Commun. Comput. Phys.
doi: 10.4208/cicp.2014.m342

Vol. **17**, No. 4, pp. 960-974
April 2015

Multi-GPU Based Lattice Boltzmann Method for Hemodynamic Simulation in Patient-Specific Cerebral Aneurysm

Changsheng Huang[1], Baochang Shi[1,*], Zhaoli Guo[2] and Zhenhua Chai[1]

[1] *School of Mathematics and Statistics, Huazhong University of Science and Technology, Wuhan 430074, China.*
[2] *State Key Laboratory of Coal Combustion, Huazhong University of Science and Technology, Wuhan 430074, China.*

Received 30 October 2013; Accepted (in revised version) 12 June 2014

Abstract. Conducting lattice Boltzmann method on GPU has been proved to be an effective manner to gain a significant performance benefit, thus the GPU or multi-GPU based lattice Boltzmann method is considered as a promising and competent candidate in the study of large-scale complex fluid flows. In this work, a multi-GPU based lattice Boltzmann algorithm coupled with the sparse lattice representation and message passing interface is presented. Some numerical tests are also carried out, and the results show that a parallel efficiency close to 90% can be achieved on a single-node cluster equipped with four GPU cards. Then the proposed algorithm is adopted to study the hemodynamics of patient-specific cerebral aneurysm with stent implanted. It is found that the stent can apparently reduce the aneurysmal inflow and improve the hemodynamic environment. This work also shows that the lattice Boltzmann method running on the GPU platform is a powerful tool to study the fluid mechanism within the aneurysms and enable us to better understand the pathogenesis and treatment of cerebral aneurysms.

AMS subject classifications: 65Y05, 76Z05, 76M28

Key words: Lattice Boltzmann method, GPU computing, MPI, cerebral aneurysm, blood flow.

1 Introduction

Cerebral aneurysms are localized dilation or ballooning of the brain blood vessel, usually caused by disease or weakening of the walls. They are particularly dangerous because their rupture often leads to subarachnoid hemorrhage, which is associated with

*Corresponding author. *Email addresses:* hcs@hust.edu.cn (C. Huang), zlguo@hust.edu.cn (Z. Guo), shibc@hust.edu.cn (B. Shi), hustczh@hust.edu.cn (Z. Chai)

 ©2015 Global-Science Press

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

high morbidity and mortality. Although the mechanisms responsible for the formation and rupture of cerebral aneurysms are not fully understood, hemodynamic factors are thought to play an important role in the pathogenesis and treatment of cerebral aneurysms [1].

A novel therapy to treat cerebral aneurysm is to implant a porous stent across the neck of the aneurysm, which has been viewed as a promising and minimally invasive treatment modality. The placement of the stent could change aneurysmal hemodynamic environment, reduce the inflow into the aneurysm, and prevent the further growth of the aneurysm. For this reason, many different methods, including experimental study, clinical observation, and numerical simulation, have been developed to investigate the effect of the stent on hemodynamic characteristics. Among them, the numerical simulation has been proved powerful and efficient [2–6]. As a mesoscopic numerical approach, the lattice Boltzmann method (LBM) [7–9], has been widely used to investigate the hemodynamics of cerebral aneurysm for its distinct advantages [10–16], including easy implementation, intrinsic parallelism, and capability in treating complex boundary conditions. However, most of the previous work, which assessed the effect of stent deployment on hemodynamics within the aneurysm, are only limited to the two-dimensional or idealized aneurysm models. In this work, we will consider a patient-specific cerebral aneurysm model which is reconstructed from medical image.

At the same time, GPU (graphics processing unit) computing, which is a novel and popular parallel computing technology, has also attracted increasing attention in the area of lattice Boltzmann modeling and simulation. Several researchers implemented the LBM on GPU platform and reported rather remarkable performance [17–22]. Besides, their results also show that the LBM is very compatible with GPU computing. However, the multi-GPU based lattice Boltzmann algorithm has not been studied extensively, especially for fluid flows in complex geometries. In this paper, an algorithm designed for multi-GPU platform is proposed, where the message passing interface (MPI) technique is adopted for GPU management. Considering the complex structure of the blood vessel, the sparse lattice representation [23] is also utilized to reduce memory consumption and improve performance of the algorithm. Then the proposed algorithm is used to study the effect of the stent on the hemodynamics of cerebral aneurysm.

The rest of this paper is organized as follows. In Section 2, we will present a brief overview on the lattice Boltzmann method. In Section 3, a multi-GPU based lattice Boltzmann algorithm and its performance analysis are presented. In the next section, the proposed algorithm is used to study the effect of the stent on hemodynamic characteristics of blood flow in a patient-specific cerebral aneurysm, and finally, some conclusions are summarized in Section 5.

