7.模拟故障

主调度器出故障：

备的服务器（BACKUP）：

用户访问正常：（Ctrl+F5）

健康状态检查：

进行实验：

检查：

主服务器：192.168.200.107

备用服务器：192.168.200.108

web节点服务器：DR模式

nginx+keepalived高可用负载均衡

1.停止ARP解析

2.开始实验

3.实现高可用

监控脚本：当nginx出问题，就把keepalived关掉

编辑配置文件：（keepalived） 192.168.200.107&192.168.200.107

模拟故障：当杀掉主的nginx时，nginx会不会漂到从上

修复nginx：

非抢占：（主的在抢，所以在配置时，只给主的配置就可以，备的就不用配）

小提问：（面试）

基于 Haproxy 构建负载均衡集群

1、HAPROXY简介

2、HAProxy的特点

3、haproxy配置文件的五部分

解决调度器单点问题：

1. 会构建双机热备系统
2. 会构建LVS+HA高可用群集

一、Keepalived双机热备基础知识

Keepalived起初是专门针对LVS设计的一款强大的辅助工具，
它和LVS的整合度非常好，它也可以作为其他软件的高可用产品方案
主要用来提供故障切换(Failover)和健康检查(Health Checking)功能
判断 LVS负载调度器、节点服务器的可用性
当master主机出现故障及时切换到backup节点保证业务正常
当masterr故障主机恢复后将其重新加入群集并且业务重新切换回master节点

Keepalived有两个功能：

1、故障自动切换

2、节点健康状态检查

Keepalived有两个工作模式：

1、抢占模式（默认）

2、非抢占模式 ----一般在工作中配非抢占

业务需求：不管VIP在什么机器上面，只要VIP能正常稳定的提供就可以了

在抢占模式中，会有中断，会影响用户体验

1. Keepalived概述及安装

Keepalived的官方网站位于http://www. keepalived. org/
本章将以YUM方式讲解Keepalived的安装，配置和使用过程
在非LVS群集环境中使用时，Keepalived 也可以作为热备软件使用

2. Keepalived 的热备方式

Keepalived ---底层协议 VRRP
Keepalived采用VRRP (Virtual Router Redundancy Protocol， 虚拟路由冗余协议)
热备份协议，以软件的方式实现Linux服务器的多机热备功能。
VRRP是针对路由器的一种备份解决方案
由多台路由器组成一个热备组，通过共用的虚拟IP地址对外提供服务;
每个热备组内同一时刻只有一台主路由器提供服务，其他路由器处于冗余状态。
若当前在线的路由器失效，
则其他路由器会自动接替(优先级决定接替顺序)虚拟IP地址，以继续提供服务，
如图所示
这种热备的方案，对外的表现是一个整体，和一台路由器一样，具备高可用的效果

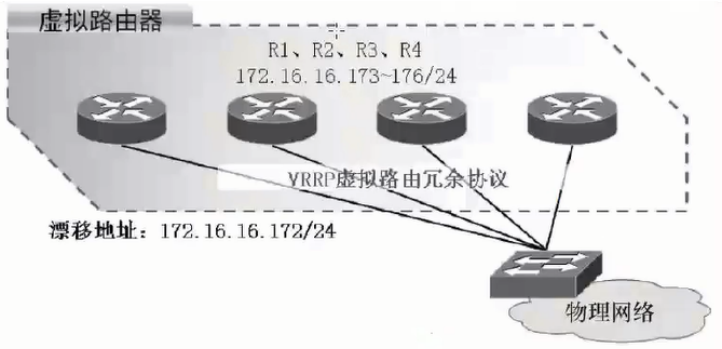


图 3.1　Keepalived 的 VRRP 热备机制

热备组内的每台路由器都可能成为主路由器，

虚拟路由器的IP地址(VIP)可以在热备组内的路由器之间进行转移，

所以也称为漂移IP地址。使用Keepalived时，

漂移地址的实现不需要手动建立虚接口配置文件(如ens33:0)，

而是由Keepalived根据配置文件自动管理

3. Keepalived 的安装与服务控制

1.安装Keepalived

在CentOS 7系统中，使用YUM方式安装keepalived. x86_ 64 0:1.2. 13-8.e17，

会自动安装Keepalived所需的软件包。

除此之外，在LVS群集环境中应用时，也需要用到ipvsadm管理工具

2.实验环境：

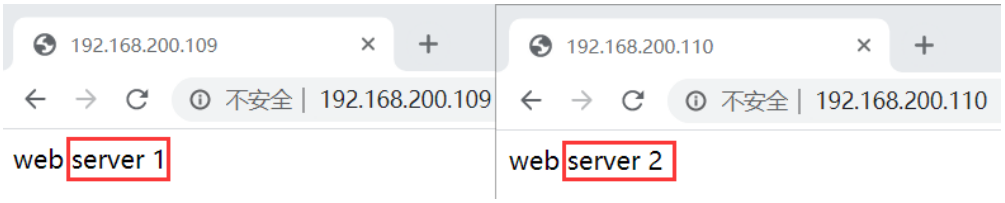
节点服务器：192.168.200.109　　192.168.200.110

安装httpd

```
[root@real2 ~]# yum -y install httpd
[root@real2 ~]# systemctl start httpd
```

写一个测试页：[root@real1 ~]# echo "web server 1" > /var/www/html/index.html

[root@real2 ~]# echo "web server 2" > /var/www/html/index.html



3.调度器配置：

```
[root@master ~]#  yum -y install keepalived  ipvsadm
[root@backup ~]# yum -y install keepalived  ipvsadm
```

4.开机自启动：

```
[root@master ~]# systemctl enable keepalived
[root@backup ~]# systemctl enable keepalived
```

