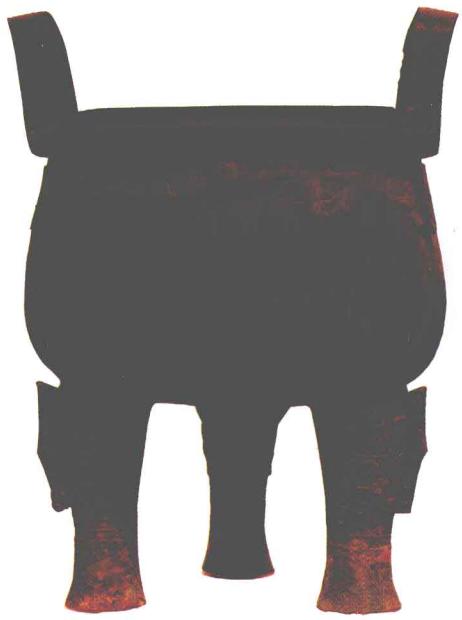
ChinaUnix.net

ITPUB www.itpub.net

资深系统运维师多年经验总结
众多系统架构和运维专家鼎力推荐
ChinaUnix、ITPub等知名社区高度推荐

高性能网站构建实战

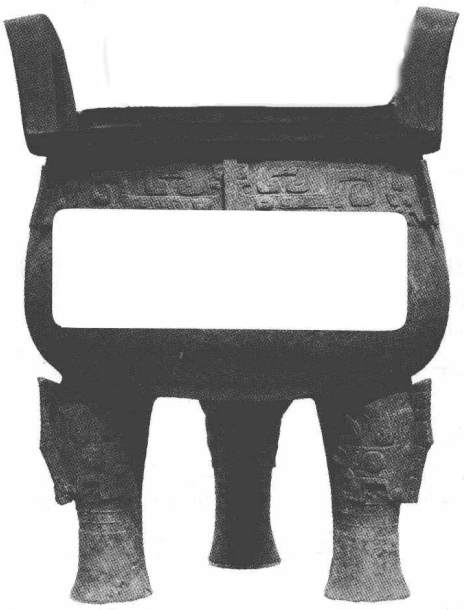
刘鑫 ◎ 著



人民邮电出版社
POSTS & TELECOM PRESS

高性能网站构建实战

刘鑫◎ 著



人民邮电出版社
北京

图书在版编目（CIP）数据

高性能网站构建实战 / 刘鑫著. -- 北京 : 人民邮电出版社, 2013.1
ISBN 978-7-115-29478-4

Ⅰ. ①高… Ⅱ. ①刘… Ⅲ. ①网站－设计 Ⅳ. ①TP393.092

中国版本图书馆CIP数据核字(2012)第223871号

高性能网站构建实战

◆ 著　　刘　鑫
责任编辑　陈冀康
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京天宇星印刷厂印刷
◆ 开本：787×1092　1/16
印张：23.25
字数：436千字　　2013年1月第1版
印数：1-4 000册　　2013年1月北京第1次印刷

ISBN 978-7-115-29478-4

定价：59.00元

读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154
广告经营许可证：京崇工商广字第0021号

目　录

第一篇　架构规划篇

第二篇　负载应用篇

第三篇 页面缓存篇

第四篇 Web 服务器篇

第五篇 数据缓存篇

第六篇　文件服务篇

第七篇 监控应用篇

第一篇
架构规划篇

第 1 章　网站架构简介

第 1 章
网站架构简介

在目前这样一个高速的信息时代，我们更希望在最短的时间内以最简单的方式获取自己需要的内容。因此，我们需要一个高性能、高可用性的网站架构来支撑网站大量的访问。所以，网站的架构在网站运营当中所占的分量是相当之重的，那么如何构建稳定而又可平滑扩展的网站结构呢？我们先来了解什么是网站架构。

网站架构通过对用户需求进行分析、了解并定位网站的目标用户，从而对网站整体架构进行规划、设计，以最大限度地进行高效资源分配与管理的设计。网站架构粗略分为硬架构和软架构。

1.1 网站的硬架构

1.1.1 机房的选择

在选择机房的时候，根据用户的地域分布，可以选择双线或多线机房，但更多时候，可能多线机房才是最合适的。越大的城市，机房价格越贵，从成本的角度看可以在一些中小城市托管服务器，比如说北京的公司可以考虑把服务器托管在天津、廊坊等地，不是特别远，但是价格会便宜很多。

1.1.2 带宽的大小

通常我们在架构网站的时候，会设定一些目标，比如网站每天要能承受千万 PV 的访问量，这时我们要估算一下大概需要多大的带宽。计算带宽大小主要有两个指标（峰值流量和页面大小），我们先做出必要的假设：

1．峰值流量是平均流量的 3 倍；

2．每次访问平均的页面大小是 100kB 左右。

如果 1000 万 PV 的访问量在一天内平均分布，每秒大约 120 次访问，如果按平均每次访问页面的大小是 100kB 字节计算，120 次访问总计大约就是 12000kB。字节的单位是 Byte，

