**LVS负载均衡**

**LVS介绍**

LVS是Linux Virtual Server 的简写（也叫做IPVS），

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。

**适用性**

后端真实服务器可运行任何支持TCP/IP的操作系统，包括Linux，各种Unix（如FreeBSD，Sun Solaris，HP Unix等），Mac/OS和windows NT/2000等。

负载均衡调度器LB能够支持绝大多数的TCP和UDP协议：

TCP ：HTTP，FTP，PROXY，SMTP，POP3，IMAP4，DNS，LDAP，HTTPS，SSMTP等

UDP ：DNS，NTP，TCP，视频，音频流播放协议等

调度器本身当前不支持windows系统。支持大多数的Linux和UINIX系统

**性能**

LVS服务器集群系统具有良好的伸缩性，可支持几百万个并发连接。配置100M网卡，采用VS/TUN或VS/DR调度技术，集群系统的吞吐量可高达1Gbits/s；如配置千兆网卡，则系统的最大吞吐量可接近10Gbits/s

**突破LVS瓶颈，LVS Cluster部署（OSPF + LVS）**

1.LVS调度机自由伸缩，横向扩展（最大8台，受限于三层设备允许的等价路由数目）

2.LVS调度资源全利用，All Active。不存在备份机

<https://my.oschina.net/lxcong/blog/143904>

3、跨机房可以用NDS调度、同一个机房可以用OSPF调度LVS

**可靠性**

LVS服务器集群软件已经在很多大型的，关键性的站点得到很好的应用，所以它的可靠性在真实应用得到很好的证实。

**软件许可证**

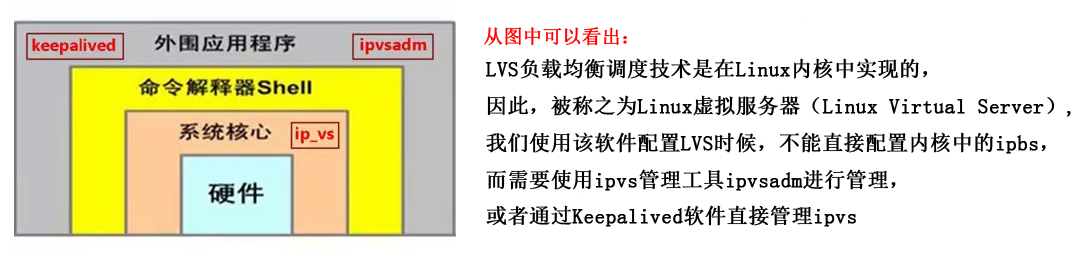
LVS集群软件是按GPL（GNU Public License）许可证发行的自由软件，这意味着你可以得到软件的源代码，有权对其进行修改，但必须保证你的修改也是以GPL方式发行。

