RC 20049 (5/2/95)
Computer Science/Mathematics

IBM Research Report

Performance Evaluation of a Parallel I/O Architecture

Sandra Johnson Baylor, Caroline Benveniste, Yarsun Hsu

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties).

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

Research Report

Performance Evaluation of a Parallel I/O Architecture

Sandra Johnson Baylor
Caroline Benveniste
Yarsun Hsu

IBM Research Division
T. J. Watson Research Center
Yorktown Heights, NY 10598

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents and will be distributed outside of IBM up to one year after the date indicated at the top of this page. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties).

I B M **Research Division**
Almaden • T.J. Watson • Tokyo • Zurich

Performance Evaluation of a Parallel I/O Architecture

Sandra Johnson Baylor
Caroline Benveniste
Yarsun Hsu

IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10594

Abstract

In recent years, more emphasis has been placed on the design and performance evaluation of processor and memory architectures than on I/O subsystems in parallel processors. As a result, a large gap exists between I/O access and processor and memory access times in these machines, causing I/O to become a prohibitive bottleneck. Presented are the results of a study conducted to evaluate the performance of parallel I/O on a massively parallel processor (MPP), consisting of a number of compute nodes and I/O nodes. A general MPP architecture is assumed. The number of nodes evaluated ranges from 16 to 128 with a variation in the ratio of compute nodes to I/O nodes. We calculate the network traversal and total processing times for I/O reads and writes while varying the I/O request rate, the request rate for non-I/O message traffic, and the request size. We also study the impact of non-I/O messages on the performance of I/O traffic and, conversely, how the I/O traffic affects the processor-processor messages. For I/O-only traffic, the results show performance advantages for I/O processing times as the number of I/O nodes in the system is increased, and as the I/O request rate increases. The system is scalable for the I/O loads considered in this study. Adding non-I/O traffic results in performance degradations ranging from 2% to up to an order or magnitude as the non-I/O message traffic increases to high loads of up to 4464 requests/sec. The scalability of this system may be limited by the I/O node saturation or considerable network contention.

I. Introduction

The advent of massively parallel processors (MPPs) has resulted in the potential for revolutionary performance in computers. While the design of MPPs was initially motivated by high performance requirements of scientific applications such as the *grand challenge* problems, MPPs now have become more general purpose, appealing to an array of scientific and commercial applications ranging from seismic processing, global climate modelling and satellite/earth station processing to multimedia applications for education and entertainment technology. These applications require high-bandwidth, low-latency access to very large amounts of data (gigabytes to petabytes).

Much effort has been expended on the design and performance evaluation of processors and memory for MPPs; however, relatively little work has been done on the I/O subsystems of these machines. In addition, while direct access storage devices have improved in recent years, the rate of improvement is far less than that for processors [1]. This has resulted in large gaps between I/O access and processor and memory access times. Since a balance has to exist between these subsystems to achieve optimum performance, I/O is widely viewed as a prohibitive bottleneck in MPPs. Architecting an I/O subsystem for MPPs that addresses this bottleneck will cause substantial gains in MPP performance and make it possible for programmers to write new, creative, and innovative parallel applications for these machines.

This paper presents the results of a study conducted to evaluate the performance of some architectural alternatives for the parallel I/O subsystem of an MPP that can reduce the access bottleneck. (The MPP architecture used as a foundation for this study is described in the next section). The average processing times for I/O reads and writes are calculated as the system size is varied from 16 to 128 nodes and as the compute node to I/O node ratio is varied from 1 to 4. Several request rates are used for both I/O and non-I/O message traffic to determine the impact of I/O traffic on other types of network traffic under various loads as well as the impact of I/O traffic on processor-processor traffic. The availability of accurate parallel I/O workload characteristics for high performance machines is limited. Therefore, stochastically generated traces are used to evaluate the performance of the parallel I/O subsystem using PIOS, a trace driven parallel I/O simulator [2]. A sensitivity analysis is done to determine the impact of various factors on machine performance.

Shown in Figure 1 is a logical view of a parallel I/O subsystem for a MPP. A distinction is made between two types of I/O; *external* and *internal*. When data is transferred between some outside source and the machine, it is denoted as *external* I/O. Transferring data between the machine's secondary storage system and a frame buffer for a graphics display or between an archival storage system via some high speed link such as HIPPI are examples of external I/O. A gateway node, as shown in Figure 1, may be used in external I/O transfers. *Internal* I/O is defined as the transfer of data between the machine's own secondary storage system and its memory or processor for use by some application. Examples of internal I/O include physical simulations that output their entire state every few timesteps, the shuffling of data between memory and disks that occurs as a result of applications using large data sets, and the accessing of video objects from disks by a video-on-demand application for data decompression. This paper focuses on the performance of internal parallel I/O architectures.

The compute nodes, interconnection network, and I/O nodes form the internal parallel I/O subsystem for the MPP shown in Figure 1. The compute nodes are used primarily for computation and therefore no I/O data is located on the disks that may be included on the node (these disks may be used for paging space). The I/O data is distributed across the I/O nodes and is accessed in parallel by the compute nodes through the high bandwidth, low latency network. Several commercial machines exhibit this

type of high level architecture including the IBM SP1 and SP2 [3], the Thinking Machines CM5 [4], the Intel Paragon [5], the nCUBE hypercubes [6], and the Tera [7].

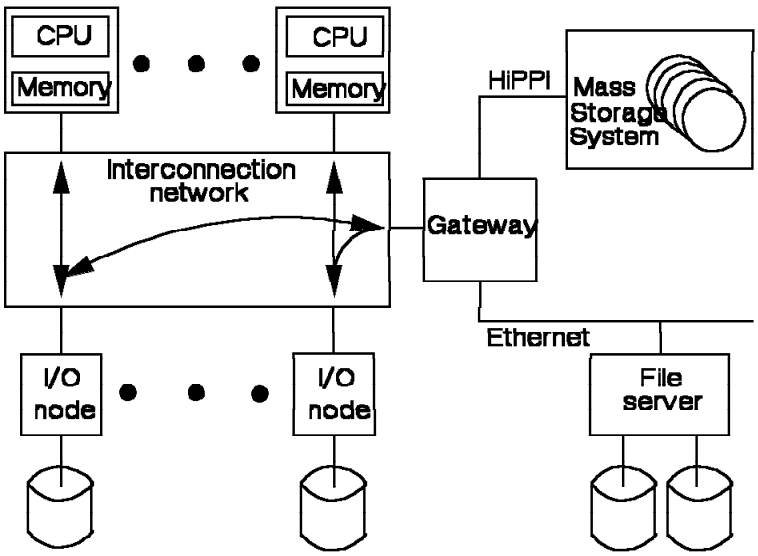

Figure 1. MPP I/O Architecture.

In the last few years some initial work has been done on the design and performance evaluation of parallel I/O subsystems. Reddy and Banerjee [8] studied I/O embedding in hypercube architectures and concentrated on I/O node placement in a hypercube. Their results were for systems with a relatively small number of processors (up to 16). Livingston and Stout [9] evaluated methods for efficiently distributing resources in hypercubes, concentrating on developing networks with a constant degree access. Their work did not focus on I/O but considered resources in general. Two studies were conducted by Baylor, *et. al.*, [10, 11]. The first study evaluated the parallel I/O performance of the Vulcan MPP [12]. This work focused on I/O node placement, the effect of varying the I/O read/write ratios in the system, scalability, and other issues for systems of up to 512 nodes. The second study included a detailed evaluation of the relationship between I/O and non-I/O message traffic traversing the network and the performance sensitivity to the request size for parallel I/O architectures. The results of both of these studies included the effects of I/O node blocking (the I/O node fails to process any subsequent requests until the processing of a write miss has completed at the I/O node) [2]. More interesting and realistic results would either not include I/O node blocking or would include I/O node blocking only when the I/O node can no longer buffer incoming requests. The results presented in this paper differs from the results of these two studies in the following ways: 1) the new results do not include I/O node blocking and 2) the new results have improved disk specifications and switch cycle times, resulting in a more realistic quantification of parallel I/O performance.

