Parallelization of the Fast Multipole Method for Molecular Dynamics Simulations on Multicore Computers

Nguyen Hai Chau

Faculty of Information Technology
VNUH University of Engineering and Technology
chaunh@vnu.edu.vn, nhchau@gmail.com

Abstract. We have parallelized the fast multipole method (FMM) on multicore computers using OpenMP programming model. The FMM is the one of the fastest approximate force calculation algorithms for molecular dynamics simulations. Its computational complexity is linear. Parallelization of FMM on multicore computers using OpenMP has been reported since the multicore processors become increasingly popular. However the number of those FMM implementations is not large. The main reason is that those FMM implementations have moderate or low parallel efficiency for high expansion orders due to sophisticated formulae of the FMM. In addition, parallel efficiency of those implementations for high expansion orders rapidly drops to 40% or lower as the number of threads increases to 8 or higher. Our FMM implementation on multicore computers using a combination approach as well as a newly developed formula and a computational procedure (A2P) solved the above issues. Test results of our FMM implementation on a multicore computer show that our parallel efficiency with 8 threads is at least 70% for moderate and high expansion orders $p = 4, 5, 6, 7$. Moreover, the parallel efficiency for moderate and high expansion orders gradually drops from 96% to 70% as the number of threads increases.

Keywords: molecular dynamics simulations, fast multipole method, multicore, OpenMP, parallelization.

1 Introduction

Molecular dynamics (MD) simulation methods [1] are orthodox means for studying large-scale physical/chemical systems. The methods were originally proposed in 1950s but they only began to use widely in the mid-1970s when digital computers became powerful and affordable. Nowadays MD methods are being continued to use widely for studying physical/chemical systems [2,3,4].

MD is simply stated that ones numerically solve the N-body problems of classical mechanics (Newton mechanics). Solving N-body problems for a large number of particles N in the considered systems requires a great amount of time and it needs powerful computers as well as efficient algorithms. Calculation of Coulombic interaction force is the most dominated task in solving N-body problem. The calculation often dominates 90-95% total calculation time of MD simulations [1,2,3,4].

The simplest and most accurate algorithm for calculation force is the direct summation (DS) algorithm. The DS calculates interaction force between every pair of particles in the system using Coulombic force formula:

$$\boldsymbol{F}_{ij} = k_e \frac{q_i q_j}{||\boldsymbol{r}_{ij}||^2} . \frac{\boldsymbol{r}_{ij}}{||\boldsymbol{r}_{ij}||}, \tag{1}$$

where $\boldsymbol{F}_{ij}$ is the Coulombic force between two particles located at $\boldsymbol{r}_i$ and $\boldsymbol{r}_i$ with charges are q_i and q_j, respectively; $k_e = \frac{1}{4\pi\epsilon_0} = \frac{c_0^2\mu_0}{4\pi} = c_o^2 10^{-7}\text{H m}^{-1}$ is the Coulomb constant and $\boldsymbol{r}_{ij} = \boldsymbol{r}_j - \boldsymbol{r}_i$. Here c_0 is the speed of light in vacuum, H is the Henry unit and m is the meter. The potential due to q_j at position $\boldsymbol{r}_i$ is

$$\Phi_{ij} = k_e \frac{q_j}{||\boldsymbol{r}_{ij}||}. \tag{2}$$

The DS has $O(N^2)$ computational complexity, where N is the number of particles in the system. However DS is only applicable for small particle systems where N is up to 10^5. Using DS for larger systems will consume a huge amount of time. To reduce force calculation cost for N-body simulations, fast algorithms such as Barnes-Hut treecode (BH) [5,6] and fast multipole method (FMM) [7,8,9] has been developed. The computational complexity of the algorithms are $O(N \log N)$ and $O(N)$, respectively.

FMM has a broad range of applications in many fields of research: large-scale molecular dynamics simulations [10], accelerating boundary elements methods [11], vortex methods [12] etc. Large-scale N-body or MD simulations those have a very large N or where periodic boundary condition is not applicable are good examples of FMM's applications.

Because of the wide usability of FMM, there are many efforts to parallelize FMM for achieving high performance simulations. There are different approaches of FMM parallelization have been done so far. The first and most popular one is parallelization of FMM on distributed memory platforms using message passing interface (MPI). The second one is usage of special-purpose hardware for parallelization of FMM, including graphics processing units (GPU), GRAPE (GRAvity piPE) computer family and others. The third one is using OpenMP programming model for multiprocessor or multicore platforms. The last one combines the mentioned above approaches.

Parallelization of FMM using OpenMP programming model has been reported since the multicore processors become increasingly popular. However, there are is a small numbers of FMM implementations for OpenMP. The reason is as follows. Due to FMM's sophisticated formulae and data structures, existing OpenMP implementations of FMM have moderate or low parallel efficiency. The parallel efficiency often drops down for high expansions orders those needed by high accuracy applications such as molecular dynamics simulations. As an example, parallel efficiency for 8 threads of a multi-level FMM implementation using OpenMP is about 40% and drops to less than 25% with 16 threads [13]. Parallel efficiency of another FMM implementation using OpenMP on a single node of the Kraken supercomputer is relatively high (78%) for a low expansion order $p = 3$ [14]. However such a low expansion order $p = 3$ is suitable for N-body simulations in astrophysics but not for molecular dynamics simulations. Note that molecular dynamics simulations often require high expansion orders to achieve higher accuracy than astrophysics simulations do. The parallel efficiency of this implementation for higher expansion orders is not reported in details and drops down rapidly. The main drawback of the authors in [14] is that the parallelism of the M2L kernel in their implementation becomes finer with high order expansions. This would affect parallel efficiency of the implementation favourably.

