応用力学論文集 Vol.8, pp.143-150 （2005年8月） 土木学会

大規模境界値問題における高速多重極積分方程式法のハイブリッド並列化

Hybrid parallelization of fast multipole accelerated BIEM for large-scale boundary value problems

池田健二*・西村直志**・吉川仁***
Kenji IKEDA, Naoshi NISHIMURA and Hitoshi YOSHIKAWA

*学生会員 京都大学大学院工学研究科 社会基盤工学専攻（〒606-8501 京都市左京区吉田本町）
**正会員 工博 京都大学学術情報メディアセンター教授（〒606-8501 京都市左京区吉田本町）
***正会員 工博 京都大学大学院工学研究科 社会基盤工学専攻助手（〒606-8501 京都市左京区吉田本町）

This paper presents an application of hybrid parallelization to the fast multipole accelerated boundary integral equation method. Expensive parts in the FMM (fast multipole method) algorithm are parallelized with MPI and OpenMP. The effectiveness of parallelization is examined with numerical examples, which include a two-dimensional potential problem with one hundred million DOFs.

Key Words : *BIEM, parallelization, fast multipole method, MPI, OpenMP*

1. はじめに

境界積分方程式法（境界要素法）[1]は工学に現れる偏微分方程式の有力な数値解法の一つであって、考える領域の境界にのみ未知数が現れるという特徴を有している。基本解の使用によりもたらされたこの特徴のため、境界積分方程式法は外部問題あるいはクラック問題の解析に特に適している。しかし、境界を離散化して得られた代数方程式の係数行列が密であるために計算時間が長くなってしまう。そのため、従来の境界積分方程式法が取り扱うことが出来る問題の規模は有限要素法や差分法等の領域型解法に比べて小さくなり、一時は境界積分方程式法の利用はごく特殊な問題に限られるようになっていた。しかし、多重極法[2]などの高速解法の出現により計算量を $O(N)$ (N: 未知数の数) 程度に下げることが可能となり、境界積分方程式法の大規模問題への適用が可能になった。さらに、多重極法を並列化することで、より大規模な問題が解けるようになると考えられる。

一方、近年、大型計算機の分野では大規模な共有メモリ型のノードを結合したクラスタが主流となっている。例えば地球シミュレータはこのようなタイプの計算機であるし、京都大学の学術情報メディアセンターにおいても、2004年3月に従来のベクトル型計算機 Fujitsu VPP 800 から、スカラ並列計算機 HPC2500 に移行した。HPC2500 は12台の SMP ノードからなり、それぞれのノードが128個の CPU を持ち512GB のメモリを共有している。このような環境においては、ノード内並列にはスレッド並列の利用が便利であり、複数ノードを用いた並列計算においてはプロセス並列が必須である。また、このようなアーキテクチャの性能を最大限に生かすためにはノード内のスレッド並列とノード間のプロセス並列を併用したハイブリッド並列が有効である。スレッド並列およびプロセス並列を実現するための API の de facto standard は、各々OpenMP[3]および MPI[4]であり、ハイブリッド並列においてはこれらを併用するのが一般的である。

多重極法の並列化に関する研究は従来から多数行なわれてきているが (例えば Yoshida ら[5]、Ying ら[6])、そのほとんどが MPI などを用いた分散メモリ型の並列計算機を想定したものである。共有メモリ型の並列計算機における多重極法の並列化の研究例として、宗像らは、2次元クラック問題および3次元静弾性問題に対して OpenMP 並列化を施して、その有効性を実証している[7]。また、大谷らは3次元時間域動弾性問題にハイブリッド並列化を施している[8]。しかし、前者は OpenMP によるスレッド並列のみ適用しており、後者は時間パラメータを含むため、各々空間自由度は100万程度に留まっている。そこで、本論文では更に大規模な解析を行なうため、2次元 Laplace 方程式における外部問題、及びクラック問題をとり上げ、OpenMP-MPI ハイブリッド並列化を適用する。特に、処理時間が大きい node 間通信を効率良く行なうための工夫について述べる。更に、得られた手法の効率を数値的に検証し、良好な台数効果が得られる事を示す。最後に、境界積分方程式法ではこれまで困難とされてきた未知数が一億元規模の問題の解析例を示し、更に大規模な解析が可能である事を示す。

