Midpoint Cell Method for Hybrid (MPI+OpenMP) Parallelization of Molecular Dynamics Simulations

Jaewoon Jung,[a] Takaharu Mori,[b,c] and Yuji Sugita*[a,b,c]

We have developed a new hybrid (MPI+OpenMP) parallelization scheme for molecular dynamics (MD) simulations by combining a cell-wise version of the midpoint method with pair-wise Verlet lists. In this scheme, which we call the midpoint cell method, simulation space is divided into subdomains, each of which is assigned to a MPI processor. Each subdomain is further divided into small cells. The interaction between two particles existing in different cells is computed in the subdomain containing the midpoint cell of the two cells where the particles reside. In each MPI processor, cell pairs are distributed over OpenMP threads for shared memory parallelization. The midpoint cell method keeps the advantages of the original midpoint method, while filtering out unnecessary calculations of midpoint checking for all the particle pairs by single midpoint cell determination prior to MD simulations. Distributing cell pairs over OpenMP threads allows for more efficient shared memory parallelization compared with distributing atom indices over threads. Furthermore, cell grouping of particle data makes better memory access, reducing the number of cache misses. The parallel performance of the midpoint cell method on the K computer showed scalability up to 512 and 32,768 cores for systems of 20,000 and 1 million atoms, respectively. One MD time step for long-range interactions could be calculated within 4.5 ms even for a 1 million atoms system with particle-mesh Ewald electrostatics. © 2014 Wiley Periodicals, Inc.

DOI: 10.1002/jcc.23591

Introduction

Molecular dynamics (MD) simulations have become a powerful tool to solve many-body problems in several research fields. In biological systems, in particular, conformational dynamics of biomolecules are simulated to investigate relationships between structure and function.[1] Most essential biological phenomena, like protein folding, ligand-receptor binding, and allosteric conformational changes depend on the dynamics of biomolecules, typically on the time-scale of microsecond or much longer.[2] Conversely, a time step Δt for integrating the discrete Newtonian equation of motion in MD simulation is limited to 1–2 fs due to the fast vibrational frequencies of covalent bonds. This short time step makes long MD simulations difficult on conventional computers. Along with computer hardware developments, various time saving computational schemes in MD simulations have been introduced recently.[3–17] Nowadays, researchers are able to perform MD simulations of large complex protein systems on the microsecond time-scale.[18,19]

The main bottleneck in MD simulations is the calculation of pair-wise nonbonded interactions. Without any approximation, computational time for the nonbonded interactions is proportional to n^2, where n is the number of particles in the system. The $O(n^2)$ calculation is reduced to $O(n\log n)$ by particle-mesh Ewald (PME) for long-range electrostatic interactions.[20] In PME, long-range electrostatic interactions are described in reciprocal space (long-range nonbonded interactions), and truncation of nonbonded interactions is introduced for real-space electrostatic and van der Waals interactions (short-range nonbonded interactions). For short-range interactions, a list of interaction pairs (pair-list) should be generated every few steps in MD simulations. Currently, the first developed Verlet list[21] and its modifications like cell-linked list[22] have become standard for computing short-range nonbonded interactions. Recently, a new Verlet list suitable for multicore central processing units (CPU) hardware has been developed.[23]

Graphics processing units (GPU) are often used in MD simulations of small biological systems like a protein or a polypeptide in solution for fast evaluation of nonbonded interactions.[24,25] The usage of GPUs, however, limits the size of target systems to about 1 million atoms. To simulate such large systems on longer time scales, most researchers focus on parallelization of nonbonded interactions using a huge number of CPU in massively parallel computers. For the parallelization, two different decomposition schemes have been proposed: (i) atomic/force decomposition and (ii) spatial decomposition (or domain decomposition). The former was

[a] J. Jung, Y. Sugita
Computational Biophysics Research Team, RIKEN Advanced Institute for Computational Science, 7-1-26
Minatojima-minamimachi, Chuo-ku, Kobe, Hyogo Kobe, 640-0047, Japan,
Fax: +81 48 467 4532
E-mail: sugita@riken.jp
[b] T. Mori, Y. Sugita
RIKEN Theoretical Molecular Science Laboratory, 2-1 Hirosawa, Wako-shi, Saitama, 351-0198, Japan
[c] T. Mori, Y. Sugita
Laboratory for Biomolecular Function Simulation, RIKEN Quantitative Biology Center (QBiC), IMDA 6F, 1-6-5
Minatojimaminami-machi, Chuo-ku, Kobe, Hyogo, 650-0047, Japan
Contract grant sponsor: High Performance Computing Infrastructure (HPCI) Strategic Program (Proposal number: hp130003) of MEXT.

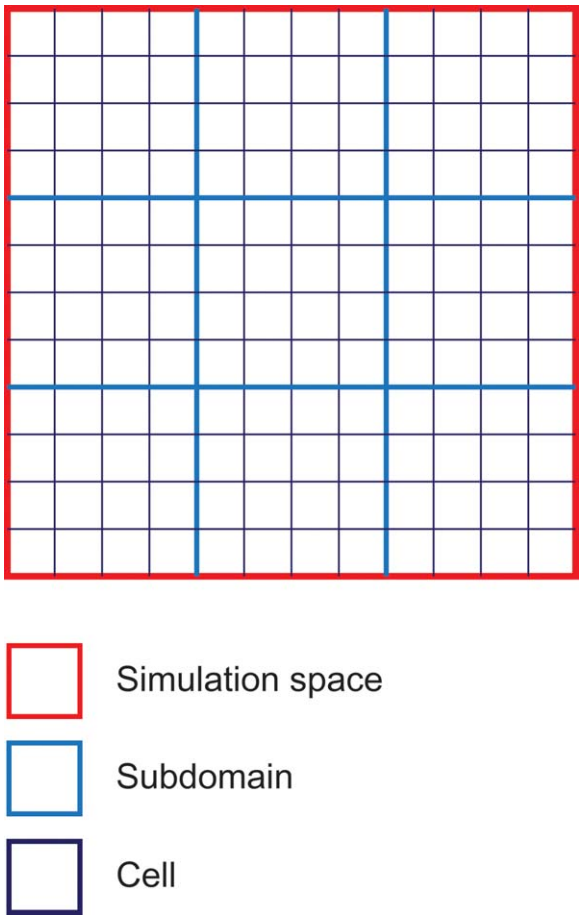

Figure 1. Spatial decomposition scheme in the midpoint cell method. A simulation space is decomposed into nine subdomains each of which is assigned to a single MPI processor. Each subdomain is again decomposed into 16 cells.

developed earlier due to its simplicity but the latter shows better parallel performance for large systems. To achieve higher parallel efficiency, hybrid parallelization in modern multicore CPU has been introduced using message passing interface (MPI) for distributed memory parallelization and OpenMP for shared memory parallelization, which is specific to clusters of multicore computers. Hybrid (MPI+OpenMP) parallelization applied to the spatial decomposition scheme should be, at least currently, the best usage of modern massively parallel supercomputers.[7,10,12,14,17]

The midpoint method was originally developed in Blue Matter MD software (version V4),[14,15] and independently suggested by Bowers et al.[26] This has several advantages over the other spatial decomposition schemes. Here, we offer an improvement on the original midpoint method by introducing a cell-wise approach.[23] First, we group the particle data cell-wise instead of storing the particle data according to particle indices. Second, we generate Verlet pair-lists[21] based on interacting cell pairs. Third, an interaction of particle pairs is evaluated on the subdomain containing the midpoint cell for a given cell pair in which each particle resides. This method, which we call the midpoint cell method, uses CPU cache more efficiently and is suitable for OpenMP shared memory parallelization.

