jh190050-NAH

原子炉内熱流動解析コードの GPU 実装および適合細分化格子法の導入

小野寺 直幸（日本原子力研究開発機構）

原子炉内熱流動解析コード JUPITER の計算時間の大部分を占める Poisson 解法に対して、直交格子およびブロック型適合細分化（AMR）格子を用いた GPU 実装を開発した。直交格子上の GPU 実装については、TSUBAME3.0 における開発・最適化後に、Summit および ABCI にて大規模な強スケーリング性能測定を実施し、512 台から 2048 台の GPU を利用した強スケーリング性能測定において、共役勾配（CG）法と省通信 CG 法のそれぞれにて 3 倍の高速化を達成した。AMR 格子上の GPU 実装では、ブロック型データ構造に適した GPU 向け階層型逐次過緩和（RB-SOR）法をスムーザとするマルチグリッド法前処理付き CG（MGCG）法を新たに開発した。MGCG 法では、MG 法を適用しない手法と比較して、反復回数を 1/7 に、計算時間を 1/4 へと削減することに成功した。TSUBAME3.0 での強スケーリング性能測定では、8 台から 216 台の GPU を利用により 3.75 倍の性能向上を達成した。

1. 共同研究に関する情報

(1) 共同研究を実施した拠点名

東京工業大学学術国際情報センター

(2) 共同研究分野

□ 超大規模数値計算系応用分野

(3) 参加研究者の役割分担

- 代表者（日本原子力研究開発機構）：
  小野寺 直幸：解析手法の設計・開発
- 副代表者 （東京工業大学）：
  青木 尊之：大規模計算に関する助言
- 協力者（日本原子力研究開発機構）：
  井戸村 泰宏：計算結果の評価
  山下 晋：計算モデル実装および計算結果の評価
  山田 進：Poisson 方程式の反復解法の開発
  真弓 明恵：Poisson 方程式の反復解法の開発
- 協力者（東京大学）：
  下川辺 隆史：解析手法の高速化に関する助言

2. 研究の目的と意義

2.1 研究目的

過酷事故（SA）時における原子炉内溶融物の移行挙動の解明は、事故時の炉内状況把握および廃炉作業の効率化の観点から非常に重要である。しかしながら、既存の SA 解析コードは事前進展シナリオが予め与えられているなど多くの不確かさが含まれており、複雑な構造物で構成される原子炉内での現象把握が困難である。その課題に対して日本原子力研究開発機構（JAEA）では実験、シミュレーション、応用数学および計算機科学の専門家を含む学際的チームにより、多相多成分熱流動解析コード JUPITER の開発を進めている。JUPITER は多種金属および炉内構造物を含んだ溶融物解析が可能であり、炉内簡略模擬体系において SA を模擬した原子炉圧力容器および下部ペデスタルにおける溶融物の移行挙動解析が行われてきた。しかしながら、多相流体解析では圧力 Poisson 方程式がボトルネックとなり、実機体系の解析で必要となる計算の大規模化や高速化が困難である。この問題に対して、JAEA では CPU 向けの省通信型マルチグリッド（MG）法の開発により Oakforest-PACS において、計算性能の向上に成功している。

上記課題を発展させ SA の全体像を解明するには、高範囲・長時間の解析が必須となる。本課題では JUPITER コードの完全 GPU 化を行うことで、計算の更なる高速化を実現する。また、原子炉内の複雑な構造物に対して省メモリ・低計算コストの効率的な解析を実施するために、適合細分化格子（AMR）法の導入を実施する。以上の高

