FDPSのAPIと内部構造

岩澤全規
理化学研究所　計算科学研究センター
粒子系シミュレータ研究チーム
フラッグシップ2020プロジェクト コデザイン推進チーム
2019年8月6日　FDPS講習会

構成

- FDPSの実装方針
- FDPSを用いた粒子シミュレーションの流れ。
  - 領域分割
  - 粒子交換
  - 相互作用計算
    - ツリー構造
    - 相互作用に必要な粒子交換
    - Barnesベクトル化
- まとめ

FDPSの実装方針

- 内部実装の言語としてC++を選択
  - 高い自由度と高い性能を両立させるためにFDPSはC++のテンプレートライブラリになっている。
- 並列化
  - 分散メモリー環境(ノード間):MPI
  - 共有メモリー環境(ノード内):OpenMP
    - FDPSが提供するAPIは並列化されており、ユーザーは並列化を意識してコードを書く必要がない。

FDPSを用いた粒子シミュレーションの流れ

1. 計算領域全体を分割する。
2. 計算領域に合わせて粒子を再配置する。
3. 各プロセスが担当する粒子への相互作用を計算する。
4. 相互作用の結果を使って粒子の情報を更新する。

- FDPSは手順1,2,3を担当。
- 手順1,2,3に対応したクラスがある。
  - DomainInfoクラス: 領域のデータを持ち、領域分割を行う。
  - ParticleSystemクラス: 粒子のデータを持ち、粒子交換を行う。
  - TreeForForceクラス: 相互作用の計算を行う。
  - ユーザーはこれらのクラスにテンプレート引数として粒子クラスを与え実体を作り、メンバ関数を呼び出すことでそれぞれの処理を行う。

FDPSを用いた粒子シミュレーションの流れ

1. 計算領域全体を分割する。
2. 計算領域に合わせて粒子を再配置する。
3. 各プロセスが担当する粒子への相互作用を計算する。
4. 相互作用の結果を使って粒子の情報を更新する。

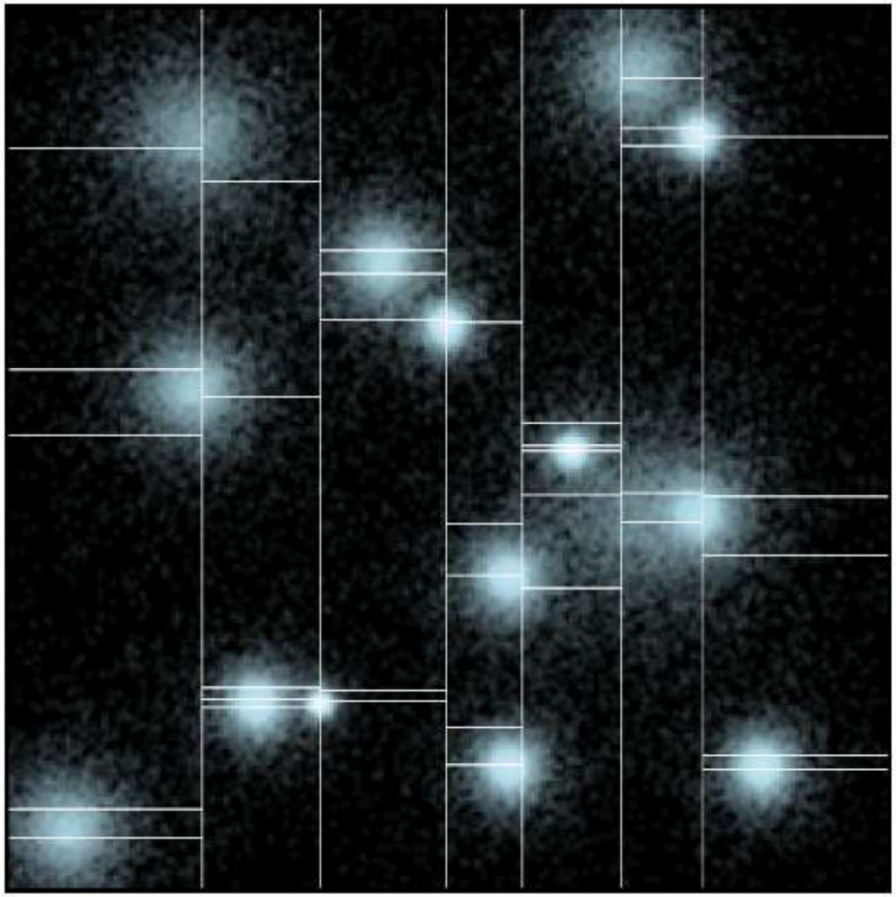
- Ver4.0以降では、木構造や相互作用リストを再利用する機能を実装
  - 再利用中は手順1、2と3の一部を計算する必要がない。
  - SPH法やMD計算等で高速に計算が可能。