**LVS应用场景说明**

LVS服务是对内部业务的，如数据库及memcache等的负载均衡、内网即可

提供Web或Web cache负载均衡的情况，此种情况特点为双网卡环境，最好一个外网地址一个内网地址。

**IPVS软件工作层次图**



**真正实现调度的工具是IPVS。工作再Linux内核层面**

**LSV自带的IPVS管理工具是ipvsadm**

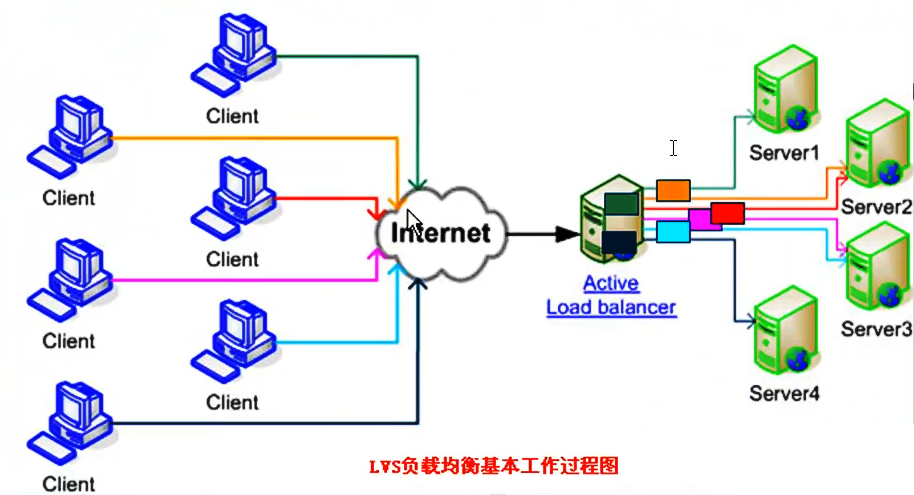
**keepalived实现管理IPVS及负载均衡器的高可用**

**LVS体系结构与工作原理简单描述**

LVS集群负载均衡器接受服务的所有入站客户端计算机请求，并根据调度算法决定哪个集群节点应该处理回复请求。负载均衡器（简称LB）有时也被称为LVS Director（简称Director）。

LVS虚拟服务器的体系结构如下图所示，一组服务器通过高速的局域网或者地理分布的广域网相互连接，在他们的前端有一个负载调度器（Load Balancer）。 负载调度器能无缝地将网络请求调度到真实服务器上，从而使得服务器集群的结构对客户是透明的，客户访问集群系统提供的网络服务就像访问一台高性能，高可用的服务器一样。客户程序不受服务器集群的影响不需要作任何修改。系统的伸缩性通过在服务集群中透明地加入和删除一个节点来达到，通过检测节点或服务进程故障和正确地重置系统达到高可用性。由于我们的负载调度技术是在Linux内核中实现的，我们称之为Linux虚拟服务器（Linux Virtual Server）。

**LVS 基本工作过程图**

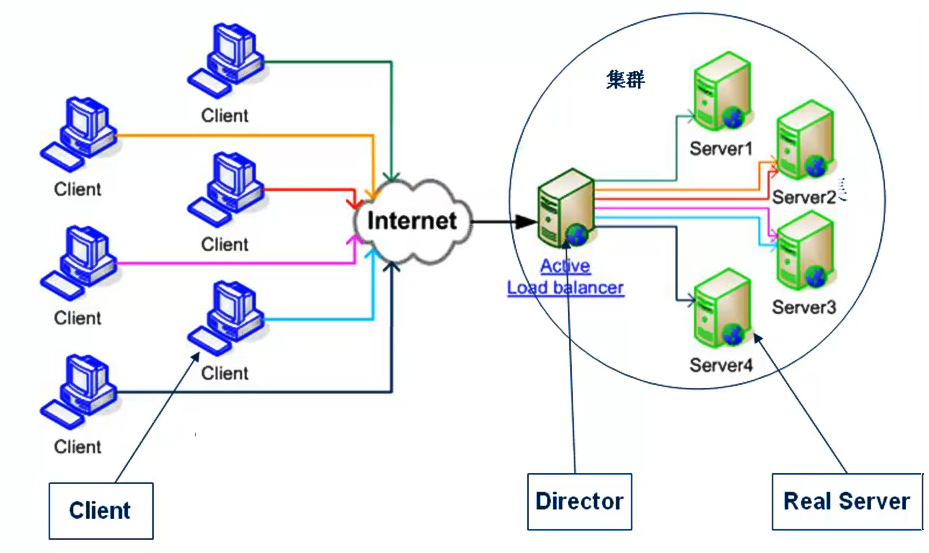


**为了方便大家探讨LVS技术，LVS社区提供了一个命名的约定，内容如下表**

| **名称** | **缩写** | **说明** |
| --- | --- | --- |
| Director Server | DS | 指的是前端负载均衡器节点 |
| Real Server | RS | 后端真实的工作服务器 |
| 虚拟IP | VIP | VIP为Director用于向客户端计算机提供服务的IP地址。比如：www.yunjisuan.com域名就要解析到vip上提供服务 |
| 真实IP地址 | RIP | 在集群下面节点上使用的IP地址，物理IP地址 |
| Dirctor的IP地址 | DIP | Director用于连接内外网络的IP地址，物理网卡上的IP地址。是负载均衡器上的IP |
| 客户端主机IP地址 | CIP | 客户端用户计算机请求集群服务器的IP地址，该地址用作发送给集群的请求的源IP地址 |

