Science in China Series B: Chemistry

www.scichina.com
chem.scichina.com
www.springerlink.com

Molecular dynamics simulation of complex multiphase flow on a computer cluster with GPUs

CHEN FeiGuo[1, 2], GE Wei[1†] & LI JingHai[1]

[1] State Key Laboratory of Multi-Phase Complex Systems, Institute of Process Engineering, Chinese Academy of Sciences, Beijing 100190, China;
[2] Graduate University of Chinese Academy of Sciences, Beijing 100039, China

Compute Unified Device Architecture (CUDA) was used to design and implement molecular dynamics (MD) simulations on graphics processing units (GPU). With an NVIDIA Tesla C870, a 20—60 fold speedup over that of one core of the Intel Xeon 5430 CPU was achieved, reaching up to 150 Gflops. MD simulation of cavity flow and particle-bubble interaction in liquid was implemented on multiple GPUs using a message passing interface (MPI). Up to 200 GPUs were tested on a special network topology, which achieves good scalability. The capability of GPU clusters for large-scale molecular dynamics simulation of meso-scale flow behavior was, therefore, uncovered.

multiphase flow, molecular dynamics, CUDA, GPU, parallel computing

1 Introduction

Molecular dynamics (MD) is a method in scientific computation that grows with computer technology. Since the first application by Alder[1] in 1956, MD has been widely applied in many fields, such as medicine, material, energy and electro-mechanics. With the increasing interest in micro-/nano-electro-mechanical-system (MEMS/NEMS) and micro-chemical technology, MD simulation of micro-/nano-scale flow has become the new focus recently. For traditional continuum hydrodynamics, it is difficult to portray the special characteristics and phenomena at such micro-/nano-scale. In final analysis, fluid flow is the collective behavior of fluidic molecules, and MD simulation can trace the movement of each molecule in detail. Therefore, the statistical properties of velocity and temperature, etc., from MD simulation, can be employed to explore the difficulties in present theories and discover new mechanisms. But the progress of MD simulation has always been limited by the computational cost.

Most sophisticated MD algorithms are based on the traditional architecture of central processing unit (CPU), but the development of CPU technology has shown a sign of slowing down: It is difficult for Moore's Law[2] to sustain and the demand for computing power is growing much higher than Moore's Law. On the other hand, the performance of modern graphic processing unit (GPU) is 1—2 orders higher than CPU. The Huang's Law can describe GPU development, named by the President of NVIDIA, Jen-Hsun Huang, faster than Moore's Law. GPU computing has the advantages of high performance-to-price ratio, high computational intensity, low power consumption, etc. Therefore, it is of highly practical significance to employ GPU in MD simulation.

Based on the structural analysis of the hardware, the high performance of GPU relies on a large number of thread processors in parallel computing and high-speed small-scale multi-cache sharing. GPU computing suits applications similar to graphic operation or data-intensive. The typical application is MD simulation with the

Received June 12, 2008; accepted October 8, 2008
doi: 10.1007/s11426-009-0069-0
†Corresponding author (email: wge@home.ipe.ac.cn)
Supported by the National Natural Science Foundation of China (Grant Nos. 20336040, 20221603 and 20490201), and the Chinese Academy of Sciences (Grant No. Kgcxz-yw-124)

features of simple molecular information, independent movement, pair-wise additive and short-range interactions, and large-scale long-time computation. But traditional GPU supports non-graphics applications little, with graphics API programming only. These difficulties in accessing GPU resource restrict the GPU performance in general purpose computation. CUDA[3] is a new hardware and software architecture for issuing and managing computation on GPUs as data-parallel computing device directly, without mapping computation to graphic operations. CUDA environment provides more freedom to implement algorithms on GPUs with C-like API, and CUDA can also be combined with other parallel modes, such as OpenMP, MPI and PVM, to achieve intra-/inter- node distributed computation.

Nowadays, there have been some cases of computation on GPU with CUDA, such as N-body simulation in astrophysics[4], LBM[5], and MD[6,7]. In these cases, GPU can achieve a speed being dozens of times to one hundred times higher than that of one core of a typical modern CPU. But MD simulation fully implemented on multiple GPUs or aiming at the multiphase system has not been seen as far. In this work, we perform MD simulation of multiphase nano-/micro-scale flow on multiple GPUs. The details of MD simulation and its implement on GPUs are described in Section 2, and the validity of the GPU-MD algorithm and its performance on GPUs are tested in Section 3. Section 4 introduces the MD simulation of cavity flow and particle-bubble interaction in liquid, using this algorithm. Section 5 summarizes the work and prospects future simulation work on general purpose GPU.

2 Approaches

2.1 Molecular dynamics simulation

The simulation approaches employed in this work are basically of the MD catalogue. The truncated Lennard-Jones potential is commonly used in the interaction between molecules, that is,

$$\phi(r_{ij}) = \begin{cases} 4\varepsilon\left[a\left(\dfrac{\sigma}{r_{ij}}\right)^{12} - b\left(\dfrac{\sigma}{r_{ij}}\right)^{6}\right], & r_{ij} = \left|\mathbf{r}_i - \mathbf{r}_j\right| \leqslant r_c, \\ 0, & r_{ij} > r_c, \end{cases} \quad (1)$$

where σ is the characteristic length of the interaction usually taken as the molecular diameter, ε is the energy parameter characterizing the interaction strength, r_{ij} is the distance of molecule i to molecule j, and r_c stands for the cutoff distance. Coefficients a and b represent the relative strength of attractive force and repulsive force, usually a=b=1, and a=1 and b=0 gives repulsive force only. Each wall molecule is assumed to be anchored at its lattice site by a Hookean spring with an additional harmonic potential[8] of

$$\phi(s) = 1/2\, Cs^2, \quad (2)$$

where C is the stiffness of the spring and s is the displacement of the wall molecule from its lattice site.

