Design and Evaluation of a High Performance Parallel File System

Li Ou, Xubin (Ben) He
Tennessee Technological University
Cookeville, TN 38505
{lou21,hexb}@tntech.edu

Stephen L. Scott
Oak Ridge National Laboratory
Oak Ridge, TN 37831
scottsl@ornl.gov

Zhiyong Xu
Suffolk University
Boston, MA 02114
zxu@mcs.suffolk.edu

Yung-chin Fang
Dell Inc.
Austin, TX 78728
Yung-Chin_Fang@Dell.com

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 environment by dividing a set of I/O servers into two groups, one group with higher performance servers as I/O nodes, while another group with relatively lower performance machines 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 same number of total nodes, iPVFS outperforms PVFS for both small requests and large requests under different workloads.

1 Introduction

Cluster computing [15] has become one of the most popular platforms for high-performance computing today, because of its high performance-cost ratio. Similar to the traditional parallel computing systems, the I/O sub-system is a bottleneck to improve overall performance. The most efficient way to alleviate the I/O bottleneck is to deploy a parallel file system, which can utilize the aggregate bandwidth and capability of exiting 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) [3], developed at the 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 caching is implemented and deployed in [14] to reduce response time. In [9, 12], several scheduling schemes are introduced in I/O nodes to re-order the service of requests to reduce the disk seeking time. A better interface and related implementation is presented in [4] to optimize the non-contiguous I/O access performance. CEFT-PVFS [17] increases availability of PVFS, while still being able to deliver a considerably high throughput.

One way of improving aggregate I/O performance of cluster computing platforms is to improve I/O performance of each storage node. In PVFS, file data is stored in local disks of I/O nodes. If performance of local disks is improved, the overall performance of PVFS is also improved. In [8], 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, which is based on PVFS and iSCSI, for cluster computing platform. We have designed the iPVFS and developed a model to simulate it. We compare the I/O response time of iPVFS with original PVFS under different configurations, and the results show dramatic performance gain of iPVFS over PVFS.

The rest of this paper is organized as follows. Background 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 is a popular parallel file system for Linux cluster computing. It provides high-speed access to file data for

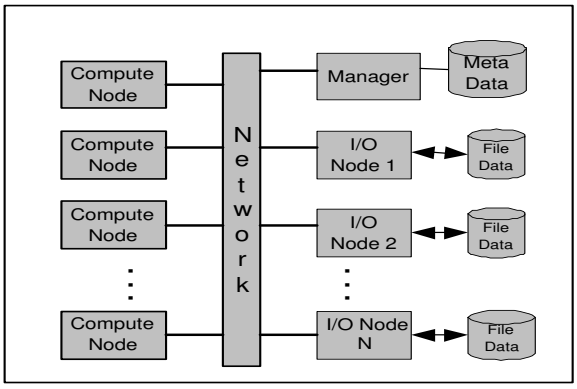


Figure 1. PVFS system diagram. Number of I/O nodes is N, and total number of storage nodes is N.

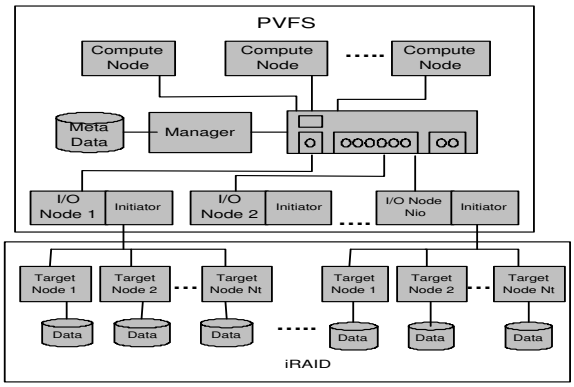


Figure 2. iPVFS Architecture. Number of I/O nodes is N_{io} , number of target nodes of each iRAID group is N_t, and total number of storage nodes is still $N = N_{io}(1 + N_t)$.

parallel applications. Fig. 1 [3] shows typical PVFS architecture and main components. There are three types of nodes in PVFS. The metadata node maintains information on files and directories stored in a PVFS file system. I/O nodes store PVFS file data, by creating files on local file systems, such as an existing ext2fs partition. Clients, or compute nodes, are nodes where application tasks run, contact the metadata server when they want to manipulate files and contact I/O servers in order to store and retrieve PVFS file data.