LVS集群内部的节点称为真实服务器（Real Server）,也叫做集群节点。

请求集群服务的计算机称为客户端计算机。  
与计算机通常在网上交换数据包的方式相同，客户端计算机，Director和真实服务器使用IP地址彼此进行通信。  
不同架构角色命名情况如下图



**LVS集群的3种常见工作模式介绍与原理讲解**

**IP虚拟服务器软件 IPVS**

在调度器的实现技术中，IP负载均衡技术是效率最高的。

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，

我们称之为VS/NAT技术（Virtual Server via Network Address Translation），

大多数商业化的IP负载均衡调度器产品都是使用NAT的方法，如Cisco的LocalDirector，F5，Netscaler的Big/IP和Alteon的ACEDirector

在分析VS/NAT 的缺点和网络服务的非对称性的基础上，我们提出通过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），他们可以极大地提高系统的伸缩性。

所以，IPVS软件实现了这三种IP负载均衡技术。

淘宝开源的模式FULLNAT

**LVS的四种工作模式**

#NAT模式

NAT（Network Address Translation）

#隧道模式

TUN（Tunneling）

**#路由模式**

**DR（Direct Routing）**

#淘宝开源的FULL模式

FULLNAT（Full Network Address Translation）

**LVS负载均衡平滑上线思路**

先使用：ipvsadm -d -t 10.10.10.12:80 -r 10.10.10.21:80 删除节点—测试—测试没问题推送到各个节点-

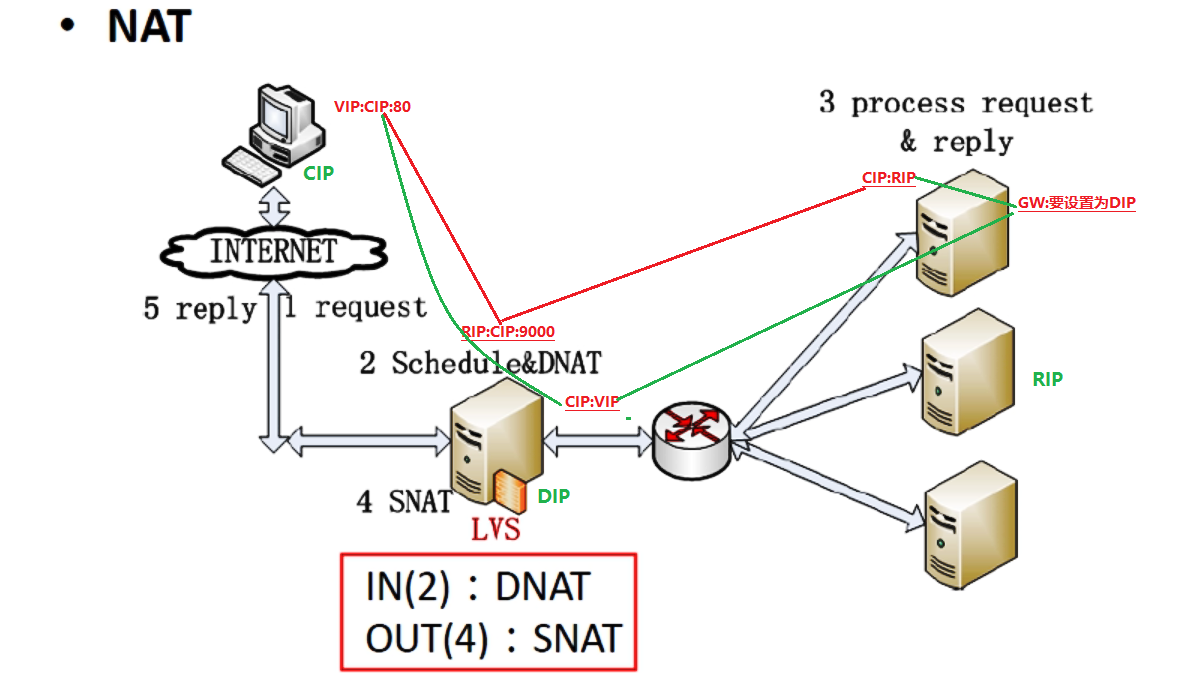
然后：ipvsadm -a -t 10.10.10.12:80 -r 10.10.10.21:80 -g -w 1 加入节点

