Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 3913968, 13 pages
https://doi.org/10.1155/2020/3913968

Research Article

The Immersed Boundary-Lattice Boltzmann Method Parallel Model for Fluid-Structure Interaction on Heterogeneous Platforms

Zhixiang Liu[1], **Huichao Liu**[1], **Dongmei Huang**[1,2] and **Liping Zhou**[3]

[1]*College of Information Technology, Shanghai Ocean University, Shanghai 201306, China*
[2]*Shanghai University of Electric Power, Shanghai 200090, China*
[3]*School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China*

Correspondence should be addressed to Dongmei Huang; dmhuang@shou.edu.cn

Received 14 May 2020; Revised 27 July 2020; Accepted 5 August 2020; Published 27 August 2020

Academic Editor: Efstratios Tzirtzilakis

Immersed boundary-lattice Boltzmann method (IB-LBM) has become a popular method for studying fluid-structure interaction (FSI) problems. However, the performance issues of the IB-LBM have to be considered when simulating the practical problems. The Graphics Processing Units (GPUs) from NVIDIA offer a possible solution for the parallel computing, while the CPU is a multicore processor that can also improve the parallel performance. This paper proposes a parallel algorithm for IB-LBM on a CPU-GPU heterogeneous platform, in which the CPU not only controls the launch of the kernel function but also performs calculations. According to the relatively local calculation characteristics of IB-LBM and the features of the heterogeneous platform, the flow field is divided into two parts: GPU computing domain and CPU computing domain. CUDA and OpenMP are used for parallel computing on the two computing domains, respectively. Since the calculation time is less than the data transmission time, a buffer is set at the junction of two computational domains. The size of the buffer determines the number of the evolution of the flow field before the data exchange. Therefore, the number of communications can be reduced by increasing buffer size. The performance of the method was investigated and analyzed using the traditional metric MFLUPS. The new algorithm is applied to the computational simulation of red blood cells (RBCs) in Poiseuille flow and through a microchannel.

1. Introduction

In the immersed boundary-lattice Boltzmann method (IB-LBM), the flow field is solved by the lattice Boltzmann method (LBM), and the interaction between the fluid and the immersed object is solved by the immersed boundary method (IB) [1]. Since Feng and Michaelides [2] first successfully applied the IB-LBM to simulate the motion of rigid particles, IB-LBM has become a popular method for studying the fluid-structure interaction (FSI) problems. Eshghinejadfard et al. [3] used IB-LBM to research turbulent channel flow in the presence of spherical particles. Cheng et al. [4] developed an IB-LBM for simulating a multiphase flow with solid particles and liquid drops. The IB-LBM is also widely used to studying red blood cells (RBCs) flow in the blood, such as the interaction between plasma flow and RBCs movement [5] and the aggregation and deformation of red blood cells in an ultrasound field [6]. Simulations of practical problems of FSI, like blood flow, demand considerable computational resources [7], which will greatly reduce the simulation time.

The architecture of the Graphics Processing Units (GPUs) offers high floating performance and memory to processor chip bandwidth [8]. The LBM simulation has the characteristics of regular, static, and local calculations [9], which can be greatly accelerated by means of GPU [10, 11]. Shang et al. [12] used MPI to show that LBM has good parallel acceleration performance. By proposing two new implementation methods, LBM-Ghost and LBM-Swap, Valero-Lara [13] reduced the huge memory requirements when simulating LBM on the GPU and proposed [14] three

different parallel approaches for grid refinement based on a multidomain decomposition. The homogeneous GPU and heterogeneous CPU + GPU approaches [15] for mesh refinement based on Multidomain and irregular meshing could achieve better performance. In terms of IB-LBM's parallel computing, Valero-Lara et al. [16] proposed a numerical approach parallelizing the IB-LBM on both GPUs and a heterogeneous GPU-Multicore platform. Boroni et al. [17] presented a fully parallel GPU implementation of IB-LBM. The GPU kernel performs fluid boundary interactions and accelerated each code execution. Recently, Beny and Latt [9] proposed a new method for the implementation of IB-LBM on GPUs, which allows handling a substantially larger immersed surfaces. Other parallel methods, compiler directive open multiprocessing (OpenMP) [18] and distributed memory clusters [19], can achieve multicore parallel processing with shared memory on the CPU, for instance, a nonuniform staggered Cartesian grid approach [20] reducing considerably the complexity of the communication and memory management between different refined levels for mesh refinement.

In this paper, a new parallel algorithm for IB-LBM is proposed on CPU-GPU heterogeneous platform. When the GPU is performing simulation, the CPU is in a waiting state, which causes a waste of computing resources. The new algorithm avoids this problem. On the one hand, the flow field is divided into CPU domains and GPU domains, which are calculated by the Compute Unified Device Architecture (CUDA) and OpenMP, respectively. According to the calculation process of IB-LBM, after the flow field has evolved once, data is exchanged at the junction of the two calculation domains. On the other hand, the calculation time is shorter than the communication time, so buffers were introduced to reduce the number of data communications. The size of the buffer determines the number of times the flow field evolves before data is exchanged. Setting a buffer of proper size can reduce data communication and improve computing performance. As expected, this strategy greatly improved calculation efficiency without affecting the calculation accuracy of the IB-LBM.

The remainder of this paper is organized as follows. Section 2 introduces IB-LBM. Section 3 illustrates the parallel algorithm on CPU-GPU heterogeneous platform that makes full use of computing resources. Section 4 shows the benchmark tests and analyzes the parallel performance. RBCs in Poiseuille flow and through a microchannel are simulated in Section 5. Finally, Section 6 concludes this study.

2. Immersed Boundary-Lattice Boltzmann Method

IB-LBM is a flexible and efficient calculation in the case of simulated fluid flow through moving or complex boundaries. The computational area of the IB-LBM consists of two types of lattice points: the Lagrangian points and the Eulerian points (see Figure 1).

The Euler point represents the entire flow field, and the Lagrangian point illustrates the boundary of the immersed

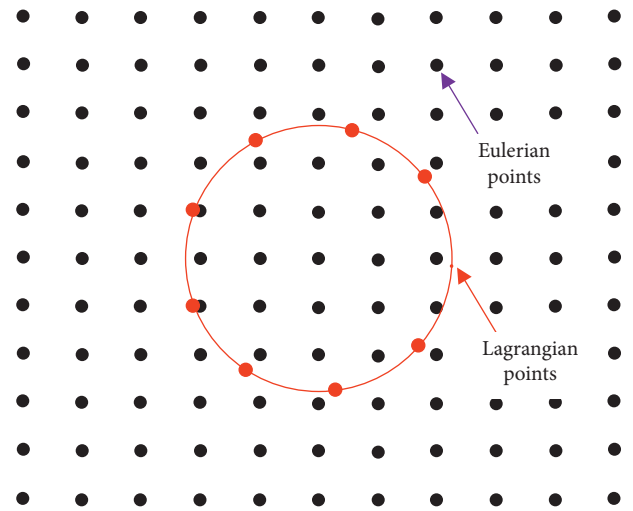

Figure 1: Two-dimensional flow field boundary structure of IB-LBM mode.

object. The body force of the object is calculated by the deformation of the boundary, and then, it is dispersed to the Euler point of the flow field using the Dirac delta function. Finally, the macroscopic density and velocity of the flow field are solved.

The DnQm LBGK is the most widely used model for solving the flow field in the IB-LBM. n is the spatial dimension, and m represents the number of the discrete velocity [21, 22]. In this paper, the most commonly used model, D2Q9 (see Figure 2), is employed, and the corresponding discrete velocity $\mathbf{e}_i$ is

$$e_i \begin{cases} 0, & i = 0, \\ \left(\cos\left[(i-1)\frac{\pi}{2}\right], \sin\left[(i-1)\frac{\pi}{2}\right]\right)c, & i = 1,2,3,4, \\ \sqrt{2}\left(\cos\left[(i-5)\frac{\pi}{2}+\frac{\pi}{4}\right], \sin\left[(i-5)\frac{\pi}{2}+\frac{\pi}{4}\right]\right)c, & i = 5,6,7,8, \end{cases} \tag{1}$$

where $c = \Delta x/\Delta t$ is the lattice velocity and Δx is the lattice space.