Considering the features of GPU computation, the linked-cell algorithm is adopted to organize and search simulation data efficiently. The simulation region is divided into sub-regions (called cell) with the size of the cutoff distance r_c or a little more. Thus, each molecule interacts only with the molecules in the same cell or in neighbor cells, and the number of these cells is independent of the simulated system size (9 for two-dimensional system and 27 for three-dimensional system). Therefore, the computational complexity is proportional to molecule number N, that is, $O(N)$.

Among a variety of schemes to realize linked-cell algorithm, organizing molecular information in the same cell with linked list scheme has the advantage of suiting any number density without parameter modification. But linked list accesses memory randomly and parallel computation within a cell is difficult. Another scheme using array of given size to store molecular information in a cell can avoid this defect and access memory continuously and in parallel, but the array size must exceed the maximum molecule amount in one cell, so memory consumption may be a problem. This work adopts the latter scheme, that is, during GPU computing, one thread block corresponds to one cell and one thread corresponds to one molecule.

2.2 GPU-MD algorithm

The linked-cell algorithm of MD simulation can be divided into several stages including moving molecules, updating the map of molecules to cells, and computing forces between molecules. And the leap-frog method[9] is employed in the path integration of molecular movement,

$$\begin{aligned} v(t+\Delta t/2) &= v(t) + \frac{f(t)}{2m}\Delta t, \\ r(t+\Delta t) &= r(t) + \Delta t v(t+\Delta t/2), \\ v(t+\Delta t) &= v(t+\Delta t/2) + \frac{f(t+\Delta t)}{2m}\Delta t, \end{aligned} \quad (3)$$

The integrating process has two steps. At time t, the ve-

locity of the molecule $v(t)$ is updated to $v(t+\Delta t/2)$ according to the force on the molecule at the current time t, and the position of the molecule $r(t)$ is also updated to $r(t+\Delta t)$. Subsequently, the force on the molecule at the time $(t+\Delta t)$—$f(t+\Delta t)$ is computed using the position at the time $(t+\Delta t)$, and the velocity is updated from $v(t+\Delta t/2)$ to $v(t+\Delta t)$, then the position and velocity of the molecules get synchronized. The flow-sheet of MD simulation on GPU is represented in Figure 1.

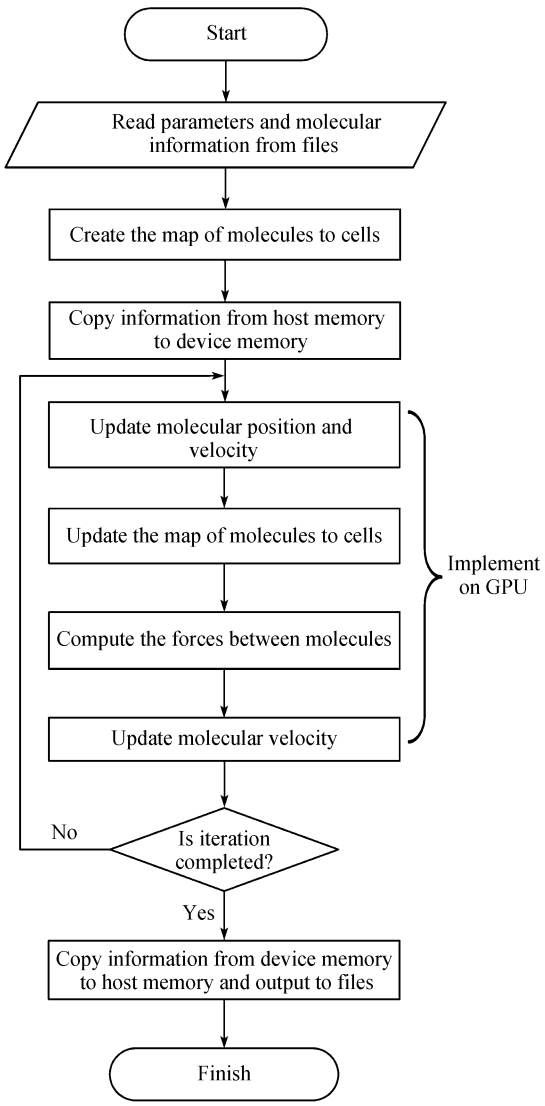

Figure 1 Flow-sheet of MD simulation on GPU.

We have tried to use GPU in the most time-consuming part of the computation, and the interaction between molecules only, but the frequent data transfer between host memory and device memory has become the bottleneck for GPU performance (the bandwidth of PCI-E Gen2 x16 bus is about 8 GB/s, far less than the bandwidth inside GPU card 76.8 GB/s[1)]), so this mode does not work efficiently. In the present work, all stages of the MD simulation algorithm are implemented on GPU. Thus, before the computing procedure, CPU (host thread) reads simulation parameters and molecular information from files and creates the map of molecules to cells, and then the map is copied to device memory for GPU calling.

