1. IPVS

负载均衡技术有很多实现方案，有基于DNS域名轮流解析的方法、有基于客户端调度访问的方法、有基于应用层系统负载的调度方法，还有基于IP地址的调度方法。

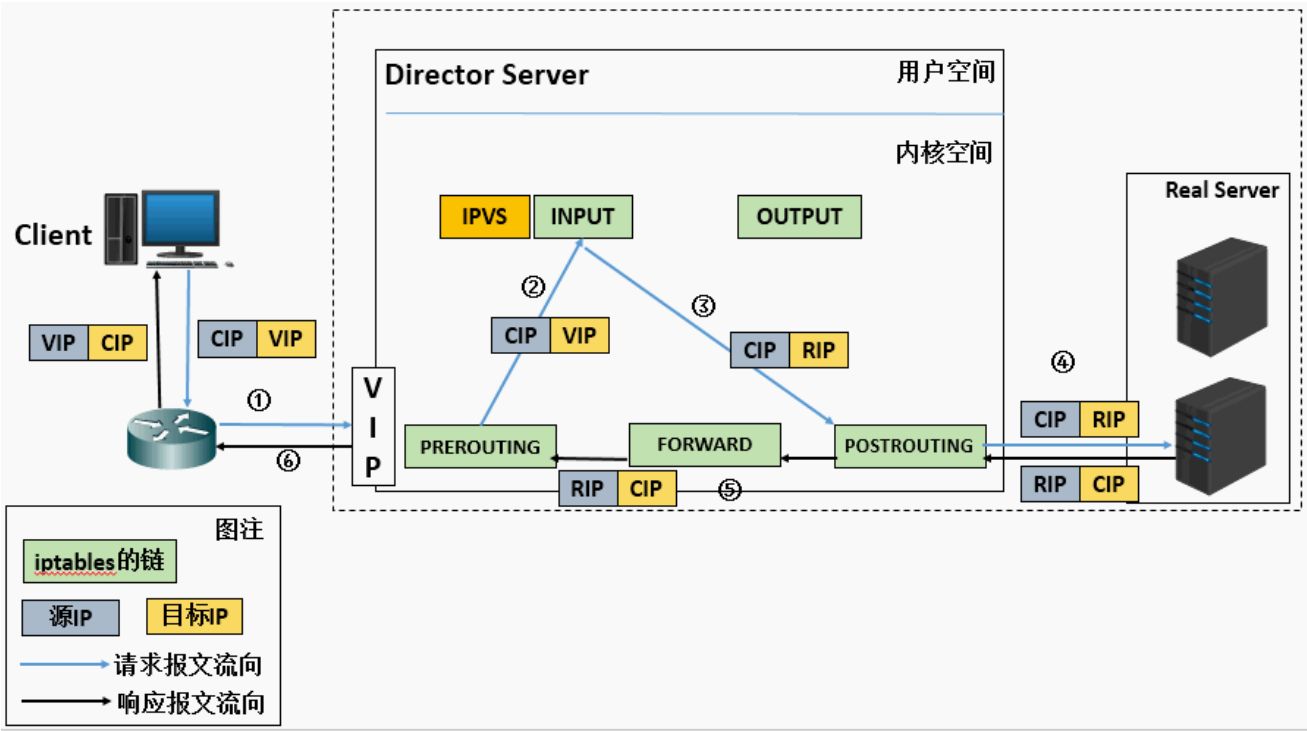
IPVS实现的IP负载均衡机制有以下三种：

1.1 工作模式

1.1.1 NAT

NAT(Network Address Translation)方式进出入流量都需要经过调度器，会成为性能瓶颈。

- RS使用私有地址，网关指向DIP
- DIP和RIP必须在同一网段内。
- 支持端口映射。

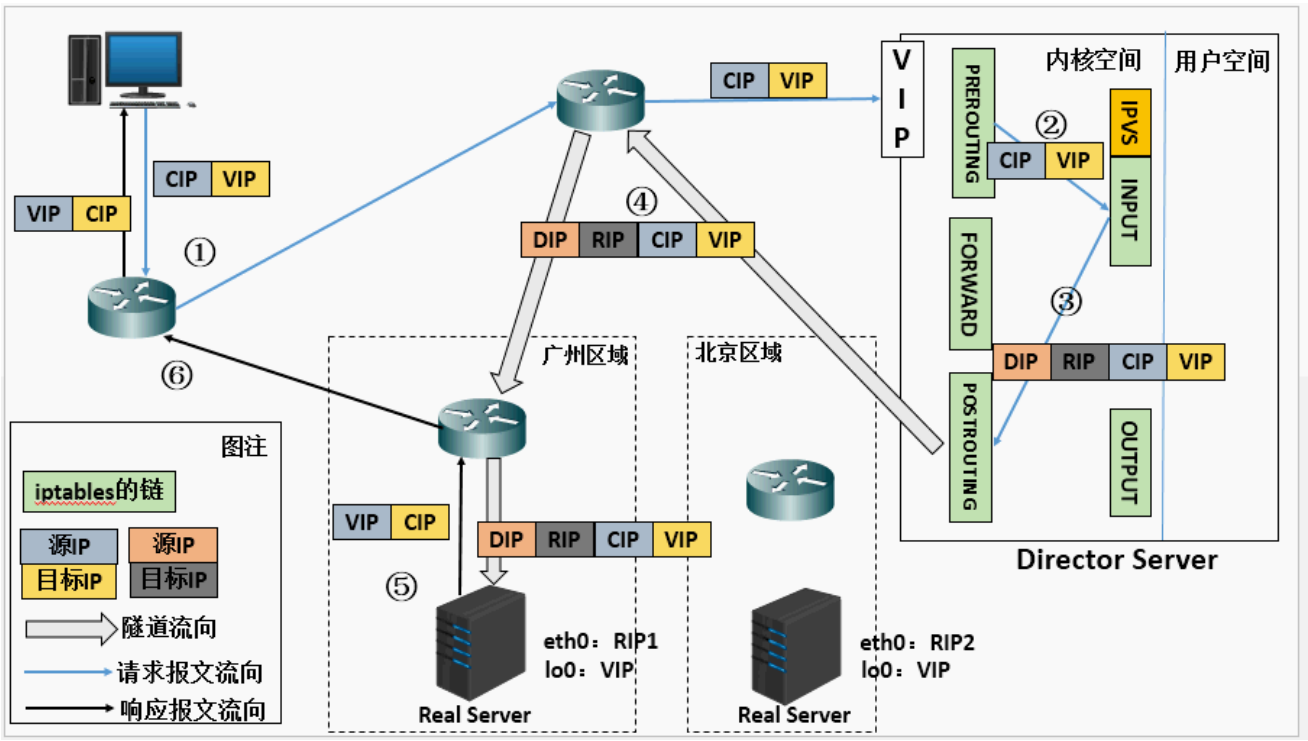


1. 当用户请求到达Director Server，此时请求的数据报文会先到内核空间的PREROUTING链。 此时报文的源IP为CIP，目标IP为VIP
2. PREROUTING检查发现数据包的目标IP是本机，将数据包送至INPUT链
3. IPVS比对数据包请求的服务是否为集群服务，若是，修改数据包的目标IP地址为后端服务器IP，然后将数据包发至POSTROUTING链。 此时报文的源IP为CIP，目标IP为RIP
4. POSTROUTING链通过选路，将数据包发送给Real Server
5. Real Server比对发现目标为自己的IP，开始构建响应报文发回给Director Server。 此时报文的源IP为RIP，目标IP为CIP
6. Director Server在响应客户端前，此时会将源IP地址修改为自己的VIP地址，然后响应给客户端。 此时报文的源IP为VIP，目标IP为CIP

