|  |
| --- |
| lvs |
| **作者：**  **归档：学习笔记**  **2017** |
| **快捷键：**  Ctrl + 1 标题1  Ctrl + 2 标题2  Ctrl + 3 标题3  Ctrl + 4 实例  Ctrl + 5 程序代码  Ctrl + 6 正文 |
| **格式说明：**  蓝色字体：注释  黄色背景：重要  绿色背景：注意 |

目 录

[第1章 1](#_Toc497836406)

[1.1 负载均衡的妙用 1](#_Toc497836407)

[1.2 为什么要学LVS 1](#_Toc497836408)

[1.3 LVS介绍 1](#_Toc497836409)

[1.4 LVS内核模块ip\_vs介绍 2](#_Toc497836410)

[1.5 LVS集群搭建 2](#_Toc497836411)

[1.5.1 环境准备 2](#_Toc497836412)

[1.5.2 安装ipvsadm管理工具（只在lb01操作） 2](#_Toc497836413)

[1.5.3 配置LVS负载均衡服务（只在lb01操作） 3](#_Toc497836414)

[1.5.4 web服务器配置（在web01/web02同时操作下面步骤） 3](#_Toc497836415)

[1.5.5 测试 4](#_Toc497836416)

[1.6 问题1：浏览器访问没有发现轮询效果 4](#_Toc497836417)

[1.7 问题2：使用抓包工具，发现进行通信的是windows的IP和lb01的80端口，但是lb01明明没有80端口？ 4](#_Toc497836418)

[1.7.1 抓包结果 5](#_Toc497836419)

[1.7.2 相关名词概念 6](#_Toc497836420)

[1.7.3 LVS集群的工作模式--DR直接路由模式 6](#_Toc497836421)

[1.8 问题3：客户端的操作有什么含义？ 8](#_Toc497836422)

[1.8.1 RealServer为什么要在lo接口上配置VIP？ 8](#_Toc497836423)

[1.8.2 在eth0网卡上配置VIP可以吗？ 8](#_Toc497836424)

[1.8.3 为什么要抑制ARP响应？ 8](#_Toc497836425)

[1.8.4 arp\_announce和arp\_ignore详解 9](#_Toc497836426)

[1.9 LVS集群的工作模式总结 11](#_Toc497836427)

[1.9.1 NAT 11](#_Toc497836428)

[1.9.2 隧道模式 11](#_Toc497836429)

[1.9.3 FULLNAT 12](#_Toc497836430)

[第2章 LVS(负载均衡)+keepalived(高可用) 13](#_Toc497836431)

[2.1 负载均衡 13](#_Toc497836432)

[2.2 安装配置Keepalived 13](#_Toc497836433)

[2.2.1 步骤1：在lb01和lb02安装Keepalive 13](#_Toc497836434)

[2.2.2 步骤2：配置Keepalive， lb01和lb02的配置文件分成三部分配置 13](#_Toc497836435)

[2.2.3 启动Keepalived 18](#_Toc497836436)

[2.3 web服务器配置 18](#_Toc497836437)

[2.4 LVS+Keepalived多实例 18](#_Toc497836438)

[2.4.1 题目 18](#_Toc497836439)

[2.4.2 lb01配置 19](#_Toc497836440)

[2.4.3 lb02配置 21](#_Toc497836441)

[2.5 测试Keepalive健康检查功能 24](#_Toc497836442)

[2.6 LVS故障排错 25](#_Toc497836443)

[2.7 常见LVS负载均衡高可用解决方案 25](#_Toc497836444)

[2.7.1 开发管理LVS脚本，类似Keepalive 25](#_Toc497836445)

# 

## 负载均衡的妙用

负载均衡提供了一种廉价、有效、透明的方法，来扩展网络设备和服务器的负载、宽带、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性。

* 单台计算机无法承受大规模的并发访问和数据流量了，此时需要搭建负载均衡集群把流量分摊到多台节点设备上分别处理，即减少用户等待响应的时间又提升了用户体验。
* 7\*24小时的服务保证，任意一个或多个有限后端节点设备宕机，不能影响整个业务的运行。

## 为什么要学LVS

* 工作在网络模型的7层，可以针对http应用做一些分流的策略，比如针对域名、

目录结构， Nginx单凭这点可利用的场合就远多于LVS了。

* 最新版本的Nginx也支持4层TCP负载，曾经这是LVS比Nginx好的地方。
* Nginx对网络稳定性的依赖非常小，理论上能ping通就就能进行负载功能，这个

也是它的优势之一，相反LVS对网络稳定性依赖比较大。

* Nginx安装和配置比较简单，测试起来比较方便，它基本能把错误用日志打印出

来。 LVS的配置、测试就要花比较长的时间了， LVS对网络依赖比较大。

简单一句话，当并发超过了Nginx上限，就可以使用LVS了。

日1000-2000W PV或并发请求1万以下都可以考虑用Nginx。

大型门户网站，电商网站需要用到LVS。

## LVS介绍

LVS是Linux Virtual Server的简写，意即Linux虚拟服务器，是一个虚拟的服务器集群系

统，可以在UNIX/LINUX平台下实现负载均衡集群功能。该项目在1998年5月由章文嵩博士组

织成立，是中国国内最早出现的自由软件项目之一。

官网： http://www.linuxvirtualserver.org/index.html

中文资料

* LVS项目介绍 http://www.linuxvirtualserver.org/zh/lvs1.html
* LVS集群的体系结构 http://www.linuxvirtualserver.org/zh/lvs2.html
* LVS集群中的IP负载均衡技术http://www.linuxvirtualserver.org/zh/lvs3.html
* LVS集群的负载调度 http://www.linuxvirtualserver.org/zh/lvs4.html

