Published in ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2006.

Performance Characterization of Molecular Dynamics Techniques for Biomolecular Simulations

Sadaf R. Alam Jeffrey S. Vetter
Future Technologies Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN, USA 37831
{alamsr,vetter}@ornl.gov

Pratul K. Agarwal Al Geist
Computational Biology Institute
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN, USA 37831
{agarwalpk,gst}@ornl.gov

Abstract

Large-scale simulations and computational modeling using molecular dynamics (MD) continues to make significant impacts in the field of biology. It is well known that simulations of biological events at native time and length scales requires computing power several orders of magnitude beyond today's commonly available systems. Supercomputers, such as IBM Blue Gene/L and Cray XT3, will soon make tens to hundreds of teraFLOP/s of computing power available by utilizing thousands of processors. The popular algorithms and MD applications, however, were not initially designed to run on thousands of processors. In this paper, we present detailed investigations of the performance issues, which are crucial for improving the scalability of the MD-related algorithms and applications on massively parallel processing (MPP) architectures. Due to the varying characteristics of biological input problems, we study two prototypical biological complexes that use the MD algorithm: an explicit solvent and an implicit solvent. In particular, we study the AMBER application, which supports a variety of these types of input problems. For the explicit solvent problem, we focused on the particle mesh Ewald (PME) method for calculating the electrostatic energy, and for the implicit solvent model, we targeted the Generalized Born (GB) calculation. We uncovered and subsequently modified a limitation in AMBER that restricted the scaling beyond 128 processors. We collected performance data for experiments on up to 2048 Blue Gene/L and XT3 processors and subsequently identified that the scaling is largely limited by the underlying algorithmic characteristics and also by the implementation of the algorithms. Furthermore, we found that the input problem size of biological system is constrained by memory available per node. In conclusion, our results indicate that MD codes can significantly benefit from the current generation architectures with relatively modest optimization efforts. Nevertheless, the key for enabling scientific breakthroughs lies in exploiting the full potential of these new architectures.

Categories and Subject Descriptors J.3 [*Computer Applications*]: Life and Medical Sciences

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PPOPP'06 March 29–31, 2006, New York City, New York, USA
Copyright © 2006 ACM 1-59593-189-9/06/0003…$5.00.

General Terms Algorithms, Measurement, Performance

Keywords Performance analysis, Workload characterization, Computational biology, Molecular Dynamics algorithms

1. Introduction

The field of biology continues to benefit from use of computing resources [12]. Particularly in the last two decades, computing has enabled scientific advances through collection, storage, analysis and interpretation of genome sequence data from a variety of organisms [10]. Computational algorithms involving pattern matching and recognition lead to novel information and insights. Today computing is aiding experimental techniques to investigate complex systems. The power of scientific computing, however, lies in investigating areas that are beyond the reach of experimental biology. In the post genomic era, interest in using detailed atomistic modeling and simulations continues to grow as they provide new insights into the working of complex biological processes involving biomolecules. These biomolecular systems (including proteins, DNA, and RNA) work as very efficient machines at the molecular level.

There is wide spread interest in using parallel computers to perform large-scale biomolecular simulations and computational modeling using molecular dynamics (MD). MD-related techniques have emerged as vital tools for investigating structure, folding, dynamics and function of the biomolecules [4, 2]. Cellular processes involving biomolecules span multiple scales of time and length. Emerging evidence from computational simulations continues to indicate that the biomolecular structure, internal dynamics and function are interlinked. Figure 1, shows that the dynamics and function of biomolecules complexes spans multiple scales of time; the wide range of internal protein motions occur on 10^{-15} to $> 10^{-3}$ seconds, which are linked to a variety of protein functions (such as enzyme catalysis) on similar time-scales. Further, the dynamical motions also span over various length scales, ranging from few angstroms to tens of angstroms ($10^{-10} - 10^{-8}$m). Internal dynamics within biomolecules, however, are difficult to investigate experimentally as it spans $>$12 orders of magnitude in time scale. Computational simulations using MD techniques continue to provide vital information at a wide range of time-scales, leading to new scientific breakthroughs in the understanding of biomolecules [3]. However, the current simulations fall short by several orders of magnitude. The state-of-art simulations can only reach 10^{-8} seconds at best for a real biological system, while the desired time-scale is 10^{-3} seconds or higher.

The main challenge in simulating biologically relevant scales is that the computing resources required to simulate the native time

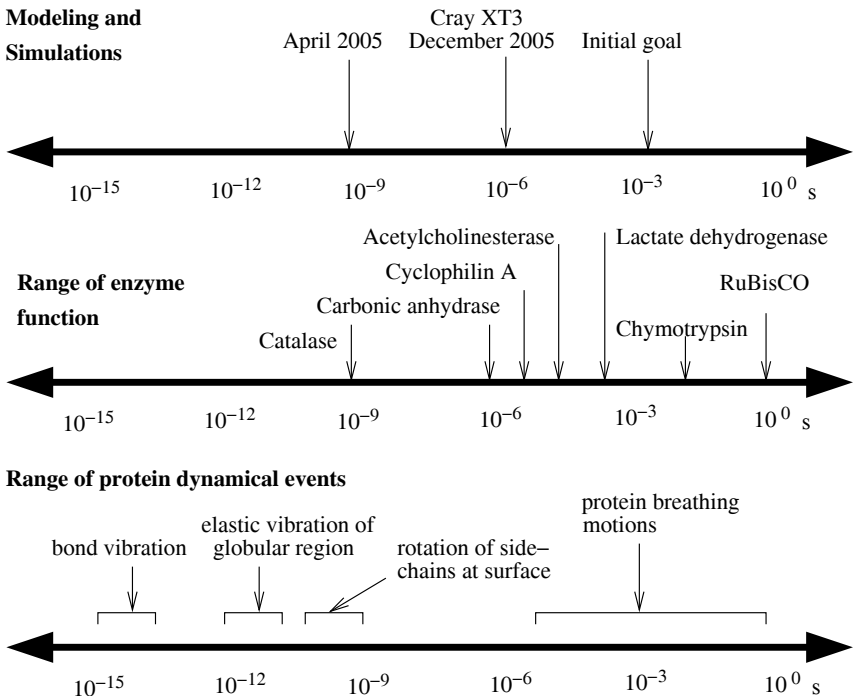

Figure 1. Computing needs for multi-scale modeling and simulations in biology. The structure, dynamics and function of biomolecules complexes spans multiple scales of time and length. The current simulations fall short by several orders of magnitude.

and length are considerably higher than what is currently available. For example, it has been suggested the computational requirements of simulating protein folding of a medium size protein can be as high as one petaFLOP/s for an entire year [19]. Despite the fact that supercomputers such as IBM Blue Gene/L [20]and Cray XT3 [22] will soon make tens of teraFLOP/s of computing power available to the biologists, progress in this domain is hindered by the lack of software scalability. Many MD applications were not designed with thousands of processors in mind.

