。**阅读，阅读，再多读些。找到你想要了解的编程主题相关的图书或者网站，就开始阅读。亲手试过每一个出现的范例并理解它们。
2. **厂商认证。**通过一些认证可以让自己的简历强大起来，比如微软就会为各个层次的角色提供认证考试。你可以通过这样的认证来向你未来的雇主展示自己的领域技能，并证明能胜任新的工作。课程可能会比较贵，如果有时间可以在网络上选择免费的资源。
3. **听课或参加研讨会。**许多厂商也会提供在线或离线的学习研讨会。也有大量关于编程主题的网络广播。你可以花时间去学习这些，掌握新的工具。
4. **授课。**一些本地的大学会提供成人继续教育课程，并寻找行业专家来授课。准备这样的课程并向别人讲授，对自己是一个很大的提升，而且可能因此获得报酬。
5. **从实践中学习。**想着去做一个小型项目，如果想不出来，就看看周围有没有慈善团体或小公司需要这样的工作。此举并不是为了赚钱，而是通过实现技术来增强你自己的技能。你也可以到[Getafreelancer](#)或者 [Scriptlance](#)注册成自由开发者来完成一些项目。同样你可能因此而获得收益。
6. **撰写文章并回答问题。**有很多这样的网站让你来撰写文章或者回答其他开发者的问题，比如[CodeProject](#)和[Stackoverflow](#)。在你为撰写文章或者回答问题而研究并编写代码时，你同时也在学习。此外你也在赢取良好的名声。

各位 InfoQ 的读者，你是怎么保鲜自己的技能的呢？欢迎大家到此新闻页面留言。

原文链接：<http://www.infoq.com/cn/news/2009/12/keep-skill-refresh>

相关内容：

- [提供和接收有效的反馈](#)
- [学徒模式](#)
- [质疑服务型领导](#)
- [Castle项目创始人加入微软](#)
- [Hyrid Camry首席工程师过劳死](#)

Clojure近况：分布式、数学运算与构建的新动向

作者 [Werner Schuster](#) 译者 [丁雪丰](#)

[FlightCaster](#)使用Clojure分析数据，使用Hadoop处理分布式相关的工作。其背后的公司现在将[Crane](#)开源出来，这个工具在FlightCaster处理了很多分布式相关的机制。

Crane目前适用于EC2；它简化了很多工作——启动EC2实例，向实例推送Clojure代码，使之能够通过ssh访问。通过Crane，能够在主控端控制远程实例，例如，可以在实例上远程执行Clojure代码（取自Readme）：

```
(eval! socket (execute (workflow some-cascading-workflow)))
```

socket中持有远程实例的连接，该实例会在本地执行第二个参数。

想要获得更多信息，可以访问[Bradford Croos关于Crane的文章](#)。

FlightCaster最近还[向Incanter贡献了统计学习代码](#)。[Incanter](#)是：

“[...] JVM 上的一个基于 Clojure 的，与 R 类似的统计计算与制图平台。

Incanter把数学运算的Java库和[Processing](#)的可视化库打包在一起。该工具既能以交互的方式使用（用Incanter发行包里的bin/clj可执行程序），也能作为Clojure程序里的一个库。

正如[FlightCaster贡献代码的声明](#)里提到的那样，目前他们正在着手让Incanter能与Hadoop协同工作。

最后，构建与依赖管理工具[Leiningen](#)发布了 1.0 版本。[使用以下安装指令来获得Leiningen](#)：

1. 下载脚本：<http://github.com/technomancy/leiningen/raw/stable/bin/lein>
2. 将它放在 path 中，并赋予可执行权限。
3. 运行：`lein self-install`

安装完毕后，`lein new PROJECT_NAME`会创建一个带有必要文件的项目框架，包括Leiningen的配置文件`project.clj`。（请注意：如果该命令出错，[系统中可能没有正确的Clojure.jar](#)）。

Leiningen不仅能够帮助进行构建，还可以替项目管理依赖，它和[Clojars](#)协作的很好，Clojars是一个构建于Maven之上的Clojure库管理工具。

[用Leiningen和Clojars.org来构建Incanter应用程序](#)一文中提供了一个使用Leiningen的例子，构建了一个将Incanter作为依赖的应用程序。

原文链接：<http://infoq.com/cn/news/2009/12/clojure-crane-incanter-leiningen>

相关内容：

- [专访开源项目Amoeba架构师陈思儒](#)
- [分布式系统中的一致性和可用性](#)
- [CRISPY，一个新的远程框架](#)
- [分布式计算的谬误在今天的意义有多大？](#)
- [Google的二进制编码格式：Protocol Buffers](#)

jBPM4 与BPMN2 将于 2010 年 1 月联姻

作者 [马国耀](#)

继 11 月 5 日 jBPM v4.2 版本发布之后 ,jBPM 又有了新的动向 ,Joram Barrez 在他最新的[博文](#)中称 :

“自从今年夏天开始 ,我们就着手在 PVM 平台上实现对原生 BPMN 的支持.....在本博文中大家将能窥见 BPMN2 当前的执行进展.....它将整合在 jBPM4.3 中发布 (原定日期是 2010 年 1 月 1 日)。

这对流程开发社区来说无疑又是一大喜事 , 因为广大业务流程开发人员不再需要将由 BPMN2 建模的流程图转换成 jPDL 等 , 而直接可以在 jBPM 平台上运行。

为什么说 jBPM 与 BPMN2 的联姻是一大喜事 ? 这需要先了解 BPMN 和 jBPM 天生支持的语言 jPDL 之间的差异。

- BPMN 即业务流程建模符号 (Business Process Modeling Notion) , 它是用一种类似于流程图的图表形式来描述业务流程的一种方法 , 目前由对象管理组织 (Object Management Group OMG) 进行维护和管理。jPDL (JBoss jBPM Process Definition Language) 是构建于 jBPM 框架上的流程语言之一 , 它并非公开的标准但却与 jBPM 有这天然的亲缘关系。BPMN2 是其升级版 , [新增了很多特性](#) , 如提高了符号的准确性、定义了行业标准的交换格式、提供了一系列扩展点以及对编排的支持等。
- jPDL 是 jBPM 的原生执行语言 , 它提供了任务 (tasks) 、等待状态 (wait states) 、计时器 (timers) 、自动处理 (automated actions) ...等术语 , 并通过图型化的方式直观地描述流程。

[JBoss 社区](#)中给出了 BPMN 与 jPDL 之间的差别简化如下 :

- BPMN 是一个公开的标准 , 而 jPDL 专属于 jBPM
- BPMN 更加侧重于流程的建模 , 是类似于 UML 的建模语言 , 而 jPDL 侧重于流程的执行 ,

它是业务流程执行语言

- BPMN 与实现无关，而从 Java 开发者的角度来看，JPDL 则更简单。

然而，jPDL 与 BPMN 之间也有很多共同之处，比如它们都使用直观的建模视图，用 XMI 的形式描述业务流程，解决的是业务流程方面的问题。正如 Joram 在博文中提到的：

“.....我们在 jBPM 中投资 BPMN 是很自然的事情，熟悉 JPDL 的人学习 BPMN 通常毫不费力，因为很多结构和概念都是系统相同或相通的。事实上从高层次看来，BPMN2 和 JPDL 在概念上解决的是相同的问题.....”

在博文中Joram给了一个简单的示例，在[Signavio](#)流程编辑器上建模，并直接部署在jBPM之中，然后用纯Java代码测试该流程。从这个例子中可以看到jBPM的新版本对BPMN2 的平滑支持。由此可以看出jBPM近期动作频频，如PVM、支持Web的流程IDE、BPMN以及与Spring的整合，InfoQ对jBPM4 发布以来几次动作总结如下：

- TheServerSide[总结](#)了jBPM的第一版发布（2009-07-11），称它引入了新的PVM概念，使之支持多种业务流程语言：
 - 支持 Eclipse 中基于 BPMN 的图形化流程设计工具
 - 将基于 Command 的服务做为主要客户端 API
 - 活动实现 API 的解耦
 - 便捷增加客户化活动
 - 数据库和流程语言的分离
- V4.1（2009-08-31）中增加了 Signavio 的 Web 流程编辑器
- V4.2（2009-11-05）中增加了对 Process ClassLoader 的支持和服务实例的版本控制

那么，V4.3 除了在 V4.2 的基础上增加了对 BPMN2 的支持，还将为我们带来哪些新的特点呢？我们翘首期待。

原文链接：<http://infoq.com/cn/news/2009/12/jbpm-bpmn2>

相关内容：

- [到底谁需要BPEL？](#)
- [BPM不是软件工程](#)

面向数据是面向服务的基石

作者 [Boris Lublinsky](#) 译者 [黄璜](#)

现今大多数的SOA文献与实现都专注于定义业务对齐的服务，而很少讨论企业数据在SOA环境中扮演的角色与产生的影响。[David Linthicum](#)如是说：

“那些投向 SOA 的人们看似对于 SOA 中数据的使用颇为疑惑。虽然其中大部分将数据视为...其实，数据，在懂行的人看来，是 SOA 作为一个项目能够成功的战略部分，或者是说是整体性的架构战略。当所有的焦点都在 SOA 的“S”即服务上的时候，问题也就来了。那些负责构建架构与系统的人，往往只注意了将服务作为交付功能行为的主张，而忽视了管理底层数据的需要。在许多场合中，数据质量与一致性的问题很快就显现出来，而由于需要在底层数据的更改后直接修改服务，SOA 的灵活性也大打折扣。

David关于数据集成作为SOA基础的重要性这一观点，在Ash Parikh最近的[文章](#)里得到了进一步的阐述。

越来越明显的是，任何面向服务的基础设施努力都必须从认真的研究数据集成开始...“数据集成”对于任何想要以速度与灵活性架构基础设施的人来说，都必须摆在首位。所以，如果你想将你的基础设施面向服务化，我建议你首先面向数据化。检查以下这五点以保证其根基满足相应的能力：

- 方便地访问所有相关的数据，包括新的和经常更新的数据源。
- 批处理或实时的处理数据，包括处理大容量的大数据集。
- 预先辨认与解决数据不准确与不一致。
- 对数据应用复杂的数据变换。
- 在需要数据的时候能精确地交付数据，要作为一项基于标准的服务。

