TSUBAME 共同利用　令和 3 年度　学術利用　成果報告書

利用課題名　LRnLA アルゴリズムを用いた物理シミュレーション
英文：Simulation of Physical Processes with LRｎLA Algorithm

善甫 康成
Yasunari Zempo

法政大学 情報科学部
Computer and Information Sciences
http://cis.k.hosei.ac.jp/

DiamondTorre LRnLA アルゴリズムは、クロスステンシル数値計算スキームのためのデータの局所化に優れた temporal blocking アルゴリズムである。ただ計算強度は高いが、菱形のデータの鳥圧 t 会はデータの coalesce および aligned データアクセスは難がある。一方 FArSh データ構造は、波面型時間ブロッキング手続きのデータ転送のために以前導入した実績がある。そこで FArSh のインデックス付けに単純な方法を、主データストレージ、1 つの菱形形状のデータ内、および DiamondTorre 間のデータ交換においても、同様にその手順のルートを決めるために最適な方法を導くことにより、この方法をさらに発展させていく予定である。

The DiamondTorre LRnLA algorithm is a temporal blocking algorithm with good data localization for cross-stencil numerical schemes. The Arithmetic intensity is high, however, the traversal of a diamond shape is not convenient for coalesced and aligned data access. FArSh data structure has been introduced previously to transfer data for wavefront-type temporal blocking procedures. Here we develop the method further by describing a simple way for FArSh indexing, as well as for space-filling curve in the main data storage, in one diamond, and in the exchange between DiamondTorre.

Keywords: LRnLA algorithms ・temporal blocking ・data structure

背景と目的

物理問題の数値シミュレーション、特に波動現象や流体力学の分野では、矩形メッシュ上のステンシルスキームが非常によく使われているが、それを使ったシミュレーションコードは、性能効率が上がらないことが多い。そこで、一般的な入れ子になったループではなく、wavefront 型のアルゴリズムを用いると、データの再利用による効率化が図れる。また Wavefront 型アルゴリズムを用いると計算ウィンドウを作ることも可能である [1]。

LRnLA (Locally Recursive non-Locally Asynchronous) アルゴリズム [2,3] の中には、wavefront に似たアルゴリズムもある。Torre という LRnLA アルゴリズムは、1D1Tの wavefront 型のアルゴリズム[1]と類似している。このアルゴリズムは、GPU-CPU データ交換の性能効率がり、効率が高い並列計算ができる temporal blocking 法[4]の一つである。DiamondTorre LRnLA アルゴリズム[3]も wavefront 型であり、十字型のステンシルに対してより良い局所性を提供する。十字型のステンシルは特に FDTD のような波動現象の数値計算スキームで頻繁に使用した効率の高い計算例もある[3]。DiamondTorre を使う上で難しいのは、データアクセスの局所性、ベクトル化のためのデータ・アライメントおよび並列アクセスのための coalescing の原則を満たすデータレイアウトを探すことである。

これまでの研究でキューブステンシルに合わせる FArSh というデータレイアウトがあることを紹介した。十字型ステンシルの数値計算スキームの局在化では、2 次元シミュレーションでは菱形のタイリングが[5]、3 次元シミュレーションでは正八面体元にしたタイリングが最適である。ただ、この種の形状に沿ってデータアクセスを行うことは、かなり複雑でありアクセスの coalescing が不十分である。

我々のプロジェクトの目的は、データレイアウトに主眼を置き、 (1) 主データストレージ、(2)DiamondTorre の中のデータレイアウト、(3)DiamondTorre の間のデータ交換のためのデータ構造を開発し、配列のインデックスの計算のオバーヘッドを減少させることである。

概要

先ず 2D 計算の 1 セルの幅の十字型のステンシルを例として DiamondTorre を構築から始める。

2D1T・DiamondTorre というアルゴリズムは、菱形を底面とする $x-y-t$ 柱体で表す、2D1T temporal blocking を実現する。3Dシミュレーションの場合には、3 番目の z 軸では通常のループを用いる。

