目次

Cover : Designed by TANI Takushi / T&T Design Lab. (September,2010)

全国共同利用版

広報

センターのコンピューティング研究

「並列スクリプト言語Xcryptによるジョブ並列処理の自動化」平石拓、安部達也、三宅洋平、岩下武史、中島浩●「宇宙機周辺プラズマ環境の粒子シミュレーション研究」三宅洋平●「電気機器設計に向けた実規模電磁界解析のための並列計算技術」高橋康人、岩下武史、中島浩●「動弾性時間域境界積分方程式法の計算コスト・メモリコストの削減について」吉川仁

ISSN 1347-3581

京都大学学術情報メディアセンター
Academic Center for Computing and Media Studies, Kyoto University

共同利用・共同研究活動について

京都大学学術情報メディアセンター長
中島　浩

前センター長の美濃導彦教授に替わって、本年４月よりセンター長の重責を担うこととなりました。また就任と同時に「学際大規模情報基盤共同利用・共同研究拠点」が発足し、いわゆる７大学センターと東京工業大学学術国際情報センターからなるネットワーク型拠点の一員として、本センターも新たな共同利用・共同研究の活動を開始しました。

この拠点の主な活動は、スーパーコンピュータをはじめとする大規模な計算資源・情報基盤を利用した公募型共同研究の推進にあり、本年４月に実施した公募の結果、３７件の課題が採択されました。この事業の特徴は、計算資源・情報基盤を介した「学際的」な共同研究、特にComputational Science と Computer Science の両分野の協力・連携による研究活動を推進・支援してくことにあります。したがって本センターがこれまで実施してきた共同研究活動、特にプログラム高度化共同研究の設計思想と軌を一にするものであり、実際にこれまでの「高度化」の実施課題に関する課題が３件採択されています。またこの３課題の他にも本センターとの共同研究を行う課題が４件採択され、その中の２件はセンター教員が有する高度な高性能計算技術の利用を、また他の２件は多数のセンターの資源を有機的な活用を、それぞれ目指したもので、いずれも拠点とその共同研究の趣旨によく合致しています。

また拠点としての活動と並行して、本センター独自のスパコン関連の共同研究事業（若手奨励、大規模計算支援、プログラム高度化）とコンテンツ作成共同研究は、本年度もこれまでと同様に実施しています。次号では、これらの共同研究成果の報告を掲載の予定です。学際的な共同研究がいかに優れた成果を生み出すかをこれらの記事から読み取っていただけると思います。

以上のように拠点の活動は順調にスタートし、本センター独自の活動も着実に進展しています。また神戸に設置される次世代スーパーコンピュータの開発も昨年末の計画修正後は順調に推移し、本センターを含む３５の機関からなる「革新的ハイパフォーマンス・コンピューティング・インフラ」（ＨＰＣＩ）の構築のためのコンソーシアムも設立されました。しかし次世代スーパーコンピュータに関する「事業仕分」に見られたように、スーパーコンピュータや高性能計算に関する政策的取組は必ずしも手厚いものではなく、また国家財政の逼迫に伴って学術・科学技術予算の大幅な縮減も取り沙汰されています。

拠点や本センターの共同研究事業の原資は、もちろん大学運営費交付金や拠点経費などの国家予算によるものであり、また共同利用・共同研究の源泉であるスーパーコンピュータ等の設置費用も大学運営費交付金から支出されています。したがって今後予想される大学関連予算の縮減に対応しつつ、共同利用サービスの質・量を維持し、かつ共同研究事業を継続・発展させていくためには、相当な覚悟での効率化やコスト削減が必須であると認識しています。もちろん本センターではあらゆる手段を講じて共同利用・共同研究の維持発展に努めて参りますが、そのためには利用者・共同研究者である皆様のご支援・ご協力が不可欠であることは言うまでもありません。本センターの活動に対する、より一層のご支援・ご声援をよろしくお願いいたします。

Vol. 9 No.1 号の発刊にあたって

京都大学学術情報メディアセンター
岩下　武史

まず、皆さまにお詫びしなければならないのは通常 6 月末発行としております No.1 号の発刊が大変遅れてしまったことです。大変、申し訳ございませんでした。この主たる原因は、表紙のデザイン変更に伴う議論、遂行に多くの時間を費やしたためです。長い時間をかけて議論した結果である本表紙のデザインコンセプトは京都大学学術情報メディアセンターで行っている全国共同利用サービスの国内での広がりに基づいています。つまり、日本地図上の各円の大きさは各機関における京大センターの全国共同利用サービス利用者の人数を表しています。京都大学のユーザ数が他機関よりもかなり多くなっていますが、全国の様々な機関の方々にご利用いただけていることが分かります（京都大学のユーザ数を表す円は表紙内に収まるように縮小しています）。スーパーコンピュータによるコンピューティングサービスはある程度まとまった計算資源を一度に利用できるという点が一つの特色であり、必ずしもユーザ数のみでそのサービスの良し悪しを判定できるものではありませんが、今後この円がさらに多くの地点に現われ、かつ大きくなるべく努力をしたいと考えております。

さて、今号では、「センターのコンピューティング研究」という特集を組ませていただきました。平成 21 年度において京大学術情報メディアセンター（以下、本センター）のコンピューティング研究部門に所属していた、あるいは主たる研究活動の場としていた 4 名の若手研究者の方々に同部門で行われている研究活動の紹介をしていただきました。三宅氏よりプラズマシミュレーションに関する記事、高橋氏（現同志社大学）より電磁場シミュレーションにおける並列処理に関する記事、吉川氏（現京大院情報学研究科）より動弾性時間域境界積分方程式法に関する記事を御寄稿いただきました。これらの 3 編の記事により、スパコンを利用してどのような解析が可能であるのかその一端をうかがい知ることができるかと思います。この他に、特にスパコンユーザの皆様に注目していただきたいのが、平石氏他による Xcrypt と呼ばれるスクリプト言語に関する記事です。本記事は、本センターが文部科学省、次世代 IT 基盤構築のための研究開発「e-サイエンス実現のためのシステム統合・連携ソフトウェアの研究開発」プロジェクト :「シームレス高生産・高性能プログラミング環境」に基づいて開発している新しい並列スクリプト言語について紹介しています。例えば、ユーザの皆様は、あるプログラムに対して、入力パラメータを様々に変化させてジョブ実行をしたり、スレッド数やプロセス数を変化させて実行したりするということはないでしょうか。本スクリプト言語を使用した場合、このような複数のジョブ実行の管理を非常に簡潔な記述で行うことができるようになります。本スクリプト言語については今後も本広報等によりユーザの皆様に紹介・解説を続けていきますので、ご活用いただければ幸いです。

今号は発刊が遅れてしまいましたが、次号は 12 月末の発行を予定しております。次号ではスパコンおよびコンテンツ作成共同研究の報告を掲載の予定です。本センターの広報をご活用いただき、皆様の研究活動に益するところがあれば幸いです。今後ともどうぞよろしくお願いいたします。

並列スクリプト言語 Xcrypt によるジョブ並列処理の自動化

平石 拓 *，安部 達也 *，三宅 洋平 *，岩下 武史 *，中島 浩*

*学術情報メディアセンター

1　はじめに

京大スパコンを始めとする近年の大規模計算資源を有効活用するためには処理の並列化が必須である．そのために，OpenMP や MPI 等によるプログラムレベルの並列化だけでなく，同一のプログラムに異なるパラメータを与えて同時または逐次に多数実行するようなジョブレベルの並列処理が利用されることも多い．

実際に行われるジョブ並列処理としては，創薬や車体設計等における大規模シミュレーションを行う際のパラメータスイープ・最適パラメータ探索などが挙げられる．また，ソフトウェア静的自動チューニングのために同一のプログラムに対して様々な最適化パラメータを適用してそれぞれの性能を測定したいこともある [3]．

このような粗粒度の並列処理は，C や Fortran などの低水準な言語を用いてプログラムレベルで並列化するより，生産性の高いスクリプト言語等を用いてジョブレベルの並列化を行うほうが適切である．生産性以外にも，NQS [5] や SGE [4]，Torque [1] などのバッチスケジューラが導入された環境では，資源管理や負荷分散処理をそれらに任せることができるという利点もある．また，将来的には計算環境のさらなる大規模並列化が進み，プログラミングレベルでは困難な 1 万－ 100 万レベルの並列化が要求されることも予想される．その際に，プログラムレベル-ジョブレベルの二段階の並列化により計算資源を有効活用することも考えられる．

通常，外部プログラム実行処理の制御はシェルスクリプトや Perl，Ruby などのスクリプト言語で記述されることが多く，上記のようなジョブ並列処理もそのような処理の一種であるため，これらの言語を利用するのが適切であると考えられる．しかし，一般的なスパコン環境においては，プログラムの起動は単純なコマンド実行では行えず，ジョブスクリプトを記述した上でバッチスケジューラに対してジョブを投入しなければならないことが多い．また，多数のジョブを非同期に実行してそれらの完了を待ち合わせたいことも多い．これらの処理を直接 Perl などで記述するのは非常に手間がかかる上，スケジューラとユーザのインターフェースはシステムごとに異なるため，スクリプトの可搬性の面でも問題がある．これらを解決するためには，スクリプト言語の上に，ジョブ並列処理に関する共通の処理に対してライブラリ等によるサポートがあるとよいと考えられる．

そこで我々は，既存のスクリプト言語である Perl をベースとし，これにジョブ並列処理の簡便な記述のために必要となる様々な支援機能を追加したジョブ並列スクリプト言語 Xcrypt を開発している．

Xcrypt では，ジョブをオブジェクトとしてシステムに非依存な記述で定義でき，ジョブの投入は引数をそのオブジェクトとする同期／非同期の手続き呼び出しとして記述することができる．これにより，ジョブの実行を処理全体の一部として扱うことが容易となり，エンドユーザは単純なパラメータスイープから複雑な最適パラメータ探索アルゴズムまで様々な処理を，通常の手続き型のプログラミングとほとんど変わらない手間で記述することができる．Xcrypt ではまた，複雑な探索アルゴリズムやジョブの同時投入数の制限などジョブ並列処理において共通に用いられる様々な機能をライブラリとして提供することができる．これにより，Perl に不慣れなユーザでも，それらのライブラリを取り込むだけで複雑な機能を利用できるようになる．

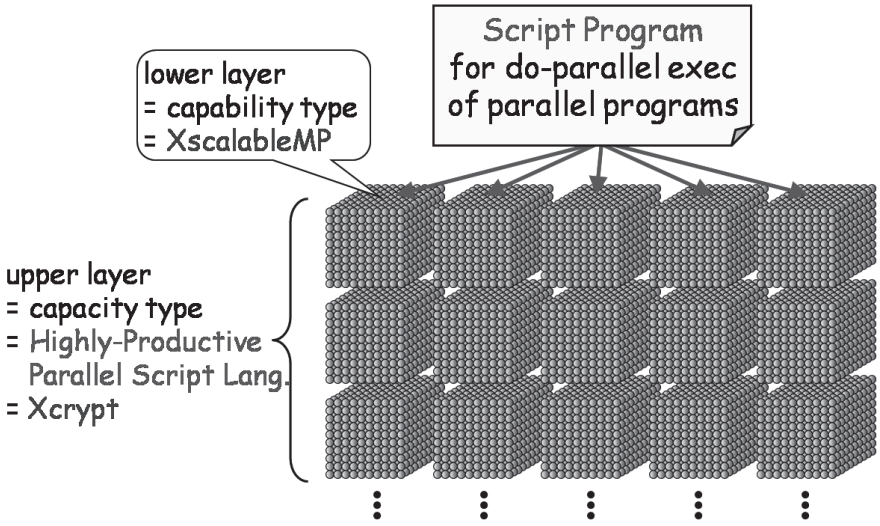


図 1: 二段階の並列化

2 Xcrypt の特徴

Xcrypt の特徴を以下にまとめる．

- 簡便性
  - ジョブの投入を，ジョブスケジューラとのインターフェースにほとんど気を遣うことなく（同期的または非同期的な）手続き呼び出しとして記述することができる．
  - それぞれのパラメータが互いに少しだけ異なるような大量のジョブ（入力ファイルの内容の一部だけが異なるなど）の実行を簡単に書くことができる．
- 可搬性
  - エンドユーザは 1 度書いたスクリプトを（NQS，SGE，Torque などバッチスケジューラ環境が異なる）様々な環境で動かすことができる．
- 柔軟性
  - Perl をベースにした言語であるため，たとえば GUI ベースのワークフローツールに比べて柔軟性が失われていない．ユーザは，単純なパラメータスイープから複雑な探索問題まで様々な処理を記述することが可能である．
- 拡張性
  - 一度にバッチスケジューラに投入するジョブの数を制限したい，プログラムを結果が収束するまで繰り返し実行したいなど，多くのジョブ並列処理において共通して利用したい機能も多い．Xcrypt では，そのような機能を拡張モジュールとして提供するための機構を備えている．エンドユーザは提供されたモジュールを手軽に利用できる．ただし，この拡張モジュール自体は実装するためには，ある程度の Xcrypt の仕様の理解や Perl の知識が要求される．（利用が手軽にできることを重要視している）
- 可用性
  - Xcrypt プロセスはログインノードで動作することを想定しているが，ログインノードにおける長時間のプロセス実行を許さないシステムも多い．そのため，手元のノート PC などで Xcrypt を起動し，そこからスパコンのバッチスケジューラに対してジョブを遠隔投入することもできるようにしている．1 つのスクリプト上から複数のシステムにジョブを投入し，結果をまとめるということも可能である．
  - ジョブの実行履歴は自動的にファイルに保存されるため，Xcrypt プロセスがシステムダウンなどで中断してしまった場合

でも，同じ Xcrypt スクリプトを再実行するだけで，高速に元の実行状態に復帰することができる．

バッチスケジューラ間のスクリプトの可搬性を実現するため，ジョブは Xcrypt におけるジョブオブジェクトとして抽象化し，そこにジョブスクリプトを生成するために必要な情報（実行ファイル名やコマンドライン引数，獲得したいコア数など）を持たせ，ジョブオブジェクトからジョブスクリプトの生成方法の違いをシステムごとの設定ファイルによって吸収する方式を採用した．

また，拡張性を実現するため，Xcrypt は Perl のオブジェクト指向の機能を利用することにした．すなわち，ジョブオブジェクトは Xcrypt が組み込みで提供するジョブ基本クラスのインスタンスであり，プログラマはこの基本クラスを拡張することで拡張モジュールを実装できる．なお，各ジョブは非同期に実行されることを考慮し，ジョブの実行前後に処理されるフックを追加できるようにすることで，柔軟な機能拡張を行えるようにした．

なお，ベース言語に Perl を選択した理由には，入出力ファイルの生成・抽出のための強力な正規表現の機能が利用できること，後述する拡張機能のために必要となるオブジェクト指向の機能を備えること，かつエンドユーザが必ずしもオブジェクト指向を意識したプログラミングをする必要がないこと，などがある．

3 Xcrypt 言語の機能

Xcrypt は Perl をベースとした 言語であり，Perl の持つ機能は基本的に全て利用できる．本章では，Xcrypt が Perl に加えて提供する機能を説明する．

3.1 使用モジュールの宣言

Xcrypt のスクリプトファイルの先頭に，

`use qw(`*module-name*$_1$ *module-name*$_2$ `...core);`

として利用する拡張モジュールを宣言する．ただし，`core` は Xcrypt の基本モジュールであり，必ず指定しなければならない．拡張モジュールについての詳細は，後の 3.4 節で述べる．

3.2 ジョブ制御関連関数

3.2.1 ジョブオブジェクトの生成

ジョブオブジェクトの生成は prepare 関数を用いて，以下のように行う．

`@`*jobs* `= prepare (%`*template*`)`

`%`*template* はジョブの情報をメンバとして持つジョブテンプレートオブジェクト（= Perl のハッシュオブジェクト）である．テンプレートオブジェクトに定義すべき主なメンバを表 1 に示す．生成されたオブジェクトのメンバの値は，基本的にもとのテンプレートオブジェクトから継承される．ここで，`JS_` を名前の接頭辞に持つメンバは，その値が後述するバッチスケジューラ設定ファイルから参照される．

`prepare` は，与えられたテンプレートオブジェクトがメンバ `RANGE`n の値を持っていると，複数のジョブオブジェクトを生成する．このとき，ジョブオブジェクトの `id` はテンプレートオブジェクトの `id` の文字列を接頭辞として連番を付与したものになる．例えば，

`prepare ('id'=>'example', 'RANGE0'=>[1..100])`

の返り値は `example1`–`example100` を `id` にもつ 100 個のジョブオブジェクトの配列となる．`RANGE0`, `RANGE1`, ... にそれぞれ $n_1, n_2, \ldots$ の長さの配列を設定した場合は，$(n_1 \times n_2 \times \ldots)$ 個のジョブオブジェクト列に展開される．

このように生成されたそれぞれのジョブオブジェクトに異なるメンバの値を持たせるには，`arg0@`のように末尾に`@`を付けたメンバ名を用いる．このメンバの値には，関数，配列，文字列のいずれかを設定する．生成された各ジョブのメンバの値は，`@`付きのメンバの値が関数の場合はその戻り値，配列の場合はその配列とジョブ列の各要素を 1 : 1 対応させたときに対応する配列の要素の値，文字列の場合はそれを Perl の `eval` 関数で評価した結果となる．なお，

表 1: テンプレートオブジェクトの主なメンバ

メンバ名 ($0 \leq n \leq 255$)	意味
`id`	ジョブ識別のための文字列（または接頭辞）
`exe`	実行ファイル
`arg`n	コマンドラインオプション
`copiedfile`n	ステージングが必要なファイル
`host`	ジョブを投入する場所
`JS_cpu`	スケジューラに獲得を要求する CPU 数
`JS_node`	スケジューラに獲得を要求する計算ノード数
`JS_queue`	スケジューラのキュー名
`RANGE`n	ジョブオブジェクト配列への展開範囲

関数や eval される文字列のコード中では，RANGEn の配列のうち各ジョブに対応する要素の値を$_[$n$] で参照することができる．RANGE を用いたジョブ生成の例は，後の 5.1 節で示す．

3.2.2 ジョブの投入

ジョブの投入は，以下の式で実行する．

submit (@*jobs*)

@*jobs* は prepare で生成したジョブオブジェクトの配列（または単一のジョブ）であり，配列に含まれる各ジョブオブジェクトに対して 1 つのジョブが投入される．より正確には，submit は各オブジェクトに対応するジョブスレッドを生成する．各スレッドは，以下の処理を行う．

1. use しているモジュールで定義された全ての before メソッドを左に書かれたモジュールのものから全て実行する[1]．
2. use しているモジュールで定義された全ての start メソッドのうち，最も左に書かれたモジュールのものを呼び出す．[2]core モジュールの start メソッドは，ジョブオブジェクトと後述するバッチスケジューラ設定ファイルの情報を元にジョブスクリプトを生成し，そのスクリプトを用いてジョブを投入（qsub）する．
3. 投入したジョブの完了（異常終了を含む）を待つ．
4. use しているモジュールで定義された全ての after メソッドを右に書かれたモジュールのものから全て実行する．

submit 呼び出し後，ジョブスレッドの生成が完了すれば呼び出し元に制御が戻る．ジョブ完了の待ち合わせには次の sync 関数を用いる．

3.2.3 ジョブ完了待ち合わせ

sync (@*jobs*)

により，@*jobs* に含まれるジョブオブジェクトに対応する全てのジョブ（ジョブスレッドの処理）の完了を待ち合わせる．

[1]core には before，after メソッドは定義されていない．

[2]new メソッドと同様，拡張モジュールの定義によっては core モジュールの start が呼ばれない可能性もある．

3.3 遠隔ジョブ投入

デフォルトでは，ジョブの投入は Xcrypt プロセスを実行したシステムに対して行われる（qsub コマンドは Xcrypt を実行したノードにおいて実行される）が，以下のように，遠隔ホストを「ホストオブジェクト」として定義し，環境オブジェクト（への参照）をジョブオブジェクトの host メンバとして設定することで，そのジョブを定義した遠隔ホスト上で投入することができる．

遠隔ホストオブジェクトの定義は以下のように行う．

$*host* = add_host (%*host_hash*)

\%*host_hash* はホストを定義するために必要なパラメータの値をメンバとして持つハッシュオブジェクトである．指定できる主なメンバを表 2 に示す．

add_host の返り値は，生成されたホストオブジェクトであり，これを prepare 関数に渡すテンプレートオブジェクトの host メンバの値にセットしておくことで，submit 関数によるジョブ投入が遠隔ジョブ投入になる．

なお，遠隔ジョブ投入を行うためには，あらかじめ以下の準備を行っておく必要がある．

- ローカルホスト（Xcrypt を起動するホスト），遠隔ホスト双方に Xcrypt をインストールしておく．
- ユーザへの問い合わせなしに OpenSSH でローカルホストから遠隔ホストにログインできるようにしておく．（秘密鍵・公開鍵のペアや SSH Agent を利用する．）
- ジョブの実行に必要なファイル（実行ファイル，入力ファイル）はジョブの投入前に遠隔ホストのファイルシステム上に置いておく（Xcrypt 実行前ではなく，Xcrypt のスクリプト処理でファイルの転送などを行ってもよい．それを簡単に書くためのユーザ関数も提供している）．

表 2: ホストオブジェクトの主なメンバ

メンバ名	意味
name	ユーザ名@ホスト名（例：foo@example.ac.jp）
wd	ジョブを実行するディレクトリ（例:/home/foo/work）

3.4 拡張モジュール

3.1 節の use において core モジュールのみが指定された場合，ジョブオブジェクトは core クラスのオブジェクトとなる．Xcrypt のライブラリ開発者は，この core クラスを拡張することにより機能拡張を行うことができ，この拡張クラス名（＝モジュール名）をエンドユーザが指定することで，ジョブオブジェクトはこの拡張クラスのオブジェクトとなり，そこで追加・拡張されたメソッドやメンバ等が利用可能になる．

