This is the published version:


Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30028584

©2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Copyright: 2009, IEEE
Parallelization of Dissipative Particle Dynamics Simulation

Ye Yue, Ying Zhao, Guohua You, Leqi Tu
School of Information Science and Technology,
Beijing University of Chemical Technology,
Beijing, 100029
{yueye, zhaoy, youguohua, tuleqi}@mail.buct.edu.cn

Atul Sajjanha
School of Engineering and Information Technology,
Deakin University, 221 Burwood HWY,
Burwood, VIC 3125,
Australia
atuls@deakin.edu.au

Abstract

The dissipative particle dynamics simulation is usually used to study polymer in mesoscopic space. The traditional methods are resource intensive, especially when the scale of research is large. Therefore, improving computing efficiency is a key point in this research area. Two major issues are addressed in this paper. First, the DPD methods are analysed and the most time-consuming parts are identified: conservative force, dissipative force and random force. Second, we describe how to parallelize the existing serial application in the Windows Compute Cluster Server (WCCS) platform. The results show that the parallel algorithm not only effectively shortens the computing time, but also improves the resource utilization rate.

1. Introduction

The mesoscopic system is between the macro system and the micro system, and not a pure solid or liquid system. The range of particles in the mesoscopic system is from 10nm to 1000nm [1]. Mesoscopic system has been widely used in material science, life science, environmental science and energy science. The Dissipative Particle Dynamics (DPD) simulation method is one of the general methods in mesoscopic simulation. It is performed to study the sensitive influence of the molecular architecture and/or segment sequence on the morphology diversity of the multicompart-ment [2] micelles. The DPD is usually considered as the bridge from micro-molecular dynamics to macro fluid dynamics. Analyzing the DPD serial algorithms shows that this method takes significant time to simulate the large amount of micro-behaviors of particles. In this paper, we use MPI technologies on WCCS platform to parallelize the DPD simulation program and obtain a speed-up ratio of 1.635. Experiments are also performed on the cluster with 8 nodes to obtain a speed-up ratio of 5.8.

The paper includes 5 parts: Section 1 introduces the paper briefly; the DPD computing method is discussed in Section 2; Section 3 explains how to parallelize DPD serial program; the experimental analysis is given in Section 4; the last part concludes the paper.

2. Dissipative Particle Dynamics

2.1. Dissipative Particle Dynamics

The dissipative particle dynamics simulation method was first introduced by Hoogerbrugge and Koelman as a mesoscopic simulation model in 1985 [3], which are suited for the study of complex fluids such as polymer solutions or colloidal suspensions.

DPD simulation system can contain up to $10^6$ atoms and study particle behavior in microsecond scope. In the DPD simulation method, the particle is a coarse-grained concept [4]. In other words, as every particle represents a large number of molecules, the interactions between them must also be coarsened, and the dynamics of the system is described by the conservative force, which can economize much computing resource and time. There are three interaction forces which are conservative force, dissipative force and random force. In each step, because of the three force action every particle gets a speed and moves to a new location in the simulation space which may cause two particles to collide. It can get the macro result by integrating all the particles. In a DPD simulation, movements of the DPD particles are governed by Newton's equation. The location and speed of particles are determined by:
\[
\frac{dr_i}{dx_i} = v_i \\
\frac{dv_i}{dt} = f_i + F_e
\]  
(1)

where \( r_i \) and \( v_i \) are the position and velocity respectively of the \( i \)th particle; \( f_i \) is the force acting on the \( i \)th particle and \( F_e \) is the foreign force. The quality of particle is the unit quality (\( m = 1 \)), the equation (2) can omit it.

\( f_i \) in equation (2) is computed by three forces which are conservative force, dissipative force and random force:

\[
f_i = \sum_{j \neq i} (F_{c,ij} + F_{D,ij} + F_{R,ij})
\]  
(3)

In equation (3), \( f_i \) is the sum of all the particles except itself in a radius \( r_c \) (when the \( r_{ij} > r_c \) the interaction force is 0).

The conservative force is given by a soft repulsion:

\[
F_{c,ij} = \begin{cases} 
  a_{ij}[1 - \frac{r_{ij}}{r_c}] & r < r_c \\
  0 & r \geq r_c 
\end{cases}
\]  
(4)

where \( r_{ij} = |r_i - r_j| \) and \( a_{ij} \) is the maximum repulsion between a pair of particles.

The dissipative force is:

\[
F_{D,ij} = -\omega D(r_{ij}) \cdot \dot{r}_{ij} \cdot r_{ij}
\]  
(5)

where \( \omega D \) is weight function of \( r_{ij} \) and when \( r_{ij} > r_c \) \( wD = 0 \); \( \dot{r}_{ij} = (v_i - v_j) \) is the relative velocity between the particles; \( \gamma \) is the friction coefficient. The dissipative force acts to reduce the amount of energy in the system by reducing particle velocities if they are approaching one another.

