Resource allocation and utilization in the Blue Gene/L supercomputer

Y. Aridor
T. Domany
O. Goldshmidt
J. E. Moreira
E. Shmueli

This paper describes partition allocation for parallel jobs in the Blue Gene®/L supercomputer. It describes the novel network architecture of the Blue Gene/L (BG/L) three-dimensional (3D) computational core and presents a preliminary analysis of its properties and advantages compared those of with more traditional systems. The scalability challenge is solved in BG/L by sacrificing granularity of system management. The system is treated as a collection of composite allocation units that contain both processing and communication resources. We discuss the ensuing algorithmic framework for computational and communication resource allocation and present results of simulations that explore resource utilization of BG/L for different workloads. We find that utilization depends strongly on both the predominant partition topology (mesh or torus) and the 3D shapes requested by the running jobs. When communication links are treated as dedicated resources, it is much more difficult to allocate toroidal partitions than mesh ones, especially for jobs of more than one allocation unit in each dimension. We show that in these difficult cases, the advantage of BG/L compared with a 3D toroidal machine of the same size is very significant, with resource utilization better by a factor of 2. In the easier cases (e.g., predominantly mesh partitions), there are no disadvantages. The advantage is primarily due to the BG/L novel multi-toroidal topology that permits coallocation of multiple toroidal partitions at negligible additional cost.

Introduction

The growing computational requirements of modern science, engineering, and finance pose significant challenges for high-performance computing. Large-scale massively parallel computations have become commonplace in science and engineering, supplementing and often supplanting traditional analytical and semianalytical methods.

Large-scale, high-precision parallel computations are commonly performed on tightly coupled parallel multicomputer systems. The computing core of a multicomputer consists of a collection of nodes. Each node has one or several central processing units (CPUs), memory, and network interfaces. A parallel job runs on a set of nodes, called a *partition*, connected (usually via a special-purpose high-performance network) in such a way that it provides a computational and communication infrastructure suitable for efficient numerical solution of the problem in question. Thus, a typical parallel job will specify not only the number of nodes (or CPUs) required to provide the necessary degree of parallelism, but possibly also a particular shape of the partition and a topology of the interconnect. For instance, the required partition may be specified as a three-dimensional (3D) rectangular block wired as a mesh or torus. The size, shape, and network topology are determined by the problem being solved, the numerical scheme chosen for the computation, the degree of precision, the time horizon

0018-8646/05/$5.00 © 2005 IBM

of the solution sought, the type of problem boundary conditions, and other factors.

To satisfy these requirements, the job-management system of a multicomputer has to perform two related tasks: allocate a partition to each arriving job and schedule the waiting jobs optimally to maximize machine utilization and reduce the job response times. To this end, the system must take into account two classes of input: the requirements of the waiting jobs and the current state of the machine, including its static configuration and the resources allocated to the jobs already running.

The Blue Gene*/L (BG/L) supercomputer [1], built at IBM Research for the Lawrence Livermore National Laboratory, represents a new level of scalability for tightly coupled parallel multicomputers. With 2^{16} dual-CPU nodes and a target peak performance of 360 teraflops, it is a leap of at least an order of magnitude in size and speed from the fastest supercomputers of today. Naturally, the job- and resource-management system has a prominent place among the challenges for system design.

The design of the individual components of BG/L, such as nodes and network links, has been described elsewhere [1]. In this paper, we present the *global* design of the BG/L computational core and its network topology, and the operation of the job-scheduling and partition-allocation system. We focus on the operational requirements and assumptions of the job-management system, relevant algorithms, and the implications for projected system resource utilization and job response times. We also demonstrate some of the advantageous characteristics of the BG/L novel multi-toroidal network architecture.

The high-performance network that connects the nodes in parallel multicomputers is designed with the job topology requirements in mind. A frequently used interconnect topology is a 3D mesh or torus, in which every node is connected to its six neighbors, two in each dimension (a torus differs from a mesh in that the six edges are connected in a wraparound fashion). This interconnect topology is simple and scalable (the number of links grows linearly with the machine size), and it suits many types of real-world computations. Examples of 3D toroidal parallel systems are the Cray T3D** and the Cray T3E** machines [2–4]. Blue Gene/L is a departure from this tradition: its core network, described in detail in the next section, is the first implementation of a variant of multi-toroidal topology [5] that has distinct advantages in terms of efficient job allocation.

Job partitions on BG/L must be isolated; i.e., the network links connecting the partition nodes must be dedicated to the partition and used by no more than one job at a time. This requirement stems primarily from the particular security needs of BGL, given the sensitive nature of some of the jobs likely to run on the machine, but it has other benefits as well. In particular, isolation means that there is no congestion associated with messages belonging to different jobs passing through shared communication links, and it simplifies allocation algorithms.

The simplest way to allocate isolated partitions is to allocate nodes contiguously; i.e., each partition consists of nodes that are geometrically adjacent in the 3D representation of the machine. Therefore, each partition has the shape of a contiguous 3D rectangle, a natural shape for most parallel jobs related to such general computational tasks as solutions to ordinary or partial differential equations, linear algebra problems, and the like.

The particular shape of the rectangle, i.e., its size in each dimension, may also be specified by the job according to the dimensions of the original problem, the desired numerical precision, the requirements of the particular numerical scheme, and so on. Therefore, while the system may sometimes be free to choose the partition shape as long as enough nodes are allocated, in many cases it may be required to allocate a particular number of nodes in each dimension. This may impose additional restrictions on partition-allocation algorithms and may adversely affect resource utilization.

Another important parameter of a parallel job is the required partition topology. Many jobs require a mesh interconnect that provides direct communication links between adjacent nodes. Sometimes a job may require a toroidal connection whereby the opposite faces of a 3D mesh must be connected to each other in all or some of the dimensions. Requirements for toroidal connections are more difficult to satisfy because additional links are needed to close the torus. If the links are treated as dedicated resources, as is the case in BG/L, this may prevent allocation of other partitions that could otherwise use those links. We show that the novel network topology of Blue Gene/L offers significant advantages for toroidal partition allocation compared with the traditional mesh and torus machines, at negligible cost.

The requirements for size, shape, and topology of partitions must be satisfied in such a way that the utilization of the system resources—primarily nodes and network links—is maximized, while the job response time is minimized. Successful partition-allocation algorithms must be able to perform this task while confronting the growing scalability challenges presented by newer, larger systems. The scalability challenge of BG/L is particularly difficult because of the order-of-magnitude or greater increase in size compared with the current generation of parallel multicomputers.

