HAProxy+HAProxyConsole+Keepalived、BIND安装配置文档

yongfengxia(夏永锋)

October 16, 2013

Contents

1 总述

1.1 部署过程

1. 创建一个临时目录并进入该目录：`mkdir -p /data/tmp/ && cd /data/tmp/`；
2. 远程拷贝软件包：`scp -P xxx xxx@xxx.xxx.xxx.xxx:/home/xxx/deploy_pack.tar.gz ./`，密码：xxx；
3. 解压缩deploy_pack.tar.gz并进入解压缩得到的deploy_pack目录：`tar -xvf deploy_pack.tar.gz && cd deploy_pack`；
4. 若其中的deploy.sh，run.sh脚本没有可执行权限，则应执行：`chmod +x deploy.sh run.sh`；
5. 执行deploy.sh脚本安装程序：master机器上执行命令`./deploy.sh master`，slave机器上执行命令`./deploy.sh slave`，然后根据提示选择安装哪些程序；
6. 根据deploy.sh脚本输出的最后几行信息，编辑HAProxy、HAProxyConsole、Keepalived、BIND的配置文件：
   - HAProxy：*/etc/rsyslog.conf*
   - HAProxyConsole：*/usr/local/haproxyconsole/conf/app_conf.ini*（仅master机器上需要）
   - Keepalived：*/etc/keepalived/keepalived.conf*、*/etc/keepalived/Mailnotify.py*
   - BIND：*/usr/local/bind9.9.3/etc/named.conf*

   每个配置如何修改见下文详细解析。

7. 完成配置文件的修改后，执行run.sh脚本启动程序：master机器上执行命令`./run.sh master`，slave机器上执行命令`./run.sh slave`，然后根据提示选择启动哪些程序；

8. 根据run.sh输出的最后几行信息，确认服务是否成功启动：

   - HAProxy：`netstat -tlnp | grep haproxy`
   - HAProxyConsole（仅master服务器上会安装）：
     (a) `netstat -tlnp | grep haproxyconsole`；
     (b) 访问web页面**http://[服务器ip]:9090**，HAProxyConsole默认端口为9090
   - Keepalived：
     (a) 在master机器上执行`ip a`，确认网卡已绑定Vip；
     (b) `tcpdump vrrp`，查看vrrp协议数据包，应只有master服务器会发出vrrp数据包
   - BIND：`/usr/local/bind9.9.3/bin/dig @机器ip www.baidu.com A`，确认能否完成DNS解析

1.2 说明

1. 由于本文讲述的HAProxy、HAProxyConsole、Keepalived、BIND都涉及主从服务器概念，为行文方便现假设：

   - 主服务器ip为：192.168.2.194
   - 从服务器ip为：192.168.2.193
   - Keepalived的Vip为：192.168.2.201

