一　基础

1 Unix Linux

--Apache Nginx Tengine Lighttpd

--Tomcat IBM WebSphere Jboss

2 Windows平台

--IIS

3 netstat

netstat命令可以查看系统中启动的端口信息，该命令常用选项如下：

-a显示所有端口的信息

-n以数字格式显示端口号

-t显示TCP连接的端口

-u显示UDP连接的端口

-l显示服务正在监听的端口信息，如httpd启动后，会一直监听80端口

-p显示监听端口的服务名称是什么（也就是程序名称）

二　nginx

Tengine 是nginx的升级版(优化版)

1,源码包安装nginx软件包

[root@proxy ~]# yum -y install gcc pcre-devel openssl-devel //安装依赖包

gcc:支持c语言,c语言解释器

pcre-devel:支持正则

openssl-devel:支持安全ssl协议，https

[root@proxy ~]# useradd -s /sbin/nologin nginx #出于安全创建运行nginx的用户

[root@proxy ~]# tar -xf nginx-1.10.3.tar.gz

[root@proxy ~]# cd nginx-1.10.3

[root@proxy nginx-1.10.3]# ./configure \

> --prefix=/usr/local/nginx \ //指定安装路径

> --user=nginx \ //指定用户

> --group=nginx \ //指定组

> --with-http\_ssl\_module //开启SSL加密功能

.. ..

--with-模块 #开启模块

--without-模块　　　　　　　　 #禁用模块

2 升级nginx

1）编译新版本nginx软件

[root@proxy ~]# tar -zxvf nginx-1.12.2.tar.gz

[root@proxy ~]# cd nginx-1.12.2

[root@proxy nginx-1.12.2]# ./configure \

> --prefix=/usr/local/nginx \

> --user=nginx \

> --group=nginx \

> --with-http\_ssl\_module

[root@proxy nginx-1.12.2]# make

特别提示：千外不能make install ,否则就真的覆盖已有版本了

2) 备份老的nginx主程序，并使用编译好的新版本nginx替换老版本

[root@proxy nginx-1.12.2]# mv /usr/local/nginx/sbin/nginx \

>/usr/local/nginx/sbin/nginxold

[root@proxy nginx-1.12.2]# cp objs/nginx /usr/local/nginx/sbin/ //拷贝新版本

–杀死老nginx，启动nginx

killadd nginx #停止老nignx

nginx 　　　　　　　#启动

--make upgrade

[root@proxy nginx-1.12.2]# make upgrade //升级

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

nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok

nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful

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

sleep 1

test -f /usr/local/nginx/logs/nginx.pid.oldbin

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

[root@proxy ~]# /usr/local/nginx/sbin/nginx –v //查看版本

3 配置文件

1)日志文件

[root@proxy jluo]# ls /usr/local/nginx/logs/

access.log #每访问一次就有一行记录

error.log 　　　　　　　　　　　　　　　　#服务器的错误记录文件

nginx.pid　　　　　　　　　　　　　　　　#服务器运行就有这个文件

2)nginx.conf

--全局配置文件

worker\_processes 1; #开启进程数

events {

worker\_connections 1024;　　　　　　#一个进程并发量

}

--一个web服务配置

http{

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

##http普通网站

server {

listen 80;

server\_name localhost;

localtion / {

root html;

index index.html index.htm;

}

error\_page 500 502 503 504 /50x.html

location = /50x.html {

root html;

}

}

}

4 用户认证

访问Web页面需要进行用户认证

用户名为：tom，密码为：123456

1）修改/usr/local/nginx/conf/nginx.conf

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

server {

listen 80;

server\_name localhost;

auth\_basic "Input Password:"; //认证提示符,不能是汉字

auth\_basic\_user\_file "/usr/local/nginx/pass"; //认证密码文件

location / {

root html;

index index.html index.htm;

}

}

2）生成密码文件，创建用户及密码

使用htpasswd命令创建账户文件，需要确保系统中已经安装了httpd-tools。[root@proxy ~]# yum -y install httpd-tools

[root@proxy ~]# htpasswd -c /usr/local/nginx/pass tom //创建密码文件

New password:

