Inter-node Communication in Peer-to-Peer Storage Clusters [*]

André Brinkmann
University of Paderborn
brinkman@hni.upb.de

Sascha Effert
University of Paderborn
fermat@upb.de

Abstract

Storage clusters try to transfer the idea of cluster computing into the storage domain and to scale capacity and performance by simply adding new cluster components. This paper presents analytical considerations on the scalability of storage clusters and presents a storage cluster architecture based on peer-to-peer computing that is able to scale up to hundreds of servers and clients. The resulting storage cluster environment has been successfully implemented and tested on a Linux based HPC-cluster. The measurement results presented in this paper demonstrate the feasibility and scalability of this architecture.

1. Introduction

Cluster-based storage tries to transfer the idea of cluster computing into the storage domain. A storage cluster is based on a set of storage appliances, called storage bricks, that work together closely and can be seen, from the outside, as a single, huge and fast storage system. The storage bricks are managed by a storage cluster middleware that is implemented as a software system managing the distributed state information about the storage cluster [13].

A major distinction between storage clusters and conventional storage architectures is that storage bricks are assembled based on commodity server architectures, enabling cost-savings compared to dedicated architectures [3]. Therefore each storage brick does not only provide storage capacity, but also computing and communication power. The computing capabilities enable a storage brick to contain a software management stack and to act as a storage appliance. The software stack inside a storage brick is responsible for a seamless integration of the brick into the cluster environment. Integrating a new storage brick therefore only involves the assignment to a storage resource pool, all other administration tasks, like authentication or rights management, are handled by the middleware. A characteristic element of storage clusters is that adding new storage bricks does not only increase storage capacity, but also the performance of the entire cluster.

Storage bricks can either use directly attached storage or networked storage as persistent storage. In the first case, only an interconnection infrastructure between the nodes of the storage cluster and to the client systems is necessary to provide scalable storage. In the second case, the bricks have to be connected to the networked storage systems over a storage area network, inducing additional costs and complexity, but also enabling the bricks to share storage devices without communication between the bricks.

The idea of a storage cluster as a collection of smaller components is closely related to storage virtualization and has been implemented first in the Petal prototype [10]. The main task of a storage cluster is to hide the complexity of the underlying storage systems by using a block-based storage virtualization environment or a distributed file system [15] [12] [7]. An example for an academic storage cluster is Ursa Minor, which provides access to objects instead of files or blocks and which is able to change data encoding and therefore performance and reliability of data objects based on attributes and access patterns [1]. The aim of the *Federated Array of Bricks (FAB)* is to deliver enterprise properties from a set of storage bricks at a fraction of the costs of an enterprise storage array [14]. The *V:Drive* project is based on randomized data distribution schemes, which are able to evenly spread data and accesses among all participating bricks and offer fast reorganization in case of failures or the integration of new bricks [5].

Storage Grids often use Ethernet as interconnection technology to the clients and between the storage bricks. In this paper we will focus on *Internet SCSI (iSCSI)* as interconnect protocol, which has been developed as an extension of the SCSI protocol environment for TCP/IP based networks [16]. Additional block level storage protocols over Ethernet are *HyperSCSI*, *NBD*, and *ENBD* [17][2]. It is of course also possible to use high speed networks like Infiniband or Myrinet as interconnect between the bricks or to the clients.

[*]This work has been partially supported by the EU within the 6th Framework Programme under contract IST-511531 *Highly Predictable Cluster for Internet Grids (HPC4U)*.

Authorized licensed use limited to: NASA Goddard Space Flight. Downloaded on July 13,2010 at 20:47:22 UTC from IEEE Xplore. Restrictions apply.

Inside this paper we investigate the sub-class of storage clusters where storage bricks use directly attached storage devices as persistent storage with a block-level interface. Furthermore, we assume that clients are not allowed to load proprietary drivers to support access to these devices. This is a key requirement for building open systems RAID-System, which are offered by vendors like Equallogic or LeftHand Networks [6].

The requirements concerning this storage cluster architecture differ significantly from approaches like Petal, where clients load an additional module that gives hints about the data location, and it differs from Ursa Minor that is based on the concept of object storage devices, where accessing clients also know where to access data. The architecture of the FAB-project is closely related to the architecture used inside this paper, but the publications do not consider the influence of the interconnection network between the peers on scalability.

