Performance of the CM-5 Scalable File System

Thomas T. Kwan*
National Center for Supercomputing Applications

Daniel A. Reed†
Department of Computer Science

University of Illinois
Urbana, Illinois 61801

Abstract

Assessing the performance and software interactions of emerging parallel input/output systems is a critical first step in input/output software tuning. Moreover, understanding the system response to well-understood, synthetic input/output patterns is itself a prelude to analysis of more complex application input/output patterns. We have conducted a series of experiments to measure the performance of the CM-5's new Scalable Disk Array (SDA) and Scalable Parallel File System (SFS) using the file system interfaces provided by the data parallel CM Fortran and message passing CMMD programming models. The results of these experiments suggest that the CM-5's parallel input/output system is an improvement over its predecessor, the CM-2 Data Vault. However, network bandwidth can be a bottleneck for the data reordering phase of input/output operations.

1 Introduction

Despite, and because of, the enormous increases in the computation performance of recently delivered massively parallel systems, input/output is now a major performance bottleneck for many applications. Moreover, although inexpensive, high-capacity disks now made it possible to construct large disk arrays with high peak performance, the software data management issues (e.g., data distribution across storage devices, data staging, and distributed caching) and their dependence on the hardware design remain largely unexplored.

Simply put, assessing the performance and software interactions of emerging parallel input/output systems is critical to understanding each system's response to input/output intensive applications. Detailed performance data is the fundamental basis for software design improvements, particularly for parallel input/output systems where our knowledge of spatial and temporal access patterns is so limited. This data takes two forms. First, one must understand the system response to well-understood, synthetic input/output patterns; this is a prelude to detailed characterization of the input/output access patterns in a wide range of parallel codes and a basis for understanding this data. Only with knowledge of system behavior and limitations, obtained with synthetic benchmarks, can one reliably interpret the more complex patterns in realistic codes.

Thinking Machines has recently begun shipping a new parallel file system for the CM-5 disk array that provides a wider variety of access methods than those on the older CM-2 Data Vault. A preliminary CM-5 performance study by LoVerso *et al* [11] reported the input/output data rate achievable via operating system calls. However, the CM-5 input/output library includes a rich set of file organization and access methods; these methods are those of most interest to application developers.

The goal of this paper is to measure the input/output performance of the CM-5 using its rich set of library input/output interfaces and to provide the low-level performance data needed to support a more extensive application characterization effort.[1] Specifically, we measure the performance of the CM-5 parallel file system's CM-Fortran (SIMD) and C/CMMD (MIMD) input/output interfaces. In addition, we explore the effects of machine size on input/output performance and the potential performance variations due to interaction among multiple, concurrent input/output streams. To-

*Supported in part by the National Science Foundation and the Advanced Research Projects Agency under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives.

†Supported in part by the National Science Foundation under grant NSF CDA87-22836, by the National Aeronautics and Space Administration under Contract NAG-1-613, and by the Advanced Research Projects Agency under Contracts DABT63-91-C-0004 and DABT63-93-C-0040.

[1] This characterization effort is the major focus of our current work and builds on our Pablo performance instrumentation software [12].

gether, these baseline performance results can be used to gauge the input/output performance of input/output intensive applications and to aid in the understanding of any captured application input/output traces.

The remainder of this paper is organized as follows. In §2, we briefly describe the input/output architecture of the CM-5 and its parallel file system interfaces. This is followed in §3–§5 by a description of our input/output benchmarks, the experimental data, and the implications of our findings, respectively. Finally, §6 concludes with a summary and directions for future research.

2 CM-5 Input/Output System

The Thinking Machines CM-5 input/output system relies on a combination of disk arrays, operating system support, and parallel input/output libraries to access files that are distributed across multiple secondary storage devices. Below, we briefly review the CM-5 architecture and the file system interfaces; see [11] for details on the operating system design.

2.1 Architecture

The CM-5 is a MIMD, distributed memory system composed of computation and input/output nodes that are interconnected by a fat-tree data network and a binary tree control network [10]. The higher bandwidth data network is responsible for data motion, whereas the lower bandwidth control network is responsible for various global operations (e.g., broadcast and reduction).

A computation node typically consists of a SPARC processor, four floating point vector units, four memory banks, and a network interface (NI) chip. These computation nodes are grouped to form a partition, the resource unit managed by the operating system and allocated to users.

Each disk storage input/output node consists of a SPARC processor with 8 megabytes of local memory, a network interface chip, 8 megabytes of disk buffer, four SCSI controllers, and eight, 1.2 gigabyte 3.5 inch disks. As an example, Figure 1 illustrates a small, 16-node fat tree, an input/output node with a disk array, and a computation node. The raw data transfer rate of each SCSI channel is ten megabytes/second; the raw read and write data rates of a disk are, respectively, 2 megabytes/second and 1.8 megabytes/second [11].

The CM-5's Scalable Disk Array (SDA) is the union of the small disk arrays on the CM-5 input/output nodes. The SDA stripes file data across all disks in units of sixteen bytes (i.e., each successive group of sixteen bytes is placed on a different disk using a modulo mapping). This sixteen byte increment was chosen because it is the same size as the payload of a data network packet. The Scalable File System (SFS) [11] logically sits atop the SDA and arbitrates input/output requests from applications. From a user perspective it is a Unix file system, albeit with parallel input/output interface extensions.

