# Molecular Dynamics Simulation of a Biomolecule with High Speed, Low Power and Accuracy Using GPU-Accelerated TSUBAME2.0 Supercomputer

Shiqiao Du*, Takuro Udagawa*, Toshio Endo* and Masakazu Sekijima*
* Tokyo Institute of Technology, Tokyo, Japan
E-mail: sekijima@gsic.titech.ac.jp Tel/Fax: +81-03-5734-3325

***Abstract*—This paper describes the usability of GPU-accelerated molecular dynamics (MD) simulations for studying biomolecules from the viewpoints of speed, power consumption, and accuracy. The results of simulations showed that GPUs were considerably faster and more energy-efficient than CPUs and their accuracy was high enough. The results computed by an 8 nodes, 16 GPU system were comparable to those of an 8 nodes, 96 CPU cores system; the GPUs produced an acceleration of 10 times and an energy reduction of 75%. These results are encouraging enough for us to use GPUs for MD simulations.**

## I. Introduction

Molecular dynamics (MD) simulation is a powerful tool for simulating the motion of molecules and obtaining detailed information on interesting systems and phenomena. During a simulation, one iteratively integrates numerically the equation of motion of the whole system and the time series of the positions and momentum of every molecule or atom, to produce what is called the *trajectory*. MD is widely used in computational biology, especially for studying the motion of proteins[1].

A critical problem of using MD as a tool for studying protein motion is that it is computationally demanding. One reason is that a solvated protein system usually contains $10^{4-5}$ atoms, which means evaluations of the potential energy or the force are exhausting tasks; the evaluation scales with $O(N^2)$, where $N$ is the number of atoms. Another reason is that as the size of the protein to be simulated increases, we need a much longer simulation (1 $\mu$s - 1 ms) to obtain a biologically meaningful result.

To enhance the performance of MD and enable longer simulations for larger systems, *Graphics Processing Units* (GPUs) have begun to be used. GPUs, which were originally developed for graphical calculations, have a lot of processors that work in SIMD style and have very high parallel performance. For example, one of the latest GPUs, the NVIDIA Tesla M2090, has 512 CUDA cores, which execute such as integer and single-precision floating-point calculations. It has a peak performance of 1331 GFLOPS at single precision. GPUs are also expected to be power-efficient devices because their processors have a simpler structure compared with CPU processors and they require less energy than CPU processors.

However, because GPUs have a simple structure, they are good at simple calculations but poor at complicated ones. Therefore, it is uncertain how much improvement can be obtained from GPUs for a particular application. Another issue is whether the results of a GPU-accelerated calculation are consistent with those of a conventional CPU calculation however efficient the GPU acceleration is.

In this paper, we report on the usefulness of GPU-accelerated MD simulations from three viewpoints. One is the speed of computation, another is the energy consumption, and the other is the accuracy.

## II. TSUBAME2.0

### *TSUBAME2.0 Architecture*

The TSUBAME2.0 supercomputer began operation at Tokyo Institute of Technology in November 2010. It has a peak performance of 2.4 PFLOPS and a storage capacity of 7.1 PBytes. The system mainly consists of 1,408 Hewlett-Packard Proliant SL390s G7 compute nodes equipped with CPUs and GPUs to achieve an excellent power-performance ratio. The nodes and storage systems are connected by a full-bisection fat-tree QDR InfiniBand interconnect.

Fig. 1 shows the node architecture. Each node has two Intel hexa-core Xeon X5670 CPUs, and twelve cores share a 54 GByte memory. Moreover a node is equipped with three NVIDIA Tesla M2050 GPUs. Each GPU has 14 streaming multiprocessors (SMs), and each SM consists of 32 CUDA cores that work in SIMD style. The peak performance of a single GPU is 515 GFLOPS at double precision and 1.03 TFLOPS at single precision, while those of a Xeon CPU are 70.4G FLOPS and 140.8 GFLOPS, respectively. Besides its higher computing performance, the M2050 GPU has a broad memory bandwidth that plays an important role in ensuring better application performance; the memory bandwidth of 150 GB/s is about five times larger than that of a Xeon CPU. On the other hand, the GPU memory, whose size is 3 GB, is separated and smaller than the CPU memory. Thus, before conducting a computation on GPUs, the input data should be copied from the CPU to GPU memory, via the PCI-Express gen2 x16 path.

