一、静态调度算法

1.1 static-rr

在 listen 或 backend 区域配置

balance static-rr

基于权重的轮询调度，不支持权重的运行时调整及后端服务器慢启动，其后端主机数量没有限制。weight 默认为 1。

```
listen web_host
bind 192.168.7.101:80,:8801-8810,192.168.7.101:9001-9010
mode http/tcp
log global
balance static-rr
server web1 192.168.7.103:80 weight 1 check inter 3000 fall 2 rise 5
server web2 192.168.7.104:80 weight 2 check inter 3000 fall 2 rise 5
```

1.2 first

在 listen 或 backend 区域配置

balance first

根据服务器在列表中的位置，自上而下进行调度，但是其只会当第一台服务器的连接数达到上限，新请求才会分配给下一台服务，因此会忽略服务器的权重设置。

如果有 maxconn，则匹配 maxconn 的最大值。

```
listen web_host
bind 192.168.7.101:80,:8801-8810,192.168.7.101:9001-9010
mode http/tcp
log global
balance first
server web1 192.168.7.103:80 maxconn 2 weight 1 check inter 3000 fall 2 rise 5
server web2 192.168.7.104:80 weight 1 check inter 3000 fall 2 rise 5
```

二、动态调度算法

2.1 roundrobin

在 listen 或 backend 区域配置，该算法为默认算法

balance roundrobin

基于权重的轮询动态调度算法，可以使用 socat 动态调整权重。

HAProxy 的 roundrobin 轮询模式不完全等于 LVS 的 RR 轮训模式，区别在于

①HAProxy 中的 roundrobin支持慢启动，新加的服务器会逐渐增加转发数（需要自己实现或手动操作，算法并不直接内置慢启动功能）；

②HAProxy 每个后端 backend 中最多支持 4095 个 real server；

③HAProxy支持对 real server 权重动态调整；

```
listen web_host
bind 192.168.7.101:80
```

```
mode http/tcp
log global
balance roundrobin
server web1 192.168.7.103:80 weight 1 check inter 3000 fall 2 rise 5
server web2 192.168.7.104:80 weight 2 check inter 3000 fall 2 rise 5
```

2.2 leastconn

在 listen 或 backend 区域配置，该算法为默认算法

balance leastconn

当前后端服务器连接最少的优先调度。

①支持加权的最少连接

②支持动态调整权重

③支持慢启动

```
listen web_host
bind 192.168.7.101:80
mode http/tcp
log global
balance leastconn
server web1 192.168.7.103:80 weight 1 check inter 3000 fall 2 rise 5
server web2 192.168.7.104:80 weight 2 check inter 3000 fall 2 rise 5

#weight 2，这表示在使用基于权重的负载均衡算法时，权重越高，被分配到的请求越多。
#check 这个关键字启用对后端服务器的健康检查，确保服务器能够响应请求。如果没有这个选项，HAProxy 可能不会自动检测服务器的健康状态。
#inter 3000 设置健康检查的间隔时间，这里是 3000 毫秒（3 秒）。这意味着 HAProxy 每 3 秒会检查一次该服务器的健康状态。
#fall 2 定义了服务器被视为“不健康”或“失败”的条件，即连续两次检查失败后，服务器将被标记为不可用。这里的 2 表示在 inter 指定的时间间隔内，如果连续两次检查失败，则认为服务器失效。
#rise 5 与 fall 相对，定义了服务器从“不健康”状态恢复到“健康”状态所需的连续成功检查次数。在这个例子中，即使服务器在一次检查中恢复，它也需要再连续五次检查成功才会被重新标记为可用。
```

三、其他调度算法

3.1 source

源地址 hash，基于用户源地址 hash 并将请求转发到后端服务器。

第一种，默认为静态即取模方式；

第二种，可以通过 hash-type支持的选项更改，动态即 hash 一致方式。

3.1.1 source 之 map-base 取模法

在 listen 或 backend 区域配置

```
mode http/tcp
balance source
```

取模法，基于服务器总权重的 hash 数组取模。
不支持在线调整权重；
不支持慢启动；
缺点：当服务器的总权重发生变化时，即有服务器上线或下线，都会因权重发生变化而导致调度结果整体改变。
例如，后端有三台服务器，每台服务器权重都为 1，则总权重为 3，haproxy 给源地址做 hash 计算，hash_value%3。其中获取的值如果为 0，发给第一台服务器；如果为 1，发给第二台服务器；如果为 2，发给第三台服务器。如果一台服务器坏掉了，其要在通过 hash_value%3 计算，因权重发生变化而导致调度结果整体改变。

3.1.2 source 之一致 hash 法

在 listen 或 backend 区域配置

```
mode http/tcp
balance source
hash-type consistent
```

一致性哈希，该 hash 是动态的。
支持在线调整权重；
支持慢启动；
优点：当服务器的总权重发生变化时，对调度结果影响是局部的，不会引起大的变动。
使用场景：在没有做 session 共享的情况下，可以做会话保持。

