Hindawi
Scientific Programming
Volume 2018, Article ID 1298313, 12 pages
https://doi.org/10.1155/2018/1298313

Research Article

Scalable Parallel Algorithm of Multiple-Relaxation-Time Lattice Boltzmann Method with Large Eddy Simulation on Multi-GPUs

Lei Xu, Anping Song, and Wu Zhang

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China

Correspondence should be addressed to Wu Zhang; wzhang@shu.edu.cn

Received 6 October 2017; Revised 15 January 2018; Accepted 1 February 2018; Published 1 April 2018

Academic Editor: Frank Hannig

The lattice Boltzmann method (LBM) has become an attractive and promising approach in computational fluid dynamics (CFD). In this paper, parallel algorithm of D3Q19 multi-relaxation-time LBM with large eddy simulation (LES) is presented to simulate 3D flow past a sphere using multi-GPUs (graphic processing units). In order to deal with complex boundary, the judgement method of boundary lattice for complex boundary is devised. The 3D domain decomposition method is applied to improve the scalability for cluster, and the overlapping mode is introduced to hide the communication time by dividing the subdomain into two parts: inner part and outer part. Numerical results show good agreement with literature and the 12 Kepler K20M GPUs perform about 5100 million lattice updates per second, which indicates considerable scalability.

1. Introduction

Driven by the market demand for real-time, high-definition 3D graphics at processing large graphics data sets, graphics processing unit (GPU) has been developed for rending tasks. Due to its high computational power and the emergence of compute unified device architecture (CUDA) [1], GPU has drawn much attention to accelerate nongraphics computing [2–5].

In the recent two decades, the lattice Boltzmann method (LBM) has been an increasingly popular numerical method of CFD derived from kinetic theory for flow computations. There are several variations of LBM including lattice Bhatnagar-Gross-Krook (LBGK) [6] or single-relaxation-time LBM (SRT-LBM) [7], entropic LBM (ELBM) [8], two-relaxation-time LBM (TRT-LBM) [9], and multiple-relaxation-time LBM (MRT-LBM) [10, 11]. In these methods, MRT-LBM can improve stability and give accurate results in solving higher Reynolds number flow simulations [12]. Moreover, LBM is especially suitable for parallel computing because of its inherent parallelism.

Large eddy simulation (LES) is a mathematical model for turbulence applied in CFD [13]. It has become an effective way to simulate physical flow. In order to capture the turbulence flow physics, Hou et al. combine SRT-LBM with Smagorinsky model [14], and Krafczyk et al. incorporated the Smagorinsky model into MRT-LBM [15]. The improved MRT-LBM with LES still remains the inherent parallelism. In this paper, the parallel algorithm of MRT-LBM-LES on multi-GPUs is studied.

Due to algorithmic simplicity and suitability for parallel computing, numerous attempts of LBM on homogeneous and heterogeneous clusters have been reported. Before the advent of GPU, many researches have been carried out on LBM to obtain high performance and scalability. Kandhai et al. [16] presented different kind of domain decomposition method and parallel strategies about D2Q9 SRT-LBM. Velivelli and Bryden [17] examined cache optimization for two-dimensional LBM in both serial and parallel implements. Pan et al. [18] investigated five domain decomposition methods and the parallel efficiency of simulating single and multiphase flow in porous medium systems using D3Q15 LBM on various parallel computing platforms. Wu and Shao [19] simulated the lid-driven cavity flow using MRT-LBM compared with SRT-LBM by parallel implementation. Schepke et al. [20] presented blocked parallel implementation of D2Q9 and D3Q19 SRT-LBM; they divided computational domain into subdomains to reduce the MPI communication time. Since GPU has many computational units and the localized communication mode of each lattice well matches the data-parallel characteristic of GPU, Tölke [21] used shared memory to propagate parts of particle distribution functions (PDFs), avoiding costly misaligned access to the global memory on a single GPU, and Habich et al. [22] extended this strategy from D2Q9 to D3Q19 LBM. Due to the improvement of NVIDIA hardware, it is not necessary to use shared memory. Obrecht et al. [23] found out that the cost of a misaligned access was nearly equal to that caused by global memory access; they presented two schemes, a split one and a revise one, without using shared memory. Zhou et al. [24] simulated flows with curved boundaries; they gave detailed implementation to deal with curved boundaries on D2Q9 SRT-LBM. In order to simulate large-scale simulations, single GPU cannot afford the computational requirement, More schemes on multi-GPUs were presented. Xian and Takayuki [25] introduced the overlapping technique between computation and communication to hide the communication time in two streams simultaneously based on 3D domain decomposition. Based on presented work, Hong et al. [3] presented schemes to simulate 3D cavity flow based on D3Q19 MRT-LBM and evaluated the performance of various schemes. Mawson and Revell [26] examined and analysed previous optimization strategies; the implementation including misaligned accesses to the global memory is found to be more efficient. Ye et al. [27] proposed the hybrid CPU-GPU computing environment of D3Q19 ELBM on single-node multi-GPU platform; their method not only takes advantage of GPU but also exploits the computing power according to model-based optimal load balancing. Tran et al. developed high performance parallelization of LBM on a GPU by reducing the overheads associated with the uncoalesced memory accesses and improving the cache locality using the tiling optimization with the data layout change [28].

The rest of the paper is organized as follows. First, a short overview of MRT-LBM with LES is given. In Section 3, the multi-GPUs architecture is introduced. Next, MRT-LBM with LES parallel strategy is presented in Section 4. Then the numerical experiment is shown in Section 5. Finally, the major conclusions are summarized in Section 6.

2. MRT-LBM with LES

The LBM based on lattice Boltzmann equation (LBE) can be described as [12]

$$\mathbf{f}\left(\mathbf{x}+\mathbf{e}_i\delta_t, t+\delta_t\right)-\mathbf{f}(\mathbf{x},t)=\mathbf{\Omega}, \tag{1}$$

where $\mathbf{f}(\mathbf{x},t)$ represents the PDFs at site $\mathbf{x}$ and time t, $\mathbf{\Omega}$ denotes the collision operator, the left side δ_t is the time step, and $\mathbf{e}_i$ expresses the particle velocity in the ith direction.

In the present study, the D3Q19 (see Figure 1) lattice model and MRT collision operator are adopted. The discrete velocities $\mathbf{e}_i$ according to D3Q19 can be defined by

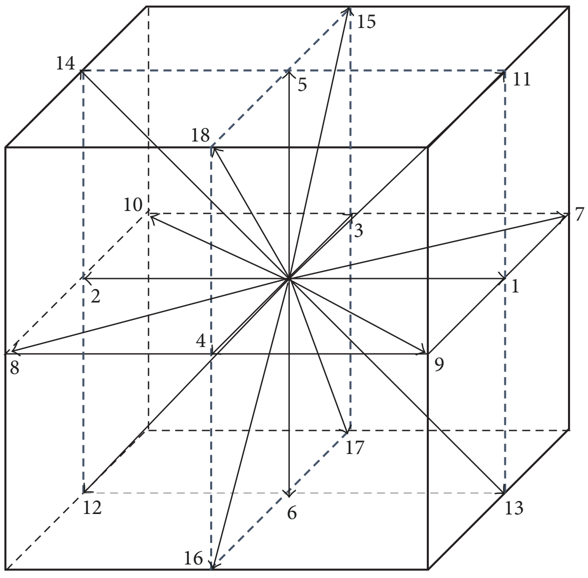


Figure 1: D3Q19.

$$\mathbf{e}_i = \begin{cases} (0,0,0), & i=0, \\ (\pm1,0,0)\,c, (0,\pm1,0)\,c, (0,0,\pm1)\,c, & i=1\sim6, \\ (\pm1,\pm1,0)\,c, (0,\pm1,\pm1)\,c, (\pm1,0,\pm1)\,c, & i=7\sim18. \end{cases} \tag{2}$$

Here, the lattice speed $c=\delta_x/\delta_t$, where δ_x is the lattice size.

