--

集​​群(cluster)​是​由​两​台​或​多​台​计​算​机​(称​​为节点node​或成员member​)共​​同​执​行​任​务​

群集

集群方式

存储集群 GFS 共享存储

负载均衡 LB load balance

高可用 HA high avavilable

高性能计算

LB

负载均衡集群

一般用于请求负载分发，也就是按一定的算法把不同的请求调度给不同的节点，

这些节点提供的都是相同的服务。解决单台服务器压力过大的问题。

客户端

调度器(lb服务器) ---- 备调度器

web1 web2 web3 (real servers)

100000 100000 100000

100000 50000 0

调度算法 rr

健康检查

会话保持　　(session复制;利用类似memcache这种缓存数据保存session信息;session粘贴)

数据一致性:rsync同步,Gluster.共享存储，分布式存储.

常用的负载均衡方式

1。客户手动选择 如：下载网站很多镜像站点

下载地址一

下载地址二

2,location跳转

rewrite跳转

www.sina.com/sports/cba/

www.sina.com/sports/nba/

3.DNS轮询方式 对同一个域名加多条A记录

dig www.sina.com

可靠性低

负载分配不均衡,没有多种算法

没有健康检查，如果一个A记录节点挂了，DNS仍然会解析。就算你把DNS这条A记录改了，但DNS缓存功能会让整个网络过很久的时间才会都认这个改变

4.四层/七层负载均衡

硬件解决方案：<----一般都是四层调度，也有些设备支持七层调度。

F5 BIG-IP

Citrix NetScaler

Cisco CSS

软件解决方案：

lvs 四层调度（算法多）

keepalived（包括LVS） rhel7自带(keepalived仅仅为负载均衡提供高可用方案，本身并不参与负载均衡功能)

nginx 七层调度

Haproxy 四层或七层调度，特色在于七层调度

squid (缓存加基本的调度)　　 --这一类不能完全归于负载均衡软件里，只能做静态页面的

应用

表示

会话

传输 IP，PORT

网络

数据链路

物理

软件调度的优势：

软件成本低

硬件技术支持不够迅速

软件的可控性比较强

负载均衡集群

LVS linux virtual server （linux虚拟主机） 开源

调度算法：（这里只实现8种）

1，rr round-robin 轮循

均等地对待每台真实服务器

2,wrr weighted round-robin 加权轮循

根据真实服务器的不同处理能力来调度访问请求，并可以动态地调整权值

3,lc least-connection 最少连接

根据连接数来分配，连接数少的服务器会多分配。

4,wlc weighted least-connection 加权最少连接

同时考量服务器的处理能力和连接数来进行分配

wlc-加权的lc(加权是为了考虑服务器性能)，是最常用的算法(并且是lvs默认算法)

5,lblc 基于局部性的最少连接

主要用于cache集群

6,lblcwr 带复制的基于局部性的最少连接

主要用于cache集群

7,dh destionation hashing 目标地址散列

8,sh source hashing 源地址散列

负载均衡技术：

NAT

通过网络地址转换，调度器重定请求报文的目标地址。，将请求分发给下端的真实服务器，真实服务器响应，再通过调度器返回给客户端。

这种架构，调度器容易成为整个架构的瓶颈。

DR DIRECTING Routing:

类似tunnel技术

这种方式没有隧道的开锁，对于后台真实服务器也没有必须支持IP隧道协议的要求；

但是要求调度器必须有一块网卡和真实服务器在同一物理网段。

=============================================================================================================

LVS－NAT

简单的可以看作是：利用DNAT的解析原理，来解析出不同的DNAT来实现负载均衡的目的。

有算法的可以DNAT多个目标的DNAT

客户端 192.168.2.x

｜

｜

｜

｜ 外网 192.168.2.59

［lvs负载均衡调度器]

| | 内网 192.168.224.10

| |

| |

| |

WEB1 WEB2 web3 web4

192.168.224.11 192.168.224.12

--web1和web2网关都要指向192.168.224.10

SIP:192.168.2.x DIP:192.168.2.59

到达LVS调度器后，通过算法调给后台web(realserver),假设调给了web1

SIP:192.168.2.x DIP:192.168.224.11

到达web1后，返回

SIP:192.168.224.11 DIP:192.168.2.x

通过网关指向回给LVS调度器

SIP:192.168.224.11 DIP:192.168.2.x

因为进来时做了DNAT，所以回去时自动SNAT

SIP:192.168.2.59 DIP:192.168.2.x

问题:

上图中的调度器能否只有一个网卡,也就是没有内外网双网卡之分?