2.2 iSCSI and iRAID

iSCSI [1, 11], also known as Internet SCSI or SCSI over IP, is a newly emerging protocol with the goal of implementing the 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 may provide greater flexibility because it provides a block level data interface which is independent to any file systems. Theoretically iSCSI storage is treated by operating systems as a local block level device over any TCP/IP network infrastructure.

iRAID[7] is introduced to improve the performance and reliability of iSCSI storage systems by organizing the iSCSI storage targets similar to RAID using striping and rotated parity techniques. In iRAID, each iSCSI storage target is a basic storage unit in the array, and it serves as a storage node. All the nodes in the array are connected to each other through a high-speed switch to form a local area network. iRAID provides a direct and immediate solution to boost iSCSI performance and improve reliability. Parallelism in iRAID leads to performance gain while using the RAID parity technique improves the reliability.

3 Architecture of iPVFS

In cluster environment, nodes can be treated as block-level storage providers and be grouped together to form distributed RAID [13]. Combining iRAID and PVFS improves parallel I/O performance since PVFS concentrates in file system level, while iRAID focuses on block level.

Above observations motivate us to propose iPVFS to improve aggregate bandwidth by utilizing iRAID as the local storage system of I/O servers of PVFS. In iPVFS, each I/O server also acts as an iSCSI Initiator, which is supported by several target nodes to form iRAID storage as shown in Fig. 2. With iRAID, the I/O servers stripe PVFS data through multiple target nodes, thus local I/O performance is improved and the higher overall aggregate bandwidth is obtained. In iPVFS, all nodes, except compute nodes and metadata node, are divided in two groups. The nodes in first group act as I/O servers of PVFS, which provide file level services for cluster computing platforms. Others are treated as target nodes of iRAID system, which provide the block level services for PVFS.

4 A queuing model for iPVFS

In a cluster environment, if the number of servers is given, the utilization of servers of PVFS and iPVFS is different. In PVFS, except metadata server, all servers are designated as I/O nodes to improve I/O bandwidth, but in iPVFS, as we mentioned in Section 3, some servers act as target nodes to speed up performance of I/O nodes, and at the same time, compared with a pure PVFS system, the

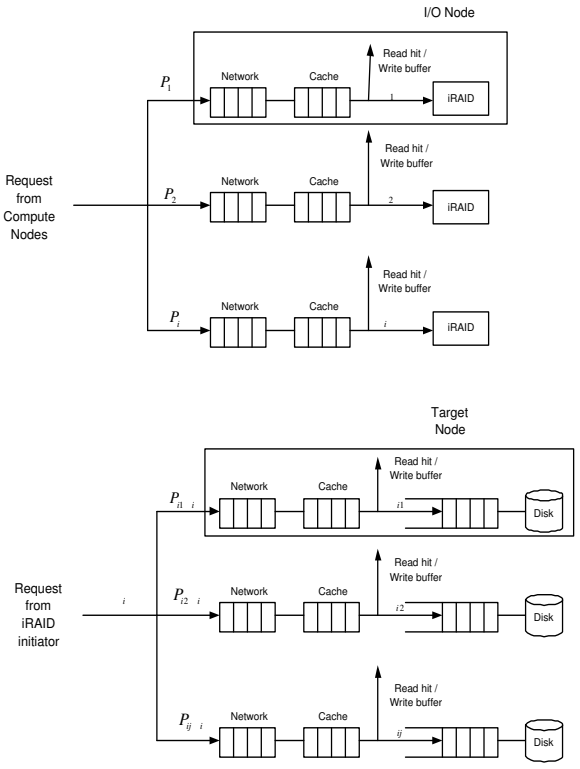


Figure 3. Queuing Model for iPVFS.

number of I/O nodes is reduced. With given number of machines, which design is better?

To answer this question, we develop a queuing model to compare performance of PVFS and iPVFS in terms of average I/O response time, as shown in Fig. 3. Our model differs from the queuing model of [6] in that we have two level network queues: one for I/O nodes and the other for target nodes. We also use some assumptions made by [6]. The number of I/O requests follows a Poisson process; with a mean arrival rate of λ , and the load on an I/O node in PVFS storage server group and on a target node in iRAID server group are all balanced.

