一　ISCSI

１，传统的存储

DAS技术(直连存储)

将存储设备通过scsi接口或光线通道直接连接到计算机上

不能实现数据与其他主机的共享

占用服务器操作系统资源，如cpu,io等

数据量越大，性能越差

NAS技术（文件系统存储）

Ｎetwork-Attached Storage

常用应用：samba，nfs

网络存储，提供文件系统的共享

提示：磁盘（块设备）－＞分区（块设备）－＞格式化（xfs,ext4,ntfs）（文件系统）

一种专用的数据存储服务器，以数据为中心，将存储设备与服务器彻底分离，集中管理数据，从而释放带宽，提高性能，降低总成本，保护投资

用户通过tcp/ip协议访问数据－－采用标准的nfs/http/cifs等

SAN技术(块存储)

Storage Area Network

常用应用：iscsi

网络存储，块存储ethernet（以太网，双绞线连接）,fc(光线)

通过光线交换机，光线路由器，光线集线器等设备将磁盘阵列，磁带等存储设备与相关服务器连接起来，形成高速专用网络

－－组成部分

路由器，光线交换机

接口：scsi,fc

通信协议：ip,scsi

Fibre channel（光口，光线传输）

一种适合于千兆数据传输的，成熟而安全解决方案

与传统的scsi相比，fc提供更高的数据传输速率，更远的传输距离，更多的设备连接支持以及更稳定的性能，更简易的安装

Fc主要组件

光线，HBA（主机总线适配器）,FC交换机

２，ISCSI

1. 服务端

－－安装软件

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

[root@proxy ~]# yum info targetcli

－－非交互分区

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

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

－－使用targetcli定义后端存储

[root@proxy ~]# targetcli

/> ls

/> backstores/block create store /dev/vdb1　＃使用targetcli定义后端存储

备注：store为任意名称

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

/>iscsi/iqn.2018-01.cn.tedu:server1/tpg1/luns create /backstores/block/store ＃绑定存储，#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。

/> iscsi/iqn.2018-01.cn.tedu:server1/tpg1/acls create iqn.2018-01.cn.tedu:client1　　　　　　＃授权客户机访问

/> iscsi/iqn.2018-01.cn.tedu:server1/tpg1/portals/ create 0.0.0.0　＃默认本机任意ip

/> saveconfig

/> exit　　　　　　＃存储绑定服务监听的地址，并保存配置

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

[root@proxy ~]# systemctl enable target

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

[root@proxy ~]# systemctl stop firewalld

[root@proxy ~]# setenforce 0

1. 客户端

--客户端安装软件并启动服务

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

--设置本机的iqn名称

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

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

注意：必须跟服务器上配置的ACL一致！

[root@web1 ~]# systemctl restart iscsid

--发现远程target存储

提示：参考man iscsiadm！

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

[root@web1 ~]# iscsiadm --mode node --targetname iqn.2018-01.cn.tedu:server1 --portal 192.168.2.5:3260 --login　　＃连接磁盘，可不操作

--客户端挂载iSCSI共享

[root@web1 ~]# systemctl restart iscsi

[root@web1 ~]# 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]

vda 252:0 0 20G 0 disk

--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

注意：不上传，不写入数据，不能多台主机同时挂载同一个iscsi服务

　　　要是只是（仅仅）共享数据，可以多台主机同时挂载同一个iscsi服务

Discover targets at a given IP address:

iscsiadm --mode discoverydb --type sendtargets --portal 192.168.1.10 --discover

Login, must use a node record id found by the discovery:

iscsiadm --mode node --targetname iqn.2001-05.com.doe:test --portal 192.168.1.1:3260 --login

Logout:

iscsiadm --mode node --targetname iqn.2001-05.com.doe:test --portal 192.168.1.1:3260 --logout

List node records:

iscsiadm --mode node

Display all data for a given node record:

iscsiadm --mode node --targetname iqn.2001-05.com.doe:test --portal 192.168.1.1:3260

