Day02 集群及LVS简介 LVS-NAT集群 LVS-DR集群

1. 集群及LVS简介
2. 集群简介

1) 什么是集群

· 高速网互联

· 多服务器集中起来提供一种服务,在客户端看来就像是只有一个服务器

· 节省成本,低成本下获得在性能,可靠性和灵活性方面的相对较高的收益

· 任务调度时集群系统中的核心技术

2) 集群目的

· 提高性能

· 降低成本

· 提高扩展性

· 增强可靠性

3) 集群分类

· 高性能计算集群HPC

- 通过以集群开发的并行应用程序,解决复杂的科学问题

· 负载均衡(LB)集群

- 客户端负载在计算机集群中尽可能平均分摊

· 高可用(HA)集群

- 避免单点故障,当一个系统发生故障时,可以快速迁移

1. LVS概述

1) LVS项目介绍

· Linux虚拟服务器(LVS)是章文嵩在国防科大就读博士期间创建的

· LVs可以实现高可用的,可伸缩的Web,Mail,Cache和Media等网络服务

· 最终目标时利用Linux操作系统和LVS集群软件实现一个高可用,高性能,低成本的服务器应用集群

/\*\*

\* LVS:性能高,功能少,不支持正则

\* nginx:性能中,功能多,正则

\*\*/

1. LVS集群组成

· 前端:负载均衡层

- 由一台或多台负载调度器构成

· 中间:服务器群组层

- 由一组实际运行应用服务的服务器组成

· 底端:数据共享存储层

- 提供共享存储空间的存储区域

/\*\*

\* 用keeplived来保证LVS的高可用

\*\*/

3) LVS术语

· Director Server:调度服务器

- 将负载分发到Real Server的服务器

· Real Server:真实服务器

- 真正提供应用服务的服务器

· VIP[virtual IP]:虚拟IP地址

- Director Server公布给用户访问的虚拟IP地址

· RIP[real IP]:真实IP地址

- 集群节点[Real Server]上使用的IP地址

· DIP:调度器连接节点服务器的IP地址

· CIP[Client IP]:客户端IP

4) LVS工作模式

· NAT模式

- 通过网络地址转换实现的虚拟服务器;

- 用户通过互联网访问VIP,LVS可以看作一个路由器;

- 要求从哪里进,从哪里出;

- 与路由器的区别:路由器只是单纯把内网服务器发布到公网,而LVS可以实现负载均衡;

- 缺点:LVS的负荷高,调度器性能会成为整个集群的瓶颈;

- 适合数据量小的集群环境;

· TUN[隧道]模式

- 用户访问北京的Director Server;

- Real Server在上海;

- 通过LVS转发请求寻找Real Server服务;

· DR模式

- 直接使用路由技术实现虚拟服务器;

- 路由器将数据转给LVS,LVS转给服务器;

- 服务器将处理结果通过路由器传输给用户;

- 适合数据量较大的集群环境;

- 节点服务器需要配置VIP,注意MAC地址广播;

5) 负载均衡调度算法

· LVS目前实现了10种调度算法

· 常用调度算法有4种

- 轮询(Round Robin)

\* 将客户端请求平均分发到Real Server

- 加权轮询(Weight Round Robin)

\* 根据Real Server权重值进行轮询调度

- 最少连接(Least Connections)

\* 选择连接数最少的服务器

- 加权最少连接(Weight Least Connections)

\* 根据Real Server权重值,选择连接数最少的服务器

- 源地址散列(Source Hashing)

\* 根据请求的目标IP地址,作为散列键(Hash Key)从静态分配的散列表找出对应的服务器

- 其他调度算法

\* 给予局部性的最少链接

\* 带复制的基于局部性最少链接

\* 目标地址的三列(Destination Hashing)

\* 最短的期望的延迟

\* 最少队列调度

1. LVS-NAT集群
2. 软件安装

1) 安装前准备

· LVS的IP负载均衡技术是通过IPVS模块实现的

· IPVS模块已成为Linux组成部分

2) 安装ipvsadm

3) ipvsadm用法

ipvsadm -A 添加虚拟服务器

ipvsadm -E 修改虚拟服务器

ipvsadm -D 删除虚拟服务器

ipvsadm -C 清空所有

ipvsadm -a 添加真实服务器

ipvsadm -e 修改真实服务器

ipvsadm -d 删除真实服务器

ipvsadm -L 查看

-s [rr|wrr|lc|wlc] 指定负载调度算法

· 创建虚拟服务器

* # ipvsadm -A -t|u 192.168.4.5:80 -s [算法]

