KEEPALIVED 权威指南

Keepalived: The Definitive Guide

作者： FinalBSD(Kevin Kuang) 二〇〇九年三月
www.sanotes.net

Kevin Kuang 拥有版权 © 2009 以及本书所有的发行版本。保留所有权利。

这份文档是免费的；在自由软件组织颁布的 GNU 通用出版许可证的条款下，你可以再版或者修改它。许可证可以是第二版，或者任何后继版本（随你意）。

目录

第一章 VRRP

Keepalived是VRRP的完美实现，因此在介绍Keepalived之前，我们有必要先了解VRRP的原理。了解VRRP最好的文档莫过于VRRP的RFC文档[1]。

1.1 VRRP协议简介

在现实的网络环境中(比如Internet)，两台需要通信的主机(end-host)大多数情况下并没有直接的物理连接。对于这样的情况，它们之间的路由怎么选择？主机如何选定到达目的主机的下一跳路由，这是一个问题。通常的解决办法有两种：

- 在主机上使用动态路由协议(比如RIP,OSPF等)
- 在主机上配置静态路由

很明显，在主机上配置动态路由协议是非常不切实际的，因为管理、维护成本以及是否支持等诸多问题。那么配置静态路由就变得十分的流行。实际上，这种方式我们至今一直在用。但是，路由器(或者说默认网关default gateway)却经常成为单点。就算配置了多个静态路由，却因为必须重启网络才能生效而变得不实用。

VRRP的目的就是为了解决静态路由单点故障问题！

VRRP通过一种竞选(election)协议来*动态的*将路由任务交给LAN中虚拟路由器中的某台VRRP路由器。这里看起来很绕，因为有两个关键名词:虚拟路由器和VRRP路由器。

VRRP路由器

VRRP路由器就是一台路由器，只不过上面运行了VRRPD这样的程序来实现VRRP协议而已，这是物理的路由器。一台VRRP路由器可以位于多个虚拟路由器。

[1]是RFC 3768而不是2338，如果你喜欢古董，就看2338吧

VRRP虚拟路由器

所谓虚拟，就是说并不是实际存在的，是一个逻辑而不是物理的路由器。虚拟路由器通常由多台(物理的)VRRP路由器通过某种方式组成，就好比这些物理的路由器都丢到一个池(pool)里面去，整个pool对外看起来就象是一台路由器，但其实内部有多台。虚拟路由器的标识称为VRID。

MASTER和BACKUP

在一个VRRP虚拟路由器中，有多台物理的VRRP路由器，但是这多台物理的机器并不同时工作[2]，而是由一台称为MASTER的负责路由工作，其他的都是BACKUP，MASTER并非一成不变，VRRP协议让每个VRRP路由器参与竞选，最终获胜的就是MASTER。MASTER有一些特权[3]，比如拥有虚拟路由器的IP地址，我们的主机就是用这个IP地址作为静态路由的。拥有特权的MASTER要负责转发发送给网关地址的包和响应ARP请求。

1.2 工作机制

VRRP通过竞选协议来实现虚拟路由器的功能，所有的协议报文都是通过IP多播(multicast)包（多播地址224.0.0.18）形式发送的。虚拟路由器由VRID(范围0–255)和一组IP地址组成，对外表现为一个周知的MAC地址：00-00-5E-00-01-{VRID} [4]。所以，在一个虚拟路由器中，不管谁是MASTER，对外都是相同的MAC和IP(称之为VIP)。客户端主机并不需要因为MASTER的改变而修改自己的路由配置，对他们来说，这种主从的切换是透明的。

在一个虚拟路由器中，只有作为MASTER的VRRP路由器会一直发送VRRP广告包(VRRP Advertisement message)，BACKUP不会抢占MASTER，除非它的优先级(priority)更高。当MASTER不可用时（BACKUP收不到广告包），多台BACKUP中优先级最高的这台会被抢占为MASTER。[5]。这种抢占是非常快速的(<1s)，以保证服务的连续性。

