Early performance data on the Blue Matter molecular simulation framework

R. S. Germain
Y. Zhestkov
M. Eleftheriou
A. Rayshubskiy
F. Suits
T. J. C. Ward
B. G. Fitch

Blue Matter is the application framework being developed in conjunction with the scientific portion of the IBM Blue Gene® project. We describe the parallel decomposition currently being used to target the Blue Gene/L machine and discuss the application-based trace tools used to analyze the performance of the application. We also present the results of early performance studies, including a comparison of the performance of the Ewald and the particle-particle particle-mesh (P3ME) methods, compare the measured performance of some key collective operations with the limitations imposed by the hardware, and discuss some future directions for research.

Introduction

One of the two major goals of the Blue Gene* project announced by IBM in December 1999 was the application of the massive computational power being developed for that project to advance our understanding of biologically important phenomena, such as protein folding via large-scale simulation [1]. Studying the mechanisms behind protein folding and related areas can require long biomolecular simulations on a wide range of system sizes. The application supporting this work must map well onto a range of parallel cluster sizes to optimize scientific throughput for a particular study. System sizes of interest range from 5,000 atoms (for a small peptide in water) through 30,000 to 50,000 atoms (for a folded 200-residue protein in water or a small protein membrane system) up to 200,000 atoms or more (for larger-scale membrane systems or an unfolded protein in water). Our design point targets system sizes in the 10,000- to 100,000-atom range.

The application effort we have made as part of the Blue Gene project also has two goals: first, to support the scientific goal of the Blue Gene project related to the use of biomolecular simulation for studies of biologically important phenomena, and second, to explore novel approaches to programming applications that target massively parallel system architectures in the context of a concrete problem. To address these goals, we have developed an application framework, called *Blue Matter*, that is currently focused on biomolecular simulation, and whose architecture has previously been described [2]. A version of Blue Matter running on conventional hardware (IBM pSeries*) has been used in production for scientific work [3], and we are now doing production runs on Blue Gene/L (BG/L) prototype hardware.

One of the principal design goals for this framework is to encapsulate the semantics of biomolecular simulation through the definition of appropriate interfaces so that explorations of performance and scalability alternatives can take place with minimal intervention from the domain experts on the team. Furthermore, we have decomposed the application into a core parallel engine that runs on BG/L and a set of support modules providing setup, monitoring, and analysis functionality that can run on other host machines. Minimizing system environmental requirements for the core parallel engine enables the use of nonpreemptive low-overhead parallel operating system kernels [4] to enable scalability to thousands of nodes.

In this paper, we describe some early performance data obtained on BG/L using the Blue Matter application. Our methodology for acquiring and analyzing performance data is described, along with the results of a comparison between two algorithmic alternatives for evaluating long-range electrostatic forces. We present a comparison of the measured performance of a key component of our application (in the context of our current parallel decomposition) with the potential performance of the hardware. Finally, we give a brief description of our plans for further explorations.

0018-8646/05/$5.00 © 2005 IBM

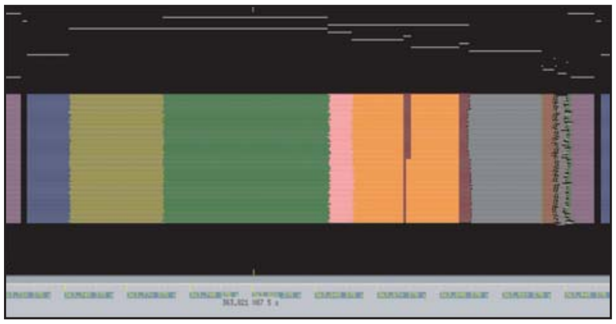

Figure 1

Cropped image of the trace visualization tool showing output from a 512-node run. Timebase synchronization between nodes can be achieved either through synchronization of the clocks across the machine, which is enabled by the hardware, or by inserting a barrier at the start of the run and using this to synchronize the time traces during post-processing. This synchronization permits the observation of skew (differences in the absolute time for the start of some procedure) between nodes as well as load imbalance. The colors in the trace figure represent the following (read from left to right): blue—bonded force calculations; yellow—Verlet list creation (happens on only some timesteps); green—real space non-bond force calculations; pink—P3ME charge meshing; orange—forward and reverse 3D FFTs; brown (between the forward and reverse FFTs)—kernel computation; brown—P3ME force calculation; gray—floating-point allreduce operation on forces; brown—rattle; purple—integer allreduce of positions.