2 The lattice Boltzmann method

The lattice Boltzmann method can be viewed as a discrete scheme of the continuous Boltzmann equation [24]. For each time step, particles collide at each lattice site and

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

then propagate to neighboring sites along discrete directions. A popular class of lattice Boltzmann models is the so-called lattice Bhatnagar-Gross-Krook (BGK) model with a single relaxation time approximation. In this model, the evolution equation reads

$$f_i(\mathbf{x}+\mathbf{c}_i\delta t,t+\delta t)-f_i(\mathbf{x},t)=-\frac{1}{\tau}\left[f_i(\mathbf{x},t)-f_i^{(eq)}(\mathbf{x},t)\right], \tag{2.1}$$

where τ is the dimensionless relaxation time, $f_i(\mathbf{x},t)$ is the density distribution function for the particle moving with velocity $\mathbf{c}_i$ at position $\mathbf{x}$ and time t, $f_i^{(eq)}(\mathbf{x},t)$ is the local equilibrium distribution function. In this paper, a three-dimensional lattice Boltzmann model with nineteen velocities (D3Q19 model) [25] is used, and the corresponding equilibrium distribution function is defined as

$$f_i^{(eq)}(\mathbf{x},t)=\omega_i\rho\left[1+\frac{\mathbf{c}_i\cdot\mathbf{u}}{c_s^2}+\frac{(\mathbf{c}_i\cdot\mathbf{u})^2}{2c_s^4}-\frac{|\mathbf{u}^2|}{2c_s^2}\right], \tag{2.2}$$

where ω_i is the weighting factor, and given by $\omega_0=1/3$, $\omega_{1-6}=1/18$, $\omega_{7-18}=1/36$. ρ, $\mathbf{u}$ are the fluid density and velocity, c_s is the sound speed. The discrete velocities $\mathbf{c}_i$ in Eq. (2.2) are defined by

$$\mathbf{c}_i=c\mathbf{e}_i=c\begin{cases}(0,0,0), & i=0,\\ (\pm1,0,0),(0,\pm1,0),(0,0,\pm1), & i=1-6,\\ (\pm1,\pm1,0),(\pm1,0,\pm1,),(0,\pm1,\pm1,), & i=7-18,\end{cases} \tag{2.3}$$

where $c=\delta x/\delta t$, and δx and δt are the lattice spacing and time step, respectively. The relation between c_s and c can be expressed by $c_s=c/\sqrt{3}$. The macroscopic density and velocity are computed by

$$\begin{aligned}\rho(\mathbf{x},t)&=\sum_{i=0}^{18}f_i(\mathbf{x},t),\\ \rho(\mathbf{x},t)\mathbf{u}(\mathbf{x},t)&=\sum_{i=0}^{18}\mathbf{c_i}f_i(\mathbf{x},t).\end{aligned} \tag{2.4}$$

For fluid flows in complex geometries, the most widely used boundary condition in the LBM is the halfway bounce-back scheme [26], which means particles that hit the solid boundary, then simply return back in the opposite direction where they came from. Generally, this process will introduce some $if\cdots else\cdots$ clauses, which will affect performance of the algorithm. In our simulations, the halfway bounce-back scheme is applied for the solid wall, while the non-equilibrium extrapolation method is adopted to treat the boundary conditions at inlet and outlet [27, 28].

3 Lattice Boltzmann method on GPU

In our GPU algorithm, two sets of the distribution functions located in global memory are used. At even time steps, the distribution function values are read from one of them

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

and stored by the other, while at odd time steps, the values are shifted in the opposite direction.

To reduce the data access time, the collision and propagation steps are combined in one kernel, and each thread is assigned to one node. It should be noted that the propagation step would lead to uncoalesced memory accesses in the processes of shifting distribution function values to neighboring nodes and executing the bounce-back rule. Thus, the collision conducted before propagation or vice versa may influence the performance of GPU algorithm since the former would bring uncoalesced write and the latter has uncoalesced read. In the present work, the scheme of "propagation conducted before collision" is adopted since the previous work has demonstrated that uncoalesced writes are more expensive than uncoalesced read [29].

In addition, due to the complex geometry of the vessel, the volume fraction of fluid nodes is very small, and is usually about 20%. For this reason, the sparse lattice representation will be utilized to reduce memory consumption.

3.1 The sparse lattice representation

To improve the efficiency of LBM in studying fluid flows in complex geometries, an algorithm coupled with a sparse lattice representation for complex structure of flow field was first proposed by Schulz et al. [23], and proved to be efficient both in achieving a higher computational performance and reducing memory consumption [30, 31]. Recently, this technique was also adopted by Bernaschi et al. in their MUPHY project [32, 33]. In the sparse lattice representation, only fluid nodes are stored (Fig. 1), thus it is quite suitable for a computational domain with a high volume fraction of solid phase. In the following, the sparse lattice representation is also incorporated into our algorithm to study blood flow in cerebral aneurysm.

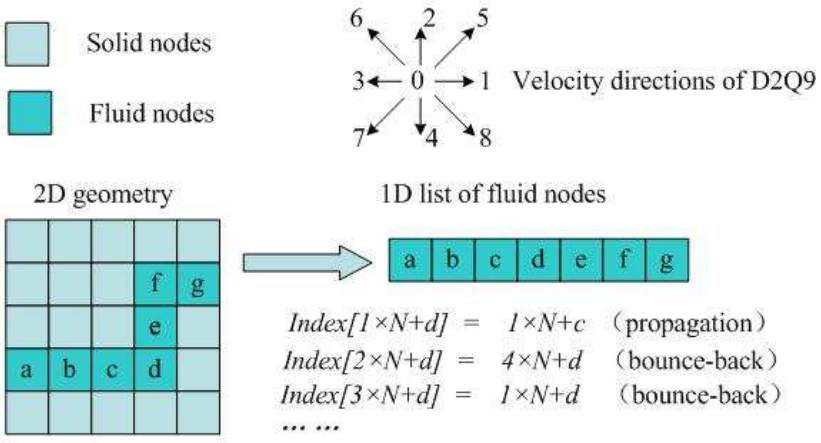


Figure 1: Two-dimensional diagram of the sparse lattice representation and indirect addressing.

