A Study of the Effects of Machine Geometry and Mapping on Distributed Transpose Performance

M. Eleftheriou
IBM Thomas J. Watson Research Center,Yorktown Heights, NY 10598-0218 USA
mariae@us.ibm.com

B. Fitch A. Rayshubskiy T.J.C. Ward
P. Heidelberger R.S. Germain
IBM Thomas J. Watson Research Center,Yorktown Heights, NY 10598-0218 USA
IBM Software Group, Hursley Park
{bgf, arayshu,philiph, rgermain} @us.ibm.com
tjcw@uk.ibm.com

Abstract

This paper describes a parallel strategy to extend the scalability of a small 3D FFT on thousands of Blue Gene/L processors. The approach is to execute the intermediate phases of the 3D FFT on smaller processor subsets. Performance measurements of the standalone 3D FFT on two communication protocols, MPI and BG/L ADE [18] are presented. While the performance of the 3D-FFT with MPI-based and BG/L ADE-based implementations exhibited qualitatively similar behavior, the BG/L ADE-based version has lower communication cost than the MPI based version for small message sizes. Measurements also show that the proposed approach is effective in improving Particle-Mesh-based N-body simulation performance significantly at the limits of scalability.

Categories and Subject Descriptors CR-number [*subcategory*]: third-level

General Terms algorithms, FFT, transposed, performance

Keywords distributed algorithms,Blue Gene/L, distributed transpose, computational biology

1. Introduction

The three dimensional Fast Fourier Transform (3D-FFT) and other spectral methods that require transpose operations are performance critical components in a number of scientific and engineering applications including molecular dynamics [13, 12, 24, 26, 22] and climate modeling. The 3D FFT algorithm and its performance on different machine architecture has been extensively studied [15, 16, 6, 8, 9, 31, 28, 19, 23, 5]. The implementation and performance of a highly scalable 3D-FFT algorithm on Blue Gene/L supercomputer [17, 1] that uses the row-column method and distributes the N^2 1D-FFTs required in each phase of a $N \times N \times N$ FFT have been described previously [10, 11]. In this paper, we present a more detailed study of the effects of machine geometry and mapping of the algorithm to the machine on the performance of the distributed transpose used for the 3D-FFT. For example, the original implementation of the distributed transpose always distributed the data over the largest number of nodes possible. In order to optimize performance, it is interesting to explore what the effects of distributing the data over a subset of nodes might be. Characterizing the trade-off between sending fewer larger messages (with the advantage that header overheads are amortized over a larger payload) and carrying out more computation on each of the nodes within the subset used is of the objective of this work. However, our goal is to provide a scalable 3D FFT on thousands of processor where the number of processors exceeds the number of independent 1D-FFTs to be computed for a given problem size. This flexibility can be used to extend the scalability of molecular dynamics applications such as Blue Matter.

As part of this study we also describe a generalization of the previously described parallel implementation that has a number of advantages:

- Supports scalability to more than N^2 nodes for computing N^3 FFT.
- Enables flexibility in the choice of the number of nodes used for the distributed transpose operation to maximize efficiency.

[Copyright notice will appear here once 'preprint' option is removed.]

- Enables more flexibility in task assignment within an application, e.g. a 3D-FFT can be performed on a subset of nodes while the other nodes can be used for other application work.
- Permits use of real-to-complex 3D FFT on larger partition sizes (real-to-complex 3D-FFT sends half as much data during the transpose operation and effectively does half as many 1D FFTs because of symmetry.
- In some applications that use a 3D FFT to evaluate a convolution, it may be possible to use spherical cutoffs, to reduce communication volumes and to eliminate computations (depending on the structure of functions being convolved).

This paper is organized as follows: Section 3 outlines the network communications on BG/L supercomputer. Sections 4 and 4.2 present performance results for the distributed transpose and the Blue Matter molecular simulation software, respectively. Finally, Section 5 summarizes the conclusions.

2. Software Implementation

The 3D Fast Fourier Transform (3D-FFT) for Blue Gene/L is a C++ template-based parallel library. The template parameters include a communication class that encapsulates the details of the all-to-allv communication collective required for the distributed transpose, a sequential 1D-FFT class and a type class that represents the data type declaration . We have implemented the all-to-allv communication operation using two different communications layers, MPI and the Blue Gene/L Advanced Diagnostic Environment (BG/L ADE) [18]. The BG/L MPI is an optimized port of the MPICH2 library while the BG/L ADE was developed by the BG/L hardware group for machine bring-up and diagnostics. Currently, 1D-FFTs from the IBM ESSL [21] product, FFTW [16] and FFTW-GEL [25] are supported as building blocks for the 3D-FFT. In the results presented here we use the FFTW-GEL library developed by the Technical University of Vienna which currently achieves the best single node performance on BG/L. Finally, the library supports double and float data types.

The 3D FFT library implementation comprises two phases, planning and compute. The planning phase includes:

- heuristic function that is used to determine machine partition based on data sizes
- classes that are used to pre-allocate all the communication and computation buffers used in the FFT phases
- construction of the appropriate group communicators in the MPI based approach.

The planning class is responsible for managing the data distribution. The user can choose the appropriate mapping at run time. If a data mapping is not specified at run time, the default technique is: Check to see if there are sufficient 1D-FFTs to keep all the processors in the system busy. If yes, it uses the previously published implementation and if not, data mapping algorithms are used.

Next, the sub-communicators required for communication are created. Once the parallel approach has been selected, the code subdivides the world communicator group into smaller row and plane shaped non-overlapping processor-groups for each FFT communication phase for the 2 decomposition approach. Note that the user can also choose to use 1D decomposition (slab decomposition); in that case the world communicator group is subdivided into a planes of nodes (perpendicular to a single axis).

In addition, the planning class is responsible for precalculating the lists with the destination and receiving information. Moreover, the data from the collective operation needs to be reordered in order to perform the sequential FFT. We prepare the list in such a way that evaluation of single 1D FFT is possible as soon as the data is received and we put the transformed data in the correct order in the send buffer for the next communication phase. This minimizes the overhead associated with data copying.

