Massively Parallel Algorithm and Implementation of RI-MP2 Energy Calculation for Peta-Scale Many-Core Supercomputers

Michio Katouda,[a] Akira Naruse,[b] Yukihiko Hirano,[b] and Takahito Nakajima*[a]

A new parallel algorithm and its implementation for the RI-MP2 energy calculation utilizing peta-flop-class many-core supercomputers are presented. Some improvements from the previous algorithm (J. Chem. Theory Comput. 2013, 9, 5373) have been performed: (1) a dual-level hierarchical parallelization scheme that enables the use of more than 10,000 Message Passing Interface (MPI) processes and (2) a new data communication scheme that reduces network communication overhead. A multi-node and multi-GPU implementation of the present algorithm is presented for calculations on a central processing unit (CPU)/graphics processing unit (GPU) hybrid supercomputer. Benchmark results of the new algorithm and its implementation using the K computer (CPU clustering system) and TSUBAME 2.5 (CPU/GPU hybrid system) demonstrate high efficiency. The peak performance of 3.1 PFLOPS is attained using 80,199 nodes of the K computer. The peak performance of the multi-node and multi-GPU implementation is 514 TFLOPS using 1349 nodes and 4047 GPUs of TSUBAME 2.5. © 2016 Wiley Periodicals, Inc.

DOI: 10.1002/jcc.24491

Introduction

Quantum chemical calculation is a reliable tool for studying chemical phenomena of molecules. The importance of massively parallel computation for the field of computational quantum chemistry increases year-by-year because of the necessity of applying quantum chemical calculations to nanoscale molecules and biological molecules.[1] In 2012, the RIKEN Advanced Institute for Computational Science and Fujitsu developed a massively parallel multi-core supercomputer: the K computer[2,3] was designed to achieve theoretical peak performance of 11 Peta FLoating Operations Per Second (PFLOPS). The K computer is a distributed-memory supercomputer consisting of 88,128 nodes and 705,024 central processing unit (CPU) cores of SPARC64 VIIIfx (8 cores/node) and connecting with a six-dimensional mesh/torus network called "Tofu." In June 2011, the K computer was ranked at the top of the TOP500 supercomputer list,[4] with peak performance of more than eight PFLOPS in the LINPACK benchmark. In November 2011, it became the first supercomputer with peak performance exceeding 10 PFLOPS. The K computer has been used for scientific studies such as earthquake and tsunami simulations[5] for protection from disasters, atmospheric simulations[6] for weather forecasting, simulations of biological molecules[7] for drug discovery, simulations of material design,[8,9] and simulations of galactic motion.[10] Among them, the first-principles density functional theory (DFT) calculation of nanomaterials in 2011[8] and the N-body simulation of the galaxy in 2012[10] earned Gordon Bell Prizes. Our research team is developing NTChem[11,12] software for the high-performance computing of molecular electronic structures. NTChem includes implementations of first-principles molecular electronic structure theories of various kinds (generally known as quantum chemistry theory) such as Hartree–Fock (HF) theory,[13] DFT,[14] Møller–Plesset perturbation theory,[15] coupled-cluster theory,[16] and the quantum Monte Carlo method.[17–19]

The second-order Møller–Plesset perturbation (MP2) method[15] is the simplest method for calculating electron correlations based on many-body perturbation theory. In the MP2 method, the perturbation expansion of the total energy is truncated at second order. It then becomes the simplest approximation to treat the correlation of two electrons. To evaluate the MP2 correlation energy, the four-center molecular orbital (MO) two electron repulsion integrals (2-ERIs) are needed and are evaluated by the four-index transformation of four-center atomic orbital (AO) 2-ERIs. This calculation requires $O(N^5)$ of high computation costs, unlike HF and DFT calculations which require $O(N^4)$ costs where N is the number of atoms in a molecule. Furthermore, $O(N^3)$–$O(N^4)$ of memory or disk space storing intermediate data during the four-index transformation is demanding. These computational bottlenecks hinder the MP2 calculation of large nanoscale molecules.

To overcome this problem, various efficient computation schemes in the MP2 framework have been developed. Among them, the resolution-of-identity MP2 (RI-MP2) method[20,21] is a promising method. In the previous study, the authors reported

[a] M. Katouda, T. Nakajima
Computational Molecular Science Research Team, RIKEN Advanced Institute for Computational Science, 7-1-26 Minatojima-minami-machi, Chuo-ku, Kobe, 650-0047, Japan
E-mail: nakajima@riken.jp

[b] A. Naruse, Y. Hirano
NVIDIA Corporation, 2-11-7 Akasaka, Minato-ku, Tokyo, 107-0052, Japan

Contract grant sponsor: Ministry of Education, Culture, Sports, Science, and Technology (MEXT), Japan, Next-Generation Supercomputer Project (the K computer project), MEXT, Japan, FLAGSHIP 2020 project, and TSUBAME grand Challenge Program, Category A (the project for TSUBAME 2.5)

an Message Passing Interface (MPI) and Open Multi-Processing (OpenMP) hybrid parallel algorithm[22] of the RI-MP2 energy calculation for massively parallel multi-core supercomputers. This algorithm was implemented into NTChem and was tested on the K computer and on an Intel-based massively parallel cluster system. We performed RI-MP2 energy calculations of a nanographene dimer $(C_{150}H_{30})_2$ with 360 atoms and 9840 AOs using 8911 nodes of the K computer with the 494 Tera FLOPS (TFLOPS; 43%) of the peak performance. However, it is difficult to perform the RI-MP2 calculations with more than 10,000 MPI processes on peta-scale supercomputers using the previously reported algorithm because of a limitation of available MPI processes.

This article describes some improvements for the MPI/OpenMP hybrid parallel algorithm and its implementation from the previously reported ones. We also present the multi-node and multi graphic processor unit (multi-GPU) implementation of the new algorithm to use multi-GPU supercomputers such as TSUBAME 2.5.[23] This article is organized as follows. An overview of the original algorithm and its implementation is presented in Previous parallel algorithm and implementation subsection in Method section. Improvements of the algorithm and its implementation are presented in Dual-level hierarchical MPI parallelization algorithm subsection, New network communication scheme subsection, and Multi-node and multi-GPU implementation subsection in Method section. The results of performance evaluations of the new implementation using the K computer and TSUBAME 2.5 are presented and discussed in Results and discussions section. Finally, concluding remarks are presented.

