On the Path to Enable Multi-scale Biomolecular Simulations on PetaFLOPS Supercomputer with Multi-core Processors

Sadaf R. Alam and Pratul K. Agarwal

Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN, USA 37831
alamsr,agarwalpk@ornl.gov

Abstract

Biological processes occurring inside cell involve multiple scales of time and length; many popular theoretical and computational multi-scale techniques utilize biomolecular simulations based on molecular dynamics. Till recently, the computing power required for simulating the relevant scales was even beyond the reach of fastest supercomputers. The availability of petaFLOPS-scale computing power in near future holds great promise. Unfortunately, the bio-simulations software technology has not kept up with the changes in hardware. In particular, with the introduction of multi-core processing technologies in systems with tens of thousands of processing cores, it is unclear whether the existing biomolecular simulation frameworks will be able to scale and to utilize these resources effectively. While the multi-core processing systems provide higher processing capabilities, their memory and IO subsystems are posing new challenges to application and system software developers. In this preliminary study, we attempt to characterize computation, communication and memory efficiencies of bio-molecular simulations on a Cray XT3 system, which has recently been upgraded to dual-core Opteron processors. We identify that the application efficiencies using the multi-core processors reduce with the increase of the simulated system size. Further, we measure the communication overhead of using both cores in the processor simultaneously and identify that the MPI communication performance can be as low as 50% as compared to the single-core execution times. We conclude that not only the biomolecular simulations need to be aware of the underlying multi-core hardware in order to achieve maximum performance but also the system software needs to provide processor and memory placement features in the high-end systems. Our results on a stand-alone dual-core AMD system confirm that combinations of processor and memory affinity schemes can result in over 12% performance gains.

1 Introduction

A better knowledge of biomolecules is the key to understanding mechanistic details of the various biochemical processes that occur in all living cells. The biomolecular structure, dynamics and function span multiple scales of time and length [4,5,6,7]. In the past, experimental techniques have provided a wealth of information into the working of biomolecules; more recently theoretical and computational multi-scale modeling techniques based upon biomolecular simulations continue to provide novel insights [7]. Till recently, the computing power required for simulating the length and time scales relevant to biomolecules were beyond the reach of even the fastest supercomputers. In particular, the dynamics and functions of biomolecules span more than 15 orders of magnitude in time; the computing power falls short by 4-6 orders of magnitude in its ability to simulate the desired time-scales [8]. The availability of petaFLOPS-scale computing power in near future holds great promise for this area. Many of the popular biomolecular simulations codes in use today were designed several decades ago based on a different programming paradigm in mind. Unfortunately, it is now becoming evident that the bio-simulations software technology has not kept up with the change in hardware. In particular, with the introduction of multi-core processing technologies in systems with tens to hundreds of thousands of processing cores, it is unclear whether the existing biomolecular simulation frameworks will be able to scale and to utilize these resources effectively.

Microprocessor vendors today have ability to produce chips with an ever-increasing number of transistors, therefore, the approach of duplicating existing cores is a straightforward way to address problems related to physical and power constraints and limited instruction-level parallelism. However, because all cores of a processor share the link between the processor's resources including memory, IO links and off-node communication contention for these resources can limit the achievable performance when using more than one core per processor. Applications, such as biomolecular simulations, can perform well on systems with these multi-core processors, but only if they expose enough parallelism to use the multiple cores within their collective memory bandwidth limitations [11].

1-4244-0910-1/07/$20.00 ©2007 IEEE.

The fundamental question for biomolecular simulation frameworks is whether multiple cores per processor can provide performance commensurate with initial expectations. The shared memory and I/O (network) bandwidth of multiple cores in a socket draws into question both how efficiently an application can use multiple cores and what methods provide the highest efficiency. In this preliminary study, we characterize computation, communication and memory efficiencies of a scalable bio-molecular simulation framework called LAMMPS [19] on a Cray XT3 system, which has recently been upgraded to dual-core Opteron processors. The early evaluation, dual-core Cray XT3 system at the Oak Ridge National Laboratory has over 10,000 processor cores with 54 teraFLOPS peak processing power. We identify that the performance gap between single and dual core execution times depends on the problem size as well as the size of the target system. In addition, we evaluated a number of processor affinity techniques for managing memory placement on multi-core systems. Our experiments on a stand-alone dual-core system show that an appropriate selection of MPI task and memory placement schemes can result in over 12% performance improvement for our target test cases.

