```
#echo "export LD_PRELOAD=/usr/local/lib/libtcmalloc.so" >> /etc/init.d/mysqld
#service mysql restart
```

18.2.3　安装和配置 Apache

下载 httpd-2.2.17.tar.gz 源码包（http://apache.freelamp.com//httpd/httpd-2.2.17.tar.gz）并解压。

1. 安装 apr 和 apr-util

apr 和 apr-util 都包含在 httpd-2.2.17 发行源码中的 srclib 目录下，进入该目录，使用以下命令安装：

```
#./configure --prefix=/usr/local/apr
#make && make install
#cd ../apr-util
#./configure --prefix=/usr/local/apr-util --with-apr=/usr/local/apr/
--with-mysql=/usr/local/mysql
#make && make install
```

2. 安装 Httpd 2.2

进入 httpd-2.2.17 源码目录，编译和安装 httpd-2.2.17。在 LNAMP 中，Apache 作为后端服务器用来处理动态内容，无需向其中添加过多的模块。因此我们在编译配置时，只添加必要的模块保证 Apache 服务器的正常运行。笔者使用的配置命令为：

```
#./configure --prefix=/usr/local/apache2 --with-mysql=/usr/local/mysql --with-apr=
/usr/local/apr/ --with-apr-util=/usr/local/apr-util/   --enable-so
--enable-rewrite --with-mpm=prefork --disable-cgid --disable-cgi
```

18.2.4　安装 PHP

1. 环境配置

首先需要配置两个环境变量 PHP_AUTOCONF 和 PHP_AUTOHEADER：

```
#export  PHP_AUTOCONF=/usr/bin/autoconf-2.13
#export  PHP_AUTOHEADER=/usr/bin/autoheader-2.13
```

大家根据自己平台的实际情况调整上面的路径。

2. 安装 PHP

PHP 5.2.14 源码安装包下载的地址为 http://us.php.net/distributions/php-5.2.14.tar.gz，下载该源码包并解压。这里还需要给 PHP 安装一个补丁，补丁的下载地址为 http://download.suhosin.org/suhosin-patch-5.2.14-0.9.7.patch.gz，下载后进入 php-5.2.14 目录，使用以下命令安装补丁：

```
gunzip suhosin-patch-5.2.14-0.9.7.patch.gz
cd php-5.2.14
patch -p 1 -i ../suhosin-patch-5.2.14-0.9.7.patch
```

接下来就可以编译和安装 PHP 了。对 PHP 编译配置时需要添加的参数比较多，笔者的配置命令和编译过程为：

```
#./buildconf --force
#./configure --with-mysqli=/usr/local/mysql/bin/mysql_config
--with-apxs2=/usr/local/apache2/bin/apxs --with-mysql=/usr/local/mysql
--with-config-file-path=/etc --with-zlib --with-libxml-dir --with-gd=/usr/local/gd
```

```
--with-freetype-dir --with-jpeg-dir --with-png-dir --with-ttf --with-iconv
--with-openssl  --with-mcrypt --enable-sockets --enable-bcmath --enable-calendar
--enable-exif --enable-libxml  --enable-magic-quotes  --enable-mbstring --with-bz2
--with-curl --with-xmlrpc --with-gettext  --enable-suhosin  --disable-cli
--disable-cgi   --disable-debug   --with-pdo   --with-pdo_mysql
--prefix=/usr/local/php
#make;make install
#cp php.ini-dist  /etc/php.ini
```

3. 安装加速器

安装 PHP 的工作完成后，可以安装 eaccelerator 用来提高 PHP 的执行效率。eaccelerator 的源码下载路径是 http://bart.eaccelerator.net/source/0.9.5.3/eaccelerator-0.9.5.3.tar.bz2。在编译配置时使用如下命令：

```
#./configure --enable-eaccelerator=shared --with-php-config=/usr/local/php/bin/
php-config
```

