7_Cluster02keepalived 热备 keepalived+lvsHAPROXY

一 keepalived

1.1 keepalived 作用

实现高可用集群;为 LVS 设计,专门监控各服务器节点的状态;加入了 VRRP 功能,防止单点故障

1.2 keepalived 运行原理

keepalived 检测每个服务点节点状态;服务器节点异常或工作出现故障,keepalived 将故障节点从集群中剔除,故障节点恢复后 keepalived 再将其加入到集群系统中;工作自动完成,无需人工干预

1.3 keepalived 三个功能(模块)

1.3.1 vrrp: 虚拟路由冗余协议 Virtual Router Redundancy Protocol

1.3.2 自动配置 LVS(ipvasdm)

1.3.3 健康检查

方法 1:　　TCP_CHECK {

　　时间

　　} #只管 80 端口是否开启

方法 2:　　HTTP_GET {

　　url {

　　path /　#/表示网页根目录

　　digest XXXXXXXX　#XXX 为该/下网页文件的哈希值

```
            }
        }  #检查 80 端口及检查 path 定义的页面文件
方法 3:     SSL_GET {
            url {
            path /
            digest XXXXXXXX
        }  #原理类同于 HTTP_GET,检查 443 端口,SSL 要加密
```

二 keepalived 案例：Keepalived 高可用服务器

2.1 问题

准备三台 Linux 服务器，两台做 Web 服务器，并部署 Keepalived 高可用软件，一台作为客户端主机，实现如下功能：

使用 Keepalived 实现 web 服务器的高可用(高可用,非负载均衡)

Web 服务器 IP 地址分别为 192.168.4.100 和 192.168.4.200

Web 服务器的浮动 VIP 地址为 192.168.4.80

客户端通过访问 VIP 地址访问 Web 页面

2.2 方案

使用 3 台虚拟机，2 台作为 Web 服务器，并部署 Keepalived、1 台作为客户端，拓扑结构如图-1 所示，主机配置如表-1 所示。

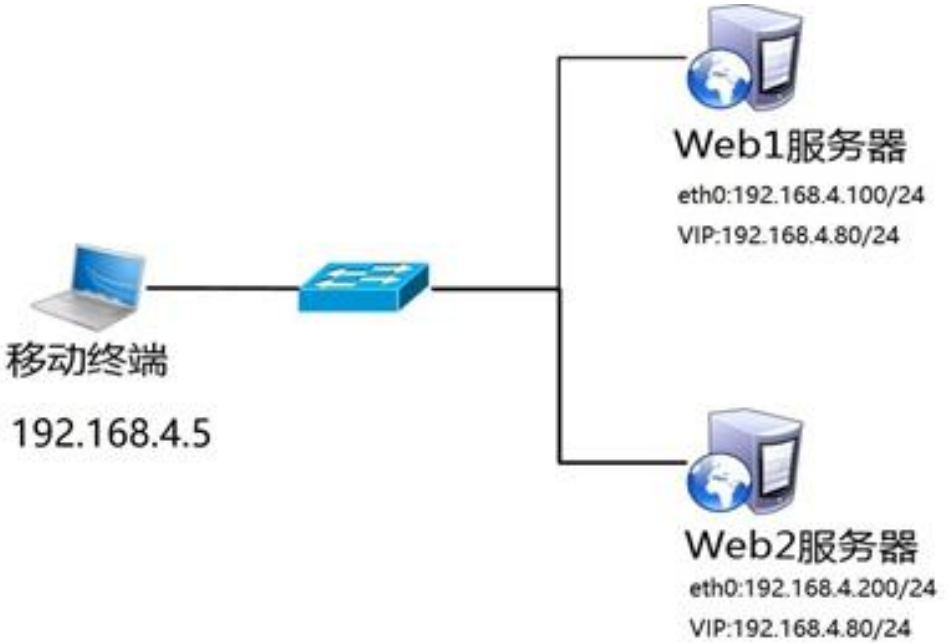

图-1

表-1

主机名	网络配置
proxy（扮演客户端的角色）	eth0:192.168.4.5
web1	eth0:192.168.4.100 VIP:192.168.4.80(keepalive 会自动配置)
web2	eth0:192.168.4.200 VIP:192.168.4.80(keepalive 会自动配置)

