jh190064-NAH

AMR 法を適用した LBM 計算の大規模化に向けたフレームワークの拡張

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（AMR）法が有効である。本研究では、様々な解像度のステンシル計算を高性能に実行する手法や GPU 間の計算負荷を均等とする動的負荷分散を実現する GPU スパコン用の AMR 法フレームワークを構築する。本年度は、近年、大規模な直交格子上での計算で多く用いられている格子ボルツマン法(LBM)に AMR 法を適用する。AMR 法を適用した LBM は、解像度ごとに時間ステップ幅が異なり、空間方向に加えて時間方向にも物理量の補間が必要となる。これをフレームワークで実現し、AMR 法を適用した LBM 計算を実現した。また、大規模計算における GPU 間通信の削減のため通信を効率化する領域分割方法を導入し、その有用性を検証した。

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

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。平成 27, 28 年度課題で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、AMR フレームワークを完成させ(平成 28、29 年度課題)、平成 30 年度課題では 100 台を超える複数 GPU 計算を達成した。

本研究では、開発中の AMR 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の適用を進めている。金属凝固成長計算は、格子ボルツマン法 (Lattice Boltzmann Method; LBM)による流体計算とフェーズフィールド法による凝固成長計算で構成される。

本年度は、AMR 法を適用した LBM 計算の大規模・高性能計算に焦点をあてながら、これまで以上に大規模な計算に向け、AMR 法フレームワークの高度化を行う。まず解像度により異なる時間ステップ幅をもつ LBM 計算

へ AMR 法を適用する。次に、LBM 計算のための通信をまとめて行う手法を開発する。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になってきている。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。さらなる大規模化のため、これらを解決することを目指す。

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、同一および異なる時間ステップ幅に対応し、スパコンに必須な通信隠蔽・削減技術と併用した、高性能 AMR 法を達成する試みは他に例はなく、世界での注目は間違いない。

本研究が対象とする LBM は、完全な陽解法に基づくアルゴリズムであり、近年、大規模な直交格子上での計算で多く用いられる。しかしながら、解像度ごとに時間ステップ幅が異なるアルゴリズムであるために、時間ステップ幅を固定したまま、単純に局所的に解像度を粗くすることができない。このため、LBM への AMR 法の導入例はあるものの、これに対応したフレームワークは少なく、流体中を流れる金属凝固成長計算のように LBM および差分計算という全く異なる二つの手法へ適用できるフレームワークの開発は他にない。

本研究は、従来の差分計算だけでなく、近年、多く用いられるようになった LBM 計算にも対応したフレームワークを構築するものである。様々なアプリケーションに適用できる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、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-L 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。

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。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。

前年度（平成 30 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 で発表した [1]。

5. 今年度の研究成果の詳細

本年度は、AMR 法を適用した格子ボルツマン法をフレームワークで実現するために、AMR 法フレームワークの高度化を進めた。格子ボルツマン法（LBM）は、これまでに本フレームワークが主に対象としてきた圧縮性流体計算とは異なる機能を要求する。まず、LBM 計算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る。異なる時間ステップ幅をもつために、時間方向への物理量の補間が必須である。また、LBM は通常より多数の変数を扱うため、これを効率的に扱える必要がある。本研究では、これらをフレームワークの機能として提供するために、フレームワークの提供するデータ型の拡張、それに対応するためのステンシル計算関数の拡張、格子ブロックの境界領域の交換を行う関数の開発を行った。これらを用いて、AMR 法を適用した LBM 計算に成功した。

また、本フレームワークでは、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になってきていることがわかってきている。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。これを解決するために、通信を効率化する領域分割方法を導入した。

以下では具体的な研究成果について説明する。

5.1. AMR 法フレームワークの概要

前年度までに構築した AMR 法フレームワークを基盤として、これを格子ボルツマン法の計算へ対応させる。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までで開発したものである。本年度は、これを基盤として、格子ボルツマン法計算へ対応するための拡張を行った。まず、格子ボルツマン法の拡張に関連した部分の AMR 法フレームワークの概要について説明する。

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。図 1 に多数の格子の物理空間配置とメモリ空間の配置の模式図を示す。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。

