在云端 云计算最佳实践

决战Nginx

高性能Web服务器详解与运维 系统卷 GAOXINGNENG WEB FUWUQI

高性能Web服务器详解与运维

- 实例来自实际项目
- proxy_cache proxy_store (emcached Varnish和NCACHE)
- 广大缓存详解与应用
- 多个模块精解
- Nginx服务器高可用实现
- 用PCRE实现强大的正则匹配
- Nginx实现灵活的域名配置

陶利军 编著

清華大學出版社

在云端
云计算最佳实践

决战Nginx

高性能Web服务器详解与运维 系统卷
GAOXINGNENG WEB FUWUQI

高性能Web服务器详解与运维

陶利军 编著

清華大學出版社
北京

内 容 简 介

在这个点击率就是生命的时代，高可用是不可少的。本书完整讲述了 Nginx 服务器的各种技术细节以及安装、部署、运维等方面的内容。

本书第一部分首先讲述了 Nginx 服务器的功能、模块管理和进程管理，然后讲述 Nginx 如何处理请求，在这个基础之上再认识 Nginx 提供的服务器的名字，Nginx 服务器最大的焦点在于高并发和反向代理，在不多却足够使用的模块下实现了更多的功能。

在第二部分中，通过具体使用实例讲述了 Nginx 的模块（包括官方模块和第三方模块），并详细介绍了充分使用 Nginx 的方式方法。同时在这里使用了 Heartbeat 服务实现 Nginx 服务器的高可用。

本书的最后一部分是关于 Nginx 使用缓存技术的方法，共列举了 Nginx 使用的五大缓存，特别是广泛使用的代理缓存、Memcached 和 Varnish，另外对于 Memcached 服务器的使用贯穿了整套书。在本书中着重讲述了它的协议、原理和使用，而在本书姊妹篇中则通过不同语言的客户端对 Memcached 服务器实现具体使用。

本书适用于广大的 Linux 爱好者、具有一定 Linux 基础的系统管理员、Linux 下的 Web 服务器管理员、Linux 服务器下动态语言开发人员、Nginx 服务器管理员、培训中心师生、运维人员以及一切应该了解和使用 Nginx 的用户。

图书在版编目（CIP）数据

决战 Nginx 系统卷：高性能 Web 服务器详解与运维/陶利军编著.—北京：清华大学出版社，2012.6

ISBN 978-7-302-28784-1

Ⅰ.①决… Ⅱ.①陶… Ⅲ.①Web 服务器 Ⅳ.①TP393.09

中国版本图书馆 CIP 数据核字（2012）第 084716 号

责任编辑：栾大成
封面设计：杨如林
责任校对：徐俊伟
责任印制：王静怡

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：188mm×260mm　**印　张**：42.75　**插　页**：1　**字　数**：1153 千字
版　　次：2012 年 6 月第 1 版　　**印　次**：2012 年 6 月第1 次印刷
印　　数：1～5000
定　　价：79.00 元

产品编号：047048-01

前　言

Nginx 服务器在互联网推波助澜的作用下脱颖而出，创下了高并发的记录，因此，在短短 10 年的发展中，在全世界前 100 万的网站中，已经有 5.1%的网站使用了 Nginx 服务器，使得 Nginx 成为继 Apache（70.2%）和 IIS（20.5%）之后的第三大 Web 服务器软件，而且它的使用数量与日俱增，直逼 Apache 的市场。有人说，Nginx 将会取代 Apache 的市场，我们且不关心 Nginx 能不能取代 Apache， 1996 年 4 月以来，Apache 一直是 Internet 上最流行的 HTTP 伺服器，而且在 Linux 系统下 Apache 几乎是不二的选择，而在 2002 年诞生的 Nginx 服务器在这种形式下能够崛起那绝对是个奇迹，它有两个方面能够打败 Apache 服务器，一是高并发，二是节省资源，即轻量级。在我们现有的应用中，基本上将 Apache 跑在了后台，而是同 Nginx 服务器代理访问了。

使用 Nginx 服务器就是使用它的两大特点，一是高并发访问，二是代理，它能够快速地解析静态文件，而对于动态语言实现的动态程序则传递到后台的服务，实现了动静网页解析的分离。

Nginx 是一个自由的、开源的、高性能的 HTTP 服务器和反向代理，同时也是一个 IMAP/POP3 的代理服务器，它是由 Igor Sysoev 于 2002 年开发，并且在 2004 年发布了第一个版本，在互联网上使用 Nginx 的主机近乎 6.55% （13.5M）。

Nginx 之所以能够脱颖而出闻名世界，是因为它的高性能、稳定性、丰富的功能设置、简单的配置和资源消耗低。

Nginx 解决了服务器的 C10K 问题，它的设计不像传统的服务器使用线程处理请求，取而代之的是使用了一个更加高级的机制——事件驱动机制，而且是异步事件驱动结构。