BG/L from an application perspective

While detailed descriptions of the BG/L hardware platform exist elsewhere [5], it is useful to review some of the aspects of the machine that are most interesting to application developers. First, there are two identical IBM PowerPC* 440 central processing units (CPUs) on each chip, and each CPU has a dual floating-point unit (FPU). This FPU executes instructions in single-instruction multiple-data (SIMD) fashion, like a two-element vector processor. Making more effective use of these double FPUs is still an area of intense activity within the Blue Gene project. Although the CPUs are identical, one of the CPUs is intended for use as a communication coprocessor when executing a communication-intensive application. There are two main modes of operation supported by the system software:

- *Coprocessor mode*, in which one CPU is used for computation and the other is used for communication.
- *Virtual node mode*, in which both CPUs carry out both communication and computation and, from the perspective of the application, support independent processes that communicate only via message passing.

The results reported here were all taken in coprocessor mode and with the compiler flag `-qarch=440`, which does not take advantage of the double FPU. As our explorations of parallel decompositions and the work on code generation by the compiler continue, we expect to remove both of these restrictions.

BG/L has two physically separate networks used for high-performance communication between nodes: a three-dimensional (3D) torus network in which every node is connected to its six nearest neighbors, and a hardware collective network that allows rapid broadcasts and reductions. The collective hardware supports fixed-point operations within reductions. For the results reported here, a fixed-point `MPI_AllReduce` implemented on the hardware collective network is used to make atom positions stored on individual nodes visible to all of the nodes in the system (an operation equivalent to `MPI_Allgatherv`). The accumulation of forces on each particle in the system is carried out using a floating-point `MPI_AllReduce` operation implemented on the torus network. The torus network is also used as the communication fabric for the 3D fast Fourier transform (FFT) [6] that forms an integral part of the particle-particle particle-mesh (P3ME) technique used to handle electrostatic forces.

Application-level tracing on massively parallel machines

To tune application performance, including identifying load balance issues on a machine such as Blue Gene/L, tools capable of acquiring, managing, and analyzing data from thousands of nodes are a necessity. To address this need for our own code development, we developed an application-based tracing methodology that is lightweight, can be turned off entirely via recompilation, and for which we have developed tools that can handle hundreds or thousands of nodes. The current implementation has C++ bindings only, but the techniques involved could be used with other languages as well.

Currently, two different sorts of analysis are performed on the trace data created in this way. First, the acquired trace data can be viewed directly in a visualization tool, as shown in **Figure 1**. Second, statistical analysis of the trace data from a series of runs at different node counts can be carried out in an automated way and output in tabular form. Because the trace visualization tool shows the actual timestamps, skew between nodes as well as load imbalance can be identified.

Parallel decomposition and early performance data

The decomposition currently implemented by the Blue Matter application is a *replicated data* decomposition in which a pair of arrays containing all of the positions of particles and the net force on each is maintained. A number of papers have described the taxonomy of parallel decompositions of molecular dynamics along with their scaling properties with respect to space per node, computational burden, and communication volume [7–9]. In particular, when large systems with several hundred thousand particles or more are the target, a spatial decomposition scheme is typically used in which the problem domain is divided into subvolumes assigned to specific nodes. A replicated data scheme was chosen as the first decomposition for exploration primarily because it simplifies load balancing. It has the disadvantage that the amount of memory required on each node and the volume of global communication required are both proportional to the number of particles being simulated and independent of the processor count. For the range of system sizes (10 to 100,000 atoms) and node counts (512 or smaller) that would be available during the initial phases, calculations based on projected hardware capabilities indicated that performance would be adequate [2].

