An Experimental Performance Evaluation of Touchstone Delta Concurrent File System[*]

Rajesh Bordawekar[†] Alok Choudhary[‡] Juan Miguel del Rosario[§]

Northeast Parallel Architectures Center
111 College Place, Rm 3-201
Syracuse University
Syracuse, NY 13244-4100

Abstract

For a high-performance parallel machine to be a scalable system, it must also have a scalable parallel I/O system. Recently, several commercial machines (e.g. Intel Touchstone Delta, Paragon, CM-5, Ncube-2) have been built that provide features for parallel I/O. However, very little is understood about the performance of these I/O systems. This paper presents an experimental evaluation of the Intel Touchstone Delta's Concurrent File System (CFS). The CFS utilizes the declustering of large files across the disks to improve the I/O performance. Data files can be read or written on the CFS using 4 access modes. We present performance measurements for the CFS on the Touchstone Delta with 512 compute nodes and 32 I/O nodes. The study focuses on file read/write rates for various configurations of I/O and compute nodes. The study attempts to show the effect of access modes, buffer sizes and volume restrictions on the system performance. The paper also shows that the performance of the CFS can greatly vary for various data distributions commonly employed in scientific and engineering applications.

[*]This work was sponsored in part by DARPA under contract # DABT63-91-C-0028 and NSF MIP-9110810. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred. We also acknowledge the support of Center for Research in Parallel Computing for providing access to the computing resources.

[†]ECE Dept., Syracuse University

[‡]ECE Dept., Syracuse University, Corresponding Author

[§]CS Dept., Syracuse University

1 Introduction

During the last decade, processor speeds have increased tremendously and this trend is likely to continue for the foreseeable future. During the same period, several important strides have been made in high-performance computing architectures. High speed processors coupled with massive parallelism are expected to provide computing power in teraflops in the next few years.

Some of the commercially available parallel computers include Intel Paragon [Int92], nCUBE [nCU92], CM-5 [Thi91]. They are computational instruments of choice in the scientific community and can be found in one form or another in almost every major academic and research institution. Several prototypes are also being developed at various industrial and academic institutions. Examples of such machines are DASH [DJK+92], Alewife [ACD+91], J Machine [DCF+86] and Tera [ACC+90]. However, compared to the attention accorded to processors, interconnection networks and memories of these parallel systems, very little attention has been paid to the scalability and performance of the I/O system.

Providing raw processing speed and large memories without balancing I/O capabilities, however, is not sufficient in solving many real-world problems. A balanced memory hierarchy which can supply data to processors at the required speeds is critical to the success of high performance computing. Although semiconductor memories have become cheaper and faster, I/O systems' performance has not kept pace with the advances in processor and memory speeds. Very few parallel I/O systems have been developed which can balance the processor speeds and the I/O bandwidth.

In this paper, we focus on the experimental evaluation of the Concurrent File System of the Intel Touchstone Delta. The goal is to study the effects of various workloads on the performance of a parallel I/O system. Based on the experimental study, we identify various parameters that affect the system in a significant way.

The rest of the paper is organized as follows. In section 2, we describe the parallel I/O model and the Intel Touchstone Delta system. Section 3 discusses the evaluation methodologies and lists some of the experiments presented in the paper. Section 4 presents performance evaluation results for various experiments. Array distribution results are presented in section 5. Finally, we conclude the paper in section 6.

2 The Touchstone Delta System

The Touchstone Delta system was developed by the Intel Corporation as a part of the Touchstone program. The Intel Touchstone Delta system is a message passing multicomputer consisting of

Figure 1: Intel Touchstone Delta System

processing nodes that communicate across the two dimensional mesh interconnection network.

The system supports various types of processing nodes (numeric, mass storage, gateway and service). Numeric nodes form the computational core of the system. The Delta system uses of Intel i860 processors as the core of computational nodes. In addition the Delta has 32 Intel 80386 processors as the core of I/O nodes. Each I/O node has 8Megabytes memory that serves as I/O cache. There are other processor nodes such as service nodes and ethernet nodes. The Delta is arranged as a mesh of 16*32 compute nodes and has 16 I/O node on each side.(Figure 1).

2.1 Concurrent File System

The Intel Touchstone Delta system consists of mesh connected compute nodes with attached set of I/O nodes. Each I/O node is connected to 2 disks, each with 1.4 Gigabytes of space. I/O nodes do not run any application processes but provide disk services for all users. A file is uniformly distributed over all 64 disks by default in a round-robin manner. The stripe unit is 4Kilobytes(one block) per disk. When a file is opened for reading or writing, data is accessed by default from 64 disks. A user, however, can restrict the number of disks on which a file is distributed. All read-write transactions are carried out in an integral number of blocks, where each block size is 4 Kilobytes.

2.2 CFS File Structure

The CFS provides a UNIX view of a file to the application programs. Each CFS file has a header and a body. CFS file header stores file information such as file size, permission and link count. The file header is always allocated the first file block. In case of small files the header contains the data whereas for large files the header contains the pointers to the indirect blocks that store the data. When the file is striped across the disks, the file header is stored on the first disk and all the subsequent blocks are distributed in a round-robin fashion over the disks.

2.3 The I/O Modes

Four I/O modes are supported in the CFS. These are described below.

- Mode 0: In this mode, each node process has its own file pointer. This mode is specifically useful for large files to be shared among the nodes. Here, sharing implies that the same data is accessed by nodes (replicated). This should be distinguished from sharing a file but distributing the data, i.e, when different nodes access different (and distinct) parts of a file. This mode is useful for accessing the file in GSA access pattern.

- Mode 1: In this mode, compute nodes share a common file pointer. I/O requests are serviced on a first-come-first-serve basis. Nodes can read and write at any point, but they use the same file pointer. Thus GSP file access pattern is obtained.

- Mode 2 : Mode 2 treats reads and writes as global operations. The set of compute nodes that open a concurrent file must read the file in a specified order (in the increasing order of the node-numbers). This mode performs global synchronization in the sense that the second request by any node is blocked until the first request by all nodes in the set is complete. This mode supports synchronized common file pointer. Using this mode, nodes can perform variable length read-write operations. Hence, the requests are serviced in a predefined order.