2.3 步骤

步骤一：配置网络环境（如果在前面课程已经完成该配置，可以忽略此步骤）

1）设置 Web1 web2 服务器网络参数、并配置 Web 服务(http)

```
web1 ~]# nmcli connection modify eth0 ipv4.method manual
ipv4.addresses 192.168.4.100/24 connection.autoconnect yes
web1 ~]# nmcli connection up eth0
web1 ~]# yum -y install httpd
```

```
web1 ~]# echo "192.168.4.100(200)" > /var/www/html/index.html
web1 ~]# systemctl restart httpd
```

2）配置 proxy 主机的网络参数（如果已经设置，可以忽略此步骤）

```
proxy ~]# nmcli connection modify eth0 ipv4.method manual
ipv4.addresses 192.168.4.5/24 connection.autoconnect yes
proxy ~]# nmcli connection up eth0
```

步骤二：web1 2 安装 Keepalived 软件

```
web1 2 ~]# yum install -y keepalived
```

步骤三：部署 Keepalived 服务

1）修改 web1 服务器 Keepalived 配置文件

```
web1 ~]# vim /etc/keepalived/keepalived.conf
global_defs {
     notification_email {
          admin@tarena.com.cn  #设置报警收件人邮箱
     }
     notification_email_from ka@localhost  #设置发件人
     smtp_server 127.0.0.1  #定义邮件服务器
     smtp_connect_timeout 30
     router_id  web1  #设置路由 ID 号（实验需要修改）
}
```

```
vrrp_instance VI_1 {
    state MASTER  #主服务器为MASTER（备服务器需要修改为BACKUP）
    interface eth0  #定义网络接口
    virtual_router_id 51  #任意,但主备服务器VRID号必须一致
    priority 100  #服务器优先级,优先级高优先获取VIP
    advert_int 1  #优先级比较间隔,单位秒
    authentication {
        auth_type pass
        auth_pass 1111  #任意,主备服务器密码必须一致
    }
    virtual_ipaddress {  #谁是主服务器谁获得该VIP（实验需要修改）
        192.168.4.80
    }
}    #剩余全部删除(33行以下全部删除,本案例中用不上33行以下的模块)
```

2）修改web2服务器Keepalived配置文件

```
web2 ~]# vim /etc/keepalived/keepalived.conf
global_defs {
    notification_email {
        admin@tarena.com.cn  #设置报警收件人邮箱
    }
```

```
    notification_email_from ka@localhost  #设置发件人
    smtp_server 127.0.0.1  #定义邮件服务器
    smtp_connect_timeout 30
    router_id  web2  #设置路由 ID 号
}
vrrp_instance VI_1 {
    state BACKUP  #备服务器为 BACKUP（实验需要修改）
    interface eth0  #定义网络接口
    virtual_router_id 51  #主辅 VRID 号必须一致
    priority 50  #服务器优先级（实验需要修改）
    advert_int 1
    authentication {
        auth_type pass
        auth_pass 1111  #主辅服务器密码必须一致
    }
    virtual_ipaddress {  #谁是主服务器谁配置 VIP（实验需要修改）
        192.168.4.80
    }
}
```

3）we1 web2 启动服务 keepalived

web1 2 ~]# systemctl start keepalived

4）配置防火墙和 SELinux(web1 和 web2)

启动 keepalived 会自动添加一个 drop 的防火墙规则，需要清空!

web1 2 ~]# **iptables -F** #Delete all rules in chain or all chains

web1 2 ~]# **setenforce 0**

步骤四：测试

1）登录两台 Web 服务器查看 VIP 信息

web1 ~]# ip addr show eth0 [ip a s]

web2 ~]# ip addr show eth0

2) 客户端访问

客户端使用 curl 命令连接 http://192.168.4.80，查看 Web 页面；关闭 Web1 服务器的网卡，客户端再次访问 http://192.168.4.80，验证是否可以正常访问服务。

