ISCSI、Multipath多路径及UDEV与NFS

CLUSTER

1. 存储的目标

— 存储是根据不同的应用环境，通过采取合理、安全、有效的方式将数据保存到某些介质上并能够保证有效的访问

— 一方面它是数据临时或长期驻留的物理媒介上

— 另一方面，它是保证数据完整安全存放的方式或行为

— 存贮就是把这两个方面结合起来，向客户提供一套数据存放的解决方案

1. 存储的技术分类
   1. SCSI小型计算机系统接口（Small Computer System Interface）

作为输入/输出接口；主要用于硬盘、光盘、磁带机等设备

* 1. DAS直连式存储（Direct -Attached Storage）

— 将存储设备通过SCSI接口或光纤通道直接连接到计算机上

— 不能实现数据与其它主机之间的共享

— 占用服务器操作系统资源，如CPU、IO等

— 数据量越大，性能越差

* 1. NAS网络技术存储（Network -Attached Storage）

1）一种专用数据存储服务器，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，降低总拥有成本，保护投资

2）用户通过TCP/IP协议访问数据

— 采用标准的NFS/HTTP/CIFS等

* 1. SAN存储区域网络（Storgae Area Network）

— 通过光纤交换机、光纤路由器、光纤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

— 如路由器、光纤交换机

— 接口：如SCSI、FC

— 通信协议：如IP、SCSI

— Fibre Channel

— 一种适合于千兆数据传输的、成熟而安全解决方案

— 于传统的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

— 光纤

— HBA（主机总线适配器）

— FC交换机

—FC交换机交换拓扑

— 点到点：point-to-point（简单将两个设备互连）

— 已裁定的环路：arbitrated loop（可多达126个设备共享一段信道或环路）

— 交换式拓扑：switched fabric（所有设备通过光纤交换机互连）

* 1. FC光纤通道
  2. ISCSI技术（Internet SCSI）

— IETF制定的标准，将SCSI数据块映射为以太网数据包

— 是一种基于IP Storage理论的新型存储技术

— 将存储行业广泛应用的SCSI接口技术与IP网络相结合

— 可以在IP网络上构建SAN

— 最初由Cisco和IBM开发

ISCSI技术的优势

— 基于IP协议技术的标准

— 允许网络在TCP/IP协议上传输SCSI命令

— 相对FC SAN ，ISCSI实现的IP SAN投资更低

— 解决了传输效率、存储容量、兼容性、开放性、安全性等方面的问题

— 没有距离限制

1）ISCSI技术客户端

— ISCSI Initiator：软件实现，成本低、性能较低

— ISCSI HBA：硬件实现，性能好，成本较高

2）ISCSI存储设备端

— ISCSI Target

3）以太网交换机

ISCSI命名规范

建议采用IQN（ISCSI限定名称）

名称必须全局唯一

IQN格式

iqn.<date\_code>.<reversed\_domain>.<string>[:<substring>]

命名示例：

iqn.2015-01.com.tarena.tech:sata.rack2.disk1

部署ISCSI服务

服务端：Target

准备新的磁盘作为后端存储用（例如/dev/vdb）

1. 安装ISCSI服务器软件

#yum -y install targetcli

1. 使用targetcli 定义后端存储

#targetcli

/>backstore/block create iscsi\_store /dev/vdb

注：iscsi\_store为任意名称

3）创建iqn对象

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

1. 绑定存储

/>iscsi/iqn.2018-01.cn.tedu:client1/tpg1/luns create /backstores/block/iscsi\_store

1. 授权客户机访问

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

/>iscsi/iqn.2018-01.cn.tedu:client1/tpg1/acls create iqn.2018-01.cn.tedu:client2

1. 存储绑定服务监听的地址，并保存配置

/> iscsi/iqn.2018-01.cn.tedu:client1/tpg1/portals create 0.0.0.0

/> saveconfig（保存配置）

/> exit

查看端口是否开启（ss -natpul | grep :3260）

查看服务的状态（systemctl is-enabled target）

1. 设置为开机自起（同时关闭防火墙）

#systemctl restart targetcli

#systemctl enable targetcli

客户端1：initiator

1. 查看软件是否安装

# rpm -q iscsi-initiator-utils

（若没有安装则进行安装）

1. 修改配置文件

# vim /etc/iscsi/initiatorname.iscsi

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

1. 发现存储：

#iscsiadm --mode discoverydb --type sendtargets --portal （服务端IP地址） --discover

1. 重起ISCSI

#systemctl restart iscsi

#systemctl enable iscsi

#lsblk （查看能否找到sda块设备）

iscsi服务用于自动登录，iscsid是守护进程

5）使用存储（根据实际情况进行分区-格式化-使用）

# mkfs.ext4 /dev/sda

# blkid （查看设备的文件系统类型）

6）安装数据库软件

# yum -y install mariadb-server

1. 自动挂载

# ll -d /var/lib/mysql

drwxr-xr-x. 2 mysql mysql 6 3月 6 2014 /var/lib/mysql

#vim /etc/fstab

追加一行：

UUID="a8e9a1fe-1455-4bc2-a49c-f2ce356e259e" /var/lib/mysql ext4 defaults,\_netdev 0 0

