**6\_Operation03NIGNX调度器优化**

**一 案例1：Nginx反向代理(http代理:7层代理)**

**代理:帮你干活的人或物,拥有调度的功能**

**反向代理功能,实现web服务器的负载均衡,服务器集群高可用;对web服务器有健康检查功能;**在案例中,Nginx服务器不部署web服务,仅作为代理(调度)服务器.

**1.1 问题**

使用Nginx实现Web反向代理功能，实现如下功能：

后端Web服务器两台，可以使用httpd实现

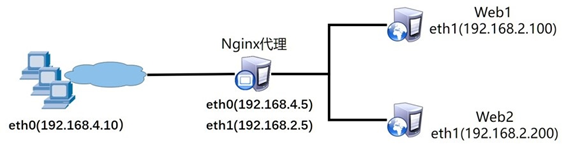
Nginx采用轮询的方式调用后端Web服务器

两台Web服务器的权重要求设置为不同的值

最大失败次数为1，失败超时时间为30秒

**1.2 方案**

使用4台RHEL7虚拟机，其中一台作为Nginx代理服务器，该服务器需要配置两块网卡，IP地址分别为192.168.4.5和192.168.2.5，两台Web服务器IP地址分别为192.168.2.100和192.168.2.200。客户端测试主机IP地址为192.168.4.10。如图所示。



**1.3 步骤**

前置条件nginx能ping通其他3台机器,client不能和web ping通

**1.3.1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。

web1 ~]# yum -y install httpd

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

web1 ~]# systemctl restart httpd

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

web1 ~]# setenforce 0

**2）部署后端Web2服务器**

web2 ~]# yum -y install httpd

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

web2 ~]# systemctl restart httpd

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

web2 ~]# setenforce 0

**1.3.2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**

proxy~]# vim /usr/local/nginx/conf/nginx.conf

**#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 #在server上面写**

**upstream webserver {**

**server 192.168.2.100:80;**

**server 192.168.2.200:80;**

**}**

#gzip on; #定位语句,在此句上面定义集群

server {

listen        80;

server\_name localhost;

location / {

**proxy\_pass <http://webserver;>**

#通过proxy\_pass将用户的请求转发给webserver集群,优先级比root高

}

}

**2）重新加载配置**

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

**3）客户端使用浏览器访问代理服务器测试轮询效果**

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

**1.3.3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。

proxy~]# vim /usr/local/nginx/conf/nginx.conf

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;**

}

#空格分隔

#weight设置服务器权重值，默认值为1,设置该台服务器在服务器集群负载 分担:**本服务器负载分担比例=本服务器权重值/服务器集群重权重值**

#max\_fails设置最大失败次数,若调度该机失败n次,则表示该机不健康

#fail\_timeout设置失败超时时间，单位为**秒**,表示多少秒内不调度该机

#down标记服务器已关机，不参与集群调度

**2）重新加载配置**

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

**3）关闭一台后端服务器**（如web1）

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**

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）**

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**

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

**1.3.4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**

proxy~]# vim /usr/local/nginx/conf/nginx.conf

upstream webserver {

**ip\_hash;** #通过ip\_hash设置调度规则为：相同客户端访问相同服务器

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）重新加载配置**

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

**3）客户端使用浏览器访问代理服务器测试轮询效果**

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

#当某客户端的相同服务器出问题时,该客户机的访问会被调度到其他服务器,当该机的相同服务器恢复后,该客户机的访问会被调度到恢复后的相同服务器.

