Journal of Information Processing Vol.21 No.3 423–432 (July 2013)

[DOI: 10.2197/ipsjjip.21.423]

Regular Paper

Automatic Parallelism Tuning Mechanism for Heterogeneous IP-SAN Protocols in Long-fat Networks

Takamichi Nishijima[1,a)] Hiroyuki Ohsaki[1,b)] Makoto Imase[1,c)]

Received: October 22, 2012, Accepted: March 1, 2013

Abstract: In this paper we propose *Block Device Layer with Automatic Parallelism Tuning* (*BDL-APT*), a mechanism that maximizes the goodput of heterogeneous IP-based Storage Area Network (IP-SAN) protocols in long-fat networks. BDL-APT parallelizes data transfer using *multiple IP-SAN sessions* at *a block device layer* on an IP-SAN client, automatically optimizing the number of active IP-SAN sessions according to network status. A block device layer is a layer that receives read/write requests from an application or a file system, and relays those requests to a storage device. BDL-APT parallelizes data transfer by dividing aggregated read/write requests into multiple chunks, then transferring a chunk of requests on every IP-SAN session in parallel. BDL-APT automatically optimizes the number of active IP-SAN sessions based on the monitored network status using our parallelism tuning mechanism. We evaluate the performance of BDL-APT with heterogeneous IP-SAN protocols (NBD, GNBD and iSCSI) in a long-fat network. We implement BDL-APT as a layer of the Multiple Device driver, one of major software RAID implementations included in the Linux kernel. Through experiments, we demonstrate the effectiveness of BDL-APT with heterogeneous IP-SAN protocols in long-fat networks regardless of protocol specifics.

Keywords: Automatic Parallelism Tuning (APT), IP-based Storage Area Network (IP-SAN), Block Device Layer, long-fat networks

1. Introduction

In recent years, IP-based Storage Area Networks (IP-SANs) have attracted attention for building SANs on IP networks, owing to the low cost and high compatibility of IP-SANs with existing network infrastructures [1], [2], [3], [4].

Several IP-SAN protocols such as NBD (Network Block Device) [5], GNBD (Global Network Block Device) [6], iSCSI (Internet Small Computer System Interface) [7], FCIP (Fibre Channel over TCP/IP) [8], and iFCP (Internet Fibre Channel Protocol) [9] have been widely utilized and deployed for building SANs on IP networks. IP-SAN protocols allow interconnection between clients and remote storage via a TCP/IP network.

IP-SAN protocols realize connectivity to remote storage devices over conventional TCP/IP networks, but still have unresolved issues, performance in particular Refs. [3], [10], [11]. Several factors affect the performance of IP-SAN protocols in a long-fat network. One significant factor is TCP performance degradation in long-fat networks [12]; IP-SAN protocols generally utilize TCP for data delivery, which performs poorly in long-fat networks.

In this paper, we propose *Block Device Layer with Automatic Parallelism Tuning* (*BDL-APT*), a mechanism that maximizes the goodput of heterogeneous IP-SAN protocols in long-fat networks and that requires no modification to IP-SAN storage devices. BDL-APT parallelizes data transfer using *multiple IP-SAN sessions* at *a block device layer* on an IP-SAN client, automatically optimizing the number of active IP-SAN sessions according to network status. A block device layer is a layer that receives read/write requests from an application or a file system, and relays those requests to a storage device. BDL-APT parallelizes data transfer by dividing aggregated read/write requests into multiple chunks, then transferring a chunk of requests on every IP-SAN session in parallel. BDL-APT automatically optimizes the number of active IP-SAN sessions based on the monitored network status by means of our APT mechanism [13], [14].

We evaluate the performance of BDL-APT with heterogeneous IP-SAN protocols (NBD, GNBD and iSCSI) in a long-fat network. We implement BDL-APT as a layer of the Multiple Device (MD) driver [15], one of the major software RAID implementations included in the Linux kernel. Through experiments, we demonstrate the effectiveness of BDL-APT with heterogeneous IP-SAN protocols in long-fat networks regardless of protocol specifics.

This paper is organized as follows. Section 2 summarizes related works. Section 3 introduces the IP-SAN protocols used in our BDL-APT experiments. Section 4 describes the overview and the main features of our BDL-APT. Section 5 explains our BDL-APT implementation. Section 6 gives a performance evaluation of our BDL-APT implementation in heterogeneous IP-SAN pro-

[1] The Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[a)] t-nisijm@ist.osaka-u.ac.jp
[b)] oosaki@ist.osaka-u.ac.jp
[c)] imase@ist.osaka-u.ac.jp

© 2013 Information Processing Society of Japan

tocols. Finally, Section 7 summarizes this paper and discusses areas for future work.

2. Related Work

Several solutions for preventing IP-SAN performance degradation in long-fat networks have been proposed [12], [13], [16], [17]. For instance, solutions utilizing multiple links [16] or parallel TCP connections [13] have been proposed to prevent throughput degradation of the iSCSI protocol.

Yang [16] improves iSCSI throughput by using multiple connections, each of which traverses a different path using multiple LAN ports and dedicated routers. However, LAN ports and dedicated routers are not always available, forcing significant restrictions on the network environment. Inoue et al. [13] improve iSCSI throughput by adjusting the number of parallel TCP connections using the iSCSI protocol's parallel data transfer feature. However, this solution cannot be used in IP-SAN protocols without that feature. Changes to the TCP congestion control algorithm for improving fairness and throughput of the iSCSI protocol have also been proposed [12]. Oguchi et al. analyze iSCSI with the proposed monitoring tool, and improve the throughput of iSCSI by adjustment of TCP socket buffer, or the correction inside a Linux kernel [17]. However, solutions that replace or modify the transport protocol are unrealistic because they require changes in all IP-SAN targets (storage devices) and initiators (clients).

While these and other solutions for specific IP-SAN protocols have been proposed, many heterogeneous IP-SAN devices have already been deployed, so it is desirable to improve IP-SAN performance independent of protocol and without need for device modifications.

In this paper, we propose an initiator-side solution, which requires no modification to IP-SAN storage devices, to the performance degradation of heterogeneous IP-SAN protocols in long-fat networks.

3. IP-SAN Protocols

In this section, we briefly introduces three IP-SAN protocols used in our BDL-APT experiments.

- NBD (Network Block Device)
  The NBD protocol, initially developed by Pavel Machek in 1997 [5], is a lightweight IP-SAN protocol for accessing remote block devices over a TCP/IP network. The NBD protocol allows transparent access of remote block devices via a TCP/IP network, and supports primitive block-level operations such as read/write/disconnect and several other I/O controls. All communication between NBD clients and servers occurs using the TCP protocol.
- GNBD (Global Network Block Device)
  The GNBD protocol is another lightweight IP-SAN protocol for accessing remote block devices over a TCP/IP network. The GNBD protocol was developed at the University of Minnesota as part of the GFS (Global File System) [6], [18]. As in the NBD protocol, the GNBD protocol allows transparent access to remote block devices via a TCP/IP network. All communication between a GNBD client and server are transferred via TCP. A notable difference between GNBD and NBD is that GNBD allows simultaneous connections between multiple clients and a single server (a block device).
- iSCSI (Internet Small Computer System Interface)
  The Internet Engineering Task Force standardized the iSCSI protocol in 2004 [7]. iSCSI protocol encapsulates a stream of SCSI command descriptor blocks (CDBs) in IP packets, allowing communication between a SCSI initiator (a client) and its target (a storage device) via a TCP/IP network. When a SCSI initiator receives read/write requests from an application or a file system, it generates SCSI CDBs and transfers those CDBs to the SCSI storage through a TCP connection. It is known that iSCSI performance is significantly degraded when the end-to-end delay (the delay between the iSCSI initiator and its target) is large [11], [13], [19], [20].

