

# NSD Operation DAY03

1. 案例1: Nginx反向代理
2. 案例2: Nginx的TCP/UDP调度器
3. 案例3: Nginx常见问题处理

## 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。

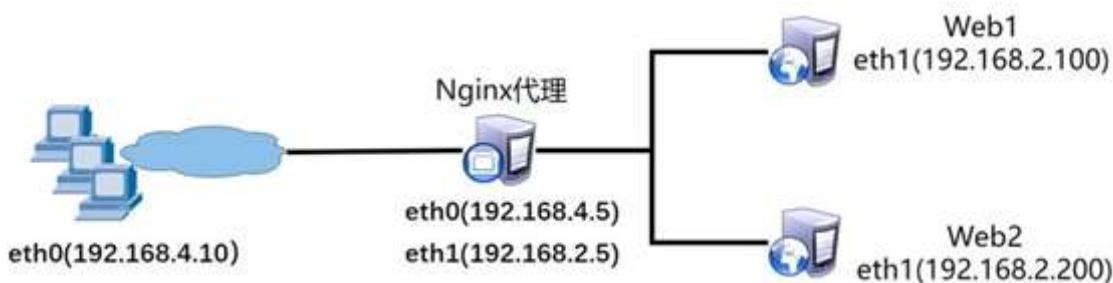


图-1

### 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。

```

01. [root@web1 ~]# yum -y install httpd
02. [root@web1 ~]# echo "192.168.2.100" > /var/www/html/index.html
03. [root@web1 ~]# systemctl restart httpd
04. [root@web1 ~]# firewall-cmd --set-default-zone=trusted
05. [root@web1 ~]# setenforce 0

```

[Top](#)

## 2) 部署后端Web2服务器

```

01. [root@web2 ~]# yum -y install httpd
02. [root@web2 ~]# echo "192.168.2.200" > /var/www/html/index.html
03. [root@web2 ~]# systemctl restart httpd
04. [root@web2 ~]# firewall-cmd --set-default-zone=trusted
05. [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

```

01. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
02. ...
03. http {
04. ...
05. #使用upstream定义后端服务器集群，集群名称任意(如webserver)
06. #使用server定义集群中的具体服务器和端口
07. upstream webserver {
08.         server 192.168.2.100:80;
09.         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

```

01. [root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload
02. #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：
03. #[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or

```

### 3) 客户端使用浏览器访问代理服务器测试轮询效果

```

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

```

## 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。

```

01. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
02. ...
03. http {
04. ...
05. upstream webserver {
06.           server 192.168.2.100 weight=1 max_fails=1 fail_timeout=30;
07.           server 192.168.2.200 weight=2 max_fails=2 fail_timeout=30;
08.           server 192.168.2.101 down;
09.       }
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
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

```

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

```

### 3) 关闭一台后端服务器 (如web1)

```
01. [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

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

### 5) 再次启动后端服务器的httpd (如web1)

```
01. [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

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

## 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

```

01. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
02. ...
03. http {
04. ...
05. upstream webserver {
06. #通过ip_hash设置调度规则为：相同客户端访问相同服务器
07.         ip_hash;
08.         server 192.168.2.100 weight=1 max_fails=2 fail_timeout=10;
09.         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;

```

[Top](#)

17.      }

18.      }

## 2) 重新加载配置

```
01. [root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload
02. #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：
03. #[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or
```

## 3) 客户端使用浏览器访问代理服务器测试轮询效果

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

# 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。

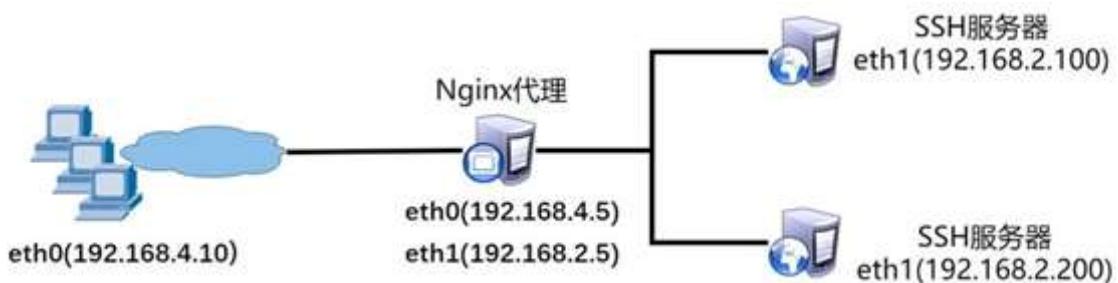


图-2

## 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。

```

01. [root@proxy ~]# yum -y install gcc pcre-devel openssl-devel //安装依赖包
02. [root@proxy ~]# tar -xf nginx-1.12.2.tar.gz
03. [root@proxy ~]# cd nginx-1.12.2
04. [root@proxy nginx-1.12.2]# ./configure \
05. > --with-http_ssl_module //开启SSL加密功能
06. > --with-stream //开启4层反向代理功能
07. [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

```

01. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
02.
03. stream {
04.     upstream backend {
05.         server 192.168.2.100:22; //后端SSH服务器的IP和端口
06.         server 192.168.2.200:22;
07.     }
08.     server {
09.         listen 12345; //Nginx监听的端口
10.         proxy_connect_timeout 1s; //连接的超时时间，可选配置
11.         proxy_timeout 3s;
12.         proxy_pass backend;
13.     }
14. }
15.
16. http {
17.     ...
18. }

```

### 2) 重新加载配置

```

01. [root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload
02. #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：
03. #[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or

