Wisteria/BDEC-01 利用事例（I）

Wisteria/BDEC-01（Odyssey）における疎行列解法の動的ループスケジューリングによる通信最適化

中 島 研 吾

東京大学情報基盤センター

0. 「Wisteria/BDEC-01 利用事例」について

東京大学情報基盤センター（本センター）では，2021 年 5 月 14 日に「計算・データ・学習」融合スーパーコンピュータシステム（Wisteria/BDEC-01）の運用を開始した〔1〕。

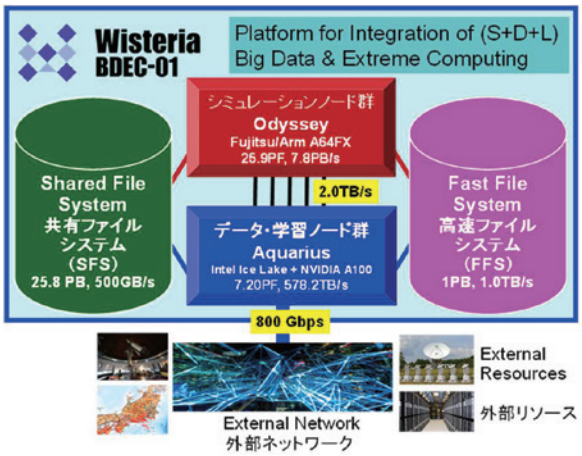

図 1　Wisteria/BDEC-01 の概要〔1〕

「Wisteria/BDEC-01」はシミュレーションノード群（Odyssey）とデータ・学習ノード群（Aquarius）の 2 つの計算ノード群を有するヘテロジニアスなシステムであり，特に「計算・データ・学習」融合が推進され，サイバー空間（仮想）とフィジカル空間（現実）を高度に融合させた Society 5.0 の実現に大きく貢献することが期待されている。「スーパーコンピューティングニュース」では，これから 1 年程度をかけて，「Wisteria/BDEC-01」の様々な利用事例を紹介していく予定である。今回は第 1 回として，シミュレーションノード群（Odyssey）における大規模並列数値解法の高速化について紹介する。

シミュレーションノード群（Odyssey）は「FUJITSU Supercomputer PRIMEHPC FX1000」20 ラックから構成され，世界最高性能を有するスーパーコンピュータ「富岳」と同じ富士通株式会社の「FUJITSU Processor A64FX」を 7,680 ノード（368,640 コア）搭載している。「A64FX」は，最先端の 7nm プロセスで製造され，48 個の演算コアと 2 個または 4 個のアシスタントコアを有し，倍精度浮動小数点演算で 3.3792 TFLOPS の理論ピーク性能を実現し，合計ピーク性能は 25.9 PFLOPS である。各ノードは 32 GiB の HBM2 メモリを搭載し，シミュレーションノード群（Odyssey）の総メモリ容量は 240 TiB，総メモリバンド幅は 7.8 PB／秒である。各ノードはバイセクションバンド幅が 13.0 TB／秒のノード間相互結合ネットワーク（Tofu インターコネクト D）で結合されている。

1. はじめに

大規模な並列計算機を使用する場合，ノード数の増加によって通信のオーバーヘッドは増加する傾向にある。並列計算において通信は必須のプロセスであるが，通信をできる限り効率的に実施し，削減することは EFLOPS 級システムにおいて重要である。通信

の削減がアルゴリズムに不安定をもたらす可能性もある。本稿では，計算科学，計算工学において広く使用されている偏微分方程式の数値解法である，有限要素法から導出される疎行列を係数とする大規模線形方程式（連立一次方程式）を，前処理付き並列反復法（Preconditioned Parallel Iterative Methods）によって解く場合について検討する。本稿の手法は著者等の先行研究〔2〕に基づくものである。

2. 並列有限要素法アプリケーション「GeoFEM/Cube」

本稿で対象としているアプリケーションは，GeoFEM プロジェクト〔3，4〕で開発された並列有限要素法アプリケーションを元に整備した性能評価のためのベンチマークプログラム「GeoFEM／Cube」であり，本センターでは，スーパーコンピュータの調達時の性能評価試験に使用している。本ベンチマークは，均質場における三次元弾性静解析問題（Cube 型モデル（図2)）に有限要素法を適用することにより導かれる大規模疎行列を係数行列とする連立一次方程式を並列前処理付き反復法（クリロフ部分空間法〔5〕）によって解くものであり，反復法ソルバーの実行時性能を様々な条件下で計測する。要素タイプは三次元一次六面体要素（tri-linear）であり，各要素 8 つの節点を有している。各要素は辺長さ＝1 の単位立方体であり，x，y，z 各方向の節点数はそれぞれ，N_x，N_y，N_z である（従って要素数は，N_x-1，N_y-1，N_z-1)。