在他[接下来的文章](#)里，Ash讨论了如何将面向服务的基础数据化的实用手段。他描述了几种

规范的推荐纲要，为企业实现数据集成提供了整体的解决方案。

- 由一个支持“标准化”访问所有企业数据源而无论其组织形式(结构化，非结构化，等)和访问方式(SQL，APIs，Web 服务，等)的数据集成平台开始。保证该平台的扩展性——快速加入新的数据源或者修改现有的数据源的能力。
- 确保所选择的平台可以有效地支持任选数据处理的延迟，不管是批处理，类实时，还是变更数据获取及实时。
- 理解不同的数据访问模式，包括大容量的小数据集，大容量的大数据集，庞大(兆)数据集等等。并且保证它们都能被所选择的数据平台所支持。在需要的时候要有相应的附加技术作为补充。
- 保证服务所消费和产生的数据在服务间的一致。许多数据平台能够提供集成的数据剖析，能够不考虑复杂性而预先地认定及修复问题。
- 除了数据访问，一个数据平台通常还提供一个集中式的数据转换场所。如果是简单的格式转换，任一现代的数据集成平台都可以支持。然而，如果有复杂转换的需求，例如聚合，连接，查找，结构转换等等，你也许就需要更为 先进的平台。
- 一个数据集成平台通常需要支持服务实现要求的多种访问机制，包括基于 SQL 的访问，Web 及 REST 服务，等等。
- 一个数据集成平台需要将服务实现与底层数据源分隔开。实际上，它需要引入一个抽象层，以允许数据层的更改而不影响服务实现或将这种影响降到最低。

总的来说，一个数据平台：

“...需要是一个单一的，能够交付这里列出的所有功能的集成平台。这是有道理的，因为所有这些推荐都是从时间和节省成本的角度考虑的。如果每一项功能都需要使用一项单独的技术，那么就违背了使用面向服务方案的初衷，即通过简洁性与灵活性以保持系统的机动能力...[这一]平台必须支持并驱动数据集成逻辑的重用。

随着 SOA 实现的范围从有限的部门解决方案延伸到企业级的实施，企业数据访问的问题很快就成为了实施过程中最重要的问题之一。如果不能一开始就正确地进行构架，企业数据访问将成为实施道路上的一个拦路虎。

原文链接：<http://infoq.com/cn/news/2009/12/ServiceData>

相关内容：

- [SOA中的数据联邦技术解密](#)
- [ADO.NET实体框架引发争论](#)
- [在SOA中整合企业数据](#)
- [重新考虑“代码优先”的Web服务](#)
- [如何对企业数据进行架构设计？](#)



我们的**使命**：成为关注软件开发领域变化和创新的专业网站

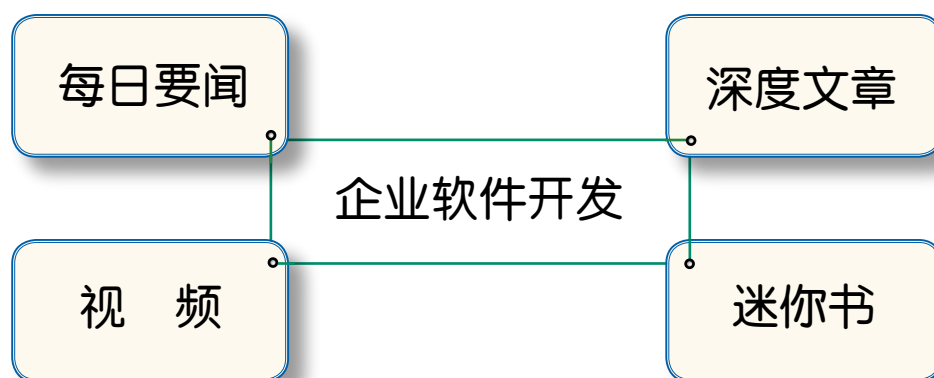
我们的**定位**：关注高级决策人员和大中型企业

我们的**社区**：Java、.NET、Ruby、SOA、Agile、Architecture

我们的**特质**：个性化RSS定制、国际化内容同步更新

我们的**团队**：超过30位领域专家担当社区编辑

.....



OOPSLA辩论：被高估的云计算

作者 [Morten Udnæs](#) 译者 [李重辉](#)

现在对云计算的炒作就如同上百人在电话会议中狂吼一样喧嚣。回顾 20 年来 IT 的演变，较为特别（其实也不那么特别）的一点就是每次新技术的诞生都发生了喧嚣的炒作。以 4 到 5 年为周期的技术更新意味着大量赚钱的良机。从最早的大型机到客户端-服务器、CASE 工具、.COM、企业架构(如 EJB 和 DCOM)、SOA，以及发展到现在的云计算，IT 一如既往地关注于如何想方设法赚钱。

云计算广受期许的原因是什么？首先对于一般开发人员和架构师来说，确实非常赞同‘IAAS’（基础设施即服务）的理念。无可置疑，这也难怪，由于在开发过程中 IT 部门和外包服务商常是制约项目进程的瓶颈，使得这些人过去多是生活在资源的“捉襟见肘”之中。但现在他们终于可以不受限制地开发应用并任意部署。

对于拥有复杂技术的大牌供应商来说，（据说）可以通过预先配置和包装来让那些缺乏足够技能和经验的用户也能够合理的配置并使用产品和服务。从而为它们的产品和服务开辟了一个全新的分销渠道。由于云计算缺少通用的标准，因为可以将用户的应用绑定在特定的供应商平台上，这样客户就会很忠诚——然后提供富余的硬件也是一条生财之道。

对于业务主管来说，云计算既支持按使用付费，又很容易平衡成本和收益——这个设想非常的美妙.....

对于学术研究者来说，云计算预示着可在诸多领域进行探索的良机。同时它还使用到了很多过去在并行计算，可伸缩性和虚拟化方面的研究成果。

云计算的真相是什么？

- 提供很多你不需要的东西
- 不是服务复用的圣杯
- 不是开发/托管全能的瑞士军刀

如果你只需一辆自行车，那么给你提供一架比自行车贵得多的航天飞机绝不是一件好事。问题的关键在于复杂度。它不工作该怎么办？它不如预期工作该怎么办？该如何操作它呢？除非自己是个火箭科学家，否则会很被动。你已经拥有所有（甚至更多）特性的功能需求吗？你在定制架构上拥有足够的技能和经验吗？如果这些条件都不具备，那么无论是用云计算还是别的什么就都没有多大区别了.....

“软件开发成本如此昂贵，所以我们要尽量复用”。这句话没错，但是请看这条新闻：四十年以来，我们一直试图复用代码但却从未成功！云计算在软件复用上也根本没有任何进步。复用要求非常相似的业务需求和上下文。而现在最主要问题恰恰在于每个用户的业务需求和上下文都有些细微差别。从技术角度上看，同类的云的集成服务会很容易，但在质量和安全领域上复杂度却会增加。简单数据服务（如地图/地理数据）的复用会带来一些好处，但这并不能改变用户仍然需要雇用开发人员来开发新功能这一事实。

IT 不仅指像 Facebook 和 Amazon 这类的互联网创业公司，它需要支持业务流程。除开少数流程可以通过云计算来实现，绝大多数都不可以。它们都需要云计算平台所不擅长的整合、安全和灵活。

SAAS（软件即服务）是个好主意，但也不够通用。即将出台的一系列标准将使 SOA 和服务看起来很容易，但是其中绝大多数都会被忽略掉，进而根本无法根据它们来做实现！重写软件又是非常昂贵且缺乏商业价值。基于上述问题，未来 5-10 年里将只出现少量的面向云的应用。而到那时，取代云计算的下一个技术革新将会发生。

云计算擅长做什么呢？以下场景都是不错的利好方案：

- 没有基础技术设施而又需要可伸缩性的创业公司。
- 地理数据，政务信息之类的数据中心。
- 桌面办公（邮件，浏览器，日历类应用）。
- 游戏（拥有服务端和三维渲染的廉价终端的这类）。
- 需要动态处理大容量数据的业务。
- 具有大容量数据和低安全要求的应用。

开发人员一直盼望的是什么呢？不是免费。IT 部门和（或）外包服务商应该提供丰富的包含安全、标准、补丁等自助服务，从而可以用来开发现实的“自行车”类方案的环境。实际上，学术研究者应该停止附和 Gartner 公司对云计算的不合实际的高估。把时间用来研究产品和

市场的需求，这才是云计算等新技术诞生并发展的源泉。

大牌供应商应该意识到“大量打包用户并不需要的东西”的做法并不合理。他们应该降低开发复杂度从而更好的帮助开发人员完成任务。

业务主管应该改变 IT 部门的工作方式。敏捷和精益思考的出现使 IT 和业务无缝协作成为可能。管理上则应该从成本导向（开发花了我多少钱）转为收入导向（上阶段发布的新功能让我们赚了多少钱）。

邀请您来挑战我们的观点并指出其中的错误。我们不需要一套套的理论和逻辑，我们需要你举出真实的场景。

原文链接：<http://www.infoq.com/cn/articles/oopsla-cloud-computing-debate>

相关内容：

- [在云计算中用经济效益衡量性能表现](#)
- [WordPress登陆Windows Azure](#)
- [IBM发布新服务，为云中的软件开发提供支持](#)
- [Geva Perry谈云中的软件应用生命周期](#)
- [云计算应用程序的几点设计准则](#)

构建自动化驶向何方？

作者 [John Smart](#) 译者 [曹如进](#)

时下大多数开发人员对持续集成（Continuous Integration，CI）的基本原理已经很熟悉，但是他们中只有一小部分人能够从优化 CI 设置中彻底受益。毫无疑问，一个有效的持续集成环境可以帮助你的团队节省时间、金钱甚至减少存有的顾虑。通过持续集成，我们可以更早地发现 bug，更轻松找出导致其发生的原因并最终有效地解决。持续集成可以更好地管理源代码，更有效地使用自动化分析工具，鼓励编写好的测试，跟踪进度以及突破开发人员活动中的瓶颈。持续集成可以让部署过程更简单且发布过程更加平稳和可靠。借助于 CI，管理者可以得到更多的图表，了解到更多的信息，而开发者可以专注于开发而不用过多交流，也就更加高兴。换句话说，如果不使用持续集成，那么就好比用记事本编写代码来开发软件，虽然可行，但是太低效了。

