A Network Bandwidth Computation Technique for IP Storage with QoS Guarantees

Young Jin Nam[1], Junkil Ryu[1], Chanik Park[1], and Jong Suk Ahn[2]

[1] Department of Computer Science and Engineering
Pohang University of Science and Technology/PIRL
Kyungbuk, Republic of Korea
{yjnam,lancer,cipark}@postech.ac.kr
[2] Department of Computer Engineering
Dongguk University, Seoul, Republic of Korea
jahn@dgu.ac.kr

Abstract. IP storage becomes more commonplace with the prevalence of the iSCSI (Internet SCSI) protocol that enables the SCSI protocol to run over the existing IP network. Meanwhile, storage QoS that assures a required storage service for each storage client has gained in importance with increased opportunities for multiple storage clients to share the same IP storage. Considering the existence of other competing network traffic in IP network, we have to provide storage I/O traffic with guaranteed network bandwidth. Most importantly, we need to calculate the required network bandwidth to assure a given storage QoS requirement between a storage client and IP storage. This paper proposes a network bandwidth computation technique that not only accounts for the overhead caused by the underlying network protocols, but also guarantees the minimum data transfer delay over the IP network. Performance evaluations with various I/O workload patterns on our IP storage testbed verify the correctness of the proposed technique; that is, allocating a part (0.6–20%) of the entire network bandwidth can assure the given storage QoS requirements.

1 Introduction

Storage Area Networks (SAN), such as Fiber Channel and Gigabit Ethernet, have enabled a plethora of storage systems to be maintained as a storage pool, resulting in reduced total cost of ownership, effective storage resource management, etc. Such SAN-based storage systems are advantageous in terms of scalability and configurability, compared with SCSI bus-based storage systems. Accordingly, a few data transmission protocols have newly emerged to support the SAN environment. Fiber Channel protocol (FCP) is developed for FC-based SAN, and iSCSI recently ratified by Internet Engineering Task Force is made for IP SAN. A main advantage of iSCSI is that the iSCSI can operate on standard network components, such as Ethernet [1]; that is, it exploits existing features and tools that have been developed for the IP network. Thus, this paper focuses on the storage environment using IP-based SAN (IP storage), where storage devices are attached to IP networks, and storage clients communicate with the storage devices via the iSCSI protocol [1]. An initiator-mode iSCSI protocol runs on the storage client, whereas a target-mode iSCSI protocol operates on the IP storage. Note that the traditional SCSI protocol operates on top of the iSCSI protocol layer that transmits a given SCSI command to its associated IP storage over the IP network.

With the advance in storage technologies in terms of storage space and I/O performance, the chances increase that multiple storage clients share the same storage. A different storage client may require a different storage service, called storage Quality of Service (QoS); that is, each storage client requires receiving a guaranteed storage service, independently of the status of the I/O services in other storage clients. Unfortunately, the storage itself does not contain any feature of assuring storage QoS. As a result, recent research efforts [2,3] try to add the QoS feature to various types of storage systems. However, notice that the previous research emphasizes the QoS issue only within the storage system, whereas it is assumed that the SAN itself has no QoS issues. Note that, FC-based SAN is used only for storage I/O traffic. IP storage, however, transmits its data over the IP network, where storage I/O traffic is likely to coexist with the other network traffic. Considering this situation leads us to preserve an amount of network bandwidth for the storage I/O traffic between a storage client and its associated IP storage to avoid any probable interference with the other network traffic. A naive approach is to allocate the full network bandwidth (or separate dedicated IP network) that is large enough to serve the storage I/O traffic with QoS guarantee for a pair of a storage client and its associated IP storage. However, it can be easily inferred that this approach ends up being with under-utilization of IP network resources, even though it can certainly guarantee a given storage QoS requirement. By contrast, unless enough network bandwidth resides between the storage client and the IP storage, the storage QoS requirement is no longer guaranteed with lower I/O throughput (I/O requests per seconds) and higher response time due to increased data transfer delays.

