Performance Evaluation of Distributed iSCSI RAID

Xubin (Ben) He, Praveen Beedanagari, Dan Zhou
Department of Electrical and Computer Engineering
Tennessee Technological University
Cookeville, TN 38505, USA
{hexb, prb0779, dz1643}@tntech.edu

Abstract

iSCSI is a newly emerging protocol with the goal of implementing the storage area network (SAN) technology over TCP/IP, which brings economy and convenience whereas it also raises performance and reliability issues. This paper identifies the performance bottleneck of iSCSI, and then proposes a distributed iSCSI RAID to improve the performance by stripping data among iSCSI targets (*S-iRAID*) and improve the reliability by using rotated parity for data blocks (*P-iRAID*). Numerical results using popular benchmark have shown dramatic performance gain. *S-iRAID* improves the average throughput from 11.7MB/s to 46.1MB/s by striping data among only three iSCSI targets. *S-iRAID* and *P-iRAID* can speed up the iSCSI performance by a factor of up to 6.6 and 2.17, respectively.

1. Introduction

iSCSI [1,15,18,19] is a newly emerging technology with the goal of implementing the storage area networks (SAN) [9,21] technology over Internet infrastructure, which brings economy and convenience whereas it also raises performance and reliability issues. On a typical software iSCSI implementation [11], we have observed one iSCSI target is not enough to saturate the network and the iSCSI initiator. We also notice that for each iSCSI operation, there will be at least 4 socket communications between the iSCSI initiator and target [8]. All these socket communications may cause much overhead which limits the iSCSI performance. In addition to the performance issue, researchers also concern the data reliability on the iSCSI targets.

RAID (Redundant Array of Independent Disks) [9] is a known, mature technique to improve performance and reliability of disk I/O through parallism and redundancy. This paper introduces a technique to stripe data among several iSCSI targets in a similar way to RAID. Since it's a distributed RAID [2,20] across several nodes (iSCSI targets), we name it iSCSI RAID, or *iRAID* for short. The difference between *iRAID* and traditional RAID is that in traditional RAID, disk is the unit, while in *iRAID* each iSCSI target is a unit. Similar to traditional RAID, we may have different layouts/RAID levels. In this paper we only focus on two layouts: striping (*S-iRAID*) and rotated parity (*P-iRAID*). By striping data among several iSCSI targets, *S-iRAID* improves the read/write performance of iSCSI dramatically. Our experiments show that the average throughput is improved from 11.7MB/s to 46.1MB/s by striping data among only three iSCSI targets using Gigabit Ethernet. *S-iRAID* improves the performance but also worsen reliability. By striping data among several iSCSI targets, any single target failure will cause the entire data loss. To address this problem, in *P-iRAID*, a rotated parity block is used to

every data stripe. In case one iSCSI target fails, data can be reconstructed from other n-1 iSCSI targets.

To quantitatively evaluate the performance potential of *iRAID* in real world network environment, we have implemented the prototype of *iRAID* under the Linux OS based on current iSCSI code [11] and Linux software RAID. We have used Iozone benchmark [12] to measure system performance. Extensive measurement results show that *S-iRAID* and *P-iRAID* can speed up the iSCSI performance by a factor of up to 6.6 and 2.17, respectively.

The rest of the paper is organized as follows. Next section presents the design and implementation of *iRAID* including *S-iRAID* and *P-iRAID*, followed by our performance evaluation. We discuss the related research work in Section 4 and conclude our paper in Section 5.

