ELSEVIER

Parallel Computing 23 (1997) 1461–1478

PARALLEL COMPUTING

An adaptive curvilinear-coordinate approach to dynamic load balancing of parallel multiresolution molecular dynamics

Aiichiro Nakano *, Timothy Campbell

Department of Computer Science, Concurrent Computing Laboratory for Materials Simulations, Louisiana State University, Baton Rouge, LA 70803-4020, USA

Received 1 April 1996; revised 6 April 1997; accepted 19 April 1997

Abstract

We present a practical experience in adding a dynamic-load-balancing capability to an existing large parallel application — multiresolution molecular dynamics (MRMD) — which is based on uniform mesh decomposition. The new load-balancing scheme uses adaptive curvilinear coordinates to represent partition boundaries. Workloads are partitioned with a uniform 3-dimensional mesh in the curvilinear coordinate system. Simulated annealing is used to determine the optimal coordinate system which minimizes load imbalance and communication costs. The number of messages for performing simulations is minimal because of the underlying regular mesh topology. Periodic boundary conditions are naturally incorporated in the new scheme. Performance of the MRMD algorithm with the new load balancer has been tested for nonuniform multimillion-atom systems. © 1997 Elsevier Science B.V.

Keywords: Multiresolution molecular dynamics (MRMD); Load balancing; Adaptive curvilinear coordinates; Simulated annealing; Communication costs; Message passing

1. Introduction

Molecular dynamics (MD) simulation is rapidly becoming an integral part of computer aided design of materials [45]. The MD approach provides the phase-space trajectories of atoms through the solution of Newton's equations. Recent developments in parallel computing technology and multiresolution numerical methods have enabled

* Corresponding author. E-mail: nakano@bit.csc.lsu.edu; URL: http://www.cclms.lsu.edu.

PII S0167-8191(97)00060-4

us to perform multimillion-atom MD simulations of realistic materials routinely [23,27,30].

The compute-intensive part of MD simulations is the calculation of interatomic interactions. We are dealing with materials in which interatomic interactions are characterized by steric repulsion, Coulomb and charge–dipole interactions, and three-body covalent interactions [47]. Highly efficient algorithms have been designed to compute these interactions on parallel machines [27]. The long-range Coulomb interaction is calculated with a divide-and-conquer scheme, called the fast multipole method (FMM) [14], which reduces the computational complexity from $O(N^2)$ to $O(N)$ for N-atom systems. The reduced cell multipole method (RCMM) [8] is used to apply the FMM to systems with periodic boundary conditions. For short-ranged two- and three-body interactions, we have employed a multiple time-step (MTS) approach [41] in which the force on an atom is subdivided into primary, secondary, and tertiary components according to spatial ranges. A significant reduction in computation is achieved by exploiting different time scales of these force components. Domain decomposition is used to implement this multiresolution molecular dynamics (MRMD) algorithm on parallel computers.

Major application areas of the MRMD algorithm have been porous glasses [26] and fracture of ceramics [28,29]. Porous silica has recently been the focus of many investigations, see Fig. 1a [10]. This environmentally safe material has numerous technological applications: It is used in thermal insulation of refrigerators; in passive solar energy collection devices; and in catalysis and chemical separation. There is also an exciting possibility of utilizing it as an embedding framework in optical switches made of quantum-confined microclusters. More recently we have performed MD simulations of nanophase ceramics [43,44], see Fig. 1b. Advanced ceramics such as silicon nitride have numerous technological applications [46]. The capability to withstand high temperatures, combined with high strength and low weight, make them highly desirable for aerospace, surface transportation, electronics, and advanced manufacturing industries. A major drawback of ceramics is their brittleness. In recent years, a great deal of effort has been made to synthesize less brittle ceramics by consolidation of nanometer size clusters [39].

Simulation of porous glasses and nanophase materials is characterized by low mass density and irregular atomic distribution. For example, we have simulated highly nonuniform porous glasses whose density is as low as one twentieth of the normal glass density (Fig. 1a) [26]. One practical problem in simulating such irregular systems on parallel computers is that of load imbalance. Suppose that processors are logically organized as a cubic array of dimensions $P_x \times P_y \times P_z$, and that we partition the simulation system into subsystems of equal volume accordingly. Because of the irregular distribution of atoms, this uniform spatial decomposition results in unequal partition of workloads among processors. As a result the parallel efficiency is degraded significantly. Another problem arises during the preparation of these materials, when global rearrangement of structures occurs. This necessitates a dynamic load balancing scheme in which workloads are repartitioned adaptively during simulations.

Various approaches have been developed for load balancing such dynamic irregular problems on parallel computers [9]. For example, recursive coordinate bisection is one

