< 2018年12月 >

日	一	二	三	四	五	六
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

公告

昵称：MacoLee
园龄：2年7个月
粉丝：36
关注：8
+加关注

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类

Git(1)
python(23)
python之路(1)
Web前端(5)
算法(1)
运维(38)

LVS安装使用详解

简介

LVS是Linux Virtual Server的简称，也就是Linux虚拟服务器，是一个由章文嵩博士发起的自由软件项目，它的官方站点是www.linuxvirtualserver.org。

现在LVS已经是Linux标准内核的一部分，在Linux2.4内核以前，使用LVS时必须要重新编译内核以支持LVS功能模块，但是从Linux2.4内核以后，已经完全内置了LVS的各个功能模块，无需给内核打任何补丁，可以直接使用LVS提供的各种功能。

LVS主要用于服务器集群的负载均衡。其优点有：

```
#工作在网络层，可以实现高性能，高可用的服务器集群技术。

#廉价，可把许多低性能的服务器组合在一起形成一个超级服务器。

#易用，配置非常简单，且有多种负载均衡的方法。

#稳定可靠，即使在集群的服务器中某台服务器无法正常工作，也不影响整体效果。

#可扩展性也非常好。
```

安装配置

linux内核2.4版本以上的基本都支持LVS，要使用lvs，只需要再安装一个lvs的管理工具：ipvsadm

```
yum install ipvsadm
```

ipvsadm用法

其实LVS的本身跟iptables很相似,而且连命令的使用格式都很相似,其实LVS是根据iptables的框架开发的,那么LVS的本身分成了两个部分：

第一部分是工作在内核空间的一个IPVS的模块,其实LVS的功能都是IPVS模块实现的,

第二部分是工作在用户空间的一个用来定义集群服务的一个工具ipvsadm, 这个工具的主要作用是将管理员定义的集群服务列表传送给工作在内核空间中的IPVS模块,下面来简单的介绍下ipvsadm命令的用法

ipvsadm组件定义规则的格式：

```
#virtual-service-address:是指虚拟服务器的ip 地址
#real-service-address:是指真实服务器的ip 地址
#scheduler: 调度方法

#ipvsadm 的用法和格式如下:
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。
-D --delete-service #删除内核虚拟服务器表中的一条虚拟服务器记录。
```

(20557)
4. LVS安装使用详解(18312)
5. linux下查看进程运行的时间(18272)
6. linux系统中rsync+inotify实现服务器之间文件实时同步(10436)
7. Pychram如何导入Django项目(8906)
8. Django数据库设计中字段为空的方式(8326)
9. Saltstack系列3：Saltstack常用模块及API(7641)
10. Keepalived安装使用详解(5999)

评论排行榜
1. Pychram如何导入Django项目(2)
2. HAProxy安装配置详解(1)
3. Python之Fabric模块(1)
4. Ansible之playbook(1)

推荐排行榜
1. HAProxy安装配置详解(5)
2. LVS安装使用详解(4)
3. Nmap命令的29个实用范例(3)
4. linux下查看进程运行的时间(2)
5. linux下进程、端口号相互查看方法(2)
6. Python之Rpyc模块(1)
7. Saltstack系列3：Saltstack常用模块及API(1)
8. CentOS下puppet安装(1)

```
-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，输出为-R 选项可读的格式
-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
-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。
-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。
-M --netmask #子网掩码
-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）
-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
--rate #显示速率信息
--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
```