2. 下文涉及的配置文件中标为红色的配置项都应根据实际情况进行修改。配置文件中标为灰色的部分表示该注释是本文笔者自己添加的以帮助读者理解配置项的含义。

2 HAProxy+HAProxyConsole+Keepalived

2.1 系统结构图

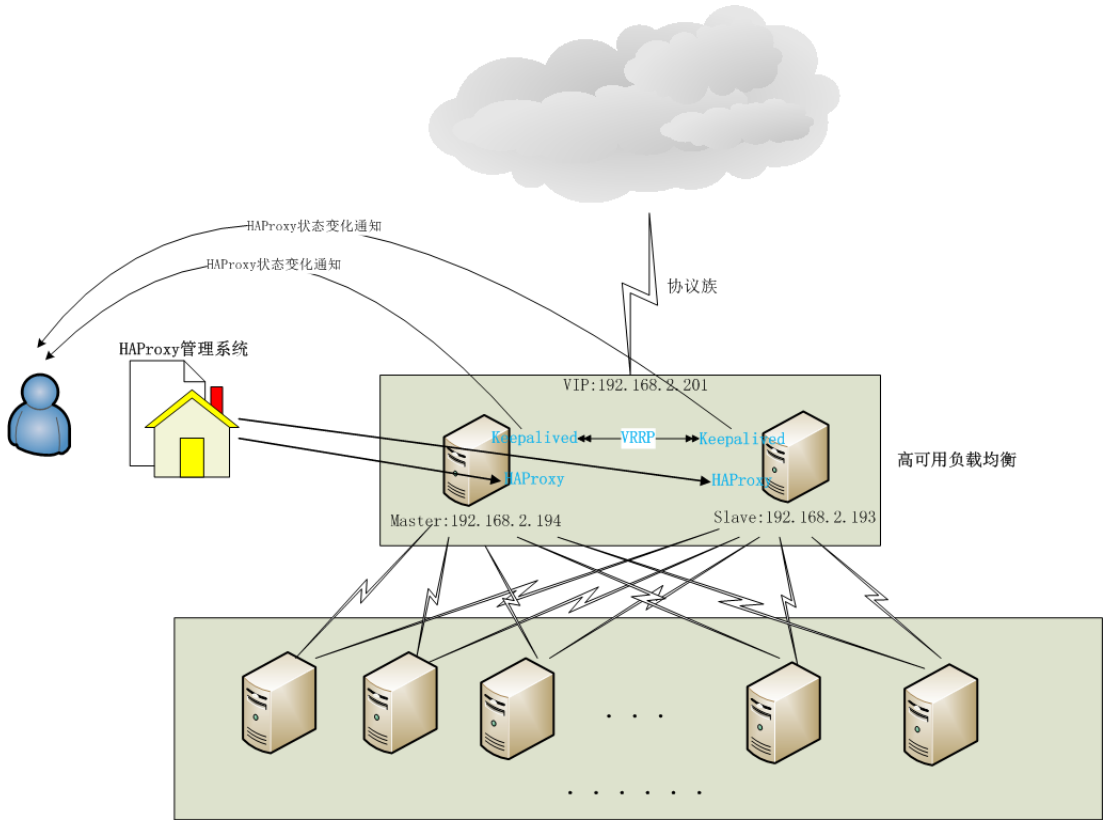

Figure 1: HAProxy+HAProxyConsole+Keepalived系统结构图

图中中间部分有两台服务器（192.168.2.194和192.168.2.193）（角色分别为主、从），其上安装了HAProxy和Keepalived。

正常情况下，Keepalived会将VIP（192.168.2.201）绑定到主服务器上，外部网络通过VIP进行服务请求，即正常情况下，仅主服务器会接收到服务请求，HAProxy则将请求根据一定规则分配转发到后端的某台服务器上。

两台服务器上的Keepalived通过VRRP协议相互通信，并监听自己所在服务器上的HAProxy进程状态。

当主服务器上的HAProxy无法正常工作时，两个Keepalived进程经过通信，从服务器上的Keepalived会将VIP绑定到从服务器上，这时外部的服务请求则由从服务器上的HAProxy进行负载均衡处理，这个切换过程一般在2、3秒内即可完成。

在HAProxy的主从状态发生切换时，各自服务器上的Keepalived都会向管理员发出通知（邮件、短信等，可通过脚本程序实现）。

管理员恢复主服务器上的HAProxy进程正常工作后，Keepalived会再次将VIP绑定到主服务器上，从而继续由主服务器的HAProxy来完成请求的负载均衡。

这样Keepalived保证了HAproxy负载均衡器不会成为系统的单点，实现高可用。

HAProxyConsole仅安装在主服务器上，负责管理主、从服务器HAProxy的配置和HAProxy进程。

2.2 HAProxy

2.2.1 简介

HAProxy是一个高性能的代理服务器方案，可以提供7层和4层代理，具有healthcheck，负载均衡等多种特性，大量互联网服务内部都有应用（如Github、Twitter、Reddit、Disqus、Instagram、Stack Overflow、阿里等）。

