Evaluation of iPVFS: A High Performance Parallel File System over iSCSI for Cluster Computing

Li Ou* and Xubin He*
Tennessee Technological University, Cookeville, TN, 38505, USA

Abstract

In this paper we propose a high performance parallel file system over iSCSI (iPVFS) for cluster computing. iPVFS provides a cost-effective solution for heterogeneous cluster environments by dividing a set of I/O servers into two groups. One group, with higher performance, serves as I/O nodes, while the other group, with relatively lower performance, serves as storage target nodes. This combination provides a higher aggregate performance because of the cooperative cache among different target nodes. We have developed a model to analyze iPVFS. Our simulation results show that using the same number of nodes, iPVFS outperforms PVFS for both small and large requests under most cases.

Key Words: Parallel file system, iSCSI, distributed I/O, cluster computing, cache.

1 Introduction

Cluster computing [20] has become one of the most popular platforms for high-performance computing today. Similar to traditional parallel computing systems, the I/O subsystems of clusters are a bottleneck to overall system performance. An efficient way to alleviate the I/O bottleneck is to deploy a parallel file system, which utilizes the aggregate bandwidth and capability of existing I/O resources on each cluster node, to provide high performance and scalable storage service for cluster computing platforms.

The Parallel Virtual File System (PVFS) [4], developed at Clemson University and Argonne National Lab, provides a starting point for I/O solutions in Linux cluster computing. Several recent works have studied how to improve parallel I/O performance of PVFS. A kernel level cache is implemented in [24] to reduce response time. In [14, 18] several scheduling schemes are introduced in I/O nodes to reduce disk seek times. A better interface is presented in [6] to optimize noncontiguous I/O access performance. CEFT-PVFS [28] increases the availability of PVFS, while still delivering a considerably high throughput. In [10] software and hardware RAIDs are adopted in PVFS I/O nodes to achieve higher aggregate I/O bandwidth.

In this paper we propose a parallel file system, iPVFS, based on PVFS [4] and iSCSI (Internet SCSI) [1, 15], for cluster computing platforms. We have designed the iPVFS and developed a model to simulate it. We compare the I/O response time of iPVFS with the original PVFS under various configurations. The results show a considerable performance gain of iPVFS over PVFS.

The rest of this paper is organized as follows. Background information is presented in Section 2. Section 3 gives the architecture of iPVFS. In Section 4, we describe a queuing model for iPVFS. Simulation and I/O response time analysis are presented in Section 5. We examine related work in Section 6. Section 7 concludes the paper.

2 Background Review

2.1 PVFS

PVFS [4] is a popular parallel file system for Linux cluster computing. It provides high-speed access to file data for parallel applications. Figure 1 [4] shows a typical PVFS architecture and the main components. There are three types of nodes in PVFS. The metadata node maintains the metadata of the file system. I/O nodes store file data on local storage devices. Clients, or compute nodes, read or write file via sending requests to the metadata server and I/O servers.

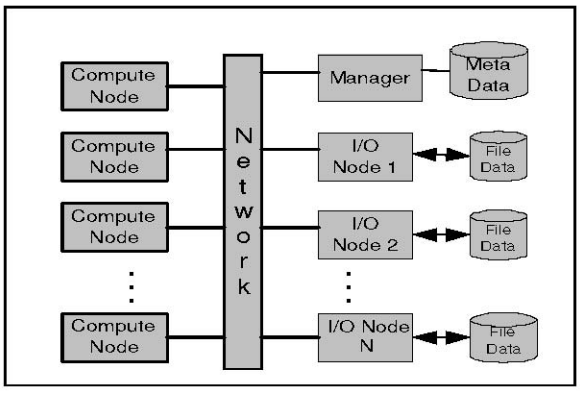


Figure 1: PVFS system diagram. The number of I/O nodes is *N*. Since all storage nodes are used as I/O nodes, the total number of nodes is *N*.

* Department of Electrical and Computer Engineering. Email: {lou21, hexb}@tntech.edu.

2.2 iSCSI and iRAID

iSCSI [1, 15] is a newly emerging protocol with the goal of implementing the storage area network (SAN) technology over the better-understood and mature network infrastructure: the Internet (TCP/IP). iSCSI encapsulates SCSI commands/data within TCP/IP connections using Ethernet, which brings economy and convenience, as SANs now can be implemented using less expensive, easily manageable components. iSCSI provides a block level data interface which is independent of any file systems.

iRAID [9] is introduced to improve the performance and reliability of iSCSI storage systems by organizing the iSCSI storage targets in such a way similar to RAID [5], using striping and rotated parity techniques. In iRAID, each iSCSI storage target is a basic storage unit in the array. All the units in the array are connected through a high-speed network. iRAID provides a direct and immediate solution to boost iSCSI performance and improve reliability.

3 Architecture of iPVFS

In a typical cluster environment nodes can be used as block-level storage providers and be grouped together to form distributed RAID system [21]. Combining iRAID and PVFS together improves parallel I/O performance because iRAID provides high bandwidth storage level services for parallel file systems.

The above observations motivate us to propose iPVFS to improve I/O performance by utilizing iRAID to provide local storage for the I/O nodes of PVFS. In iPVFS, each I/O node includes an iSCSI Initiator, which is supported by multiple target nodes to form an iRAID group (Figure 2). With iRAID, the I/O nodes stripe PVFS data across multiple target nodes. The local I/O performance of I/O nodes is improved because of possible parallel accesses to data blocks. In iPVFS, all nodes, except the compute nodes and metadata nodes, are divided in two groups. The nodes in the first group are configured as I/O nodes of a PVFS system, providing file level services for cluster computing platforms. The others are used as target nodes of the iRAID system, exposing block level services to upper-level I/O nodes.

In traditional iSCSI design, the I/O requests are serialized by the SCSI scheduler of the operating system kernel. This is reasonable for traditional hard disks, because internally such devices cannot support concurrent accesses. In iPVFS, since several target nodes within an iRAID group are exclusively designated for one I/O node, we modified the scheduler in the I/O nodes, so the multiple requests directed to different targets could be concurrently submitted by the SCSI drivers. Since all requests from an iSCSI initiator have to be sent to targets through a single network card (in the current design), the requests will be queued at the sending buffer of the network interface, no matter how many requests are submitted concurrently through the SCSI driver. Even with the queuing overhead of the network interface, the throughput of new scheduler is better than the old one, because it makes the network and targets as busy as possible and thus utilizes the potential of the iRAID system. Parts of the requests may be quickly satisfied by the buffer caches of the target nodes. If the requests must be sent to the disks of the target nodes, a local scheduler is used to serialize the concurrent accesses.

