総合演習第1ラウンド資料
No.3

滝澤 真一朗

アジェンダ

- 単純並列化
- N体問題並列化アルゴリズム
  - 粒子登録法
  - 領域分割法

単純並列化の概要

- particles配列をプロセス数で分割
- 各プロセスでparticles配列の異なる範囲を計算
- 計算後、全プロセスでparticles配列を同期

- 実装概要
  - main関数内で、各プロセスの担当粒子数、担当範囲を計算
  - do_step関数の引数にそれらを取る
  - do_step終了後 MPI_Allgather

main関数

```
part_cnt = n_particles / comm_size;
start = myrank * part_cnt;

/* calculation */
for(i = 0; i < n_steps; i++){
  if (myrank == 0) {
    printf("%d\n", i);
    fflush(stdout);
  }
  do_step(start, part_cnt);
  MPI_Allgather((double*)&particles[start], 7 * part_cnt, MPI_DOUBLE,
                (double*)&particles[0], 7 * part_cnt, MPI_DOUBLE,
                MPI_COMM_WORLD);
}
```

各プロセスの担当粒子数

各プロセスが処理する最初の粒子の位置

全プロセスで、最新のデータを共有

do_step関数

```
int last = start + len;
for(i = start; i < last; i++) {
  ...
  for(j = 0; ...
}

for(i = start; i < last; i++){
  ...
}
```

アルゴリズムの改良

単純並列化の問題点

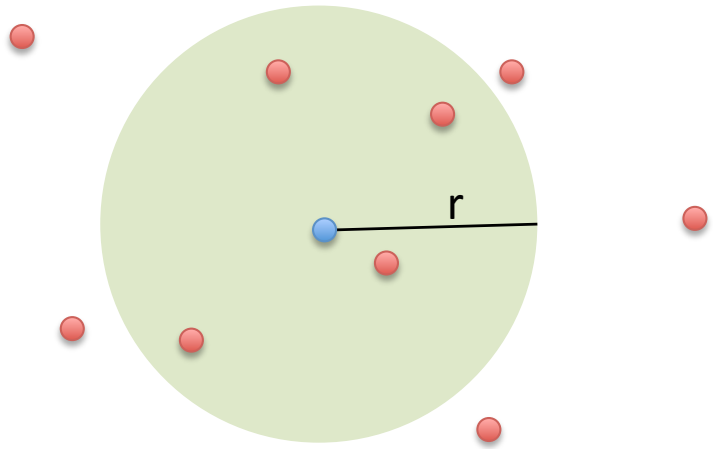
- 1粒子の次の状態を求めるときに、他のすべての星との関係を計算
  - カットオフ距離以上離れた点は考慮しなくていい

改善手法

- 粒子登録法
  - 粒子ごとに、その近傍粒子との関係のみを計算
- 領域分割法
  - 空間分割して、近傍空間に属す粒子との関係のみを計算

粒子登録法

- 各粒子ごとに、近傍粒子のリストを作成
- 近傍粒子から受ける影響のみを計算
  - do_stepの内側のループでは、近傍粒子リストを走査
- 最低1つの粒子が以前（前回リストを更新したとき）の位置より一定距離動いた場合に、リストを更新



