jh160051-NAH

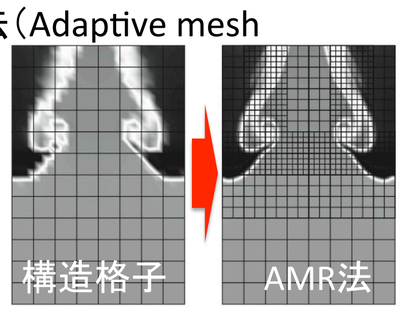
下川辺隆史　（東京工業大学）

高精細計算を実現するAMR法フレームワークの構築

1 研究背景と研究目的

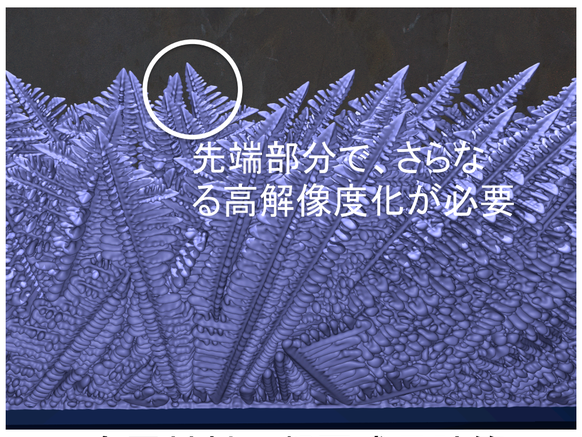
近年、ステンシル計算を用いた格子に基づいたシミュレーションでは、大規模なGPU計算が可能となり、広大な計算領域の場所によって求められる精度が異なる問題に有効な手法が要求されてきている。GPU計算では、GPUが得意なステンシル計算を活用しながら、高精度が必要な領域を局所的に高精細にできる適合細分化格子法（Adaptive mesh refinement; AMR法）が有効である。

本研究では、GPUスパコン用のAMR法を確立する。GPU上での様々な解像度の格子を効率的に管理する技術、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、局所的に高解像度となってもGPU間の計算負荷を均等とする動的負荷分散を実現する。



適合細分化格子法の適用

GPUスパコン上でAMR法をただ実現するだけでなく、これを汎用的に利用できるフレームワークとする。平成26年度、27年度にGPUおよび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開発した。本年度は、これを拡張しAMR法に対応させることでAMR法フレームワークを構築する。これを用いて金属材料の凝固成長計算などへ適用する。



金属材料の凝固成長計算

2 ステンシル計算フレームワーク

本年度に構築するAMR法フレームワークは、前年度までに構築したステンシル計算フレームワークを基にする。AMR法では様々な解像度の構造格子で多数のステンシル計算を行う。構造格子の管理、ステンシル計算の記述には、構築したステンシル計算フレームワークを用いる。

2.1 フレームワークの構造

- ✓ 複数GPU計算に対応。
- ✓ C++およびCUDAによる実装。C++から利用できる。
- ✓ フラットMPIによる並列化
- ✓ ステンシル計算はGPUあるいはOpenMP並列によるCPUで実行
- ✓ GPU計算では、自動チューニング、通信の隠蔽手法を導入

2.2 直交格子データ構造

- ✓ データの大きさと位置を保持した構造
- ✓ 効率的な記述のために、このデータ構造 ETArray を利用する

```
// 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、デバイス上に直交格子用データを作成する
ETArray<float, Range3D> f_h(whole, MemoryType::HOST_MEMORY);
ETArray<float, Range3D> f_d(whole, MemoryType::DEVICE_MEMORY);
```

2.3 ステンシル計算の記述

- ✓ ある格子点を更新する関数（C++ファンクタ）をユーザが定義
- ✓ ArrayIndexによりインデックスを表現

```
struct Diffusion3d { // ユーザ定義のステンシル関数（例は拡散方程式）
__host__ __device__
  float operator()(const float *f, const ArrayIndex &idx,
    float ce, float cw, float cn, float cs, float ct,
    float cb, float cc) {
    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)];
    return fn; // 戻り値は、ある一点の更新する値
  }
};
```

2.4 ステンシル計算の実行

- ✓ フレームワークを用いユーザ定義関数を全格子点へ適用
- ✓ GPUでは自動チューニングし実行することができる

```
view(fn,inside) = funcf<float>(Diffusion3d(), ptr(f), idx(f),
                                              ce, cw, cn, cs, ct, cb, cc);
// f,fnはETArray, insideはRange3Dで関数適用範囲の指定
// view により fn のinside領域内の格子点のみ値を更新
// ptrはETArrayのrawポインタ、idxはインデックス表現 ArrayIndex を渡す
```