The implement of molecular movement on GPU is simple. That is, each thread updates the position and velocity of the corresponding molecule. The variation of molecular position may change the map of molecules to cells, and re-mapping is needed. Therefore, each thread calculates in parallel the cell indexes of the corresponding molecule, and then molecules having new cell indexes are removed from the map of molecules to cells. To search neighboring cells (excluding the current cell, the number of neighboring cells is 26 for three-dimensional system), newly entered molecules are inserted into the map of molecules to cells of the current cell. To avoid writing conflict, the operations of remove and insertion are implemented only by one thread per thread block. The old map arrays are backed up before removing the molecules to ensure the integrity of the old map, and the inserted molecules can be found in the arrays of the backup map.

The computation of the interactions between molecules is the most time-consuming part in MD simulation. The ith thread of a thread block reads the ith molecules in cell c_0 and its neighboring cell c_n from global memory to (on-chip) shared memory. After the data for the same block is completely loaded (through the synchronizing operation among threads of the same block), the ith thread handles the force computation between the ith molecule in cell c_0 and each molecule in cell c_n. Thus, computation of the total force on every molecule in cell c_0 from neighbor molecules in cell c_n is performed. In three-dimensional system, n=0, 1,⋯, 26, that is, cell c_0 has 27 neighboring cells (including cell c_0 itself). Finally, the updated molecules in shared memory are written back to their corresponding global memory space.

2.3 GPU-MD + MPI parallel algorithm

Though performance of a single GPU is already very

1) Memory bandwidth of Tesla C870, from http://www.nvidia.com/object/tesla_c870.html.

powerful, it is far from enough for large-scale MD simulation yet. In this paper, the domain decomposition scheme of good scalability (Figure 2) and the commonly used MPI (Message Passing Interface) protocol are employed to achieve inter-nodes multiple GPU computing. In every time step, each MPI process handles the computation of molecular force, velocity and position in the corresponding sub-region. If a molecule moves out the original sub-region, it will be transferred to the new sub-region. Before force computing of the molecules in the inner boundary cell (Figure 2), the neighbor molecules in the outer boundary cell are transferred from the neighbor process.

In the multiple GPU algorithms, CUDA and MPI are transparent to and independent of each other. The computation is fully implemented on GPU, and the inter-process communication is carried out by CPU, that is, the communication mode is GPU-CPU-CPU-GPU. The bottleneck limiting computing performance is the bandwidth of PCI-Ex16 bus (for transferring data between GPU and CPU), and the bandwidth of network between nodes (gigabit Ethernet). Therefore, GPU computing time should be as long as possible (each process handles as many molecules as possible, that is, coarse-grain parallel decomposition) or overlapping of computation and communication is needed to reduce the relative communication time. The flow-sheet of MD simulation on multiple GPUs is represented in Figure 3.

3 Performance of GPU

The theoretical peak performance of GPU is very good, however, the actual performance to a great extent depends on the algorithm. The memory access pattern deeply influences GPU performance. In the CUDA environment, the memory on GPU card consists of (multiprocessor) on-chip memory and off-chip memory. For Tesla C870 GPU, off-chip memory includes global memory (1.5 GB, read-write enable, not cached), constant memory (64 KB, read-only, cached) and texture memory (maximum width is 2^{27} bounding to linear memory, read- only, cached); on-chip memory is also called (multiprocessor) local memory, and each multiprocessor has 8192 32-bit registers, 16 KB shared memory (if no bank conflict, the access speed is the same as registers) and instruction cache (not accessible to user). The bandwidth between GPU and global memory is large (76.8 GB/s), but the latency is high (400—600 clock cycles), and CUDA supports a coalesced access pattern that multiple memory accesses can be integrated into a single contiguous aligned memory access to hide latency[3].

Using the on-chip memory (shared memory and registers) to the utmost and alescing global memory access can improve the performance of GPU in computation-intensive applications. In our GPU-MD algorithm, one thread block corresponds to one cell and one thread corresponds to one molecule. The molecular information is loaded to shared memory from its global memory space and resident in shared memory until the total force on this molecule from its neighboring molecules (about several hundred neighboring molecules) is summed, and then the newer molecular information is rewritten back to the global memory space. That is to say, one global memory access goes with hundreds of computing operations, and in the same thread block, each thread accesses contiguous global memory space. In this way, the efficiency of data access and GPU performance can be improved.

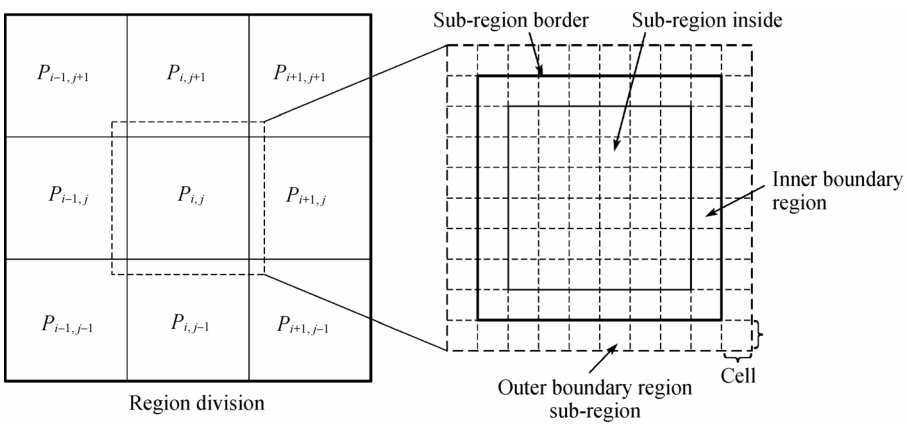

Figure 2 Illustration of two-dimensional domain decomposition.

