Blue Matter on Blue Gene/L: Massively Parallel Computation for Biomolecular Simulation

Robert S. Germain
rgermain@us.ibm.com

Blake Fitch
bgf@us.ibm.com

Aleksandr Rayshubskiy
arayshu@us.ibm.com

Maria Eleftheriou
mariae@us.ibm.com

Michael C. Pitman
pitman@us.ibm.com

Frank Suits
suits@us.ibm.com

Mark Giampapa
giampapa@us.ibm.com

IBM Thomas J. Watson Research Center
1101 Kitchawan Road/Route 134
Yorktown Heights, NY 10598 1pc

T.J. Christopher Ward
IBM Hursley Park
Hursley, UK SO212JN
tjcw@uk.ibm.com

ABSTRACT

This paper provides an overview of the Blue Matter application development effort within the Blue Gene project that supports our scientific simulation efforts in the areas of protein folding and membrane-protein systems. The design philosophy of the Blue Gene/L architecture relies on large numbers of power efficient nodes (whose technology is derived from the world of embedded microprocessors) to enable packing of many such nodes into a small volume to achieve high performance. In order for an application to exploit the potential of this architecture, the application must scale well to large node counts. Because the scientific goals of the project entail simulating very long time-scales, up to microseconds, strong scaling of a fixed size problem to these large node counts is a requirement. In pursuit of this objective we have considered a variety of parallel decompositions and explored ways to exploit and map algorithms onto the two primary high performance interconnects provided by the Blue Gene architecture, the 3D-torus network and the collective network. Our current version of the application continues to speed up through 4096 nodes and is being used for studies of a protein/lipid system (for which some results have already been published) and for protein folding/unfolding simulations.

Categories and Subject Descriptors

J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology and Genetics*; D.1.3 [**Programming Techniques**]: Concurrent Programming—*Parallel Programming*

General Terms

Algorithms, Performance

Keywords

Parallel Programming, N-body Simulations, Biomolecular Simulation, Molecular Dynamics

1. INTRODUCTION

From its inception at the end of 1999, the Blue Gene project has had an application effort to support its scientific goal[1] of using large scale simulation to improve our understanding of protein folding mechanisms and other biologically important phenomena. That application effort has always looked for ways to exploit features of the machine architecture as part of the effort to improve performance and scalability.

The scientific goals of the Blue Gene project require biomolecular simulations of modestly sized systems (10,000–100,000 atoms) for long time scales (hundreds of nanoseconds to microseconds) and because the philosophy of the Blue Gene/L hardware design has been to use massive numbers of power efficient CPUs to achieve high performance, the Blue Matter[9] application effort is required to be able to demonstrate strong scaling of fixed size problems to large node counts. Results obtained from the exploitation of prototype Blue Gene/L hardware for production scientific use in the second half of 2004 were recently published[16] and additional work is being initiated as more hardware becomes available.

2. BLUE GENE/L SYSTEM OVERVIEW

Blue Gene/L[10] is a massively parallel supercomputer developed at the IBM T.J. Watson Research Center in collaboration with Lawrence Livermore National Laboratory. In contrast to other current massively parallel supercomputers, the building block of BG/L is not derived from the fastest (and highest power density) technology available, but rather from the world of embedded microprocessors. Except for off-chip double data rate (DDR) memory and link chips that enable the partitioning of the machine, all of the functionality for a BG/L node is contained on a single ASIC chip shown

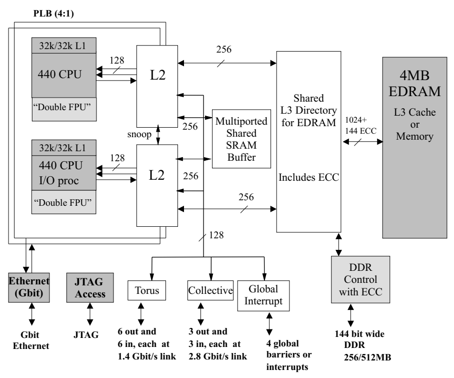


Figure 1: Schematic view of the ASIC that forms the building block for the Blue Gene/L system. Off-chip connections, including those to the DDR memory, torus network, and collective network are shown at the bottom of the figure.

schematically in Figure 1. This chip operates at a clock frequency of 700 MHz and has two PowerPC 440 CPUs with associated dual floating point units. An on-chip L3 cache comprising 4MB of DRAM is typically only 30 cycles away from registers on L1/L2 cache misses. All of the circuitry for the communications networks is also on this chip and the companion link chip. The core BG/L machine is comprised of two classes of nodes, compute nodes and I/O nodes. The I/O nodes are not counted when sizing the compute capacity of BG/L; they only act as the gateway for communication to the external world and the ratio of I/O nodes to compute nodes can be varied.

