Scheduling for Improved Write Performance in a Cost-Effective, Fault-Tolerant Parallel Virtual File System (CEFT-PVFS)

Yifeng Zhu[1], Hong Jiang[1], Xiao Qin[1], Dan Feng[2], and David R. Swanson[1]
[1]Department of Computer Science and Engineering
University of Nebraska – Lincoln, Lincoln, Nebraska, USA
[2]Department of Computer Science and Engineering
Huazhong University of Science and Technology, Wuhan, China

Abstract. Without any additional hardware, CEFT-PVFS utilizes the existing disks on each cluster node to provide RAID-10 style parallel I/O service. In CEFT-PVFS, all servers are also computational nodes and can be heavily loaded by different applications running on the cluster, thus potentially degrading the I/O performance. To minimize the degradation, I/O requests can be scheduled on a less loaded server in each mirroring pair. To help define the meaning of "load" in face of multiple resources such as CPU, memory, disk and network, this paper examines the impacts of these resources by measuring aggregate I/O throughput of the simplest CEFT-PVFS configurations, under specific and isolated workload stresses. Based on the heuristic rules found from the experimental results, a scheduling algorithm for dynamic load balancing is developed. In a CEFF-PVFS with 16 data servers, we evaluate this algorithm under different workloads. The results show that the proposed scheduling algorithm significantly improves the overall performance.

Introduction

The high performance-cost ratio of Linux cluster computing has made it as one of the most popular platforms for high-performance computing nowadays. Nevertheless, similarly with traditional massively parallel computers, the improvement of I/O performance remains a challenge. The most cost-effective approach to alleviate the I/O bottleneck is to build a parallel file system that taps into the aggregated bandwidth of existing disks on cluster nodes to deliver a high-performance and scalable storage service. This approach does not need any additional hardware. PVFS [1] is one notable example of such file systems, which stripes the data among the cluster nodes and accesses these nodes in parallel to store or retrieve data. The fatal disadvantage of PVFS is that it does not provide any fault tolerance in its current version. Like disk arrays [2], without redundancy, these parallel storage systems are too unreliable to be useful since the failure of any cluster node will make all storage services unavailable.

A Cost-Effective, Fault-Tolerant Parallel Virtual File System (CEFT-PVFS) [3][4], has been designed and implemented to meet the critical demands on reliability while still being able to deliver a considerably high throughput. While PVFS is a RAID 0 style system and it does only striping in its current implementation, CEFT-

PVFS is a RAID 10 style system, which combines striping with mirroring by first striping among the primary group of storage nodes and then duplicating all the data in the primary group to the backup group to provide fault tolerance. As shown in our previous studies [3], the experiments conducted in a cluster of 128 nodes and theoretical reliability analysis based on Markov chain models show that, in cluster environments, mirroring can improve the reliability by a factor of over 40 (4000%) while sacrificing the peak write performance by 33-58% when both systems are of identical sizes (i.e., counting the 50% mirroring disks in the mirrored system). In [4], we present that the read performance of CEFT-PVFS can be improved by 100% by doubling the parallelism: reading the first half of a file from one storage group and the second half from the other group in parallel.

In this study, we propose a dynamical scheduling algorithm in CEFT-PVFS to improve the write performance. In a cluster, all the storage server nodes in CEFT-PVFS or PVFS, as integral parts of a cluster, usually also serve as computational nodes. Application programs running on these server nodes will compete for the system resources, such as CPU, memory, disk and network bandwidth, with CEFT-PVFS or PVFS server daemons. This resources contention can significantly degrade the overall I/O performance. PVFS cannot avoid this performance degradation since there is only one copy of the desired data, and thus all the I/O requests have to be directed to the destination nodes, even if these nodes are extremely heavily loaded. On the other hand, CEFT-PVFS stores two data copies on different nodes. In this paper, we propose a heuristic scheduling algorithm, based on the availability of various kinds of resources (CPU, network bandwidth, memory and disk) on server nodes, to improve the write performance by scheduling the I/O requests to run on the nodes that can potentially deliver a higher I/O throughput.

The rest of this paper is organized as follows. We discuss the related work first and then give an overview of our CEFT-PVFS. Next the impacts of the system resources' constraints and of the striping group size on the CEFT-PVFS performance are investigated. A scheduling algorithm based on the heuristic rules is presented and evaluated under different system resource constraints (stressed workload). Finally, we conclude the paper with comments on current and future work.

Related Work

Extensive research has been conducted in the area of parallel I/O. Among the many successful parallel I/O systems proposed or implemented is a production-level file system, Parallel Virtual File System (PVFS) [1], developed at the Clemson University and Argonne National Lab. Like RAID, the files are partitioned into stripe units, and the stripes are distributed to disks on cluster nodes in a round robin fashion. Unlike RAID, PVFS provides a file-level, instead of block-level interface, and all the data traffic flows in parallel, without going through a centralized component, which can become a performance bottleneck. The experimental results show that PVFS provides high performance, even for non-contiguous I/O accesses [5], which may cause significant performance degradation in a conventional storage system. On the other hand, PVFS in its current form is only a RAID-0 style storage system without any

fault-tolerance. Any single server node failure will render the entire file system dysfunctional. One of the authors of PVFS [6] further addressed the importance and necessity to incorporate fault-tolerance into PVFS.

