NGINX服务器部署

**nginx基本配置**

原贴：<http://breezey.blog.51cto.com/2400275/1339479>

访问量统计网站：

nginx安装

wget<http://nginx.org/download/nginx-1.3.7.tar.gz>

yum install -y zlib-devel

tar xf pcre-8.30.tar.gz

cd pcre-8.30

./configure && make && make install

useradd -M -s /sbin/nologin nginx

tar zxf nginx-1.0.8.tar.gz

cd nginx-1.0.8

./configure --prefix=/usr/local/nginx --user=nginx --group=nginx --with-http\_stub\_status\_module --with-http\_ssl\_module

make && make install

ln -s /usr/local/nginx/sbin/nginx /usr/local/sbin

nginx -t 检查配置文件

如果出错：[root@localhost conf]# /usr/local/nginx/sbin/nginx

/usr/local/nginx/sbin/nginx: error while loading shared libraries: libpcre.so.1: cannot open shared object file: No such file or directory

则 ln -s /lib64/libpcre.so.0.0.1 /lib64/libpcre.so.1

nginx 启动服务

killall -s HUP nginx 重载配置

killall -s QUIT nginx 退出进程

nginx支持的几种信号：

TERM,INT 快速关闭

QUIT 从容关闭

HUP 平滑重启，重载配置

USR1 重新打开日志文件，用于日志切割

USR2 平滑升级可执行程序

WINCH 从容关闭工作进程

基本配置

1.全局配置

user nginx nginx;运行用户和组

worker\_processes 1;工作进程数量，一般与cpu核数相等

error\_log logs/error.log;错误日志文件的位置

pid logs/nginx.pid;PID文件的位置

worker\_rlimit\_nofile 65535;

2.I/O事件配置

events {

use epoll;使用epoll模型

worker\_connections 4096;每进程处理4096个连接

}

3.HTTP配置

http {

include mime.types;

default\_typeapplication/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status ￥body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for" '; 定义日志格式

access\_log logs/access.log main;访问日志位置

sendfile on;用户空间与内核空间的缓冲区

keepalive\_timeout 65;连接保持超时

gzip on; 启用压缩

server {web服务的监听配置

listen 80 default\_server;监听地址及端口,默认解析地址

server\_name www.yanruogu.com;网站名称

charset utf-8;网页的默认字符集

location / {根目录配置

root html;网页根目录的位置

index index.html index.php;默认首页

}

error\_page 500 502 503 504 /50x.html;内部错误的反馈页面

location = /50x.html {错误页面配置

root html;

}

location ~ /status {访问位置为/status

stub\_status on;打开状态统计功能

access\_log off;关闭此位置的日志记录

}

}

}

注：打开nginx的访问状态统计必须启用--with-http\_stub\_status\_module模块

设定限速：

location /download {

limit\_rate 256k;

proxy\_pass http://1.2.3.4;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

location /movice {

limit\_rate\_after 10m;

limit\_rate 100k; #缓冲10M以后限速为100k/s

}

if ($http\_user\_agent ~ Google|Yahoo|baidu) {

limit\_rate 20k;

}

**nginx 虚拟目录的配置**

来源:http://sudone.com/nginx/nginx\_alias.html

nginx貌似没有虚拟目录的说法，因为它本来就是完完全全根据目录来设计并工作的。

如果非要给nginx安上一个虚拟目录的说法，那就只有alias标签比较“像”，干脆来说说alias标签和root标签的区别吧。

最基本的区别：alias指定的目录是准确的，root是指定目录的上级目录，并且该上级目录要含有location指定名称的同名目录。另外，根据前文所述，使用alias标签的目录块中不能使用rewrite的break。

说不明白，看下配置：

location /abc/ {

alias /home/html/abc/;

}

在这段配置下，http://test/abc/a.html就指定的是/home/html/abc/a.html。这段配置亦可改成

location /abc/ {

root /home/html/;

}

这样，nginx就会去找/home/html/目录下的abc目录了，得到的结果是相同的。

但是，如果我把alias的配置改成：

location /abc/ {

alias /home/html/def/;

}

那么nginx将会从/home/html/def/取数据，这段配置还不能直接使用root配置，如果非要配置，只有在/home/html/下建立一个 def->abc的软link（快捷方式）了。

一般情况下，在location /中配置root，在location /other中配置alias是一个好习惯。

至于alias和root的区别，我估计还没有说完全，如果在配置时发现奇异问题，不妨把这两者换换试试。

**nginx日志管理**

原贴：<http://breezey.blog.51cto.com/2400275/1339489>

log\_format指令用来设置日志的记录格式

语法：

log\_format name format [format.......]

具体格式：

log\_format main '$http\_x\_forwarded\_for-$remote\_user[$time\_local]' '"$request" $status $body\_byte\_sent' '"$http\_referer" "$http\_user\_agent"'

$http\_xforwarded\_for用于记录客户机真实ip地址

$remote\_user用于记录客户端用户名称

$time\_local用于记录访问时间和时区

$request用于记录请求的URL与HTTP协议

$status用于记录请求的状态，例如成功时状态为200，页面找不到时为404

$body\_bytes\_sent用于记录发送给客户端的文件主体内容大小

$http\_referer用于记录是从哪个页面链接访问过来的

$http\_user\_agent用于记录客户端浏览器的相关信息

定义日志：

access\_log /data/log/$server\_name.log main buffer=32k

buffer=32k：表示设置内存缓冲区的大小

开启日志缓存：

对于每一条日志记录，日志文件都将先打开文件，再写入日志记录，然后马上关闭，为了提高包含变量的日志文件存放路径的性能，可以使用open\_log\_file\_cache指令来设置，格式如下：

open\_log\_file\_cache max=N[inactive=time] [min\_users=N] [valid=time] |off

该指令默认是禁止的，等同于：

open\_log\_file\_cache off;

max: 设置缓存中的最大文件描述符数量

inactive:设置一个时间，如果在设置的时间内没有使用此文件描述符，则自动删除此描述符

min\_users:在参数inactive指定的时间范围内，如果日志文件超过被使用的次数，则将该日志文件的描述符计入缓存，默认为10秒钟

valid:设置多长时间检查一次，看日志文件路径与文件名是否仍然存在，默认60秒

e.g：

open\_log\_file\_cache max=1000 inactive=20s min\_users=2 valid=1m;

日志分割：

vim cut\_nginx\_log.sh

#!/bin/bash

# This script run at 00:00

# The Nginx logs path.

logs\_path="/data/logs"

mkdir -p ${logs\_path}/$(date -d "yesterday" +"%Y")/$(date -d "yesterday" +"%m")/

mv ${logs\_path}/weblogs.log ${logs\_path}/$(date -d "yesterday" +"%Y")/$(date -d "yesterday" +"%m")/weblogs\_$(date -d "yesterday" +"%Y%m%d").log

kill -USR1 `cat /usr/local/nginx/nginx.pid`

crontab -e

00 00 \* \* \* root /data/cut\_nginx\_log.sh

**nginx内核参数优化**

原贴：http://breezey.blog.51cto.com/2400275/1339488

vi /etc/sysctl.conf

net.ipv4.tcp\_max\_tw\_buckets = 6000

timewait的数量，默认是180000。

net.ipv4.ip\_local\_port\_range = 1024 65000

允许系统打开的端口范围。

net.ipv4.tcp\_tw\_recycle = 1

启用timewait快速回收。

net.ipv4.tcp\_tw\_reuse = 1

开启重用。允许将TIME-WAIT sockets重新用于新的TCP连接。

net.ipv4.tcp\_syncookies = 1

开启SYN Cookies，当出现SYN等待队列溢出时，启用cookies来处理。

net.core.somaxconn = 262144

web应用中listen函数的backlog默认会给我们内核参数的net.core.somaxconn限制到128，而Nginx内核参数定义的NGX\_LISTEN\_BACKLOG默认为511，所以有必要调整这个值。

net.core.netdev\_max\_backlog = 262144

每个网络接口接收数据包的速率比内核处理这些包的速率快时，允许送到队列的数据包的最大数目。

net.ipv4.tcp\_max\_orphans = 262144

系统中最多有多少个TCP套接字不被关联到任何一个用户文件句柄上。如果超过这个数字，孤儿连接将即刻被复位并打印出警告信息。这个限制仅仅是为了防止简单的DoS攻击，不能过分依靠它或者人为地减小这个值，更应该增加这个值(如果增加了内存之后)。

net.ipv4.tcp\_max\_syn\_backlog = 262144

记录的那些尚未收到客户端确认信息的连接请求的最大值。对于有128M内存的系统而言，缺省值是1024，小内存的系统则是128。

net.ipv4.tcp\_timestamps = 0

时间戳可以避免序列号的卷绕。一个1Gbps的链路肯定会遇到以前用过的序列号。时间戳能够让内核接受这种“异常”的数据包。这里需要将其关掉。

net.ipv4.tcp\_synack\_retries = 1

为了打开对端的连接，内核需要发送一个SYN并附带一个回应前面一个SYN的ACK。也就是所谓三次握手中的第二次握手。这个设置决定了内核放弃连接之前发送SYN+ACK包的数量。

net.ipv4.tcp\_syn\_retries = 1

在内核放弃建立连接之前发送SYN包的数量。

net.ipv4.tcp\_fin\_timeout = 1

如果套接字由本端要求关闭，这个参数决定了它保持在FIN-WAIT-2状态的时间。对端可以出错并永远不关闭连接，甚至意外当机。缺省值是60秒。2.2 内核的通常值是180秒，你可以按这个设置，但要记住的是，即使你的机器是一个轻载的WEB服务器，也有因为大量的死套接字而内存溢出的风险，FIN- WAIT-2的危险性比FIN-WAIT-1要小，因为它最多只能吃掉1.5K内存，但是它们的生存期长些。

net.ipv4.tcp\_keepalive\_time = 30

当keepalive起用的时候，TCP发送keepalive消息的频度，缺省是2小时。

转载自：<http://developer.51cto.com/art/201003/191603.htm>

**nginx负载均衡**

原贴：<http://breezey.blog.51cto.com/2400275/1339486>

nginx支持同时设置多组负载均衡，用来给不同的server来使用

nginx的负载均衡模块upstream目前支持4种算法：

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.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，可解决session共享的问题

