LVS—负载均衡集群

零、LVS 集群常用术语：

Director：复制调度集群的主机；

VIP：Virtual IP，向外提供服务的 IP；

RIP：Real IP，内部真实提供服务的主机 IP；

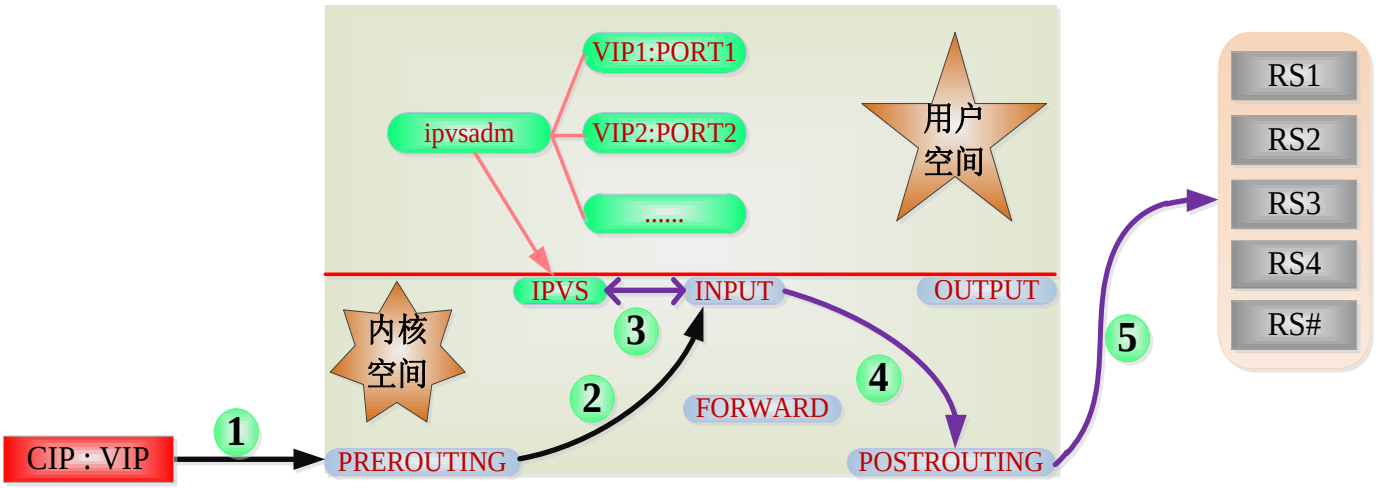
DIP：Director 主机上，向内部的 IP 通信的 IP；

CIP：客户端 IP。

一、LVS 的内核模型

LVS（Linux Virtual Server），是一个虚拟的服务器集群系统。本项目在 1998 年 5 月由章文嵩博士成立，是中国国内最早出现的自由软件项目之一。LVS 由前端的负载均衡器(Load Balancer，LB)和后端的真实服务器(Real Server，RS)群组成。RS 间可通过局域网或广域网连接。LVS 的这种结构对用户是透明的，用户只能看见一台作为 LB 的虚拟服务器(Virtual Server)，而看不到提供服务的 RS 群。当用户的请求发往虚拟服务器时，LB 根据设定的包转发策略和负载均衡调度算法将用户请求转发给 RS。RS 再将用户请求结果返回给用户。

LVS 内核模型



1、当客户端的请求到达负载均衡器的内核空间时，首先会到达 PREROUTING 链；

2、当内核发现请求数据包的目的地址是本机时，会将数据包送往 INPUT 链；

3、LVS 由用户空间的 ipvsadm 和内核空间的 IPVS 组成，ipvsadm 用来定义规则，ipvs 利用 ipvsadm 定义的规则工作，IPVS 工作在 INPUT 链上， 当数据包到达 INPUT 链时，首先会被 IPVS 检查，如果数据包里面的目标地址及端口没有在规则里面，那么这条数据将被放行至用户空间。

4、如果数据包里面的目标地址及端口在规则里面，那么这条数据报文的目标地址为将被修改为事先定义的后端服务器地址，并送往 POSTROUTING 链。

5、最后经由 POSTROUTING 链发往后端服务器

二、LVS 的集群组成

LVS 采用三层结构, 主要组成部分有：

1、负载调度器(load balancer)：它是整个集群对外的前端机, 负责将客户的请求发送到一组服务器上执行, 而客户端认为服务使来自一个 IP 地址上的. 负责调度器有许多称呼, 例如 Virtual Server、Director、Dispatcher、Balancer 等；

