RC24327 (W0708-028) August 8, 2007
Computer Science

IBM Research Report

Performance Analysis Of All-to-All Communication on the Blue Gene/L Supercomputer

Sameer Kumar, Philip Heidelberger
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Performance Analysis and Optimization of All-to-all Communication on the Blue Gene/L Supercomputer

Sameer Kumar and Philip Heidelberger
[1]IBM T.J. Watson Research Center,
Yorktown Heights, NY 10598, USA
{sameerk,philiph}@us.ibm.com

Abstract

All-to-all communication is a well known performance bottleneck for many applications. For such applications to scale to a large number of processors, optimizing all-to-all communication is critical. In this paper, we analyze the performance of all-to-all communication on the Blue Gene/L torus interconnection network, which has limited bisection bandwidth. The torus interconnect topology has link contention even for all-to-all communication operations with short messages. We observed that the performance of all-to-all communication also depends on the shape of the processor partition. We present a performance analysis of all-to-all communication on mesh and torus partitions of various shapes and sizes. We then present optimization schemes to enhance the performance of all-to-all communication. The large message optimization substantially improves all-to-all performance on an asymmetric torus. In particular, performance improved from about 70% to over 99% of peak on a 20,480 ($40 \times 32 \times 16$) node configuration, which was the largest machine to which we had access. The short message optimization can double all-to-all performance for very short messages.

1 Introduction

Since collective communication operations, such as all-to-all communication can consume a significant share of the application run-time on large machines, optimizing such operations is necessary to achieve good scaling. In this paper we study all-to-all personalized exchange, where every processor sends *distinct* data to *all* other processors. There are different types of communication architectures currently available to run scientific applications, using different network topologies such as Fat-trees, Meshes, Tori and Hypercubes. There are also several communication optimization algorithms published in literature being used for the different communication architectures. Optimization of all-to-all communication on fat-tree networks is presented in [7, 10], while on meshes and tori have been presented in [9, 11]. In this paper we present a performance analysis and optimization strategies for all-to-all (or AA for brevity) communication on Blue Gene/L.

The Blue Gene/L machine [4] is a low-power massively parallel supercomputer with a 65536 processor installation at the Lawrence Livermore National Laboratories. The Blue Gene/L machine has a three dimensional torus interconnect for application data communication. The Livermore machine is configured as a $64 \times 32 \times 32$ torus. When the number of nodes in each dimension is the same we call the torus **symmetric**, otherwise we call it **asymmetric**. On a symmetric three dimensional torus with P nodes, the bisection bandwidth is $O(P^{2/3})$, hence it is critical that all-to-all traffic can easily achieve peak performance on the available bisection bandwidth. Studies in [1] and [2] show that all-to-all achieves excellent performance on Blue Gene/L machines having a symmetric torus; typically over 95% of peak is achieved for long messages. However, performance can suffer on an asymmetric torus due to network contention; typically 70 to the low 80's percent of peak is achieved for long messages.

In this paper, we demonstrate that all-to-all performance can be substantially improved on Blue Gene/L by presenting and measuring optimization schemes for both *asymmetric tori* and *short messages*. In addition, we present performance models and analysis of all-to-all communication on symmetric and asymmetric Blue Gene/L torus partitions. We hope the performance analysis and the optimization techniques presented in this paper can be also applied for more complex many-to-many communication patterns and benefit many applications on Blue Gene/L.

We explore both direct and indirect strategies to optimize all-to-all communication. Direct strategies [9, 11] try to minimize network contention and are typically more suitable for large messages. We explore both deterministic and adaptive routing algorithms to optimize direct all-to-all communication. Indirect strategies route packets or messages through intermediate nodes. Message combining based indirect strategies [9, 3, 6] minimize startup overheads for all-to-all communication and are suitable for short messages. This technique was effectively used to optimize the HPCC Randomaccess benchmark [5] on Blue Gene/L which has a similar communication pattern as all-to-all communication. The Randomaccess optimization however had two phases of forwarding and hence had a significant CPU overhead. Our indirect strategies use only one phase of forwarding, hence allowing the processor core to keep the necessary links busy in the all-to-all communication operation. Traditionally, indirect strategies have been used to optimize short messages. However, we present two indirect strategies to optimize network throughput for large messages and startup overheads for short messages respectively. Our large message algorithm has achieved over 95% of peak on all asymmetric partitions that we have tested. Thus a strategy that uses a direct algorithm on a symmetric torus and an indirect algorithm on an asymmetric torus delivers over 95% of peak. Our short message optimization, based on algorithms in [3, 6], can more than double all-to-all performance for short 8-byte all-to-all operations.

The next section provides a summary of the Blue Gene/L architecture and a performance model for message passing on the Blue Gene/L torus interconnect. Section 3 presents and models direct strategies for all-to-all communication. Indirect strategies are presented in Section 4, and we conclude in Section 5.