2.2 Scalable File System Interfaces

Because the CM-5 supports multiple programming models, files can be accessed via interfaces in each. In the message passing model, file access primitives must be compatible with the single program multiple data (SPMD) model of the CMMD communication library. For data parallel CM Fortran codes, the CM Fortran utility library supports data parallel input/output. Finally, CMOST operating system primitives provide low-level input/output. Below, we describe the input/output interfaces used in our performance measurement study — the CM Fortran utility library interface and the CMMD interface.

2.2.1 CM Fortran Interface

In the data parallel programming model, CM Fortran utility library routines can read and write arrays that are distributed across the CM-5 node memories. Because the array data distribution depends on the size of the CM-5 partition, the library supports three file formats, fixed machine size, generic, and serial order, that allow the application developer to balance data storage generality against performance.

- *Fixed Machine Size*: This file format is dependent on the CM-5 hardware configuration, in particular, the number of disks and the size of the partition. Arrays written to the SDA in this format can *only* be read by programs that execute in a partition of the same size and that use the same array shape and data distribution as the file writer.

- *Generic*: Arrays written in this file format are first reordered into a hardware-independent intermediate form before being written to the SDA. This intermediate form, known as New Mexico Order, is a block-block declustering of the array data. It enables programs executing in a partition of any size to read files written in this file type. However, the reader still must use the same array shape and distribution as the writer.

- *Serial Order*: This format is the most flexible. Like the generic format, data written in this format can be read by programs executing in partitions of any size. In addition, the serial order format allows the reader to read arrays using shapes and distributions different from that of the writer.

See [15, 14] for additional details on the CM Fortran parallel input/output interface.

The fixed machine size format is most appropriate when the data will be re-read by the same code on partitions of the same size (e.g., checkpoint files). The generic format allows a code

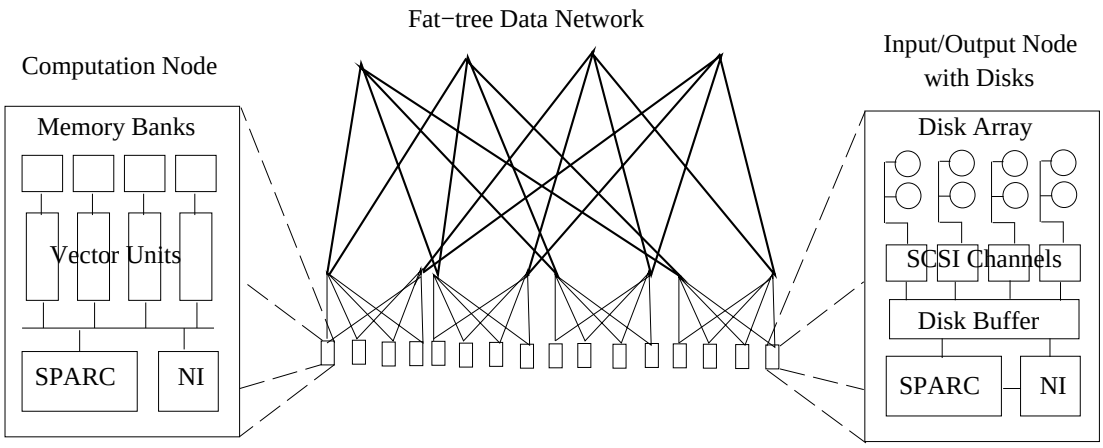


Figure 1: 16 Node CM-5 Partition

to re-read a file without the partition size restriction. Finally, the serial order format allows other codes and workstations to access the data.

2.2.2 CMMD Interface

Like the CM Fortran utility library, the CMMD message passing library supports both hardware-dependent and hardware-independent data storage formats. File stored in the CMMD hardware-dependent format are akin to CM Fortran fixed machine size files. In this case, the CMMD input/output interface requires users to read or write files in increments that are multiples of sixteen bytes times the number of disks in the system.

The CMMD library also provides four input/output modes for use with hardware-independent files. See [13] for additional details on the CMMD parallel input/output interface.

- *Local*: Each node maintains its own file descriptor and file pointer, and the nodes all operate independently.
- *Global Independent*: All the nodes must participate in the call to open the file. The open creates a single file descriptor that is shared by all the nodes. The nodes continue to maintain their own file pointers, and they can read or write the file independently.
- *Synchronous Broadcast*: In this global mode, all nodes must participate in all file operations. However, each file operation returns the same data to every node. Conceptually, file read operations are equivalent to a single node reading the file, then broadcasting the result to all the other nodes. File write operations are equivalent to a single node writing data to secondary storage.
- *Synchronous Sequential*: Like the synchronous broadcast mode, all the nodes must participate in all file operations. On file writes, data from the nodes is written in sequential order, starting with node zero (i.e., the processors logically write the data in order of node number). On file reads, data are distributed sequentially, with node zero receiving the first sequential portion.