In this article, we give a preview of the spatial decomposition scheme, midpoint method, and pair-wise Verlet lists method in sections Spatial decomposition scheme, Midpoint method, and Pair-wise Verlet list, respectively. The midpoint cell method is introduced in section Midpoint cell method: cell-wise approach of the midpoint scheme with pair-wise Verlet list. We discuss the performance of the new method for two systems in section Results and Discussions. A final conclusion is given in section Conclusions.

Method

Spatial decomposition scheme

In the spatial decomposition scheme, simulation space is geometrically divided into subdomains. Subdomain edges are greater than or equal to the cutoff distance of short-range nonbonded interactions. With MPI parallelization, each subdomain is assigned to a MPI processor, and is further divided into small cells, if necessary. The size of a subdomain is apportioned such that each subdomain communicates only to adjacent subdomains or cells. After communication, each subdomain has coordinate data for the particles in the subdomain as well as in adjacent ones. A two-dimensional (2D) representation of the spatial decomposition scheme is shown in Figure 1.

Midpoint method

In the midpoint method,[26] simulation space is divided into subdomains like other spatial decomposition schemes. An interaction subdomain, in which a pair of particles interacts with each other, is assigned as the subdomain containing the midpoint of the segment connecting two particles. Figure 2a

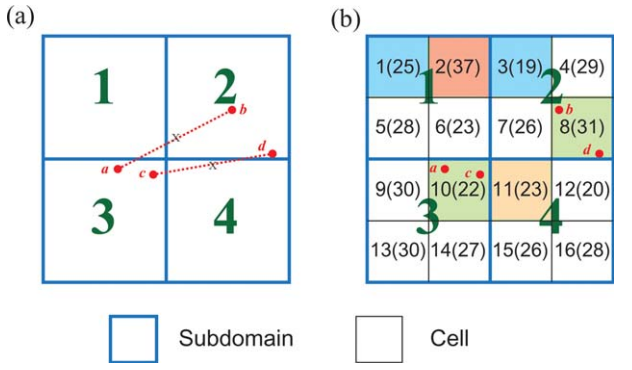

Figure 2. Assignment of particle pairs to an interaction domain in the (a) midpoint method and (b) midpoint cell method. We assume four subdomains in both methods. We further assume four cells in each subdomains in the midpoint cell method. Numbers in parentheses in each cell are numbers of atoms in each cell. Each cell has a side length greater than the cutoff distance. In the midpoint method, the interaction between ***a*** and ***b*** in subdomain **2** is according to the center position between the two particles (marked **x**). The interaction between ***c*** and ***d***, conversely, is considered in subdomain **4**. Unlike in the midpoint method, in the midpoint cell method the interaction between ***a*** and ***b***, and the interaction between ***c*** and ***d*** are performed in the same subdomain. According to Figure 2b, possible midpoint cells are either 7 or 11. The midpoint cell is assigned as the one with the least number of particles, so, in this case, cell 11 (or subdomain **4**) computes interactions for these particle pairs.

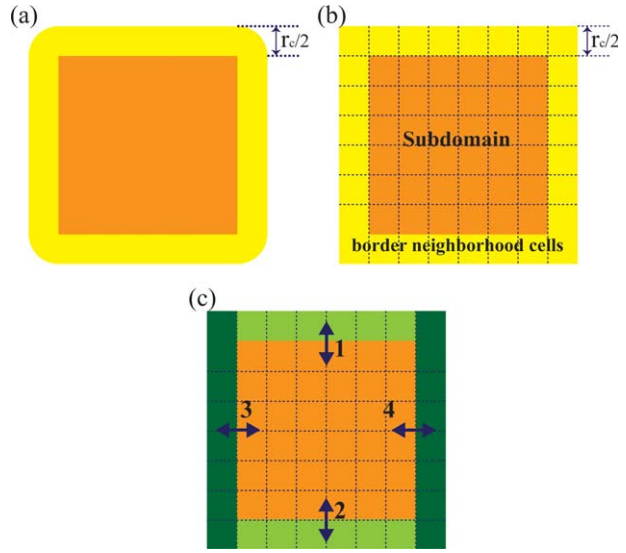

Figure 3. Import region of the 2D (a) midpoint method and (b) midpoint cell method. Here, subdomains are colored in orange and the import region is in yellow. Our midpoint cell method slightly increases the import region compared to the original midpoint method. However, using the communication pattern as shown in (c) for a 2D case, direct communication from/to corner cells is avoided and four local communications suffice to send/receive data. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

illustrates two examples in assigning interaction subdomains in the midpoint method. The interaction between particles ***a*** and ***b*** is computed in subdomain **2** because the midpoint of particle pair (***a***, ***b***) exists in subdomain **2**, whereas the computation between particles ***c*** and ***d*** is carried out in subdomain **4**. This method sometimes computes the interaction between a pair of particles on a processor on which neither particle resides and is referred to as neutral territory method.[27] When the distance between two particles is r, the distance from either particle to their midpoint is $r/2$. Therefore, each particle exists within the distance $r_c/2$ from the interaction subdomain if we consider only particle interactions within the cutoff distance, r_c. From this feature, it is sufficient to import coordinate data within a distance of $r_c/2$ from each subdomain.