**二 案例2：Nginx的TCP/UDP调度器(4层调度器)**

**2.1 问题**

使用Nginx实现TCP/UDP调度器功能，实现如下功能：

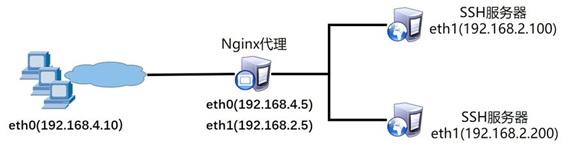
后端SSH服务器两台

Nginx编译安装时需要使用--with-stream，开启ngx\_stream\_core\_module模块

Nginx采用轮询的方式调用后端SSH服务器

**2.2 方案**

使用4台RHEL7虚拟机，其中一台作为Nginx代理服务器，该服务器需要配置两块网卡，IP地址分别为192.168.4.5和192.168.2.5，两台SSH服务器IP地址分别为192.168.2.100和192.168.2.200。客户端测试主机IP地址为192.168.4.10。如图所示。



**2.3 步骤**

**2.3.1部署支持4层TCP/UDP代理的Nginx服务器(4层代理)**

**1）部署nginx服务器**

卸载nginx:killall nginx,rm -rf /usr/local/nginx,重装

编译安装必须要使用--with-stream参数开启4层代理模块。

proxy ~]# yum -y install gcc pcre-devel openssl-devel

proxy ~]# tar -xf nginx-1.12.2.tar.gz

proxy ~]# cd nginx-1.12.2

proxynginx-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

proxynginx-1.12.2]# make && make install #编译并安装

**2.3.2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**

proxy~]# vim /usr/local/nginx/conf/nginx.conf

配置文件中书写在http{ } 之外:

**stream {**

**upstream backend { #定义集群**

**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,同1端口只能1个服务使用**

**proxy\_connect\_timeout 1s; #连接的超时时间，可选配置**

**proxy\_timeout 3s; #连接的超时时间，可选配置**

**proxy\_pass backend; #将请求转发给backend**

**}**

**}**

**2）重新加载配置**

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

**3）客户端使用访问代理服务器测试轮询效果**

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

**三 案例3：Nginx常见问题处理**

**3.1 问题**

本案例要求对Nginx服务器进行适当优化，解决如下问题，以提升服务器的处理性能：

如何自定义返回给客户端的404错误页面

如何查看服务器状态信息

如果客户端访问服务器提示“Too many open files”如何解决

如何解决客户端访问头部信息过长的问题

如何让客户端浏览器缓存数据

日志切割

开启gzip压缩功能，提高数据传输效率

开启文件缓存功能

然后客户机访问此Web服务器验证效果：

使用ab压力测试软件测试并发量

编写测试脚本生成长头部信息的访问请求

客户端访问不存在的页面，测试404错误页面是否重定向

**3.2 步骤**

实现此案例需要按照如下步骤进行。

**3.2.1自定义报错404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**

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**

proxy~]# vim /usr/local/nginx/conf/nginx.conf

charset utf-8; #仅需要中文时需要改选项，可选项

error\_page 404 /404.html; #自定义错误页面(有注释,解除即可)

proxy~]# vim /usr/local/nginx/html/404.html #生成错误页面

Oops,No NO no page …

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

**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

client~]# firefox <http://192.168.4.5/xxxxx> #访问一个不存在的页面

**4）常见http状态码**

状态码 功能描述

200 一切正常

301 永久重定向 #300类代表重定向

302 临时重定向

401 用户名或密码错误 #400类表示用户端错误

403 禁止访问(客户端IP地址被拒绝)

404 文件(页面)不存在

414 请求URI头部过长

500 服务器内部错误 #500类表示服务器端错误

502 bad gateway

**3.2.2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**

proxy~]# tar -zxvf nginx-1.12.2.tar.gz

proxy~]# cd nginx-1.12.2

proxynginx-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

proxynginx-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。**

proxy~]# /usr/local/nginx/sbin/nginx

proxy~]# netstat -anptu | grep nginx

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

proxy~]# ss -anptu | grep nginx

**3）修改Nginx配置文件，定义状态页面**

proxy~]# cat /usr/local/nginx/conf/nginx.conf

#error\_page 404行上面

location /status {

**stub\_status on;**

                 #allow IP地址; #可选项

                 #deny IP地址; #可选项

}

proxy~]# nginx

**4）优化后，查看状态页面信息**

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handled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。

