Parallel Processing Letters


# QUANTIFYING NETWORK CONTENTION ON LARGE PARALLEL MACHINES

Abhinav Bhatelé *

*Department of Computer Science*
*University of Illinois at Urbana-Champaign*
*Urbana, Illinois 61801, USA*

Laxmikant V. Kalé

*Department of Computer Science*
*University of Illinois at Urbana-Champaign*
*Urbana, Illinois 61801, USA*

Received July 2009
Revised August 2009
Communicated by A. K. Jones

ABSTRACT

In the early years of parallel computing research, significant theoretical studies were done on interconnect topologies and topology aware mapping for parallel computers. With the deployment of virtual cut-through, wormhole routing and faster interconnects, message latencies reduced and research in the area died down. This article shows that network topology has become important again with the emergence of very large supercomputers, typically connected as a 3D torus or mesh. It presents a quantitative study on the effect of contention on message latencies on torus and mesh networks.
Several MPI benchmarks are used 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 on some architectures. Results are shown for three parallel machines – ANL's IBM Blue Gene/P (Surveyor), ORNL's Cray XT4 (Jaguar) and PSC's Cray XT3 (BigBen). Findings in this article suggest that application developers should now consider interconnect topologies when mapping tasks to processors in order to obtain the best performance on large parallel machines.

*Keywords*: contention, link sharing, torus interconnects, MPI, topology aware mapping

## 1. Introduction

Interconnect topologies and their effect on message latencies in message-passing distributed supercomputers was an important factor determining performance in the

*The authors can be contacted by electronic mail at: bhatele@illinois.edu

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:

$$\frac{L_f}{B} * D + \frac{L}{B} \tag{1}$$

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. Let us say that the length of the flit is 32 bytes and the total length of the message is 1024 bytes. Now, if the message has to traverse 8 links, the first term is not negligible compared to the second one (it is one-fourth of the second term). Message sizes around 1 KB are found in several applications which deal with strong scaling to tens of thousands of processors [6, 7]. Results from simple MPI benchmarks will show that the number of hops affects message latencies for messages as big as 16 KB on a fair-sized processor partition, even in the absence of contention.

More importantly, 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` func-

tions are not implemented in a topology aware manner. 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 node to another random one, 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 no-load 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 [8, 9, 10]. 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 [11, 12]. 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 machines [13, 14]. The authors have also presented improvements from topology mapping for a simple stencil application and a *real* application for both IBM Blue Gene and Cray XT machines [15, 16].

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. [17], although using a different approach. Results in [17] 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].

Section 7 of the paper will try to put this research in perspective with regards to application performance. Using a system reservation for allocating contiguous job partitions on the XT3 machine, we will show that applications can perform better if

there is minimal interference from other jobs. And once the job scheduler is topology aware application developers or runtime systems can use this fact to their advantage for topology aware mapping (such as on the IBM Blue Gene systems).

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 [7].

## 2. Parallel Machines

Three large supercomputers, one from the IBM Blue Gene family and two from the Cray XT family were used to perform the set of experiments described above. The interconnect for both families is a three-dimensional torus or mesh but they have different processor speeds and network characteristics.

**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 [19]. 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 $\sim$ 1.6 GB/s whereas that of the network links is 3.8 GB/s [20]. 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 completely cuboidal shapes. A set of benchmarks were developed to test the claims made in this paper. 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.