2. 定式化

2.1 2次元外部 Laplace 問題

2次元 Laplace 方程式における外部 Dirichlet 問題は以下のように表される。
基礎方程式

$$u_{,ii} = 0 \quad \text{in} \quad D \tag{1}$$

境界条件 (Dirichlet 条件)

$$u = \bar{u} \quad \text{on} \quad \partial D \tag{2}$$

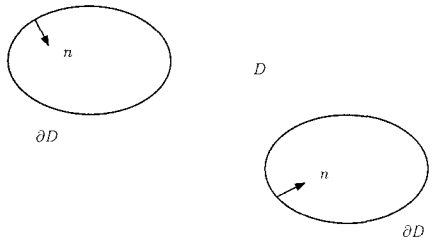
$$u(\boldsymbol{x}) \to 0 \quad \text{as} \quad |\boldsymbol{x}| \to \infty \tag{3}$$

ここで、D は 2 次元領域であり、境界 ∂D の外側である。境界 ∂D の単位法線ベクトル $\boldsymbol{n}$ は D から見て外向きとする (図 1)。また、$\bar{u}$ は与えられた関数である。

式 (1) に対応する積分方程式は次の通りである。

$$\frac{u(\boldsymbol{x})}{2} = \int_{\partial D} G(\boldsymbol{x}-\boldsymbol{y}) \frac{\partial u(\boldsymbol{y})}{\partial n} dS_y - \int_{\partial D} \frac{\partial G(\boldsymbol{x}-\boldsymbol{y})}{\partial n} u(\boldsymbol{y}) dS_y \quad \text{on } \partial D \tag{4}$$

ここに $\partial/\partial n$ は法線方向微分を表し、G は 2 次元



図–1 外部領域

Laplace 方程式の基本解

$$G(\boldsymbol{x}-\boldsymbol{y}) = -\frac{1}{2\pi} \log|\boldsymbol{x}-\boldsymbol{y}| \tag{5}$$

である。

2.2 2 次元 Laplace クラック問題

2 次元 Laplace 方程式におけるクラック問題は以下のように表される。

基礎方程式

$$u_{,ii} = 0 \quad \text{in} \quad R^2 \backslash S \tag{6}$$

クラック上での境界条件

$$\frac{\partial u}{\partial n} = 0 \quad \text{on} \quad S \tag{7}$$

クラック端での開口変位 0

$$\phi(\boldsymbol{x}) = u^+ - u^- = 0 \quad \text{on} \quad \partial S \tag{8}$$

無限遠点での u の値

$$u(\boldsymbol{x}) \to u^\infty(\boldsymbol{x}) \quad \text{as} \quad |\boldsymbol{x}| \to \infty \tag{9}$$

ここで、S はクラックであり、ϕ はクラック S の開口変位である。このとき、解の積分表示は次の式で表される。

$$u(\boldsymbol{x}) = u^\infty(\boldsymbol{x}) + \int_S \frac{\partial G(\boldsymbol{x}-\boldsymbol{y})}{\partial n_y} \phi(\boldsymbol{y}) dS_y \tag{10}$$

解の積分表示を法線微分することで、次のクラック問題の境界積分方程式が得られる。

$$\text{p.f.} \int_S \frac{\partial^2 G(\boldsymbol{x}-\boldsymbol{y})}{\partial n_x \partial n_y} \phi(\boldsymbol{y}) dS_y = -\frac{\partial u^\infty}{\partial n_x}(\boldsymbol{x}) \quad \text{on} \quad S \tag{11}$$

ここで、p.f. は発散積分の有限部分を表す。

3. 高速多重極法

前章で解説した 2 次元 Laplace クラック問題を例にとって、多重極法の手順を示す。詳しくは Nishimura[2] を参照されたい。

3.1 多重極展開

式 (11) の左辺の多重極展開を求める。まず、式 (11) の左辺の積分を複素積分で評価しておく。

$$\left(\frac{\partial}{\partial x_1} - i\frac{\partial}{\partial x_2}\right) \int_S \frac{\partial G(\boldsymbol{x}-\boldsymbol{y})}{\partial n_y} \phi(\boldsymbol{y}) dS_y = \frac{1}{2\pi i} \int_S \frac{\phi(\zeta)}{(z-\zeta)^2} d\zeta \tag{12}$$

