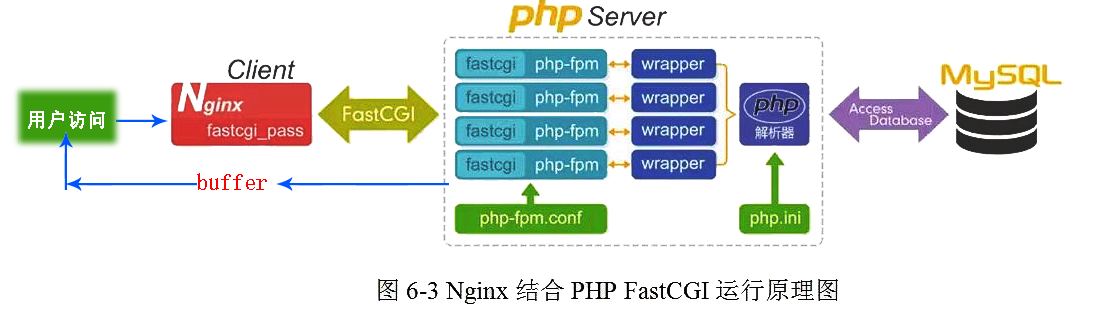
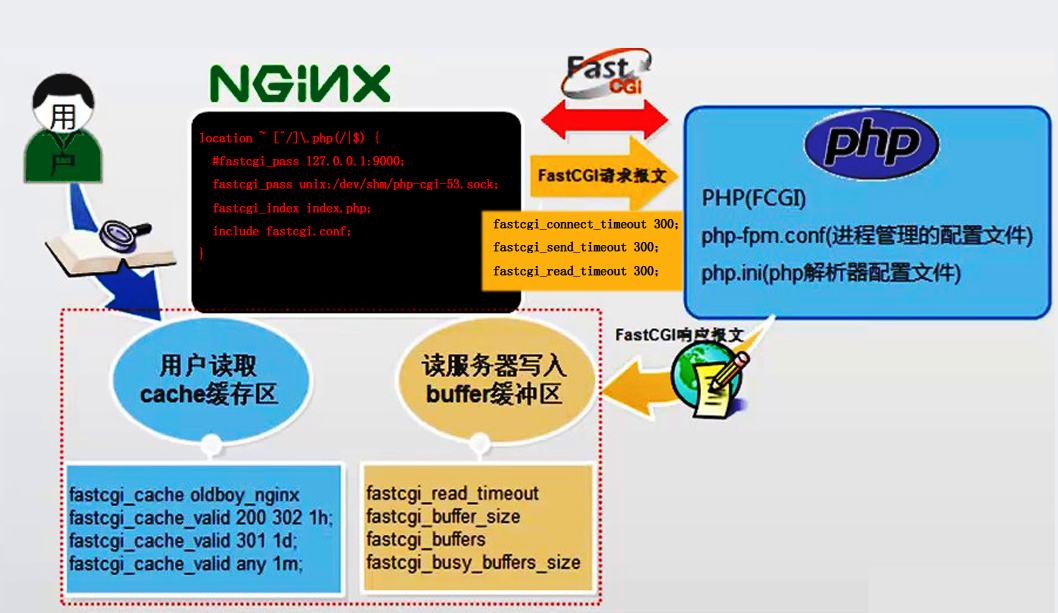
**Nginx优化**

**修改Nginx主进程启动用户**

待整理

**Fastcgi相关参数优化 ---http核心区配置**

****

****

**Nginx和Fastcgi之间的优化**

http {

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;

fastcgi\_intercept\_errors on;

{

fastcgi\_connect\_timeout 300; nginx服务端与后端Fastcgi服务器连接超时时间，默认60，这个参数值通常不超过75秒

fastcgi\_send\_timeout 300; 设置Nginx准许Fastcgi服务器端返回数据的超时时间，如果超时nginx断开连接，默认60秒

fastcgi\_read\_timeout 300; 设置nginx从Fastcgi读取响应信息的超时时间，Nginx等待后端服务器的响应时间

fastcgi\_buffer\_size 64k; 缓冲区大小、设定nginx读取从Fstcgi服务器端收到的第一部分响应信息的缓冲区大小，

fastcgi\_buffers 4 64k; 指定缓冲区大小和数量、

fastcgi\_busy\_buffers\_size 128k; 繁忙时候的buffers大小，普通buffer\*2

fastcgi\_temp\_file\_write\_size 128k; fastcgi临时文件的大小

fastcgi\_intercept\_errors on; 错误页面相关

**http核心配置**

http{

fastcgi\_intercept\_errors on;

fastcgi\_connect\_timeout 300;

fastcgi\_send\_timeout 300;

fastcgi\_read\_timeout 300;

fastcgi\_buffer\_size 64k;

fastcgi\_buffers 8 64k;

fastcgi\_busy\_buffers\_size 128k;

fastcgi\_temp\_file\_write\_size 128k;

#fastcgi\_temp\_path /data/ngx\_fcgi\_tmp;

fastcgi\_cache\_path /data/fastcgi\_cache levels=2:2 keys\_zone=ngx\_fcgi\_cache:512m inactive=1d max\_size=10g;

}

server{

location ~ [^/]\.php(/|$){

try\_files $uri =404;

fastcgi\_pass unix:/tmp/php-cgi-70.sock;

fastcgi\_index index.php;

include fastcgi.conf;

fastcgi\_cache ngx\_fcgi\_cache ;

fastcgi\_cache\_valid 200 302 1h;

fastcgi\_cache\_valid 301 1d;

fastcgi\_cache\_valid any 1m;

fastcgi\_cache\_min\_uses 1;

fastcgi\_cache\_use\_stale error timeout invalid\_header http\_500;

fastcgi\_cache\_key "$request\_method://$host$request\_uri";

#If you have a lot of static files to serve through Nginx then caching of the files' metadata (not the actual files' contents) can save some latency.