The performance of this sub-class of storage clusters mainly depends on two different aspects: The ability to evenly spread data blocks and requests to the data among the storage bricks and the communication overhead between the peers. The communication between the peers is especially important, if the hard disks are as fast as or even faster then the communication links. This can occur if a set of disks inside each storage brick is used as internal RAID environment and the access pattern is sequential or if solid-state disks are used as persistent storage. Communication between peers is always necessary, if a peer needs to access data that is stored on another peer.

After giving a short introduction into the system architecture in section 2, we analyze the influence of inter-node communication on the scalability of the network in section 3. The calculations are based on the assumption that the interconnect is the bottleneck of the network and we show that the internode-communication has got a significant influence on the performance of a storage cluster.

The analytical results of this paper are complemented by measurement results for scalable storage clusters in section 4. The measurements have been performed on a high performance computing (HPC) cluster environment under Linux. Based on a storage cluster architecture that has been composed from publicly available components and the cluster volume manager V:Drive we show that the analytical results fit very well with reality. We will present the measurement results for up to 24 cluster nodes and 24 client nodes including data replication schemes.

2. System Architecture

The system architecture is based on three major components: iSCSI enterprise target driver for Linux, iSCSI initiator driver for Linux and the cluster volume manger V:Drive for Linux. The iSCSI Enterprise Target driver is used as block level interface to client computers and as interface to other storage brick nodes, which have to directly access physical disks on their peer nodes. The iSCSI initiator, which is based on the Cisco implementation for Linux and which is delivered as standard iSCSI initiator for Red-Hat AS 4.0, is used to access data on peer nodes.

The cluster volume manager V:Drive groups physical disks in storage pools. These storage pools are not accessed directly, but by the abstract concept of virtual volumes which are exported to the accessing client computers as iSCSI volumes. Each virtual volume can be concurrently accessed by an arbitrary number of client computers.

The capacity of each disk in a storage pool is partitioned into minimum sized units of contiguous data blocks, so called *extents*. The typical size of an extent varies between 4 MByte and 512 MByte. Inside this paper, we always use an extent size of 4 MByte. The extents are distributed among the storage devices according to a randomized data distribution strategy that guarantees an almost optimal distribution of the data blocks and data accesses across all participating disks in a storage pool. If storage bricks join or leave the storage cluster, the data is redistributed according to the randomized data distribution strategy. The number of extents that have to be redistributed after any change of the infrastructure is provable minimal [5]. Data is distributed over all physical disks of a storage pool. If data is replicated according to a RAID 1 scheme, the virtual volumes should be from different storage pools. Then, data consistency, synchronization, and recovery of a mirror inside V:Drive are fully cluster aware [4]. This is not (yet) the case for RAID 5, where the parity generation is not cluster aware. Nevertheless, we also use mirroring inside this paper in a way that all virtual volumes of a mirror are placed on the same storage pool, leading to a decreased reliability.

The core component of V:Drive is a clustered metadata appliance that stores and distributes information about the storage environment. This information includes the physical volumes, the storage pools and virtual volumes, and the set of extents which are already assigned to the different virtual volumes. Inside the storage cluster, each brick node is running a small driver module that presents its virtual volumes to the host operating system.

3. Communication Overhead between Peers

Scalability inside a storage cluster is bounded by a number of factors. Important aspects concerning the scalability are the ability of the data distribution to balance data and requests among the peers and the communication overhead between the peers that is induced by the exchange of data and information between the peers and the underlying network technology. In this section we will focus on commu-

Authorized licensed use limited to: NASA Goddard Space Flight. Downloaded on July 13,2010 at 20:47:22 UTC from IEEE Xplore. Restrictions apply.

nication between peers and we will assume that this kind of communication is only necessary to exchange data blocks and that only small amounts of metadata have to be exchanged between peers. Furthermore we will assume in a first step that the underlying data distribution scheme is able to evenly distribute data among the peers, so that all peers are able to participate according to their storage capacity and performance. An even distribution of accesses can either be achieved by striping the data over the peers or by using a distributed hash function that randomly distributes data over the peers [9] [5] [8].

Communication between two peers is necessary if a peer needs to read or write data stored on another peer. Figure 1 depicts a typical read inside a storage cluster. A client is connected to one storage brick inside the cluster that acts as iSCSI target for this client. This storage brick will be called master peer for this client in the following. In a first step, the client sends a read request to its master peer. If the master peer does not store the corresponding data block, it has to forward the request to a peer in step 2. The peer reads the data block from its cache or disk subsystem and returns the data block in step 3 to the master peer. In a last step, the master peer sends the resulting data block to the client.

