超详细！使用 LVS 实现负载均衡原理及安装配置详解

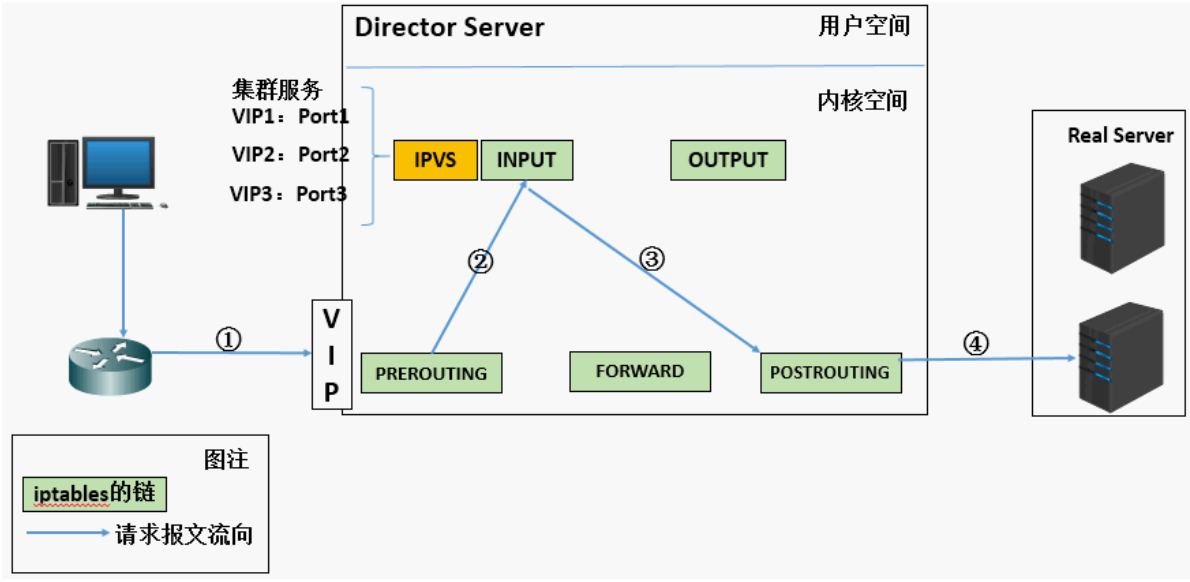
负载均衡集群是 load balance 集群的简写，翻译成中文就是负载均衡集群。常用的负载均衡开源软件有nginx、lvs、haproxy，商业的硬件负载均衡设备F5、Netscale。这里主要是学习 LVS 并对其进行了详细的总结记录。

一、负载均衡LVS基本介绍

LB集群的架构和原理很简单，就是当用户的请求过来时，会直接分发到Director Server上，然后它把用户的请求根据设置好的调度算法，智能均衡地分发到后端真正服务器(real server)上。为了避免不同机器上用户请求得到的数据不一样，需要用到了共享存储，这样保证所有用户请求的数据是一样的。

LVS是 Linux Virtual Server 的简称，也就是Linux虚拟服务器。这是一个由章文嵩博士发起的一个开源项目，它的官方网是 http://www.linuxvirtualserver.org 现在 LVS 已经是 Linux 内核标准的一部分。使用 LVS 可以达到的技术目标是：通过 LVS 达到的负载均衡技术和 Linux 操作系统实现一个高性能高可用的 Linux 服务器集群，它具有良好的可靠性、可扩展性和可操作性。从而以低廉的成本实现最优的性能。LVS 是一个实现负载均衡集群的开源软件项目，LVS架构从逻辑上可分为调度层、Server集群层和共享存储。

二、LVS的基本工作原理



1. 当用户向负载均衡调度器（Director Server）发起请求，调度器将请求发往至内核空间
2. PREROUTING链首先会接收到用户请求，判断目标IP确定是本机IP，将数据包发往INPUT链

3. IPVS是工作在INPUT链上的，当用户请求到达INPUT时，IPVS会将用户请求和自己已定义好的集群服务进行比对，如果用户请求的就是定义的集群服务，那么此时IPVS会强行修改数据包里的目标IP地址及端口，并将新的数据包发往POSTROUTING链
4. POSTROUTING链接收数据包后发现目标IP地址刚好是自己的后端服务器，那么此时通过选路，将数据包最终发送给后端的服务器