2、服务器池(server pool)：是一组真正执行客户请求的服务器, 执行的服务有 WEB、MAIL、FTP 和 DNS 等；

3、共享存储(shared storage)：它为服务器池提供一个共享存储区, 这样很容易使得服务器池拥有相同的内容, 提供相同的服务。有以下几种存储载体：

(1)NAS（Network Attached Storage,文件服务器）访问接口是文件级别(NFS,SAMBA)

(2)SAN（Storage Area Network，网络存储区域）访问接口是块级别；SCSI 协议借助于其他网络技术(FC,以太网)，例如 NFS、FTP 等

(3)DS（Distributed Storage 分布式存储）访问接口通常是文件级别，接口可是文件系统，也可以 API；ceph，内核级分布式存储；

数据结构：结构化数据：存储于 SQL 数据库中；

半结构化数据：xml,json,存储于文件系统或 NoSQL；

非结构化数据：文件系统，DS；

数据同步方案：rsync+inotify

三、LVS 的的类型：

LVS 有三种模型分别是 LVS-NAT、LVS-DR、LVS-TUN、LVS-FULLNAT。

1、LVS—NAT

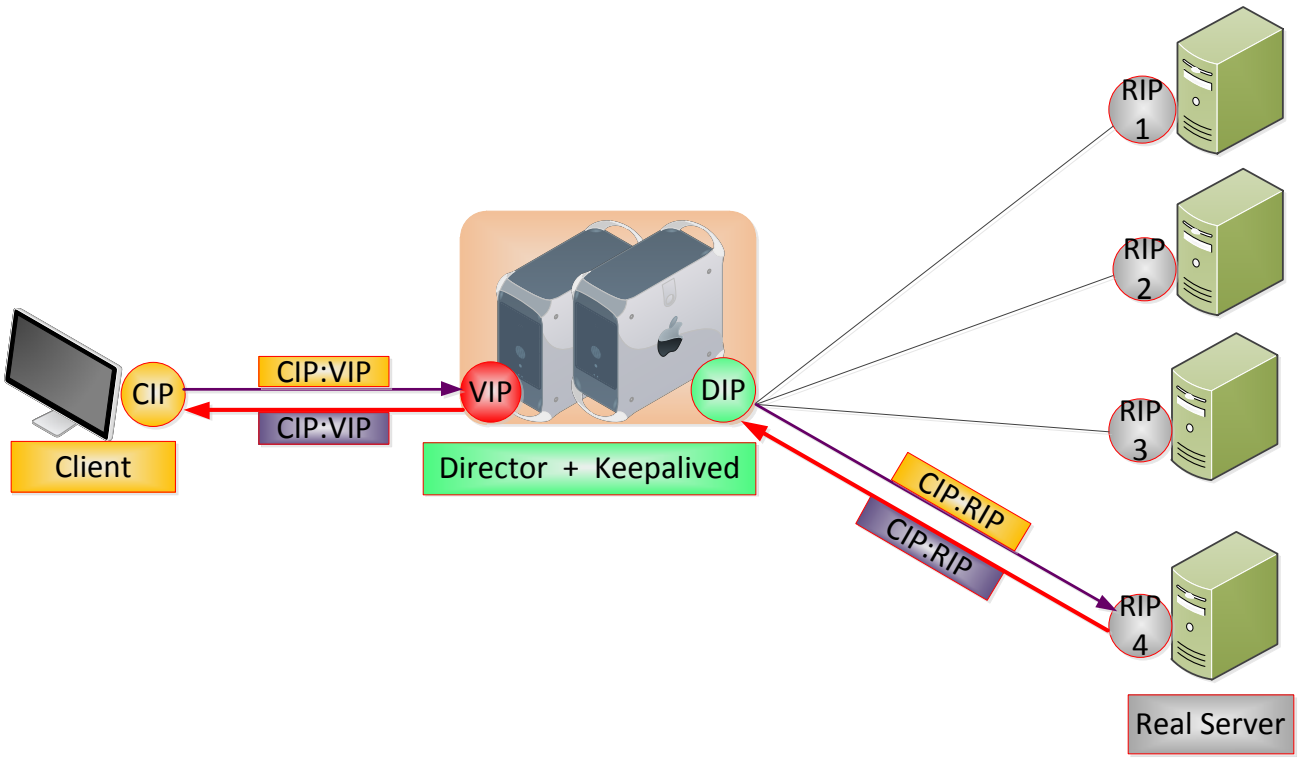
模型实质：多目标 IP 的 DNAT, 通过将请求报文中的目标地址和目标端口修改为某挑选出的 RS 的 RID 和 PORT 实现转发，实现前提：

