KEEPALIVED实现服务高可用

惨绿少年 Linux运维, Web应用, 玩转Linux, 运维基本功 0评论 来源：本站原创 73℃ 字体： 小 中 大

第1章 keepalived服务说明

1.1 keepalived是什么?

Keepalived软件起初是专为LVS负载均衡软件设计的，用来管理并监控LVS集群系统中各个服务节点的状态，后来又加入了可以实现高可用的VRRP功能。因此，Keepalived除了能够管理LVS软件外，还可以作为其他服务（例如：Nginx、Haproxy、MySQL等）的高可用解决方案软件。

Keepalived软件主要是通过VRRP协议实现高可用功能的。VRRP是Virtual Router RedundancyProtocol(虚拟路由器冗余协议）的缩写，VRRP出现的目的就是为了解决静态路由单点故障问题的，它能够保证当个别节点宕机时，整个网络可以不间断地运行。

所以，Keepalived 一方面具有配置管理LVS的功能，同时还具有对LVS下面节点进行健康检查的功能，另一方面也可实现系统网络服务的高可用功能。

keepalived官网http://www.keepalived.org

1.2 keepalived服务的三个重要功能

管理LVS负载均衡软件

实现LVS集群节点的健康检查中

作为系统网络服务的高可用性（failover）

1.3 Keepalived高可用故障切换转移原理

Keepalived高可用服务对之间的故障切换转移，是通过 VRRP (Virtual Router Redundancy Protocol ,虚拟路由器冗余协议）来实现的。

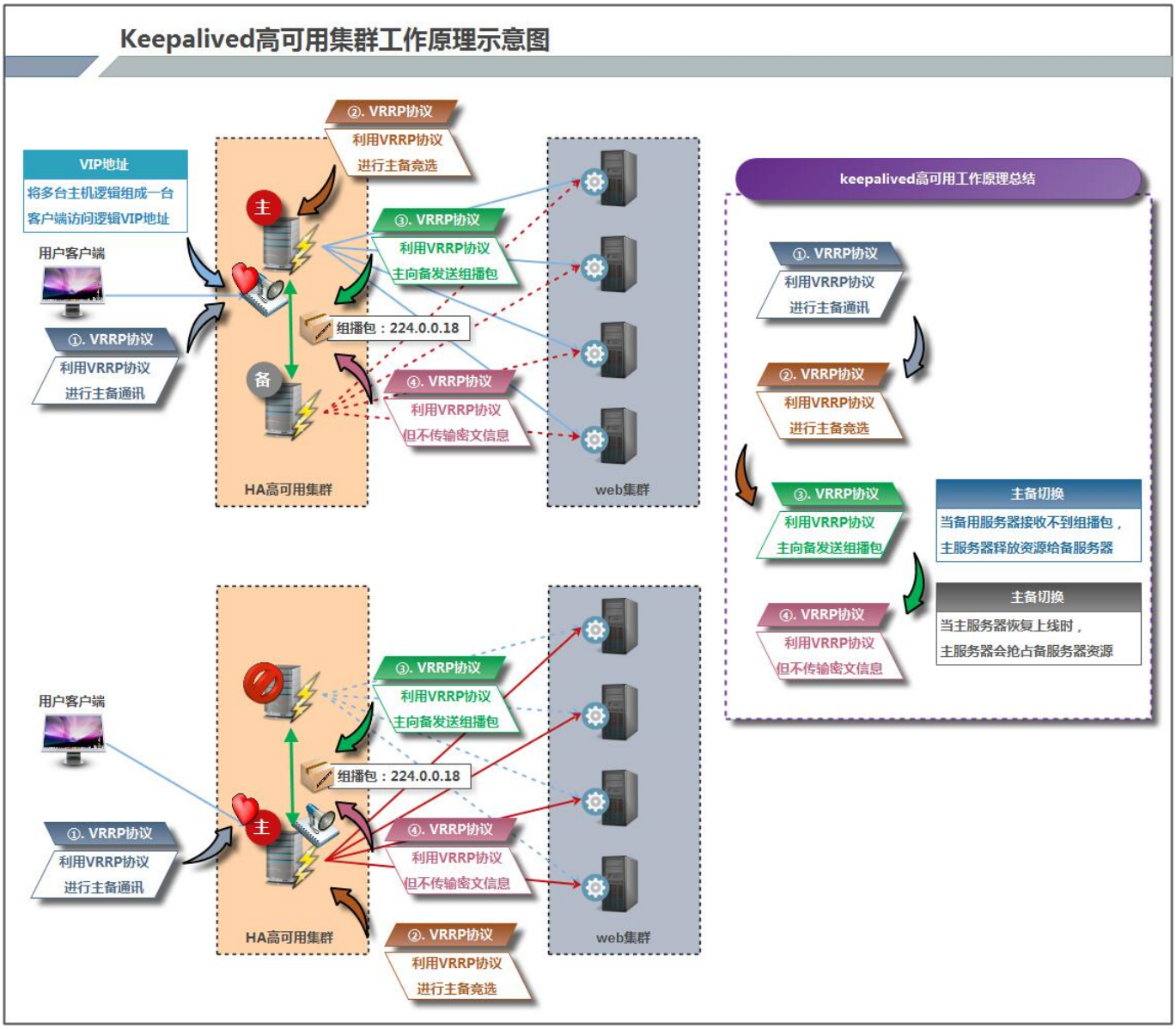
在 Keepalived服务正常工作时，主 Master节点会不断地向备节点发送（多播的方式）心跳消息，用以告诉备Backup节点自己还活看，当主 Master节点发生故障时，就无法发送心跳消息，备节点也就因此无法继续检测到来自主 Master节点的心跳了，于是调用自身的接管程序，接管主Master节点的 IP资源及服务。而当主Master节点恢复时，备Backup节点又会释放主节点故障时自身接管的IP资源及服务，恢复到原来的备用角色。

