

JavaBlog 电子“迷你书”

系统架构专题



看图说话:

封面是 5 年前太太去西半球某个国度留学时留下的照片，当我回顾到这张照片时，就有冲动要拿来作本期封面的主题。

“架构”一词来自建筑学，软件架构师与建筑师的很多原则和目标是一致的。软件架构通常看做是一组原则，可以促进重用的设计方法，任何软件架构只是一个系统的草图。架构是因需而变的，所以在现实中任何软件的系统架构都是经过演变而来，不会有任何可以一步到位的软件架构设计。



本“迷你书”采用[创作共用署名 2.5 中国大陆版许可证](https://creativecommons.org/licenses/by/2.5/)授权。

历史期刊

用最简洁的页面描述企业应用与 Java 艺术!

H.E.'s Blog 电子“迷你书”

性能优化专题



2010-2 第一期

JavaBlog 电子“迷你书”

系统架构专题



2010-4 第二期

目录

目录.....	3
首篇语(Editor' s letter)	4
Memcached集群/分布式的单点故障	6
Google Analytics(谷歌分析) 架构与原理	9
WOA面向Web的架构 离我们有多远?	15
谈谈大型J2EE系统架构	18
大型网站的web服务器	20
GlassFish JMS 集群	23
Mailinator网站架构(Linux+Tomcat+Java)	27
J2EE Web 容器集群 - Nginx+Glassfish+Memcached+ServletFilter.....	30
大型网站百万级高并发测试 - MySpace云测试CloudTest™	32
对REST浅浅的认识	35
多台服务器数据实时备份.....	37
Java+PHP整合=混血 新宠儿.....	39
J2ee核心模式 - 微架构	42
哪些大型网站采用J2EE架构	44

首篇语(Editor' s letter)

冬去春来，阳光三月天，南京的春天已经能感受着丝丝暖意，Javabloger.com 也迎来第二期“迷你书”的发布，本期以系统架构话题为主，将介绍大型网站使用的 Web 服务器端，以及美国最大的 SNS 社区 MySpace 对新产品的“云测试”部署架构，一些使用 GlassFish 和 Memcached 的技巧，网站实时在线备份的经验分享 等。

回顾 2009

主要精力放在系统架构设计、敏捷开发的学习与研究，和对开源项目 MySQLOA 的开发。Javabloger.com 网站也在孵化当中，将略懂的一些知识与经验进行与大家分享，已与[金陵软件工作室](#)紧密协作，开始接洽 SEO、企业、门户网站的应用开发。

展望 2010

把握八个字：“主动、积累、分享、反思”继续前进。Javabloger.com 将会分享更多有价值的内容和经验给大家，同时还会以正式与非正式的，自己圈内圈子外的，计划性的进行专项专题技术讲解，目前虽然站点没有大量内容，但坚持内容原创的精神，因为 Javabloger.com 在遵循基本生存法则---**内容为王**。

JavaBlogger 发布“迷你书”的目的是为了能让用户得到更好的阅读体验，网站对专题的归档处理，并发布成小册子，用户可以在离线的状态阅读本站有价值的内容。

话外音：

- 结婚后太太对我关心让我倍受感动，多次在外忙于工作晚归家时，总能看见太太脸上乐呵呵的**笑容**和一份用心准备的爱心**靓汤**。
- 2010 春来，我的小圈子内大大小小项目不断，忙于商务谈判，内部沟通，SEO 手段、内容策划，等等 杂事繁多。
- 最近对 MongoDB、Tokyo Tyrant 存储类的话题特别感兴趣，对电子商务网站类型的 SEO 学习方向依然热情不减。
- 分享更多、更好的技术话题，坚持原创。

读者内容反馈：njthnet@javablogger.com

Memcached集群/分布式的单点故障

口水: Memcached 在 2009 风靡全球，现在对Memcached态度大家各自褒贬不一，话不多说进入正题。

我看到过这样一段文字 “memcached 如何处理容错的？不处理！ :) 在 memcached 节点失效的情况下，集群没有必要做任何容错处理。如果发生了节点失效，应对的措施完全取决于用户。节点失效时，下面列出几种方案供您选择：

* 忽略它！ 在失效节点被恢复或替换之前，还有很多其他节点可以应对节点失效带来的影响。

* 把失效的节点从节点列表中移除。做这个操作千万要小心！ 在默认情况下（余数式哈希算法），客户端添加或移除节点，会导致所有的缓存数据不可用！因为哈希参照的节点列表变化了，大部分 key 会因为哈希值的改变而被映射到（与原来）不同的节点上。* 启动热备节点，接管失效节点所占用的 IP。这样可以防止哈希紊乱（hashing chaos）。”

同学们，根据上面的说法，memcached 其中一个节点失效以后，memcached 本身是没有任何策略维持失效转发的，这对于大型系统是一个无法接受的事实。

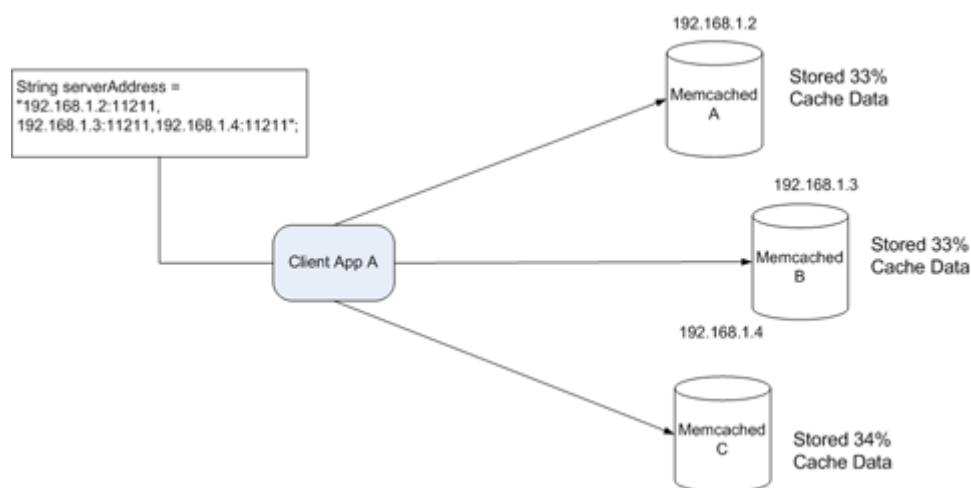
Memcached 分布式每个服务器端本身没有相互相连的关系，数据分布是由客户端来维持的，也可以说 Memcached 还没有为集群提供真的高可用方案，因为从集群的定义上来说需要满足：1.压力分载 2.失效转发。

在项目组中 lianjie.you 同学问我如果在分布式中的其中一台 Memcached 节点 down 掉了，应该如何解决？我当时愣住了，一时之间还不能给出任何完整的答案。

今早在座公车来上班的路上用手机上网 Google 了一下，发现原来在网上有很多人与我们有相同的问题，我 Google 的关键字是“Memcached 单点”、“Memcached 单点故障”。给出的搜索结果都不算让人满意，我才打算写一篇关于解决集群中 Memcached 单点故障的文章。Javabloger 只向大家提供 2 种解决思路，暂时不提供具体代码。

现象描述：

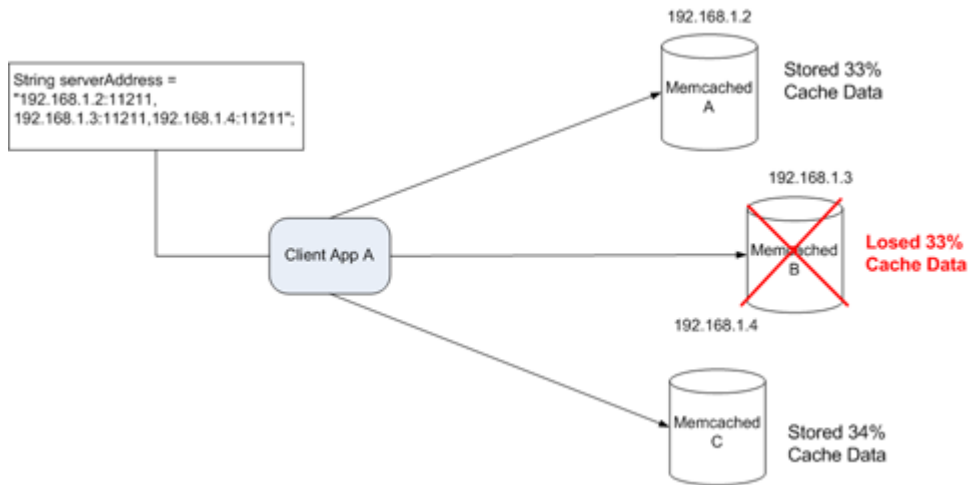
在客户端连接的部分写入多个服务器端的ip地址，客户端将会自动的把缓存数据分布的放在每个不同的机器上，如图所示：



[查看大图请点击这里](#)

