# NSD CLUSTER DAY01

案例1：配置iSCSI服务

案例2：部署Multipath多路径环境

案例3：配置并访问NFS共享

案例4：编写udev规则

## 常见存储技术

SCSI小型计算机系统接口,类似于工控板,有接口连接硬盘,圈存机上硬盘

经过改进,sata接口,优化结构,节省空间

DAS直连存储,

网盘存储:

NAS网络技术存储(文件系统)Samba nfs httpd

可直接使用,已定义好文件系统,

SAN存储区域网络(块存储):iscsi

硬盘共享:需要使用者先格式化文件系统再使用

SAN技术续1

FC光纤通道(网盘)

FC主要组件”光纤 HBA(主机总线适配置器) FC交换机

# 1 案例1：配置iSCSI服务

1.1 问题

本案例要求先搭建好一台iSCSI服务器，并将整个磁盘共享给客户端：

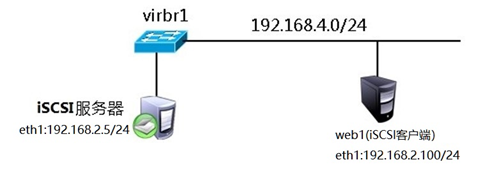
服务器上要额外配置一块硬盘

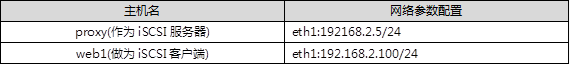
服务端安装target，并将新加的硬盘配置为iSCSI 的共享磁盘

在客户端上安装initiator，挂在服务器iSCSI，要求实现开机自动挂载

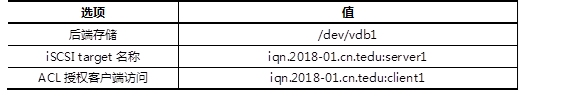
1.2 方案

使用2台RHEL7虚拟机，其中一台作为iSCSI服务器（192.168.2.5）、另外一台作为测试用的客户机（192.168.2.100），如图-1所示，主机网络地址配置如表-1所示。





在RHEL7系统中，默认通过targetcli软件包提供iSCSI服务，因此需要在服务端安装targetcli包并配置对应的服务，iSCSI服务主要配置选项如表-1所示。



客户端挂载iSCSI服务器：

* 客户端需要安装iscsi-initiator-utils软件包
* 客户端使用命令挂载后需要分区、格式化并进行挂载测试

## 步骤一：安装iSCSI服务器软件

### 1）使用yum安装targetcli软件包

[root@proxy ~]# yum -y install targetcli

.. ..

[root@proxy ~]# yum info targetcli

.. ..

## 步骤二：通过命令行配置iSCSI服务

### 1）真实主机准备底层存储磁盘

真实主机使用virt-manager工具为proxy虚拟机添加磁盘，

登陆到192.168.2.5主机，为新添加的磁盘准备分区,#parted非交互式分区

[root@proxy ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm [SWAP]

vdb 252:16 0 20G 0 disk

[root@proxy ~]# parted /dev/vdb mklabel gpt

[root@proxy ~]# parted /dev/vdb mkpart primary 1 100%

[root@proxy ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm [SWAP]

vdb 252:16 0 20G 0 disk

└─vdb1 252:17 0 20G 0 part

### 2) 使用targetcli定义后端存储

设置需要将哪个设备共享给其他主机，这里将/dev/vdb1设置为后端共享磁盘。

[root@proxy ~]# targetcli

/> ls

/> backstores/block create store /dev/vdb1

备注：store为任意名称

### 3）创建iqn对象

给iSCSI共享设置一个共享名称，客户端访问时需要使用该共享名称。

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

### 4) 授权客户机访问

类似于一个密码，设置ACL访问控制，拥有iqn.2018-01.cn.tedu:client1这个字符串的客户端才有权限访问服务器。

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

### 5) 绑定存储

将iqn共享名称（iqn.2018-01.cn.tedu:server1）与后端实际的存储设备（vdb）绑定。

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

#注意：block后面的store必须与前面步骤2定义后端存储create创建的名称一致。

### 6) 存储绑定服务监听的地址，并保存配置

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

/> saveconfig

/> exit

## 步骤三：服务管理

### 1）启动服务

[root@proxy ~]# systemctl {start|restart|stop|status} target

[root@proxy ~]# systemctl enable target

### 2）查看端口信息

[root@proxy ~]# ss -utlnp | grep :3260

### 3）关闭防火墙与SELinux

[root@proxy ~]# systemctl stop firewalld

[root@proxy ~]# setenforce 0

## 步骤四：客户端访问(web1作为客户端的角色)

### 1）客户端安装软件并启动服务

[root@web1 ~]# yum -y install iscsi-initiator-utils

### 2）设置本机的iqn名称

[root@web1 ~]# vim /etc/iscsi/initiatorname.iscsi

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

注意：必须跟服务器上配置的ACL一致！

### 3）发现远程target存储

提示：参考man iscsiadm！

[root@Web1111 ~]# man iscsiadm #搜索example,复制的时候缩小成一行

[root@web1 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.2.5 --discover

[root@web1 ~]# iscsiadm --mode node --targetname iqn.2018-01.cn.tedu:server1 --portal 192.168.2.5:3260 --login #可用重启iscsi代替此句命令(牛三牛讲的方式)

### 客户端挂载iSCSI共享

[root@web1 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 20G 0 disk #多了一个sda设备

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

[root@web1 ~]# systemctl restart iscsi

### 5）分区、格式化、挂载

[root@web1 ~]# parted /dev/sda mklabel gpt

[root@web1 ~]# parted /dev/sda mkpart primary 1 800

[root@web1 ~]# mkfs.xfs /dev/sda1

[root@web1 ~]# mount /dev/sda1 /mnt

[root@web1 ~]# umount /mnt

### 6) 扩展牛三牛iscsi操作

在实际操作中,按照1-5步骤,未成功,按照牛三牛的操作

[root@Web1111 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.2.5 --discover

执行后显示内容:

192.168.2.5:3260,1 iqn.2018-01.cn.tedu:server1

[root@Web1111 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm

[root@Web1111 ~]# systemctl restart iscsi

[root@Web1111 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 20G 0 disk

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm

## 步骤五：附加课外实验：多台FTP或者http主机使用共享存储。

这里以FTP为例，web1和web2主机都安装vsftpd软件，使用统一的后端共享存储设备。

1. web1操作(延续前面步骤三的实验)：

# 2 案例2：部署Multipath多路径环境

2.1 问题

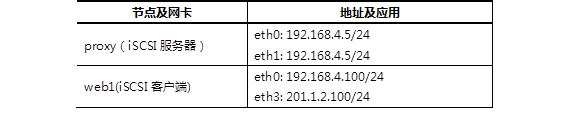
通过Multipath，实现以下目标：

* 在共享存储服务器上配置iSCSI，为应用服务器共享存储空间
* 应用服务器上配置iSCSI，发现远程共享存储
* 应用服务器上配置Multipath，将相同的共享存储映射为同一个名称

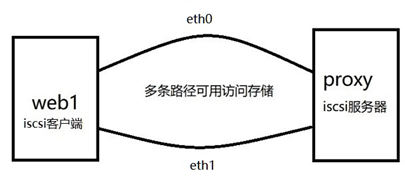
2.2 方案

配置2台虚拟机，每台虚拟机均为两块网卡：

* eth0和eth1都可用于iSCSI存储通讯
* 具体配置如表-3所示



多路径示意图，如图-8所示。



给web111新增一个IP

## 步骤一：存储服务器上添加额外的磁盘（如果已经完成案例1，此步骤可以忽略）

使用KVM软件新建（或修改）虚拟机，为虚拟机额外添加一块硬盘。

## 步骤二：存储服务器上安装并配置共享存储（如果已经完成案例1，此步骤可用忽略）

## 步骤三：在client服务器上安装并配置iSCSI客户端

### 1）安装客户端软件（前面的案例1已经完成的情况下，可以忽略此步骤）

[root@web1 ~]# yum list | grep iscsi

iscsi-initiator-utils.x86\_64 6.2.0.873-14.el6

[root@web1 ~]# yum install -y iscsi-initiator-utils

### 2）发现存储服务器的共享磁盘

因为有两条链路都可以连接到共享存储，所以需要在两条链路上都发现它。

注意：两次发现使用的IP地址不同！

分别用192.168.2.5和192.168.4.5分别发现远程target存储操作:会发现两个共享磁盘sda和sdb,为同一个磁盘.

[root@Web1111 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.2.5 --discover

192.168.2.5:3260,1 iqn.2018-01.cn.tedu:server1

[root@Web1111 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 20G 0 disk

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm [SWAP]

[root@Web1111 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.5 --discover

192.168.4.5:3260,1 iqn.2018-01.cn.tedu:server1

[root@Web1111 ~]# systemctl restart iscsi

[root@Web1111 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 20G 0 disk

sdb 8:16 0 20G 0 disk

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm [SWAP]

[root@Web1111 ~]#

### 3) 登陆共享存储

只需要将iscsi服务重启就可以自动登陆（就不需要再login了）。

在login之前，只能看到本地的存储，登陆之后，将会多出两块新的硬盘。

[root@Web1111 ~]# parted /dev/sda mkpart primary 1 100%

信息: You may need to update /etc/fstab.

[root@Web1111 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 20G 0 disk

└─sda1 8:1 0 20G 0 part

sdb 8:16 0 20G 0 disk

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm [SWAP]

[root@Web1111 ~]# parted /dev/sdb mklabel gpt

警告: The existing disk label on /dev/sdb will be destroyed and all data on

this disk will be lost. Do you want to continue?

是/Yes/否/No? n

[root@Web1111 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.2.5 --discover

192.168.2.5:3260,1 iqn.2018-01.cn.tedu:server1

[root@Web1111 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 20G 0 disk

└─sda1 8:1 0 20G 0 part

sdb 8:16 0 20G 0 disk

└─sdb1 8:17 0 20G 0 part

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm [SWAP]

提示：登陆的是同一个服务器的同一个iSCSI，但客户端看到的是两个独立的设备，sda和sdb。其实，这两个设备是同一个设备。

### 4）设置开机自启动

iscsi用于自动login远程存储，iscsid是守护进程。

[root@web1 ~]# systemctl enable iscsid

[root@web1 ~]# systemctl enable iscsi

## 步骤四：配置Multipath多路径

### 1）安装多路径软件包

[root@web1 ~]# yum list | grep multipath

device-mapper-multipath.x86\_64 0.4.9-111.el7 Server

device-mapper-multipath-libs.i686 0.4.9-111.el7 Server

device-mapper-multipath-libs.x86\_64 0.4.9-111.el7 Server

[root@web1 ~]# yum install -y device-mapper-multipath

### 2）生成配置文件

[root@web1 ~]# cd /usr/share/doc/device-mapper-multipath-0.4.9/

[root@web1 ~]# ls multipath.conf

[root@web1 ~]# cp multipath.conf /etc/multipath.conf

### 3）获取wwid

登陆共享存储后，系统多了两块硬盘，这两块硬盘实际上是同一个存储设备。应用服务器使用哪个都可以，但是如果使用sdb时，sdb对应的链路出现故障，它不会自动切换到sda。

为了能够实现系统自动选择使用哪条链路，需要将这两块磁盘绑定为一个名称。

通过磁盘的wwid来判定哪些磁盘是相同的。

取得一块磁盘wwid的方法如下：

[root@Web1~]# /usr/lib/udev/scsi\_id --whitelisted --device=/dev/sdb

36001405ddff5519835340169d64d5cfb

[root@Web1~]# /usr/lib/udev/scsi\_id --whitelisted --device=/dev/sda

36001405ddff5519835340169d64d5cfb

### 4）修改配置文件

首先声明自动发现多路径：

[root@web1 ~]# vim /etc/multipath.conf

defaults {

user\_friendly\_names yes

find\_multipaths yes

}

然后在文件的最后加入多路径声明，如果哪个存储设备的wwid和第（3）步获取的wwid一样，那么，为其取一个别名，叫mpatha。

multipaths {

multipath {

wwid "360014059e8ba68638854e9093f3ba3a0"

alias mpatha

}

}

## 步骤五：启用Multipath多路径，并测试

注意：如果做案例1时，已经挂载了iSCSI设备，一定要先umount卸载掉再启动多路径。

### 1）启动Multipath，并设置为开机启动

[root@web1 ~]# systemctl start multipathd

[root@web1 ~]# systemctl enable multipathd

### 2）检查多路径设备文件

如果多路径设置成功，那么将在/dev/mapper下面生成名为mpatha的设备文件：

[root@web1 ~]# ls /dev/mapper/

control mpatha mpatha1

### 3）对多路径设备文件执行分区、格式化、挂载操作

提示：如果前面已经对iscsi做过分区操作，则这里可以直接识别到mpatha1（就不需要再次分区了）。

[root@web1 ~]# fdisk -cu /dev/mapper/mpatha

Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel

Building a new DOS disklabel with disk identifier 0x205c887e.

Changes will remain in memory only, until you decide to write them.

After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n ＃创建分区

Command action

e extended

p primary partition (1-4)

p ＃分区类型为主分区

Partition number (1-4): 1 ＃分区编号为1

First sector (2048-4194303, default 2048): ＃起始扇区回车

Using default value 2048

Last sector, +sectors or +size{K,M,G} (2048-4194303, default 4194303): ＃回车

Using default value 4194303

Command (m for help): w ＃保存并退出

The partition table has been altered!

Calling ioctl() to re-read partition table.

