Molecular Dynamics Simulations Accelerated by GPU for Biological Macromolecules with a Non-Ewald Scheme for Electrostatic Interactions

Tadaaki Mashimo,[†,‡] Yoshifumi Fukunishi,[§] Narutoshi Kamiya,[¶] Yu Takano,[¶,$] Ikuo Fukuda,[¶] and Haruki Nakamura*[,§,¶]

[†]Japan Biological Informatics Consortium (JBIC), 2-3-26, Aomi, Koto-ku, Tokyo 135-0064, Japan

[‡]Information and Mathematical Science Bio Inc., Owl Tower, 4-21-1, Higashi-Ikebukuro, Toshima-ku, Tokyo 170-0013, Japan

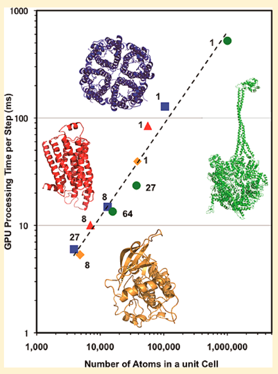
[§]Molecular Profiling Research Center for Drug Discovery (molprof), National Institute of Advanced Industrial Science and Technology (AIST), 2-3-26, Aomi, Koto-ku, Tokyo 135-0064, Japan

[¶]Institute for Protein Research, Osaka University, 3-2 Yamadaoka, Suita, Osaka 565-0871, Japan

[$]JST, CREST, 4-1-8 Honcho, Kawaguchi, Saitama 332-0012, Japan

Supporting Information

ABSTRACT: A molecular dynamics (MD) simulation program for biological macromolecules was implemented with a non-Ewald scheme for long-ranged electrostatic interactions and run on a general purpose graphics processing unit (GPU). We recently developed several non-Ewald methods to compute the electrostatic energies with high precision. In particular, the zero-dipole summation (ZD) method, which takes into account the neutralities of charges and dipoles in a truncated subset, enables the calculation of electrostatic interactions with high accuracy and low computational cost, and its algorithm is simple enough to be implemented in a GPU. We developed an MD program with the space decomposition algorithm, myPresto/psygene, and applied it to several biological macromolecular systems with GPUs implementing the ZD method. Rapid computing performance with high accuracy was obtained.



INTRODUCTION

Molecular dynamics (MD) simulations are a popular and powerful tool for analyzing the structural and functional features of biological macromolecules, such as proteins, DNAs, lipids, and their complexes,[1−3] and they are also frequently applied for drug development.[4,5] Therefore, many efforts have been focused on developing MD simulations with high accuracy and low computational cost, to represent macromolecular systems as realistically as possible.

In particular, specialized hardware systems, including MD-GRAPE,[6,7] MD-ENGINE,[8] and ANTON,[3] have been developed for such MD simulations. However, these specialized hardware systems are quite expensive, and only the MD simulations with their individual particular algorithms can be performed.

On the contrary, a general purpose graphics processing unit, GPU, is relatively cheaper than the specialized hardware, and it can be used for various applications. GPUs were originally designed as specialized hardware for accelerating graphical operations, but they have grown into exceptionally powerful, general purpose computing engines. Modern GPUs are 10−100 times faster than conventional CPUs, in terms of raw computing power.[9,10] The computer program codes for GPU can be developed based on the CUDA programming language, which is a C-like language.[11] As a result, the use of GPUs for general purpose computing has become a rapidly growing field of research. Many important algorithms have been implemented on GPUs, often leading to a performance gain of one- to two-orders of magnitude over the CPU implementations.[12]

Several studies have assessed GPU implementations of specific algorithms for MD simulations. Elsen et al. implemented a simple implicit solvent model (distance dependent dielectric),[9] and Stone et al. examined a GPU implementation for electrostatics.[10] Anderson et al. implemented several algorithms, including integrators, neighbors lists, Lennard-Jones, and force computations for the covalent bonds but not for the energies of torsions or constraints.[13] Jha et al. provided a GPU implementation of the pre-averaging method, overcoming the issue of the particular cutoff length, and obtained significant faster results than those of the CPU implementation of the conventional method.[14] Nguyen et al. evaluated the performance of revised Wolf methods on hybrid high performance computers with GPU accelerators.[15] Tanner et al. efficiently utilized GPUs and CPUs simultaneously for fast and accurate MD simulations with the generalized Born/

Received: April 27, 2013
Published: November 5, 2013

solvent-accessible surface area implicit solvent model.[16] Popular programs, including AMBER,[17,18] Gromacs,[19] NAMD,[20] and OpenMM[21] have recently been implemented on GPUs.

One of the major difficulties in executing precise and rapid MD simulations for biological macromolecules is the calculation of the electrostatic interactions, because of their long-ranged nature. It is well-known that a simple cutoff of the electrostatic interactions resulted in many artifacts.[22] On the contrary, as an alternative approach, the usage of lattice sum techniques, such as the Ewald method[23] and the Particle mesh Ewald (PME)[24,25] method, is the most popular standard, assuming the periodic boundary condition. However, the lattice sum techniques cannot readily perform rapid computations in the reciprocal space with parallel computing, because communication between many processors is required.

Recently, the artifacts produced by the simple cutoff method have been significantly reduced by several new approaches, referred to as the non-Ewald methods, which were reviewed in detail elsewhere.[22] Wu and Brooks[26,27] derived the isotropic periodic sum method, by assuming the isotropic periodicity that is defined by the images of the local region statistically distributed in an isotropic and periodic manner. It is a general method for calculating the long-range interactions[28] and has been successfully applied to a number of systems, including lipid bilayers and monolayers.[29] We have developed the zero-dipole summation (ZD) method,[30−32] which takes into account the neutralities of charges (zero-monopole) and dipoles (zero-dipole) in a truncated subset. The ZD method can be viewed as an extension of the other non-Ewald method developed by Wolf et al.,[33,34] and it provided much more accurate electrostatic energies than a simple cutoff method for a liquid NaCl system,[30] a pure TIP3P water system,[31] and even an inhomogeneous membrane protein system.[32] The algorithm of the ZD method is represented by a simple pairwise function sum and has high portability from CPU to GPU. Therefore, it is expected to provide high computational performance for applications in CPUs and GPUs.

We implemented the ZD method into our spatial subdivision MD code, myPresto/psygene, and developed the special code, myPresto/psygene-G, with a GPU port of the pairwise nonbonded interaction calculations, which consist of the van der Waals interactions and the electrostatic interactions with the ZD method. The GPU port subdivides the assigned subspaces, executes local searches, and calculates the pairwise nonbonded interactions in the cutoff sphere, as described in the "CUDA Particles", a technical report contained in the CUDA SDK.[34]

In this paper, we examined the quality and performance of the MD simulations for four protein systems: epidermal growth factor receptor (EGFR) in aqueous solvent with 38,453 atoms, β_2-adrenergic receptor (β_2AR) in a membrane with 56,121 atoms, Aquaporin 4 (AQP4) in a membrane with 104,415 atoms, and Dynein in aqueous solvent with 1,004,847 atoms. The computations of these systems were performed by myPresto/psygene on a typical PC cluster and by myPresto/psygene-G on a GPU cluster system. Consequently, significant acceleration of the MD simulations was observed with small RMS force errors and good conservation of total energy, and both afforded acceptable numerical accuracy.

■ METHODS

Zero-Dipole Summation Method. The details of the algorithm of the ZD method have been described elsewhere,[30,31] and they are briefly summarized here. The total Coulombic electrostatic energy, E_{total}, can be described as follows:

$$E_{total} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j}\frac{q_i q_j}{r_{ij}} \tag{1}$$

$$= \frac{1}{2}\sum_{i=1}^{N}\sum_{j} q_i q_j \frac{\mathrm{erf}(\alpha r_{ij})}{r_{ij}} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j} q_i q_j \frac{\mathrm{erfc}(\alpha r_{ij})}{r_{ij}} \tag{2}$$

Here, q_i is the atomic charge of the ith atom, and r_{ij} is the distance between the ith atom and the jth atom. The manner of the summation with respect to j depends on the boundary conditions. In an ordinary cutoff-based simulation, the minimum image convention is often considered with respect to j, while in the periodic boundary condition, j should be taken over all particles, except i, in the original MD cell and over all particles in all of the image cells. The error function and the complementary error function are designated as *erf* and *erfc*, respectively, with a damping factor α (≥ 0).

