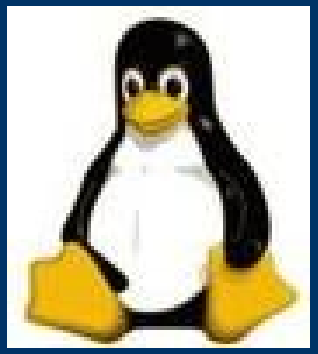
Linux 下的 Cluster 实现

啜立明



Linux 中
实现 HA 集群技术

Linux 下的 HA

- 双机技术

指由两台服务器运行某种同样的应用，为用户提供服务，当某一台出现问题时，用户的请求将由另一台服务器继续提供，从而实现高可用性。双机技术又被称为双机热备或双机容错

Linux 下的 HA

- 双机技术的实现不需要特定的硬件环境或者是操作系统 Kernel 的特定支持。因此仅需要双机 / 集群软件就可以实现

- 双机软件通过专用的信号传输通道，可以让两台服务器相互检测对方的状态，通过检测可得知对方如何。如对方出现问题可在第一时间作出反应

HA 容错运作过程

- Auto-Detect(自动检测)

通过两台主机所连接的线缆，经过负载的监听程序进行相互检测。其检测的内容有许多：

(1) 主机硬件
(2) 主机网络
(3) 主机操作系统
(4) 数据库引擎及其他应用程序
(5) 主机与磁盘整列连接线缆等

HA 容错运作过程

- Auto-Switch(自动切换)

　　如果某台主机确认对方出现故障，则将自动接手对方的工作来确保用户的请求可以得到及时处理

- Auto-Recover(自动恢复)

　　当故障主机修复完毕后可回归到生产系统中，通过一定配置可自动切换回以前状态继续工作

HA 的工作方式

- HA 的工作方式分为三种
- (1) 主从方式
- (2) 双机双工方式
- (3) 集群工作方式

HA 的工作方式

- 主从方式的工作原理

主机工作，备份机处于待命状态。当主机出现故障，备份机通过信号检测得知后将接管主机的一切工作，待主机回复正常后可以通过手工或自动配置切换到主机上运行。数据的一致性可通过其他技术解决

HA 的工作方式

- 双机双工方式

两台主机同时运行各自的服务工作且相互监督。当任何一台出现故障时，另一台会立即接管它的一切，保证工作的时效性。

HA 的工作方式

- 集群工作方式

多台主机一起工作，各自运行一个或多个服务，同时为每一个服务定义一个或多个备份主机。当主机出现故障时，备份主机将接管一切工作。

实现 Linux 下的 HA

- 实现 HA 的软件有许多种，其中包括

- 商业软件

(1)SteelEye 的 LifeKeeper for Linux
(2)Rose DataSystem 的 RoseHA
(3)Symantec 的 Verita

实现 Linux 下的 HA

- 实现 HA 的软件有许多种，其中包括

- 开源软件

(1)Heartbeat

(2)KeepAlived

实现 Linux 下的 HA

- HA

LHA 的目的就是提供一整套基于 Linux 的高可用性集群，其目标为 (RAS) 即：

Reliability(可靠性)

Availability(可用性)

Serviceablity(可服务性)

实现 Linux 下的 HA

Keepalived

keepalived 观其名可知，保持存活，在网络里面就是保持在线了，也就是所谓的高可用或热备，用来防止单点故障（单点故障是指一旦某一点出现故障就会导致整个系统架构的不可用）的发生。

keepalived 实现基础是 VRRP 协议

实现 Linux 下的 HA

VRRP

网络在设计的时候必须考虑到冗余容灾，包括线路冗余，设备冗余等，防止网络存在单点故障，那在路由器或三层交换机处实现冗余就显得尤为重要，在网络里面有个协议就是来做这事的，这个协议就是 VRRP 协议， Keepalived 就是巧用 VRRP 协议来实现高可用性 (HA) 的

