A Metascalable Computing Framework for Large Spatiotemporal-Scale Atomistic Simulations

Ken-ichi Nomura, Richard Seymour, Weiqiang Wang, Hikmet Dursun, Rajiv K. Kalia, Aiichiro Nakano, Priya Vashishta
Collaboratory for Advanced Computing and Simulations, Department of Computer Science,
Department of Physics & Astronomy, Department of Chemical Engineering & Material Science,
University of Southern California, Los Angeles, CA 90089-0242, USA
(knomura, rseymour, wangweiq, hdursun, rkalia, anakano, priyav)@usc.edu

Fuyuki Shimojo
Department of Physics, Kumamoto University, Kumamoto 860-8555, Japan
shimojo@kumamoto-u.ac.jp

Lin H. Yang
Physics/H Division, Lawrence Livermore National Laboratory, Livermore, CA 94551, USA
lyang@llnl.gov

Abstract

A metascalable (or "design once, scale on new architectures") parallel computing framework has been developed for large spatiotemporal-scale atomistic simulations of materials based on spatiotemporal data locality principles, which is expected to scale on emerging multipetaflops architectures. The framework consists of: (1) an embedded divide-and-conquer (EDC) algorithmic framework based on spatial locality to design linear-scaling algorithms for high complexity problems; (2) a space-time-ensemble parallel (STEP) approach based on temporal locality to predict long-time dynamics, while introducing multiple parallelization axes; and (3) a tunable hierarchical cellular decomposition (HCD) parallelization framework to map these O(N) algorithms onto a multicore cluster based on hybrid implementation combining message passing and critical section-free multithreading. The EDC-STEP-HCD framework exposes maximal concurrency and data locality, thereby achieving: (1) inter-node parallel efficiency well over 0.95 for 218 billion-atom molecular-dynamics and 1.68 trillion electronic-degrees-of-freedom quantum-mechanical simulations on 212,992 IBM BlueGene/L processors (superscalability); (2) high intra-node, multithreading parallel efficiency (nanoscalability); and (3) nearly perfect time/ensemble parallel efficiency (eon-scalability). The spatiotemporal scale covered by MD simulation on a sustained petaflops computer per day (i.e. petaflops•day of computing) is estimated as NT = 2.14 (e.g. N = 2.14 million atoms for T = 1 microseconds).

1. Introduction

Fundamental understanding of complex system-level dynamics of many-atom systems is hindered by the lack of validated simulation methods to describe large spatiotemporal-scale atomistic processes. The ever-increasing capability of high-end computing platforms is enabling unprecedented scales of first-principles based simulations to predict system-level behavior of complex systems [1]. An example is large-scale molecular-dynamics (MD) simulation involving multibillion atoms, in which interatomic forces are computed quantum mechanically to accurately describe chemical reactions [2]. Such simulations can couple chemical reactions at the atomistic scale and mechanical processes at the mesoscopic scale to solve broad mechano-chemistry problems such as nanoenergetic reactions, in which reactive nanojets catalyze chemical reactions that do not occur otherwise [3]. A hard problem is to predict long-time dynamics, because the sequential bottleneck of time precludes efficient parallelization [4, 5].

The hardware environment is becoming challenging as well. Emerging sustained petaflops computers involve multicore processors [6], while the computer industry is facing a historical shift, in which Moore's law due to ever increasing clock speeds has been subsumed by increasing numbers of cores in microchips [7]. The multicore revolution will mark the end of the free-ride era (i.e., legacy software will run faster on newer chips), resulting in a dichotomy—subsiding speedup of conventional software and exponential speedup of scalable parallel applications.

To address these challenges, we have developed key technologies for parallel computing with portable scalability. These include an embedded divide-and-conquer (EDC) algorithmic framework to design linear-scaling algorithms for broad scientific and engineering applications (e.g. equation solvers, constrained optimization, search, visualization, and graphs involving massive data) based on spatial locality principles [8]. This, combined with a space-time-ensemble parallel (STEP) approach [9] to predict long-time dynamics based on temporal locality [10] and a tunable hierarchical cellular decomposition (HCD) parallelization framework, maximally exposes concurrency and data locality, thereby achieving reusable "design once, scale on new architectures" (or metascalable) applications. It is expected that such metascalable algorithms will continue to scale on future multicore architectures. The "seven dwarfs" (a dwarf is an algorithmic method that captures a pattern of computation and communication), which were first identified by Phillip Colella, have been used widely to develop scalable parallel programming models and architectures [6]. We expect that the EDC-STEP-HCD framework will serve as a "metascalable dwarf" to represent broad large-scale scientific and engineering applications.

We apply the EDC-STEP-HCD framework to a hierarchy of atomistic simulation methods. In MD simulation, the system is represented by a set of N point atoms whose trajectories are followed to study material properties [4, 11, 12]. Quantum mechanical (QM) simulation further treats electronic wave functions explicitly to describe chemical reactions [13-15]. To seamlessly couple MD and QM simulations, we have found it beneficial to introduce an intermediate layer, a first principles-based reactive force field (ReaxFF) approach [16, 17], in which interatomic interaction adapts dynamically to the local environment to describe chemical reactions. The ReaxFF is trained by performing thousands of small QM calculations.

This paper is organized as follows. Section 2 describes our metascalable computing framework for large spatiotemporal-scale simulations of chemical reactions based on spatiotemporal data-locality principles. First, we describe the EDC framework to design linear-scaling algorithms for broad applications based on spatial locality. Second, we present the STEP approach to predict long-time dynamics based on temporal locality. We then discuss the tunable HCD framework to map these $O(N)$ algorithms onto parallel computers. Results of scalability tests are given in section 3, and section 4 contains conclusions.