ここで、i は虚数単位であり、

$$z = x_1 + ix_2 \quad , \quad \zeta = y_1 + iy_2 \tag{13}$$

である。

$$I_p(z) = \frac{z^p}{p!}, \quad O_p(z) = \frac{(p-1)!}{z^p} \tag{14}$$

として、式 (12) の右辺を $|z-z_0| < |\zeta - z_0|$ なる z_0 で展開すると、次のようになる。

$$\sum_{q=0}^{\infty} (-1)^q I_q(z-z_0) \sum_{p=q}^{\infty} O_{p+2}(z_0-\zeta_0) M_{p-q}(\zeta_0) \tag{15}$$

ここで、z_0 は z の近傍の点、ζ_0 は S を含むセル q の中心であり、多重極モーメント M_q は次式で定義される。

$$M_q(\zeta_0) = \frac{1}{2\pi i} \int_S I_q(\zeta - \zeta_0) \phi(\zeta) d\zeta \tag{16}$$

3.2 階層構造

高速多重極法では、対象領域に階層構造を採り入れて計算を行う。ここでは、2 次元空間に 4 分木構造を用いる。まず、考えている要素全体を含む正方形を作り、これをレベル 0 のセルと呼ぶ。次にその正方形を 4 等分して 4 つの正方形を作り、それらをレベル 1 のセルとする。この操作を繰り返して行き、セル内の要素数があらかじめ定めた値以下になった場合、そのセルに関しては分割をやめることにする。全てのセルに対して分割が終了するまでこの操作を続ける。ここで、あるレベルのセル A を分割して次のレベルのセル B が生じた場合、A は B の親であるといい、B は A の子であると呼ぶことにする。子を持たないセルのことをリーフと呼ぶ。

3.3 upward pass

リーフのセルで、セル内に含まれるすべての要素に対して、セル中心に関する多重極モーメントを式(16)より計算して足し合わせる。次に、多重極モーメントを親のセルに渡す。このとき、多重極モーメントの中心を親セルの中心に移動させる必要がある。親セルへの移動公式は次式で表され、M2M 公式とよばれる。

$$M_q(\zeta_1) = \sum_{r=0}^{q} I_{q-r}(\zeta_0 - \zeta_1) M_r(\zeta_0) \qquad (17)$$

このようにして、子セルからの寄与を足し合わせて親セルの多重極モーメントを求める。この各レベル間の多重極モーメントの移動をリーフから順に続けて行き、レベル 2 以下の全てのセルの中心回りの多重極モーメントを求める。以上の過程を upward pass と呼ぶ。

3.4 downward pass

式(15)を局所展開の形で書くと次のようになる。

$$\sum_{q=0}^{\infty} I_q(z - z_0) L_q(z_0) \qquad (18)$$

局所展開係数 L_q は次の M2L 公式により M_n と関係づけられる。

$$L_q(z_0) = \sum_{p=q}^{\infty} (-1)^q O_{p+2}(z_0 - \zeta_0) M_{p-q}(\zeta_0) \qquad (19)$$

downward pass とはレベル 2 のセルからレベルを下りながら局所展開係数を計算して行くプロセスである。局所展開係数は以下の手順で求められる。ある 1 つのセルに着目し、そのセルとは隣接してはいないが、それぞれの親のセル同士は隣り合っているようなセルを interaction list のセルと呼ぶ。最も上のレベルであるレベル 2 において、各セルは interaction list のセルの多重極モーメントから局所展開係数を M2L 公式によって計算する。次に、レベルを下げて行きながら、各セルは同様に interaction list のセルの多重極モーメントを M2L 公式によって変換し、自分の局所展開に足し合わせる。さらに、レベル 2 以外では親の局所展開係数を自分のセルの中心に移動して、自分のセルの局所展開係数に足し合わせる。その移動公式は次のようになり、これを L2L 公式と呼ぶ。

$$L_r(z_1) = \sum_{p \geq r} L_p(z_0) I_{p-r}(z_1 - z_0) \qquad (20)$$

以上の計算をリーフのセルに至るまで繰り返し、最後にリーフのセルでは自分の隣のセルの影響と自分自身のセル内の要素による影響を直接計算によって評価する。

2 次元外部 Laplace 問題においては、多重極展開は次のように表される。

$$\int_{\partial D} G(z,\zeta) \frac{\partial u(\zeta)}{\partial n} dS_\zeta - \int_{\partial D} \frac{\partial G(z,\zeta)}{\partial n} u(\zeta) dS_\zeta = \frac{1}{2\pi} \sum_{k=0}^{\infty} O_k(z - \zeta_0) M_k(\zeta_0) \qquad (21)$$