The compute phase uses the communication and compute classes. All communications are executed on processor groups using all-to-allv collective operations. In the BG/L ADE-based approach, we have implemented the all-to-allv collective required by the FFT via a low level System Programming Interface (SPI). The SPI interface provides a direct access to the BG/L network-hardware. The major differences between the SPI and MPI all-to-allv implementations are that the SPI packet-headers are prepared at the initialization time while in the MPI-based implementation the packet headers have to be evaluated at all-to-allv runtime and that the MPI implementation requires larger messages to be sent at the limits of scalability because of the messaging protocol used. The computation step simply evaluates all of the 1D-FFTs required for each 3D-FFT phase independently and leaves the data in the transformed format. The compute class is decoupled from the data decomposition which allows deferral of the choice of data decomposition until runtime. Since the library is templatized on the 1D FFT, any serial 1D-FFT implementation may be used within the compute class.

3. Communications on BG/L

The BGL/L ASIC supports five different communication networks [17], only two of which are of interest to application developers: torus [2] and collective communication network. The torus is the main communication network for performing point to point communication and it provides high bandwidth nearest neighbor connectivity. Each node has six bi-directional links to connect with of its neighbor

nodes with 175MB/sec bandwidth on each link. The network hardware delivers packets of variable length ranging from 32 bytes to 256 bytes with a granularity of 32 bytes. The packet header is of size 14 bytes. 8 bytes for link level protocol information, 2 bytes for the acknowledgment and 4 bytes for the checksum.The packets may be dynamically or statically routed.

The torus network supports adaptive routing to enhance the network performance. Adaptive routing allows packets to dynamically find the less congested and minimum path between the sender and nodes. The packet headers include six bits to indicate the direction of the packet routed $(x^-, x^+, y^-, y^+, z^-, z^+)$ [30].

The average number of hops is calculated based on the Manhattan distance between two nodes. The Manhattan distance between a pair of nodes, p and q is given by

$$Hops(p,q) = |x_p - x_q| + |y_p - y_q| + |z_p - z_q|$$

.

The average number of hops for all the messages send by a given node is given by [30].

$$< N_{Hops} >= \sum N_j^{Hops} \times B_j^{Bytes} / \sum B_j^{Bytes}$$

where N_j^{Hops} is the number of hops required for the jth message and B_j^{Bytes} is the corresponding message size.

The 3D-FFT requires an all-to-all on a subset of nodes comprising a column or plane within the three dimensional torus. Let's consider a $16 \times 32 \times 16$ torus and an all-to-all communication along the yz plane. In this case, the number of hops in y direction is double of that the z dimension. That is if x is 100% busy, then y is only 50% busy. Thus the link utilization is only 75%, which means that the bandwidth is the same as in the 32x32 plane-nodes.

The 3D FFT implementation can currently use either of two communication layers: SPI and MPI. The SPI was developed by the BG/L hardware group for machine bring-up and diagnostics. The BG/L MPI implementation is an optimized port of the MPICH2 library [3, 4]. The major difference between MPI and SPI is that the SPI interface provides direct access to the BG/L network hardware while the MPI interface provides a generic message passing and collective API for end user applications. The general purpose MPI communication layer requires additional protocol which implies larger packet headers and this eventually affects performance at the limits of scalability.

Since the all-to-all operation is essential for the 3D-FFT, we developed an efficient and scalable implementation of the all-to-all collective operation for small message sizes using the SPI layer. As an aid in understanding the performance of the transpose operation we will briefly compare the all-to-all implementations in the SPI and MPI communication layers. In the MPI implementation, the all-to-all

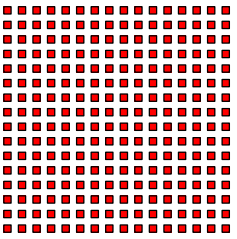


Figure 1. Full distribution. Each red square represents a processor with data

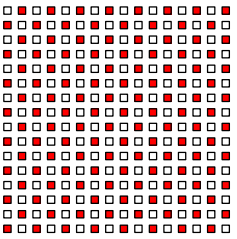


Figure 2. Compressed Checkerboard Distribution. Each red square represents a processor with data while a white square represents a processor without any data

communication has larger overhead for smaller message sizes because its smallest packet size is 64 bytes versus the 32 bytes used by the SPI implementation. Further, the SPI implementation of the all-to-all allows data transfer between the sending and receiving nodes without any protocol overhead of the receivers (eager protocol). That is, the SPI all-to-all doesn't perform flow control; the sending nodes just push data out and the receivers will know from context what to do with it. The MPI all-to-all global collective sends more context around with messages and there needs to be some way for a receiver to tell a sender to stop sending. Both the SPI and MPI all-to-all operations randomize the sending of packets to destination nodes to avoid creating network hot-spots. Finally, we have performed an application-specific optimization of the SPI implementation of the all-to-all by precalculating all the packet headers during the initialization phase while MPI generates them dynamically.

4. Results

In this section we explore the effects of mapping on the performance of the transpose operation on BG/L. Because the transpose required by the BGL3DFFT algorithm uses three all-to-all communication operations, the first along independent node columns and the rest along independent node planes, the results presented below are for columnar and planar all-to-all collectives implemented with both the MPI and SPI communication layers. Because of the lower overhead incurred by the SPI implementation, the effects

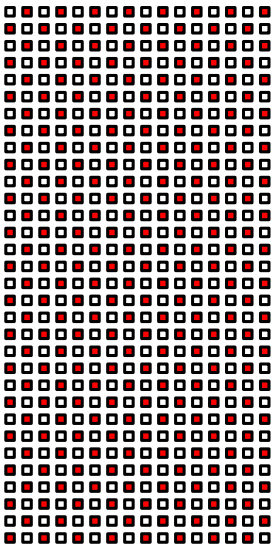


Figure 3. Expanded Checkerboard Distribution.Each red square represents a processor with data while a white square represents a processor without any data

of mapping are more pronounced in that case and our discussion of results focuses on the SPI implementation. The motivation is two fold: i) to evaluate the cost of individual transpose operations using three different data distributions; and ii) show that the expanded checkerboard distribution slightly improves 3D-FFT performance and thus extends the scalability of Blue Matter on BG/L.