As a first step toward improving the exploitation of the hardware resources of IBM Blue Gene/L and Cray XT3, we compare the computation complexity and analyze the scaling behavior of the two MD methods, particle mesh Ewald (PME) and Generalized Born (GB), on Blue Gene/L and XT3 systems. Our application-driven performance analysis of two early evaluation systems measure computation, communication and memory requirements of the main phases of MD calculations. There are two different techniques routinely used: energy minimization (EM) and time integration (commonly known as molecular dynamics). Further, the accurate modeling of biomolecules using molecular mechanics requires realistic models for the interaction of the solvent with the molecule. A variety of methods are used depending on explicit or implicit treatment of the solvent surrounding the biomolecules.

The layout of the paper is as follows: Section 2 provides a brief background to the MD calculations and introduces the software frameworks and test cases. Section 3 presents a brief overview of the two parallel systems: IBM Blue Gene/L and Cray XT3. Section 4 details the performance analysis of the test cases using the MD software framework AMBER, on the MPP systems. Scaling experiments and results are presented in section 5. The related research efforts in the field of high performance MD calculations are outlined in section 6. Finally, section 7 concludes the research findings and future research directions.

2. MD Simulations

In this section, we give a brief background of MD simulations and describe terms used in this paper. More details about MD techniques are available from other reviews and text [14, 15]. The underlying algorithms in popular MD packages or software frameworks including AMBER [16], CHARMM [6], GROMACS [11], LAMMPS [18]and NAMD [13] are similar. MD describes a system at atomistic level through the use of a mathematical function.

$$E = E_{bonds} + E_{angle} + E_{dihedral} + E_{non-bonded} \quad (1)$$

$$E_{non-bonded} = E_{electrostatic} + E_{vanderWaals} \quad (2)$$

This function, referred to as potential function, computes the molecular potential energy as a sum of energy terms that describe the deviation of bond lengths, bond angles and torsion angles away from equilibrium values, plus terms for non-bonded pairs of atoms describing van der Waals and electrostatic interactions. The set of parameters consisting of equilibrium bond lengths, bond angles, partial charge values, force constants and van der Waals parameters are collectively known as force-field. Different MD packages use slightly different mathematical expressions, and therefore, different constants for equation 1. The common force-fields in use today, have been developed by using high level quantum calculations and fitting to experimental data. The technique, known as energy minimization, is used to minimize the potential function. Lower energy states are more stable and are commonly investigated due to their role in biological processes. A MD simulation, on the other hand, computes the behavior of system as a function of time. It involves solving Newton's equation of motion, `F = ma`. Integration of Newton's laws of motion, using different integration algorithms, leads to atomic trajectories in space and time. The force on atoms are defined as the the negative gradient of the potential energy function. The EM technique is useful for obtaining a static picture for comparison between states or similar systems, while

MD provides information about the dynamic processes with the inclusion of temperature effects.

For common biological systems, the non-bonded interactions are computationally the most expensive operation. On parallel computers, non-bonded interactions also require significant off-node communication. The non-bonded term is computed as a sum of electrostatic and van der Waals interactions. It is desirable to model biomolecules in presence of solvent (water) as their function involves the solvated state. The explicit treatment of solvent adds significant computational expense due to the increase in number of atoms. Several techniques have been designed to decrease the computational expense. A common technique is the use of cutoff distance. The contributions from non-bonded interactions at long distances are relatively small; therefore, the full interaction is only computed for nearest neighbors. In Figure 2, the sphere represent the area for which a list of neighbors is maintained; this list is periodically updated as the particles move away from each other. Techniques, such as particle mesh Ewald (PME), method are used for calculation of long-range interactions [8]. The PME is a fast implementation of Ewald summation technique for calculating long range electrostatic interactions. PME method utilizes Fast Fourier Transform (FFT) and is commonly used when the simulation consists of explicit solvent particles. There is also significant interest in developing more rapid implicit solvation models because explicit treatment of water is computationally very demanding. The implicit solvation models treat the solvent as a continuous medium having the average properties of the real solvent, and surrounding the solute beginning at the van der Waals surface. A variety of continuum models have been described over the years including the Generalized Born (GB) model [21]. The GB approximation has been used with considerable success to efficiently evaluate hydration energies for small molecules. The computational complexity of an MD calculation depends on the underlying algorithm such as PME or GB. Moreover, the integration part is potentially expensive since it requires collective communication operations.

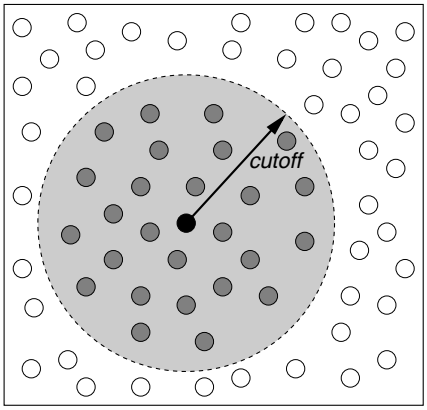

Figure 2. A cutoff radius is used to avoid direct evaluation of non-bond interactions beyond a specified distance. Non-bond neighbors list is used in MD simulations to reduce computational expense.

MD related methods are now routinely used to investigate the structure, dynamics 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, protein stability, conformational changes associated with bimolecular function, and molecular recognition of proteins, DNA, biological membrane complexes. As mentioned above, biological molecules exhibit a wide range of time and length scales over which specific processes occur. The computational complexity of a MD simulation depends on time and length scales. For realistic or higher-accuracy simulations not only the time and length scales requirements but also on the system sizes requirements are high. Typical system (including solvent) sizes of interest range from 20,000 atoms to more than 1 million atoms. For cases where implicit solvation model is used, the systems consists of a few thousands atoms to large complex systems over 100,000 atoms. The time period of simulation can range from pico-seconds to the a few micro-seconds or longer (see Figure 1).

2.1 Software frameworks

A number of commercial and open source software frameworks (also known as packages) for MD calculations are currently in use by a large community of biologists. These include AMBER, CHARMM, GROMACS, LAMMPS, NAMD, and others. These packages use slightly different forms of potential functions. Some of them have their own force-fields while others are able to use force fields of other packages. AMBER provides implementations of a range of MD algorithms including PME and GB methods that perform double-precision floating-point calculations. For use in large scale simulations, NAMD and LAMMPS have been reported to scale to up to a few thousand nodes. NAMD used a communication layer called CHARM++, which has been ported to the Blue Gene/L and XT3 systems. AMBER and LAMMPS, on the other hand, use the standard MPI communication library, which is available at the early evaluation stages on these two systems. The current version of LAMMPS, however, does not provide the energy minimization technique, which is routinely used in biological simulations. Further, it lacks many functionalities for simulations and analysis, which are provided with AMBER and CHARMM.

AMBER (version 8) consists of about 50 programs that can perform a diverse set of calculations for system preparation, EM, MD and analysis of results. AMBER's main module for EM and MD is known as *sander*, which stands for simulated annealing with NMR-derived energy restraints. We have used *sander* to investigate the performance characteristics of EM and MD techniques using the PME and GB method. We will discuss the details in the performance analysis section 4.

2.2 Test cases

The prototypical systems used for our detailed investigations were design to represent the variety of biomolecular complexes routinely investigated by computational biologists. The biomolecular systems are briefly described below.