2. A Metascalable Dwarf

2.1 Embedded Divide-and-Conquer (EDC) Algorithmic Framework

In the embedded divide-and-conquer (EDC) algorithms, the physical system is divided into spatially localized computational cells [2]. These cells are embedded in a global field that is computed efficiently with tree-based algorithms (Fig. 1).

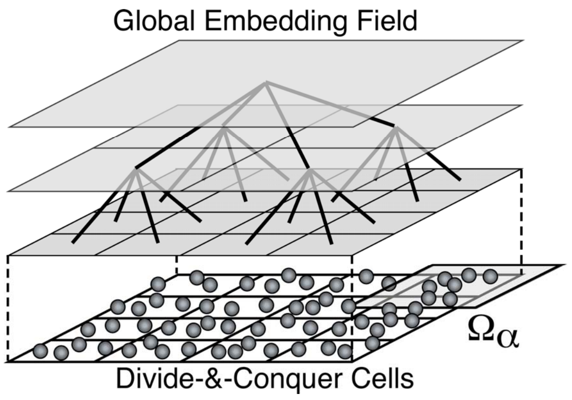


Figure 1. Schematic of embedded divide-and-conquer (EDC) algorithms. The physical space is subdivided into spatially localized cells, with local atoms constituting subproblems, which are embedded in a global field solved with tree-based algorithms.

Within the EDC framework, we have designed a number of $O(N)$ algorithms (N is the number of atoms). For example, we have designed a space-time multiresolution MD (MRMD) algorithm to reduce the $O(N^2)$ complexity of the N-body problem to $O(N)$ [11]. MD simulation follows the trajectories of N point atoms by numerically integrating coupled ordinary differential equations. The hardest computation in MD simulation is the evaluation of the long-range electrostatic potential at N atomic positions. Since each evaluation involves contributions from $N-1$ sources, direct summation requires $O(N^2)$ operations. The MRMD algorithm uses the octree-based fast multipole method (FMM) [18, 19] to reduce the computational complexity to $O(N)$ based on spatial locality. We also use multiresolution in time, where temporal locality is utilized by computing forces from further atoms with less frequency [20].

We have also designed a fast ReaxFF (F-ReaxFF) algorithm to solve the $O(N^3)$ variable N-charge problem in chemically reactive MD in $O(N)$ time [17]. To describe chemical bond breakage and formation, the ReaxFF potential energy is a function of the positions of atomic pairs, triplets and quadruplets as well as the chemical bond orders of all constituent atomic pairs [16]. To describe charge transfer, ReaxFF uses a charge-equilibration scheme, in which atomic charges are determined at every MD step to minimize the electrostatic energy with the charge-neutrality constraint. This variable N-charge problem amounts to solving a dense linear system of equations, which requires $O(N^3)$ operations. The F-ReaxFF algorithm uses the FMM to perform the matrix-vector multiplications with $O(N)$ operations. It further utilizes the temporal locality of the solutions to reduce the amortized computational cost averaged over simulation steps to $O(N)$. To further speed up the solution, we use a multilevel preconditioned conjugate gradient (MPCG) method [21]. This method splits the Coulomb interaction matrix into far-field and near-field matrices and uses the sparse near-field matrix as a preconditioner. The extensive use of the sparse preconditioner enhances the data locality, thereby increasing the parallel efficiency.

To approximately solve the exponentially complex quantum N-body problem, we use an EDC density functional theory (EDC-DFT) algorithm [15, 22]. The DFT reduces the exponential complexity to $O(N^3)$, by solving N_{el} one-electron problems self-consistently instead of one N_{el}-electron problem (the number of electrons, N_{el}, is on the order of N). The DFT problem can be formulated as a minimization of an energy functional with respect to electronic wave functions. In the EDC-DFT algorithm, the physical space is a union of overlapping domains, $\Omega = \Sigma_\alpha \Omega_\alpha$ (Fig. 1), and physical properties are computed as linear combinations of domain properties that in turn are computed from local electronic wave functions. For example, the electronic density $\rho(\mathbf{r})$ is calculated as $\rho(\mathbf{r}) = \Sigma_\alpha\, p^\alpha(\mathbf{r})\, \Sigma_n\, f(\varepsilon_n^\alpha)\, |\, \psi_n^\alpha(\mathbf{r})\, |^2$, where the support function $p^\alpha(\mathbf{r})$ vanishes outside domain Ω_α and satisfies the sum rule, $\Sigma_\alpha\, p^\alpha(\mathbf{r}) = 1$, and $f(\varepsilon_n^\alpha)$ is the Fermi distribution function corresponding to the energy ε_n^α of the n-th electronic wave function (or Kohn-Sham orbital) $\psi_n^\alpha(\mathbf{r})$ in Ω_α. For DFT calculation within each domain, we use a real-space approach based on high-order finite differencing [23], where iterative solutions are accelerated using the multigrid preconditioning [24]. The multigrid is augmented with high-resolution grids that are adaptively generated near the atoms to accurately operate atomic pseudopotentials [15]. The numerical core of EDC-DFT thus represents a high-order stencil computation [25].

