https://www.cnblogs.com/coolid/p/9023650.html

[**nginx使用用户真实IP做hash（解决经过CND后ip\_hash失效问题）**](https://www.cnblogs.com/coolid/p/9023650.html)

在nginx中常用的有以下四种负载均衡的算法，分别是：round-robin、ip-hash、least-connected和weighted。当然在实际生产中或许使用最多的就是ip-hash了，一般会这样使用：

|  |  |
| --- | --- |
| 1  2  3  4  5 | upstream h5 {     ip\_hash;     server 192.168.100.104:9080;     server 192.168.100.105:9080;   } |

如果用户是直连的话那还好，nginx可以根据用户的IP均匀地向多个服务器节点分配负载请求。但是如果我们的域名使用了CDN加速的话，那么用户在请求js、CSS、图片等静态资源时并没有直接请求到我们的服务器，而是请求的少量的CDN加速节点服务器，从而造成有少量IP（PS：CDN节点服务器IP）频繁大量访问nginx。同时又因为ip\_hash策略的原因，导致出现部分服务器的负载非常大，其他服务器却没有多少请求的现象

因此，为了解决这个问题，我们可以通过在nginx中获取用户请求时的真实IP，然后根据这些真实IP做hash策略，也就是自定义nginx的hash策略。实现步骤如下：

（1）修改nginx配置文件nginx.conf：

|  |  |
| --- | --- |
| 1 | [root@tkde-iphone ~]# vim /usr/local/nginx/conf/nginx.conf |

http {  
include mime.types; #设定mime类型,类型由mime.type文件定义  
default\_type application/octet-stream;  
log\_format main ‘$remote\_addr – $remote\_user [$time\_local] “$request” ‘  
‘$status $body\_bytes\_sent “$http\_referer” ‘  
‘”$http\_user\_agent” “$http\_x\_forwarded\_for”‘;  
access\_log logs/access.log main;

#获取用户真实IP，并赋值给变量$clientRealIP

|  |  |
| --- | --- |
| 1  2  3  4 | map $http\_x\_forwarded\_for $clientRealIp {  "" $remote\_addr;  ~^(?P<firstAddr>[0-9\.]+),?.\*$ $firstAddr;  } |

……..

include gzip.conf; #压缩配置文件  
include proxy.conf; #proxy\_cache参数配置文件  
include vhost/\*.conf; #nginx虚拟主机包含文件目录  
include mysvrhost.conf; #后端WEB服务器列表文件  
}

（2）修改nginx的配置文件mysvrhost.conf：

|  |  |
| --- | --- |
| 1 | [root@tkde-iphone ~]# vim /usr/local/nginx/conf/mysvrhost.conf |

upstream h5 {  
hash $clientRealIp;  
server 192.168.100.104:9080;  
server 192.168.100.105:9080;  
}

注：这种方式也并不是万无一失了，因为请求的Header中的HTTP\_X\_FORWARDED\_FOR参数是可以在请求时被修改的，因此就存在一定的安全隐患。不过现在的CDN一般都有加速防黑的功能，所有实际上问题也不是很大。如果实在不放心的话不是还可以使用SSL证书整站加密嘛

转自：<https://www.zifangsky.cn/659.html>

完整代码:

|  |
| --- |
| #user nobody;  worker\_processes 1;  #error\_log logs/error.log;  #error\_log logs/error.log notice;  #error\_log logs/error.log info;  pid logs/nginx.pid;  events {  worker\_connections 1024;  }  http {  include mime.types;  default\_type application/octet-stream;  log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  '$status $body\_bytes\_sent "$http\_referer" '  '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  #access\_log logs/access.log main;  sendfile on;  #tcp\_nopush on;  #keepalive\_timeout 0;  keepalive\_timeout 65;  #gzip on;    #获取用户真实IP，并赋值给变量$clientRealIP  map $http\_x\_forwarded\_for $clientRealIp {  "" $remote\_addr;  ~^(?P<firstAddr>[0-9\.]+),?.\*$ $firstAddr;  }    upstream nttey1 {  #ip\_hash;  hash $clientRealIp;  server zxq:8085 max\_fails=2 fail\_timeout=10s weight=3;  server wulin:8085 max\_fails=2 fail\_timeout=10s weight=3;  server wulinThinkPad:8085 max\_fails=2 fail\_timeout=10s weight=3;  }  server {  listen 8500;  server\_name wulin;  location / {  proxy\_pass http://nttey1;  index index.html index.htm;  }  }  # another virtual host using mix of IP-, name-, and port-based configuration  #  #server {  # listen 8000;  # listen somename:8080;  # server\_name somename alias another.alias;  # location / {  # root html;  # index index.html index.htm;  # }  #}  # HTTPS server  #  #server {  # listen 443 ssl;  # server\_name localhost;  # ssl\_certificate cert.pem;  # ssl\_certificate\_key cert.key;  # ssl\_session\_cache shared:SSL:1m;  # ssl\_session\_timeout 5m;  # ssl\_ciphers HIGH:!aNULL:!MD5;  # ssl\_prefer\_server\_ciphers on;  # location / {  # root html;  # index index.html index.htm;  # }  #}  } |