Vol.2012-HPC-133 No.29
2012/3/27

マルチ GPU を用いた AMG 法

高橋光佑† 藤井昭宏† 田中輝雄†

本研究では大規模な非構造格子の問題を高速に解ける線形解法の一つである AMG 法のマルチ GPU 上での実装について説明する．緩和法は AMG 法の性能を決める重要な要素である．CPU 上では幾つかの緩和法の実装が検討されているが，高い並列性が求められる GPU 上では十分な検討がされていない．従って，本研究ではヤコビ法，マルチカラー・ガウス・ザイデル法，弱い依存関係を排除することによって調整彩色したマルチカラー・ガウス・ザイデル法など 3 つの緩和法を用いて評価した．更に本稿では色情報に基づいたパディングや行列の並べ替えを必要とする GPU に最適な実装についても示した．

Multiple GPUs-based AMG Method

Kosuke Takahashi†, Akihiro Fujii† and Teruo Tanaka†

This study describes the implementation of the algebraic multigrid (AMG) method on multiple graphics processing units (GPUs), which is one of the fastest linear solvers for problems with a large sparse matrix. A smoother is an important component of an AMG solver that mainly determines the solver's robustness and performance. Although several types of smoothers implemented on CPUs have been studied, the best method to implement smoothers on GPUs, which require abundant fine-grain parallelism, still needs to be developed. Thus, this study evaluates the efficiency of three representative smoothers such as the Jacobi, the multicolor Gauss-Seidel method, and the modified multicolor Gauss-Seidel methods, in which unknowns are colored with a smaller number of colors by eliminating weak dependencies. In addition, this paper shows a well-suited implementation of the multicolor Gauss-Seidel method on GPU that requires padding and reordering of the matrices on the basis of color.

1. はじめに

定常物理拡散など様々な物理現象は大規模な連立一次方程式を解くことに帰着される．AMG 法[1]はこれらの連立一次方程式を高速に解く手法の一つとして知られており，構築部と解法部から成る．

GPGPU とは GPU を用いた汎目的計算技術で，近年，高速化手法として注目されている．従来，性能を引き出すために高い並列性が求められる GPGPU に合わせ，緩和法には並列性の高いヤコビ法が用いられてきた[2][3]．しかしヤコビ法はガウス・ザイデル法と比べ収束が悪く，連立一次方程式の性質によっては収束が困難な場合が存在する．これに対しマルチカラー法を用いてガウス・ザイデル法を GPU 上で並列化し，緩和法に適用することで解法の性能を向上することが出来る．

本研究ではマルチ GPU 上での AMG 法にもガウス・ザイデル法を緩和法として使用した場合の性能を評価した．

2. マルチ GPU 上での AMG 法

2.1 AMG 法と領域分割

AMG 法は大規模な非構造格子の問題を高速に解ける数値解法である．これはマルチグリッド法のように複数段階に分けて粗い行列を生成し，これら小規模な行列を用いて問題の行列を解く．AMG 法は複数の小さな行列を生成する構築部と，反復により解く解法部からなる．簡単な例として 2 段階の場合を挙げる．細かい階層とは問題の行列を指し，粗い階層とは問題の行列から生成された小規模な行列を指す．

まず構築部を完了させた後，解法部を反復することで問題の行列を解く．最初にヤコビ法（以下，JS 法）やガウス・ザイデル法（以下，GS 法）のような緩和法を細かい階層に対して複数回使用する．次に残差ベクトルを計算し，Restriction 行列を用いて行列ベクトル積を行い，粗い階層の右辺ベクトルとする．次に粗い階層を解くことにより残差を打ち消すベクトルを計算し, Prolongation 行列を用いて行列ベクトル積を行い，細かい階層の右辺ベクトルに足し込む．最後に細かい階層に対して再び緩和法を複数回使用する．以上を繰り返すことにより解を収束させる．複数の階層を行き来する様子が V 字を思わせるため，一般に解法部を V-cycle と呼ぶ．階層の数は増減が可能である．図 2-1 は 3 階層の V-cycle の例である．解法部は主に緩和法，疎行列ベクトル積，収束判定に用いる相対残差計算のための内積計算から成る．