新的分区名称应该是/dev/mapper/mpathap1，如果该文件不存在，则执行以下命令进行配置的重新载入：

[root@web1 ~]# ls /dev/mapper/ ＃再次查看，将会看到新的分区

control mpatha mpatha1

创建目录并挂载（如果已经格式化，这里就不需要再次格式化，直接挂载即可）：

[root@web1 ~]# mkfs.xfs /dev/mapper/mpatha1

[root@web1 ~]# mkdir /data

[root@web1 ~]# mount /dev/mapper/mpatha1 /data/

[root@web1 ~]# df -h /data/

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/mpatha1 20G 3.0M 19G 1% /data

### 4）验证多路径

查看多路径，sda和sdb都是running状态。

[root@web1 ~]# multipath -rr

reload: mpatha (360014059e8ba68638854e9093f3ba3a0) undef LIO-ORG ,store

size=9.3G features='0' hwhandler='0' wp=undef

|-+- policy='service-time 0' prio=1 status=undef

| `- 2:0:0:0 sda 8:0 active ready running

`-+- policy='service-time 0' prio=1 status=undef

`- 3:0:0:0 sdb 8:16 active ready running

关闭某个链路后，再次查看效果，此时会发现sdb为运行失败状态。

[root@web1 ~]# nmcli connection down eth1

[root@web1 ~]# multipath -rr

reject: mpatha (360014059e8ba68638854e9093f3ba3a0) undef LIO-ORG ,store

size=9.3G features='0' hwhandler='0' wp=undef

|-+- policy='service-time 0' prio=0 status=undef

| `- 2:0:0:0 sda 8:0 active undef running

`-+- policy='service-time 0' prio=0 status=undef

`- 3:0:0:0 sdb 8:16 active faulty running

使用-ll选项查看，仅sda为有效运行状态。

[root@web1 ~]# multipath -ll

reject: mpatha (360014059e8ba68638854e9093f3ba3a0) undef LIO-ORG ,store

size=9.3G features='0' hwhandler='0' wp=undef

`-+- policy='service-time 0' prio=0 status=undef

`- 2:0:0:0 sda 8:0 active undef running

# 3 案例3：配置并访问NFS共享

3.1 问题

服务器利用NFS机制发布2个共享目录，要求如下：

将目录/root共享给192.168.2.100，客户机的root用户有权限写入

将/usr/src目录共享给192.168.2.0/24网段，只开放读取权限

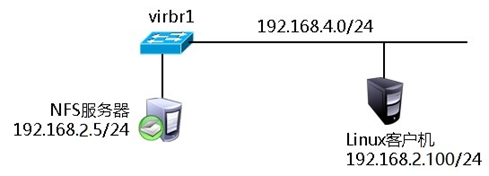
从客户机访问NFS共享：

分别查询/挂载上述NFS共享目录

查看挂载点目录，并测试是否有写入权限

3.2 方案

使用2台RHEL7虚拟机，其中一台作为NFS共享服务器（192.168.2.5）、另外一台作为测试用的Linux客户机（192.168.2.100），如图-10所示。



NFS共享的配置文件：/etc/exports 。

配置记录格式：文件夹路径 客户地址1(控制参数.. ..) 客户地址2(.. ..) 。

3.3 步骤

实现此案例需要按照如下步骤进行。

## 步骤一：配置NFS服务器，发布指定的共享

### 1）确认服务端程序、准备共享目录

软件包nfs-utils用来提供NFS共享服务及相关工具，而软件包rpcbind用来提供RPC协议的支持，这两个包在RHEL7系统中一般都是默认安装的：

[root@proxy ~]# rpm -q nfs-utils rpcbind

nfs-utils-1.3.0-0.48.el7.x86\_64

rpcbind-0.2.0-42.el7.x86\_64

根据本例的要求，需要作为NFS共享发布的有/root、/usr/src这两个目录：

[root@proxy ~]# ls -ld /root /usr/src/

dr-xr-x---. 35 root root 4096 1月 15 18:52 /root

drwxrwxr-x+ 4 root root 4096 1月 15 17:35 /usr/src/

### 2）修改/etc/exports文件，添加共享目录设置

默认情况下，来自NFS客户端的root用户会被自动降权为普通用户，若要保留其root权限，注意应添加no\_root\_squash控制参数(没有该参数，默认root会被自动降级为普通账户)；另外，限制只读的参数为ro、可读可写为rw，相关配置操作如下所示：

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

/root 192.168.2.100(rw,no\_root\_squash)

/usr/src 192.168.2.0/24(ro)

### 3）启动NFS共享相关服务，确认共享列表

依次启动rpcbiind、nfs服务：

[root@proxy ~]# systemctl restart rpcbind ; systemctl enable rpcbind

[root@proxy ~]# systemctl restart nfs ; systemctl enable nfs

使用showmount命令查看本机发布的NFS共享列表：

[root@proxy ~]# showmount -e localhost

Export list for localhost:

/usr/src 192.168.2.0/24

/root 192.168.2.100

## 步骤二：从客户机访问NFS共享

### 1）启用NFS共享支持服务

客户机访问NFS共享也需要rpcbind服务的支持，需确保此服务已开启：

[root@web1 ~]# systemctl restart rpcbind ; systemctl enable rpcbind

### 2）查看服务器提供的NFS共享列表

[root@web1 ~]# showmount -e 192.168.2.5

Export list for 192.168.2.5:

/usr/src 192.168.2.0/24

/root 192.168.2.100

### 3）从客户机192.168.2.100访问两个NFS共享，并验证权限

将远程的NFS共享/root挂载到本地的/root5文件夹，并验证可读可写：

[root@web1 ~]# mkdir /root5 //建立挂载点

[root@web1 ~]# mount 192.168.2.5:/root /root5 //挂载NFS共享目录

[root@web1 ~]# df -hT /root5 //确认挂载结果

Filesystem Type Size Used Avail Use% Mounted on

192.168.2.5:/root nfs 50G 15G 33G 31% /root5

[root@web1 ~]# cd /root5 //切换到挂载点

[root@web1 root5]# echo "NFS Write Test" > test.txt //测试写入文件

[root@web1 root5]# cat test.txt //测试查看文件

NFS Write Test

将远程的NFS共享/usr/src挂载到本地的/mnt/nfsdir，并验证只读：

[root@web1 ~]# mkdir /mnt/nfsdir //建立挂载点

[root@web1 ~]# mount 192.168.2.5:/usr/src /mnt/nfsdir/ //挂载NFS共享目录

[root@web1 ~]# df -hT /mnt/nfsdir/ //确认挂载结果

Filesystem Type Size Used Avail Use% Mounted on

192.168.2.5:/usr/src nfs 50G 15G 33G 31% /mnt/nfsdir

[root@web1 ~]# cd /mnt/nfsdir/ //切换到挂载点

[root@web1 nfsdir]# ls //读取目录列表

debug install.log kernels test.txt

[root@web1 nfsdir]# echo "Write Test." > pc.txt //尝试写入文件失败

-bash: pc.txt: 只读文件系统

！！！！ 如果从未授权的客户机访问NFS共享，将会被拒绝。比如从NFS服务器本机尝试访问自己发布的/root共享（只允许192.168.2.100访问），结果如下所示：

[root@proxy ~]# mkdir /root5

[root@proxy ~]# mount 192.168.2.5:/root /root5

mount.nfs: access denied by server while mounting 192.168.2.5:/root

### 4）设置永久挂载

[root@web1 ~]# vim /etc/fstab

.. ..

192.168.2.5:/usr/src nfsdir nfs default,ro 0 0

192.168.2.5:/root root5 nfs default 0 0

# 4 案例4：编写udev规则

4.1 问题

编写udev规则，实现以下目标：

当插入一个U盘时，该U盘自动出现一个链接称为udisk

U盘上的第1个分区名称为udisk1，以此类推

终端上出现提示信息”udisk plugged in”

4.2 方案

问题：加载一个USB设备后，系统可能识别为sda也可能识别为sdb，能不能固定呢？

对于Linux kernel 2.6及更新的操作系统版本会将设备的相关信息动态写入/sys文件系统中，而udev程序可以通过读取这些设备系信息，并根据自己的udev规则进行设备管理器，实现如下功能：

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

决定触发哪些事件

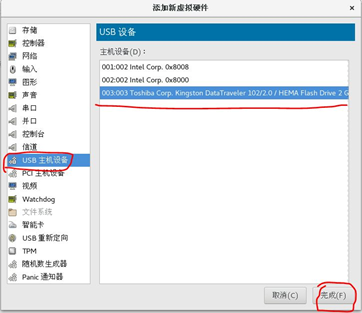
udev默认规则存放在/etc/udev/rules.d目录下，通过修改此目录下的规则实现设备的命名、属性、链接文件等。

## 步骤一：编写udev规则

## 准备USB设备

（如果使用真实机演示，下面为虚拟机添加USB设备可以忽略）

使用virt-manager为虚拟机添加USB设备，如图-5所示。注意添加设备时一定要选择正确的USB设备，图-9仅是参考案例，每个人的USB品牌与型号都有可能不一样！



### 1）查看设备属性

加载USB设备的同时实时查看设备的相关属性，可以使用monitor指令。

[root@proxy ~]# udevadm monitor --property

如果设备已经加载则无法使用monitor查看相关属性。可以使用下面的命令查看设备属性。

[root@proxy ~]# udevadm info --query=path --name=/dev/sda

[root@proxy ~]# udevadm info --query=property --path=/block/sda

单独查看某个磁盘分区的属性信息。

[root@proxy ~]# udevadm info --query=property --path=/block/sdada1

### 2）编写udev规则文件（实现插拔USB设备时有屏幕提示信息）

注意：修改规则文件不能照抄，这里的变量都是需要根据实际情况而修改的！！！

每个设备的属性都有所不同！！！一定要根据前面查询的info信息填写。

[root@proxy ~]# vim /etc/udev/rules.d/70-usb.rules

SUBSYSTEMS=="usb",ENV{ID\_VENDOR}=="TOSHIBA",ENV{serial}=="60A44CB4665EEE4133500001",RUN+="/usr/bin/wall udisk plugged in"

在virt-manager中删除、添加USB设备，测试自己的udev规则是否成功。

排错方法：通过查看/var/log/messages日志文件排错。

### 3）继续修改规则文件（实现给分区命名）

[root@proxy ~]# udevadm info --query=property --path=/block/sdb/sdb1

[root@proxy ~]# /etc/udev/rules.d/70-usb.rules

ACTION=="add",ENV{ID\_VENDOR}=="TOSHIBA",ENV{DEVTYPE}=="partition",ENV{ID\_SERIAL\_SHORT}=="60A44CB4665EEE4133500001",SYMLINK="usb%n"

在virt-manager中删除、添加USB设备，测试自己的udev规则是否成功。

### 4）继续修改规则文件（修改设备所有者和权限）

[root@proxy ~]# /etc/udev/rules.d/70-usb.rules

ACTION=="add",ENV{ID\_VENDOR}=="TOSHIBA",ENV{DEVTYPE}=="partition",ENV{ID\_SERIAL\_SHORT}=="60A44CB4665EEE4133500001",SYMLINK="usb%n",OWNER="root",GROUP="root",MODE="0644"

在virt-manager中删除、添加USB设备，测试自己的udev规则是否成功。

### 5）继续修改规则文件（插拔U盘等于启停服务）

注意：启动服务的程序systemctl，必须使用绝对路径。

[root@proxy ~]# /etc/udev/rules.d/70-usb.rules

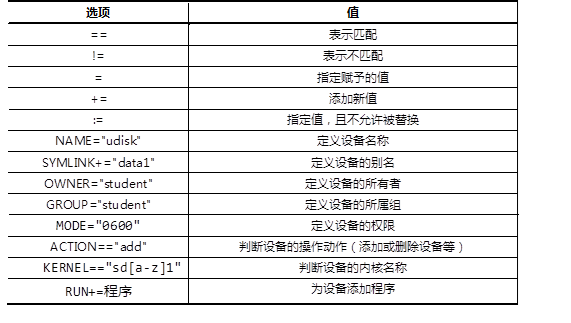
ACTION=="add",ENV{ID\_VENDOR}=="TOSHIBA",ENV{ID\_SERIAL\_SHORT}=="60A44CB4665EEE4133500001",RUN+="/usr/bin/systemctl start httpd"

ACTION=="remove",ENV{ID\_VENDOR}=="TOSHIBA",ENV{ID\_SERIAL\_SHORT}=="60A44CB4665EEE4133500001",RUN+="/usr/bin/systemctl stop httpd"

在virt-manager中删除、添加USB设备，测试自己的udev规则是否成功。

总结知识点：

udev规则文件，常见指令操作符如表-4所示。



+=给设备添加新值不覆盖旧值

udev常用替代变量：

%k：内核所识别出来的设备名，如sdb1

%n：设备的内核编号，如sda3中的3

%p：设备路径，如/sys/block/sdb/sdb1

# NSD CLUSTER DAY02

案例1：ipvsadm命令用法

案例2：部署LVS-NAT集群

案例3：部署LVS-DR集群

# 一 集群

什么是集群:

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,客户端看起来就只有一个服务器

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

集群核心:

任务调度是集群系统中的核心技术

集群目的:

提高性能:用于计算密集型应用,如天气预报/核试验模拟

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,可快速迁移

# 二 LVS项目介绍

LVS:Linux虚拟服务器,是章文嵩在国防科技大学就读博士期间创建,实现高可用的,可伸缩的web, mail, cache和media等网络服务,从2.4版本开始，linux内核默认支持LVS。