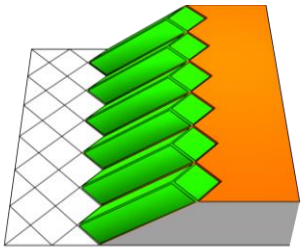

図 1. 2D1Tで二次元の DiamondTorre シミュレーションをあらわす。wavefront と同様に領域全体を覆う。緑色で示された DiamondTorre では並列計算を行うことができる。

DiamondTorre のコードの要点は、tループである。ループの繰り返しを行っている間に菱形形状の中のセルのアプデートを実施する。ループの繰り返しで、それぞれの菱形形状は1セルにx軸の右にずれていく。シミュレーション領域の右から始めると、DiamondTorre は wavefront と同様に領域全体を覆うことができる(図 1)。

(1) 主データストレージ

データの初期化および出力は、同期の瞬間に可能である(図 2(1))。 DiamondTorre の菱形の底面はこのデータからロードする。なので、主データストレージは菱形のタイルで構成される。d次元の一般的な場合、多次元配列を使用して、配列$\mathbf{j} = (j_1, j_2, \ldots j_d)$ での要素のセルの座標は $\mathbf{r} = j_1\mathbf{e}_1 + j_2\mathbf{e}_2 + \ldots + j_d\mathbf{e}_\mathrm{d}$である。ここで、

$$\begin{aligned}\mathbf{e}_1 &= (3-\mathrm{d}, -1, -1, \ldots, -1)\\ \mathbf{e}_2 &= (1, 1, 0, \ldots, 0)\\ \mathbf{e}_3 &= (1, 0, 1, \ldots, 0)\\ &\vdots\\ \mathbf{e}_\mathrm{d} &= (1, 0, 0, \ldots, 1)\end{aligned}$$

である。

d次元の配列はZ階数(モートン符号)により格納する。階数Rの配列は2^{Rd}の要素をもつ。この配列をそれぞれの軸には細分化すると、2^dの配列を取得する、その配列は$2^{(R-1)d}$の要素をもつ。細分化を進めると、最後には1セルになる。つまり、再帰的なデータ構造となっている(図 3)。

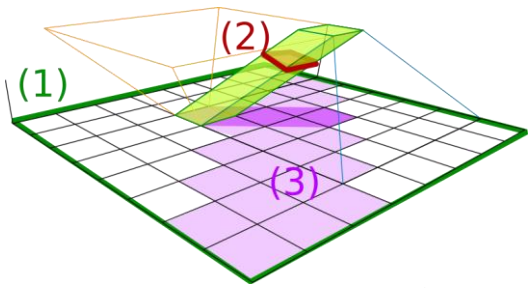

図 2. DiamondTorre のために開発されたデータ構造。

DiamondTorre の菱形の底面はその配列の中の2^{rd}の部分位置と一致する。セルのデータはAoS (array of structure) 方式で整理される。

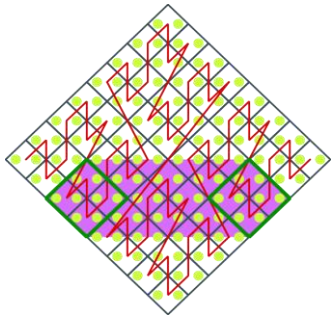

図 3. 主データストレージは再帰的データ構造をしている。ピンクは 1 つの DiamondTorre で更新されるデータである。

(2) DiamondTorre の中のデータレイアウト

DiamondTorre の中のデータは GPU レジスターにロードされて、そのデータに計算を行うので、菱形の中のデータ交換ができるだけ早い方が良い。(図 2(2))。