- **HhaI:** This 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. This model consists of 61,641 atoms with explicit treatment of solvent. The long-range forces are calculated using PME method.
- **RuBisCO with generalized Born (RUB):** This system is a model on the enzyme Ribulose Bisphosphate Carboxylase/Oxygenase (RuBisCO) based on the crystal structure 1RCX. RuBisCO is an enzyme from the Calvin cycle and the product of its activity, phospoglycerate, is used in the carbon-fixing cycle. The model consists of 73,920 atoms, and GB method for implicit solvent.

In addition, other systems used for investigations of problem size scaling behavior contained: 143,784 atoms, 194,262 atoms, 290,220 atoms with explicit solvent conditions (PME method); an additional system with 18,056 atoms was used for calculations with implicit solvent model (GB method).

3. Supercomputing systems

Large-scale MD simulations of biological systems and processes are hugely expensive. The requirements for simulating a medium-sized biological system at multiple scales can be as high as 1 petaFLOP/s for an entire year. Hence, we considered scaling of our experiments on two teraFLOPS-scale systems, IBM Blue Gene/L and Cray XT3. These distributed memory, MPP systems have unconventional design characteristics and are expected to offer unprecedented levels of computing power. Here we present the architectural features of the two systems.

3.1 Blue Gene/L

Blue Gene/L is the current fastest supercomputing system [20]. We ran the experiments on a 1024-node Blue Gene/L system at Argonne National Laboratory. A single Blue Gene/L processing node consists of an Application Specific Integrated Circuit (ASIC). The ASIC contains the code execution logic, on-chip memory and communication logic. There are two PowerPC 440 cores per processing node: a computation and a communication processor core. The total processing power is 2.8 gigaFLOP/s per processor or 5.6 gigaFLOP/s per processing node. The computation processor handles the computation workload while the communication processor offloads communication tasks. There are three levels of on-chip caches with 32 KB, 2 KB and 4 MB capacities respectively. The L2 cache bandwidth is 5.5 gigaBytes/s and L3 bandwidth is total 22 gigaBytes/s. The total memory available to an application is 512 megaBytes and the off-chip memory bandwidth is 5.5 gigaBytes/s. Two communication networks are available for message passing operations: a three-dimensional torus network and a 2.8 gigaBytes/s tree network. The torus network supports point-to-point communication operations, while the tree network primarily supports collective communication operations. The bi-directional bandwidth is 1.4 gigabits/s in six torus directions.

Two execution modes are used by our MD applications namely the co-processor mode and virtual node mode. The co-processor mode is the default execution mode, in which one MPI task is assigned to a processing node. In the virtual-node mode, both PowerPC processors can run MPI tasks separately or offer twice the compute power. However, the memory capacity is halved in this mode and memory bandwidth is shared between two PowerPC processors. MD calculations typically have low memory usage requirements and our experiments effectively utilize the additional compute power as shown in section 5.

3.2 XT3

Cray XT3 system builds upon a single processor node, or processing element (PE) [22]. We have used an early evaluation system containing over 5000 processing nodes XT3 system at the Oak Ridge National Laboratory (ORNL) for the experiments. The XT3 uses a commodity microprocessor, the AMD Opteron, at its core, and connects these processors with customized interconnect based on an ASIC called SeaStar. Each PE has one Opteron processor with its own dedicated memory and communication resource. The XT3 has two types of PEs: compute PEs and service PEs. The compute PEs are optimized for application performance by running a lightweight operating system kernel called Catamount. In contrast, the service PEs run SuSE Linux and are configured for I/O, login, network, or system functions. The XT3 uses a blade approach for achieving high processor density per system cabinet.

The ORNL XT3 uses Opteron model 150 processors, which includes an Opteron core, integrated memory controller, three 16b 800 Mhz HyperTransport (HT) links, a L1 cache, and a L2 cache. The Opteron core has a 2.4 Ghz clock, three integer units, and one floating-point unit which is capable of two floating-point operations per cycle. Hence, the peak floating point rate of this processor is 4.8 gigaFLOP/s. The memory structure of the Opteron contains a 64KB 2-way associative L1 data cache, a 64KB 2-way associative L1 instruction cache, and a 1MB 16-way associative, unified L2 cache. The Opteron has 64b integer registers, 48b virtual addresses, 40b physical addresses, sixteen 64b integer registers, and sixteen 128b SSE/SSE2 registers. The memory controller data width is 128b. Each PE has 2 gigaBytes of memory. The peak memory bandwidth per processor is 6.4 gigaBytes/s.

Each Opteron processor is directly connected to the XT3 interconnect via a Cray SeaStar chip. This SeaStar chip is a routing and communications chip and it acts as the gateway to the XT3's high-bandwidth, low-latency interconnect. The PE is connected to the SeaStar chip with a 6.4 gigaBytes/s HT path. The router in SeaStar provides six high-speed network links to connect to six neighbors in the 3D torus/mesh topology. Each of the six links has a peak bandwidth of 7.6 gigaBytes/s. With this design, the Cray XT3 bypasses communication bottlenecks, such as the PCI bus. The interconnect carries all message passing traffic as well as I/O traffic to the global parallel file system.

4. Performance analysis

An application-driven approach was adopted for algorithm characterization and performance analysis of the biomolecular MD simulations that are implemented in AMBER. First, the main steps of calculations were identified and mapped to the phases of the software simulation. Second, we investigated the communication pattern and scaling within the communication phases for different methods and as a function of system size. This was useful not only in linking the scaling behavior to the underlying algorithms but also allowed us to precisely locate the performance bottlenecks in the code on the target architectures. Finally, we investigated the memory scaling behavior for the two algorithms. Vendor specific tools on the Blue Gene/L and XT3, and mpiP are used to collect performance data presented in this paper. Call graphs are generated using the `gprof` and the Cray PAT toolkit. The hardware performance data is collected via the PAPI hardware performance counter interface [7]. Memory requirements are collected using the IBM tools on Blue Gene/L. The MPI performance data is collected with mpiP, an MPI profiling tool [1]. mpiP data and the hardware counter values are measured on an IBM p690 cluster system, which is based on Power4 processors.

Figure 3 shows the control flow within a single step of an MD calculation along with the associated message passing MPI functions. The function calls map to the values in the equation 1, presented in section 2. The PME and GB methods are shown in dotted lines since only one is called in a given simulation. Furthermore, these are the most expensive calculations in a simulation time step; energies including the van der Waal interactions are calculated within this method.

4.1 Communication requirements

The FFT routines are called when explicit solvent simulations are being performed in PME methods. For the RuBisCO with GB simulations, a different methodology is used, which does not involve computing FFTs. The communication pattern is therefore significantly different. MPI collective communication operations are called in the GB calculations while the FFT calculations require a large number of point-to-point MPI messaging.

