Vol.2012-ARC-202 No.30
Vol.2012-HPC-137 No.30
2012/12/14

GPU のダイレクト通信を用いた AMG 法

高橋光佑[†1]　藤井昭宏[†1]　田中輝雄[†1]

概要：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 上での実装手法について考察する．緩和法は AMG 法の性能を決める重要な要素であるが，高い並列性が求められる GPU 上では十分な研究がなされていない．そこで GPU 上の AMG 法ではよく利用されるヤコビ法以外にもマルチカラー・ガウス・ザイデル法や弱い依存関係を排除することによる調整彩色したマルチカラー・ガウス・ザイデル法の GPU 上での効率的な実装手法を提案する．また，本研究では MPI を用いてマルチ GPU に実装した．通信部分に GPU 間のダイレクト通信を使用することで最適化を行う．数値実験の結果，通信部分を最大で 1.8 倍高速化した．更にマルチ GPU 環境においても強い異方性の問題についてマルチカラー・ガウス・ザイデル法を適用した AMG 法はヤコビ法を緩和法としたときより早く収束することがわかった．

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いる．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．AMG 法では従来，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を用いるか[2][3]，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，緩和法に適用してきた[4][5]．著者らも[4][5]のように GPU 上の AMG 法について研究を進め，MPI によるマルチ GPU 上への実装の拡張手法[6]も提示している．

本研究では，マルチ GPU 間の通信にダイレクト通信を用いることで最適化を行った．数値実験では 2 つのスムーザーを用いてポアソン問題に対する評価を行った．

2. GPU 上での AMG 法

ここでは GPU 上で動作する AMG 法の実装手法を説明する．

2.1 AMG 法

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．構築部は逐次的な処理が多く GPU 化をしても性能が出ないという研究があり[2]，本研究では AMG の解法部を GPU 化の対象とする．

AMG 法の解法部では最初にヤコビ法（以下，JC 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．階層の移動には Restriction 行列や Prolongation 行列を用いて行列ベクトル積を計算し，ベクトルを伸縮させることで行っている．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 1 は 3 階層の V-cycle の例である．本研究では V-cycle のすべてを GPU 上で実装した． マルチ GPU 上に実装する際は後述の領域分割法を用いるため，緩和法の適用時などに境界部分の通信が必要となる．

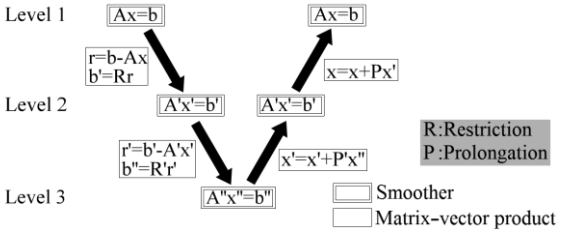

図 1　V-サイクル

Fig. 1　V-cycle.

†1　工学院大学
Kogakuin University

Vol.2012-ARC-202 No.30
Vol.2012-HPC-137 No.30
2012/12/14

2.2 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，必ずしもすべ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[7]．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[8]を彩色方法として採用した．なぜならば粗い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ことから，少しでも色数が減る期待を持つために用いる．

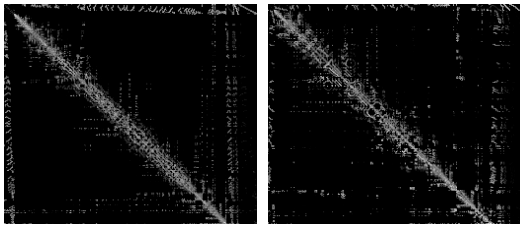

図 2　領域分割法で分散された疎行列の非ゼロ要素分布
Fig. 2　Distribution of Nonzero Elements with Domain Decomposition.

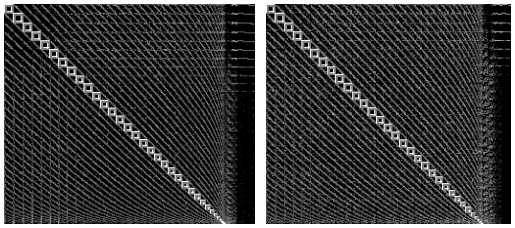

図 3　完全な彩色後，色ごとに並べ替えた後の様子
Fig. 3　Reordering with Original Coloring.

