Performance Analysis of iSCSI Middleware Optimized for Encryption Processing in a Long-Latency Environment

Kikuko Kamisaka
Graduate School of
Humanities and Sciences
Ochanomizu University
2-1-1, Otsuka, Bunkyo-ku,
Tokyo 112-8610, Japan
kikuko@ogl.is.ocha.ac.jp

Saneyasu Yamaguchi
Institute of Industrial Science
The University of Tokyo
4-6-1, Komaba, Meguro-ku,
Tokyo 153-8505, Japan
sane@tkl.iis.u-tokyo.ac.jp

Masato Oguchi
Department of Information Sciences
Ochanomizu University
2-1-1, Otsuka, Bunkyo-ku,
Tokyo 112-8610, Japan
oguchi@computer.org

Abstract

Since IP-SAN allows us to reduce initial costs and management costs of storage by using IP networks, iSCSI, common protocol in IP-SAN, is becoming more important. Although one of the key issues for iSCSI is a security measure to access remote storage over IP networks, this is not necessarily established yet. iSCSI can employ IPsec that offers a function of strong encryption. However, the IPsec encryption processing degrades throughput of storage access and increases its CPU load. In addition, it is difficult to execute encryption processing efficiently, because IPsec layer is located in a lower-level.

In this paper, we implemented a middleware system of executing encryption processing in the upper-layer for the communication using iSCSI securely and effectively. Moreover, we simulated the idea of optimization of encryption processing and evaluated its performance by running parallel processes in a long-latency network. As a result, optimazation method of encryption processing is more efficient than that of IPsec in a long-latency network.

1 Introduction

Recently, with rapid performance improvement of Internet and other networks, a volume of data stored and managed in a storage system grows exponentially. Storage Area Network (SAN), a high-speed network used to connect servers to storage, has been introduced to resolve this issue. SAN allows us to reduce the management costs by consolidating storage so as to be managed in a centralized manner.

Nowadays, current generation SAN based on Fibre Channel (FC) technology is commonly used to establish high-speed storage networks. However, due to high costs of FC hardware and lack of FC engineers, there are lots of barriers for the introduction of SAN. With the advent of broadband LAN technologies such as Gigabit Ethernet, the next generation SAN based on IP has been proposed. Because IP-SAN is established using TCP/IP protocol and Ethernet, it enables us to reduce costs of introduction and management compared with FC-SAN. In addition, it is possible to provide seamless integration with existing IP networks and construct storage networks flexibly.

Internet SCSI (iSCSI) protocol, ratified by the IETF in February 2003, is expected to become a dominant IP-SAN protocol in the near future[1]. iSCSI is a block level data transfer protocol, in which a SCSI command is encapsulated into TCP/IP packets and transferred between a server (initiator) and storage (target) over IP networks. Since standard SCSI commands are embedded in iSCSI, users can operate a remote storage device over the Internet as if they were accessing to a local disk connected to the server directly.

When we communicate through the iSCSI networks using TCP/IP protocol, we need to reflect security measures. iSCSI can employ IPsec that offers strong encryption and authentication function of IP packets. IPsec commonly employs safe and secure symmetric-key cryptographic algorithm, such as Triple Data Encryption Standard (3DES). However, since storage access using iSCSI requires sending and receiving large volumes of data, the 3DES encryption processing needs a large amount of calculations. Thus it degrades the performance of communication and burdens the CPU with a heavy load. In addition, because IPsec encrypts in the IP layer located on a lower-level, it can only sequentially process data packets passed from an upper-layer. Therefore, it cannot respond flexibly to the processing in the upper-layer such as TCP layer, so that it can be difficult to

execute encryption effectively.

In this paper, for comprehending the processing in the upper-layer and executing efficient encryption, we have implemented an iSCSI middleware system that encrypts in the upper-layer. In addition, we have evaluated our middleware system by running parallel processes for simulating optimization of encryption processing in a long-latency network. We have also analyzed the result by a throughput modeling and TCP packets visualization. As a result, our proposal, optimization method of encryption processing, is more efficient than using IPsec in a long-latency network.