3 Benchmark Methodology

The number of input/output models, the options for each, and the dependence on machine configuration and data distribution make it impossible to specify the CM-5's input/output performance with a single figure of merit. Moreover, the SFS software continues to evolve. To assess input/output performance for the common cases, we developed a set of synthetic input/output benchmarks, and measured SFS performance under a variety of loads. Below, we describe the hardware configuration used for our experiments and the synthetic benchmarks.

3.1 Experimental Configuration

All our experiments were conducted on the CM-5 at the National Center for Supercomputing Applications (NCSA). This CM-5 has 512 computation nodes, fifteen disk storage nodes, five partition managers, and two input/output control processors. NCSA's CM-5 SDA is configured with two file partitions and has a total of 120 disks. However, each file partition was configured to use only fifty-seven disks (fifty-six data disks plus one parity disk); the six unused disks are distributed among the disk storage nodes. The two file partitions use disjoint sets of SCSI channels and disks on each input/output node.

Our measurements used the CM Fortran compiler (Version 2.1 Beta 0.1 patch 1), the CMMD communication library (Version 3.1 Final), the C compiler (Version 2.1), and the CMOST operating system (Version 7.2 Rev 1).

3.2 File System Benchmarks

The plethora of possible file access modes, input/output request sizes, inter-request inter-

Job One (32 nodes)	Job Two (64 nodes)	Same Files?
read	read	yes
read	read	no
read	write	no
write	write	no

Table 1: Concurrent Input/Output Tests

vals, array data distributions, and partition sizes makes an exhaustive enumeration of all possible cases impractical. To quantify the performance of the most salient aspects of the CM-5's SDA and SFS, we constructed three sets of synthetic input/output benchmarks.

The first two benchmark sets exercised the synchronous input/output interfaces of CM Fortran and CMMD. Because of their regularity, these tests allowed us to measure the highest possible data transfer rates. The third set of benchmarks explored the interaction between multiple, concurrent input/output streams. Because, in practice, multiple parallel programs, each executing on a different CM-5 partition, will compete for access to the SDA and the SFS, this benchmark enabled us to test the file system's resilience to conflicting resource demands. Below, we describe each set of benchmarks in more detail.

1. The CM Fortran benchmarks access data on the SDA via the CM Fortran utility library. We measured the time taken to read and write one-dimensional arrays using, respectively, the fixed machine size, generic, and serial order file formats. The size of the arrays varied from 8 to 384 megabytes. To study the interaction of array data distributions and input/output performance, we also varied the partition size.

2. The message passing input/output benchmarks were written in C and accessed the SDA via the CMMD library. We measured the time to read and write file buffers of various sizes, using the synchronous broadcast mode, the synchronous sequential mode, and the hardware-dependent file operations. This set of benchmarks allowed us to compare the CM Fortran data parallel, synchronous input/output operations to those available under CMMD.

3. The multiple, concurrent input/output benchmarks measured the effective data transfer rates of the SFS when two jobs running in different partitions accessed the same and different files. Table 1 illustrates the four test cases used in this set of benchmarks. Each case was conducted twice, first with two CM Fortran programs and then with two C/CMMD programs.

We used elapsed time for all of our measurements. To measure CM Fortran input/output, we used a timer (`ITIME`) with one second resolution. CMMD codes were measured with a C timer (`get_time_of_day`) that had a one microsecond resolution. All the data arrays were double-word aligned. Unless otherwise noted, all benchmarks consisted solely of input/output operations (i.e., there is no computation between successive file system calls) and all experimental data are shown with ninety-five percent confidence intervals. Whenever feasible, we experimented with all the partition sizes regularly used on the CM-5 at NCSA.[2]

4 Benchmark Analysis

Generally, our results show that the CM-5's Scalable File System (SFS) provides 32–80 percent of the theoretical peak transfer rate from the Scalable Disk Array (SDA). Depending on the file access pattern, request size, and partition size, either the SDA or the interconnection network is the performance bottleneck.

4.1 CM Fortran Benchmarks

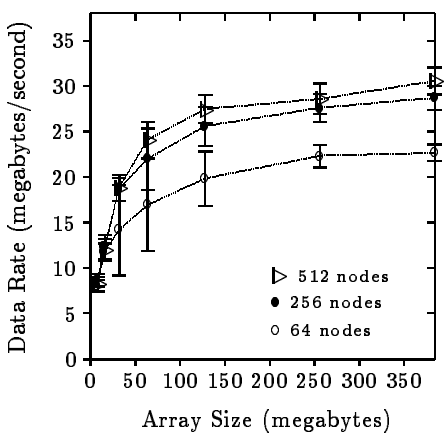
Our experiments showed that CM Fortran input/output performance depends strongly on the file format used. As with single processor Unix systems, the performance of file reads and writes differs substantially. For sequential reads, SFS exploits caching and prefetching to hide most of the hardware latency.