- Mode 3: Mode 3 is a synchronized ordered mode. The difference between mode 2 and mode 3 is that in mode 3, all read/write operations must be of the same size. This mode also supports global synchronization. Hence, the requests can be serviced in any order, but still the second request by a node is blocked until the first request of all nodes is completed. Hence this mode can be used for obtaining GSI form of file access.

The data accessed by each processor depends upon the mode . During the write operation, the resultant size of the file created depends on the file mode used. Many scientific applications involve automatic distribution of the data across processors. Using different I/O modes, it is very easy to decompose the data across the disks. Note the distinction between mode 0 and the other 3 modes. In mode 0, reads/writes are to the same data in a shared file, whereas in other modes reads/writes are to distinct data (for each node) in the shared file.

2.4 The I/O Network

Touchstone Delta does not provide an independent I/O network. The compute and I/O nodes share a common interconnection network. The same network is used for both interprocess communication and I/O communication.

In Touchstone Delta, for both interprocess communication and I/O, messages travel in the form of packets. Touchstone Delta uses *packet switched wormhole* routing as a communication protocol [Int91, Lio93, Rik92]. Each node of the machine is connected to the mesh using a mesh routine chip (MRC). The message travels from MRC to MRC until it reaches the destination node.

Each message is split into packets of a fixed size (512 Bytes). On the physical level, the packet travels through the network in form of *flits* or *flow control digits* [Lio93, M. 93]. The packet follows a XY routing protocol. The XY direction is specified in the message header. In Touchstone Delta, the message packets always travel first along X direction. During the journey, each MRC decrements the X offset. When the X offset becomes zero, the packet travels in the Y direction. The message reaches the destination when both the X and the Y offset of the message become zero.

Using the same network for interprocessor communication and I/O may cause serious network contention. Also since the I/O nodes are physically at the edge of the mesh, the position of the compute nodes might affect the I/O performance. We will investigate these points in the experimental analysis of Touchstone Delta.

3 CFS Evaluation Methodology

The overall performance of accessing data in a CFS depends on several factors including the number of compute nodes participating in an I/O operation, size of access (buffer size), number of disks, block size, I/O mode and the overall available bandwidth from the I/O system as well as that of the interconnection network (Figure 1). In principle, it is difficult to decouple the influence of some parameters on the performance. Our study includes the following experiments:

- Single Compute Node
    - Single compute node and paged I/O

In paged I/O experiments, we study the effects of buffer size, node position, number of disks and the file size on the throughput. These experiments are carried out for smaller buffer sizes and relatively small file sizes.

  – Single compute node and burst I/O

    For burst I/O experiments, the buffer size is very large. The factors that affect the burst mode throughput include the buffer size and the number of disks.

- Multiple Compute Nodes

  – Multiple compute nodes and paged I/O

    Multiple processors access the file system using various access modes. During multinode accesses, the CFS performance not only depends on the buffer size, number of processors and disks, but also on the file access modes. Hence, the paged I/O experiments are carried for all the four file access modes.

  – Multiple compute nodes and burst I/O

    In burst I/O mode, the buffer size is quite large compared to that in the paged mode. For such file accesses, the parameters that affect the throughput include the number of processors and the number of disks. Furthermore, we study the performance of the CFS when data is distributed using common data distributions found in typical scientific programs.

- Effect of Interconnection network on I/O Performance

  We will investigate the effects of using the common network for both interprocess communication and I/O. We will especially study the effect of compute node position on the I/O performance. The results will help in analyzing the mesh network performance.

Table 1 presents some definitions that will be used throughout this paper.

4 CFS Performance Evaluation

The main objective of the CFS performance evaluation is to determine the maximum read-write rates observed for different configurations. Therefore, the experiments try to saturate the I/O system with the I/O requests so as to obtain a peak performance. Similar performance measurements have been used in the study of Intel iPSC I/O system [FPD91a, FPD91b, FPD91c, FPD91b, Nit92].

Table 1: Definitions of various terms used in the paper

Term	Definition
F	The size of the file distributed over the disks.
N_p	The number of processors accessing the file.
F_p	The amount of file data per processor. $F_p = \frac{F}{N_p}$. (True only for modes 1,2 and 3.)
N_{io}	The number of I/O requests per processor.
b_{io}	The size of the data buffer. Access size requested by the processor
N_D	Total number of disks (for Delta N_D is 64).
B_D	Block Size (Amount of data per block on each disk).

4.1 Single Compute Node

The first part of the study aims to determine the maximum I/O rates obtained for a single compute node. These studies are performed both for paged as well as burst I/O modes. Paged I/O performance is important for implementing and supporting node virtual memory to fetch or store pages on disks. Burst-mode I/O is important for file accesses when a node requires reading/writing large files containing data for an application. For both types of workload we study the maximum throughput obtained from the I/O systems.

4.1.1 Paged I/O

Since there is no virtual memory support currently available on the Delta system, virtual memory was simulated by opening a file and reading (writing) it using fixed size buffers. The buffer size B_{io} indicates the amount of data fetched in each I/O request. In each experiment, the compute node opens a file and reads (writes) it using fixed size buffers. Other parameters varied for the following experiments include the file size, buffer size, node position and the number of disks N_D.