The paper organization is as follows: In section 2, we provide a brief introduction to the bio-molecular simulations, the LAMMPS framework, our test cases, and the architecture and programming environment of our target Cray XT3 system. An overview of the related research efforts in the area of biomolecular simulation frameworks on high end supercomputers is presented in section 3. Performance evaluation and data collection experiments and results are presented in section 4. Conclusions and future plans are outlined in section 5.

2 Background

2.1 Molecular Dynamics Simulations

Numerous applications use molecular dynamics (MD) for biomolecular simulations. MD and related techniques can be defined as computer simulation methodology where the time evolution of a set of interacting particles is modeled by integrating the equation of motion. The underlying MD technique is based on the law of classical mechanics—most notably Newton's law, $F = ma$. The MD steps performed in LAMMPS or other MD engines consist of three calculations: determining energy of a system and forces on atoms centers, moving the atoms according to forces, and adjusting temperature and pressure. A typical bimolecular simulation contains atoms for solute, ions, and solvent molecules. The force on each atom is represented as the combination of the contribution from forces due to atoms that are chemically bonded to it and non-bond forces due to all other atoms.

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, functions, and thermodynamics of biological molecules and their complexes. The types of biological activity that have been investigated using MD simulations include protein folding, enzyme catalysis, conformational changes associated with bimolecular function, and molecular recognition of proteins, DNA, and biological membrane complexes. Biological molecules exhibit a wide range of time and length scales over which specific processes occur, hence the computational complexity of an MD simulation depends greatly on the time and length scales considered. With an explicit solvation model, typical system sizes of interest range from 20,000 atoms to more than 1 million atoms; if the solvation is implicit, sizes range from a few thousand atoms to about 100,000. The simulation time period can range from pico-seconds (10^{-12} seconds) to a few micro-seconds or longer ($>10^{-6}$ seconds) on contemporary platforms.

2.1.1 LAMMPS

Several commercial and open source MD software frameworks are in use by a large community of biologists, including AMBER, CHARMM, LAMMPS and NAMD. These packages differ in the form of their potential function and also in their force-field parameters. Some of them are able to use force-fields from other packages as well. AMBER provides a wide range of MD algorithms. The version of LAMMPS used (released 12 Apr 2006) in our evaluation does not use the energy minimization technique. A more recent version (released 1 Oct 2006) has introduced this functionality.

LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) [19] is a classical MD code. 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. For better efficiency on parallel systems, LAMMPS uses spatial-decomposition techniques by partition the simulation domain into small 3D sub-domains, one of which is assigned to each processor. Processors communicate and store "ghost" atom information for atoms that border their sub-domain. The version we used for our experiments is written in C++ and MPI. It is the only framework that is reported to scale to 64K Blue Gene/L processors. However, the scaling numbers are reported in the weak scaling mode, i.e. not for a fixed-size problem.

2.1.2 Test cases

The bio-molecular systems used for our experiments were designed to represent the variety of complexes routinely investigated by computational biologists.

The **HhaI** system is a model for protein-DNA complex (enzyme m5C-methyltransferase M. *Hha*I with its target DNA sequence), in explicit solvent and counter-ions to allow the system to be charge neutral. This model consists of 61,641 atoms with explicit treatment of solvent using TIP3P water model. AMBER's *tleap* module was used for system preparation and the AMBER parm98 force-field was used. The long range forces are calculated using PPPM (particle-particle-particle mesh) method, which is similar to the more commonly known particle mesh Ewald (PME) method. The system was simulated under periodic boundary conditions with a cuboid box with dimensions of 84 x 80 x 93 Å. For calculations of Lennard-Jones interactions inner and outer cut-offs of 10 and 11 Å respectively were used. For electrostatic interactions computed using the PPPM method a global cut-off of 11 Å was used for calculation of the direct sums. The system was equilibrated before benchmarking runs and the time-step is 1 femto-seconds (10^{-15} seconds) for the benchmarking runs.