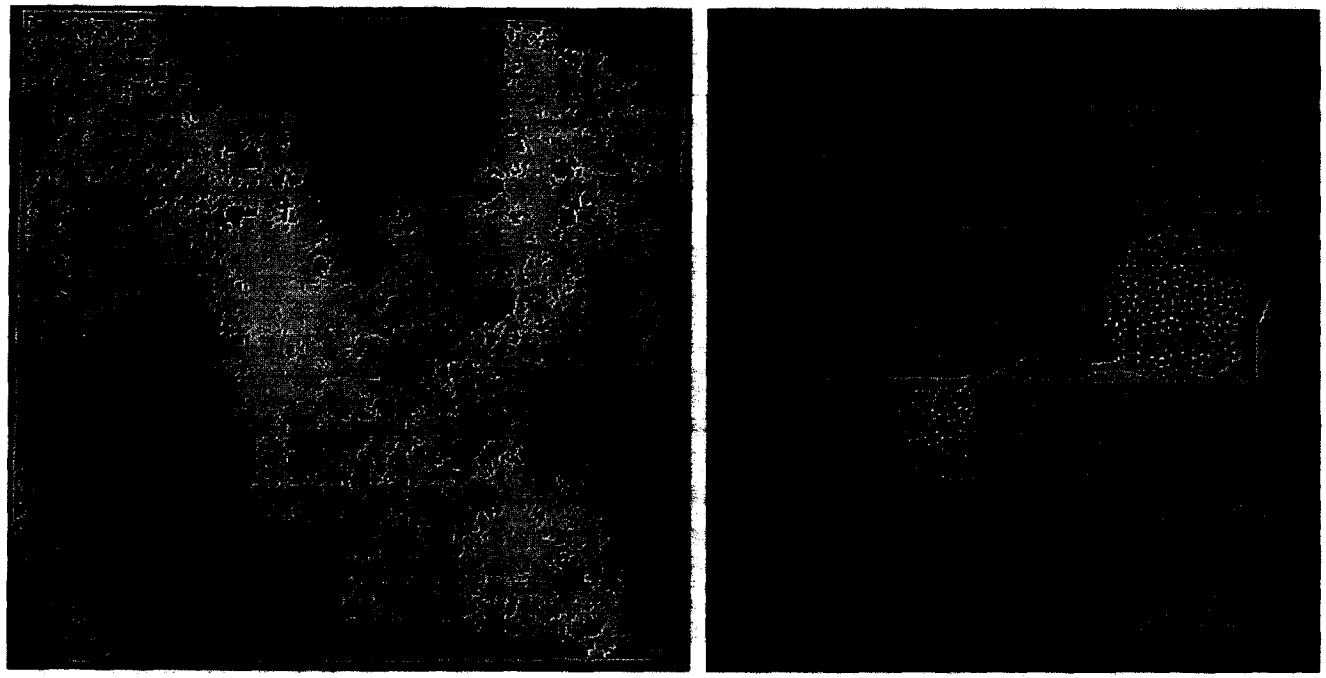

Fig. 1. (a) Snapshot of an MD porous silica at density 0.1 g/cm^3 [26]. (b) MD configuration of a nanocluster-assembled silicon nitride [43,44]. The 1,085,616-atom system consists of 108 clusters each containing 10,052 atoms. (b) also illustrates schematically a uniform mesh decomposition for parallel implementation. The walls represent partition boundaries to divide workloads. The arrows represent the flow of atomic information.

of the widely used methods [21,48,50]. The load-balancing problem can also be formulated in terms of the more general graph partitioning problem. Spectral partitioning methods use the lowest nontrivial eigenvectors of the Laplacian matrix of a graph to produce a high-quality partition [2,13,19,50]. Since the spectral method involves the diagonalization of the Laplacian matrix, its computational cost is high. Recently, multilevel algorithms have been combined with the spectral method to reduce the computational cost [2,13,19]. By constructing successive coarse approximations of the original graph, these multilevel spectral methods solve static problems efficiently where the cost to perform load balancing is tolerated.

In irregular dynamic simulations, need for repeated repartitioning necessitates low overhead load balancers. Most successful among dynamic load balancing schemes are the methods based on space-filling curves [13,33,35,36,40,49]. These methods use a bijective mapping of a 1-dimensional array to 3-dimensional grid points. Spatial locality is conserved by using recursively defined self-similar curves for the mapping. The space-filling curves have been applied to load balancing astrophysical particle simulations based on tree codes. In a dynamic load-balancer, partition can be refined incrementally during a simulation based on load-imbalance pressure [7,35] or by load diffusion [3,5,18,20].

While the load-balancing schemes cited above have been quite successful, we must take account of several additional points in designing a dynamic load-balancer for MRMD simulations. First, the uniform-mesh decomposition in the original parallel MRMD algorithm is optimal in reducing the number of messages [27]. With a three-dimensional mesh decomposition, message passing to the 26 neighbor nodes is completed in total of six steps [30]. This is achieved by sending the coordinates of boundary atoms to the six face-sharing neighbor nodes sequentially. Boundary atoms located at the corners and edges of a processor are forwarded to proper neighbor processors. A two-dimensional decomposition (e.g., for simulation of films) requires only four messages. This small number of messages is a desirable feature when we run simulations on a cluster of workstations such as the 40-node Digital Alpha system at the Concurrent Computing Laboratory for Materials Simulations of Louisiana State University. If a portable message passing system such as PVM [11] or MPI [16] is implemented with the TCP/IP protocol on a cluster of workstations, message latency rather than communication bandwidth is often the limiting factor for speedup [32]. In such a case, minimizing the number of messages is as important as minimizing the volume of each message. The message forwarding scheme used with the uniform mesh partitioning is optimal in this aspect, and it is desirable to retain this feature with a new dynamic load-balancing scheme. Secondly, it is important that a load-balancer conforms to the periodic boundary conditions used in the MRMD program. Periodic boundary conditions are used often in materials simulations to minimize the surface effects.

We have developed a low-overhead dynamic load balancing scheme which satisfies both of these requirements: minimizing the number of messages and incorporating periodic boundary conditions. The main idea is to introduce an adaptive curvilinear-coordinate system in which we apply the uniform-mesh partitioning. Coordinate transformation is then dynamically adapted to minimize the load-imbalance and communication costs.

This paper is organized as follows. In Section 2, we describe the MRMD algorithm and the adaptive curvilinear-coordinate load balancing scheme. The results of numerical tests are given in Section 3, and Section 4 contains discussions.

2. Methods

2.1. Multiresolution molecular dynamics

In MD simulations, a physical system consisting of N atoms is represented by a set of atomic coordinates, $\{\boldsymbol{x}_i | i = 1, \ldots, N\}$. We are concerned with systems for which the potential energy is a sum of pair and triple terms [47],

$$V = \sum_{\{(i,j)\}} v_{ij}^{(2)}\left(|\boldsymbol{x}_{ij}|\right) + \sum_{\{(i,j,k)\}} v_{jik}^{(3)}(\boldsymbol{x}_{ij}, \boldsymbol{x}_{ik}), \tag{1}$$

where $\boldsymbol{x}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$. The most time-consuming part of MD simulations is the calculation of the potential energy, V, and interatomic forces, $\boldsymbol{F}_i = -\partial V / \partial \boldsymbol{x}_i$. The MRMD algorithms have been designed to efficiently compute the contributions to these quantities from various spatial regimes [27].

Recently, several divide-and-conquer schemes have been implemented to reduce the computational complexity for the long-range Coulomb interaction. The fast multipole method (FMM) is based on the multipole expansion of the Coulomb interaction [14]. The rapid decay of the multipole expansion enables one to calculate the Coulomb interaction efficiently for a specific level of precision. On a sequential machine, the FMM reduces the computational complexity from $O(N^2)$ to $O(N)$. It involves: (i) decomposition of the MD box into a hierarchy of cells; (ii) calculation of multipole moments; and (iii) Taylor series expansion of the Coulomb potential.

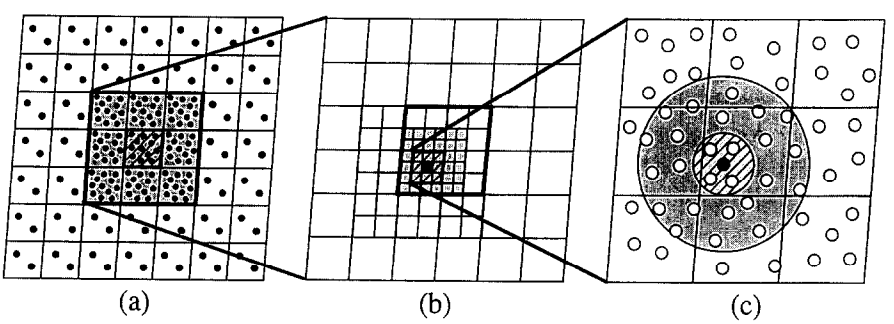

Fig. 2. Schematic representation of the multiresolution algorithm. (a) Periodically repeated images of the original MD box. Replacing each well-separated image by a small number of particles with the same leading multipole expansions reduces the computation enormously while maintaining the necessary accuracy. (b) A hierarchy of cells in the fast multipole method. (c) The near-field forces on a particle are due to primary, secondary, and tertiary neighbor atoms.

