CONCURRENCY: PRACTICE AND EXPERIENCE
Concurrency: Pract. Exper., Vol. **11**(7), 343–353 (1999)

Multiresolution load balancing in curved space: the wavelet representation

AIICHIRO NAKANO*
Department of Computer Science, Concurrent Computing Laboratory for Materials Simulations, Louisiana State University, Baton Rouge, LA 70803-4020, USA
(e-mail: nakano@bit.csc.lsu.edu)

SUMMARY
A new load-balancing scheme based on a multiresolution analysis is developed for parallel particle simulations. Workloads are partitioned with a uniform 3-dimensional mesh in an adaptive curvilinear co-ordinate system which is represented by a wavelet basis. Simulated annealing is used to determine the optimal wavelet coefficients which minimize load imbalance and communication costs. Performance tests on a parallel computer involving up to 1.04 billion particles demonstrate the scalability of the new load balancer. Copyright © 1999 John Wiley & Sons, Ltd.

1. INTRODUCTION

Molecular dynamics (MD) simulation is rapidly becoming an integral part of computer-aided material design[1]. Recent developments in parallel computing technologies and multiresolution numerical methods have enabled multimillion-atom MD simulations of realistic materials[2]. A large-scale MD approach on parallel computers has been established as a new research mode for studying how atomistic processes are related to macroscopic materials phenomena[3].

Parallel MD simulations with irregular atomic distributions (such as fracture of materials) are accompanied by the problem of load imbalance[4]. Suppose that processors are logically organized as a 3-dimensional array of dimensions $P_1 \times P_2 \times P_3$, 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. Various load balancing schemes have been developed for irregular problems on parallel computers: recursive bisection[5]; spectral partitioning[5–7]; spacefilling-curve methods[8–12]; and load-diffusion[13–16] and load-pressure[12,17] schemes.

Previously we developed a low-overhead load balancer which can be embedded in existing parallel MD programs based on regular mesh decomposition with little programming effort[18]. The method uses adaptive curvilinear co-ordinates[19] to

*Correspondence to: A. Nakano, Department of Computer Science, Concurrent Computing Laboratory for Materials Simulations, Louisiana State University, Baton Rouge, LA 70803-4020, USA.
Contract/grant sponsor: National Science Foundation; Contract/grant number: ASC-9701504.
Contract/grant sponsor: Army Research Office; Contract/grant number: DAAH04-96-1-0393.
Contract/grant sponsor: Petroleum Research Fund; Contract/grant number: 31659-AC9.
Contract/grant sponsor: Louisiana Education Quality Support Fund; Contract/grant number: LEQSF(96-99)-RD-A-10.

CCC 1040–3108/99/070343–11$17.50

Received 25 August 1997
Revised 28 October 1998

represent partition boundaries. Workloads are partitioned with a uniform 3-dimensional mesh in the curvilinear co-ordinate system. Simulated annealing (SA)[20] is used to determine the optimal co-ordinate system which minimizes load imbalance and communication costs. For a 0.63 million atom irregular SiO_2 system on a 32-processor IBM SP1, inclusion of the load balancer reduced the execution time of an MD program by a factor of 4.2. Due to the underlying regular mesh topology, the number of messages required per MD step is only six. In addition, periodic boundary conditions are naturally incorporated in the load-balancing scheme.

In our previous paper, the plane-wave basis was used to represent the co-ordinate transformation[18]. Recently, various multiresolution schemes have been demonstrated to be effective for the load-balancing problem[6,7]. In this paper, we extend our load balancer by applying a multiresolution analysis based on wavelets[21] to the co-ordinate transformation. In contrast to the plane-wave basis, the multiresolution analysis facilitates successive coarse approximations to achieve a compact representation of partition boundaries. In addition, the wavelet basis has a better spatial locality, leading to a superior scalability on massively parallel computers. In Section 2, we describe the wavelet-based load-balancing scheme. The numerical tests are described in Section 3, and Section 4 contains the conclusions.

