Multi-Toroidal Interconnects
For Tightly Coupled Supercomputers

Yariv Aridor, Tamar Domany, Oleg Goldshmidt, Member, IEEE,
Yevgeny Kliteynik, Edi Shmueli, and José E. Moreira, Member, IEEE

Abstract—The processing elements of many modern tightly
coupled multicomputers are connected via mesh or toroidal
networks. Such interconnects are simple and highly scalable, but
suffer from high fragmentation, low utilization, and insufficient
fault tolerance when the resources allocated to each job are
dedicated. High dimension interconnects may be more efficient
in certain cases, but are based on complex and expensive com-
ponents, and scale poorly. We present a novel hardware/software
architectural approach that detaches the processing elements of
the system from the interconnect and augments the traditional
toroidal topology to provide additional connectivity options and
additional link redundancy. We present the results of extensive simulation studies to show that for practically
important types of workloads the resource utilization may be
increased by 50%, and in certain cases as much as 100%
results of extensive simulation studies to show that for practically
important types of workloads the resource utilization may be
increased by 50%, and in certain cases as much as 100%

I. INTRODUCTION

A. Multicomputer Topologies

TIGHTLY coupled multicomputers are large-scale systems
intended to run massively parallel computational jobs.
Such systems consist of nodes with one or several CPUs,
memory, and network connections, capable of running one
or more concurrent execution threads each. A parallel job
requires a set of nodes, called a partition, connected in a
particular fashion. The size (in nodes) and shape of a partition
are dictated by the nature of the computational task at hand.
Many applications can be mapped to a multidimensional
mesh, where the nearest neighbors are directly and efficiently
connected, or a torus, which is a mesh with opposing faces
directly connected with a wrap-around links.

The ability to run toroidal jobs efficiently is crucial in many
applications. For instance, physical and mathematical prob-
lems with periodic boundary conditions (e.g., with spherical
or cylindrical symmetry) will prefer toroidal partitions. As
another proof point of the practical importance of toroidal jobs,
we note that acceptance tests for modern massively parallel
machines (e.g., Blue Gene/L [1]) in places like Lawrence
Livermore National Laboratory normally include toroidal jobs.

The topology of the machine is usually similar to the
expected topology of the partitions. For instance, the Cray
T3D and T3E machines ([2], [3], [4]) are connected as 3D tori.
High-dimension topologies and, in general, a high degree of
connectivity between nodes offer greater flexibility in resource
allocation and very efficient communications. However, they
require complex components, carry a high price tag, and are
not very scalable. The Earth Simulator [5] is an extreme
example with 640 nodes linked by a full crossbar interconnect.
It is doubtful that this topology can be scaled further up, as
the number of links will grow as the square of the number of
nodes. Mesh and toroidal interconnects are often preferred
due to their relative simplicity, scalability, and low price.

B. Isolated Partition Allocation

Optimal partition allocation is of critical importance for
improving resource utilization and job response times. Under
many circumstances partitions are allocated using only topo-
logically adjacent nodes. This facilitates partition isolation,
i.e. localization of the intra-job communications within the
partition, as may be required both for security reasons and to
reduce message congestion on shared network links (the ef-
ects of congestion in shared links were studied in [6]). Under
the isolation requirement, the nodes that form a partition and
the network links that connect them are dedicated resources
used by at most one job at a time. While relatively simple,
contiguous partition allocation is often not the most efficient
scheme from the point of view of resource utilization.

Manuscript received March 20, 2006; revised November 29, 2006.
Y. Aridor, T. Domany, O. Goldshmidt, Y. Kliteynik, and E. Shmueli are
with the IBM Haifa Research Lab.
J. Moreira is with the IBM Watson Research Center.
and 2 in Fig. 1 consumes all the links in the torus. If the links are dedicated resources no other partition (mesh or toroidal) can be allocated simultaneously. Generally, the likelihood of successful allocation of a new partition decreases with the number of co-allocated toroidal partitions.

2) **Allocation of non-contiguous isolated partitions is impractical.** Allocating non-contiguous partitions is attractive as a means of reducing fragmentation. However, such partitions require additional links that will not be available for other jobs. A mesh partition consisting of nodes 4 and 7 (cf. Fig. 1) will use the link between nodes 5 and 6. Therefore nodes 5 and 6 cannot be connected as another mesh partition.

3) **Insufficient fault tolerance.** Mesh and toroidal interconnects have no redundant links. A failure of a single link in Fig. 1 would disconnect the toroidal partition consisting of nodes 0, 1, and 2. For a mesh partition consisting of the same nodes a failure of either the 0 → 1 link or 1 → 2 link will be equally disastrous.

Note that all these problems are independent of the scheduling policy used.

### C. Multi-Toroidal Topology

It is clear from Section I-B that the limitations of mesh and toroidal interconnects are of a rather basic nature and are highly unlikely to be resolved by a clever new allocation algorithm. The literature survey in Section II and our own observations (cf. Section VII-B below) support this statement unequivocally. Alleviating the problems of Section I-B requires a novel architectural solution combining hardware and software elements. We present such a solution in this paper and show that it represents a practical compromise between additional hardware complexity (due to additional communication links) and better system performance (in terms of utilization), while preserving the relatively low cost, simplicity and scalability of traditional topologies.

The novel approach that we call **multi-toroidal topology** has been implemented in the Blue Gene/L supercomputer developed by IBM Research [1], currently rated the most powerful supercomputer in the world [7]. In this paper we formalize and extend the ideas that led to development of Blue Gene/L interconnect, and present a detailed analysis of the network architecture backed by extensive simulations.

We develop an algorithmic framework for resource allocation on multi-toroidal machines (see Section V). We show that multi-toroidal interconnects achieve significantly better (sometimes by a factor of 2 or more) resource utilization than toroidal ones (see also [8]), allow multi-hop allocation of isolated partitions that helps to decrease fragmentation (cf. Section IV-B), and are much less susceptible to link failures (cf. Section VII-G).