The second system we considered the RuBisCO enzyme based on the crystal structure 1RCX. The **RAQ** system that is a model on the enzyme RuBisCO in explicit solvent similar and was prepared in a way similar to **HhaI** system, as described above. This model consists of 290,220 atoms with explicit treatment of solvent. The dimensions of the simulation box are 150 x 150 x 135 Å approximately. Cut-off values mentioned above for the **HhaI** system were used (both for Lennard-Jones and electrostatic interactions) and the time-step during MD runs is 1 femto-seconds.

We are currently considering a larger system for our performance evaluation and studies, which models cellulose degrading enzyme cellulase complex. The **JSC** test system represents cellulose fiber in crystalline Iβ form and cellulase CelE4 from *Thermomonospora fusca* (crystal structure code 1JS4), which shows endo/exo cellulase activity. The model was prepared in a way similar to above two systems using AMBER and parm98 (for protein, solvent and counter-ions) and GLYCAM (for cellulose) force-fields. The system consists of 311,459 atoms with explicit solvent. The time-step during MD runs was also 1 femto-seconds.

2.2 Target Dual-core Platforms

2.2.1 Dual-core Cray XT3 at ORNL

The XT3 installed at ORNL presently uses a dual-core Opteron processor node, or processing element (PE). The XT3 connects these processors with a custom interconnect managed by a Cray-designed Application-Specific Integrated Circuit (ASIC) called SeaStar.

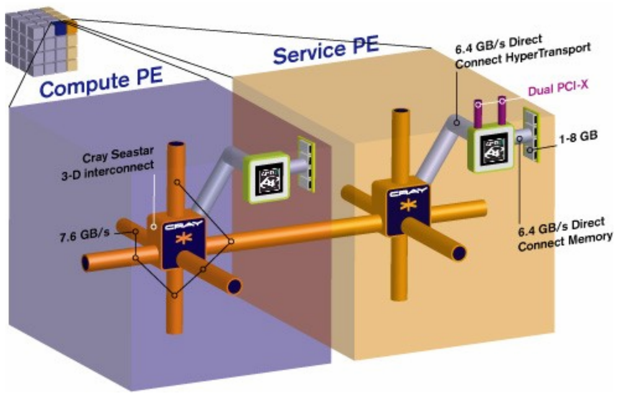
Compute PE
Service PE
6.4 GB/s Direct Connect HyperTransport
Dual PCI-X
1-8 GB
Cray Seastar 3-D interconnect
7.6 GB/s
6.4 GB/s Direct Connect Memory

Figure 1: Cray XT3 Architecture (*Image courtesy of Cray*).

Each XT3 PE has a 2.4 GHz dual-core AMD Opteron processor with its own dedicated memory and communication resource (see Figure 1). The peak performance per node is over 10 gigaFLOPS considering both cores. The XT3 has two types of PEs: compute PEs and service PEs. The compute PEs are optimized for application performance and run a lightweight operating system kernel called Catamount. The service PEs run SuSE Linux and are configured for I/O, login, or other system functions. The memory controller and the Hypertransport links are shared among the two cores of the dual-core Opteron processor. The Opteron processor is directly connected to the XT3 interconnect via a Cray SeaStar chip (Figure 1). This SeaStar chip is a routing and communications chip and acts as the gateway to the XT3's high-bandwidth, low-latency interconnect. In the XT3, the interconnect carries all message passing traffic as well as I/O traffic to the system's Lustre parallel file system.

The XT3 uses the Portals data movement layer for flexible, low-overhead inter-node communication. Portals provide connectionless, reliable, in-order delivery of messages between processes. Cray provides a Message Passing Interface (MPI) communication based on MPICH version 1.2 that uses Portals for data transfer. Details of the system can be found in [12].

2.2.2 Dual-core Test System with Linux

The Linux operating system provides a number of features to control processor and memory placements. In order to study impact of these options, we targeted a dual-core Opteron system with a standard Linux installation. This test system is a cluster of four nodes, each consisting of two dual core AMD 2.2 GHz Opteron 275 processors and 4 GBytes of shared memory, running Red Hat Enterprise Linux 4 update 3. The nodes are not connected by a high performance network, so we limit our experiments to a single node. All code was compiled using GNU v4 compilers.

A full evaluation of multi-core processors requires the use of *processor affinity*, the capability to specify that processes run only on a specific core or set of cores [11].