4.1.1 File Writes

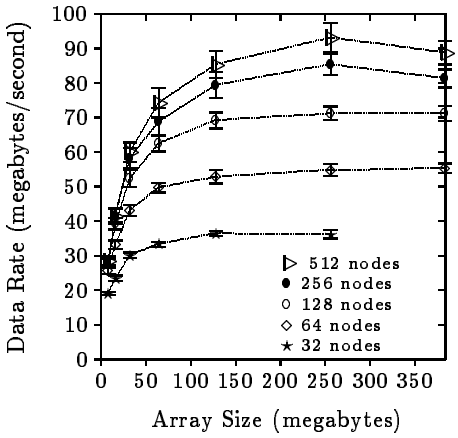
Figures 2a, c, and e show the results of the CM Fortran benchmarks for file write operations using three different partition sizes. File writes that exploit the fixed machine size file format are fastest; for sixty-four nodes, this input/output rate is roughly 32 megabytes/second. With generic or serial order format, the data rate drops to about 22 megabytes/second because the file system must reorder the data before writing it to secondary storage. Using the raw data transfer rates of §2.1, we can estimate the peak write data rate as 1.8 megabytes/second per disk $\times$ 56 disks per file partition, or 100.8 megabytes/second. Thus, the observed peak rate for the fixed machine size format is about 32/100.8, or 30 percent of the theoretical peak disk transfer rate.

In Figure 2, the data rates for the generic and serial order writes are the same, within measurement error. For small transfer sizes (i.e., small arrays), the software overhead to retrieve file block information from the file system when the operation is initiated and then to update the file state at the end of the input/output operation dominates the access cost [11]. The data rates do increase with the partition size. With a larger partition, the data in the array is distributed across more nodes, and more network

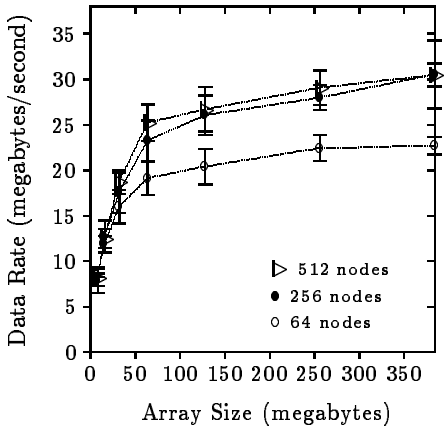
[2] For additional details on the experiments and their results, see [8].



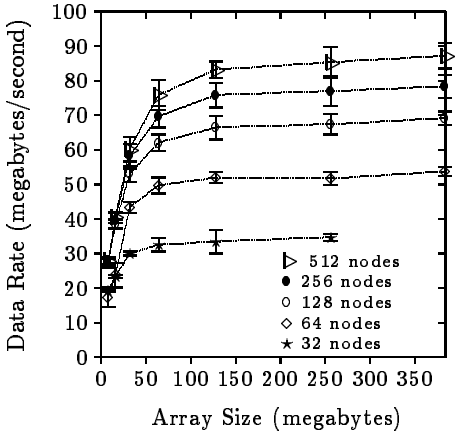
(a) Generic Order File Writes



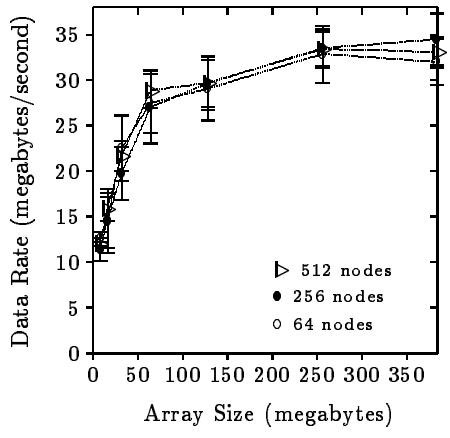
(b) Generic Order File Reads



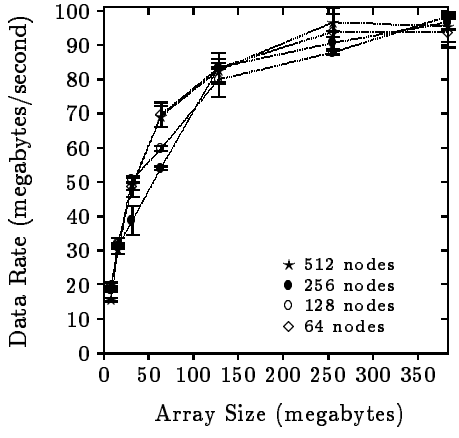
(c) Serial Order File Writes



(d) Serial Order File Reads



(e) Fixed Machine Size File Writes



(f) Fixed Machine Size File Reads

Figure 2: CM Fortran Input/Output Operations

bandwidth is available for data transmission from the computation nodes to the disk storage nodes. Interestingly, the data rate for the fixed machine size format does not increase with an increase in the partition size. Hence, we believe that either the input/output control processors or the disk storage nodes are the bottlenecks for the write operations.

4.1.2 File Reads

Figures 2b, d, and f show the data rates when reading files. For generic order reads, the input data rate increases by about 20 megabytes/second as the partition size of the reading application increases from thirty-two to sixty-four nodes. Thereafter, the data rate increases by roughly 10 megabytes/second with doubling partition size. Again, larger partitions have higher aggregate network bandwidth for data transmission and receipt, and for data reordering. Figure 2b shows that the highest data rate is about 90 megabytes/second. The peak read transfer rate from the NCSA SDA is 56 data disks $\times$ 2 megabytes/second per disk, or 112 megabytes/second. Hence, the observed data transfer rate is about 90/112, or 80 percent of the theoretical peak.