注意

在编译安装 eaccelerator 后，Linux 平台的终端上会显示 eaccelerator.so 的安装目录，该目录在后面要用到。

安装 eaccelerator 完成后，我们需要在 PHP 配置文件中添加配置，使 eaccelerator 生效。打开 php.ini 文件，添加以下内容：

```
[eaccelerator]
zend_extension="/usr/local/php/lib/php/extensions/no-debug-non-zts-20060613/eacce
lerator.so"
eaccelerator.shm_size="16"
eaccelerator.cache_dir="/tmp/eaccelerator"
eaccelerator.enable="1"
eaccelerator.optimizer="1"
eaccelerator.check_mtime="1"
eaccelerator.debug="0"
eaccelerator.filter=""
eaccelerator.shm_max="0"
eaccelerator.shm_ttl="3600"
eaccelerator.shm_prune_period="3600"
eaccelerator.shm_only="0"
eaccelerator.compress="1"
eaccelerator.compress_level="9"
```

在该配置中，zend_extension 参数的值是上面 eaccelerator.so 的绝对路径，eaccelerator.cache_dir 设置缓存的存放目录，需要手动创建此目录并更改其读写权限：

```
mkdir  /tmp/eaccelerator
chmod  0777  /tmp/eaccelerator
```

4. Apache 与 PHP 整合

Apache 与 PHP 整合不是我们重点要关注的，这里我们简单说明一下。整合的过程主要是对 Apache 的配置文件 httpd.conf、httpd-default.conf、httpd-mpm.conf、httpd-vhosts.conf 进行修改，这几个文件都存放在 Apache 安装目录下的 conf 目录中。

在 httpd.conf 文件中，主要设置 DocumentRoot 的值和<Directory “dir” >节点的值为我们网站的根目录绝对路径，在<Directory></Directory>节中将 “Deny from all” 修改为 “Allow from all”，根据自己的需要修改监听端口、用户组等参数。另外，还要将以下四条语句的注释去掉。

```
#Include conf/extra/httpd-mpm.conf
#Include conf/extra/httpd-info.conf
#Include conf/extra/httpd-vhosts.conf
#Include conf/extra/httpd-default.conf
```

httpd-mpm.conf 文件在 conf 目录下的 extra 目录中，我们主要根据实际的运行环境配置<IfModule></IfModule>节下的各参数。

httpd-default.conf 文件也在 conf 目录下的 extra 目录中，在其中添加以下配置：

```
Timeout 60
KeepAlive On
MaxKeepAliveRequests 1000
KeepAliveTimeout 5
```

httpd-vhosts.conf 文件也在 conf 目录下的 extra 目录中，在<VirtualHos></VirtualHos>中可以设置虚拟主机。

Apache 与 PHP 整合的关键主要就是这些。如果大家有特殊的需求，可以根据实际情况继续设置，我们在这里就不详细阐述了。最后启动 Apache 的 Web 服务：

```
service httpd start
```

5. 安装和配置 Nginx

为了使得 Nginx 服务器的运行效率更高，我们启用 google_perftools 模块，在编译配置 Nginx 源码时添加参数--with-google_perftools_module。同时，需要为该模块创建临时目录并设置权限：

```
#mkdir /tmp/tcmalloc
#chmod 0777 /tmp/tcmalloc
```

Nginx 服务器的编译和安装我们在前面的相关章节已经详细讲述过了，这里重点看一下 Nginx 服务器的配置：

```
user nobody nobody;
worker_processes 4;
pid logs/nginx.pid;
#配置 Google 的性能提升模块
google_perftools_profiles /tmp/tcmalloc;
worker_rlimit_nofile 51200;
events
{
    use epoll;
    worker_connections 51200;
}
http{
    include       mime.types;
    default_type application/octet-stream;
    server_names_hash_bucket_size 128;
    client_header_buffer_size 32k;
```

```
large_client_header_buffers  4  32k;
client_max_body_size  8m;
sendfile  on;
tcp_nopush    on;
tcp_nodelay on;
keepalive_timeout  60;
#配置 gzip 功能
gzip on;
gzip_min_length  1k;
gzip_buffers    4 16k;
gzip_http_version 1.0;
gzip_comp_level 2;
gzip_types      text/plain application/x-javascript text/css application/xml;
gzip_vary on;
server
{
   listen  80;
   server_name   myweb;
   index  index.html index.htm index.php;
   root    /home/www/;
   location ~ .*\.(php|php5)?$ {
         index index.php;
         root /home/www/;
         proxy_pass
   }
   #配置反向代理的相关指令
   proxy_redirect          off;
   proxy_set_header        Host $host;
   proxy_set_header        X-Real-IP $remote_addr;
   proxy_set_header        X-Forwarded-For   $proxy_add_x_forwarded_for;
   proxy_connect_timeout   30;
   proxy_send_timeout      30;
   proxy_read_timeout      60;
   proxy_buffer_size       4k;
   proxy_buffers           4  32k;
   proxy_busy_buffers_size  64k;
   proxy_temp_file_write_size  64k;
   proxy_next_upstream  error timeout invalid_header http_500 http_503 http_404;
   proxy_max_temp_file_size  128m;
   client_body_temp_path client_body 1 2;
   proxy_temp_path proxy_temp 1 2;
   #设置对静态的媒体文件进行缓存，过期时间 30 天
   location ~* \.(jpg|jpeg|gif|png|swf)$ {
         if (-f $request_filename) {
            root /home/www/;
            expires    30d;
            break;
```

```
            }
        }
        #设置对静态的 js 文件和 cs 文件进行缓存，过期时间 1 天
        location ~* \.(js|css)$ {
            if (-f $request_filename) {
                root /home/www/;
                expires    1d;
                break;
            }
    }
}
```