† 工学院大学
Kogakuin University

Vol.2012-HPC-133 No.29
2012/3/27

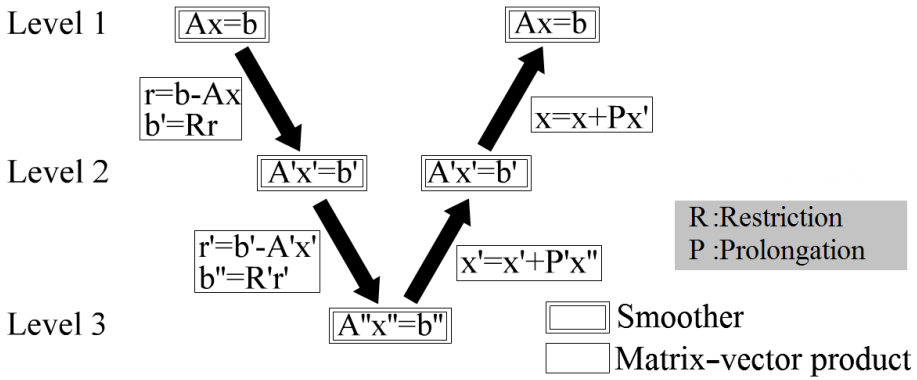

図 2-1 V-cycle

本研究では接点に基づく領域分割をしている[4]. 例えば図 2-2 はシンプルな二次元格子構造の領域分割の例である．上の領域と下の領域は異なるメモリ空間に配置される．これに対し緩和法を適用する場合，境界領域においては接点の情報を通信で相互にやり取りする必要がある．通信テーブルとしては隣接プロセスに対して，SEND テーブルと RECV テーブルをもたせている．SEND テーブルのその隣接領域の計算で参照される未知数番号の集合であり、RECV テーブルは自領域の計算で参照する隣接プロセスの未知数番号の集合である．また，Restriction や Prolongation を行う場合，緩和法の時とは異なる通信が必要となる．AMG 法では粗い行列を生成する際，アグリゲートと呼ばれる接点のグループを作り，それに基づいて行列を生成する．図 2-3 はアグリゲートの生成例である．このように AMG 法では細かいレベルが規則構造を持っていても、一段粗くなると不規則な構造をもった行列になることが分かる．また、領域分割を用いると,境界領域を跨るような形でアグリゲートが存在する可能性がある．この場合，アグリゲートはどちらかの領域に所属させることになり，所属に応じて通信を行う．図 2-4 は Restriction と Prolongation の通信例を示している．赤の領域が Restriction を行う際，アグリゲートの一部が青の領域にあることがわかる．Restriction はアグリゲートを一つの値に纏める処理なので，他の領域からアグリゲートの一部を受信する．一方で Prolongation は，一つの値を元のアグリゲートに分散させる処理なので，他の領域に値を分散させる．