FDPSを用いた粒子シミュレーションの流れ

1. 計算領域全体を分割する。
2. 計算領域に合わせて粒子を再配置する。
3. 各プロセスが担当する粒子への相互作用を計算する。
4. 相互作用の結果を使って粒子の情報を更新する。

- FDPSは手順1,2,3を担当。
- 手順1,2,3に対応したクラスがある。
  - DomainInfoクラス: 領域のデータを持ち、領域分割を行う。
  - ParticleSystemクラス: 粒子のデータを持ち、粒子交換を行う。
  - TreeForForceクラス: 相互作用の計算を行う。
  - ユーザーはこれらのクラスにテンプレート引数として粒子クラスを与え実体を作り、メンバ関数を呼び出すことでそれぞれの処理を行う。

領域分割と粒子交換



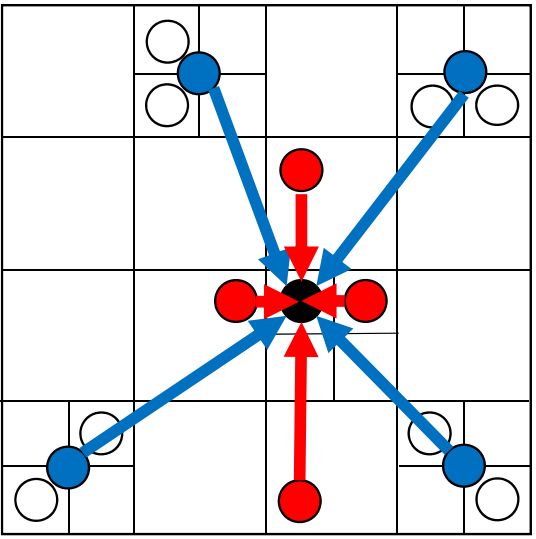
- FDPSでは領域の分割にMulti-Section法を採用(Makino2004)
  1. x軸方向にそって分割
  2. y軸方向にそって分割
  3. z軸方向にそって分割
  - プロセス数が2のべき乗であることを要求しない。
  - 領域は各プロセスからサンプルした粒子を使って計算負荷が均等になる様に決める。
  - APIはDomainInfo::decomposeDomainAll()
- 新しい領域に合わせて粒子の交換を行う。
  - APIはParticleSystem::exchangeParticle()

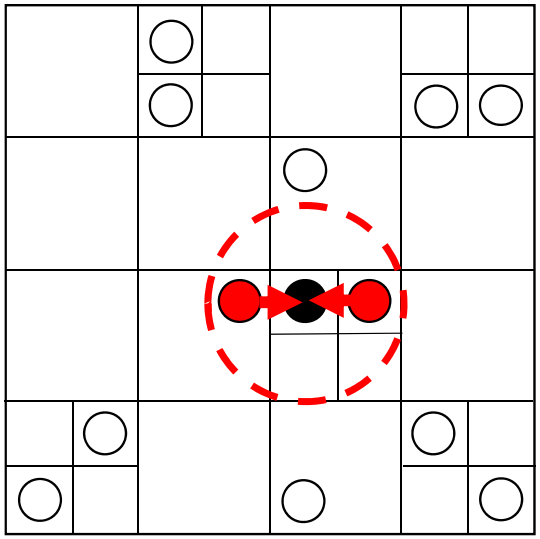
FDPSを用いた粒子シミュレーションの流れ

1. 計算領域全体を分割する。
2. 計算領域に合わせて粒子を再配置する。
3. 各プロセスが担当する粒子への相互作用を計算する。
4. 相互作用の結果を使って粒子の情報を更新する。

- FDPSは手順1,2,3を担当。
- 手順1,2,3に対応したクラスがある。
  - DomainInfoクラス: 領域のデータを持ち、領域分割を行う。
  - ParticleSystemクラス: 粒子のデータを持ち、粒子交換を行う。
  - TreeForForceクラス: 相互作用の計算を行う。
  - ユーザーはこれらのクラスにテンプレート引数として粒子クラスを与え実体を作り、メンバ関数を呼び出すことでそれぞれの処理を行う。