## LVS内核模块ip\_vs介绍

早在2.2内核时， IPVS就已经以内核补丁的形式出现。

从2.4.23版本开始， IPVS软件就合并到Linux内核的常用版本的内核补丁的集合。

从2.4.24以后IPVS已经成为Linux官方标准内核的一部分



* LVS无需安装
* 安装的是管理工具，第一种叫ipvsadm，第二种叫keepalive
* ipvsadm是通过命令行管理，而keepalive读取配置文件管理
* 后面我们会用Shell脚本实现keepalive的功能

## LVS集群搭建

### 环境准备

1.准备4台安装好CentOS7.2系统的虚拟机，内存512M。

2.所有虚拟机的防火墙和Selinux关闭

3.主机名及IP地址关系如下：

lb01 10.0.0.5

lb02 10.0.0.6

web01 10.0.0.8

web02 10.0.0.7

4.web01和web02只需要安装Nginx软件，并满足下面条件：

curl http://10.0.0.8/www/得到结果为web01

curl http://10.0.0.7/www/得到结果为web02

5.安装好wireshark 2.2.2版本 2.2.x版本以上即可

### 安装ipvsadm管理工具（只在lb01操作）

# 查看系统的LVS模块。

lsmod|grep ip\_vs

# 默认没有加载模块，需要安装管理工具才会激活。

yum -y install ipvsadm

# 查看当前LVS状态，顺便激活LVS内核模块。

ipvsadm

[root@lb01 ~]# lsmod|grep ip\_vs

ip\_vs 141092 0

nf\_conntrack 111302 1 ip\_vs

libcrc32c 12644 2 xfs,ip\_vs

### 配置LVS负载均衡服务（只在lb01操作）

步骤1：在eth0网卡绑定VIP地址（ip）

步骤2：清除当前所有LVS规则（-C）

步骤3：设置tcp、 tcpfin、 udp链接超时时间（--set）

步骤4：添加虚拟服务（-A），调度算法见man ipvsadm

步骤5：将虚拟服务关联到真实服务上（-a）

步骤6：查看配置结果（-ln）

ip addr add 10.0.0.3/24 dev eth0

ipvsadm -C

ipvsadm --set 30 5 60

ipvsadm -A -t 10.0.0.3:80 -s wrr -p 20

ipvsadm -a -t 10.0.0.3:80 -r 10.0.0.7:80 -g -w 1

ipvsadm -a -t 10.0.0.3:80 -r 10.0.0.8:80 -g -w 1

ipvsadm –ln

注：-p 轮询的超时时间

-g 网关

常用LVS调度算法：基本轮叫调度rr，加权最小连接调度wlc，加权轮叫调度wrr

固定调度算法：rr,wrr,dh,sh

动态调度算法：wlc,lc,lblc,lblcr,SED,NQ(后两种官方站点没提到，编译LVS，make过程可以看到rr|wrr|lc|wlc|lblc|lblcr|dh|sh|sed|nq

