Scalable Topology Aware Object Mapping for Large Supercomputers

Abhinav Bhatele (bhatele@illinois.edu)
Department of Computer Science, University of Illinois at Urbana-Champaign
Advisor: Laxmikant V. Kale (kale@illinois.edu)

Abstract:

The largest and fastest supercomputers in the top500 list deploy a scalable, three-dimensional torus or mesh interconnect. For these machines, the number of links (hops) traversed by a message has a direct effect on the time required to reach the destination. This is especially true in presence of bandwidth congestion, when multiple messages share links from source to destination. For large parallel machines with a significant diameter, this can become a serious performance bottleneck. Traditionally, application developers have neglected this fact because of the advantages of virtual cut-through and wormhole routing for most message sizes on small machines. This might not be true any longer due to the large diameters of machines.

This research will demonstrate the effect of network contention on message latencies and propose and evaluate techniques to minimize communication traffic and hence, bandwidth congestion on the network. This will be achieved by topology-aware mapping of tasks in an application. By placing communication tasks on processors which are in physical proximity on the network, communication can be restricted to near neighbors. This reduces link-sharing among messages and leads to a better utilization of the available bandwidth. Our aim is to minimize hop-bytes, which is a weighted sum of the number of hops between the source and destination for all messages, the weights being the message sizes. This can minimize the communication time and hence, lead to significant speed-ups for parallel applications and in certain cases, also remove scaling bottlenecks. The research will involve developing a general automatic topology-aware mapping framework which takes the task graph and processor graph as input, and outputs near-optimal mapping solutions.

Problem and Motivation:

The network topology of the largest and most scalable supercomputers today is a three-dimensional (3D) torus. Some examples are Cray's XT family (XT3, XT4 and XT5) and IBM's Blue Gene family (Blue Gene/L and Blue Gene/P). For large installations of such machines, the diameter of the network can be large (somewhere between 20 to 60 hops for Blue Gene/P and XT4.) This can have a significant effect on message latencies. When multiple messages start sharing network resources, this effect becomes more pronounced due to network congestion. Thus, it becomes necessary to consider the topology of the machine while mapping tasks to processors.

This research will demonstrate that it is not wise to assume that message latencies are independent of the distance a message travels. For a number of years, this assumption has been supported by the advantages of virtual cut-through and wormhole routing, suggesting that the message latency is independent of the distance in absence of blocking [McKinley93]. When virtual cut-through or wormhole routing is deployed, message latency is modeled by the equation,

$$L_f/B*D + L/B \qquad \ldots (1)$$

where L_f is the length of each flit, B is the link bandwidth, D is the number of links (hops) traversed and L is the length of the message. In absence of blocking, for sufficiently large messages (where $L_f << L$), the first term is very small compared to the second. But with large diameters of very large supercomputers, this is no longer true for small to medium-sized messages.

Equation (1) models message latencies in the absence of contention. In the case where multiple messages share the same network links, the situation becomes more complex. The bandwidth available per link per message is reduced since it is now being shared. The phenomenon of network resource sharing leading to contention can be explained with a simple example. Let us consider a 3D torus network of size 8 X 8 X 8. The total number of uni-directional links on the system is 512 X 6 = 3072. The diameter of this network is 4 + 4 + 4 = 12 and hence, if messages travel from one random node to another, they will traverse 6 hops on average. Now, if we have four cores per node and every processor sends a message at the same time, all these messages require 512 X 4 X 6 = 12288 links in total and hence, every link will be used for four messages on average. This leads to contention for each link and therefore increases message latencies. Describing this scenario in terms of bandwidth requirements, in order to operate at minimum latency, we need four times the total raw bandwidth available. However, this is not the case and thus the delivered bandwidth is one-fourth of the peak.

The first part of this research involves using simple MPI benchmarks for an extensive study of message latencies and their dependence on distance (hops) on several machines -- Cray's XT family, IBM's Blue Gene family and clusters like Ranger. As we shall see in the Results section, in presence of contention, the dependence of message latencies on hops becomes quite significant, especially for large-sized messages. Hence, it is important to consider topology of the machine, especially of 3D torus/mesh interconnects, to obtain the best performance.