4. LVS+Keepalived高可用群集

Keepalived的设计目标是构建高可用的LVS负载均衡群集（两个角色的集群合到一起）

Keepalived可以调用ipvsadm工具

- 创建虚拟服务器、管理服务器池
- 不仅仅用作双机热备

使用Keepalived构建LVS群集，更加简便易用，

主要优势

- 对LVS负载调度器实现热备切换，提高可用性
- 对服务器池中的节点进行健康检查，自动移除失效节点，恢复后再重新加入

实验准备

在基于LVS+Keepalived实现的LVS群集结构中，

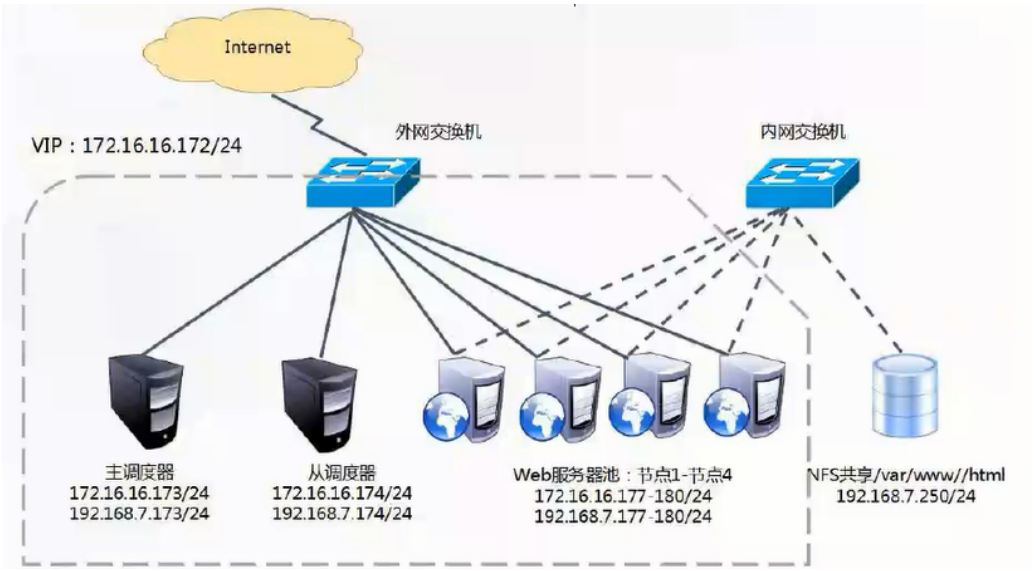
至少包括两台热备的负载调度器，三台以上的节点服务器。

本节将以DR模式的LVS群集为基础，增加一台从负载调度器，

使用Keepalived来实现主、从调度器的热备，

从而构建兼有负载均衡、高可用两种能力的LVS网站群集平台，

如图所示



LVS+Keepalived高可用群集

使用Keepalived构建LVS群集时，也需要用到ipvsadm管理工具。

但大部分工作会由Keepalived自动完成，

不需要手动执行ipvsadm（除了查看和监控群集以外）。

下面主要讲解Keepalived的服务器池设置，

关于NPS共享服务的配置、Keepalived 的热备配置等在此不再详细阐述

5.实验说明

1.主配置文件说明

```
[root@localhost~ ]# vim /etc/keepalived/keepalived. conf
.....                                                    //省略部分信息
virtual_ server 172. 16. 16. 172  80 {                   //虚拟服务器地址:(VIP)、端口
delay_loop 15                                            //健康检查的间隔时间(秒)
lb_algo rr                                               //轮询(rr)调度算法
lb_kind DR                                               //直接路由(DR)群集工作模式
persistence 60                                           //连接保持时间(秒)，若启用请去掉!号,一般不用
protocol TCP                                             //应用服务采用的是TCP协议

real_server 172. 16.16.177  80 {                         //第一个Web节点的地址、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节点的地址、端口
……                                                       //省略部分信息
}
real_server 172. 16.16.179 80 {                          //第三个Web节点的地址、端口
……              //省略部分信息
real_server 172. 16.16. 180 80 {                         //第四个Web节点的地址、端口
……              //省略部分信息
}
}
```

2.Web服务器池配置 ----LVS的功能

在Keepalieved的热备配置基础上添加“virtual_ server VIP端口{ ... }”区段来配置虚拟服务器，主要包括对参数的设置：

负载调度算法、

群集工作模式

健康检查间隔

真实服务器地址

3.重新启动Keepalived服务

重新启动Keepalived服务的命令如下

```
[root@localhost~ ]# systemctl restart keepalived
```

4.配置Web节点服务器

根据所选择的群集工作模式不同(DR 或NAT)，节点服务器的配置也有些差异

以DR模式为例，除了需要调整/proc系统的ARP响应参数以外

还需要为虚拟接口lo:0配置VIP地址，并添加一条到VIP的本地路由

5.测试LVS+Keepalived高可用群集

在客户机的浏览器中，

能够通过LVS+Keepalived群集的VIP地址(172. 16. 16. 172)正常访问Web页面内容。

当主、从调度器任何一个失效时，

Web 站点仍然可以访问(可能需要刷新或者重新打开浏览器)；

只要服务器池有两台及以上的真实服务器可用，就可以实现访问量的负载均衡。

通过主、从调度器的/var/log/messages日志文件，可以跟踪故障切换过程；

若要查看负载分配情况，可以执行“ipvsadm -ln” “ipvsadm -lnc” 等操作命令。

最终可以验证LVS+Keepalived高可用负载均衡群集的健壮性。

6.实验

正式开始实验：

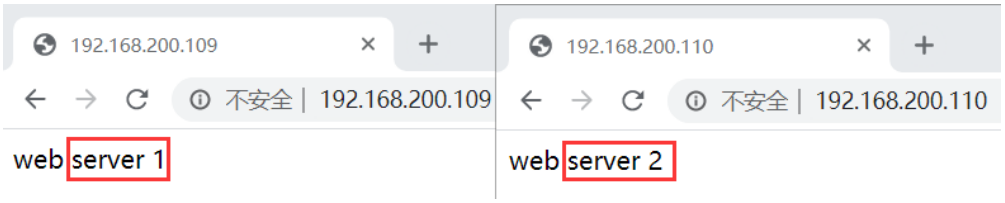
web节点服务器：192.168.200.109 192.168.200.110

安装httpd

```
[root@real2 ~]# yum -y install httpd
[root@real2 ~]# systemctl start httpd
```

写一个测试页：[root@real1 ~]# echo "web server 1" > /var/www/html/index.html

[root@real2 ~]# echo "web server 2" > /var/www/html/index.html



前期准备：192.168.200.109（110） 脚本

```
[root@master ~]# vim realserver.sh
#!/bin/bash
SNS_VIP=192.168.200.254
```

```
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/sys/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 "RealServer Stoped"
        ;;
*)
        echo "Usage: $0 {start|stop}"
exit 1
esac
exit 0
```

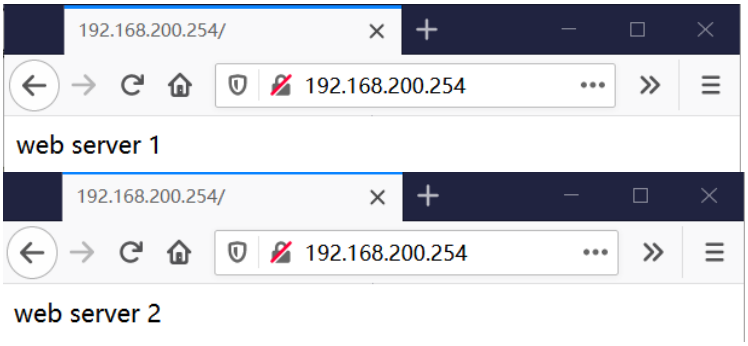
192.168.200.107 主调度器

```
[root@master ~]# cd /etc/keepalived
[root@master keepalived]# ls
keepalived.conf
[root@master keepalived]# cp keepalived.conf keepalived.conf.bak
[root@master ~]# vim /etc/keepalived/keepalived.conf
    ! Configuration File for keepalived
    global_defs {                # 全局配置（跟发邮件相关）；一般不用做修改
       router_id LVS_MASTER           # 主调度器的名称（master，slave）
    }
```

```
# 热备实例
vrrp_instance VI_1 {
#  热备的名字，如果master，slave想共用一个VIP，它们在一个实例当中，名字必须相同
    state MASTER              #  声明状态，“我是BACKUP服务器”
    interface ens32              # 它俩相互通信，相互健康检查的心跳接口 --网卡名
    virtual_router_id 51             #注意这个取值不能冲突（0-255）
    priority 100                     # 优先级
    advert_int 1
    # 每隔一秒检查一次（看看另一台是否存活，挂了就把资源抢过来）
    authentication {                           # 主、从热备认证信息
        auth_type PASS
        auth_pass 123456
    }
    virtual_ipaddress {                    # 指定集群VIP地址
        192.168.200.254
    }
}
virtual_server 192.168.200.254 80 {
    delay_loop 3
    lb_algo rr
    lb_kind DR
    protocol TCP

    real_server 192.168.200.109 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 3
            delay_before_retry 3
        }
    }
    real_server 192.168.200.110 80 {
```

```
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 3
            delay_before_retry 3
        }
    }
}
```



LVS节点模式：

- lo口上配IP
- 添加路由
- 修改ARP相关参数
- 脚本“realserver.sh”

```
[root@real1 ~]# bash realserver.sh start
RealServer Start OK
```

配置192.168.200.110:

```
[root@real2 ~]# bash realserver.sh start
RealServer Start OK
```