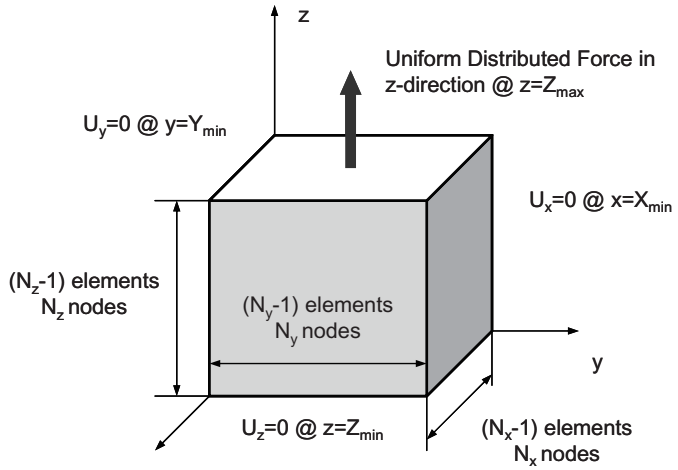

図 2　GeoFEM/Cube の解析対象〔2,3,4〕

三次元弾性問題では 1 節点あたり 3 つの自由度があるため，これらを 1 つのブロックとして取り扱っている（図 3)。係数行列はこのブロック型の特性を利用したブロック CRS 形式（Compressed Row Storage）によって格納されている。

有限要素法において（厳密にはガラーキン法に基づく有限要素法において)，三次元弾性静解析問題では係数行列が対称正定な疎行列となることから，前処理を施した共役勾配法（Conjugate Gradient，CG）法〔5〕（図 4）によって連立一次方程式を解いている。前処理手法としては，様々な手法が利用可能であるが，本稿では，図 3 に示す対角ブロックに対してフル LU 分解を適用する手法（Block Diagonal LU Factorization）〔2〕を適用している。この手法は，各節点単位で独立に前処理を適用できるため，並列化が容易であり，MPI による通信も発生しない。

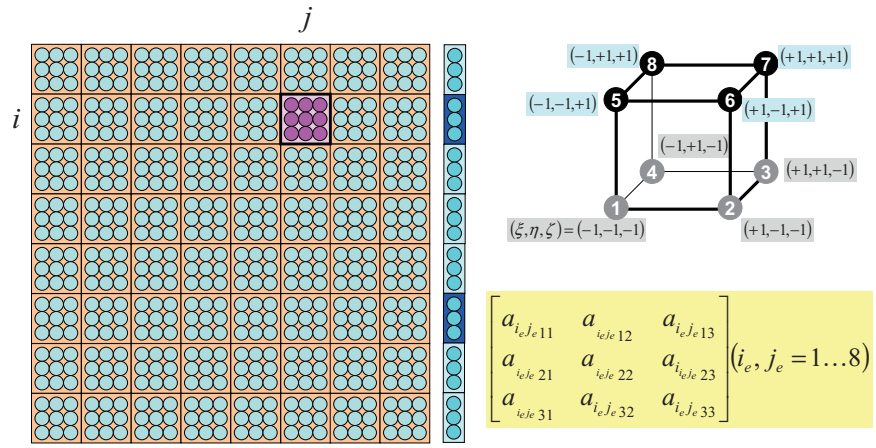

図 3　8 節点六面体要素，1 節点 3 自由度の処理〔2,3,4〕

有限要素法を並列化する場合には，オーバーラップ要素を有する節点ベースの分割手法〔2,3,4〕が広く使用されており，本稿でもそれに基づいている。有限要素法においては，大規模連立一次方程式の求解に計算時間の大部分が費やされることから，並列計算時には自由度が定義される節点（Node）が各プロセスで均等となるように領域分割を適用する。GeoFEM/Cube は規則正しい形状を扱うが，プログラム内に並列メッシュ生成機能を内蔵し，プロセス間負荷バランスを自動的に調整している。図 5 は，25 節点，16 要素（四角形）から構成される二次元解析対象を 4 分割した例である。PE#0～PE#3（PE=Processing Element）の 4 領域に分割され，各領域に MPI プロセスが割り当てられる。図 5（b）は各プロセスで扱う局所分散データを示している。

```
Compute r(0)= b−[A]x(0)
for i= 1, 2, …
    solve [M]z(i−1)= r(i−1)
    ρi−1= r(i−1) z(i−1)
    if i=1
      p(1)= z(0)
    else
      βi−1= ρi−1/ρi−2
      p(i)= z(i−1) + βi−1 p(i−1)
    endif
    q(i)= [A]p(i)
    αi  = ρi−1/p(i)q(i)
    x(i)= x(i−1) + αip(i)
    r(i)= r(i−1) − αiq(i)
    check convergence |r|
end
```

