Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Physics Communications 153 (2003) 445–461
Computer Physics Communications
www.elsevier.com/locate/cpc
ELSEVIER

Scalable and portable implementation of the fast multipole method on parallel computers ☆

Shuji Ogata [a], Timothy J. Campbell [b], Rajiv K. Kalia [c,d], Aiichiro Nakano [c,d,*], Priya Vashishta [c,d], Satyavani Vemparala [d]

[a] *Department of Applied Sciences, Yamaguchi University, Ube 755-8611, Japan*
[b] *Mississippi State University, Stennis Space Center, MS 39529, USA*
[c] *Collaboratory for Advanced Computing and Simulations, Department of Computer Science, Department of Material Science & Engineering, Department of Physics & Astronomy, Department of Biomedical Engineering, University of Southern California, Los Angeles, CA 90089, USA*
[d] *Concurrent Computing Laboratory for Materials Simulations, Biological Computation and Visualization Center, Department of Computer Science, Department of Physics & Astronomy, Louisiana State University, Baton Rouge, LA 70801, USA*

Abstract

A scalable and portable Fortran code is developed to calculate Coulomb interaction potentials of charged particles on parallel computers, based on the fast multipole method. The code has a unique feature to calculate microscopic stress tensors due to the Coulomb interactions, which is useful in constant-pressure simulations and local stress analyses. The code is applicable to various boundary conditions, including periodic boundary conditions in two and three dimensions, corresponding to slab and bulk systems, respectively. Numerical accuracy of the code is tested through comparison of its results with those obtained by the Ewald summation method and by direct calculations. Scalability tests show the parallel efficiency of 0.98 for 512 million charged particles on 512 IBM SP3 processors. The timing results on IBM SP3 are also compared with those on IBM SP4.

PACS: 02.60.-x; 02.70.-c

Keywords: Fast multipole method; Parallel computation; Stress calculation; Periodic boundary conditions; Coulomb interaction

PROGRAM SUMMARY

Title of program: FMMP

Catalogue identifier: ADRX

Program Summary URL: http://cpc.cs.qub.ac.uk/summaries/ADRX

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland

Operating systems or monitors under which the program has been tested: LINUX with MPICH, IBM SP, SGI Origin

Programming language used: Fortran77, C language preprocessor

Has the code been vectorized or parallelized: parallelized using MPI Standard

No. of bytes in distributed program, including test data, etc.: 16 167

☆ This paper can be downloaded from the CPC Program Library under catalogue identifier: http://cpc.cs.qub.ac.uk/summaries/ADRX

* Corresponding author.
E-mail address: anakano@usc.edu (A. Nakano).

0010-4655/03/$ – see front matter © 2003 Published by Elsevier B.V.
doi:10.1016/S0010-4655(03)00246-7

Distribution format: tar gzip file

Nature of physical problem
Parallel computations of Coulomb potentials, forces, and stress tensors for a collection of charged particles.

Method of solution
The fast multipole method.

Typical running time
Proportional to the number of charged points.

Unusual features of the program: None

LONG WRITE-UP

1. Introduction

The fast multipole method (FMM) developed by Greengard and Rokhlin [1] calculates electrostatic energies and forces for a collection of N charged particles with $O(N)$ operations and with predictable error bounds, whereas the optimal Ewald method is an $O(N^{3/2})$ algorithm [2]. The FMM enables atomistic simulations of realistic materials involving millions to billions of charged particles under various settings of boundary conditions [3]. We have recently developed a scalable and portable FMM code named FMMP for materials simulations on parallel computers using the Message Passing Interface (MPI) standard [4]. The FMMP features the calculation of microscopic stress tensors, which is needed for pressure-controlled simulations and local stress analyses. It also supports various boundary conditions, including two- and three-dimensional periodic-boundary conditions, which are used for slab and bulk systems, respectively. The FMMP has been used in various materials simulations, including oxidation of aluminum nanoparticles [5] and sintering of titania nanoparticles [6] in which interatomic interaction is modeled with variable-charge potentials [7,8].

In this paper, we describe technical details that are needed for using FMMP, and present accuracy and performance tests of FMMP. The FMMP code is composed of a Fortran program file fmmp.F and an include file fmmp_dim.h. The fmmp.F should be preprocessed by the C language preprocessor (CPP) before compilation with Fortran77. Control macro-parameters for CPP are set in fmmp_dim.h. Present distribution package of FMMP includes a sample driver program fmmptest.F that uses subprograms in fmmp.F.

2. Formulation of FMM

We consider a collection of N particles with charges $\{q_i \mid i = 1, \ldots, N\}$ at positions $\{\vec{a}_i \mid i = 1, \ldots, N\}$; $\vec{a}_i = (a_i, \theta_i, \varphi_i)$ with $a_i = |\vec{a}_i|$ in polar coordinates. Greengard and Rokhlin [1] proposed an $O(N)$ algorithm to compute Coulomb potentials for all particles, i.e. $\sum_{j \neq i} q_j / |\vec{a}_i - \vec{a}_j|$ $(i = 1, \ldots, N)$. The direct calculations to evaluate Coulomb potentials at N particle positions requires $O(N^2)$ operations. In this section, we summarize equations that are necessary to understand the present implementation of the FMM.

The electrostatic potential at position $\vec{r}$ may be expressed as [9]

$$\sum_i \frac{q_i}{|\vec{r} - \vec{a}_i|} = \begin{cases} \sum_{l=0}^{\infty} \sum_{m=-l}^{l} M_{lm}(\vec{A}) L_{lm}(\vec{r}) & \text{for } r > A_{\max}, \\ \sum_{l=0}^{\infty} \sum_{m=-l}^{l} L_{lm}(\vec{A}) M_{lm}(\vec{r}) & \text{for } r < A_{\min}, \end{cases} \tag{1}$$

where

$$M_{lm}(\vec{x}) \equiv \frac{1}{(l + |m|)!} x^l P_{lm}(\cos\theta) \exp(-\mathrm{i}m\varphi), \tag{2}$$

$$L_{lm}(\vec{x}) \equiv (l-|m|)!x^{-(l+1)}P_{lm}(\cos\theta)\exp(\mathrm{i}m\varphi), \tag{3}$$

$$M_{lm}(\vec{A}) \equiv \sum_i q_i M_{lm}(\vec{a}_i), \tag{4}$$

$$L_{lm}(\vec{A}) \equiv \sum_i q_i L_{lm}(\vec{a}_i). \tag{5}$$

Here, $\vec{x} = (x,\theta,\varphi)$ in polar coordinates, $\vec{A}$ denotes a collective set, $\{\vec{a}\}$, of the positions, $A_{\max(\min)}$ is the maximum (minimum) value of a_i, $\{M_{lm}(\vec{A})\}$ are the multipole moments [9] of the charges, and $\{L_{lm}(\vec{A})\}$ are the local Taylor expansion coefficients [9] of the Coulomb field at the origin. $P_{lm}(x)$ in Eqs. (2) and (3) are defined with the associated Legendre polynomials

$$P_l^{\mu}(x) \equiv (1-x^2)^{\mu/2}\frac{\mathrm{d}^{\mu}}{\mathrm{d}x^{\mu}}P_l(x) \quad (l \geqslant \mu \geqslant 0)$$

as

$$P_{lm}(x) = \begin{cases} (-1)^m P_l^m(x) & \text{for } m \geqslant 0, \\ P_l^{|m|}(x) & \text{for } m < 0. \end{cases} \tag{6}$$

