1P133

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(GraphicsProcessingUnit)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(GeneralPurposeGPU)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 次元のシュレディンガー方程式があり，次のように記述される．

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

また，流体の波動現象を記述するものとして，次の波動方程式がある．

$$\frac{\partial^2}{\partial t^2}z(x,y,t)=c^2\left\{\frac{\partial^2}{\partial x^2}z(x,y,t)+\frac{\partial^2}{\partial y^2}z(x,y,t)\right\}\cdots\text{式(2)}$$

式(2)にテイラー展開を適用して解くと，現在と過去の変位量を用いて未来の変位量を推定する次の方程式を導くことができる．

$$z(x,y,t+1)=\frac{4-8c^2t^2/d^2}{\mu t+2}z(x,y,t)+\frac{\mu t-2}{\mu t+2}z(x,y,t-1)+\frac{2c^2t^2/d^2}{\mu t+2}\{z(x+1,y,t)+z(x-1,y,t)+z(x,y+1,t)+z(x,y-1,t)\}\cdots\text{式(3)}$$

式(1)(2)の比較から，シュレディンガー方程式は流体の波動方程式との類似性が高いことがわかる．つまり，流体理論で適用できることは量子理論でも同様に適用できることが考えられるため，式(1)を式(3)の形式に解くことで，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．

量子の動的物性のモデル化に際し，2 次元断面は n×n のグリッド上に格子点を横間隔 Δx, 縦間隔 Δy 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は式(1)を用いて推定する．ψ(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．

$$\psi(x,y,t+\Delta t)=\psi(x,y,t-\Delta t)+\frac{i\hbar}{m}\left\{\frac{\psi(x+\Delta x,y,t)+\psi(x-\Delta x,y,t)-2\psi(x,y,t)}{\Delta x^2}+\frac{\psi(x,y+\Delta y,t)+\psi(x,y-\Delta y,t)-2\psi(x,y,t)}{\Delta y^2}\right\}\Delta t\cdots\text{式(4)}$$

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 より圧倒的に多い数のマルチ・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 の性能を発揮させるためには，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．グリッド内ブロック数，ブロック内スレッド数はともに 2 次元配置とした．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，ブロック内のスレッド数を 16×16 で統一した GPU(グローバルメモリ使用)，GPU(シェアードメモリ使用)によってシュレディンガー方程式を計算する 3 つのプログラムを用いて，スレッド数 128×128，256×256，512×512，1024×1024，2048×2048，4096×4096 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 とお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	1206.70	5014.20	20691.59	85877.56	347234.73	1585193.70
GPU(global memory)	50.26	147.15	509.78	1987.27	7478.69	28828.77
GPU(shared memory)	31.88	88.01	345.38	1357.17	4913.50	18697.55

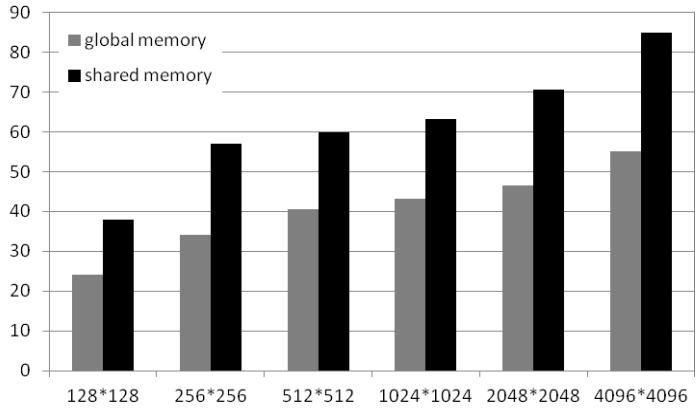

図 1　CPU に対する GPU の計算時間比

【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．

また，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．

シュレディンガー方程式と流体方程式の GPU による計算結果の比較も行い，複素数が高速化にどのような影響を与えているか考察する．

【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