## CLUSTER

Day1

-----------------------------------------------------------------------------------------------------------------------

存储:根据不同的应用环境将数据保存到某些介质上并能保证有效的访问。

SCSI小型计算机系统接口，作为输入/输出接口，主要用于硬盘、光盘等设备

DAS直连接式存储，将存储设备通过SCSI接口或光纤通道直接连接到计算机上

NAS网络技术存储，专用数据存储服务器，存储设备与服务器分离，通过TCP/IP协议访问

SAN光纤通道，通过光纤交换机、光纤路由器等将磁盘阵列等存储设备与服务器连接起来

ISCSI技术，将SCSI数据块映射为以太网数据包，基于IP Storage理论的存储技术

存储设备服务端

ISCSI Target

#yum -y install targetcli

#yum info targetcli //查看iscsi target信息

配置ISCSI target

#targetcli

//定义后端存储

/>backstores/block create iscsi\_store /dev/vdb1

//创建iqn对象

/>iscsi/ create iqn.2018-01.cn.tedu:server1

//授权客户机访问

/>iscsi/iqn.2018-01.cn.tedu:server1/tpg1/acls create iqn.2018-01.cn.tedu:client1

//绑定存储

/>iscsi/iqn.2018-01.cn.tedu:server1/tpg1/luns create /backstores/block/iscsi\_store

//绑定监听地址

/>iscsi/iqn.2018-01.cn.tedu:server1/tpg1/portals/ create 0.0.0.0

//保存配置

/>saveconfig

/>exit

//ERROR:Could not create NetworkPortal in configFS

//iscsi/iqn.2018-01.cn.tedu:server1/tpg1/portals delete 0.0.0.0 3260

服务管理

#systemctl {start|restart|enable|stop|status} target

客户端

ISCSI Initiator，软件实现，成本低、性能较低

ISCSI HBA， 硬件实现，性能好，成本低

安装软件

#yum repolist

#yum -y install iscsi-initiator-utils

//查看信息

#yum info iscsi-initiator-utils

//启动服务

#service iscsi start

//设置本机iqn名称

#vim /etc/iscsi/initiatorname.iscsi

InitiatorName=iqn.2018-01.cn.tedu:client1

//发现远程target存储(必须执行，否则发现不了硬盘)

#iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.1 --discover

错误：（原因/etc/iscsi/initiatorname.iscsi iqn错误，重新配置后加载/重启电脑？？）

//iscsiadm: Cannot perform discovery. Invalid Initiatorname.

//iscsiadm: Could not perform SendTargets discovery: invalid parameter

//登陆target

#systemctl restart iscsi

#systemctl enable iscsi

#parted /dev/sda

(parted) mklabel gpt

(parted) mkpart primary 1M 100%

(parted) quit

# mkfs.xfs /dev/sda1

mount /var/lib/msyql /dev/vda1

#blkid /dev/sda1

#vim /etc/fstab

UUID="402d1304-5781-4349-8c43-ba45eba8b427" /var/lib/mysql xfs,defaults,\_netdev 0 0

//不能多台设备同时挂载共享存储。因ext4/xfs是单节点文件系统，如果多个节点同时挂载，就会损坏文件系统，使得数据丢失。红帽GFS才能支持多节点同时挂载

-----------------------------------------------------------------------------------------------------------------------

//virsh console 连入虚拟机，ctrl+]退出；

//nmtui文本用户界面配置网络连接参数

-----------------------------------------------------------------------------------------------------------------------

Udev设备管理

devfs

--linux早期采用的静态管理方法

--/dev目录下有大量静态文件

--内核版本2.6.13开始被完全取代

udev(动态管理硬件文件的方法,如，把U盘接入到主机，主机会多一个硬盘文件，把U盘移除的时候，硬盘文件消失。)

--只有连到系统上的设备才在/dev下创建设备文件

--与主、次设备编号无关

--为设备提供持久、一致的名字

接入设备事件链

内核发现设备并导入设备状态到sysfs

Udev连接到事件通知

Udev创建设备节点或是运行指定程序

Udev通知hald守护进程

HAL探测设备信息

HAL创建设备对象结构

HAL通过系统消息总线广播该事件

用户程序也可以监控该事件

udev的作用

从内核收到添加/移除硬件事件时，udev将会分析：

-/sys目录下信息

- /etc/udev/rules.d目录中的规则

基于分析结果，udev会：

-处理设备命名

-决定要创建哪些设备文件或链接

-决定如何设置属性

-决定出发哪些事件

配置udev

主配置文件/etc/udev/udev.conf

-udev\_root:创建设备文件位置，默认为/dev

-udev\_rules: udev规则文件位置，默认为/etc/udev/rules.d

-udev\_log:syslog优先级，缺省为err

文件位置及格式

-/etc/udev/rules.d/<rule\_name>.rules

-例：75-custom.rules（数字大小顺序时规则文件执行的顺序）

规则格式

<match-key><op><value>[,...]<assignment-key><op>value[,...]