拡張モジュールの定義方法は，基本的にはオブジェクト指向 Perl プログラミングにおけるクラス拡張の convention に従う．ただし，new，before，start，after の各名前を持つメソッドは，3.2 節で説明した通り，それぞれ特別な意味を持つ．

new や start メソッドを定義することでジョブ生成・投入時の挙動を変更・拡張することができ，また，before，after メソッドを定義することでジョブ投入前後のフックを追加することができる．

3.5 バッチスケジューラ設定ファイル

Xcrypt は，多様なバッチスケジューラの環境でなるべく同一のスクリプトが動作するように設計されている．しかし，qsub コマンドのパス，qsub がどのようなメッセージを出力するか，あるいはジョブスクリプトの書式などの違いなどへの対応を自動的に行うことは困難である．

そこで Xcrypt では，それらの違いを吸収するため，バッチスケジューラ固有の情報をバッチスケジューラ設定ファイルとして定義する機構を備えている．

例えば，京都大学の T2K オープンスーパーコンピュータに導入されている富士通 NQS 用の設定ファイルは図 2 のように定義される．設定ファイルでは，各コマンドのパスのほか，ジョブスクリプト内で作業ディレクトリのパスを獲得する方法，qsub や qstat の出力メッセージからリクエスト ID を抽出する手段などを文字列・関数として保持する Perl のハッシュオブジェクトとしてスケジューラの情報を定義する．

名前が jobscript_option_*name* のメンバは，投入しようとするジョブのジョブオブジェクトのメンバ JS_*name* の値を参照してジョブスクリプトに埋め込むオプション文字列を生成するためのものであ

```
use config_common;
use File::Spec;

$jsconfig::jobsched_config{"KYOTO"} = {
 # command path
 qsub_command => "/thin/local/bin/qsub",
 qdel_command => "/usr/bin/qdel -K",
 qstat_command => "/thin/local/bin/qstat",
 # options
 jobscript_preamble => ['#!/bin/sh'],
 jobscript_workdir => '$QSUB_WORKDIR',
 jobscript_option_stdout
  => option_with_default('# @$-o ', 'stdout'),
 jobscript_option_stderr
  => option_with_default('# @$-e ', 'stderr'),
 jobscript_option_merge_output
  => boolean_option ('# @$-eo'),
 jobscript_option_node => '# @$-lP ',
 jobscript_option_cpu => '# @$-lp ',
 jobscript_option_memory => '# @$-lm ',
 jobscript_option_limit_time => '# @$-cp ',
 jobscript_option_limit_cputime => '# @$-lT ',
 jobscript_option_queue => '# @$-q ',
 jobscript_option_group => '# @$-g ',
 jobscript_option_stack => '# @$-ls ',
 jobscript_option_verbose
  => boolean_option ('# @$-oi'),
 jobscript_option_verbose_node
  => boolean_option ('# @$-OI'),
 # Extract from output messages
 extract_req_id_from_qsub_output => sub {
     my (@lines) = @_;
     if ($lines[0] =~ /([0-9]*)\.nqs/) {
         return $1;
     } else {
         return -1;
     }
 },
 extract_req_ids_from_qstat_output => sub {
     my (@lines) = @_;
     my @ids = ();
     foreach (@lines) {
         if ($_ =~ /([0-9]+)\.nqs/) {
             push (@ids, $1);
         }
     }
     return @ids;
 },
};
```

図 2: 富士通 NQS 用のバッチスケジューラ設定ファイル

る．生成方法は jobscript_option_*name* の値が文字列か関数かによって異なる．

- 文字列の場合：ジョブオブジェクトのメンバ JS_*name* に値がセットされていれば，その文字列と JS_*name* の値を連結したものをオプション文字列とする．
- 関数の場合：第 1 引数にジョブオブジェクトへの参照，第 2 引数にメンバ名 JS_*name* を与えて，その関数を呼び出したときの返り値をオプション文字列とする．

バッチスケジューラのオプションについての様々な性質を簡潔な記述で書けるようにするため，高階関数を利用することができる．

たとえば，図 2 で，jobscript_option_stdout や jobscript_option_stderr の値として用いられている option_with_default は，以下の動作をする関数を返す高階関数である．

- JS_*name* に文字列がセットされていれば，その文字列を option_with_default の第 1 引数として与えられていた文字列の後ろに連結したものを返り値とする．
- JS_*name* に値がセットされていなければ，option_with_default の第 2 引数の値がデフォルトとして与えられたものとして，上記の処理を行う．

たとえば，ジョブオブジェクトの JS_stdout の値として"output"が与えられた場合，

```
# @$-o output
```

がジョブスクリプトに埋め込まれるが，JS_stdout に値がセットされていなかった場合は，

```
# @$-o stdout
```

が埋め込まれる．

jobscript_option_merge_output などで用いられている boolean_option も高階関数であり，ジョブオブジェクトのメンバ JS_merge_output に真値がセットされていた場合にのみ文字列

```
# @$-eo
```

をジョブスクリプトに埋め込むという挙動を実現している．

4 実装

4.1 概要

Xcrypt システムのほとんどの部分は Perl のモジュール群として実装されている．Xcrypt のインタプリタを起動する xcrypt コマンド自体は，与えられたスクリプトに簡単な前処理を施し，それを perl コマンドに渡して起動するラッパーである．この前処理では，submit 関数などの定義ファイルの取り込みや必要なグローバル変数の定義を行い，ジョブ異常監視スレッドとメッセージ処理スレッドを起動した後でスクリプトの本体を実行するようコードに変更を加える．

上記二つのスレッドは Xcrypt（Perl）プロセスの起動中バックグラウンドで常に存在してジョブの状態を管理し，submit 関数における「ジョブの完了の待ち合わせ」などを実現する．この実装の詳細を次の節で述べる．

4.2 ジョブの状態管理の実装

Xcrypt は，prepare によるジョブオブジェクト生成以降のジョブの状態を (ジョブ ID, 状態名) の組の集合として内部的に管理している．状態名の一覧を表 3，状態遷移図を図 3 に示す．

これらの状態のうち，initialized, submitted, queued, finished, および qsub の失敗による aborted への遷移は，Xcrypt 自身が当該の処理を行ったときにジョブの状態を書き換えることで実現できる．

running と done への遷移については，プログラムの実行を開始・終了したことを計算ノードで実行されているジョブプロセスから通知してもらう必要がある．そこで，core::start メソッドによるジョブスクリプト生成時に

表 3: Xcrypt におけるジョブの状態

状態名	説明
initialized	初期状態（ジョブオブジェクト生成直後）
submitted	qsub によるジョブ投入を行った（成否不明）
queued	qsub コマンドが成功した
running	ジョブスクリプト内でプログラムが実行された
done	ジョブスクリプト内でプログラムが終了した
finished	ジョブスレッドの全ての処理が完了した
aborted	qsub に失敗，またはジョブが異常終了した

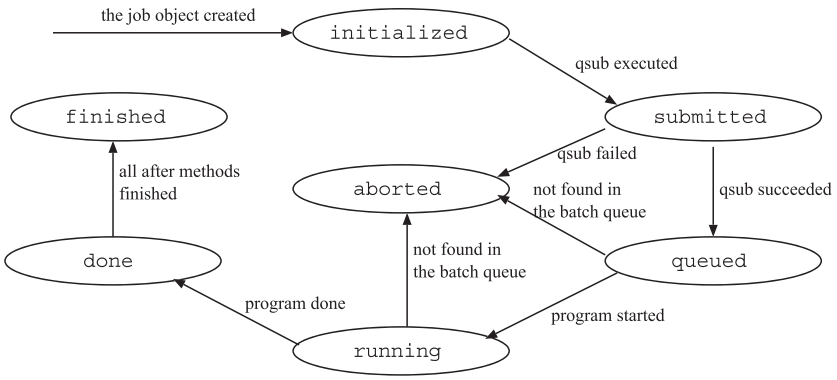


図 3: ジョブの状態遷移

`inventory_write` *hostname port_num* `running`
`./a.out input.dat output.dat`
`inventory_write` *hostname port_num* `done`

のように，プログラムの実行前後にジョブの状態遷移の通知コマンドを実行する処理を書き込むようにしている．この通知を Xcrypt のメッセージ処理スレッドが検知し，ジョブの状態を更新することでこれらの状態への遷移が実現される．ジョブプロセスと Xcrypt プロセスの通信手段は，上記の例では TCP/IP 通信が用いられているが，ファイル（NFS）経由による方法もサポートしている[3]．必要に応じて他の通信方法を実装することも可能である．

また，Xcrypt では「状態が queued または running であるにもかかわらず，その存在が qstat コマンドで確認できない」ジョブを，異常終了したジョブと定義している．ジョブ異常監視スレッドは一定間隔ごとに qstat コマンドを実行してそのような状態のジョブがないかを確認し，もしあればそのジョブの状態を aborted に更新する．

4.3 Xcrypt 中断後の復帰処理

Xcrypt は以下の条件を満たせば，Xcrypt プロセスが実行中に何らかの原因で中断しても，再度同じ Xcrypt スクリプトを実行することで高速に元の実行位置まで復帰できる仕組みを備えている．

- 同一の Xcrypt スクリプトを実行すれば，同一のジョブには同一のジョブ ID が割り当てられて生成，投入される．
- スクリプトが引き起こすファイル操作などの副作用を複数回行っても全体の結果に影響を与えない（ファイルの上書き更新などを行わない）．

この仕組みは，Xcrypt が各ジョブの進行状態（状態遷移の履歴）をファイルに残しておき，再実行時にそれを参照して前回行われた処理の一部をスキップすることで実現している．より具体的には，実行しようとしているジョブと同一のジョブ ID を持つジョブが前回実行されていれば，当該ジョブの前回実行時における最終状態に応じて，既に実行済みであると判断された前処理（`before` メソッド），投入，待ち合わせ，後処理（`after` メソッド）の処理をスキップする．ただし，ジョブオブジェクトの生成（`new` メソッド）は必ず再実行される．ジョブの最終状態と，再実行時にスキップされる処理の対応関係を表 4 に示す．

なお，Xcrypt プロセスがダウンしてジョブだけが実行し続けた場合，前節で説明したジョブスクプリプトから Xcrypt への状態遷移の通知が失敗してしまう．その際は，ジョブが `running` あるいは `done` になったことをファイルに残すことで，Xcrypt の再実行時に当該ジョブの最終状態を適切に判断できる

表 4: ジョブの最終状態と再実行時にスキップされる処理の対応関係（○がスキップされる処理）

状態名	生成	`before`	投入	待ち合わせ	`after`
initialized					
submitted		○			
queued		○	○		
running		○	○		
done		○	○	○	
finished		○	○	○	○
aborted					

[3]現在の実装では，遠隔ジョブ投入を行う場合はファイル経由の通信しかサポートしていない．

```
use base qw(limit core); # limit モジュールを利用
limit::initialize(10);

%template = (
    'id' => 'psweep',          # 任意のジョブ ID
    'RANGE0' => [1..5000],     # 展開範囲
    'exe' => './a.out',        # 実行ファイル名
    'arg0@' => 'input$_[0]',  # 入力ファイル名
    'arg1@' => 'output$_[0]', # 出力ファイル名
    'copiedfile0' => 'a.out',
    'copiedfile1@' => 'input$_[0]',
    'JS_queue' => 'myqueue',  # バッチキュー名
);
# ジョブの準備，投入および終了待ち
# まとめて prepare_submit_sync(%template); でも
可
@jobs = prepare (%template);
submit (@jobs);
sync (@jobs);
```

図 4: Xcrypt によるパラメータスイープの実装

```
package limit;

use strict;
use NEXT;
use Coro::Semaphore;

my $smph;

sub initialize {
  $smph = Coro::Semaphore->new($_);
}

sub new {
    my $class = shift;
    my $self = $class->NEXT::new(@_);
    return bless $self, $class;
}

sub before {$smph->down;}
sub after  {$smph->up;}
```

図 5: limit モジュールの定義

ようにしておく．

5　ジョブ並列処理の例

本章では，Xcrypt 言語によるジョブ並列処理の実装例を示す．

5.1　パラメータスイープ

5.1.1　ユーザスクリプト

同一のプログラムを，連番の入力ファイル・出力ファイルをコマンドライン引数として与えて複数回実行するような処理は Xcrypt で図 4 のように 10 行程度で書ける（この例では limit 拡張モジュールを利用したジョブの同時投入数制限も行っているが，詳細はすぐ後で説明する）．このように，ジョブのテンプレートを定義した後，prepare，submit，sync でジョブの準備，投入，待ち合わせを行うのが Xcrypt スクリプトの最も基本的な形態である．

テンプレートのメンバ RANGE0 に配列 [1..5000] を設定することにより，prepare はそれぞれ psweep1–psweep5000 の id を持つ 5000 個のジョブオブジェクトを生成する．

次に，exe，arg0@，arg1@によりジョブで実行したいプログラムおよびそのコマンドライン引数（ここでは入力・出力ファイル）を指定しているが，入力ファイル名および出力ファイル名はジョブごとに異なるため，@付きのメンバ名の値として Perl に eval される文字列を設定している（$_[0] の部分が RANGE0 の範囲に対応して 1–5000 に置き換わる）．なお，これらの値として

```
sub { "input$_[0]"; }
```

のように関数を与えても同じ結果が得られる．

copiedfile0，copiedfile1@によりプログラムを実行するために必要な（作業ディレクトリにコピーされる）ファイル，queue によりジョブを投入する（バッチスケジューラの）キューの名前をそれぞれ指定している．

5.1.2　拡張モジュールの利用・実装

図 4 のスクリプトは，同時ジョブ投入数制限の機能を提供する拡張モジュール limit を取り込んでいる．limit モジュールは同時に実行するジョブの数を limit::initialize 関数により設定することをそのモジュールの利用者に要請しており，この例では 10 に設定している．

limit モジュールは図 5 のように実装される．このモジュールを読み込むと，各ジョブの投入前に before メソッドにおいてセマフォの獲得が行われ，実行終了後に after メソッドにおいて解放が行われるようになる．limit::initialize で設定した数を超えるジョブを一度に実行しようとすると，セマフォの獲得待ちが発生するため，投入数の制限を実現できる．

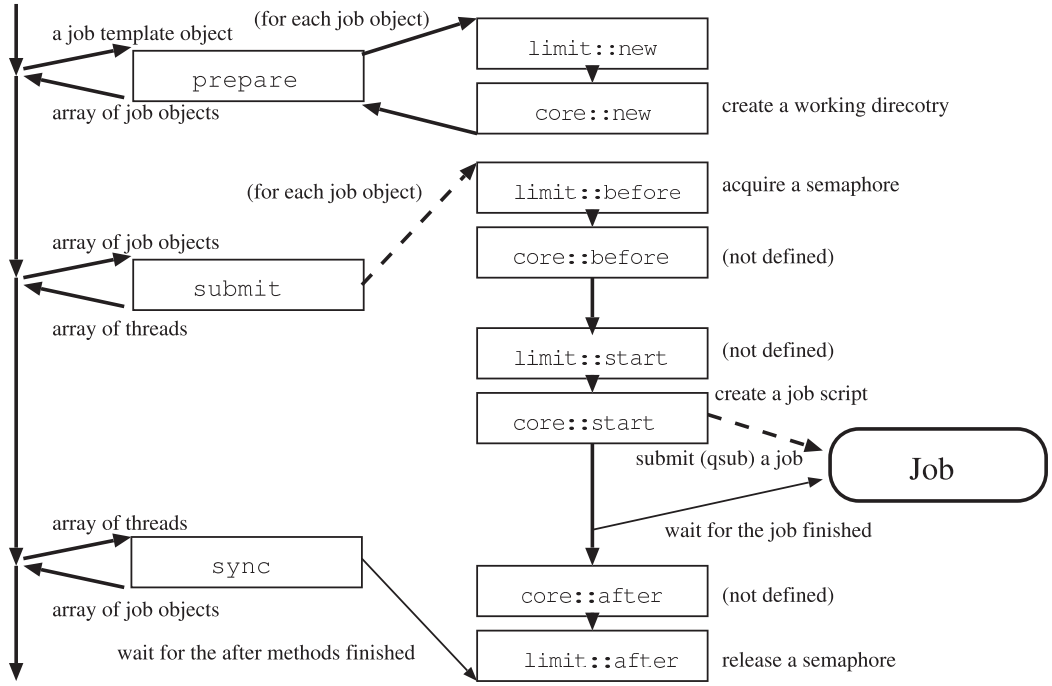


図 6: 図 4 のスクリプトの実行の流れ（破線は非同期実行を表す）

5.1.3 スクリプトの実行の流れ

図 6 に図 4 のスクリプトの実行の流れを示す[4].

prepare 関数では，テンプレートからオブジェクト列への展開後，各オブジェクトに対して limit モジュールの new メソッドを呼び出す．そこから core モジュールの new メソッドが呼び出され，作業ディレクトリの作成等の処理が行われる．

submit 関数が各ジョブオブジェクトに対して生成したジョブスレッドは，投入の前処理として，limit，core の before メソッドをこの順で呼び出した後，core モジュールの start メソッドによりジョブの投入を行う．その後，そのジョブの終了を待ち，最後に後処理として core，limit の after メソッドをこの順で呼び出す．

5.1.4 遠隔ジョブ投入

3.3 節で説明した遠隔ジョブ投入の例として，図 4 のスクリプトを，ジョブが京大スパコンに遠隔投入されるように書き換えたものを図 7 に示す．

```
use base qw(limit core); # limit モジュールを利用
limit::initialize(10);

# 遠隔ジョブ投入先となるホストの定義
my $kyoto = add_host (
 {host => 't99999@thin.kudpc.kyoto-u.ac.jp',
  wd => '/home/t/t99999/work/'}
);

%template = (
    'id' => 'psweep',          # 任意のジョブ ID
    'RANGE0' => [1..5000],     # 展開範囲
    'exe' => './a.out',        # 実行ファイル名
    'arg0@' => 'input$_[0]',   # 入力ファイル名
    'arg1@' => 'output$_[0]', # 出力ファイル名
    'copiedfile0' => 'a.out',
    'copiedfile1@' => 'input$_[0]',
    'host' => $kyoto,          # 遠隔ホスト
    'JS_queue' => 'eh',        # バッチキュー名
);

prepare_submit_sync(%template);
```

図 7: 図 4 の遠隔ジョブ投入版

[4] 3.2.2 節で述べた core モジュールの before，after メソッドのように実際には定義されていないメソッドも，概念的には呼び出し順序に含まれているので図に示している．

エンドユーザは，例えば手元の PC と京大スパコンの両方に Xcrypt をインストールし，京大スパコンの作業ディレクトリに a.out などの必要なファイルを置いておけば，既存のソースコードの先頭へのホストオブジェクトの定義の追加とテンプレートオブジェクトの host メンバの設定のみで，遠隔投入を実行することができる．

5.2 二分法による最適パラメータ探索

図 8 は二分法を用いて $f(x) = 0$ となる x の値を求める Xcrypt スクリプトである．$f(x)$ 1 回の計算がジョブ 1 回の実行に相当する．このスクリプトは 1 つのジョブを投入した後その終了を待ち合わせ結果を解析する，という処理を繰り返すものであり，「ジョブ並列」の処理ではないが，このような逐次処理も Xcrypt が想定する有効な利用法の 1 つである．（二分探索をジョブのプログラム自身で行ってもよいが，ジョブの制限時間がある場合や，プログラムのソースが手元にないなど改変が不可能な場合には Xcrypt による実現が有用である．また，二分探索を別のプログラムに流用できるという利点もある．）

なおこの例は，人工衛星の帯電を防ぐ為に最適なプラズマ放出量を求めるという実例に基づいたものである．x がプラズマの放出量であり，その x に対応する衛星帯電量 $f(x)$ を MPI で並列化されたプラズマシミュレーションプログラム [2] により求めている．スクリプト自体も言語の開発者ではなく，このシミュレーションプログラムの開発者自身（計算科学者）が，インストール作業も含め，付属ドキュメントの情報と最低限の質問対応のサポートのみを受けて書いたものである．

スクリプト中の until の 1 回の反復が $f(x)$ の計算およびその結果の評価に相当する．1 回の反復の処理内容は以下の通りである．まず，prepare 関数でジョブオブジェクト 1 つを生成した後，作業ディレクトリにコピーされた入力ファイル plasma.inp（Fotran プログラムの入力に用いられるネームリスト形式）のパラメータの 1 つを x の値に書き換える．次に，submit_sync（submit と sync を順に呼び出すのと等価）によりジョブを実行し，その出力ファイル result.dat のうち必要な部分（最終行の 3 列目に書かれている数字）を $f(x)$ の値として取り出す．その値の絶対値が許容誤差の範囲内であれば処

```
use base qw(core);
use Data_Generation;
use Data_Extraction;

%template = (
  'id' => 'bisec',
  'exe' => 'mpiexec',
  'cpu' => '1', 'proc' => '64',
  'arg0' => '-n', 'arg1' => '${QSUB_VNODES}',
  'arg2' => './mpikempo3D',
  'arg3' => '4', 'arg4' => '4', 'arg5' => '4',
  'arg6' => '< plasma.inp',
  'copiedfile0' => 'mpikempo3D',
  'copiedfile1' => 'plasma.inp',
  'JS_queue' => 'myqueue',
);

my $wpe1 = 1.0; my $wpe2 = 1.5;
my $phi = 1;

# 絶対値が許容範囲になるまで繰り返し
until (abs($phi) < 0.1) {
  # x の値
  my $wpem = ($wpe1+$wpe2)/2;
  # ジョブの id を更新
  $template{id} = $template{id} . '+';
  # ジョブオブジェクト生成
  my $job = prepare(%template);
  # 入力ファイル書き換え
  my $generator = Data_Generation
      ->new ("$job->{id}/plasma.inp");
  $generator->KeyValueReplace ('wp(3)', $wpem);
  $generator->Execute();
  # ジョブ投入，終了待ち
  submit_sync($job);
  # 出力ファイルから f(x) の値を抽出
  my $extractor = Data_Extraction
      ->new ("$job->{id}/result.dat");
  $extractor->LastLine();
  $extractor->GetColumn(3);
  $phi = $extractor->Execute();
  # f(x) を表示
  print "Potential: ", $phi, "\n";
  # f(x) の正負に応じて区間を狭める
  if ($phi < 0) {
    $wpe1 = $wpem;
  } else {
    $wpe2 = $wpem;
  }
}

print "When wp(3) = ", $wpem, ",\n";
print "phi = ", $phi, ".\n";
```

