TPT-RAID: a High Performance Box-Fault Tolerant Storage System

Yitzhak Birk
The Technion – Israel Institute of Technology
birk@ee.technion.ac.il

Erez Zilber
The Technion – Israel Institute of Technology
erezz@voltaire.com

Abstract

TPT-RAID *is a multi-box* RAID *wherein each* ECC *group comprises at most one block from any given storage box, and can thus tolerate a box failure. It extends the idea of an out-of-band* SAN *controller into the* RAID*: data is sent directly between hosts and targets and among targets, and the* RAID *controller supervises* ECC *calculation by the targets. By preventing a communication bottleneck in the controller, excellent scalability is achieved while retaining the simplicity of centralized control.* TPT-RAID*, whose controller can be a software module within an out-of-band* SAN *controller, moreover conforms to a conventional switched network architecture, whereas an in-band* RAID *controller would either constitute a communication bottleneck or would have to also be a full-fledged router. The design is validated in an InfiniBand-based prototype using i*SCSI *and i*SER*, and required changes to relevant protocols are introduced.*

1. Introduction

In most current RAIDs [1], including very large ones, any given error-correcting (ECC) [1] group resides in a single box. Regardless of the degree of internal redundancy and reliability, a single-box RAID is thus susceptible to box-level failures (e.g., cable disconnection, flood, coffee spill), as these render entire ECC groups unavailable.

In a multi-box RAID, each ECC group uses at most one block from each storage box, so the failure of such a box does not render any data inaccessible. The controller must be fault tolerant (e.g., by having a hot backup [2]), as must the network [3]. Our work focuses on multi-box RAIDs with centralized control, and we use the term Multi-box RAID to refer to such systems.

Unlike a single-box RAID that uses a DMA engine for internal data transfers, a multi-box RAID must use the network, e.g., iSCSI over TCP, for all transfers. This requires extra data copies that affect both throughput and latency,

Figure 1. In-Band controller

Figure 2. Out-Of-Band controller

and moreover burdens the CPUs. Overcoming the single point of target failure by going to a multi-box RAID thus poses several challenges: communication efficiency and prevention of a controller bottleneck. Controller fault tolerance can be handled through well-known mechanisms; it is not addressed in this paper, as the proposed architecture does not place any special demands in this respect.

1.1. In-band vs. Out-of-band RAID Controller

Current SAN controllers are either "in-band" (Fig. 1) or "out-of-band" (Fig. 2). In single-box RAIDs, the RAID controller is naturally in the data path. In a multi-box RAID, however, an in-band RAID controller is problematic:
– Connecting it to a single switch port renders it a communication bottleneck, as it is party to all communication.
– Connecting it via multiple ports may help but is costly, requires load balancing among the ports, and its internal data paths could be the bottleneck.
– Locating it inside the switch, acting as a router, would relieve the bottleneck, but the "orthogonality" of communication and other functions would be violated. A multi-box RAID [2] comprising disk boxes and a controller, all interconnected by a common network, naturally admits an out-of-

[1] We focus on erasure correcting codes, mostly XOR, and use the term ECC loosely. Nonetheless, TPT-RAID can be adapted to use any ECC.

[2] We use RAID-5 as an example, and refer to it simply as "RAID". However, this work is equally applicable to other RAID types.

24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007)
0-7695-3025-7/07 $25.00 © 2007 IEEE

band RAID controller. However, this gives rise to various performance challenges and to the location of ECC calculations, which cannot be performed by such a controller.

1.2. Contributions of this work

We present the 3rd Party Transfer multi-box RAID architecture, TPT-RAID, which partitions the RAID controller functions: the management functions are taken out of the storage box and placed in a centralized, out-of-band TPT-RAID controller, while data transfers and ECC calculations are carried out directly among targets and hosts and within targets, respectively, all under centralized control. The controller only handles control messages (with the exception of unsolicited data). TPT-RAID thus carries the idea of an out-of-band SAN controller one step further, into the RAID itself, and is consequently also described by Fig. 2. In fact, it can be implemented as a software module inside such a SAN controller. Finally, since SANs and RAIDs are often used as the back end of NAS and object based systems, TPT-RAID is also relevant to those.

