Scalable framework for 3D FFTs on the Blue Gene/L supercomputer: Implementation and early performance measurements

M. Eleftheriou
B. G. Fitch
A. Rayshubskiy
T. J. C. Ward
R. S. Germain

This paper presents results on a communications-intensive kernel, the three-dimensional fast Fourier transform (3D FFT), running on the 2,048-node Blue Gene®/L (BG/L) prototype. Two implementations of the volumetric FFT algorithm were characterized, one built on the Message Passing Interface library and another built on an active packet Application Program Interface supported by the hardware bring-up environment, the BG/L advanced diagnostics environment. Preliminary performance experiments on the BG/L prototype indicate that both of our implementations scale well up to 1,024 nodes for 3D FFTs of size 128 × 128 × 128. The performance of the volumetric FFT is also compared with that of the Fastest Fourier Transform in the West (FFTW) library. In general, the volumetric FFT outperforms a port of the FFTW Version 2.1.5 library on large-node-count partitions.

Introduction

One of the most commonly used computational approaches for modeling large biophysical systems is the molecular dynamics method, in which the Newtonian equations of motion are solved numerically. Molecular dynamics permits the computation of thermodynamic and dynamic properties of biological systems [1]. Such studies can enhance our understanding of biological functions and provide insight into processes related to the action of potential new medications [2]. One of the most computationally expensive components of this method is the calculation of long-range electrostatic forces [3]. A commonly used technique for computing these forces is the particle-particle particle-mesh Ewald (P3ME) technique [4], which requires the evaluation of a convolution using three-dimensional fast Fourier transforms (3D FFTs). Our interest has been to provide an efficient and scalable 3D FFT implementation on Blue Gene*/L (BG/L). The metric for efficiency used here is the "total time-to-solution" for the problem. Our use cases for the 3D FFT from molecular simulation require strong scaling for relatively small data sizes, such as 32^3, 64^3, and 128^3.

A distributed 3D FFT represents a considerable challenge for the communications infrastructure of a parallel machine because of the all-to-all nature of the distributed transposes required, and it stresses aspects of the machine that complement those addressed by other benchmark kernels, such as Linpack [5].

A typical decomposition for performing a 3D FFT in parallel is slabwise. With a slab decomposition, the data is partitioned along a single axis. For example, to compute an $N \times N \times N$ FFT on P nodes, each node would be assigned a slab of size $N \times N \times N/P$. While effective in reducing communications costs, the scalability of this method is limited by N, the extent of the data along a single axis. This becomes an issue when one wants to scale to very large node counts for a massively parallel machine, such as BG/L.

To address the above issues, we implemented a 3D FFT that is a good match to the volume domain decomposition natural to BG/L. The volumetric FFT achieves scalability to a larger number of nodes because it allows the distribution of work for an $N \times N \times N$ FFT over a maximum of N^2 nodes rather than N nodes for a slab decomposition, but at the cost of additional

0018-8646/05/$5.00 © 2005 IBM

communication. The description of an earlier version of the volumetric FFT implementation and its performance on a more conventional machine (POWER4* full bisectional bandwidth switch) was previously published [6].

The major contribution of our work is a highly scalable 3D FFT framework for BG/L that can use any serial 1D FFT as a building block. To assess the scalability and the performance of the volumetric FFT, we provide the following:

- Scalability characterization of our implementations on BG/L using two communication protocols.
- A comparison with a port of Fastest Fourier Transform in the West (FFTW) [7] on BG/L.

We present a detailed review of the published FFT algorithm [6] and describe a variation to the algorithm that increases its scalability (i.e., increases the number of nodes on which it is able to run). The algorithm was reimplemented to improve the uniprocessor performance and to allow performance comparison on two communication layers: Message Passing Interface (MPI) and active packets. The performance of the volumetric FFT on BG/L for both communication layers is presented. The measured performance is compared with limits inherent in the hardware capabilities of the machine, such as the bandwidth of the BG/L torus communications network. Our measurements show that the volumetric FFT scales well on the BG/L architecture. The volumetric FFT now comes within 50% of the performance of FFTW on a single processor, while outperforming FFTW at larger node counts for a 128^3 FFT. It is important to note that the MPI implementation of the volumetric FFT uses the FFTW 1D serial FFT as a building block.