1.1.2 TUN

TUN(IP Tunneling)在原报文上封装一层IP首部。

- RIP、VIP、DIP全是公网地址
- RS的网关不会也不可能指向DIP
- 不支持端口映射

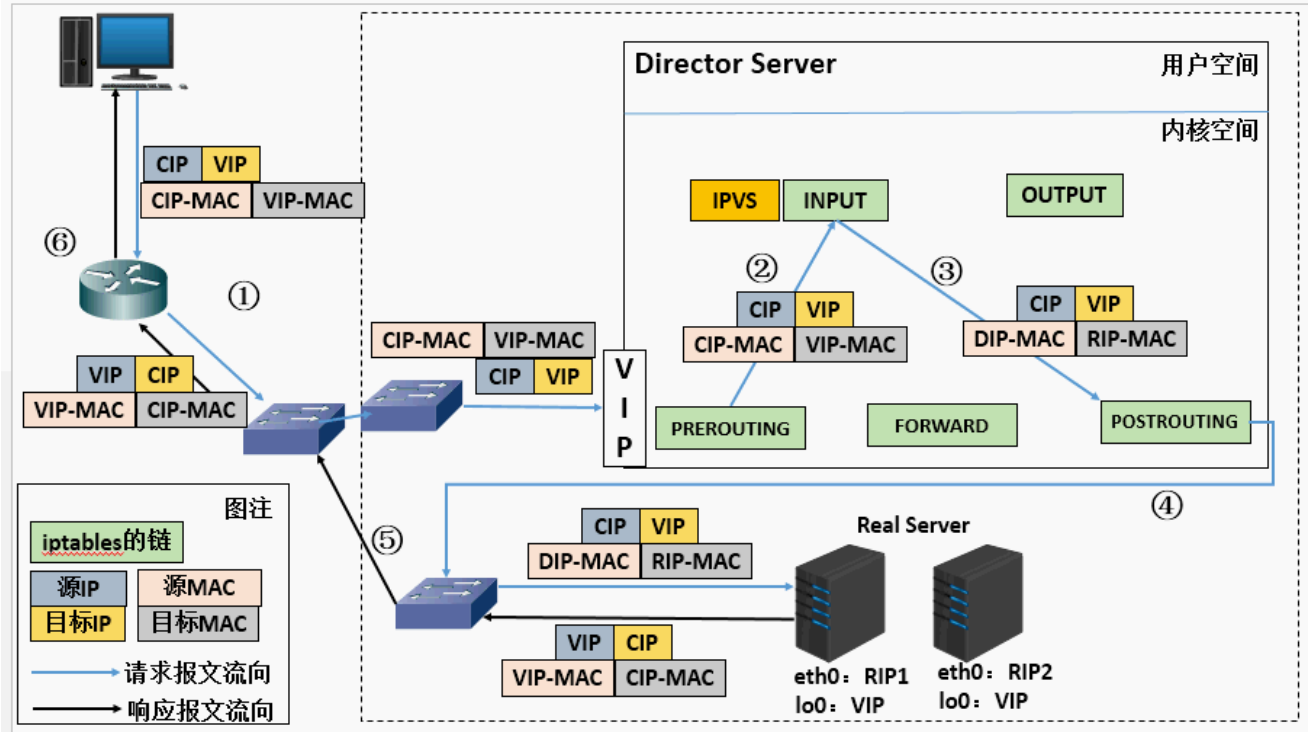


1. 当用户请求到达Director Server，此时请求的数据报文会先到内核空间的PREROUTING链。 此时报文的源IP为CIP，目标IP为VIP 。
2. PREROUTING检查发现数据包的目标IP是本机，将数据包送至INPUT链
3. IPVS比对数据包请求的服务是否为集群服务，若是，在请求报文的首部再次封装一层IP报文，封装源IP为DIP，目标IP为RIP。然后发至POSTROUTING链。 此时源IP为DIP，目标IP为RIP
4. POSTROUTING链根据最新封装的IP报文，将数据包发至RS（因为在外层封装多了一层IP首部，所以可以理解为此时通过隧道传输）。 此时源IP为DIP，目标IP为RIP
5. RS接收到报文后发现是自己的IP地址，就将报文接收下来，拆除掉最外层的IP后，会发现里面还有一层IP首部，而且目标是自己的lo接口VIP，那么此时RS开始处理此请求，处理完成之后，通过lo接口送给eth0网卡，然后向外传递。 此时的源IP地址为VIP，目标IP为CIP
6. 响应报文最终送达至客户端