The rest of the paper is organized as follows. Section II covers related works in scheduling, processor allocation, and fault tolerance. In Section III we present the model multi-toroidal topology, and Section IV we discuss some of its useful properties. Section V focuses on the details of the algorithmic framework we use for partition allocation. Section VI describes our simulation environment. Section VII presents the experiments we ran and the main quantitative results. Section VIII concludes the paper and discusses directions of future research.

## II. Related Work

There is a wealth of literature devoted to different scheduling and resource allocation algorithms. The scheduling and allocation problems are highly interdependent: suboptimal selection of resources eventually leads to fragmentation which, in turn, forces the scheduler to reorder the queue (for instance, employ backfilling, cf. [9], [10], [11]) to maintain the system responsiveness at the risk of starvation.

Resource allocation in multicomputers has been studied extensively. In particular, 2D and 3D mesh and toroidal machines have been analyzed since 1980-ies, when they first became popular in the supercomputer systems. Later fat-tree topologies (such as Myrinet) became more popular due to simpler allocation, absence of location dependencies, etc. In recent years, 3D topologies are experiencing a renaissance (e.g., in Cray and Blue Gene/L) thanks to simpler engineering and cabling on very large scales.

First-fit schemes on 2D mesh-connected machines include [12], the two-dimensional buddy algorithm ([13], [14]), frame sliding ([15], [16]), adaptive scan [17], first-fit with partitioning [18], and Q-tree [19]. The more sophisticated best-fit class, aimed at reducing fragmentation, includes the quick allocation scheme [20], busy list ([21], [22], [23]), free list [24], improved free sublist [25], best-fit Q-tree [26], and POP [27].

For 3D toroidally connected machines Choo et al. [28] suggested the scan-search scheme, using a first-fit allocation strategy, and Qiao et al. [29] combined a lookahead strategy with complete submesh recognition.

Usually, the differences between all these algorithms are wiped out when backfilling is used [30] — an observation we confirm in Section VII below.

Hardly anyone has discussed allocation of toroidal partitions before. Our earlier work [8], [31] discussed specific examples of multi-toroidal topology and very particular and limited allocation schemes. This paper develops a robust generic methodology for comparison between traditional and multi-toroidal architectures, including a comprehensive price and performance analysis of generic resource allocation algorithms.

Mao et al. [32] investigated processor allocation on a 6D toroidal machine and showed that performance is comparable to that of a fully-connected machine. The motivation of [32] is quite different from ours: while a 6D machine can also be viewed as a 3D mesh (or torus) with additional links, Mao et al. develop a resource allocation algorithm in 6 dimensions, while we are interested in improving utilization when allocating 3D partitions. The 3D multi-toroidal topology cannot be mapped to a 6D torus (this has been confirmed by [33]). Our conclusion that adding yet more links to the multi-toroidal machine is not likely to lead to a further improvement in utilization (cf. Section VII-F) is to a certain extent similar to the conclusions of that paper.
Bruck et al. [34] investigated techniques for improving fault tolerance in \( d \)-dimensional mesh and hypercube architectures by adding spare processors and communication links. Oliner et al. [35] focused on avoiding job failures in Blue Gene/L via simple prediction of where hardware failures will occur and observed significant improvement in performance. These ideas can be applicable on multi-toroidal machines, complementing our analysis in Section VII-G.

III. MULTI-TOROIDAL ARCHITECTURE

A. Switch-Based Architecture

Allocation of isolated partitions cannot be improved by modifying the computational elements (the processors) of the machine. Therefore, a successful architecture will benefit from strong separation between the processing elements and the network elements of the machine. We augment the toroidal interconnect with additional links, creating more options to connect each partition. The system must facilitate modification of connections as partitions are created and destroyed. This suggests a switch-based architecture where dynamic connection management is achieved by manipulation of the switches’ internal configurations.

Any successful architectural solution to the problems of Section I-B must preserve the scalability of the interconnect. Associating a switch with every processor may not be feasible for scalability reasons. For instance, Blue Gene/L, with its \( 64 \times 32 \times 32 = 2^{16} \) nodes, is much larger than the largest modern clusters, and its architecture is designed to scale further up. Scalability is the primary challenge for Blue Gene/L system management, and the problem is solved by performing management tasks at a coarse level of granularity [36].

Thus, we make a very general assumption that processing resources are managed in allocation units consisting of a number of nodes (a single node is a trivial special case). We also assume that the characteristic topology scale corresponds to the scale of an allocation unit (cf. Section III-B below). The assumption simplifies the analysis and the presentation, but it is not mandatory.

B. Implementation

The above considerations lead us to the following implementation of the interconnect architecture. Each allocation unit is connected to the rest of the system through three network switches, one for each dimension, forming a global 3D network, as illustrated in Fig. 2(a).

Two of the ports of each switch are connected to the opposing sides (“faces”) of the allocation unit in the corresponding dimension (cf. Fig. 2(b)). The remaining ports may be connected to other switches. These external connections are static. In addition, dynamic internal connections can be created between any two ports of the same switch (Fig. 2(b)).

The internal switch connections facilitate dynamic creation of partitions that consist of one or more allocation units. Inside an allocation unit the constituent nodes form a static mesh network. The individual allocation unit can also be connected as a torus by creating a dynamic connection between the two ports that connect the switch to the unit’s faces in a particular dimension (cf. Fig. 2(b)). Connecting a set of allocation units as a mesh or a torus will automatically create a mesh or a torus (respectively) out of the nodes comprising the units, thus satisfying the topology requirements of large jobs.

Non-adjacent allocation units can be connected through a series of switches bypassing the processing elements belonging to other units. This makes additional connectivity options possible (see, e.g., Section IV-B below).

