Scalability of a Low-Cost Multi-Teraflop Linux Cluster for High-End Classical Atomistic and Quantum Mechanical Simulations

Hideaki Kikuchi,[2] Rajiv K. Kalia,[1,2] Aiichiro Nakano,[1,2] Priya Vashishta[1,2]
[1]Collaboratory for Multiscale Simulations, Department of Computer Science, Department of Physics & Astronomy,
Department of Materials Science, Department of Biomedical Engineering,
University of Southern California, Los Angeles, CA 90089, USA
[2]Concurrent Computing Laboratory for Materials Simulations, Biological Computation and Visualization Center,
Department of Computer Science, Department of Physics & Astronomy,
Louisiana State University, Baton Rouge, LA 70803, USA
hkikuchi@phys.lsu.edu, (rkalia, anakano, priyav)@usc.edu

Fuyuki Shimojo
Faculty of Integrated Arts and Sciences, Hiroshima University, Higashi-Hiroshima, 739-8521, Japan
shimojo@minerva.ias.hiroshima-u.ac.jp

Subhash Saini
IT Modeling and Simulation, NASA Ames Research Center, Moffett Field, CA 94035, USA
saini@nas.nasa.gov

Abstract

Scalability of a low-cost, Intel Xeon-based, multi-Teraflop Linux cluster is tested for two high-end scientific applications: Classical atomistic simulation based on the molecular dynamics method and quantum mechanical calculation based on the density functional theory. These scalable parallel applications use space-time multiresolution algorithms and feature computational-space decomposition, wavelet-based adaptive load balancing, and spacefilling-curve-based data compression for scalable I/O. Comparative performance tests are performed on a 1,024-processor Linux cluster and a conventional higher-end parallel supercomputer, 1,184-processor IBM SP4. The results show that the performance of the Linux cluster is comparable to that of the SP4. We also study various effects, such as the sharing of memory and L2 cache among processors, on the performance.

1. Introduction

There is growing interest in constructing supercomputers from commodity PCs and network components [1, 2]. Recent growth in the processing power of PCs has reduced the cost of a multi-Teraflop PC cluster within the budget of an academic institute. For example, Louisiana State University (LSU) has recently acquired a Linux cluster consisting of 512 dual Intel Xeon 1.8 GHz nodes (*i.e.*, 1,024 processors) connected by Myricom's Myrinet interconnect, see Fig. 1 [3]. The performance of the $2.6 million cluster, *SuperMike*, is rated as 2.21 Tflops, according to the standard High Performance Linpack (HPL) benchmark [4], and *SuperMike* was ranked as the 11th fastest supercomputer in the world in August 2002 [5].

Figure 1. The 1,024-processor Xeon cluster, *SuperMike*, at LSU.

Although the performance of low-cost multi-Teraflop Linux clusters has thus been confirmed by standard benchmark tests, there is a continuing concern regarding the scalability of such architecture for real high-end scientific/engineering applications at the Terascale, in comparison with conventional higher-end parallel supercomputers such as IBM SP4.

Computational materials science provides an excellent test case for such a comparative study. Rich variety of simulation methods — ranging from empirical molecular-dynamics (MD) simulations to *ab initio* quantum-mechanical (QM) calculations — is being used to study advanced materials and devices at the atomistic level. Recently, we have developed a suite of scalable MD and QM programs based on space-time multiresolution algorithms [6]. The production-quality programs in the suite also feature wavelet-based computational-space decomposition for adaptive load balancing and spacefilling-curve-based adaptive data compression with user-defined error bound for scalable I/O.

This paper describes a comparative performance study of two multi-Teraflop architectures — the 1,024-processor Linux cluster, *SuperMike*, and the 1,184-processors IBM SP4 system, *Marcellus*, at the Naval Oceanographic Office (NAVO) Major Shared Resource Center (MSRC) — for MD and QM applications. In the next section, we describe linear-scaling parallel algorithms for MD and QM calculations. Section 3 describes the two parallel architectures used in this study. Results of benchmark tests are given in Section 4, and Section 5 contains conclusions.