1.1.3 DR

DR(Direct Routing)通过修改MAC地址实现转发。

- RS可以使用私有地址；也可以是公网地址.
- RS跟Director Server必须在同一个物理网络中
- 不支持地址转换，也不支持端口映射
- RS的网关绝不允许指向DIP
- RS上的lo接口配置VIP的IP地址



1. 当用户请求到达Director Server，此时请求的数据报文会先到内核空间的PREROUTING链。 此时报文的源IP为CIP，目标IP为VIP
2. PREROUTING检查发现数据包的目标IP是本机，将数据包送至INPUT链
3. IPVS比对数据包请求的服务是否为集群服务，若是，将请求报文中的源MAC地址修改为DIP的MAC地址，将目标MAC地址修改RIP的MAC地址，然后将数据包发至POSTROUTING链。 此时的源IP和目的IP均未修改，仅修改了源MAC地址为DIP的MAC地址，目标MAC地址为RIP的MAC地址
4. 由于DS和RS在同一个网络中，所以是通过二层来传输。POSTROUTING链检查目标MAC地址为RIP的MAC地址，那么此时数据包将会发至Real Server。
5. RS发现请求报文的MAC地址是自己的MAC地址，就接收此报文。处理完成之后，将响应报文通过lo接口传送给eth0网卡然后向外发出。 此时的源IP地址为VIP，目标IP为CIP
6. 响应报文最终送达至客户端

1.2 调度算法

- 轮询调度（Round Robin）
  所有的请求平均分配给每个真实服务器，不管后端 RS 配置和处理能力
- 加权轮询调度（Weighted Round Robin）
  可以给 RS 设置权重，权重越高，分发的请求数越多，权重的取值范围 0 – 100。
- 最少链接调度（Least Connections）
  根据后端 RS 的连接数来决定分发请求。
- 加权最少链接调度（Weighted Least Connections）
  比 lc 多了一个权重的概念。
- 基于局部性的最少链接（Locality-Based Least Connections）
  根据请求的目标 IP 地址寻找最近的该目标 IP 地址所有使用的服务器，如果这台服务器依然可用，并且有能力处理该请求，调度器会尽量选择相同的服务器，否则会继续选择其它可行的服务器
- 复杂的基于局部性最少链接（Locality-Based Least Connections with Replication）
- 目标地址散列（Destination Hashing）
  根据目标 IP 地址通过hash将目标 IP 与服务器建立映射关系

- 源地址散列（Source Hashing）
  根据源地址散列算法进行静态分配固定的服务器资源
- 最短期望延迟(Shortest Expected Delay)
- 无须队列等待（Never Queue）

1.3 配置

1.3.1 NAT

- 环境

```
172.16.254.200 <--> 192.168.0.8  <--> 192.168.0.18
                                      192.168.0.28
```

- 配置

```
# echo 1 > /proc/sys/net/ipv4/ip_forward

# echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects
# echo 0 > /proc/sys/net/ipv4/conf/default/send_redirects
# echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
# echo 0 > /proc/sys/net/ipv4/conf/eth1/send_redirects
```

```
# iptables -t nat -F
# iptables -t nat -X
# iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -j MASQUERADE

# ipvsadm -C
# ipvsadm -A -t 172.16.254.200:80 -s wrr
# ipvsadm -a -t 172.16.254.200:80 -r 192.168.0.18:80 -m -w 1
# ipvsadm -a -t 172.16.254.200:80 -r 192.168.0.28:80 -m -w 1
```

1.3.2 DR

- 环境

```
192.168.0.8  vip eth0:0 192.168.0.38
                 <-->  192.168.0.18  vip lo:0 192.168.0.38
                 <-->  192.168.0.28  vip lo:0 192.168.0.38
```