2 The Blue Gene Machine

The Blue Gene/L (BG/L) machine has been designed with a full system-on-chip model with two processor cores and all network interfaces and routing integrated onto the same chip. BG/L uses a torus interconnect for the majority of application messaging. A detailed description of the BG/L torus network may be found in [1]. Since BG/L does not have a DMA engine, the cores are responsible for managing communications, specifically for injecting packets into network injection FIFOs and removing packets from network receptions FIFOs at the destination node. BG/L uses virtual cut through routing, so the packets at intermediate nodes do not require handling by the cores. Each node on the BG/L torus is connected with six neighbors. When the data is in L1-cache, low level communication APIs can achieve about *five* links throughput for near neighbor communication. When the data is not in L1 the processor can only keep about *four* links busy. The lower throughput is due to the relatively slow CPU (compared to the network) and the relatively high packet processing overheads. Even on a small BG/L, a 512 node (8x8x8) midplane, the data for all-to-all will not fit in the L1-cache for message sizes larger than 64 bytes, so in the rest of paper we assume that each processor core can keep about four links busy. Note that on a midplane, the average packet in an AA takes 2 (= 8/4) hops in each of the three dimensions, so on average 1/6 of the packets arriving at a node will be destined for that node. As there are six links/node, the processors need to sustain only one link of throughput. In general, the processing demand is proportional to one over the average number of hops, which decreases as the machine size increases.

The torus router has four virtual channels, two dynamic, one “bubble normal” (for deterministic routing and deadlock prevention) and one for high priority messages. Application messages typically do not use the high-priority virtual channel. The torus router supports both deterministic and adaptive routing. Adaptive routing is enabled by default on the two dynamic VCs. For deterministic routing the packets have to be routed on the bubble normal VC. Deterministic routing, and selection of the “escape” VC for deadlock prevention are done in dimension order (first X, then Y, then Z).

2.1 Communication Model

In this paper we use the following model, for the time to send a point-to-point message, T_{ptp} is given by

$$T_{ptp} = \alpha + (m + h) * C * \beta + L \tag{1}$$

- α is the total processor and network startup overhead for sending each message that cannot be pipelined.
- β is the per byte network transfer time. The byte here is being sent out from main memory.
- m is the size of the message.
- C is the network contention delay experienced by the message, which actually depends on the message size.
- h is the size of the software header in the message.
- L is the network latency. The network latency depends on the number of hops of the packet traverses. In all-to-all communication this latency term is not very critical, as each node sends many packets that are pipelined on the network.

Analyzing Network Contention: As the BG/L torus does experience network contention, we now present an analytical technique to estimate the contention parameter for AA. On the BG/L torus with P processors, where $P = P_x \times P_y \times P_z$ and $M = max(P_x, P_y, P_z)$, let each processor send n packets to every other processor with $payload$ bytes of application data in each packet. So the network would have to move P^2n packets in the all-to-all operation. Now each of these packets would travel $M/4$ links on an average, along the maximum sized dimension. The total number of links in the maximum dimension is $2 * P$ as the interconnect has a torus topology. Hence the network time to do AA is given by :

$$T = (P^2 * M/4 * n * payload * \beta)/(2 * P) = P * (M/8) * m * \beta \tag{2}$$

From Equation 2 we can conclude that the contention parameter is $C = M/8$.

Measuring model parameters: The model parameters are measured from ping-pong benchmark and measuring all-to-all performance on with small messages on smaller processor partitions.

3 All-to-all Communication

All-to-all communication algorithms can be classified as direct or indirect[8]. With direct algorithms each processor sends its data directly to the destination processors, typically through point-to-point messages. Direct algorithms aim at exploiting specific communication architectures and emphasize sequencing of messages to avoid link and node contention. In the indirect approach [3, 6], processors combine messages into larger data blocks which are sent to intermediate processors to be routed to the final destination. Indirect approaches use message combining to reduce the total number of messages sent, and are hence most useful for small messages; while direct approaches minimize network contention overheads making them suitable for long messages.

In this section, we analyze the performance of direct strategies on the BG/L torus. The cost of implementing direct all-to-all communication, where each processor sends point-to-point messages to the (P-1) destinations is given by Equation 3 using the model presented in Section 2.1. The derivation is intuitive, the cost for AA is the non-pipelinable startup overhead and the time to get all the data on the network. The messages may then get stuck in network buffers resulting in an additional contention delay given by the $(M/8)$ term. Here M is the size of the longest dimension of the torus network.

$$T_{simple\_direct} \approx P * \alpha + P * (M/8) * (m + h) * \beta \tag{3}$$

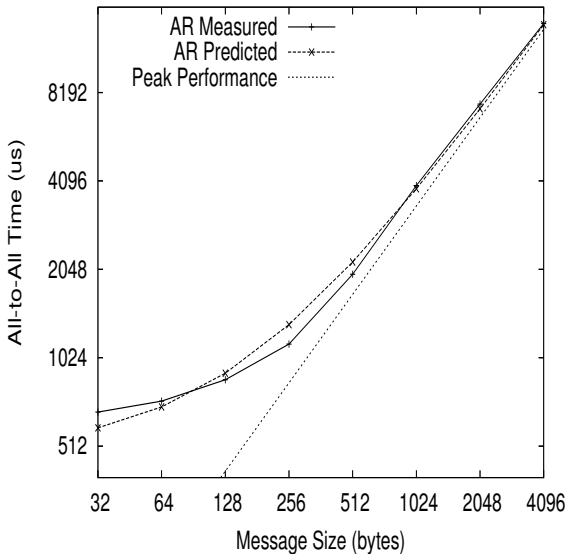

Figure 1. AR strategy performance and prediction on an 8x8x8 mid-plane of Blue Gene/L

MPI All-to-all: On BG/L, the MPI all-to-all [2] library call is implemented with a single message per node to amortize the startup costs of the all-to-all across all destinations. Implementing AA with point-to-point communications incurs a penalty for the allocation and deallocation of each message object. Additional overhead is incurred from protocol overhead and message startup times for each message. This production MPI all-to-all also injects packets in a random permutation to smoothen the areas of link contention on the network. There is a tuning parameter which injects multiple packets per destination before moving to the next destination; this parameter balances cache and memory system performance against network performance. Depending on partition shape and size, the optimal tuning parameter is usually 1 or 2 full-sized 256-byte packets per destination (such a packet generally contains 240 bytes of payload data).