2. Scalable Parallel Atomistic Simulation Algorithms

Our linear-scaling MD and QM algorithms encompass a wide spectrum of physical reality: i) Classical MD based on a many-body interatomic potential model, and ii) self-consistent QM calculation based on the density functional theory (DFT). These algorithms deal with: i) All-pairs function evaluation in the N-body problem, and ii) exhaustive combinatorial enumeration in the quantum N-body problem. This section describes general algorithmic techniques to obtain approximate solutions to these problems in $O(N)$ time, including i) clustering, ii) hierarchical abstraction, and iii) the analysis of asymptotic solution properties. This section also describes a scalable parallel-computing framework to implement these algorithms on massively parallel computers.

2.1. Multiresolution Molecular Dynamics

In the MD approach, one obtains the phase-space trajectories of the system (positions and velocities of all atoms at all time) [6-8]. Atomic force laws for describing how atoms interact with each other is mathematically encoded in the interatomic potential energy, $E_{\mathrm{MD}}(\mathbf{r}^N)$, which is a function of the positions of all N atoms, $\mathbf{r}^N = \{\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_N\}$, in the system. In our many-body interatomic potential scheme, $E_{\mathrm{MD}}(\mathbf{r}^N)$ is expressed as an analytic function that depends on relative positions of atomic pairs and triples [9]. Time evolution of $\mathbf{r}^N$ is governed by a set of coupled ordinary differential equations.

For interatomic potentials with finite ranges, the computational cost can be made $O(N)$ using a linked-cell-list approach [10]. For long-range electrostatic interactions, the fast multipole method (FMM) [10, 11] computes the interatomic forces recursively on an octree data structure in $O(N)$ time.

Our ***multiresolution molecular dynamics (MRMD)*** algorithm [10] also uses an approach called the multiple time-scale (MTS) method [12-14]. The MTS method uses different force-update schedules for different force components, *i.e.*, forces from the nearest-neighbor atoms are computed at every MD step, and forces from farther atoms are computed with less frequency. This not only reduces the computational cost but also enhances the data locality, and accordingly the parallel efficiency is increased. These different force components are combined using a reversible symplectic integrator [14], and the resulting algorithm consists of nested loops to use forces from different spatial regions. It has been proven that the phase-space volume occupied by atoms is a simulation-loop invariant in this algorithm [14], and this loop invariant results in excellent long-time stability of the solutions.

For parallelization of MD simulations, we use spatial decomposition [10]. The total volume of the system is divided into P subsystems of equal volume, and each subsystem is assigned to a processor in an array of P processors. To calculate the force on an atom in a subsystem, the coordinates of the atoms in the boundaries of neighbor subsystems are “cached” from the corresponding processors. After updating the atomic positions due to a time-stepping procedure, some atoms may have moved out of its subsystem. These atoms are “migrated” to the proper neighbor processors. With the spatial decomposition, the computation scales as N/P while communication scales in proportion to $(N/P)^{2/3}$ for an N-atom system.

2.2. Linear-scaling Quantum-mechanical Calculation Based on the Density Functional Theory

Empirical interatomic potentials used in MD simulations fail to describe chemical processes. Instead, interatomic interaction in reactive regions needs to be calculated by a QM method that can describe breaking and formation of bonds. An atom consists of a nucleus and surrounding electrons, and quantum mechanics explicitly treats the electronic degrees-of-freedom. Since each electron’s wave function is a linear combination of many states, the combinatorial solution space for the many-electron problem is exponentially large. The density functional theory (DFT) avoids the exhaustive enumeration of many-electron correlations by solving M single-electron problems in a common average environment (M is the number of independent wave functions and is on the order of N). As a result, the problem is reduced to a self-consistent matrix eigenvalue problem, which can be solved with $O(M^3)$ operations [15, 16]. The DFT problem can also be formulated as a minimization of the energy, $E_{\mathrm{QM}}(\mathbf{r}^N, \psi^M)$, with respect to electron wave functions, $\psi^M(\mathbf{r}) = \{\psi_1(\mathbf{r}), \psi_2(\mathbf{r}), ..., \psi_M(\mathbf{r})\}$, subject to orthonormalization constraints between the wave functions.