4.1 Distributed Transpose performance

Consider performing a 64^3 complex to complex (double precision) 3D-FFT on two different partition sizes, $16 \times 32 \times 16$ and $16 \times 16 \times 16$ nodes where both use only half of the available nodes in each partition for 1D-FFT computation. The nodes used for 1D-FFT computation in each partition form a checkerboard pattern. We refer to them as the "expanded checkerboard distribution" the "compressed checkerboard distribution" respectively. The data is initially distributed equally to all nodes, each of which holds, $n_x \times n_y \times n_z$ complex double data, where $n_i = \frac{N_i}{P_i}$ and i denotes x, y or z.

Figures 2 and 3 illustrate the data distributions (in a single plane) of each communication phase for both "checkerboard" distributions. Figure 1 shows the corresponding data distributions for computing the 3D-FFT using all 4096 nodes, the "full distribution", for comparison.

Tables 1, 2 and 3 show the performance data for the full and both checkerboard distributions. Column one describes the data decomposition and the number of nodes performing the 3D-FFT out of the total available nodes in the machine. For example, checkerboard (4096/8192) means that originally, the data is distributed on 8192 nodes while the 3D-FFT is computed on 4096 nodes. Columns 4 and 5 contain the total elapsed time per communication phase for the corresponding data distributions. In all the performance tables, we provide the results in terms of average elapsed time over 100 runs (omitting the first run). The error bars signify the standard deviation from the average value.

Columns 4 and 5 in Tables 1, 2, and 3 compare the distributed transpose performance in all 3D-FFT communication phases on both the MPI and SPI-based implementations. In general, all the MPI and SPI results are consistent in the sense that they exhibit similar relative performance behavior for a given case. We observe that the SPI-based all-to-all implementation is about 2 to 4 times faster than the MPI-based implementation for very small size messages. For small messages the software overhead associated with sending a packet in the MPI-based implementation is larger than the SPI. For example, in the expanded checkerboard distribution each node sends data comprising a complex double (32 bytes) in the planar transpose. We note that 32 bytes data fit in a single SPI packet while in the MPI-based approach the messages are double the size (64 bytes, MPI has 3 quads of protocol, 48 bytes). Thus, the SPI approach is superior at the limits of scalability.

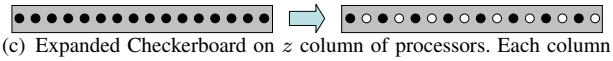
Table 1 shows the performance of the all-to-all operation along the z column nodes and Figure 4 the z plane of all studied decompositions. Let's compare the compressed checkerboard decomposition with the full decomposition using the SPI results. Since all the 1D-FFTs along the z dimension are independent we need only consider the case of a one-dimensional grid of the P_z nodes that compute $n_x \times n_y = 16$ one-dimensional FFTs of size 64. In both distributions, the number of nodes along the z axis is the same, P_z. In the full distribution every node exchanges data with all the nodes in the same column, while in the checkerboard distribution each node sends data only to half of the nodes in the column. For instance, in the full distributions, each node in a column in the torus sends packets of data length of 110 bytes. That is, a 64 byte payload (4 complex doubles) will fit into a bytes packet. Each packet also has a 4 byte CRC appended to it and results in an 8 byte acknowledgment; in addition there are typically 2 bytes of unused space in each packet. Thus a single 64 byte packet results in a 100 byte load on the link with a packet utilization of $64/110 = 58\%$.

In the checkerboard distribution, there are 16 sender nodes, each sending 8 complex doubles to 8 destination nodes along a column of nodes in the torus. That is, each nodes in the checkerboard distribution sends a packet of size 174 bytes to alternate nodes, with packet utilization $128/174 = 73\%$.

The average number of hops is 4, the same for both distributions, using the torus network. Overall we observe efficiency $29.58/35.841 = 0.82$ for the communication

(a) Full distribution on z column of processors. Each column of processor along the z axis computes 16 1D-FFTs of size 64, with each processor computing one 1D FFT. In the communication phase, all nodes of the left blue column sends data to the nodes of the column on the right of the arrow.

(b) Compressed checkerboard distribution on the z column of processors. Each column of processor along the z axis computes 16 1D-FFTs of size 64, with each alternative processor computing two 1D FFTs. In the communication phase, all nodes of the left blue column sends data to half of the nodes(filled circles) of the column on the right of the arrow.



(c) Expanded Checkerboard on z column of processors. Each column of processor along the z axis computes 8 1D-FFTs of size 64, with each alternative processor computing one 1D FFTs. In the communication phase, all nodes of the left blue column sends data to half of the nodes(filled circles) of the column on the right of the arrow.

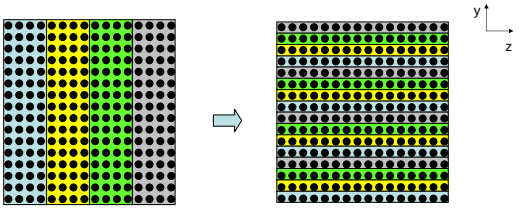
Figure 4. Data distributions along the z processor column. The filled circles represent nodes with data, while the empty represent nodes without any data. On the compute phase, each column of processors along the z axis computes $n_x \times n_y = N_x/P_x \times N_y/P_y$ 1D-FFTs of size Nz, with each processor computing $(n_x \times n_y)/P_z$ 1D FFTs with $n_i = N_i/P_i$.

cost along a given column in the compressed checkerboard distribution, which is consistent with the prediction of $58\%/73\% = 0.79$.

