Vol.2013-HPC-139 No.6
2013/5/29

前処理付きマルチスレッド並列疎行列ソルバー

中島研吾[†1 †2]

本研究では ILU，IC 等の前処理付きマルチスレッド並列疎行列ソルバーについて行列格納形式，リオーダリングによる色数の影響，収束特性も含めた性能評価を有限体積法によるポアソン方程式ソルバーを対象として実施する．Fujistu FX10，Cray XE6 による結果を紹介する．

Multi-threaded parallel spare linear solvers with preconditioners

KENGO NAKAJIMA[†1 †2]

In this work, performance evaluations of multi-threaded parallel sparse linear solvers using ILU/IC-type preconditioners are provided. Effects on performance and convergence by reordering methods and colors, as well as effects on performance by matrix storage format are considered. Target application is a Poisson equation solver based on finite-volume method (FVM). Results on Fujitsu FX10 and Cray XE6 are described.

1. はじめに

有限要素法，差分法等の科学技術アプリケーションは最終的には大規模な疎行列を係数とする連立一次方程式を解くことに帰着される．科学技術計算において最も計算時間を要するプロセスであるため，安定的で効率的な手法の研究開発は重要な技術的課題であり，特に昨今では大規模並列計算をターゲットとした前処理付き反復法に関する研究が盛んである〔1,2〕．大規模なマルチコアクラスタ，メニィクラスタではノード間のメッセージパッシング（例：MPI），ノード内のスレッド並列（例：OpenMP）に基づくハイブリッド並列プログラミングモデルが広く使用されている〔1,2〕．

疎行列計算は間接参照が多用されているため，memory-bound なプロセスであり，最先端のハードウェアの性能を引き出すために様々なチューニングが必要である．

本研究では，有限体積法によるポアソン方程式ソルバー〔3〕から導かれる対称正定な疎行列を係数とする連立一次方程式を不完全コレスキー分解前処理付き共役勾配法（Preconditioned Conjugate Gradient Method by Incomplete Cholesky Factorization，ICCG 法）によってマルチコアクラスタで効率よく解くためのリオーダリング手法，行列格納手法を検討する．ハイブリッド並列プログラミングモデルを使用することを想定し，計算ノード上において OpenMP を使用してスレッド並列化したプログラムを対象とする．本研究で検討した手法を Fujitsu PRIMEHPC FX10（Fujitsu FX10）〔4〕，Cray XE6〔5〕にて評価する．

本稿では以下，対象とするアプリケーション，使用した計算機の概要，最適化手法，計算結果について紹介する．

†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

2. 対象とするアプリケーション

本稿で対象とするアプリケーションは図 1 に示す差分格子によってメッシュ分割された三次元領域において，以下のポアソン方程式を解くものである〔3〕：

$$\Delta\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = f \qquad (1)$$

$$\phi = 0 @ z = z_{\max} \quad (2) \qquad (2)$$

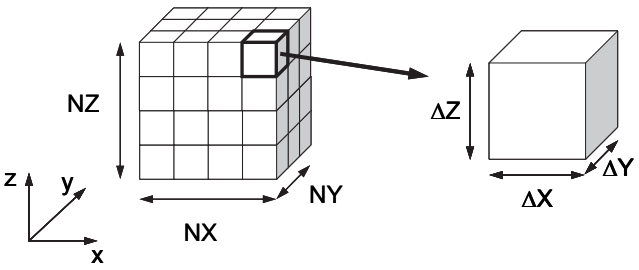

図 1　三次元ポアソン方程式ソルバーの解析対象
差分格子の各メッシュは直方体（辺長さはΔX，ΔY，ΔZ），X，Y，Z 各方向のメッシュ数は NX，NY，NZ

形状は規則正しい差分格子であるが，プログラムの中では，一般性を持たせるために，有限体積法に基づき，非構造格子型のデータとして考慮する．図 3 における任意のメッシュ i の各面を通過するフラックスについて，式（1）により以下に示す式（3）のような釣り合い式が成立する：

$$\sum_{k=1}^{6}\left[\frac{S_{ik}}{d_{ik}}\left(\phi_k - \phi_i\right)\right] + V_i f_i = 0 \qquad (3)$$

