A Packet Forwarding Method for the ISCSI Virtualization Switch

Yi-Cheng Chung[a], Stanley Lee[b]
Network & Communications Technology, Research & Service Center
Industrial Technology Research Institute
Tainan, Taiwan, R.O.C.
carbooky@itri.org.tw[a], stanleylee@itri.org.tw[b]

Chung-Ho Chen
Department of Electrical Engineering and Institute of Computer and Communication Engineering
National Cheng-Kung University
Taiwan, R.O.C.
chchen@mail.ncku.edu.tw

Abstract

We present a packet forwarding method for the implementation of the iSCSI virtualization switch used in the SANs. Using the existing virtualization software plus network protocol stacks, first we build up a network-based storage virtualization switch model in the architecture of the symmetric approach. In compliance with the same functionality, our method can reduce the overheads of protocol processing by using a packet forwarding method based on caching the structure ID of the iSCSI session. With the Address Translator and Duplicating Handler, our design can also achieve the feature of storage virtualization management. From the result of our performance experiment, our forwarding system can achieve a higher performance on the READ/WRITE operations by 25%~30% and reduce the CPU utilization by 10%~15% compared to the conventional virtualization switch.

1. Introduction

There is a tendency towards networking storage around the developing network. Storage Area Network(SAN)[1-2] is a topology to attach remote storage devices such as disk arrays or tape libraries to servers in a way that serves the clients as local attached device. SAN is primarily used in large scale, high performance enterprise storage operations via Fiber Channel connecting, but the equipments to setup the SAN is relatively expensive. The emerging internet SCSI (iSCSI) SAN technology is expected to produce cheaper SANs. ISCSI [3-6] is a newly developed network protocol that specifies the access to the SCSI storages over the TCP/IP network. Any user with an iSCSI initiator program installed on its local computer can access the iSCSI SAN by using the original Network Interface Card (NIC). In the iSCSI SAN, the iSCSI connections bring together all the individual storage devices scattered over the IP network as a single SAN. For the purpose of management, several implementations, such as XIOtech [7], IBM [8], EMC [9], etc., are made on the storage virtualization management. For the advantage of scalability and facility by virtualization, manager prefers to provide user the logical capacity rather than the physical one. Figure 1 shows the system combining the iSCSI and virtualization technologies.

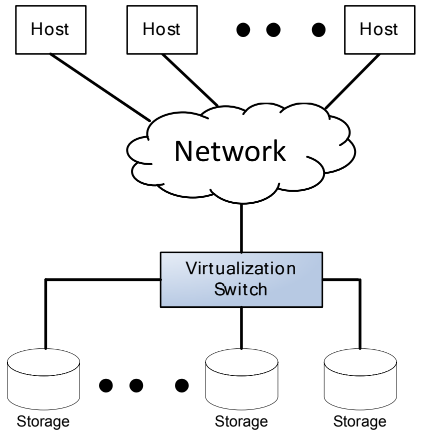


Figure 1. System of a SAN with virtualization switch

Our purpose is to build up a virtualization switch in the iSCSI SANs and manage the physical storages to provide a new storage device for the hosts. Due to the requirement of the additional software for the host by the asymmetric approach [10], we prefer the architecture of the symmetric approach to implement our virtualization switch. The symmetric approach, also known as In-band management, virtualization devices actually sit in the data path between the host and storage. In [11], it proposes a method to build a switch and route subsystem to dispatch iSCSI PDUs to the processing end nodes, but each host and storage has to apply a new interface before it can join the subsystem. In [12], the work has proposed a method to extract the iSCSI PDU header from the TCP buffer. Recently the Microsoft [13], Intel [14] and the University of New Hampshire [15] have developed their own iSCSI implementations, and it is common that the Logical Volume Manager [16] is used for storage virtualization management. We realized a conventional implementation of iSCSI virtualization switch. Compared to the conventional implementation, we proposed a new packet forwarding virtualization switch to have the same functionality while transmitting the iSCSI PDUs more efficiently. The rest of this paper is organized as follows.

Section 2 presents a conventional implementation of iSCSI virtualization switch system. Section 3 presents the architecture of the packet forwarding virtualization switch system. Section 4 discusses the verification and performance of the design. This paper is concluded in Section 5.

2. A conventional implementation of iSCSI virtualization switch system

ISCSI is a network protocol that brings SCSI commands over the IP network. Any two terminals on the IP network can be initiator/target to each other by installing the appropriate iSCSI modules. We attempt to build up a storage server that plays the role of iSCSI initiator to take those iSCSI targets as local storage devices. At the same time, the storage server plays the role of iSCSI target to provide the service of file sharing for the hosts who have installed the iSCSI initiator. For the purpose of storage virtualization, a virtualization application is involved to help abstracting the physical location of the data; meanwhile, it presents to the hosts a logical space for data storage. The virtualization system that we describe is shown in Figure 2.

