

Impact of Multicores on Large-scale Molecular Dynamics Simulations

Sadaf R. Alam, Pratul K. Agarwal, Scott S. Hampton, Hong Ong, and Jeffrey S. Vetter

Computer Science and Mathematics Division

Oak Ridge National Laboratory, Oak Ridge, TN 37831

{alamsr,agarwalpk,hamptonss,hongong,vetter}@ornl.gov

Abstract

Processing nodes of the Cray XT and IBM Blue Gene Massively Parallel Processing (MPP) systems are composed of multiple execution units, sharing memory and network subsystems. These multicore processors offer greater computational power, but may be hindered by resource contention. In order to understand and avoid such situations, we investigate the impact of resource contention on three scalable molecular dynamics suites: AMBER (PMEMD module), LAMMPS, and NAMD. The results reveal the factors that can inhibit scaling and performance efficiency on emerging multicore processors.

1. Introduction

Massively Parallel Processing (MPP) systems, composed of thousands of multicore processing devices, are becoming a dominant architectural paradigm in high performance computing. The shift in processor architecture focus from the traditional improvement in clock speed to using multiple cores introduces another level of parallelism at the processing layer. As the number of cores increases per chip, data locality, shared cache, bus contention, and memory bandwidth limits become even more difficult to manage due to increases in resource sharing. Additionally, from a parallel application's perspective, the increase in number of cores indicates that there will be more intensive intra-node communication. Therefore, it is important to identify the factors that could potentially limit the performance and scalability of applications.

In this study, we aim to quantify the costs across different variants of multicore devices and molecular dynamics (MD) frameworks through a comprehensive measurement. MD simulations typically repeat

identical sequences of operations and run for extended periods. Therefore, a small improvement or degradation in performance could have significant implications. Performance and scaling improvements across different multicore devices could be achieved by identifying bottlenecks and by understanding and identifying the cost factor in hardware and software stack as well as the application implementation. In particular, we present a methodology for characterizing the performance of a diverse range of multicore devices in the context of three scalable, production-level MD simulation frameworks.

We use the JAC (Joint Amber [13] and CHARMM [12]) test case [6], which contains 23,558 total atoms, as an input to the two parallel versions of the MD simulation frameworks, Amber PMEMD, and LAMMPS [5][17]. NAMD [8][16] is another well-known framework based on the Charm++ runtime system. For MPP runs, we selected medium-scale biological systems of few hundred thousands atoms to large-scale systems of up to 3 million atoms. We analyze performance and scaling of MD test cases on these tightly integrated MPP systems and compare and contrast these with emerging, stand-alone multicore microprocessors. On the emerging quad-core microprocessors systems [2], our finding indicates that the MD applications performance is highly sensitive to the MPI communication library implementation, tuning, and usage in the application. MPP systems, including Cray XT [3][9][10] and IBM Blue Gene [4][15], offer high network injection and bisection bandwidth to tens of thousands of processing cores. We subsequently evaluate the impact of network performance on scalable MD applications. The unique contribution of this study is a comprehensive evaluation and analysis of a range of biological systems using the major scalable MD frameworks on emerging multicore and MPP systems.

The paper outline is as follows: Section II provides an overview of the scalable MD frameworks and the motivation of the proposed study. Section III describes the testing environment, including the hardware, software, and test cases. Performance evaluation results and analysis of simulation runs is presented in section IV. Section V concludes with the key findings of this study and directions for future plans.