In the next section we present an overview of the MPP architecture assumed. In section III there is a discussion of the simulation methodology used to evaluate the parallel I/O architecture and in sections IV and V we present the results and conclusions, respectively.

II. MPP Architecture

The MPP architecture assumed includes a number of compute nodes and I/O nodes interconnected by a high performance switching network. The processing engines of the compute nodes and the I/O nodes are assumed to be commercially available 32-bit microprocessors. The compute node is assumed to have a disk capacity that is smaller than the disk capacity of the I/O node. The compute node is also assumed to have a reasonable amount of local memory (*i.e.*, up to 512 megabytes), and disk capacity (*i.e.*, up to 4 gigabytes). It is assumed that the I/O node includes a larger number of high capacity disks that may be used for I/O. This I/O node may be used as a component in the internal parallel I/O subsystem, as a server, or as a gateway node (see Figure 1). The I/O node is assumed to have larger local memory (up to 2 gigabytes) and disk capacity (up to 8 gigabytes). The total number of nodes in the systems we simulated ranged from 16 to 128. These system sizes correspond to total disk storage of 16 gigabytes to 1 terabyte and peak performance ranging from 4 to 34 GFLOPS for a 66.7 MHz clock.

The interconnection network assumed is a clustered multistage omega network composed of 4x4 bi-directional switches [12]. This packet-switched network is assumed to have a 150 megabyte/sec link bandwidth, and a less than 40 microsecond latency for application to application transfers (including hardware and software overhead). A microprocessor-based network adapter is used as a communication coprocessor, packetizing and depacketizing messages from the nodes. This frees the nodes to do computational tasks in parallel with message transfers. The adapter is assumed to have a bandwidth of at least 85 megabytes/sec and to provide DMA access, error detection and correction, and reliability. Messages are broken into packets, and the packets traverse the network as a unit. The packet size in this system is variable, but in these simulations we used a packet size of 512 bytes.

Presented in Figure 2 is diagram of the switch, which includes a central queue, bypass path and input and output ports. A packet traversing the switch may take either the bypass path or the central queue. If the packet′s destination output port is free and no other packets are queued for that port, it will take the bypass path. Otherwise, the packet will arbitrate, using an LRU policy, for placement in the central queue. The switch cycle time is 13.3 ns. Information on a switch similar to the one used in this study can be found in [13].

The MPP assumed can be used to exploit both functional parallelism, as exemplified by the MIMD paradigm and the data parallelism that is prevalent in the SPMD model, using its internal parallel I/O subsystem. Data can be accessed in parallel using a parallel file system such as the IBM Vesta Parallel File System [14, 15] or the Intel CFS [16]. This internal I/O subsystem may be one level of a hierarchical storage system. For example, the MPP node may be used to transfer data between an archival storage system and the MPP internal parallel I/O subsystem (import and export).

III. Simulation Methodology

A two-step process is used to model the MPP parallel I/O subsystem. First, a queueing model was developed to evaluate the performance of the I/O node. The performance of this node is then evaluated using RESQME [17], a simulator developed at the IBM T.J. Watson Research Center. A number of request sizes, disk cache miss ratios, read probabilities, and disks are simulated in this study. The general behavior of the results for all of these variables are similar. Therefore, only a subset of the values for these variables are presented. For example, the results of two request sizes, 4K and 32K bytes, are presented. Also, the I/O request rate is varied from 5 to over 850 requests/sec. The disk cache miss ratio is assumed to be 0.3, the I/O read (write) proba-

bility remains constant at 0.67 (0.33), and six disks are simulated per node. The average I/O read and write processing times are obtained from this model. In the second step, these processing times are used as inputs to PIOS, a parallel I/O simulator [2]. PIOS is an integration of a network simulator and the I/O node model. The trace driven simulator models the traversal of packets through the network and the processing of I/O requests in detail.

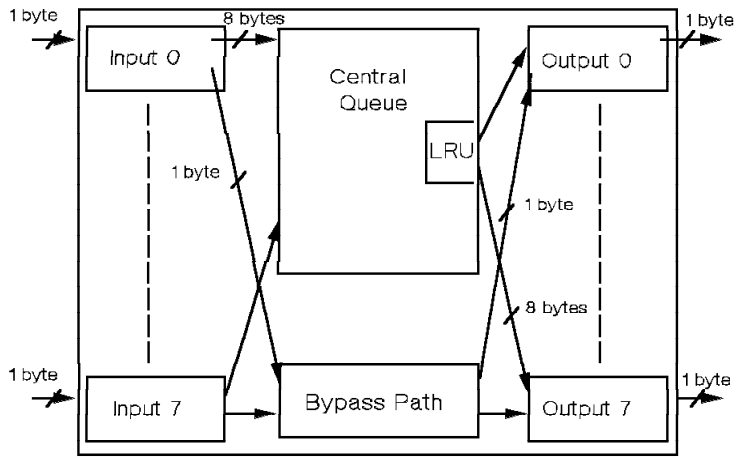

Figure 2. The Network Switch.

There has been one recent study conducted to obtain I/O workload characteristics for parallel systems [18]. This study obtained workload characteristics from a production-based Intel iPSC/860 machine with applications that used the Concurrent File System [16]. However, workload characterization is tied to the architecture of the machine on which the study is conducted, and the architecture of the parallel I/O subsystem of the Intel machine differs from the architecture of the subsystem we modeled in this study. Since no other studies provided us with I/O workloads we felt were representative of the system we were modeling, we decided to drive PIOS with stochastic traces. For each compute node, a trace is generated containing I/O requests and non-I/O messages to destinations selected by the trace generator. In this study the trace generator was programmed so that for each source, each destination had an equal probability of being selected over the course of the run. Two different message types were specified in the generator, the I/O requests targeted for I/O nodes and non-I/O messages targeted for compute nodes. A request destined for a compute node (either a read response or a non-I/O message) is assumed to be consumed immediately upon receipt. In reality, this may not occur every time a request reaches its compute node destination. Therefore, the results presented do not include any queueing effects that may occur if the requests are blocked at the compute node. PIOS is written to immediately process all I/O requests reaching the I/O node with one exception. If the buffers at the I/O node are full, then the I/O node cannot process any further requests until buffer space is available. For the results simulated in this study, the I/O node buffer space is never full, so no I/O node blocking occurs.

The system size varies from 16 to 128 nodes. Illustrated in Figures 3, 4, and 5 are the interconnection topologies used for the system sizes evaluated in this study. As shown in Figure 5, an extra stage of switching is used for the 128-node system. Although more than one path may exist between a source-destination pair, in our simulator we use static routing, and therefore do not route packets through multiple paths. The router we used chooses routes for the system so that static-load balancing takes

place. We chose to use static routing since we believe that no performance gains are obtained by allowing packets to use multiple paths between source-destination pairs.