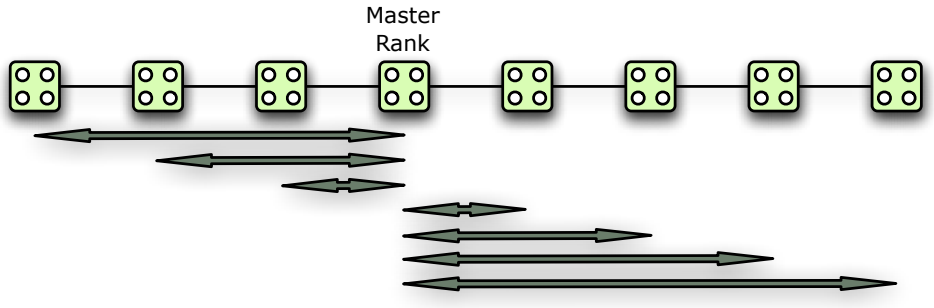


Fig. 1. Communication patterns in the `WOCON` benchmark. This diagram is a simplified one-dimensional version of the pattern in three-dimension (3D). A master ranks sends messages to all ranks in the 3D partition.

```
if(myrank == MASTER_RANK) {
  for(i=0; i<numprocs; i++) {
    if(i != MASTER_RANK) {
      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 {
  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. 2. Code fragments showing the core of `WOCON` Benchmark

**Cray XT4:** For XT4 runs, Jaguar at Oak Ridge National Laboratory (ORNL) was used which has 7,832 XT4 compute nodes, each of which has four 2.1 GHz AMD Opteron processors. Similar to XT3, the nodes are connected by a 3D torus network with a unidirectional link bandwidth of 3.8 GB/S. The bandwidth of the HT transport link in this case is twice that of the XT3, around 3.2 GB/s. Again, for smaller allocations than the whole machine, we do not get a complete torus. Since we did not have a reservation on this machine, the default job submission queue was used which does not guarantee contiguous allocation of nodes for a job.

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.

