Vol.2013-HPC-138 No.16
2013/2/22

並列計算機上での境界要素解析を支援するソフトウェアフレームワークの開発

野瀬田 裕樹[1,3]　河合 直聡[1,3]　伊田 明弘[1,3]　岩下 武史[1,3]　中島 浩[1]　美舩 健[1,3]　高橋 康人[2,3]

概要：境界要素法は，偏微分方程式の初期値・境界値問題の代表的な数値解析手法の一つであり，解析対象領域の境界部分のみを離散化の対象とする特徴を持つ．そのため，開領域問題やリメッシングを必要とするような問題の扱いに優れている．本稿では，並列計算機上における境界要素解析を支援するソフトウェアフレームワークの開発に関して述べる．フレームワークとテンプレートから構成されるソフトウェアの全体像とその背景にある概念設計について述べる．また，一般の境界要素解析を対象とした密行列演算によるソフトウェアの実装と本実装における並列化手法に関して述べる．さらに，動作確認のための Cray XE6 システムを用いた数値実験結果について紹介する．

1. はじめに

物理現象や多くの工学的な問題を記述する偏微分方程式の代表的な数値解法として，有限差分法，有限要素法，境界要素法 [1] を挙げることができる．有限差分法や有限要素法が，解析対象とする領域全体を離散化するのに対して，境界要素法は解析対象領域の境界部分のみを離散化するという特徴を持つ．そのため，境界要素法は開領域の扱いに優れ，モデルの離散化に要する人的コストや未知変数の数を低く抑えることができる利点を持つ．しかしながら境界要素法では，離散化後に導出される連立一次方程式の係数が密行列となり，本連立一次方程式の求解に要する計算時間やメモリ量が，係数行列が疎な場合と比較して多大となる問題がある．そこで，実応用解析では，高速多重極法（FMM）[2] や H 行列法 [3] といった高速化技法を活用して境界要素解析を行う場合が多い．しかし，近年の計算機に関する動向を考慮した場合，これらの高速化技法の利用の有無に関わらず，並列・分散並列処理による高速化も重要な技術要素であり，大規模解析では不可欠なものとなってきている．

境界要素解析では，支配方程式から境界積分方程式を導出し，これをモデル境界上で離散化することにより解を求める．従って，支配方程式や境界積分方程式に応じて解析プログラムは異なったものとなり，一般的にはある特定の問題を解くプログラムがそのまま他の問題に適用できるとは限らない．しかしながら境界要素解析に関する一連の処理の中で，離散化された境界要素に関する積分計算を行い，その結果に基づいて連立一次方程式を作成し，解くという処理過程は共通である．そこで，これらの共通部分をフレームワーク化し，それを分散・並列化すれば多様な境界要素解析において利用可能な並列化フレームワークが提供可能であると考えられる．フレームワークを用いることにより，並列計算機上での境界要素解析プログラムの開発コストを軽減することができる．そこで本稿では，並列計算機上での境界要素解析を支援するフレームワークの設計を行い，それを MPI 及び OpenMP により実装した結果について述べることとする．

[1] 京都大学
Kyoto University
[2] 同志社大学
Doshisha University
[3] JST CREST

2. 境界要素解析の手順

2.1 境界要素法による解析例

本小節では，3 次元領域 $\Omega \subset \mathbb{R}^3$ 上における以下のラプラス方程式を例に境界要素解析の典型的な手順について説明する．

$$\nabla^2 \phi(\boldsymbol{r}) = 0 \qquad \text{in } \Omega \tag{1}$$

ここで，$\phi(\boldsymbol{r})$ は点 $\boldsymbol{r} \in \mathbb{R}^3$ におけるポテンシャルの値である．

重み関数 $\psi(\boldsymbol{r})$ と支配方程式の積を領域全体で積分した重み付き残差式

$$\int \psi(\boldsymbol{r}) \nabla^2 \{\phi(\boldsymbol{r})\} \, \mathrm{d}\Omega = 0 \tag{2}$$

に対して，グリーンの定理を適用して以下の等式を得る．

$$\int \phi(\boldsymbol{r})\nabla^2\psi(\boldsymbol{r})\,\mathrm{d}\Omega = \int \left\{\phi(\boldsymbol{r})\frac{\partial\psi(\boldsymbol{r})}{\partial n} - \psi(\boldsymbol{r})\frac{\partial\phi(\boldsymbol{r})}{\partial n}\right\}\mathrm{d}\Gamma \quad (3)$$

