[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY03/CASE/01/index.html" \l "page_top_case)

# NSD Operation DAY03

1. [案例1：Nginx反向代理](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY03/CASE/01/index.html" \l "case1)
2. [案例2：Nginx的TCP/UDP调度器](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY03/CASE/01/index.html" \l "case2)
3. [案例3：Nginx常见问题处理](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY03/CASE/01/index.html" \l "case3)

## 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。

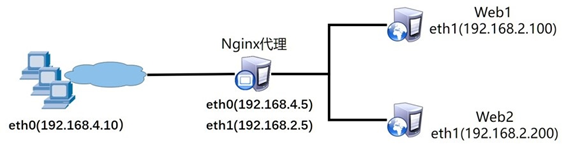


图-1

### 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。

1. [root@web1 ~]# yum -y install httpd
2. [root@web1 ~]# echo "192.168.2.100" > /var/www/html/index.html
3. [root@web1 ~]# systemctl restart httpd

2）部署后端Web2服务器

1. [root@web2 ~]# yum -y install httpd
2. [root@web2 ~]# echo "192.168.2.200" > /var/www/html/index.html
3. [root@web2 ~]# systemctl restart httpd

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

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. .. ..
3. http {
4. .. ..
5. #使用upstream定义后端服务器集群，集群名称任意(如webserver)
6. #使用server定义集群中的具体服务器和端口
7. upstream webserver {
8. server 192.168.2.100:80;
9. server 192.168.2.200:80;
10. }
11. .. ..
12. server {
13. listen        80;
14. server\_name localhost;
15. location / {
16. #通过proxy\_pass将用户的请求转发给webserver集群
17. proxy\_pass http://webserver;
18. }
19. }

2）重新加载配置

1. [root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload
2. #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：
3. #[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

1. [root@client ~]# curl http://192.168.4.5            //使用该命令多次访问查看效果
2. [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。

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. .. ..
3. http {
4. .. ..
5. upstream webserver {
6. server 192.168.2.100 weight=1 max\_fails=1 fail\_timeout=30;
7. server 192.168.2.200 weight=2 max\_fails=2 fail\_timeout=30;
8. server 192.168.2.101 down;
9. }
10. #weight设置服务器权重值，默认值为1
11. #max\_fails设置最大失败次数，测试服务器几次才确认服务器失败
12. #fail\_timeout设置失败超时时间，单位为秒
13. #down标记服务器已关机，不参与集群调度
14. .. ..
15. server {
16. listen        80;
17. server\_name localhost;
18. location / {
19. proxy\_pass http://webserver;
20. }
21. }

2）重新加载配置

1. [root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload
2. #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：
3. #[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）

1. [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

1. [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）

1. [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

1. [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

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. .. ..
3. http {
4. .. ..
5. upstream webserver {
6. #通过ip\_hash设置调度规则为：相同客户端访问相同服务器
7. ip\_hash;
8. server 192.168.2.100 weight=1 max\_fails=2 fail\_timeout=10;
9. server 192.168.2.200 weight=2 max\_fails=2 fail\_timeout=10;
10. }
11. .. ..
12. server {
13. listen        80;
14. server\_name www.tarena.com;
15. location / {
16. proxy\_pass http://webserver;
17. }
18. }

2）重新加载配置

1. [root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload
2. #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：
3. #[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

1. [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。

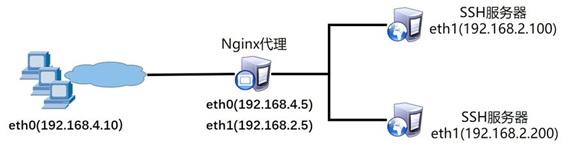


图-2

### 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

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。

1. [root@proxy ~]# yum -y install gcc pcre-devel openssl-devel        //安装依赖包
2. [root@proxy ~]# tar -xf nginx-1.12.2.tar.gz
3. [root@proxy ~]# cd nginx-1.12.2
4. [root@proxy nginx-1.12.2]# ./configure \
5. > --with-http\_ssl\_module        \                 //开启SSL加密功能
6. > --with-stream                                 //开启4层反向代理功能
7. [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

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. stream {
3. upstream backend {
4. server 192.168.2.100:22;            //后端SSH服务器的IP和端口
5. server 192.168.2.200:22;
6. }
7. server {
8. listen 12345;                    //Nginx监听的端口
9. proxy\_pass backend;
10. }
11. }
12. http {
13. .. ..
14. }