The LBGK model can be expressed as

$$f_i(x + e_i\Delta t, t + \Delta t) - f_i(x,t) = -\frac{1}{\tau}\left[f_i(x,t) - f_i^{\mathrm{eq}}(x,t)\right] + \Delta t F_i, \tag{2}$$

where $f_i(x,t)$ is the particle distribution function, x is current lattice location and t is current lattice time, Δt is the lattice time step, and τ is the relaxation time which is determined by the fluid viscosity coefficient v of the fluid:

$$v = \left(\tau - \frac{1}{2}\right)c_s^2\Delta t, \tag{3}$$

where c_s is the speed of sound. The corresponding $f_i^{\mathrm{eq}}(x,t)$ is the equilibrium distribution function. It can be written as

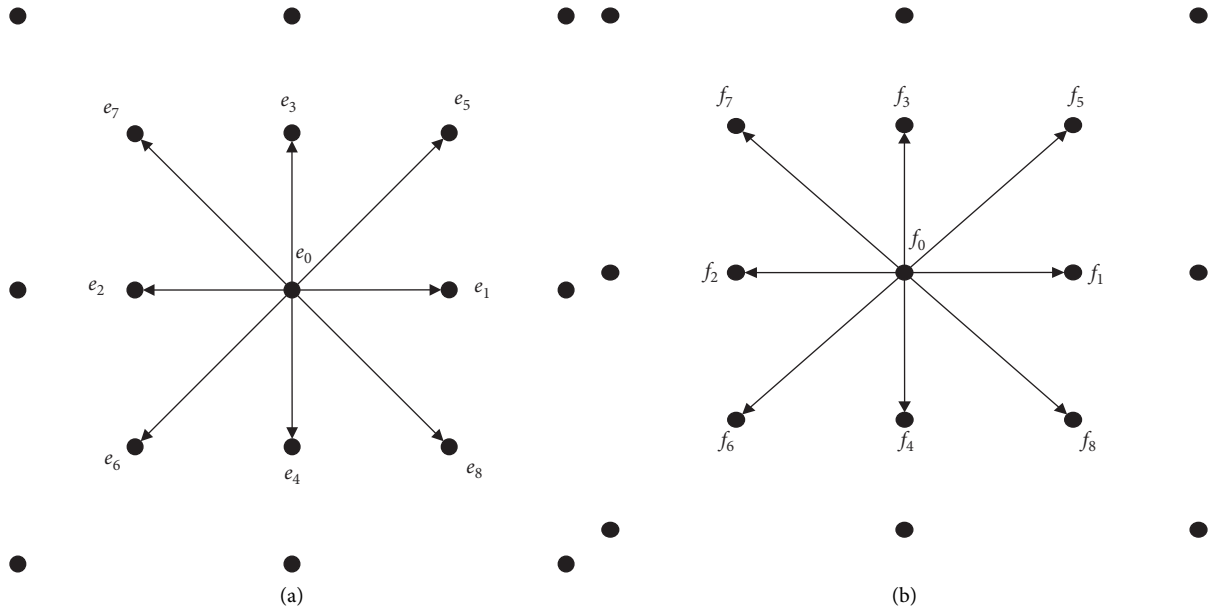

Figure 2: Velocity vectors and function vectors in D2Q9 lattice model. (a) Velocity vectors. (b) Distribution function.

$$f_i^{\mathrm{eq}}(x,t) = \rho\omega_i\left[1 + \frac{e_i u}{c_s^2} + \frac{(e_i u)^2}{2c_s^4} - \frac{|u|^2}{2c_s^2}\right], \tag{4}$$

where u is the macroscopic velocity at the current lattice point and ω_i is the weight factor of the discrete velocity. In the D2Q9 model, the value of ω_i can be written as

$$\omega_i \begin{cases} \dfrac{4}{9}, & e_i^2 = 0, \\ \dfrac{1}{9}, & e_i^2 = c^2, \\ \dfrac{1}{36}, & e_i^2 = 2c^2. \end{cases} \tag{5}$$

For boundary conditions, this paper adopts the non-equilibrium extrapolation method [23], which can be expressed as

$$f_i(x_b,t) = f_i^{\mathrm{eq}}\left(\rho\left(x_f,t\right),u_b\right) + \left[f_i\left(x_f,t\right) - f_i^{\mathrm{eq}}\left(x_f,t\right)\right]. \tag{6}$$

where $\rho(x_f,t)$ represents the density of the adjacent lattice point x_f of the boundary lattice point, x_b, and u_b is the velocity of the boundary lattice point x_b.

In the IB-LBM, the key problem is the calculation and dispersion of forces at the Lagrangian point. The penalty force method [2] is a way to calculate resilience. The basic idea of the penal force method is to apply Hook's law of elastic deformation. Regarding the boundary of the immersed object as an elastic boundary composed of Lagrangian points, the object is deformed by force, and the position of the Lagrangian point is moved to the position of the reference point (see Figure 3).

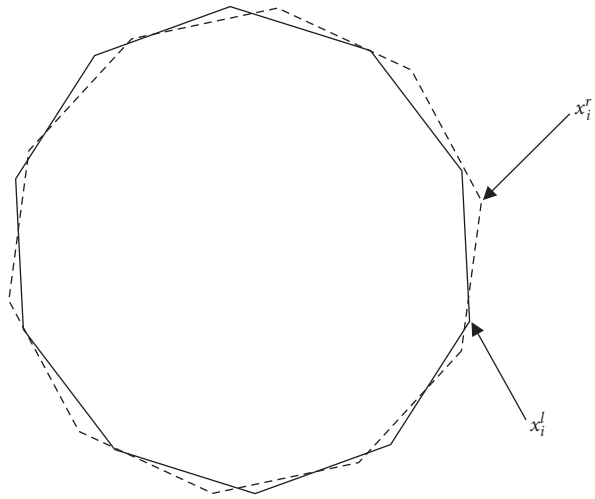

Figure 3: Boundary point and reference point of object deformation.

Restoring force due to the deformation is calculated by Hooke's law:

$$F_b = -\varepsilon \cdot \left(x_i^l - x_i^r\right). \tag{7}$$

where x_i^l is the Lagrangian point, x_i^r is the reference point after the corresponding Lagrangian point is moved, and ε is the stiffness coefficient; the value is 0.03.

In the calculation process, the position of the Lagrangian point is constant. After each lattice time step, the position of the reference points moved. The body force $F(x,t)$ in the flow field is obtained by interpolation of the restoring force F_b at the boundary point to the lattice point of the flow field. It is calculated by

$$F(X,t) = \int_{\Gamma} F_b(s,t)\delta(X - x_i(t))\mathrm{d}s = \sum_{i}^{n} F_b \cdot \delta(X - x_i(t))\mathrm{d}s_i, \quad (8)$$

where X is the Euler point in the flow field, x_i is the i-th Lagrangian point on the boundary, F_b is the corresponding restoring force, n is the total number of Lagrangian points on the boundary, and $\mathrm{d}s_i$ is the length of boundary element. $\delta(X - x_i(t))$ is a Dirac function that can be used to interpolate the restoring force at the boundary point to the Euler point of the flow field. Dirac functions have a variety of expressions. The interpolation equation used in this paper is as follows:

$$\delta(x) = \begin{cases} 1 - |x|, & 0 \le |x| \le 1, \\ 0, & 1 \le |x|. \end{cases} \quad (9)$$

After interpolating the restoring force at the boundary point of the object to the Euler point of the flow field, the body force on the Euler point needs to be discretized according to the DnQm model [24]. It can be expressed by

$$F_i(X,t) = \left(1 - \frac{1}{2\tau}\right)\omega_i \left[\frac{\mathbf{e}_i - u}{c_s^2} + \frac{(\mathbf{e}_i \cdot u)}{c_s^4}\mathbf{e}_i\right] \cdot F(X,t). \quad (10)$$

According to the calculation of the above equation, the macroscopic density and velocity of each lattice point can be written as

$$\rho = \sum_{i=0}^{8} f_i, \qquad \rho u = \sum_{i=0}^{8} \mathbf{e}_i f_i + \frac{1}{2} F \Delta t. \quad (11)$$

After getting the macro speed of each lattice point, the speed of the boundary point is calculated by interpolation. Then, the deformation displacement of the boundary point is obtained.