This process does not only involve the forwarding of control messages between the peers, but also the movement of bulk data from the peer containing the data to the master peer. This movement seems to be unnecessary, because this data block could be (theoretically) sent directly from the data source to the client. Unfortunately, this is not possible for iSCSI and other protocols, which are based on the TCP/IP protocol. iSCSI requires for each communication to build up a socket between an iSCSI initiator and an iSCSI target. If the iSCSI- or TCP/IP-stack inside the client can not be adapted to the demands of the storage cluster, it is not possible to transparently move the target endpoint of the socket connection without the use of an intelligent intermediate switch or server [11].

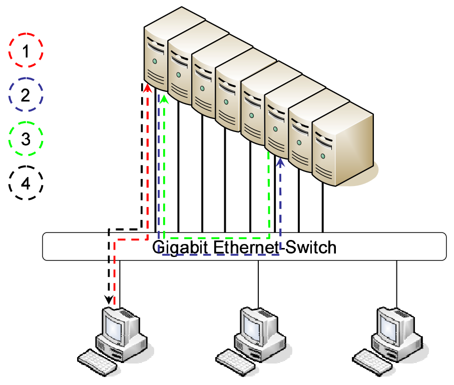

Figure 1. Typical communication pattern.

3.1. Analyzing the Expected Scalability

In a first step, we will analyze the expected scalability of storage clusters. We will assume that the performance is restricted by the interconnection network between the peers and from the peers of the storage to the clients. This assumption is valid for sequential access patterns as well as for random access patterns on Flash RAM-based hard disks. Furthermore, we will assume that each client is always connected with exactly one peer and that the clients are evenly distributed about the peers.

Assuming a fixed connection between a client and one storage brick inside the storage cluster and a striped or randomized data distribution scheme, the probability that an access can not directly be served by the master peer of a client growth linearly with the size of the storage cluster. If the cluster contains n nodes, only $1/n$-th of the requests could be served directly from a master peer. The remaining requests have to be forwarded from the master peer to peers containing the correct information.

In the following we will analyze the impact of this behavior on the scalability of the storage cluster. The network bandwidth that can be delivered from a single node system to its clients will be denoted by b. In the optimal case, the network bandwidth x_n delivered from one storage brick in a storage cluster with n nodes is equal to b and the total bandwidth delivered by n peers is $B_{total} = n \cdot x_n = n \cdot b$. In a real environment, we expect a behavior of type

$$B = f(n) \cdot \alpha \cdot b \tag{1}$$

as first order approximation, where $f(n)$ is a function expressing the scalability depending on the number of nodes n and α denotes a constant parallelization overhead (nearly) independent of n. In the investigated case, the parallelization overhead α is e.g. induced by a constant number of additional communication rounds between the peers exchanging requests.

Inside this extended abstract, we will consider the scalability of m-out-of-n codes. These codes have the advantage that the required redundancy to store data can become much smaller than for pure data replication; e.g. parity RAID with 4 data blocks and one parity block has an overhead of only 25%, compared with an overhead of 100% for mirroring with the same degree of data protection. This increase in storage efficiency normally includes a decrease in performance. The change of a sub-block requires that at least two sub-blocks are being read and two sub-blocks are being written to keep the parity block consistent. In the following, we will denote (due to consistency reasons inside this paper) the parameter n of the code as q and the parameter m as w.

We will show that this increase in storage efficiency can also be used to decrease network load if the environment

Authorized licensed use limited to: NASA Goddard Space Flight. Downloaded on July 13,2010 at 20:47:22 UTC from IEEE Xplore. Restrictions apply.

is able to write full stripes. In this case, the redundancy blocks can be calculated without reading data from storage and without straining network bandwidth. It is possible to calculate the usable bandwidth for writing data for full-duplex connections as:

$$\begin{aligned} b &= \max\left(x_n + \frac{w}{q}\cdot\frac{n-1}{n}\cdot x_n, \frac{w}{q}\cdot\frac{n-1}{n}\cdot x_n\right) \\ \Rightarrow x_n &= \frac{q\cdot n}{(q+w)\cdot n - w}\cdot b \end{aligned} \tag{2}$$

where the first term of the max-function depicts incoming communication and the second term outgoing communication from a storage node. Therefore, the overall bandwidth scales according to

$$B_{total} = \frac{q\cdot n^2\cdot\alpha\cdot b}{(q+w)\cdot n - w} \approx \frac{q\cdot n\cdot\alpha\cdot b}{(q+w)} \tag{3}$$

If q is equal to w, the term describes full-duplex writes without replication. If $w = k\cdot q$, the scaling becomes equivalent to the scaling of a k-fold replication scheme.