实现 Linux 下的 HA

Keepalived 原理
keepalived 也是模块化设计，不同模块复杂不同的功能，下面是 keepalived 的组件
core check vrrp libipfwc libipvs-2.4 libipvs-2.6

core ：是 keepalived 的核心，复杂主进程的启动和维护，全局配置文件的加载解析等
check ：负责 healthchecker(健康检查)，包括了各种健康检查方式，以及对应的配置的解析包括 LVS 的配置解析

实现 Linux 下的 HA

vrrp ： VRRPD 子进程， VRRPD 子进程就是来实现 VRRP 协议的

libipfwc ： iptables(ipchains) 库，配置 LVS 会用到

libipvs* ：配置 LVS 会用到

注意， keepalived 和 LVS 完全是两码事，只使它们相互配合。

实现 Linux 下的 HA

keepalived 原理

keepalived 启动后会有三个进程：

父进程：内存管理，子进程管理等等

子进程： VRRP 子进程

子进程： healthchecker 子进程

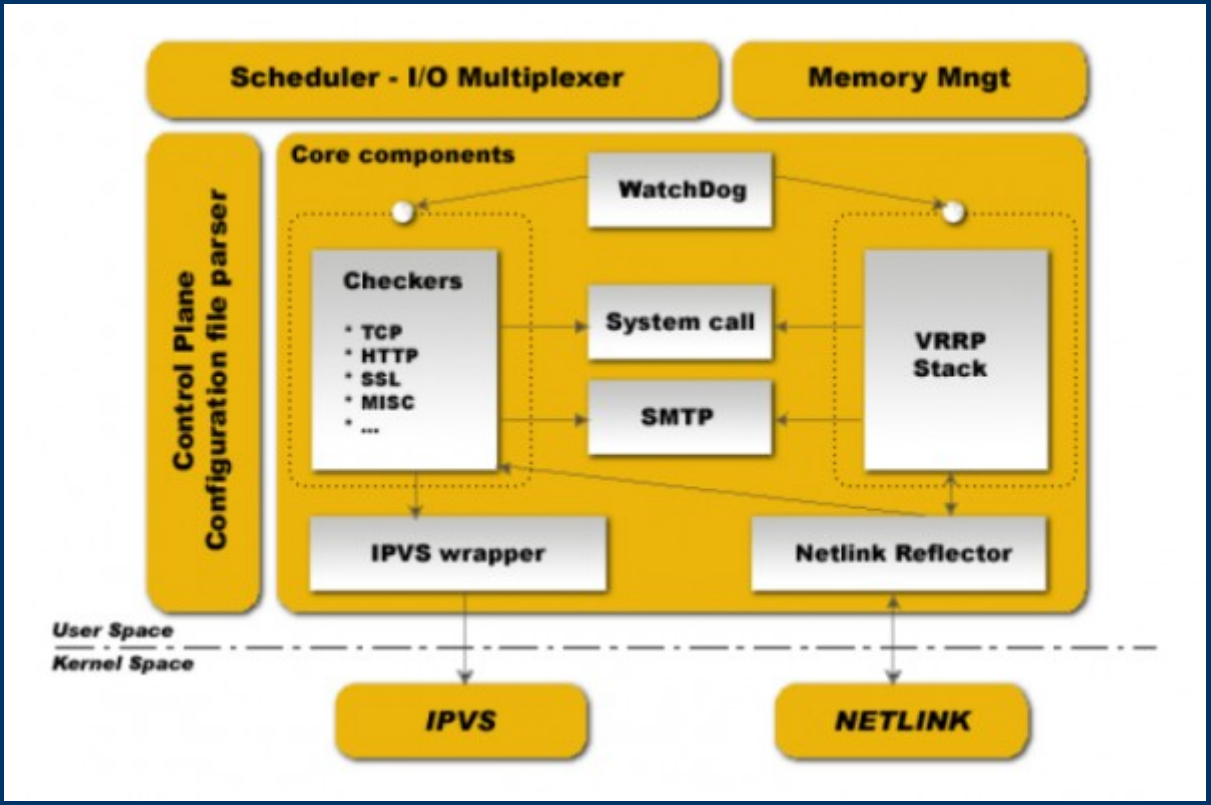
实现 Linux 下的 HA

keepalived 原理

两个子进程都被系统 WatchDog 看管，两个子进程各自复杂自己的事， healthchecker 子进程复杂检查各自服务器的健康程度，例如 HTTP ，LVS 等等，如果 healthchecker 子进程检查到 MASTER 上服务不可用了，就会通知本机上的兄弟 VRRP 子进程，让他删除通告，并且去掉虚拟 IP ，转换为 BACKUP 状态

