論文 / 著書情報
Article / Book Information

Title	Petascale molecular dynamics simulation using the fast multipole method on K computer
Author	Yousuke Ohno, Rio Yokota, Hiroshi Koyama, Gentaro Morimoto, Aki Hasegawa, Gen Masumoto, Noriaki Okimoto, Yoshinori Hirano, Huda Ibeid, Tetsu Narumi, Makoto Taiji
Journal/Book name	Computer Physics Communications, Vol. 185, No. 10, pp. 2575–2585
Issue date	2014, 6
URL	http://www.journals.elsevier.com/computer-physics-communications
DOI	http://dx.doi.org/10.1016/j.cpc.2014.06.004
Note	このファイルは著者（最終）版です。 This file is author (final) version.

License

Petascale Molecular Dynamics Simulation Using the Fast Multipole Method on K computer

Yousuke Ohno[a,b], Rio Yokota[c], Hiroshi Koyama[1b,d], Gentaro Morimoto[a,b,e], Aki Hasegawa[2b,d], Gen Masumoto[3b,d], Noriaki Okimoto[a,b], Yoshinori Hirano[a,b], Huda Ibeid[c], Tetsu Narumi[f], Makoto Taiji[a,b,d,e]

[a] *Laboratory for Computational Molecular Design, Computational Biology Research Core, RIKEN Quantitative Biology Center*
[b] *High-performance Computing Team, RIKEN Computational Science Research Program*
[c] *Computer, Electrical, and Mathematical and Engineering Division, King Abdullah University of Science and Technology*
[d] *High Performance Computing Development Team, RIKEN HPCI Program for Computational Life Sciences*
[e] *Processor Research Team, RIKEN Advanced Institute for Computational Science*
[f] *Department of Computer Science, The University of Electro-Communications*

Abstract

In this paper, we report all-atom simulations of molecular crowding – a result from the full node simulation on the "K computer", which is a 10-PFLOPS supercomputer in Japan. The capability of this machine enables us to perform simulation of crowded cellular environments, which are more realistic compared to conventional MD simulations where proteins are simulated in isolation. Living cells are "crowded" because macromolecules comprise ∼30% of their molecular weight. Recently, the effects of crowded cellular environments on protein stability have been revealed through in-cell NMR

[1] Present address: First-Principles SimulationG, Computational Materials Science Unit, Advanced Key Technologies Division, National Institute for Materials Science

[2] Present address: Laboratory for Computational Molecular Design, Computational Biology Research Core, RIKEN Quantitative Biology Center

[3] Present address: Advanced Center for Computing and Communication, RIKEN

spectroscopy. To measure the performance of the "K computer", we performed all-atom classical molecular dynamics simulations of two systems: target proteins in a solvent, and target proteins in an environment of molecular crowders that mimic the conditions of a living cell. Using the full system, we achieved 4.4 PFLOPS during a 520 million-atom simulation with cutoff of 28 Å. Furthermore, we discuss the performance and scaling of fast multipole methods for molecular dynamics simulations on the "K computer", as well as comparisons with Ewald summation methods.
Keywords: Molecular dynamics; Molecular crowding; Fast Multipole Method; K computer

1. Introduction

The human body is made up of 60 trillion cells that consist of various biomolecules. Each biomolecule plays an important role in biological activities; the functions of biomolecules are realized by their inherent structures and dynamics. Recent experimental techniques such as X-ray crystallographic analysis and nuclear magnetic resonance (NMR) spectroscopy enable us to measure both the structure and fluctuation of biomolecules. Computational approaches have also been attempted to study the dynamic and thermodynamic properties of biomolecules. Among these methods, molecular dynamics (MD) simulation is the most prevalent. In MD simulations, atoms are treated as a point mass, and their orbits are calculated using Newton's equation of motion. A typical time step size Δt of the simulation is in the order of femtoseconds, while sampling over microseconds or milliseconds is sometimes necessary to observe events of interest. Therefore, MD simu-

lations require large computational power, and they are often used as HPC benchmarks. In this paper, we report large-scale MD simulations of proteins in a crowded environment on the massively parallel "K computer".

In typical MD simulations, proteins are immersed in solvents or lipids. This corresponds to an ideal environment in a test tube – *in vitro*. On the other hand, living cells are crowded because macromolecules comprise ∼30% of their molecular weight [1, 2, 3]. An actual intracellular environment *in vivo* is extremely different from an *in vitro* environment [4]. In order to understand the structure and dynamics of biomolecules in a living cell, systems with crowded environments have been studied using crowding agents [2].

In his pioneering work, Minton proposed the basic idea that the excluded volume of macromolecules strongly influences the stabilities and conformations of biomolecules [1, 5]. The excluded-volume effect would seem to make the conformation of a protein more compact. However, in a real crowding environment, the native protein does not always maintain the compact conformation and sometimes becomes unstable.

Recently, the structural and dynamic properties of proteins in *Escherichia coli* and HeLa cell have been studied using in-cell NMR spectroscopy [6, 7]. For TTHA1718 (a protein from *Thermus thermophilus* HB8, hereafter TTHA) and ubiquitin, these studies revealed that a crowded environment reduces the stability and changes the conformation of proteins. This is contrary to the widely accepted understanding of protein behavior in a crowded environment that proteins are stabilized by the excluded-volume effect.

Elcock *et al.* performed a Brownian dynamics simulation of 1,008 molecules consisting of 50 types of proteins [8]. During the simulation, they treated a

protein as a rigid body. They estimated free energy, including protein-protein interactions, using the particle insertion method, and they proposed that the structures of some proteins become unstable in a macromolecular crowding environment in contrast to an *in vitro* one. The movie based on their simulation beautifully animated a picture of the intracellular environment by Goodsell *et al.* [9]. Their study shows the importance of protein-protein interactions on protein stability as well as the excluded-volume effect.

Previous computational studies on macromolecular crowding have used reduced models because of the high computational costs of all-atom simulations. However, explicit solvent models are known to be essential for detailed analyses of protein dynamics. The precise treatment of protein flexibility is also essential for the study of protein fluctuations. In this study, we aim to clarify the effects of these important factors, which have been omitted in previous studies, using a cutting-edge algorithm on a state-of-the-art high-performance computer. The K computer system has 82,944 processors that consist of 663,552 cores, and a theoretical peak performance of 10.6 PFLOPS. Therefore, we needed to extend the parallel efficiency up to nearly a million cores. To achieve this goal, we have developed a scalable MD simulation code for biomolecular systems, using a highly parallel fast multipole method (FMM).

The major bottleneck for the scalability of MD simulations is the long range force calculation that is commonly performed by a particle-mesh Ewald method (PME) [15]. The 3-D FFT in PME prevents the simulation from scaling to hundreds of thousands of cores. The FMM on the other hand, is known to scale to the full node of the largest supercomputers of today [42]. This dif-

ference comes from the difference in the communication pattern of FFT and FMM. A 3-D FFT requires a global transpose of the data. On P processors the optimal communication complexity is $\mathcal{O}(\sqrt{P})$ when 2-D decomposition (pencil decomposition) is used. The FMM has $\mathcal{O}(logP)$ communication complexity because the volume of communication decays logarithmically with the distance. Furthermore, our present FMM uses a hierarchical local communication scheme, which eliminates the use of `MPI_Alltoallv` type communications. This communication scheme maps well to the 6-D torus network of the K computer.

There have been a few attempts to use FMM in MD simulations [43, 44, 45, 46]. However, these previous attempts do not focus on high parallel scalability and only report results on less than 1000 cores. The present work aims for a much higher parallel performance on up to 196,608 cores.

For measurement of performance of MD simulation on "K computer", we performed all-atom classical MD simulations of TTHA using ovalbumins (Figure 1) as crowders that mimic the conditions of a living cell [10].

The paper is organized as follows. In section 2 we explain the computational methods, and in section 3 we describe the outline of the K computer. Following that, we present the benchmark results of the large-scale MD simulations in section 4. The comparison between results of FMM and PME on macromolecular crowding are presented in section 5. Finally, Discussions and future perspectives are given in section 6 from the viewpoints of both HPC and life sciences.

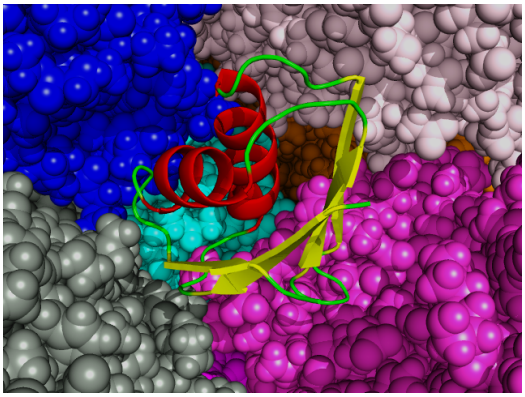


Figure 1: Vicinal area of TTHA for the *in vivo* system. The TTHA molecule is drawn using the ribbon model, and ovalbumin molecules are drawn using the space-filled model. Each ovalbumin molecule is colored differently.