1.3. Related Work

Previous studies addressed the aforementioned problems (the high price of RAIDs, tolerating a box failure and scalability of the RAID controller), but did not simultaneously solve all of them: several studies focused on multi-box (or distributed) storage systems [4, 5, 6, 7, 8]; Gray et al [2] addressed controller failures; the RAIN project [3] addressed network failures; commercial storage solutions (Hitachi Lightning [9], EMC Symmetrix DMX system [10]) address high availability.

Some commercial file systems focused on removing the controller from the data path [11, 12, 13, 14]: clients receive metadata from a metadata server, and data is transferred directly between clients and the actual storage. The idea of moving the controller (or metadata server) out of the data path is conceptually similar to our 3rd Party Transfer, but is applied at file level rather than at block level. We also incorporate InfiniBand [15] and RDMA [16], and demonstrate their efficiency.

Other commercial systems focused on removing the controller from the data path at block level. SVM [17] is a SAN appliance that provides virtual volume management and has an out-of-band controller. However, it does not use RDMA, so although data is sent directly between hosts and targets, it is not transparent to the host because it has to explicitly send/receive data packets to/from the storage. Also, when using SVM for backup, the controller is in the data path. FAB [18] is a distributed disk array that comprises multiple identical storage servers. Each server can act as a gateway for requests from clients.

The specification of SCSI [19] contains block commands that support parity calculation by the disk drive itself. This can be used to distribute the ECC work among targets and to reduce the overall required data movement.

The remainder of the paper is organized as follows. Section 2 describes the use of iSER to solve some of the problems of a multi-box RAID. Section 3 presents our 3rd Party Transfer multi-box RAID architecture (TPT-RAID). Section 4 presents our TPT-RAID prototype and some performance measurements, and Section 5 offers concluding remarks.

2. Multi-box RAID with iSER

Many networked storage systems use the Internet SCSI [20] (iSCSI) protocol for sending SCSI commands and data over a (TCP) network. iSER [21] is an IETF standard that maps iSCSI onto a network that provides more efficient and "transparent" RDMA services, e.g., TCP with RDMA services (iWARP [16, 22, 23]) and InfiniBand. With iSCSI over iSER, data is sent between the initiator and target I/O buffers without intermediate data copies. This may be viewed as a substitute for the DMA engine that is used by a single-box RAID. While being highly beneficial, however, other problems of the multi-box system remain unsolved:

– The control/data separation offered by iSER is really a protocol separation over the same physical path. All data transfers go via the controller.

– ECC calculations require additional data transfers between the controller and the disks as well as calculations by the controller.

We will use iSCSI over iSER with an in-band controller as a baseline for comparison (the Baseline system). We next introduce TPT-RAID, which extends the use of RDMA to address the remaining problems.

3. TPT-RAID

3.1. Overview

TPT-RAID, depicted in Fig. 3, is a multi-box RAID that combines a central out-of-band controller with RDMA-based data transfer. RDMA is both efficient and obviates the need for the hosts to be aware of the details of the operation. ECC calculation is performed by the storage targets. TPT-RAID uses iSCSI for sending commands and data. Other than the required changes in iSCSI, all its features can be used without change.

TPT-RAID features two main elements: 3rd Party Transfer (TPT) and ECC calculation by the targets.

TPT. Read/write data passes directly between the host and the targets under controller command, and data for parity calculation is sent among targets under controller command. Only control messages pass through the controller (with the exception of unsolicited data in WRITE requests as explained later in this section).

ECC calculation by the targets is carried out in one of the following ways, both of which employ RDMA:

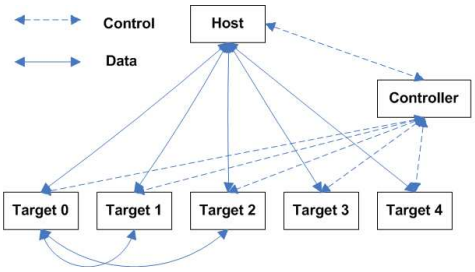


Figure 3. TPT-RAID

- Star topology: per command instance, one of the targets is chosen as the "calculator". It fetches all required blocks from other targets, computes the ECC information and writes it to its own disk or to another target(s). With a single calculation point, this approach is simple and can be used with any error/erasure correcting code. Also, it minimizes the total amount of communication. Finally, the ad hoc selection of the "calculator" is used to balance the load among targets.