在本文中，来自 Wakaleo 咨询公司的首席咨询师 John Smart，将给我们介绍如何把持续集成由一个名义上的定时作业，变为开发活动中一个有效、而且能提高生产力的“中枢”。

持续沟通流

一个好的敏捷开发环境本质特征就是尽可能的最大化小组成员间的信息流。每一个开发人员都需要尽快地知道何时构建过程失败，或者何处改动可能会对应用程序质量造成不良影响。如果构建过程失败了，那么首要工作就是要知道做了什么改动，以及为什么做这些改动：所有的这些信息都应当在开发人员敲击几下键盘后就能通过最直接的方式获得。

即便是最基本的 CI 设置，它也会在构建失败后给开发人员发送电子邮件。其实我们只需要下一点点功夫，就能比现在做得更好。一个良好设置的持续集成服务器应该像团队沟通中的中枢一样工作。除了简单地发送电子邮件进行提醒外，其实还有很多事情可以做，例如现代 CI 工具中的许多特性——这些特性可以更平稳、更有效地向开发人员反映代码改动和构建失败的情况。

电子邮件大概是 CI 提醒方式中最古老也是最常使用的方式了，只是因为它非常普遍而且易于建立。尽管如此，事实上电子邮件是一个相对低效的提醒机制。当构建过程失败时，你希望被尽快地通知到，越快越好。如果因为电子邮箱客户端的更新而等上 15 分钟的话，那么可能就浪费掉了开发时间。除此之外，电子邮件通常很容易让开发人员分心。在企业里，每天的非紧急的或是不重要的电子邮件都会有很多。事实上许多优秀的开发人员都禁用电子邮件提示，只是在一天中定时地去检查几次邮件。

即时消息相对而言则是一个更加合适的提醒方式，有好几个原因。最重要的原因是，即时消息是（几乎是）即时的。你不再需要因为电子邮件客户端在服务器中检查新邮件而等上 10 分钟，使用即时消息，你能被立即通知到。它比电子邮件更加方便，而且阅读起来也更快。不仅如此，来自构建服务器的 IM 消息也会得到相关的重视，而不会淹没在大量无用的其它信息中。

当事件发生后，使用快速即时信息和花上 10 到 15 分钟使用电子邮件也许只相差几分钟，但千万不要低估这几分钟的重要性。那几分钟足够让一个开发人员丢失重点，并转移到其它的一些事情上，结果就导致了开发人员很难回到上下文中去修复问题。

使用即时消息的另外一个好处在于它可以扩展到桌面以外的应用。像 BeeJive 这样的应用程序可以将即时消息用到移动设备诸如 Blackberries（黑莓）和 iPhones 上。这样的话，即使开发人员不在办公室，也能收到至关重要的构建失败提醒。

即时消息提醒比起电子邮件，可以更多的与 CI 服务器进行交互。作为一个沟通的中介，即时消息要比电子邮件更加的自然，它还可以进行比 Subversion 提交信息还要多的交互。通过利用即时消息，如今的一些 CI 工具不仅仅可以发送提醒，还可以让开发人员更轻松地与构建服务器以及其他组员进行交互与交流。

当然，即时消息不是仅有的另一种可用提醒方式。如果想让大家注意到构建信息的话，最好采用一种能够融入相应企业文化的提醒机制。例如，一些公司使用社交网络的工具例如 Twitter，作为一个有效的内部交流渠道。对于这些组织而言，使用 Twitter 也可以算是一种有效的构建提示方法了。

保持构建过程高效率

持续集成环境的一个首要目标是要保证开发过程的顺利进行，并且避免由集成问题带来的障碍和开发延期。当集成问题突然出现时，开发人员应当有义务尽快的去提交一些代码来解决这个问题，以避免影响到其他开发人员。如果没有持续集成环境，开发人员通常会在集成

问题上卡住而找不到一个解决方案（“嘿，它在我的机器 上能运行啊!”）。

想让开发过程一直保持最好的状态，那么团队成员（尤其是团队领头人，流程专家等）需要能够监测构建过程，这样他们可以识别定位出日常生活里降低开发人员效率的问题。其中最好的方法就是去了解如何最好地使用构建感测。

构建感测是一个用在持续集成周期中，用以描述构建长期统计数据的术语。Bamboo，一个不错的 CI 工具，它就提供了很高级的构建特性。构建感测为你提供了关于构建要花多久，是否成功，以及解决这个构建失败问题要花多久等等之类的信息。这些数据很重要，因为它可以让你知道构建过程长期以来是什么样的。正是这种数据，而不是单个构建的结果，可以帮助你最终优化构建过程。

一定数量和频率的构建失败一直是一个不错的着手点。单独的构建失败通常不需要担心——你需要做的是去检查一系列反复出现的构建失败。当一个构建反复失败 时，也许是因为开发人员正纠结于一段非常麻烦的代码，或者是团队人员没注意到。虽然这两问题的原因不一样，并且解决的方法也不一样，但是它们都应当被进一步地调查。

通过深入分析测试结果，你可以了解到更多关于为什么构建会失败的信息。许多现代的 CI 工具都可以让你研究长期的测试行为，例如，把一直经常失败的、或者要花很长时间进行解决的测试跟其他测试隔离开来。如果同样的测试不断失败的话，这就意味着有什么地方过于复杂或者代码过于脆弱，这种情况下可以做一些重构。 除此之外，这些工具还能让你知道测试运行了多久，这是另一类问题发生的源头。

事实上，构建失败不是唯一减慢开发进程的原因。缓慢的构建过程是另一个罪魁祸首。

导致构建过程缓慢的最常见的原因是结构混乱的测试套件。经验丰富的 Java 开发人员常常会将单元测试与集成测试分开来。虽然两者区别大差不差，但是 一般来 说，单元测试是相对孤立的、较小的、快速的、轻量级的测试类。它们主要是用来确保类能够独立地正常工作。而另一方面，集成测试则较为缓慢，需要更长的运行 时间，并且可能会访问外部的资源如测试数据库，或者加载复杂的配置文件。它们被用来测试应用程序中的不同模块和类如何共同工作。性能测试和集成测试差不 多，但是两者的目标有所不同。

轻量级且运行很快的单元测试可以被迅速的执行完，并且在测试失败时能够给出很快的反馈。而另一方面，如果缓慢运行的集成测试与单元测试混在一起，那 么单元 测试将要花上更多的时间来执行，结果开发人员也要等更久才会得到关于单元测试失败的消息。解决之道就是为单元测试和集成/性能测试创建单独的构建计划。使 用这种方式后，如果单元测试构建计划失败了，由于其执行速度很快，开发人员不再需要等待很长时间才被通知到构建失败。

如果单元测试成功后，集成测试和性能 测试计划才会开始。

另外，还有一个补充方法——可以使用分布式构建。例如，如果你的 web 功能测试由于要在几个不同的浏览器上运行并因此消耗很长时间，那么可以为每一个浏览器设立一个构建作业，以让它们（可能不同的机器上）并行运行。

另一个问题来自过于缓慢和效率低下的测试用例。有许多方法可以用来监测这些缓慢的测试。提高测试上的运行时间意味着有些测试会由于所需时间太长而无法运行。这也许是因为它们被设计的很糟糕，也许是因为性能问题（需要进一步探究），抑或是集成测试伪装成了单元测试。

密切关注代码质量

持续集成服务器不应当仅仅是一个自动构建的机器。它应当成为你团队中沟通的中枢，尤其在代码质量上。时刻关注编码规范和一些度量值如代码覆盖率以及代码复杂度，它们可以让应用程序更加可靠且更易维护。

有很多优秀的工具可以帮助维护应用程序中良好规范的代码。静态分析工具如 Checkstyle，PMD 和 FindBugs，它们根据代码标准、最佳 实践或 者潜在的 bug 来对代码进行分析。你所有使用的工具如何配置要依赖于你想要达到的目标。例如，Checkstyle 更多关注于编码规范和最佳实践，而 Findbugs 则更倾向于找出错误的，破碎的或者危险的代码。所有的这些工具可以很轻松地集成到自动化构建过程中，并且可以和 Ant、Maven 一起很好地工作。

覆盖测试率是代码质量的另一个方面。它用来衡量测试执行时所访问的代码行数。Java 开发人员中对覆盖测试率统计的相对值仍然有着分歧。事实上，虽然它可以告诉你应用程序中的哪些行被执行，但是没法知道那些测试是写得很彻底、写得很好，抑或仅仅是简单的走马观花。总而言之，测试覆盖率不能保证你的测试质量 很高——只有人工进行代码检查时才能确保如此。然而覆盖测试率的度量值可以很好地展示出哪些代码没有被测试过。在 Java 世界里，用得最多的代码覆盖率测试工具当属 Clover 和 Cobertura，前者是一个非常强大的商业代码覆盖率测试工具，而后者是一个更加轻量级的开源工具。它们都可以很容易地集成到基于 Ant 和 Maven 的构建脚本中。

编码规范也可以作为非常有效的培训支持和指导活动，尤其对于那些经验不足的开发人员。CI 工具可以提供这些数据如何随着时间的推移而变化的高层次图 片，还可以关注开发人员在应用他们学到的技巧时做得有多好。例如，如果一个类只有很低的代码覆盖率，甚至没有，就意味着某个新的开发人员在消化吸收小组培训的测试驱动开发和测试实践上出现了问

题。这种方法还可以通过代码审查和定期的代码质量会议(讨论任何新问题或者动向的会议)完成。

一旦构建结束——自动化部署过程

构建应用程序只是开发生命周期中的一部分。一旦代码编译测试后,需要进行其他的活动,例如部署到阶段性(staging)环境、冒烟测试、功能测试和性能测试、准备发布说明和提醒QA人员最新的发布。

将最新的构建结果自动部署到集成服务器上是一件相对简单的事情。而将其部署到阶段性环境或者生产环境下,则需要涉及一些与常规构建作业不一样的工作。一般而言你需要一个更加严格,更加正规且有更多可跟踪性和问责的过程。它通常涉及到的任务如下:

- 为阶段发布标记源代码
- 编译测试应用程序
- 发布构建产品
- 将应用程序部署到阶段性环境中
- 运行数据库更新脚本或者其他特定环境脚本
- 运行冒烟,功能和性能测试
- 准备并发布产品说明
- 提醒关于最新阶段发布的相关利益人