Method

RI-MP2 method

We briefly review the RI-MP2 method. The RI-MP2 method is an efficient scheme of MP2 calculations. The RI-MP2 method is based on the RI approximation, also called the density-fitting approximation, where the four-center AO 2-ERIs are approximated by the product sum of the three-center AO 2-ERIs and two-center AO 2-ERIs.

$$(\mu\nu|\lambda\sigma)=\sum_{lm}(\mu\nu|l)(l|m)^{-1}(m|\lambda\sigma) \quad (1)$$

Therein, μ, ν, λ, and σ denote the AOs (contracted Gaussian functions are often used in the molecular electronic structure calculations); l, m, and n denote the auxiliary basis functions. In this approximation, the products of AOs are approximated by the linear expansion of auxiliary basis functions. The expansion coefficients of auxiliary basis functions are usually determined by minimizing the self-repulsion integrals defined with the residual vector. In the closed-shell RI-MP2 calculation, the MP2 correlation energy in the MO representation is evaluated using the equation presented below.

$$E^{(2)}=\sum_{ij}^{\text{occ.}}\sum_{ab}^{\text{vir.}}\frac{(ia|jb)[2(ia|jb)-(ib|ja)]}{\varepsilon_i+\varepsilon_j-\varepsilon_a-\varepsilon_b} \quad (2)$$

In this equation, ε_i is the MO energy, i and j denote occupied MOs, and a and b denote virtual MOs. The four-center MO 2-ERIs needed to evaluate eq. (2) are evaluated as shown below.

$$(ia|jb)=\sum_n B_n^{ia}B_n^{jb} \quad (3)$$

The three-center MO 2-ERIs are evaluated from AO 2-ERI using the following equation.

$$B_n^{ia}=\sum_{\mu\nu l}(\mu\nu|l)(l|n)^{-1/2}C_{\mu i}C_{\nu a} \quad (4)$$

In this equation, $C_{\mu i}$ is the MO coefficient. The RI-MP2 method can reduce the inherent computational costs and the required memory sizes while maintaining good accuracy. The computation costs of RI-MP2 calculations are several times smaller than those of full MP2 calculations. The storage requirement of the RI-MP2 calculations is $O(N^3)$ whereas that of the full MP2 calculations is $O(N^4)$. The computational bottleneck of RI-MP2 calculations is the generation of four-center MO 2-ERIs [eq. (3)] requiring the $O(N^5)$ computational costs. Therefore, efficient computation schemes of this part are desired for the computation of large nanoscale molecules.

Previous parallel algorithm and implementation

In an earlier study,[22] we developed an MPI/OpenMP hybrid parallel algorithm of the RI-MP2 energy calculations for massively parallel multi-core supercomputers. We implemented the original algorithm into NTChem using Fortran 90, MPI-1, and OpenMP 2.5. The original code can run on the K computer and on Fujitsu's PRIMEHPC FX10 as well as on the commodity Intel-based architectures. Figure 1 presents the pseudo-code of the original parallel algorithm. The algorithm consists of three steps: (Step 1) evaluation of three-center AO 2-ERIs and 1/3 and 2/3 index transformation of three-center 2-ERIs, (Step 2) 3/3 index transformation of three-center 2-ERIs, and (Step 3) Evaluation of four-center MO 2-ERIs from three-center 2-ERIs and MP2 correlation energy. MPI parallelization is performed for the loops of Step 1 (Loop I), Step 2 (Loop A), and Step 3 (loops A and b_{proc}), where A is batches of virtual MOs and b_{proc} is the index specifying source and destination of MPI processes. We parallelized the loops of (batch of) the virtual MOs in Steps 2 and 3 to use a greater number of MPI processes with more efficient load balancing than the previous algorithms[21,24–26] where loops of occupied MOs are parallelized by MPI and the available MPI processes are limited by the number of occupied MOs (N_{occ}). Generally, the number of virtual MOs (N_{vir}) is more than four times larger than N_{occ} when a double-zeta or higher basis set is used.

Intermediate data of three-center 2-ERIs are stored in the distributed memory and are sent to the buffer memory in the remote nodes in Steps 2 and 3 by the point-to-point network communication calling MPI_ISend and MPI_IRecv. OpenMP parallelization is done for the computational procedures performed in the MPI parallelized loops. Index transformations in Steps 1 and 2 and evaluation of four-center MO 2-ERIs in Step 3 can be executed as matrix–matrix multiplications. Efficient

[Step 1]
Loop shell of auxiliary basis functions l (MPI parallelization with N_{procs} processes)
  Evaluation of $(\mu\nu|l)$ (OpenMP parallelization)
  1/3 and 2/3 index transformation of $(\mu\nu|l)$ by threaded DGEMM in BLAS
  $(ia|l) = \sum_{\nu} C_{\nu a} \sum_{\mu} C_{\mu i} (\mu\nu|l)$ (OpenMP parallelization)
End Loop l

[Step 2]
Evaluation of $(l|m)$ (OpenMP parallelization)
Cholesky decomposition and inversion of $(l|m)$ by threaded LAPACK
(OpenMP parallelization)
Loop batch of virtual MO $A \in$ Myrank (MPI parallelization)
  Loop $b_{\text{proc}} \leq N_{\text{procs}}$-1
    MPI_ISend $(ib|l)$ ($b \in$ Myrank- b_{proc}) to Myrank-b_{proc} process
    MPI_IRecv $(ia|l)$ ($a \in A$) from Myrank+b_{proc} process
  End Loop b_{proc}
  3/3 index transformation of $(ia|l)$ by threaded DGEMM in BLAS
  $B_n^{ia} = \sum_{l} (ia|l)(l|n)^{-1/2}$ ($a \in A$)(OpenMP parallelization)
End Loop A

[Step 3]
Loop batch of virtual MOs $A \in$ Myrank (MPI parallelization)
  Read $B_n^{ia} (a \in A)$
  Loop $b_{\text{proc}} \leq N_{\text{procs}}/2$ (MPI parallelization)
    MPI_ISend B_n^{ia} ($a \in A$) to Myrank-b_{proc} process (ring comm. scheme)
    MPI_IRecv B_n^{jb} ($b \in$ Myrank+b_{proc})
      from Myrank+ b_{proc} process (ring comm. scheme)
    Loop $a \in A$
      Loop $b \in$ Myrank+ b_{proc} with ($a \leq b$)
        Evaluation of four-center MO 2-ERIs by threaded DGEMM in BLAS
        $(ia \mid jb) = \sum_{n} B_n^{ia} B_n^{jb}$ (OpenMP parallelization)
        Evaluation of MP2 correlation energy $E^{(2)}$ by eq. (2) (OpenMP parallelization)
      End Loop b
    End Loop a
  End Loop b_{proc}
End Loop A
MPI_AllReduce $E^{(2)}$

Figure 1. Pseudocode of original MPI/OpenMP hybrid parallel algorithm of RI-MP2 energy calculation.

performance of floating-point operations and thread parallelization are attained by the highly optimized DGEMM subroutine in basic linear algebra subprograms (BLAS).[27] Cholesky decomposition and inversion of matrix in Step 2 are done using the threaded Linear Algebra PACKage (LAPACK)[28] library.

The first shortcoming of our previous parallel algorithm is that the available MPI processes are limited by the number of

virtual MOs in Steps 2 and 3. This limitation presents a severe problem when more than 10,000 MPI processes are used on peta-scale supercomputers such as the K computer, which has 88,128 nodes. The second shortcoming is that the overhead of the network communication in Step 3 becomes severe when vastly numerous MPI processes are used. This is because the total data size for the network communication in Step 3 ($N_{occ}N_{vir}N_{aux}N_{procs}/2$, N_{aux}: the number of auxiliary basis functions) increases with respect to the number of MPI processes N_{procs}. Therefore, it is important to parallelize other MPI unparalleled loops and to reduce the data size for the network communication in Step 3 utilizing whole nodes of peta-scale many-core supercomputers. In the next subsection, we present an improved parallel algorithm to overcome these problems and to facilitate RI-MP2 energy calculations of nanoscale molecules using almost the entire system of peta-scale many-core supercomputers.

