TSUBAME 共同利用　平成 29 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 Algorithms

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/

従来使用してきた高性能 FDTD コード DTmaxwell を TSUBAME3.0 に移植した。このコードは LRnLA アルゴリズム DiamondTorre を使用しているが、大きな変更の必要はなく本来の性能を引出すことができた。性能をテストでは LRnLA 理論から予測される性能であることが検証できた。これは NVidia GPGPU を使ったことと LRnLA アルゴリズムの成果である。TSUBAME3.0 では完全 4D 時空間での並列化ができれば更に性能を向上できることが判明した。そこで試験的にこの並列化を行う LRnLA アルゴリズム DiamondCandy を開発した。FDTD のコードを使った検証の結果、新しい GPCPU の性能を効率的に向上させることを確認することができた。

The high-performance FDTD code DTmaxwell, which uses LRnLA algorithm DiamondTorre, has been ported to the new TSUBAME3.0 supercomputer, and its performance has been tested. The efficiency of the algorithmic method allowed to predict the performance and the prediction was verified by the test runs. This experience has shown both the advantages of NVidia GPGPU hardware and the advantages of the programming with LRnLA algorithms. The necessity of a full 4D space-time localization has emerged more distinctly with the new hardware. A new FDTD code benchmark was developed with a new LRnLA algorithm DiamondCandy, and it was verified to be more efficient on the new GPGPU.

Keywords: LRnLA algorithm, GPU, FDTD, high performance

背景と目的

近年、数値シミュレーションにおける時空間局所化が一般的な話題となっている。科学技術計算において必要とされている資源効率の高い、高度な大規模計算を実現できるからである。最先端のハードウェア技術により高性能計算が可能になるため、実施アルゴリズムの進歩が不可欠である。

時空間局所化は、幾つか報告があり、”temporal blocking”、”split tiling”、”diamond tiling”、”time skewing”、”wavefront blocking”という名前で知られている[1]。”polyhedral compilation”と”cache-oblivious algorithms”とも関連している。

これらの技術の考え方は、計算ドメインが時間層ごとに更新される伝統的な「ステップワイズ」アプローチを止めてしまうことである。ステップワイズでは、 時間層tから$(t+dt)$への更新は、総てのシミュレーションドメインセルが$(t-dt)$ からtに更新された後のみ開始されるためメモリアクセスの局所性と並列スケーリングが非効率的である。

新しい時空間局所化技術は、nD1T[$(n+1)$次元時空間]依存グラフの最適な分解をアルゴリズムの局所性および並列性を最適化するブロックを探索することが必要である。LRnLA 方法は、この問題を解決する画期的な方法の 1 つであり、理論的な基盤に基づき、複数の計算効率向上を図りつつ解析できることを確認されている。

従来、我々は光学解析を行うため FDTD 法による解析のため DiamondTorre アルゴリズムを用いてきた。TSUBAME2.5 でテストを行い、性能が極めて優れていることがわかった[2]。

最近の GPGPU コンピューティング技術は急速に進歩している。新しい GPU アーキテクチャをはじめとし GPU 間の相互接続や CUDA の進歩している。これにより性能向上だけでなく重要な機能も提供されている。優れた互換性に加えて、新しいプラットフォームで古いコードを再コンパイルのみで、最高のパフォーマンスが得られるようになっている。

TSUBAME3.0 を利用することで LRnLA FDTD コード DTmaxwell のパフォーマンスがどのように改善するか、GPGPU スーパーコンピューティング動向に鑑み、どのようなアルゴリズム開発が必要かを検討することが非常に重要となっている。そこで我々の開発した手法を TSUBAME3.0 へ適用して、その性能を分析することとした。高度な内部並列性があればピーク性能が向上すれば、それに比例して性能は向上するはずである。