- Binary tree: multiple targets participate in any given ECC calculation (under controller supervision). This scheme is less general and more complex, but may reduce latency in certain low-load situations.

TPT-RAID requires several extensions to both SCSI, iSCSI and iSER, as outlined in section 3.4 and listed in [24].

3.2. Request execution

WRITE. Data may be sent to the target as unsolicited and/or solicited data [20]. In TPT-RAID, we only modify solicited writes. Unsolicited data is sent from the host to the controller as a Protocol Data Unit (PDU) without using RDMA, rendering 3rd Party Transfer irrelevant.

Fig. 4 provides a step-by-step description of writing a single block to target 0, with parity calculation and storage in target 4. Both data flow (left) and a time line (right) are shown. All data transfers employ RDMA, and all take place under controller supervision.

Multi-stripe WRITE requests contain 0-2 *partial stripes* and *full stripes*. They are handled as 0-2 partial-stripe requests and a single multiple-full-stripe request. Handling of the latter is optimized by reading the blocks of any given target for parity calculation with a single command, so different commands are used for partial and full stripes.

For full stripes, the controller sends a separate command for each block (a target may receive multiple commands) because although the data is contiguous in the host, it needn't be contiguous in the targets.

It should be noted that even for full-stripe writes, the only way to transfer fewer blocks than does TPT-RAID is to let the host compute the ECC, thereby blurring the boundary between the compute host and the storage subsystem.

READ. Here again, the 3rd Party Transfer mechanism is employed to enable direct RDMA transfers from targets to hosts under controller supervision.

For further details regarding request execution, the required protocol extensions, and the scheduling that guarantees atomicity, see [24].

3.3. Error handling

TPT-RAID can handle target, controller and network failures, and its architecture does not create new problems relative to other multi-box storage systems. The sequence of commands issued by the controller also guarantees request atomicity: upon failure, a request is either completed or rolled back, and the controller is advised accordingly [24].

3.4. Required protocol changes

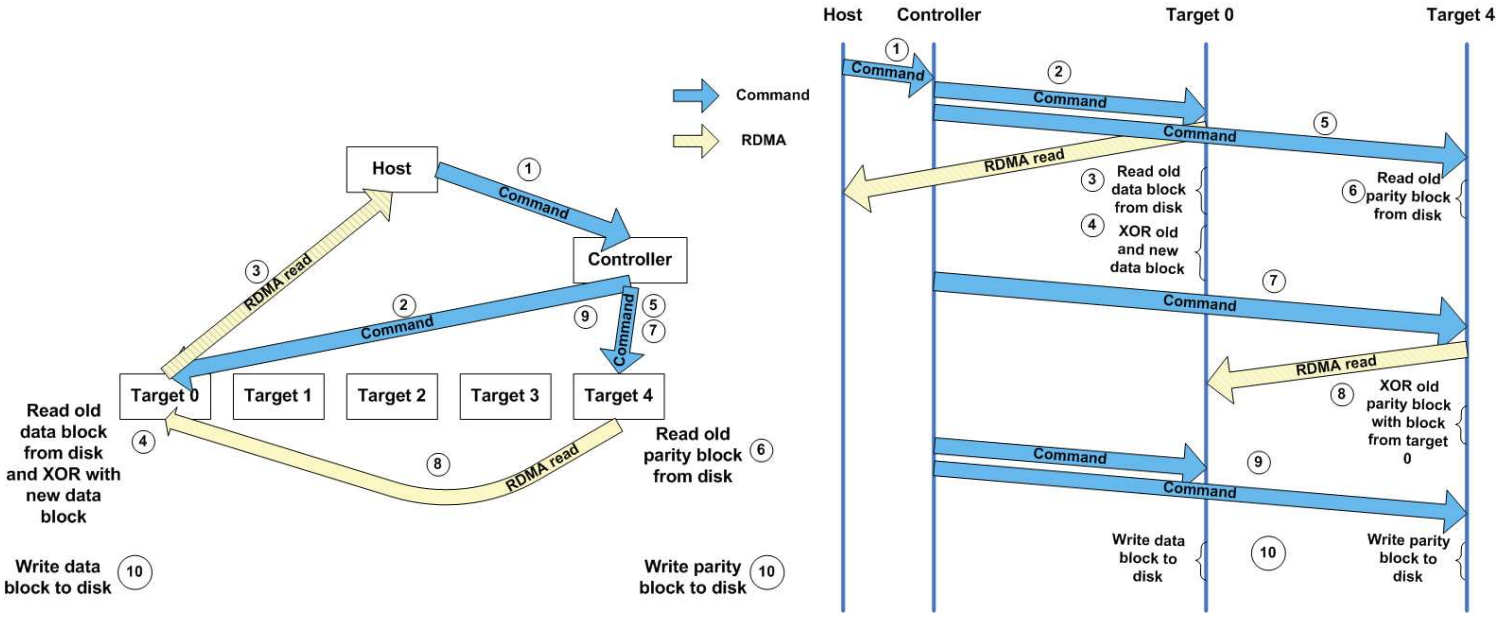
Support of 3rd Party Transfer and ECC calculation by the targets requires some protocol changes and extensions, mostly in order to enable the controller to instruct the targets to do things that they are already capable of doing in principle (e.g., parity calculation). The changes are confined to software layers, and do not complicate the standards. The changes, listed in [24], are outlined below.