2. Background and Motivation

MD simulations enable the study of complex, dynamic processes that occur in biological systems. MD methods are now routinely used to investigate the structure, dynamics, function, and thermodynamics of biological molecules and their complexes. A typical bimolecular simulation contains atoms for the solute, ions, and the solvent molecules. The motions of individual atoms can be determined by numerically solving Newton’s equations of motion, which relate the total force on an atom to its mass and acceleration.

$$F_i = ma$$

The total force on each atom is a contribution from individual forces due to chemical bonds, as well as non-bonded interactions with all other atoms. The force is calculated from the negative gradient of the potential energy function, U , which is given by,

$$U = \sum_{bonds} c_b (l_i - l_0)^2 + \sum_{angles} c_a (\theta_i - \theta_0)^2 + \sum_{torsions} c_t (1 + \cos(n\omega - \gamma)) \\ + \sum_{i=1}^N \sum_{j=1}^N 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right] + \sum_{i=1}^N \sum_{j=1}^N \frac{q_i q_j}{r_{ij}}$$

The latter two terms represent the van der Waals and electrostatics terms, respectively. Although forces are defined among all atom pairs, in practice, MD simulations evaluate only those pairs within a *cutoff* distance for computational efficiency. Each particle interacts with the nearest images of the other $N-1$ particles only in a sphere of radius R_{cutoff} . The cutoff limits the number of non-bonded interactions in the sum to be N_ρ , with ρ the atom density, as compared to $N(N-1)$ interactions without the cutoff. For the van der Waals interactions, the cutoff error is negligible within a reasonable distance since it is proportional to r_{ij}^{-6} . However, the electrostatic sum has a much larger error, as it is proportional to r_{ij}^{-1} . Ignoring the electrostatic interactions beyond the cutoff can introduce spurious effects in the energy and forces, resulting in artificial

force magnitudes. The Particle Mesh Ewald (PME) method provides a solution to this problem by interpolating energies to a grid and then using a fast Fourier transform for the calculations. PME is nearly as accurate as computing all pairs, but reduces the number of non-bonded interactions to $N \log N$ [11][14]. A number of established MD codes implement PME, including AMBER, LAMMPS, and NAMD.

3. Testing Environment

Hardware

The experimental data collection for this study is undertaken on x86-based standalone systems, Cray XT3 [9], Cray XT4 [10], IBM Blue Gene/L [15], and Blue Gene/P [4] systems. The x86-based standalone systems include an eight socket, dual-core AMD Opteron 8216 [2], an eight socket, quad-core AMD Opteron 8350 [1], and an Intel quad-core Clovertown system [18]. The AMD based systems use PGI compilers, whereas the Intel compiler is used on their platform. The cluster systems run standard versions of the Linux operating system. The Cray XT3 and XT4 systems are composed of dual-core Opteron processors. Cray XT4 contains Rev F Opteron while the Cray XT3 system is composed of an earlier release of dual-core Opteron processor. Additionally, the XT3 uses the SeaStar NIC, while the XT4 uses the SeaStar2 NIC. The SeaStar2 increases the peak network injection bandwidth of each node from 2.2 GB/s to 4 GB/s when compared to SeaStar, and increases the sustained network performance from 4 GB/s to 6 GB/s. The IBM Blue Gene/L and Blue Gene/P systems are composed of PowerPC processors. The Blue Gene/L systems has 2 compute cores per processing node, while the Blue Gene/P system, a predecessor of the Blue Gene/L system with higher frequency, has 4 execution cores per processing node. IBM compilers are used on the Blue Gene systems. Note that many of the results on the MPP systems were collected as a part of the early performance evaluation effort. We anticipate that the achievable performance and scaling could be improved by system-specific optimization and tuning as well as with the maturity of the software stack including compilers, operating and runtime systems that control the mapping and placement of parallel MPI tasks.

Software

The three MD frameworks that we used are PMEMD, LAMMPS, and NAMD. PMEMD is a module of

AMBER that has been written with the major goal of improving performance of PME in molecular dynamics simulations and minimizations by Robert E. Duke and Lee G. Pedersen. PMEMD is implemented in Fortran 90 and MPI. LAMMPS models an ensemble of particles in a liquid, solid, or gaseous state, and can be used to model atomic, polymeric, biological, metallic or granular systems. The version we used for our experiments is written in C++ and MPI. It is the only implementation that is reported to scale to 64K Blue Gene/L processors. It should be noted that the scaling numbers are reported in the weak scaling mode, i.e., not for a fixed-size problem. NAMD is a C++ based parallel program, implemented using the Charm++ parallel programming system. It uses object-based decomposition methods and measurement-based dynamic load balancing to achieve its high performance. NAMD uses a combination of spatial decomposition and force decomposition techniques to generate a high degree of parallelism. NAMD developers describe several techniques to scale it to 8,192 processors on Blue Gene/L.

