On Maximizing iSCSI Throughput using Multiple Connections with Automatic Parallelism Tuning

Fumito Inoue*, Hiroyuki Ohsaki*, Yoshihiro Nomoto † and Makoto Imase*
* Graduate School of Information Science and Technology Osaka University
1-5, Yamadaoka, Suita, Osaka, 565-0871, Japan
E-mail: {f-inoue, oosaki, imase}@ist.osaka-u.ac.jp
† NTT Service Integration Laboratories NTT Corporation
3-9-11, Midori-Cho, Musashino-Shi, Tokyo, 180-8585, Japan
E-mail: nomoto.yoshihiro@lab.ntt.co.jp

Abstract— **In this paper, we propose an iSCSI-APT (iSCSI with Automatic Parallelism Tuning) that maximizes iSCSI throughput in long-fat networks. In recent years, as a protocol for building SANs (Storage Area Networks), iSCSI has been attracting attention for its low cost and high compatibility with existing networking infrastructure. However, it has been known that iSCSI throughput degrades in a long-fat network. iSCSI supports a feature called *multiple connections*, which allows data delivery over multiple TCP connections in a single session. However, for effective utilization of the multiple connections feature, the number of multiple connections must be appropriately configured according to the network status. In this paper, we propose the iSCSI-APT that automatically adjusts the number of multiple connections according to the network status. Through experiments using our iSCSI-APT implementation, we demonstrate that iSCSI-APT operates quite effectively regardless of the network delay.**

I. Introduction

Because of rapid advancement and development of networking technologies, strong requirements on remote backup using a communication network has been emerging [1]. Since the amount of data handled by organizations or individuals has been significantly increasing, remote backup is getting more and more important. Among several approaches for data backup, remote backup should be promising for safety because backup data is stored at a geographically different location. One of major technologies for remote backup is SAN (Storage Area Network), which builds a network of storage devices connected by a communication network [2].

iSCSI (Internet Small Computer System Interface), which encapsulates a stream of SCSI CDBs (Command Descriptor Blocks) in a TCP/IP network, is one of the promising protocols for SANs because of its low cost and compatibility with existing infrastructures [3, 4]. iSCSI was standardized by IETF in 2004, and has been widely adopted in a large numbers of sites recent years [5]. iSCSI simply allows interconnection of SCSI devices via a TCP/IP network. The iSCSI layer is located between the SCSI and TCP layers. Hence, using the iSCSI protocol, existing applications can extend their reachability to remote storage devices as well as local ones without any modification to those applications.

iSCSI does realize global connectivity to remote storage devices, but it still has several issues to be solved — in particular, performance issues [6]. iSCSI naturally utilizes the TCP (Transmission Control Protocol) for encapsulating SCSI CDBs, which results in low end-to-end performance in a long-fat network. Remote backup is a bandwidth-intensive application in a sense that a large volume of data is transferred over a wide-area network. For realization of efficient and effective remote backup using the iSCSI protocol, it is crucial to achieve high iSCSI throughput even in a long-fat network.

There exist several factors that affect the performance of the iSCSI protocol in a long-fat networks. One of the most significant factors is the TCP performance in a long-fat networks. Performance degradation of the TCP protocol in a long-fat network is a well-known problem, and there have been a huge number of researches for improving the TCP performance [7, 8]. From our past studies on improving the TCP performance [9, 10], we believe that a technique of parallel TCP connections, which uses multiple TCP connections for data delivery, is one of promising approaches for achieving high TCP performance in a long-fat network.

The iSCSI protocol supports a feature called *multiple connections*, which allows data delivery over multiple TCP connections in a single session [5]. With the multiple connections feature, it is expected that the high iSCSI performance is realized in a long-fat network. However, as we will explain in Section II, the iSCSI performance is not always improved with the multiple connections feature. On the contrary, the iSCSI performance is further degraded with an appropriate usage of the multiple connections feature.