Figure 2 illustrates the performance of implementing paged I/O using various buffer sizes. As the buffer size increases from 1K, N_{io} reduces and consequently the throughput increases. For a buffer size of about 4k, the read rate is about 340Kbytes/sec for all file sizes. This convergence of performance occurs because the buffer size and the block size are both 4Kbytes, and therefore, the requested size is same as the size of data read in one operation from the disk. Thus each I/O request for 4K buffer results in reading the prefetched blocks from the node cache. Beyond 4K buffer size, the throughput increases for small files as a function of buffer size whereas it degrades slightly for larger files. In summary, 4K buffer size seems optimal for most of the files but for very small files, larger buffer sizes perform well. It should be noted that the throughput is not

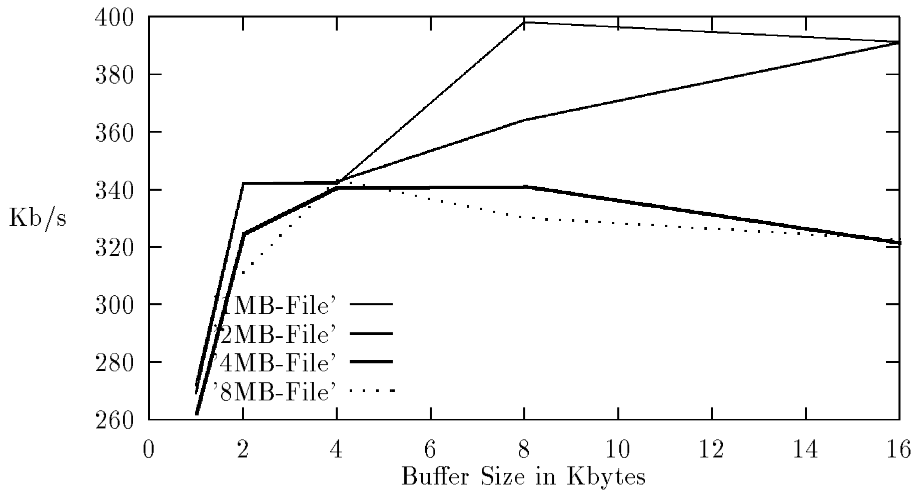

Figure 2: Single Compute Node - 32 I/O Nodes (64 Disks): Read Rates in KB/sec

limited by the I/O system, but is limited by how fast a node request is generated. For the paged I/O experiments, new I/O request is generated only when the previous one is completed. Similar results were obtained for a write operation.

In order to determine if the position of a compute node in the network has any effect on read and write operations, different nodes at various locations in the mesh were chosen as shown in Table 2. Three were on one side of the mesh (Node Nos. 56,11,24), two in middle of the mesh (Node Nos. 319,268), and the remaining three on the other side of the mesh. Keeping the buffer size fixed at 4K, file read times were observed for file sizes varying from 1M to 8M. The files were distributed over 64 disks, so each side had an equal amount of data distributed on the nearby disks (32). The read times do not change significantly as the position of the node varies. This experiment shows that the distance that a request travels in the network does not have any significant impact on the performance when there is very little contention in the network. This also shows that the inter-node "hop times" between the nodes are negligible.

4.1.2 Burst Mode I/O

When the compute nodes require to read (write) a large amount of data (large fraction or an entire file) then the operation may be performed in a burst mode. In burst mode, the buffer size is very large, maximum being the file size being accessed. Figures 3 and 4 show that burst mode operation is much faster than the paged mode. Using 64 disks, 1MB file was read in 203 ms giving a peak

Table 2: File Read Time is (ms)-Single compute node

Node Position	1 MB Read	2 MB Read	4 MB Read	8 MB Read
56	2954	5836	11642	23230
11	2926	5896	11754	23195
24	3018	5836	11749	23314
319	2968	5843	11694	23205
268	2951	5936	11729	23215
567	2994	5903	11941	23264
535	3078	6096	11790	23460
517	2972	5961	11897	23383

rate of 4.83 MBytes/sec. The peak write rate was 1.39 MB/sec. The peak read rate obtained for paged mode was about 400 KB/sec. This increase in the throughput, compared to that in paged mode is observed due to the large number of I/O requests in paged I/O mode where each request must be sent explicitly.

Generally for the single processor configuration, both for the paged and for the burst mode I/O, the read rates are much higher than the write rates. Also as the number of disk volumes increases the throughput increases upto a threshold. For large files, the trend will be the same as the "8-Mbytes" case shown in Figure 4. For small size files, the data is unevenly distributed on disks resulting in an imbalance, and therefore, we observe different trends.

4.2 Multiple Compute Nodes

The most important use of a parallel I/O system and CFS is concurrent accesses by multiple processors. This section presents performance of the file system by varying different parameters such as the number of disks, the access modes and the number of processors.

4.2.1 Mode 0: Paged I/O

Mode 0 is useful for accessing shared files by multiple compute nodes. Each processor has its own file pointer. Note that write operations are not protected in the sense that the processors can overwrite each others data.

Figure 5 shows the read throughput for paged I/O as the function of number of processors for various number of disks. There exists a threshold in terms of the number of disks beyond which a substantial performance gain can be expected. As the number of processors is increased, the performance does not change significantly when N_d is increased from 2 to 32 disks. However for 64 disks, there is a significant jump in the performance. The throughput obtained for 64 processors is

