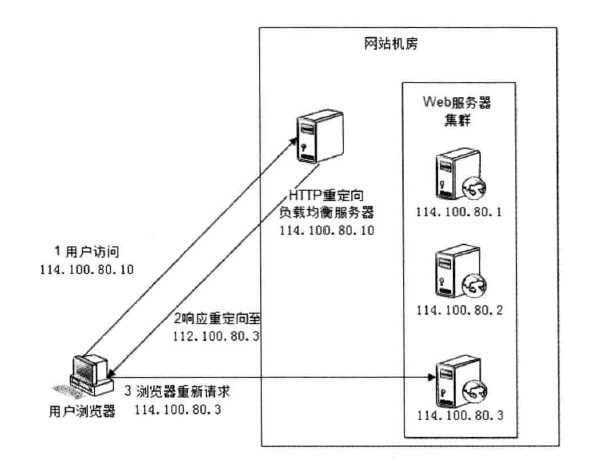
负载均衡策略：

1.http重定向，



2.dns负载均衡

在dns上配置多个a记录



3.nginx反向代理



Ip负载均衡

与反向代理类似，基于snat地址转换

负载均衡算法：

**轮询**

**加权轮询**

**随机**

**源地址散列**

**最少链接**（记录每个服务器现有的连接数）

作业帮负载

最前端负载均衡节点（**blb**，一组机器设置一个内网或外网服务地址）,负责流量转发

转发到我们自己的router机器

Router机器在rewrite中解析不同的url转发到不同**的服务组**

location ~\* "^/+(napi|access|practice|charge|appuser|assistant|biz|pay|game)/.\*" {

proxy\_pass **http://napi.na.bjcq;**

proxy\_intercept\_errors on;

include allow-origin.conf;

break; }，

proxy pass到upstream模块

**upstream napi.na.bjcq {**

**use\_bns;**

bns\_threshold 40;

server 192.168.2.187:8099 weight=10 max\_fails=20 fail\_timeout=20s;

keepalive 16;

}

Bns:（就是一个配置proxy\_pass、upstream的平台）

upstream napi.na.bjcq

{

server 192.168.1.55:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.1.56:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.1.58:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.1.60:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.1.61:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.2.55:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.2.56:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.2.60:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.2.53:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.2.54:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.2.57:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.1.6:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.2.59:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.2.62:8099 weight=6 max\_fails=20 fail\_timeout=20;

server 192.168.2.63:8099 weight=6 max\_fails=20 fail\_timeout=20;

}

服务组负载均衡的策略通常为random,也可以根据不同的机器配置不同的weight，根据不同的weight以不同的概率轮询.

服务组配置：

{

"services" : {

"group.sms.na.bjcq" : {

"service\_conn\_type" : "SHORT",

"service\_port" : 8099,

"service\_retry" : 1,

"service\_ctimeout" : 100,

"service\_rtimeout" : 2000,

"service\_wtimeout" : 2000,

"protocol" : {

"name" : "http"

},

"converter" : {

"name" : "form"

},

"SuperStrategy" : {

"Balance" : "Random",

"ConnectQueueSize" : 100,

"ConnectX1" : 10,

"ConnectX2" : 40,

"ConnectY1" : 95,

"ConnectY2" : 5,

"Hybrid" : "Off",

"CrossRoom" : "Off",

"HealthyBackupThreshold" : 3,

"HealthyCheckTime" : 3,

"HealthyMinRate" : 0.1,

"HealthyQueueSize" : 100,

"HealthyTimeout" : 100

},

"idc\_map": {

"yun": {

"prefer": "all"

},

"default": {

"prefer": "all"

}

}

}

}

}

如果后端服务器down掉,能自动剔除

Rpc负载：

**Local模式：**

[...@Service]

Name : sms-service

DefaultPort : 80

DefaultRetry : 1

…….

[....@Server]

**Hostname : sms.int.zybang.com //blb节点**

[....SuperStrategy]

Balance : Random

ConnectQueueSize : 100

**Webfoot**

[..Webfoot]

[...@Service]

Name : **group.sms.na.bjcq（服务组-和节点挂钩）**

Rename : zyb-sms

Webfoot配置name为一个服务组，

不同的服务组由一个或多个node实现，

一个node由一组机器组成。

ngiNx负载均衡

nginx的upstream目前支持的5种方式的分配

**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。   
upstream backserver {   
server 192.168.0.14;   
server 192.168.0.15;   
}   
  
**2、指定权重**  
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。   
upstream backserver {   
server 192.168.0.14 weight=10;   
server 192.168.0.15 weight=10;   
}   
  
**3、IP绑定 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。   
upstream backserver {   
ip\_hash;   
server 192.168.0.14:88;   
server 192.168.0.15:80;   
}   
  
**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 backserver {   
server server1;   
server server2;   
fair;   
}   
  
**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。   
upstream backserver {   
server squid1:3128;   
server squid2:3128;   
hash $request\_uri;   
hash\_method crc32;   
}

/\*\*\*\*\*\*\*\*\*\*\*\*location+upstream\*\*\*\*\*\*\*\*\*\*/

upstream backend {

    server 192.168.43.158:8086 weight=4 max\_fails=2 fail\_timeout=30s;

    server 192.168.41.167 weight=4 max\_fails=2 fail\_timeout=30s;

    server unix:/tmp/backends weight=4 max\_fails=2 fail\_timeout=30s;

}

server {

    listen 80 ;

    server\_name frontend.com;

    location = / {

        proxy\_pass http://backend;

        proxy\_set\_header Host backend.com;

        proxy\_set\_header Forwarded $remote\_addr;

    }

}