図 8: Xcrypt による二分法の実装

理を終了し，そうでなければ次の反復処理に移る．

次の反復を行う際，ジョブオブジェクトの生成には前回使用したテンプレートオブジェクトを再利用しているが，id は前回のジョブと重複しないよう別の名前で上書きしている．

この例は，ジョブの入力ファイルの準備，ジョブ投入およびその終了待ち合わせ，出力ファイルの解析の繰り返しというやや複雑な処理を行っているものである．通常のコマンド実行の制御を Perl で記述する場合と比較すると，最初のジョブ定義が必要という違いはあるが，それ以降の処理本体の記述に関してはほとんど差がないことがわかる．(submit_sync の呼び出しは system 関数によるコマンド実行，id の更新は作業ディレクトリ名の更新，prepare の呼び出しは作業ディレクトリの作成の処理にそれぞれ対応付けられる．)

なお，スクリプトで use しているモジュール Data_Generation および Data_Extraction は，Xcrypt が（通常の Perl モジュールとして）提供する入力ファイル生成・出力ファイルライブラリである．これらは，ネームリスト等の計算科学分野でよく登場する入出力ファイルの書式に特化することで sed や grep より高水準なテキスト処理のためのインターフェースを提供している．また，ストリーム処理が行いにくいテキスト処理を率直に実装すると，全ファイルの内容をメモリ上に展開してしまい，数 GB 以上のファイルに対応できなくなってしまうが，これらのライブラリはそのようなことが起こらないよう（かつ利便性も失われないように）バッファを利用するなどして注意深く実装されている．

6　まとめと今後の課題

本論文では，我々が開発中のジョブ並列スクリプト言語 Xcrypt についてその機能，実装，実例を通して紹介した．Xcrypt では，ジョブスクリプトの生成や投入したジョブの状態監視などの処理をシステムに任せることができるため，ジョブの投入や終了待ちを submit と sync という単純な関数呼び出しのインターフェースで扱うことができる．これにより，ジョブの実行を処理全体の一部として扱うようジョブ並列処理を，簡便な手続き型の記述によって実現できる．

今後の予定としては，最適パラメータ探索等に対応した様々なアルゴリズムを Xcrypt のモジュール機構においてモジュール化し，エンドユーザに提供していうことを検討している．また，さらなる実例による検証を重ねることで，ユーザビリティの評価も行う予定である．さらに，現在は Perl ベースとなっているエンドユーザ記述について，Lisp，Ruby，Python など他の軽量言語ベースの記述もできるように，それぞれの言語に対応する処理系も開発していきたい．

A　配布

処理系は http://super.para.media.kyoto-u.ac.jp/xcrypt/ において配布している．筑波大学，東京大学，京都大学の各大学の T2K オープンスパコンですぐに利用できるように各システム用の設定ファイルも同梱している．多くの方に利用いただき，ご意見をいただければ幸いである．

参考文献

[1] Cluster Resources Inc.: TORQUE Resource Manager. http://www.clusterresources.com/pages/products/torque-resource-manager.php.

[2] Miyake, Y. and Usui, H.: New electromagnetic particle simulation code for the analysis of spacecraft-plasma interactions, *Physics of Plasmas*, Vol. 16, No. 6, p. 062904 (2009).

[3] Seymour, K., You, H. and Dongarra, J.: A comparison of search heuristics for empirical code optimization., *CLUSTER*, IEEE, pp. 421–429 (2008).

[4] Sun Microsystems, Inc.: The Grid Engine project. http://gridengine.sunsource.net/.

[5] 富士通株式会社：HPC ミドルウェア Parallelnavi. http://jp.fujitsu.com/solutions/hpc/products/parallelnavi.html.

宇宙機周辺プラズマ環境の粒子シミュレーション研究

三宅 洋平*

*京都大学学術情報メディアセンター

1　はじめに

宇宙空間を飛翔する人工衛星やスペースシャトルなどを総称して「宇宙機」と呼びます。現在、地球周辺の宇宙空間では、種々の人工衛星や国際宇宙ステーションを始めとするたくさんの宇宙機が活躍しています。

実はこれら宇宙機の活動の場となる地球周辺の宇宙空間は真空ではなく、希薄な電離気体であるプラズマという媒質によって満たされています。このプラズマは導電性の媒質であるため、その接触により宇宙機のシステムに帯電などの悪影響を及ぼすことがあります。一方で、宇宙機の推進システムの一つであるイオンエンジンからのイオン噴射や、太陽電池パネル内で生じる電圧は宇宙機周辺のプラズマ環境に影響を与えます。以上の影響は、宇宙環境計測の精度を低下させたり、推進システムの能力を低下させるほか、最悪の場合、宇宙機のシステムそのものに重大な損傷を与えることさえ考えられます。将来の宇宙機の大電力化や高電圧化に備え、宇宙機と宇宙プラズマ環境間の相互作用を、定量的に把握することが大切です。

しかし宇宙プラズマの振る舞いは、その周りの電場と磁場 (以下、電磁場と表記) の状態と密接に関わっており、それを理論的に理解するのは容易ではありません。宇宙機周辺では、これにさらに宇宙機の形状や材質、帯電の状況が絡んでくるため、宇宙プラズマの影響を簡単な数式で評価することは非常に困難となります。また、地上の実験室内で行うチャンバー実験も、現実の宇宙機や宇宙プラズマの空間スケールの再現性に難点があります。

そこで我々は、計算機を使ってプラズマの運動 (および電磁場の変化) を支配する物理方程式を繰り返し解く計算機シミュレーションにより、上記の問題に取り組んでいます。特に近年のスーパーコンピュータの飛躍的な性能向上により、より現実に近い宇宙プラズマ環境を模擬することが可能になってきており、そこから得られた基礎データは今後の宇宙環境利用や人工衛星プロジェクトに役立つことが大いに期待されています。

我々は、宇宙プラズマを膨大な数の荷電粒子として扱うプラズマ粒子シミュレーション [1] という手法を用い、先述の課題に取り組んでいます。また、次世代スーパーコンピュータで計画されるような超並列スカラー計算機に対応できるよう、プラズマ粒子シミュレーションの高効率な並列計算手法に関する研究も行っています。本稿では、プラズマ粒子シミュレーションの基礎原理と応用例、そして次世代スーパーコンピュータに向けて我々が行っている数値手法研究の最近の動向について紹介したいと思います。

2　プラズマ粒子シミュレーション

プラズマは正負の電荷を持つ多数の粒子によって構成されています。これをコンピュータ上で表現する最も直接的な方法は、計算領域内に仮想的な荷電粒子をばらまき、個々の粒子の運動を物理方程式に従って少しずつ解き進めていく、ということになります。この荷電粒子の運動と密接に関わっているのが電磁場です。電磁場が荷電粒子の運動を変化させるのと同時に、変化を受けた荷電粒子の運動により電流が生まれ、今度はその電流が周りの電磁場を変化させます。このような荷電粒子と電磁場の相互作用を時間を追って細かく解き進めていくのがプラズマ粒子シミュレーションの基本と言えます。プラズマ粒子シミュレーションの概念と計算プロセスを図 1 に示します。

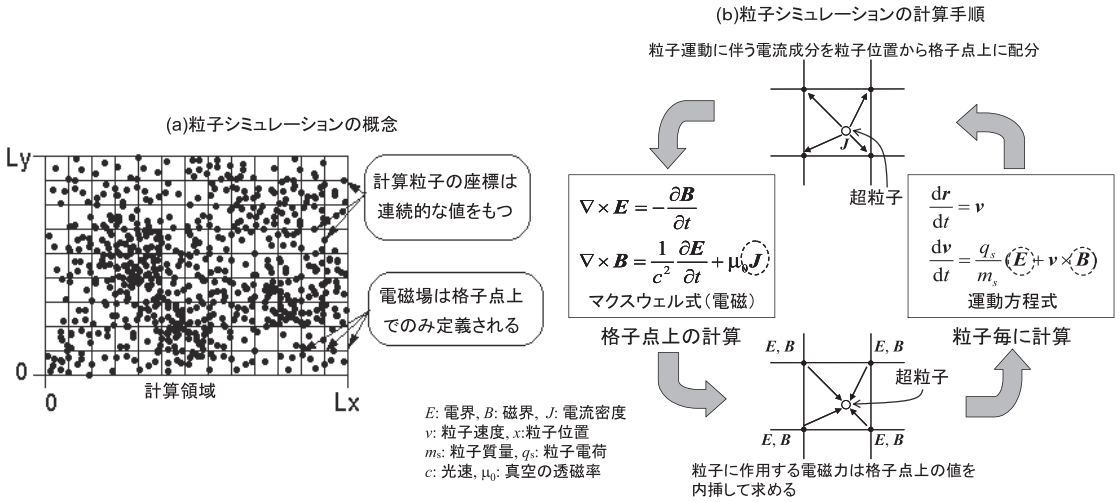


図 1: (a) 粒子シミュレーションの概念図 (実際は 3 次元)、(b) 粒子シミュレーションの基本計算サイクル.

荷電粒子と電磁場に対する基本物理方程式としてはそれぞれ運動方程式、マクスウェル方程式を用います。これらは微分方程式であるため、時間・空間方向の差分化という作業を行い、計算機で解ける形に式を直す必要があります。差分化の詳細は省きますが、図 1(a) に示すように、電磁場の値を計算領域を区切る格子点上に定義することにより、時間空間方向に差分化されたマクスウェル方程式を解くことが可能になります。一方、荷電粒子は時間方向に差分化された運動方程式に従って、計算領域上を自由に動き回ることができます（図 1(a) 内の点）。この際に荷電粒子に作用する電磁力の情報が必要ですが、これは粒子の周りの格子点の電磁場の値を粒子の位置に線形内挿することにより求められます。逆に粒子運動に伴う電流の寄与は、粒子位置から電磁場が定義された格子点上へと、適切な重みをつけて振り分けられます。このように、任意の位置をとることができる荷電粒子と格子点上のみに定義された電磁場、の２つのシステムを組み合わせるところがプラズマ粒子シミュレーションの特徴と言えるでしょう。

以上がプラズマ粒子シミュレーションの基礎となりますが、実際上は上記以外にもいくつかの計算テクニックが必要となります。例えばプラズマ中の宇宙機の解析には、物体としての宇宙機表面の数値的取扱いを考えなくてはなりません。これに関しては、これまで磁場が時間的に変化しないと仮定するモデル（静電モデルという）でいくつかの方法が試みられてきましたが、我々はこの仮定に縛られず磁場の時間変化も含めた電磁粒子シミュレーションに適用可能な宇宙機の数値取扱いを初めて開発しました（図 2）。詳細については、文献 [2] を参照していただきたいと思います。

3　応用例：宇宙プラズマ中電界センサーの数値解析

科学衛星に搭載される電界計測センサーは、宇宙空間中に自然現象として存在する微弱なプラズマ波動現象をできるだけ正確に観測するために高い精度が求められます。しかし宇宙プラズマ中でのセンサーの特性は地上とは違い、宇宙プラズマ環境や衛星本体・アンテナから放射される光電子などに影響を受けるため、地上試験でも正確に評価することは難し

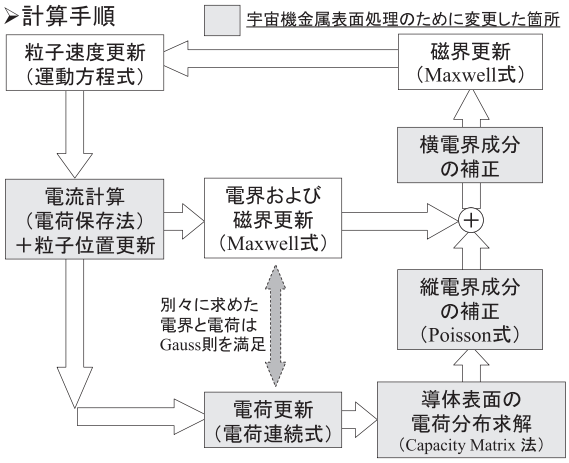


図 2: 宇宙機金属表面の取扱いを含んだ電磁粒子シミュレーションコード EMSES の計算手順.

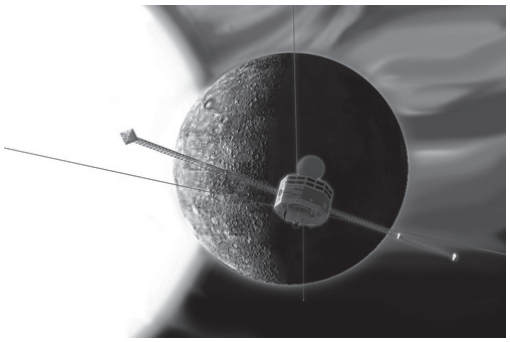


図 3: 水星磁気圏探査衛星 BepiColombo/MMO.
(画像提供：京都大学生存圏研究所)

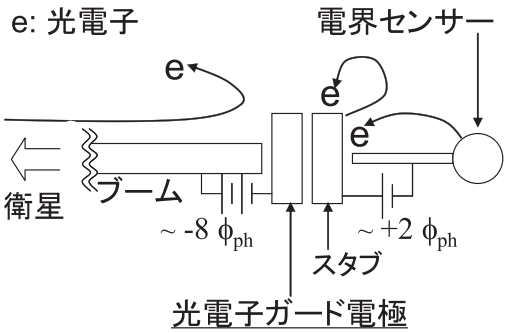


図 4: 電界センサー MEFISTO の数値モデル.

いとされます。そこで、我々は 3 次元のプラズマ粒子シミュレーションを駆使して宇宙プラズマ環境をバーチャルに再現し、その中での電界センサーの特性を評価する研究を行っています。

3.1　電界センサーの数値モデル

本稿では水星探査衛星 BepiColombo/MMO(図 3、2014 年打ち上げ予定) に搭載される電界センサー MEFISTO の解析例を紹介します。図 4 にその構造を示します。MEFISTO の特徴としては、センサーワイヤとブームの間に設置された「パック」と呼ばれる部分に、光電子ガード電極が搭載されていることがあげられます [3]。このガード電極は衛星から放出された光電子をはね返す効果を持っており、光電子がセンサー部分に到達してセンサー性能に影響を及ぼすことを避ける狙いがあります。具体的には、ガード電極が衛星基準電位に対して負の電位を持つように、電極表面に十数 V の電圧を印加します。

図 4 の数値モデルを背景宇宙プラズマで満たした 3 次元計算領域の中央に配置して、電界センサーと衛星に太陽が照射し、光電子が放出される状況をシミュレーションしました。

3.2　シミュレーション結果

図 5 は、衛星本体部分、センサー部分、ガード電極の電位の、シミュレーション開始時からの時間変化を示しています。ただし電位の値は背景のプラズマの電位を基準にしています。時間を追うごとに各部分の電位は変化していき、最終的に衛星本体とガード電極は、それぞれ正と負に帯電し、一方でセンサーはほとんど帯電しないことがわかります。

一般に、人工衛星は電気を帯びたプラズマとの接触により帯電します。今回のシミュレーションでは、光電子放出により衛星に正の電荷が蓄積する効果が最も支配的なため、衛星は正に帯電することになります。このとき衛星の帯電量は、放出された光電子のエネルギーに対応する電位差の数倍となりますが、これは大体数 V となります。ガード電極は衛星に対して十数 V 負となるように電圧をかけているので、結果として電極部分は背景プラズマに対しても負に帯電します。センサー部分に関しては、光電子放出電流の効果を打ち消す様に衛星から人為的に電流を流すということを行います。本シミュレーションでもこの操作を再現しているため、シミュレーション開始後速やかに電位は 0 になります。

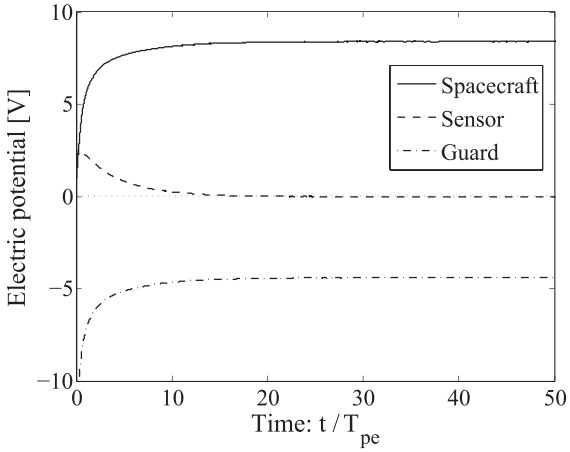


図 5: 衛星 (実線)、電界センサー (破線)、ガード電極 (一点鎖線) の電位の時間変化. $T_{\rm pe}$ は背景プラズマ電子周期を示す. なお、背景プラズマ空間の電位を基準電位とした.

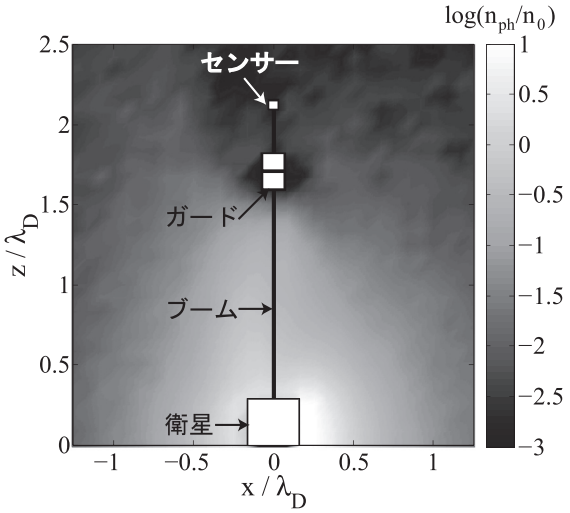


図 6: 衛星片側周辺の衛星起源光電子の密度分布．カラースケールは背景プラズマ電子密度で規格化されている．

さて、このように衛星やガード電極が帯電した状態で、光電子の分布がどのようになっているのか見ていきます。今回は衛星本体から放出された光電子にのみ着目し、分布を調べます。衛星から放出された光電子がセンサー部分に到達し、吸収されることはすなわち、衛星からセンサーへ意図しない電流が流れることを意味するため、センサーの電流電圧特性などに影響を与えるおそれがあります。この意味で衛星起源の光電子の分布を調べることは重要です。

図 6 は衛星起源の光電子の密度分布を示すグレースケールです。太陽光は衛星の右面を照らし、そこから光電子が放出されているものと想定しています。図 6 からわかるように、衛星起源光電子の密度は衛星表面付近が最も高く、衛星から遠ざかるにつれて急激に減少していますが、それでも電界センサーを支えるブームに沿って一部の光電子がブームとガード電極の接続部付近にまで到達しています。しかし負に帯電したガード電極から上では光電子密度が極端に低い領域が形成され、これがセンサー部に到達する衛星起源光電子を大きく減少させていることが読み取れます。ガード電極に印加する電圧を 0 V に設定した別のシミュレーションとの比較により、センサー部周辺の衛星起源光電子密度を 1 桁から 2 桁程度減少させる効果が、ガード電極に確認されました。

以上のようにプラズマ粒子シミュレーションによって、その効果が定量的に実証されていなかったガード電極の性能を具体的に知ることが可能になります。またガード電極の性能は、ガード電圧はもちろんのこと、電極自身のサイズ、形状など様々な要素に依存し、それぞれの性能への影響を解析的に調査することは困難です。しかしシミュレーションを使えば個々の要素をパラメータとしたときの光電子の分布の変化を 3 次元的に把握することができるので、将来のガード電極の最適設計において強力なツールになると期待されます。

4　効率的な並列計算手法の研究

4.1　動的負荷分散手法 OhHelp

粒子シミュレーションでは、プラズマの挙動を正確に再現するために一般に数億から数百億個の莫大な数の粒子を処理しなければなりません。そのためには巨大なメモリが必要となり、パーソナルコンピュータや小規模サーバはもちろん、1 TB 級の大規模な共有メモリ型のスーパーコンピュータであっても、メモリ容量が不足するというケースがあります。そのため、大容量のメモリを比較的容易に利用できる分散メモリシステムを用いたシミュレーションが不可欠です。これを効率的に行うには分散したメモリに粒子をできるだけ均等に割り付けながら、粒子と電磁場の相互作用を効率的に並列計算する負荷分散手法が必要となります。