实现 Linux 下的 HA

keepalived 原理



实现 Linux 下的 HA

- 获取 keepalived
- 在安装之前请确保系统已经安装了 Python 程序

- Debian/Ubuntu

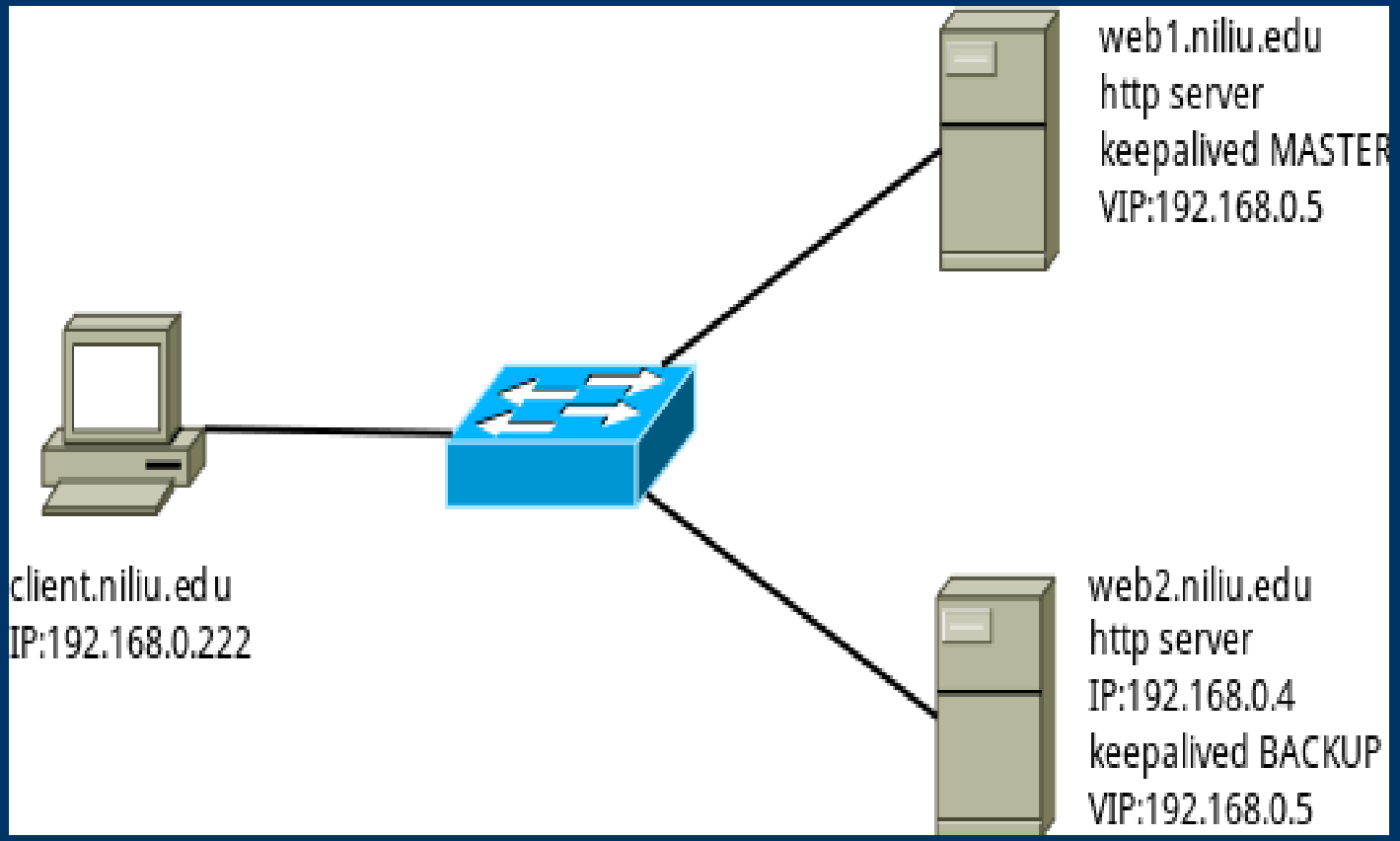
#apt-get install keepalived

- RedHat/CentOS

#yun install keepalived

实现 Linux 下的 HA

- (1) 试验拓扑图



实现 Linux 下的 HA

- 目的：

  http 提供 HA(高可用性)

- 实验基础要求

  1) 配置各自设备的 IP 地址

  2) 服务器端配置完成 httpd 服务

  (1) web1 的 index.html 内容为” web1 server”

  (2) web2 的 index.html 内容为” web2 server”

实现 Linux 下的 HA

- 实验基础要求 (续)

  3) 客户端可以成功访问 web1.niliu.edu 与 web2.niliu.edu 的 http 服务并能够显示相关的信息

实现 Linux 下的 HA

配置 keepalived
web1.niliu.edu 上配置 keepalived
#cd /etc/keepalived
#cp keepalived.conf keepalived.conf.bak
#vim keepalived.conf
//* 注释说明
! Configuration File for keepalived

//* 定义全局配置
global_defs { <- 定义全局配置

实现 Linux 下的 HA

```
//* 当 keepalived 发生切换时给指定的邮箱发送邮件
   notification_email {

//* 指定接收通知的邮箱
   root@localhost
  }

//* 设置发送的源地址为谁
   notification_email_from root@localhost
```

实现 Linux 下的 HA

```
//* 表示发送 email 时使用的 smtp 服务器地址，可用本地的 sendmail 来实现
    smtp_server 127.0.0.1

//* 指定 smtp 链接超时时间
    smtp_connect_timeout 30

//* 表示设备名称
    router_id web1.niliu.edu
} <- 结束全局配置
```