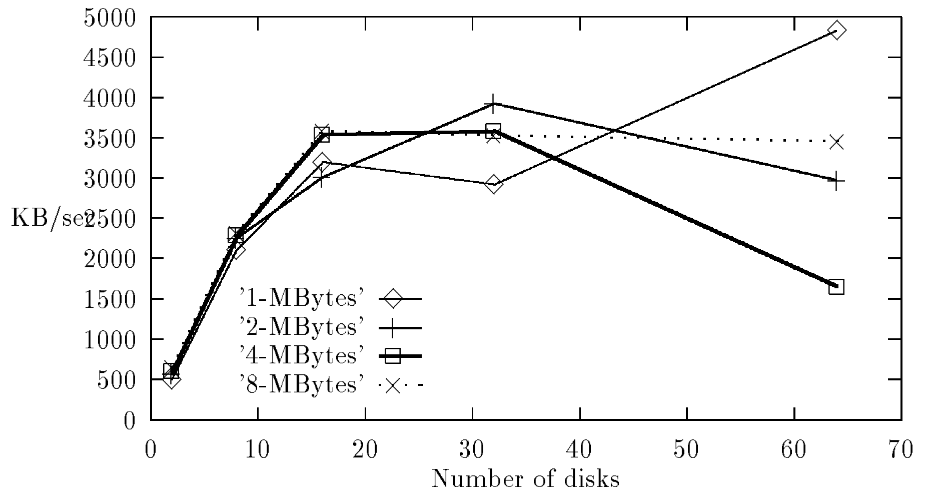

Figure 3: Read Rates for Single Compute Node-Burst Mode

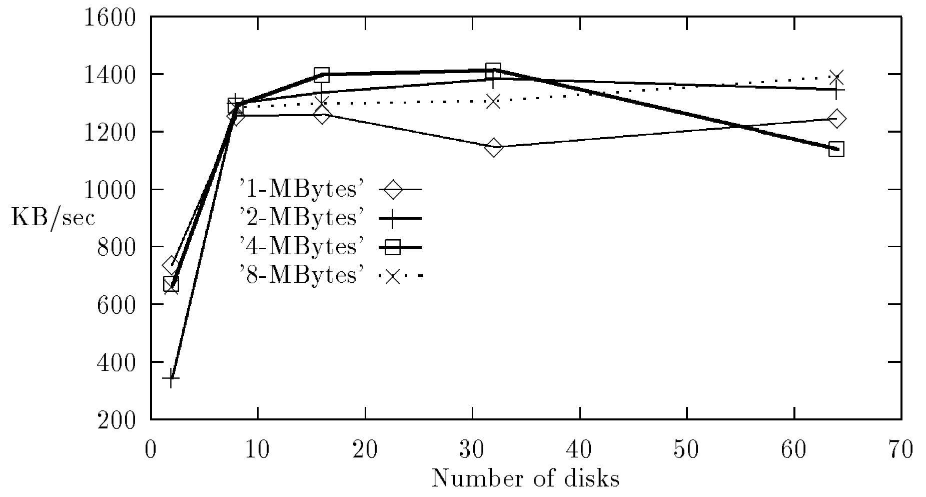

Figure 4: Write Rates for Single Compute Node-Burst Mode

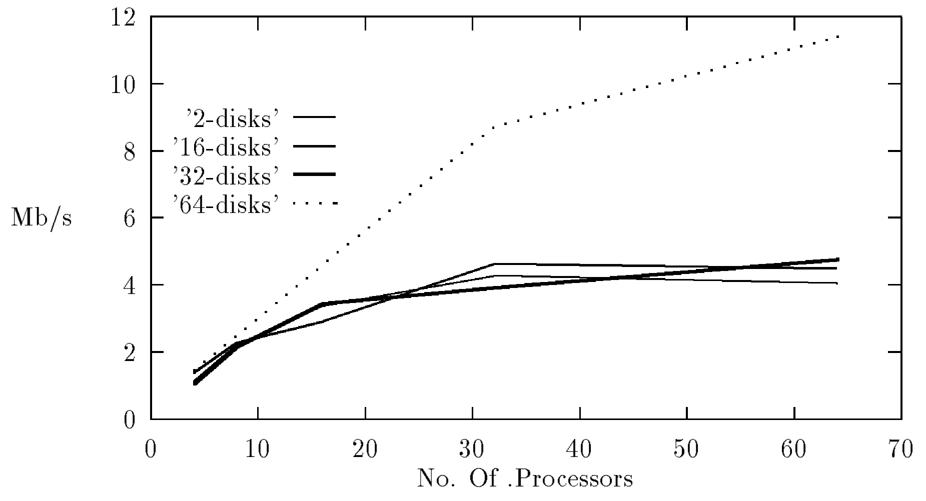

Figure 5: Read Rates For Multiple Compute Nodes - Mode 0

11.2 MBytes/sec. Whereas for 2 disks, the throughput is about 4.5 Mbytes/sec. This shows that the "declustering" of the file data is very effective.

The throughput increases as the number of processors increases. Since each processor reads the same data, as the number of processors increases, the total amount of shared data accessed increases. However, the total time required to read the data increases slowly because one node's read acts as a prefetch command for others.

Another interesting point to observe is that for a small number of processors, the effect of the number of disks on the performance is negligible. That is, the performance is limited by the bandwidth available at the computational node side rather than at the I/O subsystem side. Due to system constraints thsese experiments are carried out for small processor sizes.

4.2.2 Mode 0: Burst I/O

Table 3 shows the performance of burst-mode read/write throughput as a function of number of disks (processor size = 16). The specified buffer size at the application level is equal to the size of the file to be read/written. As we can observe, the performance improves as the number of disks is increased. For a given processor grid size, the performance depends on two parameters, namely, the file size and the number of disks. For smaller files (1 Mbytes/node), the performance saturates for smaller number of disks (32 disks) and beyond which the improvements diminish. However, as the file size increases, the threshold number of disks for which performance improves also increases

Table 3: Multinode (4*4) Burst Mode Throughput as a Function of Disk Volumes (Mode 0)

File size	Buffer size	Disks	Read (MB/s)	Write (MB/s)
1M	1M	2	0.9329	0.748
1M	1M	16	5.88	3.069
1M	1M	32	11.11	3.147
1M	1M	64	11.21	3.506
4M	4M	2	0.74	0.603
4M	4M	16	3.630	2.0878
4M	4M	32	7.705	4.947
4M	4M	64	12.11	6.055

Table 4: Throughput of accessing 1 MB file in Mode 0 (Burst Mode,64 Disks)

Mesh Size	Write Rate(MB/sec)	Read Rate(MB/sec)
4*4	3.506	11.21
4*8	5.421	11.75
8*8	7.027	11.672
8*16	3.314	11.766
16*16	3.244	11.813
16*32	2.839	11.992

as seen for the case of 4 MByte read/write per node.

Tables 4 and 5 presents the performance of burst mode I/O when the number of computational nodes is varied from 16 to 512 ($N_D = 64$). In general for burst I/O, good performance is obtained and the saturation occurs due to bandwidth limitation of the I/O subsystem. Reads perform better than writes. This is because reads can be performed from the I/O cache if a desired block exists in the cache due to an access by some other processor. However, all writes must be written onto the disks. The difference between paged and burst I/O performance is not significant because the system bandwidth is not limited.