完成相关配置后，启动 Nginx 服务器即可。

通过以上的步骤，我们手动部署和配置了一套 LNAMP 架构，这样部署的一套环境没有进行过任何优化措施，只包含了基本的 Web 服务功能。使用 PHP 建立的网站放在/home/www/目录下。

18.3　自动安装

从上节的内容我们看到，手动部署 LNAMP 的过程是相当烦琐的。由于 LNAMP 的使用非常广泛，网络上出现了“LNAMP 一键安装”的整合包，该包将所有操作集成在一个 Linux Shell 脚本文件中，使用者可以方便地进行安装、卸载、增加和删除用户虚拟主机等常规操作，这极大地方便了用户的使用。

该“LNAMP 一键安装”整合包实现了以下功能：

- 自动源码安装、卸载 Apache、Mysql、PHP、Nginx 以及 Pureftpd、Jailkit、PhpMyAdmin。
- 自动更新本机 IP，协助添加域名及更新默认虚拟主机。
- 实现增删用户及增删虚拟主机、操作数据库等管理操作。
- 默认开通用户 SSH。
- 默认开通 FTP 服务。

“LNAMP 一键安装”整合包减轻了用户的工作量，降低了 LNAMP 部署过程中的出错概率，笔者推荐大家在一般情况下使用该方法部署 LNAMP。该整合包目前已经更新到第二版，请大家到网站 http://www.wdlinux.cn/linux_lanmp 参阅相关的说明文档，笔者在这里就不详细讲解整合包的使用了。

18.4　本章小结

本章通过 Nginx 服务器的一个经典应用案例——LNAMP 来说明如何通过服务器整合方式集合各种服务器的优势之处，扬长避短，达到整体 Web 架构在功能上的全面和稳定。LNAMP 在大型 Web 应用和 Web 网站发布中应用广泛，其通过对不同服务器的有效配置，可以在功能上极大地丰富单一服务器程序提供的功能，在性能上为高并发访问提供良好的支持，而且能够保证稳定的运行和较低的系统资源消耗。希望大家能够在本章提供的基本整合配置基础上不断深入学习各个服务器的功能配置和性能优化，整合出功能丰富、性能优良的 Web 服务器架构。

附录 A

Nginx 内置变量

Nginx 服务器的配置文件中可以包含一些预设的变量，以帮助大家获取 Nginx 自身和网络等方面的信息供配置使用。本附录为大家整理和收录了 Nginx 服务器中常用的内置变量，并对变量的含义做了相应的解释，供大家在使用中查询。如果在对 Nginx 服务器配置的过程中出现此附录未收录的内置变量，请大家到 Nginx 官方网站查询。

以下是常用的 Nginx 服务器内置变量：

$arg_PARAMETER	客户端 GET 请求中 PARAMETER 字段的值
$args	客户端请求中的参数
$binary_remote_addr	远程地址的二进制表示
$body_bytes_sent	已发送的消息体字节数
$content_length	HTTP 请求信息里的 Content-Length 字段
$content_type	请求信息里的 Content-Type 字段
$cookie_COOKIE	客户端请求中 COOKIE 头域的值
$document_root	针对当前请求的根路径设置值
$document_uri	与$uri 相同
$host	请求信息中的 Host 头域值，如果请求中没有 Host 行，则等于设置的服务器名
$http_HEADER	HTTP 请求信息里的 HEADER 字段
$http_host	与$host 相同，但如果请求信息中没有 Host 行，则可能不同
$http_cookie	客户端的 cookie 信息
$http_referer	引用地址