4. Block Device Layer with Automatic Parallelism Tuning (BDL-APT)

4.1 Overview

We propose BDL-APT, a mechanism that maximizes the goodput of heterogeneous IP-SAN protocols in long-fat networks. BDL-APT realizes the data delivery over multiple TCP connections by using multiple IP-SAN sessions, and optimizes the number of parallel TCP connections automatically based on the IP-SAN goodput. BDL-APT is a mechanism that operates as a block device layer in an IP-SAN initiator (the client) (see **Fig. 1**). BDL-APT parallelizes data transfer by dividing aggregated read/write requests into multiple chunks, then transferring a chunk of requests on every IP-SAN session in parallel. BDL-APT automatically optimizes the number of active IP-SAN sessions based on the monitored network status using our parallelism tuning mechanism APT [13], which is based on a numerical computation algorithm called the Golden Section Search method.

The main advantage of BDL-APT is its independence from the underlying IP-SAN protocol. Namely, BDL-APT can operate with any IP-SAN protocol since it works as a block device layer without reliance on features specific to the underlying block device or protocol. BDL-APT realizes both parallel data transfer and network status monitoring independently from the underlying block device or protocol.

Another advantage of BDL-APT is its initiator-side implementation. Namely, BDL-APT works within an IP-SAN initiator (a

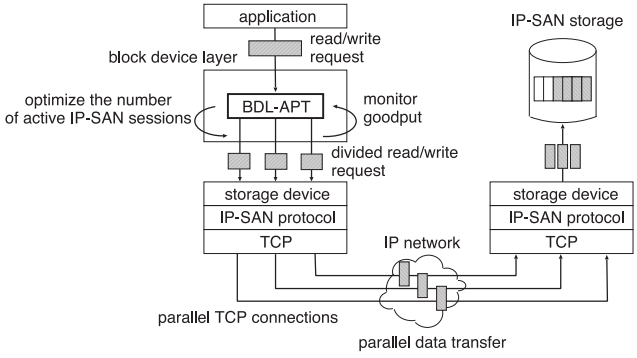


Fig. 1 Overview of BDL-APT. BDL-APT realizes the data delivery over multiple TCP connections by using multiple IP-SAN sessions, monitors the incoming/outgoing goodput, and optimizes the number of active IP-SAN sessions automatically.

© 2013 Information Processing Society of Japan

client), so modification to IP-SAN targets (storage devices) are unnecessary. Thus, BDL-APT can be easily deployed in various IP-SAN environments.

BDL-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 problematic when a large amount of data is transfered continuously.

In what follows, we explain the design and the implementation of our BDL-APT.

4.2 Block Device Layer

A block device layer is a layer that receives read/write requests from an application or a file system, and relays those requests to a storage device. In IP-SAN, a storage device driver handles data delivery to and from an IP-SAN storage device using an IP-SAN protocol (see Fig. 1).

Block device layers are not new; they are adopted, for instance, in the MD driver, a software RAID implementation in Linux, and Violin [21], a framework for extensible block-level storage. The block device layer can perform various types of processing, such as mirroring, striping, and encryption. Such block device layers can be stacked to build new types of block device, for example to improve reliability, access speed, and security.

The following describes the main features of BDL-APT, *data transfer parallelization*, *optimization of the number of parallel TCP connections*, and *goodput monitoring*.

4.3 Data Transfer Parallelization

BDL-APT realizes parallel data transfer by establishing multiple IP-SAN *sessions* to a single storage device. Note that BDL-APT intentionally establishes multiple IP-SAN sessions, instead of multiple *connections* within a single IP-SAN session. Generally, one or more TCP connections carry a single IP-SAN session, meaning that using multiple IP-SAN sessions is equivalent to using multiple TCP connections. BDL-APT splits read/write requests, then transfers split requests in parallel over multiple IP-SAN sessions, making it possible to perform parallel data transfer with any IP-SAN protocol, which does not support parallel data transfer.

BDL-APT maintains multiple IP-SAN sessions to a single IP-SAN storage device. When BDL-APT receives read/write requests (hereafter called *block I/O requests*) from an application or a file system, BDL-APT splits those block I/O requests into multiple chunks and generates multiple block I/O requests for each chunk. BDL-APT parallelizes data transfer by assigning those generated block I/O requests to multiple IP-SAN sessions.

This realizes parallel data transfer in a block device layer, without modification to underlying devices or IP-SAN protocols.

4.4 Optimization of the Number of Parallel TCP Connections

BDL-APT optimizes the number of parallel TCP connections by optimizing the number of active IP-SAN sessions. An IP-SAN protocol utilizing TCP for data delivery establishes at least one TCP connection per IP-SAN session.

BDL-APT maintains multiple IP-SAN sessions. BDL-APT determines the required number of parallel TCP connections, and dynamically changes the number of *active* IP-SAN sessions. By assigning generated block I/O requests to a subset of established IP-SAN sessions, BDL-APT optimizes the number of active IP-SAN sessions used for parallel data transfer. BDL-APT determines the required number of IP-SAN sessions using our Automatic Parallelism Tuning mechanism, APT.

We explain the overview of APT mechanism. Refer to Ref. [13] for the details of APT mechanism. The basic idea of APT is that a client splits data to transfer into blocks called *chunk*, and adjusts the number of parallel TCP connections at the end of every chunk transfer.

In what follows, N is the number of parallel TCP connections used for a chunk transfer, and $G(N)$ the IP-SAN goodput measured at the chunk transfer.

- *Searching the range of the number of parallel TCP connections covering the optimal value that maximizes the IP-SAN goodput*
  First, BDL-APT searches the bracket. BDL-APT starts from a small number of parallel TCP connections, and multiplicatively increases the number of parallel TCP connections at every chunk transfer until IP-SAN goodput decreases. BDL-APT determines the bracket — the range of the number of parallel TCP connections covering the optimal value that maximizes the IP-SAN goodput.
  We illustrate an example operation of BDL-APT. **Figure 2** shows an example operation of BDL-APT when searching for a bracket. BDL-APT searches for a bracket. The number k shown in a circle indicates the k-th chunk transfer. First, BDL-APT initializes the number N of parallel TCP connections to N_0 (= 1). BDL-APT multiplicatively increases the number of parallel TCP connections as $1 \rightarrow 2 \rightarrow 4 \rightarrow 8$ at every chunk transfer until the IP-SAN goodput starts to decrease. Since the IP-SAN goodput decreases when the number N of parallel TCP connections changes as $4 \rightarrow 8$, the bracket is determined as (2, 4, 8).
- *Using the GSS algorithm for maximizing the IP-SAN goodput within the bracket*
  Next, using the GSS algorithm which is one of numerical computation algorithm for a maximization problem, BDL-APT searches the number of parallel TCP connections that maximizes the IP-SAN goodput within the bracket (l, m, r) during succeeding chunk transfers.

