CENTOS LNMP 安装完整笔记

约束规则：

- CENTOS 系统最小安装，不安装 X-windows（减少系统负担）。
- 所有下载的软件均放在/home/soft 下
- Mysql+Nginx+Php 均安装在/usr/local/webserver/下
- 相关的数据库、Logs、网站目录，均在/data#下建立

1 更新和配置 Yum

先进入 yum 源配置目录

```
cd /etc/yum.repos.d
```

备份系统自带的 yum 源

```
mv CentOS-Base.repo CentOS-Base.repo.save
```

下载其他更快的 yum 源

中科大的 yum 源：

```
wget http://centos.ustc.edu.cn/CentOS-Base.repo
```

163 的 yum 源：

```
wget http://mirrors.163.com/.help/CentOS-Base-163.repo
```

sohu 的 yum 源

```
wget http://mirrors.sohu.com/help/CentOS-Base-sohu.repo
```

1.1 使用 yum 程序安装所需开发包（以下为标准的 RPM 包名称）

这里我们将编译安装所必须的一些小软件比如 libpng,libtiff,freetype,libjpeg 等先用 RPM 的方式一并安装好，避免手动编译浪费时间，同时也能避免很多错误，这几个小软件的编译很麻烦。这几个小软件编译错误了，安装当然安装不了，php5 的编译当然也没戏了。所以我们抓大放小，对这些小牛鬼蛇神采取快速简洁的方式进行安装。并且对服务器的性能也不能产生什么影响。

```
yum install gcc gcc-c++ gcc-g77 flex bison autoconf automake make libjpeg libjpeg-devel libpng libpng-devel freetype freetype-devel libxml2 libxml2-devel zlib-devel glibc glibc-devel glib2 glib2-devel bzip2 bzip2-devel ncurses ncurses-devel curl curl-
```

```
devel e2fsprogs e2fsprogs-devel krb5 krb5-devel libidn libidn-devel oepnldap
openldap-devel nss_ldap openldap_clients openldap-servers
```

2. mysql 编译安装

2.1 软件下载

依照约束，进入/home/soft 目录

```
cd /home/soft
wget http://blog.s135.com/soft/linux/nginx_php/mysql/mysql-5.1.38.tar.gz
```

2.2 安装数据库

添加 mysql 用户和组

```
/usr/sbin/groupadd mysql
/usr/sbin/useradd -g mysql mysql
```

开始安装

```
tar -zxvf mysql-5.1.38.tar.gz

cd mysql-5.1.38/

./configure --prefix=/usr/local/webserver/mysql/ --enable-assembler --with-extra-
charsets=complex --enable-thread-safe-client --with-big-tables --with-readline --with-
ssl --with-embedded-server --enable-local-infile --with-plugins=innobase

make

make install
```

添加相关权限

```
chmod +w /usr/local/websever/mysql
chown -R mysql:mysql /usr/local/webserver/mysql
```

2.3 数据库配置

2.3.1 创建数据库存放目录

```
mkdir -p /data0/mysql/3306/data/
chown -R mysql:mysql /data0/mysql/
```

2.3.2 以 mysql 用户账号的身份建立数剧表

```
/usr/local/webserver/mysql/bin/mysql_install_db
--basedir=/usr/local/webserver/mysql –datadir=/data0/mysql/3306/data
--user=mysql
```

2.3.3 创建 my.cnf 配置文件

```
vim /data0/mysql/3306/my.cnf
```

输入内容如下：

```
[client]
# 数据表的默认字符集
default-character-set = utf8
# 端口号
port = 3306
# 为 MySQL 客户程序与服务器之间的本地通信指定一个套接字文件(仅适用于 UNIX/Linux 系统; 默认设置一般是/var/lib/mysql/mysql.sock 文件)
socket = /tmp/mysql.sock

[mysql]
#\u：表示当前登录 mysql client 的用户
#\h：表示当前 mysql 所在的主机
#\R：表示小时（24 制）
#\m：表示分钟
#\s：表示秒数
#\d：表示当前数据库
prompt = "[\\u@\\h-\\R:\\m:\\s-\\d]> "
# 不自动重新进行哈希运算，通过该选项可使 mysql 启动得更快
no-auto-rehash

[mysqld]
#default-character-set = utf8
user = mysql
```

port = 3306

socket = /tmp/mysql.sock

使用给定目录作为根目录(安装目录)