The paper is organized as follows. We review the volumetric decomposition algorithm for computing the 3D FFT and describe the modified implementation that allows the method to scale to larger numbers of processors. Also in that section, we provide details of the 3D FFT kernel implementation on two different communication protocols. The hardware limits to the ultimate performance of the 3D FFT on a BG/L machine based on the hardware "speeds and feeds" [8] are next discussed, followed by a description of our experimental measurements and corresponding results. Finally, we summarize our work.

Parallel implementation

For a machine with a 3D torus and mesh interconnect, such as BG/L, it is natural to use volumetric decomposition to map the data to perform a 3D FFT onto the machine. Let us assume an array A of complex numbers $N_x \times N_y \times N_z$ distributed onto a $P_x \times P_y \times P_z$ logical processor mesh. Processor (x, y, z) initially stores a block of data $n_x \times n_y \times n_z$ with

$$n_m = \left\lceil \frac{N_m}{P_m} \right\rceil$$

to a local array A'. Then,

$$A'(i,\ j,\ k)[x,\ y,\ z] \equiv A(xn_x + i,\ yn_y + j,\ zn_z + k).$$

To allow scalability beyond that obtainable with the previous proposed implementation, we impose a new restriction on the sizes of the local array, A':

$$n_x \cdot n_y = \alpha \times P_z\,,$$
$$n_x \cdot n_z = \beta \times P_y\,,$$

and

$$n_y \cdot n_z = \gamma \times P_x\,,$$

where α, β, and γ are integers.

In evaluating a 3D FFT on an array A of size $N_x \times N_y \times N_z$, the "rowcolumn" method is used to decompose the problem into successive evaluations of 1D FFTs along each dimension. More specifically, we perform the 3D FFT in three phases; we first compute $N_x \times N_y$ 1D FFTs along the z dimension, then $N_x \times N_z$ 1D FFTs along the y dimension, and finally, $N_y \times N_z$ 1D FFTs along the x dimension. **Figure 1** indicates where the all-to-all exchanges of data take place for each of these three phases.

Each phase of the 3D FFT comprises the communication and computation associated with one of the three dimensions. The communication and computation associated with each phase are analogous. Here we describe only the first phase.

In the first phase, all $N_x \times N_y$ 1D FFTs of size N_z are independent. Therefore, it is sufficient to consider only the case of the 1D grid of P_z nodes that has to compute $n_x \times n_y$ 1D FFTs of size N_z. Suppose $A(0 : n_x - 1, 0 : n_y - 1, 0 : N_z - 1)$ is an $n_x \times n_y \times N_z$ array of complex numbers, block-distributed along the z dimension onto P_z nodes. If

$$A'(0 : n_x - 1, 0 : n_y - 1, 0 : n_z - 1)[p]$$

is the local array in node p, then

$$A'(i,\ j,\ k)[pz] \equiv A(i,\ j,\ pn_z + k).$$

The data along the x and y dimensions is redistributed onto the P_z nodes. That is, if $A''(0 : \alpha, 0 : N_z - 1)[p]$, with $n_x \cdot n_y = \alpha \times P_z$, is the local array in node p after the redistribution, then

$$A''(l,k)[p] \equiv A\left[\left\lfloor \frac{(p\alpha + l)}{n_y} \right\rfloor, (p\alpha + l) \bmod n_y, k)\right].$$

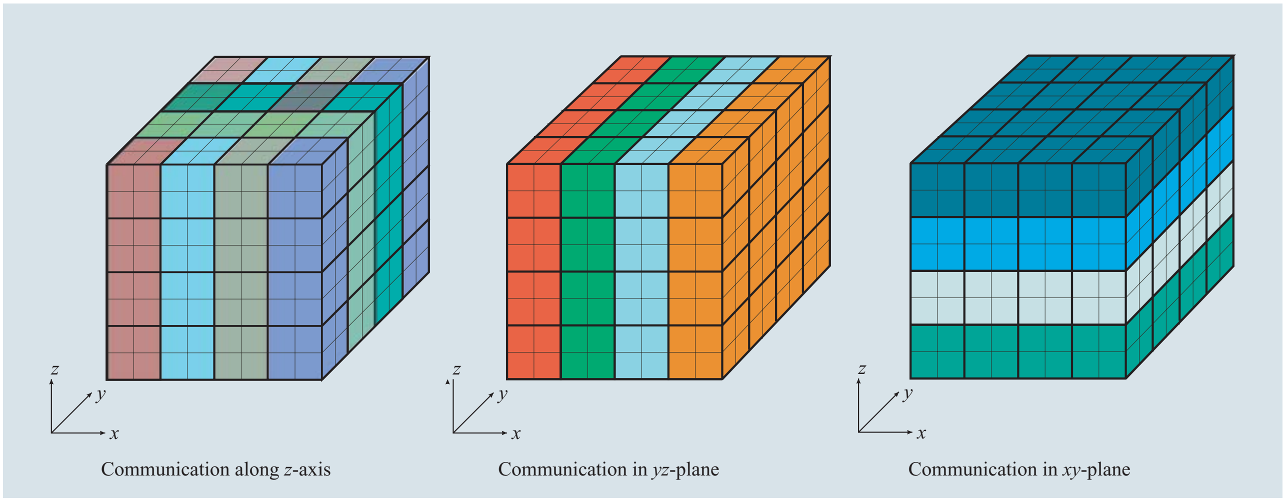
z
y
x
Communication along z-axis
z
y
x
Communication in yz-plane
z
y
x
Communication in xy-plane

