The Impact of Spatial Layout of Jobs on Parallel I/O Performance

Jens Mache
Mathematical Sciences
Lewis & Clark College
jmache@lclark.edu

Virginia Lo and Marilynn Livingston
Computer and Information Science
University of Oregon, Eugene, OR 97403
{lo, mlivings}@cs.uoregon.edu

Sharad Garg
Intel Corporation
Beaverton, OR 97006
sharad@co.intel.com

Abstract

Input/Output is a big obstacle to effective use of teraflops-scale computing systems. Motivated by earlier parallel I/O measurements on an Intel TFLOPS machine, we conduct studies to determine the sensitivity of parallel I/O performance on *multi-programmed mesh-connected* machines with respect to number of I/O nodes, number of compute nodes, network link bandwidth, I/O node bandwidth, spatial layout of jobs, and read or write demands of applications.

Our extensive simulations and analytical modeling yield important insights into the limitations on parallel I/O performance due to *network contention*, and into the possible gains in parallel I/O performance that can be achieved by tuning the *spatial layout of jobs*.

Applying these results, we devise a new processor allocation strategy that is sensitive to parallel I/O traffic and the resulting network contention. In performance evaluations driven by synthetic workloads and by a real workload trace captured at the San Diego Supercomputing Center, the new strategy improves the average response time of parallel I/O intensive jobs by up to a factor of 4.5.

1 Introduction

Input/Output is a critical problem for high-performance computing. The cost of transferring data between processors and disks is considered the biggest obstacle to effective use of teraflops-scale computing systems [35]. To alleviate the I/O bottleneck, systematic attention at all levels – applications, systems software and architecture – is called for [15, 14].

In this paper, we present results of a study on how parallel I/O performance is affected by the *spatial layout of jobs* on a mesh-connected machine. We use the term spatial layout to refer to the location of a job's compute nodes relative to the system's I/O nodes. This study is motivated by earlier measurements of the parallel I/O performance on a TFLOPS machine with up to nine I/O nodes and several hundred processor nodes at Intel [12, 13]. These experiments showed that certain layouts of jobs relative to the I/O nodes saturated the communication bandwidth of some network links, resulting in serious parallel I/O performance degradation.

In order to understand the phenomena observed, we conducted a careful analysis involving both dynamic simulation and analytic modeling. We assume a TFLOPS-like architecture, i.e. a mesh-based topology, XY-wormhole-routing, and I/O nodes configured on one side of the mesh. We model a workload of heavily I/O intensive jobs, with each job generating I/O traffic between all of its compute nodes and all of the system I/O nodes. We evaluated parallel I/O performance sensitivity to number of I/O nodes, number of compute nodes, network link bandwidth, I/O node bandwidth, spatial layout of jobs, and read or write traffic.

Our study yielded surprisingly strong results regarding (1) the limitations on parallel I/O performance due to network contention, and (2) the possible gains in parallel I/O performance that can be achieved by tuning the spatial layout of jobs. Our results are summarized below.

- If the I/O nodes are configured on one side of the mesh, increasing the number of I/O nodes yields diminishing returns for *realizable I/O throughput*. At a specific point (which we call the cutoff point), network contention levels are such that only marginal gains result from adding I/O nodes. In general, we show that under heavy I/O traffic (all compute nodes write to all I/O nodes), the theoretical cutoff point is dependent on f, the ratio of network bandwidth per link to I/O bandwidth per node, and on the layout of compute nodes relative to I/O nodes. We show that parallel I/O is not scalable beyond $4f$ I/O nodes, regardless of the size of the mesh.
- The *spatial layout* of jobs relative to the I/O nodes directly affects network contention levels and thus has a significant impact on realizable I/O throughput. By controlling both the "shape" of the compute nodes and their location relative to the I/O nodes, we can reduce network contention and approach the theoretical best case described above. Figure 1 shows the difference in parallel I/O throughput for two spatial layouts.
- Considering I/O traffic only, the best layout strategy assigns compute nodes so that they lie parallel to the

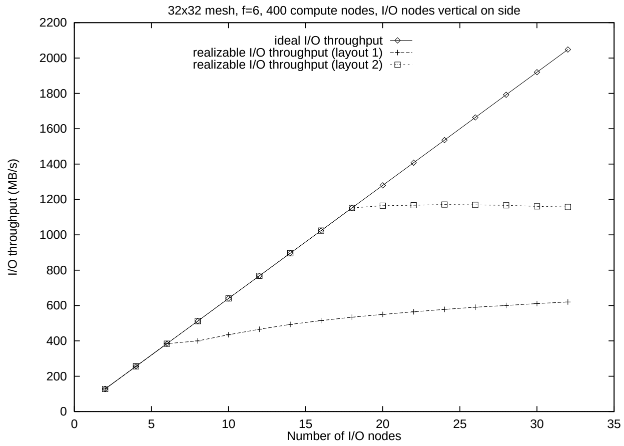


Figure 1: Degradation of parallel I/O throughput

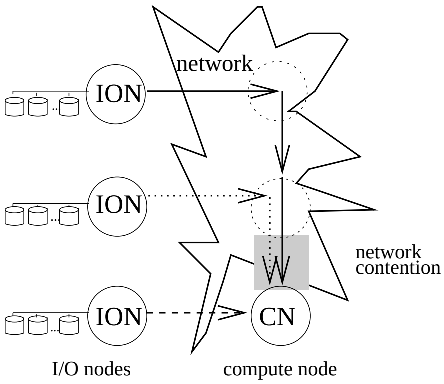


Figure 2: Parallel I/O causing network contention

I/O nodes. We devised a Parallel Layout Allocation Strategy (PLAS) and tested its performance using synthetic workloads and a real workload trace of 6087 jobs captured from a 400 node Paragon at the San Diego Supercomputing Center. Our simulations show that with respect to parallel I/O throughput and response time, PLAS consistently outperforms known allocation schemes currently in use on commercial and research machines. Average response time of jobs improved by up to a factor of 4.5.

- Contention effects are more serious for write traffic when the I/O nodes lie in the Y direction, and more serious for read traffic when the I/O nodes lie in the X direction. This effect is due to the use of XY-routing. Future allocation and scheduling algorithms should take into account whether jobs are read or write dominant in order to improve parallel I/O throughput.

The rest of this paper is organized as follows. In Section 2, we give background information on parallel I/O, network contention and spatial layout of jobs. Section 3 defines our problem model. Section 4 presents simulations that show realizable I/O throughput as a function of system parameters and spatial layout scheme. Section 5 present our analytic modeling and the theoretical limits of parallel I/O throughput. Section 6 describes how spatial layout affects parallel I/O performance and presents PLAS, a new processor allocation strategy that optimizes for parallel I/O throughput. Section 7 discusses conclusions.

2 Background and Related Work

2.1 Parallel I/O

Data-intensive applications, e.g. multimedia information systems or scientific computations with massive datasets and databases, require high-bandwidth, and low-latency access to secondary memory. Parallel I/O, accessing multiple disks in parallel, has the potential to deliver these high requirements, but achieving this potential involves management of many complex, inter-dependent issues.

Parallel I/O performance is limited by uncoordinated I/O operations, by ineffective use of disks and inefficient disk caching, and by ineffective use of the I/O network. To improve these inefficiencies, a variety of approaches have been proposed, including collective I/O algorithms [7, 33] and disk-directed I/O [17, 39]. In this paper, we focus on the effective use of the I/O network.