#open\_file\_cache max=65535 inactive=20s;

#open\_file\_cache\_min\_uses 1;

#open\_file\_cache\_valid 30s;

}

open\_file\_cache max=102400 inac0074ive=20s;

这个将为打开文件指定缓存，默认是没有启用的，max 指定缓存数量，建议和打开文件数一致，

inactive 是指经过多长时间文件没被请求后删除缓存。

open\_file\_cache\_valid 30s;这个是指多长时间检查一次缓存的有效信息。

open\_file\_cache\_min\_uses 1;

open\_file\_cache 指令中的inactive 参数时间内文件的最少使用次数，如果超过这个数字，文件描述符一直是在缓存中打开的，如上例，如果有一个文件在inactive 时间内一次没被使用，它将被移除。

**Nginx gzip压缩功能---http核心区配置—性能优化**

gzip压缩：提升网站用户体验，节约网站带宽成本，会消耗一些CPU资源，可以忽略不计。

纯文本内容压缩比较高，例如：html,js,css.xml,shtml

被压缩的纯文本资源必须要大于1kb

图片，视频，文件尽量不要压缩

依赖于 ngx\_http\_gzip\_module 模块

http {

#Gzip Compression

gzip on; #开启gzip压缩

gzip\_min\_length 1k; #设置准许压缩的页面最小字节数，小于1024的反而会越压缩越大

gzip\_buffers 4 16k; 16 8k; #压缩缓冲区大小，表示4个单位为16kb的内存作为压缩结果流缓存

gzip\_comp\_level 2; #压缩比率，1压缩最小（处理速度最快），9压缩比最大（传输速度快，处理慢，也比较消耗CPU资源）

gzip\_http\_version 1.1; #压缩版本，用于识别http协议版本，默认1.1

gzip\_proxied expired no-cache no-store private auth;

gzip\_vary on; #可以让前端缓存服务器经过gzip压缩的页面，例如squid缓存经过nginx压缩的数据

#from bt set

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

gzip\_types

text/xml application/xml application/atom+xml application/rss+xml application/xhtml+xml image/svg+xml

text/javascript application/javascript application/x-javascript

text/x-json application/json application/x-web-app-manifest+json

text/css text/plain text/x-component

font/opentype application/x-font-ttf application/vnd.ms-fontobject

image/x-icon;

gzip\_disable "MSIE [1-6]\.(?!.\*SV1)"; #IE 6不压缩

{

gzip\_types说明 在 mime.types 根据需求配置

说明：

gzip\_proxied

语法: gzip\_proxied [off|expired|no-cache|no-store|private|no\_last\_modified|no\_etag|auth|any] …  
默认值: gzip\_proxied off  
作用域: http, server, location  
Nginx作为反向代理的时候启用，开启或者关闭后端服务器返回的结果，匹配的前提是后端服务器必须要返回包含”Via”的 header头。

off – 关闭所有的代理结果数据的压缩  
expired – 启用压缩，如果header头中包含 “Expires” 头信息  
no-cache – 启用压缩，如果header头中包含 “Cache-Control:no-cache” 头信息  
no-store – 启用压缩，如果header头中包含 “Cache-Control:no-store” 头信息  
private – 启用压缩，如果header头中包含 “Cache-Control:private” 头信息  
no\_last\_modified – 启用压缩,如果header头中不包含 “Last-Modified” 头信息  
no\_etag – 启用压缩 ,如果header头中不包含 “ETag” 头信息  
auth – 启用压缩 , 如果header头中包含 “Authorization” 头信息  
any – 无条件启用压缩

**Nginx expires缓存实现性能优化--—客户端（浏览器）----server核心区配置**

一颗死拜儿

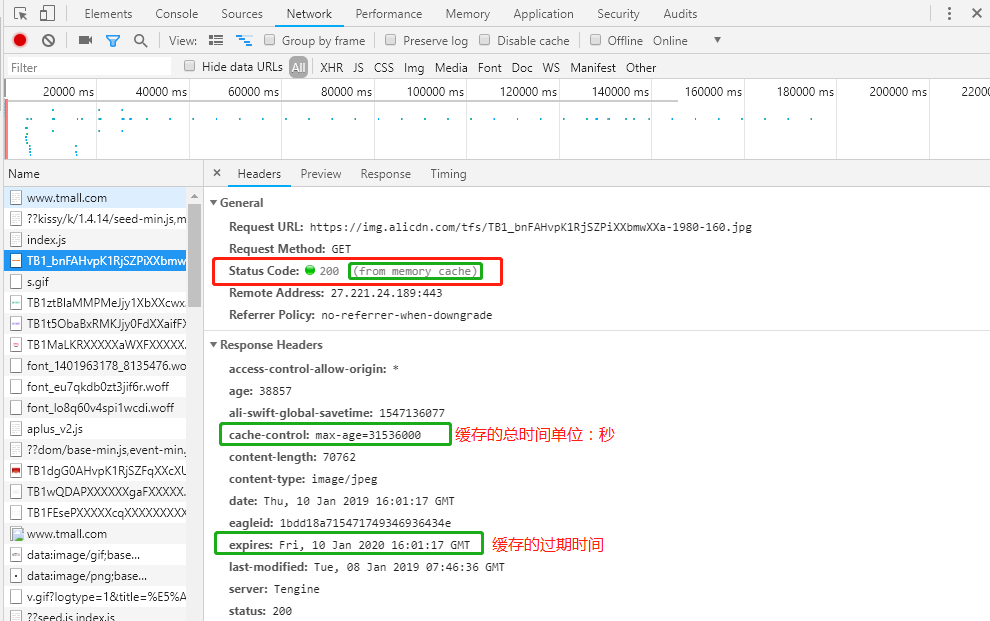
expires 降低网站的带宽，节约成本，加速用户访问网站速度，提供用户访问体验，

expires的功能就是准许通过nginx配置文件控制HTTP的”Expires”和”Cache-Control（开吃肯愁）”响应头部内容，

告诉客户端浏览器是否缓存和缓存多久以内访问的内容，

简单点说：根据nginx配置文件中的设置，把文件缓存到浏览器，包含过期时间、下次访问就会从客户端浏览缓存读取，

缓存的内容无需重复下载减少服务器访问



缓存图片：

#from bt set

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

location ~ .\*\.(gif|jpg|jpeg|png|bmp|swf|flv|mp4|ico)$ {

expires 30d;

error\_log off;

access\_log off;

}

缓存js css

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

expires 7d;

error\_log off;

access\_log off;

{

根据url中的目录进行判断：

location ~ ^/(images|javascript|js|css|flash|media|static)/ {

expires 30d;

access\_log off;

}

**说明：广告图片，网站流量统计JS代码、更新很频繁的文件， 最好别缓存**

**新浪：1天**

**京东 ：N年**

**淘宝：N年**

**Nginx站点目录及文件URL访问控制---server核心区配置**

403错误 写到php-location前面（暂未做验证）

return 404;

return 403;

deny all;

**#set from bt**

location ~ ^/(\.user.ini|\.htaccess|\.git|\.svn|\.project|LICENSE|README.md)

{

return 404;

}

**#set from one**

location ~ ^/(\.user.ini|\.ht|\.git|\.svn|\.project|LICENSE|README.md)

{

deny all;

}

location ~ /\.

