**iSCSI详解**

编 写 人：孙自翔

统 稿 人：

编写时间：2013-06-07

部 门 名：

审 核 人：

审核时间：

修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 修订内容简述 | 修订日期 | 版本 | 修订人 |
| 1 | 初稿 | 2012-12-23 | 0.1 | 孙自翔 |
| 2 | 增加open-iscsi的使用方式  扩充一些其他方面的内容 | 2013-6-07 | 0.2 | 孙自翔 |
|  |  |  |  |  |
|  |  |  |  |  |

正文目录

[1 文档说明 5](#_Toc358396264)

[1.1 参考资料 5](#_Toc358396265)

[1.2 本文档要达成的目标 5](#_Toc358396266)

[1.3 本文档不完善的地方 5](#_Toc358396267)

[2 背景知识 5](#_Toc358396268)

[2.1 SCSI介绍 5](#_Toc358396269)

[2.2 iSCSI概述 6](#_Toc358396270)

[3 iSCSI协议概览 7](#_Toc358396271)

[3.1 iSCSI术语介绍 7](#_Toc358396272)

[Network Entity（网络实体） 7](#_Toc358396273)

[Network Portal（网络入口） 7](#_Toc358396274)

[iSCSI Node（iSCSI节点） 7](#_Toc358396275)

[Session 7](#_Toc358396276)

[TSIH 7](#_Toc358396277)

[iSCSI Initiator Node 7](#_Toc358396278)

[iSCSI Target Node 7](#_Toc358396279)

[SSID (Session ID): 8](#_Toc358396280)

[targetIQN(TargetName) 8](#_Toc358396281)

[InitiatorName（InitiatorIQN） 8](#_Toc358396282)

[Portal Groups 8](#_Toc358396283)

[Target Portal Group Tag 8](#_Toc358396284)

[iSCSI Endpoints 8](#_Toc358396285)

[LUN 8](#_Toc358396286)

[LUN tag 8](#_Toc358396287)

[Backstore 9](#_Toc358396288)

[Storage Object 9](#_Toc358396289)

[ACL 9](#_Toc358396290)

[3.2 iSCSI术语的图例 9](#_Toc358396291)

[RFC3720中建议的模型 9](#_Toc358396292)

[Entity,Portal，Session等之间的关系 10](#_Toc358396293)

[3.3 SCSI 设备在 Linux 系统上的地址映射 10](#_Toc358396294)

[一、 SCSI设备地址映射的组成 10](#_Toc358396295)

[二、 映射地址各部分的含义 10](#_Toc358396296)

[三、 如何查看映射地址 11](#_Toc358396297)

[四、 LUN的进一步解释 11](#_Toc358396298)

[3.4 iSCSI安装 11](#_Toc358396299)

[3.4.1 安装及卸载 11](#_Toc358396300)

[3.4.2 启动流程 12](#_Toc358396301)

[3.5 open-iscsi协议实现的体系结构 13](#_Toc358396302)

[4 iSCSI target (lio) 14](#_Toc358396303)

[4.1 创建Storage Object 14](#_Toc358396304)

[4.1.1 创建Block IO形式的Storage Object 15](#_Toc358396305)

[4.2 创建iSCSI Endpoint(iSCSI端点) 15](#_Toc358396306)

[4.2.1 创建iSCSI端点(target) 15](#_Toc358396307)

[4.2.2 创建iSCSI网络入口 16](#_Toc358396308)

[4.2.3 CHAP验证 17](#_Toc358396309)

[4.2.4 输出一个target步骤 17](#_Toc358396310)

[5 iSCSI initiator（open-iscsi） 19](#_Toc358396311)

[5.1 open-iscsi的使用方式. 19](#_Toc358396312)

[5.1.1 持久化数据库 19](#_Toc358396313)

[5.2 初步使用 19](#_Toc358396314)

[5.2.1 发现并登陆target 19](#_Toc358396315)

[5.2.1 使用网络设备 19](#_Toc358396316)

[5.3 open-iscsi常用命令 20](#_Toc358396317)

[5.4 etc/iscsi/iscsid.conf详解 23](#_Toc358396318)

[5.4.1 etc/iscsi/iscsid.conf初步认识 23](#_Toc358396319)

[5.4.2 etc/iscsi/iscsid.conf各配置项含义 24](#_Toc358396320)

[6 参考文献 25](#_Toc358396321)

# 文档说明

## 参考资料

* 唐成的iSCSI协议分享：iSCSI协议介绍
* NBS项目中VolumeServer和HostAgent代码
* RFC3207
* open-iscsi文档介绍
* open-iscsi-2.0-873源码

## 本文档要达成的目标

* 进一步介绍open-iscsi协议的相关概念
* 进一步介绍open-iscsi 之initiator端的使用方式.。
* 相关概念的含义及层次结构:node、session、tpgt、lun等
* 浅析open-iscsi各个模块的结构，每个模块的功能.
* 介绍一些比较容易理解的配置参数。

## 本文档不完善的地方

* 本文档涉及的ISCSI协议本身(协议格式)的内容较少，在唐成的另一篇文档` iSCSI协议介绍`有所介绍。
* 涉及的open-iscsi的配置参数不全面。
* iSCSI target使用的是lio-utils，有关另外一种target iet方面的资料，请参考唐成的另一篇文章`iet使用详解`
* 其他

# 背景知识

## SCSI介绍

SCSI－[小型计算机系统接口](http://baike.baidu.com/view/611524.htm)，是一种专门为小型[计算机系统](http://baike.baidu.com/view/1130583.htm)设计的[存储单元](http://baike.baidu.com/view/1223079.htm)接口模式。SCSI定义了一系列规则提供给I/O设备，用以请求相互之间的服务用。每个I/O设备称为“逻辑单元”（LU），每个逻辑单元都有一个唯一的地址来区分它们，这个地址称为“逻辑单元号”（[LUN](#_LUN)）。

SCSI模型采用客户端/服务器（C/S，Client/Server）模式，客户端称为initiator，服务器称为target，数据传输时，initiator向target发送request，target回应response，在iSCSI协议中也沿用了这套思路。



图2-1 SCSI设备服务和任务请求交互模型

## iSCSI概述

ISCSI:INTERNET SCSI或TCP/IP OVER SCSI，即把SCSI协议与TCP/IP协议相结合，在IP网络上传输SCSI数据.也就是可以让SCSI命令通过网络传送到远程SCSI设备上，而SCSI协议只能访问本地的SCSI设备。

iSCSI使用客户/服务器模型。客户端称为initiator，服务器端称为target。

* Initiator：通常指用户主机系统，用户发送SCSI请求，并将SCSI命令和数据封装到TCP/IP包中发送到IP网络中。
* Target：通常位于存储设备上，用于解析TCP/IP包中的SCSI命令和数据。

在initiator端的TCP端口号随机选取，target的端口号默认是3260

其协议栈如下：



图2-2 iSCSI协议栈

# iSCSI协议概览

**RFC3720**详述了iSCSI协议。

## iSCSI术语介绍

这些术语是相互引用的,而且为了易于理解，后面会附图说明。在后面的章节中会引用这些术语,进而可以加深理解。

### Network Entity（网络实体）

Network Entity(网络实体)可以是一个设备或者网关。一个网络实体必须有一个或者多个[Network Portal](#_Network_Portal（网络入口）)(网络入口)，每个网络入口可以通过[iSCSI Node](#_iSCSI_Node（iSCSI节点）)(iSCSI节点)用以访问IP 网络