C. The Topology Of A Line

There are no links between switches that belong to different dimensions. This restriction may make the interconnect less efficient for workloads with certain communication patterns, but the resulting architecture is highly (linearly) scalable with the number of allocation units.

This allows the 3D machine to be decomposed into independent one-dimensional lines. The lines that belong to the same dimension are identical and independent of the other dimensions. We also assume that all the three dimensions have the same size and connectivity.

A partition that spans several lines in a given dimension consists of the same allocation units in each line. This looks like a restriction, but in practice it is enforced due to the absence of connections between switches in different dimensions. This observation allows us to consider projections of partitions onto an isolated line in the discussion that follows.

Fig. 3(a) shows a line of a mesh-connected machine. The switches in a line are connected to their neighbors in a linear fashion and are enumerated in ascending order from left to right.
Fig. 3(b) shows a line of a toroidal machine. The switches are connected in a cyclic fashion, namely \(0 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 5 \rightarrow 3 \rightarrow 1 \rightarrow 0\). One could close a torus by adding a link between switches 0 and 7 to the mesh of Fig. 3(a). However, Fig. 3(b) reflects the practical need to limit the length of the cables — the torus is often wired as shown here. A 3D torus architecture is defined by replicating the links of a single line to all the lines in the same dimension.

In [8] we proposed augmenting the toroidal line of Fig. 3(b) with additional links connecting allocation units within the line. Any such extension of the toroidal line is an example of multi-toroidal topology. To make this possible we use 6-port switches as shown in Fig. 4. The additional links are shown in bold on top of the toroidal line of Fig. 3(b).

In what follows we will prefer to use a schematic representation of lines that abstracts out the allocation units and the switches, presents a schematic view of the inter-switch links, and makes comparisons against the traditional toroidal interconnect clearer. Fig. 5(a) presents such a schematic view of the toroidal line of Fig. 3(b), while Fig. 5(b) shows the corresponding schematic representation of the multi-toroidal line of Fig. 4.

An alternative multi-toroidal topology is shown in Fig. 5(c). It represents schematically a generalization of the multi-toroidal line of Blue Gene/L— the actual Blue Gene/L interconnect has unidirectional links, which restricts on the allowed internal switch connections (cf. Fig. 2(b)).

The choice of a particular variant of multi-toroidal topology may be based on physical restrictions, such as packaging, the available number of ports, the limits on the length of cables, etc.— all of these affected the topology design of Blue Gene/L (cf. Fig. 5(c)). Symmetry and aesthetics are relevant as well, as are the expected workload patterns. We found in simulations that for a number of workloads we tried the topologies in Figures 5(b) and 5(c) possess similar characteristics. The discussion below focuses on the particular topology shown in Fig. 5(b).

The total number of additional links is linear with the size of the machine in allocation units. Specifically, for each line of \(N\) allocation units we now have \(2N - 1\) links, for \(N > 2\). Thus, the cost of such augmentation is low in terms of the total number of nodes and links in the machine, and, unlike, e.g., the full crossbar topology (where the number of links is the square of the number of nodes), this scheme remains very scalable even for a very large machine such as Blue Gene/L. This is a direct consequence of the fact there are no communication links between switches belonging to different dimensions (cf. Section III-C).

IV. PROPERTIES OF MULTI-TOROIDAL INTERCONNECTS

A. Allocation Of Multiple Partitions

The toroidal topology does not cope well with concurrent allocation of multiple partitions that consist of more than one allocation unit, when at least one partition is a torus (the first problem in Section I-B). The issue was studied in [8], and is illustrated again in Fig. 6(a). Clearly the communication links are a resource that must be scheduled and allocated together with the processors to account for this effect.

The multi-toroidal topology alleviates the problem: Fig. 6(b) presents an example of multiple toroidal partitions \((0 \rightarrow 1 \rightarrow 2 \rightarrow 0, 3 \rightarrow 4 \rightarrow 3, \text{and} 5 \rightarrow 7 \rightarrow 6 \rightarrow 5)\) allocated simultaneously, thanks to the additional links compared to the torus of Fig. 6(a). There are still links that are not used in this partitioning: one of the \(0 \rightarrow 1\) links, one of the \(6 \rightarrow 7\) links, and links \(1 \rightarrow 3, 3 \rightarrow 5, \text{and} 5 \rightarrow 6\).

In general to connect a toroidal partition we need to use switches that do not belong to any of the participating allocation units. This is true for two of the three partitions shown in Fig. 6(b): the \(3 \rightarrow 4 \rightarrow 3\) partition uses switch 2, while the \(5 \rightarrow 7 \rightarrow 6 \rightarrow 5\) partition uses switch 4.
B. Multi-Hop Allocation

In the toroidal line of Fig. 5(a) the most natural way to connect a partition is to use the direct connections between the corresponding switches. In this case any two topologically adjacent allocation units in the partition will be separated by a single inter-switch “hop.” It is natural to refer to such allocation as “contiguous.” This notion of “contiguity” is implicitly or explicitly employed in practically all of the algorithmic research on resource allocation (cf. Section II).

In a multi-toroidal machine (Fig. 5(b)) contiguity is difficult to define. Both unit 2 and unit 3 are adjacent to unit 1, and there is no clear reason why 1 → 3 connection should be preferred to 1 → 2, and vice versa. Therefore, we prefer to use a more general notion of single-hop allocation that does not imply a fixed “order” of allocation units in a line. Obviously, this is applicable to both toroidal and multi-toroidal machines.

There are cases when single-hop allocation is not possible, and it is beneficial to resort to multi-hop allocation, i.e., to connect allocation units in a partition via more than one inter-switch link, with the help of internal connections in other switches.

One example is allocating toroidal partitions. It is possible to connect, e.g., allocation units 0 → 1 → 2 → 0 as a torus using only single hop connections (cf. Fig. 7(a)). However, if unit 2 is allocated, but units 0, 1, and 3 are free, they cannot be connected as a torus using single hops only (they can form a single-hop mesh though). To close the torus, one must connect units 3 and 0 using two hops, via links shown in bold in Fig. 7(a).