Re-type new password:

Adding password for user tom

[root@proxy ~]# htpasswd /usr/local/nginx/pass jerry //追加用户，不使用-c选项

New password:

Re-type new password:

Adding password for user jerry

[root@proxy ~]# cat /usr/local/nginx/pass

3）重启Nginx服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload //重新加载配置文件

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory

5 基于域名的虚拟主机

实现两个基于域名的虚拟主机，域名分别为www.a.com和www.b.com

对域名为www.a.com的站点进行用户认证，用户名称为tom，密码为123456

1）修改Nginx服务配置，添加相关虚拟主机配置如下

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

server {

listen 80; //端口

server\_name www.a.com; //域名

　　　 auth\_basic "Input Password:"; //认证提示符

auth\_basic\_user\_file "/usr/local/nginx/pass"; //认证密码文件

location / {

root html; //指定网站根路径

index index.html index.htm;

}

}

… …

server {

listen 80; //端口

server\_name www.b.com; //域名

location / {

root www; //指定网站根路径

index index.html index.htm;

}

}

2）创建网站根目录及对应首页文件

[root@proxy ~]# mkdir /usr/local/nginx/www

[root@proxy ~]# echo "www" > /usr/local/nginx/www/index.html

3）重启nginx服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

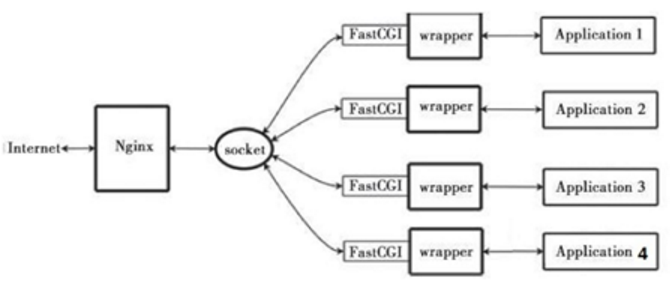
#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

4)

配置Fast-CGI支持PHP网页

创建PHP测试页面，测试使用PHP连接数据库的效果



注意，FastCGI的内存消耗问题，一个PHP-FPM解释器将消耗约25M的内存。

1）查看php-fpm配置文件（实验中不需要修改该文件）

[root@proxy etc]# vim /etc/php-fpm.d/www.conf

[www]

listen = 127.0.0.1:9000 //PHP端口号

pm.max\_children = 32 //最大进程数量

pm.start\_servers = 15 //最小进程数量

pm.min\_spare\_servers = 5 //最少需要几个空闲着的进程

pm.max\_spare\_servers = 32 //最多允许几个进程处于空闲状态

2)修改Nginx配置文件并启动服务location实现动静分离

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

location / { #匹配地址栏,/优先级最低

root html;

index index.php index.html index.htm;

#设置默认首页为index.php，当用户在浏览器地址栏中只写域名或IP，不说访问什么页面时，服务器会把默认首页index.php返回给用户

}

location ~ \.php$ {　　　　　　　　　　#匹配地址栏.php结尾，支持正则,～(正则匹配)完全匹配．php，区分大小写

root html;

fastcgi\_pass 127.0.0.1:9000; #将请求转发给本机9000端口，PHP解释器

fastcgi\_index index.php;

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

include fastcgi.conf;

}

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

３)测试

6　地址重写

所有访问a.html的请求，重定向到b.html;

所有访问192.168.4.5的请求重定向至www.tmooc.cn；

所有访问192.168.4.5/下面子页面，重定向至www.tmooc.cn/下相同的页面；

实现firefox与curl访问相同页面文件，返回不同的内容

rewrite regex replacement flag

rewrite 旧地址 新地址 [选项]

1）修改配置文件(访问a.html重定向到b.html)

--修改Nginx服务配置

location / {

root html;

index index.html index.htm;

rewrite /a\.html /b.html; #修改\.转意.

}

--重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

--客户端测试

[root@client ~]# firefox http://192.168.4.5/a.html

2)访问a.html重定向到b.html（跳转地址栏）

--修改Nginx服务配置

location / {

root html;

index index.html index.htm;

rewrite /a\.html /b.html redirect; #修改\.转意.