。一个网络实体中有一个或多个iSCSI Node(iSCSI节点)。这个概念不是iSCSI特有的。

### Network Portal（网络入口）

网络入口是网络实体的组成部分。网络入口有一个TCP/IP网络地址，网络实体中的[iSCSI Node](#_iSCSI_Node（iSCSI节点）)用这个地址建立iSCSI session连接。

Initiator中的网络入口通过IP地址来标示一个网络入口。Target通过IP地址和监听端口来标示一个网络入口

### iSCSI Node（iSCSI节点）

iSCSI节点代表一个[iSCSI initiator](#_iSCSI_Initiator_Node)或者[iSCSI target](#_iSCSI_Target_Node).一个网络实体中有一个或者多个iSCSI Node。一个iSCSI Node通过一个或多个网络入口来访问。iSCIS节点是通过iSCSI名字来唯一标示的。

Open-iscsi 使用术语node指代一个target上的网络入口(a portal on a target)。所以在node模式下，需要参数 —targetname(-T) 和 –-protal(-p)

### Session

要相互通信，[iSCSI Initiator](#_iSCSI_Initiator_Node)和 [target](#_iSCSI_Target_Node)需要建立 iSCSI 会话.Session(会话)包含连接initiator 和target的多个TCP连接。TCP链接可以添加到会话中，或从会话中移除。在一个会话中，通过每一个链接，一个initiator看到的都是同一个target。

在open-iscis中，一个node包含一个session，一个session中有一个连接。

### TSIH

即Target Session Identifying Handle ,TSIH唯一标示一个session中的target端，是在target端生成的。

注：在lio的具体实现中，没发现这个标志。

### iSCSI Initiator Node

iSCSI Initiator Node也就是 "initiator". 如果不加修饰符，"initiator"指代一个initiator port。其他场合根据上下文来判断（port或者device）.参考iSCSI Node。

### iSCSI Target Node

也就是 The "target".参考iSCSI Node。一个target中可以有多个[LUN](#_LUN)

### SSID (Session ID):

在initiator发起[login操作](#_登录已发现target。)的时候，会在iSCSI initiator和iSCSI target之间建立一个会话(session)，每个session有唯一的session ID(SID)。在open-iscsi的实现中，这可以在[session模式](#_显示所有已登录的target信息.)下查看这个SID

### targetIQN(TargetName)

iqn(iSCSI Qualified name),targetIQN表示target的限定名(iSCSI Target Name)。这个名字必须是全球唯一的。可以被initiator在[discovery阶段](#_发现target/portals(Discoverying_iSCSI)被发现。

TargetName 的命名是用户自定义的字符串，最好是按照Target’s T10 WWN的命名规则来。

样例：

TargetName=iqn.1993-11.com.disk-vendor:diskarrays.sn.45678

### InitiatorName（InitiatorIQN）

InitiatorName唯一标示一个initiator端。是全球唯一的。

当initiator发起[login操作](#_登录已发现target。)成功之后，在[session模式](#_显示所有已登录的target信息.)下可以看到发起连接端的InitiatorName。

而且，InitiatorName可以用来进行CHAP验证(连接)：Target端根据InitiatorName来指示所访问的initiator端。

在open-iscsi的实现中，在[启动iscsid](#_open-iscsi脚本启动流程)的过程中会生成这个InitiatorName，位于文件/etc/iscsi/initiatorname.iscsi中

InitiatorName的一些样例

InitiatorName=iqn.1992-04.com.os-vendor.plan9:cdrom.12345

InitiatorName=iqn.2001-02.com.ssp.users:customer235.host90

### Portal Groups

在一个[iSCSI Node](#_iSCSI_Node（iSCSI节点）)中，每一个网络入口[Network Portal](#_Network_Portal（网络入口）)都属于唯一一个Portal Groups。

iSCSI Target Portal Group 简写为TPG.

在lio的实现中，一个TPG 通过关联一个[Storage Object](#_Storage_Object)和一个或多个[LUN](#_LUN)来构成，TPG通过[TargetIQN](#_targetIQN(TargetName))和一个[TPGT](#_Target_Portal_Group)及一个[LUN tag](#_LUN_tag)来标示。

### Target Portal Group Tag

是一个16-bit的数字，唯一标示一个iSCSI Target Portal Group，简称TPGT

### iSCSI Endpoints

在lio的实现中，一个iSCSI Endpoint 通过关联一个 [targetIQN](#_targetIQN(TargetName)) 和 Target Portal Group (TPG)来构成。

这个概念等价于[iSCSI Target Node](#_iSCSI_Target_Node)，其实就是一个target

### LUN

即逻辑单元号(Logical Unit Number)。一个[target](#_iSCSI_Target_Node)中可以有多个LUN.

其详细解释，参看[SCSI 设备在 Linux 系统上的地址映射](#_SCSI_设备在_Linux)

### LUN tag

在lio的实现中 LUN tag是一个序数数字,用于表示某个[target](#_iSCSI_Target_Node)中的LUN。

### Backstore

Backstore是lio-utils实现中的概念，指的是[iSCSI Target](#_iSCSI_Target_Node)s所能输出的存储设备类型，包括

• File IO

• Block IO

• Raw SCSI disk

• RAMDISK

Backstore 是通过它Virtual [HBA](#_映射地址各部分的含义) (host bus adapter)类型和[Storage Object](#_Storage_Object) name来引用的。

上面这些存储类型要想被iSCSI识别，必须由tcm\_node转换为iSCSI可以识别的i[SCSI Storage Object](#_Storage_Object)

### Storage Object

这是lio实现中的概念。一个iSCSI Storage Object通过tcm\_node命令配置或创建一个[Backstore](#_Backstore)来生成的，这样生成的Storage Object都有一个特定的名字。

### ACL

访问控制列表（Access Control List）,这个是LIO中的概念。可以限制某个target被某个initiator访问，需要首先把Initiator加入控制列表，使得target中的每个LUN都可以被initiator访问。

