# Job Scheduling that Minimizes Network Contention due to both Communication and I/O

Jens Mache
Lewis & Clark College
jmache@lclark.edu

Virginia Lo
University of Oregon
lo@cs.uoregon.edu

Sharad Garg
Intel Corp.
sharad@co.intel.com

## Abstract

*As communication and I/O traffic increase on the interconnection network of high-performance systems, network contention becomes a critical problem drastically reducing performance. Whereas earlier allocation strategies were either sensitive to communication alone or sensitive to I/O alone, we present a new strategy that is sensitive to both communication and I/O. Our new strategy, MC-Elongated, strives to achieve (1) the compactness needed to minimize communication-based contention as well as (2) the balance and orientation relative to I/O nodes needed to minimize I/O-based contention.*

*We tested our new strategy using synthetic workloads and a real workload trace of 6087 jobs captured from a 400 node Intel Paragon. Our results show that with respect to system throughput and average job turnaround time, in environments with varying degree of communication and I/O traffic, MC-Elongated outperforms previous allocation strategies that are in use today. Regarding the tension between communication and I/O, our results show that spatial layout is more critical for I/O-intensive jobs at lower utilization levels and more critical for communication-intensive jobs at higher utilization levels; and that in general, the impact of I/O traffic is dominant.*

## 1. Introduction

The data and I/O demands of applications in many fields – especially in computational science, visualization, database, and multimedia – are increasing at an unprecedented pace. As a consequence, on high-performance parallel systems, vast amounts of data must be moved – both between compute nodes and I/O nodes, and among compute nodes. A crucial challenge is effective and efficient use of the shared medium for data transfer: the interconnection network. As the traffic on the links in the network increases, *network contention* becomes a critical problem, causing significant delays in communication traffic [2, 8, 13] (traffic of different jobs interferes) and degrading I/O performance [1, 4, 7] (traffic is distributed unevenly causing hot spots).

Our earlier analysis [8, 12, 5, 11] and that of others [3] demonstrated that the *spatial layout* of jobs can significantly affect network contention and ultimately overall performance. Spatial layout refers to the location of a job's compute nodes relative to each other and relative to the location of the I/O nodes. We have shown that allocating jobs as "compactly" as possible serves to minimize network contention due to interprocess communication [12], while "balancing" compute nodes relative to the I/O nodes is effective at minimizing contention due to I/O traffic [10]. Since these two goals (compactness and balance) are very different, our challenge was to design a strategy that achieves both goals and thus excels in environments with a mixture of communication and I/O traffic.

In this paper, we present a new allocation strategy, MC-Elongated, that minimizes network contention due to *both* interprocess communication and parallel I/O traffic. MC-Elongated strives to achieve (1) the compactness needed to minimize communication-based contention as well as (2) the balance and orientation relative to I/O nodes needed to minimize I/O-based contention. Moreover, MC-Elongated is lazy (instead of eager) at filling holes and considers all candidates in a best fit manner, in contrast to earlier I/O-sensitive strategies which choose candidates in a first fit manner.

We tested the performance of MC-Elongated by running extensive simulations using both synthetic workloads and a real workload trace of 6087 jobs captured from a 400 node Intel Paragon. Our results show dramatic performance improvements with respect to system throughput and average turnaround time compared to current allocation strategies. These results call for new allocation strategies to be used within high-performance schedulers for the computational grid or within scheduling tools such as PBS, EASY, Load Leveller and PSched.