これまで DiamondTorre を用いた既存の FDTD

コード DTmaxwell を TSUBAME3.0 に移植すると予想される性能向上は 1GPU あたり 3 倍、1 ノードあたり 5 倍となるはずである。

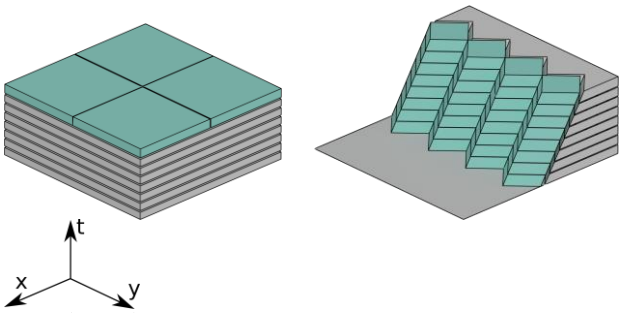

図 1 . (2D1T) の stepwise と DiamondTorre のイラスト。緑色の個々のブロックは、同期（データ交換）イベントの前に 4 つの並列プロセッサによる可能な計算を示す。

TSUBAME3.0 の重要な特徴

TSUBAME3.0（P100 の 4.76TFlops）であり、TSUBAME2.5（K20x の 1.3 TFlops）より GPU パフォーマンスが高い。また同時に、より大きなメモリスループット（P100 では 732 GB/s、K20x では 250 GB/s）がある。DTmaxwell の FDTD 法はメモリバウンドの領域内に収まる。従ってパフォーマンスとメモリの両方が 3 倍になれば 1 つの GPU でコードのパフォーマンスが 3 倍向上することが期待できる。また、PCI-e に代わって導入された NVlink 相互接続によりノード内のコードの並列効率の向上が期待できる。またメモリサイズも増加しているので（P100 GPU あたり 16GB、K20x GPU あたり 6GB）、最大限のパフォーマンスを得るために 1 つの GPU あたりのアルゴリズムパラメータを以前より自由に選択できるという特徴がある。

一方で GPU 性能の発達に伴い DiamondTorre アルゴリズムの課題が顕在化した。 DiamondTorre アルゴリズムでは、データが時間と 2 つの空間次元のみに局所化されているため、第 3 軸は CUDA スレッドを使用してステップワイズ法で処理されるところが大きな課題であった。

最近の GPU には、デバイスにもよるが数多くの SM（Streaming Multiprocessor）が搭載されている。我々のこれまでの研究によると、使用している CUDA ブロックの数が 1 つのデバイスの SM 数またはこの数の整数倍に等しい場合、DTmaxwell の性能が飽和することがわかっている。従ってドメインの規模は、並列分解が可能な程度にとる。

DiamondTorre GPU 実装では、y 軸の分解に CUDA ブロックの並列処理が使用されている。その結果、細長いドメインで最高のパフォーマンスが達成されるような設計である。

最新の GPU でのデバイスあたりの SM の数の増加に伴い、最適なパフォーマンスを得るためには、方向によって異なるアスペクト比が必要になる。高度に局所化された LRnLA アルゴリズムを使うと、オーバーヘッドが増加する。アルゴリズムの中に多彩な機能を追加した新しい実装方法を使うと、局所性を上げてもオーバーヘッドの影響を少なくすることができるはずである。

そこで最近提案された新しいタイプの LRnLA アルゴリズム DiamondCandy を用いることとした [4]。これは 3D1T の局所性を持っているが、CUDA ブロック内のスレッド間の計算は一様ではない。

従って細長いドメインのパフォーマンスは DiamondTorre よりも低いが、大規模なキューブ型ドメインでの計算処理では高い性能が出ると期待される。

パフォーマンスの検討結果

DTmaxwell コードは TSUBAME3.0 で再コンパイルした上テストを行った。これまでに達成された最高の性能は、TSUBAME2.5 の記録を、GPU デバイス 1 個当たり 3.5 倍、1 つのノードあたり 5 倍上回っていた。もちろん非同期ブロックの数が SM の数に達するとパフォーマンスは飽和する。