Table 5: Thoughput Rates of accessing 2 MB file in mode 0(Burst Mode)

Mesh Size	Write Rate(MB/sec)	Read Rate(MB/sec)
4*4	4.37	10.51
4*8	3.89	11.73
16*16	2.860	12.103
16*32	3.678	23.83

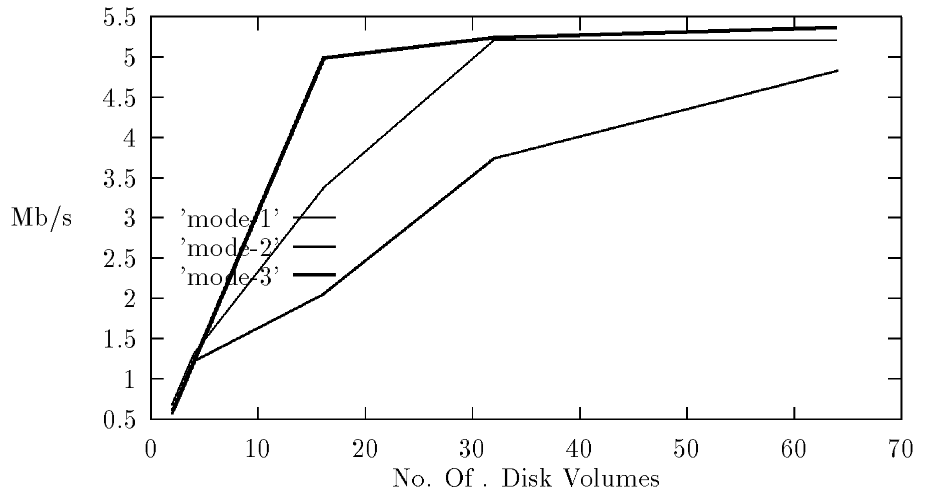

Figure 6: Read for Grid Size 4*4

4.2.3 Modes 1, 2 and 3 : Paged I/O

These modes are useful for accessing data when the data set is distributed over multiple nodes.

The first experiment was used to observe the effect of different modes and the number of disks. In this experiment a data file of size 16 Mbytes was read using modes 1, 2 and 3. As Figure 6 shows, for a 4*4 processor grid the maximum throughput is obtained for mode 3 and with 64 disks. The peak rate in this case is 5.5 Mbytes/sec. For mode 1, the peak speed is 5.12 Mbytes/sec. The lowest file read throughput is observed for mode 2.

Another important point to be noted is that as the number of disks decreases, the read throughput decreases. As the number of disks is decreased below a threshold (in this case 16 disks), the read rate reduces drastically. Hence, the optimal operating point in terms of cost-performance (no. of disks versus the throughput) will be near the knee of the curve.

In the next experiment, we use 64 disks (maximum available) and vary the number of processors. For this experiment, a 16 M file was read using a 4K buffer.

Figure 7 shows that as the number of processors increases the read throughput increases. The highest read throughput was obtained for 64 processors. Modes 1 and 3 perform comparably. The maximum read rate for mode 1 was 10 Mb/sec, while for mode 3, it was 9.8 Mb/sec. Due to access ordering and synchronization costs, mode 2 performs worse than the other modes for all cases. As the number of processors increases, the amount of data read per processor decreases. Hence, the individual processors require less time to read. In other words, the available bandwidth at

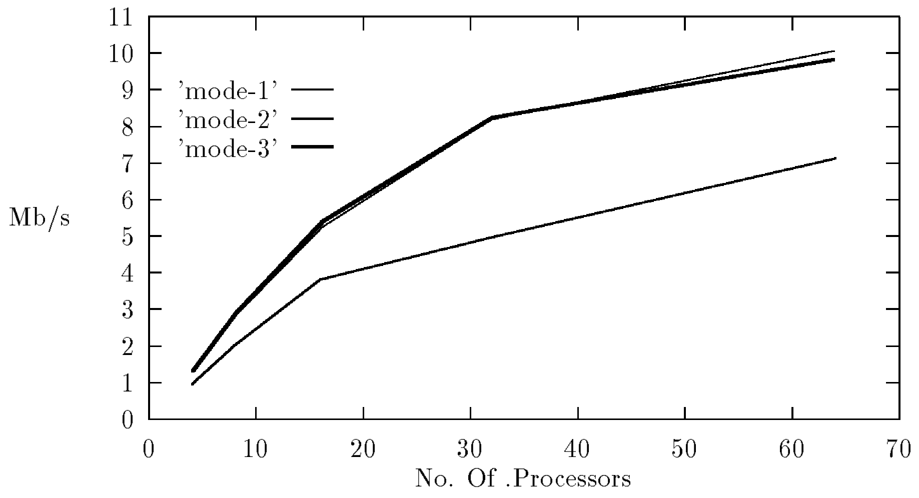

Figure 7: Reading a 16M File using 4K Buffer for 64 Disk Volumes

the computational node side increases resulting in an increase in the throughput. However, as the number of processors increases, the rate of increase in the throughput decreases indicating that the bandwidth limitation shifts to the I/O system side.

For studying the performance of each I/O mode in some more detail, we performed three additional experiments in which the number of processors and the number of disks were varied. For these experiments, a 16 MB file was opened and read by varying number of processors. Figures 8, 9, 10 show the effect of varying disk volumes and number of processors for different I/O modes.

Figure 8 shows the performance of the file system for mode 3. The peak read performance is obtained for 64 disk volumes. This figure also shows that the throughput is proportional to the number of processors. For a 64 processor grid, a 16 MB file was read at 10 MB/sec. For the same grid size, if the file is stored on 2 disk volumes, the read rate drops to about 900 KB/sec.

Note that as the number of processor is increased, the knee of the curve is observed at different points for different number of disks. Hence, as indicated earlier, the choice of number of disks depends on how many processors will be involved in an access. More experiments need to be performed to relate this performance to different file sizes. Note that the knee of the curve in these experiments signifies a point indicating the bandwidth limitation shift from the computational nodes to the I/O system.

