Design and Implementation of a Block Storage Multi-Protocol Converter

Irina Gerasimov, Alexey Zhuravlev, Mikhail Pershin, and Dennis V. Gerasimov
TechnoMages, Inc
Baltimore, MD, USA
{ira, alexey, misha, dennis}@technomagesinc.com

Abstract

We present the Block Storage Multi-Protocol Converter (BSMC) software architecture, which is able to translate and manage SCSI commands carried by different SCSI transport protocols, such as SCSI, Fibre Channel, and iSCSI, to a variety of storage devices. We discuss the internal organization of BSMC and present some performance testing results. We describe the variety of applications in Storage Area Networking that can benefit from products based on the BSMC architecture, such as a multi-protocol storage router and a multi-protocol, multi-RAID-level disk array.

1. Introduction

This paper presents the Block Storage Multi-Protocol Converter (*BSMC)* software architecture, which is able to translate and manage SCSI commands carried by different SCSI transport protocols, such as SCSI, Fibre Channel, and iSCSI among many Hosts (Initiators) and Devices (Targets).

The SCSI Command Set has become the predominant protocol for disk and tape I/O and device management. Although the SCSI Parallel Bus is a mature and well-established transport interface, providing fast access to storage devices, it has limitations such as distance (several meters) and scalability (15 devices on a bus). Fibre Channel (FC), with its mapping of the SCSI Command Set (as the "Fibre Channel Protocol," or FCP), has overcome some of the distance and address-space drawbacks of SCSI, and is currently the dominant transport protocol for Storage Area Networks (SAN). Even so, Fibre Channel has its own shortcomings: a relative lack of access-control security, and limitations in long-distance capabilities (<500m regular, <10km with special and costly equipment).

Among emerging SCSI transport protocols, iSCSI seems to be a very promising mechanism for augmenting Parallel SCSI and Fibre Channel. In iSCSI, SCSI commands and data blocks are encapsulated into TCP/IP protocol messages. Currently, version 1.0 of the iSCSI specification is in the final stages of approval by the Internet Engineering Task Force, [1]. The iSCSI protocol uses TCP flow control, congestion control, segmentation mechanisms, and IP addressing and discovery mechanisms. Thus iSCSI allows remote access to SAN storage devices, for backup, mirroring, and disaster recovery, among other possible applications.

The *BSMC* architecture was designed from the very beginning as a basis for various storage management products, and has been used to create a multi-protocol storage router and a disk array subsystem. The *BSMC* architecture is organized into three-levels: *Front End*, *Generic* and *Back End* layers. Such a layered structure has several advantages. First, it makes it easy to add and upgrade the interfaces on the *Front End* layer without compromising backwards compatibility and data integrity. Second, it makes it possible to add data management features to the *Back End* layer by means of adding or replacing the modules responsible for handling storage devices.

The *BSMC* architecture, when applied to RAID Arrays, not only makes possible controlled access by multiple Initiators, but also lets several applications with different bandwidth or I/O-operations-rate requirements co-exist together using the same RAID subsystem. For example, several local mail servers and database servers can access a BSMC-based RAID via Fibre Channel or SCSI interfaces, while remote backup can be performed using an iSCSI interface.

Palekar [2] has built a software suite similar in function to the BSMC. Their Linux SCSI Target Emulator was able to process SCSI commands received from SEP (SCSI Encapsulation Protocol), iSCSI, and Fibre Channel initiators and handle those commands to corresponding Linux block device drivers. To our knowledge, no functional validation or performance evaluation was reported for that architecture, however.

We have evaluated the performance of the *BSMC* architecture in the case where it is used to build a multi-interface RAID subsystem. Our results for RAID performance show that, for its SCSI and FibreChannel interfaces, the BSMC delivers throughputs close to those reported by vendors of SCSI-attached and Fibre Channel-attached ATA-disk arrays. In addition, we present iSCSI performance data.

The paper is organized as follows. In the next section we discuss the *BSMC* architecture design. Then we describe our performance evaluation setup and Iometer benchmark results. After that we describe how the *BSMC* architecture was applied for creating two TechnoMages, Inc. product lines: the RAID subsystem, called *InfoSlice,* and versatile storage router, called *Data Transport Processor*. Finally, we offer some concluding remarks.