2. Design of iSCSI RAID (iRAID)

We introduce iSCSI RAID, or *iRAID,* to solve the performance and reliability problems of iSCSI storage systems. The basic idea of *iRAID* is to organize the iSCSI storage targets similar to RAID by using striping and rotated parity techniques. In *iRAID*, each iSCSI storage target is a basic storage unit in the array, and it serves as a storage node as shown in Figure 1. All the nodes in the array are connected to each other through a high-speed switch to form a local area network. *iRAID* provides a direct and immediate solution to boost iSCSI performance and improve reliability. Parallelism in *iRAID* leads to performance gain while using the RAID parity technique improves the reliability. This paper focuses on two *iRAID* configurations: striped *iRAID (*S-iRAID*)* and rotated parity *iRAID (P-iRAID).*

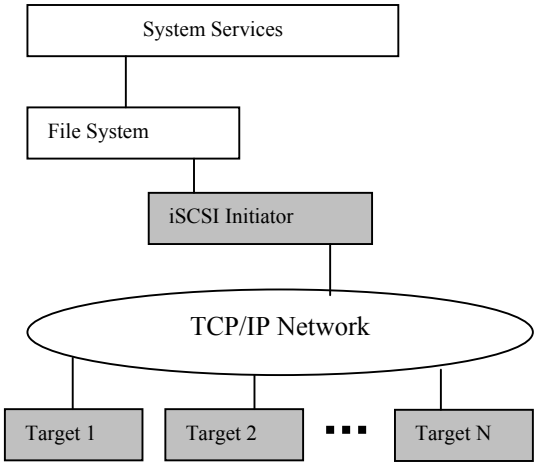

Figure 1: iRAID architecture. Data are striped over N iSCSI targets.

2.1 S-iRAID

In the striped *iRAID* (S-*iRAID*), all data are striped and distributed uniformly among all the *iRAID* nodes, which is illustrated in Figure 2. It borrows the concept from RAID level 0. Figure 2 shows the data organization of each *iRAID* node for a *S-iRAID* system consisting of n iSCSI targets, where D_{ij} indicates that data block i on iSCSI target j.

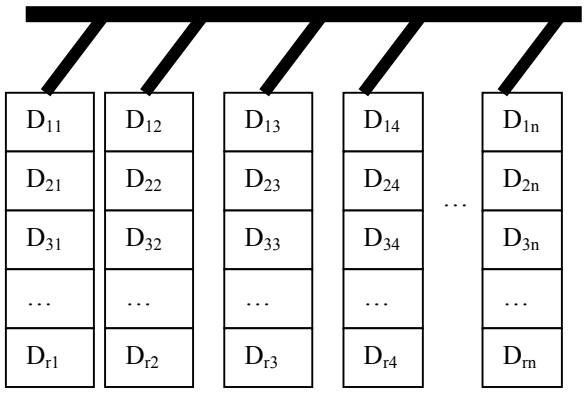

Figure 2: Data organization of S-iRAID.

2.2 P-iRAID

The *S-iRAID* increases the performance of iSCSI through parallism and also increases the security since S-*iRAID* splits data into

stripes and stores stripes in different nodes, which could be in different places over the network. But S-*iRAID* does not improve the reliability because failure of any single node will cause the data loss. To improve the reliability as well as performance, we introduce parity *iRAID* (*P-iRAID*) where in addition to data being striped and distributed among the *iSCSI* targets, a parity code for each data stripe is calculated and stored in an *iRAID* node. The parity block is rotated among the n iSCSI targets as shown in Figure 3, where the shadowed blocks are parity blocks, and others are data blocks. Each bit in a parity block is the XOR operation on the corresponding bits of the rest data blocks in each stripe. For example, $P_1 = D_{11} \otimes D_{12} \cdots \otimes D_{1,n-1}$.

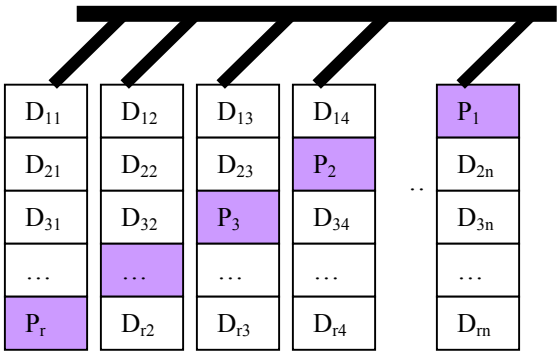

Figure 3: Data organization of P-iRAID.

size of Linux software RAID. All these machines are interconnected through a DELL PowerConnect 5012, 10-ports managed Gigabit Ethernet switch to form an isolated LAN. Each machine is running Linux kernel 2.4.18 with a 3COM 3C996B-T server network interface card (NIC) and an Adaptec 39160 high performance SCSI adaptor. STAR6 is used to monitor the network traffic over the switch. The configurations of these machines are described in Table 1 and the characteristics of the disks are summarized in Table 2.

