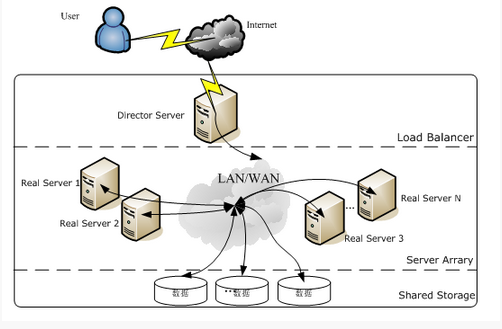
**LVS基本介绍及NAT模型配置示例**

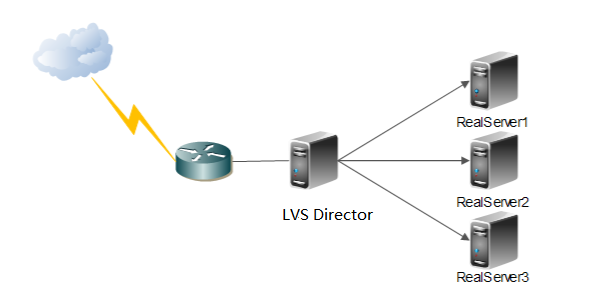
原贴：<http://breezey.blog.51cto.com/2400275/1340489>

**spacer.gif[](http://img1.51cto.com/attachment/201312/012029634.png)**

LVS是Linux Virtual Server的缩写，即linux虚拟服务器。是一款强大的基于四层的负载均衡器。它具有良好的可靠性，可拓展性和可操作性。从而以低廉的成本实现最优的性能。

**1.工作模式：**

1) NAT模型：

[](http://img1.51cto.com/attachment/201312/012111503.png)

特点：

1.director必须与realserver位于同一个网络。

2.realserver通常使用私有地址，用于各节点间的通信。

3.director位于客户端与realserver之间，所有的请求和响应都必须经由director转发。

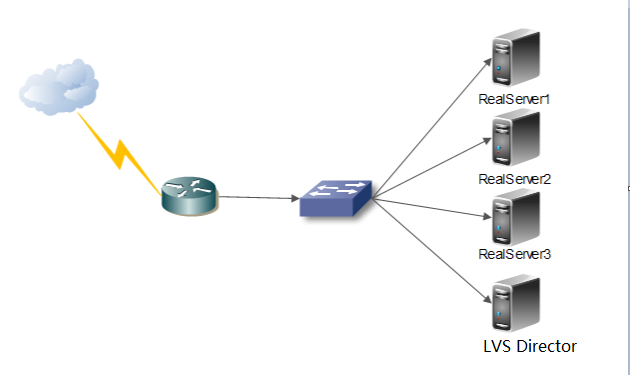
4.director作为realserver的网关。

5.支持端口映射。

6.director必须有两块网卡

7.不适用于较大规模的应用。

2）DR模型：直接路由

spacer.gif[](http://img1.51cto.com/attachment/201312/012143204.png)

DR模型的特点：

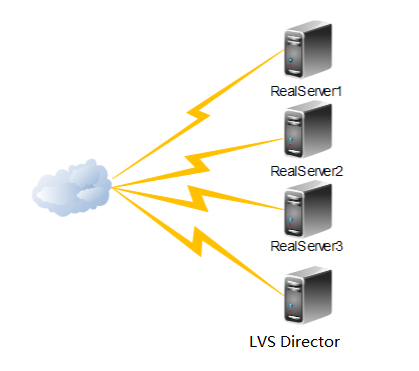
1.director必须与realserver位于同一个网络。

2.realserver可以使用公网ip地址，便于远程管理与配置。

3.director只负责处理入站请求。而响应报文则由realserver直接返回给client端。大大节省director的资源

4.不能做端口映射

3）TUN:IP隧道

spacer.gif[](http://img1.51cto.com/attachment/201312/012200140.png)

特点：

1.集群节点可以跨越Internet。

2.RIP必须是公网地址。

3.Director仅负责处理入站请求，响应报文则由Real Server直接发往客户端。

4.realserver的OS必须支持隧道功能。

5.不支持端口映射。

**2.LVS的IP地址类型：**

1）VIP：虚拟IP地址，并不提供服务，而是将用户的请求转发到后方。

2）RIP：真正IP地址，客户端真正提供服务的IP地址。

3）DIP：调试IP地址，通常是和RIP相连的LVS的IP地址。

4）CIP：客户端IP地址，用户请求时，用户的IP。

**3.调度算法：**

1）rr:轮循。调度器通过“轮循”调度算法将外部请求按顺序轮流分配到集群中的真实服务器上，它均等的对待每一台服务器，而不管服务器上扑鼻的连接数和系统负载。

2）wrr:加权轮循。根据真实服务器的不同处理能力来调度访问请求。这样可以保证处理能力强的服务器能处理更多的动态访问流量。调度器可以自动问询真实服务器的负载情况，并动态的调整其权值。

3）DH：目标hash。当某客户IP第一次访问到集群中的A服务器，在一段时间内，该客户再来访问集群的时候，如果A服务器是可用的且未超载，则依然将客户的请求转发到A服务器上。可用于解决动态网站的session共享问题。

