LNMP 安装与配置

作者:李强强
时间:2012.11.29

Ngnx 与 apache、lighttp 性能综合对比，如下图:

表 1-2　Nginx 与 Apache、Lighttpd 的综合对比

Web 服务器	Nginx	Apache	Lighttpd
反向代理	非常好	好	一般
Rewrite 规则	非常好	好	一般
FastCGI	好	差	非常好
热部署	支持	不支持	不支持
系统压力比较	很小	小	很大
稳定性	非常好	好	一般
安全性	一般	好	一般
技术资料	很少	非常多	一般
静态文件处理	非常好	一般	好
虚拟主机	支持	支持	支持
内存消耗	非常小	很大	非常小

通过表 1-2 可以看出，Nginx 在反向代理、Rewrite 规则、稳定性、静态文件处理、内存消耗等方面，表现出了很强的优势，选用 Nginx 取代传统的 Apache 服务器，将会获得多方面的性能提升。

Lnmp linux nginx mysql php
Linux
Nginx->>>
Mysql
php

三大领域:网络,服务器，开发

1.准备 php 函数的 rpm 包

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.准备 lnmp 其他的源代码包

wget http://blog.s135.com/soft/linux/nginx_php/nginx/nginx-0.8.46.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/php/php-5.2.14.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/phpfpm/php-5.2.14-fpm-0.5.14.diff.gz
wget http://blog.s135.com/soft/linux/nginx_php/mysql/mysql-5.5.3-m3.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/libiconv/libiconv-1.13.1.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-8.10.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/eaccelerator/eaccelerator-0.9.6.1.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-2.3.0.tgz

3.安装 php-5.2.14 源代码包所需要的函数支持包

```
tar zxvf libiconv-1.13.1.tar.gz
cd libiconv-1.13.1/
./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
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/libmcrypt.la /usr/lib/libmcrypt.la
ln -s /usr/local/lib/libmcrypt.so /usr/lib/libmcrypt.so
ln -s /usr/local/lib/libmcrypt.so.4 /usr/lib/libmcrypt.so.4
ln -s /usr/local/lib/libmcrypt.so.4.4.8 /usr/lib/libmcrypt.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
ln -s /usr/local/bin/libmcrypt-config /usr/bin/libmcrypt-config

tar zxvf mcrypt-2.6.8.tar.gz
cd mcrypt-2.6.8/
./configure
make
make install
cd ../
```

4. 编译安装 MySQL 5.5.3-m3

```
./configure --prefix=/usr/local/mysql --without-debug --enable-thread-safe-client --with-pthread
--enable-assembler --enable-profiling --with-mysqld-ldflags=-all-static
```

--with-client-ldflags=-all-static --with-extra-charsets=all --with-plugins=all
--with-mysqld-user=mysql --without-embedded-server --with-server-suffix=-community
--with-unix-socket-path=/tmp/mysql.sock
Make
#编译
Make install
#安装
Useradd mysql
#创建 mysql 用户
Setfacl -m u:mysql:rwx -R /usr/local/mysql
Setfacl -m d:u:mysql:rwx -R /usr/local/mysql
#设置权限
/usr/local/mysql/bin/mysql_install_db --user=mysql
#安装 mysql 和 test 数据库
/usr/local/mysql/bin/mysqld_safe --user=mysql &
#启动 mysql 服务
/usr/local/mysql/bin/mysqladmin -uroot password "123"
#修改 mysql 登录密码为 123
/usr/local/mysql/bin/mysql -uroot -p123
#用 mysql 登录
Cp /usr/local/mysql/share/mysql/my-medium.cnf /etc/my.cnf
#准备 mysql 配置文件
Vi /etc/my.cnf
Default-character-set=utf8
#修改客户端和连接字符集
Character-set-server=utf8
#修改服务器和数据库字符集
Collation-server=utf8
#修改服务器校验字符集
Pkill mysqld
/usr/local/mysql/bin/mysqld_safe --user=mysql &
#重启 mysql 服务让字符集生效

