Acceleration of Orbital-Free First Principles Calculation with Graphics Processing Unit GPU

M Aoki¹, H Tomono², T Iitaka³ and K Tsumuraya²

¹School of Management, Shizuoka Sangyo University, 1572-1, Ohwara, Iwata, 438-0043, Japan
²School of Science & Technology, Meiji University, 1-1-1, Higashi-mita, Tama, Kawasaki, 214-8571, Japan
³Computational Astrophysics Laboratory, RIKEN (The Institute of Physical and Chemical Research), Hirosawa 2-1, Wako, Saitama, 351-0198, Japan

E-mail: maoki[at mark]ssu.ac.jp

Abstract. Computational material design requires efficient algorithms and high-speed computers for calculating and predicting material properties. The orbital-free first principles calculation (OF-FPC) method, which is a tool for calculating and designing material properties, is an $O(N)$ method and is suitable for large-scaled systems. The stagnation in the development of CPU devices with high mobility of electron carriers has driven the development of parallel computing and the production of CPU devices with finer spaced wiring. We, for the first time, propose another method to accelerate the computation using Graphics Processing Unit (GPU). The implementation of the Fast Fourier Transform (CUFFT) library that uses GPU, into our in-house OF-FPC code, reduces the computation time to half of that of the CPU.

1. Introduction

Recently, first principles calculation (FPC) methods based on density functional theory (DFT)¹ have become prominent in the design of materials due to advances in the approximation of the electron-correlations. In order to study the large-scaled systems such as amorphous systems, or biomolecules such as proteins with the method, the development of fast computing methods is necessary. The orbital-free first principles calculation (OF-FPC) method developed by Pearson et al.² is one of such methods, since the method is an $O(N\log_2 N)$ method for FFT calculation and an $O(N)$ method for the other part of calculation. The method has been adopted to study the large-scaled systems such as metallic glasses³, liquid metals⁴, lattice defects⁵, and metallic clusters⁶. However, we need faster computers to study the larger system such as bio-molecules.

There have been attempts to produce CPUs using devices with high mobility of electron carriers. The mobility of electron in GaAs compound semiconductor is six times as fast as the one of silicon semiconductor. The stagnation in the production of the compound for the CPU devices allows us to develop the high-performance parallel computing, i.e. the cluster computing and the silicon CPU devices with more fine spaced wiring.

One of the other steps to overcome the stagnation is the use of the GPU (Graphics Processing Unit) for the computation. Although the GPUs have been used only for graphical processing, recently NVIDIA Corporation has released an integrated development environment CUDA (Compute Unified
Device Architecture) [7] written in C language. The use of the GPU for the numerical calculation is called the GPGPU (General-Purpose Graphics Processing Unit)[8]. A current GPU device has 240 computing cores per processor, although a current CPU has at most only 32 cores. The processing speed of the NVIDIA Tesla S1070 (four processors) is 4.14 TFlops in single-precision, although that of Intel core i7-965 is only 51.20GFlops. The speed per processor of the GPGPU is 20 times faster than that of the CPU. The NVIDIA Corporation provides us CUFFT (FFT for CUDA; Fast Fourier Transformation) written in C language.

In the present study we accelerate the calculation by implementing CUFFT into our in-house OF-FPC code. This is because the code spends more than 60%, for a large system, of the almost computation of time for the FFT calculation to evaluate the electron kinetic energy.

2. CUFFT vs. FFTW

The FFTW is one of the fastest FFT routines with CPU at the present stage of the world. First we compare the computation time of the code of the CUFFT (GPGPU calculation) with that of FFTW[9] (CPU calculation) to evaluate the extent of the GPGPU acceleration of the FFT routine.

