jh180061-NAH

高精細計算を実現する AMR 法フレームワークの高度化

下川辺　隆史（東京大学）

概要　格子に基づいたシミュレーションでは、広大な計算領域の場所によって求められる精度が異なる問題に有効な手法が要求されてきており、高精度が必要な領域を局所的に高精細にできる適合細分化格子（AMR）法が有効である。本研究課題では、GPU スパコンに適した動的負荷分散手法、通信削減技術、時間発展の最適化などを開発し、前年度課題で構築した AMR フレームワークに導入することで、フレームワークのさらなる高度化を目標とする。複数 GPU 間における通信を効率化するために通信をまとめて実行する時間ブロッキング法をフレームワークへ導入した。また、複数 GPU 計算で計算負荷を均等するために、動的負荷分散技術を導入した。AMR 法を導入した圧縮性流体計算で、東京工業大学の TSUBAME3.0 の 288GPU を用い 84%の並列化効率を達成した。また、本課題で目標とする流体中を流れながら成長する金属凝固計算では、流体計算は格子ボルツマン法を利用する。この格子ボルツマン法は時間更新幅が格子幅の定数倍に固定される計算手法であるために、解像度ごとに時間ステップ幅が異なり、時間方向にも物理量の補間が必要となる。フレームワークを時間方向の物理量の補間計算に対応させ、AMR 法を適用した格子ボルツマン法の計算を実現した。

1.　共同研究に関する情報

(1) 共同研究を実施した拠点名

東京大学、東京工業大学

(2) 共同研究分野

■　超大規模数値計算系応用分野

□　超大規模データ処理系応用分野

□　超大容量ネットワーク技術分野

□　超大規模情報システム関連研究分野

(3) 参加研究者の役割分担

- 下川辺 隆史（東京大学 情報基盤センター）：研究全体の統括、AMR 法フレームワークの完成と流体中を流れ成長する金属凝固計算への適用
- 高木 知弘（京都工芸繊維大学 機械工学系）：流体中を流れ成長する金属凝固計算コードの提供と AMR 法導入の検討
- 青木 尊之（東京工業大学 学術国際情報センター）：AMR に対応した数値計算手法に関する助言と最適化手法の検討
- 小野寺 直幸（日本原子力研究開発機構システム計算科学センター）：格子ボルツマン法に関する助言
- 星野 哲也（東京大学 情報基盤センター）：GPU 向け最適化手法に関する助言

2.　研究の目的と意義

2.1.　研究の目的

申請者らは気象計算や金属材料の凝固成長計算など格子に基づいたアプリケーションを GPU で実行する研究に取り組んできた。GPU は、従来のプロセッサの CPU と比べて計算の処理能力が非常に高く、消費電力当たりの演算性能が高いため、様々なスパコンでは GPU を大規模に搭載している。

GPU によるステンシル計算は高い性能が得られるものの、それにはアーキテクチャに適した高度な最適化を導入する必要がある。これを簡便に導入し生産性を高めるため、申請者らは、通常の C++ を記述するだけで、GPU と CPU で高速実行できる高性能・高生産フレームワークを構築してきた。これを用い、高性能な気象計算コードを開発に成功し（平成 26 年度課題）、都市気流計算コードを高生産に開発を進め（平成 27 年度課題）、フレームワーク技術は様々な GPU 実アプリケーションを高生産に開発する上で有効であることを示した。

近年、大規模 GPU 計算が可能となり、広大な計算領域の場所によって求められる精度が異なる問題に有効な手法が要求されている。GPU 計算では、GPU が得意なステンシル計算を活用しながら、高精度が必要な領域を局所的に高精細にできる適合細

分化格子法(Adaptive Mesh Refinement; AMR 法)が有効である。これまでに開発したステンシル計算用フレームワークを基盤として、単一 GPU 用 AMR フレームワークを完成させ(平成 28 年度課題)、これを発展させ GPU/CPU/Xeon Phi へ対応した AMR フレームワークの構築を進めた(平成 29 年度課題)。

