一、LVS介绍

1.1 LVS相关术语命名约定

|  |  |  |
| --- | --- | --- |
| 名称 | 缩写 | 说明 |
| 虚拟IP地址（Virtual ip address） | VIP | Director用于向客户端计算机提供服务的IP地址。 |
| 真实IP地址（Real server ip address） | RIP | 在集群下面节点上使用的IP地址 |
| Director的IP地址（Director ip adderess） | DIP | Diretor用于连接内外网络的IP地址 |
| 客户端主机IP地址（Client ip address） | CIP | 客户端计算机请求集群服务器的IP地址，即发送给集群的请求的源IP地址。 |

1.2 LVS的3种工作模式

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，还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。

在调度实现技术中，IP负载均衡技术时效率最高的。在已有的IP负载均衡技术中有通过网络地址转换（Network Address Translation）将一组服务器构成一个高性能的，高可用的虚拟服务器，称之为VS/NAT （Virtual Server via Network Address Translation）技术。大多是商品化的IP负载均衡调度器产品都使用此方法，如Cisco的LocalDiretor、F5的Big/ip和alteon的ACEDiretor。

通过IP隧道实现虚拟服务器的方法VS/TUN（Virtual Server via IP Tunneling）和通过直接路由实现虚拟服务器的方法VS/DR（Virtual Server via Direct Routing），它们可以极大地提高系统的伸缩性。所以，IPVS实现了这三种IP负载均衡技术。

1）NAT模式-收费站模式

请求和响应报文都要经过负载均衡器，进行地址重写（修改目的地址，端口，源地址和源端口）。

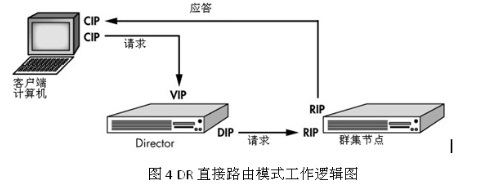
提示：来去都要经过负载均衡服务器，它只能胜任10台以下real server。

2）TUN模式-隧道模式

这个模式下，负载均衡调度器把请求报文通过IP隧道（相当于ipip，ipsec）转发至真实服务器，而真实服务器将响应直接返回给客户。这样调度器只处理请求报文，由于一般网络服务应答数据比请求报文大很多，采用VS/TUN模式，集群的最大吞吐量可以提高10倍。

3）DR模式-直接路由

VS/DR通过改写请求报文的MAC地址，将请求发送到real server，而real server将响应直接返回给客户。同VS/TUN技术一样，VS/DR技术可极大地提高集群系统的伸缩性。这样方法没有IP隧道的开销，对集群中的真实服务器也没有必须支持IP隧道协议的要求，但是要求调度器与真实服务器都有一块网卡连在同一物理网段上。



1.3 LVS三种模式优缺点比较

|  |  |  |  |
| --- | --- | --- | --- |
|  | VS/NAT | VS/TUN | VS/DR |
| Real Server | config dr gw | tunning | non-arp device/tie vip |
| Server Network | private | LAN/WAN | LAN |
| Server Number | low | High | High |
| Real Server Gateway | load balancer (DIP) | own router | own router |

二、LVS的3种模式介绍

参考：http://blief.blog.51cto.com/6170059/1745134

1、VS/NAT集群原理

1.1 VS/NAT模式原理描述

1.client发送request到LVS的VIP上，VIP选择一个Real-server，并记录连接信息到hash表中，然后修改client的request的目的IP地址为Real-server的地址，将请求发给Real-server。

2.Real-server收到request包后，发现目的IP是自己的IP，于是处理请求，然后发送reply给LVS。

3.LVS收到reply包后，修改reply包的的源地址为VIP，发送给client。

4.从client来的属于本次连接的包，查hash表，然后发给对应的Real-server。

5.当client发送完毕，此次连接结束或者连接超时，那么LVS自动从hash表中删除此条记录。

LVS-NAT模式主要是将客户端发送过来的包目标地址更改为后端RIP的ip，主要把三层的信息做了更改，其他都不变，此配置下后端RIP不需要做任何配置，但必须保证RIP的网关为与RIP相连的DIR接口ip，因为LVS-NAT主要用到DNAT和SNAT原理。

1.2 VS/NAT模式特点

1）RS应该使用私有地址

2）RS的网关必须指向DIP

3）RIP和DIP必须在同一网段内

4）请求和响应的报文都得经过Director（在高负载应用场景中，Director很可能成为系统性能瓶颈）

5）支持端口映射

6）RS可以使用任意支持集群服务的操作系统

7）非高并发请求场景，10个RS以内。可隐藏内部的DIP和RIP地址

2、VS/TUN集群原理

2.1 VS/TUN模式原理描述

1.client 发送request包到LVS服务器的VIP上。

2.VIP按照算法选择后端的一个Real-server，并将记录一条消息到hash表中，然后将client的request包封装到一个新的IP包里，新IP包的目的IP是Real-server的IP，然后转发给Real-server。