最后提供服务

**谨记：不随意更改配置文件**

**NAT模式**

通过网络地址转换，调度器LB重写请求的报文目标地址，根据预设的调度算法，将请求分配给后端的真实服务器，真实服务器响应报文通过调度器时，报文的源地址被重写，再返回给客户端，完成整个负载调度过程



1、NAT技术将请求的报文（DNAT)和响应的报文（SNAT)，通过调度器地址重写然后 在转发发给内部的服务器，报文返回时在改写成原来的地址。

2、只需要在调度器LB上配置WAN公网IP即可，调度器也要有私有LAN IP和内部节点通信，内部RS节点只需配置私有LAN IP

3、每台内部RS节点的网关地址，必须要配成调度器LB的私有LAN内物理网卡地址 (LDIP),这样才能确保数据报文返回时仍然经过调度器LB

4、由于请求与响应的数据报文都经过调度器LB,因此，网站访问量大时调度器LB有较大瓶颈，一般要求最多10-20台节点

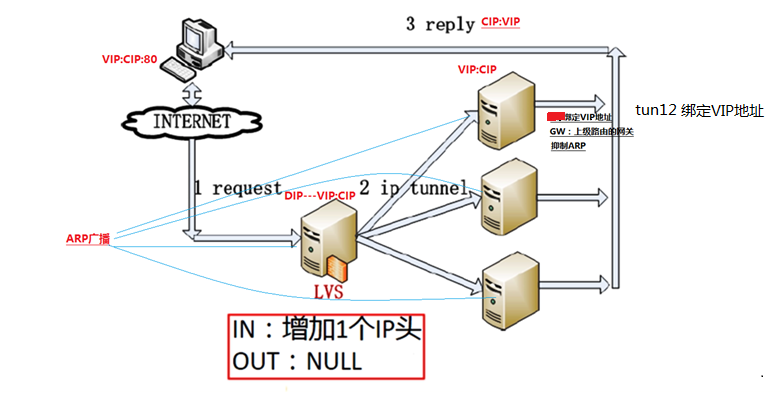
5、NAT模式支持对IP及端口的转换，即用户请求10.0.0.1:80,可以通过调度器转换到RS节点的10.0.0.2:8080

6、由于数据包来回都需要经过调度器，因此要开启内核转发 net.ipv4\_ip\_forward = 1

**TUN（Tunneling）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)转发至真实服努器**，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的工作流程如下图所示：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，动态地选择一台服务器，将请求报文封装在另一个IP报文中，再将封装后的IP报文转发给选出的真实服务器，服务器收到报文后，先将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（tun12：绑定VIP和DR模式有点区别，DR模式绑定到lo上），所以就处理这个请求、然后根据路由表响应报文直接返给客户端。

****

1、负载均衡器通过把请求的报文通过IP隧道（相当于ipip或ipsec隧道）的方式（把请求的报文不经过原目的地址的改写，而是直接封装成另外的IP报文）转发给真实服务器，而真实服务器响应处理后直接返回给客户端

