





Scaling Molecular Dynamics Beyond 100,000 Processor Cores for Large-Scale Biophysical Simulations

Jaewoon Jung ^[a] Wataru Nishima,^[b,c] Marcus Daniels,^[b] Gavin Bascom,^[d] Chigusa Kobayashi ^[a] Adetokunbo Adedoyin,^[b] Michael Wall,^[a] Anna Lappala ^[b] Dominic Phillips,^[b] William Fischer,^[b] Chang-Shung Tung,^[b] Tamar Schlick,^[d] Yuji Sugita ^[a] and Karissa Y. Sanbonmatsu^{*[b,c]}

The growing interest in the complexity of biological interactions is continuously driving the need to increase system size in biophysical simulations, requiring not only powerful and advanced hardware but adaptable software that can accommodate a large number of atoms interacting through complex forcefields. To address this, we developed and implemented strategies in the GENESIS molecular dynamics package designed for large numbers of processors. Long-range electrostatic interactions were parallelized by minimizing the number of processes involved in communication. A novel algorithm was implemented for nonbonded interactions to increase single instruction multiple data (SIMD) performance, reducing memory usage for ultra large systems.

Memory usage for neighbor searches in real-space nonbonded interactions was reduced by approximately 80%, leading to significant speedup. Using experimental data describing physical 3D chromatin interactions, we constructed the first atomistic model of an entire gene locus (GATA4). Taken together, these developments enabled the first billion-atom simulation of an intact biomolecular complex, achieving scaling to 65,000 processes (130,000 processor cores) with 1 ns/day performance. Published 2019. This article is a U.S. Government work and is in the public domain in the USA.

DOI: 10.1002/jcc.25840

Introduction

Molecular dynamics (MD) is a powerful tool to understand biological phenomena in atomistic detail. From the first MD simulation of a protein,^[1] the spatiotemporal scale has increased significantly due to various hardware and software developments: Taiji et al. accelerated MD by developing specialized hardware, MDGRAPE-3 for time-consuming nonbonded interactions.^[2] ANTON, which is also specialized to MD simulations, achieved more than 100 times speed-up compared to conventional computers, simulating 1 millisecond protein dynamics in explicit solvent.^[3–5] The graphics processing unit (GPU) or Intel Xeon Phi have also become very promising in the context of extending the time scale of MD simulations.^[6–14] As a result of recent developments in hardware and software, large systems consisting of 64 and 100 million atoms can be simulated with MD.^[15,16]

Currently, the development of MD to extend the timescale and the system size is becoming more important due to novel experimental methodologies in biology. An example of such progress in this area is a better understanding of structural genomics and chromatin dynamics; chromatin is a biological DNA-protein complex that provides compaction of the genomic information in the nucleus of a cell. In this compact structure, DNA wraps around histone proteins—a functional unit known as the nucleosome—in a controlled and structured manner, due to interactions between DNA and proteins, and higher level of compaction is achieved through specific protein–protein and protein–DNA interactions. What exactly drives the compaction of chromatin and how it takes place is unclear. With recent advances in sequencing

technology, a deluge of 3D cross-linking data (chromosome conformation capture, or single-cell Hi-C) has revealed new insights into chromatin structure. In Hi-C experiments, the probability, P_{ij} , that two genomic loci i and j are in contact can be inferred. From these data, one may construct a contact map: a two-dimensional representation of a three-dimensional structure. Genome wide contact maps have revealed the existence of chromosome territories (also known as topologically associated domains or TADs) and have established the hierarchical nature of chromatin structure on the megabase scale.

[a] J. Jung, C. Kobayashi, M. Wall, Y. Sugita

Computational Biophysics Research Team, RIKEN Center for Computational Science, Kobe 650-0047, Japan

[b] W. Nishima, M. Daniels, A. Adedoyin, A. Lappala, D. Phillips, W. Fischer, C.-S. Tung, K. Y. Sanbonmatsu

Los Alamos National Laboratory, Los Alamos, New Mexico

E-mail: kys@lanl.gov

[c] W. Nishima, K. Y. Sanbonmatsu

New Mexico Consortium, Los Alamos, New Mexico

[d] G. Bascom, T. Schlick

New York University, New York, New York

Contract Grant sponsor: JSPS KAKENHI; Contract Grant number: 26119006; Contract Grant sponsor: JST CREST; Contract Grant sponsor: Los Alamos National Laboratory; Contract Grant number: CNLS; Contract Grant sponsor: Los Alamos National Laboratory LDRD; Contract Grant sponsor: RIKEN; Contract Grant sponsor: Los Alamos National Laboratory Institutional Computing; Contract Grant sponsor: U.S. Department of Energy; Contract Grant sponsor: LANL; Contract Grant sponsor: NIGMS; Contract Grant number: R35GM122562; Contract Grant sponsor: University of Tokyo

Published 2019. This article is a U.S. Government work and is in the public domain in the USA.

This complex hierarchical structure presumably results from the requirement of several meters of DNA to be compacted in a micron-sized nucleus in human cells. Most remarkably this compaction occurs in the crowded space of a nucleus without topological entanglement or knot formation. It is now widely believed that this is achieved by the chromatin adopting a fractal globule conformation. The hierarchical structure of chromatin is not just a by-product of compaction; it also plays a pivotal role in a range of genomic functions, most notably in gene transcription, DNA replication, and repair. Yet despite recent advancements in chromatin conformation capture and high-resolution direct imaging experiments (such as Fluorescence *in situ* hybridization or FISH), chromatin structure remains rather poorly understood. In part this is due to the dramatic changes in chromatin structure during the cell cycle, as well as the knowledge of cells, which do not follow the established levels of chromatin organization.

In MD, the general protocol consists of (1) evaluation of potential energy and force, (2) integration of coordinates and momenta, and (3) thermostat or barostat calculations. For biological MD simulations with explicit water molecules, the potential energy function consists of bonded (bond, angle, proper and improper dihedral angle) and nonbonded terms (electrostatic and van der Waals). The computational cost of the bonded interactions is $O(N)$ and takes part in a small fraction of the total simulation time. The main bottleneck in MD stems from the evaluation of the nonbonded interactions. The computational cost of the nonbonded interactions is $O(N^2)$. Because the van der Waals interactions decay rapidly according to the pairwise distance, they can be computed with $O(N)$ computational cost by applying a cutoff value beyond which the interaction energy is zero. The electrostatic energy is decomposed into the real- and reciprocal-space interactions

using the particle mesh Ewald (PME) scheme.^[17,18] The real-space interaction energy decays rapidly as the pairwise distance, so we can apply the same cutoff value as for the van der Waals interaction. Long-range interactions are described in a reciprocal space using fast Fourier transform (FFT), improving the computational cost to $O(N \log N)$. For small systems, the real-space nonbonded interaction becomes the main computational challenge. Conversely, the main bottleneck moves to evaluation of reciprocal space nonbonded interactions as we increase the number of computational processes or increase the target system size.

We have developed the GENESIS MD software to overcome current size limitations of MD.^[19,20] The new domain decomposition scheme in GENESIS produces highly efficient parallelization in real-space calculation, enabling simulations of very large systems.^[21] Efficient FFT parallelization schemes have been developed in GENESIS for high performance on the K computer.^[22] Due to DNA being a highly charged polymer, the accurate calculation of long-range electrostatic interactions is required, whereby the FFT becomes the main bottleneck of parallelization. Here, we discuss effective FFT parallelization schemes in GENESIS that can be used on Intel Xeon Phi processors for large-scale MD simulations. We also discuss the new implementation of nonbonded interactions in GENESIS to increase single instruction multiple data (SIMD) performance on Intel Xeon Phi. Our new developments are tested on the second-generation Intel Xeon Phi processors (code name: Knights Landing or KNL) on the Oakforest-PACS and Trinity Phase 2 platforms. We also constructed an atomic model of an entire gene locus (83 kilobases (kb) of DNA complexed with 427 nucleosomes, Fig. 1) consisting of more than 1 billion atoms and performed simulations on Trinity Phase 2 platform.