3.Real-server收到包后，解封装，取出client的request包，发现他的目的地址是VIP，而Real-server发现在自己的 lo:0口上有这个IP地址，于是处理client的请求，然后real-server将直接relpy这个request包直接发给client。

4.该client的后面的request包，LVS直接按照hash表中的记录直接转发给Real-server，当传输完毕或者连接超时，那么将删除hash表中的记录。

由于通过IP Tunneling 封装后，封装后的IP包的目的地址为Real-server的IP地址，那么只要Real-server的地址能路由可达，Real-server在什么网络里都可以，这样可以减少对于公网IP地址的消耗，但是因为要处理IP Tunneling封装和解封装的开销，那么效率不如DR模式。

2.2 VS/TUN模式特点

1）RIP、DIP、VIP都得是公网地址

2）RS的网关不能指向DIP

3）请求报文经过Director，但响应报文一定不经过Director

4）不支持端口映射

5）RS的操作系统必须得支持隧道功能，即部署ipip模块

6）适用场景：跨互联网的请求转发

3、VS/DR集群原理

首先，来自客户端计算机CIP的请求被发送到Director的VIP。然后Director使用相同的VIP目的IP地址将请求发送到集群节点或真实服务器。然后，某个节点回复该数据包，并将该数据包直接发送到客户端计算机（不经过Director），并且以此回复数据包使用目的VIP地址作为源IP地址。因此，实际上客户计算机被“欺骗”了，客户计算机感觉它自己只和一台计算机对话。

3.1 VS/DR模式原理描述

1.首先，来自客户端计算机CIP的请求被发送到LVS调度器的VIP。

2.LVS调度器收到目标地址为VIP的客户端请求包后，将该数据包的MAC地址改为LVS调度器选择的某一台Real Server的MAC地址，并通过交换机（数据链路层）发给这台Real Server服务器。注意：此数据包的目的IP和源IP地址都没有发生改变。

3.Real Server的链路层收到数据报文后，会从链路层网上传给IP层。此时IP层发现此报文目的IP不是它的。所以，在Real Server上需要配置一VIP的loopback device。这样Real Server服务器的IP层收到数据包报文并验证目的IP为Real Server本地IP后，才会往上传送给传输层。之所以把VIP设置为loopback device，是因为loopback device是服务器本地使用的网络接口，对外是不可见的，不会跟LVS的VIP冲突。

4.Real Server处理完数据包后，将应答直接返回给客户端（此时数据包报文源地址为客户端请求报文的目的地址VIP，目的地址为客户端的CIP），返回数据包不再经过LVS调度器。因此，如果对外提供LVS负载均衡服务，则Real Server需要有公网ip（或公网网关），才能将应答包返回给客户。不过，Real Server最好带有公网IP，这样可以不经过网关直接返回给客户。如果多个Real Server使用同一个网关出口，网关会成为LVS架构的瓶颈，大大降低LVS的性能。

3.2 VS/DR模式特点

1）所有集群节点Real Server必须和Director在相同的物理网段（同一局域网）上。

2）所有客户入站（而不是出战）请求由Director首先接收，并转发给集群节点Real Server。

3）集群节点Real Server通常带有外部ip，而不使用Director及某固定机器作为默认网关。

4）所有集群节点Real Server上必须在lo网卡上绑定VIP地址，以便验证通过目的IP非Real Server的数据包。

5）所有集群节点Real Server上必须在lo网卡上绑定VIP地址，因此带来arp问题，即集群节点Real Server默认会响应本该发往Director VIP的数据包。因此要对所有集群节点Real Server做ARP抑制，把响应VIP的请求让给LVS Director。

6）很多操作系统都可以在集群内部的Real Server真实服务器上，只要该操作系统能够实现ARP隐藏。

7）LVS/DR模式不需要开启调度器转发功能，这点与LVS/NAT模式是不同。

8）LVS/DR Director可以比VS/NAT Director承受更多的并发请求更多的服务器数量。

9）不支持端口映射

三、LVS的调度算法

1、轮询（round robin, rr),加权轮询（Weighted round robin, wrr）

新的连接请求被轮流分配至各RealServer；算法的优点是其简洁性，它无需记录当前所有连接的状态，所以它是一种无状态调度。轮叫调度 算法假设所有服务器处理性能均相同，不管服务器的当前连接数和响应速度。该算法相对简单，不适用于服务器组中处理性能不一的情况，而且当请求服务时间变化 比较大时，轮叫调度算法容易导致服务器间的负载不平衡。

2、最少连接(least connected, lc)， 加权最少连接（weighted least connection, wlc）

新的连接请求将被分配至当前连接数最少的RealServer；最小连接调度是一种动态调度算法，它通过服务器当前所活跃的连接数来估计服务 器的负载情况。调度器需要记录各个服务器已建立连接的数目，当一个请求被调度到某台服务器，其连接数加1；当连接中止或超时，其连接数减一。