那么，什么是VRRP呢?

VRRP ,全 称 Virtual Router Redundancy Protocol ,中文名为虚拟路由冗余协议 ， VRRP的出现就是为了解决静态踣由的单点故障问题，VRRP是通过一种竞选机制来将路由的任务交给某台VRRP路由器的。

1.4 keepalived 原理

1.4.1keepalived高可用架构示意图



1.4.2 文字，表述

Keepalived的工作原理：

Keepalived高可用对之间是通过VRRP通信的，因此，我们从 VRRP开始了解起：

1) VRRP,全称 Virtual Router Redundancy Protocol,中文名为虚拟路由冗余协议，VRRP的出现是为了解决静态路由的单点故障。

2) VRRP是通过一种竞选协议机制来将路由任务交给某台 VRRP路由器的。

3) VRRP用 IP多播的方式（默认多播地址（224.0_0.18))实现高可用对之间通信。

4) 工作时主节点发包，备节点接包，当备节点接收不到主节点发的数据包的时候，就启动接管程序接管主节点的开源。备节点可以有多个，通过优先级竞选，但一般 Keepalived系统运维工作中都是一对。

5) VRRP使用了加密协议加密数据，但Keepalived官方目前还是推荐用明文的方式配置认证类型和密码。

介绍完 VRRP,接下来我再介绍一下 Keepalived服务的工作原理：

Keepalived高可用对之间是通过 VRRP进行通信的， VRRP是遑过竞选机制来确定主备的，主的优先级高于备，因此，工作时主会优先获得所有的资源，备节点处于等待状态，当主挂了的时候，备节点就会接管主节点的资源，然后顶替主节点对外提供服务。

在 Keepalived服务对之间，只有作为主的服务器会一直发送 VRRP广播包,告诉备它还活着，此时备不会枪占主，当主不可用时，即备监听不到主发送的广播包时，就会启动相关服务接管资源，保证业务的连续性.接管速度最快可以小于1秒。

第2章 keepalived软件使用

2.1 软件的部署

2.1.1 第一个里程碑 keepalived软件安装

yum install keepalived -y

```
/etc/keepalived
/etc/keepalived/keepalived.conf     #keepalived服务主配置文件
/etc/rc.d/init.d/keepalived         #服务启动脚本
/etc/sysconfig/keepalived
/usr/bin/genhash
/usr/libexec/keepalived
/usr/sbin/keepalived
```