In iPVFS, buffer caches are organized as two-level cache hierarchies: the upper level caches reside in I/O nodes, and

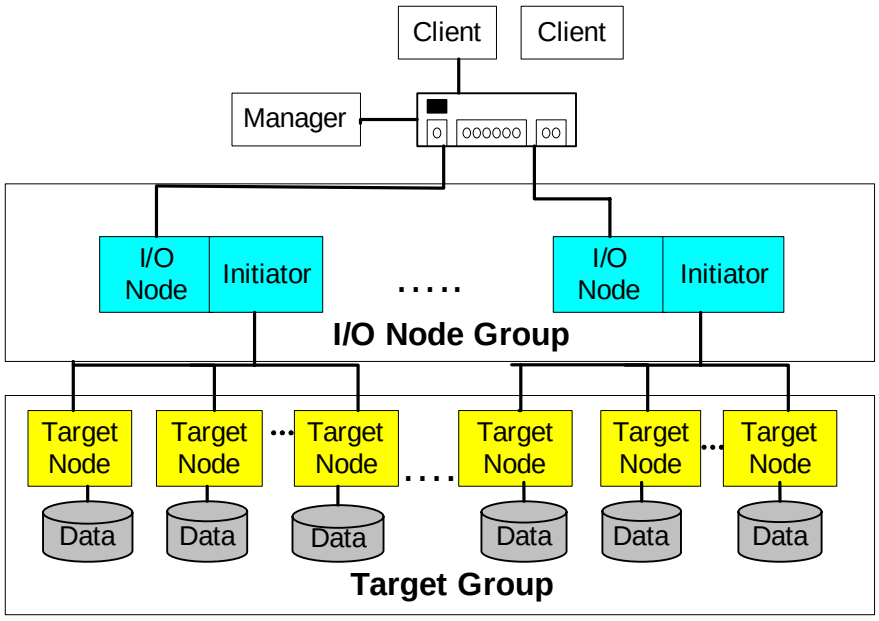


Figure 2: iPVFS Architecture. The number of I/O nodes is N_{io}, the number of target nodes of each iRAID group serving for one I/O node is N_t, and the total number of nodes is $N = N_{io}(1+N_t)$.

the lower level caches reside in target nodes. We refer to the upper level storage client caches as L1 buffer caches and the lower level storage caches as L2 buffer caches [27]. L1/L2 buffer caches are very different from L1/L2 processor caches because L1/L2 buffer caches refer to main-memory caches distributed in multiple machines. The access patterns of L2 caches show weak temporal locality [3, 8, 27] after filtering from L1 caches, which implies that a cache replacement algorithm, such as LRU, may not work well for L2 caches. Additionally, local management algorithms used in L2 caches are inclusive [25], which try to keep blocks that have been cached by L1 caches, and waste aggregate cache space. Thus, in iPVFS, we introduce a Unified Multiple-Level Cache (*uCache*) algorithm [16] to manage aggregate cache space efficiently and increase cumulative hit ratios.

The *uCach*e algorithm unified together the cache spaces of the I/O node and the corresponding target nodes. The LRU algorithm is used in the L1 cache of an I/O node. The *FBE*R algorithm [16] is deployed in the L2 caches of the target nodes to make them exclusive to the L1 cache. There is no built-in cache consistency mechanism for the I/O nodes and corresponding target nodes. In iPVFS, a group of target nodes are exclusively used by one I/O node, and it is impossible for two cache lines in different I/O nodes to map to the same disk block. We keep the consistency semantics of PVFS for the client caches. It does not implement POSIX semantics, but in typical parallel I/O applications, each client is responsible for reading and writing part of the non-overlapping I/O data (for example, stride I/O), and simple consistency semantics improve system performance. In the case that multiple clients really need to read and/or modify the same data blocks, applications may implement their own consistency semantics, for example, using the synchronization mechanisms of MPI [22].

4 A Queuing Model for iPVFS

In a cluster environment, if the number of servers is given, the utilizations of servers of PVFS and iPVFS are different. In PVFS all servers, except the metadata server, are used as I/O nodes, but in iPVFS, as mentioned in Section 3, some nodes are configured as target nodes to speed up performance of I/O nodes. With a given number of nodes, which design could deliver better I/O performance?

We develop a queuing model to analyze the performance of PVFS and iPVFS with respect to average I/O response time (Figure 3). The model includes two queues: one for I/O nodes and the other for target nodes. We use some assumptions made by [7]. The number of I/O requests follows a Poisson process with a mean arrival rate of λ. The loads on I/O nodes and target nodes are balanced.

We assume that there are N storage nodes in a cluster. The number of I/O nodes is N_{io}. The number of target nodes in an iSCSI group serving one I/O node is N_t. In PVFS, $N = N_{io}$. In iPVFS, $N = N_{io}(1 + N_t)$, since there are N_{io} groups, with each group consisting of one I/O node and N_t target nodes. For each I/O node, the arrival request rate is $P_i\lambda$, where P_i is the probability that the request is redirected to I/O node i. When the request data size L_{ip} is smaller or equal to the striping size B, P_i is equal to $1/N_{io}$.

When the request data size is larger than $N_{io}B$, the request is sent to all I/O nodes and P_i is equal to 1. Thus, the typical range of P_i is $[1/N_{io},1]$, and we calculate P_i by $P_i = \min(\lfloor L_{ip} / B \rfloor / N_{io},1)$. If a request is not satisfied by the I/O node cache, it is redirected to the iRAID initiator hosted in the I/O node, and the effective arrival rate to each iRAID system is $\lambda_i = (1 - h_{ioc}) * P_i * \lambda$, where h_{ioc} a cache hit ratio for an I/O node.

After requests arrive to the iRAID system, the initiator redirects them to target nodes. For each target node, the arrival rate is $P_{ij}\lambda_i$, where P_{ij} is the probability that the request is sent to a target node j, which belongs to the iRAID system of I/O node i. Similar to the probability that requests are redirected to I/O nodes, the typical range of P_{ij} is $[1/N_t,1]$, and we calculate P_{ij} by $P_{ij} = \min(\lfloor L_{ir} / B \rfloor / N_t,1)$, where L_{ir} is the request data size from the I/O node to the target node. If the request is not satisfied by the target node cache, it is redirected to the local disks, and the effective arrival rate to each disk is $\lambda_{ij} = (1 - h_{tc}) * P_{ij} * \lambda_i$, where h_{tc} is the cache hit ratio for the target node.