2.2 Network Contention

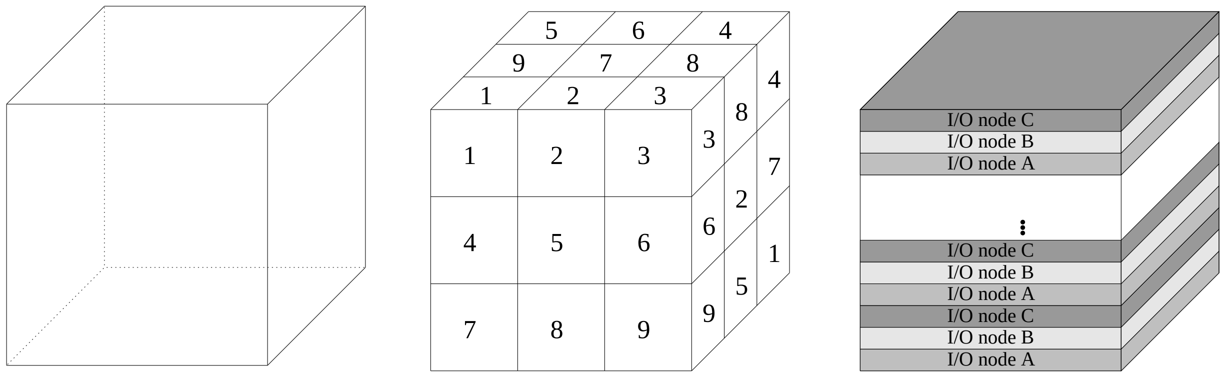
Moving data over the interconnection network – e.g. between I/O nodes and compute nodes – is quite costly. This cost is exacerbated by network contention. *Network contention* occurs if several messages want to use the same network link at the same time (see Figure 2). Because network contention affects the transfer time of data, it can severely lengthen the execution time of jobs.

Both I/O traffic and intra-job communication can suffer from network contention. Most previous research has focused on network contention *due to intra-job communication* [30, 4, 8, 29, 27, 24], whereas our focus is on contention *due to parallel I/O traffic*. Parallel I/O traffic bears many similarities to intra-job communication; yet it differs from intra-job communication, both in the size of the messages, and also in what links are stressed in the network. Especially since the I/O nodes are typically shared by all jobs in the system, parallel I/O traffic is prone to network contention and to causing nonuniform traffic spots [18].

Although parallel I/O-based contention has been recognized as a potential problem [3, 11, 17], we know of only one study that tried to minimize it. Winslett et al. [5] avoided data transfers on FDDI- or Myrinet-connected network of workstations by changing the placement of part-time I/O nodes. We were curious how spatial layout of jobs affects parallel I/O-based contention on mesh architectures such as the Intel TFLOPS ASCI/Red.

2.3 Spatial Layout of Jobs

Whereas the spatial layout of a job can be explicitly specified by the user, typically the spatial layout of jobs is controlled by the scheduling and allocation software which selects the set of processors to be allocated to an incoming job. Two characteristics define a job's spatial layout: the "shape" of the allocated compute nodes and their location within the mesh topology. The approaches used by current commercial allocation software fall into two categories: block-based allocation schemes [42] or linear allocation [28, 32]. The five spatial layouts that we consider in this study are generalizations of these known allocation schemes. They are block



3-dimensional data, logical view **(a)** | data distribution across 9 compute nodes **(b)** | file layout across 3 I/O nodes **(c)**

Figure 3: Data distribution and file layout for a scientific application

corner, block center, orthogonal, parallel, and diagonal (see Figure 5).

Previous work on spatial layout of jobs focused on fragmentation [20, 45, 22] and on contention due to intra-job communication [22, 42, 29, 40, 25]. While researchers have started to investigate the impact of I/O on adaptive partitioning [38], on gang scheduling [19], and on load balancing [6], the impact on parallel I/O from spatial layout of jobs has not been investigated yet.

3 Problem Description

Our goal is to study the effects of spatial layout of jobs on I/O-based network contention and realizable I/O throughput. In this section, we describe our system model and our I/O traffic model, and we illustrate with a simple example.

3.1 System Model

The architecture assumed is a high-performance parallel system similar to the Intel TFLOPS ASCI/Red [26, 12]. We concentrate on mesh-interconnected direct networks where messages contend for bidirectional links. We assume the current technologies of dimension-ordered XY-routing and wormhole flow control [31]. The I/O subsystem consists of disks attached to I/O nodes. We assume that the I/O nodes are fixed "vertically" at one side of the mesh. The I/O nodes connect to other nodes in the system by the same interconnection network that connects the compute nodes. The parallel file system distributes files across disks and I/O nodes such that multiple disks work on a data transfer in parallel.

3.2 I/O Traffic Model

We assume the hardest I/O traffic pattern where every compute node exchanges data with every I/O node and all the data are unique (personalized). This I/O traffic pattern, which can be modeled by a complete bipartite directed graph, is representative of a wide range of scientific applications in which a large data structure is read from or written to a file striped across I/O nodes and is then re-distributed to the compute nodes according to one of the classic data distribution patterns, such as those defined in HPF [16].

Scalar Pentadiagonal (SP) and Block Tridiagonal (BT), part of the NAS Parallel Benchmarks [2], are examples of the type of I/O-bound applications which our traffic model fits well. These two applications account for a large portion of the computational fluid dynamics workload at NASA Ames [34]. In both applications, a three-dimensional grid of variables (see Fig. 3a) is repeatably computed and saved to a striped file which can be processed by visualization software. A parallel implementation of SP and BT that provides good load balancing and coarse-grain communication employs the Multi-partition method [34]. This method uses the following *data distribution*: If there are n compute nodes, the three-dimensional grid is broken up into $n\sqrt{n}$ three-dimensional sub-cubes. The sub-cubes are distributed to the compute nodes so that for any plane of sub-cubes, each compute node gets exactly one sub-cube of data from that plane (see Fig. 3b).

One possible *file layout* is shown in Fig. 3c. A file represents a linearization of the global data structure, such as the row-major ordering of a three-dimensional array. This linearization is often called *canonical file*. In a striped file, chunks of the canonical file are distributed across I/O nodes, typically in a round-robin fashion. It is easy to see that for the data distribution of the Multi-partition method, every compute node has to write to every I/O node, regardless of the stripe size.

3.3 Example

This example, which uses two I/O nodes and two compute nodes, illustrates how the spatial layout of jobs affects parallel I/O performance. Figure 4a shows the I/O traffic graph in which each compute node (CN) sends 256 MB data to each I/O node (ION). In Figure 4b the two compute nodes are placed orthogonally in relation to the I/O nodes, while in Figure 4c, the two compute nodes are placed parallel to the I/O nodes. It turns out that the realizable I/O throughput rate for the orthogonal configuration is one half that of the parallel configuration. This difference is due to network contention incurred by the messages in transit to the I/O nodes.

Assuming that each I/O node can process data at a sustainable rate of 64 MB/s, the approximate time for each I/O node to process its incoming data is $\frac{512MB}{64MB/s} = 8$ sec-

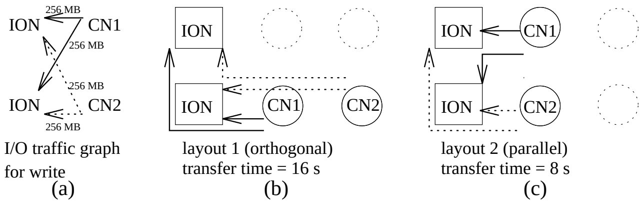


Figure 4: A simple example showing that spatial layout affects I/O throughput

onds. Also assume that each network link has the same data transfer rate of 64 MB/s. If the job is allocated as shown in Figure 4b, data from all four sender-receiver pairs (1 GB) has to traverse the west-bound channel that is located between CN1 and ION. This takes $\frac{1GB}{64MB/s} = 16$ seconds. Thus, the network limits the overall I/O throughput to $\frac{1GB}{16s} = 64$ MB/s. In contrast, if the job is allocated as shown in 4c, every network link carries at most 512 MB of data. Now, the network can accomplish the data transfer in $\frac{512MB}{64MB/s} = 8$ seconds. This matches the time it takes the I/O nodes to process the data, yielding overall I/O throughput of $\frac{1GB}{8s} = 128$ MB/s, which is twice as big.