ここで，S_{ik}：メッシュ i と隣接メッシュ k 間の表面積，d_{ik}：メッシュ i-k 重心間の距離，V_i：メッシュ i の体積，f_i：メッシュ i の体積あたりフラックスである．三次元問題の場

合，各直方体メッシュは 6 個の面を持っているため，隣接メッシュ数は（最大で）6 であり，式（3）左辺第一項は k=1~6 の和となっている．式（3）を図 1 の三次元領域に適用し，整理するとメッシュ i について式（4）のようになる：

$$\left[\sum_{k=1}^{6}\frac{S_{ik}}{d_{ik}}\right]\phi_i-\left[\sum_{k=1}^{6}\frac{S_{ik}}{d_{ik}}\phi_k\right]=+V_i f_i \qquad (4)$$

これは各メッシュ i について成立する式であるので，全メッシュ数を N とすると，N 個の方程式を連立させて，境界条件を適用し，連立一次方程式 $[A]\{\phi\}=\{b\}$ を解くことに帰着される．式（4）の左辺第一項は$[A]$の対角項，第二項は非対角項，右辺は$\{b\}$に対応している．各メッシュ i に対応する非対角成分の数は最大 6 個であるので，係数行列$[A]$は疎（sparse）な行列となる．

係数行列$[A]$は対称かつ正定（Symmetric Positive Definite, SPD）であるため，前処理付き共役勾配法（Preconditioned Conjugate Gradient Method）を適用する．前処理手法としては，対称行列向けに広く使用されている不完全コレスキー分解（Incomplete Cholesky Factorization, IC）を使用する〔6〕．本研究では，係数行列は対称であるが，プログラム内では上下三角成分を別々に記憶している〔3〕．コレスキー分解（Cholesky Factorization）は対称行列を$[A]=[L][L]^T$ または $[A]=[L][D][L]^T$ のように係数行列を上下三角行列の積に分解し，前進後退代入によって連立一次方程式の解を求める直接法（Direct Method）の一種である．非対称行列向けには$[A]=[L][U]$とする LU 分解（LU Factorization）が使用される．本研究の場合は$[A]$は疎な行列であるが，$[L]$は必ずしもそうではなく，もともと 0 であったところに非ゼロ成分（fill-in）が生じる場合もある．不完全コレスキー分解ではこの fill-in のレベルや数を制御して，前処理行列 $[M]=[\tilde{L}][\tilde{L}]^T(\approx[A])$を生成する．実用的には，IC(0)（カッコ内は fill-in のレベル），すなわち fill-in を全く考慮しない場合でも，広範囲の問題に対応できる．本研究でも fill-in を考慮しない IC(0)を使用する．IC(0)では，元の行列$[A]$と前処理行列$[M]$の非ゼロ成分の位置が同じとなる．

不完全コレスキー分解を前処理手法とする共役勾配法を ICCG 法と呼ぶ．ICCG 法では，不完全コレスキー分解生成時，前進代入，後退代入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ため，リオーダリングが必要である〔1,2,6〕．

3. ハードウェアの概要

本研究で使用した Fujitsu PRIMEHPC FX10（Fujitsu FX10）は東京大学情報基盤センターの Oakleaf-FX システム〔4〕である．Fujitsu FX10 は 16 コアから構成される SPARC64™ IXfx（1.848 GHz）を使用している．全システムは 4,800 ノード，76,800 コア，メモリ容量は 154 TB であり，ピーク性能 1.13PFLOPS である．各コアは 64 KB の L1 命令／データキャッシュを有しており，12MB の L2 キャッシュは 16 コアで共有されている．各ノードは六次元メッシュトーラス構造による Tofu ネットワークによって結合されている〔4〕．SPARC64™ IXfx では 16 コアは均一にメモリに接続している（Uniform Memory Access（UMA））（図 2（a））．

