An Evaluative Study on the Effect of Contention on Message Latencies in Large Supercomputers

Abhinav Bhatelé and Laxmikant V. Kalé
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
Email: {bhatele, kale}@illinois.edu

***Abstract*— Significant theoretical research was done on interconnect topologies and topology aware mapping for parallel computers in the 80s. With the deployment of virtual cut-through, wormhole routing and faster interconnects, message latencies reduced and research in the area died down. This paper presents a study showing that with the emergence of very large supercomputers, typically connected as a 3D torus or mesh, topology effects have become important again. It presents an evaluative study on the effect of contention on message latencies on torus and mesh networks.**

The paper uses three MPI benchmarks to evaluate the effect of hops (links) traversed by messages, on their latencies. The benchmarks demonstrate that when multiple messages compete for network resources, link occupancy or contention can increase message latencies by up to a factor of 8 times. In other words, contention leads to increased message latencies and reduces effective available bandwidth for each message. This suggests that application developers should consider interconnect topologies when mapping tasks to processors in order to obtain the best performance. Results are shown for two parallel machines – ANL's Blue Gene/P and PSC's XT3.

I. Introduction

Interconnect topologies and their effect on message latencies in message-passing distributed supercomputers was an important factor determining performance in the 80s. Significant research was done on topology-aware mapping to restrict communication to near-neighbors and optimize performance [1], [2], [3]. With the deployment of virtual cut-through [4] and wormhole routing [5] and emergence of faster interconnects in the 90s, message latencies became relatively unimportant and research reduced in this area.

The network topology of the largest and most scalable supercomputers today is a three dimensional (3D) torus. Some examples are IBM's Blue Gene family and Cray's XT family. 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/XT5.) and this can have a significant effect on message latencies. When multiple messages start sharing network resources, this effect becomes more pronounced, especially for medium to large sized messages. Hence, it becomes necessary to consider the topology of the machine while mapping tasks to processors.

This paper will demonstrate that contention for links by multiple messages can significantly increase message latencies (sometimes up to a factor of 8.) Hence, it might not be wise to ignore the machine topology. Virtual cut-through and wormhole routing suggest that, in absence of contention, message latency is independent of the distance for most message sizes [4], [5]. When virtual cut-through or wormhole routing is deployed, message latency can be modeled by the equation:

$$\frac{L_f}{B} * D + \frac{L}{B} \qquad (1)$$

where L_f is the length of the flit or header packet, 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 and 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.

Moreover when there is contention on the network, distance becomes an important factor affecting message latencies, even with wormhole routing. This is because of sharing of network links between messages. It is often assumed that contention is inconsequential on some of the faster interconnects today and hence application developers should not have to worry about network latencies and hence about topology-aware optimizations. This is evident from the fact that job scheduling on Cray XT machines is not topology-aware (on Blue Gene machines, users are allocated complete tori for their jobs). Also, there is no easy mechanism to obtain topology information on XT machines and for the same reason the `MPI_Cart` functions are not implemented efficiently. This paper will demonstrate that an application does not have to utilize close to the available bandwidth to suffer from increased message latencies. Through a simple benchmark which compares near-neighbor to random-processor communication, we will show that as soon as two processors share a common link to send messages, messages take significantly longer to reach their destination.

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 $8 \times 8 \times 8$. The total number of uni-directional links on the system is $512 \times 6 = 3,072$. 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 the average. Now, if we have four processors per node and every processor sends a message at the same time, all these messages require $512 \times 4 \times 6 = 12,288$

links in total and hence every link will be used for four messages on the average. This leads to contention for each link and hence increases message latencies. Describing this scenario in terms of bandwidth requirements, to operate at minimum latency, we need four times the total raw bandwidth available. But that is not the case and hence the delivered bandwidth is one-fourth of the no-load maximum bandwidth. The results will demonstrate that effective bandwidth can decrease by up to 8 times in presence of contention.

The problem of network congestion and efficient PE mappings to avoid it have been explored on the Cray T3D and T3E systems [6], [7], [8]. IBM systems like Blue Gene/L and Blue Gene/P have acknowledged the dependence of message latencies on distance and encourage application developers to use topology of these machines to their advantage [9], [10]. 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 by application developers to improve performance on Blue Gene/L [11], [12], [13]. The authors have also presented improvements from topology mapping for a simple stencil application and a *real* application through a preliminary study [14].

The effect of topology on application performance and the effect of congestion in the network on IBM and Cray systems has been reported by Hoisie et al. [15], although using a different approach. Results in [15] and comparisons of Natural Ring and Random Ring results in HPC Challenge [18] support the findings in this paper. This paper has a detailed study on contention for different message sizes and machines. We believe that the set of benchmarks we have developed would be useful for the HPC community to assess message 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 measure the total bandwidth available on a system but does not analyze the effects of distance or contention on message latencies [18].