Another example concerns toroidal partitions consisting of two allocation units. Such partitions cannot be connected using single hops (except the special cases 0 → 1 → 0 and 6 → 7 → 6). One always needs a multi-hop connection in this case, e.g., via links shown as dashed lines in Fig. 7(a) for partition 4 → 5 → 4 (connected through switch 6).

Even for mesh partitions multi-hop allocation is beneficial when there is no set of adjacent allocation units available for the job. Consider the situation shown in Fig. 7(b): allocation units 1, 2, 5, and 6 are already busy and we need to accommodate a job that requires 3 allocation units. There is no way a partition of size 3 can be connected using single hops only in this case, but multi-hop allocation solves the problem easily (e.g., 3 → 4 → 5 → 7 or 3 → 4 → 6 → 7). A partition of size 4 can also be connected (e.g., 0 → 1 → 3 → 4 → 5 → 7).

This shows that multi-hop allocation is a useful property of multi-toroidal interconnects (cf. problem 2 of Section I-B). The trade-off is that it uses up more links than the single-hop scheme, and thus may interfere with or prevent allocating other partitions in the future.

C. Link Redundancy

Another important feature of multi-toroidal lines is the fact that there is some degree of link redundancy, i.e., in general partitions can be connected in several different ways. This becomes very important when some of the links are already allocated or are down due to an unrecoverable failure (cf. problem 3 in Section I-B).

This property relies again on the fact that we can use switches belonging to allocation units that do not participate in the partition that is being allocated. It is easy to see, e.g., from Fig. 7 that failure or allocation of any link does not by itself preclude wiring the allocation units that it connects using more than one “hop” through other links.

D. Routing

Note that once a partition is created it effectively forms an isolated system that runs a particular job. This means that routing messages belonging to this job is performed inside the partition only. Since an isolated partition is either a mesh or a torus, any routing scheme suitable for toroidal or mesh networks will be suitable.

V. RESOURCE ALLOCATION ALGORITHMS
A. Comparison Methodology

Our main goal is investigation of resource allocation in a 3D multi-toroidal topology compared to the 3D torus and 3D mesh architectures. For a meaningful comparison we need to compare resource allocation in multi-toroidal machines with state-of-the-art allocation algorithms for the base 3D torus
architecture, similar in spirit to comparing parallel algorithms with the best known serial one to determine parallelization speedup.

The overall comparison methodology is illustrated by Table I. We shall start from a state-of-the-art allocation algorithm that can be identified from the literature — the baseline algorithm $B$ (see Section V-B below). The baseline algorithm on a 3D mesh machine is denoted as $B_M$, the best known algorithm on a 3D torus is $B_T$ ($B_T$ and $B_M$ may be the same — see below), and we will extend the baseline scheme to the multi-toroidal topology to obtain $B_{MT}$.

In addition, we will investigate the performance of a new allocation scheme that explicitly takes into account the switch-based communication architecture described in Section III-B above. We will denote this scheme as $S$. This scheme as applied to the toroidal architecture is $S_T$, and on the multi-toroidal architecture — $S_{MT}$ (see Table I).

The motivation behind the set of algorithms to compare is as follows. First, there have been many more studies of allocation on a mesh than on a torus (see Section II). Hence, while we expect a toroidal machine to be more efficient than a mesh one, it is likely that the state-of-the-art allocation scheme was developed for the mesh architecture. In that case we must start with $B_M$ and extend it to $B_T$ that will work on a 3D torus. Of course, $B_T$ must be at least as efficient as $B_M$. Extending $B_T$ further to the multi-toroidal topology ($B_{MT}$) will allow us to assess the advantages of the new architecture afforded by the additional communication links without developing any new algorithms.

At the same time, $B_T$ will not necessarily be optimal even for a 3D torus that uses switches as described in Section III-B. We also want to compare $B_T$ with our allocation and wiring scheme $S_T$ that explicitly treats communication links as a resource. While $B_T$ is the state-of-the-art algorithm from the literature, $S_T$ may be better suited to the switch-based communication infrastructure. A similar comparison between $B_{MT}$ and $S_{MT}$ is also in order.

We expect that $B_{MT}$ and $S_{MT}$ will be at least as efficient than $B_T$ and $S_T$, respectively, because the multi-toroidal interconnect is a superset of a torus. An opposite result for either scheme will indicate that it makes suboptimal allocation decisions at least in some cases. We expect that $S_{MT}$ will utilize the resources of a multi-toroidal machine better than $B_{MT}$, since it takes into account more details of the architecture. A similar comparison between $B_T$ and $S_T$ will show whether this more detailed approach is beneficial for toroidal machines as well. Finally, we will compare $S_{MT}$ to the best of $B_T$ and $S_T$. If $S_{MT}$ yields better results then we shall conclude that the best allocation schemes on a torus are less efficient than allocation on a multi-toroidal machine. We will also be able to quantify the advantage for the chosen workloads.

### B. Baseline Resource Allocation Algorithm

The mesh allocation algorithm by Kim and Yoon [25] is a good candidate for the baseline state-of-the-art allocation scheme (algorithm $B$ in Section V-A above). The original paper compares it favorably with a large number of other algorithms (see references in [25]), and Srisawat et al. [37] also recognize it independently as the best algorithm for submesh allocation on 3D mesh-connected machine.

1) **Mesh Architecture:** Kim and Yoon [25] propose a best-fit contiguous submesh allocation on a mesh-connected machine. For each request all potential partition locations and orientations in three dimensions are checked, and a list of dominant free submeshes (i.e., a free mesh-connected space that is not embedded in a larger free mesh, cf. [25]) is constructed. The partition that corresponds to the list with the largest free submesh is chosen. We forgo the various optimizations proposed in [25] and perform an exhaustive search for candidate partitions. Since we only allocate contiguous mesh partitions in this case, link allocation can be ignored in the absence of failures.