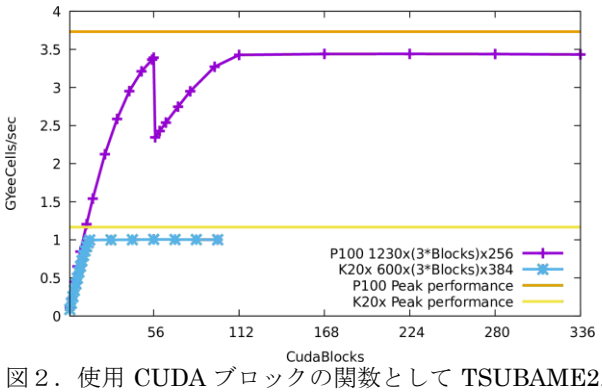

図 2 . 使用 CUDA ブロックの関数として TSUBAME2.5 と TSUBAME3.0 上の一の GPU デバイス上で達成される性能

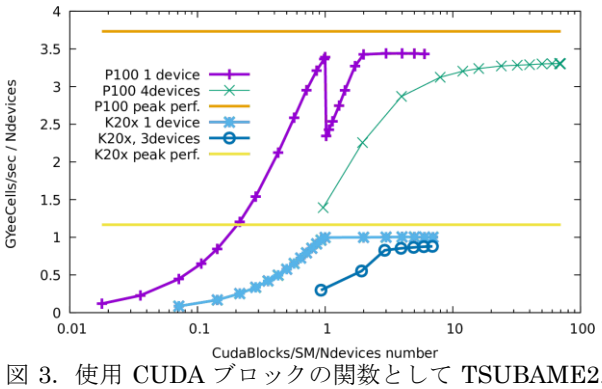

図 3. 使用 CUDA ブロックの関数として TSUBAME2.5 と TSUBAME3.0 上の一のノード上で達成される性能

SM の数は P100 の方が多いので、以前の K20x に比べパフォーマンスの向上を連続して図ることができる高い。ブロック数が SM の数の倍数でない場合、パフォーマンスが低下するので、図 2 と図 3 で横軸が 1 のところで最も高くなり一旦パフォーマンスが低下し再度向上していることがわかる。

また weak scaling についても検証の結果、非常に効果的であることがわかった。ドメインが x 軸または y 軸方向に増加すると、 並列効率が異なっ

ている。y 方向のノード並列処理は、各ステージの後、すなわち非同期 DiamondTorre 行が実行された後、ノード間の同期が必要である。

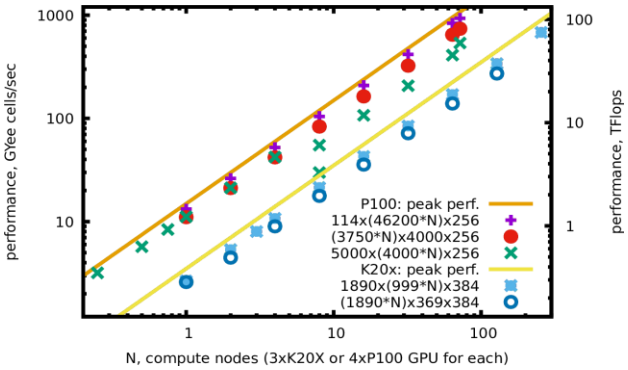

図 4．異なるドメインサイズの設定のための weak scaling 性能

4 つ以上のノードが関与すると、パフォーマンス結果が不安定になるため、ここでは最も成功した実行の結果のみ表示している。この原因は、キュー上の他のジョブの干渉である可能性がある。これは使用可能なすべてのノードが関与すると、パフォーマンスが向上するということから推測できる。

さらに x 方向のノード間並列処理のために”moving window”手法が実装されている。使用されているすべてのノードを同時に同期する必要がないためパフォーマンスの低下は見られない。