Figure 2d is the complement to Figure 2b; it shows the data rates for reading files in serial order. As with generic files, the data rates increase with the partition size of the reader, approaching a peak rate of 91 megabytes/second.

Finally, Figure 2f shows the data rates of the fixed machine size format. The sustained data rate for the sixty-four node partition is roughly 94 megabytes/second. This rate is comparable to the highest 512 node data rate in Figures 2b and 2d; no data reordering is required for the fixed machine size format. Comparing Figure 2f with Figure 2b, where the sixty-four node partition can only sustain 55 megabytes/second for generic order reads, the penalty for using the generic file format is about forty percent. As before, this degradation is due to data reordering. From Figure 2f, the highest read data rate for the 512 node partition is 95 megabytes/second. This is about 85 percent of the disk transfer rate.

4.2 CMMD Benchmarks

Although the CMMD synchronous sequential file operations and hardware-dependent file operations roughly correspond to the CM Fortran generic file operations and fixed machine size file operations, respectively, the synchronous broadcast operations have no CM Fortran analog. Our experimental data show that the performance of the CMMD synchronous broadcast operations is limited by the CM-5's control network. Below, we describe in more detail the results of the individual CMMD input/output benchmarks and, where relevant, compare the results to those of §4.1.

4.2.1 Synchronous Broadcast

Figures 3a–b show the results for synchronous broadcast reads. Performance declines with increasing partition size — the file is conceptually read by one node, and the data is broadcast to all the other nodes. In Figure 3b, the maximum data rate is less than 1 megabyte/second for modest size arrays and quickly approaches an asymptote. Because the CMMD synchronous broadcast mode uses both the data and control networks to coordinate data transmission and synchronization of the processors [2], and because the broadcast function of the CM-5's control network has a bandwidth of approximately 800 kilobytes/second [9], this limitation is a major cause for the asymptotes of Figure 3b.

Figure 3a shows the data rate for synchronous broadcast writes. In contrast to synchronous broadcast reads, where the data rate quickly approaches an asymptote, the performance of broadcast writes continues to increase with array size. Broadcast writes do not use the control network, and the achievable data rate is not limited by the data transfer rate of the control network.

4.2.2 Synchronous Sequential

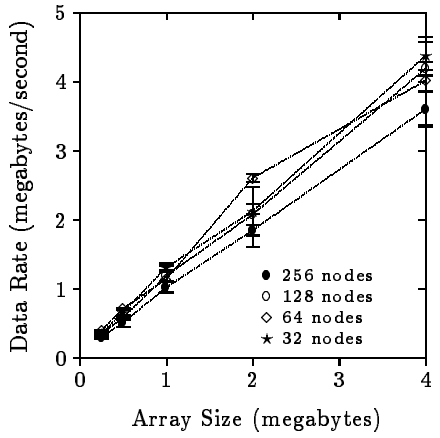
Figure 3c shows the performance of synchronous sequential writes. Their performance increases with partition size until it approaches that of the hardware-dependent mode, shown in Figure 3e.[3] Comparing Figure 3 with Figure 2 shows that the results with CMMD are similar to those obtained with CM Fortran. The CMMD synchronous sequential writes correspond roughly to the CM Fortran generic file operations. Moreover, the analysis of the CM Fortran results also is applicable. The data rates for the synchronous sequential writes are limited by the network bandwidth needed to reorder the data.

The complement to Figure 3c, Figure 3d, shows the data rates when reading files with the synchronous sequential mode. As one would expect, increasing the partition size increases the observed data rate because the available interconnection network bandwidth increases with partition size. Comparing Figure 3d to Figure 2b reveals that the read performance of the CMMD synchronous sequential mode is similar to, but slightly lower, than that of the generic mode in CM Fortran.

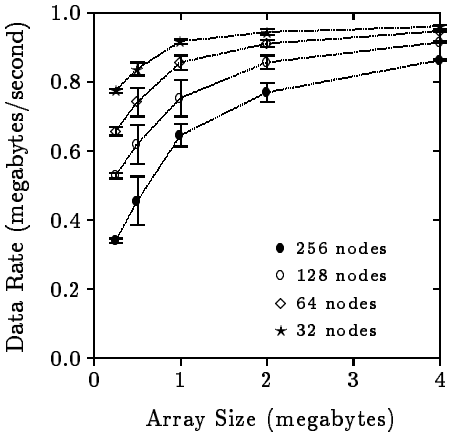
4.2.3 Hardware Dependent

Finally, Figures 3e–f show the data rates for the hardware-dependent write and read operations. Recall that these read operations do not require data reordering, but instead read the data in an order determined by the partition size and number of disks. The data rates are noticeably higher than for the synchronous sequential mode

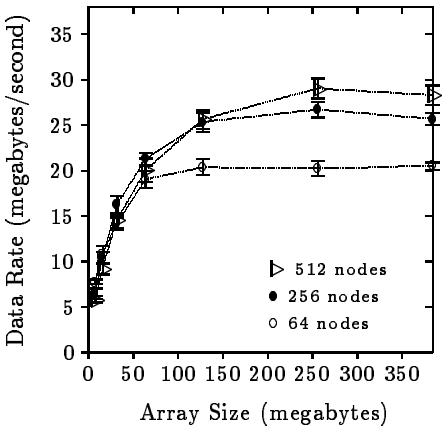
[3] Recall that in the synchronous sequential mode, each node writes a disjoint portion of the file. Hardware-dependent mode writes data in units that depend on the partition size.