This scheme corresponds to algorithm $B_M$ in Table I.

2) **Toroidal Architecture:** Extending the baseline algorithm [25] to toroidal machines is straightforward. Contiguous partitions can wrap around, using the extra links that close the torus in each dimension. Similarly, the dominant free partitions can use the wrap-around links. For contiguous submeshes link allocation can be ignored again. For toroidal partitions link allocation is essential (cf. Section I-B). The link allocation scheme is described in detail in [31].

This scheme corresponds to algorithm $B_T$ in Table I.

3) **Multi-Toroidal Architecture:** Further extension to the multi-toroidal architecture boils down to analysis of adjacency relations between allocation units. Recall that the algorithm allocates only contiguous partitions. On the multi-toroidal architecture the notion of contiguity depends on how one indexes the nodes. With the indexing of Fig. 8(a) any set of continuously indexed mesh or torus partitions can be co-allocated (cf. [8]). However, we lose some options because nodes 0 and 7 are not adjacent. We can index the nodes as in Fig. 8(b) where node 0 is a neighbor of node 7, but connecting toroidal partitions as a tori requires a lot of links. Hence, we index the nodes as in Fig. 8(a) for workloads that contains torus partitions, and as in Fig. 8(b) when the workload contains only mesh partitions. In either case no link allocation needs to be done.

This scheme corresponds to algorithm $B_{MT}$ in Table I.

### C. Switch-Based Resource Allocation Algorithm

Like the baseline scheme $B$ of Section V-B the switch-based algorithm $S$ also attempts to leave as many free resources as possible for future allocations. The scheme assumes that partitions that use fewer links are preferable over partitions...
that use many links, so multi-hop allocation (Section VII-D) will only be attempted if no single-hop partition is found. The algorithm also tries to avoid fragmentation by trying to place new partitions as close to a particular “origin” node (node 0 in all three dimensions, cf. Fig. 5(b)). The search for a suitable free partition starts from the “origin” node and progresses first in dimension X, then in Y, and then in Z. Since the machine is symmetric, this order is arbitrary without loss of generality.

We found empirically that it is somewhat advantageous to orient the longest side of the partition in the same order as the order of the search, i.e., for a job of size $2 \times 3 \times 1$ we first try to allocate a partition with size 3 in dimension X, if that is impossible, then we will try to allocate 3 units in dimension Y, and only then in dimension Z. In other words, we will try orientations $3 \times 2 \times 1$, $3 \times 1 \times 2$, $2 \times 3 \times 1$, $2 \times 1 \times 3$, $1 \times 2 \times 3$, and $1 \times 2 \times 3$, in that order.

The reason for this procedure is illustrated in Fig. 9: it is generally advantageous to fill the machine with partitions shaped as “sticks” or “sheets”. The left panel of Fig. 9 orients the partitions for jobs 1 and 2 as a “stick”, and thus leaves enough room for a $3 \times 3$ job 3, while the right panel does not.

We examine all candidate partitions to check that they can be connected to specification (using link lookup tables described in [31]), and, *ceteris paribus*, choose the partition with the smallest number of links. If there is more than one partition with the same minimal number of links we choose the first one found, which is the one that is closest to the “origin”.

This makes $S$ a best fit algorithm with criteria that differ from those of $B$. It can be applied to either toroidal or multi-toroidal architecture, corresponding to algorithms $S_T$ and $S_M$ in Table I.

VI. Simulation Environment

A. Machine Model

Our simulation software models the machine as a 3D collection of $512 \times 8 \times 8$ allocation units. The simulated machine can be connected as a 3D mesh (hereafter denoted as M), as shown in Fig. 3(a), a 3D torus (denoted as T), as shown in Fig. 3(b) and Fig. 5(a), or a multi-toroidal interconnect (denoted below as MT), as shown in Fig. 4 and Fig. 5(b).

B. Scheduling Policies

Submitted jobs are put in a queue, and the scheduler is invoked when a new job is submitted or a running job terminates. We experimented with both First Come First Served (FCFS) and aggressive backfilling.

It is well known [30] that backfilling tends to erase the differences between allocation policies that exist when FCFS scheduling is used, as well as improve results in general, hence we focus on backfilling, even though we ran our simulations with FCFS scheduler as well.

C. Workloads

1) Workload Logs: There are very few publicly available realistic workloads that would fit out simulated machine. 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 [38] and, [39]. The logs list the size, arrival time, actual and estimated runtimes, and other descriptive fields for each submitted job. Both logs have been widely used in the scheduling literature (the references can be found in [38]).

Neither of these logs corresponds to a three-dimensional toroidal machine, and there is no information regarding the shapes or topologies of the jobs. The CTC log corresponds to a 512-node machine while the SDSC one — to a 128-node machine. Therefore, we scale the job sizes in the SDSC workload to reproduce the original load on our 512-node simulated machine.

2) Job Shapes and Topologies: The missing job parameters had to be simulated. In particular, many real-life job requests come with shape specifications that reflect, e.g., precision requirements in different dimensions. We transformed the scalar sizes to 3D shapes by computing the scalar size to a group of three integers $x$, $y$, and $z$ between 1 and 8, such that $x \times y \times z$ is equal to the original job size. If no such combination exists, we increased the job size to the closest value larger than the original size that could be represented by $x \times y \times z$. 

---

**Fig. 8.** Schematic representations of an X-line with allocation units indexed to suit torus or mesh allocation using the $B_{MT}$ algorithm.