For effective use of the multiple connections feature, the parallelism (i.e., the number of multiple TCP connections established in parallel) must be appropriately configured according to network status.

In this paper, we therefore propose a parallelism tuning mechanism for the iSCSI protocol called *iSCSI-APT* (iSCSI with Automatic Parallelism Tuning), which automatically adjusts the number of multiple connections according to network status. iSCSI-APT is primarily designed for bulk data transfer applications such as remote backup. iSCSI-APT measures the network status in a passive way, and automatically adjusts the number of multiple connections so that the iSCSI throughput is maximized. iSCSI-APT operates only at an iSCSI initiator (e.g., host) ; i.e., iSCSI-APT can work with any iSCSI-compliant target (e.g., storage). iSCSI-APT is backward compatible since it does not require any modification to iSCSI targets. We implemented our iSCSI-APT in an iSCSI initiator software. With our iSCSI-APT implementation, we quantitatively investigate the effectiveness of our iSCSI-APT through experiments, and demonstrate that our iSCSI-APT maximizes the iSCSI throughput in a long-fat network.

The organization of this paper is as follows. Section II summarizes related works. Section III briefly explains the multiple connections feature of the iSCSI protocol. The overview and detail of our iSCSI-APT is described in Section IV. Section V is devoted for performance evaluation of our iSCSI-APT using our iSCSI-APT implementation and a network emulator. Finally, Section VI summarizes this paper and discusses future works.

II. Related Works

There have been several researches on performance evaluation of the iSCSI protocol in long-fat networks [11-13]. In [11-13], the performance of the iSCSI protocol is evaluated using experiment [11], simulation [12] or mathematical analysis [13]. Consequently, it has been shown that the iSCSI throughput is significantly degraded when the end-to-end delay (i.e., the delay between iSCSI initiator and target) is large.

Also, the effectiveness of the multiple connections feature of the iSCSI protocol is studied in [7, 14]. Those papers show that with the multiple connections feature, the iSCSI protocol achieves higher throughput than without the multiple connections feature. Those studies, however, examine just one side of the multiple connections feature of the iSCSI protocol. The negative effect of the multiple connections feature — performance loss caused by inappropriate configuration of the number of multiple connections — has not been well investigated.

Various solutions for preventing degradation of the iSCSI throughput in a long-fat network have been proposed. For instance, to prevent degradation of the iSCSI throughput, approaches utilizing multiple links [15, 16] and approaches replacing/modifying the transport protocol [7] have been proposed.

In [15], the iSCSI throughput is improved using multiple connections, each of which traverses a different path using a VPN multihoming. Also, in [16], the iSCSI throughput is improved using multiple connections, each of which traverses a different path using multiple LAN ports and dedicated routers.

On the other hand, in [7], the authors propose a change to the TCP congestion control algorithm for improving fairness among competing TCP connections and also improving the iSCSI throughput.

Approaches using multiple links interfaces are not general; i.e., they force significant restrictions on network environment. Also, modification to the transport protocol, TCP, is not realistic since a large number of iSCSI devices have already been in operation and those devices are usually heterogeneous. Therefore, it is desirable to improve the iSCSI throughput in a way that is independent of the network environment and requires no modification to the transport protocol.

On the other hand, there exist a large number of studies on parallel TCP connections [8, 17-19, 10]. For instance, in [8, 10], authors evaluate the performance of parallel TCP connections. It is shown that, even though parallel TCP connections is generally effective, it cannot realize high TCP throughput unless the number of parallel TCP connections is properly configured. This suggests that, if the number of multiple connections is too small or too large, the iSCSI throughput is degraded. Namely, although the multiple connections feature of the iSCSI protocol is effective for achieving high iSCSI throughput, it is important to configure the number of multiple connections appropriately.

III. iSCSI Multiple Connections

In this section, we briefly explain the multiple connection feature of the iSCSI protocol. Refer to [5] for the details of the iSCSI protocol.

iSCSI supports a feature called *multiple connections*, which allows data delivery over multiple TCP connections in a single session[1] (Fig. 1) [5].