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, 低成本的服务器应用集群

## LVS术语:

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

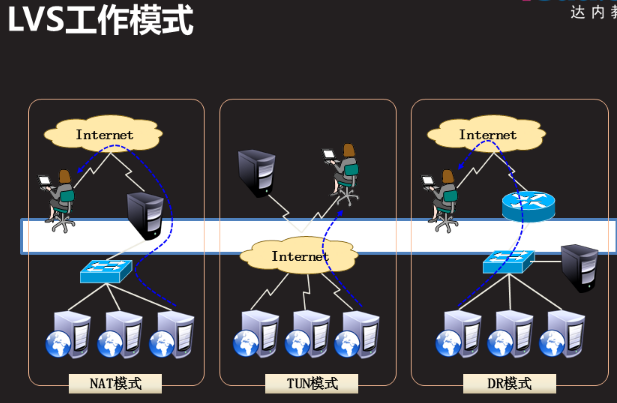
Real server: 真实服务器,真正提供应用服务的服务器

VIP:虚拟IP地址,是对客户端提供服务的IP地址；（VIP必须配置在虚拟接口）。

RIP:真实IP地址,集群节点上,后端服务器的真实IP地址；

DIP:是调度器与后端服务器通信的IP地址

## LVS的工作模式：NAT/DR/TUN



VS/NAT模式: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/TUN通过隧道方式实现虚拟服务器

VS/DR模式:直接使用路由技术实现虚拟服务器

节点服务器需要配置VIP,注意MAC地址广播

负载均衡调度算法:

LVS目前实现了10种调度算法

常用的4种算法:

轮询(round robin)--rr 调度方式:将客户端请求平均分发到real server

加权轮询(weighted round robin)--wrr 调度方式:根据real server 权重值进行轮询调度

最少连接(least connections)--lc 调度方式:选择连接数最少的服务器

加权最少连接(weighted least connections)--wlc 调度方式:根据real server权重值选 择连接数最少的服务器

# 三 ipvsadm工具介绍

　　要使用LVS的能力，只需安装一个LVS的管理工具：ipvsadm。

LVS的结构主要分为两部分：

工作在内核空间的IPVS模块(IPVS:ip虚拟服务器)。LVS的IP负载均衡技术是由IPVS模块实现。

工作在用户空间的ipvsadm管理工具。其作用是向用户提供一个命令接口，用于将配置的虚拟服务、真实服务等传给IPVS模块。

# 1 案例1：ipvsadm命令用法

1.1 问题

准备一台Linux服务器，安装ipvsadm软件包，练习使用ipvsadm命令，实现如下功能：

使用命令添加基于TCP一些的集群服务

在集群中添加若干台后端真实服务器

实现同一客户端访问，调度器分配固定服务器

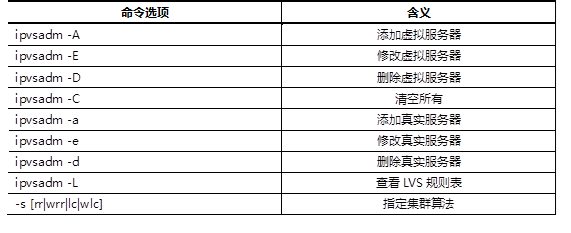
会使用ipvsadm实现规则的增、删、改

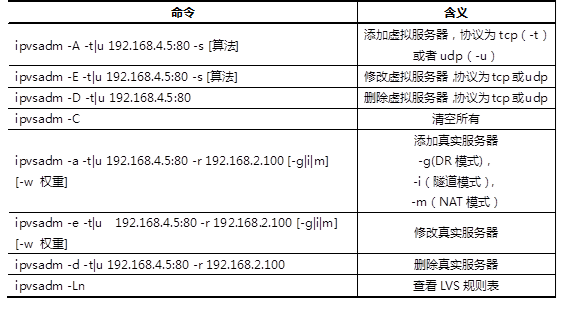
保存ipvsadm规则

1.2 方案

安装ipvsadm软件包，关于ipvsadm的用法可以参考man ipvsadm资料。

常用ipvsadm命令语法格式如表-1及表-2所示。





## 步骤一：使用命令增、删、改LVS集群规则

### 1）创建LVS虚拟集群服务器（算法为加权轮询：wrr）

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

[root@proxy ~]# ipvsadm -A -t 192.168.4.5:80 -s wrr

[root@proxy ~]# 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 wrr

### 2）为集群添加若干real server

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100 -m -w 1

[root@proxy ~]# 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 wrr

-> 192.168.2.100:80 Masq 1 0 0

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.200 -m -w 2

[root@proxy ~]# ipvsadm –a -t 192.168.4.5:80 -r 192.168.2.201 -m -w 3

[root@proxy ~]# ipvsadm –a -t 192.168.4.5:80 -r 192.168.2.202 -m -w 4

### 3）修改集群服务器设置(修改调度器算法，将加权轮询修改为轮询)

[root@proxy ~]# ipvsadm -E -t 192.168.4.5:80 -s rr

[root@proxy ~]# 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 Masq 1 0 0

-> 192.168.2.200:80 Masq 2 0 0

-> 192.168.2.201:80 Masq 2 0 0

-> 192.168.2.202:80 Masq 1 0 0

### 4）修改read server（使用-e选项，将模式改为DR模式）

[root@proxy ~]# ipvsadm -e -t 192.168.4.5:80 -r 192.168.2.202 -g

### 5）查看LVS状态

[root@proxy ~]# ipvsadm -Ln

### 创建另一个集群

（算法为最少连接算法；使用-m选项，设置工作模式为NAT模式）

[root@proxy ~]# ipvsadm -A -t 192.168.4.5:3306 -s lc

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:3306 -r 192.168.2.100 -m

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:3306 -r 192.168.2.200 -m

### 6）永久保存所有规则

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

### 7）清空所有规则

[root@proxy ~]# ipvsadm -C

# 2 案例2：部署LVS-NAT集群

2.1 问题

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

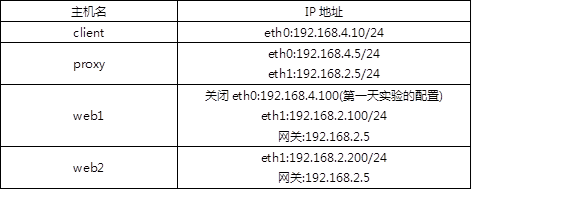
调度器内网IP地址为192.168.2.5

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

2.2 方案

实验拓扑结构主机配置细节如表-3所示。



使用4台虚拟机，1台作为Director调度器、2台作为Real Server、1台客户端，拓扑结构如图-1所示，注意：web1和web2必须配置网关地址。



## 步骤一：配置基础环境

### 1）设置Web服务器（以web1为例）

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "192.168.2.100" > /var/www/html/index.html

### 2）启动Web服务器软件

[root@web1 ~]# systemctl restart httpd

### 3)关闭防火墙与SELinux

[root@web1 ~]# systmctl stop firewalld

[root@web1 ~]# setenforce 0

## 步骤二：部署LVS-NAT模式调度器

### 1)确认调度器的路由转发功能(如果已经开启，可以忽略)

[root@proxy ~]# echo 1 > /proc/sys/net/ipv4/ip\_forward #开启路由转发

[root@proxy ~]# cat /proc/sys/net/ipv4/ip\_forward

1

[root@proxy ~]# echo "net.ipv4.ip\_forward = 1" >> /etc/sysctl.conf

#修改配置文件，开启路由转发,内核文件中添加设置永久规则

### 2）创建集群服务器

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

[root@proxy ~]# ipvsadm -A -t 192.168.4.5:80 -s wrr

#添加虚拟服务器,设置轮询机制为wrr加权轮询

### 2）添加真实服务器

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.100 -w 1 -m

[root@proxy ~]# ipvsadm -a -t 192.168.4.5:80 -r 192.168.2.200 -w 1 -m

-a:添加真实服务器, -t:tcp协议 调度服务器地址 -r:指定真实服务器(Real server) -w:权重 -m:NAT模式

### 3）查看规则列表，并保存规则

[root@proxy ~]# ipvsadm -Ln

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

## 步骤三：客户端测试

客户端使用curl命令反复连接http://192.168.4.5，查看访问的页面是否会轮询到不同的后端真实服务器。

# 3 案例3：部署LVS-DR集群

3.1 问题

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；

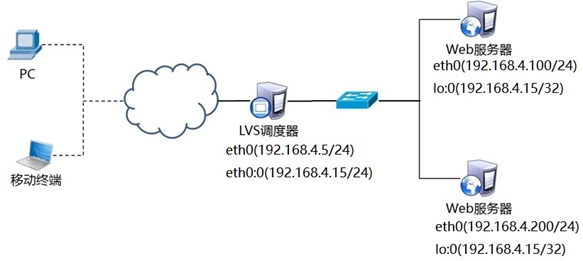
VIP:虚拟IP地址,是对客户端提供服务的IP地址；（VIP必须配置在虚拟接口）。

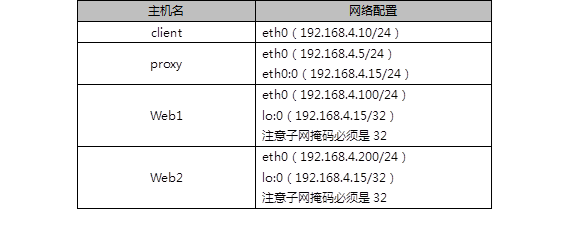
RIP:真实IP地址,集群节点上,后端服务器的真实IP地址；

DIP是调度器与后端服务器通信的IP地址

3.2 方案

使用4台虚拟机，1台作为客户端、1台作为Director调度器、2台作为Real Server，拓扑结构如图-2所示。实验拓扑结构主机配置细节如表-4所示。





3.3 步骤

实现此案例需要按照如下步骤进行。

说明：

CIP是客户端的IP地址；192.168.4.10

VIP是对客户端提供服务的IP地址；192.168.4.15

RIP是后端服务器的真实IP地址；192.168.4.100 192.168.4.200

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）。192.168.4.15

## 步骤一：配置实验网络环境

### 1）设置Proxy代理服务器的VIP和DIP

注意：为了防止冲突，VIP必须要配置在网卡的虚拟接口！！！

[root@proxy ~]# cd /etc/sysconfig/network-scripts/

[root@proxy ~]# cp ifcfg-eth0{,:0}

[root@proxy ~]# vim ifcfg-eth0

TYPE=Ethernet

BOOTPROTO=none

NAME=eth0

DEVICE=eth0

ONBOOT=yes

IPADDR=192.168.4.5

PREFIX=24

[root@proxy ~]# 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

[root@proxy ~]# systemctl restart network

### 2）设置Web1服务器网络参数

[root@web1 ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth0

接下来给web1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。

[root@web1 ~]# cd /etc/sysconfig/network-scripts/

[root@web1 ~]# cp ifcfg-lo{,:0}

[root@web1 ~]# 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

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。

[root@web1 ~]# 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

[root@web1 ~]# sysctl -p

重启网络服务，设置防火墙与SELinux

[root@web1 ~]# systemctl restart network

[root@web1 ~]# ifconfig

[root@web1 ~]# systemctl stop firewalld

[root@web1 ~]# setenforce 0

### 3）设置Web2服务器网络参数

[root@web2 ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.200/24 connection.autoconnect yes

[root@web2 ~]# nmcli connection up eth0

接下来给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。

[root@web2 ~]# cd /etc/sysconfig/network-scripts/

[root@web2 ~]# cp ifcfg-lo{,:0}

[root@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

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。

[root@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

[root@web2 ~]# sysctl -p

重启网络服务，设置防火墙与SELinux

[root@web2 ~]# systemctl restart network

[root@web2 ~]# ifconfig

[root@web2 ~]# systemctl stop firewalld

[root@web2 ~]# setenforce 0

## 步骤二：配置后端Web服务器

### 1）自定义Web页面

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "192.168.4.100" > /var/www/html/index.html

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# echo "192.168.4.200" > /var/www/html/index.html

### 2）启动Web服务器软件

[root@web1 ~]# systemctl restart httpd

[root@web2 ~]# systemctl restart httpd

## 步骤三：proxy调度器安装软件并部署LVS-DR模式调度器

### 1）安装软件（如果已经安装，此步骤可以忽略）

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

### 2）清理之前实验的规则，创建新的集群服务器规则

[root@proxy ~]# ipvsadm -C #清空所有规则

[root@proxy ~]# ipvsadm -A -t 192.168.4.15:80 -s wrr

### 3）添加真实服务器(-g参数设置LVS工作模式为DR模式，-w设置权重)

[root@proxy ~]# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.100 -g -w 1

[root@proxy ~]# ipvsadm -a -t 192.168.4.15:80 -r 192.168.4.200 -g -w 1

-a:添加真实服务器, -t:tcp协议 调度服务器地址 -r:指定真实服务器(Real server) -w:权重 -g:DR模式

### 4）查看规则列表，并保存规则

[root@proxy ~]# ipvsadm -Ln

TCP 192.168.4.15:80 wrr

-> 192.168.4.100:80 Route 1 0 0

-> 192.168.4.200:80 Route 2 0 0

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

## 步骤四：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。

proxy 调度器:不能curl http://192.168.4.15

### 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)

[root@proxy ~]# 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/vnull

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

# NSD CLUSTER DAY03

Keepalived的作用是检测服务器的状态，如果有一台web服务器宕机，或工作出现故障，Keepalived将检测到，并将有故障的服务器从系统中剔除，同时使用其他服务器代替该服务器的工作，当服务器工作正常后Keepalived自动将服务器加入到服务器群中，这些工作全部自动完成，不需要人工干涉，需要人工做的只是修复故障的服务器。

案例1：Keepalived高可用服务器

案例3：Keepalived+LVS服务器

案例1：配置HAProxy负载平衡集群

1. 写配置文件,自动执行ipvsadm配置集群
2. 健康检查
3. 做浮动VIP(虚拟IP地址,公布给用户访问的虚拟IP地址), 路由器1,路由器2 VIP在优先级高的路由器上,挂掉一台自动浮动到另一 台路由上,类似路由热备:VRRP公有协议,HSRP思科私有协议.

