Blue Matter: Scaling of N-body simulations to one atom per node

B. G. Fitch
A. Rayshubskiy
M. Eleftheriou
T. J. C. Ward
M. E. Giampapa
M. C. Pitman
J. W. Pitera
W. C. Swope
R. S. Germain

N-body simulations present some of the most interesting challenges in the area of massively parallel computing, especially when the object is to improve the time to solution for a fixed-size problem. The Blue Matter molecular simulation framework was developed specifically to address these challenges, to explore programming models for massively parallel machine architectures in a concrete context, and to support the scientific goals of the IBM Blue Gene® Project. This paper reviews the key issues involved in achieving ultrastrong scaling of methodologically correct biomolecular simulations, particularly the treatment of the long-range electrostatic forces present in simulations of proteins in water and membranes. Blue Matter computes these forces using the particle-particle particle-mesh Ewald (P3ME) method, which breaks the problem up into two pieces, one that requires the use of three-dimensional fast Fourier transforms with global data dependencies and another that involves computing interactions between pairs of particles within a cutoff distance. We summarize our exploration of the parallel decompositions used to compute these finite-ranged interactions, describe some of the implementation details involved in these decompositions, and present the evolution of strong-scaling performance achieved over the course of this exploration, along with evidence for the quality of simulation achieved.

Introduction

Numerical simulation of molecular systems can yield unique insights into the details of the structure and dynamics of biomolecules [1]. Such simulations are used to sample the configurations assumed by the biomolecule at a specified temperature and also to study the evolution of the dynamical system under some specified set of conditions. Among the many challenges facing the biomolecular simulation community, the one that stresses computer systems to the utmost is increasing the timescales probed by simulation in order to better make contact with physical experiment. Even sampling techniques that do not themselves yield kinetic information can benefit from an increased computation rate for a single trajectory.

Classical biomolecular simulation includes both Monte Carlo and molecular dynamics (MD) [2]. The focus of our work has been on MD, although the replica exchange or parallel tempering method [3], which combines MD with Monte Carlo-style moves, has been implemented in Blue Matter as well [4]. Classical MD is an N-body problem in which the evolution of the system is computed by numerical integration of the classical equations of motion. At each timestep, forces on particles are computed, and then the equations of motion are integrated to update the velocities and positions of the particles. The forces on the particles can be classified as follows:

Bonded forces—These forces act between covalently bonded atoms and include bond stretches, angle bends, and torsions.

Nonbonded forces—These forces act between all pairs of particles and include the hard-core repulsion and van der Waals interactions, which are typically modeled by a Lennard–Jones 6-12 potential of the form

$$V_{ij}^{L-J}(r_{ij}) = 4 \cdot \varepsilon_{ij} \cdot \left[(\sigma_{ij}/r)^{12} - (\sigma_{ij}/r)^{6} \right],$$

as well as the electrostatic forces, which have a potential energy of the form

0018-8646/08/$5.00 © 2008 IBM

$V_e(r_{ij}) = q_i \cdot q_j / r_{ij}$.

The forces induced by the Lennard–Jones potential drop off rapidly with distance (varying as $1/r_{ij}^{13}$ and $1/r_{ij}^{7}$) and can be modeled as finite-ranged forces. The electrostatic forces fall off much more slowly with distance (varying as $1/r_{ij}^{2}$) and cannot be approximated simply by neglecting interactions between pairs of particles beyond some cutoff distance (even when a smooth switching function is used) [5].

Ideally, a simulation would model a protein in an infinite volume of water, but this is not practical. Instead, the usual approach is to use periodic boundary conditions so that the simulation models an infinite array of identical cells that contain the biomolecules under study, along with the water. This is generally preferred over simulating a single finite box in a vacuum (or some dielectric) because it eliminates the interface at the surface of the simulation cell. The choice of simulation cell size is important. If the cell is too large, unnecessary computations will be done; too small, and the interactions between biomolecules in different cells of the periodic array can introduce artifacts. The most commonly used techniques for handling the long-range interactions with periodic boundary conditions are based on the Ewald summation method and particle-mesh techniques that divide the electrostatic force evaluation into a real-space portion that can be approximated by a potential with a finite-range cutoff and a reciprocal space portion that involves a convolution of the charge distribution with an interaction kernel [6–9]. This convolution is evaluated using a fast Fourier transform (FFT) method in all of the particle-mesh techniques, including the particle-particle particle-mesh Ewald (P3ME) technique [6, 9] used by Blue Matter. In this case, the $O(n^2)$ dependence on particle number n is reduced to $O(n \log n)$. The evaluation of the three-dimensional (3D) FFT and its inverse on every timestep imposes a global data dependency on the program. That is, the result depends on the position and value of every charge in the system.

Various algorithmic techniques for increasing the effective rate at which the kinetics of the systems evolve have been explored, including kinetic acceleration techniques [10, 11] and multiple timestepping algorithms [12]. Issues such as the difficulty in defining appropriate states between which transitions take place within a simulation have raised concerns about the applicability of the kinetic acceleration techniques to biomolecular simulation. In our experience, even with a correct splitting of the electrostatic forces [13], the use of multiple timestepping leads to significantly larger drifts in the total energy for constant particle number, volume, and energy (NVE) simulations over that obtained using the velocity Verlet integrator [14]. Our view is that direct kinetic simulation is still the most reliable technique for accessing dynamical information over a long timescale. This is in spite of the challenge of scaling a fixed-size N-body problem with some tens of thousands of particles onto a parallel computer with many thousands of nodes (strong scaling). The scale of this challenge can be envisioned by considering that a simulation rate of 1 μs per 2 weeks of wall-clock time requires each MD timestep to complete within 1.2 ms, or fewer than one million processor clock cycles on the IBM Blue Gene/L* machine when using a timestep size of one femtosecond (10^{-15} seconds). Carrying out such a long timescale simulation also potentially exposes correctness issues with the implementation of MD in a particular application. In this context, *correctness* refers to the degree to which the numerically simulated trajectory is representative of an actual trajectory within the model (potential surface) used. For a constant energy simulation, the size of both the short-term fluctuations and especially the long-term drift in the energy can be used as indicators of correctness or the lack thereof (see Reference [15] and the references therein). Given the plateauing of microprocessor clock speeds, any attempt to directly access millisecond timescales would require completion of each MD timestep in fewer than 10,000 cycles.

Massively parallel biomolecular simulation

Prior to the availability of the Blue Gene/L hardware platform [16], the highest degree of strong scaling in the published literature was that achieved by the Nanoscale Molecular Dynamics (NAMD) [17] package, which demonstrated continued speedup through about 60 atoms per processor (and a time per timestep of about 15 ms without multiple timestepping) on the Pittsburgh Supercomputing Center Lemieux system using 1,536 processors [18]. At that time, the NAMD code used a combination of volume and force decompositions [19, 20] for the evaluation of the real-space forces. This made it possible to create a large number of units of work—14 times the number of volume elements in the system, where the dimension of each volume element was larger than a cutoff radius—that could be distributed for load balancing. Subsequently, the NAMD developers incrementally increased the number of units of work available by effectively splitting volume elements in half along a selected axis or axes. Also, the parallel decomposition used for the 3D FFT in the NAMD code has thus far been a slab decomposition, which limits the distribution of work for that module to N for an $N \times N \times N$ FFT. Much of the published scientific work using NAMD has been on simulations of large biomolecular systems rather than on smaller systems at very long timescales.