2. METHODS

Molecular dynamics (MD)[1] is a typical particle-type simulation method in which a physical system consisting of N atoms is represented by a set of atomic co-ordinates, $\{\vec{x}_i \mid i = 1, \ldots, N\}$. If periodic boundary conditions are imposed on a cubic box with length L, there are identical atoms at $\vec{x}_i + \vec{v}_i$, where $\vec{v}_i = L(l, m, n)(l, m, n \in \mathbf{Z})$. The partitioning problem can be formulated as a mapping $p(\vec{x})$ from $\mathbf{R}^3$ to $\{0, 1, \ldots, P-1\}$, where P is the number of processors. The ith atom at position $\vec{x}_i$ is thus assigned to processor $p(\vec{x}_i)$. The problem is to find a mapping which minimizes the computational cost.

The most time-consuming part of MD simulations is the calculation of the potential energy, $V(\{\vec{x}_i\})$, which in our case involves summations over atomic pairs and triples[1]. In our multiresolution molecular dynamics (MRMD) program[2], the long-range electrostatic potential is computed with the Fast Multipole Method (FMM)[22]. In the FMM, the simulation box is recursively divided into smaller cells, generating a tree structure. The root of the tree is at level 0, and it corresponds to the entire simulation box. A parent cell at level l is decomposed into $2 \times 2 \times 2$ children cells of equal volume at level $l + 1$. The recursive decomposition stops at the leaf level, $l = l_L$, so that there are $2^{l_L} \times 2^{l_L} \times 2^{l_L}$ leaf cells. The potential energy is then decomposed into the near-field and far-field contributions. The near-field contribution is due to atoms in nearest-neighbor cells, and it is calculated directly by performing the pair and triple sums. The far-field contribution is evaluated by expanding the electrostatic field in terms of multipoles. The multipoles as well as the Taylor expansion of the field are computed recursively on the tree in time proportional to N. On parallel computers, the near-field calculation takes $O(N/P)$ time and involves only the nearest-neighbor communications. The far-field calculation, on the other hand, involves computation and global communication which scale as $O(\log P)$. For our coarse-grained applications ($N/P \sim 10^5$), the $O(N/P)$ contribution dominates

the total execution time[2]. Therefore, we can estimate the computational cost accurately based on the number of atoms assigned per processor.

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}:

$$T = t_{\mathrm{bal}} E_{\mathrm{bal}} + t_{\mathrm{com}} E_{\mathrm{com}} \tag{1}$$

In (1), E_{bal} is the standard deviation of the number of atoms assigned to each processor:

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

where N_p is the number of atoms assigned to processor p. The brackets in (2) denote an average over processors, e.g. $\langle N_p \rangle = \sum_p N_p / P$. (A more common definition of the load-imbalance cost is the maximum number of atoms assigned to any one processor.) The 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 nonelectrostatic interaction[1]. The communication cost may be expressed as $E_{\mathrm{com}} = |\{(i, j) \mid p(\vec{x}_i) \neq p(\vec{x}_j), |\vec{x}_i - \vec{x}_j| < r_{\mathrm{c}}\}|/P$, where $|S|$ denotes the number of elements of set S. A 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 co-ordinates of these boundary atoms must be sent to neighbor processors in order to compute interatomic interactions. The communication cost is thus expressed as

$$E_{\mathrm{com}} = \left\langle \sum_{i=1}^{N_p} f(\vec{x}_i, p) \right\rangle \tag{3}$$

The function $f(\vec{x}, p) = 1$ if $\vec{x}$ 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 uniform-mesh partitioning scheme. Such a scheme can be realized by using adaptive curvilinear co-ordinates. The main idea of our adaptive load-balancing scheme is to introduce a curvilinear-co-ordinate system, $\vec{\xi}$, which is related to the atomic co-ordinate, $\vec{x}$, by a mapping

$$\vec{\xi}(\vec{x}) = \vec{x} + \vec{u}(\vec{x}) \tag{4}$$

where $\vec{u}(\vec{x})$ is the displacement field.