现象后果：

如果其中一个缓存节点的机器down机，那么客户端存入的缓存数据将会丢失一部分，就是图中红色字体描述的“Losed 33% Cache Data”，也就是说那部分数据彻底没有了！如果是用户的关键性信息那么就玩大了，如图所示：



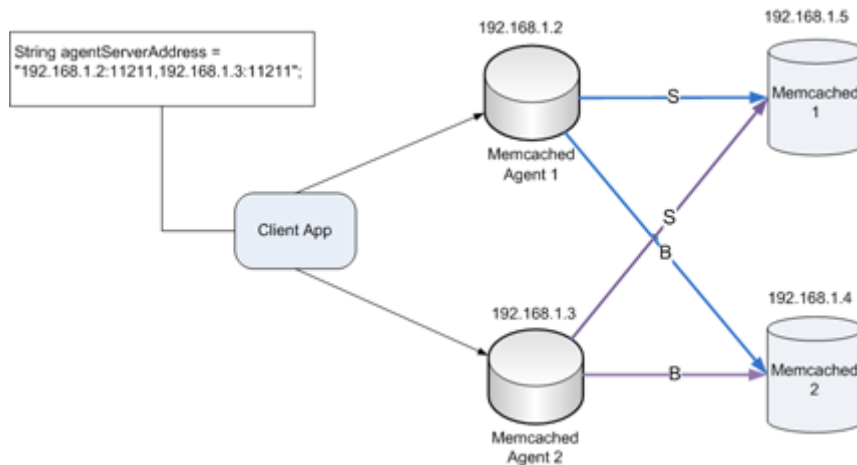
[查看大图请点击这里](#)

解决方案 1：本地备份缓存

在本地放一份缓存，同时也在分布式Memcached上放一份缓存，如果当其中一台节点当机了，客户端程序直接读取本地的缓存，本地客户端维护一个HashMap即可，这样的方案虽然很简陋，但是可以满足一部分场景的需要，当你很急需的时候可以作为临时方案暂时替代一下。

解决方案 2：采用缓存代理服务器

采用 [Magent](#) 缓存代理，防止单点现象，缓存代理也可以做备份，通过客户端连接到缓存代理服务器，缓存代理服务器连接缓存服务器，缓存代理服务器可以连接多台Memcached机器可以将每台Memcached机器进行数据同步。这样的架构比较完善了，如果其中一台缓存代理服务器down机，系统依然可以继续工作，如果其中一台Memcached机器down掉，数据不会丢失并且可以保证数据的完整性，以上描述的系统架构如图所示：



[查看大图请点击这里](#)

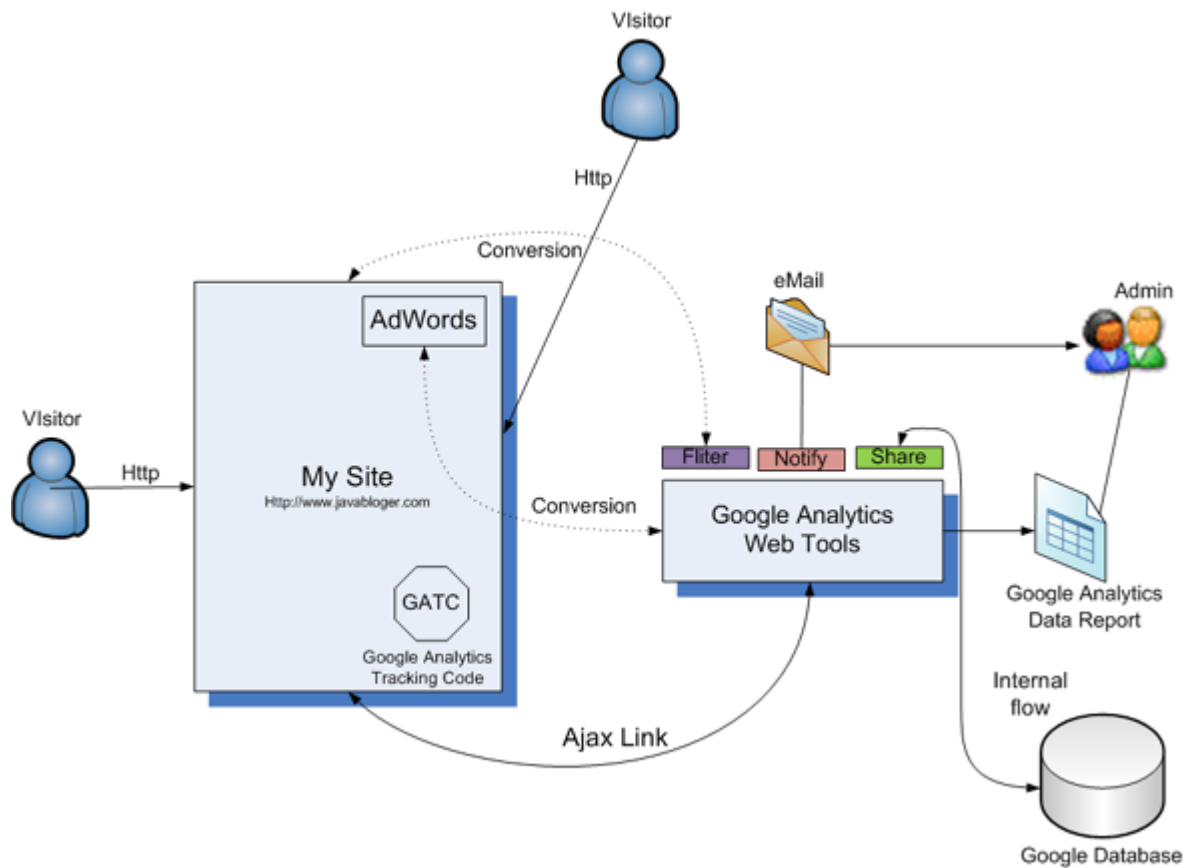
还是那句话：没有任何架构是最完美的，只是最合适的，任何架构都不可能一步到位，都是经过一步一步演变过来的。

Google Analytics(谷歌分析) 架构与原理

Google Analytics(Google 分析)是 Google 的一款免费的网站分析服务，Google Analytics 最早是由一些工程师设计的，后来被 Google 收购，现在被广受好评。对于 Google Analytics 的用户目前在国内的用户已经数以万计，本站 www.javabloger.com 也使用 Google Analytics 工具对网站的访问趋势进行分析。Google Analytics 功能非常强大，只要在网站的页面上加入一段代码，就可以提供的丰富详尽的图表式报告。

Google Analytics 的采集功能是采用 AJAX 技术来实现，还可以对 google 共享你的分析数据，并且还可以设置系统提醒功能。

整体架构如图所示：



[查看大图请点击这里](#)

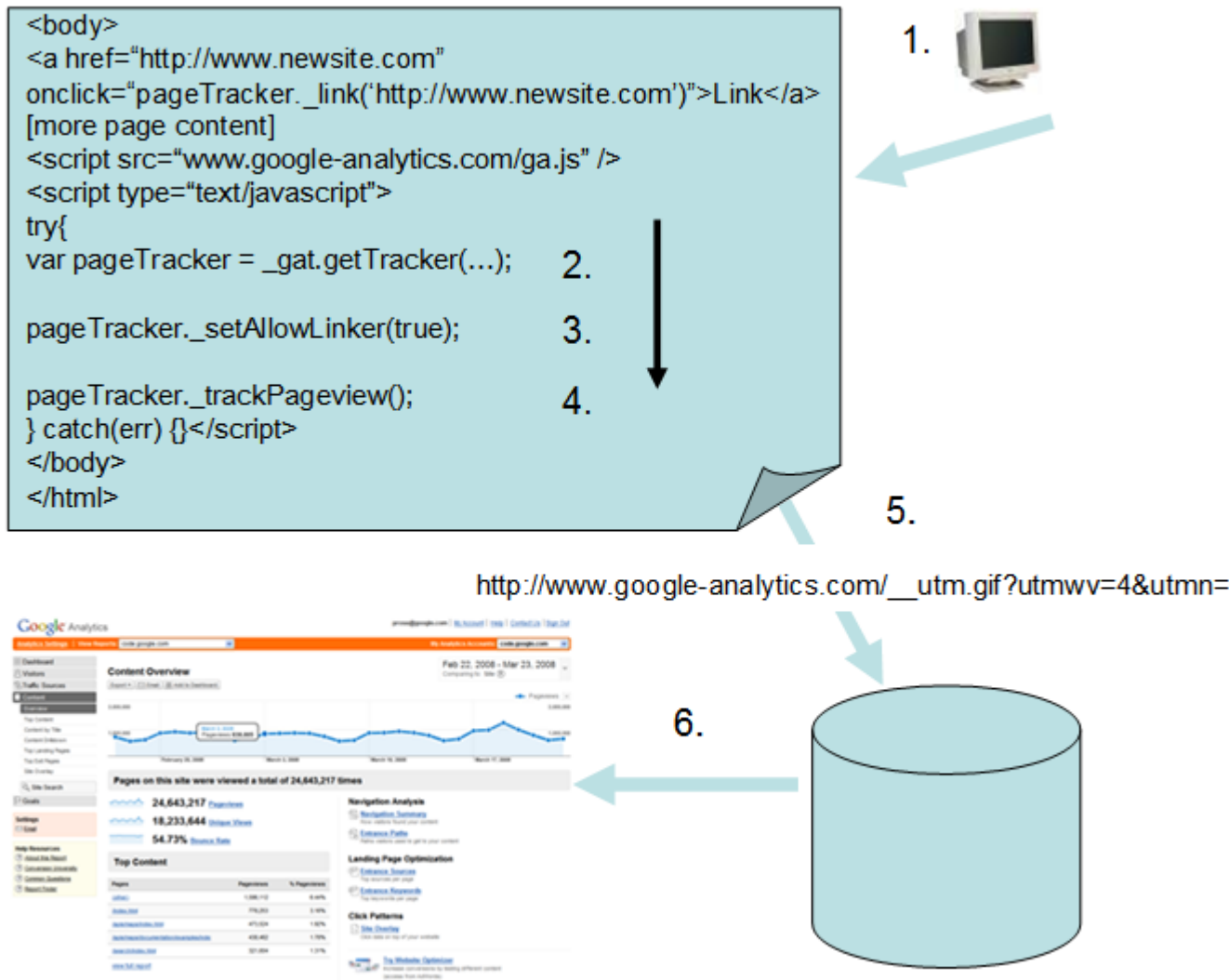
使用 Google Analytics 我们知道首先需要在你的网站中加入跟踪代码，下面我们来看看嵌入在网站中跟踪代码如何工作的。