Figure 1 shows the computation time of the 3D-CUFFT and 3D-FFTW as a function of the total number of FFT mesh $N$. We evaluate the times for the single- and double-precision FFTW(FFTW(SP) and FFTW(DP)), and the single-precision CUFFT(CUFFT(SP)), since the CUFFT library is released with only single-precision version, where SP is the single precision calculation and DP is the double precision one. We show $T[\text{FFTW(DP)}]$ for the computation time of FFTW(DP) as a function of FFT size $N$ in Fig.1. For $\log_2 N = 24$, the ratio of the times is $T[\text{CUFFT(SP)}]/T[\text{FFTW(DP)}] = 1/16.45$. This is due to the use of the CUFFT and is not due to the use of the single-precision calculation changed from the double-precision one. This is because that, although the ratio $T[\text{FFTW(SP)}]/T[\text{FFTW(DP)}]$ is only $1/1.28$, the ratio $T[\text{CUFFT(SP)}]/T[\text{FFTW(SP)}]$ amounts to $1/12.87$. Thus the implementation of CUFFT(SP) reduces the time compared with the use of FFTW(DP).

![Figure 1](chart.png)

**Figure 1** The computation time of the calculations of forward FFT and then inverse FFT for the 3D-CUFFT and 3D-FFTW as a function of the total number of FFT mesh $N$; SP: single-precision and DP: double-precision.

3. Acceleration of the OF-FPC code with CUFFT

We compare the computation time of the OF-FPC code with the CUFFT(SP) with that with FFTW(DP). The system selected is the sodium crystals containing 2, 16, 128, 1024 and 6750 atoms in each supercell. The lattice constant used is 4.225 Å. In the optimization of the electronic system, we use the Topp-Hopfield pseudopotential[10] and the Perdew-Zunger exchange correlation energy functional.[11] The cut-off energy $E_{\text{cut}}$ of the system is 11(Ry) Table 1 shows the total number of FFT
meshes $N$ in the supercells calculated in the present study. The number $N$ increases with the size of the cell, since the density of the mesh is conserved. We optimize the electron systems with the steepest-decent method. The computation times to iterate for 500 steps are counted. The OF-FPC code calls the FFT routine 10 times per each iteration step, so the code calls the FFT 5000 times per 500 steps. The machine specification used is Mother Board: Intel X58 chipset, CPU: Core i7 Quad 920 (2.66GHz), Main Memory: DDR3-1066 3GB, and GPU: GeForce GTX285 1GB.

The original OF-FPC code is written in double-precision for variables although the CUFFT library is written in single-precision. So we have changed all the OF-FPC code into a single-precision version and checked the accuracies of the code using both the total energies and the inter-atomic forces. For the system with two sodium atoms in the simple cubic lattice, the total energy of the single-precision OF-FPC code has been coincident with that of double-precision code to six decimal places; the error corresponds to $3.1 \times 10^{-3}\%$. We have calculated the force of a sodium atom when we have displaced the atom from the body-centered position to the other sodium atom by 10% of the inter-atomic distance. The error is 0.15\%. Both the errors are negligible, so the single-precision version is capable of calculating the electronic states and the dynamic states of the sodium systems.

Figure 2 shows the computation times of the OF-FPC code with FFTW and CUFFT as a function of system size $N$. The times increase with increasing the FFT meshes. The increase of the time with the CUFFT is smaller than those of the FFTW. For $\log_2 N = 24$ (6750 atoms in supercell), the ratio of the computation time $T[\text{OF-FPC(DP) with FFTW(DP)}]/T[\text{OF-FPC(SP) with CUFFT(SP)}]$ is 1/3.2, indicating the acceleration of the calculation with CUFFT(SP). This is also due to the use of CUFFT(SP) changed from the use of FFTW(SP), since the ratio of $T[\text{OF-FPC(DP) with FFTW (SP)}]/T[\text{OF-FPC(DP) with FFTW(DP)}]$ is only 1/1.28 and the $T[\text{OF-FPC(SP) with CUFFT(SP)}] / T[\text{OF-FPC(SP) with FFTW(SP)}]$ amounts to 1/2.5. Thus the implementation of CUFFT reduces the computation time of the OF-FPC code.