2. Design

In this section we describe the main components of the Block Storage Multi-protocol Converter Architecture. As can be seen from Figure 1, the architecture consists of three layers: *Front End*, *Generic* and *Back End* SCSI Target Drivers. These drivers are implemented as independent Linux kernel modules, which interact with each other using a common API. The arrows in Figure 1 show the interactions between the various parts of the *BSMC* architecture.

Front End layer. The *Front End* driver of the *BSMC* architecture is responsible for receiving and sending messages coming from and going to the SCSI transport protocol specific Host Bus Adaptors, or HBAs, which, in turn, transfer those messages to corresponding SCSI Initiators. For each type of SCSI transport protocol, SCSI, Fibre Channel, or iSCSI, *Front End* contains a corresponding Target driver, whose implementation is both protocol- and HBA-type specific. For example, FibreChannel and SCSI drivers are implemented as a single thread, while for the iSCSI driver there can be many threads created, depending on the number of connected initiators, as specified by iSCSI standard. Every Target driver contains a message queue, where it keeps received messages until the commands, which those messages contain, are completed. If a Target driver receives a message that is transport protocol specific but involves no data handling, it processes that message itself. Otherwise, the message is passed to the *Generic* SCSI Target driver using an API that is common for all *Front End* Target drivers.

Generic layer. The main functions of the Generic SCSI Target are to receive SCSI commands from *Front End* Target Drivers, place those commands into generic protocol-independent SCSI command structure, distribute the commands among a list of SCSI command queues according to the Access Control Filter, and, finally, call the *Back End* Target driver to notify it of a new command arrival. The *Generic* Target driver monitors the status of commands in the queues and upon command completion, error, or timeout, interacts with the *Front End* layer. If the command does not require specific *Back End* handling, the *Generic* Target completes it as well.

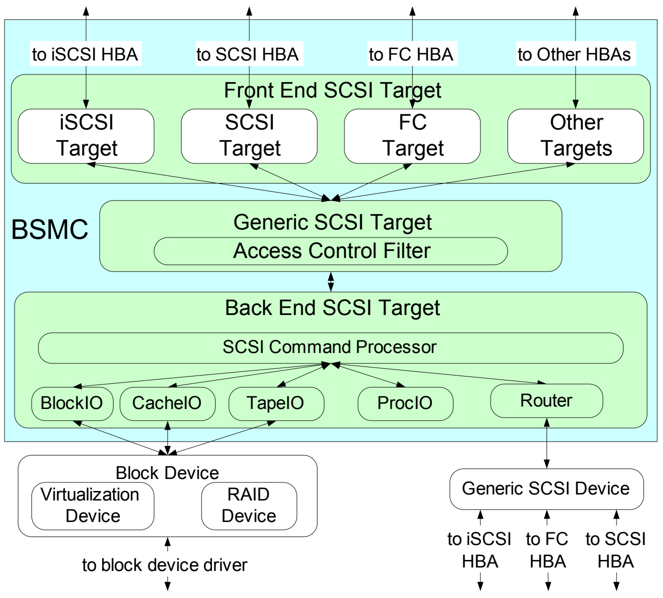


Figure 1. Block Storage Multi-Protocol Converter Architecture

By organizing SCSI commands into several queues, and by distributing the commands according to an Access Control Filter (*ACF*), the *BSMC* provides a powerful security mechanism to Administrators of storage initiators, devices and device partitions. This mechanism works as following. For each initiator, device, or device partition, a separate data structure, called *_LUN* is created. Each *_LUN* structure contains a queue with SCSI messages, which are intended for the initiator, device, or partition, corresponding to that *_LUN*. Further, in order to control access of Initiators to *_LUN* command queues there is *ACF*, which keeps a set of definitions of Initiators that can access each *_LUN* queue. Once a SCSI command is received, its Initiator is checked with Initiator definitions for the *_LUN* queue of the device the command is intended for. There are several types of Initiator definitions: SCSI transport protocol, HBA channel number, and storage protocol specific types such as SCSI ID (for SCSI), FibreChannel WWN (for Fibre Channel) and IP address or authenticated user (for iSCSI).

