GPU-FFTによる平面波基底第一原理電子状態計算の高速化

GPU-FFT による平面波基底第一原理電子状態計算の高速化
GPU-FFT Based Acceleration of Planewave First Principles Calculation

伴野 秀和[A)]、青木優[B)]、圓谷和雄*[,C)]
Hidekazu TOMONO[A)]、Masaru AOKI[B)]、Kazuo TSUMURAYA*[,C)]

Received : July 31,2009　　Accepted : September 30,2009

Synopsis: The saturation of the acceleration using the silicon devices has required the parallel computing using multiple CPUs, which has been a *de facto* standard of the high-performance computing. To further accelerate the calculations we propose an effective e use of GPU, which have been　graphic cards to output and calculate data for graphic displays. The GPU is faster in operation than CPU in the field of FFT calculations. We, for the first time, implement the GPU-FFT routines into a first principles planewave code, in which the hot spot is the FFT routines. The overall computational time reduces to 15% of the CPU with　FFTW routine　which has been one of the fastest　FFT routines.

Keywords : GPU, accelerator, first principles, planewane method

I　序論

1．第一原理電子状態計算

第一原理電子状態計算とは、経験的パラメータをもちいずに結晶あるいは分子の電子状態を計算する手法である（Payne *et al.* 1992）。これをもちいて電子状態から物質の物性値を予測することのみならず、物質設計が可能である。しかし、この計算方法で扱える原子の数は、現在 100 個から 1000 個程度までである。通常、周期境界条件を適用し、結晶系へ適用している。そこで、計算規模拡大のための手法開発が各方面で行なわれている（Goedecker 1999, Bowler 2008, 寺倉 2009）。

第一原理電子状態計算では、最終的には、コーン・シャム方程式とよばれる偏微分方程式を解く（Hohenberg and Kohn 1964、Kohn and Sham 1965）。このとき電子の波動関数と電荷密度波を平面波で展開しスペクトル空間で解くのが平面波基底第一原理計算法である。ところが、電子相関関数は実空間で定義されているので自己無撞着場であるコーン・シャム方程式の解を求めるには、FFT（Fast Fourier Transform：高速フーリエ変換）により、何度も実空間とスペクトル空間を往復する必要がある。

実際に数値計算する上で平面波展開するときには、ある有限の展開項数で打ち切らなければならない。それを決めるのがカットオフエネルギーである。波動関数を色々な平面波の線形結合で表現するとき、より複雑な形の波動関数を表現するためには短い波長、つまり大きい逆格子成分（この平方がエネルギーに相当）の平面波を必要とする。

ソースコードにして数十万行に及ぶ第一原理計算プログラムの中で、FFT 時間の占める割合は半分以上でありその割合はこのカットオフエネルギーに依存する。高い角運動量成分をもつ原子群を扱うには大きなカットオフエネルギー値をもちいる必要がある。よって、平面波基底第一原理計算法をもちいて大規模な系を解析するには、この FFT 部の高速化が必要である。本研究では、既存のソースコードの FFT 部を CPU 処理から GPU 処理へ置換することによる拘束化を試み、大規模第一原理計算への GPU 適用が可能であること

A) 明治大学大学院理工学研究科機械工学専攻　Department of Science and Technology, Meiji University, Kawasaki, Kanagawa 214-8571, Japan
B) 静岡産業大学経営学部　School of Management, Shizuoka Sangyo University, Iwata, Shizuoka 438-0043, Japan
C) 明治大学理工学部機械情報工学科　School of Science and technology, Meiji University, Kawasaki, Kanagawa 214-8571, Japan
*) E-mail: abinitio@isc.meiji.ac.jp

を示す。GPU は強力なメモリバンド幅を持っているので、これによる FFT 計算は CPU (Central Processing Unit：中央処理装置) で行なうよりも大幅に高速になる (Nukada *et al.* 2008)。

2．GPU

GPU (Graphics Processing Units) とは、パーソナルコンピュータあるいは、ワークステーション、家庭用テレビゲーム機などに使われている画像処理専用デバイスである。画像出力と、その準備のための演算を担っている。当初は 3D ゲームあるいは映画などの動きのある画像を高速に処理するために開発されたデバイスであり、大量生産かつ一般消費者向け商品である。狭義には、グラフィックカードに搭載されている演算装置のことを指すが、ここでは広義にグラフィックカードそのものを指すことにする。