図 4　前処理付き共役勾配法のアルゴリズム〔5〕

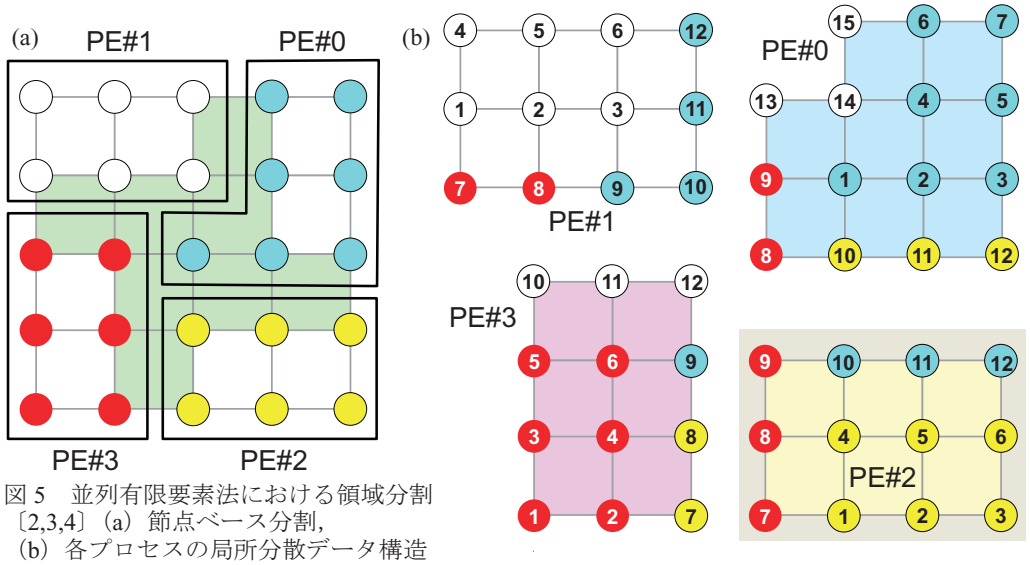

図 5　並列有限要素法における領域分割〔2,3,4〕（a）節点ベース分割，
（b）各プロセスの局所分散データ構造

図 5（b）に示す PE#2 は，図 5（a）に示すように本来黄色い節点が割り当てられているが，それ以外に，黄色い節点を含む要素，それに含まれる他の領域に割り当てられた節点（赤（PE#3），青（PE#1））が含まれている。並列有限要素法における局所分散データ構造においては，節点を以下の 3 種類のカテゴリに分類している〔2,3,4〕。

- 内点（Internal Nodes）：本来その領域に割り当てられた節点（図 5（a）PE#2 の黄色い節点）
- 外点（External Nodes）：本来異なる領域に割り当てられた節点
- 境界点（Boundary Nodes）：「内点」のうち，他の領域の「外点」である節点（外点に接する）

図 5 に示すようなオーバーラップ要素，外点を含む局所データ構造を適用することにより，要素マトリクス，全体マトリクスの生成を通信無しに実施することができ，また，共役勾配法のような反復法を並列化することも容易である〔2,3,4〕。**各局所分散データにおける局所節点番号は，図 5（b）に示す「内点⇒外点」のようにすると通信等が容易に実施できる。**なお，OpenMP/MPI ハイブリッド並列プログラミングモデルを適用する場合には，図 5（b）の各 MPI プロセスに割り当てられた局所分散データに対して OpenMP によるスレッド並列化を適用する〔2,3,4〕。

並列有限要素法で MPI プロセス間のグローバルな通信が発生するのは，本稿で扱うような線形，定常の問題の場合には，並列前処理付き反復法による線形方程式の求解部のみである。本稿では，Block Diagonal LU Factorization 前処理を適用しているため，図 4 に示すような前処理付き共役勾配法において，前処理適用部分（solve [M]$z^{(i-1)}=r^{(i-1)}$）は，各節点完全に独立に計算し，MPI プロセス間の通信は発生しない。したがって，MPI プロセス間通信が発生するのは，①内積，②疎行列ベクトル積（$q^{(i)}$= [A]$p^{(i)}$）のみである。②では，図 5（b）に示した局所分散データの特性を利用する。各 MPI プロセスで $q^{(i)}$= [A]$p^{(i)}$を計算するためには，外点における $p^{(i)}$の値を求めておく必要がある。そのためには MPI プロセス間通信によって，外点の値を他の MPI プロセスから予め受け取っておく必要がある。GeoFEM/Cube では，①境界点の値を MPI_Isend によって送信，②外点の値を MPI_Irecv によって受信，③MPI_Waitall によって通信を完了，という手順によって実施している〔2,3,4〕（図 6）。外点のことを「Halo」と呼ぶことがあり，図 6 に示すような通信を「Halo 通信」と呼ぶこともある。