In the FMM, the MD box is decomposed into a hierarchy of cells according to a tree structure (Fig. 2b). The root of the tree is at level 0, and it corresponds to the MD box. The MD box is divided into $2 \times 2 \times 2$ cells at level 1. Repeating this procedure, one obtains 8^l cells at level l. The recursive decomposition is stopped at the leaf level, $l = \mathrm{L}$, where further decomposition would cause the sides of the cells to be less than the cutoff for the non-Coulombic interaction (i.e., charge-dipole, steric, and three-body). Atoms experience only the Coulomb potential when they reside in non nearest-neighbor or well-separated cells.

The total potential energy can be decomposed into the near-field and far-field contributions. The near-field contribution is due to atoms in nearest-neighbor cells, and is calculated directly at the leaf level. The far-field contribution is evaluated by expanding the Coulomb field in terms of multipoles. The multipoles are calculated hierarchically for every cell at every level. We first determine multipoles for all the cells at the leaf level. Next the multipoles at coarser levels are calculated by combining the multipoles at deeper levels. At level l, the multipoles at the center of a cell is calculated by shifting and adding up the multipoles of its 8 children cells. By traversing the tree upward, $l = L, L-1, \ldots, 1, 0$, one computes the multipoles for all the cells at all levels.

These multipoles are then used to compute the far field at each atom's position. The separation between atoms in a given cell is small compared to the separation between atoms in well-separated cells. Therefore, instead of repeating the sum over all well-separated cells for each atom in the cell, the field is expanded in Taylor series about the center of the cell. Starting at the root level ($l = 0$) and traversing the tree downward, we compute the Taylor coefficients from the multipoles for all the cells at all levels. The Taylor coefficients are stored, and are used to calculate the far field at the positions of the atoms in the cell.

In simulations of bulk systems, periodic boundary conditions are imposed by repeating the MD box infinitely as shown in Fig. 2a. In this case, the Coulomb interaction involves the wellknown Ewald summation which expresses the interaction as a local sum in the Fourier space and a sum of short-range terms in real space [23]. The Ewald summation is efficiently calculated with the so-called reduced cell multipole method (RCMM) [8]. The main idea of the RCMM is that even though the MD box contains multimillion atoms, the contribution from ∞-27 well-separated image cells to the far field is described accurately by the first few terms of the multipole expansion. Therefore it is reasonable to replace each well-separated image cell by a reduced cell containing a small number of randomly positioned fictitious particles whose first K multipole moments are the same as those of the original N-atom system. To reproduce the multipole moments up to order K, only $M = (K+1)(K+2)(K+3)/6$ fictitious particles are needed. The charges, $\{q_i\}$, of these M fictitious particles are obtained by equating their first K multipole moments to the corresponding multipole moments of N image atoms. In RCMM, the Coulomb potential is a sum of two terms: (i) the potential generated by charges in the central MD box and image atoms in the 26 nearest-neighbor boxes; and (ii) the potential due to M fictitious charges in the remaining ∞-27 cells (see Fig. 2a). The first term (i) is computed with FMM. The second term (ii) involves the Ewald summation over M fictitious particles.

For short-ranged two-body and three-body interactions, we have employed a multiple timestep (MTS) scheme [41] in which the force on an atom is subdivided into primary, secondary, and tertiary components (Fig. 2c). A significant reduction in computation is achieved by exploiting different time scales of these force components. The primary forces arise from nearest neighbors (within a cut-off distance, r_a) and they are updated after every MD time step. Compared to primary interactions, the secondary forces (within a cut-off distance, r_c) and tertiary forces (due to atoms beyond r_c but within the nearest-neighbor leaf cells) vary slowly. Therefore they are updated after n_1 (~ 10) and n_2 (~ 60) time steps, respectively. Between updates, the secondary and tertiary forces are calculated from Taylor's series. The far-field contribution to the Coulomb interaction is updated after every n_2 time steps.

2.2. *Curvilinear-coordinate approach to load balancing*

The original MRMD algorithm was implemented on parallel computers using uniform mesh decomposition of workloads [27], see Fig. 1b. The program has since been developed into an extensive software system for materials simulations augmented with various analysis tools [26,28,29]. However these software tools lacked load-balancing capability, and we have observed performance degradation when they were applied to irregular problems. Similar uniform mesh decomposition is currently used in a variety of parallel software systems. Examples include molecular dynamics [37], electronic-structure calculation [12,31,34], and computational fluid dynamics [38]. Our purpose is to develop a load balancer which can be added to these existing uniform-mesh-based softwares without modifying their data structures.

Let $\{\boldsymbol{x}_i | i = 1, \ldots, N\}$ denote a set of atomic coordinates distributed in a box formed by vectors $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$, see Fig. 3. With periodic boundary conditions, there are identical atoms at $\boldsymbol{x}_i + v$, where $v = l\boldsymbol{a} + m\boldsymbol{b} + n\boldsymbol{c}$ (l, m, and n are integers). Partitioning problem can be formulated as a mapping $p(x)$ from $\mathbf{R}^3$ to $\{0, 1, \ldots, P-1\}$ where P is the number of processors. The ith atom at position $\boldsymbol{x}_i$ is thus assigned to processor $p(\boldsymbol{x}_i)$. The problem is to find a map which minimizes the computational cost.

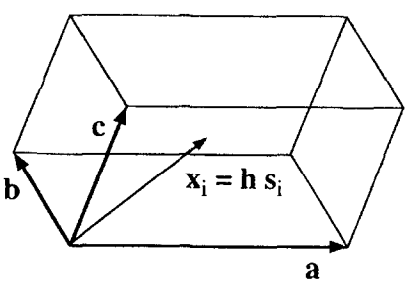

Fig. 3. An MD box is formed by three vectors, $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$. The three-dimensional coordinate, $\boldsymbol{x}_i$, of the ith atom is related to its dimensionless coordinate, s_i, through the relation, $\boldsymbol{x}_i = \boldsymbol{h}\boldsymbol{s}_i$, where $\boldsymbol{h} = (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ is the translation matrix.