4）SH：源hash。与目标hash相反，用于双线或多线接入的网络。

5）lc:最少链接。根据真实服务器已建立的连接数进行分配，将收到的访问请求优先分配给连接数最少的节点。

6）wlc：加权最少链接。在服务器节点性能差异较大的情况下，可以为真实服务器自动调整其权重，权重较高的节点将承担更大比例的活动连接。

7）sed：最短的期望的延迟

8）nq:最少队列调度。无需队列。如果真实服务器的连接数为零中，就直接分配一个连接，不需要再进行sed运算。

9）lblc:基于局部性的最少链接。针对目标IP地址的负载均衡，目前主要用于cache集群系统。该算法根据请求的目标IP地址找出该目标IP地址最近使用的服务器，比如说A服务器，若A服务器是可用的且没有超载，将请求发送到该服务器；若A服务器不存在，或者该服务器超载且有服务器处于一半的工作负载，则用“最少链接”的原则选出一个可用的服务器，比如说B服务器，将请求发送到该服务器。

10)lblcr:带复制的基于局部性的最少链接。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。与lblc算法的不同之处在于它要维护一个目标IP地址到一组服务器的映射，而lblc维护从一个目标到一台服务器的映射。该算法根据请求的目标IP地址找出该目标IP地址对应的服务器组，按“最小链接”原则从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，刚按“最小链接”原则从这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被修改，将最忙的服务器从服务组中删除，以降低复制的程度。

**4.LVS管理工具**

ipvsadm

rpm -vih ipvsadm-1.25-9.el6.i686.rpm

ipvsadm -v

#查看版本信息

创建虚拟服务器：

-A：添加虚拟服务器

-t:指定VIP地址及tcp端口

-s:指定负载调度算法

示例：

ipvsadm -A -t 172.16.16.172:80 -s rr

添加服务器节点：

-a:添加真实服务器

-t:指定VIP地址及TCP端口

-r:指定RIP地址及TCP端口

-m:使用NAT集群模式（-g DR模式、-i TUN模式）

-w:用来设置权重（权重为0时表示暂停节点）

示例：

ipvsadm -a -t 172.16.16.172:80 -r 192.168.7.21:80 -m -w 1

ipvsadm -a -t 172.16.16.172:80 -r 192.168.7.22:80 -m -w 1

ipvsadm -a -t 172.16.16.172:80 -r 192.168.7.23:80 -m -w 1

查看集群节点状态

ipvsadm -L -n #查看节点状态

ipvsadm -Lnc #查看负载连接情况

删除服务器节点

ipvsadm -d -r 192.168.7.23:80 -t 172.16.16.172:80 #删除某一个节点

ipvsadm -D -t 172.16.16.172:80 #删除整虚拟服务器

保存负载分配策略

ipvsadm-save > /etc/sysconfig/ipvsadm #保存策略

service ipvsadm stop #清除策略

service ipvsadm start #重建规则

**5.配置示例**

地址转换模式（LVS-NAT）

1）实验环境：

lvs负载调度器：eth0:172.16.16.172/24(接外网) eth1:192.168.7.254/24（接节点）

web服务器池：节点1-节点2：192.168.7.21-22/24

2）配置负载调度器

a.配置SNAT转发规则

vim /etc/sysctl.conf

——————————————–

net.ipv4.ip\_forward = 1

——————————————–

sysctl -p

iptables -t nat -A POSTROUTING -s 192.168.7.0/24 -o eth0 -j SNAT –to-source 172.16.16.172 #做SNAT

b.配置负载分配策略

service ipvsadm stop

ipvsadm -A -t 172.16.16.172:80 -s rr

ipvsadm -a -t 172.16.16.172:80 -r 192.168.7.21:80 -m -w 1

ipvsadm -a -t 172.16.16.172:80 -r 192.168.7.22:80 -m -w 1

service ipvsadm save

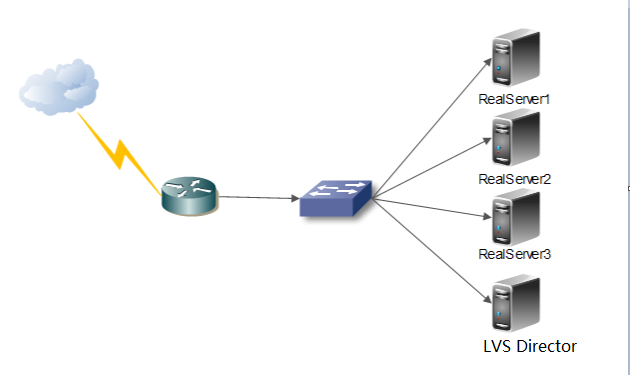
chkconfig ipvsadm on

3）配置节点服务器

安装httpd，创建测试网页,进行测试。

**LVS DR模型详解**

原贴：<http://breezey.blog.51cto.com/2400275/1340490>

spacer.gif[](http://img1.51cto.com/attachment/201312/013954373.png)

DR模型的特点：

1.director必须与realserver位于同一个网络。

2.realserver可以使用公网ip地址，便于远程管理与配置。

3.director只负责处理入站请求。而响应报文则由realserver直接返回给client端。大大节省director的资源

4.不能做端口映射

DR工作原理：

客户端通过路由器将请求交给director的VIP。director接收请求，将通过相应的算法将请求转发给相应的realserver。在转发的过程中，会修改请求包的目的mac地址，目的ip地址不变。realserver接收请求，并直接响应客户端。这时，便出现一个问题：director此时与realserver位于同一个网络中，当director直接将请求转发给realserver时，realserver检测到该请求包的目的ip是vip而并非自己，便会丢弃，而不会响应。。为了解决这个问题，我们需要在所有realserver上都配上VIP。这时会出现第二个问题：director与realserver位于同一个网络中，当出现多个vip时，通过arp广播，整个网络必然混乱，我们无法保证客户端的请求一定会发往director。为保证当客户端请求抵达网络时，只有director来响应请求。我们需要realserver忽略对vip的arp请求，并且在arp通告时，不通告vip地址。我们需要配置如下两个内核参数：

arp\_ignore ：

当一台主机某个接口接收到arp请求时，这台机上的ip地址是否通过该接口响应给对应请求的限制级别。简单来说，当这台主机有多块网卡，或者有多个ip地址时，是否通过该接口将所有ip地址响应给对应的请求。