我々は、各計算プロセスが計算を担当する粒子数と空間領域の大きさを均衡化しながら、かつ粒子と領域内の電磁場の間の相互作用を局所的に並列計算可能な負荷分散方式として OhHelp アルゴリズムを提案しました [4]。本アルゴリズムでは、まず計算領域を均等分割し、各々の部分領域とそれに含まれる粒子を各々のプロセスに割り当てます。これだけですと単純な領域分割法となるのですが、この場合、粒子の空間的な粗密による計算負荷の不均衡が問題となります。そこで OhHelp では一つのプロセスを除く全てのプロセスが本来の担当とは別の部分領域を一つだけ担当し、その領域に含まれる粒子の計算を受け持ちます。これによって粒子数と格子点数の双方において計算負荷を均衡化することが可能になります。

図 7 に上記で述べた部分領域の割り当ての一例を示します。まず、領域を単純に均等分割し、その分割した部分領域を各々のプロセスに「1 次担当領域」

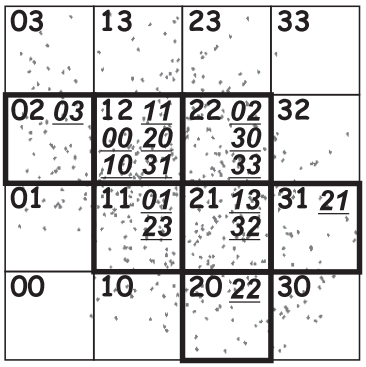


図 7: OhHelp の領域分割．背景の点は粒子の分布を表す．

として割り当てます。図中の (斜体ではない) 通常の字体の数字はプロセス番号かつそのプロセスの 1 次担当領域の番号を示しています。仮に各部分領域内の粒子数が既に均衡していれば、各プロセスはこの「1 次担当領域」内の粒子だけを処理すれば負荷均衡の観点からは十分です。このような状態は 1 次モードでのシミュレーションと呼ばれます。

一方で、図 7 のように、部分領域内の粒子数が不均衡になる場合にはシミュレーションは 2 次モードで行われます。このモードでは一つのプロセス（図では 12 番）を除く全てのプロセスは、平均より多くの粒子が存在している領域を一つだけそれぞれの 1 次担当領域と共に担当することにします。例えば、図 7 において領域 22 は、下線付き斜体で示されるプロセス 02、30、33 の「2 次担当領域」であり、これらのプロセスは領域 22 に存在する粒子の計算を分担します。すなわち粒子数が平均よりも大きな 1 次担当領域を持つプロセスは、その領域を分担するプロセスに粒子の一部を「2 次担当粒子」として委譲することで、自身が担当する粒子数を他のプロセスの担当粒子数と均衡させることができるのです。

OhHelp は、時々刻々と変化する粒子分布を常にモニターし、上記に述べた 1 次/2 次モードの切り替え、あるいは 2 次モードにおける最適な 2 次担当領域の割り当てを動的に行います。なお、2 次担当領域の割り当て方法、粒子均衡/不均衡の判定法、ある領域を 1 次・2 次担当するプロセス間で負荷均衡を保つための粒子移送などに詳細については、文献 [4] を参照していただきたいと思います。

4.2 OhHelp 適用粒子コードの性能評価

ここでは、OhHelp アルゴリズムを適用した粒子シミュレーションを用いて行った性能評価の結果を簡単に紹介します。性能評価には、京都大学の T2K オープンスーパーコンピュータである HX600 クラスタを使用しました。HX600 は各ノードに 4 個の AMD 社製クアッドコア Opteron プロセッサと 32GB(DDR2-667) のメモリを有する共有メモリマシンです。シミュレーションコードは Fortran90 で、また OhHelp ライブラリは C で記述し、それぞれ富士通のコンパイラ (Ver. 3.0) を用い、最適化オプション-Kfast を指定してコンパイルしました。

シミュレーションの内容に関しては、粒子を全空間に均等に分布させた場合と、1 部分領域の中だけに分布させた不均等の場合の 2 通りを行いました。これにより、均等分布では 1 次モードのみの、また不均等分布では 2 次モードのみのシミュレーションが、それぞれ行われることになります。本稿では、プロセス数に比例して全体の領域サイズと粒子数を増加させていく Weak Scaling の結果を示します。

図 8 は電子とイオンの両方を粒子として取り扱う全粒子シミュレーションコードに OhHelp を適用した場合の性能です。性能の単位としては、1 秒間あたりの処理粒子数を用いました。逐次性能は 2.9 Mparticle/s であり、1024 個のプロセッサを用いた場合、不均衡な粒子配置でも逐次性能と比べて 626 倍、均衡な粒子配置では 721 倍の性能を示し、空間領域サイズおよび粒子数の両者においてスケーラブルであることわかります。

図 9 は、粒子シミュレーションのもう一つの代表

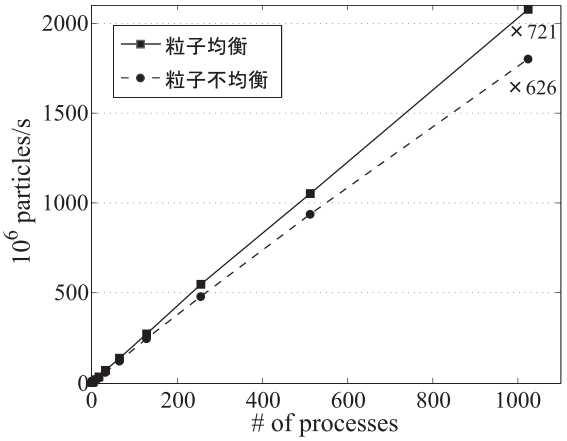


図 8: 全粒子シミュレーションの性能．

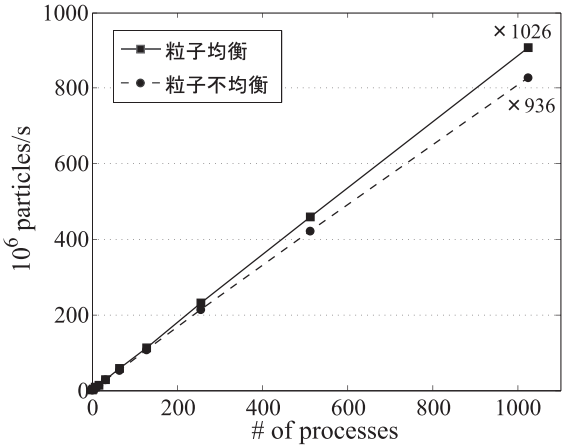


図 9: 粒子・流体ハイブリッドシミュレーションの性能.

的な手法、粒子・流体ハイブリッドコードの場合の性能です。ハイブリッドコードではイオンは粒子として取り扱う一方、電子は流体として扱うため、全粒子シミュレーションに比較して、格子点数に比例する計算の占める割合が大きいことが特徴です。図からは、このケースでも均衡・不均衡の両ケースにおいて 1024 プロセスまで良好な台数効果が得られていることがわかります。

粒子分布が不均衡な場合、均衡なケースに比較して性能が多少下がる原因としては、以下のようなものが考えられます。まず OhHelp では 2 次担当割当の変更回数が頻繁になりすぎないよう、一般に平均よりいくらか多い粒子数 (本性能評価では平均の 120 %) が部分領域に存在しても均衡状態と判断します。この 20 % 分の不均衡が一つの原因です。その他の要因としては、ある部分領域を (1 次もしくは 2 次として) 担当する複数のプロセス間の電流・電荷密度の縮約計算があげられます。しかし、上記の点を考慮しても、全粒子が 1 部分領域に集中しているとする最も厳しいケースで CPU コア数に見合った性能向上を得ることができたという点では、OhHelp の有効性が実証されたと言えるでしょう。

5 おわりに

本稿では、宇宙機周辺のプラズマ環境の数値解析研究の紹介として、プラズマ粒子シミュレーションの基礎、衛星搭載電界センサーへの適用例、そして並列計算機上での動的負荷分散手法である OhHelp について説明しました。基礎原理としては、宇宙機周辺のプラズマ環境を再現できる手法が確立されましたが、現実の宇宙環境を完全に模擬するためには、まだいくつかの課題が残されています。

そのうちの一つは広範にわたる空間スケール長の扱いです。元々宇宙プラズマは電子運動のスケールから磁気圏スケールに至るまで様々な規模の現象が相互に作用し合うことが知られていますが、宇宙機の存在はそれらとはまた異なるスケールの扱いを必要とします。例えば科学衛星に搭載される電界センサーの太さは典型的には 0.1–1 mm ですが、これは宇宙プラズマではミクロなスケール長であるデバイ長 (数 m–数 km) と比較しても非常に小さいものです。このような微細な構造に合わせて格子を切るのは計算コストの観点から現実的ではありません。これを解決するためにはセンサーの周辺のみ局所敵に細かく格子間隔をとるような技術を導入していかなくてはいけません。また衛星表面の物性など、プラズマ以外の部分の数値モデルにもまだ改良の余地が残っています。

以上のような問題が残されていますが、理論や地上実験では正確な評価が難しい宇宙機周辺プラズマ環境解析を定量的に扱える研究手法という意味で、プラズマ粒子シミュレーションは強力なツールになり得ます。特に、高効率な数値計算手法を取り入れ、より現実に近い数値解析を行うことによりこの分野は更なる発展を遂げていくものと期待されます。

参考文献

[1] Birdsall, C. K., and A. B. Langdon, *Plasma Physics via Computer Simulation*, McGraw-Hill, New York, 1985.

[2] Miyake, Y., and H. Usui, New electromagnetic particle simulation code for the analysis of spacecraft-plasma interactions, *Phys. Plasmas*, **16**, 062904, 2009.

[3] Blomberg, L. G. et al., MEFISTO - an electric field instrument for BepiColombo/MMO, *Adv. Space Res.*, **38**, 672–679, 2006.

[4] Nakashima, H., Y. Miyake, Y. Omura, and H. Usui, OhHelp: a scalable domain-decomposing dynamic load balancing for particle-in-cell simulations, *Proc. of the 23rd Intl. Conf. on Supercomputing*, 90–99, 2009.

電気機器設計に向けた実規模電磁界解析のための並列計算技術

高橋康人 ，岩下武史†，中島　浩†

同志社大学理工学部電気工学科
†京都大学学術情報メディアセンター

1　はじめに

社会基盤を支える電磁エネルギー機器の開発では，エネルギーセキュリティ向上や CO_2 排出量削減のために，近年，省エネルギー化への関心がますます高まっている．電気機器の省エネルギー設計を達成するためには，電磁現象を高精度に把握する数値シミュレーションが必須である．計算機の著しい高性能化と電磁界数値解析技術の進歩は，種々の条件下での電界・磁界計算を可能にし，機器設計や物理現象の解明に画期的な進歩をもたらした．しかし，設計対象や解析対象がますます複雑化，大規模化し，計算時間のさらなる高速化，高精度化が要求されている [1]．単体プロセッサの動作周波数はすでに頭打ちになっている現状を考えると，数値解析においてさらなる大規模・高速化を達成する方法の一つとして，並列計算技術の導入は必要不可欠である．そこで本稿では，電磁界数値解析分野における並列計算技術に関する著者らの最近の取り組みを紹介する．

電気機器設計分野では，媒質特性の非線形性・異方性や渦電流を考慮可能であるなど，その汎用性の高さから有限要素法が広く用いられている．電磁界解析における有限要素法の並列化手法として，解析領域をプロセス毎に分割し割り当てる領域分割法 [2, 3] が一般的であるが，要素数があまり多くない場合には並列化効果を達成しにくい難点がある．一方，形状の対称性，および駆動電圧が対称な多相（一般には三相）交流であることから，電気機器における物理現象は時間的な周期性を有している．2 節では，この物理現象の周期性に着目した，有限要素法に基づく時間方向並列化手法について述べる [4]．

次に，強磁性体の磁区構造解析手法として，主に磁気記録の分野で物理現象の解釈やデバイスの設計に用いられているモデリング手法 [5] であるマイクロマグネティックスの並列計算について紹介する．マイクロマグネティックスでは，磁化の定義された要素からの静磁界相互作用（反磁界）を求める必要がある．その演算量は要素数 N の 2 乗に比例し，全計算時間の大部分を占めるため，反磁界計算の演算量削減手法に関する研究が盛んに行われている [6, 7, 8]．3 節では，高速多重極法（FMM）[9, 10, 11] を用いた計算コストが $O(N)$ である新たな反磁界計算方法 [12] の並列化効果について述べる [13]．

最後に，開領域を対象とする静電界解析に適した数値解析手法である表面電荷法（間接型の境界要素法）を取り上げる．電磁界解析分野の主流である有限要素法は演算密度が低く通信量が多いため，並列化効果を得にくい難点がある．一方，境界要素法に代表される積分方程式法は，未知数の個数に比べて演算量が非常に多く，有限要素法と比較して並列化効果を得やすいと考えられる．そこで，高速多重極法を導入した表面電荷法において，100 プロセス以上の並列計算における有効性を示す [14]．

2　回転機の磁界解析における並列化時間周期有限要素法

PWM インバータで駆動されるモータの特性解析など非線形性を考慮した渦電流解析を行うためには，各種時間積分法を用いた時間方向に逐次的な計算が必要となる．各時間ステップにおいては，ニュートン・ラフソン法などにより非線形方程式の求解を行う．非線形反復計算ループは，線形化された連立 1 次方程式を解くための ICCG 法のような反復法のループを含む．したがって，一般的な電磁界解析は，時

間反復－非線形反復－連立 1 次方程式の 3 重ループ構造となる．時定数が大きい問題では誤差成分の減衰が非常に遅いため，定常解を得るまでに多くの時間ステップが必要となり，膨大な計算時間を要する．過渡状態を経ることなく時間領域の定常解を直接求める方法として，時間周期有限要素法 [15, 16] が提案されているが，(未知数の数× 1 周期（または半周期）のステップ数）次元の係数行列を扱わなければならないため，計算コストに難点がある．

そこで本節では，時間周期性を有する電磁界解析の高速化を目的として，時間周期有限要素法から得られる大規模な連立 1 次方程式に MPI を用いた並列計算を適用し，時間周期解を直接算出する．並列化時間周期有限要素法は，空間的な並列化である領域分割による有限要素解析とは異なり，時間軸方向の並列化とみなすことができる．したがって，領域分割による並列有限要素解析と比較すると，2 次元解析のように未知数が少ない問題であっても，すべての時間ステップを連立し問題規模を拡大することで，並列計算における粒度を得やすいと考えられる．また本手法は，時刻ステップごとにプロセスに未知変数を割り当てれば通信パターンが単純となるため，比較的容易に実装できる特長を有する．

2.1 定式化

$A-\phi$ 法により離散化された（回路方程式が連成された）準定常磁場解析の方程式を

$$S(\boldsymbol{x})+C\frac{\partial}{\partial t}\boldsymbol{x}=\boldsymbol{f} \tag{1}$$

とする．ここで，S および C は係数行列，$\boldsymbol{f}$は右辺ベクトル，$\boldsymbol{x}$は磁気ベクトルポテンシャル A と電気スカラポテンシャル ϕ などからなる解ベクトルであり，未知変数の数を m とする．また，$S(\boldsymbol{x})$ は非線形磁気特性の効果などにより S が$\boldsymbol{x}$に対して非線形であることを示す．(1) 式に θ 法を適用し時間方向に離散化を行うと，次式のようになる．

$$\begin{cases} \tilde{T}(\boldsymbol{x}_i)+\tilde{C}(\boldsymbol{x}_{i-1})=\tilde{\boldsymbol{f}}_i \\ \boldsymbol{r}_i=\tilde{\boldsymbol{f}}_i-\tilde{T}(\boldsymbol{x}_i)-\tilde{C}(\boldsymbol{x}_{i-1}) \end{cases} \tag{2}$$

$$\begin{cases} \tilde{T}(\boldsymbol{x}_i)=\theta S(\boldsymbol{x}_i)+\frac{1}{\Delta t}C\boldsymbol{x}_i \\ \tilde{C}(\boldsymbol{x}_i)=(1-\theta)S(\boldsymbol{x}_i)-\frac{1}{\Delta t}C\boldsymbol{x}_i \\ \tilde{\boldsymbol{f}}_i=\theta\boldsymbol{f}_i+(1-\theta)\boldsymbol{f}_{i-1} \end{cases} \tag{3}$$

ここで，下付き添え字は時間ステップ，$\boldsymbol{r}_i$ は時間ステップ i における残差を表す．時間周期有限要素法は，1（半）周期の時間ステップ数を n としたとき，時間周期境界条件 $\boldsymbol{x}_{i+n}=\pm\boldsymbol{x}_i$ を用いて (2) 式を n ステップ連立することで，以下のように得られる [16].

$$\begin{pmatrix} \tilde{T}_1 & O & \dots & O & \pm\tilde{C}_n \\ \tilde{C}_1 & \tilde{T}_2 & \dots & O & O \\ O & \tilde{C}_2 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & O \\ O & O & \dots & \tilde{C}_{n-1} & \tilde{T}_n \end{pmatrix} \begin{Bmatrix} \Delta\boldsymbol{x}_1 \\ \Delta\boldsymbol{x}_2 \\ \vdots \\ \Delta\boldsymbol{x}_n \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_n \end{Bmatrix} \tag{4}$$

反復法として，Localized ILU 前処理付き BiCGstab2 法 [17] を採用する．本稿では，時刻ステップごとにプロセス割り当てを行うものとする．図 1 に，1 周期を 9 ステップに分割し，3 プロセスで並列計算を行った場合の例を示す．このようなプロセス割り当てを行うと，行列ベクトル積演算の際に必要となる通信において，受信はひとつ前のステップを担当しているプロセスからのみ，送信は次の時間ステップを担当しているプロセスへのみとなる．また，図 1 の process 2 での行列ベクトル積を例にとると，以下の手順により通信時間を極力隠蔽できる．

1. process 1 から process 2 に $\Delta\boldsymbol{x}_3$ を非同期送信.
2. $\begin{pmatrix} \tilde{T}_4 & O & O \\ \tilde{C}_4 & \tilde{T}_5 & O \\ O & \tilde{C}_5 & \tilde{T}_6 \end{pmatrix} \begin{Bmatrix} \Delta\boldsymbol{x}_4 \\ \Delta\boldsymbol{x}_5 \\ \Delta\boldsymbol{x}_6 \end{Bmatrix}$ を計算.
3. process 2 が $\Delta\boldsymbol{x}_3$ を非同期受信.
4. $\tilde{C}_3\Delta\boldsymbol{x}_3$ を計算し，加え合わせる.

なお，上記では未知変数 $\Delta\boldsymbol{x}_i$ をすべて通信する必要はなく，行列 $\tilde{C}_i$ が非零要素を持つ部分のみを送受信すればよい．特に後退差分（θ=1）の場合を考えると，渦電流が流れる領域，および回路が連成されている場合にはコイル領域の未知変数に相当する部分のみを通信することになる．

一方，Localized ILU 前処理では，図 1 の各プロセスにおいて色つきの部分のみを考慮して ILU 分解および前進後退代入計算を行う．したがって，前処理に関わる手順において通信は発生しないが，プロセス数の増加に伴い無視する非対角部分の割合も増加するため，一般的に収束性が悪化することが知られている．しかし，本稿で取り扱う時間周期有限要

$$
\begin{bmatrix}
\tilde{T}_1 & O & O & O & O & O & O & O & \tilde{C}_9 \\
\tilde{C}_1 & \tilde{T}_2 & O & O & O & O & O & O & O \\
O & \tilde{C}_2 & \tilde{T}_3 & O & O & O & O & O & O \\
O & O & \tilde{C}_3 & \tilde{T}_4 & O & O & O & O & O \\
O & O & O & \tilde{C}_4 & \tilde{T}_5 & O & O & O & O \\
O & O & O & O & \tilde{C}_5 & \tilde{T}_6 & O & O & O \\
O & O & O & O & O & \tilde{C}_6 & \tilde{T}_7 & O & O \\
O & O & O & O & O & O & \tilde{C}_7 & \tilde{T}_8 & O \\
O & O & O & O & O & O & O & \tilde{C}_8 & \tilde{T}_9
\end{bmatrix}
\begin{Bmatrix}
\Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta x_4 \\ \Delta x_5 \\ \Delta x_6 \\ \Delta x_7 \\ \Delta x_8 \\ \Delta x_9
\end{Bmatrix}
=
\begin{Bmatrix}
r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7 \\ r_8 \\ r_9
\end{Bmatrix}
$$

process1 / process2 / process3

図 1: 時間周期有限要素法の並列化

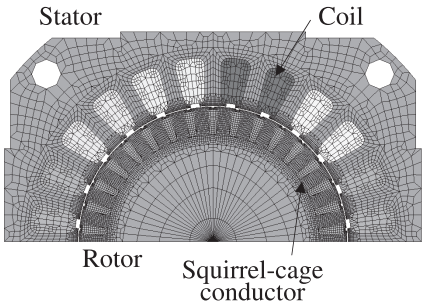


図 2: 誘導電動機モデル

素法は非零要素が対角近辺に分布しており，一般のランダムスパース行列と比較して収束性が極端に悪化しないことが期待できる．非線形方程式求解のためのニュートン・ラフソン法において残差を求める際にも通信が必要になるが，行列ベクトル積の場合と同様，受信は前時間ステップを担当しているプロセスからのみ，送信は次時間ステップを担当しているプロセスへのみとなる．

各プロセスが読み込むメッシュ情報などの入力データはすべて同一であり，トルク計算などのポスト処理やデータの出力はプロセス毎に随時行っていけばよい．したがって，比較的容易にデータの入出力も含めて並列化ができる．

