Communications in Physics, Vol. 21, No. 2 (2011), pp. 131-135

ACCELERATED MD PROGRAM USING CUDA TECHNOLOGY

HOANG VAN HUE
Tay Nguyen University
NGUYEN THI THANH HA AND PHAM KHAC HUNG
Department of Computational Physics, Hanoi University of Technology

Abstract. *Molecular dynamic (MD) simulation is proven to be an important tool to study the structure as well as the physical properties at atomic level in materials science. However, it requires a huge computing time and hence limits the ability to treat a large scale simulation. In this paper we present a solution to speed up the MD simulation using CUDA technology (Compute Unified Device Architecture). We used the GeForce GTS 250 card with Version 2.30. The simulation is implemented for Lennard-Jones systems with periodic boundary conditions which consist of 1024, 2048, 4096 and 8192 atoms. The calculation shows that the computing time depends on the size of system and could be decreased by 37 times. This result indicates a possibility of constructing a large MD model with up to* 10^5 *atoms on the usual PC.*

I. INTRODUCTION

Molecular dynamics (MD) simulation is one of the important tools widely used to study the structure as well as the dynamic behavior of materials at atomic level. However, it consumes much computing time that limits the ability to treat a large scale simulation [3-8]. For example, to construct a MD model SiO_2LiO_2 of the 3000 atoms on the PC takes about 60 hours. Hence, the solution to speed up computing process is of great interest for researchers in the field of MD simulation.

CUDA technology is one among approaches to resolve the problem just mentioned [1]. It is an developing platform designed for general-purpose applications, which allow manage the computations on the GPU (Graphics Processing Units) as a data-parallel computing device without the need of mapping them to a graphics API (application programming interface). GPU have many processing cores (processors) to perform computing, each grouped into multiprocessors. There are several levels of memory which differs in terms of access speed and scope: the registers have processor scope; the Shared Memory, Constant Cache and Texture Cache have multiprocessor scope and the Device (or Global) memory can be accessed by all cores on a chip [1]. The computation can be accelerated by data-parallel processing method of hundred of processors. According to the authors in ref. [4] the computing performance of MD simulation can be increased by 3.5 times. Later works shows the effectiveness of GPU computing reached 10-20 [5], 19 [6] or 24 times [7], even up to 100 times [8]. Recently, as noted in [7], with fast development of GPU technology there are many new possibilities in application of GPU computing in the field of scientific research, especially for MD simulations. On the other hand, the cost of GPU cards is relatively small and consistent with final possibility of many labs in Vietnam. In comparison with super computer the PCs with GPU cards could provide the similar computing environment for many scientific problems, although the cost of those computing systems is significantly different. Therefore, the study of applying the computing on PC with GPU is of great interest from both scientific and practical points of view. In this paper, we present the MD program running on GeForce GTS 250 Card and the simulation result of MD models consisting of 1024, 2048, 4096 and 8192 atoms.

The rest of present paper is organized as follows: in section 2 we describe the calculation method included the Molecular dynamic program and the algorithm based on CUDA computing. The section 3 concerns the simulation result and discussion.

II. CALCULATION METHOD

II.1. Molecular dynamic method

Initial configuration of MD model is constructed by random placing all particles in simulation box such that they do not collide. Then the interaction force exerted on each particle by remained particles is calculated. The forces are used to calculate the velocity of each particle and a new distribution of particles is obtained by Newtonian equation of motion. Usually, the potential energy is calculated as a sum of the interaction between every two particles, e.g. the pair interaction potential is employed depending only on the distance between two particles. After that, a new distribution of particles is generated and then again a new one so that eventually a long sequence of distributions evolves. Consider a system contains N atoms, and let all atoms interact with each other according to a pair potential. The potential energy of system U is given as

$$U = \sum_{i<j} \varphi(r_{ij}), \tag{1}$$

here r_{ij} is a distance between the particles i and j. The i^{th} particle feels a force

$$\vec{f_i} = -\vec{\bigtriangledown} U = m_i \vec{a_i}, \tag{2}$$

here m_i is the atom mass, a_i is its acceleration

In this paper, we simulate a system with the Lennard-Jones potential [3] which has the form:

$$\varphi(r_{ij}) = 4\varepsilon \left[\left(\frac{\delta}{r_{ij}} \right)^{12} - \left(\frac{\delta}{r_{ij}} \right)^{6} \right]. \tag{3}$$

