### 存储服务操作手册

**前言：**

1.蓝色字体表示命令行命令，正式执行时不要复制前面的#号，#号只是提示应该使用root权限操作

2.绿色字体表示注释，有时注释太多就不用绿色表示了

3.注意：本文档的所有操作请先在在测环境进行实践，请不要直接在真实的服务器中操作！

**版权声明**：

本文档以开源的形式发布，所有条款如下：

1. 无担保：作者不保证文档内容的准确无误，亦不承担由于使用此文档所导致的任何后果

2. 自由使用：任何人可以出于任何目的而自由地 阅读/链接/打印/转载/引用/再创作 此文档，无需任何附加条件

若您 阅读/链接/打印/转载/引用/再创作 本文档，则说明接受以上2个条款。

作者：李茂福

时间：2022-05-18

**章0、网络存储类型**

网络存储类型：

|  |  |
| --- | --- |
| 存储服务 | 存储类型 |
| NFS | 目录共享（文件系统） |
| Samba | 目录共享 |
| iSCSI | 磁盘分区、lvm逻辑卷、文件io |
|  |  |

**章一、NFS**

NFS（Network File System）没有认证机制，可以限制客户端ip，读写权限

**①服务端**

**★安装nfs server**

# yum install rpcbind nfs-utils #安装

# systemctl enable rpcbind

# systemctl enable nfs

# systemctl enable nfs-server

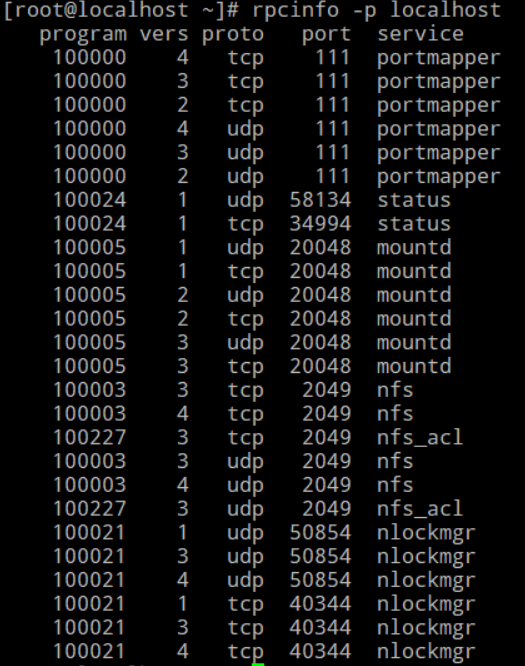
# systemctl start rpcbind #只能先启动rpcbind

# systemctl start nfs

# systemctl start nfs-server

**★修改监听端口**

# rpcinfo -p localhost #查看RPC服务端口注册状况



# cat /etc/sysconfig/nfs | grep -v '^#' #查看默认配置

# cat > /etc/sysconfig/nfs <<EOF #修改配置为以下内容

RPCNFSDARGS=""

RPCMOUNTDOPTS="-p 30003"

STATDARG="-p 30004"

SMNOTIFYARGS=""

RPCIDMAPDARGS=""

RPCGSSDARGS=""

GSS\_USE\_PROXY="yes"

BLKMAPDARGS=""

MOUNTD\_PORT=20048

RPCRQUOTADOPTS="-p 30001"

LOCKD\_TCPPORT=30002

LOCKD\_UDPPORT=30002

# STATD\_PORT=30004 #同 STATDARG="-p 30004"

# RQUOTAD\_PORT=30001 #同 RPCRQUOTADOPTS="-p 30001"

EOF

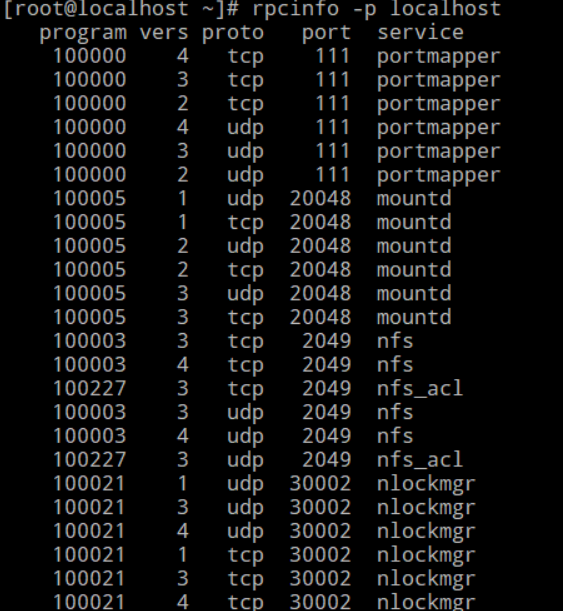
# systemctl restart rpcbind #得先重启rpcbind