The system of particles is partitioned into fragments that comprise small groups of particles that are connected by bonds. We impose the rule that a bond whose length is constrained cannot cross fragment boundaries. The current decomposition binds individual fragments to a particular node. To support the calculations required for molecular dynamics, the positions of each particle are globalized via an `AllReduce` operation by first doing a fixed-point reduction in a vector of $3N$ double-precision numbers, where N is the number of particles, and then a broadcast. (Both are currently implemented using the hardware collective network.) To avoid replication of pairwise force calculations, the summed forces on each particle are also computed via a global (floating-point) reduction, which is currently implemented on the torus network. Both of these operations are invoked via MPI collective operations on `MPI_COMM_WORLD`. The global force reduction also simplifies load balancing, because contributions to the summed force vector can come from any node.

Although these operations are nonscalable (in the sense that they should be approximately constant time, independent of node count), as long as they represent a small fraction of the total timestep, this approach should be viable. It is instructive to attempt to estimate the ultimate hardware limits to performance for these operations. For example, the position globalization uses the hardware collective and fixed-point operations that are built into the collective hardware. Since the collective network bandwidth is 4 bits per cycle, and assuming that the collective network can stream at full bandwidth in both directions, the time required is approximately 69 ns per particle. The comparisons between the limits established by this estimate and the performance measured at the application layer are given in **Table 1**.

Table 1 Comparison of the hardware bandwidth-limited time and measured times for the integer `AllReduce` on the hardware collective used in the position globalization operation on two different system sizes. The data reported here is taken for 512 nodes. The bandwidth-limited times were computed by simply taking the data volumes for the positions $3 \times N_{\text{atoms}} \times$ sizeof(*double*) and dividing by the collective bandwidth, 4 bits per cycle assuming a 700-MHz processor clock. A very large fraction of the hardware-limited bandwidth (more than 90%) is realized at the application.

Atom count	*Hardware bandwidth-limited time* (ns)	*Integer AllReduce (measured)* (ns)
5,289	363,000	396,000
43,222	2.96×10^6	3.15×10^6

Algorithmic explorations

Ewald implementation

In the direct implementation of the Ewald method [10], the computationally expensive structure factors are evaluated explicitly for all charges and reciprocal space momenta. A number of algorithmic improvements have been introduced to reduce the number of trigonometric function calls and the total number of floating-point operations.

The parallel Ewald algorithm is structured in the following way. The positions and charges of all particles are available to each node after the position globalization operation. Each node is assigned a subset of particles for which it calculates the individual structure factors for all reciprocal space momenta. The global structure factors are summed over all particles for each momentum using the global reduction operation. The forces on particles assigned to a processor are computed using both global and individual structure factors.

The only communication specific to this parallel Ewald algorithm is the global reduction of structure factors. The forces on particles from the reciprocal space contribution of the Ewald method are computed on the nodes to which these particles are assigned and do not require any additional communication. The cost of this communication step should be constant as a function of node count since the volume of floating-point data is fixed.

Parallelization of the P3ME method

In this section we describe the parallel implementation of the P3ME algorithm [11] for an efficient evaluation of the long-range forces. The P3ME method on BG/L is implemented via a volume decomposition of the simulation volume (and the corresponding FFT grid) onto the processor node mesh. We briefly describe the parallel computation of the P3ME method:

- The point charges are mapped to a grid with weights determined by the Euler exponential spline coefficients, the cardinal B-splines. It is assumed that the number of grid points is sufficiently large and the interpolation level is sufficiently small that point-charge positions required for the interpolation have been communicated to each processor as part of the real space intermolecular force evaluation.
- A forward 3D FFT is carried out on the meshed charge data using the algorithm described in a companion paper [6].
- Pointwise multiplication is performed by the kernel: $[4\pi \exp(-g^2/4\alpha_{\mathrm{ewd}}^2)/Vg^2]$.
- Inverse 3D FFT is performed on the resultant product.
- Forces are calculated in real space using the transformed product function and the derivatives of the weighting functions.
- All nodes participate in an all-reduction (floating-point summation) implemented over the torus network, with each node contributing the results of its assigned interactions (from real space and k-space).
- In the current decomposition, each atom is bound to a particular node, and the positions and velocities of each atom are propagated forward in time by numerical integration of the equations of motion.

Parallel performance measurements

Parameters used in simulations

To carry out a fair comparison of the Ewald and P3ME method performance, it is necessary to select parameters that give comparable numerical accuracy for both methods. There exist analytical estimates of the accuracy for both methods [12, 13]. We used the numerical accuracy of the energy calculation as compared with the result at large real-space cutoff and large $K_{\max}$ as a rough measure.

