H3C
数字化解决方案领导者

# iSCSI存储和存储多路径介绍

综合产品支持部——李树兵

01 iSCSI存储介绍

02 Linux DM multipath（多路径）介绍

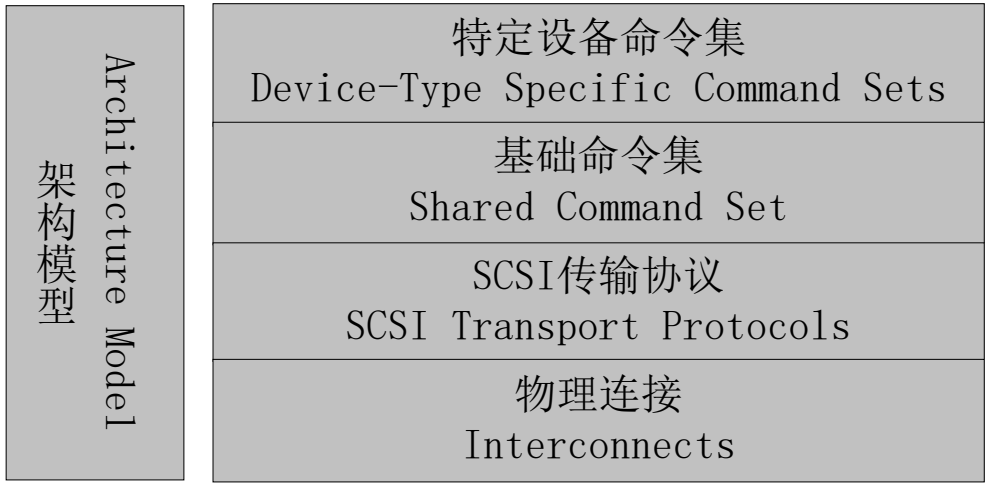
01 iSCSI存储介绍

02 Linux DM multipath（多路径）介绍

# SCSI（Small Computer System Interface）历史

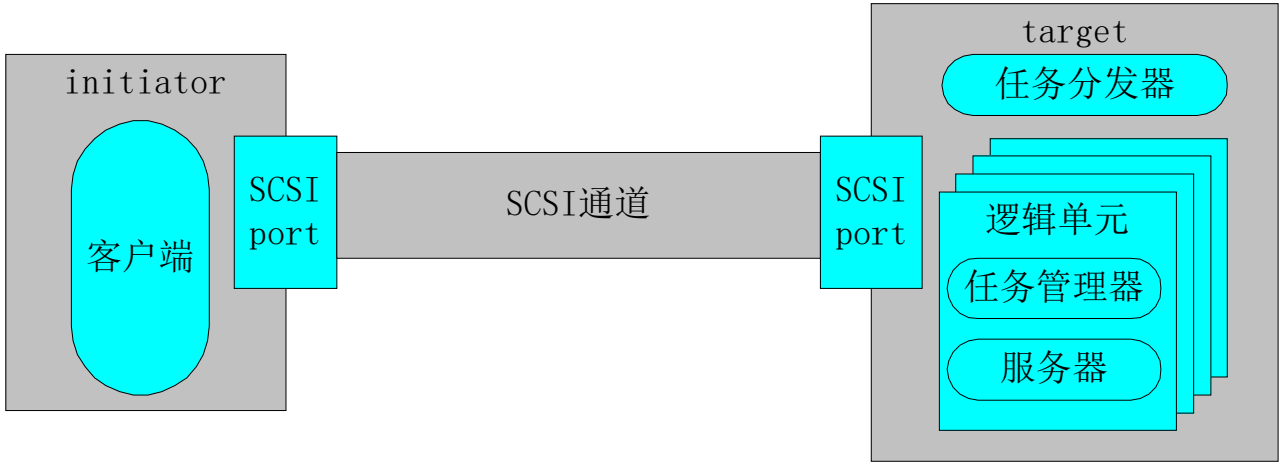
- SISI小型计算机系统接口技术是存储设备最基本的标准协议，但是他通常需要设备互相靠近并用SCSI总线连接。
- SCSI-1
  - 1986年ANSI标准，X3.131-1986
- SCSI-2
  - 1994年ANSI标准，X3.131-1994
- SCSI-3
  - 发展到SCSI-3，已经过于复杂，不适合作为单独的一个标准；在某次修订中，T10委员会决定把SCSI分为更小的单元；并且将某些单元的维护工作移交给其他组织。SCSI-3是这些子标准的集合，表示第三代SCSI。
  - iSCSI属于SCSI-3的传输协议层（类似于FCP），由IETF维护。
  - 目前还没有SCSI-4

# SCSI-3架构模型




特定设备命令集：包括磁盘设备的“SCSI块命令(SCSI Block Commands)”等
基础命令集：所有SCSI设备都必须实现的“基础命令(SCSI Primary Commands)”
SCSI传输协议：譬如iSCSI，FCP
物理连接：譬如光纤通道，internet
架构模型：定义了SCSI系统模型和各单元的功能分工

# SCSI通讯模型



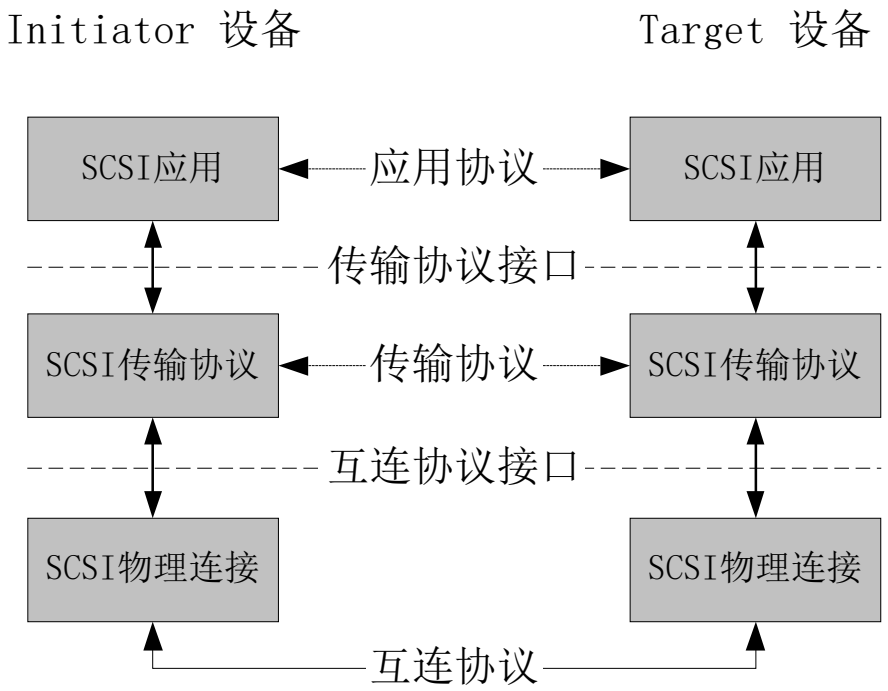

- 见下页的概念介绍
- iSCSI正是SCSI通道的一种

# SCSI主要实体介绍

- SCSI是客户端-服务器架构
  - 在SCSI术语里，客户端叫**initiator**，服务器叫**target**
  - Initiator通过**SCSI通道**向target发送请求，target通过SCSI通道向initiator发送响应
  - SCSI通道连接的是SCSI Port，SCSI initiator port和SCSI target port
- 所以initiator至少必须包括一个SCSI Port，和一个能发起请求的客户端
- 在SCSI里，target的概念比较麻烦（可能是历史原因），它并很与initiator对应
  - Target需要包括一个SCSI port，任务分发器和逻辑单元
    - SCSI port用来连接SCSI通道，任务分发器负责将客户端的请求分到到适当的逻辑单元，而逻辑单元才是真正处理任务的实体
  - 通常一个逻辑单元都支持多任务环境，所以需要包含一个任务管理器和真正响应请求的服务器
- Initiator用一种叫做CDB(Command Descriptor Blocks)的数据结构封装请求