-A 添加虚拟服务器

-t 协议为tcp

-u 协议为udp

-s 执行负载调度算法

· 添加/删除服务器节点

# ipvsadm -a -t|u 192.168.4.5:80 -r 192.168.2.100 [-g|i|m] [-w 权重]

-a 添加真实服务器

-d 删除真实服务器

-r 指定真实服务器的地址

-m 使用NAT模式

-g 使用DR模式 /\* 默认模式 \*/

-i 使用TUN模式

-w 为节点服务器设置权重,,默认为1

· 查看IPVS

* # ipvsadm -Ln

· 永久保存所有规则

* # ipvsadm-save -n > /etc/sysconfig/ipvsadm
* # cat /etc/sysconfig/ipvsadm

· 清除无关规则

* # ipvsadm -C
* # rm -f /etc/sysconfig/ipvsadm

Example:

/\*\* ipvsadm选项用法

\* -A添加集群

\* -a添加服务器

\*\*/

# ipvsadm -A -t 192.168.4.5:80 -s rr

# ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.4.5:80 rr

# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100:80

# ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.4.5:80 rr

-> 192.168.2.100:80 Route 1 0 0

1. LVS-NAT案例

· 使用LVS实现NAT模式的集群调度服务器，为用户提供Web服务：

\* 集群对外公网IP地址为192.168.4.5

\* 调度器内网IP地址为192.168.2.5

\* 真实Web服务器地址分别为192.168.2.100、192.168.2.200

\* 使用加权轮询调度算法，真实服务器权重分别为1和2

· 机器配备:

\* client 192.168.4.10/24

\* proxy 192.168.4.5/24 192.168.2.5/24

\* web1 192.168.2.100/24

\* web2 192.168.2.200/24

/\*\* 注意: web1 和 web2 的eth0网卡需要关闭 \*\*/

\* 服务器节点配置(web1/2):

1) 设置Web服务器

2) 启动Web服务

3) 关闭防火墙与SELinux

# systmctl stop firewalld

# setenforce 0

\* 调度服务器配置(proxy)

1) 确认调度器的路由转发功能

# echo 1 > /proc/sys/net/ipv4/ip\_forward

# cat /proc/sys/net/ipv4/ip\_forward

1

/\*\* 上面为临时配置 \*\*/

# echo "net.ipv4.ip\_forward = 1" >> /etc/sysctl.conf

/\*\* 这个是永久配置,可以用sysctl -p命令查看规则 \*\*/

2) 创建集群服务器

# yum -y install ipvsadm

# ipvsadm -A -t 192.168.4.5:80 -s wrr

3) 添加真实服务器

# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100:80 -w 1 -m

# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.200:80 -w 1 -m

4) 查看规则列表,并保存规则

# ipvsadm-save -n > /etc/sysconfig/ipvsadm

\* 客户端测试(client)

1) 利用自带的curl浏览器反复访问Dicretor Server(proxy)

# curl http://192.168.4.5

1. 部署LVS-DR集群
2. 问题描述

· 使用LVS实现DR模式的集群调度服务器，为用户提供Web服务：

\* 客户端IP地址为192.168.4.10

\* LVS调度器VIP地址为192.168.4.15

\* LVS调度器DIP地址设置为192.168.4.5

\* 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，web1的权重为1，web2的权重为2

\* 说明：

\* CIP是客户端的IP地址；

\* VIP是对客户端提供服务的IP地址；

\* RIP是后端服务器的真实IP地址；

1. 方案

· 使用4台虚拟机,1台作为Director调度器,2台作为Real Server

\* client eth0(192.168.4.10/24)

\* proxy eth0(192.168.4.5/24) eth0:0(192.168.4.15/24)

\* web1 eth0(192.168.4.100/24) lo:0(192.168.4.15/32)

\* web2 eth0(192.168.4.200/24) lo:0(192.168.4.15/32)

/\*\*

\* 注意:

\* 1.web1/web2的子网掩码必须是32位

\* 2.VIP必须配在虚拟网卡上

\*\*/

\* DIP是调度器与后端服务器通信的IP地址（VIP必须配置在虚拟接口）。

· proxy配置:

