Vol.2019-HPC-170 No.19
2019/7/25

AM-hCGA 法による並列多重格子法

中島研吾[†1 †2 a)] 堀越将司[†3] Balazs Gerofi[†2] 石川 裕[†2]

並列多重格子法は，エクサスケールシステムにおける大規模問題向け数値解法として注目されている．著者等の提案した Hierarchical Coarse Grid Aggregation（hCGA）法は，MPI プロセス数が 10^4 以上の場合，並列多重格子法ソルバの性能を劇的に向上させるが，著者による先行研究では，階層化のレベル数が 2 であるため，10^5-10^6 プロセス以上の場合には，Coarse Grid Solver による計算オーバーヘッドが顕著となる可能性がある．本研究では，3 レベル以上の多階層を考慮できる AM-hCGA 法(Adaptive Multilevel hCGA)を提案し，Oakforest-PACS（JCAHPC）使用した計算例を紹介する．また，軽量カーネル IHK/McKernel の効果についても検討を実施した．

1. はじめに

並列多重格子法は，エクサスケールシステムにおける大規模問題向け数値解法として注目されている．著者等による先行研究〔1〕では，Sliced Ellpak-Itpack（ELL）〔2〕に基づく新しい疎行列格納手法によりメモリアクセスの最適化を実現した．また Hierarchical Coarse Grid Aggregation（hCGA）法〔1〕の導入により並列多重格子法における Coarse Grid Solver の最適化を実現した．これらの手法は，三次元不均質多孔質媒体中の地下水流れを有限体積法によって解くプログラム pGW3D-FVM〔1〕のポアソン方程式求解部の多重格子法前処理付き共役勾配法ソルバ(MGCG)に適用され，Fujitsu PRIMEHPC FX10 スーパーコンピュータ（Oakleaf-FX，東京大学情報基盤センター）〔3〕，4,096 ノード（65,536 コア）において，弱スケーリングで約 1.61 倍，強スケーリングで約 6.27 倍の速度向上が得られている〔1〕. hCGA では 2 レベルの並列階層性の導入により，MPI プロセス数が 10^4 以上の場合においても高い並列性能を達成可能であるが，エクサスケール時代において，10^5-10^6 プロセス以上となる場合には，Coarse Grid Solver〔1〕による計算オーバーヘッドが顕著となる可能性がある．本研究では，3 レベル以上の並列階層性を導入した Adaptive Multilevel hCGA（AM-hCGA）法を提案する．提案手法を Oakforest-PACS システム（JCAHPC）〔4〕によって評価するとともに，軽量カーネル IHK/McKernel〔5,6〕の効果についても検討を実施した．以下，第 2 章：本研究のターゲットアプリケーション及び hCGA 法の概要・著，第 3 章：著者等による先行研究の結果の概要，第 4 章：提案手法の概要，第 5 章：IHK/McKernel の概要，第 6 章：計算結果，第 7 章：関連研究，第 8 章：まとめ・将来展望，についてそれぞれ述べる．

†1 東京大学情報基盤センター
Information Technology Center, The University of Tokyo
†2 理化学研究所計算科学研究センター
RIKEN Research Center for Computational Science (R-CCS)
†3 インテル株式会社
Intel Corporation
a) nakajima@cc.u-tokyo.ac.jp

2. 並列多重格子法の概要

2.1 pGW3D-FVM

pGW3D-FVM は有限体積法（FVM）に基づき，図 1 に示すような不均質多孔質媒体中の三次元地下水流れを解くプログラムである．問題は，(1) に示すようなポアソン方程式を所定の境界条件下で解くことに帰着される：

$$\nabla\cdot\left(\lambda(x,y,z)\nabla\phi\right)=q,\ \phi=0\ at\ z=z_{\max} \tag{1}$$

ここで ϕ は水頭ポテンシャル（potential of the water head），$\lambda(x,y,z)$ は透水係数（water conductivity），q は各メッシュにおける体積フラックスの値であり全メッシュで=1.00 に設定されている．

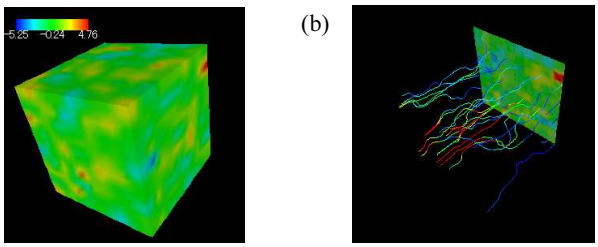

図 1 不均質多孔質媒体中の三次元地下水流れ (a)透水係数分布，(b)流線分布

図 1（a）に示す不均質な透水係数分布は地質統計学的手法〔7〕によって計算され，最小値＝10^{-5}，最大値＝10^5，平均値＝1.00 に設定されている．各メッシュは各辺長さ 1 の単位立方体であり，差分格子と同様の規則的な形状である．本研究では，128^3 メッシュの不均質透水係数分布の繰り返しによって全体メッシュを生成している〔1〕．ポアソン方程式から導かれる対称正定で疎な係数行列を有する大規模連立一次方程式は，多重格子法（Multigrid）を前処理手法とする共役勾配法（Conjugate Gradient Method preconditioned by Multigrid，MGCG）によって解かれている．

8 つのメッシュから 1 つのメッシュを生成する Octree 法を採用し，V-Cycle に基づく幾何学的多重格子法（Geometric Multigrid）を適用している．最も細かい格子レベル（level）を 1 とし，粗くなるにつれてレベルが増加する．最も粗いレベルにおける自由度数は各 MPI プロセスにおいて 1 とな

ⓒ2019 Information Processing Society of Japan

る．ブロック Jacobi 型局所 IC（0）法を加法シュワルツ法と組み合わせた緩和演算子（smoothing operator）を各レベルにおいて適用している〔1〕．OpenMP/MPI ハイブリッド型の並列プログラミングモデルが適用されている．

2.2 CGA と hCGA

著者等による初期の手法〔8〕は，図 2 に示すように，最も粗い格子レベルで各 MPI プロセスにおける自由度数が 1 になると，全プロセスの情報を 1 コアに集約して更に多重格子法により求解を継続していた．1 コアに集約後の修正方程式のソルバを Coarse Grid Solver と呼ぶ〔8〕．プロセス数が増加すると Coarse Grid Solver の負荷が高くなり，また粗いレベルでは通信のオーバーヘッドが顕著となる．