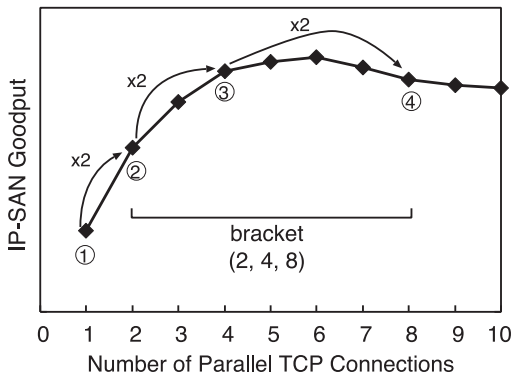


Fig. 2 Example of BDL-APT operation when searching for a bracket.

© 2013 Information Processing Society of Japan

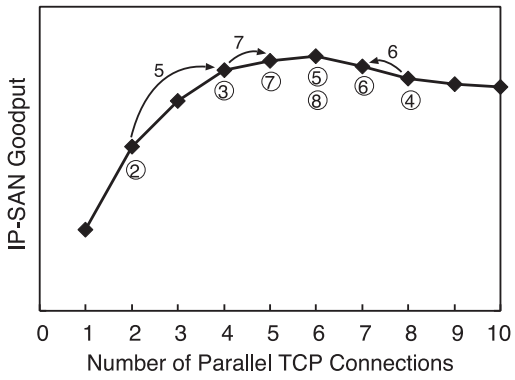


Fig. 3 Example of BDL-APT operation when searching for the optimal number of TCP connections.

BDL-APT searches the optimal number N of parallel TCP connections as follows.

(1) Update the number N of parallel TCP connections:

$$N \leftarrow \begin{cases} \mathrm{int}(l + (m-l)\nu) & \text{if } m - l > r - m \\ \mathrm{int}(m + (r-m)\nu) & \text{otherwise} \end{cases} \quad (1)$$

where ν is the golden ratio (= $(3 - \sqrt{5})/2$) and int(x) is the nearest integer of x.

(2) Transfer a chunk while measuring the IP-SAN goodput $G(N)$.

(3) If the following inequality is satisfied, proceed to the step 4.

$$G(N) > G(m) \quad (2)$$

If the above inequality is not satisfied, change the bracket as follows and return to the step 1.

$$(l, m, r) \leftarrow \begin{cases} (l, m, N) & \text{if } m < N \\ (N, m, r) & \text{otherwise} \end{cases} \quad (3)$$

(4) Change the bracket as follows, and return to the step 1.

$$(l, m, r) \leftarrow \begin{cases} (m, N, r) & \text{if } m < N \\ (l, N, m) & \text{otherwise} \end{cases} \quad (4)$$

Figure 3 shows an example operation of BDL-APT when searching for the optimal number of parallel TCP connections. Since the bracket is (2, 4, 8), the number of parallel TCP connections at the 5-th chunk transfer N is determined as $N = 6$ from Eq. (1). The IP-SAN goodput in the 5-th chunk transfer is $G(6)$, and since $G(4) < G(6)$ is satisfied, the bracket is updated as (4, 6, 8) from Eq. (4). Hereafter, in a similar way, BDL-APT changes the number of parallel TCP connections N as $6 \rightarrow 7 \rightarrow 5$, and updates the bracket as $(4, 6, 8) \rightarrow (4, 6, 7) \rightarrow (5, 6, 7)$. Finally, when the bracket is (5, 6, 7), the number of parallel TCP connections is fixed at $N = 6$, which maximizes the IP-SAN goodput.

4.5 Goodput Monitoring

During parallel data transfer, BDL-APT monitors the goodput at every goodput measurement interval Δ in the block device layer. Δ is one of APT parameters [13].

We measure the incoming/outgoing goodput as follows, respectively. BDL-APT calculates goodput by dividing total of the incoming/outgoing data size through all active IP-SAN sessions among Δ by Δ that is a measurement interval. Data are incoming from a storage device or outgoing to a storage device through multiple IP-SAN sessions during parallel data transfer. BDL-APT gathers those incoming/outgoing data and calculates goodput for them as a "chunk" of APT.

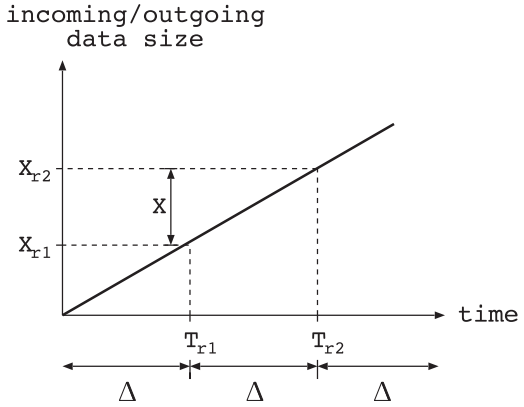


Fig. 4 Evolution of the total incoming/outgoing data size transferred through active IP-SAN sessions.

Figure 4 shows evolution of total incoming/outgoing data size transferred through active IP-SAN sessions. We calculate the goodput G between T_{r1} and T_{r2}. T_{r2} has passed Δ times since T_{r1}. X_{r1} and X_{r2} are the total data size transferred at T_{r1} and T_{r2}, respectively. X is the total data size transferred between T_{r1} and T_{r2}, and is the difference of X_{r1} and X_{r2}. Goodput G is calculated by $G = \frac{X}{\Delta}$.

When BDL-APT assigns generated block I/O requests to multiple IP-SAN sessions, BDL-APT records the size of each data transfer request. Then, BDL-APT calculates the total data size transferred.

5. Implementation

We implemented BDL-APT as a block device layer in the MD driver, a popular software RAID implementation included in the Linux kernel. The MD driver enables creation of a virtual block device composed of one or more underlying block devices.

The BDL-APT module in the MD driver is implemented based on the RAID-0 module with several added functions required for BDL-APT: data transfer parallelization, optimization of the number of parallel TCP connections, and goodput monitoring.

The structure of the RAID-0 module is shown in **Fig. 5**. The RAID-0 module (a) creates the RAID device, (b) manages striped storage devices, (c) splits block I/O requests, and (d) assigns split block I/O requests to multiple storage devices. The block device layer in the Linux kernel handles block I/O requests from/to an application or a file system as *bio structure objects*. The RAID-0 module splits bio structure objects into multiple smaller bio structure objects. It then assigns multiple bio structure objects to the striped storage devices by changing the `bio_bdev` field in each bio structure object.