0:不作任何限制。将所有本机的ip地址都响应给对应请求。

1:当请求的ip就配置在本接口上时，才予以响应。换句话说，就是只响应当前接口的ip地址。

arp\_announce：

当一台主机加入到一个新的网络。会发出ARP通告，向网络中其他主机通告自己的ip地址及mac等信息。arp\_anounce就用来限制通告的级别。当这台主机有多块网卡或者多个ip地址时，是否通过某一个接口将所有ip地址通告给该网络中的主机。

0:不作任何限制。从任意接口通告所有ip地址。

1:尽可能避免通告非本接口上的ip。

2:只通知本接口的ip。非本接口的ip不予通告。

由此我们亦可知，在realserver上配置vip时显然也不能配置的物理接口的虚接口上，而应该配置在loopback上。

**配置示例：**

1）实验环境

lvs负载调度器：eth0：172.16.16.173/24，VIP eth0:0 172.16.16.172

web服务器池：节点1-节点2：172.16.16.177-178/24

2）配置负载调度器

a.配置虚拟IP地址（VIP）

ifconfig eth0:0 172.16.16.172 broadcast 172.16.16.172 netmask 255.255.255.0 up #在eth0上配置并启动VIP

route add -host 172.16.16.172 dev eth0:0 #特殊路由，当客户端请求的ip为vip时，则由eth0:0响应。

b.调整/proc响应参数

echo 1 > /proc/sys/net/ipv4/ip\_forward #开启路由转发

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

send\_redirects：icmp发送重定向。经查阅相关资料，开启此选项有一定的危险性，容易被人伪造icmp重定向包发起攻击。若当前director并非充当路由或网关，建议关闭。默认为开启，0为关闭。

c.配置负载分配策略

service ipvsadm stop

ipvsadm -A -t 172.16.16.172:80 -s wlc

ipvsadm -a -t 172.16.16.172:80 -r 172.16.16.177 -g -w 1

ipvsadm -a -t 172.16.16.172:80 -r 172.16.16.178 -g -w 1

service ipvsadm save

chkconfig ipvsadm on

3）配置节点服务器

a.配置虚拟IP地址

ifconfig lo:0 172.16.16.172 broadcast 172.16.16.172 netmask 255.255.255.255 up

/sbin/route add -host 172.16.16.172 dev lo:0

b.调整/proc响应参数

echo 1 > /proc/sys/net/ipv4/conf/all/arp\_ignore

echo 2 > /proc/sys/net/ipv4/conf/all/arp\_announce

echo 1 > /proc/sys/net/ipv4/conf/default/arp\_ignore

echo 2 > /proc/sys/net/ipv4/conf/default/arp\_announce

echo 1 > /proc/sys/net/ipv4/conf/lo/arp\_ignore

echo 2 > /proc/sys/net/ipv4/conf/lo/arp\_announce

此几项配置即是用于限制arp请求及arp通告

c.安装httpd，进行测试。

可将整个lvs dr模型下的director和realserver上的配置写成脚本，如下：

director启动脚本：

|  |
| --- |
| #/bin/bash  #  #LVS for VS/DR  . /etc/rc.d/init.d/functions  #  VIP=172.16.16.172  RIP1=172.16.16.177  RIP2=172.16.16.178  PORT=80  #  case $1 in  start)  #clear all iptables policy  /sbin/iptables -F  /sbin/iptables -X  /sbin/iptables -Z  #configure VIP  /sbin/ifconfig eth0:1 $VIP broadcast $VIP netmask=255.255.255.0  /sbin/route add -host $VIP dev eth0:1  #enalbe ip\_forward  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  #clear all ipvsadm rules/services.  /sbin/ipvsadm -C  #Add an IP virtual service for VIP 172.16.16.172port 80  /sbin/ipvsadm -A -t $VIP:$PORT -s wlc  /sbin/ipvsadm -a -t $VIP:$PORT -r $RIP1 -g -w 1  /sbin/ipvsadm -a -t $VIP:$PORT -r $RIP2 -g -w 1  /bin/touch /var/lock/subsys/ipvsadm > /dev/null  ;;  stop)  #Reset ipvsadm  /sbin/ipvsadm -C  /sbin/ifconfig eth0:1 down  /bin/rm -f /var/lock/subsys/ipvsadm > /dev/null  ;;  status)  if [ ! -e /var/lock/subsys/ipvsadm ];then  echo "ipvs is stopped..."  else  echo "ipvs is running..."  /sbin/ipvsadm -L -n  fi  ;;  \*)  echo "Usage: $0 {start|stop|status}"  exit 1  ;;  esac |