In our previous paper[18], the displacement field was represented using the plane-wave basis,

$$\vec{u}(\vec{x}) = \sum_{\vec{Q}} \vec{x}_{\vec{Q}} \exp(i \vec{Q} \cdot \vec{x}) = \mathcal{F}^{-1}[\vec{x}_{\vec{Q}}] \tag{5}$$

where $\vec{Q}_i = 2\pi(l, m, n)/L (l, m, n \in \mathbf{Z})$ are wave vectors, $\vec{x}_{\vec{Q}}$ are plane-wave coefficients, and $\mathcal{F}^{-1}$ denotes the inverse Fourier transformation. The Fourier expansion in (5) is limited to N_Q plane waves; we consider only those $\vec{Q}$ which satisfy $|\vec{Q}| < Q_{\mathrm{c}}$, where Q_{c} is a cut-off wavenumber. This mapping conserves the periodicity of the system.

In this paper, the displacement field is instead represented by a multiresolution analysis using wavelets[21],

$$\vec{u}(\vec{x}) = \sum_j \vec{d}_j \Psi_j(\vec{x}) = \mathcal{W}^{-1}[\vec{d}_j] \tag{6}$$

where $\Psi_j(\vec{x}) = \Psi_{l_1,m_1}(x_1)\Psi_{l_2,m_2}(x_2)\Psi_{l_3,m_3}(x_3)$ is a product of one-dimensional wavelets, $j = (l_1, m_1, l_2, m_2, l_3, m_3)$, and $\mathcal{W}^{-1}$ denotes the inverse wavelet transformation. Here $\Psi_{l,m}(x)$ collectively denotes scaling functions, $\phi_{l,m}(x) = 2^{l/2}\phi(2^l x - m)$, and wavelets, $\Psi_{l,m}(x) = 2^{l/2}\Psi(2^l x - m)$, where l and m are dilation and translation indices[21]. A vector space $V_l = \text{span}\{\phi_{lm}(x) \mid m = 1, \ldots, 2^l\}$ can be decomposed into two orthogonal subspaces, $V_l = V_{l-1} \oplus W_{l-1}$, where $W_l = \text{span}\{\psi_{lm}(x) \mid m = 1, \ldots, 2^l\}$. Recursive application of this decomposition leads to a multiresolution representation, $V_{l_{\max}} = V_0 \oplus W_0 \oplus W_1 \oplus \cdots \oplus W_{l_{\max}-1}$[21]. Equation (6) embodies this multiresolution representation for the curvilinear co-ordinate transformation. As a one-dimensional wavelet basis, we use the Daubechies 4 function[23]. The discrete wavelet transform is performed on $2^{l_{\max}} \times 2^{l_{\max}} \times 2^{l_{\max}}$ mesh points.

The physical system is partitioned into a uniform 3-dimensional mesh (more precisely torus because of periodic boundary conditions) in the ξ space. Processors are logically arranged in a 3-dimensional array of dimensions $P_1 \times P_2 \times P_3$. An atom with curved co-ordinates $\vec{\xi}_i$ is assigned to a processor whose sequential ID, $p(\vec{\xi}_i)$, is given by

$$p(\vec{\xi}_i) = p_1(\xi_{i1})P_2P_3 = p_2(\xi_{i2})P_3 + p_3(\xi_{i3}) \tag{7}$$

where

$$p_\alpha(\xi_{i\alpha}) = \lfloor \xi_{i\alpha} P_\alpha / L \rfloor, \quad \alpha = 1, 2, 3 \tag{8}$$

and $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. Such a uniform decomposition in the ξ space generally results in a curved partition in the Euclidean space (see Figure 1).

Different sets of expansion coefficients, $\{\vec{d}_j\}$ in equation (6), lead to different partitions of workloads, and accordingly to different computational costs. These parameters are chosen to minimize the cost, (1), as a functional of $\vec{d}_j$. Since subsystem boundaries are defined in the ξ space through (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, the metric tensor,

$$g_{\alpha\beta} = \sum_{\gamma=1}^{3} \frac{\partial \xi^\gamma}{\partial x^\alpha} \frac{\partial \xi^\gamma}{\partial x^\beta} \tag{9}$$

is evaluated at each atom's position.