Test Cases

The bio-molecular systems used for our experiments were designed to represent the variety of complexes routinely investigated by computational biologists. We considered the following three test cases for our experiments:

- Small: 23,558 atoms (JAC) and 61,641 atoms (HhaI).
- Medium: 290,220 atoms (RuBisCO).
- Large: 1,066,628 atoms and 2,640,030 atoms

The smallest system is the JAC (Joint Amber CHARMM) benchmark. JAC is a dihydrofolate reductase (159 residue protein) in TIP3P water (23,558 total atoms), in a periodic box with constant volume and explicit solvent. PME is used for electrostatics, and van der Waals interactions are truncated at 9Å. The HhaI system is a model for protein-DNA complex (enzyme m5C-methyltransferase M. HhaI with its target DNA sequence), in explicit solvent and counterions to allow the system to be charge neutral. This model consists of 61,641 atoms with explicit treatment of solvent using the TIP3P water model. AMBER's tleap module was used for system preparation and the AMBER parm98 force-field was used. The system was equilibrated before benchmarking runs. The time-step is 10^{-15} seconds. The long-range forces are calculated using PME. A medium-scale system we

considered is the RuBisCO enzyme, which is based on the crystal structure 1RCX. The RAQ system is a model of the RuBisCO enzyme in explicit solvent and was prepared in a way similar to the HhaI system. This model consists of 290,220 atoms with explicit treatment of solvent. The time-step for each run is also 1 fs. We consider two representative large-scale biological systems with 1,066,628 atoms and 2,640,030 atoms, respectively. The first test case models Satellite Tobacco Mosaic Virus (STMV) [7]. It uses periodic boundary conditions and the PME for electrostatics. The other test case has a similar configuration except that it uses slightly different PME parameters.

4. Experiments and Results

To understand the impact of multicore technologies on MD simulation, we evaluate performance of two out-of-the-box applications on an AMD quad-core processor. Figure 1 shows the parallel efficiency results (speedup/number of cores). Note that these results compare performance starting from 2 cores rather than 1 core. This is because PMEMD requires a minimum of 2 MPI tasks. Therefore the actual efficiency could be in fact slightly smaller. We ran a small (JAC, 24K atoms) and a medium (RUB, 290K atoms) size problem on two scalable MD frameworks, LAMMPS and PMEMD. We observe that on a system with 8 quad-core processors connected in a SMP manner, the efficiency could be lower than 50% just by using only half of the total cores.

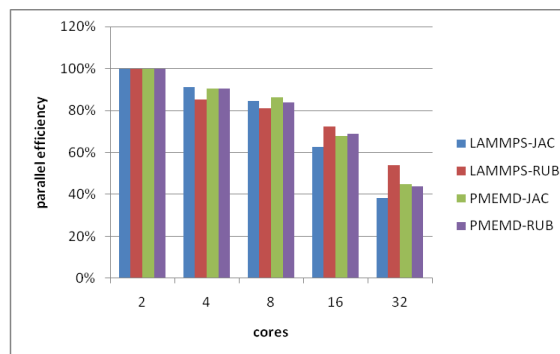


Figure 1: Parallel Efficiency on a quad-core processor

Quadcore Performance and Scaling

We begin by evaluating performance and scaling of a small-scale test case, JAC, using the PMEMD and LAMMPS frameworks. A single MPI task is mapped onto a single core in all test cases. We compare these