realserver启动脚本：

|  |
| --- |
| case $1 in  start)  #start LVS-DR on this real server machine.  /sbin/ifconfig lo down  /sbin/ifconfig lo up  echo 1 > /proc/sys/net/ipv4/conf/lo/arp\_ignore  echo 1 > /proc/sys/net/ipv4/conf/all/arp\_ignore  echo 2 > /proc/sys/net/ipv4/conf/lo/arp\_announce  echo 2 > /proc/sys/net/ipv4/conf/all/arp\_announce  /sbin/ifconfig lo:0 $VIP broadcast $VIP netmask 255.255.255.0 up  /sbin/route add -host $VIP dev lo:0  ;;  stop)  #stop LVS-DR on this real server machine.  /sbin/ifconfig lo:0 down  echo 0 > /proc/sys/net/ipv4/conf/lo/arp\_ignore  echo 0 > /proc/sys/net/ipv4/conf/all/arp\_ignore  echo 0 > /proc/sys/net/ipv4/conf/lo/arp\_announce  echo 0 > /proc/sys/net/ipv4/conf/all/arp\_announce  ;;  status)  #Sstatus of LVS-DR real server.  islothere=`/sbin/ifconfig lo:0 | grep $VIP`  isrothere=`netstat -rn | grep "lo:0" | grep $VIP`  if [ !"islothere" -o !"isrothere" ]; then  echo "LVS-DR real server Stopped."  else  echo "LVS-DR real server Running"  fi  ;;  \*)  # Invalid entry.  echo "Usage: $0 (start|stop|status)"  exit 1  ;;  esac |

以上为第一种配置方法，也是官方的常规配置方法。

在第一种配置方法中，我们提到在所有realserver的loopback上配置vip来实现接收director端的请求转发，然后通过修改内核参数来忽略arp应答与通告。那我作一个假设。假设我不在realserver上配置vip，我依然希望realserver能够接收director转发来的请求。前文我们提到，如果realserver不配置vip，默认会将director转发过来的请求包丢弃。那么除了在loopback端配置vip，还有没有其他的办法让realserver接收来自director转发的请求包？

这个时候我们需要用到iptables的重定向策略。我们不需要在realserver的loopback上配置vip，也不需要去修改内核参数来忽略arp的响应与通告。director上的配置同上并不发生改变，realserver的配置如下：

iptables -t nat -A PREROUTING -d 172.16.16.172 -p tcp --dport 80 -j REDIRECT

整个配置过程完成，是不是比上一种方法更为简单？

**lvs+keepalived**

原贴：<http://breezey.blog.51cto.com/2400275/1340653>

**1.keepalived安装**

yum install -y kernel-devel openssl-devel popt-devel

rpm -vih ipvsadm-1.25\*

wget http://www.keepalived.org/software/keepalived-1.2.2.tar.gz

tar zxf keepalived-1.2.2.tar.gz

cd keepalived-1.2.2

./configure --prefix=/ --with-kernel-dir=/usr/src/kernels/2.6.32-131.0.15.el6.i686