3.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

4.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

nginx健康检查

重新编译安装第三方模块

cd nginx-1.3.7

git clone <https://github.com/yaoweibin/nginx_upstream_check_module>

patch -p1 nginx\_upstream\_check\_module/check\_1.2.6+.patch

./configure --add-modules=./nginx\_upstream\_check\_module

实例：

upstream myserver {

ip\_hash;

server 192.168.1.1:80 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.1.2:80 weight=1 max\_fails=2 fail\_timeout=30s ;

server 192.168.1.3:80 weight=1 max\_fails=2 fail\_timeout=30s down;

server 192.168.1.4:80 weight=1 max\_fails=2 fail\_timeout=30s backup;

check interval=3000 rise=2 fall=5 timeout=1000;

}

location / {

proxy\_pass http://myserver #将请求转发给upstream中定义好的服务器

proxy\_next\_upstream http\_502 http\_504 error timeout invalid\_header; #如果后端的服务器返回502、504、执行超时等错误，自动将请求转发到upstream负载均衡池中的另一台服务器

proxy\_set\_header Host $host;

proxy\_cache cache\_one; #使用proxy\_cache\_path定义的缓存

proxy\_set\_header X-Real-IP $remote\_addr; #检测客户端真实IP

proxy\_set\_header X-Forwarded-For $remote\_addr; #将客户端请求的IP返回给后端服务器

}

weight:

默认为1，weight越大，负载的权重就越大

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

fail\_timeout:

max\_fails次失败后，暂停的时间

down:

表示当前的server暂时不参与负载

backup：

当其他所有非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

interval：主动健康检查检测间隔时间，单位为毫秒

rsie：请求2次正常的话，标记此realserver的状态为up

fall：请求5次都失败的情况下，标记此realserver的状态为down

timeout：超时时间，单位为毫秒

#添加查看realserver状态页面

location /ups\_status {

check\_status;

access\_log off;

allow 211.157.169.10;

deny all;

}

**nginx的重写功能**

原贴：<http://breezey.blog.51cto.com/2400275/1339484>

rewrite 主要功能就是实现url的重写，nginx的rewrite规则采用PCRE(Perl Compatible Regular Expressions) Perl兼容正则表达式的语法进行规则匹配，如果需要nginx的rewrite功能，在编译nginx之前，需要编译安装PCRE库

例：

if ($http\_user\_agent ~ MSIE) {

rewrite ^(.\*)$ /msic/$1 break; #当用户代理为IE浏览器，则跳转至/msic/目录下

}

if (! -f $request\_filename) {

rewrite ^/img/(.\*)$ /site/$host/images/$1 last; #当用户请求的/img/下的某文件不存在，则跳转到/site/$host/images/目录下

}

rewrite语法规则：

变量名可以使用“=”或“！=”运算符

"~"符号表示区分大小写字母的匹配

"~\*"表示不区分大小写字母的匹配

"!~"和"!~\*"与"~" "~\*"相反

"-f"和"!-f"用来判断文件是否存在

"-d"和“!-d”判断目录是否存在

"-e" 和"!-e"判断文件或目录是否存在

"-x" "!-x"判断文件是否可执行

也支持$1~$9位置化参数

return指令：

location ~ .\*\.(sh|bash)?$ {

return 403;

}

#如果访问的url以"\*.sh" "\*.bash"结尾，则返回403

if ($host ~\* ^(.\*?)\.aaa\.com$) {

set $var\_tz '1';

}

if ($host ~\* ^192\.168\.1\.(.\*?)$) {

set $var\_tz '1';

}

if ($host ~\* ^localhost) {

set $var\_tz '1';

}

if ($var\_tz !~ '1') {

rewrite ^/(.\*)$ <http://www.aaa.com/> redirect;

}

#如果访问的url非\*.aaa.com和192.168.0.\* 及localhost ,则重定向到[http://www.aaa.com](http://www.aaa.com/)

flag标记：

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，浏览器地址栏会显示跳转后的url地址

permanent: 返回301永久重定向，浏览器地址栏会显示跳转后的url地址

last和break用来实现uri重写，浏览器地址栏url地址不变

redirect和permanent用来实现url重写，浏览吕地址栏会显示跳转后的url地址

一般在根location中（即location / {...}）或直接在server标签中编写rewrite规则，推荐使用last标记，在非根location中（location /cms/ {....}），则使用break标记

e.g.

location /cms/ {

proxy\_pass http://test.yourdomain.com;

rewrite "^/cms/(.\*)\.html$" /cms/index.html break;

}

实例：

location / {

rewrite ^/data/?$ /bbs/ last; 将/data目录重定向到/bbs/

}

防止盗链：

location ~\* \.(gif|jpg|png|swf|flv)$ {

valid\_referers none blocked [www.test.com](http://www.test.com/) \*.test.com;

if ($invalid\_referer) {

rewrite ^/(.\*) http://www.test.com/error.html;

}

}

域名跳转：

#将所有对[www.bac.com](http://www.bac.com/)的访问跳转到www.test.com

server {

listen 80;

server\_name www.test.com;

index index.html index.php

root /export /home/www

if ($host = "www.abc.com") {

rewrite ^/(.\*)$ http://www.test.com/$1 permanent;

}

}

**nginx反向代理及缓存**

原贴：<http://breezey.blog.51cto.com/2400275/1339477>

#允许客户端请求的最大的单个文件字节数

client\_max\_body\_size 300m;

#缓冲区代理缓冲用户端请求的最大字节数,可以理解为先保存到本地再传给用户

client\_body\_buffer\_size 128k;

#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\_connect\_timeout 600;

#连接成功后，等候后端服务器响应时间，其实已经进入后端的排除之中等候处理

proxy\_read\_timeout 600;

#后端服务器数据回传时间，就是在规定时间之内后端服务器必须传完所有的数据

proxy\_send\_timeout 600;

#代理请求缓存区，这个缓存区间会保存用户的头信息以供nginx进行规则处理，一般只要能保存下头信息即可

proxy\_buffer\_size 16k;

proxy\_busy\_buffer\_size 64k;

proxy\_temp\_path /data1/proxy\_temp\_path #proxy\_temp\_path和proxy\_cache\_path必须指定在同一个分区

#缓存路径、缓存目录的层级、缓存在内存中的大小，缓存时间，最大缓存大小（硬盘中）

proxy\_cache\_path /data1/proxy\_cache\_path levels=1:2 keys\_zone=cache\_one:500m inactive=1d max\_size=1g;

location /sms {

proxy\_pass [http://192.168.2.1](http://192.168.2.1/); #请求后端服务器

proxy\_cache cache\_one; #使用proxy\_cache\_path定义的缓存

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for; #将客户端请求的IP返回给后端服务器

}

location ~ .\*\.(gif|jpg|jpeg|png|bmp|swf|mp4|3gp|flv|js|css|html)$ {

proxy\_cache cache\_one;

proxy\_cache\_valid 200 10m;

proxy\_cache\_valid 304 1m;

proxy\_cache\_valid 301 302 1h;

proxy\_cache\_valid any 1m; #定义cache\_one缓存的内容，根据不同的状态码设置不同的缓存时间

proxy\_cache\_key $host$uri$is\_args$args; #以域名、URI、参数组合成web缓存的key值，nginx根据key值hash,存储缓存内容到二级缓存目录中

proxy\_pass [http://192.168.3.1](http://192.168.2.1/);

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for ;

}

浏览器本地缓存：

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

expires 30d;

}

location ~ .\*\.(js|css)?$ {

expires 1h;

}

nginx添加清除缓存模块:

cd nginx-1.3.7/

#下载ngx\_cache\_purge模块

wget http://labs.frickle.com/files/[ngx\_cache\_purge-2.1.tar.gz](http://labs.frickle.com/files/ngx_cache_purge-2.1.tar.gz)

tar xf ngx\_cach\_purge-2.1.tar.gz

#重新编译nginx配置参数，加上该模块

./configure --add-module=./ngx\_cache\_purge-2.1

在主配置文件中加上：

location ~ /purge(/.\*) {

#设置只允许指定的IP或IP段才可以清除URL缓存。

allow 127.0.0.1;

allow 192.168.0.0/16;

deny all;

proxy\_cache\_purge cache\_one $host$1$is\_args$args;

}

nginx清除缓存脚本：

|  |
| --- |
| #!/bin/sh  nginxcache\_path="/home/nginx/proxy\_cache\_dir"  grep -a -r $1 $nginxcache\_path | strings |grep "KEY: " | awk -F'KEY: ' '{print "http://"$2;}' | grep [a-zA-Z0-9]$ > cache\_list.txt  #清除cat cache\_list.txt文本中url所对应的nginx  sed -i "s#1img.looklook.cn/#1img.looklook.cn/purge/#" cache\_list.txt  for url in `cat cache\_list.txt`; do  curl $url  done |