We do not consider fat-tree topologies in this paper. Torus topologies are not asymptotically scalable because the raw available bandwidth increases as a function of P, whereas the required bandwidth (assuming communicating processors are randomly chosen) increases as a function of $P^{4/3}$, where P is the number of nodes. In contrast, on fully-provisioned fat-trees, the available bandwidth keeps pace with required bandwidth - the diameter is $logP$ and the number of links is proportional to $P.logP$. However in practice, torus topologies perform well provided that mapping of tasks to processors takes the physical topology into account [12].

II. Parallel Machines

Two large supercomputers, one each from the IBM Blue Gene family and the Cray XT family were used to perform the set of experiments described above. Both are three-dimensional torus or mesh topologies but have different processor speeds and network characteristics.

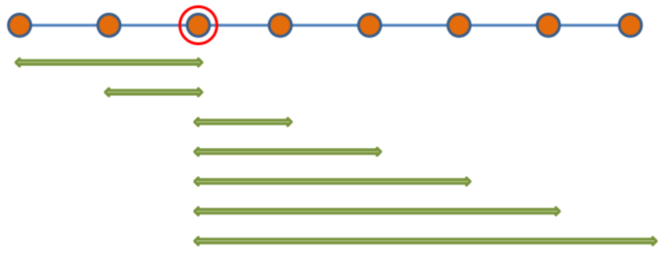


Fig. 1. Communication Patterns in the WOCON Benchmark

IBM Blue Gene/P: The smaller installation of Blue Gene/P, Surveyor at Argonne National Lab (ANL), was used for runs in this paper. It has $1,024$ compute nodes, each of which has four 850 MHz PowerPC cores. The nodes are connected by a low-latency 3D torus network with a uni-directional link bandwidth of 425 MB/s [16]. The nodes use a DMA engine to offload communication on the torus network, leaving the cores free for computation. A midplane composed of 512 nodes forms a torus of size $8 \times 8 \times 8$ in all directions. Smaller allocations than a midplane are a torus in some dimensions and mesh in others. Larger allocations than a midplane are complete tori.

Cray XT3: The other machine used was the XT3 installation (Bigben) at Pittsburgh Supercomputing Center (PSC.) This installation has 2068 compute nodes arranged in a 3D torus of dimensions $11 \times 12 \times 16$. Each node has two 2.6 GHz AMD Opteron processors and the nodes are connected by a custom SeaStar interconnect. The processors are connected to the SeaStar chip through a Hyper Transport (HT) link. The unidirectional bandwidth of the HT link is ~ 1.6 GB/s whereas that of the network links is 3.8 GB/s [17]. Since the job scheduler on XT3 does not allocate cuboidal partitions, nodes allocated for a particular job may not be contiguous. For results reported in this paper, the whole machine was reserved and then nodes were allocated (with help from PSC staff) to get contiguous cuboidal shapes. These partitions did not have any IO nodes or failed nodes in the middle. The largest partition used was $8 \times 8 \times 16$ which is 1024 nodes or 2048 cores and smaller sub-partitions were made from this one. The 1024 node partition has torus links in one dimension (which is of size 16) and mesh links in the other two. For any allocation smaller than 1024 nodes, we had a mesh in all dimensions.

A set of benchmarks were developed to test the claims made in this paper. The next three sections discuss these benchmarks and the results obtained from running them. Finally we compare across the two machines and provide broad conclusions from this work.

III. WOCON: No Contention Benchmark

This benchmark records message latencies for varying number of hops in absence of contention. One particular node is chosen from the allocated partition to control the execution. We will call this node the master node or master rank. It sends B-byte messages to every other node in the partition,