make && make install

chkconfig --add keepalived

chkconfig keepalived on

**2.简单双机热备**

环境

主服务器172.16.16.173/24

备服务器172.16.16.174/24

漂移IP172.16.16.172/24

主服务器配置

cd /etc/keepalived

cp keepalived.conf keepalived.conf.bak

vim keepalived.conf

global\_defs {

route\_id HA\_TEST\_R1 #本服务器的名称

}

vrrp\_instance VI\_1 { #定义vrrp热备实例

state MASTER #热备状态MASTER表示主服务器

interface eth0 #承载VIP地址的物理接口

virtual\_route\_id 1 #虚拟服务器的ID每个热备组保持一致

priority 100 #优先级数据值越大优先级越高

advert\_int 1 #通告间隔秒数心跳频率

authentication { #认证信息每个热备组保持一致

auth\_type PASS #认证类型

auth\_pass 123456 #密码字符

}

virtual\_ipaddress { #指定漂移地址可以有多个

172.16.16.172

}

}

service keepalived start

ip addr show eth0

备用服务器配置

vim keepalived.conf

global\_defs {

route\_id HA\_TEST\_R2 #本服务器的名称

}

vrrp\_instance VI\_1 {

state SLAVE #热备状态SLAVE表示备用服务器

priority 99 #优先级数值应低于主服务器

……

}

**3.lvs+keepalived**

环境

主调度器172.16.16.173/24 192.168.7.173/24

备调度器172.16.16.174/24 192.168.7.174/24

web服务器池172.16.16.177-180/24 192.168.7.177-180/24

nfs共享192.168.7.250/24

VIP:172.16.16.172

配置主调度器

vim keepalived.conf

global\_defs {

route\_id LVS\_HA\_R1

}

vrrp\_instance VI\_1 {

state MASTER

interface eth0

virtual\_route\_id 1

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 123456

}

virtual\_ipaddress {

172.16.16.172

}

}

#web服务器池配置

virtual\_server 172.16.16.172 80 { #VIP地址端口

delay\_loop 15 #健康检查的时间间隔秒

lb\_algo rr #轮询调度算法

lb\_kind DR #直接路由集群模式

!persistence 60 #连接保持时间若起用则去掉叹号

protocol TCP #应用服务采用TCP协议

real\_server 172.16.16.177 80 { #第一个web节点的IP 端口

weight 1 #节点的权重

TCP\_CHECK { #健康检查方式

connect\_port 80 #检查的目标端口

connect\_timeout 3 #连接超时秒

nb\_get\_retry 3 #重试次数

delay\_before\_retry 4 #重试间隔秒

}

}

real\_server 172.16.16.178 80 { #第二个web节点的IP 端口

……

}

real\_server 172.16.16.179 80 { #第三个……

……

}

……

配置从调度器类似于主

global\_defs {

route\_id LVS\_HA\_R2

}

vrrp\_instance VI\_1 {

state SLAVE

interface eth0

virtual\_route\_id 1

priority 99

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 123456

}

virtual\_ipaddress {

172.16.16.172

}

}

virtual\_server 172.16.16.172 80 {

delay\_loop 15

lb\_algo rr

lb\_kind DR

!persistence 60

protocol TCP

real\_server 172.16.16.177 80 {

weight 1

TCP\_CHECK {

connect\_port 80

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 4

}

}

real\_server 172.16.16.178 80 {

……

}

real\_server 172.16.16.179 80 {

……

}

……

}

附加案例

**集群相关基础概念**

原贴：<http://chris2013.blog.51cto.com/6931081/1177255>

集群的概念：

集群（Cluster）：组合多台计算机，来完成同一个目的。

集群的分类：

LB：Load Balance，负载均衡

HA：High Available，高可用（一年内的平均无故障时间：99.999%（用于评定HA的性能）即一年中允许出故障的时间：5分钟。

HP：High Performance，高性能

实现LB常见的软硬件：

硬件：

F5：BingIP（1000w/s）

IBM：A10（600w/s）

Citrix：NetScaler（500w/s）

软件：（开源解决方案）

LVS：Linux Virtual Server（四层交换）

Haproxy（四层交换）

实现Web负载均衡的解决方案，还有常用组合：Nginx+varnish，即反向代理的功能。

Nginx（七层交换，重点在反向代理的性能上）

varnish（既能实现反向代理，还能实现缓存）

实现HA常见的软硬件：

软件：（开源解决方案）

Heartbeat

Keepalived

Corosync+openais(pacemaker) RHCS

Ultramokey

实现HP常见的软硬件：

bowerful

**1、什么是集群？**

集群简单来说就是将多个服务器组合起来来完成一种特定的服务。

搜的定义：计算机集群简称集群是一种计算机系统， 它通过一组松散集成的计算机软件和/或硬件连接起来高度紧密地协作完成计算工作。在某种意义上，他们可以被看作是一台计算机。集群系统中的单个计算机通常称为节点，通常通过局域网连接，但也有其它的可能连接方式。集群计算机通常用来改进单个计算机的计算速度和/或可靠性。

**2、集群的分类：**

LB：Load balancing clusters 负载均衡集群

HA：High-availability (HA) clusters 高可用性集群

HP：High-performance (HPC) clusters 高性能计算集群

**负载均衡集群**：负载均衡集群运行时，一般通过一个或者多个前端负载均衡器，将工作负载分发到后端的一组服务器上，从而达到整个系统的高性能和高可用性。这样的计算机集群有时也被称为服务器群（Server Farm）。 一般高可用性集群和负载均衡集群会使用类似的技术，或同时具有高可用性与负载均衡的特点。Linux虚拟服务器（LVS）项目在Linux操作系统上提供了最常用的负载均衡软件。

**高可用性集群**：一般是指当集群中有某个节点失效的情况下，其上的任务会自动转移到其他正常的节点上。还指可以将集群中的某节点进行离线维护再上线，该过程并不影响整个集群的运行。

**高性能计算集群**：高性能计算集群采用将计算任务分配到集群的不同计算节点而提高计算能力，因而主要应用在科学计算领域。比较流行的HPC采用Linux操作系统和其它一些免费软件来完成并行运算。这一集群配置通常被称为Beowulf集群。这类集群通常运行特定的程序以发挥HPC cluster的并行能力。这类程序一般应用特定的运行库, 比如专为科学计算设计的MPI库。HPC集群特别适合于在计算中各计算节点之间发生大量数据通讯的计算作业，比如一个节点的中间结果或影响到其它节点计算结果的情况。

LVS——概念、架构、模型

原贴：<http://chris2013.blog.51cto.com/6931081/1177501>

LVS（Linux Virtual Server）是由章文嵩开发的一款自由软件。通过LVS我们可以实现：利用Linux操作系统和LVS集群软件实现一个高可用、高性能、低成本的服务器应用集群。

利用LVS架设的服务器集群系统由3部分组成：

最前端Director：提供负载均衡

中间层realserver：提供服务

最低层Shared Storage：提供数据共享存储

下面对这3层进行详细介绍：

Director：

由一台或多台负载调度器组成，本身并不提供任何服务，而是将用户请求按照调度算法转发给后端的realserver。

Linux2.6以上（包含2.6）内核完全内置了LVS的各个模块，不用任何设置就可以支持LVS功能。

Realserver：

由一组实际运行应用服务的服务器组成，可以是一个或多个服务（Director可以兼任realserver的角色）。

Realserver支持所有的操作系统。

Shared Storage：

为所有realserver提供共享存储空间（eg.存储设备）和内容一致性（eg.GFS）的存储区域。

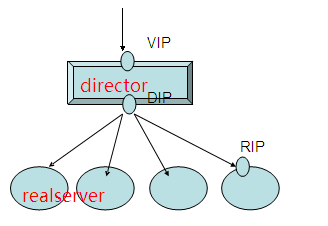
LVS架设服务器集群所涉及的IP地址详解：

CIP：client IP

VIP：virtual IP（与client交互的ip）

DIP：director IP（与realserver通信的IP）

RIP：realserver IP

[](http://img1.51cto.com/attachment/201304/135924440.png)

LVS实现负载均衡的3种模型：

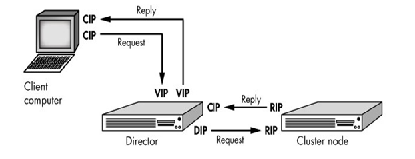
NAT：Network address translation，网络地址转换

DR：Direct routing，直接路由

TUN：IP tunneling，IP隧道

下面对这3种模型进行详细介绍：

NAT工作过程：

[](http://img1.51cto.com/attachment/201304/135948161.png)

（Request）当用户请求到达Director时，Director将请求报文的目标地址（即VIP）改写成选定的Realserver的IP地址（即RIP），同时将报文的目标端口改写成选定的Realserver的相应端口，最后将报文请求送至选定的Realserver。