第二个里程碑： 进行默认配置测试

2.1.2 配置文件说明

1-13行表示全局配置

```
global_defs {    #全局配置
   notification_email {   定义报警邮件地址
     acassen@firewall.loc
     failover@firewall.loc
     sysadmin@firewall.loc
   }
   notification_email_from Alexandre.Cassen@firewall.loc  #定义发送邮件的地址
   smtp_server 192.168.200.1   #邮箱服务器
   smtp_connect_timeout 30      #定义超时时间
   router_id LVS_DEVEL         #定义路由标识信息，相同局域网唯一
}
```

15-30行 虚拟ip配置 brrp

```
vrrp_instance VI_1 {   #定义实例
    state MASTER          #状态参数 master/backup 只是说明
    interface eth0        #虚IP地址放置的网卡位置
    virtual_router_id 51 #同一家族要一直，同一个集群id一致
    priority 100          # 优先级决定是主还是备     越大越优先
    advert_int 1         #主备通讯时间间隔
    authentication {      # ↓
        auth_type PASS     #↓
```

```
        auth_pass 1111    #认证
    }                        #↑
    virtual_ipaddress {  #↓
        192.168.200.16    设备之间使用的虚拟ip地址
        192.168.200.17
        192.168.200.18
    }
}
```

配置管理LVS

关于 LVS 详情参考 http://www.cnblogs.com/clsn/p/7920637.html#_label7

2.1.3 最终配置文件

主负载均衡服务器配置

```
[root@lb01 conf]# cat  /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
   router_id lb01
}

vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 150
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        10.0.0.3
    }
}
```

备负载均衡服务器配置

```
[root@lb02 ~]# cat /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
   router_id lb02
}

vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
```

```
        auth_pass 1111
    }
    virtual_ipaddress {
     10.0.0.3
    }
}
```

2.1.4 启动keepalived

```
[root@lb02 ~]# /etc/init.d/keepalived start
Starting keepalived:                                       [  OK  ]
```

2.1.5 【说明】在进行访问测试之前要保证后端的节点都能够单独的访问。

测试连通性. 后端节点

```
[root@lb01 conf]# curl -H host:www.etiantian.org  10.0.0.8
web01 www
[root@lb01 conf]# curl -H host:www.etiantian.org  10.0.0.7
web02 www
[root@lb01 conf]# curl -H host:www.etiantian.org  10.0.0.9
web03 www
[root@lb01 conf]# curl -H host:bbs.etiantian.org  10.0.0.9
web03 bbs
[root@lb01 conf]# curl -H host:bbs.etiantian.org  10.0.0.8
web01 bbs
[root@lb01 conf]# curl -H host:bbs.etiantian.org  10.0.0.7
web02 bbs
```

2.1.6 查看虚拟ip状态

```
[root@lb01 conf]# ip a
1: lo:  mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0:  mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:90:7f:0d brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.5/24 brd 10.0.0.255 scope global eth0
    inet 10.0.0.3/24 scope global secondary eth0:1
    inet6 fe80::20c:29ff:fe90:7f0d/64 scope link
       valid_lft forever preferred_lft forever
3: eth1:  mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:90:7f:17 brd ff:ff:ff:ff:ff:ff
    inet 172.16.1.5/24 brd 172.16.1.255 scope global eth1
    inet6 fe80::20c:29ff:fe90:7f17/64 scope link
       valid_lft forever preferred_lft forever
```

2.1.7 【总结】配置文件修改

Keepalived主备配置文件区别:

01. router_id 信息不一致

02. state 状态描述信息不一致

03. priority 主备竞选优先级数值不一致

2.2 脑裂

在高可用（HA）系统中，当联系2个节点的“心跳线”断开时，本来为一整体、动作协调的HA系统，就分裂成为2个独立的个体。由于相互失去了联系，都以为是对方出了故障。两个节点上的HA软件像“裂脑人”一样，争抢“共享资源”、争起“应用服务”，就会发生严重后果——或者共享资源被瓜分、2边“服务”都起不来了；或者2边“服务”都起来了，但同时读写“共享存储”，导致数据损坏（常见如数据库轮询着的联机日志出错）。