The design of BG/L solves the scalability problem by trading off the granularity of management. From the system management point of view, the computational

core is organized into sets of 64 nodes, each of which constitutes a single autonomous entity. Therefore, rather than managing 2^{16} individual nodes, one has to deal only with 2^{10} subsystems [6].

The job-management system takes a similar *scalability at the expense of granularity* approach to partition allocation. The system is divided into allocation units, each of which is a mesh-connected rectangular set of nodes. This arrangement allows us to significantly reduce the complexity of the partitioning algorithms. It also has a very important property: The topology (mesh or torus) of a rectangular partition created from such allocation units is the same as the topology of the underlying collection of nodes that form the partition. This is the foundation for partition allocation at the granularity level of allocation units instead of individual nodes. The basic tradeoff is that only full allocation units can now be allocated, so if a particular job can run in a partition comprising a noninteger number of allocation units, some resources will be wasted.

In the five sections that follow, we describe the architecture of Blue Gene/L and its particular properties related to partition allocation, the allocation principles and algorithms, our simulation environment, some simulation results demonstrating BG/L projected performance for allocation of different kinds of jobs, and conclusions.

Blue Gene/L topology

Blue Gene/L is a 3D machine of $64 \times 32 \times 32$ ($= 64\text{K} = 2^{16}$) nodes. The nodes are grouped into 512-node units called *midplanes*. Inside a midplane, each node is directly connected to its six nearest neighbors, forming an $8 \times 8 \times 8$ 3D mesh. Each midplane is connected to other midplanes through three network switches, one per dimension, forming a 3D machine of $8 \times 4 \times 4$ midplanes.

Figure 1(a) shows a midplane with its switches. As shown in **Figure 1(b)**, each switch has three input ports and three output ports. One of the input ports and one of the output ports are connected to the opposing sides ("faces") of the midplane in the corresponding dimension. The remaining ports may be connected to ports of other switches by communication links (hereafter referred to as *external interswitch links*) or may be left unused. Only one link can be connected to each port, and one end of each link is connected to an output port and the other end to an input port. These external links are static, but each switch can create dynamic internal connections between any of its input ports and any of its output ports. Each external link and each internal connection in a switch can carry the traffic of 64 internode links, allowing the connectivity to be extended beyond a single midplane. The combination of static external links and dynamic

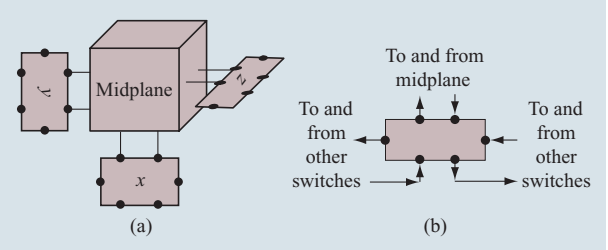

Figure 1

A midplane and its switches: (a) the three switches; (b) switch I/O ports.

internal connections is what ultimately facilitates the creation of dynamic partitions of different topologies that consist of one or more midplanes.

The external interswitch links determine the interconnect topology of the machine. In BG/L, there are no links between switches that belong to different dimensions. This separation permits a view of the 3D machine as a collection of independent 1D "lines" (hereafter, x-line, y-line, and z-line). Moreover, all of the lines that belong to the same dimension are identical. Using dimension x as an example, links exist only between switches that belong to the same x-line, and all x-lines have the same link configuration.

As noted above, the external interswitch links and the internal switch connections together allow connecting midplanes as meshes or tori. Since each midplane is a 3D mesh of nodes, the topology of a collection of midplanes and the topology of the constituent nodes will be identical. This property transforms the 64K-node machine into a more manageable and scalable 128-midplane machine. It is also clear that a midplane is the smallest unit that can be connected as a torus by creating the appropriate internal connections in its three switches.

In what follows, we describe the network topology of Blue Gene/L. **Figure 2(a)** shows the external links in a BG/L x-line of eight midplanes and eight switches. Clearly, the x-line has a multitoroidal topology [5]; i.e., multiple toroidal partitions can coexist in it. **Figure 2(b)** illustrates this property by way of an example. Note that this is achieved by modifying the traditional toroidal interconnect in only a few places (between midplanes) in each line. Thus, although there are 16 x-lines, the cost of such modification is low, and unlike the full crossbar topology, this kind of interconnect is scalable and remains practical, even for a very large machine.

Note that for various practical and engineering reasons—for example, physical limitations on the number and length of wires—BG/L does not implement the regular multi-toroidal network presented in [5]. We describe the actual BG/L topology and analyze its

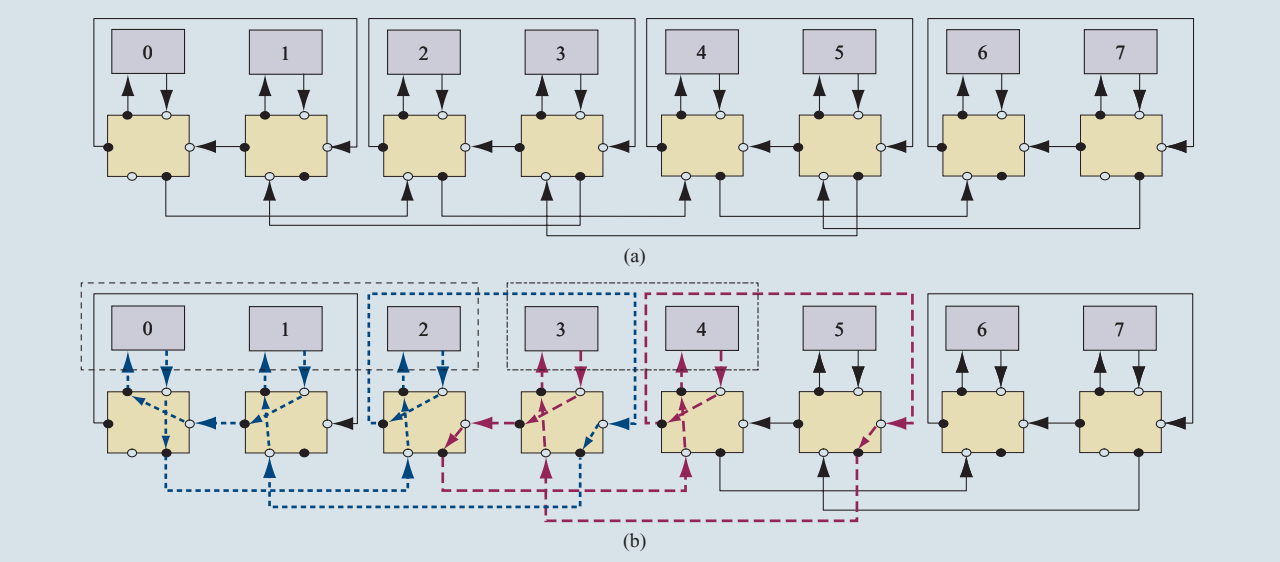

Figure 2

(a) Blue Gene/L x-line. (b) Two toroidal partitions in a single x-line.

