

轻量级 WEB 服务器 Nginx

By
Mark & Kevin

版权声明：

本文遵循“署名-非商业性使用-相同方式共享 2.5 中国大陆”协议

您可以自由复制、发行、展览、表演、放映、广播或通过信息网络传播本作品

您可以根据本作品演绎自己的作品

您必须按照作者或者许可人指定的方式对作品进行署名。

您不得将本作品用于商业目的。

如果您改变、转换本作品或者以本作品为基础进行创作，您只能采用与本协议相同的许可协议发布基于本作品的演绎作品。

对任何再使用或者发行，您都必须向他人清楚地展示本作品使用的许可协议条款。

如果得到著作权人的许可，您可以不受任何这些条件的限制。

Mark Zhu

Kevin Zou

本文目录

前言	1
第 1 章 Nginx 简介	2
第 1.1 节 WEB 服务器简介	2
第 1.2 节 几个常用 WEB 服务器简介	2
第 1.2.1 节 Apache 服务器	2
第 1.2.2 节 Lighttpd 服务器	3
第 1.2.3 节 Tomcat 服务器	3
第 1.2.4 节 IBM WebSphere	3
第 1.2.5 节 Windows IIS	4
第 1.2.6 节 Nginx 的历史	4
第 1.3 节 为什么要用 Nginx	5
第 1.4 节 获取 Nginx	6
第 1.5 节 课后练习.....	6
第 2 章 Nginx 的安装和启动	7
第 2.1 节 Nginx 的安装	7
第 2.2 节 2.2 Nginx 的启动、重启与停止	8
第 2.3 节 课后练习.....	14
第 3 章 Nginx 的基本配置	16
第 3.1 节 Nginx 主配置概述	16
第 3.2 节 Nginx 下的虚拟主机配置	23
第 3.3 节 安全的连接 https	26
第 3.4 节 Nginx 日志管理	27
第 3.5 节 访问控制	29
第 3.6 节 课后练习.....	30
第 4 章 Rewrite 规则	32
第 4.1 节 什么是 Rewrite	32
第 4.2 节 Rewrite 相关指令	32
第 4.3 节 课后练习.....	36
第 5 章 Nginx 反向代理.....	38
第 5.1 节 反向代理及其特点.....	38
第 5.2 节 Nginx 反向代理.....	38
第 5.3 节 Nginx 反向代理配置.....	40

第 5.4 节 课后练习.....	45
第 6 章 源代码搭建 LNMP	47
第 6.1 节 安装 rpm 包	47
第 6.2 节 安装 Nginx	48
第 6.3 节 编译安装 Mysql	49
第 6.4 节 编译安装 php	50
第 6.5 节 安装 php 扩展模块工具	55
第 6.6 节 使 PHP 扩展生效.....	58
第 6.7 节 整合 Nginx 与 php	60
第 6.8 节 启动 Nginx 服务	65
第 6.9 节 课后练习.....	66
第 7 章 课后练习参考答案.....	67

前言

欢迎大家来到尚观学习 Linux 课程，希望大家通过在这学习，获取到扎实有用的知识，为将来的锦绣前程铺平道路，祝大家在这度过艰苦而快乐的 4 个月。正如我常说的“辛苦 4 个月，幸福后半生。”

本文档共分 7 章，深入浅出的介绍了当前主流的轻量级 WEB 服务器 Nginx，带领大家一步一步了解 Nginx，并对它做相应的安装，配置，管理，调优。需要注意的是，本书默认你是一名具有 Linux 操作基础的读者，因此当涉及到一些 Linux 操作的知识的时候，本文档不会做过多讲解。当你学习完本文档内容后，你将会有能力去搭建一个高效、稳定、功能丰富的 WEB 服务器。

然而，学如行舟，不进则退。做为一个学者，我们应该不断的探索，大家在学完后仍应该继续深入研究，让自己站的更高，才能够看的更远。

好了，接下来请跟随我一起进入 Nginx 的世界吧。

第1章 Nginx 简介

第1.1节 WEB 服务器简介

WEB 服务器也称为 WWW(WORLD WIDE WEB)服务器，主要功能是提供网上信息浏览服务。Web 已经成为很多人在网上查找、浏览信息的主要手段。成千上万的用户通过简单的图形界面就可以访问各个大学、组织、公司等最新信息和各种服务。商业界很快看到了其价值，许多公司建立了主页，利用 Web 在网上发布消息，并将它作为各种服务的界面，如客户服务、特定产品和服务的详细说明、宣传广告以及是渐增长的产品销售和服务。商业用途促进了网络的迅速发展。如果你想通过主页向世界介绍自己或自己的公司，就必须将主页放在一个 WEB 服务器上，当然你可以使用一些免费的主页空间来发布。但是如果你有条件，你可以注册一个域名，申请一个 IP 地址，然后让你的 ISP 将这个 IP 地址解析到你的 LINUX 主机上。然后，在 LINUX 主机上架设一个 WEB 服务器。你就可以将主页存放在这个自己的 WEB 服务器上，通过它把自己的主页向外发布。

第1.2节 几个常用 WEB 服务器简介

Unix 和 Linux 平台下的常用 Web 服务器有 Apache、Nginx、Lighttpd、Tomcat、IBM WebSphere 等。其中目前应用最广泛的 Web 服务器是 Apache。Windows 平台下最常用的服务器则是微软公司的 IIS(Internet Information Server)。

第1.2.1节 Apache 服务器

Apache 起初由 Illinois 大学 Urbana-Champaign 的国家高级计算程序中心开发。此后 Apache 被开放源代码团体的成员不断的发展和加强。1996 年 4 月以来，Apache 一直是 Internet 上最流行的 HTTP 服务器，1999 年 5 月它在 57% 的网页服务器上运行，到了 2005 年 7 月这个比例上升到 69%。Apache 是目前世界上用的最多的 Web 服务器，它有优势主要在于源代码开放、有一支开放的开发队伍、支持跨平台的应用(可以运行在几乎所有的 Unix、Linux、Windows 系统平台之上)。Apache 的模块支持非常丰富，以至于它提

供了非常完善的功能。

Apache 的官方网站: <http://www.apache.org>

第 1.2.2 节 Lighttpd 服务器

Lighttpd 是一个德国人领导的开源软件，其根本的目的是提供一个专门针对高性能网站，安全、快速、兼容性好并且灵活的 Web server 环境。具有非常低的内存开销，cpu 占用率低，效能好，以及丰富的模块等特点。Lighttpd 是众多 OpenSource 轻量级的 Web server 中较为优秀的一个。支持 FastCGI, CGI, Auth, 输出压缩(output compress), URL 重写, Alias 等重要功能，而 Apache 之所以流行，很大程度也是因为功能丰富，在 Lighttpd 上很多功能都有相应的实现了，这点对于 Apache 的用户是非常重要的，因为迁移到 Lighttpd 就必须面对这些问题。

Lighttpd 的官方网站: <http://www.lighttpd.net>

第 1.2.3 节 Tomcat 服务器

Tomcat 是一个免费的开源的 Servlet 容器，它是 Apache 基金会的 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其它一些公司及个人共同开发而成。由于有了 Sun 的参与和支持，最新的 Servlet 和 Jsp 规范总能在 Tomcat 中得到体现。Tomcat 即是一个 Jsp 和 Servlet 的运行平台。同时 Tomcat 又不仅仅是一个 Servlet 容器，它也具有传统的 Web 服务器的功能：处理 Html 页面，但是与 Apache 相比，它的处理静态 Html 的能力就不如 Apache，我们可以将 Tomcat 和 Apache 集成到一块，让 Apache 处理静态 Html，而 Tomcat 处理 Jsp 和 Servlet。这种集成只需要修改一下 Apache 和 Tomcat 的配置文件即可。

Tomcat 的官方网站: <http://tomcat.apache.org>

第 1.2.4 节 IBM WebSphere

WebSphere Application Server 是一种功能完善、开放的 Web 应用程序服务器，是 IBM 电子商务计划的核心部分。基于 Java 和 Servlets 的 Web 应用程序运行环境，包含了

第 1 章 Nginx 简介

为 Web 站点提供服务所需的一切，运行时可以协同并扩展 Apache、Netscape、IIS 和 IBM 的 HTTPWeb 服务器，因此可以成为强大的 Web 应用服务器。

WebSphere 的官方网站: <http://www.ibm.com>

第 1.2.5 节 Windows IIS

IIS 是 Internet Information Server 的缩写，它是微软公司主推的服务器。IIS 与 Window NT Server 完全集成在一起，因而用户能够利用 Windows NT Server 和 NTFS (NT File System , NT 的文件系统) 内置的安全特性，建立强大，灵活而安全的 Internet 和 Intranet 站点。IIS 支持 HTTP (Hypertext Transfer Protocol , 超文本传输协议) ， FTP (File Transfer Protocol , 文件传输协议) 以及 SMTP 协议，通过使用 CGI 和 ISAPI ， IIS 可以得到高度的扩展。

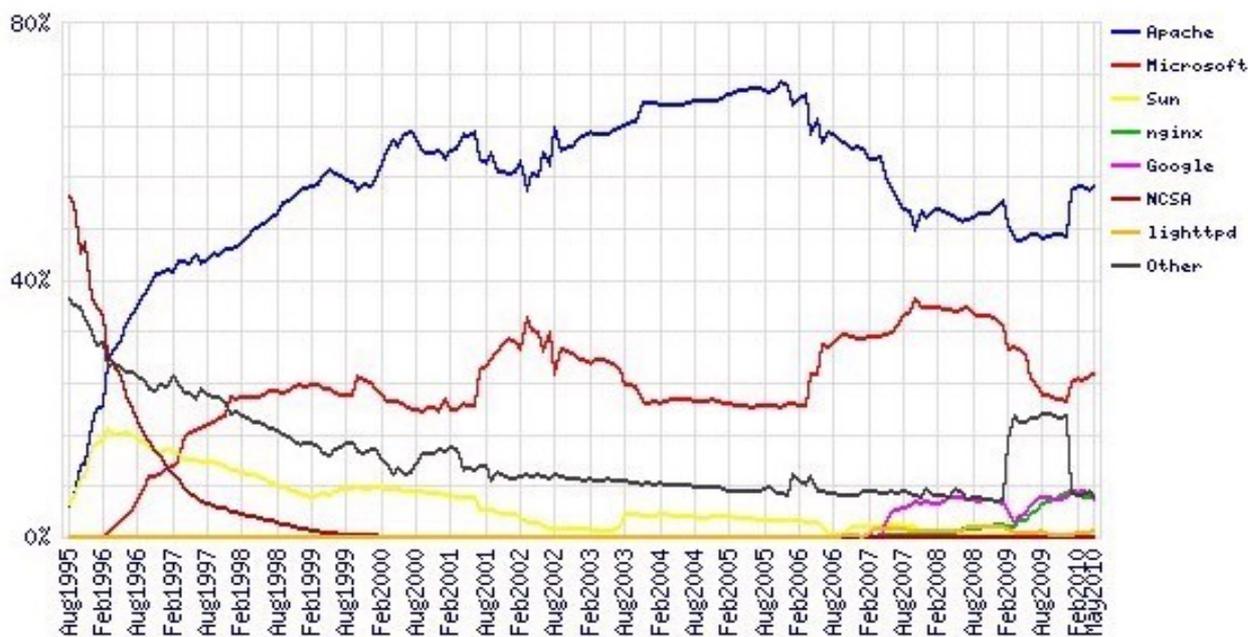
IIS 的官方网站: <http://www.iis.net>

第 1.2.6 节 Nginx 的历史

Nginx 是俄罗斯人编写的十分轻量级的 HTTP 服务器,Nginx ，它的发音为 “engine X”，是一个高性能的 HTTP 和反向代理服务器，同时也是一个 IMAP/POP3/SMTP 代理服务器。Nginx 是由俄罗斯人 Igor Sysoev 为俄罗斯访问量第二的 Rambler.ru 站点开发的，它已经在该站点运行超过两年半了。Igor Sysoev 在建立的项目时，使用基于 BSD 许可。自 Nginx 发布四年来，Nginx 已经因为它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。