2.2 Space-Time-Ensemble Parallelism (STEP) for Predicting Long-Time Dynamics

A challenging problem is to predict long-time dynamics because of the sequential bottleneck of time [4, 5]. Due to temporal locality, however, the system stays near local minimum-energy configurations most of the time, except for rare transitions between them. In such cases, the transition state theory (TST) allows the reformulation of the *sequential long-time dynamics* as computationally more efficient *parallel search* for low activation-barrier transition events [10, 26]. We also introduce a discrete abstraction based on graph data structures, so that combinatorial techniques can be used for the search [26]. We have developed a directionally heated nudged elastic band (DH-NEB) method [9], in which a NEB consisting of a sequence of S states [27], $\mathbf{R}_s \in \Re^{3N}$ (s = 0,...,S–1, $\Re$ is the set of real numbers, and N is the number of atoms), at different temperatures searches for transition events (Fig. 2(a)):

$$\mathbf{M}\frac{d^2}{dt^2}\mathbf{R}_s = \mathbf{F}_s - \mathbf{M}\gamma_s\frac{d}{dt}\mathbf{R}_s, \qquad (1)$$

where $\mathbf{M} \in \Re^{3N\times 3N}$ is the diagonal mass matrix and γ_s is a friction coefficient. Here, the forces are defined as

$$\mathbf{F}_s = \begin{cases} -\left.\dfrac{\partial V}{\partial \mathbf{R}_s}\right|_\perp + \left.\mathbf{F}_s^{\mathrm{spr}}\right|_{\|} & (1 \le s \le S-2) \\ -\dfrac{\partial V}{\partial \mathbf{R}_s} & (s = 0, S-1) \end{cases}, \qquad (2)$$

where $V(\mathbf{R})$ is the interatomic potential energy, $\mathbf{F}_s^{\mathrm{spr}}$ are spring forces that keep the states equidistance, and $\perp$ and $\|$ denote respectively the projections of a $3N$-element vector perpendicular and parallel to the tangential vector connecting the consecutive states.

We use an ensemble consisting of B bands to perform long-time simulation—molecular kinetics (MK) simulation—in the framework of kinetic Monte Carlo simulation [9]. Here, our space-time-ensemble parallel (STEP) approach combines a hierarchy of concurrency, i.e., the number of processors is

$$P = BSD: \qquad (3)$$

(1) spatial decomposition within each state (D is the number of spatial subsystems, see section 2.3); (2) temporal parallelism across S states within each band; and (3) ensemble parallelism over B bands (Fig. 2(b)).

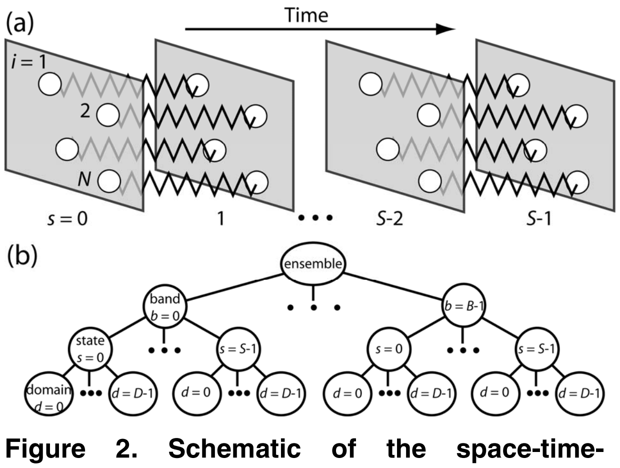


Figure 2. Schematic of the space-time-ensemble parallel (STEP) approach. (a) A

nudged elastic band consists of a sequence of *S* states (gray parallelograms), R_s (*s* = 0,...,*S*–1), where each state consists of *N* atoms (white spheres), *i* = 1,...,*N*. Corresponding atoms in consecutive states interact via harmonic forces represented by wavy lines. (b) Tree-structured processor organization in the STEP approach. An ensemble consists of *B* bands, each consisting of *S* states; each state in turn contains *D* spatial domains.

2.3 Tunable Hierarchical Cellular Decomposition (HCD) for Algorithm-Hardware Mapping

To map the $O(N)$ EDC-STEP algorithms onto parallel computers, we have developed a tunable hierarchical cellular decomposition (HCD) framework.

Massively distributed scalability via message passing—Superscalability: Our parallelization in space is based on spatial decomposition, in which each spatial subsystem is assigned to a compute node in a parallel computer. For large granularity (the number of atoms per spatial subsystem, $N/D > 10^2$), simple spatial decomposition (i.e., each node is responsible for the computation of the forces on the atoms within its subsystem) suffices, whereas for finer granularity ($N/D \sim 1$), neutral-territory [5, 28] or other hybrid decomposition schemes [4, 29-31] can be incorporated into the framework. Our parallelization framework also includes load-balancing capability. For irregular data structures, the number of atoms assigned to each processor varies significantly, and this load imbalance degrades the parallel efficiency. Load balancing can be stated as an optimization problem [32-34]. We minimize the load-imbalance cost as well the size and the number of messages. Our topology-preserving spatial decomposition allows message passing to be performed in a structured way in only 6 steps, so that the number of messages is minimized. To minimize the load imbalance cost and the size of messages, we have developed a computational-space decomposition scheme [35]. The main idea is that the computational space shrinks in a region with high workload density, so that the workload is uniformly distributed. The sum of load-imbalance and communication costs is minimized as a functional of the computational space using simulated annealing. We have found that wavelets allow compact representation of curved partition boundaries and thus speed up the optimization procedure [36].

Multicore scalability via multithreading—Nanoscalablity: In addition to the massive inter-node scalability, "there is plenty of room at the bottom," as Richard Feynman noted. At the finest level, EDC algorithms consist of a large number of computational cells (Fig. 1), such as linked-list cells in MD [11] and domains in EDC-DFT [15], which are readily amenable to parallelization. On a multicore compute node, a block of cells is assigned to each thread for intra-node parallelization. Our EDC algorithms are thus implemented as hybrid message passing + multithreading programs. Here, we use the POSIX thread standard, which is supported across broad architectures and operating systems. In addition, our framework [2] includes the optimization of data and computation layouts [37, 38], in which the computational cells are traversed along various spacefilling curves [39] (e.g. Hilbert or Morton curve). To achieve high efficiency, special care must be taken also to make the multithreading free of critical sections. This is illustrated below using MRMD as an example.