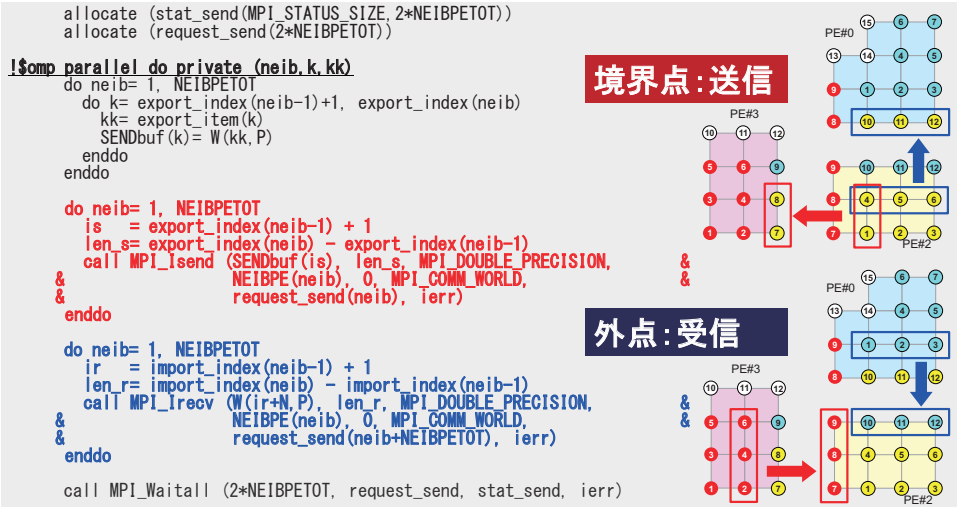

図 6　MPI_Isend/MPI_Recv/MPI_Waitall による境界点送信・外点受信（Halo 通信）〔2,3,4〕

3. 通信と計算のオーバーラップ

分散メモリ型並列計算機上で並列前処理付き反復法を実施する場合は，①疎行列ベクトル積，②内積，③前処理，で通信が発生する可能性がある。本稿における，例では前処理部分では通信は発生しない。疎行列ベクトル積では隣接プロセスとの 1 対 1 通信（Point-to-Point Communication），内積では全プロセスとの集合通信（Collective Communication）が発生する。MPI を使用する場合は，①は図 6 で示したような Halo 通信を適用し，MPI_Isend，MPI_Irecv を MPI_Waitall と組み合わせ，②においては MPI_Allreduce のような関数が使用される。

通信によるオーバーヘッドを削減するために，通信回避・削減型アルゴリズム（Communication Avoiding/Reducing Algorithm）の研究開発が盛んに進められている。パイプライン型共役勾配法（Pipelined CG Method）〔6〕は，漸化式の適用によって，図 4 に示したオリジナルの前処理付き CG 法のアルゴリズムが変わらないように計算の順序を変更する手法である。内積の直後に疎行列ベクトル積，前処理などの計算量が多く，かつ直前で実施した内積の結果を使わないような処理を実行するように計算順序が変更される。Pipelined 法では，これに MPI-3 でサポートされている MPI_Iallreduce 等の非同期集団通信（Asynchronous Collective Communication）を組み合わせることによって，集団通信と疎行列ベクトル積，前処理等の演算をオーバーラップすることができ，通信によるオーバーヘッドの隠蔽が可能となる〔6〕。

本稿では，特に疎行列ベクトル積部分の高速化について検討する。疎行列ベクトル積の計算を実施する場合には，図 6 に示すような Halo 通信によって，各 MPI プロセス間で通信を実施し，最新の p ベクトルの値を得たのち，各 MPI プロセスで内点に対して疎行列ベクトル積の演算を実施する〔2,3,4〕。内点のうち，境界点は外点に接しているため，外点の最新の値がないと疎行列ベクトル積を計算できないが，境界点以外の内点（純内点（Pure Internal Nodes））では通信の完了を待たずに計算を開始できる。これに基づき，図 6 に示す MPI プロセス間の Halo 通信と，「純内点」の計算を同時に実施する手法が，古典的な「通信と計算のオーバーラップ」〔7〕である（図 7）。図 8「Static」で計算手順の概要を紹介しているように，MPI_Isend，MPI_Irecv によって Halo 通信を起動させ，純内点の計算の後に MPI_Waitall を呼んで，通信と計算をオーバーラップさせている。純内点，境界点の計算を別々のループで実施していることから，図 7 に示すように，内点に reordering を施し，「純内点⇒境界点」となるように並び替えている。こうすることによって，各ループ内は分岐もなく，アクセスも連続となるため，計算効率が向上する〔2,8〕。