results with the Cray XT3 and XT4 results, systems that have similar characteristics except for a high memory bandwidth and high network injection bandwidth. The bisection bandwidth of the XT3 and XT4 systems is the same. Figure 2 shows performance results in (10^{-12} seconds) per simulation day on a dual-core AMD Opteron and two contemporary quad-core systems along with MPP system results by simulating the JAC benchmark in LAMMPS framework. On smaller core count, we observe that the Intel Clovertown system outperforms all other multicore systems as a result of a higher clock frequency system and a large shared cache per die. However, as all cores begin sharing the bus and cache, the resource contentions result in slower performance and limit scaling to large number of cores. The Opteron systems, on the other hand, have relatively low clock frequencies, but the on-chip memory controller and the Hyper-transport (HT) links provide better scaling. Still at the higher core count, scaling in the SMP configuration of 8 cores suffer while the 100 series single-chip, dual-core XT3 and XT4 systems that are connected to the network via HT links provide much higher efficiencies.

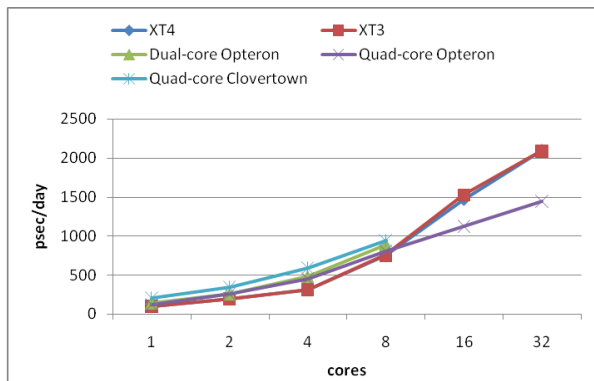


Figure 2: Per core (x-axis) performance in pico-seconds per day (y-axis) LAMMPS simulation of the JAC test case.

To quantify and understand the scaling behavior of the results presented in Figure 2, we calculated speedup on multiple cores or MPI tasks with respect to single core runtimes (speedup equals time on one core divided by time on P cores). The results are shown in Figure 3. The speedup on the XT4 system is the highest while the Clovertown system is the lowest. The two Opteron SMP systems with 8 sockets provide similar levels of scaling despite having different number of cores per socket and slightly lower frequency on the quad-core system. In other words, the performance of this simulation run is probably less sensitive to sharing

resources per socket than sharing resources across sockets.

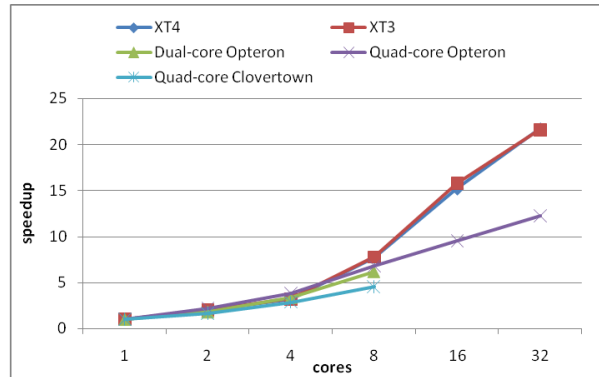


Figure 3: Speedup (Y-axis) on number of cores (X-axis) for JAC simulation using LAMMPS

We repeat the similar set of experiments with the PMEMD simulations of the JAC benchmark. The results are shown in Figure 4. Although the implementation of LAMMPS and PMEMD are significantly different, the scaling behavior of these applications is similar across the five targeted multicore platforms. In other words, the Clovertown system outperforms the other multicore platforms on smaller core counts but the scaling is limited to a few cores. Another interesting observation is the scaling of the PMEMD simulation runs on the target multicore systems. Despite the different usage patterns of MPI communication operations in PMEMD and LAMMPS, these exhibit similar scaling behavior for simulating JAC on up to 32 cores.