2. Computational Methods

In a classical MD simulation, every atom's position is updated at each discrete time step. When the number of atoms is N_{atom}, the computational cost should be proportional to the product of N_{atom} and the number of time steps if a cutoff is applied for long-range electrostatics. Therefore, our goal is to design software that can scale up to large N_{atom} and N_{node} while keeping N_{atom}/N_{node} as small as possible.

We designed our program as a core library that could be easily customized, extended, and reused in a large variety of MD programs. For this purpose, we chose C++ as the base language. The innermost kernel loop, where Coulomb and van der Waals (Lennard-Jones model) force are calculated, is also provided in Fortran. The user may select either the C++ kernel or the Fortran kernel to achieve better performance, depending on the environment.

2.1. Basic MD Functions

Our software supports the basic functions for protein MD simulations that are common for many software packages. MD simulation is composed of evaluation of forces between atoms and the integration of their trajectories. The following equations express the exact MD equations of our computations:

$$m_i \frac{d^2 \boldsymbol{r}_i}{dt^2} = -\nabla U(\boldsymbol{r}_1, \cdots, \boldsymbol{r}_{N_{atom}}) \tag{1}$$

$$\begin{aligned} U(\{\boldsymbol{r}_i\}) &= \sum_{\text{bond}} \frac{1}{2} k_b (r - r_0)^2 + \sum_{\text{angle}} \frac{1}{2} k_a (\theta - \theta_0)^2 \\ &+ \sum_{\text{torsions}} \frac{1}{2} V_n [1 + \cos(n\omega - \gamma)] \\ &+ \sum_{|r_{ij}|<R_c} \left\{ \epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] \right. \\ &\left. + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} + \Psi(r_{ij}) \right\} \end{aligned} \tag{2}$$

where m_i and $\boldsymbol{r}_i$ represent the mass and position of the i-th particle, respectively, $r_{ij} = |\boldsymbol{r}_i - \boldsymbol{r}_j|$, and a smoothed 3rd degree polynomial, Ψ, is the GROMACS's shift function [11]. In Eq. (2), the first three terms express bonded forces, and the last summation describes nonbonded forces. Note that bonded forces such as bond stretching, angle, and torsion refer only to several topologically connected atoms; therefore, their computational costs are relatively lower than that of nonbonded forces.

Since we calculate approximated forces by Eq. (2), we need to specify molecules by stating atom types and an atom-to-atom topology. In addition, it is essential to have force field parameters that map atom types to the parameters in Eq. (2).

We have implemented a symplectic time integration (Verlet method)

with SHAKE[12]/RATTLE[13] and SETTLE [14] holonomic constraint algorithms, and several well-established numerical algorithms to solve the above equations efficiently for both short-range and long-range interactions. In this work, we use a shift function for the cutoff method for short-range contributions, and FMM for the long-range contributions.

2.2. Non-bonded Short Range Force

Although almost all algorithms in MD simulations have linear complexity, nonbonded forces such as Coulomb and Lennard-Jones potentials are dominant because of the large number of interacting atom companions. When the cutoff method is used for these nonbonded force calculations, the number of interacting atoms depends on the cutoff radius. For example, a typical biomolecular circumstance involves 0.1 atoms in one cubic Angstrom; a spherical volume of 10 Å radius includes approximately 419 atoms. For the efficient calculation of the nonbonded forces, we organize a cell index and a "pair list", which is a list of atom pairs within the cutoff radius. All atoms are distributed to "cells", which are cubic space decompositions of the simulation space. For each individual cell, we build a list of cells within the reach of the cutoff radius from the cell. Since the paring atoms are located only in the cells in the list, we can limit the search volume to a region with a fixed volume of $\{2(\text{cutoff radius} + \text{cell size})\}^3$.

If pair lists are updated every time step, the total calculation cost is the same as the cost using only the cell index without the pair list. To reduce the cost, our code updates the pair list once every 40 steps. For the generation of the pair list in this case, we must use a longer distance than the cutoff radius because the list must include atoms that may move inside the cutoff

sphere in the next 40 steps. The fraction of pairs within the cutoff radius in the pair list depends on this pair-list margin and cutoff. We use 2 Å margins in our code, and the fractions are 0.51, 0.73, and 0.81 for cutoffs of 8, 18, and 28 Å respectively.

Despite the margin, the particles may jump inside the cutoff radius during the 40-step interval. In such cases, our code rolls back to the previous update step and recalculates with a decreased interval. In our 100,000-step simulation with 180 million atoms at a 28 Å cutoff, we observed 16 occurrences of such events, which correspond to only 0.6% redundant calculation. Because these events are so rare, the redundant calculations have no serious impact on performance.

2.3. Non-bonded Long Range Force

Particle mesh Ewald (PME) [15] and its variants such as Gaussian splitting Ewald (GSE)[16] are efficient methods for calculating lattice sums with periodic boundary conditions. Although, these lattice sum methods are accurate for periodic systems such as crystals, they suffer from long-range correlation artifacts and anisotropy effects for noncrystalline systems such as cellular environments[17, 18, 19]. To avoid such artifacts, variation of cutoff methods that include long-range effects have been proposed. For example, isotropic periodic sum (IPS) [20], which sums over the isotropic periodic images, and the Wolf method or zero-dipole summation [21, 22], which use potential damping under charge neutral or zero-dipole conditions. The accuracy of such cutoff methods depend on the uniformity of the far field, which means that the cutoff length must be larger than the scale of the structure. Therefore, a cutoff method with long cutoff length can be an effective tool for

MD simulation of large protein structures. Especially when arithmetic is becoming cheaper relative to bandwidth, the selection of the optimal algorithm requires careful consideration.

Another disadvantage of PME is its scalability, since it requires a large amount of global communication due to the nature of the FFT. It is difficult to even achieve weak scaling (not to mention strong scaling) for a large number of CPUs with full electrostatics. Note that several studies (e.g., [23]) have attempted to improve the network performance of long-range force calculations. On the other hand, fast multipole methods (FMM) can reduce the amount of communication in long-range interactions from $\mathcal{O}(\sqrt{p})$ to $\mathcal{O}(\log p)$, where p is the number of processes. In this work, we show that this theoretical upper bound of $\mathcal{O}(\log p)$ actually holds in a real application on a 10-PFLOPS machine.

2.4. Periodic Fast Multipole Method

In the present simulations, an FMM with periodic boundary conditions [24] is used to calculate the long range force. An FMM can approximate the long-range forces using multipole expansions and local expansions, and is able to calculate the interaction of N bodies in $\mathcal{O}(\mathcal{N})$ time. The multipole expansions and local expansions converge only when the center of expansion is close to the source and target, respectively. A tree structure is used to create a hierarchy of cells, which can be used to efficiently calculate the multipole-to-local translations between optimally separated cells.

When the distribution of bodies is not uniform, the tree structure is unbalanced and parallelization of the FMM becomes a non-trivial task. However, for molecular dynamics simulations, where water molecules uniformly fill the

entire domain, the FMM can use a full tree structure. This has many implications when constructing a highly parallel FMM. First, the data structure of a full tree is much simpler than that of an adaptive tree. Since one can assume that all nodes of the octree are full, multipole-to-local translation stencils are identical for every cell. The periodic boundary condition is another ingredient, which helps create an entirely homogeneous translation stencil even near the boundaries of the domain. This enables optimization techniques that exploit the symmetry of the translation stencils and precalculation of the multipole-to-local translation matrix. Second, the communication on distributed memory machines is much simpler for a full tree since the data structures on remote nodes are known. Again, the periodic boundary condition also helps to create a perfect load balance, since the boundary cells have a full stencil. Third, a full tree means that there is no need to reconstruct it once it is built on the first time step. It is only a matter of updating which atoms belong to which leaf cells. This is also a significant advantage when developing a highly parallel FMM code.