**nginx的gzip压缩**

原贴：<http://breezey.blog.51cto.com/2400275/1339475>

gzip(GNU-ZIP)是一种压缩技术。经过gzip压缩后页面大小可以变为原来的30%甚至更小，这样，用户浏览页面的时候速度会块得多。gzip的压

缩页面需要浏览器和服务器双方都支持，实际上就是服务器端压缩，传到浏览器后浏览器解压并解析。浏览器那里不需要我们担心，因为目前的绝大多数浏览器都支持解析gzip过的页面。

Nginx的压缩输出有一组gzip压缩指令来实现。相关指令位于http{….}两个大括号之间。

**gzip**

决定是否开启gzip模块

param: on|off

example: gzip on;

**gzip\_static （需要安装上加上--with-http\_gzip\_static\_module模块）**

决定是否开启gzip静态压缩

param: on|off

example: gzip\_static on;

**gzip\_buffers**

设置gzip申请内存缓冲区的大小,其作用是按块大小的倍数申请内存空间，默认值是申请与原始数据大小相同的内存空间来存储gzip压缩结果

param: 2:int(k) 后面单位是k

example: gzip\_buffers 4 16k; #表示申请4个单位为16K的内存作为压缩结果流缓存

**gzip\_comp\_level**

设置gzip压缩等级，等级越底压缩速度越快文件压缩比越小，反之速度越慢文件压缩比越大

param:1-9

example:gzip\_com\_level 2;

**gzip\_min\_length**

当返回内容大于此值时才会使用gzip进行压缩,以K为单位,当值为0时，所有页面都进行压缩。默认为0，建议设置成1k,小于1k可能会越压越大 。

param: int

example: gzip\_min\_length 1k;

**gzip\_http\_version**

用于识别http协议的版本，早期的浏览器不支持gzip压缩，用户会看到乱码，所以为了支持前期版本加了此选项,目前此项基本可以忽略（默认1.1，前端为squid2.5及其之前版本时，使用1.0）

param: 1.0|1.1

example: gzip\_http\_version 1.0

**gzip\_proxied**

Nginx做为反向代理的时候启用

param:off|expired|no-cache|no-sotre|private|no\_last\_modified|no\_etag|auth|any

off – 关闭所有的代理结果数据压缩

expired – 启用压缩，如果header中包含”Expires”头信息

no-cache – 启用压缩，如果header中包含”Cache-Control:no-cache”头信息

no-store – 启用压缩，如果header中包含”Cache-Control:no-store”头信息

private – 启用压缩，如果header中包含”Cache-Control:private”头信息

no\_last\_modified – 启用压缩，如果header中包含”Last\_Modified”头信息

no\_etag – 启用压缩，如果header中包含“ETag”头信息

auth – 启用压缩，如果header中包含“Authorization”头信息

any – 无条件压缩所有结果数据

expample: gzip\_proxied no-cache;

**gzip\_types**

设置需要压缩的MIME类型,非设置值不进行压缩

param: text/html|application/x-javascript|text/css|application/xml

example: gzip\_types text/plain application/x-javascript text/css application/xml;

**gzip\_vary**

和http头有关系，加个vary头，给代理服务器用的，有的浏览器支持压缩，有的不支持，所以避免浪费不支持的也压缩，所以根据客户端的http头来判断，是否需要压缩

param: on|off

example: gzip\_vary on;

配置示例：

|  |
| --- |
| gzip on;  gzip\_static on;  gzip\_min\_length 1k;  gzip\_buffers 4 16k;  gzip\_http\_version 1.0;  gzip\_proxied any;  gzip\_disable “MSIE [1-6]\.”; #当客户机浏览器版本为ie6及以前版本时，禁用压缩  gzip\_comp\_level 3;  gzip\_types text/plain application/x-javascript text/css application/xml;  gzip\_vary on; |

**nginx平滑升级**

原贴：<http://breezey.blog.51cto.com/2400275/1339480>

将nginx从1.0.0平滑升级到1.3.7(以下方法也可用于nginx添加模块)

wget <http://www.nginx.org/download>/nginx-1.3.7.tar.gz

tar xf nginx-1.3.7.tar.gz

cd nginx-1.3.7

#按照原来的配置进行重配置，如果要加上新的模块，也可以一并加上

./configure --prefix=/usr/local/nginx --with-http\_mp4\_module --with-http\_sub\_module --with-http\_dav\_module --with-http\_stub\_status\_module --with-http\_addition\_module --with-http\_gzip\_static\_module

make #不要make install

#编译完，在objs目录下有个nginx可执行文件，将其cp到/usr/local/nginx/sbin/目录中

mv /usr/local/nginx/sbin/nginx /usr/local/nginx/sbin/nginx.old #备份原文件

cp ./objs/nginx /usr/local/nginx/sbin/ #cp新文件

#检测nginx配置文件

nginx -t

#执行以下命令，让nginx把nginx.pid改成nginx.pid.oldbin 跟着启动新的nginx

kill -USR2 `cat /usr/local/nginx/logs/nginx.pid`

#退出旧的nginx

kill -QUIT `cat /usr/local/nginx/logs/nginx.pid.oldbin`

升级完成

**利用TCMalloc优化nginx性能**

原贴：<http://breezey.blog.51cto.com/2400275/1339482>

TCMalloc的全称为Thread-Caching Malloc，是谷歌开发的开源工具google-perftools中的一个成员。与标准的glibc库的Malloc相比，TCMalloc库在内存分配效率和速度上要高很多，这在很大程度上提高了服务器在高并发情况下的性能，从而降低了系统的负载。下面简单介绍如何为Nginx添加TCMalloc库支持。

要安装TCMalloc库，需要安装libunwind（32位操作系统不需要安装）和google-perftools两个软件包，libunwind库为基于64位CPU和操作系统的程序提供了基本函数调用链和函数调用寄存器功能。

1.安装libunwind库

wget [http://ftp.twaren.net/Unix/NonGNU//libunwind/libunwind-1.1.tar.gz](http://ftp.twaren.net/Unix/NonGNU/libunwind/libunwind-1.1.tar.gz)

tar xf libunwind-1.1.tar.gz

cd libunwind-1.1

./configure && make && make install

2.安装google-perftools

tar zxvf google-perftools-1.9.tar.gz

cd google-perftools-1.9

./configure && make && make install

3.重新编译nginx

cd nginx-1.5.0/

./configure --with-google\_perftools\_module

4.为goole-perftools添加线程目录

mkdir /tmp/tcmalloc

chmod 777 /tmp/tcmalloc

5.修改nginx主配置文件，在全局配置中加上：

google\_perftools\_profile /tmp/tcmalloc;

启动nginx

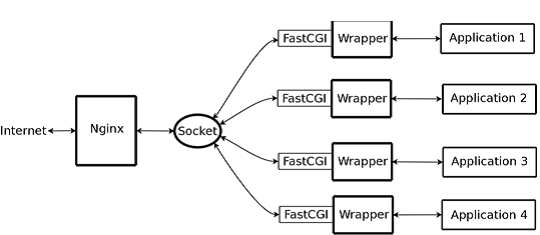
6.验证google-perftools正常加载

[173809619.png](http://img1.51cto.com/attachment/201312/173809619.png)

**Nginx支持php配置**

原贴：<http://kumu1988.blog.51cto.com/4075018/1083233>

Nginx本身是不支持对外部程序的直接调用或者解析，所有的外部程序（包括PHP）必须通过FastCGI接口来调用。FastCGI接口在Linux 下是socket，（这个socket可以是文件socket，也可以是ip socket）。为了调用CGI程序，还需要一个FastCGI的wrapper（wrapper可以理解为用于启动另一个程序的程序），这个 wrapper绑定在某个固定socket上，如端口或者文件socket。当Nginx将CGI请求发送给这个socket的时候，通过FastCGI 接口，wrapper接纳到请求，然后派生出一个新的线程，这个线程调用解释器或者外部程序处理脚本并读取返回数据；接着，wrapper再将返回的数据 通过FastCGI接口，沿着固定的socket传递给Nginx；最后，Nginx将返回的数据发送给客户端。

[](http://img1.51cto.com/attachment/201212/162718411.png)

**Nginx+FastCGI运行原理**

修改主配置文件，如下：

|  |
| --- |
| vim /usr/**local**/nginx/conf/nginx.conf |
| location / {  root /www;  **index** **index**.html **index**.htm **index**.php;  }  location ~ \.php$ {  root /www;  fastcgi\_pass 127.0.0.1:9000;  fastcgi\_index **index**.php;  fastcgi\_param SCRIPT\_FILENAME /**www**$fastcgi\_script\_name;  include fastcgi\_params;  }  #注：黑体www为web根目录，改成对应目录即可 |

location 定义文件类型, \.php$ 代表所有以 php 作为文件后缀的文件类型.  
root 定义 php 文件存放的路径, 当前以 "/www" 作为默认存放位置.  
fastcgi\_index 定义 php 文件类型中的默认索引页  
fastcgi\_param SCRIPT\_FILENAME 定义了页面请求参数, 如客户端需要访问 /t1.php 则会自动读取 /www/t1.php文件, 如客户端访问 / 则自动读取 /www/index.php 文件  
include 定义fastcgi 配置信息将会被保存到 /usr/local/nginx/conf/fastcgi\_params 文件中

nginx最常用的方法是利用 tcp/ip 协议连接 phpfastcgi 接口, 因此要连接php必须先启动fastcgi程序.