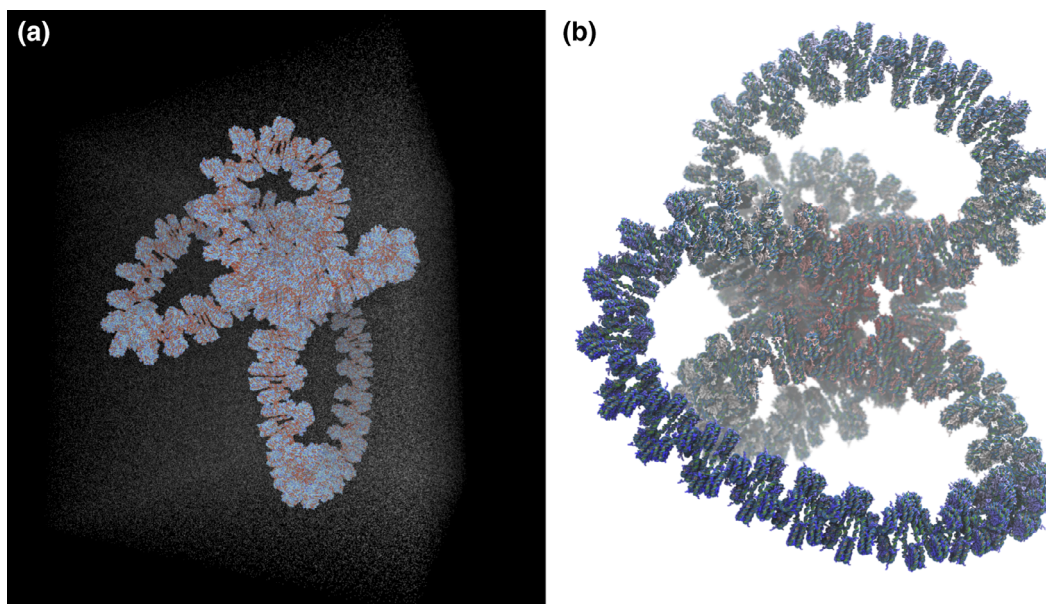


Figure 1. Explicit solvent simulations of GATA4 gene locus. a) Structure of fully solvated GATA4 gene in a periodic simulation cube, consisting of 83 kilobases of double-stranded helical DNA wrapped around 427 nucleosomes. b) A more detailed view of the gene structure. Protein tails used for programming gene expression protrude from each nucleosome. In a) and b), water molecules are not shown for image clarity. Ions are shown in a).

Methods

Atomistic model of GATA4 system

Modeling. We constructed a coarse-grained 3D scaffold using a coarse-grained mesoscale model of chromatin that led to the identification of a novel folding motif, hierarchical looping, where chromatin fibers undergo folding to produce layered networks of loops (citations to be added). By combining experimentally measured ultrastructural parameters with their previously verified mesoscale chromatin model, a detailed mesoscale model of the GATA4 gene locus was built (Fig. 1) and used to derive atomic resolution coordinates. The model was equilibrated resulting in a compact globular structure exhibiting hierarchical looping, where the positions of the loops agree with 3C data, and local folding agrees *in vivo* experiments. Our mesoscale chromatin model consists of three particle types: linker DNA, nucleosome core particle pseudo-charges, and flexible histone tails.^[23] Linker DNA is treated as a modified worm-like chain. The nucleosome core particles are represented as electrostatic objects, with shape- and surface-derived from the discrete surface charge optimization (DiSCO) algorithm that approximates the electric field of the atomic nucleosome by Debye-Hückel pseudo-charges along the surface of the complex as a function of monovalent salt.^[24] Flexible histone tails are coarse grained so that 50 beads represent the 8 histone tails, where each bead represents about 5 amino acid residues.^[23] The mesoscale model allows the direct placement of all-atom nucleosomal DNA and protein molecular structures in space, resulting in the atomistic model of the GATA4 system.

Model generation of macromolecular complexes & automated clash detection. Techniques such as X-ray crystallography, NMR spectroscopy, and electron microscopy are just some of the methods used to determine protein structure. However, such experimental data are typically insufficient to build a fully atomistic model—one must combine experimental data with homology modeling techniques (a reconstruction of a protein structure based on its amino acid sequence and knowledge of the structure of a related homologous protein). The method to generate a fully atomistic model (for a detailed description, see Carter & Tung^[25]) can be outlined as follows: (1) A helical curve with a specified pitch and diameter is generated: for nucleosomal DNA a pitch of 28 Å and a diameter of 106 Å are chosen based on crystal data. (2) The curve is digitized into 3.4 Å segments to represent the base-pair steps of the molecule. (3) The segments are mapped into a set of base-pair parameters to generate the fully atomistic model. The synthesis of homology techniques and experimental data is rarely flawless, and often steric clashes (which here we define as two atoms being separated by a distance less than 0.9 Å) are introduced between atoms. This is highly undesirable as many MD simulation packages are unable to correct for such clashes at the minimization stage and the high Van der Waals repulsion energy of the clashing atoms would lead to simulation failure. In light of this problem, we develop a procedure to identify steric clashes in a general DNA-protein structure and automatically adjust atomic

positions, without violating the known geometric constraints, such that all clashes are removed and corrected for. The DNA backbone is remarkably rigid. Besides the puckering behavior exhibited by the deoxyribose ring, the only other significant freedom of rotation lies in the phosphate group, about the O3'-C5' axis. In principle this rotational freedom, denoted ϕ , can take on any value ϕ in $[-180^\circ, 180^\circ]$, and this can be exploited for the purposes of clash correction (for geometric details see ref. [25]). Protein structures, unlike DNA, are generally more flexible, yielding more ways in which atomic positions can be modified to remove steric clashes. However, in practice we found that modifying just one torsion angle, denoted χ —where χ in $[-180^\circ, 180^\circ]$, of the bond connecting the amino acid side-group to the residue was sufficient to remove all steric clashes in our models when combined with ϕ rotations. The clash correction program was predominantly written in Python, with calls to Fortran subroutines, which calculate new atom positions upon changing ϕ or χ . Figure 2 shows a flow diagram of the program's operation.

Major characteristic of GENESIS for large-scale MD

Like existing MD programs for large-scale biological simulations (such as GROMACS^[10] and NAMD^[26]), GENESIS supports hybrid parallelization (combination of shared and distributed memory parallelization) for efficient usage of CPU cores. Based on this, for large-scale MD, GENESIS has the following major innovative characteristics:

Inverse lookup table^[27]. GENESIS makes use of a lookup table for energy and force evaluation of short-range nonbonded interactions instead of using direct calculations. The lookup table is based on the inverse distance squared calculation, which allows to perform accurate calculations for very short pairwise distances by assigning many table points, whereas fast evaluation of energy/force is available by reducing table points for longer pairwise distances.