3)日志文件

若需要排错,查看/var/log/messages

三 案例：Keepalived+LVS 服务器

3.1 问题

使用 Keepalived 为 LVS 调度器提供高可用功能，防止调度器单点故障，为用户提供 Web 服务：

LVS1 调度器真实 IP 地址为 192.168.4.5

LVS2 调度器真实 IP 地址为 192.168.4.6

服务器 VIP 地址设置为 192.168.4.15

真实 Web 服务器地址分别为 192.168.4.100、192.168.4.200

使用加权轮询调度算法，真实 web 服务器权重不同

3.2 方案

使用 5 台虚拟机，1 台作为客户端主机、2 台作为 LVS 调度器、2 台作为 Real Server，

实验拓扑环境结构如图-2 所示，基础环境配置如表-2 所示。

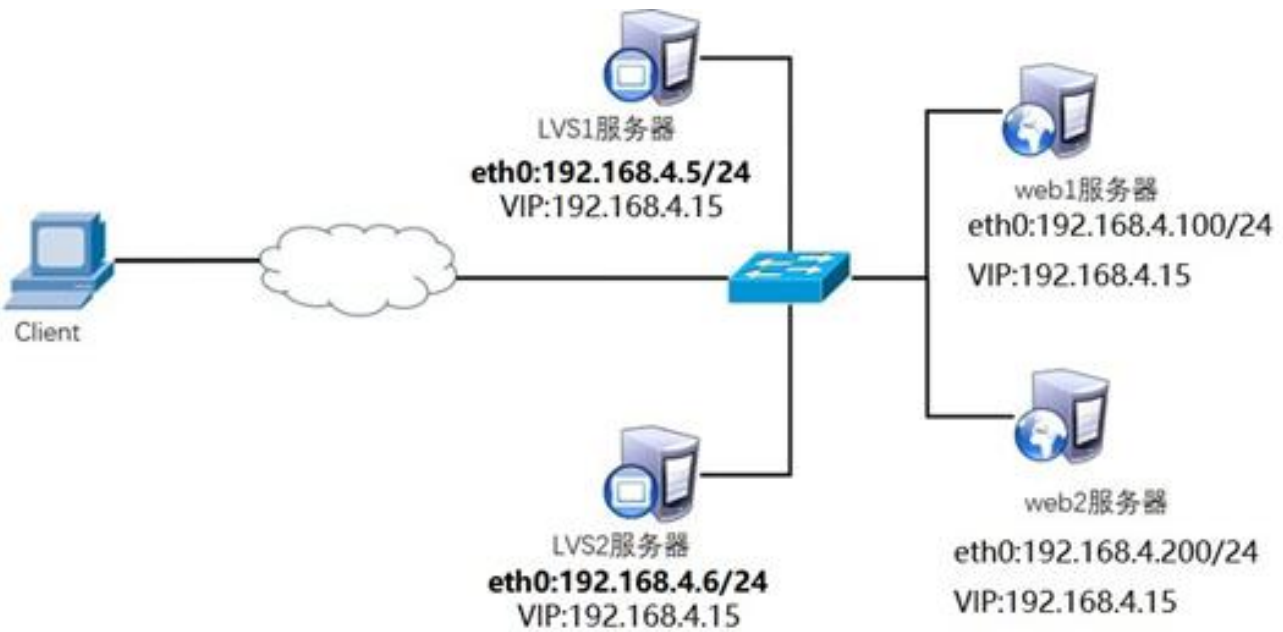

图-3

表-2

主机名	网络配置
client	eth0:192.168.4.10/24
proxy1	eth0:192.168.4.5/24
proxy2	eth0:192.168.4.6/24
web1	eth0:192.168.4.100/24
web2	eth0:192.168.4.200/24

注意：所有主机都需要配置 IP 地址与有效的 YUM 源。

3.3 步骤

步骤一：配置网络环境

1）设置 Web1 web2 服务器的网络参数

```
web1 web2 ~]# nmcli connection modify eth0 ipv4.method manual \
ipv4.addresses 192.168.4.100(200)/24 connection.autoconnect yes
web1 web2 ~]# nmcli connection up eth0
```

接下来给 web1 web2 配置 VIP 地址

注意：这里的子网掩码必须是 32（也就是全 255），网络地址与 IP 地址一样，广播地址与 IP 地址也一样。

```
web1 web2 ~]# cd /etc/sysconfig/network-scripts/
web1 web2 ~]# cp ifcfg-lo{,:0}
web1 web2 ~]# vim ifcfg-lo:0
DEVICE=lo:0
IPADDR=192.168.4.15
NETMASK=255.255.255.255
NETWORK=192.168.4.15
BROADCAST=192.168.4.15
ONBOOT=yes
NAME=lo:0
```

