**nginx负载均衡笔记**

**nginx负载均衡介绍**

Nginx除了可以用作web服务器外，

还可以用来做高性能的反向代理服务器，它能提供稳定高效的负载均衡解决方案。

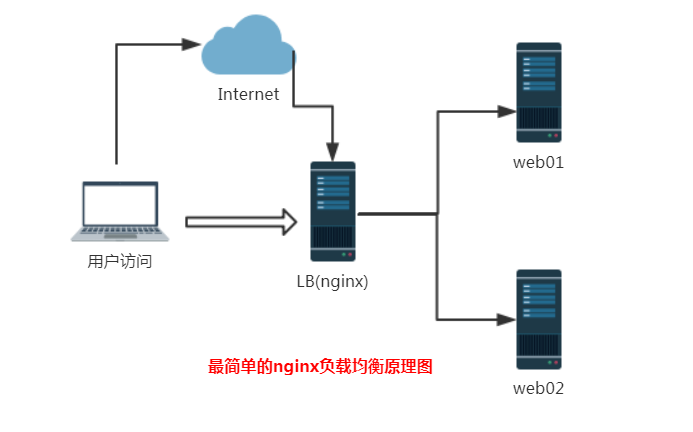
nginx可以用轮询、IP哈希、URL哈希等方式调度后端服务器，同时也能提供健康检查功能。

目前有众多公司均已经部署使用nginx实现基于七层的负载均衡功能。

Nginx负载均衡

为了实现Nginx的反向代理以及负载均衡功能，应用中需要用到两个模块，HttpProxyModule和HttpUpstreamModule模块；

1. **upstream ngx\_http\_upstream\_module 定义节点和调度算法**
2. **http\_proxy 负责请求的转发，proxy\_pass 发给定义好的指定upstream名字**



**3、负载均衡几种模式介绍**

**1、轮询(默认)**

默认选项，当weight不指定时，各服务器weight相同， 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。session会话不能保持

upstream bakend {

server 192.168.1.10;

server 192.168.1.11;

}

**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。如果后端服务器down掉，能自动剔除。 比如下面配置，则1.11服务器的访问量为1.10服务器的两倍（后端节点中配置高的服务器可以适当将weight设置大点）。

upstream bakend {

server 192.168.1.10 weight=1;

server 192.168.1.11 weight=2;

}

**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不能跨服务器的问题，实现session共享。如果后端服务器down掉，要手工处理。负载均衡实现会话保持的方法，缺点：负载不均衡

upstream resinserver{

ip\_hash;

server 192.168.1.10:8080;

server 192.168.1.11:8080;

}

**备注：**

**可以通过memcache php默认的/tmp cookies 来实现session会话保持**

**memcached：修改php.ini**

**session.save\_handler = files 为session.save\_handler = memcache**

**session.save\_path = "/tmp" 为 session.save\_path = "tcp://10.10.10.12:11211"**

**4、fair（第三方插件）**  
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 resinserver{

server 192.168.1.10:8080;

server 192.168.1.11:8080;

fair;

}

**5、url\_hash（第三方插件）**  
**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，后端服务器为缓存服务器时比较有效**。 在upstream中加入hash语句，hash\_method是使用的hash算法。

upstream resinserver{

server 192.168.1.10:8080;

server 192.168.1.11:8080;

hash $request\_uri;

hash\_method crc32;

}

1. **Nginx 官方参考----**[**ngx\_http\_upstream\_module**](http://nginx.org/en/docs/http/ngx_http_upstream_module.html)**-**

http://nginx.org/en/docs/http/ngx\_http\_upstream\_module.html

upstream backend {

server backend1.example.com weight=5;

server backend2.example.com:8080;

server unix:/tmp/backend3;

server backup1.example.com:8080 backup;

server backup2.example.com:8080 backup;

}

server {

location / {

proxy\_pass http://backend;

}

}

Example Configuration：

location / {

proxy\_pass http://localhost:8000;

proxy\_set\_header Host $host;