注：UUID可通过blkid命令进行查看，之所以使用UUID不使用/dev/sda是为了防止冲突

1. 修改/var/lib/mysql归属

# ll -d /var/lib/mysql

drwxr-xr-x. 3 root root 4096 10月 5 16:21 /var/lib/mysql/

#chown mysql.mysql /var/lib/mysql

1. 启动mysql服务

#systemctl start mariadb（第一次启动会比较慢，是在初始化数据库）

#systemctl enable mariadb

1. 创建数据

MariaDB [(none)]> create database mydb;

MariaDB [(none)]> use mydb;

MariaDB [mydb]> create table t1 (id int , name varchar(30));

MariaDB [mydb]> insert into t1 values(1,'james'),(2,'jack');

注：本地文件系统 ext2/3/4 xfs ，同时只能单台设备使用

gfs 集群文件系统（谷歌的一个文件系统）

客户端2：

先把第一台的数据库服务停止，卸载/var/lib/mysql,再进行实验

步骤与客户端1的类似

1）查看软件是否安装

# rpm -q iscsi-initiator-utils

（若没有安装则进行安装）

2）修改配置文件

# vim /etc/iscsi/initiatorname.iscsi

InitiatorName=iqn.2018-01.cn.tedu:client2

3）发现存储：

#iscsiadm --mode discoverydb --type sendtargets --portal （服务端IP地址） --discover

4）重起ISCSI

#systemctl restart iscsi

#systemctl enable iscsi

1. 挂载使用，无须格式化
2. 安装数据库
3. 启动mysql 服务
4. 查看之前的数据是否还在

# for i in `rpm -a | grep mysql`

> do

> rpm -e $i --nodeps

> done

//类似于for循环进行卸载软件，--nodeps代表强制执行，不考虑依赖关系

TREE命令，通过YUM 安装TREE得到

使用此命令，会将目标目录进行递归显示，一层一层以树状的形式显示出来

例如：# tree /var/lib/iscsi

清理工作：

1. 停止mysql服务
2. 卸载/var/lib/mysql
3. logout （登出）

# tree /var/lib/iscsi //查看iqn

# iscsiadm -m node -T iqn.2018-01.cn.tedu:client2 -p 服务器IP -u （可通过man iscsiadm查看）

UDEV

1. devfs

— Linux 早期采用的静态管理方法

— /dev目录下有大量静态文件

— 内核版本2.6.13开始被完全取代

1. udev（动态管理设备）

— 只有连接到系统上来的设备才在/dev下创建设备文件

— 与主、次设备表号无关

— 为设备提供持久、一致的名字

1. 接入设备事件链
2. 内核发现设备并导入设备状态到sysfs
3. udev接到事件通知
4. udev创建设备节点或是运行指定程序
5. udev通知hald守护进程
6. HAL（硬件抽象层）创建设备对象结构
7. HAL通过系统消息总线广播该事件
8. 用户程序也可以监控该事件

注：HAL（硬件抽象层是位于[操作系统内核](https://baike.baidu.com/item/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F%E5%86%85%E6%A0%B8" \t "/root/文档\\x/_blank)与硬件电路之间的接口层，其目的在于将硬件抽象化。它隐藏了特定平台的[硬件接口](https://baike.baidu.com/item/%E7%A1%AC%E4%BB%B6%E6%8E%A5%E5%8F%A3/444711" \t "/root/文档\x/_blank)细节,为操作系统提供虚拟硬件平台,使其具有硬件无关性,可在多种平台上进行移植。 从软硬件测试的角度来看，软硬件的测试工作都可分别基于硬件抽象层来完成，使得软硬件测试工作的并行进行成为可能）

