RC23956 (W0605-074) May 12, 2006
Computer Science

IBM Research Report

Blue Matter: Approaching the Limits of Concurrency for Classical Molecular Dynamics

Blake G. Fitch, Aleksandr Rayshubskiy, Maria Eleftheriou, T. J. Christopher Ward, Mark Giampapa, Michael C. Pitman, Robert S. Germain
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Blue Matter: Approaching the Limits of Concurrency for Classical Molecular Dynamics

Blake G. Fitch Aleksandr Rayshubskiy Maria Eleftheriou
T.J. Christopher Ward Mark Giampapa Michael C. Pitman
Robert S. Germain
IBM Research Division
IBM Thomas J. Watson Research Center
1101 Kitchawan Rd Route 134 Yorktown Heights, NY 10598

25th April 2006

Abstract

This paper describes a novel decomposition for N-body simulations that has enabled Blue Matter to approach the effective limits of concurrency for molecular dynamics using particle-mesh (FFT-based) methods for handling electrostatic interactions. Using this decomposition, Blue Matter running on Blue Gene/L has achieved simulation rates in excess of 1000 time steps per second and demonstrated significant speed-ups to $O(1)$ atom per node. Blue Matter employs a Communicating Sequential Process (CSP) style model with application communication state machines compiled to hardware interfaces. The scalability achieved has enabled methodologically rigorous biomolecular simulations on biologically interesting systems, such as membrane-bound proteins, whose time scales dwarf previous work on those systems. Further scaling improvements will require exploration of alternative algorithms for treating the long range electrostatics.

1 Introduction

The ability of biomolecular simulations to make contact with experimental data is directly related to the time-scale accessible to the simulation. This necessitates strong scaling of a fixed size N-body problem, typically a system containing tens of thousands to hundreds of thousands of particles, onto a massively parallel computer with thousands or tens of thousands of nodes. For example, to achieve a simulation rate of one microsecond every two weeks requires a single time step to complete 1.2 milliseconds, or fewer than one million machine cycles on Blue Gene/L.

Continuing to improve overall time-to-solution for these N-body simulations with correct treatment of long range electrostatic interactions while decreasing the ratio of atoms per node to on the order of one atom per node is a tremendous challenge for a parallel application. Addressing this challenge requires the exploration of a number of alternatives for both algorithms and communications programming models.

Success in meeting this challenge enables detailed atomistic simulations of biologically interesting systems at time-scales and in ensemble sizes that were previously unattainable including 26 hundred nanosecond trajectories of a G-Protein Coupled Receptor (GPCR), rhodopsin, in a realistic membrane environment[13] and multiple micro-second scale simulations of that system and others. Furthermore, since the path to increased capability now seems to require increased concurrency, even working with larger systems with hundreds of thousands of atoms may require scalability to relatively small ratios of atoms per node. While there have been some theoretical studies of scaling in this limit[19], this work represents the first implementation of classical biomolecular simulation to demonstrate scaling to this degree.

1.1 Background on Blue Matter

The Blue Matter effort was undertaken for two reasons. First, to address one the primary goals of IBM's Blue Gene project[1]: 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. Second, to provide a concrete context for the exploration of the algorithmic techniques and programming models required to exploit the massive parallelism of the Blue Gene architecture.

2 Classical Biomolecular Simulation

2.1 N-body Problem, Long Range Electrostatics

Classical biomolecular simulation includes both Monte Carlo and Molecular Dynamics[8]. The focus of our work has been on Molecular Dynamics although the Replica Exchange or Parallel Tempering Method[18] which combines Molecular Dynamics with Monte Carlo-style moves has been implemented in Blue Matter as well[4]. Classical molecular dynamics is an N-body problem in which the evolution of the system is computed by numerical integration of the equations of motion. At each time step, forces on particles are computed; and then the equations of motion are integrated to update the velocities and positions of the particles.

Because models of proteins have components with large partial charges, the long ranged electrostatic forces cannot be approximated simply by cutting off force interactions between pairs of particles further apart than some cut-off distance. The most commonly used techniques for handling these long range interactions 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 finite range cut-off and a reciprocal space portion that involves a convolution of the charge distribution with an interaction kernel. This convolution is evaluated using a Fast Fourier Transform (FFT) method in the Particle-Particle-Particle-Mesh (P3ME) technique[2] used by Blue Matter. In this case the $O(n^2)$ dependence on particle number is reduced to $O(n \log n)$. The global data dependency for each time-step is imposed by the convolution step with the evaluation of the three dimensional FFTs.