3.2 uri

基于对用户请求的 uri 做 hash 并将请求转发到后端指定服务器。

3.2.1 uri 之 map-base 取模法

在 listen 或 backend 区域配置

```
mode http
balance uri
```

3.2.2 uri 之一致 hash 法

在 listen 或 backend 区域配置

```
mode http
balance uri
hash-type consistent
```

3.3 url_param

url_param 对用户请求的 url 中的 params 部分中的参数 name（key）作 hash 计算，并由服务器总权重相除以后派发至某挑出的服务器。
url = http://www.magedu.com/foo/bar/index.php?k1=v1&k2=v2
所以：
host = "www.magedu.com"
url_param = "k1=v1&k2=v2"

3.3.1 uri 之 map-base 取模法

在 listen 或 backend 区域配置

curl http://192.168.7.101/app/index.html?name=NAME #单个参数访问

```
mode http
balance url_param name
```

curl http://192.168.7.101/app/index.html?age=AGE

```
mode http
balance url_param age
```

curl http://192.168.7.101/app/index.html?age=AGE&&name=NAME #多个参数访问

```
mode http
balance url_param name,age
```

3.3.2 uri 之一致 hash 法

在 listen 或 backend 区域配置

curl http://192.168.7.101/app/index.html?name=NAME #单个参数访问

```
mode http
balance url_param name
hash-type consistent
```

curl http://192.168.7.101/app/index.html?age=AGE

```
mode http
balance url_param age
hash-type consistent
```

curl http://192.168.7.101/app/index.html?age=AGE&&name=NAME #多个参数访问

```
mode http
balance url_param name,age
hash-type consistent
```

3.4 hdr

针对用户每个 http 头部（header）请求中的指定信息做 hash，此处由 name 指定的 http首部将会被取出并做 hash 计算，然后由服务器总权重相除以后派发至某挑出的服务器，假如无有效的值，则会使用默认的轮询调度。

3.4.1 hdr 之 map-base 取模法

在 listen 或 backend 区域配置

```
mode http
balance hdr(User-Agent)
```

举例：对 header 里的 User-Agent 字段（浏览器类型）做 hash 计算，同一个浏览器访问后端同一个服务器。

3.4.2 hdr 之一致 hash 法

在 listen 或 backend 区域配置

```
mode http
balance hdr(User-Agent)
hash-type consistent
```

3.5 rdp-cookie

服务器端的 rdp-cookie 使用客户端的 cookie 保持会话，可以实现对 windows 远程桌面（一般后端服务器为 windows server）的负载等。

3.5.1 rdp-cookie 之 map-base 取模法

在 listen 或 backend 区域配置

```
listen RDP
bind 10.0.0.17:3389
mode tcp
balance rdp-cookie
persist rdp-cookie          #启用基于 rdp-cookie 的持久连接，同一个 cookie 就转发给对应的服务器
tcp-request inspect-delay 5s   #设置内容检查期间等待数据的最大允许时间
tcp-request context accept if RDP_COOKIE   #匹配 RDP_COOKIE 存在就接受请求
server windows-server1 10.0.0.13:3389 check fall 3 rise 5 inter 2000

 # cookie 插入：HAProxy 自己设置一个 cookie：
 cookie SERVERID insert indirect nocache
 [...]
 server s1 10.0.0.1:80 check cookie s1
 server s2 10.0.0.2:80 check cookie s2

 # Cookie 前缀：HAProxy 使用现有的 cookie（通常是应用程序），并在其值前缀中添加服务器名称：
 backend mybk
 [...]
 cookie ASP.NET_SessionId prefix nocache
 [...]
 server s1 10.0.0.1:80 check cookie s1
 server s2 10.0.0.2:80 check cookie s2

 # 粘性会话表：HAProxy 可以学习并使用应用程序 cookie，而无需修改它：
 backend mybk
 [...]
 stick-table type string len 64 size 100k expire 15m
 stick store-response res.cookie(ASP.NET_SessionId)
 stick match req.cookie(ASP.NET_SessionId)
 [...]
 server s1 10.0.0.1:80 check
 server s2 10.0.0.2:80 check
```

3.5.2 rdp-cookie 之一致 hash 法

在 listen 或 backend 区域配置

```
listen RDP
bind 10.0.0.17:3389
mode tcp
```

```
balance rdp-cookie
hash-type consistent
persist rdp-cookie          #启用基于 rdp-cookie 的持久连接，同一个 cookie 就转发给对应的服务器
tcp-request inspect-delay 5s   #设置内容检查期间等待数据的最大允许时间
tcp-request context accept if RDP_COOKIE   #匹配 RDP_COOKIE 存在就接受请求
server windows-server1 10.0.0.13:3389 check fall 3 rise 5 inter 2000
```

3.6 random

在 1.9 版本开始增加一个叫做 random 的负载平衡算法，其基于一个随机数作为一致性 hash 的 key，随机负载平衡对于大型服务器场或经常添加或删除服务器非常有用。