The following recursion formulas are useful to calculate $P_{lm}(x)$

$$\begin{aligned} &P_{mm}(x) = (-1)^m\big[(2m-1)(2m-3)\cdots 1\big](1-x^2)^{m/2} && \text{for } m \geqslant 1, \\ &\quad P_{00}(x) = 1, \\ &P_{m+1,m}(x) = x(2m+1)P_{mm}(x) && \text{for } m \geqslant 0, \\ &(l-m)P_{lm}(x) = x(2l-1)P_{l-1,m}(x) - (l+m-1)P_{l-2,m}(x) && \text{for } m \geqslant 0, \\ &P_{l,-m}(x) = (-1)^m P_{lm}(x) && \text{for } m \geqslant 1. \end{aligned} \tag{7}$$

We may use the following relations to speed up computations if values of charges are real numbers:

$$\begin{aligned} &M_{l,-m} = (-1)^m M_{lm}^* && \text{for } m > 0, \\ &L_{l,-m} = (-1)^m L_{lm}^* && \text{for } m > 0, \end{aligned} \tag{8}$$

where $*$ denotes the complex conjugate.

The above recursive relations can be used to derive transformation operators between M_{lm} and L_{lm}:

$$\sum_l\sum_m M_{lm}(\vec{A})L_{lm}(\vec{r}) = \sum_l\sum_m M_{lm}(\vec{A}-\vec{b})L_{lm}(\vec{r}-\vec{b}) \tag{9}$$

with the condition of $|\vec{r}-\vec{b}| > |\vec{A}-\vec{b}|$,

$$\sum_l\sum_m M_{lm}(\vec{A}-\vec{b})L_{lm}(\vec{r}-\vec{b}) = \sum_l\sum_m L_{lm}(\vec{b}'-\vec{A})M_{lm}(\vec{b}'-\vec{r}) \tag{10}$$

with the condition of $|\vec{b}'-\vec{A}| > |\vec{b}'-\vec{r}|$, and

$$\sum_l\sum_m L_{lm}(\vec{b}'-\vec{A})M_{lm}(\vec{b}'-\vec{r}) = \sum_l\sum_m L_{lm}(\vec{c}-\vec{A})M_{lm}(\vec{c}-\vec{r}) \tag{11}$$

with the condition of $|\vec{c}-\vec{A}| > |\vec{c}-\vec{r}|$. In Eqs. (9)–(11), $M_{lm}(\vec{A}-\vec{b})$ and $L_{lm}(\vec{b}'-\vec{A})$ are short hand notations for $\sum_i q_i M_{lm}(\vec{a}_i-\vec{b})$ and $\sum_i q_i L_{lm}(\vec{b}'-\vec{a}_i)$, respectively.

We then find the following three types of transformation operations [9]:

$$M_{lm}(\vec{A}-\vec{b}) = \sum_{j=0}^{l}\sum_{k=-j}^{j} T^{\mathrm{MM}}_{l-j,m-k}(\vec{b})M_{jk}(\vec{A}), \qquad \text{[multipole-to-multipole]} \tag{12}$$

$$L_{lm}(\vec{b}-\vec{A}) \sim \sum_{j=0}^{p}\sum_{k=-j}^{j} T^{\mathrm{ML}}_{j+l,k+m}(\vec{b})M_{jk}(\vec{A}) \quad (p=\text{maximum of } l), \qquad \text{[multipole-to-local]} \tag{13}$$

$$L_{lm}(\vec{c}-\vec{A}) = \sum_{j=l}^{p}\sum_{k=-j}^{j} T^{\mathrm{LL}}_{j-l,k-m}(\vec{b}-\vec{c})L_{jk}(\vec{b}-\vec{A}) \quad (p=\text{maximum of } l), \qquad \text{[local-to-local]} \tag{14}$$

with the operators,

$$T^{\mathrm{MM}}_{l-j,m-k}(\vec{b}) = M_{l-j,m-k}(-\vec{b}), \tag{15}$$

$$T^{\mathrm{ML}}_{j+l,k+m}(\vec{b}) = L_{j+l,k+m}(\vec{b}), \tag{16}$$

$$T^{\mathrm{LL}}_{j-l,k-m}(\vec{b}) = M_{j-l,k-m}(\vec{b}). \tag{17}$$

The FMM calculates Coulomb potentials of charged particles contained in a simulation box in five steps [1,2,9]:

(i) The simulation box with dimensions (h_x, h_y, h_z) is successively subdivided. Let l^x_{bot}, l^y_{bot}, and l^z_{bot} be the prescribed numbers of recursive subdivisions along the x-, y-, and z-axes. At the finest subdivision level $l_{\mathrm{bot}} = \max(l^x_{\mathrm{bot}}, l^y_{\mathrm{bot}}, l^z_{\mathrm{bot}})$, the simulation box is decomposed into cells with dimensions $(h_x/2^{l^x_{\mathrm{bot}}}, h_y/2^{l^y_{\mathrm{bot}}}, h_z/2^{l^z_{\mathrm{bot}}})$. The largest division is the simulation box itself at level 0. At level l, the simulation box is composed of cells with dimension $(h_x/\max(2^{l^x_{\mathrm{bot}}+l-l_{\mathrm{bot}}}, 1), h_y/\max(2^{l^y_{\mathrm{bot}}+l-l_{\mathrm{bot}}}, 1), h_z/\max(2^{l^z_{\mathrm{bot}}+l-l_{\mathrm{bot}}}, 1))$.

(ii) Compute multipole moments of all the cells at the finest level of subdivision. Sweep up from the smallest cells to largest cell to obtain multipole moments of cells at all subdivision levels using the multipole-to-multipole transformation formula, Eq. (12).

(iii) Sweep down from the largest cell to cells at the next level of subdivision to obtain local expansion coefficients in the smallest cells: First, transform local expansion coefficients of larger cell to cells at the next level of subdivision using the local-to-local transformation formula for shifting the origin of a local expansion, Eq. (14). Second, add to these local expansion coefficients contribution from cells at the next level of subdivision, which have not been included and are well-separated from the cell being considered, using the multipole-to-local transformation formula, Eq. (13).

(iv) Once the preceding step has reached the finest subdivision level, evaluate the potential for each particle, Eq. (1), using the local expansion coefficients, $L_{lm}(\vec{A})$, of the smallest cell containing the particle.

(v) Add contributions from other charges in the same cell and the near neighbor cells by direct computations.

