LATTICE BOLTZMANN SIMULATIONS OF CAVITY FLOWS ON GRAPHIC PROCESSING UNIT WITH MEMORY MANAGEMENT

P. Y. Hong L. M. Huang C. Y. Chang C. A. Lin*

Department of Power Mechanical Engineering
National Tsing Hua University
Hsinchu, Taiwan

ABSTRACT

Lattice Boltzmann method (LBM) is adopted to compute two and three-dimensional lid driven cavity flows to examine the influence of memory management on the computational performance using Graphics Processing Unit (GPU). Both single and multi-relaxation time LBM are adopted. The computations are conducted on nVIDIA GeForce Titan, Tesla C2050 and GeForce GTX 560Ti. The performance using global memory deteriorates greatly when multi relaxation time (MRT) LBM is used, which is due to the scheme requesting more information from the global memory than its single relaxation time (SRT) LBM counterpart. On the other hand, adopting on chip memory the difference using MRT and SRT is not significant. Also, performance of LBM streaming procedure using offset reading surpasses offset writing ranging from 50% to 100% and this applies to both SRT and MRT LBM. Finally, comparisons using different GPU platforms indicate that Titan as expected outperforms other devices, and attains 227 and 193 speedup over its Intel Core i7-990 CPU counterpart and four times faster than GTX 560Ti and Tesla C2050 for three dimensional cavity flow simulations respectively with single and double precisions.

Keywords: Single and multi-relaxation time, Lattice Boltzmann method, Two and three dimensional lid-driven cavity flow, Graphic Processing Unit, Memory management.

1. INTRODUCTION

The lid driven cavity flow has been widely investigated numerically. Although the geometry is quite simple, the internal flow is quite complex and this serves as an ideal platform to explore performance of different numerical techniques. An early review of cavity flow was presented by Shankar and Deshpande [1].

Two dimensional square cavity flows were examined by Ghia *et al*. [2] and Schreiber and Keller [3], and these accurate solutions were regarded as benchmark solutions. On the other hand, three-dimensional cavity flows were also investigated. Iwatsu *et al*. [4], Guj and Stella [5] and Mei *et al*. [6] simulated cubic cavity flows with different numerical approaches, and results showed that steady solution exists at Reynolds number Re $\leq$ 2000. However, at lower Reynolds number Re $\sim$ 1914, Feldman and Gelfgat [7] found that the oscillatory instability occurs. This was further supported by a PIV measurement [8], where the critical Reynolds number is in region $1700 < Re < 1970$.

Previous studies on lid driven cavity has been conducted mostly using Navier-Stokes equation, and there were increasing studies adopting the lattice Boltzmann method [6, 9, 10]. The lattice Boltzmann method [11, 12] is very suitable for parallelization, due to its explicit approach and with intensive local computation. This can be realized using the Graphical Processing Unit (GPU) through the Compute Unified Device Architecture (CUDA) [13]. Lattice Boltzmann methods [14-19] have been successful implemented on GPUs. Computational power of GPUs has far exceeded that of PC-based CPUs. For example the computing platform nVIDIA GeForce Titan used in the present work has been reported to deliver up to 4494 GFLOPS (Giga Floating Point Operations Per Second) for single and 1300 GFLOPS for double precision [13].

The parallel performance of GPUs can be further elevated by the reduction of data transfer between host and device. Some efforts adopt different streaming strategy to enhance the parallel performance [14, 15]. Another way to increase the performance of a GPU is to utilize the shared memory [16]. A detailed description of CUDA memory can be found in [13].

The present study is addressed to explore the graphic processing unit memory management to increase the parallel performance. The computations are conducted on nVIDIA GeForce Titan, Tesla C2050 and GeForce

* Corresponding author (calin@pme.nthu.edu.tw)

DOI : 10.1017/jmech.2017.70
Copyright © 2017 The Society of Theoretical and Applied Mechanics
https://doi.org/10.1017/jmech.2017.70 Published online by Cambridge University Press

GTX 560Ti. The major GPU computing platform adopted is the nVIDIA GeForce Titan, which is the latest Kepler architecture with computing capability of 3.5 TFlops under single precision. This study is an extension of previous studies [18, 19] and both Single and multi- relaxation time LBM are used here to investigate their influences on the GPU performance. The results are obtained for two and three dimensional lid driven cavity flows using both single and double precision floating point operations. Issues addressed are performances of GPU with different memory managements and the number of threads per block. Optimum range of block dimension is discussed first. This is followed by the influence of the memory management. Finally, the performance of three different GPU relative to CPU is presented.

2. MATHEMATICAL FORMULATION AND BOUNDARY CONDITIONS

This section is devoted to the brief summary of LBM formulation. Detailed description of the methodology may be found in Refs. [20, 21].

2.1 Single Relaxation Time Lattice Boltzmann Model

Lattice Boltzmann equations adopting a uniform lattice with BGK collision model can be expressed as,

$$f_i\left(\vec{x}+\vec{e}_i\Delta t, t+\Delta t\right)=f_i\left(\vec{x},t\right)-\frac{1}{\tau_f}\left[f_i\left(\vec{x},t\right)-f_i^{eq}\left(\vec{x},t\right)\right] \tag{1}$$

where f_i is the particle distribution function in the $\vec{e}$ direction and τ_f is the relaxation time.