# cp ifcfg-eth0 ifcfg-eth0:0

# vim ifcfg-eth0:0

TYPE=Ethernet

BOOTPROTO=none

DEFROUTE=yes

NAME=eth0:0

DEVICE=eth0:0

ONBOOT=yes

IPADDR=192.168.4.15

PREFIX=24

# systemctl restart network

· web1配置:

# cp /etc/sysconfig/network-scripts/ifcfg-lo{,:0}

# vim /etc/sysconfig/network-scripts/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

· 防止地址冲突的问题：

\* 这里因为web1也配置与代理一样的VIP地址，默认肯定会出现地址冲突；

\* sysctl.conf文件写入这下面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，这样防止地址冲突的问题。

# 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

\* 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

# sysctl -p

/\*\* 重启网络服务，设置防火墙与SELinux \*\*/

· proxy部署调度器:

# yum -y install ipvsadm

# ipvsadm -C

/\*\* 1.清空规则 \*\*/

# ipvsadm -A -t 192.168.4.15:80 -s wrr

/\*\* 2.添加集群服务器 \*\*/

# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.100 -g -w 1

# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.200 -g -w 2

/\*\* 3.添加Real Server \*\*/

# ipvsadm -Ln

# ipvsadm-save -n > /etc/sysconfig/ipvsadm

/\*\* 4.查看规则并保存 \*\*/

· 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;

扩展知识:默认LVS不带健康检查功能,需要自己手动编写动态检测脚本,实现该功能:(参考脚本如下,仅供参考)

# vim check.sh

#!/bin/bash

VIP=192.168.4.15:80

RIP1=192.168.4.100

RIP2=192.168.4.200

while :

do

for IP in $RIP1 $RIP2

do

curl -s http://$IP &>/dev/null

if [ $? -eq 0 ];then

ipvsadm -Ln | grep -q $IP || ipvsadm -a -t $VIP -r $IP

else

ipvsadm -Ln | grep -q $IP && ipvsadm -d -t $VIP -r $IP

fi

done

sleep 1

done

Day03 Keepalived热备 Keepalived+LVS 、 HAProxy服务器

1. nginx与LVS调度器区别
2. nginx代理(我帮你去访问)
3. LVS调度(转发数据包) LVS-NAT LVS-DR LVS-TUN

* # yum install ipvsadm -y
* # ipvsadm -A -t|u VIP:port -s wrr
* # ipvsadm -a -t|u VIP:port -r RIP [-g|m|i]
* # ipvsadm -E|D|C /\*\* VIP \*\*/
* # ipvsadm -e|d /\*\* RIP \*\*/
* # ipvsadm -Ln /\*\* 查看 \*\*/

\* NAT转发一定要有网关,且RIP不能和client在同网段中

/\*\*

\* Keepalived是给LVS写的

\* 1.自动配置LVS规则,做健康检查

\* 2.Keepalived学习了路由器上的功能VRRP HSRP路由热备

\*\*/

1. Keepalived热备
2. Keepalived概述
3. Keepalived概述

· 调度器出现单点故障,如何解决?

· Keepalived实现了高可用集群

· Keepalived最初是为LVS设计的,专门监控各服务器节点的状态

· Keepalived后来加入了VRRP功能,防止单点故障

1. Keepalived运行原理

· Keepalived检测每个服务器节点状态

· 服务器节点异常或工作出现故障,Keepalived将故障节点从集群系统中剔除

· 故障节点恢复后,Keepalived再将其加入到集群系统中

· 所有工作自动完成,无需人工干预

1. Keepalived服务

/\*\*

\* 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

\* 使用Keepalived实现web服务器的高可用

\* proxy服务器的IP地址为192.168.4.5

\* 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为192.168.4.80

\* 客户端通过访问VIP地址访问web页面

\*\*/

1. Keepalived安装

/\*\* web1和web2 \*\*/

# yum install keepalived

/\*\* 装包 \*\*/

# 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 50 //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（实验需要修改）

advert\_int 1

authentication {

auth\_type pass

auth\_pass 1111 //主备服务器密码必须一致

}

virtual\_ipaddress { 192.168.4.80 } //谁是主服务器谁获得该VIP（实验需要修改）

}

/\*\* 配置 \*\*/

# systemctl restart keepalived && iptables -F

/\*\* 起服务 \*\*/

\* 当web1的keepalived服务停止时,192.168.4.80会自动切换到web2上

/\*\* 注意

\* keepalived只监控服务本身,当网卡接口停止时,需要停止keepalived服务VIP才会切换

\*\*/

1. Keepalived+LVS服务器
2. 问题

1) 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192.168.4.100,192.168.4.200