Dual-level hierarchical MPI parallelization algorithm

This subsection presents an outline of the improvement of the parallel algorithm for the RI-MP2 energy calculation and its implementation from the original ones presented in the last section. First, we explain the dual-level hierarchical MPI parallelization scheme. Second, we explain the change of the network communication scheme in Step 3. Furthermore, we explain the overview of the multi-node and multi-GPU implementation based on the improved implementation.

We developed the dual-level hierarchical MPI parallelization scheme of RI-MP2 energy calculations to use more than 10,000 MPI processes. The algorithm is outlined as the pseudo-code in Figure 2. In this scheme, we perform the MPI parallelization of tasks computed in the outermost MPI parallelized loops in addition to the parallelization of outermost MPI parallelized loops (parallelized loops in Fig. 1). A dimension of matrices is divided to produce block matrices (the dimension of occupied MO *i* for Step 1 and the dimension of auxiliary basis functions *n* for Steps 2 and 3). Each block matrix is distributed to each MPI process belonging to the MPI subcommunicator for matrix operations. (Step 1) Evaluation of the three-center AO 2-ERIs is distributed to each MPI process by each shell pair. The matrices of three-center AO 2-ERIs are collected to the replicated matrix by MPI_AllReduce. Matrix–matrix multiplications for 1/3 and 2/3 index transformation of three-center AO 2-ERI are then performed, distributing the dimension of occupied MO *i*. (Step 2) Matrix–matrix multiplication for 3/3 index transformation performed distributing the dimension of auxiliary basis functions *n*. (Step 3) Matrix–matrix multiplication to evaluate the four-center MO 2-ERIs is performed distributing the dimension of auxiliary basis functions *n* as shown in the right of Figure 3. The rectangles located on the top and the left of large square in Figure 3 indicate the three-center 2-ERI matrices where the numbers in the rectangles are the rank ID of MPI processes holding the data of three-center 2-ERIs. The rectangles in a large square indicate the task distribution of matrix-matrix multiplication for evaluation of the four-center MO 2-ERIs where the numbers in the rectangles are the rank ID of MPI processes. The matrix of four-center MO 2-ERIs is gathered by MPI_AllReduce among the MPI processes in a MPI sub-communicator (rank 0 and 1 in a rectangle in Fig. 3, for example) after the matrix–matrix multiplication is completed. Evaluation of MP2 correlation energy is parallelized by the loop indices of occupied MOs *i* and *j*.

Figure 4 depicts the distribution of matrix data and the network communication pattern of Step 3 in the dual-level hierarchical parallelization scheme. Network communication for three-center MO integral data is performed in the MPI subcommunicator between virtual MOs (red box in Fig. 4a, for example). Then, network communication for four-center MO 2-ERI data is performed in the MPI sub-communicator between the auxiliary basis functions (yellow box in Fig. 4b, for example). This scheme provides another benefit to reduce the network communication overhead for three-center MO ERIs among the parallelization of virtual MOs in Step 3 because the data sizes of network communication are reduced to $N_{occ}N_{vir}N_{aux}N_{procsmo}/2$ ($N_{procsmo}$: number of MPI processes for the parallelization of virtual MOs, $N_{procsmat}$: number of MPI processes for the parallelization of matrix algebra, $N_{procs} = N_{procsmo}N_{procsmat}$).

As mentioned in last subsection, the main computational kernel of the RI-MP2 code is the DGEMM routine in BLAS. Thus, high peak performance for computing is easily attained because the well-optimized DGEMM routine is supplied as a basic library in the high performance computing systems. This situation is also same on the K computer where 96.6% of the perk performance was measured.[8] The performance of DGEMM is strongly depending on the sizes (dimensions) of matrices. In the previous algorithm, the DGEMM computation of four-center MO 2-ERIs in Step 3 is performed for the N_{occ} by N_{aux} matrices of three-center MO 2-ERIs. However, the size of N_{occ} is generally too small to attain the efficient performance on DGEMM computation (For example, N_{occ} is 600 while N_{aux} is 16,992 for the RI-MP2 calculation of the nanographene stacked dimer $(C_{96}H_{24})_2$ using triple-zeta cc-pVTZ basis set[29] and auxiliary basis set[30]). Thus, this DGEMM computations for the different virtual MO indices distributed in a batch are merged into the single DGEMM computations for the improvement of the peak performance. The dual-level hierarchical parallelization is also suitable for the Non-Uniform Memory Access architecture where MPI parallelized tasks for matrix linear algebra are assigned to each local node. As explained in next section, this parallelization is also suitable for the multi-node and multi-GPU implementation because each MPI process for matrix multiplication is assigned straightforwardly to each GPU device installed in each node.

New network communication scheme

In the previous algorithm, peer-to-peer (P2P) network communication among half of all nodes calling MPI_ISend and MPI_IRecv was performed for getting data of the three-center MO 2-ERIs in Step 3 (see Fig. 5a). This scheme becomes demanding when the number of MPI processes is huge. We changed the network communication scheme into the ring communication pattern to perform MPI_ISend and MPI_IRecv between

[Step 1]
Loop shell of auxiliary basis functions l (MPI parallelization (1st level) with N_{procsmo} processes)
  Loop μ (OpenMP parallelization)
    Loop $\nu \leq \mu$ (MPI parallelization (2nd level) with N_{procsmat} processes)
      Evaluation of $(\mu\nu|l)$
    End Loop ν
  End Loop μ
  MPI_AllReduce $(\mu\nu|l)$ in the sub-communicator for matrix algebra
  1/3 and 2/3 index transformation of $(\mu\nu|l)$ by threaded DGEMM in BLAS

$$(ia|l) = \sum_{\nu} C_{\nu a} \sum_{\mu} C_{\mu i} (\mu\nu|l)$$

    ($i \in$ Myrankmat: MPI parallelization (2nd level) and OpenMP parallelization)
End Loop l

[Step 2]
Evaluation of $(l|m)$ (OpenMP parallelization)
Cholesky decomposition and inversion of $(l|m)$ by threaded LAPACK (OpenMP parallelization)
Loop batch of virtual MO $A \in$ Myrankmo (MPI parallelization (1st level))
  Loop $b_{\mathrm{proc}} \leq N_{\mathrm{procsmo}}$-1
    MPI_ISend $(ib|l)$ ($b \in$ Myrankmo-b_{proc}, $i \in$ Myrankmat) to Myrankmo-b_{proc} process
    MPI_IRecv $(ia|l)$ ($a \in A$, $i \in$ Myrankmat) from Myrankmo+b_{proc} process
  End Loop b_{proc}
  3/3 index transformation of $(ia|l)$ by threaded DGEMM in BLAS $B_n^{ia} = \sum_l (ia|l)(l|n)^{-1/2}$
    ($a \in A$, $i \in$ Myrankmat: MPI parallelization (2nd level) and OpenMP parallelization)
End Loop A

