Vol.2016-HPC-157 No.6
2016/12/21

パイプライン型共役勾配法の性能評価

塙 敏博[†1] 中島研吾[†1 †2] 大島聡史[†1 †2] 星野哲也[†1] 伊田明弘[†1 †2]

共役勾配法に代表されるクリロフ部分空間法の内積計算プロセスは，大規模並列システム上では性能低下の要因となる．Ghysels 等によって提案されたパイプライン型共役勾配法は，漸化式を使って本来のアルゴリズムを保ちつつ計算順序を変更したもので，MPI-3 でサポートされている非同期集団通信関数を適用することによって，集団通信と計算をオーバーラップさせ，従来手法と比較して高いスケーラビリティを得られることが知られている．本研究では，パイプライン型共役勾配法を使用した三次元有限要素法構造解析コードの Reedbush-U（東京大学情報基盤センター），Oakforest-PACS（最先端共同 HPC 基盤施設）における性能評価事例を紹介する．

Performance Evaluation of Pipelined CG Method

Toshihiro Hanawa[†1] Kengo Nakajima[†1 †2] Satoshi Ohshima[†1 †2]
Tetsuya Hoshino[†1] Akihiro Ida[†1 †2]

Significant communication overhead occurs during dot product operations in Krylov subspace methods, such as Conjugate Gradient Method (CG), on massively parallel supercomputers. Pipelined CG developed by Ghysels et al. applies recurrence relations on original CG algorithm. Although sequence of operations is different from that of original CG, algorithm of the pipelined CG is kept as that of original one. It is widely known that the pipelined CG with asynchronous collective communication supported in MPI-3 standard can hide overhead of collective communications by overlapping communications and computations. In the present work, performance evaluations of 3D finite-element applications for solid mechanics with pipelined CG on Reedbush-U (ITC, University of Tokyo), and Oakforest-PACS (JCAHPC) are demonstrated.

1. はじめに

有限要素法に代表される偏微分方程式の数値解法において，最も計算時間を要するプロセスは大規模な疎行列を係数行列とする連立一次方程式の求解であり，その最適化に向けて様々な試みがなされてきた．連立一次方程式の求解には共役勾配法（Conjugate Gradient，CG）に代表されるクリロフ部分空間法（反復法）が広く使用されているが，大規模並列計算においては通信によるオーバーヘッドが顕著となる．図 1 は連立一次方程式 Ax=b を前処理付き共役勾配法で解くアルゴリズムである：

```
1: r0:= b-Ax0; u0:= M^-1 r0; p0:= u0
2: for i=0 … do
3:     s:= Api
4:     α:= (ri,ui)/(s,pi)
5:     xi+1:= xi + αpi
6:     ri+1:= ri- αs
7:     ui+1:= M^-1 ri+1
8:     β:= (ri+1,ui+1)/(ri,ui)
9:     pi+1:= ui+1 + βpi
10: end do
```

図 1 前処理付き共役勾配法（Preconditioned Conjugate Gradient，PCG）のアルゴリズム（Alg.-1），Ax=b，M：前処理行列，赤字：疎行列ベクトル積（SpMV, Sparse Matrix Vector Multiply），青字：内積，緑字：前処理

分散メモリ型並列計算機上で並列計算を実施する場合は：

- 疎行列ベクトル積
- 内積
- 前処理

で通信が発生する可能性がある．前処理手法によって，通信パターンは異なるが，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，前者は MPI_Isend や MPI_Irecv，後者は MPI_Allreduce のような関数が使用される．

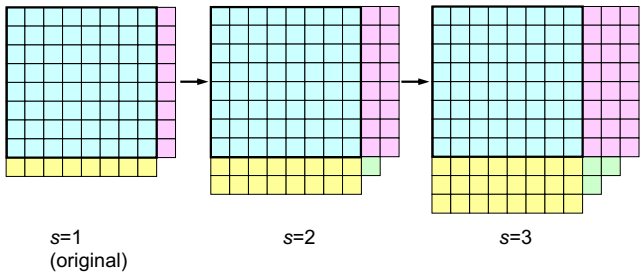


図 2 *s*-step 法のための Halo 領域拡張

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．Matrix Powers Kernel〔1〕に基づく *s*-step 法〔2〕では，*s*（*s*>1）反復分の通信を 1 反復で済ませることができ：

- 内積実施回数削減による集団通信オーバーヘッド削減

†1 東京大学情報基盤センター
Information Technology Center, The University of Tokyo
†2 科学技術振興機構 CREST
CREST, Japan Science and Technology Agency

ⓒ2016 Information Processing Society of Japan

- 並列分散データの Halo 領域を大きくとって，*s* 回分の行列ベクトル積を，通信をしないで連続して実施する（図 2）

ことが可能である．一般に *s* が大きくなると冗長計算が増加するとともに不安定性が増す．また *s*-step 法は，適用可能な前処理手法が限定されるという問題点がある．

また，陽解法で広く使用されている，通信と計算のオーバーラップ〔3〕の手法を疎行列ベクトル積に適用することも可能であるが，通信オーバーヘッドの削減効果をあげることは困難である．

本研究で扱うパイプライン型共役勾配法（Pipelined CG Method）〔4〕は，*s*-step 法の元になったアルゴリズムに基づき，漸化式の適用によって，図 1 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）〔5〕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．

疎行列ベクトル積は，限定された数の隣接プロセスとの通信であるが，集団通信は全プロセスに対する通信であり，大規模並列計算機システムにおいてノード数が増加すると，オーバーヘッドはより顕著となる．図 1 に示すオリジナルの前処理付き共役勾配法では，内積 1 回の通信量はスカラー変数 1 つ分であり，いわゆるレイテンシの効果が大きい．

本研究では，〔4〕に示されたパイプライン型共役勾配法を，三次元有限要素法構造解析コードへ適用し，Reedbush-U（RBU）（東京大学情報基盤センター）〔6〕，Oakforest-PACS（OFP）（最先端共同 HPC 基盤施設）〔7〕を使用して実施した性能評価事例を紹介する．