The intellectual point of departure for much of the work on highly scalable parallel decompositions of the

N-body problem (the real-space portion, at least) is the work of Hendrickson and Plimpton [19, 21], who demonstrated a force decomposition method for which the number of communicating partners of a single node scales like $O(\sqrt{p})$, where p is the number of nodes. Algorithms that attempted to achieve comparable theoretical scaling behavior within a volume decomposition have been published by Snir [22] and Shaw [23], although, to our knowledge, no implementation of either of these algorithms has been published. Subsequent to the publication of our initial description and performance characterization of the Blue Matter Version 4 technique in September 2005 [24, 25], which is described below, the D. E. Shaw team published a description of essentially the same technique [26]. An implementation of this algorithm on a commodity cluster was described in a subsequent publication [27], but performance was reported using aggressive approximations that included the use of single-precision arithmetic and multiple timestepping, with a reported energy drift of 7×10^{-4} K/ns (100 times larger than the worst energy drift measured on Blue Matter under production or benchmarking conditions).

Blue Matter

Blue Matter has provided a concrete context in which to explore algorithmic techniques and programming models required to exploit massively parallel machine architectures, such as the IBM Blue Gene* supercomputer, as well as providing the capability required to execute one of the primary goals of the IBM Blue Gene Project [28]: to use the unprecedented computational resource developed during the course of the project to attack grand-challenge life-sciences problems, such as advancing our understanding of biologically important processes, in particular, the mechanisms behind protein folding. Blue Matter has been used in production by computational scientists on the Blue Gene Project since 2003, initially on the IBM SP* platform and later on the Blue Gene/L platform [29–36]. Through the use of the real-space parallelization techniques, described below in the section "Parallel decompositions," and the highly scalable 3D FFT developed as part of this project [37], the Blue Matter MD code has demonstrated continued speedup through approximately one atom per node on 16,384 Blue Gene/L nodes and a time per timestep of less than 2 ms for a 43,222-atom solvated membrane protein system. All of this was achieved with methodologically rigorous methods for classical fixed-charge force fields. Our success in meeting the challenges of strong scaling has enabled detailed atomistic simulations of biologically interesting systems at timescales and with ensemble sizes that were previously unattainable, including 26 trajectories of 100 ns each of a G-protein-coupled receptor (GPCR), rhodopsin, in a realistic membrane environment [34] and multiple-microsecond-scale simulations of that system [36] and others. Furthermore, since the path to increased hardware performance now seems to lie more along the path of increasing concurrency (multiple CPU cores per chip and increased parallelism) rather than increasing clock speed [38], future work with even very large molecular systems with hundreds of thousands of atoms may require scalability to small ratios of atoms per node. While there have been some theoretical studies of scaling in this limit [39], the Blue Matter classical biomolecular simulation application running on the Blue Gene/L machine represents the first demonstration of strong scaling of such a code to this degree [15, 24, 25, 40]. With access to such timescales comes increased concern about whether the simulations are valid, and Blue Matter has also demonstrated the ability to generate trajectories with excellent energy conservation over microsecond timescales.

Inherent concurrency of MD

We describe MD as comprised of four major modules:

1. Real space, nonbonded (finite-range pair interactions).
2. K space (FFT based).
3. Bonded (graph based).
4. Integration (per particle).

Before attempting to scale an algorithm onto many thousands of nodes, it is useful to estimate how much concurrency (potential for parallelism) is inherent in various components of that algorithm. It is interesting that while the machine architecture of the Blue Gene supercomputer forced us to think this way, it is likely that this sort of analysis would be useful on other massively parallel machines. First, consider the anatomy of an MD timestep starting with the availability of the coordinates and velocities of all of the particles in the system ($\boldsymbol{r}_i$, $\boldsymbol{v}_i$):

- Compute forces on each particle due to bonded (intramolecular) interactions:
  - Bond stretches.
  - Angle bends.
  - Torsions.
- Compute forces on each particle due to nonbonded interactions (assuming periodic boundary conditions):
  - Hard-core repulsive and van der Waals forces (usually represented by a Lennard–Jones 6-12 potential function that is smoothly switched off beyond some cutoff distance R_c).

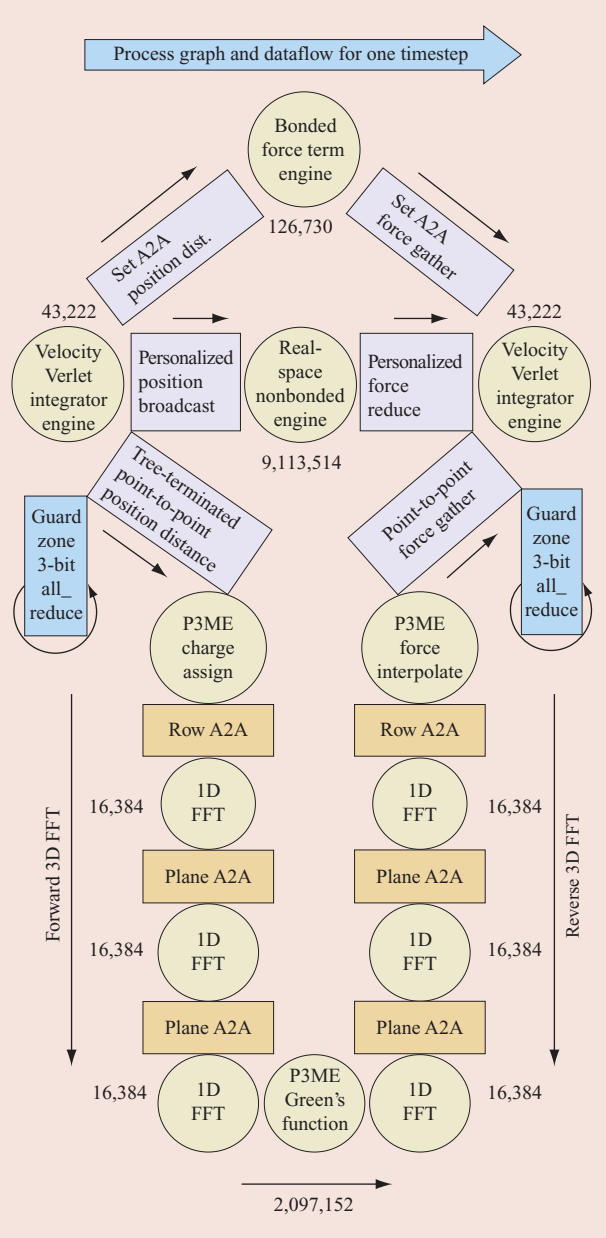


Figure 1

The data dependencies and opportunities for concurrency in various stages of the nonbonded force calculations in an MD timestep using the P3ME method. For each step, the actual number of independent work items is displayed for one of the molecular systems, rhodopsin, benchmarked in this paper. Three separate threads of computation are shown: finite-ranged pair interactions, bonded interactions, and long-range electrostatic interactions computed by the P3ME method. The center circle at the bottom of the figure represents the convolution step, which is evaluated by first Fourier transforming the meshed charge distribution, then multiplying by a kernel (Green's function), and then inverse Fourier transforming the result to obtain the meshed electrostatic potential. A detailed explanation of the P3ME method shown here can be found in [8].