[Step 3]
Loop batch of virtual MOs $A \in$ Myrankmo (MPI parallelization (1st level))
  Read B_n^{ia} ($a \in A$) to send buffer
  Loop $b_{\mathrm{proc}} \leq N_{\mathrm{procsmo}}/2$ (MPI parallelization (1st level))
    MPI_ISend B_n^{ia} ($a \in A$) to Myrankmo-1 process (P2P comm. scheme)
    MPI_IRecv B_n^{jb} ($b \in$ Myrankmo+b_{proc}) from Myrankmo+1 process (P2P comm. scheme)
    Loop $a \in A$
      Loop $b \in$ Myrankmo+b_{proc} with ($a \leq b$)
        Evaluation of four-center MO 2-ERIs by threaded DGEMM in BLAS

$$(ia|jb)_{\mathrm{part}} = \sum_{n \in \mathrm{MyRankMat}} B_n^{ia} B_n^{jb}$$

          (MPI parallelization (2nd level) and OpenMP parallelization)
        MPI_AllReduce $(ia|jb)_{\mathrm{part}}$ in the sub-communicator for matrix algebra
        Loop $i \in$ Myrankmat (MPI parallelization (2nd level))
          Evaluation of MP2 correlation energy $E^{(2)}$ by eq. (2) (OpenMP parallelization)
      End Loop b
    End Loop a
    Switch send buffer for B_n^{ia} and receive buffer for B_n^{jb}
  End Loop b_{proc}
End Loop A
MPI_AllReduce $E^{(2)}$

Figure 2. Pseudocode of improved MPI/OpenMP hybrid parallel algorithm of RI-MP2 energy calculation.

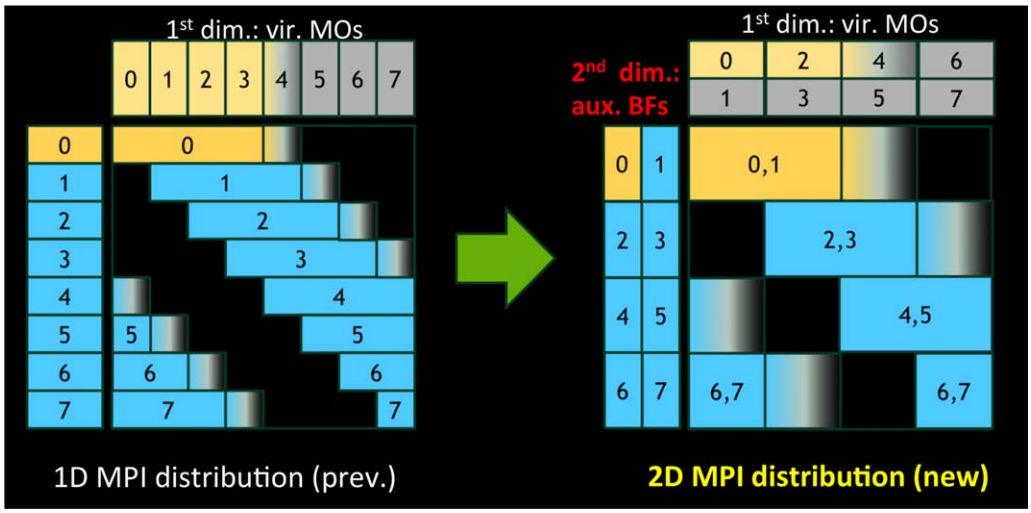


Figure 3. Dual-level hierarchical MPI parallelization scheme for Step 3 in RI-MP2 energy calculation. [Color figure can be viewed at wileyonlinelibrary.com]

nearest neighbor nodes only (Fig. 5b). Data having a next neighbor node are stored into the buffer and are used for the evaluation of four-center MO 2-ERIs. In the next step, the data in the buffer are sent to another node. In addition, we performed the overlap of the computation and the network communication to hide the overhead of communication. The MPI_ISend and MPI_IRecv for the data communication of three-center MO 2-ERIs needed for the next loop index are called before commutation of the current loop index. Then, the data are sent during the computation. MPI_Wait is called after the computation is finished waiting for the finish of the data communication. The overlap of the computation and the communication is especially efficient in the cases of jobs using more than 10,000 MPI processes.

Multi-node and multi-GPU implementation

No multi-node and multi-GPU implementation of RI-MP2 calculations has been reported in the literature, although single-node GPU implementations have been reported.[31–33] Therefore, we developed a multi-node and multi-GPU implementation of the present MPI/OpenMP parallel algorithm for the RI-MP2 energy calculation to execute massively parallel computing on multi-GPU supercomputers. For multi-node and multi-GPU implementation, we adopt the offload model that the matrix–matrix multiplication for evaluating 3-center and 4-center MO 2-ERIs done by DGEMM is offloaded to the GPU devices using the cublasDgemm() routine in the NVIDIA's cuBLAS library.[34] For the efficient use of the multiple GPU devices installed into each node, the matrix–matrix multiplication is distributed to each GPU, applying the dual-level hierarchical MPI parallelization scheme described in previous subsection. We also performed optimization of the data transfer between host CPUs and GPUs. We overlapped the data transfer between host CPUs and GPUs with other operations such as computations on GPUs, computations on host CPUs, and data transfers among nodes to conceal the overhead of data transfer between CPUs and GPUs. In addition, we

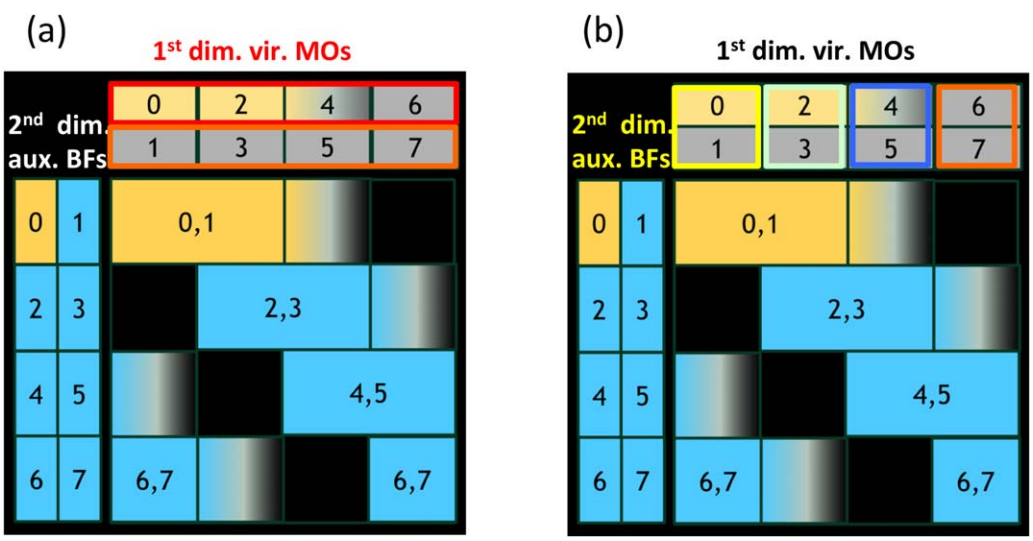
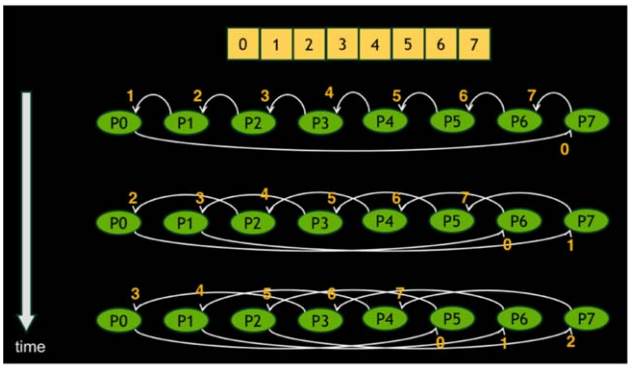
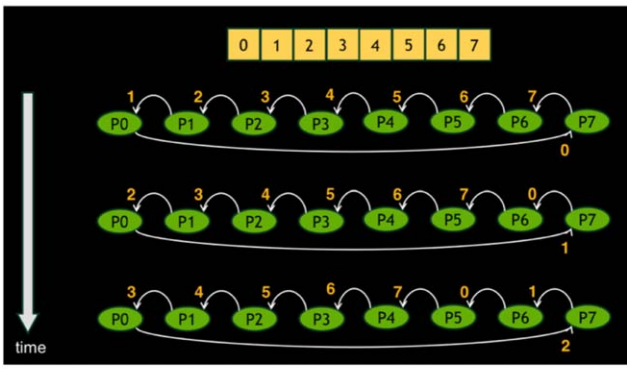


