Development of Molecular Dynamics Simulation System for Large-Scale Supra-Biomolecules, PABIOS (PArallel BIOmolecular Simulator)

Project Representative
Hisashi Ishida Japan Atomic Energy Research Institute

Authors
Hisashi Ishida [*1], Mariko Higuchi [*1], Yoshiteru Yonetani [*1], Takuma Kano [*1], Yasumasa Joti [*2], Akio Kitao [*2] and Nobuhiro Go [*1]

[*1] Japan Atomic Energy Research Institute
[*2] University of Tokyo

The Earth Simulator has the highest power ever achieved to perform molecular dynamics simulation of large-scale supra-molecular systems. Now, we are developing a molecular dynamics simulation system, called PABIOS, which is designed to run a system composed of more than a million particles efficiently on parallel computers. To perform large-scale simulations rapidly and accurately, state-of-the-art algorithms, such as Particle-Particle Particle-Mesh (PPPM) which can compute the long-range Coulomb interaction accurately and efficiently, are implemented. PABIOS uses a domain decomposition method which can achieve high parallelization. To optimize the load balance among the processors, the dynamic load-balancing algorithm was developed, and the assignment of subcells in the decomposition method for parallel computations to processors was automatically adjusted during the molecular dynamics simulation. Moreover, to make up for the insufficient parallel efficiency ratio of the PM calculation of PPPM, we have limited the number of processors which deal with the PM calculation and assigned the rest of the processors to other calculations. In order to improve the performance of PABIOS on the Earth Simulator, the algorithm to calculate short-range interactions (PP interactions) was intensively vectorized. A benchmark test was carried out using the system of a RuvA-Holliday junction DNA complex which consisted of 166,177 atoms. At present, PABIOS has achieved a parallelization efficiency ratio of 55.0%, and a vectorization ratio of 97.5% even when 15 nodes (120 processors) are used. As a result, the speed of calculation is about twice as fast as that of last fiscal year.

Keywords: large-scale supra-biomolecular MD simulation, dynamic load balance, PPPM method, Holliday junction

1. Introduction

Molecular dynamics (MD) simulation not only provides dynamic descriptions of molecules on the atomic scale, but also provides valuable information for the interpretation of experimental data. The rapid development of computer power and the elucidation of the structures of biological macromolecules by X-ray crystallography and other experiments have increased the need for large-scale MD simulations in the field of biology.

One way to achieve large-scale simulations is to utilize parallel supercomputers such as the Earth Simulator.

The two main strategies that have been used in the parallelization of MD programs are called particle decomposition (PD) and domain decomposition (DD). In the PD method all processors know all the coordinates of all the particles in the system. MD programs such as AMBER [1], CHARMM [2], GROMOS [3] which were originally designed for serial computers, have been adapted for use on parallel computers by using this method. Adapting these programs in this way is straightforward from the point of view of coding; however, this method requires extensive communication between the processors because it collects the data for all the coordinates from all the processors and distributes the particles' new coordinates to all the processors. In the DD method, which we have employed, the volume of the physical system is divided into rectangular subcells with a length longer than the potential cutoff radius. The main advantage of this method is that each processor communicates with a limited number of neighboring processors. The DD method, therefore, offers the best theoretical scalability of both memory use and the amount of communication between the processors used for MD simulations of large molecular systems on parallel computers.

We are developing an integrated molecular simulation system for biological macromolecules, called PABIOS (PArallel BIOmolecular Simulator) which is designed to run a system composed of more than a million particles efficiently on parallel computers.

PABIOS has several special features:

1. Topology of biomolecules

   The structure of PABIOS' program code is optimized for a system in which the topology of a biomolecular structure is considered.

2. A variety of force field parameters

   At present PABIOS can use both the AMBER force field and the CHARMM force field. The parameter file for the topology of biomolecules can be obtained from the PDB file by using the parameter module in PABIOS.

3. A variety of simulation methods

   A variety of simulation methods, such as energy minimization, molecular dynamics, free energy perturbation, normal mode analysis, principal component analysis and so on, are included.

4. Algorithm for non-cutoff electrostatic interactions

   PABIOS utilizes the Particle-Particle Particle-Mesh (PPPM) algorithm, which effectively calculates all the Coulomb electrostatic interactions [4]. This algorithm reduces the computational time required to calculate the electrostatic forces from conventional $O(N^2)$ to $O(N\log N)$.

5. Input and output compatibility

   PABIOS' input and output file formats are currently compatible with those used by AMBER [1] and PRESTO [5] (It will be compatible with CHARMM format in the future). By using the same molecular dynamics output format, users can then utilize the many analysis algorithms provided by PABIOS, AMBER and PRESTO.