We model the computational cost, T, as a linear combination of the load-imbalance cost, E_{bal}, and the communication cost, E_{com}, with machine-dependent prefactors, t_{bal} and t_{com}:

$$T = t_{\mathrm{bal}} E_{\mathrm{bal}} + t_{\mathrm{com}} E_{\mathrm{com}} \tag{2}$$

In Eq. (2), E_{bal} is the standard deviation of the number of atoms assigned to each processor. Communication cost, E_{com}, is the number of pairs whose elements are in different processors but within the cut-off length, r_{c}, of the short-range interaction [27]. This cost may be expressed as $E_{\mathrm{com}} = |\{(i, j)|\, p(\boldsymbol{x}_i) \neq p(\boldsymbol{x}_j),\ |\boldsymbol{x}_{ij}| < r_{\mathrm{c}}\}|/P$, where $|S|$ denotes the number of elements of set S. More practical definition of E_{com} is the average number of atoms per processor that are located within distance r_{c} from the processor boundaries, since the coordinates of these boundary atoms must be sent to neighbor processors in order to calculate interatomic interactions. The cost functions are thus expressed as

$$E_{\mathrm{bal}} = \left(\langle N_p^2 \rangle - \langle N_p \rangle^2\right)^{1/2}, \tag{3}$$

$$E_{\mathrm{com}} = \left\langle \sum_{i=1}^{N_p} f(\boldsymbol{x}_i, p) \right\rangle. \tag{4}$$

In Eqs. (3) and (4), the number of atoms, N_p, assigned to processor p is calculated as

$$N_p = \sum_{i=1}^{N} \delta_{p,p(\boldsymbol{x}_i)}, \tag{5}$$

where $\delta_{m,n} = 1$ for $m = n$ and 0 otherwise. The brackets denote an average over processors, e.g., $\langle N_p \rangle = \sum_{p=0}^{P-1} N_p/P$. The function $f(\boldsymbol{x}, p) = 1$ if $\boldsymbol{x}$ is within distance r_{c} from the subsystem boundary of processor p, and 0 otherwise.

We seek a load balancing scheme which retains the data structures of the original uniform mesh-partitioning scheme. Such a scheme can be achieved by using adaptive curvilinear coordinates. The main idea of our adaptive load-balancing scheme is to introduce a curvilinear coordinate system, ξ, which is related to the dimensionless atomic coordinate, s, by a mapping, $\xi(s)$. Here s is related to a physical atomic coordinate, $\boldsymbol{x}$, by $\boldsymbol{x} = \boldsymbol{h}s$, and the matrix, $\boldsymbol{h} = (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$, represents the simulation box (Fig. 3). We choose to represent $\xi(s)$ as [17],

$$\xi = s + \sum_{\boldsymbol{Q}} \boldsymbol{x}_{\boldsymbol{Q}} \exp(i\boldsymbol{Q}s), \tag{6}$$

where $\boldsymbol{Q} = 2\pi(l, m, n)$ (l, m, and n are integers) are reciprocal lattice vectors. The Fourier expansion in Eq. (6) is limited to $N_{\boldsymbol{Q}}$ plane waves; we consider only those $\boldsymbol{Q}$ which satisfy $|\boldsymbol{Q}| < Q_{\mathrm{c}}$, where Q_{c} is a cut-off wavenumber. This map conserves the periodicity of the system. Therefore, all the techniques related to periodic boundary conditions, including the RCMM and the Ewald summation, are straightforward to use with the new load balancer.

The system is partitioned into a uniform 3-dimensional mesh (more precisely torus because of periodic boundary conditions) in the ξ space. Processors are logically

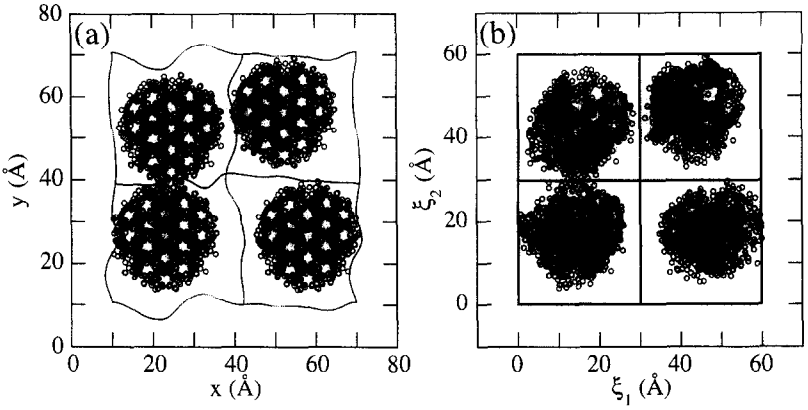

Fig. 4. Partition of 2,964-atoms into 2×2×1 processors: (a) curved partition boundaries in the Euclidean space; (b) uniform mesh partition in the curvilinear space. Circles are two dimensional projection of atoms, and solid curves represent a slice of partition boundaries.

arranged in a cubic array of dimensions $P_x \times P_y \times P_z$. An atom with curved coordinates, ξ_i, is assigned to a processor whose sequential ID, $p(\xi_i)$, is given by

$$p(\xi_i) = p_x(\xi_i) P_y P_z + p_y(\xi_i) P_z + p_z(\xi_i), \tag{7}$$

where

$$p_\alpha(\xi_i) = \lfloor \xi_{i\alpha} P_\alpha \rfloor, \quad (\alpha = x, y, z), \tag{8}$$

and $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. Such a uniform decomposition in the space generally results in a curved partition in the physical space, see Fig. 4.

Different sets of variational parameters $\{\boldsymbol{x}_Q\}$ in Eq. (6) lead to different partitions of workloads, and therefore to different computational costs. The parameters $\{\boldsymbol{x}_Q\}$ are chosen to minimize the cost, Eq. (2). Since subsystem boundaries are defined in the ξ space through Eq. (8), it is convenient to evaluate the cost function in the ξ space. However, the metric is different in this space, and therefore we must use a modified cut-off length, $r_c' = r_c \det(g_{ij})^{1/6}$, to define boundary atoms. An atom within distance r_c' from any partition boundary in the ξ space is counted as a boundary atom to be copied to neighbor processors. Here,

$$g_{ij} = \sum_{k=1}^{3} \frac{\partial \xi^k}{\partial x^i} \frac{\partial \xi^k}{\partial x^j}, \tag{9}$$

is evaluated at each atom's position.