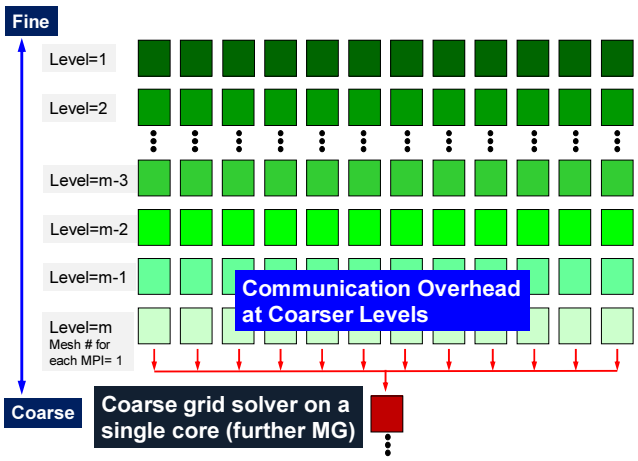

図 2　並列多重格子法と Coarse Grid Solver〔8〕

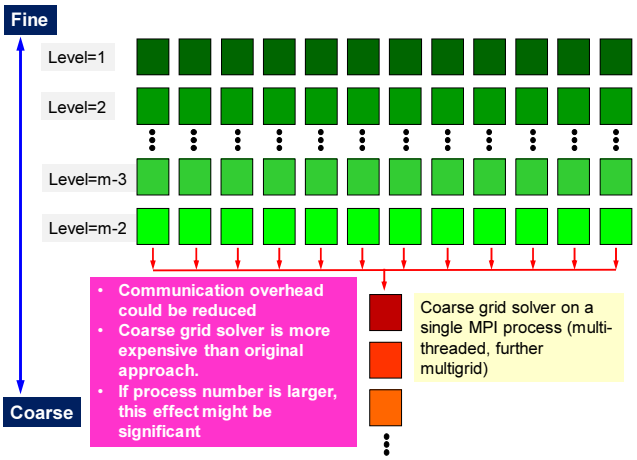

図 3　CGA（Coarse Grid Aggregation）法の概要〔9〕

〔9〕では Coarse Grid Aggregation（CGA）法として，V-Cycle のレベルの小さい（従って細かい）段階で Coarse Grid Solver に移行する手法を提案した．CGA 法によって収束の安定化，通信オーバーヘッドの削減が期待できるものの Coarse Grid Solver の扱う問題サイズは大きくなることから，Coarse Grid Solver について各 MPI プロセス上の全コアを使用し，OpenMP によってマルチスレッド化をすることによって，より高速に Coarse Grid Solver の計算を実施可能とした．

計算ノード数の増加により MPI プロセス数が増加すると，Coarse Grid Solver の問題サイズが増加し，MPI プロセス毎の計算では，計算時間がかかったり，1 ノードのメモリに入りきらない可能性あるため，対策として，ノード数を段階的に減少させる Hierarchical CGA（hCGA）法を提案した（図 4）〔1〕．

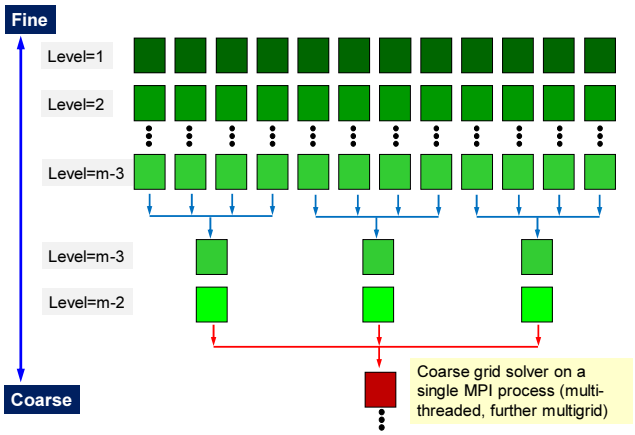

図 4 Hierarchical CGA（hCGA）法の概要〔1〕

2.3 疎行列格納手法

疎行列計算は間接参照を含むため memory-bound なプロセスである．従って疎行列演算において，演算性能と比較してメモリ転送性能の低い昨今の計算機の性能を引き出すことは困難である．係数行列の格納形式が性能に影響することは広く知られており，様々な手法が提案されている．

Compressed Row Storage（CRS）形式は，図 5（a）に示すように疎行列の非零成分のみを記憶する方法である．Ellpack-Itpack（ELL）形式は各行における非零非対角成分数を最大非零非対角成分数に固定する方法であり（図 5（b）），実際に非零非対角成分が存在しない部分は係数=0 として計算する．CRS と比較して高いメモリアクセス効率が得られることが知られているが，計算量，必要記憶容量ともに増加する．

これまで，行列格納形式に関する研究は行列ベクトル積に関するものが主であったが，著者等は IC 法，ILU 法（Incomplete LU Factorization，非対称行列向けの前処理手法）等の前処理のようなデータ依存性を有するプロセスについて検討を実施している〔1〕．差分法に見られるような規則正しいメッシュでは，各行における非零非対角成分数がほぼ固定されているため，その性質を適用することが可能である．本研究で対象としている図 1 に示すような形状では，辞書的な初期番号付けにおいては，上三角成分（自分より番号の大きい隣接要素），下三角成分（自分より番号の小さい隣接要素）の最大数は各要素において最大 3 であり，容易に ELL 形式を適用できる．スレッド並列化のためのリオーダリングに RCM 法を適用した場合もこの関係は変わらない〔1〕．

ただし ELL は，やや非効率的であり，特に不規則形状向けに，ELL 形式を拡張し，複数の配列を使用して，より効

ⓒ2019 Information Processing Society of Japan

率的な計算を実施する手法として，Sliced-ELL 形式〔2,10〕が提案されている（図 6（c））．pGW3D-FVM においては，Sliced-ELL の適用により，ELL と比較して，Oakleaf-FX 上で L2 キャッシュミス率が約 28%減少，計算性能が約 22%向上することが報告されている〔1〕．

