# 负载均衡概念

简而言之，负载均衡(Load Balance)是分布式系统架构设计中必须考虑的因素之一，它通常是指，将请求/数据【均匀】分摊到多个操作单元上执行，负载均衡的关键在于【均匀】。

# Nginx负载均衡实现

## 2.1常用负载均衡方式

1）DNS负载均衡，是最早的负载均衡技术是通过DNS来实现的，在DNS中为多个地址配置同一个名字，因而查询这个名字的客户机将得到其中一个地址，从而使得不同的客户访问不同的服务器，达到负载均衡的目的。DNS负载均衡是一种简单而有效的方法，但是它不能区分服务器的差异，也不能反映服务器的当前运行状态。

2）反向代理服务器负载均衡，使用代理服务器，可以将请求转发给内部的服务器，使用这种加速模式显然可以提升静态网页的访问速度。然而，也可以考虑这样一种技术，使用反向代理服务器将请求均匀转发给多台服务器，从而达到负载均衡的目的。反向代理（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，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上请求连接的客户端，此时代理服务器对外就表现为一个服务器。反向代理负载均衡技术是把将来自internet上的连接请求以反向代理的方式动态地转发给内部网络上的多台服务器进行处理，从而达到负载均衡的目的。

3）地址转换网关负载均衡，支持负载均衡的地址转换网关，可以将一个外部IP地址映射为多个内部IP地址，对每次TCP连接请求动态使用其中一个内部地址，达到负载均衡的目的。

4）HTTP重定向或者镜像功能，也可以自己定义协议，进行网络和负载的转发；

5）混合型负载均衡在有些大型网络，由于多个服务器群内硬件设备、各自的规模、提供的服务等的差异，可以考虑给每个服务器群采用最合适的负载均衡方式，然后又在这多个服务器群间再一次负载均衡或群集起来以一个整体向外界提供服务（即把这多个服务器群当做一个新的服务器群），从而达到最佳的性能。将这种方式称之为混合型负载均衡。此种方式有时也用于单台均衡设备的性能不能满足大量连接请求的情况下。

## 2.2 Nginx负载均衡配置方式

打开nginx.conf文件，添加如下内容

worker\_processes auto; #CPU工作核数

worker\_rlimit\_nofile 204800; #最大文件打开数

events {

use epoll; #使用epoll模式

worker\_connections 204800; #最大连接数

}

关键配置：

upstream fotamasters { #配置负载均衡的机器群

server 192.168.2.166:20007;

server 192.168.2.166:20010;

server 192.168.2.166:20013;

server 192.168.2.166:20016;

server 192.168.2.166:20019;

server 192.168.2.166:20022;

}

server {

listen 80;

server\_name fota.localhost.com; #需负载的域名

location / {

proxy\_pass <http://fotamasters>; #将请求数转至负载均衡机器群

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

}

至此，一个最简的负载均衡配置就已经搭建成功，默认采用的是轮询的方式进行工作。

# Nginx负载均衡工作方式

## 3.1轮询（默认）

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。

## 3.2 weight

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。

例如：

upstream bakend {

server 192.168.0.14 weight=10;

server 192.168.0.15 weight=10;

}

## 3.3 ip\_hash

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。

例如：

upstream bakend {

ip\_hash;

server 192.168.0.14:88;

server 192.168.0.15:80;

}

## 3.4 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。

upstream backend {

server server1;

server server2;

fair;

}

## 3.5 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。

例：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 backend {

server squid1:3128;

server squid2:3128;

hash $request\_uri;

hash\_method crc32;

}

tips:

upstream bakend{#定义负载均衡设备的Ip及设备状态

ip\_hash;

server 127.0.0.1:9090 down;

server 127.0.0.1:8080 weight=2;

server 127.0.0.1:6060;

server 127.0.0.1:7070 backup;

}

在需要使用负载均衡的server中增加

proxy\_pass http://bakend/;

每个设备的状态设置为:

1）down 表示单前的server暂时不参与负载

2）weight 默认为1.weight越大，负载的权重就越大。

3）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

4）fail\_timeout:max\_fails次失败后，暂停的时间。

5）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。

nginx支持同时设置多组的负载均衡，用来给不用的server来使用。

client\_body\_in\_file\_only 设置为On 可以讲client post过来的数据记录到文件中用来做debug

client\_body\_temp\_path 设置记录文件的目录 可以设置最多3层目录

location 对URL进行匹配.可以进行重定向或者进行新的代理 负载均衡