Figure 1

Example in which an 8 × 8 × 8 FFT mesh is mapped onto a processor mesh of dimensions 4 × 4 × 4. Each of the three stages of communication required for the FFT is illustrated in this succession of figures.

Each node p sequentially computes a independent 1D FFTs of size N_z that are stored in its local array A''. For these 1D FFT computations, we typically use the services provided by a port of the FFTW library or the FFT library [9] from the Technical University of Vienna to BG/L, although we have also used a simple 1D FFT implementation developed by our group for the packet layer work. Once the FFTs are computed, data is redistributed to the original distribution described by Equation (1).

In summary, the computation of a 3D FFT is performed in three phases, along the z, y, and x dimensions. In each phase, we first redistribute the data, perform 1D FFTs, and return the data to its original distribution. The number of communication operations in our implementation is minimized by combining the post-FFT data redistribution of one phase with the pre-FFT data redistribution of the next phase. Thus, we reduce the data redistributions to three, as opposed to six.

Hardware limits on 3D FFT performance

Lower bounds can be placed on the bandwidth-limited communication time for the 3D FFT assuming that three all-to-all communications (along a row or within a plane) are required. Simulations of the BG/L torus network indicate that the all-to-all communication time for data that is distributed over a set of nodes in a line, plane, or volume can be estimated using the expression[1]

[1]A. Gara, P. Heidelberger, and B. D. Steinmacher-Burow, private communication with author.

$$T_{\text{alltoall}} = \frac{V_{\text{received}}\overline{N_{\text{hops}}}}{N_{\text{links}}BWf},$$

where V_{received} is the volume of data received by each node; $\overline{N_{\text{hops}}}$ is the average number of hops required (for a 3D torus in which each dimension is p, $\overline{N_{\text{hops}}} = p/4$ for alltoall in a line, $\overline{N_{\text{hops}}} = p/2$ for alltoall in a plane, and $\overline{N_{\text{hops}}} = 3p/4$ for alltoall in a volume); N_{links} is the number of links available to each node (two for linear communication, four for planar communication, and six for volumetric communication); BW is the raw bandwidth of the torus per link (two bits per processor clock cycle); and f is the link utilization (assumed to be 80%). This expression indicates that the time required for all-to-all communication is independent of the dimensionality of the communication because the increase of the average hop count with dimensionality is compensated for by the increase in the number of links available for the communication. At the limits of scalability, this expression, on the basis of bandwidth considerations, will become inadequate because the hardware and software latencies associated with sending a packet will become significant.

For an idealized bound on the computation time required to compute a single 1D FFT of length N, we assume $8N \log_2 N$ cycles for a fused multiply–add machine (although the ideal limit for a 1D FFT is $5N \log_2 N$ floating-point operations, data dependencies force a fused multiply–add machine to use eight cycles when one assumes that a fused multiply–add is issued every cycle). Highly optimized FFT implementations, such as the library developed by the team at the Technical University

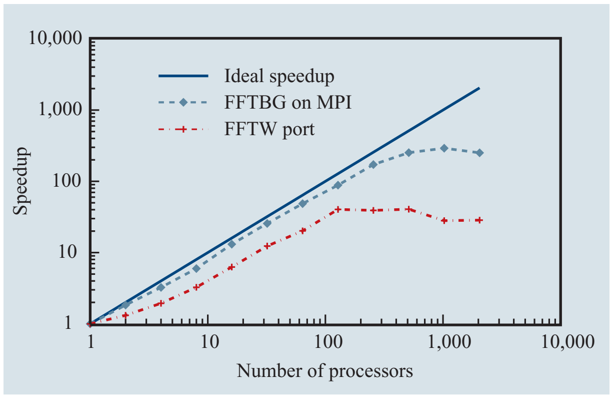

Figure 2

Speedups for the two different FFT libraries, volumetric FFT (FFTBG) on MPI, and the FFTW on the BG/L for a varying number of processors. The 3D FFT has a grid size of 128 × 128 × 128. Note that the network was configured as a full torus for only the 2,048-node data point. All of the data for smaller node counts was taken using a mesh topology for both libraries.