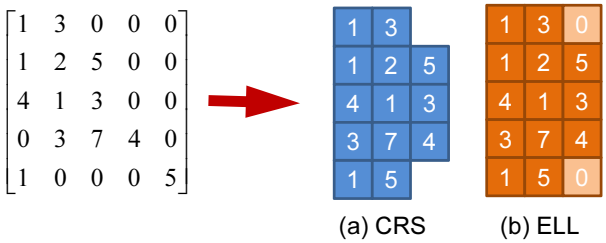

図 5　疎行列の格納形式（a）CRS，（b）ELL

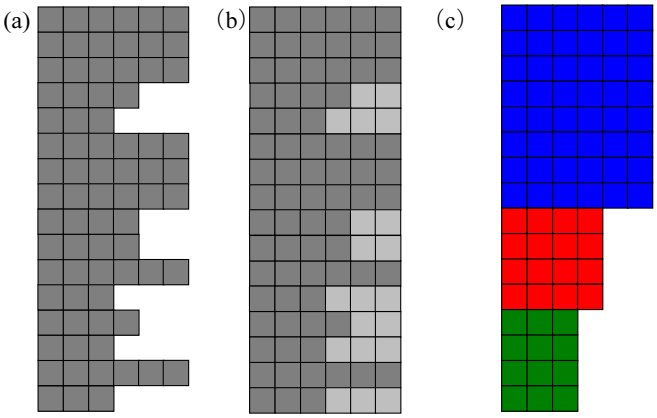

図 6　ELL 形式の不規則行列への適用例 (a) CRS，(b) ELL，(c) Sliced-ELL

3. 先行研究結果の概要

著者等による先行研究の結果〔1,8,9〕の概要を示す．Oakleaf-FX システム〔3〕の最大 4,096 ノード（65,536 コア）までを使用して実施した．Oakleaf-FX 各ノードは 16 コアから構成されており，16 コアを独立とした Flat MPI，各ノードで 4 スレッド×4 プロセス（HB 4×4），8 スレッド×2 プロセス（HB 8×2），16 スレッド×1 プロセス（HB 16×1）の各場合の計算を実施した．全体的には HB 8×2 の場合が最速である．

図 7 は HB 8×2 において，1 コア当たり 262,144 自由度 (=64^3)とした場合の，合計 4,096 ノード（65,536 コア）までの MGCG 法ソルバの計算時間である．最大問題サイズは 17,179,869,184 である．オリジナルの CRS の場合と比較すると，Sliced ELL+CGA は 4,096 ノードで約 90%の速度向上が得られており，スケーラビリティも向上している．ELL+CGA⇒Sliced ELL+CGA では 4,096 ノードで約 28%の速度向上が得られている．

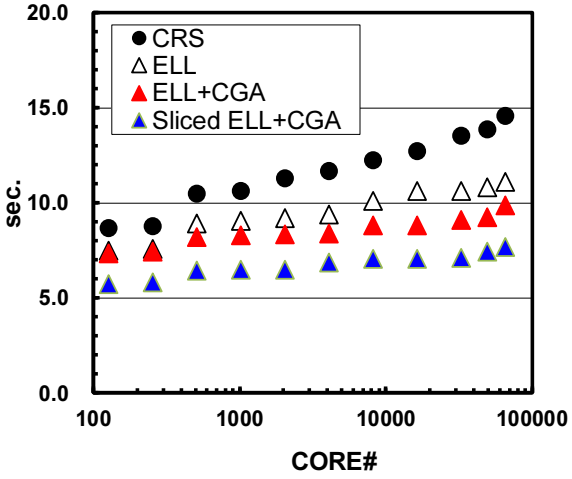

図 7 MGCG 法ソルバの Oakleaf-FX 上での計算時間，最大 4,096 ノード（65,536 コア），弱スケーリング，262,144 自由度（=64^3）/コア，最大 17,179,869,184 自由度，HB 8×2 ●△：オリジナル手法（図 2），▲▲：CGA（図 3）〔1〕

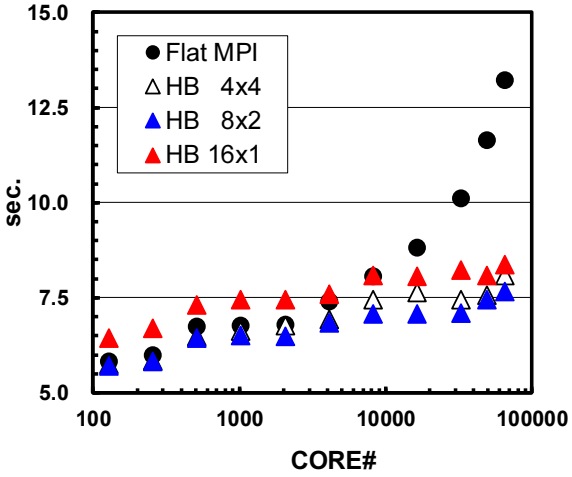

図 8 MGCG 法ソルバの Oakleaf-FX 上での計算時間（Sliced ELL+CGA），最大 4,096 ノード（65,536 コア），弱スケーリング，262,144 自由度（=64^3）/コア，最大 17,179,869,184 自由度〔1〕

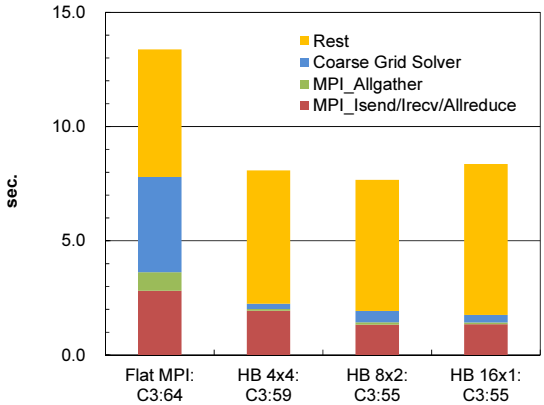

図 9 MGCG 法ソルバの Oakleaf-FX 上での計算時間の内訳（Sliced ELL+CGA），4,096 ノード（65,536 コア），17,179,869,184 自由度（図 8 の結果と同じ）〔1〕

図 8 は図 7 における最適ケース（Sliced ELL+CGA）について，各並列プログラミングモデルを比較したものである．コア数が 10^4 を超えるあたりから Flat MPI（●）の性能が急激に低下している．図 9 はそのうち 4,096 ノードにおけ

ⓒ2019 Information Processing Society of Japan

る計算時間の内訳であり，Flat MPI では Coarse Grid Solver（■）の割合が非常に大きくなっていることがわかる．これは，Flat MPI では Coarse Grid Solver の問題規模が HB 16×1 の 16 倍となるのに，利用できる計算資源が 1 コアすなわち，HB 16×1 の 16 分の 1 となり，計算効率としては理論上 256 分の 1 となるためである．