出于安全性考虑，VRRP包使用了加密协议进行加密。

[2] 尽管这看起来很浪费
[3] 老大总是有特权的
[4] 这就是为什么后面的配置virtual_router_id为什么只能是0...255
[5] 这也说明了为什么需要state，有需要priority这样的配置

第二章 KEEPALIVED

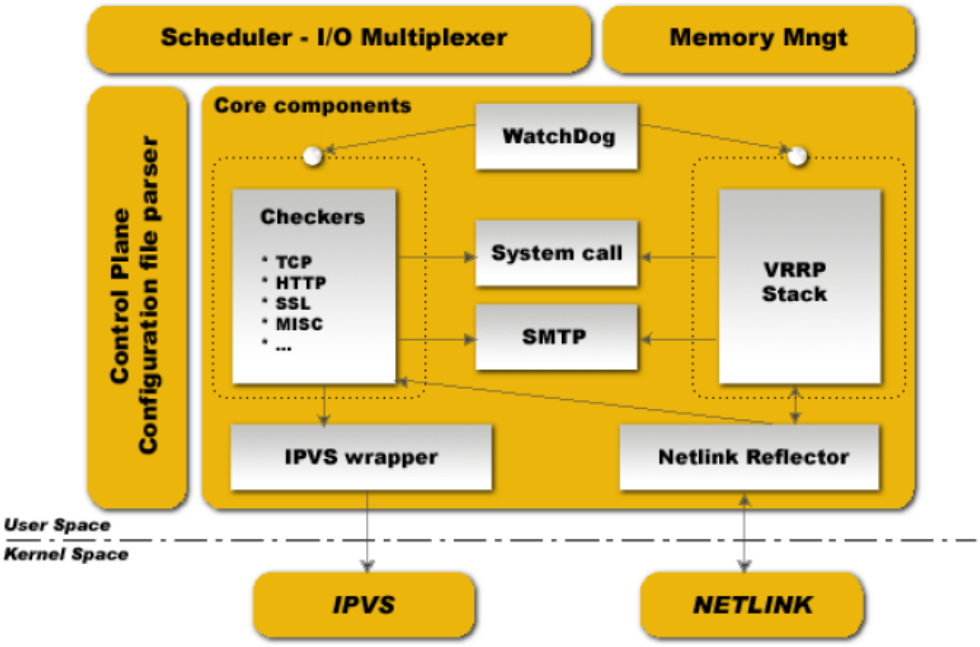
Keepalived的设计和实现虽然简单，但是配置也有不少，本章主要就阐述这些。

2.1 Keepalived的设计和实现

Keepalived是一个高度模块化设计的软件，源代码的结构似乎也很容易看出这一点，里面只有

```
check  core  libipfwc  libipvs-2.4  libipvs-2.6 vrrp
```

这么一些目录。



core keepalived的核心程序，比如全局配置的解析，进程启动等等；

vrrp Keepalived的vrrpd子进程以及相关的代码。

check keepalived的healthchecker子进程的目录，包括了所有的健康检查方式以及对应的配置的解析，LVS的配置解析也在这个里面

libipfwc iptables(ipchains)库，主要用来配置LVS中的firewall-mark。

libipvs* 也是使用LVS需要用到的。

2.1.1 多进程模式

keepalived采用了多进程的设计模式，每个进程负责不同的功能，我们在使用LVS的机器上通常可以看到这样的进程：

```
111 Keepalived    < 父进程:内存管理，监控子进程
112 \_ Keepalived < VRRP子进程
113 \_ Keepalived < healthchecker子进程
```

有些命令行参数来控制不开启某些进程，比如不运行LVS的机器上，只开启VRRP就可以了(–P), 如果只运行healthchecker子进程,使用–C。

2.1.2 控制面板

所谓的控制面板就是对配置文件的编译和解析，Keepalived的配置文件解析比较另类，并不是一次统统解析所有的配置，只在用到某模块的时候才解析相应的配置，在每个模块里面都可以看到XXX_parser.c这样的文件，就是做这个作用的。

2.1.3 WatchDog

这种框架提供了对子进程(VRRP和healthchecker)的监控.

