B-013

GPGPU における条件分岐処理を含む数値計算の高速化についての検討

Study on the speed of numerical calculation including the conditional branch processing in GPGPU

臼倉 圭亮 * 増田 信之 * 山形 健太 * 山本 未来呂 *
Keisuke Usukura Nobuyuki Masuda Kenta Yamagata Mikuro Yamamoto

1. まえがき

近年では，様々な分野において数値シミュレーションや数値計算が行われており，より大規模かつ高精度な計算が必要とされている．しかし，大規模かつ膨大な計算にはより多くの時間がかかるために高速化が求められる．計算を高速化する一つの手段として，本来では画像処理用に使用される GPU(Graphics Processing Unit) を数値計算に用いる GPGPU(General purpose computing on GPU) が一般的に使用されるようになってきた．

GPGPU は，基本的に演算を並列化することで計算の高速化を図る．そのため，演算の中に条件分岐処理が含まれる場合，計算プロセッサ同士の待ち時間が生じるため，パフォーマンスが落ちてしまう [1]．そのため本研究では，条件分岐処理が多く含まれる数値計算における GPU を用いた高速化について検討を行った．計算題材として，FDTD 法 (Finite Difference Time Domain method) を使用して，高速化についての検討を行った．

具体的には，条件分岐を含む領域と含まない領域とで計算領域を分割し，それぞれを別のカーネル関数として実行させる．結果として，計算量が少ない場合では分割数を少なく，計算量が多い場合では分割数を多くすることで演算が高速となることが分かった．

2. FDTD 法

FDTD 法は，電磁波シミュレーションになどに用いられる電磁界解析法の一つである．マクスウェル方程式を差分化し，時間領域で解くことから，最終的な結果だけでなく，結果に至るまでの電磁界の変化を解析することができる．本稿では，境界条件として Mur の吸収境界条件を用いた二次元 FDTD 法について解く．

2.1 FDTD の計算

以下に二次元 FDTD 法の式を示す．

$$H_x^{n+1/2}(i,j+1/2) = H_x^{n-1/2}(i,j+1/2) - \Delta t/\mu\Delta y\{E_z^n(i,j+1) - E_z^n(i,j)\} \quad (1)$$

$$H_y^{n+1/2}(i+1/2,j) = H_y^{n-1/2}(i+1/2,j) - \Delta t/\mu\Delta x\{E_z^n(i+1,j) - E_z^n(i,j)\} \quad (2)$$

$$E_z^{n+1}(i,j) = E_z^n(i,j) - \Delta t/\epsilon\Delta y\{H_x^{n+1/2}(i,j+1/2) - H_x^{n+1/2}(i,j-1/2)\} + \Delta t/\epsilon\Delta x\{H_y^{n+1/2}(i+1/2,j) - H_y^{n+1/2}(i-1/2,j)\} \quad (3)$$

ここで，n は時間ステップ，E は電場，H は磁場を表す．また，パラメーター ϵ と μ はそれぞれ誘電率と透磁率を表す．Mur の吸収境界条件については別紙に譲る [2]．

2.2 計算手順

FDTD 法のフローチャートを図 1 に示す．

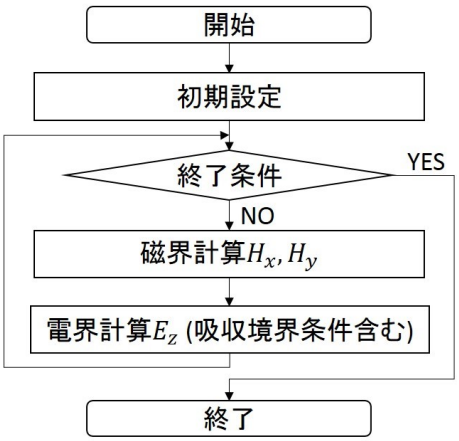

図 1: FDTD 法フローチャート

計算手順は，ある時間ステップにおける磁場と電場の値を GPU で並列計算し，計算結果をグローバルメモリ上に保存させる．ステップを進めて再び並列計算を行い，求めたい時間ステップ分繰り返し計算を行ったら演算を終了する．本稿では，磁界計算部分に条件分岐が発生しないため，吸収境界条件を含む電界計算部分についてのみカーネルの分割を検討した．

3. カーネル分割方法

本稿では，電界計算部分のカーネルを 7 通りの分割方法について比較した．カーネルの分割方法について次の図 2 に示した．

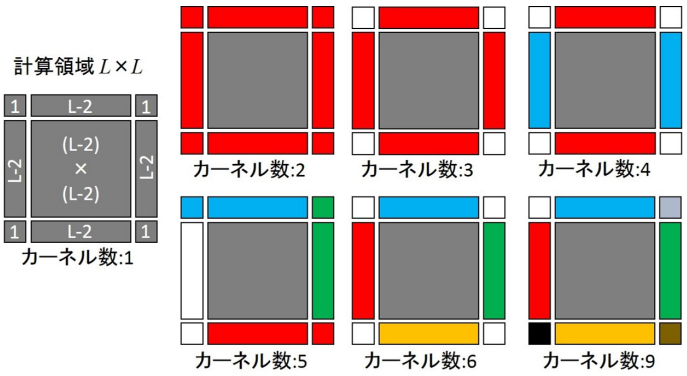

図 2: 分割方法

図 2 は，同じ色で塗られた領域が同一のカーネルとなるように表記してある．また，吸収境界条件に当たる領域は計算量が中央部分に比べて比較的少ない周辺部分が相当する．

