International Journal of Heat and Mass Transfer 109 (2017) 577–588

Contents lists available at ScienceDirect

International Journal of Heat and Mass Transfer

journal homepage: www.elsevier.com/locate/ijhmt

Accelerated lattice Boltzmann simulation using GPU and OpenACC with data management

A. Xu, L. Shi, T.S. Zhao *

Department of Mechanical and Aerospace Engineering, The Hong Kong University of Science and Technology, Hong Kong, China

ARTICLE INFO

Article history:
Received 1 December 2016
Received in revised form 9 February 2017
Accepted 13 February 2017

Keywords:
GPU computing
OpenACC
Lattice Boltzmann method
Heat and mass trasnfer

ABSTRACT

We assess the performance of the combined Open Accelerator (OpenACC) programming standard and graphics processing unit (GPU) accelerator for lattice Boltzmann (LB) simulations of fluid flow, heat and mass transfer. By optimizing the data layout, minimizing the memory access frequency, and adjusting the number of gangs and vector length, we show that the enhanced parallel computations can result in orders of magnitudes of speedup relative to the serial implementation of the LB algorithm. Based on such implementations, benchmark quality results are obtained with fine grid of 2049^2 for both two-dimensional lid driven cavity flow with Reynolds number up to 7500, and double diffusive cavity flow with solute Rayleigh number up to 10^8.

1. Introduction

The consistent demand for higher simulation accuracy and larger computing capability has driven the prosperous of high performance computing community [1]. During the past decades, the performance of central processing unit (CPU) microprocessor chip has been exponentially growing and follows the Moore's law, which indicates that the chip's performance will double every two years. As the microprocessor reduces its physical size to tens of nanometer, the Moore's law seems nearing its end [2]. On the other hand, utilizing graphics processing unit (GPU) as computational accelerators has attracted great attention due to its combined tremendous computing power and superior memory bandwidth. For example, the memory bandwidth is 480 gigabyte per second (GB/s) and the peak double precision floating point performance can reach 2910 giga floating-point operations per second (GFLOPS) for the newest NVIDIA® Tesla® K80 GPU accelerator, which is an order of magnitude higher than that for the newest Intel® Core™ i7 CPU processor.

The lattice Boltzmann (LB) method, originated from cellular automata and also derived from the Boltzmann kinetic theory, has been proven to be a promising tool for simulating fluid flows and associated transport phenomena. It is based on mesoscopic kinetic equations describing the evolution of fluid parcels, and boundary conditions can be easily implemented, thus making it suitable for simulating complex fluid systems, such as gas-liquid two-phase flow [3–6], particulate flow [7–10], flow in porous media [11–13], and so on. More importantly, the locality nature of extensive computations makes the LB method favorable for massively parallel programming, which is crucial for practical engineering applications.

Parallel computing framework utilizing GPUs includes Open Computing Language (OpenCL), Compute Unified Device Architecture (CUDA), and Open Accelerator (OpenACC). OpenCL is an open standard for programming heterogeneous platforms consisting of CPUs and GPUs. It specifies C/C++ programming language, and programmers must explicitly manage the way that a problem is decomposed to the hardware. CUDA works with programming languages such as C, C++ and Fortran. Although it reduces the effort in graphics programming compared to OpenCL, it only runs on NVIDIA's graphics card. These two programming standards require substantial changes in the original code, thus threatening the code correctness, portability, and maintainability. OpenACC is a high level, platform independent, and directive based programming standard being jointly developed by NVIDIA, Cray, PGI, and CAPS. Following hints provided by programmers as annotations to the original code, compute intensive calculations are offloaded to the accelerator device to utilize the superior computing capabilities of the accelerator.

A number of studies have been conducted to utilize the computing power of GPUs to accelerate LB simulations [14–19]. However, most of the previous studies were based on CUDA acceleration, which may pose severe restriction on the target hardware. For example, Sailfish [20], a free computational fluid dynamics solver based on lattice Boltzmann method, serves as a reference GPU

* Corresponding author.
E-mail address: metzhao@ust.hk (T.S. Zhao).

http://dx.doi.org/10.1016/j.ijheatmasstransfer.2017.02.032
0017-9310/

implementation. It is based on CUDA/OpenCL programming standard and implements LB models for single- and multi-component fluid flows. To the best of our knowledge, very few implementations of LB method incorporating OpenACC acceleration have been described in the literature [21], In Ref. [21], a D2Q37 lattice model was adopted for the two-dimensional simulation of convective turbulence; while in our work, the most widely used D2Q9 lattice model and D2Q5 lattice model will be adopted for the two-dimensional simulation of laminar fluid flows and heat/mass transfer, respectively. The benefit of using simpler lattice model is that memory storage is reduced, while there is no loss of simulation accuracy for the laminar flow simulations. The immature of OpenACC application in LB simulation may be mainly due to the perception that simplicity and portability of OpenACC come at a price of degrading parallel performance. This motivates us to address implementation issues when using OpenACC to accelerate LB simulations, i.e., optimizing the data layout to meet the requirement of coalescence memory access pattern, minimizing the memory access frequency by fusion the collision and streaming subroutine, and adjusting number of gangs and vector length to maximize utilize specific hardware architecture. We show that ultra-high computational performance can be achieved with proper implementations, including simulations involving coupled fluid flow, heat and mass transfer processes. By adopting OpenACC accelerated LB simulations, highly refined computations can be made. To demonstrate this point, three two-dimensional benchmark problems, namely, lid driven cavity problem, buoyancy driven cavity problem, and double diffusive cavity problem, with the grid size up to 2049^2 will be presented.