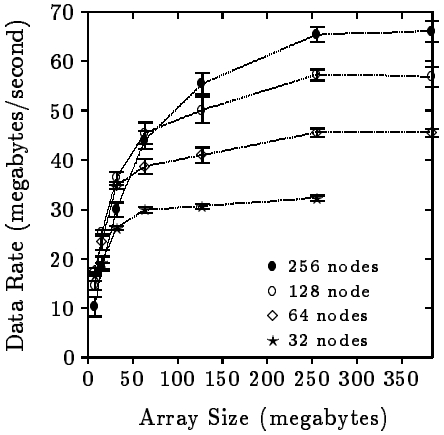
(a) Synchronous Broadcast File Writes



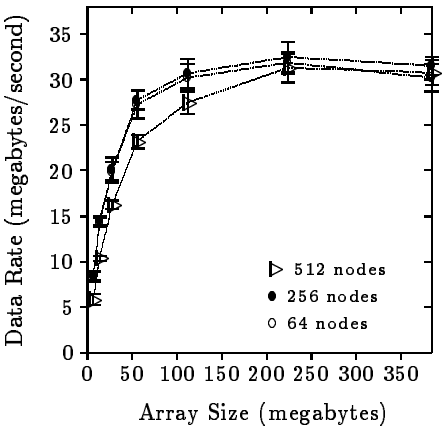
(b) Synchronous Broadcast File Reads



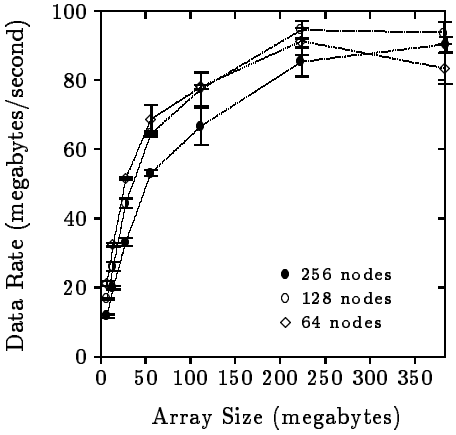
(c) Synchronous Sequential File Writes



(d) Synchronous Sequential File Reads



(e) Hardware Dependent File Writes



(f) Hardware Dependent File Reads

Figure 3: CMMD Input/Output Operations

and are comparable to those for the fixed machine size mode of CM Fortran.

4.3 Application Interactions

Recall that the goal of this set of benchmarks was to assess the effects of program interactions and resource contention. Figures 4–5 show the input/output performance of a single reader, a single writer, the interaction between two readers, and the interaction between a reader and a writer.[4]

For the CM Fortran access modes, when a second reader in another partition reads the same file, SFS detects the duplicate references and satisfies the second request stream from the file cache. Hence, the read data rates for two concurrent readers of the same files are similar to that for a single reader. However, when a program in another partition writes a different file to the SDA, Figure 4a shows that the read data rates decline by roughly thirty percent. In this case, the two programs compete for file system buffer space and disk access. Figure 4b shows the adverse effect on the writer when there is a reader in another partition. Although the degradation is smaller than for the reader, it is still substantial, over twenty percent in this case.

Finally, Figure 5 shows the effects of reader/writer interaction for CMMD file access. The interaction degrades the reader's performance, albeit not as severely as for CM Fortran.

5 Discussion

It is instructive to compare the performance data of §4 with that obtained by LoVerso *et al* [11], who recently reported the results of a series of operating system input/output experiments on the CM-5. A comparison shows that the performance of the read operations is consistent, with data rates near 90–100 megabytes/second.[5] However, the performance of write operations in our experiments was lower by a factor of two. Possible causes for the differential include the overhead of the input/output library implementation, the need to preallocate disk blocks to achieve higher performance, disk fragmentation, and the way the file system was partitioned at NCSA.

Comparing the CM-5 Scalable Disk Array (SDA) with Thinking Machines' previous generation disk array, the DataVault, reveals that the CM-5's SFS and SDA provide users with greater flexibility, allowing them to write files in a generic format that can be read by partition of any size. This was not possible on the CM-2 unless users wrote files in the serial order format. On the CM-2, this limited the input/output performance to about 4.3 megabytes/second [5]. However, to achieve the highest possible input/output performance on the CM-5, users must still use a configuration-dependent file format.

