B-015

GPU 上での粒子法陽解法におけるバケット順データレイアウト

Bucket-Ordered Data Layout for MPS Method on GPUs

富岡 我空† 富永 浩文‡ 吉田 明正†‡
Gaku Tomioka Hirobumi Tominaga Akimasa Yoshida

1 はじめに

粒子法 [1] は，流体力学シミュレーションに広く用いられる数値解析手法であり，工学分野などで応用されている．大規模な粒子数を用いたシミュレーション [2] では，膨大な計算コストを要するため，GPU による高速な並列処理 [3][4] や，メモリアクセス効率を考慮したデータレイアウト最適化 [5][6] に関する研究が行われている．

本稿では，非圧縮性流体を解析する MPS（Moving Particle Semi-implicit）法の陽解法を対象とし，その高速化を目指す．陽解法では，圧力勾配項や粘性項などを陽的に計算することで実装が比較的容易である反面，数値安定性の制約により時間刻み幅が小さくなりやすく，計算量が増大する傾向がある．そのため，データアクセスの効率化による高速化が期待される．

そこで，本稿では粒子の空間的配置に応じてデータをバケット単位で整理し，アクセス効率を向上させるバケット順データレイアウトを提案する．性能評価では，NVIDIA RTX A5500 上にて 19,136 粒子の 3 次元水柱崩壊のシミュレーションを行い，提案手法の有効性を確認する．

2 粒子法陽解法

本稿の対象である粒子法陽解法のアルゴリズムと近傍粒子探索について説明する．MPS 法における陽解法では，次のような基本方程式を用いる．式 (1) は運動量保存則に基づくナビエストークス方程式であり，式 (2) は圧力と密度の関係を示すものである．

$$\frac{D\mu}{Dt} = -\frac{1}{\rho}\nabla P + \nu\nabla^2\mu + g \tag{1}$$

$$\frac{\partial P}{\partial \rho} = c^2 \tag{2}$$

P は圧力，ρ は密度，μ は流体の速度，ν は動粘性係数，g は重力加速度，c は音速である．これらの方程式を陽的に計算するためタイムステップの計算コストが低い．陽解法における 1 ステップあたりの処理は以下の通りである．

(i) 粒子をバケットに格納する．
(ii) 粘性項と重力項から仮の加速度を求める．
(iii) 仮の加速度から仮の位置と速度求める．
(iv) 剛体衝突の運動量を計算する．
(v) 仮の圧力を求める．
(vi) 圧力勾配項から加速度の修正量を求める．
(vii) 加速度の修正量から更新後の位置と速度求める．
(viii) タイムステップ後の圧力を求める．

この手法では，(ii)，(iv)，(v)，(vi)，(viii) のような圧力勾配の計算時に，各粒子は影響半径内に存在する他の粒子との相互作用を計算する必要がある．粒子間に依存関係がないため，各粒子の処理は並列化できる．

本稿では，近傍粒子の探索効率を向上させるためにバケット法を導入している．バケット法は，シミュレーション空間を影響半径サイズの立方体に分割し，各粒子をその所属するバケットに割り当てる方法である．1 つの粒子に対する相互作用は，自身が属するバケットを中心とし，x, y, z 方向にそれぞれ ± 1 範囲にある 27 個のバケット内の粒子が対象となり，探索範囲が限定され，全体の計算量が削減される．

3 デバイスメモリにおけるデータレイアウト

本稿では，粒子法陽解法の CUDA 実装において，計算効率を向上させるために，常にバケット順のデータ配置を維持するデータ構造を提案する．本章では，従来の粒子番号順によるデータ管理の課題点と，それを解決するために導入したバケット順データレイアウト手法について述べる．

3.1 従来の粒子番号順データレイアウト

従来の粒子法における CUDA 実装では，粒子の位置や速度などの物理量を格納するために線形リスト法が一般的に使用されてきた．この手法では，各バケットに含まれる粒子の番号をリスト形式で管理し，近傍粒子探索を行う際に，このリストから粒子番号を参照して各粒子のデータにアクセスする．粒子データは AoS（Array of Structures）形式でメモリ上に保持されており，粒子番号順でメモリ空間に配置されている．

この配置方法では，空間的に近接している粒子同士がメモリ上では離れた場所に存在する可能性が高く，近傍粒子のデータへアクセスする際に非連続なメモリアクセスが頻繁に発生する．CUDA においては，グローバルメモリのアクセスが連続しているほど高速に処理されるため，このような非連続なアクセスは計算性能を低下させる．

3.2 提案するバケット順データレイアウト

本手法では，メモリアクセスの効率を改善するため，粒子データをバケット順に連続するように格納するデータレイアウト方式を採用した．図 1 に示すように，シミュレーション空間はバケットに分割されており，各粒子はその位置に応じてどこかのバケットに分類される．このバケット分割を利用して，GPU のグローバルメモリ上では粒子データをバケット順に連続して格納するように設計した．

粒子データは「バケット番号」を基準にして,「バケット数 × バケット内最大粒子数」分の固定サイズとしメ

† 明治大学大学院 先端数理科学研究科 ネットワークデザイン専攻
Network Design Program, Graduate School of Advanced Mathematical Sciences, Meiji University
‡ 明治大学 総合数理学部
School of Interdisciplinary Mathematical Sciences, Meiji University

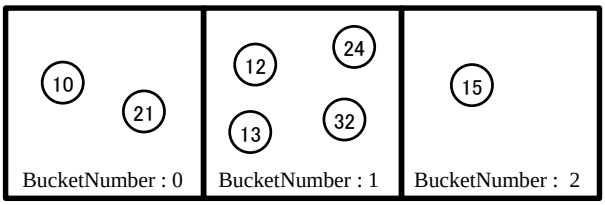

図 1　バケット内の粒子の例.

