Nginx 负载均衡的几种方式

如果Nginx没有仅仅只能代理一台服务器的话，那它也不可能像今天这么火，Nginx可以配置代理多台服务器，当一台服务器宕机之后，仍能保持系统可用。具体配置过程如下：

1. 在http节点下，添加upstream节点。

upstream linuxidc {

server 10.0.6.108:7080;

server 10.0.0.85:8980;

}

2. 将server节点下的location节点中的proxy\_pass配置为：http:// + upstream名称，即“

http://linuxidc”.

location / {

root html;

index index.html index.htm;

proxy\_pass http://linuxidc;

}

3. 现在负载均衡初步完成了。upstream按照轮询（默认）方式进行负载，每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器down掉，能自动剔除。虽然这种方式简便、成本低廉。但缺点是：可靠性低和负载分配不均衡。适用于图片服务器集群和纯静态页面服务器集群。

除此之外，upstream还有其它的分配策略，分别如下：

weight（权重）

指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。如下所示，10.0.0.88的访问比率要比10.0.0.77的访问比率高一倍。

upstream linuxidc{

server 10.0.0.77 weight=5;

server 10.0.0.88 weight=10;

}

ip\_hash（访问ip）

每个请求按访问ip的hash结果分配，这样每个访客固定访问一个后端服务器，可以解决session的问题。

upstream favresin{

ip\_hash;

server 10.0.0.10:8080;

server 10.0.0.11:8080;

}

fair（第三方）

按后端服务器的响应时间来分配请求，响应时间短的优先分配。与weight分配策略类似。

upstream favresin{

server 10.0.0.10:8080;

server 10.0.0.11:8080;

fair;

}

url\_hash（第三方）

按访问url的hash结果来分配请求，使每个url定向到同一个后端服务器，后端服务器为缓存时比较有效。

注意：在upstream中加入hash语句，server语句中不能写入weight等其他的参数，hash\_method是使用的hash算法。

upstream resinserver{

server 10.0.0.10:7777;

server 10.0.0.11:8888;

hash $request\_uri;

hash\_method crc32;

}

upstream还可以为每个设备设置状态值，这些状态值的含义分别如下：

down 表示单前的server暂时不参与负载.

weight 默认为1.weight越大，负载的权重就越大。

max\_fails ：允许请求失败的次数默认为1.当超过最大次数时，返回proxy\_next\_upstream 模块定义的错误.

fail\_timeout : max\_fails次失败后，暂停的时间。

backup： 其它所有的非backup机器down或者忙的时候，请求backup机器。所以这台机器压力会最轻。

upstream bakend{ #定义负载均衡设备的Ip及设备状态

ip\_hash;

server 10.0.0.11:9090 down;

server 10.0.0.11:8080 weight=2;

server 10.0.0.11:6060;

server 10.0.0.11:7070 backup;

}