Request delays are mainly caused by network and memory delays independent of cache hits and misses. We assume that the network service time and the cache service time are exponentially distributed [7] with average times being T_{ionet} and T_{ioc}, respectively.

$$T_{ionet} = \frac{L_{ip}}{N_{io} * BW_{net}}$$

$$T_{ioc} = \frac{L_{ip}}{N_{io} * BW_{cache}}$$

where BW_{net} and BW_{cache} are bandwidth of the network and memory caches, respectively. Therefore, the request residence times in the network and cache are modeled using the M/M/1 queuing model [7], and the average residence times of request in the network and cache of I/O nodes are modeled as:

$$W_{ionet} = \frac{T_{ionet}}{1 - P_i * \lambda * T_{ionet}}$$

$$W_{ioc} = \frac{T_{ioc}}{1 - P_i * \lambda * T_{ioc}}$$

If a request is not satisfied by the memory cache, it must be handled by iRAID system, with a probability of $1 - h_{ioc}$. Thus,

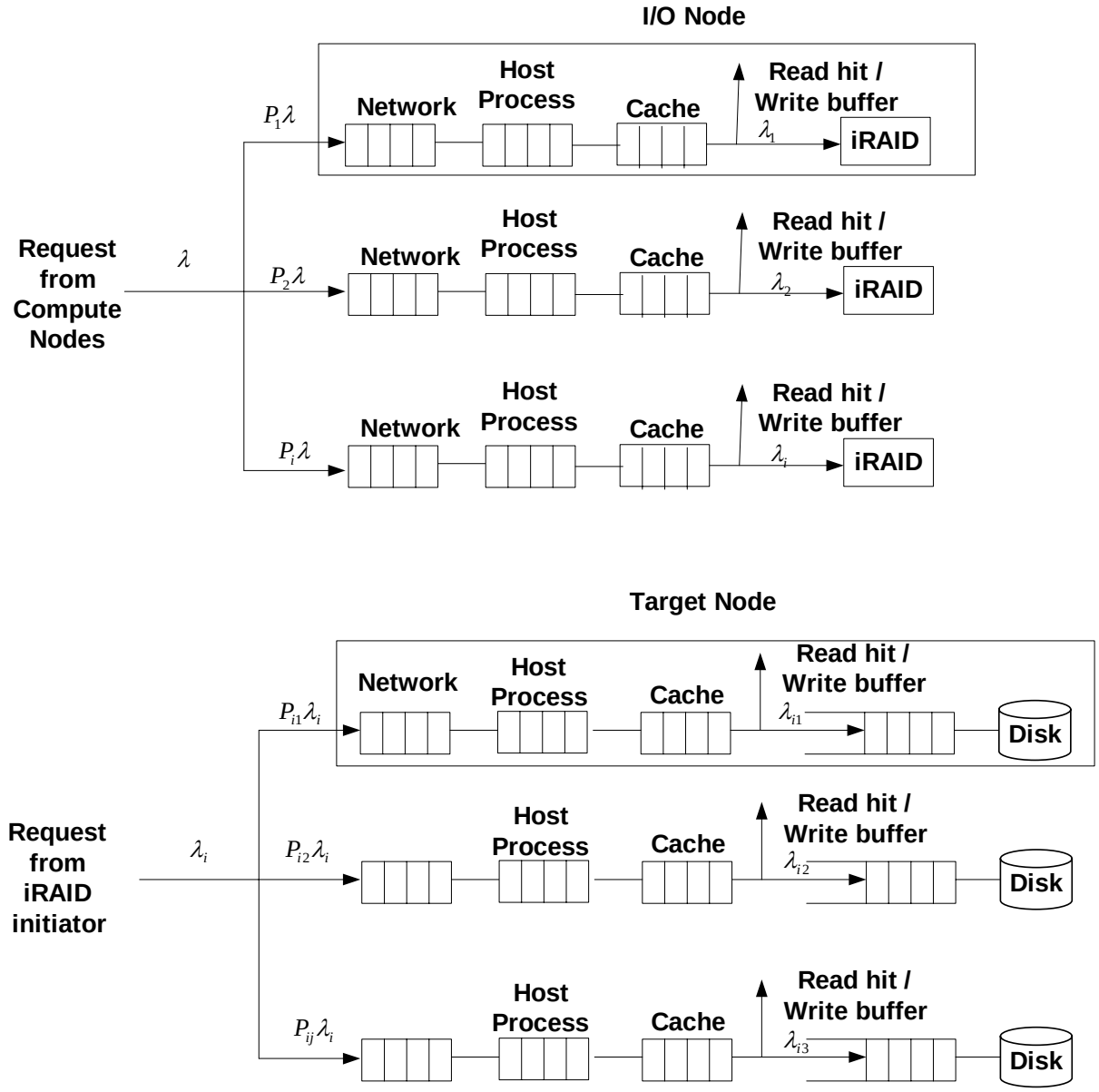


Figure 3: Queuing Model for iPVFS

the average response time of the iPVFS system is expressed as:

$$T = W_{ionet} + W_{ioc} + (1 - h_{ioc}) * W_{iRAID} + W_{node}$$

where W_{iRAID} is the request residence time in the iRAID system, and W_{node} is the processing overhead of each node. The overhead includes the time of protocol processing, and the time that messages go from the receiving buffer to the sending buffer of the network interface. Normally an I/O node acts as a server with a certain service rate. We assume that the server processing time is exponentially distributed with the average time T_{node}, and the request residence time in the node, W_{node}, is modeled using the M/M/1 queuing model.

$$W_{node} = \frac{T_{node}}{1 - P_{ij} * \lambda_i * T_{node}}$$

Although each iRAID system is designated to one I/O node, additional overheads are introduced when requests travel through the network between I/O nodes and target nodes, especially at the time multiple requests from the modified SCSI scheduler are queued at the sending buffer of the network interface, as explained in Section 3. We assume that the network service time of the iRAID is exponentially distributed with the average time T_{tnet}, and the request residence time in the network, W_{tnet}, is modeled using the M/M/1 queuing model.

$$T_{tnet} = \frac{L_{ir}}{N_t * BW_{net}}$$

$$W_{tnet} = \frac{T_{tnet}}{1 - P_{ij} * \lambda_i * T_{tnet}}$$

Another possible overhead of iRAID is the service time of memory caches in the target nodes. We assume that the cache service time is exponentially distributed with average times T_{tc}, and the request residence time in the cache of iRAID, W_{tc}, is modeled using the M/M/1 queuing model.

$$T_{tc} = \frac{L_{ir}}{N_t * BW_{cache}}$$

$$W_{tc} = \frac{T_{tc}}{1 - P_{ij} * \lambda_i * T_{tc}}$$

If a cache miss occurs, the request is redirected to the hard disks, with a probability of $1 - h_{tc}$. The real disk service times are generally distributed [13] so we adopt the M/G/1 model to analyze the disk response time.

$$W_{disk} = \frac{\lambda_{ij} * E(T_{diskio}^2)}{2 * (1 - \lambda_{ij} * E(T_{diskio}))} + E(T_{diskio})$$

where $E(T_{diskio})$ is the average disk access time for a request. Thus, the average response time for an iRAID system is expressed as:

$$W_{iRAID} = W_{tnet} + W_{tc} + (1 - h_{tc}) * W_{disk} + W_{node}$$

5 I/O Response Time Analyses

Based on the above queuing model, we simulate and compare the response times of iPVFS and original PVFS under various workloads and application environments for both fixed and dynamic cache hit rates. Some parameters are as follows: the available network bandwidth is about 75MB/s and the memory access rate is about 500MB/s. A 64KB data striping size is chosen for both PVFS and iRAID. The disks are Seagate SCSI disks (model ST318452LW), with a data transfer rate of 51MB/s, an average seek time of 3.8ms, an average latency of 5ms, and 18,497 cylinders.