本年度は、GPU スパコンに適した動的負荷分散手法、通信削減技術、時間発展の最適化などを開発し、前年度課題で構築した AMR フレームワークに導入することで、フレームワークをさらなる高度化し、より大規模な計算を目指す。動的負荷分散では、まず複数の木構造を用いたデータ構造を活用した手法を構築する。通信をまとめる技術や通信の隠蔽手法、格子解像度毎に適した時間ステップ幅で計算を行うことで高速化を図る。さらに、機械学習を基盤に、様々なデータ移動と格子配置の変形を試行させ、アプリケーションの実行時間を最小とする最適な負荷分散を学習し、これに基づいた動的負荷分散手法の確立を目指す。

次に、開発した AMR 法フレームワークを、流体中を流れながら成長する金属凝固計算に高度化したフレームワークを適用し、GPU スパコン上において流体中の金属凝固成長計算の大規模計算を実現することを目指す。

2.2. 研究の意義

ペタスケールのスパコンでは、低消費電力かつ高性能を達成するため数千台を超える GPU が搭載され、日本、米国、中国などで稼働している。格子計算はスパコンを利用する代表的なアプリケーションで、局所的に高精細な大規模計算を実現させる意義は大きい。米国エネルギー省は、AMR 法は所謂「エクサスケール」でのマルチスケール問題解決の鍵となる技術と位置付けている。エクサスケールに向けて、通信やデータ移動の少ないアルゴリズムの開発は必須であり、スパコンに必須な通信隠蔽・削減技術と併用して、高性能 AMR 法を達成する試みは意義は大きく、世界での注目は間違いない。

本研究は、個別アプリケーションに特化した AMR 法を構築するものでなく、効率的で汎用性の高い動的負荷分散技術を開発し、通信削減技術と時間発展の最適化を併用することで、AMR 法フレームワークを高度化し、多くのアプリケーションに適用できる技術とすることを目標とする。複数のアプリケーションに適用できる動的負荷分散手法や汎用フレームワークの構築には、性質の異なる計算手法への適用が重要であり、本課題では、圧縮性流体計算へ適用し、また流体中を流れながら成長する金属凝固計算という異なる二つの計算手法を併用した計算への適用を目指す。開発する手法や汎用フレームワークは実用に耐えうるものになると確信している。最終的には、これを一般に公開する予定で、誰からも使える局所的に高精細が必要な大規模 GPU アプリケーションの開発を支援する基盤技術となり、波及範囲は広い。

3. 当拠点公募型共同研究として実施した意義

本研究は、GPU を搭載したスパコン上で、局所的に高精細を実現する高性能・高生産 AMR フレームワークを開発し、これを用いて局所的に高解像度となる流体中を流れながら成長する金属凝固成長計算を実現する。GPU を搭載する東京工業大学の TSUBAME3.0 と、東京大学の Reedbush-H は本研究を遂行する上で最適な計算機環境である。本研究は、 流体中を流れ成長する金属凝固成長計算に AMR 法を適用するだけでなく、その開発技術をフレームワークとして汎用的に適用できる技術とする。有用なフレームワークの構築には、高度な計算機科学の知見に合わせて、様々な実アプリケーションへの適用が重要となる。本研究は、大規模アプリケーションの専門とする下川辺（東大）が中心となり、凝固成長計算を専門とする高木（京都工芸繊維大)、流体計算の専門とする青木（東工大)、小野寺（原研)、計算機科学を専門とする星野（東大）が共同研究を行うことでアプリケーション開発者の立場から様々なアーキテクチャに対応した実用に耐えうる AMR 法フレームワークを構築する。このように効果的に共同研究を遂行することにより、着実に成果を出すことができている。

4. 前年度までに得られた研究成果の概要

本課題では、平成 26 年度課題でステンシル計算用の GPU コンピューティング・フレームワークの基本機能を開発した。フレームワークは、ユーザが記述した格子計算のコアとなる格子点を更新するステンシル関数から最適化された GPU 実行コードを生成する。これを用い気象庁が開発する気象計算コード ASUCA を GPU スパコンへ実装した。ユーザコードに GPU 固有の最適化を記述せずに、GPU スパコンに最適化されたコードを開発した。東京工業大学の GPU スパコン TSUBAME2.5 で実行し、高い生産性・可搬性を実現しながら、高い実行性能を達成し、国際会議 SC14 で発表した。