$http_user_agent	客户端代理信息
$http_via	最后一个访问服务器的 IP 地址
$http_x_forwarded_for	相当于网络访问路径
$is_args	如果$args 有值，则等于“?”；否则等于空
$limit_rate	对连接速率的限制
$nginx_version	当前 Nginx 服务器的版本
$pid	当前 Nginx 服务器主进程的进程 ID
$query_string	与$args 相同
$remote_addr	客户端 IP 地址
$remote_port	客户端端口号
$remote_user	客户端用户名，用于 Auth Basic Module 验证
$request	客户端请求
$request_body	客户端请求的报文体
$request_body_file	发往后端服务器的本地临时缓存文件的名称
$request_filename	当前请求的文件路径名，由 root 或 alias 指令与 URI 请求生成
$request_method	请求的方法，比如 GET、POST 等
$request_uri	请求的 URI，带参数，不包含主机名
$scheme	所用的协议，如 http 或者 https，比如 rewrite^(.+)$$scheme://mysite.name$1redirect
$sent_http_cache_control	
$sent_http_connection	
$sent_http_content_length	
$sent_http_content_type	
$sent_http_keep_alive	
$sent_http_last_modified	
$sent_http_location	
$sent_http_transfer_encoding	
$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（这样会造成资源浪费）
	$server_name 请求到达的服务器名
$server_port	请求到达的服务器端口号
$server_protocol	请求的协议版本，HTTP/1.0 或 HTTP/1.1
$uri	请求的不带请求参数的 URI，可能和最初的值有不同，比如经过重定向之类的

附录 B

正则表达式语法

Nginx 服务器能很好地支持正则表达式，比如 server 指令就可以接受正则表达式的参数。并且从 Nginx-0.7.40 开始，Nginx 服务器中的正则表达式支持字符串捕获功能，可以使用小括号将正则表达式匹配成功的字符串中的一部分字符串拾取出来，放在指定的变量中供下文使用。

为了大家能够更好地理解本书的内容，在本附录中笔者为大家收录了 Nginx 服务器中正则表达式的一般语法，并列举了简单的使用范例。注意，正则表达式本身具有不同的版本，它们在解析表达式的过程中具有一定的差异。本附录中的语法适用于 Nginx 服务器的配置，其他用途只供参考。在学习正则表达式的过程中，笔者认为主要要在思想上转变，突破以前对“字符串”的认识，化整为零，在正则表达式的世界里，一切皆字符。

正则表达式描述在搜索文本正文时要匹配的一个或多个字符串。该表达式可用作一个将字符模式与要搜索的字符串相匹配的模板，一般包括普通字符（例如，a ~ z 和 A ~ Z 之间的字母、0 ~ 9 之间的数字）和特殊字符（称为“元字符”）等字符，特殊字符包含了单字符元字符的列表以及它们在正则表达式中的行为。若要匹配这些特殊字符，必须首先转义这些字符，即，在字符前面加反斜杠字符“\”。我们需要掌握的是特殊字符的使用。

正则表达式中支持以下特殊字符（元字符）：

*	零次或多次匹配前面的字符或子表达式。等效于 {0,}。 zo* 与“z”和“zoo”匹配。
+	一次或多次匹配前面的字符或子表达式。等效于 {1,}。 zo+ 与“zo”和“zoo”匹配，但与“z”不匹配。
?	零次或一次匹配前面的字符或子表达式。等效于 {0,1}。 当该字符紧随任何其他限定符（*、+、?、{n}、{n,} 或 {n,m}）之后时，匹配

模式是非贪婪的。非贪婪模式匹配搜索到的、尽可能少的字符串，而默认的贪婪模式匹配搜索到的、尽可能多的字符串。

zo? 与“z”和“zo”匹配，但与“zoo”不匹配；o+?只与“oooo”中的单个“o”匹配，而 o+与所有“o”匹配。do(es)?与“do”或“does”中的“do”匹配。