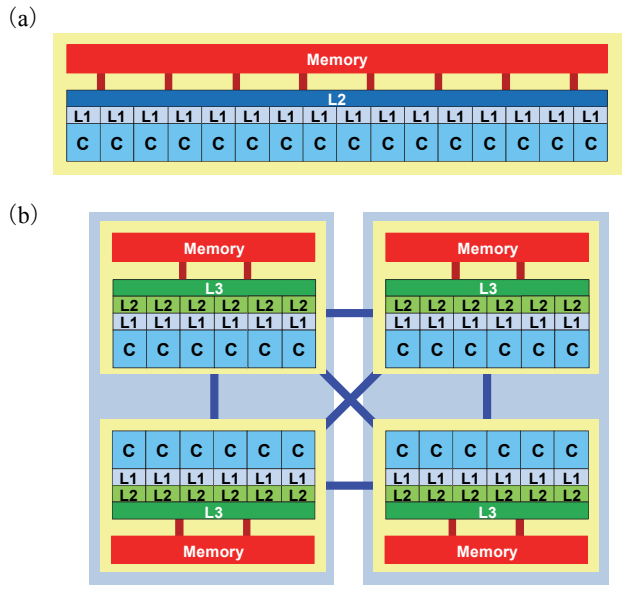

図 2 計算ノードの概要（a）Fujitsu FX10，（b）Cray XE6
（C: core, L1/L2/L3: L1/L2/L3 キャッシュ, Memory:メモリ）

表 1 各計算ノードの概要〔4,5〕

	Fujitsu FX10	Cray XE6
Core #/Node	24	16
Memory/node (GB)	32	32
Peak Performance/node (GFLOPS)	201.6	236.5
Peak Memory Bandwidth/node (GB/sec)	85.3 8×DDR3 1333MHz	
STREAM/Triad Performance/node (GB/sec) [8]	52.3	64.7
B/F Rate	0.260	0.274

本研究で使用した Cray XE6 は National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory（NERSC/LBNL）の Hopper システムである〔5〕．各計算ノードは 12 コアから構成される AMD Opteron（Magny-Cours）プロセッサ（2.1GHz）を 2 ソケット含んでおり，各ソケットは 6 コアから構成される 2 つのダイ（die）に分かれている．4 つのダイは HyperTransport によって接続されている（図 2（b））．全システムは 6,384 ノード，153,216 コア，212 TB のメモリ容量を有しており，ピーク性能は 1.28 PFLOPS である．各コアは 64 KB の L1 命令／データキャッシュ，512 KB の L2 キャッシュを有しており，6MB の L3 キャッシュは各ダイ上の 6 コアで共有されている．計算ノードは Cray 社の開発した三次元トーラス

構造ネットワーク，Gemini スイッチによって接続されている．Cray XE6 は Fujitsu FX10 とは異なり，cc-NUMA アーキテクチャ（Cache Coherent - Non-Uniform Memory Access）に基づいているため，プログラミングに当たっては注意が必要である〔1,7〕．

本研究では Fujitsu FX10，Cray XE6 の各 1 ノードを使用した．表 1 に計算ノードの概要を示す．

4. スレッド並列化，最適化に関連する項目

4.1 色づけによるリオーダリング

ハイブリッド並列プログラミングモデルでは，各ノード（ソケット）に対応した局所データを OpenMP などのマルチスレッド的な手法によって並列化に処理する．ICCG 法では不完全コレスキー分解，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〔1,2,3,9〕．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．

マルチカラー法（Multicoloring，MC）は高い並列性能とスレッド間の負荷分散を容易に達成可能であるが，悪条件問題で収束が悪化する．また，色数を増やすことによって収束を改善できるが，OpenMP のオーバーヘッドにより性能が低下する場合がある．高い並列化効率を得るためには，できるだけ各色内の要素数が多い方が都合が良い．