In this paper we describe our approach for implementation of FMM on multicore computers using OpenMP programming model [15] to overcome drawbacks of the existing implementations. We develop a new formula for L2L stage and a computational procedure to simplify thus speed up the far field force stage of the FMM. The main advantages our approach is its simplicity and parallel efficiency.

The rest parts of this paper are as follows. In section 2, we describe the original FMM and its variations. The implementation of FMM on multicore computers is presented in section 3. Section 4 describes experimental results and section 5 concludes.

2 The Fast Multipole Method and Its Variations

In this section, we describe briefly the fast multipole method (section 2.1), and the most relevant algorithms to our work: the Anderson's method (section 2.2) and the pseudoparticle multipole method (section 2.3) by Makino.

2.1 Fast Multipole Method

The FMM is an $O(N)$ approximate algorithm to calculate forces among particles. The $O(N)$ scaling is achieved by approximation of the forces using the multipole and local expansion techniques. The algorithm is applicable for both two-dimensional [7] and three-dimensional [8,9] particle systems.

Figure 1 shows schematic idea of force approximation in the FMM. The force from a group of distant particles are approximated by a multipole expansion (M2M). At an observation point, the multipole expansion is converted to local expansion (M2L). The local expansion is then evaluated by each particle around the observation point (L2L).

A hierarchical tree structure (the octree) is used for grouping the particles. In all FMM implementations, the particle system is assume to locate inside a cube refering as the root cell. The root cell is then subdivided into eight equal subcells and the subdivision process for subcells continues until certain criteria is met. The root cell and subcells form an octree and the subdivision process is the octree construction. The octree construction stops when the number of

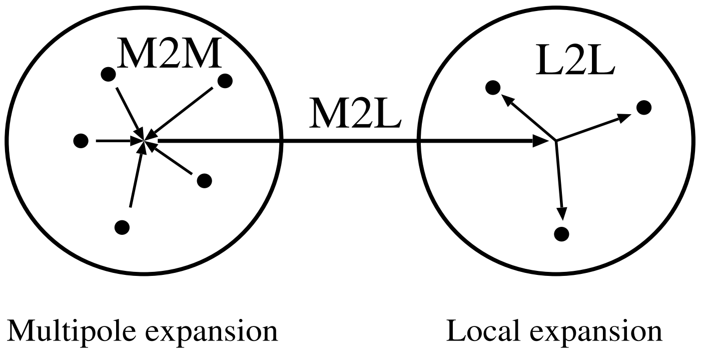

Fig. 1. Schematic idea of force approximation in FMM

octree levels reach a predefined number, often defined by the required accuracy of the simulations. In original FMM and its variations, the octree constructions are simple and similar. We refer readers to Greengard and Rokhlin's paper for details of the octree construction [7,8].

After the completion of the octree construction, the FMM goes to its main stages: multipole expansions to multipole expansions transition (M2M), multipoles expansion to local expansions conversion (M2L), local expansions to local expansions transition (L2L) and finally force evaluation. Among them, the M2L stage in the most computationally time consuming. The force evaluation stage contains two parts: near field force evaluation and far field force evalution. All the variations of the FMM follow exactly stages in the original FMM: M2M, M2L, L2L and force evalution. The only difference is that each variation uses different mathematical formulae for the stages.

In the rest part of this section (2.1) we briefly describe the mathematical formulae for M2M, M2L and L2L by Greengard, Cheng and Rokhlin [9].

Spherical Harmonics. We begin by the definition of the spherical harmonics function of degree n and order m by the formula

$$Y_n^m(\theta,\phi) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} P_n^{|m|} cos(\theta) e^{im\phi}. \tag{3}$$

Here, P_n^m is the assosiated Legendre functions, defined by Rodrigues' formula

$$P_n^m(x) = (-1)^m (1-x^2)^{m/2} \frac{d^m}{dx^m} P_n(x), \tag{4}$$

where $P_n(x)$ denotes the Legendre polynomials of degree n.