# 1 案例1：Keepalived高可用服务器

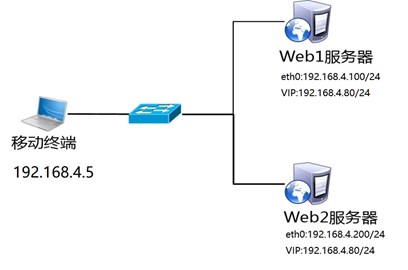
1.1 问题

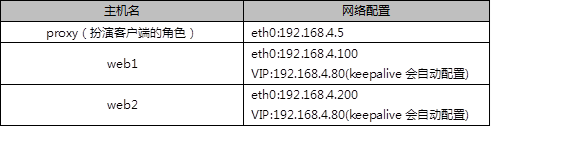
准备三台Linux服务器，两台做Web服务器，并部署Keepalived高可用软件，一台作为客户端主机，实现如下功能：

* 使用Keepalived实现web服务器的高可用
* 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

1.2 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。





## 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服务器网络参数、配置Web服务

[root@web1 ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth0

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "192.168.4.100" > /var/www/html/index.html

[root@web1 ~]# systemctl restart httpd

### 2）设置Web2服务器网络参数、配置Web服务

[root@web2 ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.200/24 connection.autoconnect yes

[root@web2 ~]# nmcli connection up eth0

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# echo "192.168.4.200" > /var/www/html/index.html

[root@web2 ~]# systemctl restart httpd

### 3）配置proxy主机的网络参数（如果已经设置，可以忽略此步骤）

[root@proxy ~]# nmcli connection modify eth0 ipv4.method manual ipv4.addresses 192.168.4.5/24 connection.autoconnect yes

[root@proxy ~]# nmcli connection up eth0

## 步骤二：安装Keepalived软件

注意：两台Web服务器做相同的操作。

[root@web1 ~]# yum install -y keepalived

[root@web2 ~]# yum install -y keepalived

## 步骤三：部署Keepalived服务

### 1）修改web1服务器Keepalived配置文件

[root@web1 ~]# vim /etc/keepalived/keepalived.conf

global\_defs { #全局配置

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）此 处不能决定主备,只能确定初始状态

interface eth0 //定义网络接口,将给这个网卡配VIP(192.168.4.80)

virtual\_router\_id 50 //主备服务器VRID号必须一致,不管默认是多少

priority 100 //服务器优先级,优先级高优先获取VIP（实验需要修改）

如果没修改,默认比较网卡MAC地址定义主备

advert\_int 1 //每隔一秒,主备进行优先级对比

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）

}

### 2）修改web2服务器Keepalived配置文件

[root@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（实验需要修改）

}

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 50 //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 { 192.168.4.80 } //谁是主服务器谁配置VIP（实验需要修改）

}

### 3）启动服务

[root@web1 ~]# systemctl start keepalived #启动之后一定要处理防火墙规则

[root@Web1111 ~]# ip a s eth0

2: eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

link/ether 52:54:00:29:3a:7d brd ff:ff:ff:ff:ff:ff

inet 192.168.4.100/24 brd 192.168.4.255 scope global eth0

valid\_lft forever preferred\_lft forever

inet 192.168.4.80/32 scope global eth0

valid\_lft forever preferred\_lft forever

inet6 fe80::db00:b29f:e4b8:2bf8/64 scope link

valid\_lft forever preferred\_lft forever

#此时192.168.4.80是在web1上

[root@web2 ~]# systemctl start keepalived

[root@Web222 ~]# ip a s eth0

2: eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

link/ether 52:54:00:f3:9a:04 brd ff:ff:ff:ff:ff:ff

inet 192.168.4.200/24 brd 192.168.4.255 scope global eth0

valid\_lft forever preferred\_lft forever

inet6 fe80::308e:8e8b:6edb:7507/64 scope link

valid\_lft forever preferred\_lft forever

### 4）配置防火墙和SELinux

启动keepalived会自动添加一个drop的防火墙规则，需要清空！

[root@web1 ~]# iptables -F #不清空是ping不通192.168.4.80

[root@web1 ~]# setenforce 0

[root@web1 ~]# ipvsadm -Ln #查看LVS规则

[root@web2 ~]# iptables -F #不清空是ping不通192.168.4.80

[root@web2 ~]# setenforce 0

[root@web2 ~]# ipvsadm -Ln #查看LVS规则

## 步骤四：测试

### 1）登录两台Web服务器查看VIP信息

[root@web1 ~]# ip addr show

[root@web2 ~]# ip addr show

[root@Web1111 ~]# systemctl stop keepalived #关闭web1上的keepalived

[root@Web1111 ~]# ip a s eth0

2: eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

link/ether 52:54:00:29:3a:7d brd ff:ff:ff:ff:ff:ff

inet 192.168.4.100/24 brd 192.168.4.255 scope global eth0

valid\_lft forever preferred\_lft forever

inet6 fe80::db00:b29f:e4b8:2bf8/64 scope link

valid\_lft forever preferred\_lft forever

[root@Web222 ~]# ip a s eth0

2: eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

link/ether 52:54:00:f3:9a:04 brd ff:ff:ff:ff:ff:ff

inet 192.168.4.200/24 brd 192.168.4.255 scope global eth0

valid\_lft forever preferred\_lft forever

inet 192.168.4.80/32 scope global eth0

valid\_lft forever preferred\_lft forever

inet6 fe80::308e:8e8b:6edb:7507/64 scope link

valid\_lft forever preferred\_lft forever

#此时192.168.4.80浮动到web2上了

### 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。

# 2 案例3：Keepalived+LVS服务器

2.1 问题

使用Keepalived为LVS的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

2.2 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。

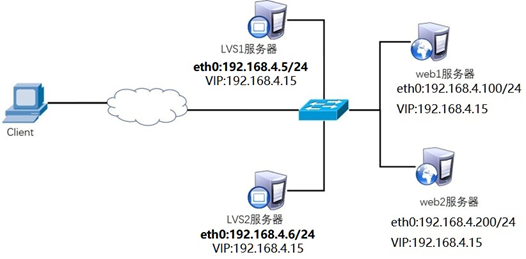
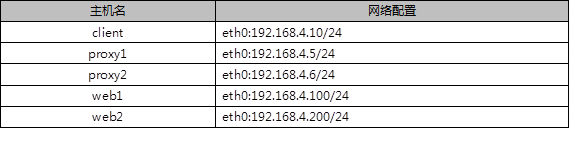


图-3

表-2



注意：所有主机都需要配置IP地址与有效的YUM源。

2.3 步骤

实现此案例需要按照如下步骤进行。

## 步骤一：配置网络环境

### 1）设置Web1服务器的网络参数

[root@web1 ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth0

接下来给web1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。

[root@web1 ~]# cd /etc/sysconfig/network-scripts/

[root@web1 ~]# cp ifcfg-lo{,:0}

[root@web1 ~]# 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。

[root@web1 ~]# 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

[root@web1 ~]# systemctl restart network

[root@web1 ~]# ifconfig

[root@web1 ~]# systemctl stop firewalld

[root@web1 ~]# setenforce 0

### 2）设置Web2服务器的网络参数

[root@web2 ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.200/24 connection.autoconnect yes

[root@web2 ~]# nmcli connection up eth0

接下来给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。

[root@web2 ~]# cd /etc/sysconfig/network-scripts/

[root@web2 ~]# cp ifcfg-lo{,:0}

[root@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web2也配置与代理一样的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。

[root@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

[root@web2 ~]# systemctl restart network

[root@web2 ~]# ifconfig

[root@web2 ~]# systemctl stop firewalld

[root@web2 ~]# setenforce 0

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

[root@proxy1 ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.5/24 connection.autoconnect yes

[root@proxy1 ~]# nmcli connection up eth0

### 4）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。

[root@proxy2 ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.6/24 connection.autoconnect yes

[root@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）

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# echo "192.168.4.100" > /var/www/html/index.html

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# echo "192.168.4.200" > /var/www/html/index.html

### 2）启动Web服务器软件(web1和web2主机)

[root@web1 ~]# systemctl start httpd ; systemctl enable httpd

[root@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

[root@proxy1 ~]# yum install -y keepalived

[root@proxy1 ~]# systemctl enable keepalived

[root@proxy1 ~]# yum install -y ipvsadm

[root@proxy1 ~]# ipvsadm -C

[root@proxy2 ~]# yum install -y keepalived

[root@proxy2 ~]# systemctl enable keepalived

[root@proxy2 ~]# yum install -y ipvsadm

[root@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

[root@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

# vrrp\_strict #此行要么注释掉要么删掉

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 50 //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 { 192.168.4.15 } //配置VIP（实验需要修改）

}

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 #每隔3s检查一次

nb\_get\_retry 3 #每次检查测三次

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

}

}

}

[root@proxy1~]# ipvsadm -C

[root@proxy1 ~]# systemctl start keepalived

[root@proxy1 ~]# iptables -F #清空防火墙规则

[root@proxy1 ~]# ipvsadm -Ln #查看LVS规则

[root@proxy1 ~]# ip a s #查看VIP配置

### 2）LVS2调度器设置Keepalived

[root@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

# vrrp\_strict #此行要么注释掉要么删掉

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 50 //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 { 192.168.4.15 } //设置VIP（实验需要修改）

}

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 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

connect\_timeout 3 //超时3S

nb\_get\_retry 3 // 连续3次

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2

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

[root@proxy2 ~]# systemctl start keepalived

[root@proxy2 ~]# ipvsadm -Ln #查看LVS规则

[root@proxy2 ~]# ip a s eth0 #查看VIP设置

## 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。

# 3 案例1：配置HAProxy负载平衡集群

HAProxy是一个使用C语言编写的自由及开放源代码软件[1]，其提供高可用性、负载均衡，以及基于TCP和HTTP的应用程序代理。

HAProxy特别适用于那些负载特大的web站点，这些站点通常又需要会话保持或七层处理。HAProxy运行在当前的硬件上，完全可以支持数以万计的并发连接。并且它的运行模式使得它可以很简单安全的整合进您当前的架构中， 同时可以保护你的web服务器不被暴露到网络上。

HAProxy实现了一种事件驱动, 单一进程模型，此模型支持非常大的并发连接数。多进程或多线程模型受内存限制 、系统调度器限制以及无处不在的锁限制，很少能处理数千并发连接。事件驱动模型因为在有更好的资源和时间管理的用户空间(User-Space) 实现所有这些任务，所以没有这些问题。此模型的弊端是，在多核系统上，这些程序通常扩展性较差。这就是为什么他们必须进行优化以 使每个CPU时间片(Cycle)做更多的工作。

包括 GitHub、Bitbucket[3]、Stack Overflow[4]、Reddit、Tumblr、Twitter[5][6]和 Tuenti[7]在内的知名网站，及亚马逊网络服务系统都使用了HAProxy。

3.1 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：

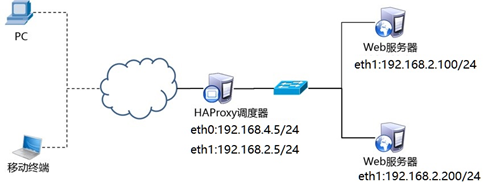
客户端访问HAProxy，HAProxy分发请求到后端Real Server

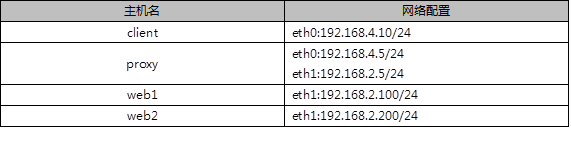
开启HAProxy监控页面，及时查看调度器状态

设置HAProxy为开机启动

3.2 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





3.3 步骤

实现此案例需要按照如下步骤进行。

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！！！

## 清理环境:

web1关闭多余的网卡与VIP，配置本地真实IP地址。

[root@web1 ~]# ifdown eth0

[root@web1 ~]# ifdown lo:0

[root@web1 ~]# nmcli connection modify eth1 ipv4.method manual \

ipv4.addresses 192.168.2.100/24 connection.autoconnect yes

[root@web1 ~]# nmcli connection up eth1

Web2关闭多余的网卡与VIP，配置本地真实IP地址。

[root@web2 ~]# ifdown eth0

[root@web2 ~]# ifdown lo:0

[root@web2 ~]# nmcli connection modify eth1 ipv4.method manual \

ipv4.addresses 192.168.2.200/24 connection.autoconnect yes

[root@web2 ~]# nmcli connection up eth1

proxy关闭keepalived服务，清理LVS规则。

[root@proxy ~]# systemctl stop keepalived

[root@proxy ~]# systemctl disable keepalived #禁止开机自启

[root@proxy ~]# ipvsadm -C #清除规则

[root@proxy ~]# nmcli connection modify eth0 ipv4.method manual \

ipv4.addresses 192.168.4.5/24 connection.autoconnect yes

[root@proxy ~]# nmcli connection up eth0

[root@proxy ~]# nmcli connection modify eth1 ipv4.method manual \

ipv4.addresses 192.168.2.5/24 connection.autoconnect yes

[root@proxy ~]# nmcli connection up eth1

## 步骤一：配置后端Web服务器

设置两台后端Web服务（如果已经配置完成，可用忽略此步骤）

[root@web1 ~]# yum -y install httpd

[root@web1 ~]# systemctl start httpd

[root@web1 ~]# echo "192.168.2.100" > /var/www/html/index.html

[root@web2 ~]# yum -y install httpd

[root@web2 ~]# systemctl start httpd

[root@web2 ~]# echo "192.168.2.200" > /var/www/html/index.html

## 步骤二：部署HAProxy服务器

### 1）配置网络，安装软件

[root@haproxy ~]# echo 'net.ipv4.ip\_forward = 1' >> /etc/sysctl.conf //开启路由转发

[root@haproxy ~]# sysctl -p

[root@haproxy ~]# yum -y install haproxy

**配置文件参考(配置不对):**

global #软件全局设置

# local2.\* /var/log/haproxy.log #日志路径

#

log 127.0.0.1 local2 #日志默认在本机[err warning info debug]

chroot /var/lib/haproxy #默认工作目录不用修改

pidfile /var/run/haproxy.pid #haproxy的pid存放路径,不用ps去查询,起了服务才有

maxconn 4000 #最大连接数，默认4000

user haproxy #软件启用用户

group haproxy #软件启用组

daemon #创建1个进程进入deamon模式运行

defaults #集群设置默认参数

mode http #默认的模式mode { tcp|http|health } ,tcp(四层调度),http(七层调度)

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 http-request 10s

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 #开监控

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 #轮询算法

server web1 192.168.2.100:80 check inter 2000 rise 2 fall 5 #web1名字随便取

server web2 192.168.2.200:80 check inter 2000 rise 2 fall 5 #web2 名字随便取

#balance roundrobin 轮询算法 leastconn 最少连接算法

#rise 2 连续两次测试没问题,视为恢复Fall 5 连续五次连不上,视为坏

### 修改配置文件

[root@haproxy ~]# vim /etc/haproxy/haproxy.cfg

----前面内容不做修改

#---------------------------------------------------------------------

# main frontend which proxys to the backends

#---------------------------------------------------------------------

####删除这个main frontend以下所有的东西,那是以前老的配置方法,添加如下调度算法

listen webs :80

balance roundrobin

server web1 192.168.4.104:80 check inter 2000 rise 2 fall 5 #web2 名字随便取

server web2 192.168.4.105:80 check inter 2000 rise 2 fall 5 #web2 名字随便取

#balance roundrobin 轮询算法 leastconn 最少连接算法

#rise 2 连续两次测试没问题,视为恢复Fall 5 连续五次连不上,视为坏

----------可做多个集群,如下再添加一个listen

listen webs :8080

balance ...

server \*\*1 .... ....

server \*\*2 ... ....