Efficient parallel implementation of DFT is possible with real-space approaches based on higher-order finite differencing [17] and multigrid acceleration [18, 19]. We include electron-ion interaction using norm-conserving pseudopotentials [20] and the exchange-correlation energy associated with electron-electron interaction in a generalized gradient approximation [21]. For larger systems ($M > 1{,}000$), however, the $O(M^3)$ orthonormalization becomes the bottleneck.

For scalable DFT calculations, linear-scaling algorithms are essential [22]. We have implemented [23, 24] an $O(M)$ algorithm [25] based on unconstrained minimization of a modified energy functional and a localized-basis approximation. This algorithm is based on the observation that, for most materials at most temperatures, the off-diagonal elements of the density matrix decays exponentially [22]. Such a diagonally dominant matrix can be represented by maximally localizing each wave function, $\psi_n(\mathbf{r})$, by a unitary transformation and then truncating it with a finite cut-off radius. A Lagrange-multiplier-like technique is also used to perform unconstrained minimization, avoiding the $O(M^3)$ orthonormalization procedure. In the parallel ***linear-scaling density functional theory (LSDFT)*** algorithm, the computation time scales as $O(M/P)$ on P processors, whereas the communication scales as $O((M/P)^{2/3})$. This is in contrast to the $O(M(M/P)^{2/3})$ communication in the conventional parallel real-space DFT algorithm. Global communication for calculating overlap integrals of the wave functions (which scales as $M^2\log P$ in the conventional DFT algorithm) is unnecessary in the linear-scaling algorithm.

2.3. Load Balancing and Data Compression

Practical simulations involving multibillion atoms are associated with a number of computational challenges, which have been addressed by a number of software tools.

For example, many MD simulations are characterized by irregular atomic distribution and associated load imbalance. We have developed a computational-space-decomposition approach to load balancing [26]. This scheme partitions the system in a computational space, which is related to the physical space by a curvilinear coordinate transformation. (The computational space shrinks where the workload density is high and expands where the density is low, so that the workload is uniformly distributed.) The optimal coordinate system is determined to minimize the load-imbalance and communication costs. We have found that wavelet representation leads to compact representation of curved partition boundaries, and accordingly to fast convergence of the minimization procedure [27].

Another challenge is massive input/output (I/O). A 1.5-billion-atom MD simulation we are currently performing produces 150 GB of data per time step (or per minute), including atomic species, positions, velocities, and stresses. For scalable I/O of such large datasets, we have designed a data compression algorithm [28]. It uses octree indexing and sorts atoms accordingly on the resulting spacefilling curve. By storing differences between successive atomic coordinates, the I/O requirement for the same error tolerance level reduces from $O(N\log N)$ to $O(N)$. An adaptive, variable-length encoding scheme is used to make the scheme tolerant to outliers and optimized dynamically. An order-of-magnitude reduction in I/O was achieved for actual MD data with user-controlled error bound.

3. Multi-Teraflop Testbeds

We test the scalability of the MRMD and LSDFT algorithms on two multi-Teraflop computers: i) A 1,024-processor Intel Xeon system connected by Myrinet and ii) a 1,184-processor IBM SP4 system. This section describes the hardware architectures and software of the two machines, which are relevant to the interpretation of the performance results.

3.1. Intel Xeon-based Cluster: *SuperMike*

Recently, LSU has acquired through Atipa Technologies a Beowulf-class supercomputer named *SuperMike*. The system consists of 512 nodes each with dual Intel Xeon 1.8GHz processors, 2GB DDR SDRAM (*i.e.*, 1TB total), on a Tyan Thunder i7500 motherboard (utilizing the Intel E7500 chipset). 512KB L2 cache is integrated on die running at core clock on each Xeon processor. The Intel E7500 chipset optimized for dual Xeon processors delivers 3.2 GB/s of bandwidth across the 400 MHz system bus.