在俄罗斯许多大网站都已经使用它，且一直表现不凡。截至 2007 年 4 月，俄罗斯大约有 20% 左右的虚拟主机是由 nginx 服务或代理的。Google 在线安全博客中统计 Nginx 服务或代理了大约所有 Internet 虚拟主机的 4%。而 Netcraft 的统计显示，Nginx 服务的主机在过去的一年里以四倍的速度增长并且在这几年里，它的排名还在不断上升，下图为 Netcraft 截止至 2010 年 5 月的统计。

Market Share for Top Servers Across All Domains
August 1995 - May 2010



Developer	April 2010	Percent	May 2010	Percent	Change
Apache	110,752,854	53.93%	112,663,533	54.68%	0.75
Microsoft	51,284,570	24.97%	52,062,154	25.27%	0.30
nginx	12,977,486	6.32%	13,490,726	6.55%	0.23
Google	13,749,829	6.70%	12,357,212	6.00%	-0.70
lighttpd	1,078,403	0.53%	1,869,658	0.91%	0.38

第1.3节 为什么要用 Nginx

Nginx 专为性能优化而开发，性能是其最重要的考量,实现上非常注重效率。它支持内核 Poll 模型，能经受高负载的考验,有报告表明能支持高达 50,000 个并发连接数。

Nginx 具有很高的稳定性，其它 HTTP 服务器当遇到访问的峰值，或者有人恶意发起慢速连接时，也很可能会导致服务器物理内存耗尽频繁交换，失去响应只能重启服务器。例如当前 Apache 一旦上到 200 个以上进程，web 响应速度就明显非常缓慢了。而 Nginx 采取了分阶段资源分配技术，使得它的 CPU 与内存占用率非常低。Nginx 官方表示保持 10,000 个没有活动的连接，它只占 2.5M 内存，所以类似 DOS 这样的攻击对 Nginx 来说

第 1 章 Nginx 简介

基本上是毫无用处的。就稳定性而言，nginx 比 lighttpd 更胜一筹。

Nginx 支持热部署，它的启动特别容易，并且几乎可以做到 7*24 不间断运行，即使运行数月也不需要重新启动。你还能够在不间断服务的情况下，对软件版本进行进行升级。

Nginx 采用 master-slave 模型，能够充分利用 SMP 的优势，且能够减少工作进程在磁盘 I/O 的阻塞延迟。

Nginx 代码质量非常高，代码很规范，手法成熟，模块扩展也很容易。

Nginx 采用了一些 os 提供的最新特性如对 sendfile (Linux2.2+)，accept-filter(FreeBSD4.1+)，TCP_DEFER_ACCEPT (Linux 2.4+)的支持，从而大大提高了性能。

当然，nginx 还很年轻，多多少少存在一些问题，比如：Nginx 是俄罗斯人创建，目前文档方面还不是很完善。因为文档大多是俄语，所以文档方面这也是个障碍。尽管 Nginx 的模块比较多，但它们还不够完善。对脚本的支持力度不够。

这些问题，Nginx 的作者和社区都在努力解决，我们有理由相信 nginx 将继续以高速增长来分享轻量级 HTTP 服务器市场，会有一个更美好的未来。

第1.4节 获取 Nginx

你可以从 Nginx 的官方主页上下载到 Nginx 的各个版本 <http://nginx.net>

其中最新版本为 Nginx 的开发版本，之前的版本为当前稳定版本。例如最新版本为 0.8.X，则当前稳定版本为 0.7.X。你还可以在官方 <http://nginx.org/en/docs> 找到一些帮助文档，另外 <http://wiki.nginx.org> 也是一个不错的选择。

第1.5节 课后练习

1. 简述常见市场常见 WEB 服务器及他们的区别。
2. 简述 Nginx 的优点。
3. 从官方下载 Nginx 源代码。

第2章 Nginx 的安装和启动

第2.1节 Nginx 的安装

Linux 下安装一个软件有三种方式，这边我们以源代码编译安装为主。

首先你需要从官方下载适合你的 Nginx 版本，<http://nginx.org/en/download.html>。本书将以 Nginx 7.67 为例讲解。

安装之前请你将你原来的 httpd 服务停止并卸载

```
[root@example tmp]# service httpd stop  
[root@example tmp]# rpm -e httpd --nodeps
```

出于管理和安全的目的考虑，也许你希望使用一个指定的普通用户身份去运行你的 web 服务器，所以我们首先会添加一个普通用户帐号用于运行我们的 Nginx

```
[root@example ~]# groupadd nginx  
[root@example ~]# useradd nginx
```

然后解压并编译安装你的 Nginx

```
[root@example tmp]# tar xf nginx-0.8.46.tar.gz  
[root@example tmp]# cd nginx-0.8.46  
[root@example nginx-0.8.46]# ./configure --user=nginx --group=nginx  
--prefix=/usr/local/nginx/ --with-http_stub_status_module --with-http_ssl_module  
[root@example nginx-0.8.46]# make  
[root@example nginx-0.8.46]# make install
```

安装过程还是比较简单的，./configure 过程可能会报一依赖关系，请自行解决依赖关系。下面我们来看看./configure 后面几个常用参数

第 2 章 Nginx 的安装和启动

```
--prefix= <dir>      指定安装主目录，默认为/usr/local/nginx
--user= <user>       指定用户身份，如果没有指定则默认使用 nobody
--group= <group>    指定组身份
--with-http_ssl_module  启用 https 支持
```

更多的参数请参考./configure --help

第2.2节 2.2 Nginx 的启动、重启与停止

安装完后我们现在可以启动 Nginx 了，我们可以进入到 Nginx 的主目录

```
[root@example nginx-0.7.67]# /usr/local/nginx/sbin/nginx -c
/usr/local/nginx/conf/nginx.conf
```

-c 是用来指定 Nginx 的主配置文件，如果没有指定则默认为 /usr/local/nginx/conf/nginx.conf

启动后可以使用 ps 和 netstat 命令来查看是否启动成功

```
[root@example nginx-0.8.46]# ps -ef | grep nginx

root    2187    1  0 14:33 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/nginx
nginx   2194   2187  0 14:34 ?        00:00:00 nginx: worker process
root    2197   2508  0 14:34 pts/0    00:00:00 grep nginx

[root@example nginx-0.8.46]# netstat -ntlp | grep 80

tcp     0      0 0.0.0.0:80          0.0.0.0:*           LISTEN   2187/nginx
```

然后我们来测试一下主页是否可以访问

```
[root@example ~]# links 127.0.0.1
```

如果显示 Welcome to nginx! 则说明服务正常

Nginx 并没有为我们提供类似 System V 服务的管理脚本，所以如果你希望它在开机的时候 Nginx 可以自动启动，可以执行以下命令。

```
[root@example ~]# echo "/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf" >> /etc/rc.local
```

当然，如果你对 System V 的服务管理脚本情有独钟的话，可以参考如下脚本

```
#!/bin/sh

#

# nginx - this script starts and stops the nginx daemon

#

# chkconfig: - 85 15

# description: Nginx is an HTTP(S) server, HTTP(S) reverse \

#               proxy and IMAP/POP3 proxy server

# processname: nginx

# config:      /etc/nginx/nginx.conf

# config:      /etc/sysconfig/nginx

# pidfile:     /var/run/nginx.pid

# Source function library.

. /etc/rc.d/init.d/functions

# Source networking configuration.
```

第 2 章 Nginx 的安装和启动

```
./etc/sysconfig/network

# Check that networking is up.
[ "$NETWORKING" = "no" ] && exit 0

nginx="/usr/local/nginx/sbin/nginx"
prog=$(basename $nginx)

NGINX_CONF_FILE="/usr/local/nginx/conf/nginx.conf"

[ -f /etc/sysconfig/nginx ] && . /etc/sysconfig/nginx

lockfile=/var/lock/subsys/nginx

start() {
    [ -x $nginx ] || exit 5
    [ -f $NGINX_CONF_FILE ] || exit 6
    echo -n "Starting $prog: "
    daemon $nginx -c $NGINX_CONF_FILE
    retval=$?
    echo
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}
```

```
}

stop() {

    echo -n $"Stopping $prog: "

    killproc $prog -QUIT

    retval=$?

    echo

    [ $retval -eq 0 ] && rm -f $lockfile

    return $retval

    killall -9 nginx

}

restart() {

    configtest || return $?

    stop

    sleep 1

    start

}

reload() {

    configtest || return $?

    echo -n $"Reloading $prog: "

    killproc $nginx -HUP
```

第 2 章 Nginx 的安装和启动

```
RETVAL=$?  
  
echo  
  
}  
  
force_reload() {  
  
    restart  
  
}  
  
configtest() {  
  
    $nginx -t -c $NGINX_CONF_FILE  
  
}  
  
rh_status() {  
  
    status $prog  
  
}  
  
rh_status_q() {  
  
    rh_status >/dev/null 2>&1  
  
}  
  
case "$1" in  
  
    start)  
  
        rh_status_q && exit 0
```

```
$1
;;
stop)
    rh_status_q || exit 0
$1
;;
restart|configtest)
    $1
    ;;
reload)
    rh_status_q || exit 7
$1
    ;;
force-reload)
    force_reload
    ;;
status)
    rh_status
    ;;
condrestart|try-restart)
    rh_status_q || exit 0
    ;;
*)
```

第 2 章 Nginx 的安装和启动

```
    echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload|
configtest}"

    exit 2

esac
```

脚本生成完了不要忘了给一个执行权限，然后使用 `chkconfig` 命令对其进行管理

```
[root@example ~]# cp nginx /etc/init.d/

[root@example ~]# chmod 755 /etc/init.d/nginx

[root@example ~]# chkconfig --add nginx
```

当我们对 Nginx 的配置文件做过一些更改后，希望在不中断当前服务的情况下，进行一个平滑的重启，可以使用如下命令。

```
[root@example ~]# service nginx reload
```

脚本中的 `reload` 函数会首先对配置文件做一个语法格式的检查，使用的是如下的命令

```
[root@example init.d]# /usr/local/nginx/sbin/nginx -t -c /usr/local/nginx/conf/nginx.conf
```

当语法格式检查通过后，会对 Nginx 发出一个标记为 1 或者说是 HUP 的信号源，Nginx 收到后会关闭旧进程，打开新进程，如果有进程正在为一个用户提供服务，则会等待这次服务结束。

当然我们也可以使用 `service nginx restart` 的方式去重启服务。它会先尝试以比较温柔的方式去结束进程，如果有进程没有被结束掉，那么 `killall -9 nginx`。

关于停止 Nginx，直接 `service nginx stop` 即可，或者 `kill` 死所有的 Nginx 进程。

第2.3节 课后练习

1. 源代码编译 Nginx。能够自行搭建 Nginx 环境。

从服务器上下载源代码。

解压并进行编译安装

启动 Nginx 服务并测试

2. 创建 Nginx 服务管理脚本。能够使用脚本简化 Nginx 服务管理。

在/etc/init.d/目录下创建 Nginx 服务脚本

修改权限

加入 chkconfig 管理列表

使用 service 命令进行重启测试

第3章 Nginx 的基本配置

第3.1节 Nginx 主配置概述

Linux 下基本上每个服务都会有它的主配置文件，该文件会定义服务应该如果去运行，使用些什么参数，启用些什么功能，相关会涉及到的一些操作文件在哪，所以主配置文件对服务是至关重要的。下面我们来分析一下 Nginx 的主配置文件。