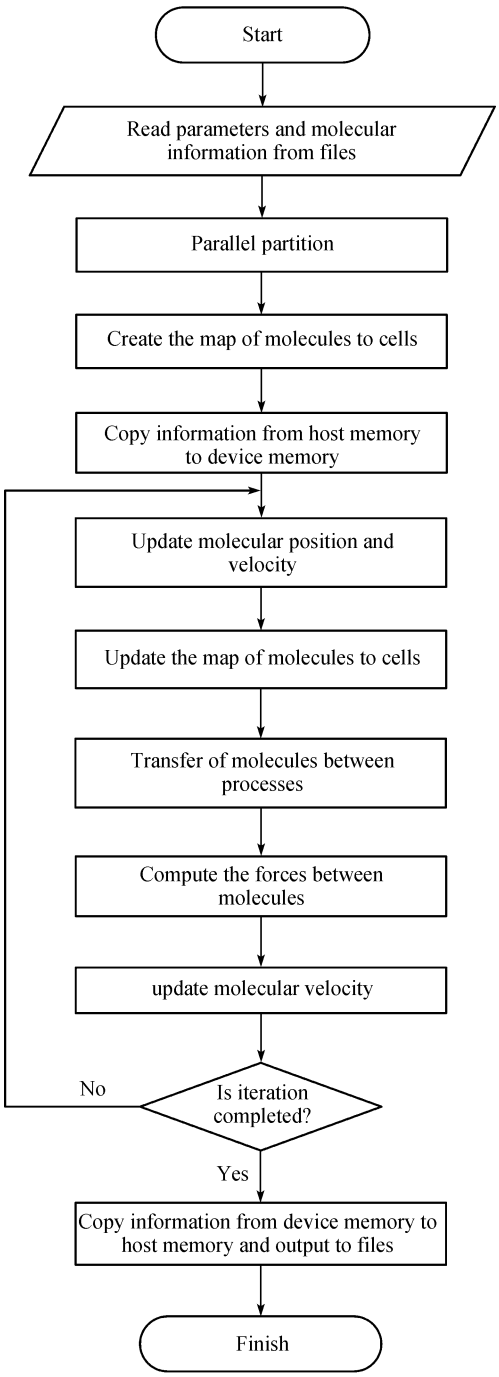

Figure 3 Flow-sheet of MD simulation on multiple GPUs.

In the parallelization mode of GPU+MPI, the force computing of the molecules at the process boundary requires neighboring molecular information in neighboring processes. The molecular position is updated in every time-step, and the communication between processes is needed in every time-step. So the communication is local, frequent and trivial. Single phase MD parallel simulation has balanced load, and therefore has higher parallel efficiency. To compare their performance, a simple MD simulation is implemented on CPU and GPU, respectively, and the simulation parameters are listed in Table 1.

In the simple MD simulation above, the performance of GPU is about 45 Gflops, and one core of Intel Xeon 5430 CPU achieves about 2.0 Gflops, speeding up about 20－30 fold. If only the most time-consuming force computing between pair molecules is considered, GPU is more advantageous, the performance is 150 Gflops, which is about 40% of its utilizable peak performance (346 Gflops), and the corresponding CPU performance is about 2.4 Gflops. The speedup of GPU over CPU increases 60 fold. We also tested larger-scale simulaton on 200 GPUs in a cluster with 100 nodes (equipped with two Tesla C870 cards per node). The performance for force computing only is about 30 Tflops, and the overall performance is about 8－12 Tflops, where the communicating time is 10%－25% (depends on simulation case and parallel decomposition) of the total. The details are listed in Table 2.

It should be pointed out that regardless of GPU or CPU, the computing performance much depends on the MD simulation case. The above-mentioned CPU program has been optimized with SSE instruction to improve force computing which has increased by 1－2 times; but for actual MD simulation with truncated length, the optimization only achieves about 30% performance increasement. As for GPU computing, the same amount of thread block has the same time consumption. Considering a simulation system containing a certain amount of molecules, higher number density means less cells (thread blocks), and therefore, the time consumption is less. In MD simulation case above, the cutoff distance

Table 1 Parameters for performance comparison

	Potential parameters					System parameters				
	r_c	ε	σ	a	b	region	temperature	number density	molecule number	timestep
MD	5	1	1	1	1	100×100×100	1	0.729	7.29×10^5	0.005
MD, only force[a)]	5	1	1	1	1	100×100×100	1	1	1×10^6	0.005

a) Only compute the force between molecules.

Table 2 Comparison of time per step and performance in MD simulation (the simulation region for each case (or one MPI process) is 100×100×100)

	Move (s/timestep)	Update cell (s/timestep)	Compute force (s/timestep)	Communicate (s/timestep)	Total (s/timestep)	Performance (Gflops)	Speedup
CPU	0.036	0.046	14.56		14.67	2.05	1
GPU	0.042	0.146	0.46		0.68	44.30	21.6
GPU+MPI	0.037	0.206	0.45	0.115	0.81	37.19	18.1
CPU, r_c=20	0.074	0.106	1054.83		1055.09	1.82	1
GPU, r_c=20	0.042	0.144	16.57		16.79	114.83	63.1
CPU, only force[a)]			29.07		29.07	2.44	1
GPU, only force			0.48		0.48	148.66	60.9

a) Only compute the force between molecules.

r_c=5 and each cell contains about 100 molecules, that is to say, only few threads handle the virtual molecules which do not contribute to the simulation, resulting in little penalty. When the cutoff distance is increased to 20 (not changing the size of cell but seraching more neighboring cells), CPU achieves slightly lower performance because of frequent memory accesses, but the performance of GPU increases by about 1.6 times (the relative force computing time increases from original 67.6% to 98.7%), showing that GPU computing is suitable for long-range force computation in multi-body systems.

