一般社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
MW2012-32,OPE2012-25,EST2012-14,MWP2012-13(2012-07)

GPGPUを用いたFDTD計算高速化における CUDA C及びCUDA Fortranの演算性能比較に関する一検討

高原 勝平　今井 卓　田口 健治　柏 達也

北見工大 〒090-8507 北海道北見市
E-mail: now_i@mail.kitami-it.ac.jp

あらまし 近年，GPU (Graphics Processing Unit) を用いた数値計算の高速化技術であるGPGPU (General Purpose Computation on GPUs) が注目されている．GPGPUプログラムの開発言語としては，汎用的プログラミング言語であるC言語をベースとしたCUDA Cが広く用いられている．一方，数値計算分野において長い歴史を持つFortran言語をベースとしたCUDA Fortranの環境も整ってきている．これらの開発言語の登場により，GPGPUプログラムの開発を比較的容易に行う事が可能となった．GPGPUプログラムの開発において，プログラミング言語の違いが数値計算の演算性能に与える影響は良く知られていない．本研究では，GPGPUを用いたFDTD計算の高速化においてCUDA C及びCUDA Fortranが演算性能に与える影響について調べた．

キーワード GPGPU，CUDA C，CUDA Fortran，FDTD法

Comparison of Computational Performance between CUDA C and CUDA Fortran in High-Speed FDTD Simulation Using GPGPU

Shohei TAKAHARA　Suguru IMAI　Kenji TAGUCHI　and　Tatsuya KASHIWA

Kitami Institute of Technology, Kitami, Hokkaido, Japan
E-mail: now_i@mail.kitami-it.ac.jp

Abstract Recently, GPGPU (General Purpose Computation on Graphics Processing Units) has been attracting attention as the technique to realize the high-speed computation. In the computer simulation using GPGPU, a CUDA C based on C language is used widely as the program development environment. A CUDA Fortran corresponding to Fortran language is also used. These programming languages can relatively-easily realize a high-level GPGPU program. In this work, the computational performances of GPGPU-FDTD simulation using the CUDA C and CUDA Fortran are investigated.

Keyword GPGPU，CUDA C，CUDA Fortran，FDTD method

1. はじめに

GPU (Graphics Processing Unit) の高い演算性能を数値計算の高速化に応用するGPGPU (General Purpose Computation on GPUs) が注目されている．従来，数値計算の高速化は，OpenMP及びMPI (Message Passing Interface) などを用いてCPU計算を並列化する事で実現されてきた[1]．一方，近年はGPUがCPUに比べて低コスト且つ低消費電力である事などから，GPUを用いた数値計算高速化が行われている[2]-[9]．GPUは消費電力あたりの演算性能が高いことから，近年のスーパーコンピュータにおいても利用されている[10]．

GPUは，元来，グラフィック用途のハードウェアであるため，高性能なGPGPUプログラムを開発するためには高度な専門的知識が必要となっていた．しかしながら，近年，数値計算に用いることを視野に入れたハードウェア及びプログラミング言語の開発により，高性能なGPGPUプログラムを比較的容易に作成することが可能となった．GPGPUプログラムの開発言語としては，汎用的プログラミング言語であるC言語をベースとしたCUDA Cが広く用いられている[11]．一方，数値計算分野において長い歴史を持つFortran言語をベースとしたCUDA Fortranによる開発環境も整ってきている[12]．しかしながら，GPGPUプログラム開発において，これら言語の違いが演算速度に与える影響は良く知られていない．

汎用的な電磁界解析手法の一つであるFDTD法[13]においてもGPGPUを用いた高速化の研究が行われており，CPU計算と比較して数十倍の高速化がなされた例も報告されている．しかしながら，計算に用いられたCPU及びGPUなどのハードウェア環境，コンパイラなどのソフトウェア環境が不明瞭な場合が見受けられた．そのため，GPGPU-FDTD計算の定量的な高速化

性能を把握する事が重要となっている.

本研究では, GPGPU を用いた FDTD 計算の高速化において CUDA C 及び CUDA Fortran が演算性能に与える影響を定量的に調べた.

2. GPGPU-FDTD 計算

時間領域の電磁界解析手法である FDTD (Finite-Difference Time-Domain) 法はマクスウェル方程式を時間と空間で直接差分化する手法である. 解析空間は直方体格子に分割され, 電界及び磁界成分が 1/2 セル離れた点に交互に配置される. 電磁界成分はリープフロッグアルゴリズムを用いて, それぞれ交互に 1/2 タイムステップ毎に更新される. FDTD 解析における計算時間の大部分は電磁界成分の更新に費やされるため, これを高速化する事が重要となる.