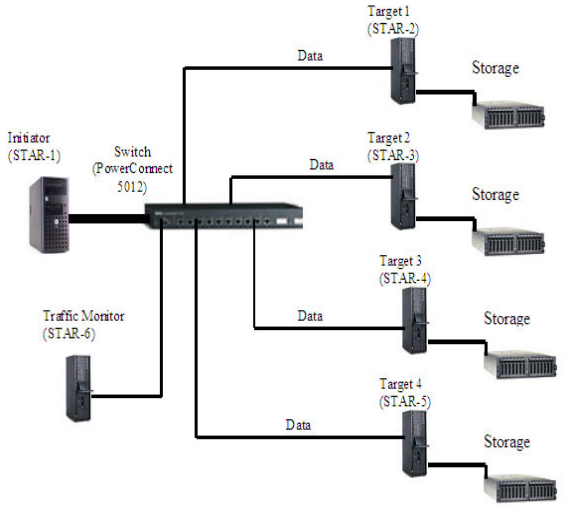

Figure 4: iRAID, where data are striped and distributed across the n iSCSI targets (S-iRAID) or n-1 iSCSI targets (P-iRAID).

3. Performance Evaluations

3.1 Experimental Setup

For the purpose of performance evaluation, we have implemented *iRAID* prototype (for both *S-iRAID* and *P-iRAID*) based on Linux software RAID and Intel iSCSI code [11]. Our experimental settings are shown in Figure 4. Six PCs are involved in our experiments, namely *STAR1* through *STAR6*. *STAR1* serves as the iSCSI initiator, and *START2-5* are four iSCSI targets, which are organized as our *iRAID*. The data block size is set to 64KB, which is the default chunk

We use the popular file system benchmark tool, Iozone [12], to measure the performance. The benchmark tests file I/O performance for a wide range of operations. We will focus on performance of sequential read/write, random read/write because those are generally the primary concerns for any storage systems. The average throughput listed here is the arithmetic average of above four I/O operations. We run Iozone for different request size and data sets under various scenarios as follows:

Iozone –Ra –S *dataset size* –r *request size* –P –i0 –i1 –I 2 –f /mnt/iRAID/test

Table 1: Machines configurations

Machines	Processor	RAM	IDE disk	SCSI Controller	SCSI disk
STAR-1	PIII 1.4GHZ/512K Cache	1024MB	N/A	Adaptec 39160, Dell PERC RAID controller	4x Seagate ST318406LC
STAR2…5	P4 2.4GHZ/512K Cache	256MB	WDC WB400BB	Adaptec 39160	IBM Ultrastar 73LZX
STAR6	P4 2.4GHZ/512K Cache	256MB	WDC WB400BB	N/A	N/A

Table 2: Disk parameters

Disk Model	Interface	Capacity	Data buffer	RPM	Latency (ms)	Transfer rate (MB/s)	Average Seek time (ms)
ST318406LC	Ultra 160 SCSI	18GB	4MB	10000	2.99	63.2	5.6
Ultrastar 73LZX	Ultra 160 SCSI	18GB	4MB	10000	3	29.2-57.0	4.9
WB400BB	Ultra ATA	40GB	2MB	7200	4.2	33.3	9.9

Where dataset size and request size are configurable. We reboot all machines after each round of test.