ここで、M_n は多重極モーメントであり、次のようになる。

$$M_k(\zeta_0) = \int_{\partial D} I_k(\zeta - \zeta_0) \frac{\partial u(\zeta)}{\partial n} dS_\zeta - \int_{\partial D} I_{k-1}(\zeta - \zeta_0) u(\zeta) n(\zeta) dS_\zeta \qquad (22)$$

また

$$O_0(z) = -\log z \qquad (23)$$

である。このとき、M2M 公式、M2L 公式、L2L 公式は次のように表される。

M2M 公式

$$M_k(\zeta_1) = \sum_{l=0}^{k} I_{k-l}(\zeta_0 - \zeta_1) M_l(\zeta_0) \qquad (24)$$

M2L 公式

$$L_l(z_0) = \frac{(-1)^l}{2\pi} \sum_{k=0}^{\infty} O_{l+k}(z_0 - \zeta_0) M_k(\zeta_0) \qquad (25)$$

L2L 公式

$$L_l(z_1) = \sum_{k=0}^{\infty} I_k(z_1 - z_0) L_{l+k}(z_0) \qquad (26)$$

4. 並列化

本論文では、大規模問題を取り扱う手段として OpenMP、MPI によるハイブリッド並列化を用いる。ノード間で MPI 並列を行うことにより、メモリを分散させて各プロセスに保持させておくことができる。ノード内では、共有メモリ型の並列計算に適した OpenMP を使うことにより、容易に各プロセス内での並列化を行うことができる。以下、多重極法をどのように並列化するかを示す。

4.1 多重極法における MPI 並列化

MPI 並列化を行うにあたって、まず最初にどのようにメモリをプロセスに割り当てるかを考える必要がある。多重極法の並列化を行う場合、セル分割を考慮したメモリ分散を行わなければならない。多重極法において、他のセルの情報が必要になるのは M2M、M2L、L2L の 3 つの通信部分のみである。各々の手順においてこの部分の通信時間を出来る限り減らすようにメモリを分散することを考える。ここで、M2M、L2L は親子間のやりとりに過ぎないので最も上のレベルでメモリを共有させておけば、それより下のレベルでは通信の必要はなくなり通信に要する時間を大幅に減らすことが出来る。

M2L 公式では、interaction list の多重極モーメントを集めて自分のセルの局所展開係数を計算する。そのため、異なるプロセス間での受け渡しが必要となる場合には MPI 通信を行う必要がある。しかし、セルのルー

プの中で通信をしていたのでは通信回数が多くなりすぎる。したがって、通信回数を減らすために、あらかじめ通信を行う多重極モーメントのリストを作っておき、一括してプロセス間通信を行うことにする。以上より、多重極法にMPI並列化を適用するためには次の4つの作業を行う部分をそれぞれ作成する必要がある。

1. セルをプロセスに振り分ける部分
2. 要素をプロセスに振り分ける部分
3. 通信を行う部分
4. 一括通信のためのリストを作る部分

1,2の部分においては、要素とその要素が属するセルの間の関係を保ちながらプロセスに振り分ける必要があるが、容易に作成可能であり、計算時間も全体からすれば無視できる程度である。3の部分はMPI並列化を行うに伴って生じる通信部分である。本研究の解析ではこの部分を、downward passの一番外側の、レベルに関するループのすぐ内側においた。4の部分は3の部分の通信時間を減らすための作業と考えて良い。しかし、プロセス内のあるセルに対してそのinteraction listを探すことは、セルに関する2重のループを回すことと等価であるから、未知数が一千万を越え、セルの数が10万を越えるような問題においては計算時間はdownward passなどと比較しても決して無視できない。そのため、このループをOpenMPで並列化することにより時間短縮を図った。この方法については、次の節で述べる。