以下，対象アプリケーション，数値アルゴリズム，計算機環境，問題設定，計算結果について述べる．

2. 対象アプリケーション

2.1 概要

本研究で対象としているのは，GeoFEM プロジェクト〔8,9,10〕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 3））に関する並列前処理付き反復法による疎行列ソルバーの実行時性能を様々な条件下で計測するものである．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．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 4）．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．

プログラムは全て OpenMP ディレクティヴを含む Fortran90 および MPI で記述されている． GeoFEM で採用されている局所分散データ構造〔8,9,10〕を使用しており，領域分割された各領域に MPI プロセスが割り当てられる．MPI，OpenMP，Hybrid（OpenMP＋MPI）の全ての環境で稼動する．

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，各 MPI プロセスの扱う領域に対して，局所化されたブロックヤコビ型 Symmetric Gauss Seidel（SGS）法を適用している（図 5）〔8,9,10〕．

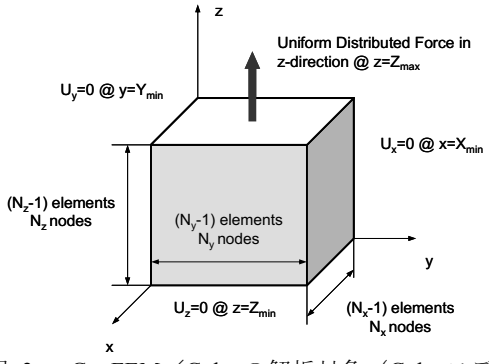


図 3　GeoFEM／Cube の解析対象（Cube モデル）

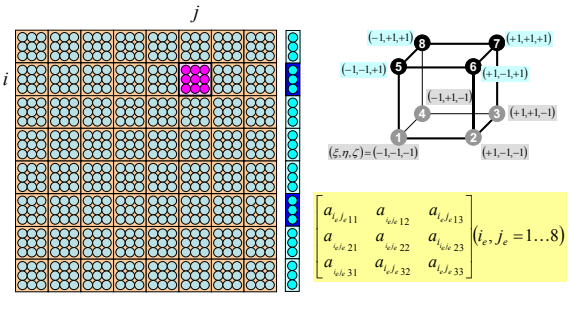


図 4　8 節点六面体要素，1 節点 3 自由度の処理

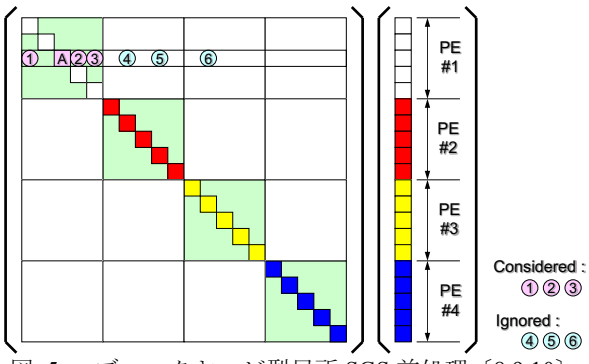


図 5　ブロックヤコビ型局所 SGS 前処理〔8,9,10〕

2.2 色づけによるリオーダリング

SGS 前処理は，前進代入，後退代入のプロセスでメモリへの書き込みと参照が同時に生じ，データ依存性が発生する可能がある．これを回避するための方法として色づけ（coloring）によるリオーダリング（reordering）が広く使用されている〔11,12〕．お互いに依存性を持たない要素群を同じ色に色づけすることによって，色内での並列処理が可能となる．

本研究では，並列性に優れたマルチカラー法（Multicoloring，MC）とより安定した収束を示す Reverse Cuthill-McKee （RCM）法を組み合わせ，RCM 法に Cyclic マルチカラー法（Cyclic Multicoloring，CM）を適用した CM-RCM(k)法を使用している〔11,12〕（一部のケースでは RCM 法を使用している）．図 6 は CM-RCM(k)法による並び替え例である．ここでは，4 色に色分けされており（CM-RCM(4)），たとえば，RCM の第 1，第 5，第 9，第 13 組の要素群が CM-RCM(k)法の第 1 色に分類される．各色には 16 の要素が含まれる．CM-RCM(k)法における色数は，各色内の要素が依存性を持たない程度に大きい必要がある．

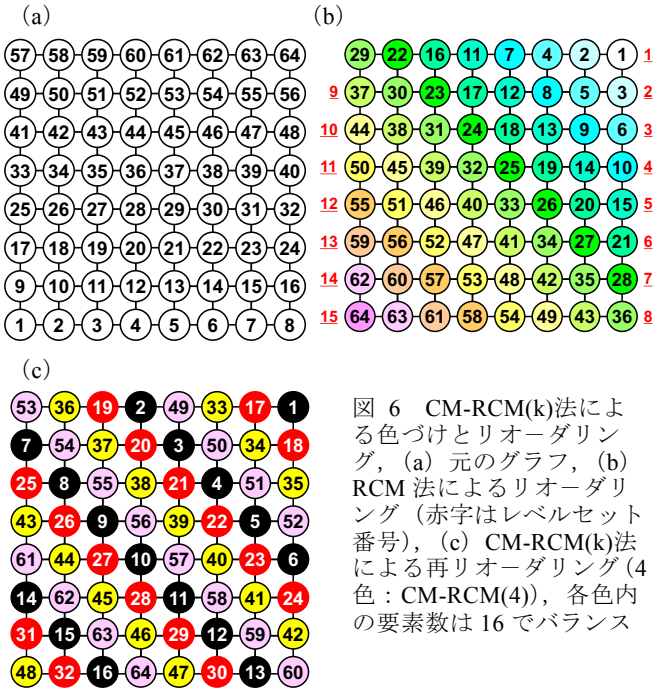


図 6 CM-RCM(k)法による色づけとリオーダリング，(a) 元のグラフ，(b) RCM 法によるリオーダリング（赤字はレベルセット番号），(c) CM-RCM(k)法による再リオーダリング (4 色：CM-RCM(4))，各色内の要素数は 16 でバランス