SCSI. The new commands are all sent from the RAID controller to targets. They are used by the target only for software purposes (i.e. the SCSI hardware that the target is connected to does not need to support these commands). Some of the commands relate to XOR operations, but differ from the SCSI XOR commands (although we do use the XDWRITE command) because these were designed for a scheme with an in-band controller while our's is out of band. However, if disks in a TPT-RAID are capable of performing XOR operations, our new commands may use them in a similar manner to their use by SCSI XOR commands.

iSCSI. The login and logout phases in TPT-RAID differ from those of a SAN with single-box RAIDs and an out-of-band controller: our controller must establish a connection with each target, and each target must establish a connection with all other targets and with each host [3]. TPT-RAID also raises potential security problems during connection establishment and login due to rogue targets that may establish connections to other targets and hosts. This is handled by TPT-RAID's login mechanism [24].

A SCSI command PDU sent from the controller to a target must contain an extra field in support of 3rd Party Transfer. The target uses this field to determine which (other) target or host should be the passive side of an RDMA

[3] The target-target and host-target connections are used only for RDMA operations. No buffers need to be allocated for these connections at either end. Therefore, these connections hardly consume any resources.



(a) System view (b) Timeline view

Figure 4. Single-block WRITE with TPT-RAID

operation. SCSI command PDUs sent from hosts to the controller remain unchanged.

iSER. Small changes are required in certain iSER primitives in support of 3rd Party Transfer. However, the RDMA mechanism remains unchanged.

4. TPT-RAID prototype and performance

In order to validate the TPT-RAID architecture, assess its performance relative to the Baseline in-band controller and its scalability, we constructed prototypes of the two systems using identical hardware for all types of boxes and for both prototypes. Each system comprises a single host, a RAID controller and 5 targets. Each machine has dual Intel Pentium 4 XEON 3.2GHz processors with 2MB L2 cache and an 800 MHz front side bus. Each machine contains a Mellanox MHEA28-1T 10Gb/sec full duplex Host Channel Adaptor (HCA) with a PCI-Express X8 interface. All machines are connected to a Mellanox MTS2400 InfiniBand switch. Since the target machines have low end SATA disks that may limit performance, we simulate targets containing multiple disks and large caches by not sending the SCSI commands to the disk. Instead, a successful SCSI response is returned immediately. (The returned data blocks contain random data.) We now briefly describe our measurements.

4.1. Scalability study

Let *request size* denote the amount of data requested in the command sent by the host, *block size* — the striping granularity, and *target set* — N parity-group targets.

Figs. 5 and 6 present a scalability comparison for READ and WRITE requests, respectively, with unlimited numbers of hosts and targets and the "star" parity calculation. Here, the controller itself or its communication links are the bottleneck. Several *request* and *block* sizes are considered.

Fig. 5 depicts maximum READ throughput.

Baseline system: block size hardly affects scalability. For a small *request size*, the controller is limited by its CPU, which is busy sending commands to targets. The number of such commands per request is independent of *block size*, so changing it hardly affects maximum throughput. As *request size* increases, the controller's InfiniBand link limits throughput to 920MB/sec.

