官网http://www.nginx.org

sed显示行号

]# sed -n '/baseurl/=' epel.repo

3

12

21

ss -antupl

a显示 所有端口

n数字格式显示端口

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

p显示监听端口的服务（程序）名称

++++++++++++++++++++++++++++++++++++++++++++++++

**三次握手和四次断开**

一共发了7个包

握手其实也是4个,只不过其中把两个包合成一个包发了

三次握手

1.建立连接之前,server处于listen监听端口状态,这时client向server发送一个SYN报文请求连接,(发完SYN报文后)client进入**SYN\_SENT**状态

2.server收到cSYN报文后,回复ACK确认收到,因为双向连接(双工),server也向client发送SYN建立连接,把SYN和ACK合成一个报文SYN+ACK发送给clinet,server进入**SYN\_RECV**状态(等待客户端的确认)

3.client收到(receive)了SYN+ACK报文后,就进入了**ESTABLISHED**(此刻是半连接状态),再回发ACK给server,server收到ACK报文后确认连接后进入**ESTABLISHED**(达到真正的双向连接)

(半连接是一方只给别一方发送数据)

四次断开

1.client发起FIN报文给server请求断开连接,后进入**FIN\_WAIT1**状态.server收到FIN报文后,回复一个ACK报文确认收到,进入**CLOSE\_WAIT**(等待关闭)

2.clietn收到severACK报文进入**了FIN\_WAIT2**状态,但因为双向连接(双工)server此时不一定把处理的数据全发完,可能还在发送数据给client

3.当server完全发送完数据后,无后续操作时,就向client发送FIN报文,server进**入LAST\_ACK**状态(最后一个ACK状态)

4.client收到server的FIN报文后,就向server发送ACK报文确定收到,clinet进入**TIME\_WAIT**状态(等待2毫秒后)再进入**CLOSED**关闭连接;server收到client的ack确定信息后进了**CLOSED**关闭连接.

这个难点

服务器主动向客户端请求断开连接(两边同时发FIN)

C FIN-> 和<-FIN S

FIN-WAIT1

两端同时发送ACK后,同时进入CLOSING(也等ACK确认),同时进入TIME-WAIT后再断开连接

重点

**SYN-RECEIVED多,证明客户端的响应非常慢,半接连攻击是这个状态多**

tcp的三次握手四次断开与状态

1. 客户端独有的：(1)SYN\_SENT (2)FIN\_WAIT1 (3)FIN\_WAIT2 (4)CLOSING (5)TIME\_WAIT

2. 服务器独有的：(1)LISTEN (2)SYN\_RCVD (3)CLOSE\_WAIT (4)LAST\_ACK

3. 共有的： (1)CLOSED (2)ESTABLISHED

LISTEN - 侦听来自远方TCP端口的连接请求；

SYN-SENT - 在发送连接请求后等待匹配的连接请求；

SYN-RECEIVED - 在收到和发送一个连接请求后等待对连接请求的确认；

ESTABLISHED - 代表一个打开的连接，数据可以传送给用户；

FIN-WAIT-1 - 等待远程TCP的连接中断请求，或先前的连接中断请求的确认；

FIN-WAIT-2 - 从远程TCP等待连接中断请求；

CLOSE-WAIT - 等待从本地用户发来的连接中断请求;

LAST-ACK - 等待原来发向远程TCP的连接中断请求的确认；

TIME-WAIT - 等待足够的时间以确保远程TCP接收到连接中断请求的确认；

CLOSED - 没有任何连接状态；

CLOSING - 同时发起关闭状态，从 FIN-WAIT-1 --> CLOSING --> TIME-WAIT

++++++++++++++++++++++++

**正向代理和反向代理的区别?**

**概念不同**

正向代理：（forward proxy） ，一个位于客户端和被客户端请求访问的目标服务器之间的服务器，为了从目标服务器获取内容，客户端向代理发送一个请求并制定目标（目标服务器），然后代理向原始服务器转发请求并将获得的内容返回给客户端，客户端才能使用正向代理。我们平时说的代理就是指正向代理。（代理对象客户端）

反向代理：（Reverse Proxy），以代理服务器来接受internet上的客户端连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求的客户端，此时代理服务器对外表现为一个反向代理服务器。（代理对像服务端）