The machine-dependent parameters, t_{bal} and t_{com} in (1), characterize the computer system under consideration. The parameter t_{bal} is the average processing time per atom per MD step, and this represents the performance of a processor. Since the time complexity of the MD 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

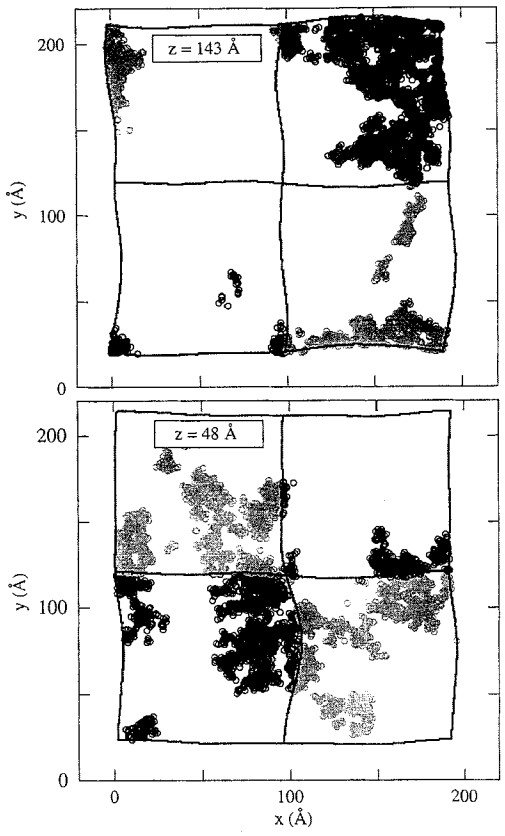

Figure 1. Partition of a 41,472*-atom porous silica system into* $2 \times 2 \times 1$ *processors. Two* 20 *Å-thick slices of the system at* $z = 48$ *Å and* 143 *Å are shown. Circles are two-dimensional projection of atoms, and solid curves represent a slice of partition boundaries*

(typically 5×10^{-5} s per atom per MD step on an IBM SP2 computer) in a sequential run is used in the load-balancing calculation in parallel MD simulations. The parameter t_{com} may be determined from the communication bandwidth of the system. For the SP2 system, the asymptotic bandwidth of the multistage Omega switch network is $r_\infty = 35$ MB/s. In the MD algorithm, we exchange the information on boundary atoms at each MD step. Four double-precision (three co-ordinates and a charge) and one integer (atomic species) numbers are sent per boundary atom per MD step, and this amounts to $t_{\text{com}} \sim 10^{-6}$ s using r_∞. However, typical applications achieve a fraction of the asymptotic bandwidth, and we use $t_{\text{com}} = 5 \times 10^{-6}$ s in the following analysis.

In addition to communication bandwidth, latency also plays an essential role in determining the scalability of the program. 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 variational parameters $\{\vec{d}_j\}$, each processor communicates with only six other processors in three-dimensional partitioning[18].

Given atomic co-ordinates, $\{\vec{x}_i\}$, the cost, T, is minimized as a function of the variational parameters $\{\vec{d}_j\}$. We use the simulated annealing (SA) approach for this optimization


```
(a) Single simulated-annealing step

randomly select a wavelet index, j
generate a random displacement, δd_j
compute the cost change, δT = T (d_j + δd_j) - T (d_j)
if exp(-δT/τ) > random_number then
  accept the displacement, d_j ← d_j + δd_j
  for each atom, i
    ξ_i = x_i + (W^-1 d)(x_i)
    if p(ξ_i) ≠ this_processor then
      add i to the list of atoms to be migrated
    endif
  endfor
  migrate the atoms whose processor-assignment have changed
endif


(b) Calculation of the computational cost

for each processor, p
  local contribution to the number of atoms per processor, n_p = 0
endfor
the number of atoms near the boundary of this_processor, n_b = 0
for each atom, i
  ξ_i = x_i + (W^-1 d)(x_i)
  compute the scaled cut-off length, r_c' = r_c det(g_αβ)^1/6, at x_i
  if ξ_i is within r_c' from the processor boundary then
    n_b ← n_b + 1
  endif
  n_p ← n_p + 1 for p = p(ξ_i)
endfor
for each processor, p
  compute the global summation, N_p, of n_p
endfor
E_bal = ( <N_p^2> - <N_p>^2 )^1/2
compute the global summation, N_b, of n_b
E_com = N_b/P
return the cost T = t_bal E_bal + t_com E_com
```