TPT-RAID: throughput is limited by the controller's CPU, most of whose work is spent on sending commands. With larger blocks, there are fewer commands per request so, for a given *request size*, maximum throughput increases with increasing *block size*. For a sufficiently large *request size*, the per-request controller work (excluding the per-block work) is negligible, so further increasing *request size* hardly changes the controller's scalability.

Comparison. For small blocks (4KB), the TPT-RAID controller (Fig. 5, "TPT (block size = 4KB)") enables higher throughput than the Baseline controller ("Baseline (block size = 4KB)") for small requests (*request size* $\leq$ 64KB). For larger requests, the bottleneck in the Baseline controller moves from the CPU to its InfiniBand link, enabling higher throughput than the TPT-RAID controller. For larger blocks, the TPT-RAID controller ("TPT (block size = 32KB)" and "TPT (block size = 128KB)") needs to send fewer com-

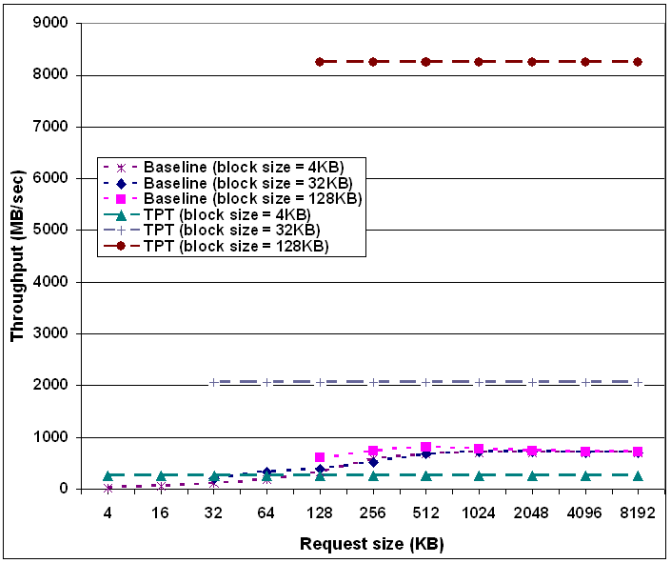


Figure 5. READ scalability

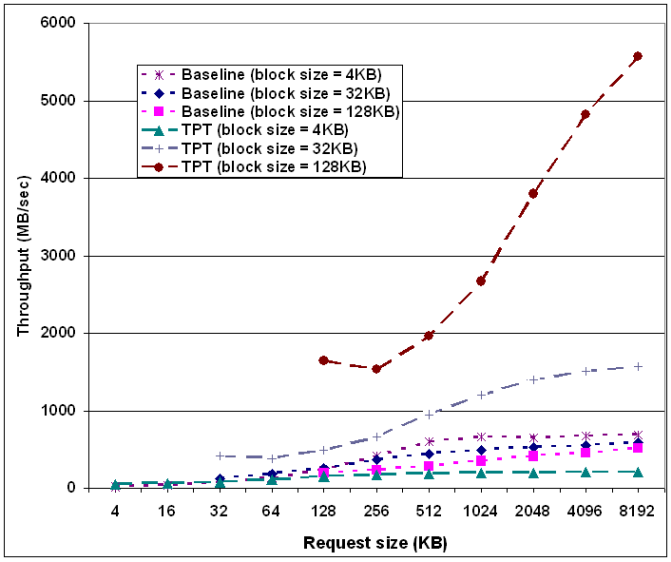


Figure 6. WRITE scalability

mands per request and enables a higher throughput than the Baseline controller ("Baseline (block size = 32KB)" and "Baseline (block size = 128KB)"). TPT-RAID's advantage increases as *block size* increases.

Fig. 6 depicts maximum WRITE throughput.

Baseline system: the controller is limited by its CPU. For small requests, the CPU is busy sending commands to targets. For larger requests, the CPU is busy performing XOR operations. Even if the controller had dedicated hardware for XOR operations, it would still have been limited by its InfiniBand link.