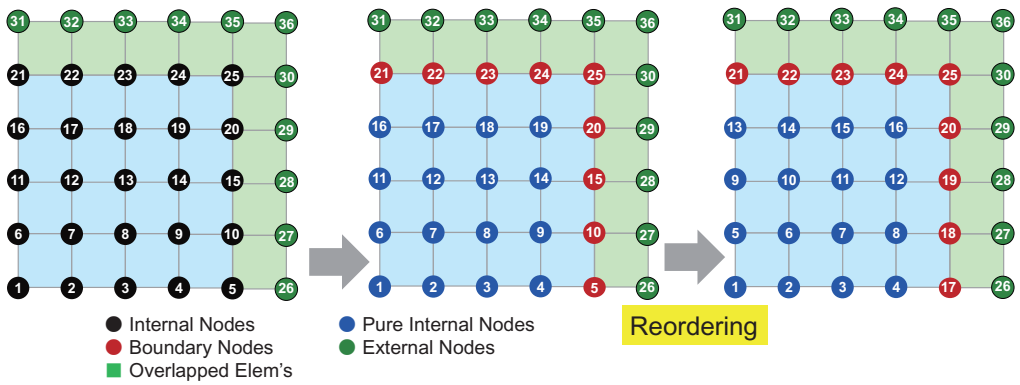

図 7　疎行列ベクトル積における通信と計算のオーバーラップ〔2,7,8〕

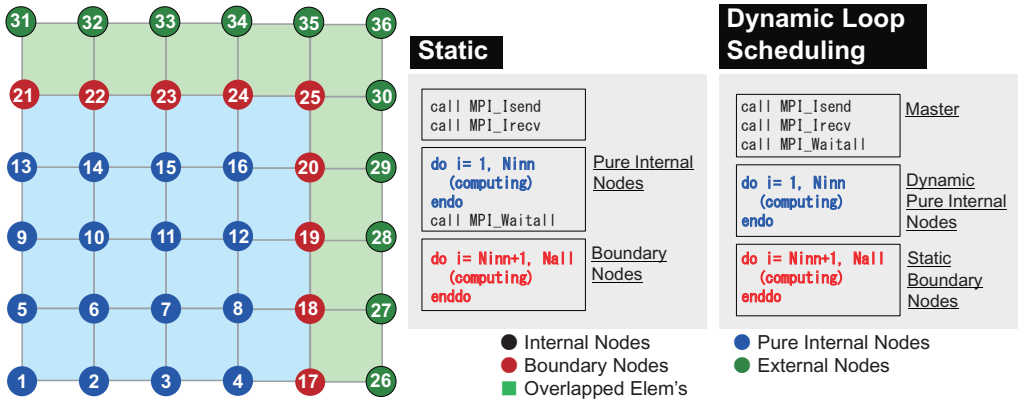

図 8　疎行列ベクトル積における計算と通信のオーバーラップ（Static：通信と純内点（Pure Internal Nodes）の計算をオーバーラップ，Dynamic Loop Scheduling：OpenMP のマスタースレッドが通信を担当する）

通信と計算のオーバーラップを更に効率化するために，Halo 通信に OpenMP の動的ループスケジューリング（Dynamic Loop Scheduling）機能を適用する手法が考案されている〔2，8〕。この手法では，Halo 通信を OpenMP のマスタースレッドが受け持ち，純内点の計算とオーバーラップさせる。図 8 の「Dynamic Loop Scheduling」がこれに相当している。

より詳細な実装を図 9 に示す。Halo 通信を OpenMP のマスタースレッドが受け持ち，残りのスレッドを使って純内点の計算を，Halo 通信とオーバーラップさせて実施している。「!$omp do schedule (dynamic,200)」とある「200」は「チャンクサイズ（Chunk Size）」である。これは，マスタースレッドが担当している通信が終了したかどうかをチェックするための間隔で，もし通信が終了していれば，マスタースレッドも純内点の計算に参加する。チャンクサイズが小さければ，それだけ細かい間隔で正確に通信の状況をチェックできるので，マスタースレッドが純内点の計算により早く参加でき，その分計算時間が短縮される可能性がある。ただ，その分，通信終了チェックのための回数が増加し，オーバーヘッドが大きくなる可能性もある。

```
!$omp parallel private (neib,j,k,i,X1,X2,X3,WVAL1,WVAL2,WVAL3)
!$omp&         private (istart,inum,ii,ierr)

!$omp master              Communication is done by the master thread (#0)
!C
!C- Send & Recv.
(…)
      call MPI_WAITALL (2*NEIBPETOT, req1, sta1, ierr)
!$omp end master

!C                          The master thread can join computing of internal
!C-- Pure Internal Nodes    nodes after the completion of communication

!$omp do schedule (dynamic,200)    Chunk Size= 200
      do j= 1, Ninn
        (…)
      enddo
!C
!C-- Boundary Nodes          Computing for boundary nodes are by all threads

!$omp do                    default: !$omp do schedule (static)
      do j= Ninn+1, N
        (…)
      enddo

!$omp end parallel
```