谷歌分析跟踪代码(Google Analytics Tracking Code GATC) 检索的网页数据如下：

- 1 浏览器请求的网页包含跟踪代码。
- 2 GATC 创建并初始化一个对象的属性与网络相关的跟踪 ID 在代码中。
- 3 GATC 执行你自定义任何跟踪方法。
- 4 跟踪代码初始化和管理的以下信息：
 - *跟踪检索查看是否包括广告系列。
 - *收集从 HTTP 请求的信息到 Google GATC 中介 的各种用户信息。

5 将访问者 HTTP 请求包含 GATC 跟踪信息收集到参数名单。

如图所示：



以上详细过程还可以参考Google的官方资料：

<http://www.google.com/support/googleanalytics/bin/answer.py?hl=en&answer=55540>

和

Google分析工具的官方帮助中心：

<http://www.google.com/support/googleanalytics/>

H.E的口水:

1.不同的用户来源需要进行时间差计算，是不是就是因为这个所以在Google分析工具里面只给使用太平洋的时间。

2.老调重弹，Google对于大数据量存储和大规模并发访问的策略做的的确是很棒。再次引发了我对Google架构的探索，在此插入一个技术类型的小广告(图片)，**如图所示**google 的搜索引擎部分的架构图：

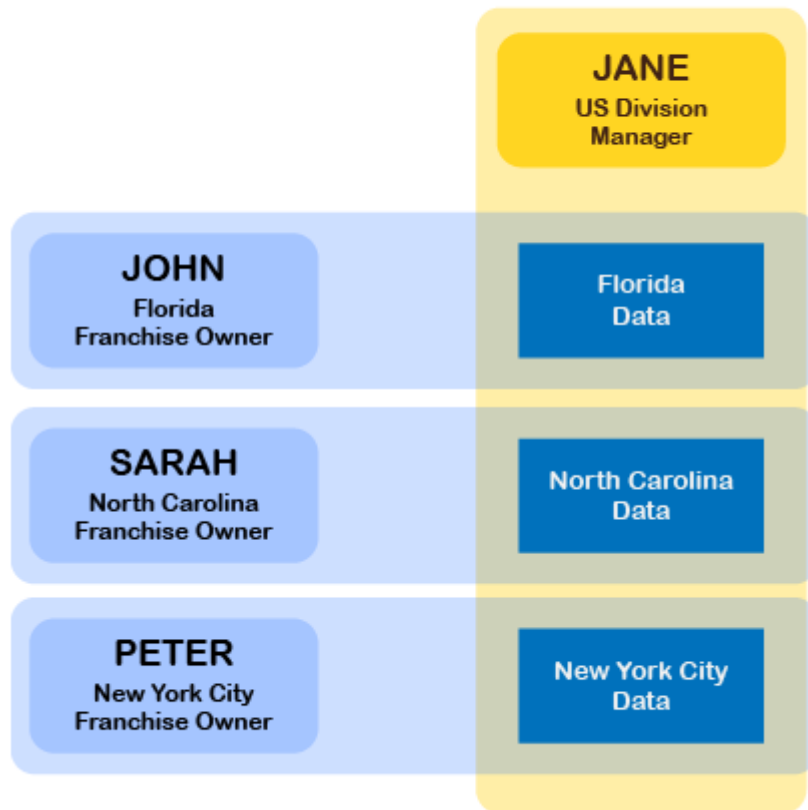
http://farm5.static.flickr.com/4039/4439963646_32420aa704_o.gif

3.从Google 分析工具的报告版面上来看，GATC的数据收集的非常详细，能做到比较全面而不占用资源，对用户透明，不容易。

当你使用 Google Analytics 分析时需要你创建 Google Analytics 的用户“网站配置文件”，创建“网站配置文件”的目的是为了方便网站多个管理员从各种对网站不同的关注角度对网站的访问进行分析，每个用户之间的分析数据也是私密性的，尽管它可能需要几分钟额外设立一个新的帐户配置文件，它通常是值得的，因为它可以帮助访问安全性和数据完整性，确保能给你更准确和有用的信息。

如图所示：

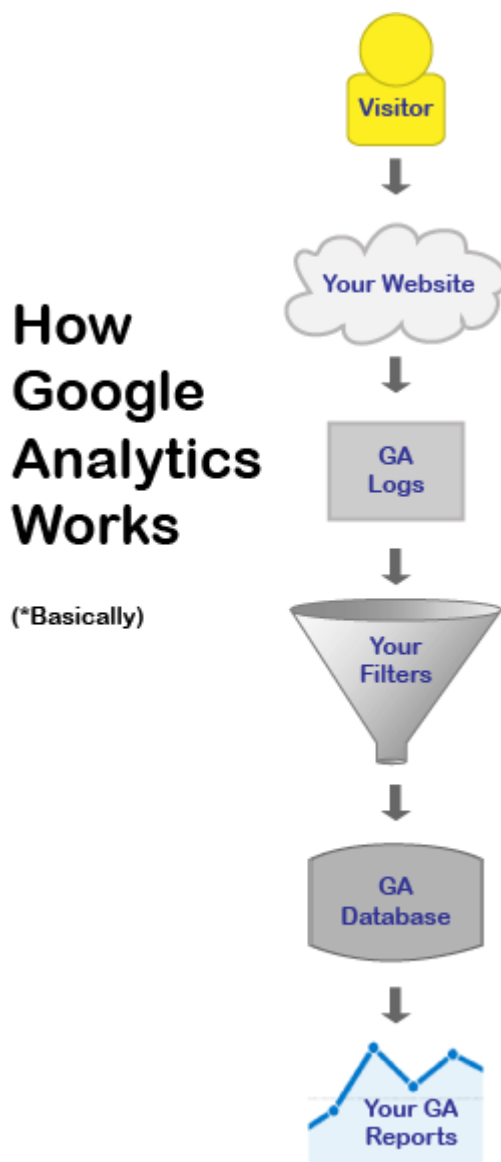
Google Analytics Profile Architecture



当你在网站内创建完成“网站配置文件”，接下来我们再来看看 访问用户、网站管理员、GA(Google Analytics) 大致是怎么样的工作过程。

- 1.访问者访问网站
- 2.触发到 GATC 代码，根据 GATC 采集用户信息，放入 Google Analytics 日志
- 3.通过你创建的过滤器对需要进行过滤的数据进行筛选。
- 4.将采用和过滤后的信息放入 Google Analytics 的数据库中
- 5.管理者通过 Google Analytics 的界面查看分析数据

整体工作流程如图所示：



根据你的不同过滤条件可以对网站的访问者统计的数据进行过滤，如图所示：

创建新过滤器

Enter Filter Information

Filter Name:

Filter Type: 预定义过滤器 Custom filter

Exclude 等于

Domain (如 mydomain.com)

Case Sensitive Yes No

Apply Filter to Website Profiles

具体使用 Google Analysis 的过滤器方法可以去问Google自己，也可以点击
这里查阅

<http://www.google.cn/search?hl=zh-CN&newwindow=1&q=Google+Analytics+%E8%BF%87%E6%BB%A4&aq=f&aqi=&aql=&oq=>

WOA面向Web的架构 离我们有多远?