Here, the density and momentum can be obtained as:

$$\rho=\sum f_i \tag{2}$$

$$\rho\vec{u}=\sum f_i\vec{e}_i \tag{3}$$

D2Q9 and D3Q19 lattice models are employed for two and three dimensional simulations. The corresponding equilibrium density distribution functions are defined as,

$$f_i^{eq}=\omega_i\rho\left[1+\frac{3\vec{e}_i\cdot\vec{u}}{C^2}+\frac{9\left(\vec{e}_i\cdot\vec{u}\right)^2}{2C^4}-\frac{3\vec{u}\cdot\vec{u}}{2C^2}\right] \tag{4}$$

where the weighting factors ω_i for D2Q9 and D3Q19 models, are respectively, as $\omega_0 = 4/9$, $\omega_{i=1\sim4} = 1/9$, $\omega_{i=5\sim8} = 1/36$, and $\omega_0 = 1/3$, $\omega_{i=1\sim6} = 1/18$, $\omega_{i=7\sim18} = 1/36$.

For the D2Q9 model, the particle velocity $\vec{e}$ is defined as,

$$\vec{e}_i=\begin{cases}(0,0)C, & i=0\\ \left(\cos\left[(i-1)\pi/2\right],\sin\left[(i-1)\pi/2\right]\right)C, & i=1\sim4\\ \left(\cos\left[(2i-9)\pi/4\right],\sin\left[(2i-9)\pi/4\right]\right)\sqrt{2}C, & i=5\sim8\end{cases} \tag{5}$$

Here, $C = \Delta x/\Delta t$ is the lattice speed, where Δx and Δt are the lattice width and time step, respectively.

For the D3Q19 model, the particle velocity $\vec{e}_i$ is defined as,

$$\vec{e}_i=\begin{cases}(0,0,0)C, & i=0\\ (\pm1,0,0)C,(0,\pm1,0)C,(0,0,\pm1)C, & i=1\sim6\\ (\pm1,\pm1,0)C,(0,\pm1,\pm1)C,(\pm1,0,\pm1)C, & i=7\sim18\end{cases} \tag{6}$$

2.2 Multi Relaxation Time Lattice Boltzmann Model

Though the single-relaxation-time (SRT) lattice Boltzmann model is commonly used, it does not deliver stable results for flows at higher Reynolds number. On the other hand, multi-relaxation-time (MRT) model [22] does enhance the numerical stability and reduce the non-physical oscillations.

The multi-relaxation-time lattice Boltzmann method proposed by Lallemand and Luo [22] can be expressed as,

$$f_i^*\left(\vec{x},t\right)=f_i\left(\vec{x},t\right)-M_{il}^{-1}S_{lj}\left[m_j\left(\vec{x},t\right)-m_j^{eq}\left(\vec{x},t\right)\right], \tag{7}$$

$$f_i^*\left(\vec{x}+\vec{e}_i\Delta t,t+\Delta t\right)=f_i^*\left(\vec{x},t\right) \tag{8}$$

Here, m is a new column vector of moments and M is the transform matrix such that

$$m\equiv Mf \tag{9}$$

The matrix M transforms a vector in the vector space V spanned by the discrete velocities into a vector in the vector space spanned by the moments of f.

https://doi.org/10.1017/jmech.2017.70 Published online by Cambridge University Press

- For D2Q9 model

$$m=\begin{bmatrix}\rho\\ e\\ \varepsilon\\ j_x\\ q_x\\ j_y\\ q_y\\ p_{xx}\\ p_{xy}\end{bmatrix}=\begin{bmatrix}1&1&1&1&1&1&1&1&1\\ -4&-1&-1&-1&-1&2&2&2&2\\ 4&-2&-2&-2&-2&1&1&1&1\\ 0&1&0&-1&0&1&-1&-1&1\\ 0&-2&0&2&0&1&-1&-1&1\\ 0&0&1&0&-1&1&1&-1&-1\\ 0&0&-2&0&2&1&1&-1&-1\\ 0&1&-1&1&-1&0&0&0&0\\ 0&0&0&0&0&1&-1&1&-1\end{bmatrix}=\begin{bmatrix}f_0\\ f_1\\ f_2\\ f_3\\ f_4\\ f_5\\ f_6\\ f_7\\ f_8\end{bmatrix}=Mf$$

The column shows the corresponding moments adopted.
The collision procedure for m is performed as follows:

$$m^*=\begin{cases}\rho^*=\rho-s_0(\rho-\rho^{eq})\\ e^*=e-s_1(e-e^{eq})\\ \varepsilon^*=\varepsilon-s_2(\varepsilon-\varepsilon^{eq})\\ j_x^*=j_x-s_3(j_x-j_x{}^{eq})\\ q_x^*=q_x-s_4(q_x-q_x{}^{eq})\\ j_y^*=j_y-s_5(j_y-j_y{}^{eq})\\ q_y^*=q_y-s_6(q_y-q_y{}^{eq})\\ p_{xx}^*=p_{xx}-s_7(p_{xx}-p_{xx}{}^{eq})\\ p_{xy}^*=p_{xy}-s_8(p_{xy}-p_{xy}{}^{eq})\end{cases}=m-S(m-m^{eq})$$

where * denotes the post collision state and S is the diagonal matrix

$$S=diag\left[s_0,s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8\right]$$

with some constrains that $s_0 = s_3 = s_5 = 0$ for conserved moments, $s_4 = s_6 = 0$ and $s_7 = s_8 = 0$ to get isotropy. In this case the bulk velocity ζ and kinematic viscosity υ are

$$\zeta=\frac{1}{3}\left(\frac{1}{s_1}-\frac{1}{2}\right),\quad \upsilon=\frac{1}{3}\left(\frac{1}{s_8}-\frac{1}{2}\right).$$