The FMMP performs steps (i)–(iv). The direct computations in step (v) should be performed separately in a subroutine supplied by the user. Such a separation in the FMM steps is appropriate since usual MD simulation codes have a linked list [10] of neighboring particles for fast computations of short-range forces and the linked list can be exploited for the direct calculations in step (v).

In FMMP, the most time-consuming part is the multipole-to-local transformation operation in step (iii). If we take terms only up to $|j+l| \leqslant p$ in Eq. (13), the computation time becomes approximately half with some loss of accuracy [9].

Forces acting on particles are calculated by differentiating the potential, using the following formulas:

$$\frac{\partial M_{lm}}{\partial x} = M_{lm}\left[\frac{lx}{x^2+y^2} + \mathrm{i}m\frac{y}{x^2+y^2}\right] - M_{l-1,m}\frac{zx}{x^2+y^2}, \tag{18}$$

$$\frac{\partial M_{lm}}{\partial y} = M_{lm}\left[\frac{ly}{x^2+y^2} - \mathrm{i}m\frac{x}{x^2+y^2}\right] - M_{l-1,m}\frac{yz}{x^2+y^2}, \tag{19}$$

$$\frac{\partial M_{lm}}{\partial z} = M_{l-1,m}. \tag{20}$$

3. Calculation of the microscopic stress-tensor

In MD simulations of realistic materials, microscopic stress tensors are often required to determine MD box deformation and to investigate stress distribution in the material [3,10,11]. However no general O(N) method has been found in the existing literature to calculate stress tensors due to Coulomb interactions. We have developed the complex-charge method (CCM) to calculate Coulomb contribution to the stress tensor field.

The Coulomb contribution to the microscopic stress tensor field at $\vec{r}$ is defined as [10,11]

$$\overleftrightarrow{\pi}(\vec{r}) = \sum_i q_i \frac{(\vec{r}-\vec{a}_i)(\vec{r}-\vec{a}_i)^{\mathrm{T}}}{|\vec{r}-\vec{a}_i|^3}, \tag{21}$$

where $\leftrightarrow$ denotes a matrix and T denotes the transposition of a vector. Apparently, Eq. (21) is a summation of a product [force due to particle i] $\times$ [distance from particle i]. In the naive usage of the FMM, one can obtain summation of forces acting at $\vec{r}$ but not the summation of the product.

The idea in CCM is to attach information of the particle position to the charge. We first calculate

$$B(\vec{r},\vec{k}) = \sum_i q_i \frac{\exp(\mathrm{i}\vec{k}\cdot\vec{a}_i)}{|\vec{r}-\vec{a}_i|} \tag{22}$$

with $\vec{k} = (\Delta k, 0, 0), (0, \Delta k, 0), (0, 0, \Delta k)$, and $(0, 0, 0)$ and $\Delta k \ll 1/L$ (L is the characteristic length of the simulation box) using the FMM. Here the particle charge may be regarded as a complex number, $q_i \exp(\mathrm{i}\vec{k}\cdot\vec{a}_i)$. We then calculate a complex force field $\vec{C}(\vec{r},\vec{k}) = \frac{\mathrm{d}}{\mathrm{d}\vec{r}}B(\vec{r},\vec{k})$ with the FMM. The $\alpha\beta$-component of $\overleftrightarrow{\pi}(\vec{x})$ is obtained by differentiating $C_\alpha(\vec{x},\vec{k})\exp(-\mathrm{i}\vec{k}\cdot\vec{x})$ with respect to k_β at $k=0$ and by taking its imaginary part:

$$\begin{aligned}\pi_{\alpha\beta}(\vec{x}) &= \lim_{k\to 0}\mathrm{Im}\left[\frac{\partial}{\partial k_\beta}\left\{C_\alpha(\vec{x},\vec{k})\exp(-\mathrm{i}\vec{k}\cdot\vec{x})\right\}\right] \\ &= \lim_{k\to 0}\sum_i q_i \frac{(x_\alpha - a_{i\alpha})(x_\beta - a_{i\beta})}{|\vec{x}-\vec{a}_i|^3}\cos\left(\mathrm{i}\vec{k}\cdot(\vec{a}_i-\vec{x})\right) \\ &= \sum_i q_i \frac{(x_\alpha - a_{i\alpha})(x_\beta - a_{i\beta})}{|\vec{x}-\vec{a}_i|^3}.\end{aligned} \tag{23}$$

The differentiation with respect to $\vec{k}$ in Eq. (23) is performed through finite difference using the FMM values of C_α at different values of $\vec{k}$. The CCM is simple to implement, since no multipole translation operator for the stress tensor needs to be derived. In FMMP, Δk is set to $10^{-3}L$.

4. Boundary conditions

The FMMP is applicable to systems of both isolated clusters of charges and periodically repeating simulation boxes. For the periodic boundary conditions, we previously implemented the reduced-cell multipole method (RCMM) [12]. In the RCMM, a well-separated image of a simulation box is represented by a small number of charged points; those positions are chosen at random and magnitudes of the charges are determined to reproduce

lower-order multipoles of the original simulation box [13]. The Ewald summation technique is used to calculate Coulomb potential energy between such periodically repeating, reduced-cell charges. However, we found that the potential energy calculated with the RCMM is sensitive to the positions of representative reduced-cell particles. In the FMMP, we use a different method that resembles the macroscopic-multipole method [13,14].

It is known that Coulomb potential energy for increasing numbers of repeating simulation boxes becomes a convergent series if aggregates of the boxes form, e.g., a spherical shape [10,13–16]. The Coulomb potential energy of an aggregate of infinite boxes includes a term arising from the finite dipole moment of the box. This dipole term creates a discontinuity in the potential at the box boundaries, i.e. two translationally identical positions on opposite sides of the box may have different potentials. In the Ewald summation method, this macroscopic dipole term is neglected, and physically this corresponds to embedding the aggregate of boxes in a "metallic" environment instead of vacuum [10,13–16].

When we choose a periodic boundary condition in the FMMP, the code calculates the Coulomb interactions with the original simulation box wrapped by a lattice of boxes forming a sphere with radius ∼7 boxes. Since the original MD box is represented by a finite number of multipoles, the computation time increases by small amounts by taking such image boxes in the FMMP. To realize the metallic environment as in the case of the Ewald summation, the user has to add dipole correction terms to the FMMP values. The potential energy in the three-dimensional Ewald summation method is then reproduced as

$$V_{\mathrm{Ewald}}^{\mathrm{3D}} = V_{\mathrm{FMM}} + \Delta V_{\mathrm{dipole}}^{\mathrm{3D}}, \tag{24}$$

where

$$\Delta V_{\mathrm{dipole}}^{\mathrm{3D}} = -\frac{2\pi}{3\Omega}|\vec{P}|^2 \tag{25}$$

with the volume of the simulation box Ω and the dipole of the simulation box $\vec{P} = \sum_{i=1}^{N} q_i \vec{a}_i$. Corresponding correction term to the force on particle i is $-\partial \Delta V_{\mathrm{dipole}}^{\mathrm{3D}}/\partial \vec{a}_i$. The correction term to the total stress tensor is

$$\overleftrightarrow{\Pi}_{\mathrm{Ewald}}^{\mathrm{3D}} = \overleftrightarrow{\Pi}_{\mathrm{FMM}} + \frac{4\pi}{3\Omega}\vec{P}\vec{P}^{\mathrm{T}} + \Delta V_{\mathrm{dipole}}^{\mathrm{3D}}\overleftrightarrow{1}. \tag{26}$$

