Time benchmarks for the OpenMP and GPU parallelized calculation in the planewave pseudopotential density functional approach

Junpei Gotou,[a] Shinya Haraguchi,[a] Masahito Tsujikawa,[a] Tatsuki Oda[b]
[a]Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa 920-1192, Japan, E-mail: gotou@cphys.s.kanazawa-u.ac.jp
[b]Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan, E-mail: oda@cphys.s.kanazawa-u.ac.jp

Abstract. *We have investigated the computational performance of the first principles molecular dynamics code which employs planewave basis set and pseudopotential of electron-ion interaction. The efficiency of task has been measured in terms of matrix multiplication (MM), fast Fourier transformation (FFT). For the implementations of the OpenMP and GP-GPU utilities, the MM was found to mark the considerable high score, whereas the FFT a fairly good one. Analyzing the resulting whole performance, efficiency of the task other than the MM and FFT has to be improved for a higher performance.*

Keywords: Car-Parrinello molecular dynamics, first-principles electronic structure calculation, parallelization, OpenMP, GP-GPU

1 Introduction

Electronic properties in the realistic system with a nano-size scale have been becoming a target in the computational material science which explicitly includes effects on the quantum mechanics. Such a calculation needs a large amount of computational source. This is why the higher computational efficiency is required. The density functional approach has been playing an important role in material science. Indeed, the target extends to many topics associated with the increase of performance in both of hardware and software. The extension has been supported by parallel computations, which are performed in many levels of computer wares.

It is important to use the parallel calculation code which is matched with its architecture. Even when we develop the computational code for the density functional calculation, some tests for new architectures are required for grasping a future efficiency in the computation. Following the message passing interface (MPI), the OpenMP has been one of the central issues for the improvement in the current computational code. Recently, the general purpose graphic processing unit (GP-GPU) is also a choice for getting a higher performance. The test of hybrid calculations could also be meaningful.

In this work, we have investigated the performance about computational time in the pseudopotential planewave code for DFT calculation. The results on the matrix multiplication (MM) and fast Fourier transformation (FFT) are reported individually in detail in the implementations of the OpenMP and GP-GPU.

2 Computational circumstances

2.1 Properties of computer archtechture

We have used three kinds of machines for the performance investigation. Their properties are summarized in Table 1. The names of machines are referred in the text.

Table 1: Properties of the computers for investigation. The machine names are used in the text.

Machine Name	CPU	GPU	CUDA
M1	Quad-Core,AMD,2.3GHz	no device	no install
M2	Intel Xeon X5590,3.33GHz	Tesla C1060	CUDA Toolkit 2.3
M3	Intel Xeon X5680,3.33GHz	Tesla M2050	CUDA Toolkit 3.1 and 3.2

The OpenMP is a parallel programming language by which the memory is shared in a node of the machine. The message communication like MPI is unnecessary in the OpenMP calculation. It is also unnecessary to make a major correction when rewriting the single processing code to a parallelized one. However, the execution may not be secure even when there is no error. When using two or more threads in the OpenMP calculation, the correspondence between a temporal thread memory and real memory may not be secured. Therefore, it is required to make attension to the synchronization between them. The speed-up factor is used for presenting parallelization performances. This factor is estimated from the inversed ratio of the time by n's processors with respect to that by single one. Originally, GPU is a special set of many processors for graphical processes. The CUDA Toolkit developed by NVIDIA has become available to the computation for general purpose.

2.2 Properties of the target code

The calculation code used in the investigation is for the Car-Parrinello molecular dynamics (CPMD), which is based on the density functional theory (DFT). Details of the scheme can be found in the reference [1, 2]. The wave functions of dynamical variables are transformed by FFT between real and inverse spaces. In the ultrasoft pseudopotential scheme, the MM can be a considerable part for the time-consuming. In such a scheme of the DFT, the task of calculation is categorized to MM, FFT and the others. The code used can employ a scheme of noncollinear magnetism, affording to treat vector type spin density. Such scheme requires twice of memory for wave functions due to the two component spinor [3]. The starting code is parallelized in **k** points (sampling wave vectors in wave functions) with using MPI.

