Multi-Toroidal Interconnects: Using Additional Communication Links to Improve Utilization of Parallel Computers

Yariv Aridor[1], Tamar Domany[1], Oleg Goldshmidt[1], Edi Shmueli[1], Jose Moreira[2], and Larry Stockmeier[3]

[1] IBM Haifa Research Labs, Haifa, Israel
{yariv,tamar,olegg,edi}@il.ibm.com,
[2] IBM Watson Research Center, Yorktown, NY
jmoreira@us.ibm.com,
[3] IBM Almaden Research Center
stock@acm.org

Abstract. Three-dimensional torus is a common topology of network interconnects of multicomputers due to its simplicity and high scalability. A parallel job submitted to a three-dimensional toroidal machine typically requires an isolated, contiguous, rectangular partition connected as a mesh or a torus. Such partitioning leads to fragmentation and thus reduces resource utilization of the machines. In particular, toroidal partitions often require allocation of additional communication links to close the torus. If the links are treated as dedicated resources (due to the partition isolation requirement) this may prevent allocation of other partitions that could, otherwise, use those links. Overall, on toroidal machines, the likelihood of successful allocation of a new partition decreases as the number of toroidal partitions increases.
This paper presents a novel "multi-toroidal" interconnect topology that is able to accommodate multiple adjacent meshed and toroidal partitions at the same time. We prove that this topology allows connecting every free partition of the machine as a torus without affecting existing partitions. We also show that for toroidal jobs this interconnect topology increases machine utilization by a factor of 2 to 4 (depending on the workload) compared with three-dimensional toroidal machines. This effect exists for different scheduling policies. The BlueGene/L supercomputer being developed by IBM Research is an example of a multi-toroidal interconnect architecture.

1 Introduction

Tightly coupled multicomputers provide a natural way to build large-scale parallel systems. A tightly coupled multicomputer consists of a collection of *nodes*. Each node has one or several CPUs, memory, and network connections. Such systems are intended to run massively parallel computational jobs. A typical job requires a set of nodes, called a *partition*, connected in a particular fashion, e.g.

a three-dimensional rectangular block wired as mesh or torus. The job management system has to allocate a partition to each job and to schedule the waiting jobs optimally in order to maximize the machine utilization and to reduce the jobs' response times.

The high performance network that connects the nodes is designed with the jobs' topology requirements in mind. A frequently used interconnect topology is a three-dimensional mesh or torus where every node is connected to six neighbors, two in each dimension (a torus differs from a mesh in that the six edges are connected in a wrap-around fashion). This interconnect topology is simple, scalable (the number of links grows linearly with the machine size), and fits many types of real-world computations. Examples of three-dimensional toroidal parallel systems are the Cray T3D and T3E machines [2–4].

Job partitions are usually allocated contiguously, i.e. the constituent nodes are geometrically adjacent. Contiguous allocation simplifies allocation algorithms and facilitates partition isolation, i.e. localization of the intra-job communications within the partition. The latter is required both for security reasons and/or to reduce message congestion on shared network links. 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.

Efficient partition allocation is of critical importance to the system performance in terms of resource utilization and job response times. As shown in the next section, a toroidal partition often requires allocation of additional links to close the torus. If the links are treated as dedicated resources, this prevents allocation of other partitions that could, otherwise, use those links. Overall, on toroidal machines, the likelihood of successful allocation of a new partition decreases as the number of toroidal partitions increases. The problem is particularly acute when a first-come-first-served (FCFS) scheduling is used. Backfilling [7, 8] is an improvement over FCFS, but we show below that the adverse effect of isolated toroidal partitions on utilization exists independently of the scheduling policy used.

In this paper we present a novel approach, hereafter called *multi-toroidal topology*, that augments the traditional toroidal interconnect with additional links to improve machine utilization while allocating isolated rectangular partitions connected as mesh or a torus. Unlike other existing solutions, such as full crossbar interconnects [5], the number of additional links is small, linear in the number of allocation units in the machine (see Section 2 for more details), and thus is inexpensive and highly scalable. A variant of multi-toroidal interconnect is implemented in the upcoming BlueGene/L supercomputer developed by IBM Research [16].

Multi-toroidal topology suggests a practical compromise between additional hardware complexity (due to additional communication links) and better system performance (in terms of utilization) gained by introducing it. Utilization is increased due to the ability to allocate multiple adjacent meshed and toroidal partitions thanks to the additional connectivity options and to the possibility

of non-contiguous allocation of isolated partitions leading to less machine fragmentation.

In this paper we shall restrict ourselves to isolated contiguous rectangular partitions only. We thus focus on the ability of multi-toroidal interconnect to accommodate multiple adjacent meshed and toroidal partitions at the same time. We will show that this property alone gives the proposed interconnect a significant advantage over the traditional three-dimensional torus machines. We will defer the discussion of non-contiguous allocation to a future work.

We experimented with partition allocation on a machine with a regular multi-toroidal interconnect (see Section 2). We measured the machine utilization for two job logs of real supercomputing centers, simulating allocation of mesh and toroidal partitions on 512-unit machine, and varying the offered load. We explored two scheduling policies: FCFS and aggressive backfilling. We compared the results with those of a basic three-dimensional toroidal architecture with the same workloads, and found a significant improvement in utilization (by a factor of 2 to 4 for purely toroidal workloads) of the multi-toroidal machine compared to the traditional one.

