GPU チャレンジ自由課題部門レポート
「GPU による地震波伝播シミュレーション」

岡元太郎* 竹中博士† 中村武史‡

平成 22 年 3 月 15 日

1 概要

我々は、地震波、すなわち地球内部を伝播する弾性波のシミュレーションを GPU によって行うことを試みた。地震波伝播の計算には、標準的に利用されている食い違い格子型の有限差分法（FDTD: finite-difference time domain）を用いた。空間差分精度は 2 次および 4 次精度の両方の場合で計算を行った。時間差分は 2 次精度とした。単一 GPU での計算では、46 GFlops（2 次）から 48 GFlops（4 次）の演算性能が得られた。これはホスト CPU（Opteron 2.4GHz）の約 29 倍（2 次）から約 42 倍（4 次）の性能に相当する。さらに、空間 4 次精度のプログラムでは MPI 並列化によってマルチ GPU に対応した。並列化すると境界処理が加わるために演算性能が低下する。また、分割した場合の総合性能は計算領域の形状や分割の仕方に大きく依存する。4 GPU のもとでの現実的な水平分割の場合には、演算性能は約 54 GFlops であった。これをホスト CPU の 1 コアの性能で単純に割算すると約 48 コアに相当する。これらのことから、地震波伝播シミュレーションにおいてもマルチ GPU 計算が有効であり、従来の CPU よりも計算を高速化できることが確認できたと考える。

2 はじめに

地震波伝播シミュレーションにおいては、地球内部の不均質構造や複雑な地表地形、海底地形（海水層）などの様々な要素を取り入れる必要がある。我々はこれらの要素を取り入れた並列計算手法を開発し、地震に伴う揺れ（強震動）に地形や流体層が及ぼす影響について研究を始めている（Nakamura et al. 2009 ほか）。さらに、地震波伝播を計算する順問題だけではなく、波形データから地球内部構造を推定する「逆問題」の研究を進めていくことも構想している。逆問題では多数の地震と観測点について波形計算を行い、構造摂動に対する偏微分係数を数値的に求める必要があるため、地震波伝播シミュレーションの計算量は膨大なものとなる。このような膨大な計算を処理するために、我々は非常に高い演算性能を持つ GPU を利用することに着目した。そこで本レポートでは、地震波伝播シミュレーションの基本的な部分を GPU で実現し、通常の CPU の場合よりも計算を高速化することを目的とする。

*東京工業大学
†九州大学
‡JAMSTEC

3 支配方程式と計算格子ほか

本レポートでは地震波伝播シミュレーションにおいて標準的な方法として広く利用されている有限差分法（FDTD: finite-difference time domain）を用いる。この有限差分法では媒質の粒子速度（$v_i (i = x, y, z)$）と応力（τ_{ij}）とを変数として、図 1 に示す食い違い格子を用いて計算領域を離散化する。この方法では変位のみを用いた場合に比べて変数が増加するが、不均質媒質の場合でも数値計算上の安定性が良いなどの実用上の利点を多く持っている。なお、空間差分精度は 2 次および 4 次精度の両方の場合で計算を行った。時間差分は 2 次精度とした。

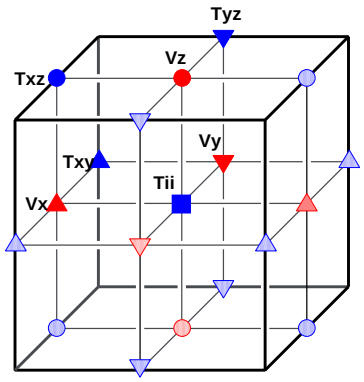

図 1 格子点での変数配置

本レポートでは完全弾性体の場合のプログラムを作成する。その場合の支配方程式は次の組になる。f_i は力源、ρ は密度、λ、μ は弾性定数で μ が剛性率である。この場合、1 単位セルあたりのデータは粒子速度・応力が計 9 個、媒質の定数が 3 個（τ_{ii} と同じ格子点に置く）、合わせて 12 変数となる。

$$\rho \frac{\partial v_i}{\partial t} = \frac{\partial \tau_{xi}}{\partial x} + \frac{\partial \tau_{yi}}{\partial y} + \frac{\partial \tau_{zi}}{\partial z} + f_i$$

$$\frac{\partial \tau_{ij}}{\partial t} = \lambda \delta_{ij} \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \right) + \mu \left(\frac{\partial v_j}{\partial x_i} + \frac{\partial v_j}{\partial x_j} \right)$$