The BDL-APT module utilizes RAID-0 module features mostly as-is (see **Fig. 6**). Data transfer parallelization is naturally realized by the RAID-0 module, which provides striping over multiple disks. The optimization of the number of parallel

© 2013 Information Processing Society of Japan

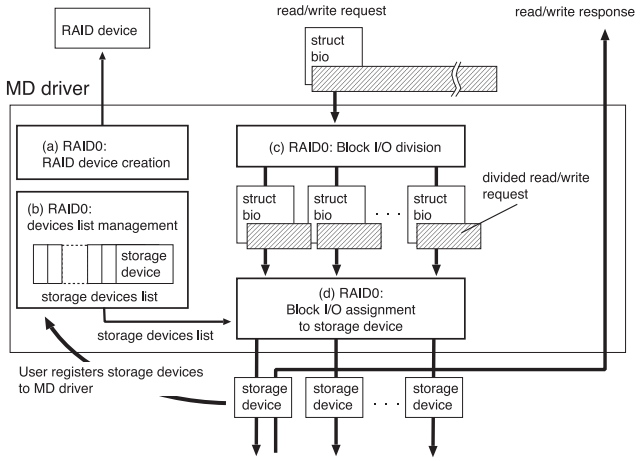


Fig. 5 Structure of the RAID-0 module in the MD driver. The RAID-0 module (a) creates the RAID device, (b) manages striped storage devices, (c) splits block I/O requests (bio structure objects), and (d) assigns split block I/O requests to multiple storage devices.

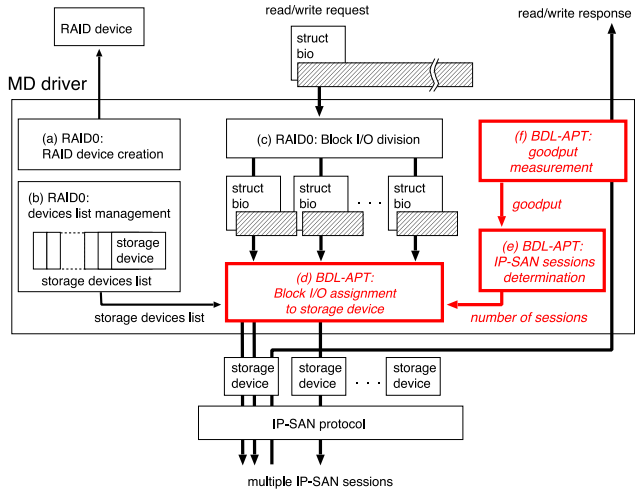


Fig. 6 Structure of the BDL-APT module, which is based on the RAID-0 module. The BDL-APT module has several added functions to the RAID-0 module. Namely, the BDL-APT module (d) dynamically assigns divided bio structure objects to a subset of storage devices, (e) optimizes the number of active IP-SAN sessions, and (f) continuously measures the goodput of block I/O requests.

TCP connections is realized by dynamically changing the number of active IP-SAN sessions module. Goodput monitoring is realized by recording the total data size of all read/write block I/O requests and calculating the goodput at regular intervals.

In what follows, we describe how three functions required for BDL-APT, data transfer parallelization, optimization of the number of parallel TCP connections, and goodput monitoring, are realized in the BDL-APT module. Refer to, for example, Refs. [15], [21] for details of the MD driver internal and the Linux block device layer.

- Data transfer parallelization
  Data transfer is parallelized by splitting bio structure objects passed from the application or the file system and assigning split bio structure objects to the storage devices corresponding to the multiple IP-SAN sessions (see Fig. 6).
  At the IP-SAN client, the BDL-APT module maintains one storage device per IP-SAN session. Data transfer over multiple IP-SAN sessions is thus realized because the BDL-APT module assigns bio structure objects to each storage device. Note that BDL-APT does not utilize all established IP-SAN sessions, but rather parallelizes data transfer for only a number of sessions determined by the network status. Thus, the BDL-APT module is modified from the RAID-0 module so that it dynamically assigns split bio structure objects to a subset of storage devices (see Fig. 6).
- Optimization of the number of parallel TCP connections
  The APT algorithm [13], [14] is implemented in the BDL-APT module. The APT algorithm automatically determines the optimal number of parallel TCP connections according to the network status. More specifically, the APT algorithm periodically determines the optimal number of parallel TCP connections based on the goodput measurement. The BDL-APT module then changes the number of storage devices used for data transfer parallelization, thereby adjusting the degree of multiplexing. Refer to Refs. [13], [14] for the details of the APT algorithm.
- Goodput monitoring
  For monitoring the goodput, the BDL-APT module records the total block I/O response size using a callback function of the block device layer in the Linux kernel. In the Linux kernel, a callback function `endio()` can be used to invoke any function immediately after the block I/O request is completed. For every bio structure object, the pointer to a callback function can be specified in the filed `bio_end_io` of the bio structure object.
  structure objects so that the total block I/O response size can be The BDL-APT module overrides the filed `bio_end_io` of all bio calculated. The BDL-APT module calculates the goodput of data transfer at a given point in time by dividing the total block I/O response size by elapsed time. Then, the BDL-APT module resets the recorded total block I/O response size.

Note that proposed BDL-APT does not need an additional data copy. However, the block device layer needs an additional data copy between buffers. Moreover, BDL-APT does not ensure atomicity and consistency. Instead, users need to ensure atomicity and consistency, they should utilize a block device layer or a file system that ensure atomicity and consistency such as Linux journaling file system.

6. Experiment

6.1 Experiment Design

We evaluated the performance of BDL-APT with several heterogeneous IP-SAN protocols (NBD, GNBD and iSCSI) in a long-fat network. To show the effectiveness of BDL-APT in a realistic environment, we conducted experiments with a network emulator while varying its bandwidth and delay settings.

The network environment comprised an IP-SAN client and storage device, and a network emulator (see **Fig. 7**). We continuously transferred data from the IP-SAN storage device to the IP-SAN client. We measured the goodput for continuous read from a storage to an application. The application repeatedly requests data of 10 [Mbyte] to the storage. We conducted ten experiments and measured the average and 95% confidence interval of mea-

© 2013 Information Processing Society of Japan

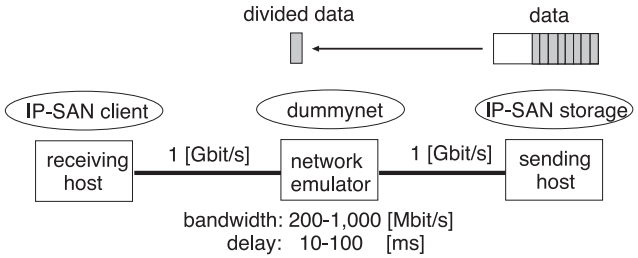


Fig. 7 Network configuration used in experiments. IP-SAN client and storage device are connected via the network emulator to simulate a long-fat network.

surements.

We used computers with Intel Xeon 3.06 [GHz] processors (SL72G) based on NetBurst microarchitecture, 2 [Gbyte] ECC PC266 DDR SDRAM (266 MHz) memory and ServerWorks GC-LE chipsets. The IP-SAN client and storage device run on Debian GNU/Linux 5.0.2 (lenny) with Linux kernel 2.6.26. The network emulator [22] runs on FreeBSD 6.4.

We used several open-source IP-SAN implementations: NBD version 2.9.11 [5], GNBD version 2.03.09 [6], Open iSCSI version 2.6-869 [23], and iSCSI enterprise target version 1.4.20 [24]. The maximum number of IP-SAN sessions in NBD, GNBD, and iSCSI are increased to 128 from their default values.