The rest of the paper is organized as follows. In section 2 we discuss multi-toroidal topology in detail. Section 3 describes our simulation environment. Section 4 presents the main quantitative results. Section 5 concludes the paper and discusses directions of future research.

2 The Multi-Toroidal Topology

We shall use a model of the architecture of a machine that provides a topologically correct representation of the traditional three-dimensional toroidal machines and, at the same time, can be generalized to more complex topologies. This architecture separates the network connections from the processing components of the machine. The model takes into account that for scalability reasons large systems may operate in terms of *allocation units* that contain a number of nodes (as a special case, an allocation unit may contain just one node). We shall assume that an allocation unit is composed of a three-dimensional mesh of nodes and has three network switches, one for each dimension[1]

Fig. 1 illustrates an allocation unit with its switches. Two of the ports of each switch are connected to the opposing sides of the allocation unit. Remaining ports can be connected to ports of other switches by communication links. Only one link can be connected to each port. Some ports may be left unused. A switch has the capability of making internal connections between any two of its ports, thus connecting allocation units with each other.

The links connecting the switches determine the interconnect topology of the machine. We will assume that the link topology is identical for all the dimensions of the computer. We also assume that no link connects switches that belong to different dimensions, i.e. different dimensions are independent of each other. This

[1] It is easy to see that this guarantees that the topology of a rectangular network of allocation units (mesh or torus) will be identical to that of the constituent nodes.

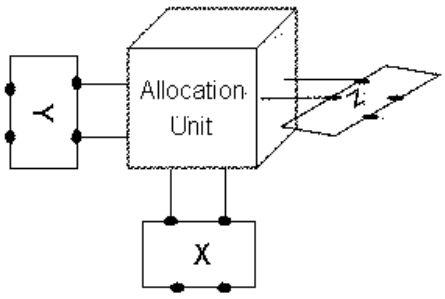


Fig. 1. An allocated unit and three switches

is the case, for example, with the BlueGene/L machine [16]. This separation of dimensions permits a view of the three-dimensional machine as a collection of independent one-dimensional "lines" in each dimension (hereafter, X-line, Y-line and Z-line). 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. This will allow us to focus on a single line rather than consider the full three-dimensional machine throughout the rest of the paper.

Fig. 2 shows an X-line of a mesh-connected machine. The switches in an X-line are connected to their nearest neighbors in a linear fashion and are enumerated in ascending order from left to right.

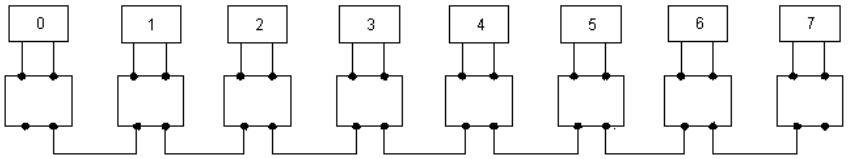


Fig. 2. An X-line of a mesh-connected machine.

Fig. 3 shows an X-line of a toroidal machine. The switches are connected in a cyclic fashion, namely 0, 2, 4, 6, 7, 5, 3, 1 and back to 0. The torus could be obtained from the mesh of Fig. 2 by adding a link between switch 0 and switch 7. However, there may be a physical limitation on the length of the cables, and the torus is often wired as shown in Fig. 3. A three-dimensional torus architecture is defined by replicating the links of a single line to all the lines in the same dimension.

Multiple meshed partitions can co-exist in a single line since each only needs links between the switches that form it. Toroidal partitions that consist of one

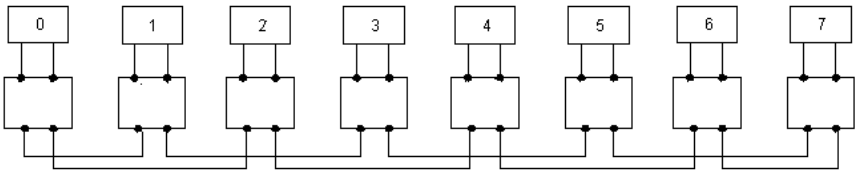


Fig. 3. An X-line of a toroidal machine.

allocation unit can live together as well: to create one it is enough to use an internal link between the two switch ports that are connected to the allocation unit. However, only one toroidal partition containing two or more allocation units can exist in a single line of a toroidal machine at a time. For example, the partition {0,1} in Fig. 4 can be connected as a torus only by using *all* of the links in the line. Therefore, this partition cannot co-exist with other partitions of more than one allocation unit, such as {6,7}, in the same line.

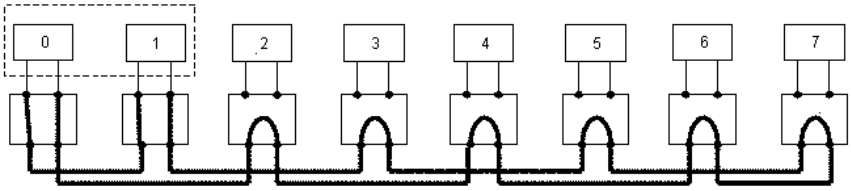


Fig. 4. A toroidal partition containing two allocation units {0,1}.