# 分布式SCSI通讯模型

- Initiator的应用层封装好SCSI CDB后，调用SCSI传输协议接口……
- Target的应用层收到SCSI CDB后，根据CDB内容进行相应处理，封装好SCSI 响应后，调用SCSI传输协议接口……
- iSCSI正是SCSI传输协议的一种



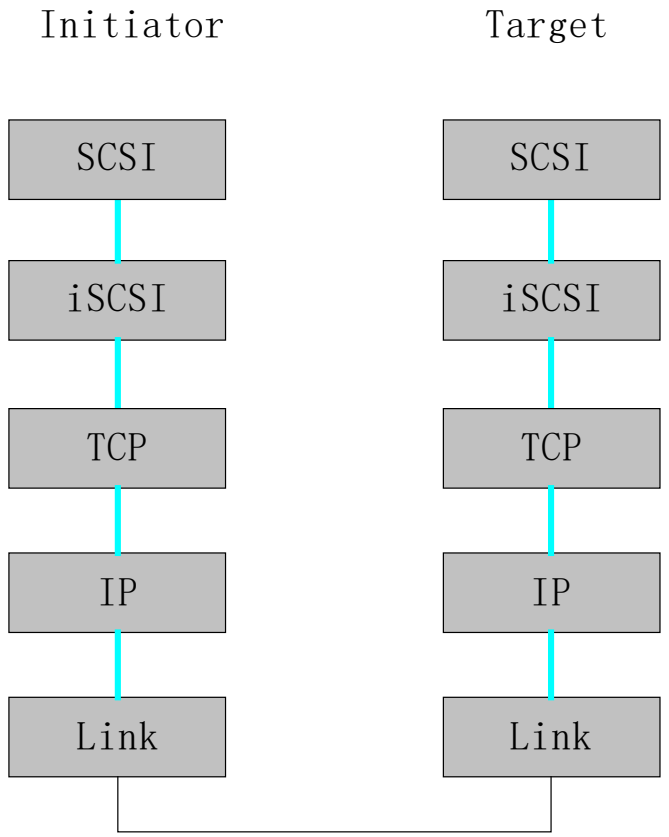

- 在initiator的应用看来，它就像调用了一个函数：
  - Service Response = Execute Command (IN ( I_T_L_Q Nexus, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [Command Reference Number], [Task Priority]), OUT ( [Data-In Buffer], [Sense Data],[Sense Data Length], Status, [Retry Delay Timer] ))

# iSCSI概述

- 由IBM，Cisco，HP发起；2000-2为IETF草案，2004-4作为正式的IETF标准
  - 现在的iSCSI协议对应SAM2，而最新的SAM是SAM4
- 主要由RFC3720描述
  - 涉及的RFC还有：3721（iSNS），3722（命令规范），3723（安全），3347（设计），3783（IP上的有序命令传送），3385（错误估计）
- 用TCP/IP作为SAN的基础设施
  - iSCSI是构建在TCP之上，而不是IP；是面向连接的，而不是无连接的。（一个iSCSI命令可能分在几个IP报文中）
  - 需要禁用TCP的NAGLE算法（见TCP/IP详解V1 19.4节）
- iSCSI的主要功能是在TCP/IP上封装，并可靠的传输SCSI命令

# iSCSI协议栈

- **Initiator**
  - SCSI层负责生成CDB，将CDB传给iSCSI
  - iSCSI层负责生成iSCSI PDU（含CDB），并通过IP网络将PDU发给target。注意，iSCSI层并不知道CDB的内容，只是简单的将整个CDB封装在PDU中。
- **Target**
  - iSCSI层收到PDU，将CDB传给SCSI层
  - SCSI层负责解释CDB的意义。必要时发送响应。