We will also demonstrate, with simple benchmarks and production codes, that topology mapping can significantly improve performance and scaling of communication-bound applications. The metric we will employ to assess the success of topology-aware schemes will be hop-bytes. Hop-bytes are the weighted sum of the number of hops between the source and destination for all messages, with the weights being the message size. This metric gives an indication of the total communication traffic on the network due to an application. Reducing the total hop-bytes reduces link sharing and contention, thereby keeping message latencies close to the ideal.

Building on our successes at mapping individual applications, we aim to develop an automatic, topology-aware mapping framework. Given the communication information of an application and topology of a machine, this framework will produce intelligent mappings to minimize communication traffic. The mapping problem can be reduced to the graph-embedding problem, which is NP-complete. Hence, we will develop a set of heuristics, each dealing with a different scenario, which together will be able to deal with most parallel applications and their communication requirements. In an effort towards making mapping techniques scalable to very large machines, we will also discuss strategies for completely distributed and hybrid mapping strategies.

Background and Related Work:

The problem of topology-aware mapping has been studied extensively and proved to be NP-complete [Bokhari81, Ercal87]. Pioneering work in this area was done by Bokhari in 1981, where he used pairwise exchanges between nodes to arrive at good solutions [Bokhari81]. Most of the techniques developed in the 80s took a long time to arrive at the solution and hence, cannot be used for a time-efficient mapping during runtime. They are almost never used in practice. Heuristic techniques such as pairwise exchanges are theoretical studies with no results on real machines. Also, most of these techniques (heuristics techniques especially) were developed specifically for hypercubes, shuffle-exchange networks or array processors.

Recent emergence of very large parallel machines has led to the necessity of topology mapping again. Most work from the 80s cannot be used in the present context because of unscalable techniques and different topologies from the ones being used today. As mentioned earlier, increasing effect of the number of hops on message latencies has fuelled such studies again. To the best of our knowledge, there has been no published research reporting network contention or quantifying it for the Cray XT family. This research is pioneering work on topology-aware research on the Cray machines. One contribution of this work is an API for providing topology information on Cray machines and demonstrating that topology-aware mapping can lead to improvements on Cray machines also.

Contrary to Cray, IBM systems like Blue Gene/L and Blue Gene/P acknowledge the dependence of message latencies on distance and encourage application developers to use topology of these machines to their advantage. On Blue Gene/L, there is a 89 nanoseconds per hop latency attributed to the torus logic and wire delays. This fact has been used both by system developers [Walkup04, Petrini04] and application developers to improve performance on Blue Gene/L [Bhanot05, Bhatele08a]. Bhanot et. al have developed a framework for optimizing task layout on BG/L. Since it uses simulated annealing, it is quite slow and the solution is developed offline.

Uniqueness of the Approach:

This work is among the first to discuss the effects of contention on Cray and IBM machines and to compare across multiple architectures. We believe that the set of benchmarks we have developed for quantifying message latencies would be useful for the HPC community to assess latencies on a supercomputer and to determine the message sizes for which number of hops makes a significant difference. The effective bandwidth benchmark in the HPC Challenge benchmark suite measures the total bandwidth available on a system but does not analyze the effects of distance or contention on message latencies.

Our experience in developing mapping algorithms for production codes and insights discussed in this work will be useful to individual application writers trying to scale their codes to large supercomputers. We believe that the proposed automatic mapping framework will be applicable to a wide variety of communication scenarios and will relieve the application writers from the burden of finding correct mapping solutions for their codes. Unlike most of the previous work, this research handles both cardinality and topological variations in the graphs. It also handles various architectures and different kinds of communication scenarios (through the use of heuristics). Therefore, it will be useful to a large body of applications running on large parallel machines.

Results and Contributions:

Contention Studies: We wrote a MPI benchmark called WICON (With Contention) to quantify message latencies in presence of contention, which is a regime not handled by the basic model of wormhole routing discussed earlier. In this benchmark, all MPI tasks are grouped into pairs and the smaller rank in the pair sends messages of size B bytes to its partner and awaits a reply. All pairs do this communication simultaneously. The average time for the message sends is recorded for different message sizes. To quantify the effect of hops on message latencies this benchmark is run in two modes:

- Near Neighbor Mode (NN): The ranks which form a pair only differ by one. This ensures that everyone is sending messages only 1 hop away (in a torus).
- Random Processor Mode (RND): The pairs are chosen randomly and thus they are separated by a random number of links.