（Reply）在服务器端得到数据后，Realserver将数据返回给用户时，需要再次经过Director将报文的源地址（即RIP）和源端口改成VIP和相应的端口，然后将数据发送给用户，完成整个负载调度过程。

NAT工作特性：

所有的集群节点都必须在同一个子网中。

RIP是私有地址，仅用于集群节点之间进行通信。

Director同时处理入站和出站请求。

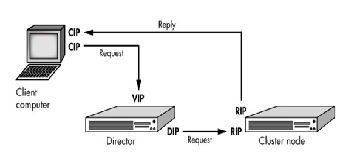
Realserver的网关要指向DIP。

可以实现端口映射。

Realserver可以是任意操作系统。

Director很容易成为系统瓶颈。

DR工作过程：

[](http://img1.51cto.com/attachment/201304/140005544.png)

DR模型也称为旁路模型，即DR通过改写请求报文的MAC地址，将请求发送至Realserver，而Realserver直接将数据响应给client，但是需要在Realserver上需要设置VIP。

DR工作特性：

集群节点和Director必须在同一个物理网络中（on the same network segment）。

RIP可以使用公网IP。好处：Director挂了，可以通过IP直接访问服务。

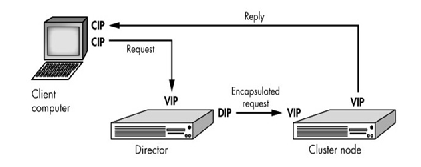
Director只处理入站请求，所以Realserver的网关不能指向DIP。

不支持端口映射：Director和Realserver必须使用同一个端口号。

Realserver可以使用的系统：必须能够支持Realserver隔离ARP广播和一块网卡上支持配置多个IP。

DR模型中的Director的服务压力远低于NAT模型中的。

TUN工作特性：

[](http://img1.51cto.com/attachment/201304/140031228.png)

TUN模型与DR模型的最大区别：

Director和Realserver可以不在同一个物理网络中，甚至可以跨越互联网。

TUN模型的隧道实现原理是在原有IP包的基础上再添加一个：源IP和目标IP。

Lvs + keepalived实现高可用负载均衡集群

原贴：<http://yangdonglin.blog.51cto.com/5404572/1309435>

**Lvs + keepalived实现高可用负载均衡集群**

环境：

|  |  |  |
| --- | --- | --- |
| Lvs--master centos6.4 | eth0:192.168.1.1 | VIP：192.168.1.100 |
| Lvs--backup centos6.4 | eth0:192.168.1.2 |
| Web1 centos5.5 | eth0:192.168.1.3 |
| Web2 centos5.5 | eth0:192.168.1.4 |
| xp | eth0:192.168.1.5 |  |

在lvs--master上面执行以下操作

|  |
| --- |
| yum install -y gcc kernel-devel openssl-devel popt-devel  安装ipvsadm工具  rpm -ivh ipvsadm-1.25-10.el6.i686.rpm  安装keepalive  tar -zxf keepalived-1.2.8.tar.gz  ./configure --sysconfdir=/etc/ --with-kerne-dir=/usr/src/kernels/2.6.32-358.el6.i686/  注意：这个参数并不是要把Keepalived编译进内核，而是指定使用内核源码里面的头文件，就是include目录，不然系统可能无法调用ipvs的模块  make && make install  ln -s /usr/local/sbin/keepalived /sbin/  chkconfig --add keepalived  chkconfig keepalived on |

|  |
| --- |
| ##############################keepalived配置文件如下###########################  ! Configuration File for keepalived  global\_defs {  notification\_email {  shalou@gmail.com #设置报警邮件地址，可以设置多个，每行一个，注意，如果要开启邮件报警，需要开启本机的sendmail服务  }  notification\_email\_from keepalived@localhost #设置邮件的发送地址  smtp\_server 127.0.0.1 #smtp server地址  smtp\_connect\_timeout 30 #设置连接smtp服务器超时时间  router\_id LVS\_DEVEL #运行keepalived机器的标识  }  vrrp\_instance VI\_1 {  state MASTER #指定keepalived的角色，MASTER表示此主机是主服务器，BACKUP表示备用服务器 注意：备用节点上要改成BACKUP  interface eth0 #指定监测的网络的接口  virtual\_router\_id 51 #虚拟路由标识，这个标识是一个数字，两个节点必须一样，以指明各个节点属于同一vrrp组  advert\_int 1 #设定MASTER与BACKUP负载均衡器之间同步检查的时间间隔，单位是秒  authentication { #设置验证类型和密码  auth\_type PASS  auth\_pass 1111  }  virtual\_ipaddress {  192.168.1.100 #设置虚拟ip地址，可以设置多个虚拟ip地址，每行一个  }  }  virtual\_server 192.168.1.100 80 { #设置虚拟服务器，需要指定虚拟ip地址和服务端口，ip与端口之间用空格隔开  delay\_loop 6 #设置健康检查时间，单位是秒  lb\_algo rr #设置lvs的调度算法，这里设置的是rr算法，即轮循算法  lb\_kind DR #设置LVS实现负载均衡的机制，可以有NAT、TUN、DR三个模式可选  nat\_mask 255.255.255.0  persistence\_timeout 50 #会话保持时间，单位是秒  protocol TCP #指定转发协议类型,有tcp和udp两种  real\_server 192.168.1.3 80 { #设置服务节点1，需要指定real server的真实ip和端口，ip和端口之间用空格隔开  weight 3 #权重值，数字越大，权重越高  TCP\_CHECK { #tcp健康检查  connect\_timeout 3 #3秒无响应超时  nb\_get\_retry 3 #重试次数  delay\_before\_retry 3 #重试间隔  connect\_port 80 #健康检查端口  }  {  real\_server 192.168.1.4 80 { #设置服务节点2，需要指定real server的真实ip和端口，ip和端口之间用空格隔开  weight 3 #权重值，数字越大，权重越高  TCP\_CHECK { #tcp健康检查  connect\_timeout 3 #3秒无响应超时  nb\_get\_retry 3 #重试次数  delay\_before\_retry 3 #重试间隔  connect\_port 80 #健康检查端口  }  }  } |