近日有人聊起：“WOA 是什么？”又有人说：“WOA 纯粹噱头,如今发明这么多概念干什么？”我今天在
《Web2.0 Architectures》那本书



作者 [Dion Hinchcliff](#) 主页上看到 “Unboxing

Web-Oriented Architecture: The 6 Aspects Of An Emergent

Architectural Style” 一文，地址：

<http://hinchcliff.org/archive/2009/06/06/16901.aspx>。

文中不断的提出 WOA 的概念，还打出了一个副标题：“WOA as a complete REST architecture”，真是火上加油啊，REST 和 WOA 都是新兴概念，可惜都不是什么特别的新技术。

下面我们来看看 所谓的 WOA 架构，如图所示：

The Elements of Web-Oriented Architecture



[查看大图请点击这里](#)

现在 REST 技术风格它将是未来的 Internet 采用的主流架构，你会发现 REST 本身似乎就是关于在 Internet 上将数据资源从一处移到另一处，而不是构造一个完整的应用。“换而言之，WOA 远比 REST 更多，而 REST 是 WOA 的基本架构风格。”

Hinchcliffe 将 WOA 分为两部分来定义：核心部分包括 REST，URL，SSL 以及 XML；还有一个“WOA 完全版”包括了协议以及接口（例如，BitTorrent），身份以及安全（例如，OpenID），分发与组件（例如，Open API），以及数据格式与描述（例如，ATOM）。这些内容以六个层次组织成了 WOA 栈（包括示例用的技术）：

- Distribution – 分发(HTTP, feeds)
- Composition – 组合(Hypermedia, Mashups)

- Security – 安全(OpenID, SSL)
- Data Portability – 数据兼容性(XML, RDF)
- Data Representation – 数据表示(ATOM, JSON)
- Transfer Methods – 转移方式(REST, HTTP)

另外，WOA 与 SOA 并不存在相互替代的说法，因此，WOA 跟 SOA 不是相互取代，而是相辅相成，共同为企业服务。WOA 与目前最热门的 SOA 采用同样的设计哲学和理念，都是以服务为中心的架构模式，只是 WOA 主要采用来自 Web 的概念和技术构建服务架构。

谈谈大型J2EE系统架构

为了应对大型J2EE系统中 可靠性、可伸缩性、可用性的要求，我们会采用**集群技术、分布式技术、存储环境** 来满足大型系统的运行要求。

集群技术是由多台服务器采用松散组合方式的构成，比如：Google、LinkedIn、Digg。虽然这些系统中使用到集群的技术，有很多台计算机构建了系统的N层架构，但是对于用户来说透明的，用户 并不知道有多少台服务器给他们提供服务，用户也不知道是哪台服务器为他们的请求作出服务。

尽管在用户高并发访问量的状态下系统出现故障的现象是很低的，因为系统中的每个环节都做了**压力分载**和**失效转发**。

在大型J2EE项目中，比如：有 100 个用户同时发出访问请求，经过负载均衡策略后，10 台机器上只应对 10 个用户的请求。再比如说有 100 个人去到火车站买票，只有一个售票窗口工作效率会很低，如果开 10 个售票窗口，再通过火车站售票大厅的广播告诉每个进来买票的人，有 10 个窗口可以提供服务，那么工作效率会明显上升，这就是集群技术特点之一——**压力分载**。

在复杂的系统中环境中，上百台机器不可能每小时 每台机器都在正常运行，出现不可预先的故障是难免的，但是对于出现故障后，能否还能保证系统的正常运行，特别是运行关键性任务的机器，绝对不能出现系统瘫痪的状态，所以需要系统中可以做到失效转发。

比如：用户发出了请求，服务器端已经应答，并且准备回送请求结果，此时硬件出现了故障，不影响用户的业务操作，只是多了一个小插曲，用户需要再请求一

次， 业务流程将继续进行下去。因为本来失效down机 机器上的用户请求信息在另外一台机器上也同样存在，这就是集群技术特点之一——**失效转发**。

所谓存储环境，不仅仅是我们经常使用到的数据库系统，还有文件系统和缓存。系统中的数据有可能是文字信息，还有可能是图片和文件。

文字信息我们建议保存在数据中，对于图片和文件数据，我们不建议保存在数据库中，因为将二进制文件保存在数据库中对数据库系统的开销是非常大的。因为在数据库系统中读取文件，是将二进制数据读取在内存中再展示给用户。跟保存在数据库 blob 字段 比较 还是直接通过文件系统存放在磁盘上比较好，因为给系统带来的压力比较小。

大型 J2EE 项目中，缓存的作用主要是降低系统每个环节上的运行压力，提供运行效率。对于大型系统中我们建议您使用类似 Memcached 的可集中、可集群的缓存系统，因为在大型系统中缓存同步是非常头痛的事情。另外还需要注意缓存的使用，在整个系统中缓存无处不在，自己写的、网络设备上的，CPU 上的，硬盘上、应用服务器上，等等，妥善的使用好缓存是一个比较深入的话题，所以尽量将缓存集中管理。

参考文献:

<http://www.theserverside.com/tt/articles/article.tss?l=J2EEClustering>

<http://www.javaworld.com/jw-02-2001/jw-0223-extremescale.html>

<http://java.sun.com/developer/technicalArticles/J2EE/applications>

大型网站的web服务器

在网上闲逛的时候看到了一篇文章，

<http://blog.csdn.net/wzwfly/archive/2007/05/10/1603691.aspx> 这篇文章虽然转载的，是发表于 @ 2007 年 05 月 10 日 21:46:00 ，凭[我](#)的个人直觉 google、Alibaba.com、Baidu.com、douban.com 这些非常活跃的家伙不会原地不动的。

所以 H.E.打算用工具探测了一下，结果果然不出所料。

Google **Web服务器:** Google Web Server – GWS是根据

Apache内核进行二次开发的

Apache.org **Web服务器:** Apache/2.2.8 (Unix) 运行在 PHP/5.2.10 环境中，目前Apache上推荐下载的最新版本是 [httpd-2.2.15.tar.gz](http://httpd.apache.org/download.cgi)

Yahoo **Web服务器:** YTS/1.17.23.1 YTS是自己的开发的，参见

Yahooz在开源大会上公布的资料([Pdf 资料](#))

alex.com **Web服务器:** Apache/2.0.55 (FreeBSD) 运行在

PHP/5.1.2 环境中

Microsoft **Web服务器:** Microsoft-IIS/7.5 也算是自己研制的吧

eBay.com **Web服务器:** Apache-Coyote/1.1 也就是 Tomcat

Alibaba.com **Web服务器:** Apache/2.2.15 ，应用服务器:

JBoss-4.0.5.GA ， JK模块: mod_jk/1.2.28

Baidu.com **Web服务器:** BWS/1.0 看见这个信息从屏幕上出现我不得

不佩服我们中国人的山寨意识，同时也表示深深的无语

Sina.com.cn **Web服务器:** Apache/2.0.54 (Unix)

blog.sina.com.cn **Web服务器:** nginx/0.7.62 缓存依然使用 Squid 运行在PHP/5.2.9 环境中

163.com **Web服务器:** nginx ,不同于新浪 photo.163.com, blog.163.com, news.163.com全部采用 nginx 服务器

sohu.com **Web服务器:** SWS 再次佩服我们中国人的山寨意识, 同时也再次的表示深深的无语

douban.com **Web服务器:** nginx

Linkedin.com **Web服务器:** Apache-Coyote/1.1 也是 Tomcat

MSN.com **Web服务器:** Microsoft-IIS/6.0, ASP.Net 的版本是 2.0.50727。

MSN.jp **Web服务器:** Microsoft-IIS/6.0, ASP.Net 的版本是 2.0.50727。

MSN.tw **Web服务器:** 不看不知道, 一看吓一跳, 做域名跳转的这台机器居然是, Apache/2.2.3 (CentOS) , Web服务器还是Linux操作系统自带的, 运行在 PHP/5.2.12 上,使用 “curl -ls www.msn.tw” 试试就知道了。不过真实运行的网站 tw.msn.com, 还是Microsoft-IIS/6.0, ASP.Net 的版本是 2.0.50727 和美国、日本一样。 经过H.E.的亲自测试可见传说中微软使用Linux

的消息不假

MSN.com.cn **Web服务器:** 最前端居然是 Big-ip F5 的硬件设备, 看来 msn每个地区的系统架构还不一样。

QQ.com **Web服务器:** nginx/0.6.39 并不是最新的版本, 估计腾讯是比较早的一批nginx用户, 后端Web缓存squid/2.6.STABLE5

sf.net **Web服务器:** nginx/0.7.63 ,最前端采用 F5 BigIP 负载均衡硬

件设备

mop.com **Web服务器:** Apache/2.0.55 (Unix) 最前端采用

MemCache

hi.baidu.com(百度空间) **Web服务器:** Apache 1.1.26.0 , 不知道这个
版本的漏洞不是很多

总结:

1.中国的山寨风暴无处不在, 让人哭笑不得, google用Apache整了一个
GWS出来, 急着跟风, 一点没有创新意识。

2.nginx 服务器 开始逐渐普及

3.MemCache可以的运用的范围非常广泛

4.Web服务器依然是Apache在独揽大部分的市场份额

5.对于Java类型的门户网站 Tomcat 还是首选产品

感谢 *ubuntu.unix-center.net Linux* 虚拟机 作为本次试验的测试环境。

GlassFish JMS 集群

Story

最近项目中一直在使用 GlassFish 作为应用服务器,使用到 GlassFish 中 JMS/MDB(消息驱动 Bean)的部分, 项目对 JMS 服务的要求比较高不仅仅需要在单台服务器上运行, 还需要在集群环境运行 JMS 服务。