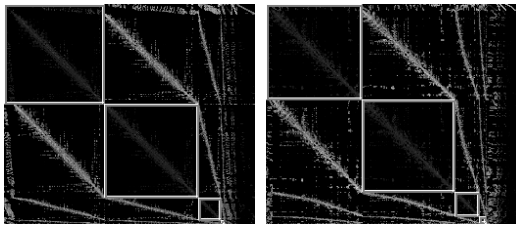

図 4　調整した不完全な彩色後，色ごとに並べ替えた様子
Fig. 4　Reordering with Modified Coloring.

図 2 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．図内の灰色や白色は非ゼロ要素を表す．白に近いほど対角成分比で値が大きい．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 3 である．対角線上の四角い強調線は，その内側に非ゼロ要素が存在しないことを示している．この空間の数は色分けの数に相当し，同一色内の行について並列性がある．つまり図 3 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 4 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．マルチカラー法を用いることはデータ構造に密接に関係する．次節ではデータ構造について説明する．

2.3 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．

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．後述する領域分割法を用いた場合も境界部分の通信対象が並べ替えられるため，通信対象を再指定する必要がある．疎行列の圧縮形式として CRS (Compressed Row Storage) 形式や CCS (Compressed Column Storage) 形式などが挙げられるが，ここでは CRS 形式を基準に話を進める．

Vol.2012-ARC-202 No.30
Vol.2012-HPC-137 No.30
2012/12/14

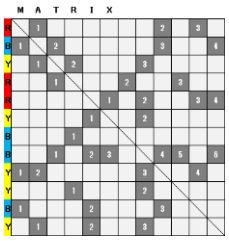

図 5　疎行列彩色例

Fig. 5　Sparse Matrix with Colors.

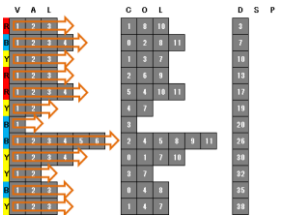

図 6　疎行列 CRS 形式

Fig. 6　Sparse Matrix with Original CRS.

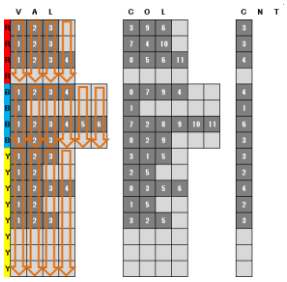

図 7　疎行列特製圧縮形式

Fig. 7　Sparse Matrix with Interleaved CRS.

例として図 5 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，Y 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 6 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 7 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．疎行列特製圧縮形式は一つの疎行列を複数の疎行列に分けた形をしている．それぞれの部分疎行列の具体的なアラインメント要素数はパディング前の行数と GPU が要求するコアレッシング条件に依存する．本研究では 256 バイトでのアライメントが要求されるため，以下の様な式で決定される．np はアラインメント要素数．n はパディング前の行数を示している．

$$np = n + \frac{256}{4} - \left(n \bmod \frac{256}{4} \right)$$

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 8 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他領域を使用した計算があるため，いわゆるのりしろ部分が存在するからである．本研究ではこのような無彩色領域に対し，最終色（図 5 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．

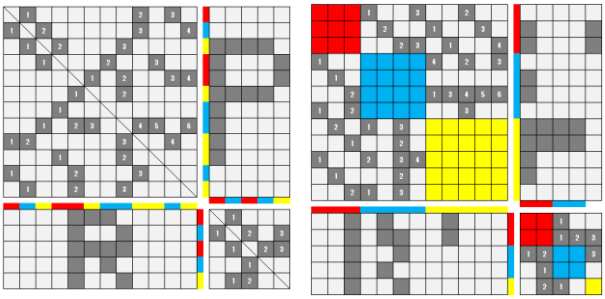

図 8　階層間のリオーダリング

Fig. 8　Prolongation and Restriction Matrix with colors.

3. マルチ GPU 化

3.1 領域分割法

本研究では接点に基づく領域分割をしている[9]．例えば図 9 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 テーブルは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．

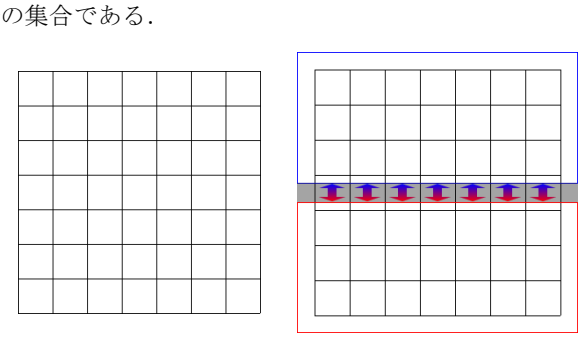

図 9　二次元構造格子と領域分割による通信

Fig. 9　Two-dimensional Grid Structure and Communication by Domain Decomposition.