We assume that there are N storage nodes in a cluster. The number of I/O nodes and target nodes serving for each I/O node are N_{io} and N_t, respectively. In PVFS, $N = N_{io}$, while in iPVFS, $N = N_{io}(1 + N_t)$. For each I/O node, the arrival rate is $P_i\lambda$, where P_i is probability that the request is directed to I/O node i. When the request data size L_{ip} is less or equal to 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 directed to every I/O node 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 $P_i = min(\frac{\lceil L_{ip}/B \rceil}{N_{io}}, 1)$. If not satisfied by I/O node cache, the request is directed to 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} is cache hit percentage for an I/O node.

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

Request delays are mainly caused by network transportation and memory cache services, no matter cache hit or miss. We assume that the network service time and the cache service time are exponentially distributed [6] with the average time 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 network and memory cache, respectively. Therefore, request residence time in the network and cache is modeled using M/M/1 queuing model [6], and the average residence time of request in network and cache of I/O node is 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 memory cache, it has to be handled by iRAID system, in the probability of $1 - h_{ioc}$. Thus, the average response time of iPVFS system is expressed as:

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

Where W_{iRAID} is request residence time in iRAID system, and T_{node} is processing time of each node.

In iRAID, additional overheads are introduced when requests travel through network between I/O nodes and target nodes, and are serviced by memory caches of target nodes. We also assume that the network service time and the cache service time are exponentially distributed with the average time T_{tnet} and T_{tc} , respectively.

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

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

Request residence time in the network and cache are also modeled using M/M/1 queuing model. So the average residence time of request in network and cache of iRAID is represented as:

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

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

If a cache miss occurs, the request is directed to a hard disk, in the probability of $1 - h_{tc}$. According to [10], the real disk service times is generally distributed, so we adopt the M/G/1 model to analyze response time of disk in target node.

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

Where $E(T_{disk_{io}})$ is average disk access time for a request. Thus, the average response time of iRAID system is expressed as:

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

5 I/O response time analysis

Based on above queuing model, we simulate and compare response times of iPVFS and pure PVFS under various workload and application environment. Some parameters are as follows: the available network bandwidth is about $75MB/s$ and the memory access rate is about $500MB/s$. $64KB$ data striping size is chosen for both PVFS and iRAID. The disks are model ST318452LW, with $51MB/s$ data transfer rate, $3.8ms$ average seek time, $2ms$ average latency, and 18497 cylinders. The number of storage nodes in our simulation is 18 unless otherwise specified.

In our simulation, we measure the I/O response time for both fixed cache hit rate and dynamic cache hit rate. iPVFS outperforms PVFS in both situations.

5.1 I/O response time for fixed cache hit rate

We assume that the cache hit rates of both I/O nodes and target nodes are fixed, and that read request percentage is 0.6 and write request percentage is 0.4, which is typical for office/engineering applications.

For pure PVFS, all 18 servers act as I/O nodes ($N = N_{io} = 18$), while for iPVFS, we may have various choices. In this section, we use 6 nodes as PVFS I/O nodes, and for each I/O node, 2 target nodes form an iRAID group ($N = 18$, $N_{io} = 6$, $N_t = 2$). Other choices are discussed in Section 5.1.3.

5.1.1 Small I/O request

Small requests are equal to or less than striping block size $64KB$, so that each request is satisfied by a single node. We assume that the request data size is equal to $64KB$. For iPVFS, arrival rate to a I/O node and a target node are λ/N_{io} and $\lambda/(N_{io}N_t)$, respectively; and for pure PVFS, arrival rate to a I/O node is λ/N.

Fig. 4 compares performance between iPVFS and pure PVFS. It shows that average I/O response time increases steadily with the increase of request rate, and how different cache hit rates influence the response time. It is obvious that response time of iPVFS is much smaller than PVFS, especially when the system is heavy loaded. At the point where pure PVFS is saturate by the large request, the iPVFS still provides acceptable services. We believe it is cache that makes the difference, since compared to network and cache, disk access time accounts most for the total response time. In pure PVFS, when cache misses occur in I/O nodes, requests have to be directed to disks, which needs much more time. In iPVFS, in case of cache misses in I/O nodes, the requests are first be directed to target nodes, in which they may be satisfied by target nodes caches.