Domain decomposition with midpoint cell method^[21]. In the midpoint cell method, each rank has the data of cells in the corresponding subdomain and adjacent cells of the subdomains. It sends/receives data to/from the neighboring subdomains. The nonbonded interaction is parallelized by distributing the cell pair lists (or midpoint cell indices) over OpenMP threads. Integration and constraint evaluations are parallelized by distributing the cell indices in each subdomain over OpenMP threads. By adopting the midpoint cell method with the volumetric decomposition of FFT described below, we can assign the same domain decomposition between real- and reciprocal-spaces, skipping communications of charge grid data before FFT.

Volumetric decomposition of FFT^[22]. GENESIS makes use of the volumetric decomposition FFT where the reciprocal space is decomposed in all three dimensions. This requires more frequent MPI_Alltoall communications than slab or pencil decompositions. However, this scheme is more favorable for large systems on massively parallel supercomputers because the amount of data in each communication is much smaller

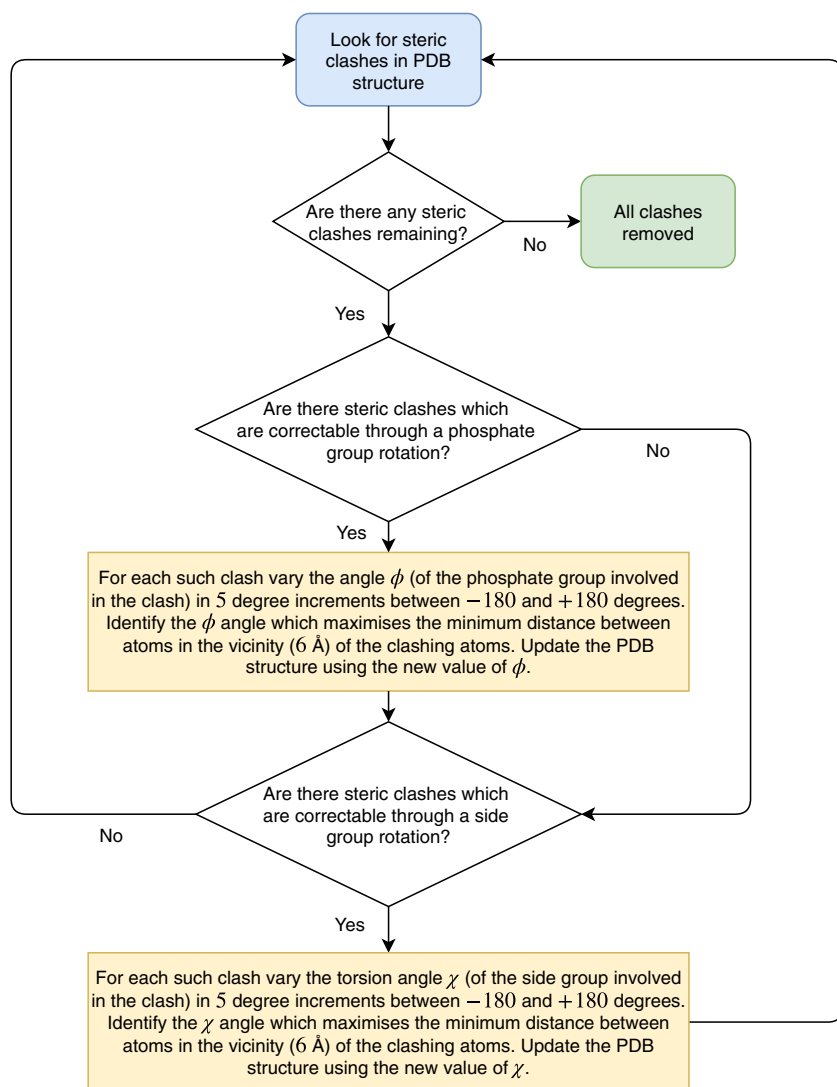


Figure 2. Automated clash detection: flow diagram of the operation of the clash correction script. Clashes correctable by a phosphate group rotation are those involving P, O5', C5', C4', C3', O3' type atoms on the DNA backbone. All other clashes involve at least one atom in a protein side group and can be corrected by a side group rotation. The vicinity of a clash is defined to be all atoms within a 6 Å radius of the atoms involved in the clash. [Color figure can be viewed at wileyonlinelibrary.com]

than the other decompositions. Moreover, we can skip communication of charge data before FFT by assigning the same domain decompositions between real and reciprocal spaces. Specifically, let us assume that we decompose the overall space by P MPI processes. If P is factorized by $P = P_x \times P_y \times P_z$, we decompose the space by P_x in x dimension, P_y in y dimension, and P_z in z dimension. To perform the FFT in each dimension, we assign MPI_Alltoall communication such that each process has global data in the specified dimension. According to our volumetric decomposition FFT scheme (1d_Alltoall), forward FFT consists of eight computation/communication procedures outlined in Appendix A.

In this scheme, we require 5 one-dimensional MPI_Alltoall communications. Intuitively, the procedure of the backward FFT (BFFT) is opposite to the forward FFT; this decomposition scheme has several advantages over other schemes from the scalability point of view despite of more frequent communications. On the one hand, the number of processes involved in communications is proportional to the cubic root of P . On the other hand, it is proportional to P or the square root of P in the case of slab or pencil decomposition FFT. When P is sufficiently

large, the communication cost could be reduced by more frequent communication with small number of processes involved in the communication. We also utilize a volumetric decomposition scheme (see Appendix A) with reduced frequency of communications, while the number of processes involved in the communication is larger (2d_Alltoall).

In this scheme, we require 3-dimensional MPI_Alltoall communications. The 2d_Alltoall scheme is similar to pencil decomposition FFT, if we neglect the first MPI_Alltoall communication. According to our investigations of 1d_Alltoall and 2d_Alltoall schemes, the performances are almost equivalent to each other on the K computer.

Parallel I/O for a large system^[19]. For an MD simulation of a very large system, single restart/trajectory files cause several problems. First, there is a memory problem in reading single restart/trajectory files. In the case of a 1 billion atom system, we require a 24 GB (6 [three coordinates and velocities] \times 1 billion [number of particles] \times 4 [byte for a real number]) file size for a single restart file. Consequently, each process needs at least 24 GB memory to read only the restart file. Because we require not only

restart but also PDB (protein data bank) and force-field parameter files, the file size becomes even larger than 100 GB, which can be prohibitively large. Writing a single restart/trajectory is even more problematic for a very large system. To write a single trajectory file, we need to accumulate 12 GB size coordinates by communication among processes. Parallel I/O is designed to avoid these problems. With parallel I/O, each process requires data of the corresponding subdomain. As we increase the number of processes, the required amount of data in each process decreases accordingly. A parallel restart file is generated from the initial PDB and parameter files using a parallel restart setup tool (`prst_setup`). During MD simulations, each MPI process writes trajectory files that contain only the coordinates of the corresponding subdomain. The number of files is identical to the number of MPI processes. After finishing the MD simulation, users can combine the trajectory files into a single file with atom selection option in order to only analyze the atoms/atom groups of interest.

Optimization of GENESIS for KNL Intel Xeon phi processors

Hardware architecture of KNL in trinity phase 2 and Oakforest-PACS. As the feature width of commodity chips approaches 10 nm, power consumption has increased substantially due to leakage currents. Many-core chips with fewer components per core and lower clock speeds therefore are being developed for future exascale computing. The 1.4 GHz Intel Knights Landing (KNL) 7250 chip in Trinity Phase 2 and Oakforest-PACS has 34 active tiles, each of which has two cores that share a single 1 MB L2 cache. Each core supports four-fold simultaneous multi-threading (SMT); thus, up to 272 hardware threads are available on a single node. They support AVX512 instructions including conflict detection, hardware gather/scatter prefetch, and exponential and reciprocal instructions. Each processor comes with 16 GB on-package High Bandwidth Memory (MCDRAM) and has access to an additional 96 GB memory via DDR4 channels. The MCDRAM and DDR support 400 GB/s and 90 GB/s streaming,

respectively; however, the latencies of MCDRAM and DDR access are comparable. The memory mode of the processor is configurable at boot time; simulations were performed using nodes booted in the quad cache mode, in which the MCDRAM is configured as L3 cache, and the DDR is configured as usual. In Oakforest-PACS and Trinity Phase 2, Intel Omni-Path and Cray Aries interconnect are assigned, respectively.