2、由于真实服务器响应处理后直接返回给客户端，因此RS最好有一个外网IP地址，这样效率高，理论上：只要能出网即可，可以不要公网IP

3、由于调度器LB只处理入站请求的报文，因此集群系统的吞吐量可以提高10倍以上。但隧道模式也会带来一定的系统开销，TUN模式适合LAN/WAN

4、LAN环境不如DR模式效率高，有的系统还要考虑IP隧道支持的问题，

5、所有的RS服务器都要在tun10上绑定VIP，抑制ARP

6、LAN环境一般多采用DR，LAN环境可以使用TUN模式，但是在WAL环境情况下，请求转发，更多的被Haproxy/nginx/DNS调度等代理取代，因此TUN模式使用量很少

**DR模式**

VS/DR模式是通过改写请求报文的目标MAC地址，将请求转发给真实服务器，而真实服务器响应后的处理结果直接返回给客户端，

同VS/UNN技术一样，VS/DR技术，可以极大提高集群系统的伸缩性，而且DR模式没有IP隧道开销，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LV与真实服务器RS都要有一块网卡连在同一物理网段上。**

1、通过在调度器LB上修改数据包的目的MAC地址实现转发。（源IP地址仍然是CIP，目的IP地址仍然是VIP）

2、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（和NAT模式相比）

3、因DR模式是通过MAC地址的改写机制实现的转发，因此，所有RS节点和调度器LB只能在一个局域网LAN中（缺点）

4、RS节点的默认网关不需要是调度器LB的DIP，而直接是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），理论讲：只要RS可以出网即可，不是必须要配置外网IP

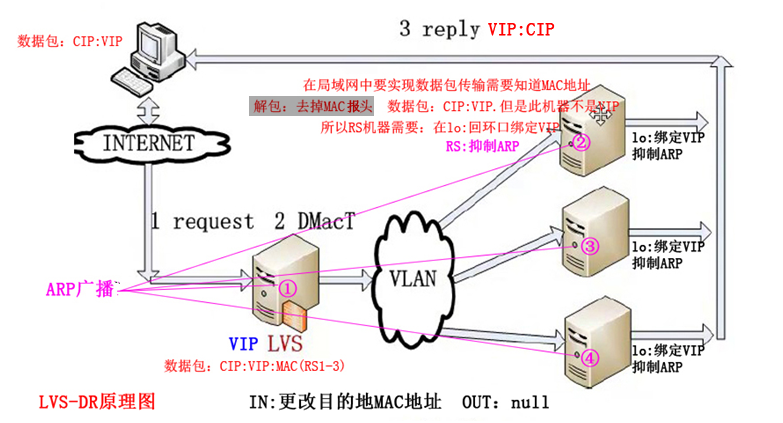
5、由于DR模式的调度器仅进行了目的MAC地址的改写，因此，调度器LB无法改变请求的报文的目的端口（缺点）

6、当前，调度器LB支持几乎所有的UNIX，LINUX系统，但目前不支持WINDOWS系统。真实服务器RS节点可以是WINDOWS系统。

7、总的来说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2000W PV或并发请求1万以下都可以考虑用haproxy/nginx（LVS NAT模式）

8、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

9、需要注意RS节点的VIP绑定（L0：vip，L0：vip1）和ARP抑制问题



|  |  |  |  |
| --- | --- | --- | --- |
|  | VS/NAT | VS/TUN | VS/DR |
| DR Real Server (节点服务器） | config dr gw | Tunneling | Non-arp device/tie vip |
| Server Network (网络） | privates | LAN/WAN | LAN |
| Server Number (节点数量） | low (10~20) | High (100) | High (100) |
| Real Server Gateway | load balancer | Own router（能出网） | Own router（能出网） |
| 优点 | 地址和端口转换 | WAN环境，加密数据 | 性能最高 |
| 缺点 | 效率低 | 隧道支持 | 不能跨出LAN |

