Feature-based Comparison of iSCSI Target Implementations

Nitin Gode
MIT College of Engineering,
Survey No. 124, Paud Road,
Pune 411038, India

Rhushida Kashalkar
MIT College of Engineering,
Survey No. 124, Paud Road,
Pune 411038, India

Deepika Kale
MIT College of Engineering,
Survey No. 124, Paud Road,
Pune 411038, India

Sukhada Bhingarkar
MIT College of Engineering,
Survey No. 124, Paud Road,
Pune 411038, India

ABSTRACT
Small Computer System Interface (SCSI) is a set of protocols used for communication and data transfer between I/O devices like computers and storage devices. SCSI standards include various commands, interfaces and protocols that are required to communicate with devices such as hard disks, DVD and printers. iSCSI is a storage standard for exposing and utilizing data storage devices over network using TCP/IP protocol. iSCSI provides remote data operations by performing SCSI commands over TCP/IP networks. This paper explains the features of various iSCSI Target Frameworks currently in use, such as SCSI Target Framework (STGT), Generic SCSI Target Subsystems for Linux (SCST), Linux I/O Target (LIO), iSCSI Enterprise Target (IET). The paper also presents a comparison of the features of these implementations.

Keywords
Storage/Repositories, Distributed File Systems

1. INTRODUCTION
Small Computer System Interface (SCSI) is a set of protocols used for communication and data transfer between I/O devices like computers and storage devices. SCSI standards include various commands, interfaces and protocols that are required to communicate with devices, mostly hard disks. SCSI is a peer-to-peer interface capable of handling 16 devices on a single bus. iSCSI is an IP based storage networking standard for linking data storage facilities. iSCSI provides location independent data storage and retrieval by carrying out standard Small Computer System Interface (SCSI) commands over IP networks. Two iSCSI hosts can thus be connected over an IP network. SCSI commands are exchanged over this network for exchanging of data. These hosts are classified as initiators and targets. An iSCSI target functions as the endpoint that waits for and services requests from a client machine (initiator). It may be a network-connected storage device, array, or an emulated target. [1, 2, 3]

Section 2 presents several works related to the paper. Section 3 contains a detailed study of the target implementations covered in the paper, namely iSCSI Target Framework (STGT), Generic SCSI Target Subsystems for Linux (SCST), Linux I/O Target (LIO), iSCSI Enterprise Target (IET). Section 4 presents a comparison table of the iSCSI Target implementations and its analysis. Section 5 presents the conclusion and future scope for the paper.

2. RELATED WORK
Related papers in which the performance of iSCSI target software is discussed include [4], [5], [6], [7] and [8]. The performance of the Ardis target framework (on which IET is based) with several legacy target frameworks not covered by this paper, including variants of the UNH target and the Ardis target is compared in [8]. In [4], Y. Lu et al compare the performance between iSCSI target and a NAS scheme. The overheads that the iSCSI subsystem introduces as compared to direct file storage systems are discussed in [5]. The performance of iSCSI protocol in various scenarios, such as in a WAN and when having virtualized disks is evaluated in [6]. Khosravi et al discuss performance of iSCSI in server environments in [7].

While there exist a number of papers that discuss the performance of iSCSI Targets, none discuss the feature set of the current target frameworks.

3. CURRENTLY AVAILABLE TARGET FRAMEWORKS
This section explores the features of several popular iSCSI target frameworks, including STGT, SCST, LIO (TCM) and IET. The following section provides the comparison between the features of these frameworks.

3.1 STGT
STGT (SCSI Target Framework) was the standard multiprotocol SCSI target in Linux. It was an alternative implementation of SCSI target framework for Linux [9, 10]. The main purpose of STGT was to simplify SCSI Target creation. In STGT, the target was implemented in userspace rather than the kernelspace. STGT was merged in earlier versions of Linux kernel as it was considered to be correctly implemented. However, STGT was replaced by LIO (from Linux kernel 2.6.38) as LIO had several advantages over STGT. STGT's goal was to integrate into SCSI layers in user space rather than kernel space. Bidirectional commands could be used in STGT. However it had several performance and complexity problems, making it unsuitable for use in a production environment.

3.2 SCST
The Generic SCSI Target Subsystem for Linux (SCST) allows creation of sophisticated storage devices from any Linux device. [10] SCST devices provide extended functionality like replication, device add/remove notification, thin provisioning, support for AEN, etc. Links that support SCSI commands like iSCSI, FCoE, SAS, Wide SCSI, etc. can be efficiently used by SCST implementation.

SCST project consists of a set of subprojects: generic SCSI target mid-layer itself (SCST core) with a set of device handlers as well as target drivers and user space utilities. [10] For communication between the target drivers and the kernel, SCST implements interfaces. These interfaces are used for connecting the backend handlers with the target drivers. The SCST core processes any requests received by the target and checks them

for errors. It also solves most execution problems, thus making implementation and functioning easier in the kernel.