5. 编译安装 PHP（FastCGI 模式）

tar zxvf php-5.2.14.tar.gz
gzip -cd php-5.2.14-fpm-0.5.14.diff.gz | patch -d php-5.2.14 -p1
#解压并打补丁，让 php 支持 fpm 来方便管理 php-cgi 进程
cd php-5.2.14/
./configure --prefix=/usr/local/php --with-config-file-path=/usr/local/php/etc
--with-mysql=/usr/local/mysql --with-mysqli=/usr/local/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
make ZEND_EXTRA_LIBS='-liconv'
make install
cp php.ini-dist /usr/local/php/etc/php.ini
cd ../

6.准备编译安装 PHP5 扩展模块

tar zxvf memcache-2.2.5.tgz
cd memcache-2.2.5/
/usr/local/php/bin/phpize
./configure --with-php-config=/usr/local/php/bin/php-config
make
make install
cd ../

tar jxvf eaccelerator-0.9.6.1.tar.bz2
cd eaccelerator-0.9.6.1/
/usr/local/php/bin/phpize
./configure --enable-eaccelerator=shared --with-php-config=/usr/local/php/bin/php-config
make
make install
cd ../

tar zxvf PDO_MYSQL-1.0.2.tgz
cd PDO_MYSQL-1.0.2/
/usr/local/php/bin/phpize
./configure --with-php-config=/usr/local/php/bin/php-config --with-pdo-mysql=/usr/local/mysql
make
make install
cd ../

tar zxvf ImageMagick.tar.gz
cd ImageMagick-6.5.1-2/
./configure
make
make install
cd ../

tar zxvf imagick-2.3.0.tgz
cd imagick-2.3.0/
/usr/local/php/bin/phpize
./configure --with-php-config=/usr/local/php/bin/php-config
make
make install
cd ../

7. 修改 php.ini 文件,让 php 模块生效

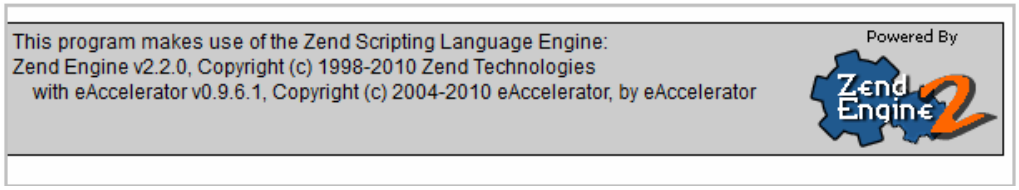
cp /lnmp/php-5.2.14/php.ini-dist /usr/local/php/etc/php.ini

vi php.ini
extension_dir = "/usr/local/php/lib/php/extensions/no-debug-non-zts-20060613/"
extension = "memcache.so"
extension = "pdo_mysql.so"
extension = "imagick.so"

再查找 output_buffering = Off
修改为 output_buffering = On

再查找; cgi.fix_pathinfo=0
修改为 cgi.fix_pathinfo=0，防止 Nginx 文件类型错误解析漏洞

8. 在 php.ini 中配置 eAccelerator 加速 PHP



mkdir -p /usr/local/eaccelerator_cache
#准备 eaccelerator 缓存目录

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

9.准备 php-cgi 和 nginx 进程执行者用户

Useradd nginx

10. 创建 php-fpm 配置文件- php-fpm.conf

vi /usr/local/php/etc/php-fpm.conf
<value name="display_errors">0</value>
#0 改成 1，页面上会输出错误日志
Unix user of processes
<value name="user">nginx</value>

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

<value name="max_children">128</value>

#最大子进程数 128,如果内存小于 2G，则 64 个最佳
<value name="rlimit_files">65535</value>
Set open file desc rlimit,同时打开的文件数,linux 系统允许同时打开的文件数为 1024,修改 linux 系统中允许同时打开的文件,ulimit -SHn 65535，而且这个参数重启后还能生效，加到 /etc/profile 全局配置文件的最后，开机就会生效,ulimit -a 查看 open files 65535
<value name="max_requests">1024</value>
#最大请求数, How much requests each process should execute before respawn.一个子进程能够回应 1042 个请求