查阅大量资料, 网上N多讲的都是Jboss的JMS集群, 很少有简述GlassFish集群 JMS的, 无意找到了一篇印度阿三的写的文章,经过反复推敲, 证明这样的实施方案虽然是SUN公司比较推崇的, 实施起来成本比较大, 将来需要投入的维护成本也不小。印度阿三提出的做法给人感觉: 要吃块牛肉而已, 现在 牵头牛过来给我了。[请看原文](#)。

将此方法放弃, 直接使用 OpenMQ JMS 服务, OpenMQ 也是 SUN 的产品, 只不过现在和 GlassFish 整合包含在 GlassFish 里面, 也可以独立使用。在项目测试环境中运行了 OpenMQ 的集群, 没有用 GlassFish 里面提供的 MDB 解决方案, 欢快的使用着 OpenMQ JMS 集群功能, 2 台机器上的 JMS 队列也在欢快的 进行消息同步。

不久后问题来了。。。。。

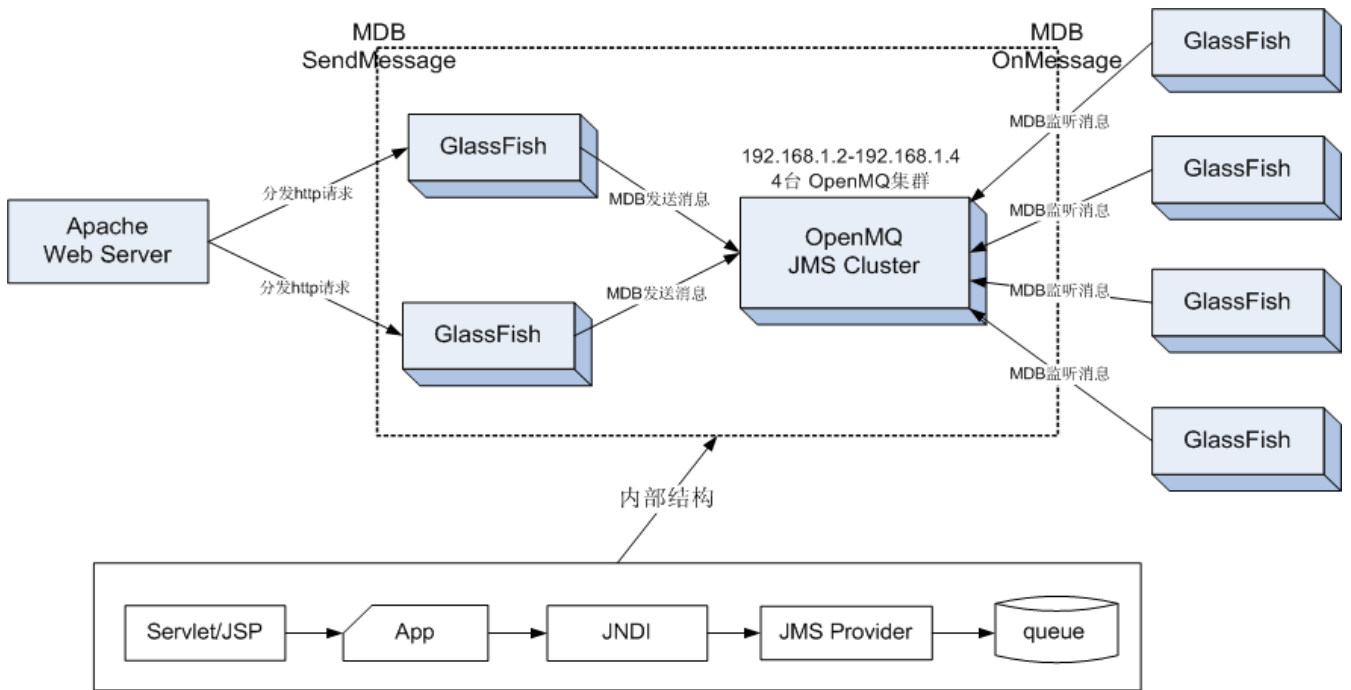
OpenMQ 和我们自己写的客户端程序建立的是 socket 连接, 也就是说服务端 OpenMQ JMS 服务一旦 down 掉, 那么 JMS 客户端程序就跟白痴一样还在等待接收消息, 却不知道 OpenMQ JMS 服务端已经 down 掉了。系统中没有体验到失效转发。项目组的一位仁兄 写了一个程序 通过轮询检测 我们自己写的

JMS 客户端程序 是否抛出异常判定 OpenMQ JMS 服务端是否还活着。嗯，这位仁兄的方法果然有效 原先我们自己写的客户端程序不是呆子了，智能了一些，知道服务端 down 掉以后该做什么。这样的方法虽然达到了效果，但是比较简陋，性能上有种说不出的痛。不知道将来会出现什么问题，还是需要依靠自己去解决，会有一定程度的风险。

How

接下来的几天，我google到一份讲述 J2EE服务器原理的书籍，通过理论上的知识让我知道，通过J2EE容器发送JMS消息，跟写通过容器中JNDI整合JDBC方式向数据库做操作一样。首先要查找数据源，建立数据库连接。JMS程序也是一样。可以通过API调用JMS服务器直接操作，也可以通过容器JNDI的方式操作。和JDBC的区别就是分为 发送者/接受者(Queue)，发送者/订阅者(Topic)。

那么换个思维方式，也就是说在 GlassFish 中配置一个 JNDI 作为一个别名，实际服务器的目标地址可以配置成本地/远程 JMS 服务器，可以是一个也可以是多个服务器。很多事情就可以交给 GlassFish 容器帮我们去做了，比如：超时重连，切换失效主机，事务等。这样可以 对项目进行重新架构，详见下图：



[查看大图请点击这里](#)

说明: 1. 客户端收/发程序 -> 部署在容器中 -> 容器JNDI -> JMS Server -> Message Queue -> Message 。

2. 发送者向JMS服务器中的消息队列发送 100 个消息，通过JMS集群接收端如果有 2 台服务器每台服务器会接收 50 个，有 4 台每台接收 25 个，以此类推，这样达到了压力分载的效果。

3. 无论在发送端还是在接收端任意一台服务器Down掉，JMS集群服务器会自动分配负载。

拓展话题: 1. JMS与EJB中MDB的关系到底是什么? 2. SUN公司推出的JMS1.0和JMS1.1 在功能上、标准上有什么区别?

口水: “没有 100%最佳的实践方案，往往最佳实践就是最折中的一种方法。”

参考资料:

<http://www.novell.com/documentation/extend52/Docs/help/MP/jms/admin/clustering.html>

参考资料: <http://www.wnetw.net/jclub/technology/read.jsp?itemid=762>

参考资料: <http://today.java.net/article/2008/01/18/jms-messaging-using-glassfish>

参考资料: <http://developers.sun.com.cn/Java/jms-messaging-using-glassfish.html>

相关文章

[大型系统中使用JMS优化技巧 – Sun OpenMQ](#)

[GlassFish 优化技巧 -GlassFish HTTP/1.1 GZIP](#)

[Glassfish\(EJB\) 与Quartz Job Scheduler整合](#)

[GlassFish 文档](#)

[GlassFish 数据库连接池的配置步骤\(图解\)](#)

Mailinator网站架构(Linux+Tomcat+Java)

采用 Javamail 和 Java 技术构建的 Mailinator 很有趣是一个免费、不需要进行注册就可以收邮件的平台，称之为“一次性电子邮件服务”。你在一个网站注册时并不想提交你的邮件地址，因为你担心将来会受到垃圾邮件的骚扰，但你此时又必须在这个网站上注册输入你的邮件地址。

此时你可以使用 Mailinator 来为你服务。但 Mailinator 他不是垃圾邮件制造器。比如：发送邮件到 javablogger@Mailinator.com，该邮箱就会自动创建。

访问 Mailinator 网站 输入你的邮箱就能进入浏览邮件了，注意：Mailinator 只是个临时邮箱，作用就是防止垃圾邮件对你的骚扰，一次只能保持 10 封邮件，不支持附件，单个邮件限 120K 以内，任何人都可以浏览，不要用来存在重要的邮件。

登陆网址：<http://mailinator.com> 经过测试对中文支持的非常友好，如图所示：



[查看大图请点击这里](#)