6. Portability

   Written in Fortran90, PABIOS is designed to be easy to read, modify and extend. Users can easily maintain the existing code, develop the current algorithms and integrate new ones efficiently.

7. Control files

   The control files for running PABIOS are described in a user-friendly manner.

8. Time-integral algorithms for long time steps

   In this fiscal year, we have implemented SHAKE and RATTLE which allow the time step taken to be larger by fixing the bond lengths and angles in the system.

9. Time-integral algorithms with high accuracy

   In this fiscal year, we have implemented Martina-Klein-Tuckermann (MKT) algorithm which produces the correct ensemble thermodynamically.

10. Reducing the amount of communication

    In the DD method, the processor assigned to a subcell needs to evaluate the interactions between the atoms in the subcell and between the atoms in 26 neighboring subcells. PABIOS employs the method for minimizing communication between processors proposed by D. Brown [6], which enables the number of processors between which data must be transferred to be reduced to only 7 of the neighboring subcells.

11. Dynamic load balance

    In this fiscal year, to overcome the load imbalance associated with irregular atomic distribution, we have implemented a dynamic load-balancing algorithm.

12. High performance

    PABIOS achieves both a high parallelization efficiency ratio and a high vectorization ratio. The techniques employed to achieve such a high performance are explained in detail below.

2. Dynamic load balance

In PABIOS, the initial data of the atomic coordinates are distributed into subcells such that the number of subcells with which each processor deals is as similar as possible. However, in MD simulations of a system which is heterogeneous, when molecules are in vacuum or when molecules such as water traverse the boundary of the subcells frequently, one processor may have a heavier load than the others, and it is this processor which determines the overall parallel performance. This irregular distribution of atoms in the subcells lowers the efficiency of the parallel MD simulation as time elapses. Therefore, in parallel computing, it is very important to optimize the distribution of the load and to minimize the communication among the processors.

In our approach to developing the dynamic load-balancing algorithm, we used an evaluation function which should be minimized when the total running time of the simulation is minimized. Let's suppose N_p is the number of processors for the parallel calculations. We assigned the position of each cell in the system by using a 3-dimensional vector $\boldsymbol{i}$. In each subcell, the value of the weight $w_{\mathbf{i}}$ is given by measuring the calculation time. Let's suppose the value q_i indicates each processor's number (from 0 to N_p-1). By assigning the set of $\{q_i\}$ to the whole system, the workload of the system is determined. In order to optimise the division of space in the system, the evaluation function H $(\{q_i\})$ is minimized

In our case, we adopted the following evaluation function:

$$H\left(\{q_{\mathbf{i}}\}\right)=\sum_{p} h_p^2 \tag{1}$$

where h_p is the sum of the weight of the subcells which processor p deals with.

$$h_p = \sum_{\{\mathbf{i}|q_{\mathbf{i}}=p\}} w_{\mathbf{i}} \tag{2}$$

Then the algorithm of the dynamic load balance implemented in PABIOS is:

1. Select randomly a subcell from $\{q_i\}$.
2. There are six subcells which have direct contact with the six surfaces of the subcell. Then count *n* (from 0 to 6). *n* equals the number of subcells which are processed by processorsother than the processor responsible for the subcell.
   (a) if n=0, then there is no contact area with other processors. Go to 3.
   (b) If n>0, then randomly select one of the *n* , and give it the number of the processor of the subcell with which it is in contact. Next, set the processor number as a candidate *q*' for the variable q_i .
3. Suppose the value of the evaluation function, in which q_i is set to be *q*', is *H*', then evaluate the change of the evaluation function between the past and the present,
   $\Delta H = H' - H$
   (a) if $\Delta H < 0$: adopt the change, $q_i = q'$
   (b) if $\Delta H \geq 0$: reject the change
4. Go back to 2.

3. Division of processors for the calculation of PP part of PPPM and PM part of PPPM

The PPPM method we have implemented has two main calculations, the Particle-Particle (PP) and the Particle-Mesh (PM). The PP interactions, which are regarded as short-range, can be calculated as usual in the DD method. On the other hand, the PM interactions, which are regarded as long-range, are approximated in the form of discrete convolutions on an interpolated three-dimensional mesh of charges. This method requires 3D fast Fourier transform (3D-FFT) of the charged mesh to perform the convolution. Owing to the insufficient parallelization of 3D-FFT and the communication in the PM calculation, the parallelization efficiency ratio of the whole PM calculation decreases as the number of the processors increases. Moreover, there is a limit to the number of processors which can feasibly be used for the PM calculation due to a limitation on the number of the partitions into which the mesh can be divided.