Nginx 的主配置文件默认情况下位于 `/usr/local/nginx/conf/nginx.conf` 以下为 Nginx 配置文件一些参数的注释。

```
#user nobody;
#指定使用的用户

worker_processes 1;
# 开启的进程数，一般设置 1-5

#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;
# 定义错误日志，以及记录的日志等级

#pid logs/nginx.pid;
# 定义 pid 文件位置

events {
```

```
# use [ kqueue | rtsig | epoll | /dev/poll | select | poll ];

# use epoll; #使用 epoll ( linux2.6 的高性能方式 )

#Nginx 支持如下处理连接的方法 ( I/O 复用方法 ) , 这些方法可以通过 use 指令指定。

#select - 标准方法。 如果当前平台没有更有效的方法 , 它是编译时默认的方法。 你可以使用配置参数
-with-select_module 和 -without-select_module 来启用或禁用这个模块。

#poll - 标准方法。 如果当前平台没有更有效的方法 , 它是编译时默认的方法。 你可以使用配置参数
-with-poll_module 和 -without-poll_module 来启用或禁用这个模块。

#kqueue - 高效的方法 , 使用于 FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 和 MacOS X. 使用双
处理器的 MacOS X 系统使用 kqueue 可能会造成内核崩溃。

#epoll - 高效的方法 , 使用于 Linux 内核 2.6 版本及以后的系统。 在某些发行版本中 , 如 SuSE 8.2,
有让 2.4 版本的内核支持 epoll 的补丁。

#rtsig - 可执行的实时信号 , 使用于 Linux 内核版本 2.2.19 以后的系统。 可是从 Linux 内核版本
2.6.6-mm2 开始 , 这个参数就不再使用了。

#/dev/poll - 高效的方法 , 使用于 Solaris 7 11/99+, HP/UX 11.22+ (eventport), IRIX 6.5.15+ 和
Tru64 UNIX 5.1A+。

#eventport - 高效的方法 , 使用于 Solaris 10. 为了防止出现内核崩溃的问题 , 有必要安装 这个 安全
补丁。

    worker_connections 1024;

# worker_connections 51200; #每个进程最大连接数 ( 最大连接=连接数 x 进程数 )
}

http {

    include    mime.types;

#文件扩展名与文件类型映射表

    default_type application/octet-stream;

#默认文件类型
```

第3章 Nginx 的基本配置

```
#log_format main '$remote_addr - $remote_user [$time_local] "$request" '
#           '$status $body_bytes_sent "$http_referer" '
#           '"$http_user_agent" "$http_x_forwarded_for";

#access_log logs/access.log main;

sendfile    on;

#开启高效文件传输模式

#tcp_nopush  on;

#该选项用于防止网络阻塞

#keepalive_timeout 0;

keepalive_timeout 65;

##长链接超时时间

#gzip on;

#打开 gzip 压缩

#fastcgi_connect_timeout 300;

#fastcgi_send_timeout 300;

#fastcgi_read_timeout 300;

#fastcgi_buffer_size 128k;

#fastcgi_buffers 4 256k;

#fastcgi_busy_buffers_size 256k;

#fastcgi_temp_file_write_size 256k;
```

```
#fastcgi_temp_path /dev/shm;

#fastcgi 连接超时时间和缓存

server {

    listen    80;

    server_name localhost;

#主机名

    #charset koi8-r;

#默认字符编码 charset gb2312

    #access_log logs/host.access.log main;

    location / {

# pass 路径匹配 能够匹配路径中带 "/" 的 不过需要注意的是如果之后也有相关 "/" 匹配项并且
# 使用比此处更精确匹配符，则以之后表达式优先

        root html;

        index index.html index.htm;

    }

    #error_page 404          /404.html;

    # redirect server error pages to the static page /50x.html

    #
```

第3章 Nginx 的基本配置

```
error_page 500 502 503 504 /50x.html;

location = /50x.html {
#精确的匹配，并且不再向下匹配

    root html;

}

#

#location ~ \.php$ {
#正则表达式匹配 一旦匹配则不再向下匹配

# proxy_pass http://127.0.0.1;

#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

#

#location ~ \.php$ {

# root html;

# fastcgi_pass 127.0.0.1:9000;
#指定 fastcgi 的地址端口

# fastcgi_index index.php;

# fastcgi_param SCRIPT_FILENAME /scripts$fastcgi_script_name;

# include fastcgi_params;

#}
```

```
# deny access to .htaccess files, if Apache's document root
# concurs with nginx's one
#
#location ~ /\.ht {
#   deny all;
# 不允许访问以.ht 开头的文件
#}
}

# another virtual host using mix of IP-, name-, and port-based configuration
#
#server {
#   listen    8000;
#   listen    somename:8080;
#   server_name somename alias another.alias;

#   location / {
#       root   html;
#       index index.html index.htm;
#   }
#}
#}
```

第3章 Nginx 的基本配置

```
#以上在配置虚拟主机

# HTTPS server

#

#server {

#   listen    443;

#   server_name localhost;

#   ssl        on;

#   ssl_certificate    cert.pem;

#   ssl_certificate_key    cert.key;

#   ssl_session_timeout    5m;

#   ssl_protocols    SSLv2 SSLv3 TLSv1;

#   ssl_ciphers    ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;

#   ssl_prefer_server_ciphers    on;

#以上为 ssl 配置

#   location / {

#       root    html;

#       index    index.html index.htm;

#   }

#}
```

```
}

```

第3.2节 Nginx 下的虚拟主机配置

利用虚拟主机技术，可以把一台真正的主机分成许多"虚拟"的主机，每一台虚拟主机都具有独立的域名和 IP 地址，具有完整的 Internet 服务器 (www, FTP, email) 功能。虚拟主机之间完全独立，在外界看来，每一台虚拟主机和一台独立的主机完全一样。效果一样但费用却大不一样了。由于多台虚拟主机共享一台真实主机的资源，每个虚拟主机用户承受的硬件费用、网络维护费用、通信线路的费用均大幅度降低，Internet 真正成为人人用得起的网络！

虚拟主机共分为三种：基于 IP 的虚拟主机，基于端口的虚拟主机和基于名称的虚拟主机。前两种由于受到成本和客户使用习惯的限制，相对使用的没有基于名称的虚拟主机多，下面我们介绍一下三种虚拟主机的配置。

Nginx 基于名称的虚拟主机配置：

```
server {
    listen 80;

    server_name www.example.com;

    ...
}

server {
    listen 80;

    server_name www.test.com;

    ...
}
```

第3章 Nginx 的基本配置

```
server {  
  
    listen 80 default;  
  
    ...  
  
}
```

上述配置中，定义了三个虚拟主机。前两个 server，通过域名“www.example.com”和“www.test.com”可以分别访问正确的网站。如果浏览器直接通过 IP 地址或者其他指向这台机器的域名访问，那么访问到的是第三个 server 配置。第三个 server 为一个默认配置，请注意它没有“server_name”指令，并且“listen”指令包含一个“default”关键字。

Nginx 基于 IP 的虚拟主机

```
[root@server nginx]# vi /usr/local/nginx/conf/nginx.conf  
  
server {  
  
    listen 10.0.0.88:80;  
  
    root 88.com;  
  
    index index.html;  
  
}  
  
server {  
  
    listen 10.0.0.87:80;  
  
    root 87.com;  
  
    index index.html;  
  
}  
  
[root@server nginx]# ifconfig eth0:0 10.0.0.88  
  
[root@server nginx]# ifconfig eth0:1 10.0.0.87
```

```
[root@server nginx]# mkdir /usr/local/nginx/88.com

[root@server nginx]# echo 'I am your father' > /usr/local/nginx/88.com/index.html

[root@server nginx]# mkdir /usr/local/nginx/87.com

[root@server nginx]# echo 'this is 87.com' > /usr/local/nginx/87.com/index.html
```

以上配置了两台虚拟主机，一台 IP 为 10.0.0.88，另一台为 10.0.0.87。它们都监听 80 端口。根据访问的 IP 地址不同，返回不同网站内容。

Nginx 基于端口的虚拟主机

```
[root@server nginx]# vi /usr/local/nginx/conf/nginx.conf

server {

    listen 80;

    root 80.com;

}

server {

    listen 8080;

    root 8080.com;

}

[root@server nginx]# mkdir /usr/local/nginx/80.com

[root@server nginx]# mkdir /usr/local/nginx/8080.com

[root@server nginx]# echo 'this is 80.com' > /usr/local/nginx/80.com/index.html

[root@server nginx]# echo 'this is 8080.com' > /usr/local/nginx/8080.com/index.html
```

以上配置了两台虚拟主机，一台使用相同 IP。一台使用 80 端口，另一台使用 8080 端口。访问 8080 端口时需要在 URL 后加上 :8080。

第3.3节 安全的连接 https

众所周知，我们在互联网上冲浪，一般都是使用的 http 协议（超文本传输协议），默认情况下数据是明文传送的，这些数据在传输过程中都可能会被捕获和窃听，因此是不安全的。https 可以说是 http 协议的安全版，就是为了满足对安全性要求比较高的用户而设计的。

如果您的邮件中有敏感数据，不希望被人窃听；如果您不希望被钓鱼网站盗用帐号信息，如果您希望您在使用邮箱的过程中更安全，那么我们推荐您使用 https 安全连接。

Nginx 的 https 配置方法：

首先生成一对证书。

你需要使用到 openssl 命令，所以请确认系统已经安装过此包

RHEL5 中在 /etc/pki/tls/certs 目录有个脚本可以帮助我们简化证书生成的过程，所以我们首先切换到此目录

```
[root@example ~]# cd /etc/pki/tls/certs
[root@example certs]# cd /etc/pki/tls/certs
[root@example certs]# make server.key
```

生成私钥

```
[root@example certs]# openssl rsa -in server.key -out server.key
```

去除密码以便使用时不询问密码

```
[root@example certs]# make server.csr
```

生成证书颁发机构，用于颁发公钥

```
[root@example certs]# openssl x509 -in server.csr -req -signkey server.key -days 365 -out server.crt
```

颁发公钥，不过由于我们并不是去 CA 证书中心申请的公钥，所以在使用的時候，客户端浏览器会跳出未受信任的警告。如果你 money 够多，请去 CA 申请。

```

server {

    listen    443;

    server_name www.example.com;

    ssl      on;

    ssl_certificate    /etc/pki/tls/certs/server.crt;

    ssl_certificate_key /etc/pki/tls/certs/server.key;

    # 以上两句在指定证书所存放的路径

    ssl_session_timeout 5m;

    ssl_protocols SSLv2 SSLv3 TLSv1;

    ssl_ciphers ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;

    #指出允许的密码，密码指定为 OpenSSL 支持的格式

    ssl_prefer_server_ciphers on;

    #依赖 SSLv3 和 TLSv1 协议的服务器密码将优先于客户端密码

    ...

}
  
```

第3.4节 Nginx 日志管理

做为一个优秀的管理员，应该能够对你所维护的服务器了如指掌，那么日志的阅读就是必不可少的，我们可以通过日志，了解到你的服务器是否正常运行，有谁在访问你的服务器，每天的访问量是多少，有没有非法的不受欢迎的访问等等。

在 Nginx 主配置中有关 Nginx 日志的相关参数主要有两条。

```

log_format    指定日志格式

access_log    指定日志存放路径
  
```

第3章 Nginx 的基本配置

对于一个日均 PV 数十万以上的网站来说，日志的增长将会很迅速，一天的日志几个 G 属于正常情况。所以为了我们单个日志文件不至于太大，我们需要定期的去截断我们的日志。不可很可惜，Nginx 默认并不支持我们所熟悉的 cronolog 方式来对每天的日志进行截断，那么我们可以通过以下方法来解决。

```
[root@example ~]# mv /usr/local/nginx/logs/access.log /usr/local/nginx/logs/access.log1
[root@example ~]# touch /usr/local/nginx/logs/access.log
[root@example ~]# kill -HUP `cat /usr/local/nginx/logs/nginx.pid`
```