CEFT-PVFS extends the work of PVFS by incorporating a fault-tolerant mechanism in the form of server mirroring. Thus, CEFT-PVFS is a RAID-10 style parallel file system, which first stripes the data across a group of storage nodes and then mirrors these data into another group. In CEFT-PVFS, a new naming mechanism, MD5 sum [7] is used for the striped data, which overcomes the inability of PVFS to back up the metadata server, the most critical component in the storage system. Four different mirroring (or duplication) protocols are designed and implemented in software, without adding any new hardware. In addition, the reliability of these protocols is theoretically analyzed based on a Markov chain model.

There are several studies related to PVFS. Ref. [8] implements a kernel level caching to improve I/O performance of concurrently executing processes. Ref. [9] analyzes the role of sensitivity of the data servers and clients, and concludes that when a node serves both as an I/O client and as a data server, the overall I/O performance will be degraded. In [10][11], a scheduling scheme is introduced for the service order of different requests on each server according to their desired locations in the space of LBA in order to reduce disk arm seeking time.

The study in this paper is different from the above studies in that it attempts to improve the write performance of CEFT-PVFS by exploiting the possible disparity in resource availability between two nodes of each mirroring pair and among mirroring pairs within a server group. To the best of our knowledge, this paper is the first to address this issue.

Overview of CEFT-PVFS

In CEFT-PVFS, we choose to use mirroring to improve the reliability. As the storage capacity increases exponentially, the storage cost decreases accordingly; according to Moore's Law, a *terabyte* of IDE storage will cost $100 US within three years. Compared with IDA [12][13] and RAID 5 style [14][15] parallel systems, the mirroring approach adds the smallest operational overhead and its recovery process and concurrency control are relatively simple and thus have higher efficiency. Another benefit from mirroring, which the other redundancy approaches can not achieve, is that the aggregate read performance can be doubled by doubling the parallelism, that is, reading data from two mirroring groups simultaneously.

CEFT-PVFS divides server nodes into two storage groups, one primary group and one backup group. The primary and backup groups have the same number of server nodes, as shown in Figure 1. The division of the primary and backup group is based on files; for different files, the division of the primary group and the backup group can be different. In each group, there is one metadata server, which records the striping information for each file. To access the data in CEFT-PVFS, all clients need to contact the metadata servers first to get the destination data server addresses and the striping information about their desired data. After that all I/O operations will take place between the clients and servers directly in parallel through the network.

Another important responsibility of the metadata server is to choose one node among each mirroring pair to form a primary storage group for each individual I/O. When an I/O request arrives, the metadata server will check whether the required metadata exists. If the metadata exists and the duplication flags indicate that the required data file on some data server nodes has not finished the duplication, then these data server nodes have to be included into the primary group. In the other cases, the metadata server is needed to choose one node among each pair according to the workloads on the data servers. Each data server node periodically collects its CPU, network, memory and disk usage information and sends it to the metadata server along with the heartbeat message. Based on this information, a scheduling algorithm, given in the later sections, will be used to form the primary storage group.

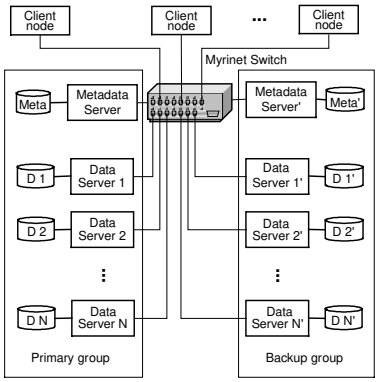

Fig. 1. Logical block diagram of CEFT-PVFS

For write accesses in CEFT-PVFS, we have designed and implemented four duplication protocols to meet different requirements for reliability and write performance. Duplication can be either synchronous or asynchronous, i.e., the completion of write accesses can be signaled after the data has already taken residence on both groups or only on the primary group. At the same time, duplications can be performed either by the client nodes themselves or by the servers in the primary group. The four protocols are created based on different combinations from these two categories. The experimental measurements and theoretical analysis based on Markov chain models indicate that protocols with higher peak write performance are inferior to those with lower peak write performance in terms of reliability, with the latter achieving a higher reliability at the expense of some write bandwidth [3].

Impact of System Recourse Constraints and Striping Group Size on Performance

Among the four duplication protocols, in this paper we only address the asynchronous server duplication protocol in which the servers do the duplication and the I/O

operations are considered completed when the data has taken residence on the primary group. This protocol benefits more from the scheduling than the other three protocols while the skipping will impact the performance of all protocols.

Environment and Benchmark

The performance results presented here are measured on the PrairieFire cluster [16] where CEFT-PVFS has been implemented and installed, at the University of Nebraska-Lincoln. At the time of our experiment, the cluster had 128 computational nodes, each with two AMD Athlon MP 1600 processors, 1 GByte of RAM, a 2 gigabits/s full-duplex Myrinet [17] card, and a 20GB IDE (ATA100) hard drive. Under the same network and system conditions in which CEFT-PVFS operates, the Netperf [18] benchmark reports a TCP bandwidth of 126.51 MBytes/s with 47% CPU utilization. The disk write bandwidth is 32 MBytes/s when writing a large file of 2 GBytes according to the Bonnie benchmark [19].