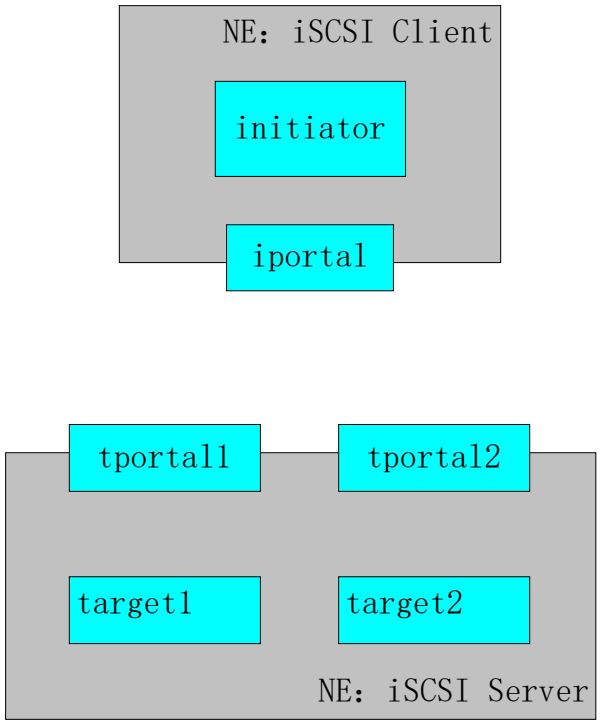


# iSCSI模块

- iSCSI主要有四个方面的内容：
  - 会话管理
  - 错误处理
  - 地址和命名规范
  - 安全和认证

# iSCSI会话概念

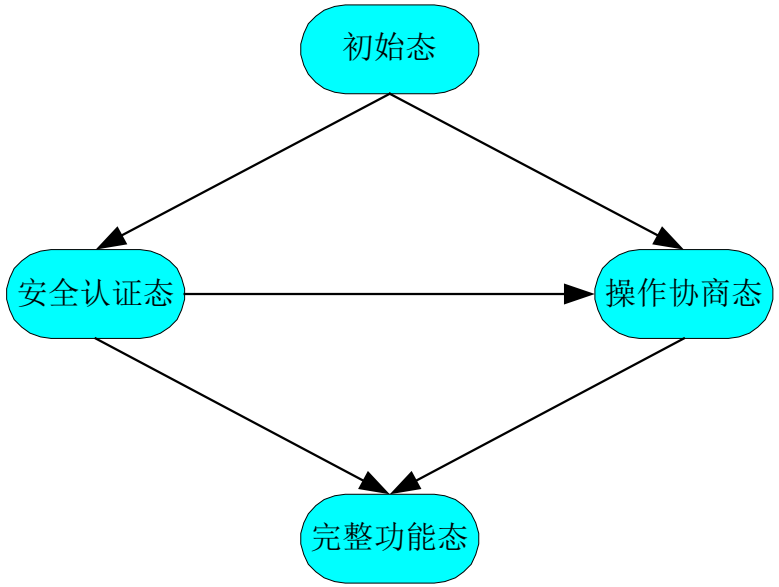
- Session对应SCSI的I_T nexus
- Session可以包含多个TCP connection
- 右图的模型里可以有许多session
  - Initiator到target1的session
    - Initiator, isid1, tportal1, target1
    - Initiator, isid1, tportal2, target1
    - …
  - Initiator到target2的session
    - Initiator, isid1, tportal1, target2
    - Initiator, isid1, tportal2, target2
    - …



# iSCSI会话类型

- 两种会话类型
  - Dicovery session
    - 用于发现target，只支持三种iSCSI报文
      - Login Req / Rsp
      - Text Req(Send Target) / Rsp
      - Logout Req / Rsp
  - **Normal session**
    - 登录阶段
      - 初始化登录阶段
      - 安全认证阶段
      - 操作协商阶段
    - 完整功能阶段
    - 退出阶段

# iSCSI登录&参数协商

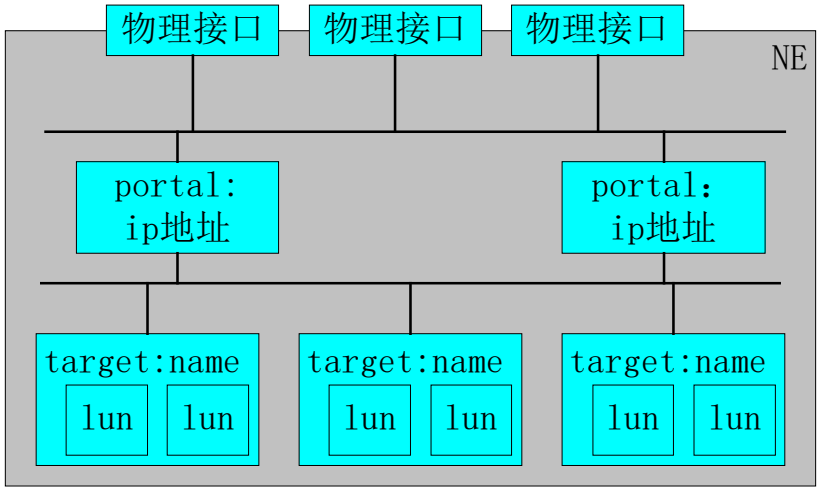



- 四个状态
  - 初始态
  - 安全认证态
  - 操作协商态
  - 完整功能态

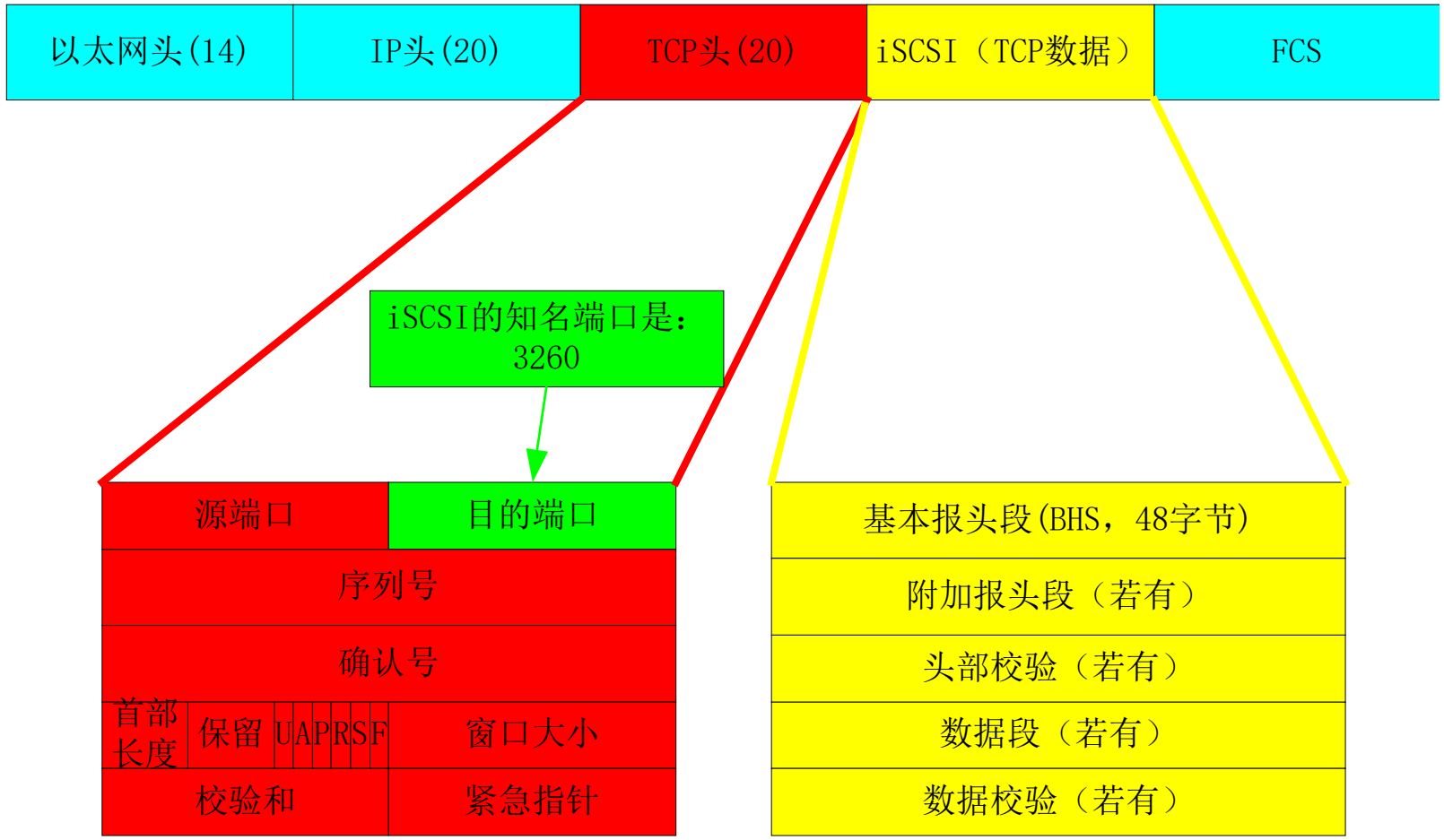
- **在initiator和target之间有两种参数协商方法**
  - 登录过程中的参数协商，通过Login Req和Login Rsp协商参数
  - 完整功能态下（已经登录）的参数协商，通过Text Req和Text Rsp进行参数协商
  - 这两种协商方法，报文的数据段语法相同
  - 有些参数只能在登录过程协商