These nodes are interconnected by Myrinet with 500 MB/s of point-to-point bandwidth in the duplex mode. The Myrinet network consists of 12 E128 switch units (8 with line cards in the "Clos64+64" configuration and 4 with line cards in the "Spine" configuration) and 512 Myrinet-2000-Fiber/PCI interface cards. With this configuration, 64 nodes are connected within each of the 8 switches, which are in turn connected through Spine. Network software such as MPI, VI, Sockets, and TCP/IP are layered efficiently over "Glenn's Messages" (GM). The Red Hat Linux 7.2 operating system, GM 1.5, MPICH-GM 1.2.4.8a, and PBS Pro 5.2 are installed on *SuperMike*.

3.2. IBM SP4: *Marcellus*

The NAVO-MSRC has acquired an IBM SP-POWER4 supercomputer named *Marcellus* — a "Scalable Parallel" system assembled with workstation-class POWER4 processors, a dedicated high-speed network, and IBM's GPFS file system.

The 1,184-processor *Marcellus* consists of 148 nodes or LPARs (Logical Partitions), each with eight 1.3 GHz POWER4 processors. (Four LPARs, *i.e.*, 32 processors, in turn constitute a physical unit.) An LPAR contains four dual-processor chips, and in each chip, the two processors share L2 cache. The total memory size is 1.4 TB, and the system runs the AIX 5.1 operating system.

Marcellus uses a proprietary network and IBM's Colony II switch to communicate between nodes. The network switch provides 360 MB/s bi-directional bandwidth with 21 μs latency. There are two switch adapters per 8-way node, for a total of 296 adapters. The aggregate bi-directional bandwidth is 53.3 GB/s.

For MPI message passing between CPUs in the same LPAR (8 CPUs/LPAR), the IBM MPI Library can use fast shared memory, rather than the slower network switch. Message passing between CPUs on different nodes must use the network switch.

4. Performance Test Results

This section describes the performance test results for the MRMD and LSDFT applications on the Linux cluster, *SuperMike*, and the IBM SP4, *Marcellus*.

Figure 2 shows the execution time of the MRMD algorithm for silica material as a function of the number of processors, *P*. In this algorithm, the interatomic potential energy is split into the long-range and short-range contributions, where the long-range contribution is computed after 10 steps. We scale the system size linearly with the number of processors, so that the number of atoms, *N* = 648,000 *P*.

On the Linux cluster, the execution time increases only slightly as a function of *P* up to *P* = 512, which signifies excellent parallel efficiency. (On 512 processors, the parallel efficiency is 87%.) The sudden increase in the execution time from 512 to 1,024 processors is due to performance degradation by sharing main memory between the two processors within a dual-processor node. This arises since the test uses only one processor per node up to *P* = 512 and both processors for *P* = 1,024.

The computational time on the SP4 is less than that on the Linux cluster, but the communication time grows much faster as a function of *P* on the SP4 than on the Linux cluster. Consequently the total wall-clock time on the Linux cluster becomes less than that on the SP4 for larger numbers of processors (*P* > 512).

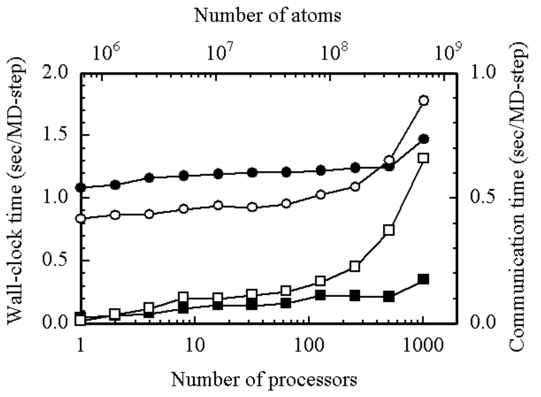

Figure 2. Wall-clock (circles) and communication (squares) times per time step of the MD algorithm with scaled workloads — 648,000 *P* atom silica systems on *P* processors (*P* = 1, ..., 1,024) of the Linux cluster (solid symbols) and the IBM SP4 (open symbols).

The above result suggests a detrimental effect of sharing main memory by the two processors within a node on the Linux cluster. To confirm this effect, Fig. 3 compares the results of two sets of MRMD benchmark tests. In the first set, only one processor per dual-processor node is used, whereas in the second set, both processors are used for *P* = 2, ..., 512.