- For D3Q19 model: The moments, transform matrix M, relaxation matrix S and corresponding equilibrium values are proposed in d'Humières *et al*. [21], which are

$$M=\begin{bmatrix}1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\ -30&-11&-11&-11&-11&-11&-11&8&8&8&8&8&8&8&8&8&8&8&8\\ 12&-4&-4&-4&-4&-4&-4&1&1&1&1&1&1&1&1&1&1&1&1\\ 0&1&-1&0&0&0&0&1&-1&1&-1&1&-1&1&-1&0&0&0&0\\ 0&-4&4&0&0&0&0&1&-1&1&-1&1&-1&1&-1&0&0&0&0\\ 0&0&0&1&-1&0&0&1&1&-1&-1&0&0&0&0&1&-1&1&-1\\ 0&0&0&-4&4&0&0&1&1&-1&-1&0&0&0&0&1&-1&1&-1\\ 0&0&0&0&0&1&-1&0&0&0&0&1&1&-1&-1&1&1&-1&-1\\ 0&0&0&0&0&-4&4&0&0&0&0&1&1&-1&-1&1&1&-1&-1\\ 0&2&2&-1&-1&-1&-1&1&1&1&1&1&1&1&1&-2&-2&-2&-2\\ 0&-4&-4&2&2&2&2&1&1&1&1&1&1&1&1&-2&-2&-2&-2\\ 0&0&0&1&1&-1&-1&1&1&1&1&-1&-1&-1&-1&0&0&0&0\\ 0&0&0&-2&-2&2&2&1&1&1&1&-1&-1&-1&-1&0&0&0&0\\ 0&0&0&0&0&0&0&1&-1&-1&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&-1&-1&1\\ 0&0&0&0&0&0&0&0&0&0&0&1&-1&-1&1&0&0&0&0\\ 0&0&0&0&0&0&0&1&-1&1&-1&-1&1&-1&1&0&0&0&0\\ 0&0&0&0&0&0&0&-1&-1&1&1&0&0&0&0&1&-1&1&-1\\ 0&0&0&0&0&0&0&0&0&0&0&1&1&-1&-1&-1&-1&1&1\end{bmatrix}$$

The relaxation matrix is defined as

$$S=diag\left(s_0,s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8,s_9,s_{10},s_{11},s_{12},s_{13},s_{14},s_{15},s_{16},s_{17},s_{18}\right)$$

https://doi.org/10.1017/jmech.2017.70 Published online by Cambridge University Press

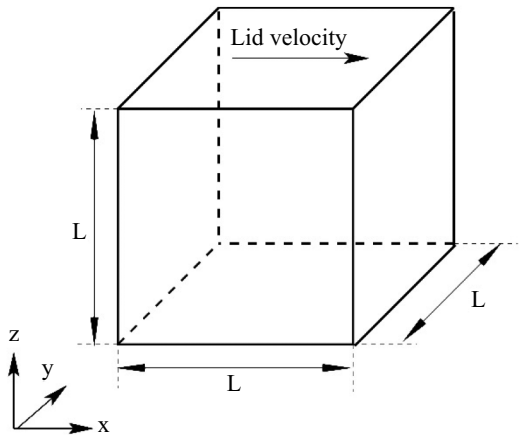


Fig. 1 Geometric setup of 3D lid driven cavity.

with some constrains that $s_0 = s_3 = s_5 = s_7 = 0$ for conserved moments, $s_8 = s_6 = s_4$, $s_{11} = s_9$, $s_{12} = s_{10}$, $s_{15} = s_{14} = s_{13}$, and $s_{18} = s_{17} = s_{16}$.

The equilibrium values of moments are given as

$$e^{eq} = -11\rho + \frac{19}{\rho}\left(j_x^2 + j_y^2 + j_z^2\right),$$

$$\varepsilon^{eq} = 3\rho - \frac{11}{2\rho}\left(j_x^2 + j_y^2 + j_z^2\right)$$

$$j_x^{eq} = j_x, \quad j_y^{eq} = j_y, \quad j_z^{eq} = j_z$$

$$q_x^{eq} = -\frac{2}{3}j_x, \quad q_y^{eq} = -\frac{2}{3}j_y, \quad q_z^{eq} = -\frac{2}{3}j_z$$

$$p_{xx}^{eq} = \frac{1}{3\rho}\left[2j_x^2 - \left(j_y^2 + j_z^2\right)\right], \quad p_{ww}^{eq} = \frac{1}{\rho}\left[j_y^2 - j_z^2\right]$$