^　匹配搜索字符串开始的位置。如果标志中包括 m（多行搜索）字符，还将匹配\n 或\r 后面的位置。如果将 ^ 用作括号表达式中的第一个字符，则会对字符集求反。

^\d{3}与搜索字符串开始处的 3 个数字匹配。[^abc]与除 a、b 和 c 以外的任何字符匹配。

$　匹配搜索字符串结尾的位置。如果标志中包括 m（多行搜索）字符，还将匹配\n 或\r 前面的位置。

\d{3}$ 与搜索字符串结尾处的 3 个数字匹配。

.　匹配除换行符\n 之外的任何单个字符。若要匹配包括\n 在内的任意字符，请使用诸如[\s\S]之类的模式。

a.c 与“abc”、“a1c”和“a-c”匹配。

[]　标记括号表达式的开始和结尾。

[1-4]与“1”、“2”、“3”或“4”匹配。[^aAeEiIoOuU] 与任何非元音字符匹配。

{}　标记限定符表达式的开始和结尾。

a{2,3}与“aa”和“aaa”匹配。

()　标记子表达式的开始和结尾。Nginx 服务器使用该元字符保存子表达式以备将来之用。

A(\d)与“A0”至“A9”匹配。保存该数字以备将来之用。

|　指示在两个或多个项之间进行选择。

z|food 与“z”或“food”匹配。(z|f)ood 与“zood”或“food”匹配。

/　表示 JScript 中的文本正则表达式模式的开始或结尾。在第二个“/”后添加单字符标志可以指定搜索行为。

/abc/gi 是与“abc”匹配的 JScript 文本正则表达式。g（全局）标志指定查找模式的所有匹配项，还可以使用 i，表示忽略大小写，使搜索不区分大小写。

\　将下一字符标记为特殊字符、文本、反向引用或八进制转义符。

\n 与换行符匹配。\(与“(”匹配。\\ 与“\”匹配。大多数特殊字符在括号表达式内出现时失去它们的意义，并表示普通字符。有关更多信息，请参见匹配字符的列表中的“括号表达式中的字符”。

\b　与一个字边界匹配；即字与空格间的位置。

er\b 与“never”中的“er”匹配，但与“verb”中的“er”不匹配。

\B	非边界字匹配
	er\B 与“verb”中的“er”匹配，但与“never”中的“er”不匹配。
\d	数字字符匹配。等效于[0-9]。
	在搜索字符串“12 345”中，\d{2}与“12”和“34”匹配。\d 与“1”、“2”、“3”、“4”和“5”匹配。
\D	非数字字符匹配。等效于[^0-9]。
	\D+与“abc123 def”中的“abc”和“def”匹配。
\w	与以下任意字符匹配：A-Z、a-z、0-9 和下划线。等效于[A-Za-z0-9_]。
	在字符串“The quick brown fox…”中，\w+与“The”、“quick”、“brown”和“fox”匹配。
\W	与除 A-Z、a-z、0-9 和下划线以外的任意字符匹配。等效于[^A-Za-z0-9_]。在字符串“The quick brown fox…”中，\W+与“…”和所有空格匹配。
[xyz]	字符集。与任何一个指定字符匹配。
	[abc]，与“plain”中的“a”匹配。
[^xyz]	反向字符集。与未指定的任何字符匹配。
	[^abc]，与“plain”中的“p”、“l”、“i”和“n”匹配。
[a-z]	字符范围。匹配指定范围内的任何字符。
	[a-j]，与“a”到“j”范围内的任何小写字母字符匹配。[E-Q]，与“E”到“Q”范围内的任何大写字母字符匹配。
[^a-z]	反向字符范围。与不在指定范围内的任何字符匹配。
	[^a-j]，与不在范围“a”到“z”内的任何字符匹配。[^E-Q]，与不在“E”到“Q”大写字母范围内的任何字符匹配。
{n}	正好匹配 n 次。n 是非负整数。
	o{2}与“Bob”中的“o”不匹配，但与“food”中的两个“o”匹配。
{n,}	至少匹配 n 次。n 是非负整数；*与{0,}相等；+与{1,}相等。
	o{2,}与“Bob”中的“o”不匹配，但与“foooood”中的所有“o”匹配。
{n,m}	匹配至少 n 次，至多 m 次。n 和 m 是非负整数，其中 n<=m。逗号和数字之间不能有空格。?与{0,1}相等。
	在搜索字符串“1234567”中，\d{1,3}与“123”、“456”和“7”匹配。
(*pattern*)	与 *pattern* 匹配并保存匹配项。在 Nginx 服务器的配置文件中，可以通过该方法从匹配的字符串中返回检索的匹配项，以数组元素形式保存。若要匹配括号字符“()”，请使用转义字符，即“\(”或者“\)”。
	(Chapter|Section) [1-9]与“Chapter 5”匹配，保存“Chapter”以备将来之用。
(?:*pattern*)	与 *pattern* 匹配，但不保存匹配项；即不会存储匹配项以备将来之用。这对于用