项目主页：http://haproxy.1wt.eu

最新稳定版本1.4的文档见：http://cbonte.github.io/haproxy-dconv/configuration-1.4.html

在我们当前应用中主要将HAProxy用作**TCP协议的负载均衡**。

2.2.2 配置

HAProxy配置文件样例如下所示：

/usr/local/haproxy/conf/haproxy.conf

```
global
    # 使用系统的syslog记录日志（通过udp，默认端口号为514）
    log 127.0.0.1 local3 debug
    # 让进程在后台运行，即作为守护进程运行，等价于命令行的“-D”选项。也可以通过命令行“-db”选项来禁用。
    daemon
    # 进程所属用户
    user root
    # 进程所属用户组
    group root
    # 限制单个进程的最大连接数为65535
    maxconn 65535
    # 指定作为守护进程运行的时候，要创建多少个进程，默认只创建一个，需要daemon开启模式
    nbproc 1
    # pid文件的存放目录
    pidfile /usr/local/haproxy/haproxy.pid

defaults
    # 在连接失败或断开的情况下，允许当前会话被重新分发
    option redispatch
    # 不记录空连接
    option dontlognull
    # 设置在一个服务器上链接失败后的重连次数
    retries 3
    # 设置服务器分配算法，此处为轮询
    balance roundrobin

    # 设置等待连接到服务器成功的最大时间
    timeout connect 5000ms
    # 设置客户端的最大超时时间
    timeout client 1800000ms
    # 设置服务器端的最大超时时间
    timeout server 1800000ms
```

```
# 启用HAProxy自带的数据统计页面
listen stats
    # 绑定8888端口
    bind *:8888
    # http模式
    mode http
    # 开启日志
    option httplog
    log global
    # 数据统计页面的相对路径URL
    stats uri /
    # 身份认证输入框的提示信息
    stats realm Haproxy  statistic
    # 页面自动刷新的时间间隔
    stats refresh 30s

    # 启用统计报告
    stats enable
    # 在统计页面上报告额外的信息
    stats show-legends
    # 启用统计管理级别
    # 管理级别允许通过web界面启用/禁用后端服务器。默认情况下，统计页面考虑到安全问题是只读的。
    stats admin if TRUE
    # 启用带身份认证的统计信息页面，并赋予某账户访问权限。
    stats auth admin:admin
    # 隐藏HAProxy版本。
    stats hide-version

# 一个“listen”部分定义一个完整的代理，将前端后端部分合并入一个部分中。对于仅是TCP的流量通常比较有用。
# “Listen-10000”为该listen部分的名称，可以赋予具有一定含义的名称，以便在统计页面上区分不同的listen部分。
listen Listen-10000
    # 绑定端口10000
    bind *:10000
    # tcp负载均衡模式
    mode tcp
    # 开启日志
    option tcplog
    log global

    # 定义该listen的后端
    # server之后的是该后端的名称，接着是该后端的ip和端口
    # weight -- 调节服务器的权重
    # check -- 允许对该服务器进行健康检查
    # inter -- 设置连续的两次健康检查之间的时间，单位为毫秒(ms)，默认值 2000(ms)
    # rise -- 指定多少次连续成功的健康检查后，即可认定该服务器处于可操作状态，默认值 2
    # fall -- 指定多少次不成功的健康检查后，认为服务器为当掉状态，默认值 3
    server 192.168.2.225:53378 192.168.2.225:53378 weight 3 check inter 2000 rise 2 fall 3
    server 192.168.2.226:53378 192.168.2.226:53378 weight 3 check inter 2000 rise 2 fall 3
```

由于我们会使用HAProxyConole来管理HAProxy，所以不需要手动编辑HAProxy配置文件。

HAProxy日志是通过系统日志服务器输出到日志文件的。CentOS中默认日志服务器是rsyslogd，为了支持HAProxy日志输出，需配置主、从服务器上的rsyslogd。

在*/etc/rsyslog.conf*中取消如下两行的注释：

```
#$ModLoad imudp.so
#$UDPServerRun 514
```

并添加一行：

```
local3.*        /var/log/haproxy.log
```

这行配置的意思是将来自local3的所有级别的日志信息都写入文件*/var/log/haproxy.log* 中。其中*local3*是与*haproxy.conf*中“log 127.0.0.1 local3 debug”一行中指定的相同的。

2.2.3 使用

1. 检查配置文件有效性：

```
/usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/conf/haproxy.conf -c
```

2. 启动HAProxy

```
/usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/conf/haproxy.conf
```

3. 重启HAProxy（即restart_haproxy.sh的内容）

```
/usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/conf/haproxy.conf -st \
                                                 ‘cat /usr/local/haproxy/haproxy.pid‘
```

2.3 HAProxyConsole

2.3.1 简介

HAProxyConsole是一个我们自己实现的简单的HAProxy负载均衡任务管理系统。由于HAProxy的负载均衡任务可能会很多，手动编辑配置文件非常不方便、不安全，所以实现一个友好的管理系统是非常必要的。

HAProxyConsole已实现功能：

1. TCP协议负载均衡任务的增删改、任务的列表展示；
2. 一键应用最新配置到主服务器或从服务器并重启HAProxy进程；
3. 修改一个配置项即可在文件存储和数据库存储之间切换，且文件存储不需任何初始化操作；
4. 内置小工具用于不同存储方式之间的数据转换；
5. 内嵌主从HAProxy自带数据统计页面，方便查看信息。