図 9　動的ループスケジューリング（Dynamic Loop Scheduling）の疎行列ベクトル積への適用例

4. 計算事例

3.までに示した手法について，Wisteria/BDEC-01（Odyssey）の最大 6,144 ノードまでを使って評価を実施し，同じく本センターの Oakforest-PACS（OFP）〔9〕を最大 2,048 ノード使用した場合と結果を比較した。Oakforest-PACS（OFP）と Odyssey のノード単体諸元を表 1 に示す。本稿に示したケースでは，OFP については Flat モードを適用し，MCDRAM のみを使用した。OFP は各ノード 68 コアを搭載しているが，64 コアを使用した。

表 1　Oakforest-PACS（OFP）と Odyssey のノード単体諸元

システム名	Oakforet-PACS〔9〕	Wisteria/BDEC-01 (Odyssey)〔1〕
略称	OFP	Odyssey
CPU 名称	Intel Xeon Phi 7250 (Knights Landing, KNL)	Fujitsu A64FX（2.2GHz）
コア数	68	48
理論演算性能（GFLOPS）	3,046	3,379
主記憶容量（GB）	MCDRAM: 16 DDR4: 96	32
メモリ性能（GB/sec） STREAM Triad〔10〕	MCDRAM: 490+ DDR4: 84.5	840+
コンパイラ	Intel Parallel Studio 2019	Fujitsu FCC

性能評価は下記の 3 種類のプログラムについて，前処理付き CG 法一反復あたりの計算時間によって実施した：

① オリジナルの GeoFEM/Cube
② ①に通信と計算のオーバーラップ（図 8 の「Static」）を適用した例（Static）
③ ②に動的ループスケジューリング（図 8 の「Dynamic Loop Scheduling」）を適用した例，チャンクサイズを最大 1,000 まで変化させた

1 ノード当たりの問題規模を下記の 3 種類のケースに設定し，ノード数を変化させて Weak Scaling として実施した：

- S：96×96×48 節点（=1,327,104 DOF）
- M：128×128×64 節点（=3,145,728 DOF）
- L：200×200×100 節点（=12,000,000 DOF）

最大問題サイズは 7.3728×10^{10}DOF である。また並列プログラミングモデルとして：

- OFP：HB 8×8（各ノード，8 スレッドの MPI プロセス×8 プロセス），HB 16×4（16 スレッド×4 プロセス）
- Odyssey：HB 6×8（6 スレッド×8 プロセス），HB 12×4（12 スレッド×4 プロセス）

を実施した。図 10，図 11 は，OFP，Odyssey について，128 ノード，1,024 ノードにおけるオリジナルの GeoFEM/Cube に対する性能改善率である。100〜1,000 の数字は動的ループスケジューリングのチャンクサイズである。

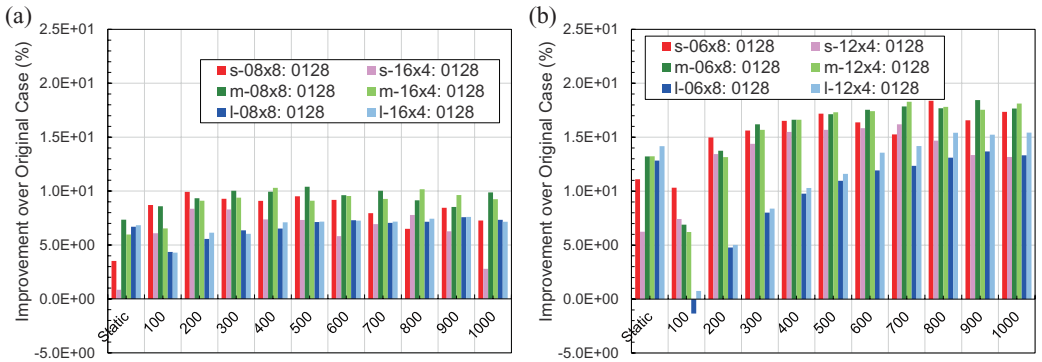

図 10　通信と計算のオーバーラップの効果：オリジナル GeoFEM/Cube に対する前処理付き反復法の速度向上率，Static：古典的オーバーラップ，数字：動的ループスケジューリングにおけるチャンクサイズ，128 ノード（a）OFP，（b）Odyssey