Multipole Expansion. Given the definition of the spherical harmonics function, the multipole expansion is defined as

$$\Phi(X) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{M_n^m}{r^{n+1}} Y_n^m(\theta, \phi), \tag{5}$$

where $X_1, X_2, ..., X_n$ are positions of N charges of strength $q_1, q_2, ..., q_n$ with spherical coordinates $(\rho_1, \alpha_1, \beta_1)$, $(\rho_2, \alpha_2, \beta_2)$, ..., $(\rho_n, \alpha_n, \beta_n)$, respectively. Assume that $X_1, X_2, ..., X_n$ are inside a sphere of radius a centered at the origin. The point X's spherical coordinate is (r, ϕ, θ). The M_n^m is defined as

$$M_n^m = \sum_{i=1}^{N} q_i \rho_i^n Y_n^{-m}(\alpha_i, \beta_i). \tag{6}$$

Local Expansion. The local expansion is defined as

$$\Phi(X) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} L_j^k Y_j^k(\theta, \phi) r^j, \tag{7}$$

where

$$L_j^k = \sum_{l=1}^{N} q_l \frac{Y_j^{-k}(\alpha_l, \beta_l)}{\rho_l^{j+1}}. \tag{8}$$

Here we have N charges of strength $q_1, q_2, ..., q_n$ located at $X_1, X_2, ..., X_n$ with spherical coordinates $(\rho_1, \alpha_1, \beta_1)$, $(\rho_2, \alpha_2, \beta_2)$, ..., $(\rho_n, \alpha_n, \beta_n)$, respectively and all the points $X_1, X_2, ..., X_n$ are outside of a sphere radius a centered at the origin. The point X's spherical coordinate is (r, ϕ, θ).

Transition of a Multipole Expansion (M2M). Assume that N charges of strength $q_1, q_2, ..., q_n$ are located inside a sphere D of radius a centered at $X_0 = (\rho, \alpha, \beta)$. Suppose that for any point $X = (r, \phi, \theta) \in \mathbb{R}^3 \setminus D$, the potential due to these charges is given by the multipole expansion

$$\Phi(X) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{O_n^m}{r'^{n+1}} Y_n^m(\theta', \phi'), \tag{9}$$

where (r', θ', ϕ') are the spherical coordinates of the vector $X - X_0$. Then for any point $X = (r, \theta, \phi)$ outside a sphere $D1$ of radius $a + \rho$ center at the origin,

$$\Phi(X) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{M_j^k}{r^{j+1}} Y_j^k(\theta, \phi), \tag{10}$$

where

$$M_j^k = \sum_{n=0}^{j} \sum_{m=-n}^{n} \frac{O_{j-n}^{k-m} i^{|k|-|m|-|k-m|} A_{j-n}^{k-m} A_n^m \rho^n Y_n^{-m}(\alpha, \beta)}{A_j^k}, \tag{11}$$

with A_n^m defined by

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}} \tag{12}$$

Conversion of a Multipole Expansion to a Local Expansion (M2L). Suppose that N charges of strengths $q_1, q_2, ..., q_n$ are located inside the sphere D_{X_0} of radius a centered at the point $X_0 = (\rho, \alpha, \beta)$, and that $\rho > (c+1)a$ for some $c > 1$. Then the corresponding multipole (13) converges inside the sphere D_0 of radius a centered at the origin. For any point $X \in D_0$ with coordinates (r, θ, ϕ), the potential due to the charges $q_1, q_2, ..., q_n$ is described by the local expansion

$$\Phi(X) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} L_j^k Y_j^k(\theta, \phi) r^j, \tag{13}$$

where

$$L_j^k = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{O_n^m i^{|k-m|-|k|-|m|} A_n^m A_j^k Y_{j+n}^{m-k}(\alpha, \beta)}{(-1)^n A_{j+n}^{m-k} \rho^{j+n+1}}, \tag{14}$$

with A_n^m defined by (12).

Translation of a Local Expansion (L2L). Suppose that X, X_0 are a pair of points in $\mathbb{R}^3$ with spherical coordinates $(\rho, \alpha, \beta), (r, \theta, \phi)$, respectively, and (r', θ', ϕ') are the spherical coordinates of the vector $X - X_0$ and p is a natural number. Let X_0 be the center of a p-th order local expansion with p finite; its expression at the point X is given by

$$\Phi(X) = \sum_{n=0}^{p} \sum_{m=-n}^{n} O_n^m Y_n^m(\theta', \phi') r'^n. \tag{15}$$

Then

$$\Phi(X) = \sum_{j=0}^{p} \sum_{k=-j}^{j} L_j^k Y_j^k(\theta, \phi) r^j, \tag{16}$$

everywhere in $\mathbb{R}^3$, with

$$L_j^k = \sum_{n=j}^{p} \sum_{m=-n}^{n} \frac{O_n^m i^{|m|-|m-k|-|k|} A_{n-j}^{m-k} A_j^k Y_{n-j}^{m-k}(\alpha, \beta) \rho^{n-j}}{(-1)^{n+j} A_n^m}, \tag{17}$$

and A_n^m defined by (12).

The M2M, M2L and L2L stages of the FMM are depicted in Figure 1. In sections 2.2 and 2.3, we describe two variations of the FMM those are the most relevant to our approach: Anderson's and Makino's methods.

2.2 Anderson's Method

Anderson [16] proposed a variant of the FMM using a new formulation of the multipole and local expansions. The advantage of his method is its simplicity. Anderson's method makes the implementation of the FMM significantly simple. The following is a brief description of Anderson's method.