4 Application of the GPU-MD algorithm

4.1 Cavity flow simulation

Cavity flow is of great scientific interest as it can display kinds of complex fluid mechanical phenomena in simple geometries, for instance, corner eddies, longitudinal vortices, nonuniqueness, transition and turbulence all occur in this flow naturally[10]. Another particularity is that the no-slip boundary condition is not held at the corner of the upper driving lid, and the corner is the so-called singular point. Cavity flow has made a lot of researchers use various methods to conduct study. Recently, many computer simulations of cavity flow have been reported, such as CFD[11], LBM[12,13] and MD[14,15]. MD simulation for cavity flow is free of arbitrary assumption on boundary conditions and the simulation results are helpful to explaining the mechanism of cavity flow.

We apply GPU-MD algorithm to cavity flow simulation. The simulation system gets much larger than ever, reaching (sub) micro-scale to a certain extent, bridging continuous hydrodynamics and discrete molecular dynamics. The simulation parameters are listed in Table 3. The number of molecules in the simulation region is 3.2×10^6 and the timestep is Δt=0.005τ (τ=$(m\sigma^2/\varepsilon)^{1/2}$ is the characteristic time). The example is implemented on 4 GPUs, with Maxwell thermal wall boundary to keep wall temperature at k_BT=0.8ε and the lid-driving velocity U along +X direction.

As seen in Figure 4, there is the characteristic primary eddy of caivty flow in each profile with almost unique location but different forms. Actually, the primary eddy is extending along the Z-direction and has three-dimensional structure[10] (Figure 5). It is also seen in Figure 4 that there are two corner eddies near the bottom floor. The velocity profiles at X=100 and Y=100 (the driving-lid is located at Y=200) are shown in Figure 6, where the arrows show the velocity direction and its length respresents velocity magnitude. The simulation results above accord with some experiments and previous simulations qualitatively, and the quantitive comparison needs larger scale system simulation and more analyses.

Table 3 Parameters for cavity flow simulation

	Density ρ	Temperature T	Viscosity μ	Sonic speed c	Region $L\times H\times D$	Lid velocity U	Reynold number $Re=U\rho H/\mu$	Mach number $Ma=U/c$
Dmensionless value	0.8	0.8	2.28	5.42	200×200×100	2	140.4	0.37
Dmensional value[a)]	350 kg/m^3	96 K	2.08×10^{-3} P	861 m/s	68 nm×68 nm×34 nm	318 m/s	140.4	0.37

a) According to argon parameters: σ=3.40×10^{-10} m, ε=1.67×10^{-21} J, m=6.63×10^{-26} kg, τ=2.14×10^{-12} s.

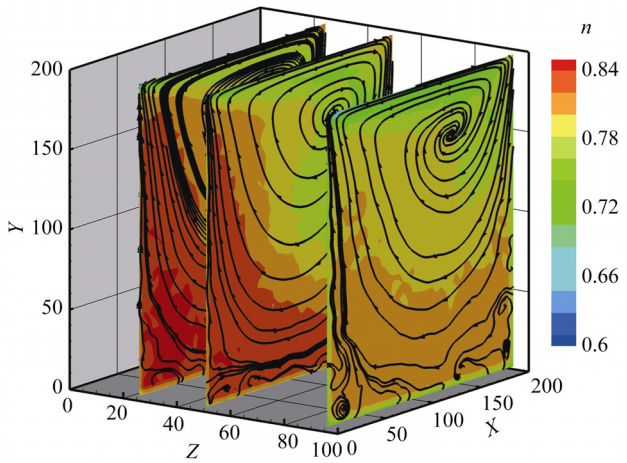

Figure 4 Profiles at Z=25, 50 and 95. The nephogram represents the molecular number density and the lines denote streamlines.