下面我们来看看这个替身英雄

系统平台

操作系统: Linux

应用服务: Tomcat

语言环境: Java

统计资料

2005 年 28068.0 万封

2006 年 450740000 封

2007 年将处理, 估计为 1.29 亿封。峰值速度 650 万封电子邮件/日, 4513/分钟, 75/秒。

开始是Mailinator 运行在AMD 2GHz的Athlon处理器, 1GB内存, 普通 80G IDE 硬盘。

即使是在不断的攻击和垃圾邮件高峰负荷, Mailinator无人值守运行几个月, 也很少有邮件丢失。

Mailinator 老的架构

收到邮件保存在磁盘上。

然后采用Java 去读取磁盘上邮箱中的邮件。

然后, 系统把所有电子邮件加载到内存中。

但是, 一旦内存中达到 20,000 电子邮件对机器运行就产生了很大瓶颈。

Mailinator 新的架构

采用了 DOM 和 AOP技术, 使用XML对数据进行归档整理, 运行AOP便于对将来的扁平扩展。

接收电子邮件后, 解析解析邮件, 进行DOM后并将其数据存储到内存中。尽

量减轻磁盘负载。

采用了多种过滤手段，每封邮件通过过滤后，并存储在RAM中。

比如：过多的邮件从一个IP发出进行屏蔽

超过 120k的电子邮件直接过滤，这样可节省内存。

每个收件箱仅限于 10 电子邮件。

电子邮件被写入磁盘时，系统性能已经下降，因此在轮询启动时加载信息，将 2 个工作交错开来，减少瓶颈。

系统开启 300 个线程同时工作，据说一直没有出现问题。

将近期内收到的邮件压缩在 RAM 中，需要阅读的时候才进行解压，节省内存空间。

从这一案例我们能看出，Mailinator 系统架构的优化完全是根据自己的业务规则和需求场景来进行。

并不是一味盲目的更换软件，不断添加服务器硬件，Mailinator而是采取了多种策略，让数据更灵活，让数据更靠近CPU。

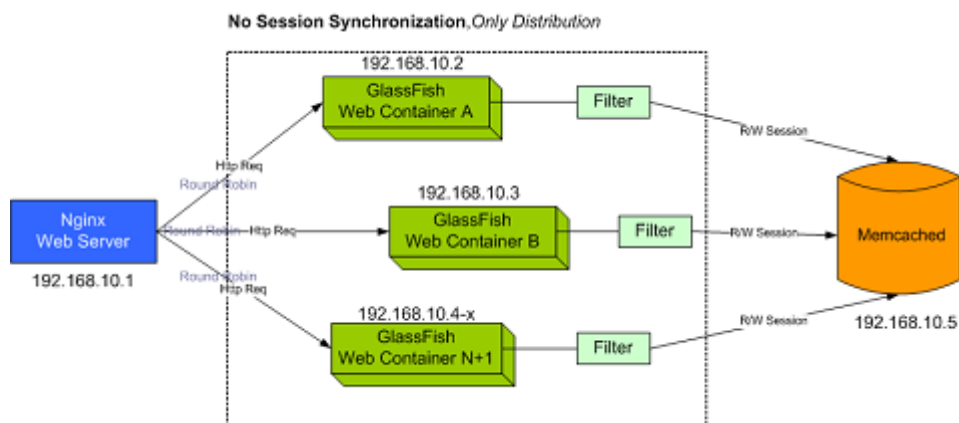
我(H.E.)相信只有了解具体的需求才能达到最大程度的优化。

J2EE Web 容器集群

- Nginx+Glassfish+Memcached+ServletFilter

这个标题可能让你看不明白，因为他来自H.E. 山寨 方法，或者说我这种方式将时下比较流行的几个东东进行了DIY整合。

先看看图，暂且告别一下文字：



[查看大图请点击这里](#)

还没有看懂没有关系，听我慢慢的口水：

系统环境：

1. 操作系统linux
2. web服务器Nginx
3. Jsp/Servlet 服务器 GlassFish，当然Tomcat/Jboss也可以。
4. session存储 Memcached
5. 自己写的Servlet Filter 读/写 session ，以上图中架构由这 5 个部分组成。

压力分载

由web服务器Nginx完成，在 Nginx服务器的 conf/nginx.conf 文件里面加入这个配置

```
#upstream backend {  
  
# server 192.168.10.2:8080 weight=1;  
  
# server 192.168.10.3:8080 weight=1;  
  
# server 192.168.10.4:8080 weight=1;  
  
#}
```

你第一次请求Nginx将去指向 192.168.10.2 ，第二次请求将去指向 192.168.10.3，第三次请求将去指向 192.168.10.4，第四次请求将去指向 192.168.10.2，第五次请求 将去指向 192.168.10.3，以此类推。 如果同时有 100 个请求，2 台机器上有 33 个请求，另外一台机器上有 34 个请求。这样简单的就实现了压力分载功能。

失效转发

通过每台机器上 Servlet Filter 组件 向中央缓存 Memcached写入，就算你的Jsp/Servlet 容器器down掉你的用户session还在 Memcached中，而最前端的 Nginx服务器会做出判断哪些机器还活着，如果其中一台机器down掉，剩下的 2 台机器继续平均分配负载。

优势：

1.比传统Apache的整合方式简单了 10 倍! 更加灵活和简单,修改 Nginx服务器中不到 30 行的配置文件就可以搞定压力分载全部功能。

2.采用 Memcached+Servlet Filter 将资源合理利用，传统的方式机器越多，需要同步的 session 会话就越多，系统负载就越大，采用这样的架构解决了一直困扰的问题。

大型网站百万级高并发测试 – MySpace 云测试 CloudTest™

2009 年 12 月 MySpace 在新西兰对用户推出了音乐和视频的服务功能，这些新功能包括能够观看音乐录像，艺术家的视频搜索，创建收藏夹列表，等等。因为 MySpace 网站在任何国家每日的访问量是巨大的，这些新功能将会带来很大用户访问量，会导致服务器的运行负载加大，为了到达更高的可用性，需要对其的新功能进行云测试，来模拟百万级的用户并发访问量。

什么是云测试？

云测试是基于云计算的一种新型测试方案。服务商提供多种平台，多种浏览器的平台，一般的用户在本地用 Selenium 把自动化测试脚本编写好，然后上传到他们网站，然后就可以在他们的平台上运行 Selenium 脚本了。

MySpace 网站的视频新的功能预计将会有 100 万的并发用户，更重要的前提条件是 100w'同时' 访问全功能。

以下是这是在测试过程中生成负载的前提条件：

- *百万虚拟用户并发
- *测试功能包括 音乐、视频搜索，评价影片，将影片新增至我的最爱，等等。
- *每秒钟数据传输 16 gbit
- *每小时超过 6TB 的数据传输
- *每秒超过 77,000 的点击次数，不包括实时交通点击
- *采用亚马逊 EC2 的 800 来生成负载

MySpace云测试环境 CloudTest™对外公布的资料

测试环境

- * 7.5 GB内存
- * 4 EC2 上计算单位
- * 850 GB的实例存储(2 × 420 GB加 10 GB的根分区)
- * CPU 64 位平台
- * Linux 操作系统 Fedora Core 8

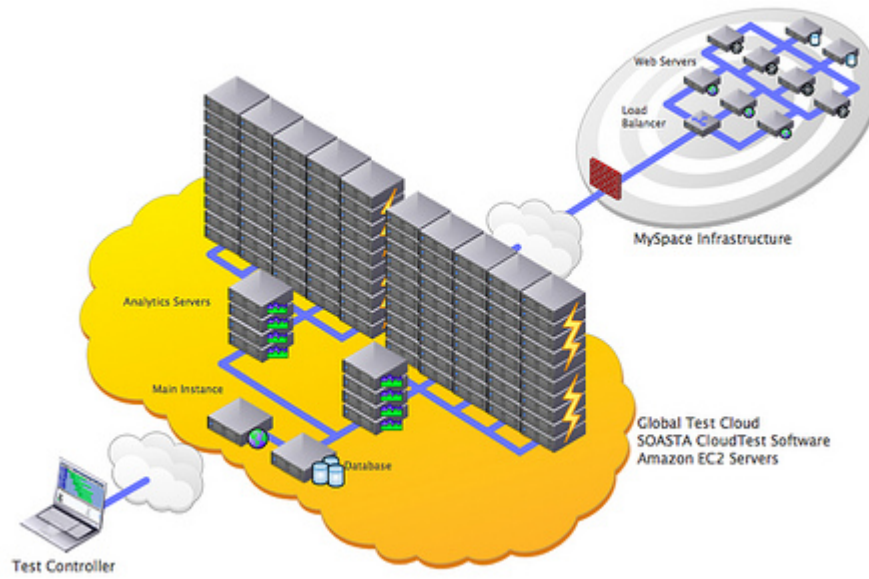
PS: 组成一个**测试云**，就是下图中有**黄色**小闪电的部分。

后端由 2 个超大的EC2 阵列 组成**测试控制器**实例，并将结果集中到数据库

- * 15 GB内存
- * 8 个EC2 上计算单元(每个EC2 单元上有 4 个虚拟计算单元，一个有 2 组)
- * 1,690 GB的实例存储(4 × 420 GB加 10 GB的根分区)
- * CPU 64 位平台
- * Linux 操作系统 Fedora Core 8
- * 开源 PostgreSQL数据库