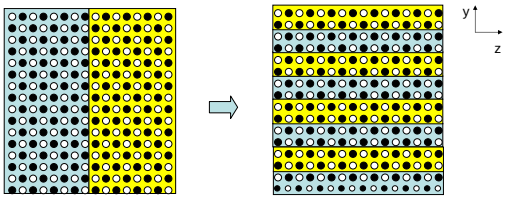
Now, let's compare the full distribution to the expanded checkerboard processor mesh. Every processor along a column of nodes has to compute 16 1D-FFTs in the full distribution and 8 1D-FFTs in the expanded checkerboard distribution. In both, each node sends messages of the same length, 110 bytes. However, in the checkerboard distribution the bandwidth required is halved since each node sends half as many packets. The ratio of the measured communication times is $35.841/19.944 = 1.90$ very close to the factor of two predicted from bandwidth considerations.

Table 2 presents the all-to-all communication cost in the zy-plane. For clarity, Figure 5 shows only the data distribution along a single plane zy of nodes for all studied distributions. Let's first compare the performance of all-to-all in the full and compressed checkerboard distribution.

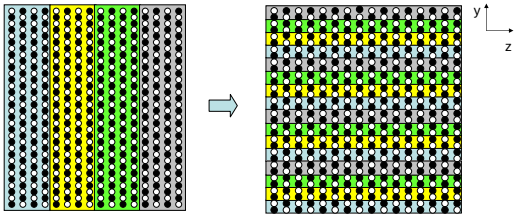
In the full distribution, all nodes (256 total) exchange messages of containing a single complex double with 64 nodes (including itself) in the plane, with packet utilization of $32/32 = 1$. In the compressed distribution, half of the nodes (128 total) exchange data with a packet size of 78 bytes (2 complex doubles) with 64 nodes in a plane of nodes, with packet utilization $64/78 = 0.82$. Thus, we expect $(32 \times 256)/(78 \times 128) = 0.85$ which is consistent with the ratio of measured times, $55.305/58.131 = 0.95$



(a) Full zy plane of 16×16 processors. Each zy plane computes 256 1D-FFTs of size 64, with each processor computing one 1D FFT. In the communication phase, all 256 nodes of the left rectangle sends data to 64 nodes of the right of the arrow.

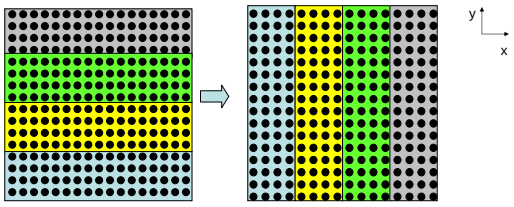


(b) Compressed checkerboard zy plane of 16×16 processors. Each zy plane of processors computes 256 1D-FFTs of size 64, with each processor computing two 1D FFT. In the communication phase, all 128 nodes (filled circles) of the left rectangle send data to 64 nodes of the rectangle on the right of the arrow.

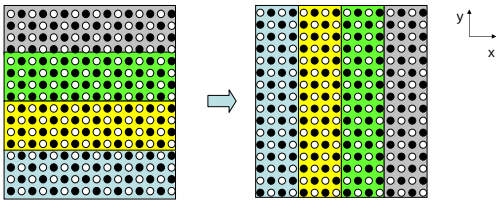


(c) Expanded checkerboard zy plane of 16×32 processors. Each zy plane of processors computes 256 1D-FFTs of size 64 along the y dimension, with each processor computing one 1D FFT. In the communication phase, all 256 nodes(filled circles) of the left rectangle send data to 64 nodes(filled circles) of the rectangles on the right of the arrow.

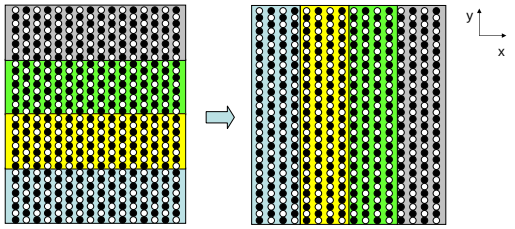
Figure 5. Data distributions along the zy processor plane of size $P_z \times P_y$. The filled circle represent nodes with data, while the empty represent nodes without any data. On the compute phase, each zy plane of processors computes $n_x \times N_z$ 1D-FFTs of size Ny, with $n_i = N_i/P_i$.



(a) Full yx plane of 16×16 processors. Each yx plane of processors computes 256 1D-FFTs of size 64, with each processor computing one 1D FFT. In the communication phase, all 256 nodes of the left rectangle send data to 64 nodes of the rectangle on the right of the arrow.



(b) Compressed checkerboard yx plane of 16×16. Each yx plane of processors computes 256 1D-FFTs of size 64, with each processor computing one 1D FFT. In the communication phase, 128 alternative nodes(filled circles) of the left rectangle send data to the 32 nodes of the rectangle on the right of the arrow.



(c) Expanded checkerboard yx plane of 32×16 processors. Each yx plane of processors computes 256 1D-FFTs of size 64 along the y dimension, with each processor computing one 1D FFT. In the communication phase, all 256 nodes(filled circles) of the left rectangles send data to 64 nodes(filled circles) of the rectangles on the right of the arrow.

Figure 6. Data distributions along the yx processor plane. The filled circle represent nodes with data, while the empty represent nodes without any data. On the compute phase, each yx plane of processors computes $N_y \times n_z$ 1D-FFTs of size Nx, with $n_i = N_i/P_i$.

Next, consider the performance of the expanded checkerboard $(4096/8192)$ distribution versus that of the full distribution $(4096/4096)$. In the expanded checkerboard decomposition the communication is performed on alternate nodes. That is, half of the nodes, 256 out of 512 don't exchange any messages. Each of the remaining 256 nodes sends packets containing one complex double to 64 (including self) destination nodes. Similarly, in the full distributions each node (256 out of 256) sends a single complex double to 64 destination nodes. Thus, the total bandwidth is the same in both distributions. Overall, the performance cost of the communication along the nodes in a plane is sightly higher in the full distribution.