GPU の演算装置は、４つの特徴を持っている。①浮動小数点専用の演算装置であり、②SIMD (Single Instruction Many Data：シムド) 型演算装置として、一度の命令で一度にたくさんのデータを処理することができる。③画像処理で使用する三角関数演算あるいは行列演算を高速に行なうことを得意とした演算装置である。④レジスタ、共有メモリ、グローバルメモリと、３種類の専用メモリを持つことである。通常の CPU (Central Processing Unit) システムでは 2 Gbit/s から 12.8 Gbit/s 程度のメモリバンド幅であるが、この GPU システムのメモリは 10 Gbit/s から 100 Gbit/s 以上のパフォーマンスをもつ。これら４つの特徴は、GPU がゲームなどの娯楽のみならず科学技術計算にも転用可能であることを示している。

3．異種マルチコア構造

これまでの計算機では、各計算機に１個の CPU が組み込まれていた。一般的には、この CPU の発展により計算速度の向上と、計算規模の拡大が進んできた。CPU の性能向上に限界が見え始めると、各計算機に複数の CPU を積み、さらには、一つの LSI のなかに複数個の CPU を搭載する　というマルチコアの方向に進んできた。科学技術用計算機でいうと、スカラー計算機、ベクトル計算機、並列計算機、マルチコア並列計算機である。これをさらに推し進めて，異種マルチコア計算機も実用化されている。例えば、理化学研究所、東京大学などで開発されてきた GRAPE 型専用計算機では、重力多体問題計算の大部分を占める重力相互作用の計算を、専用パイプラインを組み込んだ素子で高速に処理する (Makino 1998)。

4．GPGPU

異種マルチコア構造をもつ計算機として、GPU を画像出力以外の用途にも利用しようという考えが GPGPU （General-Purpose computation on Graphics Processing Units) である (Harris, 飯高 2007 - 2008)。シリコンをもちいる CPU から、より電子の易動度の高いデバイスへの移行への立ち遅れは CPU 並列による計算速度の向上で補われてきた。それによる速度向上の限界限界を克服する方法の一つとしての GPU による高速化が可能である。この高速化法は HPC (High Performance Computing：高性能計算) 市場の新たな潮流となっている。これは、GPU は CPU と比べて安価、かつ高演算能力、メモリバンド幅の高速化などのメリットがあり、さらにメモリ容量の向上により CPU と同規模の計算が可能になったことに起因している。(NVIDIA 2008)。

しかし、CPU から GPU への移行は、同時にディメリットも生じる。第一は使用言語が制限される点である。多くの科学計算用アプリケーションは Fortran 言語により書かれているが、Fortran 言語では、直接的に GPU を操作することができない。第二に、大部分の GPU のハードウェアが倍精度実数をサポートしていない点である。第三に GPU はハード構成が単純であり、演算も単純なものに限られる点である。例えば、相互依存のある配列あるいは分岐処理は演算速度が遅くなる。

そこで、これらのディメリットを克服しながら既存のアルゴリズムを書き換えることにより CPU 並列の限界の克服が可能である。

5．CUDA

CUDA (Compute Unified Device Architecture：クーダ) とは、GPU のための統合開発環境である。当初の GPU プログラミングはアセンブラベースのため、使用者は限られていた。しかし 2007 年に NVIDIA 社が CUDA をリリースすると、GPU ユーザ数が急速に拡大した (Nguyen 2007, NVIDIA: CUDA ZONE)。この理由は二つある。第一には、C 言語により並列 GPU アプリケーションを作ることができるようになったことにある。つまり、GPU 言語が一部の特殊なプログラマーのみが扱えるアセンブラ言語から、一般の科学者が扱えるプ

ログラム言語に変化したことにある。第二に、CUDA が標準的な数値計算ライブラリである FFT と BLAS (Basic Linear Algebra Subroutines：ブラス) を含んでいることにある。既に、多くの科学者が CUDA を使って様々な問題を解いており、CUDA-ZONE には流体力学、地球科学、宇宙物理化学、生命科学、金融工学など多数の GPGPU 適用例が掲載されている (NVIDIA: CUDA ZONE)。しかし 2009 年 7 月 20 日現在、科学技術計算のなかでもっとも重い計算の一つである第一原理電子状態計算への適用例は未発表である。

II　計算環境と方法

1．ハードウェアとソフトウェア

本研究で使用する機器は、デスクトップパーソナルコンピュータである。仕様は Mother: Intel X58 chipset、CPU: Core i7 Quad 920 (2.66GHz) /4.8 GT/sQPI/cash 8M、Main memory: DDR3 1066 3GB である。ただし、CPU は 1 コアのみの利用である。GPU は GeForce GTX285 1GB をもちいる。本 GeForce GTX285 も倍精度実数はサポートされておらず、浮動小数点演算は単精度演算のみが可能である。