BG/L has five networks, two of which are of particular interest to the application developers: the torus network and the collective network. The three-dimensional torus has links between each node and its six neighbors while the collective network enables low latency broadcast and reduction operations as well as providing the path for I/O to external devices. Applications communicate with the outside world via I/O nodes; these are additional BG/L nodes that are connected to the collective network (but not the torus network) and also have a Gigabit Ethernet connection to the outside world.

Two modes of operation are supported by the system software[14]: (1) "coprocessor mode" which runs a single MPI task on each compute node, and "virtual node mode" which runs two MPI-tasks on each compute node. The compute nodes run a specialized Compute Node Kernel (CNK) that supports a subset of posix calls and only allows a single thread of execution on each CPU. The I/O nodes run a port of PowerPC Linux.

3. BIOMOLECULAR SIMULATION

Biomolecular simulation can be used to obtain insights into microscopic phenomena that may be unavailable from physical experiments[12, 1]. However, like all simulations, their usefulness depends on the validity of the models used and careful validation of observables computed from simulations against experimental data is essential.

One of the most commonly practiced forms of biomolecular simulation is molecular dynamics[2]. Molecular dynamics is an instance of n-body simulation in which the dynamical evolution of a system of particles is computed by successively computing the forces on each particle, numerically integrating the classical equations of motion to obtain updated positions and velocities for each particle, and then repeating the cycle by recomputing the forces on each particle based on the updated positions. The models used to compute the forces on each particle are known as "force fields" in the context of biomolecular simulation.

The classes of interactions typically included in a force field are enumerated in Table 3. These include forces between covalently bonded particles such as bond stretches between pairs of atoms, angle bends defined by a triple of particles, and torsions defined by a quartet of particles. In addition, non-bonded forces such as electrostatics and van der Waals (typically modeled, along with hard core repulsion, by a Lennard-Jones 6-12 potential) are also included. Current practice in biomolecular simulation involves the use of periodic boundary conditions. The bonded and Lennard-Jones interactions are weak enough at large distances that they are just set to zero beyond some cut-off distance, often using a smooth cut-off function and the periodic boundary conditions can be treated using a minimum image convention in which for each particle in the central cell, the force will be computed with the nearest image (either in the central cell or in one of the 26 adjoining cells) of every other particle within the cut-off distance. However, the same finite range cut-off treatment applied to the long-range electrostatic forces can lead to unphysical behavior[4] and requires the use of a different technique.

The potential energy of a system of point charges (where the system is electrically neutral) with periodic boundary conditions can be written as

$$\sum_{\{\mathbf{n}\}}^{\prime} \sum_{i,j}^{N} \frac{q_i \, q_j}{|\mathbf{r_{ij}} + \mathbf{n}|} \tag{1}$$

where q_i is the charge on the ith particle and $\mathbf{n}$ belongs to the set of lattice vectors $\{l\,\mathbf{u} + m\,\mathbf{v} + n\,\mathbf{w}\}$ where l,m, and n are integers and $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are the lattice basis vectors. The $\prime$ on the first sum indicates that when $\mathbf{n} = 0$, the terms in the double summation with $i = j$ should be excluded. This sum is only conditionally convergent; that is, the value of the sum depends on the summation order. The Ewald summation technique[2] and related techniques rewrite the potential sum as a pair of summations; one in real space consisting of a summation of "screened" electrostatic potential terms that is unconditionally convergent and the second, a sum that is expressed as a sum of Fourier transformed terms (this is often called the "reciprocal space" sum). The most commonly practiced techniques for handling electrostatics with periodic boundary conditions, particle-mesh methods such as Particle-Particle-Particle Mesh Ewald (P3ME) technique[6], approximate the actual charge distribution by a set of weights on a uniformly spaced mesh. This allows use of the Fast Fourier Transform (FFT) in carrying out the necessary computation which is a convolution of the charge distribution with a kernel chosen to give good accuracy with the uniform mesh approximation used for the charge distribution.

