

第4章

Nginx 与 PHP (FastCGI) 的安装、配置与优化

在互联网服务器架构中，我们经常可以听到 LAMP (Linux+Apache+Mysql+Perl/PHP/Python) 架构。LAMP 这个特定名词最早出现在 1998 年。当时，Michael Kunze 为德国计算机杂志《c't》写作的一篇关于自由软件如何成为商业软件替代品的文章时，创建了 LAMP 这个名词，用来指代 Linux 操作系统、Apache 网络服务器、MySQL 数据库和 PHP (Perl 或 Python) 脚本语言的组合（由 4 种技术开头的字母组成）。由于 IT 世界众所周知的对缩写的爱好，Michael Kunze 提出的 LAMP 这一术语很快就被市场接受。O'Reilly 和 MySQL AB 更是在英语人群中推广普及了这个术语。随之 LAMP 技术成了开源软件业的一盏真正的明灯。

虽然这些开放源代码程序本身并不是专门设计成同另外几个程序一起工作的，但由于它们都是影响较大的开源软件，拥有很多共同特点，这就导致了这些组件经常在一起使用。在过去的几年里，这些组件的兼容性不断完善，在一起应用的情形变得更加普遍。并且它们为了改善不同组件之间的协作，已经创建了某些扩展功能。目前，几乎在所有的 Linux 发布版中都默认包含了这些产品。Linux 操作系统、Apache 服务器、MySQL 数据库和 Perl、PHP 或 Python 语言，这些产品共同组成了一个强大的 Web 应用程序平台。

现在，由于 Nginx 拥有超越 Apache 的卓越性能，LAMP 架构正在逐渐被 LNMP 架构所取代。LNMP 即为我们本章详细介绍的 Linux+Nginx+Mysql+PHP 架构。本章我们将一步步学习 Nginx

与 PHP (FastCGI) 的安装、配置与优化。

提高 PHP (FastCGI)，那么 FastCGI 是什么呢？

FastCGI 是语言无关的、可伸缩架构的 CGI 开放扩展，其主要行为是将 CGI 解释器进程保持在内存中并因此获得较高的性能。众所周知，CGI 解释器的反复加载是 CGI 性能低下的主要原因，如果 CGI 解释器保持在内存中并接受 FastCGI 进程管理器调度，则可以提供良好的性能、伸缩性、Fail-Over 特性等。

FastCGI 的工作原理是：

(1) FastCGI 进程管理器自身初始化，启动多个 CGI 解释器进程（多个 php-cgi 进程）并等待来自 Web Server 的连接。在本文中，采用 PHP-FPM 进程管理器启动多个 php-cgi FastCGI 进程。启动 php-cgi FastCGI 进程时，可以配置以 TCP 和 UNIX 套接字两种方式启动。

(2) 当客户端请求到达 Web 服务器 (Nginx) 时，Web 服务器将请求采用 TCP 协议或 UNIX 套接字方式转发到 FastCGI 主进程，FastCGI 主进程选择并连接到一个 CGI 解释器（子进程）。Web 服务器将 CGI 环境变量和标准输入发送到 FastCGI 子进程 php-cgi。

(3) FastCGI 子进程完成处理后将标准输出和错误信息从同一连接返回 Web 服务器 (Nginx)。当 FastCGI 子进程关闭连接时，请求便告知处理完成。FastCGI 子进程接着等待并处理来自 FastCGI 进程管理器的下一个连接。而在一般的普通 CGI 模式中，php-cgi 在此便退出了。

所以，你可以想象普通的 CGI 模式有多慢。每一个 Web 请求 PHP 都必须重新解析 php.ini、重新载入全部扩展并重新初始化全部数据结构。使用 FastCGI，所有这些都只在进程启动时发生一次。一个额外的好处是，持续数据库连接 (Persistent database connection) 可以工作。

PHP FastCGI 的优点：

(1) PHP 脚本运行速度更快。PHP 解释程序被载入内存而不用每次需要时从存储器读取，此举极大提升了依靠脚本运行站点的性能。

(2) 须要使用的系统资源更少。由于服务器不用在每次需要时都载入 PHP 解释程序，你可以将站点的传输速度提升很多而不必增加 CPU 负担。

(3) 不需要对现有的代码作任何改变。运行在 Apache+PHP 上的程序，无须修改即可适用于 PHP 的 FastCGI。

接下来，我们开始安装、搭建 LNMP (Linux+Nginx+Mysql+PHP) 平台。本文中的操作系统环境为：CentOS Linux 5.3 (Linux 2.6+内核)，另在 RedHat AS4 上也可安装成功。