AR Strategy: In this paper, all the presented results use a native communication library (below MPI) developed by the authors. In this framework we redesigned the above randomized packet all-to-all scheme with lower overheads. We refer to this optimization strategy as *AR* in the rest of the paper. This scheme also avoids overheads of the MPI layer. It also use a different randomization scheme than MPI. This low level optimization library allowed us to experiment with new direct schemes for all-to-all communication which are presented in Section 3.2. On a 512 node ($8 \times 8 \times 8$) midplane of Blue Gene/L the AR scheme has a throughput of 99% of peak compared with 97% for MPI for a 4KB message.

We have measured the startup overhead (α) of the AR scheme to be about 450 processor cycles or $640\mu s$ for each destination. We also measured the per-byte term (β) to be about 6.48ns/byte. The software header is about 48 bytes and it sent only in the first packet. The software header makes the shortest all-to-all to packet to be 64 bytes. On the Blue Gene/L torus network, packet sizes that are multiple of 32 bytes (up to 256 bytes) can be sent on the network. We are in the process of optimizing our runtime to reduce overheads allowing it to send 32 byte packets.

3.1 Performance on Symmetric Torus

All the results reported in this paper were run on the BG/L system at the IBM T.J. Watson Research Center; the largest partition on this machine is $40 \times 32 \times 16$. Figure 1 shows the performance of the adaptive routing strategy (AR) on an $8 \times 8 \times 8$ mid-plane (512 nodes) of Blue Gene/L. Figure 2 shows the performance of the AR all-to-all on 4096 nodes of Blue Gene/L. The figures also show the performance predictions from the model presented in Section 2.1 and the achievable peak performance with no startup overheads.

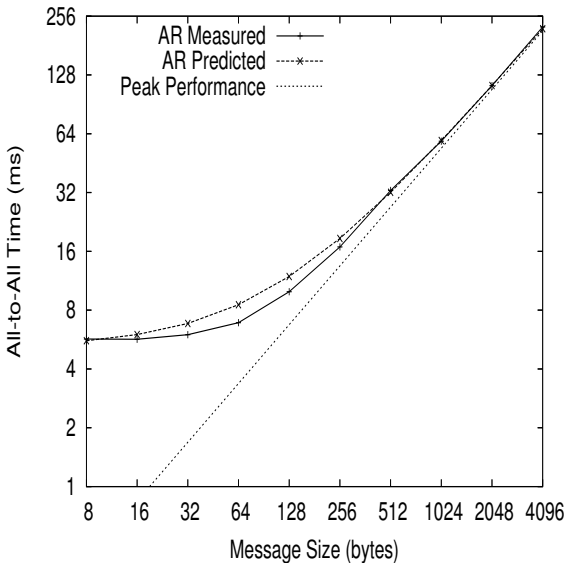

Figure 2. AR strategy performance and prediction on an 16x16x16 partition of Blue Gene/L

Partition Size	AR % of Peak achieved
8	98.2
16	97.7
8 x 8	98.7
16 x 16	99.7
8 x 8 x 8	99.0
16 x 16 x 16	99.0

Table 1. All-to-all peak performance of various symmetric partitions for large messages

Table 1 presents the peak performance of all-to-all communication symmetric lines, planes and 3D torus partitions. The percent of peak is computed using equation 2 which does not account for the start up overheads. Observe that on these symmetric torus partitions, where Xsize=Ysize=Zsize for a 3D torus, where Xsize==Ysize for a 2D torus, and for a 1D torus line, the direct AR strategy achieves very close to peak performance. This is because randomization and adaptive routing effectively prevent links and routers on the network from becoming hot-spots. For large messages, over 99% of peak is acheived on a 3D torus, meaning that the *average* link utilization during the all-to-all is over 99%. Performance increases rapidly: on 512 nodes over 90% of peak is achieved for messages as small as 1 packet or a 240 byte payload. (This is not clearly evident from Figure 1, as the message sizes in the all-to-all are not multiples of 240.)

3.2 Performance on Asymmetric Meshes and Tori

Unlike meshes, both asymmetric and symmetric tori should have no hot-spots due to the torus links. Table 2 shows the percent of peak achieved for large messages using the AR all-to-all strategy on various partition sizes of BG/L. The percent of peak is computed on the message size which can fit in memory, or when the percent of performance asymptotically reaches the maximum possible value. Figure 3 shows plots the peak bisection bandwidth per node and the bandwidth achieved by the AR strategy for one packet and for large messages. Notice that a one packet all-to-all achieves performance quite close to the achievable throughput. For symmetric partitions (512 nodes and 4096 nodes) AR achieves very close to peak throughput.

As shown in [1] and [2], we also observe that performance of all-to-all degrades on an asymmetric torus, even with

Partition Size	AR % of Peak achieved
8 x 2M	91.8
8 x 4M	89.0
8 x 16	85.7
8 x 32	84.0
8 x 8 x 2M	90.1
8 x 8 x 4M	87.7
8 x 8 x 16	81.0
8 x 16 x 16	87.0
8 x 32 x 16	73.3
16 x 32 x 16	71.0
32 x 32 x 16	73.6

Table 2. AA performance using the AR strategy for large message sizes on various processor partitions. An "M" indicates that the partition is a mesh rather than a torus in that dimension.