TSUBAME2.0's excellent power-performance ratio was cited in the Green500 supercomputer ranking (www.green500.org) in November 2010. Its power-performance ratio is 958 MFLOPS/watt, and the system

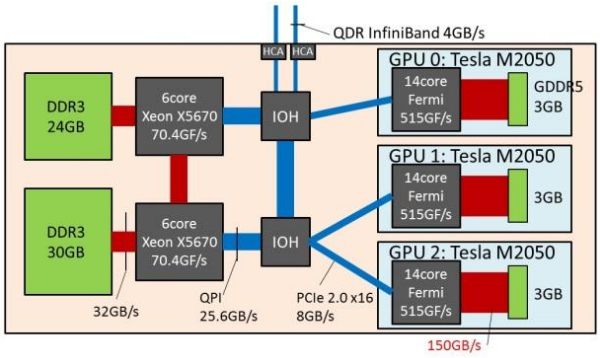


Fig. 1. TSUBAME2.0 node architecture

received "the Greenest Production Supercomputer in the World" award.

*Power Measurement*

We measured the power consumption during execution of the AMBER application by using power monitoring facility of Proliant SL390s nodes. Each node has a management system called HP Integrated Lights-Out (iLO) 3, which provides information including current power consumption in direct current (DC). We made a simple program that periodically checked and recorded the power consumption of nodes and obtained the average consumption during execution of the application. We noticed that the time resolution of the power monitoring facility is about 10 seconds, which is coarser than typical power meters; however, we consider that it is still useful to evaluate the average power consumption, since typical execution times are on the order of hours or even longer.

## III. Molecular Dynamics Simulation

MD simulations are widely used for simulating the motions of molecules in order to gain a deeper understanding of chemical reactions, fluid flow, phase transitions, and other physical phenomena due to molecular interactions. An MD simulation is a numerical solution of the Newton's equation of motion,

$$\mathbf{m}^T \frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}t^2} = -\nabla U(\mathbf{q}) \tag{1}$$

where $\mathbf{m}, \mathbf{q}$ and $U$ are mass vector, position vector and potential energy, respectively. This continuous differential equation is broken down into discrete small time steps, each of which is an iteration of two parts: the force calculation (calculating the forces from the evaluated conformational energies) and the atom update (calculating new coordinates of the molecules).

The most important part of MD simulations has to do with the parameters for calculating the potential energy or forces, namely *force-field*. A number of force-fields have been developed for simulating biomacromolecules such as proteins. *AMBER*[2] and *CHARMM*[3] are representative examples. In those force-fields, the total potential energy is the sum of different energy terms,

$$\begin{aligned} U(\mathbf{q}) = &\sum_{\text{bonds}} k^b(b-b_0)^2 + \sum_{\text{angles}} k^\theta(\theta-\theta_0)^2 \\ &+ \sum_{\text{torsions}} V_n[1+\cos^2(n\phi-\phi_0)] \\ &+ \sum_{i,j\in\text{atoms}} \epsilon_{ij}\left[\left(\frac{r^0_{ij}}{r_{ij}}\right)^{12} - 2\left(\frac{r^0_{ij}}{r_{ij}}\right)^{6}\right] + \sum_{i,j\in\text{atoms}} \frac{q_i q_j}{r_{ij}}. \end{aligned} \tag{2}$$

While the first three terms, i.e., bond, angle, and torsions, explain the interaction energies of bonded atom pairs, the last two, i.e., van der Waals (VDW) and Coulombic electrostatic, are for interactions of non-bonded atom pairs. The parameters of the force constants and equilibrium positions are determined by *ab initio* quantum chemical calculation. Thus, eq. (2) is an approximation to the exact quantum chemical energy. As can be seen from eq (2), most of the computational time is spent evaluating the non-bonded VDW and Coulomb terms, which scale with the second power of the number of atoms $O(N^2)$.