The simplicity of the partitioning and communication pattern in the current application allows us to analyze various performance models for the FMM on one of the largest systems –the K computer. An overall view of the FMM partitioning and communication is shown in Figure 2. We use different structures for the local tree and the global tree, where the local tree is always a cubic octree, while the global tree can have arbitrary dimensions of subdivision, *e.g.* $2 \times 3 \times 2$. This allows us to exploit the network topology of the K computer without sacrificing the performance benefits of having a cubic octree. Each leaf of the global tree is a root of a local tree owned by a

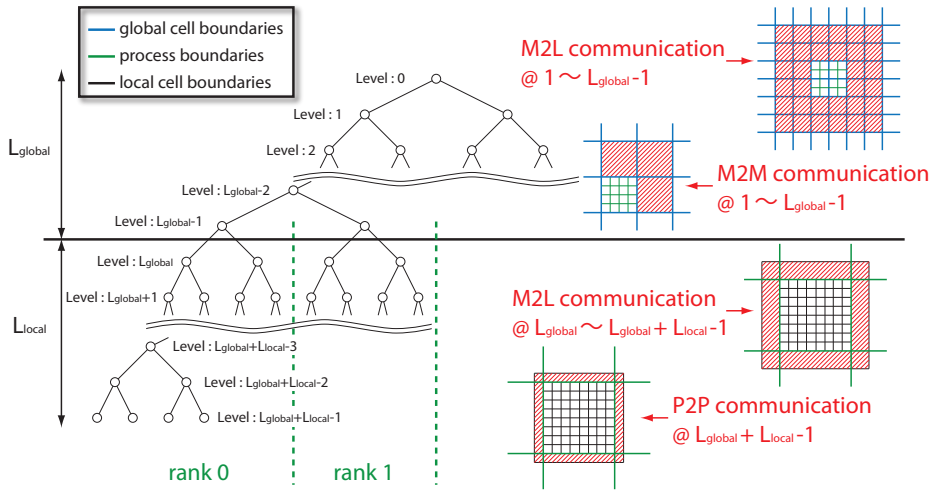


Figure 2: Partitioning of the FMM [25] global tree structure and communication stencils. Each leaf of the global tree is a root of a local tree in a particular MPI process, where the global tree has L_{global} levels, and the local tree has L_{local} levels. Each process stores only the local tree, and communicates the halo region at each level of the local and global tree as shown in the red hatched region in the four illustrations on the right. The blue, green, and black lines indicate global cell boundaries, process boundaries, local cell boundaries, respectively. The data in the red hatched region is sent using non-blocking one-to-one communication in single precision, and is overlapped with computation. P2P is the particle-to-particle kernel, M2L is the multipole-to-local kernel, and M2M is the multipole-to-multipole kernel (See [25] for details of these kernels).

particular MPI process, and the Morton index is assigned only locally. With this approach, the Morton index will never exceed the 64-bit limit even on Exascale machines. Since the tree is always full, we do not need to create explicit interaction lists or local essential trees. Furthermore, the amount of communication between all processes is known *a priori*, since the tree is always full.

The current FMM is part of the open source library *ExaFMM*[4], which is originally designed for adaptive tree structures and targets large scale heterogeneous systems [26]. We have created a separate module within *ExaFMM* that is optimized for non-adaptive trees with periodic boundary conditions on the K computer. Due to architectural similarities, users of BG/Q may benefit from the optimizations used in this version of *ExaFMM*.

3. The K Computer

3.1. System Overview

The K computer, manufactured by Fujitsu, is a 10-PFLOPS supercomputer developed with the initiative of RIKEN[27]. Each node consists of an 8-core CPU "SPARC64 VIIIfx," an interconnect chip (ICC), and memory. Both the CPU and ICC are water-cooled. It has 82,944 nodes that are connected by TOrus FUsion (Tofu), a six-dimensional (6-D) mesh/torus network. A special rack houses 96 nodes, disk drives, and I/O units. Linux is used as the operating system. Fortran/C/C++ compilers supporting OpenMP and a message passing interface (MPI) library have been developed and provided by Fujitsu.

3.2. Parallel Implementation

In many condensed-matter systems including biological systems, the density of atoms is quite uniform. Therefore, spatial domain decomposition (or geometrical decomposition) is effective and widely used for parallel MD simulations. Our code uses 3-D Cartesian cells, where the cells are used both for

[4]https://bitbucket.org/exafmm/exafmm

spatial decomposition and efficient generation of the atom pair list. Although the K computer provides an MPI interface, thread-level parallelization is recommended between cores because of the limited number of communication engines in the interconnect chip. In addition, thread-level parallelization using OpenMP (or automatic parallelization) is effective in SPARC64 VIIIfx. Therefore, hybrid parallelization at the process and thread levels is the standard programming model for the K computer. We applied thread-level parallelization using OpenMP to the force calculation kernel. Process-level parallelization using MPI was applied for inter-node communications. It is trivial to map our domain decomposed system with periodic boundary conditions to the torus network of the K computer.

3.3. Kernel Optimization

The current Fujitsu C++ compiler on the K computer cannot perform deep optimization for STL containers. For example, using a C-style array is able to achieve much higher SIMD vectorization rates than an STL vector. Therefore, the innermost subroutines were rewritten in Fortran or C-style. Basic optimization was achieved using the compiler option `-Kfast`, and further optimization was achieved by using the following additional options:

`-Ksimd=2`

: "if statement" translated using SIMD masked move or masked store

`-Kprefetch_indirect, prefetch_cache_level=1`

: prefetch indirect references to L1 cache

`-Kprefetch_iteration=8`

: prefetch to L1 cache for 8 loops ahead

The recommended optimization includes loop unrolling, software pipelin-

ing, loop blocking, loop fusion, sequential prefetching, SIMD vectorization, non-standard fast floating-point arithmetic, optimization of evaluation order, and fast evaluations of the reciprocal/the reciprocal of square root.

4. Performance Benchmarks

This section presents the results of the performance benchmarks of macromolecular simulations on the K computer. It is well known that the force calculation dominates the cost of an MD simulation. In the results of our benchmark, the force calculation and communication cost represent 83% and 14% of the CPU time, respectively. Here, we first provide the CPU core performance of the force calculation kernel and then describe the overall performance of macromolecular *in vitro* system simulations on the parallel processors. The *in vitro* system contains 418,707 atoms (4 proteins and 137,977 water molecules) in a (163 Å)^3 cube. We simply replicate this system for larger runs. The details of this molecular system are described in section 5.

Table 1: Performance on the main loop and the kernel loop for 418,707 atom simulation by 64 process

	old environment	Mar 2012	Oct 2012	
counter			[1]	detail
main loop				
Time consumption (ms/step)	129.7544	116.7608	116.1183	114.02
Performance (MFLOPS)	3,152,415	3,503,227	3,522,558	3,551,808
Efficiency	0.385	0.428	0.430	0.4336
kernel loop				
Time consumption (ms/step)	89.0891	87.1046	94.9585	77.36
Performance (MFLOPS)	4,549,128	4,660,130	4,306,453	5,180,754
Efficiency	0.555	0.569	0.526	0.632
SIMD ratio [2]	0.573	0.563	0.538	0.572
FLOP counts per pair of atom [3]	109.6	109.7	110.5	108.3

[1] Overhead of profiler was changed.

[2] Number of SIMD instructions / Number of all instructions

[3] Evaluated by the performance counter and estimated number of atom pairs

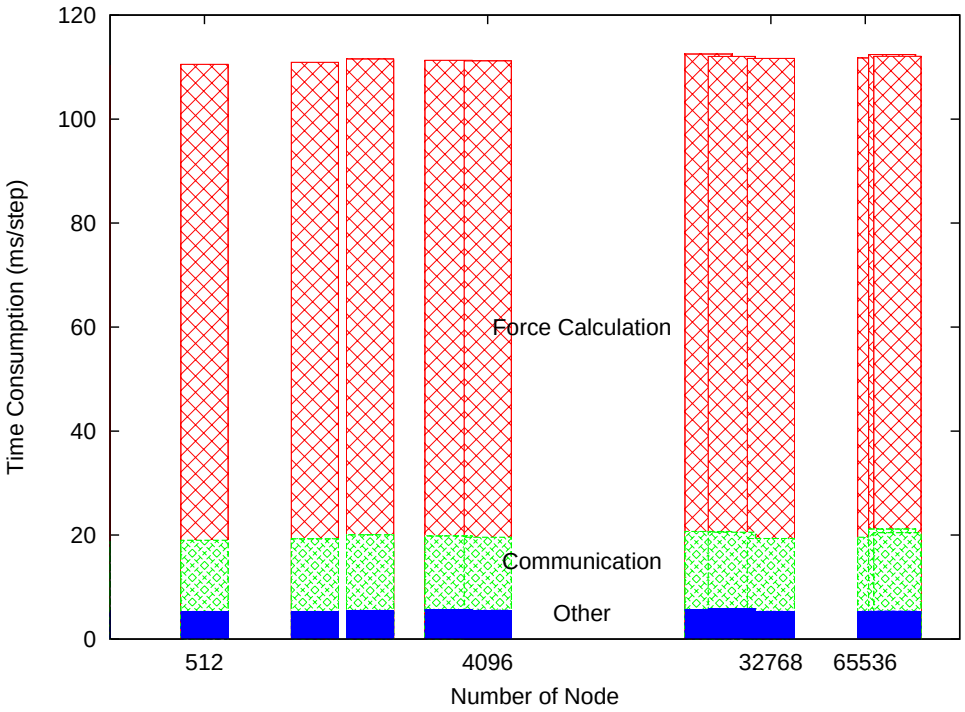


Figure 3: Weak scaling for 6,542 atoms/node. Constant wall clock times/step show the perfect scaling. Ratios of force calculation/communication/the others are also constant, and force calculation is dominant.

