学際大規模情報基盤共同利用・共同研究拠点 公募型共同研究 平成30年度採択課題
10th Symposium
Joint Usage / Research Center for Interdisciplinary Large-scale Information Infrastructures
jh180061-NAH

下川辺 隆史 (東京大学)

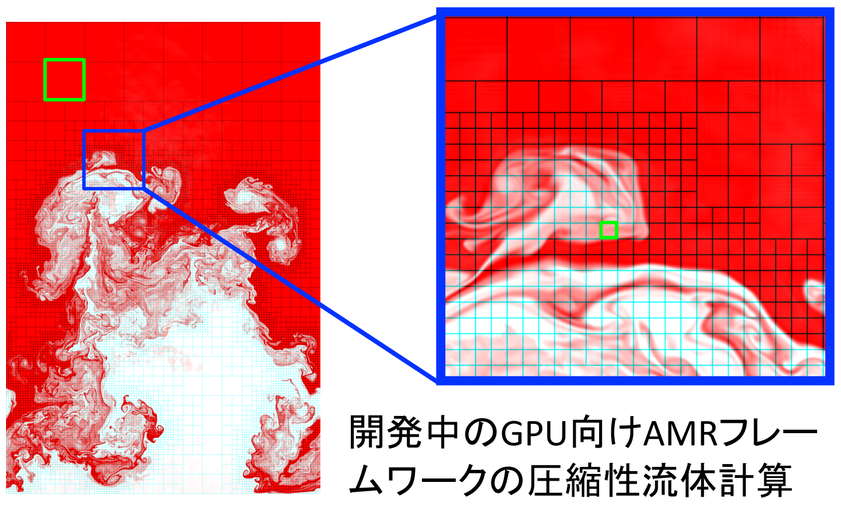
高精細計算を実現するAMR法フレームワークの高度化

1 研究背景と研究目的

近年、ステンシル計算を用いた格子に基づいたシミュレーションでは、大規模な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法（Adaptive mesh refinement; AMR法）が有効である。

本研究では、開発中のGPU向けの高生産・高性能AMRフレームワークを高度化する。前年度までに、複数GPUに対応したAMRフレームワークを構築した。しかしながら、計算負荷の分散や通信の最適化に高度化の余地がある。そこで本年度は、GPUスパコン上で実行時間を最小化するためのフレームワークの高度化を進める。最終的に、高度化したフレームワークで東京大学のReedbush-Hおよび東京工業大学のTSUBAME3.0のGPUスパコン上で局所的に高精細にできるAMRアプリケーションの開発技術の確立を目指す。

本年度は、開発したフレームワークを現在取り組んでいる流体中を流れながら成長する金属凝固計算に適用し、その高精細計算を実現することを目指す。



開発中のGPU向けAMRフレームワークの圧縮性流体計算への適用例。緑のブロックは同一数の格子点を持つ。

2 AMR法フレームワーク

AMR法フレームワークの概要を述べる。平成26、27年度課題で開発したGPU/CPUで高性能を実現するステンシル計算フレームワークを基盤とする。前年度までに、複数GPUに対応したAMRフレームワークを構築した。

2.1 フレームワークの対象

- ✓ 直交格子をベースとしたブロック型のAMR
- ✓ 各格子点上で定義される物理変数（配列）の時間変化を計算
- ✓ 物理変数の時間ステップ更新は陽的でステンシル計算で行う