Table 3 presents the all-to-all performance for the xy planes. While Figure **??** presents the yx plane of all studied decompositions. Each node (256 total) in a plane, sends a single complex double to 64 nodes in the full distribution. In the compressed checkerboard, each node (128 total), sends 4 complex doubles to 32 nodes. Thus, we expect $(256\text{nodes} \times 32\text{bytes})/(128\text{nodes} \times 96\text{bytes}) = 0.67$ for the performance ratio consistent with the measured values $43.46/60.48 = 0.71$. In the expanded checkerboard yx plane, each of the filled 256 nodes (total 512) sends data to 64 nodes. Thus, the total bandwidth is the same in both full and expanded distribution, the latency is higher in the expanded distribution.

Finally, let's consider the expanded distributions. Half of the nodes (256 out of 512) send a single complex double to 64 nodes, thus the bandwidth is the same as the one in the full distribution. However, the latency is higher since we have double the number of nodes along the y axis. Thus the measured performance cost is higher in the expanded checkerboard distribution as expected.

We also compare the performance impacts of the topology on the all-to-allv communication. The all-to-allv communication cost along the same plane is much worse on "non-cubic configurations due to internal network congestion" [3]. Similar analysis of the bandwidth and the average number of hops indicates that the performance cost of all 3 communication phases is consistent with the measured values for both geometries of node partitions.

had recently published their work on improving the performance on non-cubic partition.

4.2 Blue Matter application results

The most commonly used methods for efficient calculation of the long range electrostatic interactions interactions in molecular dynamics simulations are the Particle Mesh Ewald (PME)[7] and the Particle Particle Mesh Ewald method (P3ME)[20]. Both methods involve the use of transform techniques requiring computation of 3D-FFTs having global data dependencies. The highly scalable Blue Matter code currently uses the P3ME method in a paral-

Description	Node Count	Node Geom.	Time (μsec.)	
			MPI	SPI
Full distribution (row)	4096	$8 \times 32 \times 16$	77.568 ± 0.135	36.018 ± 0.279
Full distribution (cubic)	4096	$16 \times 16 \times 16$	77.337 ± 0.252	35.841 ± 0.310
Compressed Checkerboard (2048/4096)	4096	$8 \times 32 \times 16$	76.423 ± 0.641	29.072 ± 0.432
Compressed Checkerboard (2048/4096)	4096	$16 \times 16 \times 16$	74.424 ± 0.215	29.580 ± 0.515
Expanded Checkerboard (4096/8192)	8192	$16 \times 32 \times 16$	67.362 ± 0.242	19.944 ± 0.232

Table 1. All-to-all prior to FFT along z-axis

Description	Node Count	Node Geom.	Time (μsec.)	
			MPI	SPI
Full distribution (row)	4096	$8 \times 32 \times 16$	$218.949 \pm .778$	91.677 ± 0.819
Full distribution (cube)	4096	$16 \times 16 \times 16$	155.823 ± 1.044	55.305 ± 0.747
Compressed Checkerboard (2048/4096)	4096	$8 \times 32 \times 16$	217.588 ± 0.225	82.659 ± 1.370
Compressed Checkerboard (2048/4096)	4096	$16 \times 16 \times 16$	157.623 ± 0.980	58.131 ± 0.922
Expanded Checkerboard (4096/8192)	8192	$16 \times 32 \times 16$	186.449 ± 0.468	52.849 ± 0.882

Table 2. All-to-all prior to FFT along y-axis

Description	Node Count	Node Geom.	Time (μsec.)	
			MPI	SPI
Full distribution (row)	4096	$8 \times 32 \times 16$	237.470 ± 1.092	170.674 ± 0.915
Full distribution (cube)	4096	$16 \times 16 \times 16$	153.926 ± 0.326	60.477 ± 0.702
Compressed Checkerboard (2048/4096)	4096	$8 \times 32 \times 16$	194.075 ± 1.212	113.512 ± 0.512
Compressed Checkerboard (2048/4096)	4096	$16 \times 16 \times 16$	125.751 ± 0.756	43.459 ± 0.944
Expanded Checkerboard (4096/8192)	8192	$16 \times 32 \times 16$	213.696 ± 1.020	85.362 ± 0.767

Table 3. All-to-all prior to FFT along x-axis

lel decomposition that utilizes one of the two BlueGene processor cores for the 3D-FFT computation and the other for the calculation of the real space portion of the non-bond interactions[14]. In the limit of very strong scaling the P3ME convolution step is expected to be the limiting factor for scalability. The new parallel decomposition of the 3D-FFT presented here enables improved strong scalability, e.g. continued speed-up of a 64^3 FFT to more than 4096 nodes which is the node count at which each node performs a single 1D-FFT.

Figure 7 shows the elapsed time per time-step of the SOPE molecular system. The SOPE[27] benchmark system consists of 13,758 atoms, utilizes the Velocity Verlet integration technique[29], and performs two 3D-FFTs on every time-step. The results presented here were obtained using BlueMatter software with V5 method running on the BG/L ADE communications SPI[14].