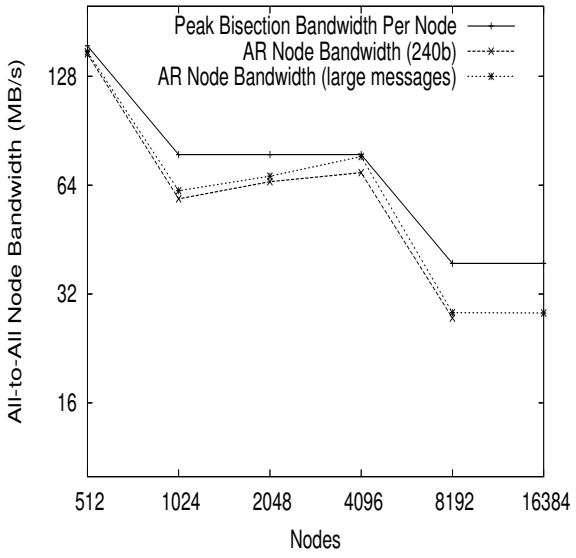

Figure 3. AR strategy per node throughput on Blue Gene/L

the randomized AR all-to-all strategy. This is due to contention inside the network caused by the asymmetry which induces unequal utilization on the links. Note that in a $2n \times n \times n$ torus, the average number of hops in the X dimension is twice that of the Y and Z dimensions, so in an all-to-all, the X links have twice the utilization of the Y and Z links.

The torus router on BG/L is input-queued [1] with support for both deterministic and adaptive routing on multiple virtual channels. With deterministically routed packets there is only one direction and virtual channel available to a packet. Adaptive routing uses a JSQ (join the shortest queue) algorithm to find a destination direction and virtual channel. Directions which have more downstream tokens and link availability are preferred Hence, adaptive routing tries to ensure that all directions are equally loaded for uniform traffic.

In all-to-all, on an asymmetric torus, network buffers can become full and degrade throughput. Suppose the longest dimension is X, so the X links are most heavily utilized. With dynamic routing, packets can enter the network on the less used Y and Z links. As the X links are busy, packets tend to move along Y and Z until no more hops are available. Eventually packets at the head of a Y or Z VC must make an X hop, however as there tend to be many packets waiting for the X link, a queue develops behind such a packet, eventually filling the VC and preventing packets on other nodes from using this Y or Z link into the node. In this way, contention builds up inside the network. Even with this phenomenon, BG/L still delivers over 70% of peak on all-to-all.

To minimize network contention we explore two new direct strategies strategies for all-to-all communication on BG/L. The are described below :

- Throttling Packets: As adaptive routing leads to congestion, we explore a strategy to limit the number of packets in the network by injecting packets at a rate driven by the bisection bandwidth (equation 2). This may prevent switches from getting overloaded and congested.

- Deterministic Routing: Here the packets are sent in a random order on the deterministic bubble virtual channel. The dimension-ordered routing prevents packets packets on a Y VC to wait for an X link as it happens in dynamic routing, thus making network congestion less likely in some cases. We expect this strategy to work best if X is the longest dimension, since then packets already in the network that are changing dimension tend to wait for less utilized links. Notice that if the longest dimension is, say Y, and a packet at the head of a Y VC is turning onto a Z link, then there is a likelihood that the outgoing Y link from this node will be idle since with high probability packets waiting in an injection FIFO cannot enter the network on this Y link (since the deterministically routed packet must enter on an X link if it has X hops to make). Thus if the longest, bottleneck, dimension is not X, we expect a dropoff in performance compared to a similar sized machine whose longest dimension is X.

The percent of peak throughput achieved by the different strategies in shown in Figure 4. Note that on *symmetric* tori, deterministic routing is not as effective as adaptive routing because of head-of-line blocking and the fact that most packets must be injected onto an X link. Deterministic routing achieves greater than 90% of peak on $2n \times n \times n$ partitions. This because the X links are the bottleneck and deterministic routing starts packets off in the X dimension. Note in particular that, as mentioned above, the performance of DR on a $16 \times 8 \times 8$ partition is better than on an $8 \times 16 \times 8$ or an $8 \times 8 \times 16$ partition. For other shaped partitions, sometimes DR is better than AR (e.g, on $8 \times 32 \times 16$ DR achieves 86% of peak compared to AR's 77% of peak), but sometimes it is worse (e.g., on $8 \times 16 \times 16$ DR achieves only 67% of peak compared to AR's 86% of peak).

The throttling enhancement to adaptive routing only helps by about 2-3% on 1024 nodes, suggesting that it does not prevent routers from being congested. From Figure 4, we can conclude that the performance direct strategies is not optimal for all processor partitions. This has motivated us to study indirect strategies presented in the next Section.

4 Indirect Strategies

In the last section we presented direct optimization schemes for all-to-all communication. The performance of these schemes range between 71-99% of peak depending on the shape of the torus partition. We now explore indirect strategies, which route all-to-all data into intermediate nodes. The intermediate node may just store and forward packets or, for short messages, coalesce data from different sources into point-to-point messages for each destination.