$M \times M$形状の菱形の場合、1つの CUDA スレッドにMセルの計算を行うことができる(図 2)。それを$\mathbf{c} = (1, 1, \ldots, 1)$ とする。番号が$\mathbf{j} = (j_1, j_2, \ldots, j_d)$ のセルから始まって、$\mathbf{j} + i\mathbf{c}, i \in Z$ というルートを考えると、このルート上のセルは、$\mathbf{j}_\mathrm{B} = \mathbf{j} - i_m\mathbf{c}$ のセルから $\mathbf{j}_F = \mathbf{j} + (M - i_M - 1)\mathbf{c}$ のセルまで$M \times M$の菱形の中にある。ここで、$i_m = \min s(i_s)$, $i_M = \max s(i_s), s = 1..d$ である。$\{\mathbf{j}_F\}$の集合または$\{\mathbf{j}_B\}$の集合は菱形のグノモンとなっている(形状によって決まる数、つまり図形数:gnomon)。

DiamondTorre というtループの繰り返しには、菱形は一セルにx軸の右にずれる。そこで、$\{\mathbf{j}_B\}$のセルの出力と、$\{\mathbf{j}_F\}$のセルの入力を行う。その他のセルは菱形形状内で移動させる。

そのためデータの局所とアライメントのため、$(d-1)$次元のZ階数配列を使用する。その配列の要素は、$\mathbf{j} + i\mathbf{c}$に対応するルートである。

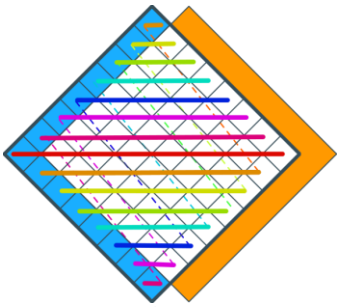

図 4. 図 2(2)菱形内のデータ構造。青は$\{\mathbf{j}_B\}$、オレンジは次の繰返しでロードして青のメモリ位置に書き換える。

(3) DiamondTorre の間のデータ交換

には、以前開発した FArSh というデータ構造を使用する(図 2,(3))。DiamondTorre には、GPU で保存されている配列から菱形の底面のデータと、FArSh からスロープのデータを読み込む。

実行後、上側の底面のデータを主データストレージに、上のスロープのデータを FArSh にセーブし、以前保存したデータを書き換える。これにより計算ウィンドウも可能になって、wavefront 上のデータ

だけ保存している。

まとめと今後の課題

TSUBAME でも実行したことがあるアルゴリズムである DiamondTorre [3]に最適なデータ構造を開発し、コードを実装した。

我々のプロジェクトにおいて開発したデータ構造を使用すると、菱形の底面をコードに実装すると非常に使いやすいものとなる。また、上述のコードで[3] に実装した2D1Tアルゴリズムだけではなく、d次元の菱形の底面でも使えるようになった。十字型のステンシルを持つ数値スキームでは、d次元の DiamondTorre を実装できるようになった。d次元の菱形のデータアクセス法を使用し、以前のコード[3]と比べ、更に高いパフォーマンスが得られるものと期待している。この結果は、他の数値計算法へも適用できるものと確信している。

参考文献

[1] Wolfe, Michael. "Loops skewing: The wavefront method revisited." International Journal of Parallel Programming **15.4** (1986): 279-293.

[2] Levchenko, V.D., Perepelkina, A.Y. Locally Recursive Non-Locally Asynchronous Algorithms for Stencil Computation. Lobachevskii J Math **39**, 552–561 (2018). https://doi.org/10.1134/S1995080218040108

[3] Zakirov, Andrey, Vadim Levchenko, Anastasia Perepelkina, and Yasunari Zempo. "High performance FDTD algorithm for GPGPU supercomputers." In Journal of Physics: Conference Series, **759**(1), 012100. IOP Publishing, 2016.

[4] Endo, Toshio. "Applying Recursive Temporal Blocking for Stencil Computations to Deeper Memory Hierarchy." 2018 IEEE 7th Non-Volatile Memory Systems and Applications Symposium (NVMSA). IEEE, 2018.

[5] Terrano, Anthony E. "Optimal tilings for iterative pde solvers." Proceedings 2nd Symposium on the Frontiers of Massively Parallel Computation. IEEE Computer Society, 1988.