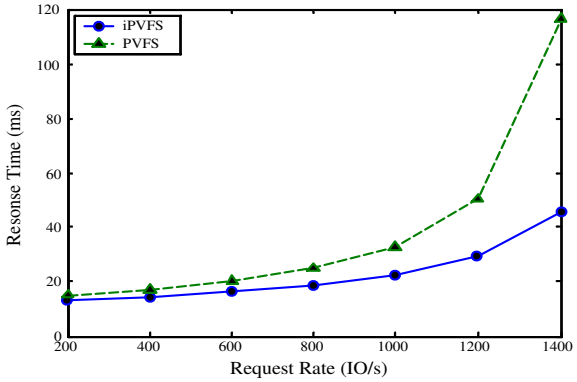
5.1.2 Large I/O request

If the request data size is much larger than striping block size $64KB$, each request is striped over several nodes. The extreme case is that the request is large enough that it is striped over all nodes. For iPVFS, the arrival rate to a I/O node and to a target node are all λ; for pure PVFS, it is λ too. Fig. 5 compares performance of iPVFS and pure PVFS in various cache hit rates for $640KB$ request size.

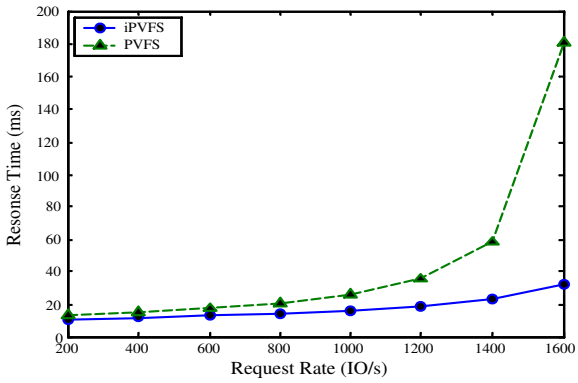
Comparing Fig. 4 and 5, we find that, the gain of iPVFS in large I/O request case is much smaller than that in small I/O request case. In large I/O request situation, each node receives much more requests, which saturate cache quickly, thus, most requests have to be directed to disks. So the system is more likely to be saturated, for both iPVFS and pure PVFS. Because of another level cache of target node, iPVFS obtains some gains, but the heavy loads make the gain trivial, unless the cache hit rate improves significantly, which is showed in Fig. 5(c).

5.1.3 Performance of various iPVFS configurations

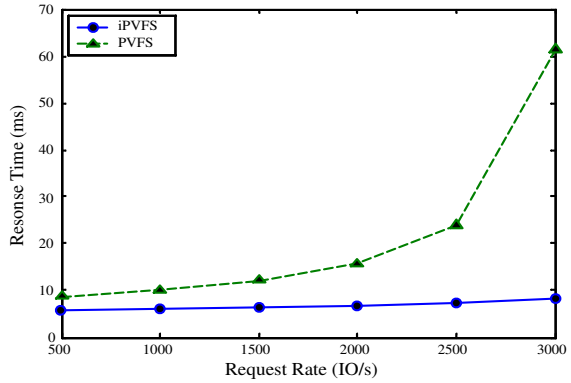
When the number of storage node is fixed for iPVFS, how should we organize our system to achieve maximum I/O performance? We may have various configurations because selection of N_{io} and N_t is not unique, as long as equation $N = N_{io}(1 + N_t)$ is satisfied. For $N = 18$, we may have three different configurations depending on different combinations of N_{io} and N_t. They are: 6 I/O nodes, 12 target nodes (2 target nodes for each iRAID group); 3 I/O nodes,