4. COMPUTATIONAL CHALLENGES

In order to support the goal of using simulation to access longer time scales routinely, the Blue Matter application has to provide good scalability to thousands of nodes for fixed size problems. Key challenges exist in the areas of load balancing and global data dependencies. Load balancing challenges arise in the context of computing finite-ranged non-bond real-space interactions. Global data

Force Class	Description and Examples	Communication
Bonded	Forces between covalently bonded molecules–includes bond stretches, angle bends, and torsions.	Limited range within a covalently bonded molecule (typically within a graph distance of four)
Real Space Non-bond	Lennard-Jones: $U_{L-J}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$, cut-off electrostatics, real space interaction portion of Ewald and related techniques	Limited range in simulation space
Reciprocal Space	Fourier space portion of long range electrostatic forces treated by the Ewald summation technique or related Particle-Particle-Particle-Mesh (P3ME) technique	global communication required for computation of 3D-FFTs

Table 1: This table summarizes the different classes of forces that must be computed in molecular dynamics, provides some examples in each class, and specifies the communication or data dependency characteristics of each class. The bonded and real-space non-bond forces have data dependencies that are local in simulation space and hence can take advantage of an appropriate domain decomposition to achieve locality of communication. The computation of the reciprocal or Fourier space contribution to the force on a particle requires knowledge about the position of every particle in the system. This global data dependency comes about through the computation of the convolution of the charge distribution with the kernel required to solve for the effects of the long range electrostatic forces in a system with periodic boundary conditions. In the case of the Particle-Particle-Particle-Mesh (P3ME) technique used in this work, the convolution is computed by first approximating the actual charge distribution by weights on a regular mesh, then computing the convolution through use of a forward and inverse FFT.

dependencies manifest themselves through the 3D-FFTs used in carrying out the convolution required by the P3ME technique.

To address load balancing, we have been carrying out explorations of different parallel decompositions, progressing from the straightforward to the more complex. The first decomposition investigated was an example of a "replicated data" decomposition[17] in which we used the fast hardware collective network on BG/L to globalize the positions of all the particles in the system and then used either the collective network or the torus network to perform a global floating point reduction of all the forces on each particle. In effect we had two global arrays, one of positions and one of forces, which allowed us to distribute the computation of all interactions arbitrarily across the entire machine. This made load balancing very simple, but our scalability was limited to 1024 nodes by the performance of the global floating point reduction. For the position globalization, we were able realize 90% of the hardware bandwidth in the collective network via an optimized MPI collective developed for Blue Gene/L[3].

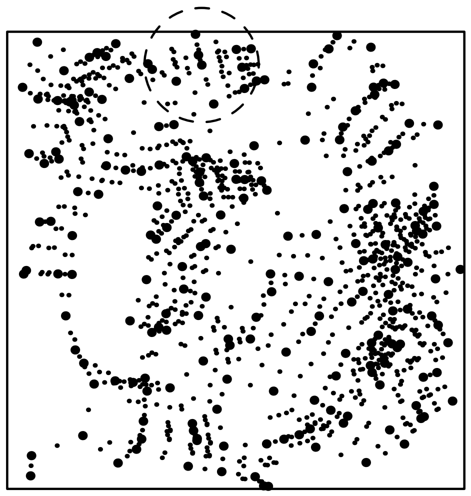
While communications issues determine the ultimate scalability of the application, total time to solution for many configurations of machine partition and system size is strongly influenced by floating point performance. The dual floating point unit on the Blue Gene ASIC chip allows the issue of SIMD-style instructions on appropriately aligned pairs of double precision numbers[18]. In library routines such as a highly optimized serial FFT implementation[13] and selected linear algebra kernels[5], very high efficiencies have been achieved using compiler intrinsic functions. On selected kernels of special interest to molecular dynamics, good results have also been obtained via C++ source code restructuring[8]. These kernels include vectors of reciprocal square roots and vectors of reciprocals as well as vectors of special functions. Although the dual floating point unit can be viewed as a two element vector unit, the five cycle latency in the floating point pipeline actually makes it advantageous to have ten data independent streams of computation in play. The compiler challenges involved in identifying these streams of computation and efficiently scheduling instructions to keep data resident in the floating point registers as well as addressing the alignment requirements imposed by the dual floating point unit in generic numerical codes continue to a topic of research and development activity.

