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

Group Representative

Hisashi Ishida Japan Atomic Energy Research Institute

Authors

Hisashi Ishida [*1] ・ Yasumasa Joti [*1] ・ Mariko Higuchi [*1] ・ Takuma Kano [*1]
Akio Kitao [*2] ・ 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-biomolecular 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 are implemented. The long-range Coulomb interaction is computed by PPPM method which reduces the computational time from conventional $O(N^2)$ to $O(NlogN)$. Moreover, PABIOS is designed to reduce the amount of communication between the processors of parallel computers as much as possible. For the calculation of short-range interactions, PABIOS uses a domain decomposition method which can achieve high parallelization. For the calculation of long-range interactions in the Particle-Mesh algorithm, a slab decomposition method is used to run 3D-FFT efficiently. In order to improve the performance of PABIOS on the Earth Simulator, the algorithms to calculate short-range interactions, and the algorithm to calculate long-range Particle-Mesh interactions were 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 50.5%, and a vectorization ratio of 96.9% even when 18 nodes (144 CPUs) were used.

Keywords: large-scale supra-biomolecular MD simulation, PPPM method, domain decomposition method, Holliday junction

1. Introduction

Molecular dynamics simulation not only provides dynamic descriptions of molecules on the atomic scale, but also adds valuable information for interpreting 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.

Various computer programs, such as AMBER [1], CHARMM [2], GROMOS [3], have been developed to perform biological MD simulations. These programs were originally developed for serial machines; however, the simulation of large molecules requires enormous computing power which is not available on such machines. One way to achieve such large-scale simulations is to utilize parallel supercomputers such as the Earth Simulator.

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 is written in Fortran90 which is an efficient high-level language for high performance computers. Fortran90 has several features, such as dynamic memory allocation, modules, derived data types and so on, which allow us to make the program easier to read, more compact and portable. For communication between processors in a parallel computer system, PABIOS employs an MPI (Message Passing Interface) library which is available on a variety of computer platforms.

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 simulation methods

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

3. Algorithm for non-cutoff electrostatic interactions

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

4. Input and output compatibility

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

5. 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.

6. Control files

The control files for running PABIOS are described in a user-friendly manner. These files are described in terms of 'TASK' and 'SUBTASK'. The file format is as follows:

```
task
        subtask
quit
end
```

'TASK' specifies the main functions such as md, minimization, nma, analysis and so on. 'SUBTASK' describes the 'TASK' in detail by specifying such items as parameters and I/O file names which control the simulation. This makes it easy for users to understand what they are calculating.

7. 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. Parallelization of PABIOS

2.1. Domain decomposition method

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 coordinates of all 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 processors because it collects data for all coordinates from all processors and distributes particles' new coordinates to all processors.

In the DD method, which we employed, the volume of the physical system is divided into rectangular subcells with a length longer than the potential cutoff radius. These subcells are then allocated to processors in such a way that neighboring subcells are located in the same processors. 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 memory use and the amount of communication between processors for MD simulations of large molecular systems on parallel computers.

2.2. Reducing amount of communication

To increase the efficiency of a program's performance, it is essential to reduce the quantity of data transferred between processors as much as possible.

A schematic 2-D representation of the communication pattern is shown in Fig. 1. The processor assigned to the red subcell needs to evaluate the interactions between the atoms in the red subcell and between the atoms in 26 (8 in 2-D) neighboring subcells. The whole computation is carried out by computing the interactions between the atoms in the same subcell and between the atoms in neighboring subcells.

By taking Newton's third law into account, the number of neighboring processors between which data must be transferred can be halved. Each processor retrieves the atomic coordinates from the neighboring subcells, evaluates the interactions and then sends the results back to the neighboring subcells. With this modification, the number of subcells between which data must be transferred can be reduced form 26 to 13 (4 in 2-D).