(a) Cache hit ratio is 0.2

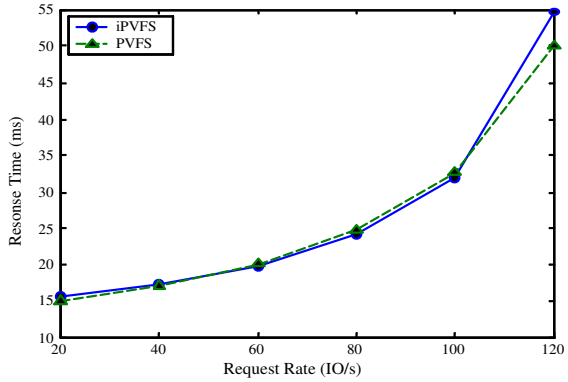


(b) Cache hit ratio is 0.3

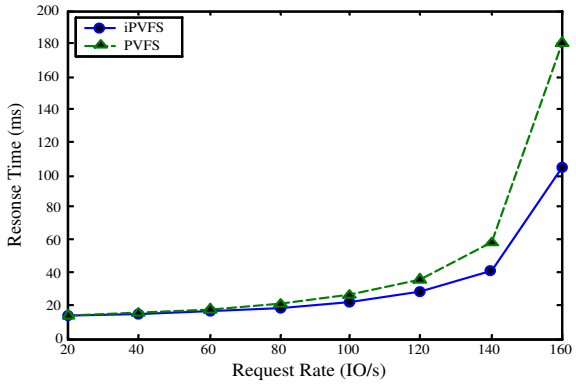


(c) Cache hit ratio is 0.8

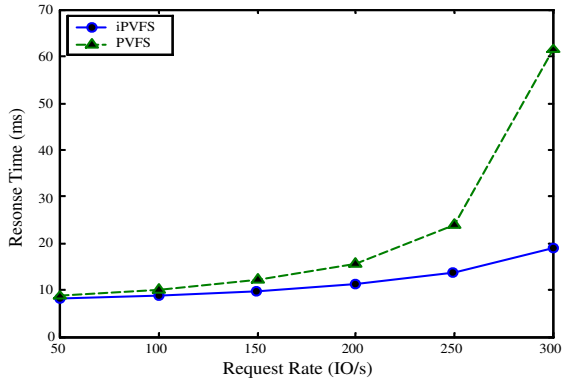
Figure 4. I/O response time with various cache hit rate for small request.



(a) Cache hit ratio is 0.2

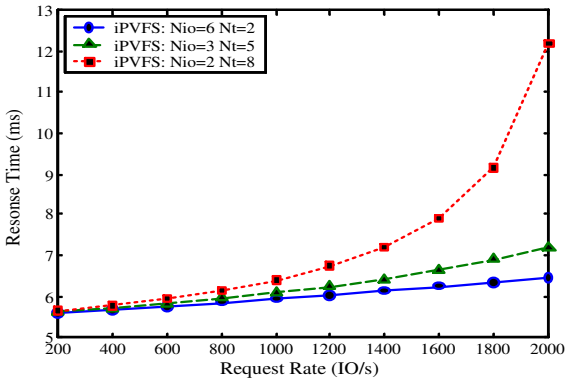


(b) Cache hit ratio is 0.3

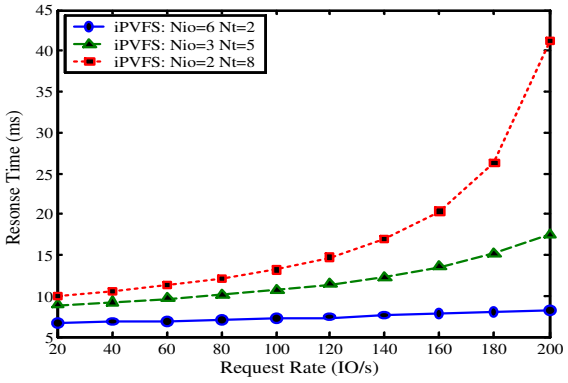


(c) Cache hit ratio is 0.8

Figure 5. I/O response time with various cache hit rate for large request.



(a) Small request



(b) Large request

Figure 6. I/O response time for various iPVFS configurations under fixed cache hit rate.

15 target nodes (5 target nodes for each iRAID group); 2 I/O nodes, 16 target nodes (8 target nodes for each iRAID group). Results of these three configurations for both small requests and large requests are shown in Fig. 6. It is obvious that the larger number of I/O nodes provides better performance, since request rate of a I/O node decreases with increasing number of I/O nodes.