-ln查看有没有配置10.0.0.3这个VIP对应的负载服务啊

ipvsadm -ln -t 10.0.0.3:80

如果只想清除一个实例 就是-D  
-A添加 -D删除  
-a添加real server  -d 删除

ipvsadm -d -t 10.0.0.3:80 -r 10.0.0.7:80

### web服务器配置（在web01/web02同时操作下面步骤）

步骤1：在lo网卡绑定VIP地址（ip）

步骤2：修改内核参数抑制ARP响应

ip addr add 10.0.0.3/32 dev lo

cat >>/etc/sysctl.conf<<EOF

net.ipv4.conf.all.arp\_ignore = 1

net.ipv4.conf.all.arp\_announce = 2

net.ipv4.conf.lo.arp\_ignore = 1

net.ipv4.conf.lo.arp\_announce = 2

EOF

sysctl -p

### 测试

使用浏览器访问http://10.0.0.3/www

不要在lb01访问！可以在lb02测试（未加入LVS“团伙”）

## 问题1：浏览器访问没有发现轮询效果

答： LVS的轮询不像Nginx明显，可以使用多个客户端访问（Windows和Linux）

## 问题2：使用抓包工具，发现进行通信的是windows的IP和lb01的80端口，但是lb01明明没有80端口？

答： Windows抓包查看，可以发现数据包的源MAC地址是web01或web02

Linux:tcpdump -nn port 80; tcpdump -nn -e port 80

**结论**：

1.从客户端发送http请求，

数据包的IP为客户端==》lb01 VIP ，

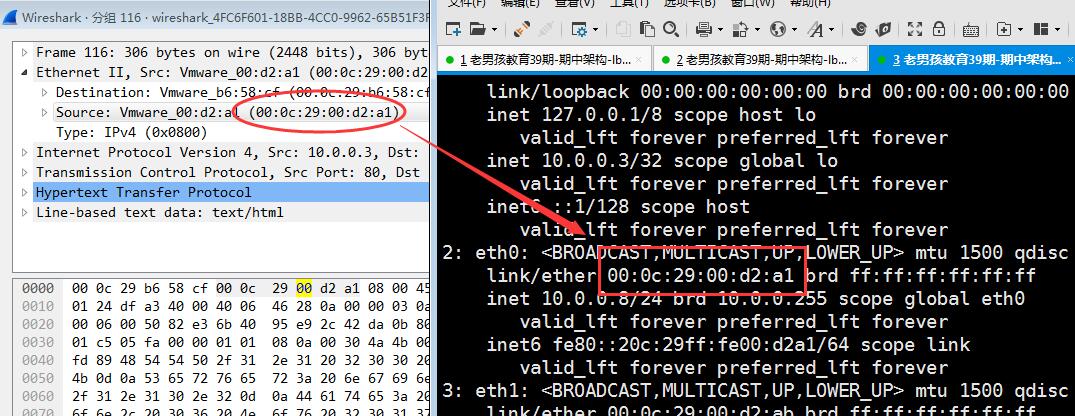
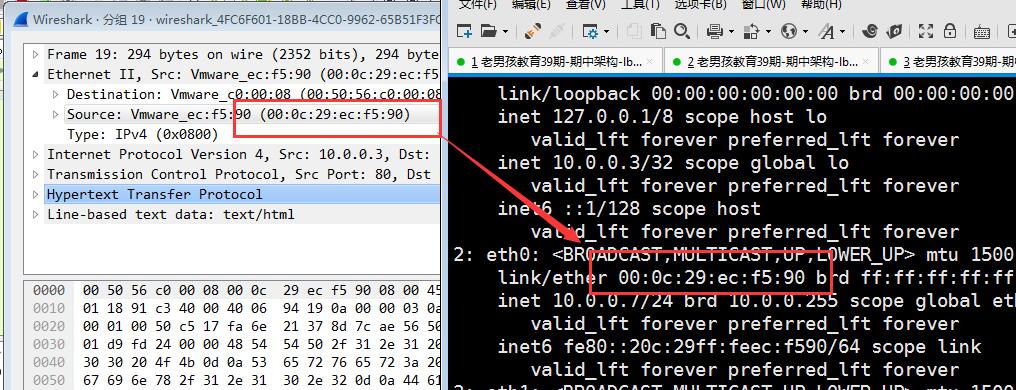
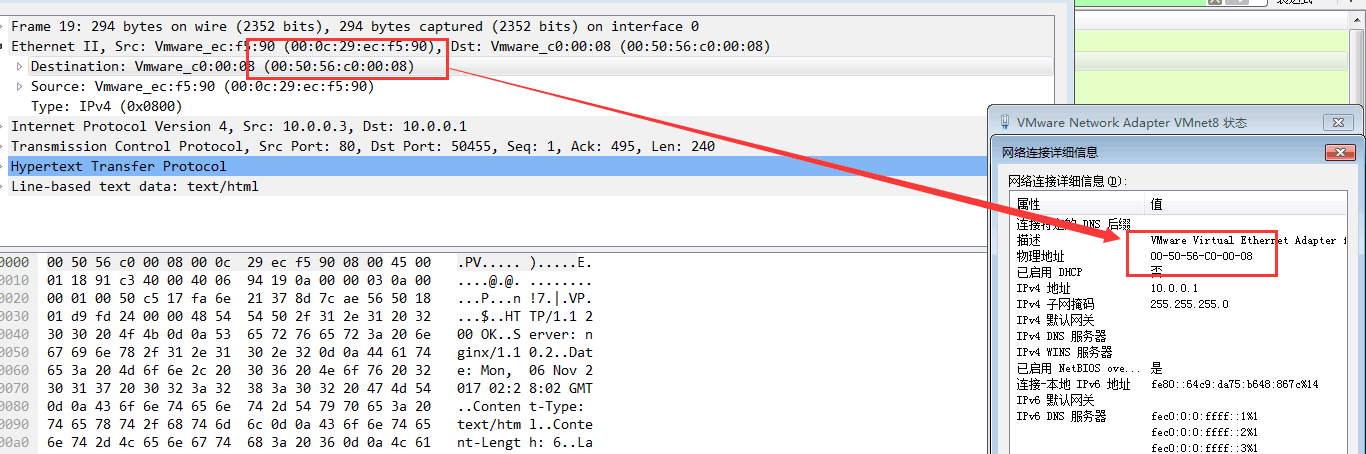
mac地址：客户端的MAC地址==》lb01 MAC地址