即使你不希望处理成千上万的并发请求，你同样能够从 Nginx 的高性能和低消耗内存（占用内存小）的结构中获益。Nginx 的使用规模很全面：从很小的 VPS 到服务器集群都可以。

Nginx 强有力地用在了一些高知名度的站点，例如 WordPress、Hulu、Github、Ohloh、SourceForge 和 TorrentReactor。

本套书所包括的内容

本套书共包括卷 1 和卷 2 两本书，共 10 个部分（包括了 86 章的内容，其中第十部分是一个独立部分，没有章节），其中卷 1 包括了前 3 部分的内容，而卷 2 包括了后面的 7 部分内容。

卷 1 内容

第 1 部分　Nginx 服务器

第 2 部分　Nginx 服务器的功能模块

第 3 部分　Nginx 与缓存

第 1 部分　Nginx 服务器

该部分包括了服务器的功能、Nginx 服务器的模块管理和进程管理、5XX 错误处理、协助用户操作 Nginx 的工具和高可用的 Nginx 等内容。

第 1 章　Nginx 的功能

本章认识 Nginx 服务器的基本功能和其扩展功能，以及 Nginx 核心模块的相关指令和变量。

第 2 章　Nginx 的模块管理和进程管理

Nginx 同 Apache 一样，同样使用了模块化管理，但是与 Apache 有很大的不同，如果说 Apache 支持“热插拔”（就是说如果对 Apache 添加模块不用重新编译 Apache，而只是添加必要的模块，然后再重新载入 Apache 就可以了），那么 Nginx 则必须“重启动”，就是说如果要对 Nginx 服务器添加模块，那么需要重新编译 Nginx 才可以添加相应的功能模块，因此在这一点上要比 Apache 服务器麻烦。

第 3 章　Nginx 如何处理一个请求

Nginx 服务器在处理一个请求时是按照两部分例子进行的，第一部分是 IP、域名，第二部分是 URI。下面本章将分析这两部分是如何进行工作的。

第 4 章　服务器名字

服务器的名字是由指令 server_name 来定义，并且也决定了使用哪一个 server 区段来提供对客户端请求的响应。服务器名字的定义可以使用准确的名字（exact name）、通配符名字（wildcard name）或者是正则表达式。

第 5 章　协助用户操作 Nginx 的工具

工具 nginx.vim 是一个辅助工具，通过下面的配置，在使用时它将成为 vim 工具的一部分，具体的作用是用于编辑 Nginx 的配置文件。

这个工具是一个 shell（Bash）脚本，是 Debian 系统下用于控制 Apache2.2 虚拟主机命令 a2ensite 和 a2dissite 的复制版，用于控制 Nginx。原始的 a2ensite 和 a2dissite 是用 Perl 语言编写，a2dissite 是 a2ensite 的一个符号链接，在本工具中，开发者遵循了同样的方法，例如，nginx_dissite 是 nginx_ensite 的一个符号链接。

如果没有安装 Apache，那么可以使用该工具来产生和管理 htpasswd 文件。

Nginx 启动脚本，由于在默认的安装包中没有提供管理脚本，为了管理方便，我们需要为它添加一个启动/关闭/脚本。

第 6 章　5XX 错误处理

本章介绍了 500、502 和 504 错误代码的原因及处理。

第 7 章　使用 TCMalloc 优化 Nginx

TCMalloc 即 Thread-Caching Malloc 的缩写，它是由 Google 公司开发的一款开源工具 google-perftools 中的一成员。TCMalloc 在内存的分配上效率和速度要比标准的 glibc 库高得

多，它不但可以用来优化高并发下的 MySQL，从而可以降低系统的负载，同样它可以用于 Nginx 实现同样的功能，因此，对于高并发的 Nginx 来说无疑是如虎添翼。

第 8 章　PCRE 正则表达式

由于在使用 Nginx 的过程中会用到正则表达式，因此有必要对这一部分内容进行一个清晰的说明。并且讲述了 pcre-config 和 pcretest 命令，以及 Nginx 服务器支持 UTF8 正则表达式匹配。

第 9 章　Nginx 高可用的实现

我们在 Nginx 下实现高可用不是担心 Nginx 服务出问题，而是担心硬件的问题，因此在设计通过 Heartbeat 服务来提供高可用时没有通过 Heartbeat 服务器来控制 Nginx 服务器的启动、停止等等，而只是通过它来提供了一个“浮动”的 IP 地址，次 IP 就是 Nginx 服务器提供访问的 IP，也就是被解析到相应域名的 IP 地址。