5.1 I/O Response Time for Fixed Cache Hit Rates

We assume that the cache hit rates of both I/O nodes and target nodes are fixed, and that the read request percentage is 53 percent, which is based on our observations of existing disk I/O traces and previously published data [11, 19].

In PVFS, all storage nodes are used as I/O nodes $N = N_{io}$. In iPVFS, we have various choices. In this section, for each I/O node, we configure an iRAID group with two target nodes $N_t = 2$. Other choices are discussed in Section 5.1.3.

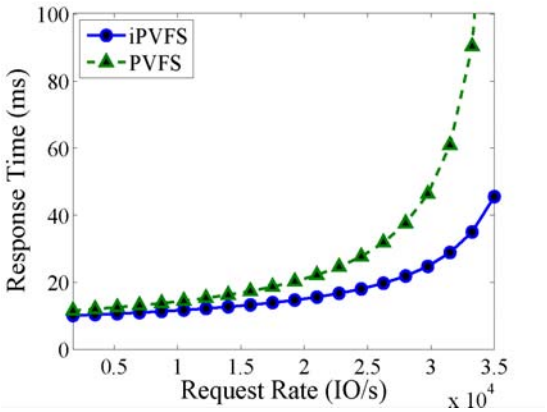
Given a fixed number of storage servers for both PVFS and iPVFS, the number of I/O nodes in iPVFS is always smaller than that of PVFS since some nodes are used as targets, so lower cache hit rates are predicted for I/O nodes in iPVFS when the memory sizes of each node are the same. Since the hit ratios of I/O nodes are more important than those of target nodes, the cache sizes of I/O nodes are expected to be larger than those of target nodes. In our simulation, we compensate for the hit ratios of I/O nodes by adding more caches for I/O nodes, but we maintain the same cache size for the whole system by reducing the cache sizes of the target nodes. Even with the configuration, we still expect that the hit ratios of I/O nodes in iPVFS are lower than I/O nodes for PVFS in real applications, so in the following simulation, we set the cache ratios of iPVFS to be 0.1 lower than their PVFS counterparts.

5.1.1 Small I/O Requests. Small requests have sizes of at most the striping block size 64KB, so that each request can be satisfied by a single node. We assume that the request data size is equal to 64KB. To validate the model with a large scale system, we set the number of storage nodes to 300 (i.e., $N = 300$). In iPVFS, the number of I/O nodes is 100 (i.e., $N_{io} = 100, N_t = 2$), and the arrival rates to an I/O node and a target node are λ / N_{io} and $\lambda / (N_{io} N_t)$, respectively. In PVFS, the arrival rate to an I/O node is λ / N.

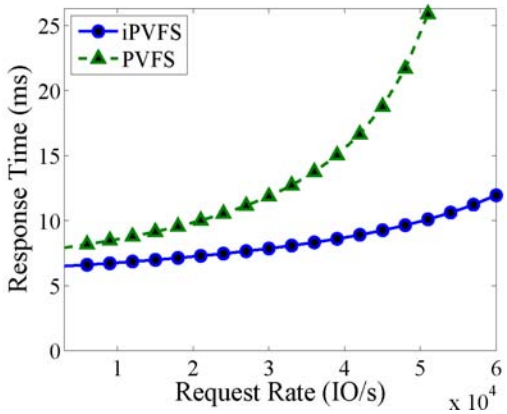
Figure 4 compares the performance of iPVFS and PVFS. It shows that the average I/O response time increases steadily with the increase of the request rate. At the point where PVFS is saturated by a large number of requests, iPVFS still provides acceptable service. We believe that the cache makes the difference, since compared to the network, the disk access time accounts for most of the total response time.

In PVFS, when cache misses occur in I/O nodes, the requests must be redirected to disks. In iPVFS, in case of cache misses in I/O nodes, the requests are first directed to the target nodes, in which they may be satisfied by the target node caches. When the hit ratios are high enough, the disk access time is not dominant because most requests are satisfied by caches. Thus, the network bottleneck is the main factor affecting the performance. Since the number of I/O nodes in iPVFS is smaller than that in PVFS, there is a point where the network is saturated by the requests in iPVFS.

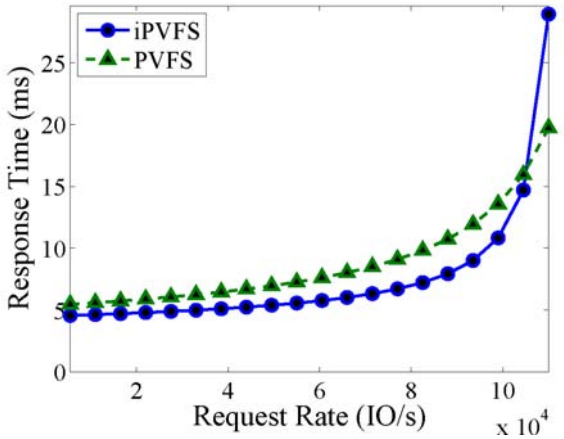
5.1.2 Large I/O Requests. If the data request size is much larger than the striping block size 64KB, each request is striped over several nodes. The extreme case is that the request is large enough to be striped over all nodes. For iPVFS, the arrival rates to an I/O node and to a target node are all λ; for PVFS, the arrival rate to an I/O node is λ too. To ensure that the request is large enough to be striped over all nodes, we did not choose a real large scale system; otherwise, the request size did not



(a) Cache hit ratios of iPVFS and PVFS are 0.2 and 0.3, respectively.



(b) Cache hit ratios of iPVFS and PVFS are 0.5 and 0.6, respectively.



(c) Cache hit ratios of iPVFS and PVFS are 0.7 and 0.8, respectively.

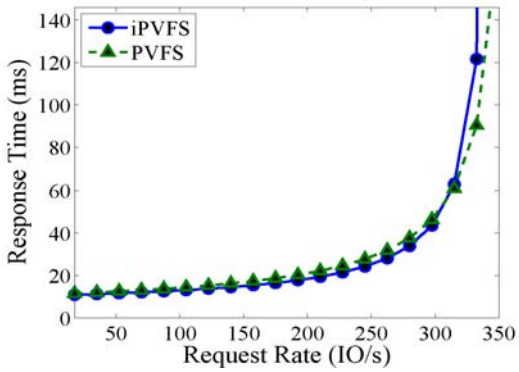
Figure 4: I/O response times with various cache hit ratios for small requests. The request size is 64KB, with $N_{io} = 100, N_t = 2$ and $N = 300$

choose a real large scale system; otherwise, the request size is too large to be real. Instead, we use a 640KB request size, 30 storage nodes ($N = 30$), and an N_{io} of 10 in iPVFS (Figure 5).