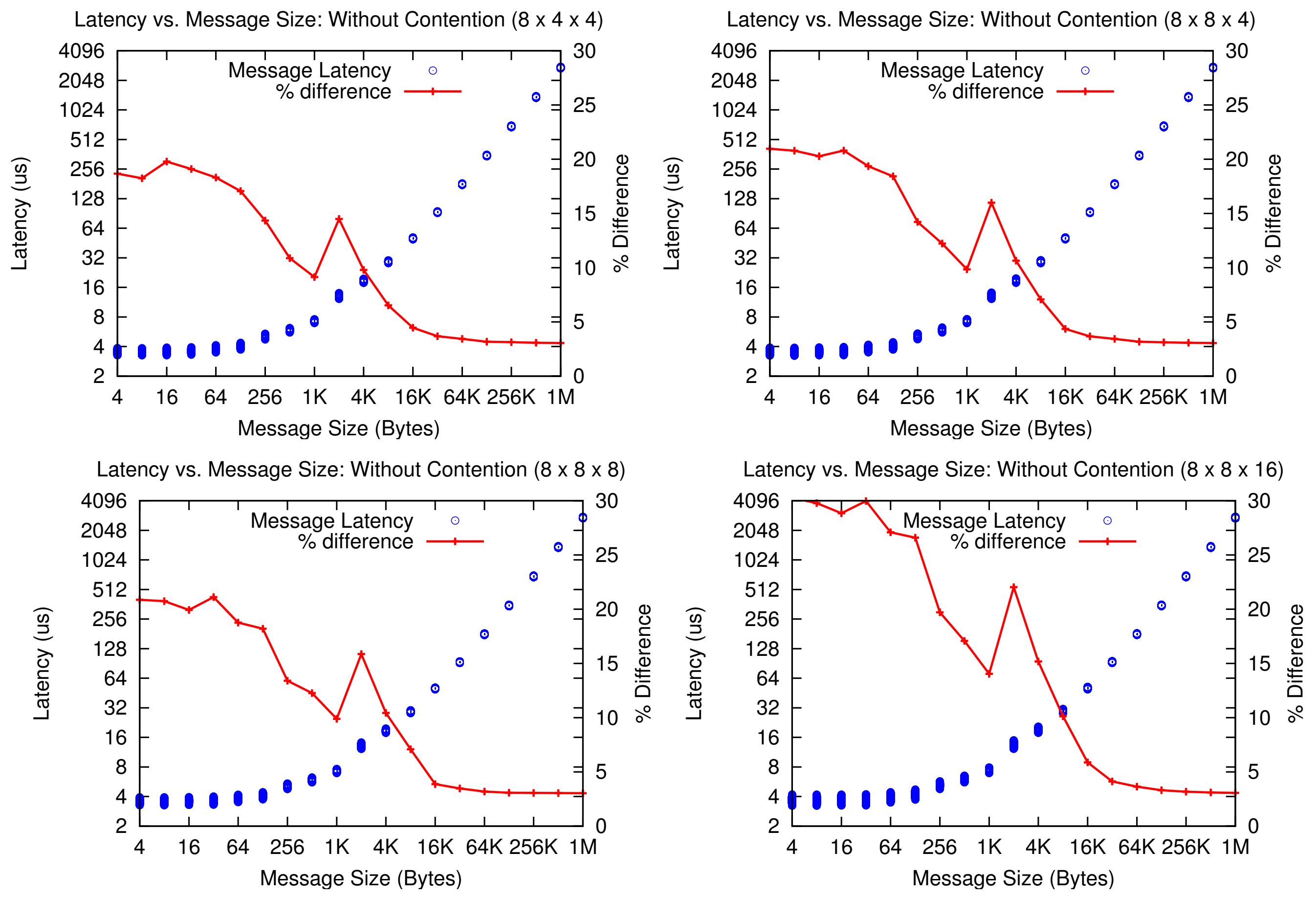


Fig. 2. Plots showing the effect of hops on message latencies in absence of contention (for torus sizes ranging from $4 \times 4 \times 4$ to $8 \times 8 \times 16$ on Blue Gene/P, Benchmark: `WOCON`)

and expects same-sized messages in return (Figure 1). The messages to each node are sent sequentially, one message at a time (pseudo-code in Figure 3). For machines with multiple cores per node, this benchmark places just one MPI task per node to avoid intra-node messaging effects. The size of the message, B is varied and for each value of B, the average time for sending a message to every other node is recorded. Since the distance from the master node to other nodes varies, we should see different message latencies depending on the distance.

Wormhole routing suggests that message latencies are independent of distance in the absence of contention, for sufficiently large message sizes. The benchmark `WOCON` was used to quantify the effect of the number of hops on small-sized messages. Figure 2 presents the results obtained from running `WOCON` on four allocations of BG/P, ranging in size from 128 to 1024 nodes (torus sizes $8 \times 4 \times 4$ to $8 \times 8 \times 16$.) There are two patterns on the plot: 1. For each message size on the x-axis, the circles represent the time for a message send from the master rank to different nodes on the allocated partition. Note that the vertical bars are actually a cluster of circles, one each for a message send to a different node; 2. Each point on the line represents the percentage difference between the minimum and maximum time for message send for a particular message size.

Message latencies should vary depending on the distance of the target rank from the master rank for very short messages. As expected, we see a regular pattern for the distribution of circles for a particular message size in the four plots (Figure 2). For small and medium-sized messages, message latencies are spread over a range, the range decreasing with increasing message sizes. This is what one would expect from the wormhole routing model. To have a clearer perception of the range in which message latencies lie, the percentage difference between the minimum and maximum latencies was calculated with respect to the minimum latency for each message size. These values have been plotted as a function of the message size. The difference between the maximum and minimum values (shown by the line) decreases with increasing message size for all the plots. We see a kink in the lines and a corresponding jump in the message latencies at the 2 KB message size. This can be explained by the use of different routing protocols on Blue Gene/P for different message sizes [10]. For message sizes greater than 1200 bytes, the MPI *rendezvous* protocol is used where an initial handshake is done before the actual message is sent.