度化により、JUPITER コードの解析範囲が拡張され、事故時の炉内状況把握および SA 解析の高度化に大きく貢献できる。

2.2 研究の意義

SA 解析は材料力学、化学反応、流体力学等、広範な事象を含むため、各物理モデルを表現するのに十分な時空間解像度をカバーするマルチスケール解析が必須となる。しかしながら、数値的な取扱いが最も困難なシミュレーションの一つである多相流体解析を高解像度かつ長時間スケールに拡張できる計算手法はこれまで未確立であり、これが機構論的な SA 解析を阻害してきた。この長年の重要課題に対して、飛躍的に計算性能が向上してきた GPU とその性能を引き出す最先端の計算手法を活用することで、計算解像度および計算速度のブレイクスルーが期待できる。JUPITER は単相流や気液二相流にも適用可能であることから、SA 時の原子炉内溶融物の移行挙動解明はもちろんのこと、沸騰水型原子炉の安全性研究に係る幅広い熱流動解析に応用可能である。また、このような多相流体解析は幅広い工学問題に適用可能であることから、燃料電池内部、電子装置冷却用ヒートパイプの二相流体解析等、産業分野に与える影響も大きい。

3. 当拠点公募型共同研究として実施した意義

本研究を円滑に推進するためには、GPU カーネルの最適化や通信性能の向上が必須である。TSUBAME3.0 は複雑なネットワークの構造を持つため、P2P 通信や集団通信での NVLink や Intel Omni-Path に関する最適化に対して、東京工業大学の学術国際情報センターと知見の共有が必須である。2019 年度の前半に、GPU 計算と MPI 通信のオーバーラップ手法に関する開発を実施していたところ、計算ノードを頻繁に落とす事象が発生した。そのような問題に対して、学術国際情報センターには、迅速に Intel Omni-Path のドライバ等の更新をしていただき問題が解決した。このように、JHPCN 課題において運用側との連携に基づくことで、課題の円滑な推進が行えたことに、大きな意義がある。

4. 前年度までに得られた研究成果の概要

該当なし。

5. 今年度の研究成果の詳細

5.1 中間報告までの研究成果の概要

JUPITER では圧力 Poisson 方程式が計算時間の大部分 (9 割以上) を占めており、その高速化が最も重要な課題となっている。Poisson 方程式は Staggered 配置に対して、２次精度の中心差分法を用いて離散化されている。

$$\frac{\partial}{\partial x_i}\left(\frac{1}{\rho}\frac{\partial p}{\partial x_i}\right)=\frac{1}{\Delta t}\frac{\partial u_i^*}{\partial x_i}$$

ここで、pは圧力、ρは液相の体積率（VOF）から求められる密度、Δtは時間刻み幅、u^*は非圧縮性 Navier-Stokes 方程式の移流・拡散項後の中間時刻での速度、添字iはアインシュタインの縮約表記で各次元を表す。２次精度離散化では、左辺は 7 重ブロック対角行列となる。

気液多相流体解析では、左辺の気液密度比が 1:1000 と非常に大きくなり行列の性質が悪くなる。そのような問題に対して、CPU 解法として、チェビシェフ基底 (CB) 省通信共役勾配 (CG) 法と修正不完全 LU 分解 (D-ILU) 法前処理を組み合わせた手法 (P-CBCG) および倍精度・単精度を組み合わせた MG 法前処理に基づく CG 法(MGCG) を開発し、Oakforest-PACS において、2000 台の CPU（Intel KNL）にて高速化に成功している[Y. Idomura, ScalA, 2018]。

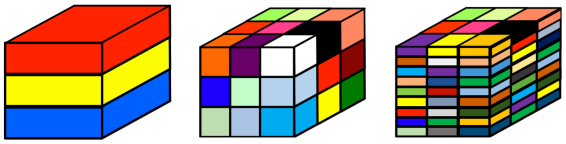
5.1.1 GPU に適した P-CG 法および P-CBCG 法の実装

2019 年度の前半は、直交格子上において圧力 Poisson 方程式解法の GPU 最適化およびノード間通信の高速化を行なった。Krylov 部分空間法として、CG 法および集団通信回数の削減が可能な CBCG 法を採用すると共に、前処理手法として D-ILU 法に Block Jacobi 領域分割を組み合わせた。