pv= wc -l access.log

uv= awk 正则统计IP数量

**3.2.3优化Nginx并发量**

**1）优化前使用ab高并发测试(没优化前的测试)**

proxy~]# ab -c 2000 -n 2000 <http://192.168.4.5/>

**-c 人数;-n number,总访问量;必须n>=c**

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**

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

}

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

**3）优化Linux内核参数（最大文件数量）**

proxy~]# ulimit -a #查看所有属性值

proxy~]# ulimit -Hn 100000 #设置硬限制（临时规则）20000最大

proxy~]# ulimit -Sn 100000 #设置软限制（临时规则）20000最大

-H:硬件限制;-S:软件限制;-n:numerber,数量

proxy~]# vim **/etc/security/limits.conf**

#该配置文件分4列，分别如下：nofile=number of file

**#用户或组 硬限制或软限制 需要限制的项目 限制的值**

**\* soft nofile 100000**

**\* hard nofile 100000**

**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）

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

**3.2.4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**

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

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**

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 1m;#最大请求包头部信息的缓存个数与容量

.. .. #4个4k,或4个1m

}

proxy~]# /usr/local/nginx/sbin/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**

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

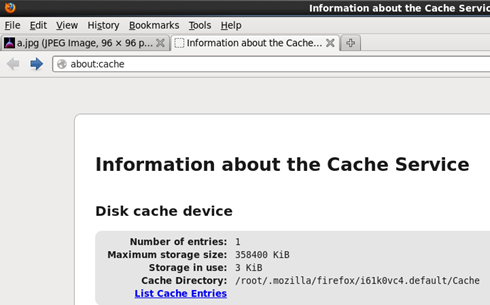
proxy~]# ./buffer.sh

**3.2.5浏览器本地缓存静态数据**

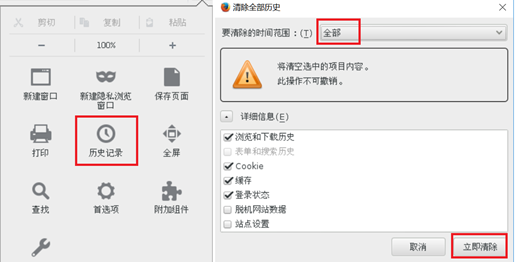
图片,音频,视频等静态资料适合缓存,缓存的时间长短右服务器决定

**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，如图所示。**



**3）修改Nginx配置文件，定义对静态页面的缓存时间**

proxy~]# vim /usr/local/nginx/conf/nginx.conf

server { **在#charset koi8-r;语句上写**

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**

} **#正则表达式:~包含 \*不区分大小写 \转义后面的.号**

}

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

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

**4）优化后，使用Firefox浏览器访问图片，再次查看缓存信息**

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。

**3.2.6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)**

**kill -USR1相当于kill -10, kill -l 查看所有kill的选项**

**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。**

proxy~]# mv access.log access2.log

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。

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

#!/bin/bash

date=`date +%Y%m%d` #定义变量date

logpath=/usr/local/nginx/logs #定义变量logpath

mv $logpath/access.log $logpath/access-$date.log #备份访问日志

mv $logpath/error.log $logpath/error-$date.log #备份错误日志

kill -USR1 $(cat $logpath/nginx.pid) #创建新日志

proxy~]# crontab -e #创建计划任务

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

**3.2.7对页面进行压缩处理**

**1）修改Nginx配置文件**

proxy~]# cat /usr/local/nginx/conf/nginx.conf

http {

**gzip on;** #服务器很重要的优化之一,提高网速 #开启压缩,解除此行的注释

**gzip\_min\_length** 1000; #压缩文件最小长度(大小)1000字节

**gzip\_comp\_level** 4; #压缩比率,范围1-9,数字越大压缩效果越好,速度最慢

**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/usr/local/nginx/conf/mime.types

}

**3.2.8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; #关闭缓存错误报告

#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

}