2.3.2 配置

由于HAProxyConsole涉及主从HAProxy配置的管理以及HAProxy的重启，所以会有较多的配置信息，其配置文件示例如下：

/usr/local/haproxyconsole/conf/app_conf.ini

```
; The configuration file consists of sections,
; led by a "*[section]*" header and followed by "*name: value*" entries
; "*name=value*" is also accepted. Note that leading whitespace is removed from values.
; The optional values can contain format strings which refer to other values in the same section,
; or values in a special DEFAULT section. Additional defaults can be provided on initialization
; and retrieval. Comments are indicated by ";" or "#"; a comment may begin anywhere on a line,
; including on the same line after parameters or section declarations.

[master]
; 主HAProxy配置文件的路径
MasterConf=/usr/local/haproxy/conf/haproxy.conf
; 主HAProxy重启脚本的路径
MasterRestartScript=/usr/local/haproxy/restart_haproxy.sh


[slave]
; 从HAProxy机器远程登录: 服务器ip:port,用户名,密码
SlaveServer=192.168.2.193:xxxx
SlaveRemoteUser=root
SlaveRemotePasswd=xxx

; 从HAProxy配置文件的路径
```

```
SlaveConf=/usr/local/haproxy/conf/haproxy.conf
; 从HAProxy重启脚本的路径
SlaveRestartScript=/usr/local/haproxy/restart_haproxy.sh


[store]
; 数据存储方式: 数据库(DB,以0表示)与文件(FILE,以1表示)两种
StoreScheme=1

; MySQL数据库连接信息
DBDriverName=mysql
; DBDataSourceName的格式为: 数据库用户名:密码@数据库访问协议(数据库服务器的ip:端口)/数据库名称?charset=utf8
; 其中“数据库访问协议”默认为tcp, “数据库”为haproxyconsole
DBDataSourceName=xxx:xxx@tcp(xxx.xxx.xxx.xxx:3306)/haproxyconsole?charset=utf8

; 若采用文件来存储负载均衡任务数据, 则需指定该文件路径
;  负载均衡任务数据文件路径
FileToReplaceDB=../conf/DB.json


[stats]
; 主HAProxy数据统计页面URL
MasterStatsPage=http://192.168.2.194:8888

; 从HAProxy数据统计页面URL
SlaveStatsPage=http://192.168.2.193:8888


[others]
; 主从HAProxy的VIP
Vip=192.168.2.201

; 根据负载均衡任务数据生成的HAProxy新配置文件存放路径
NewHAProxyConfPath=../conf/haproxy_new.conf
```

2.3.3 服务器端操作

- 若app_conf.ini中StoreScheme参数设置为0，即选择数据库（MySQL）存储方式，则需另外安装MySQL，并在MySQL中执行app.sql中的SQL语句进行数据库初始化；若StoreScheme设置为0，即选择文件存储方式，则无需数据库和初始化操作。
- 启动HAProxyConsole: `cd /usr/local/haproxyconsole/bin && ./haproxyconsole &`，默认端口为9090，可使用选项*-p*来自定义端口，如: `cd /usr/local/haproxyconsole/bin && ./haproxyconsole -p 8080`（注意该端口不应在HAProxyConsole为HAProxy负载均衡任务自动分配的端口范围之内(10000-20000)）。
- 若需转换数据存储方式，则可通过内置工具来完成: `cd /usr/local/haproxyconsole/bin && ./haproxyconsole -t`，该工具完成的操作是: 若StoreScheme设定为0，则从DBDriverName和DBDataSourceName 指定数据库的haproxymapinfo数据表中读取数据，转换成json格式存入FileToReplaceDB指定路径的JSON文件中。

2.3.4 浏览器端操作图文教程

1. 添加TCP负载均衡任务

Figure 2: TCP负载均衡任务添加页面截图

Figure 3: 填入相关信息点击“预览”后的截图

“预览”功能主要是解析出填写的ip:port列表并让用户确认。

Figure 4: 点击“提交”后返回Vip:port的截图

“提交”后，申请Vip:port成功后，会清空表单，并在表单下方显示Vip:port。

2. TCP负载均衡任务列表

Figure 5: 负载均衡任务列表

在列表中可以看到刚添加的负载均衡任务，但此时新增任务还未应用到主从服务器。点击图中右上方的“应用到主HAProxy”和“应用到备HAProxy”按钮可以将新增任务分别应用到HAProxy主从服务器。