The original midpoint method has several advantages for parallelization: (1) the midpoint method requires less communication bandwidth at moderate or high degrees of parallelism, (2) it applies not only to nonbonded interactions but also to bonded interactions like bond, angle, and dihedral angle terms in molecular force fields,[4] (3) additional communication is not needed to support charge spreading and force interpolation in PME for sufficiently large cutoff values. It is stated that the midpoint method requires less communication latency for certain network topologies like the toroidal mesh network.[26]

Pair-wise Verlet list

Pair-wise Verlet list makes use of a cell-wise approach for making nonbonded interaction lists and storing particle data.[23] It has been developed mainly for efficient shared memory parallelization in a spatial decomposition scheme. By assuming only one MPI processor, the simulation space and a subdomain are identical to each other. This simulation space is divided into cells where the size in each dimension is greater than the pair-list cutoff distance, r_v. All the particle data are stored cell-wise, that is, specifically in each one of the cells. This grouping in a cell improves the memory locality of the particle data for operations on individual cells or pair of cells. For nonbonded interactions, cell pairs are distributed over threads and local Verlet pair-lists are generated for each cell pairs. High performance could be obtained through increasing efficiency of shared memory parallelization by making each thread access a different computer memory address. In our method, we first sort cell pair-list (i,j) in the ascending order of i first and in the ascending order of j next. We distribute the sorted cell pair-list (i,j) cyclically over OpenMP threads. Interactions on the cell pair-list (i,j) are considered by generating nonbonded lists of atoms in cell j for each atom in cell i. Thus, data of cell j are read and written more often. If there are N particles in cell i and M particles in cell j, for example, coordinates in cell i are read N times, and coordinates in cell j are read $N \times M$ times. Thus, it is more critical to avoid the concurrent access of data in cell j than in cell i. By distributing the sorted cell pair-list (i,j) cyclically over OpenMP threads, concurrent data access for cell j does not happen. For data in cell i, we first copy the data to private temporary array to avoid concurrent data access.

Midpoint cell method: Cell-wise approach of the midpoint scheme with pair-wise Verlet list

The midpoint cell method is an extension of the midpoint method[26] for hybrid (MPI+OpenMP) parallelization. Here, we consider the following modifications: (1) introduction of smaller cells in each subdomain, (2) grouping the particle data cell-wise as suggested in the pair-wise Verlet list, (3) the interaction cell is decided not from particle pairs but from cell pairs, and (4) distribution of cell pairs over OpenMP threads for shared memory parallelization. As mentioned in Pair-wise Verlet list, storing the data cell-wise improves localization of particle data and thereby enables efficient shared memory parallelization for bonded and nonbonded energy and force calculations. In particular, the modification (3) could improve performance by removing unnecessary calculations to find midpoints of all particle pairs at every MD time step. For example, midpoints between particles in cell **1** and those in cell **3** are always in cell **2**. In this case, by grouping data cell-wise and just defining the midpoint cell between cell **1** and cell **3** as cell **2**, we skip the procedure for checking midpoints for all the particle pairs in cell **1** with cell **3**. Moreover, it is sufficient to define the midpoint cells only once during the setup procedure.

The midpoint cell is not always defined uniquely for a given cell pair unlike the midpoint of particle pairs. At most, eight candidates exist as the midpoint cell in three-dimensional space. In Figure 2b, for a cell pair (**8, 10**), there are two possible candidates of the midpoint cells: cell **7** or cell **11**. We need to select a cell with the least number of particles as the midpoint cell among all candidates. This enables a good load

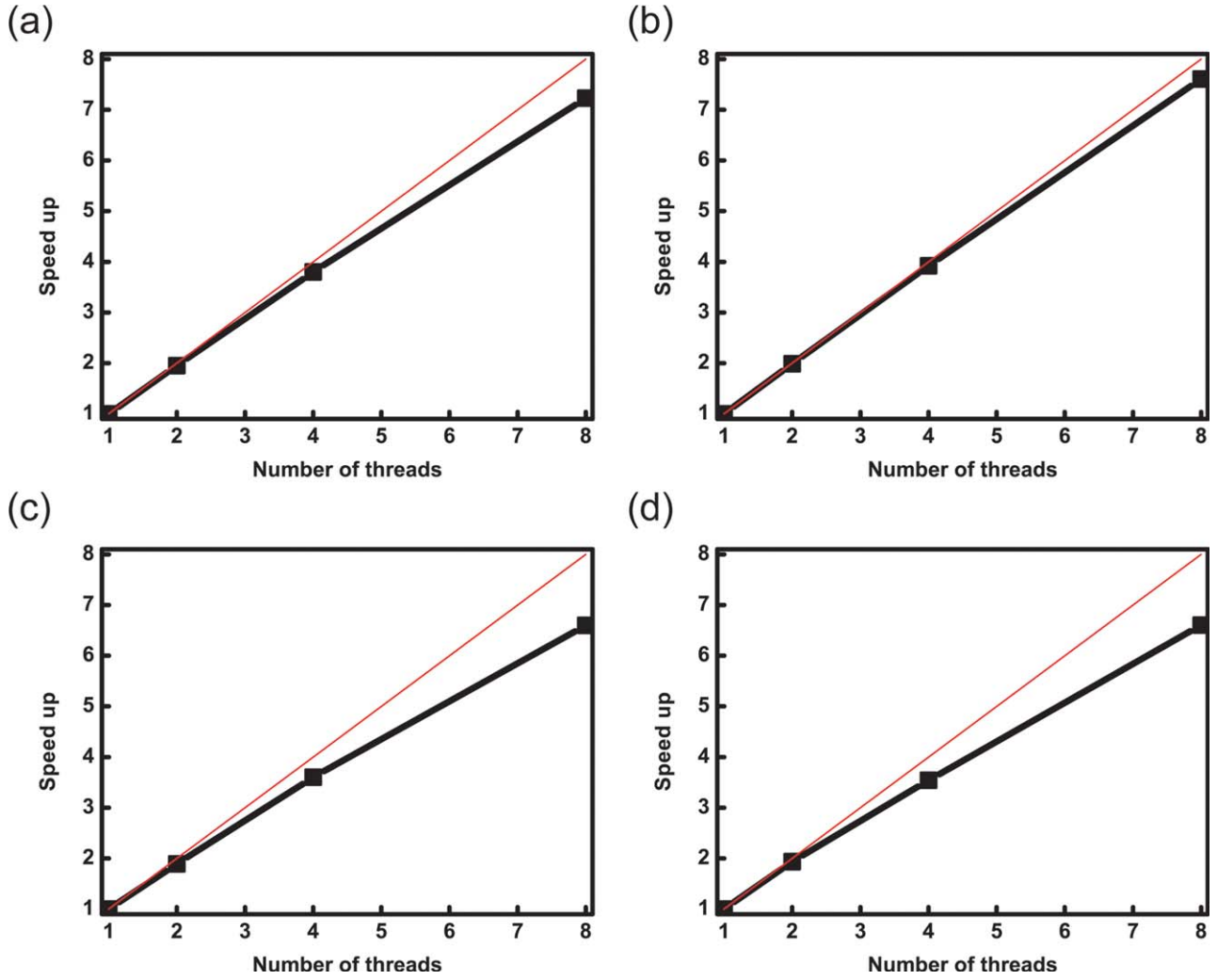

Figure 4. Speedup of total simulation time per MD step with OpenMP threads (1, 2, 4, and 8) for (a) DHFR with cutoff, (b) STMV with cutoff, (c) DHFR with PME, and (d) STMV with PME. The numbers of nodes used are 1 and 8 for DHFR and STMV, respectively. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

balance between MPI processors, because cells with less atoms finish calculations faster. In the example shown in Figure 2b, we assign cell **11** as the midpoint cell for cell pairs (**8, 10**). Consequently, the assignment of an interaction subdomain for a given particle pair can differ between the original midpoint method and our midpoint cell method.