平成 27 年度は、平成 26 年度課題で開発した構造格子用 GPU フレームワークを高度化し、単一のユーザコードを GPU および CPU などの様々なアーキテクチャで高速実行する機構などを導入した高生産フレームワークを完成させた。CPU 計算では OpenMP による並列計算に対応した。

平成 28 年度は、平成 27 年年度課題で開発した GPU および CPU に対応した構造格子用フレームワークを発展させ、AMR 法フレームワークの構築に必要となるデータ構造とそれに対する計算機構の開発を行い、単一 GPU 用 AMR 法フレームワークを構築した。AMR 法に対応した汎用的なデータ構造やそのデータ構造に対して高速に計算できるコードを簡便に記述できる機構などを開発し、フレームワーク全体とそのプログラミングモデルを決定し、構築を進めた。

前年度（平成 29 年度）は、平成 28 年年度課題で開発した AMR 法フレームワークを Xeon Phi へ対応させた。既存のフレームワークは式テンプレートを多用しており、このために Xeon Phi では高い性能が得られないことが判明し、C++11 のラムダ式を用いてステンシル計算用フレームワークおよび AMR 法フレームワークを再構築した。複数 GPU 計算に対応した AMR 法フレームワークを試作したが、再構築に時間を要したため、通信の効率化などが行えておらず、実行性能向上の課題が残った。

5. 今年度の研究成果の詳細

今年度は、動的負荷分散技術および通信削減技術を開発し、導入することでフレームワークの高度化を進めた。特に、動的負荷分散では複数の木構造を用いた AMR 法データ構造に着目し、木構造を順に辿りデータを割り当てることで、それぞれの木構造が局所的に分布することを活用し、データも局所化させることが可能となった。またカウントダウン方式の時間ブロッキング法を導入することで、ユーザコードの変更なく、通信をまとめて実行することが可能となり、性能向上を実現した。最終的には、3 次元圧縮性流体計算において、東京工業大学の TSUBAME3.0 の 288 GPU を用い、84%の並列化効率を達成することに成功した。この成果は国際会議 ICCS で発表する [2]。

フレームワークの適用を目指した流体中を流れ成長する金属凝固成長計算の流体計算は、格子ボルツマン法を用いる。この格子ボルツマン法は時間更新幅が格子幅の定数倍に固定される計算手法であるために、解像度ごとに時間ステップ幅が異なり、時間方向にも物理量の補間が必要となる。今年度後半に、フレームワークを時間方向の物理量の補間計算に対応させ、AMR 法を適用した格子ボルツマン法の計算を実現した。現在、AMR 法を適用したフェーズフィールド法とのカップリングを進めている。

当初、機械学習による動的負荷分散の最適化の検討も進めたが、AMR 法の大規模計算では動的負荷分散よりも各ステップでの GPU 間通信のコストが高く、この削減に注力した。機械学習を用いた最適化手法においても各ステップでの通信削減を考慮した最適化手法が必要であることがわかり、検討を進めた。

本 AMR 法フレームワークを適用した圧縮性流体計算は、その可視化を含め研究の重要性が評価され、第 23 回計算工学講演会グラフィックスアワード特別賞 4 賞のうち 2 賞を受賞した[5,6]。

以下では具体的な研究成果について説明する。

5.1 AMR 法フレームワーク

前年度までに構築した AMR 法フレームワークを高度化する。AMR 法データ構造の基盤と AMR 法フレームワークのコアとなるステンシル計算関数の定義と実行については前年度課題で開発したもので、本年度の前半ではこれを拡張し、通信をまとめて通信回数を削減する技術や動的負荷分散などを導入し、大規模 GPU 計算に対応する AMR 法フレームワークとした。