Each of our test systems runs the Linux operating system, which provides a few mechanisms for controlling processor affinity. On systems with Non-Uniform Memory Access (NUMA) architectures, such as our AMD Opteron test system, the `numactl` command controls processor affinity for a process and all of its children processes. It can also be used to control the operating system's memory page placement policy to ensure, for example, that a process' memory pages are always allocated in the memory that is directly attached to the socket that is running the process. Recent Linux kernels (version 2.6 and newer, and even some 2.4 versions) also contain system calls such as `sched_setaffinity` to set processor affinity. In our experiments, we used the `numactl` command to control processor and memory affinities.

3 Related Research

Qbox, is a first principle molecular dynamics (FPMD) code which has been shown to scale to relatively high number of processors [14]. FPMD differs from classical MD code, in its capability to combines a quantum mechanical description of electrons with a classical description of atomic nuclei. Qbox is a parallel implementation of the FPMD method designed specifically for large parallel platforms, including BlueGene/L. Simulations have been performed using up to 32,768 processors and performance measurements of strong scaling showed that an 86% parallel efficiency is obtained between 1k and 32k CPUs. Floating point operation counts measured with hardware performance counters show that 36% of peak performance is attained when using 4k CPUs. Qbox is limited to only a small number of atoms (~1000 atoms) due to the quantum mechanical description of electrons, while the biomolecular systems of interest can have 100,000 to > 1 million atoms.

NAMD is a C++ based parallel program, implemented using the Charm++ parallel programming system [16,18]. It uses object based decomposition 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. The developers of NAMD have claimed that it is one of the fastest and most scalable program for biomolecular simulations that is routinely used in published simulations. In a recent publication the NAMD developers described several techniques to scale NAMD to 8,192 processors of Blue Gene/L [18]. These include topology specific optimizations, new messaging protocols, load-balancing, and overlap of computation and communication. It was possible to achieve 1.2 TF of peak performance for cutoff simulations and 0.99 TF with PPPM method.

LAMMPS is a general purpose MD simulator and not restricted to biomolecules, has been shown to scale to 64K processors of Blue Gene/L. It has achieved parallel efficiency of >85% on simulating atomic fluid with Lennard-Jones potentials with 1-100 billion atoms [20].

In another related effort, our group is also working on exploring the performance of MD codes on multi-paradigm hardware including the FPGA (Field Programmable Gate Arrays) cards and the 256 gigaFLOPS IBM Cell system. We recently described a study where the MD code AMBER was ported on an SRC Computers, Inc platform using its high-level (Fortran 90), native programming interface [10]. Using the accurate performance models of our implementation, we demonstrate that the current generation FPGA devices can result in as high an order of 15x performance improvement over the current generation host processor for single-precision implementation. We are also investigating the mapping of LAMMPS on the Cell system, which is a heterogeneous multi-core processor [1].

4 Experiments and Results

We performed three set of experiments to collect and to analyze runtime and performance data. First, we collected runtime scaling data on the XT3 platforms in single and dual core mode and measured the impact of a runtime option called `small_pages`. Second, we instrumented the code with TAU (Tools and Analysis Utilities) [3] and gathered hardware counter and MPI performance data. Finally, on our dual-core system with Linux operating system, we measured the impact of different processor and memory affinity and placement schemes.

4.1 Impact on Code Scalability

Figure 2 and Figure 3 show performance of the **HhaI** and **RAQ** simulation experiments respectively on the dual-core XT3 system. SN represents the runs in the single-core mode and VN represents the run in the dual-core (virtual) mode in which two MPI tasks are assigned to a single processor. The performance of the simulation is represented in pico-seconds per day (psec/day). A higher psec/day is desirable. In addition the impact of small pages (w/ small_pages) option, which is selected at runtime is shown in the figures. The small pages option impacts the TLB usage and can therefore result in performance improvement and degradation depending on the characteristics of an application.