For any iSCSI request issued over a TCP connection, the corresponding response and/or other related PDUs must be sent over the same connection. This restriction is called *connection allegiance* [5]. This significantly simplifies the implementation of the iSCSI protocol, enabling hardware-based implementation of the iSCSI protocol.

The maximum number of TCP connections, which is negotiated between an iSCSI initiator and an iSCSI target, is stored in an iSCSI session parameter called `MaxConnections`. The value of `MaxConnections` is negotiated between the

[1]Note that, in reality, not all iSCSI implementations support the multiple connections.

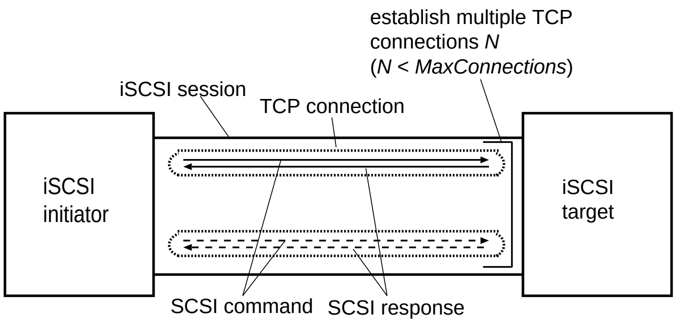

Fig. 1: iSCSI supports multiple TCP connections, which allows data delivery over multiple TCP connections in a single session.

iSCSI initiator and the iSCSI target at the login phase of an iSCSI session.

IV. iSCSI-APT

In this paper, we propose a parallelism tuning mechanism for the iSCSI protocol called *iSCSI-APT* (iSCSI with Automatic Parallelism Tuning), which automatically adjusts the number of multiple connections according to network status. iSCSI-APT operates only at an iSCSI initiator (e.g., host) ; i.e., iSCSI-APT can work with any iSCSI-compliant target (e.g., storage). iSCSI-APT is backward compatible since it does not require any modification to iSCSI targets.

iSCSI-APT adjusts the number of multiple connections according to the network status using the same algorithm as GridFTP-APT (GridFTP with Automatic Parallelism Tuning) [9]. GridFTP-APT searches for the optimal number of TCP connections using a numerical computation algorithm for a maximization problem. GridFTP-APT utilizes the fact that GridFTP throughput is a convex function for the number of multiple connections [10].

iSCSI-APT is primarily designed for bulk data transfer applications such as remote backup. It is because the problem of throughput degradation in long-fat networks is serious when a large amount of data is transfered continuously.

First, we explain iSCSI-APT (Fig. 2).

iSCSI-APT transfers data as a series of blocks called chunk. Chunk transfer is realized by transferring multiple iSCSI PDU (Protocol Data Unit)(Fig. 3). iSCSI-APT measures the iSCSI throughput G for every chunk transfer. By measuring the iSCSI throughput G, iSCSI-APT adjusts the number of multiple connections N and the chunk size x (the data size x that should be transfered by the time the next number of multiple connections is determined).

iSCSI-APT re-calculates the number of multiple connections N after every chunk transfer (after completing transfer of the last iSCSI PDU that is part of the chunk) using the

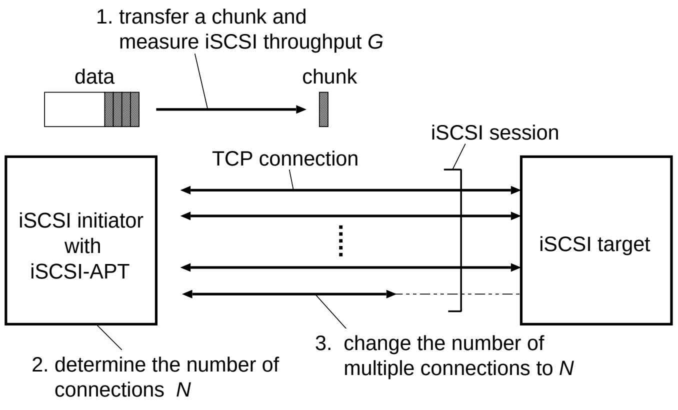
1. transfer a chunk and measure iSCSI throughput G
data
chunk
iSCSI session
TCP connection
iSCSI initiator with iSCSI-APT
iSCSI target
2. determine the number of connections N
3. change the number of multiple connections to N

