Real-time and real-space program tuned in K-computer

Y. Zempo\(^1\), N. Akino\(^2\), M. Ishida\(^2\), E. Tomiyama\(^3\) and H. Yamamoto\(^3\)

\(^1\) Computer and Information Sciences, Hosei University, 3-7-2 Kajino, Koganei, Koganei 184-8584 Japan
\(^2\) Sumitomo Chemical, Co., Ltd., 6 Kitahara, Tsukuba 300-3264 Japan
\(^3\) Research Organization for Information Science and Technology, 1-5-2 Minatojima-Minami, Chuo, Kobe 650-0047 Japan

E-mail: zempo@hosei.ac.jp

Abstract. We have transported the code from Earth Simulator (ES) to K-Computer (KC) keeping a scalability. The base code achieved remarkable linear scalability and significant parallelization efficiency in EC before. The reason of success is that, taking into account the architectural differences between ES and KC, we improve the program step by step. Actually, the program showed a 37% improvement in the total performance with the implementations such as hybrid OpenMP and MPI parallelization, and the optimal rank mapping, and the MPI communication concealment, including removing the unnecessary processes such as zero clear and data copy.

1. Introduction

Time-dependent density functional theory (TDDFT) is one of the most prominent and widely used methods for calculating excited states of medium-to-large molecules, and it is recognized as a powerful tool for studying electronic transitions of molecules [1]. In our calculations, the real-time and real-space technique is adopted in solving equation by the finite difference approach [2]. In particular, the real-space approach is suitable for large-scale parallel computing, and also allows the capture of a clear physical image. Within the framework of this approach, we can solve for the wavefunctions on the grid with a fixed domain, which encompasses the physical system of interest [3].

We have transported the code from the first generation Earth Simulator (ES) to K-Computer (KC). There are architectural differences between ES and KC: In ES each node has eight vector processors, and there is a high speed inter and intra-node communication; In KC, each nodes has eight cores per processor and each node connected by a special interconnect. Taking account of these differences, we have applied the following implementations and obtained the 37% better total performance.

First, we have stated with OpenMP and MPI hybrid parallelization, and optimal rank mapping, where the \(zyx\)-mapping is adopted in the 3D parallelization. It is fairly effective to get rid of the unnecessary processes such as zero clear and data copy. In calculation loops, the improvement of thread efficiency provides an efficient OpenMP process. Introducing the MPI test call, communication concealment in MPI processes works effectively in the code. It takes a role of a trigger for MPI send and MPI recv processes. The detail will be introduced in the presentation.
The paper is organized as follows. The base code and the numerical detail is presented in the next section. The parallelization techniques and their achievements are shown in section 3. Finally, we summarize and discuss our conclusions in the last section.

2. Base code and numerical details

2.1. Time-dependent density-functional theory

In this section, we briefly present our TDDFT calculation procedure. One of the most successful ways of obtaining the electronic structure from the first-principles is the use of computational approaches based on the density functional theory (DFT) [4] with the local density approximation (LDA). The total energy of the ground state can be derived from the Kohn-Sham equation (KS) [5]. Although, for the excited states, it is much less successful in describing the optical responses and the excitation spectra, this difficulty is, in principle, solved by the extension of DFT to the time-dependent theory. Its foundation was established by Runge and Gross [1]. In analogy to the time-independent case, the TDDFT equation of motion coupled with pseudopotentials is given by

\[
\left\{ -\frac{1}{2} \nabla^2 + V_{\text{ion}}^{\text{ps}}(\mathbf{r}) + V_H(\mathbf{r}, t) + V_{\text{XC}}[\rho(\mathbf{r}, t)] + V_{\text{ext}}(\mathbf{r}, t) \right\} \psi_i(\mathbf{r}, t) = i \frac{\partial}{\partial t} \psi_i(\mathbf{r}, t),
\]

where \( V_{\text{ion}}^{\text{ps}} \) is an ionic pseudopotential, \( V_H \) is the Hartree potential, and \( V_{\text{XC}} \) is the exchange-correlation potential. Since the exact time-dependent xc kernel is not known, the originally non-local time-dependent xc kernel is replaced with a time-independent local one. This is considered as reasonable when the density varies slowly with time. This approximation allows the use of a standard local ground-state xc functional in the TDDFT framework. The Hartree and exchange-correlation potentials can be determined from the electronic charge density, \( \rho(\mathbf{r}, t) = \sum_i |\psi_i(\mathbf{r}, t)|^2 \). The summation is over all occupied states \( i \). The Hartree potential is determined by \( \nabla^2 V_H = -4\pi \rho \), and as the xc potential \( V_{\text{XC}} \), the usual local density approximation (LDA) is used in our study. For the ionic potential, we employ the pseudopotential \( V_{\text{ion}}^{\text{ps}} \) in the separable form so that the only valence electrons are considered [6]. Prior to the calculation of optical responses, we first solve the usual, time-independent formulation of the pseudopotential-DFT method [7] to obtain the optimized electronic structure [2]. Then, we apply an external field \( V_{\text{ext}} \) to the system as a perturbation and follow the linear responses of the system in real time.