2.3 Sample systems

For investigating the computational time, the two sample systems have been considered. Both of them are slab systems; one is non-magnetic and the other magnetic with noncollinear scheme [3, 4]. Results of the former can be used for comparison with other codes and those of the latter may provide us the perspective to the application to the magnetic system with the spin-orbit interactions [5, 6]. The non-magnetic system, as shown in Fig. 1, consists of 13 monolayers (MLs) in $BaTiO_3$ with the inplane lattice constant ($a = 3.94$Å); 7 BaO MLs and 6 TiO_2 MLs (32 atoms in unit cell). In both sides of the slab, the vacuum layer with $d_{\rm v} = 5.29$Å is included. The periodic boundary condition is taken into account due to the planewave basis in which the energy cut-offs of 30Ry and 300Ry were used for the wavefunction and the density, respectively. The 6 special **k** points are used for MPI parallel computations. The magnetic system is a thin film Fe(5MLs)/MgO(6MLs)/Fe(5MLs) like a magnetic tunnel junction (22 atoms in unit cell, $a = 4.20$Å, the vacuum layer of $d_{\rm v} = 5.29$Å, the energy cut-offs of 30Ry and 300Ry, and 4 special **k** points).

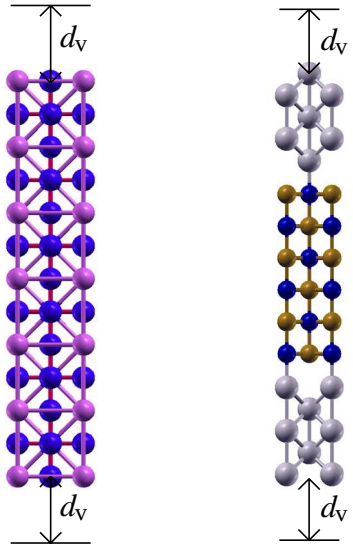

Figure 1: Non-magnetic and magnetic slab systems for the performance invesitigation; 13 monolayers (MLs) of BaTiO3 (left) and the magnetic tunnel junction of Fe(5MLs)/MgO(6MLs)/Fe(5MLs) (right). The d_v indicates the vacuum layer.

The parameters used in this work are reported in Table 2.

In these systems, the main parts for time consuming are MMs and FFTs. In the magnetic system, the MM and FFT needs the 70% and 20% of the total cpu time in the single processing calculation (without using MPI, OpenMP, and GPU). In this paper, we report the result in computational performance for OpenMP parallelization and GP-GPU computation.

3 Testings for the elements

3.1 Matrix multiplication (MM)

For the OpenMP parallelization, the MM is performed as shown in Fig. 2. Figure 3 shows parallelization efficiency for matrix multiplication. Using 8 cores, we obtain a high parallelization efficiency of 94%. It was confirmed that the MM can be parallelized with a high efficiency.

The MM can be performed using the routines of xGEMM (x =S,D,C,Z) for the typical matrices in the BLAS library. The CUDA Toolkit provides the library for the GPU. Figure 4 shows the FLOPS (floating point number operations per second) in the MM for the various matrix sizes [7]. The times of data transfer from the host computer to the device (GPU) and vice versa were included for the estimation. For the architecture of Tesla C1060, the performance is not high except for the single real matrices (SGEMM and CGEMM), whereas the performance becomes to a much higher one in Tesla M2050 even for DGEMM and ZGEMM. The performance has become to a high value in archiecture of Tesla M2050. The comparison with the performance of OpenMP (see the data of "8 thread" in Fig. 4) implies that the performance of CPMD becomes higher in the GPU of Tesla M2050 instead of the OpenMP.

Table 2: Characters and Parameters in sample systems.