basedir = /usr/local/webserver/mysql

从给定目录读取数据库文件

datadir = /data0/mysql/3306/data/

未知，知道的朋友请留言，谢谢。

open_files_limit = 10240

back_log 是要求 MySQL 能有的连接数量。当主要 MySQL 线程在一个很短时间内得到非常多的连接请求，这就起作用，然后主线程花些时间(尽管很短)检查连接并且启动一个新线程。back_log 值指出在 MySQL 暂时停止回答新请求之前的短时间内多少个请求可以被存在堆栈中。如果期望在一个短时间内有很多连接，你需要增加它。也就是说，如果 MySql 的连接数据达到 max_connections 时，新来的请求将会被存在堆栈中，以等待某一连接释放资源，该堆栈的数量即 back_log，如果等待连接的数量超过 back_log，将不被授予连接资源。另外，这值（back_log）限于您的操作系统对到来的 TCP/IP 连接的侦听队列的大小。你的操作系统在这个队列大小上有它自己的限制（可以检查你的 OS 文档找出这个变量的最大值），试图设定 back_log 高于你的操作系统的限制将是无效的。

back_log = 600

max_connections 是指 MySql 的最大连接数，如果服务器的并发连接请求量比较大，建议调高此值，以增加并行连接数量，当然这建立在机器能支撑的情况下，因为如果连接数越多，介于 MySql 会为每个连接提供连接缓冲区，就会开销越多的内存，所以要适当调整该值，不能盲目提高设值。可以过'conn%'通配符查看当前状态的连接数量，以定夺该值的大小。

max_connection = 200

由于出现某台 host 连接错误次数等于 max_connect_errors（默认 10），主机'host_name'再次尝试时被屏蔽。

可有效反的防止 dos 攻击，使用'mysqladmin flush-hosts'解除屏蔽。

max_connect_errors = 100

指定表高速缓存的大小。每当 MySQL 访问一个表时，如果在表缓冲区中还有空间，该表就被打开并放入其中，这样可以更快地访问表内容。

table_cache = 614

未知，知道的朋友请留言，谢谢。

external-locking =FALSE

当 MySQL 客户端或 mysqld 服务器收到大于 max_allowed_packet 字节的信息包时，将发出“信息包过大”错误，并关闭连接。对于某些客户端，如果通信信息包过大，在执行查询期间，可能会遇到“丢失与 MySQL 服务器的连接”错误。客户端和服务器均有自己的 max_allowed_packet 变量，因此，如你打算处理大的信息包，必须增加客户端和服务器上的该变量。一般情况下，服务器默认 max-allowed-packet 为 1MB

max_allowed_packet = 32M

Sort_Buffer_Size 是一个 connection 级参数，在每个 connection 第一次需要使用这个 buffer 的时候，一次性分配设置的内存。Sort_Buffer_Size 并不是越大越好，由于是 connection 级的参数，过大的设置+高并发可能会耗尽系统内存资源。文档说“On Linux, there are thresholds of 256KB and 2MB where larger values may significantly slow down memory allocation”

sort_buffer_size = 2M

在参加 JOIN 操作的数据列没有索引时为 JOIN 操作分配的缓存区长度(默认设置是 128K)。

join_buffer_size = 2M

#这个值表示可以重新利用保存在缓存中线程的数量,当断开连接时如果缓存中还有空间,那么客户端的线程将被放到缓存中,如果线程重新被请求，那么请求将从缓存中读取,如果缓存中是空的或者是新的请求，那么这个线程将被重新创建,如果有很多新的线程，增加这个值可以改善系统性能

#根据物理内存设置规则如下：

1G ---> 8

2G ---> 16

3G ---> 32

3G ---> 64

thread_cache_size = 16

该参数取值为服务器逻辑 CPU 数量×2，也有文章说这个变量是针对 Solaris 系统的，如果设置这个变量的话，mysqld 就会调用 thr_setconcurrency()。这个函数使应用程序给同一时间运行的线程系统提供期望的线程数目。求解

thread_concurrency = 8

查询缓存区的最大长度(默认设置是 0，不开辟查询缓存区)Magento 推荐值为 64。

query_cache_size = 32M

允许临时存放在查询缓存区里的查询结果的最大长度(默认设置是 1M)Magento 推荐值为 2M。

query_cache_limit = 2M

每次给 Query Cache 结果分配内存的大小，默认为 4K

query_cache_min_res_unit = 2k

默认使用 MyISAM 数据库

default-storage-engine = MyISAM

默认使用的数据表类型

default_table_type = MyISAM

线程使用的堆大小。

thread_stack = 192k

设定默认的事务隔离级别

transaction_isolation = READ-COMMITTED

内存中临时表的最大大小，如果一个表会比该表更大，则自动转换为磁盘表，此限制是针对单个表，不是总和。

tmp_table_size = 64M

HEAP 数据表的最大长度(默认设置是 16M); 超过这个长度的 HEAP 数据表将被存入一个临时文件而不是驻留在内存里。

max_heap_table_size = 16M

慢查询的执行用时上限(默认设置是 10s)。不要在这里使用“1”，否则会导致所有的查询，甚至非常快的查询页被记录下来

long_query_time = 2

在慢速日志中记录更多信息

log_long_fromat

把对 Data 进行修改的所有 SQL 语言规则命令(也就是 INSERT、UPDATE 和 DELETE 命令)以二进制格式记入日志(二进制变更日 志，binary update log)。这种日志的文档名是 filename.n 或默认的 hostname.n，其中 n 是一个 6 位数字的整数(日志文档按顺序编号)。