This method is effectively used for the cases where the potential is time-dependent, e.g. the time-dependent behaviors of electrons in oscillating electric and magnetic fields, and excited-state reactions. Empirically, we know that these can be described fairly well by TDDFT. Although the xc functional we adopted has been developed with respect to the electronic ground states, we have also employed in our practical calculations.

In our calculations, the real-time and real-space technique is employed in solving eq.(1) by the finite difference approach [8] without using explicit bases such as plane waves and gaussians. Within the frame work of this approach, we can solve for the wave functions on the grid with a fixed domain, which encompasses the physical system of interests. The uniform grid is used in our study for the simplicity.

The wave functions are evolved by the time evolution operator, \( \psi(t) = \exp[-iHt] \psi(0) \), with the initial wave function at \( t = 0 \),

\[
\psi_i|_{t=0} = e^{ik_zz_i}\psi_i(0),
\]

where \( H \) is the Hamiltonian of the system, and \( k_z \) is a small wave number corresponding to the external perturbation in the \( z \) direction. In the linear response, the time-dependent polarizability is proportional to the dipole matrix element \( \mu(t) = \langle \psi(t)| \sum_i z_i |\psi(t)\rangle \), where \( z_i \) with \( i = 1, 2 \) and
3 represents $x$, $y$ and $z$, respectively. The frequency-dependent polarizability in the $z$ direction $\alpha_z(\omega)$ is then obtained as the time-frequency Fourier transformation of $\mu_z(t)$,

$$\alpha_z(\omega) = \frac{1}{k} \int dt e^{-i\omega t} \mu_z(t). \tag{3}$$

The polarizability $\alpha(\omega)$ is given by the orientation average, $\alpha = (\alpha_x + \alpha_y + \alpha_z)/3$. The optical strength function, $S(\omega)$, is related to the imaginary part of the polarizability,

$$S(\omega) = \frac{2\omega}{\pi} \text{Im} \alpha(\omega). \tag{4}$$

2.2. Achievement of the base code running on ES

Base code was already tuned in the first generation Earth Simulator using MPI, and showed a good linearity in Fig. 1, which was applied to the calculation of fluoren six multimer. It performed at about 8 TFlops using 243 nodes in ES.

The ES is consisted of 640 processor nodes (PNs) connected by $640 \times 640$ single-stage crossbar switches, which control with inter-node data communications at 12.3 GB/s bidirectional transfer rate [9]. Thus the total bandwidth of inter-node network is about 8 TB/s. In each node, There were eight vector processors, which has the memory bandwidth at 256 GB/s. All CPUs in 640 nodes in ES seemed flatly connected. When we tuned our code in ES, we only paid an attention to usual vectorization for vector processors, and simple MPI libraries in the parallelization.

![Figure 1](image-url). Scalability of the base code in ES. RS means the ground stat calculation, before the time evolution calculation RS. Almost linear scaling was observed up to 243 nodes.

3. Tuning detail

The system uses scalar CPUs using SPARC64 VIIIfx, which has 8 cores and shows the performance 128 Gflops. The memory bandwidth is 64 GBs, which is fairly small to the CPU performance, compared to that of ES. To achieve high performance and high scalability to this ultra-large-scale system, the CPU additionally supports SIMD processing [10].

In this section, we would like to focused on those which significantly work for improve our calculation code, although there are, of course, other various interesting features involved in the system. One of them is its network topology called Tofu.
3.1. MPI and OpenMP hybrid
As the basis of our parallelization, we have stated with OpenMP and MPI hybrid parallelization. In each node, we used OpenMP. We have set up a sample run, which takes 510 s without taking into account I/O. Our tuning results are summarized in Fig.5. Each technique we adopted to improve our code is described in the following subsections.

3.2. MPI rank mapping
This thesis focuses on techniques of task mapping for solving problems on parallel computers with many processors on multi-dimensional networks like K-computer. The application problems can be decomposed to subtasks with the computational load. Our real space calculation is usually divided by the xyz directions into a rectangle shape of a group of meshes. According to the direction, we may use several groups which can be mapped to MPI ranks. This MPI rank mapping should be effectively adjust to the calculation loop in practical program, which is different to the natural order of the direction. In our optimization, the zyx rank mapping was significantly worked. It reduces 13% the calculation time, which is 75 s from 510 s mentioned in the previous section.