第 10 章　10 个 QA

10 个经常被问的问题。

第二部分　Nginx 服务器的功能

在中篇中通过“讲”的方式，进行了 42 讲，详细的讲述了 Nginx 提供的模块实现的功能：

第 11 章　限制流量
第 12 章　限制用户并发连接数
第 13 章　修改或隐藏 Nginx 的版本号
第 14 章　配置 FLV 服务器
第 15 章　Nginx 的访问控制
第 16 章　提供 FTP 下载
第 17 章　Nginx 与编码
第 18 章　网页压缩传输
第 19 章　控制 Nginx 如何记录日志
第 20 章　map 模块的使用
第 21 章　Nginx 预防应用层 DDoS 攻击
第 22 章　为 Nginx 添加、清除或改写响应头
第 23 章　重写 URI
第 24 章　Nginx 与服务器端包含
第 25 章　Nginx 与 X-Sendfile
第 26 章　在 Nginx 的响应体之前或之后添加内容
第 27 章　Nginx 与访问者的地理信息
第 28 章　Nginx 的图像处理
第 29 章　location 中随机显示文件
第 30 章　后台 Nginx 服务器记录原始客户端的 IP 地址

第 31 章　解决防盗链
第 32 章　Nginx 提供 HTTPS 服务
第 33 章　监控 Nginx 的工作状态
第 34 章　使用 empty_gif
第 35 章　Nginx 对响应体内容的替换
第 36 章　Nginx 的 WebDAV
第 37 章　Nginx 的 Xslt 模块
第 38 章　Nginx 的基本认证方式
第 39 章　Nginx 的 cookie
第 40 章　Nginx 基于客户端请求头的访问分类
第 41 章　通过 Upstream 模块使得 Nginx 实现后台服务器集群
第 42 章　根据浏览器选择主页
第 43 章　关于 Nginx 提供下载.ipa 或.apk 文件的处理方法
第 44 章　SCGI
第 45 章　Expires 与 ETag
第 46 章　使用 upstream_keepalive 模块实现 keep-live
第 47 章　后台服务器的健康检测
第 48 章　使用 sticky 模块实现粘贴性会话
第 49 章　Nginx 实现对后台服务器实现“公平”访问
第 50 章　Nginx 使用 redis 数据库
第 51 章　Nginx 访问 MongoDB
第 62 章　Nginx 访问 Mogilefs

第三部分　Nginx 与缓存

通过 Nginx 来实现缓存功能基本上有四类方法，其中第一类方法 Nginx 自带，即 proxy_cache、proxy_store 和 memcached，在没有 proxy_cache 之前只能用 proxy_store 缓存页面，但是 memcached 需要第三方软件 Memcached；第二类是通过添加第三方模块，下面我们通过例子来分析；第三类是通过 Varnish 服务器。这样共有五种方法来实现缓存。

第 53 章　缓存技术——proxy_cache

Nginx 的这个功能从 0.7.48 版本开始提供的，它开始支持类似 Squid 的缓存功能。它的原理是把 URL 及相关变量的组合当做 Key，再用 MD5 编码，编码后的哈希值作为文件名，然后再将缓存的文件保存在硬盘中。我们现在使用的是 0.8.53 版本，早在 0.8.31 版本中 proxy_cache 就比较完善了，之所以说它比较完善其实就是说它没有缓存清除机制，但是通过第三方模块 ngx_cache_purge 来清除指定的 URL 缓存，它支持任意 URL 链接、支持 404/301/302/200 状态码缓存，因此，在使用反向代理的同时使用 Nginx 的 proxy_cache 缓存机制是个很不错的做法。

第 54 章　缓存技术——proxy_store

在没有 proxy_cache 之前，都是使用 proxy_store，由于 Nginx 的 proxy_store 技术当时

在设计时没有采用任何刷新机制，因此，对于缓存的管理上也就更人为了一些（你不觉得这么说更好听些？!），我们可以自己写个脚本进行缓存清理等等。使用 proxy_store 的原理其实也很简单，那就是 Nginx 首先在本地查找客户端请求的内容，如果找不到就去 proxy_pass 指定的后端服务器上查找，然后会被保存到本地的缓存中。使用 proxy_store 技术有一个缺点，其实也能认为是优点，它会在本地缓存中构建一个和远程服务器——proxy_pass 指定的后端服务器目录结构完全一样的结构（真得可以叫镜像了!），这是从它的优点方面讲。从缺点方面讲，那么它会浪费很大的磁盘空间，如果没有一个足够大的硬盘或者是没有及时地清除缓存中的过时文件，那么将是一个可怕的问题，所以你要么准备一个足够大的硬盘，要么及时或是有计划地清除缓存。

第 55 章　缓存技术——Memcached

这是使用比较多的一种缓存方式，使用 Memcached 模块也会分为四部分分析，第一部分就是 Nginx 的 memcached 模块本身，而另一部分就是 Memcached 服务，第三部分就是 Nginx 的配置文件，将这二者结合起来，就是第四部分客户端。

第 56 章　缓存技术——NCACHE

这个缓存方式我们只需要了解一下就可以了。第三方模块用得较多的就是新浪网的开源项目——NCache。这是一个比较古老的模块了，NCache，Nginx Cache，它只支持 Nginx 的 0.6.x 版本，对于 Nginx 的其他版本并不支持，相反的是在后面的版本中是以 Nginx 内核形式出现的。