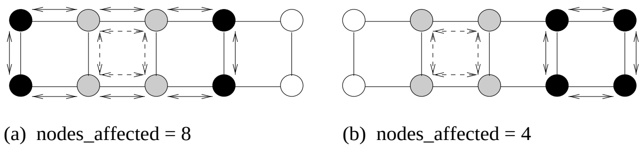

**Figure 1. An example showing that compact allocations minimize interference**

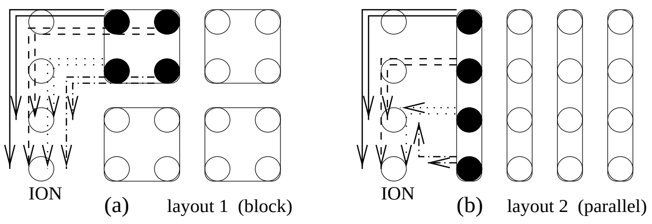

**Figure 2. An example showing that balanced allocations minimize hot spots**

## 2 Background and previous results

### 2.1 Job scheduling

Most high-performance parallel systems employ *space sharing* with *variable partitioning*: multiple jobs run concurrently on disjoint subsets of the compute nodes, and each job is assigned the number of compute nodes requested by the job, using only those nodes for the lifetime of the job. Given a workload of jobs, the system scheduler has to decide when to run each job and which compute nodes to allocate to each job.

In this study, we concentrate on the *allocation* phase of job scheduling which decides which specific processors to assign to a given job. It is the spatial layout of the allocated compute nodes – in relation to other jobs and in relation to the I/O nodes – that is the focus of our new strategy. Previous allocation strategies have targeted only communication [14, 8, 12] or only I/O [3, 11]. The problem we address is the development of a new allocation strategies that is sensitive to both communication and I/O.

The architecture assumed is a high-performance parallel system in which compute nodes are connected via a direct interconnection network. The I/O subsystem consists of disks (or RAIDs) attached to I/O nodes. The I/O nodes (ION) 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 data transfer in parallel. In this study, we concentrate on mesh-interconnected direct networks and dimension-ordered XY-routing. We assume that the I/O nodes are fixed at one side of the mesh. These assumptions are motivated by our earlier measurements on an Intel TFLOPS [5, 6] and by driving our simulations with a real workload trace taken from an Intel Paragon.

### 2.2 Compact allocations minimize communication-based contention

We first show how network contention due to interprocess communication is related to the spatial layout of a job's compute nodes. As can be seen from Fig. 1, the more *compact* the layout, the less potential network contention due to inter-job link contention. In Fig. 1(b), job A is allocated compactly, yielding no contention between its messages and job B's messages. In Fig. 1(a), however, job A's compute nodes are spread out, giving rise to contention with job B's messages.

In [12], we defined a metric that captures the compactness of a job's allocated compute nodes. We defined *nodes_affected* as the number of nodes contained in the smallest rectangle that encloses all nodes of a job. *Nodes_affected* was shown to have very high correlation with dynamic levels of communication-based network contention and thus can be used effectively by allocation strategies whose goal is minimization of network contention due to interprocess communication. In [12], we also designed and tested a strategy we call *MC* (Minimizing Contention) which was highly successful at allocating jobs compactly and thereby minimizing communication-based network contention while achieving fast average turnaround times and high system utilization [1].

### 2.3 Balanced allocations minimize I/O-based contention

The relationship between I/O-based network contention and the spatial layout of compute nodes relative to I/O nodes is illustrated in Fig. 2. This figure shows a job of size 4 in a 4x4 compute mesh with 4 I/O nodes sitting on one side, under the heaviest I/O traffic pattern (complete bipartite). Fig. 2(a) and (b) show the write traffic from the four compute nodes to the lower two I/O nodes. In (a) with a *block-based layout*, 8 out of 16 messages contend for the downward link in the middle of the I/O nodes; in (b) with a *parallel layout*, only 4 out of 16 messages overlap.

In our earlier work [11] we showed that the spatial layout of compute nodes relative to I/O nodes directly affects network contention. In a mesh-based architecture with the I/O nodes placed parallel to the Y axis, the bottleneck for I/O write traffic occurs on the middle I/O link, i.e., the link

[1] Note that MC outperformed all existing allocation strategies in use at the time because it found compact *non-contiguous* layouts. Earlier allocation strategies allocated all nodes from one job compactly in a rectangle. However, these *contiguous* allocation strategies suffered from poor average turnaround time and low utilization due to fragmentation.