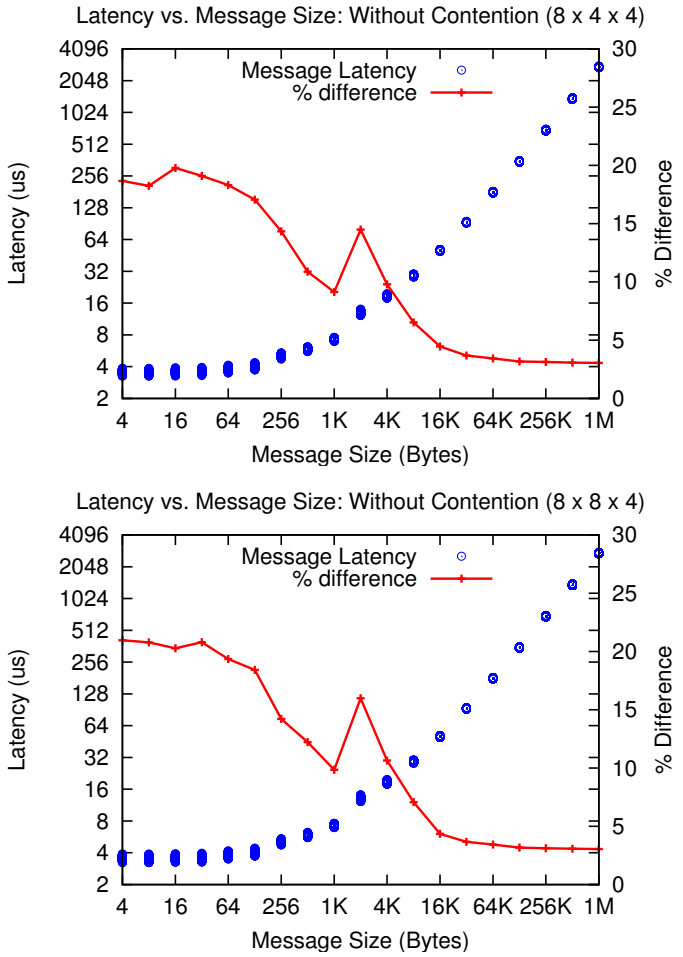


Fig. 3. Plots showing the effect of hops on message latencies in absence of contention (for $8 \times 4 \times 4$ and $8 \times 8 \times 4$ sized tori on Blue Gene/P, Benchmark: `WOCON`)

## 3. 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, 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 2). 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

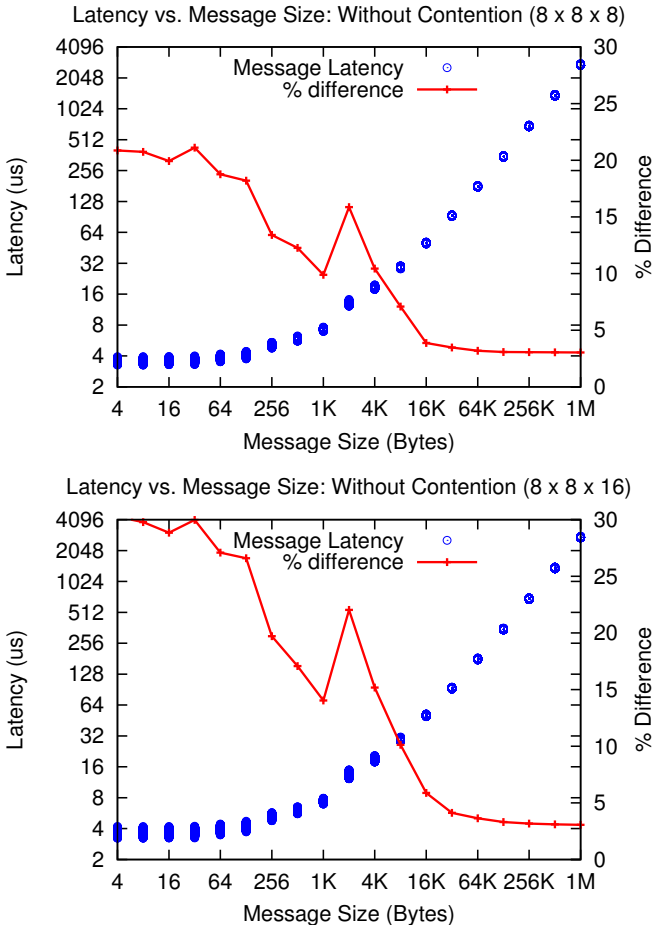


Fig. 4. Plots showing the effect of hops on message latencies in absence of contention (for $8 \times 8 \times 8$ and $8 \times 8 \times 16$ sized tori on Blue Gene/P, Benchmark: `WOCON`)

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. Figures 3 and 4 present 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.

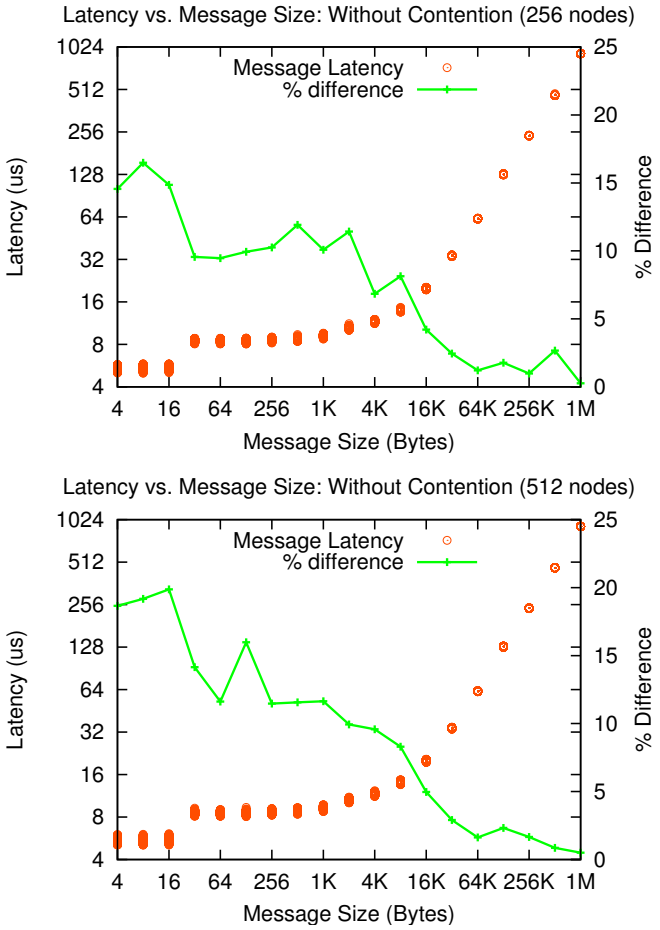


Fig. 5. Plots showing the effect of number of hops on message latencies in absence of contention (for 256 and 512 nodes of XT3, Benchmark: `WOCON`)