To deal with the problem, we have adjusted the number of processors which deal with the PM calculation and assigned the rest of the processors to other calculations. We divided the processors in the parallel calculation into two groups by using MPI communicator. One group is to be used for the PM calculations and possibly some PP calculations, while the other group is to be used exclusively for PP calculations. This is illustrated in Figure 1.

The algorithm of the dynamic load balance has been slightly modified as follows:

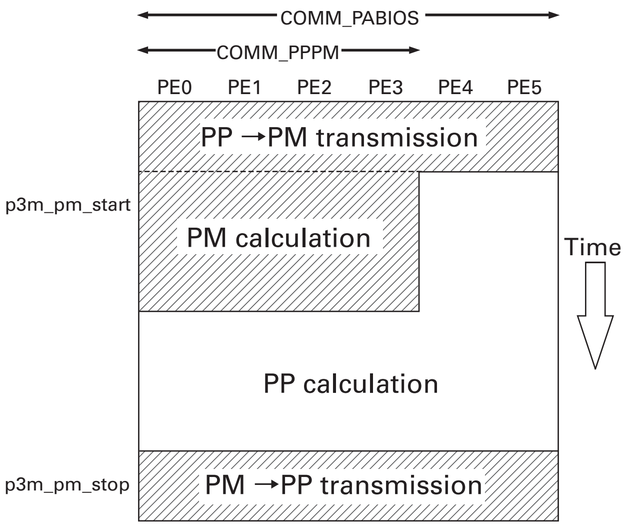

Fig. 1 The number of processors which deal with the PM calculation has been adjusted. COMM_PABIOS: a communicator used throughout the whole molecular dynamics calculation. The number of the processors, NPROCS_PABIOS is obtained from MPI_Comm_size. COMM_PPPM: a communicator used for the PM calculation. The number of processors is determined by the input file. The processor-number is ranked from 0 to NPROCS_PPPM -1 in the COMM_PABIOS. The data transmission from COMM_PABIOS to COMM_PPPM is used for the FFT calculation within the PM calculation and the resultant data from the PM calculation are sent back to COMM_PABIOS. In this Figure, NPROCS_PABIOS=6, NPROCS_PPPM=4.

$$h_p = \left\{ \begin{array}{ll} \sum_{\{i|q_i=p\}} w_i + w_{i,\mathrm{PPPM}} & \text{(for PM calculation)} \\ \sum_{\{i|q_i=p\}} w_i & \text{(otherwise)} \end{array} \right\} \tag{3}$$

Here, $w_{i,\mathrm{PPPM}}$ is the weight of the PM calculations and w_i is the weight of other calculations.

4. Vectorization of PABIOS

We have continued to vectorize the PP calculations because these non-covalent interactions are a time consuming part of an MD simulation. Last fiscal year, we vectorized the subroutine to calculate the PP interactions in the follwing way: (The algorithmic structure of the subroutine is illustrated below.)

```
do i = 1, i_atom
!CDIR NODEP
  do j = i_atom+1, j_atom
    IF (distance between i and j < cutoff) then
      Calculation of force between i and j
    endif
  enddo
enddo
```

Vectorization can be carried out in a straightforward manner by adding a vectorization directive, !CDIR NODEP, at the innermost do-loop when the inntermost do-loop is independent of the outer do-loop.

Actually, in this simple method, the assessment of whether the IF statement is true or not takes time, and the length of the vectorization of the inner do-loop is rather short. In order to reduce the amount of computer time spent rejecting the pairs which are beyond the potential cutoff range, and increase the length of vectorization, we employed a technique in which a list of interaction pairs is constructed by searching for the interactions which are within the range of r_list which is slightly larger than the cutoff. Then the algorithm has the following form:

```
Do all subcells
  Do inter =1, Number of interactions
    i = ilist(inter)
    j = jlist(inter)
        If (distance between i and j < cutoff ) then
            Calculation of force between i and j
        Endif
  End inter loop
Enddo: cell
```

The length of the do-loop is much longer than that of the inner do-loop in the former algorithm. This enables us to achieve high vectorization. During the evaluation of the interactions, only the atomicpairs from the neighbor list are considered. However, vectorization of this do-loop is rather more difficult than that of the double do-loop in the original algorithm because the identical atoms registered in the list can access the CPU memory simultaneously which is not allowed by the vectorization calculation. We have avoided this problem by using a work array. The increase in memory requirement caused by using a work array is not particularly significant.