2.1.4 IPVS封装

Keepalived里面所有对LVS的相关操作并不直接使用ipvsadm这样的用户端程序，而是直接使用IPVS提供的函数进程操作，这些代码都在check/ipwrapper.c中。

2.2 KeepAlived的安装

安装Keepalived和安装其他开源软件一样，非常的简单，configure，make，make install就可以搞定，但是我们还是需要简单的说明一下这个操作过程：

```
./configure --prefix=/ \
--mandir=/usr/local/share/man \
--with-kernel-dir=/usr/src/kernels/2.6.9-67.EL-smp-i686/
make
make install
```

说明如下：

1. prefix 这个指定为/吧，这样配置文件会放到目录下，方便操作。
2. mandir 这个也放到Linux系统默认的man目录下，方便查看。
3. with-kernel-dir 这是个重要的参数，这个参数并不表示我们要把Keepalived编进内核，而是指使用内核源码里面的头文件，也就是include目录。

如果要用到LVS，才需要这样的指定，否则是不需要的，而且如果要使用netlink，还需要link_watch.c这个文件

在confiure正确的执行后，可以得到下面的输出：

```
Keepalived configuration
------------------------
Keepalived version       : 1.1.15
Compiler                 : gcc
Compiler flags           : -g -O2
Extra Lib                : -lpopt -lssl -lcrypto
Use IPVS Framework       : Yes
IPVS sync daemon support : Yes
Use VRRP Framework       : Yes
Use LinkWatch            : Yes
Use Debug flags          : No
```

注意

Use IPVS Framework IPVS框架—也即LVS的核心代码框架，如果不使用LVS，可以在configure时指定参数*disable-lvs*，这样的话，这里看到的就是No而不是Yes。

IPVS sync daemon support IPVS同步进程，很显然，如果前面那项是No的话，那么这里肯定也是No，当然如果前面这项是Yes–即使用LVS，而不想使用LVS的同步进程(sync daemon), 可以在configure的时候指定*disable-lvs-syncd*。

Use VRRP Framework VRRP框架，这基本上是必须的，Keepalived的核心进程vrrpd。

Use LinkWatch 所谓的Linkwatch大概意思是通过接收内核发出的关于网卡的状态信息来判断网卡的状态，因为是内核发出的信息，这样在用户端只需要捕捉这些信息即可，相比直接在用户端通过其他方式来实现看起来会更省资源，Keepalived在网卡超过20块的情况下推荐使用。[1]

简而言之，如果不使用LVS功能，那么只要看到*Use VRRP Framework*为Yes就可以，反之，必须有*Use IPVS Framework*为Yes，其他都是*optional*的。

安装基本就这么简单，但是你可能还有疑问，到底在哪里下载Keepalived？直接到Keepalived的官网下吧：www.keepalived.org

2.3 KeepAlived配置详解

Keepalived的所有配置都在一个配置文件里面设置，支持的配置项也比较多。但分为三类：

1. 全局配置(Global Configuration)
2. VRRPD配置
3. LVS配置

很明显，全局配置就是对整个keepalived起效的配置，不管是否使用LVS。VRRPD是keepalived的核心，LVS配置只在要使用keepalived来配置和管理LVS时需要使用，如果仅使用keepalived来做HA[2]，LVS的配置完全是不需要的。

配置文件都是以块(block)形式组织的，每个块都在{和}包围的范围内。#和!开头的行都是注释。

[1]需要link_watch.c这个文件，此文件在Linux内核的源代码中，路径类似/usr/src/kernels/2.6.9-67.EL-smp-i686/net/core/ link_watch.c

[2]比如说做一对HAProxy的HA或者其他类似的HA

2.3.1 全局配置

全局配置包括两个子配置，即所谓的：全局定义(global definition)和静态地址路由(static ipaddress/routes)

全局定义

全局定义主要设置keepalived的通知机制和标识：

```
global_defs
{
    notification_email
    {
        admin@example.com
    }
    notification_email_from admin@example.com
    smtp_server 127.0.0.1
    stmp_connect_timeout 30
    router_id my_hostname
}
```