4 Simulations

In order to quantify the effects of spatial layout on I/O throughput, we simulated the realizable I/O throughput rate under each of the five layout schemes described in Fig. 5. We also varied architectural parameters in order to study the scalability of I/O throughput as a function of the number of I/O nodes, the number of compute nodes, and f, the ratio of network link bandwidth to I/O node bandwidth.

More precisely, our simulator models the following:

- mesh architecture of size $k \times k$, $k = 16$ and 32; these are sizes typical of the Intel Paragon and TFLOPS, respectively.
- m I/O nodes located in a vertical column on the west border of the mesh, $m \leq k$;
- packet level XY-wormhole-routing with one virtual channel;
- sustainable per node transfer rate B_O of the I/O nodes;
- sustainable per link transfer rate B_l of the interconnection network;
- n compute nodes whose I/O traffic pattern involves exchange of data with every I/O node, both read and write dominant I/O are modeled; thus, each job sends data on $m * n$ different sender-receiver pairs;
- five spatial layout patterns as described above: block corner, block center, parallel, orthogonal, and diagonal (see Fig. 5).

The **I/O node throughput rate** is the aggregate data transfer rate when all I/O nodes achieve the sustainable per node transfer rate of B_O, ignoring the network. Thus, the I/O node throughput rate grows linearly with the number of I/O nodes and equals $m * B_O$.

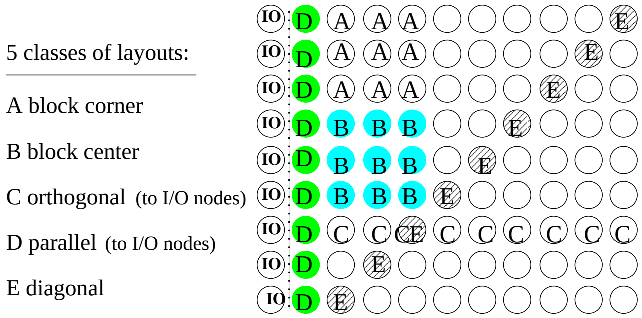


Figure 5: Five classes of spatial layout of one job of size 9

The **network throughput rate** is the aggregate data transfer rate through the interconnection network and is measured by the simulator which models packet level, wormhole, XY-routing.

Figure 6 shows throughput rates as functions of jobsize (number of compute nodes n) and number of I/O nodes (m), for a 32×32 mesh, with $f = \frac{B_l}{B_O} = 6$, the ratio of network link bandwidth to I/O node bandwidth for the TFLOPS machine, given current actual sustainable throughput rates of $B_O \approx 64$ MB/sec. and $B_l \approx 380$ MB/sec. Each of the five graphs represents one of the five spatial layout patterns.

In all five graphs, the white surface shows the I/O node throughput rate and the shaded surface shows the network throughput rate. The **realizable I/O throughput rate** is the minimum of the I/O node throughput rate and the network throughput rate, depending on whether the I/O nodes or the network is the limiting factor in handling the I/O traffic.

The key observations to be made from these graphs and Figure 1 are the following:

- If one looks at I/O throughput versus the number of I/O nodes, it is clear that at some point, adding I/O nodes is useless since the network limits the realizable I/O throughput (i.e., the shaded surface falls below the white surface).
- The choice of spatial layout affects the degree to which the network limits I/O throughput, and thus affects the cutoff point. For write traffic, the best spatial layout pattern is diagonal or parallel, followed by block center, block corner, and orthogonal. The ranking of spatial layouts from best to worst is consistent over all parameter ranges for m, n, k, and f. The ranking order for read I/O traffic is different than the ranking for write traffic, but is also consistent over all parameter ranges for m, n, k, and f. (The different rankings for read and write traffic are due to XY-routing.)

The impact of this phenomenon is brought home by looking at the specifics of the TFLOPS machine mentioned earlier. Garg et. al. [13] measured I/O throughput of the TFLOPS ASCI/Red where up to 9 I/O nodes were configured into one PFS file system, i.e. every compute node exchanged data with every I/O node. For a big job that utilized most of the machine, realizable I/O throughput scaled with the number of I/O nodes, and nine I/O nodes achieved over 0.5 GB/s throughput. Is it therefore safe to conclude that doubling the number of I/O nodes from 9 to 18 will