# systemctl restart nfs

# systemctl restart nfs-server

# exportfs -a

# rpcinfo -p localhost #查看rpc服务注册状况



可见端口已改为指定的了

**★创建共享目录**

# mkdir /nfs-share #创建nfs共享目录

# chown -R nfsnobody:nfsnobody /nfs-share #更改目录属主

# vi /etc/exports #编辑配置文件（一行为一个共享目录，可共享多个目录）

/nfs-share 10.99.1.0/24(rw,all\_squash,sync)

保存，退出

# exportfs -r #（表示重新刷新共享）使配置生效

# exportfs -v #查看导出的共享目录信息

# exportfs -a #表示将配置文件/etc/exports中的所有定义共享发布出去

# exportfs -u #表示卸载单一目录(-au 一起使用为卸载所有/etc/exports文件中的目录)

# showmount -e localhost #查看localhost导出的共享目录

**★/etc/exports 共享参数详解**

|  |  |
| --- | --- |
| ro | 指定的共享目录为 只读 |
| rw | 共享目录可 读写 |
| no\_all\_squash | 默认，访问用户先与本机用户匹配，匹配失败后再映射为匿名用户 |
| root\_squash | 默认，将访问的root用户映射为匿名用户 |
| all\_squash | 所有访问用户都映射为匿名用户或用户组 |
| no\_root\_squash | 访问的root用户保持root帐号权限 |
| anonuid=1099 | 指定匿名访问用户的本地用户UID，默认为nfsnobody(65534) |
| anongid=1099 | 指定匿名访问用户的本地用户组GID，默认为nfsnobody(65534) |
| secure | 默认，限制客户端只能从小于1024的tcp端口连接服务器 |
| insecure | 允许客户端从大于1024的tcp端口连接服务器 |
| sync | 将数据同步写入内存缓冲区与磁盘中，效率低，但可以保证数据的一致性 |
| async | 将数据先保存在内存缓冲区中，必要时才写入磁盘 |
| wdelay | 默认，检查是否有相关的写操作，如果有则将这些写操作一起执行，这样可以提高效率 |
| no\_wdelay | 若有写操作则立即执行，应与sync配合使用 |
| subtree\_check | 默认，若输出目录是一个子目录，则nfs服务器将检查其父目录的权限 |
| no\_subtree\_check | 即使输出目录是一个子目录，nfs服务器也不检查其父目录的权限，可以提高效率 |

**★防火墙配置**

# firewall-cmd --add-service=nfs

# firewall-cmd --add-service=rpc-bind

# firewall-cmd --add-service=mountd

# firewall-cmd --runtime-to-permanent

（即允许访问以下端口）

tcp dpt:2049

tcp dpt:111

udp dpt:111

tcp dpt:20048

udp dpt:20048

**★SeLinux配置**

启用selinux时nfs默认可访问/var/lib/nfs目录，其他目录/文件得打type：var\_lib\_nfs\_t

# chcon -Rt var\_lib\_nfs\_t /nfs-share #修改目录的selinux标签

# setsebool -P nfs\_export\_all\_rw on

# setsebool -P use\_nfs\_home\_dirs on

# setsebool -P httpd\_use\_nfs on #允许httpd使用nfs目录/文件

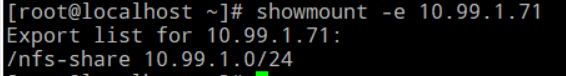
# setsebool -P samba\_share\_nfs on #允许samba共享nfs目录/文件