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

Since not all the information of the computational domain is available, we need to use indirect addressing scheme to find neighboring nodes. To this end, an array named $Index$ with a size of $N\times 19$ (N is the number of fluid nodes) is used to label the addresses of neighboring nodes. $Index[N\times q+k]$ stores the index of the distribution function value which streams into the node k in the direction of q ($q>0$, Fig. 1). If $q=0$, $Index[k]$ will be set as a flag to denote whether the node k is an inlet or outlet node. To satisfy the requirement of coalesced access [34], N is set to be a multiple of 16 by adding some ghost nodes at the end of the fluid node list.

Note that the bounce-back rule can also be implicitly implemented by this approach, for example, if the neighboring node of k in q direction is a solid node, then $Index[N\times\bar{q}+k]$ stores the index of the distribution function value of node k in q direction (See Fig. 1), where $\bar{q}$ means the opposite direction of q. In this way, the $if\cdots else\cdots$ clauses introduced by the bounce-back rule can be avoided.

To execute the non-equilibrium extrapolation method at inlet and outlet, which need to access neighboring fluid node, we use the $Index$ array to store the memory distance between inlet/outlet node and its neighboring node. In fact, for inlet/outlet node k, $Index[N\times 0+k]$ is defined as

$$Index[k]=dist\times 10+flag\times Sgn(dist), \tag{3.1}$$

where $dist$ is the distance between the index of node k and that of its neighboring node, $flag$ denotes whether the node k is an inlet or outlet node and is set to be 2 or 3 in our code, $Sgn()$ is sign function.

The execution process can be described as follows:

```
__global__ void evolution(··· ···)
{
    calculate the index k of present node;
    i=index[k];

    //load the distribution values that stream into node k;
    f0=f_dev[0×N+k+i/10];
    f1=f_dev[index[1×N+k+i/10]];
    ··· ···

    compute the macroscopic density and velocity;
    boundary treatment at inlet and outlet;
    execute collision step and save updated data.
}
```

3.2 Multi-GPU based lattice Boltzmann method

To gain a higher performance, the algorithm based on multi-GPU platform, where message passing interface (MPI) technique is adopted for GPU management, is developed.

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

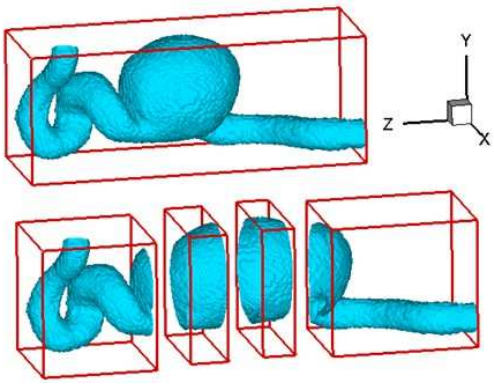


Figure 2: Domain decomposition, each sub-domain has nearly the same number of fluid nodes.

3.2.1 Domain decomposition

First of all, the computational domain is divided into sub-domains before adopting the sparse lattice representation, and one-dimensional partition is employed. In addition, the GPU based LBM generally adopts the structure-of-array (SOA) data layout to meet the requirement of coalescing memory access, i.e., $f_i((x,y,z),t)$ is stored with the index $(i\times nz\times ny\times nx+z\times ny\times nx+y\times nx+x)$. Under this layout, the computational domain is divided along z direction (major direction) with the requirement that each sub-domain has nearly the same number of fluid nodes. In the implementation, two extra layers are added in each sub-domain, which are the same as the first or last layer of neighboring sub-domains and used to exchange data.

3.2.2 Overlapping of the computation and data transfer

After domain decomposition, the sub-domain is further divided into two parts, outer layers (first or last layer) and inner layers. Then a one-dimensional list of fluid nodes is gained by utilizing the sparse lattice representation. The outer and inner layers are assigned to two CUDA streams. A stream is a sequence of commands which are executed in order, and commands within different streams can be executed concurrently [34]. The stream that deals with the outer layers is in charge of data transfer. In this way, the data exchange with the neighboring processor can be overlapped with the computation of inner layers, which can be illustrated by Fig. 3 and the following pseudo-code:

for $t=0$ to $tmax$ **do**
  evolution_inner(); //stream[1]
  data copy from GPU to CPU; //stream[0]
  cudaStreamSynchronize(stream[0]);
  MPI communication;
  data copy from CPU to GPU; //stream[0]
  evolution_outter(); //stream[0]
end for