BUS==”usb”,SYSFS{serial}==”20043512321411d34721”,NAME=”udisk”

操作符

== 表示匹配

！= 表示不匹配

= 指定赋予的值

+= 添加新值

：= 指定值，且不允许被替换

udev变量（常用替代变量）

%k 内核所识别出来的设备，如sdb1

%n 设备的内核编号，入sda3中的3

%p 设备路径，如/sys/block/sdb/sdb1

%% %符号本身

Udev应用

给iscsi磁盘创建一个软链接叫idisk

1、查看iscsi磁盘（sda）在/sys/目录中的位置

# udevadm info --query=path --name=/dev/sda1

2、通过上一步的路径，查看iscsi磁盘的信息（以和其他设备进行区别）

# udevadm info --query=all --attribute-walk

--path=/devices/platform/host2/session1/target2:0:0/2:0:0:0/block/sda/sda1

3、创建规则文件

# vim /etc/udev/rules.d/90-iscsi.rules

KERNEL=="sd[a-z]\*", ACTION=="add", SUBSYSTEMS=="scsi", ATTRS{model}=="formysql ", SYMLINK+="idisk%n"

注：KERNEL==””表示内核识别出来的设备名

ACTION＝＝“add”表示新接入设备

SUBSYSTEMS和ATTRS{model}是第（2）步查到的

SYMLINK表示创建符号链接，+=表示额外创建，%n是分区号

只有把磁盘移除再接入才能生效或是重启系统

]# systemctl stop mariadb

# umount /dev/sda1

# iscsiadm --mode node --targetname iqn.2018-07.cn.tedu.nsd1803

--portal 192.168.4.1:3260 --logout

# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.1 --discover

# systemctl restart iscsi

# ll /dev/idisk\*

-----------------------------------------------------------------------------------------------------------------------

NFS：实现网络共享的，用于unix-like（类unix）系统间的共享

端口号是2049，基于RPC（远程过程调用，端口号111）服务。NFS只提供了共享功能，底层数据传输交给RPC服务。

一、只读共享

1、node1作为服务端

（1）安装软件包

[root@node1 ~]# yum install -y nfs-utils

（2）创建共享目录

[root@node1 ~]# mkdir -pv /nfsroot/nfsro

[root@node1 ~]# cp /etc/hosts /nfsroot/nfsro

（3）修改配置文件

[root@node1 ~]# vim /etc/exports

/nfsroot/nfsro \*(ro) ->允许所有地址以只读方式访问

（4）启动服务并验证

[root@node1 ~]# systemctl start nfs

[root@node1 ~]# showmount -e 192.168.4.1

2、node3作为客户端

（1）创建挂载点

[root@node3 ~]# mkdir /mnt/nsfshare

（2）查看共享，并挂载

[root@node3 ~]# showmount -e 192.168.4.1

[root@node3 ~]# mount 192.168.4.1:/nfsroot/nfsro /mnt/nsfshare

[root@node3 ~]# ls /mnt/nsfshare

二、读写共享

不管是NFS，还是SAMBA，还是FTP，只要是读写共享，必须注意本地权限和配置文件内的授权

1、Node1作为服务器

（1）创建用于读写目录

[root@node1 ~]# mkdir -m 777 /nfsroot/nfsrw

（2）修改配置文件说明

[root@node1 ~]# vim /etc/exports

/nfsroot/nfsro \*(ro)

/nfsroot/nfsrw 192.168.4.\*(rw,sync)

（3）重新输出共享

[root@node1 ~]# exportfs -rv

2、配置客户端

（1）创建挂载点

[root@node3 ~]# mkdir /mnt/rwnfs

（2）挂载，测试

[root@node3 ~]# mount 192.168.4.1:/nfsroot/nfsrw /mnt/rwnfs

[root@node3 ~]# echo 'hello world' > /mnt/rwnfs/hi.txt

3、NFS选项

(1)no\_root\_squash表示当root创建文件时，保留文件的属主属组还是root，默认写入到共享目录中的文件属主属组是nfsnobody

/nfsroot/nfsrw 192.168.4.\*(rw,sync,no\_root\_squash)

1. all\_squash：作用是客户端任何用户写入的文件属主属组都是nfsnobody。

-----------------------------------------------------------------------------------------------------------------------