**②客户端**

# yum install rpcbind nfs-utils #安装

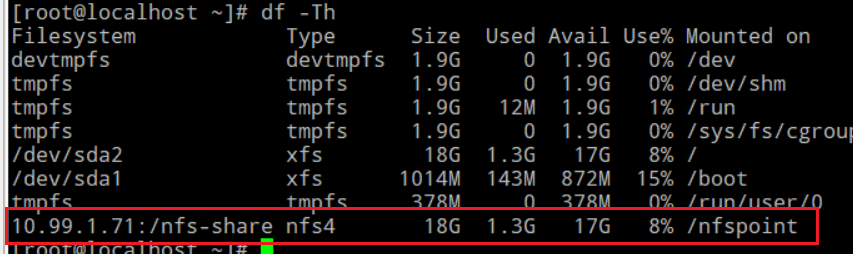
# mkdir /nfspoint #创建挂载点

# showmount -e 10.99.1.71 #先查看nfs服务端导出的共享目录情况



# mount -t nfs 10.99.1.71:/nfs-share /nfspoint -o proto=tcp -o nolock #手动挂载

# df -Th #查看磁盘挂载情况



# umount /nfspoint #卸载挂载目录

# umount -lf /nfspoint #强制卸载

**★有时 df -Th 查看时很慢**

**一般为共享目录出问题了**

# yum install strace -y

# strace df -h #查看出问题的挂载路径

**★开机自动挂载**

# vi /etc/fstab #在末尾添加一行

10.99.1.71:/nfs-share /nfspoint nfs defaults,\_netdev 0 0

保存

**★autofs配置**

/nfspoint -fstype=nfs,rw,vers=4 10.99.1.71:/nfs-share

**章二、Samba**

SMB（Server Message Block），cifs

**①服务端**

**★安装samba server**

# yum install samba samba-common #安装

# systemctl enable smb

# systemctl enable nmb

# systemctl start smb

# systemctl start nmb

**★查看默认配置**

# cat /etc/samba/smb.conf | grep -v '^#' #查看默认配置

[global]

workgroup = SAMBA #samba服务所要加入的工作组或者域

security = user #设置访问samba的验证方式，一共有四种验证方式

passdb backend = tdbsam #用户后台验证方式，支持三种

printing = cups #设置 Samba 共享打印机的类型

printcap name = cups #设置共享打印机的配置文件

load printers = yes #设置是否在启动 Samba 时就共享打印机

cups options = raw

netbios name = smbserver #NetBIOS名称，默认会使用该服务器的 DNS 名称的第一部分。netbios name 和 workgroup 名字不要设置成一样了

interfaces = lo eth0 192.168.12.2/24 192.168.13.2/24 #服务监听的网卡，也可指定ip

hosts allow = 127. 192.168.1. 192.168.10.1 #表示允许连接到 Samba Server 的客户端，多个参数以空格隔开。可以用一个 IP 表示，也可以用一个网段表示。hosts deny 与 hosts allow 刚好相反

[homes] #共享名

comment = Home Directories #描述

valid users = %S, %D%w%S #允许访问该共享的用户

browseable = No #指定该共享是否可以浏览

read only = No

inherit acls = Yes

writable = yes/no #指定该共享路径是否可写

available = yes/no #指定该共享资源是否可用

[printers] #共享名

comment = All Printers

path = /var/tmp #共享目录

printable = Yes

create mask = 0600

browseable = No

[print$]

comment = Printer Drivers

path = /var/lib/samba/drivers

write list = @printadmin root

force group = @printadmin

create mask = 0664

directory mask = 0775

#

**★创建共享目录**

# mkdir /smb-share

# chmod -R 755 /smb-share

# cat > /etc/samba/smb.conf <<EOF #正式使用时不允许行末#注释

[global]