Sample	BaTiO3	Fe/MgO/Fe
lattice	tetragonal lattice	tetragonal lattice
lattice constants	a=3.94Å, c=34.5Å	a=2.98Å, c=37.1Å
number of **k** point	6 special points	4 special points
number of atoms	32 (Ba:7, Ti:6, O:19)	22 (Fe:10, Mg:6, O:6)
number of Kohn-Sham orbitals	148	264
Energy Cutoff for wave function	25Ry	25Ry
Energy Cutoff for density	300Ry	300Ry
Fourier mesh for wave function	24×24×216	18×18×240
Fourier mesh for density	48×48×360	32×32×400
number of plane waves	7697	4731

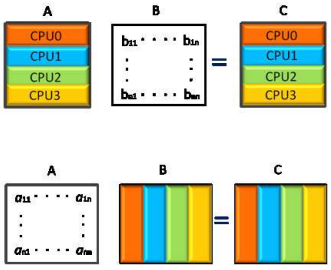

Figure 2: Parallelization schemes of matrix multiplication (MM) in the OpenMP parallelization.

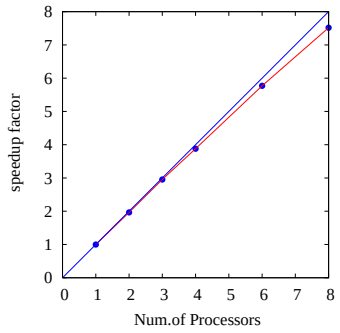

Figure 3: The parallelization efficiency (speed-up factor) of the matrix multiplication (MM) in the OpenMP of M1 for the size of 5000×5000.

In the GPU calculation, the data transfer (host to device and vice versa) is time-consuming. The typical ratios are presented in Fig. 5. In the larger sizes of matrix, the ratios of data transfer become suppressed. This implies the efficiency of GPU in the application of larger systems. It is interesting to see that the transfer from the host (CPU) to the device (GPU) needs more time compared with that from the device to the host. Such feature of results is attributed to the amount of data transfer and the bandwidth for up- and down-loading between the host and device.

3.2 Fast Fourier transformation (FFT)

In the target code, the physical value (wave functions and densities) is efficiently calculated by using one of real and reciprocal spaces [2]. Requiring the conversion between them, three dimensional FFT (3D-FFT) is used. The parallelization scheme of OpenMP for 3D-FFT is shown in Fig. 6. In this, the 3D-FFT is divided to three times of 1D-FFTs so that the number of calculation tasks is increased and we obtain the good load balance among the threads associated with the OpenMP. This sometimes works well because in the application systems of typical size of CPMD calculation, the size of FFT is not enough large. Such sizes are only several times or several score times of the