食い違い格子を用いるため、粒子速度変数と応力変数の時刻が時間刻みの半分だけずれることになる。計算上は、粒子速度と応力を片方ずつ更新していくことになる。

なお、以下の計算例では東京工業大学学術情報センターの計算機（GPU: Tesla S1070、ホスト CPU: デュアルコア Opteron 2.4GHz、PCI-Express 1.0x8）を利用させていただいた。

4　単一 GPU 計算（空間 2 次精度）

この節では、もっとも簡単な実装として 3 次元の各方向に周期境界を持つ、空間 2 次精度・時間 2 次精度のプログラムを作成した。流体力学計算の場合（青木、2009）を参考にして、計算領域を小領域に分割する。そして小領域の変数を共有メモリに置いて計算の効率化を図るのであるが、地震波計算の場合には変数が多くなるため多くの格子点の変数を共有メモリに置くことが難しくなる。

そこで、共有メモリは 2 次元格子として扱い、その面に垂直な方向ではレジスタを用いてデータの再利用を図る。図 2 には Z 方向に粒子速度の積分を進めていく過程でのメモリ配置の概念図を示す。赤色で示した変数は 2 次元 (XY) の共有メモリに格納されている。そして、グローバルメモリからレジスタに転送された変数（図 2 の例では左側の Txz）は、次の Z のレベル（図 2 の右側）でレジスタから共有メモリにコピーして、グローバルメモリへのアクセス回数を減らす。

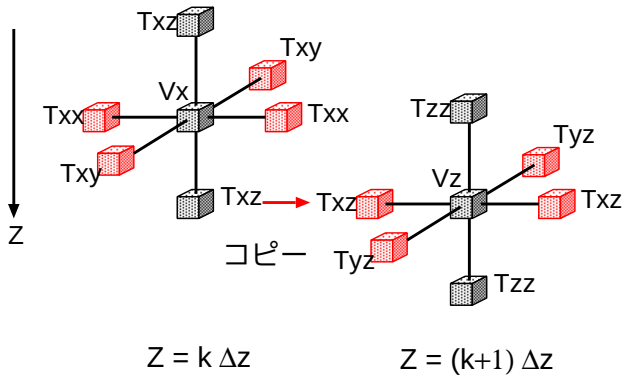

図 2

また、粒子速度・応力の積分では各変数の格子点での密度や剛性率が必要になる。これらについてはデータ転送量を減らすために、単位セル中心（τ_{ii} の格子点）に置かれた定数値を使って毎回計算する。不均質媒質での計算を精度良く行うために、密度は隣接するセルでの算術平均、剛性率は幾何平均とする（Takenaka et al., 2009）。

このような GPU プログラムを CUDA 開発環境を用いて作成した。また、同様の計算を行う通常のプログラム（PGI-Fortran でコンパイル）も作成して比較した。演算性能について図 3 と図 4 に示す。なお共有メモリに置いた変数の配列サイズは 18×18（袖領域を含む）で固定した。演算性能は、支配方程式に関わる演算回数と媒質定数の算出に関わる演算回数の合計をもとにして計算した。