Figure 2. Algorithms for: (a) a single simulated-annealing (SA) step of the wavelet-based load balancer; (b) computing the computational cost

problem[20]. The SA is based on the Monte Carlo (MC) method using the Metropolis algorithm. In each step of this algorithm, a small random displacement is given to a variational parameter, and the resulting change in the cost is estimated through (1)–(3). If the change, δT, in the cost is negative, the displacement in $\{\vec{d}_j\}$ 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; a larger τ allows the cost to increase, and thus enables the search for a better global solution.

We found that the maximum displacement $\delta d_{j\alpha}$, $\alpha = 1, 2, 3$, allowed at each MC trial must scale with the dilation indices l_1, l_2 and l_3, to achieve faster convergence. Smaller dilation indices represent global movement of partition boundaries, and accordingly larger displacements must be applied for them. We have chosen the scaling function, $\delta d_{j\alpha} = d_0/2^{l*}$, where $l_*^2 = l_1^2 + l_2^2 + l_3^2$.

In our previous paper, the SA procedure was initialized with all expansion coefficients as zero. Noting that the load-imbalance cost may be reduced by expanding the space where particle concentration is high, we propose the following analytic formula[24] for initializing the wavelet coefficients:

$$\vec{u}(\vec{x}) = \sum_{\vec{y}} A(\vec{x} - \vec{y}) \operatorname{sech}\left(\frac{|\vec{x} - \vec{y}|}{a}\right) \rho(\vec{y}) \tag{10}$$

where A and a are adjustable parameters. The density distribution, $\rho(\vec{y})$, of atoms is estimated at the $2^{l_{\max}} \times 2^{l_{\max}} \times 2^{l_{\max}}$ mesh points used in the discrete wavelet transformation.

The adaptive curvilinear-co-ordinate load balancer has been implemented on parallel computers using the Message Passing Interface (MPI) standard[25]. Each processor owns a subset of atomic co-ordinates that are assigned to it. The cost function is first calculated locally in each processor using its atomic co-ordinates. Global summation is then performed to calculate the total cost. When an MC displacement is accepted, assignment of atoms to processors may be changed. Atoms migrate to proper processors by message passing according to the new assignment. The six-stage message passing scheme is used for these migrations[2,18]. The outputs of this program are: (i) a file containing the final variational parameters; and (ii) one atomic-co-ordinate file per processor. Figure 2(a) shows the algorithm for a single simulated-annealing step, where the range of *random_number* is [0, 1]. The parallel program is written in the single-program multiple-data (SPMD) style, and *this_processor* denotes the processor ID of each processor. Figure 2(b) shows the algorithm for calculating the computational cost.

The new dynamic load-balancing scheme has also been embedded into a parallel MRMD program developed previously[2]. The change in the program is minimal since most of the computation is done using the physical co-ordinates $\vec{x}_i$. 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 (4). At every n_2 steps (typically $n_2 = 60$) when we update the far-field contribution to the long-range Coulomb interaction[2], we also update the variational parameters to dynamically adjust the partition boundaries. The number of MC trials at every n_2 steps is N_{MC} (typically $N_{\mathrm{MC}} = 5$).

3. NUMERICAL RESULTS

We first compare the performance of the wavelet-based load balancer to that with plane waves. Figure 3 shows the total cost as a function of the elapsed time during the SA procedure. We use $t_{\mathrm{bal}} = 5 \times 10^{-5}$ s and $t_{\mathrm{com}} = 5 \times 10^{-6}$ sec. The system is a porous silica with the mass density of 0.2 g/cm^3 consisting of 41,472 atoms. The length of the simulation box is $L = 190.4$ Å, and the interaction cut-off is $r_c = 5.5$ Å. We use four nodes of the IBM SP2 computer at the Maui High Performance Computing Center. The maximum level in the multiresolution analysis is $l_{\max} = 3$, and the number of plane waves is $N_Q = 123$ ($Q_c = 0.1$ Å^{-1}). The adjustable parameters are chosen as $d_0 = 1024$ Å and $\tau = 0.1$. In the absence of load-balancing, the four processors own 6455, 12,549, 12,652 and 9816 atoms, respectively. This amounts to a 22% increase of the MD computation time