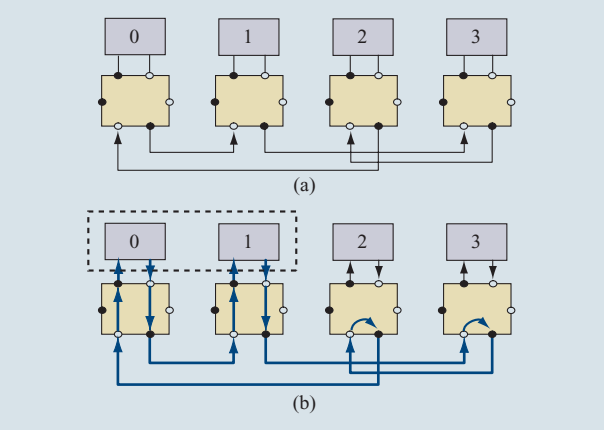

Figure 3

(a) Blue Gene/L y-line (z-lines are identical to the y-lines). (b) Toroidal partition of two midplanes {0, 1} in a y-line.

properties with respect to efficient allocation of partitions of various types. The analysis is therefore the first practical application of the theoretical study presented in [5]. We show that the BG/L multi-toroidal core network follows the same principles and enjoys the same advantages as the idealized network discussed therein.

Figure 2(b) is an example of two toroidal partitions that coexist in one x-line. One partition {0, 1, 2} consists of midplanes 0, 1, and 2, ordered 0→2→1 and back to 0, wired via the blue dotted links. The second partition is composed of midplanes {3, 4} wired with the red dashed links. Note that in both cases we are required to use switches that do not belong to the constituent midplanes. For this x-line topology, it is possible to develop a set of linking rules to guarantee that there will be sufficient external links to connect all possible sets of contiguous partitions. We omit this set of rules here because of its complexity. They are similar in spirit to the rules defined in [5]. Instead, the next section presents a general algorithmic framework that covers this and other, more complicated configurations.

Figure 3(a) illustrates a BG/L y-line, which is simpler than the x-line of Figure 2(a). The y-line switches form a single torus, 0→1→3→2→0. The deviation from the "natural" increasing order (0→1→2→3→0) is due simply to a physical limitation on the length of the cables and is not essential. The z-lines are identical to the y-lines.

In all of the dimensions, multiple mesh partitions can coexist in a single line. Toroidal partitions that consist of one midplane each can coexist peacefully as well; to create one, it is enough to use an internal link between the two switch ports that are connected to the midplane. However, in y-lines and z-lines, only one toroidal partition containing two or more midplanes can exist at any one time. For example, partition {0, 1} in **Figure 3(b)** can be connected as a torus only by using *all* of the links in the line. Therefore, this partition cannot coexist with an additional torus or mesh partition of size two in the same y-line. As Figure 2(b) shows, the multitoroidal x-lines are much more flexible.

Allocation algorithms

Blue Gene/L can be managed at different levels of granularity. At the finest level, it is a machine of $64 \times 32 \times 32$ nodes. Any job scheduler that operates at this level of granularity consumes considerable computational power and requires a very long time to reach a scheduling and allocation decision. For example, the projection of partitions (POP) algorithm [7]—which scans for the largest free rectangular partition—requires an $O(M^5)$ time to reach a decision for a machine size of $M \times M \times M$.

Obviously, fine-grained management and machine scalability are conflicting goals. For a very large machine such as BG/L, we will have to resort to heuristics to produce output in a reasonable time and may derive suboptimal decisions.

The solution is to forego granularity and manage the machine—and the jobs—at a coarser level. A natural granularity scale is that of midplanes. Treating midplanes as atomic allocation units allows the scheduler to operate on a machine of effective size $8 \times 4 \times 4$ and run its allocation algorithms in a reasonable time.

Coarse management solves the scalability problem, but it has its price. Since the minimal allocation unit is now a midplane, any job requesting a noninteger number of midplanes will keep some of the nodes idle. Exploring the tradeoff between scalability and management granularity is beyond the scope of this paper and is a part of our future research agenda. For now, we assume that each job utilizes an integer number of midplanes, so resources are not wasted. From the user perspective, the restriction means some extra precision provided by the numerical scheme used, at little or no runtime cost for a fully parallelized computation. For our analysis, allocating midplanes has an additional advantage because a midplane can be wired as a mesh or a torus, thus easing the comparison between jobs of the same shape but different topology.

In the following sections, we describe an algorithmic framework for computational and network resource allocation. It is important to note that this framework is significantly more general than is necessary for the purposes of this paper. The presented algorithm works for any network configuration as long as the dimensions are independent and all of the lines of a given dimension are identical. It also works for the allocation of partitions containing noncontiguous midplanes.

Two-phase partition allocation

The scheduler must allocate a torus or mesh partition according to the job requirements. We divide the resource-allocation process into two successive phases: the first phase selects free midplanes that can be allocated to the job, and the second phase complements the first by finding free links to connect the midplanes as requested. This corresponds directly to the two distinct types of resources that must be allocated: computational, represented by nodes (at the granularity level of midplanes), and communication, represented by interswitch links.

An advantage of this approach is its conceptual simplicity; each phase is independent of the other and can be replaced or tuned separately for optimal performance. The two-phase approach enables us to perform a comprehensive search and examine all of the candidate partitions.

Our primary concern here is optimizing the resource utilization rather than making the allocation process efficient. By sacrificing granularity, we have gained the advantage of running the algorithms on an effectively small system so that their asymptotic complexity is no longer an issue. It is also clear that the runtime of the algorithms will be short compared with the typical runtime of large parallel jobs.

The allocation procedure works as follows. First, the machine is scanned for all 3D rectangular and spatially contiguous sets of free midplanes that match the shape of the job, including possible rotations. Then, for each of the found sets, we search for an available set of free links to connect them as a mesh or torus according to the job specification.

A candidate partition is a set of free midplanes that can be connected via available free links into a free partition that fits the job requirements. Once the scheduler creates a list of all candidate partitions, it assigns a *merit* value to each of the candidates and chooses the "best" partition for the job according to the (flexible) merit criteria. If more than one candidate has the same merit value, the first one found is chosen.