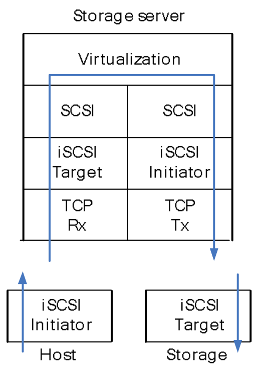
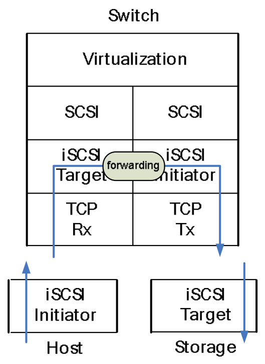


Figure 2. Packet flows on a conventional virtualization system

Each time when a packet of iSCSI protocol has reached the virtualization system, several processing of protocols, including TCP, iSCSI, and SCSI, have to be done before the true data can be extracted. After the converting of logical locations to physical ones, the virtualization system has to find a way to reach the location of physical space. Consequently, the data (or command of data requisition) has to be transformed into the type of SCSI command with payload (if any), and next a new iSCSI header is generated and appended to the whole data to become a new iSCSI Protocol Data Unit (PDU). Finally, the iSCSI PDU is transmitted to the storage with the interface of iSCSI target over the TCP/IP protocol. The response of the storage will go the same way back to the storage server, and further back to the host. It seems that there are many unnecessary processes along the packet flows from beginning to end. We propose a packet fast forwarding method to append to the conventional virtualization system, called fast path, and we regard the packet flow on the conventional virtualization system as a slow path.

3. Architecture of the packet forwarding virtualization switch system

Figure 3 shows the packet flows on the virtualization switch with packet forwarding mechanism. The forwarding mechanism starts to handle the incoming iSCSI PDUs as long as the iSCSI session reaches the status of full feature phase (FFP). On the virtualization switch, those iSCSI PDUs that follow the last successful login PDU are transferred within the iSCSI layer without the process of protocols above the iSCSI layer.



3. Packet flows on the virtualization switch with packet forwarding mechanism

3.1 Overview of the Forwarding System

Figure 4 shows the software architecture of the packet forwarding system. The system works in the middle of the two sides of TCP connections from host and storage. ISCSI Protocol Data Units (PDUs) are transferred directly between a pair of two connections without any additional process of header decomposing or rebuilding. Also, the payloads of the iSCSI PDUs are transferred under the TCP layer, no extra operation of data copy is required. The system includes several functional blocks: the Header Extractor, the Dispatcher, the Address Translator (AT), the PDUs Duplicating Handler (PDH), the Data mover, and the Header adaptor. The Host iSCSI functional block represents the conventional functionality of iSCSI virtualization system which is discussed in section 2.

3.2 The Header Extractor

The Header Extractor is the first functional block in the forwarding system. By extracting the size of the iSCSI header only, the duty of the Header Extractor is to generate a descriptor about the incoming iSCSI PDU including the fields of a pointer to the header, the connection ID, and the number of virtual LUN, and then, the descriptor will be pushed into a descriptor queue. Our implementation of the Header Extractor is actually a thread running on the operating system. The thread is spawned at the time when an iSCSI session has reached the FFP. The numbers of how may threads depend on the connections which come from the hosts.