log-bin = /data0/mysql/3306/binlog

高速缓存保存的二进制日志中的 SQL 语句的大小

binlog_cache_size = 4M

混合模式复制

binlog_format = MIXED

max_binlog_cache_size = 8M

max_binlog_size = 512M

二进制日志自动删除的天数。默认值为 0,表示”没有自动删除 “。

expire_logs_days = 7

#为了最小化磁盘的 I/O ， MyISAM 存储引擎的表使用键高速缓存来缓存索引，这个键高速缓存的大小则通过 key-buffer-size 参数来设置。

key_buffer_size = 256M

为从数据表顺序读取数据的读操作保留的缓存区的长度(默认设置是 128KB)。

read_buffer_size = 1M

类似于 read_buffer_size 选项，但针对的是按某种特定顺序(比如使用了 ORDER BY 子句的查询)输出的查询结果(默认设置是 256K)。

read_rnd_buffer_size = 16M

为一次插入多条新记录的 INSERT 命令分配的缓存区长度(默认设置是 8M)。

bulk_insert_buffer_size = 64M

MyISAM 表发生变化时重新排序所需的缓冲

myisam_sort_buffer_size = 128M

如果临时文件会变得超过索引，不要使用快速排序索引方法来创建一个索引。注释：这个参数以字节的形式给出。

myisam_max_sort_file_size = 10G

未知，求解。

myisam_max_extra_sort_file_size = 10G

未知，求解。

myisam_repair_threads = 1

未知，求解。

myisam_recover

禁止 MySQL 对外部连接进行 DNS 解析，使用这一选项可以消除 MySQL 进行 DNS 解析的时间。但需要注意，如果开启该选项，则所有远程主机连接授权都要使用 IP 地址方式，

否则 MySQL 将无法正常处理连接请求！

skip-name-resolve

如果与主控服务器的连接没有成功，则等待 n 秒(s)后再进行管理方式(默认设置是 60s)。

master-connect-retry = 10

从属服务器上的镜像工作就可能人为发生错误而中断，中断后需要有人工参与才能继续进行。

slave-skip-errors = 1032,1062,126,1114,1146,1048,1396

给服务器分配一个独一无二的 ID 编号; n 的取值范围是 1~2 的 32 次方启用二进制日志功能

server-id = 1

这个参数用来设置 InnoDB 存储的数据目录信息和其它内部数据结构的内存池大小。应用程序里的表越多，你需要在这里分配越多的内存。对于一个相对稳定的应用，这个参数的大小也是相对稳定的，也没有必要预留非常大的值。如果 InnoDB 用光了这个池内的内存， InnoDB 开始从操作系统分配内存，并且往 MySQL 错误日志写警告信息。默认值是 1MB ，当发现错误日志中已经有相关的警告信息时，就应该适当的增加该参数的大小。

innodb_additional_mem_pool_size = 16M

缓冲池大小，官方的建议是不要超过 2G。

innodb_buffer_pool_size = 1024M

未知，求解。

innodb_data_file_path = ibdata1:1024M:autoextend

文件读写 I O数，这个参数只在 Windows 上起作用。在 LINUX 上只会等于 4 。

innodb_file_io_threads = 4

并发线程。2*(内核数量+磁盘数量)

