1H-05

粒子法における空間充填曲線を利用したハイブリッド並列化

辻祥太† 藤井昭宏† 田中輝雄†
工学院大学†

1 はじめに

近年のアーキテクチャでは１ノード内のコア数が増加傾向にあり、計算速度において性能を向上させるためにノード内の並列性の抽出が必要となる。今回対象とする粒子法では、一般的にノード内においては粒子番号順にスレッドに割当てが行われるが、時間進展に伴う粒子の移動により、近傍粒子との相互作用計算の際にメモリアクセスがまばらになるという問題がある。

本研究では、並列性を抽出しメモリアクセスの局所性を向上させるために、ノード内並列化に対して空間充填曲線を適用する。空間充填曲線は、分散メモリ環境での動的負荷分散に利用されるが、分散メモリ・共有メモリの両方に適用することによる性能改善を目的とする。

今回はその前段階として、スレッド並列化に対して空間充填曲線を適用し、その性能評価を行った。

2 空間充填曲線による動的負荷分散

空間充填曲線を用いた負荷分散では、領域を分割し、各領域にある粒子のデータが対応するプロセスに分配された形で計算が行われる。領域の分割は、各領域に含まれる粒子数がほぼ均等になるように行われ、空間充填曲線を辿ることで粒子をカウントしていく。空間充填曲線は部分空間を局所的に細分化することができ、粒子が密に存在する空間を細かくカウントできるので、負荷分散の精度が保たれる。図１に、２次元での空間充填曲線を用いた負荷分散処理の手順を示す。

まず計算領域を囲うような正方形の空間を考え，バケットと呼ばれる構造格子に区切る．正方形の空間を考えるのは，細分化された部分空間をバケットがまたがないようにするためである．次に空間充填曲線により正方形の空間を部分空間に細分化する．

Hybrid parallelization of Particle method with Space-Filling Curves
Tsuji Shota†, Akihiro Fuji†, and Teruo Tanaka†

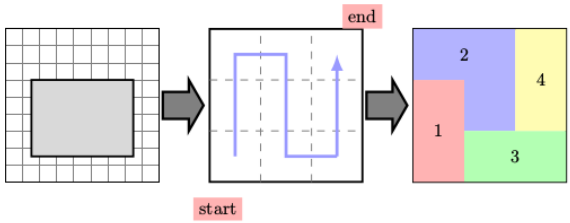

図 1 空間充填曲線を用いた領域分割

今回用いたペアノ曲線は縦横をそれぞれ3 等分するもので，一度 に9 つの部分空間に細分化される．図は細分化を1 度行った様子である．各部分空間に対して，粒子が密に存在する場合はさらに細分化を行い，これを繰り返す．最後に，細分化された空間充填曲線を矢印に沿って辿ることにより部分空間に含まれる粒子の累積和をとりながら，領域を決定していく．空間充填曲線を辿る処理は，深さ優先で再帰的に行う．例として，領域の細分化処理のアルゴリズムを載せる[1]．

Algorithm 1 Make Recursive Tree

```
function makeTree(Leaf, depth, pattern, offset)
width_local = width_whole ÷ 3^depth
Count N_p of [0...width_local]^2 from offset
if N_p < N_min or depth > depth_max then
    Leaf.end=true
else
    for i = 1...N_child do
    Set s_pattern, s_offset by pattern
    makeTree(Leaf.child[k],depth+1,s_pattern,
s_offset)
```

引数Leaf，depth は細分化された子とその子の木構造での深さである．最初の呼び出しでは子は計算領域全体となり，深さ0 である．引数pattern は木構造において子空間を辿る順序番号である．引数offsetは子空間の正方形領域の左下の点を示すものである．

makeTree 関数では最初に，正方形領域の一辺のサイズ*widthlocal* を求め，そこに含まれる粒子*Np* をカウントする．*widthwhole* は計算領域全体の正方形の一辺のサイズであり，ペアノ曲線