2.从lb01响应的数据包，

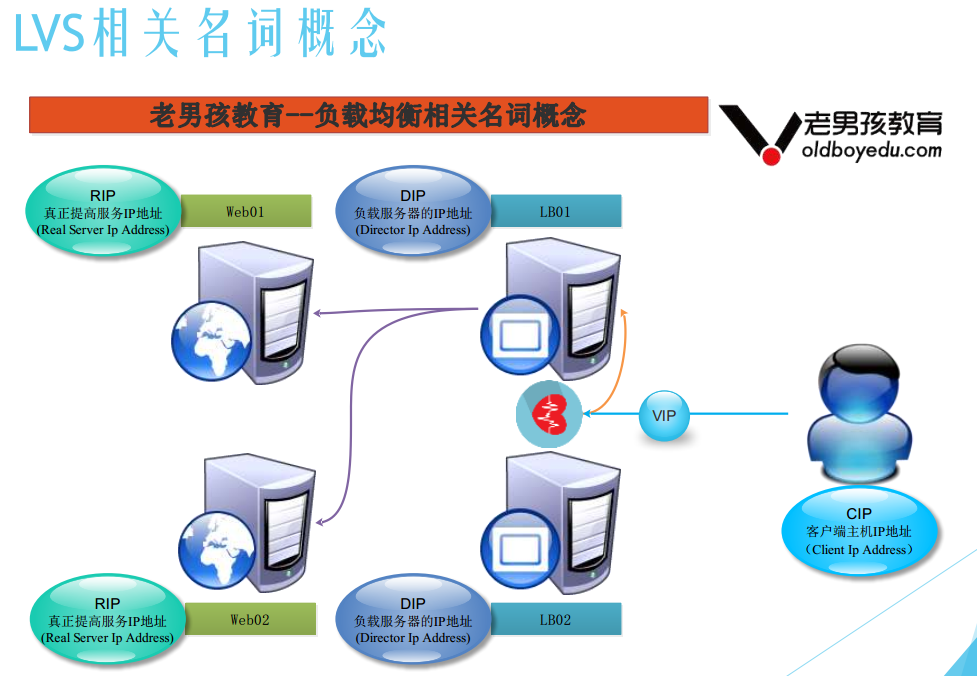
IP：lb01 VIP ==> 客户端

MAC地址：web01或者web02的MAC地址==》客户端的MAC地址

### 抓包结果



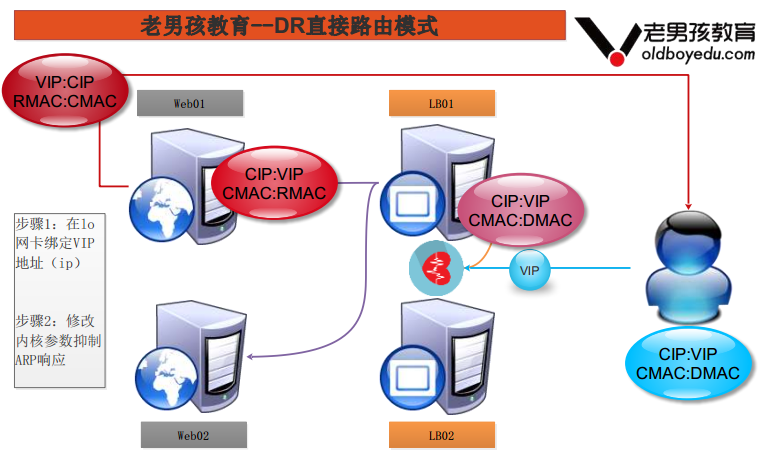
### 相关名词概念



### LVS集群的工作模式--DR直接路由模式

DR模式是通过改写请求报文的目标MAC地址，将请求发给真实服务器的，而真实服务器将响应后的处理结果直接返回给客户端用户。

DR技术可极大地提高集群系统的伸缩性。但要求调度器LB与真实服务器RS都有一块物理网卡连在同一物理网段上，即必须在同一局域网环境。



**结论**：

* 通过在调度器LB上修改数据包的目的MAC地址实现转发。注意，源IP地址仍然是CIP，目的IP地址仍然是VIP。
* 请求的报文经过调度器，而RS响应处理后的报文无需经过调度器LB，因此，并发访问量大时使用效率很高，比Nginx代理模式强于此处。
* 因DR模式是通过MAC地址的改写机制实现转发的，因此，所有RS节点和调度器LB只能在同一个局域网中。
* 需要注意RS节点的VIP的绑定(lo:vip/32)和ARP抑制问题。
* 强调下： RS节点的默认网关不需要是调度器LB的DIP，而应该直接是IDC机房分配的上级路由器的IP(这是RS带有外网IP地址的情况)，理论上讲，只要RS可以出网即可，不需要必须配置外网IP，但走自己的网关，那网关就成为瓶颈了。
* 由于DR模式的调度器仅进行了目的MAC地址的改写，因此，调度器LB无法改变请求报文的目的端口。 LVSDR模式的办公室在二层数据链路层（MAC）， NAT模式则工作在三层网络层（IP）和四层传输层（端口）
* 当前，调度器LB支持几乎所有UNIX、 Linux系统，但不支持windows系统。真实服务器RS节点可以是windows系统。
* 总之， DR模式效率很高，但是配置也较麻烦。因此，访问量不是特别大的公司可以用haproxy/Nginx取代之。这符合运维的原则：简单、易用、高效。日1000-2000W PV或并发请求1万以下都可以考虑用haproxy/Nginx(LVS的NAT模式)
* 直接对外的访问业务，例如web服务做RS节点， RS最好用公网IP地址。如果不直接对外的业务，例如：MySQL，存储系统RS节点，最好只用内部IP地址。