The machine dependent parameters, t_{bal} and t_{com} in Eq. (2), characterize the computer system to be used. The parameter, t_{bal}, denotes the average processing time per atom per MD step, and this represents the performance of a processor. Since the time complexity of the MRMD algorithm scales linearly with the number of atoms on a sequential computer, this parameter can be measured in a test run on a single processor. The measured value (typically 10^{-4} s per atom per MD step on the IBM SP computer at

Argonne National Laboratory) in a sequential run is used in the load-balancing calculation in parallel MRMD simulations. The parameter, t_{com}, may be determined from the communication bandwidth of the system. For the SP system at Argonne, we use the theoretical bandwidth (35 MBytes/s) of the multistage Omega switch network [15]. In the MRMD algorithm, we exchange the coordinates of primary boundary atoms (which are located within a distance r_a from the processor boundaries) at each MD step. The coordinates and their time derivatives (up to the fifth order) of primary and secondary boundary atoms (which are located within r_c from the processor boundaries) are exchanged at every n_1 steps (typically $n_1 \sim 10$). On average, about five double-precision numbers are sent per boundary-atom per MD step, and this amounts to $t_{com} \sim 10^{-6}$ s.

In addition to communication bandwidth, latency also plays an essential role in determining the scalability of the program [32]. Our algorithm retains the topology of the uniform mesh partitioning so that the cost associated with the number of message passings is constant. Independent of the values of $\{x_Q\}$, each processor communicates with only six other processors in three-dimensional partitioning [27]. Naive communication strategy would require message passing to all the 26 neighbor processors, and an irregular partitioning scheme could result in even larger number of messages. Our adaptive curvilinear-coordinate load balancer minimizes the number of messages using a message-forwarding scheme [27], and it reduces the volume of each message by choosing optimal variational parameters $\{x_Q\}$. There is additional communication cost associated with the message passing of boundary cells which are used in the FMM approach. In materials simulations where each leaf cell contains over 10^2 atoms, however, the communication cost associated with atoms dominates the cost associated with multipoles [27].

Given atomic coordinates, $\{x_i\}$, the cost, $T(\{x_Q\}, \{x_i\})$, is minimized as a function of variational parameters, $\{x_Q\}$. We use the simulated annealing (SA) approach for this optimization problem [25]. The SA is based on the Monte Carlo (MC) method using Metropolis' algorithm. In each step of this algorithm, a small random displacement is given to x_Q, and the resulting change in the cost is estimated through Eqs. (2)–(4). If the change, δT, in the cost is negative, the displacement in x_Q is accepted; otherwise it is accepted with a probability, $\exp(-\delta T/\tau)$. The parameter, τ is called temperature in analogy with statistical mechanics; it controls the uphill search during optimization. Larger τ allows the cost to increase, and thus enables the search for a better global solution.

We found that the maximum displacement, δx_Q, allowed at each MC trial must scale with the magnitude of a wave vector, $|Q|$, to achieve faster convergence. Smaller wavenumbers represent global movement of partition boundaries, and accordingly larger displacements must be applied for them. We have chosen the following scaling function,

$$\delta x_Q = \frac{\delta x_0}{1 + \alpha |Q|}, \tag{10}$$

where the scaling factor, α, is dimensionless.

The scaling form, Eq. (10), may be justified by the equipartition principle for the underlying physical systems [1]. In an equilibrium system at temperature τ, an oscillator

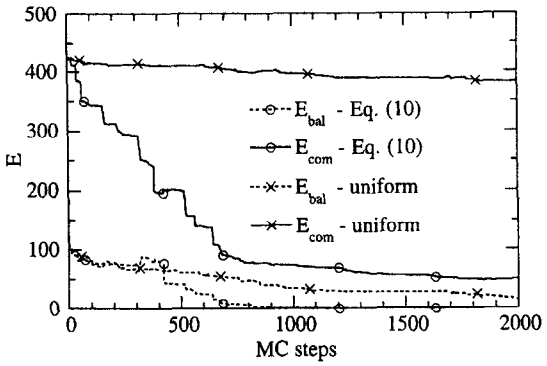

Fig. 5. Load-imbalance cost, E_{bal} (dotted curves), and communication cost, E_{com} (solid curves), as a function of MC steps. Scaled MC displacement (marked by circles) according to Eq. (10) achieves much faster convergence than uniform MC displacement (crosses).

with frequency ω has a root mean square displacement of $\sqrt{k_B T/m}/\omega$ where k_B is the Boltzmann constant and m is the mass of the oscillator. Assuming a sound-wavelike relation between the frequency and wavenumber, $\omega = c|\boldsymbol{Q}|$, the displacement is inversely proportional to $|\boldsymbol{Q}|$. Eq. (10) reflects the expected large amplitude of small-wavenumber oscillations, and at the same time it avoids the singularity at $\boldsymbol{Q} = 0$.

Fig. 4 shows an example of partitioning 2,964 atoms into $2 \times 2 \times 1$ processors. Fig. 5 shows the value of the cost function as a function of the number of MC steps during SA optimizations. We compare two scaling schemes for $\delta x_{\boldsymbol{Q}}$. The first scheme uses Eq. (10) with $\alpha = 1$ and $\delta x_0 = 2.0$ Å. In the second scheme, we use a uniform scaling, $\delta x_{\boldsymbol{Q}} = 0.1$ Å (taking other values such as $\delta x_{\boldsymbol{Q}} = 1.0$ Å or 0.05 Å results in slower convergence). The scaled scheme achieves a much faster convergence than the uniform displacement scheme. The total number of plane waves used for this example is $N_{\boldsymbol{Q}} = 377$, and the cut-off length is $r_c = 5.5$ Å.

The speed of a load balancer is greatly enhanced by applying it to a smaller system which approximates the original system. For example, materials consisting of silicon dioxide form a network of SiO_4 tetrahedral units. By simply representing workloads in terms of this larger unit instead of an individual atom, the problem size is reduced by a factor of three. Another example is a simulation of solids composed of many nanocrystals, where the shape of each crystalline is constructed by the so-called Wulff construction [43,44]. Even though the original MD simulations involve 10^2 clusters each containing $10^4 - 10^5$ atoms, we can apply the same Wulff construction algorithm to a smaller cluster (~ 10 atoms) to prepare a smaller system with a similar geometry. The variational parameters can be precomputed very quickly for this coarsened system. Coarse approximation to a graph has been used extensively for solving the graph partitioning problem [2,13,19,36] as well as the graph coloring problem [22].

2.3. Parallel implementation

The adaptive curvilinear-coordinate load balancer has been implemented on parallel computers using message-passing programming. We use the MPICH implementation of the message passing interface (MPI) standard [16].

We have implemented both: (i) a stand-alone load balancer; and (ii) a MRMD program with a dynamic load-balancing capability. In the former program, each processor owns a part of atomic coordinates that are assigned to it. The cost function is first calculated locally in each processor using its atomic coordinates. Global summation is then performed to calculate the total cost. When a MC displacement is accepted, assignment of atoms to processors is changed. Atoms migrate to proper processors by message passing according to the new assignment. The six-step message passing strategy is used for these migrations [27]. The outputs of this program are: (i) the final variational parameters, $\{x_Q\}$; and (ii) one atomic-coordinate file from each processor. These files are then used as inputs for parallel MRMD simulations.