注意：这里因为 web1 也配置与调度器一样的 VIP 地址，默认肯定会出现地址冲突。

写入这四行的主要目的就是访问 192.168.4.15 的数据包，只有调度器会响应，其他主机都不做任何响应。

```
web1 web2 ~]# vim /etc/sysctl.conf
#手动写入如下 4 行内容
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.lo.arp_ignore = 1
net.ipv4.conf.lo.arp_announce = 2
net.ipv4.conf.all.arp_announce = 2
#当有 arp 广播问谁是 192.168.4.15 时，本机忽略该 ARP 广播，不做任何回应
#本机不要向外宣告自己的 lo 回环地址是 192.168.4.15
```

重启网络服务，设置防火墙与 SELinux

```
web1 web2 ~]# systemctl stop NetworkManager
web1 web2 ~]# systemctl disable NetworkManager
web1 web2 ~]# systemctl restart network
web1 web2 ~]# ifconfig
web1 ~]# systemctl stop firewalld
web1 ~]# setenforce 0
```

2）配置 proxy1 主机的网络参数(不配置 VIP，由 keepalvied 自动配置)

```
proxy1 ~]# nmcli connection modify eth0 ipv4.method manual \
```

```
ipv4.addresses 192.168.4.5/24 connection.autoconnect yes
proxy1 ~]# nmcli connection up eth0
```

3）配置 proxy2 主机的网络参数(不配置 VIP，由 keepalvied 自动配置)

注意：按照前面的课程环境，默认没有该虚拟机，需要重新建一台虚拟机 proxy2。

```
proxy2 ~]# nmcli connection modify eth0 ipv4.method manual \
ipv4.addresses 192.168.4.6/24 connection.autoconnect yes
proxy2 ~]# nmcli connection up eth0
```

步骤二：配置后台 web 服务

1）安装软件，自定义 Web 页面（web1 和 web2 主机）

```
web1 web2 ~]# yum -y install httpd
web1 web2 ~]# echo "192.168.4.100(200)" >
/var/www/html/index.html
```

2）启动 Web 服务器软件(web1 和 web2 主机)

```
web1 web2 ~]# systemctl start httpd ; systemctl enable httpd
```

步骤三：调度器安装 Keepalived 与 ipvsadm 软件

注意：两台 LVS 调度器执行相同的操作（如何已经安装软件，可用忽略此步骤）。

安装软件

```
proxy1 ~]# yum install -y keepalived
proxy1 ~]# systemctl enable keepalived
proxy1 ~]# yum install -y ipvsadm
```

```
proxy1 ~]# ipvsadm -C
proxy2 ~]# yum install -y keepalived
proxy2 ~]# systemctl enable keepalived
proxy2 ~]# yum install -y ipvsadm
proxy2 ~]# ipvsadm -C
```

步骤四：部署 Keepalived 实现 LVS-DR 模式调度器的高可用

1）LVS1 调度器设置 Keepalived，并启动服务

```
proxy1 ~]# vim /etc/keepalived/keepalived.conf
global_defs {
     notification_email {
     admin@tarena.com.cn  #设置报警收件人邮箱
     }
     notification_email_from ka@localhost  #设置发件人
     smtp_server 127.0.0.1  #定义邮件服务器
     smtp_connect_timeout 30
     router_id  lvs1  #设置路由 ID 号(实验需要修改)
}
vrrp_instance VI_1 {
     state MASTER  #主服务器为 MASTER
     interface eth0  #定义网络接口
```

```
    virtual_router_id 51  #主辅 VRID 号必须一致
    priority 100  #服务器优先级
    advert_int 1
    authentication {
        auth_type pass
        auth_pass 1111  #主辅服务器密码必须一致
    }
    virtual_ipaddress {  #配置 VIP（实验需要修改）
        192.168.4.15
    }
}
virtual_server 192.168.4.15 80 {
#设置 ipvsadm 的 VIP 规则（实验需要修改）
    delay_loop 6
    lb_algo rr  #设置 LVS 调度算法为 RR
    lb_kind DR  #设置 LVS 的模式为 DR（实验需要修改）
    #persistence_timeout 50  #50 秒内找相同服务器（实验需要注释）
#注意这样的作用是保持连接，开启后，客户端在一定时间内始终访问相同服务器
    protocol TCP
    real_server 192.168.4.100 80 {
```

```
        #设置后端 web 服务器真实 IP（实验需要修改）
              weight 1  #设置权重为 1
              TCP_CHECK {  #对后台 real_server 做健康检查（实验需要修改）
              connect_timeout 3  #连接超时时间为 3 秒
              nb_get_retry 3  #检测 3 次
              delay_before_retry 3  #检测间隔
              }
        }
        real_server 192.168.4.200 80 {
#设置后端 web 服务器真实 IP（实验需要修改）
              weight 2  #设置权重为 1
              TCP_CHECK {  #对后台 real_server 做健康检查（实验需要修改）
                    connect_timeout 3
                    nb_get_retry 3
                    delay_before_retry 3
              }
        }
}
proxy1 ~]# systemctl start keepalived
proxy1 ~]# ipvsadm -Ln                    #查看 LVS 规则
```

```
proxy1 ~]# ip a  s                              #查看 VIP 配置
proxy1 ~]# iptables -F
```