2.2 数値解析による検証

開発手法の並列化効果を検証するために，誘導電動機を対象として 2 次元非線形渦電流解析を実行する．図 2 に，誘導電動機 K モデル [18] のメッシュ分割図を示す．要素数は 13,198，1 ステップあたりの未知変数の数は 12,705 である．本例題では未知変数の数が少なく，領域分割に基づく後退差分近似等を用いた通常の過渡解析ではプロセス数が多い場合に並列計算における十分な粒度が得られないため，並列台数効果を得るのは困難である．1 周期を 256 分割し，すべりを 1 とした．このときの未知変数の数は 3,252,480 である．使用計算機は，京都大学学術情報メディアセンターのスーパーコンピュータ Fujitsu HX600 である．1 ノードあたり Quad Core AMD Opteron 8356 (2.3 GHz) × 4 で，最大 16 ノード（256 コア）を用いて台数効果を検証した．反復法の収束判定値は 10^{-3}，ILU 分解の際の加速係数は 1.1 とした．また，ニュートン・ラフソン法は磁束密度の大きさの修正量 $|\Delta B|$ が全時刻・全要素において 10^{-2}T 以下になった時点で収束とした．

図 3 に，並列化時間周期有限要素法の求解に要した BiCGstab2 法の総反復回数を示す．なお，プロセス数に関わらず，非線形反復回数は 10 回であった．また，図 4 に，プロセス数を変化させた場合におけるデータの入力から出力までの全計算時間と速度向上率を示す．Localized ILU 前処理を用いた場合，プロセス数の増加に伴って前処理において無視する非対角ブロックの割合が増加するため，一般的には BiCGstab2 法の全反復回数も増加する傾向にある．しかし，時間周期有限要素法の非零要素は対角近辺に分布するため，本解析例においては計算時間に顕著な影響を与えるほどの収束性の悪化は見られない．したがって，8 並列を基準として考えるとほぼ理想的に速度が向上している．しかし，プロセス数に対しては半分程度の速度向上率となっている．

有限要素解析における反復法では，疎行列ベクトル積（や前処理）の計算コストが大きい．この疎行列ベクトル積は，メモリからロードするデータ量に比べて演算量が少ないため，計算時間はメモリバンド幅に律速される [19]．したがって，マルチコアプロセッサにおいては，メモリバンド幅が固定されていれば並列化を行っても性能は向上しない．本稿で用いたマルチコアプロセッサでは 2 コア程度でメモリバンド幅を使い切ることになるため，1CPU あたり 2 倍程度の性能向上までしか期待できない．このため，速度向上率は 8 並列で 4 倍程度，16 並列で 8 倍程度となっていると考えられる．

本例題のように未知数の数が小さい場合，領域分割に基づく並列有限要素解析では並列計算の粒度が十分に得られず，並列台数効果はほとんど期待できない．一方，2 次元解析のように規模の小さい問題の場合でも，時間周期有限要素法の定式化では比較的

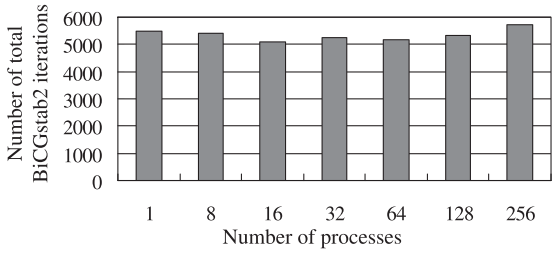


図 3: 並列化時間周期有限要素法における総反復回数

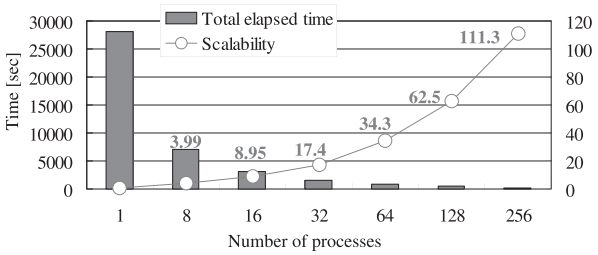


図 4: 時間周期有限要素法の計算時間と速度向上率

大きな係数行列を対象とすることになり，並列計算環境での有効性が期待できる．本手法の詳細や 1 周期の分割数が非常に多い PWM インバータ駆動 IPM モータの解析例などについては，文献 [4] を参照されたい．また，本節とは異なるアプローチとして，時間周期有限要素法と SD-EEC 法 [20, 21, 22] に基づく TP-EEC 法と呼ばれる過渡電磁界解析の収束性改善手法も提案されており，実機解析においても非常に効果的であることが報告されている [23, 24].

3　並列化高速多重極法を用いたマイクロマグネティックス計算

直方体要素分割の周期性を活用した高速多重極法に基づく反磁界計算手法 [12] では，FFT や通常の FMM を用いた従来法と比較して，計算時間・使用メモリの大幅な削減を達成した．しかしながら，マイクロマグネティックスではミクロな物理現象を考慮するために磁壁のサイズ以下の詳細な要素分割が必要となる．したがって，磁気ヘッドなどの実機へ適用する際には解析の大規模化が避けられず，さらなる高速化が要望されている．そこで本節では，大規模マイクロマグネティックス計算のさらなる高速化を目的として，MPI を用いた直方体要素法用 FMM の並列化および負荷分散方法について検討する．

3.1　定式化

マイクロマグネティックスでは，外部磁界によるエネルギー，磁気モーメント同士の相互作用である反磁界によるエネルギー，隣り合うスピン間の相互作用である交換磁気エネルギー，磁気異方性に起因する磁気異方性エネルギーの総和が極小になるよう磁化の分布を決定する．この時，磁化$\boldsymbol{M}$の歳差運動を表す方程式が，以下の Landau-Lifshitz-Gilbert （LLG）方程式である．

$$\begin{aligned}&\left(1+\alpha^2\right)\frac{\partial \boldsymbol{M}}{\partial t}=\\&-\gamma\left(\boldsymbol{M}\times\boldsymbol{H}_{\mathrm{eff}}\right)-\frac{\alpha\gamma}{M_s}\left(\boldsymbol{M}\times\left(\boldsymbol{M}\times\boldsymbol{H}_{\mathrm{eff}}\right)\right)\end{aligned}\tag{5}$$

ここで，M_s は飽和磁化の大きさ，$\boldsymbol{H}_{\mathrm{eff}}$ は実効磁界，α は減衰定数，γ はジャイロ磁気定数である．本稿では，(5) 式の時間微分を前進差分で表し，新たな時刻における単位磁化ベクトル $\boldsymbol{m}$ を求める．また，(5) 式右辺の実効磁界 $\boldsymbol{H}_{\mathrm{eff}}$ は，外部磁界 $\boldsymbol{H}^E$，反磁界 $\boldsymbol{H}^D$，磁気異方性エネルギーに起因した磁界 $\boldsymbol{H}^K$，および交換磁気エネルギーに起因した磁界 $\boldsymbol{H}^A$ の和で表される．ここで，静磁界相互作用である反磁界 $\boldsymbol{H}^D$ は，以下の積分方程式より求める．

$$\begin{aligned}\boldsymbol{H}^D(\boldsymbol{r})=\sum_{l=1}^{N}\Bigg(&\frac{1}{4\pi\mu_0}\int_{\partial V_l}\boldsymbol{M}(\boldsymbol{r}')\cdot\boldsymbol{n}'\frac{(\boldsymbol{r}-\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|^3}\,dS'\\&+\frac{1}{4\pi\mu_0}\int_{V_l}-\left(\nabla'\cdot\boldsymbol{M}(\boldsymbol{r}')\right)\frac{(\boldsymbol{r}-\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|^3}\,dV'\Bigg)\end{aligned}\tag{6}$$

V と ∂V は磁化の定義された直方体要素とその境界，下付き添え字 l は要素番号，μ_0 は真空の透磁率，$\boldsymbol{n}$ は各要素表面の外向き単位法線ベクトル，$\boldsymbol{r}$ および $\boldsymbol{r}'$ は位置ベクトルを表し，∇'および dV', dS' は $\boldsymbol{r}'$ に作用する微分および積分であることを意味する．また，(6) 式の体積積分・境界積分は各直方体要素毎に実行する．本稿では磁化ベクトルを要素内で一定としているため，(6) 式右辺第 2 項は 0 となる．

(5) 式による磁化 $\boldsymbol{M}$ の更新や，$\boldsymbol{H}^E$ と $\boldsymbol{H}^K$ の算出は各要素ごとに行うため，その演算量は $O(N)$ となる．また，$\boldsymbol{H}^A$ の算出では隣り合う要素間の相互作用を計算すればよく，この演算量も $O(N)$ である．

しかし，反磁界 $\boldsymbol{H}^D$ の算出ではすべての直方体要素からの寄与を求める必要があり，演算量は $O(N^2)$ となる．この膨大な演算量を削減するために，高速多重極法を導入する．

3.2 高速多重極法

高速多重極法（FMM）[9, 10, 11] は，多体間相互作用の近似値を必要な精度を確保しつつ高速に計算する手法である．点電荷の集合を含む空間に階層構造を取り入れ，その構造を利用して分割統治計算を進める．その計算過程において，できるだけ多くの点電荷からの影響を多重極・局所展開表現を用いて一括計算を行う．このアルゴリズムの概要について簡単に述べる．解析領域全体を含む立方体をとり，それを根（ルートセル）と呼ぶ．これをレベル 0 のセルとする．その立方体を 8 等分してレベル 1 のセルを作成する．以下同様に，レベル i のセル（親セル）を 8 等分してレベル $i+1$ のセル（子セル）を作成し，設定した最下層までセル分割を繰り返す．セル構造の最下層をリーフと呼ぶ．このセル構造に基づいて，以下の計算を実行する．

1. リーフ中心に多重極展開を定義する（MP）．
2. 子セルで定義された多重極展開を親セルに移動して加え合わせ，これをルートセルまで繰り返す（M2M）．
3. 親セルに局所展開がある場合には，その局所展開を自分に移動し，加え合わせる（L2L）．
4. 遠方セルの多重極展開を自分のセルの局所展開に変換し，加え合わせる（M2L）．
5. リーフセルにおいて，局所展開の寄与（LC）と近傍セル内点電荷の影響を加え合わせる．

M2M，L2L は親子（レベル）間の変換であるのに対し，M2L は同一レベル内遠方セル間での変換となる．セル間の遠近判定はセル構造に基づいて決定される．2 つのセル間に辺または頂点などの共有部分があるセルを近傍，親セルの近傍セルの子セルのうち 2 つのセル間に 1 つのセルが挟まれている状態をお互いに遠方にあるという．M2L 変換は，遠方セルが定義可能なレベル 2 で初めて実行される．よって，M2M 変換はレベル 2 まで行えば十分であり，L2L 変換はレベル 3 から行われる．これらの変換は，多重極・局所展開の展開半径を陽的に与える [26] ことで，すべてのレベルで同一の演算となる．したがって，変換行列をあらかじめ準備・格納しておくことで，各種変換の高速化が可能となる [25]．また，M2M および L2L 変換においては座標軸の回転操作，M2L 部分においてはさらにポテンシャルの指数関数展開を導入し，さらなる高速化を図っている [9, 10, 25]．

FMM を反磁界計算に導入することで，直接計算と比較した場合，使用メモリ・計算時間を削減できる．しかし，時間ステップごとに展開係数を計算し直すためある程度の演算量が必要であり，また近傍要素からの寄与を表す係数行列の格納にも相応のメモリを要する．したがって，磁気ヘッドなど実機を対象とした大規模解析への適用は，従来困難であった．一方，マイクロマグネティックスでは，通常均一な直方体要素分割が用いられる．均一な要素分割の周期性を活用することで，FMM における MP，LC，近傍寄与計算の演算量・使用メモリをさらに削減できる [25, 12]．そこで本稿では，均一な要素分割の周期性を活用し演算量を削減可能な直方体要素用 FMM を採用した．

3.3 直方体要素用高速多重極法の並列化

マイクロマグネティックス計算の並列化を考えた場合，式 (5) による磁化 $\boldsymbol{M}$ の更新や，外部磁界 $\boldsymbol{H}^E$ や磁気異方性エネルギーに起因した磁界 $\boldsymbol{H}^K$ の算出は，各プロセスに割り当てられた要素ごとに独立に実行できる．また，隣り合う要素間の相互作用である交換磁気エネルギーに起因した磁界 $\boldsymbol{H}^A$ の算出では，割り当てられた領域の境界に位置する要素情報について隣接領域を担当するプロセスと通信することになるが，これも容易である．そこで以下では，反磁界 $\boldsymbol{H}^D$ の算出で必要になる直方体要素用 FMM の並列化方法について述べる．基本的には粒子間相互作用を対象とした文献 [27] の方法と同様であるが，直方体要素用 FMM に適合するよう修正を行っている．

まず，セル構造に基づいて全解析領域を部分領域に分割し，部分木単位で各プロセスに割り当てる．このとき，MP，LC 演算，M2M，L2L 変換は各プロセスが記憶しているセルおよび要素のみで実行すればよく，通信を必要としない．一方，M2L 変換においては，各プロセスが担当している領域内に遠方セ

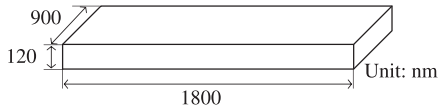


図 5: 磁性体薄膜モデル

ルがあるとは限らないため，該当遠方セルを担当しているプロセスとの通信が必要となる．各セルにおける多重極展開係数は MP または M2M 変換後に定義されるが，M2L 変換が実行されるまで参照されることはない．したがって，各セルにおいて多重極展開係数を求めた後，そのセルの情報を必要とするプロセスに非同期通信を行い，受信は M2L 変換を実行するまでに完了すればよい．この非同期通信により，M2L 変換に必要な通信処理の大部分を極力隠蔽することができる．

近傍寄与計算についても，近傍セルが同一プロセスの担当とは限らないので，通信が必要になる．近傍寄与計算のために通信される近傍セル内要素の磁化情報は，交換磁気エネルギーに起因した磁界 $\boldsymbol{H}^A$ の算出にも活用できる．なお，MP，M2M，M2L，L2L，LC および近傍寄与計算のための変換行列は，すべてのプロセスで記憶しておく．また，近傍・遠方セル以外のセルの情報は使用しないため，その部分は記憶する必要がない．

レベル 2 のセル数（3 次元の場合，最大でも 64）以上のプロセス数で並列計算を実行する際には，上位レベルにおいてプロセス数が過剰となり，一つのセルを複数のプロセスで担当することになる．このような場合，M2M および L2L 変換においても通信が必要になる．

また，負荷分散に関しては，要素の均等化のみに着目するとリーフセルが位置する最下層レベルにおいて領域分割を行い，各プロセスへの割り当てを行えばよい．しかし，一つのセルを複数のプロセスで担当する（子セルと親セルの担当プロセスが異なる）場合が増加し，結果として M2M, L2L 変換のための通信が増加する．M2M, L2L 変換のための通信を極力抑えるためには，なるべく上位レベルでプロセスを割り当てることが望ましい．そこで本稿では，マイクロマグネティックスで用いる均一な直方体要素分割の特徴を考慮した負荷分散方法を採用した．FMM の並列化方法および負荷分散法の詳細は，文献 [13] を参照されたい．

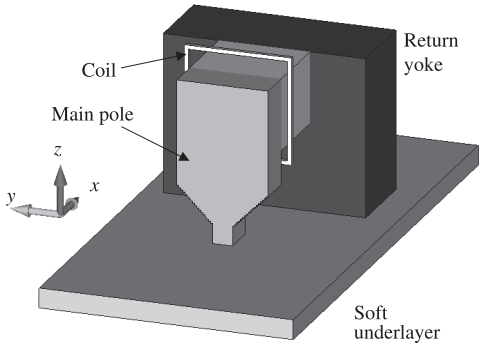


図 6: 垂直記録用単磁極磁気ヘッドモデル

3.4 数値解析による検証

開発手法の有効性を検証するために，単純形状の磁性体薄膜モデル（図 5）および形状が複雑な垂直記録用単磁極磁気ヘッドモデル（図 6）を対象とした大規模マイクロマグネティックス計算を実行した．$5\times5\times5$ nm^3 の立方体要素に分割し，要素数は 12,441,600 および 6,296,400 である．高速多重極法の設定として，実用上十分な精度を得るために多重極・局所展開項数は 10 次とし，指数関数展開においては $s(\epsilon)=18$ の積分公式を用いた [9]．また，リーフセル内の要素数は 27 要素ととした．使用計算機は，京都大学学術情報メディアセンターのスーパーコンピュータ Fujitsu HX600 である．最大 16 ノード（256 コア）を用いて台数効果を検証した．

図 7 に，それぞれの解析モデルにおける速度向上率（台数効果）を示す．図中の破線は，理想的な台数効果（＝プロセス数）を示している．プロセス数の増加に伴い通信の影響が現れるため速度向上率が若干劣化しているが，最も演算量が必要な M2L 変換に伴う通信を極力隠蔽するとともに，各プロセスが担当する要素数を均等化することで，磁性体薄膜モデルおよび SPT ヘッドモデルそれぞれにおいて 256 並列で 193 倍（100 ステップ：367 秒）および 181 倍（100 ステップ：228 秒）という優れた速度向上率を達成しており，形状の複雑さに関わらず開発手法の有効性が確認できる．

4 高速多重極表面電荷法の並列化

近年，密行列を扱う種々の積分方程式法に対し高速多重極法（FMM）が適用され，従来法では困難と

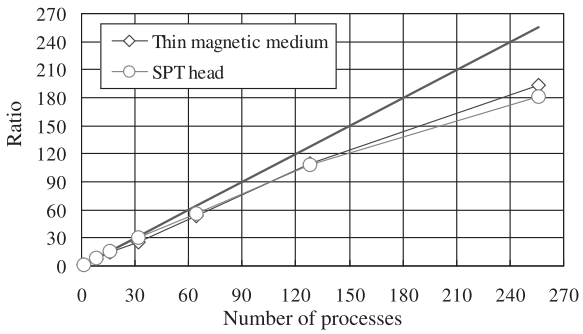


図 7: 大規模マイクロマグネティックス計算における速度向上率

されていたレベルの大規模解析が高速に実行されるようになってきた．しかしながら，積分方程式法のさらなる大規模高速化を達成するためには，FMM への並列計算の導入が必要不可欠である．そこで本節では，積分方程式法として表面電荷法を取り上げ，MPI を用いた高速多重極表面電荷法の並列化について検討する．特徴的な形状を有する解析モデルを例題とし，開発手法の並列化効果を検証する．

4.1 高速多重極表面電荷法

導体表面上を三角形要素に分割し，導体表面電荷密度 σ を定義する．電気スカラポテンシャルを ϕ とすると，以下の積分方程式が成り立つ．

$$\phi(\boldsymbol{r}) = \frac{1}{4\pi\epsilon_0} \sum_{i=1}^{n} \left(\int_{S_i} \frac{\sigma(\boldsymbol{r}')}{|\boldsymbol{r} - \boldsymbol{r}'|} \, dS' \right) \qquad (7)$$

ここで，$\boldsymbol{r}$，$\boldsymbol{r}'$ は位置ベクトル，ϵ_0 は真空中の誘電率，n は要素数である．本論文では，σ は要素内で一定とし，考察点は三角形要素の重心とする．

FMM を導入した表面電荷法の前処理方法として，反復前処理 [28] を採用する．以下に，その概要を述べる．反復解法を用いた連立方程式の求解には，係数行列 A と任意ベクトル $\boldsymbol{b}_{arb}$ の積 $A\boldsymbol{b}_{arb}$ の計算が必要である．その収束性を高めるため，通常は近似行列 A_{appx} を用いた前処理 $A_{appx}^{-1}\boldsymbol{b}_{arb}$ を行う．反復前処理では，$A_{appx}\boldsymbol{x}_{arb} = \boldsymbol{b}_{arb}$ に反復法を適用し，任意ベクトル $\boldsymbol{x}_{arb} = A_{appx}^{-1}\boldsymbol{b}_{arb}$ を求めることで前処理を実行する．つまり，$A\boldsymbol{x} = \boldsymbol{b}$ の解を求めるループ（主反復）の中に前処理部分である $\boldsymbol{x}_{arb} = A_{appx}^{-1}\boldsymbol{b}_{arb}$ を求めるループ（副反復）が存在し，入れ子状の構造となる．副反復は前処理用であるから求める値は近似値でよい．そのため，FMM の計算精度を故意に落とすことで高速化を図ることができる．このとき，A_{appx} は遠方からの寄与も含めた A の大域的な近似行列となるため，良好な収束特性が期待できる．

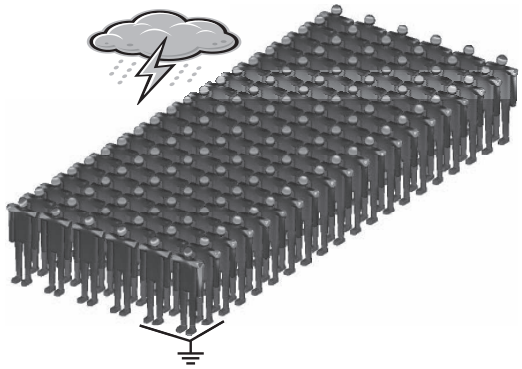


図 8: 大地上の人体形状導体モデル

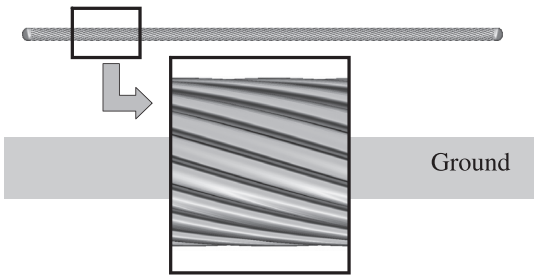


図 9: 高圧送電線モデル

本稿では，主反復には GMRES 法 [17]，副反復には IDR(s) 法 [29] を採用した．主反復および副反復の収束判定値は 10^{-8} および 10^{-5} とした．FMM の設定として，M2M，L2L，M2L 変換に回転変換，対角変換 [9] を導入し，対角変換では $s(\epsilon) = 18$ の積分公式を用いた [9]．多重極・局所展開の展開項数は，主反復ではともに 10 次，副反復ではともに 3 次とした．近傍要素からの寄与を表す係数行列はすべてメモリに格納している．FMM の並列化方法および負荷分散方法は，3 節および文献 [13] と同様とした．

4.2 数値解析による検証

大地上の人体形状導体に一様電界を印加したモデル [30]，および特別高圧送電線を模擬した大地上の裸より線の対地静電容量解析 [31] を対象に，並列化高

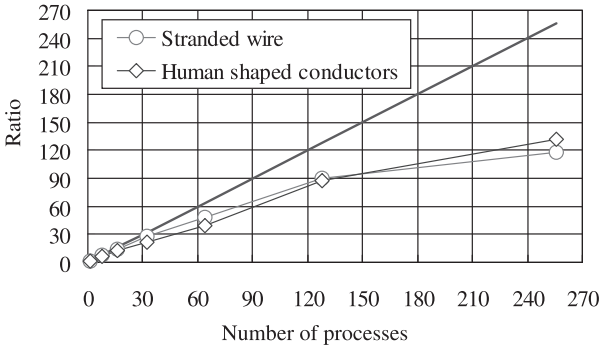


図 10: 高速多重極表面電荷法における速度向上率

速多重極表面電荷法の有効性を検証する．人体形状導体モデルでは解析領域に境界要素が比較的均等に分布しているのに対し，裸より線モデルでは境界要素が局所的に分布する．要素数（未知変数の数）は，それぞれ 2,359,680 および 2,631,464 である．使用計算機は，京都大学学術情報メディアセンターのスーパーコンピュータ Fujitsu HX600 である．図 8 および図 9 に，各モデルにおける表面電荷密度を示す．

図 10 に，プロセス数を変化させたときの各モデルにおける速度向上率を示す．図中の破線は，理想的な台数効果（プロセス数＝台数効果）を示している．プロセス数の増加に伴い，計算量の少ない副反復において通信の影響が顕著に現れるようになり，図 7 と比較すると速度向上率が若干劣化している．しかし，人体形状導体モデルおよび裸より線モデルそれぞれにおいて 256 並列で 117.2 倍（131.4 秒）および 117.6 倍（113.9 秒）という優れた速度向上率を達成しており，要素分布の違いに関わらず開発手法の有効性が確認できる．

5　まとめ

本稿では，電磁界数値解析の大規模高速化を目的とした並列計算技術に関する著者らの最近の取り組みについていくつか紹介した．複雑な物理現象を含む電磁エネルギー機器の高精度な実規模電磁界解析には大規模かつ高速な計算が必須であり，今後，電磁界解析分野においても数値計算の並列化がますます普及することが予想される．特に，電磁エネルギー機器の最適化や，熱，構造，流体等との連成解析などを視野に入れると，一層の計算時間短縮が望まれる．

参考文献

[1] 実規模電磁界解析のための数値計算技術調査専門委員会：「実規模電磁界解析のための数値計算技術」，電気学会技術報告, no. 1129 (2008).