innodb_thread_concurrency = 4

1 表示每次事务结束都写日志并刷新磁盘；

2 表示每次事务写日志但不刷新磁盘(每秒刷新)；

0(默认值)表示每秒写日志并刷新磁盘。0 表示最多丢失 1 秒的数据，但性能最好。

innodb_flush_log_at_trx_commit = 2

#日志组中的每个日志文件的大小(单位 MB)。如果 n 是日志组(innodb_log_files_in_group)中日志文件的数目，那么理想的数值为 1M 至缓冲池(innodb_log_buffer_size)大小的

```
1/n。较 大的值，可以减少刷新缓冲池的次数，从而减少磁盘 I/O。但是大的日志 文件意味着在崩溃时需要更长的时间来恢复数据。
innodb_log_file_size = 128M

# InnoDB 将日志写入日志磁盘文件前的缓冲大小。理想值为 1M 至 8M。大的日志 缓冲允许事务运行时不需要将日志保存入磁盘而只到事务被提交(commit)。 因此，如果有大的事务，设置大的日志缓冲可以减少磁盘 I/O。
innodb_log_buffer_size = 16M

# 日志组中的日志文件数目。InnoDB 以环型方式(circular fashion)写入文件。数值“3”被推荐使用。
innodb_log_files_in_group = 3

# 当 innodb 的内存分配过大，致使 Swap 占用严重时，可以适当的减小调整这个值，使达到 Swap 空间释放出来。建义：这个值最大在 90%，最小在 15%。太大，缓存中每次更新需要致换数据页太多，太小，放的数据页太小，更新操作太慢。
innodb_max_dirty_pages_pct = 90

# InnoDB 行锁导致的死锁等待时间(默认值是 50S)
innodb_lock_wait_timeout = 120

# 关闭独享表空间
innodb_file_per_table = 0


[mysqldump]
quick
max_allowed_packet = 32M
```

2.3.4 创建管理 MYSQL 的 SHELL 脚本

```
vim /data0/mysql/3306/mysql
```

输入内容如下：

```
#!/bin/sh


mysql_port=3306
mysql_username="admin"
mysql_password="windking"
```

```
function_start_mysql()
{
    printf "Starting MySQL...\n"
    /bin/sh /usr/local/webserver/mysql/bin/mysqld_safe --defaults-file=/data0/mysql/${mysql_port}/my.cnf 2>&1 > /dev/null &
}

function_stop_mysql()
{
    printf "Stoping MySQL...\n"
    /usr/local/webserver/mysql/bin/mysqladmin -u ${mysql_username} -p${mysql_password} -S /tmp/mysql.sock shutdown
}

function_restart_mysql()
{
    printf "Restarting MySQL...\n"
    function_stop_mysql
    sleep 5
    function_start_mysql
}

function_kill_mysql()
{
    kill -9 $(ps -ef | grep 'bin/mysqld_safe' | grep ${mysql_port} | awk '{printf $2}')
    kill -9 $(ps -ef | grep 'libexec/mysqld' | grep ${mysql_port} | awk '{printf $2}')
}

if [ "$1" = "start" ]; then
```

```
    function_start_mysql
elif [ "$1" = "stop" ]; then
    function_stop_mysql
elif [ "$1" = "restart" ]; then
function_restart_mysql
elif [ "$1" = "kill" ]; then
function_kill_mysql
else
    printf "Usage: /data0/mysql/${mysql_port}/mysql {start|stop|restart|kill}\n"
# chkconfig: 2345 64 36
# description: A very fast and reliable SQL database engine.
fi
```

2.3.5 赋予 mysql 启动脚本执行权限，并启动 MySQL

```
chmod +x /data0/mysql/3306/mysql
/data0/mysql/3306/mysql start
```

通过命令行登录管理 MySQL 服务器（提示输入密码时直接回车）：

```
/usr/local/webserver/mysql/bin/mysql -u admin -p -S /tmp/mysql.sock
```

执行下列语句，创建一个具有 root 权限的用户

```
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' IDENTIFIED BY 'windking';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'127.0.0.1' IDENTIFIED BY 'windking';
```

2.3.6 将 mysql 加入系统服务并添加开机自启动功能

```
ln -s /data0/mysql/3306/mysql /etc/init.d/mysqld
vim /data0/mysql/3306/mysql
```

在 fi 前面加入两行代码：

#chkconfig 64 24

#description: 数剧库啦

```
chkconfig --add mysqld
chkconfig mysqld on
```

注：加入系统服务

每个被 chkconfig 管理的服务需要在对应的 init.d 下的脚本加上两行或者更多行的注释。第

一行告诉 chkconfig 缺省启动的运行级以及启动和停止的优先级。如果某服务缺省不再任何运行级启动，那么使用 - 代替运行级。第二行对服务进行描述，可以用\跨行注释。

例如，random.init 包含三行：

#chkconfig:2345 20 80

#description:Saves and restores system entropy pool for \

#higher quality random number generation.

附加介绍一下 Linux 系统的运行级的概念：

linux 中有多种运行级，常见的就是多用户的 2，3，4，5，很多人知道 5 是运行 x-windows 的级别，而 0 就是关机了。运行级的改变可以通过 init 命令切换。

至此，Mysql 安装完成。

3. 编译安装 PHP（FastCGI 模式）

3.1 软件下载

```
wget http://blog.s135.com/soft/linux/nginx_php/php/php-5.2.10.tar.gz
```

```
wget http://blog.s135.com/soft/linux/nginx_php/phpfpm/php-5.2.10-fpm-0.5.11.diff.gz
```

libiconv 库为需要做转换的应用提供了一个 iconv()的函数，以实现一个字符编码到另一个字符编码的转换

```
wget http://blog.s135.com/soft/linux/nginx_php/libiconv/libiconv-1.13.tar.gz
```

libmcrypt 是加密算法扩展库。支持 DES, 3DES, RIJNDAEL, Twofish, IDEA, GOST, CAST-256, ARCFOUR, SERPENT, SAFER+等算法。

```
wget http://blog.s135.com/soft/linux/nginx_php/mcrypt/libmcrypt-2.5.8.tar.gz
```

mcrypt、mhash 也是加密解密扩展库。

```
wget http://blog.s135.com/soft/linux/nginx_php/mcrypt/mcrypt-2.6.8.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/mhash/mhash-0.9.9.tar.gz
```