workgroup = SAMBA #samba服务所要加入的工作组或者域

security = user #共享目录只能被授权的用户访问

passdb backend = tdbsam #用户后台验证方式，smbpasswd

printing = cups #设置 Samba 共享打印机的类型

printcap name = cups #设置共享打印机的配置文件

load printers = no #设置是否在启动 Samba 时就共享打印机

cups options = raw

netbios name = smbserver #NetBIOS名称，默认使用该服务器的DNS名称的第一部分

interfaces = lo ens33 #服务监听的网卡，也可指定ip

hosts allow = 10.99.1. #允许连接到Samba服务的客户端，多个参数以空格隔开

[smb-share] #共享名

comment = xxxx #描述

path = /smb-share #权限得另外设置,chmod 或 setfacl

valid users = cof,lee,wang #允许访问该共享的用户，逗号隔开

write list = cof,lee #可写入的用户

guest ok =no #guest账号不可访问

public = yes #

browseable = yes #该共享可以浏览

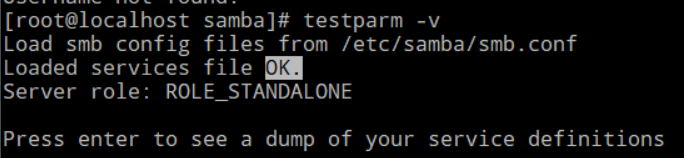
read only = no

writable = yes #该共享路径可写，同 read only = no

available = yes #该共享资源可用

EOF

# testparm -v #检查配置是否有问题

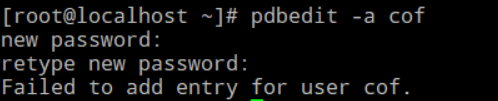


# echo "127.0.0.1 smbserver" >> /etc/hosts

**★添加smb用户**

（1）添加tdbsam用户

# pdbedit -a cof #新建 Samba账户，要求输入密码



# pdbedit -x cof #删除Samba账户

#pdbedit –L #列出 Samba 用户列表

#pdbedit –Lv #列出 Samba 用户列表的详细信息

（2）添加smbpasswd用户

# touch /var/lib/samba/private/smbpasswd

# useradd lee #先创建系统本地账号

# passwd lee

# smbpasswd -a lee #再创建 Samba账户，要求输入密码，这里的密码和系统本地账号密码不一样，客户端登录时使用这个密码验证

**★防火墙配置**

# firewall-cmd --add-service=samba

# firewall-cmd --runtime-to-permanent

（即允许访问以下端口）

udp dpt:137

udp dpt:138

tcp dpt:139

tcp dpt:445

**★SeLinux配置**

# chcon -Rt samba\_share\_t /smb-share #修改目录的selinux标签

# setsebool -P samba\_export\_all\_rw on #允许开放、共享任何文件/目录

# setsebool -P samba\_share\_nfs on #允许samba共享NFS文件系统

**②客户端**

★安装samba 组件

# yum install samba-client cifs-utils #安装

★交互式登录

# smbclient //10.99.1.71/smb-share -U lee #登录共享目录，交互式操作

Enter SAMBA\lee's password: #输入密码

smb: \> help #查看支持的命令

smb: \> exit #退出交互

**★命令挂载**

服务端密码默认hash算法： ntlm auth = ntlmv2-only

# mount -t cifs -o username="lee",password="密码" \

//10.99.1.71/smb-share /smb-point

或者：

# mount -t cifs -o username="lee",password="密码",sec=ntlmv2,port=139 \

//10.99.1.71/smb-share /smb-point

**★自动挂载**

# vi /etc/fstab

//10.99.1.71/smb-share /smb-point cifs username=lee,password=密码 0 0

保存

**章三、iSCSI**

iSCSI（internet Small Computer System Interface），又称IP-SAN

**①服务端**

★安装iscsi

# yum install targetcli

# systemctl enable target

# systemctl start target

★创建共享资源

1.创建磁盘分区 略

2.创建lvm逻辑卷 略

3.创建文件io

# mkdir /iscsi

# dd if=/dev/sda of=/iscsi/iscsi.img bs=1M count=1024 #创建1GB的file io

**★交互式配置**

# targetcli #进入交互模式

/> ls #查看配置信息

o- / .......................................... [...]

o- backstores ............................... [...]

| o- block .....一般存储资源：磁盘分区，逻辑卷... [Storage Objects: 0]

| o- fileio ......文件io...........[Storage Objects: 0]

| o- pscsi ......pscsi........... [Storage Objects: 0]

| o- ramdisk ...内存ramdisk............ [Storage Objects: 0]

o- iscsi .............................. [Targets: 0]

o- loopback .......................... [Targets: 0]

/>

/> cd /backstores/block #进入block配置层级

/backstores/block> create idisk1 /dev/sdb1 #使用磁盘分区 创建块存储共享

Created block storage object idisk1 using /dev/sdb1.

/backstores/block> cd /backstores/fileio #进入fileio配置层级

/backstores/fileio> create idisk2 /iscsi/iscsi.img #使用文件io 创建块存储共享

Created fileio idisk2 with size 1073741824

/backstores/fileio> cd /iscsi #进入iscsi配置层级

/iscsi> create iqn.2022-05.com.xxx.server1 #创建iQN

Created target iqn.2022-05.com.xxx.server1..

/iscsi> cd /iscsi/iqn.2022-05.com.xxx.server1/ #进入刚刚创建的iQN配置层级

/iscsi/iqn.20...m.xxx.server1>

/iscsi/iqn.20...m.xxx.server1> ls

o- iqn.2022-05.com.xxx.server1 ........................... [TPGs: 1]

o- tpg1 ................................... [no-gen-acls, no-auth]

o- acls ...............访问控制....................... [ACLs: 0]

o- luns .............Logical Unit Number定义输出资源.... [LUNs: 0]

o- portals ...........监听ip及端口号..................... [Portals: 1]

o- 0.0.0.0:3260 .........默认所有ip:3260........................ [OK]

/iscsi/iqn.20...m.xxx.server1>

/iscsi/iqn.20...m.xxx.server1> cd /iscsi/iqn.2022-05.com.xxx.server1/tpg1/acls

/iscsi/iqn.20...er1/tpg1/acls> create iqn.2022-05.com.xxx.server1:client1

Created Node ACL for iqn.2022-05.com.xxx.server1:client1

/iscsi/iqn.20...er1/tpg1/acls> cd iqn.2022-05.com.xxx.server1:client1

/iscsi/iqn.20...rver1:client1> set auth userid=cof

/iscsi/iqn.20...rver1:client1> set auth password=passwdxx

Parameter password is now 'passwdxx'.