Due to the diffusion of molecules, the list of interactions should be reconstructed from time to time as the simulation progresses. In our case, however, the construction of the list does not take much time if a merge region of about 1Å from cutoff is used. Then, rebuilding the list every 10 or 20 time-steps is enough.

5. Performance of PABIOS on the Earth Simulator

We have carried out a benchmark test of PABIOS on the Earth Simulator. The physical system for this test was chosen to be RuvA-Holliday junction DNA complex, as shown in Figure 2. The size of the system was 114.5 Å × 114.5 Å × 114.5 Å, and the cutoff length for the van der Waals interactions was chosen to be 9Å. On the basis of the cutoff length the system was divided into 12 × 12 × 12 = 1,728 subcells.

5.1. Dynamic Load balance

After optimizing the load balance over the calculation of 1,000 steps, we performed the MD simulations and measured the running time over 1,000 steps. The effect of the load balance was measured on 10 nodes (80 processors) as shown in Figure 3. The horizontal line indicates the number of processors used for the PM calculation. The time taken to complete the calculation when the load balance was used was 167.5 seconds, a decrease of 9.6 seconds on the previous non-load balance algorithm, which took 177.1 seconds (5.4% improved). Moreover, by restricting the number of

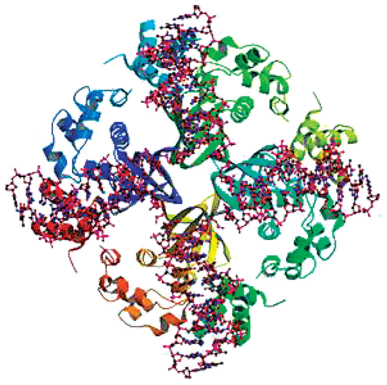

Fig. 2 The system for the benchmark test is RuvA - Holliday junction DNA complex, which is a biomolecular complex consisting of four strands of DNA and four protein molecules which executes recombination of homologous DNA strands. The size of the system and the number of atoms in the system are 114.5 Å × 114.5 Å × 114.5 Å and 166,177 atoms, respectively.

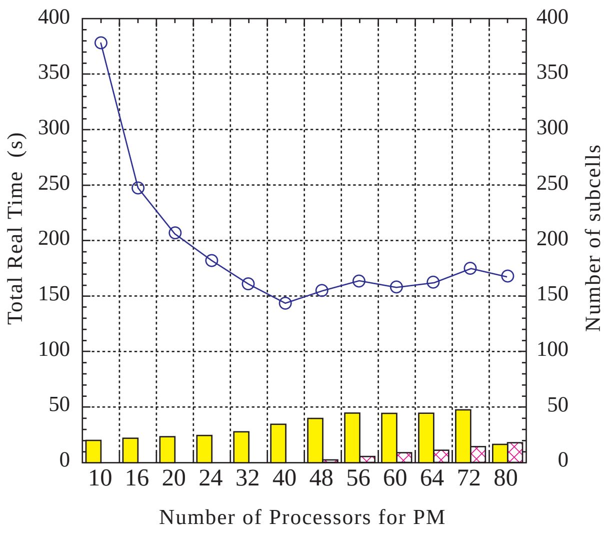

Fig. 3 The blue line represents the total real time (in seconds) of the MD simulation using 80 processors. The number of steps was chosen to be 1,000. The optimized number of processors for PM is 40. The yellow and red bars represent the number of subcells which 0th and 79th processors deal with respectively.

processors used for the PM calculation to 40, a minimum time of 144.5 seconds was achieved (18.4% improved). The number of subcells which 0th and 79th processors deal with is shown in yellow and red respectively. At 40 processors for the PM calculation, the number of subcells which processors for the PM calculation deal with is 0, indicating that the other processors were entirely responsible for calculating the PP calculation. When the number of processors for the PM calculation decreases below 40, the total time for the whole calculation starts to increase. This is because the PP calculation becomes faster and the PM calculation becomes slower; therefore, the data communication between PM and PP halts until the PM calculation has finished.

5.2. Parallelization

A parallelization efficiency ratio of 55.0% was achieved even when 15 nodes (120 processors) were used as shown in Table 1. As the number of processors increased, the parallelization efficiency ratio did not decrease very much as compared with the result presented last fiscal year. It is considered that the load balance is working efficiently.