3、基于局部性的最少链接调度（Locality-Based Least Connections Scheduling，lblc）

针对请求报文的目标IP地址的负载均衡调度，目前主要用于Cache集群系统，因为在Cache集群中客户请求报文 的目标IP地址是变化的。这里假设任何后端服务器都可以处理任一请求，算法的设计目标是在服务器的负载基本平衡情况下，将相同目标IP地址的请求调度到同 一台服务器，来提高各台服务器的访问局部性和主存Cache命中率，从而整个集群系统的处理能力。LBLC调度算法先根据请求的目标IP地址找出该目标 IP地址最近使用的服务器，若该服务器是可用的且没有超载，将请求发送到该服务器；若服务器不存在，或者该服务器超载且有服务器处于其一半的工作负载，则 用“最少链接”的原则选出一个可用的服务器，将请求发送到该服务器。

4、带复制的基于局部性最少链接调度（Locality-Based Least Connections with Replication Scheduling，lblcr）

也是针对目标IP地址的负载均衡，目前主要用于Cache集群系统。它与LBLC算法的不同之处是它要维护从一个 目标IP地址到一组服务器的映射，而 LBLC算法维护从一个目标IP地址到一台服务器的映射。对于一个“热门”站点的服务请求，一台Cache 服务器可能会忙不过来处理这些请求。这时，LBLC调度算法会从所有的Cache服务器中按“最小连接”原则选出一台Cache服务器，映射该“热门”站 点到这台Cache服务器，很快这台Cache服务器也会超载，就会重复上述过程选出新的Cache服务器。这样，可能会导致该“热门”站点的映像会出现 在所有的Cache服务器上，降低了Cache服务器的使用效率。LBLCR调度算法将“热门”站点映射到一组Cache服务器（服务器集合），当该“热 门”站点的请求负载增加时，会增加集合里的Cache服务器，来处理不断增长的负载；当该“热门”站点的请求负载降低时，会减少集合里的Cache服务器 数目。这样，该“热门”站点的映像不太可能出现在所有的Cache服务器上，从而提供Cache集群系统的使用效率。LBLCR算法先根据请求的目标IP 地址找出该目标IP地址对应的服务器组；按“最小连接”原则从该服务器组中选出一台服务器，若服务器没有超载，将请求发送到该服务器；若服务器超载；则按 “最小连接”原则从整个集群中选出一台服务器，将该服务器加入到服务器组中，将请求发送到该服务器。同时，当该服务器组有一段时间没有被修改，将最忙的服 务器从服务器组中删除，以降低复制的程度。

5、目标地址散列调度（Destination Hashing，dh）

也是针对目标IP地址的负载均衡，但它是一种静态映射算法，通过一个散列（Hash）函数将一个目标IP地址映射到一台服务 器。目标地址散列调度算法先根据请求的目标IP地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

6、源地址散列调度（Source Hashing，sh）

正好与目标地址散列调度算法相反，它根据请求的源IP地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。它采用的散列函数与目标地址散列调度算法 的相同。除了将请求的目标IP地址换成请求的源IP地址外，它的算法流程与目标地址散列调度算法的基本相似。在实际应用中，源地址散列调度和目标地址散列 调度可以结合使用在防火墙集群中，它们可以保证整个系统的唯一出入口。

7、加权最少连接调度（Weighted Least Connections）

“加权最少连接调度”是“最少连接调度”的超集。每个服务节点可以用相应的权值表示其处理能力，而系统管理员可以动态地设置相应的权值，默认权值为1。加权最 小连接调 度在分新连接请求时尽可能使服务节点的已建立连接数和其权值成正比。

算法： overhead=active\*256+inactive/weight overhead最小值胜出

8、最小期望延迟(改进的wlc)（shorttest expect delay，sed）

overhead=(active+1)\*256/weight，案例：假如DFG三台机器分别权重123，连接数也分别是123.那么如果使用WLC算法的话一个新请求进入时它可能会分给DFG中的任意一个。

使用sed算法后会进行这样一个运算：

D(1+1)/1

F(1+2)/2

G(1+3)/3

9、nq：nerver queue

增强改进的sed算法。如果有台real Server的连接数=0直接分配,不需要再进行sed运算。

四、安装LVS

4.1准备环境

|  |
| --- |
| lvsd.omcn.org：CentOS release 6.4 (Final) 2.6.32-358.el6.x86\_64  rs1.omcn.org：CentOS release 6.4 (Final) 2.6.32-358.el6.x86\_64  rs2.omcn.org：CentOS release 6.4 (Final) 2.6.32-358.el6.x86\_64  LVS：ipvsadm-1.26.tar.gz  [root@lvsd ~]# lsmod | grep ip\_vs #检查本机是否已经安装过  [root@lvsd ~]# |

1）数据库及memcache等负载均衡环境

|  |  |  |
| --- | --- | --- |
| IP地址 | 角色 | 备注 |
| 192.168.1.12 | LVS调度器（Director） | 对外提供服务VIP：192.168.1.13 |
| 192.168.1.18 | RS1 |  |
| 192.168.1.19 | RS2 |  |