这通常是一个手工任务,但是其中的大部分工作没有理由不能自动化。事实上,开发生命周期中的自动化包装,部署和发布具有很稳固的商业意义。一方面自动化能够得到更加可靠的构建:计算机不会忘记部署过程中的某一步,也不会发生在测试失败后继续进行。它还能够节约开发人员的时间:阶段性发布由之前几小时的shell脚本编程变为了只要点击一下按钮。它比以前的速度要更快,并且可以在没有人的情况下完成工作(例如,通宵或者午休时间)。

像Maven 2这样的工具也能够帮助自动化一些步骤。Maven Release插件使得Maven的用户能够自动化处理一些如“更新版本号”,“Subversion中新增标签”,以及“向Maven存储库中发布构建产品”的工作。它可以用来管理阶段构建,并决定在不同的环境部署不同的发布产品。尽管如此,一旦产品构建结束并且可以部署到阶段性环境者生产环境时,这个过程就

会变得更复杂。

千真万确，现实世界中的部署步骤数目经常要比简单的用一个 WAR 文件多。根据应用程序架构和产品平台，你可能需要在阶段性环境或者生产环境下的数据库中运行 SQL 更新脚本、用一个专用的工具部署 web 服务、运行自动化的冒烟测试或者做一定量服务端的工作。

CI 可以帮助简化比这些还要复杂的步骤。例如通过分布式构建，你可以设立阶段性环境或生产环境上的构建代理，并在该机器上直接运行相应的任务。几乎所有的现代 CI 工具都支持相当好的安全模型，目的是为了将应用和产品环境限制给一些特定的人，以及跟踪谁在什么时间运行了什么构建。

这是 CI 的一个相对较新的应用，不同的工具在处理应用程序部署时使用的方法也不一样。有一些，如 Hudson，允许在构建作业中定义多个步骤，只有当前一个步骤成功后，才能执行后续步骤。其他的像 Cruise 和 Anthill Pro，都尝试将部署生命周期中的如阶段和生产环境直接集成到构建工具中，尽管有时候这样会带来额外的复杂度开销。

还有更多低层次的操作可以和 CI 服务器联合起来使用。一个选择是使用诸如 Ant 或者 Maven 的构建工具。Ant 对于这种类型的脚本特别灵活。另一个流行的选择是古老的 Makefile，或者 Unix 上的 shell 脚本。它们的缺点是操作系统相关的，并且对于那些不熟悉 shell 脚本精髓的 Java 开发人员来说很难掌握。想要与 Java 更加友好，可以选择动态语言诸如 Groovy 或者 Gant（一个使用 Groovy 而不是 XML 来制作 Ant 脚本的工具）。Groovy 在提供所有轻量级的动态脚本语言中所有的优点的同时，也保留了对 Java 开发人员的熟悉程度和可读性。

总结

这些只是使用现代持续集成环境完善构建过程和增强团队的几个方法。持续集成环境绝对不仅仅是一个构建计划表，它还可以用来帮助打开队伍内部的沟通渠道、保持构建过程平稳有效的运行、时刻监控代码质量以及自动化发布和部署过程。

原文链接：<http://www.infoq.com/cn/articles/build-automation-ci-atlassian>

相关内容：

- [使用Clojars与Leiningen自动管理Clojure类库及依赖](#)
- [用psake来简化自动化脚本的构建](#)
- [代码规范的自动化监管](#)

使用Java构建高伸缩性组件

作者 [Zhi Gan](#) 译者 [崔康](#)

随着多核处理器成为主流,应用开发人员对于如何编写高伸缩性的应用以利用底层硬件的优势这个问题面临巨大的压力。此外,遗留系统不得不移植到新的架构上。保证应用伸缩性的一种有效方式是使用高伸缩性组件构建应用。举例来说,在各种应用中,`java.util.concurrent.ConcurrentHashMap` 可以替代同步的 `HashTable`,使应用伸缩性更好。因此,向应用 直接提供一套高伸缩性构造块以引入并行是非常有用的。

我们创建了一套高伸缩性并发 Java 组件作为 Amin 库项目的一部分。在本文中,我们将介绍一些创建该开源库的想法。

概述

作为软件工程师,我们不得不并行我们的应用使它们在多核机器上很好的伸缩。一种并行方式是把它们分成若干子任务,其中每一个子任务与应用的其他子任务通信和同步。这通常被称为基于任务的并发。我们可以基于子任务的通信模式将应用分类。

- 尴尬式并行——应用的子任务从不或者很少与其他子任务通信。通常情况下,每一个任务只处理自己的私有数据。一些例子包括：
 - 蒙特卡洛模拟程序
 - 快速排序程序,使用 `fork/join` 模式很容易实现并发。当我们处理尴尬式并行应用时容易得到很好的伸缩性。
- 粗粒度和细粒度式并行——这些应用的子任务需要互相通信。根据子任务之间的通信频率可以把程序分为粗粒度和细粒度并行。一个例子是生产者/消费者问题。生产者产生驱动消费者的数据。从生产者到消费者的数据发送需要通信。相比尴尬式并行应用,处理子任务频繁交互的应用较难得到良好的伸缩性。

经过为伸缩性而优化的组件非常有益于解决这些困难问题。举例来说，如果存在一个高伸缩性队列组件，那么实现生产者/消费者伸缩性相对容易些。本文中，我们提供了若干通用技术提高软件组件的伸缩性。

针对伸缩性的分析

应用分析是开发过程的重要方面。分析是为了了解应用的运行特征。分析数据经常被用于提高应用的性能和伸缩性。大多数分析器遵从一个简单的执行模式，其中有一个配置阶段。在该阶段，根据信息获取的类型，会配置不同层次的计算堆栈。硬件计数器可以被用来监控硬件事件。操作系统可以通过配置各种操作系统事件来监控。如果在计算层次中还有虚拟机，那么需要配置以访问虚拟机的事件。最后，应用需要配置以得到真实应用的事件。优秀的分析器能够在不要求应用重新编译的情况下完成这些配置。大多数分析器通常都配置虚拟机和应用层。应用在配置之后运行时，会生成跟踪信息，例如函数运行时间、资源使用情况、硬件性能参数、锁使用、系统时间、用户时间等等。不同的分析器会在把跟踪信息输出到文件系统前做一些计算。最后，分析器会通过显示界面可视化这些数据。

用于并发应用的分析器需要提供有关线程、锁竞争和同步点的详细信息。我们使用了下面一些分析器作为 Amino 性能分析：

- Java 锁监控器 (Java Lock Monitor) ——Java 开发人员使用该工具分析应用中锁的使用情况。在运行时，监控组件收集各种锁数据用于发现锁竞争。该工具能够通过表格展示详细的锁和竞争信息。IBM Java Lock Analyzer (JLA)是另一个提供锁信息的工具。它与 JLA 类似，只是通过图形界面显示缩合竞争数据。两个工具都支持 x86 和 Power 平台。
- THOR ——该工具用于详细分析 Java 应用的性能。它能够深入分析完整的执行栈，从硬件到应用层。信息来自于栈的四个层面——硬件、操作系统、JVM 和 应用。用户可以看到锁竞争的信息以及对应的代码、瓶颈、多核之间的线程迁移和竞争导致的性能下降。该工具支持 x86 和 Power 平台。
- AIX 性能工具——IBM 的 AIX 操作系统自带一套底层的性能分析和调试工具。其中包括：
 - Simple Performance Lock Analysis Tool (SPLAT) ——该工具是 AIX 系统上的一个锁分析工具，可以通过命令行运行，用于分析各种内核级的锁。同时，它也可以分析和显示用户级锁（读/写和 mutex）的竞争。应用必须启用跟踪选项，SPLAT 需要用到这些跟踪数据。
 - XProfiler——该分析工具基于 X-Windows，用于 C 语言应用的函数级分析，能够显

示在用户代码中计算敏感的函数。如果要使用 XProfiler，代码需要使用特殊的标志（-pg 选项）编译。

- prof、tprof 和 gprofncbr——各种 prof 命令都可以用于分析用户应用。prof 命令能够得到所有应用中的外部符号和函数。gprof 是 prof 的超集，可以获得应用的调用关系图。最后，tprof 用于得到应用的宏观和微观的分析信息。tprof 能够得到各个指令、子程序、线程、进程的计时数据，还有用户模式函数、库函数、Java 类、Java 方法、内核函数、内核扩展等的处理器使用情况。

减少热点

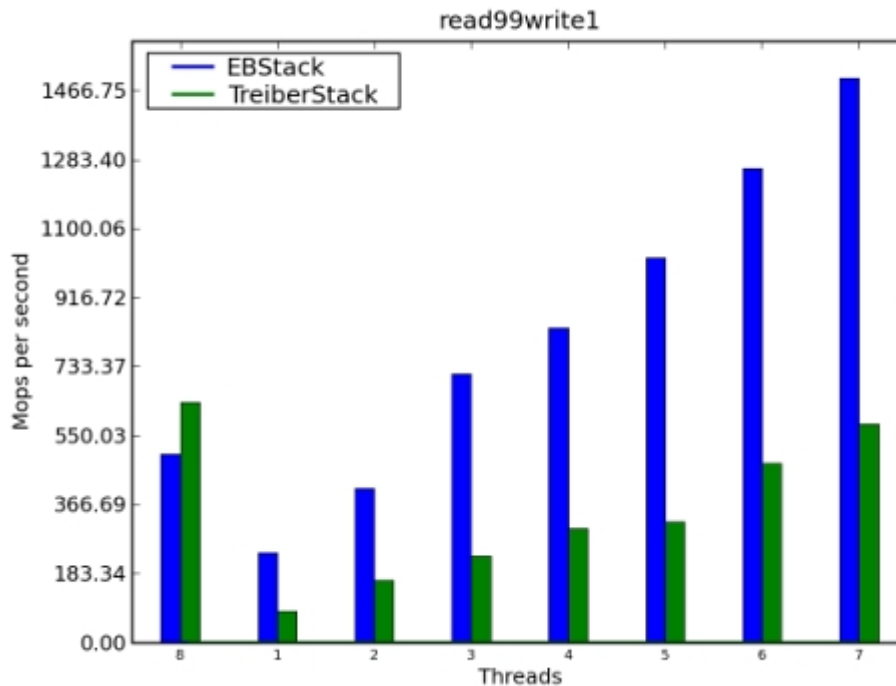
传统的性能调优方法，比如适用于顺序应用的方法也适用于并行应用。在完成传统调优之后，我们需要检查共享数据是如何被多线程访问的。通过使用 JLM 或者 JLA，我们能够发现线程在访问共享资源时是如何竞争的。