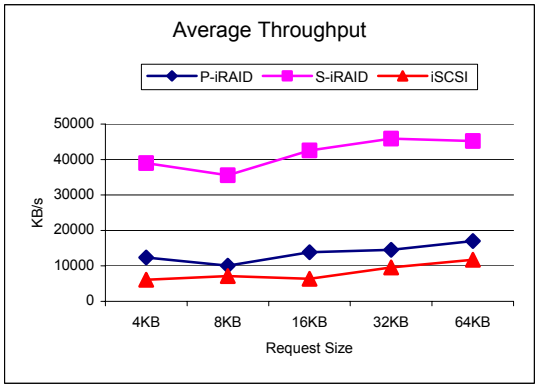

Figure 5: Average throughput
(4 targets, Gbps network, 1 GB data set)

3.2 Numerical Results

3.2.1 Throughput

Our first experiment is to use Iozone to measure the I/O throughput for iSCSI, *S-iRAID*, and *P-iRAID* using 4 targets under Gigabit Ethernet. The data set is 1G bytes and I/O request sizes range from 4KB to 64KB. Figure 5 shows the average throughputs. Both *S-iRAID* and *P-iRAID* improve the iSCSI performance dramatically. The performance improvement over iSCSI is consistent across different request sizes for both *S-iRAID* and *P-iRAID*. *S-iRAID* outperforms iSCSI by a factor of up to 6.6, and *P-iRAID* outperforms iSCSI by a factor of 2.17. It's obvious that by striping data from the iSCSI initiator among different iSCSI targets, S-iRAID and P-iRAID show great performance gains. *P-iRAID* calculates parity for each data stripe and uses one of the 4 iSCSI targets to store parity blocks. That's why *S-iRAID* performs much better than *P-iRAID*. The average performance gains of *S-iRAID* and *P-iRAID* over the iSCSI are a factor of 5.34 and 1.71, respectively.

3.2.2 Identifying the bottlenecks

Figure 5 shows that iSCSI performance is pretty low. Readers may ask what's the bottleneck. Is it the iSCSI initiator, network speed, or iSCSI target? To answer this question, we perform the following experiments.

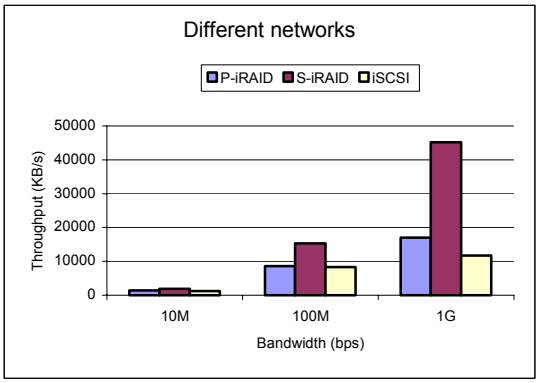

Figure 6: Average throughput
(4 targets, 1 GB data set, 64KB request size)

First, we change the network speed by setting the switch to different modes: 10Mbps, 100Mbps, and 1Gbps. Figure 6 shows the results for *S-iRAID*, *P-iRAID*, and iSCSI under different networks. It is clear that they perform similarly under a slow network (10Mbps). When the network speeds up, the *S-iRAID* performs much better than iSCSI, while *P-iRAID* show a smaller performance gain because of parity computation and less degree of parallism than *S-iRAID*.

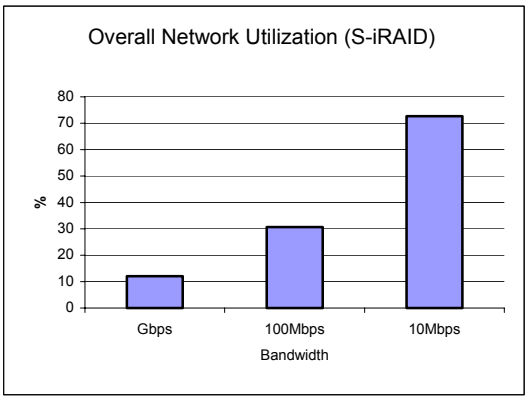

Figure 7: Network utilization
(4 targets, 1 GB data set, 64KB request size)

It seems that network bandwidth is the bottleneck for iSCSI using slow speed networks. To verify this, we monitored the network traffic over the switch. Figure 7 shows the network utilization for *S-iRAID*. We found that the network utilization is over 72% (which is very high for a network load!) using 10Mbps while only 12% using 1Gbps network. That means, if we use a slow speed network (10Mbps) for iSCSI, even we add more iSCSI targets, the performance will not increase too much because the network is the bottleneck. Our next experiment (where different number of iSCSI targets are used for different networks) further confirms our conclusion. Figure 8 shows the results, where we noticed that performance is consistent for 10Mbps network even we increase the number of iSCSI targets. For 100Mbps and 1Gbps networks, performance increases steadily with the increasing number of iSCSI targets.