4.1 获取相关开源程序

1. 【适用 CentOS 操作系统】

利用 CentOS Linux 系统自带的 yum 命令安装、升级所需的程序库（RedHat 等其他 Linux 发行版可从安装光盘中找到这些程序库的 RPM 包，进行安装）：

```
sudo -s
LANG=C
yum -y install gcc gcc-c++ autoconf libjpeg libjpeg-devel libpng libpng-devel freetype
freetype-devel libxml2 libxml2-devel zlib 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 openssl openssl-devel openldap
openldap-devel nss_ldap openldap-clients openldap-servers
```

2. 【适用 RedHat 操作系统】

RedHat 等其他 Linux 发行版可从安装光盘中找到这些程序库的 RPM 包（事先可通过类似“rpm -qa | grep libjpeg”的命令查看 RPM 包是否存在，但通常“xxx-devel”不存在，须要安装）。RedHat 可以直接利用 CentOS 的 RPM 包安装，以下是 RPM 包的下载网址：

(1) RedHat AS4 & CentOS 4:

<http://mirrors.163.com/centos/4/os/i386/CentOS/RPMS/>

http://mirrors.163.com/centos/4/os/x86_64/CentOS/RPMS/

(2) RedHat AS5 & CentOS 5:

<http://mirrors.163.com/centos/5/os/i386/CentOS/>

http://mirrors.163.com/centos/5/os/x86_64/CentOS/

(3) RPM 包搜索网站:

<http://rpm.pbone.net/>

<http://www.rpmfind.net/>

(4) RedHat AS4 系统环境，通常情况下缺少包安装支持:

I. i386 系统如代码 4-1:

代码 4-1

```
wget http://blog.s135.com/soft/linux/nginx_php/rpm/i386/libjpeg-devel-6b-33.i386.rpm
rpm -ivh libjpeg-devel-6b-33.i386.rpm
wget http://blog.s135.com/soft/linux/nginx_php/rpm/i386/freetype-devel-2.1.9-1.i386.rpm
rpm -ivh freetype-devel-2.1.9-1.i386.rpm
```

```
wget http://blog.s135.com/soft/linux/nginx_php/rpm/i386/libpng-devel-1.2.7-1.i386.rpm
rpm -ivh libpng-devel-1.2.7-1.i386.rpm
```

II. x86_64 系统如代码 4-2:

代码 4-2

```
wget http://blog.s135.com/soft/linux/nginx_php/rpm/x86_64/libjpeg-devel-6b-33.x86_64.rpm
rpm -ivh libjpeg-devel-6b-33.x86_64.rpm
wget http://blog.s135.com/soft/linux/nginx_php/rpm/x86_64/freetype-devel-2.1.9-1.x86_64.rpm
rpm -ivh freetype-devel-2.1.9-1.x86_64.rpm
wget http://blog.s135.com/soft/linux/nginx_php/rpm/x86_64/libpng-devel-1.2.7-1.x86_64.rpm
rpm -ivh libpng-devel-1.2.7-1.x86_64.rpm
```

3. 【适用 CentOS、RedHat 及其他 Linux 操作系统】

下载程序源码包:

本文中提到的所有开源软件为截止到 2009 年 09 月 18 日的最新稳定版。

(1) 从软件的官方网站下载如代码 4-3:

代码 4-3

```
mkdir -p /data0/software
cd /data0/software
wget http://sysoev.ru/nginx/nginx-0.8.15.tar.gz
wget http://www.php.net/get/php-5.2.10.tar.gz/from/this/mirror
wget http://blog.s135.com/soft/linux/nginx_php/phpfpm/php-5.2.10-fpm-0.5.11.diff.gz
wget http://dev.mysql.com/get/Downloads/MySQL-5.1/mysql-5.1.38.tar.gz/from/
http://mysql.he.net/
wget http://ftp.gnu.org/pub/gnu/libiconv/libiconv-1.13.tar.gz
wget "http://downloads.sourceforge.net/mcrypt/libmcrypt-2.5.8.tar.gz? modtime=
1171868460&big_mirror=0"
wget "http://downloads.sourceforge.net/mcrypt/mcrypt-2.6.8.tar.gz?modtime=
1194463373&big_mirror=0"
wget http://pecl.php.net/get/memcache-2.2.5.tgz
wget "http://downloads.sourceforge.net/mhash/mhash-0.9.9.9.tar.gz?modtime=
1175740843&big_mirror=0"
wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-7.9.tar.gz
wget http://bart.eaccelerator.net/source/0.9.5.3/eaccelerator-0.9.5.3.tar.bz2
wget http://pecl.php.net/get/PDO_MYSQL-1.0.2.tgz
wget http://blog.s135.com/soft/linux/nginx_php/imagick/ImageMagick.tar.gz
wget http://pecl.php.net/get/imagick-2.2.2.tgz
```