In the original midpoint method, particle data is imported only from points within distance $r_c/2$ from the midpoint subdomain, as the domain size is normally adjusted to be equal to or greater than $r_c/2$. This requires a small volume of communication space. The communication space required in the midpoint cell method is slightly larger (Figs. 3a and 3b), because cell-wise definition of the midpoint cell is applied. Particle data is imported from all the adjacent cells of the target subdomain, which we call border neighborhood cells. However, in the midpoint cell method, there is neither no need for data sorting nor for checking distances from neighboring subdomains, which is necessary in the original method. Moreover, we can efficiently control send/receive data to/from the border neighborhood cells without communicating with corner cells of the border neighborhood (Fig. 3c). Thus, overall our scheme retains the advantage of small communication space present in the original midpoint method.

The major difference from the original pair-wise Verlet list developed by Gonnet[23] is the size of cells: in the original, cell sizes in each dimension are equal to or larger than r_v, whereas in the midpoint cell method, they are equal to or larger than $r_v/2$ where r_v is the pair-list cutoff which is larger than r_c around 1–2 Å. Cell pairs are generated not only for adjacent cells but also for next adjacent cells, after assigning the midpoint cells of cell pairs in each subdomain. Although pair distances between atoms in adjacent cells are small, we still generate a pair-list for adjacent cells for the following reasons: (1) for large processor numbers, cell sizes could become larger than $r_v/2$ from the assumption that cell numbers in each direction should be divisible by process numbers of the same direction, and (2) cell sizes again could be increased when considering constraints or dealing with specific lookup tables for water–water interactions.

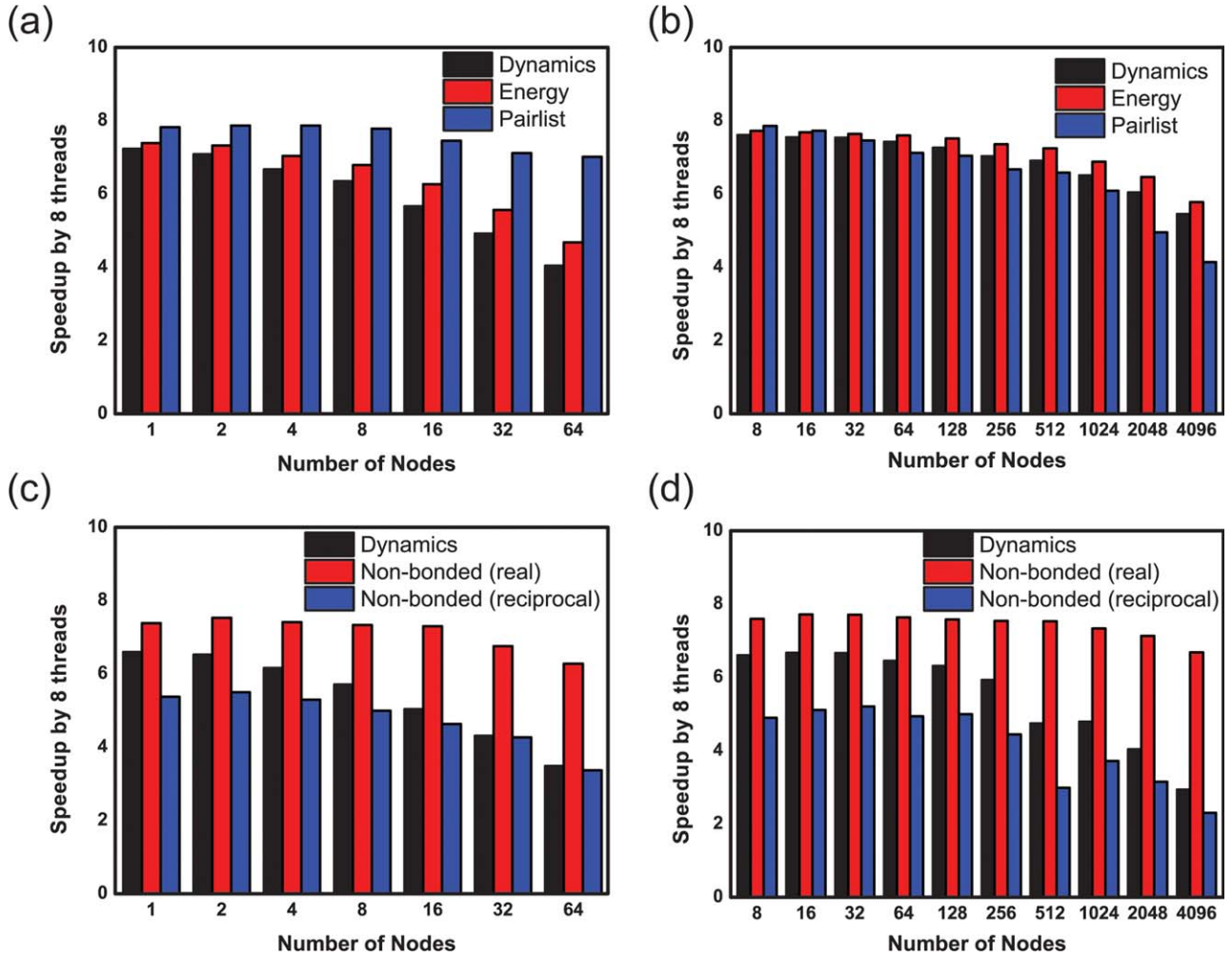

Figure 5. Speedup of OpenMP thread = 8 for (a) DHFR with CUTOFF, (b) STMV with CUTOFF, (c) DHFR with PME, and (d) STMV with PME. Dynamics includes the total simulation time without initial setup. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Results and Discussions