对付HA系统“裂脑”的对策，目前达成共识的的大概有以下几条：

1）添加冗余的心跳线，例如：双线条线（心跳线也HA），尽量减少“裂脑”发生几率；

2）启用磁盘锁。正在服务一方锁住共享磁盘，“裂脑”发生时，让对方完全“抢不走”共享磁盘资源。但使用锁磁盘也会有一个不小的问题，如果占用共享盘的一方不主动“解锁”，另一方就永远得不到共享磁盘。现实中假如服务节点突然死机或崩溃，就不可能执行解锁命令。后备节点也就接管不了共享资源和应用服务。于是有人在HA中设计了“智能”锁。即：正在服务的一方只在发现心跳线全部断开（察觉不到对端）时才启用磁盘锁。平时就不上锁了。

3）设置仲裁机制。例如设置参考IP（如网关IP），当心跳线完全断开时，2个节点都各自ping一下参考IP，不通则表明断点就出在本端。不仅“心跳”、还兼对外“服务”的本端网络链路断了，即使启动（或继续）应用服务也没有用了，那就主动放弃竞争，让能够ping通参考IP的一端去起服务。更保险一些，ping不通参考IP的一方干脆就自我重启，以彻底释放有可能还占用着的那些共享资源。

2.2.1 脑裂产生的原因

一般来说，裂脑的发生，有以下几种原因：

㉂ 高可用服务器对之间心跳线链路发生故障，导致无法正常通信。

因心跳线坏了（包括断了，老化）。

因网卡及相关驱动坏了，ip配置及冲突问题（网卡直连）。

因心跳线间连接的设备故障（网卡及交换机）。

因仲裁的机器出问题（采用仲裁的方案）。

㉂ 高可用服务器上开启了 iptables防火墙阻挡了心跳消息传输。

㉂ 高可用服务器上心跳网卡地址等信息配置不正确，导致发送心跳失败。

㉂ 其他服务配置不当等原因，如心跳方式不同，心跳广播冲突、软件Bug等。

提示： Keepalived配置里同一 VRRP实例如果 virtual_router_id两端参数配置不一致也会导致裂脑问题发生。

2.2.2 常见的解决方案

在实际生产环境中，我们可以从以下几个方面来防止裂脑问题的发生：

同时使用串行电缆和以太网电缆连接，同时用两条心跳线路，这样一条线路坏了，另一个还是好的，依然能传送心跳消息。

当检测到裂脑时强行关闭一个心跳节点（这个功能需特殊设备支持，如Stonith、feyce）。相当于备节点接收不到心跳消患，通过单独的线路发送关机命令关闭主节点的电源。

做好对裂脑的监控报警（如邮件及手机短信等或值班）.在问题发生时人为第一时间介入仲裁，降低损失。例如，百度的监控报警短倍就有上行和下行的区别。报警消息发送到管理员手机上，管理员可以通过手机回复对应数字或简单的字符串操作返回给服务器.让服务器根据指令自动处理相应故障，这样解决故障的时间更短.

当然，在实施高可用方案时，要根据业务实际需求确定是否能容忍这样的损失。对于一般的网站常规业务.这个损失是可容忍的。

2.3 如何进行脑裂情况监控

2.3.1 在什么服务器上进行监控?

在备服务器上进行监控，可以使用zabbix监控，参考http://www.cnblogs.com/clsn/p/7885990.html

2.3.2 监控什么信息?

备上面出现*vip*情况：

1）脑裂情况出现

2）正常主备切换也会出现

2.3.3 编写监控脑裂脚本

```
[root@lb02 scripts]# vim check_keepalived.sh
#!/bin/bash

while true
do
if [ `ip a show eth0 |grep 10.0.0.3|wc -l` -ne 0 ]
then
    echo "keepalived is error!"
else
    echo "keepalived is OK !"
fi
done
```

编写完脚本后要给脚本赋予执行权限

2.3.4 测试 确保两台负载均衡能够正常负载

```
[root@lb01 ~]# curl -H Host:www.etiantian.org 10.0.0.5
web01 www
```

```
[root@lb01 ~]# curl -H Host:www.etiantian.org 10.0.0.6
web01 www
[root@lb01 ~]# curl -H Host:bbs.etiantian.org 10.0.0.6
web02 bbs
 [root@lb01 ~]# curl -H Host:www.etiantian.org 10.0.0.5
web03 www
```