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，それに基づいて粗い行列を生成する．図 10 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的な構造を持っていても，一段粗くなると不規則な構造を持った行列になることがわかる．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，境界領域では所属に応じて通信を行う．図 11 は Restriction と Prolongation の通信例を示している．下の領域が Restriction を行う際，アグリゲートの一部が上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．

3.2 マルチ GPU 化に伴う通信

本研究では MPI を用いて通信を制御している．図 12 は本研究の計算機構成の概要を示している. マルチ GPU 間の通信には大きく分けて 3 つの経路がある．CPU が使用可能なメインメモリを一度経由する経路（以下，Via host），メインメモリを経由することなく CPU のチップセットのみを経由する経路（以下，Via IOH chip），PCIe switch のみを経由する経路（以下，Via PCIe switch）だ．但し Via PCIe switch はマルチプロセスでは利用できないため，MPI を用いた本研究では使用しない．図 13 に本研究の計算機構成下での通信速度の実測値を示す．通信量は AMG 法で想定される範囲とし, 完全な双方向通信で計測している. なお，計測の開始と終了直前に大域同期処理を行い，これを通信時間とした．AMG 法が扱う範囲においても Via IOH chips は Via host の二倍以上の通信速度が出ており，通信部分の高速化が期待できる．

本研究では通信の必要な領域は GPU 上のバッファにコピーして通信を行う. コピーには通信テーブルを用いるが，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．マルチカラー法に基づく疎行列特製圧縮行列と合わせて通信テーブルのリナンバリングを行い，反復部で繰り返し使用する．GPU 間の同期は各プロセスが MPI を用いてシグナルを相互通信し行なっている．

収束判定や BiCGSTAB 法で生じる内積計算についても通信が必要となる. 本研究では各 GPU 上で総和計算を行い，スカラー値を GPU 上のメモリからメインメモリに戻し，MPI の集団通信を用いて内積計算とした. GPU 上での総和計算にはシェアードメモリを用いて高速化を図っている．

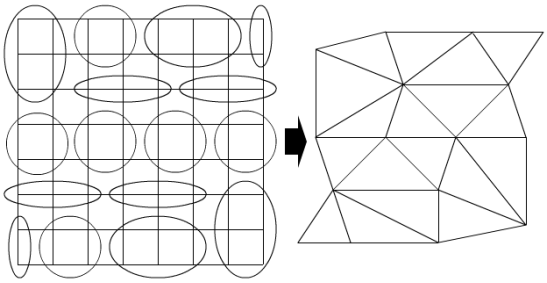

図 10　アグリゲートの生成
Fig. 10　Generation of Aggregate.

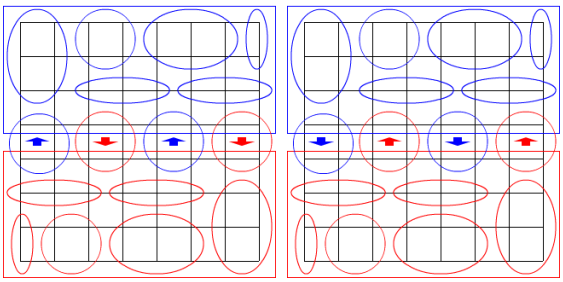

図 11　Restriction と Prolongation の通信
Fig. 11　Communication of Restriction and Prolongation.

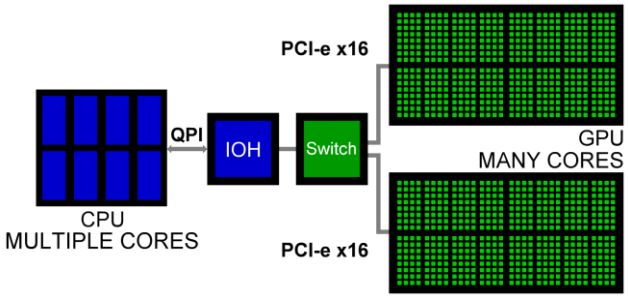

図 12　計算機構成
Fig. 12　Computer Configuration.