ここで，$\frac{\partial\phi(\boldsymbol{r})}{\partial n} = \nabla\phi(\boldsymbol{r})\cdot\boldsymbol{n}$ であり，Γ は領域 Ω の境界を表し，$\boldsymbol{n}$ は境界 Γ に対して垂直な単位法線ベクトルである．ここで重み関数としてラプラス方程式の基本解 $\phi^*(\boldsymbol{r})$ を選ぶ．基本解は点 $\boldsymbol{r}' \in \Omega$ に大きさ 1 のソースがある場合に，点 $\boldsymbol{r}$ で観測されるポテンシャルであり，

$$\nabla^2\phi^*(\boldsymbol{r},\boldsymbol{r}') = -\delta(\boldsymbol{r}-\boldsymbol{r}') \quad (4)$$

の解である．したがって，δ 関数の性質より (3) 式は以下のようになる．

$$C(\boldsymbol{r}')\phi(\boldsymbol{r}') = \int \{\Phi_1(\boldsymbol{r},\boldsymbol{r}') - \Phi_2(\boldsymbol{r},\boldsymbol{r}')\}\,\mathrm{d}\Gamma \quad (5)$$

但し，

$$\Phi_1(\boldsymbol{r},\boldsymbol{r}') = \phi^*(\boldsymbol{r},\boldsymbol{r}')\frac{\partial\phi(\boldsymbol{r})}{\partial n}$$
$$\Phi_2(\boldsymbol{r},\boldsymbol{r}') = \phi(\boldsymbol{r})\frac{\partial\phi^*(\boldsymbol{r},\boldsymbol{r}')}{\partial n}$$

本式は，境界上でポテンシャルとその微係数が分かれば，任意の点 $\boldsymbol{r}'$ におけるポテンシャルが求められることを表している．なお，$C(\boldsymbol{r}')$ は $\boldsymbol{r}'$ が領域内部の場合 1 であり，境界上の場合は境界面の形状から定まる定数である．ラプラス方程式の基本解は 3 次元の場合，

$$\phi^*(\boldsymbol{r},\boldsymbol{r}') = \frac{1}{4\pi|\boldsymbol{r}-\boldsymbol{r}'|} \quad (6)$$

で与えられる．

境界要素法では領域積分を上述のように境界積分に変換するため，離散化は境界について行う．境界 Γ を N_Γ 個の境界要素 Γ_e からなるとし，$\boldsymbol{r}'$ を境界上の点 $\boldsymbol{r}_i$ と取ると (5) 式は以下のように離散化される．

$$C_i\phi_i = \sum_{\Gamma_e}\int \{\Phi_{1i}(\boldsymbol{r}) - \Phi_{2i}(\boldsymbol{r})\}\,\mathrm{d}\Gamma_e \quad (7)$$

但し，

$$\Phi_{1i}(\boldsymbol{r}) = \phi_i^*(\boldsymbol{r})\frac{\partial\phi(\boldsymbol{r})}{\partial n}$$
$$\Phi_{2i}(\boldsymbol{r}) = \phi(\boldsymbol{r})\frac{\partial\phi_i^*(\boldsymbol{r})}{\partial n}$$

また，$C_i = C(\boldsymbol{r}_i)$，$\phi_i = \phi(\boldsymbol{r}_i)$，$\phi_i^*(\boldsymbol{r}) = \phi^*(\boldsymbol{r},\boldsymbol{r}_i)$ である．

次に，境界上に l 個の点を選び，各境界要素内のポテンシャル $\phi(\boldsymbol{r})$ 及びフラックス $q(\boldsymbol{r}) = \frac{\partial\phi(\boldsymbol{r})}{\partial n}$ が，これら l 個の点の一部，$S(\Gamma_e)$ におけるポテンシャル及びフラックスの値から以下のように近似できるものとする．

$$\phi(\boldsymbol{r}) \simeq \sum_{m\in S(\Gamma_e)} N_m(\boldsymbol{r})\phi_m, \boldsymbol{r}\in\Gamma_e \quad (8)$$

$$q(\boldsymbol{r}) \simeq \sum_{m\in S(\Gamma_e)} N_m(\boldsymbol{r})q_m, \boldsymbol{r}\in\Gamma_e \quad (9)$$

