Tuning of Density Functional Theory Simulation on Vector Processor System - Plasma Simulator Raijin -

Atsushi M. ITO, Arimichi TAKAYAMA, Osamu WATANABE[1)], Vijendra SINGH[2)], Shubham TYAGI[2)] and Shashank S. SINGH[2)]

National Institute for Fusion Science, National Institutes of Natural Sciences, 322-6 Oroshi-cho, Toki 509-5292, Japan
[1)]*NEC Corporation, 5-7-1 Shiba, Minato-ku, Tokyo 108-8001, Japan*
[2)]*NEC Technologies India Private Limited, Plot No. 20 & 21, Sector 135, Nodia, U.P.-201304, India*

(Received 29 September 2020 / Accepted 19 October 2020)

We rapidly report the benchmark of density functional theory simulation on the Plasma Simulator composed of vector processor. The improved OpenMX code on the SX-Aurora TSUBASA of the vector processor achieved performance higher than the Intel Xeon Gold of the scalar processor.

Keywords: simulation, vectorization, density functional theory

DOI: 10.1585/pfr.15.1203085

Plasma Simulator Raijin, which is the super computer system in National Institute for Fusion Science (NIFS), has vector processors and high bandwidth memory composed of the NEC SX-Aurora TSUBASA (SX). The super computer system achieved 10.5 PFlops. The vector processors and the high bandwidth memory are a better solution for field simulation with grid data such as magnetohydrodynamics (MHD) simulation, and for large scale particle simulation such as particle in cell (PIC) simulation.

On the other hand, in terms of simulation for reactor material, there is insufficient information regarding the effect of the vector processors. Therefore, in the present work, we report the benchmark of the density functional theory (DFT) [1] simulation on SX.

OpenMX [2] code of version number 3.9.2 was used for the benchmark of DFT simulation in the present work. Moreover, we have improved in terms of vectorization for SX. Note that there are several DFT codes based on several algorithms which reflect the kinds of electron basis functions. The OpenMX employs the pseudo-atomic localized orbitals as electron basis functions [3, 4].

As a typical benchmark target of researchers on plasma-material interaction and fusion reactor material, we treat a tungsten material composed of 127 tungsten atoms which corresponds to $4 \times 4 \times 4$ unit cells of body-centered cubic lattice structure having a mono-vacancy. The simulation box has three dimensional periodic boundary conditions. The Brillouin zones were sampled with $4 \times 4 \times 4$ k points. To achieve uniform calculation amounts, self-consistent field (SCF) iterations are stopped at 33 steps.

The comparison target is the JFRS-1 at the Computational Simulation Centre of International Fusion Energy Research Centre (IFERC-CSC) composed of Intel Xeon Gold 6148 (Xeon), which is a scalar processor. A unit to compare the amount of computer resources is selected as a node. One node in Plasma Simulator is one vector engine (VE) of SX which has 8 cores. One node in the JFRS-1 is two sockets of Xeon which has 20 cores in each socket. Based on the theoretical performance, the SX has 2.433 TFlops per node, while the Xeon has 2.816 TFlops per node in the case of using the AVX-512 operation. Therefore, a target performance ratio defined as the ratio of calculation speed of SX to calculation speed of Xeon is expected at 0.86 (=2.433/2.816) from the theoretical performance.

Figure 1 shows the calculation time of the present benchmark. First, the calculation performance on SX was improved by the present vectorization. Next, from the calculation time, the actual performance ratio of SX to Xeon is estimated at 0.93 with 8 nodes and 1.16 with 16 nodes, which are higher than the target performance ratio, 0.86.

We describe the example of the most effective vectorization in the present tuning of OpenMX. A dominant part of the calculation time is the integration of wave functions and quantum operators having a loop structure of the fol-

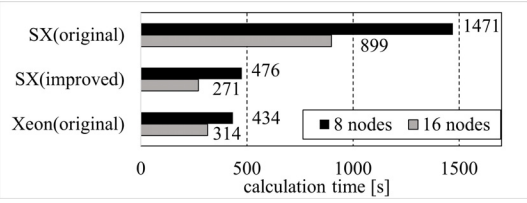

Fig. 1 Calculation times of OpenMX on the NEC SX-Aurora TSUBASA (SX) with original code and improved code, and the Intel Xeon Gold 6148 (Xeon) with original code.