- notification_email指定keepalived在发生事件(比如切换)时，需要发送email到的对象，可以有多个，每行一个。
- smtp_*指定发送email的smtp服务器，如果本地开启了sendmail的话，可以使用上面的默认配置。
- route_id运行keepalived的机器的一个标识。

静态地址和路由

所谓静态(static)就是说不会随vrrpd instance的开/关而变化的，VIP就不是static的，会随着vrrpd而添加/删除。这个配置可以用来给服务器配置静态的IP地址/路由，当然如果服务器的配置里面已经有这些配置，这里就不需要设置了。

```
    static_ipaddress
    {
        192.168.1.1/24 brd + dev eth0 scope global
```

```
    ...
}
static_routes
{
    src $SRC_IP to $DST_IP dev $SRC_DEVICE
    ...
    src $SRC_IP to $DST_IP via $GW dev $SRC_DEVICE
}
```

每一行设置一个IP，这些配置都是Linux下*ip*这个命令的参数，比如上面的`192.168.1.1/24 brd + dev eth0 scope global`，keepalived最终会直接使用*ip addr add 192.168.1.1/24 brd + dev eth0 scope global*来添加，所以这里的配置都要符合ip命令的规则。

这就是全局配置段的全部。

2.3.2 VRRPD配置

VRRPD的配置也包括2部分:VRRP同步组(synchroization group)和VRRP实例(VRRP Instance)。

VRRP Sync Groups(s)

不使用Sync Group的话,如果机器(或者说router)有两个网段，一个内网一个外网,每个网段开启一个VRRP实例，假设VRRP配置为检查内网，那么当外网出现问题时，VRRPD认为自己仍然健康，那么不会发送Master和Backup的切换，从而导致了问题。Sync group就是为了解决这个问题，可以把两个实例都放进一个Sync Group，这样的话，group里面任何一个实例出现问题都会发生切换。

```
vrrp_sync_group VG_1 {
    group {
        inside_network      # 这里是实例名(比如VI_1)
        outside_network
        ...
    }
    notify_master /path/to/to_master.sh
    notify_backup /path_to/to_backup.sh
    notify_fault "/path/fault.sh VG_1"
```

```
    notify /path/to/notify.sh
    smtp_alert
}
```

- notify_master 指定当切换到Master时，执行的脚本，这个脚本可以传入参数(引号引起)，其他2个类推。
- notify指令有3个参数，这些参数由keepalived提供：$1(GROUP—INSTANCE),$2(group或者instance的名字),$3(MASTER—BACKUP—FAULT)
- smtp_alter 使用global_defs里面定义的邮件地址和smtp服务器在切换后发送邮件通知。

VRRP实例(instance)配置

VRRP实例就表示在上面开启了VRRP协议，这个实例说明了VRRP的一些特性，比如主从、VRID等等，可以在每个interface上开启一个实例。VRRP实例配置主要定义vrrp_sync_group里面的每个组的漂移IP等。

```
vrrp_instance   inside_network {
    state MASTER
    interface eth0
    dont_track_primary

    track_interface {
        eth0
        eth1
    }

    mcast_src_ip <IPADDR>
    garp_master_delay 10
    virtual_router_id 51
    priority 100
    advert_int 1

    authentication {
        auth_type PASS
        autp_pass 1234
```

```
    }

    virtual_ipaddress {
#<IPADDR>/<MASK> brd <IPADDR> dev <STRING> scope <SCOPT> label <LABEL>
        192.168.200.17/24 dev eth1
         192.168.200.18/24 dev eth2 label eth2:1
    }

    virtual_routes {
# src <IPADDR> [to] <IPADDR>/<MASK> via|gw <IPADDR> dev <STRING> scope <SCOPE>
        src 192.168.100.1 to 192.168.109.0/24 via 192.168.200.254 dev eth1
        192.168.110.0/24 via 192.168.200.254 dev eth1
        192.168.111.0/24 dev eth2
        192.168.112.0/24 via 192.168.100.254
    }
    nopreempt
    preemtp_delay 300
    debug
}
```