3.6.1 random 之 map-base 取模法

在 listen 或 backend 区域配置

```
listen web_host
bind 192.168.7.101:80
mode http/tcp
log global
balance random
server web1 192.168.7.103:80 weight 1 check inter 3000 fall 2 rise 5
server web2 192.168.7.104:80 weight 1 check inter 3000 fall 2 rise 5
```

3.6.2 random 之一致 hash 法

在 listen 或 backend 区域配置

```
listen web_host
bind 192.168.7.101:80
mode http/tcp
log global
balance random
hash-type consistent
server web1 192.168.7.103:80 weight 1 check inter 3000 fall 2 rise 5
server web2 192.168.7.104:80 weight 1 check inter 3000 fall 2 rise 5
```

四、使用 socat 动态调整 HAProxy 后端服务器权重

安装 socat

```
yum install socat -y
```

查看 socat 的帮助命令

```
echo "help" | socat stdio /var/lib/haproxy/haproxy.sock
```

常用命令

```
disable server : disable a server for maintenance (use 'set server' instead)
enable server  : enable a disabled server (use 'set server' instead)
get weight     : report a server's current weight
set weight     : change a server's weight (deprecated)
```

show info　: report information about the running process [desc|json|typed]*

查看 haproxy 信息
echo "show info" | socat stdio /var/lib/haproxy/haproxy.sock

haproxy 配置文件
listen lck-net-80
bind 10.0.0.17:80
mode http
option forwardfor
log global
server web1 10.0.0.13:80 weight 1 check inter 3000 fall 2 rise 5
server web2 10.0.0.14:80 weight 2 check inter 3000 fall 2 rise 5

使用 socat 查看 server 权重
[root@centos7 ~]# echo "get weight lck-net-80/web1" | socat stdio /var/lib/haproxy/haproxy.sock
1 (initial 1)
[root@centos7 ~]# echo "get weight lck-net-80/web2" | socat stdio /var/lib/haproxy/haproxy.sock
2 (initial 2)

使用 socat 修改 server 权重
[root@centos7 ~]# echo "set weight lck-net-80/web1 4" | socat stdio /var/lib/haproxy/haproxy.sock
[root@centos7 ~]# echo "get weight lck-net-80/web1" | socat stdio /var/lib/haproxy/haproxy.sock
4 (initial 1)

测试修改权重后的访问结果

四、十种算法总结以及使用场景

静态：
static-rr--------->tcp/http　#做 session 共享的 web 集群
first------------->tcp/http　#使用较少

动态：
roundrobin-------->tcp/http
leastconn--------->tcp/http　#数据库
random------------>tcp/http
取决于是否有 hash_type consistent，有为动态，默认静态：
source------------>tcp/http　#基于客户端公网IP 的会话保持
uri--------------->http　　#缓存服务器，CDN 服务商，蓝汛、百度、阿里云、腾讯

url_param--------->http
hdr--------------->http　　#基于客户端请求报文头部做下一步处理
rdp-cookie-------->tcp　　#很少使用
uri、url_param、hdr 只有在 mode 为 http 的情况下才能取到值，如果 mode 为 tcp，balance 自动更改为 roundrobin。
动态与静态最主要区别：可以用 socat 动态调整权重、支持慢启动功能。

五、IP 透传

5.1 TCP 负载 IP 透传

负载均衡 haproxy 配置

```
listen web_prot_http_nodes
bind 192.168.7.101:80
mode tcp
balance roundrobin
server web1 blogs.studylinux.net:80 send-proxy check inter 3000 fall 3 rise 5
```

后端服务器 Nginx 配置

```
server {
listen 80 proxy_protocol;
server_name blogs.studylinux.net;
```

PROXY 协议允许 Nginx 和 Nginx Plus 接受来自代理服务器和负载平衡器的客户端连接信息，比如 HAproxy 和 Amazon Elastic Load Balancer (ELB) 。
通过 PROXY 协议，Nginx 可以从 HTTP，SSL，HTTP / 2，SPDY，WebSocket 和 TCP 中获取到源 IP 地址。获取到客户端的源 IP 地址，可以在为网页指定语言、设置 IP 黑名单或只是简单的日志和统计分析。
通过 PROXY 协议传输的数据是客户端的 IP 地址、代理服务器的 IP 地址和所有的端口号。

5.2 HTTP 负载 IP 透传

负载均衡 haproxy 配置

```
defaults
option forwardfor
或者：
option forwardfor header X-Forwarded-xxx
```

#option forwardfor 则后端服务器 web 格式为 X-Forwarded-For
#自定义传递 IP 参数,后端 web 服务器写 X-Forwarded-xxx，

5.3 web 服务器日志格式配置

```
apache
LogFormat "%{X-Forwarded-For}i %a %l %u %t "%r" %>s %b "%{Referer}i" "%{User-Agent}i"" combined

tomcat
pattern='%{X-Forwarded-For}i %l %T %t "%r" %s %b "%{User-Agent}i"'/>
```

```
nginx
log_format main '"$http_x_forwarded_For" - $remote_user [$time_local] "$request"
```