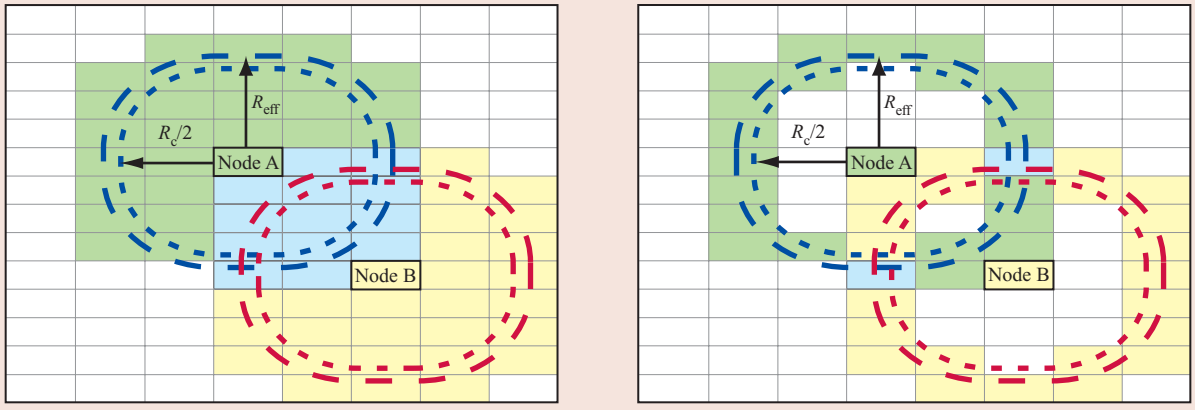
  - Electrostatic forces ($1/r^2$ forces that are most commonly evaluated using the Ewald summation technique or its mesh-based variants [9]).
- Accumulate the total force on each particle and use that force along with the current position and velocity of the particle to propagate its motion forward in time by some small increment.

It is possible to view an MD timestep (or any parallel computation) as the successive materialization of distributed data structures on which local computation takes place. Given a choice of granularity below which no parallelism will be attempted and taking into account the data dependencies in the algorithm, one can estimate the number of data-independent computations required or available at each phase. An example of such an analysis for an MD simulation using the P3ME method to treat the long-range electrostatic forces is shown in **Figure 1**. One can afford a lack of concurrency in components that impose very little computation or communication burden, but eventually even these will become bottlenecks if they are not parallelized. This is an instance of Amdahl's Law, which states that the amount of speedup possible for a fixed-size problem is limited by the fraction of the problem that is nonparallelizable [41]. More precisely, the $PotentialSpeedup = 1/(f + (1 - f)/N)$, where f is the fraction of time consumed by serial operations (or those replicated on every node) and N is the number of nodes.

In the limit of very strong scaling, the P3ME convolution step is expected to be the limiting factor, assuming that a good distribution of work can be achieved for the bonded and real-space nonbonded force computations. We conjecture that this would be the case even for an architecture with full bisectional bandwidth because of latencies due to the hardware and software overheads in the successive communication phases required by the P3ME shown in Figure 1. The development of a highly scalable 3D FFT was essential for Blue Matter and has been reported in detail previously [37].

Parallel decompositions

Our explorations of parallel decompositions have included replicated data methods that leverage the hardware facilities of the Blue Gene/L platform to globalize and reduce data structures [42, 43] and combined spatial and interaction decompositions [15, 24, 25]. This paper focuses on the two spatial decompositions used in Blue Matter for which we have strong scaling data. We refer to these two decompositions, described below, as *Version 4* (V4) and *Version 5* (V5). Our investigations began with the realization that for finite-range pair potentials, given a 3D domain decomposition of the simulation cell onto a set of nodes, it is possible to



(a)
Broadcast to nodes containing any portion of simulation space within R_{eff} of the boundary of the central node (V4).

(b)
Broadcast to nodes containing any point in simulation space that is exactly R_{eff} away from the boundary of the central node (V5 full skin before optimization).

Figure 2

These figures illustrate the basic geometric ideas behind the spatial decompositions used by Blue Matter. They represent a two- dimensional slice of the simulation space, with the domain decomposition onto nodes shown by the array of rectangular cells superimposed on the figures. Node A broadcasts positions of particles within the volume of simulation space that it manages to the nodes shaded green or blue. Node B broadcasts positions to nodes shaded yellow or blue. The coarse dashed lines are drawn at a distance R_{eff} from the central Nodes A and B. R_{eff} is somewhat larger than half the cutoff distance, R_c used for the real-space nonbonded forces to provide for a guard zone. The cells shaded blue receive positions from both Node A and Node B and can, therefore, compute forces between particles. While part (a) shows all of the possible nodes that could compute the interactions between Nodes A and B in blue, the Blue Matter V4 implementation actually assigns the interaction between two particles to the node containing the point in simulation space halfway between the two particles. Part (b) shows the starting point for the optimization process used in Blue Matter V5. The nodes shown in blue are part of the interaction option set for the node pair (A, B) whose role in the V5 optimization procedure is described in the text. (Reproduced with permission from [15]; ©2006 ACM.)

limit the broadcast of positions by any originating node to those nodes containing a portion of simulation space within half a cutoff radius of the boundary of the originating node. V4, the first spatial decomposition deployed in Blue Matter, uses geometric criteria to determine where the real-space nonbonded interaction between two particles is to be computed, namely, on whichever node contains the point halfway between the two particles. This provides a large number of distinct units of computational work that can be distributed by partitioning space using an optimal recursive bisection scheme [24, 25, 40]. The implementation of this method entails the broadcast of a particle position to a sphere with radius R_{eff}. Nominally, R_{eff} is half the MD cutoff distance R_c for real-space nonbonded interactions for both V4 and V5. However, enabling the preservation of particle assignments to nodes over several timesteps requires the introduction of a guard zone that increases the R_{eff} beyond half of the MD cutoff. The size of the guard zone is a tuning parameter. The use of this guard zone in V4 also ensures that the particle positions required by the K-space and bonded-force modules are usually available without additional pairs of communicating nodes.

The most recent parallel decomposition, V5, uses geometry primarily as a heuristic to prime the set-based optimization process that follows [15]. Whereas V4 managed data distribution (of particle positions) and reduction (of forces) by means of a data push and a caching module, V5 specializes the communication between the integrator module and the three force-computation modules described in the previous section. Whereas the V4 push and cache method required a distinct communication phase, V5 allows the overlap of the communication or computation, or both, of one module with that of another module. On the Blue Gene/L machine, which has two processors per node, modules are scheduled to cores to maximize overlap. Currently, the scheduling is static and we place the longest running module on its own core.

Achieving a reduction in the number of communicating partners per node (for the real-space nonbonded computations) was the major impetus behind the evolution from V4 to V5. In Blue Matter V4, each node (the *home* node) broadcasts the positions of the particles

Table 1 Communicating partner counts as a function of partition size for Blue Matter Version 4 (V4) volumetric broadcast, full-skin broadcast, and Version 5 (V5) set-based optimization. The V5 results give the minimum and maximum values as well as the average because the V5 algorithm result depends on the detailed distribution of particles.

Node count	*Partition*			*V4*	*V5*			
	P_x	P_y	P_z		*Full skin*	*Sparse (average)*	*Sparse (minimum)*	*Sparse (maximum)*
512	8	8	8	45	42	22	19	24
1,024	8	8	16	63	58	27	24	30
2,048	8	16	16	105	90	38	34	40
4,096	16	16	16	147	122	50	47	54
4,096	8	32	16	147	122	51	46	52
8,192	16	32	16	209	162	65	61	67
16,384	16	32	32	349	242	89	84	91

homed on that node to each node containing a volume element that intersects the volume enclosed by an imaginary shell extending half the cutoff distance (with some additional guard distance) away from the volume owned by the home node, as shown in **Figure 2(a)**. This local volumetric broadcast and the corresponding reduction of forces back to the home node involve a number of nodes that scale like $O(p)$, the same scaling obtained with a classic volumetric decomposition (though broadcasting to only one-eighth of the volume in simulation space). The advantage of a technique such as that used by V4 on a mesh interconnect topology is that the number of hops for each message is minimized.