Figure 4. Network communication pattern of matrix data in dual-level hierarchical MPI parallelization scheme for Step 3 in RI-MP2 energy calculation. [Color figure can be viewed at wileyonlinelibrary.com]



(a) Original algorithm (P2P scheme)



(b) Improved algorithm (ring scheme)

Figure 5. Network communication pattern transferring data of three-center MO 2-ERIs in a) original algorithm (P2P scheme) and b) improved algorithm (ring scheme). [Color figure can be viewed at wileyonlinelibrary.com]

programmed to use the page-locked memory (called pinned memory) attaining the high throughput data transfer between the host CPUs and the GPUs. The GPU code was implemented using CUDA[35] and interfaced with the Fortran 90 code.

Results and Discussions

We evaluated the performance of the improved algorithm and implementation using the K computer and TSUBAME 2.5. We checked the effect of the dual-level hierarchical MPI parallelization and the effect of changing the network communication scheme using the K computer. We also checked the effect of multi-node and multi-GPU implementation comparing the CPU implementation using TSUBAME 2.5.

Performance of dual-level hierarchical MPI parallelization algorithm and implementation on the K computer

The developed code was compiled using Fujitsu's Fortran compiler and was linked with the tuned BLAS and LAPACK libraries supplied by Fujitsu. We performed the RI-MP2 calculation of nano-sized molecule, a nanographene stacked dimer $(C_{96}H_{24})_2$, using the cc-pVTZ basis set[29] and the cc-pVTZ auxiliary basis set[30] for RI-based correlation calculations (6432 AOs, 600 occupied MOs, 5832 virtual MOs, 16,992 auxiliary basis functions, spherical Gaussian functions, and C1 symmetry).

We first checked the effect of the change of network communication scheme described in the previous section. Figure 6 shows a detailed profile of the elapsed times with the difference of the network communication pattern: the P2P pattern (Fig. 6a) and the ring pattern (Fig. 6b) where the 2048 nodes (corresponding to the total number of MPI processes $N_{procs} = 2048$) were used and one MPI process was assigned to each MPI subcommunicator for the matrix computing (i.e., $N_{procsmat} = 1$) corresponding the single-level parallelization scheme. Considerable reduction of the network communication time for three-center 2-ERIs was attained by the change of network communication pattern from the P2P pattern to the ring pattern. The network communication time of 3/3 transformed ERIs in Step 3 (shown in red) is reduced to about half.

We next assessed the performance of the dual-level parallelization scheme. We performed calculations using 2048 nodes,

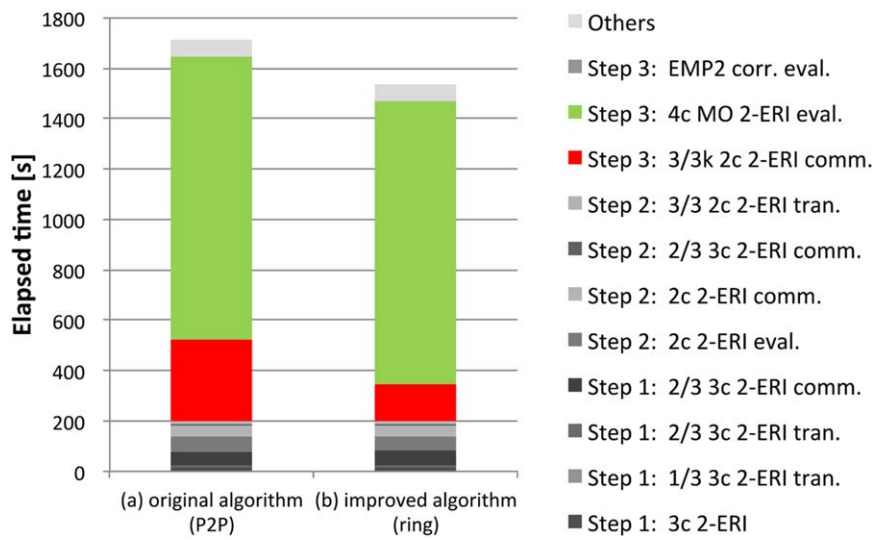


Figure 6. Detailed profile of elapsed times depending on network communication pattern: a) original algorithm (P2P) and b) improved algorithm (ring). [Color figure can be viewed at wileyonlinelibrary.com]

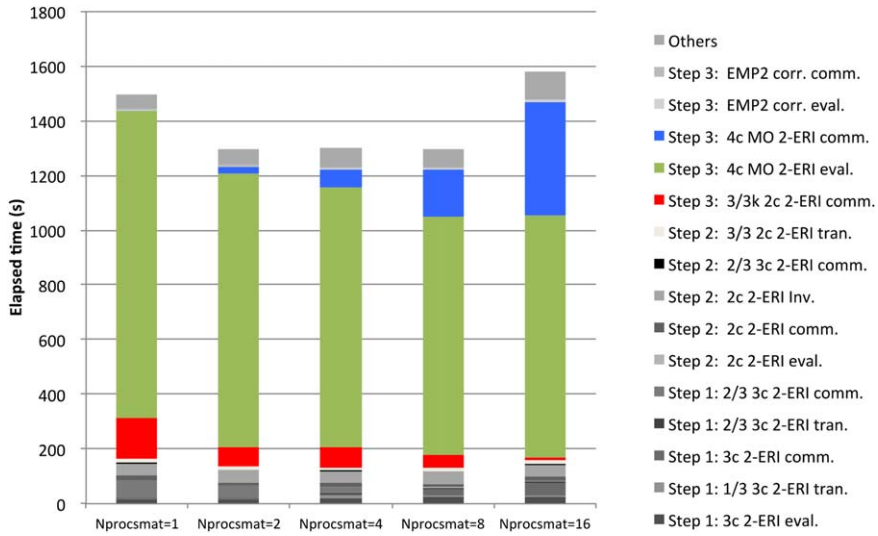


Figure 7. Detailed profile of elapsed times depending on number of MPI processes used for matrix computing (1–16 MPI processes were used in the MPI sub-communicator for matrix computing and 2048 nodes were used). [Color figure can be viewed at wileyonlinelibrary.com]