11. 启动 php-cgi(fastcgi)进程，监听 127.0.0.1 的 9000 端口，进程数为 128（如果服务器内存小于 3GB，可以只开启 64 个进程），用户为 nginx：
/usr/local/php/sbin/php-fpm start
#启动 php-cgi
/usr/local/php/sbin/php-fpm reload
#重新加载配置文件
/usr/local/php/sbin/php-fpm stop
#关闭 php-fpm,此时 nginx 肯定不上 php

12. 安装 Nginx 所需的 pcre 库
tar zxvf pcre-8.10.tar.gz
cd pcre-8.10/
./configure
make && make install
cd ../

13. 安装 Nginx
tar zxvf nginx-0.8.46.tar.gz
cd nginx-0.8.46/
./configure --user=nginx --group=nginx --prefix=/usr/local/nginx --with-http_stub_status_module --with-http_ssl_module
make && make install
cd ../

14. 修改 Nginx 配置文件
vi /usr/local/nginx/conf/nginx.conf
user nginx nginx;

worker_processes 8;
#相当于 cpu 个数

error_log logs/nginx_error.log

pid   /usr/local/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 65;

  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.s135.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;
      fastcgi_param    SCRIPT_FILENAME    $document_root$fastcgi_script_name;
      include fastcgi_params;
    }

    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;
     }
}
```

15. 在/usr/local/nginx/conf/目录中创建 fastcgi_params 文件

```
fastcgi_param  SCRIPT_FILENAME    $document_root$fastcgi_script_name;
```

#建议把 fastcgi_param 写到 nginx.conf 中而不是把它写到 fastcgi_params 配置文件中，否则配置不够灵活，比如后面默认 php 设置和 alias php 设置中，他们的 php 页面的系统地址是不同的，比如:
默认 php 文件->/usr/local/nginx/html/index.php
Alias php 文件->/mnt/bbs/index.php
这个时候你会发现 fastcgi_params 中的 SCRIPT_FILENAME 的值是相同的，这样会导致 alias php 的页面出不来，而配置在 nginx.conf 中各自配置各自的 php 系统地址，这样比较灵活.

#如果你觉得每个连接 php 的配置中都要加这一句话有点冗余，那就把它加入到 fastcgi_params 文件中，这样只需要加一次，其他所有的 nginx.conf 中的有关连接 fastcgi 的一块就不用加 fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name 这一句话了.

16.配置开机启动 nginx,php-fpm,ulimit

1)nginx
Vi /etc/rc.local
/usr/local/nginx/sbin/nginx
2)php-fpm
Vi /etc/rc.local
/usr/local/php/sbin/php-fpm start
3)ulimit
Vi /etc/profile
Ulimit -SHn 65535

17.检查 nginx 配置文件语句错误

/usr/local/nginx/sbin/nginx -t

18.平滑重启 nginx 进程

1)Pkill -HUP nginx
2)kill -HUP `pgrep -uroot nginx`
3)/usr/local/nginx/sbin/nginx -s reload

19. 编写每天定时切割 Nginx 日志的脚本

1、创建脚本/usr/local/nginx/sbin/cut_nginx_log.sh
vi /usr/local/nginx/sbin/cut_nginx_log.sh

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

# The Nginx logs path
logs_path="/usr/local/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/nginx/nginx.pid`
```

2、设置 crontab，每天凌晨 00:00 切割 nginx 访问日志
crontab -e
00 00 * * * /bin/bash /usr/local/nginx/sbin/cut_nginx_log.sh