提示：以上环境为内部环境的负载均衡模式，即LVS服务是对内部业务的，如数据库及memcache等的负载均衡。

2）web或web cache等负载均衡环境

|  |  |  |  |
| --- | --- | --- | --- |
| 外部IP地址 | 内部IP地址 | 角色 | 备注 |
| 192.168.1.12 | 10.0.0.12 | LVS调度器（Director） | 对外提供服务VIP：192.168.1.13 |
| 192.168.1.18 | 10.0.0.18 | RS1 |  |
| 192.168.1.19 | 10.0.0.19 | RS2 |  |

提示：以上环境一般是提供web或web cache负载均衡的情况，此情况特点为双网卡环境，这里把10.0.0.0/24假设为内网卡，192.168.1.0/24假设为外网卡。因为笔记本的ip是192网段，所以当做外网测试更方便。

4.2搭建Director环境

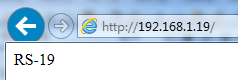
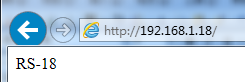
安装LVS软件。

|  |
| --- |
| **[root@lvsd ~]# ln -s /usr/src/kernels/2.6.32-358.el6.x86\_64 /usr/src/linux**  [root@lvsd ~]# ls -ld /usr/src/linux  lrwxrwxrwx. 1 root root 38 Aug 13 20:24 /usr/src/linux -> /usr/src/kernels/2.6.32-358.el6.x86\_64  **[root@lvsd ~]# rpm -qa| grep kernel-devel**  kernel-devel-2.6.32-358.el6.x86\_64  [root@lvsd Downloads]# ll -sh ipvsadm-1.26.tar.gz  64K -rw-r--r--. 1 root root 61K Aug 13 19:51 ipvsadm-1.26.tar.gz  **[root@lvsd Downloads]# tar xf ipvsadm-1.26.tar.gz && cd ipvsadm-1.26**  [root@lvsd ipvsadm-1.26]# make #报错  make -C libipvs  make[1]: Entering directory `/root/Downloads/ipvsadm-1.26/libipvs'  gcc -Wall -Wunused -Wstrict-prototypes -g -fPIC -DLIBIPVS\_USE\_NL -DHAVE\_NET\_IP\_VS\_H -c -o libipvs.o libipvs.c  gcc -Wall -Wunused -Wstrict-prototypes -g -fPIC -DLIBIPVS\_USE\_NL -DHAVE\_NET\_IP\_VS\_H -c -o ip\_vs\_nl\_policy.o ip\_vs\_nl\_policy.c  libipvs.c:1072: error: too many arguments to function ‘ipvs\_nl\_send\_message’ make[1]: \*\*\* [libipvs.o] Error 1 make[1]: Leaving directory ` /root/Downloads/ipvsadm-1.26/libipvs ' make: \*\*\* [libs] Error 2  **[root@lvsd ipvsadm-1.26]# yum install popt popt-devel libnl libnl-devel -y**  [root@lvsd ipvsadm-1.26]# make #报错  make -C libipvs  make[1]: Entering directory `/root/Downloads/ipvsadm-1.26/libipvs'  make[1]: Nothing to be done for `all'.  make[1]: Leaving directory `/root/Downloads/ipvsadm-1.26/libipvs'  gcc -Wall -Wunused -Wstrict-prototypes -g -o ipvsadm ipvsadm.o config\_stream.o dynamic\_array.o libipvs/libipvs.a -lnl  ipvsadm.o: In function `parse\_options':  /root/Downloads/ipvsadm-1.26/ipvsadm.c:395: undefined reference to `poptGetContext'  /root/Downloads/ipvsadm-1.26/ipvsadm.c:398: undefined reference to `poptGetNextOpt'  /root/Downloads/ipvsadm-1.26/ipvsadm.c:615: undefined reference to `poptBadOption'  /root/Downloads/ipvsadm-1.26/ipvsadm.c:465: undefined reference to `poptGetNextOpt'  /root/Downloads/ipvsadm-1.26/ipvsadm.c:622: undefined reference to `poptStrerror'  /root/Downloads/ipvsadm-1.26/ipvsadm.c:622: undefined reference to `poptBadOption'  /root/Downloads/ipvsadm-1.26/ipvsadm.c:625: undefined reference to `poptFreeContext'  /root/Downloads/ipvsadm-1.26/ipvsadm.c:632: undefined reference to `poptGetArg'  /root/Downloads/ipvsadm-1.26/ipvsadm.c:633: undefined reference to `poptGetArg'  /root/Downloads/ipvsadm-1.26/ipvsadm.c:634: undefined reference to `poptGetArg'  /root/Downloads/ipvsadm-1.26/ipvsadm.c:645: undefined reference to `poptGetArg'  /root/Downloads/ipvsadm-1.26/ipvsadm.c:648: undefined reference to `poptFreeContext'  collect2: ld returned 1 exit status  make: \*\*\* [ipvsadm] Error 1  **[root@lvsd Downloads]# rpm -ivh popt-static-1.13-7.el6.x86\_64.rpm**  warning: popt-static-1.13-7.el6.x86\_64.rpm: Header V3 RSA/SHA256 Signature, key ID c105b9de: NOKEY  Preparing... ########################################### [100%]  1:popt-static ########################################### [100%]  [root@lvsd Downloads]# cd ipvsadm-1.26  **[root@lvsd ipvsadm-1.26]# make**  make -C libipvs  make[1]: Entering directory `/root/Downloads/ipvsadm-1.26/libipvs'  make[1]: Nothing to be done for `all'.  make[1]: Leaving directory `/root/Downloads/ipvsadm-1.26/libipvs'  gcc -Wall -Wunused -Wstrict-prototypes -g -o ipvsadm ipvsadm.o config\_stream.o dynamic\_array.o libipvs/libipvs.a -lpopt -lnl  **[root@lvsd ipvsadm-1.26]# make install**  **[root@lvsd ipvsadm-1.26]# lsmod | grep ip\_vs**  [root@lvsd ipvsadm-1.26]#  提示：安装成功后，ipvs模块没有被加载到内核，可以执行“ipvsadm”命令或者“modprobe ip\_vs”命令把ip\_vs模块加载到内核。  **[root@lvsd ipvsadm-1.26]# ipvsadm**  IP Virtual Server version 1.2.1 (size=4096)  Prot LocalAddress:Port Scheduler Flags  -> RemoteAddress:Port Forward Weight ActiveConn InActConn  [root@lvsd ipvsadm-1.26]# lsmod | grep ip\_vs  ip\_vs 115643 0  libcrc32c 1246 1 ip\_vs  ipv6 321422 145 ip\_vs,cnic,ip6t\_REJECT,nf\_conntrack\_ipv6,nf\_defrag\_ipv6 |