举例来说，如果应用有 100 个线程，并且所有的线程都需要从一个 `java.util.HashMap` 提取/存储元素，该应用的性能由于线程竞争会下降。每一个线程需要在访问该 `HashMap` 之前等待很长时间。

如果我们使用类似 JLM 的工具就会发现，该 Hash 表关联的 monitor 使用非常频繁。解决这一问题的办法是使用能够替代 `HashMap` 的高扩展性组件。在本例中，我们可以通过 `java.util.concurrent.ConcurrentHashMap` 替换该 hash 表。`ConcurrentHashMap` 把它的内部数据结构分成若干段。在应用中使用 `ConcurrentHashMap` 替换 `HashMap` 替换使线程针对多个子组件竞争而不是单个大组件。使用 `ConcurrentHashMap` 的应用产生竞争的几率比之前要小得多。

还有一种办法降低访问共享组件的竞争几率。如果一个组件具有相反语义的操作，如栈的压入和弹出，这两种操作可以无需接触核心数据结构即可完成。这种技术最早由 Danny Hendler、Nir Shavit 和 Lena Yerushalmi 引入，已经在 Amino 库中实现。这种方法的性能改善见图 1。

图 1.性能比较：EBStack 和 TreiberStack



使用无锁/无等待算法

传统上，大家都采用基于锁的方法来确保共享数据的一致性和对关键区域的互斥访问。锁的概念易懂，但基于锁的算法引起了很多挑战。其中一些著名的问题包括死锁、活锁、优先级倒置和锁竞争等。锁竞争会减少组件和算法的可扩展性。

无锁和无等待算法到现在已经存在二十多年了。它们被认为可以能够解决大部分与锁有关的问题。这些算法支持在不使用任何锁算法的情况下更新共享数据结构，不仅如此，还有助于创建扩展性良好的算法。最初，这些无锁和无等待的算法纯粹是理论兴趣。但是，随着算法社区的发展和新硬件的支持，无锁技术在现实产品中得到了越来越多的应用，比如操作系统内核、虚拟机、线程库等。

从 JDK 1.5 以后，JDK 包含了这些无锁算法，比如 `ConcurrentLinkedQueue`、`AtomicInteger` 等。它们的伸缩性通常比对应的基于锁的组件要好。当我们在 Amino 库实现新组件时，我们会使用目前研究的最新无锁算法。这使得 Amino 数据结构扩展性和效率非常好。但是有时候，特别是在低核硬件上，无锁数据结构比基于锁数据结构的吞吐量差，但是一般来说，它们的吞吐量比较好。

尽可能减少比较—交换（CAS）操作

从 JDK 1.5 以后，JDK 包含了由 Doug Lea 开发的位于 `java.util.concurrent` 包的无锁的 FIFO（先进先出）队列。该队列采用的算法基于单链表的并发操作，最初由 Michael 和 Scott 开发。它的出列操作需要一次比较—交换，而入列操作需要两次比较—交换。这对入列和出列操作来说太容易了。但是分析数据（图 2）显示比较—交换操作占用了大部分执行时间，同时入列操作需要两次交换—比较，这增加了失败的概率。在现代多处理器上，一次成功的比较—交换操作要比一次加载或者存储的时间长一个数量级，因为它们需要独立占用和冲刷处理器写缓存。

图 2. CAS 操作的执行时间

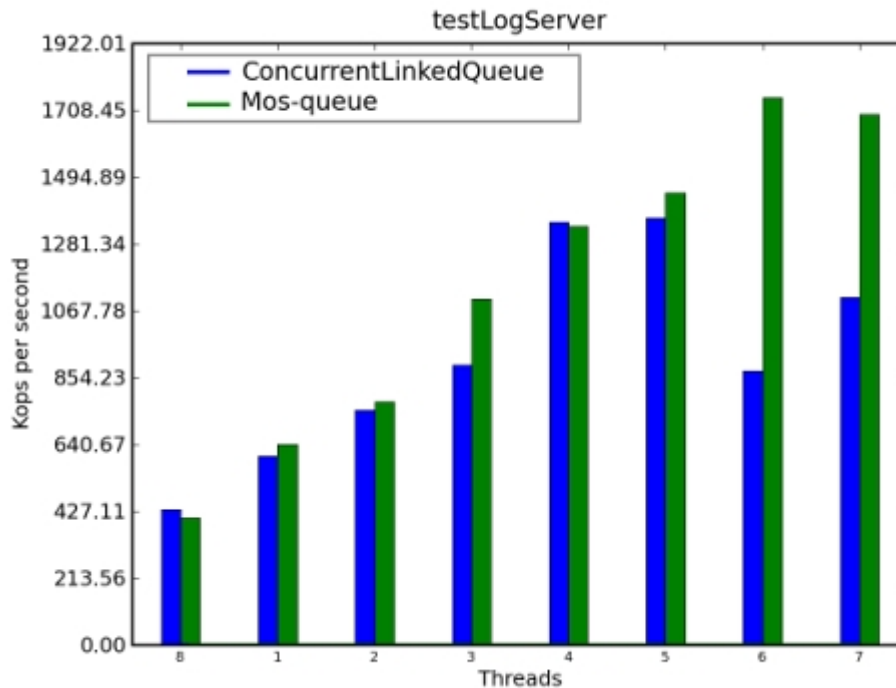
0x000001EB	0x48C1E807	SHR RAX,7	0.40	56	0
0x000001EF	0x83C07F	ADD EAX,7FH	0.46	64	0
0x000001F2	0x83E07F	AND EAX,7FH	1.11	155	0
0x000001F5	0x48C1E007	SHL RAX,7	1.01	142	0
0x000001F9	0x4809C2	OR RDX,RAX	0.89	125	0
0x000001FC	0x4889C8	MOV RAX,RCX	0.81	113	0
0x000001FF	0x4881C200	ADD RDX,10000H			0
0x00000206	0xF0480FB1	LOCK CMPXCHG QWORD PTI	0.90	126	0
0x0000020B	0x0F94C0	SETB AL	46.08	6463	0
0x0000020E	0x84C0	TEST AL,AL			0
0x00000210	0x0F848EFE	JE 0A4H			0
0x00000216	0x5B	POP EBX	0.90	126	0
0x00000217	0x5D	POP EBP	8.33	1168	0

从图 2 可以看出，CAS 操作执行时间的百分比为 46.08%，几乎占了全部执行时间的一半。分析数据有一个操作的延迟。真正的时间是在“SETB AL”指令之后发生的。

Mozes 和 Shavit（MoS-queue）在它们的无锁队列算法中，提供了一种新颖的办法来降低入列操作时 CAS 的数量。其关键点在于替换单链表，因为其中的指针在插入时需要浪费一次 CAS 操作，取而代之的是采用一个双链表，其中的指针在更新时只需要一次存储操作，如果事件乱序导致双链表不一致则可以修复。

我们做了一次基准测试来比较 `ConcurrentLinkedQueue`（JSR166y）和 `Mos-Queue` 的性能。结果在图 3 中显示。对于大量线程来说，`Mos-Queue` 的性能优于 `JDK ConcurrentLinkedQueue`。

图 3. 性能比较：ConcurrentLinkedQueue 和 Mos-Queue



更多有关 Mos-Queue 的信息参见 Mozes 和 Shavit 的论文《An Optimistic Approach to Lock-Free FIFO Queues》。

减少内存分配

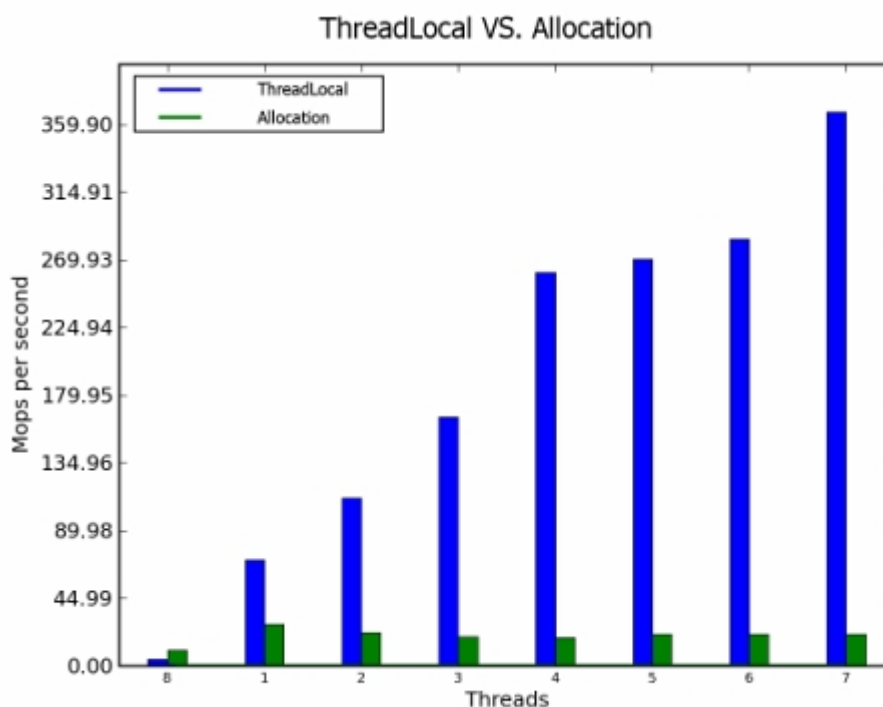
Java 虚拟机拥有一套强大、有效的内存管理体系。垃圾回收器能够压缩现有对象，以确保在垃圾回收之后 Java 堆里没有空隙。因为空闲空间在垃圾回收之后都是连续的，所以内存分配不多是增加了一个指针而已。

这对多线程应用同样有效，如果内存使用不敏感的话。JVM 为每一个线程分配了一个线程范围的缓存。在内存分配时，线程范围的缓存首先被使用。只有在线程范围的缓存被耗尽之后才会使用全局堆。