ipvsadm命令方法

LVS的10种调度算法

lvs调度算法（不区分大小写）可以分为两大类：

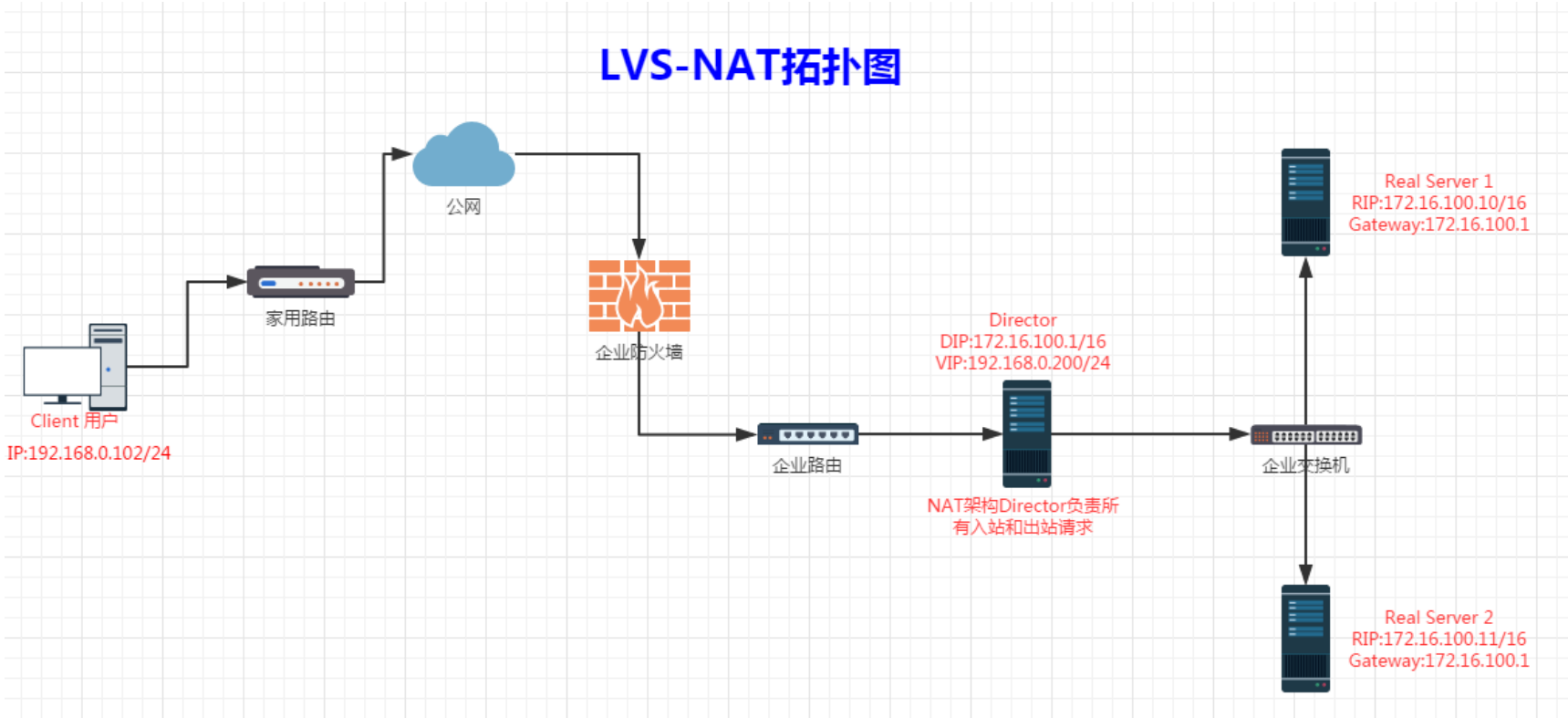
```
1.Fixed Scheduling Method 静态调服方法

RR  #轮询
#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。

WRR  #加权轮询
#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 可以自动问询真实服务器的负载情况，并动态地调整其权值。

DH  #目标地址hash
#算法也是针对目标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。

SH  #源地址hash
#算法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。



2.Dynamic Scheduling Method 动态调服方法

LC  #最少连接
#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"最小连接"调度算法可以较好地均衡负载。

WLC #加权最少连接
#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"加权最少链接"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。调度器可以自动问询真实服务器的负载情况，并动态地调整其权值。

SED #最少期望延迟
```

```
#基于wlc算法，举例说明：ABC三台机器分别权重123，连接数也分别是123，name如果使用WLC算法的话一个新请求 进入时他可能会分给ABC中任意一个，使用SED算法后会进行这样一个运算
#A:(1+1)/2
#B:(1+2)/2
#C:(1+3)/3
#根据运算结果，把连接交给C

NQ   #从不排队调度方法
#无需列队，如果有台realserver的连接数=0 就直接分配过去，不需要进行sed运算.

LBLC    #基于本地的最少连接
# "基于局部性的最少链接" 调度算法是针对目标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，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。

LBLCR    #带复制的基于本地的最少连接
#"带复制的基于局部性最少链接"调度算法也是针对目标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。
#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"最小连接"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，则按"最小连接"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。

lvs调度算法
```

LVS三种工作模式：NAT（地址转换）、DR（直接路由）、TUN（隧道）

LVS-NAT：地址转换

架构图：



工作方式：

NAT模型其实就是通过网络地址转换来实现负载均衡的。下面是它的流程：

1.用户请求VIP(也可以说是CIP请求VIP)

2,Director Server 收到用户的请求后,发现源地址为CIP请求的目标地址为VIP,那么Director Server会认为用户请求的是一个集群服务,那么Director Server 会根据此前设定好的调度算法将用户请求负载给某台Real Server。
　假如说此时Director Server 根据调度的结果会将请求分摊到RealServer1上去,那么Director Server 会将用户的请求报文中的目标地址,从原来的VIP改为RealServer1的IP,然后再转发给RealServer1

3,此时RealServer1收到一个源地址为CIP目标地址为自己的请求,那么RealServer1处理好请求后会将一个源地址为自己目标地址为CIP的数据包通过 Director Server 发出去,

4.当Driector Server收到一个源地址为RealServer1 的IP 目标地址为CIP的数据包,此时Driector Server 会将源地址修改为VIP,然后再将数据包发送给用户