三、LVS的组成

LVS 由2部分程序组成，包括 ipvs 和 ipvsadm。

1.ipvs(ip virtual server)：一段代码工作在内核空间，叫ipvs，是真正生效实现调度的代码。

2.ipvsadm：另外一段是工作在用户空间，叫ipvsadm，负责为ipvs内核框架编写规则，定义谁是集群服务，而谁是后端真实的服务器(Real Server)

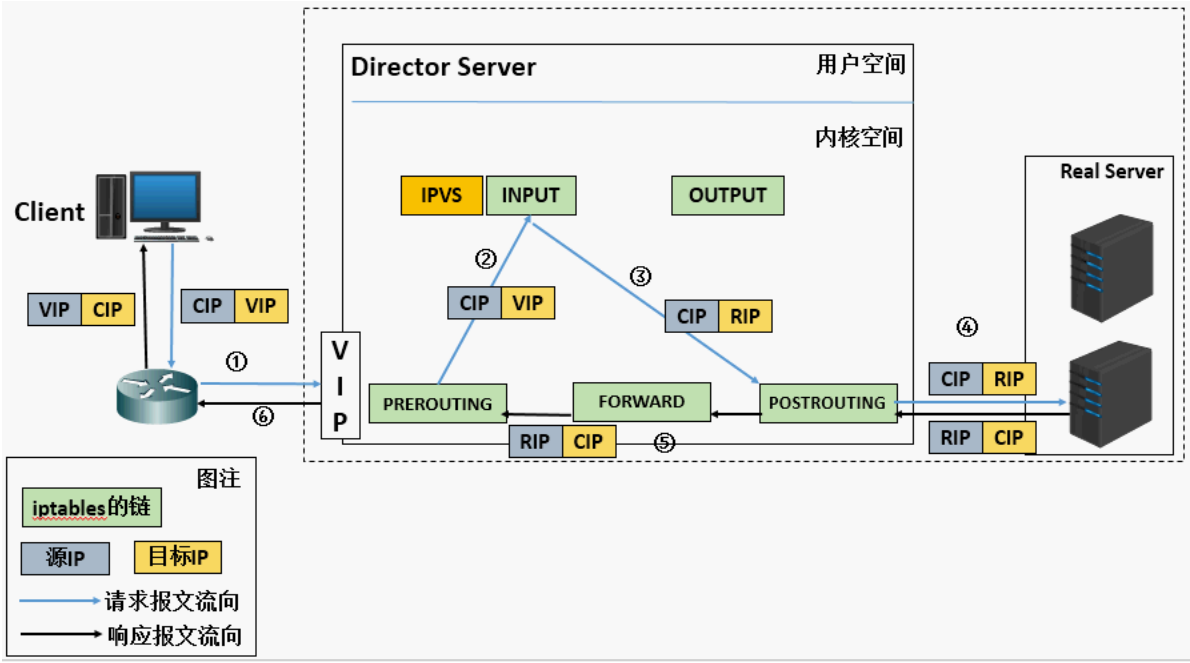
四、LVS相关术语

1. DS：Director Server。指的是前端负载均衡器节点。
2. RS：Real Server。后端真实的工作服务器。
3. VIP：向外部直接面向用户请求，作为用户请求的目标的IP地址。
4. DIP：Director Server IP，主要用于和内部主机通讯的IP地址。
5. RIP：Real Server IP，后端服务器的IP地址。
6. CIP：Client IP，访问客户端的IP地址