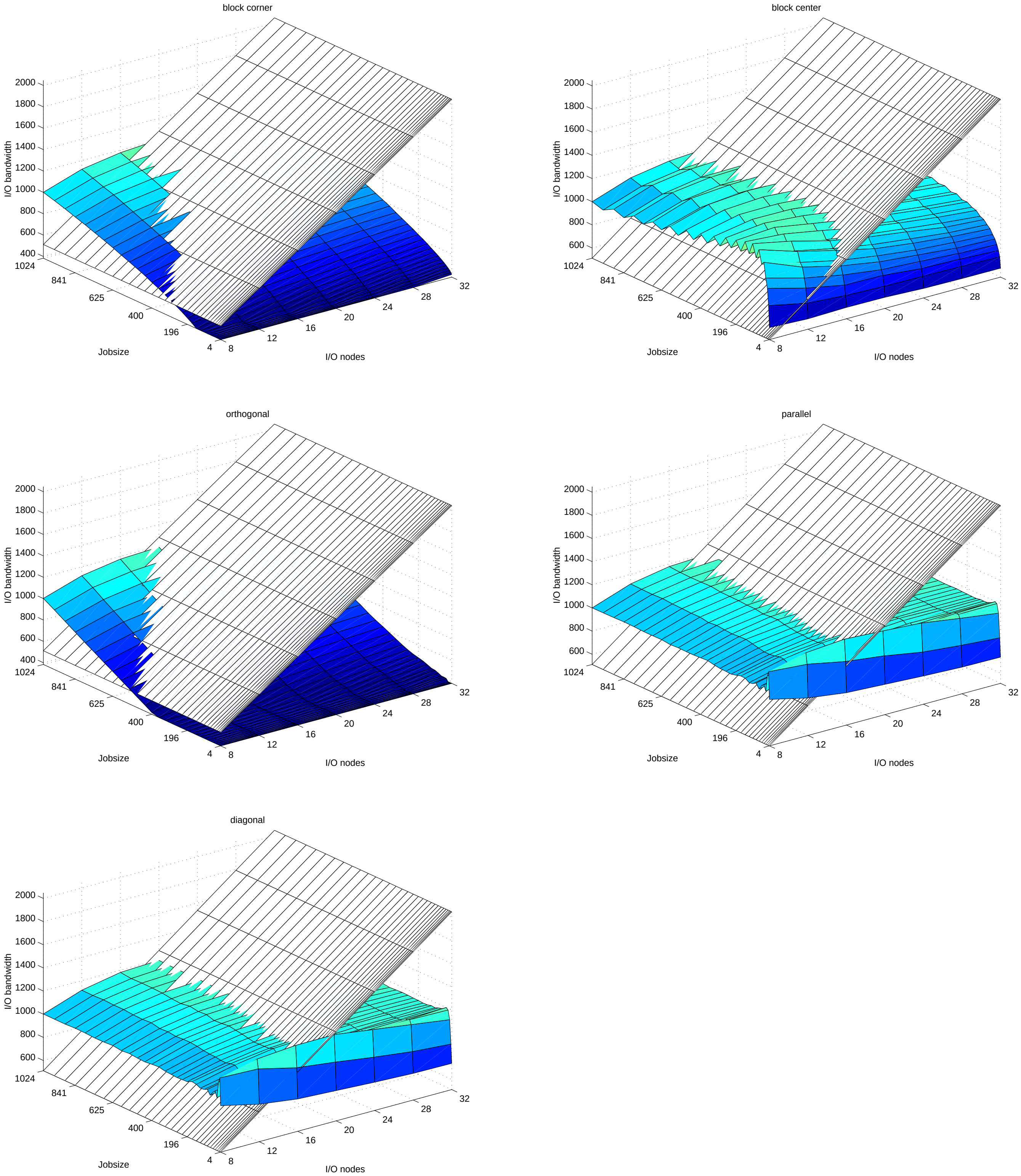


Figure 6: I/O throughput depending on number of I/O nodes and jobsize for five different spatial layouts of jobs (32×32 compute mesh, write traffic, $f = 6$)

m	number of I/O nodes
n	number of compute nodes exchanging data
k	length of one side of the quadratic mesh of compute nodes of the machine
B_0	effective bandwidth of each I/O node
B_l	effective bandwidth of each network link
f	quotient of B_l and B_0
B_{ION}	I/O throughput of all I/O nodes
B_{NET}	I/O throughput of the network
link_contention	number of distinct sender-receiver pairs using that link
hotspot	link with maximal *link_contention*
max_contention	value of *link_contention* at hotspot
I/O node throughput	data transfer rate of all I/O nodes combined
network throughput	data transfer rate through the network considering the limiting hotspot link
realizable throughput	min(I/O node throughput, network throughput)

Table 1: Notation and definitions

double the I/O throughput to 1 GB/s? Our work indicates that it depends on the spatial layout scheme whether 18 I/O nodes can be used productively and thus whether an I/O throughput of one GB/s is achievable (see Figure 1).

5 Analytic Modeling

To further understand the nature of the network contention arising from parallel I/O and its effect on I/O throughput, we conducted analytic modeling of the effects of spatial layout on network contention and I/O throughput using a graph theoretic model of the problem. Our analytic model optimistically assumes no loss in total network link bandwidth if a link is shared by several data transfers.

We looked at several performance metrics. *Max_ contention* measures the maximum number of I/O traffic sender-receiver pairs that contend for the same network link. We looked at throughput for the I/O nodes alone, throughput for the network alone in the presence of contention, and the realizable I/O throughput rate which is the minimum of I/O node and network throughput. Our mathematical analysis accurately confirms the observations derived from our simulation analysis.

5.1 Graph Theoretic Model and Definitions

Our presentation of the graph theoretic model will be informal in this section due to space limitations. All symbols and terms are summarized in Table 1.

Definition: The *system architecture graph* consists of m I/O processors located vertically on the west side of a $k \times k$ mesh of compute processors.

A given parallel application requires n compute nodes to be selected from the mesh. As described in Section 3.2, we assume the I/O requirements of the job to be that of personalized broadcast, from all m I/O nodes to all n compute nodes (allocated to the job) in the case of read, and from all n compute nodes to all m I/O nodes in the case of write.

Definition: The *I/O traffic graph* $K_{m,n}$ is a complete, directed, bipartite graph with one set of vertices representing I/O nodes and the other set representing compute nodes. Edges are directed from the set of I/O nodes to the set of compute nodes in the case of a *read* traffic graph.

We *embed* the graph $K_{m,n}$ into our architecture graph subject to the following constraints:

- the m I/O nodes of the I/O traffic graph are mapped to the m I/O processors in the architecture graph;
- the n compute nodes of the I/O traffic graph are mapped to some subset of the compute processors in the mesh; we will consider five possible mappings that correspond to the five spatial layout schemes described above (block corner, block center, parallel, diagonal, orthogonal).
- each edge from I/O node i to compute node j in the I/O traffic graph is mapped to the path from I/O processor o_i to compute processor p_j according to the XY-routing scheme of the mesh.

Definition: The *link contention* of a link l is the number of edges of the I/O traffic graph whose corresponding paths include l when we embed the I/O traffic graph into the mesh network as described above. Conceptually, link contention is the number of distinct sender-receiver pairs sharing that network link.

A link with maximal link contention is a *hotspot* and the value of this maximum contention is denoted by *max _contention*.

Referring back to the simple example of Fig. 4, the four sender-receiver pairs are mapped to different, but overlapping, paths in the network. This results in *max_contention* of 4 for layout 1 and *max_contention* of 2 for layout 2.

Also, recall the following terms from Section 4 above which we define now formally:

Definition: The *I/O node throughput rate* is the aggregate data transfer rate when all I/O nodes achieve the maximum per node bandwidth of B_O, ignoring the network. Thus, the I/O node throughput rate is

$$B_{ION} = m * B_O \quad (1)$$

Definition: The *network throughput rate* is defined as the aggregate data transfer rate through the interconnection network and is equal to the total amount of data to be transfered divided by the time incurred by the bottleneck link to transfer its *max_contention* messages. Assuming that each message carries D bytes of data, the total amount of data to be transfered is $m * n * D$, and the time incurred by the bottleneck link is $\frac{max\_contention * D}{B_l}$. Thus, the network throughput rate is

$$B_{NET} = \frac{m * n * B_l}{max\_contention} \quad (2)$$

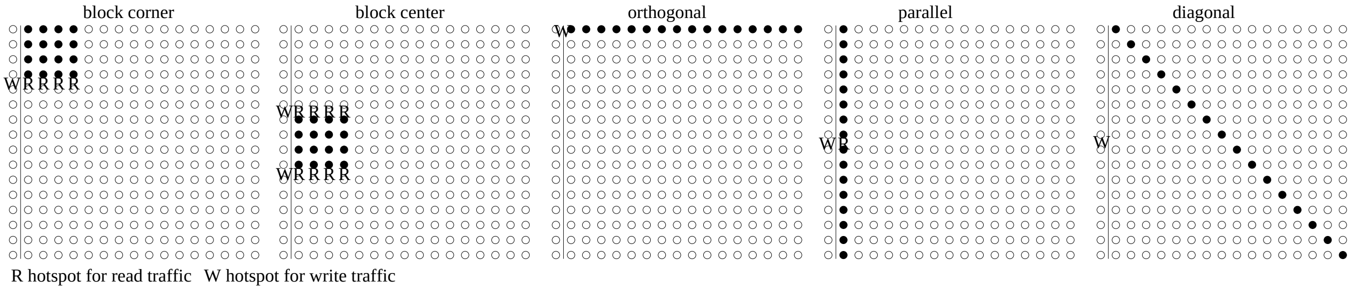


Figure 7: Location of hotspots

Definition: The *realizable I/O throughput rate* is the rate in bytes per second at which I/O data can be read or written to/from I/O nodes and is equal to the lesser of the I/O node throughput rate and the network throughput rate.

$$realizable\ I/O\ throughput = min(B_{ION}, B_{NET}) \quad (3)$$

5.2 Best and Worst Case Network Contention and Network Throughput

In this section we develop formulas for the theoretical highest (worst) and lowest (best) levels of *max_contention* possible in our system. These values can be used to determine the theoretical worst and best network throughput rates.