The next incremental step in improving the V4 algorithm was taken by noting that all interactions can still be computed if the local broadcast only goes to nodes containing a portion of the imaginary shell that forms the boundary of volumetric broadcast used in the V4 technique, as shown in **Figure 2(b)**. This improves the scaling of the number of communicating partners per node to $O(p^{2/3})$. In this method, the interaction between two particles can be computed on any of the nodes that contain the intersection of the broadcast shells for the nodes containing the two particles. In contrast to the V4 technique, there is no simple geometric construction (analogous to the choice of the midpoint in V4) to select the node that should compute the interaction.

Implementation realities can affect the purely geometric analysis of the volumetric and shell broadcast techniques as well as those of other geometric approaches [22, 23]. The use of a guard zone added onto the nominal position broadcast radius enables the assignment of particles and interaction computations to nodes in order to remain fixed over multiple timesteps. This means that, for example, the ideal broadcast to nodes that intersect a spherical surface of zero thickness becomes a broadcast to nodes that intersect a spherical surface with a thickness equal to that of the guard zone.

The insight that the real-space nonbonded algorithm can be cast as an optimization problem leads to the method actually implemented in Blue Matter V5 [15]. The full optimization problem is to minimize the average execution time per MD timestep, and this is beyond our ability to solve at present. Attempting to minimize the number of communicating partners for each node, subject to the constraint of load balance, is a much more tractable problem, particularly given a good heuristic (such as the broadcast to a shell described above) for starting this optimization process. We begin a description of the optimization process used in V5 with a set of definitions and initializations:

Table 2 Details about the systems benchmarked with Blue Matter. All runs were made with the velocity Verlet integrator [14], all runs used the P3ME technique to handle long-range electrostatic interactions, and all runs were NVE simulations. Rigid water models were used, and all heavy atom-to-hydrogen bonds in nonwater molecules were constrained using Rattle [44]. All runs performed the P3ME calculation on every timestep. Except for the SOPE (64^3) data, these choices are those used in production scientific work (hairpin, rhodopsin) or attempt to match (or exceed) benchmarking conditions reported elsewhere (ApoAl) [15, 24, 40].

System	*Total atoms*	*Cutoff/ switch* (Å)	*P3ME mesh*	*Timestep* (fs)
Hairpin	5,239	9.0/1.0	64^3	1
SOPE	13,758	9.0/1.0	64^3; 128^3	1
Rhodopsin	43,222	9.0/1.0	128^3	2
ApoAl	92,224	10.0/2.0	128^3	1

1. Begin by defining the set of nodes to which each node will potentially broadcast particle coordinates. In the case of Blue Matter V5, this is the set of nodes that contains some portion of the surface in simulation space defined by the locus of points that are exactly a broadcast distance R_{eff} away from the surface of the simulation space volume assigned to the originating, or central, node i. We call this the Candidate Send To Node Set or, in the case of V5, the Surface Node Set, which we denote C_i. Of course, the optimization process could also use the set of nodes defined by the broadcast volume in V4 or some other set of nodes as a starting point.
2. For each pair of nodes i and j, define the Interaction Assignment Option Set to be the intersection of the corresponding Candidate Send To Node Sets. In principle, the task of computing the interactions between particles in i and j can be assigned to any node in the Interaction Assignment Option Set, which we will call O_{ij}.
3. The Interacting Pair Assignment structure is defined to be an upper triangular two-dimensional integer array indexed by node identifiers i and j. When the optimization procedure is finished, the array element I_{ij} will contain the node identifier of exactly one of the nodes in O_{ij}, which is where the interactions between the node pair will be computed.
4. We also define the Sparse Send To Node Set S_i for each node i. S_i is empty at the start, but at the end, S_i will be a subset of C_i.

Next, we outline the iterative procedure used to construct the Sparse Send To Set S_i:

```
{Construct Sparse Send To Node Set}

Initialize all elements of the Interacting Pair
  Assignment structure I_ij to −1
Let sequence P = {(i, j) ∈ P × P | (i < j)} where P is
  the set of node identifiers
Sort P according to the size of the corresponding
  Interaction Assignment Option Set ||O_ij|| {smallest
  first}
for k = 0 to ||P|| − 1 do
  (i, j) = P_k
  D = O_ij {Interaction Assignment Option Set}
  if ∃a ∈ P|a ∈ (D ∩ S_i ∩ S_j) then
    {No need to add any nodes to Sparse Send To Node
    Sets}
  else if ∃a ∈ P|a ∈ (D ∩ S_i) ∨ (D ∩ S_j) then
    Choose the element a that appears in the
    smallest number of Sparse Send To Node Sets S_n
    and append it to S_i and to S_j {One of these appends
    will be a no-op because (a ∈ S_i) ∨ (a ∈ S_j)
    already}
  else
    From b ∈ D choose b|b appears in the smallest
    number of Sparse Send To Sets S_m and append it to
    S_i and to S_j
  end if
end for
```

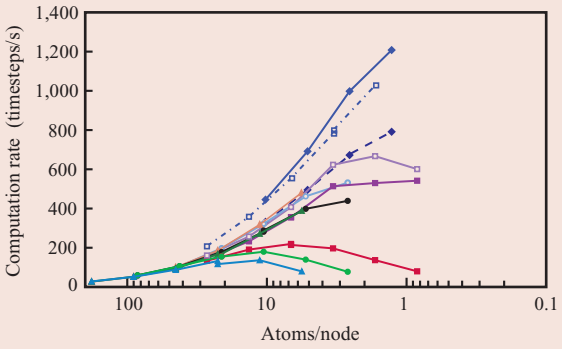


Figure 3

The computational rates on a number of molecular systems as a function of the number of atoms per node. This facilitates comparisons between systems of different sizes and explicitly shows the degree of strong scaling achieved. The results for Blue Matter using the V5 method running on the low-level communication SPI provided by the Blue Gene/L Advanced Diagnostic Environment [45] are plotted, along with results from the V4 decomposition implemented on both MPI and SPI.

The results of this optimization process in terms of the communicating partner count are shown in **Table 1**. In fact, scaling plots of the *communicating partner count* as a function of node count [15] indicate that the Blue Matter V5 algorithm equals and is sometimes better than the $O(\sqrt{p})$ scaling of the number of communicating partners achieved by the Plimpton–Hendrickson technique.

Results

Performance data

Table 2 provides information about the specific parameters used in the runs whose performance is described here. **Figure 3** shows the computational throughput in timesteps per second compared with the number of atoms per node. Plotting the data as a function

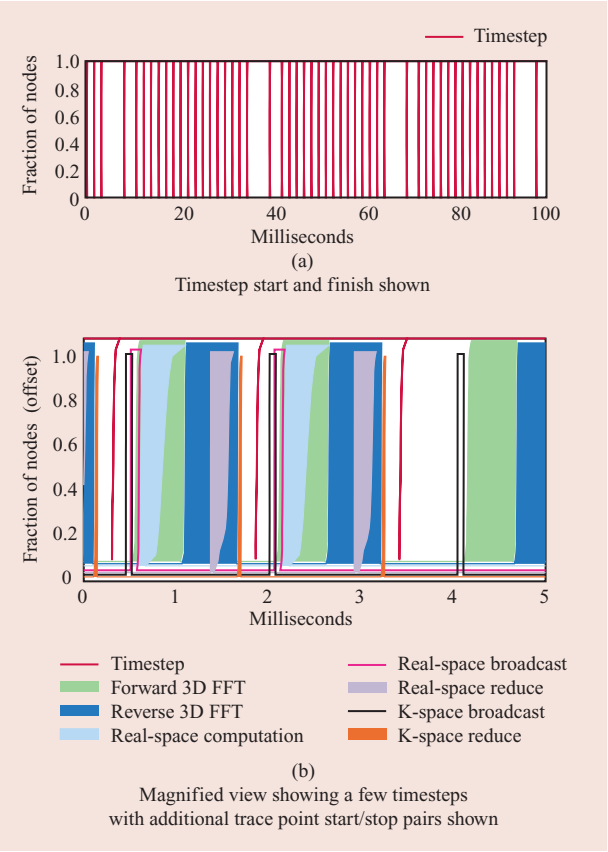


Figure 4

Timing plot in which the distribution function for the start and finish trace points, corresponding to the start and finish of a timestep on each node, is computed. For each timestep, the distribution functions $F_{\text{start}}(x)$ of the start trace points are plotted first, followed by the distributions $F_{\text{finish}}(x)$ for the finish trace points. Some of the traces are shown with fills, and the individual traces have been slightly offset in the vertical direction for clarity.