This paper emphasizes the problem of computing the required network bandwidth to meet a given storage QoS requirement. It proposes a network bandwidth computation technique that not only accounts for overhead caused by the underlying network protocols, but also guarantees the minimum data transfer delay over the IP network. In the case of FC-based SAN environments, Ward *et. al.* in [4] proposed a scheme to automatically design an FC-based SAN that not only serves a given set of storage I/O traffic between storage clients and storage systems, but also minimizes the system cost.

2 Problem Description

We begin by defining a few notations to be used throughout the paper. C_i and Q_i represent the storage client i and its storage QoS requirement, respectively. Generally, the storage QoS requirement is defined as $Q_i = \{f_i, iops_i, sz_i, s_i, rt_i\}$.

The notation of f_i represents the ratio of read I/O requests. In addition, $iops_i$, sz_i, and rt_i represent the number of I/O requests per second (briefly IOPS), an average I/O request size, and an average response time requested from C_i, respectively. I/O access pattern is random with $s_i = 0$ and purely sequential with $s_i = 1$. Note that our network bandwidth computation does not depend on the value of s_i. We denote with $b_i^{c \to s}$ and $b_i^{s \to c}$ the network bandwidth allocated for the direction from C_i and its associated IP storage and for its opposite direction, respectively. Then, the problem that this paper will solve can be described as follows: *Compute $b_i^{c \to s}$ and $b_i^{s \to c}$ that satisfy the given Q_i for the storage client C_i and its associated IP storage.* Note that we assume that the storage resources except for network resources (bandwidth) have been appropriately reserved to satisfy the storage QoS requirement Q_i.

3 The Proposed Technique

To begin, we will explain the protocol layering for iSCSI protocol and the specific behavior of the iSCSI protocol for read and write I/O requests. The iSCSI protocol data unit (PDU) consists of a 48-byte iSCSI header and iSCSI data of a variable length. The maximum iSCSI data length relies on the types of underlying Ethernet cards. Typically, it ranges from 1,394 through 8,894 bytes. The TCP/IP headers and Ethernet header respectively occupy 40 bytes and 18 bytes.

Figure 1 presents the protocol behaviors for read and write I/O requests. In the case of the read I/O request, as shown in Figure 1(a), the storage client sends the `READ` SCSI command to the IP storage. Next, after reading the requested data from its internal disk drives, the IP storage transmits the data to the storage client in the `DATA_IN` phase. Note that the data are to be fragmented into smaller pieces according to the maximum iSCSI data length. Finally, the IP storage sends the response message to the storage client. The iSCSI

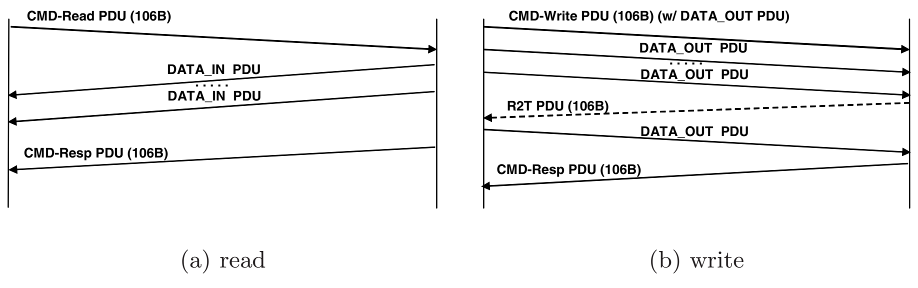

Fig. 1. iSCSI protocol behaviors for read and write I/O requests: (a) read and (b) write

protocol behavior for the write I/O request is more complicated than that of the read I/O request because of free buffer management and performance optimization techniques like immediate/unsolicited data transmission. The storage client sends the `WRITE` SCSI command to the IP storage. Unless the data size is greater than the maximum iSCSI data length, it is transmitted along with the write SCSI command. This results in collapsing the `COMMAND` phase and the `DATA_OUT` phase [1]. It is called immediate data transmission [1]. If the data size is greater than `FirstBurstLength`, the storage client transfers the data of the first `FirstBurstLength` bytes to the IP storage without receiving the `Ready to Transfer` (R2T) message from the IP storage that is used to secure free buffer space to store the write data. This process is called unsolicited data transmission [1]. Note that the immediate data transmission is combined with the unsolicited data transmission. Afterwards, the storage client transfers data only when it receives the R2T message from the IP storage. It is called solicit data transmission [1].