$$p_{xy}^{eq} = \frac{1}{\rho}j_x j_y, \quad p_{yz}^{eq} = \frac{1}{\rho}j_y j_z, \quad p_{xz}^{eq} = \frac{1}{\rho}j_x j_z$$

$$\pi_{xx}^{eq} = -\frac{1}{2}p_{xx}^{eq}, \quad \pi_{ww}^{eq} = -\frac{1}{2}p_{ww}^{eq}$$

$$m_x^{eq} = m_y^{eq} = m_z^{eq} = 0$$

The bulk velocity ζ and kinematic viscosity υ are

$$\zeta = \frac{2}{9}\left(\frac{1}{s_1} - \frac{1}{2}\right)$$

$$\upsilon = \frac{1}{3}\left(\frac{1}{s_9} - \frac{1}{2}\right) = \frac{1}{3}\left(\frac{1}{s_{13}} - \frac{1}{2}\right)$$

2.3 Boundary Conditions

The geometric setup of the present lid-driven cavity is shown in Fig. 1. Both moving and stationary boundary conditions are encountered. For stationary boundary, the boundary conditions proposed in [23] and [24] are adopted to calculate the unknown particle density distribution functions, i.e.,

$$f_i\left(\vec{x},t\right) = f\left(\vec{x},-\vec{e}_i,t\right) + \frac{\omega_i}{C}\vec{e}_i \cdot \vec{Q} \tag{10}$$

where $\vec{Q}$ is the force like corrector to enforce the required momentum. This corrector applies only to the unknown particle density distribution functions at the boundary.

This formulation delivers good results for square cavity flow. However, for deep cavity, slow convergence was found with the above boundary conditions [10, 17]. Thus, as suggested by Hou *et al*. [9], along the moving lid, the equilibrium distribution functions are imposed to speed up the convergence.

3. GPU IMPLEMENTATION

The computations are conducted on nVIDIA GeForce Titan, Tesla C2050 and GeForce GTX 560Ti. The major GPU computing platform adopted is the nVIDIA GeForce Titan, which is the latest Kepler architecture with computing capability of 3.5 TFlops under single precision. The processor is composed of five graphic processing clusters (GPC) out of which four GPC have three streaming multiprocessors (SM) and one GPC has two multiprocessors. These SM are further broken down into thread (or stream) processors. On the whole, Titan GPU is composed of 2688 processors. Titan makes use of GPU boost to dynamically adjust its clock speeds to optimize performance. The memory bandwidth is 288.4 Gigabyte per second and the available amount of memory is 6 Gigabyte.

The other two graphic cards, along with CPU which are used to compare performance are GeForce GTX 560Ti and Tesla C2050 with 336 and 448 processors respectively. GeForce GTX 560Ti, being the least expensive can easily be included in a standard workstation. Tesla C2050 is designed especially for scientific computations thereby delivers about 515 GFLOPS in double precision calculations. Table 1 shows the hardware specification of adopted graphic processing units.

Figure 2 shows the GPU kernel implementation for current LBM simulations. First, all the variables related to particle distribution function, density and velocity are declared on both host and device memory but are initialized on host memory before the execution of kernel functions. After initialization, the values of these variables are transferred from host memory to device memory for kernel computations. Kernel functions executed contain four major parts: streaming, boundary condition, collision and macro-computing step. After the computations, the value of these variables stored in the device memory are then transferred back to the host memory to get the results for macroscopic properties such as velocity and pressure.

For GPU implementation, there are several types of memory can be used on the device, i.e. GPU, such as global memory, constant memory, shared memory and register. In general, these can be categorized as on-chip and off-chip memories. Global memory is off-chip memory and has the biggest memory size. Another advantage of using global memory is that the memory can be accessed from every block. On the other hand, the on-chip memory are small and can only be accessed

https://doi.org/10.1017/jmech.2017.70 Published online by Cambridge University Press

Table 1 Different graphic processor unit specifications.

Device	Cuda cores	Memory	Bandwidth	GFLOPS (SP),(DP)
GTX 560 Ti	384	2.GB	128 GB/sec	1263.4, ——
TESLA C2050	448	3.GB	384 GB/sec	1030.4, 515.2
GTX TITAN	2688	6.GB	288.4 GB/sec	4500.0, 1300

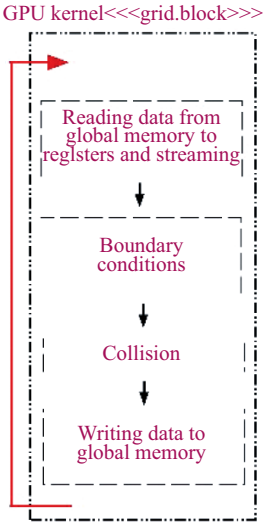


Fig. 2 Implementation of GPU simulations.

by thread within the block. However, the latency of on-chip memory is lower and the bandwidth is higher than off-chip memory, so that the data transmission on the on-chip memory is faster than the off-chip memory. The influences of using different memory during computations will be addressed in the next section.

Procedures of kernel operations in Fig. 2 using on chip memory are described below [19].