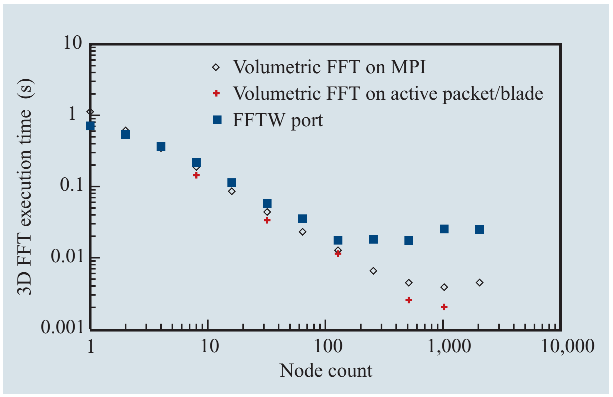

Figure 3

Parallel execution times for the volumetric FFT on both communication layers and the FFTW on the BG/L for a varying number of processors. The 3D FFT has a grid size of 128 × 128 × 128. While the results reported here for the volumetric FFT on active message interface are taken in a torus configuration for node counts of 512 and larger, all of the data on MPI for both the volumetric FFT and FFTW is taken on the mesh topology up to 1,024 nodes. All of the data for node counts below 512 was taken using a mesh topology.

of Vienna [9], can approach this idealized value. However, the nature of the 3D FFT is such that it is communications-bound for even small node counts, and the performance of the 1D FFT building block is not critical.

Parallel performance analysis

All of the performance-measurement results presented in this section were obtained on the 2,048-node Blue Gene/L prototype comprising two racks. Each rack has two 512-processor midplanes, each midplane comprises 16 node cards, each node card contains 16 compute cards, and each compute card contains two nodes. The processor clock speed in the current prototype is 700 MHz. The prototype can be configured to complete a torus in all three dimensions when it is partitioned as 512, 1,024, or 2,048 nodes. Unless otherwise noted, the scalability measurements reported here were taken in mesh mode to simplify the interpretation of the scalability data.

We performed benchmarks of our FFT implementation on the 2,048-node BG/L prototype in which we varied both the number of nodes and the size of data being transformed. For each experiment, the execution time was measured at a series of node counts, while the FFT size was held fixed. The MPI tasks were organized in a 3D grid using the MPI topology constructs. The running times reported here for the MPI implementation are averages of ten runs. Each experiment was run 11 times, and the first run was discarded. The active packet layer execution times were derived from application-specific trace data analyzed after the runs.

We compared the performance of the MPI implementation of the volumetric FFT using the 1D serial FFT from the FFTW library as a building block, and the 3D FFT from the FFTW library [10] on the mesh network. FFTW is a well-established and widely used library in the computational science community for computing multidimensional FFTs. It relies on the slab decomposition approach for running on multiple nodes.

Figure 2 shows the speedup of both the volumetric FFT and FFTW. The speedup achieved by FFTW is observed to be good up to 128 processors. However, the volumetric FFT continues to exhibit good speedups through 1,024 nodes for the 128 × 128 × 128-size FFT, while FFTW loses parallel efficiency above 128 nodes. This flattening on the speedup occurs because FFTW is based on the slab decomposition, which limits its scalability to 128 nodes.

The time-to-solution of the volumetric FFT implementations on both communication layers and FFTW are shown in **Figure 3**. The sequential FFT from the FFTW library is about 50% faster than the single-processor volumetric FFT using the MPI implementation. While both the FFTW and the volumetric FFT have comparable performance up to 128 nodes, the volumetric FFT implementations on both communication protocols continue to scale up to 1,024 nodes.