Multipath多路径

主要功能

冗余：主备模式，高可用

改进性能：主主模式，负载均衡

多路径配置

#yum -y install device-mapper-multipath

#mpathconf --user\_friendly\_names n //创建配置文件并启用多路径

#/lib/udev/scsi\_id --whitelisted --device=/dev/sdb //获取WWID

#vim /etc/multipath.conf

//配置文件中声明获取WWID方法

defaults{

user\_friendly\_names no

find\_multipaths yes

}

//为多路径设备配置别名

multipaths{

multipath{

wwid “3600140522da36310a344af6ac2ab921b”

alias mpatha

}

}

启动并验证服务

#systemctl start multipathd

#systemctl enable multipathd

验证

#ls /dev/mapper //mpatha为多路径设备

#multipath -rr //重新加载多路径信息

#multipath -ll //查看多路径信息

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

集群及LVS

一组通过高速网络互联的计算组，并以单一系统的模式加以管理，将很多服务器集中起来一起，提供同一种服务，在客户端看来就像是只有一个服务器。

可以在付出较低成本的情况下获得在性能、可靠性、灵活性方面的相对较高的收益

任务调度是集群系统中的核心技术

集群的目的

提高性能，如计算密集型应用，如天气预报、核试验模型

降低成本，相对百万美元级的超级计算机，价格便宜

提高可扩展性，只要增加集群节点即可

增强可靠性，多个节点完成相同功能，避免单点失败

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

LVS（linux虚拟服务器）

可以实现高可用的、可伸缩的web、mail、cache和media等网络服务

最终目标是利用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，数据共享存储空间的存储区域

LVS术语

Director Server调度服务器，将负载分发到real server服务器

Real Server真实服务器，真正提供应用服务的服务器

VIP虚拟IP地址，分布给用户访问的虚拟IP地址

RIP真实IP地址，集群节点上使用的IP地址

DIP调度器连接节点服务器的IP地址

LVS工作模式

VS/NAT：通过网络地址转换实现的虚拟服务器，大并发访问时调度器的性能成为瓶颈

VS/DR：直接使用路由集数实现，节点服务器需要配置VIP，注意MAC地址广播

VS/TUN：通过隧道方式实现虚拟服务器

负载均衡调度算法

轮询（Round Robin）客户端请求平均分发到Real Server

加权轮询（Weighted Round Robin）根据Real Server权重进行轮询调度

最少连接（Least Connections）选择连接数量最少的服务器

加权最少连接（Weighted 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

基于局部性的最少连接

带复制的基于局部性最少链接

目标地址散列（Destination Hashing）

最短的期望的延迟

最少队列调度

-----------------------------------------------------------------------------------------------------------------------LVS-NAT集群

客户端访问director（分配器)(VIP)时，director通过网络地址转换，调度器重写请求报文的目标地址，根据预设的调度算法。将请求分派给后端的真实服务器；真实服务器的响应报文通过调度器时，报文的源地址被重写，再返回给客户机，完成整个负载均衡调度过程，但因为服务器恢复请求全经过director，会造成director巨大负载。

(分配器：公网IP和私有IP(后端网关), real server：私有IP)

1>环境确认

#grep -i ‘ipvs’ /boot/config-3.10.0-327.el7.x86\_64

#IPVS transport protocol load balancing support

#IPVS scheduler

#IPVS SH scheduler

#IPVS application helper

//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

部署流程

1>Real Sever

配置WEB服务器

#yum -y install httpd

echo ‘real server’ > /var/www/html/index.html

#systemctl enable httpd

#systemctl restart httpd

配置网关指向分配器服务器地址192.168.4.4

# nmtui -> # ifdown eth0; ifup eth0

2>Director Server

vh04添加另一个IP地址（作为虚拟服务器地址）

eth0:192.168.4.4 eth2:201.1.1.4

安装并启用ipvsadm

#yum -y install ipvsadm

#systemctl start ipvsadm

(无法启动保存配置即可，ipvsadm-save > /etc/sysconfig/ipvsadm)

#systemctl enable ipvsadm

创建虚拟服务器

#ipvsadm -A -t VIP地址:端口 -s 调度算法

#ipvsadm -A -t 172.16.16.172:80 -s rr

//-A 添加虚拟服务器

//-t 设置群集地址（VIP，Virtual IP）

//-s 指定负载调度算法

向虚拟服务器中加入节点

#ipvsadm -a -t VIP地址:端口 -r real地址 -m -w 权重

#ipvsadm -a -t 172.16.16.172:80 -r 192.168.7.21:80 -m

从虚拟服务器中删除节点

#ipvsadm -a -t VIP地址:端口 -r real地址 -m -w 权重

#ipvsadm -d -r 192.168.7.24:80 -t 172.16.16.172:80

//-a 添加真实服务器

//-d 删除真实服务器

//-r 指定真实服务器（real server）的地址

//-m 使用NAT模式；-g、-i分别对应DR、TUN模式

//-W 为节点服务器设置权重，默认为1`

查看配置

#ipvsadm -Ln //查看IPVS

保存配置

#ipvsadm-save > /etc/sysconfig/ipvsadm

//LVS调度器需要打开IP路由转发(RHEL7默认已经打开)

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

#vim /etc/sysctl.conf

net.ipv4.ip\_forward=1

3>Client

连接虚拟服务器测试

通过web浏览器访问

使用ab进行大并发测试

#ab -c 10 -n 1000 http://vip地址/index.html

在Director上查看连接数

#ipvsadm -L -n --rate

-----------------------------------------------------------------------------------------------------------------------

Commands: Either long or short options are allowed.