4.2 多重極法におけるOpenMP並列化

OpenMPを用いた並列化では、並列化の効果が上がる部分のみを並列化することが可能である。本研究で取り扱った2次元外部Laplace問題、クラック問題を多重極法で解く場合には、次の3つが計算量の多い部分となっているので、これらをOpenMPで並列化することを考える。

- upward pass
- downward pass
- 前処理

さらに、前節で述べたMPI通信のための通信リストを作成する部分においても計算量が大きくなるのでOpenMP並列化を行う。

(1) upward pass

upward passの並列化は次のように、当該レベルでのセルに関するループにOpenMP並列化を施した。

子から親へ多重極モーメントを渡す際には、子のセルでループを回す方法と、親のセルでループを回す方法の二通りが考えられる。しかし、子のセルでループを回した場合、同一の親を持つセルが異なるスレッドに割り当てられてしまう可能性があり、スレッド間で競合が発生する危険性がある。そのため、本論文では親のセルでループを回す方法を採用している。

```
DO level=最下層のレベル −1 ,1, −1

!$OMP PARALLEL DO
    DO 当該レベルでのセル

        DO 子セル

            if (子セルがリーフ) then
                定義から多重極モーメントを計算
            else
                子セルの多重極モーメント
                を自分のセルに移動 (M2M)
            endif

        ENDDO

    ENDDO
!$OMP END PARALLEL DO

ENDDO
```

(2) downward pass

downward passの並列化は次のように、当該レベルでのセルに関するループにOpenMP並列化を施した。

```
DO level=2, (最下層のレベル)

!$OMP PARALLEL DO
    DO 当該レベルでのセル icell

        if (レベルが 2 でない) then
            親のセルから局所展開係数を移動
            (L2L)
        endif

        DO 当該レベルでのセル jcell
            if (icell と jcell が
            interaction list の関係にある) then

            多重極モーメントを
            局所展開係数に変換 (M2L)

            elseif (icell か jcell のどちらかがリーフ)
                直接計算
            endif

        ENDDO

    ENDDO
!$OMP END PARALLEL DO

ENDDO
```

(3) 前処理

前処理の並列化はリーフセルに関するループにOpenMP並列化を施した。

```
!$OMP PARALLEL DO
    DO リーフセルのブロック

        ブロックの行列の逆行列を計算して
        前処理に用いる

    ENDDO
!$OMP END PARALLEL DO
```

(4) 通信リストの作成

前節で述べたように、downward pass 内の M2L 変換において、MPI 通信の回数を少なくするためにあらかじめ、各セルの interaction list を調べておくという方針をとる。そのリストを配列

List(n, プロセス iproc, プロセス jproc, レベル l) に格納して行くことを考える。List にはレベル l においてプロセス i, プロセス j 間で通信を行わなければならないようなセルの番号が格納される。

この List を作成するための最も単純な方法は、次のような手順でこの List に値を格納するものである。ここで、iproc は i の属するプロセス、jproc は j の属するプロセスを表す。

```
DO level=2, 最下層のレベル

    DO i=当該レベルでのセル

        DO j=当該レベルでのセル

        if (i と j が interaction list の関係)
            if (iproc と jproc が異なる)
            List(0,iproc,jproc,lev)
            = List(0,iproc,jproc,lev)+1
            ntemp=List(0,iproc,jproc,lev)
            List(ntemp,iproc,jproc,lev)= i
            endif
        endif

        ENDDO

    ENDDO

ENDDO
```

しかし、先にも述べたようにこのループはセルが増えるにつれて計算時間が増大するので、OpenMP で並列化を行う必要がある。ただし、このままのコードでは OpenMP を適用することが非常に難しいので、次のように書き換えて並列化を行う。

```
!$OMP PARALLEL DO
DO m=1,(process 数)*(process 数)*(最下層のレベル-1)
iproc=mod(m-1,nprocs*nprocs)/nprocs
jproc=mod(m-1,nprocs*nprocs)-iproc*nprocs
level=(m-1)/(nprocs*nprocs)+2

    DO iproc 内のセル i

        DO jproc 内のセル j

        if (i,j が interaction list の関係である)
            List(0,iproc,jproc,lev)
            = List(0,iproc,jproc,lev)+1
            ntemp=List(0,iproc,jproc,lev)
            List(ntemp,iproc,jproc,lev)= i
        endif

        ENDDO

    ENDDO

ENDDO
!$OMP END PARALLEL DO
```