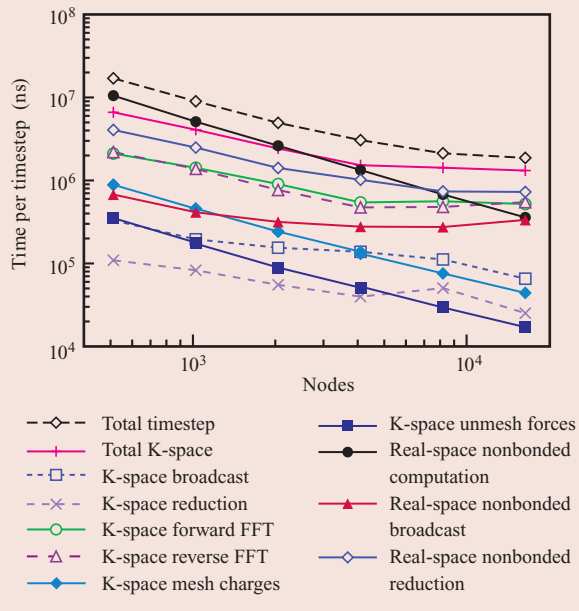


Figure 5

This figure shows the contributions of selected components to the total timestep as a function of node count for the rhodopsin system run using Blue Matter V5. Each major force engine has its own communication driver, hence the separate broadcast and reduce operations for real space and K space (as well as for the bonded forces, which are not shown). There are periods during a timestep when both cores are making use of the torus simultaneously, as can be seen in Figure 4(b). This decreases the performance of the 3D-FFT operations, but the overlapping of communication operations gives an improvement in overall performance.

of atoms per node provides some degree of normalization for system size when the real-space nonbonded interactions are the dominant contribution to the iteration time. One can observe that the scalability plots in the left-hand portion of Figure 3 (corresponding to larger values of atoms per node and lower node counts) seem to lie on a universal curve. This would correspond even more closely to a universal curve if we plotted the number of real-space nonbonded interactions per node rather than atoms per node on the horizontal axis. Figure 3 includes the Blue Matter V5 [system programming interface (SPI) only] data, as well the V4 data on both SPI and Message Passing Interface (MPI) [40].

Table 3 provides both the time per timestep and the length of time required for the neighborhood broadcast and reduce required by the V4 and V5 real-space nonbonded algorithms. **Table 4** shows a different view of V5 performance by providing the number of days required to simulate a microsecond for various molecular systems as a function of partition size. The effects of the reduction in numbers of communicating partners from V4 to V5, shown in Table 1 for rhodopsin, are manifest in the broadcast and reduction data in Table 3. The data also shows the dramatic difference in scalability between MPI and SPI for V4 that was previously reported [46]. The application-based tracing facility used by Blue Matter allows us to collect timing data from all of the nodes in the system. Trace points in this facility are placed in start–finish pairs within the source code and are turned on by compiletime macro definitions. For a single node, a timing diagram could be constructed by setting an indicator variable equal to 1 when the start-trace point is executed and resetting it to 0 after the stop-trace point is executed. In order to create something analogous to a timing diagram that contains data aggregated from all the

Table 3 Performance data for Version 4 (V4) and Version 5 (V5) algorithms. Results for two 4,096-node partition geometries are shown: $4{,}096^{r}$ is $8 \times 32 \times 16$ and $4{,}096^{c}$ is $16 \times 16 \times 16$.

β-hairpin

Node count	*Total* (ms)				*Broadcast* (ms)				*Reduce* (ms)				*Atoms per node*
	V4		*V5*		*V4*		*V5*		*V4*		*V5*		
	SPI		*SPI*		*SPI*		*SPI*		*SPI*		*SPI*		
512	3.01		2.25		0.25		0.11		0.24		0.11		10.2
1,024	2.02		1.45		0.26		0.11		0.25		0.10		5.1
2,048	1.48		1.00		0.25		0.12		0.23		0.10		2.6
$4{,}096^{r}$	1.52		1.12		0.26		0.14		0.23		0.17		1.3
$4{,}096^{c}$	1.26		0.83		0.24		0.12		0.22		0.08		1.3

SOPE

Node count	*Total* (ms)				*Broadcast* (ms)				*Reduce* (ms)				*Atoms per node*
	V4		*V5*		*V4*		*V5*		*V4*		*V5*		
	MPI	*SPI*	*SPI*	*SPI* 64^3	*MPI*	*SPI*	*SPI*	*SPI* 64^3	*MPI*	*SPI*	*SPI*	*SPI* 64^3	
512	7.47	6.81	6.22	4.83	0.56	0.35	0.16	0.13	0.44	0.35	0.19	0.16	26.9
1,024	5.25	4.30	3.89	2.79	0.63	0.31	0.14	0.10	0.53	0.30	0.16	0.12	13.4
2,048	4.66	2.81	2.45	1.80	0.88	0.25	0.12	0.09	0.86	0.23	0.15	0.10	6.7
$4{,}096^{r}$	5.61	2.57	2.11	1.28	1.38	0.25	0.14	0.09	1.33	0.24	0.13	0.10	3.4
$4{,}096^{c}$	5.08	1.95	1.60	1.25	1.40	0.22	0.12	0.08	1.31	0.21	0.15	0.08	3.4
8,192	7.31	1.89	1.50	0.97	2.49	0.23	0.14	0.08	2.21	0.21	0.11	0.08	1.7

Rhodopsin

Node count	*Total* (ms)			*Broadcast* (ms)			*Reduce* (ms)			*Atoms per node*
	V4		*V5*	*V4*		*V5*	*V4*		*V5*	
	MPI	*SPI*	*SPI*	*MPI*	*SPI*	*SPI*	*MPI*	*SPI*	*SPI*	
512	16.77	16.82	17.52	0.77	0.47	0.39	0.54	0.51	0.44	84.4
1,024	9.42	9.50	9.48	0.77	0.39	0.24	0.59	0.39	0.26	42.2
2,048	6.46	5.58	5.07	0.95	0.35	0.19	0.77	0.33	0.19	21.1
$4{,}096^{r}$	5.83	3.55	3.21	1.44	0.28	0.19	1.23	0.26	0.15	10.6
$4{,}096^{c}$	5.56	3.47	3.11	1.42	0.29	0.18	1.16	0.27	0.15	10.6
8,192	7.17	2.51	2.16	2.47	0.24	0.20	2.05	0.23	0.13	5.3
16,384	12.88	2.28	1.88	5.30	0.25	0.28	4.52	0.24	0.20	2.6