D-ILU 法は格子間の計算順序の依存関係があるため Block Jacobi の領域（ブロック）内は一つの

スレッドにて逐次処理される。一方で、各ブロックは独立に処理されるため、並列処理が可能となる。数十コアの CPU と数千コアの GPU では、Block Jacobi の領域サイズが異なり、収束性能と計算性能のトレードオフが存在する。図 1 に CPU および GPU 向けの領域分割を示す。コア数の少ない CPU では z 方向の 1 次元分割を採用し、コア数の多い GPU では 3 次元分割による細かな領域に分割した。

表 1 に Block Jacobi 領域分割の違いによる、JUPITER の小規模問題での収束回数と計算時間について示す。Case 1 は CPU 計算であり、大きな領域の確保により収束回数が最も少ない結果となった。一方で、Case 2 から Case 9 は GPU 計算となり、細かな領域分割により、Case 1 と比較して 1.4 倍から 2.5 倍程度に収束回数が増加した。



(a) z-division (b) xyz-division (c) zxy-division

図 1 Block Jacobi 領域分割。(a)CPU 向けの 1 次元領域分割、(b)GPU 向けの 3 次元領域分割、(c)GPU 向けの連続メモリアクセスを可能とする粒度の細かい領域分割。

表 1 Block Jacobi 領域分割の違いによる、収束回数と計算時間の関係（格子点数 256×128×512）。xyz 領域分割での計算時間は t_1 。転置した zxy 領域分割の計算時間は t_2 。CPU として Intel Broadwell 14thread、GPU として NVIDIA V100 を使用。

Case	x	y	z	# of iterations	t_1	t_2
1. (CPU)	256	128	36	106	34.60	-
2.	16	32	4	142	17.49	9.45
3.	16	16	4	145	38.62	7.64
4.	8	8	16	156	13.04	39.44
5.	16	16	1	161	48.69	7.27
6.	8	8	1	174	47.34	6.87
7.	4	4	4	185	22.83	22.83
8.	2	4	64	210	8.70	80.67
9.	1	8	8	263	10.79	69.13

また、領域分割の違いにより収束までの計算時間が 10 倍以上の差となることが確認された。図 1(b) のような 3 次元ブロック形状の領域分割に比べ、図 1(c)の 2 次元タイル形状の領域分割では連続メモリアクセスによって大幅に処理効率が向上する。最も高速な条件は Case 6 となり、CPU と比較して 60%程度の反復回数の増加となったが 5 倍の高速化が達成された。

GPU を用いた並列計算では、GPU 内のメモリ帯域とノード間の通信帯域の比が大きいため、通信時間を削減する技術が必要となる。集団通信の削減に関しては、P-CBCG 法を適用することで解決できる。一方で、袖領域の 1 対 1 通信に関しては、GPU 計算と MPI 通信のオーバーラップ手法が有効である。本研究では、通信と無関係な中心領域(Core)と、各軸方向の 6 つの袖領域(Surface)に分割し、それぞれに対して GPU カーネルを割り当てた（図 2)。

表 2 に P-CG 法での AXPY と SpMV カーネルを組み合わせたオーバーラップ手法の手順を示す。最初に AXPY の通信に関わる Surface 部分を計算する（Step 1)。その後に、AXPY にて更新した値の非同期袖通信の実施、および、AXPY と SpMV

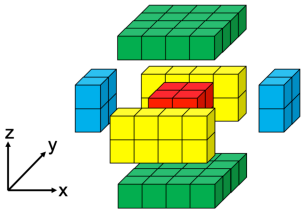


図 2 GPU 計算と MPI 通信のオーバーラップ手法での計算領域の分割。

表 2 P-CG 法での AXPY と SpMV カーネルを組み合わせたオーバーラップ手法。

Step	Stream name	operation
1	default	Surface parts of AXPY
2a	default	Halo data communication using Isend/Irecv
2b	calc	Core parts of AXPY and SpMV
3	default	Wait for Isend/Irecv to finish
4	default	Surface parts of SpMV