The parameters that affect the accuracy of the reciprocal space of the P3ME method are the mesh spacing size, which is usually selected to be between 0.5 Å and 1.0 Å, the α parameter, the number of mesh points for charge assignment, and the algorithm to compute electric fields at the mesh points. For the latter, the choices in Blue Matter are the "analytical" and "gradient" methods, which are named following [11]. They differ in the number of inverse FFTs involved in force and energy calculations, one for analytical and four for gradient. In the measurements reported here, we use the analytical method. For the Ewald method, the relevant parameters are α and the cutoff momentum, which is proportional to the parameter called $K_{\max}$ and inversely proportional to the box size. The real space contribution accuracy is the same for both methods, as determined by the value of the cutoff in the real space and the parameter α.

For the purposes of the current performance runs, the real-space cutoff was fixed at 10 Å, with the switch function in the potential gradually decreasing to zero within the 1-Å-thick shell. The function is such that the resulting forces are smooth around the switch region. The integrator is velocity Verlet [14], with the timestep size of 1 fs for the 5K-atom hairpin system and 2 fs for the 43K-atom membrane system. These timestep choices give good energy conservation for their respective systems. The charges were assigned to a cube surrounding each particle with four mesh points extant in three dimensions. In the hairpin simulation, we used a mesh size of $64 \times 64 \times 64$; for the larger rhodopsin system, the mesh size was chosen to be $128 \times 128 \times 128$. We set the α parameter to 0.28. The $K_{\max}$ parameter was set to 11 for the hairpin system and 20 for the rhodopsin system. This choice of parameters gave a reasonable accuracy (about six significant digits) for both the Ewald and P3ME methods.

Two different molecular systems were used for these studies. The smaller system was a solvated β-hairpin system comprising 5,239 atoms, including 1,660 single-point-charge (SPC) water molecules, and used the Optimized Potential for Liquid Simulation—All Atom (OPLSAA) [15] force field. The larger system, a solvated lipid/protein system, comprised 43,222 atoms, including 7,400 SPC water molecules and used the CHARMM [16] force field.

Results

We carried out this study to understand the scalability characteristics of our current implementation of the molecular dynamics application on the BG/L architecture using two different methods for evaluating the long-range forces P3ME and Ewald. Message Passing Interface (MPI) was used as the communication layer; the fixed-point collective operation that was used to globalize the particle positions was implemented on the hardware collective network; and the floating-point collective operations used to reduce the force contributions and compute the structure factors for the Ewald technique were implemented on the torus network.

We report performance numbers on 32, 128, 512, and 1,024 nodes [**Figures 2(a)–2(d)**]. All of this data was taken in coprocessor mode, with only one CPU per node used for computation and without any use of the double FPU. Load balancing and autotuning of guard zones