/iscsi/iqn.20...rver1:client1> cd /iscsi/iqn.2022-05.com.xxx.server1/tpg1/luns

/iscsi/iqn.20...er1/tpg1/luns> create /backstores/block/idisk1

/iscsi/iqn.20...er1/tpg1/luns> create /backstores/fileio/idisk2

Created LUN 1.

/iscsi/iqn.20...er1/tpg1/luns> cd /iscsi/iqn.2022-05.com.xxx.server1/tpg1/portals/

/iscsi/iqn.20.../tpg1/portals> ls

o- portals ................................................................................................. [Portals: 1]

o- 0.0.0.0:3260 .............................................................................................. [OK]

/iscsi/iqn.20.../tpg1/portals> delete 0.0.0.0 3260

/iscsi/iqn.20.../tpg1/portals> create 10.99.1.71 3260

/iscsi/iqn.20.../tpg1/portals> ls

o- portals .................................................................................................... [Portals: 1]

o- 10.99.1.71:3260 .............................................................................................. [OK]

/iscsi/iqn.20.../tpg1/portals> cd /

/>saveconfig #保存配置

/> exit #退出交互模式，**退出时自动保存！**

Global pref auto\_save\_on\_exit=true #全局默认设置 退出时自动保存配置

Configuration saved to /etc/target/saveconfig.json

# targetcli

/> get global #查看全局默认设置

/> set global auto\_save\_on\_exit=false #设置退出时不自动保存配置

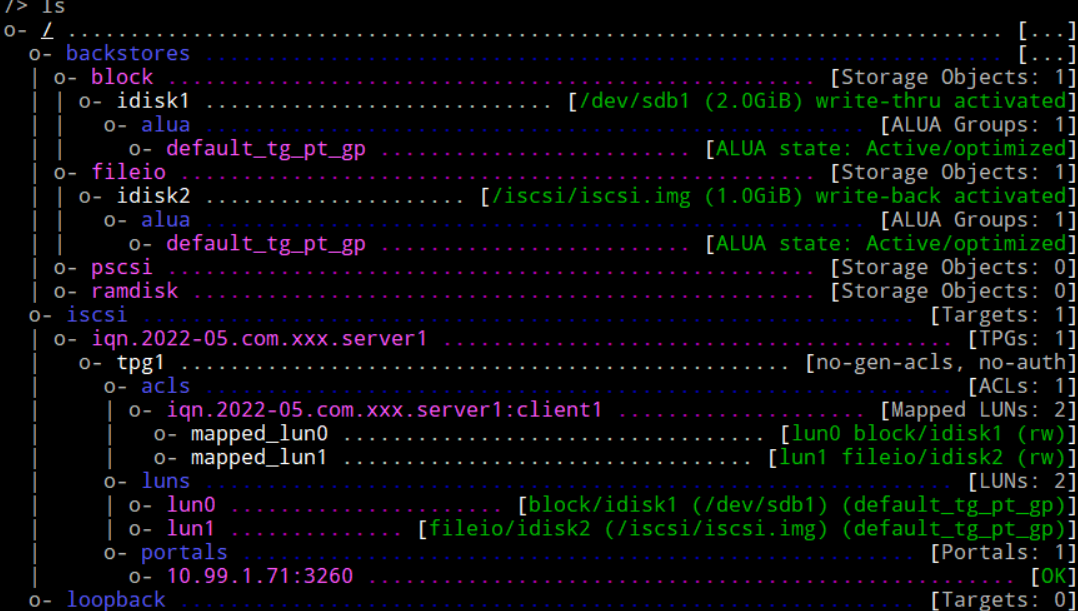
/> cd /iscsi/

/iscsi> set discovery\_auth enable=1

/iscsi> set discovery\_auth userid=cof

/iscsi> set discovery\_auth password=passwdxx

#



# cat /etc/target/saveconfig.json #查看配置文件

**★防火墙配置**

# firewall-cmd --add-port=3260/tcp

# firewall-cmd --runtime-to-permanent

**②客户端**

**★安装**

# yum install iscsi-initiator-utils

**★配置文件**

# vi /etc/iscsi/initiatorname.iscsi

删除默认那行，添加以下一行

InitiatorName=iqn.2022-05.com.xxx.server1:client1

保存

# vi /etc/iscsi/iscsid.conf #添加以下6行

node.session.auth.authmethod = CHAP

node.session.auth.username = cof

node.session.auth.password = passwdxx

discovery.sendtargets.auth.authmethod = CHAP

discovery.sendtargets.auth.username = cof

discovery.sendtargets.auth.password = passwdxx