の Core 部分を計算する（Step 2a, 2b）。最後に、通信の完了を確認（Step 3）した後に、Surface 部分の SpMV を実行する。前処理においても、計算領域を Core と Surface 部分に分けたオーバーラップが実施可能であるが、領域分割による計算カーネルの性能劣化が顕著であったため適用していない。P-CBCG 法においても、Krylov 部分空間の計算にて同様の最適化を実施した。

5.1.2 GPU スーパーコンピュータでの性能測定

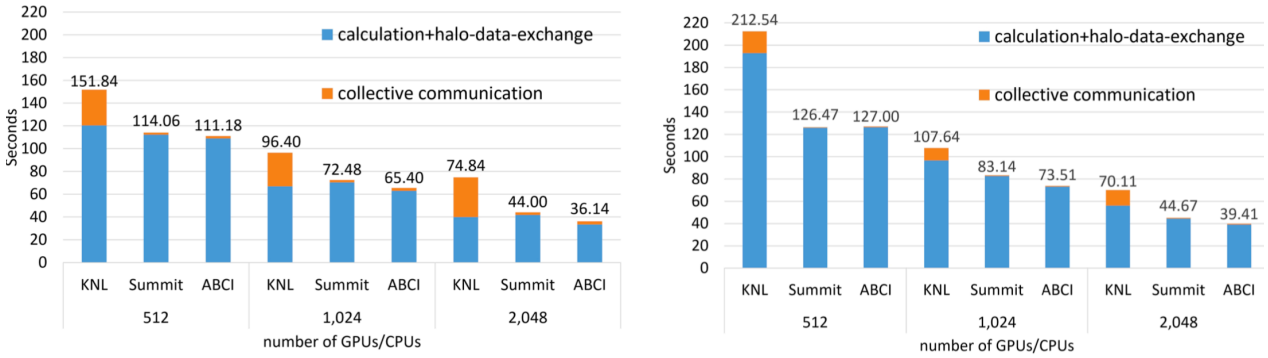
最新の GPU（NVIDIA TESLA V100）を搭載する GPU スーパーコンピュータ Summit および ABCI、並びに CPU スーパーコンピュータ Oakforest-PACS（Intel KNL）にて、強スケーリングの性能測定を実施した。図 3 に解析結果を示す。青色が計算および隣接通信時間、燈色が集団通信時間となる。計算結果より、CPU/GPU の解析において良いスケーリングが得られていることが確認できる。P-CBCG 法の解析では、P-CG 法と比較して、燈色の集団通信時間が削減できており、集団通信時間が長い KNL に対して、非常に有効であることが確認できる。P-CBCG 法の 512 並列から 2048 並列の計算において同数の CPU/GPU 性能を比較すると、Summit にて 1.2 ~ 1.6 倍の高速化、ABCI にて 1.4 ~ 1.7 倍の高速化が達成された。得られた成果は、SC19 のワークショップの採択された[業績 7-1]。

5.2 中間報告以降の研究成果の概要

2019 年度の後半は、研究代表者のグループが GPU 向けに開発を進めていたブロック型 AMR 法のフレームワーク[N. Onodera, ScalA, 2018]を JUPITER に適用した。AMR 法の木構造内のデータ構造として、Leaf に 8^3 格子を割り当てたブロック型データ構造を採用することで、十分な数の CUDA スレッドによる連続的なメモリアクセスを可能とした。圧力 Poisson 解法として、GPU スーパーコンピュータでは集団通信のコストが支配的ではなかったため、P-CG 法を選択した。前処理手法として、当初は収束性が高く堅牢な D-ILU 法の実装を検討していたが、表 1 の結果からブロック型データ構造で採用している Leaf サイズが 8^3 の格子では、十分な収束性および計算性能が得られないことが予測されたため、MG 法と逐次過緩和（RB-SOR）法を組み合わせた手法を採用した。次節に MG 法を採用した前処理手法の詳細を示す。