## 问题3：客户端的操作有什么含义？

### RealServer为什么要在lo接口上配置VIP？

答：既然要让RS能够处理目标地址为vip的IP包，首先必须要让RS能接收到这个包。

在lo上配置vip能够完成接收包并将结果返回client。

### 在eth0网卡上配置VIP可以吗？

答：不可以，将VIP设置在eth0网卡上,会影响RS的arp请求,造成整体LVS集群arp缓

存表紊乱，以至于整个负载均衡集群都不能正常工作。

### 为什么要抑制ARP响应？

先回顾ARP知识及了解arp\_announce和arp\_ignore作用。

#### **回顾--ARP网络知识**

**ARP协议**,全称"Address Resolution Protocol",中文名是地址解析协议，使用ARP协

议可实现通过IP地址获得对应主机的物理地址(MAC地址)。

10.0.0.1  ARP  00:50:56:c0:00:08

域名:老男孩教育

IP：汇德商厦403

MAC地址：教室的编号

快递员给学生送快递，最多就知道学校的地址（IP），但是不知 道学生在哪个教室。

班主任就是ARP协议，告诉快递员具体的教室。

ARP协议要求通信的主机双方必须在同一个物理网段（即局域网环境）！

为了提高IP转换MAC的效率，系统会将解析结果保存下来，这个结果叫做**ARP缓存**。

Windows查看ARP缓存命令 arp -a

Linux查看ARP缓存命令 arp -n

Linux解析IP对应的MAC地址 arping -c 1 -I eth0 10.0.0.6

* ARP缓存表是把双刃剑

①主机有了arp缓存表，可以加快ARP的解析速度，减少局域网内广播风暴。因为arp

是发广播解析的，频繁的解析也是消耗带宽的，尤其是机器多的时候。

②正是有了arp缓存表，给恶意黑客带来了攻击服务器主机的风险，这个就是**arp欺**

**骗攻击**。 有同学恶作剧，伪装班主任告诉快递员错误的教室编号。

③切换路由器，负载均衡器等设备时，可能会导致短时网络中断。因为所有的客户

端ARP缓存表没有更新。

#### **服务器切换ARP问题**

当集群中一台提供服务的lb01机器宕机后，然后VIP会转移到备机lb02上，但是

客户端的ARP缓存表的地址解析还是宕机的lb01的MAC地址。从而导致，即使在lb02

上添加VIP，也会发生客户端无法访问的情况。

解决办法是：当lb01宕机， VIP地址迁移到lb02时，需要通过arping命令通知所

有网络内机器更新本地的ARP缓存表，从而使得客户机访问时重新广播获取MAC地址。

这个是自己开发服务器高可用脚本及所有高可用软件必须考虑到的问题。

ARP广播进行新的地址解析

arping -I eth0 -c 1 -U VIP

arping -I eth0 -c 1 -U 10.0.0.3

测试命令

ip addr del 10.0.0.3/24 dev eth0

ip addr add 10.0.0.3/24 dev eth0

ip addr show eth0

arping -I eth0 -c 1 -U 10.0.0.3

windows查看arp –a

### arp\_announce和arp\_ignore详解

#### 【lvs在DR模式下需要关闭arp功能】

arp\_announce

对网络接口上，本地IP地址的发出的，ARP回应，作出相应级别的限制:

确定不同程度的限制,宣布对来自本地源IP地址发出Arp请求的接口

0 - (默认) 在任意网络接口（eth0,eth1， lo）上的任何本地地址

1 -尽量避免不在该网络接口子网段的本地地址做出arp回应. 当发起ARP请求的源IP地址

是被设置应该经由路由达到此网络接口的时候很有用.此时会检查来访IP是否为所有接口

上的子网段内ip之一.如果改来访IP不属于各个网络接口上的子网段内,那么将采用级别2

的方式来进行处理.

2 - 对查询目标使用最适当的本地地址.在此模式下将忽略这个IP数据包的源地址并尝试

选择与能与该地址通信的本地地址.首要是选择所有的网络接口的子网中外出访问子网中

包含该目标IP地址的本地地址. 如果没有合适的地址被发现,将选择当前的发送网络接口

或其他的有可能接受到该ARP回应的网络接口来进行发送.

#### 【lvs在DR模式下需要关闭arp功能】

arp\_ignore

定义对目标地址为本地IP的ARP询问不同的应答模式0

0 - (默认值): 回应任何网络接口上对任何本地IP地址的arp查询请求

1 - 只回答目标IP地址是来访网络接口本地地址的ARP查询请求

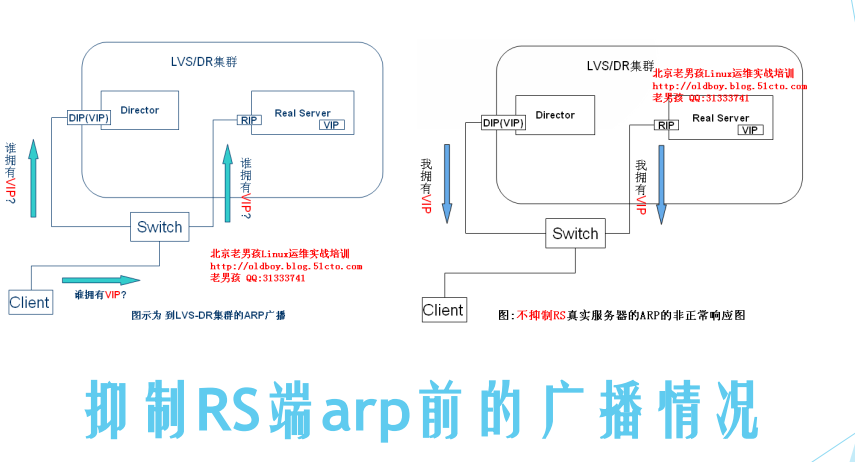
2 -只回答目标IP地址是来访网络接口本地地址的ARP查询请求,且来访IP必须在该网络接