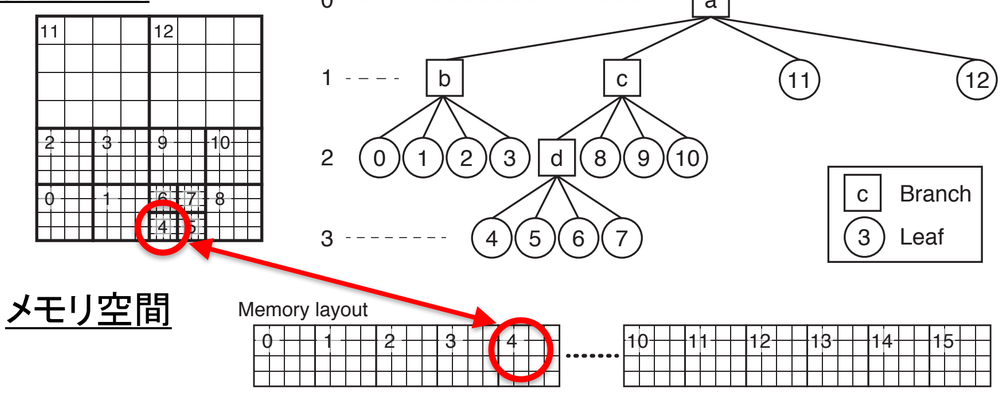
2.2 フレームワークの設計

- ✓ フレームワークはC++/CUDA/MPIで構築、複数GPU計算対応
- ✓ ユーザは基本的にステンシル計算についてのみ記述
- ✓ AMRでは様々な解像度のブロックが存在するが、ユーザはあたかも単一解像度への計算として記述できる
  → これを実現するため各ブロックは袖領域の格子を持つ
- ✓ AMRデータは木構造で管理するが、これを意識しないプログラミング
- ✓ 任意の数の物理変数（配列）を扱える

2.3 AMR法のデータ構造

- ✓ 構造格子を再帰的に細分化し、木構造で表す
- ✓ 物理空間ではリーフノードに格子ブロックを配置
- ✓ メモリ空間では複数の格子ブロックを単一配列内に配置
- ✓ リーフノードとメモリ上の格子ブロックは整数値(Id)で対応付け

物理空間



メモリ空間

2.4 ステンシル関数の記述と実行

- ✓ メモリレイアウトをフラットな構造とすることで、c++11ラムダ式で定義された直交格子用のステンシル計算関数を、一度に全格子ブロックに適用できる

```
// User-written stencil function
auto diffusion3d = [=] __host__ __device__
(const MArrayIndex &idx, int level,
 float ce, float cw, float cn, float cs, float ct,
 float cb, float cc, const FLOAT *f, float *fn) {
    fn[idx.ix()] = + 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)];
 }};
```

(i, j, k)

(i, j, k-1)

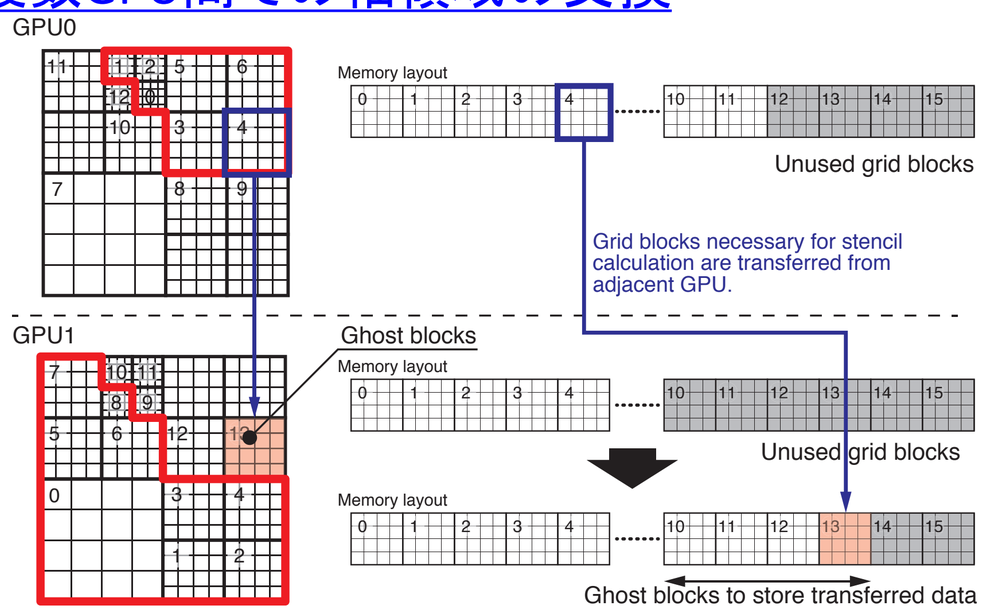
- ✓ フラットなメモリレイアウトにより複数の格子ブロックを同時に計算できる

```
Range3D inside; // 3D rectangular range where stencil functions are applied.
Engine_t engine;
engine.run(amrcon, inside, LevelGreaterEqual(1), diffusion3d,
           idx(f.range()), level(), ce,cw,cn,cs,ct,cb,cc, ptr(f), ptr(fn));
```