Table 1 lists the message sizes, count and pattern of MPI communication operations within a single simulation time step of an MD calculation per processor. Note than one simulation time step is equivalent to 1 femto-second (10^{-15} second) and 10^9 simulation steps will be needed to simulate a milli-second scale simulation. Although workload is evenly distributed, there can be slight vari-

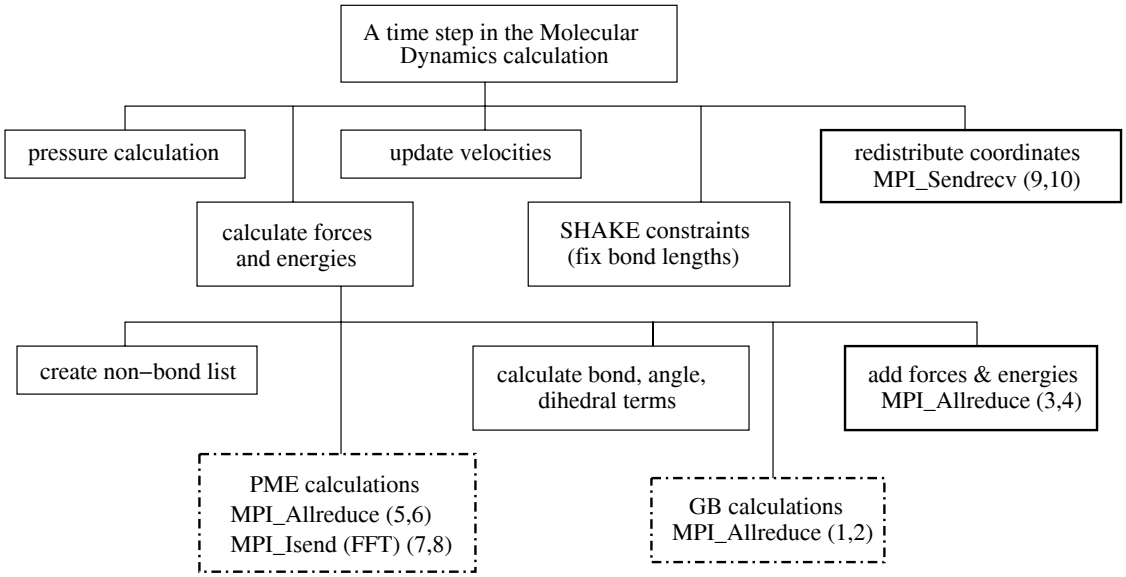

Figure 3. Control flow of the MD calculations in *sander*, which is a main program in AMBER

ation in message sizes on different processors. The table lists the maximum sizes for experiments with 8 MPI tasks.

	MPI call	HhaI (PME)	RuBisCO (GB)
1	MPI_Allreduce		($\sim 10^6$)
2	MPI_Allreduce		($\sim 10^6$)
3	MPI_Allreduce	280	280
4	MPI_Allreduce	24	24
5	MPI_Allreduce	936	
6	MPI_Allreduce	24	
7	MPI_Isend	$log_2 n$ x ($\sim 10^5$)	
8	MPI_Isend	$log_2 n$ x ($\sim 10^5$)	
9	MPI_Sendrecv	$log_2 n$ x ($\sim 10^5$)	$log_2 n$ x ($\sim 10^6$)
10	MPI_Sendrecv	$log_2 n$ x ($\sim 10^5$)	$log_2 n$ x ($\sim 10^6$)

Table 1. The message passing MPI communication behavior and maximum message sizes for HhaI and RuBisCO with GB simulations. n represents the number of MPI tasks. The MPI profile is gathered on an IBM p690 cluster system using the mpiP profiling tool.

In RuBisCO with GB, the message sizes remain constant; only the message count for the coordinate distribution function increases with the number of MPI tasks. For HhaI, the message sizes and counts for collective operations remain constant. The message counts for point-to-point messages increase while the message sizes decrease by increasing the number of MPI tasks. Moreover, the message sizes for the coordinate distribution function, non-bond methods and force field integration depend on the number of atoms in the system. In other words, these message sizes increase with an increase in the system size or total number of atoms in a system.

A first order analysis of the communication behavior shows that the point-to-point communication in the two test cases could be bandwidth bound. Hence, the high bandwidth networks of Blue Gene/L and Cray XT3 should be able to exhibit good scaling behavior. An analysis of the collective communication operations in HhaI and RuBisCO with GB suggest that the small collective operations could also be latency bound, especially with large number of MPI tasks. In HhaI, there are four MPI_Allreduce calls per iteration and two with 24 Byte messages. The point-to-point MPI latencies of the XT3 network at the early evaluation stages were higher than the Blue Gene/L MPI latencies. Thus, the latency bound HhaI simulation are likely to be effected on large number of XT3 processors. In addition, the MPI_Sendrecv for coordinate distribution subroutine also require a large number of point-to-point message passing operation2. Both Blue Gene/L and XT3 have a 3D torus network topology, therefore, the cost of the coordinate distribution operations are expected to be similar.

A potential performance and scaling bottleneck for RuBisCO with GB could be the collective operation with 5.9 x 10^5 Bytes. Typically, the collective communication networks are not optimized for large message sizes. The Blue Gene/L system has a separate tree network for collective communication operations in addition to the 3D torus network. We ran Pallas MPI benchmarks version 2.2 (now offered by Intel as the Intel MPI Benchmarks) to estimate the MPI_Allreduce performance of the Blue Gene/L and XT3 networks. We observed that for message sizes greater than 10^3 Bytes, there is a significant increase in the runtime on both systems. With a large number of MPI tasks, there is a linear increase in time to compute the reduction operation across all processing nodes. Hence, we anticipate that the GB method is unlikely to scale to large number of processors if the computation to communication ratio is small.

4.2 Computation requirements

In order to investigate the computation requirements of the PME and GB methods, we quantified the number of floating-point operations, load-store operations and memory per processor node. Figure 4 shows the maximum number of floating-point operations per processor. On a logarithmic scale, it is evident that the GB method is more compute-intensive compared to the PME method. The behavior of the load-store operation count, figure 5 is similar. Thus, we predict that the GB algorithms are likely to scale to large number of processors because of the high local computation requirements.

We also compared the local memory requirement per processor for the two methods. Figure 6 shows that the PME memory requirements scale to about 32 processors. Upon inspecting the code, we identified that a couple of large arrays, particularly the non-bonded neighbors list, is not distributed among the processors. Instead, it

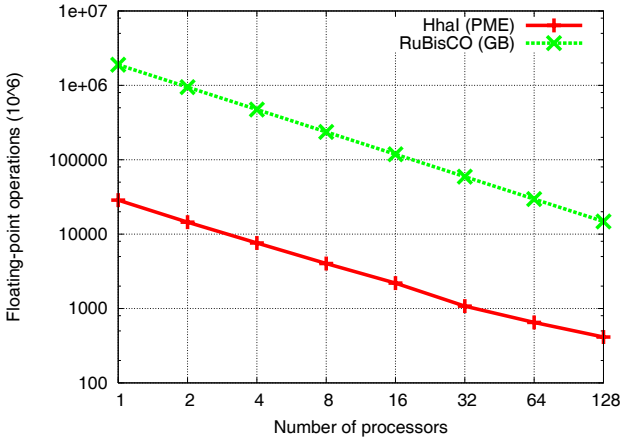

Figure 4. Avergae floating-point operation count per MPI task for simulation runs with increasing number of processors (20 simulation time steps). The floating-point operation count is measured by using the PAPI hardware counter API on an IBM p690 cluster.