In MRMD, the interatomic potential energy V consists of two-body (V_2) and three-body (V_3) terms [11]. To compute interatomic forces, the Newton's third law is usually used to reduce computation. However, for multithreading, a race condition occurs when computed force values are sent back to memory due to the random memory access pattern of MD application. We have designed a critical section-free algorithm to make all interatomic force computations independent. For example, Eq. (4) shows a conventional summation rule to compute the three-body interaction without duplicating the same calculation (the two-body computation follows a similar but simpler procedure):

$$V_3 = \sum_{i=1}^{N} \sum_{j<k}^{nbr(i)} v(\mathbf{r}_j, \mathbf{r}_i, \mathbf{r}_k), \qquad (4)$$

where $\mathbf{r}_i$ is the coordinate of the i-th atom and $nbr(i)$ is the list of neighbor atoms within the three-body cutoff length from atom i. In order to avoid a race condition, we slightly rewrite Eq. (4) and calculate the three-body force, $\mathbf{F}_l^{(3)} = -\partial V_3 / \partial \mathbf{r}_i$, on the l-th atom as

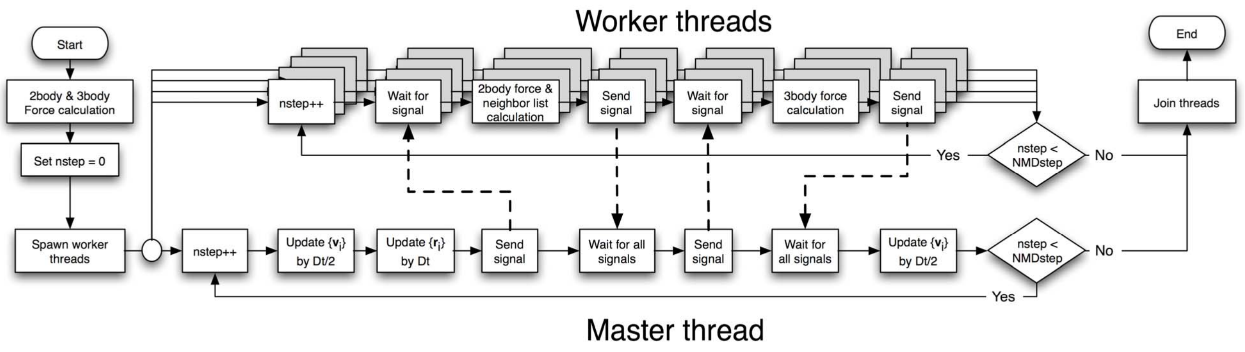


Figure 3. Flow chart of the critical section-free MD algorithm. At the beginning of simulation, a master thread computes complete force and energy before creating worker threads. Worker threads wait until master updates atomic coordinates $\{\mathbf{r}_i\}$ and velocities $\{\mathbf{v}_i\}$ (Dt is one MD time step). After receiving a signal from the master thread, the worker threads start computing two-body forces and neighbor-list, and then send a signal back to the master. Subsequently, three-body forces are computed following the same signaling scheme.

$$\mathbf{F}_l^{(3)} = -\sum_{i=1}^{N}\sum_{j=1}^{nbr(i)}\sum_{k\neq i}^{nbr(j)} \frac{\partial v(\mathbf{r}_i,\mathbf{r}_j,\mathbf{r}_k)}{\partial \mathbf{r}_i}\delta_{i,l}, \qquad (5)$$

where $\delta_{i,l}$ = 1 ($i = l$) or 0 (else). Note that Eq. (4) has atom i as the center of atomic triplet (j, $\boldsymbol{i}$, k), whereas Eq. (5) has atom i as its head, ($\boldsymbol{i}$, j, k). The modified three-body interaction computation consists of a loop over atom i, each iteration of which traverses atoms j in the neighbor list of atom i and subsequently neighbor atoms k in $nbr(j)$. By assigning different head atoms i to different threads, there cannot be any race condition.

Our multithreading is based on a master/worker model, in which a master thread coordinates worker threads that actually perform force computations. We use POSIX semaphores to signal between the master and worker threads to avoid the overhead of thread creation and joining in each MD step. There are two check points at each MD time step, where all worker threads wait a signal from the master thread: (1) before the two-body force calculation loop, which also constructs the neighbor-lists, after atomic coordinates are updated; and (2) before three-body force calculation, after having all atoms complete neighbor-list construction. Table 1 and Fig. 3 show a pseudo-code and a flow chart of our algorithm, respectively.

```
Master thread:
nstep = 0
compute 2-body and 3-body forces
spawn worker threads
while (nstep < Number_of_MD_steps)
 ·post semaphore_1 to worker threads
  to begin neighbor-list construction
  and 2-body force calculation
 ·wait for semaphore_1 signaled back
  from worker threads
 ·post semaphore_2 to worker threads
  to begin 3-body force calculation
 ·wait for semaphore_2 signaled back
  from worker threads
 ·update atom positions
 ·nstep++
join all threads

Worker threads:
while (nstep < Number_of_MD_steps)
 ·wait for semaphore_1 from master
  thread
 ·start neighbor-list and 2-body
  force calculation
 ·post semaphore_1 to master thread
 ·wait for semaphore_2 from master
  thread
 ·start 3-body force calculation
 ·post semaphore_2 to master thread
 ·nstep++
```

Table 1. Pseudo-code of the critical section-free force computation algorithm.