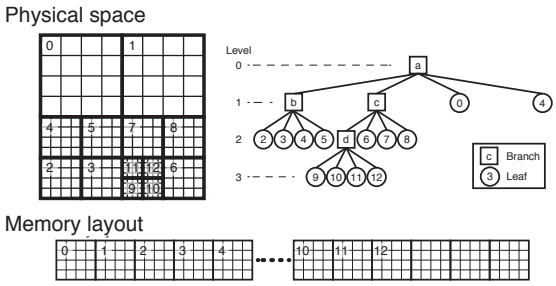

図 1　多数の格子の木構造による物理空間配置とメモリ空間配置

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` 型と多数の配列を保持するクラスであり、この配列を連続的に確保し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。

5.2. 複数変数保持のための MArray の拡張

次に、格子ボルツマン法に AMR 法を適用するために本年度行ったフレームワークの拡張について説明する。中間報告書の段階では、同等の機能のプロトタイプを実装していたが、これを整理し実用性のあるフレームワークの機能として提供するように高度化を進めた。

MArray 型の拡張について説明する。AMR 法では解像度の異なる多数の格子を扱う。前年度までの研究で、AMR 法フレームワークは独自の MArray を導入することで効率的に記述できることがわかった。

本年度に対象とした格子ボルツマン法は、流体を有限個の速度を持つ多数の仮想粒子の集合として表す。2 次元計算では 9 方向の速度で記述する D2Q9 モデルや、3 次元計算では 19 方向の速度で記述する D3Q19 モデルなどが用いられる。本研究では、D2Q9 モデルを対象とした。これらの速度モデルの他に、格子ボルツマン法では計算中に巨視的な流れ場の密度や速度を表す変数を別途用意する必要がある。

これまでに本フレームワークが対象とした圧縮性流体計算では、高々数個の変数しか用いていなかったが、格子ボルツマン法では多数の変数を用いる必要がある。AMR 法では、これら全ての変数に対して解像度の異なる時間発展計算や格子解像度の変更など整合性を保ちながら行わなければならない。そこで、本フレームワークではこれらの変数を簡便に扱えるようにするために、次のように単一の MArray 型オブジェクトで複数の物理変数を保持できるように拡張を行う。

```
// 直交格子のデータの大きさ、位置
unsigned int length[] = {nx+2*mgnx,ny+2*mgny,nz+2*mgnz};
int          begin [] = {-mgnx, -mgny, -mgnz};
Range3D whole(length, begin); // 大きさ、位置を表す

// ホスト上、デバイス上に narrays 個の直交格子用データを作成する
MArray<float, Range3D> fm_h(whole, nvar, narrays,
                            MemoryType::HOST_MEMORY);
MArray<float, Range3D> fm_d(whole, nvar, narrays
                            MemoryType::DEVICE_MEMORY);
```

narrays は 1 物理変数が利用する配列データの個数を表す。nvar が新たに導入されたパラメータで、1 変数内で確保する内部変数の個数を表す。D2Q9 モデルでは 9 を設定することとなる。GPU で性能が出やすくするため nvar × narrays 個の配列データは連続アドレスとして確保される。

格子解像度の変更や格子ブロックどうしの境界領域の交換のためデータ転送では 1 変数内で内部変数が複数個確保されても正しく処理できるように拡張されている。時間発展計算となるステンシル計算関数では、内部変数間のアドレスのオフセットを取得できるようになっており、これを用いて関数を記述することが可能となっている。

5.3. 格子ボルツマン法への AMR 法の適用のための時間発展計算の拡張と時間方向の補間

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。本フレームワークでは、これまでは、ある時刻では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。図 2 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。図は 3 解像度の場合の例である。

本研究では、まず特定の条件を満たすレベルの格子ブロックに対してステンシル計算関数や境界領域の交換を適用できるよう、`LevelGreaterEqual` に加えて適用範囲のレベルを指定できる `LevelRange` を導入し、ステンシル計算関数および境界領域の交換において対応した。例えば、`LevelRange(1, 3)`を指定することで、レベル 1 から 3 の格子ブロックにのみステンシル計算関数や境界領域の交換など特定の処理を適用させることができる。特定のレベルのみを実行するのではなく、レベルの範囲を指定して実行できるようにしたことで、GPU カーネルの実行回数を最小回数として、実行性能の向上に寄与している。