答案:不行,NAT的意思就是从一个网段转换到另一个网段,单网段不符合;

如果真的整个架构全部是同一网段,LVS也能帮你调度,但回来的时候,realserver会直接回到客户端(因为它和客户端也是同一网段,是直通的),而不会经过调度器回到客户端

实验前准备：(centos7.3平台)

1,静态ip

2,主机名配置和绑定

4,关闭iptables,selinux

6,准备客户端，只需要有firefox或elinks命令就可以了

第一大步:

lvs调度器上的配置

1,在调度器上打开ip转发,因为在上面架构图中,调度器会用到两个IP段的转发

# vim /etc/sysctl.conf

net.ipv4.ip\_forward = 1 #开启路由转发

sysctl -p #使之生效

2,在调度器(director)上安装软件包

安装ipvsadm

yum install ipvsadm -y

3:按照架构图来配置lvs进程调度

ipvsadm -A -t 192.168.2.59:80 -s rr --A参数增加服务，s参数后接调度算法，t表示 tcp服务 这里先使用rr

ipvsadm -a -t 192.168.2.59:80 -r 192.168.224.11:80 -m --a参数增加真实服务器，-r代表后接一个realserver;-m代表NAT架构

ipvsadm -a -t 192.168.2.59:80 -r 192.168.224.12:80 -m

(--这里有一个小问题，在此版本里:lvs调度器有些人eth0使用192.168.1.2/24这个IP，eth0:0使用192.168.2.59/24这个IP;使用eth0:0的网段做此实验会有问题，测试会卡住)

--上面三条写的就是访问192.168.2.59的80端口的访问会以rr算法调给192.168.224.11的80和192.168.224.12的80;如果你服务的端口不一样,直接按上面的规则修改就可以

ipvsadm -C清空规则

# ipvsadm -ln --查看ipvsadm调度规则，-l：表示列表形式，-n：参数表示以IP方式显示机器

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.2.59:80 rr

-> 192.168.224.11:80 Masq 1 0 0

-> 192.168.224.12:80 Masq 1 0 0

# ipvsadm -Sn > /etc/sysconfig/ipvsadm --规则保存到此文件 S:参数表示输出规则到标准输出

# cat /etc/sysconfig/ipvsadm

-A -t 192.168.2.59:80 -s rr

-a -t 192.168.2.59:80 -r 192.168.224.11:80 -m -w 1

-a -t 192.168.2.59:80 -r 192.168.224.12:80 -m -w 1

# systemctl start ipvsadm

# systemctl enable ipvsadm

# systemctl status ipvsadm

（# ipvsadm -d -t 192.168.2.59:80 -r 192.168.224.11:80这是删除其中一条的方法;但会删除当前使用的规则，也会删除保存的文件里的规则)

第二大步：

1,在 web1 和 web2 安装httpd

并在不同的web服务器上建立不同的主页文件内容（方便测试)，并启动httpd,网关指回192.168.224.10

在web1服务器上做

yum install httpd httpd-devel -y

echo 'web1'> /var/www/html/index.html

systemctl start httpd

systemctl enable httpd

route add default gw 192.168.224.10 --这是临时加网关，永久加网关就请配置到网络配置文件里

在web2服务器上做

yum install httpd httpd-devel -y

echo 'web2'> /var/www/html/index.html

systemctl start httpd

systemctl enable httpd

route add default gw 192.168.224.10 --这是临时加网关，永久加网关就请配置到网络配置文件里

第四大步：

在client端进行访问验证

# curl 192.168.2.59

验证结果为web1主页和web2主页轮循

===========================================================

会话保持：永久粘贴和持续性的比较

sh算法实现永久会话粘贴(类似nginx的ip\_hash)

修改策略为源地址散列算法

# ipvsadm -E -t 192.168.2.59:80 -s sh

E:参数表示修改服务，-t：表示TCP连接，-s:参数表示策略，sh：表示源地址散列算法

# ipvsadm -ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.2.59:80 sh

-> 192.168.224.11:80 Masq 1 0 11

-> 192.168.224.12:80 Masq 1 0 0

客户端测试结果:如果客户端第一次访问的是web1，那么永远访问的是web1(web1挂了还是访问web1).