2.3 加法型 Schwartz 領域分割法

局所 SGS 前処理法は，領域分割数を増加させると，反復回数が増加する傾向にあるため，収束を安定化させる手法として，領域間オーバーラップ領域に加法型 Schwartz 領域分割法（Additive Schwartz Domain Decomposition：ASDD）〔13〕を適用している．ASDD の手順は以下の通りである：

（1） 共役勾配法の前処理の計算を実施する（図 1 参照）：

$$u = M^{-1} r$$

M：前処理行列，r,z：ベクトル，である．前処理計算 M^{-1} は SGS 法の前進・後退代入に相当する

（2） 全体領域が図 7 に示すように Ω_1 および Ω_2 の 2 領域に分かれているものと仮定すると前処理（前進・後退代入）は各領域において，以下のように局所的に実施される．

$$u_{\Omega_1} = M_{\Omega_1}^{-1} r_{\Omega_1}, \quad u_{\Omega_2} = M_{\Omega_2}^{-1} r_{\Omega_2}$$

（3） 局所的な前処理を実行したのちに，領域間オーバーラップ領域 Γ_1 および Γ_2 において以下の計算を実施する（図 7（b））：

$$u_{\Omega_1}^{n} = u_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1}(r_{\Omega_1} - M_{\Omega_1} u_{\Omega_1}^{n-1} - M_{\Gamma_1} u_{\Gamma_1}^{n-1})$$
$$u_{\Omega_2}^{n} = u_{\Omega_2}^{n-1} + M_{\Omega_2}^{-1}(r_{\Omega_2} - M_{\Omega_2} u_{\Omega_2}^{n-1} - M_{\Gamma_2} u_{\Gamma_2}^{n-1})$$

ここで n は ASDD の繰り返し数である。

（4） （2）および（3）のプロセスを収束まで繰り返す。

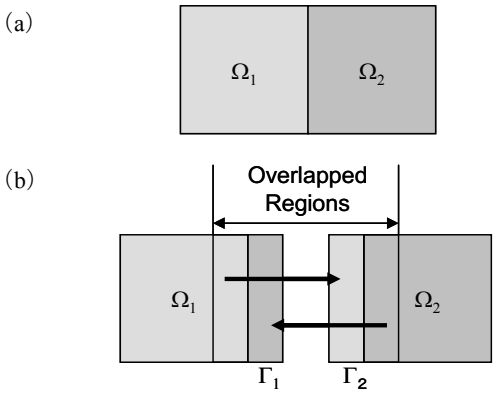


図 7 二領域間における加法型 Schwartz 領域分割法（Additive Schwartz Domain Decomposition，ASDD）処理の概要〔13〕：(a) 局所処理，(b) 大域的修正

3. パイプライン型共役勾配法

本研究では，以下の 4 種類のアルゴリズムを使用する：

- オリジナル前処理付き共役勾配法（Alg.-1）（図 1）
- Chronopoulos/Gear アルゴリズム（Alg.-2）〔2,4〕
- パイプライン型アルゴリズム（Alg.-3）〔4〕
- Gropp アルゴリズム（Alg.-4）〔14〕

Chronopoulos/Gear アルゴリズム（Alg.-2）は s-step 法〔2〕において s=1 としたもので，下記のような漸化式（1）：

$$\begin{aligned} &s_i = Au_i + \beta_i s_{i-1},\ p_i = u_i + \beta_i p_i \Leftrightarrow s_i = Ap_i \\ &x_{i+1} = x_i + \alpha_i p_i \\ &r_{i+1} = b - Ax_{i+1} = b - Ax_i - \alpha_i Ap_i \\ &\quad = r_i - \alpha_i Ap_i = r_i - \alpha_i s_i \end{aligned} \qquad (1)$$

を使用して，$s_i = Ap_i$ を陽に計算することなく求めている．結果として，図 8 に示すようなアルゴリズムが得られる．

このアルゴリズムの特徴は，γ_i，δという内積処理を連続して実施できることにある（図 8 の 10 行目，11 行目）．したがって，実装上は MPI_Allreduce を 1 回呼ぶだけで済むため，全 MPI プロセスが関わる集合通信の回数を 1 回削減することができる：

```
1: r0:= b-Ax0; u0:= M-1r0; w0:= Au0
2: α0:= (r0,u0)/(w0,u0); β0:=0; γ0:= (r0,u0)
3: for i=0 … do
4:     pi:= ui + βi pi-1
5:     si:= wi + βi si-1
6:     xi+1:= xi + αi pi
7:     ri+1:= ri- αi si
8:     ui+1:= M-1ri+1
9:     wi+1:= Aui+1
10:    γi+1:= (ri+1,ui+1)
11:    δ:= (wi+1,ui+1)
12:    βi+1:= γi+1/ γi
13:    αi+1:= γi+1/ (δ - βi+1 γi+1/αi)
14: end do
```

図 8　Chronopoulos/Gear アルゴリズム（Alg.-2）〔2,4〕

Alg.-2 に以下の漸化式（2）を適用すると，図 9 に示すアルゴリズムが得られる．このアルゴリズムはパイプライン型 Chronopoulos/Gear アルゴリズム（前処理無し）である．

$$
\begin{aligned}
&u_i = r_i,\ w_i = Au_i = Ar_i \\
&Ar_{i+1} = Ar_i - \alpha_i As_i \Rightarrow w_{i+1} = w_i - \alpha_i As_i \\
&As_i = Aw_i + \beta_i As_{i-1} \Rightarrow z_i \left(= As_i = A^2 p_i\right) = Aw_i + \beta_i z_{i-1} \\
&q_i = Aw_i = A^2 r_i \\
&\quad = r_i - \alpha_i Ap_i = r_i - \alpha_i s_i
\end{aligned}
\qquad (2)
$$