1. udev的作用
   1. 从内核收到添加/移除硬件事件时，udev将会分析：

— /sys目录下信息

— /etc/udev/rules.d目录中的规则

b) 基于分析结果，udev会：

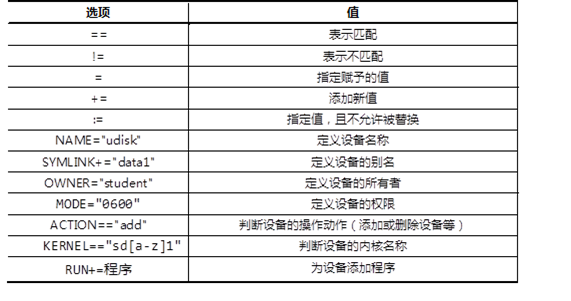
— 处理设备命名

— 决定要创建那些设备文件或链接

— 决定如何设置属性

— 决定触发哪些事件

udev常见指令操作符



1. 编写UDEV规则
2. UDEV事件监控

实时监控内核所处理的事件，比如添加/移除一块设备会提示相对应的信息

#udevadm monitor --property

1. 查看设备在内核中的属性

#udevadm info -a -p /sys/block/sda

1. 配置UDEV

主配置文件：/etc/udev/udev.conf

— udev\_root：创建设备文件位置，默认为/dev

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

— udev\_log : syslog优先级，缺省为err

文件位置及格式

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

例如：75-custom.rules

规则格式

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

查看设备的WWID （全局唯一）

# /usr/lib/udev/scsi\_id -g -u $devnode（设备名）

//-u( -- replace-whitespace ) -g( --whitelisted ) 去掉空格

//$devnode 表示磁盘

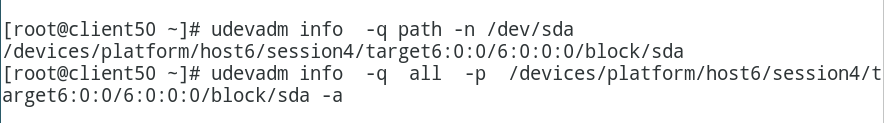
//$devpass 表示分区

例：编写UDEV规则，对设备进行改名（客户端）

#cd /etc/udev/rules.d

#udevadm info -a -p /block/sda（新开一个终端进行查看）

或者依照下图中的方式进行获取参数:



#vim 100-ipsan.rules

SUBSYSTEM=="block", //设备类型

ATTR{size}=="41943040", //设备大小

ATTRS{vendor}=="LIO-ORG ", //厂家

PROGRAM=="/usr/lib/udev/scsi\_id -g -u $devnode",

//要执行的程序

RESULT=="36001405423854f332574ae7ae493f2b0", //执行程序后的结果

SYMLINK+="ipsan" //别名

触发：使编写的链接文件生效

# start\_udev（RHEL6）

# systemctl restart systemd-udev-trigger.service （RHEL7）

# ls /dev/ipsan （会有相对应的路径）

Multipath多路径（轮询）

* 1. 多路径概述

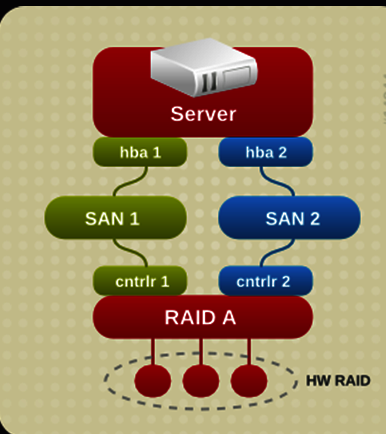
当服务器到某一存储设备有多条路径时，每条路径都会识别为一个单独的设备

多路径允许您将服务器节点和储存阵列间的多个I/O路径配置为一个单一设备

这些I/O路径时可包含独立电缆、交换器和控制器的实体SAN链接

多路径集合了I/O路径，并生成由这些集合路径组成的新设备

* 1. 多路径主要功能

— 冗余

主备模式，高可用

— 改进的性能

主主模式，负载均衡

1. 多路径设备

1）若没有DM Multipath ,从服务器节点到储存控制器的每一条路径都会被系统视为独立的设备，即使I/O路径连接的是相同的服务器节点到相同的储存控制器也是如此

DM Multipath 提供了有逻辑的管理I/O路径的方法，即在基础设备顶端生成单一多路径设备

1. 识别符

a）每个多路径设备都有一个WWID（全球识别符），它是全球唯一的， 无法更改的号码

b）默认情况下会将多路径设备的名称设定为它的WWID

c）可以在多路径配置文件中设置user\_friendly\_names选项，该选项可将别名设为格式为mpathn的节点唯一名称

d）也可以自定义存储设备名称

四、配置

环境：继续之前的实验

|  |  |
| --- | --- |
| 服务端san | 192.168.4.71  192.168.2.71 |
| 客户端：mysql1 | 192.168.4.72  192.168.2.72 |