The rest of this paper is organized as follows. Section 2 provides our proposed method. We present an performance evaluation of our implemented system in Section 3 and its analysis in Section 4 and 5. Section 6 introduces related works. In Section 7, we conclude this paper.

2 Optimization of Encryption processing in the Upper-layer

In this section, we present explanation of our proposed method.

There is an issue of performance degradation of storage access caused by IPsec in order to connect to storage securely. While IPsec can encrypt data transparently without changing upper software, the 3DES encryption processing employed in IPsec causes the performance degradation. IPsec is located on the lower-level (IP layer) and it only sequentially processes data passed from the upper-layer. Thus it is difficult to encrypt data effectively and to devise methods for improving performance.

In our proposed method[2], transferred data is encrypted in the upper-layer so as to respond processing such as in TCP layer flexibly. Thereby, it enables the system to apply a performance improvement method an optimization of encryption processing easily.

Figure 1 shows an example of sequential read access using iSCSI in the case of encrypting in the upper-layer. First, an iSCSI read command is issued from an initiator to a target. After data is read from the target's disk, it is encrypted in the upper-layer at the target. Transferred data is decrypted in the upper-layer at the initiator and Ack is sent back to the target. In this case, while the one machine is encrypting or decrypting, waiting time for communications exists at the other machine. Figure 2 shows an example of sequential read access using iSCSI in the case of optimizing the encryption processing in the upper-layer. As shown in this figure, if the optimization of encryption processing is added to the upper-layer, for instance, it is possible to encrypt the next data during waiting time for communications. Overwrapping the iSCSI encryption cycle makes effective use of CPU availability and the system performance improvement is assumed.

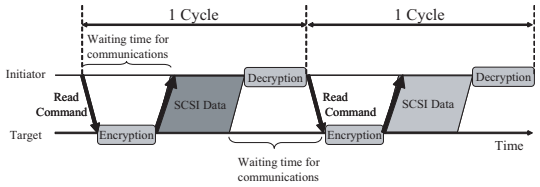

Figure 1. iSCSI Sequential Read Access with an Encryption in the Upper-layer

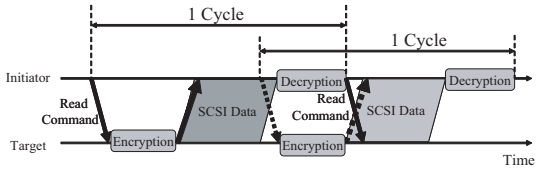

Figure 2. iSCSI Sequential Read Access with an Optimization of Encryption Processing in the Upper-layer

3 Evaluation of our middleware system

3.1 Implementation and Experimental Setup

In this paper, we have implemented a system as middleware based on the method of encryption processing in the upper-layer. Figure 3 shows our middleware system. Our system is implemented as an encryption function located on the upper-layer in the initiator. In the target, we construct the upper encryption middleware as a kernel module. Though the optimization of encryption processing mentioned in the previous section is not implemented in the system, the function is due to be incorporated in the middleware. We use the same 3DES implementation code as IPsec in our middleware.

Table 1 shows the experimental environment. iSCSI reference implementation developed by the University of New Hampshire InterOperability Laboratory is used[3]. As an IPsec implementation, FreeS/WAN for Linux[4] is used. IPsec is set up with a transport mode used to encrypt a host-to-host communication, and an ESP protocol is used.

We have constructed a no-latency and long-latency IP-SAN environment. In the no-latency network (one-way delay time is 0ms), the experimental system consists of the initiator and the target connected with the Gigabit Ethernet. In contrast, a network delay emulator is inserted between the initiator and the target in the long-latency network (one-way delay time is from 1ms to 8ms). The network delay emulator is constructed with FreeBSD Dummynet.

