## **Nginx反向代理**

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

1. 部署后端Web2服务器

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

1. [root@web2 ~]# echo "192.168.2.200" > /var/www/html/index.html
2. [root@web2 ~]# systemctl restart httpd
3. [root@web2 ~]# firewall-cmd --set-default-zone=trusted
4. [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 webs {

server 192.168.2.100:80;

server 192.168.2.200:80;

}

server {

listen 80;

server\_name localhost;

location / {

proxy\_pass [http://webs;](http://webserver;) #通过proxy\_pass将用户的请求转发给webs集群

root html

}

3）客户端使用浏览器问代访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

配置upstream服务器集群池属性

upstream webs {

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

**步骤三：配置upstream服务器集群的调度算法**

**1）设置相同客户端访问相同Web服务器**

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; #只对前三位解析

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;

2.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

## **Nginx的TCP/UDP调度器**

步骤一：部署支持4层TCP/UDP代理的Nginx服务器

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,这模块必须要

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

步骤二：配置Nginx服务器，添加服务器池，实现TCP/UDP反向代理功能

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

stream {

upstream sshs { #sshs随便命名

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; 可写可不写

# proxy\_timeout 3s; 远程连接时间只有3s

proxy\_pass sshs;

}

}

http {

.. ..

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

**Nginx常见问题处理,根据状态码可以根据日志排错**



**当用户搜索的页面不在时，修改页面的画面，给用户好的体验**

**修改配置文件/usr/local/nginx/conf/nginx.conf**

charset uts-8; #打开并修改为支持中文

error\_page 404 /404.html; #打开

在/usr/local/nginx/htm/404.html 放入页面代码

error\_page 500 502 503 504 /50x.html; 也可以把所有的报错都搞一个页面

**步骤二：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

[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

http{

… …

location /status { 在location上面手写一个

stub\_status on;

#allow IP地址;

#deny IP地址;

}

.....

**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 #历史数据，

数据0:nginx -s stop

Reading: 0 Writing: 1 Waiting: 0 #时时数据

**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。**

**步骤三：优化Nginx并发量**

1）优化前使用ab高并发测试 安装 httpd-tools

[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

use epoll;

**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 #永久设置最大文件数量，配置里添加下面

.. ..

\* soft nofile 100000

\* hard nofile 100000

#用户或组 硬限制或软限制 需要限制的项目 限制的值

**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

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

}

注意：写在http内，server外，关闭proxy\_pass的转发,缓存相当于内存

当出现414问题时，就得解决缓存大小问题，即地址栏能接收更长的地址

**浏览器本地缓存静态数据,可以不用每次打开网页都重新下载**

**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。**

**2）清空firefox本地缓存数据，如图-4所示**

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 #客户访问都是在根目录

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

**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

扩展:Kill -l 查看kill -选项 kill -9 进程id

Kill -19 进程id #在后端进行 kll -18 进程id #恢复在前端

**对页面进行压缩处理**

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; //小文件不压缩,1000字节

gzip\_comp\_level 4; //压缩比率（1-9）数字越大，效果越好，

但时间越压缩时间越长，一般取中间值

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

**服务器内存缓存**

**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;

//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

}

##################################################################################

**总结：**

一．**Nginx反向代理**

1.部署两个后端web服务器页面/var/www/html/index.html

2.配置Nginx服务器，添加服务器池，实现方向代理功能

http {

......

upstream webs { #定义后端集群

server 192.168.2.100:80;

server 192.168.2.100:80;

}

注意：添加在server括号外，http内

server {

listen 80;

....

location / {

proxy\_pass <http://webs> #通过proxy\_pass将用户

...... 请求转发给webs集群

测试：curl http://192.168.4.5

1. **配置upstream服务器集群池属性**

http {

.....

upstream webs {

Ip\_hash;

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;

Server {

.......

Location /{

Proxy\_pass http://webs

.....

1. **Nginx的TCP/UDP的调度**

./configure --with-stream

make && make install

配置添加服务器池，实现ssh的后端调用

**stream { #在http外添加**

**upstream sshs {**

**server 192.168.2.100:22;**

**server 192.168.2.200:22;**

}

server { **#另外添加**

listen 12345;

**proxy\_pass sshs;**

}

测试： ssh 1923.168.4.5 -p 123456

1. **自定义报错页面**

**修改配置文件，自定义报错页面**

error\_page 404 /404.html;

charset uts-8; #支持中文

echo “自定义报错页面” > /usr/local/nginx/html/404.htnl

**五．查看服务器状态信息**

./configure --with-http\_stub\_status\_module

make && make install

netstat -anputl | grep nginx

**1.修改配置文件，定义状态页面**

**location /status { #在location上面手写一个**

**stub\_status on; “/”后面取名，在浏览器后面也要一样**

#allow ip地址

#deny ip地址

}

查看页面状态信息

curl http://192.168.4.5/status

**2.修改配置文件，增加并发量**

worker\_processes 1; #与cpu核心数量一致

worker\_connections 5000; #每个worker最大并发连接数

1. 优化LInux内核参数（最大文件数量）

ulimit -a #查看所以属性

ulimit -Hn 100000 #设置硬限制 （临时）

ulimit -Sn 100000 #软限制 （临时）

/etc/security/limits.conf

\* soft nofile 100000

\* hard nofile 100000

#用户或组 硬限制或软限制 需要限制的项目 限制的值

测试服务器并发量 ab -n 2000 -c 2000 http://192.168.4.5/

**六．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

页面信息提示414则表示URL地址栏头部信息过大

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

.. ..

3）优化后，使用脚本测试长头部请求是否能获得响应

**七．浏览器本地缓存静态数据**

修改Nginx配置文件，定义对静态页面的缓存时间

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

2）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

**八．日志切割**

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

[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

**九．对页面进行压缩处理**

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; //小文件不压缩

gzip\_comp\_level 4; //压缩比率

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,在生产中可以直接复制类型

1. **服务器内存缓存**

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;

}

//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

##################################################################################

Proxy\_pass 与rewrite 都是转发都启用有冲突

查看CUP内核数量

lscup

....

CPU(s): 1

.... -p 123456