OS (Operating System：オペレーティングシステム) は openSUSE 11.1 86-64 をもちいる (openSUSE project)。コンパイラは、g95 Stable binary version 0.91, March 2008 である (The g95 project)。コンパイル時には-O3 オプションを付加し、GPU 上で行なう FFT 計算のみ単精度、その他は倍精度にてコンパイルする。CUDA はバージョン 2.1 である (NVIDIA: CUDA ZONE)。

平面波基底第一原理計算コードは、GNU General Public License の espresso 4.0.4 パッケージに含まれる PWscf (Plane-Wave Self-Consistent Field) をもちいる (PWscf project)。ソースコードは Fortran90 にて記述されている。

FFT ライブラリは、全 PWscf 計算を CPU 計算による場合は FFTW (Fast Fourier Transform in the West) 3.2.1、FFT 部のみを GPU 計算による場合は CUFFT 1.1 を利用する。(Frigo and Johnson, NVIDIA: CUDA ZONE) FFTW は汎用 CPU のための FFT ライブラリの中では、最速版に分類されている。CUFFT は、GPU ドライバ操作を自動化し、FFTW とほぼ同じインターフェースで FFT を GPU 並列計算するライブラリである。最適化とまでは言い難いが、多数のパラメータを持つドライバ操作を手動で設定する必要がないため、広く使われている。この両者を比べることにより、GPU-PWscf の性能を評価する。

時間計測は Fortran 組み込み関数の system_clock をもちいる。この方法での最小測定時間は 0.001 秒である。

2．解析対象系

計算する系は、ダイヤモンド型をもつシリコン結晶である。格子定数は 10.20 a.u.[1]、ノルム保存型擬ポテンシャルをもちいる (Hamann 1979)。交換相関は Perdew-Zunger LDA (Local Density Approximation：局所密度近似) である (Perdew 1981)。SCF (Self Consisted Field：自己無撞着場) 計算は DIIS 法 (Direct Inversion of the Iterative Subspace) を選択し (Pulay 1980)、k 点は Monkhorst-Pack 法による 8×8×8 点を用いる (Pulay 1980)。収束条件は、10-5 Ry[2]である。この条件下では、電子系の SCF 計算は 5 回で終了する。

3．ラッパー関数の作成

FFTW については、C 言語からの利用あるいは Fortran からの利用ともにインターフェースが用意されている。他方で、CUDA のインターフェースは C 言語のみである。これを Fortran から呼び出すためには 2 つの方法が考えられる。

第一に、FFT 計算において、メモリ確保、FFT 本計算、メモリ開放という 3 処理をセットにして C 言語としてコンパイルし、Fortran により作成された PWscf のオブジェクトファイルにリンクする方法が考えられる。しかし、これには以下の欠点がある。。メモリ操作の数と FFT 本計算の数は必ずしも一致せず、さらに、それぞれプログラムソース上の離れた場所に配置されることが多いので、メモリ確保・解除のオーバーヘッドが負担になる。また CUDA 独自の変数型で記述する GPU 上のデータを、Fortran を介してやりとりすることができないため、第一の方法では 3 個の処理をセットにして扱わざるを得ない。これには無駄が処理が付随する。我々はこの欠点を解決するために、第二の方法として、独自のラッパー関数を作成する。この関数は、CUDA 独自の変数型を整数型に変換、あるいその逆変換を行なうものである。これにより Fortran から同ラッパー関数を介して CUDA ライブラリが使えるようになる。Fortran で CUDA 独自の変数型のやりとりができるようになり、一連の操作を離した場所に配置することも可能になる。さらに、Fortran からの pinned メモリ操作[3]も可能になる。