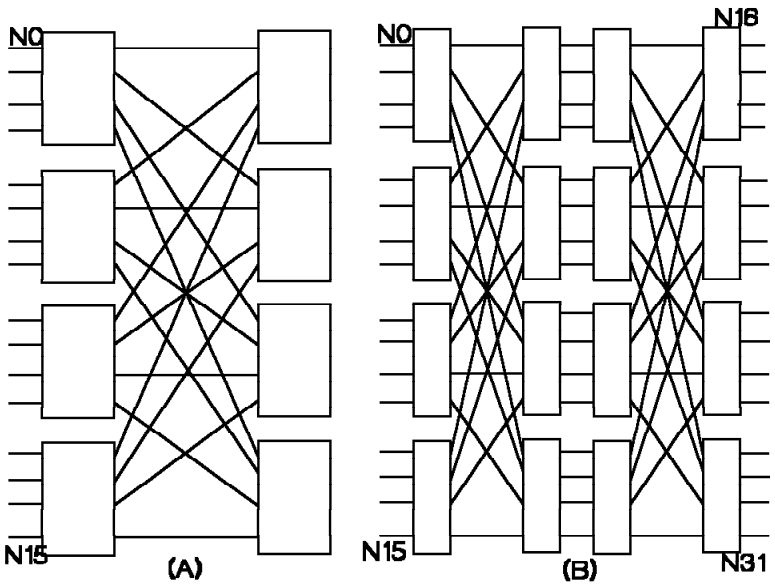

Figure 3. Interconnection Topologies for the 16-node (A) and 32-node (B) MPP.

In this study, half the total number of nodes used are always compute nodes and the ratio of compute nodes to I/O nodes is varied from 1 to 4. For example, in a 16-node system, the number of compute nodes remains constant at 8 and the number of I/O nodes is varied from 2 to 8. The I/O nodes are evenly distributed among the total number of nodes in the system. In cases where the number of compute nodes and I/O nodes do not add up to the total number of nodes in the system, the remaining nodes are assumed to be idle. This is done to evaluate the impact of increasing I/O parallelism on system performance. A number of request sizes, disk cache miss ratios, read probabilities, and disks are simulated in this study. The general behavior of the results for all of these variables are similar. Therefore, only a subset of the values for these variables are presented. For example, the results of two request sizes (4K and 32K bytes), one disk cache miss ratio (0.3), and one I/O read probability (0.67) are presented. Also, the I/O request rate is varied from 5 to over 850 requests/sec and six disks are simulated per node. The read request size is 200 bytes, the non-I/O message sizes are 2K bytes and 32K bytes, and the non-I/O loads vary from 0.1 to 0.4. A load of 1.0 means that at each cycle, each processor generates one byte of a request. Loads of 0.1 and 0.4 correspond to request rates of 1116 and 4464 requests/sec, respectively. The destinations for these requests are uniformly distributed among all possible destinations.

As stated earlier, for a given number of nodes, the number of compute nodes remains constant and the number of I/O nodes is varied. If the total number of I/O requests processed remains constant, as the number of I/O nodes is doubled, the I/O request rate per I/O node is reduced by one-half. To reduce the complexity in presenting the results, the *FLOPS/byte* parameter is used instead of the I/O request rate for PIOS results. The FLOPS/byte (F/B) is the ratio of the rate of executing floating point operations to the rate of performing I/O. (We assumed that a floating point operation occurs in one switch cycle). An F/B of 1 means that one byte of I/O is done for each floating point operation. Therefore, for a given number of compute nodes and for a given number of I/O requests, as the number of I/O nodes is varied, the I/O request rate received by each I/O node will vary, but the F/B ratio at the compute

node will remain constant. However, for a given number of compute nodes and I/O nodes, decreasing the F/B ratio corresponds to increasing the I/O request rate. Table 1 lists the F/B ratios used in this study and the corresponding I/O request rates for all compute node to I/O node ratios considered for the case of a 32K byte request.

The metrics used to evaluate parallel I/O performance include the average I/O read request time, the average I/O read response time, the average I/O read processing time, and the average I/O write processing time. The read request time is calculated from the time the request is first generated to the time the first byte arrives at the I/O node. The read response time is calculated from the time the first byte of the response leaves the I/O node to the time the last byte is received at the requesting node. The read processing time is the sum of the request time, the response time, and the time to process the read request at the I/O node (an input parameter). The write processing time is calculated from the time the request is first generated to the time the last byte is placed in the disk buffer. It therefore includes the time to process the write at the I/O node.

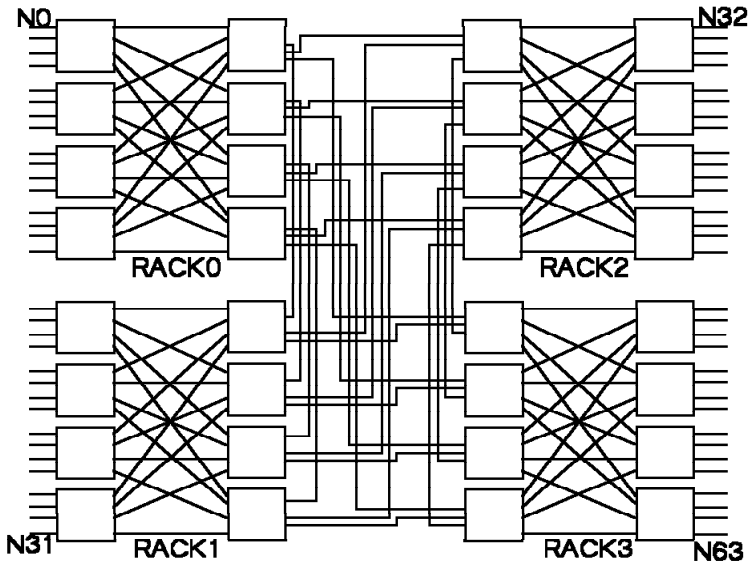

Figure 4. A Interconnection Topology for the 64-node MPP.

Also calculated is the message processing time for non-I/O messages. This time is calculated from the time the request is generated to the time the last byte is received by the compute node.

IV. Results

I/O Node Results. Presented in Figure 6 are the average I/O read and write processing times for disk cache hits and misses versus the I/O request rate for the I/O node. (Again, the disk cache miss ratio remained constant at 0.3 and the read (write) probability remained constant at 0.67 (0.33) for this study). The graphs show that for both read and write hits, the processing times are relatively small (hundreds of microseconds for 4K bytes and approximately 3 milliseconds for 32K bytes and less than 650 requests/sec). However, for misses, the processing times increase by an order of magnitude to between 14 and 25 milliseconds for up to 650 requests/sec. This is a result of the relatively large disk access time. Also, the larger 32K byte request size results in larger processing times when compared with the times for the 4K byte request size.

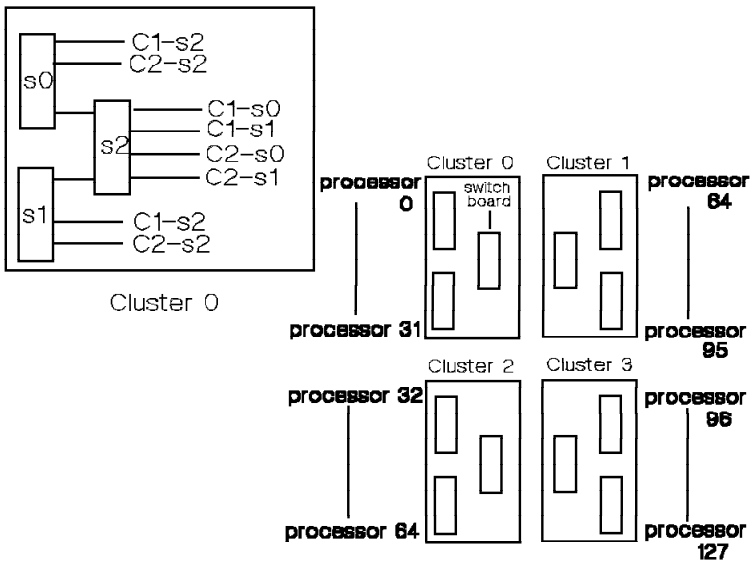

Figure 5. A Interconnection Topology for the 128-node MPP. Cx - Cluster x, Sy - Switch y.