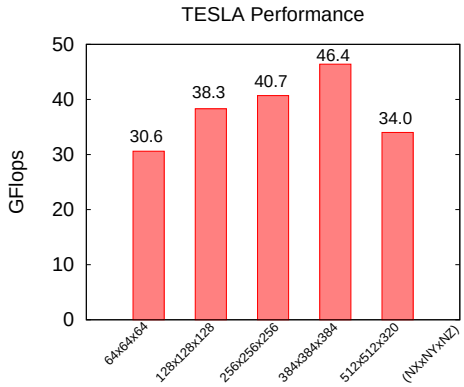

図 3　Tesla S1070 の 1 GPU による演算性能。横軸は計算領域の格子点数（単位セルの個数）を示す。数値は X 方向 ×Y 方向 ×Z 方向の順であり、以下でも同様に示す。

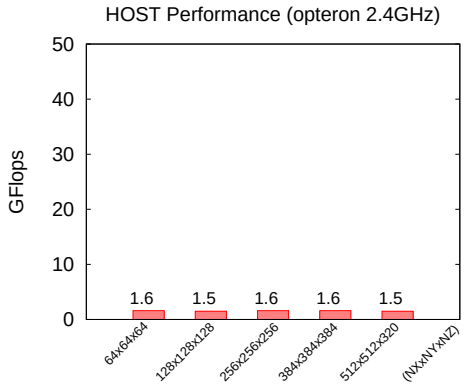

図 4　ホスト側 CPU の 1 コアによる演算性能

GPU の場合には計算領域サイズによって性能が変化する。最高では約 46 GFlops の性能を得た。これはホスト側 CPU の 1 コア性能の約 30 倍になる。また計算時間は、計算領域サイズが 384×384×384、時間 1000 ステップの場合[1]に、Tesla S1070 では 135.4 秒であった。なお、ホスト側 CPU では計算領域サイズを変えても性能があまり変化しない。これは、メモリ転送速度が性能を律速しているためかもしれない。

この結果から、メモリアクセス頻度が演算の頻度よりも大きくなる地震波計算においても、GPU は計算を高速化することが可能であることが確認された。

5　マルチ GPU 計算（空間 4 次精度）

本レポートで利用した GPU 単体のメモリ容量は 4 G バイトであるため、地震波伝播シミュレーションのように格子を扱う場合にあまり大きなサイズの計算はできない。そのため、実用計算を目指す場合には複数の GPU に計算領

[1]典型的な場合として、P 波速度（縦波速度）5.8 km/s、格子間隔 200 m、時間刻み 0.005 s、とすると、1000 ステップは 5 秒間に相当する。実際の計算では、領域サイズも時間ステップ数ももっと大きいのが普通である。

域を分割する並列計算が不可避である。そこで、本レポートでも複数の GPU による計算を試みた。

なお以下では実用計算に近づけるために、空間 4 次精度を採用し、自由表面（地表面）や側方と底の吸収境界条件（Cerjan et al. 1985）を実装した。時間積分は 2 次精度である。共有メモリに割り当てる配列サイズは 20×20 （袖領域を含む）で固定した。

5.1 1 次元分割

1 次元分割は側方境界の処理が簡単であるためしばしば用いられる。本レポートでは、水平な複数の面（Z 軸に垂直な面）で領域を分割する。これは、図 5 の袖領域のメモリが連続になるような分割である。