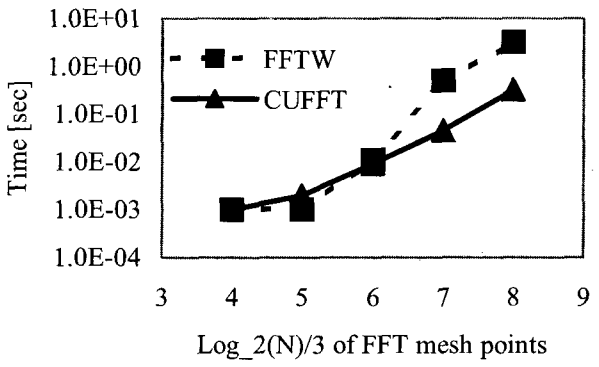

図1．三次元 FFT の計算時間を FFTW（破線＋正方マーク）と CUFFT（実線＋三角マーク）で比較した結果。横軸は、メッシュの１辺あたりのデータ数である。

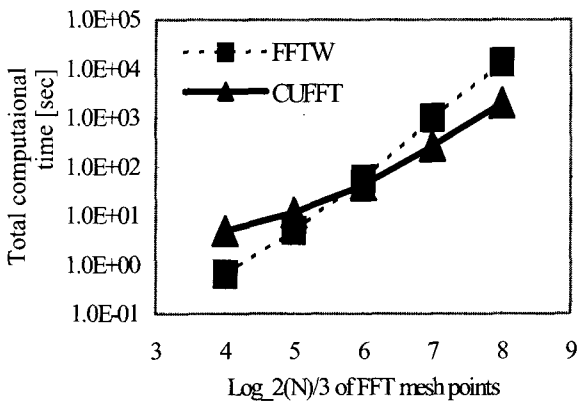

図２．2Si/菱面体のときの、FFT メッシュ数の関数としての PWscf 全計算時間を示す。目盛のとりかたと凡例は図１に同じである。

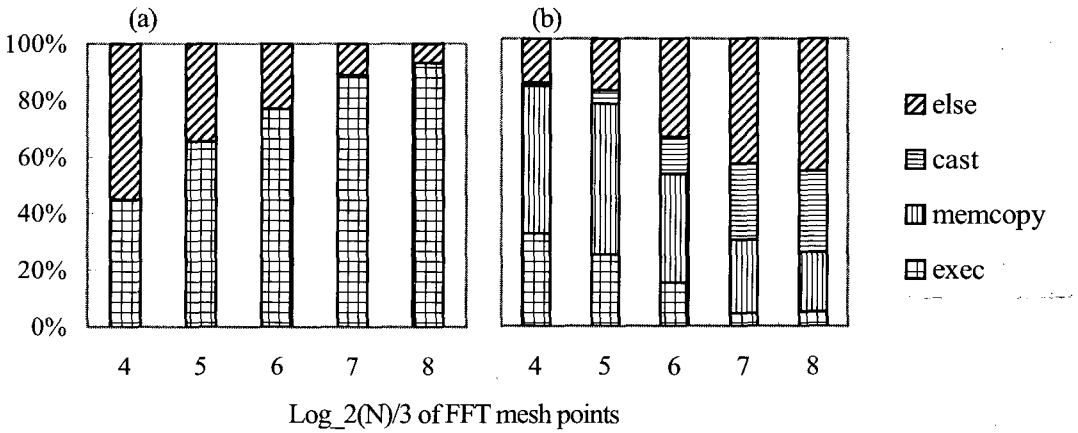

図３．2Si/菱面体のとき、FFT メッシュ数を増やした際の、PWscf 全計算に占める FFT 等計算時間の割合を示す。(a)が FFTW、(b)が CUFFT の結果である。格子縞 exec は FFT 本計算の時間をあらわし、縦縞 memcopy は CPU—GPU 間の往復データ転送時間、横縞 cast は倍—単精度間の往復変換時間、斜縞 else はその他の時間を示す。

我々は、このラッパー関数を使い、PWscf の GPU 計算に関連する変数を Fortran の allocate 関数から自作 pinned メモリ関数に置き換え、さらに FFTW におけるメモリ確保、FFT 本計算、メモリ開放の操作をそれぞれ CUFFT に置換する。

Ⅲ　結果

１．FFT

図１に三次元 FFT 単独の計算結果を示す。FFTW、CUFFT どちらも、データ数 $2^3\times2^3\times2^3=8\times8\times8$ までは 0.001 秒以下で時間測定できないが、$2^4\times2^4\times2^4=16\times16\times16$ になると 0.001 秒を観測する。$2^5\times2^5\times2^5=32\times32\times32$ では、FFTW が 0.001 秒であるのに対し、CUFFT が 0.002 秒で不利になるが、$2^6\times2^6\times2^6=64\times64\times64$ を超えると、データ数が増える程 CUFFT がより有利になる。メモリ限界の $2^{24}\times2^{24}\times2^{24}=256\times256\times256$ では、計算時間が約 10%に短縮される。速度にすると、10 倍速である。