### 3）启动服务器并设置开机启动

~]# systemctl restart haproxy.service

~]# systemctl status haproxy.service

~]# ss -ntulp |grep :80

tcp LISTEN 0 128 \*:80 \*:\* users:(("haproxy",pid=5315,fd=5))

[root@kibana103 ~]# netstat -unltp |grep :80

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 5315/haproxy

[root@haproxy ~]# systemctl enable haproxy

### 4) 客户端轮测试:

使用火狐浏览器访问http://192.168.4.5，测试调度器是否正常 或者用curl

[root@es102 ~]# curl 192.168.4.103

192.168.4.104

[root@es102 ~]# curl 192.168.4.103

192.168.4.105

[root@es102 ~]# curl 192.168.4.103

192.168.4.104

[root@es102 ~]# curl 192.168.4.103

192.168.4.105

### 5)启用服务器状态监控

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，参考图如下所示。以下配置内容添加在上面配置的最下面:

listen stats :1080 #开监控,指定访问端口

stats refresh 30s #统计页面自动刷新时间

stats uri /status #统计页面url 访问监控页面地址:192.168.\*.\*:1080/status

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

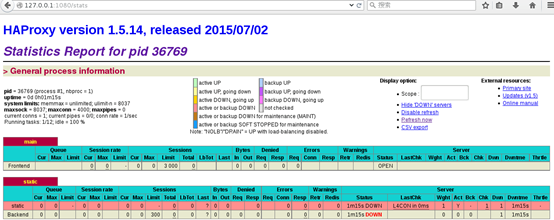
stats auth admin:admin #统计页面登录用户名和密码设置

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

配置好后重启,并查看状态,确保正常



用户名:admin 密码:admin



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)。

[root@Web1111 ~]# systemctl stop httpd

# ngixn LVS Haproxy区别

LVS:稳定快速,配置简单,只支持四层(只有ip:port),需要一定网络知识支持

nginx和haproxy:功能多,都支持四层和七层(session,cookie,http地址重写,正则)

nginx和haproxy区别

nginx:既能做代理,又能做web 可调用模块查看服务器状态信息:

haproxy:纯粹的代理

功能:nginx>Haproxy>LVS

性能:LVS > Haproxy>nginx

另外一个调度设备:F5厂商的:big-IP设备,是个硬件,较贵20万左右一个,外观像路由器, 调度更快

# NSD CLUSTER DAY04

案例1：实验环境

案例2：部署ceph集群

案例3：创建Ceph块存储

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

分布式文件系统的设计基于客户机/服务器模式(C/S)

常见分布式文件系统:Lustre Hadoop FastDFS Ceph GlusterFS

Ceph分布式文件系统:

* 具有高扩展,高可用,高性能的特点
* 可以提供对象存储,块存储,文件系统存储(唯一能提供所有存储方案)
* 可以提供PB级别的存储空间(PB--TB--GB)1PB=1024TB 1TB=1024GB

<http://docs.cephc.org/start/intro>

Ceph 组件:平台搭建

OSD ：存储设备,负责真正的存储空间, 软件包：ceph-osd

Monitor ：集群监控组件，利用Crush算法分配存储,要求至少要做三台

软件包：ceph-mon

例如：node1 node2 node3 node4

Crush 算法：将大数据打散小数据，如分成无数4M小文件，分别存入node中

取余算法：文件名 | md5sum 产生16进制数，再按0-3取余得到一个数字，文件名不变，经过md5sum 产生的16进制数不会变，那么取余也不会变，ceph不用取余算法，如果需要扩展存储空间，多了node，所有的数据需要重新计算，增加node5,数据需要重新按照0-4取余计算。

Ceph 组件:平台使用方式

RBD 块存储网管(不需要安装包):客户端可以挂载(类似iscsi),不能用fdisk工具分区

Radosgw 对象存储(云盘就是对象存储,运维搭建平台,必须写程序API调用使用,不能 被挂载,不能被mount.):可在公司搭建私有云盘

MDS 存放文件系统的元数据(对象存储和块存储不需要该组件,需要额外安装一个包,) 用户直接mount使用

块共享iscsi:客户端挂载,分区，格式化使用

文件系统共享nfs,samba：客户端直接挂载使用

对象存储共享swift,客户端需要通过API访问，API需要开发者编写程序，例如：下载 专门的云盘软件

Client ceph客户端

软件包：ceph-common

客户端访问mon，由mon分配

OSDs 存储设备,负责真正的存储空间,

Monitors 集群监控组件

Client访问 ceph-common

Mon:将数据进行md5sum加密.--->

加密后的数字%存储单元个数(取余)来定位数据存储位置, 不适合添加存储单元

ceph存储1G数据:

ceph默认3副本:Mon将1G数据拆分成多份数据,且每份数据分别复制,始终保持有三份数据,因此做ceph至少要三台机器。

RAID与Ceph 区别：

RAID：单机版

Ceph：分布式

# 1 案例1：ceph实验环境

1.1 问题

准备四台KVM虚拟机，其三台作为存储集群节点，一台安装为客户端，实现如下功能：

创建1台客户端虚拟机

创建3台存储集群虚拟机

配置主机名、IP地址、YUM源

修改所有主机的主机名

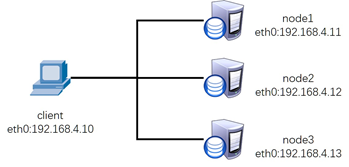
配置无密码SSH连接

配置NTP时间同步

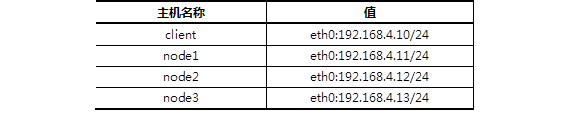
创建虚拟机磁盘

1.2 方案

使用4台虚拟机，1台客户端、3台存储集群服务器，拓扑结构如图-1所示。



所有主机的主机名及对应的IP地址如表-1所示。



## 步骤一：安装前准备

### 1）物理机为所有节点配置yum源服务器。

[root@room9pc01 ~]# yum -y install vsftpd

[root@room9pc01 ~]# mkdir /var/ftp/ceph

[root@room9pc01 ~]# mount -o loop \

rhcs2.0-rhosp9-20161113-x86\_64.iso /var/ftp/ceph

[root@room9pc01 ~]# systemctl restart vsftpd

mount -o loop loop:是将一个文件当作硬盘分区挂载

### 2）修改所有节点都需要配置YUM源（这里仅以node1为例）。

[root@node1 ~]# cat /etc/yum.repos.d/ceph.repo

[mon]

name=mon

baseurl=ftp://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/MON

gpgcheck=0

[osd]

name=osd

baseurl=ftp://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/OSD

gpgcheck=0

[tools]

name=tools

baseurl=ftp://192.168.4.254/ceph/rhceph-2.0-rhel-7-x86\_64/Tools

gpgcheck=0

用scp命令将node1 的yum文件拷贝到node2/node3/node4

### 3）修改/etc/hosts并同步到所有主机。

警告：/etc/hosts解析的域名必须与本机主机名一致！！！！

[root@node1 ~]# cat /etc/hosts

... ...

192.168.4.10 client

192.168.4.11 node1

192.168.4.12 node2

192.168.4.13 node3

警告：/etc/hosts解析的域名必须与本机主机名一致！！！！

[root@node1 ~]# for i in 10 11 12 13

> do

> scp /etc/hosts 192.168.4.$i:/etc/

> done

[root@node1 ~]# for i in 10 11 12 13

> do

> scp /etc/yum.repos.d/ceph.repo 192.168.4.$i:/etc/yum.repos.d/

> done

### 配置无密码连接(包括自己远程自己也不需要密码)。

找一台主机,无密码远程所有主机,等会安装软件,只在这台机器上连接所有机器,并安装所有软件(本实验选node1)

[root@node1 ~]# ssh-keygen -f /root/.ssh/id\_rsa -N ' ' #非交互生成秘钥文件

[root@node1 ~]# for i in 10 11 12 13

> do

> ssh-copy-id 192.168.4.$i #发送秘钥到其他主机

> done

ssh-keygen -f /root/.ssh/id\_rsa -N

ssh-copy-id 192.168.4.$i 也可以 ssh-copy-id client/node1/node2/node3

## 步骤二：配置NTP时间同步

### 1）真实物理机创建NTP服务器。

[root@room9pc01 ~]#yum -y install chrony

[root@client ~]# cat /etc/chrony.conf

server 0.centos.pool.ntp.org iburst

allow 192.168.4.0/24

local stratum 10

[root@room9pc01 ~]# systemctl restart chronyd

如果有防火墙规则，需要清空所有规则

[root@room9pc01 ~]# iptables -F

### 2）其他所有节点与NTP服务器同步时间（以node1为例）。

[root@node1 ~]# cat /etc/chrony.conf

server 192.168.4.254 iburst

[root@node1 ~]# systemctl restart chronyd

## 步骤三：准备存储磁盘

物理机上为每个虚拟机准备3块磁盘。（可以使用命令，也可以使用图形直接添加）

[root@room9pc01 ~]# virt-manager

3块磁盘 vdb vdc vdd

Vdb分两个区,vdb1 vdb2(现在不用作，后面有步骤做)

Vdb1 为vdc 的缓存,,数据存储先写入vdb1,传输完成后,再将数据拷贝到vdc

Vdb2 为vdd的缓存

因此,vdb应该选用SSD(固态硬盘),如果vdb不是SSD可不用做缓存.或者 vdc vdd也都是SSD,也没必要用vdb做缓存

本实验是为了练习，可以做

vdc vdd 为真正存储

# 2 案例2：ceph(分布式文件系统)集群

Linux持续不断进军可扩展计算空间，特别是可扩展存储空间。Ceph 最近加入到 Linux 中令人印象深刻的文件系统备选行列，它是一个分布式文件系统，能够在维护 POSIX 兼容性的同时加入了复制和容错功能。

Ceph是一种为优秀的性能、可靠性和可扩展性而设计的统一的、分布式文件系统。

1. 可轻松扩展到数 PB 容量

2. 支持多种工作负载的高性能（每秒输入/输出操作[IOPS]和带宽）

3. 高可靠性

**系统架构**

Ceph 生态系统架构可以划分为四部分：

1. Clients：客户端（数据用户）
2. cmds：Metadata server cluster，元数据服务器（缓存和同步分布式元数据）
3. cosd：Object storage cluster，对象存储集群（将数据和元数据作为对象存储，执行其他关键职能）
4. cmon：Cluster monitors，集群监视器（执行监视功能）

2.1 问题

沿用练习一，部署Ceph集群服务器，实现以下目标：

1. 安装部署工具ceph-deploy
2. 创建ceph集群
3. 准备日志磁盘分区
4. 创建OSD存储空间
5. 查看ceph状态，验证

## 步骤一：部署工具软件

### 1）在node1安装部署工具，学习工具的语法格式。

[root@node1 ~]# yum -y install ceph-deploy

[root@node1 ~]# ceph-deploy --help

ceph-deploy只是python脚本

### 2）创建目录

[root@node1 ~]# mkdir ceph-cluster #必须创建

[root@node1 ~]# cd ceph-cluster/ #ceph 的所有命令必须在这个目录里面执行

[root@node111 ceph-cluster]# ls #目前是空目录

## 步骤二：部署Ceph集群

### 1）创建Ceph集群配置。

**[root@node1 ceph-cluster]# ceph-deploy new node1 node2 node3**

Are you sure you want to continue connecting (yes/no)? yes #安装成功

报如下的错：

[root@node1 ceph-cluster]# ceph-deploy new node1 node2 node3

usage: ceph-deploy new [-h] [--no-ssh-copykey] [--fsid FSID]

[--cluster-network CLUSTER\_NETWORK]

[--public-network PUBLIC\_NETWORK]

MON [MON ...]