**Fig. 9.** Preferred orientation of a partition.
For each workload we experimented with different shape biases while keeping the same job sizes. For a given job size, we may prefer a “slim” shape, the one with the highest value of $x+y+z$, a “cubic” shape, the one with the lowest $x+y+z$, or a “fat” shape, which is a “cubic” shape with at least 2 allocation units in each dimension (for job sizes of 8 or greater). For example, a job of size 24 may have the following shapes: $2\times 3 \times 4$, $2\times 2\times 6$, $1\times 6\times 4$, or $1\times 3 \times 8$. For the “slim” workload, $1\times 3 \times 8$ will be selected, whereas for the “cubic” or “fat” workload the choice will be $2\times 3 \times 4$.

The rationale for distinguishing between “slim,” “fat,” and “cubic” shapes is as follows. A request for a “fat” or a “cubic” toroidal partition on a 3D torus will imply using many links in each dimension, while “slim” partitions tend to use few links in some dimensions at least. The difference between “fat” and “cubic” jobs is that for “fat” jobs we force the minimal partition size in each dimension, while “cubic” partitions don’t have this restriction. Thus, forcing a job to be “fat” may increase its size compared to the original size in the real workload file, and the minimal partition size is $2\times 2\times 2 = 8$.

Therefore, we expect that the advantages of the multi-toroidal topology will be more pronounced for predominantly “fat” workloads (see Section IV-A) than for “slim” ones, and we explore the different job shapes for this reason.

To explore the effect of the partition topology we run each workload twice, with all jobs requesting either mesh or toroidal partitions.

D. Allocation Algorithms

For each job selected from the queue (according to either FCFS or backfilling scheduling policy, cf. Section VI-B) the simulator scans the machine and examines all the potential locations for the job’s partition. The optimization criteria of each allocation scheme — the largest free submesh list for algorithm $B$ or the minimal number of wires for algorithm $S$ — are encapsulated in a “merit function” that is computed for each possible location, shape, and orientation of the candidate partition with available resources.

The $S$ algorithm allows for multi-hop allocation (cf. Section IV-B). We resort to multi-hop allocation only when single hop allocation is not possible. This follows from our optimization criterion — single-hop allocation uses less links to wire a given partition (cf. Section V-C).

E. Failure Model

As described in Section IV-C we expect that the multi-toroidal interconnect will improve the system’s reliability due to link redundancy. To confirm or refute this conjecture we need to incorporate a simple but plausible model of failures of both inter-switch links and allocation units into our simulation environment.

We assume that component failures are independent of each other, and that the probability distribution of a component failure is Poisson, characterized by the average uptime $U$ or its inverse, the failure rate $\mu$ (cf. Appendix I). The very preliminary failure statistics from the operational Blue Gene/L systems led us to experimenting with $\mu_L = 0.004$, $\mu_L = 0.002$, and $\mu_L = 0.001$ failures/week, corresponding to average link uptimes $U_L$ of roughly 5, 10, and 20 years, respectively. Up to uptimes of approximately 15 years link failures are dominant compared to failures of other components (cf. Appendix I), and we ignore the latter for simplicity.

We also make an additional assumption that running jobs are checkpointed at intervals much shorter than both the average link uptime and downtime and shorter than the job runtime. If a failure occurs in a busy partition, i.e., a partition in which a job is currently running, the job is inserted into the head of the queue and is restarted as soon as possible, from the point of failure.

F. Performance Metrics

We chose average machine utilization as the main performance metric. Investigation of other metrics, such as average response time, average (bounded) slowdown, etc., lead us to similar conclusions, and we omit the results for lack of space.

We present the results in terms of average machine utilization as a function of offered load. Different offered loads were simulated by scaling the job arrival times while leaving the other parameters (job sizes, runtimes, etc.) unchanged. Obviously, scaling runtimes only is equivalent.

For each offered load we calculated the average system utilization as the main performance metric. Assuming job $i$ arrives at time $a_i$ and requires $n_i$ allocation units for runtime $r_i$, out of the total of $N$ allocation units, the utilization is $\sum_i n_i r_i / [N \max_i (a_i/w + r_i)]$, where $w$ is the load scaling factor. For heavy loads ($w \rightarrow \infty$) utilization saturates at $\sum_i n_i r_i / [N \max_i r_i]$.

Additional information on our simulation environment and techniques can be found in [31].

VII. Experiments And Results

We ran simulations of both SDSC and CTC workloads, assuming different job shapes and topologies: “fat,” “cubic,” and “slim” meshes and tori (see Section VI-C.2 above). We present below a representative subset of our simulation results that illustrates our findings adequately. In most cases we describe the results that are not shown explicitly, to present a complete, albeit broad, picture. In particular, as mentioned above (cf. Section VI-B), backfilling tends to improve utilization, and is therefore the preferred scheduling policy in practice, so we focus on it, showing FCFS results only where necessary.

A. Baseline Allocation on Different Architectures

1) Mesh Allocation: Our first observation is that all the architectures deal rather well with allocation of mesh partitions.

With backfilling there is practically no difference between the 3D torus and the multi-toroidal machine, and the 3D mesh is not significantly worse, either. This is true for all the job shapes — slim, cubic, and fat. The largest observed difference is between the 3D mesh and the other two architectures (that give practically identical results) for the CTC workload and fat mesh jobs. In Fig. 10(a) the curves for the 3D torus and...
multi-toroidal machines are identical, saturating around 70%\(^1\). For the 3D mesh saturation starts around 60% load. In the other cases the differences are small, and utilization reaches 80% and even 85%.

For the 3D mesh saturation starts around 60% load. In the other cases the differences are small, and utilization reaches 80% and even 85%.

2) Tori Allocation: Our next experiment concerned allocation of toroidal partitions on the 3D torus and on the multi-toroidal machine using the baseline allocation algorithm. Obviously, there is no possibility to connect a toroidal partition on the 3D mesh machine.

As expected, the multi-toroidal machine can deal with toroidal jobs much better (by a factor of 2 or more, or by 40% to 50% in utilization) than the 3D torus (cf. Section IV-A above). As an example, Fig. 10(b) shows the average machine utilization as a function of offered load for the SDSC workload consisting of cubic tori and scheduled using aggressive backfilling. Results are similar for fat and slim tori and for the CTC workload.