Fig. 2: Overview of iSCSI-APT operation (to data transfer from iSCSI initiator to iSCSI target); By measuring the iSCSI throughput G, iSCSI-APT determines the number of multiple connections N.

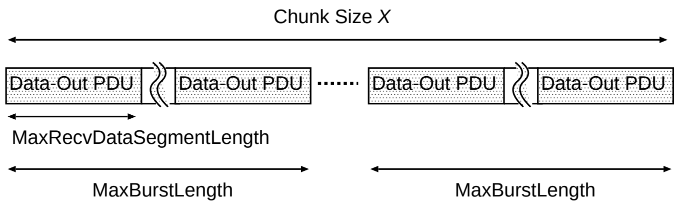

Fig. 3: iSCSI-APT transfers data as a series of blocks called chunk. Chunk transfer is realized by transferring multiple iSCSI PDU.

algorithm of GridFTP-APT [9]. According to the re-calculated value N, iSCSI-APT updates the number of multiple connections.

As mentioned above, measuring the iSCSI throughput G for every chunk transfer, iSCSI-APT determines the number of multiple connections N, which is used for the next chunk transfer.

In what follows, we explain the iSCSI-APT operations when an iSCSI initiator transfers data to an iSCSI target. Specifically, we explain (1) the method of transfering chunk, (2) the method of measuring the iSCSI throughput and (3) the method of updating the number of multiple connections.

Data transfer in the other direction, i.e., data transfer from an iSCSI target to an iSCSI initiator, can also be realized similarly. Refer to [9] for the details of the method of determining the chunk size X and the method of determining the number of multiple connections N.

(1) *Method of chunk transfer*
iSCSI-APT transfers data until the total size of transfered data is lager than the chunk size x. Then iSCSI-

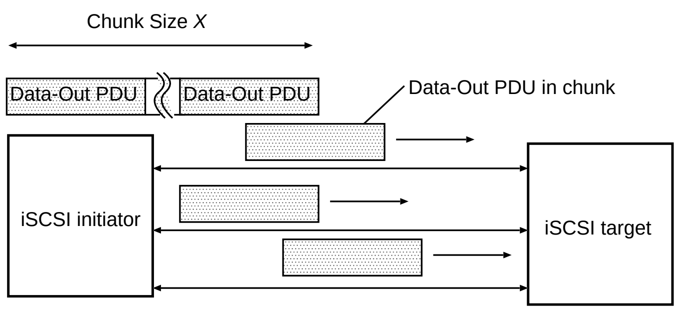

Fig. 4: Method of chunk transfer (from an iSCSI initiator to an iSCSI target); iSCSI-APT transfers a chunk as multiple iSCSI PDUs.

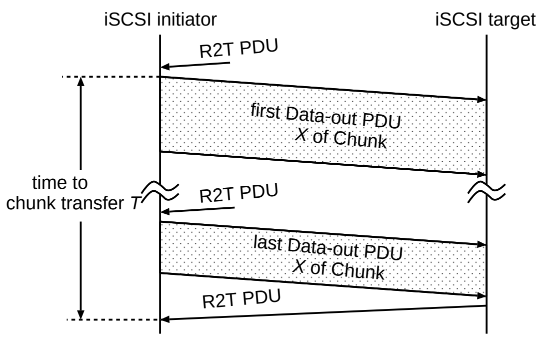

Fig. 5: Measurement method of chunk transfer time; iSCSI-APT measures the duration between the time when the first iSCSI PDU in a chunk is transfered and the time when the last iSCSI PDU in the chunk is transfered.