{

deny all;

}

**#set from oldboy**

location ~ ^/(images|uploads)/.\*\.(php|php5|sh|pl|py)$

{

deny all;

}

location ~ ^/static/.\*\.(php|php5|sh|pl|py)$

{

deny all;

}

location ~\* ^/data/(attachment|avatar)/.\*\.(php|php5)$

{

deny all;

}

**配置禁止访问指定的单个目录或多个目录**

location ~ ^/(static)/

{

deny all;

}

location ~ ^/static

{

deny all;

}

location ~ ^/(static|backup|install)/

{

deny all;

}

**配置禁止访问\*.txt**

#拒绝或者跳转到url

location ~\* \.(txt|doc)$

{

if (-f $request\_filename)

{

rewrite \*\*\*\*\*\*\*\*

}

deny all;

}

**Nginx拒绝IP访问--- server核心区配置**

location /

{

deny 10.10.10.0/24;

allow all;

}

以deny结尾，表示除了上面准许的，其他的都禁止

location /

{

deny 10.10.10.100;

allow 10.10.10.0/24

allow 127.0.0.0/24

deny all;

}

deny 一定要加一个IP、否则会直接跳转403，如果403默认页在同一域名下，会造成死循环访问

对于allow的IP段，从准许访问的段位从小到大排列：比如127.0.0.0/24 下面才能是10.10.10.0/16

**Nginx 禁止非法域名解析访问企业网站-- server核心区配置**

nginx服务器对于服务器没有绑定域名的请求，默认访问第一个conf，即第一个虚拟主机，在sever前面加入如下

方法一：

server ｛

listen 80 default\_server;

server\_name \_;

return 501;

｝

方法二：

server ｛

listen 80 default\_server;

server\_name \_;

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

｝

**Nginx图片及目录防盗链解决方案--- server核心区配置**

注重网站流量图、关注流量变化、关注异常流量

对访问日志做分析

根据http referer实现防盗链

location ~\* ^.+\.(jpg|jpeg|png|swf|flv|rar|zip)$

{

valid\_referers none blocked \*.linyaohong.com linyaohong.com;

if ($invalid\_referer){

rewrite ^/ http://bbs.linyaohong.com/img/nolink.gif;

}

expires 30d;

access\_log off;

root html/www;

}

location ~ .\*\.(jpg|jpeg|gif|png|js|css)$

{

expires 30d;

access\_log off;

valid\_referers none blocked linyaohong.com;

if ($invalid\_referer){

return 404;

}

}

开启防盗链-

**分析图片服务日志，把日志（每个图片访问次数\*图片大小的总和）排行，取top10，也就是计算每个url的总访问大小**

这个功能可以用于IDC及CDN网站流量带宽很高，然后通过分析服务器日志哪些元素占用流量过大，进而进行优化裁剪该图片

---------------------------------------------------------------------------------------------------------------------------------

awk '{array\_num[$7]++;array\_size[$7]+=$10}END{for(x in array\_num){print array\_size[x],array\_num[x],x}}' shangquan888.vip.log |sort -rn -k1|head -10 >1.log

awk '{print $7"\t" $10}' shangquan888.vip.log|awk '{S[$1]+=$2;S1[$1]+=1}END{for(i in S) print S[i],S1[i],i}'|sort -rn|head -10 >2.log

---------------------------------------------------------------------------------------------------------------------------------

**Nginx 错误日志显示或优雅显示--- server核心区配置**

#error\_page 404 /404.html;

#error\_page 502 500 503 504 /502.html;

#error\_page 502 500 503 504 <http://www.linyaohong.com/error.html>

**Nginx 限制单IP并发--- server核心区配置**

nginx主配置文件http添加

limit\_conn\_zone $binary\_remote\_addr zone=perip:10m;

limit\_conn\_zone $server\_name zone=perserver:10m;

server段添加 流量限制

limit\_conn perserver 300; #限制当前站点并发

limit\_conn perip 25; #限制单个IP访问最大并发数

limit\_rate 512k; #限制每个请求的流量上限KB

限制速率：即1r/s 1秒内处理1个请求

nginx主配置文件http添加

limit\_req\_zone $binary\_remote\_addr zone=one:10m rate=1r/s;

以请求的客户端IP作为key值，内存区域命名为zone，分配10m的空间，访问速率限制为1秒1次请求

server段添加

limit\_req zone=one burst=5;

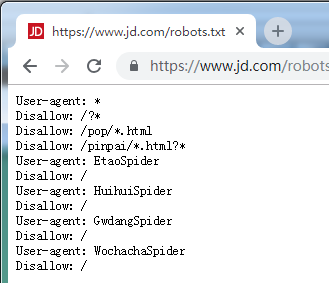
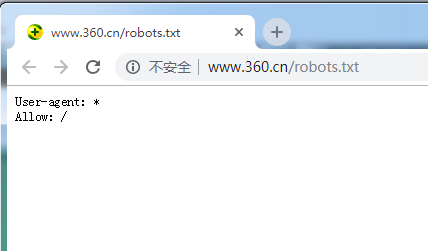
使用前面定的名为zone的内存空间，队列值为5，即有5个请求排队等待

说明：

**日他先人没测试成功**

**官方文档：http://nginx.org/en/docs/http/ngx\_http\_limit\_conn\_module.html**

**robots爬虫协议[君子协议]--- server核心区配置**

#禁止Scrapy等工具的抓取

if ($http\_user\_agent ~\* (Scrapy|Curl|HttpClient)) {

return 403;