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 specific message size in the four plots (Figures 3, 4). 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 2 KB messages. This can be

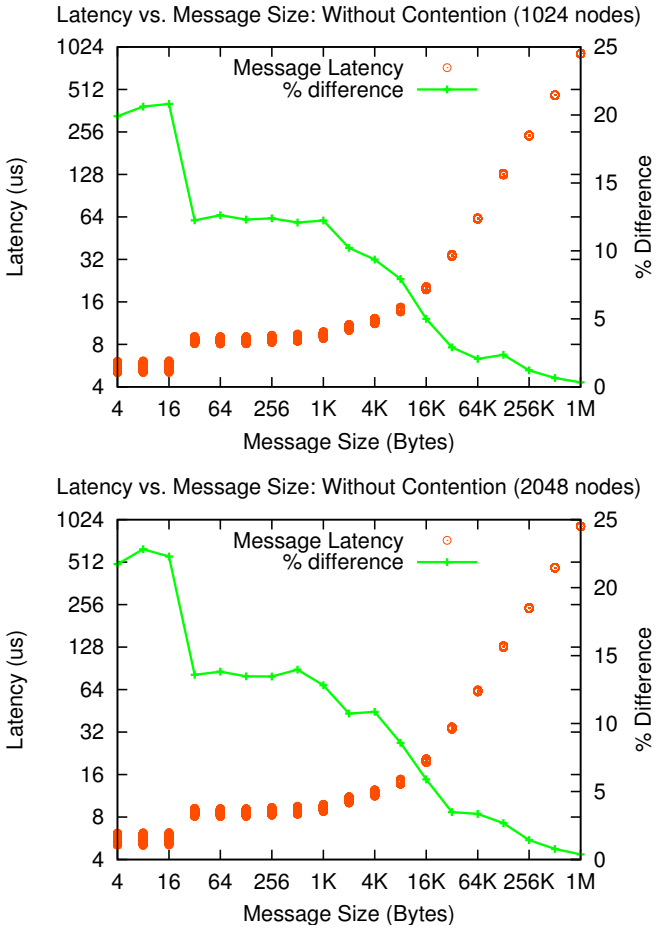


Fig. 6. Plots showing the effect of number of hops on message latencies in absence of contention (for 1024 and 2048 nodes of XT3, Benchmark: `WOCON`)

explained by the use of different routing protocols on BG/P for different message sizes [12]. 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.

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 4). 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, for a fixed message size, the difference between minimum and maximum latencies increases 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

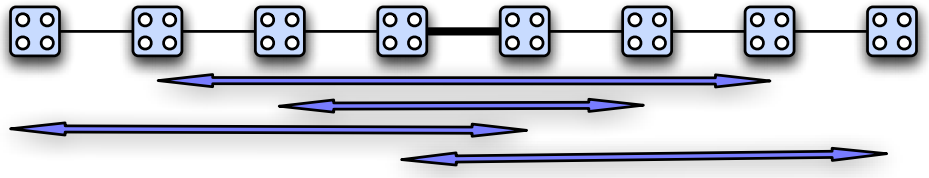

Fig. 7. Communication patterns in the `WICON` benchmark. This diagram is a simplified one-dimensional version of the pattern in three-dimension (3D). The random pairs are chosen from all over the 3D partition.