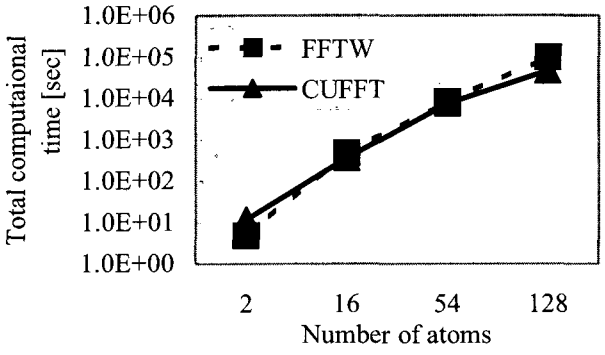

図4．FFT メッシュ数を一定にして原子数を増やした際の、PWscf 全計算時間を示す。メモリのとりかたと凡例は図1に同じである。

2．FFT メッシュ数の増加

ここでは、一定の系に対し、カットオフエネルギー値を増加させた場合の、GPGPU の効果をみる。その系とは結晶の単位胞を Si 原子2個分の菱面体である。

図2に PWscf の全計算時間を示す。FFTW 利用、CUFFT 利用どちらも、FFT データ数 $2^5\times2^5\times2^5=32\times32\times32$ までは、FFTW のほうが高速である。$2^6\times2^6\times2^6=64\times64\times64$ を超えると、CUFFT のほうが高速になる。$2^8\times2^8\times2^8=256\times256\times256$ では、計算時間は、15%に短縮される。速度比では6.9倍速である。

図3に、PWscf 全計算時間に対する FFT の占める割合を示す。(a)の FFTW の場合、FFT データ数が増えるほど、その割合も大きくなり、$2^8\times2^8\times2^8=256\times256\times256$ では93%になる。他方で、CUFFT 利用の場合はその逆で、FFT データ数が増えるほど、その割合が小さくなる。$2^8\times2^8\times2^8=256\times256\times256$ では、FFT 本計算の割合は 5%である。ただし、倍—単精度間の往復変換作業と、CPU—GPU 間の往復データ転送作業が余計に必要となる。しかし、この時間を合わせても、$2^8\times2^8\times2^8=256\times256\times256$ で54%となり、FFTW 利用よりも有利である。

最後に、FFT 部分を単精度に落とすことによる、計算結果への影響を調べるため、FFT メッシュ数 $32\times32\times32$ として全エネルギーと各原子に働く力の大きさを確認する。CUFFT 利用、FFTW 利用どちらも、完全結晶のとき原子間力が零 Ry と計算されることを確認し、1原子を最近接原子の方向に格子定数の0.2%つまり 0.02 a.u.だけ近づけた。このときの全エネルギーの差は $\Delta E_T=4.30\times10^{-6}$ Ry である。比較のため、原子数8個のときの同誤差を調べると、$\Delta E_T=1.94\times10^{-5}$ Ry であり、原子数2個の場合の約4倍になっている。つまり、原子数と全エネルギーは正比例していることがわかる。一方で原子間力の誤差は、原子数にかかわらず、1.0×10^{-6} Ry/a.u.のオーダーである。これは、実用に十分耐え得る。

3．原子数の増加

ここでは、カットオフエネルギー値一定（18 Ry）として原子数を増加させる場合の計算速度を調べる。具体的には、1、2、3、4倍の格子定数を持つ菱面体の単位胞を用意し、この4種の計算速度を比較する。それぞれのスーパーセルには Si 原子が2個、16個、54個、128個を含み、基底関数の数は2,733、22,075、74,129、175,215 である。選択する FFT メッシュの数は、$32\times32\times32$、$64\times64\times64$、$64\times64\times64$、$128\times128\times128$ である。（スーパーセルサイズとメッシュ数の関係を付録 A に示す）使用マシンのメモリ容量の上限界から、これ以上の系の計算は不能であい。図4は実際にこの計算時間を計測した結果であり、CUFFT 利用/FFTW 利用の速度比は、0.4、1.3、1.1、2.0倍である。このときの PWscf 全計算時間に対する FFT の占める割合を図5に示す。原子数が2個のときは、図2、3の結果から予測されるとおり、CUFFT 利用のほうが不利であるが、原子数が16個を超えると、基底関数の数も増えて FFT 自体は CUFFT 利用のほうが有利になるが、FFT 以外の演算時間が相対的に増えている。