We propose augmenting the toroidal interconnect with additional links to overcome this limitation. This assumes, of course, that the switches have free ports that can be connected. Fig. 5 shows additional links (in bold) on top of the toroidal line of Fig. 3. Each switch has now six ports instead of four ports. Note that the total number of links we use is linear with the size of the machine in allocation units. Specifically, for each machine line of N allocation units we now have $2N-1$ links in that line, for $N > 2$. This number may be much smaller than the total number of links in the machine. Thus, the cost of such augmentation is low and, unlike the full crossbar topology, this interconnect is still practical even for a very large machine.

This connection scheme has a very important property that conventional toroidal topology lacks; multiple toroidal partitions can co-exist in a single line. This property is the central motivation for augmenting the traditional toroidal interconnect with additional links (as well as for the "multi-toroidal" moniker).

In the following section, we will formulate and prove a central theorem that is strictly valid for the particular wiring scheme shown in Fig. 5. We do not lose generality, however: other schemes that allow wiring toroidal partitions that do not occupy the whole line will have similar advantages. The described topology is regular, scalable, efficient, and, as shown below, greatly simplifies the algorithms of partition allocation.

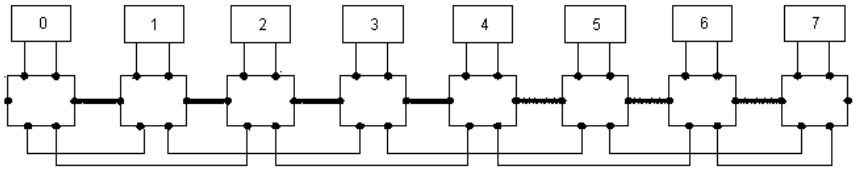


Fig. 5. Additional links (in bold) added to a toroidal line.

2.1 Properties of the Multi-Toroidal Interconnect

We describe several properties of the multi-toroidal interconnect. We focus on a single line in one dimension. The generalization to a three-dimensional machine is straightforward due to the independence of the machine dimensions and the fact that all the lines of the dimensions are identical.

Property 1: Any partition of size $N = 1$ in a particular dimension can be wired without using any links between switches in that dimension. To close a torus we will use the internal connection between the two ports that are connected to the allocation unit.

Property 2: Any mesh partition of size $N > 1$ in a particular dimension can be wired using exactly $N - 1$ links in each line it spans in that dimension. We simply connect the switches in a linear order: $i, i+1, ., i+N-1$. Such wiring uses exactly $N - 1$ links (cf. Fig. 6).

Property 3: Any toroidal partition of size $N > 2$ in a particular dimension can be wired using exactly N links in each line it spans in that dimension. We distinguish between two cases: If N is odd, we connect the switches in the following order: $i, i+2, , i+N-1, i+N-2, i+N-4, , i+1, i$ (cf. Fig. 7). If N is even the order is $i, i+2, , i+N-2, i+N-1, i+N-3, , i+1, i$ (cf. Fig. 8). With the above ordering, each switch is connected to each of its neighbors by a single link, so a total of N links are used.

Property 4: A toroidal partition of size $N = 2$ at either the left or the right boundary of the machine in a particular dimension can be wired using exactly

2 links in each line it spans in that dimension. This is obvious from Fig.5: there are two links that connect allocation units 0 and 1 (or 6 and 7).

Property 5: A toroidal partition of size $N = 2$ which is not next to either the left boundary or the right boundary of the machine in a particular dimension can be wired using exactly 3 links in each line it spans in that dimension. It is the only case where links and switches lying beyond the geometrical boundaries of the partition are used.

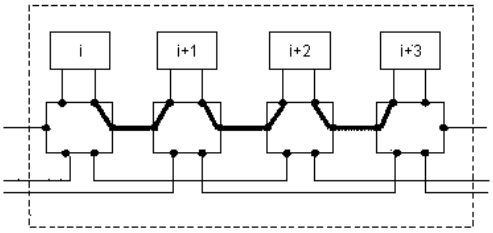


Fig. 6. A mesh partition wired in a linear manner.

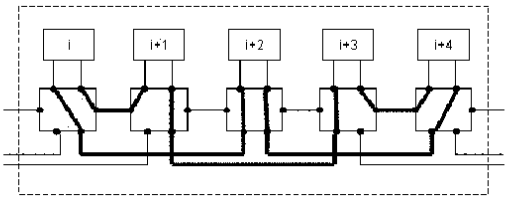


Fig. 7. A toroidal partition of odd size.

There are two ways to connect such a partition: by using an additional link to the right or to the left of the partition (*right-oriented* or *left-oriented* wiring, respectively). Fig. 9 shows an example of two such partitions. The first uses allocation units 1 and 2 and is right-oriented. The second uses allocation units 5 and 6 and is left-oriented. Fig. 9 also illustrates a fragmentation problem with a mix of co-allocated right and left-oriented partitions. Suppose we now need to connect allocation units 3 and 4 as a torus. Obviously, we lack links to close the torus, and the allocation will fail.