#

# iscsiadm -m discovery -t sendtargets -p 10.99.1.71:3260 #发现服务端的iQN

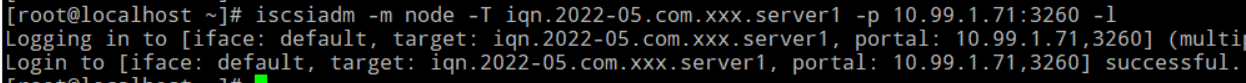
# -m（--mode） -t（--type） -p（--portal）



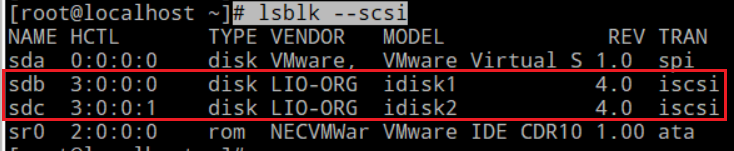
#先发现，后连接

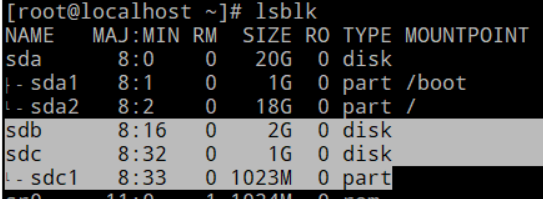
# iscsiadm -m node -T iqn.2022-05.com.xxx.server1 -p 10.99.1.71:3260 -l #连接iQN

# -T（--targetname） -l（--login）



# lsblk --scsi #查看磁盘设备信息





★上图可见，sdc为file io共享，已自动有分区sdc1且创建好了文件系统；不过其uuid同服务端的磁盘uuid，最好重新创建文件（格式化）生成新的uuid

# mkfs.xfs /dev/sdc1 -f #强制格式化

sdb为block共享，默认无分区，得先分区，再在分区上创建文件系统(ext4, xfs)； 最后挂载到某挂载点下

# sync #分区及创建文件系统后，同步一下

# iscsiadm -m session -R #当服务端新增一个luns时，客户端得刷新会话

# iscsiadm -m node -T iqn.2022-05.com.xxx.server1 -u #退出连接

# iscsiadm -m node -o delete -T iqn.2022-05.com.xxx.server1 #删除连接

**★开机自动挂载**

# vi /etc/fstab

UUID=xxxxxxxxx /iscsipoint xfs defaults,\_netdev 0 0

#建议使用uuid挂载， blkid 命令可查看

**章四、MFS**

在

**章五、GFS**

在

**章六、Ceph**

Ceph是一个开源的分布式存储系统，可提供 对象存储、块设备、文件系统的存储服务

ceph是一个对象(object)式存储系统，它把每一个待管理的数据流(文件等数据)切分为一到多个固定大小(默认4MB)的对象数据，并以其为原子单元完成数据的读写。

对象数据的底层存储服务是由多个存储主机(host)组成的存储集群，该集群也被称为RADOS存储集群

RADOS（reliable automatic distributed object store），即可靠的、自动化的、分布式的存储集群

**★ceph的构成**

|  |  |
| --- | --- |
| pool | 存储池，存储池的大小取决于底层的存储空间大小 |
| PG | (placement group)放置组，一个 pool内部可以有多个PG存在，一个 pool中有多少个PG可以通过公式计算 |
| OSD | (Object Storage Daemon)存储守护进程，ceph集群通过管理OSD来管理物理硬盘，服务器上的每一块盘都会相应地去运行一个OSD进程。 |
| librados | RADOS存储集群的API，支持大多编程语言客户端调用 |
| ceph-Mon | (Monitor)，ceph监视器,维护OSD和PG的集群状态，一个ceph集群至少要有一个mon，可以是一三五七等等这样的奇数个， |
| ceph-mgr | Ceph Manager守护程序负责跟踪运行时指标和Ceph集群的当前状态，包括存储利用率，当前性能指标和系统负载。Ceph Manager 守护程序还托管基于python 的模块来管理和公开 Ceph 集群信息，包括基于 Web的Ceph仪表板和 REST API。高可用性通常至少需要两个管理器 |
| MDS | ceph元数据服务器，代表ceph文件系统(NFS/CIFS)存储元数据，(即 Ceph块设备和 Ceph对象存储不使用 MDS) |
| Ceph管理节点 | ceph的常用管理接口是一组命令行工具程序，例如 rados、ceph、rbd 等命令，ceph管理员可以从某个特定的 ceph-mon 节点执行管理操作；推荐使用部署专用的管理节点对 ceph 进行配置管理、升级与后期维护 |

**OSD**

osd负责处理集群数据的复制、恢复、回填、再均衡，并向其他osd守护进程发送心跳，然后向Mon提供一些监控信息。当Ceph存储集群设定数据有两个副本时（一共存两份），则至少需要三个OSD守护进程即三个OSD节点，集群才能达到active+clean状态，实现冗余和高可用。