Recently there have been increasing interests in simulating surface and interfacial systems with a slab geometry. Conventional 3D Ewald summation formula cannot be used directly because of no periodicity in one of the three directions. A two-dimensional Ewald summation technique has been developed, giving explicit formulas for the Coulomb energy $V_{\mathrm{Ewald}}^{\mathrm{2D}}$ for the slab geometry [17–19]. Yeh and Berkowitz [20] performed a detailed numerical comparison between $V_{\mathrm{Ewald}}^{\mathrm{3D}}$ and $V_{\mathrm{Ewald}}^{\mathrm{2D}}$ and found an efficient way of calculating $V_{\mathrm{Ewald}}^{\mathrm{2D}}$ using $V_{\mathrm{Ewald}}^{\mathrm{3D}}$.

Let us assume that the (neutral) slab system is periodic in x- and y-directions, and that an empty space is inserted with its length in z-direction greater than or equal to $\max(L_x, L_y)$. The 2D Ewald result for the Coulomb energy of the system is reproduced as [20]

$$V_{\mathrm{Ewald}}^{\mathrm{2D}} = V_{\mathrm{Ewald}}^{\mathrm{3D}} + \frac{2\pi}{V}P_z^2 = V_{\mathrm{FMM}} + \Delta V_{\mathrm{dipole}}^{\mathrm{3D}} + \frac{2\pi}{V}P_z^2 \tag{27}$$

using the FMM value V_{FMM} in the 3D periodic-boundary condition.

5. Parallel computation

In the FMMP, the simulation box is spatially decomposed into `nx` × `ny` × `nz` subsystems, which are assigned to the same number of compute nodes. Here, `nx`, `ny`, and `nz` should be either 1 or a power of 2, i.e. $1, 2, 4, 8, \ldots$, with constraints $\log_2(\mathtt{nx}) \leqslant l_{\mathrm{bot}}^x$, $\log_2(\mathtt{ny}) \leqslant l_{\mathrm{bot}}^y$, and $\log_2(\mathtt{nz}) \leqslant l_{\mathrm{bot}}^z$. Each node has a scalar index `myid` in the range $[0, \mathtt{nx} \times \mathtt{ny} \times \mathtt{nz} - 1]$. The vector node-index (`myx`, `myy`, `myz`) satisfies the relation $\mathtt{myid} = \mathtt{myx} \times \mathtt{ny} \times \mathtt{nz} + \mathtt{myy} \times \mathtt{nz} + \mathtt{myz}$, and it specifies the (x, y, z) position of the node in

the logical 3D array of compute nodes. In the single-node case (nx = ny = nz = 1), for example, myid = 0 with (myx, myy, myz) = (0, 0, 0); in the two-node case (nx = 2, ny = nz = 1), two nodes are myid = 0 with (myx, myy, myz) = (0, 0, 0) and myid = 1 with (myx, myy, myz) = (1, 0, 0). Each node is given a number of particles Ntotn in the corresponding subsystem, charges {chg(i); i = 1, ..., Ntotn}, normalized positions {sr(1 − 3, i); i = 1, ..., Ntotn}, the simulation-box tensor $\overleftrightarrow{h}$ = (hx(1 − 3), hy(1 − 3), hz(1 − 3)). The normalized position of particle i, sr(1 − 3, i), in a node with the vector node-index (myx, myy, myz) satisfies the following inequalities: myx/nx < sr(1, i) < (myx + 1)/nx, myy/ny < sr(2, i) < (myy + 1)/ny, and myz/nz < sr(3, i) < (myz + 1)/nz.

Parameters that control the accuracy of the FMMP are the maximum subdivision levels (lbotx, lboty, lbotz), the maximum order of multipoles iptop, and the minimum separation between well-separated cells normalized by the simulation box iWS = 1 or 2 (see step (iii) in the FMM algorithm). Multipole and local expansion data at subdivision level $l \leqslant$ lglim are global and stored in all nodes, whereas the data at level $l >$ lglim are stored only in the corresponding node and transferred to different nodes through MPI_Send and MPI_Receive calls when they are required. The value of lglim is lfit (for iWS = 1) or lfit + 1 (for iWS = 2) with lfit = $\log_2$[max(nx, ny, nz)]. The lfit corresponds to the level at which the cell assumes the maximum size in a single node.

6. Selected subroutines in FMMP

The following summarizes important subroutines in the FMMP.

FMPmain: All the FMM calculations are performed in this subroutine.
Mpsetup: This subroutine sets up cell indices for upward and downward passes in the FMM.
MPup: This subroutine calculates the multipole moments for cells starting from the smallest cell toward the simulation box (level 0).
MPdown: This subroutine transforms the local expansion coefficients of a larger cell to cells at the next level of subdivision using the local-to-local transformation formula for shifting the origin of a local expansion. Then, the subroutine adds to these local expansion coefficients the contribution from cells at the next level of subdivision, which have not been included and are well-separated from the cell being considered, using the multipole-to-local transformation formula.
GetPFS: This subroutine calculates potential, force, stress tensor fields felt by particles using the local expansion coefficients for the smallest cell containing the particle.
MDwrap3: This subroutine calculates the contribution of periodic images of simulation boxes in 3 dimensions.

7. Parameters, input, and output of the FMMP

The user should set the following CPP macros in the include file, fmmp_dim.h, to determine maximum array sizes used in FMMP:

DEBUG:	Switch to write (= 1) debugging information.
INCLUDE_STRESS:	Switch to include (= 1) arrays for stress calculation.
PFLM_PRECISION:	Switch for single (1) or double (2) precision representation of multipoles.
Nsize_:	maximum number of particles in each node.
iptop_:	maximum order of multipoles.
msize_:	maximum size for the cell index. msize > $1 + 8 + 8^2 + \cdots + 8^{\text{lbot}}$ with lbot = max(lbotx, lboty, lbotz).
ibsize_:	maximum buffer size for data transfer between the nodes.

We note that including stress calculations significantly increases required memory size of the program. The fmmp.F should be preprocessed by CPP before compilation with Fortran77.

The FMM calculations are performed in subroutine FMPmain. The user has to set the number of compute nodes in each direction (`nx`, `ny`, `nz`), the scalar node-index `myid`, and the vector node-index (`myx`, `myy`, `myz`) in a common block `node_vec`. Dimensions of the simulation box $\overleftrightarrow{h} = (\texttt{hx}(1-3), \texttt{hy}(1-3), \texttt{hz}(1-3))$ should be set in a common block `MDbox`. Charge and normalized coordinates of particles (`chg(i)`, `sr`$(1-3, \texttt{i})$) and the total number of particles in each node `Ntotn` are set in a common block `node_ptcl`.