The performance degradation due to sharing main memory in the execution time is nearly constant (~ 17%) from 2 to 512 processors. Figure 3 also shows performance degradation in the communication time, because the shared memory is used for communication between two processors in the same node. The resulting congestion in memory and/or internal bus significantly degrades the performance.

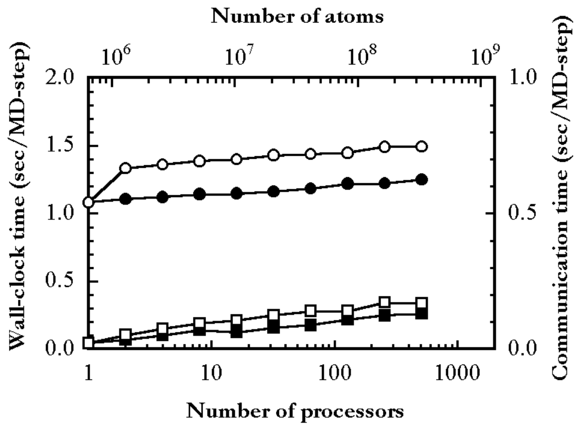

Figure 3. Wall-clock (circles) and communication (squares) times per time step of the MD algorithm with scaled workloads — 648,000 *P* atom silica systems on *P* processors (*P* = 1, ..., 512) of the Linux cluster using single processor per node (solid symbols) and two processors per node (open symbols).

On the IBM SP4, there is additional performance degradation due to the sharing of L2 cache by the two processors within a chip. To quantify this effect, we use a set of tools, *bindUtils*, provided by IBM, which allow us to use only a single processor per chip. This reduces congestions caused by sharing L2 cache between two processors in the same chip.

In Fig. 4, solid and open symbols denote the results of MRMD benchmark tests with and without *bindUtils*, respectively. In these tests, only the first chip in each node is used up to 128 processors, first and second chips for 256 processors, and all four chips for 512 processors. The use of *bindUtils* significantly reduces not only the execution time but also the communication time. The maximum reduction is observed on 64 processors, where the execution and communication times are reduced by 31% and 60%, respectively.

The two jumps (one from 8 to 16 processors and the other from 32 to 64 processors) in the communication time without *bindUtils* can be understood as communication bottleneck across 8-processor LPARs and that across 32-processor physical units, respectively. While the communication time with *bindUtils* is larger than that without the tools up to 8 processors, the wall-clock time is always reduced by *bindUtils*.

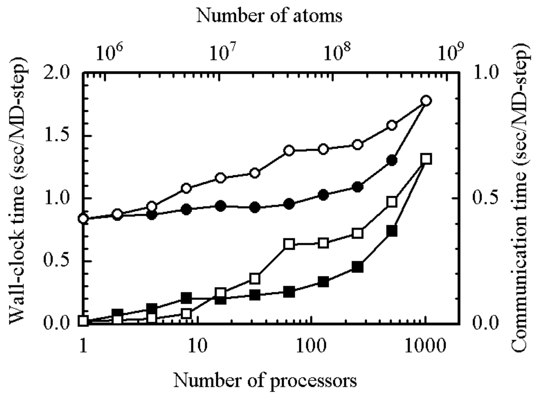

Figure 4. Wall-clock (circles) and communication (squares) times per time step of the MD algorithm with scaled workloads — 648,000 *P* atom silica systems on *P* processors (*P* = 1, ..., 1,024) of the IBM SP4 with (solid symbols) and without (open symbols) the use of *bindUtils*.

We also study the difference in performance using two different (Intel and PGI) compilers for the MRMD algorithm on the Linux cluster, see Fig. 5. The compilers used in the present benchmark tests are the Intel Compiler 6.0 and the PGI compiler 4.0. The Intel compiler makes better use of the architecture, including the Streaming SIMD Extensions 2 (SSE2) that augments the floating-point functional unit to deliver two results per cycle in the ideal case. Accordingly, the execution time using the Intel compiler is smaller than that with the PGI compiler.