1. Transferring data from global memory to on chip memory: The kernel begins by copying the density distribution functions from global memory to on-chip memory. There are two possibilities.
   (a) At the same index of array, direct reading from global memory to on-chip memory.
   (b) Streaming is executed, while reading the data. This is achieved by adding an offset [14], depending on the direction of particle velocity, to the index of array to perform the streaming operation.
2. Boundary conditions: Boundary condition proposed by Ho *et al*. [23] and Hou *et al*. [9] are adopted.
3. Collision: Before collision operation, the macroscopic quantities ρ and $\rho\vec{u}$ are computed. Using these quantities, the collision operators are performed, as are indicated in the right hand side of Eq. 1 and Eq. 7. Since the local computations are requited for the collision operator, each cell can execute this process independently.
4. Transferring data from on chip memory to global memory: At the end of the kernel operation, the post collision density distribution function will be copied to global memory. Again, there are two possibilities.
   (a) Streaming is executed, while transferring the data from global memory to on-chip memory. This is achieved by adding an offset [14], depending on the direction of particle velocity, to the index of array to perform the streaming operation.
   (b) At the same index of array, direct copy from on-chip memory to global memory.

Therefore, the kernel operations can have two alternatives, which are 1(a)-2-3-4(a) (Method A) and 1(b)-2-3-4(b) (Method B). Streaming is executed, while transferring the data from global memory to on-chip memory. This is achieved by adding an offset [14], depending on the direction of particle velocity, to the index of array to perform the streaming operation. Method A performs streaming during the writing on chip memory data to global memory by adding an offset depending on the direction of particle velocity to the index of array, Alternatively, Method B executes streaming during loading global memory data to the on chip memory by adding an offset to the index of array to perform the streaming operation.

The performance using on-chip memory and global memory is also explored. Here, global memory scheme means that each thread requests data from global memory directly without pre-loading the entire data to the on chip memory. Also, in this study, the influence of the number of threads in a block and the number of blocks in each device are further investigated.

4. RESULTS

4.1 Numerical Results

The present LBM numerical procedures are validated against the existing benchmark solutions of lid driven cavity [2, 25] at Reynolds number being 1000. Number of lattice nodes for two dimensional simulations are taken as 1024^2, while the three dimensional grid size is 128^3. Two dimensional SRT and MRT results are in excellent agreement (see Fig. 3) with the pioneering work of Ghia *et al*. [2]. Also, three dimensional simulations (see Fig. 4) are compared against the benchmark solution of Albensoeder *et al*. [25] and are in well conformity. The streamline distributions of 2D and 3D lid driven cavity flows are shown in Figs. 5 and 6, respectively.

4.2 Parallel Performance

Here, attention is directed to explore the GPU implementation performance. Numerical simulations are

https://doi.org/10.1017/jmech.2017.70 Published online by Cambridge University Press

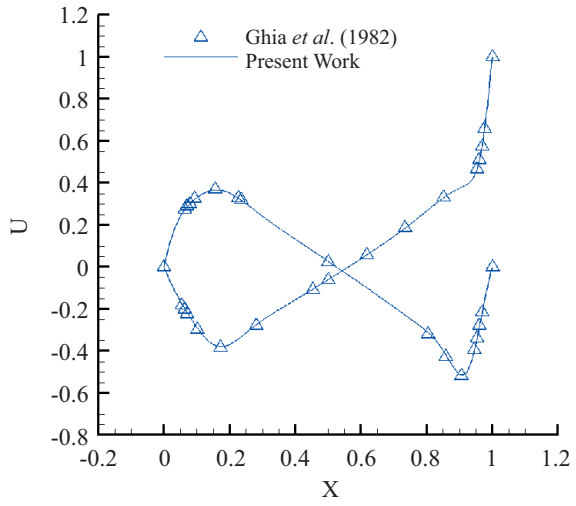


Fig. 3 Comparison with Ghia *et al*. [2] at Re = 1000, 2D cavity.

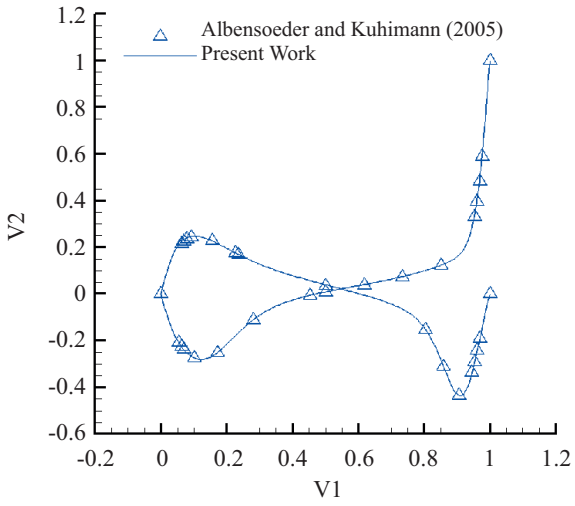


Fig. 4 Comparison with Albensoeder *et al*. [25] at Re = 1000, 3D cavity.

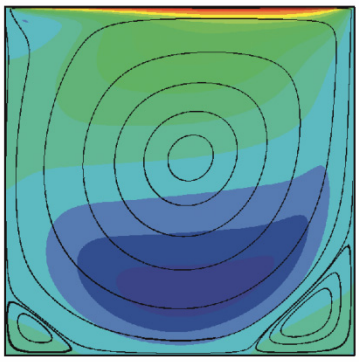


Fig. 5 Streamline distribution at Reynolds number Re = 1000, 2D cavity.