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

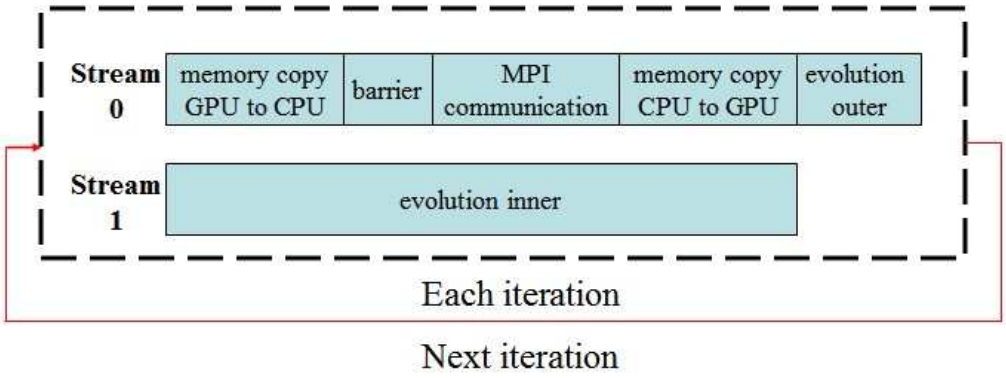


Figure 3: Flow diagram of the overlapped model.

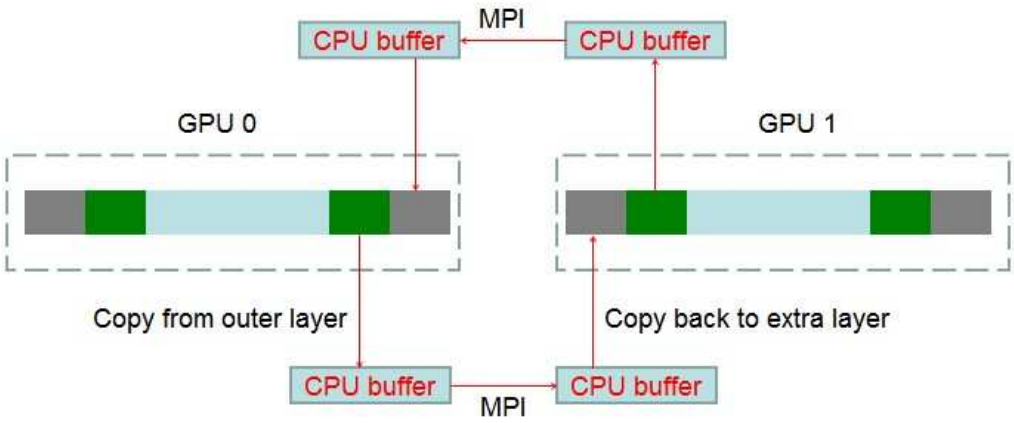


Figure 4: The process of data exchange between GPU nodes.

The data exchange between GPU nodes can be described as follows (also see Fig. 4): (1) at the beginning of each iteration, the data to be exchanged with neighboring processors are located in outer layers, then they are copied into CPU memory buffers; (2) with the help of MPI, the data are transferred into CPU memory buffer of the neighboring GPU node; (3) the data are copied into the extra layer of neighboring GPU node.

3.2.3 Optimization of data layout

As the computational domain is divided along z direction, for the fluid nodes of the outer layer, the distribution function values which will propagate to neighboring sub-domains should be copied into CPU buffer for communication. In D3Q19 model, ten velocities of the total nineteen velocities (Eq. (2.3)) have non-zero velocity in z direction (i.e., $e_{iz} \neq 0$), thus the distribution function values in ten directions ($e_{iz} = \pm 1$) should be copied into CPU buffer for communication. However, it should be noted that ten times of data copy

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

are needed since these directions are not continuous in commonly used D3Q19 model. To reduce the memory copy time, here we rearrange the directions as Eq. (3.2) so that the directions with the same z field are gathered together. In this way, the data to be exchanged are continuous in memory, and data transfer can be completed through twice data copy

$$\mathbf{e}_i = \begin{Bmatrix} 0 & 1 & -1 & 0 & 0 & 1 & -1 & 1 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 \end{Bmatrix}. \quad (3.2)$$

3.3 Performance analysis

In this section, we will test the performance of the developed algorithm. To this end, some simulations are carried out on two computers, one is equipped with four Tesla C1060 GPU cards, and the other one is equipped with two Tesla K20 GPU cards. The performance of the algorithm is measured in terms of million fluid lattice updates per second (MFLUPS), unless otherwise stated, the single precision is used.

The first numerical test considered here is the fluid flow past array of randomly arranged spheres with a diameter of 40 lattice units (Fig. 5), and the porosity is ranged from 70% to 90%. We conducted several simulations, and presented the results in Table 1. As seen from this table, our algorithm can achieve a parallel efficiency close to 90% with four Tesla C1060 GPU cards, and gains a higher MFLUPS than the previous work [32]. Besides, it is also found that the MFLUPS achieved by Tesla K20 (with ECC)

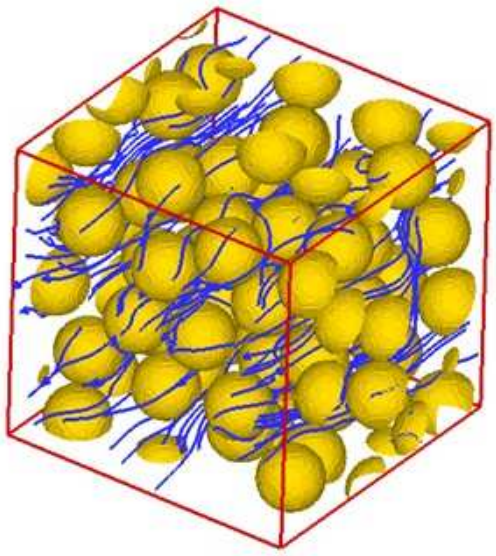


Figure 5: Fluid flow past array of random arranged spheres.