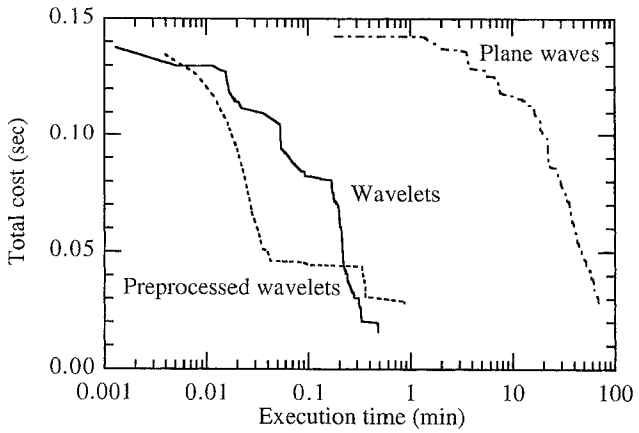

Figure 3. Total computational cost, T, as a function of the execution time. The solid and dash-dotted curves represent results with the wavelet basis and the plane-wave basis, respectively. The dashed curve represents the preprocessed results with the wavelet basis

due to load imbalance. With either wavelets or plane waves, the workloads are balanced within a few atoms. In addition, the communication cost is reduced significantly during the SA procedure. Figure 3 shows that the wavelet-based load balancer converges much faster than that based on plane waves.

Figure 3 also shows the effect of preprocessing. We choose the adjustable parameters in (10) as $A = 0.1$ and $a = 5.5$ Å. Initial convergence with the preprocessing is much faster than that of the wavelet-based load balancer without preprocessing. However, the preprocessed solution is trapped in a local minimum, and the final cost is lower in the case without the preprocessing. Therefore the preprocessing may be used to obtain a low-quality partition of workloads in a very short time.

We have performed benchmark tests of the parallel load balancer on the IBM SP2 system at Maui. The simulated systems are aggregates of silicon nitride nanoclusters. We use the memory-bounded scaling where the number of atoms is linearly proportional to the number of processors[26]; each processor on average is assigned 16,301,600 atoms. The largest system is 1,043,302,400 atoms on 64 processors. Figure 4(a) shows the elapsed time (circles) and the communication time (squares) per SA step. The execution time increases very little as the system size is increased, indicating a high parallel efficiency. The speed of the load balancer is computed as the number of atoms times the number of SA steps executed per unit time. The memory-bounded speedup is given by the ratio between the speed on P processors and that on one processor. The parallel efficiency, E, is the memory-bounded speedup divided by P. Figure 4(b) shows the parallel efficiency (circles) and communication overhead (squares). For the largest system with 1.04 billion atoms on 64 processors, $E = 0.915$, and the communication overhead is 2.6%. The load balancer is thus scalable up to billion-atom systems.

In order to be a practical load balancer, the time to perform load balancing must be short compared with the execution time of the MRMD program. Table 1 shows that the execution time per SA step of the load-balancing program is much shorter than that of the MRMD program per MD step. Since we only perform N_{MC} (= 5) SA iterations at every