The solution is to use a uniform orientation for all internal toroidal partitions of size two in a particular dimension. Without loss of generality, we chose to use

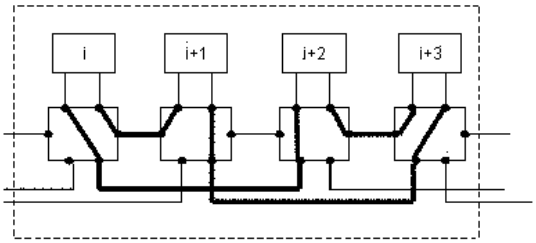


Fig. 8. A toroidal partition of even size.

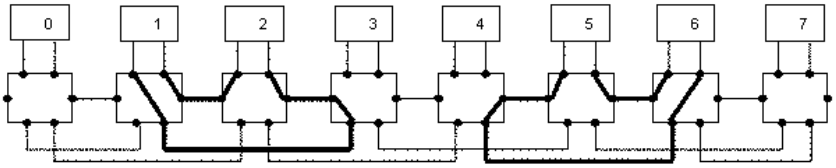


Fig. 9. Two toroidal partitions (1,2 and 5,6) of size two.

right-oriented wiring for all partitions. As seen in Fig. 10, allocation units 3 and 4 can now be connected as a torus. We have described a set of connection rules that satisfy the above properties. We now proceed to prove the main theorem behind the multi-toroidal architecture.

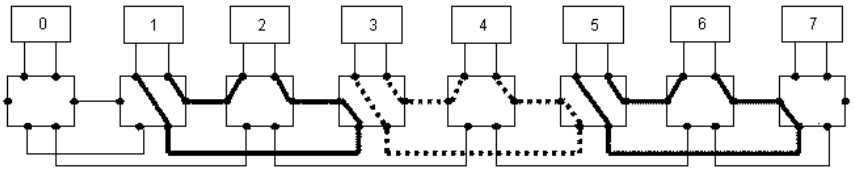


Fig. 10. Right-oriented wiring solves allocation problems.

Theorem:

Let $P_1, \ldots, P_k$, be co-allocated meshed or toroidal contiguous partitions that are wired according to the above connection rules. Let P be a new contiguous partition that can fit into the free space of a multi-toroidal machine. P can always be connected according to the same rules without changing the wiring of

$P_1, \ldots, P_k$.

Proof:

Allocation of P can fail only if one (or more) of its adjacent partitions uses one or more links that P needs in one of the lines. By assumption, we always connect partitions according to the rules described above. Assume in addition that we always use right-oriented wiring in this line. The only way P may need a link that is allocated to one of its neighbors in the line is if P and the neighbor are both of size 2 in this dimension, neither is at the edge of the machine, and if they are wired using different orientation (cf. Fig. 9). The last statement contradicts the assumption that we always choose right-oriented wiring, thus proving the theorem for a single line. Since all the lines are independent, the theorem holds for a three-dimensional machine as well.

Note that we could have equally chosen left-oriented wiring, or different orientation in different dimensions, or even different orientations in different lines in the same dimension — the proof will still hold.

The significance of the theorem lies in the proof that any new partition can be wired without modifying the existing ones, and in the following corollary that stems directly from it.

Corollary:

Allocating meshed or toroidal partitions according to the connection rules described above allows us to dispense with searching for a suitable set of links to connect the partition as requested. The above theorem guarantees that such a set exists, and the connection rules define the link set to be used.

As noted above, the theorem is strictly valid for the particular topology we have chosen. In general, however, any alternative scheme that allows allocation of multiple tori in a single line will be similarly beneficial. It may be not possible to define a set of fixed connectivity rules for any given variant of multi-toroidal topology. Accordingly, the allocation algorithms may have to be augmented with a search for a suitable link set. Multiple valid link sets may exist for the same partition, and an algorithm may be needed to determine the optimal one. Our connectivity rules are, in fact, such an algorithm: they determine a valid link set with the minimal number of wires and satisfy the above theorem. We defer discussion of more general cases to a future work.

3 The Simulation Environment

We simulated a three-dimensional machine of 512 ($8 \times 8 \times 8$) allocation units, connected as shown in Fig. 5 above. We simulated a batch system in which arriving jobs are placed in a queue in the order of arrival. The scheduler is invoked upon every job arrival and job termination to schedule queued jobs

(if any) for execution according to a specified policy. We experimented with both FCFS and aggressive backfilling scheduling policies. With the former, if no partition is found for the first job in the queue the system will wait until one or more running jobs terminate before attempting to allocate space for the first waiting job again. With backfilling, if we cannot start the first waiting job, we traverse the queue trying to schedule other jobs out of order.

We used two job logs of real parallel systems as inputs to the simulator: the Cornell Theory Center (CTC) SP2 and the San Diego Supercomputer Center (SDSC) SP2. Both logs are publicly available from [1]. Each job entry in these logs contained the job's size, arrival time, actual and estimated runtimes, and other descriptive fields. The CTC log represents a 512 node machine, and the SDSC log represents a 128 node machine. For the latter we multiplied the job sizes by 4, to scale to our 512 node machine.