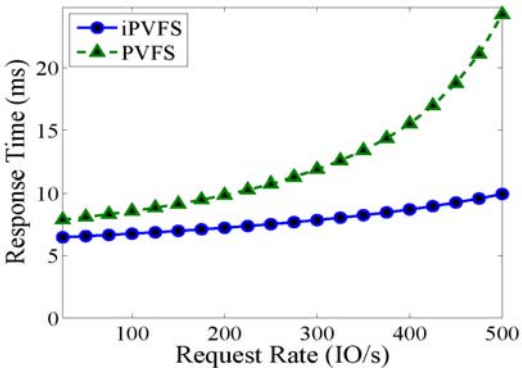
In large I/O request situations, each node receives much more requests, which quickly saturate the cache; thus, most requests must be redirected to disks. If the hit ratio is very low (Figure 5(a)), the disk access is the key factor. iPVFS is easier to saturate because a single server experiences a larger request rate than in PVFS. When the hit ratio is high enough, the network bottleneck dominates the system performance (Figure 5(c)), similar to the simulations for small requests (Section 5.1.1).

5.1.3 Performance of Various iPVFS Configurations. When the number of storage nodes is fixed for iPVFS, how should we organize our system to achieve maximum I/O performance? We may have various configurations because the selection of N_{io} and N_t is not unique, as long as the equation $N = N_{io}(1+N_t)$ is satisfied. We measure the saturated request rate at which the system cannot accept and service requests any more. We choose the same simulation parameters (Section 5.1.1 and Section 5.1.2) for small requests and large requests, but compare the saturated request rates of all possible N_{io} values in a system (Figure 6).

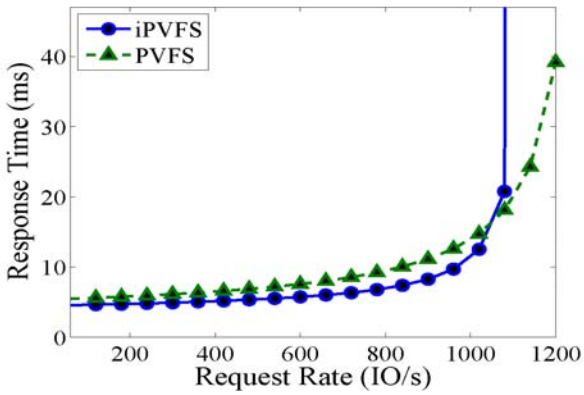
The request rates of large requests to each server are much higher than those of small requests. Generally the larger number of I/O nodes provides better performance, since the request rate of an I/O node decreases with an increasing number of I/O nodes. With small requests, while the request rates are balanced by multiple I/O servers, the hit ratios begin to influence the performance of different system configurations. Generally, performance improves with a larger number of I/O nodes, but the peak performance does not always occur in the configuration in which as many as possible storage nodes are configured as I/O nodes. The extreme case is when half of the nodes are I/O nodes, and each I/O node is only supported by one target node (most right points of Figures 6(a) and 6(b)).



(a) Cache hit ratios of iPVFS and PVFS are 0.2 and 0.3, respectively.

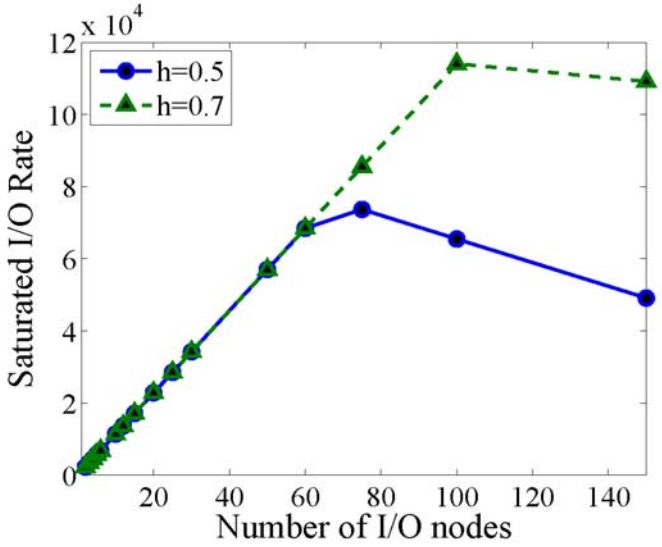


(b) Cache hit ratios of iPVFS and PVFS are 0.5 and 0.6, respectively.

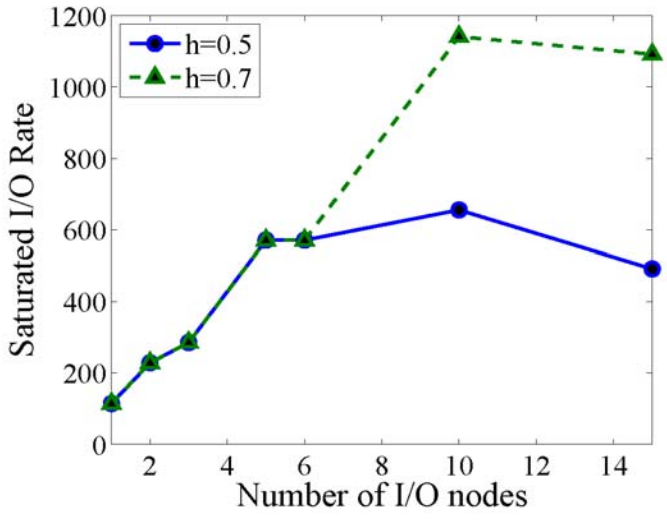


(c) Cache hit ratios of iPVFS and PVFS are 0.7 and 0.8, respectively.

Figure 5: I/O response times with various cache hit ratios for large requests. The request size is 64KB, with $N_{io} = 10, N_t = 2$ and $N = 30$



(a) Small request. The request size is 64KB and N=300



(b) Large request. The request size is 640KB and N=30

Figure 6: Saturated I/O request rates for various iPVFS configurations under different cache hit ratios. *h* is the cache hit ratio

This configuration does not provide best performance for both large and small requests, because the iRAID configuration with only one target node could not take advantage of parallel access; and thus degrade the I/O performance.

5.2 I/O Response Time for Dynamic Cache Hit Rate

In real application environments, cache hit rates are dynamically influenced by many factors. To obtain more accurate performance comparisons, it is necessary to predict cache hit rates dynamically for both iPVFS and PVFS.

5.2.1 Simulation Methodology. We use trace-driven simulations to evaluate the cumulative hit ratios of iPVFS and PVFS. We have developed a simulator to simulate two-level buffer cache hierarchies with multiple clients and storage servers. For PVFS, LRU is used as the replacement algorithm in the caches of I/O nodes, but for iPVFS, *uCache* [16] is implemented to manage the two-level cache hierarchy. We assume a cache block size of 64KB. The striping size is 64KB. The traces for the simulations are described in Table 1. The hit ratios generated from the simulator are used in our two-level queuing model to calculate average response times.