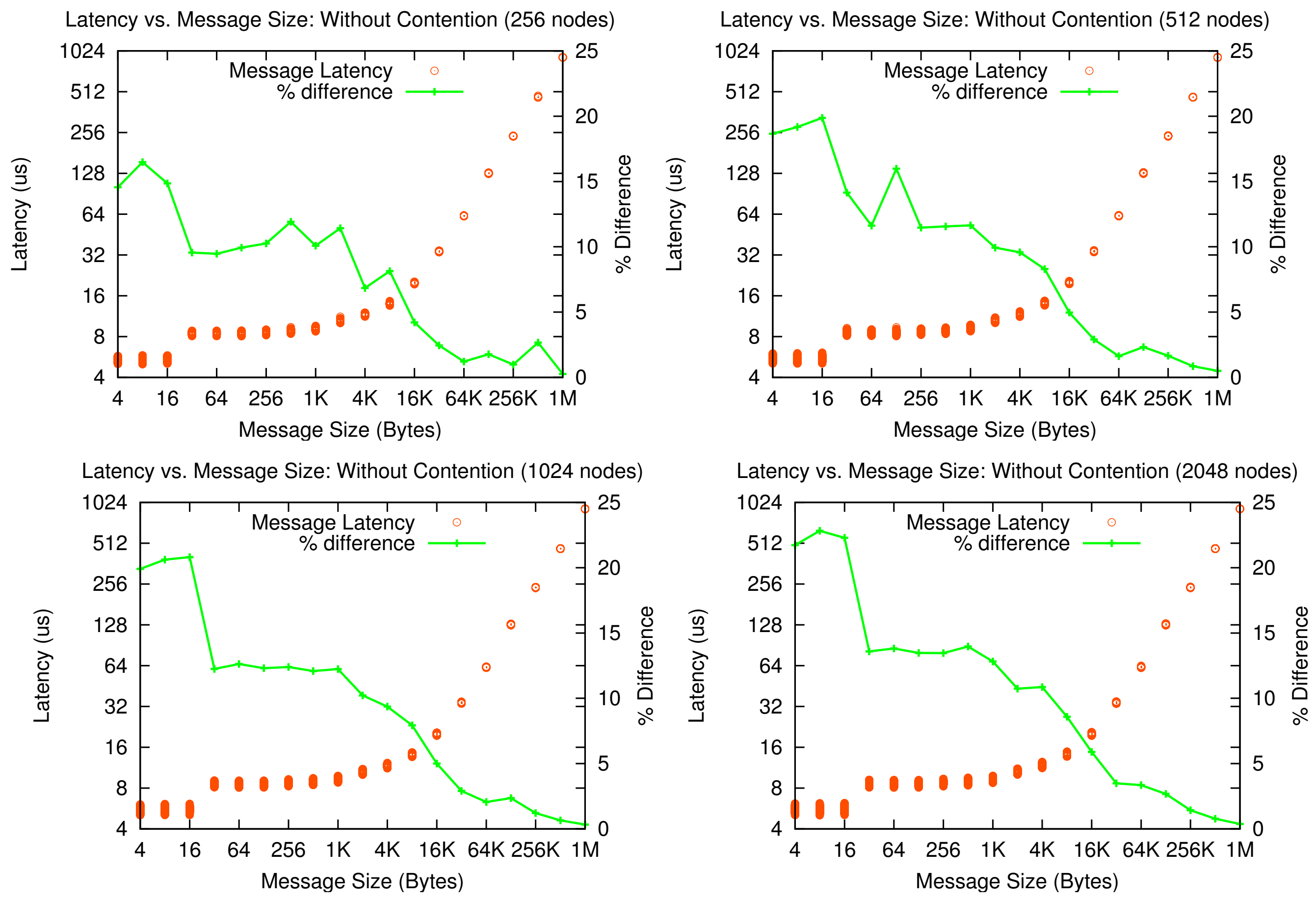


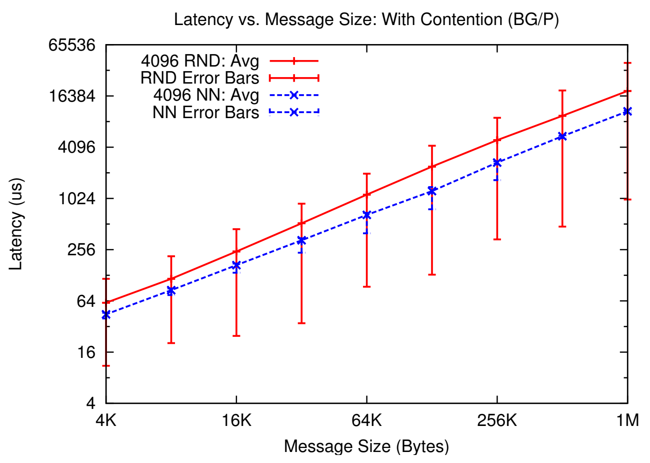
Fig. 4. Plots showing the effect of number of hops on message latencies in absence of contention (for 256 to 2048 nodes of XT3, Benchmark: `WOCON`)

```
if(myrank == MASTER_RANK) {
  for(i=0; i<numprocs; i++) {
    if(i != MASTER_RANK) {
      // warm up

      sendTime = MPI_Wtime();
      for(j=0; j<num_msgs; j++) {
        MPI_Send(send_buf, msg_size, MPI_CHAR, i,
            1, MPI_COMM_WORLD);
        MPI_Recv(recv_buf, msg_size, MPI_CHAR, i,
            1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
      }
      recvTime = MPI_Wtime();
      time[i] = (recvTime-sendTime)/(num_msgs*2);
    }
  }
} else {
  // warm up

  for(i=0; i<num_msgs; i++) {
    MPI_Recv(recv_buf, msg_size, MPI_CHAR, MASTER_RANK,
            1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    MPI_Send(send_buf, msg_size, MPI_CHAR, MASTER_RANK,
            1, MPI_COMM_WORLD);
  }
}
```

Fig. 3. Code fragments showing the core of `WOCON` Benchmark

The important observation is that the difference is in the range of 10 to 30% for message sizes up to 8 KB (in the 1024 nodes plot, Figure 1). Most fine-grained applications use messages which fall in this range and hence it is not wise to blindly assume that message latencies do not depend on hops for most practical message sizes. Strong scaling of problems to very large number of processors also brings us in this range of message sizes. Another observation is that the difference increases for a particular message size with the increase in diameter of the partition. 