# iSCSI命名规范

- 两种编码方式
  - iqn(iSCSI qualified name)：
    - 类型
    - 日期（拥有组织名的日期）
    - 组织名
    - 组织内部唯一的标志符
    - 实例：iqn.2001-04.com.h3c:storage.tape1.sys1.xyz
  - eui（主要用于FC设备接入iSCSI网络）
    - 类型
    - EUI-64 标识符（如果是FC设备，那么就是FC的WWN）
    - 实例：eui.02004567A425678D
- 一个iSCSI Node只能有一个名字（？），但可以有多个地址
  - 一个名字可以有多个IP地址；多个名字也可以共用一个IP地址
- iSCSI的认证机制是基于名字的



# iSCSI报文格式


以太网头(14)
IP头(20)
TCP头(20)
iSCSI（TCP数据）
FCS
iSCSI的知名端口是：
3260
源端口
目的端口
序列号
确认号
首部长度
保留
U
A
P
R
S
F
窗口大小
校验和
紧急指针
基本报头段(BHS，48字节)
附加报头段（若有）
头部校验（若有）
数据段（若有）
数据校验（若有）

# iSCSI报文分类

## SCSI Payload

- SCSI命令/响应
- 任务管理请求/响应
- SCSI 读写（ Data-In， Data-Out）
- Ready To Transfer

## SCSI and iSCSI Payload

- 异步消息

## iSCSI Payload

- Text请求/响应
- 登录请求/响应
- 注销请求/响应
- SNACK请求（selective negative ack）
- Reject
- NOP-Out/NOP-In

# Target配置

Target是iSCSI网络中提供存储服务的节点，能够为用户提供可用的存储资源。

第1步：安装target： yum install -y targetd targetcli

第2步：开启服务并开机自运行： systemctl start targetd && systemctl enable targetd

第3步：配置target存储资源。Targetcli是用于管理服务器存储资源的专用配置命令，它将iscsi配置内容抽象成“目录”的形式。命令行中使用 targetcli 进入配置界面。

3.1 配置后台存储卷信息，/backstores/block 是iscsi配置存储设备的位置。配置方法如下：

```
/> cd backstores/block
/backstores/block> create mystorage /dev/nvme0n1
Created block storage object mystorage using /dev/nvme0n1.
/backstores/block> ls
o- block ..... [Storage Objects: 1]
 o- mystorage ..... [/dev/nvme0n1 (150.0GiB) write-thru deactivated]
  o- alua ..... [ALUA Groups: 1]
   o- default_tg_pt_gp ..... [ALUA state: Active/optimized]
```

# Target配置

3.2 创建target名称及共享资源。iSCSI target名称是由系统自动生成的，这是一串用于描述共享资源的唯一字符串。稍后用户在扫描iSCSI服务端时即可看到这个字符串，因此我们不需要记住它。系统在生成这个target名称后，还会在/iscsi参数目录中创建一个与其字符串同名的新“目录”用来存放共享资源。我们需要把前面加入到iSCSI共享资源池中的硬盘设备添加到这个新目录中，这样用户在登录iSCSI服务端后，即可默认使用这硬盘设备提供的共享存储资源了。

```
/> cd iscsi
/iscsi> create
Created target iqn.2003-01.org.linux-iscsi.target.x8664:sn.3472bc21c28e.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.
/iscsi> ls
o- iscsi ..... [Targets: 1]
  o- iqn.2003-01.org.linux-iscsi.target.x8664:sn.3472bc21c28e ..... [TPGs: 1]
    o- tpg1 ..... [no-gen-acls, no-auth]
      o- acls ..... [ACLs: 0]
      o- luns ..... [LUNs: 0]
      o- portals ..... [Portals: 1]
        o- 0.0.0.0:3260 ..... [OK]
/iscsi> cd iqn.2003-01.org.linux-iscsi.target.x8664:sn.3472bc21c28e/
/iscsi/iqn.20....3472bc21c28e> cd tpg1/
/iscsi/iqn.20...bc21c28e/tpg1> cd luns
/iscsi/iqn.20...28e/tpg1/luns> create /backstores/block/mystorage
Created LUN 0.
/iscsi/iqn.20...28e/tpg1/luns> ls
o- luns ..... [LUNs: 1]
  o- lun0 ..... [block/mystorage (/dev/nvme0n1) (default_tg_pt_gp)]
/iscsi/iqn.20...28e/tpg1/luns> cd ..
```

# Target配置

3.3 设置acl，设置哪些客户端iqn可以访问此存储lun，这个值需要在客户端上查看并将此值添加到target中

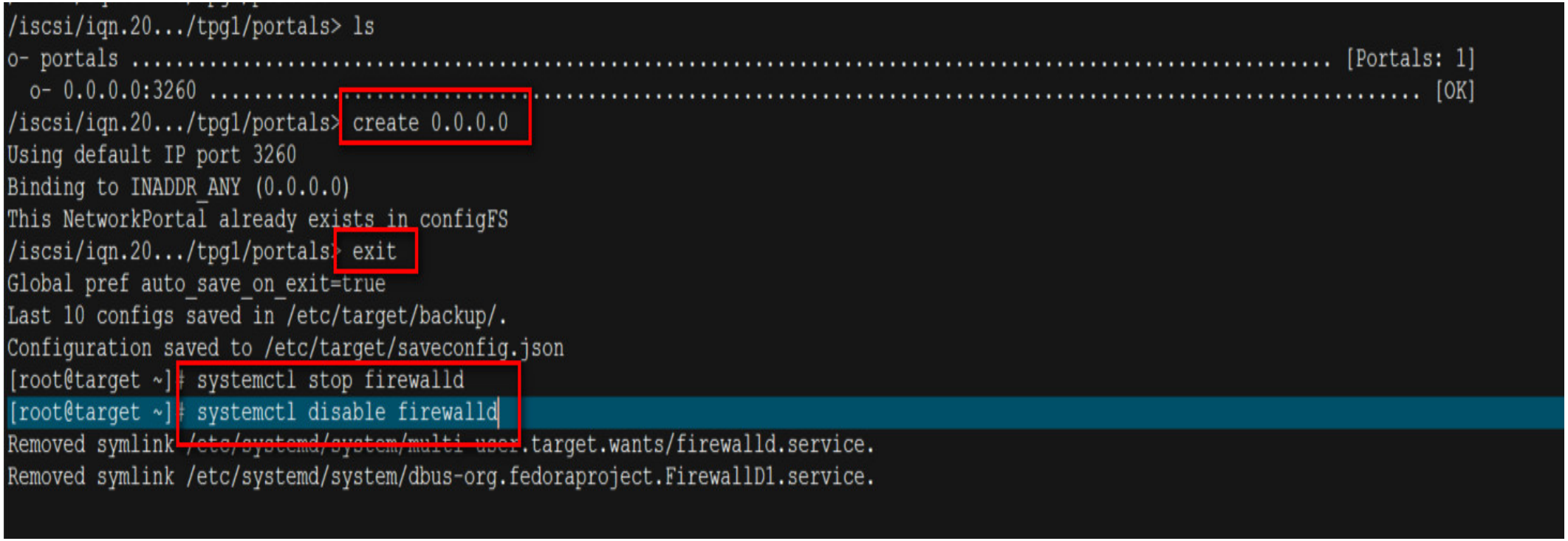
[root@iscsi-client ~]# cat /etc/iscsi/initiatorname.iscsi