2）LVS2 调度器设置 Keepalived

```
proxy2 ~]# vim /etc/keepalived/keepalived.conf
global_defs {
     notification_email {
          admin@tarena.com.cn  #设置报警收件人邮箱
     }
  notification_email_from ka@localhost  #设置发件人
  smtp_server 127.0.0.1  #定义邮件服务器
  smtp_connect_timeout 30
  router_id  lvs2  #设置路由 ID 号（实验需要修改）
}
vrrp_instance VI_1 {
     state BACKUP  #从服务器为 BACKUP（实验需要修改）
     interface eth0  #定义网络接口
     virtual_router_id 51  #主辅 VRID 号必须一致
     priority 50  #服务器优先级（实验需要修改）
      advert_int 1
     authentication {
```

```
    auth_type pass
    auth_pass 1111  #主辅服务器密码必须一致
    }
    virtual_ipaddress {  #设置 VIP（实验需要修改）
        192.168.4.15
    }
}
virtual_server 192.168.4.15 80 {  #自动设置 LVS 规则（实验需要修改）
    delay_loop 6
    lb_algo  rr  #设置 LVS 调度算法为 RR
    lb_kind DR  #设置 LVS 的模式为 DR（实验需要修改）
    #persistence_timeout 50  #（实验需要注释）
#注意这样的作用是保持连接，开启后，客户端在一定时间内始终访问相同服务器
    protocol TCP
    real_server 192.168.4.100 80 {
    #设置后端 web 服务器的真实 IP（实验需要修改）
    weight 1  #设置权重为 1
        TCP_CHECK {  #对后台 real_server 做健康检查（实验需要修改）
            connect_timeout 3
            nb_get_retry 3
```

```
                delay_before_retry 3
            }
        }
        real_server 192.168.4.200 80 {
#设置后端 web 服务器的真实 IP（实验需要修改）
            weight 2  #设置权重为 1
            TCP_CHECK {  #对后台 real_server 做健康检查（实验需要修改）
                connect_timeout 3
                nb_get_retry 3
                delay_before_retry 3
            }
        }
}
proxy2 ~]# systemctl start keepalived
proxy2 ~]# ipvsadm -Ln                  #查看 LVS 规则
proxy2 ~]# ip  a   s                    #查看 VIP 设置
proxy2 ~]# iptables -F
```

步骤五：客户端测试

客户端使用 curl 命令反复连接 http://192.168.4.15，查看访问的页面是否会轮询到不同的后端真实服务器。

四 案例：配置 HAProxy 负载平衡集群(HAproxy 高可用代理)

4.1 问题

准备 4 台 Linux 服务器，两台做 Web 服务器，1 台安装 HAProxy，1 台做客户端，实现如下功能：客户端访问 HAProxy，HAProxy 分发请求到后端 Real Server

开启 HAProxy 监控页面，及时查看调度器状态

设置 HAProxy 为开机启动

4.2 方案

使用 4 台虚拟机，1 台作为 HAProxy 调度器、2 台作为 Real Server、1 台作为客户端，拓扑结构如图-3 所示，具体配置如表-3 所示。

图-3

表-3

主机名	网络配置
client	eth0:192.168.4.10/24
proxy	eth0:192.168.4.5/24 eth1:192.168.2.5/24
web1	eth1:192.168.2.100/24
web2	eth1:192.168.2.200/24