A micro-benchmark, similar to the one used in [1][20][21][22], was used to measure the overall write performance of this parallel file system. In this benchmark, each client concurrently opens a new common file, then writes disjoint portions of this file, and finally closes it. The response time of the slowest client is considered as the overall response time. Figure 2 shows the pseudocode of this benchmark.

```
for all clients:
   synchronize all clients using MPI barrier;
   t1 = current time;
   open a file;
   synchronize all clients using MPI barrier;
   loop to write data;
   close the file;
   t2 = current time;
   ct = t1 – t2; /* overall completion time */
   send ct to client 0;

for client 0:
   find maximum of ct; /*find slowest client */
   calculate write throughput using maximum
```

Fig. 2. Pseudocode of the benchmark

Impact of CPU, disk, memory and network workload on write performance

In a cluster environment, applications usually have different requirements for system resources, primarily CPU, disk, memory and network. In this section, we will present the impact of different workload conditions of CPU, disk, memory, and network on the performance of CEFT-PVFS. Since CEFT-PVFS, an RAID-10 style system, strides the files among the data server nodes in a round-robin fashion and the write performance is largely determined by the slowest data server in one storage group, it is essential to understand the characteristics and behaviors of individual data

servers under a variety of system resource utilizations, in order to be able to make load-balancing decisions dynamically. To make this problem tractable, we measure the performance of CEFT-PVFS in its simplest configuration, in which either group contains only one data server and one metadata server, and in its simplest I/O access pattern, in which only one client writes a new file to the data server. While we artificially put different stresses on one of the recourses of the data server and keep the other resources idle, we measure the write performance with increasing I/O loads, i.e., increasing the file size. Each measurement is repeated 30 times and the average value is calculated after discarding the 2 highest and the 2 smallest measurements. The measurement results are detailed in the following paragraphs.

Impact of CPU workload on write performance

While CPUs in general are not the bottleneck for I/O operations, CPUs on the data server nodes, which usually also serve as computation nodes in a cluster, may be heavily loaded by scientific applications, especially compute-intensive programs, thus potentially increasing the I/O response time. As the experiments show, the influence of CPU load on the I/O performance is insignificant when the usage of both CPUs is below 99%. We define the average CPU load as the average number of processes ready to run during the last minute. To artificially make the load of an idle CPU a specific number, such as three, we can fork three processes and let each process execute an infinite busy loop. Figure 3 shows the write performance under different CPU loads while both CPUs on the data server node are 99% utilized and the memory, disk and network are nearly 100% idle. The experiments indicate that the write performance will be reduced by approximately 31%, 60%, 70%, and 73% on average if the CPU is busy and the average load is 1, 2, 3, and 4, respectively.

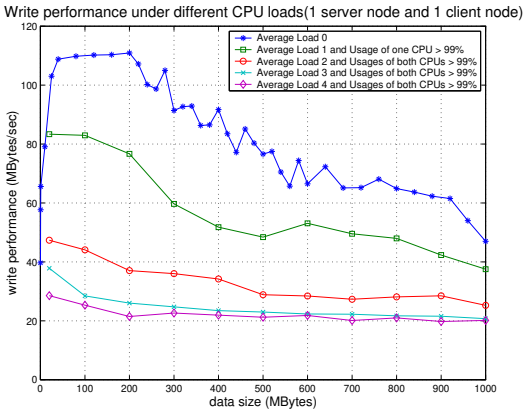

Fig. 3. The influence of CPU loads on write performance under different data size

Impact of network traffic load on write performance

CEFT-PVFS uses the TCP/IP to transfer data between the client and server nodes. The TCP/IP performance over the 2 gigabits/s full-duplex Myrinet of PrairieFire is measured using Netperf [18]. Based on the basic client-server model, Netperf measures the performance by sending bulk data between the server and the clients. Figure 4 shows the TCP/IP performance when different numbers of Netperf clients simultaneously communicate with one Netperf server. All the Netperf clients and server are located at different nodes in the cluster. As the figure shows, the server has an average of 126.51 MBytes/s TCP/IP throughput, which is shared by all the clients. Our test using gm_debug facilities [17] indicates that, while the PCI bus has a read and write throughput of 236 MBytes/sec and 209 MBytes/sec respectively, the average CPU utilization of the Netperf server is only 47% during the measurement. This suggests that the bottleneck of the TCP/IP performance is likely located at the TCP/IP stack at the server side, which requires an integrated memory copy and thus generates an extra, potentially large latency. In addition, this figure also shows that when more than five nodes concurrently communicate with the same node, the average throughput of an individual node is less than the maximum disk throughput, implying that when there are communication-intensive applications running on the CEFT-PVFS server nodes, the bottleneck of I/O operations could potentially shift from disks to the TCP/IP stack.

The write performance under different numbers of Netperf clients is measured and given in Figure 5 when Netperf server and the CEFT-PVFS data server are deliberately placed on the same node. When the data size is small, the Netperf client nodes and the CEFT-PVFS client nodes share the TCP/IP bandwidth nearly evenly. With the increase in data size, the performance further degrades due to the compounded negative impact of memory shortage.