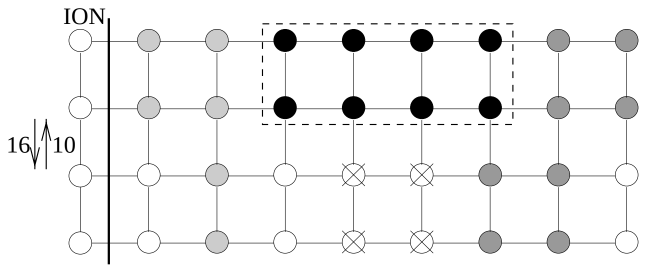

**Figure 3. A communication-sensitive layout for job A that is compact but unbalanced.**

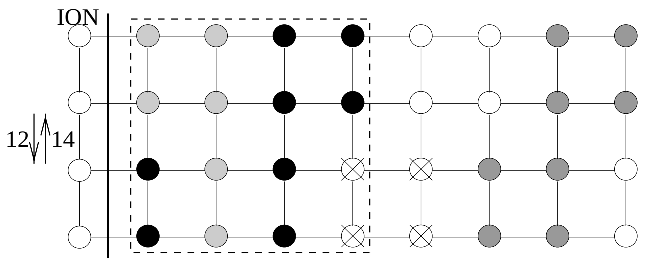

**Figure 4. An I/O-sensitive layout for Job A that is balanced but not compact.**

between the I/O nodes in rows $k/2$ and $k/2+1$ (assuming $k$ even). As a result, we proved that the best layout of compute nodes is "balanced" around the middle I/O node and oriented parallel to the I/O nodes.

In [10], we defined a metric we call *balance factor* to capture the degree to which a job's compute nodes are balanced around the middle I/O node. Intuitively, system balance factor is the absolute difference of number of nodes currently allocated in rows above the middle I/O link and number of nodes currently allocated in rows below the middle I/O link. Ideally, system balance factor is 0. In Fig. 2 system balance factor is 4 for layout 1, and 0 for layout 2. We used balance factor in a strategy called *PLAS* which was very effective at minimization of network contention due to I/O-based traffic.

## 3 Allocation strategies

### 3.1 Previous strategies

Previous allocation strategies are either sensitive to communication alone or sensitive to I/O alone. *Communication-sensitive* allocation strategies allocate blocks of compute nodes that are usually compact and rectangular in shape in order to minimize communication-based contention. However, without concern for where jobs lie relative to the I/O nodes, *unbalanced* layouts result. This yields contention due to I/O traffic. Fig. 3 shows how the communication-sensitive strategy MC [12] allocates a job of size 8. The allocation is compact but unbalanced. Balance factor is 6 (there are 16 allocated compute nodes in the upper half and 10 in the lower half), and parallel I/O traffic would thus be limited by the hot spot on the downward link between the middle I/O nodes.

In contrast, *I/O-sensitive* strategies focus on balance in order to minimize I/O-based contention. However, these strategies typically produce layouts that are *not compact*. This yields contention due to interprocess communication. Fig. 4 shows how PLAS [11, 10] allocates a job of size 8.

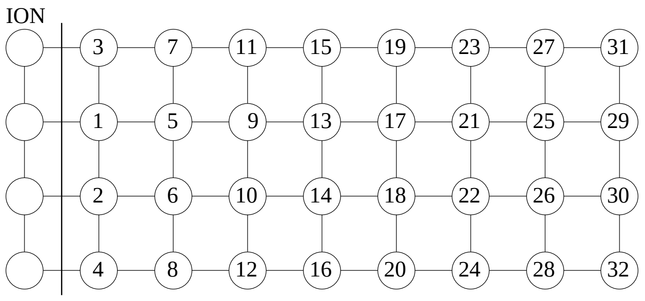

**Figure 5. Scanning order of PLAS**

(PLAS always scans for idle nodes according to the fixed order shown in Fig. 5.) This allocation is balanced but not compact. *Nodes_affected* is 16, and interprocess communication traffic would contend with interprocess communication traffic of two other jobs.