检查：

```
[root@real2 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
[root@real2 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
```

```
0.0.0.0         192.168.200.1   0.0.0.0         UG    100    0        0 ens32
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
192.168.200.0   0.0.0.0         255.255.255.0   U     100    0        0 ens32
192.168.200.254 0.0.0.0         255.255.255.255 UH    0      0        0 lo
```

到这节点搞定~

用主调度器打开两个终端，一个监控/var/log/messages,一个启服务

```
[root@master ~]# systemctl start keepalived
[root@master ~]# tail -f /var/log/messages
```

将主配置文件远程传输到192.168.200.108（从）

```
[root@master keepalived]# scp keepalived.conf 192.168.200.108:/etc/keepalived/
```

配置从的主配置文件

```
[root@backup ~]# vim /etc/keepalived/keepalived.conf
router_id LVS_BACKUP
state BACKUP
priority 90
```

7.模拟故障

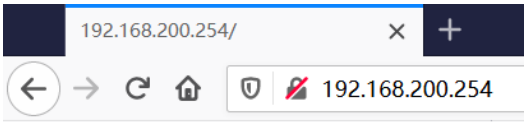
主调度器出故障:

```
[root@master ~]# systemctl stop keepalived.service
```

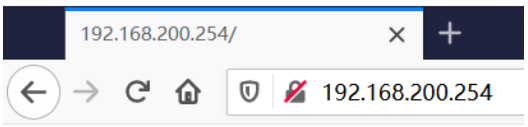
备的服务器（BACKUP）:

```
Apr 21 19:59:50 localhost Keepalived_vrrp[1516]: VRRP_Instance(VI_1) Transition to MASTER STATE
Apr 21 19:59:51 localhost Keepalived_vrrp[1516]: VRRP_Instance(VI_1) Entering MASTER STATE
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group defaul
00
    link/ether 00:0c:29:88:4e:e1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.108/24 brd 192.168.200.255 scope global noprefixroute ens32
       valid_lft forever preferred_lft forever
    inet 192.168.200.254/32 scope global ens32
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe88:4ee1/64 scope link
       valid_lft forever preferred_lft forever
```

用户访问正常：（Ctrl+F5）



web server 1



web server 2

一切以主的为准，实现故障切换

健康状态检查:

```
[root@real1 ~]# systemctl stop httpd
```

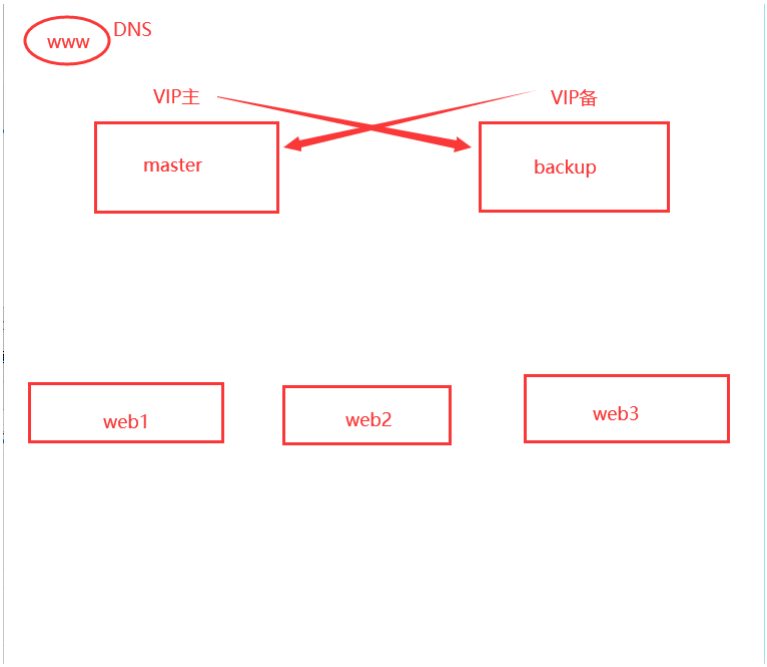
```
Apr 21 20:38:01 localhost systemd: Starting Session 112 of user root.
Apr 21 20:38:22 localhost Keepalived_healthcheckers[1515]: TCP connection to [192.168.200.109]:80 failed.
Apr 21 20:38:25 localhost Keepalived_healthcheckers[1515]: TCP connection to [192.168.200.109]:80 failed.
```

[root@real1 ~]# systemctl start httpd

Apr 21 20:40:49 localhost Keepalived_healthcheckers[1515]: Adding service [192.168.200.109]:80 to VS [192.168.200.254]:80

利用BACKUP

互为主备：DNS调度



这样两台机器就不会有处于闲置的状态的机器，老板就不会抱怨了

进行实验：

[root@master ~]# cat /etc/keepalived/keepalived.conf

.....

```
########################################
vrrp_instance VI_2 {
    state BACKUP
    interface ens32
    virtual_router_id 52
    priority 90
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.200.253
```

```
    }
}

virtual_server 192.168.200.253 80 {
    delay_loop 3
    lb_algo rr
    lb_kind DR
    protocol TCP

    real_server 192.168.200.109 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 3
            delay_before_retry 3
        }
    }
    real_server 192.168.200.110 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 3
            delay_before_retry 3
        }
    }
}
[root@backup ~]# cat /etc/keepalived/keepalived.conf
.....
##############################
vrrp_instance VI_2 {
    state MASTER
    interface ens32
    virtual_router_id 52
    priority 100
    advert_int 1
```

```
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.200.253
    }
}

virtual_server 192.168.200.253 80 {
    delay_loop 3
    lb_algo rr
    lb_kind DR
    protocol TCP

    real_server 192.168.200.109 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 3
            delay_before_retry 3
        }
    }
    real_server 192.168.200.110 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 3
            delay_before_retry 3
        }
    }
}
```

检查：

主服务器：192.168.200.107

```
[root@master ~]# ipvsadm -Ln
```

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.200.253:80 rr
  -> 192.168.200.109:80           Route   1      0          0
  -> 192.168.200.110:80           Route   1      0          0
TCP  192.168.200.254:80 rr
  -> 192.168.200.109:80           Route   1      0          0
  -> 192.168.200.110:80           Route   1      0          0
[root@master ~]# ip a
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 00:0c:29:c7:a5:46 brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.107/24 brd 192.168.200.255 scope global noprefixroute ens32
       valid_lft forever preferred_lft forever
    inet 192.168.200.254/32 scope global ens32
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fec7:a546/64 scope link
       valid_lft forever preferred_lft forever
```

备用服务器：192.168.200.108

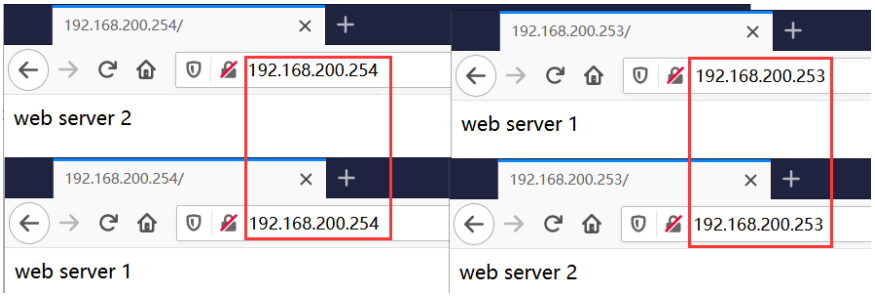
```
[root@backup ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.200.253:80 rr
  -> 192.168.200.109:80           Route   1      0          0
  -> 192.168.200.110:80           Route   1      0          0
TCP  192.168.200.254:80 rr
  -> 192.168.200.109:80           Route   1      0          0
  -> 192.168.200.110:80           Route   1      0          0
[root@backup ~]# ip a
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 00:0c:29:88:4e:e1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.108/24 brd 192.168.200.255 scope global noprefixroute ens32
```