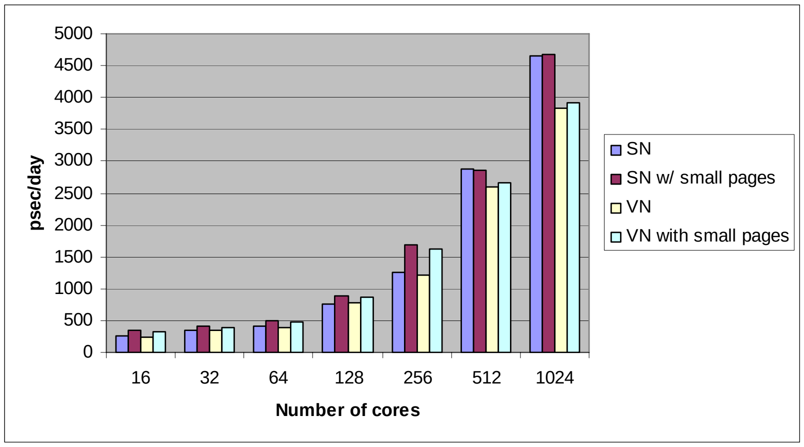

Figure 2: Performance of simulation with the HhaI system (61,641 atoms)

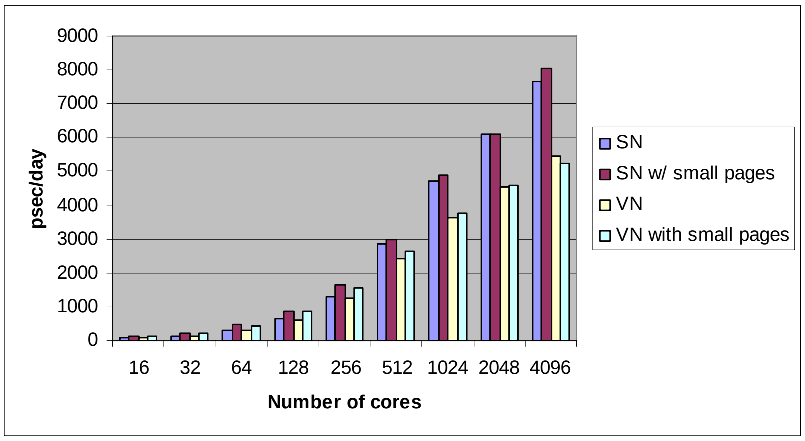

Figure 3: Performance of simulation with the RAQ system (290,220 atoms)

As shown in Figure 2 and Figure 3, performance in the SN mode is consistently higher than the dual-core performance. At the same time, the performance differences appear to be increasing with the increase in the number of MPI tasks. An interesting finding is that the performance of 2,048 tasks in the single core mode exceeds performance of 4,096 tasks in the dual core mode. Note here that no multi-core optimization techniques have been applied to the LAMMPS code for these performance runs. The aim of this study is to characterize performance of LAMMPS in dual-core mode and to subsequently identify factors that will enable performance improvements in multi-core execution modes.

Since we observed performance differences in the single (SN) and dual core (VN) modes of execution, we attempt to quantify the slowdown in the dual core mode with respect to the single core execution times. Figure 4 and Figure 5 show the percentage slowdown for the **HhaI** and **RAQ** tests respectively. Percentage slowdown is measured as:

$$\%slowdown = (Time_{VN} - Time_{SN}) * 100 \big/ Time_{SN}$$

The figures show that the rate of the slowdown increases with the increase in the number of MPI tasks, which points to a possible impact of two cores sharing Hypertransport resources for message-passing MPI operations. Also, the slowdown percentage is higher for larger systems (systems with large number of atoms). This could be an influence of higher memory and data transfer requirements on the shared memory controller. We also observe the impact of small pages on the performance differences, which is in most cases not consistent. We are currently investigating this behavior.

4.2 Analysis of Performance Data

We instrumented the LAMMPS application with TAU (Tools and Analysis Utilities) in order to collect profile data. We collected PAPI (Performance API) [2] hardware counter information and time spent in MPI operations for single-core and dual-core runs with 8, 32, 128 and 512 MPI tasks using the **HhaI** system.