In the Ewald method with the periodic boundary conditions, the first term of eq 2 can be evaluated via the Fourier space term and the self-energy term. While, as in the non-Ewald methods,[22] such as the Wolf method[33,34] and the ZD method,[30,31] the first term of eq 2 can be approximated as $-(\alpha/(\pi^{1/2}))\Sigma_{i=1}^{N}q_i^2$ for a small α. In fact the first term completely vanishes for $\alpha = 0$. As for the second term of eq 2, the evaluation with an appropriately short cutoff truncation can be employed for a relatively large α, as in the Ewald approach. Instead, in order to provide an efficient estimate of the second term even for a small α, the theoretical frame of the ZD method assumes that the interaction contribution is counted in a neutralized subset M_i that consists of certain particles inside the cutoff sphere of the individual ith atom. The subset M_i, which includes the ith atom, is characterized such that the total sums of the charges and dipole moments are both zero in M_i, and any atoms not belonging to M_i but inside the cutoff sphere are located close to the surface. The creation of subset M_i is not actually required in practical applications, but M_i is introduced conceptually to express the theoretical viewpoint. That is, by mathematical considerations with respect to the approximation of the excess energy, we can convert the summation of the original potential function $erfc(\alpha r)/r$ over M_i into an ordinary pairwise summation of a suitably reconstructed potential function inside the cutoff sphere.[30] Eq 2 is thus approximated as

$$E_{total} \approx E_{total}^{ZD} = \frac{1}{2}\sum_{i=1}^{N}\sum_{(j\neq i, r_{ij}<r_c)} q_i q_j[u(r_{ij}) - u(r_c)] - \left[\frac{u(r_c)}{2} + \frac{\alpha}{\sqrt{\pi}}\right]\sum_{i=1}^{N} q_i^2 \tag{3}$$

$$u(r) = \frac{\mathrm{erfc}(\alpha r)}{r} + \left[\frac{\mathrm{erfc}(\alpha r_c)}{2r_c} + \frac{\alpha}{\sqrt{\pi}}\exp(-\alpha^2 r_c^2)\right]\frac{r^2}{r_c^2} \tag{4}$$

For a general system with molecules having covalent bonds, the relevant modifications to eq 3 are required.[31]

From the neutralization concept, the ZD summation method can be viewed as an extension of the Wolf method,[33] since the latter only assumes the charge neutrality. The Wolf method can be derived, in fact, simply when $u(r) = erfc(\alpha r)/r$ is applied instead of eq 4, for which the accuracy of the method is governed, in a practical cutoff distance region, by the damping effect provided by a suitable choice of a positive value of α. In contrast, the opposite limit, i.e., the no-damping with $\alpha = 0$, can rather be expected in the ZD method in many systems. In this limit, eq 4 becomes a simple form

$$u(r) = \frac{1}{r}\left[1 + \frac{r^3}{2r_c^3}\right] \tag{5}$$

which is, in turn, related to the functional forms used in the reaction field method and the pre-averaging method.[22] One advantage of using this limit in the ZD method is in the accuracy, as described in our previous reports. In fact, detailed investigations on the ZD method for the TIP3P water system[31] and the β_2AR system with explicit membrane and solvent molecules,[32] with various combinations of the cutoff distance r_c and the damping factor α, showed very accurate energies and stable MD simulations even with short r_c, 12 to 14 Å, and small α values less than 0.1 Å^{-1}. In particular, MD simulations using $\alpha = 0$ provided the most accurate energies with those r_c values.[31,32] The other advantage of the $\alpha = 0$ limit is the speed up of computation without using the complementary error function. Taking these advantages of the ZD method, we challenged to attain as high speed up of MD simulation as possible in the current study.

We demonstrated the accuracy of the ZD method elsewhere. At the cutoff distance of 12 Å, the deviations of the total Coulombic energy from that obtained by the Ewald method were about 0.03% in the ionic liquid system,[30] 0.01% in the bulk water system,[31] and 0.04% in the membrane protein system.[32] In addition, the dielectric properties of water were well reproduced in the ZD method. For example, the distance-dependent Kirkwood factor was very similar to that obtained by the PME method within a 24 Å region, in spite of the use of a 12 Å cutoff distance in the ZD method. Such dielectric properties are very sensitive to the method used to evaluate the electrostatic interactions, and many cutoff-based (CB) methods have not produced accurate results. For example, although the reaction field method was derived from a certain physical consideration and has a long history, it often failed to produce accurate dielectric properties, as reviewed and investigated in detail in our previous work.[22,31,36] Reproducing such fundamental properties of the water is critical to conduct reliable simulations on bimolecular systems in vivo.

We also note that the ZD method is not simply an alternative to the PME method. The ZD method is irrelevant to the boundary condition, in principle. In contrast, the lattice sum methods, including the PME method, require the periodic boundary condition. However, except for the ideal crystal state, such exact periodicity is far from reality in bimolecular systems. Due to the existence of such a qualitative difference between the lattice sum method and the CB method, it should be crucial to develop the CB methods, such as the ZD method, and to provide a choice. Increasing the speed of the code implementing the ZD method is important from the viewpoint of this development, because it will significantly facilitate the simulations of large and complicated molecular systems via the ZD method, one of the accurate CB methods. Thus, the purposes of the current study are not only to enhance the speed but also to aim at deeper understanding and further development of the CB methods.

MD Simulation with the Space Decomposition Algorithm. Currently, massively parallel computers are composed of a distributed memory system of a few hundred to up to tens of thousands of accelerators, such as GPUs.[37] Thus, the MD program for massively parallel computers should combine the implementation of CPUs and accelerators, which execute processes in parallel while being accelerated by an accelerator and exchanging the data through high-speed network communication.

Psygene-G, a derivation of myPresto/cosgene,[38] is an MPI/GPU-combined parallel program with an NVIDIA GPU as its accelerator, which was developed for massively parallel computers. This program takes a system's constituent atoms and divides them in a coordinate space. The MD simulation of the atoms belonging to the subspaces is assigned to the components (MPI processes) of the parallel computer. Among the assigned MD calculations, the pairwise interaction calculations of the nonbonded terms are computed on GPUs. Along with the data transfer required for the migration of atoms that extend across subspaces and the control of temperature and pressure, the mutual transfer of the atomic data between neighboring subspaces needed for the pairwise interaction calculations uses the MPI communication. The parallel execution of psygene-G by more than a hundred computers accelerated by GPU could allow it to handle a system consisting of over a few million atoms.

In the pairwise interaction calculation, which consists of the electrostatic interaction and the van der Waals interaction, the forces are calculated approximately, by taking into account the influence of neighboring atoms within a short distance of the cutoff distance. However, a neighboring atom search requires $O(N^2)$ calculation time. The space decomposition method, which streamlines this search, reduces the number of calculations by performing pairwise interaction calculations for only the atom pairs included in the relevant and neighboring decomposed regions.[39]

In psygene-G, the components of the decomposed system are called cells. The cells are obtained by decomposition of the cuboid system along three axes by an arbitrary number. However, in the current implementation, the number of cells must be adjusted so that the length of one side exceeds the cutoff length by at least 2-fold. Each cell manages its respective region boundaries, constituent atom quantity, and data of each constituent atom. Then, the entire system is formed by integrating all of the cells. The cells are the smallest unit of the calculated regions handled by a parallel computer's CPU core. One CPU core as an MPI process is assigned to one cell. Since it is conceivable that, depending on the parallel computer, there may not be enough CPU cores to correspond to the optimal number of cells for a system, no upper limit is placed on the number of cells that can be assigned to a single CPU core. When the number of cells is greater than the number of CPU cores, multiple cells are assigned to a single CPU core, and they are serially processed by round robin scheduling.