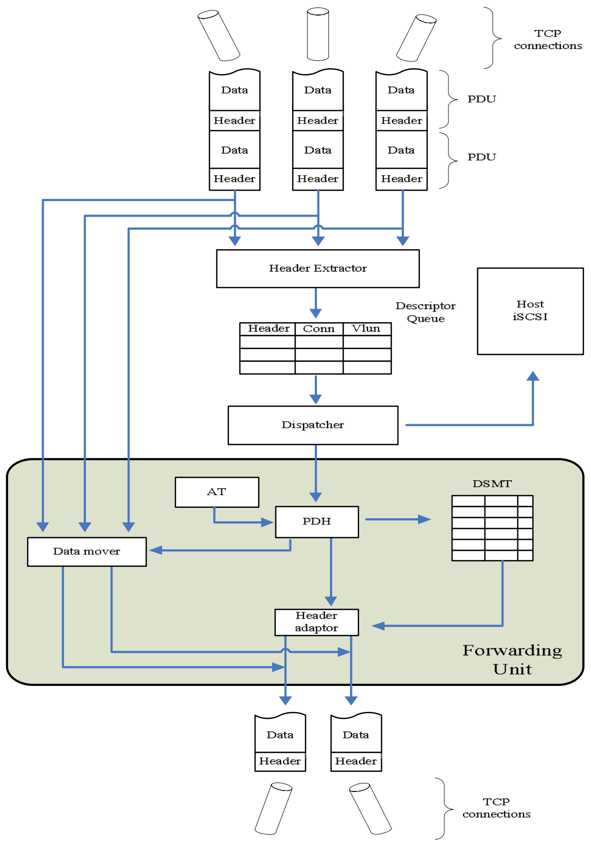


Figure 4. Architecture of the packet forwarding system

3.3 The Dispatcher

Figure 5 shows the design of the Dispatcher. The purpose of the Dispatcher design is to classify those packets according to the type of SCSI commands. The types of common READ/WRITE SCSI commands are passed to the Forwarding Unit, while the others are sent back to the slow path. The purpose of the descriptor queue design is to store the descriptors that may come concurrently from two different threads of Header Extractor.

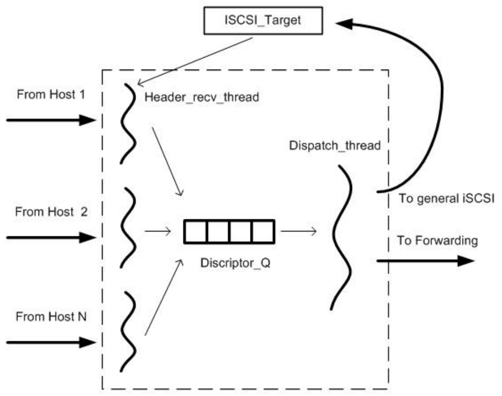


Figure 5. The software design of the Dispatcher

3.4 The Address Translator

The Address Translator (AT) plays the role of converting the logical locations to physical ones. The functionality of the AT is the same as the virtualization software does, the difference is that it has not only the simplest feature that converts the access address from logical to physical, but also it can be run independently without any other configuration tools.

3.5 The PDUs Duplicating Handler

The functional block of PDUs Duplicating Handler (PDH) has two major jobs. One job is to build the relationship between the two sides of host and storage. The second one is to deal with the problem whenever the data of a single request is spread over two more physical locations. The two main functions are explained below:

(1) The forwarding unit has the capability of transferring the iSCSI PDUs between two certain iSCSI sessions. The first iSCSI command request of a READ/WRITE transaction must be issued from the host side, and the data response to this command comes from the storage side. During the communication of the entire transaction, the rule of packet transferring must be held by an entry that is pending in the Dynamic Session Mapping Table (DSMT). Figure 6 shows the fields of the mapping entry in the DSMT. Each entry in the DSMT stands for a task that is proceeding between the host and storage. The entry contains the session ID and ITT number of both sides, and an additional field is added with the v-bit set to represent the mapping relationship as one-to-one mapping or not. The

forwarding unit may encounter a situation that the host issues a single access of data that the required data is spread over two different physical regions. The second main function of the PDH is going to solve this problem.

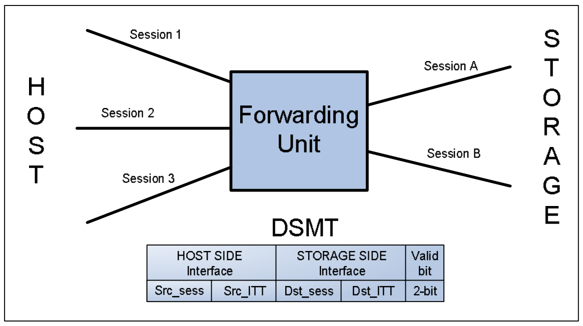


Figure 6. The DSMT in the Forwarding Unit

(2) When an iSCSI header is passed to the PDH, the virtual access location and access length of the SCSI command are extracted to be the inputs for the Address Translator (AT). After the converting of AT, the virtual access location and length are resolved into physical ones, and a great care must be taken if the result turns out to be two more physical locations over the different iSCSI sessions. For this situation, one command request must be taken apart into two requests for the data. Figure 7 illustrates that one request is split into two corresponding mapping entries by the PDH, and finally these mapping entries are added to the DSMT. On the DSMT, duplicating entries from a single request can be recognized by the value of v-bit at the end of the entry.