5. Conclusion

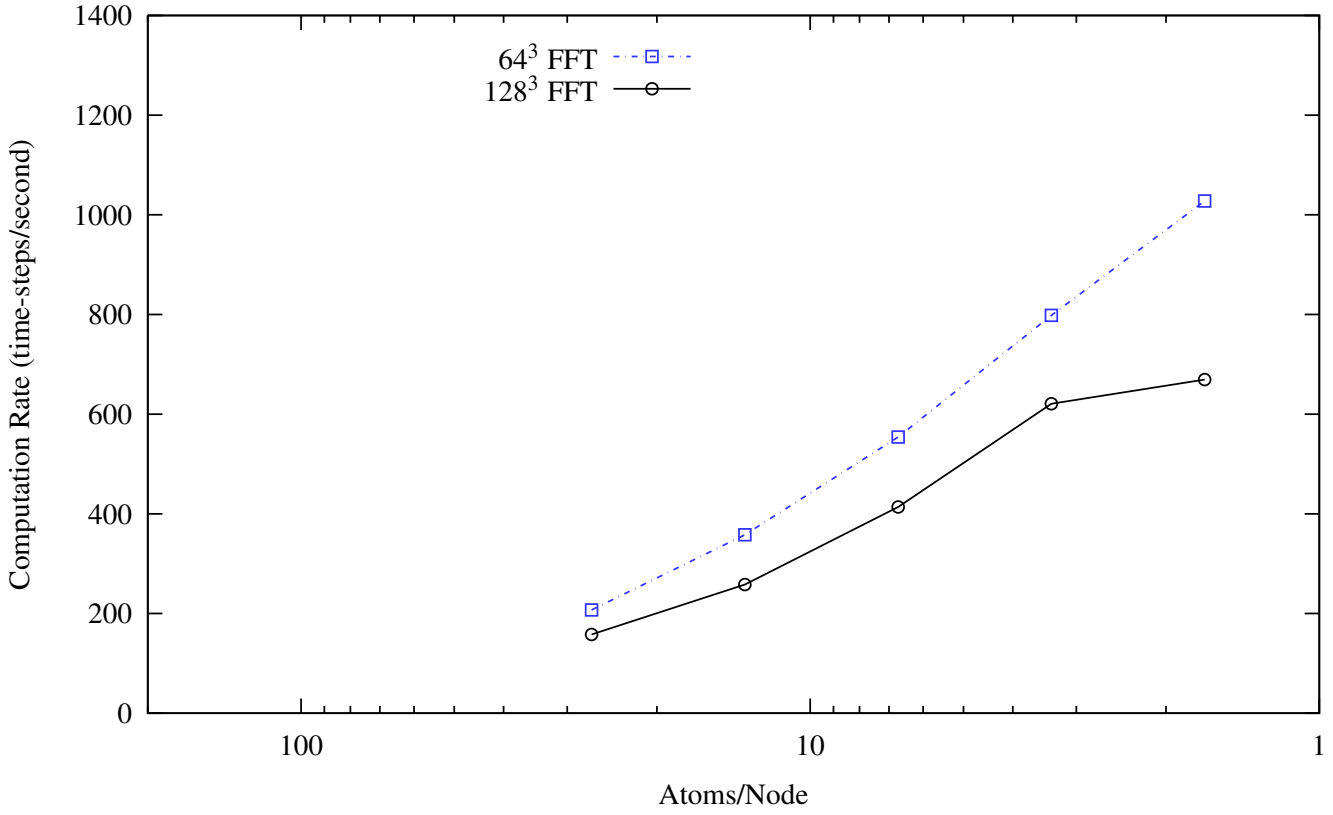
This paper provides a study of the effects of several different parallel mappings of three communication phases of a 3D-FFT on large number of processors. The results of our empirical studies using the 3D-FFT show the impact of the mapping techniques and the machine physical topology on the communication performance of distributed memory algorithm.

Our motivation is to provide a scalable 3D FFT on thousands of processor where the number of processors exceeds the number independent 1D-FFTs to be computed for a given problem size, expanded checkerboard decomposition, without any hit in the performance. The expanded checkerboard decomposition is used by the Blue Matter molecular dynamics application, which utilize one of the cores to evaluate the 3D FFT and the other core to evaluate the real space, to scale beyond one atom per BG/L node. The expanded checkerboard decomposition will enable applications which utilize one of the two BG/L cores per node to compute 3D-FFT to scale to thousands of nodes.

