

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192.168.4.100。如图-1所示。

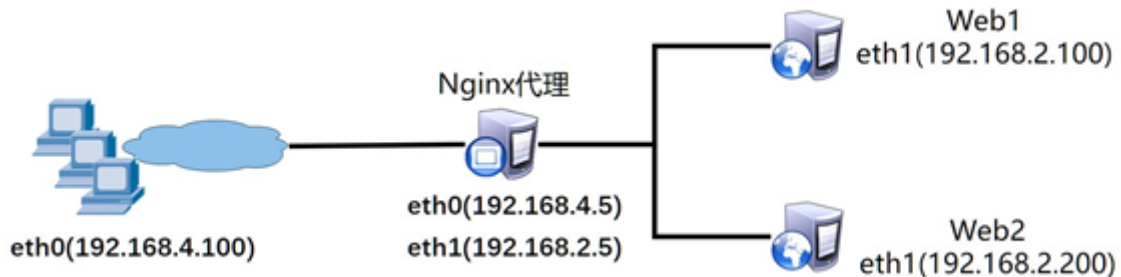


图-1

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
```

[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
```

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.     upstream webserver {
06.         server 192.168.2.100:80;
07.         server 192.168.2.200:80;
08.     }
09.     ... ..
10.     server {
11.         listen      80;
12.         server_name localhost;
13.         location / {
14.             proxy_pass http://webserver;
15.         }
16.     }
```

2) 重启nginx服务

```
01. [root@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

```
01. [root@client ~]# curl http://192.168.4.5 //使用该命令多
Top
```

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_t
07.              server 192.168.2.200 weight=2 max_fails=2 fail_t
08.  }
09.  //weight设置服务器权重值
10.  //max_fails设置最大失败次数
11.  //fail_timeout设置失败超时时间, 单位为秒
12.  .. ..
13.  server {
14.      listen      80;
15.      server_name  localhost;
16.      location / {
17.          proxy_pass http://webserver;
18.      }
19.  }

```

2) 重启nginx服务

```

01.  [root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

```

3) 关闭一台后端服务器 (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 //使用这个命令多

```

5) 再次启动后端服务器的httpd (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
```

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.             ip_hash;
07.             server 192.168.2.100 weight=1 max_fails=2 fail_t
08.             server 192.168.2.200 weight=2 max_fails=2 fail_t
09.         }
10. ... ..
11. server {
12.     listen      80;
13.     server_name  www.tarena.com;
14.     location / {
15.         proxy_pass http://webserver;
16.     }
17. }
```

2) 重启nginx服务

```
01. [root@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

[Top](#)

```
01. [root@client ~]# curl http://192.168.4.5
```

//使用该命令多

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192.168.4.100。如图-2所示。

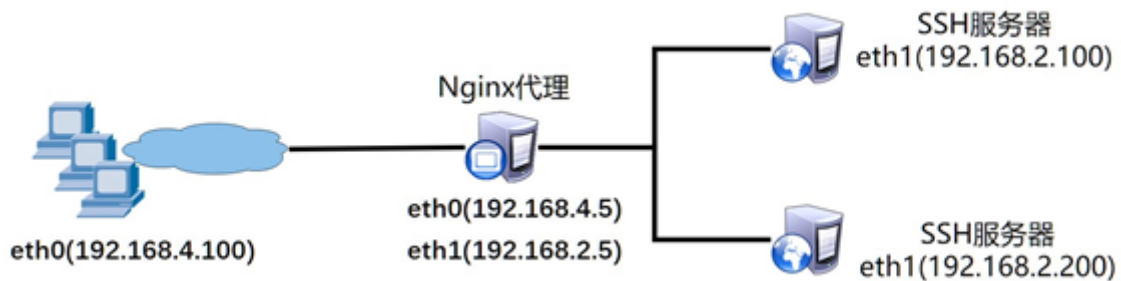


图-2

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。

```

01. [root@proxy ~]# yum -y install gcc pcre-devel openssl-devel
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
06. > --with-stream //开启4层代理
07. [root@proxy nginx-1.12.2]# make && make install //编译
  
```

[Top](#)

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
06.             server 192.168.2.200:22;
07.         }
08.         server {
09.             listen 12345;                       //Nginx监听的端口
10.             proxy_connect_timeout 1s;
11.             proxy_timeout 3s;
12.             proxy_pass backend;
13.         }
14.     }
15.
16.     http {
17.         ... ..
18.     }
```

2) 重启nginx服务

```
01. [root@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

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

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
- 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

[Top](#)

- 开启gzip压缩功能，提高数据传输效率
-

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：构建Nginx服务器

1) 源码安装Nginx软件

```
01. [root@proxy ~]# yum -y install gcc pcre-devel openssl-devel
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S
06. > --with-stream //开启TCP
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

```
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
```

步骤二：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
```

2) 修改Nginx配置文件，增加并发量

[Top](#)

```
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.  use epoll;
07.  }
08.  ... ..
09.  [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）
03.  [root@proxy ~]# ulimit -Sn 100000 //设置软限制（临时）
04.  [root@proxy ~]# vim /etc/security/limits.conf
05.  ... ..
06.  * soft nofile 100000
07.  * hard nofile 100000

```

4) 优化后测试服务器并发量

```

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

```

步骤三：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 lnmp_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=$i
07.  done
08.  curl $URL //经过5000次循环后，生成一
09.  [root@proxy ~]# ./buffer.sh
10.  ... ..
11.  <center><h1>414 Request-URI Too Large</h1></center> //提示

```

[Top](#)

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;         //大请求包头信息的缓存个数
06.     ... ..
07. }
08. [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=$i
07. done
08. curl $URL
09. [root@proxy ~]# ./buffer.sh
```

步骤四：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。

[Top](#)



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;
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.     }
```

[Top](#)

```

12.     }
13.     [root@proxy ~]# cp /usr/share/backgrounds/day.jpg /usr/local/nginx
14.     [root@proxy ~]# 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

```
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。

步骤五：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/xxxxxx //访问一
```

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.     error_page 404 /40x.html; //自定义错误页面
04.     .. ..
05.     [root@proxy ~]# vim /usr/local/nginx/html/40x.html //生成
06.     Oops, No NO no page ...
07.
08.     [root@proxy ~]# 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

```
01.     [root@client ~]# firefox http://192.168.4.5/xxxxxx //访问一
```

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

[Top](#)

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

步骤六：如何查看服务器状态信息

1) 编译安装时使用--with-http_stub_status_module开启状态页面模块

```

01. [root@proxy ~]# yum -y install gcc pcre-devel openssl-devel
02. [root@proxy ~]# tar -zxvf 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TCP/UDP代理模块
07. > --with-http_stub_status_module //开启status状态页面
08. [root@proxy nginx-1.12.2]# make && make install //编译并安装

```

2) 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.         }
06. ... ..
07. [root@proxy ~]# nginx

```

2) 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

```

[Top](#)

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。

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;
08.                                           //对特定文件压缩，类型参考mime.types
09.     ...
10. }
```

[Top](#)