図 1 に GPGPU-FDTD 計算のフローチャートを示す. CUDA C 及び CUDA Fortran では, GPU 上で実行される計算をカーネルと呼ばれる関数として実装している. カーネルは各タイムステップ t 毎にホスト(CPU)側から呼び出され, GPU 上に多数実装された計算コアを用いて並列計算される. 本研究では, 電界成分の更新及び電界入力, 吸収境界の計算, 磁界成分の更新を 3 つのカーネルに分けて実装している. FDTD 計算において最も負荷の高い電磁界成分の更新を GPU 上で処理する事により計算時間を短縮する事が可能となる.

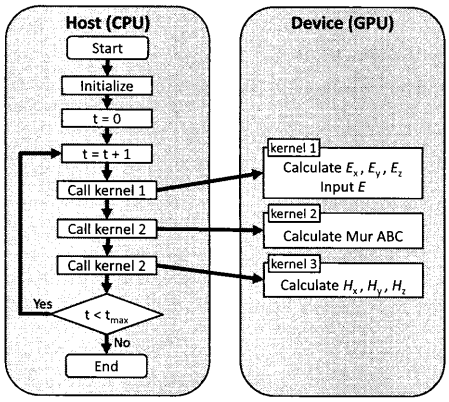


図 1　GPGPU-FDTD 計算フローチャート

3. 計算環境

表 1 及び表 2 に本研究で用いた計算機及びコンパイラを示す. 本研究では GPU としてミドルレンジ向けの NVIDIA GeForce GTX 560 Ti を用いている. また, コンパイラに関しては, CPU 計算では, 無償利用が可能である GNU compiler 4.6.2 (gcc, gfortran)を用いた. GPGPU 計算では NVIDIA CUDA 4.2 (CUDA C)及び PGI Accelerator Fortran Workstation ver. 12.5 (CUDA Fortran) を用いた. この時, 全てのコンパイラに対して最適化オプションを「-O3」に統一した. 尚, 電磁界変数は全て単精度変数を用いている. また, CUDA Fortran では, 明示的に CUDA API を用いてプログラムを実装する方法と簡便な GPU 並列化指示文を用いて実装する 2 種類の GPGPU 利用法がある. 本稿では, CUDA Fortran においても CUDA C と同様に明示的に CUDA API を用いてプログラムを実装している.

表 1　本研究で用いた計算機

CPU	Intel Core i7 2700k 3.5GHz
メモリ	DDR3-1333 2GB×4
GPU	NVIDIA GeForce GTX 560 Ti (VRAM 1GB)
OS	Windows 7 Professional (64bit)

表 2　本研究で用いたコンパイラ

CPU 計算	gcc 4.6.2
	gfortran 4.6.2
GPU 計算	NVIDIA CUDA 4.2 (CUDA C)
	PGI Accelerator Fortran Workstation version 12.5 (CUDA Fortran)

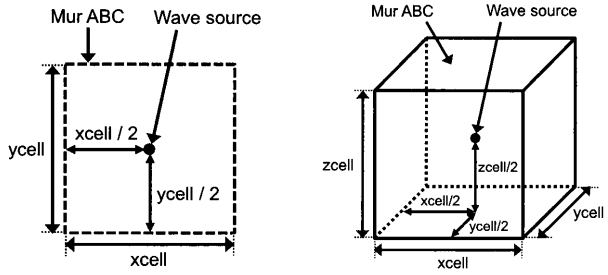
4. FDTD 計算モデル

図 2 に演算性能評価のために用いた FDTD 計算モデルを示す. 本研究では, FDTD 解析の基本的な計算時間の評価を行うため, 点波源と自由空間で構成される簡便なモデルを用いた. 点波源の周波数は 2.45GHz とし, 吸収境界条件として Mur を用いて, 2 次元及び 3 次元解析を行った. また, 各軸方向の解析領域セル数である xcell 及び ycell, zcell は, 2 次元及び 3 次元解析のそれぞれにおいて, 500～6500 及び 50～250 の範囲で変化させた. 尚, 最大のセル数は本研究で用いた GPU デバイスの VRAM 容量に基づいて設定している. また, 空間離散間隔は 2mm とした.