このアルゴリズムの特徴は，Alg.-2 と同様に γ_i，δ という内積処理を連続して実施できるだけでなく（図 9 の 3 行目，4 行目），更にこれらの内積の値は 5 行目の疎行列ベクトル積では使用されず，6 行目以降で使用されるということである．この部分に，MPI-3 でサポートされている非同期集団通信機能を適用すれば，内積計算のための集合通信と疎行列ベクトル積をオーバーラップさせることが可能となる．

```
1: r0:= b-Ax0; w0:= Au0
2: for i=0 … do
3:     γi:= (ri,ri)
4:     δ:= (wi,ri)
5:     qi+1:= Awi+1
6:     if i>0 then
7:     βi:= γi/ γi-1; αi:= γi/ (δ - βi γi/αi-1)
8:     else
9:     βi:= 0; αi:= γi/δ
10:    end if
11:    zi:= qi + βi zi-1
12:    si:= wi + βi si-1
13:    pi:= ri + βi pi-1
14:    xi+1:= xi + αi pi
15:    ri+1:= ri - αi si
16:    wi+1:= wi - αi zi
17: end for
```

図 9　パイプライン型 Chronopoulos/Gear アルゴリズム（前処理無し）〔4〕

更に，これに以下に示す漸化式（3）を適用すると，図 10 に示すパイライン型前処理付き共役勾配法のアルゴリズム（Alg.-3）〔4〕が得られる：

$$
\begin{aligned}
&\gamma_i = (u_i,u_i)_M = (Mu_i,u_i) = (r_i,u_i) \\
&\delta_i = (M^{-1}Au_i,u_i)_M = (Au_i,u_i) = (w_i,u_i) \\
&M^{-1}r_{i+1} = M^{-1}r_i - \alpha_i M^{-1}s_i \Rightarrow u_{i+1} = u_i - \alpha_i q_i \quad \left(q_i = M^{-1}s_i\right) \\
&M^{-1}s_{i+1} = M^{-1}w_i + \beta_i M^{-1}s_i \Rightarrow q_{i+1} = M^{-1}w_i + \beta_i q_i \\
&Au_{i+1} = Au_i - \alpha_i Aq_i \Rightarrow w_{i+1} = w_i - \alpha_i Aq_i \\
&Aq_i = AM^{-1}w_i + \beta_i Aq_{i-1} \Rightarrow z_i = Am_i + \beta_i z_{i-1} \\
&\quad \left(m_i = M^{-1}w_i = M^{-1}Au_i = M^{-1}AM^{-1}r_i,\ z_i = Aq_i\right)
\end{aligned}
\qquad (3)
$$

```
1: r0:= b-Ax0; u0:= M-1r0; w0:= Au0
2: for i=0 … do
3:     γi:= (ri,ui)
4:     δ:= (wi,ui)
5:     mi:= M-1wi
6:     ni:= Ami
7:     if i>0 then
8:     βi:= γi/ γi-1; αi:= γi/ (δ - βi γi/αi-1)
9:     else
10:    βi:= 0; αi:= γi/δ
11:    end if
12:    zi:= ni + βi zi-1
13:    qi:= mi + βi qi-1
14:    si:= wi + βi si-1
15:    pi:= ui + βi pi-1
16:    xi+1:= xi + αi pi
17:    ri+1:= ri - αi si
18:    ui+1:= ui - αi qi
19:    wi+1:= wi - αi zi
20: end for
```

図 10　パイプライン型（前処理付き）共役勾配法のアルゴリズム（Alg.-3）〔4〕

このアルゴリズムも Alg.-2，図 9 に示すアルゴリズムの特性を継承しており，内積（3 行目，4 行目）の後に前処理，疎行列ベクトル積の計算があり，その後初めて内積の値を使用するため，集団通信と計算のオーバーラップが可能である．この他〔4〕で紹介されている Gropp のアルゴリズム（Alg.-4）〔14〕は，Alg.-3 と良く似たアルゴリズムであるが，δ の導出法が異なっており，計算量も若干少なくなっている（図 11）．

```
1: r0:= b-Ax0; u0:= M-1r0; p0:= u0; s0:= Ap0;
γ0:= (r0,u0)
2: for i=0 … do
3:     δ:= (pi,si)
4:     qi:= M-1si
5:     αi:= γi/δ
6:     xi+1:= xi + αi pi
7:     ri+1:= ri - αi si
8:     ui+1:= ui - αi qi
9:     γi+1:= (ri+1,ui+1)
10:    wi+1:= Aui+1
11:    βi+1:= γi+1/ γi
12:    pi+1:= ui+1 + βi+1 pi
13:    si+1:= wi+1 + βi+1 si
14: end for
```

図 11　Gropp アルゴリズム〔4,14〕

ⓒ2016 Information Processing Society of Japan

図 12 は Gropp アルゴリズム(図 11)の 3～5 行目を, Fortran, OpenMP, MPI で実装した例である．内積計算用集団通信関数として MPI_Allreduce ではなく MPI-3 でサポートされている非同期集団通信関数 MPI_Iallreduce を呼んでいる．内積の計算結果（δ）を使用する直前に MPI_Wait を呼んで同期しているため，集団通信と前処理計算部（4 行目）をオーバーラップすることが可能である．

表 1 は各アルゴリズムの計算量である．DAXPY はベクトルの定数倍の加減（例：$x_i = x_i + \alpha_i p_i$）である．Alg.3, Alg.4 はオリジナルの手法と比較して，計算量が増えているが，SpMV や前処理と比較すると計算量は少なく，無視できる．