Back End layer. The *Back End* Target consists of SCSI Command Processor, which is responsible for a SCSI command execution and a number of SCSI device-specific *Back End* drivers, called *BlockIO*, *CacheIO*, *TapeIO*, *ProcIO*, and *Router*. When one of the *_LUN* SCSI command queues receives a new command, the SCSI Command processor calls the appropriate device-specific *Back End* driver to accomplish that command.

BlockIO and *CacheIO* are the modules that handle commands intended for any block device. These drivers interface with the Linux kernel block device layer. Upon receipt of a SCSI command, the *BlockIO* driver wraps that command into a structure recognizable by the *Block Device* and then hands that command to the corresponding device of the Linux kernel Block Device layer. In addition to functions performed by the *BlockIO* driver, the *CacheIO* driver contains various block caching and read-ahead optimizations, which dramatically improve *Block Device* read/write performance.

In order to implement the partitioning of locally and remotely attached RAID devices and independent disks, a new driver called *Virtualization Device*, or *VD*, was implemented as a modular component of the Linux kernel Block Device layer.

The *TapeIO* driver was designed for the creation of tape emulation devices. It is functionally similar to the *BlockIO* driver, but it converts stream-oriented SCSI commands intended for tape drives into commands for block devices, which are handled by the Linux kernel Block Device layer. This driver allows the creation of disk-based backup devices that appear to tape-oriented host software as tape devices.

The *ProcIO* driver handles commands primarily used to manage the enclosure via SCSI Enclosure Services (SES) or SCSI Controller Commands (SCC), report the status of system components, and similar tasks. This driver is also used for an implementation of the SCSI Extended Copy command [9].

The *Router* driver handles SCSI commands that are passed directly between a SCSI target device and an Initiator. The *Router* driver interfaces with the Linux mid-level generic SCSI driver. The *Router* driver passes SCSI commands received from the *Generic* Target layer to the mid-level SCSI driver, which processes a command and sends it further to the corresponding SCSI-Target device.

3. Performance Evaluation

To demonstrate the viability of the *BSMC* architecture we have evaluated the performance of an integrated RAID system based on the *BSMC*. This integrated system is utilizing the Linux RAID implementation as the underlying block device. The data path through *BSMC* consists of the three *Front End* Target drivers, iSCSI, SCSI, and FC, the *Generic* SCSI Target, and two *Back End* Targets: *BlockIO* and *CacheIO*. The results of the evaluation can be compared with the theoretical maximum performance achievable over any given interface, and with various performance results available from many RAID vendors.

3.1. Testing setup

The Initiator machine was built with a Tyan S2510 motherboard (64bit/66MHz PCI bus), one Intel Pentium III 1GHz processor, and 512MB RAM. For multi-protocol testing we used an Alteon AceNIC 1000BaseT Ethernet NIC for iSCSI, an Adaptec 29160 HBA for SCSI, and a Qlogic QLA2300F HBA for Fibre Channel. The Initiator machine was running Windows 2000 Server and the IBM iSCSI driver 1.2.2 [3]. In case if Linux OS is used on the client, the open source iSCSI driver is available [4].