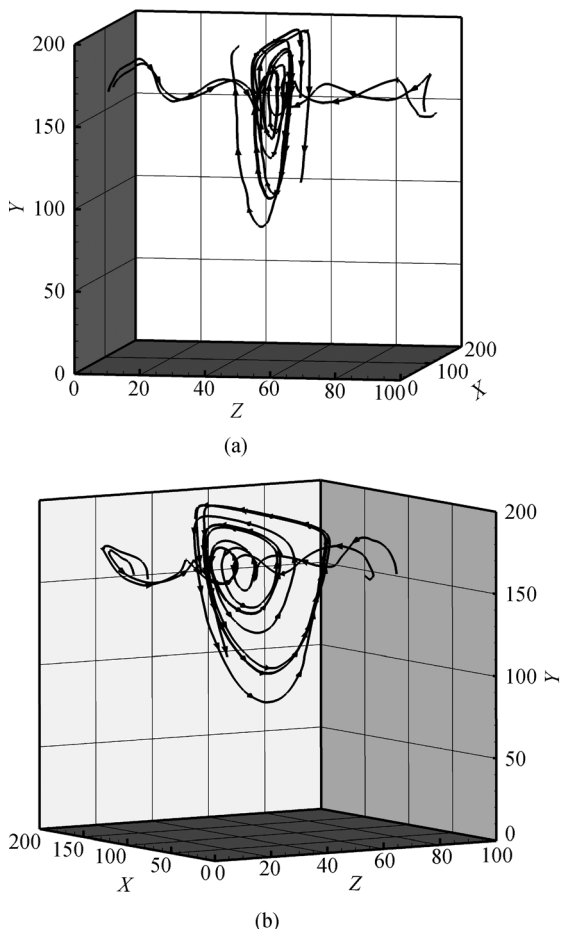

Figure 5 Three-dimensional structure of the primary eddy, as seen from different angles ((a), (b)). The lines denote two streamlines near the primary eddy. It is seen that in the Z-direction, the primary eddy is a single one within the span of these two streamlines and its three-dimensional structure is obvious.

4.2 Multiphase flow simulation

Multi-phase nano-flow and micro-flow are more practical

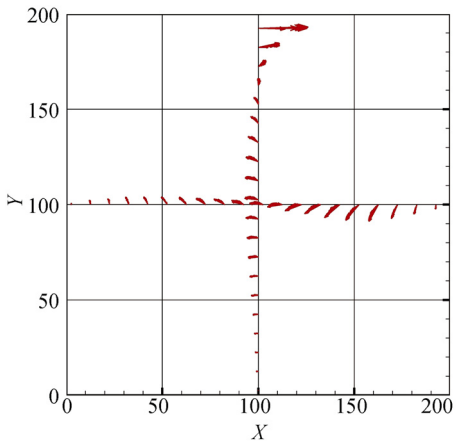

Figure 6 Velocity vector profiles at X=100 and Y=100.

for nano techology, material, biology, MEMS, micro-chemical technology, etc., but also more challenging. So far, no matured theoretical model exists for multi-phase nano- and micro-flow, and the corresponding MD simulation is more complex and needs more computational resources. Therefore, the application of multiphase flow MD simulation on GPU becomes more valuable.

There are six pair interactions (in this case all are presented as Lennard-Jones potentials) between molecules in gas-liquid-solid three-phase system. The detailed parameters are: for liquid-liquid, r_{cl}=5, σ_l=1, ε_l=1, m_l=1, a_l=b_l=1; for gas-gas, LJ potentail with the repulsive term only, r_{cg}=2, σ_g=0.5, ε_g=1, m_g=1, a_g=1, b_g=0; for solid-solid, r_{cs}=5, σ_s=1, ε_s=1, m_s=1, a_s=b_s=1; for liquid-gas, LJ potentail with the repulsive term only, r_{cgl}=3, σ_{gl}=1, ε_{gl}=1, a_{gl}=1, b_{gl}=0; for liquid-solid, r_{csl}=5, σ_{sl}=1, ε_{sl}=1, m_{sl}=1, a_{sl}=b_{sl}=1; for gas-solid, LJ potentail with the repulsive term only, r_{csg}=3, σ_{sg}=1, ε_{sg}=1, a_{sg}=1, b_{sg}=0. The stiffness of the spring anchoring solid molecules is set as C=75.

To validate the three-phase MD simulation model on GPU, a gas-liquid MD simulation on the relaxion of an initially cubic bubble to spherical bubble is implemented first. Nanobubble was observed by Gogotsi et al.[16] and its behavior has been widely studied with MD simulation (implemented on CPU). Sushko and Cieplak[17] and Chen et al.[18] simulated gas molecules directly; on the contrary, other reasearchers[19−22] dealed with the nanobubble as cavity, but did not simulate the gas molecules directly. In this case, the simulation region is set as 100×100×100. The bubble is initially cubic, set at the center of the region with the size of 50×50×50 (Figure 7(a)). The gas moelucle number is N_g=91125 with the number density n_g=0.729, and the liuqid molecule number N_l=625177 with the

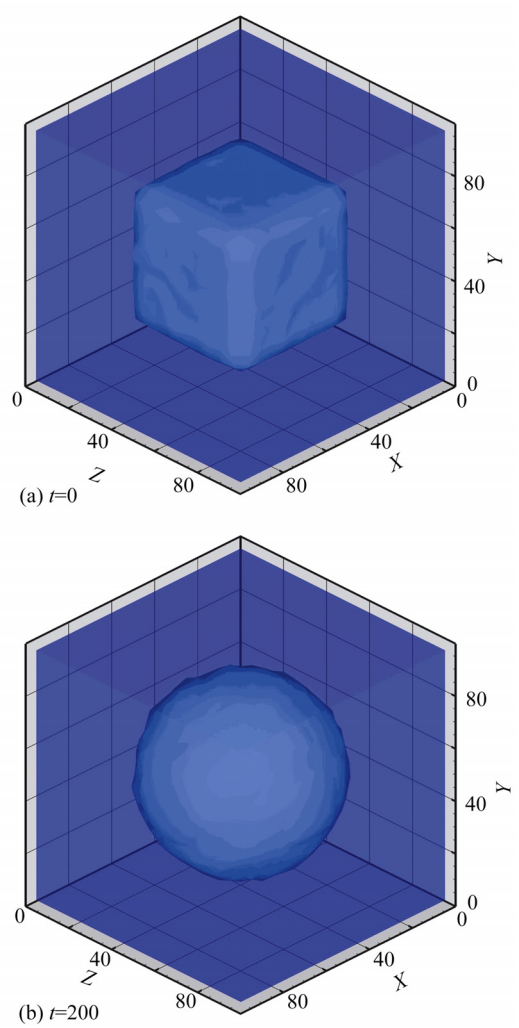

Figure 7 The initial cubic bubble (t=0) relaxes to the (nearly) spherical shape in equilibrium (t=200).

number density n_l=0.714. The temperture of the simuation system is kept at k_BT=1 and the timestep is Δt=0.005. Periodic boundary conditions (PBCs) are applied.