Figure 4 shows the measured execution times for the volumetric 3D FFT as implemented using MPI collective communications and the active packet interface on the 2,048-node prototype. Speedups in excess of 250 are observed for 128^3 FFTs at 1,024 nodes for both the

MPI and active packet implementations. We observe comparable performance on both communication layers up to 256 processors on the mesh topology. We have currently limited the data available on the two modes—mesh and torus—for a larger number of processors, and that prevents us from providing a complete characterization of the performance characteristics of all implementations on all of the available mesh and torus modes. However, we expect the active packet implementation to be faster at large node counts as a result of the smaller software overheads in the active packet implementation. In addition, the performance of the MPI implementation at the limits of scalability of the FFT should improve as more optimized implementations of the MPI_Alltoall [v] collective operation become available.

The comparison of the bounds on execution time—based on hardware bandwidths and idealized 1D serial FFT execution times with the measured total time to solution in Figure 4—shows significant divergences at small node counts. We conjecture that this divergence from the limits estimated from hardware capabilities is caused by memory hierarchy effects, since floating-point efficiencies of the 1D FFTs used as building blocks for the 3D FFT implementation are fairly high; for example, for a 64-point FFT, the efficiency of the FFT from the BG/L FFT library supplied by the Technical University of Vienna is more than 60%, and the naively vectorized FFT implementation used in the active packet implementation has an efficiency of more than 40%.

The memory access pattern of the in-memory transpose required for the 3D FFT is presumably very unfavorable for prefetching, and at low node counts the memory footprint per node of the larger-size 3D FFTs will certainly spill out of the 4-MB L3 cache. No effort has been made to tile the implementation of the memory transposes in the volumetric FFT. Of course, at the high node counts that represent the limits to scalability for the FFT, the data will sit in cache and the measured performance will more closely approach the bandwidth-limited constraints on performance. Note that even without any effort at tiling, the uniprocessor performance of the volumetric FFT implementation is within 50% of that of the FFTW library implementation. We intend to use instrumentation available on the BG/L chip that can access the hardware performance counters to eventually measure the memory hierarchy effects.

Figure 5 shows the measured data in a way that tries to capture how closely measurements approach bandwidth-limited behavior, as discussed above. We conjecture that the deviations from bandwidth-limited performance at high node counts for the smaller FFT sizes are due to software and hardware overheads that become significant at very small message sizes.

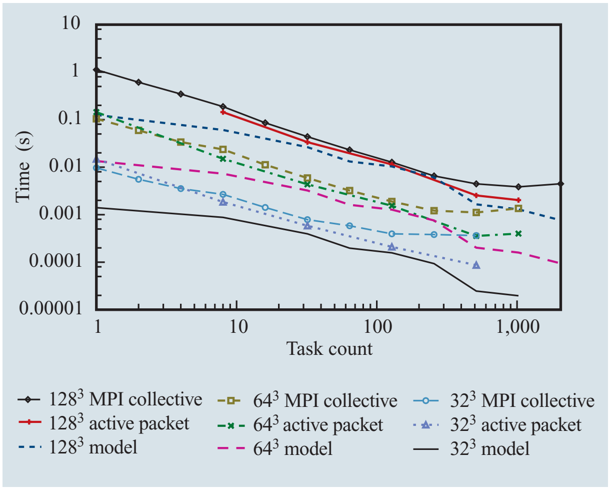

Figure 4

Measured execution times for the volumetric FFT for a series of problem sizes (32^3, 64^3, 128^3) as a function of node (task) count. Note that the network is connected as a full torus only for 2,048 nodes for FFTW and the volumetric FFT on MPI. While the results reported here for the volumetric FFT on active message interface are taken in a torus configuration for node counts of 512 and larger, all of the data on MPI is taken on the mesh topology up to 1,024 nodes. The limits to execution time using simple estimates for computation and communication costs are also shown (the limits shown assume mesh bandwidths, which are half those available on a torus for node counts below 512 and torus bandwidths for larger node counts).

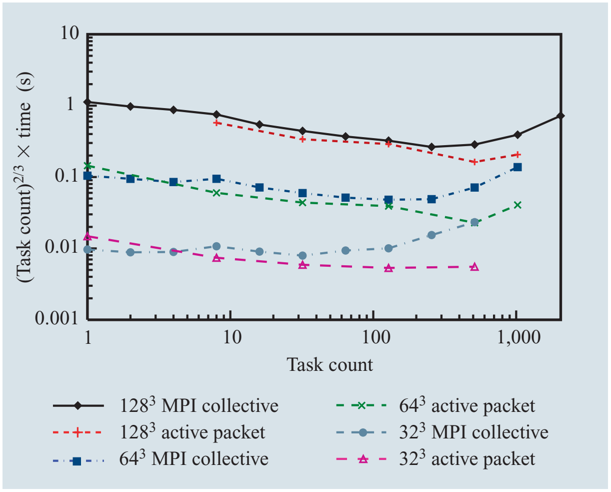

Figure 5

Execution time multiplied by node count to the 2/3 power to make the approach to (and deviations from) bandwidth-limited behavior more clear.