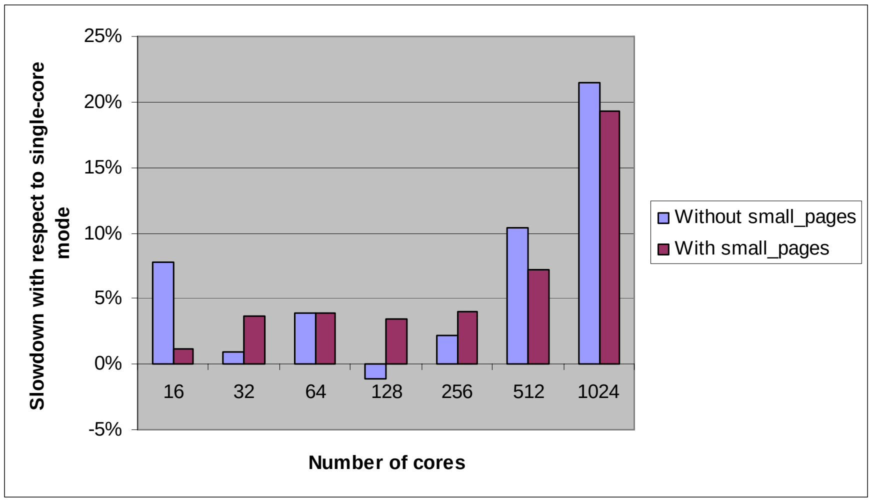

Figure 4: Percentage slowdown in the dual-core mode for the HhaI system (61,641 atoms)

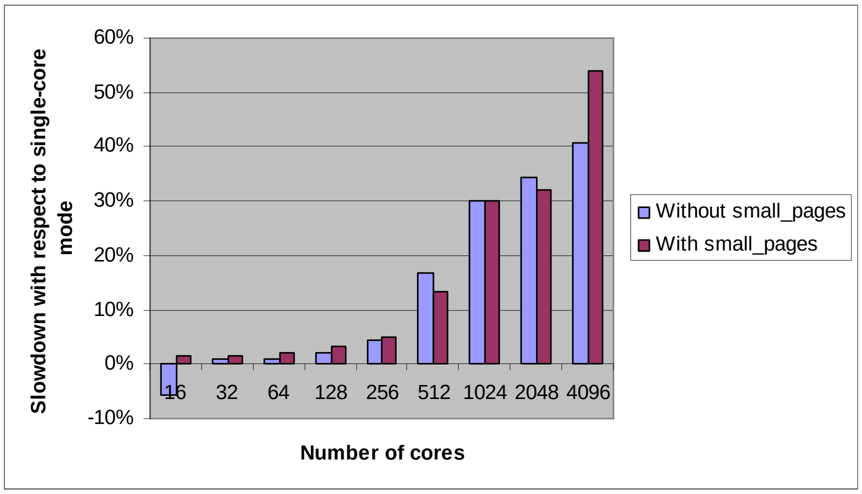

Figure 5: Percentage slowdown in the dual-core mode for the RAQ system (290,220 atoms)

Three PAPI counters are collected: total cycles, number of floating-point operations and level 1 data cache misses. We instrumented individual functions, which introduce some overheads. As a result, we did not observe significant differences in the three hardware counter values that we collected. For example, the number of total clock cycles in the single-core and dual-core execution modes is marginally different, while the runtimes are significantly different for benchmark runs.

Figure 6 shows the time spent in MPI operations with the **HhaI** experiment. We notice that the performance is comparable in the single-core and dual-core execution modes. However, there are consistent differences in performance for the `MPI_Bcast`, `MPI_Irecv` and `MPI_Send` operations. We also note that the runtime is distributed among different MPI operations. We conclude that the overall performance difference is due to an aggregated impact of various operations. In other words a single operation or hardware feature is not responsible for slowdown we noted in performance runs that are presented in the previous section.

4.3 Evaluation of Memory and Processor Affinity Schemes

In order to get an insight into the impact of different architectural features of multi-core devices on performance of our test cases using LAMMPS, we conducted experiments on a dual-core Opteron system that allows Linux non-uniform memory access options (`numactl`) on multi-core processors. Note that these options are not available on the XT3 system because it runs a light-weigth operating system.