We describe molecular dynamics as comprised of four major modules:

- real-space non-bonded (finite range pair interactions)
- k-space (FFT-based)
- bonded (graph-based)
- integration (per particle)

2.2 Inherent Concurrency of Molecular Dynamics

Before attempting to scale an algorithm onto many thousands of nodes, it is useful to consider how much concurrency is inherent in various components of that algorithm. An example of such an approach is pictured in Figure 1 for the non-bonded forces in a Molecular Dynamics simulation using the P3ME method to treat the long-range electrostatic forces. One can afford a lack of concurrency in components that impose very little computation or communication burden, but eventually even these will become bottlenecks (Amdahl's Law).

3 Description of Parallel Decomposition/Load Balancing

Our explorations of parallel decompositions have included replicated data methods that leveraged the hardware facilities of Blue Gene/L to globalize and reduce data structures[5, 11] and volume decompositions. Prior to the present work, the

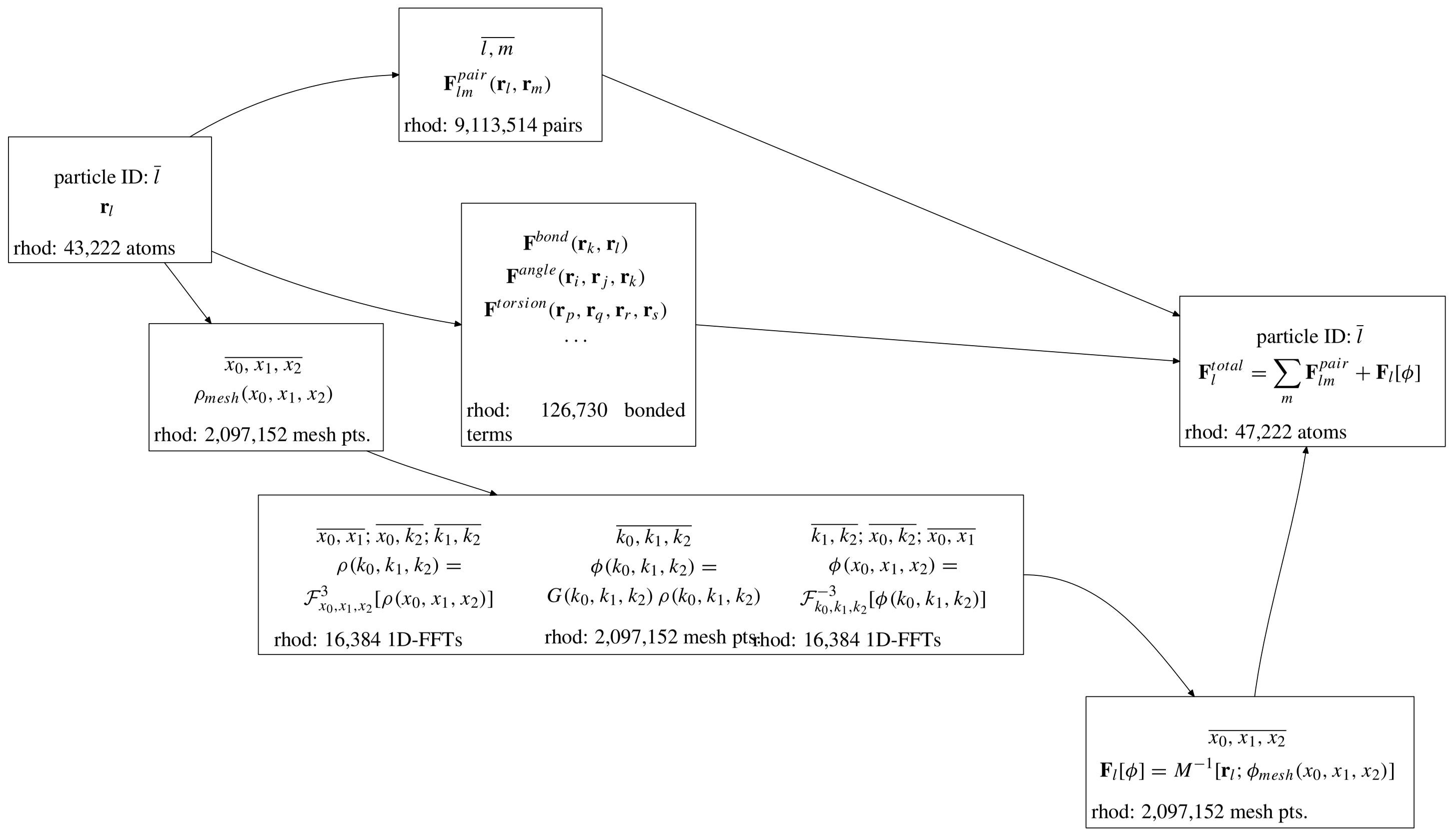
particle ID: $\bar{l}$
$\mathbf{r}_l$
rhod: 43,222 atoms
$\overline{l, m}$
$\mathbf{F}^{pair}_{lm}(\mathbf{r}_l, \mathbf{r}_m)$
rhod: 9,113,514 pairs
$\overline{x_0, x_1, x_2}$
$\rho_{mesh}(x_0, x_1, x_2)$
rhod: 2,097,152 mesh pts.
$\mathbf{F}^{bond}(\mathbf{r}_k, \mathbf{r}_l)$
$\mathbf{F}^{angle}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k)$
$\mathbf{F}^{torsion}(\mathbf{r}_p, \mathbf{r}_q, \mathbf{r}_r, \mathbf{r}_s)$
$\dots$
rhod: 126,730 bonded terms
particle ID: $\bar{l}$
$\mathbf{F}^{total}_l = \sum_m \mathbf{F}^{pair}_{lm} + \mathbf{F}_l[\phi]$
rhod: 47,222 atoms
$\overline{x_0, x_1}; \overline{x_0, k_2}; \overline{k_1, k_2}$
$\rho(k_0, k_1, k_2) = \mathcal{F}^3_{x_0,x_1,x_2}[\rho(x_0, x_1, x_2)]$
rhod: 16,384 1D-FFTs
$\overline{k_0, k_1, k_2}$
$\phi(k_0, k_1, k_2) = G(k_0, k_1, k_2)\,\rho(k_0, k_1, k_2)$
rhod: 2,097,152 mesh pts.
$\overline{k_1, k_2}; \overline{x_0, k_2}; \overline{x_0, x_1}$
$\phi(x_0, x_1, x_2) = \mathcal{F}^{-3}_{k_0,k_1,k_2}[\phi(k_0, k_1, k_2)]$
rhod: 16,384 1D-FFTs
$\overline{x_0, x_1, x_2}$
$\mathbf{F}_l[\phi] = M^{-1}[\mathbf{r}_l; \phi_{mesh}(x_0, x_1, x_2)]$
rhod: 2,097,152 mesh pts.