```

### 3) 客户端使用访问代理服务器测试轮询效果

```
01. [root@client ~]# ssh 192.168.4.5 -p 12345 //使用该命令多次访问查
```

## 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

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

2) 修改Nginx配置文件，自定义报错页面

```
01. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
02. ...
03.         charset utf-8; //仅需要中文时需要改选项，可选项
04.         error_page 404 /404.html; //自定义错误页面
05. ...
06. [root@proxy ~]# vim /usr/local/nginx/html/404.html //生成错误页面
07. Oops, No NO no page ...
08.
09. [root@proxy ~]# nginx -s reload
```

[Top](#)

10. #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：
11. #[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

01. [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
200	一切正常
301	永久重定向
302	临时重定向
401	用户名或密码错误
403	禁止访问（客户端IP地址被拒绝）
404	文件不存在
414	请求URI头部过长
500	服务器内部错误
502	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

01. [root@proxy ~]# tar -zvxf nginx-1.12.2.tar.gz
02. [root@proxy ~]# cd nginx-1.12.2
03. [root@proxy nginx-1.12.2]# ./configure \
04. > --with-http\_ssl\_module //开启SSL加密功能
05. > --with-stream //开启TCP/UDP代理模块
06. > --with-http\_stub\_status\_module //开启status状态页面
07. [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

[Top](#)

-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。

```

01. [root@proxy ~]# /usr/local/nginx/sbin/nginx
02. [root@proxy ~]# netstat -anptu | grep nginx
03.      tcp      0      0 0.0.0.0:80      0.0.0.0:*      LISTEN      1044
04. [root@proxy ~]# ss -anptu | grep nginx

```

### 3) 修改Nginx配置文件，定义状态页面

```

01. [root@proxy ~]# cat /usr/local/nginx/conf/nginx.conf
02. ...
03. location /status {
04.     stub_status on;
05.     #allow IP地址;
06.     #deny IP地址;
07. }
08. ...
09. [root@proxy ~]# nginx

```

### 4) 优化后，查看状态页面信息

```

01. [root@proxy ~]# curl http://192.168.4.5/status
02. Active connections: 1
03. server accepts handled requests
04. 10 10 3
05. 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。

[Top](#)

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

```

01. [root@proxy ~]# ab -n 2000 -c 2000 http://192.168.4.5/
02. Benchmarking 192.168.4.5 (be patient)
03. 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

```

01. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
02. ...
03. worker_processes 2;                      //与CPU核心数量一致
04. events {
05.   worker_connections 65535;           //每个worker最大并发连接数
06. }
07. ...
08. [root@proxy ~]# nginx -s reload

```

### 3) 优化Linux内核参数（最大文件数量）

```

01. [root@proxy ~]# ulimit -a                  //查看所有属性值
02. [root@proxy ~]# ulimit -Hn 100000          //设置硬限制（临时规则）
03. [root@proxy ~]# ulimit -Sn 100000          //设置软限制（临时规则）
04. [root@proxy ~]# vim /etc/security/limits.conf
05. ...
06. *           soft    nofile      100000
07. *           hard    nofile      100000
08.
09. #该配置文件分4列，分别如下：
10. #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）

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

[Top](#)

## 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

```

01. [root@proxy ~]# cat lncmp_soft/buffer.sh
02. #!/bin/bash
03. URL=http://192.168.4.5/index.html?
04. for i in {1..5000}
05. do
06.     URL=${URL}v${i}
07. done
08. curl $URL //经过5000次循环后, 生成一个长的URL
09. [root@proxy ~]# ./buffer.sh
10. ...
11. <center><h1>414 Request-URI Too Large</h1></center> //提示头部信息过

```

### 2) 修改Nginx配置文件, 增加数据包头部缓存大小

```

01. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
02. ...
03. http {
04.     client_header_buffer_size 1k; //默认请求包头信息的缓存
05.     large_client_header_buffers 4 4k; //大请求包头部信息的缓存个数与容量
06. }
07. [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

```

01. [root@proxy ~]#cat cat buffer.sh
02. #!/bin/bash
03. URL=http://192.168.4.5/index.html?
04. for i in {1..5000}
05. do
06.     URL=${URL}v${i}
07. done
08. curl $URL
09. [root@proxy ~]# ./buffer.sh

```

[Top](#)

## 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

```

01. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
02. server {
03.     listen      80;
04.     server_name localhost;
05.     location / {
06.         root    html;
07.         index  index.html index.htm;

```

[Top](#)

```

08. }
09. 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

```

#### 4) 优化后，使用Firefox浏览器访问图片，再次查看缓存信息

```
01. [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。

```

01. [root@proxy ~]# mv access.log access2.log
02. [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。

```

01. [root@proxy ~]# vim /usr/local/nginx/logbak.sh
02. #!/bin/bash
03. date=`date +%Y%m%d`
04. logpath=/usr/local/nginx/logs
05. mv $logpath/access.log $logpath/access-$date.log
06. mv $logpath/error.log $logpath/error-$date.log
07. kill -USR1 $(cat $logpath/nginx.pid)
08.

```

[Top](#)

```
09. [root@proxy ~]# crontab -e
10. 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

```
01. [root@proxy ~]# cat /usr/local/nginx/conf/nginx.conf
02. http {
03. ...
04.     gzip on; //开启压缩
05.     gzip_min_length 1000; //小文件不压缩
06.     gzip_comp_level 4; //压缩比率
07.     gzip_types text/plain text/css application/json application/x-javascript text/javascript;
08.                                         //对特定文件压缩，类型参考mime.types
09. ...
10. }
```

## 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。

```
01. http {
02.     open_file_cache           max=2000    inactive=20s;
03.             open_file_cache_valid   60s;
04.             open_file_cache_min_uses 5;
05.             open_file_cache_errors  off;
06.         //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
07.         //文件句柄的有效时间是60秒，60秒后过期
08.         //只有访问次数超过5次会被缓存
09. }
```

[Top](#)