---

state state指定instance的初始(Initial)状态，在两台router都启动后，马上会发生竞选，高priority的会竞选为Master，所以这里的state并不表示这台就一直是Master。

interface inside_network实例绑定的网卡

dont_track_primary 忽略VRRP的interface错误(默认不设置)

track_interface 设置额外的监控,里面的任意一个网卡出现问题，都会进入FAULT状态

mcast_src_ip 发送多播包的地址,如果不设置，默认使用绑定的网卡的primary IP。

garp_master_delay 在切换到MASTER状态后，延迟进行gratuitous ARP请求

virtual_router_id VRID标记(0...255)

priority 100 高优先级竞选为MASTER，MASTER要高于BACKUP至少50

advert_int 检查间隔,默认1s

virtual_ipaddress 里面指定漂移地址(VIP)，也就是切换到MASTER时，这些IP会被添加，切换到BACKUP时，这些IP被删除(传给ip addr命令),所以每台服务器上可以不绑定任何虚拟地址，而都把他们放virtual_ipaddress里面(可以多个)，keepalived会自动使用ip addr进行绑定(不需要以来ifcfg-eth0),ip add可以看到

virtual_routes 和virtual_ipaddress一样，发生切换时添加/删除路由

lvs_sync_daemon_interface lvs syncd绑定的网卡

authentication 这一段设置认证

auth_type 认证方式，支持PASS和AH

auth_pass 认证的密码

nopreempt 设置为不抢占，注意这个配置只能设置在state为BACKUP的主机上，而且这个主机的priority必须比另外一台高

preempt_delay 抢占延迟，默认5分钟

debug Debug级别

notify_master 和sync group里面的配置一样。

2.3.3 LVS配置

LVS的配置也包括2部分:虚拟主机组(virtual server group)和虚拟主机(virtual server)。这些配置都会传递给ipvsadm作为参数。

虚拟主机组

这个配置段是可选的，目的是为了让一台RealServer上的某个service可以属于多个Virtual Server，并且只做一次健康检查。

```
virtual_server_group <STRING> {
    # VIP port
    <IPADDR> <PORT>
    <IPADDR> <PORT>
```

```
    ...
    fwmark <INT>
}
```

虚拟主机

virtual_server可以以下面3种方式中的任意一种配置：

1. virtual_server IP port
2. virtual_server fwmark int
3. virtual_server group string

如下例：

```
virtual_server 192.168.1.2 80 {          # 设置一个virtual server: VIP:Vport
    delay_loop 3                         # service polling的delay时
间
    lb_algo     rr|wrr|lc|wlc|lblc|sh|dh # LVS的调度算法
    lb_kind     NAT|DR|TUN               # LVS集群模式
    persistence_timeout 120              # 会话保持时间(秒)
    persistence_granularity <NETMASK>    # LVS会话保持粒度，ipvsadm中
的-M参数，默认是0xffffffff，即根据每个客户端做会话保持。
    protocol    TCP                      # 使用的协议是TCP还是UDP
    ha_suspend                           # suspendhealthchecker's activity

    virtualhost <string>                 # HTTP_GET做健康检查时，
检查的Web服务器的虚拟主机（即Host:头）

    sorry_server <IPADDR> <PORT>         # 备用机，所有的real server失
效后启用

    # 每台RealServer都需要一个下面的配置项
    real_server <IPADDR> <PORT>
    {
        weight 1                         # 默认为1，0为失效
        inhibit_on_failure               # 在服务器健康检查失败
时，将其weight设置为0，而不是直接从IPVS里面删除。
```

```
        notify_up   <STRING> | <QUOTED-STRING>  # 在检测到service up后
执行的脚本
        notify_down <STRING> | <QUOTED-STRING>  # 在检测到service down后
执行的脚本
# 下面配置任意一种健康检查方式:HTTP_GET|SSL_GET|TCP_CHECK|SMTP_CHECK|MISC_CHECK
        HTTP_GET | SSL_GET
        {
            url {     # HTTP/SSL检查的URL，这里可以指定多个URL
                path /
                digest <STRING> # SSL检查后的摘要信息(genhash工
具算出)
                status_code 200   # HTTP检查的返回状态码
            }
            connect_port 80       # 健康检查端口
            # 以此地址发送请求对服务器进行健康检查
            bindto  <IPADD>
            connect_timeout       # 连接超时时间
            nb_get_retry    3     # 重连次数
            delay_before_retry   2  # 重连间隔时间（秒）
        } # END OF HTTP_GET|SSL_GET

       # TCP方式的健康检查
       TCP_CHECK {
          connect_port 80
          bindto 192.168.1.1
          connect_timeout 4
       } # TCP_CHECK

       # SMTP 方式健康检查
       SMTP_CHECK

              #  这里的配置意义和HTTP里面的类似
              host {
                  connect_ip <IP ADDRESS>
                  connect_port <PORT>           # 默认检查25端口
```