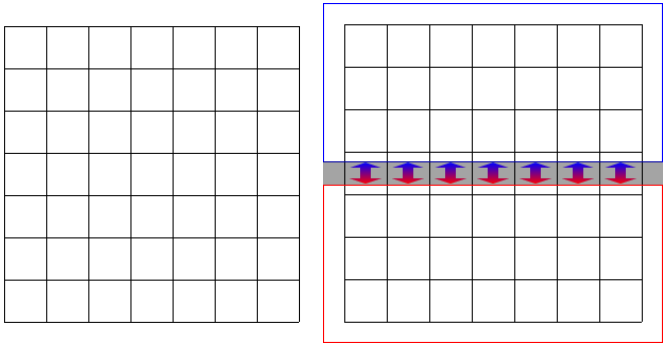

図 2-2 二次元格子構造と領域分割による通信

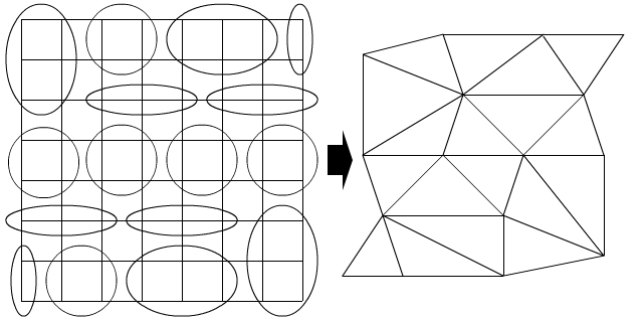

図 2-3 アグリゲートの生成

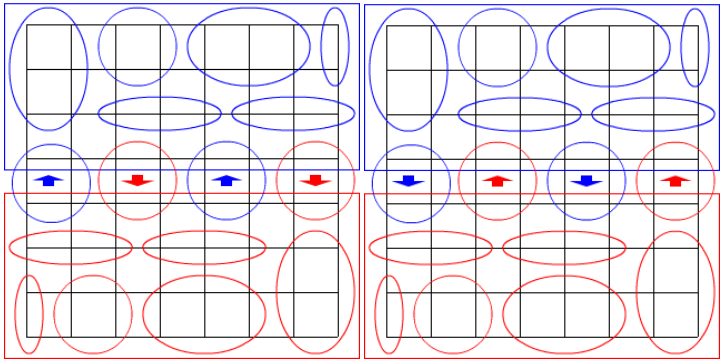

図 2-4 Restriction と Prolongation の通信

2.1 緩和法の種類

緩和法は AMG 法の解法としての性能を決める重要な構成要素である．本研究では JC 法とマルチカラーGS 法（以下，MCGS 法）と調整彩色 MCGS 法（以下，Mod-MCGS 法）を用いた．

JC 法は並列性が高いが，収束性能が低い．MCGS 法は収束性能が高いが，並列性が低い．Mod-MCGS 法は MCGS 法より並列性が高く，JC 法より収束性能が高い．しかし，同じ彩色を施した未知数を同時に更新した場合，アクセス競合が起こることで Chaotic 性（実行する度に異なる挙動を示す性質）を持つ．

GS 法では，必ずしも全ての未知数に依存関係が存在するわけではない．これは依存関係が無いようにグループ分けすることが可能であることを意味する。これをグラフ彩色問題に帰着することで依存関係のないグループを作り，並列処理を行うことをマルチカラー法（以下，MC 法）と呼ぶ[5]．本研究では細かい階層に対しては貪欲彩色法，粗い階層に対しては Welsh-Powell 法[6]を彩色方法として採用した．本研究の数値実験では細かい階層は構造格子なので，貪欲彩色法により綺麗に塗り分けが可能である．しかし，それ以外の階層では構造が複雑化し，貪欲彩色法による綺麗な塗り分けが期待できないため，少しでも色数が減る期待が持てる Welsh-Powell 法を用いた．

図 2-5 は領域分割法で 2 プロセスに分散された疎行列の非ゼロ要素分布である．AMG 法により生成された中間層の粗い行列なので，幾何的な特徴もなく複雑な分布になっているのが見て取れる．この疎行列に対して彩色を行い，色ごとに行列を並べかえたのが図 2-6 である．対角線上に四角い空間が無数に出来ているが，空間の数が色分けの数に相当する.つまり図 2-6 はマルチカラー法による並列性の確保において，十分に機能していないことを示している．これに対して調整彩色を施したのが図 2-7 である．具体的には対角成分と比較して 10%を超える非ゼロ要素のみを彩色の依存関係に考慮し，塗り分けを行っている．値の大きい非ゼロ要素のみを考慮することによって，緩和法の収束性能を維持しつつ，並列性を上げている．しかし，前述した通り Mod-MCGS 法は Chaotic 性を持つ．