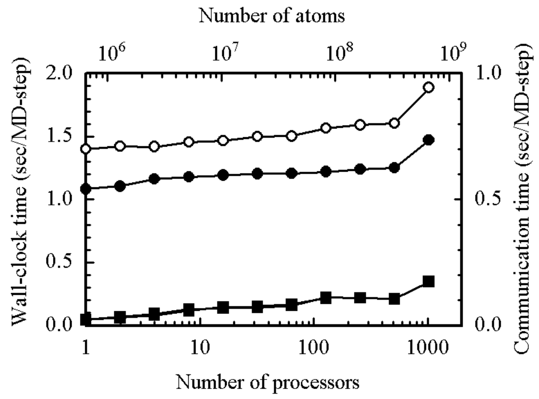

Figure 5. Wall-clock (circles) and communication (squares) times per time step of the MD algorithm with scaled workloads — 648,000 *P* atom silica systems on *P* processors (*P* = 1, ..., 1,024) of the Linux cluster with the Intel (solid symbols) and PGI (open symbols) compilers.

We also perform benchmark tests of the LSDFT algorithm for gallium arsenide material, in which the localization region for the wave functions is defined as a sphere with radius 4.4 Å. Figure 6 shows the wall-clock and communication times per conjugate gradient (CG) iteration on 1,024 Xeon and SP4 processors. The wall-clock time scales linearly with N above $N \sim 10{,}000$ (the number of wave functions, $M = 2N$). The interprocessor communication scales as $O(N^{0.6})$ for $N >$ 10,000.

The LSDFT application is characterized by heavier communication and memory access, compared with the MRMD application. The wall-clock times on the Linux cluster and the SP4 are nearly identical, though the SP4 has more advanced memory architecture (memory bandwidth is 11 GB/s) compared with the Linux cluster (memory bandwidth is 3.2 GB/s). In contrast, the communication time on the Linux cluster (using Myrinet) is significantly less than that on SP4.

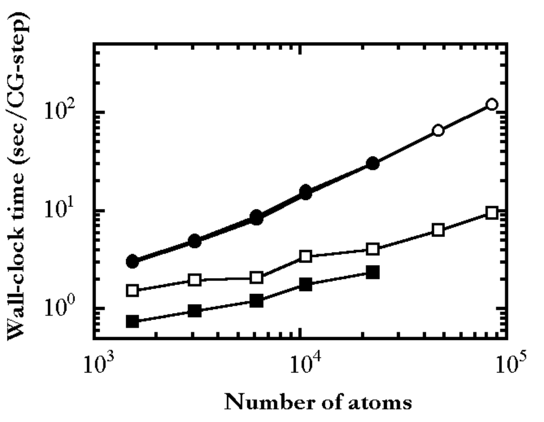

Figure 6. Wall-clock (circles) and communication (squares) times per CG step as a function of the number of atoms for the parallel LSDFT algorithm on the Linux cluster (solid symbols) and the IBM SP4 (open symbols). The system is gallium arsenide crystal in the zinc-blende structure. The number of processors is fixed at 1,024.

To understand why the advanced memory architecture of the IBM SP4 does not yield a better performance for the LSDFT application compared with that on the Linux cluster, we carry out LSDFT benchmark tests on the SP4 using four different bind configurations on 256 processors: ($N_{\rm proc}/N_{\rm chip}$, $N_{\rm chip}/N_{\rm node}$, $N_{\rm node}$) = (1, 2, 128), (1, 4, 64), (2, 2, 64), (2, 4, 32), where $N_{\rm proc}$, $N_{\rm chip}$, and $N_{\rm node}$ are the numbers of processors, chips, and nodes, respectively. The second and the third bind configurations require the same amount of communication across nodes, whereas the first and the fourth configurations are characterized by the lightest and the heaviest communication, respectively. The third and the fourth bind configurations include the effect of sharing L2 cache, but not the first and the second configurations.

Figure 7 compares the benchmark test results for the four bind configurations. The wall-clock time is almost the same for the first three configurations, whereas that of the fourth configuration is significantly larger. This indicates that the performance degradation on the IBM SP4 is not due to sharing L2 cache between processors. Instead, the heavy congestion inside a chip significantly slows down memory access. Consequently, the wall-clock time on the SP4 becomes comparable to that on the Linux cluster, even though the SP4 has more advanced memory architecture.