LVS-NAT的性能瓶颈：

在LVS/NAT的集群系统中，请求和响应的数据报文都需要通过负载调度器(Director)，当真实服务器(RealServer)的数目在10台和20台之间时，负载调度器(Director)将成为整个集群系统的新瓶颈。

大多数Internet服务都有这样的特点：请求报文较短而响应报文往往包含大量的数据。如果能将请求和响应分开处理，即在负载调度器(Director)中只负责调度请求而响应直接(RealServer)返回给客户，将极大地提高整个集群系统的吞吐量。

部署

在**RealServer**上部署**httpd**服务并测试

```
#安装httpd服务，创建httpd测试页面，启动httpd服务
[root@web1 ~]# yum -y install httpd
[root@web1 ~]# service httpd start
[root@web1 ~]# echo "RS1-web1 Allentuns.com" > /var/www/html/index.html
[root@web2 ~]# yum -y install httpd
[root@web2 ~]# echo "RS2-web2 Allentuns.com" > /var/www/html/index.html
[root@web2 ~]# service httpd start

#测试httpd服务是否OK!
[root@web1 ~]# curl http://localhost
RS1-web1 Allentuns.com
[root@web1 ~]# curl http://172.16.100.11
RS2-web2 Allentuns.com
```

在**Director**上部署**ipvs**服务并测试

添加集群服务

```
[root@LVS ~]# ipvsadm -A -t 192.168.0.200:80 -s rr #定义一个集群服务
[root@LVS ~]# ipvsadm -a -t 192.168.0.200:80 -r 172.16.100.10 -m #添加RealServer并指派调度算法为NAT
[root@LVS ~]# ipvsadm -a -t 192.168.0.200:80 -r 172.16.100.11 -m #添加RealServer并指派调度算法为NAT
[root@LVS ~]# ipvsadm -L -n #查看ipvs定义的规则列表
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port Forward Weight ActiveConn InActConn
TCP 192.168.0.200:80 rr
-> 172.16.100.10:80 Masq 1 0 0
-> 172.16.100.11:80 Masq 1 0 0
[root@LVS ~]# cat /proc/sys/net/ipv4/ip_forward #查看Linux是否开启路由转发功能
0
[root@LVS ~]# echo 1 > /proc/sys/net/ipv4/ip_forward #启动Linux的路由转发功能
[root@LVS ~]# cat /proc/sys/net/ipv4/ip_forward
1
```

测试访问**http**页面

```
[root@LVS ~]# curl http://192.168.0.200/index.html
RS2-web2 Allentuns.com #第一次是web2
[root@LVS ~]# curl http://192.168.0.200/index.html
RS1-web1 Allentuns.com #第二次是web1
[root@LVS ~]# curl http://192.168.0.200/index.html
RS2-web2 Allentuns.com #第三次是web1
[root@LVS ~]# curl http://192.168.0.200/index.html
RS1-web1 Allentuns.com #第四次是web2
```

更改**LVS**的调度算法并压力测试，查看结果

```
[root@LVS ~]# ipvsadm -E -t 192.168.0.200:80 -s wrr
[root@LVS ~]# ipvsadm -e -t 192.168.0.200:80 -r 172.16.100.10 -m -w 3
[root@LVS ~]# ipvsadm -e -t 192.168.0.200:80 -r 172.16.100.11 -m -w 1
[root@LVS ~]# ipvsadm -L -n
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port Forward Weight ActiveConn InActConn
TCP 192.168.0.200:80 wrr
-> 172.16.100.10:80 Masq 3 0 2
-> 172.16.100.11:80 Masq 1 0 2

[root@LVS ~]# curl http://192.168.0.200/index.html
RS1-web1 Allentuns.com
[root@LVS ~]# curl http://192.168.0.200/index.html
RS1-web1 Allentuns.com
[root@LVS ~]# curl http://192.168.0.200/index.html
RS1-web1 Allentuns.com
[root@LVS ~]# curl http://192.168.0.200/index.html
RS2-web2 Allentuns.com
[root@LVS ~]# curl http://192.168.0.200/index.html
RS1-web1 Allentuns.com
```

永久保存**LVS**规则并恢复

```
#第一种方法：
[root@LVS ~]# service ipvsadm save
ipvsadm: Saving IPVS table to /etc/sysconfig/ipvsadm: [确定]

#第二种方法：
[root@LVS ~]# ipvsadm -S > /etc/sysconfig/ipvsadm.s1
```

模拟清空**ipvsadm**规则来恢复

```
[root@LVS ~]# ipvsadm -C
[root@LVS ~]# ipvsadm -L -n
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port Forward Weight ActiveConn InActConn
[root@LVS ~]# ipvsadm -R < /etc/sysconfig/ipvsadm.s1
[root@LVS ~]# ipvsadm -L -n
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port Forward Weight ActiveConn InActConn
TCP 192.168.0.200:80 wrr
-> 172.16.100.10:80 Masq 3 0 0
-> 172.16.100.11:80 Masq 1 0 0
```