ApoAI

Node count	*Total* (ms)			*Broadcast* (ms)			*Reduce* (ms)			*Atoms per node*
	V4		*V5*	*V4*		*V5*	*V4*		*V5*	
	MPI	*SPI*	*SPI*	*MPI*	*SPI*	*SPI*	*MPI*	*SPI*	*SPI*	
512	35.56	36.37	38.42	1.05	1.08	0.66	0.68	0.97	0.90	180.1
1,024	19.29	19.18	18.95	1.02	0.74	0.40	0.71	0.76	0.51	90.1
2,048	11.48	10.68	9.97	1.14	0.60	0.26	0.88	0.55	0.30	45.0
$4{,}096^{r}$	8.57	6.33	5.39	1.66	0.51	0.22	1.54	0.47	0.23	22.5
$4{,}096^{c}$	7.55	5.97	5.44	1.37	0.50	0.19	1.23	0.48	0.21	22.5
8,192	7.34	3.68	3.14	2.22	0.43	0.15	2.15	0.38	0.16	11.3
16,384	12.58	2.57	2.09	4.83	0.40	0.13	4.80	0.34	0.13	5.6

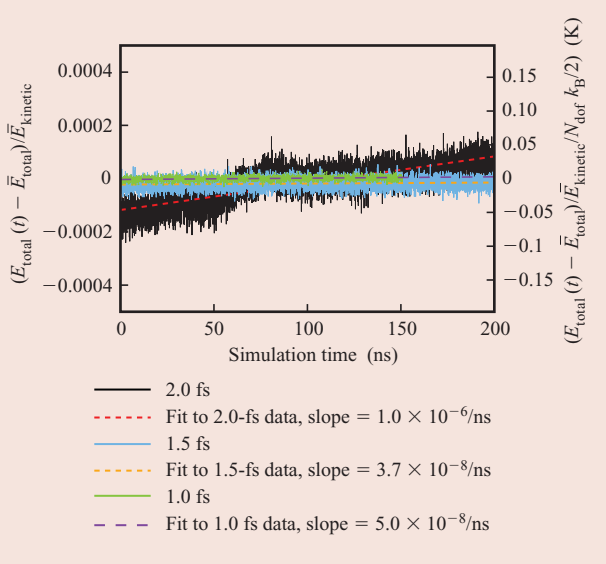


Figure 6

Energy drift for different values of Verlet integration timestep. The left axis shows the deviation from the average total energy normalized by the average kinetic energy; the right axis shows the deviation from the average total energy in equivalent temperature units (simply obtained by multiplying the left axis by the average instantaneous temperature of 394 K). The fitted slopes in terms of equivalent temperature per unit time are 2×10^{-5} K/ns, 1.5×10^{-5} K/ns, and 5.1×10^{-4} K/ns for timestep sizes of 1 fs, 1.5 fs, and 2 fs, respectively.

nodes, we compute the distribution function of timestamps (over nodes) for every timestep. The distribution function is a function of time that gives the fraction of nodes for which the application program had executed the specified trace point at time t.

This information is used to construct the plots, shown in **Figure 4**, that present the aggregated data from all 16,384 nodes for Blue Matter V5 running the rhodopsin system on the Blue Gene/L computer. In Figure 4(a), only the distribution functions of the start–finish trace-point pairs that bracket the entire timestep are shown. The periodic long timestep is caused by the need to migrate particles from one node to another as their positions shift. We avoid migration on every timestep by keeping track of particles that are somewhat farther away than half the cutoff distance (the additional amount is the *guard zone* mentioned earlier in the introduction to the section "Parallel decompositions") and also by monitoring the drift of each particle since the last migration. When one or more particles drift far enough, an alarm is raised using the fast short vector reductions available on the Blue Gene/L system, and a new migration phase follows. In Figure 1, these are the guard zone 3-bit `all_reduce` operations shown in the rectangle just below the circle containing the Velocity Verlet integrator engine. Figure 4(b) zooms in on a few

Table 4 The performance results for Blue Matter Version 5 expressed as the number of days required to simulate 1 μs for a specified timestep size. Results for two 4,096-node partition geometries are shown: $4{,}096^r$ is $8 \times 32 \times 16$ and $4{,}096^c$ is $16 \times 16 \times 16$.

	β-Hairpin	*SOPE* 64^3	*SOPE* 128^3	*Rhodopsin*	*ApoA*
Timestep (fs)	1.0	1.0	1.0	2.0	1.0
Partition size	*Days required to reach 1 μs*				
512	26.0	55.9	72.0	101.4	444.7
1,024	16.8	32.4	45.0	54.9	219.3
2,048	11.6	20.8	28.4	29.3	115.4
$4{,}096^r$	13.0	14.9	24.4	18.6	63.0
$4{,}096^c$	9.6	14.6	18.5	18.0	62.4
8,192		11.2	17.4	12.5	36.3
16,384			19.2	10.9	24.2

timesteps and shows a plot of the distribution functions for a number of important components of a timestep. It is obvious from this plot that the forward and reverse 3D-FFT operations dominate the total timestep. Also, the scheduling of communication and computation tasks on both CPUs on each node can also be seen, e.g., the overlap of the reverse 3D FFT and the real-space reduce.

A detailed breakdown of the time required for the various operations required to complete a timestep is provided for the rhodopsin system in **Figure 5**. It can be seen from the figure that with more than 2,048 nodes, the dominant contribution to the timestep switches from the real-space nonbonded computation to the total K space. At the highest node count, it is also evident that the time required for the various localized broadcasts and reductions increases, and even if the total K-space contribution could be decreased, the reductions, in particular, would become the limiting factors for scalability. The total time per timestep is not simply the sum of the various contributions shown in Figure 5 for two reasons: First, both cores are used on each node, and so, for example, the real-space nonbonded computation can overlap with portions of the total K-space work as seen in Figure 4(b), and second, some of the quantities plotted are already aggregations of other quantities in the plot, e.g., the various K-space contributions and total K space.

Energy drift in long NVE simulations

In general, in order to maximize throughput, a computational scientist wants to use the largest timestep possible consistent with providing results that adequately approximate an ideal simulation of the system (potential surface) under study. Other performance-critical

simulation parameters affecting simulation accuracy and stability include the FFT mesh spacing for P3ME methods [47] and the force-splitting scheme and timestep ratios chosen for symplectic multiple timestepping methods [12, 13, 48].

It is a considerable challenge to determine the optimal parameters for simulations that involve billions or tens of billions of timesteps and require machines running at multiteraflops or faster. **Figure 6** shows the change in the total energy in a simulation of a 66,728-atom solvated protein system, the lambda repressor, for a series of timestep sizes. Previously, we reported a measured energy drift of 6×10^{-4} K/ns (over a 1.6-μs simulation) for the 43,222-atom rhodopsin system using a 2-fs timestep [15, 46], which is consistent with the drift of 5.1×10^{-4} K/ns seen in the 2-fs timestep data for the lambda repressor.

Summary and conclusions

We have described the progression of parallel decompositions for the real-space nonbonded forces explored as part of the Blue Matter effort. The progression started with nongeometric replicated data decompositions, continued with the V4 spatial and interaction hybrid decomposition with geometric assignment of workload, and culminated in the V5 decomposition that uses geometry only as a heuristic that defines the starting point of the set-based optimization procedure for interaction assignment. As implemented in Blue Matter, these algorithms have made methodologically rigorous simulations of biologically interesting systems on the microsecond-scale routine on the Blue Gene Watson supercomputer. Furthermore, we have demonstrated that excellent energy conservation over microsecond-scale NVE simulations of solvated protein and membrane–protein systems can be achieved. The quality and time to solution that has been demonstrated by Blue Matter running on the Blue Gene/L platform enables increased ability to make contact with experiment by accessing timescales that are also accessible to physical experiment and it has already had significant scientific impact [31, 34–36].