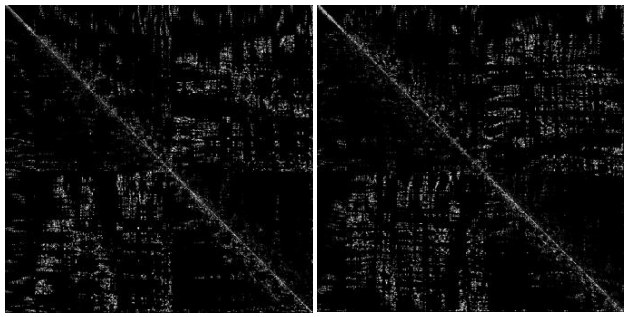

図 2-5 領域分割法で 2 プロセスに分散された疎行列の非ゼロ要素分布

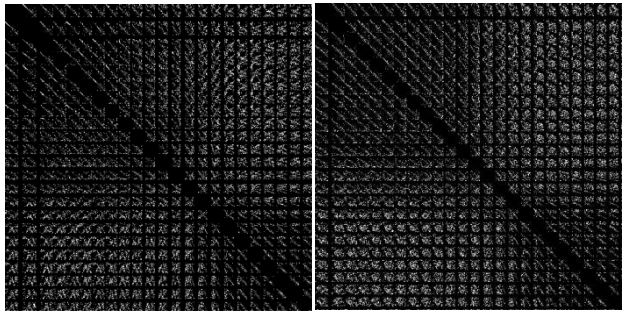

図 2-6 各領域における完全な彩色後，色ごとに並べ替えた後の様子

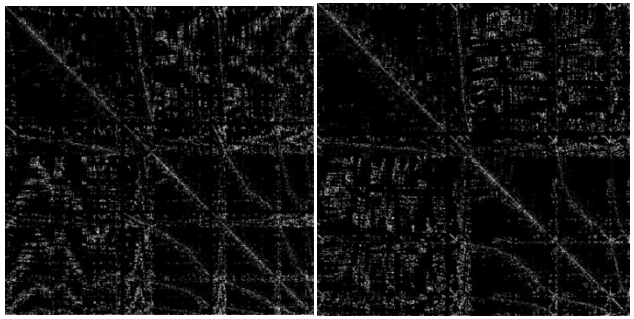

図 2-7 各領域における調整した不完全な彩色後，色ごとに並べ替えた後の様子

2.2 マルチカラー法における行列のデータ構造

本研究では多くの処理時間がメモリアクセスによって消費される．グローバルメモリアクセスのレイテンシーは GPU 上での算術演算に比べて 100 倍長くなる。疎行列の場合，ランダムアクセスになる部分とならない部分があるため，ランダムアクセスにならない部分のメモリアクセスの最適化が重要となる．

CUDA プログラミングで高い性能を得るためには，コアレッシングなグローバルメモリアクセスの技術が最も重要である．グローバルメモリアクセスはハーフワープ（16 スレッド）ごとにメモリセグメントがアラインメント（例えば１２８バイトに）されている場合，コアレスなアクセスを行う．従って，パディングと並べ替えが重要になる．

GPU 上での疎行列ベクトル積について，列優先へのフォーマットとパディングによるコアレッシングアクセスが必要になる．しかし MCGS 法では色ごとに未知数を更新し，列優先でも列方向のアクセスが不連続になるため，行の並べ替えを必要とする．

最上層以外では疎行列の非ゼロ要素の構成が複雑となり，彩色した際の色数が多くなる．これは並列性の低下を招き，計算速度が降下する．前述したように，強制的に色数を減らすことにより計算速度は確保されるが，Chaotic 性が生じてしまう．ひとつの解決法として，各階層に応じで緩和法を使い分けることにした．例えば上から二つの階層までは MCGS 法で実装し，残りの階層は JC 法で実装するといった具合だ．

彩色後，疎行列の要素を並べ替えて GPU に転送する必要がある．ここで，コアレッシングを考慮した並べ替えとパディングを行う．実行時には色ごとに並列処理が実行される．つまりコアレッシング条件を満たすためには色ごとに疎行列がまとめられている必要がある．疎行列の圧縮形式として主に CRS 形式と CCS 形式が挙げられるが，ここでは CRS 形式を基準に話を進める．