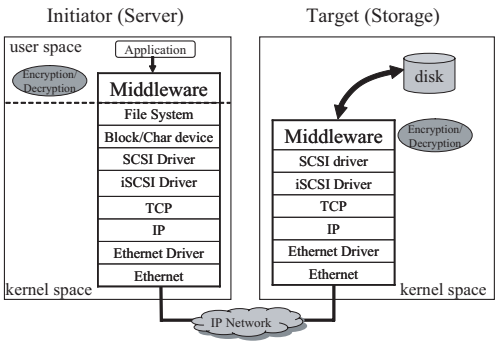

Figure 3. Our Middleware System

Table 1. Experimental System

OS	Initiator : Linux 2.4.18-3 Target : Linux 2.4.18-3 Dummynet : Free BSD 4.9 - RELEASE
CPU	Intel Xeon 2.4GHz
Main Memory	512MB DDR SDRAM
HDD	36GB SCSI HD
NIC	Initiator, Target : Intel PRO/1000XT Dummynet : Intel PRO/1000MT Server Adapter on PCI-X (64bit, 100MHz)
iSCSI	UNH-iSCSI Initiator and Target for Linux ver. 1. 5. 3
IPsec	FreeS/WAN ver. 2.01

3.2 Evaluation Experiment

We evaluate our middleware system by experiments in order to test the effectiveness of our proposed method, optimization of encryption processing. In this experiment, we measure performance of iSCSI sequential read access to the target's raw device, and compare it with the performance of IPsec.

Though we use our middleware system that encrypts data in the upper-layer, optimization of encryption processing is not implemented yet. Thus, we issue parallel processes for simulating the optimization of encryption processing. In our middleware system, first, multiple iSCSI read commands are issued as parallel processes from the initiator to the target. Next, plain data stored in the target's disk is encrypted at target's middleware located on the upper-layer. Even though waiting time for communications is long, our middleware encrypts the data consecutively, thus CPU idle time is reduced at the target. The encrypted data are transferred to the initiator via IP networks and are decrypted on the upper-layer in the initiator. As stated above, the evaluation of sequential read access using parallel processes simulates optimization of encryption processing.

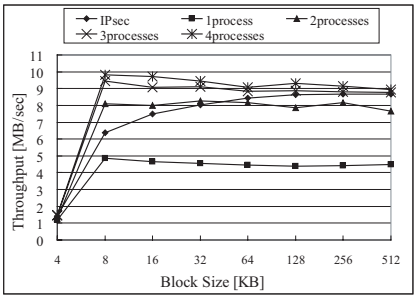

Figure 4. Throughput in a No-latency Network

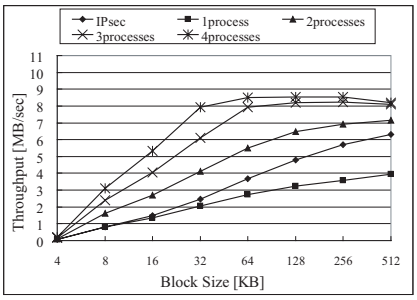

Figure 5. Throughput in 4ms Latency Network

In contrast, because IPsec cannot deal with a processing on the upper-layer, it is difficult for IPsec to realize the highly functional processing such as the optimization of encryption processing. Accordingly, in IPsec as a comparison, we evaluate iSCSI sequential read access with a single process.

3.3 Throughput Results

The experimental throughput results are shown in Figure 4, 5 and 6. In the case of large block size, there is not so much throughput difference between our system and IPsec. In contrast, in the case of small-to-medium block size, the throughput of our system is much higher than that of IPsec. In this regard, though these figure shows the results from 4KB to 512KB block size, the throughput of IPsec is saturated in a block size of more than 512KB in all cases.