## iSCSI术语的图例

### RFC3720中建议的模型

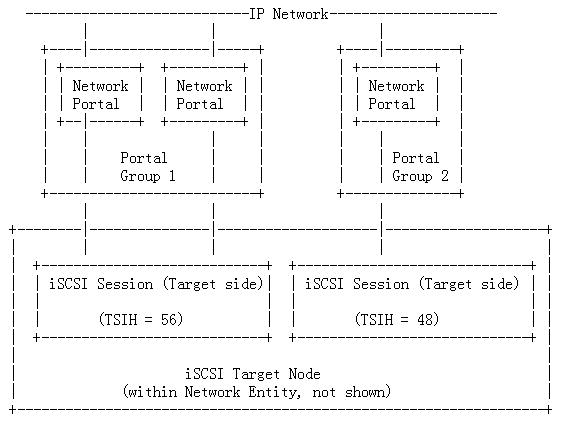


图3-1 iSCSI模型

### Entity,Portal，Session等之间的关系

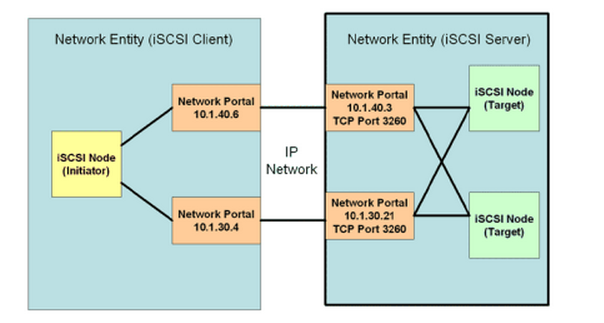


图3-2 iSCSI各个部分之间的关系

* 一个发起(initiator)设备网络入口和一个目标设备(target)网络入口之间会建立一个连接
* 一个发起(initiator)设备iSCSI节点和一个目标设备(target)iSCSI节点之间的一组TCP连接构成一个Session。
* 在open-iscsi的实现中，一个session中只有一个tcp连接(待确认)。

## SCSI 设备在 Linux 系统上的地址映射

### SCSI设备地址映射的组成

SCSI设备在Linux系统中的逻辑地址映射由四个部分组成，这四个部分是有层次结构(level hierarchical)的：

* SCSI adapter number [host]
* channel number [bus]
* id number [target]
* lun [lun]

其中方括号中使用的术语是`设备伪文件系统`(devfs)所使用的名称惯例。

将这四部分通过“:”连在一起表示，即常见的4段式的地址映射，如2:0:0:1。

iSCSI设备也不例外。如下面输出的一个iSCSI设备54:0:0:0

/sys/devices/platform/host54/session7/target54:0:0/54:0:0:0

如果一个target上有[多个lun](#_显示所有已登录的target信息.)，其目录可能为：

root@ubuntu-nbs1:/sys/devices/platform/host60/session13/target60:0:0# ls

60:0:0:0 60:0:0:1 power subsystem uevent

其中60:0:0:0 60:0:0:1 是同一个target中两个LUN。

### 映射地址各部分的含义

SCSI adapter number是一个数字，标示一个适配器卡，也称为HBA(Host Bus Adapter)，即主机总线适配器。SCSI adapter number是linux内核按从0开始按升序分配的。

每一个HBA控制着一个或多个SCSI bus。SCSI bus的类型常见的有IDE/ATA,sATA,SAS,iSCSI等。

每一个SCSI bus有多个SCSI设备和其相连接。SCSI总线自己并不直接和硬盘之类的设备通讯，而是通过控制器来和设备建立联系。一个独立的SCSI总线最多可以支持16个设备

每一个SCSI设备包含多个LUN.

因为SCSI设备的映射地址以数字的形式表现，本身就具备层次性，因此映射地址的主要作用有：

* 内核通过映射地址为设备命名
* 用户通过映射地址确定系统设备文件和实际物理设备之间的对应关系。

### 如何查看映射地址

查看SCSI设备的映射地址主要有3种方法：

* # cat /proc/scsi/scsi
* dmesg命令或者 cat /var/log/dmesg

内核识别到SCSI设备时，内核日志中会出现该设备的各种信息，包括映射地址。

* sg\_map命令

sg\_map是sg3-utils工具包中的命令，sg\_map -x可以打印出设备文件和SCSI映射地址之间的对应关系。

### LUN的进一步解释

早期的SCSI系统采3位ID，一个SCSI总线上只能接驳8个设备，除了SCSI 控制卡占用了一个ID外，只剩7个设备号可用。尽管SCSI以后采用了4位ID，扩展到了16个设备，但对于现代的存储设备，这种个位数的ID识别号显然 不够用。

LUN ID的作用在于扩充了Target ID。每个Target下都可以有多个LUN Device，我们通常简称LUN Device为LUN，这样就可以说每个设备的描述就有原来的Target x变成Target x LUN y了，这样描述设备的能力增强了，同时也能对存储空间内的磁盘进行单独的管理。

LUN ID不等于某个设备，只是个号码而已，不代表任何实体属性，在我们的实际环境里，我们碰到的LUN可能是磁盘空间、磁带机等。

从linux 的系统的SCSI设备的视角来看，LUN是SCSI地址的一个组成部分。

## iSCSI安装

lio和open-iscsi分别是target端和initiator端的一个实现,其安装包分别为lio-uitls和open-iscsi。本文中主要讲解这两种工具的使用方式

### 安装及卸载

* lio-utils的安装卸载

apt-get install lio-utils

apt-get remove lio-utils

* open-iscsi的安装卸载

apt-get install open-iscsi

apt-get remove --purge open-iscsi

使用—purge选项才能够真正的删除配置文件(/etc/iscsi/)等信息。

* open-iscsi安装成功时/etc/iscsi/initiatorname.iscsi文件内容为

root@debianNBS1:~# cat /etc/iscsi/initiatorname.iscsi

GenerateName=yes

[启动](#_open-iscsi脚本启动流程)open-iscsi的daemon进程

root@debianNBS1:~# /etc/init.d/open-iscsi start 或者

root@debianNBS1:~# service open-iscsi start

/etc/iscsi/initiatorname.iscsi文件内容如下，这时就生成了[InitiatorName](#_InitiatorName)

root@debianNBS1:~# cat /etc/iscsi/initiatorname.iscsi

## DO NOT EDIT OR REMOVE THIS FILE!

##........

InitiatorName=iqn.1993-08.org.debian:01:116391619e33

### 启动流程

#### 服务启动

lio 和open-iscsi都可以通过标准的linux service服务来启动.

* + - 1. 启动脚本文件

在debian系统下，lio和open-iscsi的启动文件分别为

/etc/init.d/target

/etc/init.d/open-iscsi

* + - 1. 启停方式

是标准的linux service启停方式

**root@ubuntu-nbs2:~# service target**

Usage: /etc/init.d/target {start|startbak|stop|status|restart} **#或者**

**root@ubuntu-nbs2:~# /etc/init.d/target**

Usage: /etc/init.d/target {start|startbak|stop|status|restart}

**root@ubuntu-nbs2:~# service open-iscsi**

Usage: /etc/init.d/open-iscsi {start|stop|restart|force-reload|status}**#或者**

**root@ubuntu-nbs2:~# /etc/init.d/open-iscsi**

Usage: /etc/init.d/open-iscsi {start|stop|restart|force-reload|status}

#### 执行lio命令时的一些行为

执行**root@ubuntu-nbs2:~# /etc/init.d/target stop时会显示**[iSCSI Target Endpoints](#_iSCSI_Endpoints)和已经配置的[storage objects](#_Storage_Object)信息，并卸载掉。

执行**root@ubuntu-nbs2:~# /etc/init.d/target status时会显示**[iSCSI Target Endpoints](#_iSCSI_Endpoints)和已经配置的[storage objects](#_Storage_Object)信息