ここで，$q_m = q(\boldsymbol{r}_m)$ である．式 (8)(9) より，式 (7) は

$$C_i\phi_i + \sum_{\Gamma_e=1}^{N_\Gamma}\sum_{m\in S(\Gamma_e)} H_{im}\phi_m = \sum_{\Gamma_e=1}^{N_\Gamma}\sum_{m\in S(\Gamma_e)} G_{im}q_m \quad (10)$$

のように書ける．但し，H_{im} および G_{im} は $N_m(\boldsymbol{r})\phi_i^*(\boldsymbol{r})$，$N_m(\boldsymbol{r})q_i^*(\boldsymbol{r})$ を各境界要素上で積分することにより得られる値である．また，$q_i^*(\boldsymbol{r}) = \frac{\partial\phi_i^*(\boldsymbol{r})}{\partial n}$ である．

上記で選んだ l 個の点に関して，式 (10) を連立させることにより，以下のような l 元連立一次方程式が得られる．

$$(H')\,\boldsymbol{\phi} = G\boldsymbol{q} \quad (11)$$

ここで，$\boldsymbol{\phi}$ 及び $\boldsymbol{q}$ は選ばれた l 個の点に関する ϕ_i および q_i を並べたベクトルである．境界条件を考慮することにより，一部の ϕ_i および q_i の値が定まり，式 (11) を解くことが可能となる．得られた境界上のポテンシャル及びフラックスの値を用い，(5) 式から任意の点のポテンシャルを求めることが可能となる．

上記のように，境界要素法の最大の特徴は境界上の離散化のみで近似解が得られることにある．その結果，最終的に解くべき連立一次方程式 (11) の元数は同様の問題を有限要素法により解く場合と比べて少なくて済む．しかし，境界要素解析から生ずる連立一次方程式では係数行列が一般に密となる．本小節で述べた問題の場合，$H_{im}, G_{im} (i,m \in \{1,2,\ldots,l\})$ の値は一般に非零となるため，式 (11) における H', G は密行列となる．

2.2 境界要素法の処理手順

境界要素解析プログラムの処理（プログラム作成のための解析的処理を含む）は以下のように与えられる．まず，支配方程式から境界積分方程式を導出する．その後，境界部分を要素に分割し，境界条件及び初期条件を適用することにより，連立一次方程式に帰着させる．この連立一次方程式を解くことによって，境界要素法による解を得ることができる．図 1 に境界要素解析で行われる処理手順を示す．

境界要素解析では，対象とする物理場により支配方程式や境界積分方程式が異なったものとなり得るが，離散化された境界上で積分計算を行った結果を元に連立一次方程式を作成し，それを解くという処理手順は共通である．そこで，本研究ではこれらの共通部分をフレームワーク化し，それを分散・並列化することにより，多様な境界要素解析において利用可能な並列化フレームワークの開発を試みる．

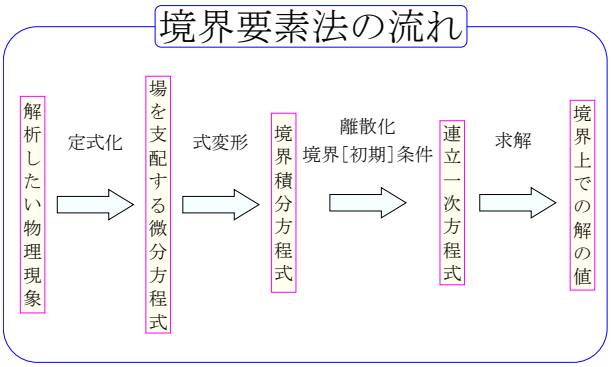

図 1 境界要素法の処理手順

3. 並列境界要素解析フレームワークの設計と実装

3.1 フレームワークの基本設計方針

本研究では，2.2 節で述べた境界要素解析の並列計算機上への実装を支援するフレームワークを開発する．本ソフトウェアの開発においては，MPI によるプロセス並列処理，OpenMP によるスレッド並列処理の両者を活用するハイブリッド並列処理を主に用いる．ハイブリッド並列処理は SMP クラスタ型の並列計算機における主要な並列プログラミングモデルであり，現在のスーパーコンピュータ上で高い実行性能を得るために不可欠な要素である．