The parallel MRMD program developed previously [27] has been modified to incorporate the new dynamic load-balancing scheme. The change in the program is minimal since most of the computation is done using the physical coordinates, x_i. The only subroutines to be modified are: (i) the subroutine for sorting out atoms which must be sent to neighbor processors for calculating interatomic interactions; and (ii) the subroutine for selecting atoms which have crossed the partition boundary and therefore must migrate to proper neighbors. Since the partition boundaries are defined in the ξ space, these subroutines need to perform the curvilinear transformation of Eq. (6). At every n_2 steps (we choose $n_2 = 60$) when we update the far-field contribution to the long range Coulomb interaction, we also update $\{x_Q\}$ to dynamically adjust the partition boundaries. The number of MC trials at every n_2 steps is N_{MC} (we choose $N_{MC} = 5$).

3. Numerical results

We have performed benchmark tests of the MRMD program augmented with the adaptive curvilinear-coordinate load balances. We have used the memory-bounded scaling where the number of atoms is linearly proportional to the number of processors [42]. We define the speed of a program as a product of the total number of atoms and time steps executed per second. The memory-bounded speedup is given by the ratio

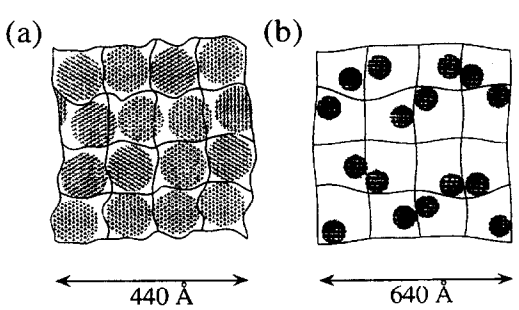

Fig. 6. (a) 3 Å slice of a high-density α-quartz nanocrystalline aggregate (mass density = 1.3 g/cm^3). Dots and solid curves represent atoms and partition boundaries, respectively. (b) 3 Å slice of a low-density α-quartz nanocrystalline aggregate (0.2 g/cm^3).

between the speed on P processors and that on one processor. The parallel efficiency, η, is the memory-bounded speedup divided by P.

We have performed two sets of benchmark tests on the 128-processor IBM SP computer at Argonne National Laboratory. In both cases, the simulated systems were aggregates of α-quartz nanoclusters. In the first set of atomic configurations, each processor on average was assigned 52,482 particles. The largest system was 3,358,848 atoms on 64 processors. The mass density was 1.3 g/cm^3 which is about half the normal α-quartz density, see Fig. 6a. In these systems, atomic distributions were less irregular because nanoclusters were densely packed. In the second set, each processor was assigned on average 19,803 atoms and the largest system contained 633,696 atoms on 32 nodes. The mass density of this set, 0.2 g/cm^3, was much lower than that of the first set. This low density made atomic distributions highly irregular, see Fig. 6b.

In both sets of benchmarks, nanoclusters were positioned randomly. We then applied the load-balancing program for a reduced systems with the same nanocrystalline geometry but each crystalline having about ten times smaller number of atoms. The precomputed parameters $\{x_Q\}$ for the smaller systems were used as initial values for the

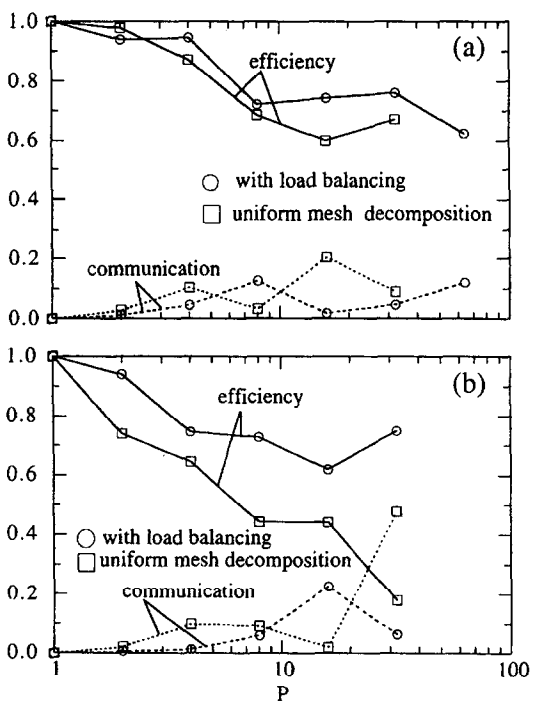

Fig. 7. (a) Parallel efficiency (solid curves) and communication overhead (dashed) of parallel MRMD algorithm for 1.3 g/cm^3 α-quartz nanocrystalline aggregates on the IBM SP machine at Argonne National Laboratory. Circles and squares represent the results with and without the adaptive curvilinear-coordinate load balancing, respectively. Because of less irregularity, the effect of load balancing is small. (b) The same as (a) for 0.2 g/cm^3 α-quartz nanocrystalline aggregates. Because of highly irregular cluster distribution, the load balancer significantly improves the parallel efficiency.

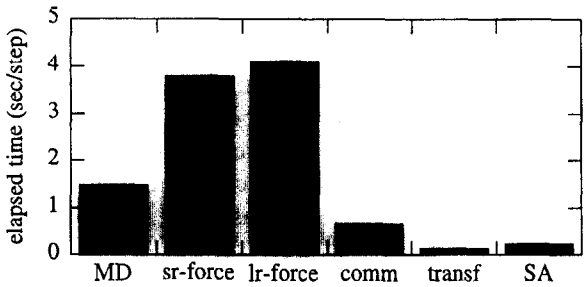

Fig. 8. Decomposition of the execution time per MD step (in seconds) into core computation tasks of the MRMD program with the curvilinear-coordinate load balancer. The system size is 633,696 atoms on 32 nodes of the SP machine. *MD* represents the integration of Newton's equations. *sr-force* and *lr-force* represent the calculation of short-range forces and long-range Coulomb forces, respectively. The copy operations of the boundary atoms and boundary cells are denoted by *comm. transf* is the curvilinear coordinate transformation necessary for load balancing. *SA* is the simulated annealing to minimize the computational cost.

benchmark of the parallel MRMD program with the larger systems. The number of plane waves was $N_Q = 93$.