4. Measurements

The aim of our measurements is to experimentally evaluate the influence of interconnection technologies on the scalability of a storage cluster which is using direct attached storage devices. The measurements have been performed on a Linux computing cluster, so it has been possible to scale up to a large number of storage bricks.

To outline the influence of interconnects on scalability, we have used internal RAM disks to be able to abstract from the influence of the used storage media. To overcome resulting caching effects inside the client computers, we have developed a virtual RAM disk that is able to consistently store a defined part of the RAM disk address space in memory and just returns random blocks for the rest. Therefore it becomes possible to write a consistent boot block on an infinitely large RAM disk.

For each test, the adaption factor α has been calculated to minimize the average quadratic deviation from the expected results. If a test consist of p test runs for different numbers of nodes, x_i denotes the measured bandwidth for the i-th test run and ξ_i denotes the expected bandwidth for the same test, α is chosen in a way that it minimizes

$$\sum_{i=1}^{p}(x_i - \alpha\cdot\xi_i)^2 \tag{4}$$

4.1. Test Environment

To test the scalability of the storage cluster, up to 24 storage brick nodes and up to 24 clients, each equipped with two 1 GHz Pentium 3 CPUs and 512 MB RAM, have been used. Each of the storage brick nodes has exported a 1 TByte virtual RAM disk via iSCSI to all other nodes inside the storage cluster. Each server has been running RedHat Enterprise Linux AS 4 with a 2.6.9.42 kernel. The nodes have been connected by a 100 MBit/s Ethernet Cisco Catalyst 5509 switch that has been equipped with six WS-X5234-RJ45 24 node expansion modules. The backplane of the switch contains three busses, where each bus has a maximum throughput of 1.2 GBit/s. The maximum measured performance of each bus segment is 900 MBit/s that can only be observed for optimized communication patterns. Even if the cluster is based on elder technology, the resulting effects also apply to recent storage clusters.

For the scalability tests, each client has been connected to one node of the storage grid. The physical (RAM) disks of the brick nodes have been grouped in one single storage pool. For each client node we have created two virtual volumes from the storage pool which have been exported to the client node via iSCSI. The data of each virtual volume has been scattered over all physical (RAM) disks of the storage pool. For iSCSI-target mode, we have used the iSCSI Enterprise Target version 0.4.12 driver. For iSCSI-initiator mode, we have used the iSCSI-initiator module that has been deployed with RedHat AS 4 and which is based on a Linux-iSCSI(sfnet)-driver.

IOmeter has been used as benchmarking environment. It consists of a set of agents for Linux, called dynamos, which are working as load generator on the client computers. The dynamos are managed by a server program on a Microsoft Windows PC. If not mentioned otherwise, the maximum number of outstanding IOs for each client has been set to 16, the access size has been set to 32 KByte, and the performance has been measured for 5 minutes for sequential writes after a ramp-up time of 30 seconds.

4.2. Measurement Results

Local Performance Many parallel solutions are able to scale performance in the number of nodes of the environment from 2 to n nodes, but have to cope with a significant parallelization overhead. This parallelization overhead often leads to a performance decrease for smaller environments compared to a local solution. Besides the examined overhead of the inter-node communication, this overhead can be induced in our case e.g. by the virtualization layer or network protocol stack.

In this section, measurement results for the performance of a local solution will be presented, where both client and

Authorized licensed use limited to: NASA Goddard Space Flight. Downloaded on July 13,2010 at 20:47:22 UTC from IEEE Xplore. Restrictions apply.

server are on the same computer system. Furthermore we investigate the influence of the virtualization layer and the iSCSI-communication between one server and one client. In all cases, two workers are accessing two volumes.

In the first case, the dynamo agents directly access the virtual RAM disk on the same node. The maximum performance of the RAM disk is a sequential read throughput of 360 MByte/s for 32 KByte blocks and it can deliver up to 11,429 32 KByte random I/Os per second. The sequential write performance drops to 136 MByte/s. Both CPUs are under significant load. The situation changes slightly when a virtualization environment is put between the RAM disk and the IOMeter agent. The sequential write throughput drops to 120 MByte/s and the IO-rate drops by the factor 3/4. The reason is based on communication with the metadata appliance, which imposes additional delays for all first accesses to new regions.