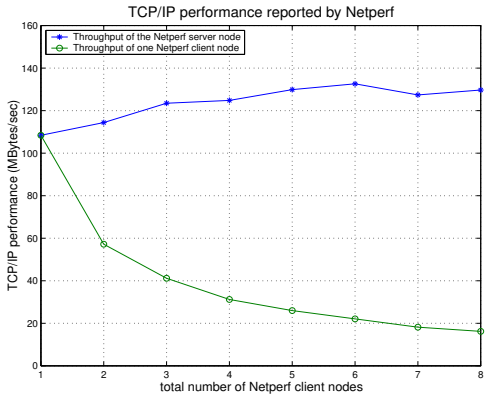

Fig. 4. TCP/IP performance when different client nodes concurrently communicate to one server node

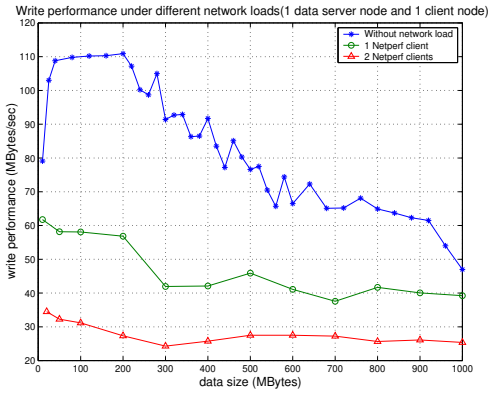

Fig. 5. Write performance under different network traffic disturbance

Impact of memory and disk load on write performance

Memory and disk are closely coupled in almost all modern operating systems. In this paper, we only analyze the overall impact of disk and memory. A simple program, shown in Figure 6, is developed to stress the disk and memory on data server nodes. In this program, the synchronous write always guarantees a disk access, but the operating system usually places the most recently used data in the cache buffer in an effort to avoid some disk accesses. Although this caching buffer can be automatically reclaimed by the operating system, the competition for memory between this program and CEFT-PVFS on the server node will certainly reduce the write performance. As we have measured, when only this program is running, both CPUs are nearly 95% idle and therefore will likely little or no negative impact on the write performance during this set of measurements. Another observation is that, like the network characteristics shown in Figure 3, the disk bandwidth is nearly equally shared by all the I/O-intensive processes running on the same node, i.e., if there are five processes concurrently writing data into the same disk, the I/O performance of each process would be around 8 MBytes/s when the maximum disk write throughput is 40MBytes/s.

```
1. M = allocate(1 MBytes);
2. create a file named F;
3. while(1)
4.   if(size(F) > 1 GB)
5.      truncate F to zero byte
6.   else
7.      synchronously append the data
8.      in M to the end of F.
9. end of while.
```

Fig. 6. Program to stress the memory and disk

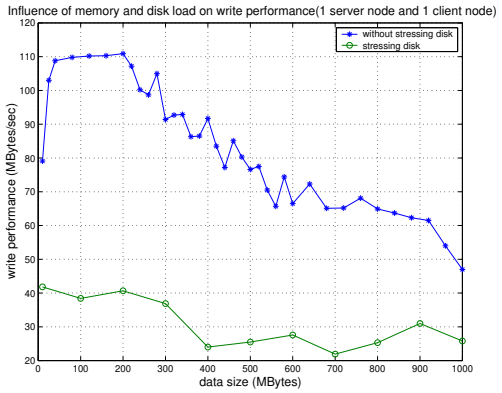

Fig. 7. Write performance when the memory and disk are heavily loaded

Figure 7 plots the write performance when the memory and disk are heavily loaded. When the above program runs on the CEFT-PVFS server node, the performance drops nearly 64% even when the data size is only a small fraction of the total size of the physical memory. Under this heavy disk and memory stress, write performance approximates the disk maximum throughput even when the file size is small. When data size is large, the write performance drops below the maximum disk throughput since the data is too large to be buffered so that the disk bandwidth has to be shared. We conclude that when the CPU load is not high, the disk-memory "compound" plays a more significant role than the network.

To skip or not to skip a busy node while striping?

When the system resources on one mirroring pair are heavily loaded, it might be beneficial to skip these nodes while striping, in order to balance the write load among the designated group of mirroring pairs. Can skipping the busy nodes compensate for the reduced parallelism? To answer this question, we need to exam how the performance scales with the total number of data server nodes when all the server nodes are lightly and equally loaded.

Figures 8 and 9 show the aggregate performances corresponding to two cases: constant-sized files being written by a variable number of client nodes, and variable-sized files being written by a constant number of client nodes, given that all the nodes are not heavily loaded. The average peak performances in the saturated region in Figure 8 of the three different CEFT-PVFS configurations are 492, 796 and 1386 MBytes/s respectively, which are nearly proportional to the total number of data servers, thus indicating a good scalability of CEFT-PVFS. This scalability, however, does not necessarily hold in the unsaturated regions in Figure 9, implying that a larger number of server nodes does not necessarily result in a proportionally higher write performance. In fact, the opposite is true when the file size falls in the range of 0.5 to 8 MBytes. In other words, for some file sizes a larger number of server nodes result in

lower performance, and vise versa. It is this counter-intuitive property that, shown in both figures, highlights the necessity of skipping some data servers to improve the overall performance, even when all the server nodes are well balanced, which will be not addressed in this paper. Nevertheless, judiciously skipping server nodes, or equivalently resizing the striping group, in a well-balanced system to improve write performance, while necessary, is beyond the scope of this paper, and thus will not be addressed.