We count the participating interswitch links for each candidate partition and use it for the merit value. Although we could incorporate more complex criteria, such as the location of the partition and the fragmentation resulting from its allocation, we observed that choosing the candidate partition that uses the *minimal* number of interswitch links yields good results, since the lack of available links can prevent future allocations. Once a partition is chosen for the job, its midplanes and links are marked as allocated (for example, with the job identification number) to prevent their assignment to other queued jobs.

If the list of candidate partitions is empty, it is up to the scheduler to decide which action to take. On the basis of the scheduling policy, it can decide to attempt to backfill [8, 9] other waiting jobs or wait for one or more running jobs to terminate.

The next two subsections describe in detail the two phases of partition allocation—the search for free midplanes and the search for available network resources.

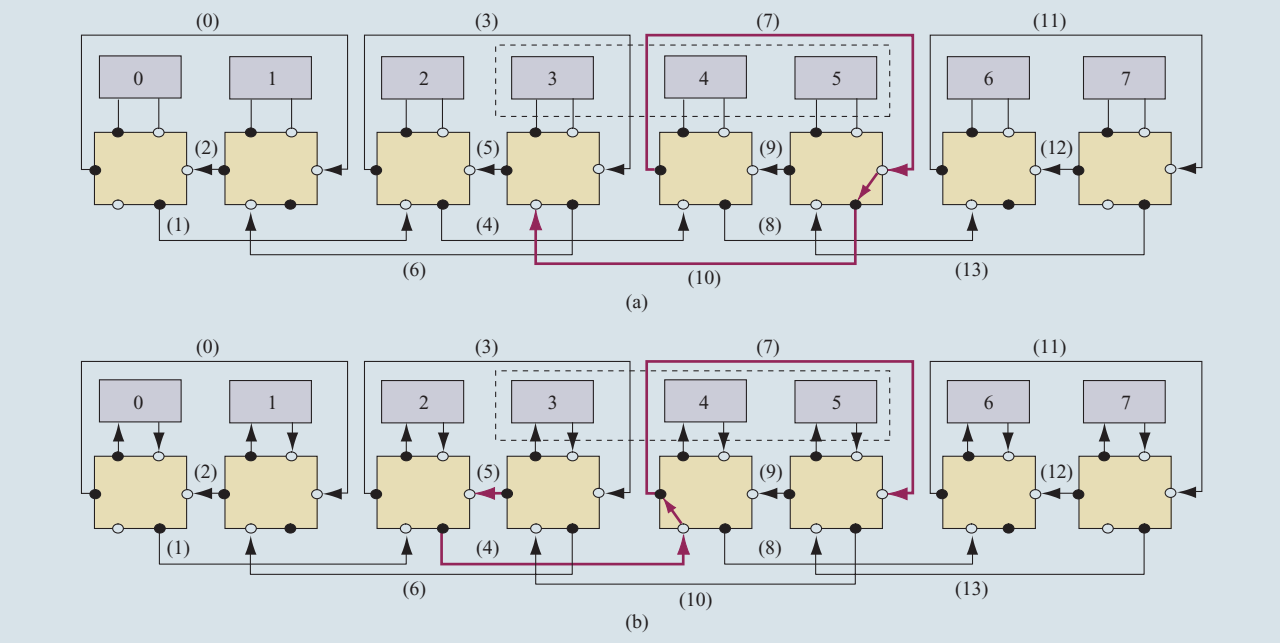

Figure 4

Two link sets suitable for mesh partition {3, 4, 5}.

Search for free midplanes

Various scanning algorithms have been suggested for finding rectangular free partitions in mesh- or torus-connected multiprocessor systems [10–16]. They differ in where the scan starts, the order in which it progresses, and the data structures used to represent and maintain the free space in the machine and the candidate partitions. We search all possible free locations that can accommodate the requested partition and then choose the best one according to predefined criteria. The search for a free partition is done as follows:

```
Given: 3D shape S <Sx, Sy, Sz>:
1. For all rotations of shape S (permutations of
   <Sx, Sy, Sz>)
2.   For all midplane locations <x, y, z>
3.     Consider that midplane as a lower left point
       of the partition and check if all midplanes
       in the 3D rectangular <x, y, z> to
       <x+Sx, y+Sy, z+Sz> are free.
4.     If true
5.       Add this collection of midplanes to the
         "free midplanes sets list"
```

For each of the free midplane sets found, we search for free links to connect it, as described in the following section. If a suitable link set is found, the partition, together with its merit value, is added to the candidate partition list.

Search for free network resources

Since there are no links between switches belonging to different dimensions, the search for the suitable link set can proceed independently in each dimension and can focus on a single isolated line. In what follows, we demonstrate the search procedure for an x-line; a more interesting case, but the exact same procedure, can also be applied to the y- and z-lines.

Consider the midplane set that consists of a set of midplanes with indices $\langle x_1, x_2, \cdots, x_n \rangle$ in the x-lines, $\langle y_1, y_2, \cdots, y_m \rangle$ in the y-lines, and $\langle z_1, z_2, \cdots, z_k \rangle$ in the z-lines. The full set of midplanes is determined by the Cartesian product of the three lists. Assume that a toroidal connection is required. For the x dimension, we need to find a set of links that connect the midplanes located at $\langle x_1, x_2, \cdots, x_n \rangle$ as a torus along the x-axis for every x-line, i.e., for every combination of the y and z coordinates. Formally, for every $(y, z) \in \langle y_1, y_2, \cdots, y_m \rangle \times \langle z_1, z_2, \cdots, z_k \rangle$, we need to find a link set that connects the midplanes located at $[x_1, y, z], [x_2, y, z], \cdots, [x_n, y, z]$ as a torus. In the version of the algorithm presented below, the link sets in all of the x-lines in the partition are identical. For example, in **Figure 4**, if link number 10 is used to wire the partition in one x-line, it is used in all of the other participating x-lines.

If a midplane is to communicate with other midplanes along the x dimension, it must use at least one of the two links that connect it to its x switch. Thus, we can assume

that these two wires are implicitly allocated whenever the midplane is allocated, and we need to look only for a set of interswitch links.

Figure 4 illustrates two link sets that can be used to connect midplanes 3, 4, and 5 as a mesh. We can use either links 7 and 10 as in Figure 4(a) or links 5, 4, and 7 as in Figure 4(b).