4.1. Efficiency of the Force Calculation Kernel

The CPU core performance was monitored by a hardware performance counter and reported using a profiler tool. Table 1 shows the result of the *in vitro* simulation model on 64 nodes (64 CPU, 512 cores). The scientific descriptions of these models are provided in the subsequent section. We have measured execution time with and without performance counters and confirmed that the overhead of the performance counter is less than 1.5%. Thus, we use the number of FLOP counts based on the performance counter as references. In Table 1, we summarize the performance count. The force

calculation kernel of the code ran at 63% of the theoretical peak, where 57% of the instructions were SIMD vectorized. We ran additional tests to analyze memory and cache performance. The result revealed that the rates of L1 and L2 cache miss were 3.6% and 0.59%, respectively, and the memory throughput was 8 GByte/s, 12% of the peak throughput. By analyzing the source code of the kernel loop, we obtained (62 + division + square-root) of FLOP counts per atom-pair calculation. This is consistent with the performance of ∼110 FLOP counts, as seen in Table 1.

In this simulation, we use 2 Å of margin with ∼28 Å cutoff on a spherical volume; thus, the hit ratio is ∼80%. In other words, ∼20% of the calculation is redundant. Action-reaction symmetry is also used to reduce half of the computation. When we use symmetry, the accumulation of the forces on the source particles $\mathbf{F}_j$, are required for every pair-wise force calculation. We found that write conflict between the cores causes serious performance degradation, and the run time increases, even though it does fewer operations. For an MD simulation with minimal operation count, a pair-list margin overhead of 80% and symmetry redundancy of 50% can be multiplied to yield a kernel efficiency of 63%. This is still 29% of the theoretical peak on the K computer, which is considered remarkably high since the peak assumes perfect utilization of fused-multiply-adds.

If a 23 Å cutoff radius with a 2 Å margin are used, building pair is 1.6 times more expensive than the kernel computation. With an update of the pair list every 40 steps, the overhead reduces to 4% of the kernel calculation. Therefore, we conclude that the pair list calculation is negligibly small.

Table 2: Conditions of the simulation for the peak performance

Model	432 *in vitro*
Number of atoms (N_{atom})	522,546,336
Cutoff radius	28 Å + 2 Å margin
Number of pairs per atom	8,835 (+2,031 in margin)
FLOP counts for 1,000 steps	510,481,566,661 MFLOP
Spatial size Å^3	$1956 \times 2119 \times 1304$
Potential energy	every time step
Topology of computation node	$48 \times 52 \times 32$
Number of nodes	79,872
Number of cores	638,976
Theoretical peak performance	10.223616 PFLOPS
Calculation time for 1,000 step	116.357 sec
Sustained performance	**4.387 PFLOPS**
Efficiency	0.429

Table 3: Time spent in blocks in millisecond per step for weak scaling (6,542 atom/64 nodes, calculate potential energy every step)

old environment

Number of nodes (N_{node})	64	512	32,768	73,728	79,872	82,944 [1]
Number of atoms (N_{atom})	418,707	3,349,656	214,377,984	482,350,464	522,546,336	723,525,696
Total T(ms/step)	127.009	127.266	137.055	144.439	132.135	198.124
Force calculation	102.310	104.345	102.312	102.271	102.729	139.699
Communication	19.205	17.402	29.233	36.713	23.921	51.634
Other	5.494	5.345	5.510	5.455	5.484	6.791
Performance (TFLOPS)	3.220	25.709	1527.86	3261.94	3862.83	3566.28
Efficiency	0.3930	0.3923	0.3643	0.3456	0.3778	0.3359

[1] The shape of the spatial decomposition is different for this case and the number of atoms per node is 8723.

Mar 2012 (trial use)

Number of nodes (N_{node})	64	3,072	6,144	12,288	24,576
Number of atoms (N_{atom})	418,707	20,097,936	40,195,872	80,391,744	160,783,488
Total T(ms/step)	116.733	122.271	123.642	124.506	120.826
Force calculation	97.436	97.082	99.380	96.476	97.441
Communication	13.772	19.245	18.755	22.483	17.868
Other	5.526	5.944	5.507	5.547	5.517
Performance (TFLOPS)	3.503	160.543	317.525	630.636	1299.66
Efficiency	0.4276	0.4083	0.4038	0.4009	0.4132

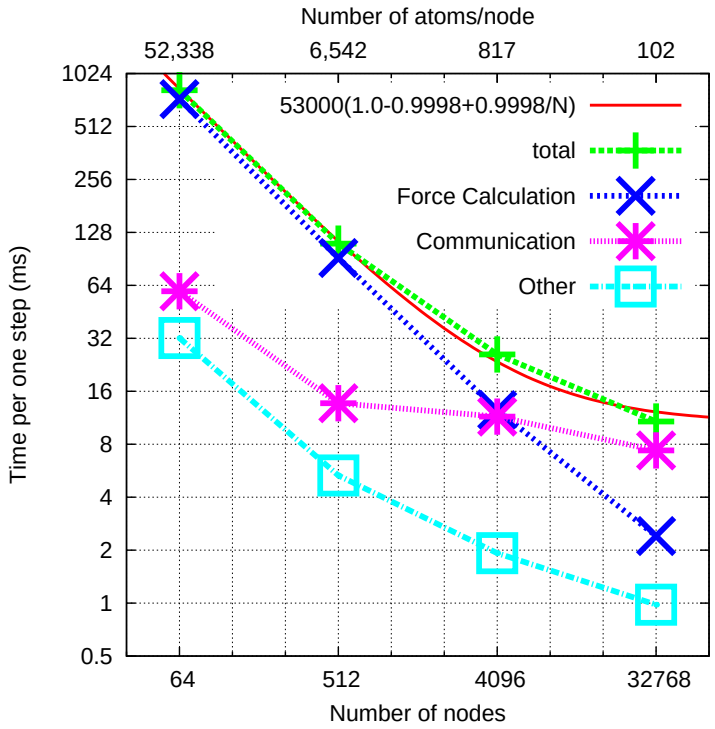


Figure 4: Strong scaling to number of atoms per node. The red curve shows Amdahl's law.

4.2. Sustained Performance of the Large-Scale Simulations

In this section, we report the sustained performance of the parallel runs. The wall-clock time of each block of the program is measured by inserting a Unix system call `gettimeofday()`. We have confirmed that the overhead of adding this system call is less than 0.5%. Using 523 millions atoms, we achieved sustained performance of 4.387 PFLOPS on 79,872 nodes, which is 42.9% of the theoretical peak of the 10.2 PFLOPS K computer. The parameters of this simulation are summarized in Table 2. Table 3 shows the results of several simulations on different system configurations. We estimated a total FLOP count in relation to the reference FLOP counts

Table 4: Time spent in blocks in millisecond per step for weak scaling (6,542 atom/64 nodes, calculate potential energy every 4 step)

	Oct 2012					
Number of nodes (N_{node})	64	512	4,096	32,768	79,872	82,944
Geometry	4x4x4	8x8x8	16x16x16	32x32x32	48x52x32	48x48x36 [1]
Number of atoms (N_{atom})	418,707	3,349,656	26,797,248	214,377,984	522,546,336	542,644,272
Total T (ms/step)	109.058	110.535	111.186	111.672	112.414	112.085
Force calculation	91.622	91.528	91.601	92.329	91.262	91.641
Communication	12.124	13.694	14.011	14.012	15.820	15.067
Other	5.312	5.313	5.574	5.332	5.332	5.376
Performance (TFLOPS)	3.000	25.00	201.0	1599	3871	4031
Efficiency	0.390	0.385	0.383	0.381	0.379	0.380

[1] In this case, the geometry of the network was different, 48x54x32.

provided in Table 1. A FLOPS value is defined as a total FLOP count divided by a wall-clock time. All benchmark runs in Table 3 were performed for 1,000 time steps using a 28 Å cutoff.

Table 3 and Figure 3 show the time spent in blocks per step for the weak scaling of the *in vitro* system (418,707 atoms/64 nodes). These times did not increase because almost all communications were local, except for a small number of "all gather" calls for the temperature calculation. Parallel efficiency, which is defined by T(64)/T(79,872), was above 0.96. Table 5 and Figure 4 show the operation time of a strong scaling *in vitro* system against number of atoms per node. Time spent on the force calculation decreased proportionally to the number of nodes N_{node}, as expected. Theoretically, the communication time is proportional to $(N_{node})^{-2/3}$, since the import volume per node is proportional to a ratio of a surface and a volume of a region assigned to each node. Though, the points of the measurements are not enough to reproduce the exact theoretical curve, we see the weak decrease of the communication time in the case of a small number of nodes. Beyond

Table 5: Time spent in blocks in millisecond per step for strong scaling (3,349,656 atom, calculate potential energy every 4 step)

Number of nodes (N_{node})	64	512	4.096	32,768
Number of atoms/node	52,338	6,542	817	102
Total	823.07	110.54	25.90	10.77
Force calculation	731.66	91.53	12.50	2.41
Communication	49.28	13.69	11.48	7.38
Other	1.00	0.93	0.50	0.15

6,400 atoms/node, we observed a saturation, which was caused by an increase in communication targets per node. In the case of 100 atoms/node, the side length of the region assigned to each node was about 10 Å, which was 1/3 of the cutoff radius (28 Å). Thus, each node had to communicate with ±3 nodes in each direction; ignoring the corner cells. In this case, the number of communication target nodes becomes 310.