而带宽的单位是 bit，它们之间的关系是 1Byte = 8bit，所以 12000k Byte 大致就相当于 96000k bit，也就是 90Mbps 的样子。实际上，我们的网站必须能在峰值流量时保持正常运行状态，所以按照假设的峰值流量计算，真实带宽的需求应该在 270Mbps 左右。

当然，这个结论是根据前面提到的两点假设得出的，具体值则需要根据公司实际情况来计算。

1.1.3　服务器的划分

一般情况下网站需要的服务器包括图片服务器、页面服务器、数据库服务器、应用服务器、日志服务器等。

对于访问量大的网站，图片服务器和页面服务器的分离是相当必要的。我们可以在图片服务器上运行 Lighttpd，在页面服务器上运行 Ngnix，当然也可以选择别的。我们也可以扩展成多台图片服务器和多台页面服务器同时运行，并设置相关域名，如 imgs.domain.com 和 www.domain.com，页面里的图片路径都使用绝对路径，如，然后配置 DNS 轮循，达到最初级的负载均衡。服务器多了就不可避免地涉及同步的问题，这时可以使用 Rsync 软件来完成。

数据库服务器是重中之重，因为网站的瓶颈问题大多出在数据库身上。现在一般的中小网站多使用 MySQL 数据库。一般而言，使用 MySQL 数据库的时候，我们应该配置为一个主从（一主多从）结构，主数据库服务器使用 InnoDB 表结构，从数据服务器使用 MyiSAM 表结构。这样充分发挥它们各自的优势，而且这样的主从结构分离了读写操作，降低了读操作的压力。我们还可以设定一个专门的从服务器作为备份服务器，有时候还需要借助 Memcached 之类的第三方软件，以便适应更大访问量的要求。MySQL 在后面的章节有具体介绍。

如果有条件，可以应用独立的日志服务器。一般网站的做法是把页面服务器和日志服务器合二为一，在凌晨访问量不大的时候计划任务运行前一天的日志计算。不过对于百万级访问量而言，即使按天归档，也会消耗很多时间和服务器资源来计算。所以分离单独的日志服务器还是有好处的，这样不会影响正式服务器的工作状态。

1.2　网站的软架构

1.2.1　框架的选择

现在的框架有很多选择，比如 PHP 的 Symfony、Zend Framework 等，至于应该使用哪种并没有唯一的答案，这要根据业务以及团队成员对各个框架的了解程度而定。很多时候，即

使没有使用框架，仍然可以写出好的程序来，据说 Flickr 就是用 Pear 和 Smarty 这样的类库写出来的。所以，是否使用框架，使用什么样的框架，这都不是最重要的，重要的是我们的编程思想里要有框架的意识。

1.2.2 逻辑的分层

网站达到一定的规模之后，前期代码逻辑设计里的不足便会给维护和扩展带来巨大的障碍，但我们的解决方式其实很简单，那就是重构，将逻辑进行分层。通常，自上而下可以分为表现层、应用层、领域层和持久层。

表现层

表现层的表现形式不应该仅仅是模板，它的范围还可以更广一些，所有和表现有关的逻辑都应该纳入表现层的范畴。比如说某处的字体要显示为蓝色、某处的开头要有空格，这些都属于表现层应解决的问题。通常，我们容易犯的错误就是把本属于表现层的逻辑放到了其他层去完成。举一个比较常见的例子：通常在列表页显示文章标题的时候，都会设定标题允许的最多字数，一旦标题长度超过了这个限制，就会被截断，并在后面显示“...”，这就是最典型的表现层逻辑。但实际上，有很多程序员都是在非表现层代码中完成数据的获取和截断，然后赋值给表现层模板。这样的代码最明显的缺点就是，同一段数据，在一个页面可能要显示前 5 个字，在另一个页面可能要显示前 10 个字，而一旦在程序中固化了这个数值，这就丧失了灵活性。正确的做法是用视图程序来专门处理此类逻辑。

应用层

应用层的主要作用是定义用户可以做什么，并把操作结果返回给表现层。至于如何做，这就不属于其职责范围（而是领域层的职责范围），应用层会通过委派把工作实现的具体方法交给领域层处理。

领域层

最直接的解释就是包含领域逻辑的层，是一个软件的灵魂所在。首先，我们来看看什么叫领域逻辑。简单来说，具有明确的领域概念的逻辑就是领域逻辑，比如在 ATM 机上取钱，过程大致是这样的：插入银行卡——输入密码——输入取款金额——确定——拿钱——打印凭证——ATM 吐出一张交易凭条。银行卡在 ATM 机中完成钱从账户划拨的过程就是一个领域逻辑，因为取钱在银行中是一个明确的领域概念，而 ATM 机吐出一张交易凭条的过程则不是领域逻辑，而仅是一个应用逻辑，因为吐出交易凭条并不是银行中一个明确的领域概念，只是一种技术手段。对应的，我们取钱后不要求打印交易凭条，而只要求发送一条提醒短信