The function (3) has a minimum with value $-\varepsilon$. Usually, both ε and δ are chosen to fit some physical properties of the material under consideration. The molecular dynamics codes applied here can be found elsewhere [1-8]. We apply a simple cube box with periodic boundary conditions. At each time step, the force is calculated between each pair of atoms within a cut-off distance taken to be a half of simulation box length. The net force on each atom is used to determine a new position of particles in accordance to $New - ton^{'}$s equation of motion. The integration is performed using a velocity-Verlet algorithm. No cell or neighbor lists were used. The other characteristics such as the pair radial distribution function (PRDF) and dynamic parameters is determined by averaging over several last thousand steps in order to decrease the statistic error related to small size of the model.

II.2. The algorithm of CUDA programming model

The new MD algorithm is designed using CUDA library on NVIDIA GPU and a set of standard commands from NVIDIA codes [1]. Similar to graphics applications, CUDA applications can be accelerated by data-parallel computation of millions of threads. This is a main concept of CUDA parallel computing model. The threads must execute the same function with different parameters. The function that contains the computations and runs parallel in many instances is called the kernel. Threads in the same thread group can synchronize with each other, by inserting synchronization points in the kernel, which must be reached by all threads in the group before continuing execution. The threads can also share data during execution. By this way, usually several hundred threads in the same block can work cooperatively. Each blocks can contain anywhere from 32 to 512 threads, but all blocks must be the same size. Any number of blocks can be run on the device, though optimum performance requires more than a hundred blocks executing in parallel. Each block executes identical lines of code and is given an index starting from 0 to identify which is a piece of the data. Within each block, threads are numbered from 0 to identify the location of the thread within the block.

Our implementation of the molecular dynamics algorithm was developed for and tested on Version 2.30, GeForce GTS 250 card. The algorithm consists of two blocks:

1/ To transfer the data from CPU to GPU for operating the parallel calculations on GPU and to transfer the result back from GPU to CPU

2/ To performance the computing on 128 processors of GPU.

The efficiency of the algorithm depends on both blocks. Such the computing time must consume much less than that the memory transferring in the second block has spent.

A Cuda program is composed of two parts: a host (CPU) code that makes kernel calls, and a device (GPU) code that actually implements the kernel. The host code is conceptually a serial C program, but the device code should be massively parallel in order to harness the power of the GPU. In our program the force calculation and the atomic movement have been operated on GPU. The remaining works is performed on CPU as in the case of MD program for CPU. Note that the speed of parallel computing on the GPU significantly depends on how divide the computing task between the processors and also to use the local memory of GPU. After intensive treating a number of possible ways of parallel computing and memory map we found that the computing speed for different ways may reach a difference in 10 times.

III. DISCUSSION AND CONCLUSION

Firstly we test the validity of our new MD program by comparing the PRDF obtained from both programs. Fig. 1 shows the PRDFs for two MD models constructed by both MD programs. One can see that they are identical indicating the correct computing result of new MD program. To evaluate the efficiency of new MD program we use two quantities:

1/ the speed-up factor determined as the ratio of computing speed of new and former programs;

2/ the real runtime.