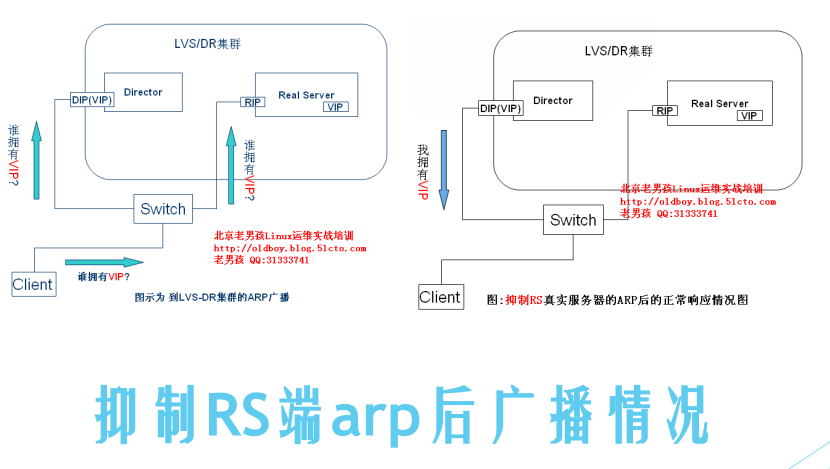
口的子网段内

3 - 不回应该网络界面的arp请求，而只对设置的唯一和连接地址做出回应

4-7 - 保留未使用

8 -不回应所有（本地地址）的arp查询





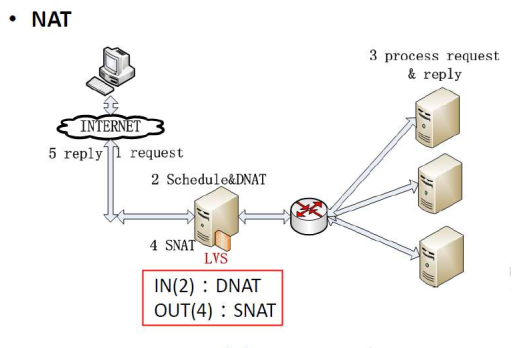
## LVS集群的工作模式总结

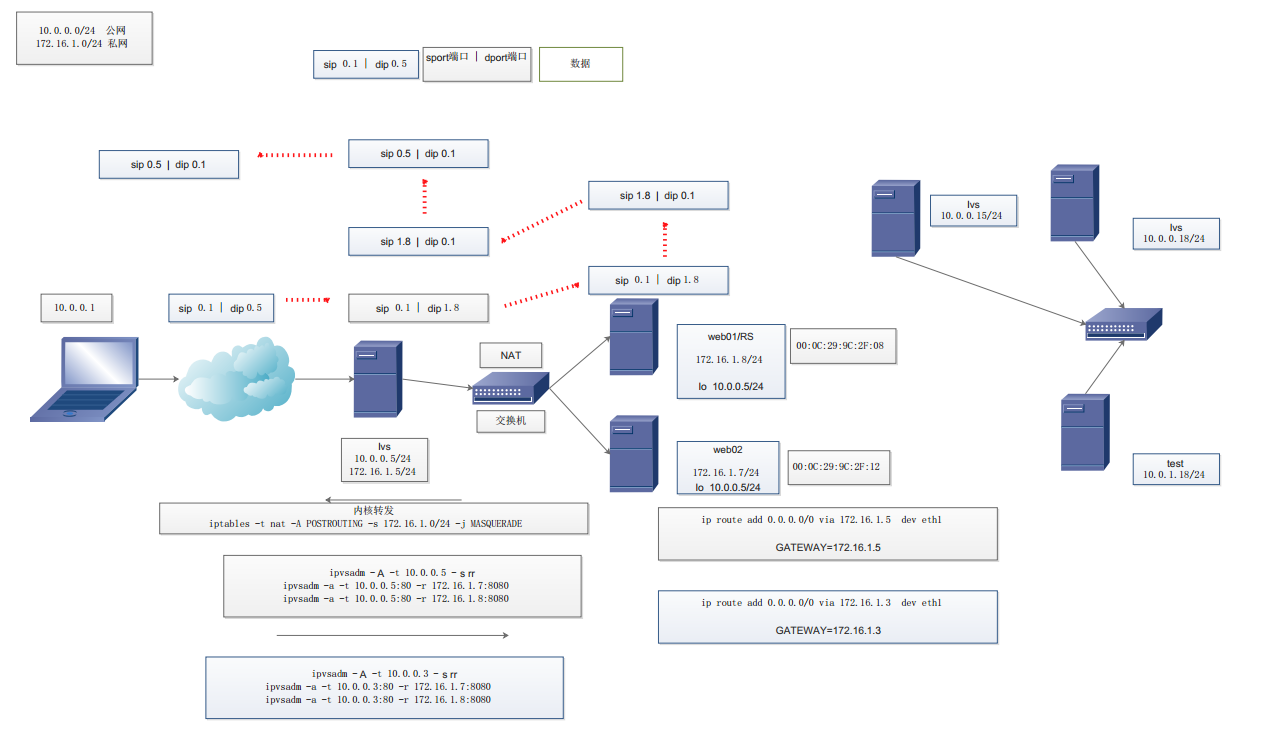
* DR（Direct Routing）直接路由模式
* NAT（Network Address Translation）
* TUN（Tunneling）隧道模式
* FULLNAT（Full Network Address Translation）

### NAT

通过网络地址转换，调度器LB重写请求报文的目标地址，根据预设的调度算法，将请求分派给后端的真实服务器，真实服务器的响应报文处理之后，返回时必须要通过调度器，经过调度器时报文的源地址被重写，再返回给客户，完成整个负载调度过程。