Table 1. Assocation Between F/B Ratio and I/O Node Request Rates.

F/B at Compute Node	Compute Node-to-I/O Node Ratio	I/O Request Rate at each I/O Node (requests/sec)
114.5	4	80.00
	3	40.00
	2	26.67
	1	20.00
57.25	4	160.00
	3	80.00
	2	53.33
	1	40.00
28.63	4	320.00
	3	160.00
	2	106.67
	1	80.00
14.32	4	640.00
	3	320.00
	2	213.33
	1	160.00
7.16	4	1280.00
	3	640.00
	2	426.67
	1	320.00

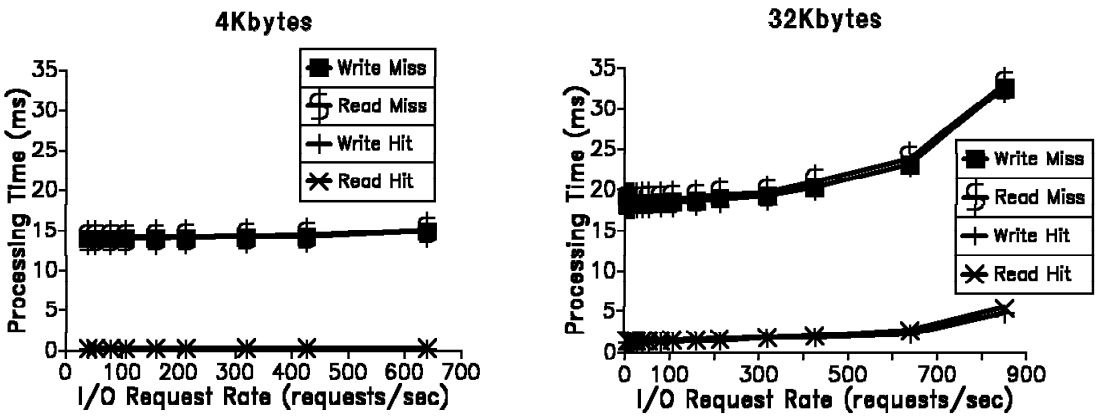

Figure 6. I/O Node Processing Times versus the I/O Request Rate and Request Sizes of 4K Bytes (left) and 32K Bytes (right).

From the graphs it is apparent that the hit processing times are independent of the I/O request rate presented, showing that even for up to 650 requests/sec, an effective disk cache can minimize any potential I/O-related contention in the node. However, for request rates greater than 850 requests/sec, the I/O node begins to saturate for the 32K byte request size. For disk cache misses, as the I/O request rate increases, the processing times increase; however the rate of increase is relatively small. The results show that the I/O node can support high I/O request rates without significant performance penalties. For the 32K byte request size, the read miss processing time is slightly larger than the write miss processing time. This is because processing a read includes transferring a 200 byte request size to the node, processing the request, and then preparing to send a 32K byte message to the requester once the data is accessed from the disk. During a write, 32K bytes of data are placed on the disk which is less time-consuming.

I/O-Only Traffic. All graphs presented are for 64 nodes and for 32K byte request sizes unless specified otherwise. The results of simulations with other values for these parameters exhibited similar behavior. Also, the data point reflecting the smallest number of I/O nodes (per total system size) for the lowest F/B ratio is not presented in any results. This is because the I/O node begins to saturate and therefore system performance degrades substantially.

Illustrated in Figure 7 are the read request and response times versus the number of I/O nodes and the F/B ratio. The read request time is relatively independent of the number of I/O nodes in the system. There are small decreases in the read response time as the number of I/O nodes increases. For a given F/B ratio, the minimal performance advantages exhibited by these parameters as the number of I/O nodes increases occur because for the F/B values (and associated I/O request rates) considered, the I/O node can process all requests upon receipt. Hence no additional network congestion is introduced as the number of I/O nodes increases. The small performance advantages for the read response time as the number of I/O nodes increases is attributed to reduced switch contention at the network stage to which the I/O nodes are attached. For a given total number of I/O requests processed, increasing the number of I/O nodes results in less work per I/O node and therefore a smaller number of requests and responses traversing the switches closest to the I/O nodes. The performance increases are not reflected in the read request time because the request size is 200 bytes and there is little or no contention associated with this small request size. Additional network contention is introduced as the F/B ratio decreases (or the I/O request rate increases). As a result, the

read request and response times increase as the F/B ratio decreases. The rate of this increase also increases as the F/B ratio decreases.

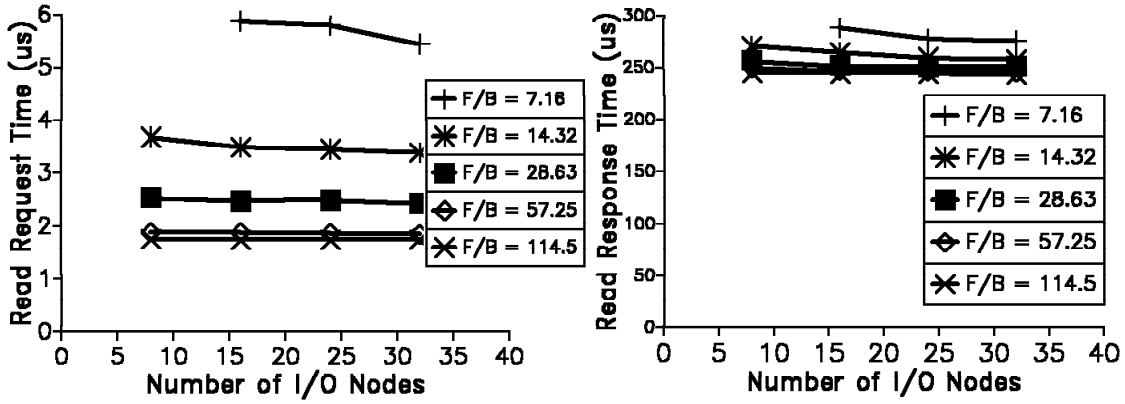

Figure 7. Network Traversal Times for I/O Reads versus the Number of I/O Nodes and the F/B Ratio. 64-node System, 32K Byte Request.

The read request and response time results differ from previous results that included I/O node blocking on every I/O write miss [11]. In those results, significant performance gains were observed as the number of I/O nodes in the system increased. Comparing those results with the results in this study shows that I/O node blocking was the primary reason for the performance gains in the earlier study. Therefore, minimizing or eliminating I/O node blocking improves the network traversal times of requests; however, it reduces the advantages of adding I/O nodes to the system.

Presented in Figure 8 are the read and write processing times versus the number of I/O nodes and the F/B ratio. These parameters decrease as the number of I/O nodes increases for small F/B ratios (less than 29). They are relatively independent of the number of I/O nodes for larger F/B ratios. Processing the smaller F/B ratios result in more I/O node contention, particularly for 8 and 16 I/O nodes. It therefore takes more time to process a request at the I/O node. With the larger F/B ratios, the requests are processed with little to no contention. The total read processing time is on the order of milliseconds while the read request and response processing times are on the order of microseconds. This shows that the total read processing time is dominated by the processing of the request at the I/O node.

