NSD Operation DAY03

1 案例1：Nginx反向代理

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-1所示。

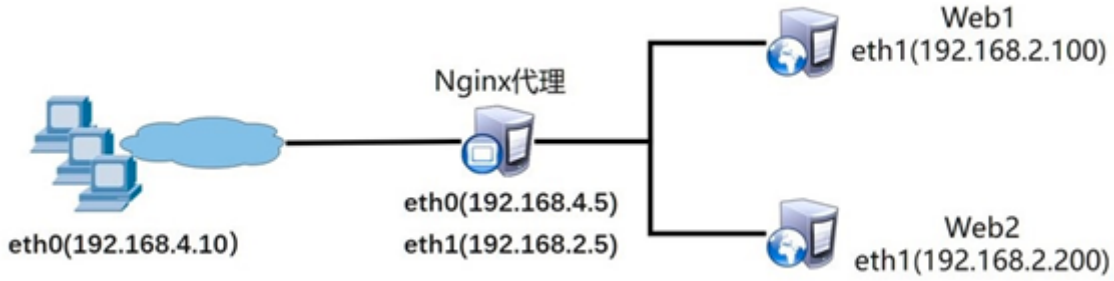

图-1

1.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：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
```

Top

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
```

步骤二：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）重新加载配置

```
[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)
```

Top

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
```

步骤二：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）重新加载配置

```
[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)
```

Top

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
```

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

```
[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;
```

Top

```
        }
    }
```

2）重新加载配置

```
[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
```

2 案例2：Nginx的TCP/UDP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-2所示。

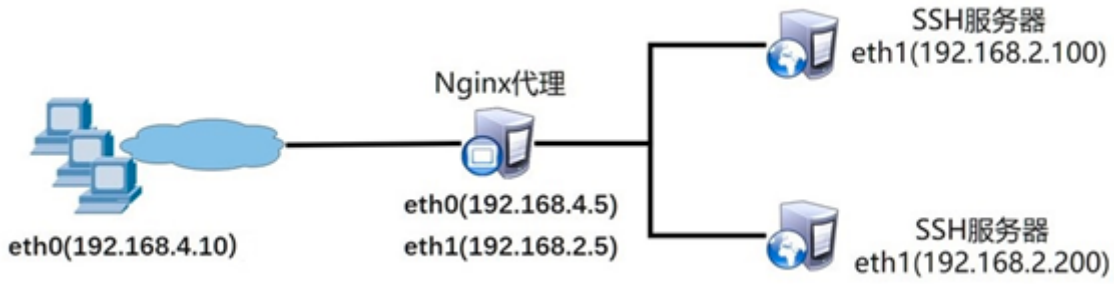

图-2

2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：部署支持4层TCP/UDP代理的Nginx服务器

Top

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
```

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

```
[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;        //连接的超时时间，可选配置
           proxy_timeout 3s;
            proxy_pass backend;
         }
}

http {
.. ..
}
```

2）重新加载配置

```
[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)
```

Top

3）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
```

3 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。

步骤一：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
.. ..
        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
.. ..
[root@proxy ~]# vim /usr/local/nginx/html/404.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)
```

Top

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
```

4）常见http状态码

常见http状态码可用参考表-1所示。

表－1 主机列表

状态码	功能描述
1) 200	2) 一切正常
3) 301	4) 永久重定向
5) 302	6) 临时重定向
7) 401	8) 用户名或密码错误
9) 403	10) 禁止访问（客户端 IP 地址被拒绝）
11) 404	12) 文件不存在
13) 414	14) 请求 URI 头部过长
15) 500	16) 服务器内部错误
17) 502	18) Bad Gateway

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 ~]# 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
```

Top

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
```

Top

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

```
[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
}
.. ..
[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
```

Top

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/
```

步骤四：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
.. ..
}
[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}
```

Top

```
05.     do
06.         URL=${URL}v$i=$i
07.     done
08.     curl $URL
09.     [root@proxy ~]# ./buffer.sh
```