Data layout. In GENESIS 1.0-1.3 packages, coordinates, velocities, and forces have an array-of-structures (AoS) type, in terms of source code organization. In other words, the x , y , and z components of coordinates, velocities, and forces are contiguous in memory. On KNL, the SIMD performance improves when making use of a structure-of-arrays (SoA) data type where x components of coordinates are contiguous in memory. To increase the SIMD performance, we changed the data layout from AoS to SoA in the module of real-space nonbonded interactions.

New algorithm for real-space nonbonded interactions: Reduced memory neighbor search scheme with improved performance.

For the efficient usage of memory on KNL, we generate a new neighbor search scheme, which reduces the memory size significantly. In GENESIS 1.0-1.3 packages, we make use of the neighbor search algorithm based on the midpoint cell scheme. In this scheme, we first check cell pair indices $icel$ and $jcel$ for a given cell pair. For each atom indices i in $icel$, we search for atom indices j in $jcel$ and write the indices of j as neighbor lists (Algorithms 1,2). The required memory size for the process becomes MN^2 for M cell pairs in a process and N atoms in a cell. The neighbor search is written every 10 or 20 steps, and the real-space nonbonded interactions are performed based on this neighbor list. For a very large system, this might make use of both MCDRAM and DDR, losing overall performance. Moreover, the algorithm does not allow contiguous memory access in force evaluation, preventing high SIMD performance (Fig. 3a).

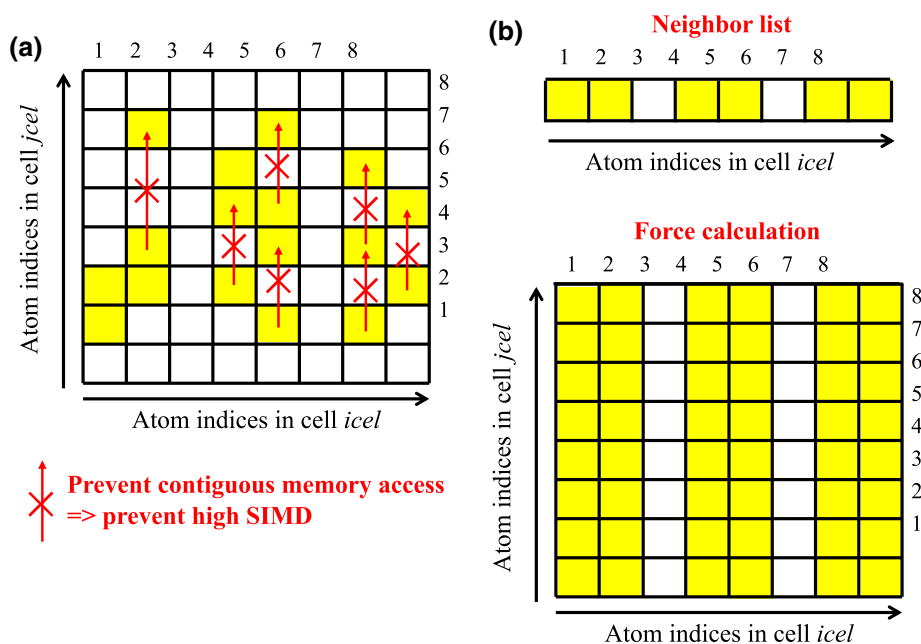


Figure 3. Nonbonded interaction schemes. (a) Nonbonded interaction scheme used in GENESIS 1.0-1.3. (b) New nonbonded interaction scheme for Intel Xeon Phi processors. [Color figure can be viewed at wileyonlinelibrary.com]

To use only MCDRAM in MD simulations with better performance, we devise a new algorithm of a neighbor search with reduced memory usage. In the proposed algorithm, for each atom i in $icel$, we check if there is at least one atom index j in $jcel$ to be considered as a neighbor. If there is at least one neighbor of i , we write $neighbor(i,j) = 1$ in the neighbor search module. In the nonbonded interaction module, we assign interactions between i in $icel$ with all j indices in $jcel$ if $neighbor(i,j) = 1$. For example, let us consider atom index 5 in $icel$, as shown in Figure 3. In this case, four atoms (indices 2, 4, 5, and 7) are considered as the neighbor list. In the new interaction scheme, we write $neighbor(5,j) = 1$ and calculate all interactions between 5-th atom in $icel$ and all atoms in $jcel$ in the nonbonded interaction module. In this neighbor search scheme, the required memory size becomes MN , so we can reduce the memory usage. This neighbor search scheme and the nonbonded interaction algorithm with SoA data are written in Algorithm 3. With our new neighbor search scheme, contiguous memory access is available, optimizing the overall performance (Fig. 3b). Therefore, the new neighbor search scheme not only decreases the overall memory usage, but also results in better performance. In addition, this reduces the computational time of neighbor list writing because we are not required to check all particle pairs.

Optimization of FFT with proper choice of the FFT decomposition scheme.

In our previous development of the parallelization scheme, the FFT is performed on the K computer with 6D mesh/torus interconnect network. On the K computer, 1d_Alltoall and 2d_Alltoall schemes works almost equivalently by properly assigning the decomposition topology for the 1024^3 PME grid. However, we could not guarantee the same parallel efficiency on the KNL machine. Therefore, one needs to investigate the parallel performance of the two developed schemes and make a choice of suitable FFT scheme for the target hardware and the target system.

Results and Discussion

Performance results of GENESIS on KNL

Although there are 68 cores per node, we assign 64 cores per node for easier selection of MPI processes. In each core, four threads are available. We considered 256 threads per node for our benchmark test. In our hybrid (MPI + OpenMP) parallelization, we assigned eight OpenMP threads, so the number of MPI ranks in a node is fixed to 32. Benchmark systems are created by placing multiple STMV (Satellite tobacco mosaic virus) systems in a single box, and also by preparing the GATA4 gene locus system. We used the CHARMM force field with modified TIP3P explicit water model.^[28,29] We used 12.0 Å cutoff thresholds, whereas van der Waals interactions were smoothed to zero from 10.0 Å to 12.0 Å using switch functions. We used the time step of 2 fs and neighbor list search and particle migrations were considered every 10 steps. Bonds related to hydrogen atoms were constrained by SHAKE/RATTLE algorithms.^[30,31] The SETTLE constraint algorithm was used for the water.^[32] On Oakforest-PACS, we investigated the parallel efficiency of our FFT scheme, and the effect of the new algorithms

of the real-space nonbonded interactions from a 92,224 atom system (apolipoprotein A-I (ApoA1), a component of high-density lipoprotein). We also performed benchmarks using 1024^3 and 2048^3 PME grids for a 230 million and 1 billion atom model system. On Trinity Phase 2, we used a 2048^3 grid for the GATA4 gene locus (1 billion atom system), but a smaller time step of 1 fs. In all cases, we used the Intel Fortran Compiler 17.0.