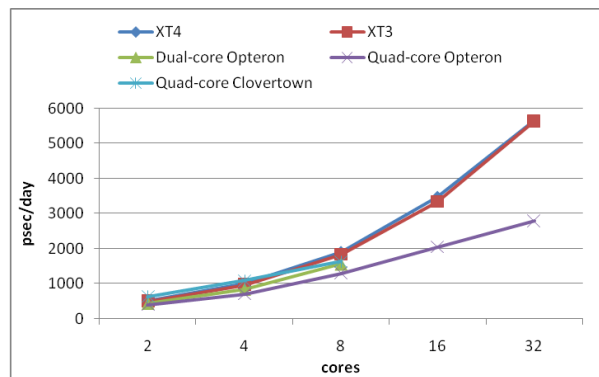


Figure 4: Per core (x-axis) performance in pico-seconds per day (y-axis) PMEMD simulation of the JAC test case

Scaling with Workload Size

Since JAC is a small problem case, one could argue that it is not capable of exploiting and saturating the computing resources available on the multicore systems. At the same time, we are also interested in understanding the scaling behavior and impact of resource contention on multicore systems with increased workload volume. We therefore ran experiments on the AMD quad-core system and compared results of two test cases, small (JAC) and medium (RUB) using LAMMPS and PMEMD. Results are shown in Figure 5. Here we note that the scaling of PMEMD is not sensitive to the problem size (24K atoms as compared to 290K atoms) while the LAMMPS scaling behavior changes significantly on the higher processor count. We attribute this behavior to the increase in the workload volume as a function of problem size. In LAMMPS the workload volume per processor does not increase with the same rate as the PMEMD communication volume. Increase in the computation volume, however, is proportional in the two implementations.

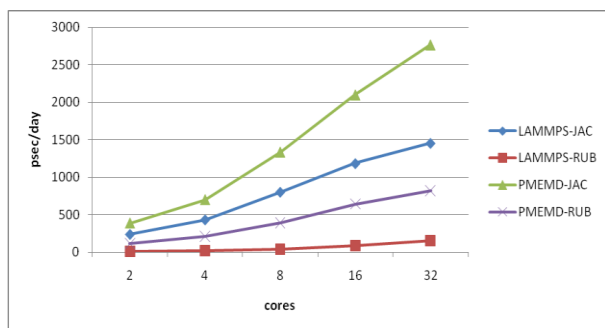


Figure 5: Per core (x-axis) performance in pico-seconds per day (y-axis) on AMD quad-core system

Parallel Efficiency on MPP Systems

We compare applications performance and scaling on Teraflop-scale contemporary MPP systems. For these experiments, we selected medium to large-scale test cases (up to 3M atoms) and ran experiments using the simulation frameworks that are known to scale to tens of thousands of MPI tasks.

A set of experiments is conducted using LAMMPS with a 62K atom system (HhAI) and 300K atom system (RUB), both with explicit solvent. The sizes of the FFT grid in these simulations determine the scaling limits. The HhAI system could scale to 1024 MPI tasks while the

system could scale to 4096 MPI tasks. Figure 6 shows performance slowdown for HhAI and RUB runs using LAMMPS on Cray XT3 and XT4 systems. These results on the dual-core systems show that the performance slow-down could be as high as 50% if an application is built and run in the default mode. Our earlier results showed that this slowdown could be even higher for the quad-core processors.

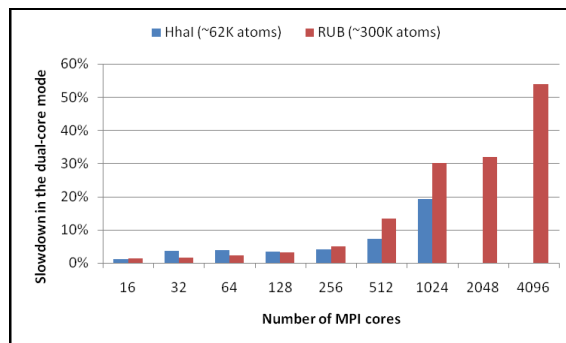


Figure 6: Per core (x-axis) performance slowdown with respect to single core (y-axis) LAMMPS simulation

No system specific modifications and optimizations are performed for these simulation runs. It is worth noting here that all these results are collected as part of early system evaluation and subsequent upgrades could result in significant performance improvements, as we note for the case of XT3 and XT4 runs.