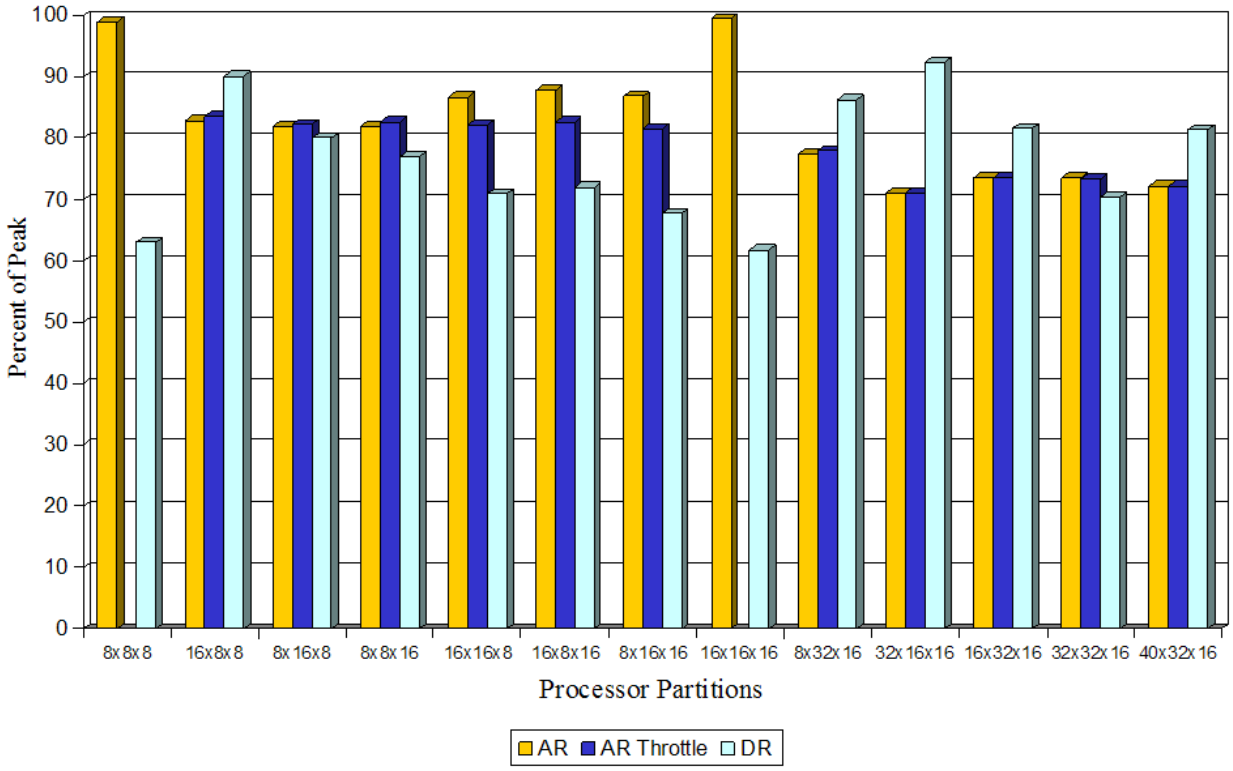

Figure 4. Performance for large messages and different direct strategies. AR = adaptive routing, DR = deterministic routing

The important price paid with indirect strategies is the space required for an all-to-all is doubled for storing and processing packets to other destinations. We explore both the scenarios of packet forwarding and coalescing.

4.1 Two Phase Schedule Strategy

In practice, we observe that many important partitions of BG/L are of the type $2n \times n \times n$ or $2n \times 2n \times n$. One can think of such partitions as a set of symmetric planes along a torus line in a third dimension. As the direct strategies hit near-peak performance for both torus lines and symmetric planes, we therefore designed the following Two Phase Schedule (TPS) all-to-all optimization strategy to take advantage of the high performance on lines and symmetric planes. This strategy essentially pipelines one all-to-all along a line with a second all-to-all along a symmetric (or nearly symmetric) plane.

- Phase 1: Pick a dimension to be the "linear" dimension. Each node sends packets along this linear dimension dimension to intermediate nodes. The intermediate node for a given final destination is chosen to have the same coordinate in the linear dimension as the final destination. For example, on a $2n \times n \times n$ torus, the (best) linear dimension is X and packets will be sent along the X dimension. Each source $(x1, y1, z1)$ will send packets to intermediate node $(x2, y1, z1)$ for all destinations $(x2, y2, z2)$.
- Phase 2: Packets are forwarded from the intermediate nodes along the other two "planar" dimensions. In the $2n \times n \times n$ example, intermediate node $(x2, y1, z1)$ sends its messages to final destinations $(x2, y2, z2)$ in this phase and thus only uses the Y and Z dimensions. This is done in a pipelined fashion allowing Phase 1 and

Partition Size	Partition Shape	Two Phase Percent of Peak	Phase 1 Dimension
512	8 x 8 x 8	77.2	Z
1024	16 x 8 x 8	99.0	X
1024	8 x 16 x 8	98.9	Y
1024	8 x 8 x 16	97.9	Z
2048	16 x 16 x 8	97.5	Z
2048	16 x 8 x 16	97.4	Y
2048	8 x 16 x 16	97.2	X
4096	8 x 32 x 16	99.5	Y
4096	16 x 16 x 16	96.1	X
8192	16 x 32 x 16	99.8	Y
8192	32 x 16 x 16	99.8	X
16384	32 x 32 x 16	96.8	Z
20480	40 x 32 x 16	99.5	X

Table 3. All-to-all performance use the Two Phase Schedule (TPS) algorithm for long messages

Phase 2 to overlap. The pipelining is possible as some of the injection FIFOs are *reserved* for each of the phases, i.e., Phase 1 packets are never *blocked* behind a Phase 2 packet in a network injection FIFO, and vice versa.