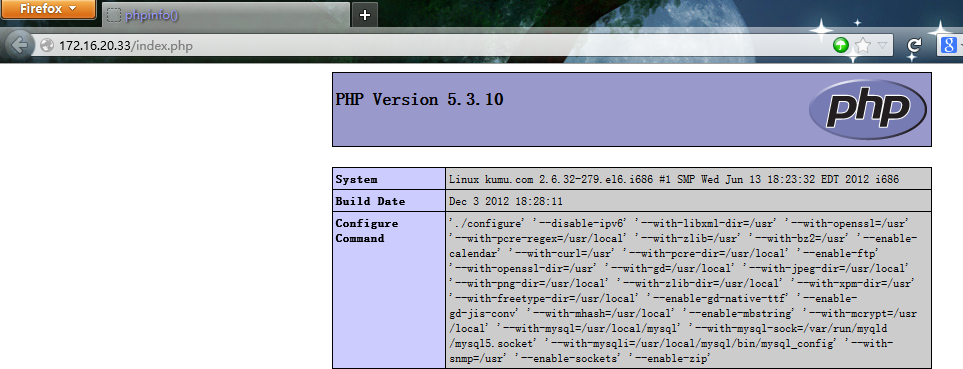
启动方法：

# **/usr/local/bin/php-cgi-b 127.0.0.1:9000 -c /usr/local/lib/php.ini &**

-b 用于指定监听地址与端口  
如果想指定配置文件位置，可以使用-c选项

注意, nginx 服务器负责所有的客户端请求, 包括了静态网页解析 (html), 页面中的其他对象如 ccs文件jpg png文件等传输, 但遇到客户需要请求 php 页面请求或 php 源码解析时, nginx将会把页面传递到fastcgi接口, 由 php-cgi 程序进行解析.

浏览器测试：[http://IP/index.php](http://ip/index.php)

[](http://img1.51cto.com/attachment/201212/162752289.png)

当前php-cgi只启用一个进程应答nginx所有连接中的php请求, 假如nginx具有一定的并发连接,那么php-cgi 将会变得无法同时应付大量的工作, 并且返回网页临时不可用信息. 将会导致所有php连接无法返回数据结果,并返回错误信息。

为有效地解决php-cgi接口无法应答大量并发连接请求, 我们可以利用 spawn-fcgi或者php-fpm,稍后送上。

参考：terry《性能调优和安全》一书

南非蚂蚁《高性能Linux服务器构建实战》Nginx章节

**nginx根据日志清除缓存脚本**

2013-12-11 17:30:57

标签：[nginx](http://blog.51cto.com/tag-nginx.html) [script](http://blog.51cto.com/tag-script.html) [purge](http://blog.51cto.com/tag-purge.html)

原创作品，允许转载，转载时请务必以超链接形式标明文章 [原始出处](http://breezey.blog.51cto.com/2400275/1339472) 、作者信息和本声明。否则将追究法律责任。<http://breezey.blog.51cto.com/2400275/1339472>

线上前端nginx服务用来做后端图片服务的缓存代理服务器，但后端图片服务器在对用户上传的图片进行处理以后，有时会出现图片无法显示等问题，当出现这种情况的时候，该图片，我们就不希望被nginx缓存，如果被缓存，则要被清除。

分别用两种不同的脚本方案来解决：

1.通过过滤日志中状态码正常（即200）及图片大小小于100字节的url路径去查找缓存路径，进而清除缓存，脚本如下：

|  |
| --- |
| #!/bin/bash  cache\_path=/home/nginx/proxy\_cache\_dir  logfile=/home/nginx/logs/access.log  datetime=`date +%d/%b/%Y`  logdir=/home/nginx/logs  #查找出所有当天nginx日志中状态码为200，大小小于100字节且被缓存命中的url，并将其保存至url.list  grep "$datetime" $logfile | awk '{if($9 == 200 && $10 < 100 && $NF ~ "HIT") print $7}' | sort -n | uniq > url.list  for i in `cat url.list`  do  #根据nginx日志中过滤的url查找缓存目录，对缓存中的url作处理后，保存到cache.list文件中  grep -a -r "$i" $cache\_path| strings |grep "KEY: " | awk -F'KEY: ' '{print "http://"$2;}' | grep [a-zA-Z0-9]$ > cache.list  #再次对url进行处理  sed -i "s#1img.looklook.cn/#1img.looklook.cn:7086/purge/#" cache.list  #清除缓存  for url in `cat cache\_list.txt`  do  curl "$url" | tee -a $logdir/purgelog/$(date +%y%m%d).log  done  done |

2.将nginx日志复制为一个副本，然后通过新的nginx日志跟副本的比较，对变化的一部分直接做awk处理

|  |
| --- |
| #!/bin/bash  F=/home/nginx/logs/access.log  O=/home/nginx/logs/old.log  if [ ! -f $O ]  then  cp $F $O  exit 3  fi  diff $F $O | \  awk '{if($10 ==200 && $11< 100 && $NF ~ "HIT") print "http://1img.looklook.cn:7086/purge"$8}' | sort | uniq | \  awk '{system("curl \""$1"\"|tee -a /home/nginx/logs/purgelog/$(date +%y%m%d).log")}'  cat $F > $O |

nginx附录

**nginx基础案例**

原贴：<http://firesk.blog.51cto.com/180373/242717>

标签：[nginx](http://blog.51cto.com/tag-nginx.html) [休闲](http://blog.51cto.com/tag-%E4%BC%91%E9%97%B2.html) [职场](http://blog.51cto.com/tag-%E8%81%8C%E5%9C%BA.html)

**Nginx 简介**

**Nginx** ("engine x") 是一个高性能的 HTTP 和 反向代理 服务器，也是一个 IMAP/POP3/SMTP 代理服务器。 Nginx 是由 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 将源代码以类 BSD 许可证的形式发布。尽管还是测试版，但是，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了。

根据最新一期（08 年 6 月份）的 NetCraft 调查报告显示，已经有超过两百万的主机使用了 Nginx，这个数字超过了另外一个轻量级的 HTTP 服务器 lighttpd， 排名第四，并且发展迅速。下面是这份报告的前几名的报表：

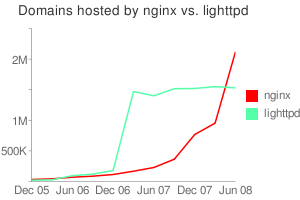
|  |  |
| --- | --- |
| **产品** | **网站数** |
| Apache | 84,309,103 |
| IIS | 60,987,087 |
| Google GFE | 10,465,178 |
| Unknown | 4,903,174 |
| nginx | 2,125,160 |
| Oversee | 1,953,848 |
| lighttpd | 1,532,952 |

关于这期调查报告的更详细信息请看下面链接：

<http://survey.netcraft.com/Reports/200806/>

下图是最近几个月使用 Nginx 和 lighttpd 的网站数比较

**图 1. 最近几个月使用 Nginx 和 lighttpd 的网站数比较**



**使用 Nginx 前必须了解的事项**

1. 目前官方 Nginx 并不支持 Windows，您只能在包括 Linux、UNIX、BSD 系统下安装和使用；
2. Nginx 本身只是一个 HTTP 和反向代理服务器，它无法像 Apache 一样通过安装各种模块来支持不同的页面脚本，例如 PHP、CGI 等；
3. Nginx 支持简单的负载均衡和容错；
4. 支持作为基本 HTTP 服务器的功能，例如日志、压缩、Byte ranges、Chunked responses、SSL、虚拟主机等等，应有尽有。

**在 Linux 下安装 Nginx**

为了确保能在 Nginx 中使用正则表达式进行更灵活的配置，安装之前需要确定系统是否安装有 PCRE（Perl Compatible Regular Expressions）包。您可以到 <ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/> 下载最新的 PCRE 源码包，使用下面命令下载编译和安装 PCRE 包：

|  |
| --- |
| # wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-7.7.tar.gz  # tar zxvf pcre-7.7.tar.gz  # cd pcre-7.7  # ./configure  # make  # make install |

接下来安装 Nginx，Nginx 一般有两个版本，分别是稳定版和开发版，您可以根据您的目的来选择这两个版本的其中一个，下面是把 Nginx 安装到 /opt/nginx 目录下的详细步骤：

|  |
| --- |
| # wget http://sysoev.ru/nginx/nginx-0.6.31.tar.gz  # tar zxvf nginx-0.6.31.tar.gz  # cd nginx-0.6.31  # ./configure --with-http\_stub\_status\_module –prefix=/opt/nginx  # make  # make install |

其中参数 --with-http\_stub\_status\_module 是为了启用 nginx 的 NginxStatus 功能，用来监控 Nginx 的当前状态。

安装成功后 /opt/nginx 目录下有四个子目录分别是：conf、html、logs、sbin 。其中 Nginx 的配置文件存放于 conf/nginx.conf，Nginx 只有一个程序文件位于 sbin 目录下的 nginx 文件。确保系统的 80 端口没被其他程序占用，运行 sbin/nginx 命令来启动 Nginx，打开浏览器访问此机器的 IP，如果浏览器出现 Welcome to nginx! 则表示 Nginx 已经安装并运行成功。