脚本

LVS-NAT服务控制脚本部署在Director上

```
#!/bin/bash
#
# chkconfig: - 88 12
# description: LVS script for VS/NAT
# . /etc/rc.d/init.d/functions # VIP=192.168.0.200
DIP=172.16.100.1
RIP1=172.16.100.10
RIP2=172.16.100.11
#
case "$1" in
start)
 # /sbin/ifconfig eth1:0 $VIP netmask 255.255.255.0 up
# Since this is the Director we must be able to forward packets
  echo 1 > /proc/sys/net/ipv4/ip_forward
 # Clear all iptables rules.
```

```
  /sbin/iptables -F
 # Reset iptables counters.
  /sbin/iptables -Z
 # Clear all ipvsadm rules/services.
  /sbin/ipvsadm -C
 # Add an IP virtual service for VIP 192.168.0.219 port 80
# In this recipe, we will use the round-robin scheduling method.
# In production, however, you should use a weighted, dynamic scheduling method.
  /sbin/ipvsadm -A -t $VIP:80 -s rr
 # Now direct packets for this VIP to
# the real server IP (RIP) inside the cluster
  /sbin/ipvsadm -a -t $VIP:80 -r $RIP1 -m
  /sbin/ipvsadm -a -t $VIP:80 -r $RIP2 -m

  /bin/touch /var/lock/subsys/ipvsadm.lock
;;

stop) # Stop forwarding packets
  echo 0 > /proc/sys/net/ipv4/ip_forward
 # Reset ipvsadm
  /sbin/ipvsadm -C
 # Bring down the VIP interface
  ifconfig eth1:0 down

  rm -rf /var/lock/subsys/ipvsadm.lock
;;

status)
  [ -e /var/lock/subsys/ipvsadm.lock ] && echo "ipvs is running..." || echo "ipvsadm is stopped..."
;;
*)
  echo "Usage: $0 {start|stop}"
;; esac

lvs-nat-director.sh
```

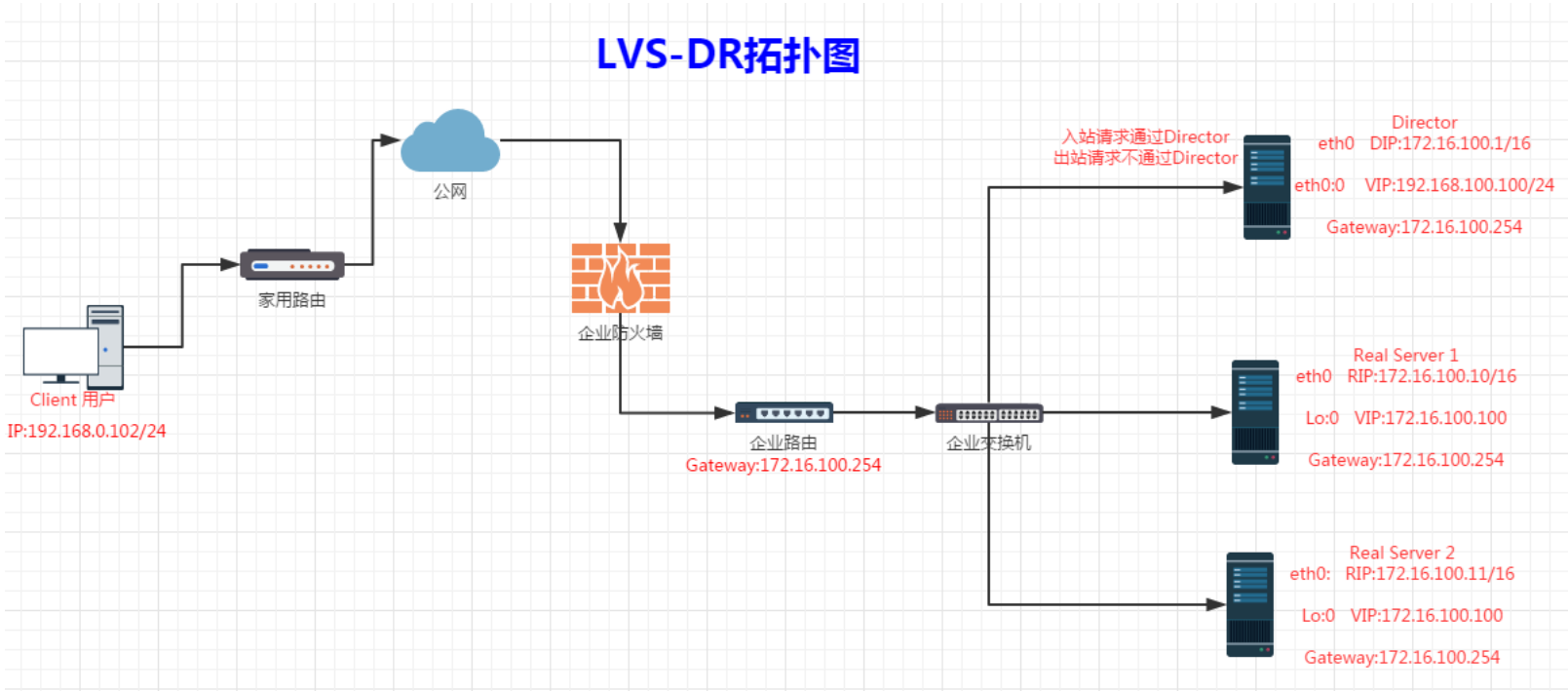
分享LVS-NAT一键安装脚本

```
#!/bin/bash
#
# 一键安装lvs-nat脚本，需要注意的是主机名成和ip的变化稍作修改就可以了
HOSTNAME=`hostname`

Director='LVS'
VIP="192.168.0.200"
RIP1="172.16.100.10"
RIP2="172.16.100.11"
RealServer1="web1"
RealServer2="web2"
Httpd_config="/etc/httpd/conf/httpd.conf"

 #Director Server Install configure ipvsadm
if [ "$HOSTNAME" = "$Director" ];then
ipvsadm -C
yum -y remove ipvsadm
yum -y install ipvsadm
/sbin/ipvsadm -A -t $VIP:80 -s rr
/sbin/ipvsadm -a -t $VIP:80 -r $RIP1 -m
/sbin/ipvsadm -a -t $VIP:80 -r $RIP2 -m
 echo 1 > /proc/sys/net/ipv4/ip_forward
 echo "============================================================" echo "Install  $Director sucess   Tel:13260071987
Qq:467754239" echo "============================================================" fi



 #RealServer Install htpd
if [ "$HOSTNAME" = "$RealServer1" ];then
yum -y remove httpd
rm -rf /var/www/html/index.html
yum -y install httpd
echo "web1 Allentuns.com" > /var/www/html/index.html
sed -i '/#ServerName www.example.com:80/a\ServerName localhost:80' $Httpd_config
service httpd start

 echo "============================================================" echo "Install $RealServer1 success Tel:13260071987
Qq:467754239" echo "============================================================" fi
 if [ "$HOSTNAME" = "$RealServer2" ];then
yum -y remove httpd
rm -rf /var/www/html/index.html
yum -y install httpd
```

```
echo "web2 Allentuns.com" > /var/www/html/index.html
sed -i '/#ServerName www.example.com:80/a\ServerName localhost:80' $Httpd_config
service httpd start
echo "Install $RealServer2"

 echo "============================================================" echo "Install $RealServer1 success Tel:13260071987
Qq:467754239" echo "============================================================" fi

lvs-nat-install
```