While we are free to chose the linear dimension to be any of the three dimensions, ideally we want to select it so that the remaining planar dimensions are symmetric, if possible. Otherwise, we select it to be the longest dimension. For example, in a $2n \times 2n \times n$ torus, the linear dimension would be Z. If the planar dimensions are symmetric, we expect near-peak performance on both the linear phase and the planar phase. If the planar dimensions are not symmetric, then choosing the linear dimension to be the longest dimension optimizes performance for that dimension, which is the bottleneck. Although the resulting planar phase may not operate at peak, it need not achieve peak in order for the overall algorithm to operate at peak. Based on the performance models shown earlier, if the longest dimension has size n and the second longest dimension has size m, then we expect near-peak performance if the planar phase operates at at least $(m/n) \times 100\%$ of peak.

A similar scheme can also be designed over a 3D torus with two phases of forwarding, where are packets are first routed along X links and then turned around in software along the Y dimension and then routed in software along the Z dimensions; this approach is similar to the HPCC Randomaccess strategy described in [5]. We believe the Two Phase scheme gains from lower overheads as it has only one forwarding phase. The performance results show that it can achieve near-peak performance on various asymmetric partitions.

Distinction from deterministic routing: One may wonder why the above indirect scheme is different from deterministic routing at least on some partition sizes. The key difference here is that we use adaptive routing on the Dynamic VC for both phases. This allows 3 network virtual channels (2 dynamic VCs and 1 bubble VC) to be dedicated to the all-to-all operation. With deterministic routing only the bubble VC would be available for the all-to-all operation which may result in head of line blocking and congestion. In addition, in the Two Phase algorithm, packets sent along the linear dimension never compete for links and VC FIFO space with packets sent along planar dimensions. For example, a Y-Z planar packet is never queued behind an X linear packet in a VC FIFO, whereas with deterministic routing a packet with only Y and Z hops remaining may be queued (in an X VC FIFO) behind a packet with X hops remaining.

4.1.1 Performance

The performance of the Two Phase schedule strategy is presented in Table 3. The peak performance is computed from equation 2. The percent of peak throughput is the achieved all-to-all time divided by the peak predicted time.

Partition Size	Partition Shape	Two Phase Latency (ms)	AR Latency (ms)
512	8 x 8 x 8	0.81	0.52
1024	8 x 8 x 16	1.64	1.25
4096	16 x 16 x 16	7.5	4.7
4096	8 x 32 x 16	8.1	12.4
16384	32 x 32 x 16	35.9	65.2

Table 4. AA 1 byte latency for different schemes

The linear dimension (Phase 1) chosen is the dimension of asymmetry or the longest dimension. In almost all cases the indirect strategy achieves a throughput in the high 90's percent of peak, as it can effectively minimize network congestion. Even when the planar phase is not symmetric, near-peak performance is achieved. For example, in the 8x32x16 case, packets are fist sent along the Y axis and then on an asymmetric X-Z plane. However, as mentioned above, the bisection bandwidth/node on the plane is twice that of the 32 way Y line, so even though the plane may not achieve peak performance it has sufficient bandwidth for the global all-to-all to reach 99% of peak performance.

For the 512 node case, the performance is lower as the processor core is unable to keep both the all-to-all packets along the Z dimension and the software forwarding on the XY plane going on at the same time with full throughput. We are working on optimizing the runtime to optimize the 512 node case. But on 512 nodes, the direct strategies already achieve full throughput.

Table 4 shows the latency for a 1byte message between the AR and indirect strategies. For small processor partitions the Two Phase schedule algorithm has higher latency as packets are passed into an intermediate node, which increases latency by a few hundred processor clocks. However, as the smallest packet sent in our messaging runtime is 64 bytes we have observed contention on larger configurations. Observe that with asymmetric partitions larger than 4096 nodes the indirect strategy is faster than the direct adaptive routing scheme. This shows that there is significant network contention even with 64 byte packets on the Blue Gene/L torus.

4.2 Optimizing Short All-to-all communication

So far we have considered all-to-all communication performance on relatively large messages. We observed that performance reaches close to peak performance for short messages a few packets in size. However, as the shortest packet in the AR protocol is at least 64 bytes, we believe that we can further optimize packets in the 1-64 byte range.

We use a 2D Virtual mesh based message combining scheme presented in [3, 6]. Here a *2D virtual mesh* or size $P = P_{vx} \times P_{vy}$ is mapped to a 3D torus processor partition. A similar 3D Mesh (on a 3D torus) optimization for the GUPS benchmark is presented in [5]. In our 2D virtual mesh scheme data exchange happens in two phases :

- Phase 1: In the first phase each processor exchanges data with its neighbors in a row of the virtual mesh. Each processor combines the data to the P_{vy} destinations in column j and sends it to its j^{th} row neighbor in the virtual mesh. Each processor sends P_{vx} messages $P_{vy} * m$ bytes in size.

- Phase 2: Here each processor exchanges data with its column neighbors in the *2D virtual mesh*. The messages from the row neighbors in phase 1 are sorted according to the column destinations and then forwarded to that column neighbor. Here, each processor sends P_{vy} messages $P_{vx} * m$ bytes in size.