粒子近傍リスト

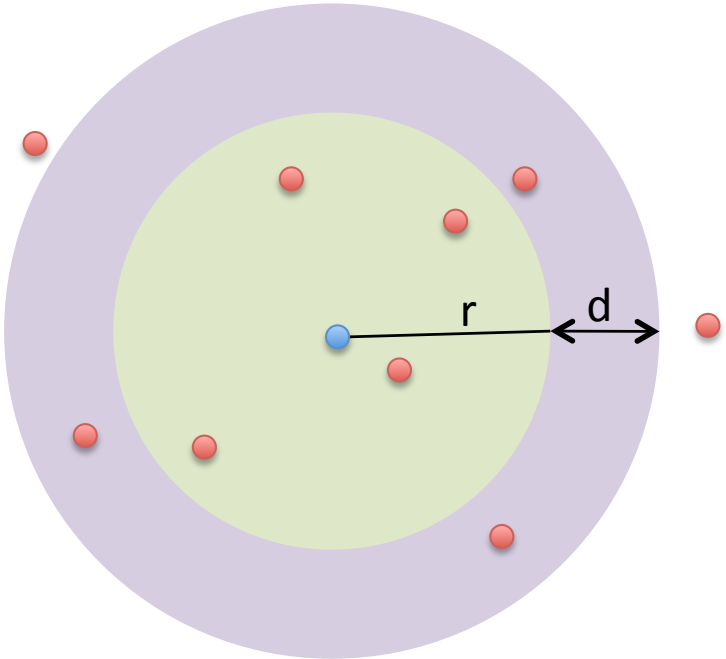
0	1 4 6 8
1	0 4 8
2	3
3	2 5

…

問題点　rの境界付近にいる粒子からはいつ影響を受けるようになるかわからない

粒子登録法

- 影響を受ける範囲よりも、少し広い範囲内の粒子をリストに入れる
  - リスト再構成の回数を削減
- リスト再構成するときの移動距離: d/2



MPI並列化方法

- プロセス毎に異なる粒子を計算
- ステップ毎に粒子が移動するので、必要に応じて、粒子情報を同期

コア部分擬似コード

main関数

```
初期化処理

build_list(...); //担当粒子の、近傍粒子リストを作成

for(i = 0; i < n_steps; i++){
  do_step(担当粒子範囲);
  プロセス間で粒子情報を交換

  if（最低1つの粒子が d/2 以上の距離移動した）{
    build_list(...); //担当粒子の、近傍粒子リストを更新
  }
}
```

do_step関数

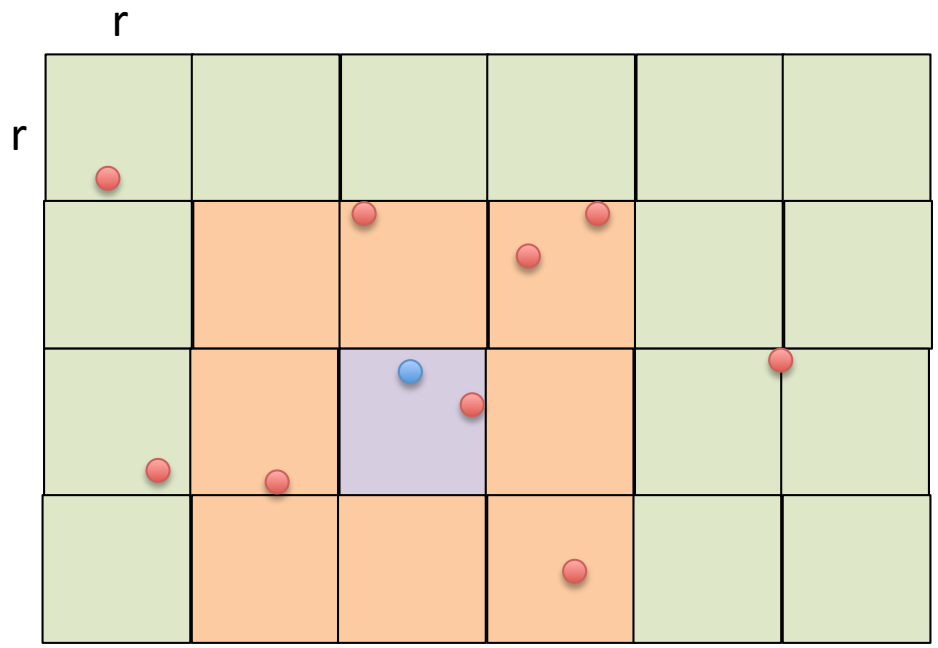
```
for（担当粒子すべてに対して）{
  for（その粒子の近傍粒子、すべてに対して）{
    重力加速度計算
  }
}
for（担当粒子すべてに対して）{
  新しい位置、速度を計算
  移動距離をチェック
}
```

build_list関数

```
for（担当粒子すべてに対して）{
  for（全粒子に対して）{
    距離を計算
    範囲内ならば、近傍リストに登録
  }
}
```