然后以 crond 的方式或者 logrotate 的方式运行。

对于习惯使用 cronolog 来做日志分割的管理人员来说，无法使用 cronolog 总会成为他心里挥之不去的痛。然而有时候事情总会很奇妙，只要你想去解决，就一定有办法解决。我在这里介绍一种通过管道符文件另辟溪径的方法使用 cronolog。

首先下载并编译安装我们 cronolog

```
[root@example ~]# tar xf cronolog-1.6.2.tar.gz
[root@example ~]# ./configure
[root@example ~]# make && make install
[root@example ~]# service nginx stop
[root@example ~]# mv /usr/local/nginx/logs/access.log
/usr/local/nginx/logs/access.log.bak
[root@example ~]# mkfifo /usr/local/nginx/logs/access.log
[root@example ~]# cronolog /usr/local/nginx/logs/access.log.%Y%m%d <
/usr/local/nginx/logs/access.log &
[root@example ~]# service nginx start
```

访问一下自己的主机，然后查看 /usr/local/nginx/logs/ 目录下是否多出个 access.log.%Y%m%d 文件，查看一下这个文件里的内容是否是我们的访问日志。

这里需要注意的是必需在 Nginx 启动之前执行 cronolog 命令，否则 Nginx 将无法正
常启动。

对于一些你不需要记录的日志内容，我们可以使用以下方式关闭对这些文件的日志记录。

```
location ~ .*\. (js|jpg|JPG|jpeg|JPEG|css|bmp|gif|GIF)$  
  
    {  
  
        access_log off;  
  
    }
```

第3.5节 访问控制

有时我们会有这么一种需求，就是你的网站并不想提供一个公共的访问或者某些页面不
希望公开，我们希望的是某些特定的客户端可以访问。那么我们可以在访问时要求进行身份
认证，就如给你自己的家门加一把锁，以拒绝那些不速之客。我们在服务课程中学习过
apache 的访问控制，对于 Nginx 来说同样可以实现，并且整个过程和 Apache 非常的相似。

```
location / {  
  
    root html;  
  
    index index.html index.htm;  
  
    auth_basic "xxxx";  
  
    auth_basic_user_file /usr/local/nginx/passwd.db;  
  
}
```

设置密码

```
[root@example nginx]# htpasswd -c /usr/local/nginx/passwd.db mark  
  
New password: (输入密码)
```

第3章 Nginx 的基本配置

```
Re-type new password: ( again )
```

```
Adding password for user mark
```

然后访问一下页面测试一下。

另外如果你想阻止别人访问你某些目录下的特定文件，比如我网站主目录下有个 test 目录，我不想让别人访问我 test 目录下的“.txt”和“.doc”的文件，那么我们可以通过 deny 的方式来做拒绝。

```
location ~* \.(txt|doc)$ {  
    root html/test;  
    deny all;  
}
```

此处 ~* 代表不区分大小写方式匹配。

第3.6节 课后练习

1. 配置两个基于名称的虚拟主机 www.test.com 和 www.example.com（注意，做此实验之前你应该已经有正确的 DNS 配置）。
修改 Nginx 主配置文件，增加两个 server 字段。
创建相关网站主目录并重启 Nginx 服务。
2. 配置一个 https。以使你的服务器可以提供安全的连接。
生成安全密钥。
修改 Nginx 主配置文件，加入 ssl 相关配置
重启 Nginx 服务。
3. 配置你的虚拟主机（www.test.com），当别人试图访问时，要求进行身份认证，允许 mark 用户可以通过密码 uplooking 进行访问。
修改 Nginx 主配置文件，加入 auth 相关配置。

创建认证使用密码文件

重启 Nginx 服务。

第4章 Rewrite 规则

第4.1节 什么是 Rewrite

Rewrite 对称 URL Rewrite，即 URL 重写，就是把传入 Web 的请求重定向到其他 URL 的过程。URL Rewrite 最常见的应用是 URL 伪静态化，是将动态页面显示为静态页面方式的一种技术。比如 `http://www.123.com/news/index.asp?id=123` 使用 URLRewrite 转换后可以显示为 `http://www.123.com/news/123.html`

对于追求完美主义的网站设计师，就算是网页的地址也希望看起来尽量简洁明快。形如 `http://www.123.com/news/index.asp?id=123` 的网页地址，自然是毫无美感可言，而用 UrlRewrite 技术，你可以轻松把它显示为 `http://www.123.com/news/123.html`。

理论上，搜索引擎更喜欢静态页面形式的网页，搜索引擎对静态页面的评分一般要高于动态页面。所以，UrlRewrite 可以让我们网站的网页更容易被搜索引擎所收录。

从安全角度上讲，如果在 url 中暴露太多的参数，无疑会造成一定量的信息泄漏，可能会被一些黑客利用，对你的系统造成一定的破坏，所以静态化的 url 地址可以给我们带来更高的安全性。

第4.2节 Rewrite 相关指令

Nginx Rewrite 相关指令有 if、rewrite、set、return 等。

if 的语法 应用于 server 和 location 环境内

```
if (condition) { ... }
```

if 可以支持如下条件判断匹配符号

~ 为区分大小写匹配

~* 为不区分大小写匹配

!~和!~*分别为区分大小写不匹配及不区分大小写不匹配

-f 和!-f 用来判断是否存在文件

-d 和!-d 用来判断是否存在目录

-e 和!-e 用来判断是否存在文件或目录

-x 和!-x 用来判断文件是否可执行

在匹配过程中可以引用一些 Nginx 的全局变量，更多的变量请参考 <http://wiki.nginx.org/NginxHttpCoreModule> 的 Variables 部分

\$args, 请求中的参数;

\$document_root, 针对当前请求的根路径设置值;

\$host, 请求信息中的"Host", 如果请求中没有 Host 行, 则等于设置的服务器名;

\$limit_rate, 对连接速率的限制;

\$request_method, 请求的方法, 比如"GET"、"POST"等;

\$remote_addr, 客户端地址;

\$remote_port, 客户端端口号;

\$remote_user, 客户端用户名, 认证用;

\$request_filename, 当前请求的文件路径名

\$query_string, 与\$args 相同;

\$scheme, 所用的协议, 比如 http 或者是 https

\$server_protocol, 请求的协议版本, "HTTP/1.0"或"HTTP/1.1";

\$server_addr, 服务器地址, 如果没有用 listen 指明服务器地址, 使用这个变量将发起一次系统调用以取得地址(造成资源浪费);

\$server_name, 请求到达的服务器名;

\$document_uri 与\$uri 一样, URI 地址;

\$server_port, 请求到达的服务器端口号;

第 4 章 Rewrite 规则

看了一大堆也许会有点晕，当然有时晕着晕着也就习惯了。不过我们还是先来看两个例子。这样更有助于理解。

例 4-1 匹配访问的 url 地址是否是个目录

```
if (-d $request_filename) {  
    ...;  
}
```

例 4-2 匹配访问的地址是否以 www 开头

```
if ($hosts ~* ^www) {  
    ...;  
}
```

rewrite 指令根据表达式来重定向 URI，或者修改字符串。可以应用于 server, location, if 环境下 每行 rewrite 指令最后应该跟一个 flag 标记，支持的 flag 标记有

last	相当于 Apache 里的[L]标记，表示完成 rewrite
break	本条规则匹配完成后，终止匹配，不再匹配后面的规则
redirect	返回 302 临时重定向，浏览器地址会显示跳转后的 URL 地址
permanent	返回 301 永久重定向，浏览器地址会显示跳转后 URL 地址

last 和 break 标记的区别在于，last 标记在本条 rewrite 规则执行完后，会对其所在的 server { ... } 标签重新发起请求，而 break 标记则在本条规则匹配完成后，停止匹配，不再做后续的匹配。另有些时候必须使用 last，比如在使用 alias 指令时，而使用 proxy_pass 指令时则必须使用 break。

例 4-3：以下这段 rewrite 会导致死循环

```
location /abc/ {
```

```
rewrite "^/abc/(.*)\.html$" /abc/index.html last;

}
```

我们应该将上面的 last 改成 break 以避免死循环。

redirect 和 permanent 区别则是返回的不同方式的重定向，对于客户端来说一般状态下是没有区别的。而对于搜索引擎，相对来说 301 的重定向更加友好，如果我们把一个地址采用 301 跳转方式跳转的话，搜索引擎会把老地址的相关信息带到新地址，同时在搜索引擎索引库中彻底废弃掉原先的老地址。

使用 302 重定向时，搜索引擎(特别是 google)有时会查看跳转前后哪个网址更直观，然后决定显示哪个，如果它觉的跳转前的 URL 更好的话，也许地址栏不会更改，那么很有可能出现 URL 劫持的现象。

我们在做 URI 重写时，有时会发现 URI 中含有相关参数，如果需要将这些参数保存下来，并且在重写过程中重新引用，我们可以用到 () 和 \$N 的方式来解决。

例 4-4：匹配访问的 url 地址是否是个目录，如果是则自动加个 /

```
if (-d $request_filename) {

    rewrite ^/(.*)([^\/]$) http://$host/$1$2/ permanent;

}
```

例 4-5：用户访问的网址为 www.test.com/abc.html 重写后真实地址是 www.test.com/login.php?user=abc

```
location ~* /php/.*\.html$ {

    rewrite /php/(.*)\.html /login.php?user=$1 last;

}
```

例 4-6：用户访问地址为/uplook/11-22-33.html 重写后真实地址为 /uplook/11/22/33.html

```
location /uplook/ {
```

第4章 Rewrite 规则

```
rewrite /uplook/([0-9]+)-([0-9]+)-([0-9]+).html /uplook/$1/$2/$3.html last;  
}
```

set 指令是用于定义一个变量，并且赋值。应用于 server,location,if 环境。

语法格式为: set \$变量名 变量值

例 4-7：当访问任意目录下的 whoami.html 都重定向到 /who.html

```
location ~* .*/whoami\.html$ {  
    set $who 'who.html';  
    rewrite .*      /$who break;  
}
```

return 指令用于返回状态码给客户端，应用于 server，location，if 环境。

例 4-8：如果访问的 .sh 结尾的文件则返回 403 操作拒绝错误

```
location ~* .*\sh$ {  
    return 403;  
}
```

第4.3节 课后练习

1. 配置虚拟主机 *.test.com。当客户端访问任意 XXX.test.com 时都能匹配此主机并将地址重写成 www.test.com/XXX/（注意，实验之前你应该保证 XXX.test.com 都能正常解析）

修改主配置文件，配置虚拟主机 *.test.com

书写 if 语句，判断如果以 test.com 结尾的由进行 url 重写

将 test.com 之前的内容保存于一个变量 domain 中

将原地址重写成 `www.test.com/$domain/`

2. 当用户访问服务器 `http://www.test.com/page-uid-123.html` 时服务器自动将地址重写成 `http://www.test.com/page.php?uid=123` , 当访问 `http://www.test.com/page-username-123.html` 重写成 `http://www.test.com/page.php?username=123` 。其中 `123.html` 可能是任意文件名 `XXX.html`

配置主配置文件并书写 rewrite 规则

使用 location 语句。

在 Rewrite 中将 `username` 和 `uid` 字段保存下来, 并以变量 `$1` 方式引用。

将 `XXX.html` 字段保存下来, 并以变量 `$2` 方式引用。

3. 当用户访问服务器 `http://www.test.com/today-pic-112.html` 时服务器自动将地址重写成 `http://www.test.com/today/pic/112.html`。(注意, 此处 `today pic` 可以更换成其他任意字母, `112` 可以更换成其他任意数字)

