GPU based acceleration of first principles calculation

GPU based acceleration of first principles calculation

H Tomono[1,2], M Aoki[3], T Iitaka[2], K Tsumuraya[1]

[1] School of Science and Technology, Meiji University, Kawasaki, Kanagawa 214-8571, Japan
[2] Computational Astrophysics Laboratory, RIKEN (The Institute of Physical and Chemical Research), Wako, Saitama 351-0198, Japan
[3] School of Management, Shizuoka Sangyo University, Iwata, Shizuoka 438-0043, Japan

E-mail: abinitio[at mark]isc.meiji.ac.jp [K Tsumuraya]

Abstract. We present a Graphics Processing Unit (GPU) accelerated simulations of first principles electronic structure calculations. The FFT, which is the most time-consuming part, is about 10 times accelerated. As the result, the total computation time of a first principles calculation is reduced to 15 percent of that of the CPU.

1. Introduction

1.1. First principles planewave method

First principles or *ab initio* method is a prized technique to calculate electronic structures of materials. Although it does not only predict solid state properties but also design new materials, Central Processing Unit (CPU) and memory abilities limit the number of atoms to a hundred or thousand. The hundred is the order of the simplest and smallest proteins or amino acids. To increase the number, the order-*N* first principles algorithms are developed.[1,2]

In the first principles calculation, we solve a partial differential equation, which is called Kohn-Sham equation and based on Shcrödinger equation.[3,4] To solve the equation, the planewave method projects the wavefunction onto a spectrum space. On the other hand, the electronic correlation function is defined in a real space. Because the solution is self-consistent, the wavefunctions need to shuttle many times in the spectral space and real space by Fast Fourier transform (FFT).

Although the FFT routine is only a part of the million lines in the *ab initio* program source codes, the FFT spends more than half proportion of the total calculation time depending on the system size. For the first principles calculations with large systems, it is necessary to speed up the FFT calculation. In this paper, we substitute the CUFFT with GPU (see Section *1.2*) for an FFT routine with the CPU and show that the GPU is available to the large-scale first principles calculations. Because the GPU has very broad memory bandwidths, its FFT time is much shorter than that of the CPU.[5]

1.2. GPU and GPGPU

Graphics Processing Unit (GPU) is a specialized device rendering and accelerating for graphic operations. It is a commercial and mass production for general consumers, and has been used in personal computers, workstations, and game consoles and so on. The original use of the GPU is to entertain us with powerful 3D games or movies.[6]

General Purpose computing on Graphics Processing Units (GPGPU) is a technique of using the GPU, which typically handles computation only for computer graphics, to perform the other computations.[7] It begins to be paid to attention, because the acceleration using the silicon devices is limited. One of the methods to break down the limit is the parallel computing using multiple CPUs. It has been a common usage in the field of the high-performance computing. Another method is the GPGPU. The GPU has advantages over CPU in the graphic, because the GPU devices are low prices, broad bandwidth, and high speed of the floating point operation.[8] The devices are faster than the CPU especially in the calculations of matrix-matrix multiplication, FFT calculation, and the *N*-body problem etc.[9] In this paper, we apply the GPGPU to the first principles electronic structure calculation.

1.3. CUDA

Compute Unified Device Architecture (CUDA) is a free integrated development environment for the GPU. In early days of the GPU programming, the programmers were limited because the language for the GPU was the assembly language, which was complicated in programming. The number of GPU programmers, however, has expanded since the NVIDIA Co. released NVIDIA® CUDA™ in 2007.[10,11] The CUDA permits us to use standard C language for the parallel application development with the GPU. In addition, the CUDA has standard numerical libraries for the FFT and BLAS (Basic Linear Algebra Subroutines). Many developers are already using the CUDA to solve problems in a variety of professional – physics simulations, oil and gas exploration, medical imaging, scientific research, and among others. No application of the GPGPU to the first principle calculation, which is one of the heaviest calculations in the scientific computation, has been published as of May, 2009.

2. Methods

2.1. Hardware and Software

Our calculator is a desktop personal computer. The specifications are mother: Intel® X58 chipset, CPU: Intel® Core™ i7 Quad 920 (2.66 GHz) /4.8 GT/sQPI/cash 8M, and main memory: DDR3 1066 3 GB. In this paper, the use of the CPU is only one core. The GPU which we use is NVIDIA GeForce® GTX285 1 GB. In generally, the GPUs process single precision. This NVIDIA GeForce GTX285 is the same condition.