Table 1 The Blue Gene 512-way prototype can be operated with the torus network enabled as a torus or as a mesh (without wrapping). Taking data in both torus and mesh modes allows a check on the amount of time taken by the FFT that is attributable to network bandwidth effects since the effective bandwidth available doubles in torus mode.

Mesh dimension	*Active packet*			*MPI collective*				*Model*	
	32^3	*64^3*	*128^3*	*32^3*	*64^3*	*128^3*	*32^3*	*64^3*	*128^3*
T_{mesh} (512)	1.36×10^{-4}	5.94×10^{-4}	4.55×10^{-3}	—	—	5.01×10^{-3}	4.66×10^{-5}	3.77×10^{-4}	3.05×10^{-3}
T_{torus} (512)	8.5×10^{-5}	3.56×10^{-4}	2.52×10^{-3}	—	—	3.17×10^{-3}	2.47×10^{-5}	2.02×10^{-4}	1.65×10^{-3}
$T_{\text{mesh}}/T_{\text{torus}}$	1.600	1.669	1.806	—	—	1.578	1.889	1.870	1.851
$T_{\text{non-bandwidth}}$	3.40×10^{-5}	1.18×10^{-4}	4.90×10^{-4}	—	—	1.34×10^{-3}	2.74×10^{-6}	2.63×10^{-5}	2.46×10^{-4}
$T_{\text{bandwidth}}$	5.10×10^{-5}	2.38×10^{-4}	2.03×10^{-3}	—	—	1.84×10^{-3}	2.19×10^{-5}	1.76×10^{-4}	1.40×10^{-3}
$T_{\text{non-bandwidth}}/[N^3 \log_2(N)]$	2.08×10^{-10}	7.5×10^{-11}	3.34×10^{-11}	—	—	9.12×10^{-11}	1.67×10^{-11}	1.67×10^{-11}	1.67×10^{-11}
$T_{\text{bandwidth}}/N^3$	1.56×10^{-9}	9.08×10^{-10}	9.68×10^{-10}	—	—	8.76×10^{-10}	6.7×10^{-10}	6.7×10^{-10}	6.7×10^{-10}

In **Table 1**, data taken on a 512-node partition in both the torus and mesh modes is presented. If the execution time of the FFT is totally dominated by bandwidth effects, the ratio of $T_{\text{mesh}}/T_{\text{torus}}$ should equal 2. A ratio in excess of 1.8 is realized for the active packet implementation for the largest FFT size, 128^3, which is impressively close to the ideal hardware behavior.

Conclusions

The parallel computation of three-dimensional FFTs has been studied from two different viewpoints. The first one is aimed at parallelizing the basic one-dimensional FFT [11–13]; in the second, the transpose method, 3D FFTs are carried out by successive evaluations of independent 1D local FFTs along each direction [10, 14–17]. In this work we have presented a volumetric decomposition FFT that belongs to the second category. We have discussed the implementation of a 3D FFT for the Blue Gene/L supercomputer. We base our approach on a volumetric decomposition of data. This decomposition maps naturally to the torus topology of BG/L and allows scaling to very large numbers of nodes. We can also leverage any serial 1D FFT as a building block for our algorithm.

In this paper, we have reviewed the 3D FFT implementation and provided lower bounds on execution time based on the hardware capabilities of the BG/L supercomputer. In our measurements, we found that the volumetric algorithm performs impressively well up through 1,024 nodes on both the active packet and MPI communication layers. Moreover, we found that the volumetric FFT outperforms FFTW by a significant margin on large numbers of nodes. The results obtained thus far are extremely encouraging with respect to our ability to exploit the capabilities of the BG/L architecture in the context of a real application kernel and to enable the scalability of applications that depend on the evaluation of 3D FFTs to the very large node counts required to achieve new levels of performance.

At the limits of scalability, approached by the 32^3 FFT at 512 nodes, the active packet implementation is significantly faster than the MPI-based FFT. This performance difference will presumably narrow as further optimization of the MPI collectives takes place. Future work will involve instrumenting the code to understand the role of memory access patterns in the performance at small node counts and continuing optimization of the implementations on both communication layers.

*Trademark or registered trademark of International Business Machines Corporation.