In the MRT model, $\mathbf{\Omega}$ can be expressed as

$$\mathbf{\Omega} = -\mathbf{M}^{-1}\cdot\mathbf{S}\cdot\left[\mathbf{m}(\mathbf{x},t)-\mathbf{m}^{\text{eq}}(\mathbf{x},t)\right], \tag{3}$$

where the transformation matrix $\mathbf{M}$ linearly transforms the PDFs $\mathbf{f}$ and equilibrium distribution functions (EDFs) $\mathbf{f}^{\text{eq}}$ to the velocity moments $\mathbf{m}$ and $\mathbf{m}^{\text{eq}}$, respectively. $\mathbf{S}$ is the relaxation diagonal matrix and

$$\mathbf{S} = \operatorname{diag}\left(0, s_1, s_2, 0, s_4, 0, s_4, 0, s_4, s_9, s_{10}, s_9, s_{10}, s_{13}, s_{13}, s_{13}, s_{16}, s_{16}, s_{16}\right), \tag{4}$$

where s_9 and s_{13} satisfy

$$\nu = \frac{1}{3}\left(\frac{1}{s_9}-\frac{1}{2}\right) = \frac{1}{3}\left(\frac{1}{s_{13}}-\frac{1}{2}\right), \tag{5}$$

and the remaining relaxation frequencies can be referenced by [29]. The EDF $f_i^{\text{eq}}(x,t)$ can be obtained by

$$f_i^{\text{eq}} = \rho\omega_i\left[1+\frac{\mathbf{e}_i\cdot\mathbf{u}}{c_s^2}+\frac{(\mathbf{e}_i\cdot\mathbf{u})^2}{2c_s^4}-\frac{\mathbf{u}^2}{2c_s^2}\right], \tag{6}$$

where c_s is the speed of sound: $c_s=1/\sqrt{3}$; and the coefficients of ω_i are weighting factor listed as follows:

$$\omega_i = \begin{cases} \dfrac{1}{3}, & i=0, \\ \dfrac{1}{18}, & i=1\sim6, \\ \dfrac{1}{36}, & i=7\sim18. \end{cases} \tag{7}$$

The macroscopic density and velocity based on PDFs can be written as

$$\rho = \sum_i f_i,$$
$$\mathbf{u} = \frac{1}{\rho}\sum_i f_i \mathbf{e}_i. \tag{8}$$

Based on (1), LBM can be expressed by two steps: collision (see (9)) and propagation (see (10)).

$$\mathbf{f}^{+}(\mathbf{x},t) = \mathbf{f}(\mathbf{x},t) + \Omega, \tag{9}$$

$$\mathbf{f}\left(\mathbf{x} + \mathbf{e}_i\delta_t, t + \delta_t\right) = \mathbf{f}^{+}(\mathbf{x},t), \tag{10}$$

where $\mathbf{f}_i^{+}(\mathbf{x},t)$ expresses the postcollision state of the PDF.

With the purpose of simulating turbulent flow at high Reynolds numbers, Krafczyk et al. incorporated MRT-LBM with LES [15]. The total viscosity ν_{total} can be expressed as the sum of ν and turbulent eddy viscosity ν_t.

$$\nu_{\text{total}} = \nu + \nu_t,$$
$$\nu_t = \left(C_s\delta_x\right)^2 \left|S_{\alpha\beta}\right|, \tag{11}$$

where C_s is the Smagorinsky constant and it is set to be 0.16 [15]; $S_{\alpha\beta}$ $(\alpha,\beta \in x,y,z)$ in Cartesian coordinates is the strain rate tensor; that is, $S_{\alpha\beta}$ can be computed directly from nonequilibrium moments ($m_k^{\text{neq}}(\mathbf{x},t) = m_k(\mathbf{x},t) - m_k^{\text{eq}}(\mathbf{x},t)$):

$$S_{xx} \approx -\frac{1}{38\rho\Delta t}\left(s_1 m_1^{\text{neq}} + 19 s_9 m_9^{\text{neq}}\right),$$
$$S_{yy,zz} \approx -\frac{1}{76\rho\Delta t}\left[2s_1 m_1^{\text{neq}} - 19 s_9\left(m_9^{\text{neq}} \mp 3m_{11}^{\text{neq}}\right)\right], \tag{12}$$
$$S_{xy,yz,xz} \approx -\frac{3s_9 m_{13,14,15}^{\text{neq}}}{2\rho\Delta t}.$$

Then, the corresponding relaxation frequencies can be modified in terms of (5).

In this present work, a nonequilibrium extrapolation method is used for the nonslip static straight wall [30] and a unified boundary treatment for curved boundary is applied to model nonslip boundary condition with curved boundaries [31]. In Figure 2, the solid circle is the solid node and the hollow circle is the fluid node, and a curved wall separates the solid nodes from the fluid nodes. The lattice node is denoted as $\mathbf{x}_f$ on the fluid side of the boundary and x_b on the solid side. $\mathbf{x}_{ff}$ is the adjacent fluid node of $\mathbf{x}_f$. The filled small rhombus on the boundary wall x_w denotes the intersections of the wall with various lattice links. The boundary velocity at x_w is denoted as $\mathbf{u_w}$. Next, this curved boundary scheme is given by

$$f_{\bar{\alpha}}\left(\mathbf{x}_f, t+\delta_t\right) = \frac{1}{1+q}\left[(1-q)\, f_{\alpha}^{+}\left(\mathbf{x}_{ff}, t\right)\right.$$
$$\left. + q f_{\alpha}^{+}\left(\mathbf{x}_f, t\right) + q f_{\bar{\alpha}}^{+}\left(\mathbf{x}_f, t\right)\right], \tag{13}$$

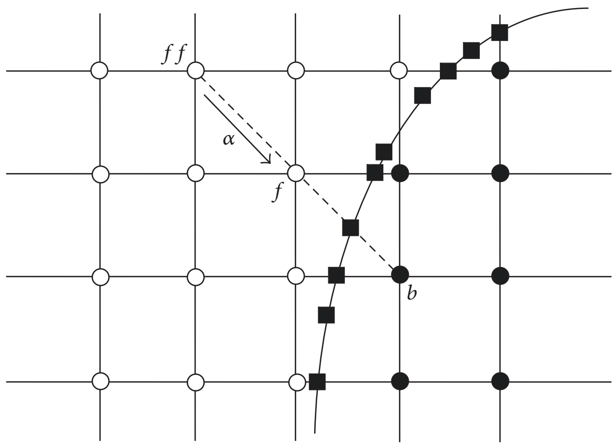


Figure 2: Illustration of curved boundary.

where $\bar{\alpha}$ is the opposite direction of α and q is written by

$$q = \frac{\left|x_f - x_w\right|}{\left|x_f - x_b\right|}, \quad 0 \le q \le 1. \tag{14}$$

When dealing with curved boundary, it is necessary to judge the lattice type (solid, fluid, or boundary lattice) and the value of q. Algorithm 1 shows the judgement of lattice type and how to obtain q. The ray method is employed to generate the lattice type. When the lattice is in the geometry, the number of intersections is even.