(2) 从 blog.s135.com 下载 (比较稳定, 只允许在本站, 或者在 Linux/Unix 下通过 Wget、Curl 等命令下载以下软件), 如代码 4-4:

代码 4-4

```
mkdir -p /data0/software
cd /data0/software
wget http://blog.s135.com/soft/linux/nginx_php/nginx/nginx-0.8.15.tar.gz
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
wget http://blog.s135.com/soft/linux/nginx_php/mysql/mysql-5.1.38.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/libiconv/libiconv-1.13.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/mcrypt/libmcrypt- 2.5.8.tar.gz
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/memcache/memcache-2.2.5.tgz
wget http://blog.s135.com/soft/linux/nginx_php/mhash/mhash-0.9.9.9.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/pcre/pcre-7.9.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/eaccelerator/eaccelerator-
0.9.5.3.tar.bz2
wget http://blog.s135.com/soft/linux/nginx_php/pdo/PDO_MYSQL-1.0.2.tgz
wget http://blog.s135.com/soft/linux/nginx_php/imagick/ImageMagick.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/imagick/imagick-2.2.2.tgz
```

4.2 安装 PHP 5.2.10 (FastCGI 模式)

(1) 编译安装 PHP 5.2.10 所需的支持库，代码如 4-5 所示：

代码 4-5

```
tar zxvf libiconv-1.13.tar.gz
cd libiconv-1.13/
./configure --prefix=/usr/local
make
make install
cd ../

tar zxvf libmcrypt-2.5.8.tar.gz
cd libmcrypt-2.5.8/
./configure
make
make install
/sbin/ldconfig
cd libltdl/
./configure --enable-ltdl-install
make
make install
cd ../../

tar zxvf mhash-0.9.9.9.tar.gz
cd mhash-0.9.9.9/
./configure
make
```

```

make install
cd ../

ln -s /usr/local/lib/libmccrypt.la /usr/lib/libmccrypt.la
ln -s /usr/local/lib/libmccrypt.so /usr/lib/libmccrypt.so
ln -s /usr/local/lib/libmccrypt.so.4 /usr/lib/libmccrypt.so.4
ln -s /usr/local/lib/libmccrypt.so.4.4.8 /usr/lib/libmccrypt.so.4.4.8
ln -s /usr/local/lib/libmhash.a /usr/lib/libmhash.a
ln -s /usr/local/lib/libmhash.la /usr/lib/libmhash.la
ln -s /usr/local/lib/libmhash.so /usr/lib/libmhash.so
ln -s /usr/local/lib/libmhash.so.2 /usr/lib/libmhash.so.2
ln -s /usr/local/lib/libmhash.so.2.0.1 /usr/lib/libmhash.so.2.0.1

tar zxvf mccrypt-2.6.8.tar.gz
cd mccrypt-2.6.8/
./sbin/ldconfig
./configure
make
make install
cd ../

```

(2) 编译安装 MySQL 5.1.38 所需的代码如 4-6 所示:

代码 4-6

```

/usr/sbin/groupadd mysql
/usr/sbin/useradd -g mysql mysql
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
chmod +w /usr/local/webserver/mysql
chown -R mysql:mysql /usr/local/webserver/mysql
cd ../

```

如果你想要在这台服务器上运行 MySQL 数据库, 则执行以下几步。如果你只是希望让 PHP 支持 MySQL 扩展库, 能够连接其他服务器上的 MySQL 数据库, 那么, 以下几步无须执行。

1) 创建 MySQL 数据库存放目录:

```

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

```

2) 以 mysql 用户账号的身份建立数据表:

```

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

```

3) 创建 my.cnf 配置文件:

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

输入代码 4-7 的内容如下:

代码 4-7

```
[client]
default-character-set = utf8
port = 3306
socket = /tmp/mysql.sock

[mysql]
prompt="(\\u:blog.domain.com:)[\\d]> "
no-auto-rehash

[mysqld]
#default-character-set = utf8
user = mysql
port = 3306
socket = /tmp/mysql.sock
basedir = /usr/local/webserver/mysql
datadir = /data0/mysql/3306/data
open_files_limit    = 10240
back_log = 600
max_connections = 3000
max_connect_errors = 6000
table_cache = 614
external-locking = FALSE
max_allowed_packet = 32M
sort_buffer_size = 2M
join_buffer_size = 2M
thread_cache_size = 300
thread_concurrency = 8
query_cache_size = 32M
query_cache_limit = 2M
query_cache_min_res_unit = 2k
default-storage-engine = MyISAM
default_table_type = MyISAM
thread_stack = 192K
transaction_isolation = READ-COMMITTED
tmp_table_size = 246M
max_heap_table_size = 246M
long_query_time = 1
log_long_format
log-bin = /data0/mysql/3306/binlog
binlog_cache_size = 4M
binlog_format = MIXED
max_binlog_cache_size = 8M
max_binlog_size = 512M
expire_logs_days = 7
key_buffer_size = 256M
read_buffer_size = 1M
```

```

read_rnd_buffer_size = 16M
bulk_insert_buffer_size = 64M
myisam_sort_buffer_size = 128M
myisam_max_sort_file_size = 10G
myisam_max_extra_sort_file_size = 10G
myisam_repair_threads = 1
myisam_recover

skip-name-resolve
master-connect-retry = 10
slave-skip-errors = 1032,1062,126,1114,1146,1048,1396

server-id = 1

innodb_additional_mem_pool_size = 16M
innodb_buffer_pool_size = 2048M
innodb_data_file_path = ibdata1:1024M:autoextend
innodb_file_io_threads = 4
innodb_thread_concurrency = 8
innodb_flush_log_at_trx_commit = 2
innodb_log_buffer_size = 16M
innodb_log_file_size = 128M
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_lock_wait_timeout = 120
innodb_file_per_table = 0
[mysqldump]
quick
max_allowed_packet = 32M

```

4) 创建管理 MySQL 数据库的 shell 脚本:

```
vi /data0/mysql/3306/mysql
```

输入代码 4-8 的内容（这里的用户名 admin 和密码 12345678 接下来的步骤会创建）如下：

代码 4-8

```

#!/bin/sh

mysql_port=3306
mysql_username="admin"
mysql_password="12345678"

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 "Stopping 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"
fi

```

5) 赋予 shell 脚本可执行权限:

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

6) 启动 MySQL:

```
/data0/mysql/3306/mysql start
```

7) 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 root -p -S /tmp/mysql.sock
```

8) 输入以下 SQL 语句, 创建一个具有 root 权限的用户 (admin) 和密码 (12345678):

```
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' IDENTIFIED BY '12345678';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'127.0.0.1' IDENTIFIED BY '12345678';
```

9) (可选) 停止 MySQL:

```
/data0/mysql/3306/mysql stop
```

(3) 编译安装 PHP (FastCGI 模式), 代码如 4-9 所示:

代码 4-9

```
tar zxvf php-5.2.10.tar.gz
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/mysq_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-xmldrpc --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 ../
curl http://pear.php.net/go-pear | /usr/local/webserver/php/bin/php
```

(4) 编译安装 PHP5 扩展模块，如代码 4-10 所示：

代码 4-10

```
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 ../
```

(5) 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, 如代码 4-11 所示:

代码 4-11

```
extension = "memcache.so"
extension = "pdo_mysql.so"
extension = "imagick.so"
再查找 output_buffering = Off
修改为 output_buffering = On
```

自动修改: 若嫌手工修改麻烦, 可执行以下 shell 命令, 自动完成对 php.ini 文件的修改, 如代码 4-12 所示:

代码 4-12

```
sed -i 's#extension_dir = "./"#extension_dir =
"/usr/local/webserver/php/lib/php/extensions/no-debug-non-zts-20060613/"
\nextension = "memcache.so"\nnextension = "pdo_mysql.so"\nnextension =
"imagick.so"\n#' /usr/local/webserver/php/etc/php.ini
sed -i 's#output_buffering = Off#output_buffering = On#'
/usr/local/webserver/php/etc/php.ini
sed -i 's#; always_populate_raw_post_data = On#always_populate_raw_post_data = On#g'
/usr/local/webserver/php/etc/php.ini
```

(6) 配置 eAccelerator 加速 PHP:

```
mkdir -p /usr/local/webserver/eaccelerator_cache
vi /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, 如代码 4-13 所示:

代码 4-13

```
[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"
```

(7) 创建 www 用户和组, 以及供 blog.domain.com 和 www.domain.com 两个虚拟主机使用的目录, 如代码 4-14 所示:

代码 4-14

```
/usr/sbin/groupadd www
/usr/sbin/useradd -g www www
mkdir -p /data0/htdocs/blog
chmod +w /data0/htdocs/blog
chown -R www:www /data0/htdocs/blog
mkdir -p /data0/htdocs/www
chmod +w /data0/htdocs/www
chown -R www:www /data0/htdocs/www
```

(8) 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 配置而无须重启 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), 如代码 4-15 所示:

代码 4-15

```
<?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 corruptions 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-derived systems allow connections regardless of permissions.
            <value name="owner"></value>
            <value name="group"></value>
            <value name="mode">0666</value>
        </value>
    </section>

```

```

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 controlling 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>

```

(9) 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。

4.3 安装 Nginx 0.8.15

(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 ../
```

(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 ../
```

(3) 创建 Nginx 日志目录:

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

(4) 创建 Nginx 配置文件。

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

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

输入以下内容, 如代码 4-16 所示:

代码 4-16

```
user www www;
worker_processes 8;
error_log /data1/logs/nginx_error.log crit;
pid /usr/local/webserver/nginx/nginx.pid;
#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 65535;
events
{
    use epoll;
    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 blog.domain.com;
    index index.html index.htm index.php;
    root /data0/htdocs/blog;

    #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;
}

server
{
    listen      80;
    server_name www.domain.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 status.blog.domain.com;
    location / {
        stub_status on;
        access_log off;
    }
}
}

```

2) 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, 如代码 4-17 所示:

代码 4-17

```

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;

```

(5) 启动 Nginx:

```

ulimit -SHn 65535
/usr/local/webserver/nginx/sbin/nginx

```

4.4 配置开机自动启动 Nginx + PHP

用 vi 编辑器打开文件/etc/rc.local:

```
vi /etc/rc.local
```

在末尾增加以下内容:

```

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

```

4.5 优化 Linux 内核参数

用 vi 编辑器打开文件/etc/sysctl.conf:

```
vi /etc/sysctl.conf
```

在末尾增加以下内容, 如代码 4-18 所示:

代码 4-18

```

# Add
net.ipv4.tcp_max_syn_backlog = 65536
net.core.netdev_max_backlog = 32768
net.core.somaxconn = 32768

net.core.wmem_default = 8388608
net.core.rmem_default = 8388608
net.core.rmem_max = 16777216

```

```
net.core.wmem_max = 16777216

net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_syn_retries = 2

net.ipv4.tcp_tw_recycle = 1
#net.ipv4.tcp_tw_len = 1
net.ipv4.tcp_tw_reuse = 1

net.ipv4.tcp_mem = 94500000 915000000 927000000
net.ipv4.tcp_max_orphans = 3276800

#net.ipv4.tcp_fin_timeout = 30
#net.ipv4.tcp_keepalive_time = 120
net.ipv4.ip_local_port_range = 1024 65535
```

使配置立即生效:

```
/sbin/sysctl -p
```

4.6 在不停止 Nginx 服务的情况下平滑变更 Nginx 配置

(1) 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
```

(2) 这时, 输入以下命令查看 Nginx 主进程号:

```
ps -ef | grep "nginx: master process" | grep -v "grep" | awk -F ' ' '{print $2}'
```

屏幕显示的即为 Nginx 主进程号, 例如:

```
6302
```

这时, 执行以下命令即可使修改过的 Nginx 配置文件生效:

```
kill -HUP 6302
```

或者无须这么麻烦, 找到 Nginx 的 pid 文件:

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

4.7 编写每天定时切割 Nginx 日志的脚本

(1) 创建脚本/usr/local/webserver/nginx/sbin/cut_nginx_log.sh:

```
vi /usr/local/webserver/nginx/sbin/cut_nginx_log.sh
```

输入以下内容, 如代码 4-19 所示:

代码 4-19

```
#!/bin/bash
# This script run at 00:00

# The Nginx logs path
logs_path="/usr/local/webserver/nginx/logs/"

mkdir -p ${logs_path}${date -d "yesterday" +"%Y"}${date -d "yesterday" +"%m"}/
mv ${logs_path}access.log ${logs_path}${date -d "yesterday" +"%Y"}${date -d "yesterday" +"%m"}/access_${date -d "yesterday" +"%Y%m%d"}.log
kill -USR1 `cat /usr/local/webserver/nginx/nginx.pid`
```

(2) 设置 crontab, 每天凌晨 00:00 切割 nginx 访问日志:

```
crontab -e
```

输入以下内容:

```
00 00 * * * /bin/bash /usr/local/webserver/nginx/sbin/cut_nginx_log.sh
```