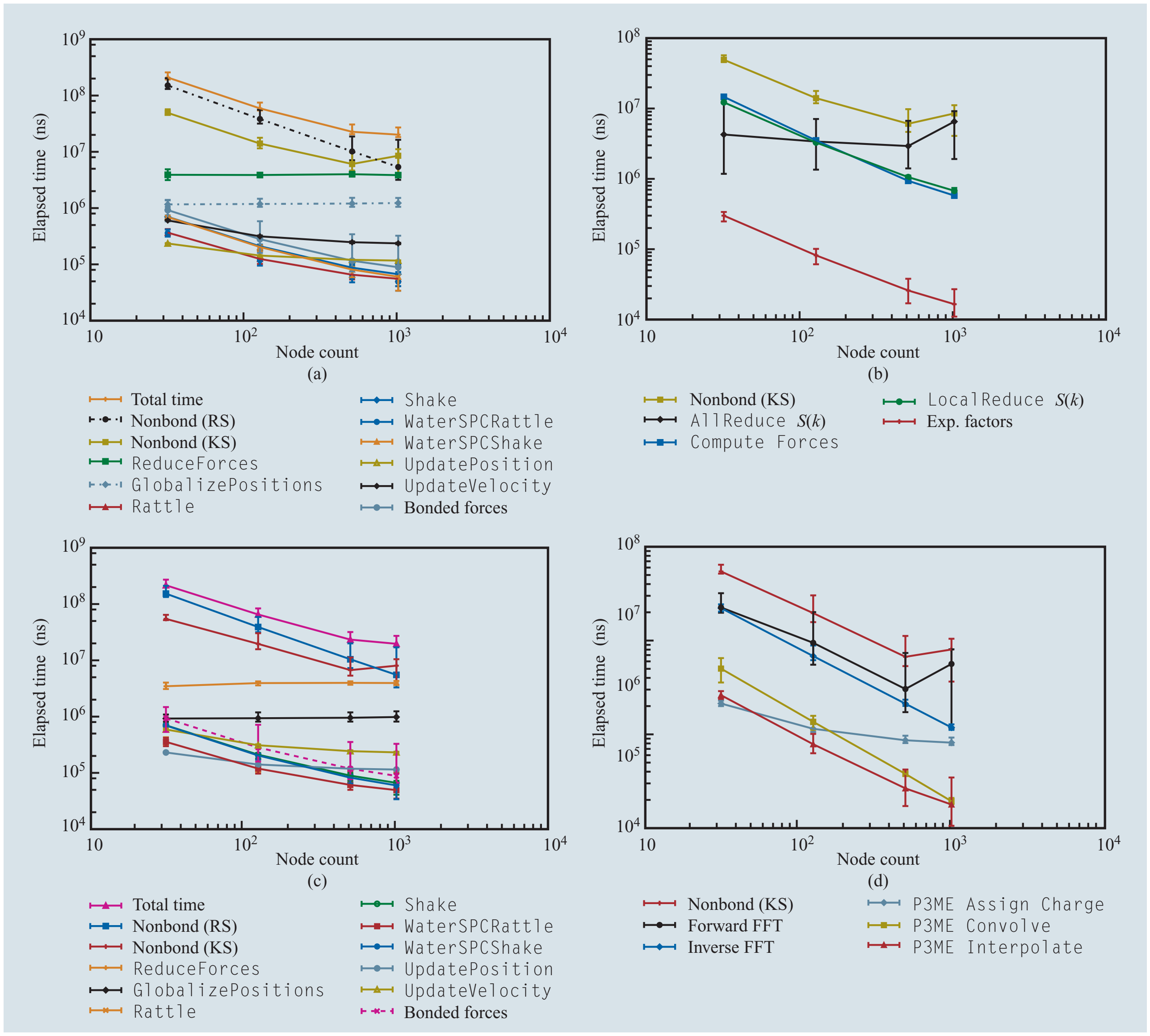

Figure 2

(a) Scaling of contributions to the total timestep using the Ewald method in a 5,289-atom β-hairpin system. The contributions shown are non-overlapping, and the error bars indicate the spread in values (minimum to maximum) over the entire set of nodes and the timesteps used to compute the average values. The major components of the timestep include the real-space (RS) and k-space (KS) nonbonded calculations, the floating-point reduction and broadcast implemented on the BG/L torus network (`ReduceForces`), and the fixed-point `All_Broadcast` implemented on the collective network (`GlobalizePositions`). The other, much smaller, contributions to the total timestep include the `Shake` and `Rattle` calculations that constrain bond lengths in the water molecules and for bonds connecting selected atoms in the peptide, the velocity Verlet integrator, and the bonded force calculations. (b) Scaling of contributions to the k-space portion of the total timestep—"Nonbond (KS)" in 2(a)—using the Ewald method in a 5,289-atom β-hairpin system. The "Exp. factors" label refers to the amount of time required to compute the exponential factors that are the starting point for the Ewald technique. The `AllReduce` operation is a floating-point reduction carried out on the torus network, while the `LocalReduce` operation is the partial reduction that takes place locally before the `AllReduce` operation is carried out. The `Compute Forces` operation comprises the local operations required to compute the k-space-related forces on each particle from the structure factors $S(k)$ and other local quantities. (c) Scaling of contributions to the total timestep using the P3ME method in a 5,289-atom β-hairpin system. The structure of this figure is the same as that of Figure 2(a). (d) Scaling of contributions to the k-space portion of the timestep in a 5,289-atom β-hairpin system. This breakdown shows the breakdown of the total Nonbond (KS) time into its components, including the times for the forward and inverse FFTs. Other contributions include the meshing of the charges (`P3ME Assign Charge`), the convolution (`P3ME Convolve`), and the computation of the k-space contribution to the electrostatic force on a particle at its actual position (`P3ME Interpolate`).