5. CURRENT WORK AND RESULTS

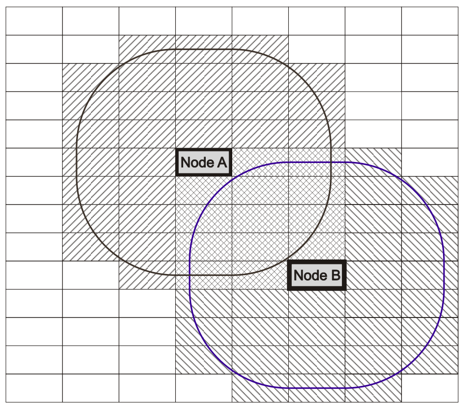
Our current exploration involves a variant of a "volume" decomposition in which the three dimensional simulation domain is divided into volume elements that are mapped directly onto BG/L nodes so that locality in simulation space is equivalent to locality on BG/L's 3D-torus network. In this case we achieve load balancing for the real-space interactions by first assigning the interaction between two atoms (or groups of atoms) to be carried out on the node containing the point (in simulation space) that lies midway between the two interacting atoms (or groups) as shown in Figure 2 and associating a cost with that position. Then we carry out successive orthogonal bisections of simulation space in such a way as to give equal loads in each section[15]. This process of orthogonal recursive bisection (ORB) is carried out until there are enough sections so that we can associate each volume section with a BG/L node and locality is preserved. The position broadcast and force reduction can now be local operations on the torus network since positions only need to be broadcast to nodes containing portions of simulation space within half the cutoff distance of the originating atom or atoms. Of course the quality of the load balance will decrease with time, but measurements indicate that the degradation is slow enough to allow the simulation to proceed for many time steps before a regeneration of the mapping is necessary.

Although we are running in co-processor mode, we can make use of both cores to allow a limited amount of overlap between communication and computation. Because the real-space non-bond computations and the reciprocal space operations can proceed independently, we overlap them by off-loading the real-space non-bond operations (which are pure computation) to the second CPU while carrying out the communications-intensive reciprocal-space operations on the first CPU. This "dual core" mode of operation gives significant improvements when the real-space and reciprocal-space operations take comparable amounts of time.

Figure 3 shows the scalability of the major components of a time-step for a system with 43,222 atoms comprising a protein, Rhodopsin,



(a) Interaction Centers



(b) Spatial Decomposition

Figure 2: Figure (a) illustrates the relationship of the particle positions (large dots) and the interaction centers (small dots) in simulation space. The interaction centers are placed at the mid-point between each pair of particles that fall within the cut-off radius. A dashed circle with radius equal to the cut-off radius chosen is drawn around one of the particles. Static or "structural" load balancing is carried out by using optimal recursive bisection to partition the simulation volume into sub-volumes that contain approximately equal computational burdens. The computational burden of a sub-volume is computed by summing the computational burden of each interaction center contained within that sub-volume. Figure (b) gives a view of the spatial decomposition showing the broadcast zones for two nodes superimposed on the spatial decomposition of the domain onto all nodes (two-dimensional view for simplicity). The nodes that contain areas of simulation space within R_b of the volume element assigned to Node A are shown with one kind of hatching while those nodes that contain areas of simulation space within R_b of the volume element assigned to Node B are shown with another type of hatching. The region of overlap between these two areas is shown with cross-hatching. The broadcast radius $R_b > R_c/2$ where R_c is the cutoff radius. The interaction between a particle stored on Node A and a particle stored on Node B can be computed on any of the nodes in the overlap (cross-hatched) region.

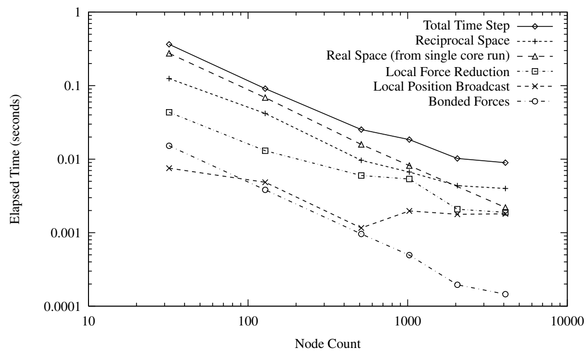


Figure 3: Scalability of major components of a time-step for the Rhodopsin system. The data were taken in dual core mode except for the "Real Space" data which had to be taken in single core mode because the trace facility only works on "Core 0" and in dual core mode the real space non-bond interactions are handled on "Core 1".

embedded in a lipid membrane with water surrounding it. Except for the real space non-bond data, the measurements were made in dual core mode with the real space and reciprocal (k) space operations overlapped. The real space measurements were made in single core mode because trace data is not available from the second core.