}

--重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

--客户端测试

[root@client ~]# firefox http://192.168.4.5/a.html

3)修改配置文件(访问192.168.4.5，重定向至www.tmooc.cn)

--修改Nginx服务配置

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

server {

listen 80;

server\_name localhost;

rewrite ^/ <http://www.tmooc.cn/>; #^/匹配根下的所有都会跳转到该页面

location / {

root html;

index index.html index.htm;

# rewrite /a.html /b.html redirect;

}

}

--重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

4)修改配置文件(访问192.168.4.5/下面子页面，重定向至www.tmooc.cn/下相同的页面)

--修改Nginx服务配置

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

server {

listen 80;

server\_name localhost;

rewrite ^/(.\*)$ http://www.tmooc.cn/$1;

location / {

root html;

index index.html index.htm;

# rewrite /a.html /b.html redirect;

}

}

--重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

５)修改配置文件(实现curl和火狐访问相同链接返回的页面不同)

用途：实现部署不同页面(手机端，电脑端)

--知识小记：

没有用’-’表示

192.168.4.254 - 　tom 　[31/Oct/2018:21:46:44 +0800] "　GET / HTTP/1.1" 　200 2 "-" "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"

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

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

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

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf　　　　　　　　　　　#加入如下三行

#这里，~符号代表正则匹配，匹配即可，\*符号代表不区分大小写，==精确匹配

if ($http\_user\_agent ~\* firefox) { //识别客户端firefox浏览器

rewrite ^(.\*)$ /firefox/$1;

}

--地址重写格式【总结】

rewrite 旧地址 新地址 [选项];

last 不再读其他rewrite

break 不再读其他语句，结束请求

redirect 临时重定向

permament 永久重定向

7 Nginx反向代理

调度处理，自动健康检查

\*\*部署实施后端Web服务器

1）部署后端Web1服务器

后端Web服务器可以简单使用yum方式安装httpd实现Web服务，为了可以看出后端服务器的不同，可以将两台后端服务器的首页文档内容设置为不同的内容。

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "192.168.2.100" > /var/www/html/index.html

[root@web1 ~]# systemctl restart httpd

[root@web1 ~]# firewall-cmd --set-default-zone=trusted

[root@web1 ~]# setenforce 0

2）部署后端Web2服务器

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# echo "192.168.2.200" > /var/www/html/index.html

[root@web2 ~]# systemctl restart httpd

[root@web2 ~]# firewall-cmd --set-default-zone=trusted

[root@web2 ~]# setenforce 0

\*\*配置Nginx服务器，添加服务器池，实现反向代理功能

1）修改/usr/local/nginx/conf/nginx.conf配置文件

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

http {

.. ..

#使用upstream定义后端服务器集群，集群名称任意(如webserver)

#使用server定义集群中的具体服务器和端口

upstream webserver {

server 192.168.2.100:80;

server 192.168.2.200:80;

}

.. ..

server {

listen 80;

server\_name localhost;

location / {

#通过proxy\_pass将用户的请求转发给webserver集群

proxy\_pass http://webserver;

}

}

2）重启nginx服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

3）客户端使用浏览器访问代理服务器测试轮询效果

[root@client ~]# curl http://192.168.4.5 //使用该命令多次访问查看效果

[root@client ~]# curl http://192.168.4.5 //使用该命令多次访问查看效果

8 配置upstream服务器集群池属性

1）设置失败次数，超时时间，权重

weight可以设置后台服务器的权重，max\_fails可以设置后台服务器的失败次数，fail\_timeout可以设置后台服务器的失败超时时间。

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

http {

.. ..

upstream webserver {

server 192.168.2.100 weight=1 max\_fails=1 fail\_timeout=30;

server 192.168.2.200 weight=2 max\_fails=2 fail\_timeout=30;

server 192.168.2.101 down;

}

#weight设置服务器权重值，默认值为1

#max\_fails设置最大失败次数

#fail\_timeout设置失败超时时间，单位为秒

#down标记服务器已关机，不参与集群调度

.. ..