注意：将以上操作在lvs--backup执行一遍

注意：keepalived配置在backup上面要修改两处1：state MASTER要改成state BACKUP 2：修改优先级

然后在后端的web1和web2上面做以下操作（需要有web服务，最好将这两个web的index.html测试文件写成不同内容，方便测试验证，安装过程略）

|  |
| --- |
| #########################################################################################  **分别在realserver上执行realserver.sh这个脚本，脚本内容如下**  #!/bin/bash  SNS\_VIP=192.168.1.100  ./etc/rc.d/init.d/functions  case "$1" in  start)  ifconfig lo:0 $SNS\_VIP netmask 255.255.255.255 broadcast $SNS\_VIP  /sbin/route add -host $SNS\_VIP 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/suys/net/ipv4/conf/all/arp\_announce  sysctl -p > /dev/null 2>&1  echo "RealServer Start OK"  ;;  stop)  ifconfig lo:0 down  route del $SNS\_VIP > /dev/null 2 >&1  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  ;;  \*)  echo "Usage: $0 {start |stop}"  exit 1  esac  exit 0  ###################################################################################  **然后做以下操作**  mkdir /sh  cp realserver.sh /sh  chmod +x /sh/realserver.sh  加入开机启动  echo “bash /sh/realserver.sh start “ >> /etc/rc.loacl  启动脚本： bash /sh/realserver.sh start |

#####################################################################################

然后分别在lvs--master和lvs-backup启动keepalived

service keepalived start

chkconfig keepalived on

然后用ip addr 命令查看vip是否已经存在

测试：1：通过xp客户看访问VIP地址，看看是否能访问正常，有没有轮循的效果

2：测试keepalived，将lvs--master关机或者断网，然后去lvs---backup上面是不是有了VIP地址，如果有，则说明成功，否则错误

报错信息

1：configure: error: Popt libraries is required

解决办法：yum install popt-devel

**Linux集群服务--网络地址转换**

原贴：<http://qiulove.blog.51cto.com/516754/142565>

概述：

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（networkAddressTranslation）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(Virtual Server via Network Address Translation),大多数商品化的

IP负载均衡调度器产品都是使用此方法。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 Direct Routing),它们可以极大地提高系统的伸缩性。

IPVS软件实现了这三种IP负载均衡技术，它们的大致原理如下：

（1）Virtual Server via Network Address translation (VS/NAT)

通过网络地址转换，调度器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通

过调度器时，报文源地址被重写，再返回给客户，完成整个负载调度过程。

（2）Virtual Server via IP Tunneling (VS/TUN)

采用NAT技术时，由于请求和响应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

，调度器把请求报文通过IP隧道转发至真实服务器，而真实服务器将响应直接返回给客户，所以调度器只处理请求报文。由于一般网络服务应

答比请求报文大许多，采用VS/TUN技术后，集群系统的最大吞吐；量可以提高10倍。

（3）Virtual Server via Direct Routing(VS/DR)

VS/DR通过改写请求报文的MAC地址，将请求发送到真实服务器，而真实服务器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。这种方法没有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。

负载调度算法基本有四种：

（1）轮叫（Round Robin）

调度器通过“轮叫”调度算法将外部请求按顺序轮流分配到集群中的真实服务器上，它均等地对待每一台服务器，而不管服务器上实际的连接

数和系统负载。

（2）加权轮叫（Weighted Round Robin）

调度器通过“加权轮叫”调度算法根据真实服务器的不同处理能力来调度访问请求。这样可以保证处理能力强的服务器处理更多的访问流量。

调度器可以自动询问真实服务器的负载情况，并动态地调整其权值。

（3）最少链接(Least Connections)

调度器通过“最少连接”调度算法动态地将网络请求调度到已建立的链接数最少的服务器上。如果集群系统的真实服务器具有相近的系统性能

，采用“最小链接”调度算法可以较好地均衡负载。

（4）加权最少链接(Weighted Least Connections)