```
       valid_lft forever preferred_lft forever
    inet 192.168.200.253/32 scope global ens32
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe88:4ee1/64 scope link
```

web节点服务器：DR模式

```
[root@real2 ~]# cat realserver.sh.bak
#!/bin/bash
SNS_VIP=192.168.200.253
case "$1" in
start)
        ifconfig lo:1 $SNS_VIP netmask 255.255.255.255 broadcast $SNS_VIP
        /sbin/route add -host $SNS_VIP dev lo:1
        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
        sysctl -p >/dev/null 2>&1
        echo "RealServer Start OK"
        ;;
stop)
        ifconfig lo:1 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 "RealServer Stoped"
        ;;
*)
        echo "Usage: $0 {start|stop}"
exit 1
esac
exit 0
```

测试：



nginx+keepalived高可用负载均衡

做下一个实验（用nginx做负载调度器）之前的环境准备：

- **抢占及非抢占**
- **nginx做负载调度器**

脚本处理，可以用一个脚本，修改来实现对254,253的修改

1.停止ARP解析

```
[root@real1 ~]# bash realserver.sh.bak stop
RealServer Stoped
[root@real1 ~]# bash realserver.sh stop
RealServer Stoped
```

2.停止keepalived服务

```
[root@backup ~]# systemctl stop keepalived.service
```

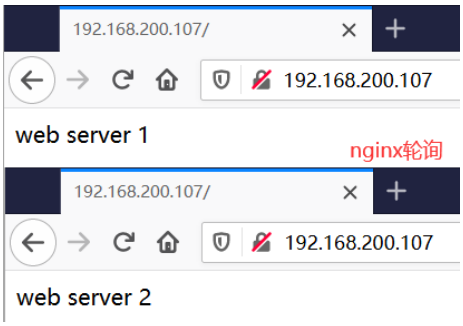
2.开始实验

192.168.200.107（108）：

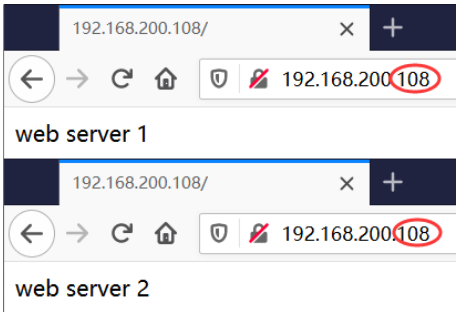
```
[root@master ~]# rz:
 nginx-1.15.9-1.x86_64.rpm
[root@master ~]# rpm -ivh nginx-1.15.9-1.x86_64.rpm
[root@master ~]# vim /usr/local/nginx/conf/nginx.conf
 35     upstream webserver {
 36         server 192.168.200.109:80 weight=1;
 37         server 192.168.200.110:80 weight=1;
 38     }
 48         location / {
 49             proxy_pass http://webserver;
 50         }
 52         location ~ \.html$ {
 53             root   html;
 54             index  index.html index.html;
```

```
 55          }
[root@master ~]# killall -HUP nginx
```

测试：




```
[root@master ~]# scp /usr/local/nginx/conf/nginx.conf
192.168.200.108:/usr/local/nginx/conf/
[root@backup ~]# killall -HUP nginx
```



3.实现高可用

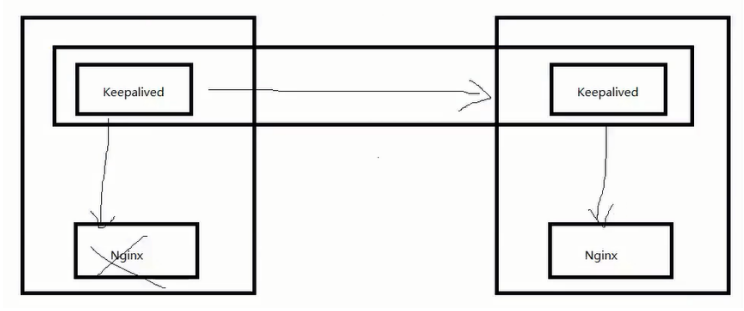
```
[root@master ~]# vim /etc/keepalived/keepalived.conf
! Configuration File for keepalived
global_defs {
   router_id LVS_MASTER
}
vrrp_instance VI_1 {
    state MASTER
    interface ens32
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
```

```
        192.168.200.254
    }
}
[root@backup ~]# cat /etc/keepalived/keepalived.conf
! Configuration File for keepalived
global_defs {
   router_id LVS_BACKUP
}

vrrp_instance VI_1 {
    state BACKUP
    interface ens32
    virtual_router_id 51
    priority 90
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.200.254
    }
}
```

测试：192.168.200.254

此时VIP是在192.168.200.107上的，108上没有VIP



问题：

否则的的话，两者是分离的，当nginx出现问题时，keepalived仍然对外提供VIP

解决办法：

使用shell脚本来让keepalived实时检查nginx 的状态

三种检查进程的方法：

```
[root@master ~]# ps aux | grep nginx
root       8876  0.0  0.2  46080  2036 ?        Ss   02:43   0:00 nginx: master
process nginx
nginx      9124  0.0  0.2  48624  2356 ?        S    03:03   0:00 nginx: worker
process
nginx      9125  0.0  0.2  48624  2356 ?        S    03:03   0:00 nginx: worker
process
root       9428  0.0  0.5 152080  5828 pts/0    S+   03:29   0:00 vim
/opt/check_nginx.sh
root       9491  0.0  0.0 112720   984 pts/1    R+   03:30   0:00 grep --
color=auto nginx
[root@master ~]# ps -C nginx
   PID TTY          TIME CMD
  8876 ?        00:00:00 nginx
  9124 ?        00:00:00 nginx
  9125 ?        00:00:00 nginx
[root@master ~]# ps -C nginx --no-header
  8876 ?        00:00:00 nginx
  9124 ?        00:00:00 nginx
  9125 ?        00:00:00 nginx
```

查看进程数量：

```
[root@master ~]# ps -C nginx --no-header | wc -l
3
```

监控脚本：当nginx出问题，就把keepalived关掉

```
[root@master ~]# vim /opt/check_nginx.sh
#!/bin/bash
nnum=$( ps -C nginx --no-header | wc -l )
if [ $nnum -eq 0 ]
then
    /usr/local/nginx/sbin/nginx
    sleep 2
```

```
    nnum=$( ps -C nginx --no-header | wc -l )
    if [ $nnum -eq 0 ]
    then
        systemctl stop keepalived
    fi
fi
[root@master ~]# chmod +x /opt/check_nginx.sh
```

编辑配置文件：（keepalived） 192.168.200.107&192.168.200.107

通过这种方式将nginx与keepalived关联上

```
[root@master ~]# vim /etc/keepalived/keepalived.conf
  7 vrrp_script chk_http_port {
  8     script "/opt/check_nginx.sh"
  9     interval 2
 10     weight -15   #脚本结果导致的优先级变更，检测失败（脚本返回非0）则优先级
-15
 11     fall 2
 12     rise 1
 13 }
 25     track_script {
 26         chk_http_port
 27     }
[root@master ~]# systemctl restart keepalived.service
```

模拟故障：当杀掉主的nginx时，nginx会不会漂到从上

```
[root@master ~]# nginx
[root@master ~]# netstat -lnpt
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
13480/nginx: master
[root@master ~]# vim /usr/local/nginx/conf/nginx.conf
(制造一点小报错从而关掉nginx进程，随便注释一行)
[root@master ~]# nginx -t
[root@master ~]# killall -9 nginx
[root@master ~]# systemctl status keepalived.service （活跃也没事）
[root@master ~]# ip a
[root@backup ~]# ip a     (VIP飘到这里了)
```

修复nginx：