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。简单的说就是将数数据调用到内存中，然后从内存中读取，从而大大提高读取速度。

```
wget http://blog.s135.com/soft/linux/nginx_php/memcache/memcache-2.2.5.tgz
```

perl 语言兼容正则表达式，是一个用 C 语言编写的正则表达式函数库。

```
wget http://blog.s135.com/soft/linux/nginx_php/pcre/pcre-7.9.tar.gz
```

eAccelerator 是一个自由开放源码 php 加速器，优化和动态内容缓存，提高了 php 脚本的缓存性能，使得 PHP 脚本在编译的状态下，对服务器的开销几乎完全消除。 它还有对脚本起优化作用，以加快其执行效率。使您的 PHP 程序代码执效率能提高 1-10 倍。

```
wget http://blog.s135.com/soft/linux/nginx_php/eaccelerator/eaccelerator-0.9.5.3.tar.bz2
```

POD（PHP Data Object）该扩展提供 PHP 内置类 PDO 来对数据库进行访问，不同数据库使用相同的方法名，解决数据库连接不统一的问题。

```
wget http://blog.s135.com/soft/linux/nginx_php/pdo/PDO_MYSQL-1.0.2.tgz
```

ImageMagick 是一套功能强大、稳定而且免费的工具集和开发包，可以用来读、写和处理超过 89 种基本格式的图片文件，包括流行的 TIFF, JPEG, GIF, PNG, PDF 以及 PhotoCD 等格式。利用 ImageMagick，你可以根据 web 应用程序的需要动态生成图片, 还可以对一个（或一组）图片进行改变大小、旋转、锐化、减色或增加特效等操作，并将操作的结果以相同格式或其它格式保存。

```
wget http://blog.s135.com/soft/linux/nginx_php/imagick/ImageMagick.tar.gz
```

一个可以供 PHP 调用 ImageMagick 功能的 PHP 扩展。使用这个扩展可以使 PHP 具备和 ImageMagick 相同的功能。

```
wget http://blog.s135.com/soft/linux/nginx_php/imagick/imagick-2.2.2.tgz
```

3.2 安装 PHP

```
tar zxvf php-5.2.10.tar.gz
```

加入 php-fpm 补丁

```
gzip -cd php-5.2.10-fpm-0.5.11.diff.gz | patch -d php-5.2.10 -p1

cd php-5.2.10/

./configure --prefix=/usr/local/webserver/php --with-config-file-path=/usr/local/webserver/php/etc --with-mysql=/usr/local/webserver/mysql --with-mysqli=/usr/local/webserver/mysql/bin/mysql_config --with-iconv-dir=/usr/local --with-freetype-dir --with-jpeg-dir --with-png-dir --with-zlib --with-libxml-dir=/usr --enable-xml --disable-rpath --enable-discard-path --enable-safe-mode --enable-
```

```
bcmath --enable-shmop --enable-sysvsem --enable-inline-optimization --with-curl
--with-curlwrappers --enable-mbregex --enable-fastcgi --enable-fpm --enable-force-
cgi-redirect --enable-mbstring --with-mcrypt --with-gd --enable-gd-native-ttf --with-
openssl --with-mhash --enable-pcntl --enable-sockets --with-ldap --with-ldap-sasl
--with-xmlrpc --enable-zip --enable-soap –without-pear

make ZEND_EXTRA_LIBS='-liconv'

make install

cp php.ini-dist /usr/local/webserver/php/etc/php.ini

cd ../
```

3.2.1 编译安装 PHP5 扩展模块

```
tar zxvf memcache-2.2.5.tgz

cd memcache-2.2.5/

/usr/local/webserver/php/bin/phpize
./configure –with-php-config=/usr/local/webserver/php/bin/php-config

make

make install

cd ../
```

```
tar jxvf eaccelerator-0.9.5.3.tar.bz2

cd eaccelerator-0.9.5.3/

/usr/local/webserver/php/bin/phpize
./configure --enable-eaccelerator=shared –with-php-
config=/usr/local/webserver/php/bin/php-config

make

make install

cd ../
```

```
tar zxvf PDO_MYSQL-1.0.2.tgz

cd PDO_MYSQL-1.0.2/

/usr/local/webserver/php/bin/phpize
./configure --with-php-config=/usr/local/webserver/php/bin/php-config –with-pdo-mysql=/usr/local/webserver/mysql

make

make install
cd ../
```

```
tar zxvf ImageMagick.tar.gz

cd ImageMagick-6.5.1-2/

./configure

make

make install

cd ../
```

```
tar zxvf imagick-2.2.2.tgz

cd imagick-2.2.2/

/usr/local/webserver/php/bin/phpize
./configure –with-php-config=/usr/local/webserver/php/bin/php-config

make

make install

cd ../
```