The random force is described as the following equation:

\[
F_{R,ij} = \sigma wR(r_{ij}) \zeta_{ij} r_{ij}
\]  
(6)

where \( wR \) is the random weight function of \( r_i \) and \( r_j \) when \( r_{ij} > r_c \) \( wR = 0 \); \( \zeta \) is a random variable with unit variance, zero mean, and no memory between time-steps or particle pairs. The equations (5) and (6) have a relationship:

\[
\omega D(r_{ij}) = w^2 R(r_{ij})
\]  
(7)

The equations (5) show that the weight function is described as follows:

\[
wD(r_{ij}) = w^2 R(r_{ij}) = \begin{cases} 
  1 - \frac{r_{ij}}{r_c} & r_{ij} < r_c \\
  0 & r_{ij} \geq r_c
\end{cases}
\]  
(8)

where \( r_c = 1 \).

\[\sigma^2 = 2\gamma k_BT\]

\( k_BT \) is the Boltzmann temperature of the system, and the weight function is described as follows:

\[
wD(r_{ij}) = w^2 R(r_{ij}) = \begin{cases} 
  1 - \frac{r_{ij}}{r_c} & r_{ij} < r_c \\
  0 & r_{ij} \geq r_c
\end{cases}
\]  
(9)

\[\text{where } r_c = 1.\ [6-9]\]

2.2. Serial DPD computing algorithm

The initial simulation space is a cube and the size is 30 × 30 × 30 (unit is 1 in algorithm), and there are three particles in it which are carbon, oxygen and a small molecule. The serial algorithm was developed by Thijs J.H.Vlugt in FORTRAN, and the algorithm was based on Linear Molecules Self-Consisting Integration Algorithm. The algorithm is divided into three parts: the first part includes creating simulation area, initializing particles location and initializing speed of particles randomly; the second part is the main computing part, in the conservative force, dissipative force and random force are computed and the new location of particles is determined; the third part mainly focuses on writing the results to files. In the three parts above, the second one is the most time-consuming part which will be parallelized in next section. There is a 22-steps test paradigm running on WCCS and the results are shown in fig1:

\[\text{Fig. 1. Time ratio of DPD text program of 22 steps}\]

The Fig. 1 shows that the computation of the three forces is the most time-consuming. So it is effective to rewrite this part with MPI.

3. Parallelization for the serial DPD algorithm

3.1. The design of Parallel DPD algorithm

The fig1 shows that the most time-consuming part in DPD serial program is the computation of forces and
there are three forces which need to be computed one by one. It is not feasible to reduce the computation time by computing the three forces in parallel because the serial program has a strict sequence. Another scenario is dividing the space. The simulation is in a cube, so the cube can be divided into a few small cubes of equal size. Every small cube can be processed by a node in the cluster.

### 3.2. DPD parallel design

The running space of DPD program is 30×30×30, which means the range is 0-30 in directions X, Y and Z. The basic unit of DPD system is a number of discrete particles with momentum which are affected by the three forces. However, the interaction force only exists within a scope [10], as shown in Fig. 2.

![Fig. 2. The interaction among the particles](image)

Thus, the whole space can be divided and distributed to different computing nodes which can improve the efficiency. However, the problem of boundary is also obvious. The particle, which is near the boundary, not only interacts with other particles in its space, but also interacts with those in the neighboring space, as shown in Fig. 3.

![Fig. 3. Interaction of particles between neighboring spaces](image)

Therefore, the overlap-divided method, shown in fig 4, is used:

![Fig. 4. The overlap-divided method](image)

The overlap-divided method has an overlapping border region of space A and B. For example, in the direction X (the discussion below is also in the X-axis), the range of space A is from 0 to 16 and the range of B is from 14 to 30. That is to say, the neighbor spaces have a share space, from 14 to 16. Furthermore, if the distance between two particles is more than 1, the interaction force is 0. So the overlap-divided method can ensure that the interaction force among the particles from 0 to 15 in space A is accurate. This paper halves the whole simulation space by the overlap-divided method which is called the main space. Each space has an additional space which is to make sure that the main space is the same as the space undivided and two spaces should exchange the data of those additional spaces after some time-steps. For instance the data from 0 to 15 in space A is accurate and the range from 15 to 16 is additional space with the error data. Therefore, the overlap-divided method can ensure that all the computing nodes have correct data and improve the efficiency of computing.

Further more this method can be extended. It can divide the whole simulation space into 8 small spaces, all of which are cubes of equal size. The fig 5 shows the whole space. 8 cubes overlap with each other. In each step, one node in the cluster holds one cube and runs separately. After every step, one cube would exchange data with its neighboring cubes. The paper discusses it in Section 4.4.