5.2.1 ブロック型データ構造でのマルチグリッド前処理手法の実装

ブロック型 AMR 法のフレームワーク上において図 4 に示す 3 段の V-cycle MG 法を実装した。簡略化のために、全ての MG 法の解像度において同じ木構造を共有し、それぞれの Leaf 内の格子を MG Lv=0 は 8^3、MG Lv=1 は 4^3、MG Lv=2 は 2^3 と設定した。



(a) P-CG cross platform results

(b) P-CBCG cross platform results

図 3 P-CG および P-CBCG 法の強スケーリング性能測定。
（格子点数：1,280x1,280x4,608、計算機：Oakforest-PACS, Summit, ABCI）

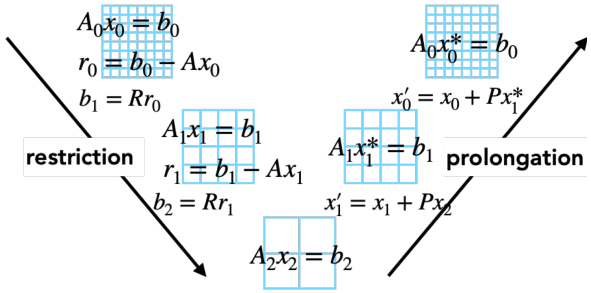


図 4　ブロック型 AMR 法の Leaf における 3 段の V-cycle マルチグリッド法

表 3 マルチグリッド法における各解像度での前処理手法

MG Lv.	smoother	iteration	subiter
0	RB-SOR	4	1 or 16
1	RB-SOR	8	1 or 8
2	RB-SOR	64	1
1	RB-SOR	8	1 or 8
0	RB-SOR	2	1 or 16

MG 法と組み合わせる前処理手法として RB-SOR 法を採用した。CUDA のスレッド・ブロックの割り当てとして、MG Lv=0 および Lv=1 では Leaf 内の格子点数と同数のスレッド数を持つブロックを設定した。Leaf 内の格子点数の少ない MG Lv=2 では、8 つの Leaf に対して 1 つの CUDA ブロックを設定した。更に、収束性能の向上を目指して、木構造を構成する Leaf および Leaf 内の格子のそれぞれに階層的に色つけを行った階層型 RB-SOR 法を実装した。階層型 RB-SOR 法では、Leaf の階層で順序付けを行った後に、Leaf 内の格子に対して RB-SOR 法の反復計算を行う。ここで、Leaf 内の反復計算において、更新される物理量を GPU の共有メモリ上に保持することで、ヒープ領域を参照しない高速な計算が可能となる。

並列計算の高速化として、GPU 計算と MPI 通信のオーバーラップ手法を導入した。ブロック型 AMR 法でのオーバーラップ手法では、Leaf を通信に関係する Surface 部分と無関係の Core 部分に分類し、それを (1) MPI 通信と Core 部分の Leaf の GPU 計算、(2) MPI 通信の完了待ち、(3)Surface 部分の Leaf の GPU 計算、の順に実行した。オーバーラップ手法は、CG 法の SpMV カーネルおよび前処理手法（RB-SOR 法）に対して適用した。

5.2.2 ブロック型データ構造での計算カーネルの性能測定

TSUBAME3.0 にて、MG-CG 法を構成する計算カーネルの性能測定を実施した。計算条件として、48^3 個の Leaf（約 56.6×10^6 格子）を設定した。表 4 に MG-CG 法を構成する主な計算カーネルの性能を示す。AXPY はベクトル和、SpMV は行列・ベクトル積、Dot product は GPU 内でのベクトルの内積となる。本計算では、CG 法で更新される変数は倍精度（FP64）、SpMV の 7 重ブロック対角行列は単精度（FP32）を採用した。また、ベクトルの内積の実装として、Warp 内（32 スレッド）の WarpShuffle 命令を用いたバタフライ演算による足し合わせ(shfl)、および共有メモリの利用による CUDA ブロック内 (~1024 スレッド) の足し合わせ（縮約）による最適化(opt)を実施した。