(1)RIP 和 DIP 必须在同一个 IP 网络, 且应该使用私网地址；

(2)RS 的网关要指向 DIP；

(3)请求报文和响应报文都必须经由 Director 转发；

(4)VS 必须是 linux 系统， RS 可以是任意系统。



LVS-NAT 的实现流程：

(1)客户端 client 将请求发往前段的负责均衡器，请求报文源 IP 是 CIP 目标 IP 为 VIP

(2)负载均衡器收到报文后，发现请求的是在规则里面存在的地址，那么它就将请求报文的目标 IP 改为后端服务器的 RIP 地址并将报文根据算法发送出去;

(3)报文送达 Real Server 后，由于报文的目标 IP 是自己，所以会响应请求，并将响应报文返还给 LVS;

(4)然后 LVS 将此报文的源 IP 修改为本机并发送给客户端。

LVS—NAT 的优缺点：

优点：可以把用户访问的端口，映射到后端不同的端口；对后端服务器无要求；只有负载均衡器需要一个合法的 IP 地址。

缺点：请求和响应报文都得经过 Director，在高负载场景中，Director 很可能成为性能瓶颈。

2、LVS—DR

LVS-DR 实现前提：

(1)保证前端路由器将目标地址为 VIP 的报文通过 ARP 解析后送往 Director。有以下三种方法：

a、静态绑定：在前端路由将 VIP 对应的目标 MAC 地址静态配置为 Director VIP 接口的 MAC 地址；

b、arptables：在各 Realserver 上，通过 arptables 规则拒绝响应对 VIP 的 ARP 广播请求；

c、修改内核参数：在 Realserver 上修改内核参数，限制 arp 通告及应答级别(最常用)，修改参数的含义：限制响应级别：/proc/sys/net/ipv4/conf/all/arp_ignore

0<——>使用本地任意接口上配置的地址进行响应；

1<——>仅在请求的目标 IP 配置在本地主机的接收报文的接口上时才给予响应

限制通告级别：/proc/sys/net/ipv4/conf/all/arp_announce

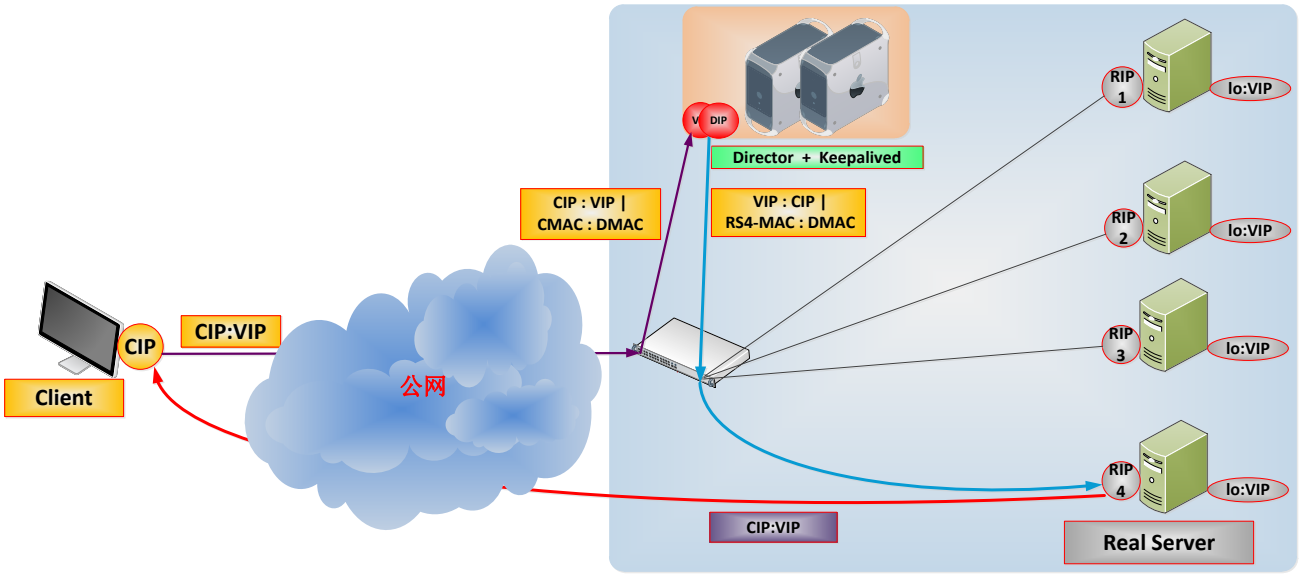
0<——>默认，把本机所有接口信息向每个接口通告

1<——>尽量避免向非本网络通告；

2<——>总是避免（DR 需采用）

(2)各 RIP 必须与 DIP 在同一个物理网络中;