配置主配置文件并书写 rewrite 规则

配置以 `html` 结尾的文件做 rewrite

将 `today pic 112` 字段分别保存下来并以 `$1 $2 $3` 方式引用

第5章 Nginx 反向代理

第5.1节 反向代理及其特点

反向代理 (Reverse Proxy) 方式是指以代理服务器来接受 internet 上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给 internet 上请求连接的客户端，此时代理服务器对外就表现为一个服务器。

反向代理又称为 Web 服务器加速，是针对 Web 服务器提供加速功能的。它作为代理 Cache，但并不针对浏览器用户，而针对一台或多台特定 Web 服务器（这也是反向代理名称的由来）。代理服务器可以缓存一些 web 的页面，降低了 web 服务器的访问量，所以可以降低 web 服务器的负载。web 服务器同时处理的请求数少了，响应时间自然就快了。同时代理服务器也存了一些页面，可以直接返回给客户端，加速客户端浏览。实施反向代理，只要将反向代理设备放置在一台或多台 Web 服务器前端即可。当互联网用户访问某个 WEB 服务器时，通过 DNS 服务器解析后的 IP 地址是代理服务器的 IP 地址，而非原始 Web 服务器的 IP 地址，这时代理服务器设备充当 Web 服务器，浏览器可以与它连接，无需再直接与 Web 服务器相连。因此，大量 Web 服务工作量被转载到反向代理服务上。不但能够很大程度上减轻 web 服务器的负担，提高访问速度，而且能够防止外部网主机直接和 web 服务器直接通信带来的安全隐患。

第5.2节 Nginx 反向代理

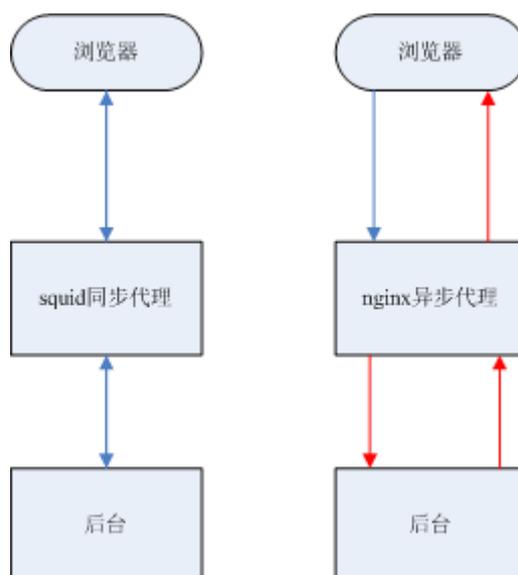
Nginx proxy 是 Nginx 的王牌功能，利用 proxy 基本可以实现一个完整的 7 层负载均衡，它有这些特色：

1. 功能强大，性能卓越，运行稳定。
2. 配置简单灵活。
3. 能够自动剔除工作不正常的后端服务器。
4. 上传文件使用异步模式。

5. 支持多种分配策略，可以分配权重，分配方式灵活。

在 nginx 的反向代理优点介绍中，提到了异步传输模式 下面就来讲解下传统的代理 (apache/squid) 的同步传输和 nginx 的异步传输的差异。

看图：



squid 同步传输：浏览器发起请求，而后请求会立刻被转到后台，于是在浏览器和后台之间就建立了一个通道。在请求发起直到请求完成，这条通道都是一直存在的。

nginx 异步传输：浏览器发起请求，请求不会立刻转到后台，而是将请求数据 (header) 先收到 nginx 上，然后 nginx 再把这个请求发到后端，后端处理完之后把数据返回到 nginx 上，nginx 将数据流发到浏览器，这点和 lighttpd 有点不同，lighttpd 是将后端数据完全接收后才发送到浏览器。

那么这到底有什么好处呢？

1) 假设用户执行一个上传文件操作，因为用户网速又比较慢，因此需要花半个小时才能把文件传到服务器。squid 的同步代理在用户开始上传后就和后台建立了连接，半小时后文件上传结束，由此可见，后台服务器连接保持了半个小时；而 nginx 异步代理就是先将此文件收到 nginx 上，因此仅仅是 nginx 和用户保持了半小时连接，后台服务器在这半小时之内没有为这个请求开启连接，半小时后用户上传结束，nginx 才将上传内容发到后台，nginx 和后台之间的带宽 是很充裕的，所以只花了一秒钟就将请求发送到了后台，由此可

第 5 章 Nginx 反向代理

见，后台服务器连接保持了一秒。同步传输花了后台服务器半个小时，异步传输只花一秒，可见优化程度很大。

2) 在上面这个例子中，假如后台服务器因为种种原因重启了，上传文件就自然中断了，这对用户来说是非常恼火的一件事情，想必各位也有上传文件传到一半被中断的经历。用 nginx 代理之后，后台服务器的重启对用户上传的影响减少到了极点，而 nginx 是非常稳定的并不需要常去重启它，即使需要重启，利用 `kill -HUP` 就可以做到不间断重启 nginx。

3) 异步传输可以令负载均衡器更有保障，为什么这么说呢？在其它的均衡器（`lvs/haproxy/apache` 等）里，每个请求都是只有一次机会的，假如用户发起一个请求，结果该请求分到的后台服务器刚好挂掉了，那么这个请求就失败了；而 nginx 因为是异步的，所以这个请求可以重新发往下一个后台，下一个后台返回了正常的的数据，于是这个请求就能成功了。还是用用户上传文件这个例子，假如不但用了 nginx 代理，而且用了负载均衡，nginx 把上传文件发往其中一台后台，但这台服务器突然重启了，nginx 收到错误后，会将这个上传文件发到另一台后台，于是用户就不用再花半小时上传一遍。

4) 假如用户上传一个 10GB 大小的文件，而后台服务器没有考虑到这个情况，那么后台服务器岂不要崩溃了。用 nginx 就可以把这些东西都拦在 nginx 上，通过 nginx 的上传文件大小限制功能来限制，另外 nginx 性能非常有保障，就放心的让互联网上那些另类的用户和 nginx 对抗去吧。

第5.3节 Nginx 反向代理配置

修改 Nginx 主配置文件，在 http 全局配置部分加入如下内容：

```

http {
    ... ..

    client_max_body_size 300m;

    #允许客户端请求的最大单个文件字节数，它出现在请求头部的 Content-Length 字段。（可以更改此参数达到限制用户上传文件大小的目的）

    client_body_buffer_size 128k;
  
```

#这个指令可以指定连接请求使用的缓冲区大小，默认值：8k/16k。如果客户端请求一个文件大于 128k，则 Nginx 会尝试在硬盘上创建临时文件。如果硬盘满了，则会报错。

```
client_body_temp_path    /dev/shm/client_body_temp;
```

#这个指令指定连接请求试图写入缓存文件的目录路径。

```
proxy_connect_timeout    600;
```

#跟后端服务器连接的超时时间，发起握手等候响应超时时间

```
proxy_read_timeout      600;
```

#默认值：proxy_read_timeout 60。决定读取后端服务器应答的超时时间，它决定 nginx 将等待多久时间来取得一个请求的应答。超时时间是指完成了两次握手后并且状态为 established 的超时时间，而不是所有的应答时间。相对于 proxy_connect_timeout，这个时间可以扑捉到一台将你的连接放入连接池延迟处理并且没有数据传送的服务器，注意不要将此值设置太低，某些情况下代理服务器将花很长的时间来获得页面应答。如果被代理服务器在设置的时间内没有传递数据，nginx 将关闭连接。

```
proxy_send_timeout      600;
```

#设置代理服务器转发请求的超时时间，同样指完成两次握手后的时间，如果超过这个时间代理服务器没有数据转发到后端服务器，nginx 将关闭连接。

```
proxy_buffer_size       16k;
```

#默认值：proxy_buffer_size 4k/8k。设置从后端服务器读取的第一部分应答的缓冲区大小，通常情况下这部分应答中包含一个小的应答头。

```
proxy_buffers           4 32k;
```

#设置用于读取应答（来自后端服务器）的缓冲区数目和大小，告诉 Nginx 保存单个用的几个 Buffer，最大用多大空间

```
proxy_busy_buffers_size 64k;
```

#如果系统很忙的时候可以申请更大的 proxy_buffers，官方推荐*2

```
proxy_temp_file_write_size 64k;
```

#设置在写入 proxy_temp_path 时缓存临时文件数据的大小，在预防一个工作进程在传递文件时阻塞太长。

第 5 章 Nginx 反向代理

```
    proxy_temp_path    /dev/shm/proxy_temp;
```

#类似于 http 核心模块中的 client_body_temp_path 指令，指定一个目录来缓冲比较大的被代理请求。

```
    upstream    server_pool    {

        server 192.168.0.88:80 weight=4 max_fails=2 fail_timeout=30s;

        server 192.168.0.89:80 weight=2 max_fails=2 fail_timeout=30s;

    }
```

#HTTP 负载均衡模块。upstream 这个字段设置一群服务器，可以将这个字段放在 proxy_pass 和 fastcgi_pass 指令中作为一个单独的实体，它们可以是监听不同端口的服务器，并且也可以是同时监听 TCP 和 Unix socket 的服务器。服务器可以指定不同的权重，默认为 1。

```
    ... ..

}
```

修改 Nginx 主配置文件，在 server 中 location / 配置部分做如下修改：

```
server {

    ... ..

    location / {

        proxy_pass http://server_pool/;

        #确定需要代理的 URL，端口或 socket。

        proxy_redirect    off;

        #如果需要修改从后端服务器传来的应答头中的"Location"和"Refresh"字段，可以用这个指令设置。

        proxy_set_header    X-Real-IP    $remote_addr;

        #这个指令允许将发送到后端服务器的请求头重新定义或者增加一些字段。这个值可以是一个文本，变量或者它们的组合。
```

```
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;

        proxy_set_header    Host      $host;

        proxy_next_upstream error timeout invalid_header http_500 http_502 http_503
http_504 http_404;

        #确定在何种情况下请求将转发到下一个服务器：

        #error - 在连接到一个服务器，发送一个请求，或者读取应答时发生错误。

        #timeout - 在连接到服务器，转发请求或者读取应答时发生超时。

        #invalid_header - 服务器返回空的或者错误的应答。

        #http_500 - 服务器返回 500 代码。

        #http_502 - 服务器返回 502 代码。

        #http_503 - 服务器返回 503 代码。

        #http_504 - 服务器返回 504 代码。

        #http_404 - 服务器返回 404 代码。

        #off - 禁止转发请求到下一台服务器。

    }

    ... ..

}
```

Nginx 的 upstream 目前支持 5 种方式的分配

1 轮询（默认）

每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器 down 掉，能自动剔除。

2 weight

指定轮询几率，weight 和访问比率成正比，用于后端服务器性能不均的情况。

例如：

第 5 章 Nginx 反向代理

```
upstream bakend {  
    server 192.168.0.88 weight=10;  
    server 192.168.0.89 weight=10;  
}
```

3 ip_hash

每个请求按访问 ip 的 hash 结果分配，这样每个访客固定访问一个后端服务器，可以解决 session 的问题。

例如：

```
upstream bakend {  
    ip_hash;  
    server 192.168.0.88:80;  
    server 192.168.0.89:80;  
}
```

4 fair (第三方)

按后端服务器的响应时间来分配请求，响应时间短的优先分配。

例如：

```
upstream bakend {  
    server 192.168.0.88:80;  
    server 192.168.0.89:80;  
    fair;  
}
```

5 url_hash (第三方)