Neither of these systems is a 3-dimensional toroidal machine. Nor do the logs contain the jobs' shapes, only scalar sizes. We used them because of the scarcity of publicly available realistic workloads that provide useful statistics. To adjust for the shortcomings, we had to transform the scalar sizes to 3-dimensional shapes and to specify the topology (mesh or torus) of each job. We used a simple algorithm for the size transformation: we calculated three integers, a, b and c, so that each of these integers was in the range of $1 \ldots 8$, and $a \times b \times c$ was equal to the job's size. These integers can easily be found in a first-match manner u sing three nested loops running from 1 to 8. We then set the job shape to be $a \times b \times c$. If all combinations of $a \times b \times c$ have been tried and none is found equal to the job size, we use the first combination at which $a \times b \times c$ is minimal but still larger than the job size. Accordingly, a job of size 6 will request a partition's shape of $1 \times 1 \times 6$ and a job of size 27 will request a partition with shape of $3 \times 3 \times 3$. 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 $1 - P_t$.

To simulate different offered loads we multiplied the jobs' arrival times in the logs by different constant factors leaving the rest of the logs' characteristics unchanged. For each simulation, we calculated the average system utilization (see [6] for details) to evaluate the system performance as a function of offered load.

For partition allocation, we used a brute force first-fit algorithm. Its input is the shape of a rectangular partition. We search for a free area with this exact shape, allowing for rotation. The search always starts from a specific node (in our case, the node in location (0,0,0) of the machine) and proceeds independently in every dimension.

We rely on the theorem of Section 2.1 to guarantee that any free partition in the multi-toroidal machine can be wired appropriately, as a mesh or a torus. The rules described in Section 2.1 prescribe link allocation. Therefore, any free partition of the required shape can be used to run a job. Note that the theorem guarantees that any space allocation algorithm usable on traditional toroidal machines (and more generally, on hypercubes) can be used instead of ours. Any

improvements in spatial allocation, including non-contiguous partition allocation, are left for future research.

4 Simulation Results

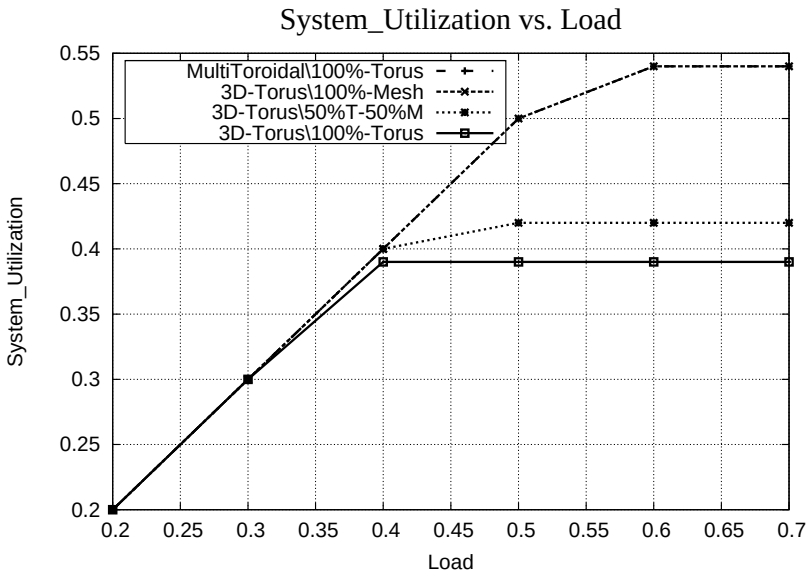
We compared the performance of a multi-toroidal machine of Fig. 5 with that of a 3-dimensional toroidal machine of equal size. For each log, we ran three sets of simulations on the toroidal machine. These simulation sets used different mixtures of toroidal and mesh jobs. In the first set, all jobs requested a mesh. In the second, 50% of the jobs requested a mesh and the rest requested a torus. In the third, all jobs requested a torus. Each set consisted of simulations with offered loads ranging between 20% and 100%, as described in Section 3. Then, we ran a single set of simulations on the multi-toroidal machine of Fig.5. In these simulations, all the jobs requested toroidal partitions.

The results for the system utilization are shown in Fig. 11 and Fig. 12 for FCFS and backfilling scheduling, respectively. Overall, utilization is lower for FCFS scheduling, which is consistent with numerous other results in the literature [9–15]. Torus-heavy workloads result in lower utilization of toroidal machines than workloads that consist of meshes only, and the utilization decreases as the percentage of toroidal jobs in the workload increases. The reason was explained in Section 2: toroidal partitions that contain more than one allocation unit can consume many links, thus preventing allocation of other partitions in a large fraction of the remaining free space.