References

1. M. Karplus and J. A. McCammon, "Molecular Dynamics Simulations of Macromolecules: A Perspective," *Nature Struct. Biol.* **9**, No. 9, 646–652 (2002).
2. F. Allen, G. Almasi, W. Andreoni, D. Beece, B. J. Berne, A. Bright, J. Brunheroto, C. Cascaval, J. Castanos, P. Coteus, P. Crumley, A. Curioni, M. Denneau, W. Donath, M. Eleftheriou, B. Fitch, B. Fleischer, C. J. Georgiou, R. Germain, M. Giampapa, D. Gresh, M. Gupta, R. Haring, H. Ho, P. Hochschild, S. Hummel, T. Jonas, D. Lieber, G. Martyna, K. Maturu, J. Moreira, D. Newns, M. Newton, R. Philhower, T. Picunko, J. Pitera, M. Pitman, R. Rand, A. Royyuru, V. Salapura, A. Sanomiya, R. Shah, Y. Sham, S. Singh, M. Snir, F. Suits, R. Swetz, W. C. Swope, N. Vishnumurthy, T. J. C. Ward, H. Warren, and R. Zhou, "Blue Gene: A Vision for Protein Science Using a Petaflop Supercomputer," *IBM Syst. J.* **40**, No. 2, 310–327 (2001); see *http://www.research.ibm.com/journal/sj/402/allen.html*.
3. R. S. Germain, Y. Zhestkov, M. Eleftheriou, A. Rayshubskiy, F. Suits, T. J. C. Ward, and B. G. Fitch, "Early Performance Data on the Blue Matter Molecular Simulation Framework," *IBM J. Res. & Dev.* **49**, No. 2/3, 447–455 (2005, this issue).
4. M. Deserno and C. Holm, "How to Mesh Up Ewald Sums (II): A Theoretical and Numerical Comparison of Various Particle Mesh Routines," *J. Chem. Phys.* **109**, No. 18, 7678–7693 (1998).
5. A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary, "HPL: A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers," Innovative

Computing Laboratory, Computer Science Department, University of Tennessee, Knoxville, TN, 2004; see *http://www.netlib.org/benchmark/hpl/*.
6. M. Eleftheriou, J. E. Moreira, B. G. Fitch, and R. S. Germain, "A Volumetric FFT for BlueGene/L," *Proceedings of the Conference on High Performance Computing*, 2003, pp. 194–203.
7. See *http://www.fftw.org/*.
8. N. R. Adiga et al. "An Overview of the Blue Gene/L Supercomputer," *Proceedings of the ACM/IEEE Conference on Supercomputing*, 2002, pp. 1–22; see *www.sc-conference.org/sc2002/*.
9. S. Kral, F. Franchetti, J. Lorenz, C. W. Ueberhuber, and P. Wurzinger, "FFT Compiler Techniques," *Proceedings of the 13th International Conference on Compiler Construction, Joint European Conferences on Theory and Practice of Software*, 2004, pp. 217–231.
10. M. Frigo and S. G. Johnson, "FFTW: An Adaptive Software Architecture for the FFT," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998, pp. 1381–1384.
11. E. L. Zapata, F. F. Rivera, I. Benavides, J. M. Garazo, and R. Peskin, "Multidimensional Fast Fourier Transform into SIMD Hypercubes," *J. IEE Proc. E: Computers & Digital Techniques* **137**, No. 4, 253–260 (July 1990).
12. A. Edelman, P. McCorquodale, and S. Toledo, "The Future Fast Fourier Transform?", *SIAM J. Sci. Computing* **20**, No. 3, 1094–1114 (1999).
13. R. C. Agarwal, F. G. Gustavson, and M. Zubair, "A High Performance Parallel Algorithm for 1-D FFT," *Proceedings of the IEEE/ACM Supercomputing Conference*, 1994, pp. 34–40.
14. M. Frigo and S. G. Johnson, "The Fastest Fourier Transform in the West," *Technical Report MIT-LCS-TR-728*, Massachusetts Institute of Technology, Laboratory for Computing Sciences, Cambridge, MA, 1997.
15. H. Q. Ding, R. D. Ferraro, and D. B. Gennery, "A Portable 3D FFT Package for Distributed Memory Parallel Architecture," *Proceedings of the 7th SIAM Conference on Parallel Processing for Scientific Computing*, 1995, pp. 70–71.
16. C. E. Cramer and J. A. Board, "The Development and Integration of a Distributed 3D FFT for a Cluster of Workstations," *Proceedings of the 4th Annual Linux Showcase and Conference*, 2000, pp. 121–128; see *http://www.linuxshowcase.org/2000/2000papers/papers/cramer/cramer_html*.
17. P. D. Haynes and M. Cote, "Parallel Fast Fourier Transforms for Electronic Structure Calculations," *Comp. Phys. Commun.* **130**, No. 1/2, 132–136 (2000).