通过持续性persistent实现非永久性的会话保持:

# ipvsadm -E -t 192.168.2.59:80 -s rr -p 10

# ipvsadm -ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.2.59:80 rr persistent 10

-> 192.168.224.11:80 Masq 1 0 0

-> 192.168.224.12:80 Masq 1 0 0

关于persistent 10

# ipvsadm -lnc 参数c表示列出所有连接

IPVS connection entries

pro expire state source virtual destination

TCP 00:06 NONE 192.168.2.x:0 192.168.2.59:80 192.168.224.11:80

TCP 01:56 TIME\_WAIT 192.168.2.x:39353 192.168.2.59:80 192.168.224.11:80

通过上面的命令看到客户端192.168.2.x访问了一次192.168.2.59:80，被调度给了192.168.224.11:80

当一个client访问vip的时候，ipvs或记录一条状态为NONE的信息，NONE状态前的面expire值是persistence\_timeout的值（我这里设为10,所以从10开始计算），然后根据时钟主键变小，在以下记录存在期间，同一client ip连接上来，都会被分配到同一个后端。

FIN\_WAIT的值就是tcp tcpfin udp的超时时间，当NONE的值为0时，如果FIN\_WAIT还存在，那么NONE的值会从新变成60秒，再减少，直到FIN\_WAIT消失以后，NONE才会消失，只要NONE存在，同一client的访问，都会分配到统一real server。

也就是说，客户端192.168.2.x第一次访问了被调度给192.168.224.11:80，要等2＊60+10=130秒左右再做第二次访问才可能被调度给另一个real server.

测试结果总结:假设第一次访问web1,在一定的时间内都是访问web1（如果web1挂了，在此时间内仍然访问web1，时间过了，就会按rr算法访问web2了)

5,dh－目标地址散列,假如调度器的后面是两台缓存服务器A,B而不是真正的REALSERVER，则会尽可能的把相同请求或者把同一用户的请求转发到同一个缓存服务器上面以提高缓存命中率（类似nginx的url\_hash算法)

客户端

client1 client2

http:/xxxx/xxx/xxx.jpg

　　　　　lvs调度

squid1 squid2

100

热点 冷点

　web1 web2

# ipvsadm -E -t 192.168.2.59:80 -s dh

# ipvsadm -ln

6，sh-源地址散列算法.同一个源IP地址来的请求都会被调度到同一个realserver，保持会话一致（类似nginx的ip\_hash算法)。但这样做，如果后台realserver挂掉，与它连接的所有会话会down掉。(建议使用持久性来做）

客户端

LVS调度

web1 web2

firewall1 firewall2

lvs

内网客户端1 内网客户端2

iptables (NEW established)

外网客户端1 外网客户端2

lvs (sh算法, new,established)

firewall1 firewall2

内网服务器

# ipvsadm -E -t 192.168.2.59:80 -s sh

# ipvsadm -ln

lblc-在dh的基础上面考虑后台服务器的连接数（dh+lc)

lblcr-在lblc的基础上，假设有A,B两台缓存服务器，某个用户第一次访问被重定向到A，第二次访问时A负载很大，B过于空闲这时也会打破原来的规则把客户的第二次访问重定向给B

======================================================================================================

ARP协议(address resolve protocol)

如:本机要 ping 10.1.1.5

过程为:

本机广播在局域网内 "谁是10.1.1.5，请把你的MAC告诉我",只有10.1.1.5这个回应，并把它的MAC返回给本机。本机就得到了10.1.1.5的MAC，并把它存放到本地的MAC地址表中缓存（通过ip neigh或arp -a等命令查看得到），缓存时间在linux下默认为15分钟.在这15分钟内，如果本机再找10.1.1.5，就直接在缓存里去找。15分钟过了，缓存被清除了，再去找10.1.1.5，那么就重复上面的过程.

[root@server ~]# arp -a

client2.example.com (192.168.224.12) at 00:0c:29:56:0e:c6 [ether] on ens33

client1.example.com (192.168.224.11) at 00:0c:29:72:d8:8d [ether] on ens33

gateway (192.168.224.2) at 00:50:56:fb:89:c6 [ether] on ens33

? (192.168.1.4) at 28:d9:8a:83:5e:89 [ether] on ens37

? (192.168.1.101) at <incomplete> on ens37

[root@server ~]# ip neigh

192.168.224.12 dev ens33 lladdr 00:0c:29:56:0e:c6 STALE

192.168.224.11 dev ens33 lladdr 00:0c:29:72:d8:8d REACHABLE

192.168.224.2 dev ens33 lladdr 00:50:56:fb:89:c6 REACHABLE

192.168.1.2 dev ens37 FAILED

　　　　DNS ARP

域名或主机名 ---> IP ---->　MAC

张三的家 xx区xx站xx小区xx楼xxx室

=============================LVS-DR=========================

LVS-DR 直接路由 （direct routing)

与之前架构不同的地方在于： 请求发出的时候经过LVS director，但是返回的时候不经过lvs director 直接经过router出去，

架构优势：减轻LVS director的压力。

客户端-192.168.2.x （宿主机模拟)

|

|

192.168.2.59

[router或firewalld]