**常用的 Nginx 参数和控制**

**程序运行参数**

Nginx 安装后只有一个程序文件，本身并不提供各种管理程序，它是使用参数和系统信号机制对 Nginx 进程本身进行控制的。 Nginx 的参数包括有如下几个：

**-c** <path\_to\_config>：使用指定的配置文件而不是 conf 目录下的 nginx.conf 。

**-t**：测试配置文件是否正确，在运行时需要重新加载配置的时候，此命令非常重要，用来检测所修改的配置文件是否有语法错误。

**-v**：显示 nginx 版本号。

**-V**：显示 nginx 的版本号以及编译环境信息以及编译时的参数。

例如我们要测试某个配置文件是否书写正确，我们可以使用以下命令

|  |
| --- |
| sbin/nginx – t – c conf/nginx2.conf |

**通过信号对 Nginx 进行控制**

Nginx 支持下表中的信号：

|  |  |
| --- | --- |
| **信号名** | **作用描述** |
| TERM, INT | 快速关闭程序，中止当前正在处理的请求 |
| QUIT | 处理完当前请求后，关闭程序 |
| HUP | 重新加载配置，并开启新的工作进程，关闭就的进程，此操作不会中断请求 |
| USR1 | 重新打开日志文件，用于切换日志，例如每天生成一个新的日志文件 |
| USR2 | 平滑升级可执行程序 |
| WINCH | 从容关闭工作进程 |

有两种方式来通过这些信号去控制 Nginx，第一是通过 logs 目录下的 nginx.pid 查看当前运行的 Nginx 的进程 ID，通过 kill – XXX <pid> 来控制 Nginx，其中 XXX 就是上表中列出的信号名。如果您的系统中只有一个 Nginx 进程，那您也可以通过 killall 命令来完成，例如运行 killall – s HUP nginx 来让 Nginx 重新加载配置。

**配置 Nginx**

先来看一个实际的配置文件：

|  |
| --- |
| user nobody;# 工作进程的属主  worker\_processes 4;# 工作进程数，一般与 CPU 核数等同  #error\_log logs/error.log;  #error\_log logs/error.log notice;  #error\_log logs/error.log info;  #pid logs/nginx.pid;  events {  use epoll;#Linux 下性能最好的 event 模式  worker\_connections 2048;# 每个工作进程允许最大的同时连接数  }  http {  include mime.types;  default\_type application/octet-stream;  #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 off;  access\_log logs/access.log;# 日志文件名  sendfile on;  #tcp\_nopush on;  tcp\_nodelay on;  keepalive\_timeout 65;  include gzip.conf;    # 集群中的所有后台服务器的配置信息  upstream tomcats {  server 192.168.0.11:8080 weight=10;  server 192.168.0.11:8081 weight=10;  server 192.168.0.12:8080 weight=10;  server 192.168.0.12:8081 weight=10;  server 192.168.0.13:8080 weight=10;  server 192.168.0.13:8081 weight=10;  }  server {  listen 80;#HTTP 的端口  server\_name localhost;  charset utf-8;  #access\_log logs/host.access.log main;  location ~ ^/NginxStatus/ {  stub\_status on; #Nginx 状态监控配置  access\_log off;  }  location ~ ^/(WEB-INF)/ {  deny all;  }  location ~ \.(htm|html|asp|php|gif|jpg|jpeg|png|bmp|ico|rar|css|js|  zip|java|jar|txt|flv|swf|mid|doc|ppt|xls|pdf|txt|mp3|wma)$ {  root /opt/webapp;  expires 24h;  }  location / {  proxy\_pass http://tomcats;# 反向代理  include proxy.conf;  }  error\_page 404 /html/404.html;  # redirect server error pages to the static page /50x.html  #  error\_page 502 503 /html/502.html;  error\_page 500 504 /50x.html;  location = /50x.html {  root html;  }  }  } |

**Nginx 监控**

上面是一个实际网站的配置实例，其中灰色文字为配置说明。上述配置中，首先我们定义了一个 location ~ ^/NginxStatus/，这样通过 http://localhost/NginxStatus/ 就可以监控到 Nginx 的运行信息，显示的内容如下：

|  |
| --- |
| Active connections: 70  server accepts handled requests  14553819 14553819 19239266  Reading: 0 Writing: 3 Waiting: 67 |

NginxStatus 显示的内容意思如下：

* active connections – 当前 Nginx 正处理的活动连接数。
* server accepts handled requests -- 总共处理了 14553819 个连接 , 成功创建 14553819 次握手 ( 证明中间没有失败的 ), 总共处理了 19239266 个请求 ( 平均每次握手处理了 1.3 个数据请求 )。
* reading -- nginx 读取到客户端的 Header 信息数。
* writing -- nginx 返回给客户端的 Header 信息数。
* waiting -- 开启 keep-alive 的情况下，这个值等于 active - (reading + writing)，意思就是 Nginx 已经处理完正在等候下一次请求指令的驻留连接。

**静态文件处理**

通过正则表达式，我们可让 Nginx 识别出各种静态文件，例如 images 路径下的所有请求可以写为：

|  |
| --- |
| location ~ ^/images/ {  root /opt/webapp/images;  } |

而下面的配置则定义了几种文件类型的请求处理方式。

|  |
| --- |
| location ~ \.(htm|html|gif|jpg|jpeg|png|bmp|ico|css|js|txt)$ {  root /opt/webapp;  expires 24h;  } |

对于例如图片、静态 HTML 文件、js 脚本文件和 css 样式文件等，我们希望 Nginx 直接处理并返回给浏览器，这样可以大大的加快网页浏览时的速度。因此对于这类文件我们需要通过 root 指令来指定文件的存放路径，同时因为这类文件并不常修改，通过 expires 指令来控制其在浏览器的缓存，以减少不必要的请求。 expires 指令可以控制 HTTP 应答中的“ Expires ”和“ Cache-Control ”的头标（起到控制页面缓存的作用）。您可以使用例如以下的格式来书写 Expires：

|  |
| --- |
| expires 1 January, 1970, 00:00:01 GMT;  expires 60s;  expires 30m;  expires 24h;  expires 1d;  expires max;  expires off; |

**动态页面请求处理**

Nginx 本身并不支持现在流行的 JSP、ASP、PHP、PERL 等动态页面，但是它可以通过反向代理将请求发送到后端的服务器，例如 Tomcat、Apache、IIS 等来完成动态页面的请求处理。前面的配置示例中，我们首先定义了由 Nginx 直接处理的一些静态文件请求后，其他所有的请求通过 proxy\_pass 指令传送给后端的服务器（在上述例子中是 Tomcat）。最简单的 proxy\_pass 用法如下：

|  |
| --- |
| location / {  proxy\_pass http://localhost:8080;  proxy\_set\_header X-Real-IP $remote\_addr;  } |

这里我们没有使用到集群，而是将请求直接送到运行在 8080 端口的 Tomcat 服务上来完成类似 JSP 和 Servlet 的请求处理。

当页面的访问量非常大的时候，往往需要多个应用服务器来共同承担动态页面的执行操作，这时我们就需要使用集群的架构。 Nginx 通过 upstream 指令来定义一个服务器的集群，最前面那个完整的例子中我们定义了一个名为 tomcats 的集群，这个集群中包括了三台服务器共 6 个 Tomcat 服务。而 proxy\_pass 指令的写法变成了：

|  |
| --- |
| location / {  proxy\_pass http://tomcats;  proxy\_set\_header X-Real-IP $remote\_addr;  } |

在 Nginx 的集群配置中，Nginx 使用最简单的平均分配规则给集群中的每个节点分配请求。一旦某个节点失效时，或者重新起效时，Nginx 都会非常及时的处理状态的变化，以保证不会影响到用户的访问。

**总结**

尽管整个程序包只有五百多 K，但麻雀虽小、五脏俱全。 Nginx 官方提供的各种功能模块应有尽有，结合这些模块可以完整各种各样的配置要求，例如：压缩、防盗链、集群、FastCGI、流媒体服务器、Memcached 支持、URL 重写等等，更关键的是 Nginx 拥有 Apache 和其他 HTTP 服务器无法比拟的高性能。您甚至可以在不改变原有网站的架构上，通过在前端引入 Nginx 来提升网站的访问速度。

本文只是简单介绍了 Nginx 的安装以及常见的基本的配置和使用，更多关于 Nginx 的信息请阅读文章后面的参考资源。在这里要非常感谢我的朋友——陈磊（chanix@msn.com），他一直在做 Nginx 的中文 WIKI（<http://wiki.codemongers.com/NginxChs>），同时也是他介绍给我这么好的一款软件。

如果您的网站是运行在 Linux 下，如果您并没有使用一些非常复杂的而且确定 Nginx 无法完成的功能，那您应该试试 Nginx 。

**nginx常用技术**

作者:NetSeek [http://www.linuxtone.org](http://www.linuxtone.org/) (IT运维专家网|集群架构|性能调优)

原贴：<http://firesk.blog.51cto.com/180373/213625>

目 录

一、 Nginx 基础知识

二、 Nginx 安装及调试

三、 Nginx Rewrite

四、 Nginx Redirect

五、 Nginx 目录自动加斜线:

六、 Nginx Location

七、 Nginx expires

八、 Nginx 防盗链

九、 Nginx 访问控制

十、 Nginx日志处理

十一、 Nginx Cache

十二、 Nginx负载均衡

十三、 Nginx简单优化

十四、 如何构建高性能的LEMP环境

十五、 Nginx服务监控

十六、 常见问题与错误处理.