図 4 で提案した hCGA 法を適用すると，図 10 に示すように Flat MPI の性能は 4,096 ノード（65,536 コア）において約 61%改善され，他の手法と変わらなくなる〔1〕．また，強スケーリングの場合には 4,096 ノードで約 6.27 倍の速度向上が得られている〔1〕．

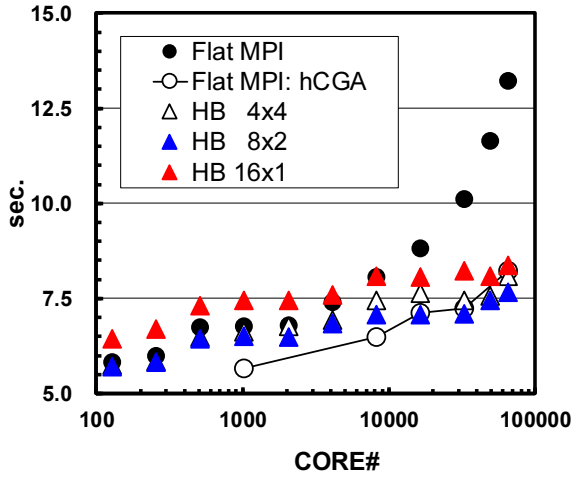

図 10 MGCG 法ソルバの Oakleaf-FX 上での計算時間（Sliced ELL+CGA/hCGA），最大 4,096 ノード（65,536 コア），弱スケーリング，262,144 自由度(=64^3)/コア，最大 17,179,869,184 自由度〔1〕

4. 提案手法の概要

4.1 AM-hCGA 法

hCGA 法によって，MPI プロセス数が増加した場合にも Coarse Grid Solver によるオーバーヘッドを回避することが可能となることが，著者等の先行研究〔1〕によって示された．MPI プロセス数が 10^4 を超えると hCGA 法が有効であると考えられるが，プロセス数が O(10^5)-O(10^6)のオーダーとなると，図 4 に示すような 2 階層＋Coarse Grid Solver では足りず，並列階層性のレベル数を 3 以上に増やす必要がある．そこで，本研究では図 11 に示すような階層性のレベル数を増やした Adaptive Multilevel hCGA（AM-hCGA）法を提案する．

4.2 Chronopoulos/Gear による共役勾配法

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 12 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：