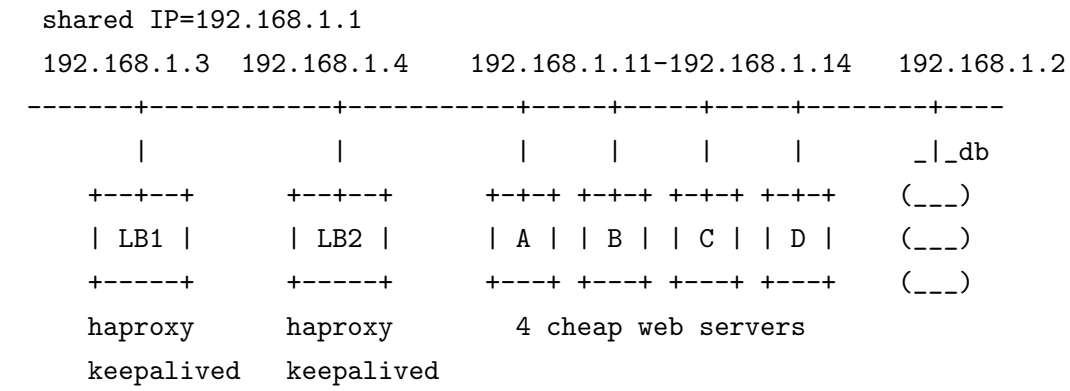
```
                bindto <IP ADDRESS>
            }
            connect_timeout <INTEGER>
            retry <INTEGER>
            delay_before_retry <INTEGER>
            # "smtp HELO"请求命令的参数，可选的。
            helo_name <STRING>|<QUOTED-STRING>
        } #SMTP_CHECK

        #MISC健康检查方式，执行一个程序
        MISC_CHECK
        {
            # 外部程序或脚本路径
            misc_path <STRING>|<QUOTED-STRING>
            # 脚本执行的超时时间
            misc_timeout <INT>

            # 如果设置了misc_dynamic的话， healthchecker程序
的退出状态码会用来动态调整服务器的权重(weight).
            # 返回0：健康检查OK，权重不被修改
            # 返回1：健康检查失败，权重设为0
            # 返回2-255：健康检查OK，权重设置为： 退出状态码-2，
比如返回255，那么weight=255-2=253
            misc_dynamic
         }


    } # Realserver
}   # Virtual Server
```

第三章 应用实例

本章主要介绍两种Keepalived的使用，一种仅使用Keepalived做HA，一种既做HA又用来配置LVS。

3.1 用Keepalived做HA

用Keepalived做HA是如此的简单，我们所有的精力都在配置Keepalived的VRRP子进程上，而完全可以不管LVS的配置。这节我们假设给两台运行HAProxy的机器做HA，提供VIP。架构如下：