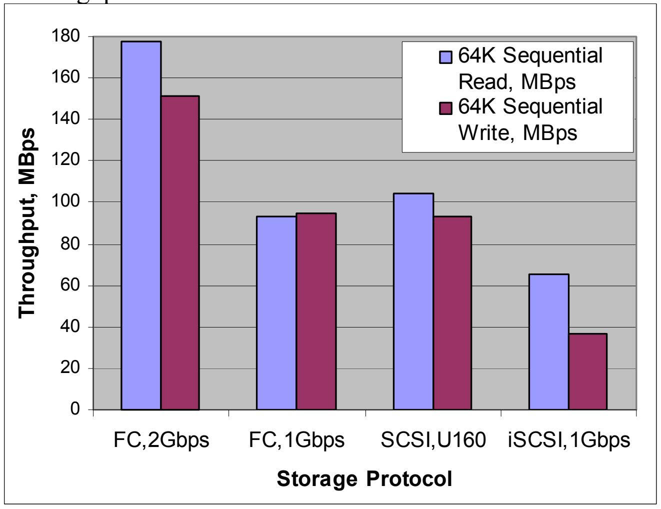
The disk array machine was built with a Tyan S2720 motherboard with one Intel 1.8 GHz Xeon processor, 512MB RAM of registered ECC type, and a 64bit/66MHz PCI bus. For multi-protocol testing we used an Alteon AceNIC 1000BaseT Ethernet NIC for iSCSI, a LSI53C1010-66 HBA for SCSI, and Qlogic QLA2200F/66 (1Gbps) and QLA2300F (2Gbps) HBAs for Fibre Channel. The disk array contained 16 Western Digital WD2000JB ATA 7,200 rpm hard disks connected to a pair of 3Ware 7810 Escalade disk controllers configured as JBOD. For this testing, the disks were combined into RAID0 disk set via the Linux RAID implementation. The disk array machine was running the TechnoMages proprietary embedded Linux distribution (2.4.19 kernel). This distribution is a minimal set of utilities primarily based on BusyBox [5], which fits into 16MB flash card.

3.2. Iometer results

We have evaluated the performance of the above-described RAID sub-system by testing it with the Iometer [6] benchmark. Initially developed at Intel and now open-sourced, Iometer is a standard benchmark used for measuring server access I/O rates. Iometer measures the I/O performance of the system while stressing it with a controlled workload. Iometer can be configured to simulate the workload of different applications and benchmarks. Iometer gathers data such as throughput, latency, and CPU utilization.

For this work we report the 64K Data Sequential Read and Write throughput, which are presented on Plot 1. We have selected large block size streaming operations for this work because we are evaluating the performance of the BSMC architecture, not the seek performance or other parameters of the particular hard drives used in the test system. As can be seen on this plot, the performance of the InfoSlice RAID subsystem, measured for 1Gbps FibreChannel and 160Ultra SCSI interfaces, is close to 100MBytes per second. These numbers show that the InfoSlice RAID subsystem, while based on the complex *BSMC* architecture, is able to provide the same performance reported for “pure” FibreChannel or SCSI RAID sub-systems manufactured by other vendors.

The iSCSI throughput rates of the *BSMC*-based disk array are lower then those measured for SCSI and 1Gbps FibreChannel interfaces. This is explained, in part, by the overhead introduced by TCP/IP protocol. This overhead includes the checksum calculations for TCP and IP headers, the fragmentation and de-fragmentation of IP datagrams over Ethernet frames, and the acknowledgement messages for TCP segments require more CPU involvement and interface bandwidth, *currently*, than the Ultra160-SCSI and Fibre Channel counterparts, and thus reduce iSCSI's sustainable throughput rates more.



Plot 1. Streaming Read/Write performance measured by the Iometer benchmark

4. *BSMC* Applications

The *BSMC* is a very flexible architecture that can be applied to create various kinds of block command converters and management devices. The implementation of this architecture was used by our company, TechnoMages, Inc. to create two lines of integrated data storage products: the *InfoSlice* RAID subsystem and the *Data Transport Processor* SAN management platform.

The *InfoSlice* line is an integrated storage subsystem with all the storage management features provided by *BSMC* integrated in a turnkey solution. TechnoMages, Inc. currently offers *InfoSlice* arrays with either SCSI or ATA disk drives. The *BSMC* architecture enables any combination of SCSI, Fibre Channel or iSCSI interfaces to be included in an *InfoSlice* array. The ability of the *BSMC* to represent any block device as a SCSI LUN enables *InfoSlice* to have a very flexible set of storage management features, such as storage virtualization and access control.

Various RAID configurations can be created on the *InfoSlice*. These RAID volumes, in turn, can be partitioned into user-visible SCSI LUNs on the fly. These LUNs can then be dynamically resized, if needed. For example, all InfoSlice disk space can be configured as one large RAID 5 volume, which is then partitioned (via the previously discussed VD driver) into several partitions according to the needs of several projects. These partitions can then be exported as independent LUNs and used by respective projects as unrelated disks. If, at a later date, one project no longer needs the storage, this partition can be deleted, and storage space, which was previously used by it, can be reallocated to other projects by growing their LUNs.