5.3. Vectorization

The vectorization ratio of PABIOS was measured on 10 nodes (80 processors). Due to the intense vectorization mentioned in the previous section, the vectorization ratio for the calculation of short-range interactions of van der Waals and PP was 99.5%, an increase of 0.3% on the previous algorithm, which achieved a vectorization ratio of 99.2%. This increase in the vectorization ratio increased the calculation speed of the short-range interactions by 2.44 fold as shown in Table 1. The total vectorization ratio was 97.2% when calculated on 10 nodes (80 processors) and maintained more than 97.0% even when 15 nodes (120 processors) were used.

Table 1 The performance of PABIOS on the Eearth Simulator.

The number of total processors	40	80	120
The number of processors for PM	40	40	120
Parallelization efficiency ratio	67.24	64.80	55.00
Vectorization ratio	97.47	97.17	97.46
Computation time (sec/step, H15)	0.4598	0.2738	0.2123
Computation time (sec/step, H16)	0.2248	0.1445	0.1135
Speedup (scale)	2.045	1.895	1.870

6. Conclusion

In order to optimize the calculation performance of PABIOS, we have developed the dynamic load balance method and divided the processors into those used for PM calculations and those used for the other calculations including the PP calculations. Next, we are going to perform a large-scale molecular dynamics simulation of a Holliday junction in order to understand how branch migration occurs.

References

[1] D.A. Pearlman, D.A. Case, J.W. Caldwell, W.R. Ross, T.E. Cheatham, III, S. DeBolt, D. Ferguson, G. Seibel and P. Kollman. AMBER, a computer program for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to elucidate the structures and energies of molecules. *Comp. Phys. Commun.* **91**, 1-41 (1995).

[2] B.R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations, *J. Comp. Chem.* **4**, 187-217 (1983).

[3] W. F. van Gunsteren and H. J. C. Berendsen. GROMOS: GROningen MOlecular Simulation software. Technical report, Laboratory of Physical Chemistry, University of Groningen, Nijenborgh, The Netherlands, (1988).

[4] R. W. Hockney and J. W. Eastwood, Computer Simulation Using Particles. McGraw-Hill, NY, (1981).

[5] K. Morikami, T. Nakai, A. Kidera, M. Saito and H. Nakamura. PRESTO : A vectorized molecular mechanics program for biopolymers. *Comput. Chem.* **16**, 243-248 (1992).

[6] D. Brown, J. H.R. Clarke, M. Okuda and T. Yamazaki. A domain decomposition parallelization strategy for molecular dynamics simulations on distributed memory machines. *Comp. Phys. Commun.* **74** 67-80 (1993).

大規模生体超分子動力学シミュレーションシステムPABIOS（PArallel BIOmolecular Simulator）の開発

プロジェクト責任者
石田　　恒　　日本原子力研究所

著者
石田　　恒*[1]，樋口真理子*[1]，米谷　佳晃*[1]，叶野　琢磨*[1]，
城地　保昌*[2]，北尾　彰朗*[2]，郷　　信広*[1]
＊1　日本原子力研究所
＊2　東京大学

地球シミュレータは従来にはない大規模生体超分子系の分子動力学シミュレーションを可能とする計算能力をもつ。そこで、我々は数百万原子の生体超分子系を扱う大規模な分子動力学シミュレーションシステム（PABIOS）を開発している。PABIOSは長距離相互作用を高速かつ高精度に計算するPPPM計算法、系のエネルギー、温度、圧力を一定に保つ様々な時間積分アルゴリズム、原子結合長を固定することにより時間ステップの増大を可能とするSHAKE, RATTLEアルゴリズムなどの高精度アルゴリズムを採用した計算性能に優れたシミュレーションシステムである。PABIOSは空間分割法を用いており高い並列化効率をもつ。本年度は、系の原子分布の異方性による並列化効率の悪化がおこらないように、動的ロードバランスを開発した。また、PPPM法のプロセッサの増加にともなう並列化効率の悪化がプログラム全体の並列化効率に悪影響をおよぼさないように、PPPM計算とそれ以外の計算を別々のプロセッサに割り当てられるようにした。更には、短距離相互作用の計算アルゴリズムのベクトル化を強化した。その結果、ホリデイ分岐DNAとDNA結合蛋白質RuvA4量体の複合体が水中に存在する系（全系16万6千原子）の分子動力学シミュレーションにおいて地球シミュレータで120個のプロセッサを用いてもPABIOSはベクトル化率95％以上、並列化効率50％以上の優れた性能を達成することに成功した。そして計算速度も昨年度と比べて約2倍程度向上した。

キーワード：大規模生体超分子動力学シミュレーション，動的ロードバランス，PPPM法，ホリデイジャンクション