Figure 1: Diagram showing the data dependencies and opportunities for concurrency in various stages of the non-bonded force calculations in a Molecular Dynamics time step using the P3ME method. For each step, the actual number of independent work items are displayed for one of the molecular systems, Rhodopsin, benchmarked in this paper.

most recent volume decomposition, which we will refer to as V4, used geometric criteria to determine where the real-space non-bonded interaction between two particles would be computed, namely on whichever node contained the point halfway between the two particles. This provided a large number of distinct units of the computational burden that could be distributed by partitioning space using an Optimal Recursive Bisection (ORB) scheme[7, 10, 6]. The implementation of this method entailed the broadcast of a particle's position to a sphere with radius of half a cut-off distance. This also provided the particle positions required by the k-space and bonded force modules.

In contrast to the purely geometric approach of V4, the method described here, referred to as V5, uses geometry primarily as heuristic to prime the set-based optimization process that follows. Where V4 managed data distribution (of particle positions) and reduction (of forces) via a data "push" and caching module, V5 specializes communications between the integrator module and the other three force computation modules described above. Where the V4 push/cache method required a distinct communications phase, V5 allows overlap of one module's communication and/or computation with another module's communication and/or computation. On Blue Gene/L, 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.

With Blue Matter V5, a programming model resembling Communicating Sequential Processes (CSP) has evolved. Although still rough in implementation, it is clear that the four main modules in Blue Matter are connected by data channels defined by application specific protocols[14]. This model helps in two ways. First it enables the application to provide as much knowledge as possible to minimize communication overhead. Second, integrated application/communication state machines can be compiled directly to hardware interfaces further reducing overheads.