The system is relaxed for 200 time units during which the cubic bubble deforms to a (nearly) spherical bubble (Figure 7(b)) under the action of interfacial tension and the interfacial energy reaches its minimum. In equilibrium, the diameter of the bubble is D_b=70.1 (its volume is V_b=1.80×10^5), and the number density of gas is n_g=0.506 and the number density of liquid is n_l=0.763. Here, the difference of density between gas phase and liquid phase is not significant like normal gas liquid two-phase system. Actually, nanobubbles are compressed to high density under high interfacial additional pressure. In fact, the bubble is still considered as gas phase because only repulsive force presents between gas molecules. As simple Lennard-Jones type potential is hard to express complex interfacial interaction, the rigidity of the bubble in this case is much weaker than real nanobubble. But from the viewpoint of the algorithm, the case can still be considered as a validation of multiphase MD simulation implemented on multiple GPUs.

Subsequently, the particle-bubble interaction in a liquid nano-channel was simulated. The detailed settings are listed below. The simulation region is 80×160×80 with three-layer walls on both sides (the wall molecule number density n_w=1 and total number N_w=76800). The bubble diameter D_b=50 (containing 45965 gas molecules) and it is placed at the bottom of the channel. A solid grain of diameter D_s=20 (containing 3544 solid molecules) is fixed at the top of the channel. The remaining space is filled with liquid molecules with the number density n_l=0.733 and the total liquid molecule number N_l=643240. The initial system temperature is k_BT=1 and the driving bulk force is g=2×10^{-3}. PBCs are applied in the Y-direction and Z-direction and the wall temperature is kept constant using the rescaling-velocity thermostat.

The gas and liquid are driven to flow by a constant bulk force (g=2×10^{-3}). Before contacting solid grain, the gas bubble moves along the center line under the balanced forces from the liquid, and it deforms to the cappy shape (Figure 8(b)). When the gas bubble meets the solid grain first time, the bubble passes through by the solid grain (Figure 8(d)); then buble movement is biased to one side due to the asymmetrical flow field around the solid grain. When the bubble contacts the solid grain again, the solid grain is not able to pass through the bubble owing to the restriction of the gas-liquid interfacial tension (Figure 8(e)), and the bubble moves along, attaching to the side wall. The movement is similar to the typical process of gas-solid attachment in flotation[23,24], but it requires further analysis.

The simulations above are only used to demonstrate the application of the GPU-MD algorithm. The phenomena gotten are only numerical results by the given potential function and parameters, which may not refer to the actual situation. But the simulation can be cosidered as an ideal model to study multiphase flow at nano- and micro-scale. In addition, MD simulation can achieve extreme conditions which are inaccessible to physical experiments, and for the above example, the employed gravitational acceleration g is over 10^{10} times the normal one on the earth.

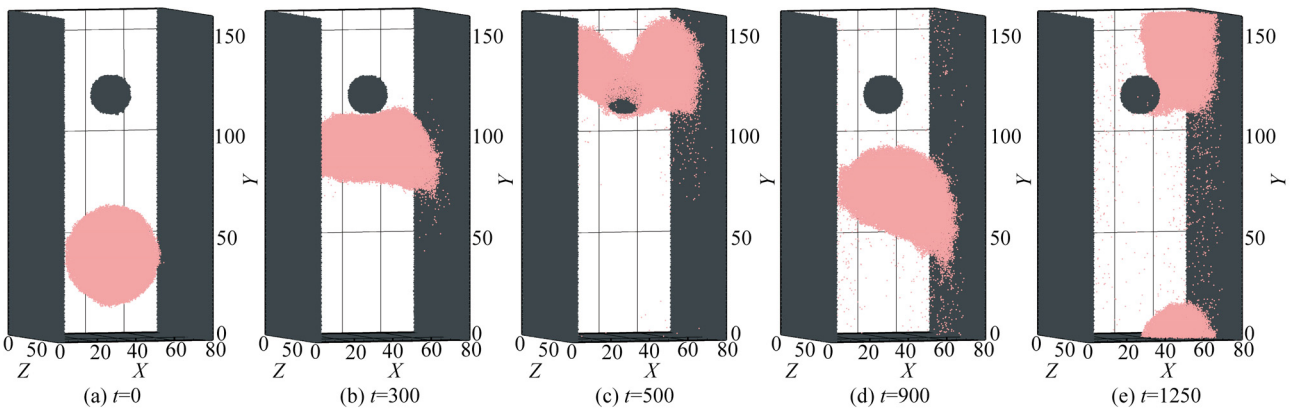

Figure 8 Snapshots of gas-liquid-solid MD simulation at different time. The red represents the gas molecule, the gray represents the solid molecule of the grain and walls, and the liquid molecules filling the remanent space are not represented.