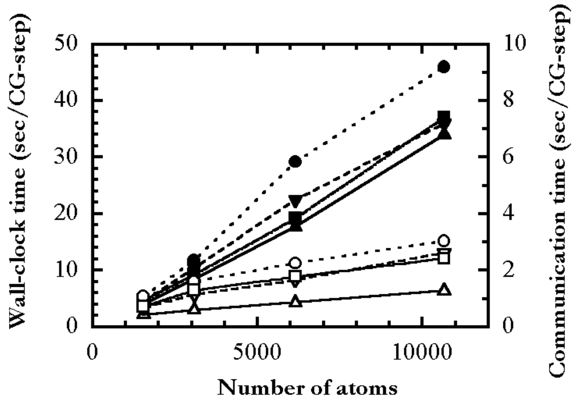

Figure 7. Wall-clock (solid symbols) and communication (open symbols) times per CG step as a function of the number of atoms for the parallel LSDFT algorithm on the IBM SP4, using four different bind configurations on 256 processors: ($N_{\rm proc}/N_{\rm chip}$, $N_{\rm chip}/N_{\rm node}$, $N_{\rm node}$) = (1, 2, 128) — triangles; (1, 4, 64) — reverse triangles; (2, 2, 64) — squares; (2, 4, 32) — circles.

5. Conclusions

The 1,024-processor Xeon cluster with Myrinet interconnect has exhibited excellent performance for two scientific applications with rather different characteristics — molecular dynamics and quantum mechanics. For these applications, the performance of the Linux cluster is comparable to (or even exceeding) that of a higher-end parallel supercomputer, IBM SP4. (Comparison with other supercomputers such as Compaq AlphaServer is in progress.) This demonstrates the viability of low-cost, PC-based, multi-Teraflop Linux clusters for real-world applications.

Considering that the 1.8 GHz Xeon used in this study is by no means the state of the art (2.8 GHz Xeon is currently available and 4.2 GHz Xeon is expected in

2003 with much enhanced system bus), this architecture, with powerful commodity-based interconnect, is very promising as high-performance computing platforms.

On the other hand, the measured performance of the MRMD and LSDFT algorithms clearly reflects the different architectures of the Linux cluster and the SP4, suggesting an urgent need to develop improved versions of these applications with portable performance.

Acknowledgements This work was partially supported by NASA, NSF, ARL, AFOSR, DOE, USC-Berkeley-Princeton DURINT, and Louisiana Board of Regents. The 1,024-processor Xeon cluster, *SuperMike*, was acquired by LSU's Center for Applied Information Technology and Learning (LSU CAPITAL) through funds appropriated by the Louisiana legislature as part of the Governor's information technology initiative. Benchmark tests were also performed on the 1,184-processor IBM SP4 computer, *Marcellus*, at Department of Defense's Naval Oceanographic Office (NAVO) Major Shared Resource Center (MSRC) under a DoD Challenge Project.

References

[1] T. Sterling (editor). *Beowulf Cluster Computing with Linux.* MIT Press, Cambridge, MA, 2001.

[2] R. Buyya (editor). *High-Performance Cluster Computing, Volumes 1 and 2.* Prentice Hall, Upper Saddle River, NJ, 1999.

[3] http://www.phys.lsu.edu/faculty/tohline/capital/beowulf.html.

[4] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary. HPL - a portable implementation of the high-performance Linpack benchmark for distributed-memory computers, http://www.netlib.org/benchmark/hpl.

[5] J. J. Dongarra. Performance of various computers using standard linear equations software. *Tech. Report, Univ. of Tennessee*, August 2002; http://www.netlib.org/benchmark/performance.ps.

[6] A. Nakano, R. K. Kalia, P. Vashishta, T. J. Campbell, S. Ogata, F. Shimojo, and S. Saini. Scalable atomistic simulation algorithms for materials research. In *Proceedings of Supercomputing 2001*, ACM, New York, 2001.

[7] S. J. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117:1-19 (1995).