上記のフレームワーク設計にあたり，著者らはその基本的な方針として，解析プログラムはフレームワークプログラムにユーザ自らがいくつかの関数（プログラム）を追加することによって与えられるモデルを採用した．具体的には，境界要素解析において，フレームワークはデータ入力，係数行列の作成，連立一次方程式の解法をハイブリッド並列化されたプログラムとして提供する．一方，ユーザは係数行列の各要素の値を返値とする関数プログラムを作成する．即ち，境界要素解析における要素間積分についてはユーザが記述することとなる．また，右辺ベクトルの生成，境界条件の設定，可視化等のための解析結果データの出力についても，フレームワークプログラムを必要に応じてユーザが改変する形で行うものとする．上記の設計モデルを採用した理由を以下に述べる．

境界要素解析では，対象となる物理場を記述する支配方程式に応じて積分の対象となる基本解が異なる．また，同一の対象場（基本解）を用いる場合においてもアプリケーションで要求される解析精度やモデル形状により，要素形状，要素の次数，積分の方式（解析的／数値的）が異なったものとなり得る．従って，汎用的なフレームワークの開発において，これらの多様性の全てに対応することは現実的ではない．そこで，このような多様性にはユーザが提供するプログラムにより対応する上記のモデルを採用した．また，本モデルの採用により，境界要素解析における主要な数理的要素である境界要素積分に関し，ユーザの新しい提案や手法を容易に取り入れることが可能となる．

一方，ユーザの中には境界要素解析のプログラムとしてより完全なものを望み，例えばモデルデータの入力のみで解析結果を得たいというユーザも想定される．そこで，そのようなユーザの要求に応えるために，著者らは典型的ないくつかの問題に対する境界要素解析に対応したテンプレートを提供することとした．テンプレートはある特定の問題領域に対して，境界要素積分の方式を規定した上で提供されるもので，設計上ユーザから提供されるべき境界要素積分に関する関数を含むプログラムである．テンプレートを使用することにより，ユーザはより完全な境界要素解析プログラムを得ることができる．また，ユーザが境界要素積分の関数プログラムを作成する場合，テンプレートを参考例として用いることができる．以上の設計方針をまとめたものが図 2 である．

3.2 密行列演算を用いたフレームワークの実装

著者らの研究グループでは，前小節で述べたフレームワークの実装にあたり，2 種類の実装を行う．一つは一般の境界要素解析に対応する密行列演算に基づいた実装であり，もう一つは H 行列法等の高速行列・ベクトル積演算に基づいた実装である．本稿では実装の完了している前者の実装について述べる．

前小節で述べたように，開発するフレームワークはモデルデータの入力部，係数行列の作成部，連立一次方程式の求解部を主要な構成要素として持つ．そこで，以下ではこれらの構成要素について，その並列化手法を含めて説明する．

3.2.1 モデルデータの入力部

密行列演算を使用するフレームワーク実装では，係数行列の非零要素数は解析モデルの自由度の 2 乗となり，モデルデータの格納に必要なメモリ量は係数行列のそれと比べて十分に小さい．そこで，モデルデータは各プロセスが重複してコピーを持つこととし，マスタプロセスがファイルシステムからモデルデータをロードし，それを MPI_BCAST 関数で各プロセスに分配する方式を採用した．本方式の採用により，ユーザが記述する要素間積分に関するプログラミングがより容易になる．

開発するフレームワークは 3 次元問題を対象とするため，モデルデータは表 1 のようなデータにより構成されるものとする．モデルデータ入力部の実装では，ASCII 形式と NetCDF (Network Common Data Form)[4] 形式の二種類を用意した．NetCDF 形式のファイルは NetCDF ライブラリが存在するすべての言語から呼び出し可能であり，本ライブラリは自己記述的な機種非依存型のデータベースを介してデータを取り扱う．NetCDF ライブラリを用いることで，複数の配列データを移植性の高いバイナリファイルとして容易に読み書きすることが可能となる．