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

Table 1: Performance (MFLUPS, single precision) of the GPU algorithm for fluid flow past array of random arranged spheres with different porosities.

GPU number	lattice size	70%	80%	90%
1 Tesla C1060	192^3	288	270	296
2 Tesla C1060	192^3	394	551	527
4 Tesla C1060	192^3	1019	907	1060
4 Tesla C1060	384^3	1016	884	1087
1 Tesla K20	192^3	554	533	559
2 Tesla K20	192^3	1036	1023	1074

Table 2: Performance (MFLUPS, double precision) of the GPU algorithm for fluid flow past array of random arranged spheres with different porosities.

GPU number	lattice size	70%	80%	90%
1 Tesla C1060	192^3	120	120	122
2 Tesla C1060	192^3	235	161	238
4 Tesla C1060	192^3	467	469	470
1 Tesla K20	192^3	334	330	337
2 Tesla K20	192^3	626	634	621

is nearly twice of that obtained by Tesla C1060. This is caused by the difference of their bandwidth considering the LBM is a memory bandwidth limited algorithm. To confirm this statement, the *bandwidthTest* program supplied by the NVIDIA CUDA SDK is used to test the memory bandwidth of Tesla C1060 and K20, and the results show that the latter (140 GB/s) is almost the twice of the former (72 GB/s). For our algorithm, the highest achieved bandwidth by the two GPUs are 63 and 119 GB/s, which are close to the memory bandwidth of the GPUs.

The performance of our algorithm with double precision is also investigated, and the results are presented in Table 2. As shown in this table, the performance with double precision is much lower than that with single precision, which is caused by two reasons: firstly, the single precision floating point capability of GPU is much higher than double precision floating point capability, and secondly, the double precision leads to a larger amount of memory access. In addition, we also compared the results with single precision and double precision, and found that the global relative error of velocity is less than 4%. For this reason and considering the efficiency of the algorithm, the single precision will be used in following simulations.

The second numerical test is the blood flow in cerebral aneurysm (see Fig. 6) where a lattice size of $449\times512\times1324$ (about 6.6×10^7 fluid nodes) is used. Due to the high memory consumption, we can only perform simulations on four Tesla C1060 GPUs, and a MFLUPS of 1090 is obtained.

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

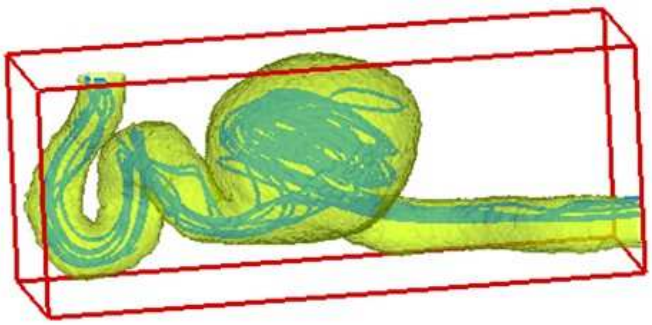


Figure 6: Blood flow in a patient-specific cerebral aneurysm.

Next, we also presented an analysis on the memory consumption. In the common full matrix mode, three arrays are needed, two of them are used to store distribution function values with a size of $(nx\times ny\times nz\times 19)$ and another one is adopted to label fluid phase with a $(nx\times ny\times nz)$ size. While in the mode of sparse lattice representation, three arrays with a size of $(N\times 19)$ $(N=nx\times ny\times nz\times p)$ are allocated provided the porosity is p. Actually, in our code N may be a little bigger than $(nx\times ny\times nz\times p)$ since some ghost nodes may be needed to make N be a multiple of 16. However, we would like to point out that, since the number of ghost nodes is much smaller than that of fluid nodes, the influence of these ghost nodes on the memory consumption can be negligible. Compared with common full matrix mode, the sparse matrix mode will consume less global memory if $p<0.68$ (Eq. (3.3)), which is usually satisfied for the realistic cerebral aneurysms,

$$p<\frac{2\times nx\times ny\times nz\times 19+nx\times ny\times nz}{3\times nx\times ny\times nz\times 19}\approx 0.68. \tag{3.3}$$

4 Hemodynamic simulation

In this section, the developed GPU algorithm is used to study the effect of the stent on hemodynamics of patient-specific cerebral aneurysm (Fig. 6), which is reconstructed from multi-slice spiral CT scans ($dx\approx 0.35mm$). To accurately describe the stent (with a wire diameter of 0.1 mm) effect, the lattice is enlarged seven times ($dx\approx 0.05mm$). On the other hand, the aneurysm model is also rotated to achieve a smaller bounding box size (from $757\times 848\times 967$ to $449\times 512\times 1324$, see Fig. 7).

To virtually deploy the stent, we mainly follow the way proposed by Appanaboyina et al. [4]: the centerline of the vessel is first fixed, then the stent is deployed along the centerline, as shown in Fig. 8. Here a helical stent with a wire diameter of 2 lattice units, and a porosity (metal free area / total unit area) of 89.53% is used. The diameter of the vessel is about $6mm$ and 120 in lattice unit.