5. 数値解析

多重極積分方程式法にハイブリッド並列化を適用して、2 次元 Laplace 外部問題、クラック問題を解いた。以下では得られた数値結果を示す。

数値計算は京都大学学術情報メディアセンターの HPC2500 にて行った。HPC2500 は 12 台の計算ノードからなり、ノード間は高速光インターコネクトを用いて接続されている。ノード内では 512GB のメモリが共有されており、ノード当たりの CPU 台数は 128 である。CPU のクロックは 1 つのノードでは 2.08GHz であり、残り 11 ノードは 1.56GHz となっている。

5.1 2 次元 Laplace 外部問題

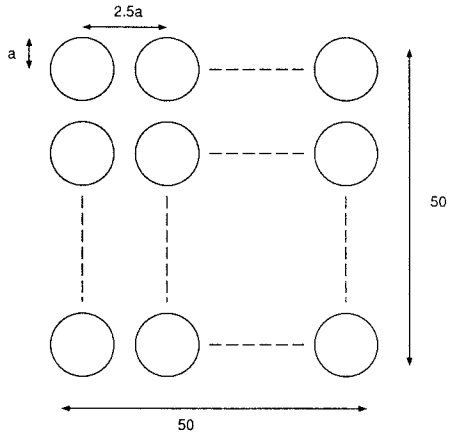
(1) 問題設定

半径 a、中心間隔 $2.5a$ の 50×50=2500 個の円孔が図 2 のように並んでいるとする。このとき、円孔の外部で Laplace 方程式が満たされているような外部問題を考える。境界条件は $u(x_1, x_2) = x_1^2 - x_2^2$ とした。一つの円孔を 100 個の要素に分割し、全未知数を 25 万とした。また、多重極展開および局所展開の打ち切り項数はともに 15 とし、前処理は行っていない。ソルバは GMRES を用いた。

(2) 解析結果

設定した問題に MPI-OpenMP ハイブリッド並列化を行い、プロセス数およびスレッド数を変えて解析を行った。それぞれの実時間をまとめると表 1 のようになった。GMRES の反復回数は 78 回であった。なお、計算は CPU のクロックが 2.08GHz のノードで行っている。表 1 より各 CPU 数において最も計算時間が速くなるような組合せは、表 2 のようになっている。

上の組合せに関して、CPU 数と speed up との関係を図 3 に示す。speed up は (CPU が 1 台のときの計算時間)/(計算時間) と定義する。(CPU 数)=(speed up)



図–2 円孔の配置図

表–1 計算時間 (外部問題)