**位置不同**

正向代理，架设在客户机和目标主机之间； （a访c,通过正向代理服务器b去访问c，c不知道被a访）

反向代理，架设在服务器端； （a访c，c没有a要求访问的服务，c可调度c内部网络（c防火墙后的其它有该要求的服务主机）为a提供服务））

**代理对象不同**

正向代理，代理客户端，服务端不知道实际发起请求的客户端；

反向代理，代理服务端，客户端不知道实际提供服务的服务端；

备注：正向代理–HTTP代理为多个人提供翻墙服务；反向代理–百度外卖为多个商户提供平台给某个用户提供外卖服务。

**用途不同**

正向代理，为在防火墙内的局域网客户端提供访问Internet的途径；

反向代理，将防火墙后面的服务器提供给Internet访问；

**安全性不同**

正向代理允许客户端通过它访问任意网站并且隐藏客户端自身，因此必须采取安全措施以确保仅为授权的客户端提供服务；

反向代理都对外都是透明的，访问者并不知道自己访问的是哪一个代理。

**应用不同**

正向代理的应用

    1. 访问原来无法访问的资源

    2. 用作缓存，加速访问速度

    3. 对客户端访问授权，上网进行认证

    4. (安全)代理可以记录用户访问记录（上网行为管理），对外隐藏用户信息

反向代理的应用

1.保护内网安全

    2. 负载均衡

3. 缓存，减少服务器的压力

4.高可用

**nginx web服务知识总结**

**为什么选择nginx**

nginx三大功能(国内使用的主要场景)

web服务(主要):支持高性能,高并发web服务软件,与apache(传统select模型)相比,nginx(epoll linux2.6内核)能够支持更多的并发连接访问,但占用的资源却更少(3万并发连接下,开启10个nginx线程消耗的内存不到200MB),效率更高.

反向代或负载均衡服务(主要):作为web服务,php等动态服务及Mencache缓存,TCP的代理服务器,同时也是一款优秀的(最早)邮件代理服务器.但nginx的代理功能相对还是简单些.

前端业务数据缓存服力:在web缓存方面,nginx能过自身的proxy\_cache模块实现类Squid等专业缓存功能.

为什为nginx总体性能比apache高

nginx使用最新的epoll(linux2.6内核)和kqueue(freebsd)异步网络I/O模型,而apache使用的是传统select模型.apache的性能随着连接数的增加,性能会急剧下降,处理成千上万的并发连接数时,性能很差,而nginx的性能基本没有下降,性能稳定,因为nginx支持更多的并发连接访问,但占用的资源却更少3万并发连接下,开启10个nginx线程消耗的内存不到200MB

(select模型的宿舍大妈,找人就和你在宿舍一个房间房间找;epoll模型的宿舍大妈,会先记下人的房间号,你来找人时直接告诉你找的人的房间号)

select epoll

性能 成千上万差 稳定

连接数 1024 65535(理想值,实60000)

机制 线性轮询 回调callback

如何正确选择web服务器

一.静态业务:若是高并发场景,尽量采用nginx或lighttpd,二者首选nginx.

二.动态业务:理论上采用nginx和apache均可,建议选择nginx,为了避免相同业务的服务软件多样化,增加额外维护成本.动态业务可由nginx兼做前端代理,再根据页面元素类型或目录,转发到后端相应的服务器进行处理.

三.既有静态业务又有动态业务,采用nginx

编译安装NGINX

pcre:"perl兼容正则表达式",安装pcre库是为了使nginx支持具备URI重写功能rewrite模块(几乎企业必备的)

#通过ip\_hash设置调度规则为：相同客户端访问相同服务器(但ip地址不同)

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;

https://jingyan.baidu.com/article/4853e1e543f1561908f7267c.html

https://www.awaimai.com/371.html

FastCGI工作原理(是一种协议)

1 Web Server启动时载入FastCGI进程管理器

2 FastCGI进程管理器初始化，启动多个CGI解释器进程

3 当客户端请求到达web server时，FastCGI进程管器选择并连接到一个CGI解释器

4 FastCGI子进程 完成处理后返回结果，将标准输出和错误信息从同一连接返回web server

缺点:内存消耗大(因为其是多进程)