The followings are our software. Operating system is openSUSE 11.1.[12] The source codes are compiled and linked with no option by g95 binary version 0.91, March 2008.[13] The version of the CUDA is 2.1.[11] The source code are double precisions except the FFT routine on GPU with single precisions. Time is measured with the "system_clock" which is one of the Fortran 90 intrinsic subroutines. The measurement precision is a millisecond.

Our first principles code is PWscf (Plane-Wave Self-Consistent Field) in the package espresso 4.0.4, which is GNU General Public License.[14] It calculates electronic structures, based on density-functional theory, planewave basis sets, and pseudopotentials.

The FFT libraries used are CUFFT 1.1 on GPU [11] and FFTW (Fast Fourier Transform in the west) 3.2.1 on the CPU.[15] The former CUFFT is one of the CUDA libraries, the latter FFTW is the one of the most popular and its performance is typically superior to that of other publicly available FFT libraries. The CUFFT is 10 times faster than FFTW on our environment.

2.2. System

The system to assess the relative GPU to CPU performances is a diamond Si crystal. There are two silicon atoms in a rhombohedra unit cell. The lattice constant is set to 10.21 a.u. The style of Pseudopotentials is norm conserving. The charge densities are calculated within the framework of Perdew-Zunger local-density approximation (LDA) exchange correlation. The k-points are selected, using Monkhorst-Pack method 8x8x8. The numbers of FFT meshes, which depend on cutoff energy of

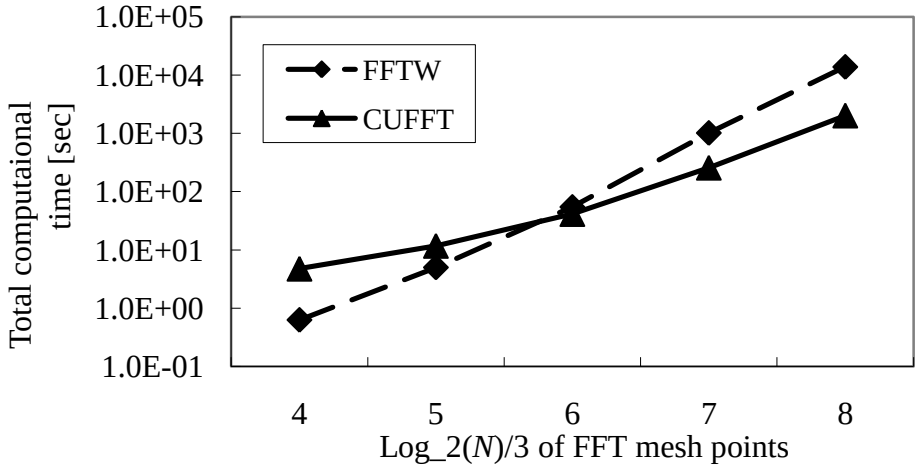

Figure 1. Total time of PWscf computations. The abscissa axis is the number of meshes per a cubic side, 2^4 x 2^4 x 2^4–2^8 x 2^8 x 2^8.

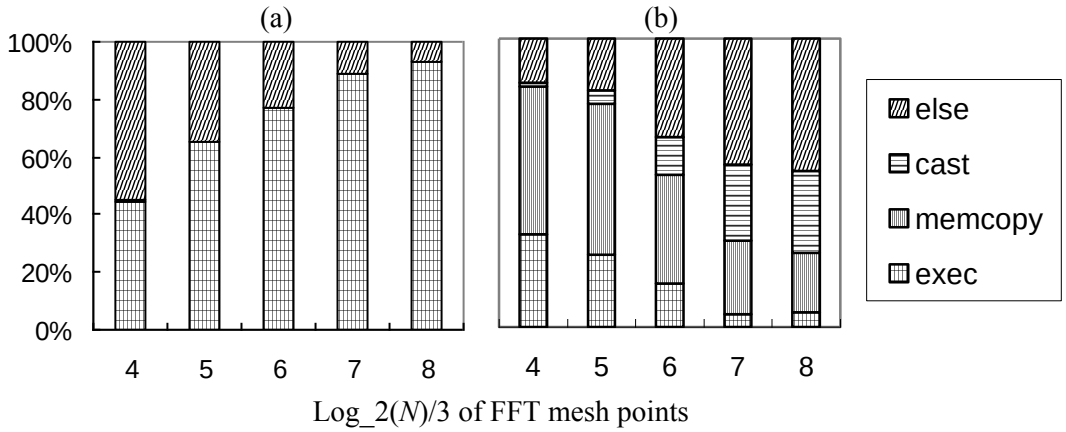

Figure 2. Time percentage of routines. (a) FFTW (left), (b) CUFFT (right). Check pattern is real FFT execution (*exec*). Vertical pattern is transfer of data (*memcpy*). Horizontal pattern is type conversions (*cast*). Oblique pattern is the others.

charge density, are 2^{3N}: N = 4, 5, 6, 7, 8. The convergence threshold for electronic selfconsistency is set to 10^{-5} Ry. Under these conditions, the electronic SCF iterations converge within five iterations.