In what follows, we derive a set of equations to compute the required network bandwidth to meet a given storage QoS requirement. We start by computing the amount of data transfer, including the underlying protocol overhead for given read and write I/O requests of size s_i. We denote with $D_r^{c\rightarrow s}(sz_i)$ the amount of data transfer from the storage client to the IP storage for the read I/O request of size sz_i, and $D_r^{s\rightarrow c}(sz_i)$ for the opposite direction. From the protocol behavior for the read I/O request, as shown in Figure 1(a), we can easily obtain $D_r^{c\rightarrow s}(sz_i)$ and $D_r^{s\rightarrow c}(sz_i)$ as follows:

$$D_r^{c\rightarrow s}(sz_i) = ov_{prot}, \tag{1}$$

$$D_r^{s\rightarrow c}(sz_i) = 2ov_{prot} + \lfloor \frac{sz_i}{\hat{S}_f} \rfloor S_f + sz_i \textbf{ mod } \hat{S}_f, \tag{2}$$

where $\hat{S}_f = S_f - ov_{prot}$. The notation of ov_{prot} represents the 106-byte protocol overhead caused by the Ethernet header, the TCP/IP header, and the iSCSI header. The notation of S_f represents the underlying Ethernet frame size.

Next, we calculate the amount of data transfer for the write I/O request. To begin, the storage client sends the agreed-upon amount of data (unsolicited data transmission) to IP storage without having an R2T message. We assume that the behavior of each data transfer follows that of the solicit data transmission because the network traffic under our consideration is heavy enough. However, if the I/O request size is not greater than the maximum iSCSI data length, the associated data is delivered to the IP storage by using the immediate data transmission. As with the read I/O request, we denote with $D_w^{c\rightarrow s}(sz_i)$ the amount of data from the storage client to the IP storage for the write I/O request, and $D_w^{s\rightarrow c}(sz_i)$ for the opposite direction. Based on the iSCSI protocol behavior for the write I/O request, as shown in Figure 1(b), $D_w^{c\rightarrow s}(sz_i)$ and $D_w^{s\rightarrow c}(sz_i)$ can be obtained as follows:

$$D_w^{c\rightarrow s}(sz_i) = \begin{cases} ov_{prot} + sz_i & \text{if } sz_i \leq \hat{S}_f \\ ov_{prot} + \lfloor \frac{sz_i}{\hat{S}_f} \rfloor S_f + sz_i \textbf{ mod } \hat{S}_f & \text{otherwise} \end{cases}, \tag{3}$$

$$D_w^{s\to c}(sz_i) = \begin{cases} ov_{prot} & \text{if } sz_i \leq \texttt{FirstBurstLength} \\ 2ov_{prot} & \text{otherwise} \end{cases}. \tag{4}$$

If the data size is not greater than the maximum iSCSI data length of $\hat{S}_f$, $D_w^{c\to z}(sz_i)$ is $ov_{prot} + sz_i$. Otherwise, it includes $\lfloor \frac{sz_i}{\hat{S}_f} \rfloor$ times of the maximum Ethernet frame size of S_f and the remaining data with the protocol overhead of ov_{prot}. As for $D_w^{s\to c}(sz_i)$, if the data size is not greater than `FirstBurstLength`, $D_w^{s\to c}(sz_i)$ is equal to the size of the iSCSI response message. Otherwise, it becomes two times of ov_{prot}, because the R2T message is also delivered to the storage client as well as the iSCSI response message.