3. Multi-GPUs Architecture

In the earliest days, computers only consisted of central processing units (CPUs). However, the many-core accelerators (GPU and Intel Xeon Phi) are becoming prevalent in the past decade. Over time, GPUs have become more and more powerful and generalized for general-purpose parallel computing tasks with excellent performance and high efficiency [1, 32]. The GPU architecture consists of a scalable array of Streaming Multiprocessors (SMX). Each SMX supports the concurrent execution of hundreds of threads so that thousands of threads can be executed concurrently on a single GPU. CUDA employs Single Instruction Multiple Thread (SIMT) architecture to manage and execute threads in a group of 32 called warps. The threads in a warp perform the same instruction at the same time; each thread in a warp has its own instruction address counter and register state and executes the current instruction on its own data. In this paper, the K20 based on GK110 architecture is tested. It contains 13 SMXs, 1.25 MB L2 cache, 5 GB memory size, and six 64-bit memory controllers. Each SMX includes 192 single-precision CUDA cores, 64 double-precision units, 32 special function units (SFU), and 32 load/store units (LD/ST). Each SMX consists of 4 warp schedulers and 8 instruction dispatchers which can issue 4 warps and execute concurrently. The theoretical peak of single-precision performance is 3.5 TFLOPS, and the memory bandwidth achieves 208 GB/s.

(1) Generate the solid lattice based on grid file using ray method.
(2) Obtain the curved boundary lattice according to the solid lattice and (2).
(3) Compute the distances $|x_f - x_w|$ and $|x_f - x_b|$; get the value of q.

Algorithm 1: Judgement of the lattice type.

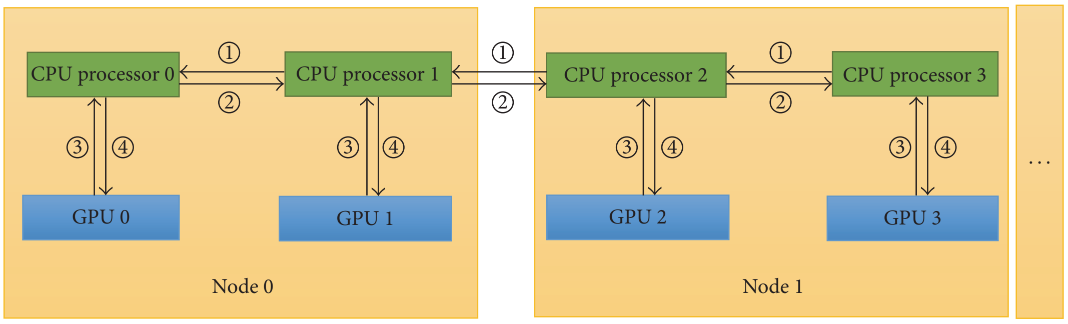


Figure 3: Message passing on multi-GPUs: ① MPI_ISend, ② MPI_IRecv, ③ cudaMemcpyDeviceToHost, and ④ cudaMemcpyHostToDevice.

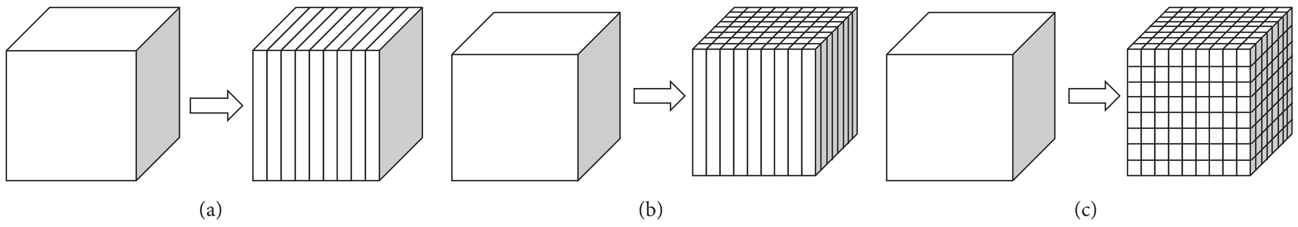


Figure 4: (a) 1D domain decomposition, (b) 2D domain decomposition, and (c) 3D domain decomposition.

As a result of the dimension of the problems treated with the LBM, a single piece of GPU cannot deal with the problems and high computing power and large memory space are required. It has to be solved on multi-GPUs. As shown in Figure 3, the CPU processor transfers data with GPU device by cudaMemcpyHostToDevice and cudaMemcpyDeviceToHost. The CPU processors communicate with MPI_ISend and MPI_IRecv.

4. MRT-LBM with LES on Multi-GPUs

4.1. Domain Decomposition Method and Data Exchange. Owing to the local nature of LBM, it is a better choice to adopt domain decomposition method that partitions the flow domain into subdomain. As for three-dimensional flow simulation, there are 1D, 2D, and 3D (see Figure 4) domain decomposition methods [20], in which the fluid domain can be divided along one, two, or three directions. It can be seen from [20, 25] that choosing the 3D one is the best option when running on cluster. In the 3D one, each MPI process deals with a part of computational domain, and the process can be denoted as a triple (i, j, k) according to the MPI process rank id. The triple can be obtained by

$$\begin{aligned} i &= \operatorname{mod}\left(\operatorname{mod}\left(id, px \times py\right), px\right), \\ j &= \frac{\operatorname{mod}\left(id, px \times py\right)}{px}, \\ k &= \frac{id}{px \times py}, \end{aligned} \tag{15}$$

where px and py denote the amount of domain decomposition along x and y, respectively. Then, the computational part along x direction of process id can be got by

$$\begin{aligned} x\text{Beg} &= i \times \frac{\text{num}X}{px} + \text{minInt}\left(i, \operatorname{mod}\left(\text{num}X, px\right)\right), \\ x\text{End} &= \begin{cases} x\text{Beg} + \dfrac{\text{num}X}{px} - 1, & \operatorname{mod}\left(\text{num}X, px\right) \le i, \\ x\text{Beg} + \dfrac{\text{num}X}{px}, & \operatorname{mod}\left(\text{num}X, px\right) > i, \end{cases} \end{aligned} \tag{16}$$

where num$X$ is the lattice node count along the x direction. xBeg is the start lattice node along x direction, and xEnd is the end one. The range of y and z directions can be calculated similarly.

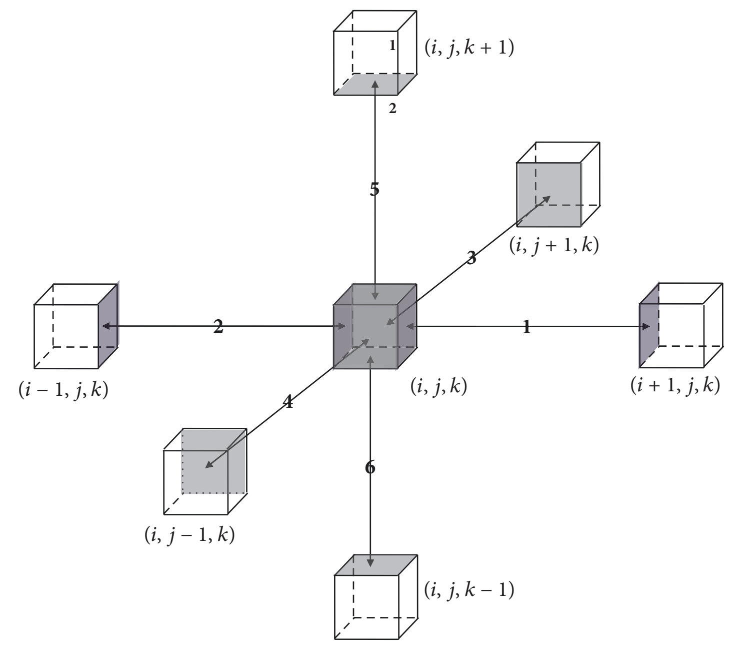


Figure 5: Surfaces needed to be exchanged.