第 57 章　缓存技术——Varnish

在 Nginx 中 Varnish 缓存是通过代理模块来实现的，在我使用的 Varnish 缓存服务器中，其中之一就是为了 Apache 而使用它。

Varnish 是一个高性能的、先进的 Web 加速器，它可以安装在 Linux 2.6，FreeBSD 6/7 和 Solaris 10 系统上，以便发挥其高效的性能。

卷 2 内容

第 1 部分　Nginx 与 php

第 2 部分　Nginx 与 Python

第 3 部分　Nginx 与 Perl

第 4 部分　Nginx 与 Java

第 5 部分　Nginx 与 Ruby

第 6 部分　Nginx 与 ASP.NET

后记　Nginx 与 Apache

第 1 部分　Nginx 与 PHP

要将 Nginx 和 PHP 结合，让 Nginx 解析静态网页，而 PHP 的动态网页交给 PHP 处理。

解决方法从大的方向有两类：（从 Nginx 角度来讲）一类是使用 Nginx 的代理模块，而另外一类则是使用 FastCGI 模块；而从 PHP 角度来讲则是 FastCGI 进程，它的方法有三种：一种是以 php-fpm 方式运行，第二种是 PHP 自带的 fastcgi server，第三种就是借助 lighttpd 带的 spawn-fcgi（听起来有点龌龊，但是确实可行，有时候还必须使用这种方法）。

第 1 章　环境部署

本章作为该部分的第 1 章，我们首先来部署环境。在 PHP 方面我们使用了 php-fpm，而在 Nginx 方面我们使用的是 FastCGI 模块。

第 2 章　PHP 访问 Memcached

为理解 Memcached 的使用，我们在这里讲述两个例子，一个是微博网站的部署，具体就是用 Memcached 存储什么数据，我没有仔细分析过，只负责部署和测试是否符合要求就够了；另一个是在我所运维环境中使用的一个例子，非常简单且实用。

在 PHP 下使用 Memcached 服务器，有两个选择，一个是纯 PHP 框架开发的 memcache，而另一个则是使用 libmemcached 的 memcached。

第 3 章　php-fpm 的状态

了解 php-fpm 的工作状态。

第 2 部分 Nginx 与 Python

本部分我们将了解到 Nginx 的 uwsgi 模块、uWSGI 服务器以及 ngx_cache_purge，当然根据需要我们还可以对 Django 架构了解一下。

第 4 章　uWSGI 服务器

uWSGI 是一个快速的、自维护的、对开发者和系统管理者友好的应用程序容器，是纯 C 语言开发的服务器。

在它的诞生之日，uWSGI 只是作为一个 WSGI 服务器，但是随着时间的推移，它现在已经演变为一个完整的网络、集群 Web 应用服务器，可以执行消息、对象传递、缓存、RPC 和进程管理。

它使用的协议是 uwsgi（注意，所有的字母都是小写，该协议已被 Nginx 和 Cherokee 的发行版本所包含），所有网络或进程间通信均使用 uwsgi 协议。

uWSGI 可以运行在预 fork 模式、线程模式、异步模式等，并且支持 green threads、coroutines 各种形式，例如 uGreen，Greenlet，Stackless 和 Fiber。

对于管理人员来说，uWSGI 服务器提供了各种配置方法：命令行、环境变量、XML、ini、yaml、json、sqlite3 数据库和 LDAP。

除此之外，它的设计完全模块化，这意味着，可以使用不同的插件以便满足不同的技术应用，从而实现兼容性。

第 5 章　Nginx 的 uwsgi 模块

该模块能够使得 Nginx 与 uWSGI 进程进行交互，并且可以控制传递给 uWSGI 进程的参

数。对于 uwsgi 协议和 uWSGI 服务器，uWSGI 服务器就是 uwsgi 协议的一个实现。

第 6 章　环境部署

在了解了 Nginx 的 uwsgi 模块、uWSGI 服务器以及 ngx_cache_purge 之后，当然根据需要我们还可以对 Django 架构了解一下，在这里我们按照一个全新的环境来布置，即从安装 Python 开始。在后面的实例部分，我们使用了 Python 的 2.43 版本，这是 Red Hat 系统自带的，同时也使用 2.7.2 版本，具体的版本根据自己的需要去选择。

第 7 章　实例运行

下面我们通过实例的方式来讲述，在下面的内容中将会讲述 8 个运行实例。

- 实例 1：运行开发服务器
- 实例 2：以 uWSGI 方式运行
- 实例 3：使用 Django 框架
- 实例 4：一个 uWSGI 实例实现对多个虚拟主机的支持
- 实例 5：分别监听在不同端口上的两个 uWSGI 实例
- 实例 6：针对 Nginx uwsgi 模块应用举例的一个具体实现
- 实例 7：集群的实现
- 实例 8：会话存储（基于数据库的方式和基于 Memcached 的两种方式）