Access to any *InfoSlice* LUN can be configured in a very flexible way based on the interface type and capabilities [7]. For example, on a Fibre Channel interface, any LUN can be made visible to either all attached initiators or just particular WWNs. Using the example of several projects sharing the *InfoSlice* storage space, one may want to set up access control in such a way that each of the projects server on a Fibre Channel network can only access his dedicated LUN, and cannot even see the other ones.

InfoSlice features several high availability features, such as hot-swappable power supplies, hot-swappable disks (yes, even ATA disks are hot-swappable), and hot-swappable cooling fans. Moreover, it is completely reconfigurable on the fly. However, the trademark of the InfoSlice product line is not the high-availability, but rather flexibility. The flexibility of the *InfoSlice* line defines type of user group, which get the most benefits from this product. The primary user group of *InfoSlices* might very well be the research community (whether government, university, or commercial), which can fully appreciate the unprecedented versatility of this device. Low cost per gigabyte, ability to upgrade disks and interfaces (which extends the initial investment in a *BSMC*-based appliance) coupled with the ability to reconfigure the storage in a multitude of ways makes it very attractive to rapidly changing scientific and engineering research environments.

Another application area where *InfoSlice* might provide significant benefits to the research community is data archival and secondary near-line storage. In the section 2 of this paper, we have discussed a "TapeIO" module of the *BSMC*, which makes it possible to represent a disk device as a tape device to the initiator. We call this function "*Tape Emulation*". *Tape Emulation* allows one to easily augment or replace an existing tape library with a faster disk device without any changes in the software that manages data archival. *Tape Emulation* can also be used to reduce the time it takes to make system backups.

The second product line created by TechnoMages, Inc. from the *BSMC* is the *Data Transport Processor* (*DTP*) line. This family of devices does not have any integrated storage; it is used to manage access to other storage devices. Our paper [8] describes various DTP applications.

One of the *DTP* models, called the *FibreFire*, is a Fibre Channel firewall. It is a filtering device with multiple Fibre Channel interfaces that can control traffic between two or more Fibre Channel networks. This device can selectively pass SCSI commands based on source and destination WWNs. This capability is particularly useful when connecting large SANs with complex switching structures, to prevent the connected SANs from affecting each other's switches, particularly zoning. Another application of the *FibreFire* is to interconnect between SANs which do not want to give full access to each other. It can provide respective SAN administrators a common checkpoint for access policies.

Another model of the *DTP* can interconnect multiple SCSI devices into a pseudo-SAN. Acting as a "SCSI switch", this model can be used to provide and manage direct access to SCSI devices, such as an automated tape library, to several computers, using only SCSI cabling and HBAs. The common backup configuration (Figure 2) involves passing all data over LAN from the backup client to the server, which has the tape library attached directly to it.