20.配置 nginx 虚拟主机

Vi /usr/local/nginx/conf/nginx.conf
==➔www.baidu.com 公司网站

```
server
 {
   listen       80;
   server_name  www.baidu.com;
   index index.html index.php index.htm;
   root  /web/baidu;
   location ~ .*\.(php|php5)?$
   {
    fastcgi_pass  127.0.0.1:9000;
    fastcgi_index index.php;
    fastcgi_param    SCRIPT_FILENAME    $document_root$fastcgi_script_name;
    include fastcgi_params;
   }

   location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
   {
    expires      30d;
   }
#设置图片在客户端缓存时间为 30days
   location ~ .*\.(js|css)?$
   {
    expires      1h;
   }
#设置 js 和 css 文件在客户端的缓存时间为 1hour
   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  logs/access.log  access;
#自定义日志区域
  }

==➔www.sina.com 公司网站

server
 {
   listen       80;
   server_name  www.sina.com;
   index index.html index.php index.htm;
   root  /web/baidu;
   location ~ .*\.(php|php5)?$
   {
     fastcgi_pass  127.0.0.1:9000;
     fastcgi_index index.php;
     fastcgi_param    SCRIPT_FILENAME    $document_root$fastcgi_script_name;
     include fastcgi_params;
   }

   location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
   {
     expires      30d;
   }
#设置图片在客户端缓存时间为 30days
   location ~ .*\.(js|css)?$
   {
     expires      1h;
   }
#设置 js 和 css 文件在客户端的缓存时间为 1hour
   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  logs/access.log  access;
#自定义日志区域
  }
```

最后在客户端测试虚拟主机 www.baidu.com 和 www.sina.com 两家公司网站

21.列表页显示

```
location / {
                autoindex on;                #打开列表页
                root      html;
```

```
        index    index.html index.php index.htm;
    }
```

22.虚拟目录设置

```
location /bbs{
                alias /mnt/bbs/;
        }
```

#这样配置 html 静态文件是可以出来的，但是 php 动态页面出不来，而且会浏览器的页面上会显示" No input file specified. "的报错，其实是 php 系统文件地址(SCRIPT_FILENAME)找不到，也就是说 fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name; 中的$document_root$fastcgi_script_name 不是真正的/mnt/bbs/index.php 的地址，这可怎么解决:

```
location /bbs {
        alias /mnt/bbs/;
        index bbs.php index.html index.php;
    }
    location ~ ^/bbs/ {
        root /mnt/;
        fastcgi_pass    127.0.0.1:9000;
        fastcgi_index   index.php;
        fastcgi_param   SCRIPT_FILENAME   $document_root$fastcgi_script_name;
        include         fastcgi_params;
        log_format   bbs   '$document_root$fastcgi_script_name ';
        access_log logs/bbs.access.log bbs;
    }
```

#后面两行是关于日志的，就是为了更好的观察由 nginx 提交给 fastcgi 的 php 的系统地址 SCRIPT_FILENAME，在这里我用$request_filename 来给 SCRIPT_FILENAME 赋值，在日志中的结果为/mnt/bbs/index.php，在这里我发现一个问题就是$request_filename 中的 root 设置为/mnt，否则$request_filename 的值为:/mnt/bbs/bbs/index.php.

由以上可以得到一个结论，就是默认 php 设置也可以这样设置关于 SCRIPT_FILENAME:

```
location ~ \.php$ {
        root          html;
        fastcgi_pass    127.0.0.1:9000;
        fastcgi_index   index.php;
        fastcgi_param   SCRIPT_FILENAME   $document_root$fastcgi_script_name;
        include         fastcgi_params;
        log_format   php   '$document_root$fastcgi_script_name ';
        access_log logs/php.access.log php;
    }
```

#此时从日志中可以看到，$request_filename 的值为/usr/local/nginx/html/index.php，而以前默认的/scripts$fastcgi_script_name 显然是错的 php 系统地址，日志中显示为/scripts/index.php

23.nginx 状态监控

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

#客户端访问网址:http://www.baidu.com/nginxstatus

24.rewrite 正则过滤

location ~ \.php$ {

proxy_pass http://127.0.0.1;

}

Rewrite 指令最后一项参数为 flag 标记，支持的 flag 标记如下:

Last 标示完成 rewrite 规则

Break 不再匹配后面的规则

Redirect 302 临时重定向

Permanent 301 永久重定向

Last 和 break 用来实现 uri 重写，浏览器地址栏的 url 地址不变，但在服务器访问的路径发生了变化，redirect 和 permanent 用来实现 url 跳转，浏览器地址栏会显示跳转后的 url 地址，使用 alias 指令时必须使用 last 标记，使用 proxy_pass 指令时要使用 break 标记，last 标记在本条 rewrite 规则执行完毕后，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，终止匹配，不再匹配后面的规则.