#### open-iscsi脚本启动流程

当执行**root@ubuntu-nbs2:~# service open-iscsi start** 时**，**

会读取脚本/etc/init.d/open-iscsi,做如下事情

* **正常检查:**

/etc/iscsi/iscsid.conf文件是否存在

检查iscsid进程文件/run/iscsid.pid是否存在，并发送kill -0 检测进程是否存在。

检查/etc/iscsi/initiatorname.iscsi是否存在

检查文件/etc/iscsi/initiatorname.iscsi[内容](#_安装及卸载)是否包含字符串 ^GenerateName=yes

如果存在字符串 ^GenerateName=yes ，判断程序/usr/sbin/iscsi-iname是否存在

然后用usr/sbin/iscsi-iname生成initiatorname写入/etc/iscsi/initiatorname.iscsi中

如果不包含字符串 ^GenerateName=yes ，判断initiatorname是否合法

* **确保必须的**[**内核模块**](#_open-iscsi协议实现的体系结构)**存在:**

modprobe -q iscsi\_tcp 2>/dev/null || :

modprobe -q ib\_iser 2>/dev/null || :

* **启动:**

启动/usr/sbin/iscsid进程，并把进程文件写入/run/iscsid.pid

* **其他工作**

执行 [/usr/bin/iscsiadm -m node --loginall=automatic](#_登录已发现target。)

等等其他操作。

#### open-iscsi脚本停止流程

当执行**root@ubuntu-nbs2:~# service open-iscsi stop** 时**，**

会读取脚本/etc/init.d/open-iscsi,做如下事情

* 判断进程文件/run/iscsid.pid是否存在，并判断进程是否存活
* 如果存在就执行命令[/usr/bin/iscsiadm -m node --logoutall=all](#_登出target。)登出所有的target，并退出进程。

## open-iscsi协议实现的体系结构

open-iscsi是iSCSI-initiator的一种实现，是RFC3720协议的不完全实现.

open-iscsi协议的实现分**两个平面**，**三个模块**。

**两个平面**指的是控制平面和数据平面。

**控制平面**：控制平面用于管理和控制open-iscsi协议的运行，如命令的发送、节点管理、故障恢复。

**数据平面**：负责tcp/ip连接的管理，数据的传送等。

**三个模块**指的是iscsiadm、iscsid及iscsi\_if,其中前两个模块是用户空间模块，隶属于控制平面。第三个模块是内核模块、属于数据平面。

* iscsiadm 用户空间的命令行工具

管理(upadate、delete 、insert)session和target的[持久化数据库](#_持久化数据库)。

通过unix域socket发送命令给iscsid

实现iSCSI discovery，把结果存入数据库。

* iscsid 用户空间iSCSI daemon程序

实现session的管理，实现[login](#_登录已发现target。)、[logout](#_登出target。)、Nop-In/Out等操作

同时通过netlink和iscsiadm和iscsi\_if模块通信，连接这两个模块。

由于iscsiadm会持久化一些信息到本地，一些iSCSI Node 或者Discovery操作不需要运行iSCSI daemon

* iscsi\_if [内核空间模块](#_open-iscsi脚本启动流程)

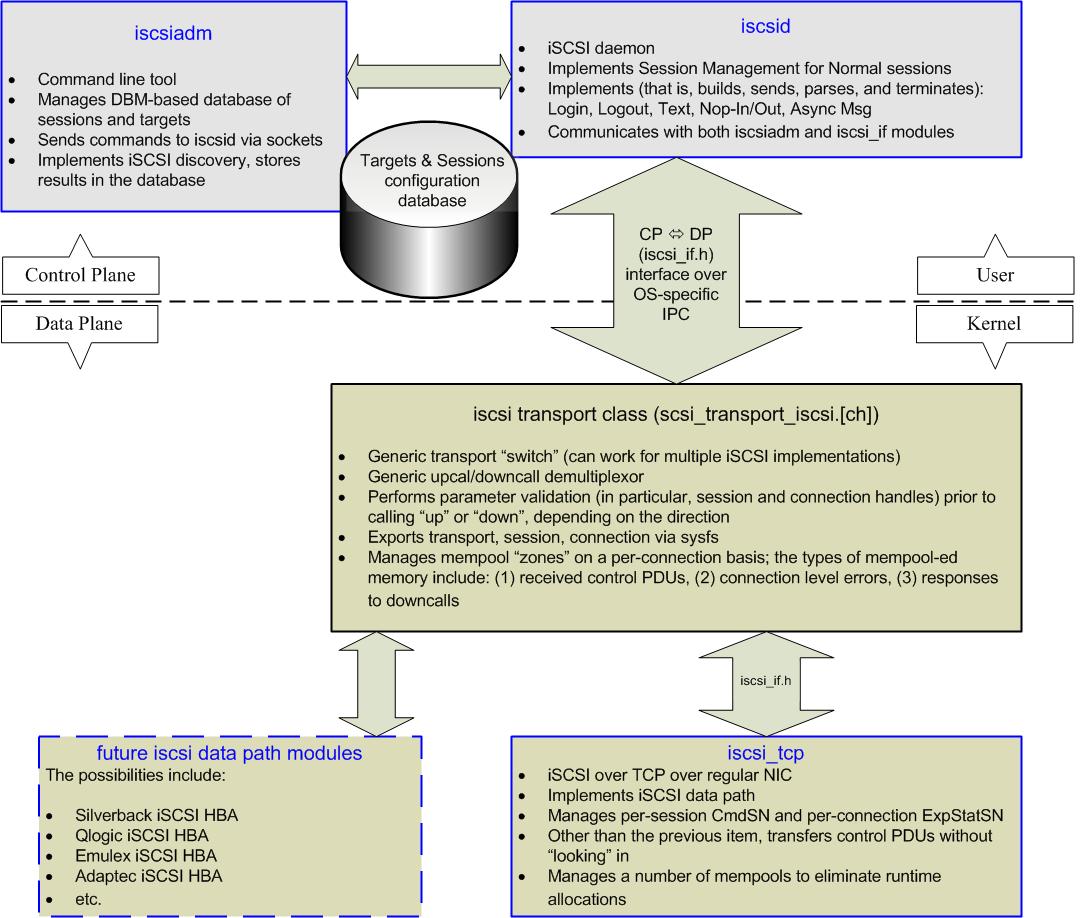
通过netlink与iSCSI daemon进程进行通信。

管理tcp/ip连接的建立和释放

实现iSCSI数据通道

等等

其实现结构大致为：



# iSCSI target (lio)

Lio是一个开源的storage target的一种实现标准，他支持所有流行存储结构：[iSCSI](http://linux-iscsi.org/wiki/ISCSI), [Fibre Channel](http://linux-iscsi.org/wiki/Fibre_Channel)([QLogic](http://www.qlogic.com/)), [FCoE](http://linux-iscsi.org/wiki/Fibre_Channel_over_Ethernet), [InfiniBand](http://linux-iscsi.org/wiki/InfiniBand)/[SRP](http://linux-iscsi.org/wiki/SRP) ([Mellanox](http://www.mellanox.com/)), [vHost](http://linux-iscsi.org/wiki/VHost)。Lio可以把一个块设备输出到iscsi　target中，其中用到两个工具tcm\_node和lio\_node。

## 创建Storage Object

创建[Storage Object](#_Storage_Object) 即是把要输出到target的设备([Backstore](#_Backstore))转换为iscisi可识别的形式，创建好的Storage Object都有一个名字来标示.