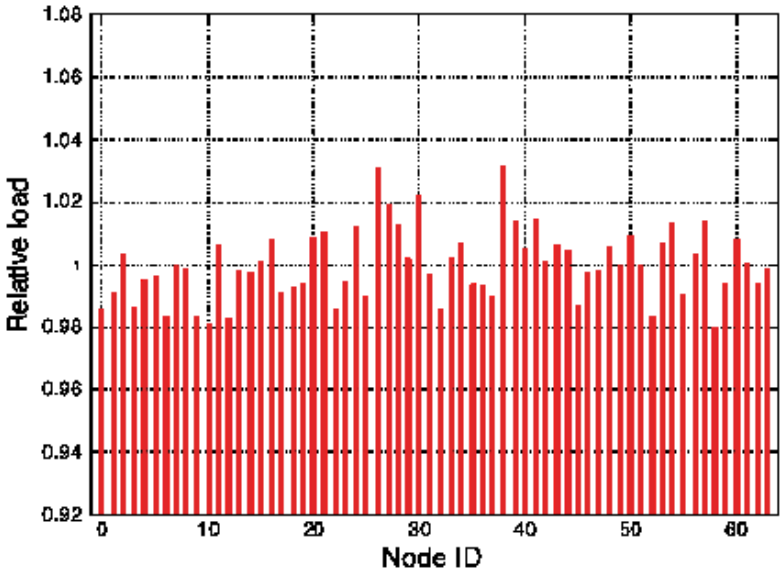
The total time of Amdahl's law [28], $T_1(1 - P + P/N_{node})$, is shown in Figure 4, where T_1 is the serial execution time, $1 - P$ is the fraction of the serial part, and N_{node} is the number of nodes. The estimated parallelization rate P is 0.9998. Note that the value of P improves as the number of atoms per node increase. At this number of atoms, the rate P is not sufficient for an 80,000-node system. The result shows that 50% efficiency of the ideal scaling was obtained at around 10,000 nodes with 8.4 million atoms. Thus, strong scaling was achieved up to 800 atoms per node. This means a system with 64 million atoms is sufficient to achieve 50% parallel efficiency for the full K computer system with 80,000 nodes.

For the simulation of the *in vitro* system using 64 nodes, we observed the average traffic of 355 Bytes of forces, and 2.19 MBytes of coordinates sent from each node every step; and 379 Bytes of forces, 3.68 MBytes of coordinates, and 3.99 KBytes of particle data were sent from each node once every 40 steps. The average traffic amounted to 4.46 MBytes/step bi-directional. The coordinate data accounts for 99.8% of the traffic. Since the communication time in the simulation was about 20 ms/step (shown in Table 3), a bandwidth of 223 MBytes/s was measured for bi-directional communication, which is 1.09% of the bandwidth between CPU and ICC. The

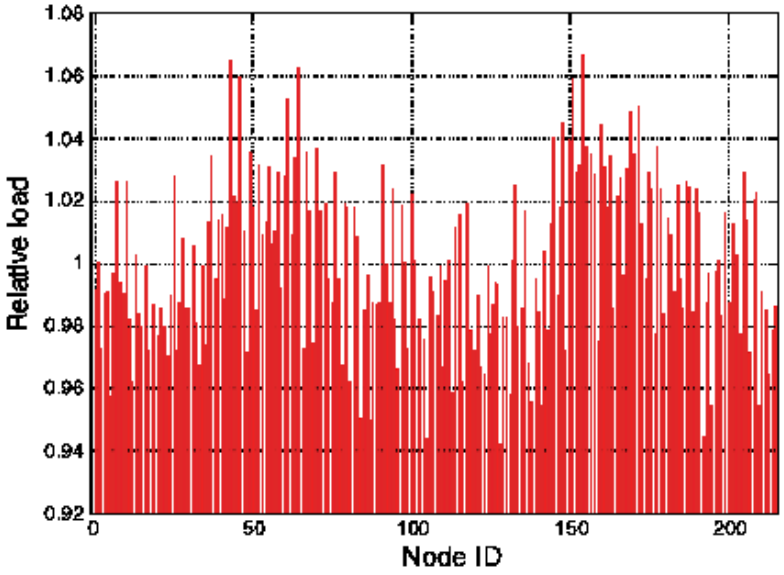
measured communication time includes the times of data packing/unpacking and copying in our code, and communications are overlapped with calculation partially in latest code. Therefore, the result does not represent the raw network performance of the K computer.

4.3. Load Balance

We now examine the workload balance among the nodes in the benchmark calculations. Figure 5 shows the calculation load of each node measured by FLOP counts for the *in vitro* and *in vivo* simulations, whose average numbers of atoms per node were 6,542 and 6,224, respectively. This figure shows a relatively large load imbalance in the *in vivo* simulation, compared to the *in vitro* one. The maximum load for the *in vitro* system exceeds the minimum by 5.2%, while the maximum for the *in vivo* system exceeds the minimum by 12.7%. Note that since the measurements are based on the FLOP counts, they do not reflect the fluctuation in the execution efficiency. As we have noted previously, most of the computational workload in MD simulations is due to the force calculation, which is organized by the atom pair lists. Because of the cutoff method used, the number of atom pairs per node depends on the local particle density. Therefore, the performance number on each node also depends on the local density. In the *in vitro* system, 98.9% of atoms belong to the water molecules, which are distributed in a spatially uniform manner. Thus, the particles in the *in vitro* system are distributed more uniformly than those in the *in vivo* system, which contains a large number of proteins. This results in a better load balance in the *in vitro* system. Dynamic load balancing could improve the load balance as shown in NAMD [29].



(a) *in vitro*



(b) *in vivo*

Figure 5: Computational workload (FLOP count) in each node.

4.4. Performance Model for FMM Communication

In this subsection we will shift our focus to the performance model of the communication in FMMs. We mentioned in the previous section that our FMM has a separate tree structure for the local and global tree. In order to construct a performance model for the communication in FMM, we estimate the amount of data that must be sent at each level of the hierarchy. Table 6 shows the number of cells that are sent, which correspond to the illustrations in Figure 2. L_{global} is the depth of the global tree, L_{local} is the depth of the local tree, N is the global number of particles, and p is the number of processes. Even though all the communications in the global tree are constant per level, the depth of the global tree L_{global} grows with $\log p$. On the other hand, it can be seen from Table 6 that the number of cells being sent in the local tree are always $\mathcal{O}(4^{L_{local}})$, since the 8^l term cancels out. The depth of the local tree L_{local} is proportional to $\log_8(N/p)$ so the upper bound for the local M2L and P2P is $\mathcal{O}(4^{\log_8(N/p)}) = \mathcal{O}((N/p)^{2/3})$. This can also be understood as the surface to volume ratio of the bottom two illustrations in Figure 2. Since N/p is constant for weak scaling and decreases for strong scaling, this part does not affect the asymptotic scalability of the FMM.

Table 6: Amount of communication in FMM

	# of Levels	# of Cells / Level	Upper bound
Global M2L	$L_{global} - 1$	26×8	$\mathcal{O}(\log p)$
Global M2M	$L_{global} - 1$	7	$\mathcal{O}(\log p)$
Local M2L	L_{local}	$(2^l + 4)^3 - 8^l$	$\mathcal{O}((N/p)^{2/3})$
Local P2P	1	$(2^l + 2)^3 - 8^l$	$\mathcal{O}((N/p)^{2/3})$

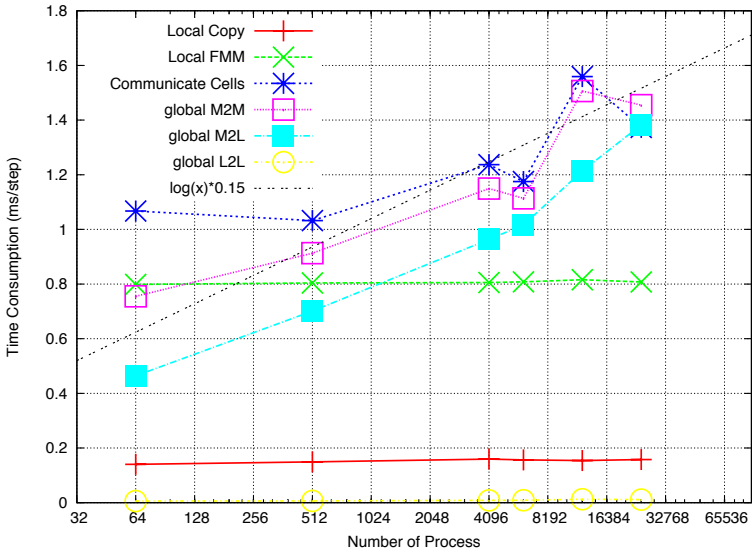


Figure 6: Weak scaling of FMM part of the MD simulation for 6,542 atoms/node. "Local FMM" is the aggregate time of all FMM kernels in the local tree, "Communicate Cells" is the time of the M2L communication in the local trees, "global M2M" and "global M2L" include both the communication time and computation time of the M2M and M2L kernels in the global tree. The dashed line is a reference for confirming the $\log p$ behavior of global communications.