Acknowledgments

We thank the members of the Blue Gene hardware team, including P. Heidelberger, A. Gara, M. Blumrich, and J. Sexton, as well as others who have made use of and contributed to the Blue Matter code over the years, particularly Y. Zhestkov, Y. Sham, F. Suits, A. Grossfield, and R. Zhou.

References

1. M. Karplus and J. A. McCammon, "Molecular Dynamics Simulations of Biomolecules," *Nat. Struct. Biol.* **9**, No. 9, 646–652 (2002).
2. D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications*, Academic Press, San Diego, CA, 1996.
3. Y. Sugita and Y. Okamoto, "Replica-Exchange Molecular Dynamics Method for Protein Folding," *Chem. Phys. Lett.* **314**, 141–151 (1999).
4. M. Eleftheriou, A. Rayshubskiy, J. W. Pitera, B. G. Fitch, R. Zhou, and R. S. Germain, "Parallel Implementation of the Replica Exchange Molecular Dynamics Algorithm on Blue Gene/L," *Proceedings of the 5th IEEE International Workshop on High Performance Computational Biology*, Rhodes Island, Greece, 2006; see *http://www.hicomb.org/HiCOMB2006/papers/HICOMB2006-08.pdf*.
5. J. S. Bader and D. Chandler, "Computer Simulation Study of the Mean Forces Between Ferrous and Ferric Ions in Water," *J. Phys. Chem.* **96**, No. 15, 6423–6427 (1992).
6. R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, Taylor & Francis Group, New York, 1988.
7. T. Darden, D. York, and L. Pedersen, "Particle Mesh Ewald: An N-log(N) Method for Ewald Sums in Large Systems," *J. Chem. Phys.* **98**, No. 12, 10089–10092 (1993).
8. U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen, "A Smooth Particle Mesh Ewald Method," *J. Chem. Phys.* **103**, No. 19, 8577–8593 (1995).
9. M. Deserno and C. Holm, "How to Mesh Up Ewald Sums. I. A Theoretical and Numerical Comparison of Various Particle Mesh Routines," *J. Chem. Phys.* **109**, No. 18, 7678–7693 (1998).
10. A. F. Voter, "Parallel Replica Method for Dynamics of Infrequent Events," *Phys. Rev. B* **57**, No. 22, R13985–R13988 (1998).
11. M. R. Shirts and V. S. Pande, "Mathematical Analysis of Coupled Parallel Simulations," *Phys. Rev. Lett.* **86**, No. 22, 4983–4987 (2001).
12. M. Tuckerman, B. J. Berne, and G. J. Martyna, "Reversible Multiple Time Scale Molecular Dynamics," *J. Chem. Phys.* **97**, No. 3, 1990–2001 (1992).
13. R. Zhou, E. Harder, H. Xu, and B. J. Berne, "Efficient Multiple Time Step Method for Use with Ewald and Particle Mesh Ewald for Large Biomolecular Systems," *J. Chem. Phys.* **115**, No. 5, 2348–2358 (2001).
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**, No. 1, 637–649 (1982).
15. B. G. Fitch, A. Rayshubskiy, M. Eleftheriou, T. J. C. Ward, M. Giampapa, M. C. Pitman, and R. S. Germain, "Blue Matter: Approaching the Limits of Concurrency for Classical Molecular Dynamics," *Proceeding of the ACM/IEEE Conference on Supercomputing*, Tampa, FL, 2006; see *http://sc06.supercomputing.org/schedule/pdf/pap246.pdf*.
16. A. Gara, M. A. Blumrich, D. Chen, G. L.-T. Chiu, P. Coteus, M. E. Giampapa, R. A. Haring, et al., "Overview of the Blue Gene/L System Architecture," *IBM J. Res. & Dev.* **49**, No. 2/3, 195–212 (2005).
17. 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. Comp. Phys.* **151**, No. 1, 283–312 (1999).
18. J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kale, "NAMD: Biomolecular Simulation on Thousands of Processors," *Proceedings of the ACM/IEEE Conference on Supercomputing*, New York, NY, 2002, p. 36.
19. S. Plimpton, "Fast Parallel Algorithms for Short-Range Molecular Dynamics," *J. Comp. Phys.* **117**, No. 1, 1–19 (1995).