We compare performance and scaling of the RUB test case on the target MPP systems in Figure 7. Here we note that subsequent generations of the systems, XT series and Blue Gene series, result in performance improvements for applications particularly on large number of MPI tasks mainly due to improvements in network and memory bandwidth. At the same time, there are smaller changes in the speedup ratios comparing the subsequent generations of the two systems suggesting that additional efforts are needed to exploit the enhanced architectural features of the target systems efficiently. In the case of MD applications, this is particularly challenging since the number of atoms or workload volume per processor is not fixed throughout the simulation runs. The result is load imbalances in computation as well as in communication. Our next set of experiments show that the scaling and performance benefits could be seriously limited due to the load imbalance and synchronization on large number of cores.

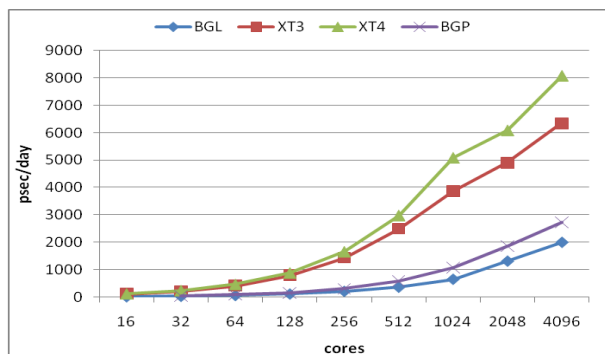


Figure 7: Per core (x-axis) performance in psec per day (y-axis) LAMMPS simulation of the RUB test case

The analysis of NAMD simulation runs reveals a slightly different set of issues. Unlike LAMMPS and PMEMD, NAMD is built on the Charm++ execution and runtime framework, which is being built on XT and Blue Gene MPI libraries. Charm++ employs virtualization techniques such that the programmer divides the program into a large number of parts independent of the number of processors. The scaling on NAMD simulation on large-scale test cases (3M atoms) is shown in Figure 8. In this particular case, we observe the performance results converge especially at higher number of MPI tasks. We used standard runtime and MPI profiling tools and the Charm++ projection tool to understand the cause of this performance behavior. Our analysis reveals that there are some severe load balancing issues and the difference between communication volume assigned to one MPI task or core can vary significantly as the numbers of cores increase.

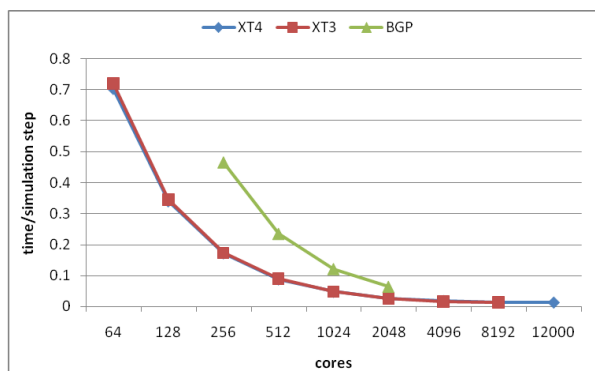


Figure 8: Per core (x-axis) performance in time per simulation step (y-axis) NAMD simulation of the 3M atoms test case

5. Conclusions and Future Plans

There are many benchmarking tools for evaluating the performance of hardware and software components. However, a majority of these benchmarking tools primarily focus on a particular system component. Thus, these tools often do not present a holistic view of a multicore based MPP system. This study showed that a systematic evaluation of the interaction and discovering the critical performance path is crucial to solve the problem at hand.