5.2 I/O Response time for dynamic cache hit rate

In real application environment, cache hit rate of storage nodes is a variable which is influenced by many factors such as server memory size, request load, and so on. To obtain more accurate performance comparisons, it is necessary to predict cache hit rate dynamically for iPVFS and PVFS.

5.2.1 Cache hit rate prediction

We use the model developed by Dan and Towsley [5] to dynamically predict LRU cache hit rate. In our simulation, the number of partitions K is set to 2, which means files have two partitions with different access probabilities. Cache hit rate is predicted as a function of cache size and the request file size, if file size S is much larger than total memory cache size of all nodes, requests may cause cache misses.

As long as same storage servers are provided for both pure PVFS and iPVFS, the number of I/O nodes in iPVFS is always less than pure PVFS since some nodes are used as targets, so the lower cache hit rate can be predicted for I/O nodes in iPVFS when same memory size of each node is provided for both pure PVFS and iPVFS. An economical way to improve iPVFS performance is to add more memory to I/O nodes to compensate its relatively less number of I/O nodes since DRAM is very cheap nowadays.

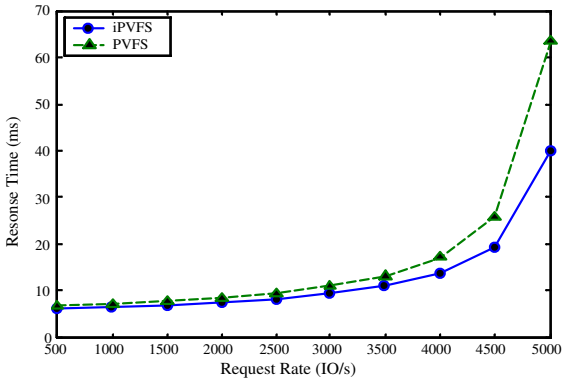
In our simulation, all 18 nodes are I/O nodes for PVFS, while for iPVFS, 6 nodes are I/O nodes, and 2 nodes act as target nodes for each iRAID group, which means 12 nodes are target nodes (other choices are discussed shortly in section 5.2.2). Request size is limited to $64KB$ to simulate small I/O request.

All nodes in PVFS are equipped with $256MB$ memory caches, but in iPVFS, cache sizes of I/O nodes and target nodes are different. First, we increase memory size of I/O nodes in iPVFS to $656MB$, but reduce memory size of target nodes to $156M$. To ensure cache misses for I/O nodes and target nodes, $6144MB$ large file is used. The response time is presented in Fig. 7(a). We find that in this situation, iPVFS has better performance, but the difference is relatively trivial. Then, we increase the memory size of target node to $196MB$ but keep other parameter unchanged, and the result is given in Fig. 7(b). We find that iPVFS is much better than pure PVFS. The same performance improvements is found in Fig. 7(c), when memory size of I/O nodes is increased to $724MB$.

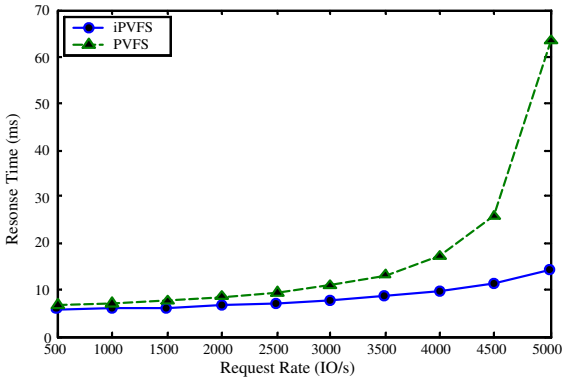
From above simulation, it is reasonable to conclude that iPVFS improves I/O system performance compared to pure PVFS. When the number of storage nodes is fixed in PVFS, performance is improved by adding more memory to each node, but in our solution, if iPVFS is deployed, performance is improved by only increasing memory of I/O nodes, which are often small parts of total storage servers. Furthermore, memory size of target nodes can be reduced to decrease the whole cost, while maintain performance improvement.