LVS-DR：直接路由

架构图：



工作方式：

上面说了NAT模型的实现方式,那么NAT模型有个缺陷,因为进出的每个数据包都要经过Director Server,当集群系统负载过大的时候Director Server将会成为整个集群系统的瓶颈,

那么DR模型就避免了这样的情况发生,DR模型在只有请求的时候才会经过Director Server, 回应的数据包由Real Server 直接响应用户不需要经过Director Server,其实三种模型中最常用的也就是DR模型了。

下面是它的工作流程：

```
1, 首先用户用CIP请求VIP

2, 根据上图可以看到,不管是Director Server还是Real Server上都需要配置VIP,那么当用户请求到达我们的集群网络的前端路由器的时候,请求数据包的源地址为CIP目标地址为VIP,
   此时路由器会发广播问谁是VIP,那么我们集群中所有的节点都配置有VIP,此时谁先响应路由器那么路由器就会将用户请求发给谁,这样一来我们的集群系统是不是没有意义了,
   那我们可以在网关路由器上配置静态路由指定VIP就是Director Server,或者使用一种机制不让Real Server 接收来自网络中的ARP地址解析请求,这样一来用户的请求数据包都会经过Director Servrer

3,当Director Server收到用户的请求后根据此前设定好的调度算法结果来确定将请求负载到某台Real Server上去,假如说此时根据调度算法的结果,会将请求负载到RealServer 1上面去,
  此时Director Server 会将数据帧中的目标MAC地址修改为Real Server1的MAC地址,然后再将数据帧发送出去

4,当Real Server1 收到一个源地址为CIP目标地址为VIP的数据包时,Real Server1发现目标地址为VIP,而VIP是自己,于是接受数据包并给予处理,当Real Server1处理完请求后,
  会将一个源地址为VIP目标地址为CIP的数据包发出去,此时的响应请求就不会再经过Director Server了,而是直接响应给用户。
```

编辑**DR**有三种方式（目的是让用户请求的数据都通过**Director Server**）

第一种方式：在路由器上明显说明vip对应的地址一定是Director上的MAC，只要绑定，以后再跟vip通信也不用再请求了，这个绑定是静态的，所以它也不会失效，也不会再次发起请求，但是有个前提，我们的路由设备必须有操作权限能够绑定MAC地址，万一这个路由器是运行商操作的，我们没法操作怎么办？第一种方式固然很简便，但未必可行。