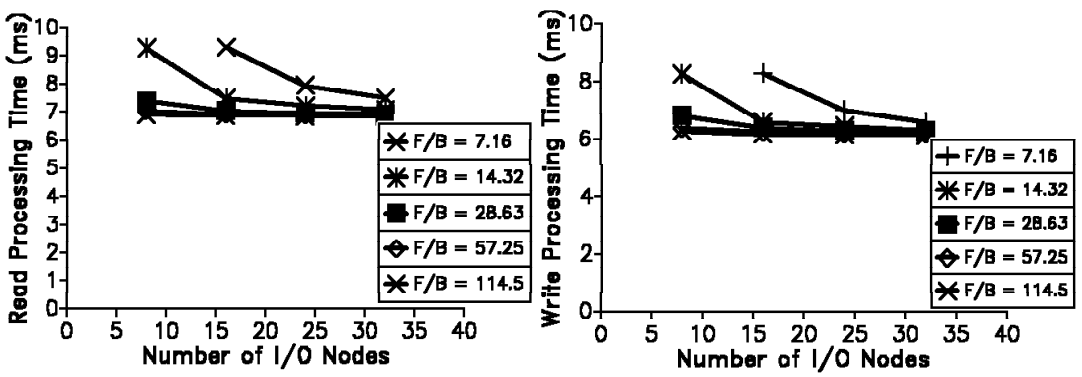

Figure 8. I/O Processing Times versus the Number of I/O Nodes and the F/B Ratio. 64-Node System, 32K Byte Request.

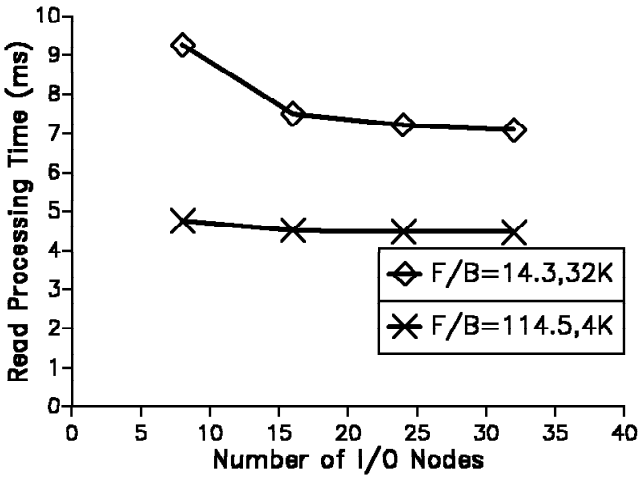

Figure 9. Read Processing Time versus the Number of I/O Nodes, the F/B Ratio and the Request Size for a 64-node System.

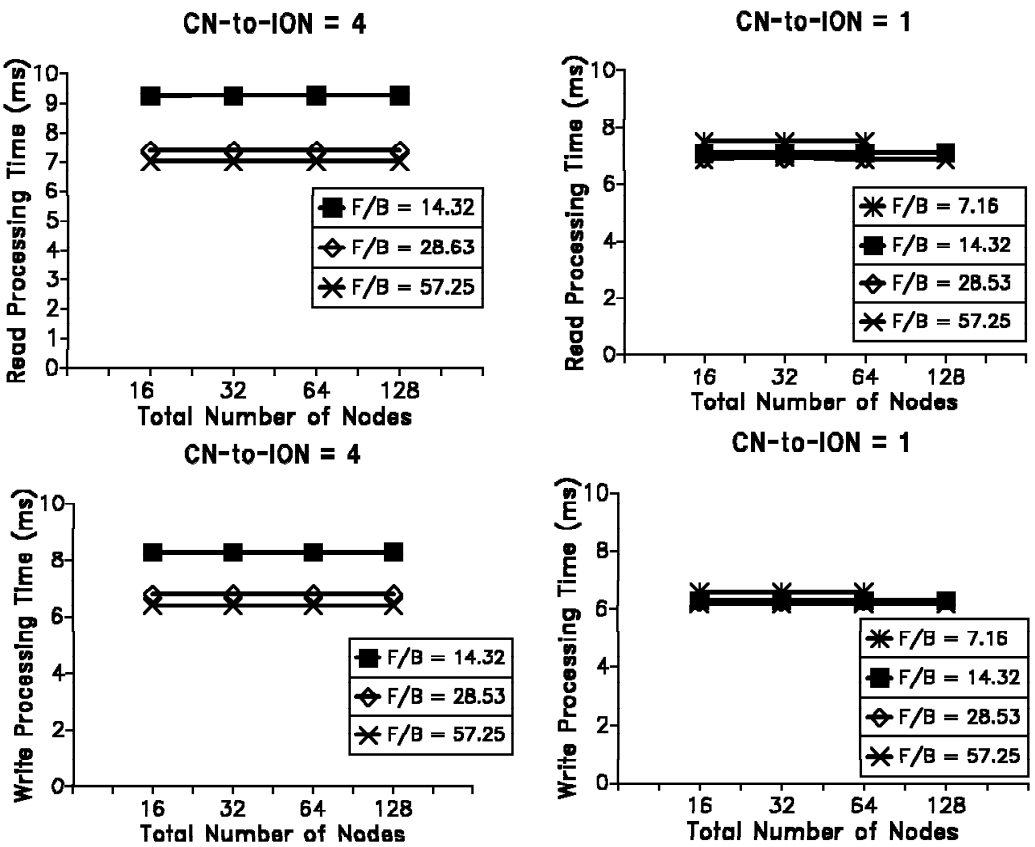

Figure 10. I/O Processing Times versus Total Number of Nodes, F/B Ratio, and Compute Node-to-I/O Node Ratio, 32K byte Requests.

Figure 9 shows the read processing time versus the number of I/O nodes for the 4K byte and 32K byte request sizes. The F/B ratio is 14.3 and 114.5 for the 4K byte and

32K byte request sizes, respectively. This results in an equivalent amount of I/O data processed (in bytes) for both request sizes. The read processing time decreases as the number of I/O nodes increase for the 4K byte request size. For the 32K byte request size, the metric decreases slightly as the number of I/O nodes increases from 8 to 16 but it is independent of the metric for 16 or more I/O nodes. The performance advantages shown for the 4K byte request size (as the number of I/O nodes increases) occur because the I/O request rate is eight times the rate for the 32K byte request size. In fact, for 8 I/O nodes, the per I/O node request rate is such that the node begins to saturate. As the number of I/O nodes increases, the per node contention is reduced, resulting in better performance. For 32K byte request sizes, the I/O node is never close to saturation for the request rates considered. There is little or no I/O node contention for 16 or more I/O nodes for this request size, resulting in near optimum processing times.

The read and write processing times versus the total number of nodes are presented in Figure 10 for compute node to I/O node (CN-to-ION) ratios of 1 and 4 and for a 32K byte request size. The F/B ratio varies from 7.16 to 57.25. This corresponds to request rates ranging from 40 to 640 requests/sec (see Table 1). The figure shows that the processing times are relatively independent of system size. The system is therefore scalable for all of these request rates.