Generally, one step in the MD calculation is divided into calculations of force and integral of the equations of motion, where the force calculations are further divided into calculations of bonded terms and nonbonded terms. The calculation of nonbonded terms is further partitioned in three ways: calculations of the van der Waals force terms, short-range electrostatic force terms, and long-range electrostatic force terms. Whereas the calculations of van der Waals force terms and short-range electrostatic force terms are directly computed by short-range pairwise interaction calculations, the calculation of long-range electrostatic force terms is, in general, computed approximately by methods such as Ewald[23] or PME.[24,25]

In psygene-G, the processing of a single step in the MD calculation is assigned to each cell and executed in parallel. When doing so, the pairwise interaction calculations, which consume the majority of the single step MD computation time, are calculated on the GPU using the ZD method, as shown in eqs 3 and 4.

Communication between Processors. For calculations of the pairwise interactions performed in each cell, it is necessary to calculate the interactions between the atoms in the relevant cell and the atoms in its neighboring cells. As shown in Figure 1, we define the 'import region' of a cell as the peripheral space in the neighboring cells, to which the interactions can extend from the relevant cell. This peripheral space should touch the relevant cell, and its thickness is the cutoff distance. Similarly, we define the 'export region' in the relevant cells with atom information shared by the neighboring cells. Thus, in the three-dimensional space decomposition, we need to send the atom information in the export region in a cell to its surrounding 26 cells and to receive it from the import region.

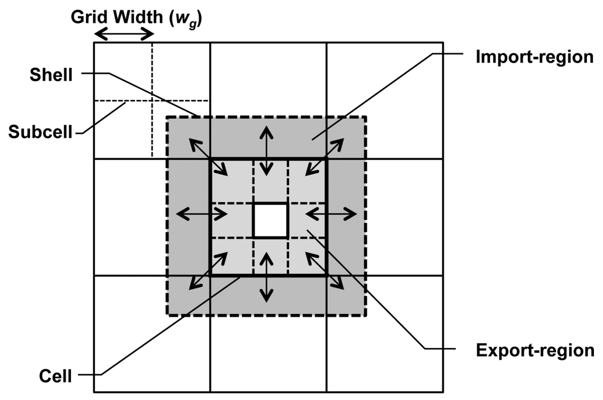


Figure 1. Space decomposition method used in psygene-G. Each cell is a decomposed system's component. Atoms in the ″Import-region″ in the contact cells are colored dark gray, and those in the ″Export-region″ are colored light gray in the center cell. The "Export-region" atoms are required for the nonbonded force calculations and are updated at every MD time step. The shell consists of the center cell and the import-regions in the peripheral 26 cells.

In psygene-G, the communication between the processors is conducted for exchanging atom information between the import and export regions in neighboring cells as well as for actualizing the migration of atoms that straddle the cells. This communication is performed using the Message Passing Interface (MPI) communication. MPI provides the operations of the collective communication and the synchronous and asynchronous one-to-one communication. Here, we applied the asynchronous one-to-one communication, in order to reduce the communication overhead. In a parallel computer, where simultaneous communication between multiple nodes is possible, the MPI communication is performed only with the neighboring cells in the 26 directions, so that the communication between the non-neighboring cells can be executed in parallel.

Atom information is exchanged between the import and export regions, and the migration traffic of the atoms straddling the cells depends on the surface area of the relevant cell. The information about the import and export regions is updated at each simulation step. A cube is the shape with the minimum surface for the import and export regions. Therefore, in principle, the traffic cost is minimized when the cell is cube-shaped. The amount of atom movement in one step of the MD procedure is so small that very few atoms migrate across the cells. This means that most of the traffic consists of atom data exchange between the import and export regions. In addition, the collective communication is also used in MPI for the control of temperature and pressure in psygene-G. Although the amount of transferred data is small, the cost of the collective communication can become very large, and thus it cannot be ignored for the large-scale parallel computation.

Calculations of Pairwise Interaction Terms on GPUs. The implementation of psygene-G for GPU was designed for the architecture where an NVIDIA GPU with CUDA[11] parallel computation was assumed as the computational environment. The NVIDIA's GPU hardware is optimized for the parallel processing of data, and its component, the GPU core, corresponds to the 'thread' in CUDA. In parallel computation, the Single Instruction Multiple Data (SIMD) processing is performed with 32 threads as the vector width. Many threads are hierarchically managed, where a set of threads is a 'block' and a set of blocks is called a 'grid'. The memory model corresponds to the thread hierarchy, where a register on a chip is occupied by only a single thread. The shared memory on the chip (48KB) is available only throughout a single block, and the global memory by DRAM (6GB) is available throughout a single grid. In addition, we can use a constant memory space (64KB), which stores constant values for the MD calculation and a texture memory for random access. By using the cache, rapid access to these memories is available.

The calculations for the pairwise interactions were accelerated by NVIDIA's GPU, including calculations by the ZD method for the electrostatic interactions, as in the following procedures. In the pairwise interaction calculations for nonbonded terms on the GPU, the atom information managed by the cells was first transferred to the GPU by CUDA API. Then, the single-precision calculations of the van der Waals and electrostatic force terms were executed on the GPU. Finally, the calculated results were returned back to the cells using the CUDA API.

The GPU implementation for the pairwise interactions of the nonbonded terms involves a double-layer space decomposition method, in which the given atom information of a cell is further spatially decomposed and a local search is performed. In the space decomposition method for the current GPU kernel implementation, we adopted a local search by the equally spaced grid, the uniform grid, similar to that described in the technical report in the CUDA SDK.[35]

The grid units obtained from the uniform grid partitioning on the GPU are called subcells. The uniform grid of psygene-G consists of the atom data array arranged in the order of the subcell index and the atom-range array, which records the range of atoms included in each subcell ('start-up' and 'end-of' atom index). Here, a subcell index indicates the integer values $(c_x, c_y, c_z) = (x_c/w_g, y_c/w_g, z_c/w_g)$, obtained from dividing the size of the given cell (x_c, y_c, z_c) along three axes by the grid width w_g. These arrays are created in the global memory by sequentially executing GPU kernels of the following four steps:

1) From the atom coordinates, calculate the hash values obtained from the indices of the subcells, to which each of the atoms belongs.