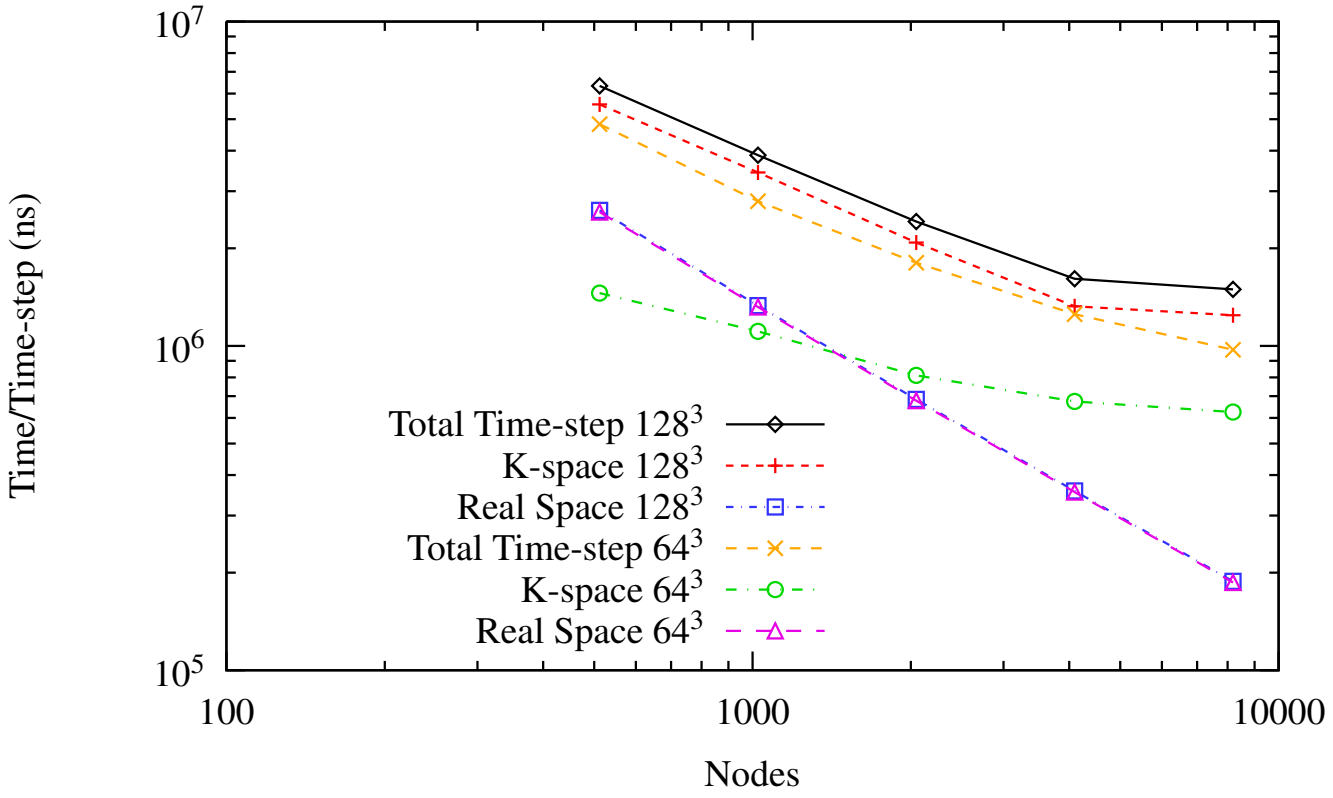
Acknowledgments

References

[1] NR Adiga, G Almasi, GS Almasi, Y Aridor, R Barik, D Beece, R Bellofatto, G Bhanot, R Bickford, M Blumrich, AA Bright, J Brunheroto, C Cascaval, J Castaños,



(a) Application throughput expressed as number of molecular dynamics time-steps per second as a function of the number of atoms per node. Benchmarks are shown for Blue Matter running with 64^3 and 128^3 FFT sizes.



(b) Scalability data for major components of the molecular dynamics time-step shown for two different 3D-FFT sizes, 128^3 and 64^3

Figure 7. The effects of changing the 3D-FFT size on the performance of the molecular dynamics time-step for the SOPE 13,758 atom system are dramatic. Note that generally use of a coarser mesh for the 3D-FFT decreases the accuracy of the computatation of the long range electrostatic forces.

W Chan, L Ceze, P Coteus, S Chatterjee, D Chen, G Chiu, TM Cipolla, P Crumley, KM Desai, A Deutsch, T Domany, MB Dombrowa, W Donath, M Eleftheriou, C Erway, J Esch, B Fitch, J Gagliano, A Gara, R Garg, R Germain, ME Giampapa, B Gopalsamy, J Gunnels, M Gupta, F Gustavson, S Hall, RA Haring, D Heidel, P Heidelberger, LM Herger, D Hoenicke, RD Jackson, T Jamal-Eddine, GV Kopcsay, E Krevat, MP Kurhekar, AP Lanzetta, D Lieber, LK Liu, M Lu, M Mendell, A Misra, Y Moatti, L Mok, JE Moreira, BJ Nathanson, M Newton, M Ohmacht, A Oliner, V Pandit, RB Pudota, R Rand, R Regan, B Rubin, A Ruehli, S Rus, RK Sahoo, A Sanomiya, E Schenfeld, M Sharma, E Shmueli, S Singh, P Song, V Srinivasan, BD Steinmacher-Burow, K Strauss, C Surovic, R Swetz, T Takken, RB Tremaine, M Tsao, AR Umamaheshwaran, P Verma, P Vranas, TJC Ward, M Wazlowski, W Barrett, C Engel, B Drehmel, B Hilgart, D Hill, F Kasemkhani, D Krolak, CT Li, T Liebsch, J Marcella, A Muff, A Okomo, M Rouse, A Schram, M Tubbs, G Ulsh, C Wait, J Wittrup, M Bae, K Dockser, L Kissel, MK Seager, JS Vetter, and K Yates. An overview of the Blue Gene/L supercomputer. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–22, November 2002.

[2] 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.

[3] G. Almasi, C. Archer, J.G. Castanos, J.A. Gunnels, C.C. Erway, P. Heidelberger, X. Martorell, J.E. Moreira, K. Pinnow, J. Ratterman, B.D. Steinmacher-Burow, W. Gropp, and B. Toonen. Design and implementation of message-passing services for the Blue Gene/L supercomputer. *IBM Journal of Research and Development*, 49(2/3):393–406, 2005.

[4] G. Almaśi, C. Archer, C. Chris Eway, Philip Heidelberger, X. Martorell, J. E. Moreira, B. D. Steinmacher-Burow, and Yili Zheng. Optimization of MPI collective operations on BlueGenełsystems. In *ICS '05: Proceedings of the 19th annual international conference on Supercomputing*, pages 253 – 262, New York, NY, USA, 2005. ACM Press.

[5] N. Anupindi, Myoung An, J.W. Cooley, and Qing Yang. A new and efficient fft algorithm for distributed memory systems. In *Parallel and Distributed Systems, 1994. International Conference on*, pages 107–112, 1994.

[6] C. E. Cramer and J. A. Board. The development and integration of a distributed 3D FFT for a cluster of workstations. In *4th Annual Linux Showcase and Conference*, pages 121–128, Atlanta, GA, October 2000.

[7] Tom Darden, Darrin York, and Lee Pedersen. Particle mesh ewald: An $n \log(n)$ method for ewald sums in large systems. *The Journal of Chemical Physics*, 98(12):10089–10092, 1993.

[8] H. Q. Ding, R. D. Ferraro, and D. B. Gennery. A portable 3D FFT package for distributed-memory parallel architecture. In *SIAM Conference on Parallel Processing for Scientific Computing*, 1995.

[9] A. Edelman, P. McCorquodale, and S. Toledo. The future fast Fourier transform? In *SIAM J. Sci. Comput.*, volume 20, pages 1094–1114, 1999.