Long-time scalability via space-time-ensemble parallelism (STEP)—Eon-scalability: With the spatial decomposition, the computational cost scales as N/D, while communication scales in proportion to $(N/D)^{2/3}$ [11]. For long-range interatomic potentials used in MD simulations, tree-based algorithms such as the fast multipole method (FMM) [18, 19] incur an $O(\log D)$ overhead, which is negligible for coarse-grained ($N/D >> D$) applications [19]. The communication cost of the temporal decomposition is $O(N/D)$ for copying nearest-neighbor images along the temporal axis, but the prefactor is negligible compared with the computation. Ensemble decomposition duplicates the band calculation, each involving SD processors, B times on $P = BSD$ processors. It involves $O((N/D)\log(BS))$ overhead to multicast the new initial state among the processors assigned the same spatial domain, i.e., those with the same p mod D [9]. Here, $p = bSD + sD + d$ is the sequential processor ID, where processor p is assigned the d-th spatial subsystem of the s-th state in the b-th band. The multicast cost at the beginning of each molecular-kinetics (MK) simulation step is greatly amortized over 10^3–10^4 MD steps performed for the DH-NEB method per MK iteration.

Intelligent tuning: The hierarchy of computational cells provides an efficient mechanism for performance optimization as well—we make both the layout and size of the cells as tunable parameters that are optimized on each computing platform [2]. Our EDC-STEP algorithms are implemented as hybrid message-passing + multithreading programs in the tunable HCD framework, in which the numbers of message passing interface (MPI) processes and POSIX threads are also tunable parameters. The HCD framework thus maximally exposes data locality and concurrency. We are currently collaborating with compiler and artificial intelligence (AI) research groups to use: (1) knowledge-representation techniques for expressing the exposed concurrency; and (2) machine-learning techniques for optimally mapping the expressed concurrency to hardware [40].

3 Scalability Tests

The scalability of our EDC-STEP-HCD applications has been tested on various high-end computing platforms including 212,992 IBM BlueGene/L processors at the Lawrence Livermore National Laboratory and 131,072 IBM BlueGene/P processors at the Argonne National Laboratory. The BlueGene/L comprises 106,496 compute nodes (CN), each with two IBM PowerPC 440 processors (at 700 MHz clock frequency) and 512 MB of shared memory. Each processor has a 32 KB instruction and data cache, a 2 KB L2 cache, and a 4 MB L3 cache, which is shared with the other processor on the CN. Each CN has two floating-point units that can perform fused multiply-add operations. In its default mode (co-processor mode), one of the processors in the CN manages the computation, while the other processor manages the communication. In an alternative mode of operation (virtual mode), both processors can be used for computation. A 3D torus network connects nearest-neighbor CNs, and a collective global tree network handles communications involving all nodes. The 163,840-processor BlueGene/P consists of 40 racks each with 32 node-cards. On every node-card, there are 32 quadcore nodes, each with 2 GB DDR2 DRAM and four 450 POWER PC processors sharing L3 cache. Compared to the BlueGene/L, the BlueGene/P has 20% higher clock frequency (850 MHz) and 1.4 times larger communication bandwidth (5.1 GB/s). While the sizes of the private L1 and L2 caches are the same on both platforms, the shared L3 cache of the BlueGene/P (8 MB) is twice as large as that of the BlueGene/L. On the BlueGene/P, hardware latency for nearest neighbors is less than 1 microsecond, and latency of one-way tree traversal is 1.3 microseconds.

3.1 Inter-node (Message-Passing) Spatial Scalability

Figure 4 shows the execution and communication times of the MRMD, F-ReaxFF and EDC-DFT algorithms as a function of the number of processors P on the IBM BlueGene/L and P. We use one processor per dual-processor chip on BlueGene/L and all four cores per quadcore chip on BlueGene/P, so that 512 MB of memory is available per process to test the same problem size on both systems. Figure 4(a) shows the execution time of the MRMD algorithm for silica material as a function of the number of processors P. We scale the problem size linearly with the number of processors, so that the number of atoms N = 2,044,416P. In the MRMD algorithm, the interatomic potential energy is split into the long- and short-range contributions, and the long-range contribution is computed every 10 MD time steps. The execution time increases only slightly as a function of P on both BlueGene/L and P, and this signifies an excellent parallel efficiency. We define the speed of an MD program as a product of the total number of atoms and time steps executed per second. The isogranular speedup is the ratio between the speed of P processors and that of one processor. The weak-scaling parallel efficiency is the speedup divided by P, and it is 0.975 on 131,072 BlueGene/P processors. Though we have not completed all the benchmark runs, the measured weak-scaling parallel efficiency on 212,992 BlueGene/L processors is 0.985 based on the speedup over 4,096 processors. Figure 4(a) also shows that the algorithm involves very small communication time.

Figure 4(b) shows the execution time of the F-ReaxFF MD algorithm for RDX material as a function of P, where the number of atoms is N = 16,128P. The computation time includes 3 conjugate gradient (CG) iterations to solve the electronegativity equalization problem for determining atomic charges at each MD time step. On 212,992 BlueGene/L processors, the isogranular parallel efficiency of the F-ReaxFF algorithm is 0.996.

Figure 4(c) shows the performance of the EDC-DFT based MD algorithm for 180P atom alumina systems. The execution time includes 3 self-consistent (SC) iterations to determine the electronic wave functions and the Kohn-Sham potential, with 3 CG iterations per SC cycle to refine each wave function iteratively. On 212,992 BlueGene/L processors, the isogranular parallel efficiency of the EDC-DFT algorithm is 0.998 (based on the speedup over 4,096 processors).