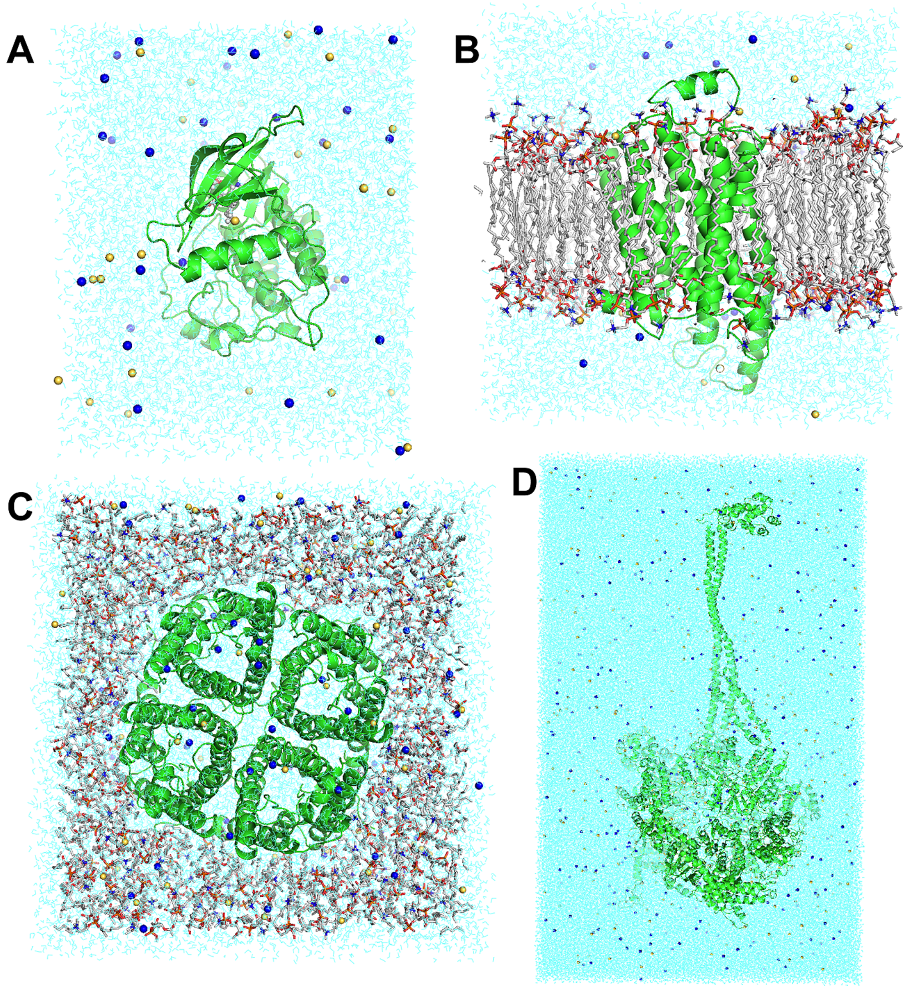


Figure 2. (A) The EGFR system consisting of 38,453 atoms, including a protein (PDB 1ml7[43]) with a green ribbon, 11,195 TIP3P waters (cyan), 30 Na^+ ions (orange), and 32 Cl^- ions (blue). The atom colors of erlotinib are the CPK colors. (B) The β_2AR system consisting of 56,121 atoms, including a protein (PDB 2rh1[47]), 160 POPCs, 9,951 TIP3P waters, 16 Na^+ ions, and 20 Cl^- ions. (C) The AQP4 system consisting of 104,415 atoms, including an AQP4 tetramer (PDB 2zz9[48]), 190 POPCs, 21,767 TIP3P waters, 61 Na^+ ions, and 61 Cl^- ions. (D) The Dynein system consisting of 1,004,847 atoms, including a Cytoplasmic Dynein motor domain (PDB 3vkg and 3vkh[49]), 4 ATPs, 317,371 TIP3P waters, 531 Na^+ ions, and 588 Cl^- ions.

2) Rearrange the pairs of the hash-value and the global-atom-index in ascending order, using the fast radix sort.[40]

3) According to the order of the rearranged global atom indices, rearrange the remaining atom information (coordinates, partial charges, atom types, and bond indices).

4) Depending on the rearranged hash-values, save the start-up and end-of atom indices, which are in each subcell.

The GPU kernels used in these steps are simple, except for the fast radix sort. The threads assigned to each atom load the data in the global memory to the register and perform the processing, and the results are saved to the global memory.

In the GPU kernel for the pairwise interaction calculations, a set of neighboring subcells was searched in the threads assigned to each atom, and the pairwise interaction calculations were executed. The local search was realized by searching, in the directions of the three axes, the range of the subcells that comprised a cube, in which the cutoff sphere was inscribed. For example, when w_g was set as the same length as the cutoff distance, a range of 27 subcells was searched, including the subcell to which the atom linked to the thread belongs. In order to obtain the local atom information, a two-step access procedure needs to be followed. First, the constituent atom range of the relevant subcell is gathered from the subcell index, which is obtained from the local search and the atom range array. Second, the atom information is procured from the atom-data array, according to the index of the constituent atoms. Thus, the information obtained about the local atoms is used for calculating the pairwise interactions with the atoms linked to each thread.

At the beginning of each step for the pairwise interactions, we first selected the nonbonded atom pairs, which should be truncated because of their longer distances than the cutoff distance, and excluded the covalently bonded atom pairs. Then, we calculated the nonbonded force terms of both the van der Waals and the electrostatic interactions between the remaining atom pairs. In the current GPU kernel, we did not load the atom information on the shared memory, because the atom-data array was not aligned in the SIMD vector width.

When the covalently bonded atoms were excluded, the selection was accomplished by using atom-list vectors of single bits, which expressed the 1−2, 1−3, and 1−4 covalent bond partners of each atom with a global atom index distance.[10] Each atom had an atom-list vector of 32 bits. The atom-ID difference between the atom in question and the bonded/nonbonded atom was N, where the N'th bit became 1/0. When the index-distance was not contained within 32, such as for disulfide bonds and for some covalent bonds in lipid molecules, we used the single bit corresponding to the index-distance from the starting-point atom as the 1−2 bonding partner. After eliminating the overlap, these bit vectors filled with these single bits are stored in the bit vector array in the constant memory. They can be indirectly accessed by the bit vector index and starting-point atom index, which are loaded from the bit vector location array and the starting-point atom array in the global memory.

For calculations of van der Waals terms, the forces were calculated with the Lennard-Jones type functions, which were directly computed in a faster manner than by making lookup tables with interpolation, as described by Ruymgaart et al.[41] Since the Lennard-Jones parameters depend on the atom types, they were saved in the texture memory, where random access is possible.

For calculations of the electrostatic force terms, the ZD method was applied. For the damping factor $\alpha > 0$, the potential function of the ZD method is expressed in a form that includes *erfc* and the exponential function, and these computations are quite time-consuming. However, when $\alpha = 0$, the expression is greatly simplified to general arithmetic operations, as in eqs 3 and 5. In addition, it has already been shown that the ZD method using $\alpha = 0$ has better accuracy than that with the nonzero α values for the appropriate cutoff distances.[31,32] Thus, the ZD method dedicated to $\alpha = 0$ was prepared and computed directly on the thread for high speed, while maintaining the accuracy. Several parameters and the self-energy term used in the ZD method were calculated at the program initialization, and they were placed in the constant memory. The calculation of the electrostatic modification terms required for the atom pairs relating to the covalent bond interactions[31] was executed on CPUs in parallel with the GPU computations, along with the calculations for the 1−2, 1−3, and 1−4 covalently bonded terms.

In general calculations for the pairwise interactions, the advantage of the force symmetry given by Newton's third law, where the reaction force is equal in magnitude and opposite in direction to the action force, is often utilized to reduce the computation cost by one-half. However, this approach was not implemented in the current psygene-G, and the forces of the atoms linked to the threads were simply calculated. The reason is that the application of Newton's third law increased the data communication, the memory usage, and the complexity of the GPU kernel in the current hardware, as mentioned by Ruymgaart et al.[41]

Hardware System. The evaluation of psygene-G was conducted using HA-PACS at Tsukuba University, which is a cluster computer composed of interconnected Appro Xtreme-X with four GPUs, 2 × Intel E5 (8 cores, 2.6 GHz), 128GB (DDR3, 1600 MHz), 4 × NVIDIA M2090 nodes, and 2 × Infiniband QDR (Mellanox ConnectX-3 dual head). Psygene-G was operated on a software system consisting of Red Hat Enterprise Linux Server release 6.1 (64bit), CUDA 4.1.28, Intel compiler 12.1, and Intel MPI4.0.3.

Targeted Biological Macromolecules. For the evaluation, we conducted measurements of the processing speed and calculation accuracy for the following four biological macromolecular systems: EGFR, β_2AR, AQP4, and Dynein.

As an evaluation system, EGFR that fulfills the neutrality condition, a system consisting of 1 EGFR, 1 erlotinib, 11,195 TIP3P waters,[42] 30 Na^+ ions, and 32 Cl^- ions (PDBID: 1m17,[43] 38,453 atoms) was prepared (Figure 2A). After performing 400 steps of minimizations by the steepest descent method and the conjugate gradient method, the NPT simulation[44] for 200 ps was performed to obtain an equilibrated system with the following conditions: periodic boundary, AMBER param99,[45] 300 K, 1 atm, cutoff = 12 Å, and 1.0 fs/step. The PME method (damping factor α = 0.35 Å^{-1}) was used for the long-range electrostatic force calculations. The preparations were performed by using myPresto/cosgene.[38]

The NVT simulation was then performed on the periodic system (68.2884 Å × 76.0471 Å × 63.8416 Å) obtained by the above NPT simulation, with the following conditions: periodic boundary, AMBER param99, 300 K, cutoff = 12 Å, 2.0 fs/step, and SHAKE[46] for hydrogen bonds only. The ZD method (α = 0.0 Å^{-1}) with various grid widths w_g (e.g., 4, 6, 8, 10, and 12 Å) was applied for the calculation of electrostatic force terms, instead of the PME method.