レベルセットによる並べ替え法である Reverse Cuthill-McKee （RCM）法は，悪条件問題に対する収束性は良いが，各レベルセットに含まれる要素数は不均一であり，並列性能は MC 法と比べると低い．

この問題を解決する手法として RCM 法によって並び替えを施された要素に対して，更にサイクリックに再番号付けする Cyclic マルチカラー法（Cyclic Multicoloring，CM）を適用する手法（CM-RCM）が考案されている〔1,2,3,9〕．図 3 は CM-RCM 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 法の第 1 色に分類されている．各色には 16 の要素が含まれている．CM-RCM 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充分大きい必要がある．

本研究ではこの CM-RCM 法を使用する．

4.2 NUMA アーキテクチャ向け最適化

Cray XE6 のような NUMA（Non-Uniform Memory Access）アーキテクチャに基づくハードウェアでは，各コアができるだけローカルなメモリ上にあるデータをアクセスできるように，データ配置等に配慮することが望ましい．

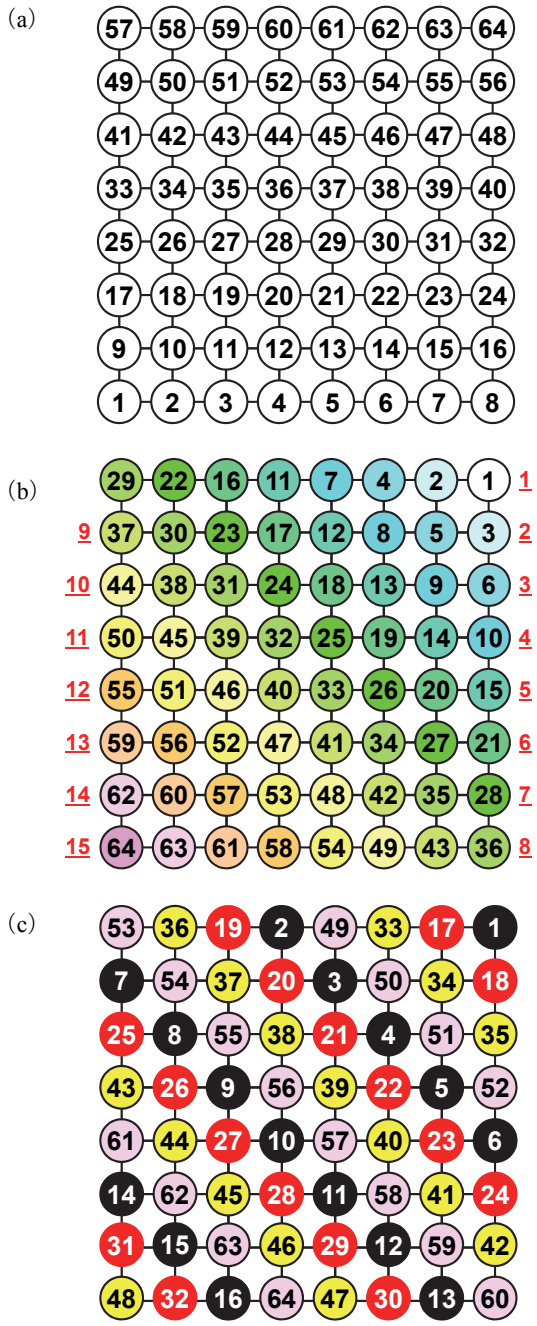

図 3 CM-RCM 法による色づけとリオーダリング，(a) 元のグラフ，(b) RCM 法によるリオーダリング（赤字はレベルセット番号），(c) CM-RCM 法による再リオーダリング（4 色：CM-RCM(4)），各色内の要素数は 16 でバランス