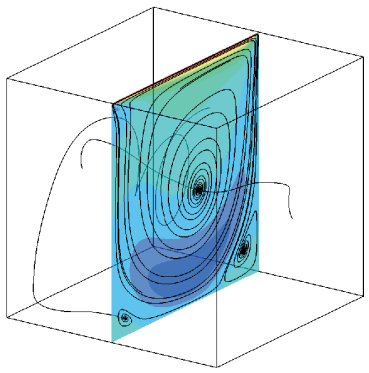


Fig. 6 Streamline distribution at Reynolds number Re = 1000, 3D cavity.

performed on lid driven cavity flow at Reynolds number being 1000. Each numerical experiment executes 5000 iterations and the computational time is used to explore the performance. Number of lattice nodes for two dimensional simulations are taken as 1024^2 while the three dimensional grid size is 128^3. The GPU platforms include Intel Core-i7 990 CPU and three different GPU as mentioned earlier, i.e. NVIDIA GeForce Titan, GTX 560Ti, and Tesla C2050. Most of the computations are executed using Titan GPU, unless stated otherwise. Here, the performance is measured based on the MLUPS (Million Lattice site Update Per Second).

The available shared memory size and register number in Titan device are respectively 48 KB and 64 KB per streaming multi-processor. For the present 3D lid driven computations with 128^3 grid, the register usages are 70 and 100 per thread for SRT and MRT schemes. For block with 128 threads, each streaming multi-processor can compute 7 and 5 blocks for SRT and MRT schemes, respectively based on the available register number in the steaming multi-processor. To further explore the influence of the grid sizes on the memory usage and hence performance, various grid densities, i.e. 32^3 to 256^3 are examined and the results are shown in Table 2, where the block size adopted is 32. Since every individual grid is computed on every respective thread, so the number of register requested for each thread is constant. The influence of grid density is only on the number of computed blocks based on the prescribed number of thread per block. Thus, the overall performance does not deteriorate.

Influence of the block size on the performance of LBM methods is investigated first. Here, the performance using SRT and MRT are explored and the streaming operating is based on Method B. Both the global memory and the on chip memory are adopted and the results are shown in Tables 3 to 4. It is clear that the parallel performance scales with the block size. This is true for both MRT and SRT LBM schemes and using single and double precisions. Also, the performance using global memory deteriorates greatly when MRT scheme is used, which is due to the scheme requesting more information from the global memory than its SRT counterpart. On the hand, using on chip memory the difference using MRT and SRT is not significant.

Further examination of the memory management can

https://doi.org/10.1017/jmech.2017.70 Published online by Cambridge University Press

Table 2 Influence of grid size on the performance of different LBM schemes - 3D lid driven cavity (SP: single precision, DP: double precision, Grid: 128^3, Block size: 32).

Grid size	32^3	64^3	128^3	256^3
(SP、SRT) On chip Memory	439.4	740.6	795.3	800.3
(SP、MRT) On chip Memory	411.6	717.8	789.7	735.9
(DP、SRT) On chip Memory	356.7	555.2	593.4	535.7
(DP、MRT) On chip Memory	343.9	516.6	552.1	516.2

Table 3 Influence of block size of the performance different LBM schemes and memory management- 2D lid driven cavity - on chip memory: streaming with Method B (SP: single precision, DP: double precision).

Block size (SP)	16	32	64	128	256	512
(SRT) Global Memory	262.8	511.5	900.8	1100.2	1066.7	1044.3
(MRT) Global Memory	112.8	215.7	350.8	355.8	299.2	186.5
(SRT) On chip	1117.9	1802.0	2510.2	2886.0	2805.1	2663.1
(MRT) On chip	993.0	1701.5	2400.4	2705.7	2655.2	2574.3
Block size (DP)	16	32	64	128	256	512
(SRT) Global Memory	202.3	371.4	541.8	556.2	545.2	490.6
(MRT) Global Memory	88.5	165.8	233.8	232.7	219.4	148.8
(SRT) On chip	812.3	1156.43	1620.0	1614.6	1451.8	1330.4
(MRT) On chip	750.6	1092.2	1451.6	1469.4	1350.6	1261.5

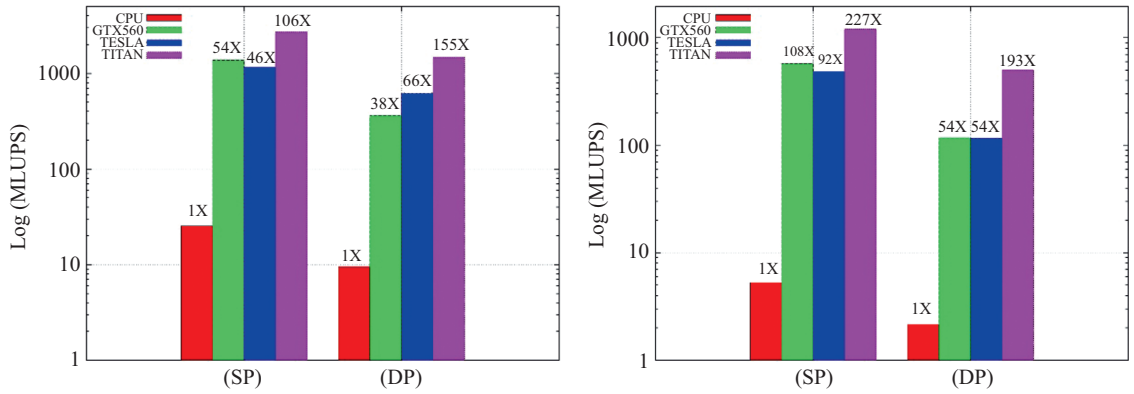
Table 4 Influence of block size on the performance of different LBM schemes and memory management- 3D lid driven cavity - on chip memory: streaming with Method B (SP: single precision, DP: double precision).