The following control parameters for FMMP are set in the common block `mpdat1`.

`lbotx(y, z)`: the finest level of subdivision in $x(y, z)$ direction; $2 \leqslant$ `lbotx(y, z)` $\leqslant 16$.
`iWS` $= 1$ or 2: the minimum separation distance in units of the simulation box between well-separated cells.
`iTR` $= 0$ or 1: If `iTR` $= 1$, the multipole-to-local translation is truncated.
`iPBC` $= 0$ or 1: If one sets `iPBC` $= 0$, free boundary condition is assumed. We note that the user is allowed to set different numbers for subdivision levels (`lbotx`, `lboty`, `lbotz`) only if `iPBC` $= 0$. If one sets `iPBC` $= 1$, the simulation box is wrapped by a 3-dimensional lattice of boxes.
`iST` $= 0$ or 1: If the CPP macros `INCLUDE_STRESS` $= 1$ and `iST` $= 1$, microscopic stress-tensors are also calculated.

After the subroutine FMPmain terminates, the potential field (`PF(i)`), negative of the force field (`FF`$(1-3, \texttt{i})$), and stress tensor field (`ST`$(1-6, \texttt{i})$) felt by each particle i are stored in the common block `node_result`. Definitions of PF, FF, and SF are

$$\texttt{PF(i)} = \sum_{j \neq i} \frac{\texttt{chg(j)}}{|\vec{a}_i - \vec{a}_j|}, \tag{28}$$

$$\texttt{FF}(1-3, \texttt{i}) = \frac{\partial \texttt{PF(i)}}{\partial \vec{a}_i} \tag{29}$$

with the first index denoting $x =$ "1", $y =$ "2", $z =$ "3" components, and

$$\texttt{SF}(1-6, \texttt{i}) = \sum_{j \neq i} \texttt{chg(j)} \frac{(\vec{a}_i - \vec{a}_j)(\vec{a}_i - \vec{a}_j)^{\mathrm{T}}}{|\vec{a}_i - \vec{a}_j|^3}, \tag{30}$$

with the first index denoting $xx =$ "1", $yy =$ "2", $zz =$ "3", $yz =$ "4", $xz =$ "5", $xy =$ "6" components. The measured computation time for each FMM step explained in Section 2 is stored in the common block `fmm_time`:

`t_setup`: setup time,
`t_comm`: total communication time,
`t_up`: time for upward pass,
`t_down`: time for downward pass,
`t_wrap`: time to take care PBC,
`t_pfs`: time to compute PF, FF, and ST.

Common block names used in the FMMP are `node_vec`, `MDbox`, `node_ptcl`, `node_result`, `mpdat1`, `mpdat2`, `pseudoC`, `PBC3`, `fmm_time`, and `fmmparity`. Those names should not be used in the main program.

8. Sample program: accuracy and scalability

We have prepared a sample driver program, fmmptest.F, to illustrate the usage of the FMMP and test its accuracy. The main module and the include file are listed in Appendices A and B, respectively. The executable fmmptest is created with the make command. The present package contains a sample makefile for Linux and Compaq workstations with MPICH installed. The fmmptest is then run using n nodes under MPI environment [4] as `mpirun -np n fmmptest`.

We perform two tests. The first test estimates the accuracy of the FMM by comparing its results with those of Ewald summations and direct calculations. In the second test, we demonstrate the scalability of the code by executing it on different numbers of nodes and study the effect of system size on the scalability of the code.

In the first part of the test, $N = 1000$ particles (charge, $+1$ or -1) are scattered randomly in a cubic box (side length, 1) such that the overall charge neutrality is maintained. The simulation box is decomposed into $P = 8$ nodes; $(\mathtt{nx}, \mathtt{ny}, \mathtt{nz}) = (2, 2, 2)$. The values of some of the parameters are: $\mathtt{iWS} = 1$, $\mathtt{iTR} = 0$, $\mathtt{lbotx} = \mathtt{lboty} = \mathtt{lbotz} = 3$, $\mathtt{iST} = 0$, and $\mathtt{iptop} = 5$. The results of the accuracy test are shown in Appendix C. This test was performed for both $\mathtt{iPBC} = 0$ and $\mathtt{iPBC} = 1$ and, for both cases, the program compares the FMM results of the potential fields felt by the particles (that include nearest neighbor contribution) with the direct calculation results. Averaged values of the relative errors are written to the standard output with timing data. Outputs of FMMP produced with PGI Fortran77 and performed on a Linux PC cluster (8 nodes, 100BaseTX-connected PentiumIII/600 MHz) are listed in Appendix C. In the present case ($\mathtt{iptop} = 5$), the averaged relative errors of the potential fields are on the order of 0.01%. When more accurate results are required, the user should set a large value for `iptop` and/or set $\mathtt{iWS} = 2$. Averaged relative errors of the force (stress) field are 2–3 (5–10) times larger than that of the potential field.

The second set of the tests examines the scalability of the FMMP on massively parallel computers. We chose the following parameters: $\mathtt{iptop} = 5$, $\mathtt{iWS} = 1$, $\mathtt{iTR} = 0$, $\mathtt{iST} = 0$, and $\mathtt{iPBC} = 1$. We performed the calculations on an IBM SP3, called "HABU", at the U.S. Naval Oceanographic Office (NAVO) Major Shared Resource Center. The HABU is configured with 375 MHz Power3 CPUs and has 334 nodes with 4 CPUs and 4 GB of memory per node, total of 1336 processors. It runs AIX 4.3 operating system and IBM XL Fortran 7.1 compiler. We also repeated the same calculations on another parallel computer, IBM SP4, called "Marcellus" at NAVO. The Marcellus is configured with 1.3 GHz Power4 CPUs and has 148 nodes with 8 CPUs and 8 GB of memory per node, total of 1184 processors.

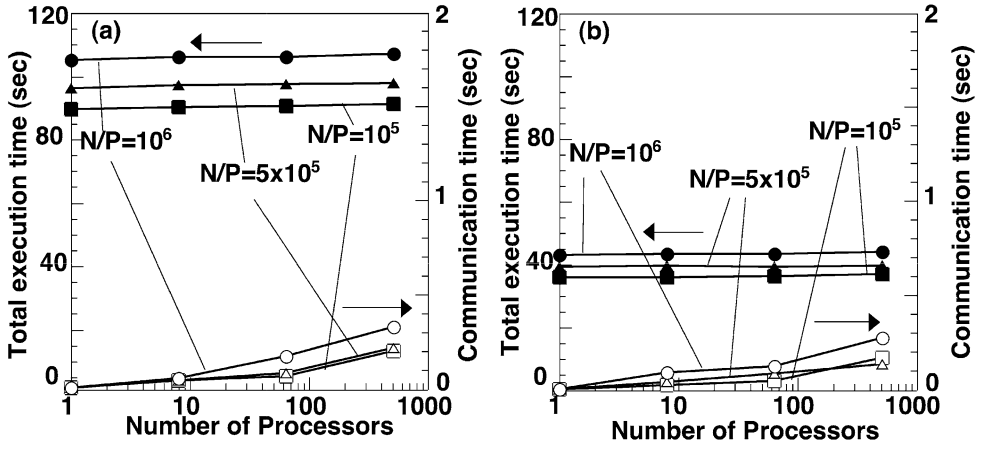

Fig. 1. Timings as function of number of processors. (a) Total execution (solid symbols) and communication (open symbols) times are plotted for the number of particles per processor, $N/P = 10^5$ (squares), 5×10^5 (triangles), and 10^6 (circles) on IBM SP3. (b) The same as (a) on IBM SP4.