InitiatorName=iqn.1994-05.com.redhat:aea894a1df85

```
/iscsi/iqn.20...bc21c28e/tpg1> cd acls
/iscsi/iqn.20...28e/tpg1/acls> create
add_mapped_luns=  wwn=
/iscsi/iqn.20...28e/tpg1/acls> create iqn.1994-05.com.redhat:aea894a1df85
Created Node ACL for iqn.1994-05.com.redhat:aea894a1df85
Created mapped LUN 0.
/iscsi/iqn.20...28e/tpg1/acls> ls
o- acls ..... [ACLs: 1]
  o- iqn.1994-05.com.redhat:aea894a1df85 ..... [Mapped LUNs: 1]
    o- mapped_lun0 ..... [lun0 block/mystorage (rw)]
/iscsi/iqn.20...28e/tpg1/acls> cd ..
```

# Target配置

3.4 设置服务器的监听IP和端口号，端口默认是3260，地址需要本地本地网卡的地址，也可以配置为0.0.0.0，即本地网卡的地址都可以提供服务。之后exit退出，重启target服务。 systemctl restart targetd.然后关闭本地防火墙（也可以配置放通tcp 3260端口 firewall-cmd –permanent –add-port=3260/tcp && firewall-cmd –reload）



# Target配置

3.5文件配置  target的配置信息保存在  /etc/target/saveconfig.json中，还有一个备份的目录文件在/etc/target/backup/目录中

[root@target backup]# ll

-rw-------. 1 root root  71 Aug  4 15:59 saveconfig-20190804-15:59:10.json

-rw-------. 1 root root 4647 Aug  4 16:06 saveconfig-20190804-16:06:06.json

我们可以手动修改这个文件，然后通过targetctl restore saveconfig.json 重新加载。

查看详细信息需要在targetcli中输入sessions detail

```
/> sessions sid=1
alias: iscsi-client     sid: 1 type: Normal session-state: LOGGED_IN
/> sessions detail
alias: iscsi-client     sid: 1 type: Normal session-state: LOGGED_IN
    name: iqn.1994-05.com.redhat:aea894a1df85 (NOT AUTHENTICATED)
    mapped-lun: 0 backstore: block/mystorage mode: rw
    address: 192.168.126.129 (TCP)  cid: 0 connection-state: LOGGED_IN
/>
```

saveconfig.json

# initiator配置

第1步：安装  yum install iscsi-initiator-utils

第2步：修改initiator配置并设置iscsid服务启动并开机自运行

vim /etc/iscsi/initiatorname.iscsi

systemctl start iscsi  && systemctl enable iscsi

第3步：  使用iscsiadm发现存储。iscsiadm是用于管理、查询、插入、更新或删除iSCSI数据库配置文件的命令行工具，需要先使用这个工具扫描发现远程iSCSI服务端，然后查看找到的服务端上有哪些可用的共享存储资源。其中，-m  discovery参数的目的是扫描并发现可用的存储资源，-t  st参数为执行扫描操作的类型，-p x.x.x.x参数为iSCSI服务端的IP地址：

[root@iscsi-client ~]# iscsiadm -m discovery -t sendtargets -p 192.168.126.128

192.168.126.128:3260,1 iqn.2003-01.org.linux-iscsi.target.x8664:sn.3472bc21c28e

# initiator配置

第4步：登录存储。-m node参数为将客户端所在主机作为一台节点服务器，-T iqn.2003-01.org.linux-iscsi.target.x8664:sn.3472bc21c28e参数为要使用的存储资源。可以直接忽略，直接使用iscsiadm –m node 登录，然后使用iscsiadm –m node –l 登录上

[root@iscsi-client ~]# iscsiadm -m node

192.168.126.128:3260,1 iqn.2003-01.org.linux-iscsi.target.x8664:sn.3472bc21c28e

[root@iscsi-client ~]# iscsiadm -m node -l

Logging in to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.x8664:sn.3472bc21c28e, portal: 192.168.126.128,3260] (multiple)

Login to [iface: default, target: iqn.2003-01.org.linux-iscsi.target.x8664:sn.3472bc21c28e, portal: 192.168.126.128,3260] successful.