4.3搭建Real Server环境

在RS1及RS2上搭建web服务器

|  |
| --- |
| [root@rs1 ~]# yum install -y httpd  [root@rs1 ~]# echo 'RS-18'>/var/www/html/index.html  #####  [root@rs2 ~]# yum install -y httpd  [root@rs2 ~]# echo 'RS-19'>/var/www/html/index.html |



提示：测试环境下RS1和RS2内容不同为了容易看懂。实际情况下，RS1和RS2内容应相同。

4.5 VS/DR模式

4.5.1配置LVS Director

1.给LVS-Director添加VIP

|  |
| --- |
| [root@lvsd ~]# ifconfig eth0:0 192.168.1.13 netmask 255.255.255.255 up  [root@lvsd ~]# ifconfig eth0:0  eth0:0 Link encap:Ethernet HWaddr 00:0C:29:C2:03:71  inet addr:192.168.1.13 Bcast:192.168.1.255 Mask:255.255.255.0  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 |

2.手工执行配置添加LVS服务并增加两台RS

|  |
| --- |
| [root@lvsd ~]# ipvsadm -C #clear the whole table  [root@lvsd ~]# ipvsadm --set 30 5 60 #set tcp(connection) tcpfin(timeout) udp(values)  [root@lvsd ~]# ipvsadm -A -t 192.168.1.13:80 -s wrr -p 20 #-t VIP,-p timeout保持会话  #以下添加两个Real Server  [root@lvsd ~]# ipvsadm -a -t 192.168.1.13:80 -r 192.168.1.18:80 -g -w 1  [root@lvsd ~]# ipvsadm -a -t 192.168.1.13:80 -r 192.168.1.19:80 -g -w 1  提示：-a 添加，-w权重，-g直接路由，-m NAT模式。  [root@lvsd ~]# ipvsadm -L -n  IP Virtual Server version 1.2.1 (size=4096)  Prot LocalAddress:Port Scheduler Flags  -> RemoteAddress:Port Forward Weight ActiveConn InActConn  TCP 192.168.1.13:80 wrr persistent 20  -> 192.168.1.18:80 Route 1 0 0  -> 192.168.1.19:80 Route 1 0 0 |

提示：删除Real Server及取消LVS服务

|  |
| --- |
| ipvsadm -d -t 192.168.1.13:80 -r 192.168.1.18:80 #删除Real Server  ipvsadm -D -t 192.168.1.13:80 -s wrr |

4.5.2配置Real Server

1.每台Real Server绑定VIP

|  |
| --- |
| [root@rs1 ~]# ifconfig lo:13 192.168.1.13 netmask 255.255.255.255 up  [root@rs1 ~]# route add -host 192.168.1.13 dev lo  [root@rs2 ~]# ifconfig lo:13 192.168.1.13 netmask 255.255.255.255 up  [root@rs2 ~]# route add -host 192.168.1.13 dev lo |

提示：每个集群节点上的环回接口（lo）设备上被绑定VIP地址（其广播地址是其本身，子网掩码是255.255.255.255，采取可变长掩码方式把网段划分成只含一个主机地址的目的是避免ip地址冲突）允许VS/DR集群中的集群节点接收发向该VIP地址的数据包。这会引起一个严重问题，集群内部的Real Server将尝试回复来自正在请求VIP客户端的ARP广播，这样所有的Real Server都声称自己拥有该VIP地址，这时客户端直接发送请求到Real Server从而破坏了DR集群的方法。因此，必须要抑制Real Server的ARP广播。