Following on the idea of the midpoint cell method, we implemented a new spatial decomposition scheme for hybrid (MPI-OpenMP) parallelization in our MD program package, GENESIS (Jung et al., Manuscript in preparation). We checked the numerical performance of MD simulations for two systems: protein dihydrofolate reductase (DHFR) [159 residues, 7023 water molecules, 23,558 atoms, r_c = 9.0 Å, r_v = 11.0 Å, PME grid size = (64,64,64)], and satellite tobacco mosaic virus (STMV) [299,855 water molecules, 773 ions, 1,066,628 atoms, r_c = 12.0 Å, r_v = 13.5 Å, PME grid size = (256,256,256)]. In both cases, migration of atoms and updating pair-lists were performed every 20 steps. These are well-known systems for benchmarking the performance of parallel MD programs. The input data generated are available from the NAMD webpage.[28] All the benchmark tests were done on the K computer at RIKEN.[29] We checked shared memory parallelization by comparing time statistics between 1 and 8 threads because a single CPU on the K computer has eight cores. The CHARMM force field[30,31] for proteins, DNA, and ions, and the TIP3P model[32] for water were used, and incorporated double precision arithmetic as in the CHARMM[4] and NAMD[5] programs. FFTE[33] was used for the fast Fourier transform (FFT) in the PME reciprocal calculation. The simulation has 1000 integration steps with a time step Δt 1 fs. In the case of PME, we used the recently developed inverse-lookup tables for short range nonbonded energy and gradient calculations,[34] and for parallelization of FFT, we assigned a volumetric decomposition scheme whereby all-to-all communications are performed in each dimension.

Benchmark performance of OpenMP parallelization

We first investigated the OpenMP parallelization performance for DHFR and STMV with a small number of MPI processors. Due to the different sizes of the systems and limitations of available memory on the K computer (16 GB per CPU), the minimum number of MPI processors available for DHFR and STMV are 1 and 8, respectively. Both cutoff with potential shift (CUTOFF) and PME were used to check performance of calculations of electrostatic interactions. In Figure 4, we plotted the speedup of simulation time per step with differing numbers of OpenMP threads (1, 2, 4, and 8). Using eight

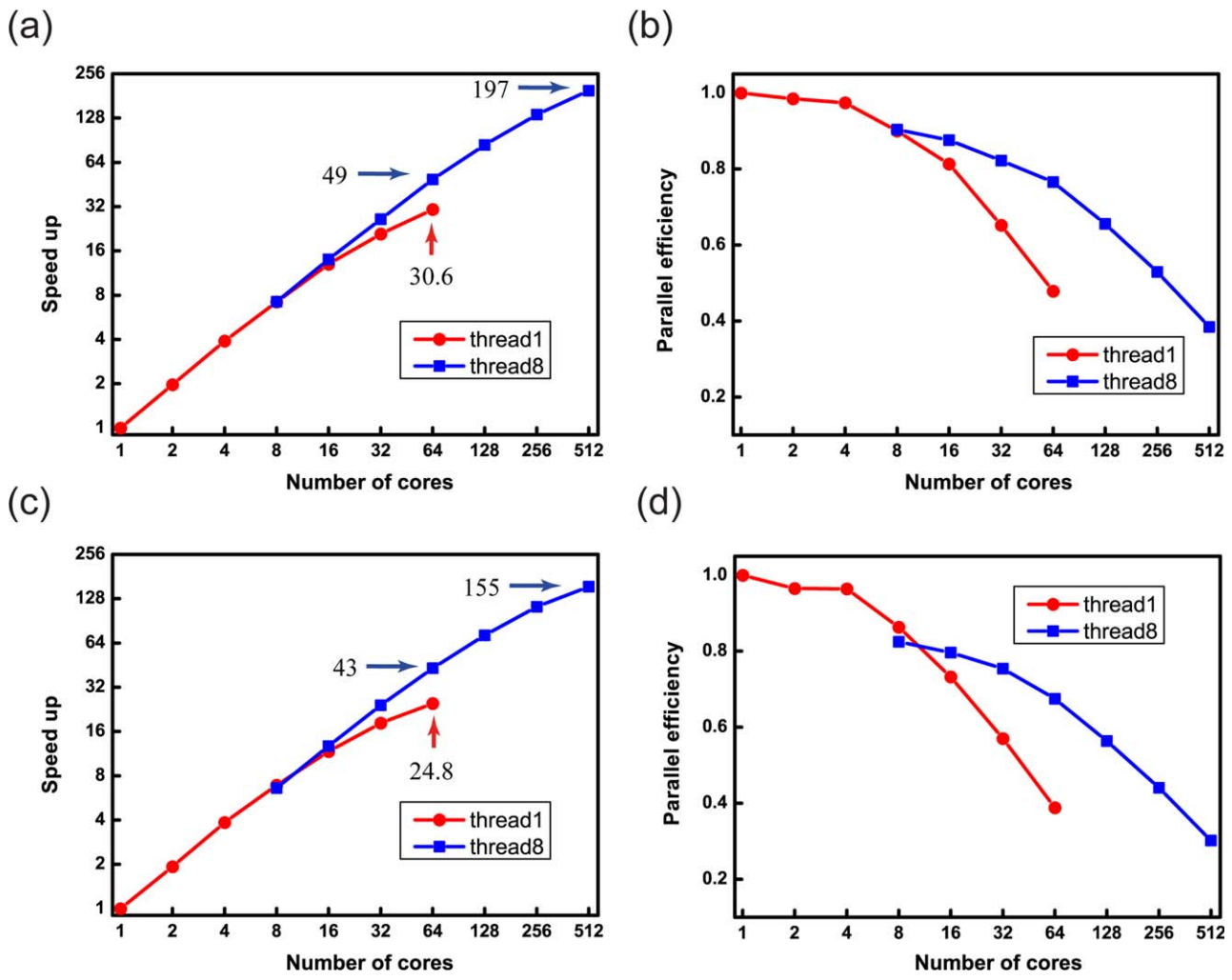

Figure 6. Benchmark MD simulation time for DHFR: (a) speedup and (b) parallel efficiency with CUTOFF, and (c) speedup and (d) parallel efficiency with PME. For single thread calculation, we make use of eight cores in a node such that the total number of nodes is identical to an eight threads case provided that we use the same number of cores. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

OpenMP threads, the MD simulation speedups for DHFR and STMV are >7.2 for CUTOFF and >6.5 in the case of PME, indicating that OpenMP shared memory parallelization in the midpoint cell method is hugely beneficial, at least with few MPI processors.

Benchmark performance of hybrid (MPI+OpenMP) parallelization

We next examined the performance of hybrid (MPI+OpenMP) parallelization with a different number of MPI processors. Again, both cutoff with potential shift and PME were used to check the performance. In MD simulations, the time-consuming exercises are (i) nonbonded energy and gradient calculations and (ii) updating of the nonbonded pair-list (Pairlist). Non-bonded interactions consist of short-range nonbonded interactions [Non-bonded (real)] and long-range electrostatic interactions [Non-bonded (reciprocal)]. In Figure 5, we show the speedup of Dynamics, Energy, and Pairlist for CUTOFF, and Dynamics, Non-bonded (real), and Non-bonded (reciprocal) for PME with eight OpenMP threads relative to single thread for each MPI processor. For example, the saving in time for dynamics on 64 nodes is judged by the speedup relative to 64 nodes using one core on each node. Here, Dynamics corresponds to the total simulation time per MD step. The average CPU time over all the MPI processors is taken into account for the time statistics of Pairlist and Non-bonded (real). In Dynamics and Non-bonded (reciprocal), all MPI processors have the same CPU time due to communications synchronization.