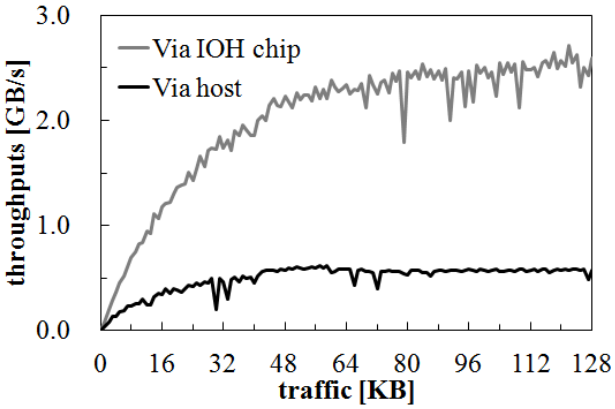

図 13　双方向通信速度
Fig. 13　P2P Throughputs for Duplex.

Vol.2012-ARC-202 No.30
Vol.2012-HPC-137 No.30
2012/12/14

4. 数値実験と考察

ここでは実行環境と計算対象を説明し，数値実験の結果と考察を示す.

表 1　実行環境

Table 1　Execution environment.

CPU	Intel Xeon E5507 Quad-core 2CPU 2.27GHz
Graphics	ETS2050-C3ER
Memory	12GB (6x 2GB DDR-3 SDARM 1333MHz)
OS	CentOS 5.8
CUDA	5.0

表 2　GPU の仕様

Table 2　GPU Spec.

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 (x8)

表 3　100^3 の異方性問題サイズ

Table 3　Test case of 100^3 problem for anisotropic.

Level	Number of rows			Number of nonzero elements		
0	500,412	+	499,588	13,242,677	+	13,220,915
1	38,279	+	38,510	3,071,769	+	3,087,250
2	2,837	+	2,232	297,068	+	229,645
3	409	+	158	36,711	+	14,952
4	81	+	0	3,273	+	0

表 4　60^3 の異方性問題サイズ

Table 4　Test case of 60^3 for anisotropic.

Level	Number of rows			Number of nonzero elements		
0	110,195	+	105,805	2,877,293	+	2,762,459
1	8,117	+	7,984	616,702	+	609,161
2	651	+	429	57,203	+	35,763
3	84	+	6	3,184	+	260

4.1 実行環境と計算対象

本研究の実行環境を表 1 に示す. GPU が二枚の構造となっている．本研究ではマルチ GPU を利用するため，MPI を用いて 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 2 に示す． GPU 間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．本環境では複数の GPU を同一チップセット上に接続する関係で，PCIe-x16 と PCIe-x8 が混在し非対称になっている．また，すべての浮動小数演算は倍精度で行なっている．

問題は三次元拡散方程式で 27 点参照の格子構造となっている．等方性については JC 法が有効と従来の研究[6]で示されているので，本数値実験では異方性のみを計測対象とした．異方性は Z 軸方向の拡散が他軸より 100 倍の拡散になっている．問題サイズは複数種用意した．マルチレベルの生成には SA-AMG 法[10][11]を用いた．表 3 と表 4 に各プロセスが持つマルチレベルの行列の一部例を示す．領域分割法を用いているため，MPI のプロセス番号ごとにマルチレベルの行列を持つ．SA-AMG 法は一定の条件を満たすまでマルチレベルを生成するため，階層数は問題サイズにより異なる．最上層はいずれも貪欲彩色法で 8 色に塗り分け可能である．また，収束条件として 2 ノルム相対残差を用い，1.0E-7 を下回れば解けたものとした．

4.2 比較手法

AMG-BiCGSTAB 法について GPUx1 版と GPUx2 版で比較を行った．スムーザー構成は互いに二種類を用意．具体的には全階層を JC 法で構成した場合と, 最上層だけ MCGS 法とし，それ以外の階層を Mod-MCGS 法で構成した場合で比較した．GPUx2 版は分割領域に基づいて MPI で実装している[9].

各階層緩和法の反復回数は最下層で 30 回, 残りを 2 回, もしくは階層数に応じて段階的に回数を減らして行い，最速なものを採用している．例えば表 3 であれば Level 昇順で 4, 3, 2, 1, 30 となる. 緩和係数は 0.1 刻みで探索した最適な値とした．調整彩色は最上層では行わず，他の階層で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た.