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

}

**6、配置节点服务器只监听虚拟IP（可选）**

其他简单的在虚拟机上需要整理：

关于虚拟VIP 10.10.10.10配置只监听虚拟IP 将来如果主nginx挂掉了，可以直接在从机器上绑定IP就完成切换

[root@web01 vhost]# netstat -lntup|grep nginx

tcp 0 0 10.10.10.10:80 0.0.0.0:\* LISTEN 4803/nginx: master

修改配置文件

server

{

listen 10.10.10.10:80;

server\_name linyaohong.com www.linyaohong.com ;

location / {

proxy\_pass http://www\_server\_pools;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

}

**配置实战-http区域**

upstream www\_server{

server 10.10.10.8:80;

server 10.10.10.7:80;

server 10.10.10.20 backup;

server 10.10.10.21 backup;

}

#轮询 可以使用backup 热备

**upstream www\_server\_pools{**

**server 10.10.10.8:80** **weight=3** **max\_fails=1 fail\_timeout=3; #测试了没生效**

**server 10.10.10.7:80 weight=7;**

**server 10.10.10.20 backup;**

**server 10.10. 10.21 backup;**

**}**

#ip\_hash 如果nginx代理Cache可能会用到、使用down可以确保客户端按照当前的hash算法访问、

upstream www\_server{

ip\_hash;

server 10.10.10.8:80;

server 10.10.10.7:80;

server 10.10.10.20:80 down;

}

#使用域名及socket的upstream配置案例,**需要内网有NDS服务器或者在负载均衡机器上做好hosts文件解析**

upstream www\_server{

server lb01.linyaohong.com;

server lb02.linyaohong.com:8088; #y域名加端口

server unix:/tmp/www\_server; #指定socket文件

}

include /server/application/nginx/conf/vhost/\*.conf;

}

|  |  |
| --- | --- |
| upstream模块内参数 | 参数说明 |
| server 10.10.10.20:80 | 负载均衡后面的RS配置，可以是IP或域名，如果端口不写默认80  高并发场景下，IP可以换成域名，通过NDS做负载均衡  后面还可以加VIP扩展更多 |
| max\_falis=1 | nginx尝试连接后端主机失败的次数、这个值配合proxy\_next\_upstream/fastcgi\_next\_upstream/memcached\_next\_upstream  这三个参数来使用，当nginx接收后端服务器返回这三个参数定义的状态没时，会将请求转发给正常工作的后端服务器。  例如404.502.503 max\_fails默认是1  企业建议2-3 京东：1 根据业务需求配置、 蓝汛（cdn）10次 |
| backup | 热备配置、当前面激活的RS都失败后，会自动启用热备RS  若主服务器全部宕机了，就会转发请求到backup  当负载均衡算法为ip\_hash时。后端服务器在负载均衡调度中的状态  不能为 weight和backup |
| fail\_timeout=10s | 在max\_falis定义的失败次数后，如果是1就检测1次  如果max\_falis是5就检测5次，如果5次都失败，  则根据fail\_timeout的值等待10秒再去检查一次，  企业2-3秒比较合理，默认是10秒  蓝汛（cdn）3秒 京东3秒 |
| down | 这标志后端服务不可用。这个参数可以配合ip\_hash来使用 |

**server配置**

server

{

listen 80;

server\_name linyaohong.com www.linyaohong.com ;

location / {

proxy\_pass http://www\_server;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

}

**Tomcat-server配置**

server {

listen 80;

server\_name linyaohong.com;

access\_log /data/wwwlogs/access\_nginx.log combined;

root /data/wwwroot/default;

index index.html index.htm index.jsp;

location ~ {

proxy\_pass http://127.0.0.1:8080;

include proxy.conf;

}

}

**案例配置**

**server基本上没区别**

server

{

listen 80;

server\_name www.linyaohong.com linyaohong.com ;

location / {

proxy\_pass http://www\_server;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

}

server

{

listen 80;

server\_name bbs.linyaohong.com ;

location / {

proxy\_pass http://bbs\_server;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