MySpace 采用在 EC2 上建立测试云向 MySpace 新功能的数据中心发送大量的负载，并且有远程控制和监控手段来控制测试的峰值和查看测试报告，让每个相关的开发、测试、架构人员 可以针对这些测试报告做出结论和评估。

如图所示：



另外，MySpace 这次找了第三方的测试商 - SOASTA，这个公司不单只提供了功能测试，而且还有性能测试。性能测试应该是利用云计算的一个非常重要而且有意义的。

对REST浅浅的认识

REST 里面没有包含什么新的技术，还是 RPC 风格，跟 WebService 看起来差不多，但请求的 url 好像有点不同。没有带参数的问号了。看起来清爽了一些。

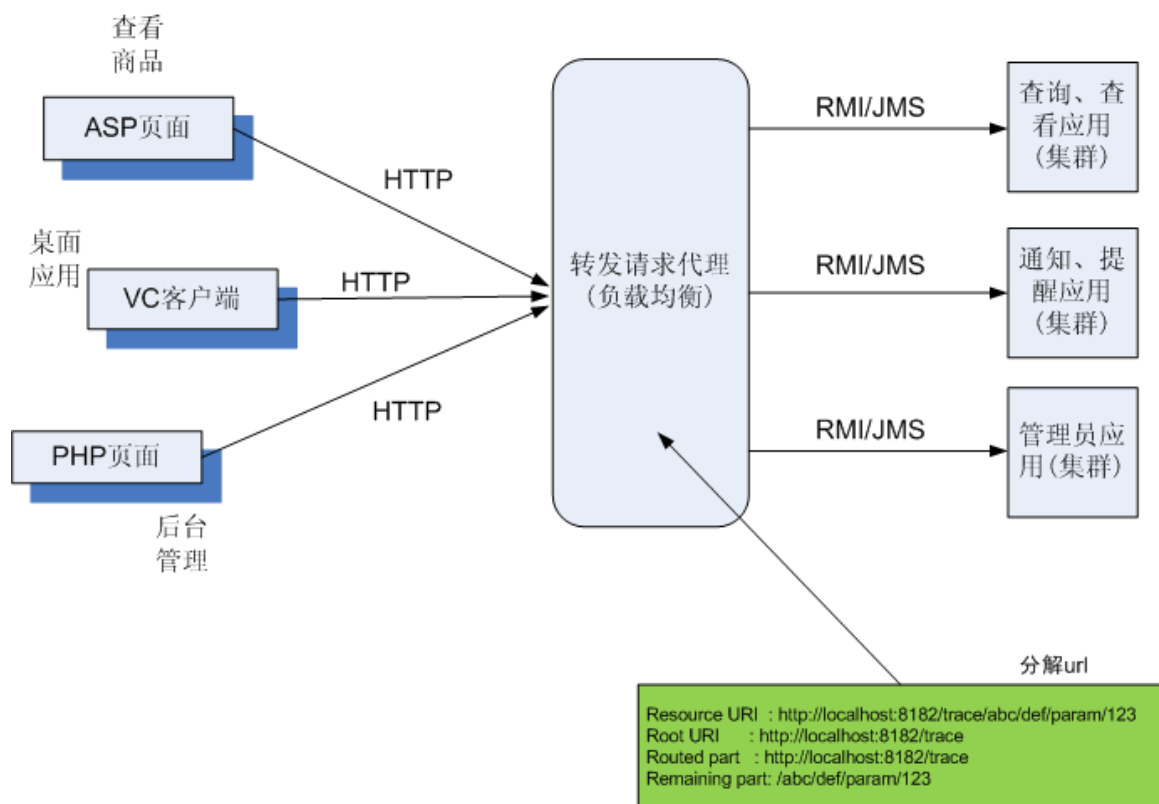
大家可以看看 www.douban.com(豆瓣网) 几乎每个 url 地址都没有问号。

通常是 <http://www.douban.com?people=someone/> 现在是

<http://www.douban.com/people/someone>。问号没了。

别小看这样的改变，对整体架构上带来了很大的改变，并且在 REST 的架构中提倡你使用缓存。

架构：



[查看大图请点击这里](#)

原理：

- 分析url的请求路径，中间通过转发，让客户端请求**直接**访问对应的各个应用层模块。

优势：

- 简单的 url 路径带来了极强的后端的伸缩性。
- 对前端的展示没有任何限制，B/S 、 C/S 架构都可以，甚至 asp,php 都可以，提高了可扩展性。
- 统一了对外部的请求访问。

劣势：

- 客户端、展示层 需要对通讯协议(JSON/XML/自定义参数)其进行解析，加大了原先的工作量。
- 中间多了一个专门针对 url 转发的工作

多台服务器数据实时备份

从商业软、硬件上采用 Symantec 的产品无可非议，例如，

Symantec Veritas Replication Exec

Symantec NetBackup

Symantec Backup Exec

Symantec Veritas Volume Replicator 完全可以满足你的任何需求。但是还是需要考虑到成本问题。

多台(3 台以上)服务器数据采用软件手段实时备份的个人见解如下：

如果从软件自主架构的角度上考虑，是非常廉价的手段，但是需要有一定技术能力，实时备份的实时是什么样子的场景，需要备份的内容是多大，都需要有一个明确的说法。

没有任何 IT 技术体系、架构能达一步到位效果和最佳实践。

没有任何 IT 技术体系、架构是不根据需求来的。

比如，你是需要对**数据库**实时备份，可以通过数据库同步的手段，mysql, msSQL, oracle 都有各种 手段，每个厂家都各自的看家本领，可以一一去 google 资料。

BTW:mysql 多台同步最好采用 6.0 以上 linux 的版本，使用集群功能。

比如，你是需要对服务器里面的 **html 页面** 文件做实时备份，可以采用 Rsync，
详见：<http://zh.wikipedia.org/wiki/Rsync>，当然前提是你的系统是
linux/unix。

比如，你是要对多台机器之间**整个磁盘**，做**分布式的数据同步**，可以采用 DRDB
或者 gluster

详见

<http://www.google.cn/search?q=glusterfs&ie=utf-8&oe=utf-8&aq=t&rls=org.mozilla:zh-CN:official&client=firefox-a>

和

<http://www.drbd.org> 前提条件也需要你的系统是非 windows 的平台，我在
邮件系统下有过这样的实践。

比如，你是需要对每台服务器里面的**缓存**做实时备份和同步的话，可以采用
Memcached 的，详见 <http://memcached.org> ，平台可以是 windows。

看你需要对什么层面做实时备份，具体针对到什么内容，涉及到现在的架构体系，
和可用性，和将来的可伸缩性。

Java+PHP整合=混血 新宠儿

在 2009 年互联网上大谈各种系统技术架构，我们看见了很多国内、国外大型网站采用了其他语言和Java(Jee)结合的方式进行工作，其中由我们熟悉的有手机之家 和 Digg 都是采用PHP和Java混合的方式进行协同工作。

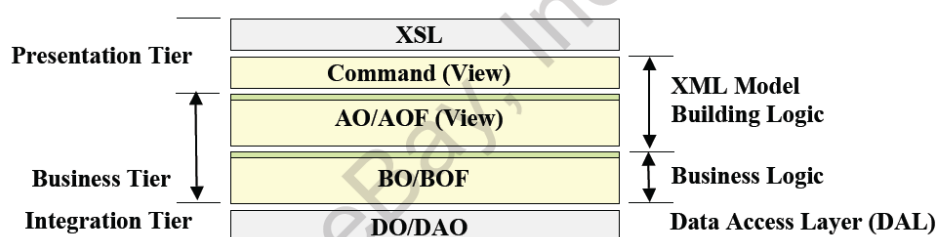
我见过最佳的实践方式也是很多人经常体验过的那就是淘宝(Taobao)，请看这个URL地

址:http://www.taobao.com/go/chn/in/thermal_underwear.php?TBG=14153.14.13&ad_id=&am_id=&cm_id=14002141798b906ee3fc&pm_id=,

你看见了什么？一个卖保暖内衣的网页？No，不完全是，是一个PHP的页面。

从我的小道消息知道淘宝(阿里公司)是中国最大的电子商务网站，并且核心业务是采用Java技术构建的，他们怎么用PHP？一点都不奇怪，他们不仅仅采用REST概念，还采用了Java+PHP的方式。虽然不能 100%知道淘宝是怎么运用PHP和Java协作的，但是我们可以通过同类型网站 eBay对外的资料看出这些大型网站是如何运用Java和其他语言共同协作的，如图所示：

- Strictly partition application into tiers
 - Presentation
 - Business
 - Integration



[查看大图请点击这里](#)