5 Conclusions

The performance of modern GPUs far exceeds mainstream CPUs, and general purpose GPU (GPGPU) computation has received more and more attention on its high performance-to-price ratio. Using CUDA and MPI, we achieved parallel MD simulation on GPUs for multiphase nano- and micro-flow and got reasonable results. Comparisons show that GPU can achieve a 20—60-fold speedup over CPU, and the hybrid mode of coarse-grain parallel computing among GPUs and fine-grain parallel computing within GPUs can get nearly linear speedup. Therefore, it will be an attractive method for MD simulation.

So far, we have only considered the Lennard-Jones potential. Practical Coulomb potential and chemical bonds may complicate the MD algorithm and it is necessary to optimize algorithm further. Overall, however, MD simulation implemented on GPU has great potential.

The authors would like to acknowledge the support and help of NVIDIA (China) Co., Ltd.

1 Alder B J, Wainwright T. Molecular dynamics by electronic computers. In: Prigogine I, ed. Proceeding of International Symposium on Transport Processes in Statistical Mechanics, Brussels. New York: Interscience, Wiley, 1956, 97—131
2 Moore G E. Cramming more components onto integrated circuits. Electronics, 1965, 38(8): 114—117
3 NVIDIA. NVIDIA CUDA Compute Unified Device Architecture Programming Guide Version 2.0. 2008.
4 Belleman R G, Bédorf J, Zwart S F P. High performance direct gravitational N-body simulations on graphics processing units II: An implementation in CUDA. New Astron, 2008, 13(2): 103—112[DOI]
5 Tölke J. Implementation of a Lattice Boltzmann kernel using the Compute Unified Device Architecture developed by nVIDIA. Comp Vis Sci, 2008, DOI: 10.1007/s00791-008-0120-2
6 van Meel J A, Arnold A, Frenkel D, Zwart S F P, Belleman R G. Harvesting graphics power for MD simulations. Mol Simulat, 2008, 34(3): 259—266[DOI]
7 Anderson J A, Lorenz C D, Travesset A. General purpose molecular dynamics simulations fully implemented on graphics processing units. J Comp Phys, 2008, 227(10): 5342—5359[DOI]
8 Liem S Y, Brown D, Clarke J H R. Investigation of the homogeneous-shear nonequilibrium-molecular-dynamics method. Phys Rev A, 1992, 45(6): 3706—3713[DOI]
9 Frenkel D, Smit B. Understanding Molecular Simulation: From Algorithms to Applications. Orlando: Academic Press, 1996. 443
10 Shankar P N, Deshpande M D. Fluid mechanics in the driven cavity. Annu Rev Fluid Mech, 2000, 32(1): 93—136[DOI]
11 Albensoeder S, Kuhlmann H C. Accurate three-dimensional lid-driven cavity flow. J Comput Phys, 2005, 206(2): 536—558[DOI]
12 Hou S, Zou Q, Chen S, Doolen G, Cogley A C. Simulation of cavity flow by the Lattice Boltzmann method. J Comput Phys, 1995, 118(2): 329—347[DOI]
13 Nie X, Robbins M O, Chen S. Resolving singular forces in cavity flow: Multiscale modeling from atomic to millimeter scales. Phys Rev Lett, 2006, 96(13): 134501—134504[DOI]
14 Qian T, Wang X-P. Driven cavity flow: from molecular dynamics to continuum hydrodynamics. Multiscale Model Simul, 2005, 3(4): 749—763[DOI]
15 Koplik J, Banavar J R. Corner flow in the sliding plate problem. Phys Fluids, 1995, 7(12): 3118—3125[DOI]
16 Gogotsi Y, Libera J A, Guvenc-Yazicioglu A, Megaridis C M. *In situ* multiphase fluid experiments in hydrothermal carbon nanotubes. Appl Phys Lett, 2001, 79(7): 1021—1023[DOI]
17 Sushko N, Cieplak M. Motion of grains, droplets, and bubbles in fluid-filled nanopores. Phys Rev E, 2001, 64(21): 021061—0210618
18 Chen F, Ge W, Wang L, Li J. Numerical study on gas-liquid nano-flows with pseudo-particle modeling and soft-particle molecular dynamics simulation. Microfluid Nanofluid, 2008, 5(5): 639—653
19 Matsumoto M, Matsuura T. Molecular dynamics simulation of a rising bubble. Mol Simulat, 2004, 30(13—15): 853—859[DOI]
20 Matsumoto M, Tanaka K. Nano bubble——Size dependence of surface tension and inside pressure. Fluid Dynam Res, 2008, 40(7—8): 546—553[DOI]
21 Nagayama G, Tsuruta T, Cheng P. Molecular dynamics simulation on bubble formation in a nanochannel. Int J Heat Mass Tran, 2006, 49(23-24): 4437—4443[DOI]
22 Park S H, Weng J G, Tien C L. A molecular dynamics study on surface tension of microbubbles. Int J Heat Mass Tran, 2001, 44(10): 1849—1856[DOI]
23 Nguyen A V, Evans G M. Attachment interaction between air bubbles and particles in froth flotation. Exp Therm Fluid Sci, 2004, 28(5): 381—385[DOI]
24 Omota F, Dimian A C, Bliek A. Adhesion of solid particles to gas bubbles. Part 2: Experimental. Chem Eng Sci, 2006, 61(2): 835—844[DOI]