## IV. Speed & Energy Consumption

To evaluate the calculation speed and energy consumption of a GPU-accelerated MD, we ran PMEMD from version 11 of the Amber software suite for a target on ten different specifications and compared their performances. The target was a nucleosome (with 25095 atoms) with the generalized Born (GB) model[4], as shown in Fig. 2. We ran the MD simulation for 10 ps with a 2-fs time step. We checked ten different specifications of cores and nodes: 6 CPU cores, 1 node; 12 CPU cores, 1 node; 24 CPU cores, 2 nodes; 48 CPU cores, 4 nodes; 96 CPU cores, 8 nodes; 1 GPU, 1 node; 2 GPUs, 1 node; 4 GPUs, 2 nodes; 8 GPUs, 4 nodes; and 16 GPUs, 8 nodes. The SPDP precision model, in which individual calculations are performed at single precision within

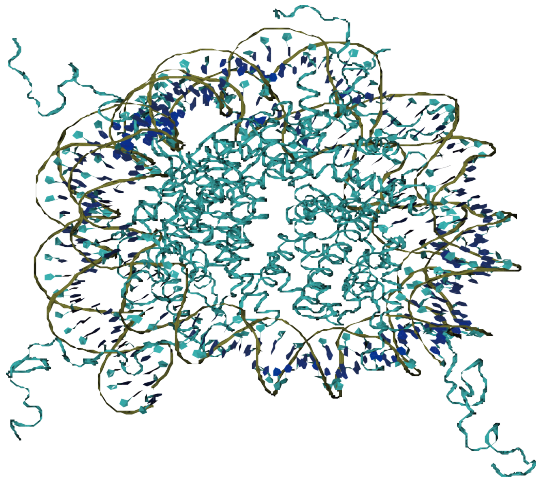

Fig. 2. Structure of the nucleosome

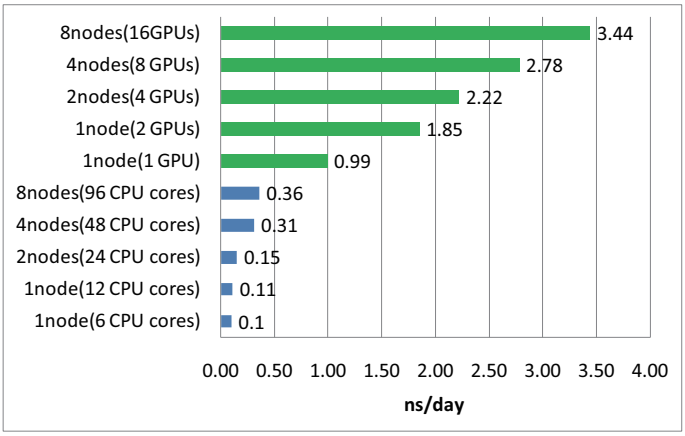


Fig. 3. Comparison of calculation speeds of all specifications in ns/day. Using GPUs had a dramatic impact on accelerating the simulation.

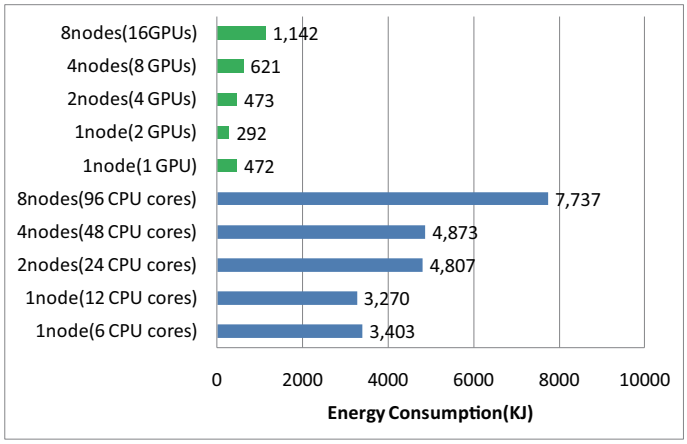


Fig. 4. Comparison of energy consumptions of all specifications in kilojoules. Energy consumption was calculated as the product of the elapsed time and the power consumption. Using GPUs led to a great reduction in energy consumption.