次に、時間方向にも物理量の補間が必要となるため、これに対応した格子ブロックの境界領域の交換手法を導入した。これまでは、単一の `MArray` 型の変数内だけで格子ブロックの境界領域の交換を行うとしていたが、これを拡張して、2 つの `MArray` 型の変数を指定可能とし、かつ `MArray` 型の内部にもつ複数の内部変数に対応できるように拡張した。また、時間方向の物理量の補間の際には解像度の変更、すなわち空間方向の補間も必要となるため（図 2 参照）、格子ボルツマン計算へも対応できるよう任意の補間関数を指定できるように拡張した。

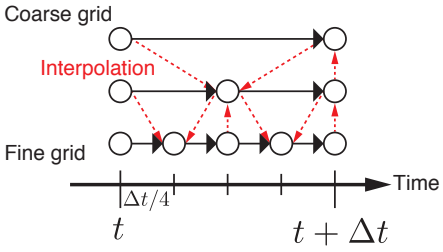

図 2　格子ボルツマン法における異なる解像度間での補間関係

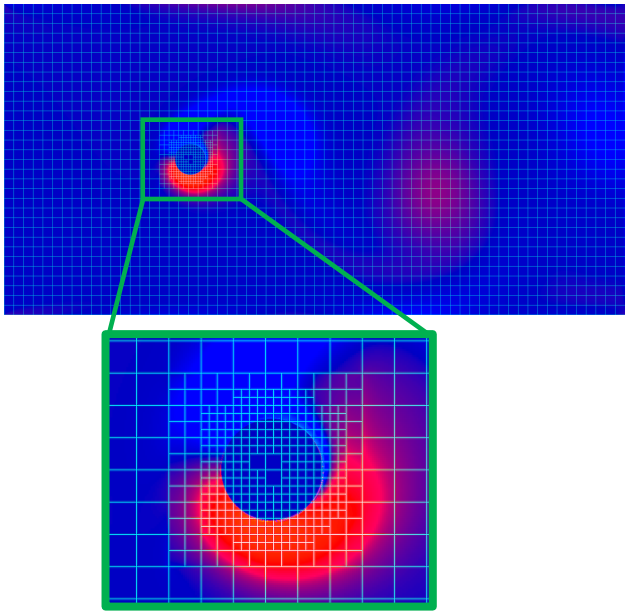

図 3 AMR フレームワークを適用した格子ボルツマン計算

5.4. 格子ボルツマン法への AMR 法の適用例

図 3 に AMR フレームワークを適用した LBM による円柱まわりの流れの計算を示す。円柱（物体）のまわりの格子が細分化されていることがわかる。

5.5. 通信を効率化する領域分割法の高度化

次に、本年度実施した領域分割方法の高度化について説明する。

本フレームワークでは、複数の木構造で物理空間を表現する。現状のフレームワークは、各 GPU のメモリ使用量を均等として負荷を均一に分散することを最優先しているため、領域分割は木構造単位と関係なく各 GPU の担う計算量が均等となるように分割される。このため各 GPU の担う計算領域の幾何学的形状が

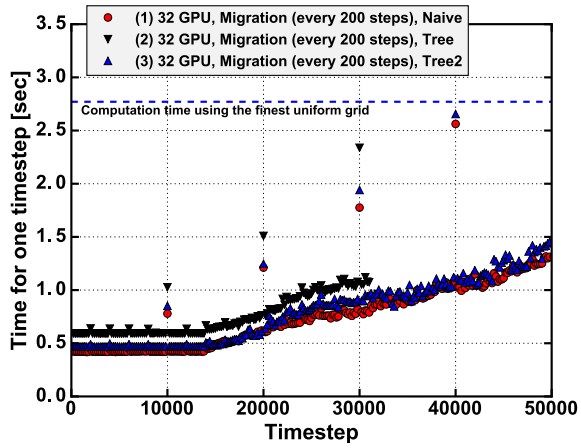

図 4　AMR 法フレームワークを適用した 3 次元圧縮流体計算における各時間ステップの計算時間。3 つの領域分割方法を導入している。

複雑となる。これにより各領域の表面積が増大するために、通信パターンの複雑化とともに通信量が増加し、性能低下の要因となる。そこで、均等に配置された木構造を単位とした領域分割方法を導入した。境界領域を整えるため、各プロセスへ計算領域を木構造単位で割り当てる Tree 方式と各プロセスへ 1/8 木構造単位で割り当てる Tree2 方式を導入した。これらの方式では、各 GPU のメモリ使用量を若干の偏りは許すものの、なるべく均等にし、境界が整うことを優先している。