Figure 9 shows the results for the same experiment for mode 2. The graph shows nearly same trends as observed for mode 3. However, for mode 2, the peak rate obtained was 7MB/sec for

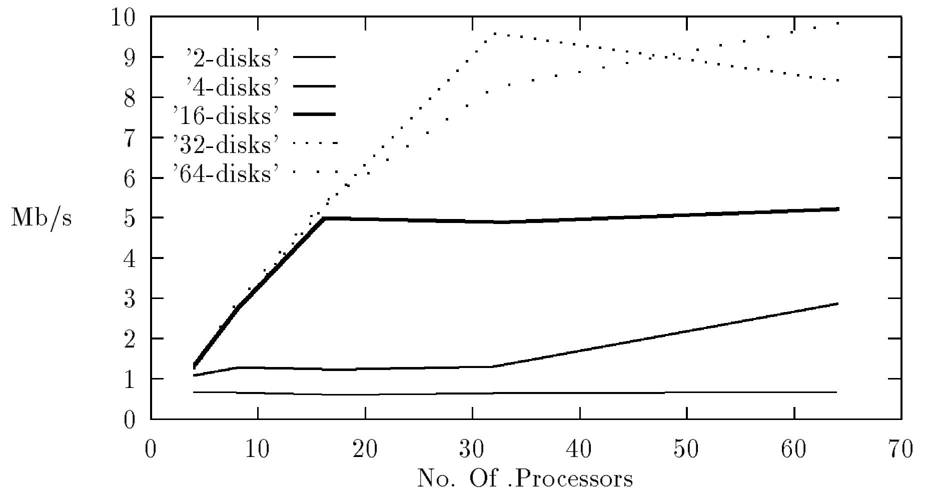

Figure 8: Multicompute Nodes Read (Mode 3 and 16 Mbytes File)

64 processor grid. Note that the performance in mode 2 is sensitive to the order of arrival of the requests because requests must be served in a fixed order. Therefore, we observe a less smooth curve as compared to that for mode 3.

Figure 10 shows the performance of the multicompute nodes for the mode 1. The peak throughput is 10 MB/sec and lowest observed throughput is 900 KB/sec. Note that mode 1 serves requests in the order of arrival and does not require synchronization for each processor to finish before going to the next phase. Therefore, it performs slightly better than mode 3 (which requires synchronization). Therefore, mode 1, is useful for log-structured files or for those computations in which order of accesses does not matter. For example, if the compute nodes perform a search operation in a file, they can access the file in a self-scheduling mode for which mode 1 will provide the best performance.

4.2.4 Modes 1, 2 and 3: Burst I/O

Tables 6 and 7 present a summary of results obtained for reads and writes using burst-mode I/O for various system configurations. The number of processors was varied form 32 (8*4 mesh) to 512 (16*32 mesh). Each processor accessed 1 Mbytes of data, and therefore, the resulting file size varied from 32 Mbytes to 512 Mbytes. Clearly, burst mode I/O is preferable for large file accesses. Furthermore, a consistent performance is observed over a wide range of processor configurations. However, beyond 256 processors (actually, between 128 and 256) the I/O system

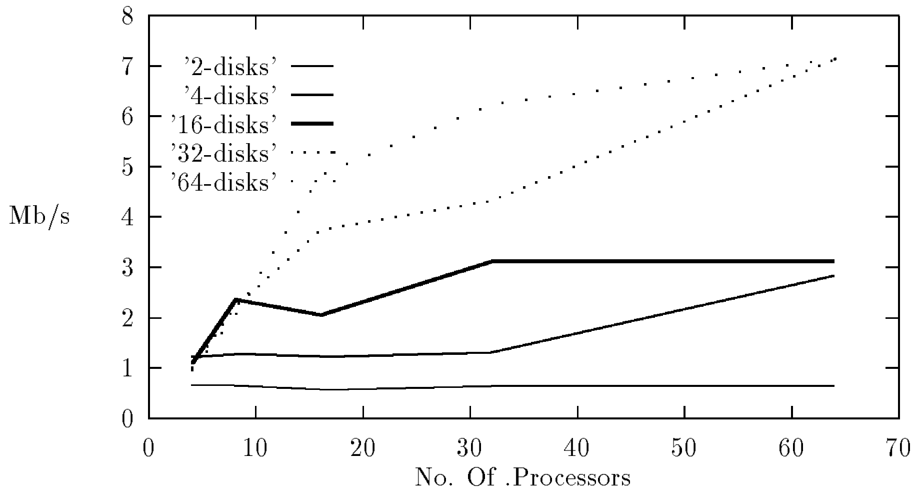

Figure 9: Multicompute Nodes Read (Mode 2 and 16 Mbytes File)

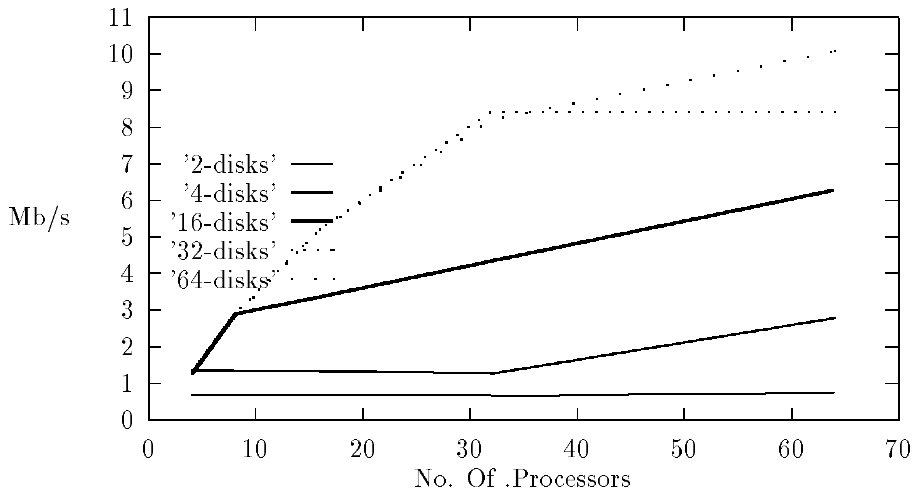

Figure 10: Multicompute Nodes File Read (Mode 1 and 16 Mbytes File)

Table 6: Read Throughput in Mbytes/sec (Burst Mode)

Mesh Size	Mode 1	Mode 2	Mode 3
8*4	8.447	8.159	8.144
8*8	4.817	8.310	9.602
16*8	8.715	8.6821	8.891
16*16	6.519	7.18	7.169
16*32	6.21	6.742	6.944

Table 7: Write Throughput in Mbytes/sec (Burst Mode)

Mesh Size	Mode 1	Mode 2	Mode 3
8*4	4.19	4.310	3.217
8*8	3.622	4.28	3.907
16*8	2.60	3.1545	2.862
16*16	2.53	2.4255	2.479
16*32	2.40	1.9845	2.286

becomes a bottleneck resulting in degraded, but still a comparable performance. It should be observed that read rates were normally 2 to 3 times faster than the write rates.