APT measure the iSCSI throughput. According to the measured iSCSI throughput, iSCSI-APT adjusts the number of multiple connections. iSCSI-APT predicts the iSCSI throughput of the next chunk transfer, and dynamically configures the chunk size so that the chunk transfer time becomes as fixed as possible.

iSCSI-APT transfers chunk as follows.

Chunk transfer is realized by transferring multiple iSCSI PDUs (Fig. 4). SCSI Data-Out and SCSI Data-In are the main vehicles by which SCSI data payload is carried between initiator and target [5]. The maximum size of the iSCSI PDU, which is negotiated between an iSCSI initiator and an iSCSI target, is stored in an iSCSI session parameter called `MaxRecvDataSegmentLength`.

In iSCSI, when the multiple connection is enabled, SCSI CDB (Command Descriptor Block) is usually transfered on each TCP connection in parallel. Note that mapping of each SCSI CDB to a TCP connection is implementation dependent; i.e., it has not been specified in the iSCSI standard [5]. iSCSI-APT calculates the total size of iSCSI PDUs that are transfered on over TCP connections. When the total size of transfered iSCSI PDUs is larger than the chunk size x, iSCSI-APT knows that chunk transfer has been completed.

(2) *Measurement method of the iSCSI throughput* The iSCSI throughput is measured by dividing chunk size X by chunk transfer time T. Specifically, iSCSI-APT measures the iSCSI throughput as follows.

iSCSI-APT obtains chunk transfer time T by measuring the duration between the time when the first iSCSI PDU in a chunk is transfered, and the time when the last iSCSI PDU in the chunk is transfered (the duration between the time when the first iSCSI PDU in chunk is transfered and the time when the R2T PDU is received (Fig. 5)).

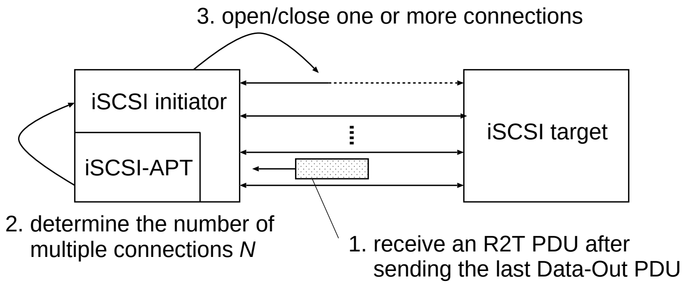
3. open/close one or more connections
iSCSI initiator
iSCSI-APT
iSCSI target
2. determine the number of
multiple connections N
1. receive an R2T PDU after
sending the last Data-Out PDU

Fig. 6: Method of specifying the number of multiple connections; iSCSI-APT notifies the iSCSI initiator of the change in the number of multiple connections.

(3) *Method of specifying the number of multiple connections*

iSCSI-APT updates the number of multiple connections by opening and/or closing connections after completing chunk transfer (Fig. 6). When iSCSI-APT increases the number of multiple connections, iSCSI-APT notifies the iSCSI initiator that the number of multiple connections is increased. In contrast, when iSCSI-APT decreases the number of multiple connections, iSCSI-APT notifies the iSCSI initiator that the number of multiple connections is decreased. Note that when iSCSI-APT decreases the number of multiple connections, iSCSI-APT closes the TCP connection after completing all unfinished SCSI CDB transfers.

V. Evaluation of iSCSI-APT

To evaluate iSCSI-APT, we performed experiments in an emulated long-fat network using our iSCSI-APT implementation and a network emulator.