5.2.2 Performance of various iPVFS configurations

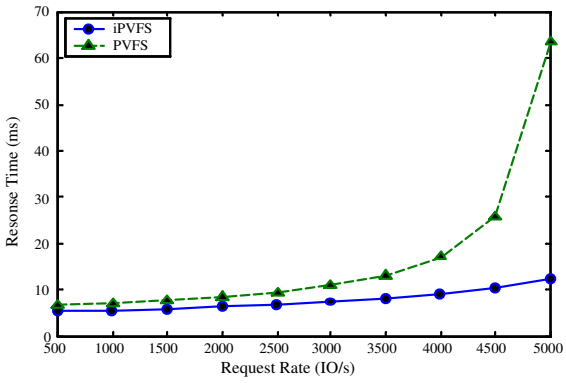
We have discussed in Section 5.1.3 about performance of various iPVFS configurations. Based on fixed cache hit rate simulations, we conclude that the more I/O nodes, the better performance. We also do the same simulation for dy-



(a) $656MB$ memory each I/O node and $156MB$ memory each target node for iPVFS



(b) $656MB$ memory each I/O node and $196MB$ memory each target node for iPVFS



(c) $724MB$ memory each I/O node and $156MB$ memory each target node for iPVFS

Figure 7. I/O response time for various memory configurations of iPVFS with dynamic cache hit rate.

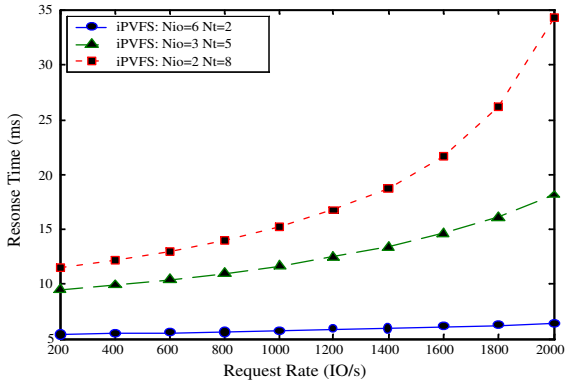


Figure 8. I/O response time for various iPVFS configurations under dynamic cache hit rate.

namic cache hit rate. There are three configurations, as what we have done for fixed cache hit rate: 6 I/O nodes, 12 target nodes (2 target nodes for each iRAID group); 3 I/O nodes, 15 target nodes (5 target nodes for each iRAID group); and 2 I/O nodes, 16 target nodes (8 target nodes for each iRAID group). Memory size of I/O nodes and target nodes in iPVFS are $724MB$ and $156M$, respectively, and $6144MB$ large file is used. The result is showed in Fig. 8. Six I/O nodes configuration is still the winner. The larger N_{io} is preferred, because larger N_{io} not only decrease request rate for each I/O node, but also increases cache hit rate, which is very important in dynamic hit rate environment.

6 Related Work

Several recent works have studied how to improve parallel I/O performance of PVFS. The kernel level client and global caching are implemented in [14] 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 will be degraded if a node serves both as an I/O client and as a data server. To reduce disk arm seeking time, several scheduling schemes [9, 12] are introduced in I/O nodes to re-order the service of requests according to their desired locations in the space of Logical Block Address. CEFT-PVFS [17] increases 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 [8], software and hardware RAIDs are used in PVFS I/O nodes to achieve higher aggregate I/O bandwidth. To eliminate communication bottleneck of network, Wu *et al.* [16, 15] use the RDMA features of high-performance interconnects, like Myrinet, Quadrics, and InfiniBand, to improve the performance of PVFS. A better interface and related imple-

mentation is presented in [4] to optimize the non-contiguous I/O access performance. Our work in parallel file systems is related but different from previous studies because *iPVFS* builds a two-level I/O architecture using iSCSI and iRAID.

Researchers have used mathematics models to analyze performance of I/O system. Feng *et al.* [6] built a queuing model to estimate the response time of CEFT-PVFS. In [10], using approximate analysis, a simple expression for a maximum delay of asynchronous disk interleaving is obtained and then verified by simulation using trace data. Our work uses a two-level queuing model to evaluate performance of iPVFS.

7 Conclusions

In this paper, we propose a parallel file system (iPVFS), based on PVFS and iRAID, for cluster computing environment, and develop a queuing model to measure and compare system response times of both iPVFS and PVFS with same numbers of nodes. Our simulation indicates that iPVFS improves the performance under different workloads.