5 Array Distribution Results

In large-scale scientific and engineering applications, parallelism is exploited by decomposing the input domain (representing the physical domain model, normally represented by multi-dimensional arrays). However, for load-balancing, expressing locality of access, reducing communications and other optimizations, several decompositions and data alignment strategies can be used

In order to enable a user to specify the decomposition, Fortran D [FHK+90], and subsequently High-Performance Fortran [For93], have been proposed. The important feature of these extensions is the set of directives that allow a user to decompose, distribute and align arrays in the most appropriate fashion for the underlying computation. The data distribution directives include BLOCK, CYCLIC and BLOCK_CYCLIC distributions (along any dimension of an array).

In this section, we study the performance of the CFS when the processors access files based on the data distributions. Note that the file provides a linear map (e.g. column major) of multidimensional data. Therefore, the number of I/O requests depend on the specified data distribution on the nodes as shown in Table 8. In this experiment a square character array was distributed across the processors. The array were stored in a column-major form on the disks. The smallest size used

Table 8: Number of I/O requests as a function of data distributions

1-D Distribution			2-D Distribution		
Distr. Type	No. Of Req.	Data per Req.	Distr. Type	No. Of Req.	Data per Req.
Column Block	P	$\frac{N^2}{P}$	Block-Block	$N * \sqrt{P}$	$\frac{N}{\sqrt{P}}$
Column Cyclic	N	N	Block-Cyclic	$N * \sqrt{P}$	$\frac{N}{\sqrt{P}}$
Row Block	N*P	$\frac{N}{P}$	Cyclic-Block	N^2	1
Row-Cyclic	N^2	1	Cyclic-Cyclic	N^2	1

was 1*1Kbytes and the maximum array that was distributed was 20*20 Kbytes (400 Mbytes). For each mesh size, the array was distributed in four ways,column block, column cyclic, row block and row cyclic.

The following is a summary of experimental results.

- Column Block: The column-block distribution implies that the matrix data is distributed along its second dimension onto the processor array. This distribution also conforms with the column-major data distribution over the disk. It requires a single application level I/O request per processor and each processor node can read the entire distributed data in one I/O access. The time required to distribute the data column-wise scales with the number of processors for a portion of the configuration space.

  Table 9 contains the data for a column-block array distribution. The table shows the size of the array, the number of processors participating in the read, the transaction completion time, and the observed bandwidth. For small size arrays and the number of nodes, the bandwidth of the I/O system is underutilized. As the data size and the number of processors increase, the I/O bandwidth is more effectively utilized. However, beyond a certain point, the I/O system becomes a bottleneck due to the large number of processors performing I/O, and the need for synchronization.

  The read rate increases quickly in proportion to the processor grid size, but plateaued at about 64 processors. Degradation in the performance was observed after 256 processors due to a large synchronization overhead. Performance for the small request (1K*1K) case was poor.

- Column Cyclic: Table 10 shows the read access times for the same parameters but with a column-cyclic data distribution on processors. Even though the degree of parallelism in the data access remains the same, the number of I/O requests increases (Table 8) because each processor must make an individual request for each column. This degrades the access time and the bandwidth as illustrated in Table 10. The degradation in the performance in

Table 9: Array Distribution (Column Block) Throughput in MBytes/sec ($N_D = 64$)

Array Size	Mesh Size	Rate (Mode 2)	Rate (Mode 3)
1K*1K	4	2.141	2.32
4K*4K	4	4.198	7.048
5K*5K	16	5.098	7.44
4K*4K	64	5.179	6.498
5K*5K	64	6.476	7.521
10K*10K	256	7.038	7.65
20K*20K	256	5.7	5.861
10K*10K	512	5.232	5.51
20K*20K	512	5.517	5.63

Table 10: Array Distribution (Column Cyclic) Throughput in Kbytes/sec ($N_D = 64$)

Array Size	Mesh Size	Rate (Mode 2)	Rate (Mode 3)
1K*1K	4	160.84	229.72
2K*2K	16	1061.85	1527.3
4K*4K	16	1791.31	3057.51
5K*5K	64	1962.16	2191.63
10K*10K	256	846.92	856.43
20K*20K	512	1522.04	1581.15

consistent for all configurations and it ranges between a factor of 2 to 10 as compared to that for column-block distribution.

- Row Block: Table 10 shows the performance for reading the data array when distributed in a row-block fashion over the processor array. Since the one-dimensional map of the file on the CFS is in column major order, this read operation essentially requires transposing the data while it is being read from disks to nodes. As shown in Table 10, the number of logical request is N*P. Hence, as observed from Table 10, the performance degradation due to this distribution is almost two orders of magnitude when compared to the performance of the column-block distribution. We do not present performance figures for larger configurations (i.e. large array and system sizes) since the time it took to complete these experiments exceeded practical limits. Thus, we merely conclude that performance for this distribution was at least more than two orders of magnitude worse than the first two configurations. The peak bandwidth obtained was 0.69 Mbytes/sec. This is only 30% of the slowest case (the 1*1 Kbytes case) for the column-block decomposition of Table 9 above. Further, the 1K*1K case for this distribution is 39 times slower than for the equivalent column-block case.