5.1.1 AMR データ構造と複数 GPU における管理の概要

本フレームワークの対象とする AMR 法では構造格子を再帰的に細分化し、その空間的配置を木構造で表す。 木構造の各リーフノードには、一つの格子ブロックを割り当てる。格子ブロックは典型的には 2 次元で 16 × 16 格子程度である。3 次元計算では八分木、2 次元空間では四分木となる。GPU は連続したメモリ領域へアクセスするときに高い実行性能となるため、格子ブロックは物理変数ごとに一つの大きな連続メモリ領域に確保する。各リーフノードは、直接は格子ブロックを保持せず、格子ブロックを特定する ID を保持する。この ID から割り当てられた格子ブロックの連続メモリ領域における位置を求め、格子ブロック上のデータを参照する。

複数 GPU による計算では MPI で並列化され、各プロセスが計算領域全体の木構造を表現する `Field` 型データ を保持する。各 MPI プロセスで発行する(1) 各リーフに対する解像度の変更の指示、 (2) 特定のリーフのデータをある GPU から別の GPU へ移行させる指示、は発行後に、MPI 通信によって全プロセスで共有する。共有後に、各プロセスは自分の保持する計算領域全体を表現する木構造を変更する。各プロセスの木構造自身は通信により明示的に同期することはないが、木構造を変更する上記の二つの「指示」を共有することで、全プロセスは常に同一の木構造を保持する。この計算領域全体の木構造の情報をもとに、適切に境界領域の交換などを行う。

5.1.2 ステンシル計算関数の定義と実行の概要

AMR 法フレームワーク上でステンシル計算を定義する方法と実行方法について説明する。本フレームワークでは、ステンシル計算は、フレームワークの提供する `MArrayIndex3D` 等を用い、C++11 で 導入されたラムダ式を使い定義する。フレームワークの提供するデータ構造 `MArray` を用いることで、効率的な記述を実現する。`MArray` は、 大きさと位置の情報を持つ `Range3D` 型と配列を保持するクラスであり、この配列を連続メモリ領域として使う。3 次元の拡散計算では、次のようにステンシル計算関数を定義し実行することができる。

```
Engine_t engine;
engine.run(amrcon, inside, LevelGreaterEqual(1),
  [=] __host__ __device__ (const MArrayIndex3D &idx,
      const float *f, float *fn) {
      const float fn = cc*f[idx.ix()]
        + ce*f[idx.ix(1,0,0)] + cw*f[idx.ix(-1,0,0)]
        + cn*f[idx.ix(0,1,0)] + cs*f[idx.ix(0,-1,0)]
        + ct*f[idx.ix(0,0,1)] + cb*f[idx.ix(0,0,-1)];
   }, idx(fa.range()), ptr(&fa), ptr(&fan));
```

`engine` は、第 1 引数に AMR データ構造を表す `Field` などを保持するオブジェクト、第 4 引数にラムダ式で表されたステンシル計算を取り、これに第 5 引数以降を渡す。`fa`、`fan` は `MArray` 型、`inside` は `Range3D` で、`ptr()` はステンシル計算関数中で `MArray` の `inside` の開始点のポインタを取得する。`level()` は計算が適用される格子ブロックの AMR のレベルを取得する。これによりレベル毎に異なる計算を行うことが可能である。`engine` は、ステンシル関数を第 2 引数で渡された `inside` 領域内に対し、第 3 引数の条件を満たすレベルの格子ブロックに対して適用する。`LevelGreaterEqual(1)` を指定することで、このステンシル計算関数は、レベル 1 以上の格子ブロックに適用される。