the simulation and all accumulations are done at double precision, was used for the GPU implementation of PMEMD. We measured the elapsed calculation time and power consumption for each specification. Then we compared their performance in terms of ns/day and energy consumption. Computational speed was directly obtained in ns/day units from the PMEMD output files. We used kilojoule units to compare the energy consumptions, which were taken to be the product of the elapsed calculation time and the power consumption. We got the power consumption of each node during the calculation every second. The average power consumption of each node was calculated after the program stopped executing. We summed the averaged power consumptions from all nodes for each specification. These summed values were taken to be the power consumption of the specifications.

Fig. 3 shows the calculation speed of each specification. The results indicate the dramatic impact of GPU acceleration. Comparing the 1 GPU, 1 node specification with the 96 CPU cores, 8 nodes specification, we see that the GPU-accelerated MD calculation was about three times faster. Comparing the specifications in which the same numbers of nodes were used, the GPU version was about ten times faster in all cases.

Fig. 4 shows the energy consumption of all specifications. This result indicates that GPU acceleration resulted in a great reduction in energy consumption. For example, the 8 nodes GPU calculation consumed about seven times less energy than the 8 nodes CPU calculation. We also found that the 1 node, 2 GPU specification was more efficient than the 1 node, 1 GPU specification and that the 1 node, 12 CPU cores specification was more efficient than the 1 node, 6 CPU cores specification. These results lead us to conclude that running more CPUs or GPUs in the same node is power efficient.

## V. Accuracy

We compared the simulated conformational space of two trajectories computed by the CPU and GPU, respectively. The difference between the two is that the trajectory of CPU was generated by a fully double precision calculation, while that of the GPU was partly made up of single precision calculations. Therefore, we can take the CPU result to be more precise ¡ to be more accurate??¿ and see if GPU result is consistent with it. For this purpose, we chose a small protein, *Chignolin*[5], since both the CPU and GPU can simulate it until equilibrium in a feasible time. The native conformation of the Chignolin is shown in Fig. 5. We ran a 1 $\mu$s MD simulation in 315 kelvine by using CPU or GPU systems. The two trajectories were then analyzed. To compare their conformational spaces, from every snapshot structure we extracted torsion angles, which are known as the $(\phi, \psi)$ angles, and carried out principal component analysis (PCA). In PCA, the sample covariance matrix of the $d$-dimensional coordinates is decomposed by its eigenvalues and eigenvectors,

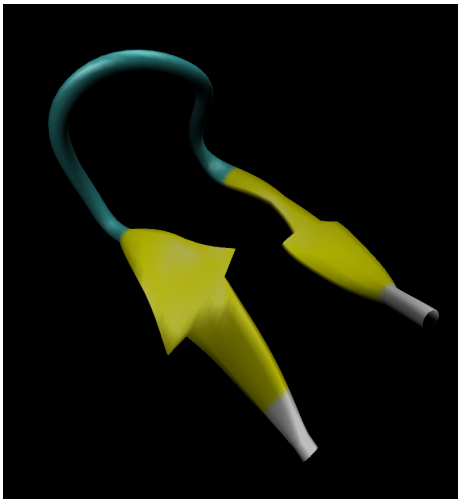

Fig. 5. Native conformation of the Chignolin (1UAO).

$$\mathbf{C} = \frac{1}{N}\sum_{n}^{N} \mathbf{q}_n \mathbf{q}_n^T = \sum_{i}^{d} \lambda_i \mathbf{v}_i \mathbf{v}_i^T, \quad (3)$$