[2] 武居　周，吉村　忍，金山　寛：「階層型領域分割法による高周波電磁場の大規模解析」，電気学会論文誌 A, vol.128, no.9, pp.591-598 (2008).

[3] 中野智仁，河瀬順洋，山口　忠，鵜飼真吾：「回転機のための並列計算の基礎的検討」, 静止器・回転機合同研資，SA-09-27/RM-09-27，(2009).

[4] 高橋康人，岩下武史，徳増　正，藤田真史，若尾真治：「回転機の電磁界解析における並列化時間周期有限要素法の有効性に関する検討」，電気学会静止器・回転機合同研究会資料，SA-09-72/RM-09-78 (2009).

[5] W. F. Brown, Jr.: *Micromagnetics*, Johon Wiley & Sons, Inc., NY (1963).

[6] C. Seberino and H. N. Bertram: Concise, Efficient Three-Dimensional Fast Multipole Method for Micromagnetics, *IEEE Trans. Magn.*, Vol. 37, no. 3, pp. 1078–1086 (2001).

[7] A. Knittel, M. Franchin, G. Bordignon, T. Fischbacher, S. Bending and H. Fangohr : Compression of Boundary Element Matrix in Micromagnetic Simulations, *J. Appl. Phys.*, Vol. 105, 07D542 (2008).

[8] N. Hayashi, K. Saito, and Y. Nakatani: Calculation of Demagnetizaing Field Distribution Based on Fast Fourier Transform of Convolution, *Jpn. J. Appl. Phys.*, Vol. 35, no. 12A, part 1, pp. 6065–6073 (1996).

[9] L. Greengard and V. Rokhlin: A new version of the Fast Multipole Method for the Laplace equation in three dimensions, *Acta Numerica*, Vol. 6, pp. 229–269 (1997).

[10] H. Cheng, L. Greengard, and V. Rokhlin: A Fast Adaptive Multipole Algorithm in Three Dimensions, *J. Comput. Phys.*, Vol. 155, pp. 468–498 (1999).

[11] 小林昭一編：波動解析と境界要素法，京都大学学術出版会 (2000).

[12] Y. Takahashi, S. Wakao, T. Iwashita, and M. Kanazawa: Micromagnetic Simulation by Using Fast Multipole Method Specialized for Uniform Brick Elements, *J. Appl. Phys.*, Vol. 105, 07D514 (2008).

[13] 高橋康人, 岩下武史，中島浩，若尾真治「直方体要素用高速多重極法を用いた大規模マイクロマグネティックス計算の並列化」, 情報処理学会論文誌　コンピューティングシステム, vol. 3, no. 1, pp. 101-111 (2010.3).

[14] 高橋康人，岩下武史，金澤正憲，若尾真治：「高速多重極表面電荷法の並列化に関する基礎検討」，電気学会全国大会, 5-166 (2009).

[15] 原　武久，内藤　督，卯本重郎：「時間周期有限要素法による高圧回転機コロナ・シールド部の電界解析（I 部　数値解析法）」, 電気学会論文誌 B, vol. 102, no. 7, pp. 423-430 (1982).

[16] T. Nakata, N. Takahashi, K. Fujiwara, K. Muramatsu, H. Ohashi, and H. L. Zhu, “Practical Analysis of 3-D Dynamic Nonlinear Magnetic Field Using Time-Periodic Finite Element Method,” IEEE Trans. Magn., vol. 31, no. 3, pp. 1416-1419 (1995)

[17] 藤野清次，張　紹良：「反復法の数理」，朝倉書店 (1996)

[18] 回転機のバーチャルエンジニアリングのための電磁界解析技術調査専門委員会：「回転機のバーチャルエンジニアリングのための電磁界解析技術」，電気学会技術報告第 776 号 (2000)

[19] S. Williams, L. Oliker, R. Vuduc, J. Shalf, K. Yelick, J. Demmel: ”Optimization of Sparse Matrix-Vector Multiplication on Emerging Multicore Platforms,” Proc. SC07 (2009)

[20] T. Iwashita, T. Mifune, and M. Shimasaki ”Similarities between implicit correction multigrid method and A-phi formulation in electromagnetic eld analysis,” *IEEE Trans. Magn.*, vol. 44, pp. 946-949 (2008).

[21] A. Kameari, “Improvement of ICCG Convergence for Thin Elements in Magnetic Field Analysis,” *IEEE Trans. Magn.*, vol. 44, no. 6, pp. 1178-181 (2008).

[22] T. Mifune, S. Moriguchi, T. Iwashita, and M. Shimasaki, ”Convergence Acceleration of Iterative Solvers for the Finite Element Analysis Using the Implicit and Explicit Error Correction Methods,” *IEEE Trans. Magn.*, vol. 45, no. 3, pp. 1104-1107 (2009).

[23] Y. Takahashi, T. Tokumasu, A. Kameari, H. Kaimori, M. Fujita, T. Iwashita, and S. Wakao: ”Convergence Acceleration of Time-Periodic Electromagnetic Field Analysis by Singularity Decomposition-Explicit Error Correction Method,” *IEEE Trans. Magn.*（to be published）.

[24] 高橋康人，徳増　正，藤田真史，若尾真治，岩下武史，金澤正憲：「時間周期有限要素法と EEC 法に基づく非線形過渡電磁場解析における時間積分の収束性改善」，電気学会論文誌，vol. 129-B, no. 6, pp.791-798 (2009).

[25] 濱田昌司，小林哲生：「ボクセルデータ用高速多重極表面電荷法による低周波磁界誘導電界計算」，電気学会論文誌，Vol. 126-A，no. 5，pp. 355–362 (2006)

[26] 濱田昌司，山本　修，小林哲生：「等価多重極モーメント法の拡張と低周波磁界誘導電界計算への応用」，電気学会論文誌，Vol. 125-A，no. 6，pp. 533–543 (2005).

[27] E. J. Lu and D. I. Okunbor: A massively parallel fast multipole algorithm in three dimensions, *in Proc. 5th IEEE Int. Symp. High Performance Distributed Computing*, pp. 40–48 (1996).

[28] S. Hamada and T. Takuma : ”Effective Precondition Technique to Solve a Full Liner System for the Fast Multipole Method,” *IEEE Trans. Magn.*, Vol. 39, No. 3, pp.1666-1669, May (2003)

[29] P. Sonneveld and M. B. van Gijzen, ”IDR(s): a family of simple and fast algorithms for solving large nonsymmetric linear systems,” *Delft University of Technology*, Report 07-07 (2007).

[30] 濱田昌司，宅間　董：「内部制御点可変型三角形パッチを用いた擬似粒子高速多重極表面電荷法」，電気学会論文誌，Vol. 123-A，no. 2，pp. 153–160 (2003).

[31] S. Hamada and O. Yamamoto, ”Numerical Electric Field Calculation around a Bare Stranded Wire,” *Proc. XIIIth ISH*, p.31 (2003).

動弾性時間域境界積分方程式法の計算コスト・メモリコストの削減について

吉川 仁

学術情報メディアセンター

1 研究背景・目的

波動方程式や動弾性方程式などの時間に依存する微分方程式を時間域境界積分方程式法 (時間域 BIEM: Boundary Integral Equation Method)[1] で解く際、基本解を用いることで境界積分方程式が空間と時間の畳み込み積分の形で得られる。時間域 BIEM では境界積分方程式を空間内挿関数、時間内挿関数を用いて離散化し代数方程式

$$\boldsymbol{A}(\Delta t)\boldsymbol{x}(n\Delta t) = \boldsymbol{b}(n\Delta t), \tag{1}$$

を得る。ここで、Δt はタイムステップ幅、$n\Delta t$ は代表時刻、$\boldsymbol{A}$ は影響係数行列、$\boldsymbol{x}$ は未知ベクトル、$\boldsymbol{b}$ は既知ベクトルである。離散化された各代表時刻 $n\Delta t, (n = 1, \cdots, N_t)$ において代数方程式を時間ステップ数 N_t 回解く必要がある。また、各代表時刻において得られる代数方程式の既知ベクトル $\boldsymbol{b}(n\Delta t)$ は、過去の解からの影響の和、つまり影響係数行列 $\boldsymbol{A}(\ell\Delta t), (\ell = 2, n-1)$ と過去の解との行列ベクトル積の和の形で計算される。このため、自由度や時間ステップ数の大きな大規模問題では、代数方程式 (1) の既知ベクトル $\boldsymbol{b}$ を求めるために、膨大な量の影響係数を計算しストアする必要が生じる。

なお、離散化された境界積分方程式の影響係数は、ソースと観測点間の距離と時間差により波動の影響が及ぶ非ゼロ成分のみをストアする。スカラー波動問題では、基本解が時間に関してデルタ関数の形をしており、各代表時刻で求められる影響係数行列は、どの時刻においても同じバンド幅を持つ。3 次元波動方程式の基本解 G は、

$$G(\boldsymbol{x}, \boldsymbol{y}, t) = \frac{\delta(t - \frac{r}{c})}{4\pi r}, \tag{2}$$

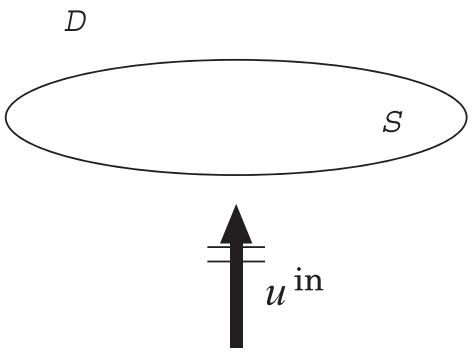


図 1: Scattering problem of plane waves by a crack.

である。ここで、$r = |\boldsymbol{x} - \boldsymbol{y}|$、$c$ は波速である。しかし、3 次元動弾性問題では、基本解の形から明らかなように、ソースから観測点への影響は、P 波の到達と共に表れ S 波と共に通り過ぎる。多くの弾性材料では、P 波は S 波の 2 倍程度の速さで伝播する。つまり、ソースと観測点の時間差が大きい影響係数行列は非ゼロとなる成分が多くなり、計算に時間を要しストアするには多くのメモリを要する。各代表時刻の影響係数行列は、時間差が大きくなるにつれ、非ゼロ成分のバンド幅が広くなる。3 次元動弾性方程式の基本解 $\boldsymbol{\Gamma}$ は等方弾性体の場合、

$$\Gamma_{ij}(\boldsymbol{x}, \boldsymbol{y}, t) = \frac{1}{4\pi\mu}\left[\frac{\delta(t - r/c_T)}{r}\delta_{ij} - {c_T}^2\frac{\partial^2}{\partial y_i \partial y_j}\left(\frac{(t - r/c_T)_+}{r} - \frac{(t - r/c_L)_+}{r}\right)\right], \tag{3}$$

である。ここで、μ はラメ定数、c_L, c_T はそれぞれ P 波速度、S 波速度、$(\ t\ )_+ = tH(t)$ を表す。$H(t)$ は Heaviside 関数である。

図 1 の様な 3 次元無限領域 D に存在する円形クラック S による平面波入射波 $\boldsymbol{u}^{\mathrm{in}}$ の散乱問題を、スカラー

波動問題と動弾性問題として解いた時に必要となる各代表時刻における影響係数行列 ($\boldsymbol{A}(\Delta t)$, $\boldsymbol{A}(5\Delta t)$, $\boldsymbol{A}(10\Delta t)$) の非ゼロ成分を、それぞれ図 2 と図 3 に示す。図 3 から、動弾性時間域 BIEM で必要となる影響係数行列のバンド幅が時間と共に広がっているのが見て取れる。

この問題を解決するために、Walker ら [2, 3, 4] が cast forward と呼ばれる手法を提案し、影響係数行列と過去の解との行列ベクトル積を行うタイミングを工夫し、影響係数 $\boldsymbol{A}(\ell), (\ell = \frac{N_t+1}{2}+1, \cdots, N_t)$ のストアを不要とした。著者ら [5, 6] は、cast forward を動弾性問題に適用しメモリ使用量を削減した。しかし、cast forward は行列ベクトル積演算を行うタイミングを変えるだけであるため、計算コストは従来の時間域 BIEM と変わらない。そのため、本報では、3 次元動弾性問題における時間域 BIEM において必要となるメモリ使用量と計算コストを減らす事を目的とし、それらを減らすアルゴリズムを示す。

2 動弾性問題における時間域境界積分方程式法

動弾性問題における時間域 BIEM の例として、3 次元無限弾性領域 D に存在する円形クラック S による平面波 $\boldsymbol{u}^{\mathrm{in}}$ の散乱問題を考え、変位 $\boldsymbol{u}$ に関する次の初期値境界値問題を解く (図 1)。

$$\mu\Delta\boldsymbol{u} + (\lambda+\mu)\nabla\nabla\cdot\boldsymbol{u} = \rho\ddot{\boldsymbol{u}} \ \text{ in } D\setminus S\times(t>0), \tag{4}$$

$$\boldsymbol{u}|_{t=0} = \dot{\boldsymbol{u}}|_{t=0} = 0 \ \text{ in } D, \tag{5}$$

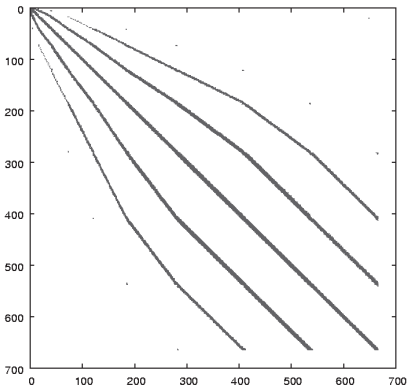
$$\mathrm{T}\boldsymbol{u}^{\pm} = 0 \ \text{ on } S\times(t\geq 0), \tag{6}$$

$$\boldsymbol{\varphi} = \boldsymbol{u}^{+} - \boldsymbol{u}^{-} = 0 \ \text{ on } \partial S, \tag{7}$$

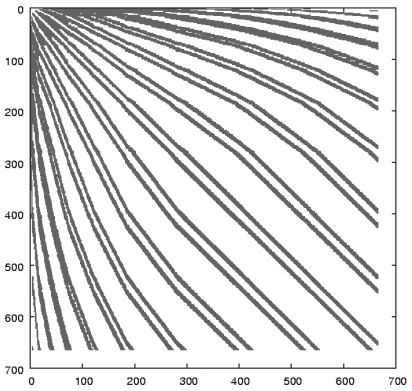
$$\text{radiation condition for } \boldsymbol{u}-\boldsymbol{u}^{\mathrm{in}} \text{ as } |\boldsymbol{x}|\to\infty, \tag{8}$$

ここで、λ, μ はラメ定数、ρ は密度、T はトラクション作用素、$+(-)$ はクラックの単位法線ベクトル $\boldsymbol{n}$ の正（負）からの極限値、‘$\dot{(\ )}$’ は時間微分、$\boldsymbol{\varphi}$ はクラックの開口変位を表す。このとき、式 (4) から次の境界積分方程式が得られる。

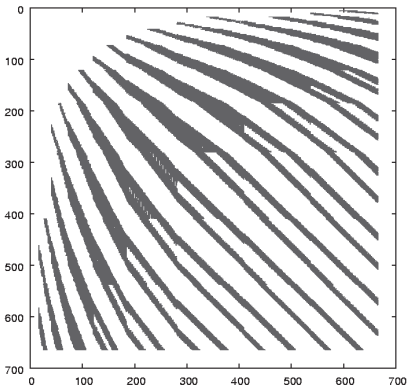
$$\boldsymbol{0} = \mathrm{T}\boldsymbol{u}^{\mathrm{in}} + \text{p.f.}\int_S \mathrm{T}\Gamma_I(\boldsymbol{x},\boldsymbol{y},t) * \varphi(\boldsymbol{y},t)dS, \quad \boldsymbol{x} \ \text{on} S, \tag{9}$$



$\boldsymbol{A}(\Delta t)$.

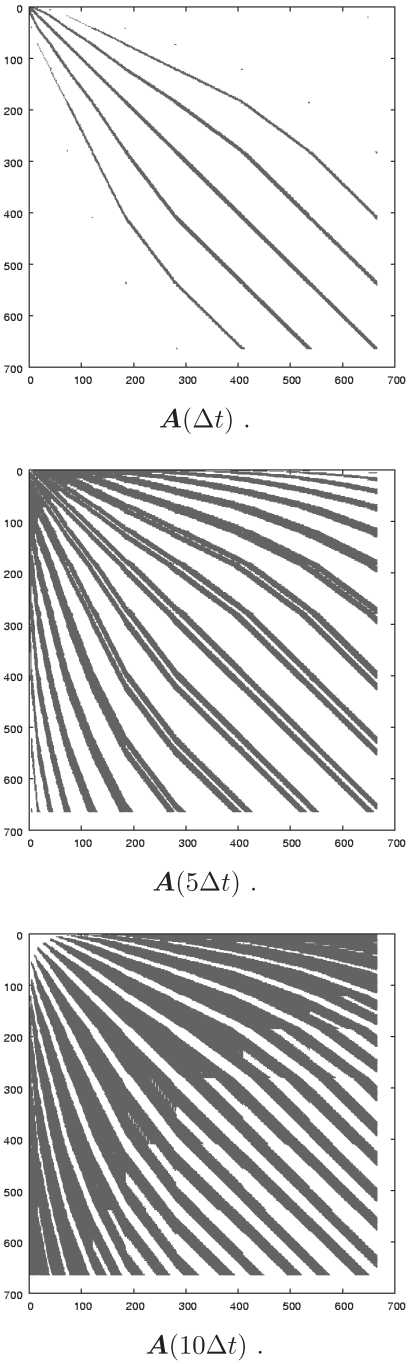


$\boldsymbol{A}(5\Delta t)$.



$\boldsymbol{A}(10\Delta t)$.

図 2: Nonzero components of the influence matrix in TD-BIEM for scalar waves.



$\boldsymbol{A}(\Delta t)$.

$\boldsymbol{A}(5\Delta t)$.

$\boldsymbol{A}(10\Delta t)$.

図 3: Nonzero components of the influence matrix in TD-BIEM for elastodynamics.