下边是三种工作模式的原理和特点总结。

五、LVS/NAT原理和特点

1. 重点理解NAT方式的实现原理和数据包的改变。



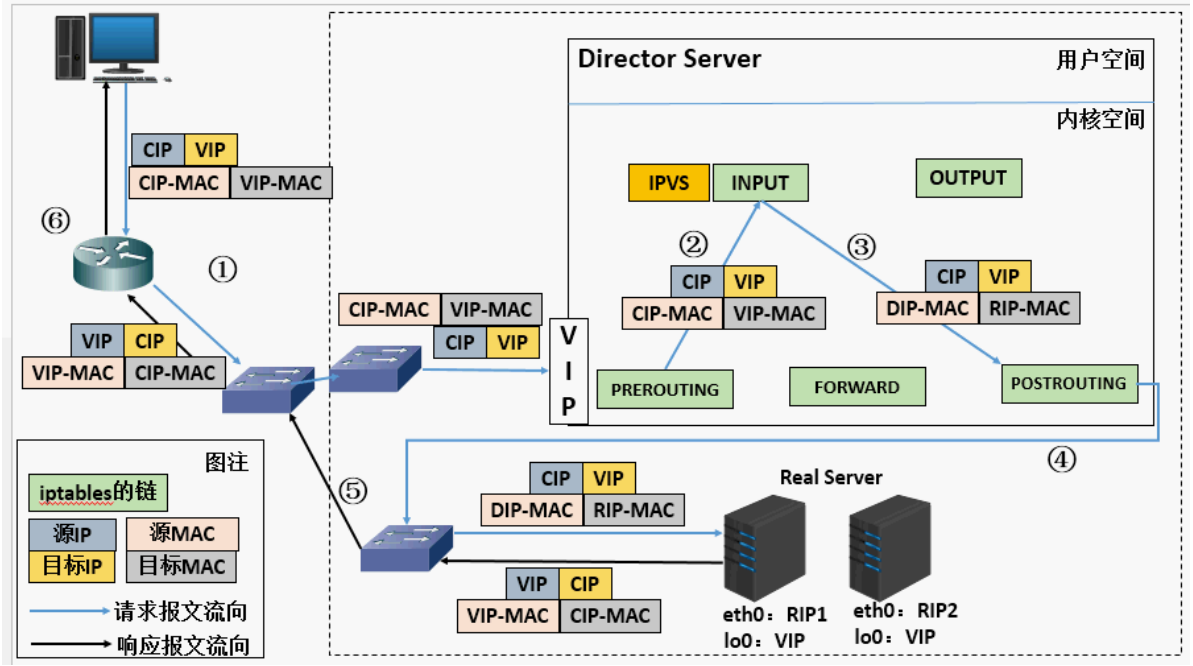
(a). 当用户请求到达Director Server，此时请求的数据报文会先到内核空间的PREROUTING链。 此时报文的源IP为CIP，目标IP为VIP
(b). PREROUTING检查发现数据包的目标IP是本机，将数据包送至INPUT链
(c). IPVS比对数据包请求的服务是否为集群服务，若是，修改数据包的目标IP地址为后端服务器IP，然后将数据包发至POSTROUTING链。 此时报文的源IP为CIP，目标IP为RIP
(d). POSTROUTING链通过选路，将数据包发送给Real Server
(e). Real Server比对发现目标为自己的IP，开始构建响应报文发回给Director Server。 此时报文的源IP为RIP，目标IP为CIP
(f). 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

2. LVS-NAT模型的特性

- RS应该使用私有地址，RS的网关必须指向DIP
- DIP和RIP必须在同一个网段内
- 请求和响应报文都需要经过Director Server，高负载场景中，Director Server易成为性能瓶颈
- 支持端口映射
- RS可以使用任意操作系统
- 缺陷：对Director Server压力会比较大，请求和响应都需经过director server

六、LVS/DR原理和特点

1.重将请求报文的目标MAC地址设定为挑选出的RS的MAC地址



(a) 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
(b) PREROUTING检查发现数据包的目标IP是本机，将数据包送至INPUT链
(c) 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
(d) 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。
(e) 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
(f) 响应报文最终送达至客户端

2. LVS-DR模型的特性

- 特点1：保证前端路由将目标地址为VIP报文统统发给Director Server，而不是RS
- RS可以使用私有地址；也可以是公网地址，如果使用公网地址，此时可以通过互联网对RIP进行直接访问
- RS跟Director Server必须在同一个物理网络中
- 所有的请求报文经由Director Server，但响应报文必须不能进过Director Server
- 不支持地址转换，也不支持端口映射
- RS可以是大多数常见的操作系统
- RS的网关绝不允许指向DIP(因为我们不允许他经过director)
- RS上的lo接口配置VIP的IP地址
- 缺陷：RS和DS必须在同一机房中