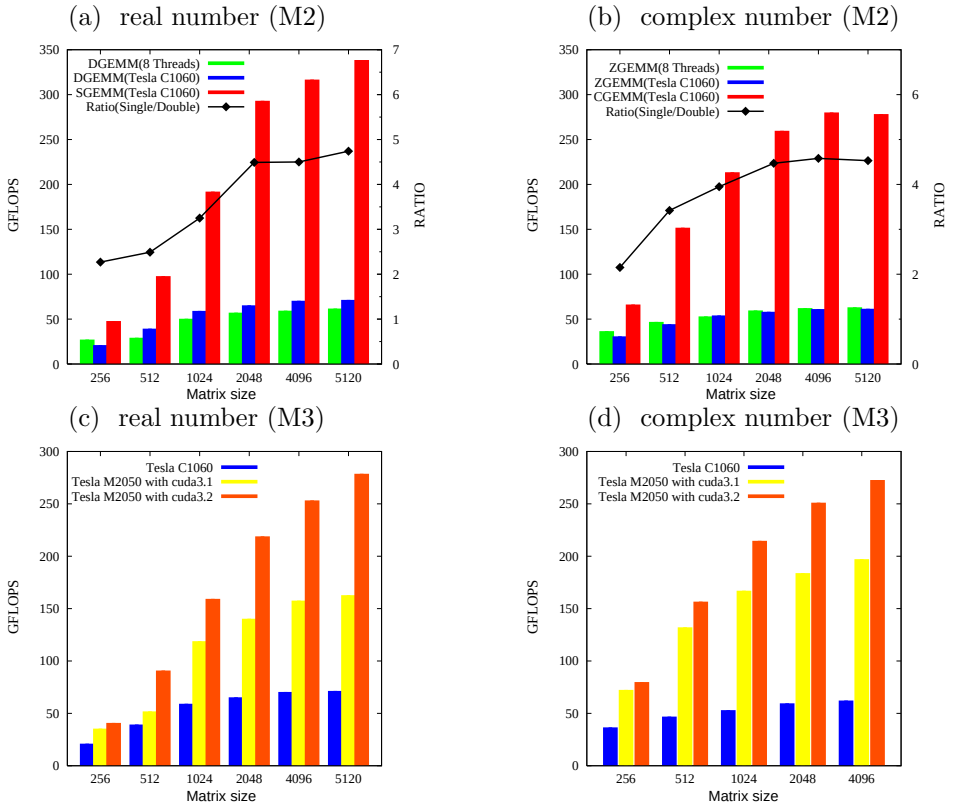

Figure 4: Effective FLOPS values of the MM for various matrix sizes (256 $\sim$ 5120) by SGEMM, CGEMM, DGEMM and ZGEMM routines in Tesla GPUs (Tesla C1060 and Tesla M2050). The upper and lower two panels are obtained in the machines of M2 (a)(b) and M3 (c)(d) (see Table 1), respectively. The times for memory transfer are included in the estimation.

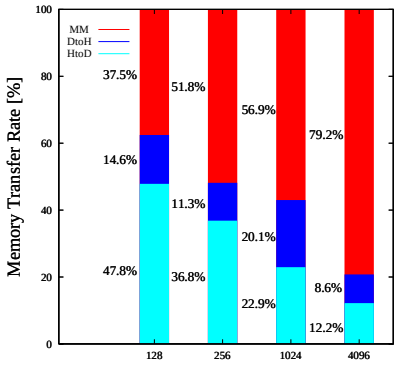

Figure 5: Data transfer ratio in DGEMM calculation for various matrix sizes in M3.

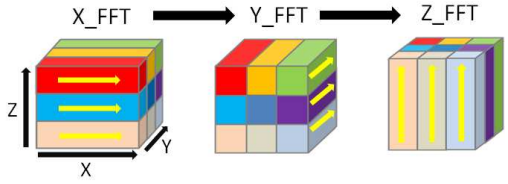

Figure 6: Parallelization scheme for three dimensional fast Fourier transformation (3D-FFT) in the OpenMP.

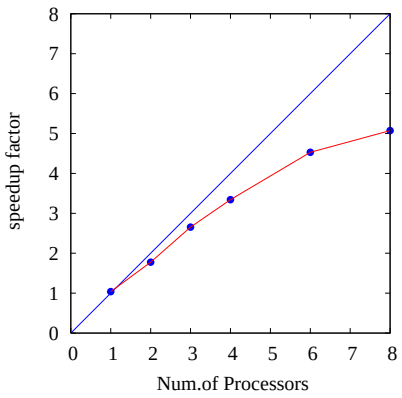

Figure 7: The parallelization efficiency (speed-up factor) of 3D-FFT in the OpenMP. The size of $48 \times 48 \times 216$ is used.

number of available threads in OpenMP.

The parallelization efficiency in OpenMP is presented in Fig. 7. Using 8 cores, we obtain the efficiency of 65%. The parallelization efficiency has decreased compared with the MM (see Fig. 3). This may be because in the scheme the temporal memory for 1D-FFT, as the overhead of calculation, is stored discontinuously from the original memory area of 3D-FFT. As another investigation for 3D-FFT, the automatic parallelized version of FFTW [8] in OpenMP are tested (not reported here), resulting the parallelization efficiency of 25% for using 8 cores.

In this work, we have carried out the test of CUFFT for 3D-FFT. The computational times are presented in Fig. 8. The efficiency is found to depend on the size of FFT (spiky structure in the figure). There are many sizes where the efficiency is worse than that of FFTW [8]. This implies the 3D-FFT by CUFFT is not appropriate for the CPMD code in which the size of FFT depends on the input parameters.

4 Results on CPMD code

4.1 MPI+OpenMP (Hybrid)

The hybrid version of code is organized with **k** point parallelization on MPI, the OpenMP calculation both in MM and FFT, and other minor parallelizations. The efficiencies are presented with those of the MPI version in Fig. 9. Using 16 cores (2 MPI+ 8 OpenMP), the performance amounts to 37%. This value is still fearly well, but the additional improvement must be required for an more efficient calculation.