--add-service -A add virtual service with options

--edit-service -E edit virtual service with options

--delete-service -D delete virtual service

--clear -C clear the whole table

--restore -R restore rules from stdin

--save -S save rules to stdout

--add-server -a add real server with options

--edit-server -e edit real server with options

--delete-server -d delete real server

--list -L|-l list the table

--zero -Z zero counters in a service or all services

--set tcp tcpfin udp set connection timeout values

--start-daemon start connection sync daemon

--stop-daemon stop connection sync daemon

--help -h display this help message

-----------------------------------------------------------------------------------------------------------------------

LVS-DR集群

director(分配器)分配请求到real server。real server 处理请求后直接回应给用户，director负载均衡器仅处理客户机与服务器一半连接。从而避免性能瓶颈，同样增加系统可伸缩性。Direct Routing由于采用物理层（修改MAC地址）技术，因此所有服务器都必须处理同一物理网段。

一、拓扑：LVS调度器只有一个IP地址，它和real server在同一网络。

二、客户机要把数据发给VIP。VIP需要出现在每台服务器上（调度器、web服务器）

三、为了地址不冲突，需要把VIP配置在调度器的eth0上，把VIP配置在web服务器的lo上

四、每台主机都有VIP，客户端发来的请求，只有LVS回应。为了实现这一点，需要改web服务器的内核参数

五、实施

1、仍然使用vh04作为调度器，对它进行清理

[root@vh04 ~]# ifdown eth2

[root@vh04 ~]# ipvsadm -D -t 201.1.1.4:80

2、在vh04上配置vip

[root@vh04 ~]# cp /etc/sysconfig/network-scripts/ifcfg-eth0{,:0}

[root@vh04 ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0:0

TYPE=Ethernet

BOOTPROTO=none

NAME=eth0:0

DEVICE=eth0:0

ONBOOT=yes

IPADDR=192.168.4.100

PREFIX=24

[root@vh04 ~]# ifup eth0:0

3、在web服务器的lo上配置VIP

[root@vh02 ~]# cp /etc/sysconfig/network-scripts/ifcfg-lo{,:0}

[root@vh02 ~]# vim /etc/sysconfig/network-scripts/ifcfg-lo:0

DEVICE=lo:0

IPADDR=192.168.4.100

NETMASK=255.255.255.255

NETWORK=192.168.4.100

BROADCAST=192.168.4.100

ONBOOT=yes

NAME=lo:0

[root@vh02 ~]# ifup lo:0

4、在web服务器上修改内核参数

[root@vh02 ~]# sysctl -a | grep arp\_ig

[root@vh02 ~]# echo "net.ipv4.conf.all.arp\_ignore = 1" >> /etc/sysctl.conf

[root@vh02 ~]# echo "net.ipv4.conf.lo.arp\_ignore = 1" >> /etc/sysctl.conf

[root@vh02 ~]# sysctl -a | grep arp\_ann

[root@vh02 ~]# echo "net.ipv4.conf.all.arp\_announce = 2" >> /etc/sysctl.conf

[root@vh02 ~]# echo "net.ipv4.conf.lo.arp\_announce = 2" >> /etc/sysctl.conf

5、配置规则

[root@vh04 ~]# ipvsadm -A -t 192.168.4.100:80 -s lc

[root@vh04 ~]# ipvsadm -a -t 192.168.4.100:80 -r 192.168.4.3

[root@vh04 ~]# ipvsadm -a -t 192.168.4.100:80 -r 192.168.4.2 -g

6、ipvsadm服务

如果启动ipvsadm服务时出错，则

[root@vh04 ~]# ipvsadm-save -n > /etc/sysconfig/ipvsadm

然后再正常控制服务状态

[root@vh04 ~]# systemctl restart ipvsadm

[root@vh04 ~]# systemctl enable ipvsadm

-----------------------------------------------------------------------------------------------------------------------HAProxy服务器

工作原理

Proxy代理,客户端把请求发送到HAProxy后，HAProxy替用户发送请求到web服务器，web服务器响应HAProxy，把页面发给HAProxy。HAProxy再把页面发给客户端。