# initiator配置

第5步：查看本地磁盘，之后可以分区、格式化、挂载使用。

通过fdisk –l  可以看到本地多加了 /dev/sdb盘。

通过 fdisik /dev/sdb  将磁盘进行分区

分区之后通过 mkfs.ext4 /dev/sdb1 将分区进行格式化，格式为ext4

然后通过mkdir 创建挂载点/mnt/isc，通过mount /dev/sdb1 /mnt/isc 进行挂载

修改/etc/fstab 文件，将挂载信息配置到文件中。

iscsid的配置文件在/etc/iscsi/iscsid.conf文件中，里面定义了iscsi的一系列参数。比如超时时间等。

比如默认的超时时间为120s

node.session.timeo.replacement_timeout = 120

查看详细会话信息可以使用 iscsiadm –m session –P 3



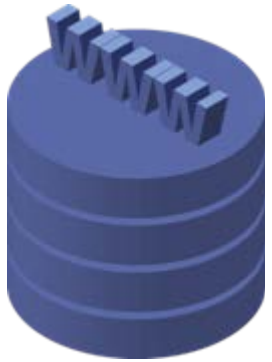
01 iSCSI存储介绍

02 Linux DM multipath（多路径）介绍

# 网络存储系统中多链路技术的应用

**常见应用服务对高可靠性的要求：**

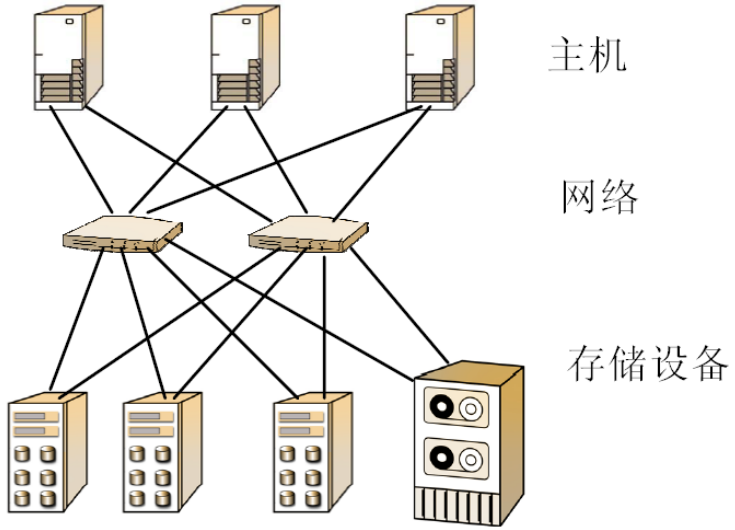
- E-Mail服务
- Web服务
- DB
- 文件服务
- 其他



# 网络存储系统中多链路技术的应用

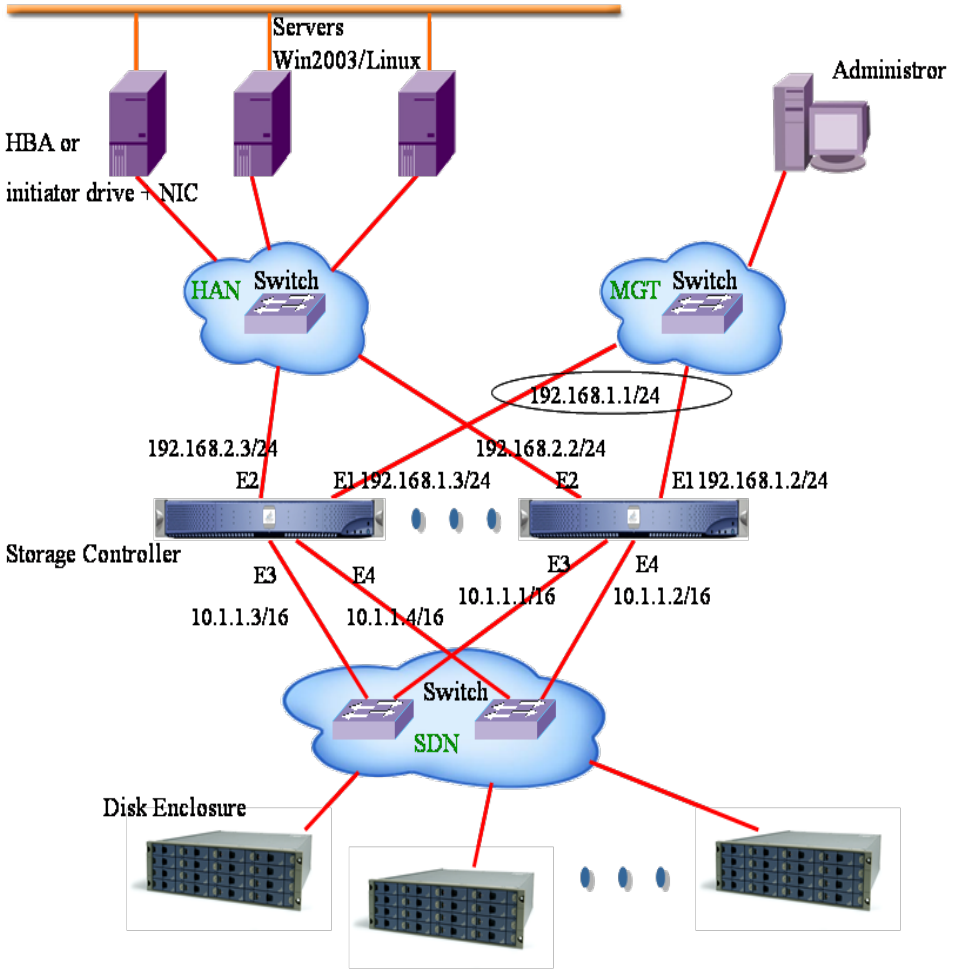
**网络存储系统对高带宽的需求:**

- 前端速率上升到千兆
- 主机到存储子系统的链路成为可能的瓶颈之一

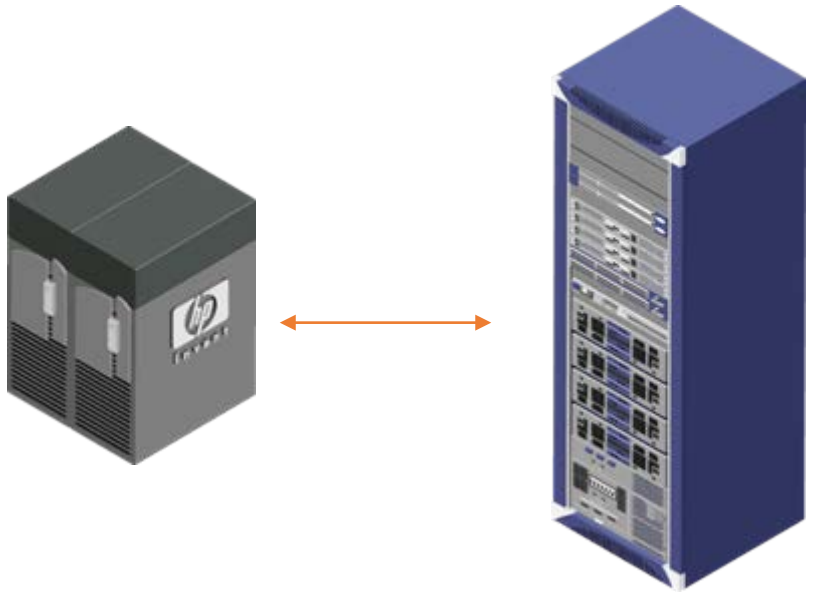


# 网络存储系统中多链路技术的应用

- 网络环境越来越复杂
- 数据流量越来越大
- 对网络流量负载均衡的要求越来越高
- 普通单网卡难以应对



# 多链路基本原理



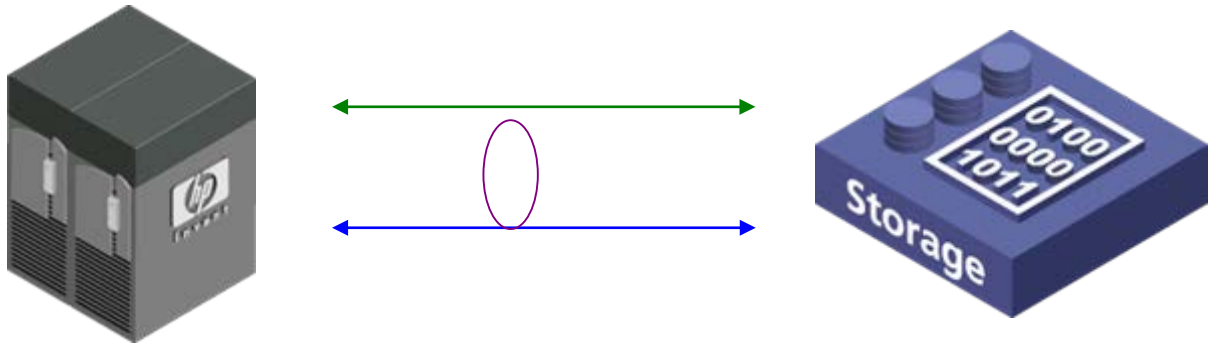
传统单链路接入方式的不足:

单链路接入方式暴露了更多的不足与缺陷，如在故障恢复，负载均衡，带宽等各方面都存在问题。

# 多链路基本原理

什么是多链路?