```
[root@master ~]# vim /usr/local/nginx/conf/nginx.conf
[root@master ~]# nginx -t
[root@master ~]# ip a          （VIP又回来了）
```

非抢占：（主的在抢，所以在配置时，只给主的配置就可以，备的就不用配）

在非抢占模式中，没有主（master）只有backup

谁在工作主要看优先级来说话

```
[root@master ~]# vim /etc/keepalived/keepalived.conf
 16     state BACKUP
 21     nopreempt
[root@backup ~]# systemctl stop keepalived.service
[root@master ~]# systemctl restart keepalived.service
[root@master ~]# ip a    （得到VIP）
[root@backup ~]# systemctl restart keepalived.service
[root@backup ~]# ip a       （非抢占模式，发现VIP依然在107上）
```

——

```
[root@master ~]# systemctl stop keepalived.service
[root@backup ~]# ip a           （得到VIP）
[root@master ~]# systemctl start keepalived.service
[root@master ~]# ip a             （依然没有，还是在备的上面，非抢占模式）
```

除非备的出故障也停止服务了，VIP才会再次飘回主的上面

小提问：（面试）

keepalived的底层协议是什么？

这个协议的英文及中文是什么？

keepalived的两个功能两个模式是什么？

在配置keepalived的主从实验中，主和备有哪些是不同的？---状态 优先级

nginx是靠什么实现负载均衡的功能？upstream proxy_pass

在nginx+keepalived的结构中，谁在负责节点的健康状态检查？

nginx（节点健康检查功能，检查后端的Apache服务）+ keepalived（VIP 自动切换）

```
    keepalive_timeout  65;
        upstream webserver {
                server 192.168.200.109:80 weight=1;
                server 192.168.200.110:80 weight=1;
        }
```

LVS+keepalived的结构中，谁在负责节点的健康状态检查？

LVS+keepalived（故障切换 健康检查）--没LVS的事儿

vim /etc/keepalived/keepalived.conf

1.这在健康检查

```
 TCP_CHECK {
            connect_port 80
            connect_timeout 3
            delay_before_retry 3
        }
```

2.这在故障切换

```
vrrp_instance VI_1 {
    state MASTER
    interface ens32
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 123456
    }
    virtual_ipaddress {
        192.168.200.254
    }
}
```

实验前准备，将调度器撤掉：（192.168.200.107 192.168.200.108）

```
[root@master ~]# systemctl stop keepalived.service
[root@master ~]# killall -9 nginx
[root@master ~]# netstat -lnpt
```

从的同上

基于 Haproxy 构建负载均衡集群

四层转发

1、HAPROXY简介

HAProxy提供高可用性、负载均衡以及基于TCP和HTTP应用的代理，支持虚拟主机，它是免费、快速并且可靠的一种负载均衡解决方案

2、HAProxy的特点

1. HAProxy支持虚拟主机。
2. HAProxy的优点能够补充Nginx的一些缺点，比如**支持Session的保持，Cookie的引导；同时支持通过获取指定的url来检测后端服务器的状态。**
3. HAProxy跟LVS类似，本身就只是一款负载均衡软件；单纯从效率上来讲HAProxy会比Nginx有更出色的负载均衡速度，在并发处理上也是优于Nginx的。
4. HAProxy**支持TCP协议的负载均衡转发**，可以对MySQL读进行负载均衡，对后端的MySQL节点进行检测和负载均衡，可以用LVS+Keepalived对MySQL主从做负载均衡。
5. HAProxy负载均衡策略非常多，HAProxy的负载均衡**算法**现在具体有如下8种：

① roundrobin，表示简单的轮询，这个不多说，这个是负载均衡基本都具备的；

② static-rr，表示根据权重，建议关注；

③ leastconn，表示最少连接者先处理，建议关注；

④ source，表示根据请求源IP，这个跟Nginx的IP_hash机制类似，我们用其作为解决session问题的一种方法，建议关注；

⑤ ri，表示根据请求的URI；

⑥ rl_param，表示根据请求的URl参数’balance url_param’ requires an URL parameter name；

⑦ hdr(name)，表示根据HTTP请求头来锁定每一次HTTP请求；

⑧ rdp-cookie(name)，表示根据据cookie(name)来锁定并哈希每一次TCP请求。

nginx的算法：五种

LVS的算法：10种

3、haproxy配置文件的五部分

1. global：参数是进程级的，通常是和操作系统相关。这些参数一般只设置一次，如果配置无误，就不需要再次进行修改
2. defaults：配置默认参数，这些参数可以被用到frontend，backend，Listen组件
3. frontend：接收请求的前端虚拟节点，Frontend可以更加规则直接指定具体使用后端的backend
4. backend：后端服务集群的配置，是真实服务器，一个Backend对应一个或者多个实体服务器，类似于nginx中的upstream
5. Listen Fronted和backend的组合体

4、案例环境

主机　　操作系统　　IP地址　　主要的软件

Haproxy　　CentOS6.6 x86_64　　192.168.200.101　　haproxy-1.4.24.tar.gz

Apache1　　CentOS6.6 x86_64　　192.168.200.109　　httpd-2.4.6-80.el7.centos.x86_64

Apache2　　CentOS6.6 x86_64　　192.168.200.110　　httpd-2.4.6-80.el7.centos.x86_64

后面两个节点，nginx，Apache，Tomcat都可以，这里我用Apache

5、安装配置Haproxy

```
[root@localhost ~]# yum -y install gcc gcc-c++ make pcre-devel bzip2-devel
[root@localhost ~]# yum -y install haproxy
[root@localhost ~]# ll /etc/haproxy/haproxy.cfg
```

修改主配置文件：192.168.200.107

```
[root@localhost ~]# vim /etc/haproxy/haproxy.cfg
 22     server  inst1 192.168.200.109:80 check inter 2000 fall 3
 23     server  inst1 192.168.200.110:80 check inter 2000 fall 3
[root@localhost ~]# systemctl start haproxy.service
[root@localhost ~]# netstat -lnpt | grep haproxy
tcp        0      0 0.0.0.0:8000            0.0.0.0:*               LISTEN
46987/haproxy
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
46987/haproxy
```

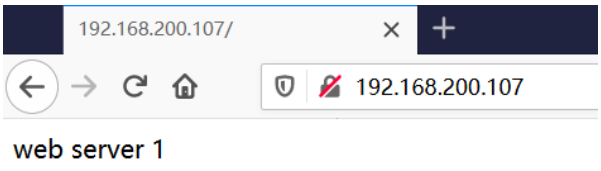
6、Haproxy 日志