*東京理科大学基礎工学部

表 3: 計測結果

計算領域 $[L \times L]$	計算時間 $[\mu s]$						
	kernel1	kernel2	kernel3	kernel4	kernel5	kernel6	kernel9
18×18	**11.385**	13.148	15.331	17.132	24.486	21.057	32.904
34×34	**10.914**	13.394	15.509	17.256	24.929	21.161	33.262
66×66	19.579	**13.998**	16.180	17.902	25.488	21.703	33.892
130×130	60.215	**16.663**	18.826	20.294	27.438	23.692	35.786
258×258	211.303	**32.048**	34.916	35.978	42.831	39.290	51.383
514×514	780.092	**91.118**	95.747	97.045	105.133	100.971	114.511
1026×1026	3069.807	**369.453**	373.347	375.044	384.084	379.484	392.455
2050×2050	12285.420	**1561.667**	1565.313	1570.258	1579.012	1572.752	1587.241
4098×4098	40608.178	**7873.115**	7874.477	7877.436	7890.070	7881.565	7894.810
8194×8194	161339.411	36509.001	**36302.942**	36489.287	36323.567	37190.540	36386.481
16386×16386	756177.895	323355.291	323264.317	323213.601	323440.952	323250.467	**323197.173**

4. 実験結果

4.1 実験環境

ここでは，磁界計算と電界計算を GPU によって行い，電界部分についての計算時間の比較を行う．電界部分におけるカーネルの分割方法は，前章で述べたように 7 通りについて比較する．計算時間の計測には nvprof を用い，CPU-GPU 間のメモリアクセス時間はグローバルメモリを用いるためどの分割方法も等しいとする．カーネル起動時間に関しては今回に限り無視し，純粋な計算時間のみの比較を行った．時間ステップ数を 850，計算領域を $L \times L$，thread 数を x 方向 y 方向共に 16 とし，初期条件として磁界は 0，電界は半分の領域を 0 もう半分を 1 とした．計測時間は 850 ステップの平均を 1 ステップ分として算出した．また，ソフトウェア開発環境を表 1 に示した．また，計算に使用した GPU について概要を表 2 に示した [3].

表 1: ソフトウェア開発環境

CPU	Intel Xeon E5-2697 v2 2.70GHz ×2
メモリ	64 GB (32GB×2)
GPU	NVIDIA GeForce GTX TITAN X
OS	CentOS Linux 7.1.1503
開発環境	CUDA 7.5

表 2: TITAN X 概要

CUDA コア	3072
ベースクロック	1000MHz
メモリ量	12GB

4.2 計測結果と考察

表 3 に各々のカーネル分割時における 1 ステップ分の電界計算にかかる計測時間を示した．表 3 より，計算量が増えるほどカーネルを多く分割する方が計算が高速となっている事がわかる．逆に，計算量がある程度少ない場合ではカーネルを分割しないほうが高速となっていることがわかる．

5. まとめと今後の課題

本稿では，条件分岐処理を含む数値計算における高速化の検討を行った．GPU 上で起動するカーネル関数を分割することで，カーネル関数内における条件分岐処理の影響を減らし，計算時間の低減を行えると仮定した．その結果，計算量が比較的多い場合では分割数を増やすことで計算時間を低減することができた．しかし，計算量が少ない場合ではカーネルを分割しないほうが高速となることがわかった．この原因として，CUDA には Warp という概念があり，32 個のスレッドを 1 つのグループとして並列化する [4]．そのため計算量が少ない場合，カーネルを分割すると境界条件部分の計算量が少ないため，GPU のリソースを活かすことができずパフォーマンスが低下する．

今回，計算量に応じて最適なカーネル数が異なることがわかった．そのため展望として，計算量に応じて最適なカーネル数を選択するといったアルゴリズムの開発を検討する．

謝辞

本研究は JSPS 科研費 25240015，15K00175，の助成を受けたものです．

参考文献

[1] John Cheng, Max Grossman, Ty McKercher, ”CUDA C プロフェッショナルプログラミング”, 株式会社インプレス, 2015

[2] 宇野 亨, ”FDTD 法による電磁界およびアンテナ解析”, コロナ社, 1998

[3] ”NVIDIA Home Page”, http://www.nvidia.co.jp/object/geforce-gtx-titan-x-jp

[4] 伊藤 智義, ”GPU プログラミング入門”, 講談社, 2013