Moreover, PABIOS employs the method for minimizing communication between processors proposed by D. Brown [6]. Suppose that the red subcell retrieves atomic coordinates from the three blue subcells only. In this case, the data which the red processor receives is sufficient to calculate not only the interactions between the atoms in the red subcell and the three blue subcells (which we call direct interaction) but also the interaction between the atoms in the two shaded blue subcells (which we call indirect interaction). The calculation of the indirect interactions can then be substituted for that of the interactions between the atoms in the red subcell and the atoms in the gray subcell. Thus, just by taking the indirect calculations into account, the number of processors between which data must be transferred can be reduced to only 7 (3 in 2-D) of the neighboring subcells. In this way, all the interactions are accounted for with the minimum amount of data transfer between processors and without creating redundant force calculations.

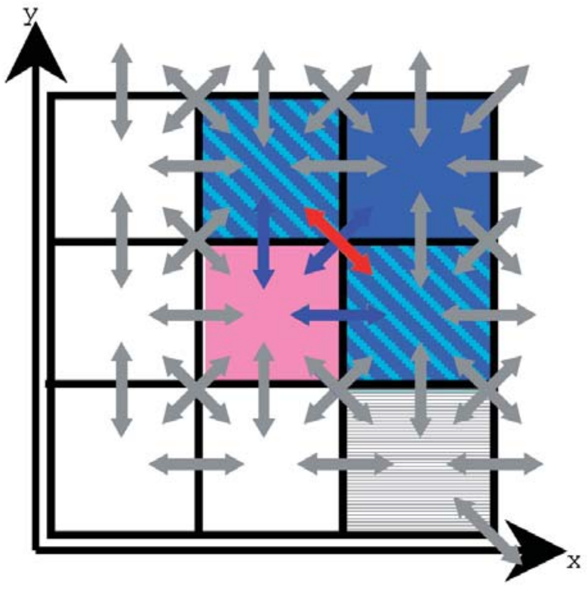

Fig. 1 A schematic 2-D representation of the communication pattern used by PABIOS. We suppose that each processor is responsible for one of nine subcells. The arrows indicate communication between the subcells. Specifically, the blue arrows indicate the direct interactions and the red arrow indicates the indirect interaction. The arrows representing communication between the top and bottom, right and left show when the periodic boundary condition is applied.

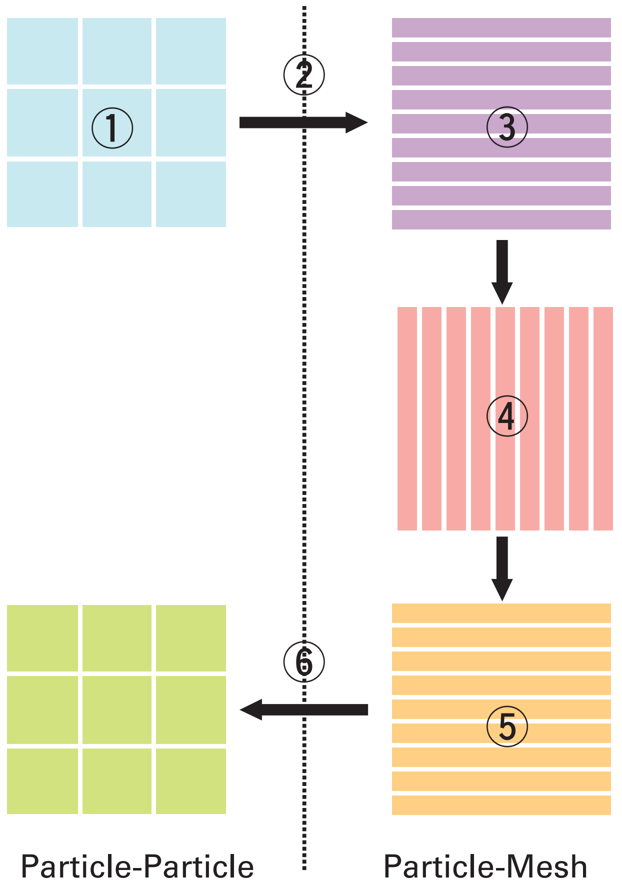

Fig. 2 PABIOS' PPPM calculations are performed in steps: ① PP calculation, ② transfer of coordinates and charge values from subcells to slabs, ③ assignment of charges on mesh in SD space, ④ 3D-FFT of charges on mesh, ⑤ calculation of PM potential on mesh, then evaluation of PM interactions and ⑥ transfer of values of PM interactions to subcells.