Weak scaling of FMM part of the MD simulation for 6,542 atoms/node is shown in Figure 6. The execution time of the local FMM kernels remains constant throughout the entire range from $p = 64$ to $p = 24,576$. The time of all communications seem to roughly follow the upper bound of $\log p$, and the absolute time spent on communication is approximately 1 millisecond per step. Taking into account the topology of the TOFU network, the communication at the coarse level of the global tree will have a latency proportional to

the number of hops $\mathcal{O}(\sqrt[3]{p})$. Therefore, the communication should ultimately scale as $\mathcal{O}(\sqrt[3]{p})$ instead of $\mathcal{O}(\log p)$ on the TOFU's 3-D torus network[5]. Extrapolating these results to the full node of the K computer suggests that a 540 million atom simulation is possible in a few milliseconds per step, although we did not have enough run time on the full node to conduct such a run. In terms of time-to-solution of large MD simulations, we were able to calculate 210 million atoms in 14 milliseconds per step on 4,096 nodes of the K computer. Total calculation times were 43.42 ms/step and 47.90 ms/step using 64 nodes and 24,576 nodes. Parallel efficiency ($T(64)/T(24,576)$) was 0.906.

4.5. Accuracy of Periodic FMM

The periodic fast multipole method approximates the infinite periodic sum by placing a finite number of periodic images around the original domain. There are three sources of error in the periodic FMM:

- The truncation error of the multipole/local expansions in the FMM
- The error from using a finite number of periodic images
- The difference between the Ewald sum and use of periodic images (dipole correction)

The first two are controllable and are a tradeoff between the computational cost. The third source of error can be removed by adding a dipole correction term [30].

[5]Note that the TOFU is not a full 6-D torus, and 3 of the dimensions are smaller than the others, and are mainly designed for fault tolerance.

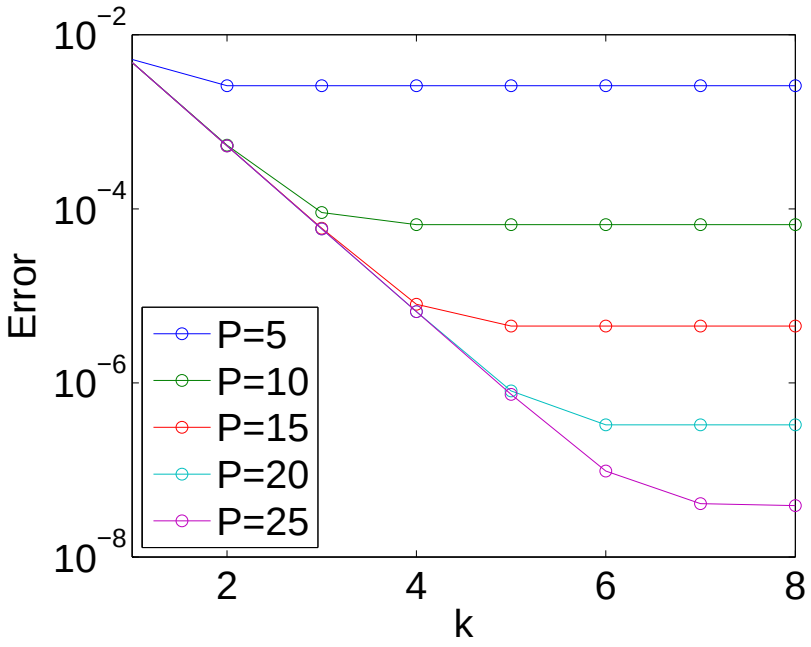


Figure 7: Error of periodic FMM with respect to the order of expansion P and number of periodic images $3^k \times 3^k \times 3^k$. The error is the relative L^2 norm of the difference between the force from the Ewald summation and periodic FMM. With this measure, 10^{-8} indicates that 8 significant digits are matching.

In Figure 7, we compare the results of a Ewald summation code, and periodic FMM code for different number of expansions P and different number of periodic images $3^k \times 3^k \times 3^k$. The number of atoms used in this test was $N = 1,000$ randomly distributed in a unit cube. When the order of expansion is too small, the FMM error dominates, and increasing the number of periodic images will not help. As P increases, it becomes possible to achieve higher accuracy with the use of sufficient number of periodic images. With $P = 15$ and $k = 4$, we are able to match 5 significant digits with the Ewald summation.

5. Macromolecular Crowding Simulations with All Atoms

5.1. Simulation Methods

Simulated Systems: To compare FMM and PME, we performed all-atom classical MD simulations of two different systems: an *in vitro* system and an *in vivo* system. The *in vitro* system consists of target proteins in a solvent, and the *in vivo* system consists of target proteins in a macromolecular crowding environment. The target protein is a putative heavy-metal binding protein TTHA1718 from *Thermus thermophilus* HB8, whose structure was determined by *in vitro* and in-cell NMR [6]. The initial structure of TTHA was taken from PDB entry 2ROE [6], which was solved by the NMR experiment *in vitro*. Ovalbumin, the major protein in egg whites, was chosen as the crowding agent. The structure of ovalbumin was taken from the PDB entry 1OVA [31]. The *in vitro* systems contained 4 TTHA molecules, 372 sodium ions, 364 chloride ions, and 137,977 water molecules in a 160 Å-side cube (vit). The *in vivo* system contained 8 TTHA molecules, 64 ovalbumin molecules, 1,200 potassium ions, 480 chloride ions, and 316,952 water molecules in a 240 Å-side cube (viv). The *in vivo* system mimics the conditions of a macromolecular crowding environment in living cells, where the macromolecules comprise ∼30% of their molecular weight. These systems were equilibrated before the MD simulations, under temperature control (T = 300 K) and pressure control (P = 1 bar), using PMEMD in the AMBER program [32].

MD Simulations: The simulations of viv and vit were performed with periodic boundary conditions using the simulation software described earlier. We adopted the AMBER99SB force field [33] and used the TIP3P rigid water

model [34] as the solvent molecule. The integration time step was 2.0 fs. The bond lengths involving hydrogen atoms were constrained to equilibrium lengths using the RATTLE method. The Coulomb interactions were assessed by applying the PME and FMM methods. PME calculation [15] with a real cut-off of 12 Å and beta-spline interpolation order of 4 was used and FMM calculation with a real cut-off of 12 Å, multipole expansion order of 6, and 27 periodic images in each direction. A smooth cutoff scheme for nonbonded interactions was used for van der Waals interactions with a cutoff distance of 12 Å. The volume was kept constant in each system, and the temperature was maintained at 300 K using the Nosé-Hoover method [35, 36, 37]. The simulation time for analysis was 1 ns for each system. It is short for study of crowding but enough to detect difference of FMM and PME. Conformations of the TTHA molecules were recorded at every 1 ps.

5.2. Simulations of Macromolecular Crowding

All-atom classical MD simulations of target proteins in solution (vit) and target proteins in a crowded environment (viv) were performed to measure the difference of FMM and PME. The root mean square deviations (RMSDs) to the initial structure that taken from the *in vitro* experiment for vit (fmm), vit (pme), viv (fmm) and viv (pme) were 1.001 ± 0.128 Å, 1.036 ± 0.142 Å, 1.310 ± 0.133 Å, and 1.211 ± 0.160 Å, respectively. It show that the difference of averaged structures taken using FMM and PME is smaller than the difference between vit and viv.