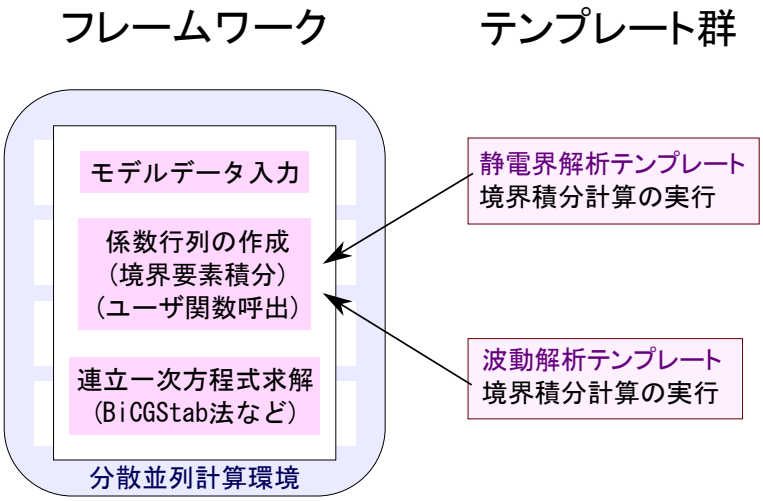

図 2 開発する並列境界要素解析支援ソフトウェアの概念図

表 1 入力モデルデータ

変数・配列名	型	サイズ	データの内容
nond	int	スカラ変数	節点数
nofc	int	スカラ変数	要素数
nond_on_face	int	スカラ変数	各要素を構成する節点数
nint_para_fc	int	スカラ変数	各要素上で定義される int 型パラメータ数
ndble_para_fc	int	スカラ変数	各要素上で定義される double 型パラメータ数
np	double	(3, nond) の配列	全節点の座標値
face2node	double	(nond_on_face, nofc) の配列	各要素を構成する節点の番号
int_para_fc	int	(nofc, nint_para_fc) の配列	各要素上で定義される int 型パラメータ
dble_para_fc	double	(nofc, ndble_para_fc) の配列	各要素上で定義される double 型パラメータ

3.2.2 係数行列の作成部

モデルデータを読み込んだ後，各プロセスは，自身が担当する係数行列の断片（行ブロック）の保持に必要なメモリ領域を確保する．ここで，ハイブリッド並列処理のために，後続のマルチスレッドによる計算を考慮したファーストタッチ [5] を実行する．次に各スレッドから，ユーザが与える係数行列の各要素を生成する関数を並行的に呼び出す．各行列要素の計算は独立であるため，本計算では高い並列化効率が期待できる．

3.2.3 連立一次方程式の求解部

開発フレームワークでは，連立一次方程式の求解部として，主に反復法を用いる．現在，BiCGSTAB 法 [6] と GPBiCG 法 [7] の実装が完了している．これらのクリロフ部分空間反復法の主たる計算核は内積演算と密行列・ベクトル積演算であり，並列化は比較的容易である．本研究では，BLAS による実装とハイブリッド並列処理に基づく自作プログラムによる実装の二種類を行っている．

3.2.4 解析結果の出力部

前節で述べたように，フレームワークには解析結果の出力部は含まない．これは，ファイル出力の対象となる配列や変数が対象とする問題に応じて変化すると考えられるためである．しかし，テンプレートを提供する際には，当該テンプレートの利用目的に合わせて改変されたフレームワークプログラムを提供するため，解析結果の出力部が含まれる場合がある．そこで，このような場合における解析結果の出力部の実装について述べる．

本研究では，解析結果の可視化ソフトウェアとして ParaView[8] の利用を想定する．ParaView (Parallel Visualization Application) は，並列環境のデータ解析と可視化を行うオープンソースのアプリケーションである．同ソフトウェアはデータ処理とレンダリングエンジンの為に Visualization Tool Kit (VTK)[9] ライブラリを用いており，VTK ファイルフォーマットをサポートしている．そこで，本研究では解析結果の出力部において，VTK フォーマットによるファイル出力を行うものとした．

4. 数値実験

4.1 小規模モデルを用いた境界要素解析

4.1.1 テストモデル

図 3 に，解析する問題のモデルを示す．空間に球体状の導体を，地面（電位 0）から球体の中心までの距離が 0.5 m になるように置く．本球状導体の半径は 0.25 m であり，導体の電位が 1 [V] となるように帯電するものとする．この導体の表面に誘起される電荷を境界要素法によって計算