Anderson's method is based on the Poisson's formula. This formula gives solution of the boundary value problem of the Laplace equation. When the potential on the surface of a sphere of radius a is given, the potential Φ at position $\boldsymbol{r} = (r, \phi, \theta)$ is expressed as

$$\Phi(\boldsymbol{r}) = \frac{1}{4\pi} \int_S \sum_{n=0}^{\infty} (2n+1) \left(\frac{a}{r}\right)^{n+1} P_n \left(\frac{\boldsymbol{s} \cdot \boldsymbol{r}}{r}\right) \Phi(a\boldsymbol{s}) ds \tag{18}$$

for $r \geq a$, and

$$\Phi(\boldsymbol{r}) = \frac{1}{4\pi} \int_S \sum_{n=0}^{\infty} (2n+1) \left(\frac{r}{a}\right)^{n} P_n \left(\frac{\boldsymbol{s} \cdot \boldsymbol{r}}{r}\right) \Phi(a\boldsymbol{s}) ds \tag{19}$$

for $r \leq a$. Note that here we use a spherical coordinate system. Here, $\Phi(a\boldsymbol{s})$ is the given potential on the sphere surface. The area of the integration S covers the surface of the unit sphere centered at the origin. The function P_n denotes the n-th Legendre polynomial.

In order to use these formulae as replacements of the multipole and local expansions, Anderson proposed a discrete version of them, i.e., he truncated the right-hand side of the Eq. (18)–(19) at a finite n, and replaced the integrations over S with numerical ones using a spherical t-design. Hardin and Sloane define the spherical t-design [17] as follows.

A set of K points $\wp = \{P_1, ..., P_K\}$ on the unit sphere $\Omega_d = S^{d-1} = \{x = (x_1, ..., x_d) \in R^d : x \cdot x = 1\}$ forms a spherical t-design if the identity

$$\int_{\Omega_d} f(x) d\mu(x) = \frac{1}{K} \sum_{i=1}^{K} f(P_i) \tag{20}$$

(where μ is uniform measure on Ω_d normalized to have total measure 1) holds for all polynomials f of degree $\leq t$ [17].

Note that the optimal set, i.e., the smallest set of the spherical t-design is not known so far for general t. In practice we use spherical t-designs as empirically found by Hardin and Sloane. Examples of such t-designs are available at `http://www.research.att.com/~njas/sphdesigns/`.

Using the spherical t-design, Anderson obtained the discrete versions of (18) and (19) as follows:

$$\Phi(\boldsymbol{r}) \approx \sum_{i=1}^{K} \sum_{n=0}^{p} (2n+1) \left(\frac{a}{r}\right)^{n+1} P_n \left(\frac{\boldsymbol{s}_i \cdot \boldsymbol{r}}{r}\right) \Phi(a\boldsymbol{s}_i) w_i \tag{21}$$

for $r \geq a$ (outer expansion) and

$$\Phi(\boldsymbol{r}) \approx \sum_{i=1}^{K} \sum_{n=0}^{p} (2n+1) \left(\frac{r}{a}\right)^{n} P_n \left(\frac{\boldsymbol{s}_i \cdot \boldsymbol{r}}{r}\right) \Phi(a\boldsymbol{s}_i) w_i \tag{22}$$

for $r \leq a$ (inner expansion). Here w_i is constant weight value and p is the number of untruncated terms.

Anderson's method uses Eq. (21) for M2M, M2L stages and (22) for L2L transistions. The procedures of other stages are the same as that of the original FMM.

2.3 Pseudoparticle Multipole Method

Makino [18] proposed the pseudoparticle multipole method (P^2M^2) – yet another formulation of the multipole expansion. The advantage of his method is that the expansions can be evaluated using simple equations Eq. (1) or Eq. (2).

The basic idea of P^2M^2 is to use a small number of pseudoparticles to express the multipole expansions. In other words, this method approximates the potential field of physical particles by the field generated by a small number of pseudoparticles. This idea is very similar to that of Anderson's method. Both methods use discrete quantities to approximate the potential field of the original distribution of the particles. The difference is that P^2M^2 uses the distribution of point charges, while the Anderson's method uses potential values. In the case of P^2M^2, the potential is expressed by point charges using Eq. (2).

In the following, we describe the formulation procedure of P^2M^2. The distribution of pseudoparticles is determined so that it correctly describes the coefficients of a multipole expansion. A naive approach to obtain the distribution is to directly invert the multipole expansion formula. For relatively small expansion order, say $p \leq 2$, we can solve the inversion formula, and obtain the optimal distribution with minimum number of pseudoparticles [19].

However, it is rather difficult to solve the inversion formula for higher p, since the formula is nonlinear. For solution with $p > 2$, Makino fixed the pseudoparticles positions given by the spherical t-design [17], and only their charges can change. This makes the formula linear, although the necessary number of pseudoparticles increases. The degree of freedom assigned to each pseudoparticle is then reduced from four to one.

Makino's approach gives the solution of the inversion formula as follows:

$$Q_j = \sum_{i=1}^{N} q_i \sum_{l=0}^{p} \frac{2l+1}{K} \left(\frac{r_i}{a}\right)^{l} P_l(\cos\gamma_{ij}), \tag{23}$$

where Q_j is charge of pseudoparticle, $\boldsymbol{r}_i = (r_i, \phi, \theta)$ is position of physical particle, γ_{ij} is angle between $\boldsymbol{r}_i$ and position vector $\boldsymbol{R}_j$ of the j-th pseudoparticle. For the derivation procedure of Eq. (23), see [18].