}

server

{

listen 80;

server\_name blog.linyaohong.com ;

location / {

proxy\_pass http://blog\_server;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

}

upstream www\_server{

server 10.10.10.8;

server 10.10.10.7;

server 10.10.10.20 backup;

server 10.10.10.21 backup;

}

**此处用的host来区分后端虚拟主机（如果后端有多个虚拟主机的情况下）**

**即：proxy\_set\_header Host $host;**

**虽然都是直接访问的IP 但是server里带了host、所以会请求后端IP对应的host主机头**

upstream blog\_server{

#ip\_hash;

#server 10.10.10.20:8082 weight=5;

#server 10.10.10.21:8082 weight=1;

server 10.10.10.20;

server 10.10.10.21;

}

upstream bbs\_server{

#ip\_hash;

#server 10.10.10.20:8081 weight=5;

#server 10.10.10.21:8081 weight=1;

server 10.10.10.20;

server 10.10.10.21;

}

upstream www\_server{

#ip\_hash;

#server 10.10.10.20:8080 weight=5;

#server 10.10.10.21:8080 weight=1;

server 10.10.10.20;

server 10.10.10.21;

}

**此处用的端口来区分后端虚拟主机（如果后端有多个虚拟主机的情况下）**

upstream blog\_server{

#ip\_hash;

#server 10.10.10.20:8082 weight=5;

#server 10.10.10.21:8082 weight=1;

server 10.10.10.20:8082;

server 10.10.10.21:8082;

}

upstream bbs\_server{

#ip\_hash;

#server 10.10.10.20:8081 weight=5;

#server 10.10.10.21:8081 weight=1;

server 10.10.10.20:8081;

server 10.10.10.21:8081;

}

upstream www\_server{

#ip\_hash;

#server 10.10.10.20:8080 weight=5;

#server 10.10.10.21:8080 weight=1;

server 10.10.10.20:8080;

server 10.10.10.21:8080;

}

**http\_proxy\_module**

**Tomcat和nginx配合常用配置**

location ~ {

proxy\_pass http://127.0.0.1:8080;

include proxy.conf;

}

**根据URL为name来转发到另外一台服务器或本地**

**url\_hash算法**

location /name/ {

proxy\_pass http://127.0.0.1/remote;

}

location /name/ {

proxy\_pass http://127.0.0.1;

}

location /name/ {

rewrite /name/([^/]+) /user?name=$1 breank;

proxy\_pass http://127.0.0.1;

}

**proxy实现动静分离**

**首先定义upstream**

upstream status\_server {

server 10.10.10.20;

server 10.10.10.21;

check interval=3000 rise=2 fall=5 timeout=1000 type=http;

}

upstream static\_server {

server 10.10.10.20;

}

upstream upload\_server {

server 10.10.10.21;

}

upstream default\_server {

server 10.10.10.20;

}

server

{

listen 80;

server\_name 10.10.10.5 ;

location / {

check\_status;

access\_log off;

}

location /status {

check\_status;

access\_log off;

}

}

server

{

listen 80;

server\_name www.linyaohong.com linyaohong.com ;

location / {

proxy\_pass http://default\_server;

include proxy.conf;

}

location /static/ {

proxy\_pass http://static\_server;

include proxy.conf;

}

location /upload/ {

proxy\_pass http://upload\_server;

include proxy.conf;

}

}

}

**总结：符合static的请求交给static\_server、符合上传的请求交给upload\_server、不符合上述规定的全部默认交给default\_server**

**或者用if语句来实现**

if ($request\_uri ~\* “^/static/(.\*)$”)

{

proxy\_pass <http://static_server>;

}

其他同理

.......................