Block size (SP)	16	32	64	128
(SRT) Global Memory	103.1	188.1	279.4	279.6
(MRT) Global Memory	59.8	98.7	116.7	118.1
(SRT) On chip	479.0	795.3	1149.5	1192.3
(MRT) On chip	430.9	789.7	1036.9	1032.1
Block size (DP)	16	32	64	128
(SRT) Global Memory	91.5	150.7	191.0	188.8
(MRT) Global Memory	42.8	71.5	75.1	75.5
(SRT) On chip	395.6	593.4	707.4	691.2
(MRT) On chip	395.4	552.1	536.0	509.6

Table 5 Influence of memory management on the performance of different LBM schemes- 2D and 3D lid driven cavity (SP: single precision, DP: double precision)

2D	SRT (SP)	MRT (SP)	SRT (DP)	MRT (DP)
Global Memory	1100.2	355.8	556.2	232.7
On chip (Method A)	1508.7	1489.9	796.2	795.4
On chip (Method B)	2886.0	2705.7	1614.6	1469.4
3D	SRT (SP)	MRT (SP)	SRT (DP)	MRT (DP)
Global Memory	279.6	118.1	188.8	75.5
On chip (Method A)	718.2	695.5	356.2	336.9
On chip (Method B)	1192.3	1032.1	691.2	509.6

https://doi.org/10.1017/jmech.2017.70 Published online by Cambridge University Press



(a) 2D lid-driven cavity flow

(b) 3D lid-driven cavity flow

Fig. 7 MLUPS performances in single and double precision (SP and SD) for MRT scheme obtained by using different platforms.

be referred to Table 5. Here, using global memory and on chip memory are addressed and the block size adopted is 128. The performance using global memory has been indicated. Here, the difference using on chip memory is addressed. It is clear streaming using offset reading is much better than adopting offset writing and the gain ranges from 50% to 100%. Even so, on chip Method A is still better than using global memory alone.

Finally, comparisons using different GPU platforms are shown in Fig. 7. Here, on chip Method B and MRT LBM are adopted for all the platforms. For single precision, performance of GTX 560Ti is slightly better than Tesla C2050. This is consistent with the GPU specifications demonstrated in Table 1. However for double precision computations, Tesla C2050 is better than its GTX 560Ti counterpart for two dimensional case. However, for three dimensional flow, GTX 560Ti unexpectedly equals Tesla C2050. Titan as expected outperforms other devices, and attains 227 and 193 speedup over its Intel Core i7-990 CPU counterpart and four times faster than GTX 560Ti and Tesla C2050 for three dimensional cavity flow simulations respectively with single and double precisions.

5. CONCLUSIONS

Lattice Boltzmann methods with single and multi-relaxation time are adopted to simulate two and three-dimensional flows within lid driven cavity on graphic processing unit. The computational platform consists of nVIDIA GeForce Titan, Tesla C2050 and GeForce GTX 560Ti. Reference solutions within two and three-dimensional lid driven cavities are used to validate the numerical procedure and good agreements are obtained. It is also found that the MLUPS performance scales with the block sizes. This is true for both MRT and SRT LBM schemes and using single and double precisions. Also, the performance using global memory deteriorates greatly when MRT scheme is used, which is due to the scheme requesting more information from the global memory than its SRT counterpart. However, the difference using MRT and SRT is not significant using on chip memory. Also, streaming operation performed using offset data reading during different memory transfer is much better than employing offset data writing and this applies to both SRT and MRT LBM. Finally, comparisons using different GPU platforms indicate that Titan as expected outperforms other devices, and attains 227 and 193 speedup over its Intel Core i7-990 CPU counterpart and four times faster than GTX 560Ti and Tesla C2050 for three dimensional cavity flow simulations respectively with single and double precisions.

6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the supports by the Taiwan National Science Council and Low carbon energy research center of National Tsing Hua University and the computational facilities provided by the Taiwan National Center for High-Performance Computing.

REFERENCES

1. Shankar, P. N. and Deshpande, M. D., "Fluid Mechanics in the Driven Cavity," *Annual Review of Fluid Mechanics,* **32**, 93 (2000).
2. Ghia, U., Ghia, K. N. and Shin, C. T., "High-Resolutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method," *Journal of Computational Physics*, **48**, 387 (1982).
3. Schreiber, R. and Keller, H. B., "Driven Cavity Flows by Effcient Numerical Techniques," *Journal of Computational Physics*, **49**, 310 (1983).
4. Iwatsu, R., Ishii, K., Kawamura, T., Kuwahara, K. and Hyun, J. M., "Numerical Simulation of Three-Dimensional Flow Structure in a Driven Cavity," *Fluid Dynamics Research*, **5**, 173 (1989).
5. Guj, G. and Stella, F., "A Vorticity-Velocity Method

https://doi.org/10.1017/jmech.2017.70 Published online by Cambridge University Press

for the Numerical Solution of 3D Incompressible Flows," *Journal of Computational Physics*, **106**, 286 (1993).