```
[root@localhost ~]# vim /etc/haproxy/haproxy.cfg
  2     log /dev/log    local0 info
  3     log /dev/log    local0 notice
[root@localhost ~]# vim /etc/rsyslog.d/haproxy.conf
if ($programname == 'haproxy' and $syslogserverity-text == 'info') then
-/var/log/haproxy/haproxy-info.log
&~
if ($programname == 'haproxy' and $syslogserverity-text == 'notice') then
-/var/log/haproxy/haproxy-notice.log
```

&~

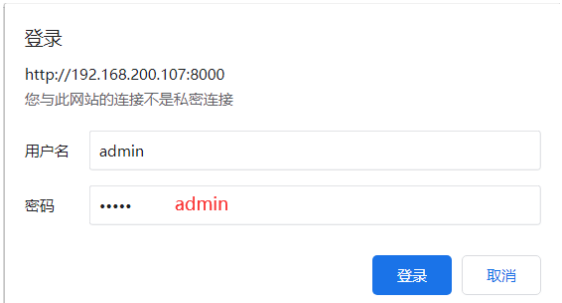
[root@localhost ~]# systemctl restart rsyslog

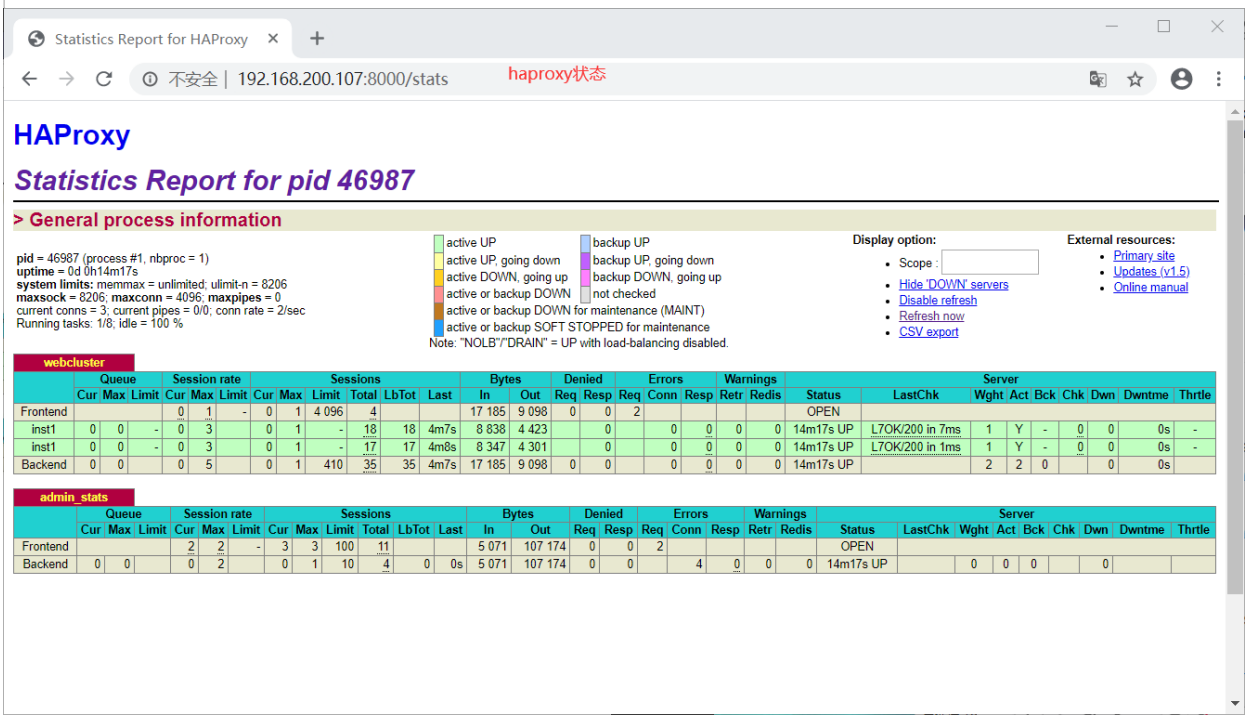
测试：



[root@localhost ~]# mkdir /var/log/haproxy

[root@localhost ~]# ll /var/log/haproxy





192.168.200.108：

这是从107传到108的：

[root@localhost ~]# scp /etc/haproxy/haproxy.cfg 192.168.200.108:/etc/haproxy

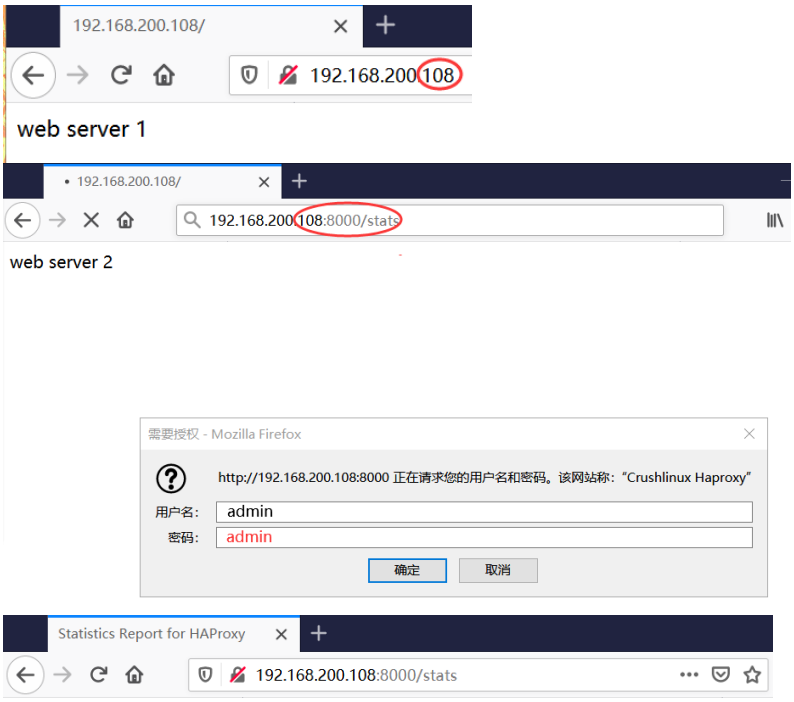
[root@localhost ~]# systemctl start haproxy

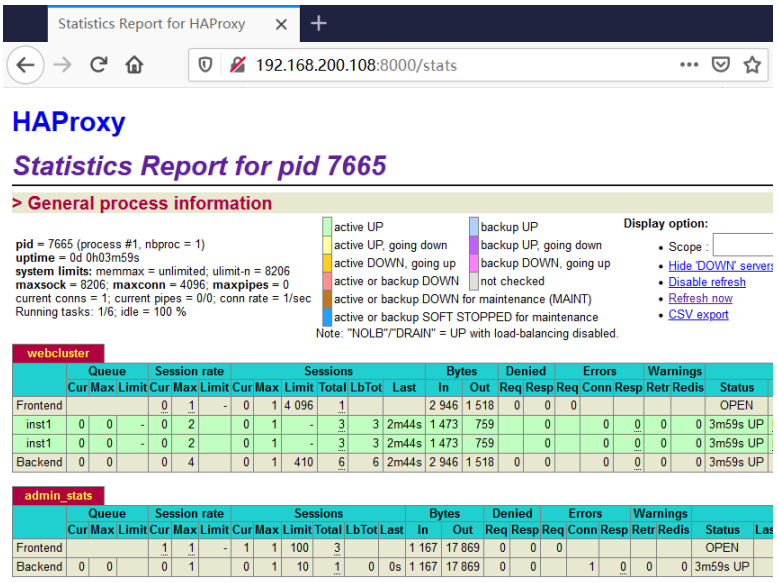
[root@localhost ~]# scp /etc/rsyslog.d/haproxy.conf

192.168.200.108:/etc/rsyslog.d/haproxy.conf

[root@localhost ~]# mkdir /var/log/haproxy

[root@localhost ~]# systemctl restart rsyslog





haproxy+keepalived 制作高可用：

```
[root@master ~]# vim /etc/keepalived/keepalived.conf
  8      script "/opt/check_haproxy.sh"
[root@master ~]# cp /opt/check_nginx.sh /opt/check_haproxy.sh
[root@master ~]# vim /opt/check_haproxy.sh
#!/bin/bash
nnum=$( ps -C haproxy --no-header | wc -l )
if [ $nnum -eq 0 ]
then
    /usr/local/nginx/sbin/nginx
    sleep 2
    nnum=$( ps -C haproxy --no-header | wc -l )
    if [ $nnum -eq 0 ]
    then
```

```
        systemctl stop keepalived
    fi
fi
[root@master ~]# systemctl start keepalived.service
[root@master ~]# ip a
[root@master ~]# scp /opt/check_haproxy.sh 192.168.200.108:/opt/
```

192.168.200.108：

```
[root@localhost ~]# vim /etc/keepalived/keepalived.conf
  7     script "/opt/check_haproxy.sh"
[root@localhost ~]# systemctl start keepalived.service
[root@localhost ~]# tail -f /var/log/messages
```

测试：

模拟故障：