なお，ブロック数に対するスレッド数は 32，64，128，256，512 を試し，最も早かった値を最適値とした．ブロック数は並列可能な処理数を超えるスレッドが生成されるように設定している.

比較として CPU 版の評価も行う. すべてのスムーザーに逐次処理の対称ガウス・ザイデル法を用いている．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．並列処理として領域分割法による 8 分割が行われている.

Vol.2012-ARC-202 No.30
Vol.2012-HPC-137 No.30
2012/12/14

4.3 計算結果

GPUx1 版，GPUx2 版，CPU 版について表 5, 表 6, 表 7 に計算結果を示す．GPU 版の”Jacobi”はすべての緩和法を JC 法とした構成を示し，”MCGS”は前述の最上層だけ MCGS 法とし，それ以外の階層を Mod-MCGS 法とした構成を示している．CPU 版の”SGS”はすべての緩和法を対称ガウス・ザイデル法とした構成を示している．

4.4 考察

まず CPU 版と GPU 版を比較する．表 5, 表 6, 表 7 によるとすべての問題サイズで GPU 版が優位である．CPU 版は GPU 版と比べて領域内での並列要求度が低く，GPU 版より強力な緩和法を掛けられるのが特徴の一つだ．しかし本数値実験では GPU 版が各階層緩和法の反復回数を増やすことで全体の性能を上げられたのに対し，CPU 版では 3 回以上の緩和法反復回数の増加は計算時間の増加を招いた. 結果として収束までの反復回数においても GPU 版が優位となった．GPU 版は最大で CPU 版の約 10 倍の性能が出た．

次に緩和法の違いに触れる．従来の GPU 実装では JC 法のみが用いられてきた. 本研究では MCGS 法を用いることにより，反復回数の減少を促している．結果として異方性の問題において反復回数の減少が確認できた．しかし表 5 や表 6 からわかるように緩和法による収束時間の差が反復回数の差より小さいことがわかる．これは従来よりも計算速度が低下することにより全体の収束時間は長くなったからだ．ただし，問題規模が大きくなるほど計算速度の差は小さくなる．図 14 は GPUx1 版の反復ごとの計算時間を示している．問題サイズ最大では，MCGS 法を使用することによる反復ごとの計算時間の増加率は 1.5 倍までに抑えられているのがわかる．結果としてほぼ全体を通して MCGS 法を利用した解法が最も高速となっていた．

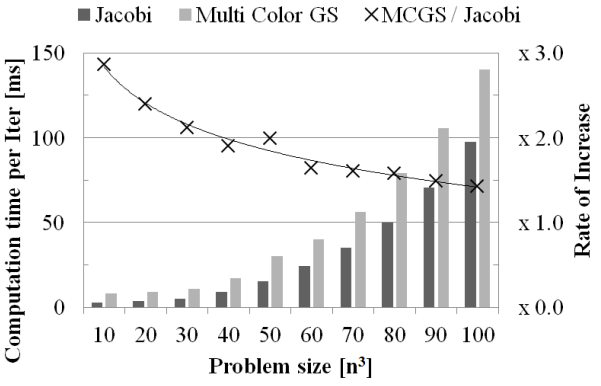

図 14　GPUx1 の反復ごとの計算時間
Fig. 14　Computation time per Iter of GPUx1.

表 5　GPUx1 計算結果
Table 5　Result for GPUx1.

size	Time [sec]		Iter		Omega	
	Jacobi	MCGS	Jacobi	MCGS	Jacobi	MCGS
10^3	0.06	0.11	25	14	0.4	1.1
20^3	0.08	0.12	24	16	0.4	0.9
30^3	0.15	0.14	29	13	0.4	0.9
40^3	0.33	0.25	35	14	0.4	1
50^3	0.49	0.41	30	13	0.4	0.9
60^3	0.77	0.58	24	11	0.4	1
70^3	1.22	0.89	26	12	0.4	0.9
80^3	1.77	1.28	26	12	0.4	1
90^3	3.06	2.14	45	15	0.4	1
100^3	4.10	2.37	45	13	0.4	1

表 6　GPUx2 計算結果
Table 6　Result for GPUx2.