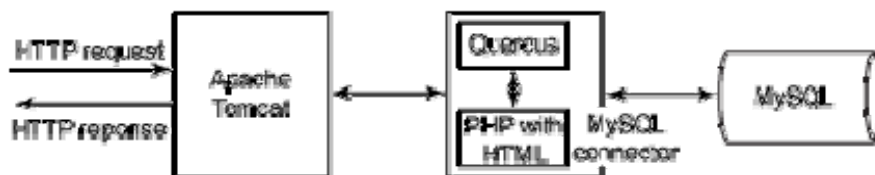
流言飞语中，我们常常听到ASP是运行起来最快的语言，遗憾的是ASP对Linux/Unix友好程度差到极点，对于Web应用开发而言，PHP和Java相比无论是部署还是开发PHP都比Java/Jsp简单，在某些方面PHP的运行效率不比Jsp差，参见一位网友写的[《JSP与PHP详细性能测试》](#)。从这篇文章的测试中我们能看出2种语言各有千秋。

在听听Php和Java程序员在说什么？

Php程序员：用直观，快速，简单的方式解决问题，注重于结果。

Java程序员：注重积累和重用，注重于过程，有时过度设计。

如果能把PHP和Java结合起来将是一件很美好的事情，Java用于后台的数据库查询、存储。而PHP作为最前端的页面展示，用户体验上应该是很不错的。所以现在有一个解决方式出来了，就是Quercus ([Refer](#))。原理如图所示：

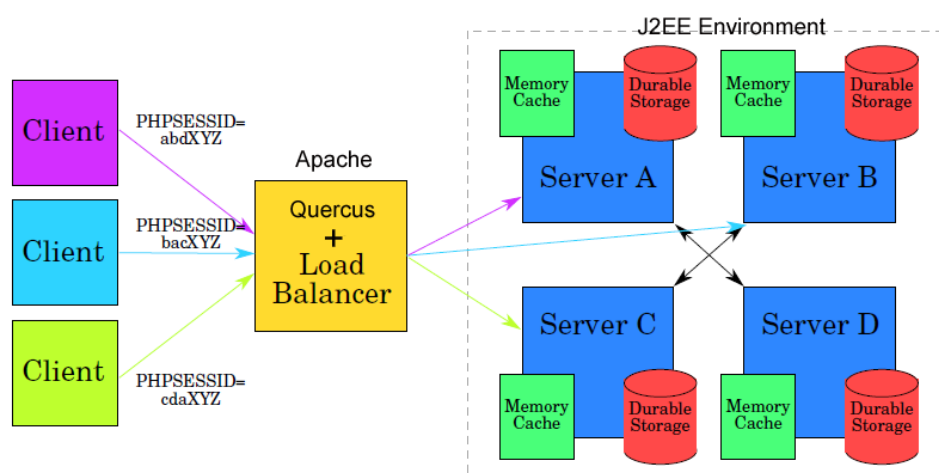


图中展示了一个页面，通过该页面，用户可以执行一些操作(增/删/改/查)，能看出HTTP请求被Apache+Tomcat接收。web.xml中的配置定义了解析php文件的QuercusServlet对象。这个对象是一个Java servlet，它提供与Quercus库的接口。在这个样例应用程序中，在php文件中实例化的一个PDO对象通过MySQL连接器实例化与MySQL数据库的连接。PDO (PHP Data Object, PHP数据对象) 以一种统一的方式提供数据库访问，包括预处理语句等高级特性。

你有兴趣的话还可以去看看 Quercus提供的一个[HelloWorld 的例子](#),这样你会更明白具体是怎么一回事。其实在实践中并不是上述说的这样，还可以采用 php+Java+SOAP或者自定义xml传输协议的方式。

Quercus 在 Java+PHP 的方案中更重要的是可以进行分布式计算，让

PHP+Java 的混合方案 在大型应用中可以更好体验他的伸缩性，如图所示：



[查看大图请点击这里](#)

希望将来在更多的项目中看见 PHP+Java 实现。

J2ee核心模式 – 微架构

介绍

Sun 公司在 Java 语言出来不久后，Sun 公司的成员们就开始面向全世界的用户收集各种案例，进行归纳，通过不断的收集和总结在几年以后将收集、归纳的结果通过技术性大会反馈给用户，首次是在 2001 年，才对外公布了收集的结果，也就是现在的 J2EE 核心设计，吸取业界的宝贵经验，让开发者们进行借鉴，目的是为了让你采用 J2EE 技术对软件设计、系统架构的实践能力。

作为一个有求知欲望的软件开发者来说光是单一的掌握 Java/J2EE API 是远远不够，所以在 J2EE 核心设计模式中，主要讲述使用：Servlet、JSP、EJB、JMS、JDBC、JNDI 这几种技术规范对系统进行构建的经验。并且强调每种模式在系统中并不是独立存在的，而是相互关联的。让开发者们合理的使用技术规范，并且可以借鉴大量的前人经验。

J2EE Core Patterns 主要讲述使用 J2EE 技术应该对系统如何进行设计，让使用者懂得如何使用 J2EE 技术得到最佳的实践手段，如何避免系统设计时的不当，并且让开发者知道如何改善已存在的系统中的设计不佳实践。

J2EE Core Patterns 是 Sun 公司对企业应用提出解决方案，不是某种技术，是一种方法，所谓的方法是借鉴前人的成功案例得出的结论。J2EE 核心模式与 Gof23 设计模式本质相同都是在业务场景中去简化设计，将复用最大化，将重构的成本最小化。

在软件的设计中 J2EE Core Patterns 只是其中的一种，看待任何一种单一的从代码的角度上去看待模式是错误的，应该从业务的需求上去看待模式。J2EE 核心设计模式的侧重点是 从系统架构的角度上看待设计模式。而 Gof23 侧重点的是从代码的角度去看待设计模式。2 者虽然角度不同，但却是本质相同。

J2EE 核心设计模式以及其他软件设计模式的特性分为以下几种：

- 来源于前人经验的分享。
- 通过结构化的记录进行积累。
- 某种模式出现的意义是为了避免重新设计，少走弯路。
- 对于相同部分的设计进行不同程度的抽象。
- 可以经过不断的完善进行重新的组合。
- 多个模式可以在同一系统中进行使用。
- 让系统设计更简单，降低复杂度。
- 让系统可以达到最大化的复用。

目的

借鉴 J2EE 核心设计模式可以对普片的问题提供通用的解决办法，在某一个领域模式经过实践证明是解决让你少走弯路的一种手段。如果你已经将 J2EE 核心设计模式作为你项目设计的范本，因为使用了规范化的设计，在系统中使用 J2EE 核心设计模式的好处大致可以归纳为以下几点：

- 权衡项目中已经证实的解决方案。
- 成员之间的更加容易沟通达成共识。
- 对系统的设计起到一定程度的规范性。

场景

任何设计模式，包括 Sun 提出的 J2EE 核心设计模式，还是 4 人帮提出的 gof23 的设计模式只能作为开发中的一种手段和工具，还需要开发者自己对系统、对需求、对业务场景有详细的了解，所以在使用 J2EE 核心设计模式之前有些前提条件，例如：

- 确定当前的应用场景。

- 确定运行当前的业务场景的前提条件。
- 界定系统中每个用户角色的权限范围。
- 与上层模块，上层系统的关联性关系。

另外，不管在任何情况下使用 J2EE 核心设计模式有一个规则，这个规则由三部分组成，分为： 特定环境、特定系统作用，以及特定软件配置之间的关系。

哪些大型网站采用J2EE架构

国外基于J2EE的平台有：

eBay、GMail、Amazon、hi5.com 和 Google AdWords

国内基于J2EE的平台有：

猫扑、网易邮箱、中国移动门户网站、淘宝、支付宝 和 易趣 、ChinaRen 、
搜狐/新浪/网易的部分频道 都是采用J2EE技术构建。

但是从下面 2 个角度上来看Java J2EE类型的网站不得不承认，需要付出的成本将高于PHP、Perl类型的网站。

从技术的角度看：

- * 用Php / Perl 开发快且简单，而J2EE相对复杂
- * 作为Web网站开发LAMP稳定且流行，而J2EE条条框框规范太多。
- * Php / Perl 对Web系统来说几乎没有要求，J2EE需要应用服务器。
- * 从短期看，轻量级Web语言（Php/Perl）应对变化更加灵活（当然从长期看任何改变的花销都非常高）。
- * 运行J2EE的机器配置性能相对要比Php / Perl高。

从财务的角度看：

- * Perl / Php开发成本比J2EE 便宜
- * Perl / Php学习曲线比J2EE上市时间更短
- * Perl / Php托管服务器要比J2EE 便宜

无论从技术使用者还是财务的角度来看J2EE，相比其他架构而言J2EE的门槛相对要高，需要开发者掌握的技能 and 开发环境的条件要高于现在流行的脚本语言架构。

但是从另外 2 个角度上来看 J2EE：

1.纯粹的商业、企业应用，例如：银行支付交易、电信计费。**J2EE有着不可动摇的地位。**

2.金融公司有大量预算并倾向于扩展硬件，而Web公司倾向于扩展软件。

我们希望能看到更多的最佳实践方法，如果能将PHP和Java进行完美的结合相互发挥自己的优势所在将是一件令人愉快的事情。请参见以前提到过的文章

“Java+PHP=混血新宠儿” 地

址：http://www.javabloger.com/article/java_and_php_mixed.html