For each time step, after the integration module has run, a new set of atom positions are made available to each of the force-generating modules to be sent to the nodes where work has been distributed. Each force-generating module returns force partial sums to the integration module at the end of an operational phase. The force-generating modules manage work distribution using an initialization or planning phase to configure structures that will be stable for many iterations as well as dynamic activities that occur more frequently to manage the diffusion of particles. Periodically, the lists used to manage the pair interactions that must be computed (the Verlet lists), must be regenerated because of particle diffusion. Less frequently, diffusion also requires the assignment of particles to nodes to be updated. Outside of these special time steps, essentially all of the data communicated between nodes is consumed in useful computational work.

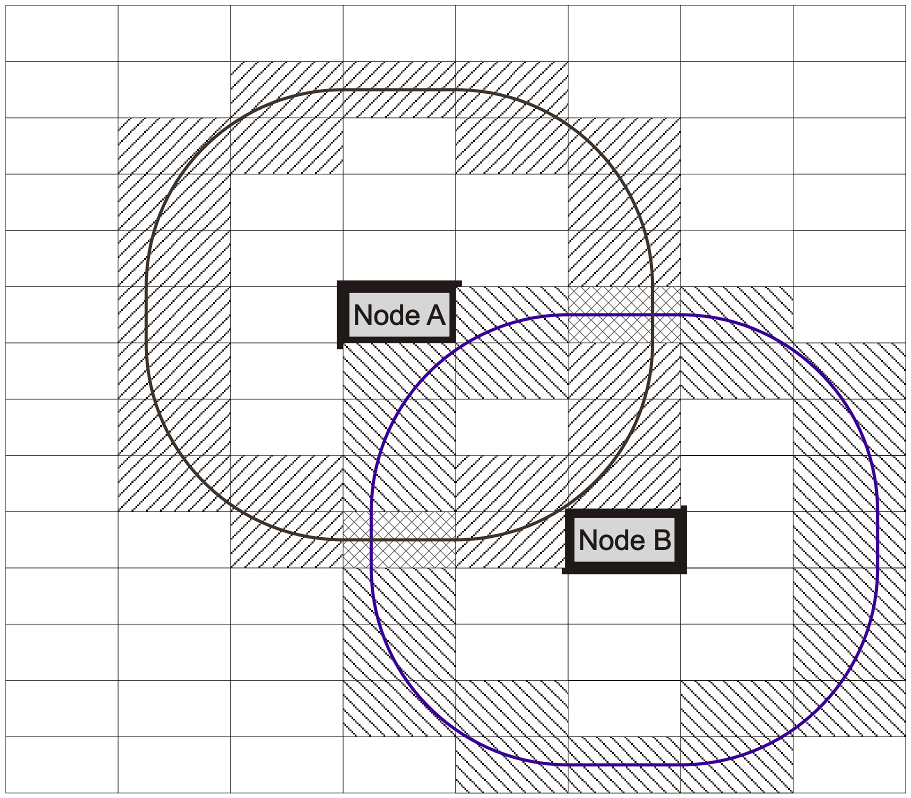

Figure 2: Spatial decomposition showing the “surface sets” for two nodes superimposed on the spatial decomposition of the domain onto all nodes (two-dimensional view for simplicity). The surface sets are comprised of nodes containing portions of the surface in simulation space defined by the set of points which are at distance R_b from the surface of any volume element assigned to Node A or Node B. The two surface sets are shown in two types of hatching with cross-hatching used to indicate “surface intersection set”. 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 node in the surface intersection set (the nodes with cross-hatching).

3.1 Real Space Non-bond

The real space non-bond (RSNB) module must distribute particle pair interactions so that computational load is balanced and communication is minimized. V5 uses a set-based optimization scheme primed with a spatial heuristic. During initialization, a node is assigned to compute any interaction for each pair of nodes which could home particles that are within cutoff. These node interaction assignments define the communication that is required. The V5 algorithm optimizes for both load balance and maximal reuse of communicated data. In the discussion that follows we will use node and node volume (the volume element of simulation space "owned" by that node) interchangeably.

The heuristic used by V5 to prime a set-based optimizer is based on the observation that for a given cutoff distance, each pair of nodes can define a set of nodes that *could* compute their interactions by those nodes which are half the cutoff distance from each member of the pair. In practice, for each node a "surface set" of nodes is defined as the nodes which intersect a surface a half cutoff distance away from the originating node volume's surface. For each pair of nodes within cutoff distance, a "surface intersection set" is defined as the intersection of the node pair's surface sets as shown in Figure 2.