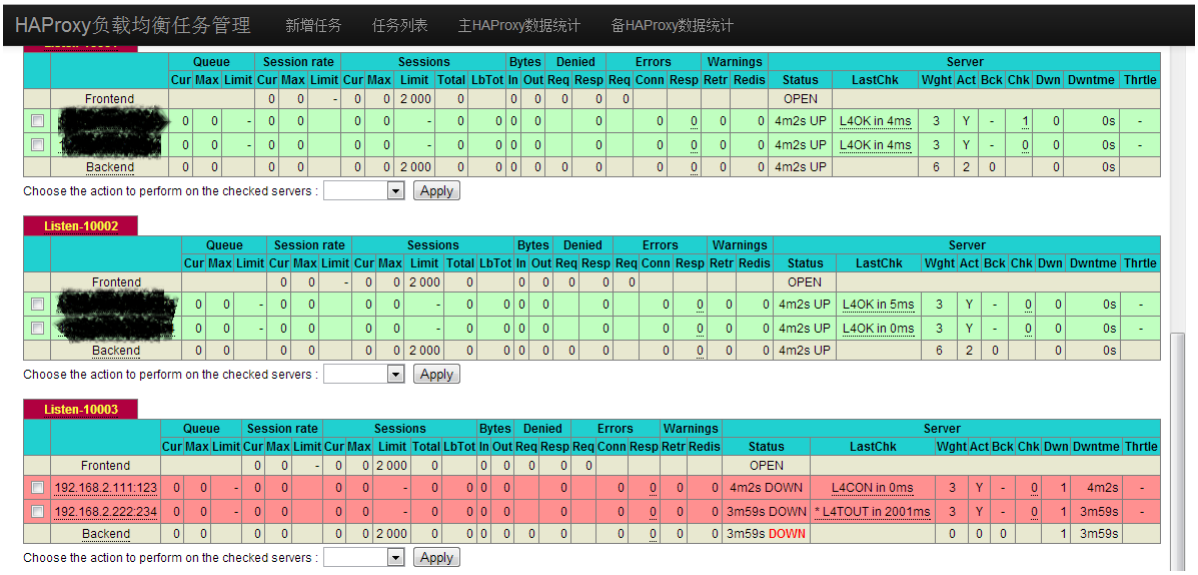

Figure 6: 主HAProxy服务器数据统计页面

在Figure 4中点击“应用到主HAProxy”按钮后，在“主HAProxy数据统计”页面中可以看到新增任务的数据统计项。该项显示为“红色”，是因为这两个后端ip:port并不存在，HAProxy心跳检测结果就将这两个后端判定为DOWN（宕机）。

3. 删除、修改TCP负载均衡任务

Figure 7: 删除、编辑按钮

若要删除或修改刚才添加的任务，可以在该任务所在行的“申请/更新时间”一列中鼠标左击一下就会显示出“删除”“修改”按钮（再次左击一下就会消失）。删除或修改任务后，应点击“应用到主HAProxy”和“应用到备HAProxy”按钮来生成新的HAProxy配置文件分别应用到主备HAProxy。

2.4 Keepalived

2.4.1 简介

项目主页：http://www.keepalived.org

Keepalived是一个高可用方案，通过VIP(即虚拟IP)和VRRP协议来实现高可用。

在当前应用中，Keepalived是用于**实现HAProxy负载均衡服务的高可用**。

2.4.2 配置

安装部署第4个步骤中的keepalived.conf即为Keepalived的配置文件，其内容大致如下（该配置文件内容在主、从服务器上有所不同，请仔细阅读注解）：

/etc/keepalived/keepalived.conf

```
! Configuration File for keepalived
global_defs {
```

```
    ! 运行keepalived机器的一个标识，从为HAProxyBackup
    router_id HAProxyMaster
}
! Keepalived脚本chk_haproxy
vrrp_script chk_haproxy {
    ! 脚本内容为“killall -0 haproxy”
    script "killall -0 haproxy"
    ! 脚本执行的时间间隔为2秒
    interval 2
    weight 2
}
! Keepalived实例
vrrp_instance VI_1 {
    ! 指定当前Keepalived服务器为主服务器（MASTER），若是从服务器，则为BACKUP
    state MASTER
    ! 指定实例绑定的网卡
    interface eth0
    ! 运行keepalived机器的一个标识ID，主从一致
    ! 相同的VRID为一个组，它将决定多播的MAC地址
    virtual_router_id 51
    ! 优先级，高优先级竞选为MASTER，从Keepalived服务器的设置为99
    priority 100
    ! 多长时间发送一次VRRP通告报文，默认1秒
    advert_int 1
    ! 设置主从认证
    authentication {
        ! 认证方式
        auth_type PASS
        ! 认证密码
        auth_pass admin
        }
    ! Vip
    virtual_ipaddress {
        192.168.2.201
    }
    ! 跟踪脚本chk_haproxy的执行结果
    track_script {
        chk_haproxy
    }
    ! 指定当切换到master模式时，执行的邮件通知脚本
    notify_master "/usr/bin/python /etc/keepalived/Mailnotify.py master"
    ! 指定当切换到slave模式时，执行的邮件通知脚本
    notify_backup "/usr/bin/python /etc/keepalived/Mailnotify.py backup"
    ! 发生故障时执行的邮件通知脚本
    notify_fault "/usr/bin/python /etc/keepalived/Mailnotify.py fault"
}
```