2.在Real Server上抑制ARP响应

|  |
| --- |
| [root@rs1 ~]# echo "1" >/proc/sys/net/ipv4/conf/lo/arp\_ignore  [root@rs1 ~]# echo "2" >/proc/sys/net/ipv4/conf/lo/arp\_announce  [root@rs1 ~]# echo "1" >/proc/sys/net/ipv4/conf/all/arp\_ignore  [root@rs1 ~]# echo "2" >/proc/sys/net/ipv4/conf/all/arp\_announce |

4.6 VS/NAT模式

4.6.1配置LVS Director

|  |
| --- |
| [root@lvsd ~]# ifconfig eth0:0 192.168.1.13 netmask 255.255.255.255 up  [root@lvsd ~]# ifconfig eth0:0  eth0:0 Link encap:Ethernet HWaddr 00:0C:29:C2:03:71  inet addr:192.168.1.13 Bcast:192.168.1.255 Mask:255.255.255.0  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 |

|  |
| --- |
| [root@lvsd ~]# ipvsadm -C #clear the whole table  [root@lvsd ~]# ipvsadm --set 30 5 60 #set tcp(connection) tcpfin(timeout) udp(values)  [root@lvsd ~]# ipvsadm -A -t 192.168.1.13:80 -s lblc -p 20 #-t VIP,-p timeout保持会话  #以下添加两个Real Server  [root@lvsd ~]# ipvsadm -a -t 192.168.1.13:80 -r 192.168.1.18:80 -m -w 1  [root@lvsd ~]# ipvsadm -a -t 192.168.1.13:80 -r 192.168.1.19:80 -m -w 1 |

4.6.2配置Real Server

将real server的默认网关地址设置为VIP。

|  |
| --- |
| [root@rs1 ~]# route add default gw 192.168.1.13  [root@rs2 ~]# route add default gw 192.168.1.13 |

4.7检验LVS运行

该检测结果是基于DR模式。

|  |
| --- |
| [root@lvsd ~]# ipvsadm -L -n --stats  IP Virtual Server version 1.2.1 (size=4096)  Prot LocalAddress:Port Conns InPkts OutPkts InBytes OutBytes  -> RemoteAddress:Port  TCP 192.168.1.13:80 4 20 0 2228 0  -> 192.168.1.18:80 2 10 0 1114 0  -> 192.168.1.19:80 2 10 0 1114 0  [root@lvsd ~]# ipvsadm -Ln --timeout |

上表显示，RS1和RS2访问次数相同（权重相同）。

五、常见的相关问题

5.1 ipvsadm命令指南

ipvsadm -A|E -t|u|f virutal-service-address:port [-s scheduler] [-p [timeout]] [-M netmask]

ipvsadm -D -t|u|f virtual-service-address

ipvsadm -C

ipvsadm -R

ipvsadm -S [-n]

ipvsadm -a|e -t|u|f service-address:port -r real-server-address:port [-g|i|m] [-w weight]

ipvsadm -d -t|u|f service-address -r server-address

ipvsadm -L|l [options]

ipvsadm -Z [-t|u|f service-address]

ipvsadm --set tcp tcpfin udp

ipvsadm --start-daemon state [--mcast-interface interface]

ipvsadm --stop-daemon

ipvsadm -h

命令选项解释：

有两种命令选项格式，长的和短的，具有相同的意思。在实际使用时，两种都可以。

-A --add-service 在内核的虚拟服务器表中添加一条新的虚拟服务器记录。也就是增加一台新的虚拟服务器。

-E --edit-service 编辑内核虚拟服务器表中的一条虚拟服务器记录。 修改定义过的集群服务

-D --delete-service 删除内核虚拟服务器表中的一条虚拟服务器记录。

-C --clear 清除内核虚拟服务器表中的所有记录。

-R --restore 恢复虚拟服务器规则

-S --save 保存虚拟服务器规则

-a --add-server 在内核虚拟服务器表的一条记录里添加一条新的真实服务器记录。也就是在一个虚拟服务器中增加一台新的真实服务器

-e --edit-server 编辑一条虚拟服务器记录中的某条真实服务器记录

-d --delete-server 删除一条虚拟服务器记录中的某条真实服务器记录

-L|-l --list 显示内核虚拟服务器表 userver 列表；

-Z --zero 虚拟服务表计数器清零（清空当前的连接数量等）

--set tcp tcpfin udp 设置连接超时值

--start-daemon 启动同步守护进程。他后面可以是master 或backup，用来说明LVS Router 是master 或是backup。在这个功能上也可以采用keepalived 的VRRP 功能。

--stop-daemon 停止同步守护进程

-h --help 显示帮助信息

其他的选项:

-t --tcp-service service-address 说明虚拟服务器提供的是tcp 的服务[vip:port] or [real-server-ip:port]

-u --udp-service service-address 说明虚拟服务器提供的是udp 的服务[vip:port] or [real-server-ip:port]

-f --fwmark-service fwmark 说明是经过iptables 标记过的服务类型。通常用于将两个或两个以上的服务绑定为一个服务进行处理时使用；

-s --scheduler scheduler 使用的调度算法，有这样几个选项 rr|wrr|lc|wlc|lblc|lblcr|dh|sh|sed|nq,默认的调度算法是：wlc.