提供高可用、负载均衡以及基于TCP和HTTP应用的代理

负载均衡器：nginx/haproxy/lvs/F5

代理：

正向代理：帮助客户端缓存服务器上的数据

反向代理：帮助服务器缓存数据

衡量负载均衡性能因素

Session rate 会画率 每秒钟产生的会话数

Session concurrency 并发会话数 服务器处理会话的时间越长，并发会话数越多

Date rate 数据速率 以MB/s或Mbps衡量，大的对象导致并发会话数增加，高会话 数、高数据速率要求更多的内存

HAProxy工作模式

mode http，客户端请求被深度分析后再发往服务器

mode tcp，客户端与服务器之间建立会话，不检查第七层信息

mode health，仅做健康状态检查，已经不建议使用

HTTP协议解析

当HAProxy运行在HTTP模式下，HTTP请求（Request）和响应（Reponse）均被完全分析和索引，这样便于创建恰当的匹配guize

理解HTTP请求和响应，对于更好的创建匹配规则至关重要

HTTP事务模型

HTT P协议是事务驱动的

每一个请求（Request）仅能对应一个响应（Response）

常见模型：

HTTP close

客户端向服务器建立一个TCP连接

客户端发送请求给服务器

服务器响应客户端请求后即断开连接

如果客户端到服务器的请求不只一个，那么要不断建立连接

TCP三次握手消耗相对较大的系统投资源，同时延迟较大

Keep-alive

一次连接可传递多个请求

客户端需要知道传输内容的长度，以避免无期限的等带传输结束

降低两个http事务间的延迟

需要相对较少的服务器资源

Pipelining

仍然使用Keep-alive

在发送后续请求前，不管等前面的求求已经得到回应

适用于有大量图片的页面

降低了多次请求只见的网络延迟

HTTP头部信息

包含许多有关的客户端环境和请求正文的有用信息，如浏览器使用的语言、请求正文的长度等

HAProxy配置实例

-----------------------------------------------------------------------------------------------------------------------

1、安装

[root@vh04 bin]# yum install -y haproxy

2、修改配置文件

[root@vh04 bin]# vim /etc/haproxy/haproxy.cfg

//把# main frontend which proxys to the backends后面部分全部删除，增加以下内容：定义一个监控页面

listen stats

bind 0.0.0.0:1080 # 监控页面的端口号

stats refresh 30s # 页面自动刷新时间是30秒

stats uri /monitor # 页面网址

stats realm HaManager

stats auth admin:admin # 用户名：密码

listen myweb 0.0.0.0:80

cookie SERVERID rewrite

balance roundrobin # 调度算是rr

server web1 192.168.4.2:80 cookie a1i1 check inter 2000 rise 2 fall 5

#每2000ms检查一次服务器，2次成功是好的，5次失败表示故障

server web2 192.168.4.3:80 cookie a1i2 check inter 2000 rise 2 fall 5

-----------------------------------------------------------------------------------------------------------------------

配置文件/etc/haproxy/haproxy.cfg构成

default 为后续的其他部分设置缺省参数，缺省参数可以被后续部分重置

frontend 描述接收客户端侦听套接字（socket）集

backend 描述转发链接的服务器集

listen 把frontend和backend结合到一起的完整声明

-----------------------------------------------------------------------------------------------------------------------

3、启服务

[root@vh04 bin]# systemctl start haproxy

1. 访问http://192.168.4.4可以实现负载均衡轮询调度，

访问http://192.168.4.4:1080/mystats可以看到监控页面

-----------------------------------------------------------------------------------------------------------------------

日志构成

# man 5 rsyslog.conf

Facility.priority 相当于是服务.优先级

Facility可以是这些关键字：auth, authpriv, cron, daemon, kern, lpr, mail, mark, news,

security，syslog, user,uucp以及local0到local7

Local0到local7是预留出来的接口，供第三方应用调用。

Priority可以使用的关键字：debug, info,notice, warning, warn ，err, error，crit,

alert,emerg, panic

Debug是最不严重的级别，panic是最严重的级别。如果日志记录优先级是info，表示

比info严重的日志都需要记录。

配置haproxy日志

1、配置本机接受通过网络发来的日志

[root@node4 ~]# vim /etc/rsyslog.conf

# Provides UDP syslog reception

$ModLoad imudp

$UDPServerRun 514

# Provides TCP syslog reception

$ModLoad imtcp

$InputTCPServerRun 514

[root@node4 ~]# systemctl restart rsyslog

查看日志

[root@node4 ~]# tail -f /var/log/messages

访问haproxy调度器，可以看到日志信息。

=======================================================================Keepalived热备份

VRRP：虚拟冗余路由协议，IETF公共标准

HSRP：热备份路由协议，，思科私有

运行原理

Keepalived检测每一个服务器节点状态，服务器节点异常或工作出现故障，Keepalive将故障节点从集群系统中剔除。故障节点恢复候，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

部署

软件安装

#yum -y install keepalived

配置文件/etc/keepalived/keepalived.conf

# vim /etc/keepalived/keepalived.conf

# vrrp\_strict

vrrp\_instance VI\_1 {

state MASTER #辅助写BACKUP

interface eth0

virtual\_router\_id 51 #虚拟路由器ID号

priority 150 #优先级

advert\_int 1 # 心跳消息发送间隔

authentication { # 集群成员共享密码

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.4.200 # vip

}

}

Heartbeat：心跳

[root@node2 ~]# systemctl start keepalived

4、查看、验证

[root@node2 ~]# ip address show eth0

双主配置：两台主机，两个服务，每个主机是一个服务的主

-----------------------------------------------------------------------------------------------------------------------Keepalived+LVS