The density and viscosity of the blood are $1.087\times 10^3 kg/m^3$, $3.695\times 10^{-3} pa\cdot s$, respec-

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

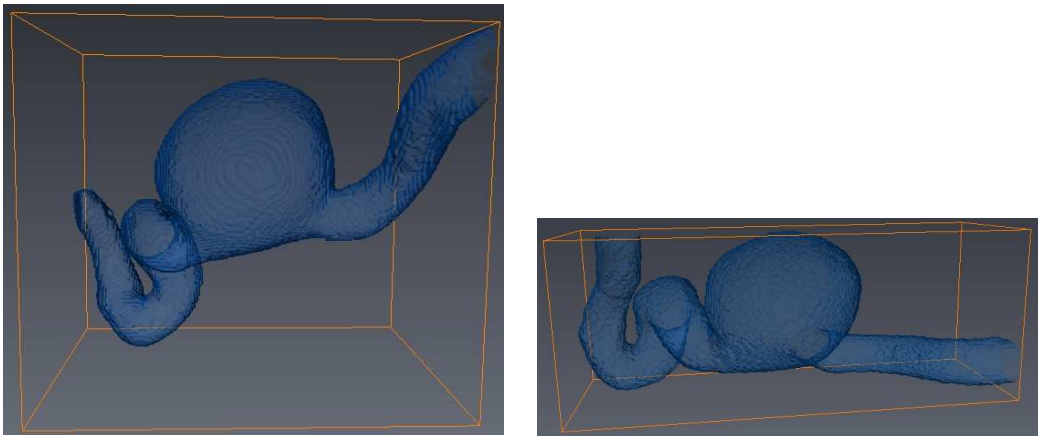


Figure 7: The aneurysm model before and after the rotate process.

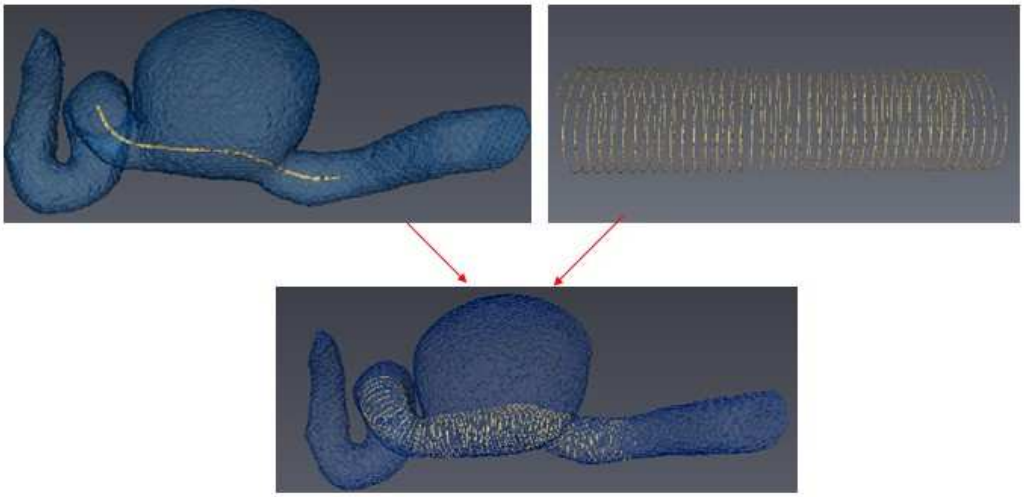


Figure 8: The process of virtual stent deployment.

tively. To simplify the numerical study, the initialized dimensionless density $\rho = 1.0$ and viscosity $v = 0.036$ are used. At the inlet, a steady velocity of 0.1 is imposed, while at the outlet, the density is fixed as $\rho = 1.0$.

The flow patterns within the cerebral aneurysm before and after stent deployment are shown in Fig. 9. As seen from this figure, a much simpler flow pattern in the cerebral aneurysm is observed after the stent deployment. Actually, some previous researchers have found that ruptured aneurysms usually have complex flow patterns, while unruptured aneurysms are often related to simple flow patterns [35]. Thus the results shown in Fig. 9 indicate that the stent can be used to reduce the rupture risk of cerebral aneurysm.

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

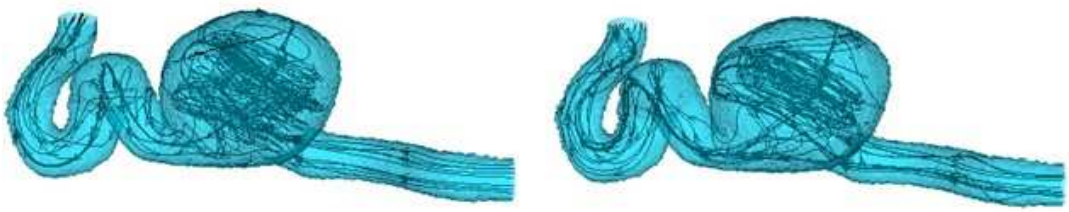


Figure 9: Flow pattern within the cerebral aneurysm, left: before stenting; right: after stenting.