The compute nodes for this test were chosen as $(\mathtt{nx}, \mathtt{ny}, \mathtt{nz}) = (1,1,1), (2,2,2), (4,4,4)$, and $(8,8,8)$ with the number of particles in each node, N/P, fixed. We chose $N/P = 100{,}000$, 500,000, and 1,000,000. We also varied the values of `lbotx` = `lboty` = `lbotz` from 4 to 7. The data and the results are shown in Appendix C and in Figs. 1–3.

Fig. 1(a) shows the results on the IBM SP3. We see that the total execution time of the FMMP code with a fixed value of N/P is nearly constant, when the total number of particles, N, is increased by varying the number of compute nodes, P. The parallel efficiency for the case of $P = 512$ and $N/P = 1{,}000{,}000$ is as high as 0.98. The communication time is only a small fraction of the total execution time. Fig. 1(b) shows the results on the IBM SP4, which is almost 2.5 times faster than the IBM SP3 in terms of computation, and a slight improvement in terms of communication time.

We also observe in Fig. 1 that the total execution time is not proportional to N/P for a fixed value of P, but is of the form $a + b(N/P)$, where $a \gg b$. This is understood by analyzing the individual timing for each subroutine in the FMMP code. Fig. 2 shows the timing data for subroutines for three different cases of $N/P = 100{,}000, 500{,}000$, and 1,000,000 with $P = 512$. We note that the dominant computation time is for the subroutine `MPdown`, which is constant and does not change with N/P. The next two dominant computations are from the subroutines `GetPFS` and `MPsetup`, which increase with N/P. This is because the number of particles contained in a cell increases in proportion to N/P. The other subroutines `MPup` and `MDwrap3` give very small contributions to the total time. Combination of the increasing times for `GetPFS` and `MPsetup` and the constant time for `MPdown` explains the observed behavior in the total execution time as a function of N/P.

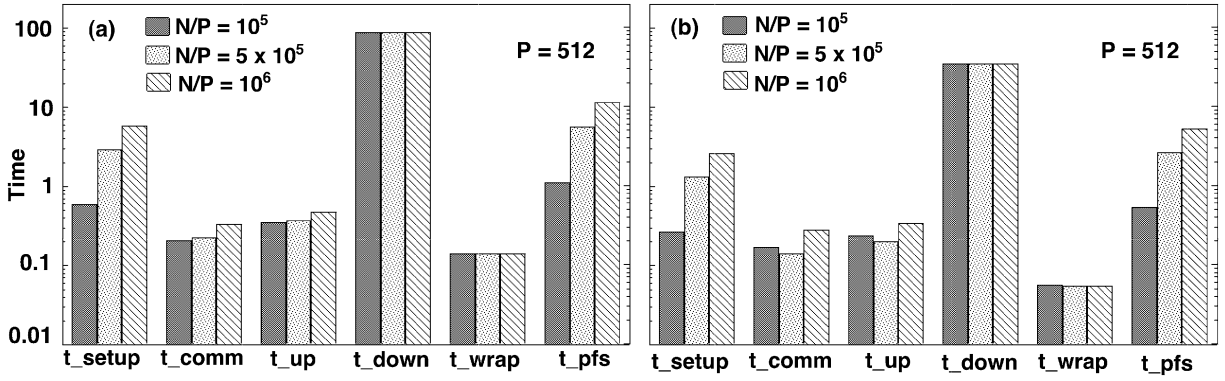

Fig. 2. Partial timing data for different subroutines for $P = 512$ for $N/P = 10^5, 5 \times 10^5$, and 10^6 on (a) SP3 and (b) SP4.

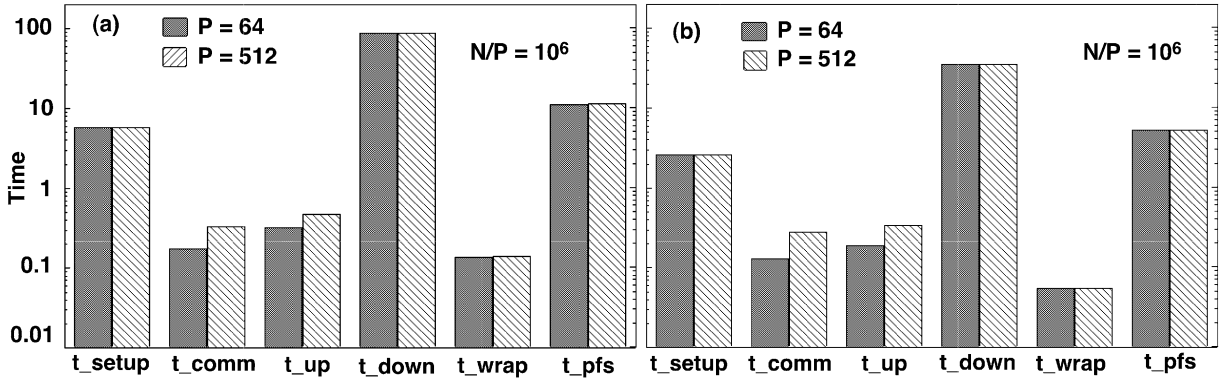

Fig. 3. Partial timing data for different subroutines for $N/P = 10^6$ on $P = 64$ and 512 processors on (a) SP3 and (b) SP4.

Fig. 3 compares the timing data for various subroutines with $N/P = 1{,}000{,}000$ between $P = 64$ and $P = 512$. We see that the timings for all the dominant subroutines, `MPdown`, `GetPFS`, and `MPsetup` are nearly the same between the two cases of P. Only the communication time and the timing for `MPup` increase slightly as a function of P.

9. Concluding remarks

The present code, FMMP, is scalable and portable implementation of the FMM. It can be used not only in materials simulations but also in various fields of simulations including plasmas and astronomical objects interacting through $1/r$ potential. The user, however, should be cautious in its application to highly ordered systems such as a crystalline lattice of charged points. Depending on the lattice structures, values of the potential energy may vary considerably as a function of `iptop` in the case of periodic boundary conditions. This results from the fact that many of the lower-order multipoles are zero or nearly zero and only some higher-order multipoles assume non-zero values in such regular lattices. The user needs to set a rather higher order of multipoles for such systems.