(3)RS 的 RIP 可以使用私有地址，也可以使用公网地址，Realserver 不能将网关指向 DIP。



LVS—DR 实现流程：

(1)客户端将请求发往前端的负载均衡器，请求报文源地址是 CIP，目标地址为 VIP；

(2)负载均衡器收到报文后，发现请求的是在 IPVS 规则里面存在的地址，那么它将客户端请求报文的源 MAC 地址改为 DIP 的 MAC 地址，目标 MAC 改为 RIP 的 MAC 地址，并将此包发送给算法选中的 RS；

(3)RS 发现请求报文中的目标 MAC 是该主机的，就会将此次报文接收下来，处理完请求报文后，将响应报文通过 lo:1 接口送给 eth0 网卡直接发送给客户端。

LVS—NAT 的优缺点：

优点(特性)：负载均衡器只是分发请求，应答包通过单独的路由方法返回给客户端，大大减

轻了 Director 的压力，与 VS-TUN 相比，VS-DR 这种实现方式不需要隧道结构，因此可以使用大多数操作系统做为物理服务器；RIP 可以使用私有地址，也可以使用公网地址；每个 Real Server 上都有两个 IP：VIP 和 RIP，VIP 是隐藏的，不会接收请求，用来做请求响应的源 IP，Director 上只需要一个网卡利用别名配置两个 IP：VIP 和 DIP