[Backstore](#_Backstore)的类型可以为file、block或者ramdisk，本文主要讲述block设备。

### 创建Block IO形式的Storage Object

Block IO Backstore的类型可以为如下几种(需要提供一个udev path)

• /dev

• /dev/mapper

• /dev/VolumeGroup/LogicalVolume

• /dev/disk

由Block IO Backstore生成iSCSI Storage Object形式如下*：*

**tcm\_node --block <HBA>/<StorageObjectName> <PathName>/<BlockDeviceName>**

* 创建*Storage Object*

**# tcm\_node --block iblock\_0/test\_lv0 /dev/vgszx2/lv0**

Status: DEACTIVATED Execute/Left/Max Queue Depth: 0/128/128 SectorSize: 512 MaxSectors: 512

iBlock device: dm-3 UDEV PATH: /dev/vgszx2/lv0

Major: 253 Minor: 3 CLAIMED: IBLOCK

* 创建Storage Object的同时生成T10 WWN Unit Serial序列号。这个序列号在这个文件中：

/sys/kernel/config/target/core/iblock\_0/test\_lv0/wwn/vpd\_unit\_serial。

可以 用命令查看

**# tcm\_node --wwn iblock\_0/test\_lv0**

T10 VPD Unit Serial Number: 9b48a694-35ec-409a-ac1f-804f26da9908

而且<HBA>/<StorageObjectName>可以看做Storage Object的全限定名，可以唯一标示这个设备。

* 显示设备的属性

**# tcm\_node --listdevattr iblock\_0/test\_lv0**

\TCM Storage Object Attributes for /sys/kernel/config/target/core/iblock\_0/test\_lv0

unmap\_granularity\_alignment: 0

**……**

* 查看Storage Object所对应的设备路径udev Path

**# tcm\_node --udevpath iblock\_0/test\_lv0**

/dev/vgszx2/lv0

* 释放*Storage Object*

**tcm\_node --freedev iblock\_0/test\_lv0**

* 查看所有的Storage Object的参数信息

# tcm\_dump --s

* 其他的命令行选项可以通过 # tcm\_node -h来查看

## 创建iSCSI Endpoint(iSCSI端点)

之后，就可以利用生成好的iSCSI Storage Object和lio\_node命令创建[iSCSI Endpoint](#_iSCSI_Endpoints)了。

一个iSCSI 端点由两部分构成 : IQN和 TPG.

* [TPG](#_Portal_Groups) (iSCSI Target Portal Group)通过关联一个[Storage Object](#_Storage_Object)构成，其中Storage Object可以包含一或多个[LUN](#_LUN)，TPG具体是一个序数。
* IQN (iSCSI Qualified Name)是用户自定义的一个字符串，推荐使用Target’s T10 WWN，可以使得这个iSCSI端点全球唯一。

### 创建iSCSI端点(target)

* 从逻辑卷Backstore创建[iSCSI端点](#_iSCSI_Endpoints)，也就是target.

lio\_node --addlun <TargetIQN> <TPG#> <LUN#> <LUNName> <HBA>/<StorageObjectName>

**# lio\_node --addlun com.netease 1 0 lun\_my\_lv iblock\_0/test\_lv0**

Successfully created iSCSI Target Logical Unit

其中 iblock\_0/test\_lv0 即为之前生成的Storage Object

可以在这个iSCSI端点上(即一个target中) 可以通过指定不同的<LUN#>输出[多个LUN](#_显示所有已登录的target信息.)。

注意查看目录/sys/kernel/config/target/iscsi/com.netease生成了，里面存放所有LUN的信息。

* 列出所有iSCSI 端点

**# lio\_node --listendpoints**

\------> test:123456:loop3

\-------> tpgt\_1 TargetAlias: LIO Target

TPG Status: ENABLED

TPG Network Portals:

\-------> 0.0.0.0:3260

TPG Logical Units:

\-------> lun\_1/iscsi01 -> target/core/iblock\_0/test\_loop3

\-------> lun\_0/iscsi01 -> target/core/iblock\_0/test\_loop4

* 从iSCSI端点上删除一个iSCSI LUN

lio\_node --dellun <TargetIQN> <TPG#> <LUN#>

**# lio\_node --dellun com.netease 1 0**

Successfully deleted iSCSI Target Logical Unit

* 更多的命令查看lio\_node –h

### 创建iSCSI网络入口

上面生成的iSCSI端点并不能通过网络来识别，用命令查看

**# iscsiadm -m discovery -t sendtargets -p 114.113.199.19:3260**

……

114.113.199.19:3260,-1 com.netease

…..

发现第二项是-1

* 创建iSCSI网络入口([Network Portal](#_Network_Portal（网络入口）))

lio\_node --addnp <TargetIQN> <TPG#> <IPaddr>:<Port#>

**# lio\_node --addnp com.netease 1 0.0.0.0:3260**

Successfully created network portal: 0.0.0.0:3260 created com.netease TPGT: 1

再次查看，# iscsiadm -m discovery -t sendtargets -p 114.113.199.19:3260

仍然是-1

* 此时的iSCSI端点仍然不能从Initiator登陆，要想登陆必须打开选项：

lio\_node --enabletpg <TargetIQN> <TPG#>

**# lio\_node --enabletpg com.netease 1**

Successfully enabled iSCSI Target Portal Group: com.netease 1

* 打开登陆选项之后，发现仍然不能登陆。原因是登陆时必须经过[CHAP验证](#_CHAP验证)，

此时可以通过关闭验证(Disabling Authentication)的方式来解决。

需两个步骤

* 打开demo模式

打开demo模式意味着所有的Initiator可以访问iSCSI 端点上所有的LUN

lio\_node --demomode <TargetName> <TPG#>

**# lio\_node --demomode com.netease 1**

Successfully disabled Initiator ACL mode (Enabled DemoMode) on iSCSI Target Portal Group: com.netease 1

* 关闭CHAP验证

lio\_node --disableauth <TargetName> <TPG#>

**# lio\_node --disableauth com.netease 1**

Successfully disabled iSCSI Authentication on iSCSI Target Portal Group: com.netease 1

经过以上步骤，iSCSI Initiator就可以登录iSCSI 端点来访问上面的LUN了。

**# iscsiadm -m node -T com.netease -p 114.113.199.19:3260 --login**

Logging in to [iface: default, target: com.netease, portal: 114.113.199.19,3260]

Login to [iface: default, target: com.netease, portal: 114.113.199.19,3260] successful.