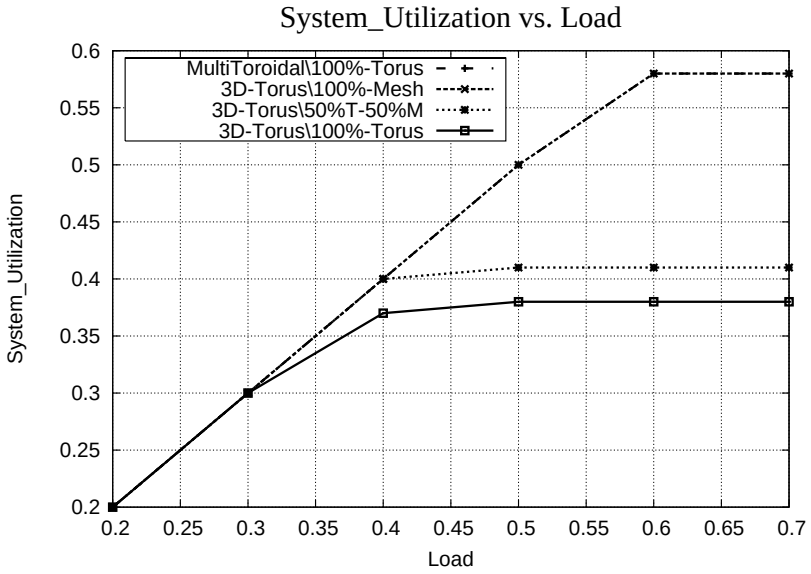
The utilization of the multi-toroidal machine is higher compared to the toroidal machine, even though all the jobs are toroidal. Specifically, it is equal to the utilization of the toroidal machine with mesh jobs only. If we think of allocation of only toroidal partitions as a worst case scenario, then the worst case performance of the multi-toroidal machine is equal to the best-case performance (with allocation of only mesh jobs) of the toroidal machine. This readily follows from the central theorem of Section 2.1.

This effect is qualitatively independent of the scheduling strategy; the advantage of the multi-toroidal topology is manifested clearly for both FCFS and backfilling scheduling schemes. To assess the difference quantitatively, one should focus on the points where each graph first deviates from a straight line: even though utilization can be improved further with higher loads, in online systems this deviation means that more jobs keep arriving than the system can accommodate, and the system will saturate unless some jobs are rejected. For fully toroidal workloads derived from the SDSC logs, the multi-toroidal machine may improve utilization by as much as a factor of 2 over the toroidal machine: 60% vs. 30% with FCFS scheduling (Fig. 11b), 80% vs. 40% with aggressive backfilling (Fig. 12b).

For fully toroidal workloads derived from the CTC logs, the improvement is more modest: 50% vs. 40% with FCFS scheduling (Fig. 11a), 80% vs. 60% with aggressive backfilling. This is partly due to the particular algorithm we used to generate the jobs' shapes from their scalar sizes (cf. Section 3 above). It eases

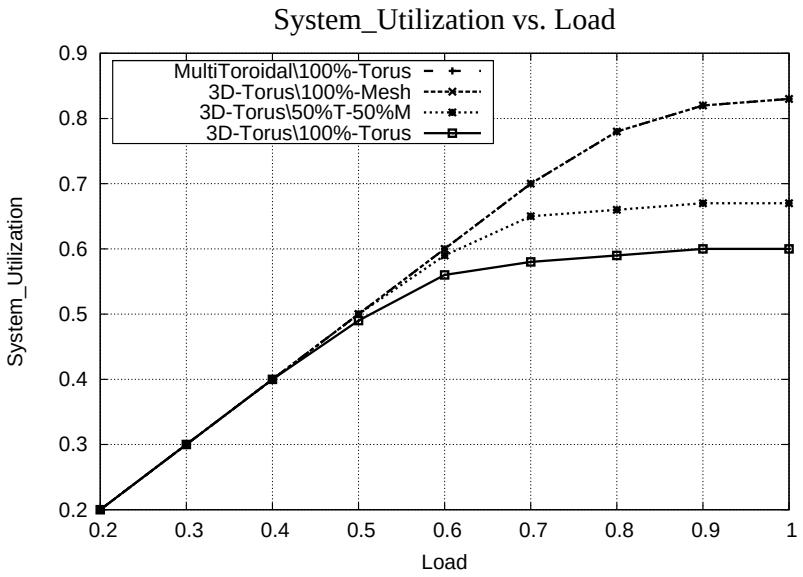


(a) CTC

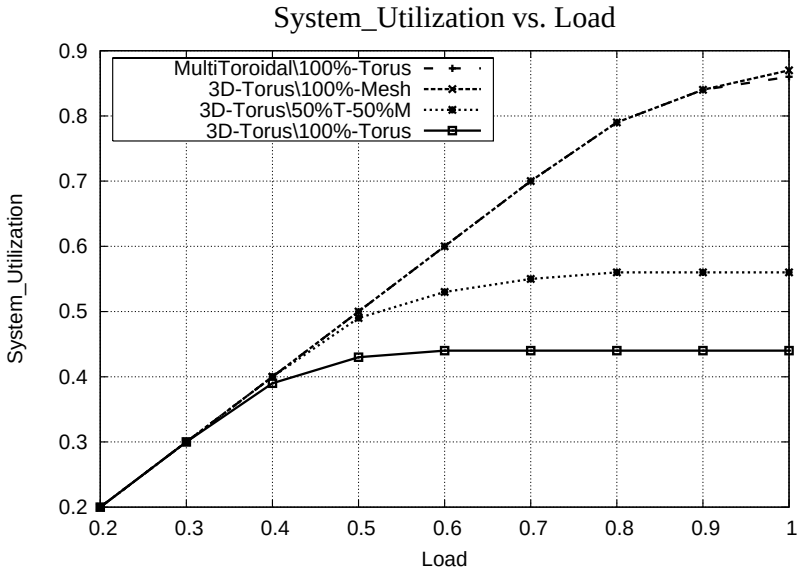


(b) SDSC

Fig. 11. System utilization of a simulated multi-toroidal machine versus a simulated three-dimensional toroidal machine — scheduling of different mixtures of torus and mesh requests with FCFS: The graphs for the multi-toroidal machine with toroidal jobs only and toroidal machine with mesh jobs only completely overlap.