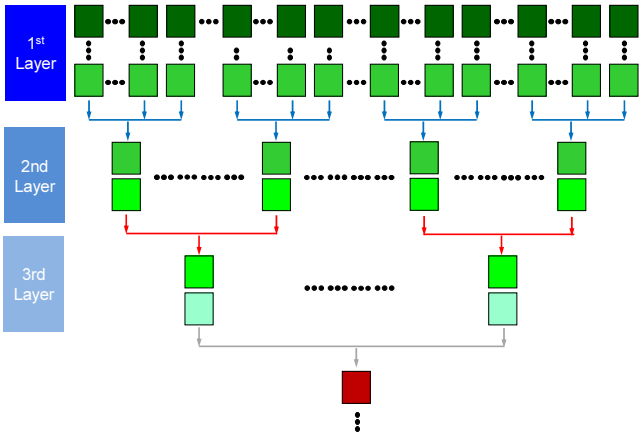

図 11 Adaptive Multilevel hCGA 法（AM-hCGA）の概要

```
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 12 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，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．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，MPI_Irecv，後者は MPI_Allreduce 等が使用される．

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．Matrix Powers Kernel〔11〕に基づく *s*-step 法〔12〕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：

① 内積実施回数削減，集団通信オーバーヘッド削減
② 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

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．

パイプライン型共役勾配法（Pipelined CG Method）〔13〕は，*s*-step 法の元になったアルゴリズムに基づき，漸化式

ⓒ2019 Information Processing Society of Japan

の適用によって，図 12 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．

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 12 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．

著者等による先行研究〔14〕では，〔13〕に示されたパイプライン型共役勾配法，及び関連手法を，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）〔3〕で計算を実施している．

本研究では，図 12 に示したオリジナルの前処理付き共役勾配法のアルゴリズム（Alg.1）の他，Chronopoulos/Gear アルゴリズム（Alg.2）〔12,13〕についての検討も実施する．

Chronopoulos/Gear アルゴリズム（Alg.2）は *s*-step 法〔12〕において *s*=1 としたもので，下記のような漸化式（2）：

$$
\begin{aligned}
&s_i = Au_i + \beta_i s_{i-1},\ p_i = u_i + \beta_i p_i \Leftrightarrow s_i = Ap_i \\
&x_{i+1} = x_i + \alpha_i p_i \\
&r_{i+1} = b - Ax_{i+1} = b - Ax_i - \alpha_i Ap_i \\
&\quad = r_i - \alpha_i Ap_i = r_i - \alpha_i s_i
\end{aligned}
\qquad (2)
$$

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 13 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 13 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 13　Chronopoulos/Gear アルゴリズム（Alg.2）〔12,13〕

図 14 は Reedbush-U 上による強スケーリングの結果で，Chronopoulos/Gear アルゴリズム（C/G アルゴリズム）（Alg.2）はオリジナルの共役勾配法と比較して高い性能が得られていることがわかる〔14〕．

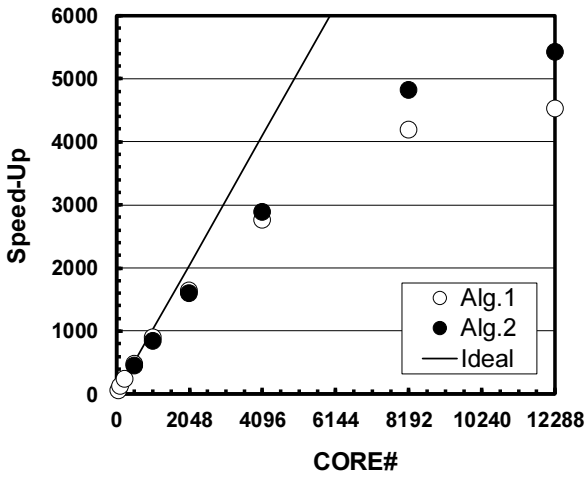

図 14　Reedbush-U による並列 CG 法の計算性能（Hybrid），Flat MPI　2 ノード（4 ソケット，64 コア）（Alg.1）の性能を 64.0 としたときの速度向上率（最大 384 ノード，12,288 コア，各ノード当たり 36 コアのうち 32 コアを使用）〔14〕

5. IHK/McKernel

IHK/McKernel〔5,6〕は理化学研究所で開発されている HPC システム向けの軽量カーネル（Light-Weight Kernel，LWK）である．IHK（Interface for Heterogeneous Kernels）は Linux ドライバとして実装されているハードウェア資源管理及び軽量カーネル管理モジュールの名称，McKernel は LWK そのものの名称である．京コンピュータ，Intel Xeon Phi（Oakforest-PACS 等）上で稼働しており，スーパーコンピュータ「富岳（ポスト京）」向けの軽量カーネルとしても研究開発が進められている．IHK/McKernel（以下「McKernel」）の開発動機は以下の通りである〔5〕：

① 大規模計算のためのスケーラブルな実行環境の提供
② アプリケーションに対して Linux とのフルコンパチビリティを提供する，すなわち Linux 上で動作しているバイナリがそのまま動作する
③ 新しいメモリ技術や電力管理機能などの新しいハードウェア機能に対して，システムソフトウェアが即座に対応できる

McKerne は Linux カーネルと共存し，McKernel が実装していない Linux システムコールは Linux 側に移譲することにより Linux フルコンパチビリティを実現している。これにより，性能センシティヴな OS 機能や新しい機能は Linux カーネルに実装するのではなく，軽量カーネル側に実装できるようになり，利用者は Linux 機能と共に軽量カーネル側で実装された機能を使用できる。IHK が有する再構成機能により，ノード単位でアプリケーションに応じた軽量カーネルを実行することが可能である．McKernel の利用により，OS ノイズが低減され，計算時間が短縮される事例に

ⓒ2019 Information Processing Society of Japan

ついては Oakforest-PACS 上でも既に報告されている〔5〕. 図 15 は IHK/McKernel の概要とシステムコール移譲機能（offloading）の概略である：

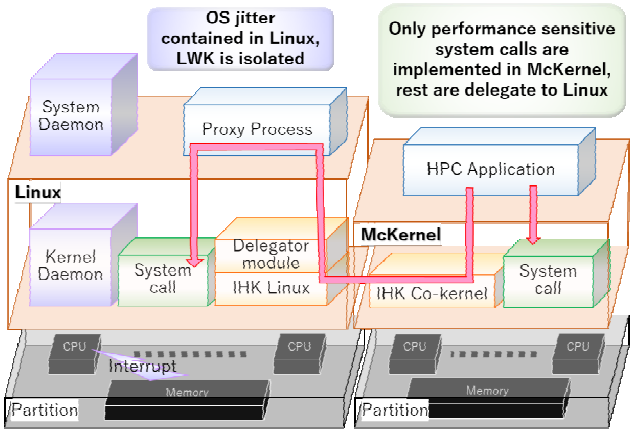

図 15 IHK/McKernel 概要・システムコール移譲機能概略〔5,6〕

6. 計算結果

6.1 計算機環境

本研究では，筑波大・東大による最先端共同 HPC 基盤施設（JCAHPC）の運用する Oakforest-PACS システム（OFP）〔4〕を使用した. OFP は Intel Xeon Phi 7250（Knights Landing, KNL）8,208 ノードを有するピーク性能 25 PFLOPS のシステムで，各ノードは Omini-Path Architecture により 100 Ggbps の Full-bisection Fat Tree ネットワークにより連結されている．各ノードは 68 コア（1.40GHz）を有し，メモリは DDR4（96GB，STREAM Triad 80.1 GB/sec）と高速な MCDRAM（16GB，同 490 GB/sec）から構成されている．各コアは最大 4 スレッドを実行可能で，512-bit のベクトルユニット 2 つを有し，AVX-512 SIMD 命令をサポートしている．様々なメモリモードと NUMA モードの組み合わせが考えられるが，本研究では，McKernel の利用も考慮して Flat/Quadrant モードを適用した．また，本研究では，68 コアのうちの 64 ノードを使用し，1 コア当たり 1 スレッド，メモリは MCDRAM のみを使用した．

表 1　Oakforest-PACS（OFP）ノード仕様の概要

Architecture	Intel Xeon Phi 7250 (Knights Landing)
Frequency (GHz)	1.40
Core #/CPU (socket) (Max Effective Thread #)	68 (272)
Peak Performance (GFLOPS)	3,046.4
Memory Size (GB)	MCDRAM: 16 DDR4: 96
Memory Bandwidth (GB/sec, STREAM Triad [8])	MCDRAM: 490 DDR4: 84.5
Compiler & MPI Library	Intel Parallel Studio 2018 XE, Intel MPI 2018

6.2 予備評価

まず，Oakforest-PACS の 8,192 ノードを使用して予備的な評価を実施した．ノード当たりの問題規模を〔1〕の場合と同じ（=4,194,304 自由度）とした．

Oakleaf-FX における最適コード〔1〕を，Oakforest-PACS（OFP）向けに更に最適化した．図 16，図 17 に示すように，非ゼロ非対角成分の計算を書き下し，再内側ループにベクトル化を施した．ベクトル化は十分ではないが，64 ノード，Flat MPI（合計 4,096 コア）の計算を実施した場合，図 17 のコードは図 16 の場合と比較して約 27%の性能向上が得られた．

```
!$omp parallel private (ip,icel,j,VAL,icol)
      do icol= 1, NHYP(lev)
!$omp do
        do ip  = 1, PEsmpTOT
        do icel= SMPindex(icol-1,ip,lev)+1,      &
                 SMPindex(icol,ip,lev)
          VAL= Bmg(icel)
          do j= 1, 3
            VAL= VAL AL(j,icel)*Xmg(IAL(j,icel))
          enddo
          Xmg(icel)= VAL*DDmg(icel)
        enddo
        enddo
      enddo
!$omp end parallel
```

図 16　Oakleaf-FX における最適コード，Sliced-ELL，IC (0) における前進代入処理部〔1〕