第二种方式：在给别主机上（例如：红帽）它们引进的有一种程序arptables,它有点类似于iptables,它肯定是基于arp或基于MAC做访问控制的，很显然我们只需要在每一个real server上定义arptables规则，如果用户arp广播请求的目标地址是本机的vip则不予相应，或者说相应的报文不让出去，很显然网关（gateway）是接受不到的，也就是director相应的报文才能到达gateway，这个也行。第二种方式我们可以基于arptables。

第三种方式：在相对较新的版本中新增了两个内核参数(kernelparameter)，第一个是arp_ignore定义接受到ARP请求时的相应级别;第二个是arp_announce定义将自己地址向外通告是的通告级别。【提示：很显然我们现在的系统一般在内核中都是支持这些参数的，我们用参数的方式进行调整更具有朴实性，它还不依赖于额外的条件，像arptables,也不依赖外在路由配置的设置，反而通常我们使用的是第三种配置】

arp_ignore:定义接受到ARP请求时的相应级别

0：只要本地配置的有相应地址，就给予响应。

1：仅在请求的目标地址配置请求到达的接口上的时候，才给予响应（当别人的arp请求过来的时候，如果接收的设备上面没有这个ip，就不响应，默认是0，只要这台机器上面任何一个设备上面有这个ip，就响应arp请求，并发送MAC地址应答。）

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：定义将自己地址向外通告是的通告级别;

0：将本地任何接口上的任何地址向外通告

1：试图仅想目标网络通告与其网络匹配的地址

2：仅向与本地借口上地址匹配的网络进行通告

部署

在Real Server1 和Real Server2上做以下配置

```
# 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
#以上命令需填加到/etc/rc.local文件中让其开机自动生效

# vim /etc/sysconfig/network-scripts/ifcfg-lo:0 内容如下
DEVICE=lo:0
IPADDR=172.16.100.100
NETMASK=255.255.255.255
BROADCAST=172.16.100.100
ONBOOT=yes
NAME=loopback


# ifdown lo:0
# ifup lo:0
# route add -host 172.16.100.100 dev lo:0
# echo "route add -host 172.16.100.100 dev lo:0" >> /etc/rc.local
```

在Director Server上做以下配置

```
# vim /etc/sysconfig/network-scripts/ifcfg-eth2:0 内容如下
DEVICE=eth2:0
IPADDR=172.16.100.100
NETMASK=255.255.255.255
BROADCAST=172.16.100.100
ONBOOT=yes # ifdown eth2:0

#命令
# ifup eth2:20
# route add -host 172.16.100.100 dev eth2:0
# echo "route add -host 172.16.100.100 dev eth2:0" >> /etc/rc.local
# echo "1" > /proc/sys/net/ipv4/ip_forward
# echo "echo "1" > /proc/sys/net/ipv4/ip_forward" >> /etc/rc.local
# ipvsadm -A -t 172.16.100.100:80 -s wlc
# ipvsadm -a -t 172.16.100.100:80 -r 172.16.100.10 -g -w 2
# ipvsadm -a -t 172.16.100.100:80 -r 172.16.100.11 -g -w 1
```

脚本

Director脚本

```
#!/bin/bash
#
# LVS script for VS/DR
# . /etc/rc.d/init.d/functions # VIP=172.16.100.100
RIP1=172.16.100.10
RIP2=172.16.100.11
PORT=80
 #
case "$1" in
start)

  /sbin/ifconfig eth2:0 $VIP broadcast $VIP netmask 255.255.255.255 up
  /sbin/route add -host $VIP dev eth2:0
 # Since this is the Director we must be able to forward packets
  echo 1 > /proc/sys/net/ipv4/ip_forward
 # Clear all iptables rules.
  /sbin/iptables -F
 # Reset iptables counters.
  /sbin/iptables -Z
 # Clear all ipvsadm rules/services.
  /sbin/ipvsadm -C
 # Add an IP virtual service for VIP 192.168.0.219 port 80
# In this recipe, we will use the round-robin scheduling method.
# In production, however, you should use a weighted, dynamic scheduling method.
  /sbin/ipvsadm -A -t $VIP:80 -s wlc
 # Now direct packets for this VIP to
# the real server IP (RIP) inside the cluster
  /sbin/ipvsadm -a -t $VIP:80 -r $RIP1 -g -w 1
  /sbin/ipvsadm -a -t $VIP:80 -r $RIP2 -g -w 2

  /bin/touch /var/lock/subsys/ipvsadm &> /dev/null
;;

stop) # Stop forwarding packets
```

```
  echo 0 > /proc/sys/net/ipv4/ip_forward
 # Reset ipvsadm
  /sbin/ipvsadm -C
 # Bring down the VIP interface
  /sbin/ifconfig eth2:0 down
  /sbin/route del $VIP

  /bin/rm -f /var/lock/subsys/ipvsadm

  echo "ipvs is stopped..."
;;

status)
  if [ ! -e /var/lock/subsys/ipvsadm ]; then
    echo "ipvsadm is stopped ..."
  else
    echo "ipvs is running ..."
    ipvsadm -L -n
  fi
;;
*)
  echo "Usage: $0 {start|stop|status}"
;; esac

Director.sh
```