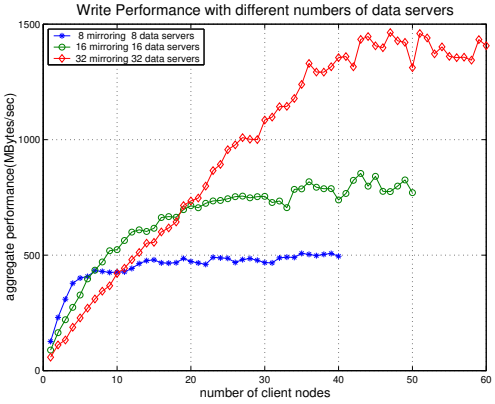

Fig. 8. Aggregate write performance of CEFT-PVFS when each client writes 16 MBytes data to the servers

Fig. 9. Aggregate write performance of CEFT-PVFS when the total number of clients nodes is 16

In a realistic setting of cluster computing, the workload on all the data servers could be significantly different since parallel scientific applications usually are scheduled to run on only a portion of nodes, instead of every node. It is possible, in fact, rather

likely, that one mirroring pair are both heavily loaded, thus degrading the overall performance substantially. In such cases, skipping the busy pair helps alleviate the negative impact of the pair due in part to their dual roles in the cluster as a CEFT-PVFS server node and as a compute node. Experiment results show that if a mirroring pair is heavily loaded and the maximum I/O throughput that they can provide is only about half of the disk bandwidth, skipping this mirroring pair usually improves the overall performance. This observation is helpful in developing the heuristic scheduling algorithm, to be described later.

The skipping is most beneficial to the data that are written once and are rarely rewritten or read. A cluster is a highly dynamic environment with a constantly changing workload on each node. Therefore, although skipping a server may boost the write performance at the time of the first write, if the file is going to be rewritten or read many times, the performance benefit will be amortized by future reads and writes. Thus in this paper, only written once application is addressed for skipping.

Scheduling algorithm

Previous sections presented, quantitatively, the impact of resource availability of various kinds on the behaviors of write operations in its simplest configuration and under the simplest workload pattern. In addition, experimental results suggest that judiciously skipping some server nodes while striping can be potentially beneficial to performance enhancement, especially for write-once applications. While such simplistic but quantitative results about performance impact of resource availability may not be directly extended to a CEFT-PVFS with multiple data servers and more complex I/O workload, the relative sensitivities of resource availability of different kind and the scalability information implied can give useful heuristic hints to the development of a dynamic scheduling algorithm for load balancing.

Since the metadata server is responsible for all the scheduling work, which can potentially form a bottleneck, we try to keep the schedule algorithm as simple as possible to reduce the scheduling overhead. A straightforward algorithm is developed in this paper. In this algorithm, we only consider skipping at most one data server in a striping group to reduce the intrinsic scheduling complexity. Based on our experiences, skipping one node that can provide at most half of the maximum disk throughput significantly boosts the overall performance. Thus the value of one half of the maximum disk throughput is used as the threshold to decide on skipping.

The basic idea of this algorithm is that for each mirroring pair, if it is not heavily stressed, choose one node that could potentially deliver a higher I/O throughput from each mirroring pair to construct the primary storage group. In addition, according to the skipping reasons, all these pairs are sorted into four groups, CPU, memory, disk, and network. While each group is assigned a different priority and the priorities from the highest to the lowest are network, memory, disk and CPU, a pair in the non-empty group with the highest priority will be randomly chosen to be skipped.

In this dynamic scheduling algorithm, the available disk throughput D_i on node *i* is estimated as $\min(D_{max}-D_{used}, D_{max}/(n+1))$, where D_{max}, D_{used} and *n* are the maximum disk throughput, the disk throughput of the last interval, and the total number of

processes that are doing I/O operations respectively. The available network throughput is estimated in a similar way. The size of the free memory space on a node is obtained from the memory management system of the operating system kernel. All these parameters are stored on the metadata server. The data server nodes collect this information and send it to the metadata server every one-second.