(1) Execute propagation operation while reading data from global memory.
(2) Deal with boundary conditions including curved boundary and nonslip static straight wall.
(3) Calculate macroscopic quantities (density ρ and velocity $\mathbf{u}$) according to (8).
(4) Do the collision operation.

Algorithm 2: Out-of-place scheme.

Based on (9) and (10), the only communication occurs in the propagation step. After collision, each MPI process needs to exchange the interface between neighboring MPI processes. In 3D, each MPI process (i, j, k) has to send data of 6 surfaces and 12 edges and receive data of 6 surfaces (see Figure 5) and 12 edges (see Figure 6). When transferring data f_i^+, there is no need to transfer all lattice data of f_i^+; only the expected data in propagation operation are required along the direction. For example, only the data of f_1^+, f_7^+, f_9^+, f_{11}^+, and f_{13}^+ are expected when transferring data from MPI process (i, j, k) to $(i + 1, j, k)$.

4.2. Data Structure. Group of threads into warps is relevant not only to computation but also to global memory access [32]. GPU coalesces global memory loads and stores accessed by threads of a warp to minimize DRAM bandwidth. In the GPU implement of LBM, it is a simple and efficient way to assign a lattice node to a thread. Each thread needs to store $\mathbf{f}$, $\mathbf{f}^+$, ρ, $\mathbf{u}$, and lattice style. Figures 7 and 8 illustrate the memory layout of Array of Structures (AoS) and Structure of Arrays (SoA), respectively. Storing the data of PDFs in AoS format on the GPU and executing an operation that only requires f_0 would result in a 18/19 loss of bandwidth as other PDFs are implicitly loaded in each 32-byte segment or 128-byte cache line. In the meantime, the AoS format would also waste the L2 cache space on other unneeded PDFs values. Storing the data in SoA format can make full use of GPU memory bandwidth which provides coalesced memory access. In order to obtain coalescence of global memory access and efficient global memory utilization, it is better to choose SoA type rather than AoS type.

4.3. Framework of GPU Implement of MRT-LBM-LES. There are two propagation schemes: out-of-place propagation (see Figure 9), in which the collision step is carried out before the propagation step, and in-place propagation (see Figure 10), in which the collision step is executed after the propagation step [23]. Hong et al. [3] compare the two schemes. It can be seen that out-of-place propagation which we take is more efficient than in-place propagation on multi-GPUs. Based on [3], the reverse scheme can improve the performance greatly. The out-of-place scheme can be illustrated as Algorithm 2.

Based on out-of-place scheme, the framework of GPU implement of MRT-LBM with LES can be described as Algorithm 3 and Figure 11. Compared with [3], it is not necessary to execute the collision step in the lattices stored in the buffers when the MPI communication occurs after collision step. In order to improve parallel efficiency, overlapping mode

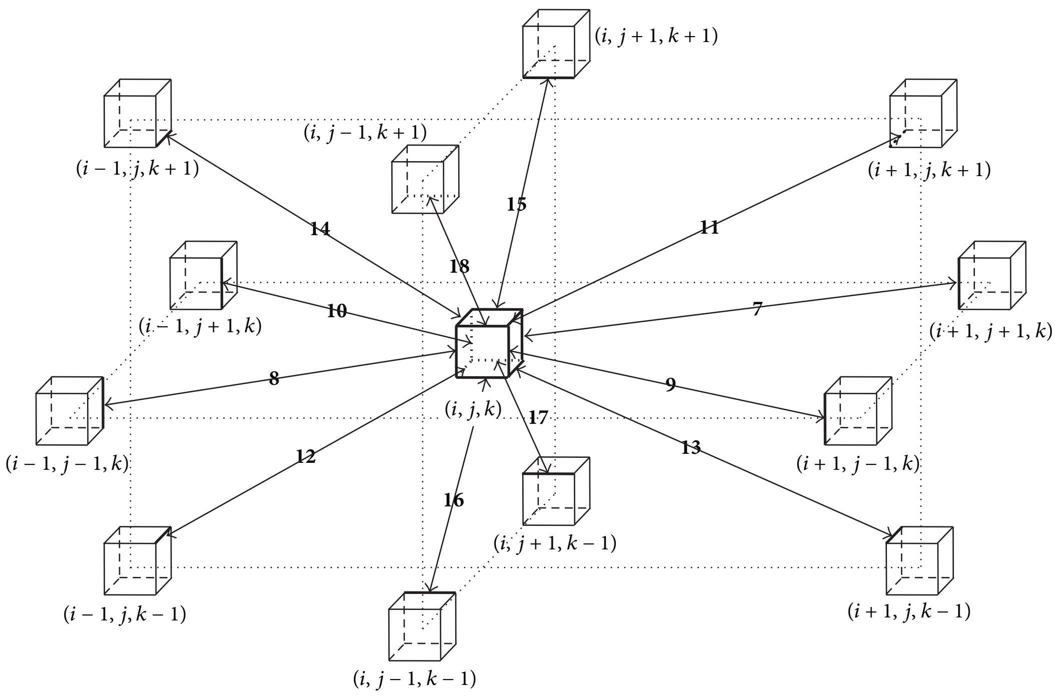
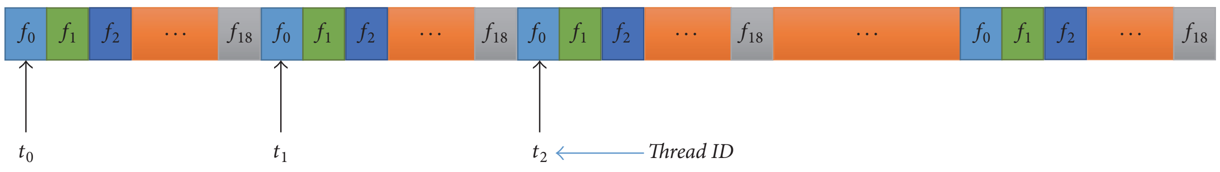
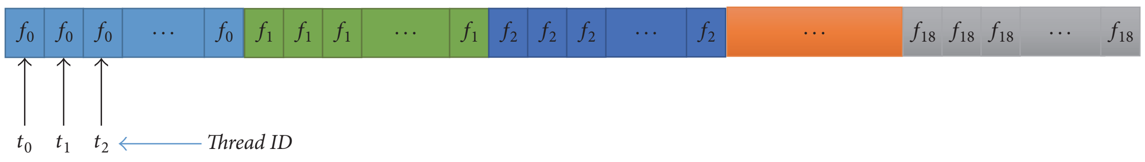


Figure 6: Edges needed to be exchanged.



f0 f1 f2 ··· f18 f0 f1 f2 ··· f18 f0 f1 f2 ··· f18 ··· f0 f1 f2 ··· f18
t0
t1
t2
Thread ID

Figure 7: AoS memory layout.



f0 f0 f0 ··· f0 f1 f1 f1 ··· f1 f2 f2 f2 ··· f2 ··· f18 f18 f18 ··· f18
t0
t1
t2
Thread ID

Figure 8: SoA memory layout.

is applied to hide the communication time. The subdomain is divided into two parts: inner part and outer part [25]. The inner part does not need to be exchanged with other MPI processes. But PDFs of outer part of subdomain are needed by neighbor MPI processes after propagation. In our implement, two streams are designed: one is for inner part and the other is for boundary part (see Figure 12). When executing the computation of inner part, the MPI communication part of boundary part is processed simultaneously.

