https://blog.csdn.net/xqhys/article/details/81788358

**nginx负载均衡策略：ip\_hash、url\_hash**

2018年08月18日 03:17:41 [逆境中徘徊](https://me.csdn.net/xqhys) 阅读数：334更多

个人分类： [nginx](https://blog.csdn.net/xqhys/article/category/7958437)

转发：<https://blog.csdn.net/dengjiexian123/article/details/53105918>

**前言：**

目前，许多的web应用，或者web接口，都会在服务器的入口处，使用一个服务器容器来监听端口，然后进行请求转发，例如nginx Apache等。

服务器容器对应整个web服务有着至关重要的作用，包括：可以很好的管理服务进程，进行代理，对请求的预处理，以及负载均衡。

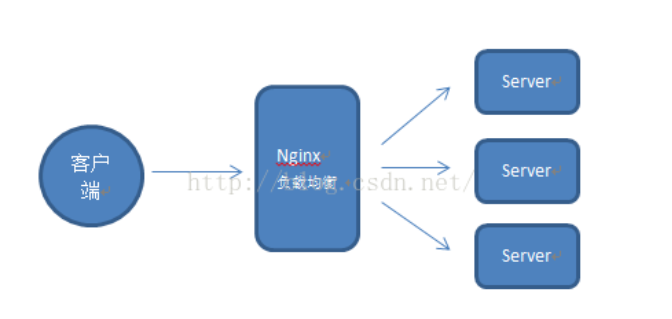
今天要讨论的重点为在服务器集群中，合理使用nginx的hash策略做更有意义的负载均衡。

**概要：**

当我们的服务是由一台服务器支撑时，就丝毫不存在负载均衡的概念。只有当服务由多台服务器（也就是服务器集群）支撑时，才会使用到负载均衡。

负载均衡顾名思义，是一种策略，用于防止一台服务器过载，而其他服务器闲置情况发生的策略。通过该策略可以使得提供相同服务的服务器负载基本相同。说得直白一点，就是当客户端发起一个请求之后，负载均衡会通过预先设定好的策略将该请求转发给上游的一台服务器进行处理。

如图所示：



负载均衡是一个很成熟的技术，其中对后端服务器进行轮询（默认）；通过客户端请求IP进行hash；对后端服务器指定权重等，是较为常见的负载均衡策略。这里不再赘述。

对服务盲目的采用负载均衡策略，是不太合理的。负载均衡默认情况下是轮询策略，这在一些场景下并不高效。

**更有意义的负载均衡：**

今天讲解的重点是，两种常见的负载均衡hash策略，以及对应的使用场景。

**1、ip\_hash（通过客户端请求ip进行hash，再通过hash值选择后端server）:**

当你服务端的一个特定url路径会被同一个用户连续访问时，如果负载均衡策略还是轮询的话，那该用户的多次访问会被打到各台服务器上，这显然并不高效（会建立多次http链接等问题）。甚至考虑一种极端情况，用户需要分片上传文件到服务器下，然后再由服务器将分片合并，这时如果用户的请求到达了不同的服务器，那么分片将存储于不同的服务器目录中，导致无法将分片合并。所以，此类场景可以考虑采用nginx提供的ip\_hash策略。既能满足每个用户请求到同一台服务器，又能满足不同用户之间负载均衡。

配置代码如下：

1. upstream backend{
2. ip\_hash;
3. server 192.168.128.1:8080 ;
4. server 192.168.128.2:8080 ;
5. server 192.168.128.3:8080 down;
6. server 192.168.128.4:8080 down;
8. }
9. server {
10. listen 8081;
11. server\_name test.csdn.net;
12. root /home/system/test.csdn.net/test;
13. location ^~ /Upload/upload {
14. proxy\_pass http:*//backend;*
16. }
18. }

上述是一个极简的监听8081端口的的nginx服务，其中当请求url 为/Upload/upload时，会走ip\_hash策略; upstream是nginx的负载均衡模块，此处，配置了策略为ip\_hash，参与负载均衡的机器有四台，其中后两台末尾添加了down关键字，表示下线的意思。

**2、url\_hash（通过请求url进行hash，再通过hash值选择后端server）：**

一般来讲，要用到urlhash，是要配合缓存命中来使用。举一个我遇到的实例：有一个服务器集群A，需要对外提供文件下载，由于文件上传量巨大，没法存储到服务器磁盘中，所以用到了第三方云存储来做文件存储。服务器集群A收到客户端请求之后，需要从云存储中下载文件然后返回，为了省去不必要的网络带宽和下载耗时，在服务器集群A上做了一层临时缓存（缓存一个月）。由于是服务器集群，所以同一个资源多次请求，可能会到达不同的服务器上，导致不必要的多次下载，缓存命中率不高，以及一些资源时间的浪费。在此类场景下，为了使得缓存命中率提高，很适合使用url\_hash策略，同一个url（也就是同一个资源请求）会到达同一台机器，一旦缓存住了资源，再此收到请求，就可以从缓存中读取，既减少了带宽，也减少的下载时间。

配置代码如下：

1. upstream somestream {
2. hash $request\_uri;
3. server 192.168.244.1:8080;
4. server 192.168.244.2:8080;
5. server 192.168.244.3:8080;
6. server 192.168.244.4:8080;
8. }
9. server {
10. listen 8081 default;
11. server\_name test.csdn.net;
12. charset utf-8;
13. location /get {
14. proxy\_pass http:*//somestream;*
16. }
17. }

上述同样也是一个极简的监听8081端口的nginx服务，当请求url为/get时，会走url\_hash；同样配置了upstream模块，hash $request\_uri表明了是按照url规则进行hash策略。

**总结：**

以上就是本文要介绍的全部内容，本文侧重讲解了ip\_hash和url\_hash的使用场景和基本配置，另外，在进行nginx server 配置时，可以灵活一些，不同的location采用不同的策略，可以使得服务策略更加的合理。希望此文能为各位带来些许帮助。