Ⅳ　考察・結論

本研究では、大規模第一原理電子状態計算において最も計算量の多い部分が FFT であることを確認し、この部分に対する GPGPU の適用とその効果を評価した。

GPU をもちいる制約から FFT 計算が倍精度から単精度に落ちるが、これによる原子間力の誤差は 1.0×10^{-5} Ry/a.u 未満であるので、十分に使用に耐える。現在、倍精度計算可能な GPU 基盤が市販され始めているが、それによる倍精度計算は単精度計算よりも遅く、高価である。ゆえに、本研究の GPU による単精度計算が最もローコスト・ハイリターンである。

システムサイズを固定した上で、FFT メッシュの数を増加して第一原理電子状態計算の速度を比較した。その結果、GPGPU 適用により、計算時間を最大15%、速度にして約6.9倍加速できた。メッシュ数の増大は、より高い周波数成分の平面波まで考えるということに

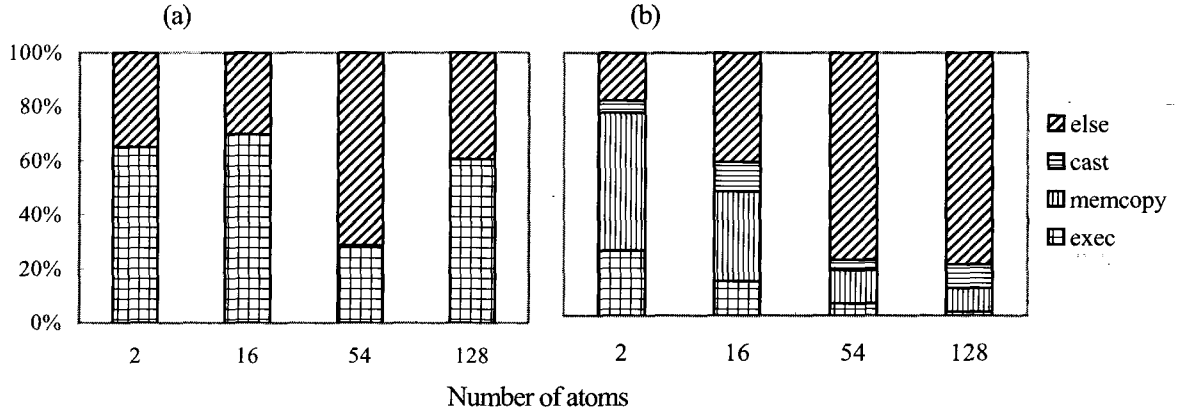

図５．FFT メッシュ数を一定にして原子数を増やした際の、PWscf 全計算に占める FFT 等計算時間の割合を示す。(a)が FFTW、(b)が CUFFT の結果である。凡例は図３に同じである。

なり計算精度向上を意味する。固定したシステムサイズでの FFT メッシュ数増大は、解析対象の精度向上に対応する。。

原子数を増やした場合は、FFT 以外の計算時間も増えてしまう。具体的には、行列演算に関連する部分である。FFT は原子数の増加に対して計算量が $O(N\log N)$ で増えるのに対し、基底関数に関連する行列演算の部分は $O(N^2)$ で増える。よって、さらなる効果を出すためには、この行列演算の部分を GPU にて加速する必要がある。

謝辞

第一原理計算プログラムの時間計測とは別に、プログラム開発にあたっては、明治大学情報基盤本部の大型計算機 samba00 と isc-pcc、flamingo を利用した。

付録Ａ　第一原理計算における FFT メッシュ数

結晶軸を $\mathbf{a}_1$、$\mathbf{a}_2$、$\mathbf{a}_3$ とすると、結晶内の任意の位置は

$$\mathbf{r} = n_1\mathbf{a}_1 + n_2\mathbf{a}_2 + n_3\mathbf{a}_3 \tag{1}$$

と書ける。この 3 個の実空間ベクトルが作る単位胞の体積 Ω は、

$$\Omega = \mathbf{a}_1 \cdot (\mathbf{a}_2 \times \mathbf{a}_3) \tag{2}$$

である。一方で、波動関数 Ψ をフーリエ級数展開すると、

$$\Psi(\mathbf{r}) = \sum_{\mathbf{G}} A_{\mathbf{G}} e^{i\mathbf{G}\cdot\mathbf{r}} \tag{3}$$