* 去掉iSCSI端点的写保护属性

此时登录上去的设备不能被[正常挂载](#_使用网络设备)，必须去掉写保护属性，然后重新挂载。

**# lio\_node --settpgattr=com.netease 1 demo\_mode\_write\_protect 0**

### CHAP验证

CHAP(Challenge Handshake Authentication Protocol)是一台计算机向另一个台计算机进行身份认证的一种认证机制。

CHAP验证的过程为

* 链路建立阶段结束之后，认证者向对端点发送“challenge”消息。
* 对端点用经过单向哈希函数计算出来的值做应答
* 认证者根据它自己计算的哈希值来检查应答，如果值匹配，认证得到承认；否则，连接应该终止。
* 经过一定的随机间隔，认证者发送一个新的 challenge 给端点，重复步骤 1 到 3 。

Intimate和target都可以进行CHAP验证，但这里只说明对target进行CHAP验证。上一节输出的target可以被任何initiator访问。可以通过CHAP验证只允许特定的initiator访问。

* 打开CHAP验证

首先确保Initiator已经登出了这个target，这个开关缺省是打开的。

**# lio\_node --enableauth test:123456:loop0 1**

* 使iSCSI LUN可以被特定的Initiator访问

iSCSI Initiator是通过 [InitiatorIQN](#_InitiatorName)来引用的

lio\_node --addlunacl <TargetIQN> <TPG#> <InitiatorIQN> <LUN#> <MappedLUN#>

可以把一个InitiatorIQN加入[ACL](#_ACL)中，使得这个<TargetIQN>可以访问这个<TargetIQN>中的<LUN#>,<MappedLUN#>的概念是，initiator LUN ACLs可以映射任意的<LUN#>值，两者一般一致的.

**#lio\_node --addlunacl test:123456:loop0 1 iqn.1993-08.org.debian:01:fcc9228c36cc 0 0**

* 设置用户名和密码

lio\_node --setchapauth <TargetIQN> <TPG#><InitiatorIQN> <LoginName> <Password>

LoginName为登录名,Password为登陆密码 .在<InitiatorIQN>登陆这个target时必须提供<LoginName> 和<Password>

这种信息可以使用命令或者在文件(/etc/iscsi/nodes/<targetName>/defalt)中看到。

**# lio\_node --setchapauth test:123456:loop0 1 iqn.1993-08.org.debian:01:fcc9228c36cc nbs netease**

* 设置TPG的属性

设置TPG的属性为cache\_dynamic\_acls

lio\_node --settpgattr=TARGET\_IQN TPGT <ATTRIB> <VALUE>

**lio\_node --settpgattr=test:123456:loop0 1 cache\_dynamic\_acls 0**

* 允许Initiator登陆：

lio\_node --enabletpg <TargetIQN> <TPG#>

**# lio\_node --enabletpg com.netease 1**

### 输出一个target步骤

总的来说

* 输出一个可以被访问的、没有CHAP验证的target需要下面几个步骤

**#!/bin/bash**

**device=$1 # 设备名**

**targetIQN=test:123456:${device}**

**lun=iscsi00**

**# 生成Storage Object**

**tcm\_node --block iblock\_0/test\_${device} /dev/${device}**

**# 输出target**

**lio\_node --addlun test:123456:${device} 1 0 $lun iblock\_0/test\_${device}**

**# 可以查看已经输出的target**

**lio\_node –listendpoints**

**# 创建 network portal**

**lio\_node --addnp test:123456:${device} 1 0.0.0.0:3260**

**#允许从 iSCSI initiator 登陆**

**lio\_node --enabletpg $targetIQN 1**

**# demo模式**

**lio\_node --demomode $targetIQN 1**

**# 去掉验证**

**lio\_node --disableauth $targetIQN 1**

**# 去掉写保护**

**lio\_node --settpgattr=$targetIQN 1 demo\_mode\_write\_protect 0**

* 输出一个可以被访问的、有CHAP验证的target需要下面几个步骤

**#!/bin/bash**

**# 生成Storage Object**

**tcm\_node --block iblock\_0/test\_loop0 /dev/loop0**

**# 输出target**

**lio\_node --addlun test:123456:loop0 1 0 iscsi00 iblock\_0/test\_loop0**

**# 创建 network portal**

**lio\_node --addnp test:123456:loop0 1 0.0.0.0:3260**

**# 打开验证**

**lio\_node --enableauth test:123456:loop0 1**

**# 允许特定的initiator访问**

**lio\_node --addlunacl test:123456:loop0 1 iqn.1993-08.org.debian:01:fcc9228c36cc 0 0**

**#　设置用户名和密码**

**lio\_node --setchapauth test:123456:loop0 1 iqn.1993-08.org.debian:01:fcc9228c36cc nbs netease**

**lio\_node --settpgattr=test:123456:loop0 1 cache\_dynamic\_acls 0**

**lio\_node --enabletpg test:123456:loop0 1**

* 登陆一个可以被访问的、有CHAP验证的target需要下面几个步骤

**#!/bin/bash**

**# 发现target**

**iscsiadm -m discovery -t st -p 192.168.146.233:3260**

**iscsiadm -m node -T test:123456:loop0 --op=update --name=node.session.auth.authmethod --value=CHAP**

**iscsiadm -m node -T test:123456:loop0 --op=update --name=node.session.auth.username --value=nbs**

**iscsiadm -m node -T test:123456:loop0 --op=update --name=node.session.auth.password --value=netease**

**iscsiadm -m node -T test:123456:loop0 -p 192.168.146.233:3260 -l**

# iSCSI initiator（open-iscsi）

[open-iscsi](#_open-iscsi协议实现的体系结构) 是iSCSI initiator的一种实现标准，其命令行工具为iscsiadm

## open-iscsi的使用方式.

### 持久化数据库

[open-iscsi](#_open-iscsi协议实现的体系结构)有两个持久化数据库，包括

* /etc/iscsi/nodes/
* /etc/iscsi/send\_targets

这两个目录中的内容是在target发现的时候写入的，里面存放着每个target的iSCSI参数信息。删除持久化信息不会影响已经login的target。

## 初步使用

### 发现并登陆target

* 发现target：

# iscsiadm -m discovery -t sendtargets -p 114.113.199.19:3260

…..

114.113.199.19:3260,1 com.netease

* 登陆已发现target：

# iscsiadm -m node -T com.netease -p 114.113.199.19:3260 --login

Logging in to [iface: default, target: com.netease, portal: 114.113.199.19,3260]

Login to [iface: default, target: com.netease, portal: 114.113.199.19,3260] successful.

或者iscsiadm -m node -T com.netease -p 114.113.199.19:3260 -l

* 登出target

iscsiadm -m node -T com.netease -p 114.113.199.19:3260 --logout

Logging out of session [sid: 1676, target: com.netease, portal: 114.113.199.19,3260]

Logout of [sid: 1676, target: com.netease, portal: 114.113.199.19,3260] successful.