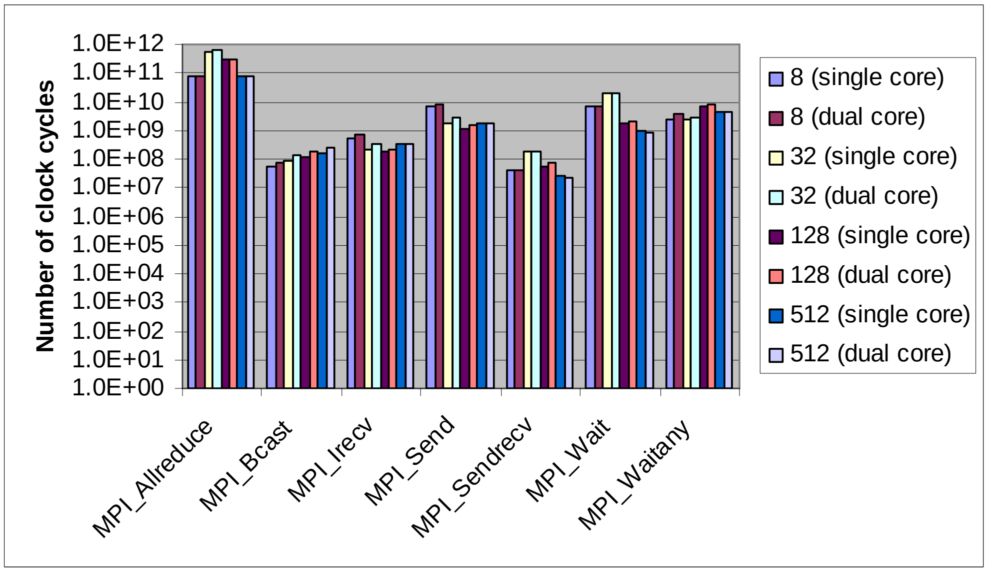

Figure 6: Number of cycles spent in MPI tasks for the HhaI system (61,641 atoms)

Name	Description
Default	Default (no numactl)
One MPI+Local Alloc	One MPI task per socket and local allocation policy
One MPI+Membind	One MPI task per socket with explicit memory binding per core
Two MPI+Local Alloc	Two MPI tasks per socket and local allocation policy
Two MPI+Membind	Two MPI tasks per socket with explicit memory binding per core
Interleave	Interleaved memory allocation

Table 1: `numactl` options used for experiments

MPI tasks	System	Default	One MPI + Local Alloc	One MPI + Membind	Two MPI + Local Alloc	Two MPI + Membind	Interleave
2	**HhaI**	121.574	121.84	125.247	124.585	124.799	125.505
4	**HhaI**	66.9479	—	—	66.8357	70.9545	68.9576
2	**RAQ**	787.533	787.686	878.659	797.81	802.674	830.71
4	**RAQ**	387.266	—	—	386.632	433.08	407.549

Table 2: Impact of numactl options on runtime (seconds)

Table 1 lists a combination of memory and processor affinity schemes that are used in running the LAMMPS experiments. In the default mode, no `numactl` option is used; the simulations are run using the standard `mpirun` command. The `cpubind` option is used to a control placement of MPI tasks onto the cores. Similarly, memory binding option is used to assign memories to cores. Additional memory placement schemes, `localalloc` and `interleave`, are also investigated for LAMMPS runs.

Table 2 lists the runtimes (in seconds) for the two test cases: **HhaI** and **RAQ**. Both experiments are run for 100 simulation time steps. We observe that both the memory and processor placement options impact performance of experiments. As we observed in the XT3 experiments, the slowdown increases with the increase in system size (number of atoms) and number of MPI tasks. For example, the difference between the fastest and the slowest runtime option can be about ~12%. In other words, we can control, monitor and train biomolecular simulation experiments at scale if some of these options are available on the future high-end supercomputing systems with multi-core processors.

5 Conclusions and future plans

In this study, we have described performance characterization of a large scale parallel molecular dynamics code, LAMMPS, for simulating biologically relevant systems. The focus of this study is to characterize the performance on multi-core processors as the upcoming generations of supercomputers including Cray's XT3 and XT4 as well as IBM's Blue Gene continue to pack increasingly more computing power through utilization of multi-core processors. The results on the XT3 system indicate that the slowdown in performance in the dual-core Opteron processor due to shared resources depend on the system size (number of atoms) as well as the number of MPI tasks. Our investigation of performance data reveals that a single architectural or application feature is not responsible for the slowdown. Instead it is cumulative effect of memory and message passing performance of the application. Furthermore, we demonstrate that an appropriate selection of processor and memory placement scheme can result in over 12% performance gain for bio-molecular simulation runs using the LAMMPS framework.