測定結果より、AXPY および SpMV において、TESLA P100 の理論メモリ帯域である 732 GB/s の 6 割程度の性能が得られることが確認された。ベクトル内積では、WarpShuffle 命令に加えて共有メモリを駆使した最適化により、240 GB/s から 506 GB/s の 2 倍以上の性能向上を達成した。

表 5 に前処理手法の主なカーネルである RB-SOR 法の性能を示す。ここで # of subiter は、階層型 RB-SOR 法の Leaf 内での反復回数となる。測定結果より、Leaf size が 8^3 や 4^3 の計算においては、十分なスレッド数の確保および連続メモリアクセスが可能により、理論メモリ帯域の 5 割以上の十分なメモリ帯域を引き出せていることが確認できる。Leaf size が小さい 2^3 の条件においては計算性能が低下するが、MG 法による格子点数の削減により 1 ステップに必要な計算時間を短縮した。表 5 の下段に階層型 RB-SOR 法の性能を示す。CUDA の共有メモリの利用により、Leaf size が 8^3 の条件において、16 回の追加の反復計算を通常の RB-SOR 法の 2.5 倍程度の計算時間にて実現した。CG 法と組み合わせた収束性能については次節に示す。

表 4　CG 法の計算カーネルの実行性能。1 ステップの計算時間 (time) およびヒープ領域から読み書きされるメモリ帯域幅（BW)。

kernel	precision	time(msec)	BW(GB/s)
AXPY	FP64	2.79	487.8
SpMV	FP64/FP32	5.88	423.5
Dot product(shfl)	FP64	3.77	240.2
Dot product(opt)	FP64	1.79	506.8

表 5　前処理における計算カーネルの実行性能。1 ステップの計算時間 (time) およびヒープ領域から読み書きされるメモリ帯域幅（BW)。

Leaf size	# of subiter	time(msec)	BW(GB/s)
8 (Lv.0)	1	5.71	396.4
4 (Lv.1)	1	0.78	363.1
2 (Lv.2)	1	0.43	82.3
8 (Lv.0)	16	14.31	158.3
4 (Lv.1)	8	1.30	217.0

5.2.2 MG-CG 法の収束性能測定

MG 法を適用した提案手法に対して、収束性能を測定した。計算領域 $(x, y, z) \in (-1 \sim 1)$ に対して、Leaf 数 $= 96^3$（格子点数 約 453×10^6）を設定し、8 台の GPU を使用した。圧力 Poisson 方程式の係数行列として、原点から距離 0.2 の領域の密度を周辺の 1000 倍に設定し、右辺は $b = -((2\pi)^2 + (4\pi)^2 + (8\pi)^2) \cdot \cos(2\pi x) \cdot \cos(4\pi y) \cdot \cos(8\pi z)$ と設定した。MG 法での前処理として、表 3 に示す RB-SOR 法の反復および 階層型 RB-SOR 法による Leaf 内の内部反復を適用した。計算手法として、P-CG 法と細かな格子（Lv.0）での RB-SOR 前処理（pcg)、MG 前処理の適用（pcg mg)、更に階層型 RB-SOR 法の内部反復の適用（pcg mg sub)、および、単精度の前処理 (pcg mg sub fp32）の 4 つを比較した。

図 5 に各手法の収束履歴を示す。MG 法を適用しない pcg においては収束に 700 回の反復計算が必要となる。一方で、MG 法を適用した pcg mg においては 300 回程度、更に内部反復を行なった pcg mg sub では 100 回程度の反復回数にて収束した。また、単精度の前処理を用いた手法では倍

