社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－MPS－65 (10)
2007／6／25

Optimization of Molecular Dynamics Simulation on Cell Processor

Manami Sasaki* Masakazu Sekijima† Masami Takata* Kazuki Joe*
manami41@ics.nara-wu.ac.jp
* Graduate School of Humanities and Sciences Nara Women's University
† Computational Biology Research Center
National Institute of Advanced Industrial Science and Technology

Abstract

Molecular dynamics simulations to analyze biological molecular functions are expected to clarify the principle of fluctuation of a protein that participates in a disease, and to discover peptides that are bound to the protein. Such simulations require large-scale and complicated calculation as typical scientific simulations. Furthermore, the molecular dynamic simulation is based on the calculation precision by femto second. In this paper, we report that a molecular dynamic simulator is implemented on the Cell Broadband Engine in PLAYSTATION3 to reduce the calculation time.

Cellプロセッサへの分子動力学シミュレーションの最適化

佐々木 愛美* 関嶋 政和† 高田 雅美* 城 和貴*
* 奈良女子大学大学院 人間文化研究科
† 産業技術総合研究所 生命情報工学研究センター

概要

生体分子の機能解析に利用される分子動力学シミュレーションは，疾病に関与するタンパク質の揺らぎの原理を明らかにし，そのタンパク質と結合するペプチドを発見することを期待されている．このようなシミュレーションは，典型的な科学シミュレーションのように大規模で複雑な計算を必要とする．さらに，分子動力学シミュレーションは，フェムト秒単位での正確な計算に基づいている．本論文では，計算時間を短縮するために，分子動力学シミュレーションを PLAYSTATION3 の Cell Broadband Engine 上で実行し，報告する．

1 Introduction

In the bioinformatics research field, many simulation programs have been developed to analyze biomolecular functions. One of them is molecular dynamics (MD). MD simulation carries out the simulation of the motion of molecules. In an MD program, interactions are calculated by integral Newton's equations of motion of atoms. It is thought that this simulation will be able to clarify the principle of fluctuation of proteins that participate in disease and to discover peptide which is bound to the proteins. MD programs treat large-scale complex systems like biomolecules (e.g., proteins) surrounded by solvent water molecules. In addition, it is required that MD programs perform calculation precisely by femto seconds. Therefore, it must perform an enormous calculation. MD programs sometimes require huge computer resource comparable to a supercomputer with TFLOPS computational power like Blue Gene/L or Earth Simulator. It is, however, very difficult for researchers except some research organizations to establish, occupy, and use such a huge computer system. Recently, Cell Broadband Engine (Cell processor) was developed by IBM, SONY, and TOSHIBA. The Cell processor combines a general-purpose Power Architecture core and several vector processing cores. Theoretical performance of the Cell processor in single-precision floating-point operations is over 200GFLOPS. In this paper, we implement an MD program on the Cell processor and evaluate its performance. PLAYSTATION3 is used to execute the optimized MD program.

In section 2, we briefly describe MD. We explain about Cell Broadband Engine in section 3. In section 4, the implementation method of an MD program to the Cell processor is explained. We describe the result of experiments in section 5.

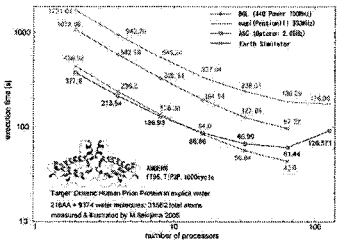

図 1: Comparison of the AMBER execution time

2 Molecular dynamics

MD simulations are widely used for simulating the motion of molecules in order to gain deeper understanding of chemical reactions, fluid flow, phase transitions, and other physical phenomena due to molecular interactions. In such simulations, continuous process is broken down into discrete small time-steps, which repeat iterative two operations: force calculation (calculating the forces from the evaluated conformational energies) and atom update (calculating new coordinates of the molecules) [1]. Needless to say, the computation time is higly affected by the time-step size Δt.

Fig.1 shows the comparison of execution times of four high performance computer systems. This evaluation is measured by an MD program called the sander of AMBER package on dimeric human prion protein consisted of 216 amino acid residues and 9,374 water molecules (31,562 atoms in total). We measured the performance by Blue Gene/L, magi (Pentium III 933 MHz cluster), ASC (Opteron 2GHz cluster) and ES (Earth Simulator) [2].

We describe the MD program flow implemented in this study.

1. Load initial coordinates and velocities from an input file.
2. Judge whether the i-th molecule interact with the j-th one.
3. If it interacts, calculate their intermolecular forces. Otherwise, add 1 to j and go to step 2.
4. Get acceleration of the i-th molecule.
5. Calculate new coordinates and velocities of i-th molecule, add 1 to i and go to step 2.
6. Update the coordinates.
7. Increase a time step and go to step 2.