In a cluster environment, iPVFS, instead of pure PVFS, can be deployed to improve I/O performance, as long as we provide high performance servers for I/O nodes, which are at most half number of total storage nodes. Therefore, in our design, all storage nodes are divided into two groups: one group is equipped with more powerful servers acting as I/O nodes, another group with relatively lower performance servers acting as target nodes. This cost effective solution provides higher I/O performance than pure PVFS while keep low cost. Our design suits for large scientific computation environment, in which multiple clients manipulate very large data simultaneously in a heterogeneous environment.

Acknowledgments

This work was supported in part by Research Office under Faculty Research Grants and Center for Manufacturing Research at Tennessee Technological University, ORAU under the Ralph E. Powe Junior Faculty Enhancement Award, and the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, Office of Science, U. S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC. The authors would like to thank the anonymous referees for their insightful and constructive comments. They would also like to thank two REU students, April Childers and Karthik Sankar for helping collecting some experimental data. They are supported by the US National Science Foundation under a REU grant SCI-0453438.

References

[1] S. Aiken, D. Grunwald, A. Pleszkun, and J. Willeke. Performance analysis of the iscsi protocol. In *20th IEEE Conference on Mass storage Systems and Technologies*, 2003.

[2] A. W. Apon, P. D. Wolinski, and G. M. Amerson. Sensitivity of cluster file system sccesses to I/O server selection. In *ACM/IEEE International Symposium on Cluster Computing and the Grid*, pages 183–192, 2002.

[3] P. H. Carns, W. B. L. III, R. B. Ross, and R. Thakur. PVFS: A parallel file system for linux clusters. In *Proc. 4th Annual Linux Showcase and Conference*, pages 317–327, Atlanta, GA, October 2000.

[4] A. Ching, A. Choudhary, W. Liao, R. Ross, and W. Gropp. Noncontiguous I/O through PVFS. In *Proc. 2002 IEEE International Conference on Cluster Computing*, Sept 2002.

[5] A. Dan and D. Towsley. An approximate analysis of the LRU and FIFO buffer replacement schemes. In *ACM SIGMETRICS*, pages 143–152, May 1990.

[6] 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.

[7] X. He, P. Beedanagari, and D. Zhou. Performance evaluation of distributed iSCSI RAID. In *the 2003 International workshop on Storage Network Architecture and Parallel I/Os (SNAPI'03)*, September 2003.

[8] J. Hsieh, C. Stanton, and R. Ali. Performance evaluation of software RAID vs. hardware RAID for parallel virtual file system. In *9th International Conference on Parallel and Distributed Systems*, December 2002.

[9] W. B. L. III and R. B. Ross. Server-side scheduling in cluster parallel I/O systems. *Calculateurs Paralleles Journal Special Issue on Parallel I/O for Cluster Computing*, October 2001.

[10] M. Kim and A. N. Tantawi. Asynchronous disk interleaving: Approximating access delays. *IEEE Trans. on Computer*, 1991.

[11] Y. Lu and D. Du. Performance study of iSCSI-based storage systems. *IEEE Communications*, 41(8), 2003.

[12] R. B. Ross. *Reactive Scheduling for Parallel I/O Systems*. PhD dissertation, Clemson University, Dec 2000.

[13] M. Stonebraker and G. A. Schloss. Distributed RAID - A new multiple copy algorithm. In *Proc. Sixth International Conference on Data Engineering*, pages 430–437, February 1990.

[14] M. Vilayannur, A. Sivasubramaniam, M. Kandemir, R. Thakur, and R. Ross. Discretionary caching for I/O on clusters. In *ACM/IEEE International Symposium on Cluster Computing and the Grid*, May 2002.

[15] J. Wu, P. Wyckoff, and D. K. Panda. PVFS over InfiniBand: Design and performance evaluation. In *International Conference on Parallel Processing*, Oct 2003.

[16] J. Wu, P. Wyckoff, and D. K. Panda. Supporting efficient noncontiguous access in PVFS over InfiniBand. In *Cluster 2003 Conference*, December 2003.

[17] 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). In *Proc. Cluster World Conference*, June 2003.