Figure 8 shows the root-mean square fluctuation (RMSF) curves of vit (fmm and pme), and viv (fmm and pme). These curves indicated that the difference of the fluctuations between FMM and PME was smaller than the

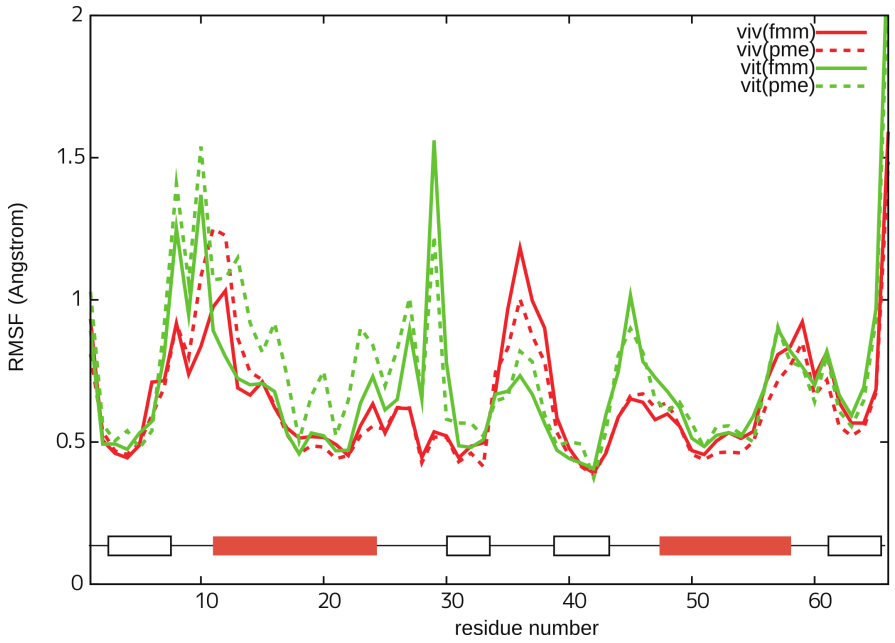


Figure 8: Conformational fluctuations of TTHA for *in vivo* (viv) and *in vitro* systems (vit). The red and green lines indicate the RMSFs of TTHA in viv (fmm: solid, pme: dashed) and vit (fmm: solid, pme: dashed), respectively. The abscissa axis is the residue number of TTHA and the ordinate axis is the RMSF value (in Angstrom). Red and white boxes indicate alpha-helices and beta-sheets, respectively.

difference of the fluctuations between vit and viv.

6. Conclusions and Discussion

We have performed all-atom MD simulations for molecular crowding on the K computer. The performance benchmarks have shown excellent scalability of our classical MD code on up to 79,872 processors, and 638,976 cores. The sustained performance of 4.387 PFLOPS has been achieved using the K computer system with a nominal peak performance of 10.2 PFLOPS for the simulation with 523 million atoms. Good weak scalability, 0.96 and

0.91, were achieved for the simulations using the cutoff technique and FMM. We simulated the protein TTHA in ovalbumins as crowders using millions of atoms. The results obtained from the MD simulation using FMM were in good agreement with those obtained from PME, while the communication was reduced from $\mathcal{O}(p)$ in the PME to $\mathcal{O}(\log p)$ in the FMM for p processes.

Table 7: Large-scale MD simulations

Year	2006	2008	2012
Machine	MDGRAPE-3 [38]	Roadrunner [39]	K computer
N_{atom}	1.7×10^7	4.89×10^{10}	5.43×10^8
Force field	Amber	Lennard-Jones	Amber
Cutoff (Å)	44.5	N/A	28+2
FLOP/step	2×10^{14}	3.74×10^{15}	4.51×10^{14}
PFLOPS	0.185	0.369	4.031
MPI ranks	192	12,240	82,944
Time/step	0.55 s	10.14 s	0.112 s

6.1. Beyond Petascale MD Simulations

It is interesting to see how large-scale MD simulation has developed over time. In Table 7, we summarize several noteworthy high-performance MD simulations from the past five years. In 2006, the MDGRAPE-3 system, which is a special-purpose hardware for MD calculation, achieved performance of 185 TFLOPS using 17 million atoms. Two years later, in 2008, the general-purpose machine, Roadrunner, overtook the FLOPS performance, while 1.4–5.5 million atoms per MPI rank were necessary for the weak scaling of performance (see [39]). On the other hand, our simulations on the

K computer used only 6,542 atoms per MPI rank, with 818 atoms per core. Since a long timescale is imperative for MD simulation studies, the computing time per step should remain as small as possible: strong scaling is preferable, especially in MD simulations of biosystems. A special-purpose computer system Anton has succeeded in a millisecond simulation of small proteins by realizing such strong scaling [40]. As shown in Table 5, we achieved performance gain up to 24,676 nodes for 20 million atoms, which means 800 atoms per node, with 100 atoms per core. Conversely, better performance with a larger number of atoms is possible as far as the memory allows.

6.2. Future Vision of the Macromolecular Crowding Effect

Large-scale and all-atom simulations are essential for a deep understanding of the macromolecular crowding effect. In the current study, we obtained a prevision of the macromolecular crowding effect from all-atom simulations in a large-scale system containing about 1.35 million atoms. However, the simulation lengths were insufficient for equilibrium of the systems. Longer time simulations can provide clear information on the conformation and dynamics of all macromolecules in a crowded environment. Innovations in the field of high-performance computing would lead to an understanding of life phenomena such as immune responses and intracellular signaling, which are topics that attract a lot of medical attention.

Our current MD simulation exhibits $0.2\mu m$ of spatial scale, while a typical biological cell has a size of at least $1\mu m$. Exascale supercomputing will enable cellular scale all-atom simulations, and our simulation is the first step towards molecular simulations of a whole cell.

6.3. Long-range Force Calculation on Exascale Computers

A significant issue to be resolved for parallel MD simulations is the parallelization of the long-range force (i.e., Coulomb interaction) calculation. Since the governing Poisson equation is an elliptical PDE, it is inherently global and requires information to propagate from one end of the domain to the other at every step. The mainstream approach in the MD community at the moment is PME [15] or PPPM [41]. The underlying engine that drives these methods is FFT, which requires high bisection bandwidth. FMM is an interesting alternative since it can reduce the amount of communication on p processes from $\mathcal{O}(\sqrt{p})$ to $\mathcal{O}(\log p)$, as we have shown in our results.

There also seems to be an interesting issue that will arise just before we reach Exascale. The electrostatic field propagates with the speed of light, which travels $0.3\mu m$ within the 1 femtosecond time resolution of MD simulations. The current simulation on a 10 PFLOPS machine already has a scale of $0.2\mu m$, and is at the limit of requiring an all-pairs interaction. This shift from classical MD to relativistic MD will put a cap on the global communication requirements and hence the $\mathcal{O}(\log p)$ communication in the FMM could be reduced to $\mathcal{O}(1)$. We would like to point out that this paradigm shift will be encountered before whole cell simulations ($1\mu m$) become possible.

Acknowledgments

This work was partially supported by the Integrated Simulation of Living Matter Project, commissioned by the Ministry of Education, Culture, Sports, Science and Technology, Japan. This work was partially supported in part by the Japan Science and Technology Agency (JST) Core Research

of Evolutional Science and Technology (CREST) research programs "Highly Productive, High Performance Application Frameworks for Post Petascale Computing". Part of the results is obtained by using the K computer at the RIKEN Advanced Institute for Computational Science (early access and HPCI systems research project:Proposal number hp120068). We thank the Next-Generation Supercomputer R&D Center, especially Kazuo Minami, Akiyoshi Kuroda and Masaaki Terai (in the Application Development Team) for their outstanding support. We appreciate Mikio Hondou and Hikaru Inoue, from Fujitsu Co. Ltd., for their optimization support. We also thank the RIKEN Integrated Cluster of Clusters (RICC) for the computer resources used for our code development and calculations. We gratefully acknowledge the exceptional support provided by the late Takayuki Shigetani at RIKEN Advanced Center for Computing and Communication (ACCC). We also thank Toru Takinaka at NEC Informatec Systems, Ltd. for the supporting code development.

References

[1] G. Rivas, F. Ferrone, J. Herzfeld, Life in a crowded world, EMBO Rep. 5 (2004) 23–27.

[2] N. A. Chebotareva, B. I. Kurganov, N. B. Livanova, Biochemical effects of molecular crowding, Biochemistry 69 (2004) 1239–1251.

[3] J. F. Watson, V. P. Butler, Biologic activity of digoxin-specific antisera, J. Clin. Invest. 51 (1972) 638–648.

[4] N. A. Chebotareva, I. E. Andreeva, V. F. Makeeva, N. B. Livanova, B. I.

Kurganov, Effect of molecular crowding on self-association of phosphorylase kinase and its interaction with phosphorylase b and glycogen, J. Mol. Recognit. 17 (2004) 426–432.

[5] A. P. Minton, Influence of excluded volume upon macromolecular structure and associations in 'crowded' media, Curr. Opin. Biotechnol. 8 (1997) 65–69.

[6] D. Sakakibara, A. Sasaki, T. Ikeya, J. Hamatsu, T. Hanashima, M. Mishima, M. Yoshimasu, N. Hayashi, T. Mikawa, M. Walchli, B. O. Smith, M. Shirakawa, P. Guntert, Y. Ito, Protein structure determination in living cells by in-cell NMR spectroscopy, Nature 458 (2009) 102–105.

[7] K. Inomata, A. Ohno, H. Tochio, S. Isogai, T. Tenno, I. Nakase, T. Takeuchi, S. Futaki, Y. Ito, H. Hiroaki, M. Shirakawa, High-resolution multi-dimensional NMR spectroscopy of proteins in human cells, Nature 458 (2009) 106–109.