Theorem 1 The worst and best case values for *max_contention* are $m * n$ and $\frac{m*n}{4}$, respectively.

Proof idea: Given XY-routing, the level of contention on any given link can be computed by straightforward formulas. For example, the link contention on any westbound channel from $(x+1, y)$ to (x, y) is given by the number of senders (u, v) with $u > x$ and $v = y$ times the number of receivers (s, t) with $s \leq x$.

The highest value for *max_contention* is $m * n$ which occurs when the compute nodes lie orthogonally in relation to the I/O nodes. In this case, all $m * n$ messages contend for the hotspot horizontal link in Figure 7 (orthogonal).

The lowest value for *max_contention* is $\frac{m*n}{4}$, which occurs when messages sent from half of the n compute nodes to half of the m I/O nodes contend for the hotspot vertical link. See Figure 7 (parallel).

Assuming that the I/O nodes are on one side, no lower value is possible. The proof boils down to finding the minimum i, $0 \leq i \leq n$, of the function $max(\frac{m}{2} * i, \frac{m}{2} * (n-i))$, which represents the link contention on the south-bound and on the north-bound channel in the middle of the I/O nodes.

Theorem 2 The worst and best case values for network throughput rate are B_l and $4 * B_l$, respectively.

Proof: These formulae are derived directly by substituting the worst and best case values for *max_contention* in the definition of network throughput rate.

The above two theorems confirm that when a complete bi-partite I/O traffic graph is mapped to a mesh architecture with I/O nodes on the boundary, the network contention occurring on the bottleneck link enforces an upper bound on the network throughput rate. Furthermore, the network throughput rates are independent of the number of I/O nodes.

5.3 The Limitations of Adding I/O Nodes

The next theorem describes the limitations of adding I/O nodes to the system configuration. Recall the definitions from Table 1 and formulas (1) and (2).

Theorem 3 In the best case, at most $4f$ I/O nodes can be used to increase realizable I/O throughput, where $f = \frac{B_l}{B_O}$. In the worst case, at most f I/O nodes can be used to increase realizable I/O throughput.

Proof: Theorem 2 tells us that in the best case, $B_{NET} = 4 * B_l$. Thus, $m > 4f \Rightarrow m * B_O > 4 * B_l \Rightarrow B_{ION} > B_{NET}$. Thus, when m, the number of I/O nodes, is greater than $4f$, realizable I/O throughput is limited to the network I/O throughput rate which is independent of the number of I/O nodes.

Similarly, Theorem 2 tells us that in the worst case, $B_{NET} = B_l$. Thus, $m > f \Rightarrow m * B_O > B_l \Rightarrow B_{ION} > B_{NET}$. Thus, when m the number of I/O nodes is greater than f, realizable I/O throughput is always limited to the network I/O throughput rate which is independent of the number of I/O nodes.

Below these cutoff points ($m > 4f$ or $m > f$, respectively), realizable I/O throughput depends on $B_{ION} = m * B_O$ which grows with m.

For a mesh architecture with I/O nodes on the boundary, this theorem gives a much tighter upper bound on the number of I/O nodes than Feitelson et. al. [11], who did not take network contention into account and thus gave an upper bound of $m \leq n * f$.

We next analyze the network contention levels and network throughput rates achieved by our five spatial layout schemes which lie between the theoretical best and worst cases extremes.

6 Impact of Spatial Layout

In this section, we show how spatial layout affects the levels of network contention incurred and thus the realizable I/O throughput rate. We present the results of both analytic modeling and simulations. We then suggest a new spatial layout scheme (processor allocation scheme) to maximize realizable I/O throughput. We compare our scheme to several known processor allocation algorithms used on commercial and research machines using synthetic workloads and a real workload trace from the San Diego Supercomputer Center. Our simulations show that our PLAS scheme outperforms all others by up to a factor of 4.5.

Layout	write $max\_contention$	r.	read $max\_contention$	r.
diagonal	$\lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil$	1	n	1
parallel	$\lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil$	1	$\lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil$	4
block center	$\frac{n(n-\sqrt{n})}{2}$	2	$\frac{\sqrt{n}(n-\sqrt{n})}{2}$	2
block corner	$n(n-\sqrt{n})$	3	$\sqrt{n}(n-\sqrt{n})$	3
orthogonal	n^2	4	n	1

Table 2: $Max\_contention$ including ranking (analytic modeling, $k = n = m$)

Layout	write	read
diagonal	$B_l * 4$	$B_l * n$
parallel	$B_l * 4$	$B_l * 4$
block center	$B_l * 2 * \sqrt{n}/(\sqrt{n}-1)$	$B_l * 2 * n/(\sqrt{n}-1)$
block corner	$B_l * \sqrt{n}/(\sqrt{n}-1)$	$B_l * n/(\sqrt{n}-1)$
orthogonal	B_l	$B_l * n$

Table 3: Network throughput (analytic modeling, $k = n = m$)

6.1 Analytic Modeling

Our analytic modeling shows that network contention levels and realizable I/O throughput are sensitive to the spatial layout of the compute nodes, differing by a factor of four from the best spatial layout to the worst.

Figure 7 shows the hotspot link for each of the five spatial layout schemes for both read and write I/O traffic. Table 2 gives formulas and ranking orders for $max\_contention$ on the bottleneck link for the case where $m = n = k$, i.e. number of compute nodes equals number of I/O nodes equals one side of the mesh. In general, ranking of spatial layout schemes, from best to worst, matches that shown in Table 2 for read and for write, respectively, across all possible values for m, n, and k. Additional formulas are given in [23] for the other possible cases involving m, n and k. Table 3 gives formulas for network throughput.

6.2 Simulations

The graphs shown earlier in Figure 6 illustrate how spatial layout affects realizable I/O throughput. We summarize our results in Figure 8 which shows I/O throughput as a function of the number of I/O nodes. In this figure, the diagonal line gives the I/O node throughput rate (ignoring the limitations of the network). The two horizontal lines show the best and worst case values of network throughput, $4 * B_l$ and B_l, respectively. The other five lines show the network throughput rate for the five spatial layouts.

From this graph, we see that by careful choice of spatial layout scheme we can approach the theoretical best case for network throughput. We can also see that given, a spatial layout scheme, the point where the line for I/O throughput intersects that for network throughput is the point at which addition of I/O nodes has no further effect.

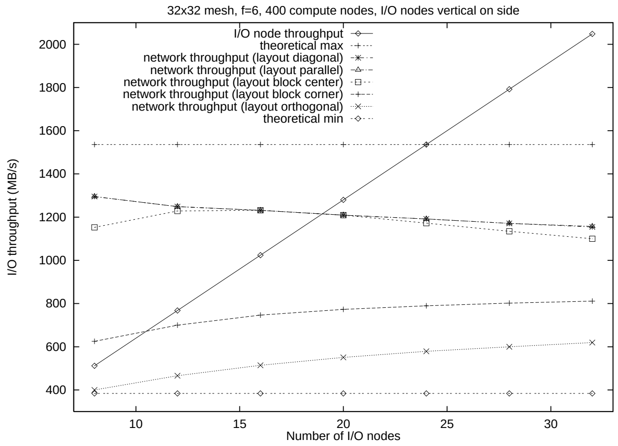


Figure 8: I/O throughput depending on number of I/O nodes (simulation results, $k = 32$)

6.3 A new parallel I/O sensitive allocation strategy

In this section, we apply the results of our research to the design of a new processor allocation strategy that is sensitive to the impact of spatial layout on parallel I/O throughput.