Table 1: Characteristics of traces

Trace	Clients	IOs (millions)	Volume
Cello92	4	0.5 per day	10.4GB
HTTPD	7	1.1	0.5GB
DB2	8	3.7	5.2GB

The *HP Cello92* trace was collected at *Hewlett-Packard* Laboratories in 1992 [19]. It captured all L2 disk I/O requests in *Cello*, a timesharing system used by a group of researchers to do simulations, compilation, editing, and e-mail, from April 18 to June 19. The Cello92 is a serial workload. To test a parallel file system, we use trace files collected within four days as the workload for one client, and simulate multiple clients using trace files collected within one month.

The *HTTPD* workload was generated by a seven-node *IBM SP2* parallel web server [12] serving a 524MB data set. Multiple HTTP servers share the same set of files.

The *DB2* trace-based workload was generated by an eight-node *IBM SP2* system running an *IBM DB2* database application that performed join, set and aggregation operations on a 5.2GB data set. [23] used this trace in their study of I/O on parallel machines. Each *DB2* client accesses disjoint parts of the database. No blocks are shared among the eight clients. We use the *DB2* workload as the low-correlated workload for the multiple-client simulation.

The cache size of HP 9000/877 server is only 10-30MB, which is very small by current standard. The *Cello92* trace and the *HTTPD* trace show high temporal locality, and a small client cache may achieve a high hit ratio. In our simulation, the cache size of each client is 16MB for both traces for PVFS, providing a cache hit ratio of more than 70 percent.

The *DB2* trace shows very low temporal locality, because the *reuse distances* [27] of most blocks are less than 150K. We assume the cache size of each client is 128MB in PVFS, providing a cache hit ratio of about 45 percent.

To compare the average response times of PVFS and iPVFS under different configurations, we vary the total number of storage servers to be 6 or 12 for the HP Cello92 and HTTPD traces, and to be 40 or 50 for the *DB2* traces. For iPVFS, each I/O node is supported by two targets. The request size is limited to 64KB to simulate small I/O requests.

In Section 5.1 we explained that a lower cache hit rate is predicted for I/O nodes in iPVFS when the memory size of each node is the same. In our simulation, the aggregate cache sizes of PVFS and iPVFS are the same, but the organization of the caches is different. With Cello92 and HTTPD traces, in PVFS, the cache size of each I/O node is 16MB. In iPVFS, since each I/O node is supported by two targets, we increase the cache size of each I/O node to 24MB, and decrease the cache size of the target nodes to 12MB. With the DB2 trace, in PVFS, the cache size of each I/O node is 128MB, and in iPVFS, the cache sizes of each I/O node and target node are 256MB and 64MB, respectively.

5.2.2 Simulation Results. First we get the cache hit ratios using the simulator under the Cello92, HTTPD, and DB2 traces. The results are given in Tables 2, 3, and 4. Although the hit ratios of I/O nodes in iPVFS is lower than in PVFS, the cumulative hit ratios provided by both the I/O nodes and targets in iPVFS are higher.

The response times of iPVFS and PVFS under the Cello92, HTTPD, and DB2 traces are presented in Figures 7, 8, and 9. With the small number of storage servers, both the PVFS the and iPVFS cannot sustain larger number of requests because of the bottleneck from disks. The iPVFS has better performance (Figure 7(a), Figure 8(a), and Figure 9(a)) because the service rate of the I/O nodes are not the major bottleneck. With the large number of storage servers, systems could handle more requests. The iPVFS performs well when the number of requests is relatively small, but the PVFS is better than iPVFS with really large number of requests, because the service rate of the I/O nodes is truly a bottleneck in such cases (Figure 7(b), Figure 8(b), and Figure 9(b)).

Table 2: Hit ratios of PVFS and iPVFS under the Cello92 trace

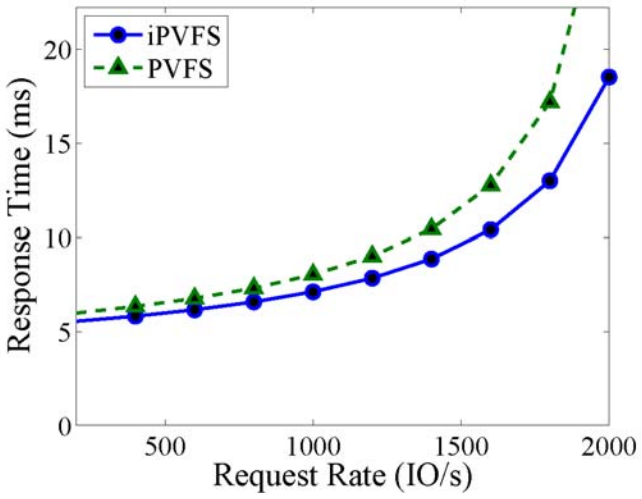
Storage Servers	Hit Ratio of PVFS	Hit Ratio of iPVFS		
		I/O Node	Target	Cumulative
6	0.782	0.714	0.42	0.834
12	0.835	0.782	0.512	0.894

Table 3: Hit ratios of PVFS and iPVFS under the HTTPD trace

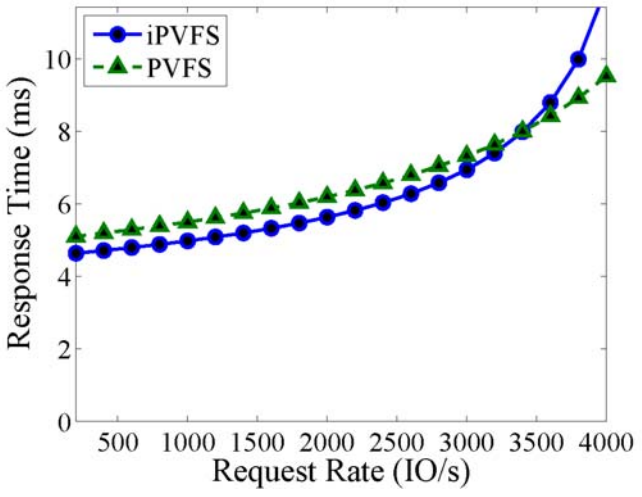
Storage Servers	Hit Ratio of PVFS	Hit Ratio of iPVFS		
		I/O Node	Target	Cumulative
6	0.731	0.664	0.32	0.772
12	0.777	0.77	0.372	0.856

Table 4: Hit ratios of PVFS and iPVFS under the DB2 trace

Storage Servers	Hit Ratio of PVFS	Hit Ratio of iPVFS		
		I/O Node	Target	Cumulative
40	0.42	0.22	0.38	0.51
50	0.48	0.39	0.2	0.52