}

**模拟蜘蛛抓取**

curl -I -A 'YisouSpider' linyaohong.com

#禁止指定UA及UA为空的访问

if ($http\_user\_agent ~\* "FeedDemon|Indy Library|Alexa Toolbar|AskTbFXTV|AhrefsBot|CrawlDaddy|CoolpadWebkit|Java|Feedly|UniversalFeedParser|ApacheBench|Microsoft URL Control|Swiftbot|ZmEu|oBot|jaunty|Python-urllib|lightDeckReports Bot|YYSpider|DigExt|HttpClient|MJ12bot|heritrix|EasouSpider|Ezooms|^$" ) {

     return 403;

}

#禁止指定UA及UA为空的访问

if ($http\_user\_agent ~ "WinHttp|WebZIP|FetchURL|node-superagent|java|FeedDemon|Jullo|JikeSpider|Indy Library|Alexa Toolbar|AskTbFXTV|AhrefsBot|CrawlDaddy|Java|Feedly|Apache-HttpAsyncClient|UniversalFeedParser|ApacheBench|Microsoft URL Control|Swiftbot|ZmEu|oBot|jaunty|Python-urllib|lightDeckReports Bot|YYSpider|DigExt|HttpClient|MJ12bot|heritrix|EasouSpider|Ezooms|BOT/0.1|YandexBot|FlightDeckReports|Linguee Bot|^$" ) {

return 403;

}

**^$ 会拒绝- 即 没有user\_agent 的客户端**

#禁GET方式的抓取，例如图片上传服务器

if ($request\_method ~\* ^(GET)$) {

return 403;

}

#禁止非GET|HEAD|POST方式的抓取

if ($request\_method !~ ^(GET|HEAD|POST)$) {

return 403;

}

#屏蔽单个IP的命令是

#deny 123.45.6.7

#封整个段即从123.0.0.1到123.255.255.254的命令

#deny 123.0.0.0/8

#封IP段即从123.45.0.1到123.45.255.254的命令

#deny 124.45.0.0/16

#封IP段即从123.45.6.1到123.45.6.254的命令是

#deny 123.45.6.0/24

# 以下IP皆为流氓

deny 58.95.66.0/24;

##禁止IE和火狐浏览器案例

if ($http\_user\_agent ~\* "Firefox|MSIE") {

return 403;

}

**附录：UA 收集**

FeedDemon 内容采集

BOT/0.1 (BOT for JCE) sql注入

CrawlDaddy sql注入

Java 内容采集

Jullo 内容采集

Feedly 内容采集

UniversalFeedParser 内容采集

ApacheBench cc攻击器

Swiftbot 无用爬虫

YandexBot 无用爬虫

AhrefsBot 无用爬虫

YisouSpider 无用爬虫（已被UC神马搜索收购，此蜘蛛可以放开！）

jikeSpider 无用爬虫

MJ12bot 无用爬虫

ZmEu phpmyadmin 漏洞扫描

WinHttp 采集cc攻击

EasouSpider 无用爬虫

HttpClient tcp攻击

Microsoft URL Control 扫描

YYSpider 无用爬虫

jaunty wordpress爆破扫描器

oBot 无用爬虫

Python-urllib 内容采集

Indy Library 扫描

FlightDeckReports Bot 无用爬虫

**Nginx 配置https 对指定的url不跳转https**

location / {

#如果url不匹配这个则进行跳转https，匹配则走本地的root查询内容

if ($request\_uri !~ '^/images/') {

return 301 <https://$server_name$request_uri>;