For the other three systems, similar preparation procedures were performed. For β_2AR, a G-protein coupled receptor (GPCR), a system was prepared consisting of β_2AR, 160 POPCs (1-palmitoyl-2-oleoyl-sn-glycero-3-phosphocholine), 9,951 TIP3P waters, 16 Na^+ ions, and 20 Cl^- ions (PDBID: 2rh1,[47] 56,121 atoms) (Figure 2B).[32] The equilibrated periodic system size (80.396 Å × 81.844 Å × 81.831 Å) was obtained after 1 atm NPT simulation for 200 ps at 300 K, and the successive NVT simulation was performed at 300 K with 2.0 fs/step and SHAKE.[46] This system is exactly the same as what was investigated in details comparing with the conventional Ewald and PME method.[32]

For AQP4, which is also a membrane protein, a system consisting of an AQP4 tetramer, 190 POPCs, 21,767 TIP3P waters, 61 Na^+ ions, and 61 Cl^- ions (PDBID: 2zz9,[48] 104,415 atoms) was prepared (Figure 2C). The equilibrated periodic system size (98.485 Å × 102.109 Å × 99.780 Å) was obtained after 1 atm NPT simulation for 200 ps at 300 K, and the successive NVT simulation was performed at 300 K with 2.0 fs/step and SHAKE.[46]

For Dynein, a system consisting of a Cytoplasmic Dynein motor domain, 4 ATPs, 317,371 TIP3P waters, 531 Na^+ ions, and 588 Cl^- ions (PDBID: 3vkg and 3vkh,[49] 1,004,847 atoms) was prepared (Figure 2D). The equilibrated periodic system size (196.791 Å × 318.680 Å × 159.983 Å) was obtained after 1 atm NPT simulation for 200 ps at 300 K, and the successive NVT simulation was performed at 300 K with 2.0 fs/step and SHAKE.[46]

As an evaluation system for energy conservation, an EGFR system was separately prepared with the following computational conditions: NVE, 300 K initial temperature, periodic boundary, AMBER param99, cutoff = 12 Å, and 1.0 fs/step. The ZD method (α = 0.0 Å^{-1}) with 5 types of grid widths, w_g = 4, 6, 8, 10, and 12 Å, was applied, and the energies were output at every 1,000 steps.

In addition to the above systems, we also measured the processing speeds for several typical systems, DHFR (dihydrofolate reductase: 23,558 atoms),[50] ApoA1 (apolipoprotein A1: 92,242 atoms),[51] and STMV (satellite tobacco mosaic virus: 1,066,624 atoms),[51] which have often been used as common benchmarks, for comparison with other studies. The details of the systems were described in their original articles.

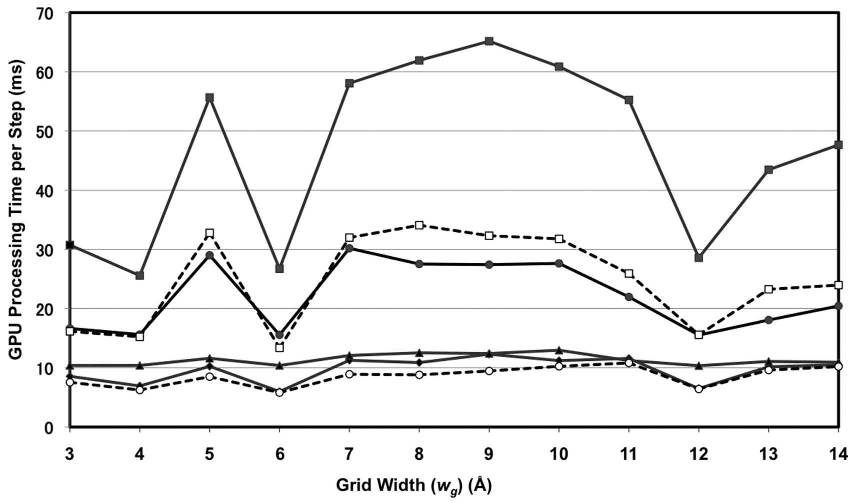


Figure 3. GPU performance as a function of grid widths with 12 Å cutoff. Each GPU processing time is indicated for the four systems, which are decomposed into 8−64 cells. Filled squares with solid lines are for the Dynein system with 27 cells, and filled squares with dashed lines are for that with 64 cells. Filled circles with solid lines are for the AQP4 system with 8 cells, and filled circles with dashed lines are for that with 27 cells. Open triangles with solid lines are for the β_2AR system with 8 cells, and open diamonds with solid lines are for the EGFR system with 8 cells.

Table 1. Psygene-G Performance for Four Protein Systems (ZD Method α = 0.0 with 12 Å Cutoff)

		processing time with GPU acceleration							
protein system[a] (no. of atoms)	cells[b]	atoms in a unit cell[c]	GPU (ms/step)	SHAKE and integration (ms/step)	others (ms/step)	total (ms/step)	total (ns/day)	processing time with only CPUs total (ms/step)	acceleration factor[d]
EGFR	1	38452.0	39.43	15.59	12.44	67.46	2.56	3631.44	53.83
(38452)	8	4806.5	5.33	2.14	2.40	9.87	17.50	456.24	46.22
β_2AR	1	56121.0	84.35	22.77	18.86	125.98	1.37	6151.17	48.83
(56120)	8	7015.0	10.05	2.18	4.11	16.34	10.57	773.60	47.34
AQP4	1	104414.0	127.83	74.71	32.79	235.33	0.73	14708.82	62.50
(104414)	8	13051.8	14.89	4.94	5.83	25.66	6.73	1827.38	71.22
	27	3867.2	6.00	1.94	8.70	16.63	10.39	368.77	22.17
Dynein	1	1004846.0	522.71	1073.72	379.80	1976.24	0.087	248148.26	125.57
(1004846)	27	37216.5	23.49	35.02	31.55	90.06	1.92	9581.55	106.39
	64	15700.7	13.44	16.11	20.30	49.85	3.47	2664.80	53.46

[a]In all cases, w_g was set to 6 Å. [b]The number of unit cells. The same number of GPU or CPU cores was used. [c]Average number of atoms in a unit cell. [d]The ratio of the total processing time per step with GPU acceleration versus that with the same number of CPU cores.

■ RESULTS AND DISCUSSION

Processing Speed. We performed 1,000-step (2 ps) NVT MD simulations to evaluate the processing speed, scalability, and optimal conditions (w_g size), using model systems of 8-cell EGFR, 8-cell β_2AR, 8- or 27-cell AQP4, and 27- or 64-cell Dynein. Here, the *n*-cells correspond to the number *n* of GPUs, and *n* is 1−64 for the almost cubic-shaped cells, except for the Dynein system.

First, we evaluated the w_g-size dependence of the processing time. We changed w_g from 3 Å to 14 Å (in increments of 1 Å) and measured the GPU processing times, as shown in Figure 3. For all systems, excluding the 8-cell β_2AR and 27-cell Dynein, the processing time is shortest at w_g = 6 Å. The processing times at w_g = 4 Å and w_g = 12 Å are almost as short as that at w_g = 6 Å. For 27-cell Dynein, w_g = 4 Å is the condition with the shortest processing time, and for 8-cell β_2AR, w_g = 12 Å is the condition with the shortest processing time. These optimal w_gs are equal to either the sizes of the cutoff or its divisors. There are two reasons for the w_g-size dependence of the processing time. The first is that the calculation amount is minimized, because the volume outside the cutoff region is minimized in a local search with w_g equal to the cutoff or its divisors. The other reason is that the processing efficiency is significantly increased, because around 8, 32, and 192 atoms, whose memory sizes correspond to the multiples of the cache line sizes (32 and 128 bytes) of the NVIDIA GPU, are stored in w_g = 4, 6, and 12 Å subcells, respectively.