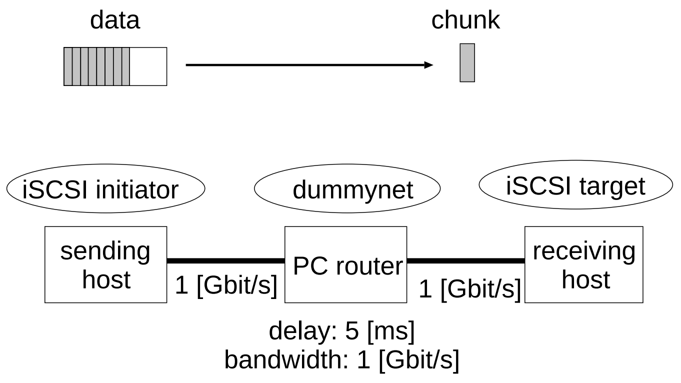

Fig. 7: Network configuration used for the experiments; We transfered continuous data from the iSCSI initiator to the iSCSI target by issuing SCSI write commands of 1 [Mbyte]

TABLE I
PARAMETER CONFIGURATION USED IN EXPERIMENTS

Bandwidth of network emulator	1 [Gbit/s]
Buffer size of router	500 [packet]
Queue discipline of router	DropTail
TCP socket buffer size	1 [Mbyte]
Initial number of parallel TCP connections N_0 [9]	4
Multiplicative increase factor α [9]	2
Target value of chunk transfer time Δ [9]	3 [s]
MaxConnections	500
MaxBurstLength	1 [Mbyte]
ImmediateData	no
InitialR2T	yes

Figure 7 shows the network configuration used for those experiments. We used the same computers with an Intel Xeon 3.06 [GHz] processor with 2 [Gbyte] memory for the iSCSI target, iSCSI initiator and the network emulator. For the iSCSI target and iSCSI initiator, we used our iSCSI-APT implementation based on UNH-iSCSI 1.7.0 [20] running on Debian GNU/Linux 3.1 (Linux kernel 2.6.8). We used FreeBSD 6.1 and dummynet 1.3.14.1 for the network emulator. Unless explicitly stated, parameters shown in Tab. I are used in the following experiments. In the following experiments, we transfered continuous data from the iSCSI initiator to the iSCSI target by issuing SCSI write commands of 1 [Mbyte] To avoid the bottleneck of disk access, the iSCSI initiator was modified to generate a random bit sequence, and the iSCSI target was modified to simply discard the received data.

First, to investigate the effectiveness of iSCSI-APT, we measured the iSCSI throughput when enabling/disabling iSCSI-APT. Figure 8 shows the evolution of iSCSI throughput when iSCSI-APT is enabled/disabled. When iSCSI-APT is disabled, the number of multiple connections N is fixed at 1, 4, 8 and 16. The delay of the network emulator is 5 [ms].

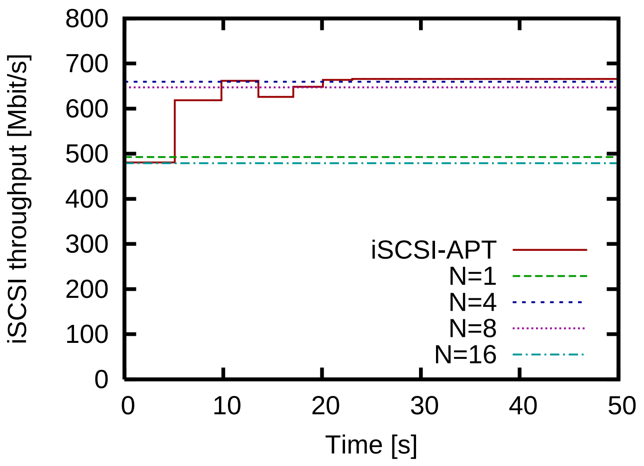

Fig. 8: Evolution of iSCSI throughput (the delay of network emulator is 5 [ms]); When iSCSI-APT is enabled, iSCSI-APT optimizes the number of multiple connections at approximately 20 [s] and maximizes iSCSI throughput.

From the result when iSCSI-APT is disabled, we find that the effectiveness of iSCSI greatly depends on the number of multiple connections. In this experiment, we find that iSCSI throughput is maximized when the number of multiple connections N is 4.