128 and 256 node partitions are not complete tori in all dimensions and hence their diameter is the same as that of the 512 node partition – 12. The diameter of the 1024 node partition is 16 and hence a steep increase in the percentage difference for the small and medium messages (as an example the % difference for a 64 byte message increases from 19 to 27 as we go from the third plot to fourth). As we increase the size of the partition from 1K to 64K nodes, the diameter would increase from 16 to 64 and we can imagine the impact that will have on message latencies.

Figure 4 shows similar plots for Bigben, the Cray XT3 machine. The XT3 plots were obtained from runs on contiguous allocations of 256 to 2048 nodes of Bigben. Since runs were performed under similar conditions on XT3 as on‘ BG/P, we would expect similar results. As expected, dependence on hops is significant for message sizes up to 8 KB as seen by the lines on the plots. The only difference from the BG/P numbers is



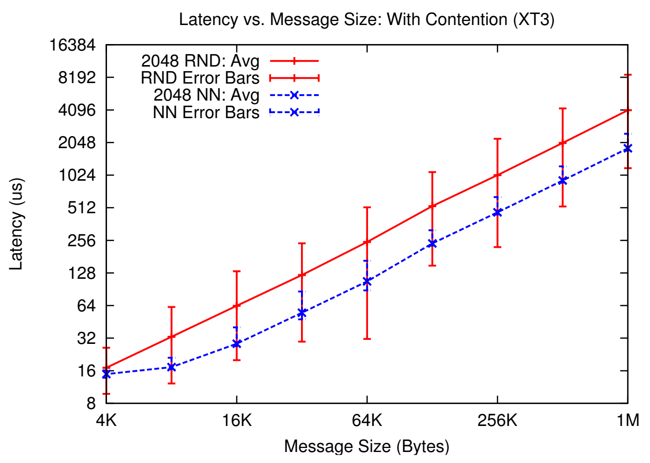


Fig. 5. Plots showing the results of `WICON` on Blue Gene/P and XT3

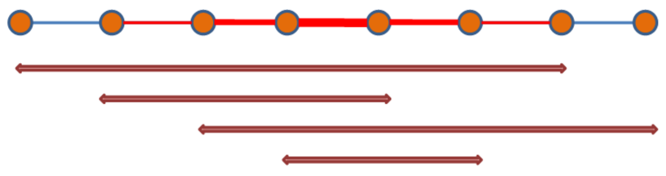


Fig. 6. Communication Patterns in the WICON Benchmark

that message latencies on XT3 are significantly greater than the observed latencies on BG/P for very small messages. This will be discussed in detail in Section VI.

IV. WICON: Random Contention Benchmark

The second benchmark is used to quantify message latencies in presence of contention which is a regime not handled by the basic model of wormhole routing discussed earlier. It should be noted that unlike `WOCON`, this benchmark places one MPI task on each core to create as much contention as possible. 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 (Figure 6). The average time for the message sends is recorded for different message sizes (pseudo-code in Figure 7). 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.