```
!$omp parallel private (ip,icel,j,VAL,icol)
      do icol= 1, NHYP(lev)
!$omp do
        do ip  = 1, PEsmpTOT
!$omp simd
        do icel= SMPindex(icol-1,ip,lev)+1,      &
                 SMPindex(icol,ip,lev)
          VAL= Bmg(icel)
          VAL= VAL - AL(icel,1)*Xmg(IAL(icel,1))
          VAL= VAL - AL(icel,2)*Xmg(IAL(icel,2))
          VAL= VAL - AL(icel,3)*Xmg(IAL(icel,3))
          Xmg(icel)= DDmg(icel) * VAL
        enddo
        enddo
      enddo
!$omp end parallel
```

図 17　OFP における現状の最適コード，Sliced-ELL，IC (0) における前進代入処理部，図 16 と同じ部分，最内側ループをベクトル化

続いて，HB 4×16（各ノード当たり 4 スレッド×16 プロセス）並列プログラミングモデルを適用して，弱スケーリングの計算を実施した．図 18 は最大 8,192 ノード，344,738,234,368 自由度の例である．Sliced-ELL/CGA と Sliced-ELL/hCGA を比較している．OFP においては，図 17 に示すような実装を適用している．ノード数が増加した場合の CGA の性能低下は OFP においてはより顕著であり，8,192 ノードではメモリに Coarse Grid Solver の問題が収まりきらず計算が正常に終了しなかった．ノード数が少ない

ⓒ2019 Information Processing Society of Japan

場合の OFP と Oakleaf-FX の性能比は 3-4 倍程度であるが，ノード数が多くなるにつれて，OFP の有意性は低下している．

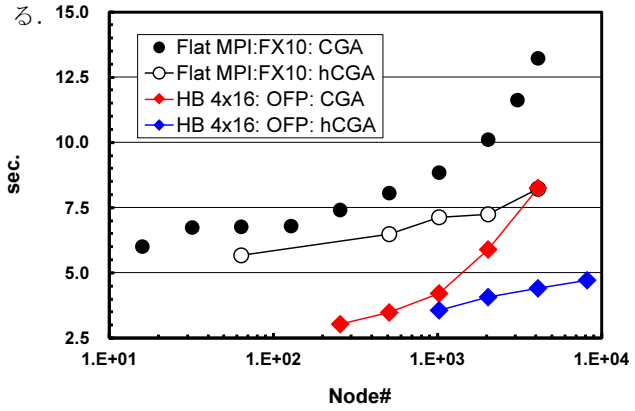

図 18 MGCG 法ソルバの計算時間（弱スケーリング）（Sliced-ELL CGA/hCGA），FX10：Oakleaf-FX 最大 4,096 ノード，65,536 コア，最大自由度数：17,179,869,184，OFP：Oakforest-PACS 最大 8,192 ノード，524,288 コア，最大自由度数：344,738,234,368

6.3 計算結果

続いて，OFP の 2,048 ノード（131,072 コア）までを使用して，弱スケーリングの計算を実施した．計算の概要は下記の通りである：

① 図 12 に示すオリジナルの共役勾配法のアルゴリズム（Alg.1）に対して，CGA（図 2），hCGA（図 3），AM-hCGA（図 11）の評価を実施した．128・256・512・1,024・2,048 ノードの各ノード数について実施した．
② 問題サイズとしては下記の二種類を実施した：
- ✓ Medium：コア当たり 64×32×32＝65,536，ノード当たり 4,194,304，最大 8,589,934,592
- ✓ Small：コア当たり 32×16×16＝8,192，ノード当たり 524,288，最大 1,072,741,824

③ 各ノード数において，①の最速なケースについて Alg.2（図 13）を適用した計算を実施した．
④ 各ノード数において，①の最速なケースについて McKernel を適用した計算を実施した．
⑤ ③の各ケースについて McKernel を適用した計算を実施した．

図 19 は Medium，Small の各問題設定について，各ノード数（コア数）における CGA 法，hCGA 法，AM-hCGA 法の各ベストケースの MGCG 法ソルバの計算時間である．それぞれ CGA 法 128 ノードの場合の計算時間で無次元化してある．hCGA 法，AM-hCGA 法の場合は，それぞれほぼスケーリングしているが，CGA 法の場合はノード数が 1,024 以上に増加すると，Coarse Grid Solver の負荷が大きくなるため，2,048 ノードでは計算時間が 10 倍以上増加している．

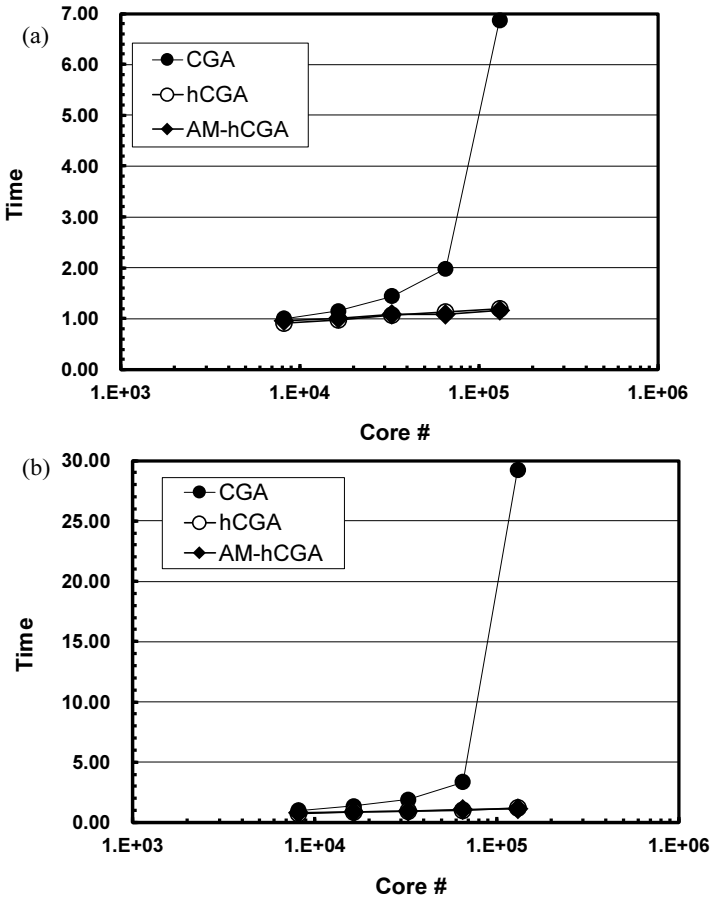

図 19　MGCG 法ソルバ（Alg.1）の計算時間（弱スケーリング）（Sliced-ELL，OFP 向け最適化済み，CGA 法 128 ノードの計算時間（MGCG 法ソルバ）で無次元化），最小 128 ノード（8,192 コア）・最大 2,048 ノード（131,072 コア）(a) Medium（最大 8,589,934,592 自由度），（b）Small（最大 1,072,741,824 自由度）

図 20 は図 19 の Time=1.00 付近を拡大し，①のベストケースに Alg.2 を適用した場合（③），McKernel を適用した場合（④），Alg.2 と McKernel の両方を適用した場合（⑤）を比較したものである．

Medium，Small の場合のいずれも 2,048 ノードでは AM-hCGA 法が hCGA 法おをわずかではあるが（Medium：2.18%，Small：6.01%）上回っている．特に Small の場合は AM-hCGA 法の効果が明瞭に出ているのは，問題サイズが小さく，Coarse Grid Solver の負荷の全体に占める割合がより大きいためと考えられる．

McKernel により計算時間は減少しており，2,048 ノードでは hCGA 法のベストケースと比較して，5.82%（Medium），19.2%（Small）と，特に Small の場合には 20%近い性能改善が達成されている．

McKernel の OS ノイズ低減効果は，コア当たり問題規模が小さく，ノード数が多い場合に顕著である．Alg.2 の影響についても，MGCG 法ソルバでは計算時間の大部分が多重格子法の緩和演算子（smoother）の計算に占められるため，余り大きくないと考えられるが，Small の場合には Alg.1 と Alg.2 の差が Medium と比較して明瞭であり，特にノー

ⓒ2019 Information Processing Society of Japan

ド数が多い場合のその差は大きくなる．