Additional insights into the parallel input/output performance of the CM-5 are possible by comparing our results with those obtained from other large parallel systems, notably the Intel Touchstone Delta. Table 2 compares the input/output configuration and the achievable data rates of the CM-5 at NCSA with that of the Touchstone Delta at Caltech.[6] Both machines have 512 processing nodes and roughly same number of disks. However, the data rates for the read and write operations of the CM-5 are nearly any order of magnitude faster than the Delta. Intel is redressing this differential with the Parallel File System (PFS) on the Intel Paragon XP/S [7]. PFS supports many of the same features as SFS, including disk striping, parity for fault tolerance, and distributed file caching.

Recently, Bordawekar *et al* [4] proposed a two-phase access strategy to improve input/output performance on the Intel Delta and other parallel file systems. The first phase reads data from the disks with the data distribution and stripe size that best matches the distribution of the data stored on disk; the second phase rearranges the data among the processing nodes to conform to the distribution required by the application program. This two-phase access strategy is similar to the CM-5 implementation. For the CM-5, our data suggest that the network bandwidth often is the bottleneck in the second phase of the algorithm (i.e. data reordering).

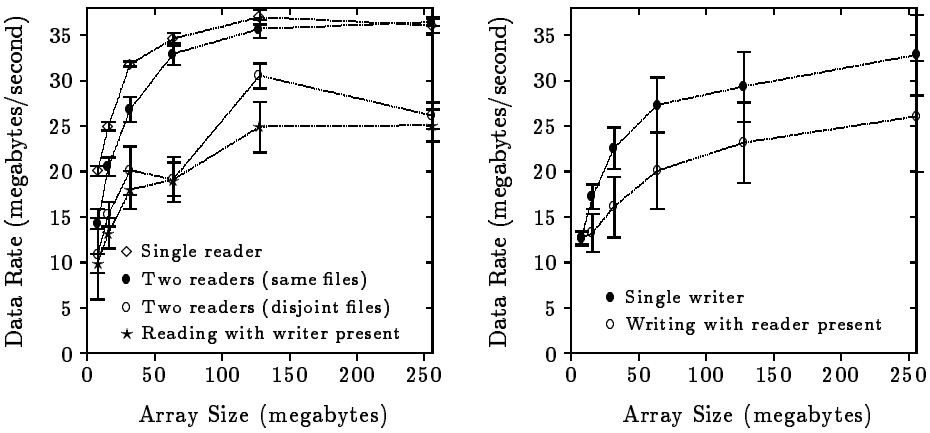
In general, the data reordering operation is expensive. Potentially, all the data in each computation node must be re-distributed to other computation nodes to form a canonical layout (New Mexico Order in the case of the CM-5). Minimizing the cost of data reordering is possible via either hardware or software. A hardware solution mandates either sufficient interconnection network bandwidth for rapid reordering of data from its canonical storage format to the desired memory distribution or the design of dedicated data turning networks (e.g., as found in the Goodyear/NASA MPP [1]).

Alternatively, rather than enforcing a fixed set of storage formats, exporting control of secondary storage data distributions to the application developer would obviate the need for complex data turning algorithms and would allow users to manage input/output data distributions just as they do in high-performance Fortran [6]. Similarly, user control of file cache allocations and prefetch algorithms would allow them to balance the needs of multiple file request patterns and ameliorate the cache thrashing of Figures 4–5.

[4] Attempts to measure the degradation due to multiple writers failed — the disk controllers and input/output control processors hung each time we ran this experiment. Thinking Machines is currently testing a software patch to be included in the CMOST 7.3 final release.

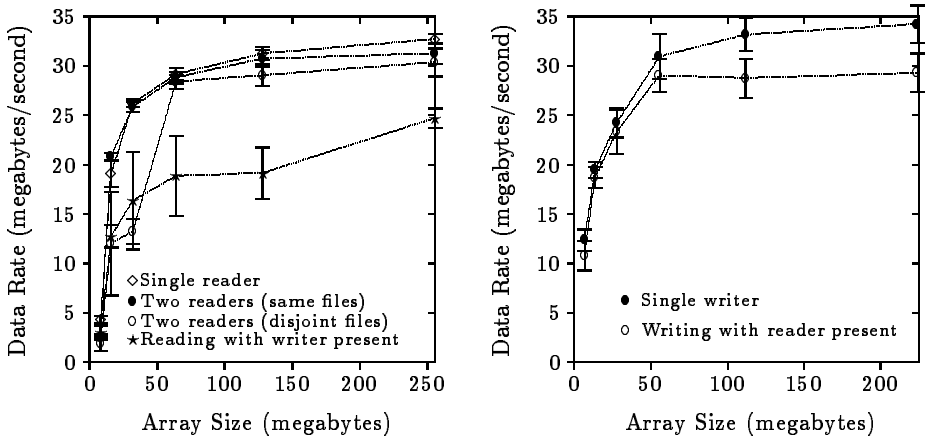
[5] Although the hardware configurations used by [11] differ from those in the our study, they provide enough data to bound the expected data rates for the NCSA CM-5 configuration.

[6] The data for the Delta are from [3].