十七、 相关资源下载

【前言】：

编写此技术指南在于推广普及NGINX在国内的使用，更方便的帮助大家了解和掌握NGINX的一些使用技巧。本指南很多技巧来自于网络和工作中或网络上朋友们问我的问题.在此对网络上愿意分享的朋友们表示感谢和致意！欢迎大家和我一起丰富本技术指南提出更好的建议！请朋友们关注: [http://www.linuxtone.org](http://www.linuxtone.org/) 技术分享社区! 互想学习共同进步!

**一、 Nginx 基础知识**

1、简介

Nginx ("engine x") 是一个高性能的 HTTP 和 反向代理 服务器，也是一个 IMAP/POP3/SMTP 代理服务器。 Nginx 是由 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 将源代码以类BSD许可证的形式发布。尽管还是测试版，但是，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了。

更多的请见官方wiki: <http://wiki.codemongers.com/>

2、 Nginx的优点

nginx做为HTTP服务器，有以下几项基本特性：

1) 处理静态文件，索引文件以及自动索引；打开文件描述符缓冲．

2) 无缓存的反向代理加速，简单的负载均衡和容错．

3) FastCGI，简单的负载均衡和容错．

4) 模块化的结构。包括gzipping, byte ranges, chunked responses, 以及 SSI-filter等filter。如果由FastCGI或其它代理服务器处理单页中存在的多个SSI，则这项处理可以并行运行，而不需要相互等待。

5) 支持SSL 和 TLS SNI．

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来说基本上是毫无用处的。就稳定性而言, nginx比lighthttpd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，即使运行数个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。当采用select()/poll()调用时，还可以限制每个进程的连接数。

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。特别值得一提的是强大的Upstream与Filter链。 Upstream为诸如reverse proxy, 与其他服务器通信模块的编写奠定了很好的基础。而Filter链最酷的部分就是各个filter不必等待前一个filter执行完毕。它可以把前一个 filter的输出做为当前filter的输入，这有点像Unix的管线。这意味着，一个模块可以开始压缩从后端服务器发送过来的请求，且可以在模块接收 完后端服务器的整个请求之前把压缩流转向客户端。

Nginx采用了一些os提供的最新特性如对sendfile (Linux 2.2+)，accept-filter (FreeBSD 4.1+)，TCP\_DEFER\_ACCEPT (Linux 2.4+) 的支持，从而大大提高了性能

**二、 Nginx 安装及调试**

1、Pcre 安装

1. ./configure
2. make && make install
3. cd ../

复制代码

2. nginx 编译安装

1. ./configure --user=www --group=www --prefix=/usr/local/nginx/ --with-http\_stub\_status\_module --with-openssl=/usr/local/openssl
2. make && make install

复制代码

更详细的模块定制与安装请参照官方wiki.

3、Nginx 配置文件测试：

1. # /usr/local/nginx/sbin/nginx -t //Debug 配置文件的关键命令需要重点撑握.
2. 2008/12/16 09:08:35 [info] 28412#0: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
3. 2008/12/16 09:08:35 [info] 28412#0: the configuration file /usr/local/nginx/conf/nginx.conf was tested successfully

复制代码

3、Nginx 启动：

1. # /usr/local/nginx/sbin/nginx

复制代码

4、Nginx 配置文件修改重新加载：

1. # kill -HUP `cat /usr/local/nginx/logs/nginx.pid

复制代码

`

**三、Nginx Rewrite**

1. Nginx Rewrite 基本标记(flags)

last - 基本上都用这个Flag。

※相当于Apache里的[L]标记，表示完成rewrite，不再匹配后面的规则

break - 中止Rewirte，不再继续匹配

redirect - 返回临时重定向的HTTP状态302

permanent - 返回永久重定向的HTTP状态301

**※原有的url支持正则 重写的url不支持正则**

2. 正则表达式匹配，其中：

\* ~ 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

3. 文件及目录匹配，其中：

\* -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

3. Nginx 的一些可用的全局变量，可用做条件判断：

1. $args
2. $content\_length
3. $content\_type
4. $document\_root
5. $document\_uri
6. $host
7. $http\_user\_agent
8. $http\_cookie
9. $limit\_rate
10. $request\_body\_file
11. $request\_method
12. $remote\_addr
13. $remote\_port
14. $remote\_user
15. $request\_filename
16. $request\_uri
17. $query\_string
18. $scheme
19. $server\_protocol
20. $server\_addr
21. $server\_name
22. $server\_port
23. $uri

复制代码

**四、 Nginx Redirect**

将所有linuxtone.org与netseek.linuxtone.org域名全部自跳转到[http://www.linuxtone.org](http://www.linuxtone.org/)

1. server
2. {
3. listen 80;
4. server\_name linuxtone.org netseek.linuxtone.org;
5. index index.html index.php;
6. root /data/www/wwwroot;
7. if ($host !~ "^www.linxtone.org$") {
8. rewrite ^(.\*) http://www.linuxtone.org$1 redirect;
9. }
10. ........................
11. }

复制代码

**五、 Nginx 目录自动加斜线:**

1. if (-d $request\_filename){
2. rewrite ^/(.\*)([^/])$ http://$host/$1$2/ permanent;
3. }

复制代码

**六 Nginx Location**

1．基本语法:[和上面rewrite正则匹配语法基本一致]

**location [=|~|~\*|^~] /uri/ { … }**

\* ~ 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

示例1:

location = / {

# matches the query / only.

# 只匹配 / 查询。

}

**匹配任何查询，因为所有请求都已 / 开头。但是正则表达式规则和长的块规则将被优先和查询匹配**

示例2:

location ^~ /images/ {

# matches any query beginning with /images/ and halts searching,

# so regular expressions will not be checked.

# 匹配任何已 /images/ 开头的任何查询并且停止搜索。任何正则表达式将不会被测试。

示例3:

location ~\* .(gif|jpg|jpeg)$ {

# matches any request ending in gif, jpg, or jpeg. However, all

# requests to the /images/ directory will be handled by

}

**# 匹配任何已 gif、jpg 或 jpeg 结尾的请求。**

**七、 Nginx expires**

1．根据文件类型expires

1. # Add expires header for static content
2. location ~\* .(js|css|jpg|jpeg|gif|png|swf)$ {
3. if (-f $request\_filename) {
4. root /data/www/wwwroot/bbs;
5. expires 1d;
6. break;
7. }
8. }

复制代码

2、根据判断某个目录

1. # serve static files
2. location ~ ^/(images|javascript|js|css|flash|media|static)/ {
3. root /data/www/wwwroot/down;
4. expires 30d;
5. }

复制代码

**八、 Nginx 防盗链**

1. 针对不同的文件类型

1. #Preventing hot linking of images and other file types
2. location ~\* ^.+.(gif|jpg|png|swf|flv|rar|zip)$ {
3. valid\_referers none blocked server\_names \*.linuxtone.org linuxtone.org http://localhost baidu.com;
4. if ($invalid\_referer) {
5. rewrite ^/ ;
6. # return 403;
7. }
8. }

复制代码

2. 针对不同的目录

1. location /img/ {
2. root /data/www/wwwroot/bbs/img/;
3. valid\_referers none blocked server\_names \*.linuxtone.org http://localhost baidu.com;
4. if ($invalid\_referer) {
5. rewrite ^/ ;
6. #return 403;
7. }
8. }

复制代码

3. 同实现防盗链和expires的方法

1. #Preventing hot linking of images and other file types
2. location ~\* ^.+.(gif|jpg|png|swf|flv|rar|zip)$ {
3. valid\_referers none blocked server\_names \*.linuxtone.org linuxtone.org http://localhost ;
4. if ($invalid\_referer) {
5. rewrite ^/ ;
6. }
7. access\_log off;
8. root /data/www/wwwroot/bbs;
9. expires 1d;
10. break;
11. }

复制代码

**九、 Nginx 访问控制**

1. Nginx 身份证验证

1. #cd /usr/local/nginx/conf
2. #mkdir htpasswd
3. /usr/local/apache2/bin/htpasswd -c /usr/local/nginx/conf/htpasswd/tongji linuxtone
4. #添加用户名为linuxtone
5. New password: (此处输入你的密码)
6. Re-type new password: (再次输入你的密码)
7. Adding password for user
8. http://count.linuxtone.org/tongji/data/index.html(目录存在/data/www/wwwroot/tongji/data/目录下)
9. 将下段配置放到虚拟主机目录，当访问http://count.linuxtone/tongji/即提示要密验证:
10. location ~ ^/(tongji)/ {
11. root /data/www/wwwroot/count;
12. auth\_basic "LT-COUNT-TongJi";
13. auth\_basic\_user\_file /usr/local/nginx/conf/htpasswd/tongji;
14. }

复制代码

2. Nginx 禁止访问某类型的文件.

如，Nginx下禁止访问\*.txt文件，配置方法如下.

1. location ~\* .(txt|doc)$ {
2. if (-f $request\_filename) {
3. root /data/www/wwwroot/linuxtone/test;
4. #rewrite …..可以重定向到某个URL
5. break;
6. }
7. }

复制代码

方法2:

1. location ~\* .(txt|doc)${
2. root /data/www/wwwroot/linuxtone/test;
3. deny all;
4. }

复制代码

实例:

禁止访问某个目录

1. location ~ ^/(WEB-INF)/ {
2. deny all;
3. }

复制代码

3. 使用ngx\_http\_access\_module限制ip访问

1. location / {
2. deny 192.168.1.1;
3. allow 192.168.1.0/24;
4. allow 10.1.1.0/16;
5. deny all;
6. }

复制代码

详细参见wiki: <http://wiki.codemongers.com/NginxHttpAccessModule#allow>

4. Nginx 下载限制并发和速率

1. limit\_zone linuxtone $binary\_remote\_addr 10m;
2. server
3. {
4. listen 80;
5. server\_name down.linuxotne.org;
6. index index.html index.htm index.php;
7. root /data/www/wwwroot/down;
8. #Zone limit
9. location / {
10. limit\_conn linuxtone 1;
11. limit\_rate 20k;
12. }
13. ..........
14. }

复制代码

只允许客房端一个线程,每个线程20k.