Alg.1～Alg.4 はアルゴリズム的には同じ計算内容であるが，計算順序が変わっているため，丸め誤差の伝播の仕方が異なっている．従って，悪条件問題の場合，アルゴリズムによって収束の履歴が異なる場合がある．〔4〕では，そのような事例が実際にあるため，50 反復に一回残差ベクトル（r_i）の値を $r_i = b - Ax_i$ によって補正している．本研究で対象としている問題では，そのような補正は不要で，各アルゴリズム全く同じ履歴，同じ回数で収束した．

```
!C> 3: δ:= (p_i,s_i)
      DLO= 0.d0
!$omp parallel do private(i) reduction(+:DLO)
      do i= 1, 3*N
        DLO= DLO + P(i)*S(i)
      enddo
      call MPI_Iallreduce                              &
     &    (DLO, Delta, 1, MPI_DOUBLE_PRECISION,        &
     &     MPI_SUM, MPI_COMM_WORLD, request, err)

!C> 4: q_i:= M^-1 s_i
   (前処理：省略)

   call MPI_Wait (request, status, err)

!C> 5: α_i:= γ_i/δ
   Alpha= Gamma / Delta
```

図 12　Gropp アルゴリズムの実装例（非同期集団通信と計算のオーバーラップ）

表 1　各アルゴリズムの計算量(SpMV:疎行列ベクトル積, DAXPY：ベクトル定数倍加減)，内積の（+1）は残差ベクトルのノルム計算

Alg.		SpMV	前処理	内積	DAXPY
1	Original CG	1	1	2+1	3
2	Chronopoulos/Gear	1	1	2+1	4
3	Pipelined CG	1	1	2+1	8
4	Gropp	1	1	2+1	5

4. 計算機環境

本研究では以下の 2 種類の計算機環境を使用した：

- RBU：Reedbush-U（東京大学情報基盤センター）〔6,7〕
- OFP：Oakforest-PACS（最先端共同 HPC 基盤施設）〔8〕

Reedbush-U の計算ノードは，CPU として Intel Xeon E5-2695v4 (Broadwell-EP)を 2 ソケット搭載している．CPU の各コアには，AVX2 命令に対応したベクトルユニットが 2 基ずつ搭載されている．これらの計算ノード 420 ノードは，InfiniBand-EDR によりフルバイセクションバンド幅を持つ Fat-tree 網で接続されている[7].

一方, Oakforest-PACS の計算ノードは, Intel Xeon Phi 7250 (Knights Landing: KNL) 1 ソケットで構成され，それまでの Xeon Phi x100 シリーズ (Knights Corner)がアクセラレータとして Xeon プロセッサなどと PCIe 接続される必要があったのに対し，それ自身がブート可能な，メニーコアシステムである．各コアには AVX512 命令に対応したベクトルユニットが 2 基ずつ搭載されている．これらのコア 2 つで 1 つのタイルを構成し，これらのタイルがメモリコントローラ等と併せてチップ内部で 2 次元メッシュ状に接続されている．

KNL の CPU 上には MCDRAM と呼ばれる 3 次元積層メモリが搭載され，DDR4 メモリに比べて 5 倍以上のメモリバンド幅を持つ．KNL は，この MCDRAM を利用するためのモードとして，以下のメモリモード，クラスタリングモードを備えており，BIOS によって切り替えることができる．

- メモリモード
  1. Flat：MCDRAM に対して DDR4 とは異なるメモリアドレスが与えられ，明示的にアクセスが可能になる．memkind ライブラリにより，プログラム中で C 言語では hbw_malloc 関数，Fortran では FastMem 指示子によって動的に確保するか，プログラム実行時に numactl などによって MCDRAM に対して bind することによって利用可能になる．
  2. Cache: MCDRAM を DDR4 メモリのキャッシュのように扱う．プログラム中から明示的に MCDRAM をアクセスすることはできない．KNL では Direct Map による L3 キャッシュとして振る舞う．
  3. Hybrid: MCDRAM の容量を分割して，Flat モードと Cache モードの両方が利用可能になる．
- クラスタリングモード

  MCDRAM のメモリコントローラは, KNL チップを上下左右に分割した各象限上に合計 4 チャネル搭載されており, チップ上のリソースを各象限に 4 分割して考えるのが自然な設計になっている．ここでは，それを踏まえて Quadrant モードと SNC-4 モードについて説明する．
  1. Quadrant: 全体を 1 ソケットの CPU のように扱うが, 内部的なメモリアクセスのトラフィックとしては各象限に閉じたローカルなアクセスになるようなアドレッシングになる．
  2. SNC-4: 各象限が独立した NUMA ドメインとして扱われ，プログラムからは，あたかも 4 ソケットシス

ⓒ2016 Information Processing Society of Japan

テムのように見える．これをサブ NUMA クラスタリングと呼ぶ．適切なアフィニティ制御を行うことで Quadrant よりも高い性能が期待できる．

OFP では，OS ジッタの影響を極力避けるため，タイマ割り込みを受け付けないコアを指定する，full tickless の設定を行っている[9]．タイマ割り込みを受け付けるコアは 0 に設定されているが，さらに L2 キャッシュへの影響を考慮して，同一タイル中で L2 キャッシュを共有するコア 1 も避けて割り当てをすることで，性能を大きく改善することができる．

プログラムは Fortran90 で記述してあり， Intel Comliler（Ver.17）／Intel Parallel Studio XE 2017 を使用し，ノード内スレッド並列化には OpenMP を使用している．表 2 に計算機環境の概要を示す．

MPI にはいずれも Intel Parallel Studio XE 2017 に含まれる Intel MPI を使用した．計算ノード間インタコネクトは，RBU では Mellanox InfiniBand-EDR, OFP では Intel OmniPath Architecture を用いているが, 共に 100 Gbps リンクであり，フルバイセクションバンド幅を持つ Fat-tree 網で構成されている．