RealServer脚本

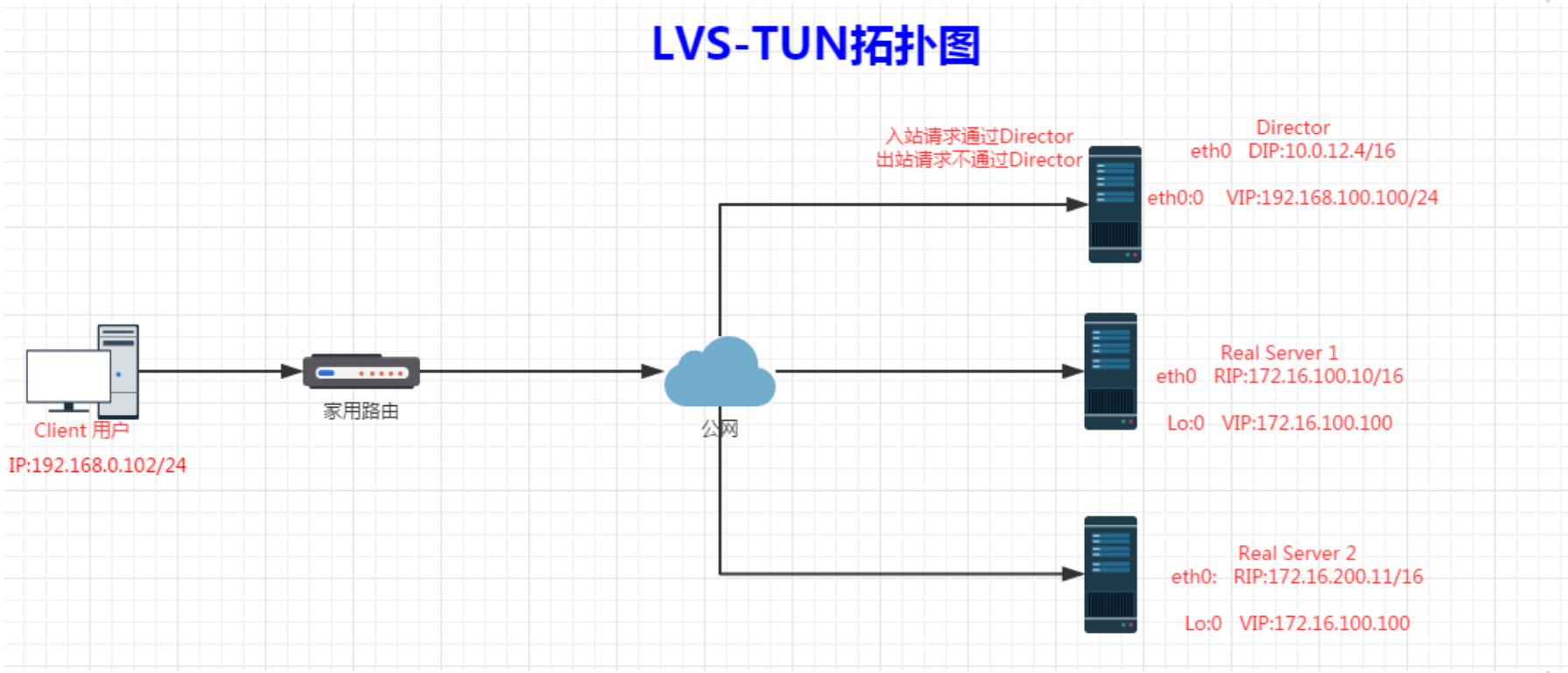
```
#!/bin/bash
#
# Script to start LVS DR real server.
# description: LVS DR real server
# .  /etc/rc.d/init.d/functions

VIP=172.16.100.100
host=`/bin/hostname`
 case "$1" in
start)
```

```
        # Start LVS-DR real server on this machine.
        /sbin/ifconfig lo down
        /sbin/ifconfig lo up
        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

        /sbin/ifconfig lo:0 $VIP broadcast $VIP netmask 255.255.255.255 up
        /sbin/route add -host $VIP dev lo:0

;;
stop)

        # Stop LVS-DR real server loopback device(s).
        /sbin/ifconfig lo:0 down
        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

;;
status)

        # Status 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
            # Either the route or the lo:0 device
            # not found.             echo "LVS-DR real server Stopped."
        else
            echo "LVS-DR real server Running."
        fi
;;
*)
            # Invalid entry.
            echo "$0: Usage: $0 {start|status|stop}"
            exit 1
;; esac
```