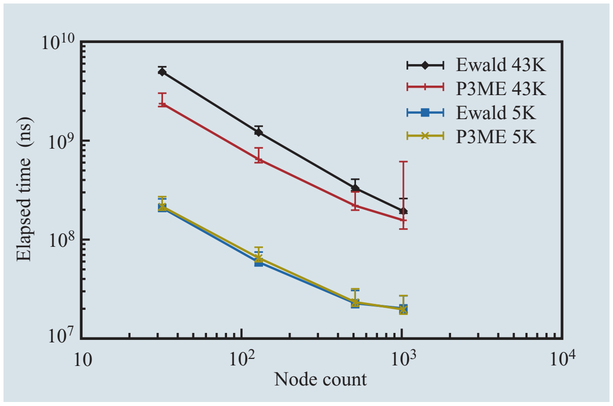

Figure 3

Execution time per timestep as a function of node count for Blue Matter using the Ewald summation technique compared with using the P3ME method. Data for two systems, a solvated β-hairpin comprising 5,239 atoms and a membrane/protein system comprising 43,222 atoms, is shown. The scaling data shown was obtained using executables compiled with `-O3 -g` for the 5,289-atom system and with `-O2 -g` for the 43,222-atom system (because of compiler problems).

for Verlet lists take place during the first few hundred timesteps of the simulation, so our data is obtained by running the simulation for 1,300 timesteps and using the last 100 steps to compute the averages for the CPU times. The component contributions to the total execution time can be measured using the trace tool described in the section above on application-level tracing.

For the Ewald method, a plot of the major contributions is shown in Figure 2(a). The k-space portion of the nonbonded interactions, the portion of the calculation that is different in the P3ME method and in the Ewald method, is further broken down in Figure 2(b). The `AllReduce` is the communication step that performs the global reduction of data over all of the processors to compute the global structure factors $S(k)$. The cost of this communication is essentially independent of the number of processors and is the only nonscalable part that is specific to this Ewald implementation (other nonscalable components are common to both the current Ewald and P3ME implementations). This is the contribution that limits the performance of the reciprocal space Ewald computations at high node counts.

Figure 2(c) shows the parallel performance of the Blue Matter application using the P3ME technique for calculating the long-range electrostatic forces. The data shown in the figure was obtained for a molecular system comprising a solvated β-hairpin (peptide chain) containing 5,289 atoms. In the breakdown of the total time per timestep, the major contributors are, once again, the real space and k-space nonbond calculations; at the higher node counts, the contribution of the floating-point collective (`ReduceForces`) becomes significant. We should note that the initial implementation of the collective used for the `GlobalizePositions` operation was on the torus and had about the same performance as the `ReduceForces` operation. The utilization of the hardware collective network to implement the `GlobalizePositions` operation resulted in significant performance improvement at high node counts, and we expect that eventual optimization of the floating-point collective operation used for the `ReduceForces` operation will yield additional improvements.

Figure 2(d), a more detailed view of the k-space contributions, shows that the forward and inverse FFTs are the dominant contributors, and both scale well up through 512 nodes. At 1,024 nodes, the forward FFT displays a slowdown, while the inverse FFT continues to scale. We believe that this is an artifact caused by differences in data alignment and/or an issue with the current MPI driver rather than a fundamental limitation of this P3ME implementation. The FFT itself would ideally be limited only by the bisectional bandwidth of the machine, which scales as $p^{2/3}$, but in reality, as the node count becomes very large, the message sizes in the FFT decrease to the point at which hardware and software latencies become significant [6].

We also need to discuss the contribution labeled `P3ME Assign Charge` in Figure 2(d). While its contribution is relatively small, its scaling is significantly worse than other parts of the P3ME reciprocal space calculation. The reason is fundamental to the P3ME algorithm itself, because each charge is distributed over several surrounding nodes on the FFT mesh. When this mesh is distributed over more processors, each particle begins to affect more processors, and more communication and/or replicated computation is required. While the scaling is an issue, we believe that further optimization of the `P3ME Assign Charge` operation is possible.