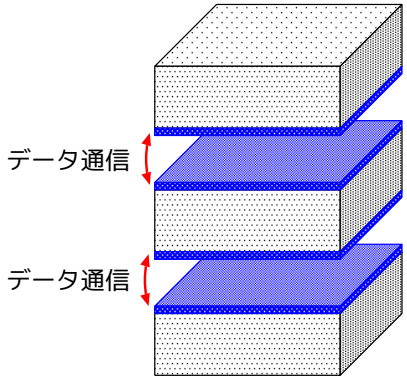

図 5 1 次元分割の概念図。青い部分（袖領域）のデータを隣接するプロセスがやり取りする。

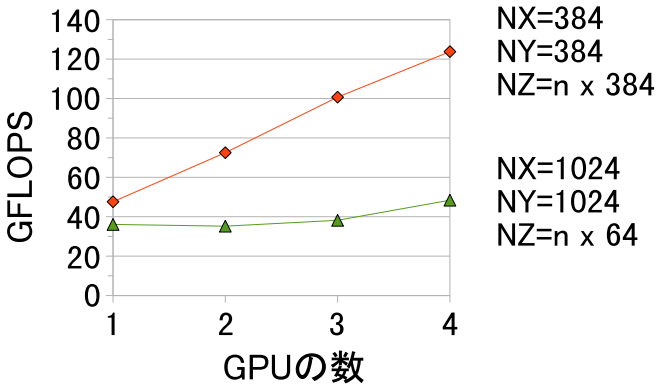

図 6 1 次元分割のときの演算性能。n を GPU 数としたとき、全領域サイズは赤グラフが 384×384× (384 n)、緑グラフが 1024×1024×(64 n) である。

図 6 に、全計算領域サイズが GPU の個数に比例して増加する場合における演算性能の一例を示す。1GPU の場合の性能はこの空間 4 次精度プログラムでは最大で 48 GFlops 程度になる。その場合、4 個の GPU で最大約 2.6 倍の性能向上が見られた（図 6 の赤グラフ）。ただしこの例では 1 つの方向（深さ方向）に厚い計算領域となるため、現実的な計算ではない。より現実的に地表面を広く取って水平方向に長い計算領域を仮定すると、演算性能はかなり低下する（図 6 の緑グラフ）。性能向上（赤グラフの場合で約 2.6 倍）があまり大きくないのは、ホストのバス規格が PCI-Express 1.0x8 であることも関係しているかもしれない。

また、ホスト CPU でも空間 4 次精度のプログラムを作成し（ただし吸収境界条件は省略した）、演算性能を調べたところ 1 コアで約 1.1 GFlops であった（領域サイズが 384×384×384 の場合）。1GPU 計算では最大で約 48 GFlops であったので、最大でホスト CPU の約 42 倍の演算性能であると言える。

なお、図 6 の赤グラフについて、1GPU の場合の計算時間は 204.5 秒であった（領域サイズは 384×384×384、時間ステップ数は 1000）。

6 3 次元分割

1 次元分割は比較的に簡単であるが、上で見たように現実的な計算領域サイズに対しては性能が向上しにくい面がある。そのため、水平方向にも計算領域を分割した 3 次元分割が、大規模計算のためには不可欠である（図 7）。

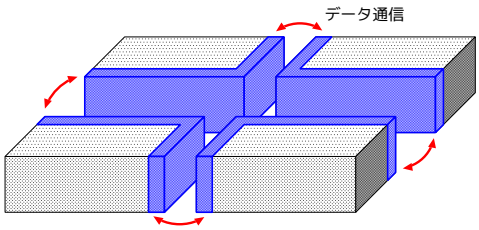

図 7 水平方向への分割の概念図

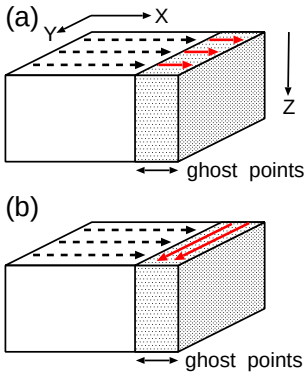

図 8 袖領域（Ghost points）のメモリ並びの概念図。矢印がメモリ並びの方向を示す。(a) 計算領域の延長としてメモリを確保した場合。(b) 計算領域とは別の配列として袖領域を確保した場合。

3 次元分割では袖領域のメモリをどのように確保するかが重要になる。計算領域の配列サイズを 1（2 次精度）ないし 2（4 次精度）だけ延長して袖領域に割り当てる方法がしばしば採用される。このとき向きによっては袖領域の内部で不連続なメモリ領域に変数が細切れに割り当てられる（図 8a）。そのため GPU からホストに袖領域を転送し

ようとすると、細切れのメモリ領域を莫大な回数のメモリ転送関数の呼び出しで転送する必要が出てくる。しかし、これは通信に要する時間を大きく増加させてしまうため全く実用にならない。