size	Time [sec]		Iter		Omega	
	Jacobi	MCGS	Jacobi	MCGS	Jacobi	MCGS
10^3	0.14	0.12	28	11	0.4	0.9
20^3	0.19	0.22	31	16	0.4	1
30^3	0.17	0.16	27	13	0.4	1
40^3	0.24	0.19	28	14	0.4	1
50^3	0.37	0.29	29	13	0.4	0.9
60^3	0.59	0.59	29	15	0.4	1
70^3	0.76	0.63	25	14	0.4	1.1
80^3	1.10	0.90	34	13	0.4	0.9
90^3	1.48	1.10	26	13	0.4	0.9
100^3	2.32	1.78	30	15	0.4	1.1

表 7　CPU 計算結果
Table 7　Result for CPU.

size	Time [sec]	Iter	Omega
	SGS	SGS	SGS
10^3	0.23	21	0.8
20^3	0.36	28	0.8
30^3	0.44	21	0.8
40^3	0.85	26	0.8
50^3	1.42	25	0.8
60^3	3.28	22	0.8
70^3	5.96	23	0.8
80^3	9.38	23	0.8
90^3	12.72	25	0.8
100^3	18.62	25	0.8

Vol.2012-ARC-202 No.30
Vol.2012-HPC-137 No.30
2012/12/14

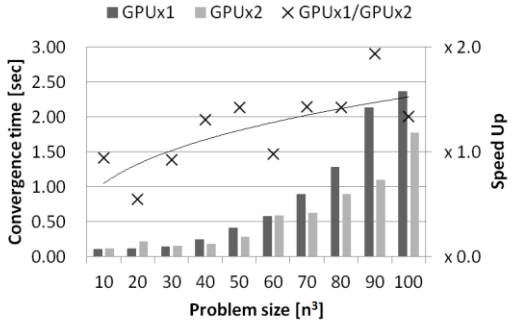

図 15　MCGS 法を用いた収束時間

Fig. 15　Convergence time with MCGS method.

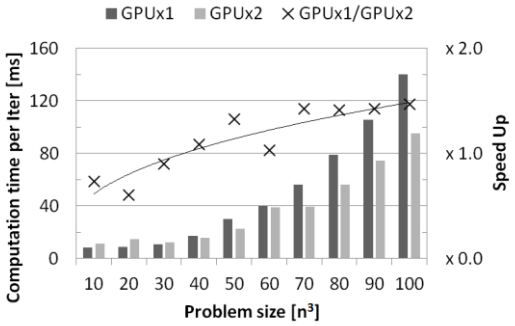

図 16　MCGS 法を用いた反復ごとの計算時間

Fig. 16　Computation time per Iter with MCGS method.

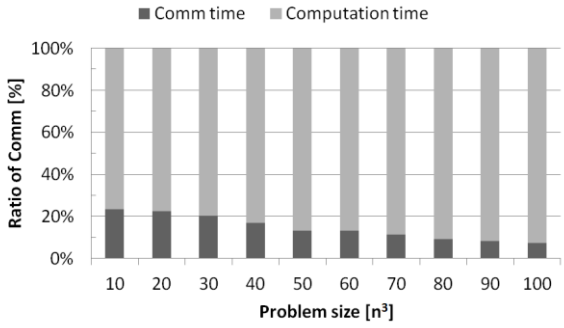

図 17　境界通信時間とその他の時間の割合

Fig. 17　Comm time and Other time.

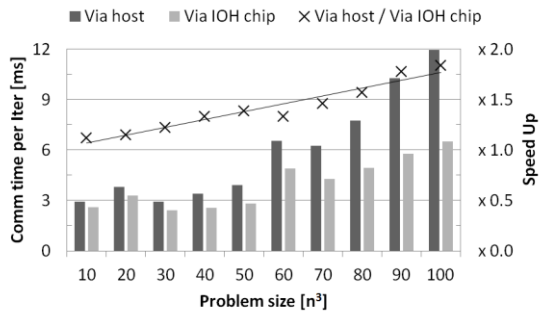

図 18　Via host 版と Via IOH chip 版の境界通信時間の比較

Fig. 18　Via host time and Via IOH chip time.