表 2　各計算環境の概要

システム名	Reedbush-U（RBU）	Oakforest-PACS（OFP）
CPU	Intel Xeon E5-2695 v4 (Broadwell-EP)	Intel Xeon Phi 7250 (Knights Landing)
動作周波数（GHz）	2.10	1.40
コア数（最大有効スレッド数）	18 (18)	68 (272)
理論演算性能（GFLOPS）	604.8	3,046.4
主記憶容量（GB）	128	MCDRAM: 16 DDR4: 96
メモリバンド幅性能（GB/sec., Stream Triad）	65.5	MCDRAM: 490 DDR4: 84.5
インタコネクト	Mellanox InfiniBand EDR 4x (100 Gbps) Full-bisection BW Fat-tree	Intel OmniPath Architecture (100 Gbps) Full-bisection BW Fat-tree

注）Oakforest-PACS におけるメモリバンド幅性能は、メモリモード Flat, サブ NUMA クラスタリングモード Quadrant における性能である．

5. 計算結果（RBU）

5.1 概要

Reedbush-U（RBU）を使用した計算の概要は表 3 の通りである．1 ソケットの 18 コアのうち 16 コアを使用し，最小 2 ノード（4 ソケット, 64 コア）から最大 384 ノード（768 ソケット，12,288 コア）までの Strong Scaling 性能を評価した．

表 3　Reedbush-U における計算の概要

問題規模	Small：256×128×144 節点（14,155,776 DOF） Medium：256×128×288 節点（28,311,552DOF）
利用ノード数	最小 2 ノード，4 ソケット 最大 384 ノード，768 ソケット 1 ソケットあたり 16 コア使用，最大 12,288 コア Small : コア当り最小問題サイズ＝1,152 DOF/core Medium：同＝2,304 DOF/core
並列プログラミングモデル	Flat MPI：1 コアあたり 1MPI プロセス Hybrid : 1 ソケットあたり 1MPI プロセス，16 スレッド
ノード内リオーダリング法	RCM
ASDD 反復数	2 回
CG 法収束条件	$\|r\|/\|b\|= \|b-Ax\|/\|b\| < 1.0\times10^{-8}$

5.2 計算結果

まず，図 13 にコア数と反復回数の関係を示す．ブロックヤコビ型局所 SGS 前処理を適用しているため，コア数が増加すると反復回数が増加する傾向があるものの，ASDD（加法型 Schwartz 領域分割法）の効果によって Hybrid の場合は 64 コアから 12,288 コアまで 10%程度の反復回数増加に留まっているが，Flat MPI の場合は 20%以上となっている．図 14 は Hybrid の 384 ノード，12,288 コアまでの速度向上率である．Flat MPI　2 ノード（4 ソケット, 64 コア）の場合の性能を 64.0 としている．Small，Medium のいずれの場合にも Alg.1 の速度向上率が低く，Alg.2 がややそれより良く，Alg.3，Alg.4 は更に良く，両者の速度向上率はほぼ同じである．Medium での Alg.3，Alg.4 の速度向上率は 12,888 コアで 9,000 を超えており，約 75%の並列化効率が得られている．

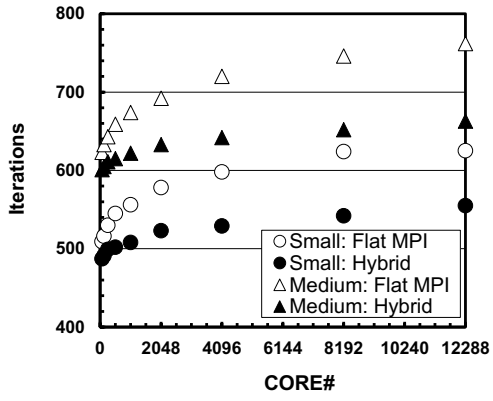
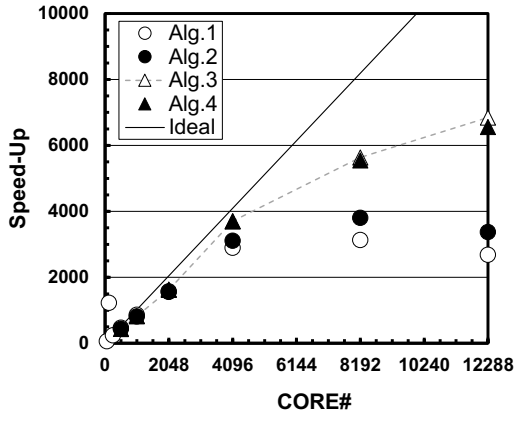


図 13　CG 法の反復回数とコア数の関係

(a)



ⓒ2016 Information Processing Society of Japan

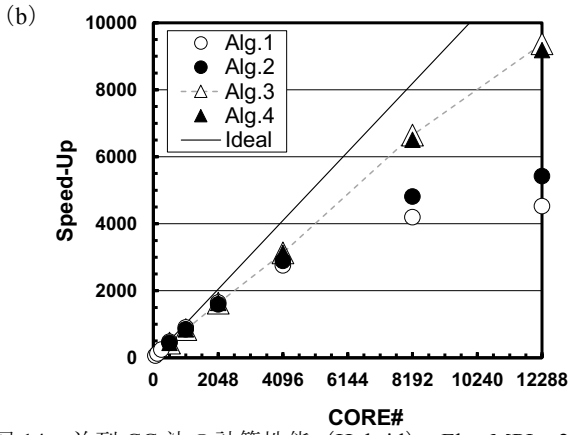


図 14　並列 CG 法の計算性能（Hybrid），Flat MPI　2 ノード（4 ソケット，64 コア）の場合の性能を 64.0 としたときの速度向上率（a）Small，（b）Medium