The calculation steps for the IB-LBM can be summarized as follows (see Figure 4).

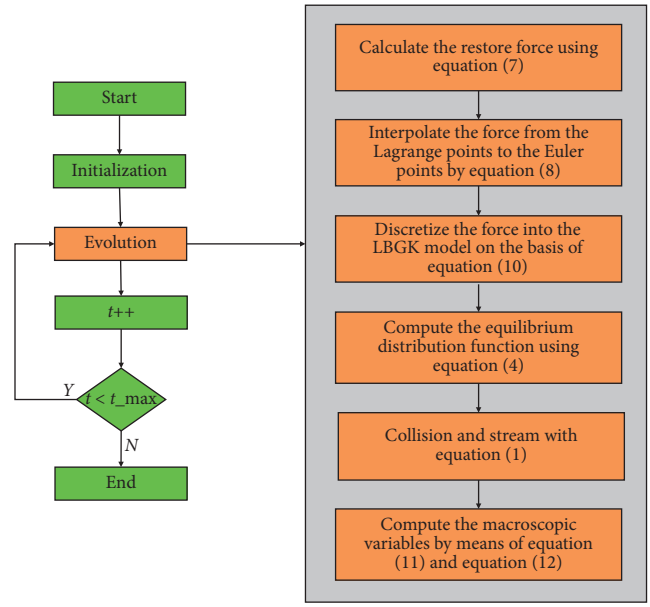

Figure 4: Flow chart of IB-LBM for penalty method.

3. IB-LBM Parallel Model on Heterogeneous Platforms

For the development of parallel algorithm of IB-LBM, it is important to make the most of computing resources on CPU-GPU heterogeneous platforms. The traditional parallel algorithm on CPU-GPU heterogeneous platforms is implemented by dividing the algorithm into GPU part, which is responsible for computing the LBM part, and CPU part, which consists of executing the IBM part. However, the amount of calculation between IBM and LBM is inconsistent, which may cause the GPU or CPU to wait for the other. On the other hand, after the evolution of each time step, some GPU and CPU data need to be transferred, which leads to extra time consumption. This paper attempts to provide a new approach to make full use of the computing resources of both GPU and CPU. The new method improves the traditional algorithms.

A GPU contains multiple multiprocessors, and each multiprocessor contains multiple processors, called cores. Therefore, in the single-instruction multiple-data mode, the GPU can process a large amount of data at the same time. The solution of IB-LBM to fluid-structure interaction problems is relatively local. This feature has a high agreement with the GPU architecture and can achieve good performance results. To simulate IB-LBM on the GPU, the flow field needs to be divided into grid and blocks.

In the structure of the GPU, threads are grouped by blocks, and all blocks form a grid, which can make a three-dimensional shape. In the case of three-dimensional, the thread in a block can be determined by the thread coordinates: threadId$x.x$, threadIdx.y, and threadId$x.z$. Similarly, the position of the block can be determined by the coordinates of the block: blockId$x.x$, blockId$x.y$, and blockId$x.z$. This method is also applicable to two-dimensional and one-dimensional situations.

As shown in Figure 5, the flow field is mapped to a two-dimensional grid, which contains $n \times m$ blocks; a block contains $i \times j$ threads. Each thread calculates a lattice point in the flow field.

The index of the array can be obtained by

$$\begin{aligned} \text{index\_}x &= \text{thread Id}x.x + \text{block Id}x.x \cdot \text{blockDim}.x, \\ \text{index\_}y &= \text{thread Id}x.y + \text{block Id}x.y \cdot \text{blockDim}.y,. \end{aligned} \quad (12)$$

where blockDim.x and blockDim.y are the dimensions of the block and the number of threads in the X and Y directions, respectively. threadId$x.x$, threadId$x.y$, blockId$x.x$, and blockId$x.y$ are the coordinates of the thread and block, respectively. To divide the entire flow field into multiple two-dimensional subregions of a specified size, that is blocks, the

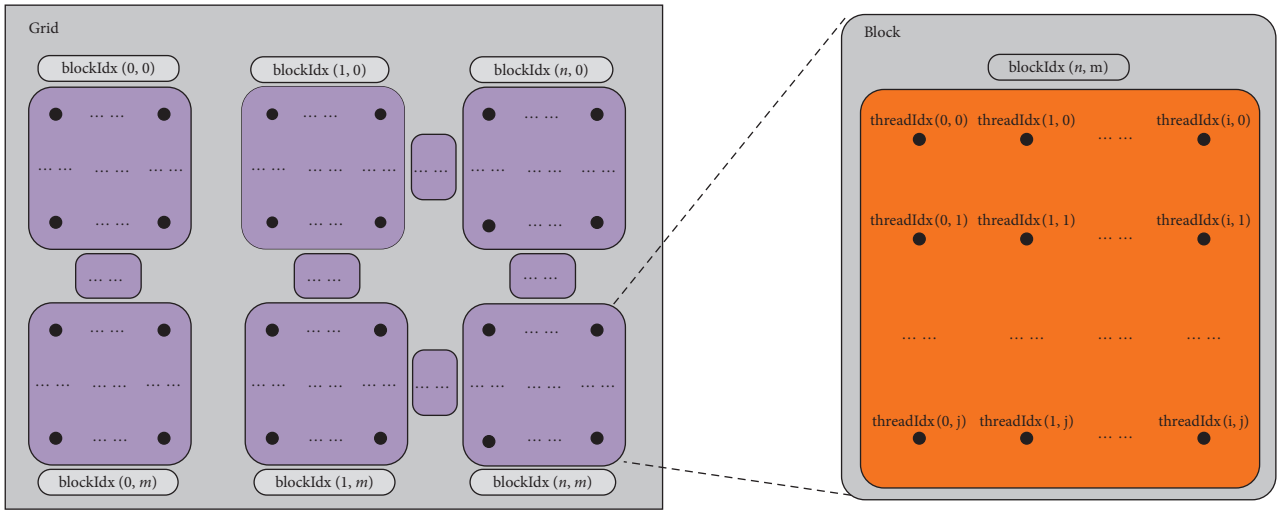

Figure 5: The flow field is divided according to the grid structure and block structure of the GPU.

total number of threads must be consistent with the total number of lattice points in the flow field, which can avoid unnecessary resource consumption and extra calculations.

Although the efficiency of IB-LBM simulation on the GPU has been greatly improved, it still caused a waste of computing resources. After the kernel function is launched, the calculations on the GPU are performed on the GPU. At the same time, the control of the program returns to the CPU, and the CPU is in a waiting state. This paper proposes an algorithm on CPU-GPU heterogeneous platform to avoid this situation. We divide the flow field into two parts: GPU computing domain and CPU computing domain. When control returns to the host, the CPU immediately performs IB-LBM simulation on the CPU computing domain. The specific execution steps are shown in Figure 6.

The launch time of the kernel function is very short and can be ignored. GPU and CPU can be seen as evolving at the same time on the CPU and GPU computing domain, respectively. Multiple kernel functions can be launched on the CPU. After the kernel functions are launched, they are queued up for execution on the GPU, and control is returned to the CPU. The CPU launches some kernel function of a time step and then performs a time step simulation on the CPU, so that, after the calculation on the CPU and the GPU is completed, a time step simulation of the entire flow field is completed. In order to ensure that the calculation has been completed before data transmission, GPU synchronization needs to be set. It is necessary to wait for the GPU-side calculation to end when the synchronization position is reached. The data exchange is completed on the CPU, so the CPU simulation does not need to set synchronization, and the CPU can achieve automatic synchronization. After synchronization is achieved, only a small amount of data needs to be exchanged, the density, velocity, and distribution function at the boundary of the calculation domain.