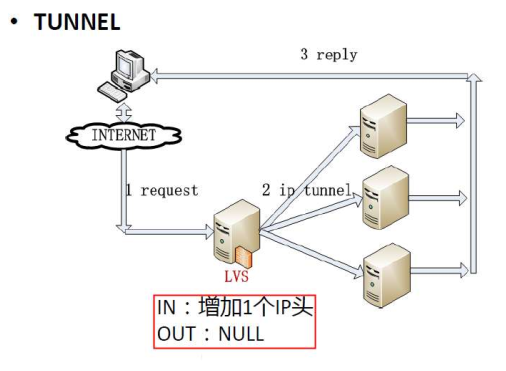
收费站模式---来去都要经过LB负载均衡器





### 隧道模式

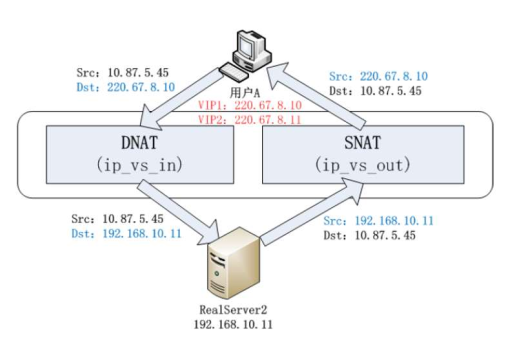
采用NAT技术时，由于请求和响应的报文都必须经过调度器地址重写，当客户请求越来越多时，调度器的处理能力将成为瓶颈，为了解决这个问题，调度器把请求的报文通过IP隧道(相当于ipip或ipsec )转发至真实服务器，而真实服务器将响应处理后直接返回给客户端用户，这样调度器就只处理请求的入站报文。由于一般网络服务应答数据比请求报文大很多，采用 VS/TUN技术后，集群系统的最大吞吐量可以提高10倍。VS/TUN工作流程,它的连接调度和管理与VS/NAT中的一样，只是它的报文转发方法不同。调度器根据各个服务器的负载情况，连接数多少，动态地选择一台服务器，将原请求的报文封装在另一个IP报文中，再将封装后的IP报文转发给选出的真实服务器；真实服务器收到报文后，先将收到的报文解封获得原来目标地址为VIP地址的报文, 服务器发现VIP地址被配置在本地的IP隧道设备上(此处要人为配置)，所以就处理这个请求，然后根据路由表将响应报文直接返回给客户。



### FULLNAT

LVS的DR和NAT模式要求RS和LVS在同一个vlan中，导致部署成本过高；TUNNEL模式虽然可以跨vlan，但RealServer上需要部署ipip隧道模块等，网络拓扑上需要连通外网，较复杂，不易运维。为了解决上述问题，开发出FULLNAT，该模式和NAT模式的区别是：数据包进入时，除了做DNAT，还做SNAT（用户ip->内网ip），从而实现LVS-RealServer间可以跨vlan通讯， RealServer只需要连接到内网。