On the other hand, we find that iSCSI-APT optimizes the number of multiple connections at approximately 20 [s] and maximizes iSCSI throughput when iSCSI-APT is enabled. Note that the maximum iSCSI throughput is only 640 [Mbit/s] nevertheless the bandwidth of the network emulator is 1 [Gbit/s]. This degradation of iSCSI throughput is caused by the performance limitation of the iSCSI implementation [21] on which our iSCSI-APT is implemented.

From these result, we demonstrate that iSCSI-APT adjusts the number of multiple connections automatically, and maximizes the iSCSI throughput.

To investigate the effect of the bottleneck link delay on iSCSI throughput, we performed experiments by changing the delay of the network emulator. Figures 9 shows the iSCSI throughput when the delay of the network emulator is changed as 10–40 [ms].

For comparison purposes, iSCSI throughput in steady state when disabling iSCSI-APT and fixing the number of multiple connections at 1 is also plotted in Fig. 9. Figure 9 shows that the throughput when iSCSI-APT is enabled is much larger than the throughput when iSCSI-APT is disabled. Note that iSCSI throughput degrades when the bottleneck link delay is

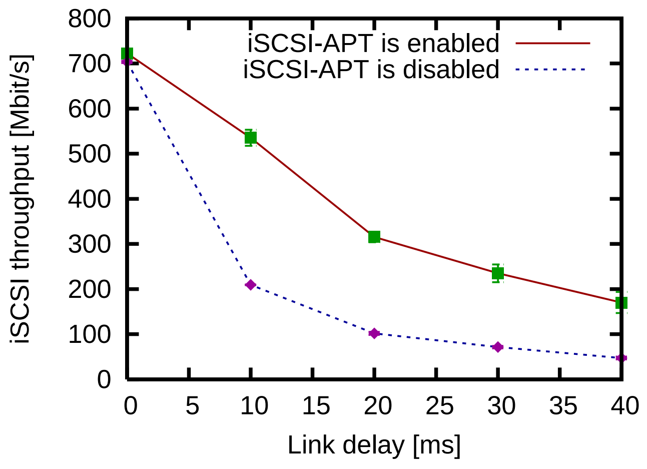

Fig. 9: Bottleneck link delay vs. iSCSI throughput; the throughput when iSCSI-APT is enabled is much larger than the throughput when iSCSI-APT is disabled.

large. Again, this degradation of iSCSI throughput is caused by the performance limitation of the iSCSI implementation [21] on which our iSCSI-APT is implemented.

From these observations, we conclude that iSCSI-APT operates effectively regardless of the bottleneck link delay.

VI. Conclusion

In this paper, we propose iSCSI-APT (iSCSI with Automatic Parallelism Tuning) that automatically adjusts the number of TCP connections appropriately according to the condition of the network. We have also investigated the effectiveness of iSCSI-APT in an emulated long-fat network using our iSCSI-APT implementation and a network emulator. Consequently, we have demonstrated that iSCSI-APT adjusts the number of TCP connections automatically and maximizes iSCSI throughput.

Mainly for simplicity and ease of implementation, we used completely identical algorithm as GridFTP-APT [9], However, the design concept of GridFTP protocol and iSCSI protocol greatly differ. We believe that the further improvement of iSCSI-APT performance is possible by taking account of several characteristic of the iSCSI protocol. As future work, it is therefore of great value to further enhance the performance of out iSCSI-APT.

Acknowledgements

The authors would like to thank Masayuki Murata and Takeshi Ito of Osaka University for their fruitful suggestions.

References

[1] R. P. King, N. Halim, H. Garcia-Molina, and C. A. Polyzois, "Management of a remote backup copy for disaster recovery," *ACM Transactions on Database Systems (TODS)*, vol. 16, pp. 338–368, June 1991.

[2] W. Zheng, F. Wang, and Y. Y. Zhang, "A new backup model based on SAN system," in *Proceedings of The 8th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2004)*, pp. 702–707, July 2003.