6. Mei, R., Shyy, W., Yu, D. and Luo, L. S., "Lattice Boltzmann Method for 3-D Flows with Curved Boundary," *Journal of Computational Physics*, **161**, 680 (2000).
7. Feldman, Y. and Gelfgat, A. Y., "Oscillatory Instability of a Three-Dimensional Lid-Driven Flow in a Cube," *Physics of Fluids*, **22**, 093602 (2010).
8. Liberzon, A., Feldman, Y. and Gelfgat, A. Y., "Experimental Observation of the Steady-Oscillatory Transition in a Cubic Lid-Driven Cavity," *Physics of Fluids*, **23**, 084106 (2011).
9. Hou, S., Zou, Q., Chen, S., Doolean, G. and Cogley, A. C., "Simulation of Cavity Flow by the Lattice Boltzmann Method," *Journal of Computational Physics*, **118**, 329 (1995).
10. Lin, L. S., Chen, Y. C. and Lin, C. A., "Multi Relaxation Time Lattice Boltzmann Simulations of Deep Lid Driven Cavity Flows at Different Aspect Ratios," *Computers & Fluids*, **45**, 233 (2011).
11. Ghadyani, M. and Esfahanian, V., "A More Robust Compressible Lattice Boltzmann Model by Using the Numerical Filters," *Journal of Mechanics*, **30**, 515 (2014).
12. Lin, S. Y. *et al*., "A Unified Wall-Boundary Condition for the Lattice Boltzmann Method and Its Application to Force Evaluation," *Journal of Mechanics*, **31**, 55 (2015).
13. *Cuda C Programming Guide 5.0*, http://developer.nvidia.com/cuda-gpus (2012)
14. Obrecht, C., Kuznik, F., Tourancheau, B. and Roux, J. J., "A New Approach to the Lattice Boltzmann Method for Graphics Processing Units," *Computers and Mathematics with Applications*, **61**, 3628 (2011).
15. Kuznik, F., Obrecht, C., Rusaouen, G. and Roux, J. J., "LBM Based Flow Simulation Using GPU Computing Processor," *Computers and Mathematics with Applications*, **59**, 2380 (2010).
16. Tölke, J., "Implementation of a Lattice Boltzmann Kernel Using the Compute Unified Device Architecture Developed by nVIDIA," *Computing and Visualization in Science*, **13**, 29 (2010).
17. Lin, L. S., Chang, H. W. and Lin, C. A., "Multi Relaxation Time Lattice Boltzmann Simulations of Transition in Deep 2D Lid Driven Cavity Using GPU," *Computers & Fluids*, **80**, 381 (2013).
18. Chang, H. W., Hong, P. Y., Lin, L. S. and Lin, C. A., "Simulations of Flow Instability in Three Dimensional Deep Cavities with Multi Relaxation Time Lattice Boltzmann Method on Graphic Processing Units," *Computers & Fluids*, **88**, 866 (2013).
19. Hong, P. Y., Huang, L. M., Lin, L. S. and Lin, C. A., "Scalable Multi-Relaxation-Time Lattice Boltzmann Simulations on Multi-GPU Cluster," *Computers & Fluids*, **110**, 1 (2015).
20. Qian, Y. H., d'Humières, D. and Lallemand, P., "Lattice BGK Models for Navier-Stokes Equation," *Europhys Lett*, **17**, 479 (1992).
21. Humières, D., Ginzburg, I., Krafczyk, M., Lallemand, P. and Luo, L. S., "Multiple-Relaxation-Time Lattice Boltzmann Models in Three Dimensions," *Philosophical Transactions of the Royal Society of London A*, **360**, 437 (2002).
22. Lallemand, P. and Luo, L. S., "Theory of the Lattice Boltzmann Method: Dispersion, Dissipation, Isotropy, Galilean Invariance, and Stability," *Physical Review E*, **61**, 6546 (2000).
23. Ho, C. F., Chang, C., Lin, K. H. and Lin, C. A., "Consistent Boundary Conditions for 2D and 3D Laminar Lattice Boltzmann Simulations," *CMES-Computer Modeling in Engineering and Sciences*, **44**, 137 (2009).
24. Chang, C., Liu, C. H. and Lin, C. A., "Boundary Conditions for Lattice Boltzmann Simulations with Complex Geometry Flows," *Computers and Mathematics with applications*, **58**, 940 (2009).
25. Albensoeder, S. and Kuhlmann, H. C., "Accurate Three-Dimensional Lid-Driven Cavity Flow," *Journal of Computational Physics,* **206**, 536 (2005).

(Manuscript received January 9, 2017,
accepted for publication June 13, 2017.)

https://doi.org/10.1017/jmech.2017.70 Published online by Cambridge University Press