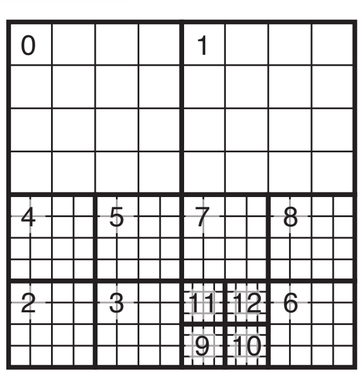
3 AMR法フレームワークへの拡張

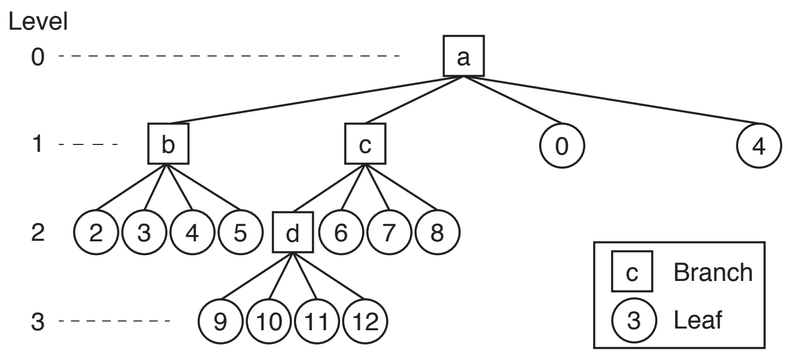
ステンシル計算フレームワークを最大限に活用し、高性能を実現するため、AMR法フレームワークを以下のように設計する。

3.1 適合格子細分化法（AMR法）の構造

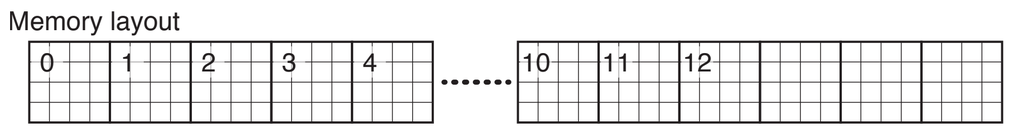
- ✓ 構造格子を再帰的に細分化し、木構造で表す
- ✓ 物理空間ではリーフノードに構造格子を配置
- ✓ メモリ空間では構造格子をフラットに配置
- ✓ 各構造格子はETArrayと同等のデータ型を用いることで、ステンシル計算フレームワークで開発した機能が利用可能

物理空間



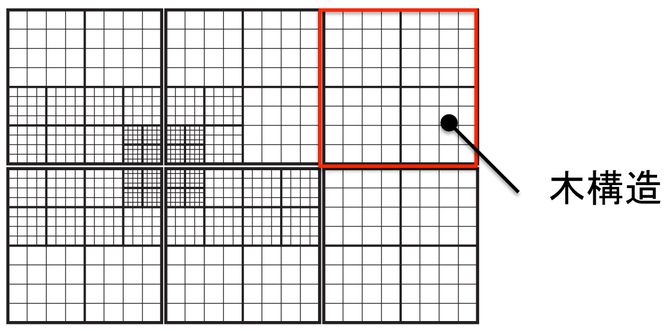


メモリ空間



3.2 任意の領域形状への対応

- ✓ 複数の木構造を物理空間に配置することで任意の領域形状に対応



3.3 フレームワークとしての汎用化

- ✓ 木構造を管理するクラスと構造格子の実体を管理するクラスを別とすることで任意の数の変数を扱う
- ✓ ステンシル計算関数をそのまま利用できるようにするため、リーフノード上の構造格子は袖領域を持つ
- ✓ 構造格子の実体を管理するクラスに対してステンシル関数および式テンプレートを用いた全要素計算を適用できる

4 まとめと今後の研究計画

前年度までに構築したステンシル関数を基に、AMR法の実装およびAMR法フレームワークの構築を進めている。特に、木構造の採用、リーフノードにある構造格子の管理方法、任意の領域形状への対応などを行った。また、ステンシル計算フレームワークの高生産性を高めているステンシル計算関数や式テンプレートを用いた全要素計算を利用できる設計とした。

完成させたAMR法フレームワークを金属材料の凝固成長計算へ適用し、最終的に、GPUスパコンおよびCPU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。