20. S. Plimpton and B. Hendrickson, "A New Parallel Method for Molecular Dynamics Simulation of Macromolecular Systems," *J. Comp. Chem.* **17**, No. 3, 326–337 (1996).
21. B. Hendrickson and S. Plimpton, "Parallel Many-Body Simulations Without All-to-All Communication," *J. Parallel Distrib. Comp.* **27**, No. l, 15–25 (1995).
22. M. Snir, "A Note on N-Body Computations with Cutoffs," *Theory of Computing Systems* **37**, No. 2, 295–318 (2004).
23. D. E. Shaw, "A Fast, Scalable Method for the Parallel Evaluation of Distance-Limited Pairwise Particle Interactions," *J. Comp. Chem.* **26**, No. 13, 1318–1328 (2005).
24. B. G. Fitch, A. Rayshubskiy, M. Eleftheriou, T. J. C. Ward, M. Giampapa, Y. Zhestkov, M. C. Pitman, et al., "Blue Matter: Strong Scaling of Molecular Dynamics on Blue Gene/L," *Research Report RC-23688*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, August 2005.
25. R. S. Germain, B. Fitch, A. Rayshubskiy, M. Eleftheriou, M. C. Pitman, F. Suits, M. Giampapa, and T. J. C. Ward, "Blue Matter on Blue Gene/L: Massively Parallel Computation for Biomolecular Simulation," *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, Jersey City, NJ, 2005, pp. 207–212.
26. K. J. Bowers, R. O. Dror, and D. E. Shaw, "The Midpoint Method for Parallelization of Particle Simulations," *J. Chem. Phys.* **124**, No. 18, 184109–184109-11 (2006).
27. K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, et al., "Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters," *Proceedings of the ACM/IEEE Conference, on Supercomputing*, Tampa, FL, 2006; see *http://sc06.supercomputing.org/schedule/pdf/pap259.pdf*.
28. F. Allen, G. Almasi, W. Andreoni, D. Beece, B. J. Berne, A. Bright, J. Brunheroto, et al., "Blue Gene: A Vision for Protein Science Using a Petaflop Supercomputer," *IBM Syst. J.* **40**, No. 2, 310–327 (2001).
29. W. C. Swope, J. W. Pitera, F. Suits, M. Pitman, M. Eleftheriou, B. G. Fitch, R. S. Germain, et al., "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).
30. M. C. Pitman, F. Suits, A. D. MacKerell, Jr., and S. E. Feller, "Molecular-Level Organization of Saturated and Polyunsaturated Fatty Acids in a Phosphatidylcholine Bilayer Containing Cholesterol," *Biochemistry* **43**, No. 49, 15318–15328 (2004).
31. M. C. Pitman, A. Grossfield, F. Suits, and S.E. Feller, "Role of Cholesterol and Polyunsaturated Chains in Lipid-Protein Interactions: Molecular Dynamics Simulation of Rhodopsin in a Realistic Membrane Environment," *J. Am. Chem. Soc.* **127**, No. 13, 4576–4577 (2005).
32. M. C. Pitman, F. Suits, K. Gawrisch, and S. E. Feller, "Molecular Dynamics Investigation of Dynamical Properties of Phosphatidylethanolamine Lipid Bilayers," *J. Chem. Phys.* **122**, No. 24, 244715–244715-10 (2005).
33. F. Suits, M. C. Pitman, and S. E. Feller, "Molecular Dynamics Investigation of the Structural Properties of Phosphatidylethanolamine Lipid Bilayers," *J. Chem. Phys.* **122**, No. 24, 244714–244714-9 (2005).
34. A. Grossfield, S. E. Feller, and M. C. Pitman, "A Role for Direct Interactions in the Modulation of Rhodopsin by ω-3 Polyunsaturated Lipids," *Proc. Nad. Acad. Sci.* **103**, No. 13, 4888–4893 (2006).
35. M. Eleftheriou, R. S. Germain, A. K. Royyuru, and R. Zhou, "Thermal Denaturing of Mutant Lysozyme with both the OPLSAA and the CHARMM Force Fields," *J. Am. Chem. Soc.* **128**, No. 41, 13388–13395 (2006).
36. K. Martinez-Mayorga, M. C. Pitman, A. Grossfield, S. E. Feller, and M. F. Brown, "Retinal Counterion Switch Mechanism in Vision Evaluated by Molecular Simulations," *J. Am. Chem. Soc.* **128**, No. 51, 16502–16503 (2006).
37. M. Eleftheriou, B. Fitch, A. Rayshubskiy, T. J. C. Ward, and R. Germain, "Performance Measurements of the 3D FFT on the Blue Gene/L Supercomputer," *Proceedings of Euro-Par 2005*, Lisbon, Portugal, 2005, pp. 795–803.
38. D. Greer, "Industry Trends: Chip Makers Turn to Multicore Processors," *IEEE Computer* **38**, No. 5, 11–13 (2005).
39. V. E. Taylor, R. L. Stevens, and K. E. Arnold, "Parallel Molecular Dynamics: Implications for Massively Parallel Machines," *J. Parallel Distrib. Comput.* **45**, No. 2, 166–175 (1997).
40. B. G. Fitch, A. Rayshubskiy, M. Eleftheriou, T. J. C. Ward, M. Giampapa, Y. Zhestkov, M. C. Pitman, et al., "Blue Matter: Strong Scaling of Molecular Dynamics on Blue Gene/L," *Proceedings of the International Conference on Computational Science*, Reading, England, 2006, pp. 846–854.
41. G. M. Amdahl, "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities," *Proceedings of the AFIPS Conference*, Anaheim, CA, 1967, pp. 483–485.
42. B. G. Fitch, R. S. Germain, M. Mendell, J. Pitera, M. Pitman, A. Rayshubskiy, Y. Sham, et al., "Blue Matter, an Application Framework for Molecular Simulation on Blue Gene," *J. Parallel Distrib. Comput.* **63**, No. 7/8, 759–773 (2003).
43. 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, 447–456 (2005).
44. H. C. Andersen, "Rattle: A 'Velocity' Version of the SHAKE Algorithm for Molecular Dynamics Calculations," *J. Comp. Phys.* **52**, No. 1, 24–34 (1983).
45. M. E. Giampapa, R. Bellofatto, M. A. Blumrich, D. Chen, M. B. Dombrowa, A. Gara, R. A. Haring, et al., "Blue Gene/L Advanced Diagnostics Environment," *IBM J. Res. & Dev.* **49**, No. 2/3, 319–332 (2005).
46. B. G. Fitch, A. Rayshubskiy, M. Eleftheriou, T. J. C. Ward, M. Giampapa, M. C. Pitman, and R. S. Germain, "Progress in Scaling Biomolecular Simulations to Petaflop Scale Platforms," *Proceedings of the International Euro-Par Workshops*, Dresden, Germany, 2006, pp. 279–288.
47. 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, 7694–7701 (1998).
48. J. C. Sexton and D. H. Weingarten, "Hamiltonian Evolution for the Hybrid Monte Carlo Algorithm," *Nuclear Phys. B* **380**, No. 3, 665–677 (1992).

Received February 13, 2007; accepted for publication April 2, 2007; Internet publication December 19, 2007

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 is a Senior Software Engineer. He joined the IBM Research Division in 1985 as a student. He received a B.S. degree in computer science from Antioch College in 1987 and remained at IBM to pursue interests in parallel systems. Mr. Fitch's research interests are in 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 system 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.

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 researcher at the IBM Thomas J. Watson Research Center. She holds an M.S. degree in engineering and a Ph.D. degree in theoretical and computational chemistry, both from Brown University. She has worked the past few years mainly on the Blue Gene Project. In particular, she has made contributions to the design and implementation of parallel algorithms, parallel applications, and parallel programming models, and studied the performance of parallel scientific applications for the Blue Gene/L architecture. Another area of Dr. Eleftheriou's interest is large-scale simulations addressing questions in the field of biology, particularly in the area of protein folding.

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.

Mark E. Giampapa *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (giampapa@us.ibm.com)*. Mr. Giampapa is a Senior Engineer in the Exploratory Server Systems Department. He received a B.A. degree in computer science from Columbia University. He joined the IBM Research Division in 1984 to work in the areas of parallel and distributed processing, and he has focused his research on distributed memory and shared memory parallel architectures and operating systems. Mr. Giampapa has received three IBM Outstanding Technical Achievement Awards for his work in distributed processing, simulation, and parallel operating systems. He holds 15 patents, with several more pending, and has published ten papers.

Michael C. Pitman *IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (pitman@watson.ibm.com)*. Dr. Pitman received his Ph.D. degree in chemistry in 1995 from the University of California at Santa Cruz. He joined the Biomolecular Dynamics and Scalable Modeling Group within the Computational Biology Center at the IBM Thomas J. Watson Research Center and continued work in the area of computational drug design methods. He began a leading role in the Blue Gene Protein Science program in 2001, focusing on large-scale membrane and membrane protein simulation. His research interests are focused on understanding the nature of protein–membrane interactions. Dr. Pitman conducts large-scale all-atom simulations of membrane proteins in explicit, biologically relevant environments.

Jed W. Pitera *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (pitera@us.ibm.com)*. Dr. Pitera is a Research Staff Member in the Science and Technology Department at the IBM Almaden Research Center. His research focuses on the use of computer simulation to address questions in biology and chemistry, particularly in the areas of protein folding, molecular recognition, and self-assembly. He received undergraduate training in biology and chemistry at the California Institute of Technology. He subsequently pursued graduate studies in biophysics at the University of California at San Francisco. He pursued similar work in a postdoctoral position at the Swiss Federal Institute of Technology Zurich, where his research focused on novel methods of calculating free energies for ligand design. He has been a member of the IBM Blue Gene Project Science and Application team since February of 2001. Dr. Pitera is also an adjunct assistant professor in the UCSF Department of Pharmaceutical Chemistry.

William C. Swope *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (swope@almaden.ibm.com)*. Dr. Swope has been engaged with the IBM Blue Gene Protein Science Project since 2000, with strong emphasis on biomolecular simulation methodology and the development of practical techniques to simulate protein folding kinetics and thermodynamics. He received his undergraduate degree in chemistry and physics from Harvard University and his Ph.D. degree in quantum chemistry from the University of California at Berkeley. He performed postdoctoral research on the statistical mechanics of solvation and condensed phases in the chemistry department at Stanford University. Dr. Swope maintains a number of scientific relationships and collaborations with academic and commercial scientists involved in the life sciences, specifically related to drug development.

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. He holds an A.B. degree in

physics from Princeton University and M.S. and Ph.D. degrees in physics from Cornell University. Since 2000, Dr. Germain has been responsible for the science and associated application portions of the Blue Gene Project. 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, the ACM, and the American Physical Society.