後述するように，本研究ではノード内の全てのコア数に等しい数のスレッドを使用して並列化を実施するため，特に注意が必要である．

よく知られた方法として以下の 2 種類がある〔1,3,7〕：

- First Touch Data Placement の適用
- Sequential Reordering によるデータ再配置

(1) **First Touch Data Placement**

NUMA アーキテクチュアでは，プログラムにおいて変数や配列を宣言した時点では，物理的メモリ上に記憶領域は確保されず，ある変数を最初にアクセスしたコア（の属するソケット）のローカルメモリ上に，その変数の記憶領域が確保される．これを First Touch Data Placement〔7〕と呼び，配列の初期化手順により大幅な性能の向上が達成できる場合もある．具体的には，配列を使って実際の計算の手順にしたがって配列を初期化することによって実現できる．

(2) **Sequential Reordering**

4.1 で示した CM-RCM 法による並べ替えでは，図 4（a）に示すように：

- 同一の色に属する要素は独立であり，並列に計算可能
- 「色」の順番に各要素を番号付けする
- 色内の要素を各スレッドに振り分ける

という方式を採用しているが，同じスレッド（すなわち同じコア）に属する要素は連続の番号では無いため，効率が低下している可能性がある．このような番号付けを Coalesced Numbering と呼ぶ．

Sequential Reordering は CM-RCM による Coalesced Numbering に対して再番号付けを適用し，同じスレッドで処理するデータを連続に配置するように更に並び替えるものである（図 4（b））．Sequential Reordering は NUMA アーキテクチャだけでなく Fujitsu FX10 のような UMA アーキテクチャにも有効である．

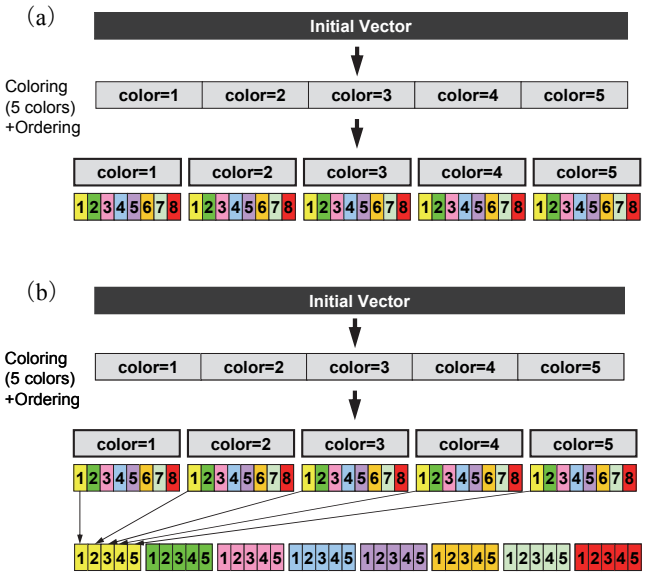

図 4 要素の番号付け（a）CM-RCM 法による番号付け（Coalesced Numbering），（b）Sequential Reordering による再番号付け（各スレッド上の要素は連続な番号付け）

(3) **疎行列格納形式**

既に述べたように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〔6〕．

最も広く使用されているのは Compressed Row Storage（CRS）形式である．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り，著者による先行研究でも使用されている〔1,2,3〕．Ellpack-Itpack（ELL）形式は各行における非零非対角成分数は最大非零非対角成分数に固定する方法で（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〔10〕，本研究では 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する．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，ELL を適用することは比較的容易と考えられる．しかし本研究のように前処理付き反復法，特に IC 法，ILU 法などの前処理を含む場合には，再番号付けによって上下三角成分の数が変わる可能性があるため注意が必要である．

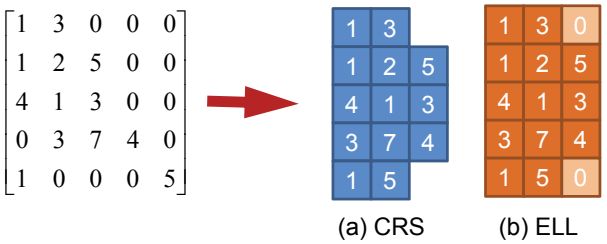

図 5 疎行列の格納形式（a）CRS，（b）ELL

5. 計算例

5.1 プログラムの概要

表 2 3 種類のプログラムの概要

	並列化向け色付け手法	番号付け	First Touch Data Placement	係数行列格納形式
Case-1	CM-RCM	Coalesced（図 4 (a)）	無し	CRS
Case-2		Sequential（図 4 (b)）	有り	
Case-3				ELL