GPUx1 版と GPU x 2 版による比較を行う．ほぼ全体を通して緩和法には MCGS 法を用いた場合が最速であるため，緩和法に MCGS 法を用いた AMG-BiCGSTAB 法に焦点を絞って比較する．図 15 によると問題規模が大きくなるにつれ GPUx2 版が高速という結果になった．ここで GPU x 1 版と GPU x 2 版で反復回数が異なることに触れる．これは領域分割法により境界部分についてはアグリゲートを優先的に生成し，マルチレベルの構造が大きく変化するからだ．本数値実験ではこれにより反復回数が減少するという結果になった．

そこで反復回数を考慮し，反復ごとの計算時間を比較したものを図 16 に示す．問題規模に応じて安定して計算速度が上昇していることがわかる．問題規模 60^3 での速度低下について説明する．表 4 が示すように，プロセス番号ごとに階層の総数が異なり，最下層についての通信が発生しているためである．問題規模 20^3 についても同様である．GPU においても領域分割の方法が重要であることがわかる結果となった．

次に GPUx2 版の通信時間について触れる．図 17 は通信時間と計算時間の割合を示している．GPU 上で計算することにより，通信時間の割合が 10%を超えている．ゆえに通信時間がボトルネックとして表面化しやすいことがわかる．さらに通信時間の割合は問題規模に応じて減少が見られるが，これは三次元空間に問題規模が大きくなるのに対し，境界は二次平面で広がるからである．本数値実験では 2 領域分割を扱ったが，分割数が増えれば通信の割合は大きくなると考えられる．

最後にダイレクト通信の有無による差を示す．図 18 は通信経路 Via host 版と Via IOH chip 版の反復ごとの境界通信時間の比較である．計測では境界通信以外のすべての計算を行わず，この経過時間を通信時間とした．前述までの結果にはすべて Via IOH chip 版を採用している．これによると問題規模が増加することで上昇率が向上している．これは問題規模が大きくなるほど 1 回あたりの通信データのサイズが大きくなるためだ．問題規模 50^3 の最大通信データサイズが約 20KB なのに対し，問題規模 100^3 の最大通信データサイズは約 80KB に増加する．通信経路を Via IOH chip にすることで最大 1.8 倍の最適化効果が出る結果となった．

5. おわりに

本研究ではマルチ GPU を用いた AMG-BiCGSTAB 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し，JC 法

に有利な小さな簡単な問題でも性能差は小さい．MC 系 GS 法は異方性を持った問題では最も高速な解法となっている．

GPU 間の通信インタフェースにはダイレクト通信を用い，事前のリナンバリングにより，リオーダリングしたデータに対するスムーズな通信を実現した．従来のメインメモリを介した通信より最大で 1.8 倍の高速な通信が行える．三次元拡散方程式の異方性問題を用いて数値実験を行い，GPUx1 版と GPUx2 版 で比較を行ったところ，問題サイズ 30^3 までは GPUx1 版の場合が計算速度において優位だったが，より大きな問題サイズ になると GPUx2 版の場合が優位であることを示せた．

今後は大規模クラスタ環境での評価を目指す．同環境では計算対象を多数の領域に分割することで通信の割合が大きくなり，これがボトルネックとなる公算が高い．クラスタ環境ではダイレクト通信が使用可能な接続と不可能な接続が混在する．ダイレクト通信が可能な接続を有効に使うことで，解法全体の性能向上が期待できる．

謝辞 本研究の一部は，JST CREST「進化的アプローチによる超並列複合システム向け開発環境の創出」による．

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 (2011).

[4] 高橋光佑，藤井昭宏，小柳義夫: GPGPU を用いた AMG 法，情報処理学会 第 129 回ハイパフォーマンスコンピューティング研究会，Vol.2011-HPC-129，No.21 (2011).

[5] K. Takahashi, A. Fujii, and T. Tanaka.: *GPGPU-based Algebraic Multigrid Method*, In Proceedings of the 23rd IASTED International Conference on Parallel and Distributed Computing and Systems, Track: 757-061, ACTA Press, (2011).

[6] 高橋光佑，藤井昭宏，田中輝雄: マルチ GPU を用いた AMG 法，情報処理学会 第 133 回ハイパフォーマンスコンピューティング研究会，Vol.2012-HPC-133，No.29 (2012).

[7] 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 (2008).

[8] 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).

[9] 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).

[10] P. Vanek, J. Mandel and M. Brezina.: *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, COMPUTING, 56, pp.179-196 (1996).

[11] A. Fujii, A. Nishida and Y. Oyanagi.: *Evaluation of parallel aggregate creation orders: smoothed aggregation algebraic multigrid method*, International Federation for Information Processing, 172, pp.99-122 (2005).