While all the processing times are larger for the CN-to-ION ratio of 4 when compared to the times for the CN-to-ION ratio of 1, they are visibly larger for the smaller F/B ratios. This can be attributed to increased contention at the I/O node. With a CN-to-ION ratio of 4, there are fewer I/O nodes (relative to the smaller CN-to-ION ratio) and therefore more requests to process per node. For a given CN-to-ION ratio, reducing the F/B ratio results in increasing the I/O request rate per node. Both of these factors cause more contention at the I/O nodes and therefore longer processing times. However, the increased contention is more prevalent as the CN-to-ION ratio is increased. ***Non-I/O Message Effects***.

To study the impact of non-I/O message traffic on the performance of I/O traffic, we ran PIOS with both I/O and non-I/O message traffic. Two message sizes were used for the non-I/O message traffic, 2048 and 32K bytes. Small messages made up 95% of the non-I/O messages issued. Unless otherwise specified, the results presented for this part of the study assume a 32-node system. All other system sizes evaluated exhibited similar behavior.

Presented in Figure 11 is the I/O read request time versus the number of I/O nodes used in the system, the F/B ratio, and the non-I/O traffic loads of 0.0 (I/O-only traffic), 0.1, and 0.4 for a 32-node system size (loads of 0.1 and 0.4 correspond to request rates of 1116 and 4464 requests/sec, respectively). For all F/B ratios presented, adding non-I/O traffic with a load of 0.1 doubles the read request time. Increasing the non-I/O traffic load to 0.4 results in an order of magnitude increase in this metric. This is because the high request rates associated with this load results in increased network contention.

For all of the I/O loads considered, the read request time is independent of the number of I/O nodes for I/O-only traffic and for non-I/O loads of 0.1. This is consistent with the results presented in Figure 7. However, at the non-I/O traffic load of 0.4, there is a reduction in the read request time as the number of I/O nodes increases to 12. Further increasing the number of I/O nodes from 12 to 16 results in a performance degradation for this metric. In this study, increasing the number of I/O nodes results in less I/O traffic per node for a given total number of I/O requests processed. This reduces network contention at the switches closest to the I/O nodes which in turn reduces the time for read requests to reach these nodes. This explains the read request time behavior for 4 to 12 I/O nodes in Figure 11 (non-I/O loads of 0.4).

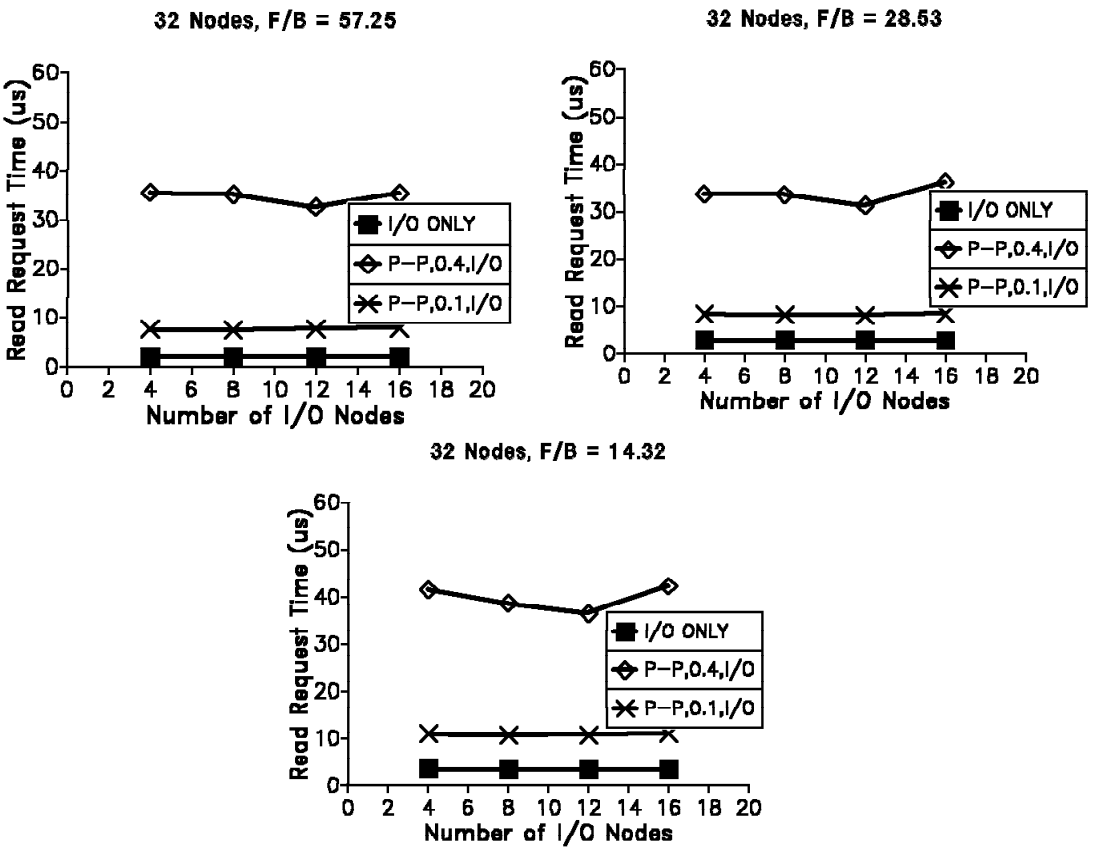

Figure 11. Read Request Time versus the Number of I/O Nodes, the F/B Ratio, and the Non-I/O (P-P) Traffic Load, 32K byte Request.

However, increasing the number of I/O nodes in the system also results in an increase in the number of switch ports to which an I/O node is connected (since in all cases where the CN-to-ION was greater than one, some of the input ports were unconnected.) Therefore, as the number of I/O nodes is increased, there is additional contention at the the switches attached to the extra I/O nodes. This works to offset the performance gains described earlier. When the number of I/O nodes increases from 12 to 16 in Figure 11 (non-I/O loads of 0.4) the additional switch contention of the new I/O nodes is larger than any performance gains associated with reduced contention at the switches associated with the I/O nodes already present in the system. Hence, the read request time increases in this case. The increased performance seen by small increases in the number of I/O nodes and the reduced performance seen by larger increases in the number of I/O nodes is not present for the I/O-only and smaller non-I/O loads. This is because there are fewer numbers of bytes traversing the network and therefore less contention. This suggests that increasing the number of I/O nodes in the system from a small to moderate amount (4 to 12 for a 32-node system in this case) has some performance advantages with very high network loads.

Presented in Figure 12 is the read response time versus the number of I/O nodes and the non-I/O message traffic load for several F/B ratios. These response times increase with the additional network load, and the rate of increase also increases with the higher network load. However, compared to the read request time, the performance degradation is not as severe. This is because the size of the response (32K bytes) is much larger than the size of the majority of the non-I/O messages (2K bytes). This is not the case for the I/O read request size of 200 bytes. When the smaller read request

message contends with a non-I/O message, the percentage increase in network traversal times are much greater when compared to the read response messages.

The read response time is relatively independent of the number of I/O nodes when there is no non-I/O message traffic or when the non-I/O message traffic is not high. When the non-I/O message traffic is high, this time decreases as the number of I/O nodes increases.

Shown in Figure 13 are the read and write processing times versus the number of I/O nodes, the F/B ratio and the non-I/O message traffic load. All of these parameters are relatively independent of the non-I/O message traffic load. This is because these processing times are dominated by the time to process the request at the I/O node. Therefore, any performance advantages or disadvantages resulting from reduced or increased switch contention, respectively, is negligible compared to the I/O node processing times.