Extensive simulations show clearly that strategies that address one form of network contention suffer with respect to the other. Thus, we developed MC-Elongated to address this need.

### 3.2 MC-Elongated

#### 3.2.1 Mechanism

MC-Elongated strives to achieve (1) the compactness needed to minimize communication contention as well as (2) the balance and orientation relative to I/O nodes needed to minimize I/O-based network contention. To achieve compactness, MC-Elongated does *concentric searches* to find and evaluate candidate layouts. This mechanism was successfully used in the MC strategy [12]. To achieve balance, MC-Elongated tries to (1) balance each job and (2) re-balance the system (after job departures) by filling holes. These mechanisms were successfully used in the PLAS strategy [11, 10]. To do so, MC-Elongated concentrates on *whole columns*: Each concentric search starts with whole columns and extends by one column at a time.

```
mc-elongated_allocate(jobsize){
  for each column i
    candidate = empty list; tcost = 0; s = 0
    add idle nodes to candidate that are in
      jobsize/sidelength columns around i
    while |candidate| < jobsize
      search shell_(++s)
      if node idle {add to candidate; tcost+=s}
  select candidate with minimal tcost
  allocate those nodes, update busy array
}
```

**Figure 6. Pseudocode of MC-Elongated**

### 3.2.2 Description and example

We discuss MC-Elongated as applied to two-dimensional meshes of sidelength $k$. Assume that the chosen job requests $j$ compute nodes. The MC-Elongated algorithm first constructs several *candidate layouts*, one per column $i$, and then selects the "best" candidate layout. A candidate layout is centered around column $i$ and contains idle nodes arranged within *shells* radiating from the center out. For the purpose of defining shells, we disregard whether nodes are idle (unallocated) or busy (allocated to other jobs).

Definition: For a given column $i$ and a request size $j$, $shell_0$ is the contiguous rectangle of $\lfloor \frac{j}{k} \rfloor$ columns centered around column $i$, i.e. the $k \times \lfloor \frac{j}{k} \rfloor$ rectangle starting at column $i - \lfloor \frac{j}{2k} \rfloor$. $Shells_{s+1}$, $s \geq 0$ are successive rectangular rings of nodes. More precisely, $shell_{s+1}$ contains the nodes that are at distance 1 from at least one node in $shell_s$, but not contained in any $shell_k$, $0 \leq k \leq s$, where distance is defined as $max(\Delta x, \Delta y)$.

To construct a candidate layout, idle nodes are selected from successive shells, beginning with $shell_0$, until enough idle nodes have been selected to satisfy the request.

To evaluate a candidate layout, we define cost in the following way.

Definition: The cost of a node is equal to the value of that node's shell number. The cost of a candidate layout is the sum of the costs of the nodes in that layout.

The candidate layout with minimal total cost is allocated. Pseudocode for the algorithm is given in Fig. 6.

We illustrate the algorithm given the situation in Fig. 7 and a job request of 8. One idle node from each column $i$ does a concentric search to find a compact layout of 8 nodes with column $i$ as the center. Fig. 7 shows the shells for column 5. $Shell_0$ consists of columns 5 and 6. However, only two nodes are idle in $shell_0$, thus MC-Elongated keeps searching through bigger shells until enough idle nodes are found. $Shell_1$ extends $shell_0$ by one ring of nodes, and so on. In the end, the candidate layout centered around column 5 includes 8 idle nodes for a total cost of $0+0+1+1+1+1+2+2=8$. Similarly, the candidate layouts centered