を用いた場合3 のべき乗となっている．そのため深さから*widthlocal* が求まる．パラメータとして*Nmin* と*depthmax* があり，それぞれ，1つの子空間に含まれる最小粒子数，木構造を構築する際の最大深さである．*Np* が最小粒子数より少なくなった場合，または*depth* が最大深さを超えた場合，そこで子空間での木の構築を終了し，親へ戻る．上記の条件に合致しない場合，さらに子空間の細分化を行う．*Nchild*はある子空間を親として分割してできる子空間の個数であり，2 次元の場合，ペアノ曲線は9 分木となるため，その個数は9 個となる．子に対して順にmakeTree 関数の呼び出しを行う．以上が再帰による領域細分化（木構造の構築）処理となる．

3　共有メモリ環境での並列化手法

空間充填曲線を用いることで、スレッドが担当する領域内の粒子の空間的局所性が保たれる。シミュレーションが進むにつれて各スレッドが担当する領域内の粒子数が不均一になる。これに対し、空間充填曲線による負荷分散を再度行うことで、領域内ごとの粒子数を均一に調整する。この際，シミュレーションには流体粒子，壁粒子があり[2]，それらを区別せずカウントして分配を行う．スレッド並列化における空間充填曲線を用いた負荷分散は，全体の領域がどのように分割されたかを表すテーブルを更新することで行われる．各スレッドはそのテーブルを参照し，自身が担当する領域内の粒子について計算を行う．

4　数値実験

4.1　実験内容と計測環境

数値実験として，相互作用計算の際に，粒子数を均等に分割し各スレッドが粒子番号順に処理を行うものと本研究の空間充填曲線により分割された領域を各スレッドに割当てて処理を行うものの計算時間を比較した．計測環境は，東京大学が提供しているReedbush-UのIntel Xeon Broadwell-EP @2.1GHz(1 ノードあたり36 コア)である．データセットは，2 次元の水柱崩壊を16,230 個の粒子を用いてシミュレーションしたものである．500 タイムステップ分の実行での総計算時間の比較と，最初の200 タイムステップ分における各タイムステップでの計算時間の比較を行った．どちらの実験においても，空間充填曲線による動的負荷分散は100タイムステップ毎に行うこととした．

4.2　実験結果

従来手法のスレッド並列化したものと空間充填曲線を適用しスレッド並列化したものの200 タイムステップでの経過時間を図2 に，タイムステップごとの経過時間の様子を図3 に示す．スレッド数を増やした場合，従来手法と本研究での手法のどちらも高速化されていることが分かるが，従来手法がどの場合においても高速であるという結果になった．タイムステップごとに比較を行うと，最初の十数ステップにおいては本研究の手法がより高速であり，あるステップを堺に従来手法より遅くなるという結果になった．

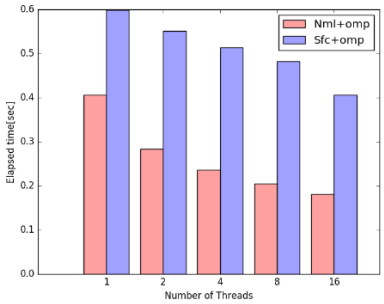

図 2　500 タイムステップでの計算速度比較

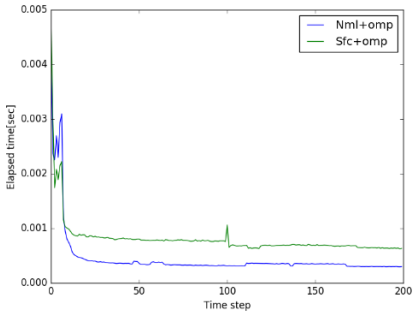

図 3　各タイムステップでの計算時間

5　おわりに

空間充填曲線の粒子法への適用を，スレッド並列化に対して行った．従来手法が本研究の手法より高速であるという結果となった．これは，流体粒子，壁粒子を区別せずに領域分割して並列化を行っているため，実質的な計算量に不均衡が生じていることが原因と考えられる．今後は，上記の不均衡による並列化への影響の解析を行っていきたい．

参考文献

[1] 都築怜理．"GPU スパコンにおける動的負荷分散を用いた大規模粒子法シミュレーション"．東京工業大学，修士論文，2016.
[2] 越塚誠一，柴田和也，室谷浩平．粒子法入門．丸善, 2014.