高可用调度器

使用Keepalived高可用解决调度器单点失败问题，主、备调度器上配置LVS，主调度器异常时，Keepalived启用备用调度器

搭建高可用、负载均衡的web集群

1、新建node5虚拟机，作为额外的lvs服务器。虽然有两台调度器，也是DR模式。

2、在web服务器的lo上配置VIP

3、调整web服务器的内核参数

4、调度器上不要再手工的为eth0配置VIP了。因为VIP出现在活跃的调度器上，活跃设备由keepalived决定。应该把已经存在的调度器VIP移除。

[root@node4 ~]# ifdown eth0:0

[root@node4 ~]# mv /etc/sysconfig/network-scripts/ifcfg-eth0:0 ~

5、把node4上的haproxy停掉

[root@node4 ~]# systemctl stop haproxy

6、将node4上的lvs规则清掉。因为lvs规则将由keepalived配置文件指定。

[root@node4 ~]# ipvsadm -D -t 192.168.4.100:80

7、在node5上安装ipvsadm

[root@node5 ~]# yum install -y ipvsadm

8、在node4/5安装keeaplived

[root@node4 ~]# yum install -y keepalived

9、在node4上配置keepalived

[root@node4 ~]# vim /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

root@localhost # 收件人

}

notification\_email\_from admin@tedu.cn # 发件人

smtp\_server 127.0.0.1 # 邮件服务器地址

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL

vrrp\_skip\_check\_adv\_addr

# vrrp\_strict

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_instance VI\_1 {

state MASTER

interface eth0

virtual\_router\_id 51

priority 150

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.4.100

}

}

virtual\_server 192.168.4.100 80 {

delay\_loop 6

lb\_algo rr

lb\_kind DR

persistence\_timeout 50

protocol TCP

real\_server 192.168.4.2 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.4.3 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

10、启动服务验证

[root@node4 ~]# systemctl restart keepalived.service

-----------------------------------------------------------------------------------------------------------------------

集群调度软件对比

nginx分析

优点：

工作在7层，可以针对http做分流策～

正则表达式比HAProxy强大

安装、配置、测试间大，通过日志可以解决多数问题

并发量可以达到几万次

Nginx还可以作为Web服务器使用

缺点：

仅支持http、https、mail协议，应用面小

监控检查仅通过端口，无法使用url检查

LVS分析

优点

负载能力强，工作在4层，对内存、CPU消耗低

配置性低，没有大多可配置性，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

HAProxy分析

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

正则弱于nginx

日志依赖于syslogd，不支持apache日志

-----------------------------------------------------------------------------------------------------------------------Ceph

Ceph是一个分布式文件系统

具有高扩展、高可用、高性能的特点

Ceph可以提供对象存储、块存储、文件系统存储

Ceph可以提供PB级别的存储空间（PB--->TB--->GB）

软件定义存储（Software Defined Storage）作为存储行业的一大发展趋势，已经越来越受到市场的认可

分布式文件系统（Distributed file system）

是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连，分布式文件系统的设计基于客户机/服务器模式

常用分布式文件系统

Lustre、Hadoop、FastDFS、Ceph、GlusterFS

ceph组件

OSDs 存储设备

Monitros 集群监控组建

MDSs 存放文件系统的元数据（对象存储和块存储不需要该组件）

Client ceph客户端

环境部署

配置yum源

1>物理机创建网络yum源服务器

#yum -y install vsftpd

#mkdir /var/ftp/ceph

#mount -o loop rhecs2.0-rhos9-20161113-x86\_64.iso /var/ftp/ceph

#systemctl restart vsftpd

2>虚拟机调用yum源

#vim /etc/yum.repos.d/ceph.repo

[mon]

name=mon

baseurl=ftp://192.18.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/MON

gpgcheck=0

enabled=1

[osd]

name=osd

baseurl=ftp://192.18.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/OSD

gpgcheck=0

enabled=1

[tools]