第 8 章　缓存

对于 Memcached 服务器，Python 客户端方式选择有三种：一是 python-memcached，它的下载地址是 http://www.tummy.com/Community/software/python-memcached/，最新版本为 1.47；二是 cmemcache，它的下载地址为 http://gijsbert.org/cmemcache/，最新版本为 0.95；三是 libmemcached，它的下载地址为 http://download.tangent.org/，最新版本为 0.9。

第 9 章　会话

session 是 Django 中的一个高级工具，它可以存储用户的个人信息，以便用户在下次访问该网站时使用这些信息，session 的基础还是 cookie，但是它能够提供一些更加高级的功能。

Django 提供了完全支持匿名会话的功能，它的会话结构让每个网站的访问者存储并且检索任意数据。它将数据存储在服务器端并且对发送和收到的 cookies 做摘要，cookies 包含一个会话 ID，而不是数据本身。Django 只在需要的时候才发送 cookie，如果我们没有设定任何的 session 数据，它不会送出 cookie。

此外，还需要明白一点，Django 的会话（session）框架是完全基于 cookie 的，并且它也只能是基于 cookie，而不会像其他一些软件（例如 PHP）那样，在 session 不能正常工作时，就会把 session ID 放到 URL 中。任何事情只要存在就有它的道理，这一决定是经过深思熟虑的，将 session ID 放到 URL 中的那种方法不仅使得 URL 很丑陋，并且 session ID 还有可能通过 Referer 头泄漏出去，从而给网站带来安全隐患，这就是 Django 基于 cookie 的原因。

第 3 部分　Nginx 与 Perl

在 Nginx 的设计上并不支持 CGI，这不是它本身的缺陷，而是一个重要的举措：因为 Nginx

不能够直接执行外部程序（CGI），因此，怀有恶意的人就不能随意地直接执行外部脚本程序。当然，什么事情都是有方法可循的，例如，PHP FastCGI 脚本的支持，当我们将一个 PHP 脚本上传到一个可以执行 PHP FastCGI 的目录中，那么该脚本就可以执行，这个我们已经了解过了，但是这种方法是有一点难度，但是相对来说比较安全。但是有时候我们也需要简单的 CGI 程序支持，我们将介绍一个简单的 CGI 来替代 FastCGI，我们称这样的程序为 CGI 程序，它是用 Perl 语言实现的。

本部分实现了三个应用即 CGI、perl-FCGI 和 Nginx 内置的 Perl 模块。

第 10 章　Nginx 提供 Perl CGI 访问

第 11 章　Nginx 与 perl FastCGI

要将 Nginx 和 Perl 结合，让 Nginx 解析静态网页，而 Perl 的动态网页交给 Perl 处理。解决方法从大的方向有两类：（从 Nginx 角度来讲）一类是使用 Nginx 的代理模块，而另外一类则是使用 FastCGI 模块，而从 Perl 角度来讲则是 FastCGI 进程。

有关 Memcached 服务器的使用通过以下客户端作为实现，Perl 的 memcached 客户端有：

- Cache::Memcached
- Cache::Memcached::Fast
- Memcached::libmemcached
- Cache::Memcached::libmemcached

第 12 章　Nginx 通过内置的 Perl 模块执行 Perl 程序

通过使用该模块，Nginx 服务器可以直接在 Nginx 内部执行 Perl，或者是通过 SSI 来调用 Perl。

第 4 部分 Nginx 与 Java

在 Java 部分我们选择了 Tomcat 服务器作为 Java 的解析器。在这里着重分析了 Tomcat 的配置文件，及其实际应用中的配置。

第 13 章　环境部署

在我们实际使用 Nginx 时，往往不是 Nginx 单独工作，相反的是，它总是与一些动态语言所使用的“解析器”（这个名字是我起的，不是权威，可不要效仿！）构成动态网站，例如，我们将要接触的 Tomcat。Tomcat 在与 Nginx 的搭配中，我们使用 Nginx 的反向代理功能。

第 14 章　Nginx 与 Tomcat 的结合

Nginx 与 Java 的实现方式是通过代理模块来实现的，在 Nginx 方面使用代理模块，而 Java 方面，可以使用 Tomcat 也可以使用 Resin，还可以是其他的应用服务器。因此在这里我们分为三方面来讲述这个问题：一是代理模块、二是 Tomcat 或 Resin 的配置，三是这两者的结合。将 Nginx 作为前台服务器而把 Tomcat 作为后台服务器，由 Nginx 解析静态文件，而将动态的 JSP 网页发送到后台的 Tomcat 服务器。Nginx 在默认时就将该代理（Proxy）模块建立在内，通过使用该模块允许你将客户端的 HTTP 请求转发到后台服务器。