It can be seen from Figure 6 that the GPU and the CPU need to exchange data to ensure that the flow field can evolve correctly after the evolution of the flow field is performed once. While the data is being exchanged, both the GPU and the CPU need to wait. At the same time, frequent data exchange adds extra time consumption. In order to optimize these problems, this paper introduces the concept of a buffer at the junction of two computational domains (see Figure 7).

In Figure 7, the blue area is the buffer with the same number of lattice at the junction of the GPU and CPU computing domains. In this way, there is no need to exchange data after each evolution. After the introduction of the buffer, the corresponding number of evolutions can be executed according to the size of the buffer, and then data exchange can be performed. The red area in Figure 7 is the area that needs to be copied. The algorithm overwrites the value of CPU buffer-zone with the value of GPU copy-zone and then overwrites the value of GPU buffer-zone with the value of CPU copy-zone. In this way, the evolution of the CPU and GPU computing domains has been correctly designed. By introducing a buffer, the number of data exchanges and CPU-to-GPU communications is greatly reduced, and additional time consumption is reduced. The parallel algorithm on CPU-GPU heterogeneous platforms can be illustrated as Algorithm 1.

The IB-LBM simulation contains four different calculations: IBM part, the entire flow field, boundary processing, and buffer assignment. According to these four kinds of calculations, different grid divisions Grid_1, Grid_2, Grid_3, Grid_4 and different block divisions Block_1, Block_2, Block_3, and Block_4 are used in Algorithm 1. Proper division can make better use of the performance of the GPU. The GPU can start multiple kernel functions simultaneously and then execute them sequentially. The start-up time of the kernel function is extremely short and can be ignored. Therefore, Algorithm 1 approximates that the GPU and CPU start performing calculations at the same time.

The key problem of the algorithm proposed in this paper is how to divide the GPU computing domain and the CPU computing domain. Optimal performance can be achieved with appropriate mesh sizes in the two computational domains. The IB-LBM simulation with the same mesh amount is performed on the GPU and the CPU before the calculation

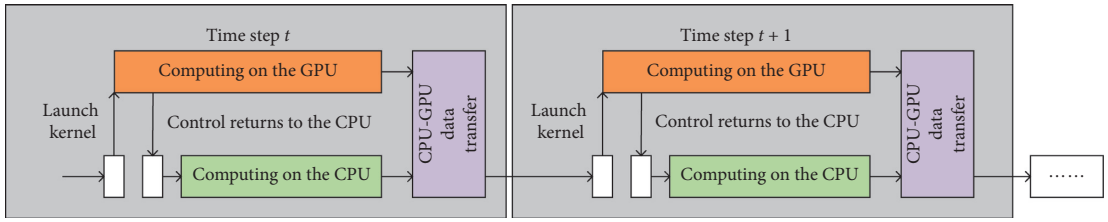

Figure 6: IB-LBM calculation process and data exchange process on heterogeneous platforms.

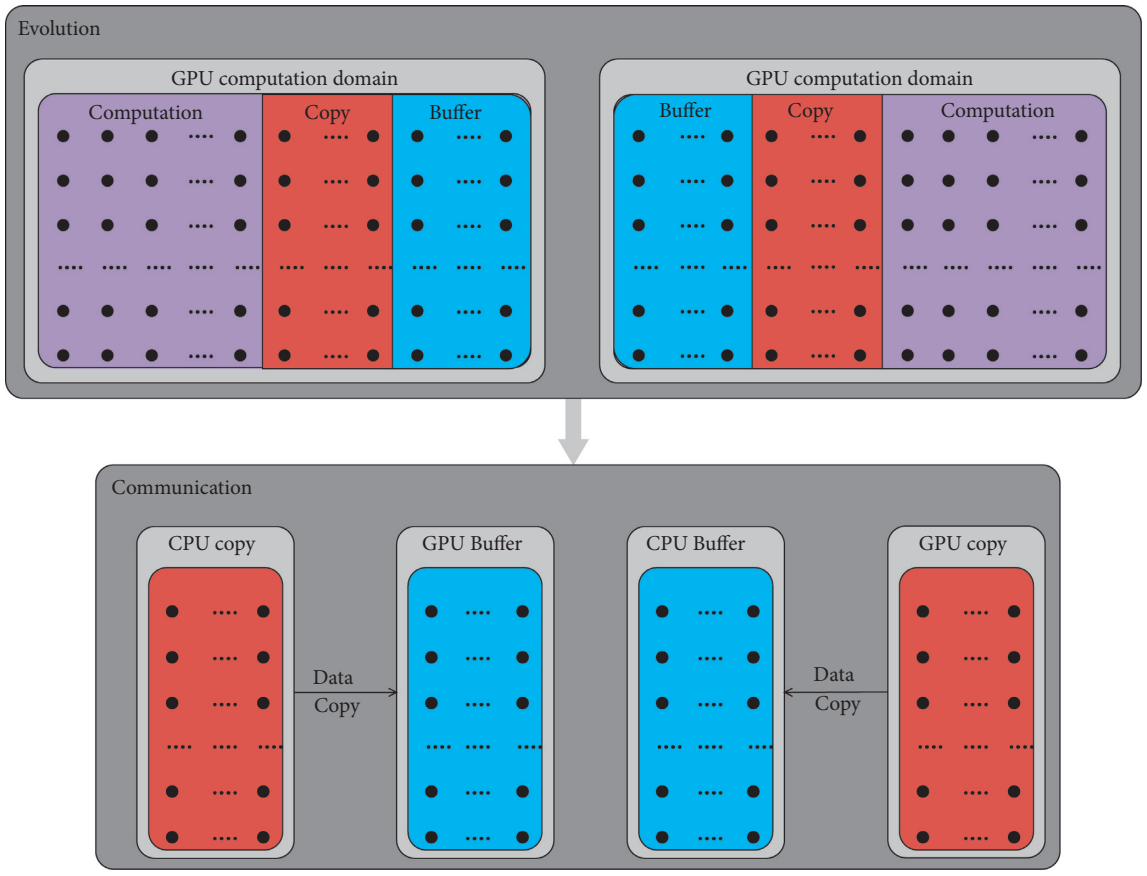

Figure 7: Data exchange between buffers in the GPU and CPU computation domains.

domain is divided. According to the proportion of GPU and CPU time consumption, the flow field is divided into equal proportions.

Through the above design, a new algorithm on CPU-GPU heterogeneous platform is generated to simulate IB-LBM (see Figure 8). The GPU and CPU simulate at the same time and exchange data through the buffer. The CUDA programming model is used on the GPU side and OpenMP is used on the CPU side to perform parallel processing on IB-LBM.

4. Performance Analysis

This section tests the performance of IB-LBM on a CPU-GPU heterogeneous platform by conducting experiments flow around a cylinder. The main characteristics of the experimental platform are shown in Table 1.