The *processor allocation strategy* controls the spatial assignment of a set of nodes to each scheduled job. Most high-performance parallel systems employ space sharing and variable partitioning. In *space sharing*, multiple jobs run on different processor nodes at the same time. *Variable partitioning* [10] means that each job is assigned as many compute nodes as it requested and that it runs on those nodes for the lifetime of the job. The goal of processor allocation strategies is to minimize the response time of jobs and maximize the system throughput over a stream of jobs.

We designed a Parallel Layout Allocation Strategy (PLAS) that is based on two observations regarding spatial layout: (1) parallel spatial layout is superior to block-based and orthogonal layouts, and (2) it is desirable that one half of the compute nodes are placed above the link in the middle of the I/O nodes and one half are placed below. Note that we do not consider diagonal layout because it is known to have poor intra-job communication performance due to interference.

When PLAS is asked to allocate a new job, it scans for idle compute nodes in a predefined scanning order (see Fig. 9), always starting from the same "middle" point. Applying our results about optimal shape of allocations, PLAS scans parallel to the I/O nodes, one column after another. In each column, PLAS scans "middle-out". In the example of Fig. 9, a job requesting six compute nodes would get the compute nodes labeled 1 through 6 if these are currently idle. Until it has found enough idle nodes to fulfill the request, PLAS keeps on searching according to the scanning order. Pseudocode is shown in Fig. 10.

We conducted extensive simulation experiments to compare PLAS to two well-established allocation strategies:

- **MBS** [22] is a block-based strategy that typically allocates several blocks of sizes $2^i \times 2^i$. MBS is similar to the M2DB strategy[42] used at the San Diego Supercomputing Center.
- **Paging** [22] scans for idle compute nodes in row-major

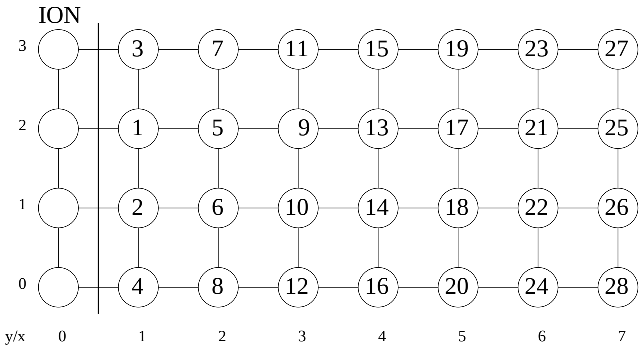


Figure 9: Scanning order of PLAS (parallel layout allocation strategy)

```
plas_allocate(jobsize){
  initialize column, middle, offset and sign
  while (jobsize > 0) {
    node_x = column; node_y = middle + sign * offset
    if node idle then allocate node; jobsize--
    else {
      flip sign
      every second time increase offset
      every mesh_height time increase column and reset offset
  }
}
```

Figure 10: Pseudocode of PLAS

	jobsize	interarrival time	# jobs
I: synthetic	exponential mean $= 4 * 4$	exponential mean varies	1000
II: real (SDSC)	mean $= 14.5$ c.v. $= 1.5$	mean varies c.v. $= 3.7$	6087

Table 4: Jobsize, interarrival time and number of jobs of the synthetic workload and the real workload trace

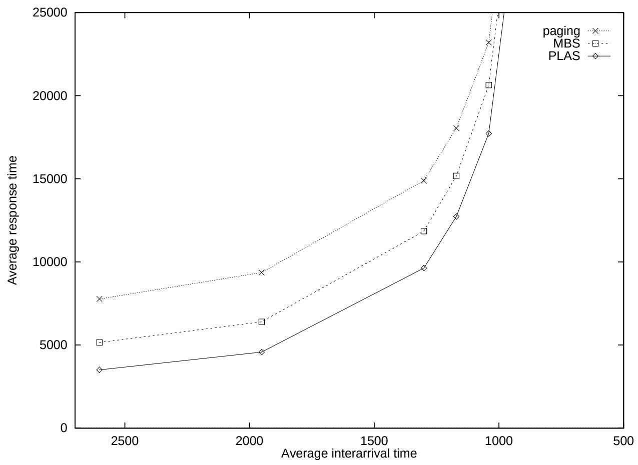


Figure 11: Average response time (6087 I/O intensive jobs)

order, and thus typically allocates several rows of compute nodes. Paging was shown to have very good performance for computation or communication-intensive workloads [22] and is used at NASA Ames Research Center.

Our simulation experiments were conducted using the newest version of ProcSimity [44], a simulation tool for evaluating job scheduling and processor allocation algorithms for distributed memory parallel machines.

We model wormhole switching, minimal dimension- ordered XY routing and mesh topologies with two uni- directional links between adjacent nodes. To allow for both a realistic evaluation and for easy comparison with previous experiments, our performance evaluation includes the use of a synthetic workload and a real workload trace. As summarized in Table 4, Workload I is a synthetic workload, a stream of 1000 jobs whose sizes and interarrival times are exponentially distributed. Workload II is a trace-derived workload, a stream of 6087 real production jobs recorded over a three months period from October 1 to December 31, 1996, from the Intel Paragon at the San Diego Supercomputer Center. The traced job stream is taken only from the 352 node NQS partition [43] of the machine, through which all batch jobs were scheduled. The trace had the following statistical characteristics: the mean interarrival times was 1301 seconds, with a coefficient of variance of 3.7; the average job size was 14.5 nodes, with a coefficient of variance of 1.5.

To challenge these allocation strategies, we multiply job arrival times by constant factors c. As c decreases, simulated interarrival times decrease, resulting in increased system load. As c increases, simulated interarrival times increase and system load decreases. The simulated interarrival time is chosen such that average system utilization ranges from about 10% to about 90%.

Since our focus is on traffic hotspots due to parallel I/O traffic, we model a heavy I/O load, no computation and no communication. For both Workload I and Workload II, job service time depends on the amount of traffic generated and the amount of network contention encountered. Each job transfers data according to a complete bipartite I/O pattern, i.e. each compute node allocated to the job sends a personalized message to every I/O node. The number of messages sent is correlated to jobsize because we have each job execute the same number of iterations of the complete bipartite I/O pattern before it leaves the system.

In the results reported, we model a mesh of 16×22 compute nodes, matching the batch partition of the Intel Paragon at SDSC. We assume a configuration where 16 I/O nodes are located vertically on the west side of the compute nodes. We assume that compute nodes write to I/O nodes (i.e. write traffic only) since our analysis showed that the hotspots due to write traffic are much more severe than the hotspots due to read traffic if I/O nodes are placed vertically. We simulate the detailed message-passing behavior at the level of packets.

Fig. 11 shows average response times for Workload I. For the SDSC workload, PLAS results in up to a factor of 2.2 speedup in average response time compared to previous non-contiguous allocation strategies that were not parallel I/O sensitive. For Workload II, Figs. 12, 13 and 14 show average response times, average service time and average max_contention, respectively. For the synthetic workload, PLAS results in up to a factor of 4.5 speedup in average response time. Clearly, tuning the spatial layout of compute nodes resulted in impressive performance gains.