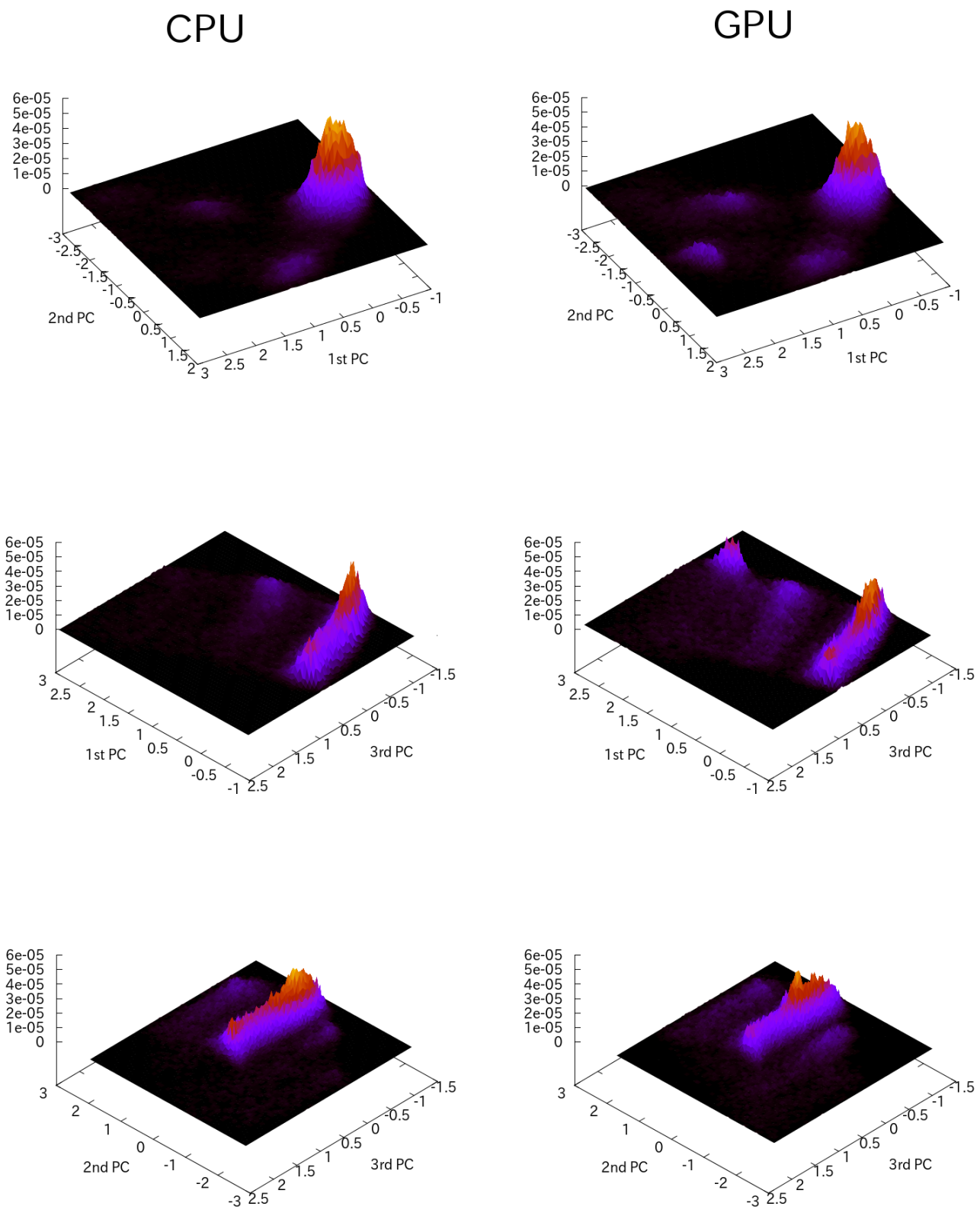


Fig. 6. Comparison of the conformational distributions in reduced principal space.