第 15 章　配置 server.xml 文件

从本章开始我们将来认识 Tomcat 的配置文件。对于我们运维人员来说，服务器的精髓都在配置文件中。

第 16 章　配置 web.xml 文件

首先我们要确定的是这个文件会有两种位置：一种是在$CATALINA_BASE/conf/目录下，而另一种情况是在$CATALINA_BASE/webapps/[webapp]/WEB-INF/目录下，前者可以叫做全局 web.xml 配置，而后者是各个 Web 应用程序自己的配置，Tomcat 在部署 Web 应用程序时，可能会包括两种情况，一种是 Tomcat 在启动时，第二种是 Tomcat 对应用程序进行重新加载时。在这两种情况下，Tomcat 都会先去读取全局配置（即 conf/web.xml），然后再去读取各自 web 应用程序目录下的配置（即 WEB-INF/web.xml）。

但是我们需要明白两点：一是全局配置（conf/web.xml）会将配置文件中的配置应用到所有的 Web 应用程序，而每个 Web 应用程序自己的配置只能应用自己，因此不要在全局配置文件中配置基于特定应用程序的资源；二是对于每个 Web 应用程序来说，它既可以有自己的 web.xml，也可以没有，如果没有该配置文件，Tomcat 会给出提示，但不会停止部署该应用程序。

第 17 章　配置 context.xml 文件

本章我们来了解 context.xml 文件的配置情况。

第 18 章　配置 tomcat-users.xml 文件

本章将要了解的是 Tomcat 的用户配置文件，包括用户、角色和密码。

第 19 章 配置 catalina.policy 文件

对于本文件来说，最大的特点是只有开启了 Security Manager 后，该文件才被使用，如果没有开启，那么这个配置就是一个摆设。

这一部分内容严格地来说属于 Java 的安全，和 Tomcat 本身关系并不大，但是对于我们运维来说，真不知道什么是不需要的，老实说这一部分内容如果处理不好，一是可能会引发安全问题，二是可能会导致软件工程师所写的 Java 程序在 Tomcat 上运行不了。

第 20 章 配置 catalina.properties 文件

在该部分内容中讲述两个内容：一是 catalina.properties 文件分析，二是 Loader 元素和类的加载器。

虽然两者看起来是不同的内容，但是在一定程度上，它们确实完成相同的工作。

第 21 章 在容器元素中可以使用的过滤器

Tomcat 提供了许多过滤器（Filters），可以将这些过滤器配置在所有 Web 应用程序都可以使用的$CATALINA_BASE/conf/web.xml 文件中，也可以配置在单独的 WEB-INF/web.xml 文件中，以便应用到相应的 Web 应用程序中。

第 5 部分　Nginx 和 Ruby

本部分的内容为：Ruby、Rails，即 RoR 和 Passenger。

首先讲述了 Ruby 的安装及相关工具 gem，然后是 Passenger，Passenger 与 Nginx 的结合安装方式有两种，即 Passenger 安装和 Nginx 模块式安装，以及 Passenger 提供的相关分析和系统维护工具。Rails 架构是一个纯 Ruby 的 MVC 架构，我们在这部分从运维的角度详细地分析了 Rails，包括 Rails 的相关技术和项目实例分析。

Rails 提供了缓存技术，在分析 Rails 提供的缓存技术基础之上我们着重使用了 Dalli 缓存接口和 Memcached 服务器。

第 22 章　环境部署

本章是该部分的第 1 章，因此我们首先要部署 Ruby 环境，Nginx 与 Python 的结合是通过 Passenger 来实现的，因此在本章中要介绍两种部署 Passenger 的方法，Passenger 模块及其在 Nginx 中的配置。

第 23 章　走进 Rails

在这一章中我们将认识 Ruby 的一个框架 Rails，也就是 RoR 的另一个 R。

第 24 章　缓存

Rails 架构提供了不同的缓存技术，对于行为（action）和片段（fragment）缓存策略来说可以使用这些缓存技术，而对于页面（page）缓存技术来说，它总是存储在磁盘上。

缓存技术有以下几种：

- 内存缓存技术
- 文件系统缓存技术
- Memcached 服务器技术
- Ehcache 缓存技术
- Dalli —— Memcached 的客户端

第 6 部分　Nginx 与 ASP.NET

本部分讲述的是通过 Mono 将 ASP.NET 程序运行在 Linux 操作系统下，ASP.NET 跑在 Linux 系统下的使用实例不是没有，但也不是很多，好像是越来越多的出现这种跨平台的移植现象。

第 25 章　Mono

在本章中我们简单地认识一下 Mono。

在本章需要了解两个问题，一个就是什么是 Mono，另一个是 Mono 的使用级别。

第 26 章　Nginx 与 ASP.NET 的解决方案

在本章我们实施了三个方案，每一个方案都有自己的特点，在具体的实施中可以做适当的选择。在这部分中讲述了三种结合方案：

- 方案一：Nginx+mono+ fastcgi-mono-server

- 方案二：Nginx+mono+Jexus
- 方案三：Nginx+mono+xsp

第 27 章　Session 存储

由于 HTTP 协议是无状态的协议，因此在客户端每次访问 Web 页面时，作为服务器端都要重新打开会话，作为开发人员，或者说是从用户使用的角度来看待这种无状态的协议是不会自动维护客户端所访问的环境信息，因此需要 session。