}

}

禁止解析.php .sh .pl **Nginx 目录权限--- 其他优化**

上传upload集群

程序目录 f 644 d 755 root root

上传目录 f 644 d 755 web web

禁止解析.php .sh .pl.py

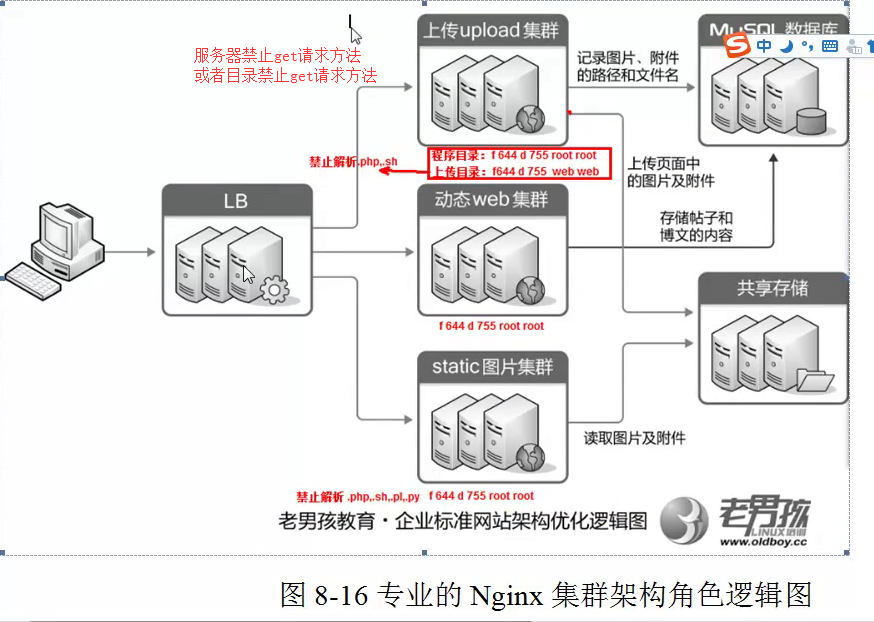
动态web集群

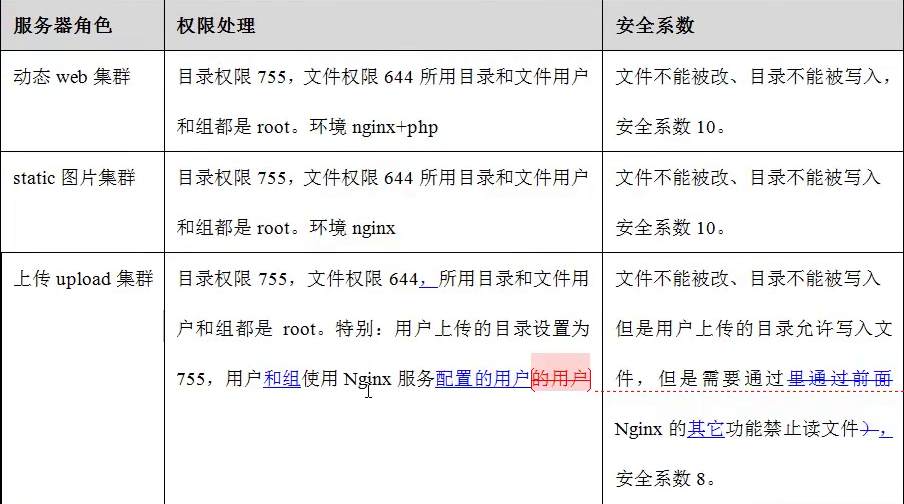
f 644 d 755 root root

static图片集群

禁止解析.php .sh .pl.py

r 644 d 755 root root





<https://www.cnblogs.com/zhang-shijie/p/5428640.html>

**关于内核参数的优化**

|  |  |
| --- | --- |
| 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\_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\_keepalive\_time = 600 |  |
| net.ipv4.tcp\_keepalive\_probes = 3 |  |
| 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.ipv4.tcp\_max\_orphans = 327680 | 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 = 16384 | 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\_fin\_timeout = 15 | 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， |
| net.core.netdev\_max\_backlog = 32768 | 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.core.somaxconn = 32768 | 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 定义的 NGX\_LISTEN\_BACKLOG 默认为 511，所以有必要调整这个值。 |
| 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 |
|  |  |

**Nginx优化小结**

**l.gzip压缩优化**

**2、expires缓存优化**

**3、网络IO事件模型优访**

**4、隐藏软件名称和版本**

**5.防盗链优化**

**6、禁止恶意域名解析**

**7、禁止通过ip地址访问网站**

**8、HTTP请求方法优化**

**9、防DOS攻击单IP并发连接的控制，与连接速率控**

**10、严格设置Web站点目录的权限**

**11、将Nginx进程以及站点运行于监牢模式**

**12、通过robot协议以及HTTP\_USER\_AGENT防爬虫优化**

**13、配置错误页面根据错误码指指定网页反馈给用户**

**14、Nginx曰志相关优设**

**访问日志切割轮询、不记录指定元素日志、最小化日志目录权限**

**15、限制上传到资源目录的程序被访问，防止木马入侵系统破坏文件**

**16、FastCGI参数buffer和cache以及超时等的优化**

**17、php.ini和php-fpm.conf配置文件的优化**

**18、有关Web服务的litmx内核方面深度优化(网络连接、IO、内存等)**

**19、Nginx加密传输优化SSL**

**20、web服务器磁盘挂载及网络文件系统优化**

**21、使用nginx\_cache缓存**

**22、nginx\_waf lua**

**Nginx由浅入深安装**

**jemalloc优化MySQL、Nginx内存管理**

**内存优化:ptmalloc、tcmalloc和jemalloc**

**系统的物理内存是有限的，而对内存的需求是变化的, 程序的动态性越强，内存管理就越重要，选择合适的内存管理算法会带来明显的性能提升。**

**软件github地址：https://github.com/jemalloc/jemalloc/tags**

**安装步骤：**

/usr/bin/wget http://106.14.199.213/sm/src/jemalloc- 5.1.0.tar.bz2

tar xjf jemalloc- 5.1.0.tar.bz2

./configure

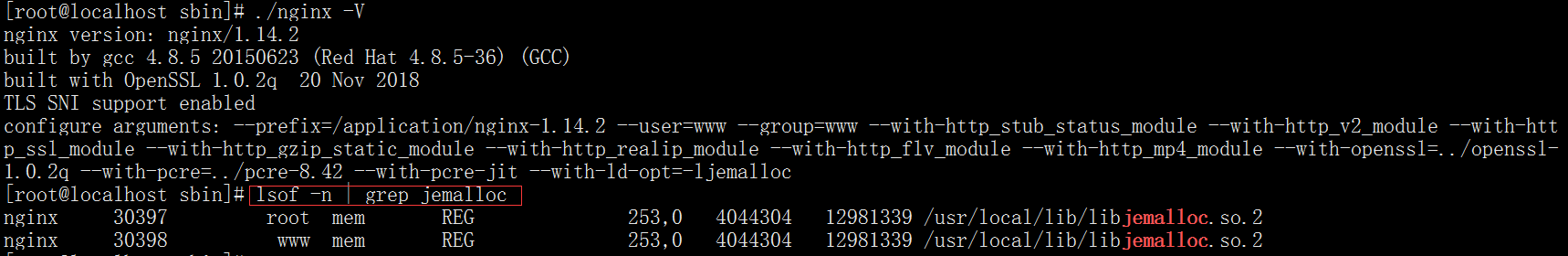
make && make install

ln -s /usr/local/lib/libjemalloc.so.2 /usr/lib64/libjemalloc.so.1

echo '/usr/local/lib' > /etc/ld.so.conf.d/local.conf

安装完成后可以

lsof -n | grep jemalloc 来查看效果



**安装脚本**

#!/bin/bash

jemalloc\_ver=5.1.0

THREAD=$(grep 'processor' /proc/cpuinfo | sort -u | wc -l)

yum install bzip2

if [ ! -e "/usr/local/lib/libjemalloc.so" ]; then

/usr/bin/wget http://106.14.199.213/sm/src/jemalloc-${jemalloc\_ver}.tar.bz2

tar xjf jemalloc-${jemalloc\_ver}.tar.bz2

cd jemalloc-$jemalloc\_ver

LDFLAGS="${LDFLAGS} -lrt" ./configure

make -j ${THREAD} && make install

unset LDFLAGS

cd ..