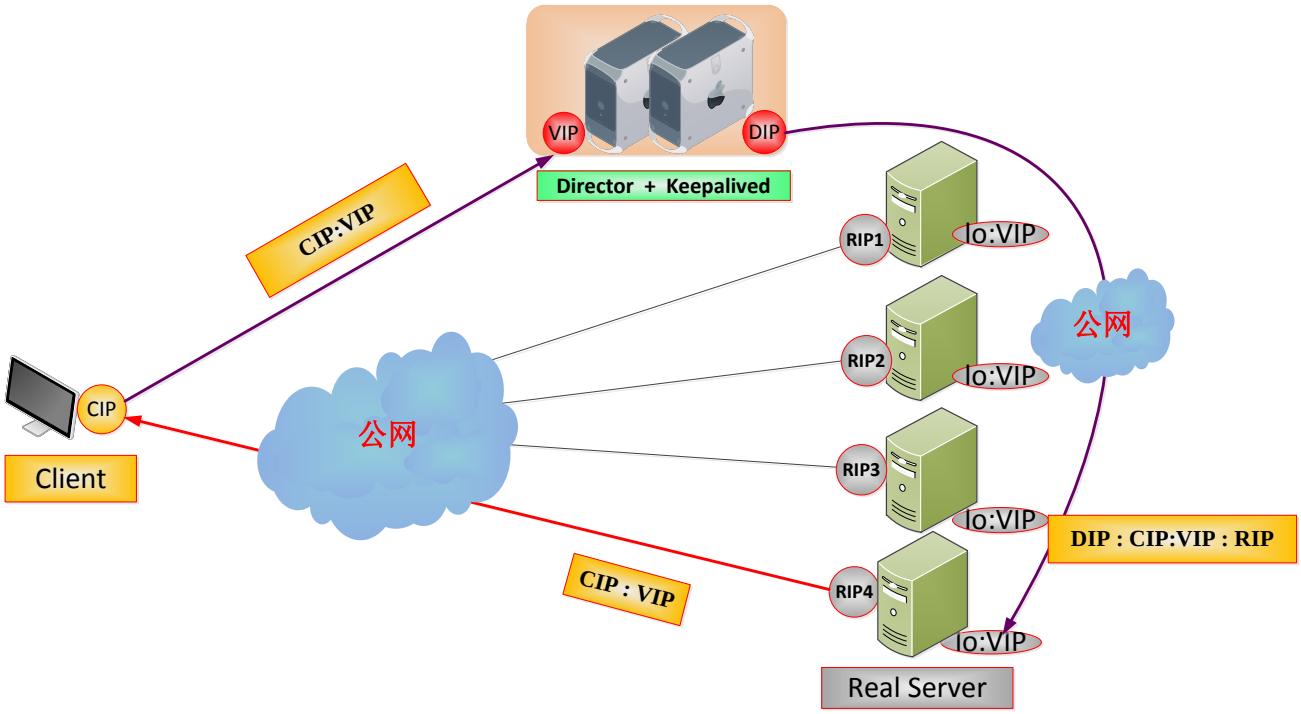
缺点：要求负载均衡器的网卡必须与物理网卡在一个物理段上。

3、LVS—TUN

LVS—TUN 实质是：在原 IP 报文之外再封装一个 IP 首部（源 IP 是 DIP，目标 IP 是 RIP），将报文发往挑选出的目标 RS； RS 接收到请求以后，先拆除第一层封装后拆除第二层封装，然后把响应数据直接传输给 Client。实现前提：

(1)Director 的 VIP 和 RIP 必须为公网 IP;且仅处理入站请求，响应报文由 RS 直接发往客户端；

(2)Real Server 的网关不能指向 Director，并且需支持隧道协议。



LVS—TUN 实现流程

(1)客户端将请求发往前端的负载均衡器，请求报文源地址为 CIP,目标地址为 VIP

(2)负载均衡器收到报文后，发现请求是规则里面存在的地址，那么它将在客户端请求报文的首部再封装一层 IP 报文，将源地址改为 DIP,目标地址改为 RIP，并将此包发送给 RS

(3)RS 收到请求包文后，首先会拆开第一层封装，然后发现里面还有一层 IP 首部的目标地址是自己的 lo:1 接口上的 VIP ,所以会处理此请求报文，并将响应报文通过 lo:1 接口送给 eth0 网卡直接送给客户端。

LVS—TUN 的优缺点

优点：负载均衡器只负责将请求包分发给后端节点服务器，而 RS 将应答包直接发给用户。减少了负载均衡器的大量数据流动，负载均衡器不再是系统的瓶颈，可处理很巨大的请求量。一台负载均衡器能够为很多 RS 进行分发。而且在公网上进行不同地域的分发。

缺点：隧道模式的 RS 节点需要合法 IP，这种方式需要所有的服务器支持”IP Tunneling”(IP Encapsulation)协议，服务器可能只局限在部分 Linux 系统上。

4、LVS-FUNT 模型

通过同时修改请求报文的源 IP 地址(CIP–>DIP)和目标 IP 地址(VIP–>RIP)进行转发；特点：

(1)VIP 是公网地址，RIP 和 DIP 是私网地址，且通常不在同一网络中，但需经由路由器互通；

(2)RS 收到的请求报文源 IP 为 DIP，因此响应报文将直接响应给 DIP;

(3)请求和响应报文都经由 Director；

(4)支持端口映射。

三、LVS 集群调度算法

根据其调度时是否考虑后端主机的当前负载，可分为静态和动态方法两类。

1、静态方法

只根据算法进行调度，而不考虑后端主机的实际连接情况和负载情况，可分以下四种算法：

(1) **RR** (Round Robin, 轮询/轮叫/轮调)

调度器通过”轮叫”调度算法将外部请求按顺序轮流分配到集群中的真实服务器上，它均等地对待每一台服务器，而不管服务器上实际的连接数和系统负载

(2) **WRR**(Weight RR, 加权轮询)

调度器通过“加权轮叫”调度算法根据真实服务器的不同处理能力来调度访问请求。这样可以保证处理能力强的服务器处理更多的访问流量。调度器可以自动问询真实服务器的负载情况，并动态地调整其权值。

(3) **DH**(Destination Hashing, 目标地址哈希)

和 SH 类似，DH 将请求的目标地址进行哈希，将相同目标 IP 的请求发送至同一主机。当 Realserver 为透明代理缓存服务器时，提高缓存的命中率。

(4) **SH**(Source Hash, 源地址哈希)

源地址哈希，对客户端地址进行哈希计算，保存在 Director 的哈希表中，一段时间内，同一个客户端 IP 地址的请求会被调度至相同的 Realserver。实现 session affinity（会话绑定），一定程度上损害了负载均衡的效果。

2、动态方法：根据算法及各 RS 当前的负载状态进行调度

(1)**LC**(Least Connections，最少链接)