本研究では，図 1 において NX=NY=NZ=128（2,097,152 要素）として計算を実施した．利用した計算機は Fujitsu FX10, Cray XE6 それぞれ 1 ノードである．並列化は OpenMP

によるスレッド並列化を適用し，スレッド数＝コア数とした．したがって，Fujitsu FX10 では 16 スレッド，Cray XE6 では 24 スレッドである．

表 2 に示す 3 種類のプログラムを実行した．Case-2 が NUMA アーキテクチャ向けの最適ケースとして著者による先行研究で適用されたものである〔2,3〕．

ポアソン方程式から得られる対称正定な行列を係数行列とする連立一次方程式を，ICCG 法によって解く計算時間を評価した．係数行列は対称であるが，2．で述べたように上下三角成分は別々に記憶している．

図 1 に示すような三次元有限体積法によるポアソン方程式ソルバーから得られる係数行列は，表面メッシュを除けば非零非対角成分は各行あたり 6 であり，初期番号付け（図 3（a）参照）に従えば，上下三角成分は 3 つずつである．ELL 形式を適用する場合，最大非零非対角成分数は上下共に 3 とすれば良い．

RCM 法では図 1 に示すような形状の場合，各要素番号の大小関係は変化しないため，上下三角成分の数も変化しない（図 3（b）参照）．しかしながら，MC 法では特に色数が少ない場合に，6 つの隣接要素番号が全て自分の番号より大きく（あるいは小さく）なるような場合がある〔2,3〕．このような場合には，最大非零非対角成分数が 6 となり，図 5（b）において係数=0 となる部分の割合が増加するため，ELL 形式のメリットが得られない可能性がある．

しかしながら，本研究で使用している CM-RCM 法の場合は，図 3（c）からもわかるように，CM-RCM の色数を m 色とすれば，1 色目・m 色目を除くと RCM 法と同様に各要素番号の大小関係は変化しない．本研究では，1 色目・m 色目に含まれる要素については，最大非零非対角成分数=6 の他の要素とは異なる配列を係数行列格納のために準備し，不完全コレスキー分解，前進・後退代入，行列ベクトル積については 1 色目，m 色目は他の色とは別に計算を実施することとした．ただし m=RCM のレベルセット数の場合は，CM-RCM(m)=RCM となるので，1 色目，m 色目は他と区別していない．

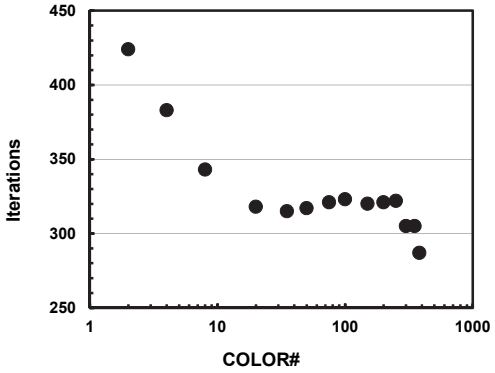

図 6　ICCG 法ソルバーの計算性能：色数と反復回数の関係，要素数：128^3（=2,097,152）

5.2 計算結果

色数を 2 から 382（=RCM におけるレベルセット数）まで変化させた場合の反復回数（$\varepsilon=10^{-8}$）を図 6 に示す．本ケースの場合，CM-RCM(382)=RCM である．色数の増加とともに反復回数は減少する〔9〕．