另一个文件Mailnotify.py是用于在Keepalived检测到HAProxy进程状态发生变化时邮件通知管理员，其内容如下所示：

/etc/keepalived/Mailnotify.py

```
#!/usr/bin/python
#coding: utf-8

'''
Keepalived邮件通知系统

Author: yongfengxia
'''

import sys
import smtplib
```

```
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

#将本机ip填写在这里
LOCAL_IP = ''
#填写邮件发送方的SMTP服务器地址
SMTP_SERVER = 'smtp.xxx.com'
#发送方邮箱
FROM_ADDR = 'xxx@xxx.com'
#发送方邮箱登陆密码
PASSWD = 'xxxxxx'
#接收方邮箱，支持多个
TO_ADDR = ['xxx@xxx.com', 'xxx@xxx.com']

msg = MIMEMultipart('alternative')
msg['Subject'] = '负载均衡器HAProxy邮件通知'
msg['From'] = FROM_ADDR
msg['To'] = ';'.join(TO_ADDR)

if sys.argv[1] != 'master' and sys.argv[1] != 'backup' and sys.argv[1] != 'fault':
    sys.exit()
else:
    notify_type = sys.argv[1]

text = '您好! %s 上HAProxy %s 模式被激活，请注意! ' % (LOCAL_IP, notify_type)
html = '''
<html>
    <head></head>
    <body>
        <p>您好，</p>
        <font color=red>%s 上HAPrpoxy负载均衡器 %s 模式已被激活，请注意! </font>
        <br /><br /><br />
        %s，HAProxy + Keepalived高可用负载均衡架构
    </body>
</html>
''' % (LOCAL_IP, notify_type, LOCAL_IP)

text_part = MIMEText(text, 'plain')
html_part = MIMEText(html, 'html')
msg.attach(text_part)
msg.attach(html_part)

smtp = smtplib.SMTP()
smtp.connect(SMTP_SERVER)

# 使用SSL/TLS加密，对于发送邮箱是Gmail的需要启用以下三行代码
#smtp.ehlo()
#smtp.starttls()
#smtp.ehlo()

smtp.login(FROM_ADDR, PASSWD)
smtp.set_debuglevel(1)
smtp.sendmail(FROM_ADDR, TO_ADDR, msg.as_string())
smtp.quit()
```

2.4.3 使用

1. 启动Keepalived:

```
/usr/local/keepalived/sbin/keepalived -P -D -S 4 -d
```

2. 重启Keepalived（先kill进程，然后启动）

```
pkill keepalived
/usr/local/keepalived/sbin/keepalived -P -D -S 4 -d
```

3 BIND

3.1 DNS工作原理简介

以访问http://www.google.com.hk为例。在浏览器导航栏中输入http://www.google.com.hk并回车后，浏览器首先根据域名www.google.com.hk通过DNS查找找到对应的ip。DNS 查找过程为：

1. 查找浏览器的DNS缓存，如果未找到对应的ip，则继续；
2. 查找本地机器的host文件，如果未找到，则继续；
3. 本地机器发送DNS查询请求到本地机器网络设定的域名服务器（Linux在/etc/resolv.conf中由参数nameserver设定）（这里以localDNS指代），localDNS在自己的配置和缓存中查找，若未找到，则将该DNS查询请求发送到互联网的根域名服务器（.）；
4. 根域名服务器根据请求的域名www.google.com.hk （根域名服务器有*.hk*域名服务器的记录）响应告诉localDNS应该去*.hk*域名服务器查找；
5. localDNS再次将DNS查询请求发送到*.hk*，*.hk*中有域名服务器*.com.hk*的记录，所以响应告诉localDNS应去*.com.hk*查找；
6. localDNS再次将DNS查询请求发送到*.com.hk*，*.com.hk*中有域名服务器*.google.com.hk*的记录，所以响应告诉localDNS应去*.google.com.hk* 查找；
7. localDNS再次将DNS查询请求发送到*.google.com.hk*，*.google.com.hk*查找自己的记录，发现有主机www（即域名www.google.com.hk）记录，则将该记录中的ip 发送给localDNS；
8. localDNS接收到结果后再将结果响应发送给本地机器，从而完成域名解析过程。