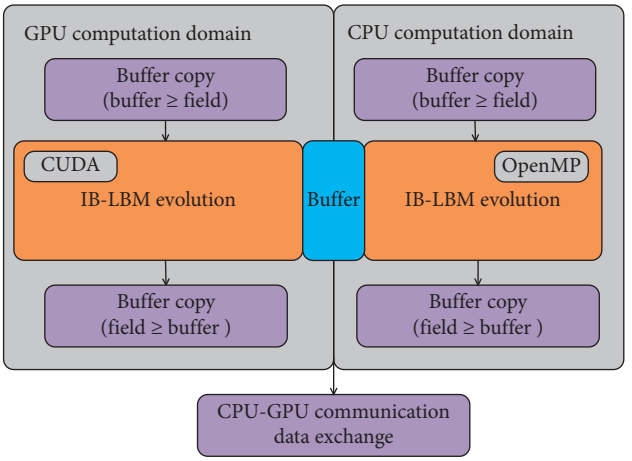

Figure 8: The new IB-LBM parallel model on heterogeneous platforms.

```
Heterogeneous:
    //The data in the array is assigned to the buffer in the flow field.
    buffer_to_field<<<Grid_4, Block_4>>>()
    //The GPU performs IB-LBM simulation.
  for i = 0 ⟶ buffer do
    compute_force<<<Grid_1, Block_1>>>()
    spread_force<<<Grid_1, Block_1>>>()
    discrete_force<<<Grid_2, Block_2>>>()
      collision<<<Grid_2, Block_2>>>()
    streaming<<<Grid_2, Block_2>>>()
    boundary<<<Grid_3, Block_3>>>()
    compute_macroscopic<<<Grid_2, Block_2>>>()
    interpolation_velocity<<<Grid_1, Block_1>>>()
    lagrangian_move<<<Grid_1, Block_1>>>()
    end for
    //Copy the data of the buffer area in the flow field to a separate array.
    field_to_buffer<<<Grid_4, Block_4>>>()
    //The CPU performs IB-LBM simulation.
for j = 0 ⟶ buffer do
    compute_force ()
      spread_force ()
    discrete_force ()
    collision ()
    streaming ()
    boundary ()
    compute_macroscopic ()
    interpolation_velocity ()
    lagrangian_move ()
cudaMemcpy (cudaMemcpyDeviceToHost)
exchange_data ()
cudaMemcpy (cudaMemcpyHostToDevice)
step_count+ = buffer
if (step_count! = total_step)
    goto heterogeneous
```

Algorithm 1: The IB-LBM parallel algorithm on CPU-GPU heterogeneous platforms

This paper used the conventional MFLUPS metric (millions of fluid lattice updates per second) to assess the performance. The following formula shows how the peak number of potential MFLUPS is calculated for a specific simulation [25]:

$$\text{MFLUPS} = \frac{s \cdot N_{fl}}{T_s \cdot 10^6}, \tag{13}$$

where s is the total number of evolutions, N_{fl} is the number of grids in the flow field, and T_s is the execution time.

The rules for dividing the flow field into GPU computing domain and CPU computing domain can be expressed by

$$\begin{aligned} N_{\text{cpu}} &= \frac{\text{MFLUPS}_{\text{gpu}} \times N_{fl}}{\text{MFLUPS}_{\text{cpu}} + \text{MFLUPS}_{\text{gpu}}} + E, \\ N_{\text{gpu}} &= \frac{\text{MFLUPS}_{\text{cpu}} \times N_{fl}}{\text{MFLUPS}_{\text{cpu}} + \text{MFLUPS}_{\text{gpu}}} + E, \end{aligned} \tag{14}$$

where N_{cpu} and N_{cpu} are the number of GPU computing domains and CPU computing domains, $\text{MFLUPS}_{\text{cpu}}$ and $\text{MFLUPS}_{\text{cpu}}$ are the MFLUPS value calculated by the multicore CPU and a GPU, and E is the number of lattice errors; the value range is $-10^4 \le E \le 10^4$.

Table 1: Main features of the platforms.

Platform	Intel Xeon	NVIDIA GPU
Model	Gold 6130	GeForce RTX 2080Ti
Frequency	2.1 GHz	1.318 GHz
Cores	16	4352
Memory	256 GB (DDR4)	11 GB (GDDR6)
Bandwidth	21.3 GB/s	616 GB/s
Compiler	gcc 5.4.0	nvcc 9.0.176

The choice of buffer size is a key issue, and the appropriate buffer size can greatly improve the calculation efficiency. After a large number of experiments and analyzing the experimental results, the choice of buffer size can be based on the following expression:

$$N_{\text{buffer}} = \frac{N_{\text{cpu}}}{25} + E, \tag{15}$$

where N_{buffer} is the number of buffer grids, N_{cpu} is the number of CPU computing domains, and the value of E is the same as the value in equation (14).

The size of the buffer has a relatively small impact on the performance of the computation simulation on the GPU side and has a relatively large impact on the performance of the computation simulation on the CPU side. Therefore, the size of the buffer can be calculated based on the size of the CPU computing domain. Choose buffers of different sizes for multiple experiments, and compare the optimal results with the CPU computing domain. The above experiments were conducted under different flow field scales. Finally, the conclusion of equation (15) is reached. It can be seen from the expression that the size of the buffer is related to the grid amount of the CPU computing domain. The amount of grid in the CPU computing domain is related to platform equipment and problem scale. Therefore, the size of the buffer depends on platform features and problem scale.

The single-precision and double-precision performance tests of the algorithm are shown in Tables 2 and 3.

As can be seen from Table 2 and Table 3, using a separate GPU to simulate IB-LBM can also achieve good performance results. The parallel performance of the CPU is far from the effect of the GPU; it is still a computing resource that can be utilized. In some cases, the hardware implements double precision, then single precision is emulated by extending it there, and the conversion will cost time. This results in double-precision performance being faster than single-precision performance. Using a single GPU for computing has greatly improved computing performance. However, due to the computing characteristics of the CUDA model, the CPU only executes control of the kernel functions, so the computing resources of the CPU are not fully utilized. In the proposed algorithm, CPU controls the start of the kernel function and performs multi-threaded calculations without causing a waste of resources. Besides that, compared with the traditional heterogeneous method, the communication between the CPU and the GPU is reduced. It can be concluded from Tables 2 and 3 that the performance of the new algorithm has been improved. However, due to the small size of the flow field, the performance improvement is not obvious.

It can be seen from the experimental result that the performance with double precision is much lower than that with single precision on GPU. This can be caused by two reasons: On the one hand, the computing power of GPU single-precision floating-point numbers is higher than that of double-precision floating-point numbers. On the other hand, double-precision floating-point numbers increase memory access.

Figure 9 shows the performance image of the enlarged flow field scale in the single precision case. The performance improvement can be more clearly observed.

It can be seen from Figure 9 that, after the scale of the flow field reaches 6 million, the MFLPUPS of the CPU-GPU heterogeneous platform with buffer is similar to the sum of GPU and CPU multithreading. When the number of grids is 1 million, the simulation effect of a single GPU is the best. This is due to the relatively small number of grids in the CPU computing domain divided by a heterogeneous CPU-GPU platform. The time spent on thread development and destruction has a greater impact on the total CPU simulation time. When performing parallel simulation on CPU-GPU heterogeneous platforms, not only will communication affect performance, but synchronization will also affect performance. The introduction of the buffer reduces the number of communications and synchronization at the same time. It can also be seen from Figure 9 that the effect of using the buffer is better than not using the buffer.

To study the actual FSI, the scale of the problem is often relatively large. At the same time, many evolutionary calculations are often required. Parallel algorithms can effectively reduce computation time. Most clusters now include GPU and CPU devices. The algorithm proposed in this paper can make full use of the resources of the cluster, and it can reduce the simulation time in the research of RBC aggregation and deformation in ultrasonic field [6], viscous flow in large distensible blood vessels [26], suspending viscosity effect on RBC dynamics and blood flow behaviors in microvessels [27], and other studies.

5. Application

RBCs are an important component in blood [28]. Recently, the IB-LBM has been used for simulating the deformation and motion of elastic bodies immersed in fluid flow including RBCs [29]. Esmaily et al. [30] used IB-LBM to investigate motion and deformation of both healthy and sick RBCs in a microchannel with stenosis. Tan et al. [31] presented a numerical study on nanoparticle (NP) transport and dispersion in RBCs suspensions by an IB-LBM fluid solver.

In this paper, the RBCs are suspended in blood plasma which has a density $\rho = 1$ and Reynolds number $\mathrm{Re} < 1$. In the boundary processing, the nonequilibrium extrapolation is used. The cross section profile of a RBC in x–y plane [32] is given by the following relation:

$$\begin{aligned} y &= \sin\theta, \\ x &= \cos\theta\left[c_0 + c_1 \sin\theta - c_2 \sin^2\theta\right], \\ 0 &< \theta < 2\pi. \end{aligned} \tag{16}$$

with $c_0 = 0.207$, $c_1 = 2.002$, and $c_2 = 1.122$.