if [ -f "/usr/local/lib/libjemalloc.so" ]; then

#if [ "${OS\_BIT}" == '64' -a "${OS}" == 'CentOS' ]; then

ln -s /usr/local/lib/libjemalloc.so.2 /usr/lib64/libjemalloc.so.1

#else

#ln -s /usr/local/lib/libjemalloc.so.2 /usr/lib/libjemalloc.so.1

#fi

[ -z "`grep /usr/local/lib /etc/ld.so.conf.d/\*.conf`" ] && echo '/usr/local/lib' > /etc/ld.so.conf.d/local.conf

ldconfig

echo "jemalloc module installed successfully! "

rm -rf jemalloc-${jemalloc\_ver}

else

echo "jemalloc install failed, Please contact the author! "

kill -9 $$

fi

else

echo "libjemalloc.so is existing"

fi

说明：CFLAGS 表示用于C编译器的选项   
CXXFLAGS 表示用于C++编译器的选项

**进阶安装nginx：**

**####针对编译参数三：支持Nginx\_WAF**

wget http://luajit.org/download/LuaJIT-2.0.5.tar.gz

wget <https://github.com/simplresty/ngx_devel_kit/archive/v0.3.0.tar.gz>

wget <https://github.com/openresty/lua-nginx-module/archive/v0.10.13.tar.gz>

tar zxvf LuaJIT-2.0.5.tar.gz

tar zxvf v0.10.13.tar.gz

tar zxvf v0.3.0.tar.gz

[root@localhost home]# ll

total 8268

drwxr-xr-x. 9 1001 1001 186 Jan 8 00:41 nginx-1.14.2

-rw-r--r--. 1 root root 1015384 Jan 7 09:02 nginx-1.14.2.tar.gz

drwxr-xr-x. 21 root root 4096 Jan 8 00:43 openssl-1.0.2q

-rw-r--r--. 1 root root 5345604 Jan 7 09:02 openssl-1.0.2q.tar.gz

drwxr-xr-x. 9 1169 1169 8192 Jan 8 00:44 pcre-8.42

-rw-r--r--. 1 root root 2081413 Jan 7 09:02 pcre-8.42.tar.gz

drwxr-xr-x. 14 501 games 4096 Jan 15 2017 zlib-1.2.11

编译参数一

./configure --prefix=/application/nginx-1.14.2 --user=www --group=www --with-http\_stub\_status\_module --with-http\_v2\_module --with-http\_ssl\_module --with-http\_gzip\_static\_module --with-http\_realip\_module --with-http\_flv\_module --with-http\_mp4\_module --with-openssl=../openssl-1.0.2q --with-pcre=../pcre-8.42 --with-pcre-jit --with-ld-opt='-ljemalloc'

编译参数二

./configure --prefix=/application/nginx-1.14.2 --user=www --group=www --with-http\_stub\_status\_module --with-http\_v2\_module --with-http\_ssl\_module --with-http\_gzip\_static\_module --with-http\_realip\_module --with-http\_flv\_module --with-http\_mp4\_module --with-openssl=../openssl-1.0.2q --with-pcre=../pcre-8.42 --with-pcre-jit --with-ld-opt='-ljemalloc' --with-zlib=../zlib-1.2.11

编译参数三：

./configure --prefix=/application/nginx-1.14.2 --user=www --group=www --with-http\_stub\_status\_module --with-http\_v2\_module --with-http\_ssl\_module --with-http\_gzip\_static\_module --with-http\_realip\_module --with-http\_flv\_module --with-http\_mp4\_module --with-openssl=../openssl-1.0.2q --with-pcre=../pcre-8.42 --with-pcre-jit --with-ld-opt='-ljemalloc' --with-zlib=../zlib-1.2.11 --with-file-aio --with-http\_dav\_module --with-http\_addition\_module --add-module=../lua-nginx-module-0.10.13/ --add-module=../ngx\_devel\_kit-0.3.0/

**编译参数三：（未整理）**

--add-module=/www/server/nginx/update/src/ngx\_cache\_purge --add-module=/www/server/nginx/update/src/nginx-sticky-module --with-http\_image\_filter\_module --with-http\_gunzip\_module --with-stream --with-ipv6 --with-http\_sub\_module --with-http\_addition\_module --with-ld-opt=-Wl,-E --with-pcre=pcre-8.40 --with-ld-opt=-ljemalloc

**--with-http\_dav\_module 这个模块增加一些HTTP和webdav扩展动作（PUT, DELETE, MKCOL, COPY和MOVE）**

**--with-file-aio enable file AIO support**

**--prefix= 指向安装目录**

**--sbin-path= 指向（执行）程序文件（nginx）**

**--conf-path= 指向配置文件（nginx.conf）**

**--error-log-path= 指向错误日志目录**

**--pid-path= 指向pid文件（nginx.pid）**

**--lock-path= 指向lock文件（nginx.lock）防止安装文件被别人利用，或自己误操作。**

**--user= 指定程序运行时的非特权用户**

**--group= 指定程序运行时的非特权用户组**

**--with-http\_stub\_status\_module 获取nginx自上次启动以来的工作状态**

**--with-http\_v2\_module 支持http2**

**--with-http\_ssl\_module 使支持https请求，需已安装openssl**

**--with-http\_gzip\_static\_module 支持在线实时压缩输出数据流**

**--with-http\_realip\_module 允许从请求标头更改客户端的IP地址值，默认为关**

**--with-http\_flv\_module 为Flash Video(FLV)文件 提供服务端伪流媒体支持支持提供寻求内存使用基于时间的偏移量文件，默认为关**

**--with-http\_mp4\_m**

**odule 为H.264/AAC文件，主要是以 .mp4、.m4v、和.m4a为扩展名的文件， 提供伪流媒体服务端支持。**

**--with-pcre-jit  编译PCRE库时增加“实时编译（**[**pcre\_jit**](http://tengine.taobao.org/nginx_docs/cn/docs/ngx_core_module.html#pcre_jit)**）”支持。**

**--with-openssl= 指向openssl安装目录**

**Nginx-proxy.conf（备用）**

cat >/application/nginx/conf/proxy.conf << EOF

proxy\_connect\_timeout 300s;

proxy\_send\_tmeout 900;

proxy\_read\_timeout 900;

proxy\_buffer\_size 32k;

proxy\_buffers 4 64k;

proxy\_busy\_buffers\_size 128k;

proxy\_redirect off;

proxy\_hide\_header Vary;

proxy\_set\_header Accept-Encoding '';

proxy\_set\_header Referer \$http\_referer;

proxy\_set\_header Cookie \$http\_cookie;

proxy\_set\_header Host \$host;

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

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

proxy\_set\_header X-Forwarded-Proto \$scheme;

EOF

**整理版nginx.conf主配置文件**

备注：

黄色加深部分必须修改的部分

user www www;

worker\_processes auto;

error\_log /server/logs/error\_nginx.log crit;

pid /var/run/nginx.pid;

worker\_rlimit\_nofile 51200;

events {

use epoll;

worker\_connections 51200;

multi\_accept on;

#accept\_mutex on;