或者iscsiadm -m node -T com.netease -p 114.113.199.19:3260 -u

iscsiadm -m node -T iqn.2005-03.com.max -p 192.168.120.198 –u

* 更多的用法查看 参看5.3节

### 使用网络设备

通过iscsiadm登录进去之后，就可以使用该设备了。

* 登录

**iscsiadm -m node -T com.netease -p 114.113.199.19:3260 –login**

* 查看设备

**# iscsiadm -m session -P3**

Target: com.netease

Current Portal: 114.113.199.19:3260,1

……..

scsi1682 Channel 00 Id 0 Lun: 0

Attached scsi disk **sdat** State: running

可以看出，检测到的网络设备为sdat

* 挂载设备

现在可已挂载该设备并使用了。

* 正如[4.2.2节](#_创建iSCSI网络入口)所述，如果不去掉iSCSI端点的写保护属性会报如下错误

**# mount /dev/sdal /home/ebs/dir/sdat**

mount: block device /dev/sdal is write-protected, mounting read-only

mount: you must specify the filesystem type

* 格式化并挂载成功之后用df查看

/dev/sdat 194M 5.6M 179M 4% /home/ebs/dir/sdat

* 使用设备

现在使用设备都是在操作网络设备。

## open-iscsi常用命令

#### 发现target/portals(Discoverying iSCSI targets/portals )

在**登录**target之前，必须先**发现(discovery)target**。

**root@ubuntu-nbs1:~/ # iscsiadm -m discovery -t sendtargets -p 192.168.146.233:3260**

其中

* 参数 -m, --mode op 指定模式op，op必须是discovery，discoverydb, node, session等之一。如果不指定其他参数，对于 discoverydb 和 node模式，会打印所有各自的记录；对于 session 模式,会显示所有活动的session和连接。
* 参数-t, --type=type type必须是sendtargets (简写为st), slp, isns 或者fw.之一。目前用到的只有sendtargets 。这个选项仅在discovery模式下使用。指定type为sendtargets时必须指定-p，以指示去哪个portal下去发现target.
* 参数-p, --portal=ip[:port]。所使用的target portal，用ip-address和port来标记，默认的端口号为3260.其中ip还可以使用hostname.

这个选项仅用在discovery模式，或者在node模式下的new操作。而且在node模式下必须有参数—target

上面的命令会做以下动作

* 检查iscsid是否启动，如果没有，就启动isicsid进程。
* 使用配置文件[/etc/iscsi/iscsid.conf](#_/etc/iscsi/iscsid.conf详解)中的配置项来发现target
* 把target信息持久化到目录/etc/iscsi/send\_targets/和/etc/iscsi/nodes/中
* 等价于 **iscsiadm -m discoverydb -t st -p 192.168.146.233:3260 --discover**

**删除持久化信息**

root@ubuntu-nbs1:~/ # iscsiadm -m discovery -p 192.168.146.233:3260 --op=delete

其中

* -o, --op=op 指定数据库操作op，op必须是new, delete,update, show或者nonpersistent之一。在discovery模式下仅支持delete和show操作。
* 这个动作会删除所有指定portal的持久化数据库，即目录/etc/iscsi/send\_targets/和/etc/iscsi/nodes下的信息.但是不会断开已经login的tcp连接。

**显示指定portal的参数信息**

**root@ubuntu-nbs1:~/ # iscsiadm -m discovery -p 192.168.146.233:3260 --op=show**

**显示所有discovery记录的粗略信息。**

**root@ubuntu-nbs1:~/program/iscsi/Open-iscsi/log# iscsiadm -m discovery --print=1**

其中

* 参数-P, --print=printlevel，根据指定的模式指定打印级别，级别越高打印的越详细。Printlevel根据特定命令而定,介于[0,1]或[0,3]等
* 这个动作不需要iscsid在运行，仅仅是去目录/etc/iscsi/send\_targets/下去读取持久化的target信息。

#### 显示target节点信息。

**查看节点信息**

执行target descovery动作之后，可以在node模式下查看所有的target节点信息。

**root@ubuntu-nbs1:~/ # iscsiadm -m node**

这个动作不需要iscsid在运行，仅仅是去目录/etc/iscsi/nodes/ 下去读取持久化的node信息。

另外还可以指定target、portal、op等

**root@ubuntu-nbs1:~/ # iscsiadm -m node --targetname=test:123456:loop2**

其中

* 参数-T, --targetname=targetname 指定所使用的targetname，
* 这个操作显示test:123456:loop2的所有参数的详细信息。

**root@ubuntu-nbs1:~/ # iscsiadm -m node -p 192.168.146.233:3260,1**

显示portal中所有target的详细信息

#### 登录已发现target。

**登录特定target**

**root@ubuntu-nbs1:~/ # iscsiadm -m node -T test:123456:loop0 -p 192.168.146.233 –l**

**root@ubuntu-nbs1:~/ # iscsiadm -m node -T test:123456:loop0 –l**

**其中**

* 参数-l, --login。对于node模式，**登录**特定的target，对于discovery模式，登录所有已发现的target.这个选项仅对discovery和node 模式合法。
* 上面的命令表示登录portal 192.168.146.233上指定的target test:123456:loop0.登录时可以不指定portal。
* 登录的过程是和特定portal建立TCP连接的过程。
* 每次登录，在session模式下显示的SID会有所不同。

**登录所有已发现target**

**root@ubuntu-nbs1:~/# iscsiadm -m discovery -t sendtargets -p 192.168.146.233:3260 –l**

* 上面的命令会登录在指定portal中发现的所有的target
* 相当于先discovery，然后login

**登录所有的target**

**root@ubuntu-nbs1:~/ # iscsiadm -m node -L all**

**其中**

* 参数-L, --loginall==[all|manual|automatic]

这个选项仅仅用在node模式下，登录所有的target.

#### 登出target。

**登出特定的target**

**root@ubuntu-nbs1:~/ # iscsiadm -m node -T test:123456:loop0 -p 192.168.146.233 –u**

**其中**

* 参数-u, --logout 表示登出特定的target .这个选项只在node和session模式下有效。
* 上面的命令表示登出targettest:123456:loop0
* 登出的过程是和特定portal断开TCP连接的过程。

**登出所有的target**

**root@ubuntu-nbs1:~/ # iscsiadm -m node -U all**

**其中**

* 参数 -U, --logoutall==[all,manual,automatic] 登出所有的的target.这个选项仅适用于node模式。

#### 显示所有已登录的target信息.

**显示已登录的所有target列表**

**root@ubuntu-nbs1:~/ # iscsiadm -m session**

* Session模式下，这个命令显示所有活动的session和连接的信息。
* 只有login的target才能在这个模式下看到。

**显示session详细信息**

**root@ubuntu-nbs1:~/ # iscsiadm -m session -P 2**

* 其中显示级别位于[0,3],级别越高显示的越详细。

1 = 打印基本session信息，包括sessionID等。

2 = 打印iscsi参数的详细信息。‘

3 = 打印SCSI信息，包括LUNs，设备状态等。

* 里面包含有 SID 、挂载的盘符, Initiatorname, Host Number等详细信息。

注：1.SID在每次登录（login）都会有所不同。

2.Host Number可以在目录/sys/devices/platform下寻找这个iSCSI设备的详细信息，譬如/sys/devices/platform/host60

其中一个[target可以输出多个lun](#_创建Block_IO形式的Storage_Object)，比如在target test:123456:loop3中有两个lun时的详细信息,而且这些信息都可以在文件/sys/devices/platform/下看到参看[SCSI设备地址映射的组成](#_SCSI设备地址映射的组成)。

root@ubuntu-nbs1:~/program/iscsi/example# iscsiadm -m session -r 13 -P3

iSCSI Transport Class version 2.0-870

version 2.0-873

Target: test:123456:loop3

Current Portal: 192.168.146.233:3260,1

Persistent Portal: 192.168.146.233:3260,1

\*\*\*\*\*\*\*\*\*\*

………………………………………………………….

SID: 13

\*\*\*\*\*

………………………………………………………….