上述过程中涉及的域名服务器通常都是逐层授权的域名服务器。为了加速DNS查询过程，可以在本地机器与localDNS之间加一个非授权的缓存DNS服务器，利用该DNS服务器的缓存功能，在查询localDNS之前先查询该缓存DNS服务器。该缓存服务器也可以用于实现DNS劫持。

3.2 BIND简介

BIND是DNS协议的一种实现，全名为**B**erkeley **I**nternet **N**ame **D**aemon，是现今互联网上最常使用的DNS服务器软件，使用BIND作为服务器软件的DNS服务器约占所有DNS服务器的九成。BIND现在由互联网系统协会（Internet Systems Consortium）负责开发与维护。

项目主页：http://www.isc.org/downloads/bind/

当前应用中是将BIND作为**主从缓存DNS服务器**来使用。

3.3 BIND配置

bind9.9.3文件夹中：

- bin子目录中有一些DNS查询工具，如nslookup，dig等。
- data子目录用于存放BIND相关的数据，如域配置文件、统计数据文件等。
- etc子目录用于存放BIND相关配置文件。
- sbin子目录中有BIND主程序named、BIND进程管理工具rndc主程序等。

BIND的配置文件（/usr/local/bind9.9.3/etc/named.conf）主要由三个部分构成：

第一部分

```
options {
    // 数据默认放置的目录，如named.stats、named_dump.db等
    directory "/usr/local/bind9.9.3/data/";
    // 在自身缓存中未找到域名对应的记录时，转发请求到上级DNS服务器，这里以Google的公共DNS服务器为例。
    forwarders {
        8.8.8.8;
        8.8.4.4;
    };
    // 可不设定，代表全部接受
    allow-query {any;};
    // 允许在主DNS服务器变更时，主动通知从DNS服务器更新
    notify yes;
};
```

第一部分用于设置BIND的全局参数。

主服务器上的第二部分

```
// 劫持域xxx.com的DNS查询请求
zone "xxx.com" {
    // 指明当前服务器为DNS主服务器
    type master;
    // 指明域配置文件的路径
    file "master/zone.xxx.com";
    // 允许本区配置文件同步至特定的从DNS服务器
    allow-transfer { 192.168.2.193; };
};
```

第二部分用于实现DNS劫持（具体需要劫持什么域，请按需求设置），该部分可选，也可以存在多个。该部分的配置信息**在主、从服务器上是不同的**，从服务器上对应该部分的配置示例如下：

从服务器上的第二部分

```
//劫持对域“xxx.com”的DNS查询请求
zone "xxx.com" {
    // 指明当前服务器为DNS从服务器
    type slave;
    // 指明从哪个master同步配置文件
    masters {
        // 设置为主服务器的ip
        192.168.2.194;
    };
    // 指明从主DNS服务器同步过来的域配置文件的存放路径
    file "slaves/zone.xxx.com";
};
```

第三部分

```
key "rndc-key" {
        algorithm hmac-md5;
        secret "tahthw+V9UfRd0q8E63vPw==";
};

controls {
        inet 127.0.0.1 port 953
                allow { 127.0.0.1;} keys { "rndc-key"; };
};
```

第三部分是用于实现rndc对DNS进程的管理。该部分配置信息由如下过程得来（这里仅做说明，实际部署时不需重新生成）：

1. rndc自身也需要配置文件，其配置文件是这样生成的：

/usr/local/bind9.9.3/sbin/rndc-confgen -s 127.0.0.1 -r /dev/urandom > /usr/local/bind9.9.3/etc/rndc.conf，生成的配置文件内容示例如下：

/usr/local/bind9.9.3/etc/rndc.conf

```
# Start of rndc.conf
key "rndc-key" {
```

```
    algorithm hmac-md5;
    secret "tahthw+V9UfRd0q8E63vPw==";
};

options {
   default-key "rndc-key";
   default-server 127.0.0.1;
   default-port 953;
};
# End of rndc.conf

# Use with the following in named.conf, adjusting the allow list as needed:
# key "rndc-key" {
#   algorithm hmac-md5;
#   secret "tahthw+V9UfRd0q8E63vPw==";
# };
#
# controls {
#   inet 127.0.0.1 port 953
#           allow { 127.0.0.1; } keys { "rndc-key"; };
# };
# End of named.conf
```

