

時間依存シュレディンガー方程式に基づく 量子シミュレーションのGPUによる高速化 (豊橋技術科学大学院) 増田拓弥, 関野秀男 High Performance Simulation of TDSE using GPU (TOYOHASHI UNIVERSITY OF TECHNOLOGY)

Takuya Masuda, Sekino Hideo

【研究背景・目的】

原子や分子の動的物性の予測を行う場合、量子力学における基本方程式であるシュレディンガー方程式を解くことは必須である。しかしながら、分子の電子状態に対する予測の精密な結果を得るためには膨大な計算領域と解析時間を有するため、高速演算のアルゴリズム開発と並列環境の実装は重要である。

並列演算に特化した演算装置として GPU(Graphics Processing Unit)がある。GPUは、主に画像処理用の演算装置であるが、近年、その高い演算処理能力から、画像以外の一般的な用途に利用する GPGPU(General Purpose GPU)という概念が注目されている。

そこで、本研究は NVIDIA 社製 GPU と C 言語の統合開発環境 CUDA(Compute Unified device Architecture)を用いることによる時間依存シュレディンガー方程式に基づく量子シミュレーションの高速化を行った。

【シュレディンガー方程式の差分化】

量子力学の基本方程式として、2次元シュレディンガー方程式があり、次のように記述される。

$$i\hbar \frac{\partial}{\partial t} \psi(x, y, t) = -\frac{\hbar^2}{2m} \left\{ \frac{\partial^2}{\partial x^2} \psi(x, y, t) + \frac{\partial^2}{\partial y^2} \psi(x, y, t) \right\} \dots \text{式(1)}$$

そして、(1)を3次元に拡張すると、次のように記述することができる。

$$i\hbar \frac{\partial}{\partial t} \psi(x, y, z, t) = -\frac{\hbar^2}{2m} \left\{ \frac{\partial^2}{\partial x^2} \psi(x, y, z, t) + \frac{\partial^2}{\partial y^2} \psi(x, y, z, t) + \frac{\partial^2}{\partial z^2} \psi(x, y, z, t) \right\} \dots \text{式(2)}$$

式(1)(2)のシュレディンガー方程式にテイラー展開による差分近似を適用して解くことで、量子の動的物性の計算におけるGPUによる高速化の影響を比較できる。

2次元シュレディンガー方程式による量子の動的物性のモデル化に際し、2次元断面は $n \times n$ のグリッド上に格子点を間隔 d で配置されたメッシュで表現し、各格子点における波動関数の値を式(1)を用いて推定する。 $\psi(x, y, t)$ は量子の波動関数であり、 x, y は x, y 方向の離散化された空間座標、 t は時間間隔 t の離散化された時間座標である。導関数の近似にテイラー展開による2階差分近似を用いると、 x, y に関する2階導関数を得ることができ、中心差分近似を用いると t に関する1階導関数を得ることができる。これらの導関数を式(1)の微分項に代入し、整理すると次のように書くことができる。

$$\psi(x, y, t + \Delta t) = \psi(x, y, t - \Delta t) + \frac{i\hbar}{m} \left\{ \frac{\psi(x + \Delta x, y, t) + \psi(x - \Delta x, y, t) - 2\psi(x, y, t)}{\Delta x^2} + \frac{\psi(x, y + \Delta y, t) + \psi(x, y - \Delta y, t) - 2\psi(x, y, t)}{\Delta y^2} \right\} \Delta t \dots \text{式(3)}$$

この方程式は、前回・現在のその格子点における波動関数値と、現在の隣接格子点における波動関数値をもとに、時間 t だけ経過した未来の波動関数値を推定するものである。

【CUDA を用いたプログラミング】

GPUは、数万スレッドにも及ぶCPUより圧倒的に多数のマルチ・スレッド方式で処理することによって、高い並列演算処理を実現することができる。

このGPUを動作させるために必要なものが、C言語の統合開発環境CUDAである。プログラム構成は、CPU側で動き、配列の確保などを行うホストコードと、GPUに処理させたい内容を記述するデバイスコードで成り立っている。メモリ構造には、高速アクセスが可能なメモリであるシェアードメモリと、大容量であるがアクセスが低速なグローバルメモリがある。また、グリッド、ブロックという階層を用いることで、膨大なスレッド数を管理している。つまり、GPUの性能を發揮させるためには、使用するメモリの特性とスレッド階層構造を考慮したプログラミングが求められる。

本研究では、GPUを使用したプログラムとして、グローバルメモリのみを使用したプログラム、グローバルメモリとシェアードメモリを併用したプログラムの2つを作成した。1格子点当たり1スレッドで計算するようにし、その中で時間間隔 t 毎にデバイスコードに記述した式(3)の処理を時間発展させることで、時間依存シュレディンガー方程式を解いた。

【検証方法】

CPU, GPU(グローバルメモリ使用), GPU(シェアードメモリ使用)によって時間依存シュレディンガー方程式を計算するプログラムを用いて、スレッド数 $128 \times 128, 256 \times 256, 512 \times 512, 1024 \times 1024, 2048 \times 2048, 4096 \times 4096$ における各プログラムの計算時間を計測する。なお計測は、それぞれ3回ずつ行い、その平均値をとった。

【結果】

スレッド数毎の各プログラムにおける計算時間の計測結果を表1, CPUを1といただいた場合の計算時間比を図1に示す。なお、図1の縦軸は計算時間比であり、[CPU計測時間/GPU計測時間]とした。

表1 各プログラムにおける計算時間(単位: ms)

	128*128	256*256	512*512	1024*1024	2048*2048	4096*4096
CPU	533.166	2172.627	8830.128	36338.460	145875.988	630161.670
GPU (global memory)	7.564	25.372	89.906	379.253	1409.449	5646.680
GPU (shared memory)	6.583	22.049	76.186	299.150	1136.246	4828.202

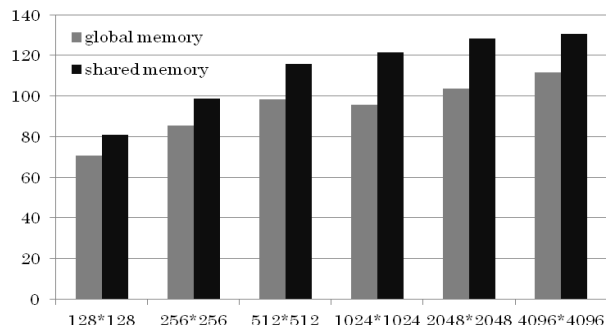


図1 CPUに対するGPUの計算時間比

【考察】

表1より、計算速度はGPU(シェアードメモリ) >> GPU(グローバルメモリ) >> CPUとなった。つまり、シュレディンガー方程式の計算の高速化において、GPUはともにCPUより有効で、メモリとしてシェアードメモリを使用することで最も高速化されることがわかった。

図1からは、スレッド数の拡大に伴って、計算の高速化がなされていることがわかる。つまり、GPUの性能は、より膨大なスレッド数での計算において最大限に活かされる。実際に、2次元シュレディンガー方程式と3次元シュレディンガー方程式CPUに対するGPUの計算時間比を比較した結果、3次元シュレディンガー方程式が、より高速化率が高いことが確認できた。

ポスターでは3次元シュレディンガー方程式の計算結果および、2次元シュレディンガー方程式の計算結果との比較も行い、次元の差異による高速化に対する影響を考察する。

【参考文献】

[1] 青木尊之, 額田彰. はじめてのCUDAプログラミング. 工学社, 2009