Next, we define with $\hat{b}_i^{c\to s}$ the average amount of data transfer including the underlying protocol overhead from the storage client to the IP storage for the sz_i sized I/O request. In addition, we define with $\hat{b}_i^{s\to c}$ for the opposite direction. From Equation (1)–(4), we can obtain $\hat{b}_i^{c\to s}$ and $\hat{b}_i^{s\to c}$ as follows:

$$\hat{b}_i^{c\to s} = \{f_i D_r^{c\to s}(sz_i) + (1-f_i)D_w^{c\to s}(sz_i)\}iops_i, \tag{5}$$

$$\hat{b}_i^{s\to c} = \{f_i D_r^{s\to c}(sz_i) + (1-f_i)D_w^{s\to c}(sz_i)\}iops_i. \tag{6}$$

The network bandwidth allocation with $\hat{b}_i^{c\to s}$ and $\hat{b}_i^{s\to c}$ is expected to assure the requested maximum storage bandwidth derived by multiplying sz_i and $iops_i$. However, they may not guarantee the demanded response time of rt_i. For example, notice that Equation (5) and (6) result in a lower network bandwidth with a smaller $sz_i \cdot iops_i$. This implies that the chances increase that each I/O request of size sz_i experiences a longer transmission delay on IP network that is most likely to entail a violation of the demanded response time.

Thus, we introduce the minimum network bandwidth to assure the demanded response time of rt_i. We denote with $m_i^{c\to s}$ and $m_i^{s\to c}$ the minimum network bandwidth for each direction. The values of $m_i^{c\to s}$ and $m_i^{s\to c}$ are determined such that the transmission delay of each I/O request are not greater than $\alpha_i \cdot rt_i$, where $0 < \alpha_i < 1$. Usually, α_i is determined according to the marginal response time to meet rt_i in the phase of designing the associated IP storage [2,5,3]. For example, if the IP storage is designed to assure 15msec for given $rt_i = 20$msec, the values of α_i can range from 0 through 0.25. We compute $m_i^{c\to s}$ and $m_i^{s\to c}$ from a simple relationship that the expected transmission delay is inversely proportional to the allocated network bandwidth without accounting for the effects of traffic congestion control, IP routing, the transmission buffer size, TCP retransmission, etc. The $m_i^{c\to s}$ and $m_i^{s\to c}$ are written as follows:

$$m_i^{c\to s} = \frac{max\{\lfloor f_i \rfloor D_r^{c\to s}(sz_i), \lfloor 1-f_i \rfloor D_w^{c\to s}(sz_i)\}}{\alpha_i rt_i}, \tag{7}$$

$$m_i^{s\to c} = \frac{max\{\lfloor f_i \rfloor D_r^{s\to c}(sz_i), \lfloor 1-f_i \rfloor D_w^{s\to c}(sz_i)\}}{\alpha_i rt_i}. \tag{8}$$

By denoting with $b_i^{c\to s}$ and $b_i^{s\to c}$ the network bandwidth for each direction required to guarantee a given Q_i, we finally have $b_i^{c\to s}$ and $b_i^{s\to c}$ as follows:

$$b_i^{c\to s} = max\{\hat{b}_i^{c\to s}, m_i^{c\to s}\}, \tag{9}$$

$$b_i^{s\to c} = max\{\hat{b}_i^{s\to c}, m_i^{s\to c}\}. \tag{10}$$

4 Performance Evaluations

We set up an experimental testbed for IP storage to evaluate the performance of the proposed technique.We use two Intel Pentium III based desktops for the storage client and the IP storage. Both systems are attached to a Gigabit IP network via Gigabit Ethernet cards and a switch. Assume that no other traffic exists between the two systems. The maximum size of the Ethernet frame is 1500 bytes. The Linux kernel 2.4.18 works on top of the storage client and IP storage. The storage client includes the initiator-mode iSCSI driver developed by the University of New Hampshire [6], and the IP storage contains the target-mode iSCSI driver for operating the iSCSI protocol. The network bandwidth is controlled by Token Bucket Filtering (TBF) [7]. We believe that this type of end-to-end traffic control works because no other traffic exists between the storage client and the IP storage.