プロセス数	スレッド数	実時間 (秒)
1	1	2914

プロセス数	スレッド数	実時間 (秒)
1	8	431
2	4	417
4	2	388
8	1	439

プロセス数	スレッド数	実時間 (秒)
1	16	236
2	8	219
4	4	209
8	2	220
16	1	229

プロセス数	スレッド数	実時間 (秒)
2	16	125
4	8	116
8	4	120
16	2	126

表–2 組合せ

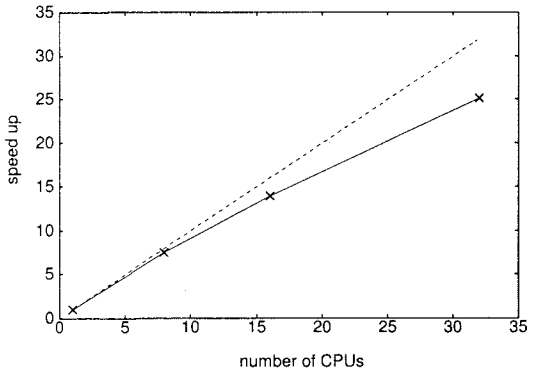
CPU 数	プロセス	スレッド
8	4	2
16	4	4
32	4	8

となるような理想的なグラフを点線で示す。図 3 から、良好な台数効果が得られていることが分かる。

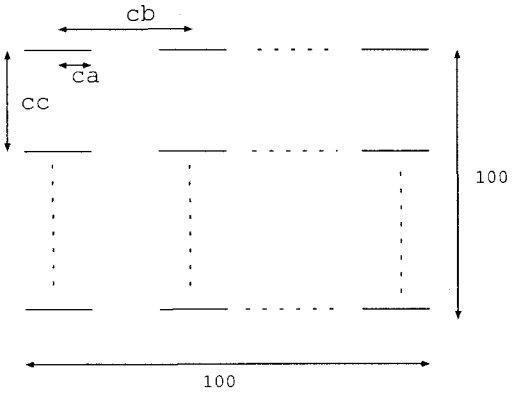
5.2 2 次元 Laplace クラック問題

(1) 問題設定

Laplace 方程式を満たす 2 次元の無限領域にクラックが図 4 のように、一様に分布している問題を考える。



図–3 CPU 数に対する speed up



図–4 クラックの配置図

クラックの長さと間隔は図 4 で *ca*:*cb*:*cc*=2:5:5 となるようにした。ここではクラックを縦方向に 100 個、横方向に 100 個の計 10000 個を配置する。1 つのクラックに要素が 100 個含まれるようにしたので、未知数の数は 100 万である。また、多重極展開および局所展開の打ち切り項数はともに 10 とした。前処理はリーフに対応するブロック対角行列を用いて右前処理を行った。ソルバは GMRES を用いた。

(2) 解析結果

2 次元外部 Laplace 問題同様、設定した問題に MPI-OpenMP ハイブリッド並列化を行い、プロセス数およびスレッド数を変えて解析を行った。それぞれの実時間をまとめると表 3 のようになった。GMRES の反復回数は 16 回であった。すべて CPU のクロックが 2.08GHz のノードで計算した。

表 3 より各 CPU 数において最も計算時間が速くなるような組合せは、表 4 のようになっている。上の組合せに関して、CPU 数と speed up との関係を図 5 に示す。(CPU 数)=(speed up) となるような理想的なグラフを点線で示す。図 5 から、良好な台数効果が得られていることが分かる。

(3) 問題規模の拡大

前節で扱った 2 次元 Laplace クラック問題を、クラックの数を増やすことで問題をさらに大型化して解析を行う。クラックの数を縦方向、横方向ともに 1000 個と

表–3 計算時間 (クラック問題)

プロセス数	スレッド数	実時間 (秒)
1	1	566

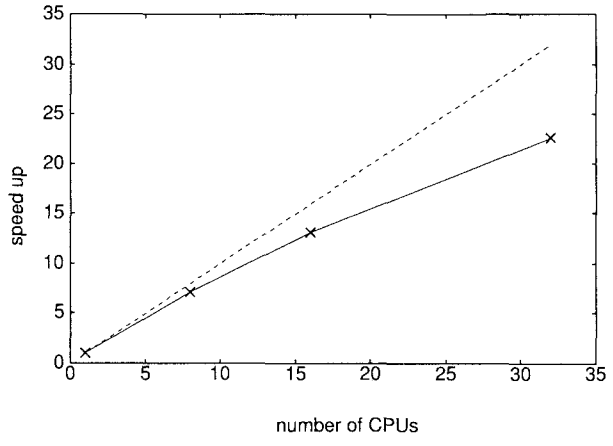
プロセス数	スレッド数	実時間 (秒)
1	8	85
2	4	90
4	2	87
8	1	79

プロセス数	スレッド数	実時間 (秒)
1	16	55
2	8	47
4	4	49
8	2	43
16	1	48

プロセス数	スレッド数	実時間 (秒)
1	32	39
2	16	30
4	8	32
8	4	34
16	2	25

表–4 組合せ

CPU 数	プロセス	スレッド
8	8	1
16	8	2
32	16	2



図–5 CPU 数に対する speed up

して各クラックを 100 個の要素に分割する。このとき、要素の数は 1000×1000×100 であり、未知数が 1 億の問題となる。このような大規模な問題ではスレッド並列のみによる並列化は不可能であり、メモリを分散させて計算を行う事が必要となる。なお、これまで粒子系の高速解法においては、MPI によるプロセス並列で 21 億自由度の問題を扱った例が知られているが[6]、線形方程式の求解を伴う積分方程式の解析で 1 億元を越えた解析の例は著者の知る限り見当たらない。