```
pe = partner[myrank];
if(myrank < pe) {
  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;
} else {
  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;
}
```

Fig. 8. Code fragments showing the core of `WICON` Benchmark

– 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.

Figures 5 and 6 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 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 6.

## 4. 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

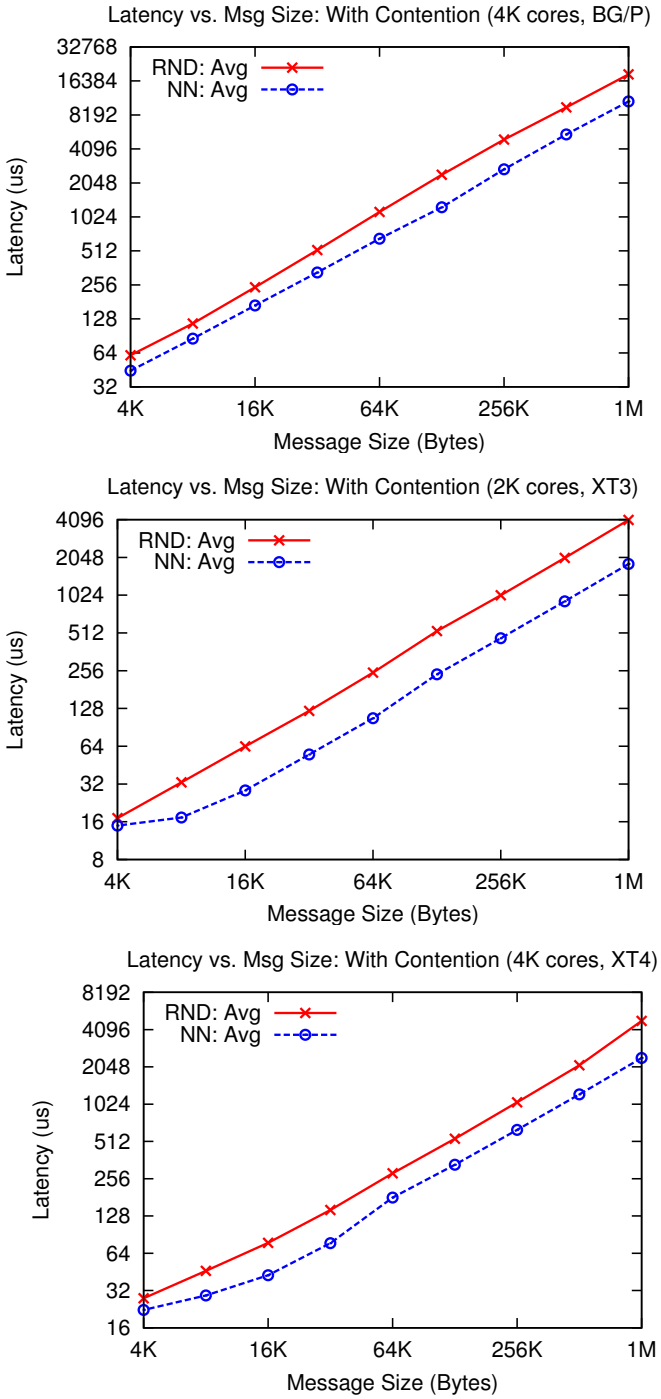


Fig. 9. Plots showing the results of `WICON` on Blue Gene/P, XT3 and XT4

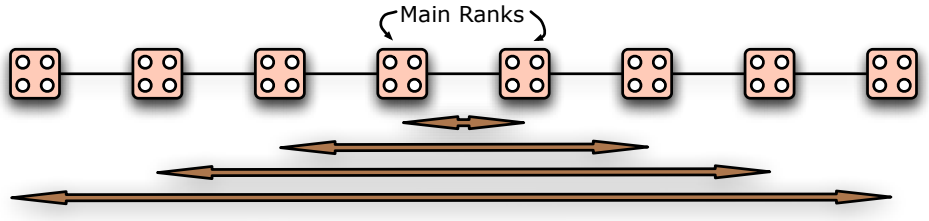


Fig. 10. For increasing stress on a given link, pairs are chosen along the $Z$ dimension. A baseline run is done with the middle pair and then other pairs are added around the middle one.

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 7). The average time for the message sends is recorded for different message sizes (pseudo-code in Figure 8). 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 9 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 and XT4 to understand the effects of contention on Cray XT machines. The second plot in Figure 9 presents the results for `WICON` benchmark on $2,048$ cores of XT3 and the third plot for $4,096$ cores of XT4. We see a significant difference between the NN and RND lines (a factor of 2.25 at 1 MB messages for XT3 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 [7].

## 5. Controlled Contention Experiments

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

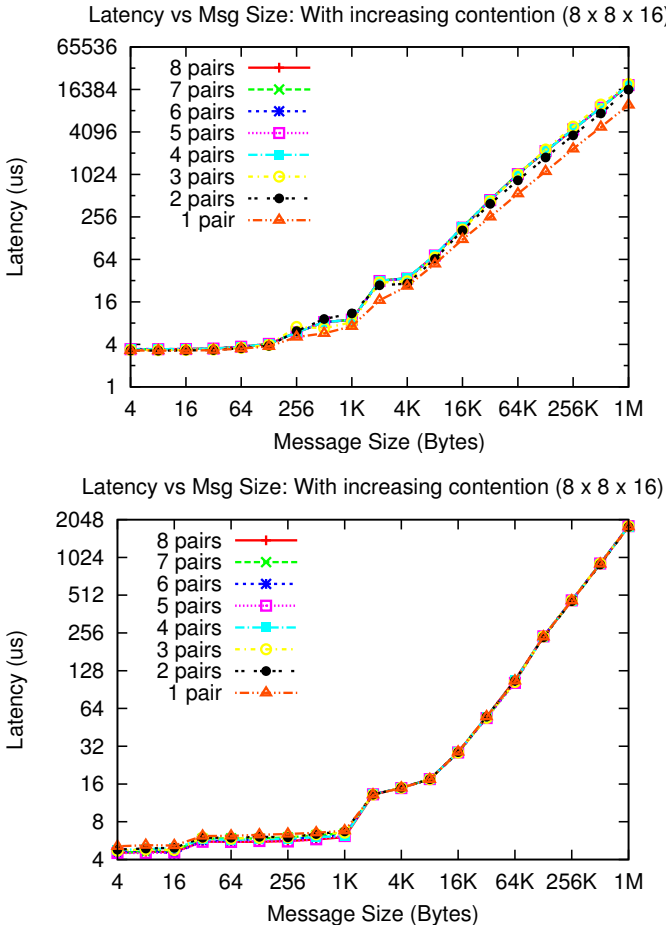


Fig. 11. Plots showing the results of *stressing-a-link* benchmark on Blue Gene/P and XT3

conduct controlled experiments. The next two subsections list the results of running these experiments where we inject congestion along one direction of the torus.

### 5.1. *Benchmark stressing a given link*

In this experiment, we try to see the effect on message latencies when a particular link is progressively used to send more and more messages. From all ranks in the $Z$ dimension with a specific $X$ and $Y$ coordinate, we choose the pair of ranks in the middle and measure the message latency between them. Then we keep adding pairs around this pair in the $Z$ dimension and measure the impact of added congestion on the link in the center. (Figure 10).

Figure 11 shows the results from running this benchmark on BG/P and XT3 for different message sizes. On BG/P (top plot), we see that as message size increases,

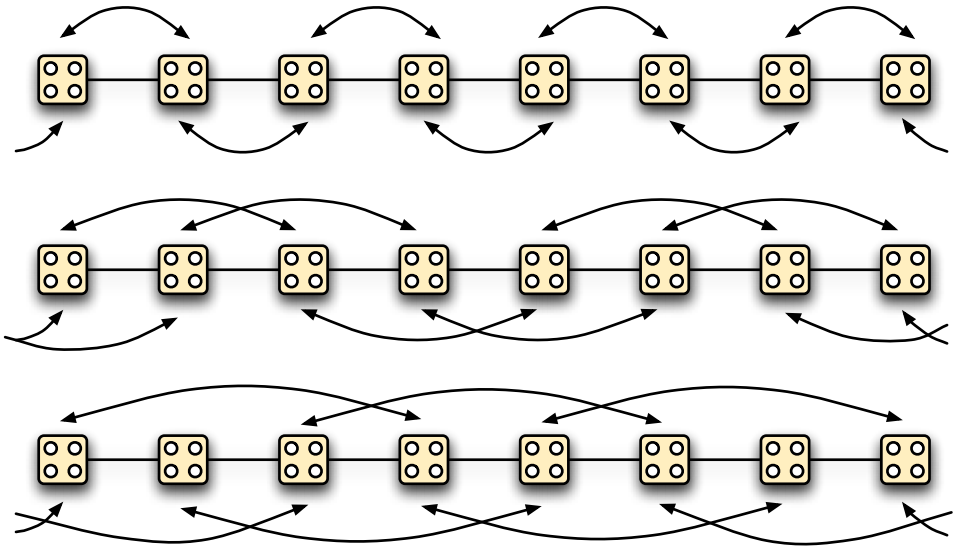

Fig. 12. For creating pairs of processors, with the same distance between the partners in each pair, strategies as shown in this diagram were used. Pairs are created along the $Z$ dimension and here we show distance=1, 2 and 3.

we see increased message latencies with more and more pairs. Additional pairs around the main ranks create contention on the middle links, hence, slowing down their messages. The difference between the message latencies for the 1 pair of nodes versus all 8 pairs of nodes sending messages is about 2 times (similar to what we observed for the `WICON` benchmark.)