The two-dimensional RBC shape is shown in Figure 10.

The red points represent the Lagrangian points on the RBC, and the black points are Euler points.

The force at the Lagrangian points on RBC consists of two parts defined as

$$F(s,t) = F_s(s,t) + F_b(s,t), \tag{17}$$

where s is the arc length, F_s is the stretching/compression force, and F_b is the bending force. F_s and F_b are obtained by the following relation [33]:

Table 2: Single-precision performance of CPU-GPU heterogeneous platform.

Domain size	CPUMFLUPS	OpenMPMFLUPS	GPUMFLUPS	CPU-GPUMFLUPS
625×89 (55625)	0.801	19.449	367.375	368.564
750×107 (80250)	0.867	22.435	435.120	436.649
900×128 (115200)	0.888	23.023	485.470	486.983
1080×154 (166320)	0.869	24.017	501.921	503.572

Table 3: Double-precision performance of CPU-GPU heterogeneous platform.

Domain size	CPUMFLUPS	OpenMPMFLUPS	GPUMFLUPS	CPU-GPUMFLUPS
$62\,5 \times 89$ (55625)	0.874	19.841	193.750	193.835
750×107 (80250)	0.953	23.902	240.037	240.849
900×128 (115200)	0.958	24.057	294.730	295.036
1080×154 (166320)	0.956	24.793	321.114	322.597

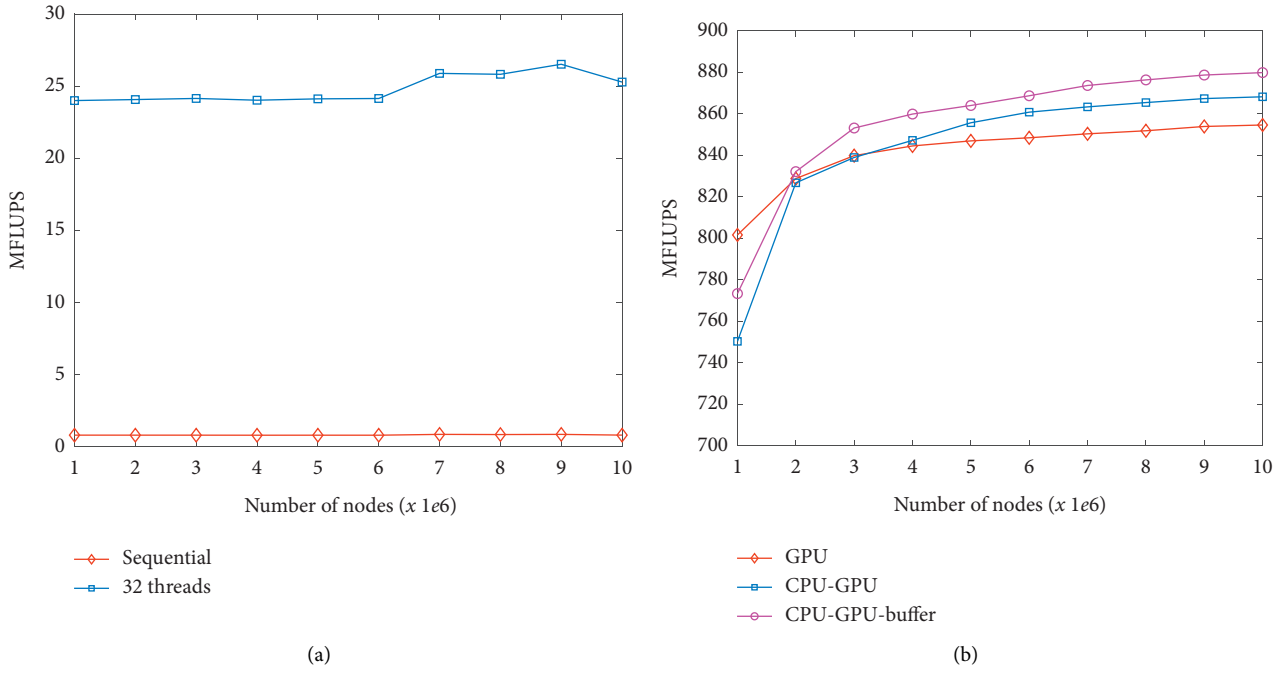

Figure 9: Performance of simulating flow around a cylinder on different platforms. (a) CPU sequential and multi-threads. (b) GPU and CPU-GPU heterogeneous platform.

$$(F_s)_k = \frac{E_s}{(\Delta s)^2} \times \sum_{j=1}^{N-1} \left\{ \left(\left| X_{j+1} - X_j \right| - \Delta s \right) \times \frac{X_{j+1} - X_j}{\left| X_{j+1} - X_j \right|} \left(\delta_{j,k} - \delta_{j+1,k} \right) \right\},$$
$$(F_b)_k = \frac{E_b}{(\Delta s)^4} \times \sum_{j=2}^{N-1} \left\{ \left(X_{j+1} - 2X_j + X_{j-1} \right) \left(2\delta_{j,k} - \delta_{j+1,k} - \delta_{j-1,k} \right) \right\}. \tag{18}$$

Here, N is the total number of the Lagrangian points on the RBC, $k = 1, 2, \ldots, N$, $(F_s)_k$ and $(F_b)_k$ are elastic Lagrangian forces associated with the node k, the values of F_s and F_b are 1.0×10^{-10} and 1.0×10^{-12}, respectively, $X(s,t)$ is the Lagrangian points, and $\delta_{j,k}$ is the Kronecker delta function.

Blood flow in the microvessels is better approximated by a Poiseuille flow than a shear flow, which makes the study of RBCs in Poiseuille flows more physiologically realistic [34]. Therefore, this paper first applies the algorithm to study the dynamical RBC behavior in Poiseuille flows; the simulation domain is 30×100 grid points (see Figure 11).

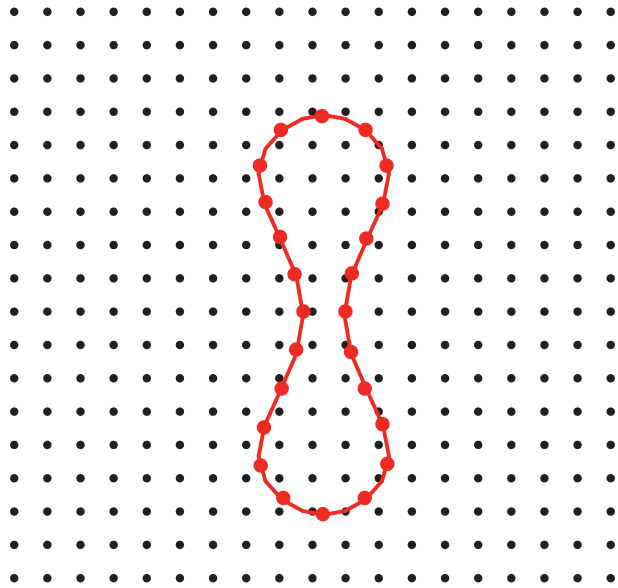

Figure 10: The two-dimensional structure of RBC in the flow field.

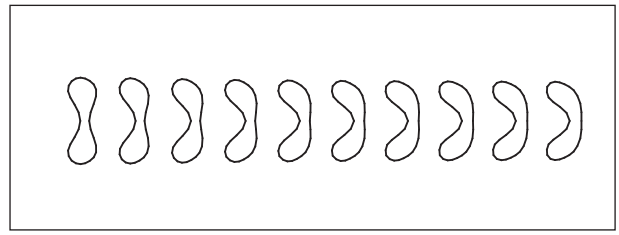

Figure 11: Snapshots of RBC behavior in Poiseuille flows.

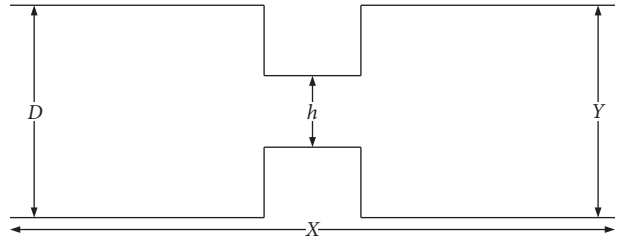

Figure 12: Geometry of the microchannel with stenosis.