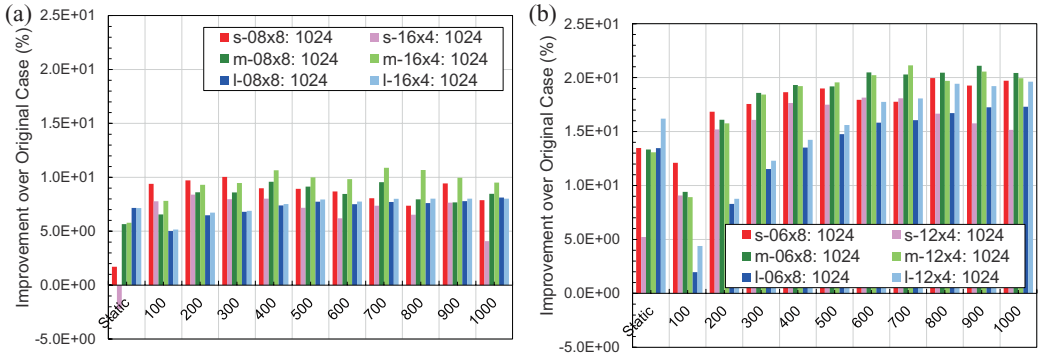

図 11　通信と計算のオーバーラップの効果：オリジナル GeoFEM/Cube に対する前処理付き反復法の速度向上率，Static：古典的オーバーラップ，数字：動的ループスケジューリングにおけるチャンクサイズ，1,024 ノード（a）OFP，（b）Odyssey

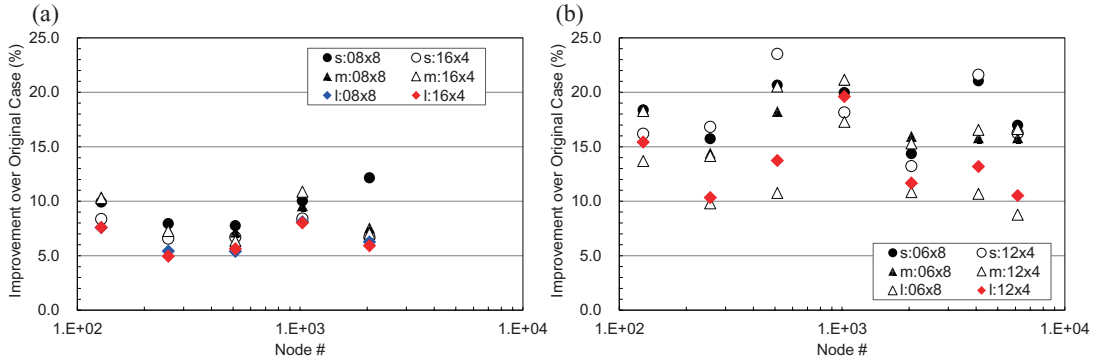

図 12　通信と計算のオーバーラップの効果：各ノード数におけるベストケースのオリジナル GeoFEM/Cube に対する前処理付き反復法の速度向上率，(a) OFP : HB 8×8 Original に対する比，最大 2,048 ノード（b）Odyssey：HB 6×8 Original に対する比，最大 6,144 ノード

OFP は最大 10%程度の速度向上率であるが，Odyssey は 20%程度である。OFP は 128 ノードと 1,024 ノードでは，やや 1,024 ノードの方が速度向上率が高いが，図 12 にも示すように，Odyssey において速度向上率とノード数の関係は必ずしも一様ではない。OFP と比較すると Static による速度向上は Odyssey においてより顕著である。また Odyssey においてチャンクサイズが小さい場合（700 以下）では，ノード当たり問題サイズが大きい場合の速度向上率が比較的低い。OFP，Odyssey ともにノード当たり問題サイズが小さい場合は，HB 8×8，HB 6×8 の性能が良く，大き

い場合は，HB 16×4，HB 12×4 の性能が良くなっている。

図 12 は OFP，Odyssey，各ノードにおけるオリジナルの HB 8×8（OFP），HB 6×8（Odyssey）に対するベストケースにおける速度向上率である。傾向は図 10，図 11 の場合と同じであるが，Odyssey についてはノード数と性能向上の関連が明かでなく，検討が必要である。

図 13 は，1,024 ノードにおける Odyssey（HB 6×8）と OFP（HB 8×8）の性能比である。ノード当たり問題規模が小さい場合，Odyssey は OFP の 2 倍以上，大きい場合でも 1.75 倍程度の計算性能であることがわかる。疎行列計算は memory-bound なプロセスであり，メモリ性能の影響が大きい。表 1 に示すように，Odyssey と OFP のメモリ性能比は STREAM Triad〔10〕で 1.71：1.00 であることを考慮すると，この性能比は妥当な数字である。

図 13 は M サイズ，128 ノード，HB 12×4 において，前処理付き共役勾配法部分の計算時間の内訳を Odyssey 上で詳細プロファイラ〔1〕を使用して測定したものである。動的ループスケジューリングによって，メモリスループットが 29.4%⇒37.8%と約 30%向上しており，計算時間から算定した速度向上率も 30%程度である。これは，メモリ性能の 40%程度しか活用できていない，ということでもあり，全体的な性能向上が必要である。