Next, under the conditions with w_g = 6 Å, the processing time was measured for each system. Evaluations were also made for the 1-cell systems. As shown in Table 1 and Figure 4, the GPU processing time (ms/step) is almost scalable with respect to the number of GPUs, as the increase in the GPU processing time is proportional to the number of atoms in a cell. Likewise, the total processing time with GPU acceleration (ms/step) indicates the good scalability with the GPU parallelization in Table 1. Here, the total processing time includes the nonbonded forces calculation, the integration of Newtonian equations, the SHAKE constraint calculation, the MPI communication, and the output at every 1,000 steps (2 ps). The processing time for the nonbonded forces by the GPU acceleration was overlapped with the bonded force calculations by the CPU system, and the latter was always much less than the former, because there were far fewer covalent-bonded interactions than nonbonded interactions.

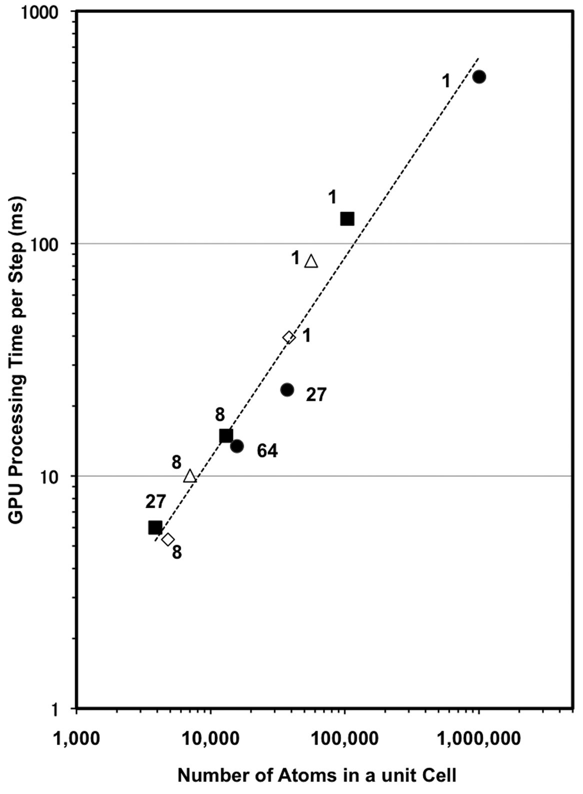


Figure 4. GPU performance as a function of atoms per GPU with the number of GPUs. Filled circles are for the Dynein system, filled squares are for the AQP4 system, open triangles are for the β_2AR system, and open diamonds are for the EGFR system. The number accompanying each symbol indicates the number of cells (GPUs). The approximate curve for all of the protein systems is shown by a dashed line.

As reported in ref 18, the ratio of the GPU processing time to the total processing time should generally decrease with an increase of the MPI communication with the number of processors. In the current MD simulation, several procedures, such as the integration of the Newtonian equation and the SHAKE constraint calculation, were performed on the normal CPU cores. Most of them were well parallelized with the numbers of CPU cores, which were comparable to those of the GPUs. Thus, as shown in Table 1, the time required for those procedures was reduced as the number of cells increased. However, when the number of cells increases, the time required for the MPI communication included in the "other" column should become large. Thus, in the cases of AQP4 and Dynein, the scalability in the total processing time was not very good, when comparing those with 8 cells and 27 cells in the former case, and those with 27 cells and 64 cells in the latter case.

In general, the performance of the MD program is improved by reducing computation and communication for massively parallel computers. The ZD method contributes not only toward omitting the long-range electrostatic force calculation but also reducing its MPI communication. Thus, the typical performances of the total processing speed in Table 1 correspond to 17.5 ns/day for EGFR with 8-cells, 10.6 ns/day for β_2AR with 8-cells, 10.4 ns/day for AQP4 with 27-cells, and 3.5 ns/day for Dynein with 64-cells. Namely, scalable expansion in terms of the number of GPUs is expected.

The performance of psygene-G was also evaluated for several typical systems, DHFR, ApoA1, and STMV, which have frequently been used as common targets for benchmarks. Table 2 summarizes the GPU processing time (ms/step) and the total processing time (ms/step and ns/day) calculated from our 1,000 step computations, with 2 fs/step and a 12 Å cutoff length for both the electrostatic and van der Waals forces.

The precise speeds of some of the other methods using the similar hardwares to the current study are summarized in Table S1 in the Supporting Information. The performance of our system was not as rapid as the best speed obtained by ref 18 for a small DHFR system, although it is not easy to directly compare the absolute speeds among several program systems, because of differences in the hardware systems. However, the processing time by GPU in our current system was very scalable, as shown in Figure 4. Since we only accelerated the nonbonded interaction terms by using GPUs, it may be possible to increase the speed by tuning other computations. For the large ApoA1 and STMV systems, our method works as fast or faster than the other systems, as shown in Table S1.

Calculation Accuracy. In the current NVIDIA's GPU (Fermi, K10), single precision floating-point arithmetic is more than twice as fast as double-precision floating-point arithmetic.[52] In the GPU implementation of psygene-G, single precision floating-point arithmetic was adopted for calculations of every energy and force, and the force accumulation was performed with double-precision floating point arithmetic.[14] Hence, it is necessary to confirm whether psygene-G's GPU implementation can maintain the accuracy required for an MD simulation. We adopted two methods for accuracy evaluation in this study: arithmetic validation of single precision floating-

Table 2. Psygene-G Performance for Three Benchmark Systems (ZD Method α = 0.0 with 12 Å Cutoff)

		processing time with GPU acceleration					
protein system[a] (no. of atoms)	cells[b]	atoms in a unit cell[c]	GPU (ms/step)	SHAKE and integration (ms/step)	others (ms/step)	total (ms/step)	total (ns/day)
JAC (DHFR)	1	23558.0	35.33	9.14	8.39	52.85	3.27
(23558)	8	2944.8	4.57	1.34	1.77	7.67	22.52
ApoA1	1	92224.0	89.36	52.44	28.65	170.45	1.01
(92224)	8	11528.0	10.98	4.04	4.59	19.61	8.81
STMV	1	1066624.0	699.68	1263.38	402.78	2365.83	0.073
(1066624)	8	133328.0	54.86	122.26	45.00	243.51	0.71
	27	39504.6	23.51	40.97	40.28	104.76	1.65
	64	16666.0	15.70	16.76	26.87	59.34	2.91

[a]In all cases, w_g was set to 6 Å. [b]The number of unit cells. The same number of GPU or CPU cores was used. [c]Average number of atoms in a unit cell.

Table 3. Accuracy of Psygene-G Calculation with GPUs (Single Precision) Compared with Psygene Calculation Only with CPUs (Double Precision) for Four Protein Systems (ZD Method $\alpha = 0.0$ with 12 Å Cutoff)

		relative difference of RMS force		relative difference of van der Waals energy		relative difference of electrostatic energy	
protein system[a] (no. of atoms)	no. of GPUs	av ($\times 10^{-6}$)	SD ($\times 10^{-8}$)	av ($\times 10^{-7}$)	SD ($\times 10^{-8}$)	av ($\times 10^{-8}$)	SD ($\times 10^{-8}$)
EGFR (38452)	8	2.75	2.34	1.73	6.19	2.86	2.12
β_2AR (56120)	8	2.27	3.68	14.9	75.6	3.23	2.38
AQP4 (104414)	8	3.29	3.95	3.81	14.6	2.86	2.04
	27	3.29	3.93	3.87	13.9	2.50	1.66
Dynein (1004846)	27	7.24	7.12	1.43	4.58	2.17	1.47
	64	7.24	7.11	1.43	4.38	2.08	1.34

[a]In all cases, w_g was set to 6 Å.

point operations in the force calculation within the GPU code[17,53] and MD validation of total energy conservation over the long-term NVE simulation (energy error estimation). Since the validation of the ZD method on the ion liquid, ion crystal, bulk liquid water, and biomembrane systems by comparison with the Ewald method has already been reported,[30−32] we here report only the arithmetic validation of the GPU implementation of the ZD method.