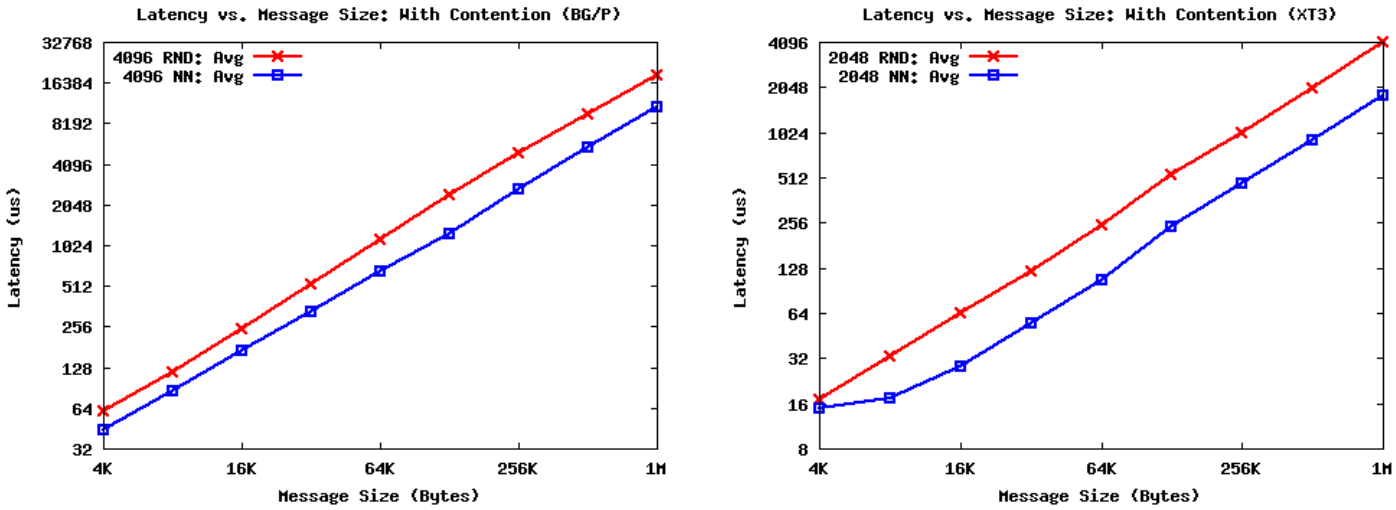

Figure 1. Plots showing the results of the WICON benchmark on Blue Gene/P and XT3

Figure 1 shows the results of running WICON in the NN and RND modes on Blue Gene/P and XT3. The first plot shows the results of WICON on 4,096 cores of BG/P. It is clear that the random-processor (RND) latencies are more than the near-neighbor (NN) latencies (by a factor of 1.75 for large messages.) This is expected based on the assertion that hops have a significant impact on the message latencies in the presence of contention, which increases with larger messages because of a proportional increase in packets on the network. Similar experiments were repeated on XT3 to understand the effects of contention on Cray XT machines. The second plot in Figure 1 presents the results for WICON benchmark on 2,048 cores of XT3. We see a significant difference between the NN and RND lines (a factor of 2.25 at 1 MB messages which is greater than that on BG/P.) This is not unexpected and a quantum chemistry code has shown huge benefits (up to 40%) from topology-aware mapping on XT3 [Bohm07].

The benchmark in the previous section injects random contention on the network. To quantify the effects of contention under controlled conditions, WICON was modified to conduct a controlled experiment. Again, all ranks are divided into pairs but now the pairs are chosen such that they are a fixed number of hops, say n, away from each other. All pairs send messages simultaneously and the average time for message sends of different sizes for varying hops is recorded. Pairs are chosen only along one dimension of the torus, in this case, the Z dimension.