Fig. 7a shows the parallel efficiency (solid curves) and communication overhead (dashed curves) for the higher-density (1.3 g/cm^3) systems. The results with and without the adaptive curvilinear-coordinate load balancer are represented by circles and squares, respectively. In these higher-density systems, uniform partitioning achieves a reasonable load balance. With the load balancer, however, the partition boundaries are curved to avoid passing through nanoclusters, and thus to reduce the communication cost. At this mass density, the effect of load imbalance is small. Fig. 7b shows the results for the lower-density (0.2 g/cm^3) systems. At this density, simulations without the load balancer are subjected to considerable load imbalance. Accordingly, lower efficiency is observed compared with that with the load balances. On 32 processors, inclusion of load balancing reduces the execution time by a factor of 4.2. Note that random generation of nanocrystalline positions causes the fluctuation in the performance curves shown in Fig. 7.

Fig. 8 shows the decomposition of the execution time per MD step (in seconds) into core computation tasks of the MRMD program which incorporates the curvilinear-coordinate load balancer. The system size is 633,696 atoms on 32 nodes of the SP computer. *MD* represents the integration of Newton's equations. *sr-force* and *lr-force* represent the calculation of short-range forces and long-range Coulomb forces, respectively. The copy operations for the boundary atoms and boundary cells are denoted by *comm. transf* is the curvilinear coordinate transformation necessary for load balancing. *SA* is the simulated annealing to minimize the computational cost. The overhead for load balancing is the sum of *transf* and *SA*, which amounts to 3.7% of the total elapsed time.

In this performance test, the variational parameters $\{x_Q\}$ are precomputed for a smaller representation of the system. The computation time for this precomputing (about an hour on one processor of Digital Alpha 2100 4/275) is negligible compared with 10^3 processor-hours on the SP system for a typical MD simulation for 10^5 steps. The timing

of the MRMD program includes simulating annealing procedure (5 MC steps at every 60 MD steps) for incrementally adjusting $\{x_Q\}$. Since any global conformation change in MD simulations takes 10^3–10^4 steps, this annealing schedule sustains reasonable dynamic load balancing.

4. Discussion

A new dynamic load balancing scheme has been developed for parallel multiresolution molecular dynamics simulations, based on adaptive curvilinear coordinates to represent partition boundaries for workloads. The curvilinear coordinate system is chosen to minimize load imbalance and communication costs using simulated annealing. The load balancer has been shown to improve parallel efficiency for highly irregular multimillion-atom systems.

In this paper, the cost function, T, is minimized as a function of the variational parameters, $\{x_Q\}$, by simulated annealing. However, the minimization may be done more efficiently by dynamical simulated annealing (DSA) [4] or other methods which use the derivative of the cost, $\delta T/\delta x_Q$ (such as the conjugate gradient method). This additional information can be utilized for achieving faster convergence compared with the SA. To take functional derivative of the cost function, we must extend Eqs. (4) and (5) in such a way that functions $\delta_{p,p(x)}$ and $f(x, p)$ take continuous values. A similar idea of using a continuous field (separator field) to represent partition boundaries has been used in a derivation of the spectral bisection method for the graph-partitioning problem [50]. We are currently implementing the DSA load-balancer based on a continuous valued cost function.

In this paper, we have used a heuristic scaling, Eq. (10), for MC displacement. When multiple length scales are involved in a complex optimization problem, however, selecting an optimal scaling form becomes a nontrivial problem. There has been a great advance in using the multigrid method to accelerate the convergence in such complex optimization problems [24]. Another effective method to cope with multiple length scales is multiresolution analysis using wavelets [6]. In particular, an abrupt change in atomic distributions (and consequently in partition boundaries) in a localized region is compactly represented by using wavelets rather than plane waves. We are exploring the possibilities to combine the multigrid method and wavelets with the adaptive curvilinear-coordinate load balancer.

Acknowledgements

This work was supported by Army Research Office, National Science Foundation CAREER Program, Petroleum Research Fund, and Louisiana Education Quality Support Fund (LEQSF). T.C. was supported by National Science Foundation Graduate Trainee-

ships Program, Grant No. 9355007, under the supervision of Dr. Rajiv K. Kalia and Dr. Priya Vashishta of the Concurrent Computing Laboratory for Materials Simulations (CCLMS) of Louisiana State University. A part of the numerical experiments were performed on the 128-node IBM SP computer at Argonne National Laboratory. The computations were also performed on parallel machines at the CCLMS. The facilities in the CCLMS were acquired with the Equipment Enhancement Grants awarded by the Louisiana Board of Regents through LEQSF. We wish to thank Dr. Tsuruta for his help in preparing Fig. 1b.

References

[1] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Oxford University Press, Oxford, 1987, p. 46.
[2] S.T. Barnard, H.D. Simon, Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems, Concurrency 6 (1994) 101–117.
[3] J.E. Boillat, Load balancing and Poisson equation in a graph, Concurrency 2 (1990) 289–313.
[4] R. Car, M. Parrinello, Unified approach for molecular dynamics and density-functional theory, Phys. Rev. Lett. 55 (1985) 2471–2474.
[5] G. Cybenko, Dynamic load balancing for distributed memory multiprocessors, J. Par. Distrib. Comput. 7 (1989) 279–301.
[6] I. Daubechies, Ten Lectures on Wavelets, SIAM, Philadelphia, 1992.
[7] Y. Deng, R.A. McCoy, R.B. Marr, R.F. Peiers, O. Yasar, Molecular dynamics for 400 million particles with short-range interactions, in: Supercomputing '94, IEEE Comp. Soc., Washington, DC, 1994.
[8] H.-Q. Ding, N. Karasawa, W.A. Goddard, The reduced cell multipole method for Coulombic interactions in periodic systems with million-atom unit cell, Chem. Phys. Lett. 196 (1992) 6–10.
[9] G.C. Fox, R.D. Williams, P.C. Messina, Parallel Computing Works, Morgan Kaufmann, San Francisco, 1994, ch. 11.
[10] J. Fricke, SiO_2-aerogel: Modifications and applications, J. Non-Cryst. Solids 121 (1990) 188–192.
[11] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, PVM, MIT Press, Cambridge, 1994.
[12] M.J. Gillan, First-principles calculations on materials using massively parallel computing, Future Generation Comput. Syst. 10 (1994) 213–222.
[13] A.Y. Grama, V. Kumar, A. Sameh, Scalable parallel formulation of the Barnes-Hut method for N-body simulations, in: Supercomputing '94, IEEE Comp. Soc., Washington, DC, 1994.
[14] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (1987) 325–348.
[15] W. Gropp, E. Lusk, User's guide for the ANL IBM SPx, Technical Report, Argonne National Laboratory, 1994, http://www.mcs.anl.gov/Projects/sp/guide-r2/guide-r2.html.
[16] W. Gropp, E. Lusk, A. Skjellum, Using MPI, MIT Press, Cambridge, 1994.
[17] F. Gygi, Electronic-structure calculations in adaptive coordinates, Phys. Rev. B 48 (1993) 11692–11700.
[18] A. Heirich, S.A. Taylor, A parabolic load balancing method, in: Proceedings of the 24th International Conference on Parallel Processing, CRC Press, New York, 1995, pp. 192–202.
[19] B. Hendrickson, R. Leland, An improved spectral graph partitioning algorithm for mapping parallel computations, SAND92-1460, Sandia National Laboratory, 1992.
[20] G. Horton, A multi-level diffusion method for dynamic load balancing, Parallel Comput. 19 (1993) 209–218.
[21] Y.-S. Hwang, R. Das, J.H. Saltz, A data-parallel implementation of molecular dynamics programs for distributed memory machines, in: S. Ranka (Ed.), Proceedings of Workshop on Solving Irregular Problems on Distributed Memory Machines, Santa Barbara, 1995, pp. 28–34.