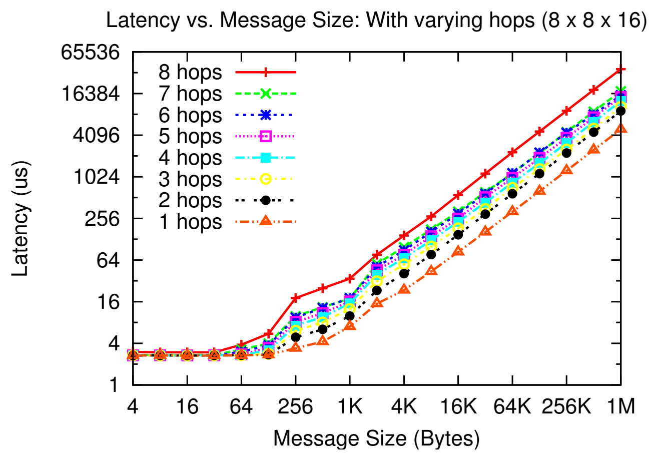
Figure 5 shows the results of running `WICON` in the NN and RND modes on Blue Gene/P and XT3. The first plot shows the results of WICON on $4,096$ cores of BG/P. It is clear that the random-processor (RND) latencies are more than the near-neighbor (NN) latencies (by a factor of 1.75 for large messages.) This is expected based on the assertion that hops have a significant impact on the message latencies

```
pe = partner[myrank];
if(myrank < pe) {
  // warmup

  sendTime = MPI_Wtime();
  for(i=0; i<NUM_MSGS; i++)
    MPI_Send(send_buf, msg_size, MPI_CHAR, pe,
          1, MPI_COMM_WORLD);
  for(i=0; i<NUM_MSGS; i++)
    MPI_Recv(recv_buf, msg_size, MPI_CHAR, pe,
          1, MPI_COMM_WORLD, &mstat);
  recvTime = (MPI_Wtime() - sendTime) / NUM_MSGS;
  // cooldown
} else {
  // warmup

  sendTime = MPI_Wtime();
  for(i=0; i<NUM_MSGS; i++)
    MPI_Recv(recv_buf, msg_size, MPI_CHAR, pe,
          1, MPI_COMM_WORLD, &mstat);
  for(i=0; i<NUM_MSGS; i++)
    MPI_Send(send_buf, msg_size, MPI_CHAR, pe,
          1, MPI_COMM_WORLD);
  recvTime = (MPI_Wtime() - sendTime) / NUM_MSGS;
  // cooldown
}
```

Fig. 7. Code fragments showing the core of `WICON` Benchmark

in the presence of contention, which increases with larger messages because of a proportional increase in packets on the network. Error bars show the minimum and maximum values for message latencies for the two cases for the different processor pairs. The error bars for the RND case go even below the NN latencies because unlike the NN case, we can have a pair on the same node. This data might be useful for applications where the slowest process determines the overall runtime.

Similar experiments were repeated on XT3 to understand the effects of contention on Cray XT machines. The second plot in Figure 5 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 [12].