Table 11: Array Distribution (Row Block) For Modes 2 and 3 ($N_D = 64$)

Array Size	Mesh Size	Mode	Rates Kbytes/sec
1K*1K	4	2	56.23
2K*2K	4	2	123.84
4K*4K	16	2	114.09
5K*5K	16	2	142.34
1K*1K	4	3	58.64
2K*2K	4	3	154.04
4K*4K	16	3	224.70
5K*5K	16	3	273.11

- Row Cyclic: The row-cyclic distribution involved the largest number of I/O requests. Also the request size was the smallest. It took approximately 15 minutes to distribute 1K*1K character array in row-cyclic order versus the 467 msec it would require in the column-block form. This shows that the direct row distribution of an array is very slow, hence, not possible in practice.

6 Conclusions

To summarize the results, we conclude that:

- The "declustering" of the files improves the read and write performance of the file system for both single and multiple compute nodes.

- For single compute nodes, using the paged I/O mode, the read rate is higher than the write rate. The file access rates depend on the buffer size used in file access. For the file read, normally, as the buffer size increases, the performance improves to a certain point. The buffer size which provides a reasonably good performance for various configurations is 4 Kbytes which is same as the block size and the stripe size. Currently, user has no control over the block size and the stripe size. Further experiments are needed to study the effect of stripe sizes.

- Using the burst-mode I/O, the file access rates improve significantly. It is observed that in general the buffer size for burst I/O access should be as large as possible for the best performance.

- For the single compute node case, the position of the node in the Touchstone Delta mesh does not affect the file access times. This shows that the "inter-node" hops between the compute nodes are very small. This shows that there is no possible overhead in using the communication network for the I/O transactions.

- For the multiple nodes case, the data accessed depends on the modes of file access. Mode 0 should be used for reading the shared data, whereas modes 1 to 3 should be used for data distributions. The access throughput depends on the number of processors and it also depends on the total number of disks on which the data is stored. The performance increases initially as the number of processors increase then it remains steady. In general, for small processor grids, the bandwidth is limited on the computational node size, but as the number of processors is increased, it shifts to the I/O system. the point at which this shift occurs depends on the number of processors as well as on the number of disks.
- For the multinode configuration, the performance can be further improved by using the burst mode of operation. Using a large buffer size (2 MBytes) for mode 0, the peak performance of 23.83 MBytes/sec is observed. For the same mode, the peak write rate is about 7 MBytes/sec. For the remaining three file access mode, burst mode gives a better performance than the paged I/O mode.
- The choice between various modes (except mode 0) depends on the type of access pattern and data organization. Mode 2 should be used only when the size of data access can vary and cannot be determined in advance. Mode 3 should be used if access size is known in advance. Mode 1 should be the choice when order of access is not important because mode 1 does not improve global synchronization.

References

[ACC+90] Robert Alverson, David Callahan, Daniel Cummings, Brian Koblenz, Allan Porterfield, and Burton Smith. The Tera Computer System. In *1990 International Conference on Supercomputing*, 1990.

[ACD+91] Anant Agarwal, D. Chiaken, G. D'Souza, Kirk Johnson, and D. Kratz et. al. The MIT Alewife Machine: A large-scale distributed memory multiprocessor. In *Scalable Shared Memory Multiprocessors.* Kluwer Academic Publishers, 1991.

[DCF+86] W.J. Dally, A. Chien, S. Fiske, W. Howart, J. Keen, and M. Larivee. The J Machine: A Fine Grain concurrent Computer. *Information Processing 89, Proceedings of the IFIP Conference*, pages 1147–1153, August 1986.

[DJK$^+$92] Lenowski Daniel, Laudon James, Gharachorloo Kourosh, Weber Wolf-Dietrich, Gupta Anoop, Hennessy John, Horowitz Mark, and Lam Monica. The Stanford Dash Multiprocessor. *IEEE Computer*, March 1992.

[FHK$^+$90] Geoffrey Fox, Seema Hiranandani, Ken Kennedy, Uli Kremer, and Chau-Wen Tseng. Fortran D Language Specification. Technical Report Rice COMP TR90-141, Rice University, Houston, Texas, December 1990.

[For93] High Performance Fortran Forum. High Performance Fortran Language Specification Version 1.0. Technical Report CRPC-TR92225, Center for Research in Parallel Computing,Rice University, January 1993.

[FPD91a] James C. French, Terrence W. Pratt, and Mriganka Das. Performance Measurement of a Parallel Input/Output System for the Intel iPSC/2 Hypercube. *Proceedings of the 1991 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 178–187, 1991.

[FPD91b] James C. French, Terrence W. Pratt, and Mriganka Das. Performance Measurement of a Parallel I/O Subsystem for Hypercube Multicomputers. *Proceedings of the 1991 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 178–187, 1991.

[FPD91c] James C. French, Terrence W. Pratt, and Mriganka Das. Performance Measurement of a Parallel Input/Output System for the Intel iPSC/2 Hypercube. Technical Report IPC-TR-91-002, Institute for Parallel Computation, University of Virginia, 1991. Appeared in, Proceedings of the 1991 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems.

[Int91] Intel. Touchstone Delta System Description. Technical Report Intel Advanced Information, Intel Corporation, 1991.

[Int92] Intel. Paragon XP/S System Description. Technical Report Intel Advanced Information, Intel Corporation, 1992.

[Lio93] Lionel M. Ni and Philip K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Computer*, February 1993.

[M. 93] M. Barnett and R. Littlefield and D. G. Pyne and R. van de Geijn. Global Combine on Mesh Architectures with Wormhole Routing. *Proceedings of 7'th International Parallel Processing Symposium*, April 1993.

[nCU92] nCUBE. nCUBE 2 Systems: Technical Overview. Technical report, nCUBE Corporation, 1992.

[Nit92] Bill Nitzberg. Performance of the iPSC/860 Concurrent File System. Technical Report RND-92-020, NAS Systems Division, NASA Ames, December 1992.

[Rik92] Rik Littlefield. Tuning Communication. *Proceedings of the Delta Advanced User Training Class Notes*, pages 99–120, July 1992.

[Thi91] Thinking Machines Corp. CM-5 System Description. Technical report, Thinking Machines Corporation, 1991.