This clearly indicates the advantage of the multi-toroidal topology over 3D torus for allocation of toroidal partitions, at least for the baseline allocation scheme, confirming the conclusions of [8]. Note that we made no changes in the allocation algorithm, and the algorithm is not even optimized for the switch-based architecture.

---

\(^1\)Past the saturation point the machine cannot deal with the offered load, submitted jobs accumulate in the queue, and their average waiting time becomes impractical.
rule, however, and the reason is relatively poor performance of $B_{MT}$ on this workload. For slim or cubic jobs the maximal machine utilization using $B_{MT}$ reaches 77%, similar to $S_{MT}$. The only other workload where there is a noticeable advantage to $S_{MT}$ is the case of fat mesh jobs, where $S_{MT}$ yields a 4% better utilization (in relative terms) than $B_{MT}$.

We conclude that on multi-toroidal machines $S_{MT}$ performs better (up to about 10% absolute difference in maximal utilization, or 15% relative improvement) than $B_{MT}$, even using aggressive backfilling.

C. Switch-Based Allocation On Different Architectures

1) Mesh Allocation: Fig. 13(a) compares utilization of the toroidal and the multi-toroidal machines when “fat” mesh jobs are allocated using the $S$ scheme. The 3D torus copes rather well with mesh jobs, but the multi-toroidal machine certainly does no worse. The results are similar for the SDSC workload.

2) Tori Allocation: Switch-based allocation of toroidal partitions is also much more efficient on a multi-toroidal machine compared to a toroidal one, as evident from Fig. 13(b). Without any changes in the allocation scheme the machine utilization is almost doubled, from peak utilization of 39% on a 3D torus to 77% on a multi-toroidal machine of the same size.

With other workloads we obtain very similar results.

D. Multi-Hop Allocation

Figures 11(a), 11(b), 12(a), and 12(b) also demonstrate the effect of multi-hop allocation on the multi-toroidal machine utilization (cf. Section IV-B).

If the job scheduler employs aggressive backfilling then multi-hop allocation will not improve utilization a lot compared to single hop. If the complexity and the runtime performance of the allocator are important, then single hop allocation method can be employed with good results. In the case of a FCFS scheduler, multi-hop allocation may yield a few extra per cent of utilization, and the case for it is somewhat more solid.

E. Influence of Toroidal Jobs

Clearly, it is more difficult to allocate toroidal partitions than mesh ones (cf. Section I-B), and the multi-toroidal machine manages better than the toroidal one. Fig. 14 shows maximal utilization of the toroidal and the multi-toroidal machines as a function of the percentage of toroidal jobs in the SDSC workload. The utilization of the toroidal machine degrades very fast when toroidal jobs form even a relatively small fraction of the workload. The multi-toroidal machine fares much better. The utilization still goes down a little bit as the fraction of toroidal jobs in the workload increases, because it becomes more and more difficult to connect co-allocated tori, even with multi-hop allocation. However, the difference between the two architectures is very substantial.
F. Do We Need Even More Links?

Can utilization of the multi-toroidal architecture be further improved by adding even more communication links? To answer this question we simulated (using the S allocation scheme) a full crossbar interconnect, whereby every allocation unit in each line is directly connected to every other unit of the same line\(^2\). Therefore, allocation of a partition is reduced to finding a space for the job (under the restrictions described in Section III-C), and connectivity is guaranteed.

Figures 12(a) and 12(b) include this limiting case. The multi-hop allocation scheme on the multi-toroidal machine yields results that are quite close (within 1% in the case of backfilling, cf. Fig. 12(b)), to the upper limit given by the line-wise full crossbar. This means that adding even more links will not result in significant gains in utilization.

G. Allocation In Presence Of Link Failures

We implemented the failure model described in Section VI-E (see also Appendix I) in our simulator and investigated the effect of link failures on the toroidal and multi-toroidal machines.

Table II shows that as the failure rate rises from 0 to 0.001 failures per week (i.e., each link fails once in 20 years) the machine utilization by mesh-connected jobs is 2.5 to 3.8 per cent (in relative terms) lower than when there are no failures. When the failure rate grows to 0.002 failures per week (roughly once in 10 years) the utilization drops by more than 25\%. This is practically independent of the scheduling policy or of the job shape (“slim”, “cubic”, or “fat”). With 0.004 failures per week (once in 5 years) the machine cannot cope with the workload at all. Nor can it cope with a workload of toroidal jobs, for any of the above failure rates.

Table III, on the other hand, shows that there is practically no degradation for mesh allocation on a multi-toroidal machine when the failure rate is 0.001 or 0.002 failures per week. Even when the failure rate is 0.004 per week the machine copes well with the mesh workload, and the utilization drops by less than 10\% relative to the no failure case.

Table IV shows that the multi-toroidal machine copes with

---

**Table II**

The maximal machine utilization in our simulations and the relative decrease in utilization (in parentheses) due to link failures on a toroidal machine using the SDSC workload and aggressive backfilling. The link failure rates \(\mu_L\) are in failures per week.

<table>
<thead>
<tr>
<th>Job</th>
<th>(\mu_L = 0)</th>
<th>(\mu_L = 0.001)</th>
<th>(\mu_L = 0.002)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slim</td>
<td>79%</td>
<td>77% (–2.5%)</td>
<td>57% (–27.8%)</td>
</tr>
<tr>
<td>Cubic</td>
<td>77%</td>
<td>75% (–2.6%)</td>
<td>57% (–26.0%)</td>
</tr>
<tr>
<td>Fat</td>
<td>78%</td>
<td>75% (–3.8%)</td>
<td>57% (–26.9%)</td>
</tr>
</tbody>
</table>

**Table III**

The maximal machine utilization in our simulations (in per cent) and the relative decrease in utilization (in parentheses, also in per cent) due to link failures on a multi-toroidal machine using the SDSC workload and aggressive backfilling. The link failure rates \(\mu_L\) are in failures per week.