按访问 url 的 hash 结果来分配请求，使每个 url 定向到同一个后端服务器，后端服务器为缓存时比较有效。

例如：

```
upstream backend {  
    server 192.168.0.88:3128;  
    server 192.168.0.89:3128;  
    hash $request_uri;  
    hash_method crc32;  
}
```

每个设备的状态设置为:

1. down 表示单前的 server 暂时不参与负载
2. weight 默认为 1.weight 越大，负载的权重就越大。
3. max_fails : 允许请求失败的次数默认为 1.当超过最大次数时，返回 proxy_next_upstream 模块定义的错误
4. fail_timeout:max_fails 次失败后，暂停的时间。
5. backup : 其它所有的非 backup 机器 down 或者忙的时候，请求 backup 机器。所以这台机器压力会最轻。

Nginx 支持同时设置多组的负载均衡，用来给不同的 server 来使用。

第5.4节 课后练习

1：配置 Nginx 做为前端反向代理（IP 为 192.168.0.188），将 http 请求以轮询的方式转发至后端两台 apache 服务器（两台 apache 服务器 IP 分别为 192.168.0.220 192.168.0.221）。

配置 apache 服务器

第 5 章 Nginx 反向代理

配置 Nginx 代理服务器

设置 proxy 相关参数

配置 Nginx 接收到客户请求后使用 upstream 地址池

第6章 源代码搭建 LNMP

LNMP 是一个缩写，它指一组通常一起使用来运行动态网站或者服务器的自由软件：Linux，操作系统；Nginx，网页服务器；MySQL，数据库管理系统（或者数据库服务器）；PHP 和有时 Perl 或 Python 脚本语言。

在前期课程中大家已经接触过 LAMP 环境，它可以为我们提供一个功能强大的 WEB 平台，但是由于 Apache 的已不能上远远低于 Nginx，所以越来越多的企业从 LAMP 环境迁移到 LNMP 环境下，以提供一个高性能的，多功能的 WEB 平台。

接着我们开始进入 LNMP 搭建。

第6.1节 安装 rpm 包

首先安装如下 rpm 包，以满足 LNMP 环境需求。

```
libjpeg-devel
libpng-devel
libtiff-devel
fontconfig-devel
freetype-devel
libXpm-devel
gettext-devel
openssl-devel
libtool-ltdl-devel
```

libjpeg-devel，libpng-devel，libtiff-devel，fontconfig-devel，freetype-devel，libXpm-devel 这些都是图片与字体相关的开发包，为了使 php 可以对其做更好的支持。gettext 是语言相关的一个函数库。openssl-devel 是一套工具，用于生成 X.509 协议中所使用的密钥，公钥等文件。libtool 是一个通用库支持脚本，在 php 编译过程中会需

第 6 章 源代码搭建 LNMP

要使用到。

第6.2节 安装 Nginx

安装 pcre，系统自带的 pcre 版本过低，不能满足我们的需求。pcre 是一个正则表达式相关的包，要想 Nginx 使用 Rewrite，那么就需要正则的支持。

```
[root@server lnmp]# tar xf pcre-8.10.tar.bz2
[root@server lnmp]# cd pcre-8.10
[root@server pcre-8.10]# ./configure
[root@server pcre-8.10]# make && make install
```

编译安装 Nginx

```
[root@server lnmp]# useradd nginx
[root@server lnmp]# tar xf nginx-0.8.46.tar.gz
[root@server lnmp]# cd nginx-0.8.46
[root@server nginx-0.8.46]# ./configure \
    --user=nginx \
    --group=nginx \
    --prefix=/usr/local/nginx/ \
    --with-http_stub_status_module \
    --with-http_ssl_module \
    --with-sha1=/usr/lib
[root@server nginx-0.8.46]# make && make install
```

我们使用--user 与--group 的参数指定 Nginx 运行时的所有者，以提高安全性。--

prefix 指定了 Nginx 的安装位置。--with-http_stub_status_module 可以让 Nginx 获取到相关的工作状态，以便我们平时可以做更好的监控。--with-http_ssl_module 为启用 ssl 的支持。--with-sha1 指定希哈函数库位置。

第6.3节 编译安装 Mysql

```
[root@server lnmp]# tar xf mysql-5.1.49.tar.gz

[root@server lnmp]# cd mysql-5.1.49

[root@server mysql-5.1.49]# autoreconf --force --install

[root@server mysql-5.1.49]# ./configure \
    "--prefix=/usr/local/mysql" \
    "--localstatedir=/var/lib/mysql/" \
    "--enable-asm" \
    "--with-extra-charsets=all" \
    "--enable-thread-safe-client" \
    "--with-client-ldflags=-all-static" \
    "--with-mysqld-ldflags=-all-static" \
    "--with-pthread" \
    "--with-big-tables" \
    "--without-debug" \
    "--with-ssl"

[root@server mysql-5.1.49]# make && make install

[root@server mysql-5.1.49]# useradd mysql

[root@server mysql-5.1.49]# /usr/local/mysql/bin/mysql_install_db --user=mysql
```

第 6 章 源代码搭建 LNMP

```
[root@server ~]# cd /usr/local/mysql

[root@server mysql]# chown -R root:mysql .

[root@server mysql]# chown mysql. /var/lib/mysql -R

[root@server mysql]# cp share/mysql/my-huge.cnf /etc/my.cnf

[root@server mysql]# cp share/mysql/mysql.server /etc/init.d/mysql

[root@server mysql]# chmod 755 /etc/init.d/mysql

[root@server mysql]# chkconfig --add mysqld

[root@server mysql]# service mysqld start

[root@server mysql]# echo 'export PATH=$PATH:/usr/local/mysql/bin' >> ~/.bashrc

[root@server mysql]# . ~/.bashrc
```

Autoconf 可以用来分析用户的系统，判断是否符合 POSIX 标准，并提供相应解决方法。编译安装 mysql 时我们尽量以静态化方式编译，提高 mysql 性能，在安装之前你应该已经停止机器上原有的 mysql，甚至应该将原本的卸载。--enable-asm 参数将会使 mysql 使用一些字符函数的汇编版本，--with-extra-charsets 设置了支持的字符集，--enable-thread-safe-client 和--with-pthread 这两个参数要求 mysql 使用线程库并以线程方式编译了客户端。将 /usr/local/mysql 目录的所有者改为 root:mysql 是为了安全性。源代码编译 mysql 时，我们还需要手工创建配置文件和 system V 服务脚本。Mysql 的安装可能需要花费一些时间，所以编译安装时可以休息一会。

第6.4节 编译安装 php

在安装 php 之前要首先安装以下几个库

```
gd2
libiconv
libmcrypt
```

```
mhash  
spawn-fcgi  
libevent
```

gd2 是一个用以生成图形图片的库，RHEL 自带 gd2 的开发包，但是版本比较低，生成的图片是黑白的，非常不美观，因此这边单独编译一个高版本。

libiconv 用于实现一个字符编码到另一个字符编码的转换。

libmcrypt 可以使 php 支持更多加密算法。

mhash 是一个哈希演函数库，它可以支持多种哈希演算法，例如最出名的 MD5、SHA1 或 GOST，还有其它多种的哈希演算法。

spawn-fcgi spawn-fcgi 原本是 lighttpd 的一个用来控制 php-cgi 的工具，现在已从 lighttpd 中独立出来成为一个开源项目。我们可以用它来启动，控制，管理 php-cgi 的进程。使用它有诸多好处，比如我们可以将运行 php-cgi 的主机与 Nginx 分离开来，以减轻 Nginx 服务器的负担，或者用来扩展架构，做更多的负载分担。减少进程的重复创建所带来的性能消耗等等。

libevent 是一个事件触发的网络库，适用于 windows、linux、bsd 等多种平台，内部使用 select、epoll、kqueue 等系统调用管理事件机制。

编译安装 gd2：

```
[root@server lnmp]# tar xf gd-2.0.35.tar.bz2  
[root@server lnmp]# cd gd-2.0.35  
[root@server gd-2.0.35]# ./configure --prefix=/usr/local/gd2  
[root@server gd-2.0.35]# make && make install
```

编译安装 libiconv：

```
[root@server lnmp]# tar xf libiconv-1.13.1.tar.gz  
[root@server lnmp]# cd libiconv-1.13.1
```

第6章 源代码搭建 LNMP

```
[root@server libiconv-1.13.1]# ./configure
[root@server libiconv-1.13.1]# make && make install
```

编译安装 libmcrypt :

```
[root@server lnmp]# tar xf libmcrypt-2.5.7.tar.gz
[root@server lnmp]# cd libmcrypt-2.5.7
[root@server libmcrypt-2.5.7]# ./configure
[root@server libmcrypt-2.5.7]# make && make install
```

编译安装 mhash :

```
[root@server lnmp]# tar xf mhash-0.9.9.9.tar.bz2
[root@server lnmp]# cd mhash-0.9.9.9
[root@server mhash-0.9.9.9]# ./configure
[root@server mhash-0.9.9.9]# make && make install
```

编译安装 spawn-fcgi:

```
[root@server lnmp]# tar xf spawn-fcgi-1.6.3.tar.gz
[root@server lnmp]# cd spawn-fcgi-1.6.3
[root@server spawn-fcgi-1.6.3]# ./configure --prefix=/usr/local/spawn-fcgi
[root@server spawn-fcgi-1.6.3]# make && make install
```

编译安装 libevent:

```
[root@server lnmp]# tar xf libevent-1.4.14b-stable.tar.gz
[root@server lnmp]# cd libevent-1.4.14b-stable
[root@server libevent-1.4.14b-stable]# ./configure
```

```
[root@server libevent-1.4.14b-stable]# make && make install
```

编译安装 php :

```
[root@server lnmp]# tar xf php-5.2.14.tar.bz2
[root@server lnmp]# cd php-5.2.14
[root@server php-5.2.14]# ./configure \
    "--prefix=/usr/local/php" \
    "--enable-fastcgi" \
    "--enable-discard-path" \
    "--enable-force-cgi-redirect" \
    "--with-config-file-path=/usr/local/php/etc" \
    "--with-mysql=/usr/local/mysql" \
    "--with-mysqli=/usr/local/mysql/bin/mysqli_config" \
    "--with-iconv-dir" \
    "--with-freetype-dir" \
    "--with-jpeg-dir" \
    "--with-png-dir" \
    "--with-gd=/usr/local/gd2/" \
    "--with-zlib" \
    "--with-libxml-dir" \
    "--with-curl" \
    "--with-curlwrappers" \
    "--with-openssl" \
```

第 6 章 源代码搭建 LNMP

```
--with-mhash \  
"--with-xmlrpc" \  
"--with-mcrypt" \  
"--enable-xml" \  
"--enable-bcmath" \  
"--enable-shmop" \  
"--enable-sysvsem" \  
"--enable-inline-optimization" \  
"--enable-mbregex" \  
"--enable-mbstring" \  
"--enable-gd-native-ttf" \  
"--enable-ftp" \  
"--enable-pcntl" \  
"--enable-sockets" \  
"--enable-zip" \  
"--disable-debug" \  
"--disable-ipv6"  
  
[root@server php-5.2.14]# make ZEND_EXTRA_LIBS='-liconv'  
  
[root@server php-5.2.14]# make install  
  
[root@server php-5.2.14]# cp php.ini-dist /usr/local/php/etc/php.ini
```

--enable-fastcgi 启动 fast-cgi

其中 "--enable-discard-path" 打开这个选项，用户就不能透过浏览器读取类似 .htaccess 的系统安全相关的文件。

"--enable-force-cgi-redirect" 若使用 CGI VERSION 模式来执行 PHP 的设，打开本选项会增加安全性。例如用户读 `http://my.host/cgi-bin/php/secret/doc.html` 遇到比较了解 PHP 系统的黑客级用户可能会自己输入以下网址 `http://my.host/secret/doc.html` 来读取相关信息。若 PHP 和 Apache 编译在一起，让 PHP 变成 Apache 的一部份，则不需要加入本选项。