- 配置
  - DS

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
# ifconfig eth0:0 down
# ifconfig eth0:0 192.168.0.38 broadcast 192.168.0.38 netmask 255.255.255.255 up
# route add -host 192.168.0.38 dev eth0:0
```

```
# ipvsadm -C
# ipvsadm -A -t 192.168.0.38:80 -s wrr
# ipvsadm -a -t 192.168.0.38:80 -r 192.168.0.18:80 -g -w 3
# ipvsadm -a -t 192.168.0.38:80 -r 192.168.0.28:80 -g -w 1
```

- RS

```
# ifconfig lo:0 192.168.0.38 broadcast 192.168.0.38 netmask 255.255.255.255 up
# route add -host 192.168.0.38 dev lo:0
```

```
# echo "1" >/proc/sys/net/ipv4/conf/lo/arp_ignore
# echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce
# echo "1" >/proc/sys/net/ipv4/conf/all/arp_ignore
# echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
```

2. ipset

ipset是iptables的扩展。iptables通过"-m set"扩展匹配模块，可以创建匹配整个集合的iptables规则，ipset的集合可以是ip(v4/v6)地址，(TCP/UDP)端口，IP/MAC对，IP/Port对等。

普通的iptables链中的规则是线性存储和过滤，当规则较多时，在线性链表中匹配指定的节点就会存在性能问题。而集合存储在带索引的数据结构中，这种结构即使集合比较大时也可以进行高效的查找。

2.1 ipset命令

创建集合：`# ipset create SETNAME TYPENAME [ CREATE-OPTIONS ]` 销毁集合：`# ipset destroy [ SETNAME ]` 查看集合：`# ipset list [ SETNAME ]` 添加ENTRY：`# ipset add SETNAME ADD-ENTRY [ ADD-OPTIONS ]` 删除ENTRY：`# ipset del SETNAME DEL-ENTRY [ DEL-OPTIONS ]`

- CREATE和AND选项
  - timeout 设置集合的有效期(in seconds)。向该集合添加entry后，entry默认有效期为timeout，过期该entry会被自动删除：

    ```
    # ipset create test hash:ip timeout 300
    # ipset add test 192.168.0.1 timeout 60
    # ipset -exist add test 192.168.0.1 timeout 600
    ```

  - coumters, packets, bytes

```
# ipset create foo hash:ip counters
# ipset add foo 192.168.1.1 packets 42 bytes 1024
```

- comment

```
# ipset add foo 1.1.1.1 comment "this comment is \"bad\""
```

- skbinfo, skbmark, skbprio, skbqueue 该扩展允许为每个 ENTRY 存储 metainfo(firewall mark, tc class and hardware queue)信息，并通过SET netfilter target的“--map-set”选项将其映射到数据包。skbmark选项格式为：MARK or MARK/MASK，MARK为32bit的十六进制数据 skbprio选项格式为：MAJOR:MINOR skbqueue选项为十进制数据

```
# ipset create foo hash:ip skbinfo
# ipset add foo skbmark 0x1111/0xff00ffff skbprio 1:10 skbqueue 10
```

- hashsize 该参数仅对hash类型有效，默认hash大小为1024，可以通过该参数指定创建集合时初始的hash size。

```
# ipset create test hash:ip hashsize 1536
```

- maxelem 该参数仅对hash类型有效，默认hash表中的元素为65536。可以通过该参数设置最大存储的条目。

```
# ipset create test hash:ip maxelem 2048
```

- family { inet | inet6 } 该参数仅对hash类型有效(除hash:mac)。

```
# ipset create test hash:ip family inet6
```

- nomatch 在匹配集合中的元素时，标记为不匹配的条目将被跳过，就好像这些条目未添加到集合中一样，可以用于构建具有例外的集合。
- forceadd 创建集合时如果指定该选项，当集合已满时，可以强制向集合中添加entry，并随机逐出一条entry。