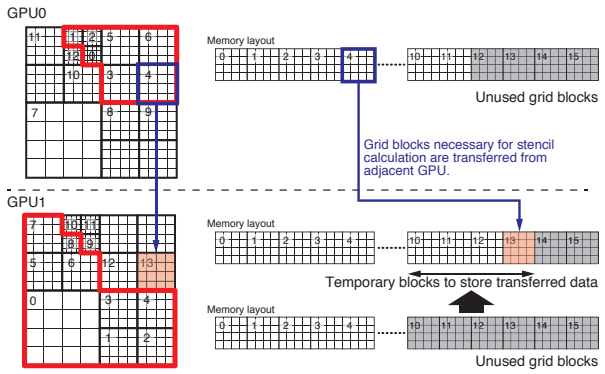

図 1 異なる GPU 間における格子ブロック間の袖領域データ交換

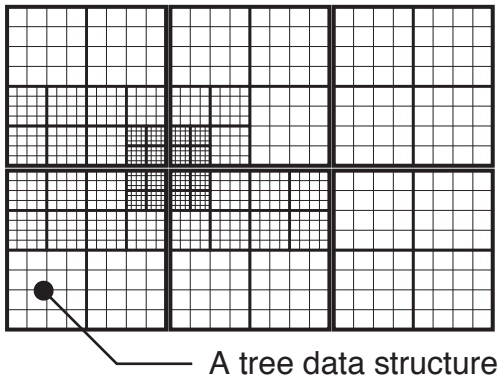

図 3 複数木構造による物理空間の表現

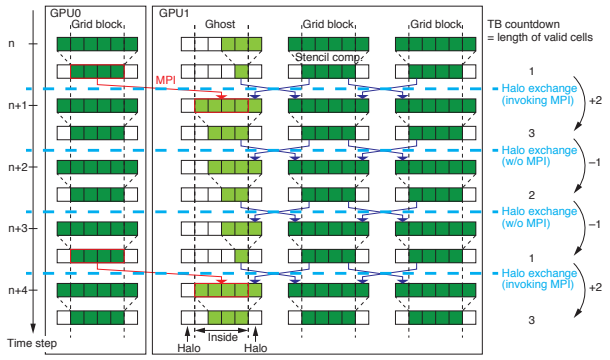

図2 カウントダウン方式を利用した時間ブロッキング法の AMR 法フレームワークへの導入

5.1.3 異なる GPU 間における格子ブロックの袖領域データ交換と時間ブロッキング法の適用

複数 GPU 計算では、時間発展させるためには、同一 GPU 内における格子ブロック間の袖領域データ交換とともに、異なる GPU に割り当てられた格子ブロックの袖領域データも交換する必要がある。図 1 に袖領域データ交換の手順を示す。ステンシル計算に必要な格子ブロックを隣接GPU から転送するとき、まずフレームワークは、各プロセスにおいて、連続メモリ領域から未使用な格子ブロックの一部を一時領域として確保する。次に、MPI 通信を利用して実際に隣接 GPU からステンシル計算に必要となる格子ブロックを転送し、一時領域へ保存する。各プロセスでのステンシル計算は、この一時領域を参照して実行される。

異なる GPU 間の袖領域の交換では、ステンシル計算で必要となる格子ブロックの袖領域だけでなく、格子ブロックの全領域のデータを転送する（図 1 のオレンジ色の領域）。従来の実装では、このデータ転送により大幅な性能低下につながっていたが、ステンシル計算では複数の時間ステップに必要な通信をまとめて実行できる時間ブロッキング法が有効であることがわかっており、図 2 に示すように、直交格子用フレームワークで導入したカウントダウン方式の時間ブロッキング法の実行方法を AMR 法に適用し導入することに成功した。カウントダウン方式の時間ブロッキング手法を用いることで、フレームワークを使う側のユーザコードはほぼ変更を行う必要はなく、通信による性能低下を抑えることに成功した。

5.1.4 GPU 間のデータ移行と動的負荷分散

AMR 法では、時間発展とともに高解像度となる局所的な領域の大きさや位置が変化する。複数 GPU による AMR 法では、性能低下を抑えるために、常にそれぞれの GPU にほぼ同数の格子ブロックが割り当てられるようにして、計算負荷を均等にする必要がある。

フレームワークは、格子ブロックをある GPU から他の GPU へ移行させることを指示する関数を提供する。格子ブロックのデータ移行前に各 MPI プロセスは、全リーフノードの格子ブロックの移行元と移行先を共有する。その後、実際の格子ブロックのデータ移行を並列で実行する。図 3 に示す