实现 Linux 下的 HA

VRRP 实例设定

VRRP:

网络在设计的时候必须考虑到冗余容灾，包括线路冗余，设备冗余等，防止网络存在单点故障，那在路由器或三层交换机处实现冗余就显得尤为重要，在网络里面有个协议就是来做这事的，这个协议就是 VRRP 协议， Keepalived 就是巧用 VRRP 协议来实现高可用性 (HA) 的

实现 Linux 下的 HA

VRRP 实例设定
 //* 定义 VRRP 实例名称为 VI_1
vrrp_instance VI_1 {

//* 设定本机为 MASTER(主)
state MASTER

//* 指定实例所需要绑定的网卡，以便于 VIP 使用
Interface enp0s3

实现 Linux 下的 HA

VRRP 实例设定
//* 设定 VRID ，相同的 VRID 为一个组，同组的 VRID 将决定多播的 MAC 地址
virtual_router_id 51

//* 设定优先级，数字越高越优先
priority 100

//* 设定心跳广播间隔 (秒)
advert_int 1

实现 Linux 下的 HA

VRRP 实例设定

```
 //* 设定设备之间的认证
authentication {

//* 认证方式为密码认证
  auth_type PASS

//* 密码为 1111
    auth_pass 1111
}
```

实现 Linux 下的 HA

设定 VIP

```
  virtual_ipaddress {
    192.168.0.5
  }
}
```

实现 Linux 下的 HA

web2.niliu.edu 配置 keepalived

```
#cat /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
    notification_email {
        root@localhost
    }
    notification_email_from root@localhost
    smtp_server 127.0.0.1
```

实现 Linux 下的 HA

web2.niliu.edu 配置 keepalived

```
    smtp_connect_timeout 1
    router_id web2.niliu.edu
}

vrrp_instance VI_1 {
    state BACKUP
    interface enp0s3
    virtual_router_id 51
    priority 50
```