In the next case, the server has been connected via iSCSI with a client computer. To directly measure the influence of the iSCSI communication between two computers, the server exports two virtual disk that only access the RAM disk. The throughput for sequential write accesses is 10.02 MByte/s for a 100 MBit/s Ethernet connection. The random I/O write performance is 313 I/Os per second or 9.78 MByte/s is nearly as fast as the sequential throughput. The last test measures the case when the RAM disk is exported as two virtual volumes. The sequential write performance drops slightly to 9,1 MByte/s, while the random I/O write performance decreases to 195 write I/Os. The decrease is based on the communication with the metadata server.

Scalability using RAID 1 The next test series is based on virtual RAM disks again and investigates the behavior of storage clusters applying data replication (additional test results, e.g. without data replication or for real disks are given in the full version of this paper). The RAM disks of all storage bricks are grouped inside a single storage pool. Each virtual disk derived from the storage pool has got a capacity of 40 GByte and two virtual volumes are combined to one mirror volume. Each client computer is again connected with exactly one grid node and imports two mirror volumes. Therefore, 96 virtual volumes are set up for the 24 node test. Writes are send to both virtual volumes of a mirror volume, reads are performed by an arbitrary of both (see also [4]).

The measured performance for write tests only increases according to Equation 3 by a factor $1/3$. After scaling well from two nodes to 16 nodes, the performance for 24 storage nodes lacks behind the expected performance. This is based on the deployed switch. The measured performance of 42 MByte/s produces additional, internal traffic of 56 MByte/s between the three leaf boards, saturating the backplane with an overall traffic of 790 MBit/s. The ex-

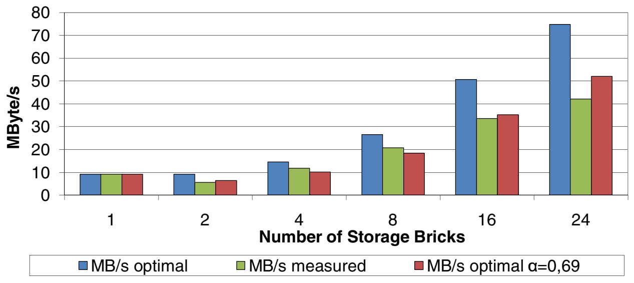

Figure 2. Scalability of mirrored RAM disks.

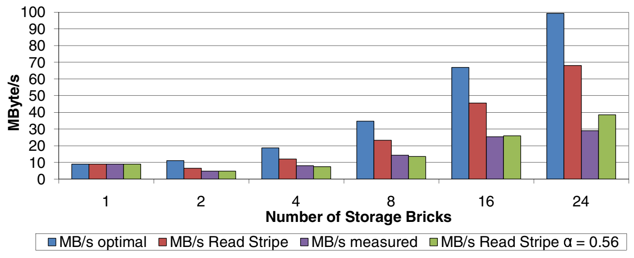

Figure 3. Scalability of RAID 5.

pected performance of 52 MByte/s would already produce an overall traffic of 970 MBit/s on the backplane. Therefore, it is important to consider the internal traffic between the peers that can become much bigger than the external traffic. The factor α has been set to 0.69 to minimize the error, neglecting the test for 24 storage nodes[1].

Scalability of RAID 5 The theoretical assumptions leading to Equation 3 promise that the network load can be significantly reduced, compared to a k-replication of the data, by using m-out-of-n codes. Based on a single server write throughput of 9 MByte/s, this would theoretically lead to a throughput of up to 100 MByte/s for a 24 nodes 4-out-of-5 storage cluster, compared to a theoretical maximum throughput of 73 MByte/s for a 24 nodes cluster that mirrors data. Besides this expectations, the measured write performance lags behind the expected performance and even behind the measured performance for Mirroring.

The reason is the special handling of request inside the iSCSI target driver inside the storage nodes. Each 32 KByte write request is split into 4 KByte requests which are successively handled by the underlying page cache layer and block layer. Therefore, the first block of each stripe is handled as a new stripe and requires to fetch the remaining blocks of the stripe from the other peers to calculate the new parity block, leading to additional 32 KByte of read requests. Furthermore, 40 KByte of data have to be written

[1]The used iSCSI configuration producing the test results has not been optimized and better results can be achieved which would lead to a higher α-value.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: NASA Goddard Space Flight. Downloaded on July 13,2010 at 20:47:22 UTC from IEEE Xplore. Restrictions apply.