也是可能的。如果要求取款后必须吐出交易凭条，则吐出交易凭条的过程已经和取款过程紧密结合，那么就可以把吐出交易凭条的过程看作是领域逻辑的一部分，一切都以问题的具体情况而定。

持久层

持久层用于把领域模型保存到数据库中。因为程序代码是面向对象风格的，而数据库一般是关系型的数据库，所以需要把领域模型碾平，才能保存到数据库中。可以使用的方法有行数据入口（Row Data Gateway）或者表数据入口（Table Data Gateway），或者把领域层和持久层合二为一变成活动记录（Active Record）的方式。

1.3　网站架构需要考虑的几个问题

1.3.1　HTML 静态化

众所周知，效率最高、消耗最小的就是纯静态化的 HTML 页面，所以我们尽可能使网站架构上的页面采用静态页面来实现，最简单的方法往往也是最有效的。但是对于内容大量并且频繁更新的网站，我们无法全部手动去逐一实现，于是出现了常见的信息发布系统（CMS），像各个门户站点的新闻频道等，都是通过信息发布系统来管理和实现的。信息发布系统可以实现最简单的信息录入并自动生成静态页面，还具备频道管理、权限管理、自动抓取等功能。对于一个大型网站来说，一套高效、可管理的 CMS 是必不可少的。

除了门户和信息发布类型的网站，对于交互性要求很高的社区类型网站来说，尽可能地静态化也是提高性能的必要手段。将社区内的帖子、文章进行实时的静态化，有更新的时候再重新静态化都是大量使用的策略，像 MOP 的大杂烩、网易社区等就是如此。

同时，HTML 静态化也是某些缓存策略使用的手段，对于系统中频繁使用数据库查询但内容更新很小的应用，可以考虑使用 HTML 静态化来实现，比如论坛的公用设置信息。目前，主流论坛都可以对这些信息进行后台管理并将其存储在数据库中，这些信息其实大量被前台程序调用，但是更新频率很小。可以考虑将这部分内容在后台更新时进行静态化，这样就避免了大量的数据库访问请求。

1.3.2　图片服务器分离

对于 Web 服务器来说，不管是 Apache、IIS 还是其他 Web 服务器，图片是最消耗资源的，

所以有必要将图片与页面进行分离，这是大型网站都会采用的策略，他们都有独立的一台或图片服务器。这样的架构可以降低提供页面访问请求的服务器系统的压力，并且可以保证系统不会因为图片问题而崩溃。在应用服务器和图片服务器上，可以进行不同的优化配置，比如 Apache 在配置 ContentType 的时候可以只支持必要的类型，配置 LoadModule 的时候只加载必要的模块，保证更高的系统消耗和执行效率。

1.3.3 数据库集群和库表散列

大型网站都有很多复杂的应用程序，这些应用程序必须使用数据库，那么在面对大量访问的时候，数据库的瓶颈很快就显现出来，这时一台数据库无法满足应用要求，所以需要使用数据库集群或者库表散列。在数据库集群方面，很多数据库都有自己的解决方案，Oracle、Sybase 等都有很好的方案，常用的 MySQL 数据库提供的 Master/Slave 也是类似的方案，业务上使用了什么样的 DB，就参考使用相应的解决方案。

由于数据库集群在架构、成本、扩张性方面都受到所采用 DB 类型的限制，于是需要从应用程序的角度来考虑改善系统架构。库表散列是常用并且最有效的解决方案。在应用程序中，安装业务和应用或者功能模块将数据库进行分离，不同的模块对应不同的数据库或者表，再按照一定的策略对某个页面或者功能进行更小的数据库散列，比如按照用户 ID 进行对用户表散列，这样不仅能够低成本地提升系统的性能而且使系统具有很好的扩展性。一些大型论坛就是采用了这样的架构，将论坛的用户、设置、帖子等信息进行数据库分离，然后对帖子、用户按照板块和 ID 进行数据库和表散列，最终只需在配置文件中进行简单的配置便能让系统随时增加一台低成本的数据库进来补充系统性能。

1.3.4 缓存

相信程序开发人员都了解，很多地方要用到缓存。网站架构和网站开发中的缓存也是非常重要。这里先讲述最基本的两种缓存，高级和分布式的缓存将在后面进行讲述。

架构方面的缓存：对 Apache 比较熟悉的人都知道，Apache 拥有自己的缓存模块，也可以使用外加的 Squid 进行缓存，这两种方式均可以有效提高 Apache 对访问的响应能力。

网站程序开发方面的缓存：Linux 上提供的 MemoryCache 是常用的缓存接口，可以在 Web 开发中使用，比如用 Java 开发的时候就可以调用 MemoryCache 对一些数据进行缓存和通信共享，一些大型社区便使用了这样的架构。另外，在进行 WEB 开发的时候，各种语言基本都有自己的缓存模块和方法，如 PHP 有 Pear 的 Cache 模块。