1. 部署Multipath多路径环境

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

－－发现存储服务器的共享磁盘

[root@web1 ~]# iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.5 --discover

201.1.2.5:3260,1 iqn.2018-01.cn.tedu:client1

[root@web1 ~]# service iscsi restart

停止 iscsi： [确定]

正在启动 iscsi： [确定]

[root@web1 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 20G 0 disk

├─sda1 8:1 0 20G 0 part

sdb 8:0 0 20G 0 disk

├─sdb1 8:1 0 20G 0 part

vda 252:0 0 20G 0 disk

├─vda1 252:1 0 1G 0 part /boot

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。

－－配置Multipath多路径

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

＃＃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

＃＃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

36001405e80f94c806f94df18360c4454

36001405e80f94c806f94df18360c4454

＃＃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

}

}

1. 启用Multipath多路径，并测试

＃＃启动Multipath，并设置为开机启动

[root@web1 ~]# systemctl start multipathd

[root@web1 ~]# systemctl enable multipathd

＃＃检查多路径设备文件

如果多路长设置成功，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

＃＃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 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 20G 0 disk

└─mpatha 253:2 0 20G 0 mpath

└─mpatha1 253:3 0 9.3G 0 part

sdb 8:16 0 20G 0 disk

└─mpatha 253:2 0 20G 0 mpath

└─mpatha1 253:3 0 9.3G 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]

＃＃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 mpathap1

＃＃创建目录并挂载：

[root@web1 ~]# mkfs.xfs /dev/mapper/mpathap1

[root@web1 ~]# mkdir /data

[root@web1 ~]# mount /dev/mapper/mpathap1 /data/

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

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/mpathap1 2.0G 3.0M 1.9G 1% /data

＃＃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

使用-rr选项查看，仅sda为有效运行状态。

multipath -rr　　＃重新加载多路径信息

multipath -ll #查看多路经信息

[root@web1 ~]# multipath -rr ## multipath -ll 也可以查看

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

提示：首先保证sda,sdb没有进行任何挂载，否则多路经multipath操作最后将挂载不上

３udev配置

1. 设备文件管理方法

－－devfs(静态管理)

Linux早期采用的静态管理方法

/dev目录下有大量静态文件

内核版本2.6.13开始被完全取代

－－udev(动态管理)

/sys目录下

只有连接到系统上来的设备才在/dev下创建设备文件

与主，次设备编号无关

为设备提供持久，一致的名字

2）编写udev规则

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”

－－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！

＃＃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　　＃动态看usb属性，插拔可看到

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　　＃查看设备路径

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

[root@proxy ~]# udevadm info --query=all --attribute-walk --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/sda/da1

＃＃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",ATTRS{manufacturer}=="TOSHIBA",ATTRS{serial}=="60A44CB4665EEE4133500001",RUN+="/usr/bin/wall udisk plugged in"

RUN＋＝:不破坏原来系统的动作，执行现有命令

RUN=:破坏原来系统的动作，只执行现有命令

写完立即生效

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。

＃＃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。

＃＃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"

％n:内置变量，设备编号

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

＃＃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

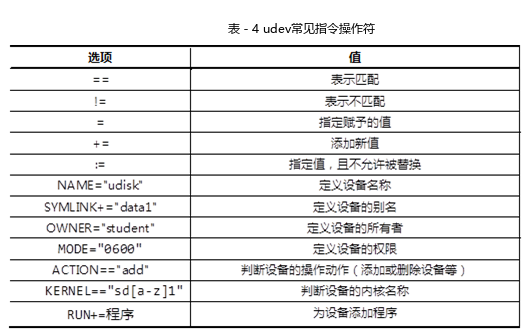
ACTION=="add",ENV{ID\_VENDOR}=="TOSHIBA",ENV{ID\_SERIAL\_SHORT}=="60A44CB4665EEE4133500001",RUN+="/usr/bin/systemctl start httpd"

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。

３）总结知识点：

udev规则文件，常见指令操作符如表-4所示。



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

%%：%符号本身

二　NFS

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