服务端：

#yum -y install targetcli

#targetcli

/>backstore/block create iscsi\_store /dev/vdb

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

/>iscsi/iqn.2018-01.cn.tedu:client1/tpg1/luns create /backstores/block/iscsi\_store

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

/> iscsi/iqn.2018-01.cn.tedu:client1/tpg1/portals create 0.0.0.0

#systemctl restart targetcli

客户端：

从192.168.2.71与192.168.4.71这两个链路发现设备

#iscsiadm --mode discoverydb --type sendtargets --portal 192.168.2.71 --discover

#iscsiadm --mode discoverydb --type sendtargets --portal 192.168.4.71 --discover

#systemclt restart iscsi

#tree /var/lib/iscsi

1. 安装多路径软件
   1. # yum -y install multipath
   2. # yum -y install device-mapper-multipath.x86\_64
2. 产生配置文件
   1. # rpm -ql device-mapper-multipath
   2. # mpathconf --user\_friendly\_names n (直接生成)

或者

* 1. # cd /usr/share/doc/device-mapper-multipath-0.4.9/

# cp multipath.conf /etc

3、修改配置文件

* 1. # /usr/lib/udev/scsi\_id -g -u /dev/sda //获得设备的WWID

# vim /etc/multipath.conf （60，70-74行进行复制并修改）

multipaths {

multipath {

wwid 36001405423854f332574ae7ae493f2b0

alias san1

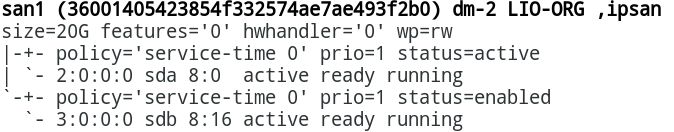
}

}

4、启动服务

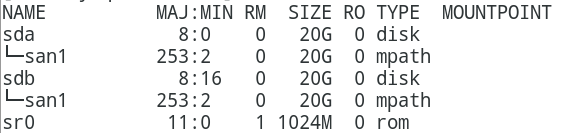
# systemctl restart multipathd.service

# systemctl status multipathd.service

1. 进行验证
   1. # multipath -ll 会有类似下面的输出 

b) # ls /dev/mapper 会多出一个设备，在这里会出现san1（跟之前设置的名字有关）

c) #lsblk （会有类似于下图的内容）



注：multipath -F （删除找到的多路径）

multipath -ll （查看多路径）

multipath -v2/3 （发现多路径）

multipath -rr （重新加载多路径信息）

1. 挂载使用

# mount /dev/mapper/san1 /var/lib/mysql

NFS

1. NFS（网络文件系统）

A：概述

a）文件系统的类型

— EXT3/4、SWAP、NTFS、....... ——本地磁盘

b）为文件系统

— /proc 、/sys、 ....... ——内存空间

c）网络文件系统

— NFS （Network File System） ——网络存储空间

d）共享协议

Unix/Linux最基本的文件共享机制

— 1980年由SUN公司开发

— 依赖于RPC（远程过程调用）映射机制

— 存取位于远程此盘中的文档数据，对应用程序是透明的，就好像访问本地的文件一样

1. 配置

环境：NAS --nfs

192.168.4.71 /temp

--- 192.168.4.0/24（rw,no\_root\_squash）

--- 192.168.2.0/24（ro）

no\_root\_squash :保留来自客户端的root权限

all\_squash : 客户端权限都将为nfsnobody

sync\async : 同步写、异步写入

* 1. 服务端
     + 1. 创建共享目录

# mkdir /temp

* + - 1. 查看软件是否安装

# rpm -qa | grep nfs

* + - 1. 修改配置文件

# vim /etc/exports



* + - 1. 启动服务/设置为开机自起
         1. systemctl restart/enable rpcbind
         2. systemctl restart/enable nfs-server

1. 查看nfs进程
   1. # ps -e | grep nfs

b) 客户端

1. 查看服务端的共享

# showmount -e 192.168.4.71（192.168.2.71）

1. 挂载使用
   1. # mount -t nfs 192.168.4.71:/temp /mnt

//此时应该具有写的权限

b) # mount -t nfs 192.168.2.71:/temp /mnt

//此时仅具有读权限，没有写权限

客户端实现按需挂载：

1. 查看软件包

# rpm -q autofs

# yum -y install autofs

1. 修改主配置文件

# vim /etc/ auto.master （定义挂载点的父目录，或指定监控点）

/misc /etc/auto.misc

# vim /etc/auto.misc

nfsdir -fstype=nfs,rw 192.168.4.71:/temp

1. 启动服务

# systemctl restart autofs

#systemctl enable autofs

1. 触发挂载：（访问挂载点/misc/nfsdir即可）

# ls /misc/nfsdir