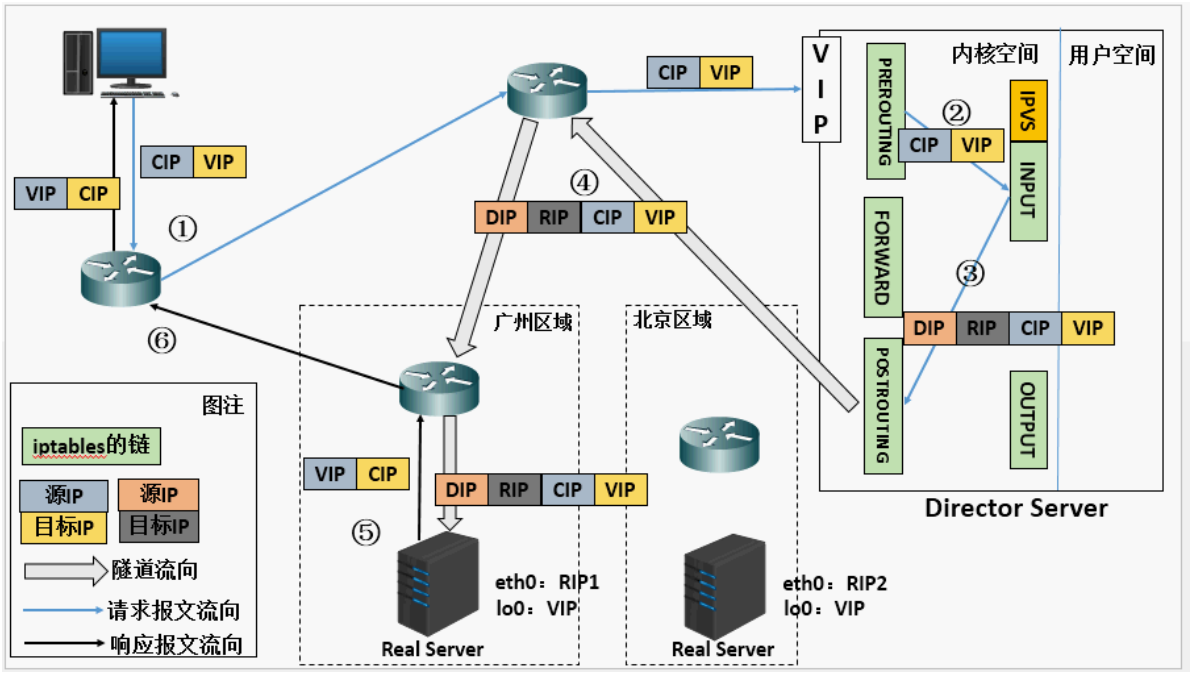
3. 特点1的解决方案：

- 在前端路由器做静态地址路由绑定，将对于VIP的地址仅路由到Director Server
- 存在问题：用户未必有路由操作权限，因为有可能是运营商提供的，所以这个方法未必实用

- arptables：在arp的层次上实现在ARP解析时做防火墙规则，过滤RS响应ARP请求。这是由iptables提供的
- 修改RS上内核参数（arp_ignore和arp_announce）将RS上的VIP配置在lo接口的别名上，并限制其不能响应对VIP地址解析请求。

七、LVS/Tun原理和特点

在原有的IP报文外再次封装多一层IP首部，内部IP首部(源地址为CIP，目标IIP为VIP)，外层IP首部(源地址为DIP，目标IP为RIP)



(a) 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 。
(b) PREROUTING检查发现数据包的目标IP是本机，将数据包送至INPUT链
(c) 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为为DIP，目标IP为RIP。然后发至POSTROUTING链。 此时源IP为DIP，目标IP为RIP
(d) 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
(e) 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
(f) 响应报文最终送达至客户端

LVS-Tun模型特性

- RIP、VIP、DIP全是公网地址
- RS的网关不会也不可能指向DIP
- 所有的请求报文经由Director Server，但响应报文必须不能进过Director Server

- 不支持端口映射
- RS的系统必须支持隧道

其实企业中最常用的是 DR 实现方式，而 NAT 配置上比较简单和方便，后边实践中会总结 DR 和 NAT 具体使用配置过程。

八、LVS的八种调度算法

1.轮叫调度 rr

这种算法是最简单的，就是按依次循环的方式将请求调度到不同的服务器上，该算法最大的特点就是简单。轮询算法假设所有的服务器处理请求的能力都是一样的，调度器会将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，非常均衡地分发下去。