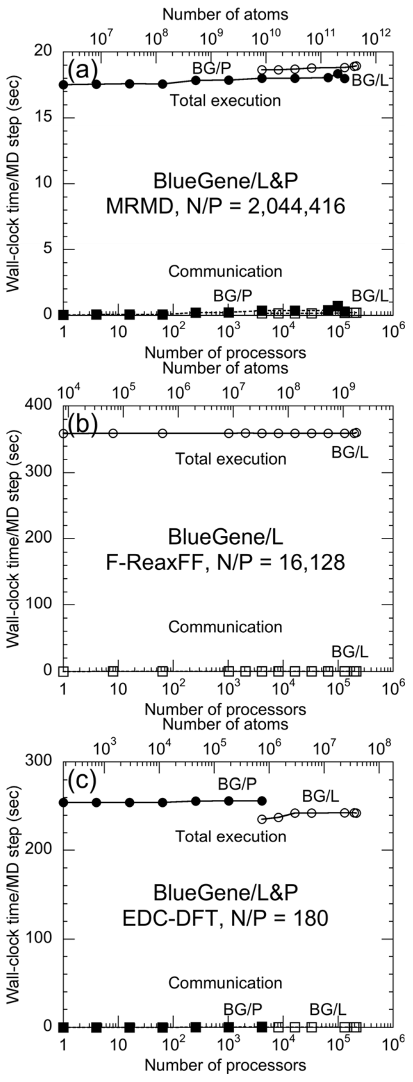


Figure 4. Total execution (circles) and communication (squares) times per MD time step as a function of the number of processors *P* of BlueGene/L (open symbols) and BlueGene/P (solid symbols) for three MD simulation algorithms: (a) MRMD for 2,044,416*P* atom silica systems; (b) F-ReaxFF MD for 16,128*P* atom RDX systems; and (c) EDC-DFT MD for 180*P* atom alumina systems.

Our largest benchmark tests include 217,722,126,336-atom MRMD, 1,717,567,488-atom F-ReaxFF, and 19,169,280-atom (1,683,216,138,240 electronic degrees-of-freedom) EDC-DFT calculations on 212,992 BlueGene/L processors. Though the absolute strong scaling (i.e. solving the same problem from P = 1 to 212,992) is not meaningful for such massive parallelism, the three algorithms exhibit excellent differential strong scalability, i.e., insensitivity of the speed on the granularity N/P. To quantify this effect, Fig. 5(a) plots the program speed (measured by the product of the number of atoms and MD time steps executed per second) for the MRMD algorithms as a function of the granularity (the number of atoms per processor, N/P) on BlueGene/L. The program maintains the same speed over a wide range of granularity for each P. The same data points are shown in Fig. 5(b) as a function of the number of processors, P. The MRMD program thus achieves a nearly perfect linear speedup as a function of the number of processors independent of the value of N/P.

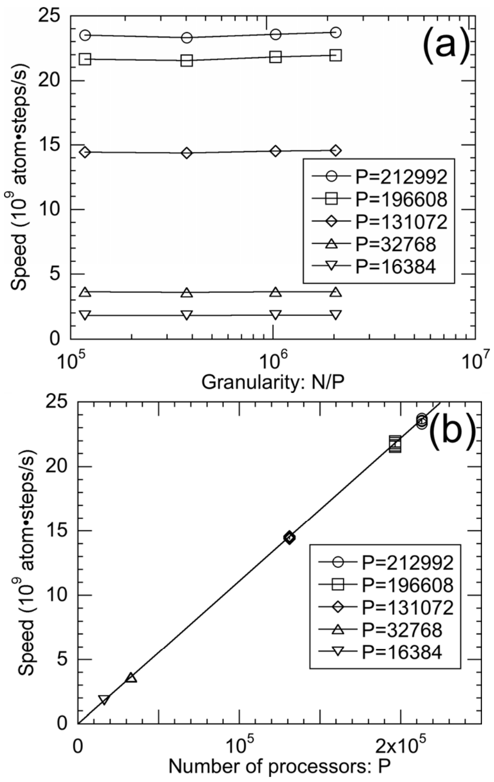


Figure 5. (a) Speed of MRMD on BlueGene/L as a function of the granularity *N/P*, for different numbers of processors *P*. (b) Speed of MRMD on BlueGene/L as a function of *P* for various granularities ranging from *N/P* = 117,912 to 2,044,416.

3.2 Intra-node (Multithreading) Spatial Scalability

We have tested the multithreading scalability of MRMD on a dual Intel Xeon quadcore platform. Figure 6 shows the speedup of the multithreaded code over the single-thread counterpart as a function of the number of worker threads. In addition to the speedup of the total program, Fig. 6 also shows the speedups of the code segments for two-body and three-body force calculations separately. We see that the code scales quite well up to 8 threads on the 8-core platform. We define the multithreading efficiency as the speedup divided by the number of threads. The efficiency of two-body force calculation is 0.927, while that for three-body force calculation is 0.436, for 8 threads. The low efficiency of the three-body force calculation may be due to the redundant computations introduced in Eq. (5) to eliminate critical sections. Nevertheless, the efficiency of the total program is rather high (0.811), since the fraction of the three-body calculation is about one third of the two-body force calculation. This result shows that the semaphore-based signaling between master and worker threads is highly effective. In a test calculation for a 12,228-atom silica system, the running time is 13.6 milliseconds per MD time step.

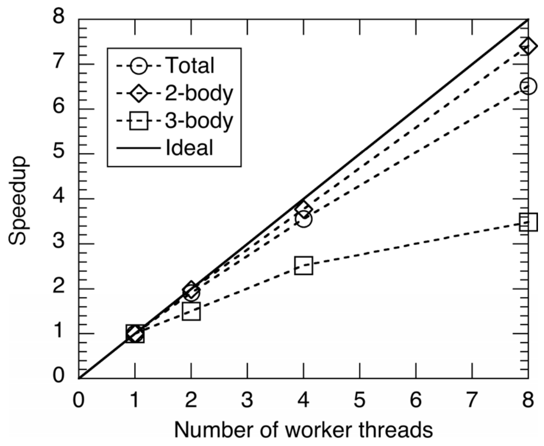
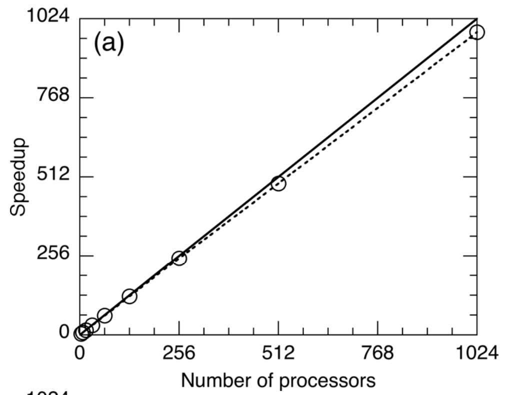


Figure 6. Speedup of the multithreaded MRMD algorithm over a single-threaded counterpart for the total program (circles), the two-body force calculation (diamonds), and three-body force calculation (squares). The solid line shows the ideal speedup.