FFT communication time on Oakforest-PACS. To study the performance of GENESIS on KNL, we investigated the communication costs using the volumetric decomposition FFT schemes. Our target PME grid numbers here are 1024^3 and 2048^3 . In MD simulations, these grid numbers are usually used for a system consisting of 100 million to one billion atoms. FFT timing statistics are obtained by running MD simulations: the communication time includes time of waiting before communication and all MPI_Alltoall communications. Because the PME grid number in the x dimension is not divisible by the number of processes (if the input grid number in the x dimension is n , the grid number in the x dimension becomes $\frac{n}{2} + 1$ after the FFT by considering real-to-complex FFTs), each process has a different communication time. In Figure 4, we depict the timing statistics of total FFT execution and communications in FFT by choosing the time of the most time-consuming process. As mentioned in the previous section, we assigned eight OpenMP threads, so the total number of threads used is obtained by multiplying eight by the number of MPI processes. In the first step, the wait time could be large due to different starting setup times, so we

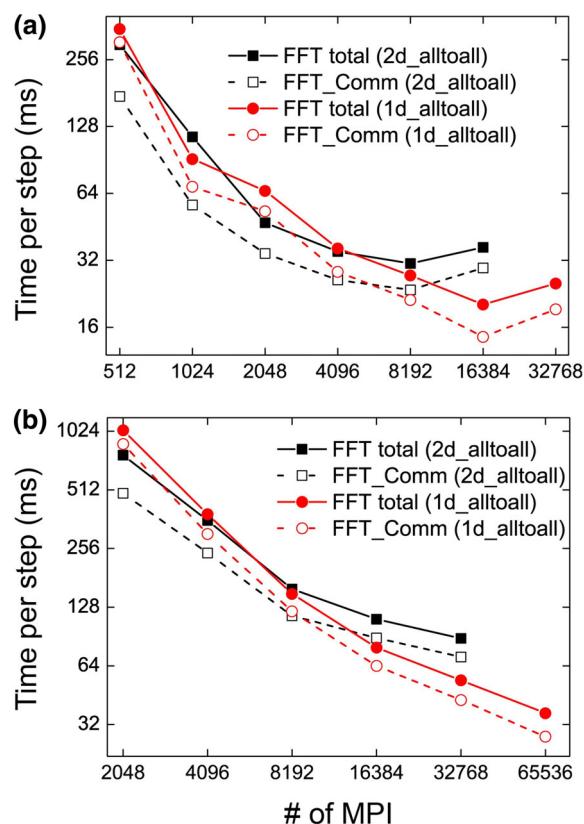


Figure 4. FFT time of a) 1024^3 and b) 2048^3 PME grids using GENESIS on Oakforest-PACS. [Color figure can be viewed at wileyonlinelibrary.com]

measured the communicational time from 1001 to 2000 MD steps. This includes timing statistics of 1000 forward and backward FFTs. For both 1024^3 and 2048^3 PME grids, 2d_Alloall scheme is better than 1d_Alloall, if the number of processes is not sufficiently large. On the other hand, 1d_Alloall becomes better than 2d_Alloall for larger number of processes. Therefore, more frequent communication with small number of processes involved in the communication is preferred, if we want to make use of large number of KNL processors. For the MD simulation of 1 billion atom size system with 2048^3 PME grid sizes using more than 32,768 MPI processes, 1d_Alloall scheme is a better choice. This is different from the performance results of the K computer on which performances of 1d_Alloall and 2d_Alloall are almost equivalent to each other even using very large number of MPI processes. The main difference between two tests on KNL and K is the number of MPI processor in a node: we used one MPI per node in the case of K whereas on KNL, 32 MPIs were assigned per node. We compared the FFT performance between our 1d_Alloall and MKL FFT library used in GROMACS MD package version 2016.1,^[10] which makes use of the pencil decomposition FFT. Using MKL FFT, we observe that the lowest communication time of FFT for 2048^3 grids is 34.22 ms, which is slightly higher than 1d_Alloall scheme (27.70 ms). From this comparison, we expect that 1d_Alloall could be the most suitable FFT parallelization scheme for a very large system on a large number of KNL processors.

Effect of new nonbonded interaction scheme with reduced memory.

To understand the effect of the new neighbor search scheme with SoA data, we performed four MD simulations of ApoA1 (92,224 atoms) system on Oakforest-PACS: Algorithms 1,2 with and without AoS data and Algorithms 3,4 with and without SoA data. For one node, we assigned 32 MPIs with 8 OpenMP threads. The CPU time for the real-space nonbonded interactions and neighbor search generations are obtained from the profile of 0-th thread of the MPI rank 0 by using VTune Amplifier. In Table 1, we show the memory usage for the neighbor search and the total number of pairwise interactions in real-space nonbonded interactions. In Table 2, we describe the timing statistics of neighbor search generation and real-space nonbonded interactions for 2000 MD steps. Here, we assigned the neighbor search list generation every ten steps. Our new algorithm (Algorithm 3 with SoA) has 1.5× better performance than that used in GENESIS 1.0-1.3 (Algorithm 1 with AoS) although the number of interactions is 2.5 times larger and just one sixth of the memory is used for neighbor search. The higher speed of Algorithm 3 in the neighbor list is due to less computational

Table 1. Memory usage (GiB) for the neighbor search and number of pairwise interactions evaluated in real-space nonbonded interactions (32 MPI with 8 OpenMP case).

	Memory usage	Number of interactions
Algorithm 1 (AoS)	305.88 MiB	47,364,470
Algorithm 1 (SoA)	305.88 MiB	47,364,470
Algorithm 3 (AoS)	48.40 MiB	122,404,436
Algorithm 3 (SoA)	48.40 MiB	122,404,436

Table 2. Calculation time (sec) of neighbor search + real-space nonbonded interaction for 2000 MD steps.

	Neighbor list	Nonbond	Total
Algorithm 1 (AoS)	9.95	36.42	46.37
Algorithm 1 (SoA)	6.60	33.71	40.31
Algorithm 3 (AoS)	4.91	102.46	107.37
Algorithm 3 (SoA)	4.15	27.86	32.01

cost. With Algorithm 1, we need to check all particle pairs. On the other hand, we can skip such a checking procedure frequently. The better performance of the real-space nonbonded interactions is mainly due to less L2 HW prefetcher allocations. With Algorithm 3 with SoA, the L2 HW prefetcher allocations are around 4.2×10^7 . With Algorithm 1, this becomes 1.5 times larger. The large HW prefetch problem is highly related to the indirect memory access. With Algorithm 1, the index of the most inner loop is k , but we evaluate all interactions using index j by converting from the index k to j using neighbor_list. This indirect memory access becomes the main problem for performance improvement and can be reduced with Algorithm 3. We found that Algorithm 1 can be accelerated by adopting SoA data, but still shows less performance than Algorithm 3 because of the indirect memory access. Algorithm 3 with AoS data is four times slower than the same algorithm with SoA, showing the importance of SoA data for performance. We also tested these algorithms on other CPU architecture: Intel's Sandy bridge, Ivy bridge, Haswell, Broadwell, and Skylake micro-architectures, and SPARC64 VIIIfx on the K computer. For Haswell, Broadwell, and Skylake, Algorithm 3 works better than Algorithm 1. On the other hand, Algorithm 1 is better than Algorithm 3 for other processors. Even on Haswell, Broadwell, and Skylake, Algorithm 1 works better if we do not use INTEL compilers.

Benchmark of 230M and 1B system on Oakforest-PACS.