3.1.1 HAProxy和web服务器配置

这部分不属于本文讨论范围内，我们只关注在haproxy的机器上的Keepalived相关配置。

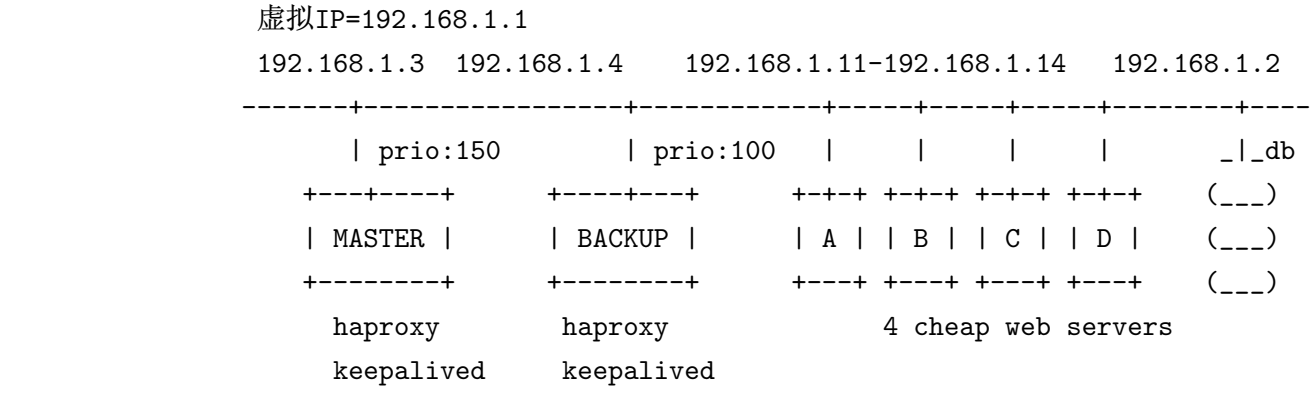
3.1.2 Keepalived配置

假设haporxy和后端的web服务器都准备好了，现在就可以开始配置Keepalived了。这种情况下，因为我们只用到Keepalived的HA，即做virtual

router的功能，所以只需要完成第二章里面描述的全局和VRRP配置即可。配置前需要准备的信息为：

1. VRID—我们使用默认的51[1]
2. VIP—上图的192.168.1.1, VIP也即Keepalived里面的vritual address的ip地址
3. LB的IP—这里就是两台haproxy的地址（192.168.1.3和1.4，并假设前者为MASTER）
4. 优先级–priority，在多个BACKUP竞选MASTER的时候，优先级高会获选。这里假设192.168.1.3的优先级为150，另外一台为100

所以，实际要配置为：



不抢占

当MASTER出现问题后，BACKUP会竞选为新的MASTER，那么当之前的MASTER重新ONLINE后，是继续成为MASTER还是变成BACKUP呢？默认情况下，如果没设置不抢占，那么之前的MASTER起来后会继续抢占成为MASTER，也就是说，整个过程需要发生2次切换：

1. MASTER->BACKUP
2. BACKUP->MASTER

这样对业务频繁的切换是不能容忍的，因此我们希望MASTER起来后，成为BACKUP！所以要设置不抢占。Keepalived里面提供了nopreempt这个配置，但是这个配置只能用在state为BACKUP的机器上，但是我们

[1]真是幸运五十一啊！

明明希望的是MASTER不进行抢占，没办法，MASTER的state也得设置成BACKUP。也就是说192.168.1.3和192.168.1.4都要将state设置为BACKUP！

那么到底谁是MASTER？抢占吧，通过priority，所以我们在两台BACKUP上面通过设置不同的priority来让他们一起来就抢占，高priority的192.168.1.3成为最初的MASTER。

安装

MASTER和BACKUP上的安装过程完全一样，使用下面的命令(这里假设在RedHat Enterprise AS4 Update 4上安装)：

```
$wget http://www.keepalived.org/software/keepalived-1.1.17.tar.gz
$tar xzvf keepalived-1.1.17.tar.gz
$cd keepalived-1.1.17
$./configure --prefix=/ \
--mandir=/usr/local/share/man \
--with-kernel-dir=/usr/src/kernels/2.6.9-67.EL-smp-i686/
$make
#make install
#cp keepalived/etc/init.d/keepalived.rh.init /etc/init.d/keepalived
#chmod +x /etc/init.d/keepalived
#cp keepalived/etc/init.d/keepalived.sysconfig /etc/sysconfig/keepalived
#chkconfig --add keepalived
#chkconfig --level 345 keepalived on
```