在匹配的过程中，nginx 将首先匹配字符串，然后再匹配正则表达式，匹配到第一个正则表达式后，会停止搜索，如果匹配到正则表达式，则使用正则表达式的搜索结果，如果没有匹配到正则表达式，则使用字符串的搜索结果.

可以使用前缀"^~"来禁止匹配到字符串后，再去检查正则表达式，匹配到 url 后，将停止查询.

使用前缀"="可以进行精确的 url 匹配，如果找到匹配的 uri，则停止查询，例如"location=/"，只能匹配到"/"，而"/test.html"则不能被匹配.

正则表达式的匹配，按照它们在配置文件中的顺序进行，写在前面的优先.

```
Location = / {
    #仅仅匹配 /
    [configuration A]
}
Location / {
    #匹配任何以/开头的查询，但是正则表达式及较长的字符串(/bbs/)将被优先匹配.
    [configuration B]
}
Location ^~ /images/ {
    #匹配任何以/images/开头的字符串，并且停止搜索，所以正则表达式将不会被检查.
    [configuration C]
}
Location ~* \.(gif|jpg|jpeg)$ {
    #匹配以.gif、.jpg、.jpeg 结尾的任何请求，但是，/images/内的请求将使用 configuration c 的配置
    [configuratoin D]
}
```

请求处理匹配结果示例:

/ -> configuration A;

/documents/document.html -> configuration B;

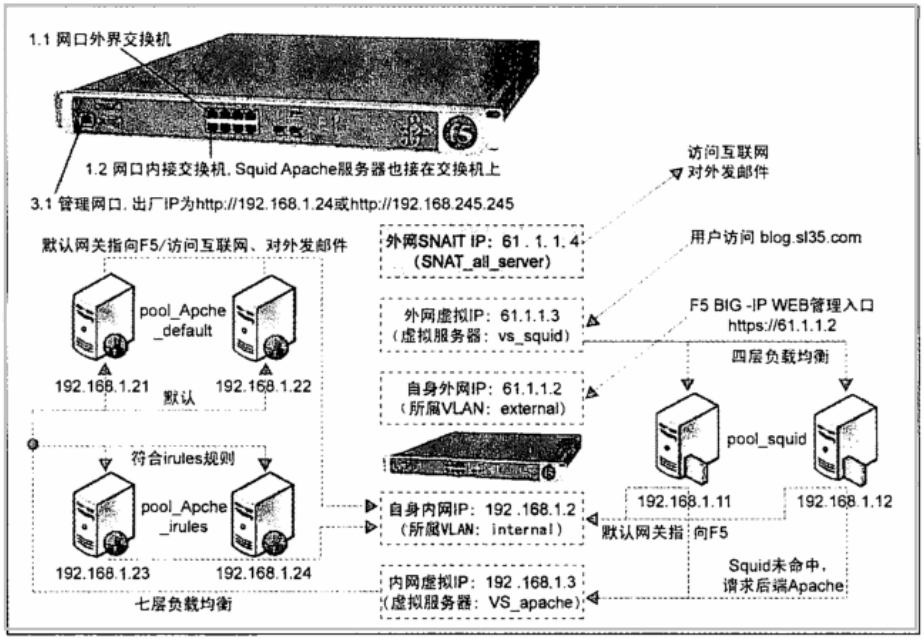
/images/1.gif -> configuration c;
/documents/1.jpg -> configuration D;