4.2 MPI+GPU

The MPI+GPU version of CPMD code has been checked mainly in M2, which has 8 cores and 3 GPUs. The $BaTiO_3$ system is used for the computation. The time report is presented in Figs. 10 and 11. In the latter, the times for MM, FFT are included. In these figures, the labels of BLAS and FFTW indicate the single processing calculation for the MM and the FFT, respectively. The

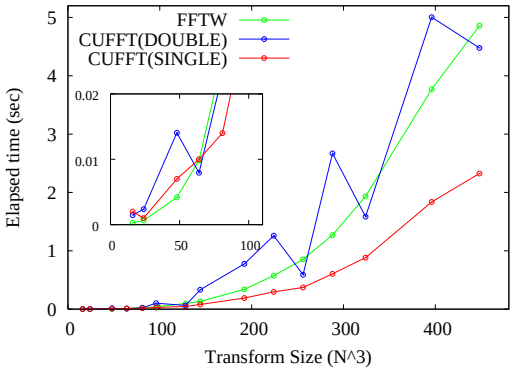

Figure 8: The computational times with respect to the various sizes for the CUFFT routine of 3D-FFT (GPU) in M2.

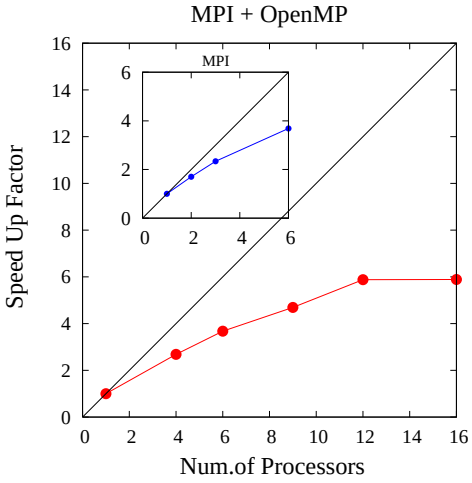

Figure 9: Parallelization efficiency for the MPI and Hybrid versions for $BaTiO_3$ system.

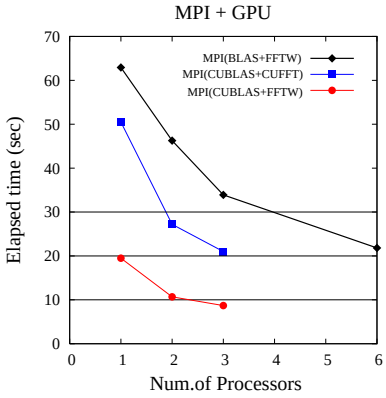

Figure 10: Conputational times (single CPMD iteration in second) with respect to the number of MPI threads for MPI+GPU (Tesla C1060) version. The single MPI thread can access to a single GPU in M2.

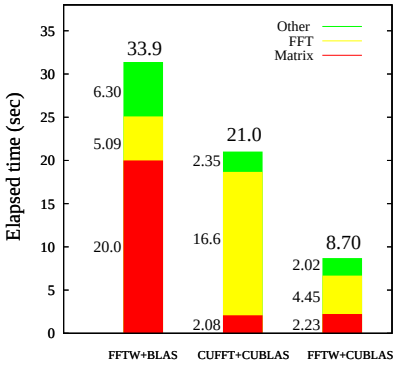

Figure 11: Conputational times (single CPMD iteration in second) of MM, FFT, and others for $BaTiO_3$ by MPI+ GPU (Tesla C1060) version (3MPI+3GPU).

GPU calculation for the MM (CUBLAS) is improved considerably, whereas the FFT by CUFFT is more time-consuming than the FFTW, as expected from the result in Fig. 8.