步骤五：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。

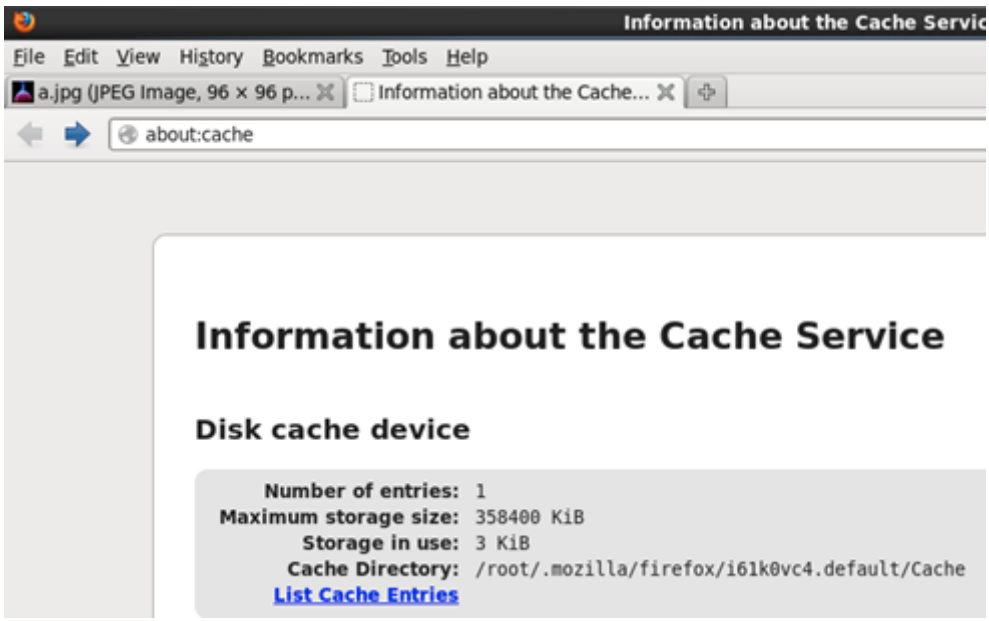

图-3

2）清空firefox本地缓存数据，如图-4所示。

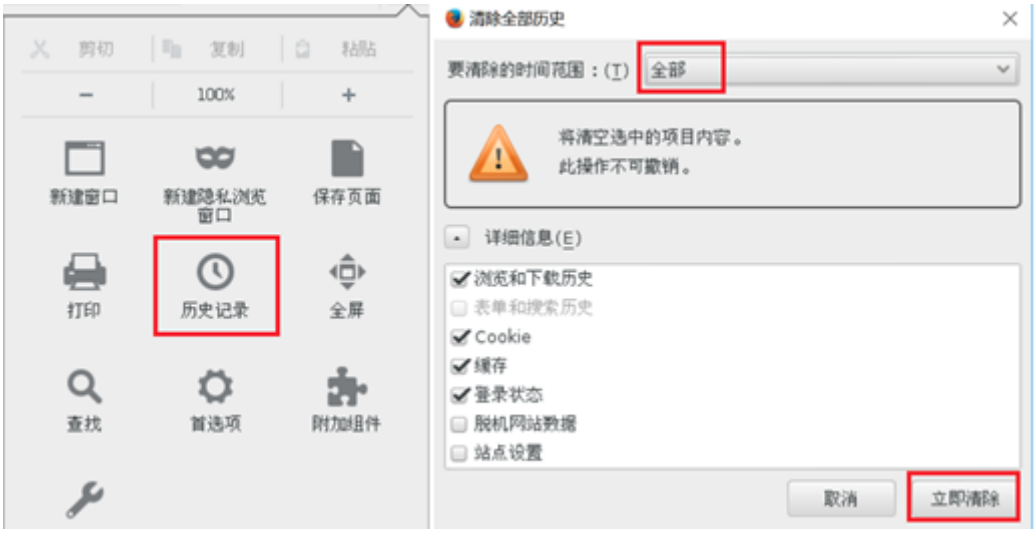

图-4

3）修改Nginx配置文件，定义对静态页面的缓存时间

Top

```
[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。

步骤六：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。

Top

```
[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
```

步骤七：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
gzip_comp_level 4;              //压缩比率
gzip_types text/plain text/css application/json application/x-javascript text/xml applica
                                //对特定文件压缩，类型参考mime.types
.. ..
}
```

步骤八：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;
//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
}
```

Top

Top