25.代理负载均衡技术

```
upstream myweb1 {
    server 192.168.1.2:80;
    server 192.168.1.3:80;
    server 192.168.1.4:80;
}
location / {
proxy_pass http://myweb1;
proxy_set_header Host $host;
proxy_set_header X-Forwarded-For $remote_addr;
}
```

#Host $host 这句话的意思是把 xp 访问的域名传递到后端的 nginx 服务器的虚拟主机服务，否则后端 nginx 无法获取 xp 在前端输入的域名

#其中$remote_addr 是 xp 机器的 ip,把它赋值给 X-Forwarded-For，这样在代理后端的 nginx 服务器中可以在日志中用$http_x_forwarded_for 来得到最前端的 xp 的 ip 地址，而不是否味的得到是前端的 nginx 服务器连接后端 nginx 服务器的 ip 地址.



26.模块设置

Error_log
#错误日志
Include
#包含子配置文件，0.6 版本以后子配置文件放在 nginx.conf 所在的路径下

Pid
#主进程 id 号
User
#nginx nginx 表明 nginx 进程的执行者和组
Worker_processes
#与 cpu 个数相同，4 核 cpu 为 4
Worker_rlimit_nofile 65535
#打开的文件描述符，不过提前得设置 ulimit -SHn 65535，即 linux 允许的打开文件个数
Worker_connectiones 65535
#客户端最大连接数 65535
Alias
#虚拟目录
Error_page
#404,500 错误跳转页面
Index
#index index.html，设置默认首页
Keepalive_timeout
#即 tcp 持续连接超时时间
Limit_rate
#limit _rate 100k，即限速为 100KB/s
Limit_rate_after
#limit_rate_after 1m，即下载文件超过 1m，则进入 limit_rate 限速阶段
Listen
#listen 192.168.100.1:80，即设置 ip 和端口
Location
#该指令允许对不同的 uri 进行不同的配置，可以是字符串、正则表达式
Resolver
#resolver 8.8.8.8，为 nginx 设置 dns 域名指向
Root
#设置网站根目录
Send_timeout
#超时时间是指进行了两次 tcp 握手，还没有转为 established 状态的时间，如果超过这个时间，客户没有响应，nginx 则关闭连接，可以用来防止 ddos 攻击
Sendfile
#启用或禁用 sendfile()函数，作用于拷贝两个文件描述符之间的操作函数，这个拷贝是在内核中操作的，比 read 和 write 拷贝高效得多
Server
#普通 web 配置或虚拟主机的配置的区域
Server_name
#根据客户端请求 header 头信息中的 host 域名，来匹配该请求应该由哪个虚拟主机配置或服务器的 ip
Tcp_nodelay
#封装 tcp/ip 数据包的等待时间，也叫纳格算法，在 keepalive 开启才有用
Tcp_nopush

#要求 sendfile 开启的时候才起作用，设置该选择的原因是 nginx 在 linux 上，试图在一个包中发送它的 httpd 应答头
Allow
#allow 192.168.100.254，只允许 192.168.100.254 访问
Deny
#deny all，拒绝其他任何人访问
Autoindex
#autoindex on，即开启列表页功能
Charset
#charset utf8;source_charset gbk，把服务器上的 gbk 网页编码转换成 utf8 输出给客户端
Fastcgi_pass
#fastcgi_pass 127.0.0.1:9000;
#fastcgi_index index.php;
#fastcgi_param SCRIPT_FILENAME /scripts$fastcgi_script_name;
#include fastcgi_params;
#fastcgi_pass 后跟的是 php-cgi 进程的 ip 和端口
Access_log
#正确日志
Proxy_pass
proxy_pass http://myweb1，即后跟的是 nginx 代理负载池 upstream 中的服务器
Proxy_set_header
proxy_set_header Host $host，设置把$host 带给后端的 nginx 服务器
Proxy_temp_path
#用户指定一个本地目录缓冲较大的代理请求，类似于 client_body_temp_path
Stub_status
stub_status on，即开户状态监控
Image_filter
#它指定适用于图片的转换类型