To avoid the disk drives on IP-SAN client and storage to become the bottleneck, we implemented a virtual storage device in the Linux kernel. When the network bandwidth is high enough, the access speed of the disk drives on either IP-SAN client or storage might become the performance bottleneck. In our experiments, we used our implementation of a virtual storage device, which is a virtual disk drive of an arbitrary size. The virtual storage device does not perform any physical disk drive access. Namely, reading from the virtual storage device simply returns a dummy data, and writing to the virtual storage device always succeeds but all data are simply discarded. In all experiments, we created 128 virtual storage devices with 500 [Gbyte] size. We should note that the goodput of the virtual storage device was approximately 3.2 [Gbit/s], which is sufficiently faster than the network bandwidth in our experiments (i.e., 1 [Gbit/s] at maximum).

We note that the parameter, read-ahead size of the MD driver (`/sys/block/md0/bdi/read_ahead_kb`), in the block device must be configured appropriately in long-fat networks. The default value of read-ahead size of the MD driver in Linux 2.6.26 is the value that multiplied the number of striped storage devices by 128 [KByte], which is not large enough in long-fat networks.

Table 1 shows the parameter configuration used in the experiments.

6.2 Evolution of IP-SAN Goodput

First, we investigate the optimization of the number of parallel TCP connections in realistic network configurations with NBD protocol. **Figure 8** shows the aggregated NBD goodput (the total goodput of all active NBD sessions) when the bandwidth of the network emulator was fixed at 900 [Mbit/s] and the delay of the network emulator was fixed at 40 [ms]. For comparison purposes, NBD goodput in steady state when fixing the number of

Table 1 Default parameters used in experiments; see Ref. [13] for the meaning of BDL-APT parameters.

BDL-APT parameters		
chunk size	128	[Kbyte]
initial number of IP-SAN sessions N_0	4	
maximum number of IP-SAN sessions	128	
multiplicative increase factor α	2	
target value of chunk transfer time Δ	100	[s]
Block device parameter		
read-ahead size of the MD driver	128	[Mbyte]
NBD parameter		
block size	1,024	[Kbyte]
GNBD parameters		
No parameter		
iSCSI parameters		
MaxBurstLength	16	[Mbyte]
ImmediateData	no	
InitialR2T	yes	
TCP parameter		
TCP socket buffer size	512	[Kbyte]
TCP version	NewReno	
TCP SACK	enable	
NIC parameter		
MTU	1,500	[Byte]
Network emulator		
bandwidth	900	[Mbit/s]
delay	10	[ms]
packet loss rate	0	
buffer size	5,000	[packet]

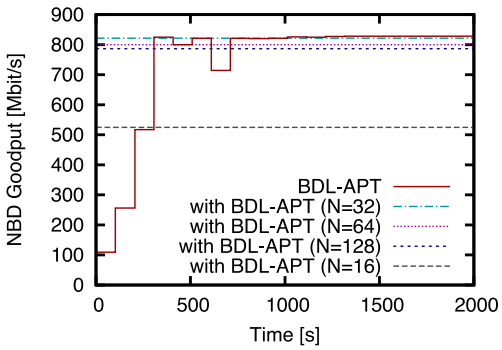


Fig. 8 Evolution of NBD goodput. the number of active NBD sessions is optimized at approximately 1,200 [s], and the NBD goodput converges to 828 [Mbit/s].

active NBD sessions at 16, 32, 64 and 128 are also plotted in the figure. One can find that the optimal number of active NBD sessions seems to exist between 16–64 from the NBD goodput with the fixed number of active NBD sessions. Moreover, this figure shows that the number of active NBD sessions is optimized at approximately 1,200 [s], and the NBD goodput converges to 828 [Mbit/s].

The evolution of the number of active NBD sessions in this scenario is shown in **Fig. 9**. This figure shows that the number of active NBD sessions converges to 33 in 12 steps (i.e., approximately 1,200 [s] with $\Delta = 100$ [s]). This agrees with the result in Fig. 8 where the optimal number of active NBD sessions exists between 16–64. From these observations, we find that BDL-APT optimizes the number of active NBD sessions at approximately 1,200 [s], and utilizes the network resource quite effectively.

In our experiments, the number of active IP-SAN sessions converged in 6–17 steps (i.e., 615–1,740 [s] with $\Delta = 100$ [s]),

© 2013 Information Processing Society of Japan

and the average steps to converge was 12.1 (i.e., 1,240 [s] with $\Delta = 100$ [s]).

6.3 Effect of Network Bandwidth

First, the goodput of the IP-SAN protocols with and without BDL-APT in steady state was measured by changing the bottleneck link bandwidth (the bandwidth throttling at the network emulator). **Figure 10** shows the aggregated IP-SAN goodput (the total goodput of all active IP-SAN sessions after the number of active IP-SAN sessions is optimized) when the bandwidth of the network emulator was varied between 200–1,000 [Mbit/s] while the delay of the network emulator was fixed at 10 [ms]. For comparison, the steady-state IP-SAN goodput with 2, 8, 32, and 128 fixed IP-SAN sessions are also plotted. In order to fix the number of active IP-SAN sessions at 2 or more, we used IP-SAN protocols with our data transfer parallelization in the BDL (see Section 4.3). Figure 10 (a) shows the aggregated NBD goodput when the bottleneck link bandwidth is changed. Similarly, Fig. 10 (b) shows the aggregated GNBD goodput, and Fig. 10 (c) shows the aggregated iSCSI goodput.

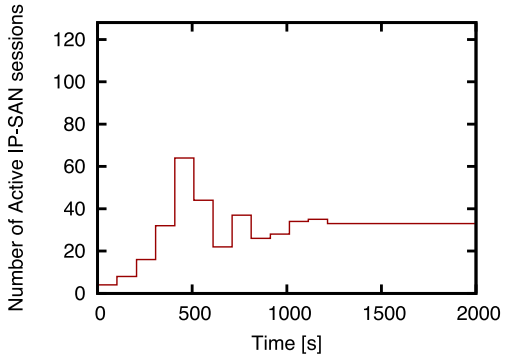


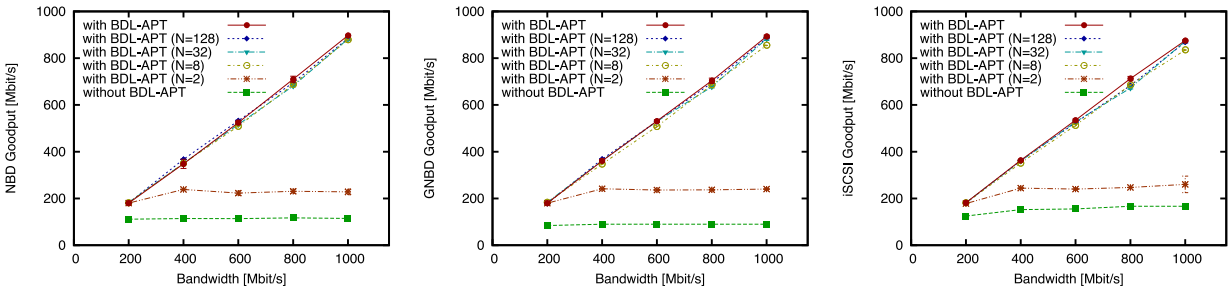
Fig. 9 Evolution of the number of active NBD sessions. the number of active NBD sessions converges to 33 at approximately 1,200 [s].