MASTER的配置

MASTER的配置—192.168.1.3的配置

```
global_defs {
   notification_email {
       finalbsd@gmail.com
  }
   notification_email_from finalbsd@gmail.com
   smtp_server 127.0.0.1
   smtp_connect_timeout 30
   router_id haproxy-ha
```

```
}

vrrp_sync_group VG1 {
        group {
                VI_1
        }
}

vrrp_instance VI_1 {
    state BACKUP
        smtp_alert
        notify_fault "/root/script/notify_mail.sh fault"
        notify_master "/root/script/notify_mail.sh master"
        nopreempt
        interface eth1
    track_interface {
        eth0
        eth1
    }
    virtual_router_id 51
    priority 150
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass fsaf..7&f
    }
    virtual_ipaddress {
        192.168.1.1/24 dev eth1 scope global
    }
}
```

BACKUP的配置

BACKUP的配置—192.168.1.4的配置
BACKUP的配置和MASTER基本一样，但有2个不同点：

1. priority 设置为100

2. 不设置nopreempt

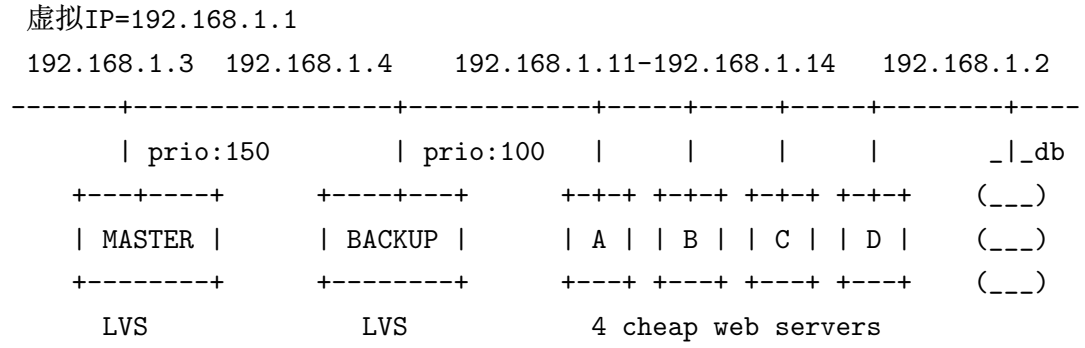
启动服务

分别在两台上执行service keepalived start启动服务。

3.2 用Keepalived配置LVS

我们假设下面的情形：

4台web服务器通过一对LVS进行调度，LVS转发模式为NAT。一对LVS通过Keepalived做HA，



在LVS中，多了几个概念：

virtual server 就是VIP+VPORT，VIP需要通过VRRPD配置段进行配置。

real server 这里LVS调度的web服务器的IP地址，即图中的192.168.11–14

lb_algo 调度算法，这里使用wlc

lb_kind 包转发模式，这里使用NAT

weight 权重，默认这里都设置为3

安装

安装和上一节3.1.2讲的安装完全一样，这里不再赘述。

MASTER和BACKUP配置

Keepalived的全局和vrrp配置段和前一节3.1.2完全一样，这里我们只需要添加virtual_server配置段：

```
virtual_server 192.168.1.1 80 {
        delay_loop 3
        lb_algo wlc
        lb_kind DR
        persistence_timeout 1200
        protocol TCP
        ha_suspend

        real_server 192.168.1.11 80 {
                weight 3
                TCP_CHECK {
                        connect_timeout 3
                }
        }
        real_server 192.168.1.12 80 {
                weight 3
                TCP_CHECK {
                        connect_timeout 3
                }
        }
}
```

参考文献

[1] RFC 3768 — Virtual Router Redundancy Protocol (VRRP)

[2] man 5 keepalived.conf

[3] man 8 keepalived

[4] www.keepalived.org

[5] keepalived源代码