RBCs are perpendicular to the direction of fluid flow in Figure 11; the geometric shape gradually changes from a double concave to a parachute shape. The curvature after RBC deformation is closely related to the deformability of the boundary [35]. RBC can recover its initial shape associated with the minimal elastic energy when the flow stops [36–38].

Many diseases narrow the blood vessels. The degree of stenosis is high, which will affect the circulatory function of the human body. The study of RBCs passing through the stenotic area of the blood vessel is more popular. This paper applies the algorithm to this situation. Figure 12 shows geometry of the microchannel.

In the geometry of the microchannel, X and Y are number of Eulerian points in x and y direction, the values are 30 and 150, D is the diameter of vessel, and h is the diameter of constriction. The ratio of D and h is $D = 3h$. Figure 13 shows the motion and deformation of an RBC through a microchannel with stenosis.

It can be seen from Figure 13 that the velocity of the fluid increases in the stenosis area, resulting in more deformation of the RBCs [39]. After passing through this area, the RBCs return to their initial shape due to the decrease in the velocity of the fluid. This is because RBC is very flexible and yet resilient enough to recover the biconcave shape whenever the cell is quiescent [40].

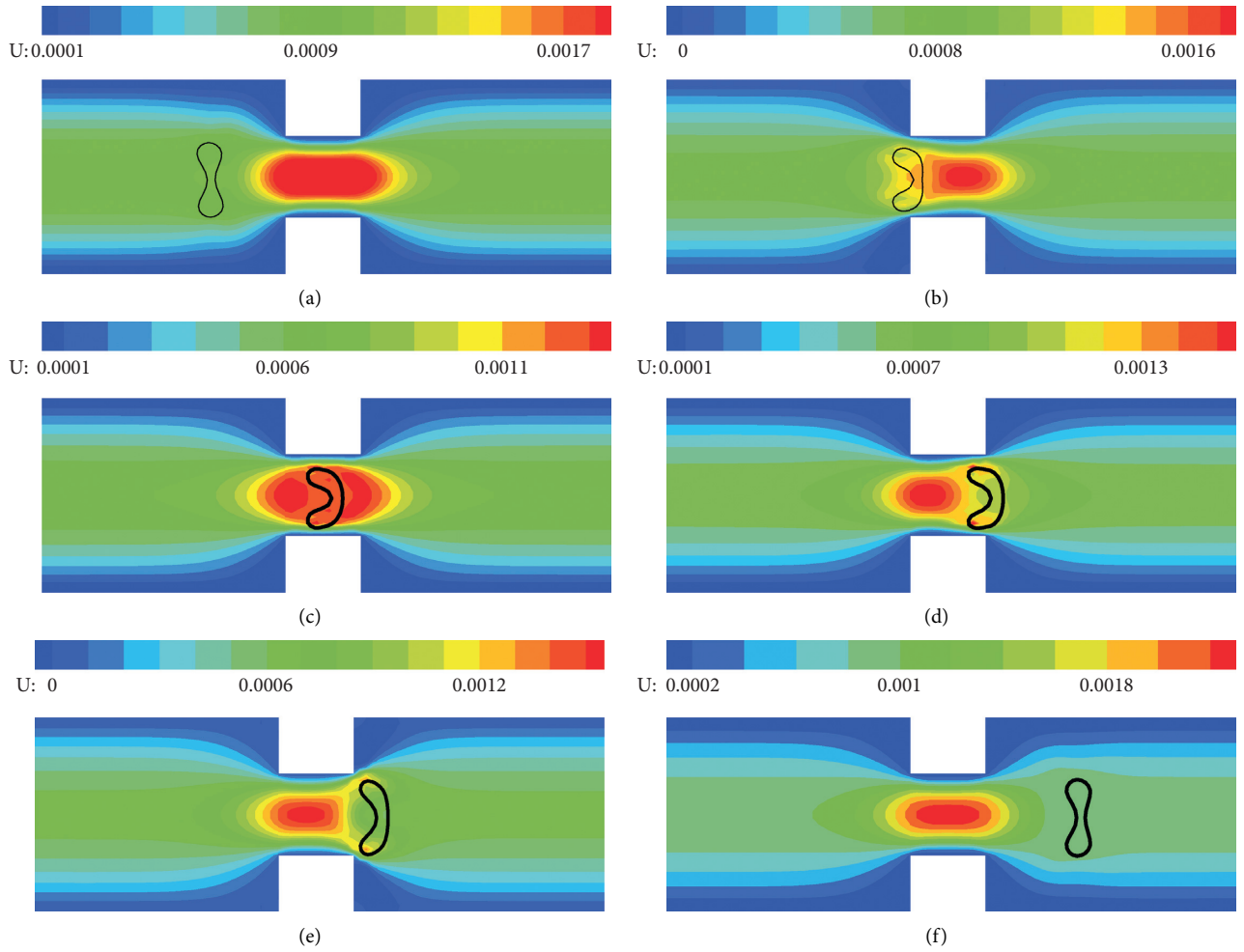

Figure 13: An RBC through a microchannel with stenosis at different times.

6. Conclusions

In this paper, a new parallel algorithm on CPU-GPU heterogeneous platforms is proposed to implement a coupled lattice Boltzmann and immersed boundary method. Good performance results are obtained by investigating heterogeneous platforms. The main contribution of this paper is to make full use of the CPU and GPU computing resources. CPU not only controls the launch of the kernel function but also performs calculations. A buffer is set in the flow field to reduce the amount of data exchanged between the CPU and the GPU. The CUDA programming model and OpenMP are applied to the GPU and CPU, respectively, for numerical simulation. This method can approximate the simultaneous simulation of different regions of the flow field by the GPU and CPU. The final experiments have obtained relatively good performance results. The CPU-GPU heterogeneous platform MFLUPS value is approximately the sum of GPU and CPU multi-threads. In addition, the proposed algorithm is used to simulate the movement and deformation of RBCs in Poiseuille flow and through a microchannel with stenosis. The simulation results are in good agreement with other literatures. Our future work is to implement the method on large-scale GPU clusters and large-scale CPU clusters to simulate three-dimensional FSI problems.

Data Availability

All data, models, and codes generated or used during the study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported by the Shanghai Sailing Program (Grant no.18YF1410100), National Key Research and Development Program (Grant no. 2016YFC1400304), National Natural Science Foundation of China (NSFC) (Grant nos. 41671431 and 61972241), the Natural Science Foundation of Shanghai (no. 18ZR1417300), and the Program for the Capacity Development of Shanghai Local Colleges (Grant no. 17050501900).

References

[1] J. Wu, Y. Cheng, W. Zhou, and W. Zhang, "GPU acceleration of FSI simulations by the immersed boundary-lattice Boltzmann coupling scheme," *Computers and Mathematics with Applications*, vol. 78, no. 9, pp. 1194–1205, 2019.

[2] Z.-G. Feng and E. E. Michaelides, "The immersed boundary-lattice Boltzmann method for solving fluid-particles interaction problems," *Journal of Computational Physics*, vol. 195, no. 2, pp. 602–628, 2004.

[3] A. Eshghinejadfard, A. Abdelsamie, S. A. Hosseini, and D. Thévenin, "Immersed boundary lattice Boltzmann simulation of turbulent channel flows in the presence of spherical particles," *International Journal of Multiphase Flow*, vol. 96, pp. 161–172, 2017.

[4] M. Cheng, B. Zhang, and J. Lou, "A hybrid LBM for flow with particles and drops," *Computers & Fluids*, vol. 155, pp. 62–67, 2017.

[5] Y. Wei, L. Mu, Y. Tang, Z. Shen, and Y. He, "Computational analysis of nitric oxide biotransport in a microvessel influenced by red blood cells," *Microvascular Research*, vol. 125, Article ID 103878, 2019.