(a)　(b)

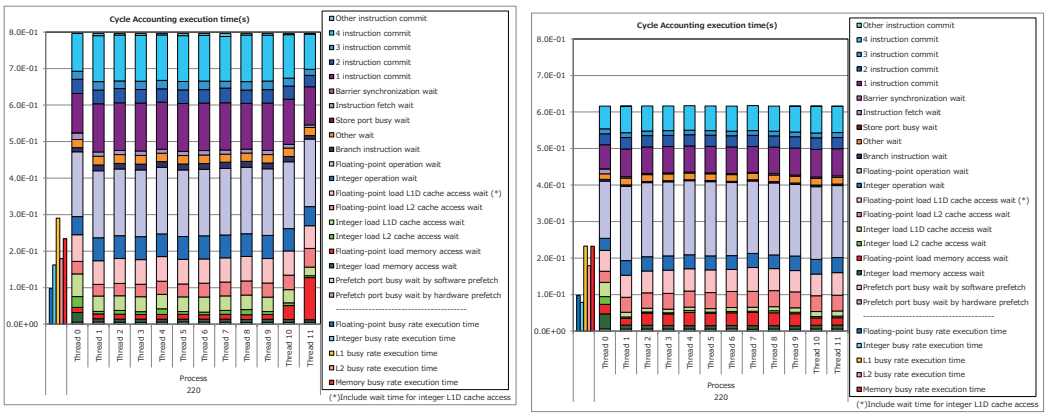

図 13　Odyssey 詳細プロファイラによる 0 番プロセス各スレッドにおける計算時間測定結果（前処理付き共役勾配法部分，サイズ M，128 ノード，HB 12×4），512 プロセスのうち＃220 プロセス，（a）オリジナル（メモリスループット：29.4%），（b）動的ループスケジューリング（チャンクサイズ＝700）（メモリスループット：37.8 %）

5. まとめ

Wisteria/BDEC-01（Oddysey）において，並列有限要素法から導かれる大規模線形方程式の，並列前処理付き反復法による求解プロセスのうち，疎行列ベクトル積演算部分に，通信と計算のオーバーラップ，動的ループスケジューリングを適用することによって，最大 20%以上の速度向上が得られることがわかった。今回は疎行列格納法が CRS であったため，性能は低く，メモリスループットは 40%未満であった。より高性能が期待される，ELL，SELL-C-σ等の手法を適用し，評価する必要がある。また，ノード数が 2,000 を超える場合については，ノード配置も含めた更なる詳細な検討が必要である。

参考文献

[1] Wisteria/BDEC-01: https://www.cc.u-tokyo.ac.jp/supercomputer/wisteria/system.php

[2] Nakajima, K., Hanawa, T., Communication-Computation Overlapping with Dynamic Loop Scheduling for Preconditioned Parallel Iterative Solvers on Multicore/Manycore Clusters, IEEE Proceedings of 10th International Workshop on Parallel Programming Models & Systems Software for High-End Computing (P2S2 2017) in conjunction with the 46th International Conference on Parallel Processing (ICPP 2017), Bristol, UK, 2017

[3] Nakajima, K. Okuda, H., Parallel Iterative Solvers for Unstructured Grids using an OpenMP/MPI Hybrid Programming Model for the GeoFEM Platform on SMP Cluster Architectures, Lecture Notes in Computer Science 2327, 437-448, 2002

[4] Nakajima, K., Parallel Iterative Solvers of GeoFEM with Selective Blocking Preconditioning for Nonlinear Contact Problems on the Earth Simulator, ACM/IEEE Proceedings of SC 2003, 2003

[5] Saad, Y., Iterative Methods for Sparse Linear Systems (2nd Edition), SIAM, 2003

[6] Ghysels, P. and W. Vanroose, Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm, Parallel Computing 40-7, 224-238, 2014

[7] Shimokawabe, T., Aoki, T., Takaki, T., Yamanaka, A., Nukada, A., Endo, T., Maruyama, N., Matsuoka, S., Peta-scale Phase-Field Simulation for Dendritic Solidification on the TSUBAME 2.0 Supercomputer, ACM/IEEE Proceedings of SC'11, 2011

[8] 伊奈拓也，朝比 祐一，井戸村泰宏，テラフロップス級メニーコアアーキテクチャにおけるステンシル計算の最適化手法の開発，情報処理学会研究報告（2015-HPC-152-10），日本情報処理学会第 152 回 HPC 研究会，2015

[9] Oakforest-PACS: https://www.cc.u-tokyo.ac.jp/supercomputer/ofp/system.php

[10] STREAM: https://www.cs.virginia.edu/stream/