This scheme is similar to the Two Phase schedule scheme presented in the previous section when the 2D virtual mesh is aligned with the coordinates of the 3D torus, ie when a row or column of the virtual mesh is a plane in the 3D torus. We have observed that for the best performance the sizes of rows and columns should be similar. So we choose a decomposition which keeps number of rows and columns to be about the same.

In both phases, each processor sends $P \times m$ bytes of data. Equation 4 gives the predicted performance for the 2D virtual mesh strategy. There are two additional terms γ and *proto*. The term γ refers to memory copying overhead

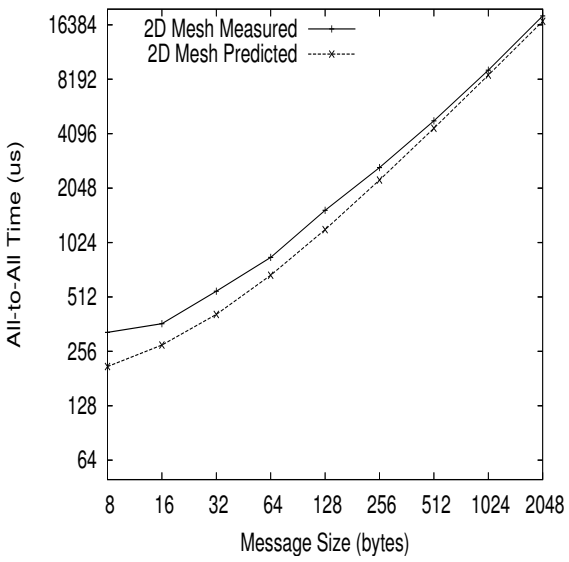

Figure 5. VMesh AA performance prediction on a 512 nodes

to move data from row messages to column messages on the intermediate node. We estimate the copying overhead to be about 1.1byte/cycle or 1.6ns/byte for short copies. The term *proto* is the protocol header carries the size of the message and the source and is about 8 bytes in size. It is different and smaller the software header 'h' presented before for direct strategies and the Two Phase scheduling scheme. As phases do not overlap, each message is injected twice into the network and this optimization may not be effective for larger messages. The other difference with other strategies presented before is that we use message passing as opposed to a packet based communication here resulting is a slightly higher α overhead of about 1170 processor cycles or $1.7\mu s$.

$$T_{bgl-mesh} \approx (P_{vx} + P_{vy}) * \alpha + 2 * P * (m + proto) * ((M/8) * \beta + \gamma) \quad (4)$$

When we compare equation 4 to equation 3, we find that for large processor partitions (with a large M) the β terms will dominate for even very short messages. Comparing the two β terms notice that the change-over point between AR and vmesh scheme would be when $m = h - 2 * proto$ bytes. Substituting h=48 and proto=8 we get the change-over point to be about 32 bytes. As the network throughput is higher for 256 byte packets than for 32 or 64 byte packets this change over point may actually be larger than 32 bytes.

Performance: We explored a virtual 2D Mesh of dimensions 32×16 on an $8 \times 8 \times 8$ torus mid-plane. The 32 processors of each row of the 2D virtual mesh are spread out on half of an XY plane of the physical 3D torus. In the first phase of the mesh strategy, all processors in 8×4 half-XY planes perform all-to-all exchange with each other. In the second phase, Z pencil i and $i + 4$ exchange messages with each other. The performance prediction of the 2D mesh strategy is presented in Figure 5. The prediction assumes $\alpha = 1.7us$, $\beta = 6.48ns/byte$, $\gamma = 1.6ns/byte$.

Figure 6 compares the performance of the virtual mesh scheme and AR direct. For very short messages the performance of the mesh strategy is about two times better than the AR scheme on a midplane. However, for large messages as the network term (β) is two times larger the AA time for VMesh is twice that of AR. The change over point for the best scheme is between 32 and 64 bytes as predicted earlier.

We did a similar test run on an asymmetric $8 \times 32 \times 16$ torus with 4096 nodes. We map a 128x32 virtual mesh to the torus with the 128 node rows of the virtual mesh mapped to the XZ planes and the columns mapped to the Y lines. Figure 7 compares the performance of the AR, Two Phase Schedule and Virtual Mesh optimization schemes. For an 8 byte message the VMesh scheme is about two times better than TPS and three times better than the AR scheme. The change over point between TPS and VMesh is at 64 bytes. As the AR scheme has network contention on an