It remains to describe the reciprocal space operations that are common to both of the decompositions discussed above. The P3ME method as implemented in Blue Matter requires two 3D-FFTs for each iteration. A 3D-FFT can be computed by the "row-column" method involving successive evaluations of 1D-FFTs along one axis followed by a transpose of the data to permit one dimensional transforms of the data along another axis. To minimize communication, typical distributed 3D-FFT implementations use a "slab" decomposition in which the three dimensional FFT mesh is partitioned along a single mesh coordinate axis so that each node has a "slab" of the entire FFT mesh. This allows use of an optimized 2D-FFT on each node and a single transposition of the data followed by a series of 1D-FFTs. For our application, typical FFT sizes range from 64^3 to 128^3 which means that a distributed 3D-FFT using the "slab" decomposition would not scale beyond 128 nodes for a 128^3 FFT. Since our target is scalability to thousands of nodes, another approach had to be taken.

At each phase of the "row-column" method, there are a large number (N^2 for a 3D-FFT of dimensions $N \times N \times N$) of independent 1D-FFTs to be computed. Distributing these 1D-FFT computations over the nodes in the system would allow scalability of the 3D-FFT to much larger node counts, but at the cost of three communications-intensive transpose operations rather than one. With efficient use of the torus network on Blue Gene/L, we have been able to demonstrate an extremely scalable distributed 3D-FFT implementation[7]. As Figure 3 shows, the time required to carry out the reciprocal-space operations begins to become the limiting factor for an iteration at the highest node counts.

The current MPI-based implementation of the 3D-FFT takes advantage of a highly optimized `MPI_ALLTOALLV` implementation developed for BG/L[3]. This MPI collective, which is functionally equivalent to a collection of point-to-point messages between

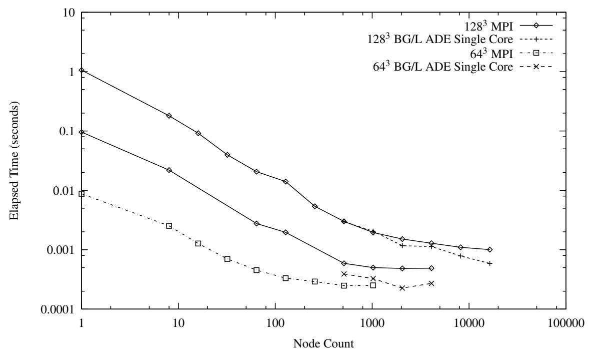


Figure 4: Performance measurements of the execution time for the volumetric 3D-FFT[7] running on MPI and on low level communications interfaces derived from the BG/L Advanced Diagnostics Environment[11].

all of the nodes, is used in the implementation of the distributed data transpose required by the 3D-FFT. For the transpose, in most regimes, the execution time is limited by the bisectional bandwidth of the machine, therefore a fixed size FFT will speed up as $p^{2/3}$ where p is the number of nodes and we assume that the nodes are arranged in a cubical array. However, at the limits of scalability for the FFT, where individual messages may contain only a single complex number, the software and hardware overheads associated with sending a single packet become important and we have investigated whether the use of lower-level interfaces can improve performance. Figure 4 compares the execution time obtained for 64^3 and 128^3 FFTs using MPI and using low level (hardware packet) interfaces derived from communications routines developed for hardware performance testing and diagnostics[11]. The data show that at very high node counts, the implementation using low level interfaces currently outperforms the MPI-based implementation.

6. SUMMARY

We have provided an overview of the Blue Matter molecular simulation application that was developed in support of and in conjunction with the Blue Gene science program. As part of that effort we have developed a new variant of spatial decomposition for n-body simulations that permits effective load balancing of real space interactions and which is compatible with a three-dimensional mesh/torus connected machine. To address scalability challenges encountered in correctly treating long range electrostatic interactions with periodic boundary conditions, we have demonstrated a distributed 3D-FFT that, for the sizes required for our application, continues to scale well to many thousands of nodes. These developments have allowed us to achieve sub-ten millisecond time steps on a 43K atom membrane protein system using a 4096 node Blue Gene/L partition which enables running microsecond simulations in less than two months. Scientific results have already appeared resulting from this work and further scientific studies are underway on additional systems of fundamental biological interest.