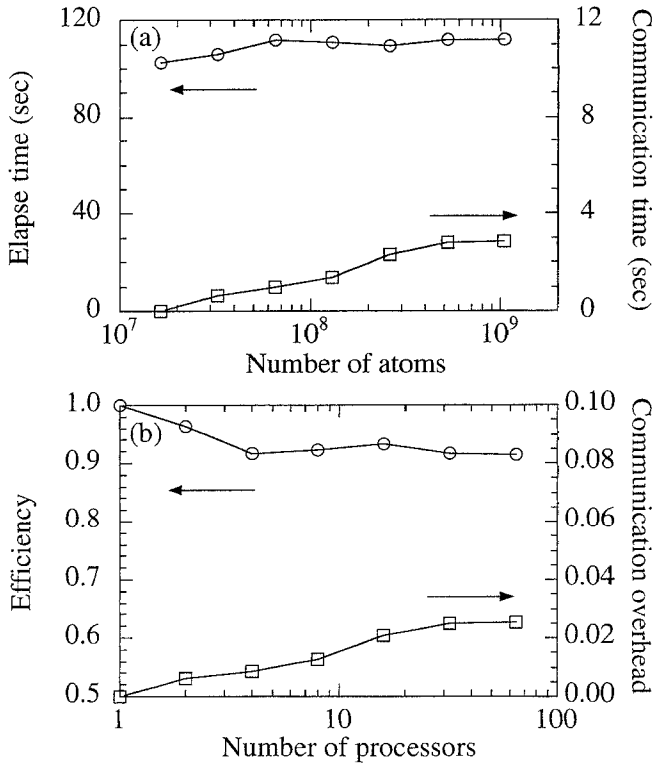

Figure 4. Performance of the wavelet-based load balancer on an IBM SP2 computer: (a) elapsed time (circles) and time spent for communication (squares) per MC step as a function of number of atoms, N. The size of the silicon nitride system scales linearly with the number of processors, p : N = 16,301,600 p; (b) parallel efficiency (circles) and communication overhead (squares) as a function of p

Table 1. Execution and communication times of the multiresolution molecular dynamics (MRMD) program and the wavelet-based load balancer per iteration for a 50.3 million atom SiO_2 system on 64 IBM SP2 nodes

	Elapsed time, s	Communication time, s
Multiresolution molecular dynamics	103.1	1.89
Wavelet-based load balancing	4.9	0.13

n_2 (= 60) MD steps, the time to perform load balancing is negligible compared with the MD computations.

4. CONCLUSIONS

In this paper, a new dynamic load balancing scheme has been developed for parallel particle simulations based on adaptive curvilinear co-ordinates to represent partition boundaries. The curvilinear co-ordinate system has been chosen to minimize load-

imbalance and communication costs using simulated annealing. Multiresolution analysis based on wavelets has sped up the minimization significantly. An efficient preprocessing scheme for the wavelet coefficients has also been proposed. Performance tests involving up to 1.04 billion particles have demonstrated the scalability of the new load balancer. The new scheme is easily embedded in existing parallel programs based on uniform mesh decomposition, and it is effective for periodically extended systems with fractal-like irregular structures. Therefore it will complement existing load-balancing schemes[4–17] as a low-overhead alternative in such areas as materials simulations.

ACKNOWLEDGEMENTS

This work was supported by National Science Foundation CAREER Program (grant no. ASC-9701504), Army Research Office (grant no. DAAH04-96-1-0393), Petroleum Research Fund (grant no. 31659-AC9), and Louisiana Education Quality Support Fund (grant no. LEQSF(96-99)-RD-A-10). Numerical experiments were performed on the IBM SP2 computer at the Maui High Performance Computing Center (MHPCC) and the IBM SP3 computers at Argonne National Laboratory. Computations were also performed on parallel architectures at the Concurrent Computing Laboratory for Materials Simulations (CCLMS) at Louisiana State University. The facilities in the CCLMS were acquired with the Equipment Enhancement Grants from the LEQSF. The author would like to thank Dr. Rajiv K. Kalia and Dr. Priya Vashishta for useful discussions and Mr. Guodong Qin for his help with numerical experiments.