である。ここで δ_{ij} をクロネッカーのデルタとし、

$$\mathbf{b}_i \cdot \mathbf{a}_j = 2\pi\delta_{ij} \tag{4}$$

をみたすような逆空間（スペクトル空間）ベクトル $\mathbf{b}_i$ を導入し、

$$\mathbf{G} = m_1\mathbf{b}_1 + m_2\mathbf{b}_2 + m_3\mathbf{b}_3 \tag{5}$$

という格子ベクトルをつくる。すると式(2)と(4)から、

$$\begin{cases} \mathbf{b}_1 = \dfrac{2\pi}{\Omega}\mathbf{a}_2 \times \mathbf{a}_3 \\ \mathbf{b}_2 = \dfrac{2\pi}{\Omega}\mathbf{a}_3 \times \mathbf{a}_1 \\ \mathbf{b}_3 = \dfrac{2\pi}{\Omega}\mathbf{a}_1 \times \mathbf{a}_2 \end{cases} \tag{6}$$

である。また、式(4)から $\mathbf{G}$ と $\mathbf{R}$ の内積が、

$$\mathbf{G}\cdot\mathbf{R} = 2\pi\,\mathbb{Z} \tag{7}$$

となるので、

$$e^{i\mathbf{G}\cdot\mathbf{R}} = 1 \tag{8}$$

となり、式(3)であらわした波動関数 Ψ は、周期性

$$\Psi(\mathbf{r}+\mathbf{R}) = \Psi(\mathbf{r}) \tag{9}$$

表 Ａ１．基底関数 N_{PW} による必要 FFT メッシュ数 N_{FFT}。

N_{PW}	N_{FFT}
4 まで	8 以上
33 まで	64 以上
268 まで	512 以上
2,144 まで	4,096 以上
17,156 まで	32,768 以上
137,248 まで	262,144 以上
1,097,985 まで	2,097,152 以上
8,783,882 まで	16,777,216 以上

表 A2. 結晶 Si の場合の原子数 N_{atom}、基底関数 N_{PW}、FFT のメッシュ数 N_{FFT} の関係。

N_{atom}	N_{PW}	N_{FFT}	N_{PW}/N_{FFT}
2	2,733	32,768	0.0834
16	22,075	262,114	0.0842
54	74,129	262,114	0.2828
128	175,215	2,097,152	0.0835
250	342,133	2,097,152	0.1631
432	591,539	2,097,152	0.2821
686	938,529	2,097,152	0.4475
1,024	1,401,043	16,777,216	0.0835

をみたす。

ブロッホの定理から周期境界条件下の電子状態は、波数ベクトル $\mathbf{k}$ とブロッホ周期関数

$$\mathrm{u}_{\mathbf{k}}(\mathbf{r}+\mathbf{R}) = \mathrm{u}_{\mathbf{k}}(\mathbf{r}) \tag{10}$$

を使って、

$$\Psi_{\mathbf{k}}(\mathbf{r}) = e^{-i\mathbf{k}\cdot\mathbf{r}}\,\mathrm{u}_{\mathbf{k}}(\mathbf{r}) \tag{11}$$

と書けるので、式(3)によりこれを逆格子ベクトルに展開し、

$$\Psi_{\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{G}} c_{\mathbf{k}+\mathbf{G}} e^{-i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}} \tag{12}$$

と展開できる。

式(12)の波動関数は、$\mathbf{G}$ についての無限の和であるが、実際に計算する際にはどこかで打ち切らなければならない。それを決めるのがカットオフエネルギー E_{cut} であり、

$$|\mathbf{k}+\mathbf{G}|^2 < E_{cut} \tag{13}$$

にある平面波のみを扱うことになる。逆空間の球で定義され、球の半径（カットオフ半径）

$$k_c = \sqrt{E_{cut}} \tag{14}$$

よりも小さい波数ベクトルの平面波を含む。k_c 内の平面波の数 N_{PW} は、

$$N_{PW} = \frac{\frac{4\pi}{3}k_c^{\,3}}{\frac{(2\pi)^3}{\Omega}} \tag{15}$$

と見積もられる。ここから、格子定数が変化したとしても計算精度を一定に保つためには、平面波の数が一定になるように k_c を調整する必要があることがわかる。他方で、FFT 計算のメッシュ数 N_{FFT} は、連続的な整数値をとることができない。そこで、実際の N_{FFT} は、必要な数よりも大きい値を設定することになる。本稿では議論の簡易化のため、N_{FFT} は 2 のべき乗に限定している。

N_{PW} と N_{FFT} の比は、逆空間における FFT メッシュの作る立方体の体積と平面波球の作る球の体積の比にほぼ等しい。よって、その最大値は図 A 1 のように、FFT 体に平面波球が内接する場合で、

$$\frac{N_{PW}}{N_{FFT}} = \frac{\frac{4}{3}\pi k_c^{\,3}}{(2k_c)^3} \approx 0.5 \tag{17}$$