}

http {

include mime.types;

default\_type application/octet-stream;

server\_tokens off;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 60 ;

client\_header\_timeout 15;

client\_body\_timeout 15;

send\_timeout 25;

client\_max\_body\_size 16m;

client\_body\_buffer\_size 8m;

server\_names\_hash\_bucket\_size 128;

client\_header\_buffer\_size 32k;

large\_client\_header\_buffers 4 32k;

fastcgi\_connect\_timeout 100;

fastcgi\_send\_timeout 100;

fastcgi\_read\_timeout 100;

fastcgi\_buffer\_size 64k;

fastcgi\_buffers 4 64k;

fastcgi\_busy\_buffers\_size 128k;

fastcgi\_temp\_file\_write\_size 128k;

fastcgi\_intercept\_errors on;

fastcgi\_cache\_path /data/fastcgi\_cache levels=2:2 keys\_zone=ngx\_fcgi\_cache:512m inactive=1d max\_size=10g;

gzip on;

gzip\_min\_length 1k;

gzip\_buffers 6 16k;

gzip\_comp\_level 2;

gzip\_http\_version 1.1;

gzip\_proxied expired no-cache no-store private auth;

gzip\_vary on;

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

gzip\_disable "MSIE [1-6]\.(?!.\*SV1)";

limit\_conn\_zone $binary\_remote\_addr zone=perip:10m;

limit\_conn\_zone $server\_name zone=perserver:10m;

access\_log off;

#Brotli Compression

#brotli on;

#brotli\_comp\_level 6;

#If you have a lot of static files to serve through Nginx then caching of the files' metadata (not the actual files' contents) can save some latency.

#open\_file\_cache max=1000 inactive=20s;

#open\_file\_cache\_valid 30s;

#open\_file\_cache\_min\_uses 2;

#open\_file\_cache\_errors on;

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

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

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

log\_format logstash\_json '{"@timestamp":"$time\_iso8601",'

'"host":"$server\_addr",'

'"clientip":"$remote\_addr",'

'"size":$body\_bytes\_sent,'

'"responsetime":$request\_time,'

'"upstreamtime":"$upstream\_response\_time",'

'"upstreamhost":"$upstream\_addr",'

'"http\_host":"$host",'

'"url":"$uri",'

'"domain":"$host",'

'"xff":"$http\_x\_forwarded\_for",'

'"referer":"$http\_referer",'

'"agent":"$http\_user\_agent",'

'"status":"$status"}';

server {

listen 80 default\_server;

server\_name \_;

return 501;

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

}

include vhost/\*.conf;

}

**整理版nginx-server配置文件**

备注：

黄色加深部分必须修改的部分

php引用conf下面的配置（方便切换版本）

rewrite规则引用（方便管理）

server

{

listen 80;

#listen 443 ssl http2;

server\_name linyaohong.com www.linyaohong.com;

index index.php index.html index.htm default.php default.htm default.html;

root /server/wwwroot/linyaohong.com;

limit\_conn perserver 20;

limit\_conn perip 1;

limit\_rate 512k;

#allow 10.10.10.200;

#allow 127.0.0.1;

#deny all;

#301-START

# if ($host ~ '^linyaohong.com')

# {

# return 301 http://www.linyaohong.com$request\_uri;

# }

#301-END

#HTTP\_TO\_HTTPS\_START

#if ($server\_port !~ 443){

# rewrite ^(/.\*)$ https://$host$1 permanent;

#}

error\_page 404 403 /404.html;

error\_page 502 /502.html;

#AUTH\_START

#auth\_basic "Authorization";

#auth\_basic\_user\_file /application/nginx/conf/passwd/linyaohong.com.pass;

#AUTH\_END

#SSL-START SSL

#ssl\_certificate /etc/ssl/fullchain.pem;

#ssl\_certificate\_key /etc/ssl/privkey.pem;

#ssl\_protocols TLSv1.1 TLSv1.2;

#ssl\_ciphers ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE;

#ssl\_prefer\_server\_ciphers on;

#ssl\_session\_cache shared:SSL:10m;

#ssl\_session\_timeout 10m;

#error\_page 497 https://$host$request\_uri;

#SSL-END

#SECURITY-START

location ~ .\*\.(jpg|jpeg|gif|png|js|css)$

{

expires 30d;

access\_log off;

valid\_referers none blocked www.linyaohong.com linyaohong.com;

if ($invalid\_referer){

return 404;

}

}

#include enable-php-53.conf;

include enable-php-54.conf;

#include enable-php-55.conf;

#include enable-php-56.conf;

#include enable-php-70.conf;

#REWRITE-START URL

#include /

application/nginx/conf/rewrite/www.linyaohong.com.conf;

#REWRITE-END

location ~ ^/(\.user.ini|\.htaccess|\.git|.svn|\.project|LICENSE|README.md) {

deny all;

}

location ~ ^/(images|uploads)/.\*\.(php|html|sh|pl|py)$ {

deny all;

}

location = /favicon.ico {

log\_not\_found off;

access\_log off;

}

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

{

expires 30d;

error\_log off;

access\_log off;

}

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

{

expires 12h;

error\_log off;

access\_log off;

}

#

#if ($http\_user\_agent ~\* (Scrapy|Curl|HttpClient)) {

# return 403;

# }

if ($request\_method !~ ^(GET|HEAD|POST)$) {

return 403;

}

#if ($http\_user\_agent ~\* "CoolpadWebkit|WinHttp|YisouSpider|WebBench|WebZIP|FetchURL|node-superagent|java|FeedDemon|Jullo|JikeSpider|Indy Library|Alexa Toolbar|AskTbFXTV|AhrefsBot|CrawlDaddy|Java|Feedly|Apache-HttpAsyncClient|UniversalFeedParser|ApacheBench|Microsoft URL Control|Swiftbot|ZmEu|oBot|jaunty|Python-urllib|lightDeckReports Bot|YYSpider|DigExt|HttpClient|MJ12bot|heritrix|EasouSpider|Ezooms|BOT/0.1|YandexBot|FlightDeckReports|Linguee Bot" ) {

# return 403;

# }

#access\_log /server/logs/www.linyaohong.com.log;

error\_log /server/logs/www.linyaohong.com.error.log;

access\_log /server/logs/www.linyaohong.com.access.log main;

#access\_log /server/logs/www.linyaohong.com.access.log logstash\_json;