changing the number of MPI processes used in the MPI sub-communicator for the matrix computation $N_{procsmat}$ to check the effects and the overheads of the dual-level hierarchical MPI parallelization scheme. Figure 7 presents a detailed profile of the elapsed times depending on $N_{procsmat}$. It is noteworthy that the elapsed times for the network communication of 3/3 transformed MO 2-ERIs (shown in red) decrease with respect to $N_{procsmat}$ increases and become about 8.4% of the elapsed times at $N_{procsmat} = 1$ when $N_{procsmat} = 16$. This is because the data sizes of three-center 2-ERIs communicated among next neighboring nodes in the parallelization of virtual MOs ($N_{occ}N_{vir}N_{aux}N_{procsmo}/2$) decreases linearly with $N_{procsmo}$ decreasing. The elapsed times for the computation of four-center MO 2-ERIs (shown in green) are decreased with respect to increasing $N_{procsmat} = 1$ to $N_{procsmat} = 8$. They converge at $N_{procsmat} = 8$. This is because the performance of DGEMM computations is improved by the merge of three-center 2-ERIs matrices with respect to $N_{procsmat}$. The elapsed times for the network communication of matrix of four-center MO 2-ERIs (shown in blue) are increased linearly with respect to $N_{procsmat}$. Consequently, the total elapsed times involved for the computation of four-center MO 2-ERIs become the tradeoff of the overhead of the computation and the network communication of four-center MO 2-ERIs. The best performance is attained when $N_{procsmat} = 4$ using 2048 nodes. We then checked the parallel scalability of the present algorithm for the number of nodes (i.e., N_{procs}) and $N_{procsmat}$. Table 1 presents the elapsed time and speedups of parallel calculations of RI-MP2/cc-pVTZ calculation of nanographene stacked dimer $(C_{96}H_{24})_2$ using up to 65,536 nodes of the K computer. The elapsed times and the parallel scalability applying the dual-level parallelization are better than those of the original single-level parallelization corresponding to $N_{procsmat} = 1$. Results show that $N_{procsmat} = 4$ gives the best performance up to 4096 nodes, $N_{procsmat} = 8$ gives the best performance from 8192 nodes to 16,384 nodes, and $N_{procsmat} = 16$ gives the best performance from 32,768 nodes to 65,536 nodes. Table 2 shows the peak performance of parallel calculations of RI-MP2/cc-pVTZ using up to 65,536 nodes of the K computer. The elapsed time is 282 s and the peak-performance is 763 TFLOPS using 65,536 nodes. The reason for the reduction of parallel scalability in this case is that most demanding computation procedures are Cholesky decomposition and inversion of the matrix of two-center 2-ERIs in Step 2, where MPI parallelization is not performed (thread parallelization is only performed using the optimized LAPACK library). In the future, we will perform the MPI parallelization of these procedures applying the ScaLAPACK[36] library. Next, we present the scalability in the case of calculations of a

Table 1. Elapsed time (second) and parallel speedups of the nanographene dimer $(C_{96}H_{24})_2$ RI-MP2/cc-pVTZ energy calculations on K computer.

		$N_{procsmat} = 1$		$N_{procsmat} = 2$		$N_{procsmat} = 4$		$N_{procsmat} = 8$		$N_{procsmat} = 16$	
Node	Cores	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup
1024	8192	2366	8192	2289	8192	2211	8192	2346	8192		
2048	16384	1491	12999	1292	14515	1291	14025	1315	14615	1581	(16384)[a]
4096	32768	1451	13354	856	21901	743	24375	773	24873	895	(28927)[a]
8192	65536			826	22716	532	34078	490	39250	541	(47881)[a]
16384	131072					543	33331	366	52452	391	(66174)[a]
32768	262144							359	53515	298	(87005)[a]
65536	524288									282	(91938)[a]

[a] Speedup with respect to 2048 nodes.

Table 2. Peak performance (TFLOPS) and efficiency of nanographene dimer $(C_{96}H_{24})_2$ RI-MP2/cc-pVTZ energy calculations on K computer.

		$N_{procsmat} = 1$		$N_{procsmat} = 2$		$N_{procsmat} = 4$		$N_{procsmat} = 8$		$N_{procsmat} = 16$	
Node	Cores	TFLOPS	Efficiency (%)	TFLOPS	Efficiency (%)	TFLOPS	Efficiency (%)	TFLOPS	Efficiency (%)	TFLOPS	Efficiency (%)
1024	8192	91	69	94	72	97	74	92	70	N/A	N/A
2048	16384	144	55	166	64	166	64	163	62	136	52
4096	32768	148	28	251	48	289	55	278	53	240	46
8192	65536			260	25	405	39	439	42	397	38
16384	131072					396	19	587	28	549	26
32768	262144							599	14	722	17
65536	524288									763	9

larger nanoscale molecule. We performed RI-MP2/cc-pVTZ calculations of nanographene stacked dimer $(C_{150}H_{30})_2$ using up to 80,199 nodes of the K computer (9840 AOs, 930 occupied MOs, 8910 virtual MOs, 45,992 auxiliary basis functions, spherical Gaussian functions, and C1 symmetry). Table 3 presents the elapsed time, the speedups, and the peak performance of parallel calculations using up to 80,199 nodes of the K computer. In these calculations, $N_{procsmo}$ was fixed at 8911, which is one more the number of virtual MOs N_{vir}. This is because load balancing of tasks in Step 3 (i.e., the task distribution of (A, B) batch pairs with the constraint $B \geq A$) is better using odd number of $N_{procsmo}$ rather than even number of $N_{procsmo}$ when $N_{procsmo}$ is more than half of N_{vir}. Uniform task distribution is possible up to $N_{procsmo} - 1$ processes when $N_{procsmo}$ is odd while it is limited up to $N_{procsmo}/2$ process (much smaller than $N_{procsmo} - 1$) when $N_{procsmo}$ is even. In this case, one process becomes idle but the effect of this process is negligible and $N_{procsmo} - 1$ can be utilized efficiently. $N_{procsmat}$ are changed from $N_{procsmat} = 1$ to $N_{procsmat} = 9$. The parallel scalability is good up to 35,644 nodes (61% of the parallel scalability is attained compared with the result obtained using 8911 nodes). The peak performance of present implementation using 8911 nodes (0.7 PFLOPS and 62% efficiency) is 1.4 times improved than that of previous implementation (0.5 PFLOPS and 43% efficiency[22]). The parallel performance decreases monotonically but still scales up to 80,199 nodes. The reason for the degradation of the parallel performance is that the computation times for computation procedures that are not parallelized (Cholesky decomposition and inversion of the matrix of two-center 2-ERIs where MPI parallelization in Step 2) become remarkable and that the overhead of the network-communication (MPI_AllReduce) for the four-center MO 2-ERIs in Step 3 is the second dominant part during the calculation. The elapsed time is 759 s and 3.1 PFLOPS of peak performance (30% of efficiency) is attained using 80,199 nodes. This result exhibits the best peak performance among the reported MP2 parallel calculations.