3.3 Time/Ensemble Scalability

Scalability of the STEP-MRMD algorithm (note that the STEP approach can be combined with any of the MRMD, F-ReaxFF and EDC-DFT algorithms to compute interatomic forces) is tested on a cluster of dual-core, dual-processor AMD Opteron (at clock frequency 2 GHz) nodes with Myrinet interconnect, with 4 GB of memory per 4-core node. We define the speed of a program as a product of the total number of atoms and MK simulation steps executed per second. The speedup is the ratio between the speed of P processors and that of one processor. The parallel efficiency is the speedup divided by P.



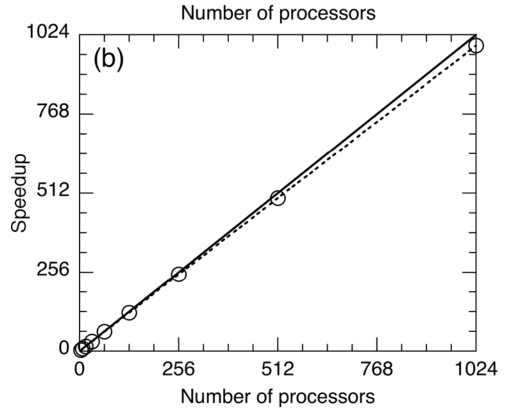


Figure 7. (a) Speedup of temporal decomposition in the STEP-MRMD algorithm (normalized so that the speedup is 4 for *P* = 4) as a function of the number of processors *P* (*P* = 4–1024) for a 192-atom amorphous SiO_2 system on dual-core, dual-processor AMD Opteron nodes, where we fix *B* = *D* = 1. The circles are measured speedups, whereas the solid line denotes the perfect speedup. (b) Speedup of ensemble decomposition in the STEP-MRMD algorithm as a function of the number of processors *P* (= 4,...,1024) for silica material (*N* = 192 atoms). Here, we fix the number of states per band *S* = 4 and the number of spatial domains per state *D* = 1, while the number of bands is varied from *B* = 1 to 256.

We first test the scalability of temporal decomposition, where we fix the number of bands $B = 1$ and the number of domains per state $D = 1$. We vary the number of states per band $S = 4$ to 1024. Here, the simulated system is amorphous SiO_2 consisting of $N = 192$ atoms, and we perform 600 MD steps per MK simulation step. The test uses all four cores per node. Figure 7(a) shows the speedup of the STEP-MRMD program (we normalize the speedup on 4 processors as 4). The measured speedup on 1,024 processors is 980.2, and thus the parallel efficiency is 0.957.

Next, we test the scalability of ensemble decomposition, where we fix the number of states per band S = 4 and the number of spatial domains per state D = 1. The number of bands per ensemble is varied from B = 1 to 256. The simulated system is amorphous SiO_2 consisting of N = 192 atoms. Although multiple events are generated independently by different processor groups, the parallel algorithm involves sequential bottlenecks such as the selection of an event that occurs, and accordingly the parallel efficiency does degrade for a larger number of processors. Figure 7(b) shows the speedup of the STEP-MRMD program on the Opteron cluster as a function of the number of processors (normalized to be 4 on 4 processors). On 1,024 processors, the measured speedup is 989.2, and thus the parallel efficiency of ensemble decomposition is 0.966, which is slightly higher than that of temporal decomposition on the same number of processors.

4 Conclusions

In summary, we have developed high-end reactive atomistic simulation programs to encompass large spatiotemporal scales with common algorithmic and computational frameworks based on spatiotemporal data-locality principles. In fact, the metascalable dwarf can reduce diverse applications (including all of the original seven dwarfs) to a highly scalable form by common techniques of embedding and divide-and-conquer. According to the scalability tests presented in this paper, they are likely to scale on future architectures beyond petaflops. The simulation algorithms are already enabling million-to-billion atom simulations of mechano-chemical processes, which have applications in broad areas such as energy and environment [3, 41, 42].

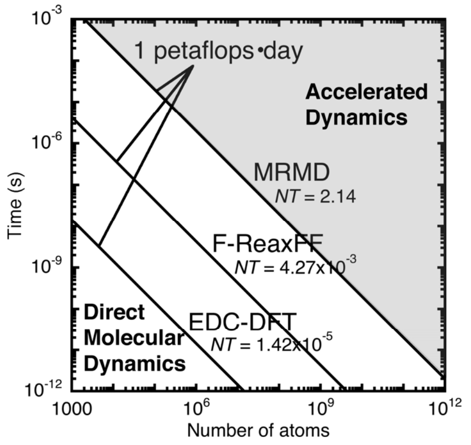


Figure 8. Spatiotemporal scales *NT* accessible by direct molecular-dynamics (white background) and approximate accelerated-dynamics (gray) simulations with a petaflops•day of computing. The lines are the *NT* achieved per petaflops•day of computing for MD (MRMD), chemically reactive MD (F-ReaxFF), and quantum-mechanical MD (EDC-DFT) simulations, respectively.