First, the arithmetic difference in the GPU implementation of the ZD method was measured with the relative RMS force difference, ΔF, defined in the following equation:[53]

$$\Delta F = \sqrt{\frac{\sum_i \sum_{\xi \in x,y,z} (F_{i,\xi} - F^{*}_{i,\xi})^2}{\sum_i \sum_{\xi \in x,y,z} (F^{*}_{i,\xi})^2}} \tag{6}$$

Here, i is the atom index, and ξ is the axis direction. $F_{i,\xi}$ and $F_{i,\xi}{}^*$ represent the force values calculated in the GPU and CPU implementations, respectively. The measurements were performed for four systems with various number of GPUs under conditions where w_g = 6 Å was maintained. After an energy minimization and a 200 ps equilibration of the NPT simulation, 1,000 step NVT simulations at 300 K of the systems were made, and the results are shown in Table 3 with a comparison between the CPU and GPU nonbonded force calculations on each time step, 2 fs. The average relative RMS force difference for each system is on the order of 10^{-6}, with the standard deviation not exceeding 10^{-7}. Therefore, we concluded that sufficient arithmetic accuracy of the GPU implementation was achieved. The mean relative RMS force difference increased in proportion to the total number of atoms in the system.

We also measured the relative differences between the GPU and CPU results for the van der Waals energy and the ZD electrostatic energy. The deviations of the average relative van der Waals energy are on the order of 10^{-7}−10^{-6}, with the standard deviation ranging from 10^{-8}−10^{-7}. This shows that sufficient arithmetic accuracy was obtained in the van der Waals energy calculation. The energy accuracies of the membrane systems (AQP4 and β_2AR) were less accurate than those of the other nonmembrane systems. Since the simulations of the membrane systems were the NVT simulations with the fixed cell size, the anisotropic distortion should increase the van der Waals atomic collisions of the membrane systems. The accuracy of the electrostatic energy by the ZD method is 1 order of magnitude higher than that of the van der Waals energy and does not depend on the system scale, because the GPU source codes (α = 0.0) of the ZD method are simpler than those of the van der Waals terms.

Next, the total energy fluctuation, ΔE_t, during the simulation time t, was evaluated using the following quantity (no unit):[53]

$$\Delta E_t = \frac{1}{N_t} \sum_{\tau=1}^{N_t} \left| \frac{E_0 - E_\tau}{E_0} \right| \tag{7}$$

Here, τ is the time step, N_t is the number of steps taken up to time t, E_τ is the total energy of the τth step, and E_0 represents the total energy of the initial step. The standard for acceptable accuracy is $\Delta E_t \leq 0.003$.[54] The measurement was conducted with 5-NVE simulations with grid widths w_g = 4, 6, 8, 10, and 12 Å for 8-cell EGFR, where the time step was set to 1 fs. After a 2 ns equilibration of the NVE simulation, we calculated ΔE_t over a 3 ns simulation ($E_0 = E_{2\text{ns}}$). All ΔE_t's in the 5 NVE simulations were 0.0001−0.0002, without exceeding the standard for accuracy.[54] Thus, we concluded that acceptable total energy stability was obtained.

CONCLUSIONS

Psygene-G is a simple space decomposition MPI/GPU-combined parallel program, which decomposes a coordinate space into uniform grid units (cells), assigns MPI processes to each obtained cell, and executes MD processing in parallel with GPU acceleration of the pairwise interaction calculations. We confirmed that introducing the ZD method realized sufficient calculation speed, acceptable calculation accuracy, and scalable expansion in terms of the number of cells (number of GPUs).

Our GPU implementation of the ZD method enabled high speed and quite accurate MD simulations of soluble proteins and biomembrane systems without any direct calculation of long-range electrostatic force terms, such as in Ewald or PME methods. The GPU implementation of the ZD method maintained acceptable arithmetic accuracies in the pairwise interaction calculation and the total energy stability. Even for large-scale systems such as Dynein, sufficient accuracies were maintained for both the force and energy calculations. In addition, the energy error was sufficiently small in the nanosecond-order simulation. We also showed that our GPU code was up to 100 times faster than a conventional implementation running only on CPU cores.

The current study provides a fast and reliable MD scheme utilizing the ZD method, which is physically accurate[30−32,55] and based on the cutoff scheme.[22] Thus, we believe that the current product leads to deeper understanding and further development of the cutoff-based methods, which do not require artificial exact periodic boundary conditions.

ASSOCIATED CONTENT

Supporting Information

GPU-accelerated performances of Psygene-G with several other programs on three benchmark proteins (DHFR, ApoA1, and

STMV) shown in Table S1. This material is available free of charge via the Internet at http://pubs.acs.org.

AUTHOR INFORMATION

Corresponding Author

*E-mail: harukin@protein.osaka-u.ac.jp.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

This work was supported by the New Energy and Industrial Technology Development Organization of Japan (NEDO) and by the Ministry of Economy, Trade, and Industry (METI) of Japan. It is also supported in part by the HA-PACS Project for advanced interdisciplinary computational sciences by exa-scale computing technology. We would thank Takashi Kurosawa (Hitachi Solutions East Japan, Ltd.) for his support in the development of the psygene coding. H.N. is grateful for JSPS KAKENHI Grant Numbers 23370071 and 24118008. N.K. and I.F. are grateful for Grant-in-Aids for Scientific Research (C) (25440070 and 25390156, respectively) from JSPS.