例として図 2-8 のような疎行列を考える．各スレッドには各行成分に関する計算を担当させる．同じ色に彩色された行に関しては同時に処理が行え，並列性が存在する．例の場合，黄色に彩色された行要素について最大 5 並列まで実行が可能である．図 2-9 は CRS 形式で疎行列を圧縮した例である．そのままの CRS 形式の場合，「ハーフワープ（16 スレッド）ごとにメモリセグメントがアラインメントされている」点を満たしていないので，コアレッシングは利用されない．図 2-10 はパディングを用いて仮に 4 要素にアラインメントし，スレッドがアクセスするアドレスが連続した領域になるように並べ替えた例である．

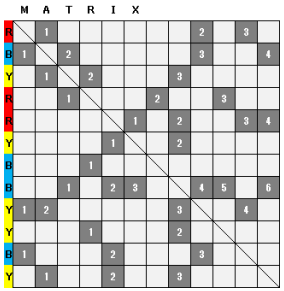

図 2-8 疎行列彩色後例

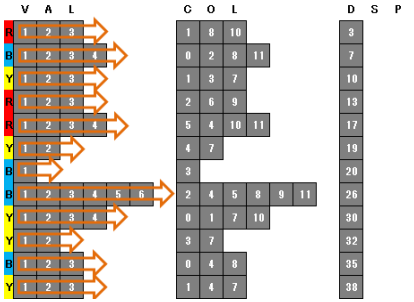

図 2-9 疎行列 CRS 形式

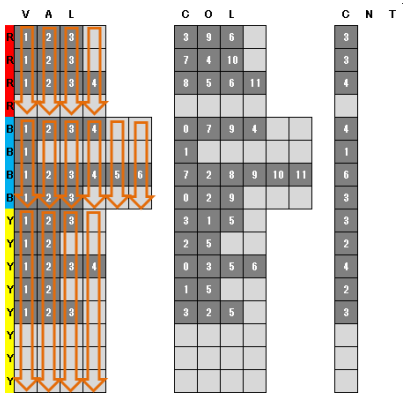

図 2-10 疎行列特製圧縮形式

本節と関連して，AMG 法の Prolongation 行列と Restriction 行列に関しても同様なデータの並べ換えと書き換えが必要になる．ここで注意しなければならないのは，連立一次方程式の疎行列 A に関しては同階層の彩色のみ考慮するのに対し，階層間に存在する疎行列の Prolongation 行列と Restriction 行列は，行と列に関して別階層の彩色を考慮してデータを整形する．図 2-11 にその様子を示す．また，領域分割の場合に彩色とは関連のない行部分が Restriction 行列に存在する．これは領域境界にあるアグリゲートに関して別領域のための計算があるため，自領域の彩色とは関係のない接点が発生するためである．本研究ではこのような無彩色領域に対し，最終色（図 2-8 を例とするならば Y 色）に所属させることで対処している．

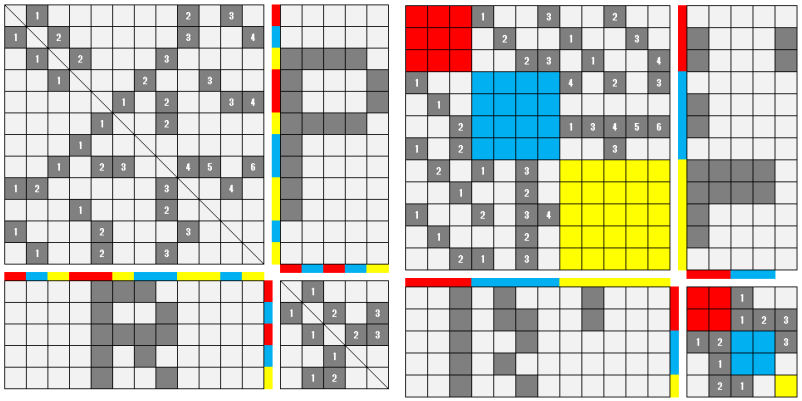

図 2-11 階層間のリオーダリング

通信テーブルに関してリオーダリングは行わない．何故ならば，本研究では MPI を用いて CPU 側で通信を行うからである．リオーダリングは主にコアレスなアクセスを実現するために行うもので，CPU 上では不要になるからである．しかし，色分けに基づいて問題行列の行が入れ替わっているため，リナンバリングの必要がある．