The interactions between each pair of nodes could be assigned to any member of the surface intersection set for that pair. Load balance for RSNB computations can be achieved by assigning pair interactions to surface intersection sets in an optimized fashion. The optimizer will have the complete intersection set for each node pair from which to assign the pair computational load.

As fixed size simulations are scaled to higher node counts, surface sets become larger and correspondingly, the surface intersection sets become larger. A balance can be struck between load balance and communication cost by reducing the number of nodes in each surface set if one ensures at least one node in each surface intersection set required to cover in-range node pairs.

In order to further reduce the communication costs, V5 creates a "sparse surface set" via the following algorithm. Each node is considered to have an empty initial sparse surface set. For every pair of interacting nodes, the surface intersection set is generated. These sets are then sorted by the number of nodes in each. Starting with those surface intersection sets with the fewest members, begin adding to the sparse surface sets. When a surface intersection set has only one member, both nodes with intersecting surface sets must add that node to their sparse surface sets. In general, the following selection method is used for each node pair with a non null surface intersection set:

1. If one of the members of the surface intersection set already appears in the sparse surface set of both nodes in the node pair, the pairs interactions are

		Sparse Surface Node Count		
BG/L Partition Dimension	Surface Node Count	Avg.	Min.	Max
$8 \times 8 \times 8$	42	22	19	24
$8 \times 8 \times 16$	58	27	24	29
$8 \times 16 \times 16$	90	38	34	40
$16 \times 16 \times 16$	122	50	46	52
$8 \times 32 \times 16$	122	51	47	53
$16 \times 32 \times 16$	162	65	61	67

Table 1: This table shows the results of the algorithm described in the text to reduce the number of communication partners required by a node in the real-space non-bond module. The reduction from the Surface Node count in the second column to the Sparse Surface Node count characterized in the rightmost three columns is significant and becomes more pronounced as the partition size increases.

covered—do not add to sparse surface sets.

2. If none of the members of the surface intersection set appears in either sparse surface set for the node pair, choose the member that currently appears in the smallest number of sparse surface sets.

3. If members of the surface intersection set appear in either one of the node pairs sparse surface sets (but not both) then the member which currently appears in the smallest number of sparse surface sets is added to the other nodes sparse surface set.

The result of this process is a sparse surface set for each node which is guaranteed to have at least a one member intersection set with the sparse surface set of each node within cutoff distance. Measurements of the reduction in node count achieved by this algorithm are provided in Table 1. Using the sparse surface sets, for each node within range a sparse surface intersection set is created. Load balance is achieved by optimizing the assignment of computational burden within the options defined by each node pair's sparse surface intersection set. This scheme has the advantage that the granularity with which load can be manipulated decreases as the number of nodes increases (for a fixed size system).

3.2 K-space Module

The V5 k-space module has a distribution function set by a naturally mapped fully distributed FFT[3]. The FFT represents the bulk of the communication time of P3ME, but additional communication is required to send atom position data to those nodes where they will contribute to FFT mesh points and force partial sums resulting from contributions of those mesh points must be returned to the node owning the originating atom at the end of the phase. Since atoms diffuse during the simulation, an atom may contribute to a different set of FFT mesh points during each time step, requiring its position to be sent to different nodes. Ideally, atom positions will only be sent to those nodes managing the appropriate mesh points. However, this results in a fresh set of communicating node pairs on each timestep.

The communication protocol designed for the k-space module on Blue Gene/L takes advantage of the hardware global collective network[9, 12]. When the k-space module begins, it receives new position values from the integration module and for each, determines the list of target nodes owning relevant FFT mesh points. Each node then enters a loop sending positions via the hardware torus network to target nodes and receiving positions from other sources. As each node completes sending all local positions their targets, it contributes a value of the number of torus packets sent to an asynchronous hardware supported integer reduction. The nodes then continue to receive hardware torus packets while polling the global reduction hardware. The first time a global integer reduction completes, each node knows the total number of packets sent. Subsequently and until the receive phase is done, each time an integer reduction completes, each node contributes the number of torus packets received to the next reduction. The phase ends when the global number of torus packets received equals the global number of packets sent. The incremental cost of this method of terminating the communication phase is approximately that of a single integer all-reduce on the global collective network.