3.3. Removing unnecessary zero-clear
Traditionally, arrays are zero-cleared prior to MPI communications because of the safety in the data. In Fortran90, it is too easy to write this memory swipe with one line. Sometimes, one is not aware of this type of simple wrong implementation. However, the large array may be handled on the fairly small bandwidth ration. It takes a large time to realize it. Removing this type of zero clear process from the algorithm, we can obtain the program, which runs 8 % faster (40 s shorter) than above.

3.4. Software operational optimization
In this section, we would like to mention about the software optimization such as redundant arithmetic calculation, getting rid of valuables from several loops, which can be considered as constant valuables inside, and replace the directive of OpenMP to work OpenMP thread to be optimized. In particular, we have introduce the specific CPU feature, i.e. called the SECTOR cache. It is a selectable mechanism in the user’s program for reusable data. The SECTOR cache can be divided into two sectors. Reusable data by allocated on the SECTOR1 is prevented to be replaced on the cache memory. With these improvements, we reduced 46 s (9 %) faster.

![Image of using SECTOR cache. (a) Initial stage of both SECTOR caches, (b) The cache SECTORs are getting filled with data, but SECTOR1 is keeping the data. (c) When SECTOR0 is completely filled with data, the data in SECTOR0 will be swiped out. On the other hand, the data allocated in SECTOR1 is not affected by SECTOR0.](image)

3.5. MPI communication concealment
MPI concealment is expected to make the parallel calculation efficient. During the operation, MPI_send and MPI_receive, the other MPI process is expected to work. However, this kind
of operation, we need some trigger. Otherwise, the communication is still waiting for the next MPI_isend and MPI_ireceive call. To realize the concealment, we have place a MPI_test call each different type of MPI_isend or MPI_ireceive call. With this MPI communication concealment, the code runs 2.3% faster before. It actually corresponds to the reduction of 12 s.

![Image](image_url)

**Figure 3.** Image of MPI communication concealment. MPI_test will be a trigger to start the background communication.

### 3.6. Removing unnecessary copies

The memory copies is practically not a big matter in ES. However, it quite affects the calculation in K-computer, which has relatively small bandwidth. In this type of computer system, the leapfrog calculation in the time evolution provides significant effects. It is also related the memory access in Sec.3.3. It will be a trend in the recent parallel computation using huge number of CPUs, in which the interconnect network is getting multiple-stage. In the improvement shown in Fig.4, we removed unnecessary copies in time evolutions. Thus, we made up for the lost time 25 s (4.9%). Including the previous Sec.3, we have totally 37% improvement. It corresponds to

![Image](image_url)

**Figure 4.** Time evolution scheme with leapfrogging. Calculated data is not necessary to be copied for the next step.

the reduction of the calculation time from 510 s to 322 s, as summarized in Fig. 5.

### 4. Concluding remarks

We transported our code from ES to K-computer with fairly good efficiency. There are several architectural differences between ES and K-computer system. Some of them are due to the memory bandwidth and the operational optimization. In the trend of the large scale computer system, in particular, many CPUs are required to realize HPC (High Performance Computing)
in various scientific fields. Then, the more larger number of CPUs, the system might be the smaller memory bandwidth. In the parallel programming, we have to take care of this tendency.

![Graph](image)

**Figure 5.** Tuning techniques and the performance of the code. (Left) The tuning technique and its effect. (Right) Scaling with the increase number of the nodes. The performance is measured by the results using 128 nodes.

**Acknowledgments**
The authors would like to express their gratitude to Professor K. Yabana for providing his code as our base code and also for useful discussions and encouragement. This work was partially supported by JSPS Grants-in-Aid for Scientific Research (C) Grant number 25390158, Sumitomo Chemical Co., Ltd. and Simulatio Corporation.

**References**
[1] Runge E and Gross E K U 1984 *Phys. Rev. Lett.* **52**, 997
[2] Yabana K and Bertisch G F 1996 *Phys. Rev.* **B54**, 4484
[3] Zempo Y, Akino N, Ishida M, Ishitobi M and Kurita Y 2008 *J. Phys. Condens. Matter* **20**, 064231
[4] Hohenberg P and Kohn W 1964 *Phys. Rev.* **136**, B864
[5] Kohn W and Sham L. J 1965 *Phys. Rev.* **140**, A1133
[6] Troullier N and Martins J L 1991 *Phys. Rev.* **B43**, 1993
[7] e.g. Pickett W E 1989 *Comput. Phys. Reports* **9**, 115
[8] Chelikowsky J, Troullier N, Wu K and Saad Y 1994 *Phys. Rev. B* **50**, 11355
[9] http://www.jamstec.go.jp/es/jp/es1/system/index.html
[10] http://www.aics.riken.jp/en/k-computer/system