ここで、'$*$' は時間に関する畳み込み積分 $f(t)*g(t) = \int f(t-s)g(s)ds$ であり、p.f. は発散積分の有限部分を表す。また、$\boldsymbol{\Gamma}_I$ は二重層核であり、動弾性問題の基本解 $\boldsymbol{\Gamma}$ を用いて次式で表される。

$$\begin{aligned} & \Gamma_{Iij}(\boldsymbol{x},\boldsymbol{y},t) \\ = \quad & C_{jklm}\frac{\partial}{\partial y_l}\Gamma_{im}(\boldsymbol{x},\boldsymbol{y},t)n_k \\ = \quad & -\lambda\Gamma_{il,l}(\boldsymbol{x},\boldsymbol{y},t)n_l \\ & -\mu\Gamma_{ik,j}(\boldsymbol{x},\boldsymbol{y},t)n_k - \mu\Gamma_{ij,k}(\boldsymbol{x},\boldsymbol{y},t)n_k, \end{aligned} \tag{10}$$

ここで、$(\quad)_{,i} = \dfrac{\partial}{\partial x_i}$、$c_L = \sqrt{\dfrac{\lambda+2\mu}{\rho}}, c_T = \sqrt{\dfrac{\mu}{\rho}}$ である。時間域の境界積分方程式法では、積分方程式 (9) を時間内挿関数 $M^\ell(t)$、空間内挿関数 $N^q(\boldsymbol{x})$ を用いて離散化して解く。離散化された境界積分方程式を次式に示す。

$$\begin{aligned} - \quad & \sum_q A_{ij}^{pq}(\Delta t)\phi_j(\boldsymbol{x}^q, n\Delta t) \\ = \quad & \mathrm{T}u_i^{\mathrm{in}}(\boldsymbol{x}^p, n\Delta t) \\ + \quad & \sum_q \sum_{\ell=1}^{n-1} A_{ij}^{pq}((n+1-\ell)\Delta t)\phi_j(\boldsymbol{x}^q, \ell\Delta t), \end{aligned} \tag{11}$$

$$\begin{aligned} & A_{ij}^{pq}(\ell\Delta t) \\ = \quad & \int_S \int \mathrm{T}\Gamma_{Iij}(\boldsymbol{x}^p, \boldsymbol{y}, \tau)M^\ell(\tau)N^q(\boldsymbol{y})d\tau dS, \end{aligned} \tag{12}$$

ここで、点 $\boldsymbol{x}^p$ は境界要素 S_p の代表点である。

3 メモリ使用量、計算コスト削減のためのアルゴリズム改良

時間域の BIEM を解く際、代数方程式 (11) の影響係数行列 $\boldsymbol{A}^{pq}$ を式 (12) の積分により計算しストアする必要がある。空間内挿関数 N^q の台であるソース要素 S_q から発生する時間幅 ΔT(時間内挿関数 M^ℓ の台) を持つ弾性波動が、観測点 $\boldsymbol{x}^p$ に到達してから通過し終わるまでの間、影響係数 $\boldsymbol{A}^{pq}(\ell\Delta t)$ は非ゼロの値を持つ。時間域 BIEM では、メモリ使用量を減らすため通常、係数行列の非ゼロ成分のみを計算しストアする。動弾性問題を扱う場合、ソース要素 S_q と観測点 $\boldsymbol{x}^p$ との影響係数は P 波の到達から S 波の通過までが非ゼロとなる。

影響係数行列の非ゼロ成分の計算量、ストアに必要なメモリ量を減らすために、動弾性問題の基本解 (式

(3)) に着目する。式 (3) から明らかなように、ソース点 $\boldsymbol{y}$ から観測点 $\boldsymbol{x}$ への影響は P 波到達時間 $t = \dfrac{r}{c_L}$ から S 波の到達時刻 $t = \dfrac{r}{c_T}$ までが非ゼロとなる。さらに、P 波到達時刻から S 波到達時刻まで $\dfrac{r}{c_L} \leq t \leq \dfrac{r}{c_T}$ では、基本解は

$$\Gamma_{ij}(\boldsymbol{x}, \boldsymbol{y}, t) = \frac{c_T^2}{4\pi\mu} \frac{\partial^2}{\partial y_i \partial y_j} \frac{t - r/c_L}{r}, \qquad (13)$$

となり時間 t に関して線形である。基本解の時間について線形な特徴を活かし、代数方程式の影響係数 $\boldsymbol{A}^{pq}(\ell\Delta t)$ を求める。

ここで、非ゼロの影響係数が計算される時間区間に次の様に 4 つの時刻 t_1, t_2, t_3, t_4 を設定する。

- 時刻 t_1: 観測点 $\boldsymbol{x}^p$ に最も近いソース要素 S_q の内点から時刻 $t = 0$ に発生した波動の P 波成分が観測点 $\boldsymbol{x}^p$ に到達する時刻
- 時刻 t_2: 観測点 $\boldsymbol{x}^p$ に最も遠いソース要素 S_q の内点から時刻 $t = \Delta T$ に発生した波動の P 波成分が観測点 $\boldsymbol{x}^p$ に到達する時刻
- 時刻 t_3: 観測点 $\boldsymbol{x}^p$ に最も近いソース要素 S_q の内点から時刻 $t = 0$ に発生した波動の S 波成分が観測点 $\boldsymbol{x}^p$ に到達する時刻
- 時刻 t_4: 観測点 $\boldsymbol{x}^p$ に最も遠いソース要素 S_q の内点から時刻 $t = \Delta T$ に発生した波動の S 波成分が観測点 $\boldsymbol{x}^p$ を通過する時刻

このとき、$t_2 < t_3$ 且つ $t_2 < m\Delta t,\ n\Delta t < t_3$ となる整数 $m, n (m < n)$ が存在すれば、影響係数 $A_{ij}^{pq}(m\Delta t)$ を計算しメモリにストアすれば、影響係数 $A_{ij}^{pq}(\ell\Delta t), (\ell = m+1, \cdots, n)$ はストアされている値 $A_{ij}^{pq}(m\Delta t)$ と時間線形性より次式で容易に計算可能であり、また得られた値をメモリにストアする必要もない。

$$\boldsymbol{A}_{ij}^{pq}(\ell\Delta t) = \boldsymbol{A}_{ij}^{pq}(m\Delta t) + (\ell - m)\Delta A_{ij}^{pq}, \quad (\ell = m+1, \cdots, n), \qquad (14)$$

ここで、ΔA_{ij}^{pq} は時間に関して定数である。つまり、基本解の時間線形性を利用し、影響係数 A_{ij}^{pq} の計算を可能な限り式 (12) による積分ではなく、式 (14) の線形計算により求め、計算コストと使用メモリを削減する。このアイデアをコードに組込み動弾性問題の計算コストと使用メモリの削減を目指す。

図 4: Source element S_q and observation point $\boldsymbol{x}^p$.

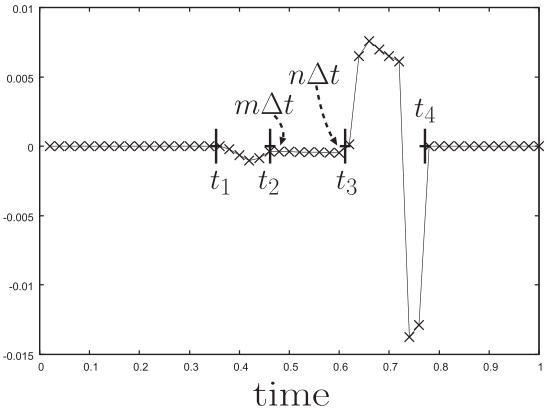


図 5: Influence coefficients $A_{32}^{pq}(t)$.

検証のため、図 4 に示す同一面上のソース要素 S_q と観測点 $\boldsymbol{x}^p$ との影響係数成分 $\boldsymbol{A}_{32}^{pq}(t)$ を図 5 にプロットした。ここで、弾性波速度はそれぞれ、$c_L = \sqrt{3}, c_T = 1$ とし、時間ステップ幅は $\Delta t = 0.02$ とした。また、ソース要素 S_q からは $\Delta T = 2\Delta t$ の台を持つ波動が発信されるとした。影響係数は、時刻 $t_1 \leq t \leq t_4$ で非ゼロの値となっている。また、図 5 の時刻 $t_1 \leq t \leq t_3$ 付近を拡大し図 6 に示した。$t = \ell\Delta t, (\ell = m, \cdots, n)$ において線形の挙動が確認できる。

4 数値結果

提案手法の有効性を検証するために、3 次元無限弾性領域 D の $x_3 = 0$ 上に存在する半径 $a = 1$ の円形クラック S による平面波 $\boldsymbol{u}^{\mathrm{in}}$ による散乱問題を解いた (図 1)。なお、入射波には、入射波によるトラクションが

$$\left\{\mathrm{T}\boldsymbol{u}^{\mathrm{in}}(\boldsymbol{x}, t)\right\}_3 = H(t - x_3/c_L), \qquad (15)$$

となるような平面波を用いた。弾性波速度はそれぞれ、$c_L = \sqrt{3}, c_T = 1$ とした。円形クラックを 2712

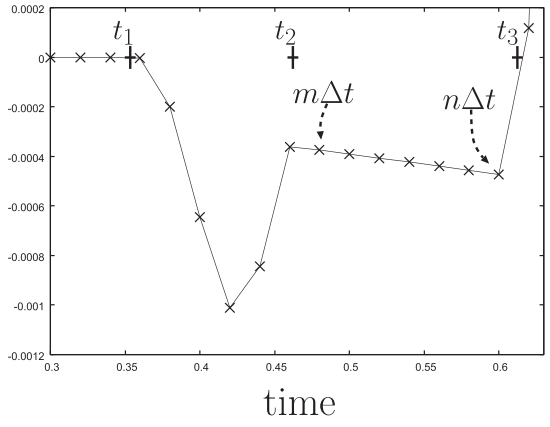


図 6: Influence coefficients $A_{32}^{pq}(t)$ (Close up of Fig. 5).

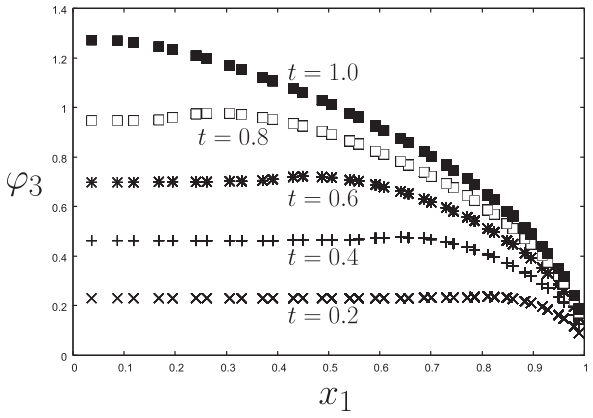


図 7: Opening displacement of the crack.

要素 (DOF:8136) に分割し、時間ステップ幅を $\Delta t = 0.02$ として、時間ステップ数が 25 ステップと 50 ステップまでの計算を行った。離散化には、$\Delta T = 2\Delta t$ の台を持つ区分線形の時間内挿関数 $M^{\ell}(t)$、区分一定の空間内挿関数 $N^{q}(\boldsymbol{x})$ を用いた。なお、区分線形の時間内挿関数 $M^{\ell}(t)$ を用いた場合、式 (14) の定数項は表れず、影響係数は、

$$\boldsymbol{A}_{ij}^{pq}(\ell\Delta t) = \ell\Delta A_{ij}^{pq}, (\ell = m+1, \cdots, n), \qquad (16)$$

と計算でき、動弾性問題の基本解の時間線形性を用いた影響係数の計算は非常に容易となる。数値解析により得られたクラックの開口変位の時間変動を図 7 に示す [1]。

使用メモリ、計算コストを従来の時間域 BIEM(表中の original) と cast forward 時間域 BIEM(表中の cf) と cast forward に本報で提案する時間線形性を利用した手法を加えた時間域 BIEM(表中の cf+proposed) で比較し、それぞれ表 1、表 2 に示す。cast forward では、従来の時間域 BIEM と計算コストに殆ど差はないが、本提案手法により削減出来ていることがわかる。また、メモリ使用量も提案手法では、 cast forward 時間域 BIEM よりもさらに削減できている。また、タイムステップ数が大きい程、基本解の線形性を利用した計算コスト・メモリコストの削減効果は顕著である [7]。なお、数値計算は、京都大学のスーパーコンピュータ Fujitsu HX600(Quad Core AMD Opteron) を用いて、1 コアによる逐次計算にて行った。

表 1 Memory requirements.(unit: GB)

N_t	original	cf	cf+proposed
25steps	1.80	0.436(24.2%)	0.389(21.6%)
50steps	7.62	1.80(23.6%)	1.17(15.4%)

表 2 Computational time. (unit: sec)

N_t	original	cf	cf+proposed
25steps	188	182(96.8%)	130(69.1%)
50steps	715	669(93.6%)	337(47.1%)

5 結論

3 次元動弾性問題の基本解の時間線形性を利用したアルゴリズムを組み込むことにより、時間と共に非ゼロ成分が増大する動弾性時間域境界積分方程式法の影響係数の計算量、メモリ使用量を削減した。

本論文では、例題としてクラック問題を取り扱った。有限のサイズのクラック問題では、ある程度の時間が経過すれば、クラック内のソース要素からの影響は、すべての観測点を通り過ぎてしまい、影響係数行列はいずれゼロ行列となる。本報に挙げた円形クラック問題の例では、$t = 2.0$ 以降、つまり $\boldsymbol{A}(\ell\Delta t), (\ell = 100, \cdots)$ はゼロ行列である。有限サイズの境界を持つ問題に対しては、本手法の有効性はある程度で頭打ちとなる。しかし、時間の経過と共にソース要素からの影響範囲に含まれる観測点の数が増大する無限の境界を持つ様な問題には、本手法は非常に有効である。今後は、無限境界を持つ問題に本手法を適用し、その有効性を確認する必要がある。

参考文献

[1] 小林昭一 他: 波動解析と境界要素法, (2000), 京都大学学術出版会.

[2] Walker, S.P.: Scattering analysis via time-domain integral equations: Methods to reduce the scaling of cost with frequency, IEEE Ant. Prop. Mag., **39** (1997), pp. 13–20.

[3] Walker, S.P. and Lee, B.H.: Reduced-cost methods for large time domain integral equation scattering analyses, Commun. Num. Meth. Eng., **14** (1998), pp. 751–761.

[4] Dodson, S.J., Walker, S.P. and Bluck, M.J.: Costs and cost scalings in time domain integral equation analysis of electromagnetic scattering, IEEE Ant. Prop. Mag., **40** (1998), pp. 12–21.

[5] 吉川仁、西村直志、小林昭一: 3 次元時間域動弾性問題における境界積分方程式法のアルゴリズム改良と並列化, 土木学会応用力学論文集 **5**(2002), pp. 199–206.

[6] H. Yoshikawa and N. Nishimura: An improved implementation of time domain elastodynamic BIEM in 3D for large scale problems and its application to ultrasonic NDE Electronic Journal of Boundary Elements **1**,Issue 2 (2003), pp. 201–217.

[7] 吉川仁, 西村直志: 基本解の時間線形性を利用した動弾性時間域 BIEM のメモリ使用量と計算コストの削減に関する研究, 計算数理工学論文集 **9**(2009), pp. 85–89.

Thin SMP クラスタ運転状況　(2009/10 ～ 2010/3)

1）保守作業に伴うサービス休止およびシステムダウン障害発生状況

保守作業に伴うサービス休止

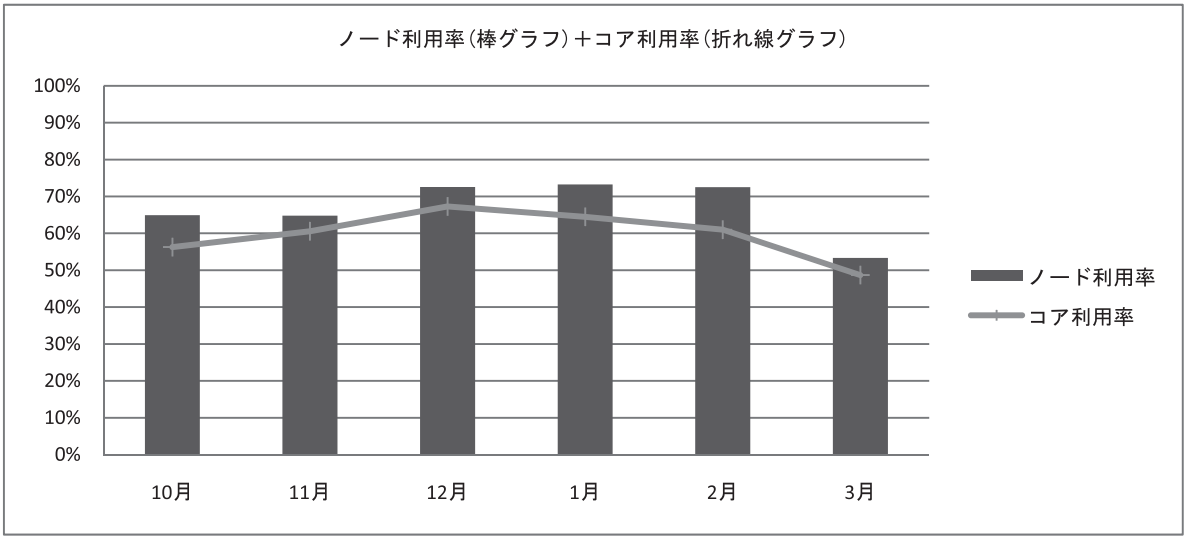
保守開始日時	サービス再開日時	保守時間[h]
2009/10/12　7:00	2009/10/13　17:15	34.3
2010/03/30　17:00	2010/04/01　0:00	31.0

システムダウン障害発生状況

障害発生日時	サービス再開日時	ダウン時間[h]
2010/01/22　8:50	2010/01/22　14:45	5.9
2010/03/08　17:20	2010/03/08　22:40	5.3

2）サービス状況

	サービス時間[h]	バッチ						TSS			
		処理件数	経過時間[h]	占有時間[h]	CPU時間[h]	平均稼動ノード数	ノード利用率	セッション数	セッション時間[h]	CPU時間[h]	平均稼動ノード数
10 月	710	33,321	211,911	2,348,915	2,141,864	364.9	64%	13,926	60,720	79,290	39.4
11 月	720	29,353	177,105	2,567,761	2,446,553	371.0	64%	16,465	65,217	118,269	40.0
12 月	744	31,843	228,908	2,947,943	2,794,117	371.0	72%	15,293	74,444	135,323	40.0
1 月	738	71,593	216,127	2,802,124	2,654,433	369.9	73%	15,037	76,229	151,816	39.9
2 月	672	36,852	192,691	2,413,186	2,316,287	373.0	72%	12,297	68,096	112,930	40.0
3 月	708	30,355	112,432	2,029,612	1,703,783	362.9	52%	9,743	46,204	95,014	39.1
計	4,292	233,317	1,139,174	15,109,541	14,057,037	368.8	66%	82,761	390,910	692,643	39.7



※ 占有時間 = 合計(経過時間×占有コア数)
※ 平均稼動ノード数 = 電源 ON 状態のノード数の月平均 (10 分間隔のサンプリングデータより算出)
※ ノード利用率 = 稼動ノードに対するジョブが実行されているノードの割合
※ TSS = ログインノード＋専用クラスタについてのデータ

Fat SMP クラスタ運転状況 （2009/10 ～ 2010/3）

1）保守作業に伴うサービス休止およびシステムダウン障害発生状況

保守作業に伴うサービス休止

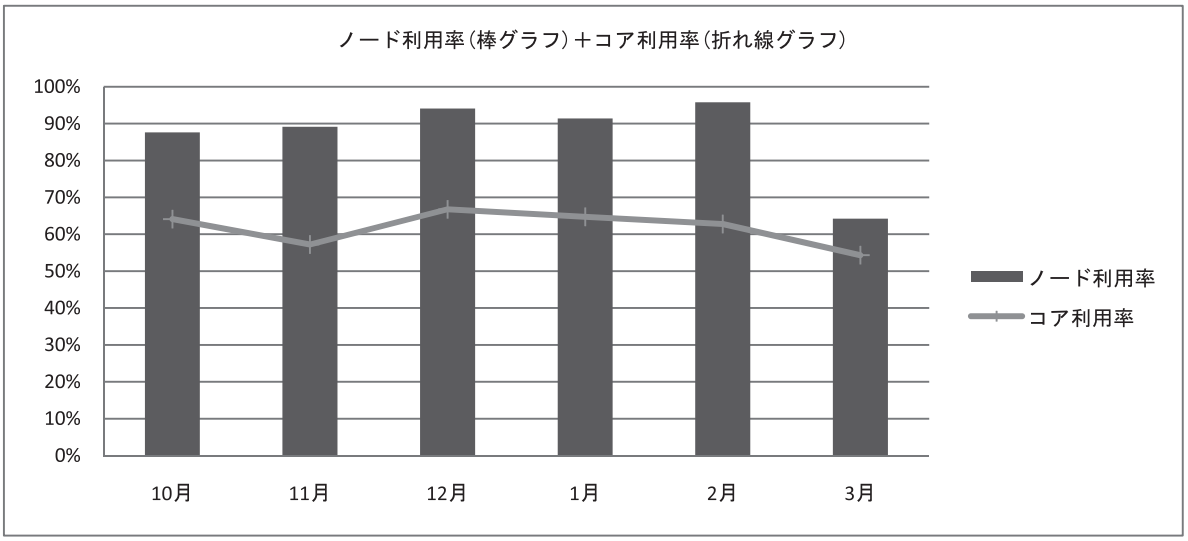
保守開始日時	サービス再開日時	保守時間[h]
2009/10/12 7:00	2009/10/13 17:15	34.3
2010/03/30 17:00	2010/04/01 0:00	31.0

システムダウン障害発生状況

障害発生日時	サービス再開日時	ダウン時間[h]
2010/01/22 8:50	2010/01/22 14:45	5.9
2010/03/08 17:20	2010/03/08 22:40	5.3
2010/03/18 15:10	2010/03/18 17:30	2.3
2010/03/18 20:05	2010/03/18 22:20	2.3