5. Numerical Results and Discussion

The numerical experiments are tested on a three-node GPU cluster; each node is equipped with four NVIDIA Kepler K20M GPU devices and two Intel Xeon E5-2680 CPUs. Each node is connected by 56 Gb FDR InfiniBand network.

5.1. Validation of Flow Past a Sphere. In this section, the simulation of the 3D incompressible flow past a sphere with a constant velocity profile, $u = U_\infty = \{0.2Ma, 0, 0\}$, is a validation test. The sphere radius $D = 20$ is taken. Figure 13 shows the flow geometry, coordinate system, and computational domain. The length of the computational domain is $25.6D$, the width is $6.4D$, and the height is $6.4D$. The lattice scale is $512 \times 128 \times 128$.

Figure 14 shows the sphere surface grid, which is used for the test. Figures 15 and 16 show the lattices around sphere on the slice of XOZ and YOZ plane, respectively.

```
(1) Read grid file.
(2) Domain decomposition.
(3) Memory allocation on host and device.
(4) Initialization.
(5) Copy data from host to device.
(6) Judgement of the lattice style.
(7) Iterative computation until satisfying convergence condition on GPUs.
     (1) Read data from global memory and propagation.
     (2) Deal with boundary condition.
     (3) Calculate macroscopic quantities.
     (4) Collision and write data back.
     (5) Data exchange of outer subdomain using MPI.
(8) Copy data from device to host.
(9) Gather and write data back to host memory.
```

Algorithm 3: Multi-GPUs of MRT-LBM-LES.

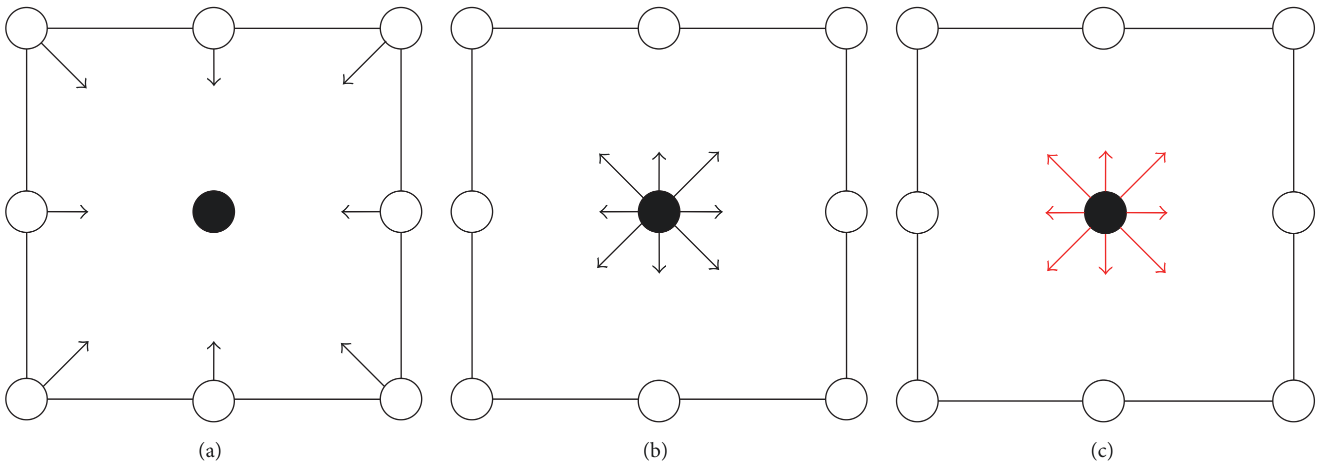


Figure 9: Out-of-place propagation: (a) prepropagation, (b) precollision and postpropagation, and (c) postcollision.

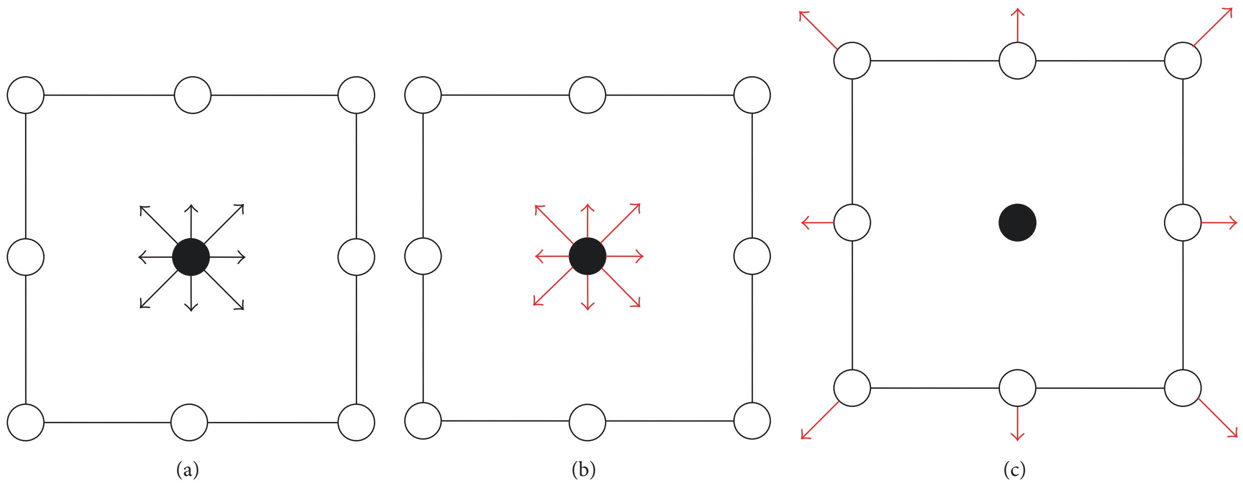


Figure 10: In-place propagation: (a) precollision, (b) postcollision, and (c) postpropagation.