**LVS的调度算法**

LVS的调度算法决定了如何在集群节点之间分布工作负荷。

当Director调度器收到来自客户端计算机访问它的VIP上的集群服务的入站请求时，Director调度器必须决定哪个集群节点应该处理请求。Director调度器可用于做出该决定的调度方法分成两个基本类别：  
固定调度方法：rr,wrr,dh,sh  
动态调度算法：wlc,lc,lblc,lblcr,SED,NQ

| **算法** | **说明** |
| --- | --- |
| **rr** | 轮循调度，它将请求依次分配不同的RS节点，也就是在RS节点中均摊请求。这种算法简单，但是只适合于RS节点处理性能相差不大的情况 |
| **wrr** | 权重轮循，它将依据不同RS节点的权值分配任务。权值较高的RS将优先获得任务，并且分配到的连接数将比权值较低的RS节点更多。相同权值的RS得到相同数目的连接数 |
| dh | 目的地址哈希调度，以目的地址为关键字查找一个静态hash表来获得需要的RS |
| sh | 源地址哈希调度，以源地址为关键字查找一个静态hash表来获得需要的RS |
| **wlc** | 加权最小连接数调度，实际连接数除以权值，最小的RS作为分配的RS |
| lc | 最小连接数调度，连接数最小的RS作为分配的RS |
| lblc | 基于地址的最小连接数调度，将来自同一目的地址的请求分配给同一台RS节点 |
| lblcr | 基于地址带重复最小连接数调度。（略） |
| SED | 最短的期望的延迟（不成熟） |
| NQ | 最小队列调度（不成熟） |

**LVS的调度算法的生产环境选型**

**一般的网络服务，如Http，Mail，MySQL等，常用的LVS调度算法为：**

基本轮叫调度rr算法

加权最小连接调度wlc

加权轮叫调度wrr算法

基于局部性的最少链接LBLC和带复制的基于局部性最少链接LBLCR主要适用于Web Cache和Db Cache集群，但是我们很少这样用。（都是一致性哈希算法）

**LVS集群的特点**

实现三种IP负载均衡技术和10种连接调度算法的IPVS软件。在IPVS内部实现上，采用了高效的Hash函数和垃圾回收机制，能正确处理所调度报文相关的ICMP消息（有些商品化的系统反而不能）。虚拟服务的设置数目没有限制，每个虚拟服务都有自己的服务器集。它支持持久的虚拟服务（如HTTP Cookie 和HTTPS等需要该功能的支持），并提供详尽的统计数据，如连接的处理速率和报文的流量等。针对大规模拒绝服务（Deny of service）攻击，实现了三种防卫策略：有基于内容请求分发的应用层交换软件KTCPVS，它也是在Linux内核中实现。有相关的集群管理软件对资源进行检测，能及时将故障屏蔽，实现系统的高可用性。主，从调度器能周期性地进行状态同步，从而实现更高的可用性。

**安装**

lsmod|grep ip\_vs

uname -r

如果 /usr/src/kernels/底下没有目录则执行yum -y install kernel-devel

如果有2个则选择和uname -r 一样的做如下软连接

ln -s /usr/src/kernels/3.10.0-957.5.1.el7.x86\_64/ /usr/src/linux

**特别注意：**  
此ln命令的链接路径要和uname -r输出结果内核版本对应，工作中如果做安装虚拟化可能有多个内核路径  
如果没有/usr/src/kernels/2.6.32-431.el6.x86\_64/路径，很可能是因为缺少kernel-devel软件包。可通过yum进行安装

下载：

wget http://www.linuxvirtualserver.org/software/kernel-2.6/ipvsadm-1.24.tar.gz # <===适合5.x系统

wget http://www.linuxvirtualserver.org/software/kernel-2.6/ipvsadm-1.26.tar.gz # <===适合6.x系统