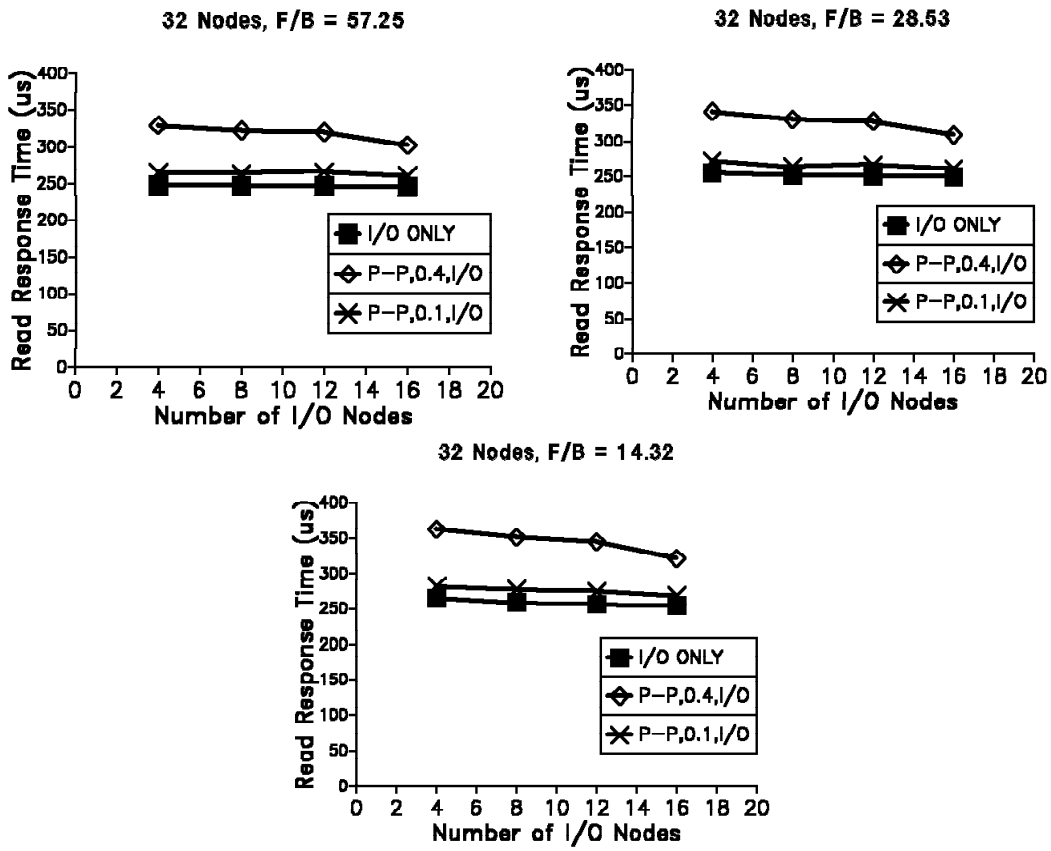

Figure 12. Read Response Time versus the Number of I/O Nodes and the Non-I/O (P-P) Traffic Load, 32K byte Request.

Effects of I/O traffic on non-I/O message traffic. We evaluated the effects of I/O traffic on non-I/O message traffic by using PIOS to simulate non-I/O message traffic in systems with and without I/O nodes. Presented in Figure 14 are the normalized message processing times versus the number of I/O nodes, the non-I/O message traffic load, and the F/B ratio for 32 nodes. The normalized time is calculated by dividing the average non-I/O message processing time with I/O traffic by the average non-I/O message processing time without I/O traffic. The results show that the non-I/O

message passing time degrades as the F/B ratio for the I/O traffic decreases and as the non-I/O message traffic load increases. The degradation ranges from 2% to 20%.

For the high non-I/O message traffic load of 0.4, the normalized parameter decreases as the number of I/O nodes increases up to 12. Increasing the number of I/O nodes beyond 16 results in an increase in the normalized non-I/O message processing time. Again, the increased contention at this high load results in performance improvements as the number of I/O nodes increases to 12. However, the increased switch contention at the switches directly associated with the additional I/O nodes counteract these performance improvements for more than 12 I/O nodes. This behavior is also seen in the results of non-I/O message traffic loads of 0.1, however the effect is smaller.

All of these results show that the architecture assumed is highly scalable, even when I/O and non-I/O traffic traverse the same network. Adding I/O nodes to the system is advantageous under two conditions: 1) when the request rates at the I/O nodes are high, causing the I/O node to approach saturation, and 2) when the non-I/O traffic in the network is very high, causing considerable network contention. These are also the two factors that may limit the scalability of the system.

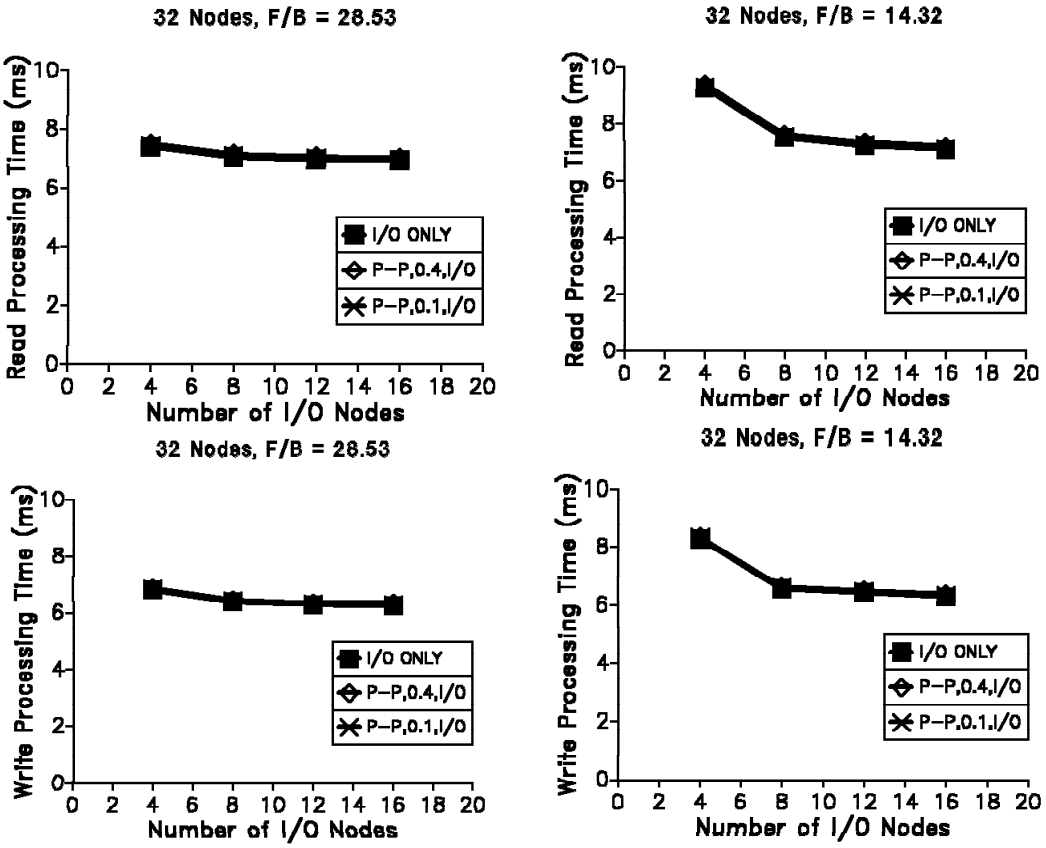

Figure 13. Read and Write Processing Times versus the Number of I/O Nodes and the non-I/O (P-P) Traffic Load, 32K byte Request.