在Ceph中，如果要写数据，只能向主OSD写，然后再由主OSD向从OSD同步地写，只有当从OSD返回结果给主OSD后，主OSD才会向客户端报告写入完成的消息。如果要读数据，不会使用读写分离，而是也需要先向主OSD发请求，以保证数据的强一致性。

以OSD权重作为主/从OSD节点的选择 的唯一标准，而权重又以对应硬盘剩余存储容量作为唯一决定因素，所以意味着剩余存储容量是数据分布和读写操作中选择主次OSDd唯一决定因素

**Monitor**

用于维护集群状态映射(maintains maps of the cluster state)，比如 ceph 集群中有多少存储池、每个存储池有多少 PG 以及存储池和 PG 的映 射关系等；此外监视器还负责管理守护程序和 客户端之间的身份验证(认证使用 cephX 协议)，通常至少需要三个监视器才能实现冗余和高 可用性

**CURSH算法**

（Controllers replication under scalable hashing）可控的、可复制的、可伸缩的一致性hash算法

Ceph使用CURSH 算法来存放和管理数据，它是 Ceph的智能数据分发机制。Ceph使用CRUSH算法来准确计算数据应该被保存到哪里，以及应该从哪里读取。和保存元数据不同，是CRUSH按需计算出元数据，因此它就消除了对中心式的服务器/网关的需求，它使得Ceph客户端能够计算出元数据，该过程也称为CRUSH查找，然后和OSD直接通信。

如果是把对象直接映射到OSD之上会导致对象与OSD的对应关系过于紧密和耦合，当OSD由于故障发生变更时将会对整个ceph集群产生影响。于是 ceph将一个对象映射到 RADOS 集群的时候分为两步走： 首先使用一致性 hash算法将对象名称映射到PG，然后将PG ID基于CRUSH算法映射到OSD即可查到对象

**★Ceph相较于其他存储方案的优势**

CRUSH算法：

a. CURSH是ceph的两大创新之一（另一大就是去中心化），ceph摒弃了传统的集中式存储元数据寻址的方案，转而使用CRUSH算法计算的方式完成数据的寻址操作；