2）サービス状況

	サービス時間[h]	バッチ						TSS			
		処理件数	経過時間[h]	占有時間[h]	CPU時間[h]	平均稼動ノード数	ノード利用率	セッション数	セッション時間[h]	CPU時間[h]	平均稼動ノード数
10月	710	3,076	34,112	378,086	386,799	6.9	86%	2,751	22,826	1,543	1.0
11月	720	3,107	34,594	342,808	359,433	7.0	88%	3,626	37,288	4,047	1.0
12月	744	2,685	23,798	413,167	414,771	7.0	93%	3,221	36,352	9,332	1.0
1月	738	2,783	24,442	397,346	381,154	7.0	90%	3,354	37,333	5,649	1.0
2月	672	2,718	24,216	351,018	334,048	7.0	95%	2,905	29,992	7,855	1.0
3月	703	1,522	14,430	320,017	280,358	6.9	59%	2,198	26,239	7,832	1.0
計	4,287	15,891	155,592	2,202,443	2,156,563	7.0	85%	18,055	190,031	36,259	1.0



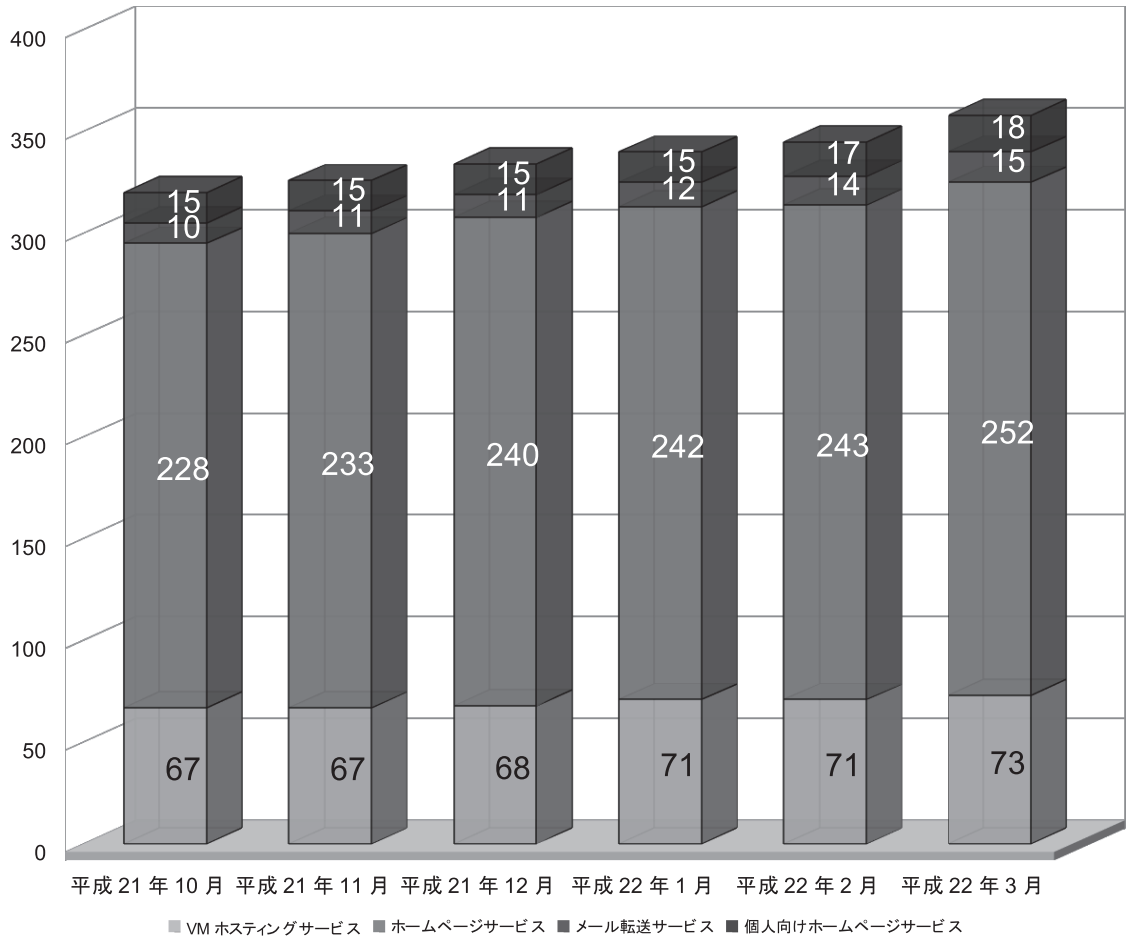
※ 占有時間 = 合計(経過時間×占有コア数)

※ 平均稼動ノード数 = 電源 ON 状態のノード数の月平均 (10 分間隔のサンプリングデータより算出)

※ ノード利用率 = 稼動ノードに対するジョブが実行されているノードの割合

汎用コンピュータシステムのサービス状況

1　ホスティング・ホームページサービス利用状況



（平成 21 年 10 月から平成 22 年3月）

大型計算機システム利用承認件数について

平成 22 年3月末現在大型計算機システムの利用件数は、2,693 件となっています。

2010 年度　京都大学学術情報メディアセンター
コンテンツ作成共同研究の実施について

京都大学学術情報メディアセンターでは、新たな学術・教育コンテンツの開発を行うことを目的とした「コンテンツ作成共同研究」を実施致しております。本年度の募集は既に締切りましたが、サービスの記録のため、その募集要項を掲載します。(3件の研究を採択。)

なお、2011 年におきましても同様の募集を行う予定ですので、ご興味をもたれた方は募集案内 Web サイト http://cpt.media.kyoto-u.ac.jp/collaborative/2010/ をご照会ください。

2010 年度　京都大学学術情報メディアセンターコンテンツ作成共同研究　募集要領

＜研究内容＞

具体的な内容としては，学術教育機関からの教育・研究活動の発信と活動そのものの発展に関わるコンテンツ開発を含む計画，学術教育資源の高度アーカイブ化 とその利活用を促進するようなコンテンツ開発，インターフェイス設計などといった実験・研究的要素の高い計画が考えられます．採択された計画は，その実施 に必要なコンテンツ作成室の作業工数負担金の一定の範囲内をセンターにて負担いたします．コンテンツ作成支援サービスとは異なり，作成に専門の設備や技能が必要となるだけではなく，制作会社等に発注することが困難であるが，共同研究実施後その成果を広く社会に寄与しうるコンテンツ作成の提案を募集します．

＜応募資格＞

研究申請代表者は、京都大学学術情報メディアセンターの全国共同利用規程の資格を持つ研究者・教員であること(具体的には以下の通り)。また、本センターの教員が1名以上共同研究者として加わる研究グループであること。

* 大学、短期大学、高等専門学校又は大学共同利用機関の教員及びこれに準ずる者
* 大学院の学生及びこれに準ずる者
* 学術研究を目的とする国又は自治体が所轄する機関に所属し、専ら研究に従事する者
* 科学研究費補助金等の交付を受けて学術研究を行う者
* その他センター長が必要と認めた者

＜応募方法・募集期間＞

所定の様式にしたがって申請書(3 ページ)を作成の上、2010 年 6 月 1 日(火曜日)～6 月30日(水曜日)の期間に郵送・学内便もしくは電子メールでご提出ください。記載された個人情報につきましては、本申請に関わることのみについて利用させて頂きます。ご不明な点は事前にご相談ください。

＜問い合わせ先・申請書提出先＞

応募に関するお問い合わせも、このアドレスまでお送りください。

郵送・学内便の場合

〒606-8501 京都市左京区吉田二本松町　京都大学学術情報メディアセンター南館コンテンツ作成室

電子メールの場合

京都大学 学術情報メディアセンター コンテンツ作成室 cpt@media.kyoto-u.ac.jp

＜審査方法、審査＞

応募された提案は，京都大学学術情報メディアセンターコンテンツ作成共同研究企画委員会において，研究内容の新規性と独自性，計画の妥当性，また，本セン ターにおけるコンテンツ作成共同研究のための予算の範囲内で実施可能なものかを審査の上採否を決定します．採否は委員会での決定後、電子メールにて７月上 旬に連絡いたします．

＜研究の実施、研究成果公開、知財登録＞

採択後は，研究グループとコンテンツ作成室で実施計画を作成し、それに基づき企画を実施することとなります。研究終了時には，研究報告書をご提出頂き，京都大学学術情報メディアセンター「全国共同利用広報」にて公開することを条件とします．共同研究の成果として得られたコンテンツは，京都大学産官学連携センター知的財産室ソフトウェア・コンテンツ分野に著作物として登録する，または本学の研究資源アーカイブに研究資源として登録する事を原則とします．なお，研究成果の一部は，本センターの共同研究実施例として研究報告から抜粋し，センターWeb サイトやパンフレット等へ情報の掲載を行うことがありま す．また採択者が本共同研究による研究成果（コンテンツ）を学術論文誌や各種メディア等において公開する場合，本共同研究で開発作成を行ったことを明記してください．

＜学術情報メディアセンターコンテンツ作成共同研究企画委員会委員名簿＞

委員長:河原 達也　　京都大学 学術情報メディアセンター 教授

委員:角所 考 関西学院大学 理工学部 教授

菊池 誠 大阪大学 サイバーメディアセンター 教授

黒江 康明 京都工芸繊維大学 大学院工芸科学研究科 教授

柴山 守 京都大学 東南アジア研究所

教授 杉万 俊夫 京都大学 大学院人間・環境学研究科 教授

吉岡 洋 京都大学 大学院文学研究科 教授

奥村 昭夫 京都大学 学術情報メディアセンター 客員教授

土佐 尚子 京都大学 学術情報メディアセンター 特定教授

美濃 導彦 京都大学 学術情報メディアセンター 教授

椋木 雅之 京都大学 学術情報メディアセンター 准教授

元木 環 京都大学 学術情報メディアセンター 助教

赤坂 浩一 京都大学 情報環境部 情報基盤課 学術情報基盤グループ

小西 満 京都大学 情報環境部 情報基盤課 共同利用グループ

大型計算機システム利用負担金　（2009年4月1日より）

別表1　スーパーコンピュータシステム

			利用負担額	提供サービス					
コース	タイプ	セット		システム	バッチ	システム資源	経過時間（時間）	ディスク（GB）	利用者番号
エントリ	—	基本	12,600 円/年	Thin SMP	共有	最大1ノード相当（並列数16、メモリ32GB）	1	60	—
パーソナル	タイプ1	基本	100,000 円/年	Thin SMP	共有	最大2ノード相当（並列数32、メモリ64GB）	168	600	—
	タイプ2	基本	100,000 円/年	Fat SMP	共有	最大2ソケット相当（並列数8、メモリ64GB）	168	600	—
グループ	タイプ1	最小	250,000 円/年	Thin SMP	優先	2ノード（（16コア、メモリ32GB）×2）	336	2,000	6
		追加単位	250,000 円/年				—	2,000	6
	タイプ1B	最小	300,000 円/年	Thin SMP	準優先	4ノード（（16コア、メモリ32GB）×4）	336	2,400	12
		追加単位	150,000 円/年			2ノード（（16コア、メモリ32GB）×2）	—	1,200	6
	タイプ1C	最小	750,000 円/年	Thin SMP	占有	4ノード（（16コア、メモリ32GB）×4）	336	4,000	12
		追加単位	375,000 円/年			2ノード（（16コア、メモリ32GB）×2）	—	2,000	6
	タイプ2	最小	400,000 円/年	Fat SMP	優先	4ソケット(16コア、メモリ128GB)	336	4,000	12
		追加単位	200,000 円/年			2ソケット(8コア、メモリ64GB)	—	2,000	6
	タイプ2B	最小	240,000 円/年	Fat SMP	準優先	4ソケット(16コア、メモリ128GB)	336	2,400	12
		追加単位	120,000 円/年			2ソケット(8コア、メモリ64GB)	—	1,200	6
大規模ジョブ	タイプ1	最小	24,000 円/週(7日)	Thin SMP	優先	4ノード（（16コア、メモリ32GB）×4）	—	—	—
		追加単位	6,000 円/週(7日)			1ノード（16コア、メモリ32GB）	—	—	—
	タイプ2	最小	20,000 円/週(7日)	Fat SMP	優先	4ソケット(16コア、メモリ128GB)	—	—	—
		追加単位	5,000 円/週(7日)			1ソケット(4コア、メモリ32GB)	—	—	—
専用クラスタ	—	最小	750,000 円/年	Thin SMP	—	4ノード（（16コア、メモリ32GB）×4）	—	4,000	12
	—	追加単位	375,000 円/年			2ノード（（16コア、メモリ32GB）×2）	—	2,000	6
ライセンスサービス			20,000 円/年	可視化ソフト（AVS,ENVI/IDL)およびプリポストウェアの1ライセンスにつき					

備考

1. 利用負担額は、年度単位で算定している。また、総額表示である。

2. 大型計算機システムの全ての利用者は、上記表のサービスの他、次のサービスを受けることができる。
   1） 大判プリンタサービス
   2） その他、大型計算機システムが提供するサービス、機器の利用

3. 上記表の大規模ジョブコース、ライセンスサービスの申請には、大型計算機システムの利用者であることが必要である。

4. 「共有」: 当該カテゴリのユーザ間で一定の計算資源を共有するベストエフォートのスケジューリングを行う。
   「準優先」: 定常稼働状況において記載値（以上）の計算資源が確保されるように優先スケジューリングを行う。
   また、稼働状況によらず記載値の1/4の計算資源が確保されることを保証する。
   「優先」: 定常稼働状況において記載値（以上）の計算資源が確保されるように優先スケジューリングを行う。
   また、稼働状況によらず記載値の1/2の計算資源が確保されることを保証する。
   「占有」: 稼働状況によらず記載値（以上）の計算資源が確保されることを保証する。

5. ディスク容量はバックアップ領域（最大で総容量の1/2)を含む。

6. グループコース及び専用クラスタコースのシステム資源は、下記の負担額を支払うことにより増量することができる。
   なお増量は各月1日に実施し、増量した資源は当該年度末までの期間にわたって利用されるものとする。

コース	タイプ	追加負担金額（増量単位あたり）	システム資源増量単位	ディスク増量（GB）
グループ	タイプ1	25,000 円/月	2ノード（（16コア、メモリ32GB）×2）	2,000
	タイプ1B	15,000 円/月	2ノード（（16コア、メモリ32GB）×2）	1,200
	タイプ1C	37,500 円/月	2ノード（（16コア、メモリ32GB）×2）	2,000
	タイプ2	20,000 円/月	2ソケット(8コア、メモリ64GB)	2,000
	タイプ2B	12,000 円/月	2ソケット(8コア、メモリ64GB)	1,200
専用クラスタ	—	37,500 円/月	2ノード（（16コア、メモリ32GB）×2）	2,000

7. グループコース及び専用クラスタコースを通年でなく利用する場合には、下記の負担額を支払うものとする。
ただし、利用期間は当該年度内に限るものとする。

利用期間			3ヶ月	6ヶ月	9ヶ月
グループコース	タイプ1	最小	100,000 円	150,000 円	225,000 円
		追加単位	100,000 円	150,000 円	225,000 円
	タイプ1B	最小	120,000 円	180,000 円	270,000 円
		追加単位	60,000 円	90,000 円	135,000 円
	タイプ1C	最小	300,000 円	450,000 円	675,000 円
		追加単位	150,000 円	225,000 円	337,500 円
	タイプ2	最小	160,000 円	240,000 円	360,000 円
		追加単位	80,000 円	120,000 円	180,000 円
	タイプ2B	最小	96,000 円	144,000 円	216,000 円
		追加単位	48,000 円	72,000 円	108,000 円
専用クラスタ コース	—	最小	300,000 円	450,000 円	675,000 円
	—	追加単位	150,000 円	225,000 円	337,500 円

8. グループコース及び専用クラスタコースの利用者番号は利用者あたり年額5,000円を負担することで追加できる。

9. 機関・部局定額制度
他機関又は学内における部局(『国立大学法人京都大学の組織に関する規程』第3章第2節から第11節で定める組織をいう。)の組織が、その組織単位でグループコースサービス(年間)の利用を申請する場合、料金表(年間)に掲載額の1.5倍を利用負担金とする。なお、利用負担金額が150万円未満の場合は100人、150万円を超える場合は、150万円毎に100人までの利用者を認める。

別表2 汎用コンピュータシステム

区分	利用負担額	単位
VMホスティングサービス	126,000円／年	1仮想マシンにつき
ホームページサービス	31,500円／年	1ドメイン名につき
個人向けホームページサービス	12,600円／年	1アカウントにつき
メール転送サービス	12,600円／年	1ドメイン名につき

備考

1. 利用負担額は、総額表示である。

2. 上記表の汎用コンピュータシステムのサービスを利用するためには、大型計算機システムの利用者であることが必要である。

3. ホームページサービス及びVMホスティングサービスにおいて、下記の負担額を支払うことによりオプションサービスを利用することができる。

オプションサービス種別	利用負担額	単位
データベース(Oracle)	63,000円／年	1アカウントにつき
ストリーミング(Helix	31,500円／年	1アカウントにつき

4. VMホスティングサービスのシステム資源は、下記の負担額を支払うことにより増量することができる。

種別	利用負担額	単位
ディスク	10,500円／年	100GBにつき
システム資源	100,800円／年	1台につき

システム資源1台とは、CPU:2コア,メモリ:2GB である。

5. VMwareを用いたVMホスティングサービスは、下記の負担額を支払うことにより利用・増量することができる。
ただし、システム資源が非常に限られているためサービスを提供できる場合が限定される。

種別	利用負担額	単位
標準機能サポート	25,200円／年	1仮想マシンにつき
ディスク	10,500円／年	100GBにつき
システム資源	201,600円／年	1台につき

システム資源1台とは、CPU:1コア,メモリ:2GB である。

6. 利用負担額は、当該年度(4月から翌年3月まで)の利用に対して年額として算定するが、年度途中から利用を開始する場合には月数に応じて減額する。

全国共同利用版広報・Vol.8(2009)総目次

［共同研究制度（大規模計算支援枠）研究報告］

［サービスの記録・報告］

［資料］

［編集後記］

― サービス利用のための資料一覧 ―

1. スーパーコンピュータシステム・ホスト一覧

- Thin SMP クラスタ：thin.kudpc.kyoto-u.ac.jp
  - Exceed onDemand 接続：thinX11.kudpc.kyoto-u.ac.jp
- Fat SMP クラスタ：fat.kudpc.kyoto-u.ac.jp

※ ホストへの接続は SSH(Secure Shell)のみ、telnet, ftp は不可

2. 問い合わせ先 & リンク集

- 情報環境機構のホームページ
  http://www.iimc.kyoto-u.ac.jp/

- 学術情報メディアセンターのホームページ
  http://www.media.kyoto-u.ac.jp/

- スーパーコンピュータシステムに関する問い合わせ先
  - 利用申請などに関する問合わせ先
    【共同利用支援グループ・共同利用担当（北館窓口）】
    E-mail：zenkoku-kyo@media.kyoto-u.ac.jp / Tel：075-753-7424 / Fax：075-753-7449
    URL: http://www.iimc.kyoto-u.ac.jp/ja/services/comp/
  - システムの利用など技術的な問い合わせ
    【コンピューティンググループ】
    E-mail：consult@kudpc.kyoto-u.ac.jp / Tel：075-753-7426
    URL: http://web.kudpc.kyoto-u.ac.jp/

- ホームページ・ホスティングサービス関する問合わせ先
  【学術情報基盤グループ】
  E-mail：whs-qa@media.kyoto-u.ac.jp / Tel：075-753-7494
  URL: http://www.iimc.kyoto-u.ac.jp/ja/services/whs/

- コンテンツ作成支援サービスに関する問合わせ先
  【コンテンツ作成室】
  E-mail：cpt@media.kyoto-u.ac.jp / Tel：075-753-9012
  URL: http://www.iimc.kyoto-u.ac.jp/ja/services/content/

編 集 後 記

以前、飛行機に乗ることが夢で、あちこち飛んでいきたいと思っておりました。最近機会があって、夢の実現というか、幸運にも何度も飛行機に乗る機会に恵まれ、全国を毎月、北の稚内空港から南の石垣空港まで飛びまわっております。今年の１月に、初めてニューヨーク ジョン F.ケネディー国際空港へ行きましたが、本当に遠かったこと、往路１２時間復路１４時間往復２１,０００ｋｍの飛行機搭乗旅は、私を本当に堪能させてくれました。当日は、天候も良く地上の風景を太平洋から北米大陸へ昼・夜・そして夜明けから朝と楽しむことができました。今後も、時間と体力と予算の許す限り１００万マイルの搭乗（現在６７万マイル）を目指して、搭乗したいと思っております。

飛行機オタク

W杯サッカーで日本中が盛り上がっている頃に発行される筈であった本号は、結局8月も過ぎ、欧州各国リーグも始まってから発刊されることになりました。今年はW杯で活躍した選手を中心に日本から欧州リーグに挑戦する選手が増え、日本でも欧州サッカーに注目が集まっているようです。私はと言えば、そんな日本選手達には目もくれず（というほどでもないですが）、今年もイタリアの某チームを応援していく所存です。Forza Roma！

ロマニスタ

京都大学学術情報メディアセンター全国共同利用版広報　Vol. 9, No. 1

2010年　9月　　28日発行

編集者　京都大学学術情報メディアセンター
　　広報教育委員会・全国共同利用版広報編集部会
発行者　〒606-8501　京都市左京区吉田本町
　　京都大学学術情報メディアセンター
　　Academic Center for Computing and Media Studies
　　Kyoto University
　　Tel. 075-753-7400
　　http://www.media.kyoto-u.ac.jp/
印刷所　京都市中京区壬生花井町３番地
　　ニッシャビジネスサービス株式会社