Benchmarks of 230 million and 1 billion atoms (230M and 1B system) were performed by creating a system of $6 \times 6 \times 6$ and $10 \times 10 \times 10$ copies of STMV, respectively. As mentioned in results and discussion Section 1 A-B, we applied 1024^3 and 2048^3 PME grids, corresponding to 1.2 Å and 1.1 Å PME grid spacings. We used velocity Verlet integrator with NVE ensemble, where the reciprocal-space calculation with FFT is performed every step. As shown in Figure 5, GENESIS shows good scalability even though we calculate reciprocal-space calculation every time step. The most time-consuming process is the evaluation of the real-space interaction for small numbers of processes. As we increase the number of MPI processes, the main bottleneck moves from the real-space to reciprocal space interactions because there is no communication in the real-space interaction, while reciprocal-space interaction includes MPI_Alloall communications in FFT. For example, in the case of 230M system using 2048 MPI ranks, the CPU time of real-space interaction is twice as large as the CPU time of reciprocal-space interaction. If we increase the number of MPI processes, the ratio of the CPU time becomes opposite: the CPU time for reciprocal-space interaction becomes twice larger than the real-space interaction. Due to our efficient FFT parallelization, we could reduce the CPU time of both real- and

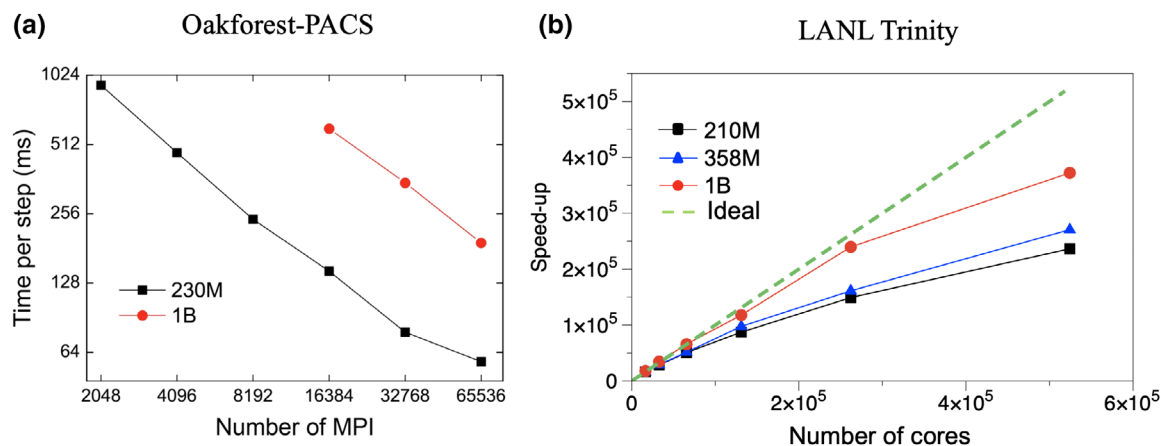


Figure 5. a) Benchmark performance of 230M and 1B system on Oakforest-PACS. b) Strong scaling of GENESIS on LANL Trinity Phase 2 KNL platform: speed-up as a function of number of cores.

reciprocal-space nonbonded interactions and obtain the scaling as shown in Figure 5. For the 1 billion atom system, the performance reported by us is 1 ns/day using 65,000 MPIs, which corresponds to 130,000 processor cores or 2048 nodes. We compare the performance of GENESIS with other MD software (NAMD^[26] and GROMACS^[10]) for large-scale MD simulations. We found that GROMACS shows very good performance on real-space nonbonded interactions; however, the code was not applicable for large systems consisting of more than 100 million atoms. To compare the performance of GENESIS with previously published NAMD benchmark performance results, we estimate approximately 1.5 ns/day when considering the system size and platform architecture. Direct comparison is also complicated by different simulation conditions between NAMD's and GENESIS' benchmarks. For example, NAMD assigned FFT grid spacing 2 Å while it is 1.1 Å in our case. Therefore, we used almost $(\frac{2}{1.1})^3 \approx 6$ times larger FFT grid sizes than NAMD's benchmark case. In addition, NAMD performed reciprocal-space nonbonded calculation with FFT every three steps whereas we performed it every time step.

MD simulation of the GATA4 system on Trinity phase 2. On Trinity phase 2, we performed approximately 1 ns MD simulations of the gene locus with 1 fs time step. Parallel restart files are generated by "prst_setup" tool in GENESIS to reduce the file size required in each MPI rank. Here, we assigned 65,536 MPIs, so the number of files is also 65,536. The system size is $1907 \times 2252 \times 2485 \text{ \AA}^3$, and it is subdivided by $32 \times 32 \times 64$ subdomains according to the number of MPIs. Table 3 shows the timing statistics of each evaluation used in MD. Here, time for integration includes the time of integration of coordinates

Table 3. Calculation time (s) of each component for 50 ps MD simulation of gene locus (time step is 1 fs).

	Total time (s)	Ratio (%)
Total	9936.63	100.00
Integration	520.03	13.68
Bonded	13.68	0.14
Nonbonded (real)	2272.25	22.87
Nonbonded (reciprocal)	5493.88	55.29

and velocities and that of communication of coordinates and forces. Bonded interactions consist of energy/force evaluation of bonds, angles, and dihedral angles. As shown in the table, the evaluation of nonbonded interactions takes around 80% of the overall simulation time. For nonbonded interactions, the evaluation of the reciprocal-space interaction takes more than twice of the real-space interaction, and it covers more than 50% of the simulation time. Communication times of forces and coordinates among subdomains are almost negligible. This observation suggests that the main problem of MD simulations of large systems on a large number of processors is the evaluation of the FFT included in the reciprocal-space interaction. We believe that our parallelization scheme of FFT is most suitable for large-scale MD simulations on a large number of processors, considering the scalability of our FFT.

Implications for chromatin biophysics: Electrostatics of the crowded environment inside compacted chromatin

We performed the first billion atom biomolecular MD simulation of a gene locus in explicit solvent. The achievement is notable considering the necessity of including long-range forces, as well as the large memory footprint typically associated with biomolecular complexes, in contrast to large MD simulations in materials science, which typically neglect long-range forces. Importantly, the study is not only the first simulation of an entire gene locus, but also the first simulation of a gene to examine DNA sequence level. One of the motivations to simulate large-scale systems on the atomic level is electrostatics—it plays an important role in many biological systems, including DNA, RNA, ion channels, and chromatin complexes. Typically, electrostatic potentials are computed using the Poisson-Boltzmann method.^[33] While this method is excellent at characterizing the general features of the overall complex, many important aspects of electrostatic effects occur on short length scales, within the Debye length. In the future, once the technical obstacles of running biologically relevant (microsecond in the case of chromatin dynamics) time-scales are overcome, large-scale massively parallel explicit solvent simulations of this kind will allow us to more accurately simulate nucleosome-nucleosome interactions and nucleosome interactions with linker DNA, where charges

are not fully screened by the ionic environment. In these regimes, the finite size and discrete nature of charged side chains, nucleic acid phosphates, and ions themselves make important contributions, which are not described by the Poisson-Boltzmann approximation.^[34] Nevertheless, our benchmark simulations provide, for the first time, an example of a nanosecond time-scale dynamics of an entire gene consistent with a realistic force field potential and based on experimental data—an important first step in the direction of simulating a complete biological system over long times on an atomistic level. Rapid improvements in both hardware and software needed to achieve this goal are inevitable in the foreseeable future.