The future plans include assessment of options similar to `numactl` on high-end systems, study of additional hardware counter information, and investigation of performance features of upcoming quad-core systems. We also plan to work with biomolecular application developers to incorporate features that will make these applications aware of underlying multi-core platforms. In particular, we are currently investigating hybrid programming approaches, for instance, (OpenMP + MPI) for tightly-coupled, homogeneous multi-core processors. These techniques will enable us to scale MD calculations to a large-number of cores thereby allowing longer time-scales to be simulated on supercomputing systems with multi-core processors.

Acknowledgements

The authors would like to thank National Center for Computational Sciences (NCCS) for access to Cray XT3 and support (INCITE award). This research used NCCS at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

1. The Cell project at IBM research, http://www.research.ibm.com/cell/
2. Performance API (PAPI), http://icl.cs.utk.edu/papi/
3. Tuning and Analysis Utilities (TAU), http://www.cs.uoregon.edu/research/tau/
4. P. K. Agarwal, A. Geist, A. Gorin (2004), "Protein Dynamics and Enzymatic Catalysis: Investigating the Peptidyl-Prolyl cis/trans Isomerization Activity of Cyclophilin A", *Biochemistry*, **43**, 10605-10618.
5. P. K. Agarwal (2004), "Computational studies of the mechanism of cis/trans isomerization in HIV-1 catalyzed by cyclophilin A", *Proteins*, **56**, 449-463.
6. P. K. Agarwal (2005), "Role of Protein Dynamics in Reaction Rate Enhancement by Enzymes", *J. Am. Chem. Soc.*, **127**, 15248-15246.
7. P. K. Agarwal (2006), "Enzymes: An integrated view of structure, dynamics and function", *Microbial Cell Factories*, **5**:2.
8. P. K. Agarwal and S. R. Alam (2006), "Biomolecular simulations on petascale: promises and challenges", *J. Phys.: Conference Series (SciDAC 2006),* **46**, 327-333.
9. S. R. Alam, P. K. Agarwal, Al Giest and J. S. Vetter (2006), "Performance Characterization of Bio-molecular Simulations using Molecular Dynamics," *ACM Symposium on Principle and Practices of Parallel Programming (PPOPP).*
10. S. R. Alam, P. K. Agarwal, D. Caliga, M. C. Smith and J. S. Vetter (2007), "Achieving Supercomputer Performance for Biomolecular Simulations on Reconfigurable Systems," *IEEE Computer, Special Issue on High Performance Reconfigurable Computing* (to appear).
11. S. R. Alam, R. F. Barrett, J. A. Kuehn, P. C. Roth and J. S. Vetter (2006), "Characterization of Scientific Workloads on Systems with Multi-Core Processors," *IEEE International Symposium on Workload Characterization*.
12. S.R. Alam, R.F. Barrett, *et. al*. (2007), "An Evaluation of the ORNL Cray XT3," *J. High Performance Computing Applications* (to appear).
13. B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus (1983) "CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations", *J. Comput. Chem.*, **4**, 187-217.
14. F. Gygi, R.K. Yates, J. Lorenz, E.W. Draeger, F. Franchetti, C.W. Ueberhuber, B.R. de Supinski, S. Kral, J.A. Gunnels, J.C. Sexton. Large-Scale First-Principles Molecular Dynamics simulations on the BlueGene/L Platform using the Qbox code, *Supercomputing 05*
15. C. Huang, G. Almasi, L.V. Kale, S. Kumar, "Achieving strong scaling with NAMD on Blue Gene/L", *IEEE Parallel and Distributed Processing Symposium (IPDPS).*
16. L. V. Kal´e. The virtualization model of parallel programming: Runtime optimizations and the state of art. In *LACSI 2002*, Albuquerque, October 2002.
17. D. A. Pearlman, D. A. Case, J. W. Caldwell, W. S. Ross, T. E. Cheatham, S. Debolt, D. Ferguson, G. Seibel, and P. Kollman (1995), "AMBER, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules", *Comput. Phys. Commun.*, **91**, 1-41.
18. J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. V. Kale, and K. Schulten. "Scalable molecular dynamics with NAMD." *Journal of Computational Chemistry*, 26:1781-1802, 2005.
19. S. J. Plimpton (1995), "Fast Parallel Algorithms for Short-Range Molecular Dynamics", *J. Comp. Phys.*, 117, 1-19; http://www.cs.sandia.gov/~sjplimp/lammps.html
20. LAMMPS benchmarks, http://lammps.sandia.gov/bench.html