Figure 8 also shows that for high-speed network, iSCSI targets may become the new bottleneck, that's why we can use more iSCSI targets in *iRAID* to improve the iSCSI performance. Figure 9 shows the results for *S-iRAID* using different number of iSCSI targets for different request sizes. The performance improvements are consistent across all the different request sizes. *S-iRAID* improves the iSCSI performance from 11.7MB/s to 46.1MB/s only using 3 iSCSI targets for 64KB request size.

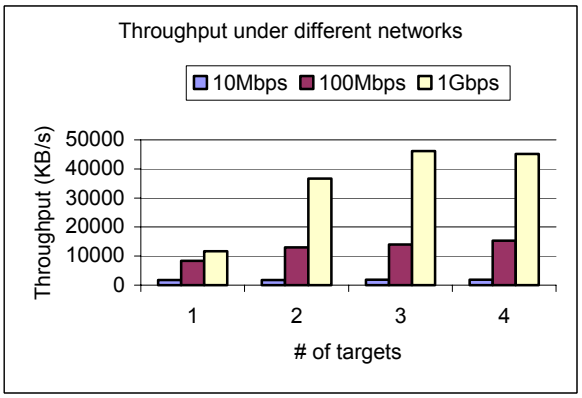

Figure 8: Average throughput
(1 GB data set, 64KB request size)

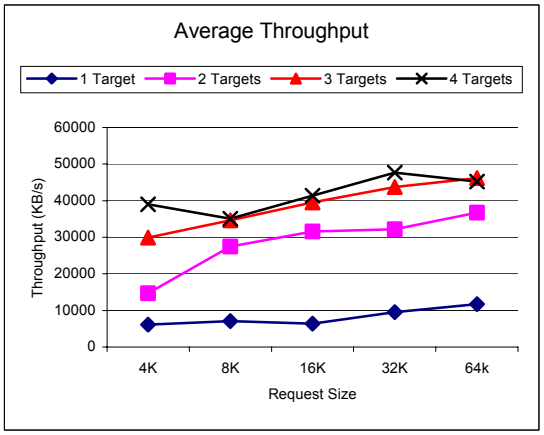

Figure 9: Average throughput
(1 GB data set, 1Gbps network)

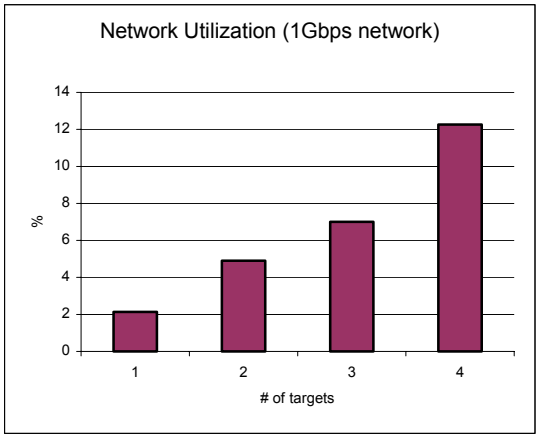

Figure 10: Network utilization for different number of iSCSI targets (1 GB data set, 64KB request size)

Both Figure 8 and 9 also unveil a fact: when the number of iSCSI targets exceed a threshold (3 in our experiment), the performance will not increase even we add more iSCSI targets. That means in our environment, 3 iSCSI targets is enough to saturate the 1Gbps network. iSCSI targets are not the bottleneck anymore when we use more than 3 iSCSI targets. So where is the bottleneck now? Is it the network? Probably not! Because we measured the network traffic and found that the network utilization is just 12% even using 4 iSCSI targets (Figure 10). Since now the bottleneck is neither the iSCSI targets not the network, we can conservatively assume the bottleneck is the iSCSI initiator! This problem is addressed by iCache [7], a cache scheme to improve the initiator performance.