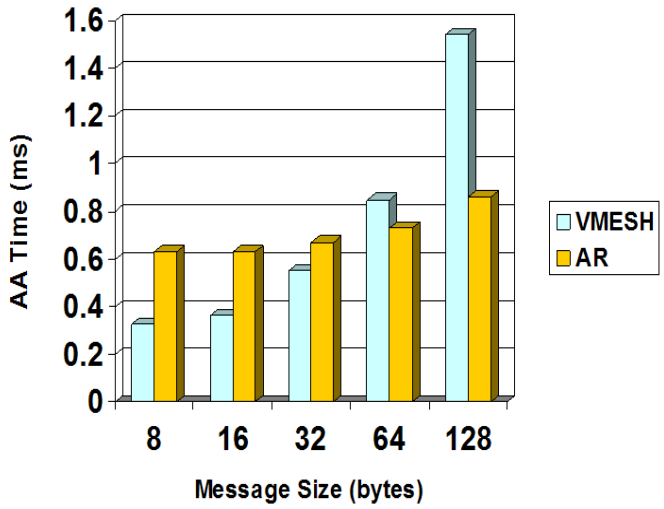

Figure 6. AA performance comparison on 512 nodes

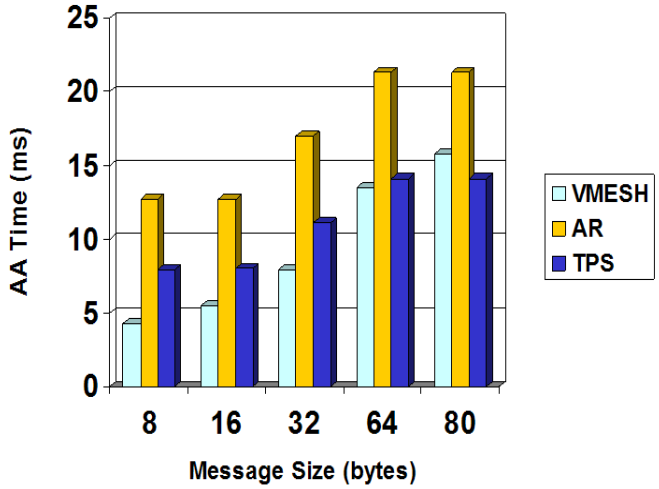

Figure 7. AA performance comparison on 4096 nodes

asymmetric torus it has lower performance as compared with the other two schemes even at 80 bytes.

5 Summary and Future Work

In this paper, we presented a performance model and measurements of all-to-all on Blue Gene/L. The performance model was shown to be an accurate predictor of actual performance on BG/L. More importantly, we presented algorithms for improving all-to-all in several important practical situations. First, for long messages, a Two Phase algorithm was shown to substantially increase all-to-all performance on an asymmetric torus. For example, on our largest partition ($40 \times 32 \times 16$), performance was increased from 72% of peak to over 99% of peak. For all partitions test, all-to-all performance in excess of 95% of peak can be achieved by using our best algorithm: a direct algorithm on a symmetric torus or the Two Phase algorithm on an asymmetric torus. In addition, we presented one possible indirect algorithm for improving small message all-to-all performance by aggregating messages. Better use of network bandwidth is obtained by the larger packet size that comes with message aggregation. Over a two times improvement in performance was obtained for small messages on 512 and 4096 node partitions.

Some modifications to these algorithms would be needed for production quality implementations. In particular, extra memory has to be put aside for the intermediate node forwarding. For very large messages, one would like to bound the amount of extra memory. To do so in a manner that guarantees that the intermediate memory is not overrun requires some sort of flow control. This can be solved fairly easily, and we believe quite efficiently, by a credit-based flow control algorithm in which the intermediate nodes send back short "credit" packets to the sources after forwarding along some number of (large) packets. There is a tradeoff between the amount of intermediate memory and the overhead (bandwidth and processing cycles) of the credit packets. Notice, for example, if one 32 byte credit packet is sent for every ten 256 byte all-to-all packets, the bandwidth overhead is only about 1%.

References

[1] 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):265–276, 2005.

[2] G. Almasi, C. Archer, D. Erway, P. Heidelberger, X. Martorell, J. Moreira, B. Steinmacher-Burow, and Y. Zheng. Optimization of mpi collective communication on bluegene/l systems. In *Proceedings of the 19th annual international conference on Supercomputing ICS '05*, pages 253–262. ACM, 2005.

[3] C. Christara, X. Ding, and K. Jackson. An efficient transposition algorithm for distributed memory clusters. In *13th Annual International Symposium on High Performance Computing Systems and Applications*, 1999.

[4] A. Gara, M. A. Blumrich, D. Chen, G. L.-T. Chiu, P. Coteus, M. E. Giampapa, R. A. Haring, P. Heidelberger, D. Hoenicke, G. V. Kopcsay, T. A. Liebsch, M. Ohmacht, B. D. Steinmacher-Burow, T. Takken, and P. Vranas. Overview of the Blue Gene/L System Architecture. *IBM Journal of Research and Development*, 49(2/3):195–212, 2005.

[5] R. Garg and Y. Sabharwal. Software routing and aggregation of messages to optimize th peformance of hpcc randomaccess benchmark. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing SC '06*. ACM, 2006.

[6] L. V. Kale, S. Kumar, and K. Vardarajan. A Framework for Collective Personalized Communication. In *Proceedings of IPDPS'03*, Nice, France, April 2003.

[7] D. Roweth and A. Moody. Performance of All-to-All on QsNetII. *Quadrics White Paper avaible at www.quadrics.com*, 2005.

[8] A. Tam and C. Wang. Efficient scheduling of complete exchange on clusters. In *ISCA 13th International Conference On Parallel And Distributed Computing Systems*, August 2000.

[9] Thakur and Choudhary. All-to-all communication on meshes with wormhole routing. In *IPPS: 8th International Parallel Processing Symposium*. IEEE Computer Society Press, 1994.

[10] V. Tipparaju and J. Niplocha. Optimizing All-to-All Collective Communication by Exploiting Concurrency in Modern Networks, 2006.

[11] Y. Yang and J. Wang. Near-optimal all-to-all broadcast in multidimensional all-port meshes and tori. *IEEE Transactions on Parallel and Distributed Systems*, 13(2), 2002.