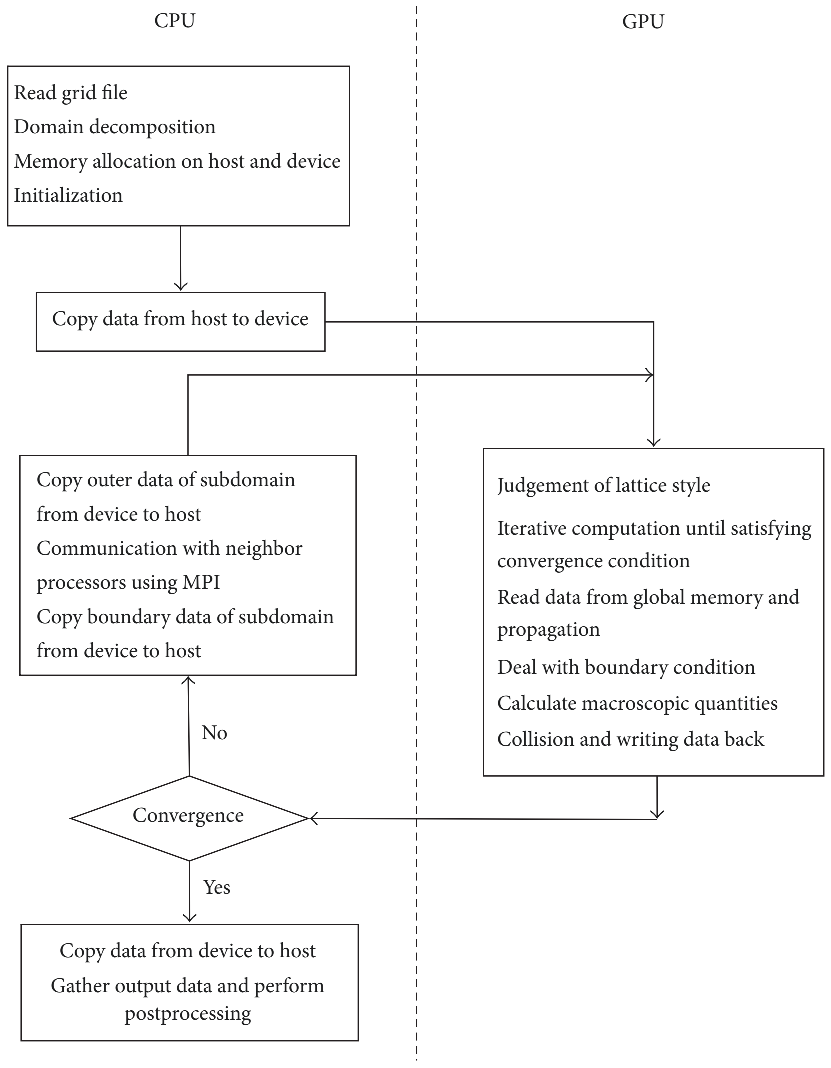

CPU
GPU
Read grid file
Domain decomposition
Memory allocation on host and device
Initialization
Copy data from host to device
Copy outer data of subdomain from device to host
Communication with neighbor processors using MPI
Copy boundary data of subdomain from device to host
Judgement of lattice style
Iterative computation until satisfying convergence condition
Read data from global memory and propagation
Deal with boundary condition
Calculate macroscopic quantities
Collision and writing data back
No
Convergence
Yes
Copy data from device to host
Gather output data and perform postprocessing

FIGURE 11: Framework of multi-GPUs implement of MRT-LBM-LES.

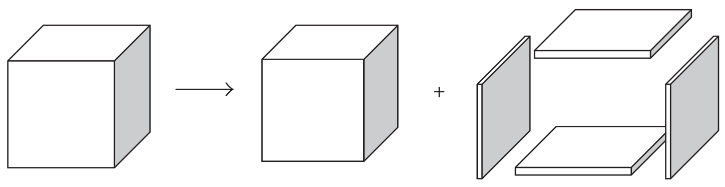


FIGURE 12: Subdomain is divided into inner and boundary parts.

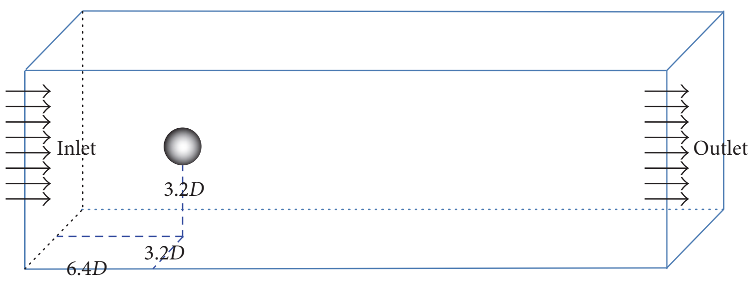


FIGURE 13: Flow geometry, coordinate system, and computational domain.

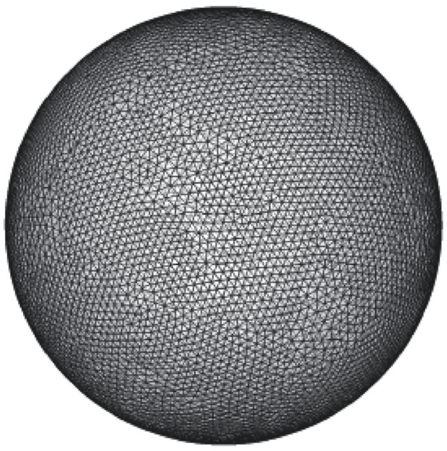


Figure 14: Sphere surface grid.

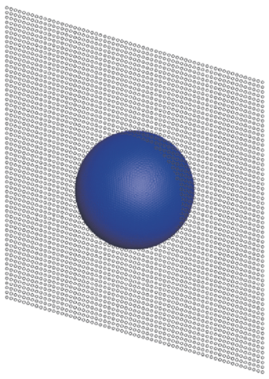


Figure 15: Lattices on the slice of XOZ plane.

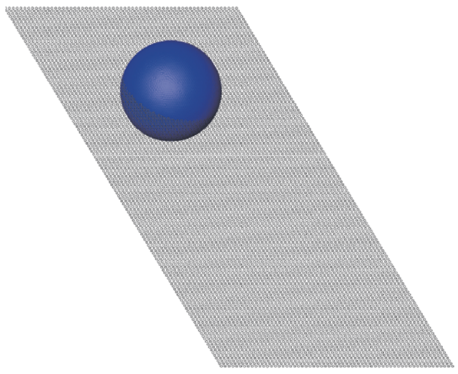


Figure 16: Lattices on the slice of YOZ plane.

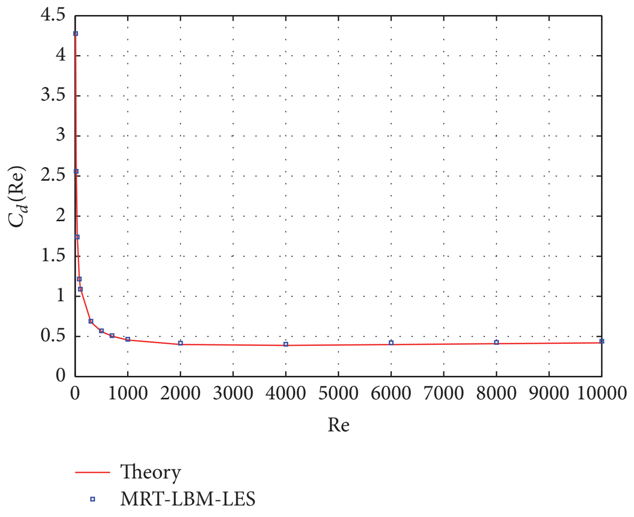


Figure 17: Drag force coefficient.

The drag force on the sphere F_d is computed with the momentum-exchange method [33] and the drag force coefficient C_d [34] can be expressed by

$$C_d = \frac{F_d}{(1/2)\,\rho U_\infty^2 D}. \tag{17}$$

Figure 17 shows C_d compared with the theoretical value. Figures 18 and 19 describe the vortical structure when Re = 1000 and 3700, respectively.