3.3 Reliability analysis

As we mentioned above, *P-iRAID* improves the reliability by using rotated parity. The parity block is rotated among the n iSCSI targets. Each bit in a parity block is the XOR operation on the corresponding bits of the rest data blocks in each stripe. For example in an *n* iSCSI targets *P-iRAID*, in case of the failure of any single iSCSI target *i*, data on iSCSI target i can be recovered through the rest of n-1 iSCSI targets by performing XOR operations.

To verify the reliability of our *P-iRAID*, we performed a simple experiment as follows.

1) We mounted our 4-target *P-iRAID* as a local drive (***/mnt/p-iRAID***) on the initiator;
2) Copied the Linux source tree ***(/usr/*src*)*** to this *P-iRAID* drive;
3) Rebooted all those machines (initiator and targets), and formatted one of the iSCSI target drive (this will erase all data on it) to emulate one target failure;
4) We mounted the 4-target *P-iRAID* as a local drive (***/mnt/p-iRAID***) on the initiator again;
5) We compiled the Linux source tree ***/mnt/p-iRAID/usr/src*** successfully. That means our *P-iRAID* does improve the reliability and is safe from single iSCSI target failure.

4. Related Work

RAID is a mature technology developed to improve I/O performance and/or reliability. Distributed RAID concept was originally presented by Stonebraker and Schloss [20] in 1990. Since then several distributed RAID techniques are used in cluster computing or direct attached storage systems. RAID-x [10] makes use of orthogonal striping and mirroring technique in a serverless cluster to improve the aggregate I/O bandwidth for parallel writes. TickerTAIP [2] offers a parallel RAID architecture for supporting parallel disk I/O with multiple controllers. However, the TickerTAIP was implemented as a centralized I/O subsystem. MAID [3] builds a mass storage utilizing idle disk resources. Prototyping of distributed RAID started with the Swift/RAID [14] and *Petal* [13]. Swift/RAID provides fault tolerance in the

distributed environment in the manner as RAID level 4 and 5. Petal uses a collection of NAS-like storage servers interconnected using specially customized LAN to form a unified virtual disk space to clients at block level.

NAS [5] and SAN [6] are two major solutions to deploy storage over the network. The NAS technology provides direct network connection for hosts to access through network interfaces. It also provides file system functionality. NAS-based storage appliances range from terabyte servers to a simple disk with Ethernet plug. The main difference between NAS and SAN is that NAS provides storages at file system level while SAN provides storages at block device level. Another difference is that NAS is attached to the same LAN as the one connecting servers accessing storages, while SAN has a dedicated network connecting storage devices without competing for network bandwidth with the servers. Recently emerged *iSCSI* (Internet SCSI) [19] provides an ideal alternative to Petal's customized LAN-based SAN protocol. Taking advantage of existing Internet protocols and media, it is a natural way for storage to make use of TCP/IP as demonstrated by earlier research work (VISA [16] by Meter et al. of USC) to transfer SCSI commands and data using IP protocol. iSCSI protocol is a mapping of the SCSI remote procedure invocation model over the TCP/IP protocol [19]. Gabber et al propose StarFish [4] to improve availability for IP-based block storage using replicas.

iRAID provides a solution to deploy SAN using iSCSI over the existing and mature Ethernet protocol. While many existing techniques such as striping and rotated parity [17] may be borrowed for the *iRAID* and *iCache* implementations, the novelty of our work is the new concept of applying array technique to iSCSI storage systems.

5. Conclusions

In this paper, we have identified the iSCSI performance bottlenecks under different situations. For low-speed networks, the network bandwidth is the main factor to limit the performance. For high-speed networks, both iSCSI target and initiator could be the bottleneck. We have introduced *S-iRAID* to improve the performance by striping data among several iSCSI targets, and introduced *P-iRAID* to improve the reliability and performance by striping data and rotating parity over several iSCSI targets. We have carried out prototype implementations of *S-iRAID* and *P-iRAID* under the Linux operating system. Extensive measurement results using Iozone have shown that *S-iRAID* and *P-iRAID* can speed up the iSCSI performance by a factor of up to 6.6 and 2.17 in terms of average throughput.