2.6适用于kernel2.3.28及之后的内核版本，还需要 yum install libnl\* popt\* gcc gcc-c++ -y

安装：

[root@lb01\_lvs /server/tools/ipvsadm-1.26]# make

make -C libipvs

make[1]: Entering directory `/server/tools/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

make[1]: gcc: Command not found

make[1]: \*\*\* [libipvs.o] Error 127

make[1]: Leaving directory `/server/tools/ipvsadm-1.26/libipvs'

make: \*\*\* [libs] Error 2

如上报错解决方案：

yum -y install gcc gcc-c++

yum -y install libnl\* popt\*

[root@lb01\_lvs /server/tools/ipvsadm-1.26]# lsmod|grep ip\_vs

[root@lb01\_lvs /server/tools/ipvsadm-1.26]# ipvsadm 或者modprobe ip\_vs

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

**#代表安装成功，并且加载到内核**

[root@lb01\_lvs /server/tools/ipvsadm-1.26]# lsmod|grep ip\_vs

ip\_vs 141092 0

nf\_conntrack 111302 1 ip\_vs

libcrc32c 12644 2 xfs,ip\_vs

**DR模式**

**#添加VIP**

[root@lb01\_lvs ~]# ifconfig ens33:0 10.0.0.12/24 up

route add -host 10.10.10.12 dev ens33

[root@lb01\_lvs /server/tools/ipvsadm-1.26]# route -n

Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

0.0.0.0 10.10.10.1 0.0.0.0 UG 100 0 0 ens33

10.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 ens33

10.10.10.0 0.0.0.0 255.255.255.0 U 100 0 0 ens33

10.10.10.12 0.0.0.0 255.255.255.255 UH 0 0 0 ens33

[root@lb01\_lvs ~]# ipvsadm -C

[root@lb01\_lvs ~]# ipvsadm --set 30 5 60

[root@lb01\_lvs ~]# ipvsadm -A -t 10.10.10.12:80 -s wrr -p 20

[root@lb01\_lvs ~]# ipvsadm -a -t 10.10.10.12:80 -r 10.10.10.20:80 -g -w 1

[root@lb01\_lvs ~]# ipvsadm -a -t 10.10.10.12:80 -r 10.10.10.21:80 -g -w 1

[root@lb01\_lvs ~]# ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 10.10.10.12:80 rr persistent 20

-> 10.10.10.20:80 Route 1 0 0

-> 10.10.10.21:80 Route 1 0 0

[root@lb01\_lvs ~]#**ipvsadm -d -t 10.10.10.12:80 -r 10.10.10.21:80 #删除某个RS服务器**

[root@lb01\_lvs /server/scripts]# **ipvsadm -lnc**

IPVS connection entries

pro expire state source virtual destination

TCP 00:41 NONE 10.10.10.200:0 10.10.10.12:80 10.10.10.20:80

TCP 00:21 SYN\_RECV 10.10.10.200:62445 10.10.10.12:80 10.10.10.20:80

**手工在RS节点服务器Lo上绑定VIP**

ifconfig lo:12 10.10.10.12 netmast 255.255.255.255 up #RS节点服务器都需要绑定

**ip addr add 10.10.10.12/32 dev lo label lo:12**

route add -host 10.10.10.12 dev lo

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

**arp抑制技术参数说明**

**arp\_ignore-INTRGER**

定义对目标地址为本地IP的ARP询问不同的应答模式

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。

**arp\_announce-INTEGER**

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（默认值）：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。如果该来访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。限制了使用本地的vip地址作为优先的网络接口

**2.3 自动配置脚本**

**LVS 自动化配置脚本**

#!/bin/bash

# Author:Lin yao hong

#LVS scripts

. /etc/init.d/functions

Netdev=ens33

Gw=10.10.10.1

VIP=10.10.10.12

SUBNET="${Netdev}:`echo $VIP | awk -F "." '{print $4}'`"