TPT-RAID: as for READ requests, it is again limited by the controller's CPU, which is busy sending commands to targets. With larger blocks, fewer commands are required for any given *request size*. As described in section 3.2, full-stripe WRITEs are handled as a single multi-stripe request. Therefore, with large requests, the multi-stripe request contains more blocks and fewer commands are required per block. Indeed, the maximum throughput enabled by the controller increases with increasing *block* and *request* sizes.

Comparison. For small blocks (4KB), the two controllers (Fig. 6, "Baseline (block size = 4KB)" and "TPT (block size = 4KB)") enable almost the same throughput for small requests (*request size* $\leq$ 64KB). For larger requests, the bottleneck in the Baseline controller remains in the CPU, which is now busy with XOR calculations, and throughput is higher than in the TPT-RAID controller. For larger blocks, the TPT-RAID controller ("TPT (block size = 32KB)" and "TPT (block size = 128KB)") needs to send fewer commands for a given *request size* and enables higher throughput than the Baseline controller ("Baseline (block size = 32KB)" and "Baseline (block size = 128KB)") for any *block size*. TPT-RAID's advantage increases as *block size* and *request size* increase.

Table 1. RAID controller scalability

Block (KB)	Req (KB)	READ requests		WRITE requests	
		Multi	Single	Multi	Single
4	any	< 1	< 1	< 1	< 1
32	≥ 128	2	2	< 1	< 1
128	≥ 512	8	5	2	2

Degraded mode. Relative performance of the two systems for READs, WRITEs and reconstruction in degraded mode is similar to relative WRITE performance in normal mode [24].

Table 1 summarizes the maximum number of hosts that can be connected to a single-controller TPT system that comprises a single/multiple target sets (N = 5) without having the controller or the targets become a bottleneck. (Using multiple target sets ensures that targets are not a bottleneck). For the Baseline system, the controller is always a bottleneck, even if only a single host is connected to it.

Remark. The measured systems used memory disks. With real disk drives, a single controller may be connected to even more targets without becoming a bottleneck.

4.2. Latency

In most cases, TPT-RAID's zero-load latency is equal to or lower than the Baseline RAID's. As the load increases, latency is dominated by queuing delay. Here, the lower load factor on the higher-capacity TPT-RAID results in its clear latency advantage.

5. Conclusion

TPT-RAID takes the idea of an out-of-band SAN controller one step further, into the RAID, and a TPT-RAID controller can in fact be implemented as a software component of an out-of-band SAN controller. It enables the construction of high-performance, box-fault tolerant SAN-based storage systems from relatively simple and inexpensive components while retaining simplicity. TPT-RAID enables higher throughput than the Baseline in-band controller whenever block (and request) sizes exceed 32KB, in which case the savings in controller data-communication and parity calculation work outweigh the larger control-message work. This performance advantage increases with further increases in block or request size.

TPT-RAID's two main underlying mechanisms, 3rd-party transfer and ECC calculation by the targets, do not require hardware changes and only few changes are required in SCSI, iSCSI and iSER protocols. While any communication infrastructure can be used, the advantage of using ones like InfiniBand that support RDMA is clear.

Our performance measurements focused on the basic operations. In view of the promising results, it will be interesting to experiment with TPT-RAID as part of a full-fledged system, running standard benchmarks and operating in various modes.

Acknowledgments and credits. The measurements were assisted by a project carried out by Yevgenia Alperin and Michael Lyulko in the Parallel Systems Lab, EE Department, Technion. The authors thank Intel, Mellanox and Voltaire for their generous equipment grants and support.

References

[1] D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *Proc. ACM Intl. Conf. on Management of Data (SIGMOD)*, pp. 109–116, June 1988.

[2] J. Gray, B. Horst, and M. Walker, "Parity striping of disc arrays: low-cost reliable storage with acceptable throughput," *Proc. 16th intl. conf. on very large databases (*VLDB*)*, pp. 148–161, San Francisco, CA, 1990, Morgan Kaufmann Publishers Inc.

[3] V. Bohossian, C. C. Fan, P. S. LeMahieu, M. D. Riedel, J. Bruck, and L. Xu, "Computing in the RAIN: A reliable array of independent nodes," *IEEE Trans. Parallel and Distrib. Syst.*, 12(2):99–114, 2001.