```
# ipset create foo hash:ip forceadd
```

2.2 集合类型

集合类型由**数据的存**储**方式**和**存**储**在集合中的数据**类**型**两个部分构成。create命令中TYPENAME参数用于指定集合的类型，语法如下：

```
TYPENAME := method:datatype[,datatype[,datatype]]
```

当前，存储方式包含以下三种：`list`，`hash`，`bitmap`。数据类型包含以下几种：`ip`,`net`,`mac`,`port`,`iface`。集合的大小等于其TYPENAME中的数据类型的数量。bitmap和list类型使用固定大小的存储。hash类型使用散列来存储元素。

2.2.1 bitmap

bitmap:ip

```
# ipset create foo bitmap:ip range 192.168.0.0/16
# ipset add foo 192.168.1/24
# ipset test foo 192.168.1.1
```

bitmap:ip,mac

```
# ipset create foo bitmap:ip,mac range 192.168.0.0/16
# ipset add foo 192.168.1.1,12:34:56:78:9A:BC
# ipset test foo 192.168.1.1
```

bitmap:port

```
# ipset create foo bitmap:port range 0-1024
# ipset add foo 80
# ipset test foo 80
# ipset del foo udp:[macon-udp]-[tn-tl-w2]
```

2.2.2 hash

hash:ip

```
# ipset create foo hash:ip netmask 30
# ipset add foo 192.168.1.0/24
# ipset test foo 192.168.1.2
```

hash:mac

```
# ipset create foo hash:mac
# ipset add foo 01:02:03:04:05:06
# ipset test foo 01:02:03:04:05:06
```

hash:net

```
# ipset create foo hash:net
# ipset add foo 192.168.0.0/24
# ipset add foo 10.1.0.0/16
# ipset add foo 192.168.0/24
# ipset add foo 192.168.0/30 nomatch
```

hash:net,net

```
# ipset create foo hash:net,net
# ipset add foo 192.168.0.0/24,10.0.1.0/24
# ipset add foo 10.1.0.0/16,10.255.0.0/24
# ipset add foo 192.168.0/24,192.168.54.0-192.168.54.255
# ipset add foo 192.168.0/30,192.168.64/30 nomatch
```

hash:ip,port

```
# ipset create foo hash:ip,port
# ipset add foo 192.168.1.0/24,80-82
# ipset add foo 192.168.1.1,udp:53
# ipset add foo 192.168.1.1,vrrp:0
# ipset test foo 192.168.1.1,80
```

hash:net,port

```
# ipset create foo hash:net,port
# ipset add foo 192.168.0/24,25
# ipset add foo 10.1.0.0/16,80
# ipset test foo 192.168.0/24,25
```

hash:ip,port,ip

```
# ipset create foo hash:ip,port,ip
# ipset add foo 192.168.1.1,80,10.0.0.1
# ipset test foo 192.168.1.1,udp:53,10.0.0.1
```

hash:ip,port,net

```
# ipset create foo hash:ip,port,net
# ipset add foo 192.168.1,80,10.0.0/24
# ipset add foo 192.168.2,25,10.1.0.0/16
# ipset test foo 192.168.1,80.10.0.0/24
```

hash:ip,mark

```
# ipset create foo hash:ip,mark
# ipset add foo 192.168.1.0/24,555
# ipset add foo 192.168.1.1,0x63
# ipset add foo 192.168.1.1,111236
```

hash:net,port,net

```
# ipset create foo hash:net,port,net
# ipset add foo 192.168.1.0/24,0,10.0.0/24
# ipset add foo 192.168.2.0/24,25,10.1.0.0/16
# ipset test foo 192.168.1.1,80,10.0.0.1
```

hash:net,iface

```
# ipset create foo hash:net,iface
# ipset add foo 192.168.0/24,eth0
# ipset add foo 10.1.0.0/16,eth1
# ipset test foo 192.168.0/24,eth0
```

2.2.3 list

list:set

存储集合名称。

```
# iptables -m set --match-set a src,dst -j SET --add-set b src,dst
```