である。これらより、実際の第一原理計算にて利用される N_{PW} と N_{FFT} の関係を計算したものが表 A1 である。そして、第Ⅲ章３節にて扱った系の N_{PW} と N_{FFT} の関係は表 A2 である。

一定の N_{PW} に対しては、できるだけ N_{FFT} が大きくなるような場合が FFT の計算割合が大きい計算となる。つまり、N_{PW}/N_{FFT} が小さい計算が GPGPU に有利といえる。本文の結果も、確かにこの傾向を確認できる。

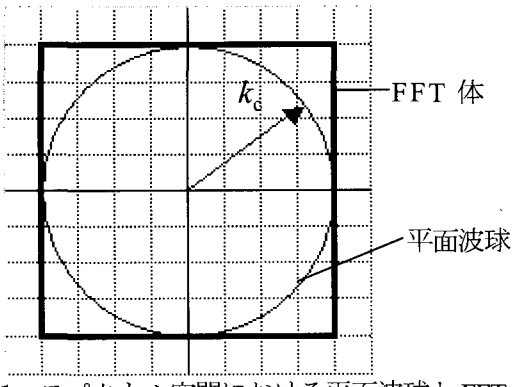

図 A 1．スペクトル空間における平面波球と FFT 体の関係。

【注】

1) a.u.：原子単位系における長さの単位。岩波理化学辞典第 4 版によると、1 a.u. = 0.529×10^{-10} m である。
2) Ry：原子単位系におけるエネルギーの単位。岩波理化学辞典第 4 版によると、1 Ry. = 21.80×10^{-19} J である。
3) pinned メモリ操作：GPU 上のメモリ領域と CPU 側のメモリ領域との間でデータ転送を効率的に行なうために、CPU 側のメモリ領域の確保を最適化する手法。

【参考文献】

Hamann, D. R., M. Schluter and C. Chiang (1979) Norm-Conserving Pseudopotentials, ***Phys. Rev. Lett.***,43: 1494.

Hohenberg, P., and W. Kohn (1964) Inhomogeneous Electron Gas, ***Phys. Rev.***, 136: B864 - B871.

飯高敏晃（2007 - 2008）「カエルにもわかるGPUによる科学計算入門（1）－（4）」日本計算工学会編『計算工学』（1）第12巻第4号1698－1703，（2）第13巻第1号1766－1771，（3）第13巻第2号1817－1822，（4）第13巻第3号1882－1887.

Kohn, W., and L. J. Sham (1965) Self-Consistent Equations Including Exchange and Correlation Effects, ***Phys. Rev.***, 140: A1133 - A1138.

Makino, J, and T. Taiji: (1998) ***Scientific Simulations with Special-Purpose Computers—the GRAPE Systems***, Wiley Blackwell.

Monkhorst, M. J. and J. D. Pack (1976) Special points for Brillouin-zone integrations, ***Phys. Rev. B***, 13: 5188.

Nguyen, H. (2007) ***GPU Gems 3*** Addison - Wesley Professional.

Nukada, A., Y. Ogata, T. Endo, and S. Matsuoka (2008) Bandwidth intensive 3-D FFT kernel for GPUs using CUDA, SC '08 Proceedings of the 2008 ACM/IEEE conference on Supercomputing, ***IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis***, Article No. 5, Austin: IEEE press.

NVIDIA Co. (2008) ***NVIDIA CUDA™ Programming Guide***, ver. 2.1.

Payne, M. C., M. P. Teter, D. C. Allan, T. A. Arias, and J. D. Joannopoulos (1992) Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients, ***Rev. Mod. Phys.***, 64: 1045－1097.

Perdew, J. P. and A. Zunger (1981) Self-interaction correction to density-functional approximations for many-electron systems, ***Phys. Rev. B,*** 23: 5048.

Pulay, P (1980) Convergence acceleration of iterative sequences. the case of scf iteration, ***Chem. Phys. Lett.***, 73: 393.

寺倉清之編（2009）「特集：電子状態の第一原理計算の現状と課題」日本物理学会編『日本物理学会誌』第64巻第4号，241－296.

【URL】

Frigo, M., and S. G Johnson ***FFTW*** (http://www.fftw.org).

Harris, M. ***GPGPU.org*** (http://gpgpu.org/).

NVIDIA ***CUDA ZONE*** (http://www.nvidia.com/cuda/).

openSUSE Project (http://www.opensuse.org/).

PWscf project (http://www.pwscf.org/); Quantum ESPRESSO (http://www.quantum-espresso.org/).

The G95 Project (http://www.g95.org/).