Conclusions

Fully atomistic simulations based on coarse-grained models play a vital role for the understanding of chromatin structure. It is the properties of the structure on the atomic scale, such as how the

DNA wraps around histone proteins to form nucleosomes and how the multiple nucleosomes wrap into a 30 nm fiber that ultimately dictate the megabase-scale hierarchy of chromatin. It is also known that electrostatic interactions between dissolved ions and the fibers strongly affect folding dynamics, a process which can only be modeled on an atomistic level. Such studies enable the examination of histone tail dynamics, histone-histone and nucleosome-nucleosome interactions in crowded environments, all of which are important unsolved issues in chromatin dynamics and chromatin compaction, key processes in cell division, development, cancer and neuropsychiatric disorders. Improved calculations of the electrostatic potential will allow us to investigate the mechanism of interaction between histone tails of nearby nucleosomes, be it ion-mediated, water-mediated, direct charge-charge or some other as yet unknown mechanism.

Biomolecular simulations are particularly challenging computationally since they require the inclusion of long-range forces,

Algorithm 1. Neighbor search used for GENESIS 1.0-1.3.

```
do ij = 1, M
  icel = cell_index(1, ij)
  jcel = cell_index(2, ij)
  do i = 1, N(icel)
    do j = 1, N(jcel)
      rij(1) = coord(1, i, icel) - coord(1, j, jcel)
      rij(2) = coord(2, i, icel) - coord(2, j, jcel)
      rij(3) = coord(3, i, icel) - coord(3, j, jcel)
      dij = sqrt(rij(1)2 + rij(2)2 + rij(3)2)
      if (dij < pairlist_cutoff) then
        Neighbor(i, ij) = Neighbor(i, ij) + 1
        k = Neighbor(i, ij)
        write Neighbor_list(k, i, ij)
      end if
    end do
  end do
end do
```

M: number of cell pairs
First cell index of the cell pair
Second cell index of the cell pair
N(icel): Number of atoms in icel-th cell

Algorithm 2. Real-space nonbonded interaction kernel used for GENESIS 1.0-1.3.

```
do ij = 1, M
  icel = cell_index(1, ij)
  jcel = cell_index(2, ij)
  do i = 1, N(icel)
    force_temp(1:3) = 0.0
    do k = 1, Neighbor(i, ij)
      j = Neighbor_list(k, i, ij)
      rij(1) = coord(1, i, icel) - coord(1, j, jcel)
      rij(2) = coord(2, i, icel) - coord(2, j, jcel)
      rij(3) = coord(3, i, icel) - coord(3, j, jcel)
      dij = sqrt(rij(1)2 + rij(2)2 + rij(3)2)
      calculate f(1:3): force component from given distance
      force_temp(1) = force_temp(1) - f(1)
      force_temp(2) = force_temp(2) - f(2)
      force_temp(3) = force_temp(3) - f(3)
    end do
  end do
```

```

    force(1,j,jcel) = force(1,j,jcel) + f (1)
    force(2,j,jcel) = force(2,j,jcel) + f (2)
    force(3,j,jcel) = force(3,j,jcel) + f (3)
  end do
  force(1,i,icel) = force(1,i,icel) + force_temp(1)
  force(2,i,icel) = force(2,i,icel) + force_temp(2)
  force(3,i,icel) = force(3,i,icel) + force_temp(3)
end do
end do

```

Algorithm 3. New neighbor search algorithm.

```

do ij = 1, M
  icel = cell_index(1,ij)
  jcel = cell_index(2,ij)
  Neighbor(i,ij) = 0
  do i = 1, N(icel)
    do j = 1, N(jcel)
      rij(1) = coord(1,i,icel)-coord(1,j,jcel)
      rij(2) = coord(2,i,icel)-coord(2,j,jcel)
      rij(3) = coord(3,i,icel)-coord(3,j,jcel)
      dij = sqrt(rij(1)2+rij(2)2+rij(3)2)
      if (dij pairlist cutoff) then
        Neighbor(i,ij) = 1
        exit the do loop
      end if
    end do
  end do
end do

```

Algorithm 4 Real-space nonbonded interaction kernel used for KNL.

```

do ij = 1, M
  icel = cell_index(1,ij)
  jcel = cell_index(2,ij)
  do i = 1, N(icel)
    if (Neighbor(i,ij) == 0) cycle
    force_temp(1:3) = 0.0 do j = 1, N(jcel)
      rij(1) = coord(i,1,icel)-coord(j,1,jcel)
      rij(2) = coord(i,2,icel)-coord(j,2,jcel)
      rij(3) = coord(i,3,icel)-coord(j,3,jcel)
      calculate f(1:3):force component from given distance
      force_temp(1) = force_temp(1) - f(1)
      force_temp(2) = force_temp(2) - f(2)
      force_temp(3) = force_temp(3) - f(3)
      force(j,1,jcel) = force(j,1,jcel) + f(1)
      force(j,2,jcel) = force(j,2,jcel) + f(2)
      force(j,3,jcel) = force(j,3,jcel) + f(3)
    end do
    force(i,1,icel) = force(i,1,icel) + force_temp(1)
    force(i,2,icel) = force(i,2,icel) + force_temp(2)
    force(i,3,icel) = force(i,3,icel) + force_temp(3) end do
  end do

```

which are often neglected in MD calculations in materials science. Here, we optimized the most time-consuming real-space nonbonded interactions by using a simple neighbor scheme with reduced memory, contiguous memory access, and change of data type from array of structures (AoS) to structure of arrays (SoA), particularly useful for the Intel Xeon Phi processor. We found that the new algorithm accelerates the speed by increasing SIMD performance. Long-range electrostatic interactions with the FFT were optimized by minimizing the number of processes involved in communications by volumetric decomposition. Our developments have been tested for 230 million atom and 1 billion atom systems with 1024^3 and 2048^3 PME grids, showing good parallel efficiency even when using more than 65,000 MPI processes, giving rise to the first billion atom simulation and also the first fully atomistic MD simulation of an entire gene.

Acknowledgments

This research was conducted using the Fujitsu PRIMERGY CX600M1/CX1640M1 (OakforestPACS) in the Information Technology Center of the University of Tokyo (project ID: gh50) and joint Center for Advanced HPC by HPCI system research project (project ID: hp180155) and Los Alamos National Laboratory high performance computing resources. This research was supported in part by a Grant-in-Aid for Scientific Research on Innovative Areas (JSPS KAKENHI Grant no. 26119006) (to YS), a MEXT grant as "Priority Issue on Post-K computer (Building Innovative Drug Discovery Infrastructure Through Functional Control of Biomolecular Systems)" (to YS), a grant from JST CREST on "Structural Life Science and Advanced Core Technologies for Innovative Life Science Research" (to YS). NIGMS support from award R35GM122562 to TS is gratefully acknowledged. KS was supported by LANL LDRD. AL was supported by the Center for Nonlinear Studies (CNLS). We thank RIKEN pioneering project on Integrated Lipidology and Dynamic Structural Biology (to YS). We gratefully acknowledge the support of the U.S. Department of Energy through the LANL LDRD program as well as Los Alamos National Laboratory Institutional Computing.