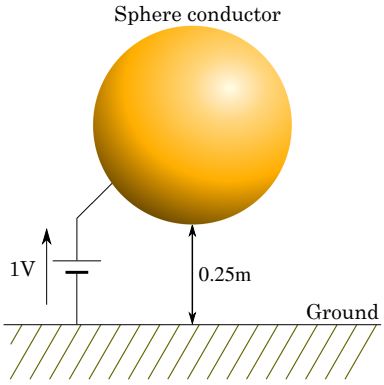

図 3 解析する問題のモデル

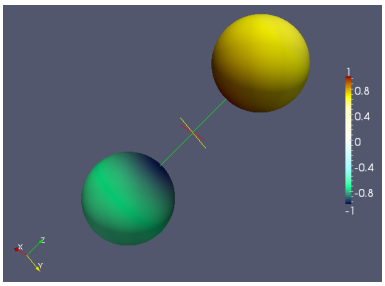

図 4 解析結果

する．これにより空間中の任意の点における電位を求めることができる．本モデルを解析する場合，地面の電位が 0 [V] となるように，地面に関して球状導体と対称な位置に同様の球状導体を電位 −1[V] で配置する．これらの二つの導体球の表面を 21600 個の三角形要素で離散化する．

本解析における支配方程式はラプラス方程式で与えられ，基本解は 2.1 節で述べた (6) 式に一致する．各三角形要素に 1 つの未知変数（自由度）を置き，係数行列の各要素の決定に必要な境界要素積分には文献 [10] の解析的な手法を用いる．なお，本境界要素積分のプログラムは表面電荷法テンプレートとして公開している（http://ppopenhpc.cc.u-tokyo.ac.jp/）．

4.1.2 数値解析結果

前小節の問題を，開発したフレームワークによって並列計算機上で解析した．連立一次方程式の求解ソルバとして，BLAS を用いた実装による BiCGStab 法を使用した．図 4 に，導体球面上に誘起される電荷密度を示す（電荷密度を最大値が 1 となるように規格化して表示）．各導体球において，地面に近い側に，より電荷が誘起されていることが分かる．なお，本解析結果と別途作成したプログラムによる影像電荷法による解析結果は良好に一致しており，本問題においてフレームワークが正しく動作していることが確認された．

4.1.3 並列性能の評価

本フレームワークにおける並列性能を調査するために，京都大学学術情報メディアセンターの並列計算機 Cray XE6 上で使用コア数を変化させ数値実験を行った．Cray XE6

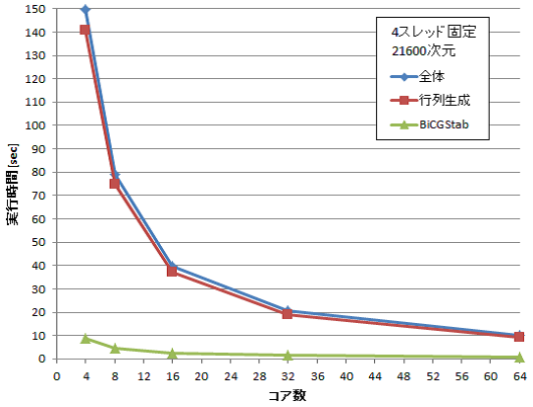

図 5 コア数に対する実行時間の変化 (XE6)

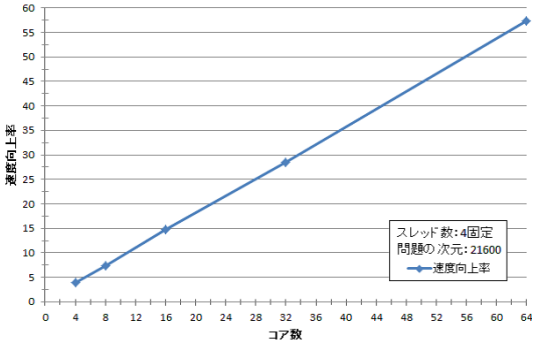

図 6 台数効果 (XE6)

システムは各計算ノードに 16 コアの AMD Opteron プロセッサを 2 基と 64GB（DDR-1600）のメモリを有している．また，各計算ノードは Gemini インターコネクタによる 3 次元トーラス網で結合されている．