[8] J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kalé. NAMD: Biomolecular simulations on thousands of processors. In *Proceedings of Supercomputing 2002*, IEEE Computer Society, Los Alamitos, CA, 2002.

[9] P. Vashishta, R. K. Kalia, and A. Nakano. Large-scale atomistic simulation of dynamic fracture. *Computing in Science & Engineering*, 1(5):56-65 (1999).

[10] A. Nakano, R. K. Kalia, and P. Vashishta. Multiresolution molecular dynamics algorithm for realistic materials modeling on parallel computers. *Computer Physics Communications*, 83:197-214 (1994).

[11] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325-348 (1987).

[12] A. Nakano. Fuzzy clustering approach to hierarchical molecular dynamics simulation of multiscale materials phenomena. *Computer Physics Communications*, 105:139-150 (1997).

[13] A. Nakano. A rigid-body based multiple time-scale molecular dynamics simulation of nanophase materials. *The International Journal of High Performance Computing Applications*, 13:154-162 (1999).

[14] M. E. Tuckerman, D. A. Yarne, S. O. Samuelson, A. L. Hughes, and G. J. Martyna. Exploiting multiple levels of parallelism in molecular dynamics based calculations via modern techniques and software paradigms on distributed memory computers. *Computer Physics Communications*, 128:333-376 (2000).

[15] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Physical Review*, 136:B864-B871 (1964).

[16] W. Kohn and P. Vashishta. General density functional theory. In *Inhomogeneous Electron Gas*, eds. N. H. March and S. Lundqvist, pages 79-184. Plenum, New York, 1983.

[17] J. R. Chelikowsky, Y. Saad, S. Ögüt, I. Vasiliev, and A. Stathopoulos. Electronic structure methods for predicting the properties of materials: Grids in space. *Phyica Status Solidi (b)*, 217:173-195 (2000).

[18] J.-L. Fattebert and J. Bernholc. Towards grid-based $O(N)$ density-functional theory methods: Optimized nonorthogonal orbitals and multigrid acceleration. *Physical Review B*, 62:1713-1722 (2000).

[19] T. L. Beck. Real-space mesh techniques in density-functional theory. *Reviews of Modern Physics*, 72:1041-1080 (2000).

[20] N. Troullier and J. L. Martins. Efficient pseudopotentials for plane-wave calculations. *Physical Review B*, 43:1993-2006 (1991).

[21] J. P. Perdew, K. Burke, and M. Ernzerhof. Generalized gradient approximation made simple. *Physical Review Letters*, 77:3865-3868 (1996).

[22] S. Goedecker. Linear scaling electronic structure methods. *Reviews of Modern Physics*, 71:1085-1123 (1999).

[23] F. Shimojo, T. J. Campbell, R. K. Kalia, A. Nakano, P. Vashishta, S. Ogata, and K. Tsuruta. A scalable molecular-dynamics-algorithm suite for materials simulations: Design-space diagram on 1,024 Cray T3E processors. *Future Generation Computer Systems*, 17:279-291 (2000).

[24] F. Shimojo, R. K. Kalia, A. Nakano, and P. Vashishta. Linear-scaling density-functional-theory calculations of electronic structure based on real-space grids: Design, analysis, and scalability test of parallel algorithms. *Computer Physics Communications*, 140:303-314 (2001).

[25] F. Mauri and G. Galli. Electronic-structure calculations and molecular-dynamics simulations with linear system-size scaling. *Physical Review B*, 50:4316-4326 (1994).

[26] A. Nakano and T. J. Campbell. An adaptive curvilinear-coordinate approach to dynamic load balancing of parallel multi-resolution molecular dynamics, *Parallel Computing*, 23:1461-1478 (1997).

[27] A. Nakano. Multiresolution load balancing in curved space: The wavelet representation. *Concurrency: Practice and Experience*, 11:343-353 (1999).

[28] A. Omeltchenko, T. J. Campbell, R. K. Kalia, X. Liu, A. Nakano, and P. Vashishta. Scalable I/O of large-scale molecular-dynamics simulations: A data-compression algorithm. *Computer Physics Communications*, 131:78-85 (2000).