实现 Linux 下的 HA

web2.niliu.edu 配置 keepalived

```
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }

    virtual_ipaddress {
        192.168.0.5
    }
}
```

实现 Linux 下的 HA

启动 keepalived 服务 (web1 与 web2)

#systemctl enabled keepalived

#systemctl start keepalived

查看 VIP 已经作用在 web1.niliu.edu 上

#tail -f /var/log/messages

客户端测试

客户端用浏览器访问 VIP, 可看到 web1 上的内容

实现 Linux 下的 HA

开启 web2 的 messages

#tail -f /var/log/messages

停止 web1 上的 keepalived 服务

#systemctl stop keepalived

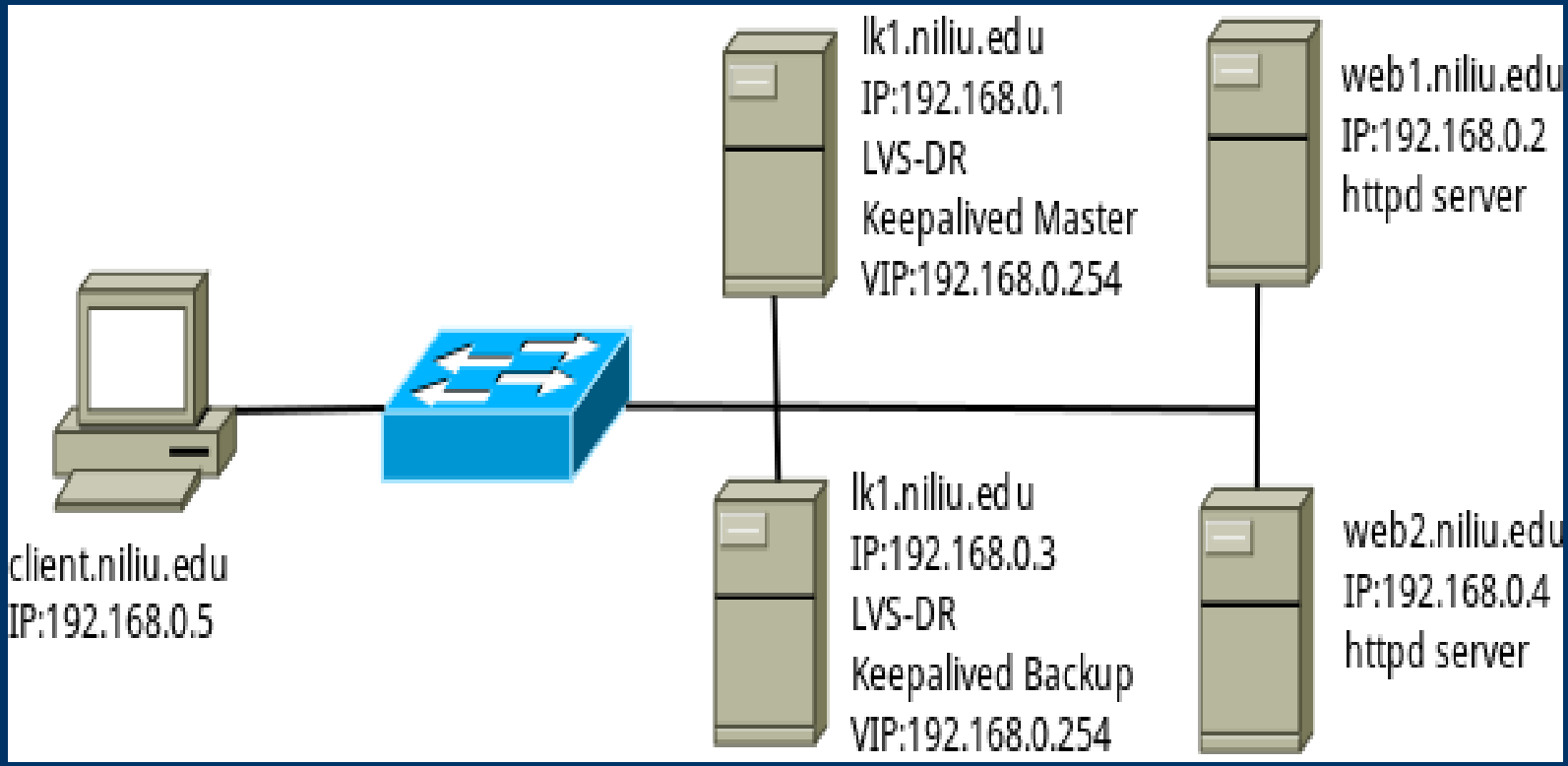
观察 VIP 作用在 web2.niliu.edu 上

客户端测试

客户端用访问 VIP, 可看到 web2 上的内容

实现 Linux 下的 HA

LVS+Keepalived
实验环境



实现 Linux 下的 HA

1. 各自配置完成 IP ，且可以通信

2. 安装 web1/web2 的 httpd 服务器

3. 定义 web1/web2 的 httpd 内容
   web1:web1.niliu.edu
   web2:web2.niliu.edu

实现 Linux 下的 HA

4. 在 lk1/lk2 编写 LVS-DR 规则

#vim lvs-dr.sh

```
#! /bin/bash
ipvsadm -C
ipvsadm -A -t 192.168.0.254:80 -s rr
ipvsadm -a -t 192.168.0.254:80 -r 192.168.0.3:8
0 -g
ipvsadm -a -t 192.168.0.254:80 -r 192.168.0.4:8
0 -g
```

实现 Linux 下的 HA

5. 配置 web1/web2

#ifconfig lo:0 192.168.0.254 netmask 255.255.255.255 up

#route add -host 192.168.0.254 dev lo:0

#echo 1 > /proc/sys/net/ipv4/conf/lo/arp_ignore

#echo 2 > /proc/sys/net/ipv4/conf/lo/arp_announce

实现 Linux 下的 HA

5. 配置 web1/web2

#echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore

#echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce

#systemctl restart httpd

实现 Linux 下的 HA

6. lk1.niliu.edu 配置 keepalived

```