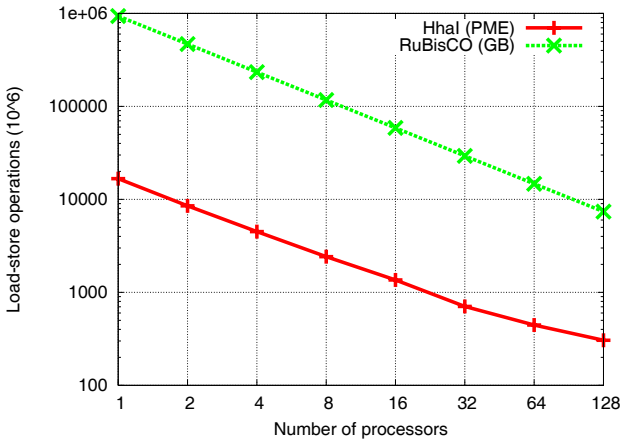

Figure 5. Average load and store operation count per MPI task for simulation runs with increasing number of processors (20 simulation time steps). The load-store operation count measurements are taken using the PAPI API on an IBM p690 cluster system.

is replicated to all the processors. The memory requirements of the GB method, on the other hand, increase slightly from uni-processor runs to multiple processor runs, primarily because of the creation of array lists for global reduction operations.

5. Experiments and results

We performed investigations on IBM Blue Gene/L and Cray XT3 system on up to 2048 processors using the two prototypical test cases. AMBER 8 has a hard-coded limit of a maximum of 128 processors. This limit had been enforced by not only as a constant declaration but also as fixed size arrays, one for each processor or MPI task. We therefore modified sections of code to scale the application to 4096 processors. In addition, the simulation experiments rely on input files that contain list of atoms and force-field parameters and their coordinates. Minor modifications were also needed for I/O capability of AMBER to allow it to read large files. Furthermore, machine-specific modifications include the `massv` library on Blue Gene/L along with dual floating-point optimization `440d`. On XT3, the `fastsse` option has been used.

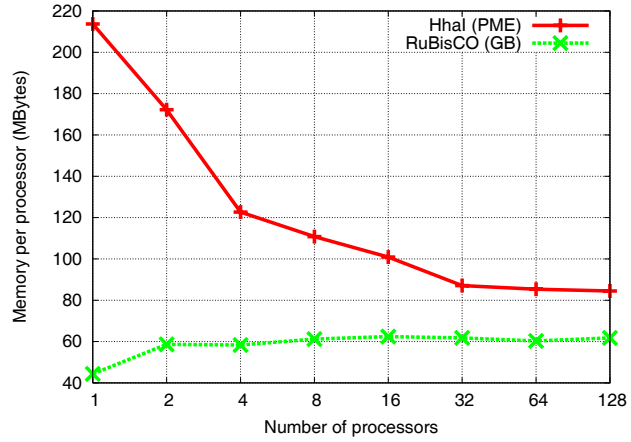

Figure 6. Average memory requirement per processor for runs with increasing number of processors. The average memory requirements is calculated on the Blue Gene/L system using the mpitrace_f profiling library.

5.1 Test case with explicit solvent with PME method: HhaI

The results of scaling of HhaI experiments on Blue Gene/L are listed in Table 2. The speed up and the parallel efficiency are calculated as:

$$SpeedUp = \frac{time_{serial}}{time_{parallel}} \quad (3)$$

$$Efficiency = \frac{SpeedUp_{NumberOfTasks}}{NumberOfTasks} \quad (4)$$

The HhaI system simulations show no scaling beyond 128 processing nodes. Table 3 breakdowns the percentage of time spent in the calculation phases. As we predicted in the last section, the cost of collective operations are dominating the calculations as we include very large number of MPI tasks. The FFT communication time is also increasing in the non-bond calculation time as shown in table 3. At the same time, the cost of the code blocks performing computation only exhibit a near linear behavior. This led us to conclude that the workload is evenly distributed among the processing nodes. AMBER outputs the run times for individual processing nodes, which even for 2048 processor run, show no more than 5% variation.

The XT3 performance results are listed in Table 4 and the breakdown of the execution time is shown in Table 5. The XT3 systems shows a higher speed up compared to the Blue Gene/L system primarily due to high speed Opteron processor. However, the communication costs on XT3, particularly the FFT communication cost limited the scaling of the code beyond 128 processors. The cost of collective operation is significant in both cases and is one of the main limiting factor. Note that the results on the XT3 have been collected on the early evaluation stages of the system. It is expected that the FFT communication cost will be significantly reduced after the communication fabric upgrade.

Figure 7 shows the simulation performance in pico-seconds per simulation day (psec/day) for the HhaI experiment. The desired metric is $> 10^3$ for simulating native timescale simulations. We also compared the run time with an unmodified version of amber on an IBM Power4 cluster with 2.6 gigaFLOP/s performance per

Processors	1	2	4	8	16	32	64	128	256	512	1024
Speed up	1.00	1.86	3.31	5.36	8.27	11.05	12.73	14.02	13.98	13.15	10.54
Efficiency	1.00	0.93	0.83	0.67	0.52	0.35	0.20	0.11	0.05	0.03	0.01

Table 2. HhaI performance on Blue Gene/L

Processors	1	2	4	8	16	32	64	128	256	512	1024
Non-bond (NB) force	97.56	96.85	95.98	94.02	90.76	85.89	75.58	65.49	56.02	49.23	31.03
Bond,Angle,Dihedral	0.98	0.96	0.91	0.83	0.75	0.62	0.42	0.27	0.28	0.00	0.00
Collect & add forces	0.00	0.45	1.19	2.65	5.22	8.99	17.05	23.64	29.13	29.34	32.21
Distribute coordinate	0.00	0.19	0.33	0.68	1.03	1.86	3.79	6.52	9.52	15.56	32.55
FFT (% NB time)	4.75	5.31	5.63	5.80	5.96	6.50	11.14	9.13	14.50	18.13	27.17

Table 3. Percentage of execution time (HhaI on Blue Gene/L)

Processors	1	1.95	4	8	16	32	64	128	256	512	1024
Speed up	1.00	2.00	3.49	6.28	9.8	13.42	19.33	22.09	21.37	16.57	12.54
Efficiency	1.00	0.97	0.87	0.78	0.61	0.42	0.30	0.17	0.08	0.03	0.01

Table 4. HhaI performance on XT3

Processors	1	2	4	8	16	32	64	128	256	512	1024
Non-bond (NB) force	96.83	96.70	96.02	95.27	92.52	88.63	83.19	78.12	73.91	73.98	68.42
Bond,Angle,Dihedral	0.45	0.38	0.41	0.38	0.34	0.25	0.19	0.11	0.07	0.03	0.01
Collect & add forces	0.00	0.25	0.56	1.13	3.12	5.71	9.96	13.95	16.25	14.84	16.93
Distribute coordinate	0.00	0.12	0.22	0.50	1.48	2.49	4.42	6.07	7.96	9.29	12.95
FFT (% NB time)	3.60	5.42	9.91	15.87	24.24	39.37	40.08	48.96	61.23	75.84	82.79