第 28 章　缓存

关于缓存我们已经讨论了很多，从缓存存储的位置来讲，归纳起来无非三种：

- 服务器端缓存
- 第三方缓存
- 客户端缓存

从缓存内容来讲，分为两种：

- 缓存动态内容
- 缓存静态内容

对于静态文件的缓存，由于我们通过 Nginx 做了动静分离，因此，通常静态文件是通过 Nginx 提供服务。

从缓存由动态程序产生的缓存来看，又分为以下类型：

- 全部内容缓存
- 部分内容缓存

那么我们在这里作为总结分析一下。

第 29 章　Nginx 代理 IIS

与 Mono 相比，更多的 ASP.NET 是运行在 IIS 服务器之下。在一个大型的网站中，往往会出现使用多种语言的情况。如果使用了 ASP.NET 技术，就免不了有 Windows 的系统，也就是使用了 IIS。在这种情况下，为了使用 Nginx 的高效性（运行在 Linux 下），可以使用 Nginx 的代理模块来实现，将 IIS 作为后台服务器来运行，而让 Nginx 服务器运行在前台

后记　Nginx 与 Apache

这是一个独立的部分，不再有章节，一是为了怀念 Apache，二是为了更好地使用 Apache，在没有 Nginx 的那个年代，Apache 为我们的网站作出了不可磨灭的贡献！

Nginx 服务器的功能再强大，不要忘记为我们立下汗马功劳的 Apache 服务器，否则你就是一个忘恩负义的人。

我不赞成用 Nginx 替换掉 Apache，相反我们可以使用混合的环境，对于这种 Nginx、Apache 混合使用的架构中，一是根据实际的应用来逐步使用 Nginx，二是在现有基础上实现 Nginx+Apache，具体的方法还是反向代理，让 Apache 运行在后台，而 Nginx 运行在前台。

使 用 对 象

- 广大的 Linux 爱好者。
- 具有一定 Linux 基础的系统管理员。
- Linux 下的 Web 服务器管理员。
- Linux 服务器下动态语言开发人员。
- Nginx 服务器管理员。
- 培训中心。
- 运维人员。
- 一切应该了解和使用 Nginx 的用户。

内 容 声 明

关于本书内容的说明，如果你在哪里看到了与本书雷同的内容，你需要确定一下它的内容是否来自于相应软件的官方网站、man 文档、howto、README，Changelog、INSTALL、LICENSE、*.conf 等，这些是原创，在我看来，什么是原创，只有这些才是原创（我个人的观点，别拿砖头拍我！），我们只不过是对它们的衍生和应用，本书中的内容就是这样，这是我个人的一个学习方法，对于每一个新使用的软件，我都会看它提供的相关文档和其官方网站，配置文件绝对是软件的精华所在，因此在本书中讲述了大量的配置文件，没办法，Linux 下的服务不就是命令加上配置吗？

由于这些官方网站、man 文档、howto、README，Changelog、INSTALL、LICENSE、*.conf 等都是英文的，因此对于我们的认识和阅读是不方便的，事实上我们也正是缺乏这些文档的知识才导致我们一直徘徊在技术的门口，因此本人就是基于这个基础来编写本书，将这些最基础也是最权威的文档通过理解来实现汉语化，以便更多的国人阅读，以个人的感觉，这些东西实际上是我们最需要的，它是认知的第一步，毕竟我们的官方语言是汉语。

书中的内容是我在工作中的一个总结，我没有去刻意改变一个说法，相反只要是官方文档中有的，我就尽可能地采用它们的提法、说法及方法。

与儿子的对话

有一天我儿子问我：“写这套书能挣多少钱？”

这一句话问的不是我，而是问到了人的灵魂，我不知道是否又要归功于教育？有一首歌叫做《我在马路边捡到一分钱》，回忆我小时候，我们学过这首歌，我做到了捡到交公。

我问我儿子：“你学过这首歌吗？”

我儿子说没有学过，他没有学过，但是他也做到了，不但他做到了，而且也帮助他的同学做到了，对于这些我不便多说，报纸有报道。

但是儿子能够问我“写这套书能挣多少钱？”，我确实没有想到。

于是我对他说，写这套书我有三个目的。

第一，对我这么多年做系统管理的一个总结；

第二，帮助那些需要帮助的系统管理和运维人员；

第三，劳动者总是光荣的，出版者给我的报酬是一个物质的体现。

这三个目的是有联系的，没有经验就无法总结，没有总结何谈帮助别人，一个人生活在人世间不只有索取才能感到快乐，相反，给别人带来快乐才是真正的快乐，我希望每个人的奉献要远远大于索取，就是说奉献是我们真正要做的，那么我们的人类才会有真正意义上的和谐，而不是一句口号、一个形式或一个章程！