3 Implementation of the FMM on Multicore Computers Using OpenMP

3.1 A New Calculation Procedure for L2L Stage

As described above, the FMM has five stages including the octree construction, M2M transition, M2L conversion, L2L transition and force evaluation. The force evaluation stage contains two parts: near field and far field force evaluation. Anderson's method uses outer expansion (Eq. (21)) for M2M and M2L stages and inner expansion (Eq. (22)) for L2L stage. The P^2M^2 method by Makino is only applicable for M2M stage. However using Makino's method the M2L stage is simplied significantly by using direct pair-wise interaction given in Eq. (2).

We have done two implementations of the FMM for the special-purpose computer GRAPE so far. In the first implementation (hereafter FMMGRAPE1) [20], we combined Anderson's method and Makino's method. The P^2M^2 formula is used for M2M stage, then Eq. (2) is used for M2L. Next, the inner expansion by Anderson in Eq. (22) is used for L2L. With this approach, M2L is simplified using Eq. (2) and speeded up thanks to the special-purpose computer GRAPE. Another advantage of the combination of Anderson's and Makino's methods is that it is easy to parallelize the M2L stage for multicore architecture. However a new computational bottleneck appears in far-field force calculation as follow.

Using Eq. (22), the far field potential on a particle at position $\boldsymbol{r}$ can be calculated from the set of potential values of the leaf cell that contains the particle. Consequently, the far field force is calculated using derivative of Eq. (22) [20]:

$$-\nabla\Phi(\boldsymbol{r}) = \sum_{i=1}^{K}\sum_{n=0}^{p}\left(nr\boldsymbol{r}P_n(u) + \frac{u\boldsymbol{r} - \boldsymbol{s}_i\, r}{\sqrt{1-u^2}}\nabla P_n(u)\right)(2n+1)\frac{r^{n-2}}{a^n}g(a\boldsymbol{s_i})w_i, \quad (24)$$

where $u = \boldsymbol{s_i}\cdot\boldsymbol{r}/r$. Force calculation using Eq. (24) is complicated and hard to parallelize efficiently. In FMMGRAPE1, calculation of Eq. (24) dominates a significant part of the total calculation [20].

In the second implementation (hereafter FMMGRAPE2) [21], we fixed the bottleneck by developing a new formula and a new conversion procedure named A2P. We first developed a new formula for inner expansion using pseudoparticles. Eq. (23) by Makino gives the solution for outer expansion. We followed a similar approach [22] and proved that the solution for inner expansion is

$$Q_j = \sum_{i=1}^{N} q_i \sum_{l=0}^{p} \frac{2l+1}{K}\left(\frac{a}{r_i}\right)^{l+1} P_l(\cos\gamma_{ij}). \quad (25)$$

The A2P conversion is used to obtain a distribution of pseudoparticles that reproduces the potential field given by Anderson's inner expansion. Once the distribution of pseudoparticles is obtained, L2L stage can be performed using formula (Eq. (25)), and then the force evaluation stage is totally done using the simple Eq. (1). Hereafter we describe the A2P procedure indetails.

For the first step, we distribute pseudoparticles on the surface of a sphere with radius b using the spherical t-design. Here, b should be larger than the radius of the sphere a on which Anderson's potential values $g(a\boldsymbol{s}_i)$ are defined. According to Eq. (25), it is guaranteed that we can adjust the charge of the pseudoparticles so that $g(a\boldsymbol{s}_i)$ are reproduced. Therefore, the relation

$$\sum_{j=1}^{K} \frac{Q_j}{|\boldsymbol{R}_j - a\boldsymbol{s}_i|} = \Phi(a\boldsymbol{s}_i) \tag{26}$$

should be satisfied for all $i = 1..K$. Using a matrix $\mathcal{R} = \{1/|\boldsymbol{R}_j - a\boldsymbol{s}_i|\}$ and vectors $\boldsymbol{Q} = {}^T[Q_1, Q_2, ..., Q_K]$ and $\boldsymbol{P} = {}^T[\Phi(a\boldsymbol{s}_1), \Phi(a\boldsymbol{s}_2), ..., \Phi(a\boldsymbol{s}_K)]$, we can rewrite Eq. (26) as

$$\mathcal{R}\boldsymbol{Q} = \boldsymbol{P}. \tag{27}$$

In the next step, we solve the linear equation (27) to obtain charges Q_j. For a given cell with edge length is 1.0, the radius a and b for outer expansion and inner expansion are 0.75 and 6.0, respectively. Because of that, solving the linear equations system (27) is simply performing matrix-vector multiplication of $\mathcal{R}^{-1}$ and $\boldsymbol{P}$. Once solution of Q_j is obtained, far-field force calculated in Eq. (24) is replaced by the calculation pairwise interactions with Q_j using Eq. (1) that is much simpler. Note that the calculation of $\mathcal{R}^{-1}$ is simple and takes a negligible amount of time.