Figure 10 shows that the IP-SAN protocols without BDL-APT cannot fully utilize the network bandwidth when the number of active IP-SAN sessions is fixed at a small value or BDL-APT is not utilized. In particular, when BDL-APT is not used, the IP-SAN protocol cannot fully utilize the network bandwidth regardless of IP-SAN protocol. This is because the bandwidth delay product increases as the network bandwidth becomes large. Therefore, the number of parallel TCP connections required for fully utilizing the network resources increases. Figure 10 shows that BDL-APT fully utilizes the bottleneck link bandwidth, regardless of IP-SAN protocols.

6.4 Effect of Network Delay

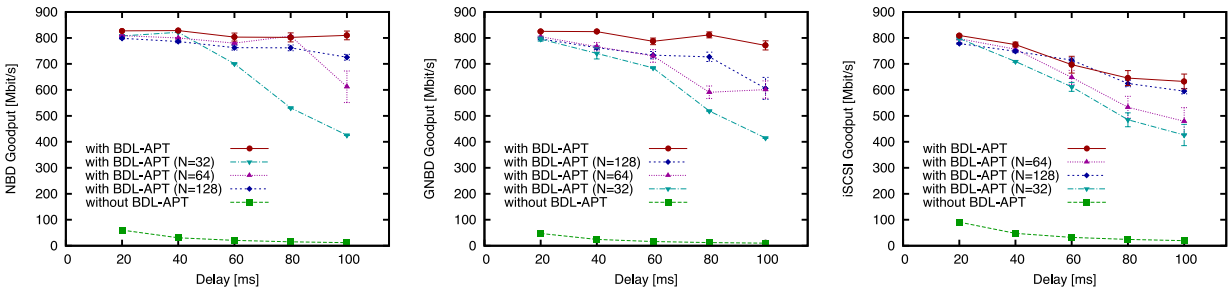
We next measured the goodput of the IP-SAN protocols with and without BDL-APT in steady state by changing the network delay (the delay at the network emulator). **Figure 11** shows the aggregated IP-SAN goodput when the network emulator delay was varied between 20–100 [ms] while the bandwidth of the network emulator was fixed at 900 [Mbit/s]. For comparison, the steady-state IP-SAN goodput with 32, 64, and 128 fixed IP-SAN sessions are also plotted. Figure 11 (a) shows the aggregated NBD goodput when the bottleneck link delay varies. Similarly, Fig. 11 (b) shows the aggregated GNBD goodput, and Fig. 11 (c) shows the aggregated iSCSI goodput.

Figure 11 shows that the IP-SAN goodput drops rapidly as the delay increases when the number of active IP-SAN sessions is not fixed at an optimal value. This result shows that NBD goodput is the highest when the number of NBD sessions is fixed at 32 with 20–40 [ms] delay, at 64 with 60–80 [ms] delay, and at 128 with 100 [ms] delay. This result also shows that GNBD goodput and iSCSI goodput are the highest when the numbers of GNBD sessions and iSCSI sessions are fixed at 64 with 20–40 [ms] delay and at 128 with large delay. In particular, when BDL-APT is not



(a) NBD (b) GNBD (c) iSCSI

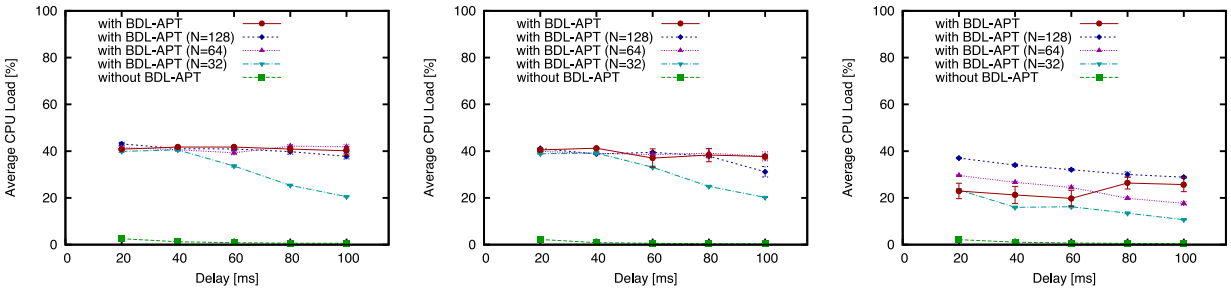
Fig. 10 Bottleneck link bandwidth vs. IP-SAN goodput.



(a) NBD (b) GNBD (c) iSCSI

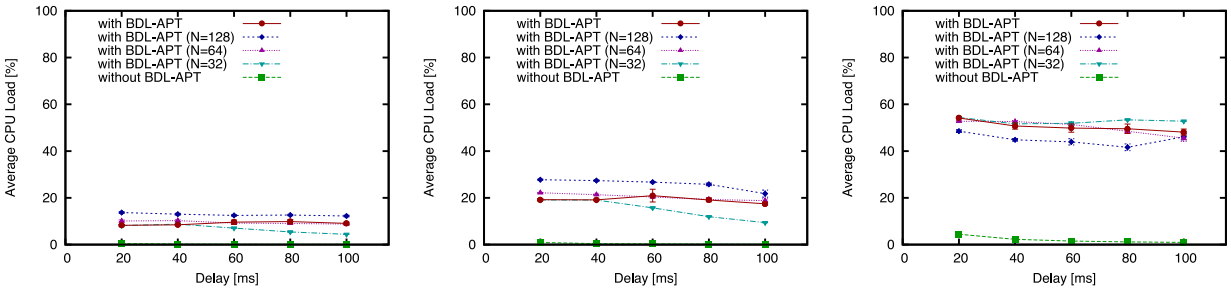
Fig. 11 Bottleneck link delay vs. IP-SAN goodput.

© 2013 Information Processing Society of Japan



(a) NBD (b) GNBD (c) iSCSI

Fig. 12 The average CPU load of the IP-SAN client.



(a) NBD (b) GNBD (c) iSCSI

Fig. 13 The average CPU load of the IP-SAN storage.

used, the aggregated IP-SAN goodput was less than 20 [Mbit/s] regardless of IP-SAN protocol with 100 [ms] delay, despite the 900 [Mbit/s] network bandwidth.

Figure 11 shows that BDL-APT improves goodput regardless of bottleneck-link delay and IP-SAN protocol. We found that BDL-APT resolves TCP performance degradation which causes IP-SAN performance degradation and improves goodput in a long-fat network. In particular, when the NBD and GNBD protocols are used, the network bandwidth can be mostly used up regardless of bottleneck link delay. Conversely, goodput degrades as the bottleneck link delay increases when the iSCSI protocol is used. This is because in long-fat networks there are other factors that degrade the performance of the iSCSI protocol besides the performance degradation of TCP. When the bottleneck link delay is large and the iSCSI protocol is used, goodput also degrades even more than with fixed values. When parallel data transfer was performed using the iSCSI protocol, goodput was unstable in some cases. This may have caused failed adjustments, and requires further investigation. However, goodput significantly increased even when BDL-APT was used with iSCSI, and we think that practical applications will perform satisfactorily. To realize further goodput gains, custom iSCSI protocol tuning is required.