我儿子又问我：“那干嘛不免费发表在互联网上？”

我对儿子是这样说的：

在这个社会形式下，没有钱是无法生存的。达尔文的进化论说了“适者生存”，达尔文的进化论也说了“要么选择，要么适应”，因此，我们需要适应这个社会。首先我不知道怎么适者生存，但我们要解决自己的温饱问题。

还有一个问题，我发表在网上的文章被别人抄袭了，而且有很多地方都抄错了，最后还有人问我是不是抄袭了别人的，这个很让我郁闷！其实我写在网上的文章就是让别人看的，“天下文章一大抄，就看你会抄不会抄”，这个是小学就明白的道理，技术又何尝不是如此，看看我们所使用的这些技术，有哪个是我们自己的技术，人家外国人什么时候找过我们的麻烦！因此，关于本书的内容，我在“内容声明”中做了特别的说明。总言之，第二个不发表在网上的原因就是本书具有可行性（照着本书，每一个实例都能实现），因此，在转载中的修改对于最终的读者会有障碍。

这是我与我 9 岁儿子的对话。

关 于 读 者

为了使读者快地进入 Nginx 世界，可以从以下四个方面来认识 Nginx。

从功能上要认识到以下五点：

- 提供静态文件和 index 文件，生成自动索引，打开文件描述符缓存；
- 使用缓存加速反向代理，简单的负载平衡和容错；
- 使用缓存机制加速远程 FastCGI 服务器的访问，简单的负载平衡和容错；
- 模块化结构
- 支持 SSL 和 TLS SNI。

对于“邮件代理服务器功能”也可以做适当的了解，毕竟也是 Nginx 的一个功能。

从使用上要认识到以下两点：

于 Nginx 的使用，我们有两点要认识：

- 高并发访问，解决了 10K 的问题；
- 代理，作为代理是它最主要的功能，因此，我们在学习 Nginx 时这是它的主线。

对于 Nginx 的工作机制要认识以下八点：

- 一个 master 进程和几个 workers 进程，workers 进程由非特权用户运行。
- 消息通知方法：kqueue（FreeBSD 4.1+），epoll（Linux 2.6+），rt signals（Linux 2.2.19+），/dev/poll（Solaris 7 11/99+），event ports（Solaris 10），select 和 poll。
- 支持 kqueue 的各种功能，包括 EV_CLEAR，EV_DISABLE（禁用临时事件，NOTE_LOWAT，EV_EOF，number of available data，错误代码。
- 支持 sendfile（FreeBSD 3.1+，Linux 2.2+，Mac OS X 10.5），sendfile64（Linux 2.4.21+），和 sendfilev（Solaris 8 7/01+）。
- File AIO（FreeBSD 4.3+，Linux 2.6.22+）。
- 支持 Accept-filters（FreeBSD 4.1+）和 TCP_DEFER_ACCEPT（Linux 2.4+）。
- 10 000 个非活动 HTTP keep-alive 连接用掉 2.5MB 内存。
- 数据复制操作控制在最低限度。

对于安装平台要认识以下五点：

- FreeBSD 3 — 8 / i386，FreeBSD 5 — 8 / amd64。
- Linux 2.2 — 2.6 / i386，Linux 2.6 / amd64。
- Solaris 9 / i386，sun4u，Solaris 10 / i386，amd64，sun4v。
- MacOS X / ppc，i386。
- Windows XP。Windows Server 2003。

作 者 声 明

本书的内容是我在工作中的一个总结，在生产环境中都使用过，并非纸上谈兵，但是书中的例子，我尽可能地不使用生产环境中的例子，**一是怕对你造成误导，二是不想说什么是权威。**

我在前面说了**文稿内容**的来源，对于文稿的构成，一部分是对员工培训的文稿，一部分是在培训中心的文案，还有一部分是我在学习中的笔记，由这三部分融合而成，而非简单的拼凑。

另外，毕竟我们都是做互联网的，每天面对着无数个页面，我所要说的是，如果读者在阅读本书的过程中发现有和网络上相似的内容，那么确定一下是否是两者（即笔者和您看到大文章的作者）参考了同一个官方的资料，本人绝对没有有意抄袭其他作者的内容，这是第一；第二，如果真的是我写的内容确实是和您的内容有相同之处，那么及时和我联系（绝对是缘分！）；第三，互联网给了我们发展，也给了我们交流，如果您在看本书的过程中发现有个别说法、提法和您的相同，那么请您海涵，往往是一个提法、说法用久了就觉得是自己的说法了（我相信谁都会犯这种错！）；第四，由于本人是做运维（系统管理和网络管理），因此在写作风格上也是按照自己的认知过程所写，既没有受过专业的训练也没有模仿某个作者