[10] M. Eleftheriou, B. Fitch, A. Rayshubskiy, T.J.C. Ward, and R.S. Germain. Performance measurements of the 3d FFT on the Blue Gene/L supercomputer. In J.C. Cunha and P.D. Medeiros, editors, *Euro-Par 2005 Parallel Processing: 11th International Euro-Par Conference, Lisbon, Portugal, August 30-September 2, 2005*, volume 3648 of *Lecture Notes in Computer Science*, pages 795–803. Springer-Verlag, 2005.

[11] M. Eleftheriou, B.G Fitch, A. Rayshubskiy, T.J.C. Ward, and R.S. Germain. Scalable framework for 3d FFTs on the Blue Gene/L supercomputer: Implementation and early performance measurements. *IBM Journal of Research and Development*, 49(2/3):457–464, 2005.

[12] B.G. Fitch, R.S. Germain, M. Mendell, J. Pitera, M. Pitman, A. Rayshubskiy, Y. Sham, F. Suits, W. Swope, T.J.C. Ward, Y. Zhestkov, and R. Zhou. Blue Matter, an application framework for molecular simulation on Blue Gene. *Journal of Parallel and Distributed Computing*, 63:759–773, 2003.

[13] Blake G. Fitch, Aleksandr Rayshubskiy, Maria Eleftheriou, T.J. Christopher Ward, Mark Giampapa, Michael C. Pitman, and Robert S. Germain. Blue matter: Approaching the limits of concurrency for molecular dynamics. Research Report RC23956, IBM Research Division, April 2006. To appear in the Proceedings of the 2006 ACM/IEEE conference on Supercomputing http://sc06.supercomputing.org/schedule/pdf/pap246.pdf.

[14] Blake G. Fitch, Aleksandr Rayshubskiy, Maria Eleftheriou, T.J. Christopher Ward, Mark Giampapa, Michael C. Pitman, and Robert S. Germain. Molecular dynamics—blue matter: approaching the limits of concurrency for classical molecular dynamics. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 87, New York, NY, USA, November 2006. ACM Press.

[15] M. Frigo and S. G. Johnson. The fastest Fourier transform in the west. Technical Report MIT-LCS-TR-728, Laboratory for Computing Sciences,MIT, Cambridge, MA, 1997.

[16] M. Frigo and S. G. Johnson. FFTW: An adaptive software architecture for the FFT. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 3, pages 1381–1384, 1998.

[17] A. Gara et al. Overview of the Blue Gene/L system architecture. *IBM Journal of Research and Development*, 49(2/3):195–212, 2005.

[18] M.E. Giampapa, R. Bellofatto, M. A. Blumrich, D. Chen, M. B. Dombrowa, A. Gara, R. A. Haring, P. Heidelberger, D. Hoenicke, G. V. Kopcsay, B. J. Nathanson, B. D. Steinmacher-Burow, M. Ohmacht, V. Salapura, and P. Vranas. Blue Gene/L advanced diagnostics environment. *IBM Journal of Research and Development*, 49(2/3):319–332, 2005.

[19] P. D. Haynes and M. Cote. Parallel fast Fourier transforms

for electronic structure calculations. *Comp. Phys. Comm.*, 130:121, 2000.

[20] R.W. Hockney and J.W. Eastwood. *Computer Simulation Using Particles*. Institute of Physics Publishing, 1988.

[21] IBM. *ESSL for AIX V4.2/ESSL for Linux on POWER V4.2.2 Guide and Reference*, 4 edition, November 2005. SA22-7904-03.

[22] Laxmikant Kale, Robert Skeel, Milind Bhandarkar, Robert Brunner, Attila Gursoy, Neal Krawetz, James Phillips, Aritomo Shinozaki, Krishnan Varadarajan, and Klaus Schulten. Namd2: Greater scalability for parallel molecular dynamics. *Journal of Computational Physics*, 151(1):283–312, May 1999.

[23] Rajkumar Kettimuthu and Sankara Muthukrishnan. A performance study of parallel FFT in clos and mesh networks. In Hamid R. Arabnia, editor, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2005, Las Vegas, Nevada, USA, June 27-30, 2005, Volume 3*, pages 1056–1062. CSREA Press, June 2005.

[24] Ross A. Lippert, Kevin J. Bowers, Ron O. Dror, Michael P. Eastwood, Brent A. Gregersen, John L. Klepeis, Istvan Kolossvary, and David E. Shaw. A common, avoidable source of error in molecular dynamics integrators. *Journal of Chemical Physics*, 126:046101, 2007.

[25] J. Lorenz, S. Kral, F. Franchetti, and C.W. Ueberhuber. Vectorization techniques for the Blue Gene/L double FPU. *IBM Journal of Research and Development*, 49(2/3):437–446, 2005.

[26] D. A. Pearlman, D. A. Case, J. W. Caldwell, W. S. Ross, T. E. Cheatham III, DS. DeBolt, D. Ferguson, G. Seibel, and P. Kollman. *Comp. Phys. Commun.*, 91:1–41, 1995.

[27] Michael C. Pitman, Frank Suits, Klaus Gawrisch, and Scott E. Feller. Molecular dynamics investigation of dynamical properties of phosphatidylethanolamine lipid bilayers. *Journal of Chemical Physics*, 122(24):244715, 2005.

[28] Mohammad Zubair Ramesh C. Agarwal, Fred G. Gustavson. A high performance parallel algorithm for 1-d fft. 1994.

[29] W.C Swope, H.C. Andersen, P.H. Berens, and K.R. Wilson. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *Journal of Chemical Physics*, 76:637–649, 1982.

[30] H Yu, I-H Chung, and J.E Moreira. Topology mapping for Blue Gene/L supercomputer. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, pages 52–52, November 2006.

[31] E.L. Zapata, F.F. Rivera, I. Benavides, J.M. Carazo, and R. Peskin. Multidimensional fast fourier transform into simd hypercubes. *Computers and Digital Techniques, IEE Proceedings-*, 137(4):253–260, 1990.