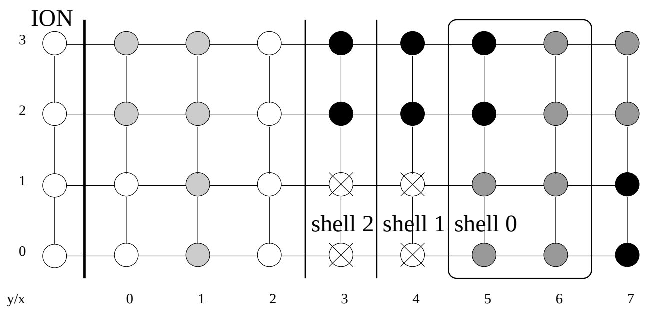

**Figure 7. How MC-Elongated constructs the candidate layout for column 5**

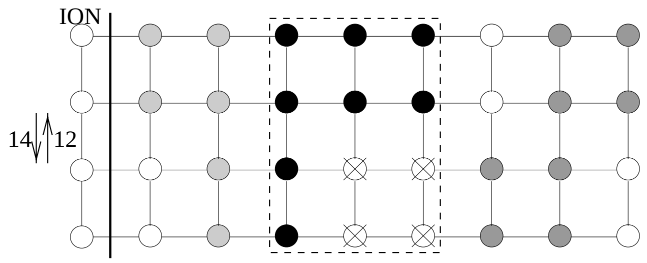

**Figure 8. The spatial layout chosen by MC-Elongated is both compact and balanced.**

around columns 0, 1, 2, 3, 4, 6 and 7 have total cost of 8, 4, 2, 4, 6, 12 and 18, respectively. The layout centered around column 2 (see Fig. 8) has minimal cost and will be allocated. This spatial layout has a balance factor of 2 and *nodes_affected* of 12. Good I/O throughput and interprocess communication can be expected.

### 3.2.3 Characteristics and comparison

MC-Elongated has the following four main characteristics:

Property 1: MC-Elongated is neither conservative nor eager at filling holes.

This is in contrast to PLAS, Fig. 4 showed that PLAS is conservative in that it first fills the holes before it utilizes columns further to the right. PLAS and earlier strategies like MBS [8] are "eager" in picking up small leftovers before touching bigger idle spaces, in anticipation of saving the bigger space for a bigger job that may be coming later. This conservative attitude almost always results in less compact allocations, and in many situations, it does not even help the allocation of future jobs. The conservative approach has no advantage if either the preserved block will be split up anyways by a later job that is not that big or if the preserved block will not be used anyways because either no big job arrives before other jobs leave or the next big job is too big for the preserved block anyways. The approach of

MC-Elongated is more along the line of *carpe diem*, seize the day.

Property 2: MC-Elongated considers candidates in a best fit manner.

One idle compute node in each column executes a concentric search to find a candidate layout. The candidate layout with minimal total cost is allocated. This is in contrast to PLAS which chooses candidates in a first fit manner.

MC-Elongated's algorithm is still efficient. Its complexity is $n^{1.5}$ in a mesh of $n$ compute nodes. This overhead is acceptable for the following two reasons. First, is has been observed [9] that interarrival times of jobs are typically on the order of minutes and runtimes of jobs are on the order of hours. In this context, spending a short amount of time to find a good spatial layout is very worthwhile. Second, MC-Elongated is inherently parallelizable. One idle node per column can independently find and evaluate a candidate layout. To allocate a new job, a distinguished system node sends the busy array, a bitvector representing the status of each node, to each of those idle node and – after collecting the total cost (and node selections) of the candidate layouts – allocates the job to the best candidate layout. Thus, we employ just idle nodes and need only limited communication in the form of a scatter-gather pattern.

Property 3: MC-Elongated finds compact candidate layouts.

Like MC, MC-Elongated never leaves idle nodes unselected on communication paths between selected nodes. This is due to the concentric searches and the definition of cost. This characteristic leads to compact allocations, lower inter-job link contention and good performance under communication-intensive workloads.

However, while MC always allocates a contiguous layout if one exists, MC-Elongated sacrifices this property in order to achieve better balance. Fig. 3 shows an example where MC allocates a contiguous layout (achieving the highest possible degree of compactness), whereas MC-Elongated (see Fig. 8) allocates a layout that is less compact, possibly causing inter-job link contention. Here we trade-off compactness for more balance.

Property 4: MC-Elongated achieves balance.