REFERENCES

(1) Karplus, M.; Kuriyan, J. *Proc. Natl. Acad. Sci. U. S. A.* **2005**, *102*, 6679−6685.
(2) Bastug, T.; Kuyucak, S. *Biophys. J.* **2012**, *4*, 271−282.
(3) Shaw, D. E.; Maragakis, P.; Lindorff-Larsen, K.; Piana, S.; Dror, R. O.; Eastwood, M. P.; Bank, J. A.; Jumper, J. M.; Salmon, J. K.; Shan, Y.; Wriggers, W. *Science* **2010**, *330*, 341−346.
(4) Durrant, J. D.; McCammon, J. A. *BMC Biol.* **2011**, *9*, 71.
(5) Wada, M.; Kanamori, E.; Nakamura, H.; Fukunishi, Y. *J. Chem. Inf. Model.* **2011**, *51*, 2398−2407.
(6) Narumi, T.; Ohno, Y.; Okimoto, N.; Suenaga, A.; Yanai, R.; Taiji, M. NIC Workshop 2006, From Computational Biophysics to Systems Biology, Meinke, J., Zimmermann, O., Mohanty, S., Hansmann, U. H. E., Eds.; John von Neumann Institute for Computing: J̌ulich, NIC Series, 2006; Vol. *34*, pp 29−36.
(7) Kikugawa, G.; Apostolov, R.; Kamiya, N.; Taiji, M.; Himeno, R.; Nakamura, H.; Yonezawa, Y. *J. Comput. Chem.* **2009**, *30*, 110−118.
(8) Toyoda, S.; Miyagawa, H.; Kitamura, K.; Amisaki, T.; Hashimoto, E.; Ikeda, H.; Kusumi, A.; Miyakawa, N. *J. Comput. Chem.* **1999**, *20*, 185−199.
(9) Elsen, E.; Houston, M.; Vishal, V.; Darve, E.; Hanrahan, P.; Pande, V. N-body simulation on GPUs. SC '06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing. New York, NY, 2006.
(10) Stone, J. E.; Phillips, J. C.; Freddolino, P. L.; Hardy, D. J.; Trabuco, L. G.; Schulten, K. *J. Comput. Chem.* **2007**, *28*, 2618−2640.
(11) CUDA Zone. URL: http://www.nvidia.com/object/cuda_home_new.html (accessed April 24, 2013).
(12) Owens, J. D.; Luebke, D.; Govindaraju, N.; Harris, M.; Kruger, J.; Lefohn, A. E.; Purcell, T. J. *Comput. Graph. Forum* **2007**, *26*, 80−113.
(13) Anderson, J. A.; Lorenz, C. D.; Travesset, A. *J. Comput. Phys.* **2008**, *227*, 5342−5359.
(14) Jha, P. K; Sknepnek, R.; Guerrero-García, G. I.; Olvera de la Cruz, M. *J. Chem. Theory Comput.* **2010**, *6*, 3058−3065.
(15) Nguyen, T. D.; Carrillo, J.-M. Y.; Dobrynin, A. V.; Brown, W. M. *J. Chem. Theory Comput.* **2013**, *9*, 73−83.
(16) Tanner, D. E.; Phillips, J. C.; Schulten, K. *J. Chem. Theory Comput.* **2012**, *8*, 2521−2530.
(17) Goetz, A. W.; Williamson, M. J.; Xu, D.; Poole, D.; Le Grand, S.; Walker, R. C. *J. Chem. Theory Comput.* **2012**, *8*, 1542−1555.
(18) Salomon-Ferrer, R.; Goetz, A. W.; Poole, D.; Le Grand, S.; Walker, R. C. *J. Chem. Theory Comput.* **2013**, *9*, 3878−3888.
(19) Goga, N.; Marrink, S.; Cioromela, R.; Moldoveanu, F. GPU-SD and DPD parallelization for Gromacs tools for molecular dynamics simulations. Proceedings of the 2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE), Larnaca, Cyprus, 2012.
(20) Phillips, J. C.; Stone, J. E.; Schulten, K. Adapting a Message-Driven Parallel Application to GPU-Accelerated Clusters. SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, Piscataway, NJ, 2008.
(21) Eastman, P.; Friedrichs, M. S.; Chodera, J. D.; Radmer, R. J.; Bruns, C. M.; Ku, J. P.; Beauchamp, K. A.; Lane, T. J.; Wang, L.-P.; Shukla, D.; Tye, T.; Houston, M.; Stich, T.; Klein, C.; Shirts, M. R.; Pande, V. S. *J. Chem. Theory Comput.* **2013**, *9*, 461−469.
(22) Fukuda, I.; Nakamura, H. *Biophys. Rev.* **2012**, *4*, 161−170.
(23) Ewald, P. P. *Ann. Phys. Leipzig* **1921**, *64*, 253−287.
(24) Darden, T.; York, D.; Pedersen, L. G. *J. Chem. Phys.* **1993**, *98*, 10089−10092.
(25) Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pederson, L. *J. Chem. Phys.* **1995**, *103*, 8577−8593.
(26) Wu, X.; Brooks, B. R. *J. Chem. Phys.* **2005**, *122*, 044107.
(27) Wu, X.; Brooks, B. R. *J. Chem. Phys.* **2008**, *129*, 154115.
(28) Brooks, B. R.; Brooks, C. L., 3rd; Mackerell, A. D., Jr.; Nilsson, L.; Petrella, R. J.; Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch, S.; Caflisch, A.; Caves, L.; Cui, Q.; Dinner, A. R.; Feig, M.; Fischer, S.; Gao, J.; Hodoscek, M.; Im, W.; Kuczera, K.; Lazaridis, T.; Ma, J.; Ovchinnikov, V.; Paci, E.; Pastor, R. W.; Post, C. B.; Pu, J. Z.; Schaefer, M.; Tidor, B.; Venable, R. M.; Woodcock, H. L.; Wu, X.; Yang, W.; York, D. M.; Karplus, M. *J. Comput. Chem.* **2009**, *30*, 1545−1614.
(29) Venable, R. M.; Chen, L. E.; Pastor, R. W. *J. Phys. Chem. B* **2009**, *113*, 5855−5862.
(30) Fukuda, I.; Yonezawa, Y.; Nakamura, H. *J. Chem. Phys.* **2011**, *134*, 164107.
(31) Fukuda, I.; Kamiya, N.; Yonezawa, Y.; Nakamura, H. *J. Chem. Phys.* **2012**, *137*, 054314.
(32) Kamiya, N.; Fukuda, I.; Nakamura, H. *Chem. Phys. Lett.* **2013**, *568−569*, 26−32.
(33) Wolf, D.; Keblinski, P.; Phillpot, S. R.; Eggebrecht, J. *J. Chem. Phys.* **1999**, *110*, 8254−8282.
(34) Fukuda, I.; Yonezawa, Y.; Nakamura, H. *J. Phys. Soc. Jpn.* **2008**, *77*, 114301.
(35) Green, S. CUDA Particles. URL: http://docs.nvidia.com/cuda/samples/5_Simulations/particles/doc/particles.pdf (accessed April 24, 2013).
(36) Yonezawa, Y.; Fukuda, I.; Kamiya, N.; Shimoyama, H.; Nakamura, H. *J. Chem. Theory Comput.* **2011**, *7*, 1484−1493.
(37) TOP 500 super computer sites. URL: http://www.top500.org/lists/2013/06/ (accessed Oct 25, 2013).
(38) Fukunishi, Y.; Mikami, Y.; Nakamura, H. *J. Phys. Chem. B* **2003**, *107*, 13201−13210. URL: http://presto.protein.osaka-u.ac.jp/myPresto4/index_e.html (accessed April 24, 2013).
(39) Heffelfinger, G. S. *Comput. Phys. Commun.* **2000**, *128*, 219−237.
(40) Satish, N.; Harris, M.; Garland. M. Designing effcient sorting algorithms for manycore GPUs. IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing, Rome, Italy, 2009.
(41) Ruymgaart, A. P.; Cardenas, A. E.; Elber, R. *J. Chem. Theory Comput.* **2011**, *7*, 3072−3082.
(42) Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. *J. Chem. Phys.* **1983**, *79*, 926−935.
(43) Stamos, J.; Sliwkowski, M. X.; Eigenbrot, C. *J. Biol. Chem.* **2002**, *277*, 46265−46272.
(44) Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; DiNola, A.; Haak, J. R. *J. Chem. Phys.* **1984**, *81*, 3684−3690.
(45) Case, D. A.; Darden, T. A.; Cheatham, T. E., III; Simmerling, C. L.; Wang, J.; Duke, R. E.; Luo, R.; Merz, K. M.; Wang, B.; Pearlman, D. A.; Crowley, M.; Brozell, S.; Tsui, V.; Gohlke, H.; Mongan, J.; Hornak, V.; Cui, G.; Beroza, P.; Schafmeister, C.; Caldwell, J. W.;

Ross, W. S.; Kollman, P. A. *AMBER 8*; University of California: San Francisco, CA, 2004.

(46) Ryckaert, J. P.; Ciccotti, G.; Berendsen, H. J. C. *J. Comput. Phys.* **1977**, *23*, 327−341.

(47) Cherezov, V.; Rosenbaum, D. M.; Hanson, M. A.; Rasmussen, S. G.; Thian, F. S.; Kobilka, T. S.; Choi, H. J.; Kuhn, P.; Weis, W. I.; Kobilka, B. K.; Stevens, R. C. *Science* **2007**, *318*, 1258−1265.

(48) Tani, K.; Mitsuma, T.; Hiroaki, Y.; Kamegawa, A.; Nishikawa, K.; Tanimura, Y.; Fujiyoshi, Y. *J. Mol. Biol.* **2009**, *389*, 694−706.

(49) Kon, T.; Oyama, T.; Shimo-Kon, R.; Imamula, K.; Shima, T.; Sutoh, K.; Kurisu, G. *Nature* **2012**, *484*, 345−350.

(50) JAC benchmark. http://ambermd.org/amber8.bench2.html (accessed November 11, 2013).

(51) ApoA1 and STMV benchmarks. http://www.ks.uiuc.edu/Research/namd/utilities/ (accessed November 11, 2013).

(52) NVIDIA Tesla. URL: http://www.nvidia.com/object/tesla-servers.html (accessed April 24, 2013).

(53) Shan, Y.; Klepeis, J. L.; Eastwood, M. P.; Dror, R. O.; Shaw, D. E. *J. Chem. Phys.* **2005**, *122*, 054101.

(54) Watanabe, M.; Karplus, M. *J. Chem. Phys.* **1993**, *99*, 8063−8074.

(55) Arakawa, T.; Kamiya, N.; Nakamura, H.; Fukuda, I. *PLoS One* **2013**, *8*, e76606.