プログラム全体の実行時間及び係数行列生成部と BiCGStab 法による連立一次方程式の求解部の実行時間を図 5 に示す．本実験では，プロセスあたりのスレッド数を 4 に固定し，プロセス数を 1 から 16 に増加させ解析を行った．逐次実行の場合における全体の実行時間は 585 秒であり，4 プロセス 4 スレッドのハイブリッド並列処理を行った場合には実行時間が 40 秒に短縮された．次に，図 6 に本数値実験における台数効果を示す．16 プロセス 4 スレッドの場合で約 57 倍の台数効果が得られており，コア数に対して線形に近いほぼ理想的な速度向上が得られていることが分かる．次小節では，より大規模なモデルに対して並列数を増やしたときの実行時間と速度向上率について調査する．

4.2 大規模モデルを用いた境界要素解析

4.2.1 テストモデル

大規模なテストモデルのために，球状導体を 42 個並べたモデルを作成した．前節の問題では導体に固定電位を与える境界条件を用いたが，本問題では鉛直方向に+1 [V/m] の外部電界が与えられる条件を用いた．各導体を 10800 個の三角形要素で離散化し，合計 453600 要素の大規模モデ

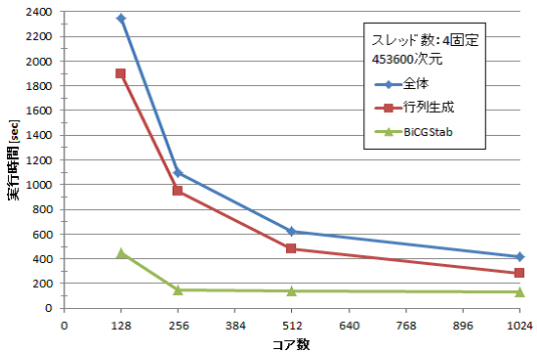

図 7 コア数に対する実行時間の変化

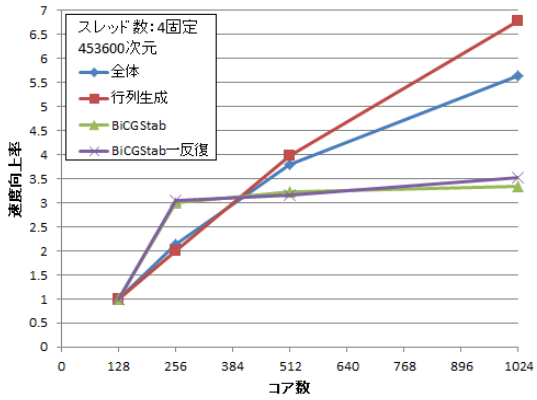

図 8 台数効果

ルを作成した．本問題においても，各要素に 1 つの未知変数（自由度）をおくため，解析の自由度は 453600 となる．

4.2.2 並列性能評価結果

本大規模モデルを計算する際の全体の実行時間と行列生成部及び求解部の実行時間のコア数に対する変化を調べた．図 7 にその結果を示す．同図は，スレッド数を 4 に固定してプロセス数を 32，64，128，256 と変化させたときの行列生成部及び求解部の実行時間の変化を表す．なお，本解析ではメモリ使用量の観点からいずれの場合も 32 ノードを使用し，プロセス数が 32 を超える場合は各ノードに複数のプロセスを動作させて数値実験を行った．また，図 8 に解析全体と各計算部分に関する速度向上率を各々の 32 プロセス，4 スレッド実行時における実行時間を基準として示す．図 8 より，本テストモデルにおいても解析全体としては良好な速度向上が得られていることが分かる．しかしながら，BiCGSTAB 法による求解部では，コア数を 256 以上に増やしても計算時間の短縮が見られず，速度向上率が飽和している．BiCGSTAB 法における主要な計算核は密行列・ベクトル積であり，本計算核では係数行列に関するデータについて再利用性がない．従って，BiCGSTAB 法の実行性能は実効メモリバンド幅に影響されやすい．一方，本実験では使用する計算ノード数はいずれの使用コア数に対しても 32 と固定されている．そこで，使用コア数を 256（ノード内コア数を 8）以上にした場合には，メモリバンド幅に律速され，速度向上率が飽和したと考えられる．