PORT=80

RIP=(

10.10.10.20

10.10.10.21

)

function start(){

if [ `ifconfig | grep $VIP | wc -l` -ne 0 ];then

stop

fi

#ifconfig $SUBNET $VIP broadcast $VIP netmask 255.255.255.0 up

ifconfig $SUBNET $VIP/24 up

route add -host ${VIP} dev ${Netdev}

ipvsadm -C

ipvsadm --set 30 5 60

ipvsadm -A -t $VIP:$PORT -s wrr -p 20

for ((i=0;i<${#RIP[\*]};i++))

do

ipvsadm -a -t $VIP:$PORT -r ${RIP[$i]} -g -w 1

done

}

function stop(){

arping -c 1 -I ${Netdev} -s ${VIP} ${Gw}

ipvsadm -C

if [ `ifconfig | grep $VIP | wc -l` -ne 0 ];then

ifconfig $SUBNET down

fi

route del -host $VIP dev ${Netdev} &>/dev/null

}

case "$1" in

start)

start

action "ipvs is started" /bin/true

;;

stop)

stop

action "ipvs is stopped" /bin/true

;;

restart)

stop

action "ipvs is stopped" /bin/true

start

action "ipvs is started" /bin/true

;;

\*)

echo "USAGE:$0 {start | stop | restart}"

esac

**RS\_server自动化配置脚本**

#!/bin/bash

# Author:Lin yao hong

# RS\_sever scripts

. /etc/rc.d/init.d/functions

VIP=(

10.10.10.12

10.10.10.13

)

case "$1" in

start)

for ((i=0;i<${#VIP[\*]};i++))

do

echo "start LVS of REALServer IP"

interface="lo:`echo ${VIP[$i]} | awk -F "." '{print $4}'`"

#/sbin/ifconfig $interface $VIP broadcast $VIP netmask 255.255.255.255 up

/sbin/ifconfig $interface ${VIP[$i]} netmask 255.255.255.255 up

#/sbin/ifconfig $interface ${VIP[$i]}/32 up

route add -host ${VIP[$i]} dev $interface

done

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

;;

stop)

for ((i=0;i<${#VIP[\*]};i++))

do

echo "start LVS of REALServer IP"

interface="lo:`echo ${VIP[$i]} | awk -F "." '{print $4}'`"

#/sbin/ifconfig $interface $VIP broadcast $VIP netmask 255.255.255.255 up

route del -host ${VIP[$i]} dev $interface

/sbin/ifconfig $interface ${VIP[$i]} down

done

if [ ${#VIP[$i]} -le 1 ];then

echo "0" > /proc/sys/net/ipv4/conf/lo/arp\_ignore

echo "0" > /proc/sys/net/ipv4/conf/lo/arp\_announce

echo "0" > /proc/sys/net/ipv4/conf/all/arp\_ignore

echo "0" > /proc/sys/net/ipv4/conf/all/arp\_announce

fi

;;

\*)

echo "Usage: $0 {start|stop}"

exit 1

esac

**NAT模式**

LVS添加时使用-m 指定NAT模式

ipvsadm -a -t 10.10.10.12:80 -r 10.10.10.21 **-m** -w 1

**RS节点服务器、把网关设置为VIP即可**

net.ipv4\_ip\_forward = 1

[root@web01 ~]# route del default gw 10.10.10.1

[root@web01 ~]# route add default gw 10.10.10.7

[root@web01 ~]# route -n

Kernel IP routing table

Destination Gateway Genmask Flags Metric Ref Use Iface

0.0.0.0 10.10.10.12 0.0.0.0 UG 0 0 0 ens33

10.10.10.0 0.0.0.0 255.255.255.0 U 100 0 0 ens33

抓包工具

tcpdump -i ens33 tcp port 80 -x -vv -X -s 1500

tcpdump -i ens33 tcp port 80 -s 1500

说明：草塔码没测试成功

**监控相关**

还可以设置对rul监（极少用）：具体可以查看keepalived.conf默认参数

可以通过genhash 来获取MD5值

例:genhash -s 192.168.1.21 –p 8010 -u /test/html