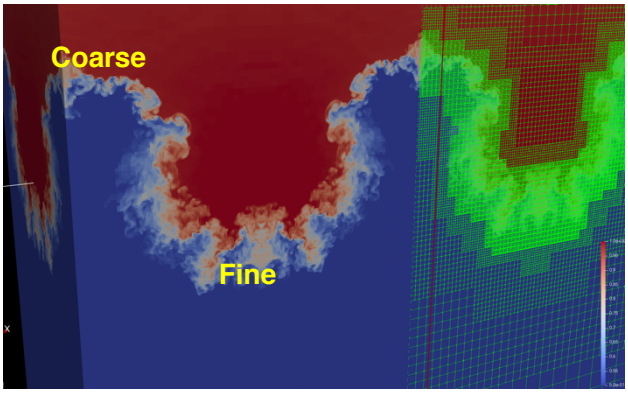

図 4 AMR 法フレームワークを用いた複数 GPU による 3 次元圧縮流体計算によるレイリーテイラー不安定性。格子状の緑色線は各リーフノードが持つ格子ブロックを表す。

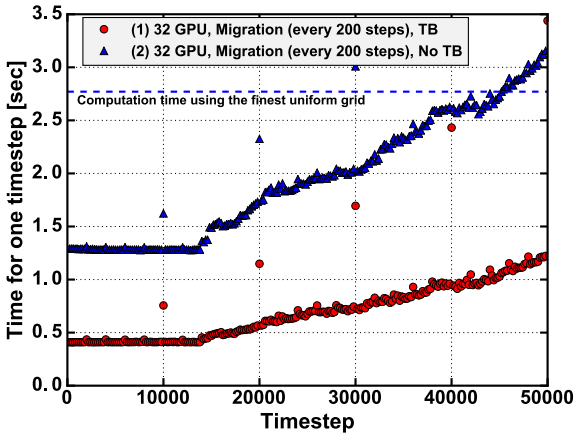

図 5 AMR 法フレームワークを適用した 3 次元圧縮流体計算における各時間ステップの計算時間

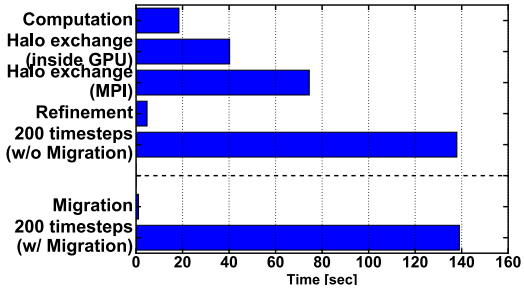

図 6 3 次元圧縮流体計算の計算時間のブレークダウン

ように、本フレームワークでは任意の物理空間を木構造で表現するために、複数の木構造を利用する。格子ブロックは、全ての木構造を順に辿りながら、木構造上の全リーフノードへ割り当てられる。このため、同一の MPI プロセスが管理する格子ブロックは木構造の空間的分布に従い局在化する。フレームワークでは GPU が管理する格子ブロック数に偏りが生じ、ある閾値を超えた場合、全木構造を最初から順に辿り、その上の全リーフノードを各 GPU へ再割り当てする。木構造自体の空間的位置は変わらないため、この再割り当ては効率的に行うことができる。現状の計算では、従来使われるような空間充填曲線は導入する必要なく動的負荷分散に成功している。

5.1.5 3 次元圧縮性流体計算への適用と実行性能

複数 GPU に対応した AMR フレームワークの実行性能を確認するため、3 次元 3 次精度風上手法を用いた圧縮性流体計算へ適用する。中間報告書の時点では、実装の問題で 72GPU までしか実行することができなかったが、これらを改善し、現在では 300GPU 近くまでの計算で問題ないことを確認した。これらの成果をまとめ[2]として発表する。

図 4 に東京工業大学の TSUBAME3.0 スパコンの 8 ノードを用い、各ノードから 4 台、合計 32 台の NVIDIA Tesla P100 GPU を用いた圧縮性流体計算によるレイリーテイラー不安定性の計算結果を示す。2048 × 2048 × 4096 の均一格子と同等の計算領域を 5 レベルの AMR を適用する。1 つの格子ブロックは 20^3 格子で、最大格子の幅は最小格子の幅の 16 倍となる。