```
[root@master ~]# systemctl stop keepalived.service
[root@localhost ~]# ip a
    inet 192.168.200.254/32 scope global ens32
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe88:4ee1/64 scope link
       valid_lft forever preferred_lft forever
[root@localhost ~]# tail -40 /var/log/messages
Apr 22 19:25:39 localhost Keepalived_vrrp[8019]: VRRP_Instance(VI_1) Transition
to MASTER STATE
Apr 22 19:25:40 localhost Keepalived_vrrp[8019]: VRRP_Instance(VI_1) Entering
MASTER STATE
```

非抢占模式：

修复192.168.200.107

```
[root@master ~]# systemctl start keepalived.service
[root@master ~]# ip a        （依然没有VIP）
```

```
[root@localhost ~]# tail -f /var/log/messages
Apr 22 19:28:18 localhost haproxy[7665]: 192.168.200.128:1134
[22/Apr/2020:19:28:18.842] webcluster webcluster/inst1 151/0/0/1/152 200 253 - -
---- 1/1/0/1/0 0/0 "GET / HTTP/1.1"
```

这里的日志格式与[root@localhost ~]# vim /etc/haproxy/haproxy.cfg里面的定义有关

```
[root@localhost ~]# vim /etc/haproxy/haproxy.cfg
    server  inst1 192.168.200.109:80 check inter 2000 fall 3
    server  inst1 192.168.200.110:80 check inter 2000 fall 3
```

MySQL数据库系统部署及SQL语句基础

MySQL是一个真正的多线程，多用户的关系型数据库(RDBMS) 服务管理软件，

（二维表格的管理技术，跟生活中用的Excel表格差不多，表格与表格之间有相互的关联性，可以相互查询）

特点：查询速度快，高性能，高可靠和易于使用

成为服务器领域中最受欢迎的开源关系型数据库系统

在2008之前，MySQL项目由MySQL AB公司进行开发,发布与支持

SUN在2008年以10亿美元收购了MySQL AB

2009年4月份Oracle甲骨文宣布将以74亿美元并购SUN公司

目前MySQL项目由Oracle公司负责运营和维护

RDBMS:关系型数据库　　relationship　database manage system

DBMS:数据库管理系统　　database manage system

MySQL与MariaDB:

MariaDB数据库管理系统是MySQL的一个分支,主要由开源社区在维护，采用GPL授权许可开发

这个分支的原因之一:

甲骨文公司收购了MySQL后，有将MySQL闭源的潜在风险

因此社区采用分支的方式来避开这个风险

MariaDB的目的是完全兼容MySQL

包括API和命令行，使之能轻松成为MySQL的代替品

MariaDB由MySQL的创始人麦克尔维德纽斯主导开发，他早前曾以10亿美元的价格，
将自己创建的公司MySQL AB卖给了SUN,此后，随着SUN被甲骨文收购
MySQL 的所有权也落入Oracle的手中

MariaDB直到5.5版本，均依照MySQL的版本发展
因此，使用MariaDB5.5的人会从MySQL5.5
中了解到MariaDB的所有功能
从2012年11月12日起发布的10.0.0版开始，不再依照MySQL的版号
10.0.x版以5.5版为基础，加上移植自MySQL 5.6版的功能和自行开发的新功能
CentOS7系列Linux操作系统中采用MariaDB替代了MySQL数据库产品

关系型数据库管理系统RDBMS:
商业: Oracle (Oracle->Oracle Enterprise Linux), DB2 (IBM->AIX) 、SQL Server (微软)、
Sybase, Infomix
开源: MySQL (MariaDB) , PostgreSQL (Apple) , pgsql, EnterpriseDB

非关系型数据库NoSQL （not only SQL）:
web2.0 ---交互式
MongoDB Redis Memcached HBase InfluxDB

MySQL版本:

- Community Edtion社区版（**CE**），免费,由社区人员维护，测试及更新 **---使用较多**
- Enterprise Edtion企业版（**EE**）,收费，MySQL官方维护团队人员维护,测试及更新

安装MySQL

1)安装Mysq|数据库 ---检查防止冲突
[root@localhost ~] rpm -q mysq| mysql-server mariadb mariadb-server
未安装软件包mysql
未安装软件包mysql-server
未安装软件包mariadb
未安装软件包mariadb-server
安装ncurses-devel依赖包

[root@localhost ~]# yum -y install ncurses-devel

[root@localhost ~]# rpm -9 ncurses-devel

ncurses-devel-5.9-14.20130511.el7_ 4.x86_ 64

建议采用yum安装方式cmake

[root@localhost ~]# yum -y install cmake

[root@localhost ~]# rpm -q cmake

cmake-2.8.12.2-2.el7.x86_ 64

创建运行用户

[root@localhost ~]# useradd -M -s /sbin/nologin mysql

解包,配置,编译，安装

[root@localhost ~]# tar xf mysql-5.7.24.tar.gz -C /usr/src/

[root@localhost ~]# cd /usr/src/mysql-5.7.24/

[root@localhost mysql-5.7 .24]# cmake -DCMAKE_ _INSTALL_ PREFIX=/usrlocal/mysql

-

DDEFAULT_ CHARSET=utf8 -DDEFAULT_ COLLATION=utf8_ general _ci -

DWITH_ EXTRA_ _CHARSETS=all -DSYSCONFDIR=/etc && make -j 2 && make install

- -DCMAKE_ INSTALL_ PREFIX-=/sr/local/mysql // 数据库程序安装目录
- -DDEFAULT_ CHARSET=utf8 // 指定字符集编码
- -DDEFAULT_ COLLATION=utf8_ general_ ci //默认的字符集校对规则,
- utf8_ general_ ci 适用于utf-8字符集的通用规则
- -DWITH_ EXTRA_ CHARSETS=all //指定额外支持的字符集编码
- -DSYSCONFDIR=/etc //指定配置文件存放目录

解决办法是:

1.在/usrlocal下创建一个名为boost的文件夹

[root@localhost ~]# mkdir /usr/ocal/boost

2.进入目录并下载boost

[root@localhost ~]# cd /usrlocal/boost

[root@localhost boot]# wget

hts:/ocrcefofrge.netpiccstotfiles/boost/1.59./boost 159_ _0.tar.gz

3.解压boost

[root@localhost boost]# tar xf boost 1 59 0.tar.gz

4.继续cmake,添加，上红色部分

```
[root@localhost boost]# cd /usr/src/mysql-5.7.24/
[root@localhost mysql-5.7.24]# cmake -DCMAKE_ INSTALL. _PREFIX=/usrlocal/mysq| -
DDEFAULT_ _CHARSET=ut8 -DDEFAULT_ COLL ATION=utf8_ general_ ci -
DWITH_ EXTRA_ _CHARSETS=all -DSYSCONFDIR=/etc -DWITH_ BOOST=/usr/local/boost &&
make - 2 && make install
------------------------------------------------------------------------------
------
```

2)安装后的调整

对数据库目录进行权限设置

```
[root@localhost ~]# chown -R mysql:mysql /usrlocal/mysqU
[root@localhost ~]# cd /usrlocal/mysql
```

建立配置文件(CentOS7系统默认支持MariaDB数据库,系统默认的/etc/my.cnf配置文件是MariaDB的配置文件)

```
[root@localhost mysq|]# vim /etc/my.cnf
[mysqld]
datadir=/usr/local/mysql/data|
socket-/tmp/mysql.sockI
[mysqld_ safe]
log-or/us/local/mysql/data/mysql.log
pi-ile=/us/local/mysql/data/mysql.pid
```

3)初始化数据库