SCST core emulates necessary functionality of SCSI host adapter, as from the perspective of a remote initiator, a SCSI target acts as a SCSI host with its own devices. [10] This is important in cases where multiple initiators are connected. Particularly, incoming requests can be processed in the caller's context or in one of the internal SCST core's tasklets without any extra context switches. [10] For example, pass-through device handlers allow exporting real SCSI hardware and vdisk device handler allows exporting files as virtual disks. [10]

3.3 LIO

LIO target is an open-source implementation of an iSCSI target framework which is included in the upstream Linux kernel (since Linux 2.6.38). It supports a large number of fabrics, including FCoE, Fiber Channel, iSCSI and vHost. Its advanced feature set has made it possible for it to achieve VMware vSphere 4 Ready Certification and vSphere 5 Ready Certification [11].

The target engine is based on a high performance SCSI engine which implements the semantics of SCSI Architecture Model 2 (SAM 2) and the SCSI Primary Commands specification (SPC-3 and SPC-4) [11]. It supports Memory Copy RAMDISK, which provides comprehensive SCSI emulation and separate memory mapping per initiator [11]. It also supports specialized functionality, such as MC/S (Multiple Connections/Session), Asymmetric Logical Unit Assignment (ALUA) and Error Recovery Levels (ERL 0, 1, 2).

LIO supports a large number of backstores, including FILEIO (buffered and non-buffered), IBLOCK (any block device), Pass-through SCSI (PSCSI) devices and Ramdisk. It has in-built support for virtualization, including native support for libvirt, OpenStack (beginning with Release 2013.1) and KVM.

LIO has a fully kernel-based architecture, with several user-space configuration tools including IOCTLs, ProcFS, ConfigFS, rtslib (Python based API), targetcli (command line interface) and configshell.

3.4 IET

iSCSI Enterprise Target (IET) is a kernel based target driver based on Ardis target implementation created by Ardis Technology. It was created with the goal of creating an enterprise ready iSCSI target which is scalable enough and versatile enough to meet the rapidly changing requirements of SAN storage technology. It includes several features not present in the original Ardis implementation, such as SMP support, Linux 2.6 support, dynamic configuration and iSNS support.[11]

IET consists of a kernel-based target driver and user-space configuration tools. The kernel driver can be configured using IOCTLs, Netlink and ProcFS interfaces [12]. IET includes a configuration tool ietadm, which allows dynamic configuration. The basic configuration of IET is stored in /etc/ietd.conf

IET supports various features over Ardis implementation. Dynamic addition and deletion of targets, volumes and authentication accounts is supported. It supports FILEIO and BLOCKIO backstores, with zero copy read/write support for BLOCKIO devices. It supports failover clusters, and Multipath IO (MPIO), a method by which data can take multiple redundant paths between server and storage. It also supports SCSI-2 RESERVE/RELEASE and SCSI-3 PR [11]. It also supports I/O context grouping between I/O threads, which can improve performance to a large scale in kernels using CFQ I/O Scheduler. Similarly it supports iSCSI redirects and Internet Storage Name Service (iSNS) protocol.

IET however does not support any transport fabrics other than iSCSI. IET has an unsafe implementation of Task Management Commands – it processes ORDERED commands in the same way as SIMPLE commands, which may lead to data corruption. It has been replaced by other target implementations in several popular Linux distributions, because of lack of support and of modern feature.

4. COMPARISON OF LINUX SCSI TARGETS

From the study of the target, it is possible to draw several conclusions.

Of the above discussed frameworks, LIO requires minimum effort to use, as it is already merged with the mainstream kernel. All implementations have a generic target engine, except IET, which has support only for iSCSI. LIO and SCST have kernel-space architecture, while IET has a split architecture (data transfer is kernel-space, while management is user-space) and STGT has completely user-space architecture.

In the target frameworks, both LIO and SCST support target drivers in user-space as well as kernel space (SCST via scst_local, LIO via tcm_loop), whereas IET does not. In addition, SCST and LIO support kernel-space backstore handlers. LIO also supports Memory Copy RAMDISK.

SCST in particular provides an extended set of functionality not supported by any other initiators, including automatic session reassignments (changes in access control immediately seen by all initiators), support for AENs (asynchronous event notifications) and notifications for device add/remove/resize (through AENs/Unit Attentions). It also supports bidirectional SCSI commands. All the target frameworks satisfy SCSI safe RESERVE/RELEASE requirements; though only SCST has a fully safe implementation of task management commands (LIO has safe implementation only for LUN RESET).

With respect to the iSCSI target, most implementations support only zero-copy data send. SCST does not support MC/S (supported by LIO and IET), a feature which allows several connections per session, making failover recovery faster. However, SCST allows a more fine-grained control over visibility of targets, such as per-portal and per-initiator target visibility control. In addition, both SCST and LIO support use of hardware instructions for digest calculations.

Table 1 summarizes the comparison of the iSCSI Target Implementations.