查看nginx服务器状态信息(用--with-http\_stub\_status\_module)

location /status { 注意这里的/不能有空格

stub\_status on;

}

curl http://192.168.4.5/status

修改/usr/local/nginx/conf/nginx.conf后,都需要重新加载,使配置生效.这是因为nginx启动时已把所有配置信息都加载到内存中,这样设计的目的是大幅度提升nginx服务的访问性能.(提高访问性能)

+++++++++++++++++++++++++++++++++++++++++++++++

**nginx的优化**

一.并发量高时发生too many open files问题

1.增加并发量

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

use epoll;

2.优化linux内核参数(最大文件数量)

[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

3.优化后测试服务器并发量

]# ab -n 2000 -c 2000 <http://192.168.4.5/> (ab命令要装httpd-tools)

(-c并发数(连接人) -n总请求数(连接数))

二.优化nginx数据头缓存

~]# 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; //(最大4 1M也足够了)大请求包头部信息的缓存个数与容量

三.修改nginx配置文件静态页面的缓存时间(~代表正则表达式,\*代表) 保存到客户端让客户端访问更快(对像客户端)

location ~\* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ {

expires 30d; //定义客户端缓存时间为30天

四.定义自定义报错页面(优化用户体验效果)

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

.. ..

error\_page 404 /40x.html; //自定义错误页面

.. ..

[root@proxy ~]# vim /usr/local/nginx/html/40x.html //生成错误页面

Oops,No NO no page …

常见的http状态码

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客户端地址被拒绝访问

404文件不存在

414请求URI头部过长(数据头缓存)

500服务器内部错误

502Bad Gateway

五.对页面进行压缩处理(节省存储空间,浏览器自带解压功能,gzip格式,提高传输效率)

]# cat /usr/local/nginx/conf/nginx.conf

http {

.. ..

gzip on; //开启压缩

gzip\_min\_length 1000; //小文件不压缩(不能小于1M)

gzip\_comp\_level 4; //压缩比率(范围1到9)

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

(小文件越压越大,起码1M或以上压才有意义,一般对文档类型压缩,多媒体文件不能压缩:jpg,flv,mp3)

六.服务器内存缓存(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)

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

Nginx status (--with-http\_stub\_status\_module)

Server{

listen 80;

server\_name localhost;

location / {

stub\_status on;

allow 10.0.0.0/24;

deny all;

}

}

Server

**location匹配**

location [=|~|~\*|^~] 四种

= 精确僻配

~和~\*两种特殊字符的区别

~ 匹配大小写

~\* 不匹配大小写

(同时也可以在前面加上!对上面的匹配取反)

^~ 匹配常规字符串后,不做正则表达式的匹配

location = / {

[configuration A]

} 对应用户URI输入https://www.qq.com/

(= 精确匹配优先级最高,无论放置顺序如何,最先匹配)

location / {

[configuration B]

} 对应用户URI输入https://www.qq.com/index.html

(/ 默认匹配,如没有匹配上其他的location,最后执行默认匹配)

location /documents/ {

[configuration C]

} 对应用户URI输入https://www.qq.com/documents/document.html

(路径匹配,匹配/documents/里的html)

location ^~ /images/ {

[configuration D]

} 对应用户URI输入https://www.qq.com/images/1.gif

(^~/images/是匹配任何以/images/开头的查询并停止其它匹配了,也包括不匹配正则了)

(路径匹配,优先匹配路径)

location ~\* \.(gif|jpg|jpeg)$ {

[configuration E]

} 对应用户URI输入https://www.qq.com/documents/1.jpg

(~\*是匹配所有,这里是匹配所有后缀是gif ,jpg ,jpeg的文件)

问题,自己location匹配的思路还是不太清晰????

+++++++++++++++++++++++++

**nginx rewrite(uri地址重写)**

nginx的rewrite功能是通过pcre软件包(pcre-devel)支持,即通过perl兼容正则表达式语法进行正则匹配.(默认参数编译安装时,nginx会自动安装支持rewrite的模块,但是也必须要有pcre软件的支持,故要先yum安装pcre-devel依赖包)

地址重写格式【总结】

rewrite 旧地址 新地址 [选项];

last 不再读其他rewrite

break 不再读其他语句，结束请求

redirect 临时重定向

permament 永久重定向

**nginx相关问题的解答**