2. 加权轮叫 wrr
这种算法比 rr 的算法多了一个权重的概念，可以给 RS 设置权重，权重越高，那么分发的请求数越多，权重的取值范围 0 – 100。主要是对rr算法的一种优化和补充， LVS 会考虑每台服务器的性能，并给每台服务器添加要给权值，如果服务器A的权值为1，服务器B的权值为2，则调度到服务器B的请求会是服务器A的2倍。权值越高的服务器，处理的请求越多。

3. 最少链接 lc
这个算法会根据后端 RS 的连接数来决定把请求分发给谁，比如 RS1 连接数比 RS2 连接数少，那么请求就优先发给 RS1

4. 加权最少链接 wlc

这个算法比 lc 多了一个权重的概念。

5. 基于局部性的最少连接调度算法 lblc
这个算法是请求数据包的目标 IP 地址的一种调度算法，该算法先根据请求的目标 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

6. 复杂的基于局部性最少的连接算法 lblcr
记录的不是要给目标 IP 与一台服务器之间的连接记录，它会维护一个目标 IP 到一组服务器之间的映射关系，防止单点服务器负载过高。

7. 目标地址散列调度算法 dh
该算法是根据目标 IP 地址通过散列函数将目标 IP 与服务器建立映射关系，出现服务器不可用或负载过高的情况下，发往该目标 IP 的请求会固定发给该服务器。

8. 源地址散列调度算法 sh

与目标地址散列调度算法类似，但它是根据源地址散列算法进行静态分配固定的服务器资源。

九、实践LVS的NAT模式

1、实验环境

三台服务器，一台作为 director，两台作为 real server，director 有一个外网网卡(172.16.254.200) 和一个内网ip(192.168.0.8)，两个 real server 上只有内网 ip (192.168.0.18) 和 (192.168.0.28)，并且需要把两个 real server 的内网网关设置为 director 的内网 ip(192.168.0.8)

2、安装和配置

> 两个 real server 上都安装 nginx 服务
>
> # yum install -y nginx
>
> Director 上安装 ipvsadm
>
> # yum install -y ipvsadm

Director 上编辑 nat 实现脚本

> # vim /usr/local/sbin/lvs_nat.sh
>
> # 编辑写入如下内容：
>
> #! /bin/bash
>
> # director服务器上开启路由转发功能:
>
> echo 1 > /proc/sys/net/ipv4/ip_forward
>
> # 关闭 icmp 的重定向
>
> 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
>
> # director设置 nat 防火墙
>
> iptables -t nat -F
>
> iptables -t nat -X
>
> iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -j MASQUERADE
>
> # director设置 ipvsadm
>
> IPVSADM='/sbin/ipvsadm'
>
> $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
```

保存后，在 Director 上直接运行这个脚本就可以完成 lvs/nat 的配置

```
/bin/bash /usr/local/sbin/lvs_nat.sh
```

查看ipvsadm设置的规则

```
ipvsadm -ln
```

3、测试LVS的效果

通过浏览器测试2台机器上的web内容 http://172.16.254.200 。为了区分开，我们可以把 nginx 的默认页修改一下：

```
在 RS1 上执行

# echo "rs1rs1" >/usr/share/nginx/html/index.html


在 RS2 上执行

# echo "rs2rs2" >/usr/share/nginx/html/index.html
```

注意，切记一定要在两台 RS 上设置网关的 IP 为 director 的内网 IP。

十、实践LVS的DR模式

1、实验环境

三台机器：

- Director节点： (eth0 192.168.0.8 vip eth0:0 192.168.0.38)
- Real server1： (eth0 192.168.0.18 vip lo:0 192.168.0.38)
- Real server2： (eth0 192.168.0.28 vip lo:0 192.168.0.38)

2、安装

```
两个 real server 上都安装 nginx 服务

# yum install -y nginx


Director 上安装 ipvsadm

# yum install -y ipvsadm
```

3、Director 上配置脚本

```
# vim /usr/local/sbin/lvs_dr.sh
#! /bin/bash
echo 1 > /proc/sys/net/ipv4/ip_forward
ipv=/sbin/ipvsadm
vip=192.168.0.38
rs1=192.168.0.18
rs2=192.168.0.28
ifconfig eth0:0 down
ifconfig eth0:0 $vip broadcast $vip netmask 255.255.255.255 up
route add -host $vip dev eth0:0
$ipv -C
$ipv -A -t $vip:80 -s wrr
$ipv -a -t $vip:80 -r $rs1:80 -g -w 3
$ipv -a -t $vip:80 -r $rs2:80 -g -w 1
```

执行脚本：