-p --persistent [timeout] 持久稳固的服务。这个选项的意思是来自同一个客户的多次请求，将被同一台真实的服务器处理。timeout 的默认值为300 秒。 持久连接；

-M --netmask netmask persistent granularity mask

-r --real-server server-address 真实的服务器[Real-Server:port]

-g --gatewaying 指定LVS 的工作模式为直接路由模式（也是LVS 默认的模式） DR模型

-i --ipip 指定LVS 的工作模式为隧道模式

-m --masquerading 指定LVS 的工作模式为NAT 模式

-w --weight weight 真实服务器的权值

--mcast-interface interface 指定组播的同步接口

-c --connection 显示LVS 目前的连接 如：ipvsadm -L -c

--timeout 显示tcp tcpfin udp 的timeout 值 如：ipvsadm -L --timeout

--daemon 显示同步守护进程状态

--stats 显示统计信息 ipvsadm -Ln --state 总共的数量

--rate 显示速率信息 ipvsadm -Ln --rete 平均值

--sort 对虚拟服务器和真实服务器排序输出

--numeric -n 输出IP 地址和端口的数字形式

**集群相关的命令参数**：

-A --add-service 在内核的虚拟服务器表中添加一条新的虚拟服务器记录。也就是增加一台新的虚拟服务器。

-t --tcp-service service-address 说明虚拟服务器提供的是tcp 的服务[vip:port] or [real-server-ip:port]

-u --udp-service service-address 说明虚拟服务器提供的是udp 的服务[vip:port] or [real-server-ip:port]

-s --scheduler scheduler 使用的调度算法，有这样几个选项 rr|wrr|lc|wlc|lblc|lblcr|dh|sh|sed|nq,默认的调度算法是：wlc.

-p --persistent [timeout] 持久稳固的服务。这个选项的意思是来自同一个客户的多次请求，将被同一台真实的服务器处理。timeout 的默认值为300 秒。 持久连接；

-E --edit-service 编辑内核虚拟服务器表中的一条虚拟服务器记录。 修改定义过的集群服务

-D ：删除指定集群服务

**RS相关的命令参数：**

-a --add-server 在内核虚拟服务器表的一条记录里添加一条新的真实服务器记录。也就是在一个虚拟服务器中增加一台新的真实服务器 向指定的CS中添加RS

-r --real-server server-address 真实的服务器[Real-Server:port]，只有支持端口映射的LVS类型才允许此处使用跟集群服务中不同的端口

LVS 类型：

-g ： gateway，DR # 指定集群类型为LVS/DR

-i ipip，TUN # 指定集群类型为LVS/TUN

-m：masquerade，NAT # 指定集群类型为 NAT

-w：指定RS权重：

-e：修改指定的RS属性

-d ：从指定的集群服务中删除某RS

5.2 arp问题

VS/DR或VS/TUN 应用的一种模型中（所有机器都在同一个物理网络），所有机器（包括Director和RealServer）都使用了一个额外的IP地址，即VIP。当一 个客户端向VIP发出一个连接请求时，此请求必须要连接至Director的VIP，而不能是RealServer的。因为，LVS的主要目标就是要 Director负责调度这些连接请求至RealServer的。

因此，在Client发出至VIP的连接请求后，只能由Director将其 MAC地址响应给客户端（也可能是直接与Director连接的路由设备），而Director则会相应的更新其ipvsadm table以追踪此连接，而后将其转发至后端的RealServer之一。

如果Client在请求建立至VIP的连接时集群内部的Real Server将尝试回复来自正在请求VIP客户端的ARP广播，这样所有的Real Server都声称自己拥有该VIP地址，这时客户端直接发送请求到Real Server从而破坏了DR集群的方法。

为了解决此问题，可以通过在路由器上设置其转发规则来实现。当然，如果没有权限访问路由器并做出相应的设置，则只能通过传统的本地方式来解决此问题了。

这些方法包括：

1、禁止RealServer响应对VIP的ARP请求

2、在RealServer上隐藏VIP，以使得它们无法获知网络上的ARP请求

3、基于“透明代理（Transparent Proxy）”或者“fwmark （firewall mark）”

4、禁止ARP请求发往RealServers

幸运的是，2.4.26和2.6的内核中引入了两个新的 调整ARP栈的标志（device flags）：arp\_announce和arp\_ignore。基于此，在DR/TUN的环境中，所有IPVS相关的设定均可使用 arp\_announce=2和arp\_ignore=1/2/3来解决“ARP问题”了。

arp\_annouce：Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface；

0 - (default) Use any local address, configured on any interface.