```
[root@localhost mysq]# ./bin/mysqld --user=mysql -basedir=/usrloca/mysql --
datadir=/usrlocal/mysql/data -initialize
2018-12-08T01:51:39.798903Z 1 [Note] A temporary password is generated for
root@localhost:
TVC:Rm1ZlxtG
```

- -basedir=/usr/local/mysql/ //指定安装目录(产品目录)
- -datadir=/us/ocal/mysql/data //指定数据目录
- --user=mysq| //指定用户身份

4)设置环境变量

```
[root@localhost mysq|-5.7 .24]# echo "PATH=$PATH:/usrlocal/mysq/bin" >>
/etc/profile
```

5)添加系统服务

添加MySQL为系统服务,以便通过systemctl命 令进行管理

方法一:

```
root@localhost mysq|l-5.7 .24]# cp support-iles/mysql.server
/us/local/mysq/bin/mysqld.sh
[root@localhost mysql-5.7 .24]# chmbd +x /usrlocal/mysq/bin/mysqld.sh
[root@localhost ~]# vim /usr/ib/systemd/system/mysqld.service
[Unit]
Description=MySQL Server
After=network.target
[Service]
User=mysql #指定程序运行的用户账户
Group=mysql #指定程序运行的组账户
Type=forking
PIDFile=/us/rloca/mysq/data/mysql.pid #指定PID文件的位置，默认为“主机名pid"
ExecStart=/usrlocal/mysq/bin/mysqld.sh start
ExecStop-/us/local/mysq/bin/mysqld.sh stop
[Install]
WantedBy=mutil-user.target
```

方法二:

```
[root@localhost ~]# cp /usr/src/mysql-5.24/support-files/mysql.server
/et/init.d/mysqld
[root@localhost ~]# chmod +x /etc/init./mysqld
[root@localhost ~]# systemctl daemon-reload
[root@localhost ~]# systemctl start mysqld
[root@localhost ~]# chkconfig mysqld on I
[root@localhost ~]# systemctl status mysqld
[root@localhost ~]# netstat -Inpt | grep mysqld
```

基于通用二进制方式安装MySQL-5.7.24

这种安装方式：相当于已经编译好的，已经过了cmake make ,非常节省时间

192.168.200.107：

```
[root@localhost ~]# tar xf mysql-5.7.24-linux-glibc2.12-x86_64.tar.gz -C
/usr/src/
[root@localhost ~]# rpm -q libaio
libaio-0.3.109-13.e17.x86_64
[root@localhost ~]# mv /usr/src/mysql-5.7.24-linux-glibc2.12-x86_64/
/usr/local/mysql
```

```
[root@localhost ~]# cd /usr/local/mysql/
```

创建程序用户：

```
[root@localhost ~]# useradd -M -s /sbin/nologin mysql
[root@localhost ~]# chown -R mysql:mysql /usr/local/mysql/
```

递归修改用户和组

初始化：

```
[root@localhost ~]# /usr/local/mysql/bin/mysqld --user=mysql --
basedir=/usr/local/mysql --datadir=/usr/local/mysql/data --initialize
```

修改文件：

```
[root@localhost ~]# vim /etc/my.cnf
  2 datadir=/usr/local/mysql/data
  3 socket=/tmp/mysql.sock
 12 log-error=/usr/local/mysql/data/mysql.log
 13 pid-file=/usr/local/mysql/data/mysql.pid
[root@localhost ~]# cp /usr/local/mysql/support-files/mysql.server
/etc/init.d/mysqld
[root@localhost ~]# chmod +x /etc/init.d/mysqld
[root@localhost ~]# chkconfig --add mysqld
[root@localhost ~]# systemctl start mysqld
[root@localhost ~]# netstat -lnpt | grep 3306
tcp6       0      0 :::3306                 :::*                    LISTEN
14662/mysqld
[root@localhost ~]# ln -s /usr/local/mysql/bin/* /usr/local/bin/
```

替代了变量，做链接的目的就是为了找到位置就行

```
[root@localhost ~]# mysqladmin -uroot -p'QSAGl&4rOhwd' password '123456'
```

登录

```
[root@localhost ~]# mysql -uroot -p123456
```

MySQL客户端命令

MySQL数据库系统也是典型的C/S (客户端/服务器)架构的应用

C/S客户端/服务器　　B/S 浏览器/服务器

连接时需要专用的客户端工具，Linux 下通过mysq|命令工具

(如果是通过rpm格式安装软件需要安装mysql软件包)

连接并登录到MySQL操作环境

mysql

-u指定用户名

-p指定密码(选项和密码之间不能有空格)

-h指定主机

-P指定端口

-S指定Socket文件

-e指定SQL命令语句(非交互模式)

[root@localhost ~]# mysql -uroot -p123456 ; history -c

登录时会有明文密码显示，所以要加上history -c

显示当前连接用户:

```
mysql> select user();
+----------------+
| user()         |
+----------------+
| root@localhost |          #这里的root跟系统用户没有任何关系，只是名字一样而已
+----------------+
1 row in set (0.00 sec)
```

查看数据库服务的基本信息:

```
mysql> status;
--------------
mysql  Ver 14.14 Distrib 5.7.24, for linux-glibc2.12 (x86_64) using  EditLine
wrapper
Connection id:          5
Current database:
Current user:           root@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         5.7.24 MySQL Community Server (GPL)
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    utf8
Conn.  characterset:    utf8
```

```
UNIX socket:            /tmp/mysql.sock
Uptime:                 15 min 20 sec
```

下面是一些跟性能相关的信息：

```
Threads: 1  Questions: 14  Slow queries: 0  Opens: 106  Flush tables: 1  Open
tables: 99  Queries per second a
vg: 0.015
--------------
```

退出mysq|操作环境

```
mysq|l> exit
```

或者

```
mysql> \q
```

Ctrl+C取消命令执行

设置数据库用户的初始密码:

```
[root@localhost ~]# mysqladmin -u root password '123456'
```

后期修改数据库用户的密码:

```
mysqladmin -u root -p'旧的密码 password '新的空码'
[root@localhost ~]# mysqladmin -u root -P TvC:Rm1Z1xtG' password '123456'
```

MySQL破解root管理员密码

```
[root@localhost ~]# systemctl stop mysqld            //业务低峰期
[root@localhost ~]# netstat -Inpt I grep 3306
```

跳过加载授权表过程:

```
[root@localhost ~]# mysqld_safe --skip-grant-tables &
[root@localhost ~]#  netstat -lnpt | grep 3306
[root@localhost ~]# mysql
```

注意：这种登录方式很危险的

```
mysql> select user,authentication_string from mysql.user;
+---------------+-------------------------------------------+
| user          | authentication_string                     |
+---------------+-------------------------------------------+
| root          | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
| mysql.session | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| mysql.sys     | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
+---------------+-------------------------------------------+
```

```
3 rows in set (0.00 sec)
```

使用语句修改密码：

```
mysql> update mysql.user set authentication_string=password('123123') where
user='root';
mysql> select user,authentication_string from mysql.user;
+---------------+-------------------------------------------+
| user          | authentication_string                     |
+---------------+-------------------------------------------+
| root          | *E56A114692FE0DE073F9A1DD68A00EEB9703F3F1 |
| mysql.session | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| mysql.sys     | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
+---------------+-------------------------------------------+
3 rows in set (0.00 sec)
mysql> flush privileges;
```

因为这种跳过授权表的方式很危险；

处理方法：

```
[root@localhost ~]# ps aux | grep mysql
[root@localhost ~]# kill -9 20831
[root@localhost ~]# kill -9 20979
[root@localhost ~]# systemctl start mysqld
```

先拿旧密码测试一下：

```
[root@localhost ~]# mysql -uroot -p123456
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password:
YES)
[root@localhost ~]# mysql -uroot -p123123 ; history -c
```