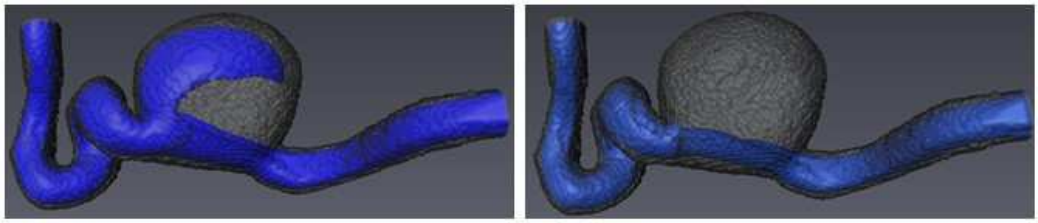


Figure 10: Iso-velocity surfaces of $v=0.06$, left: before stenting; right: after stenting.

To study the effect of the stent serviced as flow diverter, we presented the iso-velocity surface of $v=0.06$ before and after stent deployment in Fig. 10, where a significant difference between them is observed. Before the stent deployment, the blood flow streams into the aneurysm cavity from distal part of the aneurysm neck, while after the stent deployment, the intra-aneurysmal flow is greatly reduced. This phenomenon is of benefit to creating stagnant flow, and finally inducing thrombus formation and aneurysm occlusion. In addition, we also calculated the mean velocity magnitude inside the aneurysm, and found that the mean velocity magnitude is 0.0455 before the stent deployment, while 0.0099 after the stent deployment. This means that the velocity of the flow in the aneurysm can be significantly reduced after the stent deployment.

5 Conclusion

In this paper, a multi-GPU based LBM algorithm is designed for fluid flows in complex geometries, where the sparse lattice representation is used. We first presented a study on the performance of the algorithm, and found that a parallel efficiency close to 90% can be achieved on a single-node cluster equipped with four Tesla C1060 GPU cards. Then the algorithm is applied to study blood flow characteristics in a patient-specific cerebral aneurysm, and the results show that the stent can be used to improve aneurysmal hemodynamic environment. Our future work is to implement our algorithm on large-scale GPU clusters.

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

Acknowledgments

C.H. would like to thank the ICMMES committee and NVIDIA Company for NVIDIA award of Tesla K20. This work is financially supported by the National Natural Science Foundation of China (Grant Nos. 11272132, 51306133, 51006040 and 11302073), the National Science Fund for Distinguished Young Scholars of China (51125024), and the China Postdoctoral Science Foundation (Grant No. 2012M521424).

References

[1] G. N. Foutrakis, H. Yonas, R. J. Sclabassi, Saccular aneurysm formation in curved and bifurcating arteries, Am. J. Neuroradiol., 20 (7)(1999), 1309-1317.
[2] G. R. Stuhne, D. A. Steinman, Finite-element modeling of the hemodynamics of stented aneurysms, J. Biomech. Eng., 126 (2004), 382-387.
[3] A. Radaelli, L. Augsburger, J. Cebral, M. Ohta, D. Rüfenacht, R. Balossino, G. Benndorf, D. Hose, A. Marzo, R. Metcalfe, et al., Reproducibility of haemodynamical simulations in a subject-specific stented aneurysm model – A report on the virtual intracranial stenting challenge 2007, J. Biomech., 41 (10) (2008), 2069-2081.
[4] S. Appanaboyina, F. Mut, R. Löhner, C. Putman, J. Cebral, Simulation of intracranial aneurysm stenting: Techniques and challenges, Comput. Method. Appl. Mech. Engr., 198 (45-46) (2009), 3567-3582.
[5] J. R. Cebral, F. Mut, D. Sforza, R. Löhner, E. Scrivano, P. Lylyk, C. Putman, Clinical application of image-based cfd for cerebral aneurysms, Int. J. Numer. Meth. Bio., 27 (7) (2011), 977-992.
[6] J. Cebral, F. Mut, M. Raschi, E. Scrivano, R. Ceratto, P. Lylyk, C. Putman, Aneurysm rupture following treatment with flow-diverting stents: Computational hemodynamics analysis of treatment, Am. J. Neuroradiol., 32 (1), (2011) 27-33.
[7] R. Benzi, S. Succi, M. Vergassola, The lattice Boltzmann equation: Theory and applications, Phys. Rep., 222 (3) (1992) 145-197.
[8] S. Chen, G.D. Doolen, Lattice Boltzmann method for fluid flows, Annu. Rev. Fluid Mech., 30 (1998) 329-364.
[9] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Oxford University Press, Oxford, 2001.
[10] M. Hirabayashi, M. Ohta, D. A. Rüfenacht, B. Chopard, Characterization of flow reduction properties in an aneurysm due to a stent, Phys. Rev. E, 68 (2) (2003), 021918.
[11] B. Chopard, R. Ouared, D. A. Rüfenacht, A lattice boltzmann simulation of clotting in stented aneursysms and comparison with velocity or shear rate reductions, Math. Comput. Simulat., 72 (2-6) (2006), 108-112.
[12] J. Bernsdorf, D. Wang, Non-newtonian blood flow simulation in cerebral aneurysms, Comput. Math. Appl., 58 (5) (2009), 1024-1029.
[13] X. He, G. Duckwiler, D.J. Valentino, Lattice Boltzmann simulation of cerebral artery hemodynamics, Comput. Fluids, 38 (4) (2009), 789-796.
[14] Y. H. Kim, X. Xu, J. S. Lee, The effect of stent porosity and strut shape on saccular aneurysm and its numerical analysis with lattice boltzmann method, Ann. Biomed. Eng., 38 (7) (2010), 2274-2292.