192.168.224.10

|

|- - - - - - - - - - - - |

| [lvs director] 192.168.224.13

| |

| |

|--------------------------|

web1 web2

192.168.224.11 192.168.224.12 网关指向router的同网段IP 192.168.224.10

lo:0 192.168.224.13/32 lo:0 192.168.224.13/32

1，客户端请求

sip:192.168.2.x dip:192.168.2.59

smac:客户端MAC dmac:192.168.2.59的MAC

在router上DNAT

sip:192.168.2.x dip:192.168.224.13

192.168.2.59smac:客户端MAC dmac:192.168.224.13的mac

2,在director上调度（假设调给web1)

数据包里的sip和dip不变，只把dmac变为web1的，然后调给web1

sip:192.168.2.x dip:192.168.224.13

smac:客户端MAC dmac:192.168.224.11的mac （注意这里IP与MAC不同）

3，数据包到web1后（因为是通过mac地址到的），这里又会有一个问题，web1收到这个包，但它并没有192.168.224.13这个IP，所以不会解析这个包

如果在web1上虚拟192.168.224.13这个IP，那么它会与director的冲突

所以解决方法为:在web1和web2上用lo:0来虚拟192.168.224.13这个IP，那么又可以响应解析调度过来的包，又可以互相不冲突 （因为lo:0为本地回环网卡，是不与其它人通信的）

4，请求到数据后，需要返回给客户端

sip:192.168.224.13 dip:192.168.2.x

smac:192.168.224.11的mac dmac:客户端MAC

5，返回给192.168.2.x这个IP，但web1和web2并没有这个网段路由，所以把网关指向公司路由器的IP192.168.224.10（不能指director的192.168.224.13，因为DR-LVS回去时不通过director)

====================

下面开始做实验。！！！最好准备干净的没有做过实验的虚拟机！！！

机器配置：

物理主机:192.168.2.x

Router: 192.168.2.59, 192.168.224.10 (Host:Server) (双网卡)

LVS: 192.168.224.13 (Host: Client3)

WEB1: 192.168.224.11 (Host: Client1)

Web2: 192.168.224.12 (Host: Client2)

注意：

本次实验使用192.168.224.10来作为LVS前端的路由器。所以Client3的角色有所改变。

本次实验需要绑定网卡，在命令行中网卡的名字要与实际一致。网卡MAC地址也要与实际一致。

第一大步:配置router

1,打开ip\_forward

# vim /etc/sysctl.conf

net.ipv4.ip\_forward = 1

#sysctl -p 使之生效

2,先清空原来的iptables所有规则,再在router上加上两条防火墙规则(我这里使用iptables做的，你也可以换成firewalld来做)

# yum install iptables-services iptables -y

iptables -F -t nat

iptables -F

iptables -t nat -A PREROUTING -p tcp --dport 80 -i ens37 -j DNAT --to-destination 192.168.224.13

-- 特别注意CentOS7中的网卡名字规范有所改变，不再叫ethx了。替换为实际的网卡名字。

--这条是表示从ethx网卡(也就是192.168.2.59的网卡)进来访问80的包,DNAT到192.168.224.13(也就是lvs调度器的IP)

# iptables -t nat -A POSTROUTING -p tcp --dport 80 -o ens33 -j SNAT --to-source 192.168.224.10

--关于这一条（为了192.168.2.x访问192.168.2.59，变成192.168.224.10访问192.168.224.13)，这样可以实现LVS调度器能回客户端。如果你不用这条SNAT的话，也可以在LVS调度器上route add default gw 192.168.224.10指一个网关回去也可以,因为DNAT的目标机器需要一个网关才能回给client)

保存iptables规则

iptables-save > /etc/sysconfig/iptables

systemctl start iptables.service

systemctl enable iptables.service

注意:设置好转发规则后，只能从外网的机器，以外网的IP进行访问转发端口。本地访问无效，因为转发规则中设置了进出网卡

第二大步:

配置LVS调度器（在LVS上执行下面的步骤）

1, yum install ipvsadm -y

--如果是用lvs-nat做过的机器来做，先使用ipvsadm -C清空规则

2，配置调度规则。 使用rr轮询策略

ipvsadm -A -t 192.168.224.13:80 -s rr

ipvsadm -a -t 192.168.224.13:80 -r 192.168.224.11:80 -g

ipvsadm -a -t 192.168.224.13:80 -r 192.168.224.12:80 -g

--这里的-g就是表示使用路由架构;LVS调度器就会把数据包调给192.168.224.11或192.168.224.12时,就只修改MAC地址,不修改目标IP直接路由过去