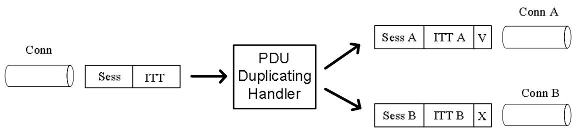


Figure 7. Two entries from one single request

After the duplicating process of PDH, the transaction of a single request on the forwarding system is showed in Figure 8. When a request comes from the client side that actually contains two physical locations (Block A+B), it will be separated into two requests for the two different physical locations, and then they will be issued to the storage side one by one. By the feature of the PDH, the client will get the same response as issuing an access to a single storage device whenever it generates a request to the forwarding system.

3.6 Header adaptor

The header adaptor is designed to form a legal iSCSI header that ready to be transmitted. By the information of DSMT, the header adaptor can find the structure of iSCSI session, and it can get enough information to carry out the replacements for several fields on the iSCSI header.

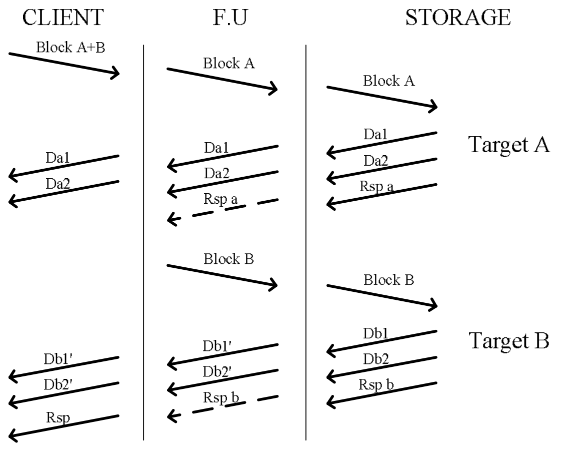


Figure 8. The transaction of a single request to two targets

3.7 The Data mover

The data mover is designed to generate the arguments for the need of TCP socket transferring. It is activated by the PDH after the header adaptor has finished the transmission of its new iSCSI header. The payloads that stay at the TCP socket buffer will be redirected to be the transmitting data by adapting the receiving arguments for transmitting ones.

4. Verification and Performance Evaluation

In testing environment, all the hosts with different OS could access LUN correctly as long as they have installed the iSCSI initiator driver, and the packet forwarding virtualization switch could control the various storage systems well. This proves that the forwarding switch is compatible with multi OS and with various storage. To verify the functionality of our design, we evaluated the performance of the packet forwarding system with the iometer, which is a standard benchmark used for measuring I/O performance. It can measure the READ/WRITE performance in a sequential/random manner and test the I/O latency. The software iSCSI virtualization switch runs on a system illustrated in Table 1. We mapped one disk to one host, and the host issued random read/write commands with different data block sizes to the remote disks. The two implementations, conventional and forwarding, were under test, and an end-to-end implementation was also shown to indicate the best performance between the host and disk by connecting each other directly without any server or switch involved in its data path. Figure 9 shows the read throughput of the three implementations, and Figure 10 shows the write results.

Table 1. Measurement Environment for Software iSCSI virtualization switch

Host	CPU: Pentium IV 2.8GHz, 1GB memory, Windows XP, Microsoft iSCSI initiator 1.06
Storage	CPU: Pentium IV 2.8GHz, 1GB memory, Fedora 1 default Kernel 2.4.22, unh-iscsi-target 1.5.03
Virtualization switch	CPU: Pentium IV 2.8GHz, 1GB memory, Fedora 1 default Kernel 2.4.22, unh-iscsi-initiator 1.5.03, and unh-iscsi-target 1.5.03
Network	Ethernet 1Gbps