[8] S. R. McGuffee, A. H. Elcock, Diffusion, crowding & protein stability in a dynamic molecular model of the bacterial cytoplasm, PLoS Comput. Biol. 6 (2010) e1000694.

[9] D. S. Goodsell, Inside a living cell, Trends Biochem. Sci. 16 (1991) 203–206.

[10] A. P. Minton, Implications of macromolecular crowding for protein assembly, Curr. Opin. Struct. Biol. 10 (2000) 34–39.

[11] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, H. J. Berendsen, GROMACS: fast, flexible, and free, J. Comput. Chem. 26 (2005) 1701–1718.

[12] J. Ryckaert, G. Ciccotti, H. Berendsen, Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes, J. Comput. Phys. 23 (1977) 327–341.

[13] H. Andersen, Rattle: A "velocity" version of the shake algorithm for molecular dynamics calculations, J. Comput. Phys. 52 (1983) 24–34.

[14] S. Miyamoto, P. Kollman, Settle: an analytical version of the shake and rattle algorithm for rigid water models, J. Comput. Chem. 13 (1992) 952–962.

[15] T. Darden, D. York, L. Pedersen, Particle mesh ewald: An n log (n) method for ewald sums in large systems, J. Chem. Phys. 98 (1993) 10089–10092.

[16] Y. Shan, J. L. Klepeis, M. P. Eastwood, R. O. Dror, D. E. Shaw, Gaussian split Ewald: A fast Ewald mesh method for molecular simulation, J. Chem. Phys. 122 (2005) 54101.

[17] P. Hünenberger, J. McCammon, Ewald artifacts in computer simulations of ionic solvation and ion–ion interaction: a continuum electrostatics study, J. Chem. Phys. 110 (1999) 1856–1872.

[18] P. Smith, B. Pettitt, Ewald artifacts in liquid state molecular dynamics simulations, J. Chem. Phys. 105 (1996) 4289–4293.

[19] F. Figueirido, G. S. Del Buono, R. M. Levy, On finite-size effects in computer simulations using the ewald potential, J. Chem. Phys. 103 (1995) 6133–6142.

[20] X. Wu, B. R. Brooks, Isotropic periodic sum: a method for the calculation of long-range interactions, J. Chem. Phys. 122 (2005) 44107.

[21] D. Wolf, P. Keblinski, S. Phillpot, J. Eggebrecht, Exact method for the simulation of coulombic systems by spherically truncated, pairwise r^{-1} summation, J. Chem. Phys. 110 (1999) 8254–8282.

[22] I. Fukuda, Y. Yonezawa, H. Nakamura, Molecular dynamics scheme for precise estimation of electrostatic interaction via zero-dipole summation principle, J. Chem. Phys. 134 (2011) 164107.

[23] D. F. Richards, J. N. Glosli, B. Chan, M. R. Dorr, E. W. Draeger, J.-L. Fattebert, W. D. Krauss, T. Spelce, F. H. Streitz, M. P. Surh, J. A. Gunnels, Beyond homogeneous decomposition: scaling long-range forces on massively parallel systems, in: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, ACM, New York, NY, USA, 2009, pp. 60:1–60:12. URL: http://doi.acm.org/10.1145/1654059.1654121. doi:10.1145/1654059.1654121.

[24] C. Lambert, T. Darden, J. Board Jr, A multipole-based algorithm for efficient calculation of forces and potentials in macroscopic periodic assemblies of particles, J. Comput. Phys 126 (1996) 274–285.

[25] H. Cheng, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm in three dimensions, J. Comput. Phys. 155 (1999) 468–498.

[26] R. Yokota, L. Barba, Hierarchical n-body simulations with autotuning for heterogenous systems, Comput. Sci. Eng. 14 (2012) 30–39.

[27] A. Yonezawa, T. Watanabe, M. Yokokawa, M. Sato, K. Hirao, Advanced Institute for Computational Science (AICS): Japanese national high-performance computing research institute and its 10-petaflops supercomputer "K", in: State of the Practice Reports, SC '11, ACM, 2011, pp. 13:1–13:8. doi:10.1145/2063348.2063366.

[28] G. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in: Proceedings of the April 18-20, 1967, spring joint computer conference, ACM, 1967, pp. 483–485.

[29] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, K. Schulten, Scalable molecular dynamics with NAMD, J. Comput. Chem. 26 (2005) 1781–1802.

[30] C. G. Lambert, J. A. Board, A Multipole-Based Algorithm for Efficient Calculation of Forces and Potentials in Macroscopic Periodic Assemblies of Particles, Technical Report 95-001, Duke University Technical Report, 1995.

[31] P. E. Stein, A. G. Leslie, J. T. Finch, R. W. Carrell, Crystal structure of uncleaved ovalbumin at 1.95 A resolution, J. Mol. Biol. 221 (1991) 941–959.

[32] D. A. Case, T. A. Darden, T. E. Cheatham, C. L. Simmerling, J. Wang, R. E. Duke, R. Luo, R. C. Walker, W. Zhang, K. M. Merz, B. Wang, S. Hayik, A. Roitberg, G. Seabra, I. Kolossváry, K. F. Wong, F. Paesani, J. Vanicek, J. Liu, X. Wu, S. R. Brozell, T. Steinbrecher, H. Gohlke, Q. Cai, X. Ye, J. Wang, M.-J. Hsieh, V. Hornak, G. Cui, D. R. Roe, D. H. Mathews, M. G. Seetin, C. Sagui, V. Babin, T. Luchko, S. Gusarov, A. Kovalenko, P. A. Kollman, B. P. Roberts, Amber 11, University of California, San Francisco, 2010. URL: `http://ambermd.org/#Amber11`.

[33] V. Hornak, R. Abel, A. Okur, B. Strockbine, A. Roitberg, C. Simmerling, Comparison of multiple Amber force fields and development of improved protein backbone parameters, Proteins 65 (2006) 712–725.

[34] W. Jorgensen, J. Chandrasekhar, J. Madura, R. Impey, M. Klein, et al., Comparison of simple potential functions for simulating liquid water, J. Chem. Phys. 79 (1983) 926–935.

[35] S. Nosé, A unified formulation of the constant temperature molecular dynamics methods, J. Chem. Phys. 81 (1984) 511–519.

[36] W. Hoover, Canonical dynamics: equilibrium phase-space distributions, Phys. Rev. A 31 (1985) 1695–1697.

[37] G. Martyna, M. Tuckerman, D. Tobias, M. Klein, Explicit reversible integrators for extended systems dynamics, Mol. Phys. 87 (1996) 1117–1157.

[38] T. Narumi, Y. Ohno, N. Okimoto, T. Koishi, A. Suenaga, N. Futatsugi, R. Yanai, R. Himeno, S. Fujikawa, M. Taiji, et al., A 55 TFLOPS

simulation of amyloid-forming peptides from yeast prion Sup35 with the special-purpose computer system MDGRAPE-3, in: Proceedings of the 2006 ACM/IEEE conference on Supercomputing, ACM, 2006, p. 49.

[39] S. Swaminarayan, T. Germann, K. Kadau, G. Fossum, 369 Tflop/s molecular dynamics simulations on the roadrunner general-purpose heterogeneous supercomputer, in: High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for, IEEE, 2008, pp. 1–10.

[40] D. Shaw, R. Dror, J. Salmon, J. Grossman, K. Mackenzie, J. Bank, C. Young, M. Deneroff, B. Batson, K. Bowers, et al., Millisecond-scale molecular dynamics simulations on anton, in: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, ACM, 2009, p. 39.

[41] R. Hockney, J. Eastwood, Computer simulation using particles, Institute of Physics, 1992.

[42] I. Lashuk, A. Chandramowlishwaran, H. Langston, T.-A. Nguyen, R. Sampath, A. Shringarpure, R. Vuduc, L. Ying, D. Zorin, G. Biros, A massively parallel adaptive fast multipole method on heterogeneous architectures, Communications of the ACM 55 (2012) 101–109.

[43] N. Chau, A. Kawai, T. Ebisuzaki, Acceleration of Fast Multipole Method Using Special-Purpose Computer GRAPE, Int. J. High Perform. Comput. App. 22 (2008) 194–205.

[44] N. Chau, Parallelization of the Fast Multipole Method for Molecular Dynamics Simulations on Multicore Computers, in: Proceedings of the ICCSAMA, 2013, p. 209–224.

[45] J. Lupo, Z. Wang, A. McKenney, R. Pachter, W. Mattson, A Large Scale Molecular Dynamics Simulation Code Using the Fast Multipole Algorithm (FMD): Performance and Application, J. Mol. Graph. Model. 21 (2002) 89–99.

[46] K. Lorenzen, M. Shworer, P. Troster, S. Mates, P. Tavan, Optimizing the Accuracy and Efficiency of Fast Hierarchical Multipole Expansions for MD Simulations, J. Chem. Theory Comput., 8, (2012), 3628–3636.