author's e-mail: ito.atsushi@nifs.ac.jp

lowing example code.

```
/* example 1 */
#pragma omp for
for (int g = 0; g < Ng; ++g){
  // .. routine depending on g.. //
  for (int i=0; i < Ni; ++i){
    for (int j=0; j < Nj; ++j){
      // .. routine depending in i, j ..//
      target[g][i][j] = value_ijg;
    }}}
```

The loop variable g corresponds to the space grid and its loop length Ng is greater than several thousand, while the loops for the variables i and j correspond to the indices of the atomic orbitals on each atom. Numbers of atomic orbitals Ni and Nj are approximately 15 to 30 in the case of metals, and their values are not determined before the run-time of simulation. Note that, in the actual code, the variable g is often the index for indirect access to a grid point, and the loops of variables i and j are split for more multi-nested loops for variables such as principal quantum number, azimuthal quantum number, magnetic quantum number, spin quantum number. In addition, there are loops for atoms outside of the example structure.

In the original version of OpenMX which had been optimized for scalar processor, the long loop for variable g is located outside of the short loops for variables i and j . The long loop for variable g is parallelized by OpenMP thread. On the other hand, the short loops for variables i and j are sometimes vectorized by Single Instruction/Multiple Data (SIMD).

However, the loop structure of the example 1 is not appropriate for vector processor. The vector length of SX, 256, is much longer than the vector length of SIMD, 8 or less. For this reason, on SX, it is recommended to vectorize the long loop for the variable g . Because the loop to be vectorized must be placed on the innermost side, the position of loops are replaced from the example 1.

The simple way of vectorization for the SX is to change the loop structure as follows.

```
/* example 2 */
for (int i=0; i < Ni; ++i){
  for (int j=0; j < Nj; ++j){
    // .. routine depending in i, j ..//
    #pragma _NEC vector
    for (int g = 0; g < Ng; ++g){
      // .. routine depending on g.. //
      target[i][j][g] = value_ijg;
    }}}
```

Such vectorization had actually improved computational speed in most parts in OpenMX. However, the performance ratio was not higher than the target performance ratio. The reason why performance is not high is that the calculation cost of the routines depending on variable g increased. For example, the calculation cost of the routines depending on variable g in the example 2 is Ni × Nj times greater than that in the example 1.

Furthermore, to suppress the calculation cost, we attempted to unroll the short loop for variable j at the inside of the vectorized loop for variable g . However, the loop length Nj is not determined until run-time. Therefore, we employed the following loop structure.

```
/* example 3 */
if(Nj <= 30) {  // "30" is magic number //
  for (int i=0; i < Ni; ++i){
    // .. routine depending in i .. //
    #pragma _NEC vector
    for (int g = 0; g < Ng; ++g){
    // .. routine depending on g .. //
      #pragma _NEC unroll(30)
      for (int j=0; j < 30; ++j){
        if(j < Nj){
          // .. routine depending on j .. //
          target[i][j][g] = value_ijg;
        }}}}}
else{
  // The example 1 or 2 is described here//
}
```

In this example, the calculation path is divided by the loop length Nj . An important point is that in the former calculation path, the length of the innermost loop for the variable j becomes the constant value, 30. This short loop with constant length is completely unrolled. Consequently, the long loop for the variable g has no inner loop and can be certainly vectorized. The magic number of 30 was chosen because we knew that Nj was less than 30 in almost cases, empirically. In our experience, it is appropriate that the magic number is smaller than or same order as the square root of the hardware vector length of 256. As a result, the improved OpenMX achieved the present performance.

Finally, we can show good performance of DFT simulation on Plasma Simulator Raijin. OpenMX still has room for optimization. In addition, for researchers on plasma-material interaction and reactor materials, simulation methods other than DFT should be optimized for Plasma Simulator Raijin. We will report them in future.

The present work was supported by KAKENHI (19H011882, 19K21870, and 19H01874). The benchmark calculation was performed on “Plasma Simulator” (NEC SX-Aurora TSUBASA) of NIFS with the support and under the auspices of the NIFS Collaboration Research program (NIFS20KNSS130) and the JFRS-1 supercomputer system at IFERC-CSC in Rokkasho Fusion Institute of QST (Aomori, Japan).

[1] W. Kohn and L.J. Sham, Phys. Rev. **140**, A1133 (1965).
[2] T. Ozaki and contributors, http://www.openmx-square.org
[3] T. Ozaki, Phys. Rev. **B 67**, 155108 (2003).
[4] T. Ozaki and H. Kino, Phys. Rev. **B 69**, 195113 (2004).