最少连接，根据 overhead = active*256 + inactive 计算负载状态，每次选择 overhead 最小的服务器。

(2)**WLC**(Weighted Least Connections，加权最少连接)系统默认采用

加权最少连接，根据 overhead = (active*256+inactive)/weight 来计算负载，每次选择 overhead 最小的服务器。

(3)**SED**(Shortest Expected Delay，最短延迟调度)

最短期望延迟，不对 inactive 状态的连接进行计算，根据 overhead = (active+1)*256/weight 计算负载，选择 overhead 最小的服务器进行调度。非活动连接过多缺陷：当权限过大的时候，会倒置空闲服务器一直处于无连接状态。

(4)**NQ**(Never Queue Scheduling NQ，永不排队/最少队列调度)

无需队列。如果有 realserver 的连接数为 0 就直接分配过去，不需要再进行 sed 运算，保证不会有一个主机无连接。在 SED 基础上无论加多少，第二次一定给下一个，保证不会有一个主机无连接，不考虑非活动连接，才用 NQ，SED 要考虑活动状态连接，对于 DNS 的 UDP 不需要考虑非活动连接，而 httpd 的处于保持状态的服务就需要考虑非活动连接给服务器的压力。

(5)**LBLC**(locality-Based Least Connections，基于局部性的最少链接)

基于局部性的最少链接”调度算法是针对目标 IP 地址的负载均衡，目前主要用于 Cache 集群系统。基于本地的最少连接，相当于 dh + wlc，正常请求下使用 dh 算法进行调度，如果服务器超载，则使用 wlc 算法调度至其他服务器。

(6)**LBLCR**(Locality-Based Least Connections with Replication，带复制的基于局部性最少连接)

基于本地的带复制功能的 LBLC，判断后端连接数，当 A 的连接很多，而 B 的很空闲，会将 A 的部分连接分配到 B 上，避免大范围不公平。主要用于 Cache 集群系统

参考网址：http://www.178linux.com/61462

http://www.178linux.com/56672

四、ipvsadm/ipvs 管理工具的使用

1、集群服务管理(调度器)

```
##       Usage: ipvsadm -A|E -t|u|f  service-address  [-s scheduler]  [-p [timeout]]
##参数释义：        -A|E VIP                                  #添加|修改服务地址
                  -D -t|u|f VIP                             #删除集群
                  -t|u|f 类型                               #tcp 协议 | udp 协议 | 防火墙标记
                  -s                                        #scheduler(调度)指定集群调度算法，默认 wlc
```

2、RS 管理：

```
##       Usage: ipvsadm -a|e -t|u|f VIP -r RIP [-g|i|m] [-w weight]
##参数释义：
                  -a|e                                  #添加|修改 RS
                  -g|i|m                                # lvs 模型： DR | TUN | NAT，默认 DR
                  -w                                    #指定权重(weight)
##其他用法：
         ipvsadm -C        #清空定义
         ipvsadm -ln --excat -c,--connection  --stats  --rate#查看
         ipvsadm -S = ipvsadm-save          #保存
         ipvsadm -R = ipvsadm-restore       #重载
```

3、示例：

```
ipvsadm -A -t 10.1.235.55:80 -s wrr                    #添加一个调度器，采用默认 DR 模型，加权论调算法，
ipvsadm -a -t 10.1.235.55:80 -r 10.1.235.6 -g -w 2   #在添加的调度上添加 RS，指定 IP，权重为 2
ipvsadm -a -t 10.1.235.55:80 -r 10.1.235.7 -g -w 3   #在添加的调度上添加 RS，指定 IP，权重为 3
```

4、附加：防火墙标记、持久连接

Firewall Mark，借助防火墙标记来分类报文，而后基于标记定义集群服务，可将多个不同的应用使用同一个集群服务进行调度：

打标记方法（在 Director 主机）：

```
Usage: iptables -t mangle -A PREROUTING -d $vip -p $proto -dport $port -j MARK -set-mark NUM
##基于标记定义集群服务：
Usage: ipvsadm -A -f NUM [options]
```

lvs persistence：持久连接

持久连接，实现无论使用任何算法，在一段时间内，实现将来自同一个地址的请求始终发往同一个 RS；

```
Usage: ipvsadm -A|E -t|u|f service-address:port [-s scheduler] [-p [timeout]]
##注：基于 0 端口(所有端口)定义集群服务，即将客户端对所有应用的请求统统调度至后端主机，而且可使用持久连接进行绑定；
```