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Attached SCSI devices:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Host Number: 60 State: running

scsi60 Channel 00 Id 0 Lun: 0

Attached scsi disk sdd State: running

scsi60 Channel 00 Id 0 Lun: 1

Attached scsi disk sde State: running

**显示指定session的详细信息**

**root@ubuntu-nbs1:~/program/iscsi/Open-iscsi/log# iscsiadm -m session -P 3 --sid=23**

**其中**

* 参数-r, --sid=sid | sysfsdir。所使用的session ID。这个ID可以通过iscsiadm -m session -P 3找到。这个参数仅适用于session模式**。**

#### discoverydb模式

**显示targe信息**

**root@ubuntu-nbs1:~/ # iscsiadm -m discoverydb -P1**

其中

* 参数 -m discoverydb 表示在discoverydb 模式下。
* 打印级别有两个**：**

0 = 平铺(flat)的方式 .

1 = 树形的信息.

* 这个动作不需要iscsid在运行，仅仅是去目录/etc/iscsi/send\_targets/下去读取持久化的target信息。
* -t和-p必须同时指定或者不指定。

**iscsiadm -m discoverydb -t sendtargets -p192.168.146.233:3260**

**发现target,并持久化信息到数据库**

**root@ubuntu-nbs1:~/ # iscsiadm -m discoverydb -t st -p 192.168.146.233:3260 --discover**

等价于

**root@ubuntu-nbs1:~/ # iscsiadm -m discovery -t st -p 192.168.146.233:3260**

其中

* 参数-D, --discover。指示iscsiadm去执行discovery操作。

**更新数据库**

**root@ubuntu-nbs1:~/ # iscsiadm -m discoverydb -t st -p 192.168.146.233:3260 -o update –discover**

这个命令做以下动作

* 如果数据库中已经存在相应记录，读取iscsi.conf配置文件更新记录信息。

等价命令 **iscsiadm -m discovery -t st -p 192.168.146.233:3260 -o update**

--op=[op] [--name=[name] --value=[value]]

有关update的进一步使用，比如如何更新配置项以覆盖全局的配置，请参考[/etc/iscsi/iscsid.conf详解](#_/etc/iscsi/iscsid.conf详解)

**其他**

iscsiadm -m discoverydb -t st -p 192.168.146.233:3260 -o new --discover

显示详细配置信息：iscsiadm -m node -o show -T test:123456:loop0

## /etc/iscsi/iscsid.conf详解

### /etc/iscsi/iscsid.conf初步认识

/etc/iscsi/iscsid.conf是open-iscsi的配置文件，这个文件同样可以位于 ~/.iscsid.conf。在[discovery](#_发现target/portals(Discoverying_iSCSI)等阶段都会首先读取这一配置文件。而且配置文件中的配置项是全局的，在需要做修改时，需要用[update操作](#_discoverydb模式)来更新这些配置项。

举例说明，在登陆一个有CHAP验证的target时，必须设置

* 验证类型，用户名，密码，在配置文件中有如下片段

# \*\*\*\*\*\*\*\*\*\*\*\*\*

# CHAP Settings

# \*\*\*\*\*\*\*\*\*\*\*\*\*

# To enable CHAP authentication set node.session.auth.authmethod

# to CHAP. The default is None.

#node.session.auth.authmethod = CHAP

# To set a CHAP username and password for initiator

# authentication by the target(s), uncomment the following lines:

#node.session.auth.username = username

#node.session.auth.password = password

* 如果不使用这个全局配置项，可以用update操作来更新以覆盖原来的配置,譬如

iscsiadm -m node -T test:123456:loop0 --op=update --name=node.session.auth.authmethod --value=CHAP

iscsiadm -m node -T test:123456:loop0 --op=update --name=node.session.auth.username --value=nbs

iscsiadm -m node -T test:123456:loop0 --op=update --name=node.session.auth.password --value=netease

### /etc/iscsi/iscsid.conf各配置项含义

配置文件中的配置项很丰富，这里不完全列举/etc/iscsi/iscsid.conf配置文件中的一些配置，措辞有可能不准确.

|  |  |  |  |
| --- | --- | --- | --- |
| 配置项种类 | 配置项 | 含义 | 备注 |
|  | iscsid.startup = /usr/sbin/iscsid | iscsi daemon程序位置 |  |
| Target端CHAP验证 | node.session.auth.authmethod = CHAP | 使用CHAP验证 |  |
| node.session.auth.username = username  16 | 设置target端验证的用户名 |  |
| #node.session.auth.password = password | 设置target端验证的密码 |  |
| 超时值(Timeouts) | node.session.timeo.replacement\_timeout = 120 | session 重建的超时值 | 默认值为120S，如果设为0，IO将立即返回，如果小于0：IO will remain queued until the session is logged back in, or until the user runs the logout command |
| node.conn[0].timeo.login\_timeout = 15 | 设置login完成的等待时间 | Open-iscsi的login是异步的。 |
| node.conn[0].timeo.logout\_timeout = 15 | 设置logout完成的等待时间 |  |
| session 设设备的队列深度 | .  node.session.cmds\_max = 128 | Session中排队的命令的最大个数 | 默认值为128，必须是介于2-2048的，且是2的幂次方。 |
| node.session.queue\_depth = 32 | 设备的队列深度 |  |
|  | node.session.xmit\_thread\_priority = -20 | 设置线程优先级 | 对于内核模块iscsi (iscsi\_tcp) and iser (ib\_iser)，每一个session都有一个线程用来排队数据到硬件。  设置的优先级越低，吞吐量越高，延迟越低。最低值为-20，设置一个较大的值会导致IO性能降低，但是不至于过多的占用CPU。  这个值介于[-20 ,20]默认为-20 |
|  | node.session.iscsi.ImmediateData = Yes | 是否开启立即数据 |  |

# 参考文献

<http://www.ietf.org/rfc/rfc3720.txt>

RFC3720,iscsi协议最权威的指南。

<http://www.open-iscsi.org/>

open-iscsi协议的官方网站。

<http://www.open-iscsi.org/docs/README>

open-iscsi协议的README。

<http://linux-iscsi.org/wiki/Lio-utils>

lio-utils的官方网站，tcm\_node及lio\_node的使用主要来自于这里。

<http://hi.baidu.com/wangpeng1314/item/839823f7211fd0b730c19919>

<http://publib.boulder.ibm.com/infocenter/ibmxiv/r2/index.jsp?topic=%2Fcom.ibm.help.xivgen3.doc%2Fxiv_apiiscsitargetportsobjectmodel.html>

<http://hi.baidu.com/wangpeng1314/item/839823f7211fd0b730c19919>

上面是iscsi协议的另外一些文章。

<http://www.linuxidc.com/Linux/2012-05/59674.htm>

<http://blog.csdn.net/gobitan/article/details/5903342>

上面是介绍linux service服务的文章

<http://zhumeng8337797.blog.163.com/blog/static/100768914201191810145564/>

<http://baike.baidu.cn/view/541220.htm>

<http://searchstorage.techtarget.com/definition/logical-unit-number>

<http://en.wikipedia.org/wiki/Logical_unit_number>

<http://www.zhixing123.cn/ubuntu/24795.html>

<http://tldp.org/HOWTO/SCSI-2.4-HOWTO/scsiaddr.html>

上面的文章介绍LUN及SCSI设备地址的层次结构