Performance of CPU/GPU implementation on TSUBAME 2.5

Next, we present results of a performance comparison of the CPU implementation and the CPU/GPU implementation. We performed test calculations on the CPU/GPU hybrid supercomputer system TSUBAME2.5 at Tokyo Institute of Technology. TSUBAME 2.5 consists of 1408 nodes equipped with 2 socket CPUs/node (12 cores/node, 6 cores/CPU Xeon 5670; Intel Corp.) and three GPU boards/node (Tesla K20X; NVIDIA Corp.). Each node is connected with the InfiniBand network (Grid Director 4700; Mellanox) with non-blocking and full bisectional bandwidth. The CPU code and GPU code were compiled with Intel compiler and CUDA 5.0 compiler, respectively. Intel Math Kernel Library was used for BLAS and LAPACK library on the CPUs and cuBLAS library was used on the GPUs. For every CPU jobs, 12 OpenMP threads per node were used. For the CPU/GPU jobs, three MPI processes and four OpenMP threads per node were used, where each MPI process uses each GPU device.

We first verified the speedup by offloading to the GPUs and the parallel scalability for the nanographene $C_{96}H_{24}$ at the RI-MP2/cc-pVTZ level (3216 AOs, 300 occupied MOs, 2916 virtual MOs, 8496 auxiliary basis functions, spherical Gaussian functions, and C1 symmetry). $N_{procsmat} = 4$ and $N_{procsmat} = 3$ are used for the CPU jobs and the CPU/GPU jobs, respectively. Table 4 shows the elapsed time and speedups of parallel calculations up to 512 nodes of TSUBAME 2.5. Results of the CPU/GPU computation are 4.1–6.6 times faster than those of the CPU computation. The parallel scalability is good up to 512 nodes both in the CPU/GPU implementation and in the CPU implementation.

Table 3. Elapsed time (second), parallel speedups, peak performance (PFLOPS), and efficiency of nanographene dimer $(C_{150}H_{30})_2$ RI-MP2/cc-pVTZ calculations on K computer.

Nodes	CPU cores	Time	Speedups	PFLOPS	Efficiency (%)
8911	71288	2692	71288	0.7	62
17822	142576	1627	117939	1.2	54
35644	285152	1095	175248	2.0	44
53466	427728	881	217671	2.6	37
71288	570304	783	245246	2.9	32
80199	641592	759	252893	3.1	30

Table 4. Elapsed time (second) and parallel speedups of nanographene $C_{96}H_{24}$ RI-MP2/cc-pVTZ calculations using CPU/GPU and CPU on TSUBAME 2.5.

		CPU/GPU		CPU	
Nodes	CPU cores	Time	Speedups	Time	Speedups
64	768	198	768	1277	768
128	1536	123	1244	806	1216
256	3072	92	1648	531	1845
512	6144	86	1780	349	2808

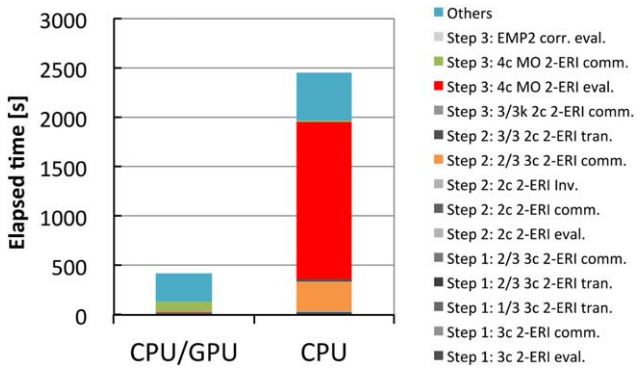


Figure 8. Detailed profile of elapsed time of CPU/GPU and CPU computation on TSUBAME 2.5 for RI-MP2/cc-pVTZ calculation of nanographene dimer $(C_{96}H_{24})_2$. [Color figure can be viewed at wileyonlinelibrary.com]

We then performed the RI-MP2/cc-pVTZ calculation of nanographene dimer $(C_{96}H_{24})_2$ (6432 AOs, 600 occupied MOs, 5832 virtual MOs, 16,992 auxiliary basis functions, spherical Gaussian functions, and C1 symmetry) using the 1349 nodes and 4047 GPUs of TSUBAME 2.5. $N_{procsmat} = 1$ and $N_{procsmat} = 3$ are used for the CPU jobs and the CPU/GPU jobs, respectively. This calculation, which is the largest reported RI-MP2 calculation using the CPU/GPU hybrid supercomputer system, demonstrates the potential of the CPU/GPU hybrid computing of the RI-MP2 method for the productive job on the nanoscale molecules. Figure 8 presents a detailed profile of the elapsed time of CPU/GPU and CPU jobs. The CPU/GPU implementation is 5.9 times faster than the CPU implementation because the remarkable reduction of time for DGEMM computations decreases to very short times, as shown in Figure 8 (see the red color for the DGEMM computation evaluating the four-center MO 2-ERIs in Fig. 8). The elapsed times of CPU/GPU job and CPU job are, respectively, 419 s and 2467 s. The peak performance values of CPU/GPU and CPU code are, respectively, 514.7 TFLOPS (9.3%) and 87.5 TFLOPS (42%). The peak performance of CPU/GPU implementation is much lower than that of the CPU implementation. This is because the most time consuming part of the CPU/GPU implementation is no longer matrix–matrix multiplication using DGEMM but other operations such as data transfers among nodes for three-center MO 2-ERIs in Steps 2 and 3, and the computation of three center AO 2-ERIs in Step 1 (see Fig. 8). To date, we have not optimized our code for network communications to match the network of TSUBAME 2.5. We are planning code optimization to make it suitable for the TSUBAME 2.5 network.

Concluding Remarks

This article presents improvements of the MPI/OpenMP hybrid parallel algorithm for the RI-MP2 energy calculations from the previous algorithm developed by the authors. The dual-level parallelization scheme has been developed utilizing more than 10,000 MPI processes on peta-scale supercomputers such as the K computer and TSUBAME 2.5. We have implemented a ring communication scheme and an overlapping scheme of computation and network communication in Step 3 to reduce the overhead of network communications. Test calculations of a nano-scale molecule $(C_{150}H_{30})_2$ using up to 80,199 nodes of the K computer demonstrate that the dual-level parallelization scheme is efficient. The parallel performance scales up to the 80,199 nodes; 3.1 PFLOP peak performance was attained using 80,199 nodes. We also performed the multi-node and multi-GPU implementation based on the present algorithm. The DGEMM matrix–matrix multiplication during the RI-MP2 calculations is offloaded to the GPU devices. The present dual-level parallelization scheme is suitable for the computation using multi-nodes and multi-GPUs, where the computation of sub-matrices is assigned to each GPU. Considerable speedup of the RI-MP2 calculations can be attained using the present multi-node and multi-GPU implementation. The peak performance of the CPU/GPU hybrid computation and the CPU computation of nano-size molecule $(C_{96}H_{24})_2$ using 1349 nodes and 4047 GPUs of TSUBAME 2.5 are, respectively, 514.7 TFLOPS and 87.5 TFLOPS.

The code developed in this study has been released as a part of Fiber mini app suite.[37] The code is also available as a part of NTChem for users of the K computer and some other supercomputers in Japan.[38,39] This new algorithm and implementation extend the applicability of the *ab initio* electron correlation theory to the research of the computational material science and the biological science exploiting the peta-scale supercomputer systems. Recently, we performed calculations of heats of formation of some higher fullerenes up to C_{180} with the double-hybrid DFT (DSD-PBE-PBE/cc-pVQZ) level on the K computer utilizing the new parallel RI-MP2 code.[40] This new algorithm is expected to underpin the development of more efficient algorithms of electron correlation calculations functioning on exa-scale supercomputers, for which more numerous CPU cores can be used to create homogeneous or heterogeneous architectures.