(a) Effect on the read data rate (b) Effect on the write data rate

Figure 4: CM Fortran Reader/Writer Interactions



(a) Effect on the read data rate (b) Effect on the write data rate

Figure 5: CMMD Reader/Writer Interactions

In summary, possible improvements to the CM-5 SFS include lowering the latency required for input/output of smaller arrays by moving part of the file system code into the operating system kernel of the computation nodes [11], and avoiding the use of the control network when handling large data requests for the CMMD synchronous read operation. More generally, we believe that SFS and other parallel file systems should allow users greater control over data placement and caching policies.

6 Summary

Enormous increases in the computation performance of recently delivered massively parallel systems have exposed input/output as the major performance bottleneck for many applications. Detailed performance data is the fundamental basis for software design improvements, particularly for parallel input/output systems where our knowledge of spatial and temporal access patterns is so limited. The first step in acquiring performance data is understanding the system response to well-understood, synthetic input/output patterns. This is a prelude to detailed characterization of the input/output access patterns in a wide range of parallel codes and a basis for understanding this data.

We have conducted a series of performance experiments on the CM-5's Scalable Disk Array and Scalable Parallel File System using the file system interfaces provided by the data parallel CM Fortran and message passing CMMD programming models. We found the maximum read and write data rates to be roughly 95 and 35 megabytes/second, respectively. The results of these experiments suggest that the CM-5's parallel input/output system is an improvement

Parameter	TMC CM-5	Intel Delta
Number of processing nodes	512	512
Number of disk I/O nodes	15	32
Number of disks (per file partition)	57	64
Number of I/O control processors	2	N.A.
I/O disk buffer/cache memory (Mbytes)	8	8
I/O node processor	SPARC	Intel 80386
File System	Scalable File System	Concurrent File System
Maximum write data rate (Mbytes/second)	35	3.7
Maximum read data rate (Mbytes/second)	95	23.8

Table 2: Parallel System Comparison

over its predecessor. However, limited interconnection network bandwidth, contention for storage node and disks, and interaction between input/output jobs can potentially degrade performance.

Open input/output resource management questions for the CM-5 and other massively parallel systems include the efficacy of the data prefetching and caching policies for irregular access patterns, integration of tertiary and secondary storage management, support for real-time data streams, and development of application interfaces for controlling data placement, caching, and prefetching. Continued performance measurement, together with capture and study of application resource demands and system software responses will be needed to resolve the input/output bottleneck.

Acknowledgments

Our thanks to Curt Canada of NCSA for arranging dedicated time on the CM-5, to Mike Best and Mike D'Mello of Thinking Machines for clarifying technical details, and to Charlie Catlett of NCSA, without whom this work would not have been possible.

References

[1] Batcher, K. E. Design of a Massively Parallel Processor. *IEEE Transactions on Computers C-29*, 9 (Sept. 1980), 836–842.

[2] Best, M. Personal Communication.

[3] Bordawekar, R., Choudhary, A., and del Rosario, J. M. An Experimental Performance Evaluation of Touchstone Delta Concurrent File System. In *Proceedings of the International Conference on Supercomputing* (July 1993), pp. 367–377.

[4] Bordawekar, R., del Rosario, J. M., and Choudhary, A. Design and Evaluation of Primitives for Parallel I/O. In *Supercomputing 1993* (Nov 1993), pp. 452–461.

[5] Fineberg, S. A. Implementing the NHT-1 Application I/O Benchmark. In *Workshop on I/O in Parallel Computer Systems* (April 1993).

[6] HPFF. High-Performance Fortran Language Specfication, version 1.0. Tech. rep., High Performance Fortran Forum, May 1993.

[7] Intel Supercomputer Systems Division. Paragon User's Guide, Oct 1993.

[8] Kwan, T. T. Performance Evaluation of the CM-5. Master's thesis, University of Illinois at Urbana–Champaign, Department of Computer Science, April 1994.

[9] Kwan, T. T., Totty, B. K., and Reed, D. A. Communication and Computation Performance of the CM-5. In *Supercomputing 1993* (Nov 1993), pp. 192–201.

[10] Leiserson, C. E., et al. The Network Architecture of the Connection Machine CM-5. In *Proceedings of Parallel Algorithms and Architectures Symposium* (May 1992), pp. 272–285.

[11] LoVerso, S. J., Isman, M., Nanopoulos, A., Nesheim, W., Milne, E. D., and Wheeler, R. *sfs*: A Parallel File System for the CM-5. In *Proceedings of the 1993 Summer Usenix Conference* (1993), pp. 291–305.

[12] Reed, D. A., Aydt, R. A., Noe, R. J., Roth, P. C., Shields, K. A., Schwartz, B. W., and Tavera, L. F. Scalable Performance Analysis: The Pablo Performance Analysis Environment. In *Proceedings of the Scalable Parallel Libraries Conference* (1993), A. Skjellum, Ed., IEEE Computer Society.

[13] Thinking Machines Corporation. CMMD Reference Manual Version, 3.0 Beta, Dec 1992.

[14] Thinking Machines Corporation. CM-5 I/O System Programming Guide, Version 7.2, September 1993.

[15] Thinking Machines Corporation. CM Fortran Utility Library Reference Manual, Version 2.0 Beta, January 1993.