表 2 は hCGA 法・Alg.1 の場合のベストケースにおいて Small 問題を 1,024 ノード（65,536 コア）を使用して解いた場合の MGCG 法ソルバの計算時間（sec）である．5 回計算を実行しているが，McKernel を適用すると全体的に計算時間が 20%-40%増加している．McKernel を適用すると，計算時間が短く，かつばらつきも少ないことがわかる．

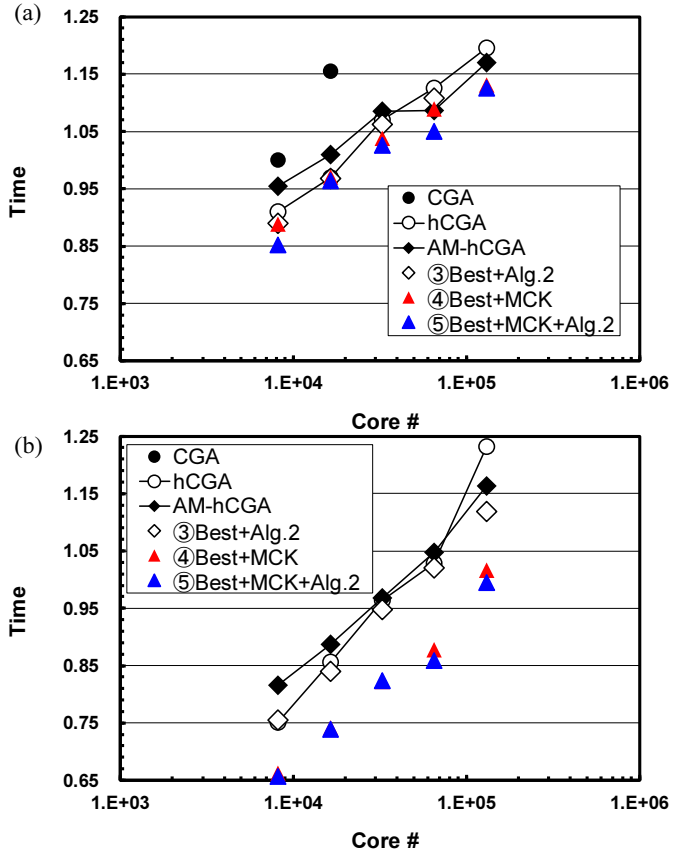

図 20 MGCG 法ソルバの計算時間（弱スケーリング）（Sliced-ELL，OFP 向け最適化済み，CGA 法 128 ノードの計算時間（MGCG 法ソルバ）で無次元化），最小 128 ノード（8,192 コア）・最大 2,048 ノード（131,072 コア）（a）Medium（最大 8,589,934,592 自由度），（b）Small（最大 1,072,741,824 自由度）

表 2 MGCG 法ソルバの計算時間，hCGA 法・Alg.1，1,024 ノード，Small，5 回実行時の MGCG 法ソルバの計算時間（sec.）・平均時間・標準偏差，McKernel の有無による比較

	w/o McKernel	with McKernel
1st Attempt	8.66E-01	6.96E-01
2nd Attempt	9.79E-01	6.94E-01
3rd Attempt	8.20E-01	6.84E-01
4th Attempt	9.28E-01	6.97E-01
5th Attempt	8.30E-01	6.91E-01
平均値	8.85E-01	6.92E-01
標準偏差	6.75E-02	5.32E-03

7. 関連研究

著者等による先行研究で評価した hCGA 法に関しては，アイディアそのものは新規なものではなく，既にいくつかの実例がある〔15,16,17,18〕が，本研究で提案している AM-hCGA 法のようにマルチレベルに適用する手法については，実用例は少ない．関連研究の多くは代数的マルチグリッド（Algebraic Multigrid）に関するものであり，多重格子法のレベル数は比較的少なく，1 プロセス当たりの問題サイズが $O(10^3)$となったときに通信オーバーヘッド削減のために，repartitioning（領域再分割）を行う例が多い．本研究，及び先行研究では，領域再分割は 1 プロセス当たり $O(10^1)$-$O(10^2)$の場合に実施しており，Coarse Grid Solver の負荷軽減が目的となっており，やや役割が異なっていると考えられる．

8. まとめ

並列多重格子法は，エクサスケールシステムにおける大規模問題向け数値解法として注目されている．本研究ではノード数が増加した場合に Coarse Grid Solver の計算負荷を軽減し，スケーラビリティを保つために，Adaptive Multilevel hCGA（AM-hCGA）を提案し，三次元不均質多孔質媒体中の地下水流れを有限体積法によって解くプログラム pGW3D-FVM のポアソン方程式求解部の多重格子法前処理付き共役勾配法ソルバ（MGCG）に実装し，最大 Oakforest-PACS 2,048 ノード（131,072 コア）を使用して良好なスケーラビリティを得た．AM-hCGA 法は特にノード数が大きい時に効果を発揮することが示された．また，内積計算を効率的に行う用に計算順序を変更した Chronopoulos/Gear アルゴリズムに基づく共役勾配法を実装し，ノード数が多い場合に明瞭な効果を発揮することが示された．軽量カーネル IHK/McKernel を適用し，特にノード当たり問題規模が小さく，ノード数が多い場合に，OS ノイズ削減効果を発揮し，Linux 使用時と比較して，最大 20%程度の計算時間改善を達成し，また，計算時間のばらつきも少ないことが確認された．

現状では OFP の 8,208 ノードのうち 2,048 ノードを使用しているのみであるが，AM-hCGA 法，Chronopoulos/Gear アルゴリズム，McKernel のいずれもノード数が多い場合に効果を発揮するため，より大規模なノード数を使用したケースを実施する予定である．

hCGA 法，AM-hCGA 法は階層レベル数が増すほど，パラメータが増加する．パラメータによって，計算・通信のバランスが変化する他，収束までの反復回数が変化するために，事前に最適パラメータを予測することは困難であるが，更に多数の場合について計算を実施することによって，自動的な最適パラメータ決定に関する研究開発も継続して実施する予定である．

ⓒ2019 Information Processing Society of Japan

本文中でも述べたように，ノード内の最適化（ベクトル化）は十分ではなく，SELL-C-σ等のベクトル計算向けの手法〔19〕を適用することにより，更なる高速化を図る必要がある．

謝辞

本研究の一部は、学際大規模情報基盤共同利用・共同研究拠点、および、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（JHPCN-HPCI)）の支援によるものです（課題番号: jh180041-NAHI（題名：Innovative Multigrid Methods，課題代表者：中島研吾（東京大学）及び jh180042-NAH（題名：高性能・変動精度・高信頼性数値解析手法とその応用，課題代表者：中島研吾（東京大学））。また Oakforest-PACS の計算資源は最先端共同 HPC 基盤施設（JCAHPC）「大規模 HPC チャレンジ」の支援を受けています。本研究の実施にあたってご協力いただいた坂口吉生氏（富士通株式会社）に感謝の意を表します。

参考文献

[1] Nakajima, K., Optimization of Serial and Parallel Communications for Parallel Geometric Multigrid Method, Proceedings of the 20th IEEE International Conference for Parallel and Distributed Systems (ICPADS 2014) 25-32, Hsin-Chu, Taiwan, 2014
[2] Monakov, A., Lokhmotov, A., Avetisyan, A., Automatically tuning sparse matrix-vector multiplication for GPU architectures, Lecture Notes in Computer Science 5952, 112-125, 201
[3] 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　https://www.cc.u-tokyo.ac.jp/