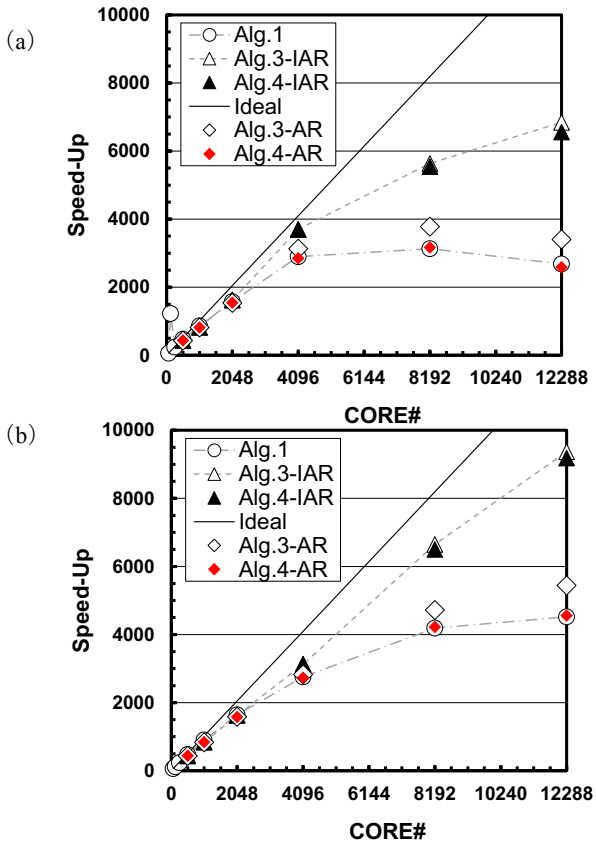


図 15　並列 CG 法の計算性能（Hybrid），Flat MPI　2 ノード（4 ソケット，64 コア）の場合の性能を 64.0 としたときの速度向上率（a）Small，（b）Medium, IAR：MPI_Iallreduce 使用，AR：MPI_Reduce 使用

図 15 は Alg.3，Alg.4 において非同期集団通信関数（MPI_Iallreduce）を使った場合（IAR）と通常の MPI_Allreduce を使った場合（AR）と Alg.1 を比較したものである．非同期集団通信関数を使用しない場合は，Alg.1 とほぼ同じ挙動を示していることがわかり，非同期集団通信機能の適用により，通信と計算のオーバーラップを実現し，大幅な性能向上が得られていることがわかる．

図 16 は，Alg.4 における Hybrid と Flat MPI の性能比（CG 法の計算時間の比，1 より大きいと Hybrid の方が速い）である．4,096 コアまでは両者の計算時間はほぼ同じであるが，コア数が増大するほどその差は大きくなり，12,288 コアでは Small の場合 4:1, Medium の場合 2:1 となっている．

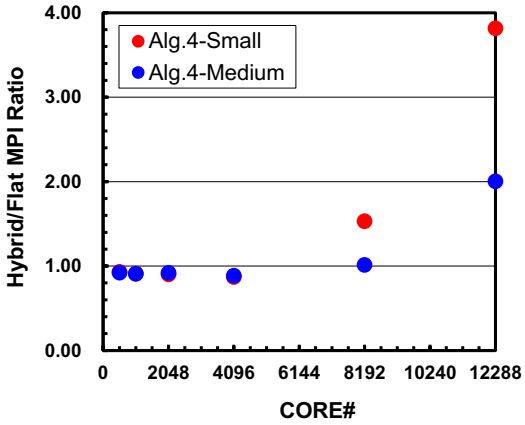


図 16　Alg.4 における Hybrid と Flat MPI の性能比（CG 法の計算時間の比，1 より大きいと Hybrid の方が速い）

6. 計算結果（OFP）

6.1 概要

Oakforest-PACS を使用した計算の概要は表 4 の通りであり，特に記載のない項目は RBU と共通で，表 3 に示した通りである．1 ノード 68 コアのうち，2 コアは OS サービスのために明示的に割り当てから除外し，残りから 64 コアを用いて各メモリモード，クラスタリングモードにおける性能を比較した．

ここで，SNC4 では，ノードにおけるサブ NUMA ごとに MPI プロセスを割り当て，各プロセス 16 スレッドで実行した．　Quadrant では，MPI プロセスに順に 16 スレッドずつ割り当てた．

コードは RBU のものと同一の物を用い，コンパイル時の最適化オプションについては Knights Landing 向けに変更した．

表 4　Oakforest-PACS における計算の概要

問題規模	Small：128×128×48 節点（2,359,296 DOF）
利用ノード数	128 ノード 1 ノードあたり 64 コア使用，最大 8,192 コア Small：コア当り最小問題サイズ＝288 DOF/core
メモリモードとクラスタリングモード	Flat-Quadrant, Flat-SNC4 Cache-Quadrant, Cache-SNC4
並列プログラミングモデル	Hybrid：1 ノードあたり 4MPI プロセス，16 スレッド
ノード内リオーダリング法	CM-RCM(10)

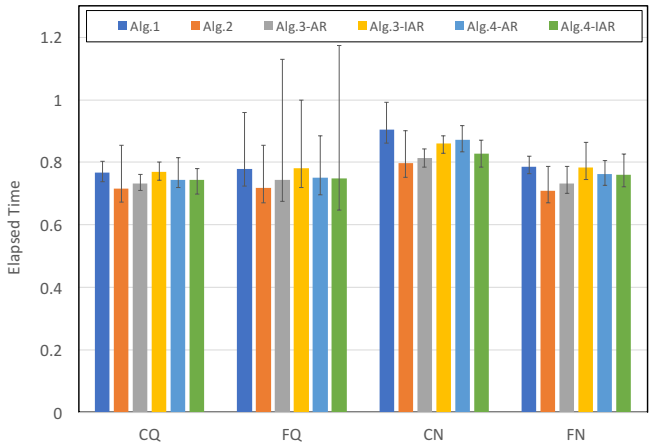


図 16 並列 CG 法の 10 回の試行における平均計算時間. CQ: Cache-Quadrant, FQ: Flat-Quadrant, CN: Cache-SNC4, FN: Flat-SNC4 での実行結果を示す．エラーバーは，試行中の最大値・最小値を示す．