# ipvsadm -ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.224.13:80 rr

-> 192.168.224.11:80 Route 1 0 0

-> 192.168.224.12:80 Route 1 0 0

ipvsadm -Sn > /etc/sysconfig/ipvsadm

systemctl start ipvsadm

systemctl enable ipvsadm

第三大步: 配置WEB服务器，以下操作在web1，web2上配置。

配置后台的两个web(无论调度几个web,每个web服务器都要做下面的步骤)

1，安装httpd监听80端口，并使用两个不同的主页方便后面的测试(真实情况,后台的web可以使用共享存储来保证数据的一致性,这里是为了测试方便,才使用不同的主页)

(安装步骤参考前边的HTTP安装以及创建网页)。

配置完毕后，在Route和LVS服务器上分别访问web1和web2，确保网络配置没有问题。

web1:

yum -y install httpd

systemctl start httpd

echo web1 > /var/www/html/a.html

web2:

yum -y install httpd

systemctl start httpd

echo web2 > /var/www/html/a.html

curl 192.168.224.12/a.html

curl 192.168.224.11/a.html

2，

# ifconfig lo:0 192.168.224.13 netmask 255.255.255.255

--注意掩码为4个255,想要永久生效，写一个ifcfg-lo:0的网卡配置文件就好了

--这一步是非常重要的,因为路由方式扔过来的包,目标IP不变,也就是说还是192.168.224.13,只是通过找192.168.224.11或者192.168.224.12的MAC地址扔过来的;

--所以web服务器上需要也需要有一个192.168.224.13这个IP来解析请求;用lo网卡来虚拟就是为了尽量不要与lvs的网卡造成ARP广播问题

问题:为什么netmask为4个255，而不是255.255.255.0；

答案:如果为255.255.255.0，那么192.168.100.0/24整个网段都无法和web服务器通迅。而我们这里只要求lvs调度器和web不能通迅就可以了（后面使用arp -s绑定解决）

3,真实服务器把默认路由指向router同物理网段的IP

# route add default gw 192.168.224.10

--想要永久生效，写到网卡配置文件

--网关指向router的网卡,就是为了回去时跳过lvs调度器(提高调度器性能),直接回到router,再回给客户端(如果前面router上做了SNAT,那么这个网关可以不用加，原因可以直接分析数据包变化过程得到)

4,# vim /etc/sysctl.conf --加上下面四句

net.ipv4.conf.lo.arp\_ignore = 1

net.ipv4.conf.lo.arp\_announce = 2

net.ipv4.conf.all.arp\_ignore = 1

net.ipv4.conf.all.arp\_announce = 2

# sysctl -p

使它生效

解释：

arp\_ignore = 1 --表示系统只回答目的IP是本地的IP的包，也就是对广播包不做响应

arp\_announce = 2 --表示系统忽略IP包的源地址，而根据目标主机选择地址

5，

# yum install arptables\_jf -y

--安装arp防火墙对目标地址为192.168.224.13的包都drop掉

# arptables -A INPUT -d 192.168.224.13 -j DROP --添加这条规则。

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

#systemctl start arptables

#systemctl enable arptables

--上面的第4和5小步都是用来解决DR架构中的arp问题;只需要用其中一个就可以了;当然两步都做也是可以的;

第四大步:

一个重要的补充步骤

1,回到lvs调度器上用ip neigh命令或者arp -a命令来查看是否有192.168.224.11或者192.168.224.12的物理地址;

就算是有,也只是你刚才可能和它们通迅过,这些MAC地址还保存着(一般15分钟后就会消失),你需要再和192.168.224.11和192.168.224.12通迅;

但你会发现你都ping不通192.168.224.11和192.168.224.12了;

原因是因为你在web服务器上配置了lo:0 192.168.224.13这个网卡

所以LVS调度器192.168.224.13去ping 192.168.224.11可以过去,但回不来,因为回来时他会直接找自己的lo:0的192.168.224.13

解决这个问题:

在LVS调度器上永久绑定web1和web2的MAC地址(可以加入/etc/rc.local里永久生效)

arp -s 192.168.224.11 00:0c:29:4e:51:b0

arp -s 192.168.224.12 00:0c:29:0b:5c:5d

第五大步:

测试

在客户端上curl 192.168.2.59/a.html测试

注意：只能在外网客户机，以外网IP的方式进行访问。在网管机上无法访问

验证：

在LVS上无法ping通 web1和web2，因为这两台机器具有和LVS相同的IP地址，所以无法回应icmp请求