Function:
choose a data server between node *i* and its mirrored node *j*
Inputs:
NET_i, NET_j: available network throughput on node *i* and *j*
MEM_i, MEM_j: size of available free memories on node *i* and *j*
$DISK_i$, $DISK_j$: available disk throughput on node *i* and *j*
CPU_{i1}, CPU_{i2}, CPU_{j1}, CPU_{j2}: average CPU usages on node *i* and *j*
$LOAD_i$, $LOAD_j$: average CPU load on node *i* and *j*
$DISK_{max}$: maximum disk throughput
FSIZE: size of the desired portion of the destination file

Algorithm:
1. if *min*(CPU_{i1},CPU_{i2},CPU_{j1},CPU_{j2}) ≥ 99% and *min*($LOAD_i$,$LOAD_j$) > 2
2. set the skipping flag;
3. else if *min*(CPU_{i1}, CPU_{i2}) ≥ 99% and $LOAD_i$ > 2
4. choose node *j*;
5. else if *min*(CPU_{j1}, CPU_{j2}) ≥ 99% and $LOAD_j$ > 2
6. choose node *i*;
7. else if MEM_i > FSIZE and MEM_j > FSIZE
8. if NET_i ≤ 0.5*$DISK_{max}$ and NET_j ≤ 0.5*$DISK_{max}$
9. set the skipping flag;
10. else
11. choose *i* if NET_i ≥ NET_j; otherwise, choose *j*;
12. end
13. else if MEM_i ≤ FSIZE and MEM_j ≤ FSIZE
14. if min($DISK_i$, NET_i) ≤ 0.5*$DISK_{max}$
15. and min($DISK_j$, NET_j) ≤ 0.5*$DISK_{max}$
16. set the skipping flag;
17. else
18. choose *i* if min($DISK_i$, NET_i) > min($DISK_j$, NET_j);
19. otherwise *j*;
20. end
21. else
22. choose *i* if MEM_i > FSIZE > MEM_j and NET_i ≥ 0.5*$DISK_{max}$
23. choose *j* if MEM_j > FSIZE > MEM_i and NET_j ≥ 0.5*$DISK_{max}$
24. choose *i* if MEM_j > FSIZE > MEM_i and NET_i ≥ 0.5*$DISK_{max}$
25. choose *j* if MEM_i > FSIZE > MEM_j and NET_j ≥ 0.5*$DISK_{max}$
26. otherwise set the skipping flag;
27. end

Fig. 10. Scheduling algorithm for I/O requests

Performance evaluation

In this section, we will evaluate our heuristic scheduling algorithm in a CEFT-PVFS that has eight data servers in one striping group. To fairly compare the performance with scheduling and without scheduling, the benchmark programs need to be executed in the same environment with identical workload. In a real cluster in production mode, such as the PrairieFire in which CEFT-PVFS is installed, unfortunately, it is nearly impossible to obtain such a repeatable environment since the workload on each node is constantly changing with the progression of applications running in the cluster. Therefore, instead of doing comparisons in a real environment, we compare performances in an artificially created environment in which the load of a specific kind of resource on a server node is kept approximately constant by using the programs described in the previous sections, although interferences from other computation programs running on the cluster can not be avoided.

To make sure that the bottleneck of the I/O operation is located on the server side rather than the client side, 16 client nodes are used to simultaneously write to the server and the aggregate performance is measured. Two sets of experiments are conducted. In the first set, the workload stress is applied only on one node while its mirroring node is kept almost idle so that skipping will not be necessary. In the second set, the workload stress is put on both nodes of a mirroring pair so that it will become necessary to skip. In each set of experiments, the CPU, network, and the disk-memory compound are each stressed in turn, and the results are presented in the following figures. In each figure, the average write performance of the scheduling algorithm is shown, since under different stress conditions of the same resource, the performances of the scheduling algorithm are very close.

Figures 11 and 12 show results of experiments in which the CPU and network of one primary node are stressed, respectively. In experiments reported in Figure 13, both the disk and memory are stressed on one node or on two nodes in the same striping group. In Figures 14 and 15, the CPU and network of one mirroring pair are stressed simultaneously. Figure 16 gives the comparison when both the disk and memory on one mirroring pair are stressed. The performance of the scheduling is significantly better than the performance of non-scheduling in the vast majority of the test cases.

In the cases of skipping, shown in Figures 14-16, the aggregate performance of the scheduling algorithm starts to decrease sharply when the data size of each client is larger than 64MBytes. This sharp decrease is due to the fact that, as data size from each client node increases, the total file size allocated on one server node becomes so significant that the negative impact of load redistribution (as a result of skipping) onto the remaining 7 server nodes quickly offsets the positive gain from skipping. These figures show that when one of the resources on a server node is heavily loaded, our scheduling algorithm derived from the heuristic observations, can significantly improve the write performance.