A closer observation reveals that for any given set of midplanes, there is a small number of link sets that connect these midplanes as a torus or mesh. Since the interswitch link topology is static and identical for all of the lines in each dimension, we can precompute a list of all possible sets of midplanes and all of the corresponding valid mesh and torus link sets for x-, y-, and z-lines and store this information in a lookup table. The table is used for an efficient online search.

The next subsubsection describes the creation of the lookup tables, followed by a description of the online search for valid link sets.

Creating lookup tables of link sets

The lookup tables maintain a mapping from all possible sets of midplanes to all valid mesh and torus link sets. We generate a separate table for every dimension, but we need to store information on only a single representative line in each dimension. **Figure 5** presents a schematic view of the x-line lookup table. The left table contains an entry for every unordered set of midplanes from one line (e.g., {1, 2, 3} and {1, 3, 2} share the same entry). Each such entry points to one or more entries in the link set table on the right. Each link set connects the midplane set either as a mesh or as a torus. For example, midplanes 1, 2, and 3 can be connected as a mesh using link sets {3, 6} and {1, 2, 3}, and as a torus using link set {1, 2, 3, 6}. Note that the references can be ordered according to some criterion—for example, pointing to the smallest link set first. By doing so, the online search for a suitable link set is automatically optimized according to the chosen criterion. The tables are generated only once, saved in persistent storage, and can be used even between reboots as long as the link configuration of the machine does not change.

There are different ways to generate the lookup tables. One simple method is to represent each line as a graph in which the midplanes are the nodes and the links are the arcs. For each legal set of midplanes for each possible order, we find a path on the graph for the mesh and a cycle for the torus that connect all of the nodes representing the participating midplanes. The tables can be populated the same way for different link configurations and for noncontiguous partitions.

The lookup tables contain only the static configuration information. The availability of the links (whether they are up and free) is kept in a different dynamic data structure.

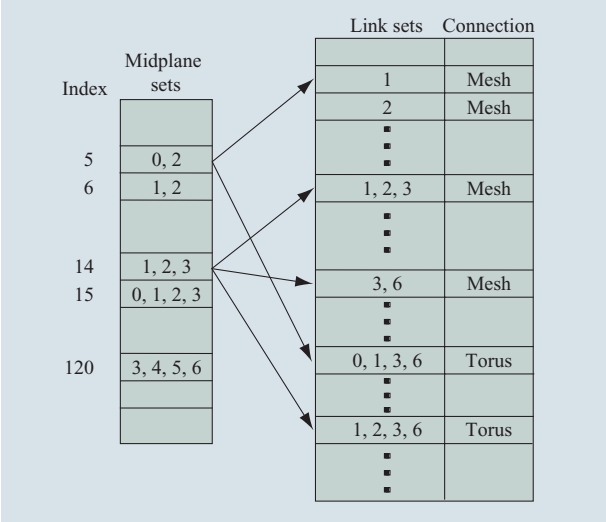

Figure 5

Lookup table for the x-line link set.

Online search for link sets

The lookup tables are used for an efficient search for suitable link sets (LS). The following algorithm describes the online search procedure for an x-line. The link sets are identical for all x-lines in a partition, which is not mandatory but simplifies the search significantly.

Online search for identical x-line link sets:

Given

- a set of midplane locations in the x dimension $\langle x_1, x_2, \cdots, x_n \rangle$,
- a set of x-lines, identified by their y and z coordinates $(y, z) \in \langle y_1, y_2, \cdots, y_m \rangle \times \langle z_1, z_2, \cdots, z_k \rangle$,
- and a requested connectivity pattern (mesh or torus),

find the partition entry in the partition table:

```
1.  for each of its link sets LS {
2.    if the LS provides the correct connectivity {
3.      for each participating x-line {
4.        if not all the links in LS are available
          (operational and unallocated)
5.          break (move to the next LS)
6.      }
7.        return LS (this LS can connect all the
          x-lines, and has the best merit value)
8.    }
9.  }
10. return NIL (no suitable link set found)
```

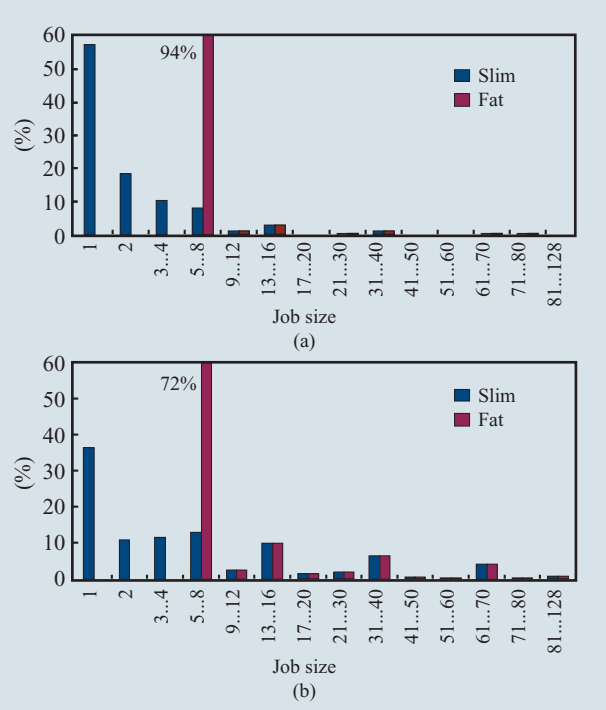

Figure 6

Job-size histograms for (a) CTC-based workloads; (b) SDSC-based workloads.

The algorithm shown above is applied for all dimensions. A positive result is returned only if a suitable link set is found in every dimension. The algorithm assumes that the referenced link sets for each midplane set entry in the table are sorted by the chosen merit value in decreasing order; i.e., the first suitable LS found is automatically the best one. For the purpose of the simulations below, the merit value is calculated by counting the links in all of the lines in all dimensions for each link set. The LS is sorted by the number of links in increasing order.

Simulation environment

Our simulation software models BG/L as a 3D collection of 128 ($8 \times 4 \times 4$) midplanes, connected as shown in Figure 2(a) and Figure 3(a). Submitted jobs are pushed to the tail of an input queue, and the scheduler is invoked whenever a new job is submitted or a running job terminates. We run an aggressive backfill scheduling; if the job at the head of the input queue cannot be accommodated, we try to schedule other jobs out of order.

We based our simulated workloads on the job logs of real parallel systems: the Cornell Theory Center (CTC) SP2 and the San Diego Supercomputer Center (SDSC) SP2. Both logs are publicly available from [17]. The logs list the size, arrival time, actual and estimated runtimes, and other descriptive fields for each submitted job. The CTC log is for a 512-node machine, and the SDSC log is for a 128-node machine. Therefore, we divided the job sizes by 4 in the CTC log to scale to our 128-allocation-unit machine.