相互作用の計算





- 力の種類はTreeForForceクラスのテンプレート引数として与える。
- どちらの場合も粒子を木構造で管理する事で高速に計算可能

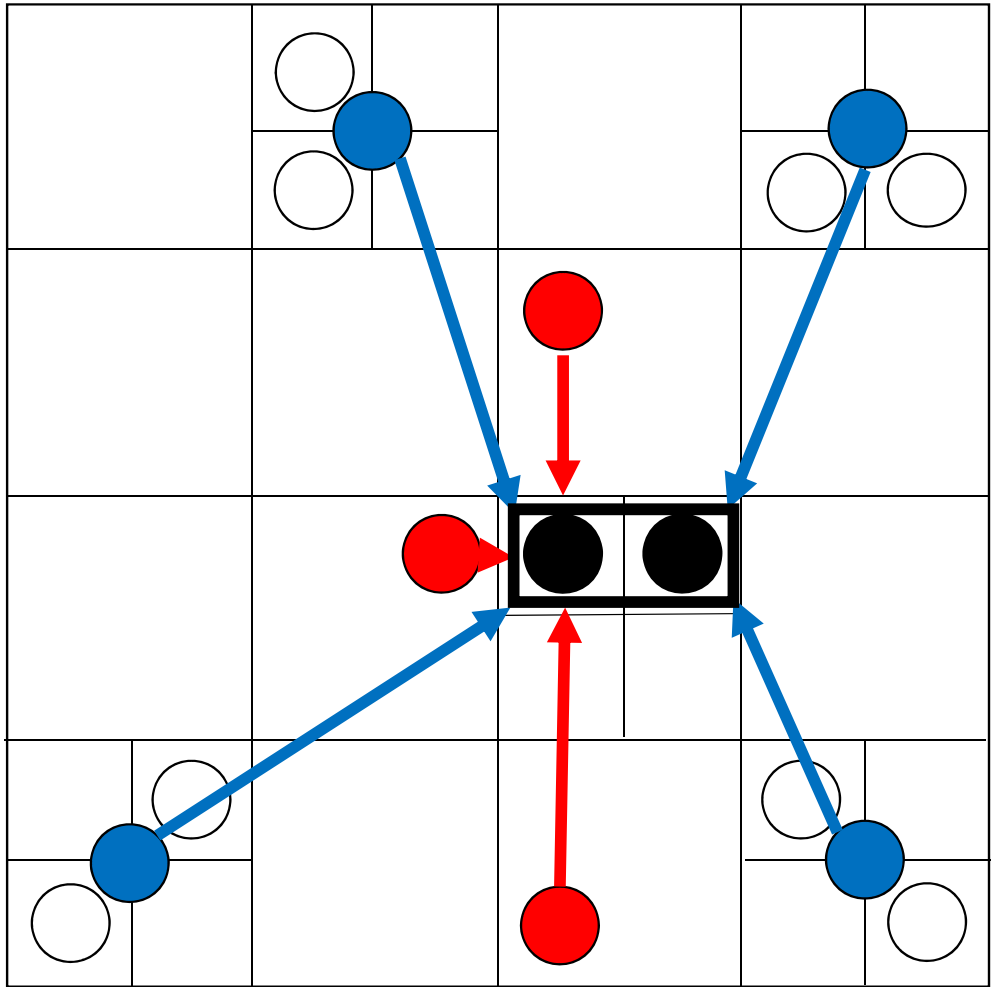
- FDPSでは相互作用を短距離力型と長距離力型の2つの型に分けている。
  - 短距離力型:
    - 分子間力等、遠くの粒子からの寄与が無視できる場合。
    - 流体シミュレーションでは、物理量は近傍粒子の重ね合わせで表現されるため、近距離力型。
  - 長距離力型:
    - 重力やクーロン力の様に遠くの粒子の寄与が無視できない場合。
    - 遠くの粒子からの寄与は小さい為、粒子をまとめて計算(Barnes-Hut tree法)。
      - SP(Super particle)
      - EP(Essential particle)
        - EPは相互作用に必要なデータだけを持っていればよい。これにより、異なる粒子種が同じ相互作用を受ける場合、FDPSでは一つのシステムとして扱う事ができる。
    - 近距離の粒子と遠距離の粒子で違う相互作用関数を与える。
      - サンプルコードでは、遠距離力は単極子までの近似をしているので同じ関数を使用。

並列計算機における相互作用計算の手順 (Makino2004, Ishiyama et al.2009)

1. 自分が担当する粒子からツリー構造を作る。
2. ツリー構造を使って相互作用に必要な粒子を交換する。
3. 送られてきた粒子情報を元にツリーを再構築する。
4. ツリー法を使って力の計算を行う。

- TreeForForce::calcForceAllAndWriteBack()で手順1-4全てが実行される。
- FDPSでは全ての手順でOpenMPによる並列化がされている。

相互作用の計算



- Barnes の方法を使う。(Barnes 1990)
  - 1粒子毎にツリーをたどるのではなく、近傍にいる複数の粒子をまとめてツリーをたどる。
  - ツリーをたどる回数が減らせる。
  - 複数の粒子が同じ粒子群と相互作用するため、SIMD化が可能。
- Ver2.0以降ではアクセラレータを有効に利用するために、Multiwalk法(Hamada et al. 2009)も利用可能。
  - 複数の粒子グループについて複数の相互作用リストを作り、それらをまとめてアクセラレータに送り計算する。

まとめ

- FDPSは以下の流れで粒子シミュレーションを実現する。
  - 領域分割
    - MS法
    - APIはDomainInfo::decomposeDomainAll()
  - 粒子交換
    - APIはParticleSystem::exchangeParticle()
  - 相互作用計算
    - Barnesベクトル化したツリー法により計算
    - APIはTreeForForce::calcForceAllAndWriteBack()