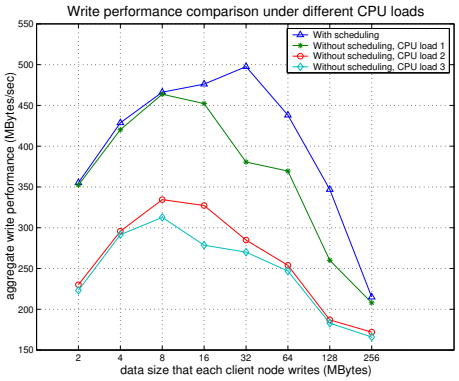

Fig. 11. Stress CPUs on one server node

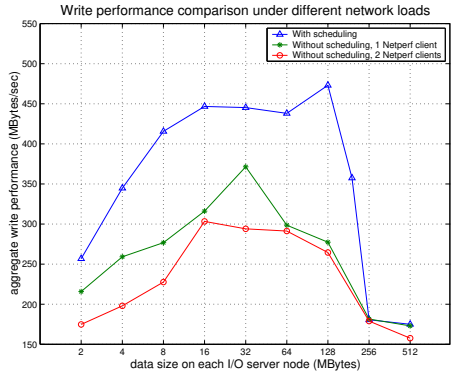

Fig. 12. Stress network on one server

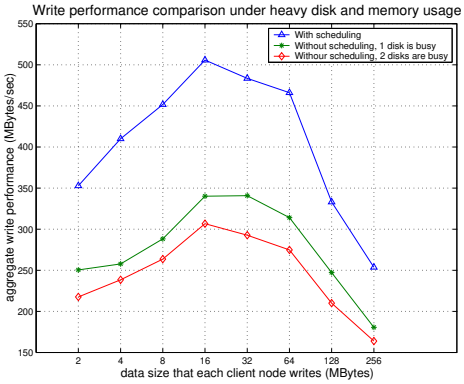

Fig. 13. Stress disk and memory on one server node

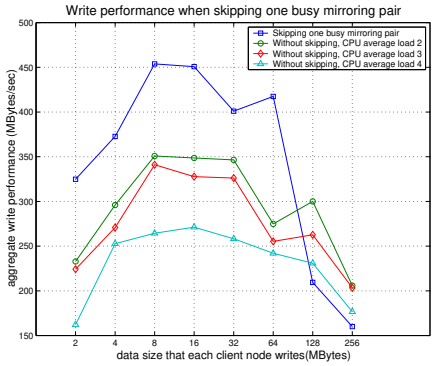

Fig. 14. Stress CPUs on one mirroring pair

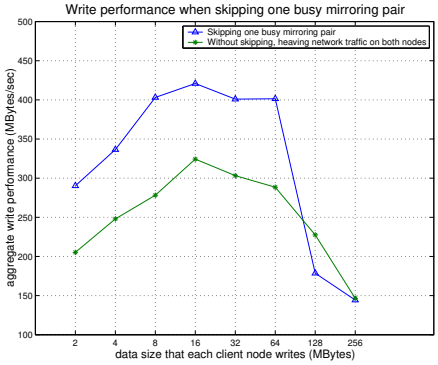

Fig. 15. Stress network on one mirroring

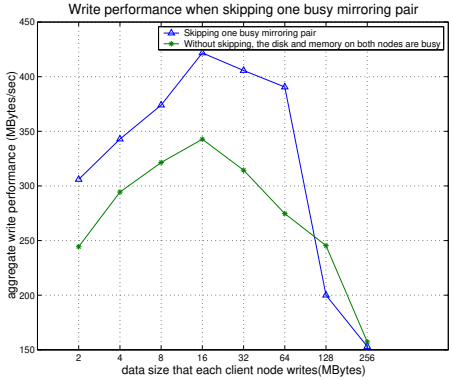

Fig. 16. Stress disk and memory on one mirroring pair

Figure 17 shows the comparison of our scheduling algorithm with two other algorithms, one solely based on the availability of disk and memory, and the other solely based on the availability of network bandwidth. This figure clearly shows that two simplistic algorithms are inferiors to ours since both of them are limited by the amount of information on which their decisions are based while our algorithm bases its decision on a more comprehensive piece of system workload information. The performance in Figure 17 is a little higher than the performance in the other figures because Figure 17 is measured immediately after the reboot of our cluster and there is almost no computation application except our load stress program.

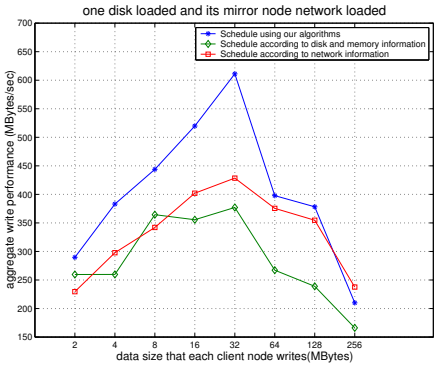

Fig. 17. Stress disk and memory on one node and stress the network on its mirroring node

Conclusions and Future Work

A reliable and high-performance I/O and storage system is of paramount importance to parallel and distributed computing. CEFT-PVFS extends PVFS by incorporating mirroring to provide fault tolerance. Although the peak performance of CEFT-PVFS can reach the level of GBytes/sec, the performance can be significantly reduced if the data servers, which also serve as computational nodes in a cluster, are heavily loaded by applications running in the cluster. Since in CEFT-PVFS each data item has two copies and they are stored in two different nodes, it provides an opportunity for scheduling algorithms to exploit the workload disparity between these two nodes in a mirroring pair or among all mirroring pairs within a striping group. Therefore, in this paper a heuristic scheduling algorithm is developed to schedule the I/O requests to the node with less workload in a pair or to skip a heavily loaded pair altogether while striping, in a hope to improve I/O write performance. To reduce the intrinsic complexity of this scheduling problem and to understand the workload characteristics in a cluster, we conducted experiments on simplest CEFT-PVFS configurations, under specific and isolated workload conditions. From the experimental results, we drew heuristic hints to help design the algorithm. For example if the CPU usage and

load are less than 99% and 2 respectively, CPU will not have a significant influence on the overall performance. Otherwise, overall performance will be substantially degraded. Based on these heuristic hints, a simple but effective algorithm is developed and its performance is evaluated in a CEFT-PVFS with 16 servers under different workload conditions. The performance measured shows that this scheduling algorithm is capable of significantly improving the overall I/O write performance when the system is under an unbalanced workload.