4.3 步骤

注意事项：

将前面实验 VIP、LVS 等实验的内容清理干净！！！！！！

删除所有设备的 VIP，清空所有 LVS 设置，关闭 keepalived！！！

web1 web2 关闭多余的网卡与 VIP，配置本地真实 IP 地址。

web1 web2 ~]# **ifdown lo:0** #清理 VIP(或删除 ifcfg-lo:0 文件)

web1 web2 ~]# nmcli connection modify **eth1** ipv4.method manual

ipv4.addresses 192.168.2.100(200)/24 connection.autoconnect yes

web1 web2 ~]# **nmcli connection up eth1**

proxy 关闭 keepalived 服务，清理 LVS 规则。

proxy ~]# systemctl stop keepalived #关闭 keepalived

proxy ~]# systemctl disable keepalived #禁用 keepalived

proxy ~]# ipvsadm -C #清除所有 LVS 设置

proxy ~]# nmcli connection modify **eth0** ipv4.method manual

ipv4.addresses 192.168.4.5/24 connection.autoconnect yes

proxy ~]# nmcli connection up eth0

proxy ~]# nmcli connection modify **eth1** ipv4.method manual \

ipv4.addresses 192.168.2.5/24 connection.autoconnect yes

proxy ~]# nmcli connection up eth1

步骤一：配置后端 Web 服务器

设置两台后端 Web1 web2 服务（如果已经配置完成，可用忽略此步骤）

```
web1 ~]# yum -y install httpd

web1 ~]# systemctl start httpd

web1 ~]# echo "192.168.2.100(200)" > /var/www/html/index.html
```

步骤二：部署 HAProxy 服务器

1）配置网络，安装软件

```
haproxy ~]# yum -y install haproxy
```

2）修改配置文件

```
haproxy ~]# vim /etc/haproxy/haproxy.cfg

global  #全局设置端

log 127.0.0.1 local2  #[err warning info debug]

chroot /usr/local/haproxy

pidfile /var/run/haproxy.pid  #haproxy 的 pid 存放路径

maxconn 4000#最大连接数，默认 4000,后续的所有实际并发连接的和不超过 4000

user haproxy  #用户

group haproxy  #组

daemon  #创建 1 个进程进入 deamon 模式运行

defaults  #默认设置段

mode http  #mod 默认为 http { tcp|http|health }
```

option dontlognull #不记录健康检查的日志信息

option httpclose #每次请求完毕后主动关闭 http 通道

option httplog #日志类别 http 日志格式

option forwardfor #后端服务器可以从 Http Header 中获得客户端 ip

option redispatch #serverid 服务器挂掉后强制定向到其他健康服务器

timeout connect 10000 #如果 backend 没有指定，默认为 10s

timeout client 300000 #客户端连接超时

timeout server 300000 #服务器连接超时

maxconn 3000 #最大连接数

63 行以下全部删除[命令:999dd]

集群设置的 2 种格式:

第 1 种: linsten 集群名 *:80

balance 算法格式

server web1

server web2

第 2 种: frontend 集群名 *:80

use_backend 后台名

backend 后台名

server1

server2

```
retries  3  #3 次连接失败就认为服务不可用，也可以通过后面设置
listen stats 0.0.0.0:1080  #监听端口(0.0.0.0 同 *)
    stats refresh 30s  #统计页面自动刷新时间
    stats uri /stats  #统计页面的 url
    stats realm Haproxy Manager  #进入管理解面查看状态信息
    stats auth admin:admin  #设置统计页面用户名和密码

listen  websrv-rewrite 0.0.0.0:80  #监听端口(0.0.0.0 同 *)
   balance roundrobin
   server  web1 192.168.2.100:80 check inter 2000 rise 2 fall 5
   server  web2 192.168.2.200:80 check inter 2000 rise 2 fall 5
```

#check 表示健康检查;inter 表示健康检查间隔时间,默认毫秒;检查时连续成功 2 次即认为服务器健康,连续失败 5 次即认为服务器不健康

3）启动服务器并设置开机启动

```
haproxy ~]# systemctl start haproxy
haproxy ~]# systemctl enable haproxy
```

步骤三：客户端验证

客户端配置与 HAProxy 相同网络的 IP 地址，并使用火狐浏览器访问 http://192.168.4.5，测试调度器是否正常工作，客户端访问 http://192.168.4.5:1080/stats 测试状态监控页面是否正常。访问状态监控页