【注】limit\_zone linuxtone $binary\_remote\_addr 10m; 这个可以定义在主的

5. Nginx 实现Apache一样目录列表

1. location / {
2. autoindex on;
3. }

复制代码

6. 上文件大小限制

主配置文件里加入如下，具体大小根据你自己的业务做调整。

client\_max\_body\_size 10m;

十、 Nginx 日志处理

1.Nginx 日志切割

#contab -e

59 23 \* \* \* /usr/local/sbin/logcron.sh /dev/null 2>&1

[root@count ~]# cat /usr/local/sbin/logcron.sh

1. #!/bin/bash
2. log\_dir="/data/logs"
3. time=`date +%Y%m%d`
4. /bin/mv ${log\_dir}/access\_linuxtone.org.log ${log\_dir}/access\_count.linuxtone.org.$time.log
5. kill -USR1 `cat /var/run/nginx.pid`

复制代码

更多的日志分析与处理就关注(同时欢迎你参加讨论):<http://bbs.linuxtone.org/forum-8-1.html>

2.利用AWSTATS分析NGINX日志

设置好Nginx日志格式,仍后利用awstats进行分析.

请参考: <http://bbs.linuxtone.org/thread-56-1-1.html>

3. Nginx 如何不记录部分日志

日志太多，每天好几个G，少记录一些，下面的配置写到server{}段中就可以了

location ~ .\*.(js|jpg|JPG|jpeg|JPEG|css|bmp|gif|GIF)$

{

access\_log off;

}

**十一、Nginx Cache服务配置**

如果需要将文件缓存到本地，则需要增加如下几个子参数：

1. proxy\_store on;
2. proxy\_store\_access user:rw group:rw all:rw;
3. proxy\_temp\_path 缓存目录;

复制代码

其中，

proxy\_store on用来启用缓存到本地的功能，

proxy\_temp\_path用来指定缓存在哪个目录下，如：proxy\_temp\_path html;

在经过上一步配置之后，虽然文件被缓存到了本地磁盘上，但每次请求仍会向远端拉取文件，为了避免去远端拉取文件，必须修改

1. proxy\_pass：
2. if ( !-e $request\_filename) {
3. proxy\_pass http://mysvr;
4. }

复制代码

即改成有条件地去执行proxy\_pass，这个条件就是当请求的文件在本地的proxy\_temp\_path指定的目录下不存在时，再向后端拉取。

更多更高级的应用可以研究ncache,详细请参照[http://bbs.linuxtone.org](http://bbs.linuxtone.org/) 里ncache相关的贴子.

**十二、Nginx 负载均衡**

1. Nginx 负载均衡基础知识

nginx的upstream目前支持4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。

2)、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。

3)、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。

4)、url\_hash（第三方）

2. Nginx 负载均衡实例1

1. upstream bbs.linuxtone.org {#定义负载均衡设备的Ip及设备状态
2. server 127.0.0.1:9090 down;
3. server 127.0.0.1:8080 weight=2;
4. server 127.0.0.1:6060;
5. server 127.0.0.1:7070 backup;
6. }

复制代码

在需要使用负载均衡的server中增加

proxy\_pass <http://bbs.linuxtone.org/>;

每个设备的状态设置为:

a) down 表示单前的server暂时不参与负载

b) weight 默认为1.weight越大，负载的权重就越大。

c) 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

d) fail\_timeout:max\_fails次失败后，暂停的时间。

e) 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，用来给不用的server来使用。

client\_body\_in\_file\_only 设置为On 可以讲client post过来的数据记录到文件中用来做debug

client\_body\_temp\_path 设置记录文件的目录 可以设置最多3层目录

location 对URL进行匹配.可以进行重定向或者进行新的代理 负载均衡

3. Nginx 负载均衡实例 2

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,也可以用作提高Squid缓存命中率.

简单的负载均等实例:

#vi nginx.conf //nginx主配置文件核心配置

1. ……….
2. #loadblance my.linuxtone.org
3. upstream my.linuxtone.org {
4. ip\_hash;
5. server 127.0.0.1:8080;
6. server 192.168.169.136:8080;
7. server 219.101.75.138:8080;
8. server 192.168.169.117;
9. server 192.168.169.118;
10. server 192.168.169.119;
11. }
12. …………..
13. include vhosts/linuxtone\_lb.conf;
14. ………
15. # vi proxy.conf
16. proxy\_redirect off;
17. proxy\_set\_header Host $host;
18. proxy\_set\_header X-Real-IP $remote\_addr;
19. proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;
20. client\_max\_body\_size 50m;
21. client\_body\_buffer\_size 256k;
22. proxy\_connect\_timeout 30;
23. proxy\_send\_timeout 30;
24. proxy\_read\_timeout 60;
25. proxy\_buffer\_size 4k;
26. proxy\_buffers 4 32k;
27. proxy\_busy\_buffers\_size 64k;
28. proxy\_temp\_file\_write\_size 64k;
29. proxy\_next\_upstream error timeout invalid\_header http\_500 http\_503 http\_404;
30. proxy\_max\_temp\_file\_size 128m;
31. proxy\_store on;
32. proxy\_store\_access user:rw group:rw all:r;
33. #nginx cache
34. #client\_body\_temp\_path /data/nginx\_cache/client\_body 1 2;
35. proxy\_temp\_path /data/nginx\_cache/proxy\_temp 1 2;

复制代码

#vi linuxtone\_lb.conf

1. server
2. {
3. listen 80;
4. server\_name my.linuxtone.org;
5. index index.php;
6. root /data/www/wwwroot/mylinuxtone;
7. if (-f $request\_filename) {
8. break;
9. }
10. if (-f $request\_filename/index.php) {
11. rewrite (.\*) $1/index.php break;
12. }
13. error\_page 403 http://my.linuxtone.org/member.php?m=user&a=login;
14. location / {
15. if ( !-e $request\_filename) {
16. proxy\_pass http://my.linuxtone.org;
17. break;
18. }
19. include /usr/local/nginx/conf/proxy.conf;
20. }
21. }

复制代码

**十三、Nginx简单优化**

1. 减小nginx编译后的文件大小 (Reduce file size of nginx)

默认的nginx编译选项里居然是用debug模式(-g)的（debug模式会插入很多跟踪和ASSERT之类），编译以后一个nginx有好几兆。去掉nginx的debug模式编译，编译以后只有几百K

在 auto/cc/gcc，最后几行有：

# debug

1. CFLAGS=”$CFLAGS -g”

复制代码

注释掉或删掉这几行，重新编译即可。

2. 修改Nginx的header伪装服务器

1) 修改nginx.h

1. #vi nginx-0.7.30/src/core/nginx.h
2. #define NGINX\_VERSION "1.8"
3. #define NGINX\_VER "LTWS/" NGINX\_VERSION
4. #define NGINX\_VAR "NGINX"
5. #define NGX\_OLDPID\_EXT ".oldbin"

复制代码

2) 修改nginx\_http\_header\_filter\_module

#vi nginx-0.7.30/src/http/ngx\_http\_header\_filter\_module.c

将如下

1. static char ngx\_http\_server\_string[] = "Server: nginx" CRLF;

复制代码

修改为

1. static char ngx\_http\_server\_string[] = "Server: LTWS" CRLF;

复制代码

a) 修改nginx\_http\_header\_filter\_module

#vi nginx-0.7.30/src/http/ngx\_http\_special\_response.c

将如下：

1. static u\_char ngx\_http\_error\_full\_tail[] =
2. "<hr><center>" NGINX\_VER "</center>" CRLF
3. "</body>" CRLF
4. "</html>" CRLF
5. ;

复制代码

1. static u\_char ngx\_http\_error\_tail[] =
2. "<hr><center>nginx</center>" CRLF
3. "</body>" CRLF
4. "</html>" CRLF
5. ;

复制代码

修改为:

1. static u\_char ngx\_http\_error\_full\_tail[] =
2. "<center> "NGINX\_VER" </center>" CRLF
3. "<hr><center>http://www.linuxtone.org</center>" CRLF
4. "</body>" CRLF
5. "</html>" CRLF
6. ;
7. static u\_char ngx\_http\_error\_tail[] =
8. "<hr><center>LTWS</center>" CRLF
9. "</body>" CRLF
10. "</html>" CRLF
11. ;

复制代码

修改后重新编译一下环境,

404错误的时候显示效果图（如果没有指定错误页的话）：

利用curl命令查看服务器header

**3.为特定的CPU指定CPU类型编译优化.**

默认nginx使用的GCC编译参数是-O

需要更加优化可以使用以下两个参数

--with-cc-opt='-O3'

--with-cpu-opt=opteron

使得编译针对特定CPU以及增加GCC的优化.

此方法仅对性能有所改善并不会有很大的性能提升，供朋友们参考.