Like PLAS, starting with an empty system and allowing no job departures, MC-Elongated achieves perfect per job balance factors and perfect system balance factors. To do so, the order in which idle nodes are selected within each shell is "middle out" (as in PLAS). This is in contrast to MC, where the scanning order (shorter sides first, longer sides second, corners last) was tuned for optimal degree of compactness.

However, since MC-Elongated is only lazy at filling holes, it does not re-balance the system as quickly as PLAS, in general situations including job departures. Here we trade-off balance for more compactness.

**Table 1. Jobsize, interarrival time and number of jobs of the synthetic workload and the real workload trace**

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

A further characteristics of MC-Elongated is that it easily extends to higher dimensional and toroidal topologies because extending both the concentric search and the definition of shells is straightforward.

## 4 Performance evaluation

### 4.1 Experimental method

System Model: In the results reported, we model a mesh of $16 \times 22$ compute nodes, matching the batch partition of the Intel Paragon at SDSC [9]. We assume a configuration where 16 I/O nodes are located vertically on the west side of the compute nodes. We model wormhole switching, minimal dimension-ordered XY routing and mesh topologies with two uni-directional links between adjacent nodes. All message-passing is modeled at the packet-level.

Workload: Our performance evaluation includes the use of a synthetic workload and a real workload trace. The characteristics of these workloads are summarized in Table 1. Workload II is a trace-derived workload recorded over a three month period from an Intel Paragon at the San Diego Supercomputer Center. The traced job stream is taken only from the 352 node NQS partition [9] of the machine, through which all batch jobs were scheduled. To challenge the allocation strategies, we speed up job arrivals along the x-axis such that average system utilization ranges from about 10% to about 90%.

I/O and Communication Traffic: We model both heavy I/O traffic and heavy communication traffic, i.e. complete bipartite I/O and all-to-all broadcast communication. For I/O traffic for a given job, each compute node allocated to that job sends messages to each of the I/O nodes. The ratio of I/O traffic to communication traffic was varied from 100% I/O to 0% I/O.

Allocation Strategies: We compare MC-Elongated to MC (communication-sensitive), to PLAS (I/O-sensitive) and to two well-established allocation strategies: (1) MBS [8], a block-based strategy very similar to the M2DB strategy[14] that has been used at the San Diego Supercomputing Center, and (2) Paging [8] that produces allocations that are perpendicular to the I/O nodes. Paging

**Table 2. Average service time (t) and ranking (r) for different ratios of I/O traffic vs. communication traffic (interarrival times are chosen such that system utilization is between 60% and 70%)**

| traffic ratio | 100% I/O | | 80% I/O | | 60% I/O | | 40% I/O | | 20% I/O | | 0% I/O | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| interarrival time | 500 | | 400 | | 350 | | 250 | | 125 | | 20 | |
| | t | r | t | r | t | r | t | r | t | r | t | r |
| MC-Elongated | 8529 | 3 | 6693.6 | 2 | 4948.7 | 1 | 2784.0 | 1 | 1405.0 | 1 | 395.3 | 2 |
| PLAS | 8140 | 1 | 6384.6 | 1 | 4929.5 | 1 | 2847.9 | 2 | 1470.9 | 2 | 501.9 | 3 |
| MC | 9076 | 4 | 7265.1 | 4 | 5481.7 | 3 | 3258.8 | 3 | 1603.4 | 3 | 366.0 | 1 |
| Random | 8488 | 2 | 7046.6 | 3 | 5563.9 | 4 | 3334.9 | 4 | 1810.0 | 4 | 738.9 | 6 |
| MBS | 9939 | 5 | 8072.3 | 5 | 6004.2 | 5 | 3890.3 | 5 | 2085.2 | 5 | 651.4 | 5 |
| Paging | 12096 | 6 | 9653.6 | 6 | 6911.8 | 6 | 4341.1 | 6 | 2225.7 | 6 | 514.5 | 4 |

was shown to have very good performance for computation or communication-intensive workloads [8], and it has been used at NASA Ames Research Center.