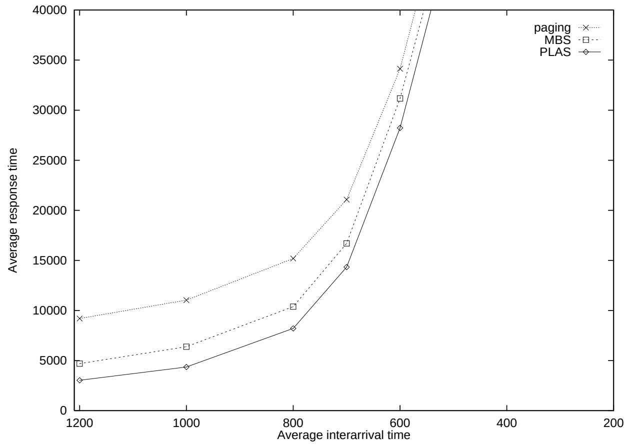


Figure 12: Average response time (synthetic workload of 1000 I/O-intensive jobs)

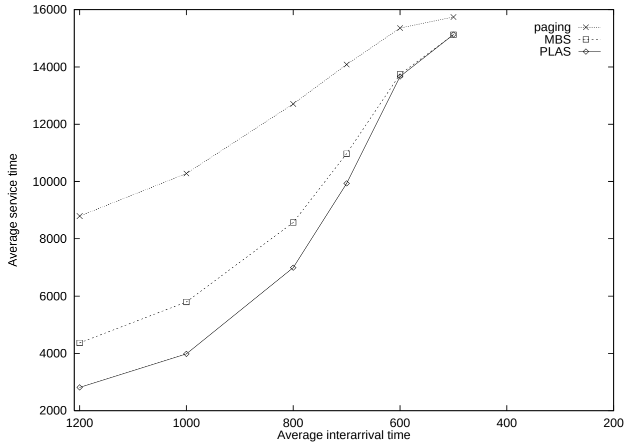


Figure 13: Average service time (synthetic workload of 1000 I/O-intensive jobs)

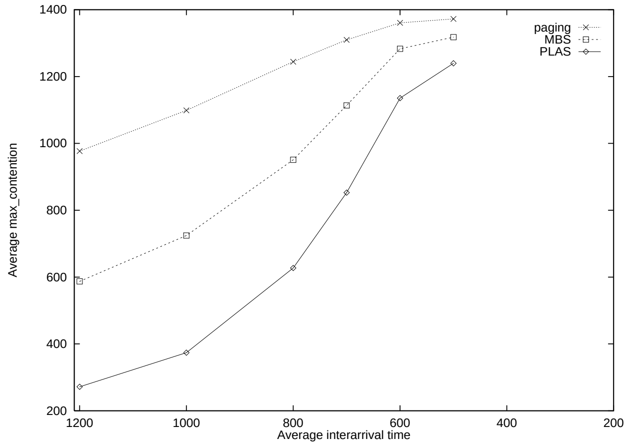


Figure 14: Average *max_contention* (synthetic workload of 1000 I/O-intensive jobs)

We plan to test PLAS on communication-intensive workloads and possibly refine it further such that it also accommodates the sometimes conflicting need to minimize contention due to intra-job communication.

7 Conclusions and Future Work

In this paper we have reported on the performance sensitivity of parallel I/O on mesh architectures such as the Intel TFLOPS ASCI/Red. Our analytic results together with extensive simulations lead us to conclude the following:

- *Network contention* can limit parallel I/O performance. On a 2D mesh with I/O nodes on one side, realizable I/O throughput is – in the best case – limited by 4 times the effective bandwidth of the interconnection network, B_l. Thus, adding I/O nodes (of effective bandwidth B_0) does not pay beyond $4 * B_l/B_0$. This is a much tighter upper bound on the number of I/O nodes than $n * B_l/B_0$ [11] which did not take network contention into account.
- *Spatial layout of jobs* in relation to the I/O nodes affects network contention and thus parallel I/O performance. If the I/O nodes are on one side of the mesh, tuning the spatial layout of jobs makes a difference of up to a factor of 4. If network throughput is the bottleneck, spatial layout must be taken into account.
- *Processor allocation strategies* that are sensitive to parallel I/O traffic can improve I/O throughput significantly. We devised a new strategy that improved the average response time of I/O bound jobs by up to a factor of 4.5 in performance evaluations driven by synthetic workloads and by a real workload trace captured at the San Diego Supercomputing Center.
- One direction of parallel I/O traffic performs much worse than the other direction. If the I/O nodes are placed vertically, write traffic (to the I/O nodes) is the bottleneck; if the I/O nodes are placed horizontally, read traffic is the bottleneck. If processor allocation strategies are sensitive to whether a job is read or write dominant, further improvements should be possible.

This study presents guidelines for new resource management policies and represents a first step in the development of future scheduling, allocation, and I/O placement policies that aim at improving parallel I/O performance. Our continuing and future work in this area looks beyond some of the constraints assumed in this study.

- *Investigation of alternate I/O placement schemes.* We are looking at I/O placement schemes such as two columns on both sides of the mesh or one column in the middle of the mesh; and at distributed I/O node placement schemes, such as those reported in [1, 41, 36, 21, 37]. These schemes all hold promise because they reduce I/O bottlenecks by distributing I/O traffic over more network links.
- *Balancing I/O traffic and traffic due to intra-job communication.* Spatial layouts that are optimal for I/O traffic are often undesirable with respect to intra-job communication because they can cause the messages from different jobs to interfere with each other. We are

analyzing the effect scheduling and allocation strategies have on both I/O traffic and intra-job communication with the goal of developing new algorithms that can accommodate both. The workloads used to drive our simulation experiments are being expanded to model I/O, communication, and computation, rather than focusing exclusively on I/O.

- *Investigation of other network topologies.* We will look at multi-dimensional meshes, tori, and hypercubes and at indirect networks such as that used in the IBM SP/2, extending the work of [3]. These topologies are not exclusive to massively parallel machines since workstation clusters have recently been configured with regular interconnections [9].

References

[1] M. Bae and B. Bose. Resource placement in torus-based networks. *IEEE Transactions on Computers*, 46(10):1083–1092, October 1997.

[2] David H. Bailey, Eric Barszcz, Leonardo Dagum, and Horst D. Simon. NAS parallel benchmark results. *IEEE Parallel and Distributed Technology*, 1(1):43–51, February 1993.

[3] Sandra Johnson Baylor, Caroline Benveniste, and Yarsun Hsu. Performance evaluation of a massively parallel I/O subsystem. In Ravi Jain, John Werth, and James C. Browne, editors, *Input/Output in Parallel and Distributed Computer Systems*, volume 362 of *The Kluwer International Series in Engineering and Computer Science*, chapter 13, pages 293–311. Kluwer Academic Publishers, 1996.

[4] S. H. Bokhari and D. M. Nicol. Balancing contention and synchronization on the Intel Paragon. *IEEE Concurrency*, 5(2):74–83, 1997.

[5] Y. Cho, M. Winslett, M. Subramaniam, Y. Chen, S. Kuo, and K. E. Seamons. Exploiting local data in parallel array I/O on a practical network of workstations. In *Proceedings of the 5th Workshop on I/O in Parallel and Distributed Systems, SC '97*, pages 1–13, 1997.

[6] S. P. Dandamudi and H. Hadavi. Performance impact of I/O on sender iniatiated and receiver initiated adaptive load sharing in distributed systems. In *Proceedings of the International Conference on Parallel and Distributed Computing Systems*, 1996.

[7] Juan Miguel del Rosario, Rajesh Bordawekar, and Alok Choudhary. Improved parallel I/O via a two-phase run-time access strategy. In *Proceedings of the IPPS '93 Workshop on Input/Output in Parallel Computer Systems*, pages 56–70, Newport Beach, CA, 1993. Also published in Computer Architecture News 21(5), December 1993, pages 31–38.