ceph-deploy new: error: hostname: node2 is not resolvable

原因：地址解析不对。解决：需要在部署主机上管理所有集群，本案例为node1，就在node1上配置hosts 如下：

**[root@node1 ceph-cluster]# cat /etc/hosts**

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.4.11 node1

192.168.4.12 node2

192.168.4.13 node3

如需要node2 node3都成功，就都配置/etc/hosts 如上

[root@node111 ceph-cluster]# ls

ceph.conf ceph-deploy-ceph.log ceph.mon.keyring #在当前目录里面创建的目录

[root@node111 ceph-cluster]# cat ceph.mon.keyring #用户和密码文件

[mon.]

key = AQBDcFJcAAAAABAAcYzYESIjc4nSbUcbkGLRiA==

caps mon = allow \*

[root@node1 ceph-cluster]# cat ceph.conf #ceph-deploy查看配置文件

[global]

fsid = ac0fd3ee-cd93-401a-bf52-71a5070e45bd

mon\_initial\_members = node1, node2, node3

mon\_host = 192.168.4.11,192.168.4.12,192.168.4.13

auth\_cluster\_required = cephx

auth\_service\_required = cephx

auth\_client\_required = cephx

### 2）给所有节点安装软件包。

[root@node1 ceph-cluster]# ceph-deploy install node1 node2 node3

[root@node111 ~]# rpm -qa |grep ceph

libcephfs1-10.2.2-38.el7cp.x86\_64

ceph-osd-10.2.2-38.el7cp.x86\_64

ceph-deploy-1.5.33-1.el7cp.noarch

ceph-base-10.2.2-38.el7cp.x86\_64

ceph-mds-10.2.2-38.el7cp.x86\_64 #启动文件系统共享

ceph-common-10.2.2-38.el7cp.x86\_64

ceph-mon-10.2.2-38.el7cp.x86\_64

ceph-selinux-10.2.2-38.el7cp.x86\_64

python-cephfs-10.2.2-38.el7cp.x86\_64

ceph-radosgw-10.2.2-38.el7cp.x86\_64 #网关，做对象存储，装这个软件包其服务

#node1 node2 node3 都验证下所有是否安装成功

### 初始化所有节点的mon服务（主机名解析必须对）

**[root@node1 ceph-cluster]# ceph-deploy mon create-initial**

#启动所有集群的服务

**[root@node1 ceph-cluster]# systemctl status ceph-mon.target**  #查看mon是否正常

[root@node111 ceph-cluster]# ceph -s #查看

cluster de2d1fda-8abc-41c9-8e36-fb3579a1976a

health HEALTH\_ERR

clock skew detected on mon.node222

64 pgs are stuck inactive for more than 300 seconds

64 pgs stuck inactive

no osds

Monitor clock skew detected

monmap e1: 3 mons at {node111=192.168.4.11:6789/0,node222=192.168.4.12:6789/0,node333=192.168.4.13:6789/0}

election epoch 8, quorum 0,1,2 node111,node222,node333

osdmap e1: 0 osds: 0 up, 0 in #存储盘

flags sortbitwise

pgmap v2: 64 pgs, 1 pools, 0 bytes data, 0 objects

0 kB used, 0 kB / 0 kB avail

##3个mons osds:0个

常见错误及解决方法（非必要操作，有错误可以参考）：

如果提示如下错误信息：

[node1][ERROR ] admin\_socket: exception getting command descriptions: [Error 2] No such file or directory

解决方案如下（在node1操作）：

先检查自己的命令是否是在ceph-cluster目录下执行的！！！！如果时确认是在该目录下执行的create-initial命令，依然保存，可以使用如下方式修复。

[root@node1 ceph-cluster]# vim ceph.conf #文件最后追加以下内容

public\_network = 192.168.4.0/24

修改后重新推送配置文件:

**[root@node1 ceph-cluster]# ceph-deploy --overwrite-conf config push node1 node2 node3**

## 步骤三：创建OSD

### 1）准备磁盘分区（node1、node2、node3都做相同操作）

[root@node111 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm [SWAP]

vdb 252:16 0 20G 0 disk

vdc 252:32 0 20G 0 disk

vdd 252:48 0 20G 0 disk

[root@node111 ceph-cluster]# parted /dev/vdb mklabel gpt

信息: You may need to update /etc/fstab.

[root@node111 ceph-cluster]# parted /dev/vdb mkpart primary 1M 50%

信息: You may need to update /etc/fstab.

[root@node111 ceph-cluster]# parted /dev/vdb mkpart primary 50% 100%

信息: You may need to update /etc/fstab.

[root@node111 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm [SWAP]

vdb 252:16 0 20G 0 disk

├─vdb1 252:17 0 10G 0 part

└─vdb2 252:18 0 10G 0 part

vdc 252:32 0 20G 0 disk

vdd 252:48 0 20G 0 disk

临时修改权限：

[root@node1 ceph-cluster]# chown ceph.ceph /dev/vdb1

[root@node1 ceph-cluster]# chown ceph.ceph /dev/vdb2

#给这两个分区赋予权限,用.和:作用一样

//这两个分区用来做存储服务器的日志journal盘

永久修改权限：新建70-vdb.rules文件

Udev文件:可以更改设备名(如网卡名),做链接.

[root@node1 ceph-cluster]# vim /etc/udev/rules.d/70-vdb.rules

ENV{DEVNAME}=="/dev/vdb1",OWNER="ceph",GROUP="ceph"

ENV{DEVNAME}=="/dev/vdb2",OWNER="ceph",GROUP="ceph"

解释:如果(ENV)发现(DEVBNAME)这个”/dev/vdb2”硬件,就将所有者改为”ceph”,所属组改为”ceph”使得权限永久生效

### 2）初始化清空磁盘数据（仅node1操作即可）

[root@node1 ceph-cluster]# ceph-deploy disk zap node1:vdc node1:vdd

[root@node1 ceph-cluster]# ceph-deploy disk zap node2:vdc node2:vdd

[root@node1 ceph-cluster]# ceph-deploy disk zap node3:vdc node3:vdd

### 创建OSD存储空间（仅node1操作即可）

可在此处增加磁盘

[root@node1 ceph-cluster]# ceph-deploy osd create \

node1: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缓存，

vdd为集群提供存储空间，vdb1提供JOURNAL缓存，

//一个存储设备对应一个缓存设备，缓存需要SSD，不需要很大

#如果没有SSD(固态硬盘)做缓存,就只要 执行(ceph-deploy osd create node1:vdc node1:vdd)即可

[root@node1 ceph-cluster]# ceph-deploy osd create \

node2:vdc:/dev/vdb1 node2:vdd:/dev/vdb2

[root@node1 ceph-cluster]# ceph-deploy osd create \

node3:vdc:/dev/vdb1 node3:vdd:/dev/vdb2

[root@node1 ~]# ceph osd tree

ID WEIGHT TYPE NAME UP/DOWN REWEIGHT PRIMARY-AFFINITY

-1 0.11691 root default

-2 0.03897 host node1

0 0.01949 osd.0 down 0 1.00000

1 0.01949 osd.1 down 0 1.00000

-3 0.03897 host node2

2 0.01949 osd.2 up 1.00000 1.00000

3 0.01949 osd.3 up 1.00000 1.00000

-4 0.03897 host node3

4 0.01949 osd.4 up 1.00000 1.00000

5 0.01949 osd.5 up 1.00000 1.00000

[root@node1 ~]# systemctl restart ceph-osd.target

[root@node1 ~]# ceph osd tree

ID WEIGHT TYPE NAME UP/DOWN REWEIGHT PRIMARY-AFFINITY

-1 0.11691 root default

-2 0.03897 host node1

0 0.01949 osd.0 up 1.00000 1.00000

1 0.01949 osd.1 up 1.00000 1.00000

-3 0.03897 host node2

2 0.01949 osd.2 up 1.00000 1.00000

3 0.01949 osd.3 up 1.00000 1.00000

-4 0.03897 host node3

4 0.01949 osd.4 up 1.00000 1.00000

5 0.01949 osd.5 up 1.00000 1.00000

### 4）常见错误（非必须操作）

使用osd create创建OSD存储空间时，如提示run 'gatherkeys'，可以使用如下命令修复：

[root@node1 ceph-cluster]# ceph-deploy gatherkeys node1 node2 node3

## 步骤四：验证测试

### 1) 查看集群状态

[root@node111 ceph-cluster]# ceph -s

cluster de2d1fda-8abc-41c9-8e36-fb3579a1976a

health HEALTH\_WARN #WARN是警告 OK完全正常

clock skew detected on mon.node222 #提示node222时间不对,没同步

15 pgs peering

Monitor clock skew detected

monmap e1: 3 mons at {node111=192.168.4.11:6789/0,node222=192.168.4.12:6789/0,node333=192.168.4.13:6789/0}

election epoch 8, quorum 0,1,2 node111,node222,node333

osdmap e32: 6 osds: 6 up, 6 in

flags sortbitwise

pgmap v69: 64 pgs, 1 pools, 0 bytes data, 0 objects

203 MB used, 119 GB / 119 GB avail

39 active+clean

10 activating

10 peering

5 remapped+peering

[root@node111 ceph-cluster]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm [SWAP]

vdb 252:16 0 20G 0 disk

├─vdb1 252:17 0 10G 0 part

└─vdb2 252:18 0 10G 0 part

vdc 252:32 0 20G 0 disk

└─vdc1 252:33 0 20G 0 part /var/lib/ceph/osd/ceph-0

vdd 252:48 0 20G 0 disk

└─vdd1 252:49 0 20G 0 part /var/lib/ceph/osd/ceph-1

### 2）常见错误（非必须操作）

如果查看状态包含如下信息：

health: HEALTH\_WARN

clock skew detected on node2, node3…

clock skew表示时间不同步，解决办法：请先将所有主机的时间都使用NTP时间同步！！！

Ceph要求所有主机时差不能超过0.05s，否则就会提示WARN，如果使用NTP还不能精确同步时间，可以手动修改所有主机的ceph.conf，在[MON]下面添加如下一行：

mon clock drift allowed = 1

如果状态还是失败，可以尝试执行如下命令，重启ceph服务：

[root@node1 ~]# systemctl restart ceph\\*.service ceph\\*.target

Ceph集群创建后之后,可以在任意集群节点上执行ceph命令

至此：前面实验完成了ODS和Monitor

OSD ：存储设备,负责真正的存储空间, 软件包：ceph-osd

Monitor ：集群监控组件，利用Crush算法分配存储,要求至少要做三台

软件包：ceph-mon

接下来是客户端共享使用

# 3 案例3：创建Ceph块存储

3.1 问题

沿用练习一，使用Ceph集群的块存储功能，实现以下目标：

* 创建块存储镜像
* 客户端映射镜像
* 创建镜像快照
* 使用快照还原数据
* 使用快照克隆镜像
* 删除快照与镜像

创建共享盘(image镜像),镜像必须放在共享池,默认有个rbd共享池

共享池内:

Image镜像1(共享磁盘)--/dev/存储设备

Image镜像2(共享磁盘)--/dev/存储设备

Image镜像3(共享磁盘)--/dev/存储设备

## 步骤一：创建镜像

### 查看存储池。

[root@node1 ~]# ceph osd lspools

0 rbd,

### 创建镜像、查看镜像

demo-image功能 --image-feature 支持快照

[root@node1 ~]# rbd create demo-image --image-feature layering --size 10G

[root@node1 ~]# rbd create rbd/image --image-feature layering --size 10G

[root@node1 ~]# rbd list #查看已创建镜像文件名(node2,node3都能查看到)

demo-image #创建的两个镜像

Image

[root@node1 ~]# rbd info demo-image #查看demo-image镜像共享的相信信息

rbd image 'demo-image':

size 10240 MB in 2560 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.d3aa2ae8944a

format: 2

features: layering

## 步骤二：容量动态调整

### 1）缩小容量

[root@node1 ~]# rbd resize --size 7G image --allow-shrink

#--allow-shrink 作用:防止缩小容量发生事故

[root@node1 ~]# rbd info image

rbd image 'image':

size 7168 MB in 1792 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.1032238e1f29

format: 2

features: layering

flags:

### 2）扩容容量

[root@node1 ~]# rbd resize --size 15G image

[root@node1 ~]# rbd info image

rbd image 'image':

size 15360 MB in 3840 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.1032238e1f29

format: 2

features: layering

flags:

## 步骤三：通过KRBD访问

### 1）集群内将镜像映射为本地磁盘(本试验中未做)

[root@node1 ~]# rbd map demo-image #挂载使用

/dev/rbd0

[root@node1 ~]# lsblk

… …

rbd0 251:0 0 10G 0 disk

[root@node1 ~]# mkfs.xfs /dev/rbd0

[root@node1 ~]# mount /dev/rbd0 /mnt

集群内有ceph.client.admin.keyring 文件.里面有用户名和密码,可直接挂载使用

[root@node1 ceph-cluster]# rbd unmap demo-image #卸载镜像

### 2）客户端通过KRBD访问

#客户端需要安装ceph-common软件包

#拷贝配置文件（否则不知道集群在哪）

#拷贝连接密钥（否则无连接权限）

[root@client ~]# yum -y install ceph-common

[root@client ~]# ls /etc/ceph #安装完之后有rbdmap 程序

Rbdmap

[root@client ~]# scp 192.168.4.11:/etc/ceph/ceph.conf /etc/ceph/

#拷贝配置文件（否则不知道集群在哪）

[root@client ~]# ls /etc/ceph

ceph.conf rbdmap

[root@client ~]# scp 192.168.4.11:/etc/ceph/ceph.client.admin.keyring \

/etc/ceph/

#拷贝连接密钥文件（否则无连接权限）

[root@client ~]# ls /etc/ceph/

ceph.client.admin.keyring ceph.conf rbdmap

[root@client ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm [SWAP]