We have done numerical tests for accuracy of potential and force calculation performed with Eq. (25) [21]. In the tests, we approximate force and potential exerted from a particle q to a point L using Eq. (25) and the A2P procedure and compare results with potential and force calculated using Eq. (1) and Eq. (2). We change the distance r from q to L in a range of [1,10] and calculate relative error for both potential and force exerted from q to L. The test results shows that for expansion orders $p = 1$ to 5, potential error scales as $r^{-(p+2)}$ and force error scales as $r^{-(p+1)}$ as theoretically expected. For $p = 6$, potential and force error scales as $r^{-(p+2)}$ and $r^{-(p+1)}$ for $r < 6$, respectively and slowly descreasing for $r \geq 6$. The test results show that the Eq. (25) gives similar numerical accuracy of Anderson's given in Eq. (22).

Tables 1 and 2 compare formulae in original FMM to its variations and describe the mathematical formulae we have used in stages of our own FMM implementations.

3.2 Parallelization of FMM Using OpenMP

As shown in Tables 1 and 2, we combine methods of Anderson and Makino and our new calculation procedure A2P to implement the FMM. We apply the same approach to implement FMM on multicore computer. Hereafter we refer this implementation as FMMOpenMP. The main advantages of this combination is that we are able to use very simple mathematical formulae for stages of the FMM. In computational aspect, we have four kernels of calculation. The first one is the Eq. (1) used for near and far field force calculation. The second one is

Table 1. Mathematical formulae used in different variations of FMM

Stages	Original FMM	Anderson's method	Makino's method
M2M	Eq. (9), (10)	Eq. (21)	Eq. (23)
M2L	Eq. (13), (14)	Eq. (21)	Eq. (2)
L2L	Eq. (15), (16)	Eq. (22)	*Not available*
Near field force	Eq. (1)	Eq. (1)	Eq. (1)
Far field force	Evaluation of local expansions	Eq. (24)	*Not available*

the Eq. (2) used for M2L stage. The third one is the Eq. (23) used for M2M stage and the last one (Eq. (22)) used for L2L stage. Eq. (25) and Eq. (26) do not dominate computationally. However, Eq. (25) and Eq. (26) help us to calculate the far field force using the simple Eq. (1).

Table 2. Mathematical formulae used in our implementations of FMM

Stages	FMMGRAPE1	FMMGRAPE2	FMMOpenMP
M2M	Eq. (23)	Eq. (23)	Eq. (23)
M2L	Eq. (2)	Eq. (2)	Eq. (2)
L2L	Eq. (22)	Eq. (22)	Eq. (22)
Near field force	Eq. (1)	Eq. (1)	Eq. (1)
Far field force	Eq. (24)	Eq. (1), Eq. (25), Eq. (26)	Eq. (1), Eq. (25), Eq. (26)

Parallelization of the FMM using OpenMP becomes easier with our combination method. We need to parallelize the loops for potential/force pairwise interaction and the loops that used Eq. (23) and Eq. (22). We have developed a flops counter to find optimal level of the octree. As reported from the counter, number of flops due to pairwise interactions takes at least 90% number of FMM floating point operation. Therefore parallelization of the pairwise interaction is the most important task. Thanks to Eq. (1), Eq. (2) for their simplicity, the parallelization pseudocode of Eq. (1) (for near and far field force calculation) and Eq. (2) (for M2L stage) is straightforward as follows:

```
#pragma omp parallel for default(shared) private(i,j,...)
for (j=0;j<k;j++) { // For each destination particle
   for (i=0;i<n;i++) { // For each source particle
      // Calculate distance between particle i and particle j
                            ...
      // Calculate Coulombic force (Eq. (1)) or
      // Coulombic potential (Eq. (2))
                            ...
   }
}
```

Parallelization of M2M and L2L stages are also simple. The following is the parallelization pseudocode of the M2M calculation:

```
for (l=levels-1;l>=0;l--) { // Traverse octree from leaf to root
   #pragma omp parallel for default(shared) private(node,...)
   for (node=0;node<num;node++) { // For every node in this level
      // if the current node is a leaf then calculate
      // pseudoparticles masses based on postions and masses
      // of real particles inside the node,
      // otherwise calculate pseudoparticles masses based on
      // positions and masses of the pseudoparticles
      // of its children nodes.
   }
}
```

and the parallelization pseudocode of L2L is

```
#pragma omp parallel for default(shared) private(i,j,...)
   for (i=0;i<nbchild;i++) { // For every child of the current node
      // if the child contains no real particles then ignore,
      // otherwise perform L2L using inner expansion formula
   }.
```

We see that the parallelization of the FMM using OpenMP is relatively simple using our calculation scheme. Our calculation scheme has been implemented for the special-purpose computer GRAPE and achieved a speedup from 3-60 depending on accuracy of force calculation. Since our method uses simple mathematical formulae, it is also simple to parallelize FMM on multicore computers as shown above. We will show our experimental results in the next section.

4 Experimental Results

The FMMOpenMP is tested on a multiprocessor computer. The computer is equiped with four dual-core Intel(R) Xeon(R) CPU X5355 2.66 GHz processors and 4 GB RAM so that it is able to run 8 threads in parallel. The computer runs x86_64 Ubuntu operating system with 2.6.32-21 Linux kernel. We use `gcc 4.4.3` compiler with POSIX threading model enabled. We develop the FMMOpenMP using C++ programming language.