Table 2 shows an average throughput improvement ratio against IPsec in each one-way delay time. Though the throughput of our system with a single process is lower than that of IPsec, the throughput of our system with three or four processes improve considerably compared with IPsec. Our system's throughput of four processes in a no-latency network achieves about 1.2 times of IPsec. Moreover, as one-way delay time increases, the improvement ratio be-

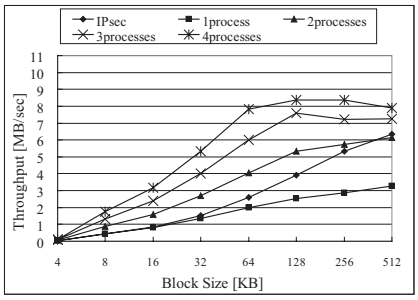

Figure 6. Throughput in 8ms Latency Network

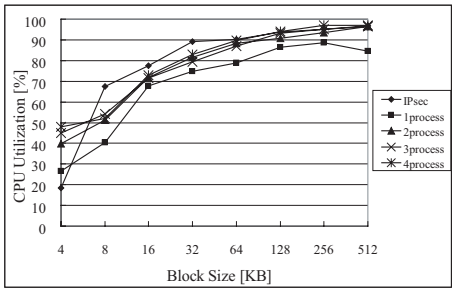

Figure 7. CPU Utilization in a No-latency Network (Target)

Table 2. Throughput Improvement Ratio

one-way delay time	IPsec	our system 1process	2processes	3processes	4processes
0ms	1.000	0.606	0.997	1.120	1.164
1ms	1.000	0.667	1.237	1.608	1.889
2ms	1.000	0.696	1.342	1.796	2.127
4ms	1.000	0.800	1.571	2.194	2.658
8ms	1.000	0.777	1.556	2.239	2.865

Table 3. Average CPU Utilization (Target)

one-way delay time	IPsec	our system 1process	2processes	3processes	4processes
0ms	78.571	68.491	76.672	77.768	79.201
8ms	24.632	17.356	35.389	47.499	56.387

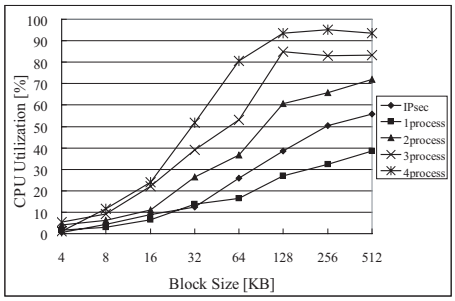

Figure 8. CPU Utilization in 8ms Latency Network (Target)

comes higher. Our system's throughput of four processes in 8ms one-way delay time achieves 2.9 times of IPsec. This is because CPU availability (waiting time for communication) becomes longer as one-way delay time becomes longer. In fact, the advantage of optimized encryption processing increases in a long-latency network by encrypting next data consecutively before the previous encryption cycle is finished.

3.4 CPU utilization results

The experimental CPU utilization results are shown in Figure 7 and 8. The CPU utilization is measured in every second by "iostat" tool at the target. Table 3 shows an average of CPU utilization.

In our system, as the number of process is larger, the CPU load becomes heavier. In the case of the no-latency network, the CPU utilization is saturated and reaches about 80% in both our system and IPsec. In contrast, in the case of the long-latency network, the CPU utilization of our system with more than two processes is higher than that of IPsec. However, our system's CPU in the long-latency network can still afford to process different from the case with the no-latency network. The average of CPU utilization of our system with four processes is 56% maximum. We assume that the next data encryption processing begins concurrently during the waiting time for communications because communication time becomes longer in the case of the long-latency network.

3.5 Security Concerns

From a security standpoint, in the case of a tunnel mode that encrypts transferred data from gateway to gateway, IPsec encrypts data and header in TCP layer. In this experiment, our system does not encrypt TCP header. However, in this evaluation, because IPsec uses a transport mode, IP header is not encrypted in both our system and IPsec. Thus, security level is almost the same in both cases.