We have shown that the performance and scaling of the MD simulations on multicore platforms depend on a range of factors including the hardware design features, software stack and implementation of the simulation framework. On stand-alone dual and quad-core systems, the applications showed sensitivity to the implementation and usage of the MPI communication library. On the other hand, on the large-scale MPP systems based on the multicore processors, the load balancing and maturity of the software stack is more critical for sustaining performance and scaling MD applications. Our results capture the workload characteristics of a production-level application and scaling limiting factors for existing test cases and future problem configurations.

We plan to develop platform-independent symbolic models for our target applications to identify and to subsequently address scaling-limiting features in their computation and communication behavior. On the multicore platforms, we will experiment with alternate MPI library implementations and configurations. We also plan to explore system software stack for optimal scheduling and mapping of MPI tasks in MD simulations. On the application front, particularly at large scale, we anticipate that alternate algorithms, programming models and implementation will be investigated to reduce the load balancing problems.

Acknowledgements

The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. The authors would like to thank National Center for Computational Sciences (NCCS) for access to Cray XT3 and support (INCITE award).

References

- [1] AMD Opteron Rev. F, details available at <http://multicore.amd.com/>
- [2] AMD quad-core platform, <http://multicore.amd.com/us-en/quadcore/>
- [3] Cray XT Systems. Available from: <http://info.nccs.gov/resources/jaguar>.
- [4] IBM Blue Gene/P system, <http://www-03.ibm.com/servers/deepcomputing/bluegene.html>
- [5] LAMMPS Molecular Dynamics Simulator, <http://lammmps.sandia.gov/>
- [6] MD benchmarks for AMBER, CHARMM and NAMD, <http://amber.scripps.edu/amber8.bench2.html>
- [7] Molecular Dynamics of Viruses; Available from: <http://www.ks.uiuc.edu/Research/STMV/>
- [8] NAMD Scalable Molecular Dynamics Code, <http://www.ks.uiuc.edu/Research/namd/>
- [9] S.R. Alam, R.F. Barrett, *et. al.* (2007), "An Evaluation of the ORNL Cray XT3," *J. High Performance Computing Applications* (to appear).
- [10] S.R. Alam, R.F. Barrett, *et. al.* (2007), "Cray XT4: An Early Evaluation for Petascale Scientific Simulation," *ACM/IEEE Supercomputing Conference (SC07)*.
- [11] S. R. Alam, P. K. Agarwal, *et. al.* (2006), "Performance Characterization of Bio-molecular Simulations using Molecular Dynamics," *Principle and Practices of Parallel Programming (PPOPP'06)*.
- [12] B. R. Brooks, R. E. Bruccoleri, *et al.*, (1983) "CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations", *J. Comp. Chem.*, 4, 187-217.
- [13] D. Case, *et. al.*, "The Amber Bio-molecular Simulation Programs," *J. of Comp. Chemistry*: 1668-1688, 2005.
- [14] M. Crowley, *et. al.*, "Adventures in Improving the Scaling and Accuracy of Parallel Molecular Dynamics Program," *J. of Supercomputing*, 11, 1997.
- [15] M. Ohmacht, R. A. Bergamaschi, *et al.*, "Blue Gene/L compute chip: Memory and Ethernet subsystem," *IBM Journal of Research and Development*, Vol. 49, No. 2/3, 2005.
- [16] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kale, and Klaus Schulten. "Scalable molecular dynamics with NAMD." *Journal of Computational Chemistry*, 26:1781-1802, 2005.
- [17] S. J. Plimpton (1995), "Fast Parallel Algorithms for Short-Range Molecular Dynamics", *J. Comp. Phys.*, 117, 1-19; <http://www.cs.sandia.gov/~sjplimp/lammmps.html>
- [18] R. M. Ramanathan, "Intel Multi-core Processors: Making the move to Quad-core and Beyond", white paper available at <http://www.intel.com/technology/architecture/downloads/quad-core-06.pdf>