<table>
<thead>
<tr>
<th>Job</th>
<th>(\mu_L = 0)</th>
<th>(\mu_L = 0.001)</th>
<th>(\mu_L = 0.002)</th>
<th>(\mu_L = 0.004)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slim</td>
<td>79%</td>
<td>79% (–0.0%)</td>
<td>79% (–0.0%)</td>
<td>77% (–2.5%)</td>
</tr>
<tr>
<td>Cubic</td>
<td>75%</td>
<td>75% (–0.0%)</td>
<td>75% (–0.0%)</td>
<td>70% (–9.1%)</td>
</tr>
<tr>
<td>Fat</td>
<td>76%</td>
<td>76% (–0.0%)</td>
<td>76% (–0.0%)</td>
<td>71% (–9.0%)</td>
</tr>
</tbody>
</table>

**Table IV**

The maximal machine utilization in our simulations and the relative decrease in utilization (in parentheses) due to link failures on a multi-toroidal machine using the SDSC workload and aggressive backfilling. The link failure rates \(\mu_L\) are in failures per week.

<table>
<thead>
<tr>
<th>Job</th>
<th>(\mu_L = 0)</th>
<th>(\mu_L = 0.001)</th>
<th>(\mu_L = 0.002)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slim</td>
<td>79%</td>
<td>76% (–3.8%)</td>
<td>57% (–27.8%)</td>
</tr>
<tr>
<td>Cubic</td>
<td>77%</td>
<td>68% (–11.7%)</td>
<td>57% (–26.0%)</td>
</tr>
<tr>
<td>Fat</td>
<td>77%</td>
<td>69% (–10.4%)</td>
<td>57% (–26.0%)</td>
</tr>
</tbody>
</table>

---

\(^2\)Note that we still operate under the restriction that there are no links between switches belonging to different dimensions (cf. Section III-C). This is why we consider a line-wise full crossbar as the limiting case.
allocation of toroidal jobs roughly as well as the toroidal machine with mesh jobs, for the same rate of failures. The machine gives up only when we crank up the failure rate to 0.004 per week. It is evident that the multi-toroidal machine has a clear advantage as far as tolerance of link failures is concerned.

To complete the analysis, Table V presents a sample of link and job failures in the above simulations. We run a simulation of 70000 jobs on a toroidal machine (with $8 \times 64 \times 3 = 1536$ links) and a multi-toroidal one (with $15 \times 64 \times 3 = 2880$ links). With $\mu_L = 0.002$ failures per week each link fails roughly once in 10 years. During a simulation covering approximately 2 years (the absolute duration of the simulation is not constant because we scale the jobs’ runtimes to simulate different offered loads) we obtain system-wide failure counts as shown in Table V.

There are more link failures in the multi-toroidal machine as it has more links. More jobs fail in the multi-toroidal machine than in the toroidal one because the former is better utilized, on average. On a toroidal machine, 0.13% to 0.19% of the jobs fail and are restarted due to link failures. The corresponding number for the multi-toroidal machine is 0.19% to 0.28%.

We performed the same experiments with FCFS scheduling and with the CTC workload as well, with qualitatively similar results. We omit them for space reasons.

VIII. Conclusions

This paper presents a detailed exploration of the novel multi-toroidal interconnect for tightly coupled multicompurers. The multi-toroidal topology overcomes a number of limitations of the traditional (e.g., toroidal) interconnects while preserving their best features.

The new architecture separates computational resources from the network with the help of switches and augments the toroidal machine with additional network links that provide new connectivity options and higher redundancy. The scheme is highly scalable in every respect: the number of additional links is small, and the system management may be done at a granularity coarser than the scale of an individual processor.

The multi-toroidal interconnect facilitates efficient co-allocation of multiple toroidal partitions — a crucial task that strains toroidal machines. Our simulations show that the utilization of a multi-toroidal machine is 50%, and in certain cases by as much as 100% higher than that of a toroidal one, especially when toroidal partitions must be allocated. This conclusion is valid for both FCFS and aggressive backfilling scheduling policies, for different allocation algorithms, and for various partition shapes. The novel architectural approach combining system hardware and software leads to a much better resource utilization than the state of the art scheduling schemes (such as backfilling) for practically important classes of parallel workloads.

A very important conclusion is that the multi-toroidal architecture yields machine utilization close to those of the full crossbar (per dimension) interconnect. Thus, investing in a more expensive machine with even more links will not lead to a significant improvement in utilization.

Finally, we explore the tolerance of the new architecture to link failures and find that it is significantly higher than that of a toroidal machine. In some cases (e.g., for workloads dominated by toroidally connected jobs) the toroidal machine is not even able to cope with failures, while the multi-toroidal one keeps working, although with somewhat reduced utilization.

In Appendix II we show that despite the higher price and the higher annual maintenance cost it may be economically advantageous to prefer a multi-toroidal machine because of its better utilization. The decision depends on the projected workload, in particular on the fraction of toroidal jobs: the higher this fraction the more advantageous the multi-toroidal topology is.

Acknowledgments

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

The authors are grateful to Eitan Frachtenberg, Dror Feitelson, Uri Silbershtein, and Ishai Rabinovitz for many useful discussions.

References


Yevgeny Kliteynik  Yevgeny Kliteynik received his B.S. degree in computer science from the Technion, Israel Institute of Technology in 2004. He joined the IBM Haifa Research Laboratory in 2001 and has been working on job management for Blue Gene/L since 2004.

José E. Moreira  José 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. José was the System Software Architect for the IBM eServer Blue Gene solution from 2001 to 2005 and since early 2006 José has been the Chief Architect for the Commercial Scale Out project at IBM Research. José is the author of more than 100 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.