A critical issue, however, is the time scale studied by MD simulations. We define the spatiotemporal scale, NT, of an MD simulation as the product of the number of atoms N and the simulated time span T. On petaflops computers, direct MD simulations can be performed for NT = 1-10 atom•seconds (i.e. multibillion-atom simulation for several nanoseconds or multimillion-atom simulation for several microseconds). More specifically, a day of computing on a sustained petaflops computer (i.e. one petaflops•day of computing) achieves NT = 2.14 (e.g. 1 million atoms for 2.14 microseconds) (Fig. 8), according to the benchmark test in section 3 (i.e., extrapolated from the measured MRMD performance on the BlueGene/L, which is rated as 0.478 petaflops according to the Linpack benchmark). Accelerated-dynamics simulations [10] such as STEP molecular-kinetics simulations [9] will push the spatiotemporal envelope beyond NT = 10, but they need to be fully validated against direct MD simulations at NT = 1-10. Such large spatiotemporal-scale atomistic simulations are expected to advance scientific knowledge. This work was supported by NSF-ITR/PetaApps/EMT, DOE-SciDAC/BES, ARO-MURI, DTRA, and Chevron-CiSoft. We thank the staff of the Argonne Leadership Computing Facility for their help on the BlueGene/P benchmark.

References

[1] S. Emmott and S. Rison, *Towards 2020 Science* (Microsoft Research, Cambridge, UK, 2006).

[2] A. Nakano *et al.*, Int'l J High Performance Comput Appl **22**, 113 (2008).

[3] K. Nomura *et al.*, Phys Rev Lett **99**, 148303 (2007).

[4] J. C. Phillips *et al.*, in *Proc of Supercomputing (SC02)* (ACM/IEEE, 2002).

[5] D. E. Shaw *et al.*, ACM SIGARCH Computer Architecture News **35**, 1 (2007).

[6] K. Asanovic *et al.*, *The Landscape of Parallel Computing Research: A View from Berkeley* (University of California, Berkeley, 2006).

[7] J. Dongarra *et al.*, CTWatch Quarterly **3**, 11 (2007).

[8] A. Nakano *et al.*, Comput Mater Sci **38**, 642 (2007).

[9] A. Nakano, Comput Phys Commun **178**, 280 (2008).

[10] A. F. Voter, F. Montalenti, and T. C. Germann, Annual Rev Mater Res **32**, 321 (2002).

[11] A. Nakano *et al.*, in *Proc of Supercomputing (SC01)* (ACM/IEEE, 2001).

[12] J. N. Glosli *et al.*, in *Proc of Supercomputing (SC07)* (ACM/IEEE, 2007).

[13] J. Nieplocha, R. J. Harrison, and R. J. Littlefield, in *Proc of Supercomputing (SC94)* (ACM/IEEE, 1994).

[14] F. Gygi *et al.*, in *Proc of Supercomputing (SC05)* (ACM/IEEE, 2005).

[15] F. Shimojo *et al.*, Phys Rev B **77**, 085103 (2008).

[16] A. C. T. van Duin *et al.*, J Phys Chem A **105**, 9396 (2001).

[17] K. Nomura *et al.*, Comput Phys Commun **178**, 73 (2008).

[18] L. Greengard and V. Rokhlin, J Computat Phys **73**, 325 (1987).

[19] S. Ogata *et al.*, Comput Phys Commun **153**, 445 (2003).

[20] A. Nakano, R. K. Kalia, and P. Vashishta, Comput Phys Commun **83**, 197 (1994).

[21] A. Nakano, Comput Phys Commun **104**, 59 (1997).

[22] W. T. Yang, Phys Rev Lett **66**, 1438 (1991).

[23] J. R. Chelikowsky *et al.*, Physica Status Solidi B **217**, 173 (2000).

[24] J.-L. Fattebert and J. Bernholc, Phys Rev B **62**, 1713 (2000).

[25] K. Datta *et al.*, in *Proc of Supercomputing (SC08)* (IEEE/ACM, 2008).

[26] A. Nakano, Comput Phys Commun **176**, 292 (2007).

[27] G. Henkelman and H. Jonsson, J Chem Phys **113**, 9978 (2000).

[28] D. E. Shaw, J Comput Chem **26**, 1318 (2005).

[29] S. J. Plimpton, J Comput Phys **117**, 1 (1995).

[30] B. G. Fitch *et al.*, Lecture Notes in Computer Science **3992**, 846 (2006).

[31] M. Snir, Theor Comput Syst **37**, 295 (2004).

[32] R. D. Williams, Concurrency: Practice and Experience **3**, 457 (1991).

[33] K. D. Devine *et al.*, Appl Numer Math **52**, 133 (2005).

[34] U. V. Catalyurek *et al.*, in *Proc of International Parallel and Distributed Processing Symposium (IPDPS)* (IEEE, 2007).

[35] A. Nakano, and T. J. Campbell, Parallel Comput **23**, 1461 (1997).

[36] A. Nakano, Concurrency: Practice and Experience **11**, 343 (1999).

[37] J. Mellor-Crummey, D. Whalley, and K. Kennedy, Int'l J Parallel Prog **29**, 217 (2001).

[38] M. M. Strout, and P. D. Hovland, in *Proc of the Workshop on Memory System Performance* (ACM, 2004).

[39] B. Moon *et al.*, IEEE Trans Knowledge Data Eng **13**, 124 (2001).

[40] B. Bansal *et al.*, in *Proc of the Next Generation Software Workshop, International Parallel and Distributed Processing Symposium (IPDPS)* (IEEE, 2007).

[41] I. Szlufarska, A. Nakano, and P. Vashishta, Science **309**, 911 (2005).

[42] Y. C. Chen *et al.*, Phys Rev Lett **99**, 155506 (2007).