The logs do not have any information regarding the shapes or topologies of the jobs, only scalar sizes, since neither of the systems in question is a 3D toroidal machine. Because of the lack of publicly available realistic workloads that provide useful statistics, we used these logs but had to simulate the missing parameters. We transformed the scalar sizes to 3D shapes and specified the topology (mesh or torus) for each job. For the size transformation, we computed three integers, a, b, and c, in the range of $1 \ldots 8$, such that $a \times b \times c$ was equal to the original job size. The calculation found the first match using three nested loops running from 1 to 8 in one loop and 1 to 4 in the other two loops. We then set the job shape to be $a \times b \times c$. If no combination equal to the job size was found, we used the first combination for which $a \times b \times c$ was minimal but still larger than the job size. Thus, a job of size 6 runs in a partition of $1 \times 1 \times 6$, and a job of size 27 requires a partition of $3 \times 3 \times 3$.

Note that this process preferentially generates "slim" jobs (that is, job shapes will likely resemble "sticks" or "sheets"). For example, a toroidal partition of size 8 will be assigned a shape of $1 \times 1 \times 8$ and will automatically consume the minimal number of links. In the x-line, all of the midplanes are used, so that there is no need for more links for future allocation. In the y-line and z-line, the partition is of size 1—that is, no interswitch links are needed. The situation is similar for jobs of other sizes; e.g., a job of size 16 acquires a shape of $1 \times 2 \times 8$, occupying two x-lines.

In real life, users may request "fat" jobs; e.g., a job of size 8 may have a shape of $2 \times 2 \times 2$, not necessarily $1 \times 1 \times 8$, etc. For many applications, the requested partition shape is determined by the precision requirements of the chosen numerical scheme or other application-specific considerations and cannot be arbitrarily decided by the job-management system. To explore how this affects partition allocation on Blue Gene/L, we chose to fatten job shapes at will in our simulated workload. For a fat job, the minimal size in each dimension is 2 (i.e., we do not have any jobs smaller than $2 \times 2 \times 2$). A job larger than 8 is allocated to the smallest possible partition that has at least two midplanes in each dimension. We applied the fattening algorithm at random with probability 0, 0.5, and 1, thus simulating workloads with mostly slim, mostly fat, and mixed job shapes.

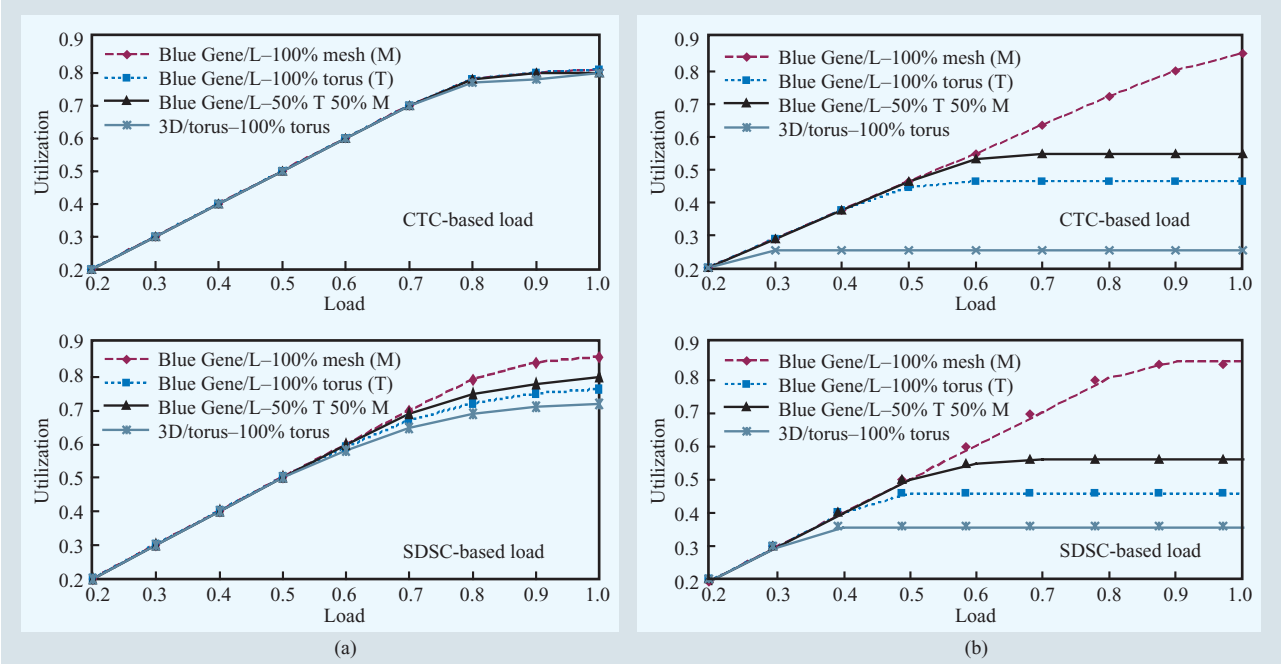

Figure 7

System utilization by (a) slim jobs and (b) fat jobs of a simulated Blue Gene/L machine compared with a simulated 3D toroidal machine for different mixtures of torus and mesh partitions.

Note that fattening affects the distribution of job sizes as well, since the minimal size of a fat job is 8, and there are many jobs smaller than 8 in the original workloads. To illustrate the effect, **Figure 6** presents histograms of job sizes in both CTC and SDSC logs before and after the fattening process.

To determine the topology, we used a simple probabilistic model that outputs "torus" with a probability of P_t and "mesh" with a probability of $(1 - P_t)$.

Different offered loads were simulated by scaling the job arrival times by different factors while leaving the job sizes, shapes, and runtimes unchanged. For each offered load, we calculated the average system utilization (see [7] for details) as the main characteristic of the job-management system performance.

For partition allocation, we used the algorithm described in the allocation algorithm section. We left any improvements in spatial allocation, including noncontiguous partition allocation, for future research.