**根据url进行转发的应用情况，被称为第7层（应用层）的负载均衡**

**而LVS的负载均衡一般用户TCP等的转发。因此被称为第4层（传输层）的负载均衡**

**proxy实现动静分离应用场景：**

1. **业务的域名没有拆分，或者不希望拆分、但是希望动静分离、多业务分离**
2. **不通的客户端设备，使用同一个域名访问同一个业务网站**

**根据客户端设备user\_agent转发**

location / {

if ($http\_user\_agent ~\* “MSIE”) {

proxy\_pass <http://static_server>;

{

#if ($http\_user\_agent ~\* “Fierfox”) {

if ($http\_user\_agent ~\* “Chrome) {

proxy\_pass <http://upload_server>;

}

proxy\_pass http://default \_server;

include proxy.conf

}

**如果使用注意提前定义upstream**

location / {

if ($http\_user\_agent ~\* “android”) {

proxy\_pass <http://android_server>;

{

if ($http\_user\_agent ~\* “iphone”) {

proxy\_pass <http://iphone_server>;

}

proxy\_pass <http://pc_server>;

include proxy.conf

}

**根据文件扩展名实现代理转发--------也可以实现动静分离**

location ~ .\*.( gif|jpg|jpep|png|bmp|swf|css|js)$ {

proxy\_pass <http://static_server>;

include proxy.conf

}

**if方法**

location / {

if ($request\_uri ~\* “.\*\.(gif|jpg|jpep|png|bmp|swf|css|js)$)” {

proxy\_pass <http://static_server>;

}

if ($request\_uri ~\* “.\*\.(php|php5)$”) {

proxy\_pass <http://php_server>;

}

if ($request\_uri ~\* “.\*\.(jsp|jsp\*|do|do\*)$)”{

proxy\_pass <http://java_server>;

}

}

**http proxy 模块参数**

|  |  |
| --- | --- |
| **http proxy模块相关参数** | **说明** |
| proxy\_pass http://www\_server; | 通过proxy通能,把用户的请求转发到反向代理定义的upstream服务器池 |
| proxy\_set\_header | 设置http请求header项传给后端服务器节点  例如：可以实现让代理后端的服务器节点、获取访问客户端用户的真实IP |
| proxy\_set\_header Host $host; | 在代理向后端服务器发送的http请求头部中加入host字段信息，  当后端有多个虚拟主机的时候、来区分访问的哪一个 |
| proxy\_set\_header X-Forwarded-For $remote\_addr; | 在搭理向后端发送http请求头部中加入X-Forwarded-For 字段信息，用户记录用户访问的真实IP而不是代理服务器的IP |
| client\_body\_buffer\_size | 用户指定客户端请求主体缓冲区大小 |
| proxy\_connect\_timeout 60; | 表示反向代理与后端服务器连接超时的世界，即发起握手等候响应的时间 |
| proxy\_send\_tmeout 60; | 表示代理后端服务器的数据回传时间。  即：在这规定时间后端服务器必须传完所有的数据，否则nginx断开这个连接 |
| proxy\_read\_timeout 60; | 设置nginx从代理的后端服务器获取信息的时间，  表示连接建立成功后nginx等待后端服务器的响应时间，  其实是nginx已经进入后端的排队之中等待处理的时间 |
| proxy\_buffer\_size 32k; | 设置缓冲区大小。默认该缓冲区等于指令proxy\_buffer 设置的大小 |
| proxy\_buffers 4 32k; | 设置缓冲区的数量和大小。nginx从代理的后端服务器获取的响应信息，会放置到缓冲区 |
| proxy\_busy\_buffers\_size 64k; | 用户设置系统很忙时，可以使用proxy\_buffer大小，官方推荐为proxy\_buffer\*2 |
| proxy\_temp\_file\_write\_size 64k; | 指定proxy缓存临时文件的大小 |
|  |  |

location / {

proxy\_pass http://bbs\_server;

include proxy.conf

}

**说明：把proxy写在一个配置文件里 这样整洁规范**

**from boy：**

proxy\_set\_header Host $host;

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

proxy\_connect\_timeout 60;

proxy\_send\_timeout 60;

proxy\_read\_timeout 60;

proxy\_buffer\_size 4k;

proxy\_buffers 4 32k;

proxy\_busy\_buffers\_size 64k;

proxy\_temp\_file\_write\_size 64k;

**from one**

proxy\_set\_header Host $host;