server {

listen 80;

server\_name localhost;

location / {

proxy\_pass http://webserver;

}

}

2）重启nginx服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

3）关闭一台后端服务器（如web1）

[root@web1 ~]# systemctl stop httpd

4）客户端使用浏览器访问代理服务器测试轮询效果

[root@client ~]# curl http://192.168.4.5 //使用该命令多次访问查看效果

5）再次启动后端服务器的httpd（如web1）

[root@web1 ~]# systemctl start httpd

6）客户端再次使用浏览器访

问代理服务器测试轮询效果

[root@client ~]# curl http://192.168.4.5 //使用该命令多次访问查看效果

７)配置upstream服务器集群的调度算法(防止重复登录等)

--设置相同客户端访问相同Web服务器

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

http {

.. ..

upstream webserver {

#通过ip\_hash设置调度规则为：相同客户端访问相同服务器

ip\_hash;

server 192.168.2.100 weight=1 max\_fails=2 fail\_timeout=10;

server 192.168.2.200 weight=2 max\_fails=2 fail\_timeout=10;

}

.. ..

server {

listen 80;

server\_name www.tarena.com;

location / {

proxy\_pass http://webserver;

}

}

ip\_hash　　#对前三个8字节ip进行hash(md5sum加密)，md5sum%２判断是否是同一个ip地址

--重启nginx服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

--客户端使用浏览器访问代理服务器测试轮询效果

[root@client ~]# curl http://192.168.4.5 //使用该命令多次访问查看效果

9　Nginx的TCP/UDP调度器

ngx\_stream\_core\_module模块，使用—with-stream开启该模块

提示：nginx1.9开始才支持

配置Nginx服务器，添加服务器池，实现TCP/UDP反向代理功能

1）部署nginx服务器

编译安装必须要使用--with-stream参数开启4层代理模块。

[root@proxy ~]# yum –y install gcc pcre-devel openssl-devel //安装依赖包

[root@proxy ~]# tar -xf nginx-1.12.2.tar.gz

[root@proxy ~]# cd nginx-1.12.2

[root@proxy nginx-1.12.2]# ./configure \

> --with-http\_ssl\_module //开启SSL加密功能

> --with-stream //开启4层反向代理功能

[root@proxy nginx-1.12.2]# make && make install //编译并安装

2）修改/usr/local/nginx/conf/nginx.conf配置文件

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

stream {

upstream backend {

server 192.168.2.100:22; //后端SSH服务器的IP和端口

server 192.168.2.200:22;

}

server {

listen 12345; //Nginx监听的端口

proxy\_connect\_timeout 1s;

proxy\_timeout 3s;

proxy\_pass backend;

}

}

http {

.. ..

}

3）重启nginx服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

4）客户端使用访问代理服务器测试轮询效果

[root@client ~]# ssh 192.168.4.5 -p 12345 //使用该命令多次访问查看效果

10　自定义报错页面

1）优化前，客户端使用浏览器访问不存在的页面，会提示404文件未找到

[root@client ~]# firefox http://192.168.4.5/xxxxx //访问一个不存在的页面

2）修改Nginx配置文件，自定义报错页面

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

error\_page 404 /40x.html; //自定义错误页面

.. ..