Table 1 shows the measured $iops_i$ and rt_i for various QoS requirements of Q_1–Q_4 when the amount of network bandwidth is computed by the proposed technique. In addition, they are compared with the results with the full network bandwidth for the same QoS requirements. We denote with `prot-rt` and `full` for the proposed technique and the full network bandwidth, respectively. Each

Table 1. Results of $iops_i$ and rt_i for the various QoS requirements of Q_1–Q_8 by the proposed technique (`prot-rt`) and the full network bandwidth allocation (`full`)

QoS	Technique	Toward IP storage	Toward client	$iops_i$	rt_i
Q_1	`prot-rt`	0.05MB/s	0.60MB/s	84(100%)	9.74ms
	`full`	100.00MB/s	100MB/s	82(100%)	9.70ms
Q_2	`prot-rt`	1.84MB/s	0.17MB/s	265(99%)	2.59ms
	`full`	100.00MB/s	100MB/s	266(100%)	2.58ms
Q_3	`prot-rt`	0.03MB/s	19.19MB/s	77(100%)	15.66ms
	`full`	100.00MB/s	100MB/s	76(100%)	15.34ms
Q_4	`prot-rt`	15.67MB/s	0.02MB/s	211(100%)	20.28ms
	`full`	100.00MB/s	100MB/s	208(100%)	21.54ms

of the storage QoS requirements is represented as follows: $Q_1 = \{f_1 = 1, iops_1 = 82, sz_1 = 1KB, s_1 = 0, rt_1 = 10ms\}$, $Q_2 = \{f_2 = 0, iops_2 = 266, sz_2 = 1KB, s_2 = 0, rt_2 = 3ms\}$, $Q_3 = \{f_3 = 1, iops_3 = 76, sz_3 = 64KB, s_3 = 0, rt_3 = 18ms\}$, and $Q_4 = \{f_4 = 0, iops_4 = 208, sz_4 = 64KB, s_4 = 0, rt_4 = 22ms\}$. The first two requirements are for the small read and write I/O requests as in OLTP applications, and the others are for the large read and write I/O requests in scientific applications [5]. Since the network bandwidth allocation is independent of storage access patterns, we assume that all the storage access patterns are purely random. In addition, the demanded IOPS of $iops_i$ and its associated response time of rt_i are configured by injecting a set of I/O workload patterns and measuring each performance. Note that the $iops_i$ and rt_i of the QoS requirements fall into neither too light traffic that makes the IP storage mostly idle nor too heavy traffic that overloads the system. The results reveal that the allocation of only 0.6–20% of the full network bandwidth computed by the proposed technique can meet the given storage QoS requirements. Observe that the proposed technique can compute an appropriate amount of network bandwidth to provide the same quality of storage service as the case when the full network bandwidth is allocated. The percentage values in the $iops_i$ column represent the percentage of the measured IOPS with respect to the demanded IOPS.

Table 2 shows the measured $iops_i$ and rt_i for the QoS requirements of Q_1–Q_4 for the `prot` and `naive` techniques. The `prot` technique computes the required network bandwidth only by considering the underlying protocol overhead, as shown in Equation (5)–(6); that is, $\hat{b}_i^{c\rightarrow s}$ and $\hat{b}_i^{s\rightarrow c}$. The `naive` technique simply calculates the network bandwidth by multiplying $iops_i$ and sz_i. It assigns the same bandwidth for each direction. As expected, the `naive` technique cannot guarantee even demanded $iops_i$, because it does not account for the underlying protocol overhead in Ethernet, TCP/IP, and iSCSI layers at all. Notice that