5. FDTD 計算における演算速度の比較

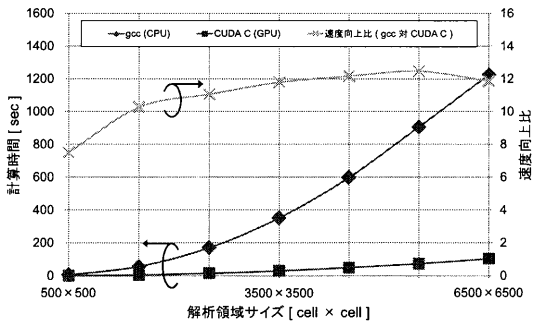
本節では, FDTD 計算の演算速度を用いて GPGPU による演算高速化性能の定量的な評価を行う. 本研究

では FDTD 計算において最も計算負荷が大きい電磁界成分の更新及び波源入力，吸収境界の計算に要する時間を測定した．この時，解析周期は 50 周期とし，3 回の測定結果の平均値を用いた．尚，2 次元解析における総タイムステップ数は 4,400 ステップ，3 次元解析においては 5,400 ステップとなっている．

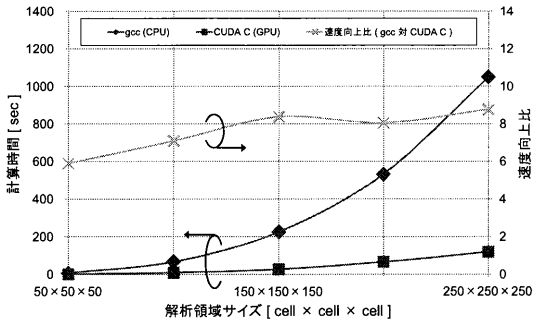


(a) 2 次元解析　　(b) 3 次元解析

図 2　FDTD 計算モデル



(a)　2 次元解析



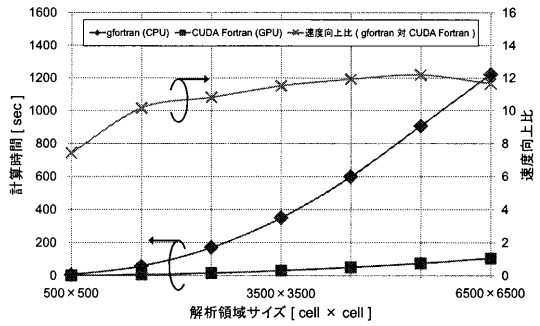
(b)　3 次元解析

図 3　C 言語を用いた GPGPU-FDTD 計算と CPU-FDTD 計算の速度比較

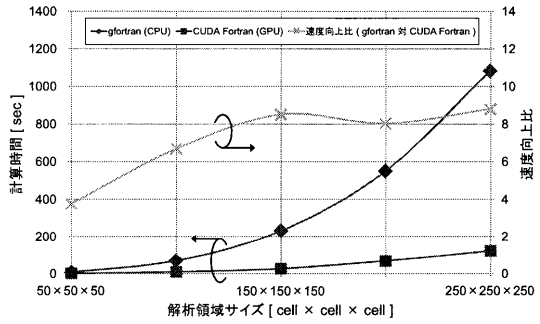
5.1. GPGPU 及び CPU 計算の演算速度比較

図 3 及び 4 にそれぞれ C 及び Fortran 言語を用いた GPGPU-FDTD 計算と CPU-FDTD 計算の速度比較を示す．尚，速度向上比は CPU 計算時間を GPGPU 計算時間で割る事により求めている．

GPGPU 計算では，何れのプログラミング言語においても解析領域が大きくなれば CPU 計算に対してほぼ一定の速度向上比が得られる事が示されている．この時，速度向上比は，2 次元解析では約 12 倍，3 次元解析では約 8 倍を示した．しかしながら，解析領域が比較的小さい場合には速度向上比が小さくなる事が示されている．これは，解析領域が小さい場合，CPU 内のキャッシュに解析データが収まるために，通常に比べて CPU 計算が高速になったためと考えられる．