図 5 は、この計算が各時間ステップに要した計算時間を示す。時間ブロッキング手法(TB)の有無で測定した。導入した時間ブロッキング法を用いることで、GPU 間の袖領域交換手法の実行性能が向上し、各時間ステップに要する計算時間が短くなっている。同等の計算を均一格子で行う場合、1 ステップで 2.7 秒かかるため、AMR 法により大幅な実行時間の減少が確認された。なお、動的負荷分散のためのデータ移行は、200 ステップに一度実行する。

図 6 に、8GPU 利用時の計算時間および通信にかかった時間の内訳を示す。図に示すように、AMR 法

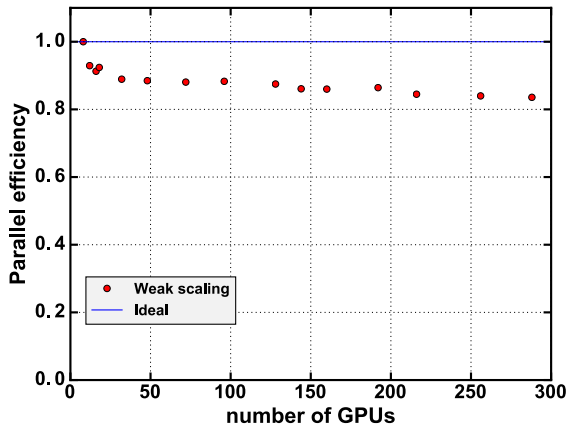

図 7 AMR 法フレームワークを適用した 3 次元圧縮流体計算の弱スケーリング

では計算時間よりも GPU 内および MPI を使った GPU 間の袖領域の通信に大幅に時間がかかっていることがわかった。一方で、動的負荷分散のためのデータ移行（Migration）は、実行回数が少ないこともあり、予想よりも影響は小さく、全体の実行性能にはそれほど影響を与えていない。このことから、実行性能全体の性能向上のためには、動的負荷分散性能の向上よりも各時間ステップにおける通信性能を最適化する必要があることがわかった。

図 7 はこの計算の TSUBAME3.0 での弱スケーリングを示す。GPU 数に合わせて水平方向の空間を拡大する。中間報告書時点では 72GPU までの実行にとどまっていたが、現在、288GPU を利用した計算が可能であることを確認した。288GPU の結果は 8GPU に対して 84%の並列化効率を達成した。

5.1.6 格子ボルツマン法への AMR 法の適用

フレームワークの適用を目指した流体中を流れ成長する金属凝固成長計算の流体計算は、格子ボルツマン法を用いる。この格子ボルツマン法は時間更新幅が格子幅の定数倍に固定される計算手法であるために、解像度ごとに時間ステップ幅が異なり、時間方向にも物理量の補間が必要となる。図 8 に AMR を適用した格子ボルツマン法で必要となる解像度間と時間ステップ間の補間関係を示す。本フレームワークでは、これまで、ある時刻では

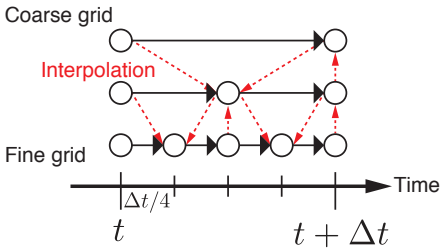

図 8 格子ボルツマン法における異なる解像度間での補間関係

全ての解像度でデータを保持していることを前提としていたため、この補間計算の実装に時間を要した。最終的にフレームワークを時間方向の物理量の補間計算に対応させ、AMR 法を適用した格子ボルツマン法の計算を実現した。

6. 今年度の進捗状況と今後の展望

本研究計画では、GPU スパコンに適した動的負荷分散手法、通信削減技術、時間発展の最適化などを開発し、フレームワークを高度化する。これを流体中を流れながら成長する金属凝固計算に適用することを目指す。当初の計画通り、通信削減技術および動的負荷分散技術を開発し、導入することでフレームワークの高度化を進めた。今年度の前半は、TSUBAME3.0 の 72 GPU を用い、88%の並列化効率を達成した。今年度の後半では、実装をさらに高度化することで、さらなる大規模化を行い TSUBAME3.0 の 288 GPU を用いた計算に成功し、84%の並列化効率を達成することに成功した。これを[2]としてまとめ発表する。