Table 5. Percentage of execution time (HhaI on XT3)

Processors	1	2	4	8	16	32	64	128	256	512	1024	2048
Speed up	1.00	2.00	3.97	7.89	15.53	30.13	56.78	101.72	158.23	243.26	271.04	279.31
Efficiency	1.00	1.00	0.99	0.99	0.97	0.94	0.89	0.79	0.62	0.48	0.26	0.14

Table 6. RuBisCO with GB performance on Blue Gene/L

Processors	1	2	4	8	16	32	64	128	256	512	1024	2048
Non-bond (GB method)	99.75	99.66	99.59	99.52	99.39	99.16	98.56	97.28	94.22	87.67	68.32	37.06
Bond,Angle,Dihedral	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.19	0.21	0.19	0.14	0.14
Collect & add forces	0.00	0.01	0.02	0.05	0.12	0.25	0.68	1.44	3.28	6.64	17.54	30.11
Distribute coordinate	0.00	0.00	0.01	0.01	0.03	0.06	0.12	0.35	1.04	3.32	10.04	29.43

Table 7. Percentage of execution time (RuBisCO with GB on Blue Gene/L)

Processors	1	2	4	8	16	32	64	128	256	512	1024	2048
Speed up	1.00	2.00	3.98	7.95	15.81	31.29	63.91	124.50	386.26	554.69	493.60	312.23
Efficiency	1.00	1.00	1.00	0.99	0.99	0.98	0.99	0.97	0.89	0.75	0.54	0.24

Table 8. RuBisCO with GB performance on XT3

Processors	1	2	4	8	16	32	64	128	256	512	1024	2048
Non-bond (GB method)	99.82	99.79	99.76	99.72	99.56	99.28	98.01	97.02	95.06	88.35	72.16	34.65
Bond,Angle,Dihedral	0.16	0.16	0.15	0.15	0.15	0.15	0.15	0.15	0.14	0.14	0.09	0.04
Collect & add forces	0.00	0.01	0.02	0.03	0.12	0.27	0.58	0.99	2.91	6.65	14.96	42.35
Distribute coordinate	0.00	0.00	0.01	0.02	0.05	0.12	0.23	0.53	1.34	3.76	15.57	21.06

Table 9. Percentage of execution time (RuBisCO with GB on XT3)

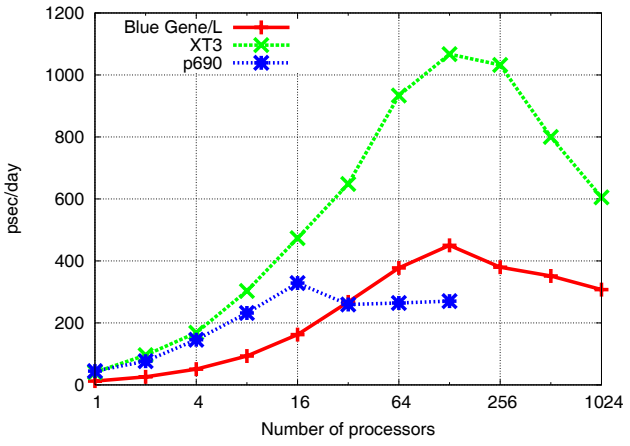

Figure 7. Parallel efficiency of HhaI simulations. An unmodified version of AMBER8 was used on the IBM Power4 (SMP cluster. The modified version of AMBER8 was used for experiments on XT3 and Blue Gene/L. Performance is measured in psec/day or number of pico-seconds that can be simulated in a single day.

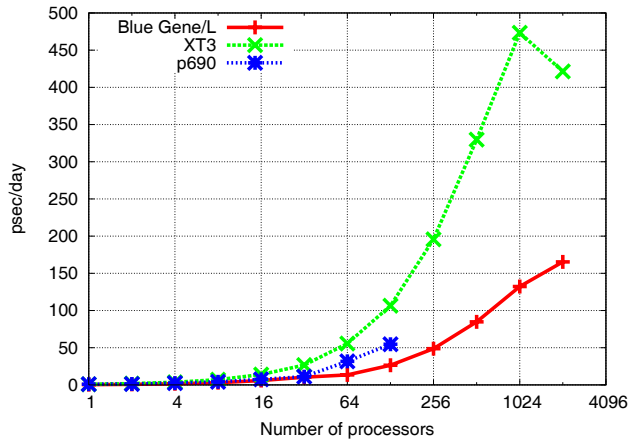

Figure 8. Parallel efficiency of RUB with GB simulations on IBM Power4 (SMP cluster) with an unmodified version of AMBER8 and on XT3 and Blue Gene/L with modifications. A large value of psec/day (pico-second per day) is required for biomolecular MD simulations.

processor. Although the XT3 has a clear advantage for having a 4.8 gigaFLOP/s processor, the scaling characteristics of the HhaI algorithm limit the parallel performance of the simulation. Nevertheless, over 10^3 psec/day for a simulation with over 61K atoms with explicit solvent is a significant contribution for the computational biology simulations.

5.2 Test case with GB implicit solvation method: RUB

As compared to HhaI, which can be simulated on contemporary clusters, RuBisCO is a relatively larger system with higher computation requirements. Moreover, RUB uses a different algorithm for calculating solvation effects, that does not require a large number of point-to-point communication operations. Instead, a small number of collective operations are required for each time step. The message sizes for this collective operation are listed in Table 1.

The performance and scaling results of RuBisCO with GB on Blue Gene/L are presented in Table 6. It scales to 2048 processors. Taking into account the contribution of different phases of the calculation that are listed in Table 7 and measured cost of simulation phases listed in Table 7 and 9, it is evident that the total runtime is dominated by the phases of calculation that perform communication operations. The MPI_Sendrecv cost is higher for RUB simulations as compared to the HhaI simulation because of relatively large number of atoms. As a result, not only the cost of $log_2 n$ MPI_Sendrecv operations for the global distribution function but also the MPI_Allreduce operations in the GB function is higher. On 2048 processors run, over 60% of time is spent in collecting, adding and distributing force arrays.

Similar behavior is shown for the RUB experiments on XT3 in Table 8 and Table 9. Like HhaI runs, the XT3 experiments show higher speed ups due to the 4.8 GHz Opteron processors and the relatively high bandwidth network. Nevertheless, the collective communication operations are dominant on large processor runs. A metric for achieved simulation time scale (psec/day) on different parallel platforms is shown in Figure 8. The performance advantage of parallelism beyond 128 processing nodes has been demonstrated in the RUB experiments. The Blue Gene/L system has a lower performance per processor as compared to a Power4 processor in the IBM p690 cluster. However, on 2048 nodes, the Blue Gene/L provides over a 150 psec/day, whereas the peak at 128 p690 processors is around 50 psec/day. We managed to simulate a 73K atoms system for close to 500 psec/day on the XT3 system. We expect the scaling will improve as the point-to-point MPI latency on XT3 decreases. Together, with a dual-core Opteron node, we envisage that the XT3 could bring us close to our target of around half a million atoms with 1,000 pico-second per day.