2. 根据配置文件的注释说明将注释部分的配置（除去#）复制到named.conf的最后，重启named，然后就可以使用rndc命令来管理DNS服务器了。

“主服务器上的第二部分”中指定的域配置文件master/zone.xxx.com，其内容示例如下：

/usr/local/bind9.9.3/data/master/zone.xxx.com

```
@    IN    SOA    xxx.com.    root.xxx.com. (
                                130918011 ;Serial
                                10 ;Refresh
                                5 ;Retry
                                604800 ;Expire
                                3600) ;Minimum
     IN    NS     xxx.com.
     IN    A      192.168.2.194
www  IN    A      192.168.2.202
```

这段配置信息每一行的格式为 [**name**] [**ttl**] [**class**] [**type**] [**data**]。以下是每个字段的说明：

- name：可以是网域名称或是主机名称，如果不写的话表示与上一个设定相同。
- ttl：是数据要存活的时间(time to live)，也就是cache server 将保留在它的cache 中的时间。如果不写的话表示和SOA中的设定相同。
- class：指定网络的类型，这个字段应该都是使用IN 代表internet。
- type：设定该项数据的类型，例如：MX, A, CNAME, PTR, NS 等。
- data：就是实际设定数据的部份。

以下针对每一行来做说明。首先请看**@ IN SOA xxx.com. root.xxx.com. (…)**这部分，这个部分虽然跨过很多行，但实际上是一个设定项目，只是二个括号中间的设定可以被分为很多行以利阅读。这一行的设定是最基本的，也是最详尽的，在该行中，开头的**@**代表网域名称**xxx.com**，IN表示为internet的数据类型。**SOA**后面接的是xxx.com.，表示这台xxx.com.机器是xxx.com 网域中的主要名称服务器。而root@xxx.com表示管理者的Email是root@xxx.com。接下来看一下括号中的设定所代表的意义：

- Serial：这个设定的版本，这次修改的数字必须比上次的数字大，也就是每次该配置文件时，都要将这个数字提高，这样从服务器才会将数据同步更新。一般而言，我们会以日期加上几位的数字来表示，如2004040301表示2004年4月3日的第一次设定。
- Refresh：这个数字是次要名称服务器要多久和主要名称服务器比对数据并更新。
- Retry：如果比对失败，要在几秒后再向主要名称服务器查询。

- Expire：表示如果次要名称服务器一直连不上主要名称服务器，这笔数据要多久无法比对便失效。这个字段一样是以秒计算。
- Minimum：表示别的快取服务器可以将你的设定存放多久。

接下来的**IN NS xxx.com.**表示将xxx.com这个网域的DNS服务器是xxx.com.这台机器。这一行中，省略了name及ttl的字段，直接指定了class、type、及data。
IN A 192.168.2.194表示将xxx.com.这台机器的IP设为192.168.2.194。前面省略了主机名称，表示设定的是@的主机。A代表的就是指定address，就是将xxx.com.这台机器的IP地址设定为192.168.2.194。
www IN A 192.168.2.202表示将www.xxx.com的IP设定为192.168.2.202。你可以看到这里使用了字段name class type data。
“从服务器上的第二部分”中指定的域配置文件slaves/zone.xxx.com是在从服务器上BIND启动/重启或按一定的时间间隔或接收到主服务器通知后从主服务器上同步过来的，为二进制文件。

3.4 BIND使用

1. 启动BIND：`/usr/local/bind9.9.3/sbin/named &`
2. 重新载入BIND配置：`/usr/local/bind9.9.3/sbin/rndc reload`
3. 关闭BIND：`/usr/local/bind9.9.3/sbin/rndc stop`
4. 将BIND的统计数据写入文件*/usr/local/bind9.9.3/data/named.stats*：`/usr/local/bind9.9.3/sbin/rndc stats`
5. 将BIND的缓存信息写入文件*/usr/local/bind9.9.3/data/named_dump.db*：`/usr/local/bind9.9.3/sbin/rndc dumpdb`
6. 清空BIND缓存：`/usr/local/bind9.9.3/sbin/rndc flush`

更多rndc用法请查看rndc帮助文档：/usr/local/bind9.9.3/sbin/rndc -h