5.2. Parallel Performance. The parallel performance is examined in terms of MLUPS (million lattice updates per second), and MLUPS can be obtained by

$$\text{MLUPS} = \frac{\text{lattice scale} \times \text{number of iterations} \times 10^{-6}}{\text{total iterative time}}. \tag{18}$$

Figures 20 and 21 show the strong scaling and weak scaling, respectively. According to Figure 20, the ideal MLUPS is proportional to the number of GPUs and the present MLUPS of our algorithm is nearly equal to the ideal MLUPS. The weak scaling test from Figure 21 shows that the MLUPS on each GPU remains almost the same when the grid size is assigned to each GPU equally regardless of the number of GPUs. The computation of multi-GPUs of MRT-LBM-LES has considerable strong and weak scalability when fixed grid size is $512 \times 128 \times 128$. $X \times Y \times Z$ indicates that the x direction of the fluid domain is divided into X parts, y direction is splitted into Y part, and z direction is decomposed as Z parts. In order to compare overlapping and nonoverlapping modes, Figure 22 shows that the nonoverlapping mode can obtain 66% improvement and improve the scalability when our present algorithm is executed on 12 GPUs.

6. Conclusion

In this paper, a scalable parallel algorithm of D3Q19 MRT-LBM with LES on 12 NVIDIA Kepler K20M GPUs platform is presented to simulate three-dimensional flow past a sphere. The numerical results of flow past a sphere are compared with the literature and show good agreements. In order to simulate complex geometry, a method to judge lattice style is also described. The 3D domain decomposition method and overlapping mode are employed to improve the scalability for

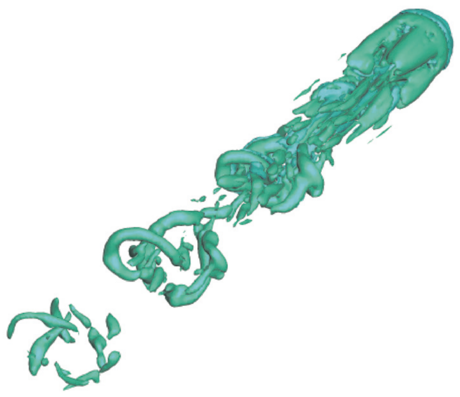


Figure 18: Vortical structure: Re = 1000.

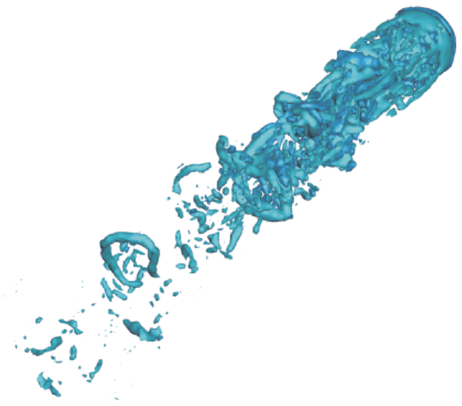


Figure 19: Vortical structure: Re = 3700.

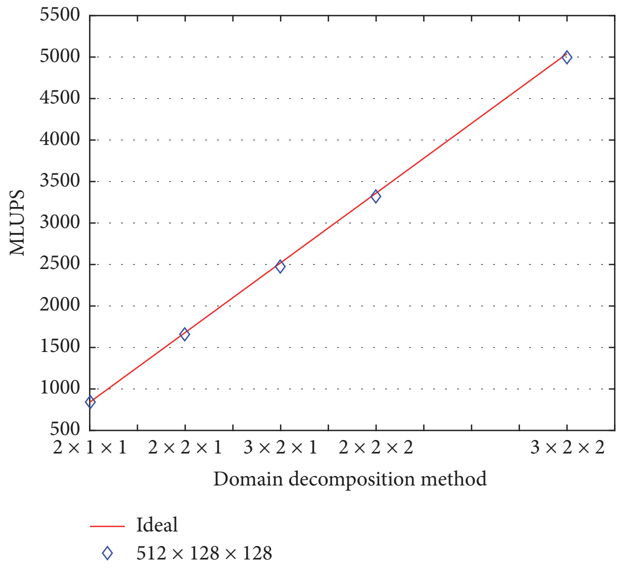


Figure 20: MLUPS of overlapping multi-GPUs with different domain decomposition method.

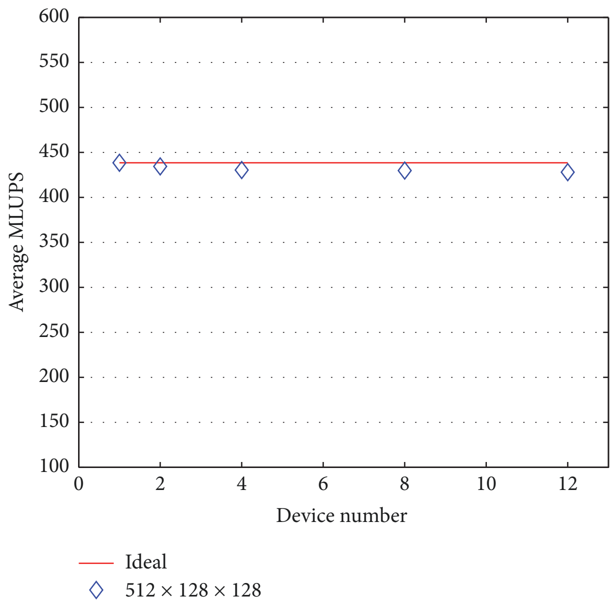


Figure 21: Weak scaling of overlapping multi-GPUs.

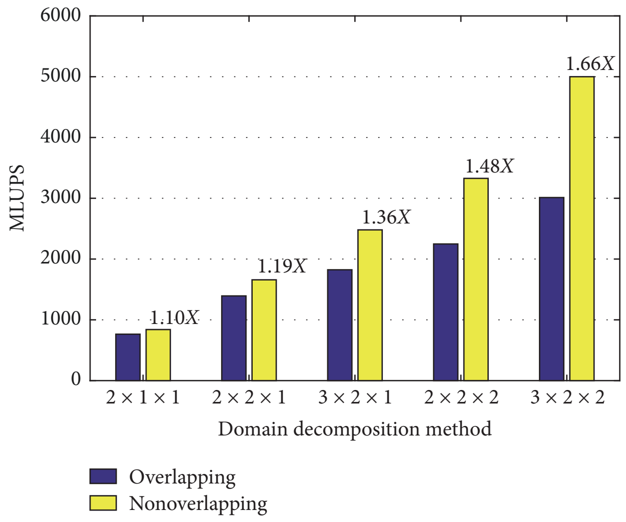


Figure 22: Performance comparison of overlapping and nonoverlapping with different domain decomposition method.

heterogeneous cluster, which can obtain 66% improvement. Numerical results show that the present algorithm can perform 5100 MLUPS on 12 Kepler K20M GPUs, which indicates that the present algorithm is efficient and scalable.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors thank all members of the high performance computing group at Shanghai University for their good advice. This research is supported by the National Natural Science

Foundation of China (no. 91630206 and no. 11601315) and Shanghai Sailing Program (no. 16YF1404000).

References

[1] nVidia., CUDA C Programming Guide 7.0 (2015).

[2] M. Frederic, A. Cheik, and P. Roman, "Auto-tuned Krylov methods on cluster of graphics processing unit," *International Journal of Computer Mathematics*, vol. 92, no. 6, pp. 1222–1250, 2015.

[3] P.-Y. Hong, L.-M. Huang, L.-S. Lin, and C.-A. Lin, "Scalable multi-relaxation-time lattice Boltzmann simulations on multi-GPU cluster," *Computers & Fluids. An International Journal*, vol. 110, pp. 1–8, 2015.