図 7 は Fujitsu FX10，Cray XE6 による ICCG 法ソルバーの計算時間と色数の関係である．色数が増加すると各色内の要素数が減る．各色内で並列化を実施するため，色数が増加すると同期のオーバーヘッドの影響のため，一反復あたりの計算時間は増加する．図 7 に示すように本ケースの場合は，色数の増加とともに計算時間が減少する傾向が見られる．

Case-1 と Case-2 の違いは NUMA アーキテクチャ向けの最適化の有無である．NUMA アーキテクチャに基づく Cray XE6 ではこの効果は大きく，Case-1⇒Case-2 で計算時間は半分になっている．これに対して，UMA アーキテクチャである Fujitsu FX10 ではこの効果はあまりなく，色数が 200 以上の場合に Case-2 が 10%程度優位となっている．

Case-2 と Case-3 の違いは CRS⇒ELL である．図 7（a）からもわかるように Fujitsu FX10 においてはこの効果は顕著であり計算時間によると 50%～60%の改善が見られる．Cray XE6 における効果はあまり大きくなく 5%～10%程度の改善である．

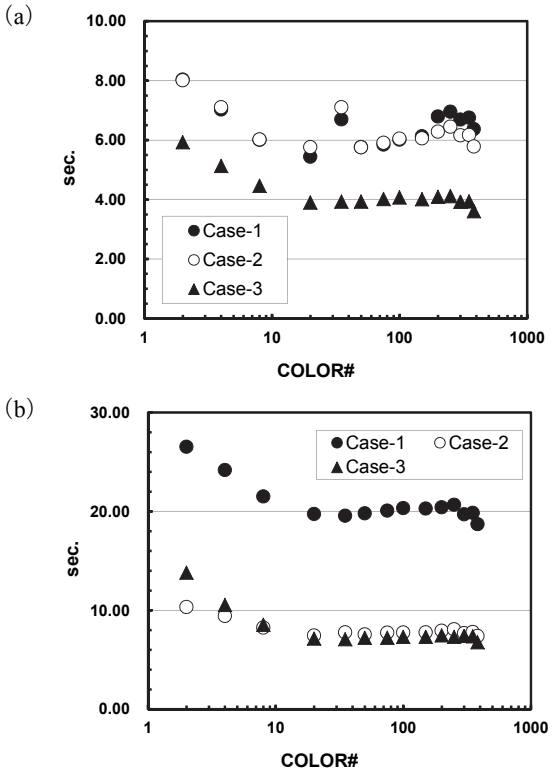

図 7　ICCG 法ソルバーの計算性能：色数と計算時間の関係，（a）Fujitsu FX10，（b）Cray XE6

表 3 は色数=20，382（=RCM）の場合の ICCG 法ソルバーの全計算時間，一反復あたりの計算時間である．色数増加とともに Fujitsu FX10，Cray XE6 ともに OpenMP の同期オーバーヘッドの影響により，反復あたりの計算時間は増加している．

Fujitsu FX10 では色数が多い場合には NUMA アーキテクチャ向け最適化による性能の改善が見られる．First Touch Data Placement の有無による性能の変化は認められないが，Sequential Reordering により，RCM では 10%程度の性能の改善が見られる．色数が多くなると各色内での要素数が少なくなる．また，Case-1 では次の色に移る際にアクセスするデータのアドレス変動が大きくなるため，キャッシュミスが生じやすくなっている可能性がある．Case-2 ではスレッド（=コア）上での番号が連続となっているため，次の色へ移る際のアドレス変動が少ない．

表 4 はプロファイラによって求めた Fujitsu FX10 における Case-2, Case-3 の性能諸元である．CRS⇒ELL によって，Instruction 数が半減し，SIMD 化率，Memory Access Throughput 共に大幅に改善していることがわかる．