2）重新加载配置

1. [root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload
2. #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：
3. #[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

3）客户端使用访问代理服务器测试轮询效果

1. [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

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

1. [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

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. .. ..
3. charset utf-8; //仅在需要中文时修改该选项
4. error\_page 404 /404.html;    //自定义错误页面
5. .. ..
6. [root@proxy ~]# vim /usr/local/nginx/html/404.html        //生成错误页面
7. Oops,No NO no page …
8. [root@proxy ~]# nginx -s reload
9. #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：
10. #[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

1. [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



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

1. [root@proxy ~]# tar -zxvf nginx-1.12.2.tar.gz
2. [root@proxy ~]# cd nginx-1.12.2
3. [root@proxy nginx-1.12.2]# ./configure \
4. > --with-http\_ssl\_module                        //开启SSL加密功能
5. > --with-stream                                //开启TCP/UDP代理模块
6. > --with-http\_stub\_status\_module                //开启status状态页面
7. [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。

1. [root@proxy ~]# /usr/local/nginx/sbin/nginx
2. [root@proxy ~]# netstat -anptu | grep nginx
3. tcp        0        0 0.0.0.0:80        0.0.0.0:\*        LISTEN        10441/nginx
4. [root@proxy ~]# ss -anptu | grep nginx

3）修改Nginx配置文件，定义状态页面

1. [root@proxy ~]# cat /usr/local/nginx/conf/nginx.conf
2. … …
3. location /status {
4. stub\_status on;
5. #allow IP地址;
6. #deny IP地址;
7. }
8. … …
9. [root@proxy ~]# nginx

4）优化后，查看状态页面信息

1. [root@proxy ~]# curl http://192.168.4.5/status
2. Active connections: 1
3. server accepts handled requests
4. 10 10 3
5. 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。

步骤三：优化Nginx并发量

1）优化前使用ab高并发测试

1. [root@proxy ~]# ab -n 2000 -c 2000 http://192.168.4.5/
2. Benchmarking 192.168.4.5 (be patient)
3. 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

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. .. ..
3. worker\_processes 2;                    //与CPU核心数量一致
4. events {
5. worker\_connections 65535;        //每个worker最大并发连接数
6. }
7. .. ..
8. [root@proxy ~]# nginx -s reload

3）优化Linux内核参数（最大文件数量）

1. [root@proxy ~]# ulimit -a                        //查看所有属性值
2. [root@proxy ~]# ulimit -Hn 100000                //设置硬限制（临时规则）
3. [root@proxy ~]# ulimit -Sn 100000                //设置软限制（临时规则）
4. [root@proxy ~]# vim /etc/security/limits.conf
5. .. ..
6. \* soft nofile 100000
7. \* hard nofile 100000
8. #该配置文件分4列，分别如下：
9. #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）

1. [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

1. [root@proxy ~]# cat lnmp\_soft/buffer.sh
2. #!/bin/bash
3. URL=http://192.168.4.5/index.html?
4. for i in {1..5000}
5. do
6. URL=${URL}v$i=$i
7. done
8. curl $URL                                //经过5000次循环后，生成一个长的URL地址栏
9. [root@proxy ~]# ./buffer.sh
10. .. ..
11. <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

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. .. ..
3. http {
4. client\_header\_buffer\_size 1k;        //默认请求包头信息的缓存
5. large\_client\_header\_buffers 4 4k;        //大请求包头部信息的缓存个数与容量
6. .. ..
7. }
8. [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

1. [root@proxy ~]#cat cat buffer.sh
2. #!/bin/bash
3. URL=http://192.168.4.5/index.html?
4. for i in {1..5000}
5. do
6. URL=${URL}v$i=$i
7. done
8. curl $URL
9. [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。

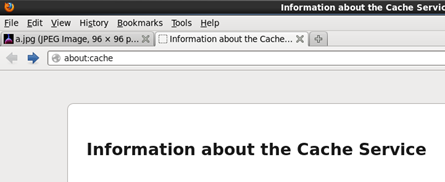


图-3

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

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. server {
3. listen 80;
4. server\_name localhost;
5. location / {
6. root html;
7. index index.html index.htm;
8. }
9. location ~\* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ {
10. expires        30d;            //定义客户端缓存时间为30天
11. }
12. }
13. [root@proxy ~]# cp /usr/share/backgrounds/day.jpg /usr/local/nginx/html
14. [root@proxy ~]# nginx -s reload
15. #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：
16. #[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

1. [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。