https://blog.csdn.net/sd4493091/article/details/54894479?utm\_source=itdadao&utm\_medium=referral

**Nginx在局域网中使用ip\_hash负载均衡策略，访问全部分发到同一个后台服务器**

2017年02月06日 16:43:01 [Sunlight\_Rain\_](https://me.csdn.net/sd4493091) 阅读数：4132 标签： [nginx](http://so.csdn.net/so/search/s.do?q=nginx&t=blog)[iis](http://so.csdn.net/so/search/s.do?q=iis&t=blog)[负载均衡](http://so.csdn.net/so/search/s.do?q=%E8%B4%9F%E8%BD%BD%E5%9D%87%E8%A1%A1&t=blog)

现象：Nginx和客户端处于同一个局域网，使用ip\_hash负载均衡策略，不同客户端的请求全部都分发到同一个后台服务器

原因：

1、请看官方解释：  
This directive causes requests to be distributed between upstreams based on the IP-address of the client.   
The key for the hash is the class-C network address or the entire IPv6-address of the client. IPv6 is supported for ip\_hash since 1.3.2 or 1.2.2. This method guarantees that the client request will always be transferred to the same server. But if this server is considered inoperative, then the request of this client will be transferred to another server. This gives a high probability clients will always connect to the same server. （简译：将客户端ip转化成C类网络地址，然后将该网络地址当作hash关键字，来保证这个客户端请求总是被转发到一台服务器上）

2、请看Nginx的ip hash算法

1. for ( ;; ) {
3. for (i = 0; i < 3; i++) {
4. hash = (hash \* 113 + iphp->addr[i]) % 6271; *//iphp->addr[i]为ip的点分十进制法的第i段*
5. }
7. p = hash % iphp->rrp.peers->number;
9. n = p / (8 \* sizeof(uintptr\_t));
10. m = (uintptr\_t) 1 << p % (8 \* sizeof(uintptr\_t));
12. if (!(iphp->rrp.tried[n] & m)) {
14. ngx\_log\_debug2(NGX\_LOG\_DEBUG\_HTTP, pc->log, 0,
15. "get ip hash peer, hash: %ui %04XA", p, m);
17. peer = &iphp->rrp.peers->peer[p];
19. */\* ngx\_lock\_mutex(iphp->rrp.peers->mutex); \*/*
21. if (!peer->down) {
23. if (peer->max\_fails == 0 || peer->fails < peer->max\_fails) {
24. break;
25. }
27. if (now - peer->accessed > peer->fail\_timeout) {
28. peer->fails = 0;
29. break;
30. }
31. }
33. iphp->rrp.tried[n] |= m;
35. */\* ngx\_unlock\_mutex(iphp->rrp.peers->mutex); \*/*
37. pc->tries--;
38. }
40. if (++iphp->tries >= 20) {
41. return iphp->get\_rr\_peer(pc, &iphp->rrp);
42. }
43. }

主要代码

1. for(i = 0; i < 3; i++) {
2. hash = (hash \* 113+ iphp->addr[i]) % 6271;
3. }

for循环i取 012三个值，而ip的点分十进制表示方法将ip分成四段（如：192.168.1.1），但是这里循环时只是将ip的前三个段作为参数加入hash函数。这样做的目的是保证ip地址前三位相同的用户经过hash计算将分配到相同的后端server。

作者的这个考虑是极为可取的，因此ip地址前三位相同通常意味着来着同一个局域网或者相邻区域，使用相同的后端服务让nginx在一定程度上更具有一致性。

通过上述解释，已经基本判断出问题所在了。。  
主要原因就是，公司使用的局域网前三段ip相同，这样Nginx在ip\_hash（for循环后三个参数统一计入hash值）的时候，就将该类所有ip都转发到一个后端了。  
   
另，不论A类B类C类等网络地址，Nginx的ip\_hash算法都将一个ip地址的前三段作为hash的关键字。

参考：

http://www.linuxidc.com/Linux/2014-02/96868.htm

http://www.linuxidc.com/Linux/2014-02/96869.htm