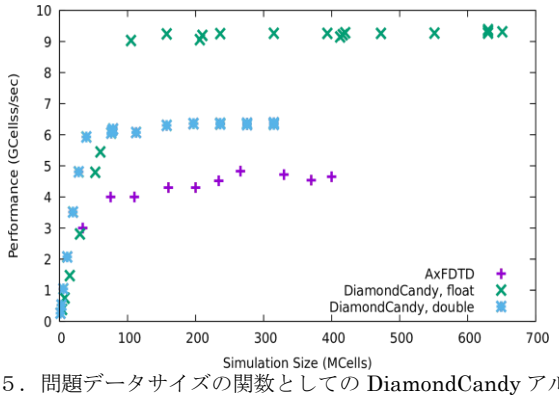

図 5．問題データサイズの関数としての DiamondCandy アルゴリズムの性能テスト結果

また新しく開発した DiamondCandy アルゴリズムでの初期ベンチマーク結果も以下に報告する。図 5 に問題のデータサイズに対するパフォーマンスの依存性を、図 6 に P100 の roofline 図と 3 つのコード(DTmaxwell、DiamondCandy、効率的な最先端の市販ソフト AxFDTD [3])の性能テストの結果を示す。新しいアプローチの利点を示す結果となっている。

まとめ

TSUBAME2.5 と TSUBAME3.0 の 2 のマルチ GPU スーパーコンピュータで DTmaxwell コードを実行した課題は、GPU ハードウェアと CUDA によってできたことである。この移植では、コードの再コンパイル以外の作業は必要なく、パフォーマンスはハードウェア機能の増加に比例して増加した。LRnLA 理論からの性能予測が可能であることが確かめられた。実際 1 つの GPU で 3 倍のパフォーマンス向上を得て、1 つのノードのパフォーマンスに対する最大 5 倍の性能向上が得られた。

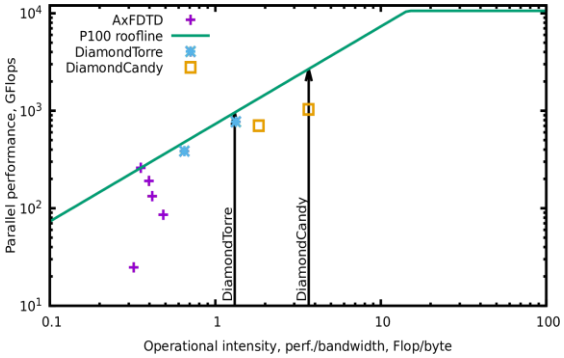

図 6．DiamondTorre アルゴリズムと DiamondCandy アルゴリズムのルーフライン制限と最大パフォーマンス結果。市販ソフト AxFDTD コードによる結果と比較している。 DiamondCandy と DiamondTorre の 2 つのマーカーは、単精度と倍精度の結果を示す。

また TSUBAME3.0 の 72 ノードで 1 秒あたり 10^{12} Yee セルの更新が達成されたが、同じ結果を得るために TSUBAME2.5 の 512 ノードが必要であった。ただドメインアスペクトによる効果が最大となるように調整している。また新しいマシンでは並列性を上げることができるので、そのための新しいアルゴリズム DiamondCandy を導入した。3 次元の局所性があり、より多くの非同期性が可能になるという特徴がある。今回、この最初のベンチマークを行いその効果を確認した。

今後の課題として新しいアルゴリズムの実装、テスト、実際問題の応用について検討する予定である。

[1] T. Fukaya and T. Iwashita. Time-space tiling with tile-level parallelism for the 3D FDTD method. In Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, p. 116–126. ACM, 2018.

[2] A. Zakirov, V. Levchenko, A. Perepelkina, Y. Zempo, “High performance FDTD algorithm for GPGPU supercomputers”, *J. Phys.: Conf. Ser.* **759** 012100, 2016

[3] AxFDTD GPU Performance (v11.1.12). http://www.acceleware.com/fdtd-solvers, retrieved Mar 2018.

[4] V. D. Levchenko and A. Y. Perepelkina. Locally recursive non-locally asynchronous algorithms for stencil computation. *Lobachevskii Journal of Mathematics*, **39**(4), 2018.