Accuracy and speed of the FMMP are controlled by the parameters (`lbotx`, `lboty`, `lbotz`, `iptop`, `iWS`, `iTR`). Formulas for the error bound in the potential have been derived in Refs. [1,9]. However, actual error varies depend significantly on configurations, and it is not recommended to predetermine the values of the control parameters based on these formulas. Rather, the control parameters need to be determined through test runs.

Acknowledgement

This work at Yamaguchi was partially supported by YU-VBL overseas scholarship and by ACT-JST. The work at USC and LSU was partially supported by ARL, NSF, DOE, NASA, and USC-Berkeley-Princeton DURINT. Benchmark tests were performed at DoD Major Shared Resource Centers under CHSSI and Challenge projects.

Appendix A. Listing of the sample driver program fmmptest.F

```
c
c   Sample program to use FMMP Ver. 2.1 : check accuracy and timings.
c
     program main
     implicit real*8(a-h,o-z)
#include "mpif.h"
c-----Maximum number of particles.
      parameter(Ntotmax=1000)
c-----Charges and normalized positions of all particles.
      real*8 chgg(Ntotmax),srg(3,Ntotmax)

#include "fmmp_dim.h"
      parameter(Nsize=Nsize_)
c-----Common blocks for FMMP
      common/node_vec/myid,myx,myy,myz,nx,ny,nz
      common/MDbox/hx(3),hy(3),hz(3)
      common/node_ptcl/chg(Nsize),sr(3,Nsize),Ntotn
      common/mpdat1/iws,iTR,iPBC,iST,
     & lbotx,lboty,lbotz,lbotdx,lbotdy,lbotdz,lbot,
     & lfit,lglim,ncup(0:16),memswt(0:16)
#if INCLUDE_STRESS
```

```
      common/node_result/PF(Nsize),FF(3,Nsize),
     &                   ST(6,Nsize)
#else
      common/node_result/PF(Nsize),FF(3,Nsize)
#endif
      common/fmm_time/t_setup,t_comm,t_up,t_down,
     &                t_wrap,t_pfs

c-----index to remember global particle-ID's
      integer*4 idg(Nsize)

      call MPI_INIT(ierr)
      call MPI_COMM_RANK(MPI_COMM_WORLD,myid,ierr)

c-----Computer-node layout
      nx=2
      ny=2
      nz=2
      nodes=nx*ny*nz
      myx=myid/(ny*nz)
      myy=mod(myid/nz,ny)
      myz=mod(myid,nz)

c-----Setup control parameters for FMMP
      iWS=1
      iTR=0
      lbotx=3
      lboty=lbotx
      lbotz=lbotx
      iST=0
     if(myid.eq.0)then
        write(*,*)'iPBC=(0 or 1)?'
        read(*,*)iPBC
      endif
      call MPI_BCAST(iPBC,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)

c-----Setup simulation Box
      hx(1)=1d0
      hx(2)=0d0
      hx(3)=0d0
      hy(1)=0d0
      hy(2)=1d0
      hy(3)=0d0
      hz(1)=0d0
      hz(2)=0d0
      hz(3)=1d0

c-----Spatial region of this node.
      xmin_my=dble(myx)/nx
      xmax_my=dble(myx+1)/nx
      ymin_my=dble(myy)/ny
      ymax_my=dble(myy+1)/ny
```

```
      zmin_my=dble(myz)/nz
      zmax_my=dble(myz+1)/nz

      dseed=12345
      inc=0
c-----Total number of particles. Even number.
      Ntot=1000
c-----Setup charged points at random positions
      do i=1,Ntot
        call myrnd(rndx,dseed)
        call myrnd(rndy,dseed)
        call myrnd(rndz,dseed)
        chgg(i)=mod(i,2)*2-1
        srg(1,i)=rndx
        srg(2,i)=rndy
        srg(3,i)=rndz
c-------Each node stores data of particles in the corresponding region.
        if(rndx.ge.xmin_my .and. rndx.lt.xmax_my .and.
     &     rndy.ge.ymin_my .and. rndy.lt.ymax_my .and.
     &     rndz.ge.zmin_my .and. rndz.lt.zmax_my)then
          inc=inc+1
          idg(inc)=i
          chg(inc)=chgg(i)
          sr(1,inc)=srg(1,i)
          sr(2,inc)=srg(2,i)
          sr(3,inc)=srg(3,i)
       endif
     enddo
c-----Total number of particles in this node
      Ntotn=inc

c-----FMM calculations
      call FMPmain()

      error=0d0
c-----Free boundary conditions
c-----Compare potential field with that obtained by direct calculations.
      if(iPBC.eq.0)then
        do node_id=0,nodes-1
c---------Each node writes the results
          if(myid.eq.node_id)then
            do i=1,Ntotn
c-------------Get global particle-ID
              ig=idg(i)
c-------------Contribution from particles in near neighbor cells
              call NNcont(ig,PFNN,chgg,srg,Ntot)
c-------------Direct calculations
              call getDirect(ig,PFdirect,chgg,srg,Ntot)
              error=error+abs((PF(i)+PFNN-PFdirect)/PFdirect)
            enddo
          endif
        enddo
```

```
c-----Periodic boundary conditions (3dim.)
c-----Compare potential field with that obtained by Ewald calculations.
      elseif(iPBC.eq.1)then
        call Ewaldini(chgg,srg,Ntot)
        do node_id=0,nodes-1
c---------Each node writes the results
          if(myid.eq.node_id)then
            do i=1,Ntotn
c-------------Get global particle-ID
              ig=idg(i)
c-------------Contribution from particles in near neighbor cells
              call NNcont(ig,PFNN,chgg,srg,Ntot)
c-------------Ewald calculations
              call PEewald(ig,PFewld)
c-------------Macroscopic-dipole correction to the FMM results
              call dipole(ig,PFdipole)
              error=error+abs((PF(i)+PFNN+PFdipole-PFewld)/PFewld)
            enddo
          endif
        enddo
      else
c-------check iPBC
        write(*,*)'Not supported in this test program, iPBC=',iPBC
        stop
      endif

c-----Sum up relative error. Normalize it by Ntot.
      call MPI_ALLREDUCE(error,temp,1,MPI_DOUBLE_PRECISION,
     &     MPI_SUM,MPI_COMM_WORLD,ierr)
      error=error/Ntot

c-----Write relative error
      if(myid.eq.0)then
        write(*,*)'averaged relative-error of PF =',error
      endif

c-----Write timings for FMM calculations
      if(myid.eq.0)then
        write(*,*)'t_setup, t_comm, t_up, t_down, t_wrap,',
     &          '   t_pfs(sec)='
        write(*,1100)t_setup,t_comm,t_up,t_down,t_wrap,t_pfs
1100    format(6(1pe10.2))
      endif

      call MPI_FINALIZE(ierr)

      end
```