#cat /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
    notification_email {
        root@localhost
    }
    notification_email_from root@localhost
    smtp_server 127.0.0.1
```

实现 Linux 下的 HA

```
    smtp_connect_timeout 30
    router_id lk1.niliu.edu
}

vrrp_instance VI_1 {
    state MASTER
    interface enp0s3
    virtual_router_id 51
    priority 100
```

实现 Linux 下的 HA

```
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }

    virtual_ipaddress {
        192.168.0.254
    }
}
```

实现 Linux 下的 HA

7. lk2.niliu.edu 配置 keepalived

```
#cat /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
   notification_email {
     root@localhost
   }
   notification_email_from root@localhost
   smtp_server 127.0.0.1
```

实现 Linux 下的 HA

```
    smtp_connect_timeout 30
    router_id lk2.niliu.edu
}

vrrp_instance VI_1 {
    state BACKUP
    interface enp0s3
    virtual_router_id 51
    priority 50
```

实现 Linux 下的 HA

```
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }

    virtual_ipaddress {
        192.168.0.254
    }
}
```

实现 Linux 下的 HA

8. 启动 lk1/lk2 关联服务
   #chmod 700 ~/lvs-dr.sh
   #./lvs-dr.sh
   #systemctl start keepalived

9. 观察 lk1/lk2 下 keepalived 信息
   #tail -f /var/log/messages

10. 客户端测试

实现 Linux 下的 HA

关于 keepalived 的 LVS

通过配置 keepalived 可以直接实现 LVS 功能。好处在于维护方便。

keepalived 有其自己的语法格式。

实现 Linux 下的 HA

keepalived 的 LVS 配置格式

```
    virtual_server VIP port {
        Options

        real_server  IP port {
            Options
        }
}
```