在线程范围内分配内存对应用性能非常有益，但是如果分配过于频繁，则会不起作用。根据我们的经验，线程范围的缓存在分配频率高的情况下会很快耗尽。

如果在循环中需要暂时性的对象，则线程范围的类会比较有益。如果我们在线程本地对象中存储临时对象，我们可以在每一次循环迭代中重用它们。虽然，线程本地类会增加额外的负担，但是大多数时候都比在内存中频繁分配内存要好。

图 4. 性能比较：线程本地和分配



结论

在本文中，我们介绍了使用 Java 创建高扩展性组件的几个重要原则。一般来说，这些原则通常很有帮助，但是它们无法替代谨慎的测试和性能调优。

资源

- JLA, IBM Lock Analyzer for Java
- "[CAS-Based Lock-Free Algorithm for Shared Deques](#)" by Maged. It is a highly efficient algorithm of lock-free deque.
- "[Simple, fast, and practical non-blocking and blocking concurrent queue algorithms](#)" by Michael and Scott. It is a highly efficient algorithm of lock-free queue.
- "[An Optimistic Approach to Lock-Free FIFO Queues](#)" by Mozes and Shavit. It is another highly efficient algorithm of lock-free queue.
- In the Website for [Amino project](#), get the sources of Amino project website.
- Browse the [technology bookstore](#) for books on these and other technical topics.
- A Scalable Lock-Free Stack Algorithm, Danny Hendler, Nir Shavit and Lena Yerushalmi, SPAA

2004, pages 206-215.

关于作者

Zhi Gan 是 IBM 中国开发中心的软件工程师，在上海交通大学获得计算机安全方向的博士学位。Gan 博士在 SOA（面向服务的架构）、AOP（面向切面编程）和 Eclipse 领域拥有丰富的经验。他目前主要关注于使用 Java/C++ 进行并行软件开发。

Raja Das 是 IBM 软件集团的软件架构师。目前，他正在开发支持多核/众核系统的库和框架。他之前是 WebSphere® Partner Gateway 的产品架构师。Das 博士感兴趣的领域包括编程语言、并行软件和系统。

Xiao Jun Dai 是 IBM 中国开发中心的软件工程师，在中国科学院软件所获得计算机科学硕士学位。他在敏捷方法和编程语言方面拥有丰富的经验。他目前关注于并发编程和多核系统。

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

相关内容：

- [JRuby新IR奠定未来性能提升之路](#)
- [Node.js基于Google V8 提供了基于事件I/O处理](#)
- [在云计算中用经济效益衡量性能表现](#)
- [Silverlight 4 中的高速通信](#)
- [PDC 09：并行和异步编程中的挑战及F#的应对方案](#)

为网站和智能手机构建FlightCaster前台应用

作者 [Werner Schuster](#) 译者 [张文钊](#)

本文是采访[FlightCaster](#)团队的第二部分内容（[第一部分讲的是FlightCaster如何使用Clojure](#)），主要集中在Rails和Heroku的使用方面，如何从多个数据源收集并整合数据，为不同手机设备创建多种用户界面并将它们集成到系统当中。

InfoQ采访了Flightcaster的James Bracy、Jon Bracy、Jared Strate、Jonathan Chase，以及[Inmite](#)（FlightCaster的黑莓合作伙伴）的Pavel Petre。

InfoQ：你们如何在 Web 前台处理长时间运行的过程？

Jon：我们使用 Heroku 的 Delayed Jobs 插件来实时捕获多种来源的数据。如果我们不持续捕获数据，它们就会丢失，所以可靠性是非常重要的。目前，DJs 一天要处理二百万次的更新。每个数据源采集器都有自己的工作者队列。如果一个 DJ 挂了，Heroku 就会启动另一个继续工作。如果需要更多的工作者，简单的跑个指令即可。我们还用 DJs 来做批处理、格式化及转换。例如，时间就是在 DJ 里被格式化的。

我们碰到的唯一问题是在拉取 FlighStats 数据时的内存限制，因为 XML 文件实在是太大了。老实说，这暴露了我们在处理数据上的缺陷，并迫使我们采取更好的处理方式。因此，我们使用了 SAX 解析器而非 DOM 解析器。

InfoQ：在你们网站上线的第一周，对 Heroku 感觉如何？

Jon：Heroku 会根据网站前台需求自动扩展，即便是在 New York Times、Wall Street Journal、Reuters、Techcrunch、InfoQ 和 slashdot 都在报道 FlightCaster 时的峰值期间也没什么问题。在此期间，我还在线部署了一些小的修改，并没有什么不便之处，运作都十分正常。它们也有内建缓存，因此花两分钟设定之后，就没什么太担心的了。另外，对于数据捕获及原始输入数据处理，我们还得到了极佳的扩展能力。

InfoQ：到目前为止，你们对 Rails 有何体会？

Jared：我们的网站不全是 Rails，还有一些 Javascript/Ajax 来简化输入和提升用户体验。航班延误因素并不是算法的一部分，它们是通过把逻辑应用于航班数据而得出的。程序分析天气数据（航空例行天气报告）并应用一些基础逻辑来判断与天气相关的延误，然后通知旅客天气情况。分析官方情况、预计抵港/离港时间来判断一些明显的航班延误。分析进港航班数据来判断明显的延误。我们还做了一些简单的阈值数学运算以决定如果显示预报。除此之外，时区也是一个大问题，这取决于用户、进港航班、目的地等的位置。

InfoQ：你们使用了 Rails 的哪些部分？全部的 ORM、REST 吗？

Jon：Rails 所有部分在不同的地方都用到了，没有哪一部分没有用到。但我们从没有使用 Rails 用到的一个属性域。我们一开始就使用 'updated_at' 和 'created_at' 属性域来存放更新数据时的时间戳信息，而不是在收到数据并将其放入数据库的时间。为此，但是你得告诉 Rails 不要写这两个属性域，而且你必须始终记得要这么做。

Jared：用户输入被加以分析并自动补全；Rails 的自动补全功能棒极了。

InfoQ：数据是如何从后台抓取到前台的？

Jon：Rails Web 服务器读取由 Clojure/Cascading/Hadoop 端产生的 JSON，用其解释输入的实时数据并形成预报。客户端通过 REST API 接收预报和实时数据。客户端只是用来查看数据的，逻辑保留在服务器上。

InfoQ：你们是如何获取数据的？是否有什么地方能够获得延误及其他所需数据？

Jon：目前我们从联邦航空管理局（FAA）获取航班数据、从国家海洋气象局获取天气信息、从 Flightstats 获取航空公司信息。我们也开始与某些航空公司接触以便能直接从他们那获取数据。

InfoQ：Mashup 很流行——你们在整合来自多个数据源的数据的时候有没有碰到问题？

Jon：日期和时间一直是最大的问题。时区和夏令时都必须考虑进去，不然你可能会在搜索今天的班机时却得到明天的。要找到机场当地的时区也很困难，好在机场的经度和纬度比较容易获得，因此我们可以利用时区向量地图用经纬度来确定时区。

有些数据源也会给我们无效的日期和时间（比如 26:00）。像这种无效的日期，通常丢掉即可。尽管我们能得到关于该无效日期的通知并记录下来，但我们系统的数据总量允许我们这样做（丢弃），这取决于你要做什么。

我们有很多关于航空公司和 FAA 数据的问题。IATA、ICAO 和 FAA 给的航空公司和机场代

码导致出现许多问题。比如，IATA 航线代码是两位字符，但规格说明它可以是三位字符。而且，飞全球不同地区的航空公司可以共享同样的 IATA 代码。ICAO 代码相对比较好处理一些。全体航空公司/机场的标识符是个很大的问题，因为三个组织给出的是三个不同的标识符，而且所有标识符是可选的。所以即使我们以 ICAO 代码为主，我们也不能期望一定会有 ICAO 代码可以使用。

处理政府数据也是非常痛苦的。例如，FAA 将 ICAO 代码存放在指定给 FAA 代码的属性域。我们还发现 Flightstats 会报告一个虚构的航空公司（大洋航空）。

InfoQ：你们的手机界面复杂么，用到类似定位或加速度计这样的设备特有功能没有？

James：我们没有用到任何设备特有功能。我们想过增加‘晃动有机随机选取航班’的功能，但是每个应用程序都已经有了‘晃动……’的功能。实际上这些功能并不那么重要，这一版也就没做这种功能。

Pavel：也许有机会用到 GPS，比如根据当前地点搜索机场等。

InfoQ：你们考虑过跨平台的 HTML5 解决方案吗？

Jamas：HTML5 解决方案还未成为主流。或许可以了解一下，但我们更愿意使用本地 API。

InfoQ：手机应用是如何连接到 Flightcaster 后台的？

James：我们的后台就是一个 RESTful API over HTTP，我们所有的客户端应用都是用它。

InfoQ：对于 iPhone、黑莓和 Android 手机，你们有什么体会？

Pavel：要想在 RIM OS 4.2.1+黑莓手机上达到 iPhone 应用程序的外观效果，唯一可能就是全部自己写。与 Android 相比，在黑莓上开发一个“漂亮的程序”需要做许多额外工作。对于黑莓手机，在安装包大小、多种分辨率支持及多操作系统版本之间寻找平衡非常重要。我们最终提供了两个版本：一个是为老版 4.2.1 设备准备的；另一个给 4.3 及以上版本设备（包括 5.0 触摸设备）准备的。这两个版本都提供了一些额外的代码来适应实际设备。而在 Android 上，支持不同配置要容易得多。只需指定不同的配置目录（比如 layout-en-finger-480x320）覆盖默认配置即可。由于手机有多种数据连接方式、运营商、企业级的 BB 策略，因而用通用程序来切换传输方式（WiFi、BES、BIS、直连 TCP、...）变得更加困难——我们给用户提供了选择传输方式的界面。在这方面我们正在做一些改进，这样大多数用户就不用进入该设置界面了。但是很不幸，黑莓对通讯层的处理不像在 Android 上那么简单，Android 可以自动切换连接。