的内容，参考图-4 所示。

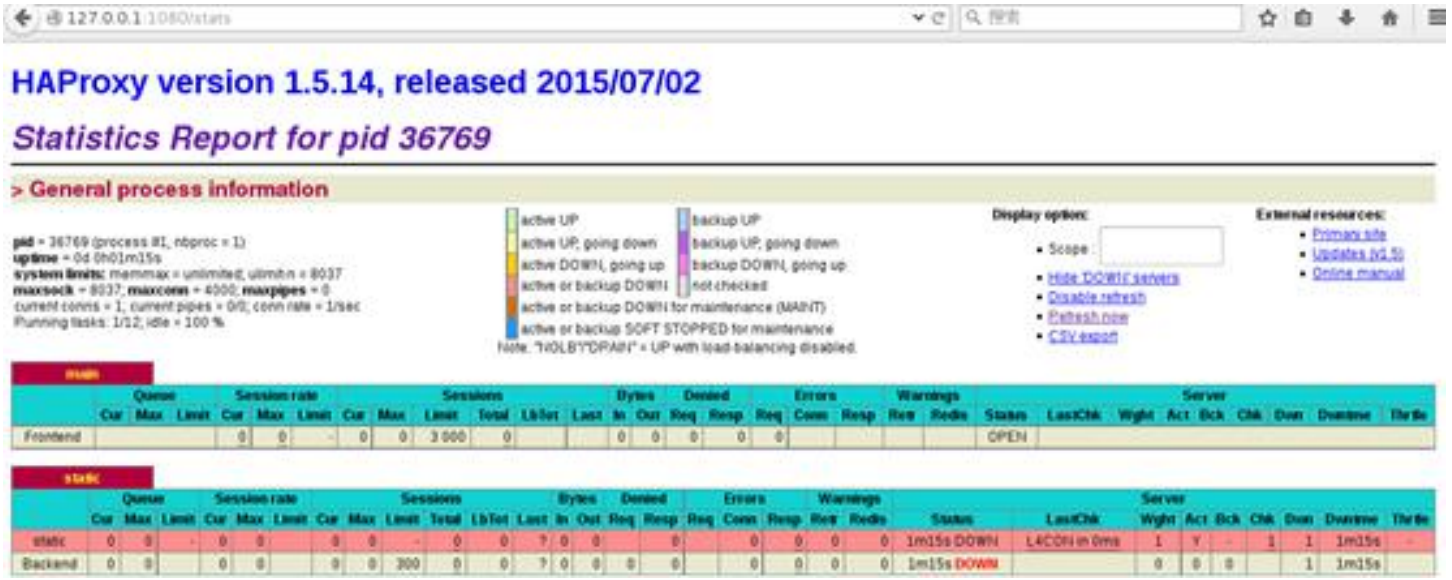

图-4

备注：

Queue 队列数据的信息（当前队列数量，最大值，队列限制数量）；

Session rate 每秒会话率（当前值，最大值，限制数量）；

Sessions 总会话量（当前值，最大值，总量，Lbtot: total number of times a server was selected 选中一台服务器所用的总时间）；

Bytes（入站、出站流量）；

Denied（拒绝请求、拒绝回应）；

Errors（错误请求、错误连接、错误回应）；

Warnings（重新尝试警告 retry、重新连接 redispatches）；

Server(状态、最后检查的时间（多久前执行的最后一次检查）、权重、备份服务器数量、down 机服务器数量、down 机时长)。

HAProxy 帮助文档:/usr/share/doc/haproxy-1.5.18/configuration.txt

五 集群调度软件对比

功能上 **LVS<HAproxy<Nginx**

性能上 **LVS>HAproxy>Nginx**

5.1 Nginx 分析(支持 4 层 7 层代理)

优点: 工作在 7 层,可以针对 http 做分流策略

1.9 版本开始支持 4 层代理

正则表达式比 HAProxy 强大

安装\配置\测试简单,通过日志可以解决大多数问题

并发量可以达到几万次

Nginx 还可以作为 web 服务器使用

缺点: 7 层代理仅支持 http https mail 协议,应用面小

监控检查仅通知端口,无法使用 url(即网页)检查

5.2 LVS 分析(支持 4 层代理)

优点: 负载能力强,工作在 4 层,对内存 CPU 消耗低

配置性低,没有太多可配置性,减少人为错误

应用面广,几乎可以为所有应用提供负载均衡

缺点: 不支持正则表达式,不能实现动静分离

如果网站架构庞大,LVS-DR 配置比较繁琐

5.3 HAProxy

优点: 支持 session cookie 功能

可以通过 url 进行健康检查

效率\负载均衡速度,高于 Nginx,低于 LVS

HAProxy 支持 TCP,可以对 MySQL 进行负载均衡

调度算法丰富

缺点: 正则若于 Nginx

日志依赖于 syslogd