3. 数値実験と考察

3.1 実行環境と計算対象

本研究の実行環境を表 3-1 に示す．GPU が二枚の構造となっている．本研究では MPI を用いてマルチ GPU を利用するため，2 プロセスを立ち上げて、それぞれのプロセスが一枚の GPU を担当する形でマルチ GPU を管理している．また，メインメモリ空間に関しては NUMA コントロールを用いて分割されないように調整した．GPU の詳細を表 3-2 に示す．GPU の理論値性能は倍精度演算で 515Gflosp，メモリ帯域幅が 144GB/s である．今回の手法はメモリ帯域幅を使い切ることで計算性能が向上するので，後者のほうが重要である．また，GPU 間の通信を行うために CPU を介してのデータ転送が頻発するため．I/O シリアルインタフェースがボトルネックになる可能性がある．

表 3-1 実行環境

CPU	Intel Xeon X5570 Quad-core 2CPU 2.93GHz
Graphics	ETS2050-C3ER 2GPU
Memory	12GB (6x 2GB DDR3-1333 DIMM)
OS	CentOS 5.5 (64bit)

表 3-2 GPU の仕様

GPU Chip	NVIDIA TESLA C2050
Peak Performance	515Gflops
Number of CUDA core	448
CUDA core Clock	1.15GHz
Memory Transfer	144GB/s
Memory Interface	384bit
Video memory	3GB GDDR5
I/O Serial Interface	PCI Express 2.0 x16

問題は三次元拡散方程式で問題サイズは50^3と100^3のものを用意．いずれも貪欲彩色法で 8 色に塗り分け可能である．マルチレベルの生成には SA-AMG 法[7][8]を用いた．

3.2 比較手法

1-GPU 版と 2-GPU 版で比較を行った．スムーザー構成は互いに四種類を用意．全階層 JC 法，最下層のみ JC 法で残りを MCGS 法，最下層のみ JC 法で残りを Mod-MCGS 法，最上層のみ MCGS 法で残りを JC 法．2-GPU 版は分割領域に基づいて MPI で実装している[4].

各階層緩和法の反復回数は最下層で 30 回，残りを 2 回で行った．本実験では 1 反復あたりの通信時間の構成などを分析する．

3.3 計算結果

表 3-3 問題サイズ 50x50x50 に於ける V-cycle スピード（ms/loop）

スムーザー / GPU	JC 法	MCGS 法	Mod-MCGS 法	MCGS 法-JC 法
GPU x1	7.42	63.10	11.38	7.74
GPU x2	15.95	66.77	18.44	16.39

表 3-4 問題サイズ 100x100x100 に於ける V-cycle スピード（ms/loop）

スムーザー / GPU	JC 法	MCGS 法	Mod-MCGS 法	MCGS 法-JC 法
GPU x1	48.25	163.24	59.21	55.25
GPU x2	32.76	134.35	43.20	37.41

表 3-5 全体の通信時間の割合（%）

スムーザー / サイズ	JC 法	MCGS 法	Mod-MCGS 法	MCGS 法-JC 法
50x50x50	30.3	7.3	27.9	28.8
100x100x100	12.8	3.9	10.7	10.4

表 3-6 通信効率（MB/sec）

スムーザー / サイズ	JC 法	MCGS 法	Mod-MCGS 法	MCGS 法-JC 法
50x50x50	77.56	76.63	72.77	79.46
100x100x100	350.63	280.67	318.57	379.85

3.4 考察

V-cycle のスピードを比較すると，50^3の場合は GPUx1 構成が総じて早いが，100^3の場合は GPUx2 構成が総じて早い．また，通信の割合を見ると，50^3が 7～30%なのに対し，100^3は 4～13%に抑えられている．これは今回の計算対象が三次元問題であり，領域分割における境界が二次平面上に展開されるため，問題サイズが 8 倍になるのに対し通信量が高々4 倍（V-cycle につき 375KB から 1478KB に増加）で済む点が大きい．また，通信効率が PCI Express 2.0 x16 のピーク性能（同方向 8GB/sec）に対して低い値を示しているが，通信データの粒度が小さく，纏まりのある通信が行われないのが原因である．この点も問題サイズが大きくなることによりデータの粒が大きくなるため，通信効率の向上が見られる．