3.3 Bonded Force Module

The V5 bonded force module is responsible for executing force generating computations on sets of atoms that are connected by covalent bonds (representable as graphs). Because atoms are migrated through the spatial decomposition without regard to these graph based interactions, after atom migration each node with an atom that participates in graph operations that are not fully local must discover which nodes to exchange data with in support of these operations. Since graph based operations can involve up to four atoms and three bonds, the spatial range of nodes that share graph based computations is limited. When atoms are migrated to maintain the spatial decomposition, positions are sent to all nodes which might share a graph based operation. The majority of this set will not share an interaction and this knowledge is preserved for future bonded module communication phases until the next migration timestep. The actual number of nodes exchanging data on non-migration timesteps is the exact number required by the assignment of graph based operations typically an order of magnitude fewer than are in range.

				Time/time-step (milliseconds)			
Total	P_x	P_y	P_z	**Hairpin**	**SOPE**	**Rhodopsin**	**ApoA1**
512	8	8	8	2.24	6.22	17.51	38.42
1024	8	8	16	1.44	3.89	9.47	18.95
2048	8	16	16	1.00	2.45	5.07	9.96
4096	16	16	16	0.82	1.60	3.10	5.43
4096	8	32	16	1.11	2.10	3.21	5.39
8192	16	32	16		1.49	2.16	3.14
16384	32	32	16		1.66	1.87	2.08

Table 2: Performance in time/time step as a function of node count (and partition geometry) for β-Hairpin (5239 atoms), SOPE (13,758 atoms), Rhodopsin (43,222 atoms), and ApoA (92,224)

4 Benchmarking Results

All of the Blue Matter benchmarking data for V5 presented in Table 2 and included in Figure 3 was taken by averaging 180 time steps taken at the end of a 1000 time step run. Figure 3 plots the computational throughput in time steps per second versus the number of atoms per node. Plotting the data as a function of atoms per node provides some degree of normalization for system size and one can observe that the scalability plots at larger values of atoms/node (lower node counts) seem to lie on a "universal" curve. The is true in the limit that the real-space non-bonded interactions are the dominant contribution to the iteration time since the number of these pair interactions will scale with system size since they are truncated beyond the cutoff distance.

The β-Hairpin runs used a $64 \times 64 \times 64$ FFT mesh while all the other Blue Matter runs used a $128 \times 128 \times 128$ FFT mesh. All of the Blue Matter runs were made out to the largest node counts supported by our current FFT implementation, 4096 for the systems using a 64^3 mesh and 16,384 nodes for the systems using a 128^3 mesh. For comparison with the V5 results, Figure 3 includes the Blue Matter V4 data on both SPI and MPI[6], published results on the ApoA1 system for ports of NAMD to Blue Gene/L using both MPI and a lower level messaging layer[15], and finally benchmarking data on the same system from 2002 for NAMD on the PSC Lemieux system[16].

The V5 results show clear improvements over the V4 SPI benchmarks and in some cases actually approach time/time-step values limited by the k-space (FFT) module. The β-hairpin V5 results showed a 50% improvement over the V4 benchmarks and achieved an iteration time of 0.83 milliseconds or only 581,000 cycles