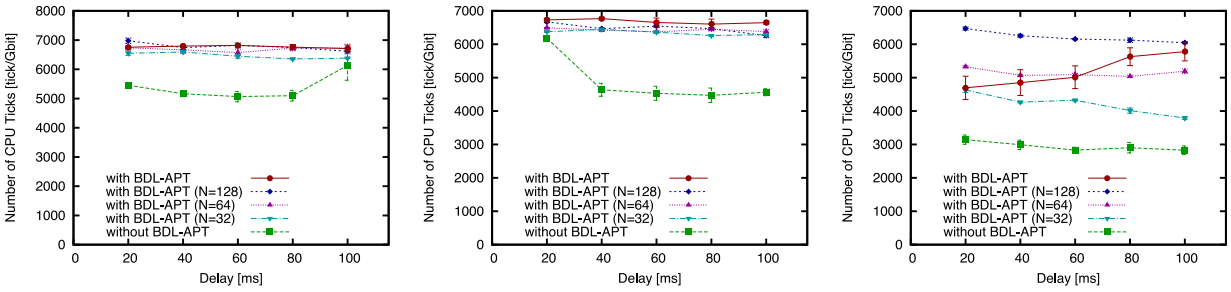
6.5 CPU Processing Load

As we have explained In Section 4, BDL-APT parallelizes data transfer at IP-SAN session level, rather than at TCP connection level. Establishing multiple (and sometimes many) IP-SAN sessions may cause a significant amount of CPU processing overhead. In this section, we therefore evaluate the amour of CPU processing overhead caused by the introduction of BDL-APT compared with vanilla NBD, GNBD, and iSCSI. The average CPU processing load during data transfer was measured by monitoring `/proc/stat` in the Linux kernel. Numbers that can be obtained through `/proc/stat` are *cumulative number of ticks*, each of which corresponds to the amount of time, measured in units of `USER_HZ`, that the system spent in idle, running, I/O-request, or interrupt states of the Linux kernel [25].

Figures 12 and **13** show the average CPU load of the IP-SAN client (i.e., receiver) and the IP-SAN storage (i.e., sender), respectively. In both figures, the average CPU loads with and without BDL-APT are plotted for NBD, GNBD, and iSCSI protocols. Similar to Section 6.4, the bandwidth of the network emulator was set to 900 [Mbit/s] and the bottleneck ilk delay was varied from 20 to 100 [ms]. For comparison, the average CPU loads with 32, 64, and 128 fixed IP-SAN sessions are also plotted. Not surprisingly, these figures show that the average CPU loads of both the IP-SAN client and the IP-SAN storage with BDL-APT are significantly higher than that without BDL-APT and those with BDL-APT with the fixed number of active IP-SAN sessions. This is simply because the goodput with BDL-APT is much higher than that without BDL-APT or those with BDL-APT with the fixed number of active IP-SAN sessions (see Fig. 11). As can be seen from Fig. 11, the goodputs with and without BDL-APT are comparable when the bottleneck link delay is very small. For instance, when the bottleneck link delay is 20 [ms] and NBD protocol is used, the goodputs with BDL-APT and with BDL-APT ($N = 32$) are approximately 800 [Mbit/s] (see Fig. 11). In this case, the average CPU load with BDL-APT and with BDL-APT ($N = 32$) are almost identical, which indicates that multiple IP-SAN sessions causes non-negligible CPU processing overhead but the effect of APT (Automatic Parallelism Tuning) algorithm in BDL-APT on the CPU processing load is negligible.

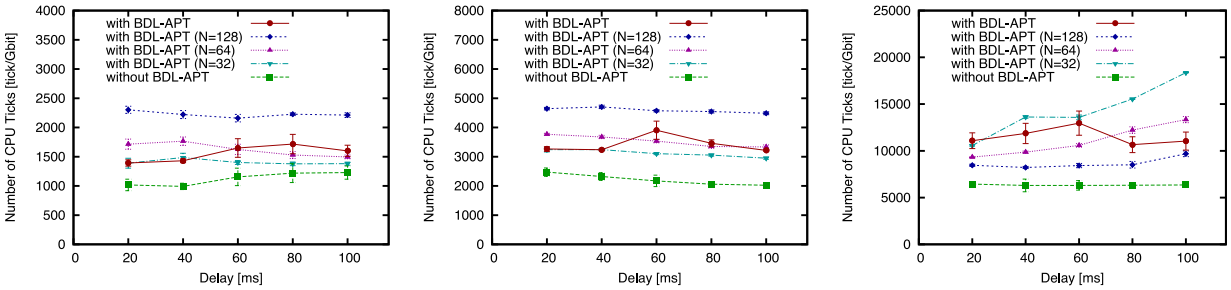
It is not surprising that aggregating multiple IP-SAN sessions results in higher goodput than a single IP-SAN session at the cost of non-negligible CPU processing overhead. But it is still unclear how efficient the aggregation of multiple IP-SAN sessions

© 2013 Information Processing Society of Japan



(a) NBD (b) GNBD (c) iSCSI

Fig. 14 The number of CPU ticks consumed for a successful bit transfer on the IP-SAN client.



(a) NBD (b) GNBD (c) iSCSI

Fig. 15 The number of CPU ticks consumed for a successful bit transfer on the IP-SAN storage.

is compared with a single IP-SAN session in terms of *the amount of CPU processings per a successful bit transfer*.

We therefore calculated the number of CPU ticks consumed for a successful bit transfer (see **Figs. 14** and **15**). These figures show the number of CPU ticks consumed for a successful bit transfer, which is defined as the total number of CPU ticks consumed during a file transfer divided by the size of the transferred file, with NBD, GNBD, and iSCSI, respectively.

These figures show introduction of multiple IP-SAN sessions increases the amount of CPU processing per a successful bit transfer. Note that different protocols, NBD, GNBD, and iSCSI, show different tendencies. Namely, the overhead of multiple IP-SAN sessions in NBD is minimal (i.e., approximately 20–30% in both the IP-SAN client and the IP-SAN storage). On the contrary, the overhead of multiple IP-SAN sessions in iSCSI reaches approximately 100% in both the IP-SAN client and the IP-SAN storage. Such a difference should be caused by the difference in IP-SAN protocol implementations (in particular, multiple IP-SAN sessions management), which implies that current IP-SAN protocol implementations could be improved to make it more scalable to the number of active IP-SAN sessions.

7. Conclusion

In this paper, we proposed *BDL-APT*, a mechanism that maximizes the goodput of heterogeneous IP-SAN protocols in long-fat networks and that requires no modification to IP-SAN storage devices. We implemented BDL-APT as a layer of the MD driver, one of the major software RAID implementations included in the Linux kernel. We evaluated the performance of BDL-APT with heterogeneous IP-SAN protocols (NBD, GNBD and iSCSI) in a long-fat network.