5.3 Problem size scaling

As mentioned earlier, many biological systems demand models with a large number of atoms for capturing different aspects of the system. In the previous sections, we have discussed the scaling behavior, communication patterns and computation requirements of the two methods for a fixed number of atoms. We have also explored the scaling behavior of the two algorithms as a function of number of atoms. The scaling of the PME simulation, as shown in Figure 9, is largely limited by the underlying algorithm and implementation; these simulations do not scale beyond 128 processors. Figure 10 shows the paralleling efficiency of two GB simulations, which scales to 1,024 processors.

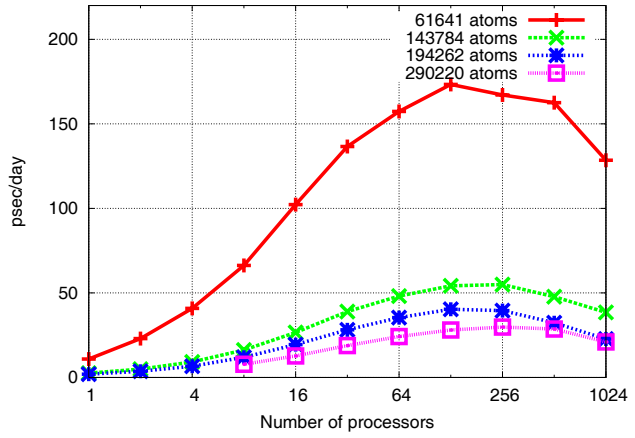

Figure 9. Scaling behavior of PME algorithm-based simulations as a function of number of atoms (simulations experiments are conducted on the Blue Gene/L system).

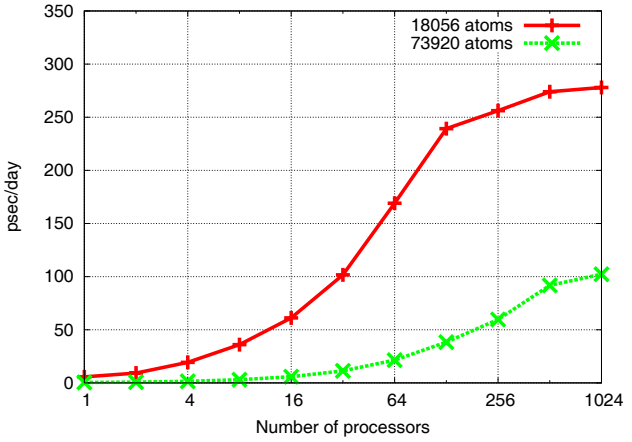

Figure 10. Scaling behavior of GB algorithm-based simulations as a function of number of atoms. The runtime measurements are taken on the Blue Gene/L system.

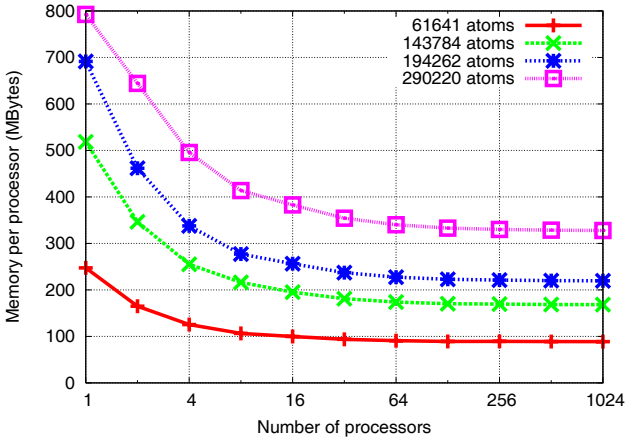

Figure 11. Per processor memory requirements and scaling as a function of number of atoms. IBM mpitrace_f library, which is available on the Blue Gene/L system, was used to collect the memory requirement per processing node.

The results confirm our prediction that, by simply increasing the problem size, we cannot overcome the scaling limitations; these are inherent properties of the algorithm and implementation. For instance, the PME algorithm's scaling characteristics do not change by increasing the number of atoms or workload requirements per processor as shown in Figure 9. Similar scaling behavior is observed on the XT3 system. Alternatively, the GB algorithm has a distinctive algorithmic and communication characteristics; hence, scaling with a relatively small system, 18,056 atoms, also scale to up to 1,024 processors (Figure 10).

5.4 Memory requirements

The available per processor memory on Blue Gene/L and Cray XT3 is smaller than what is available with commodity cluster computers. In the virtual node mode on Blue Gene/L, the available memory is less than 256 MB per processor. Therefore, we investigated the requirement of memory with increasing problem sizes. Figure 11 shows the memory usage as a function of processor count for several systems investigated in this study. For explicit solvent cases, our results indicate that after about 64 processors the required memory per processors does not change significantly. This is expected to be a limiting factor in simulating larger biological systems with greater than 500,000 atoms. We have extrapolated the curve to compute memory requirements for larger systems (see Figure 12). On Blue Gene/L, it is expected that the biggest system which can be simulated is about 400,000 atoms. In the virtual node more this is expected to be near 220,000 atoms. On the XT3, the user processors are allowed only about one megaBytes of memory, which will restrict the size of system to be no more than about 600,000–700,000 atoms. With further investigations, we have found that the replication of non-bonded neighbors list to every processors leads to higher memory requirements. Strategies for enabling simulations for larger systems may include reduction in memory requirement per processor by distributing the non-bonded list among processors.

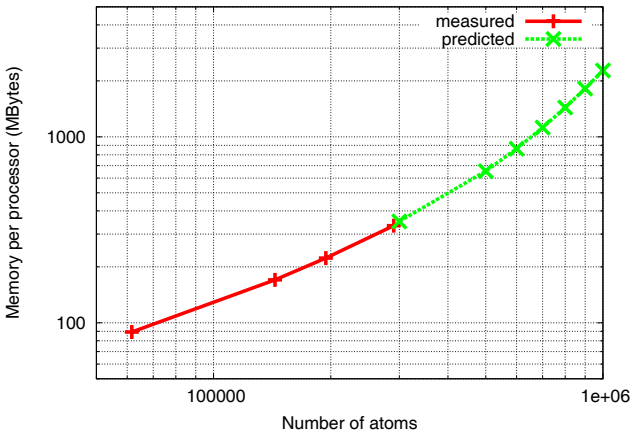

Figure 12. Increase in the memory requirements as a function of number of atoms for 128 processors run. A cubic curve was used for extrapolation.