[root@client ~]# rbd map image #添加镜像盘image,前面创建了两个

/dev/rbd0

[root@client ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm [SWAP]

rbd0 251:0 0 15G 0 disk #多了rbd0

[root@client ~]# rbd showmapped

id pool image snap device

0 rbd image - /dev/rbd0

### 3) 客户端格式化、挂载分区

[root@client ~]# mkfs.xfs /dev/rbd0

[root@client ~]# mount /dev/rbd0 /mnt/

[root@client ~]# echo "test" > /mnt/test.txt

[root@client ~]# umount /mnt #卸载挂载

[root@client ~]# rbd unmap image #卸载镜像

ceph快照:基于COW(copy in write)---写时复制

例如:

原始磁盘100G:

## 步骤四：创建镜像快照

防止误删数据，有3副本是无法解决人为修改数据，

### 1) 查看镜像快照

[root@node1 ~]# rbd snap ls image

### 2) 创建镜像快照

[root@node1 ~]# rbd snap create image --snap image-snap1

[root@node1 ~]# rbd snap ls image

SNAPID NAME SIZE

4 image-snap1 15360 MB

#针对image镜像做的快照

### 3) 删除客户端写入的测试文件

[root@client ~]# rm -rf /mnt/test.txt

### 还原快照

Ceph不支持在线还原

[root@node1 ~]# rbd snap rollback image --snap image-snap1

#客户端重新挂载分区

[root@client ~]# umount /mnt

[root@client ~]# mount /dev/rbd0 /mnt/

[root@client ~]# ls /mnt

### 5) 实验中遇到错误

[root@client ~]# mount /dev/rbd0 /mnt

mount: 文件系统类型错误、选项错误、/dev/rbd0 上有坏超级块、

缺少代码页或助手程序，或其他错误

有些情况下在 syslog 中可以找到一些有用信息- 请尝试

dmesg | tail 这样的命令看看。

解决:在集群中重新执行快照还原

[root@node1 ~]# rbd snap rollback image --snap image-snap1

[root@client ~]# mount /dev/rbd0 /mnt

## 步骤五：创建快照克隆

### 1）克隆快照

[root@node111 ~]# rbd snap ls image

SNAPID NAME SIZE

4 image-snap1 15360 MB

[root@node1 ~]#rbd snap protect image --snap image-snap1

#保护image-snap1镜像,防止误删,导致整个镜像损坏

[root@node1 ~]#rbd snap unprotect image --snap image-snap1

#取消保护image-snap1(本实验此时不操作)

[root@node1 ~]# rbd snap rm image --snap image-snap1

#前面做了保护,此时删除镜像会失败

[root@node1 ~]# rbd clone image --snap image-snap1 image-clone --image-feature layering

//使用镜像image的快照image-snap1克隆一个新的镜像image-clone

### 查看克隆镜像与父镜像快照的关系

[root@node111 ~]# rbd list

demo-image

image

image-clone #多了image-clone镜像

[root@node1 ~]#rbd info image-clone

rbd image 'image-clone':

size 15360 MB in 3840 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.d3f53d1b58ba

format: 2

features: layering

flags:

parent: rbd/image@image-snap1 #这个镜像的父镜像是image@image-snap1

overlap: 15360 MB

#克隆镜像很多数据都来自于快照链

#如果希望克隆镜像可以独立工作，就需要将父快照中的数据，全部拷贝一份，但比较耗时！！！

[root@node1 ~]#rbd flatten image-clone

#将父镜像image@image-snap1的数据全部拷贝一份到image-clone上

[root@node1 ~]#rbd info image-clone

rbd image 'image-clone':

size 15360 MB in 3840 objects

order 22 (4096 kB objects)

block\_name\_prefix: rbd\_data.d3f53d1b58ba

format: 2

features: layering

flags:

#注意，父快照信息没了！

## 步骤六：其他操作

### 1） 客户端撤销磁盘映射

[root@client ~]# umount /mnt

[root@client ~]# rbd showmapped

id pool image snap device

0 rbd image - /dev/rbd0

//语法格式:

[root@client ~]# rbd unmap /dev/rbd0

### 2）删除快照与镜像

[root@node1 ~]# rbd snap rm image --snap image-snap

[root@node1 ~]# rbd list

[root@node1 ~]# rbd rm image

### 3) 查询

[root@node333 ~]# ps aux |grep osd

ceph 1448 0.2 4.5 905268 46624 ? Ssl 09:00 0:12 /usr/bin/ceph-osd -f --cluster ceph --id 4 --setuser ceph --setgroup ceph

ceph 1673 0.2 4.7 908256 48480 ? Ssl 09:00 0:12 /usr/bin/ceph-osd -f --cluster ceph --id 5 --setuser ceph --setgroup ceph

root 3755 0.0 0.0 112676 980 pts/0 S+ 10:21 0:00 grep --color=auto osd

# NSD CLUSTER DAY05

案例1：块存储应用案例

案例2：Ceph文件系统

案例3：创建对象存储服务器

# 虚拟化

(虚拟机):镜像文件+配置文件

真机上创建了一个20G的zdd虚拟机，如何安装操作系统选用网络，一路默认，强制关机



### 虚拟机镜像文件:

[root@room9pc01 ~]# cd /var/lib/libvirt/images/

[root@room9pc01 images]# ls -h zdd.qcow2 #创建虚拟机后会产生一个\*.qcow2的文件

zdd.qcow2

[root@room9pc01 images]# ll -h zdd.qcow2

-rw------- 1 root root 21G 2月 1 10:29 zdd.qcow2

[root@room9pc01 images]# du -h zdd.qcow2 #查看文件大小

3.4M zdd.qcow2

### 虚拟机配置文件:

[root@room9pc01 images]# cd /etc/libvirt/qemu/

[root@room9pc01 qemu]# ls zdd.xml

zdd.xml

[root@room9pc01 qemu]# vim zdd.xml

#可修改参数比如修改CPU个数

#将这两个文件拷贝到新的真机上,将会增加一个虚拟机

## 真机实际操作

### 1:虚拟机镜像快照

[root@room9pc01 ~]# cd /var/lib/libvirt/images/

[root@room9pc01 images]# ls .rh7\_template.img

.rh7\_template.img

[root@room9pc01 images]# ls .rhel7.xml

.rhel7.xml

[root@room9pc01 images]# qemu-img create -f qcow2 \ #创建快照 #类型是qcow2

-b .rh7\_template.img newzdd.qcow2 10G

#模板是.rh7\_template.imgc创建出来的新快照是newzdd.qcow2且大小是10G

打出命令后显示的内容如下:

Formatting 'newzdd.qcow2', fmt=qcow2 size=10737418240 backing\_file='.rh7\_template.img' encryption=off cluster\_size=65536 lazy\_refcounts=off

[root@room9pc01 images]# ls -lh newzdd.qcow2 #查看快照

-rw-r--r-- 1 root root 193K 2月 1 11:22 newzdd.qcow2

#虽然只有193k 但是启用这个

### 2:拷贝配置文件并修改

[root@room9pc01 images]# cp .rhel7.xml /etc/libvirt/qemu/newzdd.xml

[root@room9pc01 images]# cd /etc/libvirt/qemu/

[root@room9pc01 qemu]# ls

A-1-clinet-192.168.4.10.xml A-3-Web111-192.168.2.100.xml B-4.xml C-3-11.xml desktop.xml node1-192.168.4.11.xml server.xml

A-2-proxy-192.168.4.5.xml A-4-Web222-192.168.2.200.xml C-1-09.xml C-4-12.xml networks node2-192.168.4.12.xml win2008.xml

A-2-proxy-192.168.4.6.xml B-3.xml C-2-10.xml classroom.xml newzdd.xml node3-192.168.4.13.xml zdd.xml

[root@room9pc01 qemu]# vim newzdd.xml

所改内容:

<name>newzdd</name>

<devices>

<emulator>/usr/libexec/qemu-kvm</emulator>

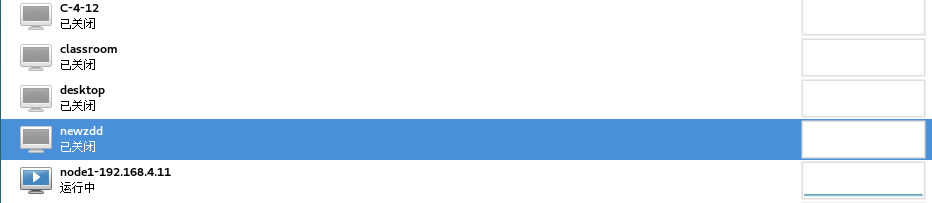
<disk type='file' device='disk'>

<driver name='qemu' type='qcow2'/>

<source file='/var/lib/libvirt/images/newzdd.qcow2'/>

[root@room9pc01 qemu]# virsh define /etc/libvirt/qemu/newzdd.xml #加载

定义域 newzdd（从 /etc/libvirt/qemu/newzdd.xml）



新建虚拟机.虚拟机的磁盘是真实机上的一个文件

新建虚拟机.虚拟机的磁盘调用的是ceph的镜像image

# 1 案例1：块存储应用案例

1.1 问题

延续Day04的实验内容，演示块存储在KVM虚拟化中的应用案例，实现以下功能：

* Ceph创建块存储镜像
* 客户端安装部署ceph软件
* 客户端部署虚拟机
* 客户端创建secret
* 设置虚拟机配置文件，调用ceph存储

1.2 方案

使用Ceph存储创建镜像。

KVM虚拟机调用Ceph镜像作为虚拟机的磁盘。

## 1）创建磁盘镜像。

[root@node1 ~]# rbd create vm1-image --image-feature layering --size 10G

[root@node1 ~]# rbd create vm2-image --image-feature layering --size 10G

[root@node1 ~]# rbd list

[root@node1 ~]# rbd info vm1-image

[root@node1 ~]# qemu-img info rbd:rbd/vm1-image

image: rbd:rbd/vm1-image

file format: raw

virtual size: 10G (10737418240 bytes)

disk size: unavailable

## 2）确认Ceph认证账户。

Ceph默认开启用户认证，客户端需要账户才可以访问，

默认账户名称为client.admin，key是账户的密钥，

可以使用ceph auth添加新账户（案例我们使用默认账户）。

[root@node1 ~]# cat /etc/ceph/ceph.conf //配置文件

[global]

mon\_initial\_members = node1, node2, node3

mon\_host = 192.168.2.10,192.168.2.20,192.168.2.30

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

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

auth\_client\_required = cephx //开启认证

[root@node1 ~]# cat /etc/ceph/ceph.client.admin.keyring //账户文件

[client.admin]

key = AQBTsdRapUxBKRAANXtteNUyoEmQHveb75bISg==

## 3）部署客户端环境。

注意：这里使用真实机当客户端！！！

客户端需要安装ceph-common软件包，拷贝配置文件（否则不知道集群在哪），

拷贝连接密钥（否则无连接权限）。

[root@room9pc01 ~]# yum -y install ceph-common

[root@room9pc01 ~]# scp 192.168.4.11:/etc/ceph/ceph.conf /etc/ceph/

[root@room9pc01 ~]# scp 192.168.4.11:/etc/ceph/ceph.client.admin.keyring \

/etc/ceph/

## 4）创建KVM虚拟机。

使用virt-manager创建2台普通的KVM虚拟机。

## 5）配置libvirt secret。

编写账户信息文件（真实机操作）

[root@room9pc01 ~]# vim secret.xml //新建临时文件，内容如下

<secret ephemeral='no' private='no'>

<usage type='ceph'>

<name>client.admin secret</name>

</usage>

</secret>

#使用XML配置文件创建secret

[root@room9pc01 ~]# virsh secret-define --file secret.xml

生成 14f1c616-3945-4058-a09f-8776ecbef010

//随机的UUID，这个UUID对应的有账户信息

编写账户信息文件（真实机操作）

[root@room9pc01 ~]# cat /etc/ceph/ceph.client.admin.keyring #安装ceph即生成

[client.admin]