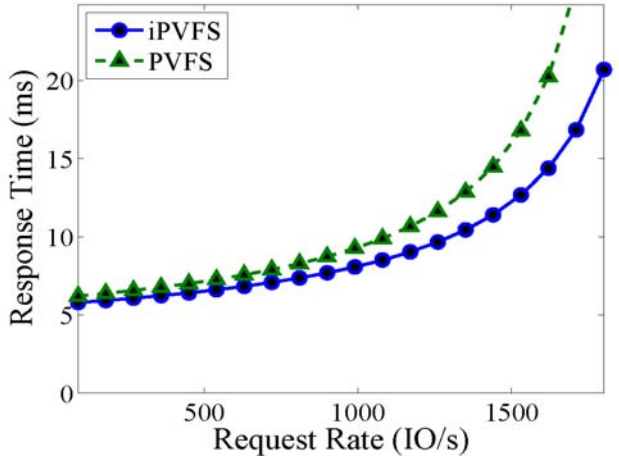


(a) 6 storage servers

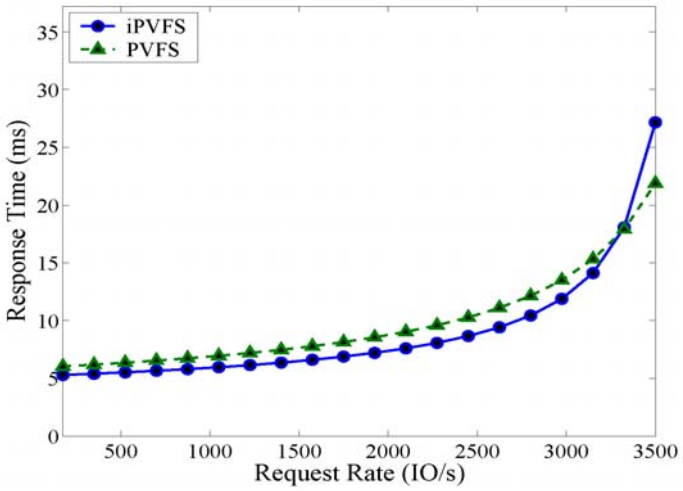


(b) 12 storage servers

Figure 7: I/O response times for various iPVFS configurations under the Cello92 trace

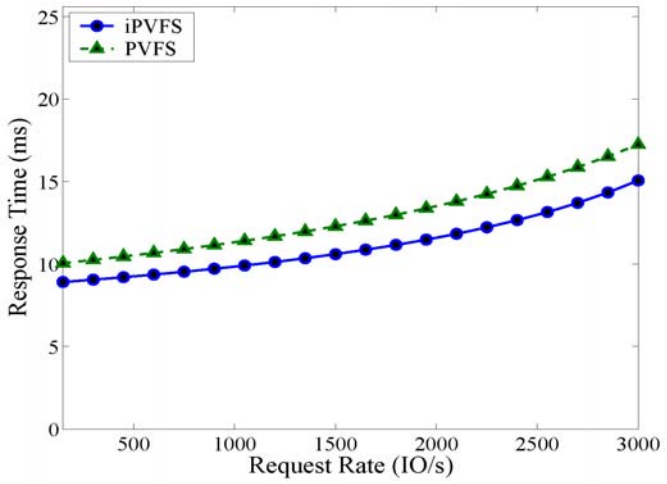


(a) 6 storage servers

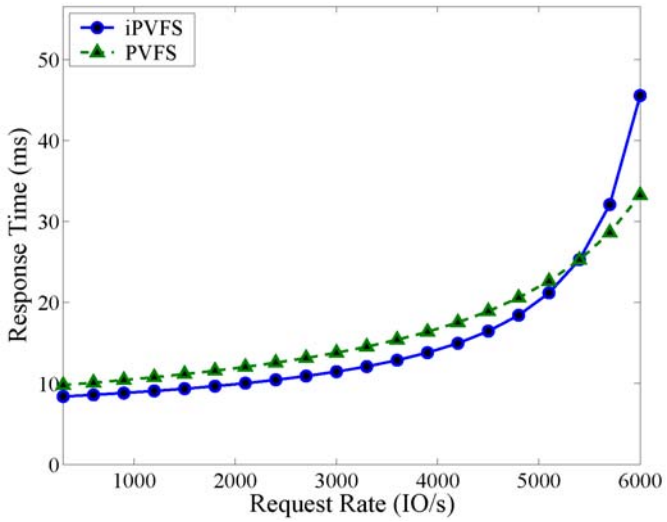


(b) 12 storage servers

Figure 8: I/O response times for various iPVFS configurations under the HTTPD trace.



40 storage servers



50 storage servers

Figure 9: I/O response times for various iPVFS configurations under the DB2 trace

From above simulations we found that when the number of storage nodes is fixed in PVFS, performance is improved by adding more memory to each node. However, in our solution, if iPVFS is deployed, performance is improved by only increasing the memory of the I/O nodes, which are often small parts of the total storage servers. Furthermore, memory size of target nodes can be reduced to decrease the entire cost, while maintaining performance improvement.

6 Related Work

Recent studies have shown how to improve the I/O performance of PVFS. Kernel level client and global caching are implemented in [24] to improve the I/O performance of concurrently executing processes in PVFS. Apon et al. [2] analyzed the role of sensitivity of the I/O nodes and compute nodes and concluded that the overall I/O performance is degraded if a node serves both as an I/O client and as a data server. To reduce disk arm seek time, several scheduling schemes [14, 18] are introduced in I/O nodes to re-order requests according to their desired locations in the space of logical block addresses. CEFT-PVFS [28] increases the availability of PVFS by adopting a RAID-10 architecture. It delivers a considerably high throughput by carefully designing duplication protocols and utilizing mirror data in read operations. In [10] software and hardware RAIDs are used in PVFS I/O nodes to achieve higher aggregate I/O bandwidths. To eliminate communication bottlenecks of networks, Wu et al. [26] use the RDMA features of high-performance interconnects, InfiniBand, to improve the performance of PVFS. A better interface and related implementation is presented in [6] to optimize the performance of non-contiguous I/O accesses. Our work in parallel file systems is different from previous studies because *iPVFS* builds a two-level I/O architecture using iSCSI and iRAID.

Researchers have used mathematical models to analyze the performance of I/O systems. Feng et al. [7] built a queuing model to estimate the response time of CEFT-PVFS. Using approximate analysis, [13] provides a simple expression for a maximum delay of asynchronous disk interleaving and then verifies it by simulation using trace data. Our work uses a two-level queuing model to evaluate the performance of iPVFS.

7 Conclusions

In this paper, we present a parallel file system (iPVFS), based on PVFS and iRAID, for cluster computing environments, and develop a queuing model to measure and compare the system response times of both iPVFS and PVFS with the same number of nodes for various workloads. Our simulation results indicate that iPVFS improves performance under most cases.

In our design, all storage nodes are divided into two groups: one group with more powerful servers acts as the I/O nodes, while the other group with relatively lower performance servers acts as the target nodes to provide a cost effective solution. In a cluster environment, iPVFS can be deployed to improve I/O performance, as long as we provide high performance servers for I/O nodes.

Acknowledgments

This work was supported in part by the US National Science Foundation under grants SCI0453438 and CNS-0617528. The authors would like to thank the associate editor and the anonymous referees for their insightful comments towards the improvement of this work. They are also grateful to Stephen Scott, Zhiyong Xu, and Yung-chin Fang for discussion on the initial ideas and Martha Kosa for proofreading the manuscript and providing many constructive suggestions. A short and preliminary version of this paper was presented at the 30th Annual IEEE Conference on Local Computer Networks (LCN2005) [17], Sydney, Australia, November 15, 2005.