[8] Andreas Eberhart and Jingke Li. Contention-free communication scheduling on 2D meshes. In *Proceedings International Conference on Parallel Processing*, 1996.

[9] Dong Chen et. al. QCDSP machines: Design, performance and cost. http://www.supercomp.org/sc98/awards/#Gordon Bell.

[10] D. G. Feitelson. A survey of scheduling in multiprogrammed parallel systems. Technical Report RC 19790 (87657), IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598, October 1994.

[11] Dror G. Feitelson, Peter F. Corbett, Sandra Johnson Baylor, and Yarson Hsu. Parallel I/O subsystems in massively parallel supercomputers. *IEEE Parallel and Distributed Technology*, 3(3):33–47, Fall 1995.

[12] Sharad Garg. TFLOPS PFS: Architecture and design of a highly efficient parallel file system. In *Proceedings of Supercomputing '98*, 1998.

[13] Sharad Garg, Robert Godley, Richard Griffiths, Andrew Pfiffer, Terry Prickett, David Robboy, Stan Smith, T. Mack Stallcup, and Stephen Zeisset. Achieving large scale parallelism through operating system resource management on the Intel TFLOPS supercomputer. *Intel Technology Journal*, 1st quarter 1998.

[14] Garth A. Gibson, Jeffrey Scott Vitter, and John Wilkes. Strategic directions in storage I/O issues in large-scale computing. *ACM Computing Surveys*, 28(4):779–793, December 1996.

[15] Ravi Jain, John Werth, and James C. Browne, editors. *Input/Output in Parallel and Distributed Computer Systems*, volume 362 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, 1996.

[16] C. Koelbel, D. Loveman, R. Schreiber, G. Steele, and M. Zosel. *High Performance Fortran Handbook*. MIT Press, 1994.

[17] David Kotz. Disk-directed I/O for MIMD multiprocessors. In *Proceedings of the 1994 Symposium on Operating Systems Design and Implementation*, pages 61–74. USENIX Association, November 1994. Updated as Dartmouth TR PCS-TR94-226 on November 8, 1994.

[18] Tomas Lang and Lance Kurisaki. Nonuniform traffic spots (nuts) in multistage interconnection networks. *Journal of Parallel and Distributed Computing*, 10:55–67, 1990.

[19] Walter Lee, Matthew Frank, Victor Lee, Kenneth Mackenzie, and Larry Rudolph. Implications of I/O for gang scheduled workloads. In *Proceedings of the 3nd Workshop on Job Scheduling Strategies for Parallel Processing, IPPS '97*, 1997.

[20] K. Li and K.-H. Cheng. A two-dimensional buddy system for dynamic resource allocation in a partitionable mesh connected system. *Journal of Parallel and Distributed Computing*, 12:79–83, 1991.

[21] Marilynn Livingston and Quentin Stout. Distributing resources in hypercube computers. In *Proceedings of the 3rd Conference on Hypercube Concurrent Computers and Applications*, 1988.

[22] V. M. Lo, K. Windisch, W. Liu, and B. Nitzberg. Non-contiguous processor allocation algorithms for mesh-connected multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, 8(7):712–726, July 1997.

[23] Jens Mache. *Parallel I/O- and Communication-Sensitive Scheduling on High-Performance Parallel Computers.* PhD thesis, Department of Computer and Information Science, University of Oregon, March 1999.

[24] Jens Mache and Virginia Lo. The effects of dispersal on message-passing contention in processor allocation strategies. In *Proceedings of the 3rd Joint Conference on Information Sciences, Volume 3, Sessions on Parallel and Distributed Processing*, 1997.

[25] Jens Mache, Virginia Lo, and Kurt Windisch. Minimizing message-passing contention in fragmentation-free processor allocation. In *Proceedings of the 10th International Conference on Parallel and Distributed Computing Systems*, 1997.

[26] Timothy G. Mattson, David Scott, and Stephen Wheat. A TeraFLOP supercomputer in 1996: the ASCI TFLOP system. In *Proceedigs of the International Parallel Processing Symposium*, 1996.

[27] D. Min and M. W. Mutka. A multipath contention model for analyzing job interaction in 2-D mesh multicomputers. In *Proceedings of the 8th International Parallel Processing Symposium*, pages 744–751, April 1994.

[28] Bernd Mohr. Processor allocation on the Cray T3E. Personal communication, 1998.

[29] S. Q. Moore and L. M. Ni. The effects of network contention on processor allocation strategies. In *Proceedings of the 10th International Parallel Processing Symposium*, pages 268–274, 1996.

[30] Csaba A. Moritz and Matthew I. Frank. LoGPC: Modeling network contention in message-passing programs. In *Proceedings of Sigmetrics*, 1998.

[31] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Transactions on Computers*, pages 62–76, February 1993.

[32] Bill Nitzberg. Processor allocation on the Intel Paragon. Personal communication, 1998.

[33] Bill Nitzberg and Virginia Lo. Collective buffering: Improving parallel I/O performance. In *Proceedings of the Sixth IEEE International Symposium on High Performance Distributed Computing*, pages 148–157, Portland, OR, August 1997. IEEE Computer Society Press.

[34] William J. Nitzberg. *Collective Parallel I/O.* PhD thesis, Department of Computer and Information Science, University of Oregon, December 1995.

[35] James C. T. Pool and Paul Messina. Scalable I/O initiative. http://www.cacr.caltech.edu/SIO/.

[36] P. Ramananthan and S. Chalasani. Resource placement with multiple adjacency constraints in k-ary n-cubes. *IEEE Transactions Parallel and Distributed Systems*, 6(5):511–519, May 1995.

[37] A. L. N. Reddy, P. Banerjee, and S. G. Abraham. I/o embedding in hypercubes. pages 331–338, August 1988.

[38] Emilia Rosti, Giuseppe Serazzi, Evgenia Smirni, and Mark S. Squillante. The impact of I/O on program behavior and parallel scheduling. In *Proceedings of Sigmetrics*, 1998.

[39] K. E. Seamons, Y. Chen, P. Jones, J. Jozwiak, and M. Winslett. Server-directed collective I/O in Panda. In *Proceedings of Supercomputing '95*, San Diego, CA, December 1995. IEEE Computer Society Press.

[40] K. Suzaki, H. Tanuma, S. Hirano, Y. Ichisugi, C. Connelly, and M. Tukamoto. Multi-tasking method on parallel computers which combines a contiguous and a non-contiguous processor partitioning algorithm. In *Lecture Notes in Computer Science 1184*, pages 641–650, 1996.

[41] N. F. Tzeng and G. L. Feng. Resource allocation in cube network systems based on the covering radius. *IEEE Transactions Parallel and Distributed Systems*, 7(4):328–342, April 1996.

[42] M. Wan, R. Moore, G. Kremenek, and K. Steube. A batch scheduler for the Intel Paragon MPP system with a non-contiguous node allocation algorithm. In *Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing, IPPS '96*, 1996.

[43] K. Windisch, V. Lo, D. Feitelson, B. Nitzberg, and R. Moore. A comparison of workload traces from two production parallel machines. In *Proceedings of the Sixth Symposium on the Frontiers of Massively Parallel Computation*, 1996.

[44] K. Windisch, J. V. Miller, and V. Lo. ProcSimity: an experimental tool for processor allocation and scheduling in highly parallel systems. In *Proceedings of the Fifth Symposium on the Frontiers of Massively Parallel Computation*, pages 414–421, 1995.

[45] Y. Zhu. Efficient processor allocation strategies for mesh-connected parallel computers. *Journal of Parallel and Distributed Computing*, 16:328–337, 1992.