key = AQCikFJcZUi1BhAAO3vuNjbMl8IE6VJh2xmssA==

设置secret，添加账户的密钥

[root@room9pc01] virsh secret-set-value \

--secret 14f1c616-3945-4058-a09f-8776ecbef010 \

--base64 AQCikFJcZUi1BhAAO3vuNjbMl8IE6VJh2xmssA==

secret 值设定

//这里secret后面是之前创建的secret的UUID

//base64后面是client.admin账户的密码

//现在secret中既有账户信息又有密钥信息

## 虚拟机的XML配置文件。

[root@room9pc01 ~]# vim /etc/libvirt/qemu/vm1.xml

#编辑vm1虚拟机的配置文件.改完之后重启生效

[root@room9pc01 ~]# virsh edit vm1

#编辑vm1虚拟机的配置文件.改完立即生效每个虚拟机都会有一个XML配置文件，包括：虚拟机的名称、内存、CPU、磁盘、网卡等信息

[root@room9pc01 ~]# vim /etc/libvirt/qemu/vm1.xml

//修改前内容如下

<disk type='file' device='disk'>

<driver name='qemu' type='qcow2'/>

<source file='/var/lib/libvirt/images/vm1.qcow2'/>

<target dev='vda' bus='virtio'/>

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

</disk>

#不推荐直接使用vim修改配置文件，推荐使用virsh edit修改配置文件，

//修改后效果如下：

[root@room9pc01] virsh edit vm1 #vm1为虚拟机名称

<disk type='network' device='disk'> #磁盘类型为网络盘

<driver name='qemu' type='raw'/> #raw:表示共享盘为裸盘

<auth username='admin'>

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

</auth>

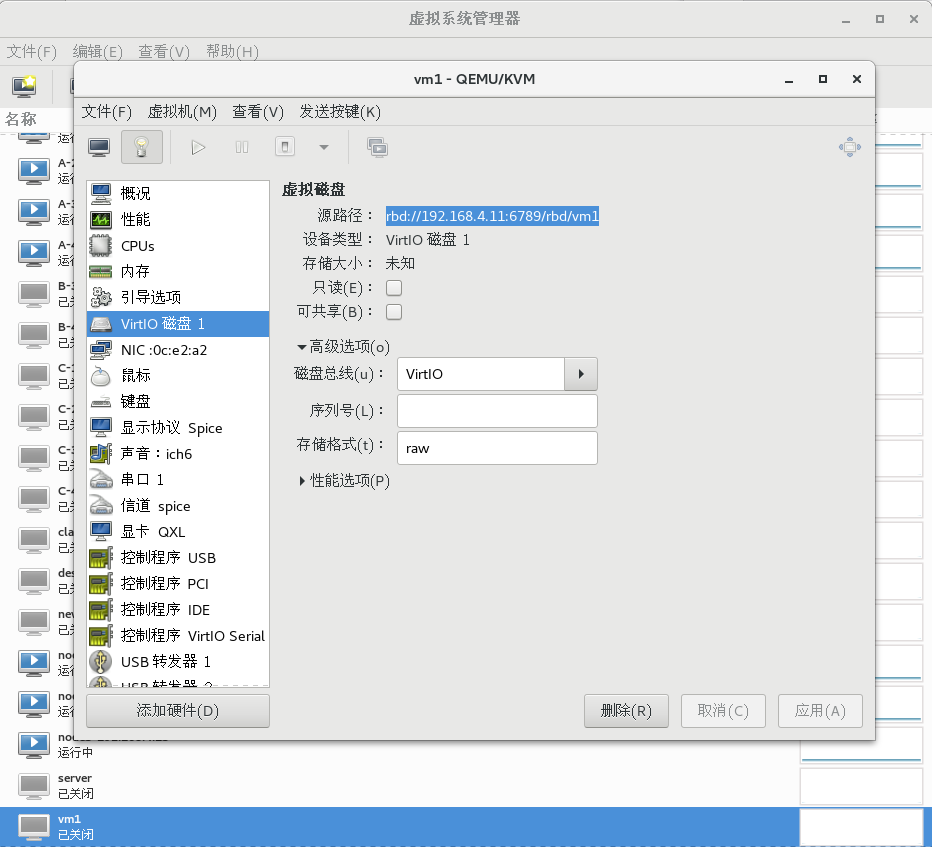
<source protocol='rbd' name='rbd/vm1-image'>  
 <host name='192.168.4.11' port='6789'/>  
 </source>

<target dev='vda' bus='virtio'/> #vda磁盘名可以改

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

</disk>

最终效果:虚拟机磁盘的源路径为网络盘



# 2 案例2：Ceph文件系统

## 文件系统:

磁盘--分区--格式(文件系统)

文件系统组成:inode,block

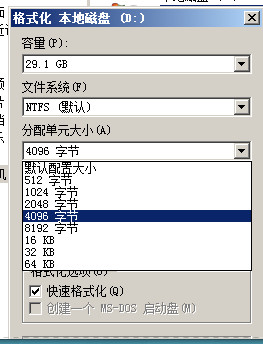
inode 存放数据描述信息和数据存放地方, 例如:数据放在block哪些格子里面

block 存放数据,文件类型定义了格子大小。

常见的文件系统的类型：

inode删除标记:文件数据执行删除操作之后,inode的描述信息被打上删除标记,真正的数据却没有被删除,原存放这部分数据的block区域变成可用空间.可以重新使用,此时可做数据恢复.如果被存放了另外的东西覆盖了这些区域,数据就真没有了.

存储单元:linux与windows 默认4096字节=4k



2.1 问题

延续前面的实验，实现Ceph文件系统的功能。具体实现有以下功能：

部署MDSs节点 专门用来做文件系统，实际环境中建议新建一台做MDS

创建Ceph文件系统

客户端挂载文件系统

2.2 方案

添加一台虚拟机，部署MDS节点。

主机的主机名及对应的IP地址如表-1所示。

表－1 主机名称及对应IP地址表

Node4 192.168.4.14

## 1）添加一台新的虚拟机，要求如下：

IP地址:192.168.4.14

主机名:node4

配置yum源（包括rhel、ceph的源）

与Client主机同步时间

node1允许无密码远程node4

## 2）部署元数据服务器

登陆node4，安装ceph-mds软件包

[root@node4 ~]# yum -y install ceph-mds

登陆node1部署节点操作

[root@node1 ~]# cd /root/ceph-cluster

//该目录，是最早部署ceph集群时，创建的目录

[root@node1 ceph-cluster]# ceph-deploy mds create node4

//给nod4拷贝配置文件，启动mds服务

同步配置文件和key

[root@node1 ceph-cluster]# ceph-deploy admin node4

## 3）创建存储池

[root@node4 ~]# ceph osd pool create cephfs\_data 128

//创建存储池，对应128个PG

[root@node4 ~]# ceph osd pool create cephfs\_metadata 128

//创建存储池，对应128个PG

[root@node4 ~]# ceph osd lspools

//查看存储池

## 5）创建Ceph文件系统

[root@node4 ~]# ceph mds stat //查看mds状态

e2:, 1 up:standby

**[root@node4 ~]# ceph fs new myfs1 cephfs\_metadata cephfs\_data**

内容显示：new fs with metadata pool 2 and data pool 1

//注意，先写metadata池，再写data池

//默认，只能创建1个文件系统，多余的会报错

[root@node4 ~]# ceph fs ls

name: myfs1, metadata pool: cephfs\_metadata, data pools: [cephfs\_data ]

[root@node4 ~]# ceph mds stat

e4: 1/1/1 up {0=node4=up:creating}

## 6）客户端挂载

[root@client ~]# mount -t ceph 192.168.4.11:6789:/ /mnt/cephfs/ \

-o name=admin,secret=AQBTsdRapUxBKRAANXtteNUyoEmQHveb75bISg==

//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

# 3 案例3：创建对象存储服务器

3.1 问题

延续前面的实验，实现Ceph对象存储的功能。具体实现有以下功能：

安装部署Rados Gateway

启动RGW服务

设置RGW的前端服务与端口

客户端测试

3.2 步骤

## 步骤一：部署对象存储服务器

### 1）准备实验环境，要求如下：

IP地址:192.168.4.15

主机名:node5

配置yum源（包括rhel、ceph的源）

与Client主机同步时间

node1允许无密码远程node5

修改node1的/etc/hosts，并同步到所有node主机

### 2）部署RGW软件包

[root@node1 ~]# ceph-deploy install --rgw node5

同步配置文件与密钥到node5

[root@node1 ~]# cd /root/ceph-cluster

[root@node1 ~]# ceph-deploy admin node5

### 3）新建网关实例

启动一个rgw服务

[root@node1 ~]# ceph-deploy rgw create node5

登陆node5验证服务是否启动

[root@node5 ~]# ps aux |grep radosgw

ceph 4109 0.2 1.4 2289196 14972 ? Ssl 22:53 0:00 /usr/bin/radosgw -f --cluster ceph --name client.rgw.node4 --setuser ceph --setgroup ceph

[root@node5 ~]# systemctl status ceph-radosgw@\\*

### 4）修改服务端口

登陆node5，RGW默认服务端口为7480，修改为8000或80更方便客户端记忆和使用

[root@node5 ~]# vim /etc/ceph/ceph.conf

[client.rgw.node5]

host = node5

rgw\_frontends = "civetweb port=8000"

//node5为主机名

//civetweb是RGW内置的一个web服务

## 步骤二：客户端测试

### 1）curl测试

[root@client ~]# curl 192.168.4.15:8000

<?xml version="1.0" encoding="UTF-8"?><ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"><Owner><ID>anonymous</ID><DisplayName></DisplayName></Owner><Buckets></Buckets></ListAllMyBucketsResult>

### 2）使用第三方软件访问

登陆node5（RGW）创建账户

[root@node5 ~]# radosgw-admin user create \

--uid="testuser" --display-name="First User"

… …

"keys": [

{

"user": "testuser",

"access\_key": "5E42OEGB1M95Y49IBG7B",

"secret\_key": "i8YtM8cs7QDCK3rTRopb0TTPBFJVXdEryRbeLGK6"

}

],

... ...

#

[root@node5 ~]# radosgw-admin user info --uid=testuser

//testuser为用户，key是账户访问密钥

### 3）客户端安装软件

[root@client ~]# yum install s3cmd-2.0.1-1.el7.noarch.rpm

修改软件配置（注意，除了下面设置的内容，其他提示都默认回车）

[root@client ~]# s3cmd --configure

Access Key: 5E42OEGB1M95Y49IBG7B  
Secret Key: i8YtM8cs7QDCK3rTRopb0TTPBFJVXdEryRbeLGK6

S3 Endpoint [s3.amazonaws.com]: 192.168.4.15:8000

[%(bucket)s.s3.amazonaws.com]: %(bucket)s.192.168.4.15:8000

Use HTTPS protocol [Yes]: No

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

Save settings? [y/N] y

//注意，其他提示都默认回车

### 4）创建存储数据的bucket（类似于存储数据的目录）

[root@client ~]# s3cmd ls

[root@client ~]# s3cmd mb s3://my\_bucket

Bucket 's3://my\_bucket/' created

[root@client ~]# s3cmd ls

2018-05-09 08:14 s3://my\_bucket

[root@client ~]# s3cmd put /var/log/messages s3://my\_bucket/log/

[root@client ~]# s3cmd ls

2018-05-09 08:14 s3://my\_bucket

[root@client ~]# s3cmd ls s3://my\_bucket

DIR s3://my\_bucket/log/

[root@client ~]# s3cmd ls s3://my\_bucket/log/

2018-05-09 08:19 309034 s3://my\_bucket/log/messages

测试下载功能

[root@client ~]# s3cmd get s3://my\_bucket/log/messages /tmp/

测试删除功能

[root@client ~]# s3cmd del s3://my\_bucket/log/messages

# 4 案例2Ceph文件系统实际实验

2.1 问题

延续前面的实验，实现Ceph文件系统的功能。具体实现有以下功能：

部署MDSs节点

创建Ceph文件系统

客户端挂载文件系统

2.2 方案

添加一台虚拟机，部署MDS节点。

主机的主机名及对应的IP地址如表-1所示。

表－1 主机名称及对应IP地址表

Node4 192.168.4.14

## 1）添加一台新的虚拟机，要求如下：

以下第四天实验已安装:(如果新环境需要重新做)

IP地址:192.168.4.13 #实际中用的是node3

主机名:node3

配置yum源（包括rhel、ceph的源）

与Client主机同步时间

node1允许无密码远程node3

## 2）部署元数据服务器

登陆node3，安装ceph-mds软件包

[root@node3 ~]# yum -y install ceph-mds

登陆node1部署节点操作

[root@node1 ~]# cd /root/ceph-cluster

//该目录，是最早部署ceph集群时，创建的目录

[root@node1 ceph-cluster]# ceph-deploy mds create node3

//给nod4拷贝配置文件，启动mds服务

同步配置文件和key

[root@node1 ceph-cluster]# ceph-deploy admin node3 #之前已经做了

## 3）创建存储池

不创建就默认有个0 rdb存储池

[root@node333 ~]# ceph osd lspools

0 rbd,

创建名为:cephfs\_data的存储池

[root@node3 ~]# ceph osd pool create cephfs\_data 128

//创建存储池，无固定的值,尽量是2的次方.推荐是32-128之间.对应128个PG

创建名为:cephfs\_metadata的存储池

[root@node3~]# ceph osd pool create cephfs\_metadata 128

//创建存储池，对应128个PG

[root@node333 ~]# ceph osd lspools

0 rbd,1 cephfs\_data,2 cephfs\_metadata,

## 4）创建Ceph文件系统

[root@node4 ~]# ceph mds stat //查看mds状态

e2:, 1 up:standby

[root@node4 ~]# ceph fs new myfs1 cephfs\_metadata cephfs\_data

new fs with metadata pool 2 and data pool 1

#用两个池子创建了文件系统

//注意，先写medadata池，再写data池

//默认，只能创建1个文件系统，多余的会报错

[root@node4 ~]# ceph fs ls #查看

name: myfs1, metadata pool: cephfs\_metadata, data pools: [cephfs\_data ]

[root@node4 ~]# ceph mds stat

e4: 1/1/1 up {0=node4=up:creating}

## 客户端挂载

[root@client ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sr0 11:0 1 1024M 0 rom

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

└─vda2 252:2 0 19G 0 part

├─rhel-root 253:0 0 17G 0 lvm /

└─rhel-swap 253:1 0 2G 0 lvm

[root@client ~]# df -h

文件系统 容量 已用 可用 已用% 挂载点

/dev/mapper/rhel-root 17G 3.5G 14G 21% /

devtmpfs 481M 0 481M 0% /dev

tmpfs 497M 0 497M 0% /dev/shm

tmpfs 497M 7.0M 490M 2% /run

tmpfs 497M 0 497M 0% /sys/fs/cgroup

/dev/vda1 1014M 161M 854M 16% /boot

tmpfs 100M 0 100M 0% /run/user/0

[root@client ~]# mount -t ceph 192.168.4.11:6789:/ /mnt/cephfs/ \

-o name=admin,secret=AQBTsdRapUxBKRAANXtteNUyoEmQHveb75bISg==

-t 类型 -o 用户名密码

//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node3的/etc/ceph/ceph.client.admin.keyring中找到

[root@client ~]# df -h

文件系统 容量 已用 可用 已用% 挂载点

/dev/mapper/rhel-root 17G 3.5G 14G 21% /

devtmpfs 481M 0 481M 0% /dev

tmpfs 497M 0 497M 0% /dev/shm

tmpfs 497M 7.0M 490M 2% /run

tmpfs 497M 0 497M 0% /sys/fs/cgroup

/dev/vda1 1014M 161M 854M 16% /boot

tmpfs 100M 0 100M 0% /run/user/0

192.168.4.13:6789:/ 120G 552M 120G 1% /mnt

查询

[root@node1 ~]# ceph -s