3 複数GPU間での袖領域の交換と効率化

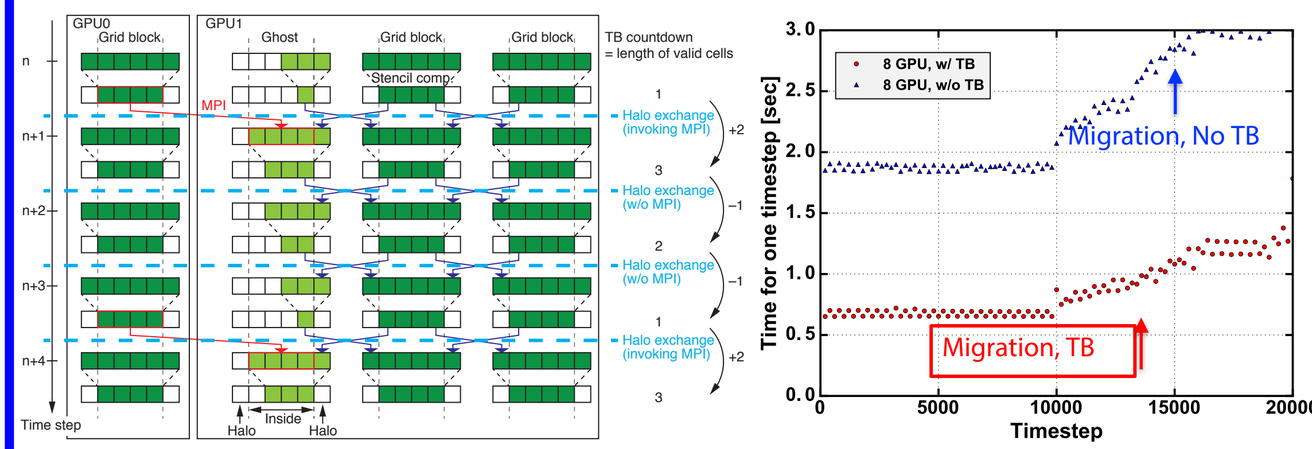
複数GPU計算では、GPU間の格子ブロックの袖領域の交換による性能低下が大きい。時間ブロッキング手法を適用し性能向上を実現する。

3.1 複数GPU間での袖領域の交換



- ✓ ステンシル計算を行うため、隣接GPUから隣接する格子ブロックを転送する
- ✓ 未使用の格子ブロックから、隣接GPUから転送された格子ブロックを保持する「ゴースト格子ブロック」を確保し、そこへ転送する
- ✓ 袖領域の交換では、格子ブロックに含まれる全データが転送される

3.2 時間ブロッキング手法による通信の効率化



- ✓ 格子ブロックに含まれる全データを転送することによる性能低下を抑えるため、複数タイムステップの通信をまとめて行い、通信なしで複数のタイムステップを進められる時間ブロッキング手法を導入する
- ✓ 時間ブロッキング手法を導入したことで計算時間が約36%となっている

4 まとめと今後の研究計画

前年度までに構築を進めた複数GPU向けAMR法フレームワークの高度化を進める。特に計算負荷の分散や通信の最適化により、GPUスパコン上で実行時間が最小化となるよう最適化を進める。また、流体中を流れ成長する金属凝固計算へAMR法フレームワークを適用し、高解像度計算を実現する。最終的に、高度化したフレームワークでGPUスパコン上で局所的に高精細にできるAMRアプリケーションの開発技術の確立を目指す。