4.3 MPI+OpenMP+GPU

In the experience which is encountered in the previous sections, the FFT by GPU cannot be carried out so high performance. Taking into account this, the MPI+OpenMP+GPU (hybrid+GPU) version is organized with the MPI (**k** point parallelization), the MM by GPU, the FFT by OpenMP (6 threads in OpenMP), and other minor parallelization. This is tested in the magnetic system (Fe/MgO/Fe) with M3 (4 threads in the MPI and 2 GPUs). The result with the CUDA Toolkit

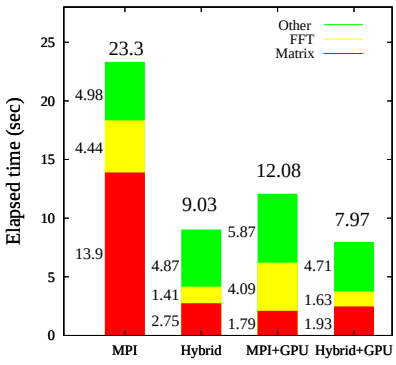

Figure 12: Computation times (single CPMD iteration in second) for the slab of Fe(5MLs)/MgO(6MLs)/Fe(5MLs) in M3.

3.1 is represented in Fig. 12. In the MPI+GPU calculation, the FFTW is used in the 3D-FFT for each **k** point sampling. The hybrid+GPU calculation marks the highest score among the versions investigated. By introducing OpenMP and GPU, the time ratio of MM has become to 24% from 70% in the single processing calculation and from 60% in the calculation. If preparing one GPU per MPI thread, instead of one GPU per two MPI threads, the MM can be accelarated. Introducing the MPI, OpenMP, and GPU to the CPMD code, the computation other than the MM and FFT becomes to the major part of time-consuming. It should be improved in future.

5 Conclusions

We have investigated computational performance of the CPMD code for the OpenMP parallelization and the GPU architecture in addition to the MPI parallelization. Our test has succeeded to reduce the computing time considerably, compared with the MPI version for the magnetic slab system which is an accesible model for typical magnetic tunnel junction. Based on such development of the code which accords with the new architectures, we may approach electronic structures and molecular dynamics for larger realistic systems with the high performance computation.

Acknowledgment

One of authors (T.O.) would like to thank the Japan Society for the Promotion of Science (JSPS) for financial supports (Grant 22104012, 22360014, and 22340106). One of authors (M.T.) acknowledges the JSPS Research Fellowships (Grant No.20-6647) for Young Scientists.

References

[1] D. Vanderbilt (1990). Soft self-consistent pseudopotentials in a generalized eigenvalue formalism, *Phys. Rev. B*, **41**, 7892-7895.

[2] K. Laasonen, A. Pasquarello, R. Car, C. Lee, and D. Vanderbilt (1990). Car-Parrinello molecular dynamics with Vanderbilt ultrasoft pseudopotentials, *Phys. Rev. B*, **47**, 10142-10153.

[3] T. Oda, A. Pasquarello, and R. Car (1998). Fully Unconstrained Approach to Noncollinear Magnetism: Application to Small Fe Clusters, *Phys. Rev. Lett.*, **80**, 3622-3625.

[4] T. Oda and A. Hosokawa (2005), Fully relativistic two-component-spinor approach in the ultrasoft-pseudopotential plane-wave method, *Phys. Rev. B*, **72**, 224428(1-4).

[5] K. Sakamoto, T. Oda, A. Kimura, K. Miyamoto, M. Tsujikawa, A. Imai, N. Ueno, H. Namatame, M. Taniguchi, P. E. J. Eriksson, R. I. G. Uhrberg (1997). Abrupt Rotation of the Rashba Spin to the Direction Perpendicular to the Surface, *Phys. Rev. Lett.*, **102**, 096805(1-4).

[6] M. Tsujikawa and T. Oda (2009). Finite Electric Field Effects in the Large Perpendicular Magnetic Anisotropy Surface Pt/Fe/Pt(001): A First-Principles Study, *Phys. Rev. Lett.*, **102**, 247203(1-4).

[7] http://www.softek.co.jp/SPG/Pgi/TIPS/public/accel/cublas-matmul.html

[8] Matteo Frigo and Steven G. Johnson, "manual for FFTW 3.2.2" July 2009, http://www.fftw.org/.