.. The 2-7 steps are iteratively performed for given times.

We use a recurrence formula obtained by solving an equation of motion using the Verlet method for the calculation of intermolecular forces.

We use the Lennard-Jones's potential for function $\varphi(r_{ij})$ that describes intermolecular potential.

$$\varphi(r_{ij}) = \sum_{ij} 4\varepsilon \left\{ \left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6} \right\}$$

We use a periodic boundary condition where certain fundamental space is provided. Since the boundary is not a physical wall, molecules can move over the boundary. Reduction of the number of molecules and burning off energy do not occur by calculating the whole system because of the fundamental space continuity.

3 Cell Broadband Engine^{TM}

Cell Broadband Engine is the next-generation microprocessor developed by three companies, IBM, Sony, and Toshiba, and is a heterogeneous multi-core processor which has two kinds of cores. One is a general-purpose processor core called "Power Processor Element" (PPE) based on the 64bit Power Architecture. The other is a 128bit SIMD type RISC processor called "Synergistic Processor Element" (SPE). The Cell processor is equipped with cores of one PPE and eight SPEs. The theoretical performance of a single precision floating-point arithmetic operation on an SPE is 192GFLOPS in 3.0GHz [3].

4 Implementation method

On a Cell Processor, when a program is simply compiled to be executed, the program is executed only on a PPE which is a main processor core. In other words, we can not take advantage of the computational ability by several SPEs. In order to elicit high performance computing ability of the Cell processor, parallelization for SPEs is indispensable. A user must create an SPE program to make use of SPE cores by hand, since any compiler does not generate parallelized programs automatically, so far. Consequently, it is necessary to divide an existing program into a PPE program and an SPE program. The program flow and initialization of variables are performed in a PPE program. All parts of parallelizable processes in an MD program should be performed in SPE programs.

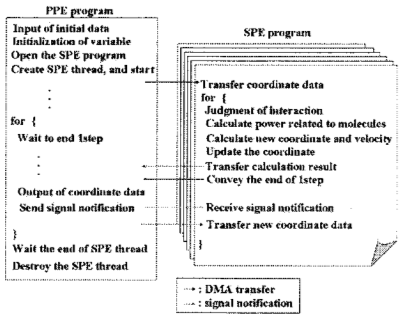

図 2: The flow of Cell program

A library called libspe is used in order to handle SPE cores in the Cell programming. There are series libspe1 and libspe2 in libspe. In this paper, we use libspe1. The basic flow using SPEs with libspe1 is described below.

1. spe_open_image()
2. spe_create_thread()
3. spe_wait()
4. spe_close_image()

An SPE thread created from spe_create_thread always leads to creation of a PPE thread (pthread). Therefore, the creation API of an SPE thread (spe_create_thread) finishes immediately and the PPE thread can perform other computation continuously. Consequently, if spe_create_thread is called multiple times, the same number of SPEs can be used [4][5].

Fig.2 shows the program flow of an MD for the Cell processors. These operations are iterated through the simulation. Atomic data must be divided so that the capacity of LS may not be exceeded. In order to reduce the number of DMA transfer as few as possible, the notice of a signal to SPE from PPE is used for transmission of the synchronization flag [6].

5 Experiment

An MD program is executed by NEV ensemble only on a PPE to measure the execution time for the parallelization on SPEs. Measurement is performed using the times routine in the standard library. Generally, biomolecular simulation is performed on a system with 1,000 to 100,000 atoms. In this paper, the data set to be used is given as the initial coordinates and velocities of 27 atoms because of the capacity limit of LS.

The measuring result of 20,000 time steps is 46.45 seconds. This is the execution time of the whole program on a PPE. Calculation of step 2-3 in the MD program occupies 38.89 seconds among this measurement time. That is, 83.7% of the whole time is spent on the judgment of the existence of interactions, and the calculation of forces which work between molecules. In addition to the occupation of a great portion of the whole execution time, step 2-3 can be executed in parallel. For this reason, parallelizing step 2-3 is the optimal method for SPEs. Moreover, execution time of step 4-7 is 2.46 seconds. It accounts for just a rate of 5.2 % of the whole time. These steps can be also executed in parallel. Therefore, most program sources can be parallelized for SPEs.