[root@proxy ~]# vim /usr/local/nginx/html/40x.html //生成错误页面

Oops,No NO no page …

[root@proxy ~]# nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

3）优化后，客户端使用浏览器访问不存在的页面，会提示自己定义的40x.html页面

[root@client ~]# firefox http://192.168.4.5/xxxxx //访问一个不存在的页面

11　如何查看服务器状态信息（非常重要的功能）

1）编译安装时使用--with-http\_stub\_status\_module开启状态页面模块

[root@proxy ~]# yum -y install gcc pcre-devel openssl-devel //安装常见依赖包

[root@proxy ~]# tar -zxvf nginx-1.12.2.tar.gz

[root@proxy ~]# cd nginx-1.12.2

[root@proxy nginx-1.12.2]# ./configure \

> --with-http\_ssl\_module //开启SSL加密功能

> --with-stream //开启TCP/UDP代理模块

> --with-http\_stub\_status\_module //开启status状态页面

[root@proxy nginx-1.12.2]# make && make install //编译并安装

2）启用Nginx服务并查看监听端口状态

ss命令可以查看系统中启动的端口信息，该命令常用选项如下：

-a显示所有端口的信息

-n以数字格式显示端口号

-t显示TCP连接的端口

-u显示UDP连接的端口

-l显示服务正在监听的端口信息，如httpd启动后，会一直监听80端口

-p显示监听端口的服务名称是什么（也就是程序名称）

注意：在RHEL7系统中可以使用ss命令替代netstat命令，功能一样，选项一样。

[root@proxy ~]# /usr/local/nginx/sbin/nginx

[root@proxy ~]# netstat -anptu | grep nginx

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 10441/nginx

[root@proxy ~]# ss -anptu | grep nginx

3）修改Nginx配置文件，定义状态页面

[root@proxy ~]# cat /usr/local/nginx/conf/nginx.conf

… …

location /status {

stub\_status on;

#allow IP地址;

#deny IP地址;

}

… …

[root@proxy ~]# nginx

4）优化后，查看状态页面信息

[root@proxy ~]# curl http://192.168.4.5/status

Active connections: 1

server accepts handled requests

10 10 3

Reading: 0 Writing: 1 Waiting: 0

Active connections：当前活动的连接数量。

Accepts：已经接受客户端的连接总数量。

Handled：已经处理客户端的连接总数量。

（一般与accepts一致，除非服务器限制了连接数量）。

Requests：客户端发送的请求数量。

Reading：当前服务器正在读取客户端请求头的数量。

Writing：当前服务器正在写响应信息的数量。

Waiting：当前多少客户端在等待服务器的响应。

Http1.0 一次连接一次请求

http1.1 一次连接多次请求

keepalive\_timeout 30; 　#保持连接时间，超时断开tcp连接

测试：

[root@client ~]# yum -y isntall httpd-tool

[root@client ~]# ab -c 200 -n 200 http://192.168.4.5/

c：client 客户端

n：number 访问数量

100个客户端访问200次

12　优化Nginx并发量

1）优化前使用ab高并发测试

[root@proxy ~]# ab -n 2000 -c 2000 http://192.168.4.5/

Benchmarking 192.168.4.5 (be patient)

socket: Too many open files (24) //提示打开文件数量过多

2）修改Nginx配置文件，增加并发量

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

worker\_processes 2; //与CPU核心数量一致

events {

worker\_connections 65535; //每个worker最大并发连接数,尽量写大一点

use epoll;

}

.. ..

[root@proxy ~]# nginx -s reload

3）优化Linux内核参数（最大文件数量）

[root@proxy ~]# ulimit -a //查看所有属性值

[root@proxy ~]# ulimit -Hn 100000 //设置硬限制（临时规则）

[root@proxy ~]# ulimit -Sn 100000 //设置软限制（临时规则）

[root@proxy ~]# vim /etc/security/limits.conf

.. ..

\* soft nofile 100000

\* hard nofile 100000

#该配置文件分4列，分别如下：

#用户或组 硬限制或软限制 需要限制的项目 限制的值

4）优化后测试服务器并发量（因为客户端没调内核参数，所以在proxy测试）

[root@proxy ~]# ab -n 2000 -c 2000 http://192.168.4.5/

5)ss

ss是Socket Statistics的缩写。顾名思义，ss命令可以用来获取socket统计信息，它可以显示和netstat类似的内容。但ss的优势在于它能够显示更多更详细的有关TCP和连接状态的信息，而且比netstat更快速更高效。

ss -l 显示本地打开的所有端口

ss -pl 显示每个进程具体打开的socket

ss -t -a 显示所有tcp socket

ss -u -a 显示所有的UDP Socekt

ss -o state established '( dport = :smtp or sport = :smtp )' 显示所有已建立的SMTP连接

ss -o state established '( dport = :http or sport = :http )' 显示所有已建立的HTTP连接

ss -x src /tmp/.X11-unix/\* 找出所有连接X服务器的进程

ss -s 列出当前socket详细信息:

13 优化Nginx数据包头缓存

1）优化前，使用脚本测试长头部请求是否能获得响应

[root@proxy ~]# cat lnmp\_soft/buffer.sh

#!/bin/bash

URL=http://192.168.4.5/index.html?

for i in {1..5000}

do

URL=${URL}v$i=$i

done

curl $URL //经过5000次循环后，生成一个长的URL地址栏

[root@proxy ~]# ./buffer.sh

.. ..