类比地铁站多个闸机。



# LVS(负载均衡)+keepalived(高可用)

声明：接下来的实验又是一个新的开始，配置负载均衡不需要ipvsadm这个工具，因为keepalived本来就是另外一个自动管理工具。

## 负载均衡

软件负载

nginx、lvs nat 反向代理型负载均衡

lvs dr 负载均衡

haproxy

硬件负载

F5 A10

## 安装配置Keepalived

### 步骤1：在lb01和lb02安装Keepalive

yum -y install keepalived

### 步骤2：配置Keepalive， lb01和lb02的配置文件分成三部分配置

1.global\_defs 全局定义

2.vrrp 实例配置  VIP

3.virtual\_server配置 lvs的配置

【**第一部分：全局定义**】

###########lb01###########

global\_defs {

router\_id LVS\_01

}

###########lb02###########

global\_defs {

router\_id LVS\_02

}

【第二部分： VIP配置】

###########lb01###########

vrrp\_instance VI\_1 {

state MASTER

interface eth0

virtual\_router\_id 51

priority 150

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.0.0.3/24

}

}

###########lb02###########

vrrp\_instance VI\_1 {

state BACKUP

interface eth0

virtual\_router\_id 51

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.0.0.3/24

}

}

【**第三部分：lvs的配置**】

这部分两台负载一样。根据配置文件对比前面学过的ipvsadm命令



#### lb01配置

global\_defs {

router\_id LVS\_01

}

vrrp\_instance VI\_1 {

state MASTER

interface eth0

virtual\_router\_id 51

priority 150

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.0.0.3/24

}

}

virtual\_server 10.0.0.3 80 {

delay\_loop 6

lb\_algo wrr

lb\_kind DR

nat\_mask 255.255.255.0

persistence\_timeout 50

protocol TCP

real\_server 10.0.0.7 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

real\_server 10.0.0.8 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

}

#### lb02配置

global\_defs {

router\_id LVS\_02

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth0

virtual\_router\_id 51

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.0.0.3/24

}

}

virtual\_server 10.0.0.3 80 {

delay\_loop 6

lb\_algo wrr

lb\_kind DR

nat\_mask 255.255.255.0

persistence\_timeout 50

protocol TCP

real\_server 10.0.0.7 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

real\_server 10.0.0.8 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

}

### 启动Keepalived

检查lb01的vip， ipvsadm -ln是否还在，清除掉。

systemctl start keepalived.service

systemctl status keepalived.service

ip addr show eth0

ipvsadm -ln

可以测试keepalive高可用，故障转移(包含VIP及LVS配置)。

## web服务器配置

步骤1：在lo网卡绑定VIP地址（ip）

步骤2：修改内核参数抑制ARP响应

ip addr add 10.0.0.3/32 dev lo

cat >>/etc/sysctl.conf<<EOF

net.ipv4.conf.all.arp\_ignore = 1

net.ipv4.conf.all.arp\_announce = 2

net.ipv4.conf.lo.arp\_ignore = 1

net.ipv4.conf.lo.arp\_announce = 2

EOF

sysctl –p

## LVS+Keepalived多实例

### 题目

lb01和lb02 互为准备

lb01 作为LVS A集群的master VIP 10.0.0.3 lb02作为备节点

lb02 作为LVS B集群的master VIP 10.0.0.4 lb01作为备节点

A集群管理的web机器有10.0.0.7 10.0.0.8

B集群管理的web机器有10.0.0.9 10.0.0.10（机器不够用10.0.0.7 10.0.0.8模拟）

### lb01配置

global\_defs {

router\_id LVS\_01

}

vrrp\_instance VI\_1 {

state MASTER

interface eth0

virtual\_router\_id 51

priority 150

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.0.0.3/24

}

}

vrrp\_instance VI\_2 {

state BACKUP

interface eth0

virtual\_router\_id 52

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 2222

}

virtual\_ipaddress {

10.0.0.4/24

}

}

virtual\_server 10.0.0.3 80 {

delay\_loop 6

lb\_algo wrr

lb\_kind DR

nat\_mask 255.255.255.0

persistence\_timeout 50

protocol TCP

real\_server 10.0.0.7 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

real\_server 10.0.0.8 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

}

virtual\_server 10.0.0.4 80 {

delay\_loop 6

lb\_algo wrr

lb\_kind DR

nat\_mask 255.255.255.0

persistence\_timeout 50

protocol TCP

real\_server 10.0.0.7 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

real\_server 10.0.0.8 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

}

### lb02配置

global\_defs {

router\_id LVS\_02

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth0

virtual\_router\_id 51

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.0.0.3/24

}

}

vrrp\_instance VI\_2 {

state MASTER

interface eth0

virtual\_router\_id 52

priority 150

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 2222

}

virtual\_ipaddress {

10.0.0.4/24

}

}

virtual\_server 10.0.0.3 80 {

delay\_loop 6

lb\_algo wrr

lb\_kind DR

nat\_mask 255.255.255.0

persistence\_timeout 50

protocol TCP

real\_server 10.0.0.7 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

real\_server 10.0.0.8 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

}

virtual\_server 10.0.0.4 80 {

delay\_loop 6

lb\_algo wrr

lb\_kind DR

nat\_mask 255.255.255.0

persistence\_timeout 50

protocol TCP

real\_server 10.0.0.7 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

real\_server 10.0.0.8 80 {

weight 1

TCP\_CHECK {

connect\_timeout 8

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

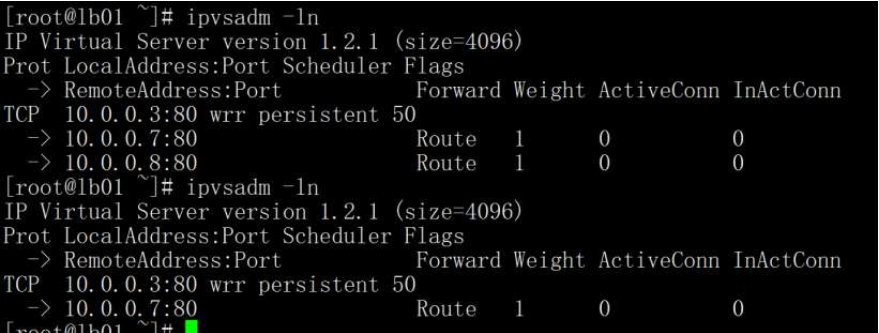
}

}

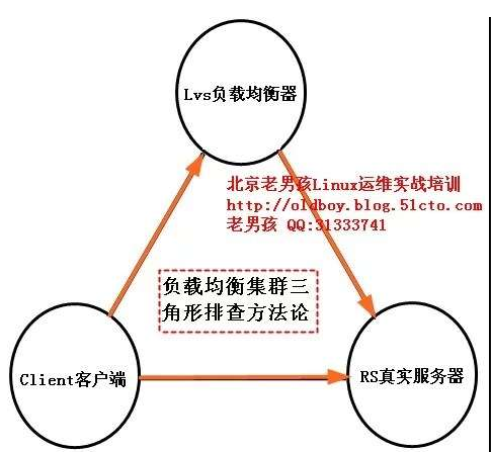
}

## 测试Keepalive健康检查功能

关闭web01



## LVS故障排错



## 常见LVS负载均衡高可用解决方案

* 开发类似keepalived的脚本，早期的办法，现在不推荐使用。
* heartbeat+lvs+ldirectord脚本配置方案，复杂不易控制，不推荐使用
* RedHat工具piranha，一个web界面配置LVS。
* LVS-DR+keepalived方案，老师推荐最优方案，简单、易用、高效。

### 开发管理LVS脚本，类似Keepalive

添加真实服务器

ipvsadm -a -t 10.0.0.3 -r 10.0.0.7:80

删除真实服务器

ipvsadm -d -t 10.0.0.3 -r 10.0.0.7:80

**企业Shell面试题21：开发脚本管理服务端LVS案例**

请在LVS负载均衡主节点上，开发管理LVS服务的脚本ip\_vs。实现：利用ipvsadm可以启

动并配置好LVS服务，脚本形式： /etc/init.d/lvs{start|stop|restart}

**企业Shell面试题22： LVS节点健康检查及管理脚本案例**

请在LVS负载均衡主节点上，模拟keepalived健康检查功能管理LVS节点，当节点挂掉从

服务器池中剔除，好了再加到服务器池中来。

**企业Shell面试题23： LVS客户端配置脚本案例**

请在LVS客户端节点上，开发LVS客户端设置VIP以及抑制ARP的管理脚本实现：

/etc/init.d/lvsclient{start|stop|restart}

**企业Shell面试题24：模拟keepalved软件高可用案例**

请在LVS服务端备用节点上，模拟keepalved vrrp功能，监听主节点，如果主节点不可访

问则备节点启动并配置LVS实现接管主节点的资源提供服务（提醒：注意ARP缓存），提

示此题要借助19.1.21的功能。