(a)　2 次元解析



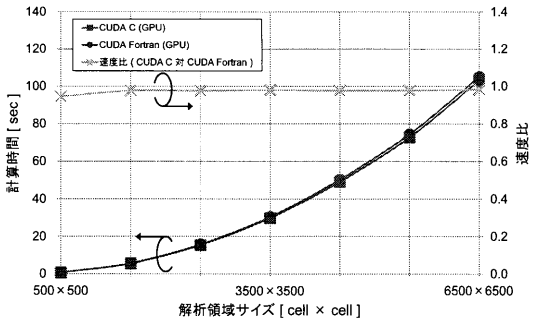
(b)　3 次元解析

図 4　Fortran 言語を用いた GPGPU-FDTD 計算と CPU-FDTD 計算の速度比較

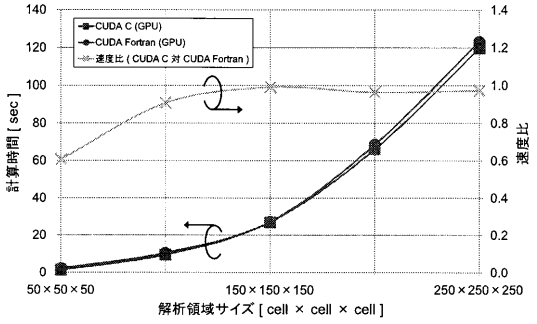
5.2. CUDA C 及び CUDA Fortran を用いた GPGPU 計算の演算速度比較

図 5 に CUDA C 及び CUDA Fortran を用いた GPGPU-FDTD 計算の演算速度比較を示す．尚，速度比は CUDA C の計算時間を CUDA Fortran の値で割る事により求めている．

CUDA C 及び CUDA Fortran の何れにおいても解析領域が大きい場合，ほぼ同等の演算性能を有する事が示されている．しかしながら，解析領域が小さい場合は，CUDA Fortran の演算性能が若干低下する事が示されている．



(a) 2 次元 FDTD 解析



(b) 3 次元 FDTD 解析

図 5 CUDA C 及び CUDA Fortran を用いた GPGPU-FDTD 計算の速度比較

6. むすび

本研究では, GPGPU-FDTD 計算において CUDA C 及び CUDA Fortran が演算速度に与える影響を定量的に調べた. その結果, CUDA C 及び CUDA Fortran は同等の演算性能を有している事が示された. また, C 及び Fortran 言語の何れにおいても CPU 計算に対して GPGPU 計算の速度が向上する事が示された. この時, 2 次元解析では約 12 倍, 3 次元解析では約 8 倍の速度向上比となる事が示されている.

今後は, CUDA Fortran における GPU 並列化指示文及び複数枚の GPU デバイスを用いた演算性能の評価を行う予定である.

謝 辞

本研究の一部は独立行政法人日本学術振興会の科研費(23560433)の助成を得たものである. ここに謝意を表する.

文 献

[1] 打矢, 柏, “HPF 及び MPI を用いた FDTD 並列スーパーコンピューティング,” 信学技報, MST2001-6, pp. 39-46, Sep. 2001.

[2] S. Piotr, D. Adam, and M. Michal, “How to render FDTD computation more effective using a graphics accelerator,” IEEE Trans. Mag, vol. 45, no. 3, pp. 1324-1327, Mar. 2009.

[3] 高田, 他, “GPU による共有メモリを効率的に用いた FDTD 法差分計算の高速化,” FIT2009, C-103, pp. 457-462, Sep. 2009.

[4] 長岡, 渡辺, “GPU を用いた生体電磁ドシメトリの 3 次元 FDTD 計算,” 信学総大, B-4-52, Mar. 2010.

[5] 園田, 佐藤, “コンパイラディレクティブを用いた FDTD 法の GPU 実装,” 信学総大, B-1-34, Mar. 2011.

[6] 河田, 大久保, 土屋, “GPU 計算を用いた時間領域電磁界数値解析の高速化性能に関する比較,” 信学総大, CS-1-8, Mar. 2011.

[7] 小関, 園田, 昆, 佐藤, “FDTD 法を用いた GPR シミュレーションの GPU 実装による高速化,” 信学技報, EMCJ2011-50, pp. 37-42, July 2011.

[8] 河田, 大久保, 土屋, “マルチ GPU 並列計算による電磁界数値解析の高速化に関する検討,” 信学論 (B), vol. J94-B, No. 3, pp. 480-483, Mar. 2011.

[9] B. Meyer, C. Plessl, and J. Forstner, “Transformation of scientific algorithms to parallel computing code: single GPU and MPI multi gpu backends with subdomain support,” SAAHPC 2011, pp. 60-63, July 2011.

[10] TOP500 Supercomputer Sites, http://www.top500.org/

[11] NVIDIA CUDA C programming guide version 4.2, NVIDIA, 2012.

[12] CUDA Fortran Programming guide and reference release 2012, The Portland Group, 2012.

[13] A. Taflove, S. C. Hagness, Computational electrodynamics: the finite-difference time-domain method, 3rd edition, Artech House, 2005.