そのため、袖領域の変数を一度のメモリ転送関数で転送できるように、計算領域とは別個の連続なメモリ領域を袖領域に割り当てることが実用計算のためには必要となる（図 8b）。そこで、側方（本レポートでは X 方向と Y 方向）には袖領域を独立の領域として確保するような、3 次元分割対応のプログラムを作成した。袖領域と計算領域の間でのデータのやり取りはすべて GPU 内部で行い、ホスト側では袖領域を他のプロセスと交換するだけに限定している。

この並列化プログラムの単一 GPU での演算性能は約 24 GFlops（384×384×384）であり、境界処理のために性能が低下していることがわかる。

マルチ GPU 計算の場合、図 7 に示すような水平方向の 4 分割のもとで計算領域サイズが 768×768×384 の場合に 51.8 GFlops、1024×1024×256 の場合に 53.6 GFlops という性能が得られた。境界処理のために性能が低下するものの、ホスト CPU と比べた場合におよそ数十倍に相当する高速化を達成できたと考えられる（1 コア性能で単純換算すると約 48 倍となる）。

7　議論とまとめ

3 次元分割は大規模な実用シミュレーションを行う上で不可避であると言える。本レポートでは、3 次元分割によって性能が低下するが、それでも従来型 CPU に比べて高速に計算を実行できることを確認できた。しかし、境界処理のために演算性能が低下する点は依然として重要な課題であり、今後の更なる工夫が必要となる。

現行の GPU（NVIDIA）では、今回の地震波伝播シミュレーションのように、変数の多い格子計算では共有メモリやレジスタ数の制約が強くなるようである。実際、空間 4 次精度のプログラムでは一部の変数がローカルメモリに割り当てられていた。本レポートでは共有メモリ上のブロックサイズを固定していたが、これらの制約を低減するために共有メモリの利用にさらに工夫（メモリを節約する手法の導入、例えば 青木・額田 (2009)）が必要である。また、新製品（Fermi）では共有メモリサイズが大きくなることから、プログラム上の制約がある程度低減されると期待できる。

本レポートでは、3 次元地震波伝播シミュレーションの問題について以下のことを確認できた。

- 単一 GPU の場合、空間 2 次精度で約 46 GFlops、空間 4 次精度では約 48 GFlops の性能を得た。これらはホスト側 CPU（Opteron 2.4 GHz）の 1 コア性能の約 29 倍から約 42 倍である。
- MPI 並列化を施したプログラムで、袖領域の配列メモリが連続になるように 1 次元分割した場合、4 個の GPU で最大で 124 GFlops の演算性能が得られた。ただし演算性能は領域の形状やサイズに強く依存する。
- 実用的な 3 次元分割プログラムでは、袖領域の処理のために性能が半分程度に低下する。それでもなお従来型 CPU に比べて数十倍程度に高速な計算を行えることを確認できた。

8　謝辞

東京工業大学学術情報センターで行われた GPU コンピューティングに関する講習会は大変参考になりました。本研究の一部に科学研究費補助金（課題番号：19540448）を利用しました。

9　参考文献

青木尊之、フル GPU による CFD アプリケーション、情報処理、**50**、No.2、107–115、2009.

青木尊之、額田 彰、はじめての CUDA プログラミング、工学社、2009.

Cerjan, C., D. Kosloff, R. Kosloff, and M. Reshef, A nonreflecting boundary conditions for discrete acoustic and elastic wave equations, *Geophysics*, **50**, 705–708, 1985.

Nakamura, T., H. Takenaka, T. Okamoto, and Y. Kaneda, Finite-difference simulation of strong motion from a sub-oceanic earthquake: modeling effects of land and ocean-bottom topographies, American Geophysical Union, Fall Meeting, S43B-1981, 2009.

Takenaka, H., T. Nakamura, T. Okamoto, and Y. Kaneda, A unified approach implementing land and ocean-bottom topographies in the staggered-grid finite-difference method for seismic wave modeling, Proceedings of the 9th SEGJ International Symposium, CD-ROM Paper No.37, 2009.