今年度は、フレームワークを流体中を流れ成長する金属凝固成長計算へ適用を目指し進めた。この予備検討として、これまでに開発した AMR を適用した圧縮性流体計算（差分法）と金属凝固成長計算であるフェーズフィールド法とのカップリングを進めたが、物理的に安定して計算することが難しいことがわかった。このため流体計算には格子ボルツマン法を用いる。この格子ボルツマン法は時間更新幅が格子幅の定数倍に固定される計算手法であるために、解像度ごとに時間ステップ幅が異なり、時間方向にも物理量の補間が必要とな

る。本フレームワークは、時間方向の物理量の補間計算に未対応であったため、今年度前半の途中から、この実装の開発を進めた。最終的に、フレームワークを時間方向の物理量の補間計算に対応させ、AMR 法を適用した格子ボルツマン法の計算を実現した。この実装に時間を要すると考えて当初の計画よりも前倒しに開始したが、計画よりも多くの時間を要した。このために AMR 法を適用したフェーズフィールド法とのカップリングを進めたものの、完了には至らなかった。当初、機械学習を用いたさらなる動的負荷分散の向上を計画していたが、動的負荷分散の最適化よりも各時間ステップ中の通信コストを削減する効果が大きいことが次第にわかってきたため、通信コストの削減を中心に検討を進めた。

AMR 法を適用した格子ボルツマン法は、像度ごとに時間ステップ幅が異なるため、開発済みの差分法用の通信をまとめる技術はそのまま適用できない。また、動的負荷分散が実現し、大規模計算を進めるにつれ、各時間ステップの計算で必要となる袖領域更新のための GPU 間の通信が性能低下の大きな原因であることがわかってきた。さらに、動的負荷分散のための領域分割による複雑な幾何形状は、通信量を増加させるとともに、通信のための一時メモリの使用量を増加させ、大規模化の大きな妨げとなり課題である。格子ボルツマン法とフェーズフィールド法とのカップリングを実現し、さらなる大規模化のためには、これらを解決することが必要不可欠である。

そこで、今後は、AMR 法を適用した格子ボルツマン法の大規模・高性能計算の実現に焦点をあてながら、複数 GPU 向けの AMR 法フレームワークの高度化を目指す。最終的には、これを AMR 法を適用したフェーズフィールド法とカップリングさせ、流体中を流れ成長する金属凝固成長を完成させる。最終的に、この大規模計算を実現することを目指す。

7. 研究成果リスト

(1) 学術論文

なし

(2) 国際会議プロシーディングス

[1] N. Onodera, Y. Idomura, Y. Ali and *T. Shimokawabe*, ″Communication Reduced Multi-time-step Algorithm for Real-time Wind Simulation on GPU-based Supercomputers,″ the 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA) 2018, Dallas, US, pp. 1-8, 2018.

[2] *Takashi Shimokawabe* and Naoyuki Onodera, ″A High-productivity Framework for Adap- tive Mesh Refinement on Multiple GPUs,″ International Conference on Computational Science (ICCS) 2019, Faro, Algarve, Portugal, Jun. 2019. (to appear)

(3) 国際会議発表

[3] *Takashi Shimokawabe*, ″AMR Framework with multiple GPUs to Realize Effective High-Resolution Simulations,″ GPU Technology Conference (GTC) 2019, San Jose, CA, USA, March 23 - 26, 2019. (ポスター)

(4) 国内会議発表

[4] *下川辺隆史*、小野寺直幸, “複数 GPU を用いた高精細計算を実現する AMR 法フレームワークの開発”, 日本計算工学会 第 23 回計算工学講演会,名古屋, 2018 年 6 月.

(5) その他（特許，プレス発表，著書等）

[5] 第 23 回計算工学講演会 グラフィックスアワード特別賞 (MSC Apex 賞) 受賞, 2018 年 6 月.

[6] 第 23 回計算工学講演会 グラフィックスアワード特別賞 (Visual Computing 賞) 受賞, 2018 年 6 月.