4 Analysis of TCP packets transfer

In this section, for confirming the encryption behavior of our system and IPsec, we visualize a behavior of TCP packets transferred between the initiator and the target by

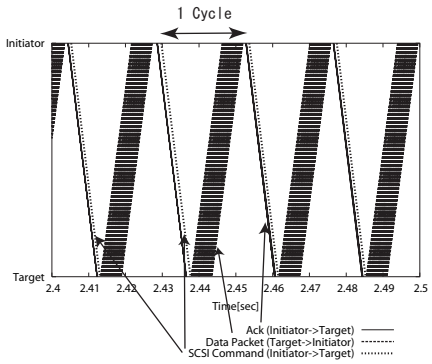

Figure 9. TCP packet transfer : IPsec

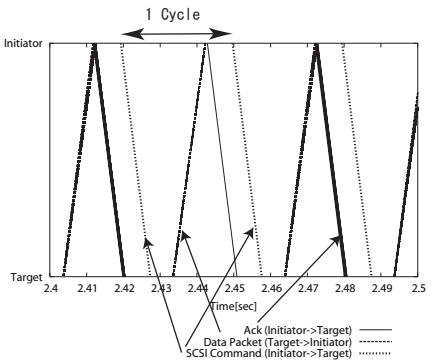

Figure 10. TCP packet transfer : our system (a single process)

capturing them with a tcpdump tool. The block size is 64KB and one-way delay time is 8ms. Figure 9 shows an example of a TCP packets transfer in IPsec. Figure 10 and 11 show examples of a TCP packets transfer in our system using a single and two processes, respectively.

In the case of IPsec, a data segment is divided into a small size packets and encrypted after they are passed to IP layer located on the lower-level. Thus, the encryption time is short per one data packet and it is about 0.1ms as shown in Figure 9. In contrast, in the case of our system in a single process, data segments are encrypted in bulk in the upper-layer. The encryption time is long per one packet and it is about 6.7ms as shown in Figure 10.

In addition, more SCSI read commands are issued concurrently in our system Figure 11 compared with a single process case. Therefore, these figures illustrate that the encryption processing in two processes can be parallelized different from IPsec.

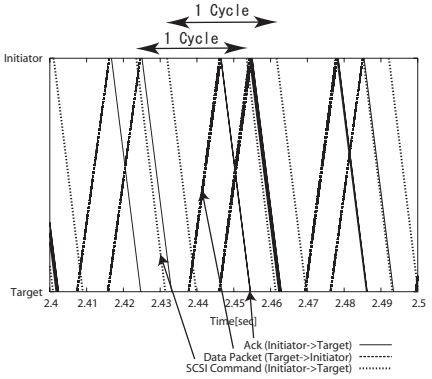

Figure 11. TCP packet transfer : our system (two processes)

5 Throughput modeling

In this section, we state an analysis of the experimental results in our middleware system by modeling the throughput.

First, the initiator issues a SCSI read command. Next, encrypted data segments are transferred from the target and decrypted in the initiator. We refer to this sequence as 1 cycle time in this section. The following equation is a relational expression of 1 cycle time (1CYCLE), data transfer time (TRANSFER), Round Trip Time (RTT), encryption time (ENC) and decryption time (DEC).

1CYCLE

$$= RTT + TRANSFER + ENC + DEC \qquad (1)$$

Thus, the projected throughput of sequential read access in our system with a single process is modeled as the following Equation (2) using block size (BLOCK), encryption throughput (ECN_TH) and decryption throughput (DEC_TH).

THROUGHPUT

$$= \frac{BLOCK}{RTT + \frac{BLOCK}{SOCKET} + \frac{BLOCK}{ENC\_TH} + \frac{BLOCK}{DEC\_TH}} \qquad (2)$$

The throughput of the lower-layer (SOCKET) is that of below iSCSI-layer. In fact, this is the value in a simple socket communication without iSCSI.

On the other hand, as shown in Figure 2, the relational expression of throughput in the optimization is modeled as the following Equation (3). The 3DES algorithm works about the same in both an encryption and a decryption. We assume that the decryption time is the same as encryption time. Because the next data segments are encrypted during waiting time for communications, decryption time is hidden by encryption time.