指一条以上的网络链路聚合在一起以达到更大的吞吐能力以及冗余与负载均衡的目的。



# 多链路的优势

- **提高可靠性，故障的切换的恢复**
  - 避免在单链路环境下，当一条链路出现故障，系统即无法工作的问题
- **负载均衡能力**
  - 当某条链路上的流量很大甚至拥塞时，自动将流量分散到其他空闲链路上，提高系统整体性能
- **增加带宽**
  - 通过使用多链路技术，可以部分解决系统的带宽瓶颈，增加带宽
- **磁盘的虚拟化**
  - 操作系统中一条路径认为是一块磁盘，使用多路径技术可以将多条路径认为是一个磁盘，统一管理。

# 多路径管理软件

H3C
数字化解决方案领导者

- **多路径带来的问题**
- 存储网络越来越发达了，很多时候一个LUN有多条通路可以访问
- 服务器使用多个HBA连接到存储网络，存储网络又可能是由多个交换设备组成，而存储系统又可能有多个控制器和链路，LUN到服务器的存储网络链路又可能存在着多条不同的逻辑链路。
- 同一个physical LUN在服务器上必然被识别为多个设备。因为os区别设备用的是总线，target id，LUN id，只要号码不同，就认为是不同的设备。
- 这样就使得存储管理越来越复杂，出现差错的几率也越来越大。

# 多路径管理软件



- **多路径管理软件应运而生**：

能使操作系统正确的识别到physical LUN，具体的做法，就是生成一个特别的设备文件，操作系统操作这个特殊的设备文件来实现对LUN的控制。设备文件+driver+firmware的一个作用，就是告诉操作系统该怎么使用这个设备。也就是说，多路径管理软件从driver和设备文件着手，告诉了操作系统怎么来处理这些身份复杂的LUN。

# 多链路的类型

- 活动-活动(Active-Active)模式

  多链路的两块网卡始终都处于工作状态,没有主从的区别

- 活动-备用(Active-Passive)模式

  其中一块网卡处于工作状态,另一块处于后备状态,只有当工作网卡出现故障时,备用网卡才接替原网卡工作

# FC SAN/IP SAN多路径软件

- 现在不论是FC SAN还是IP SAN，都会使用多路径软件来实现多路径，下面是一些常见厂家的多路径软件。

| 存储厂商/操作系统平台 | Win | Linux | Solaris | HP-UX | AIX |
|---|---|---|---|---|---|
| **EMC** | PowerPath | | | | |
| **HDS** | HDLM | | | | |
| **IBM** | RDAC(DS5000)/SDD(DS8000) | | | | |
| **HP** | Secure Pathing | | | | |
| **NetApp FC SAN** | MPIO | Qlogic SAN Surfer | Veritas VxVM DMP | PVLink | DotHill SANPath |

- 各个存储厂家提供了适合自己存储的多路径软件，但这些软件通常都只支持自己品牌型号的存储设备。在兼容性方面存在很大的问题。而且软件和硬件也不是一起卖，如果要使用多路径软件，还需要购买license。Linux系统使用开源免费的multipathd，可以支持大多数的存储设备。

# Linux系统multipathd工具使用



•多路径的全称为设备映射多路径（Device Mapper（DM）Multipath）是Linux系统中默认开源的多路径管理工具。

•DM Multipath可以提供两个功能：冗余（主备方式）、增加存储带宽（主主模式，将IO以轮询机制分布到所有的路径中，在有些配置中，DM Multipath能够检测到IO路径的负载，并重新动态平衡负载。）

•默认情况下，DM Multipath支持大多数的存储阵列，如果要了解支持的默认值和支持设备的信息可以使用如下命令进行查看：

•#multipathd show config

•#multipath –t

•如果阵列支持但是默认情况下没有配置，可以将他加入到DM Multipath的配置文件(/etc/multipath.conf)中。

# DM multipath组件

- DM Multipath中有如下组件：

| 组件 | 描述 |
| --- | --- |
| dm-multipath kernel模块 | 为路径和路径组重新指定I/O并支持故障转移。 |
| mpathconf工具程序 | 配置并启用DM Multipath。 |
| multipath指令 | 列出并配置多路径设备。通常使用/etc/rc.sysinit启动，还可以在添加块设备时通过udev程序启动。 |
| multipathd守护进程 | 监视路径；若路径故障并返回，它可能会启动路径组切换。允许对多路径设备进行交互式修改。若需对/etc/multipath.conf文件进行任何修改，都必须重新启动本守护进程。 |
| kpartx命令 | 为设备分区生成设备映射器。对于带DM Multipath的基于DOS的分区来说，使用此命令很有必要。kpartx命令包含在它自己的软件包当中，但是device-mapper-multipath软件包需要依赖它。 |

# 多路径设备



•若没有DM Multipath，从服务器节点到存储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存储控制器也是如此。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。

•多路径设备识别符：WWID

•每个多路径设备都有一个全球识别符（WWID），它是一个全球唯一的无法更改的号码。默认情况下会将多路径设备的名称设定为它的WWID。另外，还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。

•例如：有两个HBA的节点通过单一不分区FC交换机附加到有两个端口的存储控制器中时，可看到四个设备：/dev/sda、/dev/sdb、dev/sdc以及/dev/sdd。DM Multipath会生成由唯一WWID的单一设备，该设备可根据多路径配置将I/O重新路由到那四个基础设备。user_friendly_names配置选项的值被设为yes时，多路径设备的名称会被设定为mpathn。

•新设备被纳入DM Multipath控制时，该设备会显示在/dev目录的两个不同位置：dev/mapper/mpathn和/dev/dm-n。

/dev/mapper中的设备是在引导过程中生成的。可使用这些设备访问多路径设备，例如在生成逻辑卷时。

# 多路径设备



•当将user_friendly_names配置选项设为yes时，该多路径设备的名称对于节点来说是唯一的，但不保证对使用多路径设备的所有节点都一致。同样，如果您为multipath.conf配置文件的multipaths部分中的设备设定alias选项，该名称不会自动在集群的所有节点中保持一致。如果您使用LVM在多路径设备中创建逻辑设备，这不应是问题。但如果您需要将您的多路径设备名称在集群中的每个节点上都保持一致，请不要将user_friendly_names选项设定为yes，且不要为那些设备配置别名。默认情况下，如果您不将user_friendly_names设定为yes，或者为某个设备配置别名，则设备名称将是该设备的WWID，它是不会变的。

# DM Multipath使用



•第一步：安装：配置好yum源之后使用  yum install -y device-mapper-multipath 进行安装

•第二步：使用modprobe将dm-multipath和dm-round-robin加入内核模块

```
# modprobe dm-multipath

# modprobe dm-round-robin
```

第三步：使用#  mpathconf  --enable生成multipath.conf文件，默认此文件不会生成，只有敲完此命令之后才会生成。

# DM Multipath使用

H3C
数字化解决方案领导者

[root@iscsi-client ~]# mpathconf -h

usage: /usr/sbin/mpathconf <command>

Commands:

Enable: --enable　开启

Disable: --disable　关闭

Only allow certain wwids (instead of enable): --allow <WWID>　允许指定的WWID

Set user_friendly_names (Default y): --user_friendly_names <y|n> 设置友好显示

Set find_multipaths (Default y): --find_multipaths <y|n>　寻找多路径

Load the dm-multipath modules on enable (Default y): --with_module <y|n> 加载dm-multipath模块

start/stop/reload multipathd (Default n): --with_multipathd <y|n>　加载多路径配置

select output file (Default /etc/multipath.conf): --outfile <FILE> 设置多路径文件输出路径

# DM Multipath使用



•第四步：开启多路径服务及设置开机自启动

• # systemctl start multipathd.service && systemctl enable multipathd

•第五步：查看多路径信息： ll /dev/mapper/ 这里可以看到多路径的信息。

•第六步（高级设置）：获取WWID信息：

• /lib/udev/scsi_id --whitelisted --device=/dev/sdb或者multipath –v3

•第七步：根据要求设置multipath.conf文件，比如根据WWID设置负载分担方式，默认是failover（主备模式），可以将其配置为multibus（负载分担）模式，实现多链路同时传输数据。

• 第八步：通过multipath –ll 查看多路径信息，正常有显示多个路径，状态显示active的话说明多链路正常。

# Multipath.conf文件配置

多路径配置文件包含以下几个部分：

blacklist　　不被视为多路径的具体设备列表。

blacklist_exceptions　　根据blacklist部分中的参数列出不在黑名单中的多路径设备。

defaults　　DM Multipath的常规默认设置。

multipaths

各个独立多路径设备的特性设置。这些数值覆盖了在配置文件的defaults和devices部分中指定的数值。

devices

各个存储控制器的设置。这些数值覆盖了在配置文件的defaults部分指定的数值。如果要使用不是默认支持的存储阵列，则可能需要为阵列创建devices子部分。

系统决定多路径设备的属性时，会先检查多路径设置，然后检查设备设置，最后才检查多路径系统默认设置。

# Multipath.conf文件配置

H3C
数字化解决方案领导者

blacklist设置：

多路径配置文件的blacklist部分指定在系统配置多路径设备时不能使用的设备。黑名单中的设备将无法分组到多路径设备中。

在旧版本中，multipath总是尝试为每个没有明确列入黑名单的路径创建多路径设备。但在新版本中，如果find_multipaths配置参数被设定为yes，multipath将只在满足以下三个条件之一时创建设备：

① 至少有两个使用同一WWID的路径没有被列入黑名单。

② 用户可使用multipath命令手动强制创建该设备。

③ 有与之前创建的多路径设备相同WWID的路径（即使那个多路径设备目前不存在）。无论何时，创建多路径设备后，多路径会记住该设备的WWID，以便在它看到有使用那个WWID的路径时即自动再次创建该设备。这可允许让多路径自动选择正确的路径以便创建多路径设备而无需编辑多路径黑名单。

# Multipath.conf文件配置

可以使用如下方法设置黑名单：

①根据WWID将设备加入黑名单

```
blacklist{
wwid26353900f02796769
}
```

②根据设备名称将设备加入黑名单

```
blacklist{
devnode"^sd[a-z]"
}
```

③根据设备类型将设备加入黑名单

# Multipath.conf文件配置

以在配置文件的blacklist部分与device一同指定具体设备类型。以下实例将所有IBMDS4200和HP设备放入黑名单。

设备的信息可以通过multipath –v3 来查看。

```
blacklist {
        device {
                vendor "IBM  “
                product "3S42  “
        }
        device {
                vendor  “HP  “
                product  “*  “
        }
}
```

# Multipath.conf文件配置

H3C
数字化解决方案领导者

blacklist_exceptions设置：此部分为被默认加入黑名单的设备启用多路径

例如：有大量设备，但只有一个需要多路径（WWID为3600d0230000000000e13955cc3757803），不需要将想要使用多路径的设备之外的每个设备单独加入黑名单，您只需要将所有设备都加入黑名单，然后在/etc/multipath.conf文件中添加以下行以便只允许想要使用多路径的设备：

```
blacklist {
        wwid"*"
       }
blacklist_exceptions {
        wwid"3600d0230000000000e13955cc3757803"
}
```

当在配置文件的blacklist_exceptions指定设备时，您必须以指定黑名单的相同方法来指定例外情况。例如：在devnode黑名单条目中指定的设备无法使用WWID将其指定为例外情况，即使列入黑名单的设备和该WWID关联也不行。同样，devnode例外也只适用于devnode条目，而device例外只适用于device条目。

# Multipath.conf文件配置

defaults设置：DM Multipath的常规默认设置。

要覆盖任意配置参数的默认值，您可将这个模板中相关的行复制到defaults部分并取消其注释。例如：要覆盖path_grouping_policy参数以便用multibus覆盖默认的failover，请将模板中正确

的行复制到配置文件的defaults部分并取消对它的注释，如下。

```
defaults {
        user_friendly_names     yes
        path_grouping_policy multibus
}
```

多路径默认参数设置：

defaults参数

# Multipath.conf文件配置

multipaths设置：各个独立多路径设备的特性设置。这些数值覆盖了在配置文件的defaults和devices部分中指定的数值。

显示在multipath.conf　配置文件multipaths部分中可为每个特定多路径设备设置的属性这些属性只适用于一个指定的multipath这些默认属性可供 DM Multipath 使用，并且能覆盖 multipath.conf 文件中defaults和devices部分设置的属性。

多路径属性参数设置：

multipath参数

# 多路径一些常见命令

H3C
数字化解决方案领导者

表 5.1. 有用的 multipath 命令选项

| 选项 | 描述 |
| --- | --- |
| **-l** | 显示来自 **sysfs** 和设备映射器的当前多路径配置。 |
| **-ll** | 显示来自 **sysfs**、设备映射器以及系统中其他所有可用组件的当前多路径配置。 |
| **-f *device*** | 删除命名的多路径设备。 |
| **-F** | 删除所有不使用的多路经设备。 |
| **-w *device*** | 从 **wwids** 文件中删除指定设备的 **wwid**。 |
| **-W** | 重新设定 **wwids** 文件使其只包含当前 multipath 设备。 |

# CAS 与3PAR存储对接双活中多路径配置

每台CVK主机的多路径配置文件中增加此段

```
devices {
   device {
      rr_min_io "100"
      product "VV"
      vendor "3PARdata"
      features "0"
      prio "alua"
      path_checker "tur"
      path_grouping_policy "group_by_prio"
      path_selector "round-robin 0"
      hardware_handler "1 alua"
      checker "tur"
      no_path_retry "25"
      dev_loss_tmo "infinity"
      failback "immediate"
      fast_io_fail_tmo "10"
   }
}
```

配置完成后，重启一下多路径服务：

service multipath-tools restart

# Thanks!

新华三集团
www.h3c.com