The data dependency is investigated for the parallelization for SPEs. The existence of an interaction is judged for every pair of atoms. Therefore, each SPE is required to keep the newest coordinates data of all atoms. When calculating the new coordinates and the velocities of atoms, the present and last times steps of them are required. Therefore, it is necessary to keep the coordinate data of three time steps simultaneously. For the above reason, in order to parallelize them for SPEs, atoms data are required to be divided into each SPE as equally as possible. Furthermore, for every completion of a time step, it must communicate and share the newest coordinate data in each SPE.

Based on the above thing, the existing MD program is ported and parallelized for the Cell processor. The times routine is used for measurement of execution time of the PPE program. Moreover, a counter register called SPU Decrementer is used on the SPE program. Each SPU contains dedicated 32bit decrementers, and the decrement of the value is carried out for every CPU clock. The value of SPU Decrementer is read before and after the execution that the user wants to measure. It is possible to measure execution time from the difference of two read values [7].

First, the execution time of the whole program when performing on a PPE or an SPE is measured using the times routine on the PPE program. A measuring result is shown in Tab.1.

In Tab.1, both of PPE and one SPE execution are sequential processing. The theoretical performance in the single precision floating point arithmetic operations of PPE is lower than one SPE. It is about 20 GFLOPS. However the execution time on one SPE is about 3 times slower than on PPE. Unlike execution on PPE, the execution on one SPE requires to cre-

表 1: Comparison of the execution time in a PPE program

	user time (sec)	system time (sec)
PPE	46.45	0.24
1 SPE	117.84	0.47

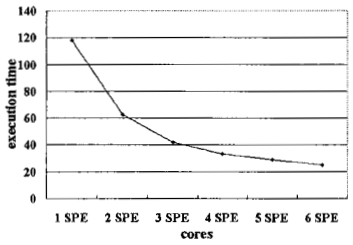

図 3: Comparison of the execution time in a SPE program

ate a SPE thread, transmission to LS of the program and data, etc. Namely, the result contains considerable overhead.

Next, the parallel execution time is measured on the SPE program. The measuring result is shown in Fig.3. Execution time on two or more SPEs shows the average of execution time in each SPE.

In Fig.3, the execution time on three SPEs is about 40 seconds. From this result, it is thought that the difference of the execution time in Tab.1 is removable by dividing into three or more data. In Fig.4, the parallelization efficiency on three SPEs is the maximum. This is because each load to SPEs gets balanced since the number of atomic data used in this experiment is 27 pieces.

6 Conclusions

In this paper, we accelerated execution time of an MD program for the Cell processor. When the execution times of the sequential execution on PPE and one SPE were compared, the execution time on one SPE which requires SPE thread generation etc. is about three times of the execution time on PPE. And, every parallelization efficiency in comparison with the execution time on one SPE brought a result exceeding 0.8.

The data set used for the experiment was very small (27 atoms). Therefore, SIMD commands did not work in high efficiency. However, when treating

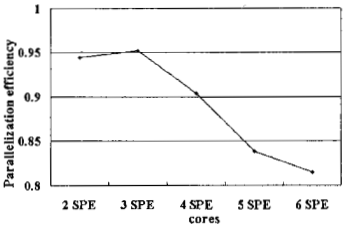

図 4: Parallelization efficiency

larger-scale data sets, such as a biomolecular simulation, the SIMD commands will be effective and expect great reduction of execution time. Moreover, it is thought possible to further accelerate execution by using general optimization techniques, such as concealment of the memory access latency by DMA double buffering and loop unrolling.

参考文献

[1] M. Sekijima, et al.: "Automatic Improvement of Scheduling Policies in Parsley Parallel Programming Environment", 14th IASTED PDCS2002, pp.380-385 (2002).

[2] M.Sekijima: www.cbrc.jp/~sekijima/

[3] Cell Broadband Engine (CBE) Processor Tutorial: www.cs.unc.edu/~geom/EDGE/SLIDES/perrone.ppt#256,1,Cell

[4] SPE Runtime Management Library: www-306.ibm.com/chips/techlib/techlib.nsf/techdocs/771EC60D862C5857872571A8006A206B/$file/libspe_v1.2.pdf

[5] Cell Broadband Engine Programming Tutorial: www-306.ibm.com/chips/techlib/techlib.nsf/stechdocs/FC857AE550F7EB83872571A80061F788/$file/CBE_Tutorial_v2.1_1March2007.pdf

[6] C/C++ Language Extensions for Cell Broadband Engine Architecture: cell.scei.co.jp/pdf/Language_Extensions_for_CBEA_v23.pdf

[7] Cell Broadband Engine Programming Handbook: www-306.ibm.com/chips/techlib/techlib.nsf/techdocs/9F820A5FFA3ECE8C8725716A0062585F/$file/CBE_Handbook_v1.1_24APR2007_pub.pdf