3. Results

Figure 1 shows PWscf's total computational time. Horizontal axis is the length of cubic. For example, "4" is 2^4 x 2^4 x 2^4 cubic mesh. The number "8" is the limit of our GPU memory. Under the "5", the time of the CUFFT is longer than that of the FFTW. Over at "6", the time of the CUFFT and the FFTW reverses the situation. In addition, the difference between the CUFFT and the FFTW increases with the number of meshes. At "8", CUFFT time is 15% of FFTW time i.e., the CUFFT speed is 6.9 times as fast as the FFTW speed.

Figure 2 shows the percentage of the FFT time in the total computational time. In the case of (a) FFTW, the FFT percentage increases with the number of the FFT meshes. In the case of (b) CUFFT, the fraction of the exec decreases with increase of the FFT mesh; the exec percentage is only 5% at "8". The CUFFT needs two kinds of extra works: type conversions (*cast)* between single and double precisions for the GPU's processing with single precision, and transfer (*memcpy*: memory copy) of

data in the each dedicated memory to CPU and GPU. The percentage increases to 54% even if the cast and memcpy time is included in the real FFT time, which is superior to FFTW.

The use of the single precision calculation for the FFT routine is negligible for the errors in the PWscf calculations. The errors of total energies are the order of 10^{-6} Ry which is equal to the order of the convergence threshold order; the force errors are less than 10^{-7} Ry/a.u. for the displacement of an atom by 0.2% of the lattice constant in the <111> direction. We have no problems using the results of the CUFFT for the PWscf calculation.

4. Conclusions and Discussion

The implementation of the GPU FFT routines into the first principles code has reduced the total computational time of PWscf to only 15% of the CPU. It is because of the FFT routines are the hot spot of the code. This performance has been measured under fixing the system size and increasing the number of the FFT meshes. Increasing the number, high-frequency components of planewaves are counted and the computational accuracy is improved. Increasing the number with the constant system corresponds to enlarging the system.

Both the accuracies on the total energy and on the atomic force have been conserved in replacing the double precision FFT routine with the single precision CUFFT routine. The products of GPUs that process double precisions are expensive and the speed of processing double precisions is slower than that of single precisions. For example, in the case of NVIDIA Tesla™ C1060, the double precision speed is 8% of the single precision speed. Therefore, at this moment, the processing of the single precision with GPU is the most efficient at the smallest cost to accelerate the first principles calculation.

Acknowledgement

Developing and computations of the program source codes were performed in part using SCore systems at the Information Science Center in Meiji University and Altix 3700 BX2 at YITP in Kyoto University.

References

[1] Goedecker S 1999 *Rev. Mod. Phys* **71** 1085
[2] Bowler D R, Fattebert J–L, Gillan M J, Haynes P D, Skylaris C–K 2008 *J. Phys.: Comdens. Matter* **20** 2290301
[3] Hohenberg P, Kohn W 1964 *Phys. Rev*. **136** B864
[4] Kohn S, Sham L J 1965 *Phys. Rev*. **140** A1133
[5] Nukada A, Ogata Y, Endo T, Matsuoka S 2008 *Bandwidth intensive 3-D FFT kernel for GPUs using CUDA, SC '08; Proceedings of the 2008 ACM/IEEE conference on Supercomputing* (IEEE Press)
[6] In a broad sense, GPU is a graphics card. On the other hand, in a narrow sense, GPU means the processor in the graphic card. In this paper, we define GPU as the former broad sense.
[7] Harris M *GPGPU.org* http://gpgpu.org
[8] NVIDIA *CUDA™ Programing Guide*
[9] Hamada T, Iitaka T *The Chamomile Scheme An Optimized Algorithm for N-body simulations on Programmable Graphoics Processing Unit,* (astro-ph/0703100) http://uk.arxiv.org/abs/astro-ph/0703100
[10] Nguyen H 2007 *GPU Gems 3* (Addison-Wesley Professional)
[11] NVIDIA *CUDA ZONE* http://www.nvidia.com/cuda
[12] openSUSE project http://www.opensuse.org
[13] The g95 project http://www.g95.org
[14] PWscf project http://www.pwscf.org; Quantum ESPRESSO http://www.quantum-espresso.org
[15] Frigo M, Johnson S G *FFTW* http://www.fftw.org