#LOG Write to buffer first, 5s seconds write to hard disk

#access\_log /server/logs/www.linyaohong.com.access.log main gzip buffer=32k flush=5s;

**enable-php php**

[root@web01 conf]# cat enable-php-55.conf

location ~ [^/]\.php(/|$) {

#fastcgi\_pass 127.0.0.1:9000;

fastcgi\_pass unix:/dev/shm/php-cgi-53.sock;

fastcgi\_index index.php;

include fastcgi.conf;

include pathinfo.conf;

}

说明：

TCP是使用TCP端口连接 127.0.0.1:9000

Socket是使用unix domain socket连接套接字/dev/shm/php-cgi.sock（很多教程使用路径/tmp，而路径/dev/shm是个tmpfs，速度比磁盘快得多

**pathinfo.conf**

set $real\_script\_name $fastcgi\_script\_name;

if ($fastcgi\_script\_name ~ "^(.+?\.php)(/.+)$") {

set $real\_script\_name $1;

set $path\_info $2;

}

fastcgi\_param SCRIPT\_FILENAME $document\_root$real\_script\_name;

fastcgi\_param SCRIPT\_NAME $real\_script\_name;

fastcgi\_param PATH\_INFO $path\_info;

**enable-rewrite**

已上传

**Nginx启动脚本**

**说明：修改黄色pid安装目录 conf目录即可**

#! /bin/sh

# chkconfig: 2345 55 25

# Description: Startup script for nginx webserver on Debian. Place in /etc/init.d and

# run 'update-rc.d -f nginx defaults', or use the appropriate command on your

# distro. For CentOS/Redhat run: 'chkconfig --add nginx'

### BEGIN INIT INFO

# Provides: nginx

# Required-Start: $all

# Required-Stop: $all

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Short-Description: starts the nginx web server

# Description: starts nginx using start-stop-daemon

### END INIT INFO

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

NAME=nginx

NGINX\_BIN=/application/nginx-1.14.2/sbin/$NAME

CONFIGFILE=/application/nginx-1.14.2/conf/$NAME.conf

PIDFILE=/var/run/$NAME.pid

GREEN\_COLOR='\E[1;32m'

cgreen="\E[32m"

cend="\E[0m"

case "$1" in

start)

echo -n "Starting $NAME... "

if [ -f $PIDFILE ];then

mPID=`cat $PIDFILE`

isStart=`ps ax | awk '{ print $1 }' | grep -e "^${mPID}$"`

if [ "$isStart" != '' ];then

echo "$NAME (pid `pidof $NAME`) already running."

exit 1

fi

fi

$NGINX\_BIN -c $CONFIGFILE

if [ "$?" != 0 ] ; then

echo " failed"

exit 1

else

echo -e "${cgreen} [OK]${cend}"

fi

;;

stop)

echo -n "Stoping $NAME... "

if [ -f $PIDFILE ];then

mPID=`cat $PIDFILE`

isStart=`ps ax | awk '{ print $1 }' | grep -e "^${mPID}$"`

if [ "$isStart" = '' ];then

echo "$NAME is not running."

exit 1

fi

else

echo "$NAME is not running."

exit 1

fi

$NGINX\_BIN -s stop

if [ "$?" != 0 ] ; then

echo " failed. Use force-quit"

exit 1

else

echo -e "${cgreen} [OK]${cend}"

fi

;;

status)

if [ -f $PIDFILE ];then

mPID=`cat $PIDFILE`

isStart=`ps ax | awk '{ print $1 }' | grep -e "^${mPID}$"`

if [ "$isStart" != '' ];then

echo "$NAME (pid `pidof $NAME`) already running."

exit 1

else

echo "$NAME is stopped"

exit 0

fi

else

echo "$NAME is stopped"

exit 0

fi

;;

restart)

$0 stop

sleep 1

$0 start

;;

reload)

echo -n "Reload service $NAME... "

if [ -f $PIDFILE ];then

mPID=`cat $PIDFILE`

isStart=`ps ax | awk '{ print $1 }' | grep -e "^${mPID}$"`

if [ "$isStart" != '' ];then

$NGINX\_BIN -s reload

echo -e "${cgreen} [OK]${cend}"

else

echo "$NAME is not running, can't reload."

exit 1

fi

else

echo "$NAME is not running, can't reload."

exit 1

fi

;;

config)

echo -n "Test $NAME configure files... "

$NGINX\_BIN -t

;;

\*)

echo "Usage: $0 {start|stop|restart|reload|status|configtest}"

exit 1

;;

esac

**备份（网站-计划任务-服务器上常用脚本）脚本，并且做MD5校验、通过rsync备份到rsync\_backup服务器**

#!/bin/bash

IP=`ifconfig ens33|awk -F " " 'NR==2{print $2}'`

Path=/backup/${IP}

[ ! -d ${Path} ] && mkdir ${Path} -p

#bak

cd / && tar zcvf ${Path}/conf\_$(date +%F).tar.gz etc/rc.d/rc.local var/spool/cron/root server/scripts/

find ${Path}/ -type f -name "\*.tar.gz"|xargs md5sum > ${Path}/flag\_$(date +%F)

# bak to rsync server

rsync -az /backup/ backup@10.10.10.41::backup --password-file=/etc/rsync.passwd

#del before 7 day ago

find ${Path}/ -type f -name "\*.tar.gz" -mtime +7|xargs rm -f

**Nginx日志切割脚本**

#!/bin/bash

date=`date +%F`

#切割后备份的目录

logs\_backup\_path="/server/logs\_backup"

#日志存放目录

logs\_path="/server/logs/"

nginx\_path="/server/application/nginx"

if [ ! -d $logs\_path ];then

echo "The log\_path file does not exist"

exit 1

fi

[ ! -d $logs\_backup\_path ] && mkdir -p $logs\_backup\_path

cd ${logs\_path}

#for循环nginx日志文件，全部切割、适合多站点网站日志

for logs\_access in `ls -l|awk '{print $9}'|egrep -v "^$"`

do

/usr/bin/mv ${logs\_access} ${logs\_backup\_path}/${date}\_${logs\_access}

done

cd ${logs\_backup\_path}

if [ $? -eq 0 ];then

find . -type f -atime +7 -name "\*.log"|xargs rm -f

else

echo "delete 7 day fail"

exit 1

fi

${nginx\_path}/sbin/nginx -s reload