RealServer.sh

LVS-TUN：隧道

架构图：



工作方式：

TUN的工作机制跟DR一样，只不过在转发的时候，它需要重新包装IP报文。这里的real server（图中为RIP）离得都比较远。

用户请求以后，到director上的VIP上，它跟DR模型一样，每个realserver上既有RIP又有VIP，Director就挑选一个real server进行响应，但director和real server并不在同一个网络上，这时候就用到隧道了，Director进行转发的时候，一定要记得CIP和VIP不能动。

我们转发是这样的，让它的CIP和VIP不动，在它上面再加一个IP首部，再加的IP首部源地址是DIP，目标地址的RIP的IP地址。收到报文的RIP，拆掉报文以后发现了里面还有一个封装，它就知道了，这就是隧道。

其实数据转发原理和DR是一样的，不过这个我个人认为主要是位于不同位置（不同机房）；LB是通过隧道进行了信息传输，虽然增加了负载，可是因为地理位置不同的优势，还是可以参考的一种方案；

优点：负载均衡器只负责将请求包分发给物理服务器，而物理服务器将应答包直接发给用户。所以，负载均衡器能处理很巨大的请求量，这种方式，一台负载均衡能为超过100台的物理服务器服务，负载均衡器不再是系统的瓶颈。

使用VS-TUN方式，如果你的负载均衡器拥有100M的全双工网卡的话，就能使得整个Virtual Server能达到1G的吞吐量。

不足：但是，这种方式需要所有的服务器支持"IP Tunneling"(IP Encapsulation)协议；

LVS的健康状态检查

在LVS模型中，director不负责检查RS的健康状况，这就使得当有的RS出故障了，director还会将服务请求派发至此服务器，这种情况对用户、企业都是很不爽的，哪个用户倒霉说不定就遇到类似了。

为了让Director更人性化、可靠还要给director提供健康检查功能；如何实现？Director没有自带检查工具，只有手动编写脚本给director实现健康状态检查功能！

```
#!/bin/bash
# VIP=172.16.100.100
CPORT=80
FAIL_BACK=127.0.0.1
RS=("172.16.100.10" "172.16.100.11")
declare -a RSSTATUS
RW=("2" "1")
RPORT=80
TYPE=g
CHKLOOP=3
LOG=/var/log/ipvsmonitor.log
addrs() {
ipvsadm -a -t $VIP:$CPORT -r $1:$RPORT -$TYPE -w $2
[ $? -eq 0 ] && return 0 || return 1
}
delrs() {
ipvsadm -d -t $VIP:$CPORT -r $1:$RPORT
[ $? -eq 0 ] && return 0 || return 1
}
checkrs() {  local I=1 while [ $I -le $CHKLOOP ]; do if curl --connect-timeout 1 http://$1 &> /dev/null; then return
0 fi let I++ done return 1
```

```
}
initstatus() {  local I local COUNT=0;
for I in ${RS[*]}; do if ipvsadm -L -n | grep "$I:$RPORT" && > /dev/null ; then
RSSTATUS[$COUNT]=1
else
RSSTATUS[$COUNT]=0
A++
Dir[0]=$A  fi let COUNT++
done
}
initstatus  while :; do let COUNT=0 for I in ${RS[*]}; do if checkrs $I; then if [ ${RSSTATUS[$COUNT]} -eq 0 ]; then
addrs $I ${RW[$COUNT]}
[ $? -eq 0 ] && RSSTATUS[$COUNT]=1 && echo "`date +'%F %H:%M:%S'`, $I is back." >> $LOG
fi else if [ ${RSSTATUS[$COUNT]} -eq 1 ]; then
delrs $I
[ $? -eq 0 ] && RSSTATUS[$COUNT]=0 && echo "`date +'%F %H:%M:%S'`, $I is gone." >> $LOG
fi fi
let COUNT++ done sleep 5 done

check-lvs-health.sh
```

参考资料:

http://www.cnblogs.com/lixigang/p/5371815.html

分类: 运维

标签: 运维, shell, lvs

MacoLee
关注 - 8
粉丝 - 36
+加关注

« 上一篇：HAProxy安装配置详解
» 下一篇：LVS+Keepalived负载均衡配置

posted on 2016-09-09 15:36 MacoLee 阅读(18312) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

相关博文：
· LVS 详解
· lvs原理及安装部署详解（参考）
· LVS原理详解
· 使用 LVS 实现负载均衡原理及安装配置详解
· lvs keepalived 安装配置详解

最新新闻：
· 滴滴又迎来一个新对手，上汽集团推出「享道出行」打车服务
· 互金行业进入最冷一年：减员两到三成 薪酬普遍腰斩
· 美光科技第一财季营收79.13亿美元 净利同比增23%
· 《自然》年度十大人物：天才少年曹原居首，贺建奎成反面

· ARM推出针对自动驾驶汽车传感器的芯片
» 更多新闻...