Table 1 Feature based comparison of iSCSI Target Implementations [10]

	SCST	STGT	IET	LIO/TCM
Generic Target Engine	Yes	Yes	iSCSI only	Yes
Architecture	Kernel only	User space only	Split	Kernel only
Target drivers in kernel space	Yes	No	No	Yes
Backstorage handlers in kernel space	Yes	No	No	Yes
Backstorage handlers in user space	Yes	Yes	No	No
Automatic sessions reassignment	Yes	No	No	No
Support for Asynchronous Event Notifications (AEN)	Yes	No	No	No
Notifications for devices added/removed or resized through AENs or Unit Attentions	Yes	No	No	No
Bidirectional Commands	Yes	Yes	No	Yes
Extended CDB (size >16 bytes)	Yes	Yes	No	Yes
According to SCSI requirements safe RESERVE/RELEASE implementation	Safe	Safe	Safe	Not safe
Implementation of Task Management commands	Safe	Not safe	Not safe	LUN RESET - safe. Other TM commands not implemented.
Supported Transport and Hardware	iSCSI, SRP	iSCSI, iSER	iSCSI	iSCSI, SRP (Preliminary), iSER (Preliminary)
Supported Backstore	FILEIO (Kernel and User), BLOCKIO	FILEIO (User)	FILEIO (Kernel), BLOCKIO	FILEIO (Kernel), BLOCKIO, Ramdisk
Interface with user space	IOCTL/Netlink/SysFS (or obsolete ProcFS)	Not Applicable	IOCTL/ProcFS/ Netlink	IOCTL/ProcFS/ConfigFS
Zero-copy data send/receive	Send only	In some cases, send only	Send only	Send only
Multiple connections per session (MS/C)	No	No	Yes	Yes
Max Error Recovery Level (ERL)	0	0	0	2
Support for limiting number of initiators allowed to connect to a target	Yes	No	Yes	No
Per-portal targets visibility control	Yes	No	Yes	No
Per-initiators targets visibility control	Yes	Yes	Yes	No
Support for AHS	Yes	Yes	No	No
Support for iSCSI redirects	Yes	Yes	Yes	No
Support for iSNS	Yes	Yes	Yes	No
Implementation of connections and sessions reinstatement	Safe	Not safe	Not safe	Not safe
For digest calculations, Usage of hardware instructions	Yes	No	No	Yes

5. CONCLUSION AND FUTURE SCOPE

This paper has described the functionality of the various Linux iSCSI Targets available currently and their comparison with respect to various functionality that they offer. From this it can be concluded that LIO requires least efforts to use in a server environment, as it is already present in the Linux kernel. LIO and SCST support target drivers in both user space and kernel space, while IET does not. Both SCST and LIO offer a varied and greater functionality as compared to IET and STGT, and may be considered as mature enterprise-ready target implementations.

The scope of this paper is limited to the functionality of the targets. In future works, this scope may be extended to include several other parameters, such as performance, specifically performance of IO operations considering the type of backstores used, network delays introduced (if any) and difference in execution time of IO operations and an analysis of the overheads introduced by each framework.

6. ACKNOWLEDGEMENTS

We would like to thank our mentors, Vishal Tripathi (Calsoft Inc.), Sumeet Gandhare (Calsoft Inc.), Zubraj Singha (Calsoft Inc.) for their help and guidance in our project and the subject.

7. REFERENCES

[1] RFC 3720 - Internet Small Computer Systems Interface http://www.ietf.org/rfc/rfc3720.txt

[2] iSCSI: The Universal Storage Connection – John L. Hufferd ISBN: 978-0201784190

[3] IP SANS: A Guide to iSCSI, iFCP, and FCIP Protocols for Storage Area Networks – Tom Clark ISBN: 978-0201752779

[4] Yingpin Lu. "Performance study of iSCSI-based storage subsystems." *Communications Magazine, IEEE Volume:41 , Issue: 8*

[5] Bianco, A., Dip. di Elettron. et al. "Distributed storage on networks of Linux PCs using the iSCSI protocol." *High Performance Switching and Routing, 2008. HSPR 2008. International Conference*

[6] Xinidis D., Bilas A. et al, "Performance evaluation of commodity iSCSI-based storage systems." *Mass Storage Systems and Technologies, 2005. Proceedings of 22nd IEEE / 13th NASA Goddard Conference*

[7] Khosravi, H.M., Abhijeet Joglekar, Iyer, R. "Performance characterization of iSCSI processing in a server platform." *Performance, Computing, and Communications Conference, 2005, IPCCC 2005. 24th IEEE International*

[8] Fujita Tomonori, Ogawara Masanori. "Analysis of iSCSI target software." Proc. of. *SNAPI '04 Proceedings of the international workshop on Storage network architecture and parallel I/Os*

[9] Linux SCSI target framework (tgt) project stgt.sourceforge.net/

[10] SCST: A Generic SCSI Target Subsystem for Linux scst.sourceforge.net/

[11] Linux SCSI Target, http://linux-iscsi.org/

[12] Debian Wiki - iscsitarget, https://wiki.debian.org/SAN/iSCSI/iscsitarget.