We found that BDL-APT improved IP-SAN goodput regardless of the IP-SAN protocol used. We showed that network bandwidth could be mostly used up in long-fat networks when the NBD and GNBD protocols were used. In long-fat networks, we found that BDL-APT resolves TCP performance degradation which causes IP-SAN performance degradation but does not maximize IP-SAN goodput under the iSCSI protocol. This is because in long-fat networks there are other factors that degrade the performance of the iSCSI protocol besides the performance degradation of TCP.

In the network environment with little traffic changes, such as the leased line, our BDL-APT accelerates a data transfer sufficiently. Since storage networks are mainly used on leased lines now, we believe that increase in the speed of storage networks using the leased line is sufficient. However, we expect that the future storage networks might be used on the networks with intense traffic changes, such as the Internet. High speed data transfer for such a network is also required.

Therefore, investigation of the high speed data transfer technology for networks where traffic changes dynamically is one of our future research directions. Specifically, we show clearly robustness of BDL-APT to traffic changes. Moreover, to achieve high goodput constantly against changes in traffic, we will develop the fast adjustment of the number of sessions.

References

[1] Sarkar, P. and Voruganti, K.: IP storage: The challenge ahead, *Proc. 19th IEEE Symposium on Mass Storage Systems*, pp.35–42 (2002).

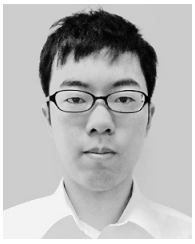
[2] Wang, P., Gilligan, R.E., Green, H. and Raubitschek, J.: IP SAN – from iSCSI to IP-addressable Ethernet disks, *Proc. 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, pp.189–193 (2003).

[3] Yang, H.: Fibre channel and IP SAN integration, *Proc. 12th NASA Goddard/21st IEEE Conference on Mass Storage Systems and Technologies*, pp.101–115 (2004).

[4] Aiken, S., Grunwald, D., Pleszkun, A.R. and Willeke, J.: A performance analysis of the iSCSI protocol, *Proc. 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, pp.123–134 (2003).

© 2013 Information Processing Society of Japan

[5] Breuer, P.T., Lopez, A.M. and Ares, A.G.: The network block device, *Linux Journal*, Vol.73 (2000).
[6] GNBD Project Page, available from ⟨http://sourceware.org/cluster/gnbd/⟩.
[7] Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M. and Zeidner, E.: Internet Small Computer Systems Interface (iSCSI), Request for Comments (RFC) 3720 (2004).
[8] Rajagopal, M., Rodriguez, E. and Weber, R.: Fibre Channel over TCP/IP (FCIP), Request for Comments (RFC) 3821 (2004).
[9] Monia, C. et al.: iFCP - A Protocol for Internet Fibre Channel Storage Networking, Request for Comments (RFC) 4172 (2005).
[10] Ng, W. et al.: Obtaining high performance for storage outsourcing, *Proc. 1st USENIX Conference on File and Storage Technologies*, pp.145–158 (2002).
[11] Lu, Y., Farrukh, N. and Du, D.H.C.: Simulation study of iSCSI-based storage system, *Proc. 12th NASA Goddard/21st IEEE Conference of Mass Storage Systems and Technologies*, pp.101–110 (2004).
[12] Kancherla, B.K., Narayan, G.M. and Gopinath, K.: Performance evaluation of multiple TCP connections in iSCSI, *Proc. 24th IEEE Conference on Mass Storage Systems and Technologies*, pp.239–244 (2007).
[13] Inoue, F., Ohsaki, H., Nomoto, Y. and Imase, M.: On maximizing iSCSI throughput using multiple connections with automatic parallelism tuning, *Proc. 5th IEEE International Workshop on Storage Network Architecture and Parallel I/Os*, pp.11–16 (2008).
[14] Ito, T., Ohsaki, H. and Imase, M.: GridFTP-APT: Automatic parallelism tuning mechanism for GridFTP in long-fat networks, *IEICE Trans. Comm.*, Vol.E91-B, pp.3925–3936 (2008).
[15] Icaza, M., Molnar, I. and Oxman, G.: The linux RAID-1, 4, 5 code, Linux Expo (1997).
[16] Yang, Q.K.: On performance of parallel iSCSI protocol for networked storage systems, *Proc. 20th International Conference on Advanced Information Networking and Applications*, pp.629–636 (2006).
[17] Oguchi, M. et al.: Performance improvement of iSCSI remote storage access, *Proc. 4th International Conference on Ubiquitous Information Management and Communication*, pp.232–338 (2010).
[18] Soltis, S.R., Ruwart, T.M. and O'Keefe, M.T.: The global file system, *Proc. 5th NASA Goddard Conference on Mass Storage Systems and Technologies*, pp.319–342 (1996).
[19] Lu, Y. and Du, D.H.C.: Performance study of iSCSI-based storage subsystems, *IEEE Communications Magazine*, Vol.41, No.8, pp.76–82 (2003).
[20] Gauger, C.M., Kohn, M., Gunreben, S., Sass, D. and Perez, S.G.: Modeling and performance evaluation of iSCSI storage area networks over TCP/IP-based MAN and WAN networks, *Proc. 2nd International Conference on Broadband Networks*, pp.915–923 (2005).
[21] Flouris, M.D. and Bilas, A.: Violin: A framework for extensible block-level storage, *Proc. 22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies*, pp.128–142 (2005).
[22] Rizzo, L.: Dummynet: A simple approach to the evaluation of network protocols, *ACM Comput. Comm. Review*, Vol.27, No.1, pp.31–41 (1997).
[23] Open iSCSI project, available from ⟨http://www.open-iscsi.org/⟩.
[24] The iSCSI enterprise target project, available from ⟨http://iscsitarget.sourceforge.net/⟩.
[25] proc(5) - linux manual page, available from ⟨http://www.kernel.org/doc/man-pages/online/pages/man5/proc.5.html⟩.



Takamichi Nishijima received B.S. in Information Science and M.S. in Master of Information Science degrees from Osaka University in 2009 and 2011, respectively. He has been a graduate student of the Graduate School of Information Science and Technology, Osaka University since April 2011. He is a member of IEEE and IEICE.



Hiroyuki Ohsaki received his M.E. degree in Information and Computer Sciences from Osaka University, Osaka, Japan, in 1995. He also received his Ph.D. degree from Osaka University, Osaka, Japan, in 1997. He is currently an Associate Professor at the Department of Information Networking, Graduate School of Information Science and Technology, Osaka University, Japan. His research work is in the area of traffic management in high-speed networks. He is a member of IEEE, IEICE, and IPSJ.



Makoto Imase receied his B.E. and M.E. degrees in Information Engineering from Osaka University in 1975 and 1977, respectively. He received his D.E. degree from Osaka University in 1986. From 1977 to 2001, he was engaged Nippon Telegraph and Telephone Corporation (NTT). From 2002 to 2012 He was a Professor of Graduate School of Information Science and Technology at Osaka University. He has been a Vice President of National Institute of Information and Communications Technology (NICT) since April 2012. He is also a Fellow of IPSJ and a Emeritus Professor of Osaka University. His research interests are in the area of information networks, distributed systems and graph theory. He is a member of IPSJ and IEICE.

© 2013 Information Processing Society of Japan