Finally, **Figure 3** compares the performance of the Ewald and P3ME methods on the two systems described above. In the current implementations, the results on the smaller system are comparable, while on the larger system, the P3ME technique still has a significant advantage, although that advantage appears to decrease as the node count increases. At the higher node counts, the `P3ME Assign Charge` operation is taking a significant fraction of the k-space nonbond time and will be a target for further optimization.

It should also be noted that the compilation flags used for the two sizes of systems differed. We were forced to use `-O2`[1] for the compilations on the larger system in

[1] `-O2`, `-O3` are compiler options that control how aggressively the code is optimized (sometimes the optimizations can change the order of operations or possibly the semantics of the code).

this scaling study, while we were able to use `-O3` for the smaller system. The absolute performance on the 43,222-atom system when `-O3` can be used is about 100 milliseconds per timestep (107 ns) at 512 nodes, about twice as fast as the performance measured with `-O2` shown in Figure 3.

As we consider alternative parallel decomposition schemes, it is possible to eliminate the dependency that both methods share on the nonscalable communications operations, position globalization, and global force reduction. However, the global floating-point reduction required by the Ewald technique to compute the structure factors $S(q)$ does not appear to be easily dispensed with, and this may handicap the Ewald technique in future comparisons. One should note that the replication of position and force data on each node also limits the size of the molecular system that can be handled, since the memory footprint required for the replicated data is $6N$ sizeof(*double*), where N is the number of particles.

Summary and future directions

We have presented a snapshot of early performance results on a molecular dynamics application, Blue Matter, that has been developed as an adjunct to the Blue Gene science mission [1]. Using lightweight trace points, we are able to measure the scalability of individual contributions to the execution time of the code, which has allowed us to identify key MPI collective operations whose further optimization will improve both absolute performance and scalability.

As a first step in exploring algorithmic alternatives for the efficient evaluation of electrostatic interactions on massively parallel computers, we have compared the performance of our implementations of Ewald and the P3ME techniques on two system sizes. At large node counts, the performance of the Ewald technique is comparable to that of P3ME for the smaller system size, while, as expected, P3ME shows a significant advantage for larger system sizes. Both of these approaches will benefit from additional improvements in the implementation of the MPI floating-point reduction collective. Our planned explorations include alternative parallel decompositions to reduce or eliminate the nonscalable collective operations currently in use. Further explorations of algorithmic alternatives for the long-range interactions may include the fast multipole method [17] as applied to periodic systems [18]. The results reported here were taken in a mode in which only one of the two PowerPC 440 CPUs on each node were used for computation and without any attempt to have the compiler generate code for the double floating-point unit. Significant improvements in the absolute performance of the compute-intensive contributions to the total timestep duration should be obtainable when both CPUs can be used and as the compiler code generation for the double floating-point unit improves.

Acknowledgments

We acknowledge the contributions of others who have participated in the development of the Blue Matter code, including Michael Pitman, Jed Pitera, Yuk Sham, William Swope, and Ruhong Zhou. We also acknowledge the contributions of the Blue Gene/L hardware and system software teams whose efforts and assistance made it possible for us to use the Blue Gene/L prototype hardware.

*Trademark or registered trademark of International Business Machines Corporation.

References

1. 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).
2. 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," *J. Parallel & Distr. Computing* **63**, 759–773 (2003).
3. W. C. Swope, J. W. Pitera, F. Suits, M. Pitman, M. Eleftheriou, B. G. Fitch, R. S. Germain, A. Rayshubskiy, T. J. C. Ward, Y. Zhestkov, and R. Zhou, "Describing Protein Folding Kinetics by Molecular Dynamics Simulations. 2. Example Applications to Alanine Dipeptide and a β-Hairpin Peptide," *J. Phys. Chem. B* **108**, No. 21, 6582–6594 (2004).
4. B. G. Fitch and M. E. Giampapa, "The Vulcan Operating Environment: A Brief Overview and Status Report," *Parallel Supercomputing in Atmospheric Science*, G.-R. Hoffman and T. Kauranne, Eds., World Scientific Publishing Co., Inc., Riveredge, NJ, 1993, p. 130.
5. 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 J. Res. & Dev.* **49**, No. 2/3, 195–212 (2005, this issue).
6. 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 J. Res. & Dev.* **49**, No. 2/3, 457–464 (2005, this issue).
7. T. P. Straatsma, M. Philippopoulos, and J. A. McCammon, "NWChem: Exploiting Parallelism in Molecular Simulations," *Computer Phys. Commun.* **128**, No. 1/2, 377–385 (2000).
8. L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten, "NAMD2: Greater Scalability for Parallel Molecular Dynamics," *J. Comput. Phys.* **151**, No. 1, 283–312 (1999).