James : 在 iPhone 上开发 GUI 相对简单，但性能不如期望的好。例如，应用程序加载时第一个视图滚动得不是很流畅。我们很快找到了问题所在。iPhone 社区和文献都非常不错，有足够的信息可以让你确保应用程序正常工作。iPhone SDK 也是经过周全考虑的。这是我们做的第一个手机应用，此后我们还开发了一个 Android 应用。相比起来，iPhone 应用开发起来最容易，而且用起来很惬意。在开发 Android 时我发现其 SDK 也不错，但是设备响应不如 iPhone 快。而且即便是和 Objective-C 比，用 Java 写程序也非常啰嗦。Apple 以设计著称，这一点从 GUI 设计方面展现出来了。界面构建器确实让设计变得简单了。在 iPhone 上编程时让我感觉特别有趣的一点是，手动分配内存实际上帮了我的忙，它让我放慢了速度并真正彻底理解其工作方式。

Jared : 在 Android 上，获取数据并传进视图比预期工作量要大。Java 太啰嗦了，或许在手机平台上用另一种不同的语言可能会更理想。用基于 xml 的布局简单输出到视图上很容易，然而复杂的视图开发起来既慢又难，特别是对非一致数据更是如此。不过，自动补全的文本域却是一个吸引人的特性，而且简单易用。

Chase : 在我们利用 Apple 绘图方法而非高层 API 改写了部分视图后，iPhone GUI 响应得到了改善。展现迟缓的原因是我们使用了透明特性，这在 iPhone 上非常耗性能。这个应用程序上架审核的过程跟我们预想的差不多（大约两周）。尽管我们已经修正了一些大 bug，它们还是导致该应用程序只获得了 1 星评价。

原文链接： <http://www.infoq.com/cn/articles/flightcaster-heroku-rails>

相关内容：

- [DeMarco反思 40 年软件工程发展之路](#)
- [语言约束和责任感，我们应该信赖谁？](#)
- [跨平台开发——Banshee/Mono启示录](#)
- [Wolfram|Alpha，菱形六十面体背后的细节](#)
- [当SOA遇到形式化方法](#)

SOA与服务识别

作者 [Rathina Dhandapani](#) 译者 [胡键](#)

简介

面向服务架构(SOA)已被作为一种通过对齐 IT 和业务来促进业务机动性的方法被广泛接受。这种方法的主要不同之处在于,用相对较低的成本就能轻易地获得某种机动性。在较高的层次,该方法试图将第 n 次业务变更所产生的增量成本降低到零或接近于零。组织推行 SOA 项目目的就是为了在他们的 SOA 之旅当中尽早达到这个难以捉摸的第 n 个迭代。在实践中,要达到这种最佳状态的时间可能得花数年,甚至更长时间。本文正是为那些参与 SOA 项目的业务分析师、企业/系统架构师而写的。

WYI2WYG (所识即所得 , What You Identify Is What You Get)

服务识别是迈向 SOA 终点漫长旅程的关键第一步。它是一个迭代步骤,需要在分析和规划阶段就要尽早地组织和发起。该步骤决定了整体的服务蓝图,蓝图 中包含了被识别的服务,它们的名字、分类、优先级,甚至还对相关的实现过程进行了恰当的路线图阶段划分。服务识别阶段奠定了基于服务的生态环境的基础,本文则指出了业务服务识别相关的最佳实践。

1) 了解项目的性质

根据所处环境, SOA 项目可以分为以下类别:

1. **SOA 转型**: 这是指企业所开展的那些将现有项目向 SOA 转型的项目。几乎所有应用是功能成熟的,虽然不适于快速变更。在这种状态下的企业通常面临的是对于变更的刚性阻抗。此处的重点在于,在解决服务空白的同时从遗留应用和商业成品软件(COTS)中暴露和(或)重新设计服务。
2. **SOA 采纳**: 这是指那些已经拥有服务于关键业务流程的应用的企业所实施的项目,尽管新应用的范围是支持某个其他已经存在的核心业务功能。这里的重点是,开发新的应用和服务,同时逐步重新调整现有应用。

3. **SOA 启程**：这适用于基于服务的产品开发，以及那些拥有需要再造的分离系统的企业。这里的重点是从零开始审视功能和开发服务。通常会采用契约优先的方式。这种企业处于一个相对不错的位置，可以实现长期的回报。

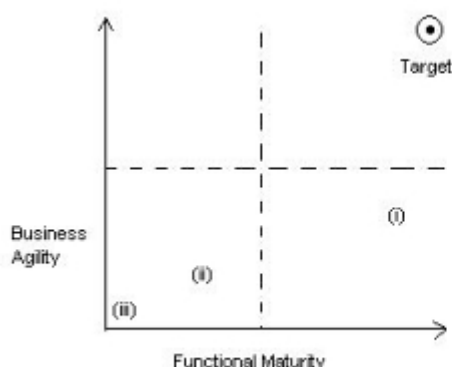


Figure 1

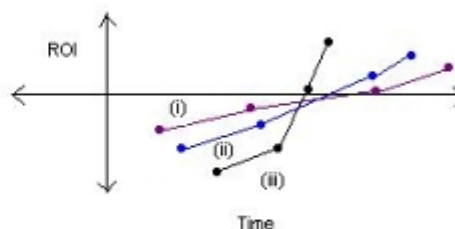


Figure 2

图 1：本图显示了各种初始业务的机动性和功能成熟度，同时还显示了通过 SOA 实现的最佳状态。

图 2：本图显示了四种假想但关联的变更随时间推移的关联 ROI。在 SOA 转型 (i) 分类下的企业利用功能成熟度可以快速、极早的获益，但随着时间的推移回报相对平缓。相反，在 SOA 启程 (iii) 分类下的企业，从长期来看，借助可预期而且更快推向市场的时间的潜力和更低的集成成本，可以获得快速上升的回报。

确定分类将有助于确定重点和期望值，同时也有助于明确服务识别方法。

2) 业务牵头

在全球经济低迷的背景下，展示切实、快速的投资回报率 (ROI) 的需求在升高。在竞争日益激烈的环境中，业务团队和 IT 团队应当紧密合作，借助彼此能力迅速支持激烈、非常规的业务举措。需要一再强调服务的识别，这在很大程度上是一个业务引导、IT 跟进的步骤。

3) 跟企业远景和相关驱动力一致

来自战略、远景和长期业务目标/目的的投入对于建立一个强大、可适应变化的基础至关重要。这些投入在 SOA 线路图及相关阶段中得到了体现。它通过注入面向未来的服务蓝图抽象层，以及延长服务契约的使用寿命来影响服务的识别（如，专门从事信贷产品的金融企业可能有意尝试保险和经纪业务，这可能会要在蓝图中加入一些产品无关的服务）。

4) 关注端到端的业务流程

这些是形成流程蓝图主体的主要流程，包含了核心和非核心功能。这些流程通常可以在不同的抽象层次来描述，这种层次在流程模型中被称为流程层次。这时，可以采用自顶向下的方法，将服务从多重这样的层次中抽取出来。高抽象层产生组合服务，低层次产生细粒度的服务候选。如此关注流程和服务候选将有助于识别整个企业的功能冗余。不管是否考虑 SOA 项目的性质（上文已经指出），这个实践很重要。

5) 利用工具加速识别

SOA 规划和治理工具可以简化和加速服务识别和沟通的过程。它们甚至能把服务之间的关系可视地描述出来。服务仓库相关的工具还有助于识别服务变更的提议所带来的影响

6) 重用行业制品

多年来，在垂直行业中，诸如电信、公共事业等，拜相关组织（IFX，SWIFT，OAGi，Accord，ETIS，ISO 等）所赐，SOA 领域有了长足的发展。少量的跨行业制品也可以找到。可以看看并利用这些标准化的服务契约（接口和操作）、数据模型和第三方服务的列表。这会让你大力推进迈向 SOA 的行动，因为大部分这些制品（基于 XML）很可能是可扩展和向后兼容的。

然而，这些制品在使用前必须经过周密的调查研究。例如，当采用这些标准的时候，企业可能需要询问这样几个问题：该数据模型是部分采用还是全部采用？是否有必要对规范模型提供本地支持，或者是只要限制它与外部集成的接触点就够了？是否还有什么潜在成本？它有没有得到厂商广泛的支持？

7) 建立契约基线

服务契约应该同时强调功能和非功能的能力。基线需要被建立起来，可以通过识别作为契约一部分的相关属性来完成。功能属性可能包括诸如服务描述、消息结构和数据模型这样的细节。非功能属性可能包括诸如服务质量（如响应时间，可用性），成本构成（数量或周期）和安全性这样的细节。这样的基线标准化了 WSDL 文件、XML 模式和 WS-Policy 定义（加强行为约束的元数据）的内容，保证了整个企业中契约的一致性。

8) 完善服务属性等级矩阵（ARMS）

只扩展流程和业务目标等来识别巨大的候选服务列表及相应契约相当自然。这些服务在帮助机动性方面可以做得跟它的来源一样好，而且可能会导致服务扩增。包含服务属性（如重用性、组合性、抽象性、竞争力差异等）的矩阵和它们的相对加权平均可以用于显示、评级和完善候选清单。要想获得成功，这个等级不应该比根据分类、线路图阶段等预先确定的目标等级低。可以修改粒度，以及反复修改契约以满足需求矩阵。

9) 使用业务机动性场景仿真 (BASS) 进行测试

在实现之前,这是评估系统灵活性的非常轻松的一步。来自路线图后续阶段的业务场景和用例或是不符合当前阶段的典型业务场景需要被编撰出来。接着,通过仿真,使用包含契约的服务目录来说明这些场景,同时评估对系统的影响。评估的关键指标包括:

1. 服务重用率 (重用的服务数/场景中的服务总数)
2. 服务使用率 (重用的服务数/目录中的服务总数)
3. 服务修订率 (修订的服务数/重用的服务数)
4. 服务创建率 (新建的服务数/场景中的服务总数)
5. 服务利用率 (针对某服务,已识别的服务消费者数/场景中的服务总数)

这些 IT 指标规范了复杂度,同时具有联合评估对推向市场时间的影响的特性。这会进一步调整和影响服务列表。已经经过路线图初始阶段的企业可以在 BASS 中包含历史数据,以产生更好的业务和财务指标。