Appendix B. Listing of fmmp_dim.h for a sample program fmmptest.F

```
c================================c
c fmmp_dim.h
c Header file for fmmp.F
c Version 2.1
c================================c

c----------Begin user section

#define INCLUDE_STRESS 0
#define PFLM_PRECISION 1
#define DEBUG 0

#define Nsize_ 10000
#define iptop_ 5
#define msize_ 4800
#define ibsize_ 60000

c----------End user section

#if PFLM_PRECISION == 2
#define PFLM_TYPE complex*16
#else
#define PFLM_TYPE complex*8
#endif
```

Appendix C. Output of sample program fmmptest

Case: iPBC = 0

averaged relative-error of PF = 0.1077E–03					
t_setup	t_comm	t_up	t_down	t_wrap	t_pfs (sec)
1.45E–03	5.38E–02	5.67E–02	5.68E–01	3.00E–06	3.17E–03

Case: iPBC = 1

averaged relative-error of PF = 0.4799E–03					
t_setup	t_comm	t_up	t_down	t_wrap	t_pfs (sec)
1.42E–03	5.49E–02	5.77E–02	1.15E+00	1.47E–01	3.16E–03

Scalability test data

SP3 HABU

N/P = 100,000

P	lbot	N	t_setup	t_comm	t_up	t_down	t_wrap	t_pfs	total	efficiency
1	4	1e+05	0.573	0.012	0.155	87.7	0.138	1.1	89.7	1.00
8	5	8e+05	0.586	0.0514	0.195	88.4	0.139	1.12	90.4	0.99
64	6	6.4e+06	0.589	0.0725	0.218	88.5	0.138	1.12	90.6	0.99
512	7	5.12e+07	0.591	0.208	0.353	88.8	0.140	1.12	91.3	0.98

N/P = 500,000

P	lbot	N	t_setup	t_comm	t_up	t_down	t_wrap	t_pfs	total	efficiency
1	4	5e+05	2.84	0.0121	0.155	87.6	0.138	5.52	96.3	1.00
8	5	4e+06	2.92	0.0481	0.192	88.3	0.138	5.58	97.1	0.99
64	6	3.2e+07	2.92	0.0873	0.231	88.5	0.138	5.59	97.5	0.98
512	7	2.56e+08	2.92	0.222	0.367	88.8	0.139	5.60	98.0	0.98

N/P = 1,000,000

P	lbot	N	t_setup	t_comm	t_up	t_down	t_wrap	t_pfs	total	efficiency
1	4	1e+06	5.8	0.0121	0.156	87.6	0.138	11.1	105.0	1.00
8	5	8e+06	5.83	0.0595	0.204	88.3	0.139	11.2	106.0	0.99
64	6	6.4e+07	5.82	0.177	0.321	88.4	0.138	11.1	106.0	0.99
512	7	5.12e+08	5.84	0.332	0.47	88.9	0.140	11.4	107.0	0.98

SP4 MARCELLUS

N/P = 100,000

P	lbot	N	t_setup	t_comm	t_up	t_down	t_wrap	t_pfs	total	efficiency
1	4	1e+05	0.253	0.0056	0.066	35.2	0.055	0.53	36.1	1.00
8	5	8e+05	0.259	0.0307	0.092	35.2	0.053	0.53	36.1	0.99
64	6	6.4e+06	0.260	0.0515	0.113	35.2	0.056	0.53	36.2	0.99
512	7	5.12e+07	0.266	0.173	0.235	35.6	0.056	0.54	36.9	0.98

N/P = 500,000

P	lbot	N	t_setup	t_comm	t_up	t_down	t_wrap	t_pfs	total	efficiency
1	4	5e+05	1.26	0.0054	0.066	35.2	0.055	2.65	39.2	1.00
8	5	4e+06	1.31	0.0457	0.113	35.4	0.056	2.68	39.6	0.99
64	6	3.2e+07	1.30	0.0884	0.150	35.2	0.054	2.67	39.5	0.98
512	7	2.56e+08	1.32	0.140	0.202	35.3	0.056	2.69	39.7	0.98

N/P = 1,000,000

P	lbot	N	t_setup	t_comm	t_up	t_down	t_wrap	t_pfs	total	efficiency
1	4	1e+06	2.52	0.0057	0.066	35.2	0.055	5.29	43.1	1.00
8	5	8e+06	2.60	0.0921	0.154	35.2	0.055	5.35	43.5	0.99
64	6	6.4e+07	2.60	0.129	0.191	35.2	0.054	5.34	43.5	0.99
512	7	5.12e+08	2.62	0.278	0.339	35.3	0.056	5.35	43.9	0.98

References

[1] L. Greengard, V. Rokhlin, J. Comput. Phys. 60 (1985) 187;
L. Greengard, The Rapid Evolution of Potential Fields in Particle Systems, MIT, Boston, 1987.

[2] E.L. Pollock, J. Glosli, Comp. Phys. Commun. 95 (1996) 93.
[3] A. Nakano, et al., IEEE Comput. Sci. Engrg. 5 (1998) 68.
[4] W. Gropp, E. Lusk, A. Skkjellum, Using MPI Portable Parallel Programming with the Message-Passing Interface, MIT, Boston, 1994.
[5] T.J. Campbell, R.K. Kalia, A. Nakano, P. Vashishta, S. Ogata, S. Rodgers, Phys. Rev. Lett. 82 (1999) 4866.
[6] S. Ogata, H. Iyetomi, K. Tsuruta, F. Shimojo, A. Nakano, R.K. Kalia, P. Vashishta, J. Appl. Phys. 88 (2000) 6011.
[7] F.H. Streitz, J.W. Mintmire, Phys. Rev. B 50 (1994) 11996.
[8] S. Ogata, H. Iyetomi, K. Tsuruta, F. Shimojo, R.K. Kalia, A. Nakano, P. Vashishta, J. Appl. Phys. 86 (1999) 3036.
[9] C.A. White, M. Head-Gordon, J. Chem. Phys. 101 (1994) 6593.
[10] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Oxford Univ. Press, New York, 1987.
[11] A. Nakano, R.K. Kalia, P. Vashishta, Comp. Phys. Commun. 83 (1994) 197.
[12] H. Ding, N. Karasawa, W.A. Goddard III, Chem. Phys. Lett. 196 (1992) 6.
[13] A.Y. Toukmaji, J.A. Board Jr., Comp. Phys. Comm. 95 (1996) 73.
[14] C.G. Lambert, T.A. Darden, J.A. Board Jr., J. Comp. Phys. 126 (1996) 274.
[15] S.W. De Leeuw, J.W. Perram, E.R. Smith, Proc. Roy. Soc. London A 373 (1980) 27.
[16] M.W. Deem, J.M. Newsam, S.K. Sinha, J. Phys. Chem. 94 (1990) 8356.
[17] D.E. Parry, Surf. Sci. 49 (1975) 433; Surf. Sci. 55 (1976) 195.
[18] D.M. Heyer, M. Barber, J.H. Clarke, J. Chem. Soc., Faraday Trans. II 73 (1977) 1485.
[19] S.W. de Leeuw, J.W. Perram, Mol. Phys. 37 (1979) 1313.
[20] I.-C. Yeh, M.L. Berkowitz, J. Chem. Phys. 111 (1999) 3155.