The benchmark results on Non-bonded (real) and Pairlist are directly related to the parallel efficiency of the midpoint cell method. In the case of CUTOFF and PME, the OpenMP speedup by eight threads for Energy and Nonbonded (real) is >6 even using the largest number of MPI processors (DHFR: 64, STMV: 4096). This guarantees good shared memory parallelization, as a result of near perfect load balance in our midpoint cell method. When very large MPI processors are used for a given system, the size of each subdomain diminishes. In the case of the smallest subdomain size using the largest MPI processors, the number of cells and boarder neighborhood cells are 8 and 56, respectively (the number of face, edge, and

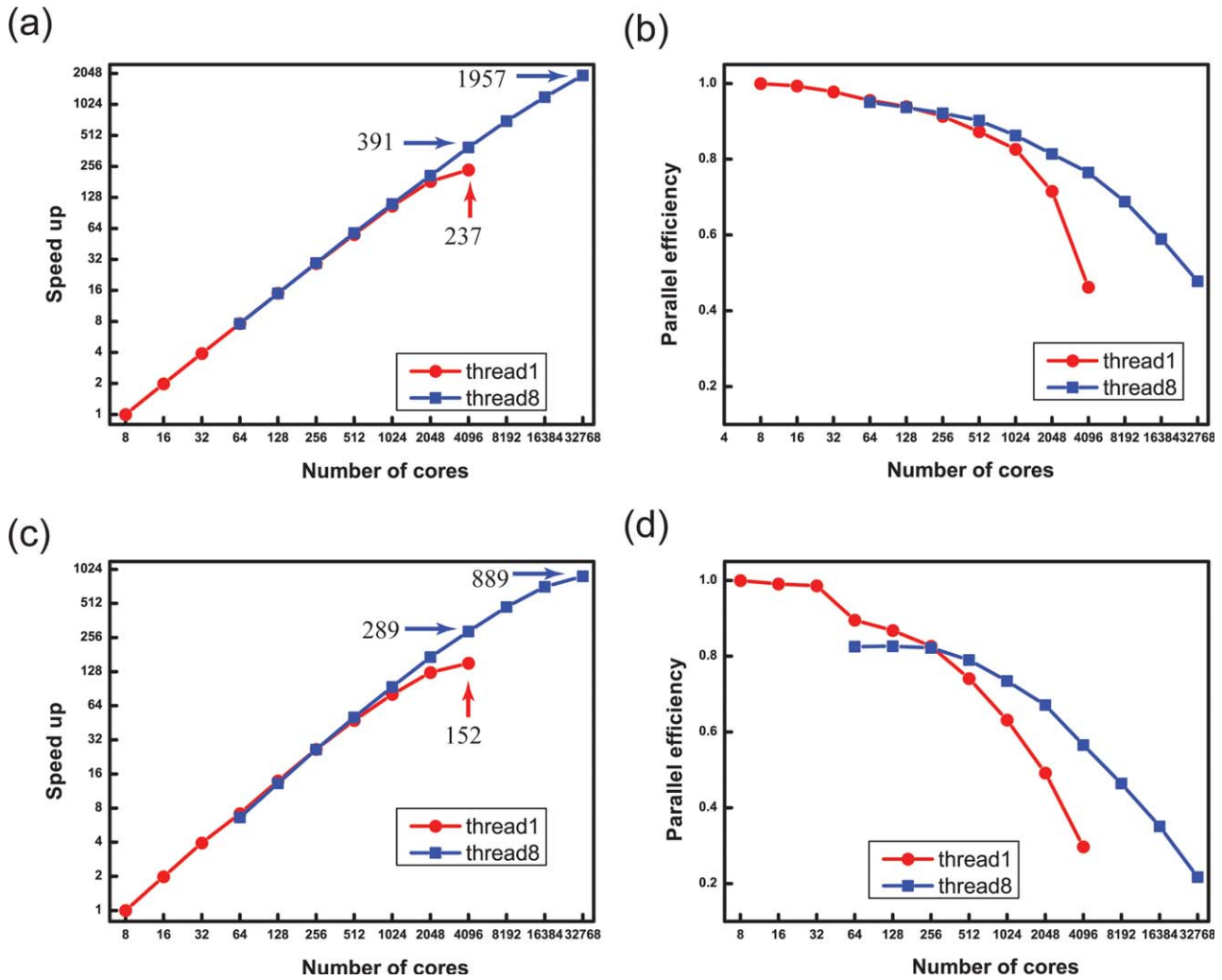

Figure 7. Benchmark MD simulation time for STMV: (a) speedup and (b) parallel efficiency with CUTOFF and (c) speedup and (d) parallel efficiency with PME, which are relative to eight cores. For single thread calculation, a single core is assigned for nodes up to 32 cores and eight cores for nodes with greater than or equal to 64 cores. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

vertex neighboring boarder cells are 24, 24, and 8, respectively). Even in these limited cases, the number of cell pairs is much greater than that of OpenMP threads and a good load balance is obtained. By comparing Figures 5a and 5b, we can understand how Pairlist shows better speedup than Energy for a small number of MPI processors, while the opposite pertains for many more. Evidently force accumulation impacts more in limiting shared memory parallelization for a small number of processors, and writing pair-list on memory is more crucial for large number of processors.