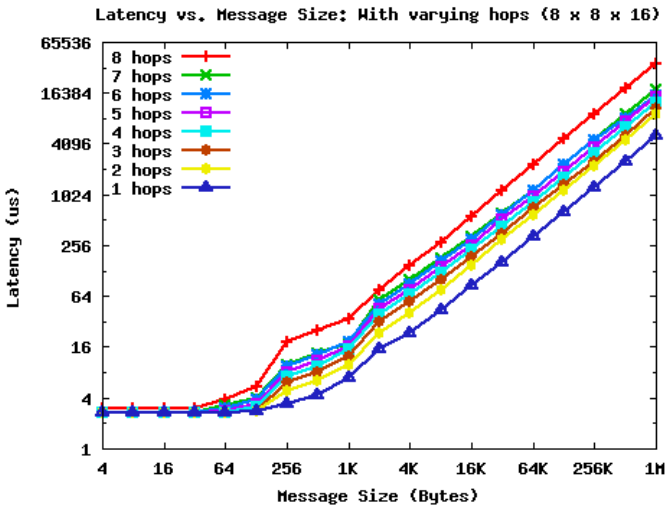

Figure 2. Plots showing the results of WICON2 on BG/P

Figure 2 shows the results of running the WICON2 benchmark on Blue Gene/P. On each plot there are several lines, one each for a specific pairing which is n hops away. The tests were done on a torus of dimensions 8 X 8 X 16. Since messages are sent along Z, maximum number of hops possible is 8 and hence there are 8 lines on the plot. The Blue Gene/P plot on the right shows that the message latencies for large messages for the 1 hop and 8 hops case can differ by a factor of 8!

As all messages travel more hops, links are shared by a greater number of messages, increasing the contention on the network and decreasing the available effective bandwidth. This is what applications have to deal with during communication in practice. This huge difference between message latencies indicates that it is very important to keep communicating tasks

close by and minimize contention on the network [Bhatele09a]. This is especially true for communication bound applications. Next, we look at mapping successes for two applications which have different communication characterstics.

Dynamic Irregular Communication: NAMD is a production Molecular Dynamics (MD) application used for simulation of bio-molecules [Bhatele08b]. NAMD is parallelized by use of Charm++ objects called patches and computes. The simulation box is spatially divided into smaller cells called patches. The force calculation for every pair of patches is assigned to a different compute. Thus, communication in NAMD consists of section multicasts from patches to computes and back. Every patch multicasts its atom data to multiple computes, whereas each compute receives data from only two patches. Patches are statically assigned to a few processors during start-up and computes are distributed evenly by a load balancer. Let us now see the deployment of topology-aware techniques in the static placement of patches and the load balancers.

Topology placement of patches: Since patches form a geometric decomposition of the simulation space, they constitute a 3D group of objects which can be mapped nicely onto the 3D torus of machines. An ORB (Orthogonal Recursive Bisection) of the torus is used to obtain partitions equal in number to the patches and then, a one-to-one mapping of the patches to the processor partitions is done.

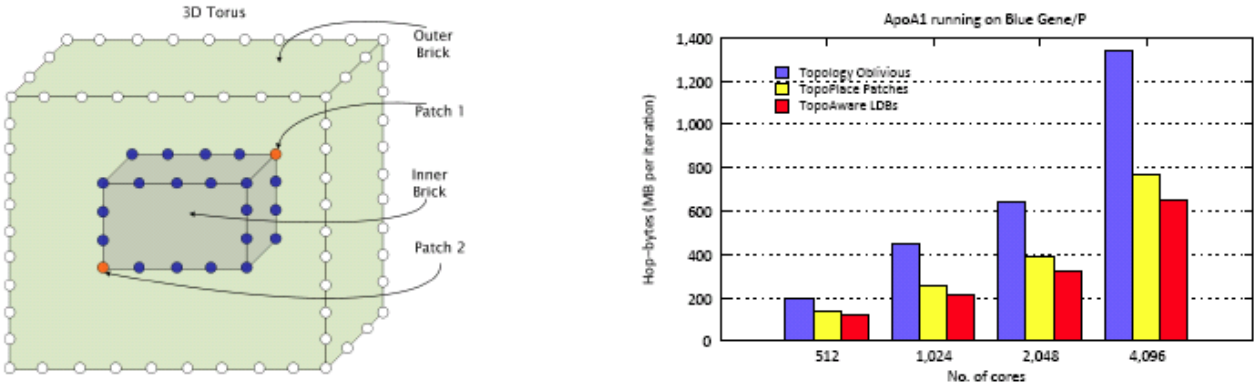

Figure 3. (a) Placement of a compute withinn the inner brick, (b) Improvement in hop-bytes using topology-aware mapping on Blue Gene/P

Topology-aware Load Balancers: Once patches have been statically assigned onto the processor torus, computes which interact with these patches should be placed around them. Consider Figure 3(a), which shows the entire 3D torus on which the job is running. When placing a compute, it should be placed topologically close to the two processors that house the patches it interacts with. The two patches define a smaller brick within the 3D torus (shown in dark grey in the figure). The sum of distances from any processor within this brick to the two patches is minimum.