name=tools

baseurl=ftp://192.18.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/Tools

gpgcheck=0

enabled=1

配置无密钥登陆

#for i in 10 11 12 13

>do

>ssh-copy-id 192.168.4.$i

>done

NTP时间同步

创建服务器

#yum -y intall chrony

#vim /etc/chrony.conf

allow 192.168.4.0/24

local stratum 10

#systemctl restart chronyd

客户端

#yum -y intall chrony

#vim /etc/chrony.conf

server 192.168.4.100 iburst

#systemctl restart chronyd

准备存储磁盘

#cd /var/lib/libvirt/images

#qemu-img create -f qcow2 mode1-vdb.vol 10G

\*\*\*\*\*\*

#virt-manager //图形界面添加硬盘

安装软件

#yum -y install ceph-deploy

创建目录

#mkdir ceph-cluster

#cd ceph-cluster

部署存储集群

1>创建Ceph集群

创建Ceph集群配置（所有节点都为mon）

#ceph-deploy new node1 node2 node3

给所有节点安装Ceph软件包

#ceph-deploy install node1 node2 node3

初始化所有节点的mon服务（主机名解析必须对）

#ceph-deploy mon create-initial

2>创建OSD

所有节点准备磁盘分区（node1为例）

#parted /dev/vdb mklabel gpt

#parted /dev/vdb mkpart primary 1M 50%

#parted /dev/vdb mkpart primary 50% 100%

#chown ceph.ceph /dev/vdb1

#chown ceph.ceph /dev/vdb2

初始化清空磁盘数据（仅node1操作）

#ceph-deploy disk zap node1:vdc node1:vdd

#ceph-deploy disk zap node2:vdc node2:vdd

#ceph-deploy disk zap node3:vdc node3:vdd

创建OSD存储空间（仅node1操作即可）

#ceph-deploy osd create node1:vdc:/dev/vdb1 node1:vdd:/dev/vdb2

#ceph-deploy osd create node2:vdc:/dev/vdb1 node1:vdd:/dev/vdb2

#ceph-deploy osd create node3:vdc:/dev/vdb1 node1:vdd:/dev/vdb2

//创建osd存储设备，vdc为集群提供存储空间，vdb1提供JOURNAL日志，一个存储 设备对应一个日志设备，日志需要SSD，不需要很大

3>验证

查看集群状态

#ceph -s

#systemctl restart ceph\\*.service ceph\\*.target

-----------------------------------------------------------------------------------------------------------------------Ceph块存储

Ceph块设备也叫做RADOS块设备（RBD：RADOS block device）

RBD驱动已经很好的集成在linux内核中

RBD提供了企业功能，如快照、COW克隆等等

RBD还支持内存缓存，从而能够大大提高性能

块存储集群

创建镜像池

#ceph osd lspools //查看存储池（默认有一个rbd池）

创建镜像、查看镜像

#rbd create demo-image --image-feature layering --size 10G

#rbd create rbd/image --image-feature layering --size 10G

#rbd list

#rbf info demo-image

动态调整大小

缩小容量

#rbd resize --size 7G image --allow-shrink

#rbd info image

扩容容量

#rbd resize --size 15G image

#rbd info image

集群内通过KRBD访问

将镜像影射为本地磁盘

#rbd map demo-image

#lsblk

格式化

#mkfs.xfs /dev/rbd0

#mount /dev/rbd0 /mnt

客户端通过KRBD访问

客户端需要安装ceph-common软件包

#yum -y install ceph-common

拷贝配置文件（否则不知道集群在哪）

#scp 192.168.4.11/etc/ceph/ceph.conf /etc/ceph/

拷贝连接密钥（否则无连接权限）

#scp 192.168.4.11:/etc/ceph/ceph.client.admin.keyring /etc/ceph

影射镜像到本地磁盘

#rbd map image

#lsblk

#rbd showmapped

客户端格式化、挂载分区

#mkfs.xfs /dev/rbd0

#mount /dev/rbd0 /mnt/

#echo “test” > /mnt/test.txt

创建镜像快照

#rbd snap is image

#rbd snap create image --snap image-snap1

#rbd snap is image

使用快照恢复数据

#rbd snap rollback image --snap image-snap1

#umount /mnt

#mount /dev/rbd0 /mnt/

#ls /mnt/

快照克隆

#rbd snap protect image --snap image-snap1

#rbd clone image --snap image-snap1 image-clone --image-feature layering

#rbd info image-clone

#rbd flatten image-clone

#rbd info image-clone

客户端撤销磁盘映射

#umount /mnt

#rbd showmapped

#rbd unmap /dev/rbd/{poolname}/{imagename}

#rbd unmap /dev/rbd/rbd/image

删除快照与镜像

#rbd snap rm image --snap image-snap

#rbd list

#rbd rm image

---------------------------------------------------------------------------------------------------------------------创建磁盘镜像

#rbd create vm1-image --image-feature layering --size 10G

#rbd create vm2-image --image-feature layering --size 10G

#rbd list

#rbd info vm1-image

#qemu-imag info rbd:rbd/vm1-image

Ceph认证账户

Ceph默认开启用户认证，客户端需要账户才能访问

默认账户名称client.admin，key是账户的密钥

可以使用ceph auth添加新账户

#vim /etc/ceph/ceph.conf

[global]