In the case of PME, the nonbonded calculations include both Non-bonded (real) and Non-bonded (reciprocal). For a small number of MPI processors and using eight OpenMP threads, the speedup exceeds 6.5 for the smallest MPI processors (DHFR: 1, STMV: 8), and even with many processors the simulation speedup is >3. In Figures 5c and 5d, the PME reciprocal calculation has less OpenMP speedup than for the real-space nonbonded calculation. This is mainly due to MPI all-to-all communications in PME reciprocal calculations, which could

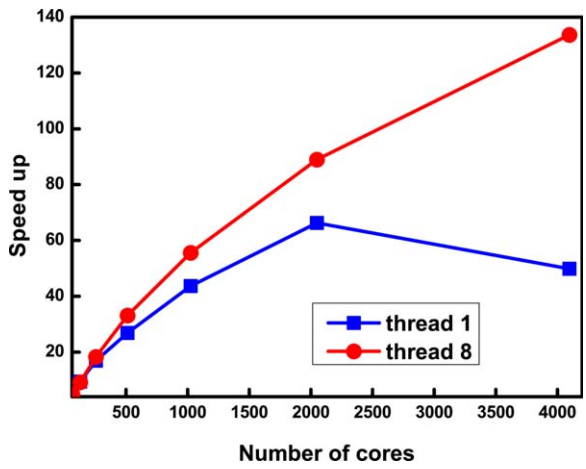

Figure 8. Speedup of the PME reciprocal calculation relative to eight cores for STMV. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

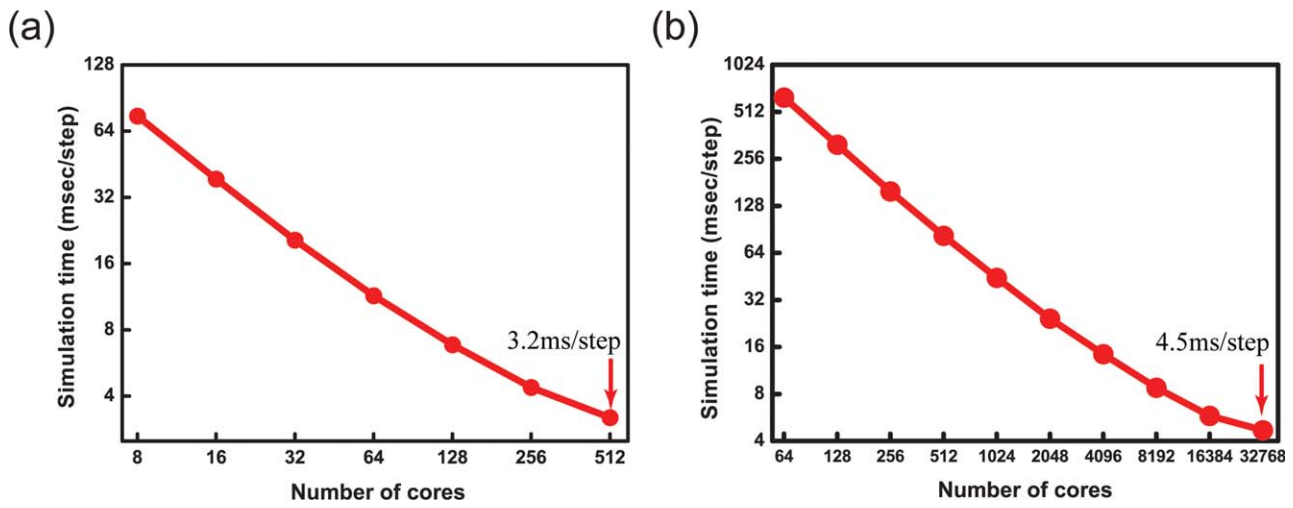

Figure 9. Time statistics of (a) DHFR and (b) STMV for PME electrostatic treatments. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

not be parallelized in shared memory. The speedups of Dynamics by eight OpenMP threads with the largest MPI processors are 3.5 and 3 for DHFR and STMV, respectively, suggesting that the overall performance of the hybrid (MPI+OpenMP) parallelization of the midpoint cell method is advantageous for not only CUTOFF but also for PME.

Comparison between hybrid (MPI+OpenMP) and MPI parallelization

Here, we discuss the advantage of hybrid (MPI+OpenMP) parallelization in MD simulations. As we mentioned, the most time-consuming part in the simulations is the nonbonded interactions which consist of short-range interaction and long-range interactions. CUTOFF includes only the short-range nonbonded interactions, whereas PME needs both the short-range nonbonded interactions (PME real) and long-range nonbonded interactions (PME reciprocal). We consider that hybrid (MPI-OpenMP) parallelization is useful for both interactions, although the midpoint cell method introduced here is directly related to the short-range nonbonded interaction. In the midpoint cell method, there is a limitation in the number of MPI processors (or the number of subdomains), as the minimum cell size is determined by the cutoff distance. The maximum numbers of MPI processors are 64 and 4096 for DHFR (r_c = 9.0 Å, r_v = 11.0 Å) and STMV (r_c = 12.0 Å, r_v = 13.5 Å), respectively. As each CPU of K computer contains eight cores, the maximum numbers of CPU cores are 512 and 32,768 cores for DHFR and STMV, respectively. In Figures 6 and 7, the performances of total computational time per a single MD step using hybrid parallelization are compared to those using MPI parallelization with single thread (the total number of nodes used in hybrid and MPI only are identical to each other for a given core number). In DHFR, we set the computational time per MD step for a single MPI with single thread as a standard and plotted the speedup ratio as a function of the number of CPU cores. The maximum speedups for DHFR on MPI parallelization are 30 and 25 in CUTOFF and PME schemes, whereas they exceed more than 150 on hybrid parallelization (197 and 155 for CUTOFF and PME schemes, respectively). In STMV, we investigated the speedups relative to the simulation time using eight MPIs with single threads, and the results are depicted in Figure 7. The maximum speedups with STMV on MPI parallelization with single thread are 237 and 152 in CUTOFF and PME schemes, whereas they exceed more than 880 with hybrid parallelization using eight OpenMP threads (1957 and 889 for CUTOFF and PME schemes, respectively).

In long-range electrostatic interactions, PME reciprocal calculation is known to be the most problematic part of MPI parallelization. This includes FFT, which requires all-to-all MPI communication. The computational time is easily saturated even with a small number of MPI processors. For this processing, hybrid (MPI+OpenMP) is better than MPI, if we compare their performance using the same number of CPU cores. In Figure 8, speedup of the PME reciprocal calculations with one thread and eight threads are plotted as a function of CPU cores (from 64 to 4096). A noticeable difference in performance exists and eight-threads calculations improve performance as the number of cores increases. This is because the number of MPI processors dealing with FFT communications is reduced eightfold using eight OpenMP threads. Thus, hybrid parallelization comes to the fore in communicating PME reciprocal calculation in FFT. The necessary condition of OpenMP threads with MPI is taken care of in our approach—there is no loss of performance in computing real space nonbonded interactions and building the pair-list. These observations prove the good scaling behavior of our OpenMP implementation.

Finally, the total simulation time for DHFR and STMV using the PME electrostatic treatment was checked and is depicted in Figure 9. The lookup table shows that we can perform a single step of MD simulation within 4.5 ms even for a 1 million particle system.

Conclusions

In this article, we describe a new hybrid (MPI+OpenMP) parallelization scheme by combining the midpoint method with the pair-wise Verlet list scheme and demonstrate its parallel

efficiency in MD simulations on the K computer. For both CUT-OFF and PME, a small system (DHFR) and a large system (STMV) are scalable up to 512 and 32,768 cores by combining eight OpenMP threads with 64 and 4096 MPI processors. Even for the large system, a step of 4.5 ms is available. This method is, in particular, suitable to MD simulations of large biomolecular systems on massively parallel supercomputers.

