https://blog.csdn.net/lxb15959168136/article/details/53113996/

**nginx 负载均衡时，一台tomcat宕机时的问题 自动切换**

2016年11月10日 11:19:32 [lxb15959168136](https://me.csdn.net/LXB15959168136) 阅读数：22906

如果[Nginx](https://www.baidu.com/s?wd=Nginx&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)没有仅仅只能代理一台[服务器](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)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](https://www.baidu.com/s?wd=backup&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)： 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;  
}

Nginx的配置与部署研究，Upstream负载均衡模块  <http://www.linuxidc.com/Linux/2013-04/82526p2.htm>

[CentOS](http://www.linuxidc.com/topicnews.aspx?tid=14) 6.2实战部署Nginx+MySQL+PHP <http://www.linuxidc.com/Linux/2013-09/90020.htm>

使用Nginx搭建WEB服务器 <http://www.linuxidc.com/Linux/2013-09/89768.htm>

搭建基于Linux6.3+Nginx1.2+PHP5+MySQL5.5的Web服务器全过程 <http://www.linuxidc.com/Linux/2013-09/89692.htm>

CentOS 6.3下Nginx性能调优 <http://www.linuxidc.com/Linux/2013-09/89656.htm>

CentOS 6.3下配置Nginx加载ngx\_pagespeed模块 <http://www.linuxidc.com/Linux/2013-09/89657.htm>

CentOS 6.4安装配置Nginx+Pcre+php-fpm <http://www.linuxidc.com/Linux/2013-08/88984.htm>

Nginx安装配置使用详细笔记 <http://www.linuxidc.com/Linux/2014-07/104499.htm>

Nginx日志过滤 使用ngx\_log\_if不记录特定日志 <http://www.linuxidc.com/Linux/2014-07/104686.htm>

**Nginx 的详细介绍**：[请点这里](http://www.linuxidc.com/Linux/2012-03/56786.htm)  
**Nginx 的下载地址**：[请点这里](http://www.linuxidc.com/down.aspx?id=342)

用了nginx负载均衡后，在两台tomcat正常运行的情况下，访问http://localhost 速度非常迅速，通过测试程序也可以看出是得到的负载均衡的效果，但是我们试验性的把其中一台tomcat（server localhost:8080）关闭后，再查看[http://localhost](http://localhost/)，发现反应呈现了一半反映时间快，一半反映时间非常非常慢的情况，但是最后都能得到正确结果.

解决办法：

问题解决，主要是proxy\_connect\_timeout   这个参数， 这个参数是连接的超时时间。 我设置成1，表示是1秒后超时会连接到另外一台服务器。  
#user  nobody;  
worker\_processes  1;  
  
#error\_log  logs/error.log;  
#error\_log  logs/error.log  notice;  
#error\_log  logs/error.log  info;  
  
#pid        logs/nginx.pid;  
  
  
events {  
    worker\_connections  1024;  
}  
  
  
http {  
    include       mime.types;  
    default\_type  application/octet-stream;  
  
     upstream localhost {  
       #ip\_hash;  
       server 127.0.0.1:8081;  
       server 127.0.0.1:8080;  
     }  
  
    #log\_format  main  '$remote\_addr - $remote\_user [$time\_local] "$request" '  
    #                  '$status $body\_bytes\_sent "$http\_referer" '  
    #                  '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  
  
    #access\_log  logs/access.log  main;  
  
    sendfile        on;  
    #tcp\_nopush     on;  
  
    #keepalive\_timeout  0;  
    keepalive\_timeout  65;  
  
    #gzip  on;  
  
    server {  
        listen       80;  
        server\_name  localhost;  
  
    listen 80;  
    server\_name localhost;  
    location /{  
    proxy\_pass http://localhost;  
    proxy\_set\_header Host $host;  
    proxy\_set\_header X-Real-IP $remote\_addr;  
    proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
**proxy\_connect\_timeout       1;  
    proxy\_read\_timeout          1;  
    proxy\_send\_timeout          1;**  
    }  
        #charset koi8-r;  
  
        #access\_log  logs/host.access.log  main;  
  
        #error\_page  404              /404.html;  
  
        # redirect server error pages to the static page /50x.html  
        #  
        error\_page   500 502 503 504  /50x.html;  
        location = /50x.html {  
            root   html;  
        }  
  
        # proxy the PHP scripts to Apache listening on 127.0.0.1:80  
        #  
        #location ~ \.php$ {  
        #    proxy\_pass   http://127.0.0.1;  
        #}  
  
        # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000  
        #  
        #location ~ \.php$ {  
        #    root           html;  
        #    fastcgi\_pass   127.0.0.1:9000;  
        #    fastcgi\_index  index.php;  
        #    fastcgi\_param  SCRIPT\_FILENAME  /scripts$fastcgi\_script\_name;  
        #    include        fastcgi\_params;  
        #}  
  
        # deny access to .htaccess files, if Apache's document root  
        # concurs with nginx's one  
        #  
        #location ~ /\.ht {  
        #    deny  all;  
        #}  
    }  
  
  
    # another virtual host using mix of IP-, name-, and port-based configuration  
    #  
    #server {  
    #    listen       8000;  
    #    listen       somename:8080;  
    #    server\_name  somename  alias  another.alias;  
  
    #    location / {  
    #        root   html;  
    #        index  index.html index.htm;  
    #    }  
    #}  
  
  
    # HTTPS server  
    #  
    #server {  
    #    listen       443;  
    #    server\_name  localhost;  
  
    #    ssl                  on;  
    #    ssl\_certificate      cert.pem;  
    #    ssl\_certificate\_key  cert.key;  
  
    #    ssl\_session\_timeout  5m;  
  
    #    ssl\_protocols  SSLv2 SSLv3 TLSv1;  
    #    ssl\_ciphers  ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;  
    #    ssl\_prefer\_server\_ciphers   on;  
  
    #    location / {  
    #        root   html;  
    #        index  index.html index.htm;  
    #    }  
    #}  
  
}