The bottom plot shows the results from running the same benchmark on a 1024 node contiguous partition of XT3. Surprisingly, leaving aside the small perturbation for small messages, we see no impact of stressing a particular link with messages on XT3. This might be due to better algorithms for congestion control in the SeaStar routers or the interconnect or in the MPI implementation.

### 5.2. *Benchmark using equidistant pairs*

Similar to the `WICON` benchmark, 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 12).

Figure 13 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

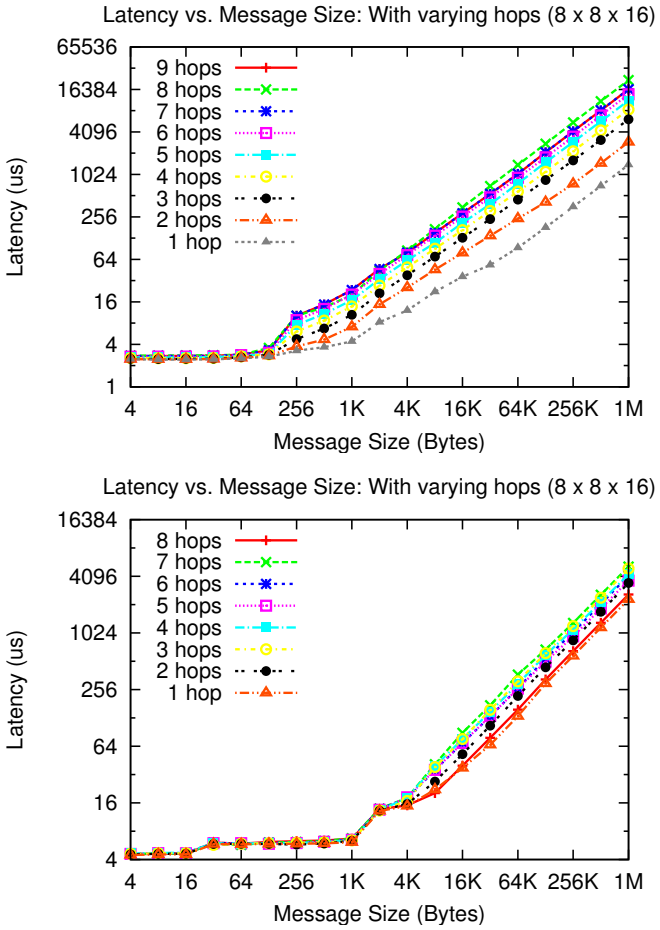


Fig. 13. Plots showing the results of the *equidistant-pairs* benchmark on Blue Gene/P and XT3

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 be contention for the Hyper Transport (HT) link which connects the nodes to the SeaStar router instead of the network links. Another reason might be higher bandwidth and better capability of XT3 to handle random contention.

Section 3.5 in [17] uses a similar benchmark and demonstrates results similar to ours.

## 6. Comparison across Machines

It is interesting how Blue Gene/P and XT3 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 unidirectional link bandwidth) and hence we would expect XT3's network to perform better than BG/P. Table 1 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* (less than 224 bytes), *eager* and *rendezvous* (greater than 1200 bytes) routing protocols are used respectively on BG/P.

Table 1. Comparison of message latencies (in $\mu s$) on Blue Gene/P and XT3 in absence of contention (`WOCON`) and presence of contention (`WICON`)

| **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 |

In the absence of contention (`WOCON`), both NN and RND messaging on BG/P is two times as fast as on XT3 for 32 byte messages. It is 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 (`WICON`), 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 worse in presence of contention. Not only does this suggest that topology mapping should not be ne-

glected on the XT machines, it shows that topology mapping would lead to higher benefits on the XT family.

However, the absolute message latencies on XT3 are still better than Blue Gene/P and this might be the reason why much research has not been done on topology-aware mapping on the Cray machines. Also, benchmarking results from Sections 5.2 and 5.1 suggest that the much higher bandwidth on Cray machines helps mitigate the effects of contention. The increase in message latencies for XT3 is only 2 times compared to nearly 8 times for BG/P for the first benchmark and for the second benchmarks, XT3 does not get affected by contention at all.

## 7. Impact on Application Performance

Three applications, one written in Charm++ and two in MPI were chosen to evaluate the impact of topology aware job scheduling on application performance. NAMD is a highly scalable production Molecular Dynamics code [6]. MILC stands for MIMD Lattice computation and is used for large scale numerical solutions to study quantum chromodynamics [21]. DNS is a turbulence code developed at Sandia National Laboratory [22].