[4] X. Chen, L. Ren, Y. Wang, and H. Yang, "GPU-accelerated sparse LU factorization for circuit simulation with performance modeling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 786–795, 2015.

[5] G. Felix, H. Andreas, S. L. Ole, K. Fabian, and N. Uwe, "GPU-accelerated sparse matrix-matrix multiplication by iterative row merging," *SIAM Journal on Scientific Computing*, vol. 37, no. 1, pp. C54–C71, 2015.

[6] Y. H. Qian, D. D'Humières, and P. Lallemand, "Lattice BGK models for Navier-Stokes equation," *EPL (Europhysics Letters)*, vol. 17, no. 6, p. 479, 1992.

[7] S. Chen, H. Chen, D. Martnez, and W. Matthaeus, "Lattice Boltzmann model for simulation of magnetohydrodynamics," *Physical Review Letters*, vol. 67, no. 27, pp. 3776–3779, 1991.

[8] S. Ansumali and I. V. Karlin, "Entropy function approach to the Lattice Boltzmann method," *Journal of Statistical Physics*, vol. 107, no. 1-2, pp. 291–308, 2002.

[9] I. Ginzburg, F. Verhaeghe, and D. d'Humières, "Two-relaxation-time lattice Boltzmann scheme: about parametrization, velocity, pressure and mixed boundary conditions," *Communications in Computational Physics*, vol. 3, no. 2, pp. 427–478, 2008.

[10] D. D'Humieres, "Generalized lattice Boltzmann equation, Rarefied Gas Dynamics: Theory and Simulations," *Astronautics and Aeronautics*, vol. 159, pp. 450–458, 1992.

[11] P. Lallemand and L.-S. Luo, "Theory of the lattice Boltzmann method: dispersion, dissipation, isotropy, Galilean invariance, and stability," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 61, no. 6, part A, pp. 6546–6562, 2000.

[12] L.-S. Luo, W. Liao, X. Chen, Y. Peng, and W. Zhang, "Numerics of the lattice Boltzmann method: Effects of collision models on the lattice Boltzmann simulations," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 83, no. 5, Article ID 056710, 2011.

[13] J. Smagorinsky, "General circulation experiments with the primitive equations: I. The basic equations," *Monthly Weather Review*, vol. 91, pp. 99–164, 1963.

[14] S. Hou, J. Sterling, S. Chen, and G. D. Doolen, "A lattice Boltzmann subgrid model for high Reynolds number flows," *Fields Institute Communications*, vol. 6, pp. 151–166, 1963.

[15] M. Krafczyk, J. Tölke, and L.-S. Luo, "Large-eddy simulations with a multiple-relaxation-time LBE model," *International Journal of Modern Physics B*, vol. 17, no. 1-2, pp. 33–39, 2003.

[16] D. Kandhai, A. Koponen, A. G. Hoekstra, M. Kataja, J. Timonen, and P. M. A. Sloot, "Lattice-Boltzmann hydrodynamics on parallel systems," *Computer Physics Communications*, vol. 111, no. 1–3, pp. 14–26, 1999.

[17] A. C. Velivelli and K. M. Bryden, "A cache-efficient implementation of the lattice Boltzmann method for the two-dimensional diffusion equation," *Concurrency and Computation: Practice and Experience*, vol. 16, no. 14, pp. 1415–1432, 2004.

[18] C. Pan, J. F. Prins, and C. T. Miller, "A high-performance lattice Boltzmann implementation to model flow in porous media," *Computer Physics Communications*, vol. 158, no. 2, pp. 89–105, 2004.

[19] J.-S. Wu and Y.-L. Shao, "Simulation of lid-driven cavity flows by parallel lattice Boltzmann method using multi-relaxation-time scheme," *International Journal for Numerical Methods in Fluids*, vol. 46, no. 9, pp. 921–937, 2004.

[20] C. Schepke, N. Maillard, and P. O. A. Navaux, "Parallel lattice boltzmann method with blocked partitioning," *International Journal of Parallel Programming*, vol. 37, no. 6, pp. 593–611, 2009.

[21] J. Tölke, "Implementation of a Lattice Boltzmann kernel using the compute unified device architecture developed by nVIDIA," *Computing and Visualization in Science*, vol. 13, no. 1, pp. 29–39, 2010.

[22] J. Habich, T. Zeiser, G. Hager, and G. Wellein, "Performance analysis and optimization strategies for a D3Q19 lattice Boltzmann kernel on nVIDIA GPUs using CUDA," *Advances in Engineering Software*, vol. 42, no. 5, pp. 266–272, 2011.

[23] C. Obrecht, F. Kuznik, B. Tourancheau, and J.-J. Roux, "A new approach to the lattice Boltzmann method for graphics processing units," *Computers & Mathematics with Applications*, vol. 61, no. 12, pp. 3628–3638, 2011.

[24] H. Zhou, G. Mo, F. Wu, J. Zhao, M. Rui, and K. Cen, "GPU implementation of lattice Boltzmann method for flows with curved boundaries," *Computer Methods Applied Mechanics and Engineering*, vol. 225/228, pp. 65–73, 2012.

[25] W. Xian and A. Takayuki, "Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster," *Parallel Computing*, vol. 37, no. 9, pp. 521–535, 2011.

[26] M. J. Mawson and A. J. Revell, "Memory transfer optimization for a lattice Boltzmann solver on Kepler architecture nVidia GPUs," *Computer Physics Communications*, vol. 185, no. 10, pp. 2566–2574, 2014.

[27] Y. Ye, K. Li, Y. Wang, and T. Deng, "Parallel computation of entropic lattice Boltzmann method on hybrid CPU-GPU accelerated system," *Computers & Fluids. An International Journal*, vol. 110, pp. 114–121, 2015.

[28] N.-P. Tran, M. Lee, and S. Hong, "Performance Optimization of 3D Lattice Boltzmann Flow Solver on a GPU," *Scientific Programming*, vol. 2017, Article ID 1205892, 2017.

[29] D. D'Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, and L.-S. Luo, "Multiple-relaxation-time lattice Boltzmann models in three dimensions," *The Royal Society of London. Philosophical Transactions. Series A. Mathematical, Physical and Engineering Sciences*, vol. 360, no. 1792, pp. 437–451, 2002.

[30] Z.-L. Guo, C.-G. Zheng, and B.-C. Shi, "Non-equilibrium extrapolation method for velocity and pressure boundary conditions in the lattice boltzmann method," *Chinese Physics*, vol. 11, no. 4, pp. 366–374, 2002.

[31] D. Yu, R. Mei, and W. Shyy, "A unified boundary treatment in lattice Boltzmann method," in *Proceedings of the 41st Aerospace Sciences Meeting and Exhibit 2003*, Reno, Nevada, USA, January 2003.

[32] J. Cheng, M. Grossman, and T. McKercher, *Professional CUDA C Programming*, John Wiley Sons, 2014.

[33] R. Mei, D. Yu, W. Shyy, and L.-S. Luo, “Force evaluation in the lattice Boltzmann method involving curved geometry,” *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 65, no. 4, Article ID 041203, p. 041203/14, 2002.

[34] M. Stiebler, M. Krafczyk, S. Freudiger, and M. Geier, “Lattice Boltzmann large eddy simulation of subcritical flows around a sphere on non-uniform grids,” *Computers & Mathematics with Applications. An International Journal*, vol. 61, no. 12, pp. 3475–3484, 2011.