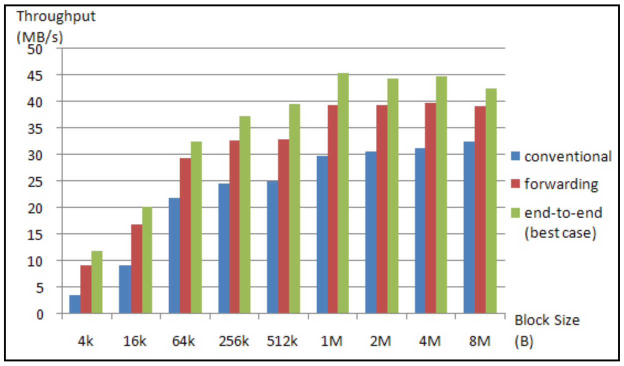


Figure 9. The read throughput

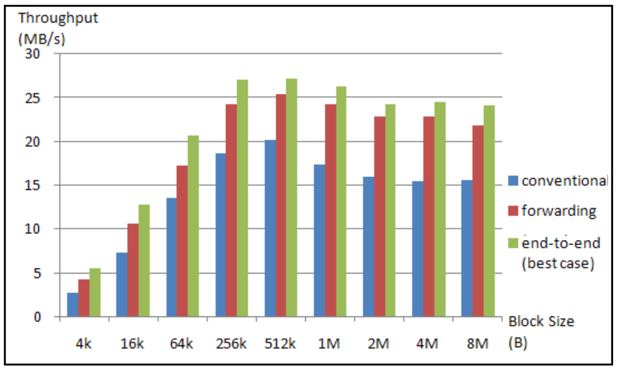


Figure 10. The write throughput

The results show that the forwarding switch has an average of 30% read throughput better than the conventional switch, and a less of about 5 MB/s to achieve the best throughput measured by the end-to-end implementation. In the case of write test, the forwarding switch has an about 25% throughput better than conventional switch, and a less of about 3 MB/s to achieve the best performance. The CPU utilization of the two implementations is shown in Figure 11. The CPU utilization of the forwarding system was 10%~15% lower than the conventional implementation after the access block size is greater than 512KB. For the results, the forwarding implementation has a higher data throughput while having a lower CPU utilization, proving that it can improve the performance of the conventional virtualization switch.

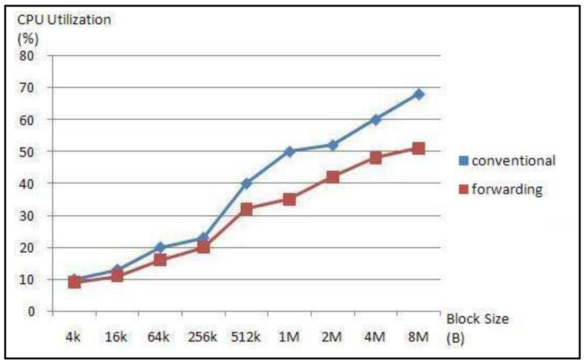


Figure 11. The CPU utilization

5. Conclusion

This paper presents the design and analysis of an iSCSI virtualization switch. We have described a packet forwarding method to reduce the protocol overheads of the upper layers. First, we design a conventional implementation of iSCSI virtualization switch. The proposed forwarding subsystem can be easily integrated into the contemporary virtualization switch system. Our performance evaluation shows that the forwarding system has increased about 30% of READ throughput and about 25% of WRITE throughput compared to the conventional system on the typical block size of 512kB. The proposed forwarding system reduces the CPU utilization about 10%~15% compared to the conventional system.

References

[1] B.Phillips, "Have Storage Area Networks Come of Age?" *[J] IEEE Computer*, vol.31,no.7, 10-12, July 1998.
[2] R. Khattar, et al., *Introduction to Storage Area Network: Redbooks Publications (IBM)*,1999.
[3] John L. Hufferd, *ISCSI The Univeral Storage Connection*, Addison-Wesley, ISBN 0-202-78419-X, 2002.
[4] Internet Small Computer Systems Interface (iSCSI), RFC 3720, *http://www.ietf.org/rfc/rfc3720.txt*.
[5] Kalman Z. Meth and Julian Satran, "Design of the iSCSI Protocol", Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'03).
[6] Mallikarjun Chadalapaka, "iSCSI State Diagrams", Networked Storage Architecture, NSSO, Rev 0.7, Jan. 07, 2002.
[7] XIOTech Corp., http://www.xiotech.com/,May, 2004.
[8] IBM Corp.

http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg2454 70.pdf, March 2003.

[9] EMC Corp., *http://www.emc.com/products/storage_management/contr olcenter/pdf/H1140_cntrlctr_srm_plan_ds_ldv.pdf*, May 2004.

[10] Andr'e, B., Michael, H.V., "Drive - Costs and Benefits of an Out-of-Band Storage Virtualization System". In Proceedings of the 12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies, College Park, Maryland, USA 2004.

[11] William Todd Boyd, Douglas J. Joseph, Michael Anthony Ko, and Renato John Recio, "iSCSI Driver To Adapter Interface Protocol", US Patent # 20040049603.

[12] Shay Mizarchi, Rafi Shalom, and Ron Grinfeld, "iSCSI Receiver Implementation", US patent # 20030058870.

[13] Microsoft Corp, "iSCSI Software Initiator", *http://www.microsoft.com* .

[14] Intel Corp, "Intel iSCSI project", *http://sourceforge.net/projects/intel-iscsi*,2001.

[15] Ashish Palekar, "Design and Implementation of A Linux SCSI Target for Storage Area Networks", Proceedings of the 5th Annual Linux Showcase & Conference. 2001.

[16] David C. Teigland, Heinz Mauelshagen, *Volume Managers in Linux*, Sistina Software Inc. *http://www.sistina.com*, 2001.