[4] Oakforest-PACS, Joint Center for Advanced High Performance Computing (JCAHPC):
[5] Balazs Gerofi，高木将通，石川裕，中島研吾，塙敏博，朴泰祐，Oakforest-PACS 上での IHK/McKernel の評価，スーパーコンピューティングニュース Vol.20-4，21-30，201
[6] McKernel: https://www.sys.r-ccs.riken.jp/ResearchTopics/os/mckernel
[7] Deutsch, C.V., Journel, A.G., GSLIB Geostatistical Software Library and User's Guide, Second Edition. Oxford University Press, 1998
[8] Nakajima, K., New strategy for coarse grid solvers in parallel multigrid methods using OpenMP/MPI hybrid programming models, ACM Proceedings of the 2012 International Workshop on Programming Models & Applications for Multi/Manycores (PMAM) in conjunction with PPoPP 2012, 2012
[9] Nakajima, K., OpenMP/MPI Hybrid Parallel Multigrid Method on Fujitsu FX10 Supercomputer System, IEEE Proceedings of 2012 International Conference on Cluster Computing Workshops, IEEE Digital Library: 10.1109/ClusterW.2012.35, 199-206, 201
[10] 中島研吾, 大島聡史, 塙敏博, 星野哲也, 伊田明弘, ICCG 法ソルバーの Intel Xeon Phi 向け最適化，情報処理学会研究報告（2016-HPC-157-16）2016
[11] Demmel, J., Hoemmen, M., Marghoob, M., Yelick, K., Avoiding Communication in Sparse Matrix Computations, IEEE Proceedings of 22nd IEEE International Parallel & Distributed Processing Symposium (IPDPS 2008), 200
[12] Chronopoulos, A.T., Gear, C.W., *s*-step iterative methods for symmetric linear systems, Journal of Computational and Applied Mathematics 25-2, 153-168, 1989
[13] Ghysels, P., Vanroose, W., Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm, Parallel Computing 40, 224-238, 2014
[14] 塙敏博，中島研吾，大島聡史，星野哲也，伊田明弘，パイプライン型共役勾配法の性能評価，情報処理学会研究報告（2016-HPC-157-6），2016
[15] Baker, A., Gamblin, T., Schultz, M., Yang, U., Challenge of Scaling Algebraic Multigrid across Modern Multicore Architectures, Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium (IPDPS'11) 275-286, 2011
[16] Lin, P.T., Improving multigrid performance for unstructured mesh drift-diffusion simulations on 147,000 cores, International Journal for Numerical Methods in Engineering 91, 971-989, 2012
[17] Sundar, H., Biros, G., Burstedde, C., Rudi, J., Ghattas, O., Stadler, G., Parallel Geometric-Algebraic Multigrid on Unstructured Forests of Octrees, ACM/IEEE Proceedings of the 2012 International Conference for High Performance Computing, Networking, Storage and Analysis (SC12) , 2012
[18] Nomura, N., Fujii, A., Tanaka, T., Marques, O.A., Nakajima, K., Algebraic Multigrid Solver Using Coarse Grid Aggregation with Independent Aggregation, Proceedings of the 13th International Workshop on Automatic Performance Tuning (iWAPT2018) conjunction with IEEE International Parallel and Distributed Processing Symposium (IPDPS), Lecture Notes in Computer Science (LNCS) 10680, 1104-1112, 2018
[19] Kreutzer, M., Hager, G., Wellein, G., Fehske, H. Bishop, A.R.: A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide SIMD units. SIAM Journal on Scientific Computing 36-5, C401‑C423, 2014

ⓒ2019 Information Processing Society of Japan