図 4 に Tree, Tree2 および従来の各 GPU のメモリ使用量を完全に均等とする Naive 方式を利用した圧縮性流体計算の各時間ステップの計算時間を示す。計算には東京工業大学の TSUBAME3.0 の 32 GPU を用いた。1 ノードあたり 4 GPU 利用している。Tree2 は幾何学形状が単純となり通信パターンが単純化されたことで、一部の時間ステップにおいて Naive の性能を超えている。一方、Tree, Tree2 は境界領域を整えることを優先したため、各 GPU に割り当てられた負荷に偏りが生じる。このため、Naive よりも性能が悪い時間ステップがある。今後は、分割方法も工夫することで、さらなる性能向上を行う予定である。

6. 今年度の進捗状況と今後の展望

本研究計画は、AMR 法を適用した LBM 計算の大規模・高性能計算に焦点をあてながら、これまで以上に大規模な計算に向け、AMR 法フレームワークを LBM 計算へ対応させ、大規模化に向けた通信などの高度化を行う。

本年度は、当初の計画通り、AMR 法を適用した格子ボルツマン法をフレームワークで実現するために、AMR 法フレームワークの高度化を進めた。LBM 計算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。これに対応するために、異なる時間ステップ幅で効率的にステンシル関数を実行する方法、時間方向への物理量の補間を行う機能などを導入した。これらを用いて、AMR 法を適用した LBM 計算に成功した。また、当初の計画通り、大規模計算時の性能を向上させるためにフレームワークの高度化を行った。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っていたため、通信を効率化する領域分割方法を導入した。

当初の計画では、LBM 計算のための通信をまとめて行う手法の開発と通信を効率化する領域分割方法を LBM 計算へ適用することを目標としていた。本課題では、AMR 法の導入が効果的である流体中を移動する物体を扱う LBM 計算コードに対しても AMR 法の適用を進めた。しかしながら、その開発および検証に時間を要したため、直交格子での計算には成功したものの、そのコードに関しては AMR 法の適用まで至らず、新しい領域分割方法の効果を確認するに至らなかった。また、通信を効率化する領域分割方法を導入したものの、計算負荷の分散や通信の最適化に高度化の余地があり、これは今後の課題である。

7. 研究業績一覧（発表予定も含む）

(1) 学術論文 （査読あり）

なし

(2) 国際会議プロシーディングス （査読あり）

[1] *Takashi Shimokawabe* and Naoyuki Onodera, "A High-productivity Framework for Adaptive Mesh Refinement on Multiple GPUs," International Conference on Computational Science (ICCS) 2019, Faro, Algarve, Portugal, Jun. 2019.

(3) 国際会議発表 （査読なし）

[2] Takashi Shimokawabe and Naoyuki Onodera, “AMR Framework to Realize Effective High-Resolution Simulations on Multiple GPUs,” International Conference on High Performance Computing in Asia-Pacific Region (HPCAsia) 2020, Fukuoka, Japan, January, 2020.（ポスター発表）

[3] Takashi Shimokawabe and Naoyuki Onodera, "AMR Framework for Large-scale Simulations on Multiple GPUs," SIAM Conference on Parallel Processing for Scientific Computing 2020, Seattle, Washington, U.S., Feb. 2020.

(4) 国内会議発表 （査読なし）

[4] 下川辺隆史、小野寺直幸, “AMR 法フレームワークの大規模 GPU 計算に向けた発展”, 日本計算工学会 第 24 回計算工学講演会, 大宮, 2019 年 5 月.

(5) その他（特許, プレスリリース, 著書等）

なし