各アルゴリズムで比較すると，OFP では Alg.2 が最も性能が高い傾向がある．一方，RBU では大きな改善が見られていた，Alg.3，Alg.4 における同期 collective・非同期 collective の差がほとんど見られない．これは，OmniPath Architecture では，InfiniBand EDR に比較して，CPU コアへの処理の依存度が大きく，同期・非同期の差が現れにくいのではないか，と考えられる．

メモリモード，クラスタリングモードの比較では，

- Flat-SNC4 の性能が最も高いが，Cache-Quadrant, Flat-Quadrant の性能と大きく変わらない．
- Cache-SNC4 については性能低下が見られる．これに関して，Quadrant であまり差が見られないことから考えると，本来は Flat-SNC4 程度の結果が得られることが見込まれる．
- Flat-Quadrant で性能のばらつきが大きい．

という傾向が見られる．

RBU の同規模の実行時間と比較すると，非同期 collective を使わなかった場合には，最大で 3 倍程度 OFP の性能が高いことが分かった．

7. まとめ

共役勾配法に代表されるクリロフ部分空間法の内積計算プロセスは，大規模並列システム上では性能低下の要因となる．Ghysels 等によって提案されたパイプライン型共役勾配法は，漸化式を使って本来のアルゴリズムを保ちつつ計算順序を変更したもので，MPI-3 でサポートされている非同期集団通信関数を適用することによって，集団通信と計算をオーバーラップさせ，従来手法と比較して高いスケーラビリティを得ることができる．本研究では，パイプライン型共役勾配法を三次元有限要素法構造解析コードに適用し，Reedbush-U（東京大学情報基盤センター），Oakforest-PACS（最先端共同 HPC 基盤施設）によって性能評価を実施した．

Reedbush-U では，最大 384 ノード，12,288 コアを使用した評価を実施し，256 ノード（8,192）を超えた場合にパイプライン型共役勾配法の優位性が示され，384 ノードでは従来手法と比較して約 4 倍の性能向上が得られた．非同期集団通信関数の適用による集団通信と計算のオーバーラップの効果は顕著である．今後は，Reedbush-U で利用可能な様々な MPI の実装を適用した評価を継続して実施する予定である．

Oakforest-PACS では，128 ノード，8,192 コアを使用した評価を実施した．今回の評価では，パイプライン型共役勾配法の効果が見られなかった．今後は，非同期 collective における詳細な挙動を調査するとともに，ノード数，問題規模を変化させたときの性能向上率についても調査を行う予定である．

謝辞

本研究の一部は，東京大学情報基盤センターと筑波大学計算科学研究センターの協力のもとに最先端共同 HPC 基盤施設（JCAHPC）が共同運用する Oakforest-PACS を用いて実施されました．JCAHPC メンバーの皆様に感謝いたします．

参考文献

1) Demmel, J., Hoemmen, M., Marghoob, M., and Yelick, K., Avoiding Communication in Sparse Matrix Computations, IEEE Proceedings of 22nd IEEE International Parallel & Distributed Processing Symposium (IPDPS 2008) (2008)
2) Chronopoulos, A.T., and Gear, C.W., *s*-step iterative methods for symmetric linear systems, Journal of Computational and Applied Mathematics 25-2, 153-168 (1989)
3) Shimokawabe, T., Aoki, T., Takaki, T., Endo, T., Yamanaka, A., Maruyama, N., Nukada, A., and Matsuoka, S., Peta-scale phase-field simulation for dendritic solidification on the TSUBAME 2.0 supercomputer, ACM/IEEE Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11) (2011)
4) Ghysels, P., and Vanroose, W., Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm, Parallel Computing 40, 224-238 (2014)
5) MPI: A Message-Passing Interface Standard Version 3.0, Message Passing Interface Forum, http://mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf (2012)
6) 東京大学情報基盤センター（スーパーコンピューティング部門），http://www.cc.u-tokyo.ac.jp/
7) 塙敏博，中島研吾，大島聡史，伊田明弘，星野哲也，田浦健次朗，データ解析・シミュレーション融合スーパーコンピュータシステム Reedbush-U の性能評価，情報処理学会研究報告（HPC-156） (2016)

ⓒ2016 Information Processing Society of Japan

8) 最先端共同 HPC 基盤施設，http://jcahpc.jp/
9) RedHat Enterprise Linux Performance Tuning Guide, https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Performance_Tuning_Guide/index.html
10) GeoFEM：並列有限要素法による固体地球シミュレーションプラットフォーム，http://geofem.tokyo.rist.or.jp
11) Nakajima, K., Parallel Iterative Solvers of GeoFEM with Selective Blocking Preconditioning for Nonlinear Contact Problems on the Earth Simulator, ACM/IEEE Proceedings of SC2003, (2003)
12) 中島研吾，片桐孝洋，マルチコアプロセッサにおけるリオーダリング付き非構造格子向け前処理付反復法の性能，情報処理学会研究報告（HPC-120）（2009）
13) 中島研吾，拡張型 Sliced-ELL 行列格納手法に基づくメニィコア向け疎行列ソルバー，情報処理学会研究報告（HPC-147）（2014）
14) Washio, T., Maruyama, K., Osoda, T., Shimizu, F., and Doi, S., Efficient implementations of block sparse matrix operations on shared memory vector machines. Proceedings of The 4th International Conference on Supercomputing in Nuclear Applications (SNA2000) (2000)
15) Smith, B., Bjφrstad, P. and Gropp, W., *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press (1996)
16) Gropp, W., Update on libraries for Blue Waters, (2010) http://jointlab.ncsa.illinois.edu/events/workshop3/pdf/presentations/Gropp-Update-on-Libraries.pdf>.

ⓒ2016 Information Processing Society of Japan