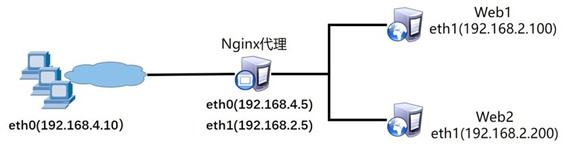
**Morning：**

1. Nginx的HTTP调度



-Nginx代理服务器：屏蔽web服务器信息、实现负载均衡、动态健康检查

**1）部署实施后端Web服务器**

Web1：192.168.2.100（Apache）

Web2：192.168.2.200（Apache）

**2）配置Nginx服务器，添加服务器池，实现HTTP反向代理功能（轮询）**

[root@proxy ~]#vim usr/local/nginx/conf/nginx.conf

http{ #先把所有rewrite注释掉

upstream webs { #定义后端服务器集群，名字自定义

server 192.168.2.100:80 weight=1 max\_fails=1 fail\_timeout=30;

server 192.168.2.200:80 weight=2 max\_fails=2 fail\_timeout=30; server 192.168.2.101 down;

#server定义集群的具体服务器:端口，不写默认为80

#weight设置服务器权重值，默认值为1（让性能优的服务器权重大）

#max\_fails设置最大失败次数（与ping -c2同理）

#fail\_timeout设置失败超时时间，单位为秒（失败后xx秒再测一下）

#down标记服务器已关机，不参与集群调度

}

server {

...

location / {

proxy\_pass http://webs;

#通过proxy\_pass将用户的请求转发给webs集群

}

}

}

[root@proxy ~]# nginx -s reload #重启nginx服务

[root@client ~]# curl http://192.168.4.5 #客户端测试轮询效果

1. **配置hash调度算法**

hash调度算法：让相同客户端访问相同服务器

ip(1.1.1.1) | md5sum --> 16进制数字%n(结果0..n-1)

（hash只对IP的前三个8位字节作hash密钥）

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

http{

upstream webs {

ip\_hash; #设置调度规则为：相同客户端访问相同服务器

server ... ;

}

server {

...

location / {

proxy\_pass http://webs;

}

}

}

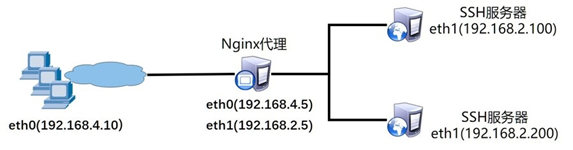
[root@proxy ~]# nginx -s reload #重启nginx服务

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

#客户端测试：相同客户端访问相同服务器

**Afternoon：**

1. Nginx的TCP/UDP调度



ftp：20+21+随机 Samba：137+138+139（多端口upstream，暂时不会）

**1）部署支持4层TCP/UDP代理的Nginx服务器**

[root@proxy ~]# rm -rf /usr/local/nginx/ #卸载nginx

... ...

[root@proxy nginx-1.12.2]# ./configure \

> --with-http\_ssl\_module

> --with-stream #开启4层反向代理功能

[root@proxy nginx-1.12.2]# make && make install #重装

**2）配置Nginx服务器，添加服务器池，实现TCP/UDP反向代理功能**

[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（自定义）

proxy\_connect\_timeout 1s;

proxy\_timeout 3s; #允许连接时间

proxy\_pass backend; #转发upstream后端服务器

}

}

http { ... ... }

[root@proxy ~]# nginx -s reload #重启服务

[root@client ~]# ssh 192.168.4.5 -p 12345

#客户端测试远程轮询效果

1. Nginx常见优化

**-常见问题**

·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

**-常见返回码**



2开头：正常访问 3开头：重定向 4开头：访问错误

5开头：服务器错误

**优化1：自定义报错页面（支持中文）**

http {

...