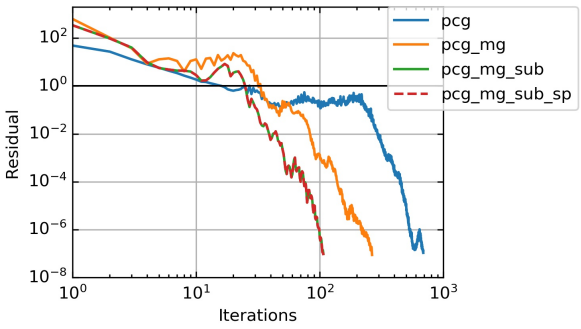


図 5　MG-CG 法の収束履歴（収束回数、残差)。

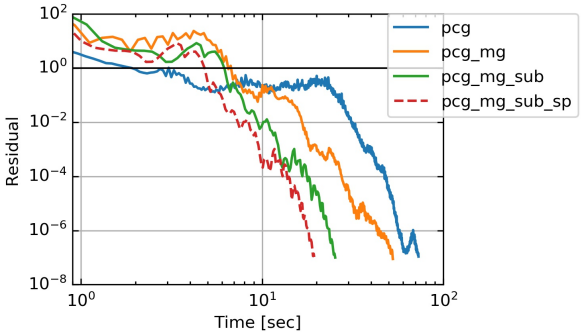


図 6　MG-CG 法の収束時間履歴（時間、残差)。

精度と同じ収束履歴が得られることが確認された。

図 6 に各手法の計算時間に対する収束履歴を示す。測定結果より、MG 法を適用することで pcg の 80 秒から pcg mg の 55 秒程度へと計算時間の短縮に成功した。更に、内部反復の適用により pcg mg sub の 25 秒への削減、および単精度の採用による pcg mg sub fp32 の 20 秒への更なる高速化が達成された。

5.2.2 MG-CG 法の強スケーリング性能測定

階層型 RB-SOR 法（pcg mg sub fp32）に対して、8 台から 216 台の GPU を用いた強スケーリング性能測定を実施した（図 7)。計算格子として Leaf 数 $= 96^3$（格子点数 約 453×10^6）を設定した。測定時間の内訳として、青色が CG 法の計算、赤色が内積での集団通信、緑色が前処理手法となる。オーバーラップ手法無しと有りを比較すると、MPI 通信の隠蔽が容易な 8 台の GPU を用いた条件においてオーバーラップ手法の適用により 10% 程度の性能向上が実現された。一方で、216 台の

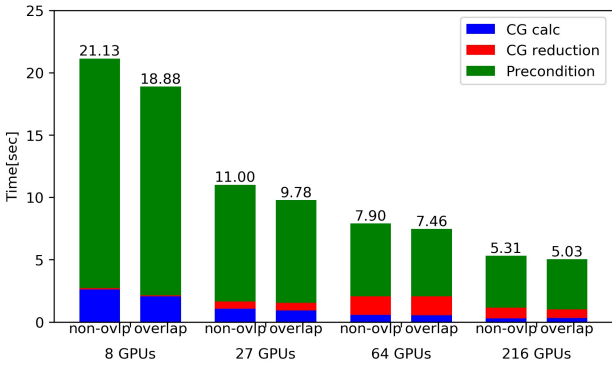


図 7　MG-CG 法の強スケーリング性能測定。(左) オーバーラップ無し、(右) オーバーラップ有り。Leaf 数 96^3 (格子点数 約 453×10^6)。