3.3 PHP 配置

3.3.1 修改 php.ini 文件

手工修改：查找/usr/local/webserver/php/etc/php.ini 中的 extension_dir = "./"

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

并在此行后增加以下几行，然后保存：

extension = "memcache.so"

extension = "pdo_mysql.so"

extension = "imagick.so"

再查找 output_buffering = Off

修改为 output_buffering = On

3.3.2 配置 eAccelerator 加速 PHP：

```
mkdir -p /usr/local/webserver/eaccelerator_cache
vim /usr/local/webserver/php/etc/php.ini
```

按 shift+g 键跳到配置文件的最末尾，加上以下配置信息：

```
[eaccelerator]
zend_extension="/usr/local/webserver/php/lib/php/extensions/no-debug-non-zts-20060613/eaccelerator.so"
eaccelerator.shm_size="64"
eaccelerator.cache_dir="/usr/local/webserver/eaccelerator_cache"
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"
```

3.3.3 创建 www 用户和组，以及虚拟主机使用的目录：

/usr/sbin/groupadd www

/usr/sbin/useradd -g www www

mkdir -p /data0/htdocs/web

chmod +w /data0/htdocs/web

chown -R www:www /data0/htdocs/web

3.3.4 创建 php-fpm 配置文件

php-fpm 是为 PHP 打的一个 FastCGI 管理补丁，可以平滑变更 php.ini 配置而无需重启 php-cgi：

在/usr/local/webserver/php/etc/目录中创建 php-fpm.conf 文件：

rm -f /usr/local/webserver/php/etc/php-fpm.conf

vi /usr/local/webserver/php/etc/php-fpm.conf

输入以下内容（如果您安装 Nginx + PHP 用于程序调试，请将以下的<value name="display_errors">0</value>改为<value name="display_errors">1</value>，以便显示 PHP 错误信息，否则，Nginx 会报状态为 500 的空白错误页）：

```
<?xml version="1.0" ?>
<configuration>

 All relative paths in this config are relative to php's install prefix

 <section name="global_options">

  Pid file
  <value name="pid_file">/usr/local/webserver/php/logs/php-fpm.pid</value>

  Error log file
  <value name="error_log">/usr/local/webserver/php/logs/php-fpm.log</value>

  Log level
  <value name="log_level">notice</value>

  When this amount of php processes exited with SIGSEGV or SIGBUS ...
```

```
    <value name="emergency_restart_threshold">10</value>

    ... in a less than this interval of time, a graceful restart will be initiated.
    Useful to work around accidental curruptions in accelerator's shared memory.
    <value name="emergency_restart_interval">1m</value>

    Time limit on waiting child's reaction on signals from master
    <value name="process_control_timeout">5s</value>

    Set to 'no' to debug fpm
    <value name="daemonize">yes</value>

  </section>

  <workers>

    <section name="pool">

      Name of pool. Used in logs and stats.
      <value name="name">default</value>

      Address to accept fastcgi requests on.
      Valid syntax is 'ip.ad.re.ss:port' or just 'port' or '/path/to/unix/socket'
      <value name="listen_address">127.0.0.1:9000</value>

      <value name="listen_options">

        Set listen(2) backlog
        <value name="backlog">-1</value>

        Set permissions for unix socket, if one used.
        In Linux read/write permissions must be set in order to allow connections from
web server.
        Many BSD-derrived systems allow connections regardless of permissions.
        <value name="owner"></value>
```

```
  <value name="group"></value>
  <value name="mode">0666</value>
</value>

Additional php.ini defines, specific to this pool of workers.
<value name="php_defines">
  <value name="sendmail_path">/usr/sbin/sendmail -t -i</value>
  <value name="display_errors">1</value>
</value>

Unix user of processes
  <value name="user">www</value>

Unix group of processes
  <value name="group">www</value>

Process manager settings
<value name="pm">

  Sets style of controling worker process count.
  Valid values are 'static' and 'apache-like'
  <value name="style">static</value>

  Sets the limit on the number of simultaneous requests that will be served.
  Equivalent to Apache MaxClients directive.
  Equivalent to PHP_FCGI_CHILDREN environment in original php.fcgi
  Used with any pm_style.
  <value name="max_children">128</value>

  Settings group for 'apache-like' pm style
  <value name="apache_like">

    Sets the number of server processes created on startup.
    Used only when 'apache-like' pm_style is selected
    <value name="StartServers">20</value>
```