CPUD类型确定: # cat /proc/cpuinfo | grep "model name"

编译优化参数参考：<http://en.gentoo-wiki.com/wiki/Safe_Cflags>

**4.Tcmalloc优化Nginx 性能**

1. # wget http://download.savannah.gnu.org/releases/libunwind/libunwind-0.99-alpha.tar.gz
2. # tar zxvf libunwind-0.99-alpha.tar.gz
3. # cd libunwind-0.99-alpha/
4. # CFLAGS=-fPIC ./configure
5. # make CFLAGS=-fPIC
6. # make CFLAGS=-fPIC install
7. # wget http://google-perftools.googlecode.com/files/google-perftools-0.98.tar.gz
8. # tar zxvf google-perftools-0.98.tar.gz
9. # cd google-perftools-0.98/
10. # ./configure
11. # make && make install
12. # echo "/usr/local/lib" > /etc/ld.so.conf.d/usr\_local\_lib.conf
13. # ldconfig
14. # lsof -n | grep tcmalloc

复制代码

编译nginx 加载google\_perftools\_module:

./configure --with-google\_perftools\_module

在主配置文件加入nginx.conf 添加:

google\_perftools\_profiles /path/to/profile;

5.内核参数优化

# vi /etc/sysctl.conf #在末尾增加以下内容：

1. net.ipv4.tcp\_fin\_timeout = 30
2. net.ipv4.tcp\_keepalive\_time = 300
3. net.ipv4.tcp\_syncookies = 1
4. net.ipv4.tcp\_tw\_reuse = 1
5. net.ipv4.tcp\_tw\_recycle = 1
6. net.ipv4.ip\_local\_port\_range = 5000 65000

复制代码

#使配置立即生效

/sbin/sysctl -p

**十四、如何构建高性的LEMP**

请参见: <http://www.linuxtone.org/lemp/lemp.pdf>

1、提供完整的配置脚本下载：<http://www.linuxtone.org/lemp/scripts.tar.gz>

2、提供NGINX常见配置范例含(虚拟主机，防盗链，Rewrite,访问控制，负载均衡

Discuz相关程序静态化及等等),你只要稍稍修改即可线上应用。 3、将原版的xcache替换成EA，并提供相关简单调优脚本及配置文件。

更多的及更新资料请关注: [http://www.linuxtone.org](http://www.linuxtone.org/)

**十五、Nginx监控**

1、 RRDTOOL+Perl脚本画图监控

先安装好rrdtool ，关于rrdtool本文不作介绍，具体安装请参照linuxtone监控版块.

#cd /usr/local/sbnin

#wget <http://blog.kovyrin.net/files/mrtg/rrd_nginx.pl.txt>

#mv rrd\_nginx.pl.txt rrd\_nginx.pl

#chmod a+x rrd\_nginx.pl

#vi rrd\_nginx.pl //配置脚本文件设置好路径

#!/usr/bin/perl

use RRDs;

use LWP::UserAgent;

# define location of rrdtool databases

my $rrd = '/data/www/wwwroot/nginx/rrd';

# define location of images

my $img = '/data/www/wwwroot/nginx/html';

# define your nginx stats URL

my $URL = "http://219.232.244.13/nginx\_status";

…………

【注】根据自己具体的状况修改相应的路径.

#crontab –e //加入如下

\* \* \* \* \* /usr/local/sbin/rrd\_nginx.pl

重启crond后，通过配置nginx虚拟主机指到/data/www/wwwroot/nginx/html目录，通过crond自动执行perl脚本会生成很多图片.

http://xxx/connections-day.png即可看到服务器状态图。

2、 官方Nginx-rrd 监控服务（多虚拟主机）（推荐）

网址：<http://www.nginx.eu/nginx-rrd.html>

此解决方案其实是基于上述监控方案的一个改进和增强，同样先安装好rrdtool这个画图工具和相应的perl模块再做如下操作：

# yum install perl-HTML\*

先建立好生成的库存和图片存放录

1. #mkdir -p /data/www/wwwroot/nginx/{rrd,html}
2. #cd /usr/local/sbin
3. #wget http://www.nginx.eu/nginx-rrd/nginx-rrd-0.1.4.tgz
4. #tar zxvf nginx-rrd-0.1.4.tgz
5. #cd nginx-rrd-0.1.4
6. #cd etc/
7. #cp nginx-rrd.conf /etc
8. #cd etc/cron.d
9. #cp nginx-rrd.cron /etc/cron.d
10. #cd /usr/local/src/nginx-rrd-0.1.4/html
11. # cp index.php /data/www/wwwroot/nginx/html/
12. #cd /usr/local/src/nginx-rrd-0.1.4/usr/sbin
13. #cp \* /usr/sbin/

复制代码

#vi /etc/nginx-rrd.conf

1. #####################################################
2. #
3. # dir where rrd databases are stored
4. RRD\_DIR="/data/www/wwwroot/nginx/rrd";
5. # dir where png images are presented
6. WWW\_DIR="/data/www/wwwroot/nginx/html";
7. # process nice level
8. NICE\_LEVEL="-19";
9. # bin dir
10. BIN\_DIR="/usr/sbin";
11. # servers to test
12. # server\_utl;server\_name
13. SERVERS\_URL="http://219.32.205.13/nginx\_status;219.32.205.13 http://www.linuxtone.org/nginx\_status;www.linuxtone.org""

复制代码

//根据你的具体情况做调整.

SEVERS\_URL 格式 http://domain1/nginx\_status;domain1 http://domain2/nginx\_status;domain2

这种格式监控多虚拟主机连接状态：

重点启crond服务，仍后通过<http://219.32.205.13/nginx/html/> 即可访问。配置过程很简单！

3、 CACTI模板监控Nginx

利用Nginx\_status状态来画图实现CACTI监控

nginx编译时允许http\_stub\_status\_module

# vi /usr/local/nginx/conf/nginx.conf

1. location /nginx\_status {
2. stub\_status on;
3. access\_log off;
4. allow 192.168.1.37;
5. deny all;
6. }

复制代码

1. # kill -HUP `cat /usr/local/nginx/logs/nginx.pid`
2. # wget http://forums.cacti.net/download.php?id=12676
3. # tar xvfz cacti-nginx.tar.gz
4. # cp cacti-nginx/get\_nginx\_socket\_status.pl /data/cacti/scripts/
5. # cp cacti-nginx/get\_nginx\_clients\_status.pl /data/cacti/scripts/
6. # chmod 755 /data/cacti/scripts/get\_nginx\*

复制代码

检测插件

1. # /data/cacti/scripts/get\_nginx\_clients\_status.pl http://192.168.1.37/nginx\_status

复制代码

在cacti管理面板导入

cacti\_graph\_template\_nginx\_clients\_stat.xml

cacti\_graph\_template\_nginx\_sockets\_stat.xml

**十六、常见问题与错误处理**

1、400 bad request错误的原因和解决办法

配置nginx.conf相关设置如下.

client\_header\_buffer\_size 16k;

large\_client\_header\_buffers 4 64k;

根据具体情况调整，一般适当调整值就可以。

2、Nginx 502 Bad Gateway错误

proxy\_next\_upstream error timeout invalid\_header http\_500 http\_503;

或者尝试设置:

large\_client\_header\_buffers 4 32k;

3、Nginx出现的413 Request Entity Too Large错误

这个错误一般在上传文件的时候会出现，

编辑Nginx主配置文件Nginx.conf，找到http{}段，添加

client\_max\_body\_size 10m; //设置多大根据自己的需求作调整.

如果运行php的话这个大小client\_max\_body\_size要和php.ini中的如下值的最大值一致或者稍大，这样就不会因为提交数据大小不一致出现的错误。

post\_max\_size = 10M

upload\_max\_filesize = 2M

4、解决504 Gateway Time-out(nginx)

遇到这个问题是在升级discuz论坛的时候遇到的

一般看来, 这种情况可能是由于nginx默认的fastcgi进程响应的缓冲区太小造成的, 这将导致fastcgi进程被挂起, 如果你的fastcgi服务对这个挂起处理的不好, 那么最后就极有可能导致504 Gateway Time-out

现在的网站, 尤其某些论坛有大量的回复和很多内容的, 一个页面甚至有几百K。

默认的fastcgi进程响应的缓冲区是8K, 我们可以设置大点

在nginx.conf里, 加入： fastcgi\_buffers 8 128k

这表示设置fastcgi缓冲区为8×128k

当然如果您在进行某一项即时的操作, 可能需要nginx的超时参数调大点，例如设置成60秒：send\_timeout 60;

只是调整了这两个参数, 结果就是没有再显示那个超时, 可以说效果不错, 但是也可能是由于其他的原因, 目前关于nginx的资料不是很多, 很多事情都需要长期的经验累计才有结果, 期待您的发现哈!

5、如何使用Nginx Proxy

朋友一台服务器运行tomcat 为8080端口,IP:192.168.1.2:8080,另一台机器IP:192.168.1.8. 朋友想通过访问[http://192.168.1.8](http://192.168.1.8/)即可访问tomcat服务.配置如下:

在192.168.1.8的nginx.conf上配置如下:

1. server {
2. listen 80;
3. server\_name java.linuxtone.org
4. location / {
5. proxy\_pass http://192.168.1.2:8080;
6. include /usr/local/nginx/conf/proxy.conf;
7. }
8. }

复制代码

6、如何关闭Nginx的LOG

access\_log /dev/null; error\_log /dev/null;

**十七、相关资源下载**

1.nginx配置示例及脚本下载:

# wget <http://www.linuxtone.org/lemp/scripts.tar.gz> #此脚本范例定期更新.