mon\_initial\_menbers=node1,node2,node3

mon\_host=192.168.2.10,192.168.2.20,1192.168.2.30

auth\_cluster\_required = cephx //开启认证

auth\_server\_required = cephx //开启认证

auth\_client\_required = cephx //开启认证

创建KVM虚拟机

创建初始化虚拟机

#virt-manager //图像界面操作

配置libvirt secret

KVM虚拟机需要使用librbd才可以访问ceph集群，librbd访问ceph又需要账户认证

#vim secret.xml //账户信息文件

<secret ephemeral=’no’ private=’no’>

<usage type=’ceph’>

<name>client.admin secret</name>

</usage>

</secret>

#virsh secret-define --file secret.xml //使用xml配置文件创建secret

随机的UUID，这个UUID对应的有账户信息

#ceph auth get-key client.admin

#cat /etc/ceph/ceph.client.admin.keyring

#virsh secret-set-value --secret UUID --base64 client.admin账户的密码

虚拟机的XML配置文件

每个虚拟机都会有一个XML配置文件，包括名称、内存、CPU、磁盘、网卡等

#vim /etc/libvirt/qemu/vm1.xml或#vish edit vm1

<disk type=’network’ device=’disk’>

<dirver name=’qemu’ type=’raw’/>

<auth username=’admin’>

<secret type=’ceph’ uuid=’733f0fd1-e3d6-4c25-a69f-6681fc19802b’/>

//secret的uuid，有client.admin账户的密钥匙、信息

</auth>

<source protocol=’rbd’ name=’rbd/vm1’>

<host name=’192.168.4.11’ port=’6789’/>

</source>

//说明使用账户连接哪台ceph主机和端口，访问哪个池和镜像

<target dev=’vda’ bus=’virtio’/>

//说明将获取的镜像，设置为虚拟机的vda磁盘

<address type=’pci’ domain=’0x0000’ bus=’0x00’ slot=’0x07’ function=’0x0’/>

</disk>

-----------------------------------------------------------------------------------------------------------------------分布式文件系统

分布式文件系统是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连

CephFS使用Ceph集群提供与POSIX兼容的文件系统

允许Linux直接将Ceph存储mount到本地

**元数据服务器（Metadata）**

任何文件系统中的数据分为数据和元数据

数据是指普通文件中的实际数据

元数据指用来描述一个文件的特征的系统数据（比如访问权限、文件拥有者以及文件数据块的分布信息等）

部署

登陆node4，安装ceph-mds软件包

#yum -y install ceph-mds

登陆node1部署节点操作

#cd /roo/ceph-cluster

//该目录是最早部署ceph集群时，创建的目录

#ceph-deploy mds create node4

//给node4拷贝配置文件，启动mds服务

同步配置文件和key

#ceph-deploy admin node4

**文件系统服务器**

创建存储池

文件系统需要至少2个池，一个池用于存储数据，一个池用于存储元数据

#ceph osd pool create cephfs\_data 128

//创建存储池，对应128个PG

#ceph osd pool create cephfs\_metadata 128

//创建存储池，对应128个PG

创建Ceph文件系统

#ceph mds stat //查看mds状态

#ceph fs new myfs1 cephfs\_metadata cephfs\_data

#ceph fs ls

#ceph mds stat

客户端挂载

#mount -t ceph 192.168.4.11:6789:/ /mnt/cephfs/ -o

name=admin,secret=AQBTsdRapUxBKRAANXtteNUyoEmQHveb75blSg==

//注意：文件系统类型为ceph

//192.168.4.11为mon节点的ip（不是MDS节点）

//admin是用户名，secret是密钥

//密钥可以在/etc/ceph/ceph.client.admin.keyring中找到

Ceph对象存储

对象存储也就是键值存储，通其接口指令，也就是简单的GET、PUT、DEL和其他扩展，向存储服务上传下载数据

对象存储中所有数据都被认为是一个对象，所以任何数据都可以存入对象存储服务器，如图片、视频、音频等

RGW全称是Rados Gateway

RGW是Ceph对象存储网关，用于向客户端应用呈现存储界面，提供RESTful API访问接口

部署

部署RGW软件包

#ceph-deploy install --rgw node5

#cd /roo/ceph-cluster

#ceph-deploy admin node5

#ceph-deploy rgw create node5

#ps aux | grep radosgw

#systemctl stauts ceph-radosgw@\\*

#vim /etc/ceph/ceph.conf

#curl 192.168.4.15:8000

使用第三方软件访问

#radosgw-admin user create --uid=”testuser” --display-name=”First User”

#radosgw-admin user info --uid=testuser

//testuser为用户，key是账户访问密钥

#yum -y install s3cmd-2.0.1-1.el7.noarch.rpm //客户端安装软件

#s3cmd -configure //配置软件

Access Key:5E42OEGB1M95Y49IBG7B

Secret Key:i8YtM8cs7QDCK3rTRopb0TTPBFJVXdEryRbeLGK6

S3 Endpoint[s3.amazonaws.com]:192.168.4.25:8000

User HTTPS protocol [Yes]:No

Test access with supplied credentials?[Y/n]Y

Save settings?[y/N]y

//注意，其他提示都默认回车

#s3cmd ls //客户端测试

#s3cmd mb s3://my\_bucket //创建存储数据的bucket（类似于存储数据的目录）

#s3cmd ls

#s3cmd put /var/log/messages s3://my\_bucket/log/

#s3cmd get s3://my\_bucket/log/messages/tmp/ //测试下载功能

#s3cmd del s3://my\_bucket/log/messages //测试删除功能