· 使用加权轮询调度算法,真实iweb服务器权重不同

2) 机器配备:

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)

1. 操作

1) 设置web1服务器的网络参数

2) 为web1配置VIP地址lo:0(子网掩码必须是32)

3) 重启网络服务,设置防火墙与SELinux

4) 给web2重复上面操作

5) 配置proxy1与proxy2网络参数(VIP由keepalived自动配置)

6) 调度器安装Keepalived和ipvsadm

1. 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

/\*\* 上接VIP \*\*/

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 wrr //设置LVS调度算法为WRR

lb\_kind DR //设置LVS的模式为DR

#persistence\_timeout 50

#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

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.4.200 80 { //设置后端web服务器真实IP（实验需要修改）

weight 2 //设置权重为2

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

/\*\* 健康检查种类 \*\*/

real server {

TCP\_CHECK{

connect\_timeout 3 /\*\* 超时时间 \*\*/

nb\_get\_retry 3 /\*\* 尝试3次 \*\*/

delay\_before\_retry 3 /\*\* 每3秒重试一次 \*\*/

} /\*\* 以端口可用性为依据 \*\*/

HTTP\_CHECK{} /\*\* 以网页为依据 \*\*/

SSL\_GET{

path /

digest md5sum /\*\* 以md5码为验证 \*\*/

} /\*\* 以网页为依据 \*\*/

}

1. 配置HAProxy负载均衡集群
2. 调度器宏观对比
3. 调度器种类Nginx,LVS,HAProxy,F5 big-ip

速度:F5 > LVS > HAProxy > Nginx

\* LVS 4层调度,不支持7层

\* HAProxy 有一定的功能,速度还不错,对正则的支持不如Nginx

HAProxy既能做4层调度,也能做7层调度

\* Nginx 4,7层调度器

4.9以后支持4层调度,但还不够成熟

1. 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,实现以下功能

1) 客户端访问HAProxy,HAProxy分发请求到后端Real Server

2) 开启HAProxy监控页面,及时查看调度器状态

3) 设置HAProxy为开机启动

1. 清除前面的实验环境

· web1/web2:

* # rm -f network/ifcfg-lo:0
* # vim /etc/sysctl.conf
* # ifdown eth0
* # systemctl restart network

· proxy1/proxy2

# systemctl stop keepalived.service

1. 配置后端服务器

· 配置web1/2的httpd

· 在proxy1上安装HAProxy

# yum install haproxy -y

· 修改配置文件

* # vim /etc/haproxy/haproxy.cfg

global

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

user haproxy

group haproxy

daemon ###创建1个进程进入deamon模式运行

defaults

mode http ###默认的模式mode { tcp|http|health } log global ###采用全局定义的日志

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 60000 ###最大连接数

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

bind 0.0.0.0:1080 #监听端口

stats refresh 30s #统计页面自动刷新时间

stats uri /stats #统计页面url

stats realm Haproxy Manager #统计页面密码框上提示文本

stats auth admin:admin #统计页面用户名和密码设置

#stats hide-version #隐藏统计页面上HAProxy的版本信息

listen websrv-rewrite 0.0.0.0:80

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

参数解析

global /\*\* 全局设置 \*\*/

maxconn=2000 [全局并发量]

defaults /\*\* 默认设置 \*\*/

maxconn=1000 [默认并发量]

listen集群 /\*\* 单集群设置 \*\*/

maxconn=500 [单集群并发]

5.两种方式定义集群:

frontend+backend

listen

listen xxx \*:80

server ip1

server ip2

server web1 192.168.2.100:80 check inter 2000 rise 2 fall 5 /\*\* 健康检查:5次算失败,每20s再次检查,成功两次后放回集群 \*\*/

listen status 0.0.0.0:1080

stats refresh 30s

stats uri /stats

stats realm HAProxy Manager

stats auth admin:admin

6.帮助文档

# ls /usr/share/doc/haproxy-1.5.18/configuration.txt