[3] K. Z. Meth and J. Satran, "Features of the iSCSI protocol," *IEEE Communications Magazine*, vol. 41, pp. 72–75, Aug. 2003.

[4] C. Mcknight, "Cost Analysis and Long Term Planning Over the Lifecycle of an Enterprise Storage Solution," *Journal of Technology Management & Innovation*, vol. 1, pp. 87–95, Dec. 2006.

[5] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)." RFC 3720 (Proposed Standard), Apr. 2004. Updated by RFCs 3980, 4850, 5048.

[6] W. Ng, B. Hillyer, E. Shriver, and E. Gabber, "Obtaining high performance for storage outsourcing," in *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, pp. 145–158, Jan. 2002.

[7] B. K. Kancherla, G. M. Narayan, and K. Gopinath, "Performance evaluation of multiple TCP connections in iSCSI," in *Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies*, pp. 239–244, IEEE Computer Society, Sept. 2007.

[8] L. Qiu, Y. Zhang, and S. Keshav, "On individual and aggregate TCP performance," in *Proceedings of Internetl Conference on Network Protocols*, pp. 203–212, Oct. 1999.

[9] T. Ito, H. Ohsaki, and M. Imase, "GridFTP-APT: Automatic parallelism tuning mechanism for data transfer protocol GridFTP," in *Proceedings of 6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid2006)*, pp. 454–461, May 2006.

[10] T. Ito, H. Ohsaki, and M. Imase, "On parameter tuning of data transfer protocol GridFTP in wide-area Grid computing," in *Proceedings of Second International Workshop on Networks for Grid Applications (GridNets 2005)*, pp. 415–421, Oct. 2005.

[11] Y. Lu and D. H. C. Du, "Performance study of iSCSI-based storage subsystems," *IEEE Communications Magazine*, vol. 41, pp. 76–82, Aug. 2003.

[12] Y. Lu, N. Farrukh, and D. H. C. Du, "Simulation study of iSCSI-based storage system," in *Proceedings of 12th NASA Goddard & 21st IEEE Conference of Mass Storage Systems and Technologies (MSST 2004)*, pp. 101–110, Apr. 2004.

[13] C. M. Gauger, M. Kohn, S. Gunreben, D. Sass, and S. G. Perez, "Modeling and performance evaluation of iSCSI storage area networks over TCP/IP-based MAN and WAN networks," pp. 915–923, Oct. 2005.

[14] G. Motwani and K. Gopinath, "Evaluation of advanced TCP stacks in the iSCSI environment using simulation model," *Proceedings of the 22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)*, pp. 210–217, Apr. 2005.

[15] N. Chishima, S. Yamaguchi, and M. Oguchi, "Analysis of performance and TCP parameter in the case of multi-routing VPN access on iSCSI storage," *IEICE DEWS2007*, Feb. 2007.

[16] Q. K. Yang, "On performance of parallel iSCSI protocol for networked storage systems," in *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA 2006)*, vol. 1, pp. 629–636, Apr. 2006.

[17] H. Sivakumar, S. Bailey, and R. L. Grossman, "PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks," in *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, Nov. 2000.

[18] T. J. Hacker, B. D. Athey, and B. Noble, "The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network," in *Proceedings of the 16th IEEE-CS/ACM International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 434–443, Apr. 2002.

[19] D. Lu, Y. Quao, P. Dinda, and F. Bustamante, "Modeling and taming parallel TCP on the wide area network," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, Apr. 2005.

[20] "Software implementations of an initiator and a target– UNH-iSCSI." available at `http://unh-iscsi.sourceforge.net/`.

[21] S. Yamaguchi, M. Oguchi, and M. Kitsuregawa, "Performance evaluation of sequential storage access using iSCSI protocol in long-delayed high throughput network," in *Proceedings of The 14th Data Engineering Workshop (IEICE DEWS 2003)*, pp. 137–144, July 2003.