7. ACKNOWLEDGEMENTS

The authors would like to acknowledge the contributions of others who have worked on the Blue Matter application and those who have used it for production scientific work, including Yuriy Zhestkov, Alan Grossfield, Jed Pitera, Bill Swope, Ruhong Zhou, Yuk Sham, and Alex Balaeff. All of this work was made enabled by the efforts of the Blue Gene/L hardware and system software teams

8. REFERENCES

[1] F. Allen et al. Blue Gene: a vision for protein science using a petaflop supercomputer. *IBM Systems Journal*, 40(2):310–327, 2001. `http://www.research.ibm.com/journal/sj/402/allen.pdf`.

[2] M. Allen and D. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, New York, NY, 1989.

[3] G. Almasi et al. Design and implementation of message-passing services for the Blue Gene/L supercomputer. *IBM Journal of Research and Development*, 49(2/3):393–406, 2005. `http://www.research.ibm.com/journal/rd/492/almasi.pdf`.

[4] J. Bader and D. Chandler. Computer simulation study of the mean forces between ferrous and ferric ions in water. *The Journal of Physical Chemistry*, 96(15), 1992.

[5] S. Chatterjee et al. Design and exploitation of a high-performance SIMD floating-point unit for Blue Gene/L. *IBM Journal of Research and Development*, 49(2/3):377–392, 2005. `http://www.research.ibm.com/journal/rd/492/chatterjee.pdf`.

[6] 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(18):7678–7693, 1998.

[7] M. Eleftheriou, B. Fitch, A. Rayshubskiy, T. Ward, and R. Germain. Scalable framework for 3d FFTs on the Blue Gene/L supercomputer: Implementation and early performance measurements. *IBM Journal of Research and Development*, 49(2/3):457–464, 2005. `http://www.research.ibm.com/journal/rd/492/eleftheriou.pdf`.

[8] R. Enenkel et al. Custom math functions for molecular dynamics. *IBM Journal of Research and Development*, 49(2/3):465–474, 2005. `http://www.research.ibm.com/journal/rd/492/enenkel.pdf`.

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

[10] A. Gara et al. Overview of the Blue Gene/L system architecture. *IBM Journal of Research and Development*, 49(2/3):195–212, 2005. `http://www.research.ibm.com/journal/rd/492/gara.pdf`.

[11] M. Giampapa et al. Blue Gene/L advanced diagnostics environment. *IBM Journal of Research and Development*, 49(2/3):319–332, 2005. `http://www.research.ibm.com/journal/rd/492/giampapa.pdf`.

[12] M. Karplus and J. McCammon. Molecular dynamics simulations of biomolecules. *Nature Structural Biology*, 9(9):646–652, September 2002.

[13] J. Lorenz, S. Kral, F. Franchetti, and C. Ueberhuber. Vectorization techniques for the Blue Gene/L double FPU. *IBM Journal of Research and Development*, 49(2/3):437–446, 2005. `http://www.research.ibm.com/journal/rd/492/lorenz.pdf`.

[14] J. Moreira et al. Blue Gene/L programming and operating environment. *IBM Journal of Research and Development*, 49(2/3):367–376, 2005. `http://www.research.ibm.com/journal/rd/492/moreira.pdf`.

[15] L. Nyland, J. Prins, R. Yun, J. Hermans, H.-C. Kum, and L. Wang. Achieving scalable parallel molecular dynamics using dynamic spatial decomposition techniques. *Journal of Parallel and Distributed Computing*, 47(2):125–138, December 1997.

[16] 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. *Journal of the American Chemical Society*, 127(13):4576–4577, 2005. `http://dx.doi.org/10.1021/ja042715y`.

[17] S. Plimpton and B. Hendrickson. A new parallel method for molecular dynamics simulation of macromolecular systems. *Journal of Computational Chemistry*, 17(3):326–337, 1996.

[18] C. Wait. IBM PowerPC 440 FPU with complex-arithmetic extensions. *IBM Journal of Research and Development*, 49(2/3):249–254, 2005. `http://www.research.ibm.com/journal/rd/492/wait.pdf`.