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

[15] C. Huang, Z. Chai, B. Shi, Non-Newtonian effect on hemodynamic characteristics of blood flow in stented cerebral aneurysm, Commun. Comput. Phys., 13 (3) (2013), 916-928.
[16] F. Weichert, L. Walczak, D. Fisseler, T. Opfermann, M. Razzaq, R. Münster, S. Turek, I. Grunwald, C. Roth, C. Veith, M. Wagner, Simulation of intra-aneurysmal blood flow by using different numerical methods, Comput. Math. Methods Med., 2013 (2013), doi: 10.1155/2013/527654.
[17] J. Tölke, M. Krafczyk, TeraFLOP computing on a desktop PC with GPUs for 3D CFD, Int. J. Comput. Fluid D., 22 (7) (2008), 443-456.
[18] F. Kuznik, C. Obrecht, G. Rusaouen, J.-J. Roux, LBM based flow simulation using GPU computing processor, Comput. Math. Appl., 59 (7) (2010), 2380-2392.
[19] C. Obrecht, F. Kuznik, B. Tourancheau, J.-J. Roux, The TheLMA project: Multi-GPU implementation of the lattice Boltzmann method, Int. J. High Perform. C., 25 (3) (2011), 295-303.
[20] C. Obrecht, F. Kuznik, B. Tourancheau, J.-J. Roux, Multi-GPU implementation of the lattice Boltzmann method, Comput. Math. Appl., 65 (2) (2013), 252-261.
[21] J. Toelke, G. De Prisco, Y. Mu, Lattice Boltzmann multi-phase simulations in porous media using multiple GPUs, AGU Fall Meeting Abstracts (2011).
[22] Q. G. Xiong, B. Li, J. Xu, X. J. Fang, X. W. Wang, L. M. Wang, X. F. He, W. Ge, Efficient parallel implementation of the lattice Boltzmann method on large clusters of graphic processing units, Chinese Sci. Bull., 57 (7) (2012), 707-715.
[23] M. Schulz, M. Krafczyk, J. Tölke, E. Rank, Parallelization strategies and efficiency of cfd computations in complex geometries using lattice boltzmann methods on high-performance computers, in: High Performance Scientific and Engineering Computing, Springer, 2002, 115-122.
[24] X. He, L.-S. Luo, Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation, Phys. Rev. E, 56 (6) (1997): 6811.
[25] Y. Qian, D. d'Humières, P. Lallemand, Lattice BGK model for Navier-Stokes equation, Europhys. Lett., 17 (6) (1992), 479-484.
[26] X. He, Q. Zou, L.-S. Luo, M. Dembo, Analytic solutions of simple flows and analysis of nonslip boundary conditions for the lattice Boltzmann BGK model, J. Stat. Phys., 87 (1) (1997), 115-136.
[27] Z. L. Guo, C. G. Zheng, B. C. Shi, An extrapolation method for boundary conditions in lattice Boltzmann method, Phys. Fluids 14 (6) (2002) 2007-2010.
[28] Z. H. Chai, B. C. Shi, L. Zheng, Simulating high Reynolds number flow in two-dimensional lid-driven cavity by multi-relaxation-time lattice Boltzmann method, Chin. Phys. 15 (8) (2006) 1855-1863.
[29] C. Obrecht, F. Kuznik, B. Tourancheau, J. J. Roux, Global memory access modelling for efficient implementation of the lattice boltzmann method on graphics processing units, in: High Performance Computing for Computational Science - CVECPAR2010, in: Lecture Notes in Computer Science, Springer, 2010.
[30] C. Pan, J. F. Prins, C. T. Miller, A high-performance lattice boltzmann implementation to model flow in porous media, Comput. Phys. Commun., 158 (2) (2004), 89-105.
[31] L. Axner, J. Bernsdorf, T. Zeiser, P. Lammers, J. Linxweiler, A. G. Hoekstra, Performance evaluation of a parallel sparse lattice boltzmann solver, J. Comput. Phys., 227 (10) (2008), 4895-4911.
[32] M. Bernaschi, M. Fatica, S. Melchionna, S. Succi, E. Kaxiras, A flexible high-performance lattice Boltzmann GPU code for the simulations of fluid flows in complex geometries, Concurr. Comput.: Pract. Exp., 22 (1) (2010), 1-14.

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press

[33] M. Bernaschi, M. Bisson, M. Fatica, S. Melchionna, S. Succi, Petaflop hydrokinetic simulations of complex flows on massive GPU clusters, Comput. Phys. Commun., 184 (2) (2013), 329-341.
[34] NVIDIA, NVIDIA CUDA Compute Unified Device Architecture: Programming Guide (Version 3.2).
[35] J. R. Cebral, M. A. Castro, J. E. Burgess, R. S. Pergolizzi, M. J. Sheridan, C. M. Putman, Characterization of cerebral aneurysms for assessing risk of rupture by using patient-specific computational hemodynamics models, Am. J. Neuroradiol., 26 (10) (2005), 2550-2559.

https://doi.org/10.4208/cicp.2014.m342 Published online by Cambridge University Press