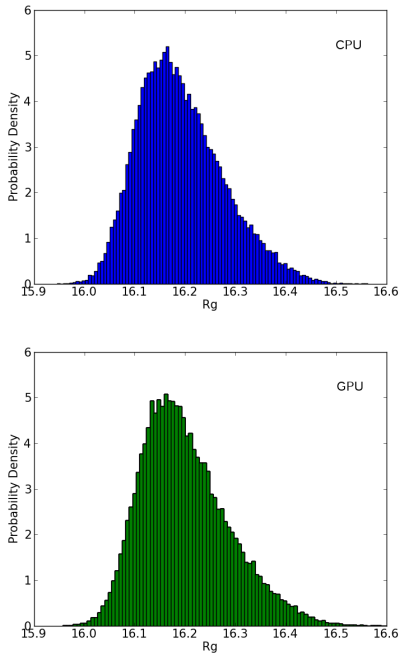


Fig. 7. Comparison of radii of gyration (Rg). The x-axis is the value of Rg in Å units, and the y-axis is the probability distribution. The blue and green histograms correspond to CPU and GPU data, respectively.

where $\mathbf{C}$ is the covariance matrix and $\lambda_i$ and $\mathbf{v}_i$ are the $i$th eigenvalue and eigenvector pair, respectively. The corresponding $i$th principal component of the $n$th observed coordinate is just the projection $\mathbf{v}_i^T \mathbf{q}_n$.

We made the covariance matrix from the CPU trajectory and projected it on the obtained principal axes. Tthe GPU trajectory was also projected by using the same coefficients obtained by the CPU. Fig. 6 compares the conformational distributions in reduced principal component space. We chose the first three principal components and plotted every pair of axes, i.e. (1st-2nd, 1st-3rd, and 2nd-3rd principal components). As the plots show, the conformational spaces generated by the CPU and GPU are almost equivalent. The peaks in the conformational distribution were found to be at the same location in each case. Note that we started the MD simulation with the same random number seed and thus the differences between CPU and GPU were due to whether single or double precision was used.

We also plotted the time series of the radius of gyration (Rg) for the two cases in Fig. 7. Rg is a characteristic property for polymer chains such as proteins. It can be calculated as

$$\mathrm{Rg} = \sqrt{\langle (\mathbf{q} - \langle \mathbf{q} \rangle)^2 \rangle} \tag{4}$$

where the bracket means the sample average and $\mathbf{q}$ is the coordinate vector. The CPU and GPU data had equivalent distributions of Rg, implying that the GPU data were essentially equivalent to those of the CPU calculation.

## VI. Conclusions

In this study, we performed molecular dynamics simulation using GPU on TSUBAME2.0 supercomputer. Then the performance and result was compared to those of conventional CPU molecular dynamics regarding calculation speed, energy consumption and accuracy. Our experiment showed that using GPUs could yield a great improvement in both the calculation speed and energy consumption. Comparing 8 nodes 16 GPUs result and 8 nodes 96 CPU cores result, we could achieve about 10 times acceleration and 75 % energy reduction. And the results of GPU molecular dynamics was consistent with that of CPU molecular dynamics within acceptable range.

## Acknowledgment

We would like to thank Akira Nukada for his help with the power measurement. This work was supported in part by a JST CREST research program entitled "Ultra-Low-Power HPC".

## References

[1] S.A. Adcock and J.A. McCammon. Molecular dynamics: survey of methods for simulating the activity of proteins. *Chem Rev*, 106(5):1589–1615, 2006.

[2] D.A. Case, T.E. Cheatham III, T. Darden, H. Gohlke, R. Luo, K.M. Merz Jr, A. Onufriev, C. Simmerling, B. Wang, and R.J. Woods. The Amber biomolecular simulation programs. *J Comput Chem*, 26(16):1668–1688, 2005.

[3] B. R. Brooks, C. L. Brooks III, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus. CHARMM: the biomolecular simulation program. *J Comput Chem*, 30(10):1545, 2009.

[4] P.S. Shenkin, F.P. Hollinger, and W.C. Still. The GB/SA continuum model for solvation. A fast analytical method for the calculation of approximate Born radii. *J Phys Chem A*, 101(16):3005–3014, 1997.

[5] S. Honda, K. Yamasaki, Y. Sawada, and H. Morii. 10 residue folded peptide designed by segment statistics. *Structure*, 12(8):1507–1518, 2004.