THROUGHPUT

$$= \frac{BLOCK}{RTT + \frac{BLOCK}{SOCKET} + \frac{BLOCK}{DEC\_TH}} \qquad (3)$$

Table 4. Calculated Throughput and Actual Measurement in a Single Process

Block Size	Calculated Value (MB/sec)	Actual Measurement (MB/sec)
4KB	3.867	1.192
8KB	4.266	4.844
16KB	4.348	4.649
32KB	4.608	4.547
64KB	4.624	4.446
128KB	4.663	4.381
256KB	4.698	4.428
512KB	4.734	4.477

Table 5. Calculated Throughput and Actual Measurement in Two Processes

Block Size	Calculated Value (MB/sec)	Actual Measurement (MB/sec)
4KB	6.192	1.284
8KB	7.301	8.106
16KB	7.781	7.995
32KB	8.409	8.282
64KB	8.546	8.164
128KB	8.672	7.858
256KB	8.762	8.150
512KB	8.840	7.647

Table 4 shows calculated values from the Equation (2) and actual measurements with a single process in a no-latency network. The values of the calculated throughput using the modeling are very close to the actual measurements except a 4KB block size. It is demonstrated that the throughput modeling by Equation (2) is almost correct.

Table 5 shows calculated values from the Equation (3) and actual measurements with two processes in a no-latency network. From Table 5 in the case of optimizing of encryption processing, though the values of actual measurement have varied a bit through the block sizes, the calculated throughput and the actual measurements are about the same except a 4KB block size.

As mentioned above, from these modeling equations, the optimization of encryption processing by overwrapping the encryption cycle is efficient.

6 Related Work

Some studies present performance evaluation of iSCSI.

Sarkar et al.[5] have evaluated iSCSI software implementations and hardware implementations such as TCP Offload Engine (TOE) and Host Bus Adapter (HBA). They presented while such hardware is effective for reducing CPU utilization , it does not achieve better performance than that of the software implementation.

Radkov et al.[6] have investigated performance evaluation of sequential access in iSCSI and NFS and they presented that iSCSI outperforms NFS in terms of throughput and CPU utilization.

In a security-related work, Tang et al.[7] have compared IPsec and SSL security schemes using iSCSI. They mentioned that SSL outperforms IPsec in a large block size although a throughput of IPsec is higher than SSL in small block size.

Many papers about iSCSI performance evaluation are presented until now. However, security techniques such as an encryption for improving performance using iSCSI are not discussed.

7 Conclusion

In this paper, for the realization of secure storage access on iSCSI networks, we implemented the middleware system of an encryption in the upper-layer, instead of using IPsec encryption that leads to the performance degradation. We simulate the optimization of encryption processing by using parallel processes and evaluate our middleware system in a long-latency network experimentally. Moreover, we visualize a behavior of TCP packet transfer and analyze the experimental results in our system by modeling the throughput. As a result, our proposal, optimization of encryption processing in the upper-layer, is more efficient than the method using IPsec.

As a part of future work, we will complete the implementation by building the optimization function in the middleware.

References

[1] iSCSI Draft, http://www.ietf.org/rfc/rfc3720.txt.

[2] Kamisaka, K., Yamaguchi, S. and Oguchi, M.: Performance Evaluation of iSCSI System Optimized for Encryption Processing in the Upper Layer, *Proc. the International Special Workshop on Databases For Next Generation Researchers (SWOD2005) in conjunction with IEEE International Conference on Data Engineering (ICDE2005)*, pp. 204–207 (2005).

[3] InterOperability Lab in the University of New Hampshire, http://www.iol.unh.edu/.

[4] FreeS/WAN Project, http://www.freeswan.org/.

[5] Sarkar, P., Uttamchandani, S. and Voruganti, K.: Storage over IP: When Does Hardware Support help?, *Proc. FAST 2003, USENIX Conference on File and Storage Technologies*, pp. 231–244 (2003).

[6] Radkov, P., Yin, L., Goyal, P., Sarkar, P. and Shenoy, P.: Performance Comparison of NFS and iSCSI for IP-Networked Storage, *Proc. FAST 2002, USENIX Conference on File and Storage Technologies*, pp. 101–114 (2004).

[7] Tang, S.-Y., Lu, Y.-P. and Du, D. H. C.: Performance Study of Software-Based iSCSI Security, *Proc. First International IEEE Security in Storage Workshop*, pp. 70–79 (2002).