nginx负载服务器(1)

nginx负载服务器?

nginx

Nginx ("engine x") 是一个高性能的 HTTP 和 反向代理 服务器，也是一个 IMAP/POP3/SMTP 服务器。

nginx负载均衡分配方式

0)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。

1)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。

2)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。

3)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。

4)url_hash（第三方）

多台服务器，不同的服务器缓存不同的资源，当相同的url访问会到达同一台服务器。

nginx的优点

1.高并发连接

2.内存消耗少

3.配置文件简单

4.开源

5.内置的健康检查功能

6.稳定性高

nginx负载均衡配置

Nginx不单可以作为强大的web服务器，也可以作为一个反向代理服务器，而且nginx还可以按照调度规则实现动态、静态页面的分离，可以按照轮询、ip哈希、URL哈希、权重等多种方式对后端服务器做负载均衡，同时还支持后端服务器的健康检查。

配置步骤

在配置之前先 ping 下业务服务器

代理服务器 192.168.1.180

业务服务器 192.168.1.181 和 192.168.1.182

1. 在http节点里添加:#定义负载均衡设备的 Ip及设备状态

```
  upstream myServer {
       server 192.168.1.181:8080
      server 192.168.1.182:8080 weight=2;
      server 192.168.1.183 down;
      server 192.168.1.184 backup;
   }
```

PS：

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。

2.在需要使用负载的Server节点下添加

```
server{
        listen 80;
        server_name www.180.com;
        location / {
          proxy_pass        http://myServer;
          proxy_set_header   Host             $host;
          proxy_set_header   X-Real-IP        $remote_addr;
          proxy_set_header   X-Forwarded-For  $proxy_add_x_forwarded_for;
      }
 }
```

2. 在业务服务器的配置虚拟主句的配置文件里面更改如下参数

```
server{
        listen 8080;
        server_name myServer;
        index index.html index.php;
        root /data0/htdocs/www;
}
```

PS：

Nginx还支持多组的负载均衡,可以配置多个upstream 来服务于不同的Server.

配置负载均衡比较简单,但是最关键的一个问题是怎么实现多台服务器之间session的共享

下面有几种方法(以下内容来源于网络,第四种方法没有实践.)

1）不使用session，换作cookie

能把session改成cookie，就能避开session的一些弊端，在从前看的一本J2EE的书上，也指明在集群系统中不能用session，否则惹出祸端来就不好办。如果系统不复杂，就优先考虑能否将session去掉，改动起来非常麻烦的话，再用下面的办法。

2）应用服务器自行实现共享

asp.net可以用数据库或memcached来保存session，从而在asp.net本身建立了一个session集群，用这样的方式可以令 session保证稳定，即使某个节点有故障，session也不会丢失，适用于较为严格但请求量不高的场合。但是它的效率是不会很高的，不适用于对效率 要求高的场合。

以上两个办法都跟nginx没什么关系，下面来说说用nginx该如何处理：

3）ip_hash

nginx中的ip_hash技术能够将某个ip的请求定向到同一台后端，这样一来这个ip下的某个客户端和某个后端就能建立起稳固的session，ip_hash是在upstream配置中定义的：

```
upstream backend {
  server 127.0.0.1:8080 ;
  server 127.0.0.1:9090 ;
   ip_hash;
```

}

ip_hash是容易理解的，但是因为仅仅能用ip这个因子来分配后端，因此ip_hash是有缺陷的，不能在一些情况下使用：

1/ nginx不是最前端的服务器。ip_hash要求nginx一定是最前端的服务器，否则nginx得不到正确ip，就不能根据ip作hash。譬如使用的是squid为最前端，那么nginx取ip时只能得到squid的服务器ip地址，用这个地址来作分流是肯定错乱的。

2/ nginx的后端还有其它方式的负载均衡。假如nginx后端又有其它负载均衡，将请求又通过另外的方式分流了，那么某个客户端的请求肯定不能定位到同一台session应用服务器上。这么算起来，nginx后端只能直接指向应用服务器，或者再搭一个squid，然后指向应用服务器。最好的办法是用 location作一次分流，将需要session的部分请求通过ip_hash分流，剩下的走其它后端去。

4) upstream_hash

为了解决ip_hash的一些问题，可以使用upstream_hash这个第三方模块，这个模块多数情况下是用作url_hash的，但是并不妨碍将它用来做session共享：

假如前端是squid，他会将ip加入x_forwarded_for这个http_header里，用upstream_hash可以用这个头做因子，将请求定向到指定的后端：

可见这篇文档：http://www.sudone.com/nginx/nginx_url_hash.html

在文档中是使用$request_uri做因子，稍微改一下：

hash $http_x_forwarded_for;

这样就改成了利用x_forwarded_for这个头作因子，在nginx新版本中可支持读取cookie值，所以也可以改成：

hash $cookie_jsessionid;

假如在php中配置的session为无cookie方式，配合nginx自己的一个userid_module模块就可以用nginx自发一个cookie，可参见userid模块的英文文档：

http://wiki.nginx.org/NginxHttpUserIdModule

另可用姚伟斌编写的模块upstream_jvm_route：http://code.google.com/p/nginx-upstream-jvm-route/

---

nginx内置缓存功能?

配置方法:

在http域内添加如下参数

```
proxy_temp_path /usr/local/nginx/sy;
proxy_cache_path /usr/local/nginx/proxy_cache levels=1:2 keys_zone=cache_zone:20m inactive=1d max_size=100m;
```

在location里面使用proxy_cache，

配置方法如下:

```
location ~* \.php$ {
proxy_cache cache_zone;
proxy_cache_key $host$uri$is_args$args;
proxy_cache_valid any 1d;
proxy_pass http://127.0.0.1:8080;
}
```

了解更多