Sets the desired minimum number of idle server processes.
Used only when 'apache-like' pm_style is selected
<value name="MinSpareServers">5</value>

Sets the desired maximum number of idle server processes.
Used only when 'apache-like' pm_style is selected
<value name="MaxSpareServers">35</value>

</value>

</value>

The timeout (in seconds) for serving a single request after which the worker process will be terminated
Should be used when 'max_execution_time' ini option does not stop script execution for some reason
'0s' means 'off'
<value name="request_terminate_timeout">0s</value>

The timeout (in seconds) for serving of single request after which a php backtrace will be dumped to slow.log file
'0s' means 'off'
<value name="request_slowlog_timeout">0s</value>

The log file for slow requests
<value name="slowlog">logs/slow.log</value>

Set open file desc rlimit
<value name="rlimit_files">65535</value>

Set max core size rlimit
<value name="rlimit_core">0</value>

Chroot to this directory at the start, absolute path

```
    <value name="chroot"></value>

    Chdir to this directory at the start, absolute path
    <value name="chdir"></value>

    Redirect workers' stdout and stderr into main error log.
    If not set, they will be redirected to /dev/null, according to FastCGI specs
    <value name="catch_workers_output">yes</value>

    How much requests each process should execute before respawn.
    Useful to work around memory leaks in 3rd party libraries.
    For endless request processing please specify 0
    Equivalent to PHP_FCGI_MAX_REQUESTS
    <value name="max_requests">102400</value>

    Comma separated list of ipv4 addresses of FastCGI clients that allowed to connect.
    Equivalent to FCGI_WEB_SERVER_ADDRS environment in original php.fcgi (5.2.2+)
    Makes sense only with AF_INET listening socket.
    <value name="allowed_clients">127.0.0.1</value>

    Pass environment variables like LD_LIBRARY_PATH
    All $VARIABLEs are taken from current environment
    <value name="environment">
      <value name="HOSTNAME">$HOSTNAME</value>
      <value name="PATH">/usr/local/bin:/usr/bin:/bin</value>
      <value name="TMP">/tmp</value>
      <value name="TMPDIR">/tmp</value>
      <value name="TEMP">/tmp</value>
      <value name="OSTYPE">$OSTYPE</value>
      <value name="MACHTYPE">$MACHTYPE</value>
      <value name="MALLOC_CHECK_">2</value>
    </value>

</section>
```

```
  </workers>

</configuration>
```

3.3.5 启动 php-cgi 进程

监听 127.0.0.1 的 9000 端口，进程数为 200（如果服务器内存小于 3GB，可以只开启 64 个进程），用户为 www：

```
ulimit -SHn 65535
/usr/local/webserver/php/sbin/php-fpm start
```

注：/usr/local/webserver/php/sbin/php-fpm 还有其他参数，包括：start|stop|quit|restart|reload|logrotate，修改 php.ini 后不重启 php-cgi，重新加载配置文件使用 reload。

关于 ulimit -SHn

Linux 默认打开文件数为 1024 个，通过 ulimit -a 可以查看 open files

显然这个数值在高并发的 WEB 服务器上是不够用的，根据您的需要可使用 ulimit 命令修改此值，如：

ulimit -SHn 65535

修改后，重启服务器将失效，为使开机即生效，可修改/etc/security/limits.conf

```
vim /etc/security/limits.conf
```

在：

```
#*              soft    core            0
#*              hard    rss             10000
#@student       hard    nproc           20
#@faculty       soft    nproc           20
#@faculty       hard    nproc           50
#ftp            hard    nproc           0
#@student       -       maxlogins       4
```

后加入这两行：

```
* soft nofile 65532  #Linux 对当前用户的进程同时打开的文件数量的软限制(soft limit)
* hard nofile 65533  #Linux 对当前用户的进程同时打开的文件数量的硬限制(hardlimit)
# End of file
```

就可以不用每次重启服务器的时候，都输入该命令了。

4. 编译安装 Nginx

4.1 软件下载

```
wget http://blog.s135.com/soft/linux/nginx_php/nginx/nginx-0.8.15.tar.gz
```

perl 语言兼容正则表达式，是一个用 C 语言编写的正则表达式函数库。

```
wget http://blog.s135.com/soft/linux/nginx_php/pcre/pcre-7.9.tar.gz
```

4.2 安装 Nginx

4.2.1 安装 Nginx 所需的 pcre 库

```
tar zxvf pcre-7.9.tar.gz

cd pcre-7.9/

./configure

make

make install

cd ../
```