We used a $92,222$-atom system called Apolipoprotein-A1 (ApoA1) for benchmarking NAMD. For benchmarking MILC, we use the application *ks_imp_dyn*, which is used for simulating full QCD with dynamical Kogut-Susskind fermions. We did weak scaling for MILC starting with a grid of size $16 \times 16 \times 16 \times 16$ on 256 cores. For benchmarking of DNS, we used the *purest* form of the code: Navier-Stokes with deterministic low wave number forcing and a grid size of $128^3$.

Table 2. Impact of interference from other jobs on Cray XT3 for three different applications: NAMD, MILC and DNS

| | **NAMD (ms/step)** | | | **MILC (secs/step)** | | | **DNS (secs/step)** | | |
|---|---|---|---|---|---|---|---|---|---|
| **No. of cores** | 256 | 512 | 1024 | 256 | 512 | 1024 | 256 | 512 | 1024 |
| ***Batch Queue*** | 8.82 | 5.14 | 3.08 | 0.31 | 0.368 | 0.419 | 0.126 | 0.151 | 0.139 |
| ***Special Queue*** | 8.78 | 5.10 | 3.01 | 0.31 | 0.352 | 0.372 | 0.124 | 0.148 | 0.146 |

Table 2 presents the results of the application benchmarking. The runs were done at different times. The runs labeled as *Batch Queue* were submitted to the default queue of Bigben and the results were collected. The runs labeled *Special Queue* were done in a reservation where we get 3D contiguous shapes for the runs and there is no interference from other jobs. The performance of NAMD is similar for the two modes which is expected because of the latency tolerant nature of NAMD. Topology starts affecting NAMD's performance for very fine grainsizes at scaling limits with very large number of processors. MILC on the other hand observes an improvement of nearly 5% on 512 cores and 11% on 1024 cores. MILC has a near-neighbor communication pattern and it maps nicely if the job partition is a 3D mesh

or torus. Hence, even without using the topology information for explicit mapping of ranks to physical processors, MILC has a performance improvement. This can be attributed to the minimal interference from other jobs and a implicit topology aware layout of its ranks to processors.

DNS, like NAMD, does not show any observable improvement from the *Batch* to *Special* runs. DNS is a turbulence code which does multiple Fast Fourier Transforms (FFTs) resulting in a large number of small message sends and `MPI_Waitall`s. This application might require a mapping algorithm for its FFTs onto the physical processors for performance improvements. In conclusion, a topology aware scheduler is beneficial for certain kinds of applications but not for others. For communication bound applications, we see a performance improvement due to two reasons: 1. There is low interference from other jobs and all ranks for a given job are closely packed leaving no outliers far away on the machine; 2. Once we have a 3D job partition, the application can do an intelligent topology aware mapping depending on its communication patterns. This makes a case for Cray XT machines to have topology aware job schedulers too, like IBM Blue Gene machines.