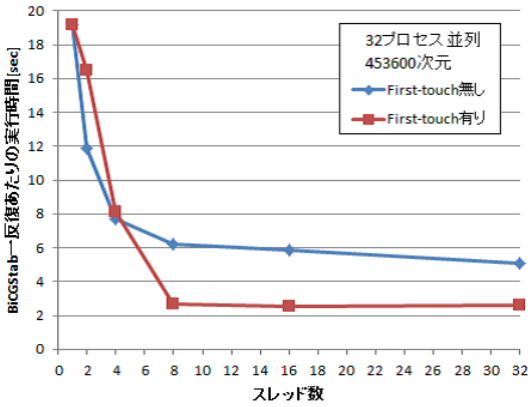

図 9 First-touch の効果

次に，本モデルを用いて First-touch の有無によるスレッド並列性能の変化について調査した．本実験ではプロセス数を 32 に固定し，スレッド数を 1 から 32 に増加させ解析を行った．BiCGStab 一反復あたりの実行時間の変化を図 9 に示す．図 9 によると，本解析ではスレッド数が 8 以上の場合に First-touch の使用が有効となっている．本解析で用いたスレッドのコアに対する割り当て方式では，8 スレッド以上の場合に 2 つのソケットに跨ってスレッドが割り当てられるため，複数ソケット使用時では First-touch が有効となることが分かる．

5. まとめ

本稿では，並列境界要素解析を支援するフレームワークの設計と MPI／OpenMP によるハイブリッド並列処理に基づいた実装に関して述べた．ソフトウェアフレームワークの設計にあたっては，境界要素解析をフレームワークとテンプレートにより支援するモデルを採用した．フレームワークは並列境界要素解析に共通の構成要素であるモデルデータの入力，連立一次方程式の作成とその求解をサポートする．一方，ユーザは係数行列の各要素を与える関数や境界条件の設定，解析結果の出力に関するプログラムを提供する．ただし，特定の問題領域に対してはこれらのユーザが提供するべきプログラムをテンプレートとして提供する．即ち，フレームワークはユーザが提供するプログラムまたはテンプレートと共にコンパイル，実行され，境界要素解析を実現する．

次に，並列計算機 Cray XE6 を用いた数値実験を行った．21600 自由度の問題では，逐次実行において 590 秒であった実行時間が 4 プロセス 4 スレッドでは 40 秒に短縮された．また，台数効果においてもほぼ理想的な効果が得られた．さらに，453600 自由度の問題を解いた場合，32 プロセス 4 スレッド並列時に約 2350 秒であった実行時間が，256 プロセス 4 スレッド並列時では約 420 秒に短縮された．

参考文献

[1] 樫山和男, 牛島省, 西村直志：並列計算法入門, 計算力学レクチャーシリーズ, 丸善株式会社 (2003).
[2] H. Cheng, L. Greengard and V. Rokhlin : A Fast Adaptive Multipole Algorithm in Three Dimensions, J. Comput. Phys., Vol. 155, pp. 468-498 (1999).
[3] L. Grasedyck and W. Hackbusch : Construction and arithmetics of Hmatrices, Computing, vol. 70, pp. 295-334 (2003).
[4] Unidata, "NetCDF (Network Common Data Form)," http:// www.unidata.ucar.edu/software/netcdf/ (accessed 2012-06-24).
[5] W. J. Bolosky, M. L. Scott, R. P. Fitzgerald, R. J. Fowler and A. L. Cox, NUMA policies and their relation to memory architecture, Proc. 4th Int. Conf. Architectural support for programming languages and operating systems, (1991), pp. 212-221.
[6] Henk A. van der Vorst, Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing, vol. 13, pp. 631-644 (1992).
[7] S.L. Zhang, GPBi-CG: Generalized Product-type Methods Based on Bi-CG for Solving Nonsymmetric Linear Systems, SIAM J. Sci. Comput., vol. 18, pp. 537-551 (1997).
[8] Kitware, "ParaView," http://www.paraview.org/ (accessed 2012-06-24).
[9] Kitware, "VTK (Visualization Toolkit)," http://www.vtk.org/ (accessed 2012-06-24).
[10] T. Kuwabara and T. Takeda, "Boundary Element Method Using Analytical Integration for Three-Dimensional Laplace Problem", Electrical Engineering in Japan, Vol. 106, No. 6, pp. 25-31, (1986).