Received July 20, 2004; accepted for publication September 24, 2004; Internet publication April 12, 2005

Maria Eleftheriou *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (mariae@us.ibm.com).* Dr. Eleftheriou is a Research Staff Member at the IBM Thomas J. Watson Research Center. She received a B.S. degree (with honors) in physics from Saint Joseph's University, and an M.S. degree in engineering and a Ph.D. degree in theoretical and computational chemistry, both from Brown University, in 1995 and 1999, respectively. She subsequently worked as a Postdoctoral Fellow in the Columbia Center for Biomolecular Simulation at Columbia University. Since joining IBM, she has worked primarily on the Blue Gene project. Dr. Eleftheriou has contributed, in particular, to the design and implementation of parallel algorithms and parallel programming models, and studied the performance of parallel scientific applications for the Blue Gene/L architecture.

Blake G. Fitch *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (bgf@us.ibm.com).* Mr. Fitch joined the IBM Thomas J. Watson Research Center in 1985 as a student. He received his B.S. degree in computer science from Antioch College in 1987 and remained at IBM to pursue interests in parallel systems. He joined the Scalable Parallel Systems Group in 1990, contributing to research and development that culminated in the IBM scalable parallel system (SP*) product. Mr. Fitch's research interests have focused on application frameworks and programming models suitable for production parallel computing environments. Practical application of this work includes contributions to the transputer-based control system for the IBM CMOS S/390* mainframes (IBM Boeblingen, Germany, 1994) and the architecture of the IBM Automatic Fingerprint Identification System parallel application (IBM Hursley, UK, 1996). Mr. Fitch joined the Blue Gene project in 1999 as the application architect for Blue Matter, a scalable molecular dynamics package.

Aleksandr Rayshubskiy *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, New York 10598 (arayshu@us.ibm.com).* Mr. Rayshubskiy received an M.E. degree in computer science from Cornell University in 2002. He worked in the Biomolecular Dynamics and Scalable Modeling Group within the Computational Biology Center at the IBM Thomas J. Watson Research Center in 2000 as an intern, joining the group as a full-time software engineer in 2003. Mr. Rayshubskiy worked primarily on the development of the Blue Matter molecular dynamics package. His current research interests include parallel applications, load balancing, performance tuning, and lower-level hardware interfaces to the application.

T. J. Christopher Ward *IBM United Kingdom Limited, Hursley House, Hursley Park, Winchester, Hants SO21 2JN, England (tjcw@uk.ibm.com).* Mr. Ward graduated from Cambridge University in 1982 with a first-class honors degree in electrical engineering. He has worked for IBM in various hardware and software development roles, always finding ways of improving performance of products and processes. He was a member of the IBM Computational Biology Center at the IBM Thomas J. Watson Research Center from 2001 to 2004, arranging for the Blue Gene/L hardware and compilers and the Blue Matter protein folding application to work effectively together and achieve the performance entitlement. Mr. Ward currently works for IBM Hursley as part of the IBM Center for Business Optimization, enabling customers of IBM to take advantage of the opportunities afforded by the rapidly decreasing cost of supercomputing services.

Robert S. Germain *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (rgermain@us.ibm.com)*. Dr. Germain manages the Biomolecular Dynamics and Scalable Modeling Group within the Computational Biology Center at the IBM Thomas J. Watson Research Center. He received his A.B. degree in physics from Princeton University in 1982 and his M.S. and Ph.D. degrees in physics from Cornell University. He joined the Thomas J. Watson Research Center as a Research Staff Member in the Physical Sciences Department after receiving his doctorate in 1989, and later the VLSI/Scalable Parallel Systems Packaging Department. Dr. Germain was project leader, from 1995 to 1998, for the development of a large-scale fingerprint identification system using an indexing scheme (FLASH) developed at IBM Research. He has been responsible for the science and associated application portions of the Blue Gene project since 2000. His current research interests include the parallel implementation of algorithms for high-performance scientific computing, the development of new programming models for parallel computing, and applications of high-performance computing to challenging scientific problems in computational biology. Dr. Germain is a member of the IEEE and the American Physical Society.