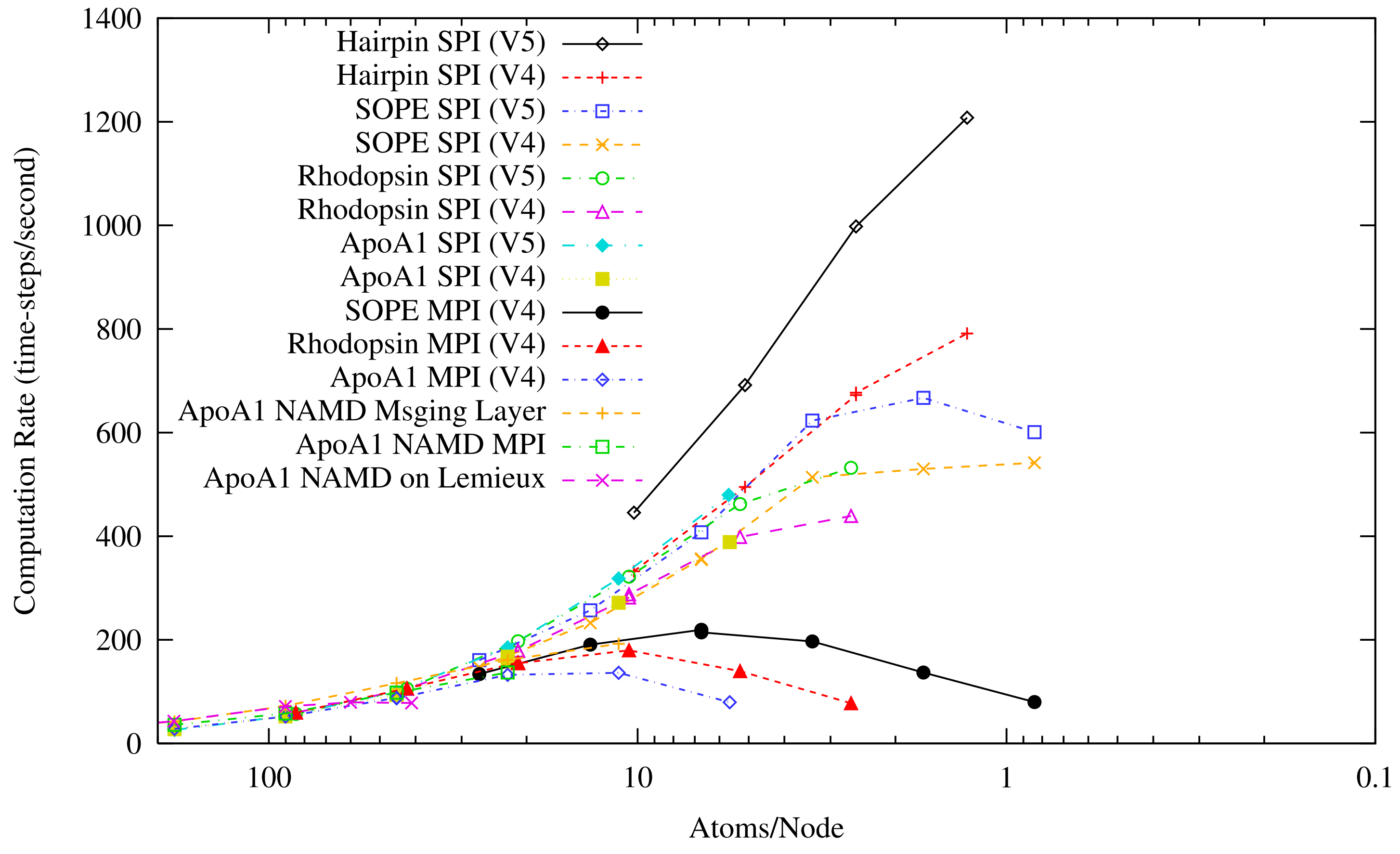

Figure 3: This plot shows computational rates on a number of molecular systems as a function of the number of atoms per node. This facilitates comparisons between systems of different sizes and also explicitly shows the degree of strong scaling achieved. Results for Blue Matter using the V5 method running on the low level communications System Programming Interface (SPI) provided by the Blue Gene/L Advanced Diagnostic Environment[12] are plotted along with results of a prior decomposition (V4) implemented on both MPI and SPI and data for NAMD on Blue Gene/L[15] and on the PSC Lemieux system[16].

on Blue Gene/L. The SOPE system also showed significant improvement, which was sufficient to expose the effects of overheads and caused speedup to end at 8192 nodes.

5 Summary and Conclusions

In this paper we have described a novel N-Body decomposition and a programming model used in Blue Matter to scale classical molecular dynamics simulations to the effective limits of parallel concurrency. This work has resulted in the demonstration of an unprecedented computation rate of over 1200 time steps per second for a small β-Hairpin system on 4096 nodes and continued speed-up for larger systems through 16,384 nodes. We have achieved this by removing nearly all non-scaling overheads from our application decomposition and our communication collectives. The time-to-solution achieved significantly contributes to increasing the potential overlap between simulation and experiment and has already had significant impact in the area of biomolecular simulation of membrane-bound protein systems[17, 13].

We have used very low level programming interfaces in this work; however, we believe the communicating sequential processes represented by the four modules in this program will allow similar results to be achieved via higher level programming constructs. It will be necessary for these programming models to compile down to nearly overhead free, integrated application/communication state machines. Where possible, these state machines should make use of "planning", or saved state to reduce overhead. We have hand generated such dedicated state machines for 3D FFT, neighborhood communications, all-to-all, and hardware reduce terminated one-sided communications and in each case exceeded the performance of available standard communication libraries.

This work suggests future research in two areas. First, although Blue Matter V5 exploits the BG/L hardware to achieve speedups through the practical limits of scalability of the 3D-FFT, it would be possible to improve efficiency at moderate node counts by optimizing all the communications and computations using techniques similar to ones described here for the real-space non-bond. We also anticipate formalizing our CSP-like programming model using dedicated channel protocols and light-weight processes compiled to hardware interfaces in our next version of the code (V6).