```
# bash /usr/local/sbin/lvs_dr.sh
```

4、在2台 rs 上配置脚本：

```
# vim /usr/local/sbin/lvs_dr_rs.sh
#! /bin/bash
vip=192.168.0.38
ifconfig lo:0 $vip broadcast $vip netmask 255.255.255.255 up
route add -host $vip 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
```

rs 上分别执行脚本：

```
bash /usr/local/sbin/lvs_dr_rs.sh
```

5、实验测试

测试方式同上，浏览器访问 http://192.168.0.38

注意：在 DR 模式下，2台 rs 节点的 gateway 不需要设置成 dir 节点的 IP 。

参考链接地址：

http://www.cnblogs.com/lgfeng/archive/2012/10/16/2726308.html

十一、LVS结合keepalive

LVS可以实现负载均衡，但是不能够进行健康检查，比如一个rs出现故障，LVS 仍然会把请求转发给故障的rs服务器，这样就会导致请求的无效性。keepalive 软件可以进行健康检查，而且能同时实现 LVS 的高可用性，解决 LVS 单点故障的问题，其实 keepalive 就是为 LVS 而生的。

1、实验环境

4台节点

- Keepalived1 + lvs1(Director1)：192.168.0.48
- Keepalived2 + lvs2(Director2)：192.168.0.58
- Real server1：192.168.0.18
- Real server2：192.168.0.28
- IP: 192.168.0.38

2、安装系统软件

Lvs + keepalived的2个节点安装

```
# yum install ipvsadm keepalived -y
```

Real server + nginx服务的2个节点安装

```
# yum install epel-release -y
# yum install nginx -y
```

3、设置配置脚本

Real server节点2台配置脚本：

```
# vim /usr/local/sbin/lvs_dr_rs.sh
#! /bin/bash
vip=192.168.0.38
ifconfig lo:0 $vip broadcast $vip netmask 255.255.255.255 up
route add -host $vip 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节点rs 上分别执行脚本：
bash /usr/local/sbin/lvs_dr_rs.sh
```

keepalived节点配置(2节点)：

```
主节点( MASTER )配置文件
vim /etc/keepalived/keepalived.conf
```

```
vrrp_instance VI_1 {
  state MASTER
  interface eth0
  virtual_router_id 51
  priority 100
  advert_int 1
  authentication {
    auth_type PASS
    auth_pass 1111
  }
  virtual_ipaddress {
    192.168.0.38
  }
}

virtual_server 192.168.0.38 80 {
  delay_loop 6
  lb_algo rr
  lb_kind DR
  persistence_timeout 0
  protocol TCP

  real_server 192.168.0.18 80 {
    weight 1
    TCP_CHECK {
```

```
    connect_timeout 10
    nb_get_retry 3
    delay_before_retry 3
    connect_port 80
  }
 }

 real_server 192.168.0.28 80 {
  weight 1
  TCP_CHECK {
    connect_timeout 10
    nb_get_retry 3
    delay_before_retry 3
    connect_port 80
  }
 }
}
```

从节点(BACKUP)配置文件

拷贝主节点的配置文件keepalived.conf，然后修改如下内容：

> state MASTER -> state BACKUP
>
> priority 100 -> priority 90

keepalived的2个节点执行如下命令，开启转发功能：

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

4、启动keepalive

> **先主后从分别启动keepalive**
>
> service keepalived start

5、验证结果

实验1

手动关闭192.168.0.18节点的nginx，service nginx stop 在客户端上去测试访问 http://192.168.0.38 结果正常，不会出现访问18节点，一直访问的是28节点的内容。

实验2

手动重新开启 192.168.0.18 节点的nginx， service nginx start 在客户端上去测试访问 http://192.168.0.38 结果正常，按照 rr 调度算法访问18节点和28节点。

实验3

测试 keepalived 的HA特性，首先在master上执行命令 ip addr ，可以看到38的vip在master节点上的；这时如果在master上执行 service keepalived stop 命令，这时vip已经不再master上，在slave节点上执行 ip addr 命令可以看到 vip 已经正确漂到slave节点，这时客户端去访问 http://192.168.0.38 访问依然正常，验证了 keepalived的HA特性。

lvs 介绍：http://www.it165.net/admin/html/201401/2248.html