Simulation results

In this section, we present performance characteristics of the partition-allocation algorithms on the simulated Blue Gene/L and a comparison with a 3D toroidal machine of equal size. The graphs in **Figure 7** show BG/L utilization with different loads (between 20% and 100%, as described in the section above) and different mixtures of toroidal and mesh jobs (100% mesh jobs, 50% mesh and 50% toroidal jobs, and 100% toroidal jobs). The results are compared with simulations run on a 3D toroidal machine in which all of the jobs requested toroidal partitions.[1]

The results in Figure 7(a) are for the slim job workloads. For the CTC-based log, the utilization for all of the different experiments is almost identical. The explanation can be found in the histogram of Figure 6(a). The workload file is dominated by jobs of size 1 that do not require interswitch links and jobs smaller than eight midplanes, which are allocated on a single x-line with no interswitch links in the y, z dimension. The results for the 3D toroidal machine are also very close to the BG/L results, indicating that there is no shortage of links. For the SDSC-based workload, some variation exists between the different sets, but it is still very small. The larger variation is again explained by comparison between the size distributions in Figures 6 above; the slim SDSC-based workload has a higher percentage of large jobs than the corresponding CTC-based one.

The results in the graphs in Figure 7(b) are the experiments with fat jobs. It is evident that torus-heavy

[1]The results for mesh partitions for Blue Gene/L and the 3D torus are very similar, since mesh partitions do not consume additional links like toroidal partitions.

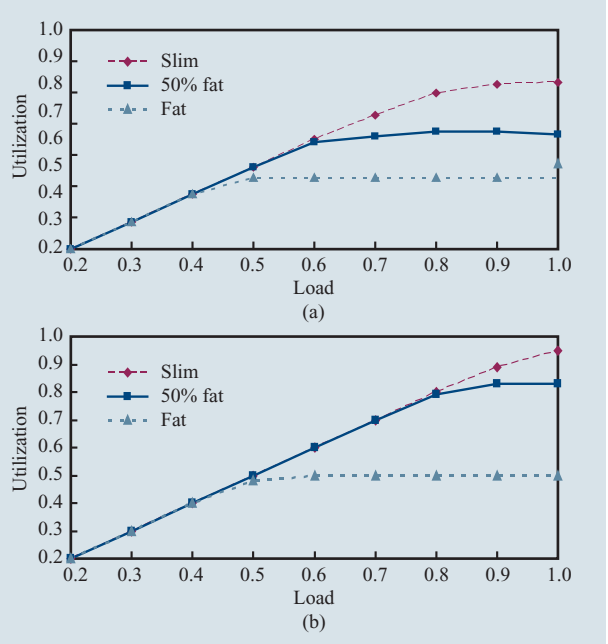

Figure 8

System utilization of a simulated BG/L machine for different job shapes: (a) SDSC; (b) CTC.

workloads result in lower utilization than workloads that consist only of meshes; the utilization decreases as the percentage of toroidal jobs in the workload increases. The reason was explained in the section on topology; toroidal partitions that contain more than one midplane in a line can consume many interswitch links, thus preventing allocation of other partitions in a large fraction of the remaining free space. Note that the graphs for the SDSC-based workload lie somewhat lower than the corresponding graphs for the CTC-based one. Again, this can be explained by the higher proportion of large jobs in the SDSC-based workload (compare Figure 6). A comparison of the BG/L results with those for the simulated 3D torus (50% compared with 26% for purely toroidal CTC-based workload and 46% compared with 36% for the corresponding SDSC-based one) emphasizes the benefits of the multitoroidal topology of BG/L x-lines for allocation of fat toroidal partitions.

Figure 8 demonstrates the influence of job shapes (slim compared with fat) on machine utilization. For the purpose of this comparison, we adjusted the sizes of the slim jobs in the same way as for the fat jobs, as described in the previous section. Thus, the distributions of job sizes are the same in both cases, and only the partition shapes differ. Note also that all partitions are toroidal. As can be seen, the fatter the workload (on average), the worse the system utilization. This decrease in utilization (up to 40%) is a consequence of the increasing demand for interswitch links by fat jobs.

Concluding remarks

In this paper, we have presented the scalable approach to partition allocation used in the Blue Gene/L job-management system. By foregoing some degree of system granularity and treating the machine as a collection of 128 midplanes rather than 64K nodes, we can use virtually any known partition-allocation algorithm without undue scalability concerns. We have presented a generic two-phase resource allocation scheme that efficiently allocates two different types of resources—computational (nodes) and communication (links)—independently. This scheme may serve as an infrastructure for allocation problems that are more advanced and sophisticated than those discussed in this paper, including resource discovery and node and link failures.

We have presented a detailed description of the BG/L novel connectivity scheme, which is multi-toroidal in the longest x dimension and simple toroidal in the other two dimensions. We have shown that for isolated contiguous rectangular partitions, the new topology can improve machine utilization by a factor of 2 (depending on the workload) compared with the traditional toroidal interconnect. This improvement is due to the ability to coallocate multiple toroidal partitions in the x dimension of the machine. The advantage of the multi-toroidal interconnect compared with the traditional 3D torus is especially pronounced for workloads requiring allocation of fat toroidal partitions that contain more than one midplane in each dimension.

The multi-toroidal topology of BG/L suggests other possible advantages, for instance, for noncontiguous allocations of partitions by leveraging the additional links to connect nonadjacent nodes. Another advantage is the degree of redundancy offered by the new topology, which leads to increased fault tolerance. These are some of the topics of our ongoing research.

Acknowledgments

We are grateful to Yoav Gal, Eitan Frachtenberg, and Yevgeny Kliteynik for fruitful discussions of the ideas in this paper, and to Uri Silbershtein for his simulation results and useful comments.

The Blue Gene/L project has been supported and partially funded by the Lawrence Livermore National Laboratory on behalf of the United States Department of Energy under Lawrence Livermore National Laboratory Subcontract No. B517552.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Cray Inc. in the United States, other countries, or both.