1 - Try to avoid local addresses that are not in the target's subnet for this interface.

2 - Always use the best local address for this target.

arp\_ignore: Define different modes for sending replies in response to received ARP requests that resolve local target IP address.

0 - (default): reply for any local target IP address, configured on any interface.

1 - reply only if the target IP address is local address configured on the incoming interface.

2 - reply only if the target IP address is local address configured on the incoming interface and both with the sender's IP address are part from same subnet on this interface.

3 - do not reply for local address configured with scope host, only resolutions for golbal and link addresses are replied.

4-7 - reserved

8 - do not reply for all local addresses

中文：

**arp\_announce**

定义了网卡在向外宣告自己的MAC-IP时候的限制级别，有三个值：

0：默认值，不管哪块网卡接收到了ARP请求，只要发现本机有这个MAC都给与响应

1：尽量避免响应ARP请求中MAC不是本网卡的，一个主机有多块网卡，其中一块网卡接收到了ARP请求，发现所请求的MAC是本机另一块网卡的，这个时候接收到ARP请求的这块网卡就尽量避免响应

2：总是使用最合适的网卡来响应，一个主机有多块网卡，其中一块网卡接收到了ARP请求，发现所请求的MAC是本机另一块网卡的，这个时候接收到ARP请求的这块网卡就一定不响应，只有发现请求的MAC是自己的才给与响应

**arp\_ignore**

定义了网卡在响应外部ARP请求时候的响应级别，这里有8个值，但我们只使用了2个。

0：默认值，不管哪块网卡接收到了ARP请求，只要发现本机有这个MAC都给与响应

1：总是使用最合适的网卡来响应，一个主机有多块网卡，其中一块网卡接收到了ARP请求，发现所请求的MAC是本机另一块网卡的，这个时候接收到ARP请求的这块网卡就一定不响应，只有发现请求的MAC是自己的才给与响应

|  |
| --- |
| [root@rs1 ~]# vim /etc/sysctl.conf  net.ipv4.conf.eth0.arp\_ignore = 1  net.ipv4.conf.eth0.arp\_announce = 2  net.ipv4.conf.all.arp\_ignore = 1  net.ipv4.conf.all.arp\_announce = 2 |

5.3空闲超时时间

为了保证其时效性，Hash table中“连接追踪”信息被定义了“生存时间”。LVS为记录“连接超时”定义了三个计时器

1、空闲TCP会话（tcp的空闲超时时间，同内核参数tcp\_fin\_timeout）

2、客户端正常断开连接后的TCP会话（lvs收到客户端tcp fin的超时时间）

3、无连接的UDP数据包（记录其两次发送数据包的时间间隔）

如何设置：

# ipvsadm --set 60 30 600

5.4 LVS的持久连接

持久连接即不考虑LVS的转发方法，确保把同一个用户的连接转发到同一个RealServer去。原理即把同一个client的请求信息记录到lvs的hash表里，保存时间使用persistence\_timeout控制，单位为秒。persistence\_granularity 参数是配合persistence\_timeout的，在某些情况特别有用，他的值是子网掩码，表示持久连接的粒度，默认是255.255.255.255，也就是单独的client ip，如果改成255.255.255.0就是client ip一个网段的都会被分配到同一个real server。

如何设置：

# ipvsadm -A -t 192.168.0.13:80 -s rr -p N

-p # 表示此连接为持久连接

N # 表示维持此持久连接的时间，默认6分钟。当超过这个时间后，如果网页还没有关掉，仍处于激活状态，重新复位时间为2分钟。

# ipvsadm -Lcn

工作原理：

当一个client访问vip的时候，ipvs或记录一条状态为NONE的信息，expire初始值是persistence\_timeout的值，然后根据时钟主键变小，在以下记录存在期间，同一client ip连接上来，都会被分配到同一个后端。

FIN\_WAIT的值就是tcp tcpfin udp的超时时间，当NONE的值为0时，如果FIN\_WAIT还存在，那么NONE的值会从新变成60秒，再减少，直到FIN\_WAIT消失以后，NONE才会消失，只要NONE存在，同一client的访问，都会分配到统一real server。

5.5关闭网卡LRO和GRO

现在大多数网卡都具有LRO/GRO功能，即 网卡收包时将同一流的小包合并成大包 （tcpdump抓包可以看到>MTU 1500bytes的数据包）交给 内核协议栈；LVS内核模块在处理>MTU的数据包时，会丢弃；

因此，如果我们用LVS来传输大文件，很容易出现丢包，传输速度慢；

解决方法，关闭LRO/GRO功能，命令：

ethtool -k eth0 查看LRO/GRO当前是否打开

ethtool -K eth0 lro off 关闭GRO

ethtool -K eth0 gro off 关闭GRO

禁用ARP，增大backlog并发数

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

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

net.core.netdev\_max\_backlog = 500000 （在每个网络接口接收数据包的速率比内核处理这些包的速率快时，允许送到队列的数据包的最大数目）