Figure 3(b) shows the hop-bytes for all messages per iteration when running NAMD on Blue Gene/P on different sized partitions. A standard benchmark used in the MD community was used for the runs: 92,227-atom ApoLipoprotein-A1 (ApoA1). As we would expect, hop-bytes consistently increase as we go from a smaller partition to a larger one. The three strategies compared are: topology oblivious mapping of patches and computes (Topology Oblivious), topology-aware static placement of patches (TopoPlace Patches) and topology-aware placement for both patches and load balancing for computes (TopoAware LDBs).

No. of cores	512	1024	2048	4096	8192	16384
Topology Oblivious	13.93	7.96	5.40	5.31	-	-
TopoPlace Patches	13.85	7.87	4.57	3.07	2.33	1.74
TopoAware LDBs	13.57	7.79	4.47	2.88	2.03	1.25

Topology aware schemes for the placement of patches and the load balancer help in reducing the hop-bytes for all processor counts. Also, the decrease in hop-bytes becomes more significant as we go to larger-sized partitions. This is due to the fact that the average distance traveled by each message increases as we increase the partition size in the case of default mapping; however, it becomes controlled when we do a topology-aware mapping. Since the actual performance of the load balancers depends on several metrics, the question remains as to whether the reduction in hop-bytes leads to an actual improvement in performance. As it turns out, we also see a reduction in the number of proxies and in the max-to-average ratio for topology-aware load balancers, which is reflected in the overall performance of NAMD on Blue Gene/P (see table above). The topology oblivious scheme stops scaling around 4,096 cores and hence, we did not obtain numbers for it beyond that. We see an improvement of 10% at 16,384 cores with the use of topology-aware load balancers. For further details, please refer to [Bhatele09b].

Static Regular Communication: OpenAtom is a fine-grained parallelization of the CPAIMD method to understand dynamics of atoms at a quantum scale [Bohm07]. Computation in OpenAtom is divided into a large number of objects, enabling scaling to tens of thousands of processors. Calculating the electrostatic energy involves computing several terms. Hence, CPAIMD computations involve a large number of phases with high inter-processor communication. These phases are discretized into a large number of objects, which generate a lot of communication, but ensures efficient interleaving of work. The entire computation is divided into ten phases, which are parallelized by decomposing the physical system into fifteen chare arrays.

Since multiple chare arrays interact among one another, the communication dependencies are complex and mapping is a challenging task. OpenAtom provides us with a scenario where the load on each object is static (under the CPAIMD method) and the communication is regular and clearly understood. Hence, it should be possible to intelligently map the arrays in this application to minimize inter-processor communication and maintain load balance. Because of space limitations and a fairly involved mapping acheme, we will not go into the details of how the mapping is done but present performance results.

	WATER_32M_70Ry		WATER_256M_70Ry	
Cores	Default	Topology	Default	Topology
256	0.395	0.324	-	-
512	0.248	0.205	-	-
1024	0.188	0.127	10.78	6.70
2048	0.129	0.095	6.85	3.77
4096	0.114	0.067	4.21	2.17
8192	-	-	3.52	1.77

	WATER_32M_70Ry		WATER_256M_70Ry	
Cores	Default	Topology	Default	Topology
256	0.226	0.196	-	-
512	0.179	0.161	7.50	6.58
1024	0.144	0.114	5.70	4.14
2048	0.135	0.095	3.94	2.43

We studied the strong scaling (fixed problem size) performance of OpenAtom with and without topology aware mapping. Two benchmarks commonly used in the CPMD community: the minimization of WATER_32M_70Ry and WATER_256M_70Ry were used. As shown in the table on the left, performance improvements from topology-aware mapping for Blue Gene/P (BG/P) can be quite significant. As the number of cores and likewise, the diameter of the torus grows, the performance impact increases until there is 40% improvement for WATER_32M_70Ry at 4096 and 50% for WATER_256M_70Ry at 8192 cores. The improvements from topological awareness on Cray XT3, presented in the table on the right, are comparable to those on BG/P. There is an improvement of 20% on XT3 for WATER_256_70Ry at 1024 cores, compared to the improvement of 38% on BG/P at 1024 cores.