References

1. N. R. Adiga et al., "An Overview of the Blue Gene/L Supercomputer," *Proceedings of the ACM/IEEE Conference on Supercomputing*, 2002, pp. 1–22; see *www.sc-conference.org/sc2002/*.
2. Cray Research, Inc., *Cray T3D System Architecture Overview*, HR-04033 (September 1993).
3. D. G. Feitelson and M. A. Jette, "Improved Utilization and Responsiveness with Gang Scheduling," *Proceedings of the Job Scheduling Strategies for Parallel Processing Workshop (JSSPP)*, 1997, pp. 238–261.
4. R. Kessler and J. Schwarzmeier, "Cray T3D: A New Dimension for Cray Research," *Proceedings of the 38th IEEE Computer Society International Conference (COMPCON)*, 1993, pp. 176–182.
5. Y. Aridor, T. Domany, O. Goldshmidt, J. Moreira, E. Shmueli, and L. Stockmeyer, "Multi-Toroidal Interconnects: Using Additional Communication Links to Improve Utilization of Parallel Computers," *Proceedings of the Job Scheduling Strategies for Parallel Processing Workshop (JSSPP)*, 2004, pp. 72–88.
6. G. Almasi, L. Bachega, R. Bellofatto, J. Brunheroto, C. Caşcaval, J. Castaños, P. Crumley, C. Erway, J. Gagliano, D. Lieber, P. Mindlin, J. E. Moreira, R. K. Sahoo, A. Sanomiya, E. Schenfeld, R. Swetz, M. Bae, G. Laib, K. Ranganathan, Y. Aridor, T. Domany, Y. Gal, O. Goldshmidt, and E. Shmueli, "System Management in the BlueGene/L Supercomputer"; see *http://www.haifa.il.ibm.com/projects/systems/bluegene/papers/system_management.pdf*.
7. E. Krevat, J. G. Castaños, and J. E. Moreira, "Job Scheduling for the BlueGene/L System," *Proceedings of the Job Scheduling Strategies for Parallel Processing Workshop (JSSPP)*, 2002, pp. 38–54.
8. A. W. Mu'alem and D. Feitelson, "Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling," *IEEE Trans. Parallel & Distr. Syst.* **12**, No. 6, 529–543 (June 2001).
9. D. Lifka, "The ANL/IBM SP Scheduling System," *Proceedings of the Job Scheduling Strategies for Parallel Processing Workshop (JSSPP)*, 1995, pp. 295–303.
10. D. Das Sharma and D. K. Pradhan, "A Fast and Efficient Strategy for Submesh Allocation in Mesh-Connected Parallel Computers," *Proceedings of the 5th IEEE Symposium on Parallel and Distributed Processing*, December 1993, pp. 682–689.
11. P. J. Chuang and N. F. Tzeng, "An Efficient Submesh Allocation Strategy for Mesh Connected Systems," *Proceedings of the 11th International Conference on Distributed Computing Systems*, 1991, pp. 256–263.
12. J. Ding and L. N. Bhuyan, "An Adaptive Submesh Allocation Strategy for Two-Dimensional Mesh Connected Systems," *Proceedings of the International Conference on Parallel Processing*, 1993, pp. 193–200.
13. W. Qiao and L. M. Ni, "Efficient Processor Allocation for 3D Tori," Technical Report, Michigan State University, East Lansing, MI 48824, 1994.
14. H. Choo, S.-M. Yoo, and H. Y. Youn, "Processor Scheduling and Allocation for 3D Torus Multicomputer Systems," *IEEE Trans. Parallel & Distr. Syst.* **11**, No. 5, 475–484 (May 2000).
15. B. S. Yoo and C. R. Das, "Processor Management Techniques for Mesh-Connected Multiprocessors," *Proceedings of the International Conference on Parallel Processing*, August 1995, pp. 105–112.
16. Y. Zhu, "Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers," *J. Parallel & Distr. Computing* **16**, 328–337 (December 1992).
17. Parallel Workload Archive; see *http://www.cs.huji.ac.il/labs/parallel/workload*.

Received July 20, 2004; accepted for publication September 15, 2004; Internet publication April 12, 2005

Yariv Aridor *IBM Research Division, Haifa Research Laboratory, University Campus, Mount Carmel, Haifa 31905, Israel (yariv@il.ibm.com)*. Dr. Aridor is a Research Staff Member at the IBM Haifa Research Laboratory and manages the Distributed Computing Systems Group. He received M.S. and Ph.D. degrees in computer science from Tel Aviv University in 1989 and 1995, respectively. He joined IBM after completing his Ph.D. degree (on concurrent object systems for massively parallel machines). Dr. Aridor has several years of practical experience in server cluster middleware for performance scalability and high availability, and distributed object and mobile agent technology. His research interests include high-performance computing, cluster technology, and distributed programming models. Dr. Aridor has published papers in more than a dozen academic journals and top-level conferences.

Tamar Domany *IBM Research Division, Haifa Research Laboratory, University Campus, Mount Carmel, Haifa 31905, Israel (tamar@il.ibm.com)*. Ms. Domany received her B.S. degree in computer science from the Technion–Israel Institute of Technology in 1996. She joined the IBM Haifa Research Laboratory in 1997 and has worked on several memory management projects. Ms. Domany has been leading the job management for the Blue Gene/L project since 2002.

Oleg Goldshmidt *IBM Research Division, Haifa Research Laboratory, University Campus, Mount Carmel, Haifa 31905, Israel (olegg@il.ibm.com)*. Dr. Goldshmidt received his Ph.D. degree in physics from Tel Aviv University in 1995. He headed development of the industry-leading option pricing products for Bloomberg L.P., now widely used in the world financial markets. Dr. Goldshmidt subsequently created and led the Algorithm Research and Development Group at Comgates Ltd., developing a novel architecture and advanced algorithms for quality of service provisioning for real-time applications in next-generation networks. He later led research and development of algorithms and software for protection against distributed denial of service attacks, based on real-time analysis of network traffic, in his capacity as Director of Development for a small start-up company. In 2002 Dr. Goldshmidt joined the Storage and Systems Department at the IBM Haifa Research Laboratory, where he works on development of novel computer system architectures, including Blue Gene/L.

José E. Moreira *IBM Systems and Technology Group, 3605 Highway 52 N., Rochester, Minnesota 55901 (jmoreira@us.ibm.com)*. Dr. Moreira received B.S. degrees in physics and electrical engineering in 1987 and an M.S. degree in electrical engineering in 1990, all from the University of São Paulo, Brazil. He received his Ph.D. degree in electrical engineering from the University of Illinois at Urbana–Champaign in 1995. Since joining IBM in 1995, he has been involved in several high-performance computing projects, including the teraflop-scale ASCI Blue-Pacific, ASCI White, and Blue Gene/L. Dr. Moreira was a manager at the IBM Thomas J. Watson Research Center from 2001 to 2004; he is currently the Lead Software Systems Architect for the IBM eServer Blue Gene solution. Dr. Moreira is the author of more than 70 publications on high-performance computing. He has served on various thesis committees and has been the chair or vice-chair of several international conferences and workshops. Dr. Moreira interacts closely with software developers, hardware developers, system installers, and customers to guarantee that the delivered systems work effectively and accomplish their intended missions successfully.

Edi Shmueli *IBM Research Division, Haifa Research Laboratory, University Campus, Mount Carmel, Haifa 31905, Israel (edi@il.ibm.com)*. Mr. Shmueli received his M.A. degree in computer science from Haifa University in 2004; his thesis research was on job scheduling for parallel computers. He joined the IBM Haifa Research Laboratory in 1995 and has been working on job management for Blue Gene/L since 2002.