Acknowledgments

Benchmark calculations were performed on the K computer (RIKEN, Kobe, Japan) and the TSUBAME 2.5 system (Tokyo Institute of Technology, Tokyo, Japan). Preliminary calculations were performed on the supercomputer system of the Research Center for Computational Science (National Institute of Natural Sciences, Okazaki, Japan).

Keywords: massively parallel algorithm · GPGPU · electron correlation theory · second-order Møller–Plesset perturbation theory · K computer · TSUBAME 2.5 · NTChem

How to cite this article: M. Katouda, A. Naruse, Y. Hirano, T. Nakajima. *J. Comput. Chem.* **2016**, *37*, 2623–2633. DOI: 10.1002/jcc.24491

[1] W. A. de Jong, E. J. Bylaska, N. Govind, C. L. Janssen, K. Kowalski, T. Müller, I. M. B. Nielsen, H. J. J. van Dam, V. Veryazov, R. Lindh, *Phys. Chem. Chem. Phys.* **2010**, *20*, 6896.
[2] K computer. Available at: http://www.aics.riken.jp/en/k-computer/, accessed on June 17, **2016**.

[3] M. Yokokawa, F. Shoji, A. Uno, M. Kurokawa, T. Watanabe, In Proceedings of the 17th IEEE/ACM International Symposium on Low-Power Electronics and Design, ISLPED '11, Fukuoka, Japan, Aug 1–3, **2011**, pp. 371–372.
[4] TOP 500. Available at: http://www.top500.org/, accessed on June 17 **2016**.
[5] T. Maeda, T. Furumura, S. Noguchi, S. Takemura, S. Sakai, M. Shinohara, K. Iwai, S. Lee, *Bull. Seism. Soc. Am.* **2013**, *103*, 1456.
[6] T. Miyakawa, M. Satoh, H. Miura, H. Tomita, H. Yashiro, A. T. Noda, Y. Yamada, C. Kodama, M. Kimoto, K. Yoneyama, *Nat. Commun.* **2014**, *5*, 3769.
[7] N. Yoshii, Y. Andoh, K. Fujimoto, H. Kojima, A. Yamada, S. Okazaki, *Int. J. Quantum Chem.* **2015**, *115*, 342.
[8] Y. Hasegawa, M. Kurokawa, H. Inoue, I. Miyoshi, M. Yokokawa, J.-I. Iwata, M. Tsuji, D. Takahashi, A. Oshiyama, K. Minami, T. Boku, F. Shoji, A. Uno, In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11); ACM Press, New York, New York, **2011**, Vol. *1*, pp: 1:1–1:11.
[9] K. Ushirogata, K. Sodeyama, Y. Okuno, Y. Tateyama, *J. Am. Chem. Soc.* **2013**, *135*, 11967.
[10] T. Ishiyama, K. Nitadori, J. Makino, In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12). IEEE Computer Society Press, Los Alamitos, CA, **2012**, Article 5, pp. 5:1–5:10.
[11] NTChem. Available at: http://labs.aics.riken.jp/nakajimat_top/ntchem_e.html, accessed on June 17, **2016**.
[12] T. Nakajima, M. Katouda, M. Kamiya, Y. Nakatsuka, *Int. J. Quantum Chem.* **2015**, *115*, 349.
[13] A. Szabo, N. S. Ostlund, Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory; Dover: Mineola, NY, **1996**.
[14] R. G. Parr, W. Yang, *Ann. Rev. Phys. Chem.* **1995**, *46*, 701.
[15] D. Cremer, *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2011**, *1*, 509.
[16] R. J. Bartlett, M. Musiał, *Rev. Mod. Phys.* **2007**, *79*, 291.
[17] Y. Nakatsuka, T. Nakajima, *J. Chem. Phys.* **2012**, *137*, 154103.
[18] Y. Nakatsuka, T. Nakajima, K. Hirao, *J. Chem. Phys.* **2010**, *132*, 174108.
[19] Y. Nakatsuka, T. Nakajima, M. Nakata, K. Hirao, *J. Chem. Phys.* **2010**, *132*, 054102.
[20] M. W. Feyereisen, G. B. Fitzgerald, A. Komornicki, *Chem. Phys. Lett.* **1993**, *208*, 359.
[21] D. E. Bernholdt, R. J. Harrison, *Chem. Phys. Lett.* **1996**, *250*, 477.
[22] M. Katouda, T. Nakajima, *J. Chem. Theory Comput.* **2013**, *9*, 5373.
[23] TSUBAME2.5. Available at: http://tsubame.gsic.titech.ac.jp/en, accessed on June 17, **2016**.
[24] D. E. Bernholdt, *Parallel Comput.* **2000**, *26*, 945.
[25] C. Hättig, A. Hellweg, A. Köhn, *Phys. Chem. Chem. Phys.* **2006**, *8*, 1159.
[26] M. Katouda, S. Nagase, *Int. J. Quantum Chem.* **2009**, *109*, 2121.
[27] BLAS. Available at: http://www.netlib.org/blas/, accessed on June 17, **2016**.
[28] LAPACK. Available at: http://www.netlib.org/lapack/, accessed on June 17, **2016**.
[29] T. H. Dunning, *J. Chem. Phys.* **1989**, *90*, 1007.
[30] D. E. Bernholdt, R. J. Harrison, *J. Chem. Phys.* **1998**, *109*, 1593.
[31] L. Vogt, R. Olivares-Amaya, S. Kermes, Y. Shao, C. Amador-Bedolla, A. Aspuru-Guzik, *J. Phys. Chem. A* **2008**, *112*, 2049.
[32] M. A. Watson, R. Olivares-Amaya, R. G. Edgar, A. Aspuru-Guzik, *Comput. Sci. Eng.* **2010**, *12*, 40.
[33] R. Olivares-Amaya, M. A. Watson, R. G. Edgar, L. Vogt, Y. Shao, A. Aspuru-Guzik, *J. Chem. Theory Comput.* **2010**, *6*, 135.
[34] CuBLAS. Available at: http://docs.nvidia.com/cuda/cublas/, accessed on June 17, **2016**.
[35] NVIDIA CUDA ZONE. Available at: https://developer.nvidia.com/cuda-zone/, accessed on June 17, **2016**.
[36] ScaLAPACK. Available at: http://www.netlib.org/scalapack/, accessed on June 17, **2016**.
[37] Fiber Miniapp Suite. Available at: https://fiber-miniapp.github.io/, accessed on June 17, **2016**.
[38] Research Center for Computational Science, National Institutes of Natural Sciences. Available at: https://ccportal.ims.ac.jp/en/, accessed on June 17, **2016**.
[39] Foundation for Computational Science. Available at: http://www.j-focus.or.jp/focus/, accessed on June 17, **2016**.
[40] B. Chan, Y. Kawashima, M. Katouda, T. Nakajima, K. Hirao, *J. Am. Chem. Soc.* **2016**, *138*, 1420.

Received: 1 July 2016
Revised: 25 August 2016
Accepted: 26 August 2016
Published online on 16 September 2016