Acknowledgements

This research is partially supported by the Center for Manufacturing Research at Tennessee Technological University.

References

[1] S. Aiken, D. Grunwald, A. Pleszkun, and J. Willeke, "A Performance Analysis of the iSCSI Protocol," *20th IEEE Conference on Mass storage Systems and Technologies*, 2003.

[2] P. Cao, S. B. Lim, S. Venkataraman, J. Wilkes. "The TickerTAIP Parallel RAID Architecture," *ACM Transactions on Computer Systems* 12(3): 236-269 (1994).

[3] D. Colarelli and D. Grunwald, "Massive Arrays of Idle Disks For Storage Archives," *Proceedings of Super*

Computing (SC'2002), Baltimore, MD, November 2002.

[4] E. Gabber, et al., "StarFish: highly-available block storage," *Proceedings of the FREENIX track of the 2003 USENIX Annual Technical Conference*, San Antonio, TX, June 9--14, 2003, pp. 151-163.

[5] G. Gibson, R. Meter, "Network Attached Storage Architecture," *Communications of the ACM*, Vol. 43, No 11, pp.37-45, November 2000.

[6] M. Gupta, *Storage Area Network Fundamentals*, Cisco Press, ISBN: 1-58705-065-x, 2002.

[7] X. He, et al., "A Caching Strategy to Improve iSCSI Performance," *IEEE Annual Conference on Local Computer Networks*, Nov. 6-8,2002.

[8] X. He, Q. Yang, and M. Zhang, "Introducing SCSI-To-IP Cache for Storage Area Networks," in *2002 International Conference on Parallel Processing (ICPP'2002)*, Vancouver, Canada, August 18-21, 2002.

[9] R. W. Horst, D. Garcia, "ServerNet SAN I/O Architecture," *Hot Interconnects V*, 1997.

[10] K. Hwang, H. Jin, and R. S. Ho, "Orthogonal Striping amd Mirroring in Distributed RAID for I/O-Centric Cluster Computing", *IEEE-Trans. on Parallel and Distributed Systems*, 2001.

[11] Intel iSCSI project, URL: *http://sourceforge.net/projects/intel-iscsi*, Jan. 2003.

[12] Iozone file system benchmark, URL: *http://www.iozone.org*.

[13] E. K. Lee, C. A. Thekkath. "Petal: Distributed Virtual Disks." *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS VII),* pp. 84-92, Oct. 1-5,1996.

[14] D. E. Long, B. R. Montague, L. Cabrera, "Swift/RAID: A Distributed RAID System," *Computing Systems* 7(3): 333-359 (1994).

[15] Y. Lu and D. Du, "Performance Study of iSCSI-based Storage Systems," *IEEE Communications*, Vol. 41, No. 8, 2003.

[16] R. V. Meter, G. G. Finn, S. Hotz. "VISA: Netstation's Virtual Internet SCSI Adapter." In *Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS VIII)*, pp. 71-80, October 4-7,1998.

[17] D.A. Patterson, et al., "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *ACM International Conference on Management of Data (SIGMOD)*, pp. 109-116, 1988.

[18] P. Sarkar, S. Uttamchandani, and K. Voruganti, "Storage Over IP: When Does Hardware Support Help?" *USENIX Conference on File And Storage Technologies*, 2003.

[19] J. Satran, et al. "iSCSI draft standard," *URL: http://www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-20.txt*, Jan. 2003.

[20] M. Stonebraker and G. Schloss, "Distributed RAID- a New Multiple Copy Algorithm," *Proceedings of the 6th International Conference on Data Engineering*, Feb. 1990, pp. 430-437.

[21] P. Wang, et al.,"IP SAN-from iSCSI to IP-Addressable Ethernet Disks," *20th IEEE Conference on Mass storage Systems and Technologies*, 2003.