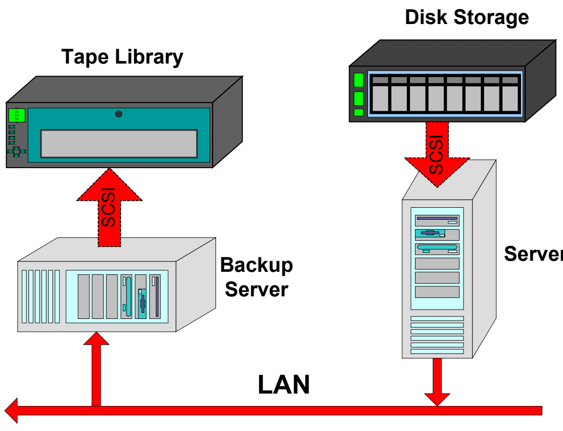


Figure 2. Backup over LAN

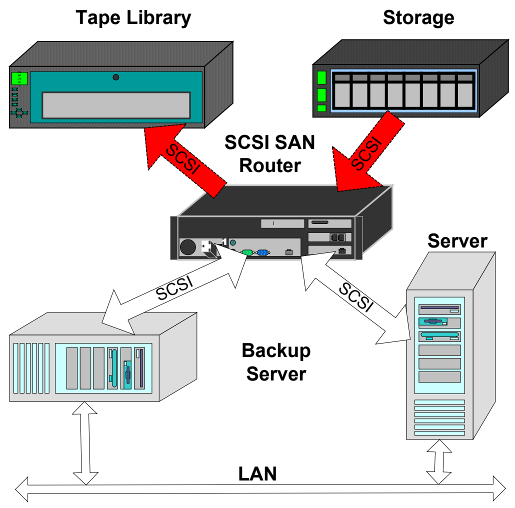


Figure 3. Backup over SCSI SAN

An alternative setup shown on Figure 3 includes DTP router, to which tape library and disk storage are connected. In this configuration the tape library can be used directly by the client for backups without passing traffic through a LAN. Using the SCSI Extended Copy command, implemented in the DTP, can further enhance this by moving data directly from the disk storage to tape devices without passing it through the backup server.

This setup, depending on the configuration, may provide for significantly faster backups than backups over LAN. Its speed is comparable to a Fibre Channel configuration, and may even exceed it for some configurations. At the same time, it is significantly less expensive than a Fibre Channel SAN. In this configuration, DTP may also perform storage virtualization and management functions for all or some of the attached disk storage.

The *Data Transport Processor* with iSCSI interfaces is useful for applications that need remote storage. Bridging distant SAN islands, remote mirroring, remote backup and remote SAN debugging are just a few examples. Even where directory- and file-serving protocols (e.g., NFS, SMB, CIFS) have already been deployed for long-distance data storage and retrieval, iSCSI might provide some benefits. In another paper [10], we reported iSCSI and NFS performance comparison, and found that iSCSI can provide multi-fold performance advantages over NFS in certain applications. Replacing NFS with iSCSI moves file system handling from the storage-server to the storage-client, which makes it possible to cache data in a more efficient way. The protocol overhead of iSCSI, although significant, is still much smaller then that of NFS, especially when accessing small files. Where NFS is only used as a convenient method of remote storage access, iSCSI is a valid replacement for NFS, and may be a preferred method (not only for performance reasons, but perhaps for data-confidentiality reasons as well).

5. Conclusions

In this work we have presented the Block Storage Multi-Protocol Converter (*BSMC*), a novel modular SCSI Target architecture capable of handling SCSI commands carried by different SCSI transport protocols: Parallel SCSI, Fibre Channel and iSCSI/Ethernet. The three-layered modular nature of the architectural design enables the easy addition of such advanced features as device virtualization and access control. At the same time, the *BSMC* architecture is very conservative in its usage of the resources and performs quite well.

The *BSMC* architecture naturally lends itself as a basis for the creation of several data storage products. In conjunction with Linux RAID code and a collection of hard drives in one integrated unit, it makes a very versatile and adaptable RAID storage subsystem. When integrated with a collection of interface adapters it can serve as a SCSI/SAN router providing a rich set of storage management features in addition to protocol conversion.

References

[1] Internet Engineering Task Force, http://www.ietf.org/html.charters/ips-charter.html

[2] Palekar, A.; Ganapathy N., Chadda, A.; Russel, R. D.; "Design and Implementation of a Linux SCSI Target for Storage Area Networks," 5th Annual Linux Showcase and Conference, Oakland, CA, November 5-10, 2001.

[3] IBM iSCSI Initiator drivers, http://www-1.ibm.com/support

[4] Open Source iSCSI Initiator driver for Linux, http://www.sourceforge.net/projects/linux-iscsi/

[5] BusyBox software. http://www.busybox.net/

[6] see http://sourceforge.net/projects/iometer/

[7] "InfoSlice Security and Access Control overview", TechnoMages, Inc. white paper, http://www.technomagesinc.com/papers/IS_ACL_config.htm

[8] "Data Transport Processor applications," TechnoMages, Inc., http://www.technomagesinc.com/papers/DTP_Apps.html

[9] SCSI-3 Primary Commands (SPC-2), Section 7.2: Extended Copy Command. ftp://ftp.t10.org/t10/drafts/spc2/spc2r20.pdf

[10] "Performance Comparison of iSCSI and NFS IP Storage protocols," TechnoMages, Inc. white paper, http://www.technomagesinc.com/papers/ip_paper.htm