--with-config-file-path 在指定 php 主配置文件的路径

--with-mysql 和 --with-mysqli 在指定你的 mysql 的位置和它的相关工具

--with-iconv-dir, --with-freetype-dir, --with-jpeg-dir, --with-png-dir, --with-gd, --with-zlib, --with-libxml-dir 这些都是在启用对某种文件的支持

--with-curl 和 --with-curlwrappers 是用于支持 curl 函数，此函数允许你用不同的协议连接和沟通不同的服务器

--with-openssl, --with-mhash, --with-mcrypt 这都是和加密有关的参数，启用它们是为了让 php 可以更好的支持各种加密。

"--enable-bcmath" 高精度数学运算组件。

"--enable-shmop" 和 "--enable-sysvsem" 使得你的 PHP 系统可以处理相关的 IPC 函数。IPC 是一个 Unix 标准通讯机制，它提供了使得在同一台主机不同进程之间可以互相通讯的方法。

"--enable-inline-optimization" 栈堆指针和优化线程。

"--enable-pcntl" 多线程优化。

make ZEND_EXTRA_LIBS='-liconv' 手工指定将 iconv 加到 php 额外库中，一般来说这些库的增加 php 可以自动完成，只是 iconv 貌似不太合群，需要手工操作。

make 完了之后，如果没有错误，你也可以执行一下 make test 看看是否有错误，不过时间会比较长。

第6.5节 安装 php 扩展模块工具

安装扩展模块是为了进一步完善我们的 php，或提高性能，或提高安全性，或扩展功能，

第 6 章 源代码搭建 LNMP

或增加稳定性等。我们接下来会安装如下扩展模块：

```
memcache  
eaccelerator  
PDO_MYSQL  
ImageMagick
```

memcache 是一个高性能的分布式的内存对象缓存系统，通过在内存里维护一个统一的巨大的 hash 表，它能够用来存储各种格式的数据，包括图像、视频、文件以及数据库检索的结果等。

eAccelerator 加速引擎是基于 memcache 开发的 PHP 加速优化软件。通过编译和缓存来增加 PHP 脚本的性能，使得 PHP 脚本在编译的状态下降低服务器负载，对服务器的开销几乎完全消除。它还对脚本起优化作用，能加快其执行效率，提高 PHP 应用执行速度最高达 10 倍。

PDO_MYSQL 是一个 php 的扩展模块，可以让 php 更好调用 mysql。

ImageMagick 是一个用于查看、编辑位图文件以及进行图像格式转换的开放源代码软件套装。

编译安装 memcache:

```
[root@server tmp]# tar -zxf memcache-3.0.4.tgz  
[root@server tmp]# cd memcache-3.0.4  
[root@server memcache-3.0.4]# /usr/local/php/bin/phpize  
[root@server memcache-3.0.4]# ./configure --with-php-config=/usr/local/php/bin/php-config  
[root@server memcache-3.0.4]# make && make install
```

编译安装 eaccelerator:

```
[root@server lnmp]# tar xf eaccelerator-0.9.6.1.tar.bz2
```

```
[root@server lnmp]# cd eaccelerator-0.9.6.1

[root@server eaccelerator-0.9.6.1]# /usr/local/php/bin/phpize

[root@server eaccelerator-0.9.6.1]# ./configure --enable-eaccelerator=shared --with-php-config=/usr/local/php/bin/php-config

[root@server eaccelerator-0.9.6.1]# make && make install
```

编译安装 PDO_MYSQL:

```
[root@server lnmp]# tar xf PDO_MYSQL-1.0.2.tgz

[root@server lnmp]# cd PDO_MYSQL-1.0.2

[root@server PDO_MYSQL-1.0.2]# /usr/local/php/bin/phpize

[root@server PDO_MYSQL-1.0.2]# ./configure --prefix=/usr/local/pdo-mysql --with-php-config=/usr/local/php/bin/php-config --with-pdo-mysql=/usr/local/mysql

[root@server PDO_MYSQL-1.0.2]# make && make install
```

编译安装 ImageMagick:

```
[root@server lnmp]# tar xf ImageMagick-6.5.9-10.tar.bz2

[root@server lnmp]# cd ImageMagick-6.5.9-10

[root@server ImageMagick-6.5.9-10]# ./configure \

    --enable-shared \

    --with-modules \

    --without-x \

    --with-gs-font-dir=default \

    --with-perl=yes \

    --with-zlib=yes \

    --with-jpeg=yes
```

第6章 源代码搭建 LNMP

```
[root@server ImageMagick-6.5.9-10]# make && make install
```

编译安装 imagick:

```
[root@server lnmp]# tar xf imagick-3.0.0.tgz
[root@server lnmp]# cd imagick-3.0.0
[root@server imagick-3.0.0]# /usr/local/php/bin/phpize
[root@server imagick-3.0.0]# ./configure --with-php-config=/usr/local/php/bin/php-config
[root@server imagick-3.0.0]# make && make install
```

--with-php-config 在指定 php 的配置工具，/usr/local/php/bin/phpize 是用来扩展 php 的扩展模块的，通过 phpize 可以建立 php 的外挂模块。

第6.6节 使 PHP 扩展生效

修改 php 主配置文件，以使 php 支持扩展模块。

```
[root@server ~]# cat >> /usr/local/php/etc/php.ini << ENDF
[eAccelerator]
zend_extension="/usr/local/php/lib/php/extensions/no-debug-non-zts-20060613/eaccelerator.so"
eaccelerator.shm_size="32"
eaccelerator.cache_dir="/usr/local/cache/ea"
eaccelerator.enable="1"
eaccelerator.optimizer="1"
eaccelerator.check_mtime="1"
eaccelerator.debug="0"
eaccelerator.filter=""
```

```
eaccelerator.shm_max="0"

eaccelerator.shm_ttl="0"

eaccelerator.shm_prune_period="0"

eaccelerator.shm_only="0"

eaccelerator.compress="1"

eaccelerator.compress_level="9"

ENDF

[root@server ~]# mkdir -p /usr/local/cache/ea

[root@server ~]# vi php.ini

将

expose_php = On

改为

expose_php = Off

找到

extension_dir

改为

extension_dir = "/usr/local/php/lib/php/extensions/no-debug-non-zts-20060613/"

并在其下面追加三行内容

extension = "memcache.so"

extension = "pdo_mysql.so"

extension = "imagick.so"

将

output_buffering = Off
```

第 6 章 源代码搭建 LNMP

改为

```
output_buffering = On
```

`expose` 是 `php` 的一个参数，关闭它则会显示更少的 `php` 消息，以提高安全性。
`output_buffering` 是一个缓存有关的参数选项。

更多 `php` 核心参数的配置请参考 <http://manual.phpv.net/zh/ini.core.php>

第6.7节 整合 Nginx 与 php

在 `http {}` 全局配置部分加入如下内容:

```
http
{
    ...

    fastcgi_connect_timeout 300;
    fastcgi_send_timeout 300;
    fastcgi_read_timeout 300;

    fastcgi_buffer_size 128k;
    fastcgi_buffers 4 128k;
    fastcgi_busy_buffers_size 128k;
    fastcgi_temp_file_write_size 128k;
    fastcgi_intercept_errors on;
```

```
proxy_temp_path    /dev/shm/proxy_temp;

fastcgi_temp_path  /dev/shm/fastcgi_temp;

client_body_temp_path /dev/shm/client_body_temp;

#spawn-fcgi 开启 9000 跟 9001 两个端口，利用 nginx 的 upstream 负载均衡 php 程序到不同的
#fcgi 端口上面

    upstream spawn {

        server 127.0.0.1:9000 max_fails=0 fail_timeout=30s;

        server 127.0.0.1:9001 max_fails=0 fail_timeout=30s;

    }

    ...

}
```

修改监听 80 端口的虚拟主机 "location ~ \.php { ... }" 的配置

```
server

{

    ....

    location ~ \.php$ {

        root    html;

#        fastcgi_pass 127.0.0.1:9000;

        fastcgi_pass spawn;

    }

}
```

第6章 源代码搭建 LNMP

```
fastcgi_index index.php;

fastcgi_param SCRIPT_FILENAME /scripts$fastcgi_script_name;

#   include    fastcgi_params;

   include spawn_php5.conf;

}

...

}
```

创建 spawn_php5.conf

```
[root@server ~]# vi /usr/local/nginx/conf/spawn_php5.conf

fastcgi_param GATEWAY_INTERFACE CGI/1.1;

fastcgi_param SERVER_SOFTWARE    nginx;

fastcgi_param QUERY_STRING       $query_string;

fastcgi_param REQUEST_METHOD     $request_method;

fastcgi_param CONTENT_TYPE       $content_type;

fastcgi_param CONTENT_LENGTH     $content_length;

fastcgi_param SCRIPT_FILENAME    $document_root$fastcgi_script_name;

fastcgi_param SCRIPT_NAME        $fastcgi_script_name;

fastcgi_param REQUEST_URI        $request_uri;

fastcgi_param DOCUMENT_URI       $document_uri;

fastcgi_param DOCUMENT_ROOT      $document_root;
```

```
fastcgi_param SERVER_PROTOCOL $server_protocol;

fastcgi_param REMOTE_ADDR $remote_addr;

fastcgi_param REMOTE_PORT $remote_port;

fastcgi_param SERVER_ADDR $server_addr;

fastcgi_param SERVER_PORT $server_port;

fastcgi_param SERVER_NAME $server_name;
```

配置 spawn-fcgi 启动脚本

```
[root@server ~]# vi /usr/local/php/bin/spawn-fcgi

#!/bin/sh

set -e

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin :/usr/local/spawn-fcgi/bin

DESC="spawn-fcgi daemon"

NAME=spawn-fcgi

DAEMON=/usr/local/spawn-fcgi/bin/$NAME

# Gracefully exit if the package has been removed.

test -x $DAEMON || exit 0

d_start() {

    $DAEMON -a 127.0.0.1 -p 9000 -C 128 -u nginx -g nginx -f /usr/local/php/bin/php-cgi
> /dev/null 2>&1

    $DAEMON -a 127.0.0.1 -p 9001 -C 128 -u nginx -g nginx -f /usr/local/php/bin/php-cgi
> /dev/null 2>&1 || echo -n " adress already inuse"
```

第6章 源代码搭建 LNMP

```
}

d_stop() {
    /usr/bin/killall -9 php-cgi > /dev/null 2>&1 || echo -n " not running"
}

case "$1" in
    start)
        echo -n "Starting $DESC: $NAME"

        d_start

        echo "ok"

        ;;
    stop)
        echo -n "Stopping $DESC: $NAME"

        d_stop

        echo "ok"

        ;;
    restart)
        echo -n "Restarting $DESC: $NAME"

        d_stop

        sleep 1

        d_start

        ;;
```

```
*)  
  
    echo "Usage: $SCRIPTNAME {start|stop|restart}" >&2  
  
    exit 3  
  
    ;;  
  
esac  
  
exit 0  
  
[root@server conf]# chmod +x /usr/local/php/bin/spawn-fcgi
```

第6.8节 启动 Nginx 服务

先启动 spawn-fcgi CGI 服务：

```
[root@server ~]# /usr/local/php/bin/spawn-fcgi start
```

然后测试 Nginx 服务配置是否正确：

```
[root@server ~]# /usr/local/nginx/sbin/nginx -t
```

如果返回如下信息，说明配置正确

```
the configuration file /usr/local/nginx//conf/nginx.conf syntax is ok  
configuration file /usr/local/nginx//conf/nginx.conf test is successful
```