Acknowledgements: We would like to thank the members of the Blue Gene hardware team including P. Heidelberger, A. Gara, and M. Blumrich as well as others who have made use of and contributions to the Blue Matter code over the years, particularly Y. Zhestkov, F. Suits, W. Swope, J. Pitera, A. Grossfield, and

R. Zhou.

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.

[2] Markus Deserno and Christian 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.

[3] M. Eleftheriou, B. Fitch, A. Rayshubskiy, T.J.C. Ward, and R.S. Germain. Performance measurements of the 3d FFT on the Blue Gene/L supercomputer. In J.C. Cunha and P.D. Medeiros, editors, *Euro-Par 2005 Parallel Processing: 11th International Euro-Par Conference, Lisbon, Portugal, August 30-September2, 2005*, volume 3648 of *Lecture Notes in Computer Science*, pages 795–803. Springer-Verlag, 2005.

[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. In *Fifth IEEE International Workshop on High Performance Computational Biology*, April 2006.

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

[6] Blake G. Fitch, Aleksandr Rayshubskiy, Maria Eleftheriou, T.J. Christopher Ward, Mark Giampapa, Yuri Zhestkov, Michael C. Pitman, Frank Suits, Alan Grossfield, Jed Pitera, William Swope, Ruhong Zhou, Scott Feller, and Robert S. Germain. Blue Matter: Strong scaling of molecular dynamics on Blue Gene/L. In V. Alexandrov, D. van Albada, P. Sloot, and J. Dongarra, editors, *International Conference on Computational Science (ICCS 2006)*, volume 3992 of *LNCS*, pages 846–854. Springer-Verlag, 2006.

[7] Blake G. Fitch, Aleksandr Rayshubskiy, Maria Eleftheriou, T.J. Christopher Ward, Mark Giampapa, Yuri Zhestkov, Michael C. Pitman, Frank Suits, Alan Grossfield, Jed Pitera, William Swope, Ruhong Zhou, Robert S. Germain, and Scott Feller. Blue matter: Strong scaling of molecular dynamics on blue gene/l. Research Report RC23688, IBM Research Division, August 2005. `http://domino.research.ibm.com/library/`

`cyberdig.nsf/1e4115aea78b6e7c85256b360066f0d4/58073ef405c70e59852570540067da52?OpenDocument\&Highlight=0,RC23688`.

[8] D. Frenkel and B. Smit. *Understanding Molecular Simulation*. Academic Press, San Diego, CA, 1996.

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

[10] Robert S. Germain, Blake Fitch, Aleksandr Rayshubskiy, Maria Eleftheriou, Michael C. Pitman, Frank Suits, Mark Giampapa, and T.J. Christopher Ward. Blue Matter on Blue Gene/L: massively parallel computation for biomolecular simulation. In *CODES+ISSS '05: Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 207–212, New York, NY, USA, 2005. ACM Press.

[11] 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 Journal of Research and Development*, 49(2/3):447–456, 2005.

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

[13] Alan Grossfield, Scott E. Feller, and Michael C. Pitman. A role for direct interactions in the modulation of rhodopsin by omega-3 polyunsaturated lipids. *PNAS*, 103(13):4888–4893, 2006.

[14] C.A.R Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[15] Sameer Kumar, Chao Huang, Gheorghe Almasi, and Laxmikant V. Kale. Achieving strong scaling with NAMD on Blue Gene/l. 20th IEEE International Parallel and Distributed Processing Symposium. IEEE, 2006. http://charm.cs.uiuc.edu/papers/NAMDIDPDS06.pdf.

[16] J.C. Phillips, G. Zheng, S. Kumar, and L.V. Kale. NAMD: biomolecular simulation on thousands of processors. In *Supercomputing 2002 Proceedings*, 2002. `http://www.sc2002.org/paperpdfs/pap.pap277.pdf`.

[17] Michael C. Pitman, Alan Grossfield, Frank Suits, and Scott 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.

[18] Y. Sugita and Y. Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chem. Phys. Lett.*, 314:141–151, 1999.

[19] V.E. Taylor, R.L. Stevens, and K.E. Arnold. Parallel molecular dynamics: implications for massively parallel machines. *Journal of Parallel and Distributed Computing*, 45(2):166–175, September 1997.