b. crush算法有强大的扩展性（即高扩展性），理论上支持上千个存储节点规模。

高可用：

a. 数据副本数可以灵活调整；

b. 可以通过crush算法指定副本的物理存放位置以分割故障域，支持数据强一致性；

c. 支持多种故障场景自动尝试进行修复；

d. 支持多份强一致性副本，副本能够垮主机、机架、机房、数据中心存放，安全可靠；

e. 存储节点可以自管理、自动修复。无单点故障，容错性强

高性能：

a. 因为是多个副本，因此在读写操作时能够做到高度并行化，理论上，节点越多，整个cpeh集群的IOPS和吞吐量就越高；

b. ceph客户端读写数据可直接与存储设备-OSD交互，在块存储和对象存储中无需元数据-MDS服务

特性丰富：

a. 支持三种存储接口：对象存储，块存储，文件存储，三种方式可一同使用

b. 支持自定义接口，支持多种语言驱动

#

**★ceph上层接口**

基于LIBRADOS层开发的三个接口，其作用是在librados库的基础上提供抽象层次更高、更便于应用或客户端使用的上层接口。

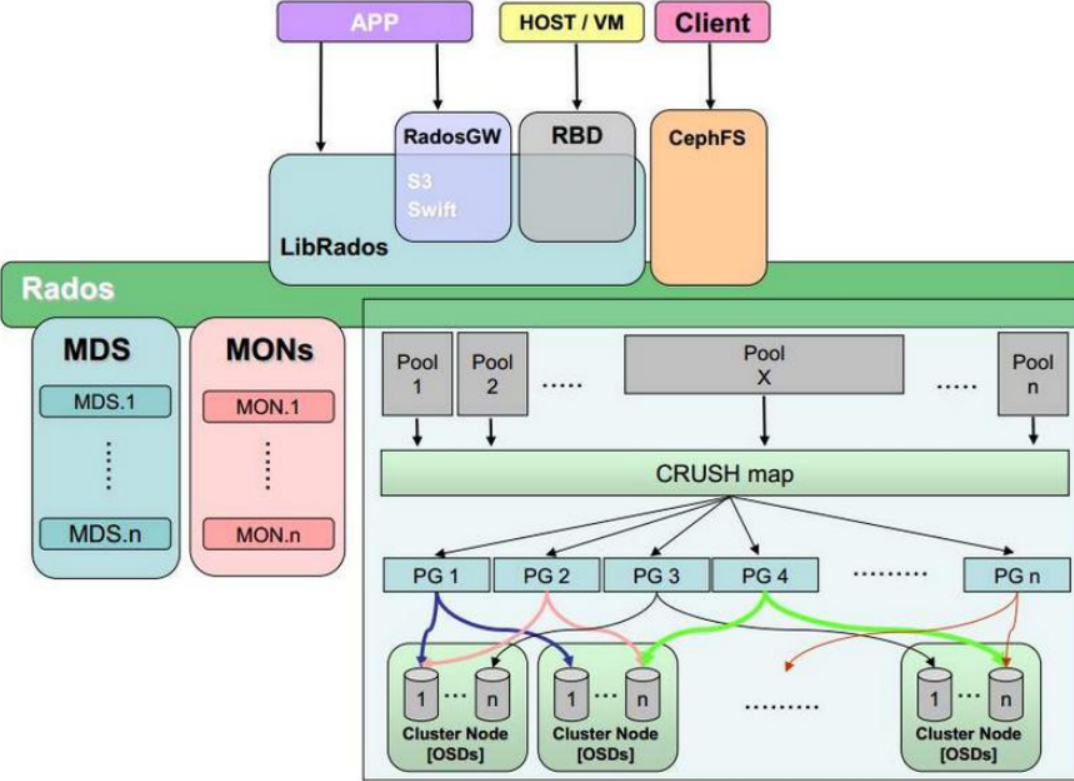
（1）RADOS GW（简称RGW）提供对象存储服务，是一套基于RESTFUL协议的网关，支持对象存储，兼容S3和Swift

（2）RBD提供分布式的块存储设备接口，主要面向虚拟机提供虚拟磁盘，可以被映射、格式化，像磁盘一样挂载到服务器使用。

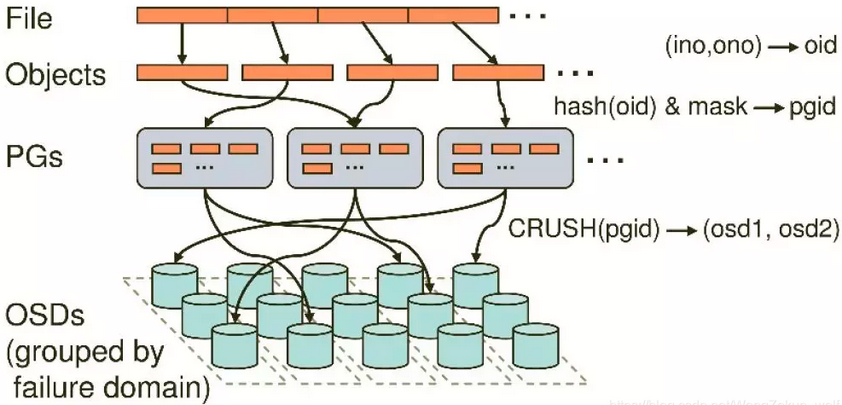
（3）CephFS是一个POSIX兼容的分布式文件系统，依赖MDS来跟踪文件层次结构，基于librados封装原生接口，它跟传统的文件系统如 Ext4 是一个类型的，但区别在于分布式存储提供了并行化的能力，像NFS等也是属于文件系统存储

**★ceph底层存储过程**

无论使用哪种存储方式（对象，块，文件），存储的数据在底层保存时，都会被切分成一个个大小固定的对象（Objects），对象大小可以由管理员自定义调整，RADOS中基本的存储单位就是Objects，一般为2MB或4MB（最后一个对象大小有可能不同）



，



文件（File）被切割成大小固定的Objects后，将这些对象分配到一个PG（Placement Group）中，然后PG会通过多副本的方式复制几份，随机分派给不同的存储节点（也可指定）

★pg数量计算公式

Total PGs = ( Total\_OSD\_number \* Target PGs per OSD ) / max\_replication\_count / pool\_count

（Target PGs per OSD = 如果osd将来不扩展则置100，将来扩展则置200）

即：总PG数 = 总osd数 \* 每osd目标pg数 / 最大文件副本数 / pool数量

例：

5个磁盘，将来不扩展，最大副本3，pool数量1 ，则总pg数为

5 x 100 /3 /1 = 166

常用值（3副本数）：

|  |  |  |
| --- | --- | --- |
| pg数量 | osd数量（将来不增加） | osd数量（将来增加） |
| 128 | 3.84 | 1.92 |
| 256 | 7.68 | 3.84 |
| 512 | 15.36 | 7.68 |
| 1024 | 30.72 | 15.36 |
| 2048 | 61.44 | 30.72 |
| 4096 | 122.88 | 61.44 |

大于50个osd，建议手动计算