While we designed and implemented the prototype of the dynamic scheduling algorithm, many important challenges remain. Our future work is to provide a more generic and platform-independent algorithm. We need to use more realistic benchmark to measure the I/O performance. More experiments are necessary to justify some parameters of the heuristic rules and detail the rationale behind our algorithm.

Acknowledgement

This work was partially supported by an NSF grant (EPS-0091900) and a Nebraska University Foundation grant (26-0511-0019). Work was completed using the Research Computing Facility at University of Nebraska – Lincoln. Finally, we are grateful to our anonymous reviewers.

References**:**

[1] P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur, “PVFS: A Parallel File System For Linux Clusters," in *Proceedings of the 4th Annual Linux Showcase and Conference*, Atlanta, GA, Oct. 2000, pp. 317-327.

[2] David A. Patterson, Garth Gibson, and Randy H. Katz, “A case for redundant arrays of inexpensive disks (RAID),” in *Proceedings of the 1988 ACM SIGMOD international conference on Management of data.* 1988, pp. 109-116.

[3] Yifeng Zhu, “CEFT-PVFS: A cost-effective, fault-tolerant parallel virtual file system”, *Master Thesis*, Department of Computer Science and Engineering, University of Nebraska – Lincoln, Dec. 2002.

[4] Yifeng Zhu, Hong Jiang, Xiao Qin, Dan Feng, and David R. Swanson, “Improved read performance in a Cost-Effective, Fault-Tolerant Parallel Virtual File System (CEFT-PVFS),” in *Proceeding of IEEE/ACM Workshop on Parallel I/O in Cluster Computing and Computational Grids, in conjunction with IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, Tokyo, Japan, May 2003

[5] Avery Ching, Alok Choudhary, Wei-keng Liao, Robert Ross, and William Gropp, "Noncontiguous I/O through PVFS," in *Proceedings of 2002 IEEE International Conference on Cluster Computing*, Sept. 2002.

[6] W.B Ligon III, “Research Directions in Parallel I/O for Clusters,” keynote speech, in *Proceedings of 2002 IEEE International Conference on Cluster Computing*, Sept. 2002.

[7] Joseph D. Touch, "Performance analysis of MD5," in *ACM Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, New York, NY, USA, Oct. 1995, pp. 77-86.

[8] Murali Vilayannur, Mahmut Kandemir, and Anand Sivasubramaniam, "Kernel-level caching for Optimizing I/O by exploiting inter-application data sharing," in *Proceedings of 2002 IEEE International Conference on Cluster Computing*, Sept. 2002.

[9] Apon, A.W., Wolinski, P.D., and Amerson, G.M. "Sensitivity of cluster file system access to I/O server selection," in *Cluster Computing and the Grid 2nd IEEE/ACM International Symposium CCGRID2002*, 2002, pp. 183-192.

[10] R. B. Ross, "Reactive Scheduling for Parallel I/O Systems," *Ph.D. Dissertation*, Clemson University, Dec. 2000.

[11] Ligon, III, W.B., and Ross, R. B., "Server-Side Scheduling in Cluster Parallel I/O Systems," *Calculateurs Parallèles Journal Special Issue on Parallel I/O for Cluster Computing*, Oct. 2001.

[12] Michael O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance", *Journal of the ACM*, vol. 36, no. 2, pp. 335-348, Apr. 1989

[13] Azer Bestvaros, "IDA-based redundant arrays of inexpensive disks", in *Proceedings of the First International Conference on Parallel and Distributed Information Systems*, pp. 2-9, Dec. 1991

[14] J. H. Hartman and J. K. Ouserhout, "The Zebra striped network file system," *ACM Transactions on Computer Systems*, vol. 13, no. 3, pp. 274-310, Aug. 1995

[15] Luis-Felipe Cabrera and Darrel D. E. Long, "Swift: using distributed disk stripping to provide high I/O data rates," in *Computing Systems*, vol. 4, no. 4, 1991

[16] Prairiefire Cluster at University of Nebraska - Lincoln, http://rcf.unl.edu, 2003.

[17] Myrinet, http://www.myrinet.com/, 2003.

[18] Netperf, http://www.netperf.org/netperf/NetperfPage.html, 2003.

[19] Bonnie, http://www.textuality.com/bonnie, 2003

[20] Hakan Taki and Gil Utard, "MPI-IO on a parallel file system for cluster of workstations," in *Proceedings of the IEEE computer Society International Workshop on Cluster Computing,* Melbourne, Australia, 1999, pp. 150-157

[21] Rosario Cristaldi, Giulio Iannello, and Francesco Delfino, "The cluster file system: Integration of high performance communication and I/O in clusters," in *Proceeding of the 2nd IEEE/ACM international symposium on Cluster computing and the grid,* Berlin, Germany, May 2002.

[22] Steven A. Moyer and V. S. Sunderam, "PIOUS: A scalable parallel I/O system for distributed computing environments," in *Proceedings of the Scalable High-Performance Computing Conference,* 1994, pp. 71-78.