Keywords: high performance computing · biomolecular simulation · 3D modeling · GENESIS MD software

How to cite this article: J. Jung, W. Nishima, M. Daniels, G. Bascom, C. Kobayashi, A. Adedoyin, M. Wall, A. Lappala, D. Phillips, W. Fischer, C.-S. Tung, T. Schlick, Y. Sugita, K. Y. Sanbonmatsu. *J. Comput. Chem.* **2019**, , .. DOI: 10.1002/jcc.25840

- [1] J. A. McCammon, B. R. Gelin, M. Karplus, *Nature* **1977**, 267, 585. <https://doi.org/10.1038/267585a0>.
- [2] T. Narumi et al. Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, Tampa, Florida. ACM: New York, **2006**.
- [3] D. E. Shaw et al. ISCA '07: Proceedings of the 34th Annual International Symposium on Computer Architecture, San Diego, California. ACM: New York, **2007**.
- [4] D. E. Shaw et al. SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, Louisiana. IEEE press: Piscataway, NJ, USA, **2014**.
- [5] D. E. Shaw, P. Maragakis, K. Lindorff-Larsen, S. Piana, R. O. Dror, M. P. Eastwood, J. A. Bank, J. M. Jumper, J. K. Salmon, Y. Shan, W. Wriggers, *Science* **2010**, 330, 341. <https://doi.org/10.1126/science.1187409>.
- [6] M. J. Harvey, G. Giupponi, G. De Fabritiis, *J. Chem. Theory Comput.* **2009**, 5, 1632. <https://doi.org/10.1021/ct9000685>.
- [7] A. P. Ruymgaart, A. E. Cardenas, R. Elber, *J. Chem. Theory Comput.* **2011**, 7, 3072. <https://doi.org/10.1021/ct200360f>.
- [8] L. C. Pierce, R. Salomon-Ferrer, F. d. O. C. Augusto, J. A. McCammon, R. C. Walker, *J. Chem. Theory Comput.* **2012**, 8, 2997. <https://doi.org/10.1021/ct300284c>.
- [9] J. C. Phillips, Y. Sun, N. Jain, E. J. Bohm & L. V. Kalé. SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, Louisiana. IEEE press: Piscataway, NJ, USA, **2014**.
- [10] M. J. Abraham, et al., *Software X* **2015**, 1-2, 19.
- [11] W. M. Brown, J. M. Y. Carrillo, N. Gavhane, F. M. Thakkar, S. J. Plimpton, *Comput. Phys. Commun.* **2015**, 195, 95. <https://doi.org/10.1016/j.cpc.2015.05.004>.
- [12] J. Jung, A. Nourse, C. Kobayashi, Y. Sugita, *J. Chem. Theory Comput.* **2016**, 12, 4947. <https://doi.org/10.1021/acs.jctc.6b00241>.
- [13] P. J. Needham, A. Bhuiyan, R. C. Walker, *Comput. Phys. Commun.* **2016**, 201, 95. <https://doi.org/10.1016/j.cpc.2015.12.025>.
- [14] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L. P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, V. S. Pande, *PLoS Comput. Biol.* **2017**, 13, e1005659. <https://doi.org/10.1371/journal.pcbi.1005659>.
- [15] G. P. Zhao, et al., *Nature* **2013**, 497, 643. <https://doi.org/10.1038/nature12162>.
- [16] I. Yu, T. Mori, T. Ando, R. Harada, J. Jung, Y. Sugita, M. Feig, *Elife* **2016**, 5, e19274. <https://doi.org/10.7554/eLife.19274>.
- [17] T. Darden, D. York, L. Pedersen, *J. Chem. Phys.* **1993**, 98, 10089. <https://doi.org/10.1063/1.464397>.
- [18] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, L. G. Pedersen, *J. Chem. Phys.* **1995**, 103, 8577. <https://doi.org/10.1063/1.470117>.
- [19] J. Jung, T. Mori, C. Kobayashi, Y. Matsunaga, T. Yoda, M. Feig, Y. Sugita, *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2015**, 5, 310. <https://doi.org/10.1002/wcms.1220>.
- [20] C. Kobayashi, J. Jung, Y. Matsunaga, T. Mori, T. Ando, K. Tamura, M. Kamiya, Y. Sugita, *J. Comput. Chem.* **2017**, 38, 2193. <https://doi.org/10.1002/jcc.24874>.
- [21] J. Jung, T. Mori, Y. Sugita, *J. Comput. Chem.* **2014**, 35, 1064. <https://doi.org/10.1002/jcc.23591>.
- [22] J. Jung, C. Kobayashi, T. Imamura, Y. Sugita, *Comput. Phys. Commun.* **2016**, 200, 57.
- [23] G. D. Bascom, K. Y. Sanbonmatsu, T. Schlick, *J. Phys. Chem. B* **2016**, 120, 8642. <https://doi.org/10.1021/acs.jpcc.6b03197>.
- [24] Q. Zhang, D. A. Beard, T. Schlick, *J. Comput. Chem.* **2003**, 24, 2063. <https://doi.org/10.1002/jcc.10337>.
- [25] E. S. Carter, II, C. S. Tung, *Comput. Appl. Biosci.* **1996**, 12, 25.
- [26] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, K. Schulten, *J. Comput. Chem.* **2005**, 26, 1781. <https://doi.org/10.1002/jcc.20289>.
- [27] J. Jung, T. Mori, Y. Sugita, *J. Comput. Chem.* **2013**, 34, 2412. <https://doi.org/10.1002/jcc.23404>.
- [28] J. Huang, A. D. MacKerell, *J. Comput. Chem.* **2013**, 34, 2135. <https://doi.org/10.1002/jcc.23354>.
- [29] D. J. Price, C. L. Brooks, *J. Chem. Phys.* **2004**, 121, 10096. <https://doi.org/10.1063/1.1808117>.
- [30] J. P. Ryckaert, G. Ciccotti, H. J. C. Berendsen, *J. Comput. Phys.* **1977**, 23, 327. [https://doi.org/10.1016/0021-9991\(77\)90098-5](https://doi.org/10.1016/0021-9991(77)90098-5).
- [31] H. C. Andersen, *J. Comput. Phys.* **1983**, 52, 24. [https://doi.org/10.1016/0021-9991\(83\)90014-1](https://doi.org/10.1016/0021-9991(83)90014-1).
- [32] S. Miyamoto, P. A. Kollman, *J. Comput. Chem.* **1992**, 13, 952. <https://doi.org/10.1002/jcc.540130805>.
- [33] N. A. Baker, D. Sept, S. Joseph, M. J. Holst, J. A. McCammon, *Proc. Natl. Acad. Sci. U. S. A.* **2001**, 98, 10037. <https://doi.org/10.1073/pnas.181342398>.
- [34] R. L. Hayes, J. K. Noel, U. Mohanty, P. C. Whitford, S. P. Hennyly, J. N. Onuchic, K. Y. Sanbonmatsu, *J. Am. Chem. Soc.* **2012**, 134, 12043. <https://doi.org/10.1021/ja301454u>.

APPENDIX A.

Forward FFT with 1d_Alltoall scheme

1. MPI_Alltoall communication in x dimension (*MPI_Alltoallx*)
2. FFT in x dimension (*FFTx*)
3. MPI_Alltoall communication in x dimension (*MPI_Alltoallx*)
4. MPI_Alltoall communication in y dimension (*MPI_Alltoally*)
5. FFT in y dimension (*FFTy*)
6. MPI_Alltoall communication in y dimension (*MPI_Alltoally*)

7. MPI_Alltoall communication in z dimension (*MPI_Alltoallz*)
8. FFT in z dimension (*FFTz*).

5. MPI_Alltoall communication in z dimension (*MPI_Alltoallz*)
6. FFT in z dimension (*FFTz*).

Forward FFT with 2d_Alltoall scheme

1. MPI_Alltoall communication in x dimension (*MPI_Alltoallx*)
2. FFT in x dimension (*FFTx*)
3. MPI_Alltoall communication in xy dimension (*MPI_Alltoallxy*)
4. FFT in y dimension (*FFTy*)

Received: 31 December 2018
Revised: 6 March 2019
Accepted: 20 March 2019
Published online on 17 April 2019