[22] M.T. Jones, P.E. Plassmann, Computation of equilibrium vortex structures for type-II superconductors, Int. J. Supercomputer Appl. 7 (2) (1993) 129–143.
[23] R.K. Kalia, S.W. de Leeuw, A. Nakano, P. Vashishta, Molecular-dynamics simulations of Coulombic systems on distributed-memory MIMD machines, Comput. Phys. Commun. 74 (1993) 316–326.
[24] D. Kandel, E. Domany, D. Ron, A. Brandt, E. Loh, Simulations without critical slowing down, Phys. Rev. Lett. 60 (1988) 1591–1594.
[25] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[26] A. Nakano, L. Bi, R.K. Kalia, P. Vashishta, Structural correlations in porous silica: Molecular dynamics simulation on a parallel computer, Phys. Rev. Lett. 71 (1993) 85–88.
[27] A. Nakano, R.K. Kalia, P. Vashishta, Multiresolution molecular dynamics algorithm for realistic materials modeling on parallel computers, Comput. Phys. Commun. 83 (1994) 197–214.
[28] A. Nakano, R.K. Kalia, P. Vashishta, Growth of pore interfaces and roughness of fracture surfaces in porous silica: Million particle molecular-dynamics simulations, Phys. Rev. Lett. 73 (1994) 2336–2339.
[29] A. Nakano, R.K. Kalia, P. Vashishta, Dynamics and morphology of brittle cracks: A molecular-dynamics study of silicon nitride, Phys. Rev. Lett. 75 (1995) 3138–3141.
[30] A. Nakano, P. Vashishta, R.K. Kalia, Parallel multiple-time-step molecular dynamics with three-body interaction, Comput. Phys. Commun. 77 (1993) 303–312.
[31] J.S. Nelson, S.J. Plimpton, M.P. Sears, Plane-wave electronic-structure calculations on a parallel supercomputer, Phys. Rev. B (1993) 1765–1774.
[32] N. Nupairoj, L. Ni, Performance evaluation of some MPI implementations on workstation clusters, in: Scalable Parallel Library Conference '94, Mississipi State University, 1994, pp. 98–105.
[33] C. Ou, S. Ranka, G. Fox, Fast mapping and remapping algorithm for irregular and adaptive problems, in: Proceedings of The 1993 International Conference on Parallel and Distributed Systems, Taipei, 1993, pp. 279–283.
[34] M.C. Payne, M.P. Teter, D.C. Allan, T.A. Arias, J.P. Joannopoulos, Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients, Rev. Mod. Phys. 64 (1992) 1045–1097, Sec. X.
[35] J.R. Pilkington, S.B. Baden, Dynamic partitioning of non-uniform structured workloads with spacefilling curves, IEEE Trans. on Parallel and Distributed Systems, to appear.
[36] R. Ponnusamy, N. Mansour, A. Choudhary, G.C. Fox, Graph contraction and physical optimization methods: A quality-cost tradeoff for mapping data on parallel computers, in: International Conference on Supercomputing, Tokyo, 1993.
[37] D.C. Rapaport, Multi-million particle molecular dynamics II. design considerations for distributed processing, Comput. Phys. Commun. 62 (1991) 217–228.
[38] C.J. Ribbens, A moving mesh scheme for adaptive domain decomposition, in: P. Mehrotra, J. Saltz, R. Voigt (Eds.), Unstructured Scientific Computation on Scalable Multiprocessors, MIT Press, Cambridge, 1991, pp. 205–219.
[39] R.W. Siegel, in: F.E. Fujita (Ed.), Physics of New Materials, Springer-Verlag, Heidelberg, 1994, p. 65.
[40] J.P. Singh, C. Holt, J.L. Hennessy, A. Gupta, A parallel adaptive fast multipole method, in: Supercomputing '93, IEEE Comp. Soc., Washington, DC, 1993, pp. 54–65
[41] W.B. Streett, D.J. Tildesley, G. Saville, Multiple time-step methods in molecular dynamics, Mol. Phys. 35 (1978) 639–648.
[42] X.-H. Sun, J. Gustafson, Toward a better parallel performance metric, Parallel Comput. 17 (1991) 1093–1109.
[43] K. Tsuruta, A. Omeltchenko, R.K. Kalia, P. Vashishta, Early stage of sintering of silicon nitride clusters: A molecular dynamics study on parallel machines, Europhys. Lett. 33 (1996) 441–446.
[44] R.K. Kalia, A. Nakano, A. Omeltchenko, K. Tsuruta, P. Vashishta, Role of ultrafine microstructures in dynamic fracture in nanophase silicon nitride, Phys. Rev. Lett. 78 (1997) 2144–2147.
[45] P. Vashishta, R.K. Kalia, S.W. de Lecuw, D.L. Greenwell, A. Nakano, W. Jin, J. Yu, L. Bi, W. Li, Computer simulation of materials using parallel architectures, Comput. Mater. Sci. 2 (1994) 180–208.
[46] P. Vashishta, R.K. Kalia, I. Ebbsjo, Low-energy floppy modes in high-temperature ceramics, Phys. Rev. Lett. 75 (1995) 858–861.

[47] P. Vashishta, R.K. Kalia, J.P. Rino, I. Ebbsjo, Interatomic potential for SiO_2: A molecular dynamics study of structural correlations, Phys. Rev. B 41 (1991) 12197–12209.
[48] M.S. Warren, J.K. Salmon, Astrophysical *N*-body simulations using hierarchical data structures, in: Supercomputing '92, IEEE Comp. Soc., Washington, DC, 1992, 570-576.
[49] M.S. Warren, J.K. Salmon, A portable parallel particle program, Comput. Phys. Commun. 87 (1995) 266–290.
[50] R.D. Williams, Peformance of dynamic load balancing algorithms for unstructured mesh calculations, Concurrency 3 (1991) 457–481.