10) 拥抱变更

以服务为基础的系统,其特点是可以同时接受计划中的和计划外的变更。服务清单是一个关键的驱动力,同时提供一个跟 SOA 治理流程配合良好的、迭代的 SOA 识别过程也很关键。这个迭代会与路线图的各个阶段、持续改进项目或来自内外部触发条件(如降低服务开销的技术进步可能会允许添加新的细粒度服务)的结果对齐。这些变更可能会导致服务创建、服务修订和服务退役。

总结

组合应用由服务清单中的服务装配而来,促进了业务机动性。服务识别产生了这个业务和技术服务的列表。识别服务集合相对容易,但是,ROI 则受 SOA 项目性质、变更频率和变更大小影响。本文指出了在实现之前识别、验证和核实服务清单内容的关键最佳实践。

原文链接: <http://www.infoq.com/cn/articles/soa-service-identification>

相关内容:

- [SOA的管理策略](#)
- [回顾之回顾](#)



Java — .NET — Ruby — SOA — Agile — Architecture

Java社区：企业Java社区的**变化与创新**

.NET社区：.NET和微软的其它**企业软件开发**解决方案

Ruby社区：面向Web和企业开发的Ruby，主要关注**Ruby on Rails**

SOA社区：关于大中型企业内**面向服务架构**的一切


Agile社区：敏捷软件开发和**项目经理**

Architecture社区：设计、技术趋势及**架构师**所感兴趣的话题

新品推荐 | New Products

8.8.8.8——用于快速浏览的DNS服务器

作者 [Abel Avram](#) 译者 [崔康](#)

 谷歌提供了两台公共 DNS 服务器，分别是 8.8.8.8 和 8.8.4.4，以进一步提高浏览网页的速度。

原文链接：<http://www.infoq.com/cn/news/2009/12/Public-DNS-Google>

TorqueBox：JVM上的Rails企业级解决方案

作者 [丁雪丰](#)

TorqueBox

不久前，JBoss 的 TorqueBox 发布了基于 JRuby 1.4 的最新版本，构建于 JBoss AS 之上的 TorqueBox 为 Rails 应用程序提供了一个强大的企业级运行环境。

原文链接：<http://www.infoq.com/cn/news/2009/12/TorqueBox>

IBM WebSphere拥抱REST

作者 [Dilip Krishnan](#) 译者 [黄璜](#)

在 Connect09 分析师会议上，AIM(应用集成与中间件)的总经理 Craig Hayman 主持了题为联邦连接性——企业内外的智慧集成的会议。谈到这一会议，来自 RedMonk 的业界分析师 James Governor 表示，“上周四我说我正在努力总结 IBM 的 Connect09 分析师会议。直到现在我都在做这件事呢”。

原文链接：<http://www.infoq.com/cn/news/2009/12/ibm-websphere-rest>

GWT 2.0：新性能工具——Speed Tracer

作者 [Abel Avram](#) 译者 [张凯峰](#)



GWT 2.0 的新特性有：Speed Tracer——一个性能分析工具、开发模式、UiBinder、布局面板以及更多的 JavaScript 代码级优化。

原文链接：<http://www.infoq.com/cn/news/2009/12/Speed-Tracer-GWT-2>

一种新的可视化处理数据的语言——Vedea

作者 [Abel Avram](#) 译者 [王波](#)



Vedea，或称为来自微软研究院计算科学实验室的微软可视化语言，是一门用于创建交互的对数据进行可视化处理的语言。

原文链接：<http://www.infoq.com/cn/news/2009/12/Vedea>

ECMAScript 5 正式发布

作者 [Alex Blewitt](#) 译者 [张凯峰](#)



这周 ECMAScript 5 也即众所周知的 JavaScript 正式发布了，在给基本库带来更新的同时，还引入了更加严格的运行时模型，来帮助定位并移除通常的代码错误。

原文链接：<http://www.infoq.com/cn/news/2009/12/ecmascript5>

IntelliJ IDEA 9 : Java EE 6、OSGi、Flex及更多

作者 [Craig Wickesser](#) 译者 [张凯峰](#)



JetBrains 最近发布了他们的获奖 IDE——IntelliJ IDEA 9。它包含对一整套新技术的支持、对已有特性的改善、性能的提升以及更加现代化的用户界面。

原文链接：<http://www.infoq.com/cn/news/2009/12/intellij-idea-9>

Spring 3.0 发布：基于Java5，添加了新的表达式语言和对REST的支持

作者 [Ryan Slobjan](#) 译者 [徐会生](#)



Spring 3.0 于 12 月 16 日发布啦。InfoQ 采访了 Spring Framework 项目的技术负责人 Juergen Hoeller，向他了解这次发布的一些情况以及其对 Spring portfolio 带来的改变。

原文链接：<http://www.infoq.com/cn/news/2009/12/spring30>

Mono迈上新台阶：Mono 2.6、MonoDevelop 2.2 和Moonlight 2 发布

作者 [朱永光](#)



今年，Novell 在 Mono 平台方面动作频频。前几个月，接连发布了 2 个 Mono 相关的商业软件后，又于前几天分别发布了 Mono 2.6、MonoDevelop 2.2 和 Moonlight 2。这一系列的产品发布，预示着 Mono 已经迈上新台阶。

原文链接：<http://www.infoq.com/cn/news/2009/12/mono-new-level>

Dojo 1.4 发布：性能改进、稳定性提升

作者 [Dionysios G. Synodinos](#) 译者 [张龙](#)



近日 Dojo 团队发布了 Dojo 1.4，该版本的性能和稳定性都得到了极大的提升，同时增加了大量的新特性。

原文链接：<http://www.infoq.com/cn/news/2009/12/dojo-1.4>

封面植物

桂滇桐



濒危种。桂滇桐是 1957 年采到标本，1975 年发表的新种，只有一个分布点，1982 年到原产地没有找到，尚待进一步调查。

形态特征乔木，高 12 米，树皮褐色；小枝干时紫色，疏被淡黄褐色星状短柔毛。叶互生，厚纸质，长椭圆状披针形或长椭圆形，长 7—9 厘米，宽 2.5—4 厘米，先端长渐尖，基部锐尖或楔形，边缘有明显的小锯齿，上面几无毛，下面疏被淡黄褐色短柔毛，侧脉 6—7 对，网脉纤细而明显；叶柄长 1.8—2.5 厘米，略被短柔毛。果序生于上部叶腋，为二歧聚伞花序式，略被淡黄褐色星状短柔毛。蒴果长圆状椭圆形，由 5 个心皮形成，长 2.5—3 厘米，直径 2—2.4 厘米，顶端和基部均浑圆，疏被星状短柔毛；鲜时白色，于时淡黄褐色；成熟心皮具翅，翅薄纸质，扁平，宽 8—10 毫米，有二叉分枝的横脉；果柄长约 12 毫米，被淡褐色

短柔毛；种子每室 4 个，排成二列，椭圆形，两端尖，红褐色，长约 8 毫米。

滇桐属仅两种。其中桂滇桐为广西特产，而且局限在一个分布点上，植株也极为稀少。目前处于濒临灭绝状态。同时本属的系统位置尚未肯定。

特性分布区低平地方的年平均温 20.9℃，1 月平均温 12.0℃，7 月平均温 27.2℃，年降水量 1158.2 毫米。散生于石灰岩常绿、落叶阔叶混交林中，为偶见种。

1kg.org 多背一公斤

爱自然 | 更爱孩子





架构师 1月刊

每月 8 日出版

本期主编：霍泰稳

总编助理：刘申

编辑：李明 胡键 宋玮 郑柯 朱永光 郭晓刚

读者反馈：editors@cn.infoq.com

投稿：editors@cn.infoq.com

交流群组：

<http://groups.google.com/group/infoqchina>

商务合作：sales@cn.infoq.com 13911020445

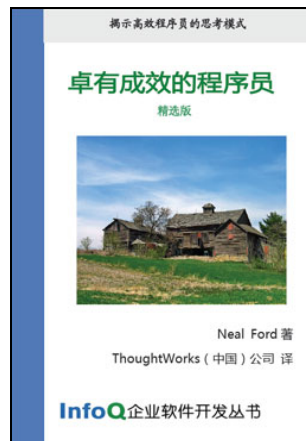
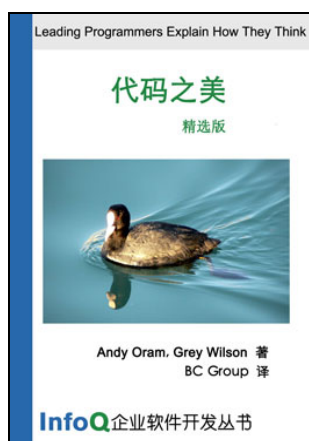
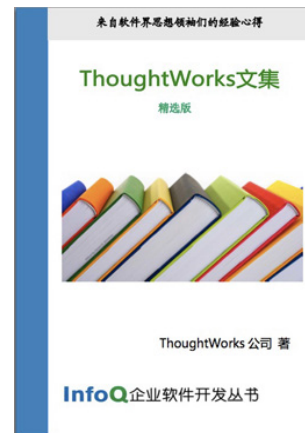
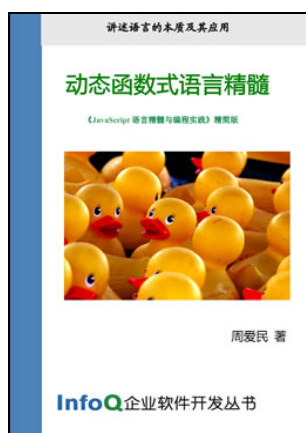
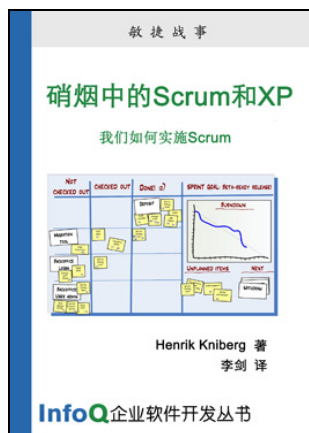


本期主编：霍泰稳，InfoQ 中文站总编辑

霍泰稳，InfoQ 中文站总编辑，有多年的软件开发经验和媒体从业经历，以技术传播为己任，关注企业软件开发领域的变化与创新。曾先后参与《程序员》杂志、《MSDN 开发精选》杂志、《开源大本营》图书和《开源技术选型手册》2008 版图书的策划编辑工作。。

InfoQ企业软件开发丛书

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