[6] X. Ma, B. Huang, G. Wang, X. Fu, and S. Qiu, "Numerical simulation of the red blood cell aggregation and deformation behaviors in ultrasonic field," *Ultrasonics Sonochemistry*, vol. 38, pp. 604–613, 2017.

[7] L. Mountrakis, E. Lorenz, O. Malaspinas, S. Alowayyed, B. Chopard, and A. G. Hoekstra, "Parallel performance of an IB-LBM suspension simulation framework," *Journal of Computational Science*, vol. 9, pp. 45–50, 2015.

[8] C. Feichtinger, J. Habich, H. Köstler, U. Rüde, and T. Aoki, "Performance modeling and analysis of heterogeneous lattice Boltzmann simulations on CPU-GPU clusters," *Parallel Computing*, vol. 46, pp. 1–13, 2015.

[9] J. Beny and J. Latt, "Efficient LBM on GPUs for dense moving objects using immersed boundary condition," in *Proceedings of the CILAMCE*, Paris, Compiègne. France, November 2018.

[10] M. Bernaschi, M. Fatica, S. Melchionna, S. Succi, and E. Kaxiras, "A flexible high-performance Lattice Boltzmann GPU code for the simulations of fluid flows in complex geometries," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 1, pp. 1–14, 2010.

[11] P. R. Rinaldi, E. A. Dari, M. J. Vénere, and A. Clausse, "A Lattice-Boltzmann solver for 3D fluid simulation on GPU," *Simulation Modelling Practice and Theory*, vol. 25, pp. 163–171, 2012.

[12] Z. Shang, M. Cheng, and J. Lou, "Parallelization of lattice Boltzmann method using MPI domain decomposition technology for a drop impact on a wetted solid wall," *International Journal of Modeling, Simulation and Scientific Computing*, vol. 5, no. 2, Article ID 1350024, 2014.

[13] P. Valero-Lara, "Reducing memory requirements for large size LBM simulations on GPUs," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 24, Article ID e4221, 2017.

[14] P. Valero-Lara and J. Jansson, "Multi-domain grid refinement for lattice-Boltzmann simulations on heterogeneous platforms," in *Proceedings of 18th IEEE International Conference on Computational Science and Engineering (CSE)*, Porto, Portugal, October 2015.

[15] P. Valero-Lara and J. Jansson, "Heterogeneous CPU+ GPU approaches for mesh refinement over Lattice-Boltzmann simulations," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 7, 2017.

[16] P. Valero-Lara, P. Pinelli, and M. Prieto-Matias, "Accelerating solid-fluid interaction using lattice-Boltzmann and immersed boundary coupled simulations on heterogeneous platforms," in *Proceedings of 14th International Conference on Computational Science (ICCS)*, Cairns, Australia, June 2014.

[17] G. Boroni, J. Dottori, and P. Rinaldi, "FULL GPU implementation of lattice-Boltzmann methods with immersed boundary conditions for fast fluid simulations," *The International Journal of Multiphysics*, vol. 11, no. 1, 2017.

[18] F. Massaioli and G. Amati, "Achieving high performance in a LBM code using OpenMP," in *Proceedings of the Fourth European Workshop on OpenMP (EWOMP 2002)*, Roma, Italy, September, 2002.

[19] P. Valero-Lara and J. Jansson, "LBM-HPC-an open-source tool for fluid simulations. case study: Unified parallel C (UPC-PGAS)," in *Proceedings of IEEE International Conference on Cluster Computing (CLUSTER)*, Chicago, USA, September 2015.

[20] P. Valero-Lara and J. Jansson, "A non-uniform Staggered Cartesian grid approach for Lattice-Boltzmann method," in *Proceedings of International Conference on Computational Science (ICCS)*, Reykjavík, Iceland, June 2015.

[21] Y. H. Qian, D. D'Humières, and P. Lallemand, "Lattice BGK models for Navier-Stokes equation," *Europhysics Letters (EPL)*, vol. 17, no. 6, pp. 479–484, 1992.

[22] 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.

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

[24] Z. Guo, C. Zheng, and B. Shi, "Discrete lattice effects on the forcing term in the lattice Boltzmann method," *Physical Review E*, vol. 65, no. 4, Article ID 046308, 2002.

[25] A. P. Randles, V. Kale, J. Hammond, W. Gropp, and E. Kaxiras, "Performance analysis of the lattice Boltzmann model beyond Navier–Stokes," in *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Boston, MA, USA, May. 2013.

[26] H. Fang, Z. Wang, Z. Lin, and M. Liu, "Lattice Boltzmann method for simulating the viscous flow in large distensible blood vessels," *Physical Review E*, vol. 65, no. 5, 2002.

[27] J. Zhang, "Effect of suspending viscosity on red blood cell dynamics and blood flows in microvessels," *Microcirculation*, vol. 18, no. 7, pp. 562–573, 2011.

[28] M. Navidbakhsh and M. Rezazadeh, "An immersed boundary-lattice Boltzmann model for simulation of malaria-infected red blood cell in micro-channel," *Scientia Iranica*, vol. 19, no. 5, pp. 1329–1336, Oct. 2012.

[29] A. Hassanzadeh, N. Pourmahmoud, and A. Dadvand, "Numerical simulation of motion and deformation of healthy and sick red blood cell through a constricted vessel using hybrid lattice Boltzmann-immersed boundary method," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 20, no. 7, pp. 737–749, 2017.

[30] R. Esmaily, N. Pourmahmoud, and I. Mirzaee, "Numerical simulation of interaction between red blood cell with surrounding fluid in Poiseuille flow," *Advances in Applied Mathematics and Mechanics*, vol. 10, no. 1, pp. 62–76, 2018.

[31] J. Tan, W. Keller, S. Sohrabi, J. Yang, and Y. Liu, "Characterization of nanoparticle dispersion in red blood cell suspension by the lattice Boltzmann-immersed boundary method," *Nanomaterials*, vol. 6, no. 2, 2016.

[32] J. Zhang, P. C. Johnson, and A. S. Popel, "Red blood cell aggregation and dissociation in shear flows simulated by lattice Boltzmann method," *Journal of Biomechanics*, vol. 41, no. 1, pp. 47–55, 2008.

[33] A. Ghafouri, R. Esmaily, and A. a. Alizadeh, "Numerical simulation of tank-treading and tumbling motion of red blood cell in the Poiseuille flow in a microchannel with and without obstacle," *Iranian Journal of Science and Technology, Transactions of Mechanical Engineering*, vol. 43, no. 4, pp. 627–638, 2019.

[34] T. Wang, T. W. Pan, Z. Xing, and R. Glowinski, "Numerical simulation of rheology of red blood cell rouleaux in microchannels," *Physical Review E*, vol. 79, no. 4, Article ID 041916, 2009.

[35] Y. Xu, F. Tian, and Y. Deng, "An efficient red blood cell model in the frame of IB-LBM and its application," *International Journal of Biomathematics*, vol. 6, no. 1, 2013.

[36] T. M. Fischer, "Shape memory of human red blood cells," *Biophysical Journal*, vol. 86, no. 5, pp. 3304–3313, 2004.

[37] T. W. Pan and T. Wang, "Dynamical simulation of red blood cell rheology in microvessels," *International Journal of Numerical Analysis & Modeling*, vol. 6, no. 3, pp. 455–473, 2009.

[38] C. Pozrikidis, "Axisymmetric motion of a file of red blood cells through capillaries," *Physics of Fluids*, vol. 17, no. 3, Article ID 031503, 2005.

[39] R. Esmaily, N. Pourmahmoud, and I. Mirzaee, "An immersed boundary method for computational simulation of red blood cell in Poiseuille flow," *Mechanika*, vol. 24, no. 3, pp. 329–334, 2018.

[40] J. Li, G. Lykotrafitis, M. Dao, and S. Suresh, "Cytoskeletal dynamics of human erythrocyte," *Proceedings of the National Academy of Sciences*, vol. 104, no. 12, pp. 4937–4942, 2007.