2.3. Parallellization of PM part of PPPM

For simulations where the interactions in the system are regarded as short-range and the cutoff method calculates the electrostatic interaction accurately enough, the domain decomposition shows excellent performance. However, when biological systems are simulated, it is known that long-range electrostatic interactions beyond the neighboring subcells should be taken into account.

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. To use this 3D-FFT efficiently, the slab decomposition (SD) method is used by PABIOS as shown in Fig. 2.

3. Vectorization of PABIOS

We concentrated on the vectorization of van der Waals interactions, PP and PM interactions because these non-covalent interactions are the most time consuming part of an MD simulation.

First, we vectorized the subroutines to calculate the van der Waals interactions and the PP interactions. The algorithmic structure of these subroutines is illustrated below:

```
do i = 1, i_atom
 do j = i_atom+1, j_atom
  Calculation of force between i and j
 enddo
enddo
```

Vectorization can be carried out in a straightforward manner by adding a vectorization directive at the innermost do-loop.

Second, we vectorized the subroutines of the PM calculations. The procedure for assigning the charges (③ in Fig. 2) and interaction force interpolation (⑤ in Fig. 2) is done by the same assigning function. As well as FFT calculation, this procedure also requires a significant amount of time.

The force on particle i is given by:

$$\mathbf{F}_i = q_i \sum_{\mathbf{r}_p \varepsilon M} \mathbf{E}_M(\mathbf{r}_p)\, W^{(p)}(\mathbf{r}_i - \mathbf{r}_p)$$

where $\mathbf{E}_M(\mathbf{r}_p)$ is the electric field on mesh point $\mathbf{r}_p$, and $W^{(P)}$ is P-th order even-function for assigning atomic charges to the mesh points. The interpolation order $P = 7$ is large enough for biological MD simulations. The algorithm corresponding to the above equation is written as follows:

```
particle: do n = 1, nparticle
  call $W^{(P)}$(x,y,z,p,wx(P),wy(P),wz(P),i0,k0,j0)
  mesh_z: do k1 = 1, P
    mesh_y: do j1= 1, P
      mesh_x: do i1 = 1, P
        i = I(i0,i1), j = J(j0,j1), k = K(k0,k1)
        ww = wx(i)*wy(j)*wz(k)
        F(n) = F(n) + q(n) * E(i,j,k)*ww
      enddo mesh_x
    enddo mesh_y
  enddo mesh_z
enddo particle
```

The innermost do-loop is not recursive, thus, the outermost do-loop which handles with the particles can be moved into the innermost do-loop by explicitly writing out the assignment function $W^{(P)}$ instead of calling the function.

The assignment of the atomic charges to the mesh can be written in the following way:

$$\rho_M(\mathbf{r}_p) = \sum_i q_i W(\mathbf{r}_p - \mathbf{r}_i)$$

The algorithm for the assignment of charges has a similar structure to that of the force interpolation as shown below:

```
particle: do n = 1, nparticle
  call $W^{(P)}$(x,y,z,P,wx(P),wy(P),wz(P),i0,j0,k0)
  mesh_z: do k1 = 1, P
    mesh_y: do j1 = 1, P
      mesh_x: do i1 = 1, P
        i = I(i0,i1) j = J(j0,j1) k = K(k0,k1)
        rho(i,j,k) = rho(i,j,k) + q(n) * wx(i)*wy(j)*w(k)
      enddo mesh_x
    enddo mesh_y
  enddo mesh_z
enddo particle
```

In this algorithm, however, the innermost do-loop is recursive so we vectorized the assignment function and force interpolation separately. The subroutine for $W^{(P)}$ was explicitly written out to vectorize the do-loop which handles the particles, and the double do-loop of mesh_y and mesh_x was unified into a single do-loop to increase the length of the vectorization.

4. 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 Fig. 3. 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 9 Å. On the basis of the cutoff length the system was divided into 12 × 12 × 12 = 1,728 subcells.