正式启动 Nginx 服务：

```
[root@server ~]# /usr/local/nginx/sbin/nginx
```

建立 phpinfo 测试页面

第 6 章 源代码搭建 LNMP

```
[root@server ~]# vi /usr/local/nginx/html/test.php

<?php

phpinfo

?>
```

搭建 discuz 环境测试。

第6.9节 课后练习

1：源代码搭建 lnmp 环境。

第7章 课后练习参考答案

第一章 略

第二章

1.源代码编译 Nginx。

挂载服务器共享目录，并从服务器上将 nginx 源代码拷备到本地

```
[root@example ~]# mount 192.168.0.254:/var/ftp/pub /mnt  
[root@example ~]# cp /mnt/tools/iterm/nginx-0.8.46.tar.gz /tmp
```

解压并进行编译安装

```
[root@example ~]# cd /tmp  
[root@example tmp]# tar xf nginx-0.8.46.tar.gz  
[root@example tmp]# cd nginx-0.8.46  
[root@example nginx-0.8.46]# ./configure --user=nginx --group=nginx  
--prefix=/usr/local/nginx/ --with-http_stub_status_module --with-http_ssl_module  
[root@example nginx-0.8.46]# make  
[root@example nginx-0.8.46]# make install
```

启动 Nginx 服务并测试

```
[root@example nginx-0.8.46]# /usr/local/nginx/sbin/nginx  
[root@example nginx-0.8.46]# links localhost
```

2.创建 Nginx 服务管理脚本。

在/etc/init.d 目录下创建 Nginx 服务脚本

第7章 课后练习参考答案

```
[root@example ~]# vi /etc/init.d/nginx

#!/bin/sh

#

# nginx - this script starts and stops the nginx daemon

#

# chkconfig: - 85 15

# description: Nginx is an HTTP(S) server, HTTP(S) reverse \
#             proxy and IMAP/POP3 proxy server

# processname: nginx

# config:      /etc/nginx/nginx.conf

# config:      /etc/sysconfig/nginx

# pidfile:     /var/run/nginx.pid

# Source function library.

./etc/rc.d/init.d/functions

# Source networking configuration.

./etc/sysconfig/network

# Check that networking is up.

[ "$NETWORKING" = "no" ] && exit 0
```

```
nginx="/usr/local/nginx/sbin/nginx"

prog=$(basename $nginx)

NGINX_CONF_FILE="/usr/local/nginx/conf/nginx.conf"

[ -f /etc/sysconfig/nginx ] && . /etc/sysconfig/nginx

lockfile=/var/lock/subsys/nginx

start() {
    [ -x $nginx ] || exit 5
    [ -f $NGINX_CONF_FILE ] || exit 6
    echo -n "Starting $prog: "
    daemon $nginx -c $NGINX_CONF_FILE
    retval=$?
    echo
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}

stop() {
    echo -n "Stopping $prog: "
```

第7章 课后练习参考答案

```
killproc $prog -QUIT

retval=$?

echo

[ $retval -eq 0 ] && rm -f $lockfile

return $retval

    killall -9 nginx
}

restart() {

    configtest || return $?

    stop

    sleep 1

    start

}

reload() {

    configtest || return $?

    echo -n $"Reloading $prog: "

    killproc $nginx -HUP

    RETVAL=$?

    echo

}
```

```
force_reload() {  
    restart  
}  
  
configtest() {  
    $nginx -t -c $NGINX_CONF_FILE  
}  
  
rh_status() {  
    status $prog  
}  
  
rh_status_q() {  
    rh_status >/dev/null 2>&1  
}  
  
case "$1" in  
    start)  
        rh_status_q && exit 0  
        $1  
        ;;  
    stop)  
        rh_status_q || exit 0
```

第7章 课后练习参考答案

```
$1
;;
restart|configtest)

    $1
    ;;
    reload)

        rh_status_q || exit 7

        $1
        ;;
        force-reload)

            force_reload

            ;;
        status)

            rh_status

            ;;
        condrestart|try-restart)

            rh_status_q || exit 0

            ;;
    *)

        echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload|
configtest}"

        exit 2
esac
```

修改权限

```
[root@example ~]# chmod 755 /etc/init.d/nginx
```

加入 chkconfig 管理列表

```
[root@example ~]# chkconfig --add nginx
```

使用 service 命令进行重启测试

```
[root@example ~]# service nginx restart
```

第三章

1. 配置两个基于名称的虚拟主机 `www.test.com` 和 `www.example.com`（注意，做此实验之前你应该已经有正确的 DNS 配置）。

修改 Nginx 主配置文件，增加两个 `server` 字段。

```
[root@example ~]# vi /usr/local/nginx/conf/nginx.conf
```

```
#在 http 字断中加入以下两断内容
```

```
server {  
  
    listen 80;  
  
    server_name www.example.com;  
  
    root exmaple.com  
  
    index index.html  
  
}  
  
server {  
  
    listen 80;  
  
    server_name www.test.com;  
  
    test.com
```

第7章 课后练习参考答案

```
    index index.html  
  
}
```

创建相关网站主目录并重启 Nginx 服务。

```
[root@example ~]# mkdir /usr/local/nginx/html/example.com  
  
[root@example ~]# echo "www.example.com" >  
/usr/local/nginx/html/example.com/index.html  
  
[root@example ~]# mkdir /usr/local/nginx/html/test.com  
  
[root@example ~]# echo www.test.com > /usr/local/nginx/html/test.com/index.html  
  
[root@example ~]# service nginx restart
```

2. 配置一个 https。以使你的服务器可以提供安全的连接。

生成安全密钥。

```
[root@example ~]# cd /etc/pki/tls/certs  
  
[root@example certs]# cd /etc/pki/tls/certs  
  
[root@example certs]# make server.key  
  
[root@example certs]# openssl rsa -in server.key -out server.key  
  
[root@example certs]# make server.csr  
  
[root@example certs]# openssl x509 -in server.csr -req -signkey server.key -days 365 -out  
server.crt
```

修改 Nginx 主配置文件，加入 ssl 相关配置。

```
[root@example certs]# vi /usr/local/nginx/conf/nginx.conf  
  
#在 http 段中加入以下配置  
  
server {
```

```
listen    443;

server_name www.example.com;

ssl       on;

ssl_certificate    /etc/pki/tls/certs/server.crt;

ssl_certificate_key /etc/pki/tls/certs/server.key;

ssl_session_timeout 5m;

ssl_protocols SSLv2 SSLv3 TLSv1;

ssl_ciphers ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;

ssl_prefer_server_ciphers on;

    location / {

        root html;

        index index.html;

    }

}
```

重启 Nginx 服务。

```
[root@example ~]# service nginx restart
```

3. 配置你的虚拟主机 (www.test.com) , 当别人试图访问时, 要求进行身份认证, 允许 mark 用户可以通过密码 uplooking 进行访问。

修改 Nginx 主配置文件 增加 auth 相关配置

```
[root@example ~]# vi /usr/local/nginx/conf/nginx.conf

#修改 location / 字段

location / {
```

第7章 课后练习参考答案

```
root html;

index index.html index.htm;

auth_basic "xxxx";

auth_basic_user_file /usr/local/nginx/passwd.db;

}
```

创建认证使用密码文件

```
[root@example nginx]# htpasswd -c /usr/local/nginx/passwd.db mark

New password: ( 输入密码 uplooking )

Re-type new password: ( again )

Adding password for user mark
```

重启 Nginx 服务。并访问测试

```
[root@example ~]# service nginx restart
```

第四章

1. 配置虚拟主机 *.test.com。当客户端访问任意 XXX.test.com 时都能匹配此主机并将地址重写成 www.test.com/XXX/（注意，实验之前你应该保证 XXX.test.com 都能正常解析）

修改主配置文件，配置虚拟主机 *.test.com

书写 if 语句，判断如果以 test.com 结尾的由进行 url 重写

将 test.com 之前的内容保存于一个变量 domain 中

将原地址重写成 [www.test.com/\\$domain/](http://www.test.com/$domain/)

```
server {

    listen 80;
```

```
server_name *.test.com;

if ( $http_host ~* "^(.*)\.test\.com$" ) {

    set $domain $1;

    rewrite ^(*) http://www.test.com/$domain/ break;

}

}
```

2. 当用户访问服务器 `http://www.test.com/page-uid-123.html` 时服务器自动将地址重写成 `http://www.test.com/page.php?uid=123` , 当访问 `http://www.test.com/page-username-123.html` 重写成 `http://www.test.com/page.php?username=123` 。其中 `123.html` 可能是任意数字文件名 `XXX.html`

配置主配置文件并书写 rewrite 规则

使用 location 语句。

在 Rewrite 中将 username 和 uid 字段保存下来，并以变量\$1 方式引用。

将 XXX.html 字段保存下来，并以变量\$2 方式引用。

```
location ~* \.html$ {

    rewrite /page-(uid|username)-([0-9]+)\.html      /page.php?$1=$2 last ;

}
```

3. 当用户访问服务器 `http://www.test.com/today-pic-112.html` 时服务器自动将地址重写成 `http://www.test.com/today/pic/112.html`。（注意，此处 today pic 可以更换成其他任意字母，112 可以更换成其他任意数字）

配置主配置文件并书写 rewrite 规则

配置以 html 结尾的文件做 rewrite

将 today pic 112 字段分别保存下来并以\$1 \$2 \$3 方式引用

第 7 章 课后练习参考答案

```
location ~* \.html {  
    rewrite /([a-z]+)-([a-z]+)-([0-9]+)\.html /$1/$2/$3.html last;  
}
```

第五章

1. 配置 Nginx 做为前端反向代理 (IP 为 192.168.0.188) , 将 http 请求以轮询的方式转发至后端两台 apache 服务器 (两台 apache 服务器 IP 分别为 192.168.0.220 192.168.0.221) 。

配置 apache 服务器

登录到 192.168.0.220 配置 httpd 服务

```
[root@example nginx]# rpm -ivh httpd*  
  
[root@example nginx]# echo 'this is 192.168.0.220' > /var/www/html/index.html  
  
[root@example nginx]# service httpd start
```

登录到 192.168.0.221 配置 httpd 服务

```
[root@example nginx]# rpm -ivh httpd*  
  
[root@example nginx]# echo 'this is 192.168.0.221' > /var/www/html/index.html  
  
[root@example nginx]# service httpd start
```

配置 Nginx 代理服务器

设置 proxy 相关参数

```
[root@example nginx]# vi /usr/local/nginx.conf  
  
#在 httpd 字段中加入如下内容  
  
http {  
    .....
```

```
client_max_body_size 300m;

client_body_buffer_size      128k;

client_body_temp_path       /dev/shm/client_body_temp;

proxy_connect_timeout       600;

proxy_read_timeout          600;

proxy_send_timeout          600;

proxy_buffer_size           16k;

proxy_buffers                4 32k;

proxy_busy_buffers_size     64k;

proxy_temp_file_write_size  64k;

proxy_temp_path             /dev/shm/proxy_temp;
```

#配置 Nginx 接收到客户请求后使用 upstream 地址池

```
upstream    server_pool  {

    server 192.168.0.220:80 weight=8 max_fails=2 fail_timeout=30s;

    server 192.168.0.221:80 weight=8 max_fails=2 fail_timeout=30s;

}

}
```

修改 Nginx 主配置文件，在 server 中 location / 配置部分做如下修改：

```
[root@example nginx]# vi /usr/local/nginx.conf
```

```
location / {

    proxy_pass http://server_pool;

    proxy_redirect    off;
```

第7章 课后练习参考答案

```
proxy_set_header    X-Real-IP    $remote_addr;

proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;

proxy_set_header    Host      $host;

proxy_next_upstream error timeout invalid_header http_500 http_502 http_503
http_504 http_404;

}

[root@example nginx]# service nginx restart
```

测试

```
[root@example nginx]# links 192.168.0.188
```