We performed tests on performance and parallel efficiency of the FMM OpenMP. In all the tests, we distributed particles uniformly in a unit cube centered at the origin and evaluated force on all particles. The number of particles is from 64K to 8M, where K denotes 1024 and M denotes 1024×1024. The accuracy of FMM force calculation for uniform distribution of particles is described in table 3. Since the accuracy of force calculation with $p = 1$ and $p = 2$ is not enough for production runs, we perform tests for $p \geq 3$. Figure 2 shows FMM performance versus that of direct summation. We can see that the direct summation algorithm scales as $O(N^2)$ while FMM scales as $O(N)$. When N is

Table 3. Accuracy of FMM force calculation

Expansion order	Force RMS relative error
$p = 1$	1.6×10^{-1}
$p = 2$	2.1×10^{-2}
$p = 3$	4.9×10^{-3}
$p = 4$	7.4×10^{-4}
$p = 5$	1.6×10^{-4}
$p = 6$	5.7×10^{-5}
$p = 7$	1.7×10^{-5}

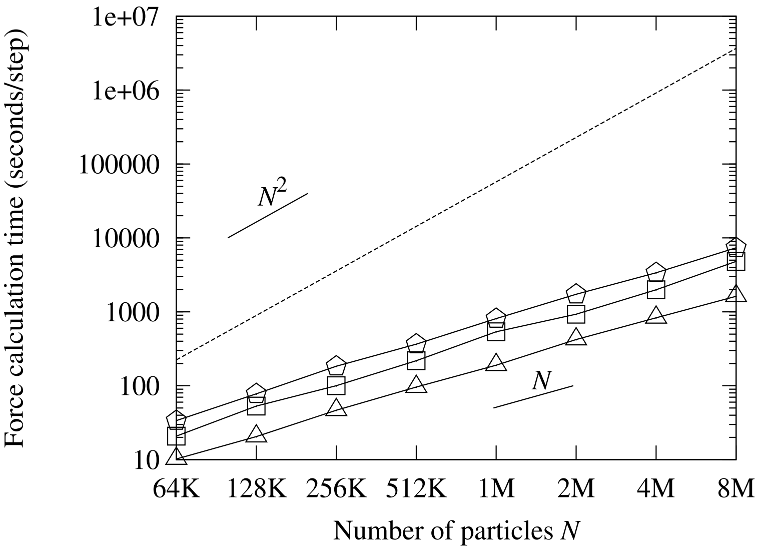

Fig. 2. Performance of direct summation and FMM algorithms. The dashed and solid curves represent the direct summation's and FMM's performance, respectively. Triangles, squares and pentagons denote performance of FMM with $p = 3$, $p = 5$ and $p = 7$, respectively.

8M, FMM runs faster than direct summation from 497 times to 2253 times when p runs from 7 to 3.

In the next tests, we show speedup and parallel efficiency of the FMMOpenMP on the computer described above. If P is the number of threads, T_1 is the execution time of the sequential FMM and T_P is the execution time of FMMOpenMP with P threads then speedup is

$$S_p = \frac{T_1}{T_P} \tag{28}$$

and parallel efficiency is

$$E_P = \frac{S_P}{P}. \tag{29}$$

Table 4. Speed up S_P of FMMOpenMP for N=8M

P	$p = 3$	$p = 4$	$p = 5$	$p = 6$	$p = 7$
2	1.52	1.91	1.91	1.93	1.91
4	2.23	3.59	3.60	3.62	3.62
8	2.36	5.47	5.52	5.60	5.53

Table 5. Parallel efficiency E_P of FMMOpenMP for N=8M

P	$p = 3$	$p = 4$	$p = 5$	$p = 6$	$p = 7$
2	76.1%	95.5%	95.7%	96.4%	95.4%
4	55.7%	89.7%	90.1%	90.5%	90.5%
8	29.5%	68.3%	69.0%	70.0%	69.1%

Tables 4 and 5 show speedup and parallel efficiency of FMMOpenMP for a test with 8M uniform distribution particles. The test results are similar for other values of N. Test results shows that speedup S_P of FMMOpenMP is low with expansion order $p = 3$ and becomes much higher with moderate and high order expansions $p = 4, 5, 6, 7$.

As a result, the parallel efficiency E_P of the FMMOpenMP rapidly drops from 76.1% to 29.5% when the number of threads increases from 2 to 8. However the parallel efficiency gradually drops from 96% to 70% with moderate and high expansions orders $p = 4, 5, 6, 7$. We can see that FMMOpenMP efficiency is better than that of Pan et. al. [13] which equal or lower than 40% for 8 threads.

As shown in Table 5, for a given number of threads the FMMOpenMP's parallel efficiency increases or unchanges as expansion order p increases. This behaviour is opposite to that of Yokota et. al. [14]. The parallel efficiency of Yokota's implementation is relatively high (78%) for low expansion order $p = 3$ but becomes low with high expansion orders [14]. The behaviour of parallel efficiency in our FMMOpenMP implementation shows that our approach is better than that of Yokota for applications require high accuracy of force calculation.