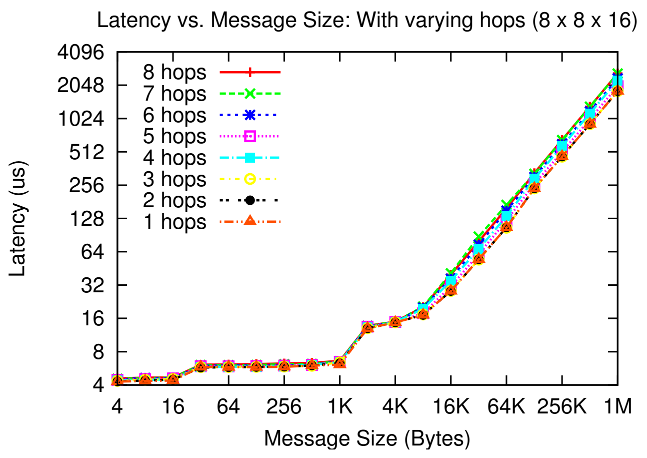


Fig. 8. Plots showing the results of `WICON2` on Blue Gene/P and XT3

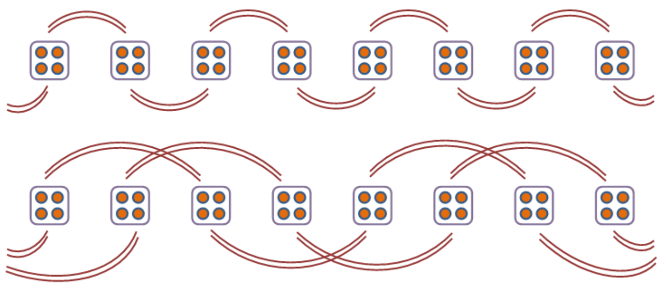


Fig. 9. Communication Patterns in the WICON2 Benchmark

TABLE I
COMPARISON OF MESSAGE LATENCIES (IN MICROSECONDS) ON BLUE GENE/P AND XT3

Msg Size (bytes)	**32**		**1024**		**16384**	
	NN	RND	NN	RND	NN	RND
WOCON BG/P	3.253	3.732	6.947	7.415	49.38	50.82
WOCON XT3	8.116	8.606	8.785	9.348	19.52	20.03
WICON BG/P	5.238	5.823	24.67	37.56	337.1	487.4
WICON XT3	11.48	12.98	13.05	13.99	56.94	127.5

V. WICON2: CONTROLLED EXPERIMENT

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. Again, 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 9).

Figure 8 shows the results of running the `WICON2` benchmark on BG/P and XT3. 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 $8 \times 8 \times 16$. 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 left 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 more and more messages increasing the contention on the network and decreasing the available effective bandwidth. This is what applications have to deal with during communication. This huge difference between message latencies indicates that it is very important to keep communicating tasks close by and minimize contention on the network. This is especially true for communication bound applications.

The second plot shows the results from the same benchmark on XT3. In this case, the difference between latencies for large messages is around 2 times. This deviation from the results on BG/P needs further analysis. One possible reason for this might contention for the Hyper Transport (HT) link which connects the nodes to the SeaStar router instead of the network links. Section 3.5 in [15] uses a similar benchmark and demonstrates results similar to ours.

VI. COMPARISON ACROSS MACHINES

It is interesting how the two machines compare against each other in terms of message latencies in absence and presence of contention. Let us keep in mind that XT3 cores are much faster than BG/P cores (2.6 GHz versus 850 MHz) and the network bandwidth on XT3 is much higher than on BG/P (3.8 GB/s versus 450 MB/s) and hence we would expect XT3's network to perform better than BG/P. Table I presents data from the first two benchmarks on these machines for three different message sizes, 32 bytes, 1 KB and 16 KB. All runs were done on 1.024 nodes. These message sizes fall in different regimes where the MPI *short*, *eager* and *rendezvous* routing protocols are used respectively on BG/P.

In the absence of contention, both NN and RND messaging on BG/P is two times as fast as on XT3 for 32 byte messages. They are still faster on BG/P up to 1 KB messages. For the larger 16 KB messages, XT3 becomes 2.5 times faster than BG/P. In presence of contention, for 32 bytes messages, BG/P is still faster than XT3. But this time, XT3 is faster than BG/P for a 1 KB message. Further, the difference is significant: XT3 is twice as fast for NN and thrice as fast for RND messages. As we go to 16 KB messages, the NN message latency on XT3 is six times better than on BG/P. This happens because the NN message latencies rise steeply (up to seven times) in presence of contention for BG/P but not for XT3. The RND latencies for XT3 also four times better than on BG/P.

The case of random-processor (RND) messages in presence of contention is interesting. BG/P shows an expected rise in the message latencies from the no-contention case. But in case of XT3 for 16 KB messages, the rise in RND message latencies is more than the rise in NN message latencies. This leads to a difference of 120% between the NN and RND message latencies on XT3 compared to the 45% difference on BG/P. These results suggest that both Blue Gene/P and XT3 suffer losses with contention due to random neighbors but XT3 behaves relatively better in presence of contention due to multiple cores on neigboring nodes. This might provide some insight into the results of the `WICON2` benchmark on XT3.

VII. Conclusion and Future Work

This paper analyzes the dependence of message latencies on hops in absence and presence of contention. This study is essential to determine if performance improvements can be derived from topology-aware mapping of applications on a machine. If topology-aware mapping is important, one would like to identify the resource for which contention occurs and the methods to avoid such contention in parallel algorithms.

We conclude that in presence of contention, message latencies increase significantly with increasing number of hops messages travel, due to increased contention. The difference between the near-neighbor and farthest node latencies can be as high as 800% sometimes. The results also suggest that topology-aware codes might see more performance improvement on the XT machines than BG/P (for certain message sizes) although the absolute message latencies on those machines are small compared to BG/P. To summarize, both in the absence and presence of contention, hops affect messages latencies to different extents. This fact should not be neglected by assuming that wormhole routing and high bandwidths on the current machines make message latencies small and independent of distance in all practical scenarios.