9. S. J. Plimpton and B. A. Hendrickson, "A New Parallel Method for Molecular-Dynamics Simulation of Macromolecular Systems," *J. Comput. Chem.* **17**, No. 3, 326–337 (1996).
10. P. P. Ewald, "Evaluation of Optical and Electrostatic Lattice Potentials," *Ann. Phys. Leipzig* **64**, 253–287 (1921).
11. M. Deserno and C. Holm, "How to Mesh Up Ewald Sums. II. An Accurate Error Estimate for the Particle–Particle–Particle-Mesh Algorithm," *J. Chem. Phys.* **109**, No. 18, 7678–7693 (November 1998).
12. H. G. Petersen, "Accuracy and Efficiency of the Particle Mesh Ewald Method," *J. Chem. Phys.* **103**, No. 9, 3668–3679 (September 1995).
13. J. Kolafa and J. W. Perram, "Cutoff Errors in the Ewald Summation Formulae for Point Charge Systems," *Molec. Simul.* **9**, 351–368 (1992).
14. 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," *J. Chem. Phys.* **76**, 637–649 (1982).
15. W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives, "Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids," *J. Amer. Chem. Soc.* **118**, 11225–11236 (1996).
16. A. D. MacKerell, Jr., B. Brooks, C. L. Brooks III, L. Nilsson, B. Roux, Y. Won, and M. Karplus, "CHARMM: The Energy Function and Its Parameterization with an Overview of the Program," *The Encyclopedia of Computational Chemistry*, Vol. 1, P. v. R. Schleyer et al., Eds., John Wiley & Sons, Berne, Switzerland, 1998, pp. 271–277.
17. L. Greengard and V. Rokhlin, "A Fast Algorithm for Particle Simulations," *J. Comput. Phys.* **73**, No. 2, 325–348 (December 1987).
18. F. Figueirido, R. M. Levy, R. Zhou, and B. J. Berne, "Large Scale Simulation of Macromolecules in Solution: Combining the Period Fast Multipole Method with Multiple Time Step Integrators," *J. Chem. Phys.* **106**, No. 23, 9835–9849 (June 1997).

Received July 29, 2004; accepted for publication October 21, 2004; Internet publication April 12, 2005

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 IBM 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.

Yuriy Zhestkov *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (yuriyz@us.ibm.com).* Dr. Zhestkov is a Research Staff Member in the Blue Gene Application Development Group. He received an M.S. degree (with honors) in applied mathematics and physics from the Moscow Physical and Technical Institute (MPTI), Dolgoprudnyi Moscow region, in 1991. He then worked at the Joint Institute for Nuclear Research in Dubna, Moscow region. In 2001 he received his Ph.D. degree in physics from Columbia University. He then joined the IBM Thomas J. Watson Research Center, primarily working on a highly scalable parallel Blue Matter application for molecular dynamics simulations, targeting the Blue Gene/L supercomputer. Dr. Zhestkov's areas of interest include long-range electrostatics methods in periodic boundary conditions, pressure and temperature control, time-reversible algorithms, and multiple timestep integrators.

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.

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.

Frank Suits *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (suits@us.ibm.com).* Dr. Suits is a member of the Biomolecular Dynamics and Scalable Modeling Group within the Computational Biology Center at the IBM Thomas J. Watson Research Center. This group is responsible for the software and science involved in the protein simulations that are integral to the Blue Gene project. Although his degree is in optical physics, he has worked on a wide variety of projects at the IBM Thomas J. Watson Research Center, including optical storage, magnetic storage materials, scientific visualization, and queuing systems. At present, Dr. Suits is focusing on the analysis of the protein and membrane simulations currently running on BG/L.

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.

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.