<center><h1>414 Request-URI Too Large</h1></center> //提示头部信息过大

2）修改Nginx配置文件，增加数据包头部缓存大小

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

.. ..

http {

client\_header\_buffer\_size 1k; //默认请求包头信息的缓存

large\_client\_header\_buffers 4 4k; //大请求包头部信息的缓存个数与容量,生产环境够用了

.. ..

}

[root@proxy ~]# nginx -s reload

3）优化后，使用脚本测试长头部请求是否能获得响应

[root@proxy ~]#cat cat buffer.sh

#!/bin/bash

URL=http://192.168.4.5/index.html?

for i in {1..5000}

do

URL=${URL}v$i=$i

done

curl $URL

[root@proxy ~]# ./buffer.sh

14　浏览器本地缓存静态数据

1）使用Firefox浏览器查看缓存

以Firefox浏览器为例，在Firefox地址栏内输入about:cache将显示Firefox浏览器的缓存信息，如图-3所示，点击List Cache Entries可以查看详细信息

1. 清空firefox本地缓存数据，如图-4所示

3）修改Nginx配置文件，定义对静态页面的缓存时间

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

server {

listen 80;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

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

expires 30d; //定义客户端缓存时间为30天

}

}

[root@proxy ~]# cp /usr/share/backgrounds/day.jpg /usr/local/nginx/html

[root@proxy ~]# nginx -s reload

#请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

4）优化后，使用Firefox浏览器访问图片，再次查看缓存信息

[root@client ~]# firefox http://192.168.4.5/day.jpg

在Firefox地址栏内输入about:cache，查看本地缓存数据，查看是否有图片以及过期时间是否正确。

15　日志切割

日志文件越来越大怎么办？单个文件10G? 如何切割？（非常常见的面试题）

步骤：

1. 把旧的日志重命名

2.kill USR1 PID(nginx的进程PID号)

1）手动执行

备注：/usr/local/nginx/logs/nginx.pid文件中存放的是nginx的进程PID号。

[root@proxy ~]# mv access.log access2.log

[root@proxy ~]# kill -USR1 $(cat /usr/local/nginx/logs/nginx.pid)

2）自动完成

每周5的03点03分自动执行脚本完成日志切割工作。

[root@proxy ~]# vim /usr/local/nginx/logbak.sh

#!/bin/bash

date=`date +%Y%m%d`

logpath=/usr/local/nginx/logs

mv $logpath/access.log $logpath/access-$date.log

mv $logpath/error.log $logpath/error-$date.log

kill -USR1 $(cat $logpath/nginx.pid)

[root@proxy ~]# crontab -e

03 03 \* \* 5 /usr/local/nginx/logbak.sh

16　对页面进行压缩处理

提示：对多媒体文件不要压缩，太大了

1. 修改Nginx配置文件

[root@proxy ~]# cat /usr/local/nginx/conf/nginx.conf

http {

.. ..

gzip on; //开启压缩

gzip\_min\_length 1000; //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

gzip\_comp\_level 4; //压缩比率　　（1－9），数字越高，花费时间越多，压缩效率越好，一般选中间

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

//对特定文件压缩，类型参考mime.types，/usr/local/nginx/conf/mime.types

.. ..

}

17服务器内存缓存

1）如果需要处理大量静态文件，可以将文件缓存在内存，下次访问会更快。

http {

open\_file\_cache max=2000 inactive=20s;

open\_file\_cache\_valid 60s;

open\_file\_cache\_min\_uses 5;

open\_file\_cache\_errors off; #关闭缓存过期报错

//设置服务器最大缓存2000个文件句柄，关闭20秒内无请求的文件句柄

//文件句柄的有效时间是60秒，60秒后过期

//只有访问次数超过5次会被缓存

}

18