## 4.2 Results

We conducted experiments using the newest version of ProcSimity [9] to compare MC-Elongated to the other non-contiguous allocation strategies. We varied the percentage of I/O traffic vs. communication traffic from 100% I/O traffic to 0% I/O traffic in increments of 25% for the SDSC workload (and in increments of 20% for the synthetic workload).

Table 2 summarizes average service time and ranking for the synthetic workload. Interarrival times are chosen such that system utilization is between 60% and 70%. While MC-Elongated is not as good as PLAS for 100% I/O traffic or as good as MC at 0% I/O traffic, MC-Elongated achieves the best average service time for 60%, 40% and 20% I/O traffic. The results for the SDSC workload or other utilization levels are very similar.

The success of MC-Elongated is due to its ability to achieve both compactness and balance. Fig. 9 shows that dispersal as measured by *nodes_affected* of MC-Elongated is much better (lower) than that of PLAS. Fig. 10 shows that the average balance factor of MC-Elongated is much better (lower) than that of MC.

## 4.3 General observations

Sifting through hundreds of graphs, we made three important observations. First, varying system load has quite the opposite effect on balance and I/O-intensive workloads than it has on compactness and communication-intensive workloads. As the system load increases (interarrival time decreases, utilization increases), the average balance factor of Paging, MBS and MC decreases. The fuller the system the more likely we get good balance for free. If the system is full, it is automatically balanced. This is also the reason why at high loads, relative improvements in service time decreases. In other words: For I/O-intensive workloads, spatial layout (balance) is more important at lower loads.

However, as system load increases, dispersal – the opposite of compactness – increases (see Fig. 9). It becomes harder to find compact allocations. Random and MBS have high dispersal. At low loads they get away with it since often there are no other jobs present to interfere with. However, as the load increases, their service time decreases much faster than that of the other strategies. In other words: for communication-intensive workloads, compactness is more important at higher loads.

Second, the impact of I/O traffic dominates over communication traffic with respect to network contention, service time and turnaround time. Even at a traffic ratio of only 20% I/O traffic, the performance graphs (shapes, dependence on system load as described above) are more similar to those at a 100% I/O traffic than to those at a 0% I/O traffic.

Third, the increase in service time should not be underestimated. In all the strategies and experiments, we assumed unrestricted admission, i.e. a job is always allocated no matter how full the system or how good or bad the spatial layout determined by the allocation strategy. This has the advantage of not wasting compute power, by leaving compute nodes idle. However, as all the service time graphs show for all allocation strategies, the faster jobs arrive and the fuller we pack the system, the higher is the average service time of jobs due to communication interference and I/O hot spots. Besides applying a good allocation strategy that minimizes network contention, there probably is a point where it does not pay to pack yet another job into the system, because all the jobs get slowed down. It may be of advantage for all the jobs to restrict admission in order to avoid this increase in service time. This is an area of future work.

## 5 Conclusions

Taking on the challenge of minimizing network contention due to both interprocess communication and parallel