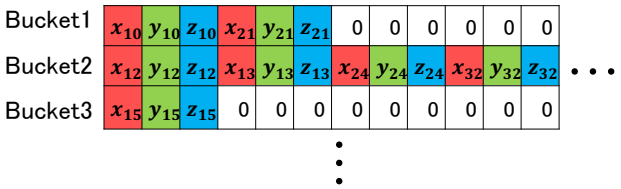

図 2　バケット順データレイアウト.

モリ上に配置される．図 2 に粒子の物理量のバケット順データレイアウトを示す．各粒子の位置・速度ベクトルなどの物理量は AoS 形式でメモリ上に格納される．各粒子には「バケット番号 b」と「バケット内のインデックス j」が割り当てられ，物理量にアクセスする際に物理量のインデックスを b×最大粒子数+j×3 とする．

粒子はタイムステップごとに移動するため，所属するバケットやデータの格納位置も毎回変化する．これにより，各タイムステップの冒頭で，粒子の空間位置に基づいて再度バケットに再格納し，データ配列内の適切な位置に粒子情報を更新する処理が必要となる．本手法では，この再格納処理を GPU 上で並列に行うことで，処理時間の増加を抑えつつ，バケット順データ構造を維持している．このデータレイアウトにより物理空間で近い粒子同士がメモリ上でも近接して配置されるため，近傍粒子探索計算において必要となるデータが連続的に読み出せるようになる．

4　RTX A5500 搭載 Xeon サーバ上での性能評価

本章では，NVIDIA RTX A5500 を用いてバケット順データレイアウトを実装した粒子法陽解法の性能評価を行う．

4.1　性能評価環境

本性能評価においては，粒子数を 19,136，タイムステップを 0.0005[s] として，1 秒分 (2000 フレーム) の 3 次元の水柱崩壊のシミュレーションを行った．性能評価に用いたマシンの構成を表 1 のとおりとする．

4.2　バケット順データレイアウトによる CUDA 実装の性能評価

図 3 に粒子番号順およびバケット順のデータレイアウトにおける粒子法陽解法の処理時間の結果を示す．スレッド数/ブロック設定は 64, 128, 256, 512 の 5 種類の実行時間を測定した．

すべてのスレッド数設定において，粒子番号順よりもバケット順でデータにアクセスするほうが処理時間が短くなっている．128 スレッド/ブロック設定では，粒子番号順による処理時間が 2.596 秒であったのに対し，バケット順では 1.828 秒となり，29.6%の高速化が得られた．最も高速な結果は，256 スレッド/ブロック設定時のバケット順アクセスの 1.809 秒であり，このときの粒子番号順は 2.592 秒であったため，30.2%短縮されている．これらの結果から，バケット順データレイアウトの有効性が示された．

表 1　性能評価マシンの構成.

CPU	Intel Xeon Gold 6326 2.9GHz 16 コア
GPU	NVIDIA RTX A5500
メモリ	256GB
OS	Ubuntu 20.04LTS
CUDA Toolkit	11.6

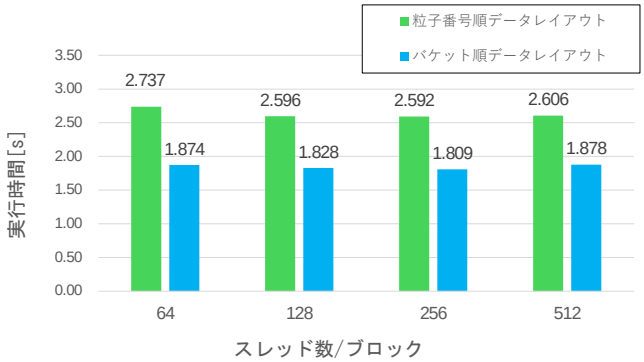

図 3　バケット順データレイアウトの性能評価 (19,136 粒子).

5　おわりに

本稿では，GPU 上でのバケット順データレイアウト手法を提案した．NVIDIA RTX A5500 上での 3 次元水柱崩壊のシミュレーションを用いた性能評価の結果，提案手法は、256 スレッド実行において、粒子番号順データレイアウトと比較して 30.2%の実行時間短縮が得られ，本手法の有効性が確認された．今後の課題としては，さらなる高速化を図るため，本手法をマルチ GPU 環境に拡張することが挙げられる．

参考文献

[1] 越塚誠一, 柴田和也, 室谷浩平．粒子法入門 流体シミュレーションの基礎から並列計算と可視化まで C/C++ソースコード付，丸善出版, 2014.

[2] Kohei MUROTANI: Large Scale Numerical Simulation Reproducing of Tsunami Behavior against a Station Building, 公益財団法人 鉄道総合技術研究所 QR of RTRI, Vol. 60, No. 1, Feb. 2019.

[3] Manfred Orsztynowicz, Hideharu Amano, Kenichi Kubota, and Takaaki Miyajima: Exploiting temporal parallelism in particle-based incompressive fluid simulation on FPGA, 2020 Eighth International Symposium on Computing and Networking, 2020.

[4] Shuai Zhang, Wenjin Gou, Yuqi Wang, Jifa Zhang and Yao Zheng: Direct numerical simulation of atomization by jet impact using moving particle semi-implicit method with GPU acceleration, Springer Computational Particle Mechanics, Vol.9, 2022.

[5] 齊藤大輔, 吉田明正: MPS 法における単精度 AVX-512 Intrinsics を用いたループ/SIMD 並列処理，情報処理学会研究報告，Vol.2025-HPC-198 No.56, 2025.

[6] 富岡我空，吉田明正: GPU 上での粒子法陽解法におけるデータアクセス順序を考慮したデータ再配置，情報処理学会第 87 回全国大会, 6J-07, 2025.