to the peers. This leads to the following calculation for the used 4-out-of-5 code:

$$\begin{aligned} b &= \max\left(\frac{13\cdot n-9}{4\cdot n}\cdot x_n, \frac{9\cdot n-9}{4\cdot n}x_n\right) \\ \Rightarrow x_n &= \frac{4\cdot n}{13\cdot n-9}\cdot b \qquad (5) \end{aligned}$$

and therefore the environment scales according to

$$B_{total} = \frac{4\cdot n^2\cdot\alpha\cdot b}{13\cdot n-9} \approx \frac{4\cdot n\cdot\alpha\cdot b}{13} \qquad (6)$$

Using an adaption constant α of 0.56 leads to a nearly perfect approximation of the measured behavior. Again, the 24 node test requires too much communication to be able to scale according to the predicted results.

5. Conclusion

Storage bricks offer an interesting opportunity to scale small storage appliances, so-called storage bricks, to large-scale enterprise storage systems. As shown inside this paper, the interconnects can easily become the limiting performance factor for sequential accesses, especially if data has to be replicated. Nevertheless, storage clusters are really able to scale performance in the number of nodes. This performance increase is not only based on a larger number of spindles, but also on more communication interfaces and larger, aggregated caches. Summarized, storage clusters are able to scale performance and capacity while delivering a high degree of reliability and are able to overcome limitations imposed by centralized storage architectures.

References

[1] M. Abd-El-Malek, W.V. Courtright, C. Cranor, et al. Ursa Minor: Versatile Cluster-based Storage. In *Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST)*, February 2005.

[2] P. T. Breuer, A. Marin Lopez, and A. G. Ares. The Network Block Device. *Linux Journal*, 73, 2000.

[3] A. Brinkmann and S. Effert. Cost effectiveness of Storage Grids and Storage Clusters. In *Proceedings of the 15th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2007.

[4] A. Brinkmann, S. Effert, M. Heidebuer, and M. Vodisek. Distributed MD. In *Proceedings of the 3rd International Workshop on Storage Network Architecture and Parallel I/Os (SNAPI)*, 2005.

[5] A. Brinkmann, K. Salzwedel, and C. Scheideler. Compact, Adaptive Placement Schemes for Non-Uniform Distribution Requirements. In *Proceedings of the 14th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2002.

[6] Equallogic. PS Series - Intelligent iSCSI Storage Arrays. Product Brochure, 2005.

[7] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[8] R. J. Honicky and Ethan L. Miller. Replication Under Scalable Hashing: A Family of Algorithms for Scalable Decentralized Data Distribution. In *Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2004.

[9] D. Karger, E. Lehman, T. Leighton, et al. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In *Proceedings of the 29th ACM Symposium on Theory of Computing (STOC)*, 1997.

[10] E. K. Lee and C. A. Thekkath. Petal: Distributed Virtual Disks. In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 1996.

[11] V. Olaru and W.F. Tichy. On the Design and Performance of Kernel-level TCP Connection Endpoint Migration in Cluster-Based Servers. In *Proceedings of the 5th IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, 2005.

[12] K. W. Preslan, A. P. Barry, J. Brassow, et al. Implementing Journaling in a Linux Shared Disk File System. In *Proceedings of the 17th IEEE Conference on Mass Storage Systems and Technologies (MSST)*, 2000.

[13] A. Rajasekar, M. Wan, R. Moore, and T. Guptil. Data Grids, Collections and Grid Bricks. In *Proceedings of the 20th IEEE Conference on Mass Storage Systems and Technologies (MSST)*, 2003.

[14] Y. Saito, S. Frølund, A. C. Veitch, A. Merchant, and S. Spence. FAB: building distributed enterprise disk arrays from commodity components. In *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2004.

[15] F. B. Schmuck and R. L. Haskin. GPFS: A Shared-Disk File System for Large Computing Clusters. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST)*, 2002.

[16] F. Tomonori and O. Masanori. Analysis of iSCSI Target Software. In *Proceedings of the 2nd International Workshop on Storage Network Architecture and Parallel I/Os (SNAPI)*, 2004.

[17] W.Y.H. Wang, H.N. Yeo, Y.L. Zhu, and T.C. Chong. Design and development of Ethernet-based storage area network protocol. In *Proceedings of the 12th IEEE International Conference on Networks (ICON)*, 2004.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: NASA Goddard Space Flight. Downloaded on July 13,2010 at 20:47:22 UTC from IEEE Xplore. Restrictions apply.