PABIOS' performance on the Earth Simulator was excellent as shown in Fig. 4.

4.1. Parallelization

A parallelization efficiency ratio of 50.5% was achieved even when 18 nodes (144 CPUs) were used. The reason the parallelization efficiency ratio decreased as the number of processors increased is that unparallelized calculations remained in all the processors.

4.2. Vectorization

PABIOS was vectorized until the vectorization ratio exceeded 95% on 10 nodes (80 CPUs). The vectorization ratio for the calculation of short-range interactions of van der Waals and PP was 99.2%. The vectorization ratio for the calculation of force interpolation reached 98.5%, an increase of 4.9% on the original algorithm which achieved a vectorization ratio of 93.6%. The vectorization ratio for the calculation of the charge assignment reached 99.7%, an increase of 2.2% on the original algorithm which achieved a vectorization ratio of 97.5%. The total vectorization ratio was 96.9% when calculated on 10 nodes (80 CPUs). The total vectorization ratio maintained more than 96.0% even when 20 nodes (160 CPUs) were used.

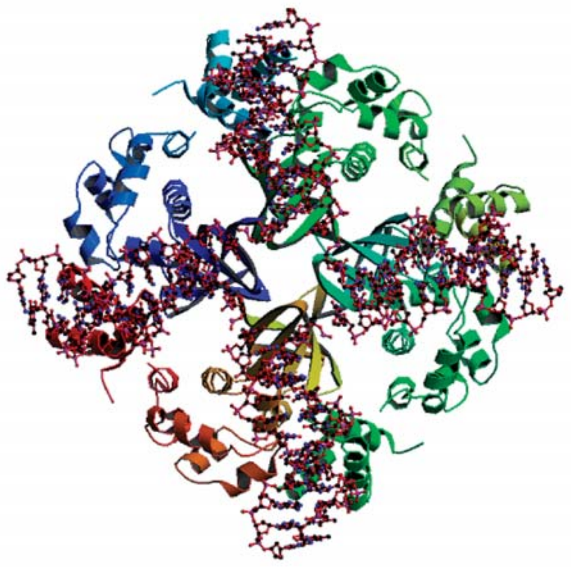
Fig. 3 The system for the benchmark test is RuvA - Holliday junction DNA complex, which is a biomolecular complex consisting of four strands of DNA and four protein molecules which executes recombination of homologous DNA strands. The size of the system and the number of atoms in the system are 114.5 Å × 114.5 Å × 114.5 Å and 166,177 atoms, respectively.

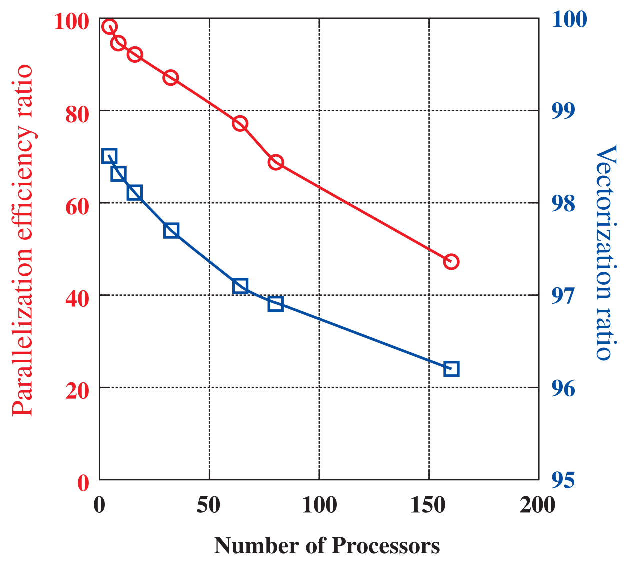

Fig. 4 Present performance of PABIOS on the Earth Simulator. The parallelization efficiency ratio and vectorization ratio are drawn in red and blue, respectively.

5. Conclusion remark

We are going to further develop PABIOS. The relatively complicated communication between the domain decomposition and the slab decomposition in the PM inhibits the computational performance. To deal with this problem, we are now developing a dynamic load balance method.

Once we have implemented this method, 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).