[4] M. Stonebraker and G. A. Schloss, "Distributed RAID - a new multiple copy algorithm," *Proc. 6th Intl. Conf. on Data Engr.*, pp. 430–437, Washington, DC, Feb. 1990, IEEE Computer Society.

[5] P. Cao, S. B. Lin, S. Venkataraman, and J. Wilkes, "The TickerTAIP Parallel RAID Architecture," *ACM Trans. on Computer Systems*, 12(3):236–269, Aug. 1994.

[6] E. K. Lee and C. A. Thekkath, "Petal: Distributed Virtual Disks," *Proc. 7th Intl. Conf. on Architectural Support for Programming Lang. and Oper. Sys. (ASPLOS VII)*, pp. 84–92, Oct. 1996.

[7] X. B. He, P. Beedanagari, and D. Zhou, "Performance evaluation of distributed iSCSI RAID," *Proc. Intl. Workshop on Storage Network Architecture and Parallel I/Os (SNAPI)*, pp. 11–18, New York, NY, Sept. 2003, ACM Press.

[8] K. Rao, J. L. Hafner, and R. A. Golding, "Reliability for Networked Storage Nodes," *Proc. Intl. Conf. on Dependable Sys. and Networks (DSN'06)*, pp. 237–248, Washington, DC, 2006, IEEE Comp. Soc.

[9] Lightning 9900 V Series, Hitachi, www.hds.com/assets/pdf/9900v_architecture_guide_437_02.pdf.

[10] Symmetrix DMX Series, EMC, www.emc.com/products/systems/symmetrix/ DMX_series/pdf/DMX_series.pdf.

[11] Lustre, Cluster File Systems, www.lustre.org.

[12] P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur, "PVFS: A parallel file system for linux clusters," *Proc. 4th Annual Linux Showcase and Conf.*, pp. 317–327, Atlanta, GA, 2000, USENIX Assoc.

[13] G. Gibson and P. Corbett. "pNFS Problem Statement," July 2004, www.pdl.cmu.edu/pNFS/archive/gibson-pnfs-problem-statement.html.

[14] G. Gibson, B. Welch, G. Goodson, and P. Corbett, "Parallel NFS Requirements and Design Considerations," Oct. 2004, www.pdl.cmu.edu/pNFS/archive/gibson-pnfs-reqs.html.

[15] InfiniBand Architecture Specification Release 1.2, InfiniBand Trade Association, Oct. 2004, www.infinibandta.org.

[16] R. Recio, P. Culley, D. Garcia, J. Hilland, and B. Metzler, "A Remote Direct Memory Access Protocol Specification," Sept. 2006, www.ietf.org/internet-drafts/draft-ietf-rddp-rdmap-07.txt.

[17] Storage Virtualization Manager, LSI, www.lsi.com/storage_home/products_home/software/SVM/index.html.

[18] Y. Saito, S. Frolund, A. Veitch, A. Merchant, and S. Spence, "FAB: building distributed enterprise disk arrays from commodity components" *SIGOPS Oper. Syst. Rev.*, 38(5):48–58, 2004.

[19] SCSI Block Commands - 2 (SBC-2), INCITS T10 Technical working group, 2004.

[20] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)," RFC 3720 (Proposed standard), Apr. 2004, www.ietf.org/rfc/rfc3720.txt.

[21] M. Ko, M. Chadalapaka, U. Elzur, H. Shah, P. Thaler, and J. Hufferd, "iSCSI extensions for RDMA specification," IETF Draft, Nov. 2006, www.ietf.org/internet-drafts/draft-ietf-ips-iser-06.txt.

[22] P. Culley, U. Elzur, R. Recio, S. Bailey, and J. Carrier, "Marker PDU Aligned Framing for TCP Specification," Oct. 2006, www.ietf.org/internet-drafts/draft-ietf-rddp-mpa-08.txt.

[23] H. Shah, J. Pinkerton, R. Recio, and P. Culley, "Direct Data Placement over Reliable Transports," Sept. 2006, www.ietf.org/internet-drafts/draft-ietf-rddp-ddp-07.txt.

[24] Y. Birk and E. Zilber, "The TPT-RAID Architecture for Box-Fault Tolerant Storage Systems," *CCIT Tech. Rep. 629, Elec. Engr. Dept., Technion – Israel Inst. of Technology*, 2007.