(a) CTC



(b) SDSC

Fig. 12. System utilization of a simulated multi-toroidal machine versus a simulated three-dimensional torus — scheduling of mixtures of torus and mesh requests with aggressive backfilling: The graphs for the multi-toroidal machine with toroidal jobs only and toroidal machine with mesh jobs only completely overlap.

allocation of tori on traditional toroidal machines by generating partition shapes that are "slim" in one or two dimensions. For example, a toroidal partition of size 8 will be assigned a shape of $8 \times 1 \times 1$ and will be wired as a torus without consuming extra links. Similarly, a partition of size 64 will be shaped as $8 \times 8 \times 1$, and there will be no extra links involved in wiring it as a torus, either. "Fatter" partitions such as $2 \times 2 \times 2$ and $4 \times 4 \times 4$ will consume many more links when connected as tori (cf. Section 2). In fact, only two $4 \times 4 \times 4$ partitions can be fit into an $8 \times 8 \times 8$ torus simultaneously, for the overall utilization of 25%.

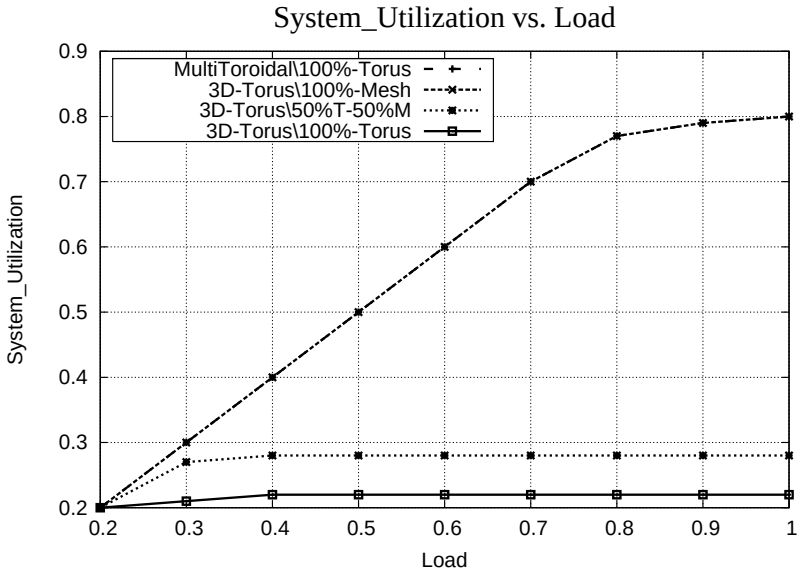
We experimented with "fat" shapes. To this end, we modified our shape-generating algorithm to start from a minimal size of 2 in each dimension. Thus, the smallest job will request a $2 \times 2 \times 2$ partition. The results of these the simulations with backfilling are shown in Fig. 13. It is obvious that the utilization of a three-dimensional torus is simply dismal in the case of torus-heavy workloads, while the results for the multi-toroidal machine (and for the purely mesh workloads) are the same as in Fig. 12, as could be expected. The multi-toroidal machine improves the resource utilization by toroidal jobs by a factor that lies in the range of 2.5 (Fig. 13b) to 4 (Fig. 13a).

Note that the spatial allocation algorithm we used in all our experiments is not optimized in any way — it is a first fit brute force search, as described in Section 3. A significant body of research comparing different allocation schemes for mesh partitions on three-dimensional toroidal machines: first fit, best fit, worst fit, lookahead allocation, etc [9–15] show that the difference in machine utilization between the different allocation schemes in not significant — at most of the order of a few per cent. Part of our future research agenda is to experiment with various algorithms for different kinds of multi-toroidal machines and to validate that the advantage of the multi-toroidal interconnect is maintained.