We wish to extend this work by evaluating other topologies like fat-tree (NCSA's Abe and TACC's Ranger) and the Kautz Graph (SiCortex machines.) Topology might not have such an impact on performance in dynamically routed fat-tree networks and this paper suggests that this needs a further empirical study. We would also like to compare link contention with other factors on the Cray XT machines, such as contention for the HyperTransport link shared between different cores. These issues need further analysis and will lead to a enhanced understanding of interconnect topologies. Such information will be beneficial to application developers writing topology-aware algorithms.

Acknowledgments

This work was supported in part by a DOE Grant B341494 funded by CSAR, DOE grant DE-FG05-08OR23332 through ORNL LCF, and a NIH Grant PHS 5 P41 RR05969-04 for Molecular Dynamics. The authors would like to thank Chad Vizino from the Pittsburgh Supercomputing Center for setting up a reservation on Bigben (under TeraGrid [19] allocation grant ASC050040N supported by NSF) and allocating contiguous partitions for the runs, which is not possible using the default job scheduler. We also used running time on the Blue Gene/P at Argonne National Laboratory, which is supported by DOE under contract DE-AC02-06CH11357. We would also like to thank Matt Reilly from SiCortex Inc., Sameer Kumar from IBM Research, Larry Kaplan from Cray Inc. and Eric Bohm from Parallel Programming Laboratory (UIUC) for the numerous e-mail exchanges/discussions which enhanced our understanding of the subject.

References

[1] Shahid H. Bokhari. On the mapping problem. *IEEE Trans. Computers*, 30(3):207–214, 1981.

[2] S. Wayne Bollinger and Scott F. Midkiff. Processor and link assignment in multicomputers using simulated annealing. In *ICPP (1)*, pages 1–7, 1988.

[3] P. Sadayappan and F. Ercal. Nearest-neighbor mapping of finite element graphs onto processor meshes. *IEEE Trans. Computers*, 36(12):1408–1424, 1987.

[4] James W. Dolter, P. Ramanathan, and Kang G. Shin. Performance analysis of virtual cut-through switching in harts: A hexagonal mesh multicomputer. *IEEE Trans. Comput.*, 40(6):669–680, 1991.

[5] Lionel M. Ni and Philip K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62–76, 1993.

[6] Thierry Cornu and Michel Pahud. Contention in the Cray T3D Communication Network. In *Euro-Par '96: Proceedings of the Second International Euro-Par Conference on Parallel Processing-Volume II*, pages 689–696, London, UK, 1996. Springer-Verlag.

[7] M. Muller and Michael Resch. PE mapping and the congestion problem in the T3E. In *Proceedings of the Fourth European Cray-SGI MPP Workshop*, Garching, Germany, 1998.

[8] Eduardo Huedo, Manuel Prieto, Ignacio Martín Llorente, and Francisco Tirado. Impact of PE Mapping on Cray T3E Message-Passing Performance. In *Euro-Par '00: Proceedings from the 6th International Euro-Par Conference on Parallel Processing*, pages 199–207, London, UK, 2000. Springer-Verlag.

[9] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas. Blue Gene/L torus interconnection network. *IBM Journal of Research and Development*, 49(2/3), 2005.

[10] IBM System Blue Gene Solution. Blue Gene/P Application Development Redbook, 2008. http://www.redbooks.ibm.com/abstracts/sg247287.html.

[11] Francois Gygi, Erik W. Draeger, Martin Schulz, Bronis R. De Supinski, John A. Gunnels, Vernon Austel, James C. Sexton, Franz Franchetti, Stefan Kral, Christoph Ueberhuber, and Juergen Lorenz. Large-Scale Electronic Structure Calculations of High-Z Metals on the Blue Gene/L Platform. In *Proceedings of the International Conference in Supercomputing*. ACM Press, 2006.

[12] Eric Bohm, Glenn J. Martyna, Abhinav Bhatele, Sameer Kumar, Laxmikant V. Kale, 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: Applications of Massively Parallel Systems*, 52(1/2):159–174, 2008.

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

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

[15] Adolfy Hoisie, Greg Johnson, Darren J. Kerbyson, Michael Lang, and Scott Pakin. A performance comparison through benchmarking and modeling of three leading supercomputers: Blue gene/l, red storm, and purple. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 74, New York, NY, USA, 2006. ACM.

[16] IBM Blue Gene Team. Overview of the IBM Blue Gene/P project. *IBM Journal of Research and Development*, 52(1/2), 2008.

[17] Cray Inc. Scalable Computing at Work: Cray XT4 Datasheet, 2006. www.cray.com/downloads/Cray_XT4_Datasheet.pdf.

[18] Jack Dongarra and P Luszczek. Introduction to the HPC Challenge Benchmark Suite. Technical Report UT-CS-05-544, University of Tennessee, Dept. of Computer Science, 2005.

[19] C. Catlett and et. al. TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications. In Lucio Grandinetti, editor, *HPC and Grids in Action*, Amsterdam, 2007. IOS Press.