实现 Linux 下的 HA

keepalived 的 LVS 格式说明

//* 定义 VIP 所监听的服务，如：
virtual_server 192.168.0.254 80 {

//* 其下有一些常用选项
//* 设置服务轮询时间间隔
delay_loop6

实现 Linux 下的 HA

keepalived 的 LVS 格式说明

//* 设置 LVS 的调度算法
lb_algo rr|wrr|lc|wlc|lblc|sh|dh

//* 设置 LVS 集群模式
lb_kind NAT|DR|TUN

//* 会话保持时间（秒为单位）
persistence_timeout 120

实现 Linux 下的 HA

keepalived 的 LVS 格式说明

//*LVS 会话保持粒度， ipvsadm 中的 -M 参数，默认是 0xffffffff ，即每个客户端都做会话保持
persistence_granularity <NETMASK>

//* 健康检查用的是 TCP 还是 UDP
protocol TCP
//* 暂停健康检测 ha_suspend

实现 Linux 下的 HA

keepalived 的 LVS 格式说明

//*HTTP_GET 做健康检查时，检查的 web 服务器的虚拟主机（即 host ：头）
virtualhost <string>

//* 当所有 realserver 不可用时，可把请求发送至指定服务器上
sorry_server <IPADDR> <PORT>

实现 Linux 下的 HA

keepalived 的 LVS 格式说明

//* 配置 real_server, 如
real_server 192.168.0.2 80 {

//* 设置权重 ,0 为失效
weight 1

实现 Linux 下的 HA

keepalived 的 LVS 格式说明

//* 表示在节点失败后，将其权重设置成 0 ，而不是从 IPVS 中删除
inhibit_on_failure

实现 Linux 下的 HA

keepalived 的 LVS 格式说明

```
//* 健康检查
HTTP_GET    <-- 健康检查方式
{
url {              <-- 要坚持的 URL ，可以有多个
path /             <-- 具体路径
digest <STRING> <-- 值
status_code 200  <-- 返回状态码
}
```

实现 Linux 下的 HA

keepalived 的 LVS 格式说明

//*digest 值的取得
#genhash -s 192.168.2.188 -p 80 -u /index.htm

-s: 指定远程的 IP
-p: 指定远程的端口
-u: 指定远程的资源

实现 Linux 下的 HA

keepalived 的 LVS 格式说明

```
//*TCP 健康检测
TCP_CHECK {
        connect_timeout 3  # 表示 3 秒无响应超时
        nb_get_retry 3# 表示重试次数
        delay_before_retry 3   # 表示重试间隔
       }
```

实现 Linux 下的 HA

keepalived 的 LVS 实现

1. 在 lk1 的 keepalived.conf 下增加如下行

```
virtual_server 192.168.0.254 80 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    persistence_timeout 50
    protocol TCP
```

实现 Linux 下的 HA

```
real_server 192.168.0.2 80 {
    weight 1
    TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 2
        }
}
```

实现 Linux 下的 HA

```
real_server 192.168.0.4 80 {
    weight 1
    TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 2
        }
}
} <- 结束 virtual_server
```

实现 Linux 下的 HA

2. 在 lk2 的 keepalived.conf 下增加如下行

```
virtual_server 192.168.0.254 80 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    persistence_timeout 50
    protocol TCP
```

实现 Linux 下的 HA

```
real_server 1921.68.0.2 80 {
    weight 1
    TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 2
        }
}
```

实现 Linux 下的 HA

```
real_server 1921.68.0.4 80 {
    weight 1
    TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 2
        }
}
} <- 结束 virtual_server
```

实现 Linux 下的 HA

3. 重启 lk1/lk2 的 keepalived 服务
4. 可用 ipvsadm 查看信息
5. 客户端测试

HA 集群实现

- 使用 Keepalived 实现 Linux 下的 HA
- 试验目的 : 掌握 Keepalived 实现 HA 集群
- 试验人员 : 个人
- 所需要计算机设备 : 至少 3-5 台计算机
- 试验时间 :30 分钟

Linux 下的 Cluster 实现

结　　束

master.chuai@gmail.com

304630723

152990419