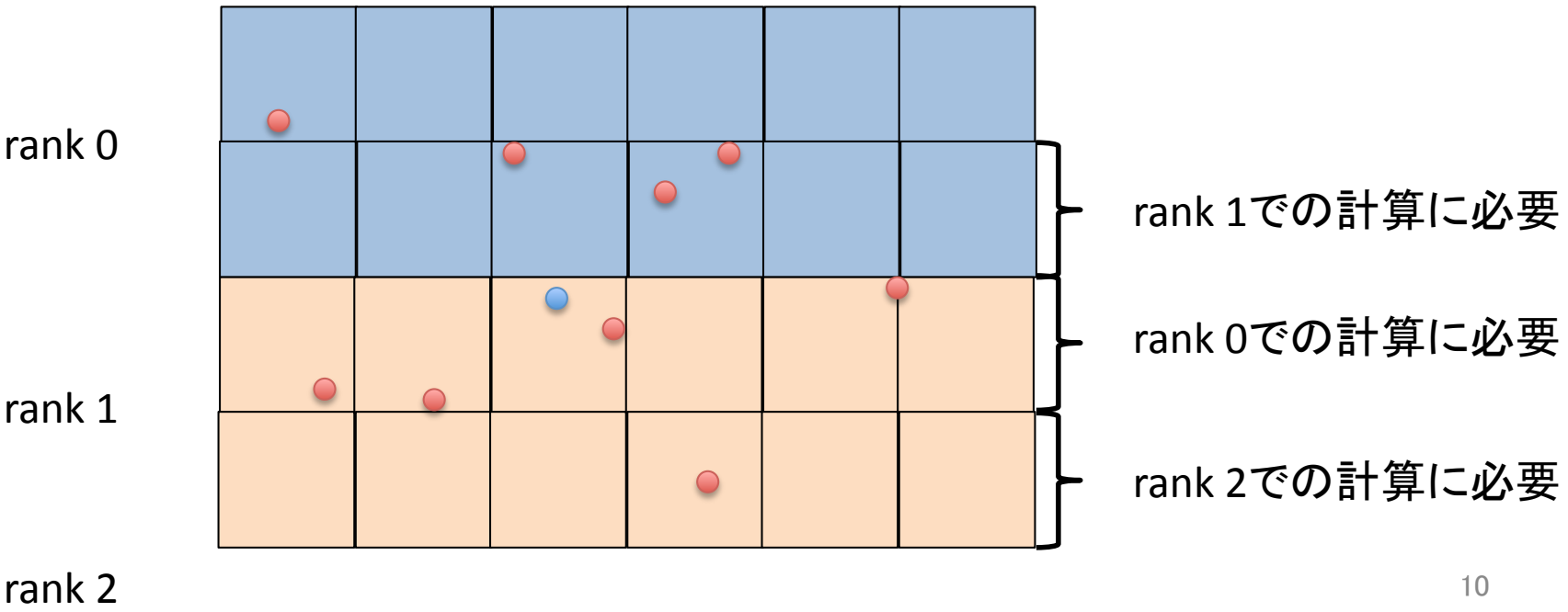
領域分割法

- 空間をカットオフ距離四方の正方形格子で分割
- ある領域に属する粒子の計算には、隣接8領域内の粒子情報のみ用いる



領域分割法

- プロセスは、行、または（かつ）、列方向に格子を分割、属する粒子の計算を担当
  - 隣接面の格子ない粒子の情報をプロセス間で交換しあう必要あり ⇒ しかし、通信はこれだけ!!



領域分割法の注意点

- 粒子データを領域に沿ってソートする必要がある
  - particles配列中の粒子の位置はランダム
  - 一方で、verifyの時には、最初の順番通りに並んでいなければならない

⇒ ソートする際には、もとのparticles配列中のインデックスも保持すること
  - 格子にIDを割り振り、粒子の属す格子IDも持つと良い

```
typedef struct {
  double m;
  double x, y;
  double vx, vy;
  double idx;
  double cellid;
}
```

課題

- N体問題の逐次実装のプログラムを並列化せよ
- 提出物
  - ソースコード本体（mpi_main.cのみでよい）
  - 実行結果、アルゴリズム、工夫点をまとめたレポート
- 締め切り:11/9
  - 満足いくまで、2/1まで何度でも再提出を受け付けます