本研究では、この問題を MPI 並列 8 プロセス、OpenMP 並列 32 スレッドの合計 256CPU で実行した。ただし、ここでは CPU はクロックが 1.56GHz のものを用いた。多重極展開、局所展開の打ち切り項数は要素数 100 万の場合と同じく 10 とし、クラックの配置に関しても同じく ca:cb:cc=2:5:5 とした。

計算に要した時間は実時間で 6277 秒であった。反復解法として GMRES を用い、16 回で収束した。

各ルーチンごとに要した時間は次のようになっている。

表–5 計算時間

ルーチン	所要時間 (秒)
クラックのデータの作成	1.8
階層構造の作成	64.6
MPI 並列化の準備	856.5
連立方程式の解	4984.8
結果の出力	369.3
合計	6277.3

'MPI 並列化の準備' には、4 章の第一節で述べた、セルをプロセスに振り分ける部分、要素をプロセスに振り分ける部分、一括通信のためのリストを作る部分が含まれており、それぞれの所要時間は次のようになった。

表–6 計算時間

ルーチン	所要時間 (秒)
プロセス振り分け	74
MPI 通信のリスト作成	782

'連立方程式の解' には upward pass、downward pass、前処理、多重極モーメントの MPI 通信といった部分が含まれており、それぞれの所要時間は次のようになった。

表–7 計算時間

ルーチン	所要時間 (秒)
upward pass	74
downward pass	4733
前処理	29
MPI 通信	28

これらの結果から次のことが言える。前処理を行う箇所と upward pass に関しては OpenMP の並列効果が大きく、未知数が一億となるような巨大な問題においても計算時間を短い時間に抑えることができる。また、MPI 通信全体での時間は大幅に短縮されるが、その分 downward pass と MPI 通信リスト作成部分の全体に占める割合が大きくなる。

また、解析を行う上で必要な配列の大きさは 1 プロセスあたり約 26GB であった。先に述べたように京都

大学のHPC2500は1ノードあたり512GBの共有メモリを有しているので、メモリの点から言えばまだまだ余裕があると言える。さらに、今回の解析ではCPUを256個しか使用しなかったが、本研究で示した方法によってCPUの数を増やして並列化を行えば、さらに大規模な問題をより速く解くことは十分可能であると考えられる。この様に、現存の計算機においてさえもさらに大規模な解析が可能であり、計算機技術の進歩を考慮すれば、今後提案する手法で扱いうる問題の規模はさらに拡大するものと考えられる。

6. 結論

MPI-OpenMPのハイブリッド並列を高速多重極積分方程式に適用すれば、要素数が1億にのぼるような巨大問題でも現実的な時間で解くことが可能であることが示された。必要メモリ、計算時間の両面から考えても、今後さらに大きな問題を解くことは十分可能である。

また、本論文で行った並列化の手法は、静弾性問題などの境界積分方程式を用いる他の問題に対しても適用できると考えられ、今後、適用範囲を広げて行く予定である。

参考文献

1) 小林昭一 編著 (2000): 波動解析と境界要素法, 京都大学学術出版会
2) N. Nishimura (2002): Fast multipole accelerated boundary integral equation methods, *Appl. Mech. Rev.*, Vol.55, pp.299–324
3) http://www.openmp.org/
4) http://accc.riken.go.jp/HPC/training/text.html
5) K.Yoshida and N.Nishimura (2003): A parallel implementation of the fast multipole boundary integral equation method in elastostatic crack problems in 3D, 計算数理工学コンファレンス論文集, Vol.3, pp.61–66
6) L. Ying, G. Biros D. Zorin and H. Langston (2003): A new parallel kernel -independent fast multipole method, Proc. SC2003, `http://www.sc-conference.org/sc2003/paperpdfs/pap166.pdf`
7) H. Munakata and N. Nishimura (2004): Parallelization of fast multipole accelerated BIEM for SMP computers, 応用力学論文集, Vol.7, pp.287–294
8) 大谷佳広, 西村直志 (2004): 共有メモリー計算機における3次元時間域動弾性高速境界積分方程式法の並列化について, 応用力学論文集, Vol.7, pp.295–304

(2005年4月15日受付)