“or”字符“|”组合模式部件的情况很有用。

industr(?:y|ies)与 industry|industries 相等。

(?=*pattern*)　积极的预测先行。找到一个匹配项后，将在匹配文本之前开始搜索下一个匹配项。不会保存匹配项以备将来之用。

^(?=.*\d).{4,8}$，匹配长度介于 4 到 8 个字符之间，并且必须至少包含一个数字的字符串。在该模式中，.*\d 查找后跟有数字的任意多个字符。对于搜索字符串“abc3qr”，这与“abc3”匹配。从该匹配项之前（而不是之后）开始；{4,8}与包含 4-8 个字符的字符串匹配。这与“abc3qr”匹配。这个正则表达式可以用于判定密码的复杂度。

^和$　指定搜索字符串的开始和结束位置。这将在搜索字符串包含匹配字符之外的任何字符时阻止匹配。

(?!*pattern*)　消极的预测先行。匹配与 *pattern* 不匹配的搜索字符串。找到一个匹配项后，将在匹配文本之前开始搜索下一个匹配项。不会保存匹配项以备将来之用。

\b(?!th)\w+\b 与不以“th”开头的单词匹配。该模式中，\b 与一个字边界匹配。对于搜索字符串“quick”，这与第一个空格匹配。(?!th) 与非“th”字符串匹配。这与“qu”匹配。从该匹配项开始，\w+ 与一个字匹配。这与“quick”匹配。

\cx　匹配 x 指示的控制字符。x 的值必须在 A-Z 或 a-z 范围内。如果不是这样，则假定 c 就是文本“c”字符本身。

\cM　与 Ctrl+M 或一个回车符匹配。

\xn　匹配 n，此处的 n 是一个十六进制转义码。十六进制转义码必须正好是两位数长。允许在正则表达式中使用 ASCII 代码。

\x41 与“A”匹配。\x041 等效于后跟有“1”的“\x04”（因为 n 必须正好是两位数）。

\num　匹配 num，此处的 num 是一个正整数。这是对已保存的匹配项的引用。

(.)\1 与两个连续的相同字符匹配。

\n　标识一个八进制转义码或反向引用。如果\n 前面至少有 n 个捕获子表达式，则 n 是反向引用。否则，如果 n 是八进制数 (0-7)，那么 n 是八进制转义码。

(\d)\1 与两个连续的相同数字匹配。

\nm　标识一个八进制转义码或反向引用。如果\nm 前面至少有 nm 个捕获子表达式，那么 nm 是反向引用。如果\nm 前面至少有 n 个捕获子表达式，则 n 是反向引用，后面跟有文本 m。如果上述情况都不存在，当 n 和 m 是八进制数字(0-7)时，\nm 匹配八进制转义码 nm。

\nml 当 n 是八进制数字(0-3)，m 和 l 是八进制数字(0-7)时，匹配八进制转义码 nml。

\011　与制表符匹配。

\un	匹配 n，其中 n 是以四位十六进制数表示的 Unicode 字符。 \u00A9 与版权符号(©)匹配。
\f	换页符。等效于\x0c 和\cL。
\n	换行符。等效于\x0a 和\cJ。
\r	回车符。等效于\x0d 和\cM。
\t	Tab 字符。等效于\x09 和\cI。
\s	任何空白字符，包括空格、制表符和换页符。等效于[\f\n\r\t\v]。
\S	等效于任何非空白字符。等效于[^\f\n\r\t\v]。
\v	垂直制表符。等效于\x0b 和\cK。