2.4 排错过程

1）利用负载均衡服务器，在服务器上curl所有的节点信息（web服务器配置有问题）

2）curl 负载均衡服务器地址，可以实现负载均衡

3）windows上绑定虚拟IP，浏览器上进行测试

keepalived日志文件位置 /var/log/messages

2.5 更改nginx反向代理配置 只监听vip地址

修改nginx监听参数 listen 10.0.0.3:80;

修改内核参数，实现监听本地不存在的ip

```
echo 'net.ipv4.ip_nonlocal_bind = 1' >>/etc/sysctl.conf
sysctl -p

[root@lb02 conf]# cat /proc/sys/net/ipv4/ip_nonlocal_bind
```

2.6 让keepalived监控nginx

```
ps -ef |grep nginx |grep -v grep |wc -l
```

编写执行脚本

```
#!/bin/bash

while true
do
if [ `ps -ef |grep nginx |grep -v grep |wc -l` -lt 2 ]
then
   /etc/init.d/keepalived stop
   exit
fi
done
```

注意脚本的授权

```
[root@lb01 scripts]# chmod +x check_www.sh
```

2.6.1 使用keepalived的监控脚本

说明 执行的脚本名称尽量不要和服务名称相同或相似

```
[root@lb01 scripts]# cat /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
   router_id lb01
}

vrrp_script check {     #定义脚本
   script "“/server/scripts/check_web.sh"  --- 表示将一个脚本信息赋值给变量check_web
   interval 2    --- 执行监控脚本的间隔时间
   weight 2  ---利用权重值和优先级进行运算，从而降低主服务优先级使之变为备服务器（建议先忽略）
}

vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 150
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        10.0.0.3/24 dev eth0 label eth0:1
    }
    track_script {     #调用脚本
       check
    }
}
```

2.7 多实例的配置

2.7.1 lb01的keepalived配置文件

```
[root@lb01 scripts]# cat  /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
   router_id lb01
}

vrrp_script check {
   script "/server/scripts/check_www.sh"
   interval 2
   weight 2
}

vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 150
    advert_int 1
    authentication {
        auth_type PASS
```

```
        auth_pass 1111
    }
    virtual_ipaddress {
        10.0.0.3/24 dev eth0 label eth0:1
    }
    track_script {
       check
    }
}
vrrp_instance VI_2 {
    state BACKUP
    interface eth0
    virtual_router_id 52
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        10.0.0.4/24 dev eth0 label eth0:2
    }
}
```

2.7.2 修改lb02的keepalived配置文件

```
[root@lb02 conf]# cat /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
   router_id lb02
}

vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
     10.0.0.3 dev eth0 label eth0:1
    }
}
vrrp_instance VI_2 {
    state MASTER
    interface eth0
    virtual_router_id 52
    priority 150
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
```

```
    virtual_ipaddress {
     10.0.0.4 dev eth0 label eth0:2
    }
}
```

修改nginx配置文件，让bbs 与www分别监听不同的ip地址

```
worker_processes  1;
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;
    upstream server_pools {
      server 10.0.0.7:80;
      server 10.0.0.8:80;
      server 10.0.0.9:80;
    }
    server {
        listen 10.0.0.3:80;
        server_name www.etiantian.org;
        location / {
            proxy_pass http://server_pools;
            proxy_set_header Host $host;
            proxy_set_header X-Forwarded-For $remote_addr;
        }
    }
    server {
        listen 10.0.0.4:80;
        server_name bbs.etiantian.org;
        location / {
            proxy_pass http://server_pools;
            proxy_set_header Host $host;
            proxy_set_header X-Forwarded-For $remote_addr;
        }
    }
}
```

lb01

```
[root@lb01 scripts]# netstat -lntup |grep nginx
tcp        0      0 10.0.0.3:80                 0.0.0.0:*                   LISTEN      84907/ng
tcp        0      0 10.0.0.4:80                 0.0.0.0:*                   LISTEN      84907/ng
```

lb02

```
[root@lb02 conf]# netstat -lntup |grep nginx
tcp        0      0 10.0.0.3:80                 0.0.0.0:*                   LISTEN      12258/ng
tcp        0      0 10.0.0.4:80                 0.0.0.0:*                   LISTEN      12258/ng
```

2.8 keepalived双主模式示意图