4.2.2 安装 Nginx

```
tar zxvf nginx-0.8.15.tar.gz

cd nginx-0.8.15/

./configure --user=www --group=www --prefix=/usr/local/webserver/nginx --with-
http_stub_status_module –with-http_ssl_module

make

make install

cd ../
```

4.3 Nginx 配置

4.3.1 创建 Nginx 日志目录

```
mkdir -p /data1/logs
chmod +w /data1/logs
chown -R www:www /data1/logs
```

4.3.2 创建 Nginx 配置文件

在/usr/local/webserver/nginx/conf/目录中创建 nginx.conf 文件：

```
rm -f /usr/local/webserver/nginx/conf/nginx.conf
vim /usr/local/webserver/nginx/conf/nginx.conf
```

输入以下内容：

```
user  www www;

worker_processes 8;

error_log  /data1/logs/nginx_error.log  crit;

pid        /usr/local/webserver/nginx/nginx.pid;

#指定最大文件描述数量.
worker_rlimit_nofile 65535;

events
{
  # 使用网络 I/O 模型，Linux 系统推荐 epoll 模型；
  use epoll;
  # 允许的连接数
  worker_connections 65535;
}

http
```

```
{
  include       mime.types;
  default_type  application/octet-stream;

  #charset  gb2312;

  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;

  keepalive_timeout 60;

  tcp_nodelay on;

  fastcgi_connect_timeout 300;
  fastcgi_send_timeout 300;
  fastcgi_read_timeout 300;
  fastcgi_buffer_size 64k;
  fastcgi_buffers 4 64k;
  fastcgi_busy_buffers_size 128k;
  fastcgi_temp_file_write_size 128k;

  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;

#limit_zone  crawler  $binary_remote_addr  10m;

server
{
  listen       80;
  server_name  192.168.1.17;
  index index.html index.htm index.php;
  root  /data0/htdocs/web;

  #limit_conn   crawler  20;

  location ~ .*\.(php|php5)?$
  {
    #fastcgi_pass  unix:/tmp/php-cgi.sock;
    fastcgi_pass  127.0.0.1:9000;
    fastcgi_index index.php;
    include fcgi.conf;
  }

  location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
  {
```

```
    expires      30d;
  }

  location ~ .*\.(js|css)?$
  {
    expires      1h;
  }

  log_format  access  '$remote_addr - $remote_user [$time_local] "$request" '
            '$status $body_bytes_sent "$http_referer" '
            '"$http_user_agent" $http_x_forwarded_for';
  access_log  /data1/logs/access.log  access;
    }

# 要增加多的虚拟项目，格式如下：
 # server
 # {
 #   listen       80;
 #   server_name  www.s135.com;
 #   index index.html index.htm index.php;
 #   root  /data0/htdocs/www;

 #   location ~ .*\.(php|php5)?$
 #   {
 #     #fastcgi_pass  unix:/tmp/php-cgi.sock;
 #     fastcgi_pass  127.0.0.1:9000;
 #     fastcgi_index index.php;
```

```
#    include fcgi.conf;
#  }

#  log_format  wwwlogs  '$remote_addr - $remote_user [$time_local] "$request" '
#             '$status $body_bytes_sent "$http_referer" '
#             '"$http_user_agent" $http_x_forwarded_for';
#  access_log  /data1/logs/wwwlogs.log  wwwlogs;
# }

 server
 {
   listen  80;
   server_name  192.168.1.17;

   location / {
   stub_status on;
   access_log   off;
   }
 }
}
```

在/usr/local/webserver/nginx/conf/目录中创建 fcgi.conf 文件：

```
vi /usr/local/webserver/nginx/conf/fcgi.conf
```

输入以下内容：

```
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;

# PHP only, required if PHP was built with --enable-force-cgi-redirect
fastcgi_param  REDIRECT_STATUS    200;
```

4.3.3 启动 Nginx

```
/usr/local/webserver/nginx/sbin/nginx
```

注：修改/usr/local/webserver/nginx/conf/nginx.conf 配置文件后，请执行以下命令检查配置文件是否正确：

```
/usr/local/webserver/nginx/sbin/nginx -t
```

如果屏幕显示以下两行信息，说明配置文件正确：

```
the configuration file /usr/local/webserver/nginx/conf/nginx.conf syntax is ok
the configuration file /usr/local/webserver/nginx/conf/nginx.conf was tested
successfully
```

使用 HUP 让 Nginx 平滑重启

```
kill -HUP `cat /usr/local/webserver/nginx/nginx.pid`
```

4.3.4 将 nginx+PHP 加入开机启动

vi /etc/rc.local

在末尾增加以下内容：

```
/usr/local/webserver/php/sbin/php-fpm start
/usr/local/webserver/nginx/sbin/nginx
```

至此，所有软件都已安装完成。