Table 1 The system sizes calculated in the present study.

| Num. of atoms in unit cell | Num. Of basis functions$^a$ | Total number $N$ of FFT meshes |
|----------------------------|-------------------------------|-------------------------------|
| 2                          | 305                           | 4,096 (=2$^{12}$)             |
| 16                         | 2,517                         | 32,768 (=2$^{15}$)            |
| 128                        | 20,005                        | 262,144 (=2$^{18}$)           |
| 1024                       | 160,467                       | 2,097,152 (=2$^{21}$)         |
| 6750                       | 1,283,951                     | 16,777,216 (=2$^{24}$)        |

$^a$The number of basis functions for the fixed cut-off energy $E_{cut}=11$ Ry.

Figure 3 shows the fraction of the computation time of the FFTW to the total computation time of the OF-FPC code, where the total computation times have been shown in Fig.2 as a function of size $N$. The fraction increases with the number of FFT mesh. The OF-FPC method is $O(N)$ method and the computational cost of FFT is proportional to $O(M\log_2 N)$, the fraction of the computational time of FFT to the total computation time becomes large as the system size increases. However, the fraction of the use of CUFFT shows a reverse tendency as shown in Fig.4. This is due to the drastic decrease of the
time of CUFFT computation comparing with that of FFTW computation shown in Fig.1. This acceleration leads to the decrease of the fraction of the CUFFT computation time to the total computation time for the large-scaled systems.

![Figure 3](image3.png)  
**Figure 3** The fraction of the FFTW(SP) computation time to the total computation time of the OF-FPC(SP) code. The 'Others' is the computation time other than the FFTW calculation.

![Figure 4](image4.png)  
**Figure 4** The fraction of the CUFFT(SP) computation time to the total computation time of OF-FPC(SP) code. The ‘memcpy’ is the time for data transfer between the CPU and the GPU; the ‘Others’ is the time other than the CUFFT calculation.

4. Conclusions

We have accelerated our in-house OF-FPC code by implementing the CUFFT in CUDA. The implementation of the Fast Fourier Transform (CUFFT) library that uses GPU, into our in-house OF-FPC code, reduces the computation time to half of that of the CPU with the FFTW for $\log_2 N = 24$ with 6750 atoms in supercell.

The numerical calculations were partly carried out using SCore cluster systems in Meiji University and Altix3700 BX2 at YITP in Kyoto University.

References

[1] Hohenberg P and Kohn W 1964 *Phys. Rev. B* **136** 864; Kohn W and Sham L J 1965 *Phys. Rev. A* **140** 1133

[2] Pearson M, Smargiassi E and Madden P A 1993 *J. Phys. Condens. Matter*, 5 3221

[3] Aoki M I and Tsumuraya K 1996 *J. Chem. Phys.* **104** 6719; Aoki M I and Tsumuraya K 1997 *Phys. Rev. B* **56** 2962

[4] Foley M, Smargiassi E and Madden P A 1994 *J. Phys. Condens. Matter*, 6 5231

[5] Smargiassi E and Madden P A 1995 *Phys. Rev. B* **51** 129; Smargiassi E and Madden P A 1995 *Phys. Rev. B* **51** 117

[6] Shah V, Nehete D and Kanhere D G 1994 *J. Phys. Condens. Matter* **6** 10773; Nehete D, Shah V and Kanhere D G 1996 *Phys. Rev. B* **53** 2126

[7] NVIDIA, "CUDA ZONE", http://www.nvidia.com/object/cuda_home.html

[8] Harris M, "GPGPU.org", http://www.gpgpu.org/

[9] Frigo M, and Johnson S G, "FFTW", http://www.fftw.org/

[10] Topp W C and Hopfield J J 1973 *Phys. Rev. B* **7** 1295

[11] Perdew J P and Zunger A 1981 *Phys. Rev. B* **23** 5048