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

proxy\_connect\_timeout 300s;

proxy\_send\_tmeout 900;

proxy\_read\_timeout 900;

proxy\_buffer\_size 32k;

proxy\_buffers 4 64k;

proxy\_busy\_buffers\_size 128k;

proxy\_redirect off;

proxy\_hide\_header Vary;

proxy\_set\_header Accept-Encoding '';

proxy\_set\_header Referer \$http\_referer;

proxy\_set\_header Cookie \$http\_cookie;

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

proxy\_set\_header X-Forwarded-Proto \$scheme;

**from bt**

proxy\_temp\_path /www/server/nginx/proxy\_temp\_dir;

proxy\_cache\_path /www/server/nginx/proxy\_cache\_dir levels=1:2 keys\_zone=cache\_one:20m inactive=1d max\_size=5g;

client\_body\_buffer\_size 512k;

proxy\_connect\_timeout 60;

proxy\_read\_timeout 60;

proxy\_send\_timeout 60;

proxy\_buffer\_size 32k;

proxy\_buffers 4 64k;

proxy\_busy\_buffers\_size 128k;

proxy\_temp\_file\_write\_size 128k;

proxy\_next\_upstream error timeout invalid\_header http\_500 http\_503 http\_404;

proxy\_cache cache\_one;

## [proxy\_next\_upstream实现容灾和重复处理问题](https://www.cnblogs.com/linkenpark/p/7807934.html)

proxy\_next\_upstream error timeout invalid\_header http\_500 http\_502 http\_503 http\_504 http\_404;

当其中一台返回错误码404,500...等错误时，可以分配到下一台服务器程序继续处理，提高平台访问成功率，多可运用于前台程序负载

proxy\_next\_upstream off

因为proxy\_next\_upstream 默认值: proxy\_next\_upstream error timeout;

场景:当访问A时，A返回error timeout时，访问会继续分配到下一台服务器处理，

就等于一个请求分发到多台服务器，就可能出现多次处理的情况，

如果涉及到充值，就有可能充值多次的情况，这种情况下就要把proxy\_next\_upstream关掉

**Nginx负载均衡检测节状态**

wget [https://github.com/yaoweibin](https://github.com/yaoweibin/nginx_upstream_check_module/archive/master.zip)[/nginx\_upstream\_check\_module/archive/master.zip](https://github.com/yaoweibin/nginx_upstream_check_module/archive/master.zip)

unzip master.zip

cd nginx-1.14.2

patch -p1 < ../nginx\_upstream\_check\_module-master/check\_1.14.0+.patch

/server/application/nginx/sbin/nginx -V

./configure --prefix=/server/application/nginx-1.14.2 --user=www --group=www --with-http\_stub\_status\_module --with-http\_v2\_module --with-http\_ssl\_module --with-http\_gzip\_static\_module --with-http\_realip\_module --with-http\_flv\_module --with-http\_mp4\_module --with-openssl=../openssl-1.0.2q --with-pcre=../pcre-8.42 --with-pcre-jit --with-ld-opt=-ljemalloc --with-zlib=../zlib-1.2.9 --add-module=../nginx\_upstream\_check\_module-master/

#备份老的

cp /server/application/nginx/sbin/nginx /server/application/nginx/sbin/nginx.bak

#删除老的配置文件

rm -f /server/application/nginx/sbin/nginx

#把打过补丁的nginx文件移动sbin目录下

cp /server/tools/nginx-1.14.2/objs/nginx /server/application/nginx/sbin/

/etc/init.d/nginx –s

upstream status\_server {

server 10.10.10.20;

server 10.10.10.21;

**check** **interval=3000 rise=2 fall=5 timeout=1000 type=http;**

**#对此池所有的节点 每隔3秒检测1次，请求2次正常则标记realserver状态为up**

**#如果5次都失败认为realserver为down，超时时间为1秒，检查协议为http**

}

server

{

listen 80;

server\_name 10.10.10.5 ;

location /status {

check\_status;

access\_log off;

}

}