V. Conclusion

The parallel I/O subsystem of the MPP assumed is evaluated by varying the number of I/O nodes in the system for a given number of compute nodes. Machine sizes range from 16 to 128 nodes. Both the impact of non-I/O message traffic on I/O traffic, and the effect of I/O messages on processor-processor communication are evaluated. Metrics used to measure performance include the I/O processing times and the non-I/O message passing times. For I/O-only traffic, the results show that the network traversal time is relatively independent of the number of I/O nodes in the system. These same times increase as the I/O request rate increases. Also, the total I/O processing times are relatively independent of the number of nodes in the machine for low to high I/O loads. As processing at the I/O node reaches saturation, increasing the number of I/O nodes in the system improves the I/O processing times. Introducing non-I/O message traffic in the network does impact I/O performance, particularly at high loads. The performance degradation ranges from 2% to an order of magnitude (however, the actual processing times range from microseconds to tens of microseconds). I/O node saturation and considerable network contention are the two factors that may limit the scalability of the architecture assumed.

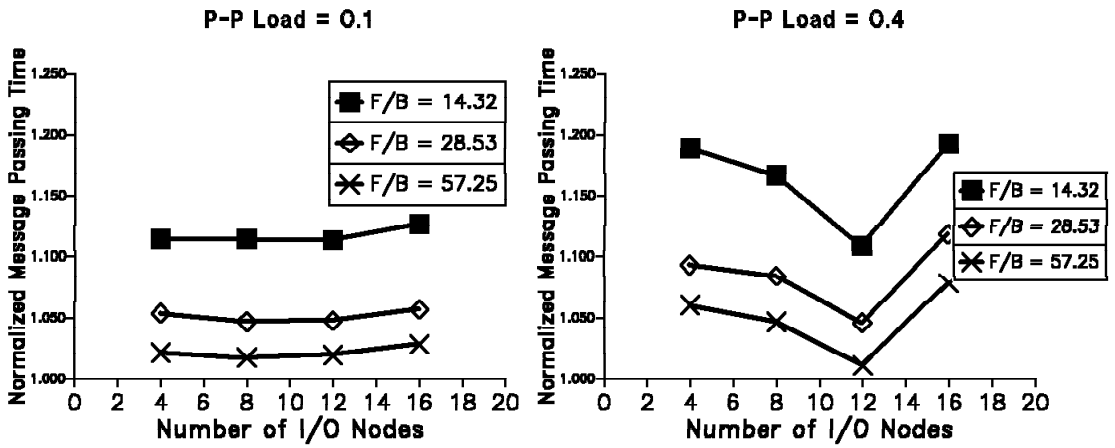

Figure 14. Non-I/O Traffic Message Passing Times versus the Number of I/O Nodes and Total Number of Nodes, 32K byte Request.

All of these results are calculated based on stochastically generated requests. Future work will concentrate on obtaining realistic I/O workload characteristics from applications written for MPPs (this may include much larger request sizes than presented in this paper as well as very different request rates) and the statistics obtained will then be used to drive PIOS. Also, this study did not assume that the data for a given I/O request is striped across multiple I/O nodes. Therefore, the subject of future work includes varying the percentage and size of I/O requests striped across multiple nodes to determine the performance impact of the resulting burstiness in I/O traffic. These issues are central to the effectiveness of parallel I/O. The I/O processing results are not limited by the network but by the processing of requests at the I/O node. Improved I/O node data management strategies, including increased disk cache hit ratios would greatly improve the performance of parallel I/O subsystems. All of these issues are central to the effectiveness of parallel I/O

REFERENCES

[1] Katz, R. H., G. A. Gibson, and D. A. Patterson, "Disk System Architectures for High Performance Computing," *Proceedings of the IEEE*, vol. 77, no. 12, pp. 1842-1858, December, 1989.

[2] Baylor, S. J., Benveniste, C., and Beolhouwer, L. J., "A Methodology for Evaluating Parallel I/O Performance for Massively Parallel Processors," *Proc. 27th Annual Simulation Symposium*, pp. 31-40, April 1994.

[3] International Business Machines Corporation, Scalable POWERparallel Systems High-Performance Technical Computing Solutions, Publication # GH23-2485-00, March 1994.

[4] Leiserson, C.E., et. al., "The Network Architecture of the Connection Machine CM-5," *4th Symposium on Parallel Algorithms and Architectures*, pp. 272-285, June 1992.

[5] Messina, P., "The Concurrent Supercomputing Consortium: Year 1," *IEEE Parallel and Distributed Technology*, vol. 1, no. 1, pp. 9-16, February 1993.

[6] Hayes, J.P., Mudge, T., Stout, Q.F., Colley, S., and Palmer, J., "A Microprocessor-based Hypercube Supercomputer," *IEEE Micro*, vol. 6, no. 5, pp. 6-17, October 1986.

[7] Alverson, R., Callahan, D., Cummings, B., Koblenz, B., Porterfield, A., and Smith, B., "The Tera Computer System," *International Conference on Supercomputing*, pp. 1-6, June 1990.

[8] Reddy, A.L.N. and P. Banerjee, "Design, Analysis, and Simulation of I/O Architectures for Hypercube Multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 2, pp. 140-151, April 1990.

[9] Livingston, M. and Q.F. Stout, "Distributing Resources in Hypercube Computers," *Proc. 3rd Confernce on Hypercube Concurrent Computer Applications*, pp. 40-48, January 1988.

[10] Baylor, S. J., Benveniste, C., and Hsu, Y., "Performance Evaluation of a Massively Parallel I/O Subsystem," *Proc. of the IPPS Workshop on Parallel I/O (also appeared in Computer Architecture News, pp. 5-10, September, 1994)*, pp. 1-8, April 1994.

[11] Baylor, S. J., Benveniste, C., and Hsu, Y., "Performance Evaluation of a Parallel I/O Subsystem," *IBM Research Report #RC 19749*, pp. 1-8, September 21, 1994.

[12] Stunkel, C.B., et. al., "Architecture and Implementation of Vulcan," *Proc. of the International Parallel Processing Symposium*, pp. 268-274, April 1994.

[13] Stunkel, C.B., D.G. Shea, D.G. Grice, P.H. Hochschild, and M. Tsao, "The SP1 High-Performance Switch," *Proc. of the Scalable High Performance Computing Conference*, pp. 150-157, May 1994.

[14] Corbett, P., S. J. Baylor, and D. Feitelson, "Overview of the Vesta Parallel File System," *Computer Architecture News*, vol. 21, no. 5, pp. 7-14, December 1993.

[15] Corbett, P., Baylor, S. J., and Feitelson, D., "Overview of the Vesta Parallel File System," *Proceedings of the Workshop on Input/Output in Parallel Computer Systems, IPPS '*93,, pp. 1-16, April 1993.

[16] Pierce, P., "A Concurrent File System for a Highly Parallel Mass Storage Subsystem," *Proceedings of the Fourth Conference on Hypercube Computers and Applications*, pp. 155-160, 1989.

[17] Gordon, R.F., and MacNair, E.A., The RESearch Queueing Package Modeling Environment (RESQME), Yorktown Heights, NY: IBM, Revised, September 14 1993.

[18] Kotz, D. and Nieuwejaar, N., "Dynamic File-Access Characteristics of a Production Parallel Scientific Workload," *Supercomputing '*94, pp. 640-649, November 1994.

Copies may be requested from:

IBM Thomas J. Watson Research Center
Distribution Services F-11 Stormytown
Post Office Box 218
Yorktown Heights, New York 10598