References

[1] S. Aiken, D. Grunwald, A. Pleszkun, and J. Willeke, "Performance Analysis of the iSCSI Protocol," 20th IEEE Conference on Mass Storage Systems and Technologies, pp. 123-134, 2003.

[2] A. W. Apon, P. D. Wolinski, and G. M. Amerson, "Sensitivity of Cluster File System Accesses to I/O Server Selection," ACM/IEEE International Symposium on Cluster Computing and the Grid, pp. 183-192, 2002.

[3] R. B. Bunt, D. L. Willick, and D. L. Eager, "Disk Cache Replacement Policies for Network File Servers," *Proc. of the IEEE International Conference on Distributed Computing Systems-ICDCS '93*, pp. 2-11, June 1993.

[4] 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 Conference*, Atlanta, GA, pp. 317-327, October 2000.

[5] P. M. Chen and D. A. Patterson, "Maximizing Performance in a Striped Disk Array," *Proc. 17th Annual International Symposium on Computer Architecture*, pp. 322-331, May 1990.

[6] A. Ching, A. Choudhary, W. Liao, R. Ross, and W. Gropp, "Noncontiguous I/O through PVFS," *Proc. 2002 IEEE International Conference on Cluster Computing*, pp. 405-414, September 2002.

[7] D. Feng, H. Jiang, and Y. Zhu, "I/O Response Time in a Fault-Tolerant Parallel Virtual File System," Technical Report, Department of Computer Science and Engineering, University of Nebraska-Lincoln, July 2003.

[8] K. Froese and R. B. Bunt, "The Effect of Client Caching on File Server Workloads," *Proc. 29th Hawaii International Conference of System Sciences*, pp. 150-159, January 1996.

[9] X. He, P. Beedanagari, and D. Zhou, "Performance Evaluation of Distributed iSCSI RAID," 2003 International Workshop on Storage Network Architecture and Parallel I/Os (SNAPI'03), September 2003.

[10] J. Hsieh, C. Stanton, and R. Ali, "Performance Evaluation of Software RAID vs. Hardware RAID for Parallel Virtual File System," 9th International Conference on Parallel and Distributed Systems, December 2002.

[11] Y. Hu, T. Nightingale, and Q. Yang, "RAPID-Cache–A Reliable and Inexpensive Write Cache for High Performance Storage Systems," *IEEE Transactions on Parallel and Distributed Systems*, 13(2):290-307, 2002.

[12] E. D. Katz, M. Butler, and R. McGrath, "A Scalable HTTP Server: The NCSA Prototype," *Computer Networks and ISDN Systems*, 27(2):155-164, Nov 1994.

[13] M. Kim and A. N. Tantawi, "Asynchronous Disk Interleaving: Approximating Access Delays," *IEEE Trans. on Computer*, 40(7):801-810, 1991.

[14] W. B. Ligon III and R. B. Ross, "Server-Side Scheduling in Cluster Parallel I/O Systems," *Parallel I/O for Cluster Computing*, ISBN: 1903996503,ISTE Publishing Company, November 2003.

[15] Y. Lu and D. Du, "Performance Study of iSCSI-Based Storage Systems," *IEEE Communications*, 41(8):76-82, 2003.

[16] L. Ou, X. He, M. J. Kosa, and S. L. Scott, "A Unified Multiple-Level Cache for High Performance Storage Systems," *Proc. of the 13th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 143-150, September 2005.

[17] L. Ou, X. He, S. Scott, Z. Xu, and Y. Fang, "Design and Evaluation of a High Performance Parallel File System over iSCSI," *Proceedings of the 30th Annual IEEE Conference on Local Computer Networks*, pp. 100-107, November 2005.

[18] R. B. Ross, *Reactive Scheduling for Parallel I/O Systems*, PhD Dissertation, Clemson University, December 2000.

[19] C. Ruemmler and J. Wilkes, "Unix Disk Access Patterns," *Proc. of Winter 1993 USENIX Conference*, San Diego, CA, pp. 405-420, January 1993.

[20] M. Seager, "Linux Clusters for Extremely Large Scientific Simulation," IEEE International Conference on Cluster Computing, 2003.

[21] M. Stonebraker and G. A. Schloss, "Distributed RAID - A New Multiple Copy Algorithm," *Proc. Sixth International Conference on Data Engineering*, pp. 430-437, February 1990.

[22] R. Thakur, W. Gropp, and B. Toonen, "Optimizing the Synchronization Operations in Message Passing Interface One-Sided Communication," *International Journal of High Performance Computing Applications*, 19(2):119-128, 2005.

[23] M. Uysal, A. Acharya, and J. Saltz, "Requirements of I/O Systems for Parallel Machines: An Application-Driven Study," Technical Report CS-TR-3802, Dept. of Computer Science, University of Maryland, May 1997.

[24] M. Vilayannur, A. Sivasubramaniam, M. Kandemir, R. Thakur, and R. Ross, Discretionary Caching for I/O on Clusters," ACM/IEEE International Symposium on Cluster Computing and the Grid, May 2002.

[25] T. Wong and J. Wilkes, "My Cache or Yours? Making Storage more Exclusive," *Proc. USENIX Annual Technical Conference*, pp. 161-175, 2002.

[26] J. Wu, P. Wyckoff, and D. K. Panda, "PVFS over InfiniBand: Design and Performance Evaluation, International Conference on Parallel Processing, October 2003.

[27] Y. Zhou, Z. Chen, and K. Li, "Second-Level Buffer Cache Management," *IEEE Transactions on Parallel Distributed Systems*, 15(6):505-519, June 2004.

[28] Y. Zhu, H. Jiang, X. Qin, D. Feng, and D. Swanson, "Scheduling for Improved Write Performance in a Cost-Effective, Fault-Tolerant Parallel Virtual File System (CEFT-PVFS)," *Proc. of the Cluster World Conference*, pp. 730-735, June 2003.

LI OU received the Ph.D Degree in Computer Engineering from The Tennessee Technological University in December 2006, and the BS degree in information technology, and MS Degree in Computer Science from the University of Electronics Science & Technology of China, in 1997 and 2003, respectively. He is currently a researcher at DELL. His research interests include computer architecture, storage and I/O systems, and high performance cluster computing.

XUBIN HE received the PhD Degree in Electrical Engineering from the University of Rhode Island, USA, in 2002 and both the BS and MS degrees in Computer Science from the Huazhong University of Science and Technology, China, in 1995 and 1997, respectively. He is an Associate Professor of Electrical and Computer Engineering at the Tennessee Technological University. His research interests include computer architecture, storage systems, computer security, and performance evaluation. He received the Ralph E. Powe Junior Faculty Enhancement Award in 2004 and the TTU Chapter Sigma Xi Research Award in 2005. He is a member of the IEEE Computer Society.