REFERENCES

1. P. Vashishta, R. K. Kalia, S. W. de Leeuw, D. L. Greenwell, A. Nakano, W. Jin, L. Bi and W. Li, 'Computer simulation of materials using parallel architectures', *Comput. Mater. Sci.*, **2**, 180–208 (1994).
2. A. Nakano, R. K. Kalia and P. Vashishta, 'Multiresolution molecular dynamics algorithm for realistic materials modeling on parallel computers', *Comput. Phys. Commun.*, **83**, 197–214 (1994).
3. R. K. Kalia, A. Nakano, A. Omeltchenko, K. Tsuruta and P. Vashishta, 'Role of ultrafine microstructures in dynamic fracture in nanophase silicon nitride', *Phys. Rev. Lett.*, **78**, 2144–2147 (1997).
4. G. C. Fox, R. D. Williams and P. C. Messina, *Parallel Computing Works*, Morgan Kaufmann, San Francisco, CA, 1994.
5. R. D. Williams, 'Performance of dynamic load balancing algorithms for unstructured mesh calculations', *Concurrency, Pract. Exp.*, **3**, 457–481 (1991).
6. B. Hendrickson and R. Leland, 'An improved spectral load balancing method', *Proc. Sixth SIAM Conf. Parallel Processing for Scientific Computing*, SIAM, Philadelphia, 1993, pp. 953–961.
7. S. T. Barnard and H. D. Simon, 'Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems', *Concurrency, Pract. Exp.*, **6**, 101–117 (1994).
8. A. Y. Grama, V. Kumar and A. Sameh, 'Scalable parallel formulation of the Barnes-Hut method for N-body simulations', *Supercomputing '94*, IEEE Comp. Soc., Washington, DC, 1994.
9. J. P. Singh, C. Holt, J. L. Hennessy and A. Gupta, 'A parallel adaptive fast multipole method', *Supercomputing '93*, IEEE Comp. Soc., Washington, DC, 1993, pp. 54–65.
10. M. Kaddoura, C.-W. Ou and S. Ranka, 'Partitioning unstructured computational graphs for nonuniform and adaptive environments', *IEEE Parallel Distrib. Technol.*, Fall, pp. 63–69 (1995).
11. M. S. Warren and J. K. Salmon, 'A portable parallel particle program', *Comput. Phys. Commun.*, **87**, 266–290 (1995).

12. J. R. Pilkington and S. B. Baden, 'Dynamic partitioning of non-uniform structured workloads with spacefilling curves', *IEEE Trans. Parallel Distrib. Syst.*, **7**, 288(1996).
13. G. Cybenko, 'Dynamic load balancing for distributed memory multiprocessors', *J. Parallel Distrib. Comput.*, **7**, 279–301 (1989).
14. J. E. Boillat, 'Load balancing and Poisson equation in a graph', *Concurrency Pract. Exp.*, **2**, 289–313 (1990).
15. G. Horton, 'A multi-level diffusion method for dynamic load balancing', *Parallel Comput.*, **19**, 209–218 (1993).
16. A. Heirich and S. A. Taylor, 'A parabolic load balancing method', *24th International Conference on Parallel Processing*, CRC Press, New York, NY, 1995, pp. 192–202.
17. Y. Deng, R. A. McCoy, R. B. Marr, R. F. Peiers and O. Yasar, 'Molecular dynamics for 400 million particles with short-range interactions', *Supercomputing '94*, IEEE Comp. Soc., Washington, DC, 1994.
18. A. Nakano and T. Campbell, 'An adaptive curvilinear-coordinate approach to dynamic load balancing of parallel multi-resolution molecular dynamics', *Parallel Comput.*, **23**, 1461–1478 (1997).
19. F. Gygi, 'Electronic-structure calculations in adaptive coordinates', *Phys. Rev. B*, **48**, 11692–11700 (1993).
20. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, 'Optimization by simulated annealing', *Science*, **220**, 671–680 (1983).
21. I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, PA, 1992.
22. L. Greengard and V. Rokhlin, 'A fast algorithm for particle simulations', *J. Comput. Phys.*, **73**, 325–348 (1987).
23. W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in FORTRAN*, 2nd ed., Cambridge Univ. Press, New York, NY, 1992.
24. F. Gygi, 'Ab initio molecular dynamics in adaptive coordinates', *Phys. Rev. B*, **51**, 11190–11193 (1995).
25. W. Gropp, E. Lusk and A. Skjellum, *Using MPI*, MIT Press, Cambridge, MA, 1994.
26. X.-H. Sun and J. Gustafson, 'Toward a better parallel performance metric', *Parallel Comput.*, **17**, 1093–1109 (1991).