スムーザーごとの比較を見ると，JC 法が総じて計算速度の点で優位であることが分かる．しかし，100^3を例に見ると，JC 法に対しての Mod-MCGS 法，MCGS 法-JC 法の 1 反復あたりの計算時間増加は 14～32%であり，Mod-MCGS 法や MCGS 法-JC 法にすることで 14～32%の反復減少が見込める場合は解法として後者が優位であると言える．また，50^3の場合はスムーザーによる計算効率の差が GPUx1 の時に大きいのに対し，100^3の場合は GPUx2 の時のほうが大きい．これは問題サイズが大きくなればなるほど MCGS 方を用いることでの優位性が失われる可能性を示している．それでもマルチ GPU に於いて MCGS 法などを適用することでの計算時間の増加が 14～32%で済むことは，収束のしづらい問題に於いて大いに優位に働くと考える．

4. おわりに

本研究ではマルチ GPU を用いた AMG 法の実装を行った．緩和法にヤコビ法，マルチカラー・ガウス・ザイデル法，調整彩色マルチカラー・ガウス・ザイデル法を導入し，最適な実装手段を示した．MC 系 GS 法でも JC と比較して，計算速度差が大きくないように実装できた．GPU 間の通信インタフェースには MPI を用い，事前のリナンバリングによるスムーズな通信を実現した．三次元拡散方程式の問題を用いて数値実験を行い，問題サイズ50^3と100^3で比較を行ったところ，問題サイズ50^3では GPUx1 の場合が計算効率に於いて優位だったが，問題サイズ100^3になると GPUx2 の場合が優位であることを示せた．また緩和法の種類による 1 反復の計算時間の変動も 14～32%に抑え，収束しづらい問題に対しても対応できる結果を出せた．

今後は GPU 間でのダイレクト通信の実装を試みる．現在の実装は一度ホストメモリ側にデータを戻し，再び異種の GPU メモリにデータを転送する構成になる．これを GPU 同士でのダイレクト通信に切り替えることにより，理論上通信時間を半分に減らすことができる．これにより更なる高速化が見込まれ，問題サイズの小さな場合でもマルチ GPU が優位な状態にできると考えられる．

Vol.2012-HPC-133 No.29
2012/3/27

参考文献

1) F. H. Pereira, S. L. L. Verardi, and S. I. Nabeta, A fast algebraic multigrid preconditioned conjugate gradient solver, Applied Mathematics and Computation 179, pp.344-351, (2006).
2) G. Haase, M. Liebmann, C. Douglas, and G. Plank. :A Parallel Algebraic Multigrid Solver on Graphical Processing Units, HPCA-16, LNCS 5938, pp.38-47, Bangalore, India, January (2010).
3) N. Bell, S. Dalton, and L. Olson, Exposing fine-grained parallelism in algebraic multigrid methods, in NVIDIA Technical Report NVR-2011-002, June (2011).
4) 藤井昭宏，小柳義夫: 科学技術シミュレーションにて多用される代数的多重格子法の評価, シミュレーション 第 28 巻第 4 号，pp.9-14，(2009)
5) 佐藤陽平: マルチカラー・ガウス・ザイデル法を用いた非構造格子粘性流体解析コード SURF の並列化，第 8 回海上技術安全研究所研究発表会，8th，pp.319-320，June (2008).
6) D. J. A. Welsh and M. B. Powell, An upper bound for the chromatic number of a graph and its application to timetabling problems, The Computer Journal, 10, pp.85-86, (1967).
7) P. Vanek, J. Mandel, & M. Brezina, Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems, COMPUTING, 56, pp.179-196, (1996).
8) A. Fujii, A. Nishida, & Y. Oyanagi, Evaluation of parallel aggregate creation orders: smoothed aggregation algebraic multigrid method, International Federation for Information Processing, 172, pp.99-122, (2005).