Table 2. Results of $iops_i$ and rt_i for the various QoS requirements of Q_1–Q_8 by the technique using Equation (5)–(6) (`prot`) and the technique using $iops_i{\cdot}sz_i$ (`naive`)

QoS	Technique	Toward IP storage	Toward client	$iops_i$	rt_i
Q_1	`naive`	0.06MB/s	0.06MB/s	64(77%)	28.82ms
	`prot`	0.01MB/s	0.12MB/s	80(96%)	12.99ms
Q_2	`naive`	0.26MB/s	0.26MB/s	238(89%)	9.25ms
	`prot`	0.29MB/s	0.03MB/s	262(98%)	3.54ms
Q_3	`naive`	4.78MB/s	4.78MB/s	69(91%)	22.31ms
	`prot`	0.01MB/s	5.19MB/s	74(96%)	19.33ms
Q_4	`naive`	13.06MB/s	13.06MB/s	194(93%)	25.13ms
	`prot`	14.08MB/s	0.02MB/s	208(99%)	22.05ms

the smaller sized I/O request causes higher protocol overhead, as expected from Equation (1)–(4). In addition, the results show that read I/O requests creates more protocol overhead than write I/O requests. It can be expected mainly from Equation (2)–(3). By contrast, the `prot` technique guarantees more than 96% of the required $iops_i$. However, it does not satisfy the demanded response time of rt_i because of the transmission delay on IP network. Recall that, as shown in Table 1, the proposed `prot-rt` technique can meet both the $iops_i$ and the rt_i by effectively allotting the network bandwidth for each direction between the storage client and the IP storage.

5 Conclusion and Future Work

This paper addressed the problem of effectively allocating network bandwidth to assure a given QoS requirement for IP storage. It defined a specification of the storage QoS requirement, and it proposed a technique to compute the demanded network bandwidth to meet the storage QoS requirement that not only accounts for the overhead caused by the underlying network protocols, but also guarantees the minimum data transfer delay over the IP network. Performance evaluations with various I/O workload patterns on our IP storage testbed verified the correctness of the proposed technique; that is, allocating a part (0.6–20%) of the entire network bandwidth can assure the given storage QoS requirements. Currently, we have been revising Equation (9)–(10) to additionally account for real-world I/O workload patterns featured by self-similarity and the condition of traffic congestion.

Acknowledgments

The authors would like to thank the Ministry of Education of Korea for its financial support toward the Electrical and Computer Engineering Division at POSTECH through its BK21 program. This research was also supported in part by the Korea Science and Engineering Foundation (KOSEF) under grant number R01-2003-000-10739-0 and by HY-SDR IT Research Center.

References

1. Meth, K., Satran, J.: Design of the iscsi protocol. In: Proceedings of the Mass Storage Systems and Technologies/20th IEEE/11th NASA Goddard Conference. (2003)
2. Nam, Y.J.: Dynamic Storage QoS Control for Storage Cluster and RAID Performance Enhancement Techniques. Ph.D Dissertation, POSTECH (2004)
3. Anderson, E., Hobbs, M., Keeton, K., Spence, S., Uysal, M., Veitch, A.: Hippodrome: Running rings around storage administration. In: Proceedings of Conference on File and Storage Technologies. (2002)
4. Ward, J., O'Sullivan, M., Shahoumian, T., Wilkes, J.: Appia: Automatic storage area network design. In: Proceedings of Conference on File and Storage Technologies. (2002)
5. Alvarez, G., Borowsky, E., Go, S., Romer, T., Becker-Szendy, R., Golding, R., Merchant, A., Spasojevic, M., Veitch, A., Wikes, J.: Minerva: An automated resource provisioning tool for large-scale storage systems. ACM Transactions on Computer Systems **19** (2001) 483–518
6. UNH: iscsi reference implementation. http://www.iol.unh.edu/consortiums/iscsi/ (2004)
7. Hurbert, B.: Linux advanced routing & traffic control. http://lartc.org/howto (2003)