![Fig. 5. 8 Cube overlap with each other](image)

### 4. Experimental analysis

WCCS operating system is designed specifically for cluster computing and can be used in supercomputers. It provides easy installation and configuration, friendly interface and convenient job submission compared with the traditional MPI platform on Linux. WCCS also provides task management, job scheduling functions which is better than traditional parallel computing platform. At the same time, WCCS is Windows-based high-performance computing clusters and better integrated, easier to set up and more efficient. Integrating with MS MPI combination of the VS2005 can achieve the functions of parallel debugging, and support a variety of compiler
languages. Active Directory provides end-to-end user and security management.

The parallel algorithm uses two computing nodes in WCCS platform and 8 nodes in cluster. The iterative times are 10,000 which can get a better result. In every step the three forces of every particle should be computed to calculate the speed, which is used to obtain the next location of every particle for the next step. It can achieve macroscopically polymer after 10,000 steps.

### 4.1. Test Environment

This paper does the experiment on two small server computers: first one has a Intel (R) Xeon (TN) 3.0 CPU with 4-core processor and memory is 4G; the second one has a Intel (R) Xeon (TN) 3.0 CPU with dual-core processor and memory is also 4G.

Every node in the cluster has Intel (R) Xeon (TM) 3.0 CPU and total memory is 2G.

### 4.2. Result analysis of serial program on WCCS

The serial program runs only on server A and in WCCS platform. And iterative times are 10,000. The result time is 3 hours 51 minutes 25 seconds and three-dimensional graphics of result is like fig6:

![a. Front](image1.png)  ![b. Back](image2.png)

**Fig. 6. Serial program result on WCCS (View with RasMol)**

Carbon atoms (white part) aggregate together from fig6 and result is better.

### 4.3. Result analysis of parallel program on WCCS

The parallel program uses server A and B. Iterative times also are 10,000. The time result is 2 hours 21 minutes and 31 seconds which is better than serial program. The fig7 show the intermediate results at 5,000 steps:

![a. particle in Server A](image3.png)  ![b. particle in Server B](image4.png)

**Fig. 7. Parallel intermediate results at 5,000 steps (view with RasMol)**

And Fig. 8 shows the final result:

![a. Front](image5.png)  ![b. Back](image6.png)

**Fig. 8. Parallel program result (view with RasMol)**

Serial execution time is 3 hours 51 minutes 25 seconds, or 13,885 seconds, that is, $T_{sw} = 13885s$;

Parallel program execution time 2 hours 21 minutes and 31 seconds, or 8491 seconds, that is, $T_{pw} = 8491s$.

As a result, a computable speed-up ratio $S_w = T_{sw} / T_{pw} = 13885 / 8491 = 1.635$.

### 4.4. Result analysis of serial program on cluster

The serial program runs on one node in a cluster and the number of iterations is 10,000. The computation time is 23842 seconds (about 6 hours 32 minutes) and three-dimensional graphics of result is like fig9:

![a. Front](image7.png)  ![b. Back](image8.png)

**Fig. 9. Serial program result on cluster**
4.5. Result analysis of parallel program on WCCS

The parallel program uses 8 nodes in a cluster and the number of iterations is 10,000. The computation time is 4111.125 seconds (about 1 hour 8 minutes and 31 seconds) which is better than the serial program. The fig10 shows the intermediate results at 5,000 steps:

Fig. 10. Parallel intermediate results at 5,000 steps in one node(view with RasMol)

And Fig. 11 shows the final result:

Fig. 11. Parallel program result (view with RasMol)

Serial execution time is 23842 seconds, that is, \( T_{sw} = 23842s \);

Parallel program execution time is 3559.125 seconds or about 1 hours 42 minutes, that is, \( T_{pw} = 4111.125s \).

As a result, a computable speed-up ratio \( Sw = T_{sw} / T_{pw} = 23842 / 4111.125 = 5.8 \).

5. Conclusion

As can be seen from the results, DPD simulation can have better representation of the relative motion in the field of micro-particles (a collection of atoms or molecules). However, the serial program needs a long time to finish. Therefore, the parallel environment can increase efficiency and shorten the computing time. We used two computing nodes and speed-up ratio reached 1.635. In WCCS platform, the debugging of the program, running and submission is more convenient. In addition, we also performed experiments on 8 nodes of a cluster and the speed-up ratio is up to 5.8. In short, the use of the parallel approach, with high-performance computing platform, as well as cluster server computing resources, can effectively improve the efficiency of the chemical simulation and reduce the costs. As a result, high-performance computing in the field of chemical industry has a broad prospect.

References