表 3　ICCG 法ソルバーの計算性能：CM-RCM(20)・CM-RCM(382)（=RCM）における計算時間

		CM-RCM(20)		CM-RCM(382) = RCM	
		計算時間（秒）	一反復当たり計算時間（秒）	計算時間（秒）	一反復当たり計算時間（秒）
Fujitsu FX10	Case-1	5.44	1.71×10^{-2}	6.37	2.22×10^{-2}
	Case-2	5.76	1.81×10^{-2}	5.78	2.02×10^{-2}
	Case-3	3.90	1.23×10^{-2}	3.61	1.26×10^{-2}
Cray XE6	Case-1	19.7	6.26×10^{-2}	18.7	6.52×10^{-2}
	Case-2	7.45	2.34×10^{-2}	7.40	2.58×10^{-2}
	Case-3	7.14	2.25×10^{-2}	6.77	2.36×10^{-2}

表 4　ICCG 法ソルバーの計算性能（Fujitsu FX10）：プロファイラによる性能諸元，上段 : CM-RCM(20)，下段 : RCM

	Instructions	SIMD（%）	Memory Access Throughput（%）
Case-2	1.83×10^{11}	7.17	50.2
CRS	1.83×10^{11}	6.90	44.5
Case-3	6.71×10^{10}	16.3	69.8
ELL	5.96×10^{10}	16.2	67.0

6. まとめ

三次元有限体積法によるポアソン方程式ソルバーから得られる対称正定な行列を係数行列とする連立一次方程式を，OpenMP によって並列化された ICCG 法によって解く場合に，行列格納形式，リオーダリング手法に着目した性能評価を実施した．Fujitsu FX10，Cray XE6 のそれぞれ 1 ノードを使用した．データ依存性を回避する並列化抽出のための色付け手法としては CM-RCM 法を使用した．

Sequential Reordering によってスレッド上のデータの連続性が高まるため，特に色数が多くなると Fujitsu FX10 のような UMA アーキテクチャでもキャッシュミス削減に効果があるものと考えられる．

係数行列格納形式を CRS から ELL に変えることの効果は Fujitsu FX10 において特に顕著で, 50%～60%程度の性能改善が見られた．一方 Cray XE6 では NUMA アーキテクチャ向け最適化と比較して効果は少なく, 5%～10%程度の改善にとどまった．

今後は Xeon Phi を含む様々なアーキテクチャで特に ELL の効果を検証するとともに，有限要素法で得られるようなより複雑な構造のマトリクスへの適用について検討していきたい．

謝辞　本研究実施にあたって貴重な助言を頂いた富士通株式会社の皆様に，謹んで感謝の意を表する．

参考文献

1) Nakajima, K.: Flat MPI vs. Hybrid: Evaluation of Parallel Programming Models for Preconditioned Iterative Solvers on “T2K Open Supercomputer”, IEEE Proceedings of the 38th International Conference on Parallel Processing (ICPP-09), pp.73-80 (2009)
2) 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; p.199-206 (2012)
3) 中島研吾：T2K オープンスパコン（東大）チューニング連載講座（その 5），OpenMP による並列化のテクニック：Hybrid 並列化に向けて，スーパーコンピューティングニュース（東京大学情報基盤センター）11-1（2009）
4) Information Technology Center, The University of Tokyo: http://www.cc.u-tokyo.ac.jp/
5) NERSC, Lawrence Berkeley National Laboratory: http://www.nersc.gov/
6) Saad, Y.: Iterative Methods for Sparse Linear Systems Second Edition, SIAM (2003)
7) Mattson, T.G., Sanders, B.A., Massingill, B.L.: Patterns for Parallel Programming, Software Patterns Series (SPS), Addison-Wesley (2005)
8) STREAM (Sustainable Memory Bandwidth in High Performance Computers): http://www.streambench.org/
9) Washio, T., Maruyama, K., Osoda, T., Shimizu, F., 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)
10) Kotakemori, H., Hasegawa, H., Kajiyama, T., Nukada, A., Suda, R., and Nishida, A.: Performance Evaluation of Parallel Sparse Matrix-Vector Products on SGI Altix3700, Lecture Notes in Computer Science 4315, pp.153-163 (2005)