在集群系统中的服务器性能差异较大的情况下，调度器采用“加权最少连接”调度算法优化负载均衡性能，具有较高权值的服务器将承受较大

比例的活动连接负载。调度器可以自动询问真实服务器的负载情况，并动态地调整其权值。

实验环境：Vmware 虚拟机

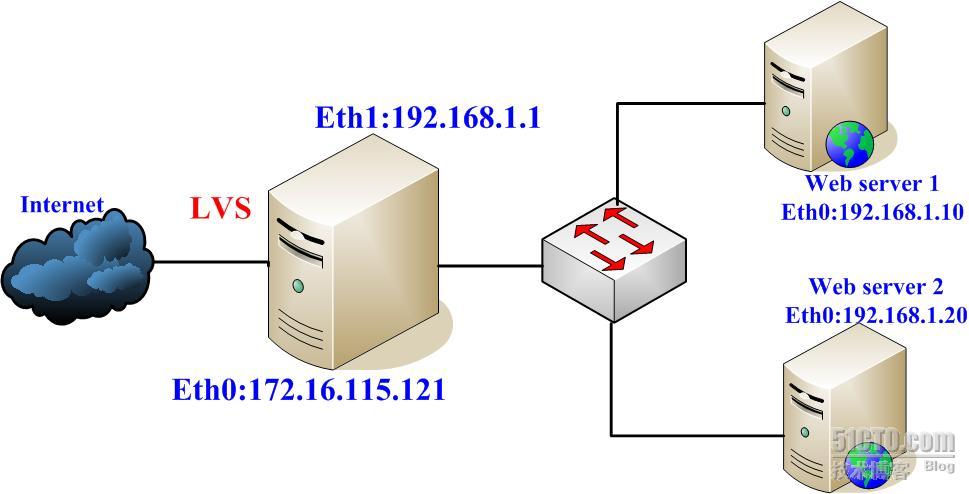
操作系统：Centos 5.2

实验目的: VS/NAT之轮叫算法(RR)

在本实验中，我使用了局域网中的两台电脑，两台电脑用集线器相连，本机安装了一台Centos 5.2操作系统，另一台安装了两台Centos 5.2操

作系统，为便于操作，在本机上用Putty登陆三台机子，每台机子之间可以互访。

如图：



本机作为LVS服务器，要求安装两块网卡。外网eth0:172.16.115.121 内网eth1:192.168.1.1

另两台IP：192.168.1.10和192.168.1.20

当外网访问LVS服务器时，将通过网卡eth1映射到server1与server2，从而交替显示server1与server2的内容，达到轮叫机制的作用。

首先，主机要安装ipvsadm软件包，并且ipvsadm要求内核在2.4.24以后的版本。

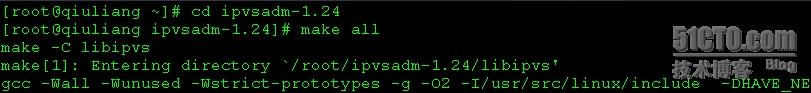
安装前先要对内核进行软连接，因为ipvsadm在安装的过程中先要去查找/usr/src/linux 这个信息，所以我们要先把内核信息做软连接到

/usr/src/linux。如果不这样做，在安装的过程中会报错。

请输入内容http://img1.51cto.com/attachment/200903/200903281238218510125.jpg

然后我们进行安装:





http://img1.51cto.com/attachment/200903/200903261238037649421.bmp

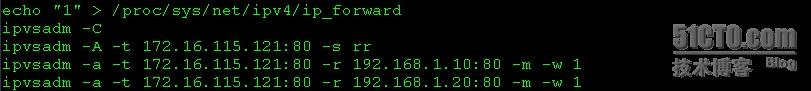
查看是否已经安装成功：

http://img1.51cto.com/attachment/200903/200903261238037698703.bmp

下面简单的做一些NAT包的转换和配置，由于逐条输入命令较慢，所以我们建立一个脚本一起执行：

http://img1.51cto.com/attachment/200903/200903261238037770843.bmp

在脚本中输入以下内容：



--打开包的转发

--清除ipvsadm一系列规则

--两个服务器的报文转发，rr代表轮叫算法，使用80端口

--配置两个web服务器，从172.16.115.121转发到192.168.1.10和192.168.1.20

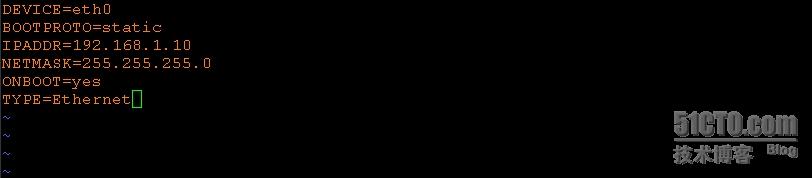
然后我们给脚本附上相应的权限并执行脚本：

http://img1.51cto.com/attachment/200903/200903261238037897015.bmp

接下来我们配置两个web服务器

修改eth0的IP地址，使其和主机的eth1在同一个网段:

http://img1.51cto.com/attachment/200903/200903261238037954375.bmp



保存退出

然后配置网关:

http://img1.51cto.com/attachment/200903/200903261238038045437.bmp

http://img1.51cto.com/attachment/200903/200903261238038061031.bmp

重启网卡：



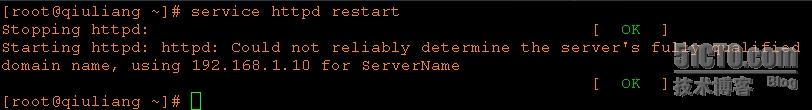
然后我们建一个web页面:

http://img1.51cto.com/attachment/200903/200903261238038160640.bmp

写入如下内容:

http://img1.51cto.com/attachment/200903/200903261238038189359.bmp

重启apache服务:



同样，我们在web2上配置其IP为：192.168.1.20 网关：192.168.1.1

为了便于区别，我们在web2上的web页面设成和web1不一样的内容

http://img1.51cto.com/attachment/200903/200903261238038266343.bmp

http://img1.51cto.com/attachment/200903/200903261238038289140.bmp

同样重启网卡和apache服务:

service network restart

service httpd restart

然后我们打开浏览器，输入172.16.115.121

出现访问页面：

http://img1.51cto.com/attachment/200903/200903261238038347265.bmp

然后刷新，有出现不同的访问页面：

http://img1.51cto.com/attachment/200903/200903261238038379218.bmp

到此，整个实验已基本完成，在实际应用中两个web服务器可以放置相同的web页面，它们可以通过nfs文件共享进行连接，当外界访问LVS服务

器的时候就会出现相同的web页面，从而达到了Linux负载均衡的目的。