server {

error\_page 404 /40x.html; #自定义错误页面

charset utf-8; #支持万国编码utf-8

}

}

**优化2：查看服务器状态信息（非常重要的功能）**

1）重装Nginx

[root@proxy ~]# ./configure \

> --with-http\_ssl\_module \

> --with-stream \

> --with-http\_stub\_status\_module #status状态页面模块

1. 启动Nginx服务并查看监听端口状态

[root@proxy ~]# /usr/local/nginx/sbin/nginx #启动nginx服务

[root@proxy ~]# ss -anptu | grep nginx

#查看端口信息，与netstat功能一样

3）修改Nginx配置文件，定义状态页面

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

server {

location /status { #自定义地址栏

stub\_status on; #显示状态信息

allow 127.0.0.1; #允许IP

allow 192.168.4.0/24; #允许网段

deny all; #拒绝所有

}

}

[root@proxy ~]# nginx #启动nginx服务

4）优化后，查看状态页面信息

[root@proxy ~]# curl http://192.168.4.5/status

Active connections: 1 #实时并发量

server accepts handled requests

10 10 33

#历史接受客户端连接、处理客户端连接次数、客户端发送请求次数

Reading: 0 Writing: 1 Waiting: 0

#实时读取客户端请求数量、服务器写响应信息数量、客户端等待响应数量

Tips：http1.0 一次连接一次请求

http1.1 一次连接多次请求

keepalive\_timeout 65 #[配置]用户空闲65s强制断开

**优化3：Nginx并发量**

1）优化前使用ab高并发测试

[root@proxy ~]# yum -y install httpd-tools

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

#-c2000个client，-n点击2000次

1. 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最大并发量

use epoll；

}

#实验室Nginx最大并发量50000;Apache最大30000

3）优化Linux内核参数（单个进程最大文件数量）

[root@proxy ~]# ulimit -a #查看所有属性值

[root@proxy ~]# ulimit -Hn 100000 #设置硬限制标准（临时规则）

[root@proxy ~]# ulimit -Sn 100000 #设置软限制警告（临时规则）

[root@proxy ~]# vim /etc/security/limits.conf

.. ..

\* soft nofile 100000

\* hard nofile 100000

#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（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/>

**优化4：优化Nginx数据包头缓存**

（实验前，先把upstream的内容注释掉，做单机实验）

1）测试地址栏头部过长是否能获得响应

[root@proxy ~]# ./lnmp\_soft/buffer.sh #5000次循环的URL

414 Request-URI Too Large #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，共16k（经验值）

}

[root@proxy ~]# nginx -s reload

3）优化后，测试长头部请求是否能获得响应

[root@proxy ~]# ./lnmp\_soft/buffer.sh

#本实验设large\_client\_header\_buffers 4 1m;

Tips：网页www.jd.com/login?user=tom,passwd=asDI#jasdfl!

?后面为用户请求的参数，用于服务端匹配数据库

**优化5：优化浏览器本地缓存**

缓存的概念（相对）：硬盘 ---> 内存 ---> L1,L2(CPU)

1）firefox查看缓存：about:cache

2）清空firefox本地缓存数据：菜单--历史记录--清除历史记录

3）修改Nginx配置文件，定义对静态页面的缓存时间

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

server {

...

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 html]# cp /usr/share/backgrounds/day.jpg .

[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

[root@client ~]# firefox http://192.168.4.5/day.jpg

#输入about:cache，查看本地缓存信息、存放位置、过期时间等

**优化6：日志切割（非常常见的面试题）**

问题：日志文件越来越大怎么办？单个文件10G? 如何切割？

1. 手动执行：

[root@proxy ~]# mv access.log access2.log #旧日志重命名

[root@proxy ~]# kill -USR1 $(cat /usr/local/nginx/logs/nginx.pid)

#通知Nginx日志改名了，需要新建一个日志

#kill 给程序发命令，-l查看所有，默认-15结束， -9杀死

1. 自动完成：

[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

#日志重命名备份脚本+计划任务，每周五03：03完成日志切割

**优化7：对页面进行压缩处理**

1）修改Nginx配置文件

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

http {

...

gzip on; #开启压缩

gzip\_min\_length 1000; #小文件不压缩(字节)

gzip\_comp\_level 4;

#压缩比率(1-9)，越高效果越好，耗时越长

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mime.types（类型 扩展名）

...

}

Tips： -所有浏览器都支持gzip解压

-小文件不能压缩，越压越大

-多媒体文件不要压缩。mp4/mp3已是压缩格式；且多媒体文件一般比较大，压缩时间长。

**优化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。

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

http {

open\_file\_cache max=2000 inactive=20s;

open\_file\_cache\_valid 60s;

open\_file\_cache\_min\_uses 5;

open\_file\_cache\_errors off; #关闭缓存报错（过期）

#设置服务器最大缓存2000个文件，关闭20秒内无请求的文件

#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（热点数据）

}