Acknowledgment

Parts of the results were obtained using the K computer at the RIKEN Advanced Institute for Computational Science (Proposal number ra000009).

Keywords: molecular dynamics · hybrid parallelization · midpoint cell method · domain decomposition

How to cite this article: J. Jung, T. Mori, Y. Sugita. *J. Comput. Chem.* **2014**, *35*, 1064–1072. DOI: 10.1002/jcc.23591.

[1] M. Karplus, J. A. McCammon, *Nat. Struct. Biol.* **2002**, *9*, 646.
[2] J. L. Klepeis, K. Lindorff-Larsen, R. O. Dror, D. E. Shaw, *Curr. Opin. Struct. Biol.* **2009**, *19*, 120.
[3] D. A. Case, T. E. Cheatham, III, T. Darden, H. Gohlke, R. Luo, K. M. Merz, Jr., A. Onufriev, C. Simmerling, B. Wang, R. J. Woods, *J. Comput. Chem.* **2005**, *26*, 1668.
[4] B. R. Brooks, C. L. Brooks, III, A. D. Mackerell, Jr., L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, M. Karplus, *J. Comput. Chem.* **2009**, *30*, 1545.
[5] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, K. Schulten, *J. Comput. Chem.* **2005**, *26*, 1781.
[6] S. Kumar, C. Huang, G. Zheng, E. Bohm, A. Bhatele, J. C. Phillips, H. Yu, L. V. Kale, IBM *J. Res. Dev.* **2008**, *52*, 177.
[7] Y. Sun, G. Zheng, C. Mei, E. Bohm, J. Phillips, T. Jones, L. Kale, In proceedings of the 2012 ACM/IEEE conference on Supercomputing (SC12); Salt Lake City, Utah, **2012**.
[8] B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl, *J. Chem. Theory Comput.* **2008**, *4*, 435.
[9] M. J. Abraham, *J. Comput. Chem.* **2011**, *32*, 2041.
[10] S. Pronk, S. Pall, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel, B. Hess, E. Lindahl, *Bioinformatics* **2013**, *29*, 845.
[11] I. T. Todorov, W. Smith, K. Trachenko, M. T. Dove, *J. Mater. Chem.* **2006**, *16*, 1911.
[12] K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, J. K. Salmon, Y. Shan, D. E. Shaw, In ACM/IEEE Conference on Supercomputing (SC06); Tampa, FL, **2006**.
[13] D. E. Shaw, M. M. Deneroff, R. O. Dror, J. S. Kuskin, R. H. Larson, J. K. Salmon, C. Young, B. Batson, K. J. Bowers, J. C. Chao, M. P. Eastwood, J. Gagliardo, J. P. Grossman, C. R. Ho, D. J. Ierardi, I. Kolossvary, J. L. Kelpeis, T. Layman, M. A. McLeavey, M. A. Moraes, R. Mueller, C. Edward, Y. Shan, J. Spengler, M. Theobald, B. Towles, S. C. Wang, In 34th Annual International Symposium on Computer Architecture (ISCA'07), San Diago, CA, **2007**.
[14] B. G. Fitch, A. Rayshubskiy, M. Eleftheriou, T. J. C. Ward, M. Giampapa, Y. Zhestkov, M. C. Pitman, F. Suits, A. Grossfield, J. Pitera, W. Swope, R. H. Zhou, S. Feller, R. S. Germain, In Computational Science—ICCS 2006, Part 2, Proceedings, Vol. 3992; Springer; Berlin, Heidelberg, **2006**; pp. 846–854.
[15] B. G. Fitch, A. Rayshubskiy, M. Eleftheriou, T. J. C. Ward, M. E. Giampapa, M. C. Pitman, J. W. Pitera, W. C. Swope, R. S. Germain, IBM *J. Res. Dev.* **2008**, *52*, 145.
[16] K. J. Oh, J. H. Kang, H. J. Myung, *Comput. Phys. Commun.* **2012**, *183*, 440.
[17] Y. Andoh, N. Yoshii, K. Fujimoto, K. Mizutani, H. Kojima, A. Yamada, S. Okazaki, K. Kawaguchi, H. Nagao, K. Iwahashi, F. Mizutani, K. Minami, S. Ichikawa, H. Komatsu, S. Ishizuki, Y. Takeda, M. Fukushima, *J. Chem. Theory Comput.* **2013**, *9*, 3201.
[18] K. Y. Sanbonmatsu, *Curr. Opin. Struct. Biol.* **2012**, *22*, 168.
[19] F. Khalili-Araghi, J. Gumbart, P. C. Wen, M. Sotomayor, E. Tajkhorshid, K. Schulten, *Curr. Opin. Struct. Biol.* **2009**, *19*, 128.
[20] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, L. G. Pedersen, *J. Chem. Phys.* **1995**, *103*, 8577.
[21] L. Verlet, *Phys. Rev.* **1967**, *159*, 98.
[22] M. P. Allen, D. J. Tildesley, Computer Simulation of Liquids; Clarendon Press, Oxford University Press: Oxford England, New York, **1990**.
[23] P. Gonnet, *J. Comput. Chem.* **2012**, *33*, 76.
[24] J. A. Anderson, C. D. Lorenz, A. Travesset, *J. Comput. Phys.* **2008**, *227*, 5342.
[25] P. H. Colberg, F. Hofling, *Comput. Phys. Commun.* **2011**, *182*, 1120.
[26] K. J. Bowers, R. O. Dror, D. E. Shaw, *J. Chem. Phys.* **2006**, *124*, 184109.
[27] D. E. Shaw, *J. Comput. Chem.* **2005**, *26*, 1803.
[28] Available at: http://www.ks.uiuc.edu/Research/namd, accessed on June, 2012.
[29] Available at: http://www.aics.riken.jp, accessed on April, 2012.
[30] A. D. MacKerell, D. Bashford, M. Bellott, R. L. Dunbrack, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, M. Karplus, *J. Phys. Chem. B* **1998**, *102*, 3586.
[31] J. Huang, A. D. MacKerell, *J. Comput. Chem.* **2013**, *34*, 2135.
[32] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, M. L. Klein, *J. Chem. Phys.* **1983**, *79*, 926.
[33] Available at: http://www.ffte.jp/, accessed on November, 2011.
[34] J. Jung, T. Mori, Y. Sugita, *J. Comput. Chem.* **2013**, *34*, 2412.

Received: 19 December 2013
Revised: 3 March 2014
Accepted: 6 March 2014
Published online on 23 March 2014