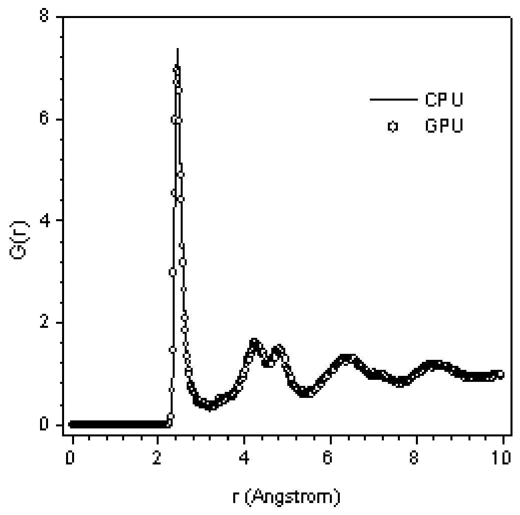

Fig. 1. The radial distribution function running GPU and CPU

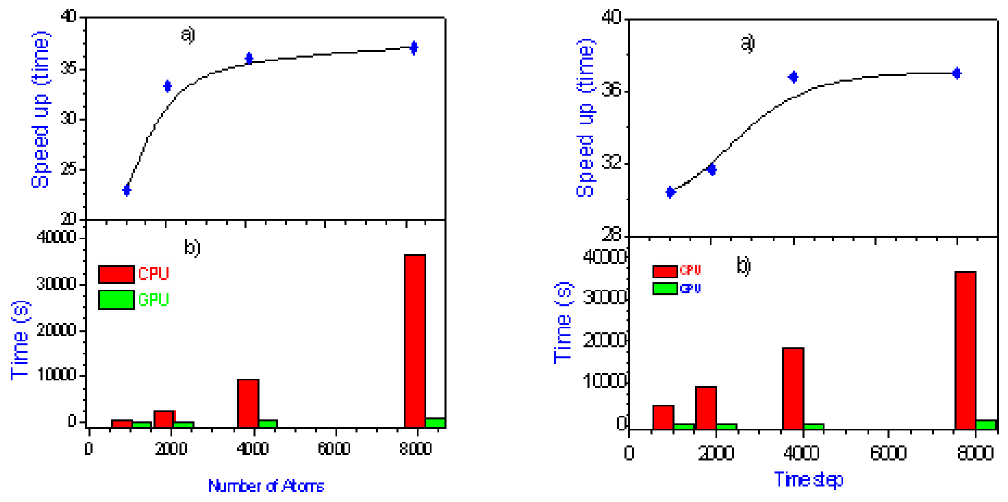

Fig. 2. The dependence of the speedup- factor (a) and runtime (b) on the number of atoms (left) and the dependence of the speedup-factor (a) and runtime (b) on the number of time steps (right)

Fig. 2 (left) shows the results of our MD simulation under different sizes of simulation box (number of atoms). The speed-up factor and runtime are presented as a function of model size. As the number of atoms varies from 1000 to 4000, the speed-up factor increases, but the speed-up factor remains unchanged with further increasing model size. The maximal value of speed-up factor is about 37 times.

Fig. 2 (right) presents the result of dependence of GPU efficiency on the number of time step. We again observe the similar trend as in the case of model size. The GPU efficiency may reach 36 times when the MD simulation runs up to 6000. After running up over 8000 steps the averaged runtime per one step is 5 and 0.9 seconds for CPU and GPU MD programs respectively.

Such, our simulation shows that the MD program running on on GeForce GTS 250 Card allows speed up to 37 times. It opens a new possibility to make a large scale simulation. For example, to construct a MD model of iron liquid of 5000 atoms consumes about 30 hours on usual PC. Using a new MD program the model of 10^5 iron atoms can be prepared over about the same time on the same PC with GeForce GTS 250 Card.

REFERENCES

[1] NVIDIA CUDA Programming Guide, *available from http://www.nvidia.com/object/cuda develop.html*.

[2] S. Plimpton, *Journal of Computational Physics* **117** (1995) 1.

[3] F.H. Stillinger, D.K. Stillinger, *Mechanics of Materials* **38** (2006) 958.

[4] M.C.Schatz, C.Trapnell, A.L.Delcher, A.Varshney, *BMC Bio Informatics* **8** (2007) 474.

[5] Daniel Castaño-Díez, Dominik Moser, Andreas Schoenegger, Sabine Pruggnaller, Achilleas S. Frangakis, *Journal of Structural Biology* **164** (2008) 153.

[6] Weiguo Liu, Bertil Schmidt, Gerrit Voss, Wolfgang Mller-Wittig, *Computer Physics Communications* **179** (2008) 634.

[7] Yusuke Okitsu, Fumihiko Ino, Kenichi Hagihara, *Parallel Computing* **36** (2010) 129.

[8] F. Molnar *et al.*, *Computer Physics Communications* **181** (2010) 105.

Received 25 September 2010.