Contributions: Significant work was done in the 80s on topology-aware mapping but it was directed towards interconnect topologies like hypercubes and shuffle-exchange networks which are not used in practice today. Work done in those years was directed towards machines consisting of a few hundred processors. The proposed work is highly relevant for the machines of the petascale era, such as Blue Gene and XT. Apart from MPI applications, it is directed towards applications with virtualization (having multiple objects per physical processor and multiple object graphs), which has not been explored previously. As opposed to earlier research in this area, this work is directed towards high scalability and fast runtime solutions.

This research demonstrates that topology-aware mapping is important for communication bound applications. Results quantifying message latencies in presence of contention will be useful to application writers trying to optimize their codes. Application-specific techniques discussed here are being used in production codes, NAMD and OpenAtom, and have helped scientists in getting their results faster. The TopoManager API which obtains the topology information about machines at runtime is already being used by other application groups (such as the US Lattice QCD collaboration). It is especially useful because information on Cray machines is not readily available and this API provides a single wrapper for different machines. We hope that this API will be eventually used to implement MPI_Cart_create and other virtual topology functions on Cray machines.

The final goal of this research is to utilize the experience from application-specific mapping, in developing an automatic framework which can develop topology-aware mapping solutions. The work on the automatic mapping framework will relieve the application writers of doing the mapping themselves. We also foresee acceptance of the idea that applications running on Cray XT machines will benefit as much as those on Blue Gene machines. This might even led to modification of the batch schedulers on these machines to allocate contiguous 3D mesh partitions for jobs.

References:

[McKinley93] Lionel M. Ni and Philip K. McKinley, A Survey of Wormhole Routing Techniques in Direct Networks, IEEE Computer, 26(2): 62-76, 1993.

[Bokhari81] Shahid H. Bokhari, On the Mapping Problem, IEEE Trans. Computers, 30(3): 207-214, 1981.

[Ercal87] P. Sadayappan and F. Ercal, Nearest-Neighbor Mapping of Finite Element Graphs onto Processor Meshes, IEEE Trans. Computers, 36(12): 1408-1424, 1987.

[Walkup04] George Almasi and Siddhartha Chatterjee and Alan Gara and John Gunnels and Manish Gupta and Amy Henning and Jose E. Moreira and Bob Walkup, Unlocking the Performance of the Blue Gene/L Supercomputer, Proceedings of the 2004 ACM/IEEE conference on Supercomputing, 2004.

[Petrini04] Kei Davis and Adolfy Hoisie and Greg Johnson and Darren J. Kerbyson and Mike Lang and Scott Pakin and Fabrizio Petrini, A Performance and Scalability Analysis of the Blue Gene/L Architecture, Proceedings of the 2004 ACM/IEEE conference on Supercomputing, 2004.

[Bhanot05] G. Bhanot and A. Gara and P. Heidelberger and E. Lawless and J. C. Sexton and R. Walkup, Optimizing task layout on the Blue Gene/L supercomputer, IBM Journal of Research and Development, 49 (2/3):489-500, 2005.

[Bhatele08a] Abhinav Bhatele and Laxmikant V. Kale, Benefits of Topology Aware Mapping for Mesh Interconnects, Parallel Processing Letters (Special issue on Large-Scale Parallel Processing), 18(4): 549-566, 2008.

[Bohm07] Eric Bohm and Glenn J. Martyna and Abhinav Bhatele and Sameer Kumar and Laxmikant V. Kale and John A. Gunnels and Mark E. Tuckerman, Fine Grained Parallelization of the Car-Parrinello ab initio MD Method on Blue Gene/L, IBM Journal of Research and Development, 52 (1/2): 159-174, 2008.

[Bhatele09a] Abhinav Bhatele and Laxmikant V. Kale, An Evaluation of the Effect of Interconnect Topologies on Message Latencies in Large Supercomputers, To appear in Proceedings of Workshop on Large-Scale Parallel Processing (IPDPS), 2009.

[Bhatele08b] Abhinav Bhatele and Sameer Kumar and Chao Mei and James C. Phillips and Gengbin Zheng and Laxmikant V. Kale, Overcoming Scaling Challenges in Biomolecular Simulations across Multiple Platforms, Proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2008.

[Bhatele09b] Abhinav Bhatele and Laxmikant V. Kale and Sameer Kumar, Dynamic Topology Aware Load Balancing Algorithms for Molecular Dynamics Applications, To appear in Proceedings of 23rd ACM International Conference on Supercomputing, 2009.