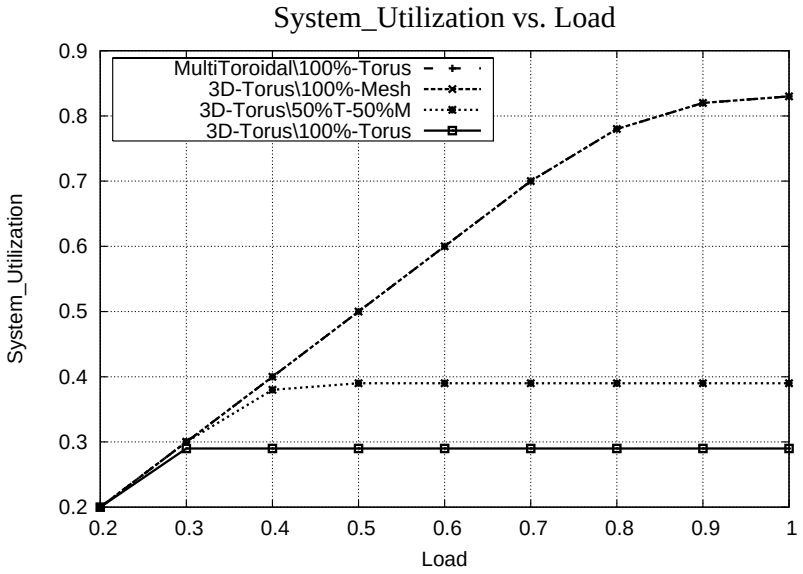
5 Concluding Remarks

We have presented a new connectivity scheme that augments the topology of toroidal parallel machines with additional links. The new "multi-toroidal" topology is d designed to allow connecting a new partition as a mesh or a torus without modifying any existing partition, as long as the new partition can fit into the free space of the multicomputer. We have shown that for isolated contiguous rectangular partitions the new topology can improve machine utilization by a large factor of 2 to 4 (depending on the workload) compared with the traditional toroidal interconnect, for different scheduling policies (FCFS or aggressive backfilling). This improvement is due to the ability to co-allocate multiple toroidal partitions in the same line of any dimension of the machine.

The multi-toroidal topology suggests other possible advantages such as non-contiguous allocations of partitions by leveraging the additional links to connect non-adjacent nodes. Non-contiguous allocation will require management (including discovery, selection and allocation) of the communication links as dedicated resources, since one will be generally able to connect a partition using one of multiple valid link sets. Another advantage is the degree of redundancy offered



(a) CTC



(b) SDSC

Fig. 13. System utilization of a simulated multi-toroidal machine versus a simulated 3-dimensional torus for different mixtures of torus and mesh requests — scheduling of "fat" jobs with aggressive backfilling: The graphs for the multi-toroidal machine with toroidal jobs only and toroidal machine with mesh jobs only completely overlaps.

by the new topology that leads to increased fault tolerance. These are some of the topics of our ongoing research.

6 Acknowledgments

We would like to thank Yoav Gal and Eitan Frachtenberg for fruitful discussions on the ideas in this paper.

References

1. Parallel workload archive. http://www.cs.huji.ac.il/labs/parallel/workload.
2. Cray Research, Inc. Cray T3D System Architecture Overview, Technical Report, Sept. 1993.
3. D. G. Feitelson and M. A. Jette. Improved Utilization and Responsiveness with Gang Scheduling, Job Scheduling Strategies for Parallel Processing workshop, Lecture Notes in Computer Science, v. 1291, pp. 238-261. Springer-Verlag, 1997.
4. R. Kessler and J. Schwarzmeier. CRAY T3D: A New Dimension for Cray Research, COMPCON, pp. 176-182. 1993.
5. Earth Simulator : http://www.es.jamstec.go.jp
6. E. Krevat, J. G. Castanos, and J. E. Moreira. Job Scheduling for the BlueGene/L System, Job Scheduling Strategies for Parallel Processing workshop, Lecture Notes in Computer Science v. 2537, pp. 38-54, Springer-Verlag, 2002.
7. A. Mualem Weil and D. Feitelson. Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling, IEEE Trans. Parallel & Distributed Syst., 12(6), 2001.
8. D. Lifka. The ANL/IBM SP Scheduling System, Job Scheduling Strategies for Parallel Processing workshop, Lecture Notes in Computer Science, v. 949, pp. 295-303, Springer-Verlag, 1995.
9. D. Das Sharma and D. K.Pradhan. A Fast and Efficient Strategy for Submesh Allocation in Mesh-Connected Parallel Computers, IEEE Symposium on Parallel and Distributed Processing, pp 682-689, 1993.
10. P. J.Chuang and N. F. Tzeng. An Efficient Submesh Allocation Strategy for Mesh Connected Systems, International Conference. on Distributed Computing Systems., pp. 256-263, 1991.
11. J. Ding and L. N. Bhuyan. An Adaptive Submesh Allocation Strategy for Two-Dimensional Mesh Connected Systems, International Conference on Parallel Processing, pp. 193-200, 1993.
12. W. Qiao and L. M. Ni., Efficient Processor Allocation for 3D Tori., Technical Report, Michigan State University, East Lansing, MI, 48824-1027, 1994.
13. S.M. Yoo, H. Choo, and H.Y.Youn. Processor Scheduling and Allocation for 3D Torus Multicomputer Systems, IEEE Trans. on Parallel and Dist. Syst., pp. 475-484, 2000.
14. S. Yoo and C. R. Das. Processor Management Techniques for Mesh-Connected Multiprocessors, International Conference on Parallel Processing, pp. 105-112, 1995.
15. Y. Zhu., Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers, Journal of Parallel and Distributed Computing, v. 16, pp. 328-337, 1992.
16. N. R. Adiga et al. An Overview of the BlueGene/L Supercomputer, Supercomputing 2002.