6. Related research

Scaling MD framework on supercomputers is of considerable interest, as is reflected by number of groups who are working on related research projects. At IBM, a new framework for molecular dynamics on the IBM Blue Gene/L is being developed; it is known as Blue Matter [9]. Initial results have indicated that the Blue Matter can take significant advantage of the hardware to scale small systems up to 512-1024 processors. Blue Matter, however, is closely linked to the system performance of the IBM Blue Gene series of systems. Almasi *et. al.* [5] provided a detailed mapping of a molecular dynamics application on Blue Gene multi-threaded cellular architecture using analytical models and simulation techniques. The Blue Gene/L system and the Cray XT3 machines do not support multi-threading. Therefore, currently, the applicability of Almasi *et. al.*'s performance optimization techniques for the molecular dynamics algorithm using multi-threading appears to be limited on distributed-memory systems like Blue Gene/L and XT3.

NAMD is a software framework that is reported to scale to many thousands of processors [13]. It has been developed by a research group at the University of Illinois using optimized communication patterns to improve scaling of MD simulations. NAMD's underlying communication library, CHARM++, is ported to XT3 and Blue Gene/L using the MPI communication interface. A study [23] based on the Blue Gene/L simulators, BigSim and BignetSim, suggested that the PME algorithm implementation in NAMD and CHARM++ will benefit from the extra tree network,

which has been optimized for collective communication operations. CHARM++ is reported to achieve higher performance and scaling when it is implemented using the low-level communication API [17]. Hence, further investigations are needed to identify performance characteristics and scaling of the NAMD software on the Blue Gene/L and XT3 systems. Another software framework, LAMMPS from Sandia National Laboratory is reported to scale very well in large number of processors [18]. As mentioned earlier, a number of MD algorithms including GB that are implemented in AMBER are not currently supported by the LAMMPS framework. Hence, we were not able to compare the experiments conducted using AMBER with the equivalent LAMMPS implementation. Although both NAMD and LAMMPS are capable of using popular force-fields, there is wide interest in community to continue using AMBER, due to the diverse set of functionalities it provides.

7. Conclusions

In order to achieve higher and sustainable parallel efficiencies for large scale biomolecular simulations using molecular dynamics, we are exploring algorithmic and implementation alternatives by targeting factors that limit the scalability of MD calculations. In this paper, we presented detailed profiling information of two commonly used MD methods (PME and GB) and investigated their scaling behavior on contemporary MPP systems. After removing the limits in the software framework, we were able to scale the GB method to up to 2048 processors. Through the application-driven analysis, we have identified the scaling bottlenecks in these two MD methods. We observed that on Blue Gene/L and XT3 systems, the available memory per processor limits the size of biological systems that can be simulated and suggest that the changes in parallel implementation, particularly in the array distribution scheme, can overcome this limitation. Using prototypical test systems, we show that it is possible to simulate over 1,000 psec/day for a protein-DNA complex with over 60,000 atoms. This is a remarkable improvement over the state-of-art simulations as previously reported. Further improvements will bring simulations closer to native biological time-scales. Through efficient exploitation of the hardware resources, we envisage that the teraFLOP/s scale computing power of Blue Gene/L and Cray XT3 will enable simulations of larger systems on longer time-scales leading to scientific breakthroughs.

Acknowledgments

The authors would like to thank Buddy Bland and other staff of the National Center for Computational Sciences (NCCS), Carlos Sosa of IBM, and MCS division at Argonne National Laboratory (for access to Blue Gene/L). This research used resources of the Center for Computational Sciences at 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] mpiP: Lightweight, Scalable MPI Profiling. *http://www.llnl.gov/CASC/mpip/*.

[2] P. K. Agarwal. Enzymes: An integrated view of structure, dynamics and function. *Microbial Cell Factories*, 5:2, 2006.

[3] P. K. Agarwal. Role of Protein Dynamics in Reaction Rate Enhancement by Enzymes. *Journal of American Chemical Society*, 2005.

[4] P. K. Agarwal, A. Geist, and A. Gorin. Protein Dynamics and Enzymatic Catalysis: Investigating the Peptidyl-Prolyl cis/trans Isomerization Activity of Cyclophilin A. *Biochemistry*, 43, 2004.

[5] G. S. Almasi, C. Cascaval, J. G. Castanos, M. Denneau, W. E. Donath, M. Eleftheriou, M. Giampapa amd H. Ho, D. Lieber, J. E. Moreira, D. M. Newns, M. Snir, and H. S. Warren Jr. Demonstrating the Scalability of a Molecular Dynamics Application on a Petaflops Computer. *International Journal of Parallel Programming*, 30(4), 2002.

[6] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, and M. Karplus. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, 4, 1983.

[7] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci. A portable programming interface for performance evaluation on modern processors. *The International Journal of High Performance Computing Applications*, 2000.

[8] T. Darden, D. York, and L. Pederson. Particle mesh Ewald–an Nlog(N) method for Ewald sums in large systems. *Journal of Chemical Physics*, 98, 1993.

[9] B. G. Fitch, R. S. Germain, M. P. Mendell, J. Pitera, M. Pitman, A. Rayshubskiy, Y. Y. Sham, F. Suits, W. C. 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, 2003.

[10] N. Goodman. Biological data becomes computer literate: new advances in bioinformatics. *Curr. Opin. Biotechnol.*, 13(1), 2002.

[11] GROMACS. http://www.gromacs.org/.

[12] K. Hiroaki. Computational systems biology. *Nature*, 420, 2002.

[13] L. Kale, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and title = "NAMD2 : Greater scalability for parallel molecular dynamics K. Schulten". NAMD2: Greater scalability for parallel molecular dynamics. *Journal of Computational Physics*, 151, 1999.

[14] M. Karplus and G.A. Petsko. Molecular dynamics simulations in biology. *Nature*, 347, 1990.

[15] A. R. Leach. *Molecular Modelling: Principles and Applications*. Prentice Hall, 2001.

[16] D. A. Pearlman, D.A. Case, J.W. Caldwell, W.S. Ross, III T.E. Cheatham, S. DeBolt, D. Ferguson, G. Seibel, and P.Kollman. AMBER, a package of computer programs for applying molecular mechanics, normalmode analysis, molecular dynamics and free energy calculationsto simulate the structural and energetic properties of molecules. *Computer Physics Communication*, 91, 1995.

[17] J. C. Phillips, G. Zheng, and L. Kale. NAMD: Biomolecular simulation on thousands of processors. In *Supercomputing, 2002*.

[18] S. J. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117, 1995.

[19] IBM Blue Gene team. Blue Gene: A vision for protein science using a petaflop supercomputer. *IBM Systems Jornal*, 40, 2001.

[20] IBM Blue Gene/L Team. An Overview of the Blue Gene/L supercomputer. In *Supercomputing 2002*.

[21] V. Tsui and D.A. Case. Theory and applications of the generalized born solvation model in macromolecular simulations. *Biopolymers (Nucl. Acid. Sci.)*, 56, 2001.

[22] J. Vetter, S. Alam, T. Dunigan, M. Fahey, P. Roth, and P. Worley. Early evaluation of the Cray XT3. In *20th IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2006*.

[23] G. Zheng, T. Wilmarth, P. Jagadishprasad, and L. V. Kale. Simulation-based Performance Prediction for Large Parallel Machines. *International Journal of Parallel Programming*, 33(2-3), 2005.