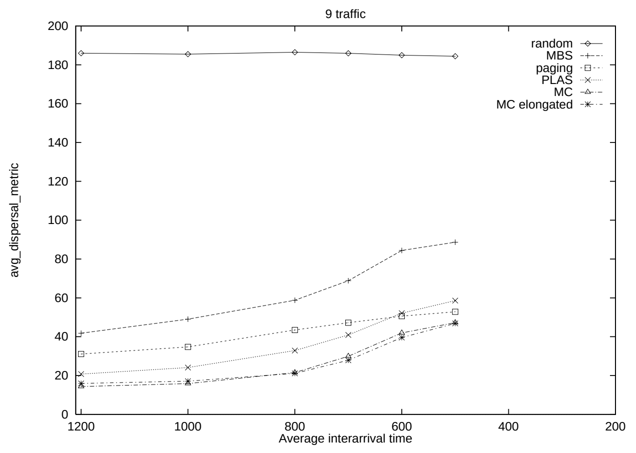

**Figure 9. Average dispersal metric**

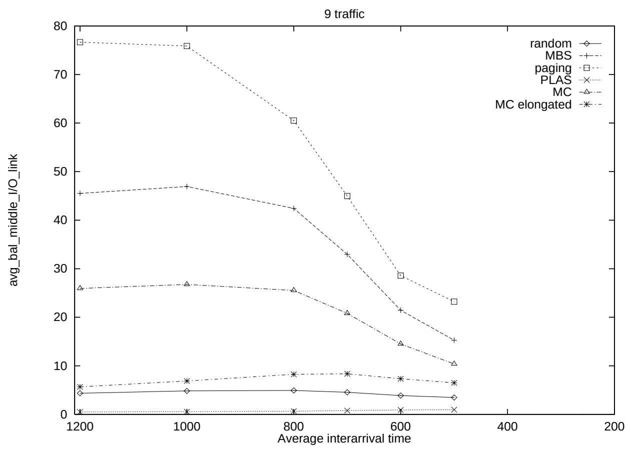

**Figure 10. Average balance factor**

I/O traffic, we designed and tested a new allocation strategy called MC-Elongated. Our extensive simulations lead us to conclude the following:

Whereas earlier allocation strategies were either sensitive to communication alone or sensitive to I/O alone, MC-Elongated is sensitive to both communication and I/O by finding allocations that are both compact and balanced. In agreement with the results of previous research, compact allocations help minimizing inter-job link contention due to interprocess communication [12], and balanced allocations help alleviate hot spots due to parallel I/O traffic [10].

Regarding average turnaround time, MC-Elongated outperforms allocation strategies that are in use today (MBS, Paging) by up to a factor of 3.834. Compared to strategies that are optimized for communication only (MC) or I/O only (PLAS), MC-Elongated wins in environments with a mixture of communication and I/O traffic. Moreover, if the traffic is unknown, MC-Elongated would be a cautious choice, it is up to a factor of 1.537 better than wrongly optimizing for the opposite kind of traffic, while it is only up to 13.7% worse than perfect optimization.

Regarding the tension between communication and I/O, our simulations show that spatial layout is more critical for I/O-intensive jobs at lower utilization levels and more critical for communication-intensive jobs at higher utilization levels; and that in general, the impact of I/O traffic is dominant.

## References

[1] Sandra Johnson Baylor, Caroline Benveniste, and Yarsun Hsu. Performance evaluation of a parallel I/O architecture. In *Proceedings of International Conference on Supercomputing*, 1995.

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

[3] Y. Cho, M. Winslett, S. Kuo, Y. Chen, J. Lee, and K. Motukuri. Parallel I/O on networks of workstations: Performance improvement by careful placement of I/O servers. In *Proceedings of High Performance Computing on Hewlett-Packard Systems*, 1998.

[4] 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.

[5] Sharad Garg. Parallel I/O architecture of the first ASCI TFLOPS machine. In *Proceedings of Intel Supercomputer Users Group*, 1997.

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

[7] David Kotz. Disk-directed I/O for MIMD multiprocessors. *ACM Transactions on Computer Systems*, 15(1):41–74, February 1997.

[8] 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.

[9] Virginia Lo, Jens Mache, and Kurt Windisch. A comparative study of real workload traces and synthetic workload models for parallel job scheduling. In *Proceedings of the 4th Workshop on Job Scheduling Strategies for Parallel Processing, IPPS '98*, 1998.

[10] Jens Mache, Virginia Lo, and Sharad Garg. How to schedule parallel I/O intensive jobs. In *Proceedings of the 6th Conference on Parallel and Real-Time Systems*, 1999.

[11] Jens Mache, Virginia Lo, Marilynn Livingston, and Sharad Garg. The impact of spatial layout of jobs on parallel I/O performance. In *Proceedings of the 6th Workshop on I/O in Parallel and Distributed Systems, FCRC'99*, 1999.

[12] 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.

[13] 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.

[14] 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.