## 8. 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.) Other studies have shown that contention can impact message latencies on Infiniband networks [23]. It remains to be studied whether topology aware mapping will impact performance in dynamically routed fat-tree networks. 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 and DOE grant DE-FG05-08OR23332 through ORNL LCF. The authors would like to thank Chad Vizino from the Pittsburgh Supercomputing Center for setting up a reservation on Bigben (under TeraGrid [24] allocation grant ASC050040N supported by NSF) and allocating contiguous partitions for the runs, which is not possible using the default job scheduler. We used running time on the Blue Gene/P at Argonne National Laboratory, which is supported by DOE under contract DE-AC02-06CH11357. We also used running time on Jaguar which is a resource of the National Center for Computational Sciences at Oak Ridge National Laboratory, and is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725 .

**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] Dilip D. Kandlur and Kang G. Shin. Traffic Routing for Multicomputer Networks with Virtual Cut-Through Capability. *IEEE Trans. Comput.*, 41(10):1257–1270, 1992.

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

[6] 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*, April 2008.

[7] Eric Bohm, Abhinav Bhatele, Laxmikant V. Kale, Mark E. Tuckerman, Sameer Kumar, John A. Gunnels, and Glenn J. Martyna. 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] Abhinav Bhatelé, Laxmikant V. Kalé, and Sameer Kumar. Dynamic Topology Aware Load Balancing Algorithms for Molecular Dynamics Applications. In *23rd ACM International Conference on Supercomputing*, 2009.

[15] 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.

[16] Abhinav Bhatelé, Eric Bohm, and Laxmikant V. Kalé. A Case Study of Communication Optimizations on 3D Mesh Interconnects. In *Euro-Par 2009, LNCS 5704*, pages 1015–1028, 2009.

[17] 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.

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

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

[21] Claude Bernard, Tom Burch, Thomas A. DeGrand, Carleton DeTar, Steven Gottlieb, Urs M. Heller, James E. Hetrick, Kostas Orginos, Bob Sugar, and Doug Toussaint. Scaling tests of the improved Kogut-Susskind quark action. *Physical Review D*, (61), 2000.

[22] Mark A. Taylor, Susan Kurien, and Gregory L. Eyink. Recovering isotropic statistics in turbulence simulations: The Kolmogorov 4/5th law. *Physical Review E*, (68), 2003.

[23] T. Hoefler, T. Schneider, and A. Lumsdaine. Multistage switches are not crossbars: Effects of static routing in high-performance networks. In *Cluster Computing, 2008 IEEE International Conference on*, pages 116–125, October 2008.

[24] 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.