GPU を用いた計算では、5%程度の性能向上に留まった。これは、袖領域の Leaf 数に対して内部の Leaf 数の割合が少なくなるため通信隠蔽が困難であること、および、集団通信の遅延が原因であると考えられる。オーバーラップ手法有りの全体の計算時間を比較すると、8 GPU の 18.88 秒に対して、27 GPU では 9.78 秒 (×1.9)、64 GPU では 7.45 秒 (×2.5)、216 GPU では 5.03 秒 (×3.75) が得られた。8 GPU と 216 GPU の計算時間の内訳として、8 GPU においては集団通信時間が 0.06 秒で全体の 1%以下に対して、216 GPU においては 0.69 秒で全体の約 14%に増加した。また、64 GPU において不自然に集団通信時間が増大しているが、数回の測定においても同様の傾向が見られたため、原因を特定できなかった。今後の Poisson 解法の高速化の課題として、通信隠蔽の更なる最適化 (複数カーネルの融合) および集団通信削減手法 (CBCG 法) 等の採用による、強スケーリング性能の向上が挙げられる。

［参考文献］

1. Y. Idomura, T. Ina, S. Yamashita, N. Onodera, S. Yamada, and T. Imamura, Communication avoiding multigrid preconditioned conjugate gradient method for extreme scale multiphase CFD simulations , Proc. of 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), (2018)
2. N.Onodera, Y.Idomura, Y.Ali, and T.Shimokawabe, Communication Reduced Multi-time-step Algorithm for Real-time Wind Simulation on GPU-based Supercomputers, 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), (2018)

6. 今年度の進捗状況と今後の展望

原子炉内熱流動解析コード JUPITER の計算時間の大部分を占める Poisson 解法の GPU 実装およびブロック型 AMR 法を適用した Poisson 解法を新たに開発した。直交格子上での Poisson 解法においては、TSUBAME における性能測定後に、Summit および ABCI にて大規模な強スケーリング性能測定を実施し、512 台から 2048 台の GPU を利用した強スケーリング性能測定において、P-CG 法および P-CBCG 法のそれぞれにて 3 倍の高速化を達成した。得られた成果は、SC19 のワークショップに採択された[業績 7-1]。

ブロック型 AMR 法での Poisson 解法として、MG 法の適用および GPU の共有メモリを活用した階層型 RB-SOR 法を提案した。上記の手法の適用により、反復回数を 1/7 に、計算時間を 1/4 に削減することに成功した。また、8 台から 216 台の GPU を利用した強スケーリング性能測定において、3.75 倍の性能向上を達成した。

研究計画で実施できなかったものとして、JUPITER の流体計算カーネルのブロック型 AMR 法への GPU 移植、および、その流体解析の実現が挙げられる。原因として、AMR 法でのマルチグリッド Poisson 解法の開発に難航したことが挙げられる。今後の展望として、引き続き流体計算カーネルの移植を進め、計画していた原子炉内の多相流体解析を実施していく予定である。

7. 研究業績一覧

(1) 学術論文

(2) 国際会議プロシーディングス

1. Y. Ali, N. Onodera, Y. Idomura, and T. Ina, GPU Acceleration of Communication

Avoiding Chebyshev Basis Conjugate Gradient Solver for Multiphase CFD Simulations, Proc. of 10th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA 2019), pp. 1-8, (2019), 査読有り

(3) 国際会議発表

2. Y. Ali, N. Onodera, Y. Idomura, Y. Hasegawa, and T. Ina, Performance portability of large scale distributed Krylov solvers with OpenACC and CUDA, OpenACC Annual Meeting 2019, (2019) , 査読無し

(4) 国内会議発表

3. 小野寺 直幸, 井戸村 泰宏, ユスフ アリ, 山下 晋, 伊奈 拓也, 今村 俊幸, GPU による多相流解析コード JUPITER の Poisson 方程式の高速化(口頭発表), 第 33 回数値流体力学シンポジウム, (2019), 査読無し
4. 小野寺 直幸, 井戸村 泰宏, ユスフ アリ, *下川辺 隆史, 青木 尊之*, ブロック型適合細分化格子での Poisson 解法の GPU 高速化, 第 25 回計算工学講演会, (2020), 査読無し

(5) その他（特許，プレス発表，著書等）