Reasons to explain our FMMOpenMP's behaviours include the usage of combination approach and simple mathematical formulae thanks to the A2P procedure. With the combination approach and A2P, the easy-to-parallelize pairwise interaction of FMMOpenMP, that has high parallel efficiency, dominates the total calculation. As expansion order p increases, the number of pairwise interaction increases accordingly and so does the parallel effciency.

5 Conclusions

We have successfully implemented the fast multipole method for multicore computers using OpenMP programming model. Test results show that it is simple to parallelize our FMM code using OpenMP thanks to the combination approach and A2P procedure. Our implementation's parallel efficiency for moderate and high expansion orders p are higher than those of Pan et. al. [13] and Yokota et. al. [14]. This behaviour makes our approach is suitable for high accuracy demand applications. The parallel efficiency of FMMOpenMP drops gradually for moderate and high expansion orders. This also shows another advantage of our implementation over those of Pan et. al. and Yokota et. al.

We still have room for improvements since the parallelization of M2M and L2L kernels is not optimized yet. The speedup and parallel efficiency of FMMOpenMP will become higher once the issues for improvements have been solved.

Acknowledgement. This work is partly supported by research project No. QG.2013 “Acceleration of the fast multipole method using graphics processing units and multicore computers for molecular dynamics simulations” granted by Vietnam National University, Hanoi.

References

1. Haile, M.: Molecular dynamics simulation: Elementary methods. Wiley-Interscience (1997)
2. Rappaport, D.C.: The art of molecular dynamics simulation, 2nd edn. Cambridge University Press (2004)
3. Satou, A.: Introduction to Practice of Molecular Simulation: Molecular Dynamics, Monte Carlo, Brownian Dynamics, Lattice Boltzmann and Dissipative Particle Dynamics. Elsevier (2010)
4. Griebel, M.: Numerical Simulation in Molecular Dynamics: Numerics, Algorithms, Parallelization, Applications. Springer (2010)
5. Barnes, J.E., Hut, P.: A hierarchical $O(N \log N)$ force calculation algorithm. Nature 324, 446–449 (1986)
6. Barnes, J.E.: A modified tree code: Don’t laugh, it runs. Journal of Computational Physics 87, 161–170 (1990)
7. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. Journal of Computational Physics 73, 325–348 (1987)
8. Greengard, L., Rokhlin, V.: A new version of the fast multipole method for the Laplace equation in three dimensions. Acta Numerica 6, 229–269 (1997)
9. Cheng, H., Greengard, L., Rokhlin, V.: A fast adaptive multipole algorithm in three dimensions. Journal of computational physics 155, 468–498 (1999)
10. Lupo, J.A., Wang, Z.Q., McKenney, A.M., Pachter, R., Mattson, W.: A large scale molecular dynamics simulation code using the fast multipole algorithm (FMD): performance and application. Journal of Molecular Graphics and Modelling 21, 89–99 (2002)
11. Gumerov, N.A., Duraiswami, R., Zotkin, D.N., Fantalgo, M.D.: Fast Multipole Accelerated Boundary Elements for Numerical Computation of the Head Related Transfer Function. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2007 (2007)
12. Gumerov, N.A., Duraiswami, R.: Efficient FMM accelerated vortex methods in three dimensions via the Lamb-Helmholtz decomposition. Submitted to Journal of Computational Physics, arXiv:1201.5430
13. Pan, X.M., Pi, W.C., Sheng, X.Q.: On OpenMP parallelization of the multilevel fast multipole algorithm. Progress in Electromagnetics Research 112, 199–213 (2011)

14. Yokota, R., Barba, L.: A Tuned and Scalable Fast Multipole Method as a Preeminent Algorithm for Exascale Systems, arXiv:1106.2176v2 [cs.NA] (2011)
15. `http://www.openmp.org`
16. Anderson, C.R.: An implementation of the fast multipole method without multipoles. SIAM Journal on Scientific and Statistical Computing 13(4), 923–947 (1992)
17. Hardin, R.H., Sloane, N.J.A.: McLaren's improved snub cube and other new spherical design in three dimensions. Discrete and Computational Geometry 15, 429–441 (1996)
18. Makino, J.: Yet another fast multipole method without multipoles - pseudoparticle multipole method. Journal of Computational Physics 151, 910–920 (1999)
19. Kawai, J.M.: Pseudoparticle multipole method: A simple method to implement a high-accuracy treecode. The Astrophysical Journal 550, L143–L146 (2001)
20. Chau, N.H., Kawai, A., Ebisuzaki, T.: Implementation of fast multipole algorithm on special-purpose computer MDGRAPE-2. In: Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics, SCI 2002, Orlando, Colorado, USA, July 14-18, pp. 477–481 (2002)
21. Chau, N.H., Kawai, A., Ebisuzaki, T.: Acceleration of fast multipole method using special-purpose computer GRAPE. International Journal of High Performance Computing Applications 22(2), 194–205 (2008)
22. Chau, N.H.: A new formulation for fast calculation of far field force in molecular dynamics simulations. Journal of Science (Mathematics-Physics) 23(1), 1–8 (2007)