2. Numerical method

2.1. LB model for fluid flow

Fluid flows described by the incompressible Navier-Stokes equations can be written as

$$\nabla \cdot \mathbf{u} = 0 \tag{1a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} \tag{1b}$$

where $\mathbf{u}$ is the fluid velocity, ρ is the fluid density, p is the pressure, and ν is the kinematic viscosity.

In LB method, to solve Eqs. (1a) and (1b), the evolution equation of density distribution function based on D2Q9 lattice model can be written as

$$f_i(\mathbf{x} + \mathbf{e}_i\delta_t, t + \delta_t) - f_i(\mathbf{x}, t) = -(\mathbf{M}^{-1}\mathbf{S})_{ij}\left[\mathbf{m}_j(\mathbf{x}, t) - \mathbf{m}_j^{(\mathrm{eq})}(\mathbf{x}, t)\right] \tag{2}$$

where f_i is the density distribution function, t is the time, δ_t is the time step, $\mathbf{x}$ is the fluid parcel position, and $\mathbf{e}_i$ is the discrete velocity along the ith direction. To approximate incompressible flows and minimize the round-off error, we only consider density fluctuation $\delta\rho$ in various parts of equilibrium following Luo et al. [22]. Then, the equilibrium moments $\mathbf{m}^{(\mathrm{eq})}$ is given by

$$\mathbf{m}^{(\mathrm{eq})} = \left[\delta\rho,\ -2\delta\rho + \frac{3}{\rho_0}(j_x^2 + j_y^2),\ \delta\rho - \frac{3}{\rho_0}(j_x^2 + j_y^2), j_x,\ -j_x,\ j_y,\right.$$
$$\left. -j_y,\ \frac{1}{\rho_0}(j_x^2 - j_y^2),\ \frac{1}{\rho_0}j_x j_y\right]^T \tag{3}$$

The relaxation matrix is given by $\mathbf{S} = \mathrm{diag}(s_\rho, s_e, s_\epsilon, s_j, s_q, s_j, s_q, s_\nu, s_\nu)$, where the relaxation parameters are given as $s_\rho = s_j = 0, s_e = s_\epsilon = s_\nu = 1/\tau_f, s_q = 8(2\tau_f - 1)/(8\tau_f - 1)$, and τ_f is determined by the kinematic viscosity of the fluids.

The density variation $\delta\rho$ and velocity $\mathbf{u}$ are obtained from

$$\delta\rho = \sum_{i=0}^{8} f_i, \quad \mathbf{u} = \sum_{i=0}^{8} \mathbf{e}_i f_i \tag{4}$$

and the kinetic viscosity ν is given by

$$\nu = \frac{1}{3}c^2\left(\tau_f - \frac{1}{2}\right)\delta_t \tag{5}$$

where $c = \delta_x/\delta_t$ is lattice constant, and $c = \delta_x = \delta_t = 1$ is adopted in this work.

2.2. LB model for fluid flow and heat transfer

In incompressible thermal flows, temperature variation will cause density variation thus resulting in buoyancy effect. Following the Boussinesq approximation, the temperature can be treated as a passive scalar, and its influence to the velocity field is through the buoyancy term [23,24]. Then, the governing equations can be written as

$$\nabla \cdot \mathbf{u} = 0 \tag{6a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho_0}\nabla p + \nu \nabla^2 \mathbf{u} + g\beta_T(T - T_0)\hat{\mathbf{y}} \tag{6b}$$

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \kappa \nabla^2 T \tag{6c}$$

where T is the temperature, β_T is the thermal expansion coefficient, and κ is the thermal diffusivity. g is the gravity value, and $\hat{\mathbf{y}}$ is unit vector in the vertical direction. ρ_0 and T_0 are reference density and temperature, respectively.

In LB method, to solve Eqs. (6a) and (6b), the evolution equation of density distribution function is similar to Eq. (2) except with the addition of forcing term, which is written as

$$f_i(\mathbf{x} + \mathbf{e}_i\delta_t, t + \delta_t) - f_i(\mathbf{x}, t) = -(\mathbf{M}^{-1}\mathbf{S})_{ij}\left[\mathbf{m}_j(\mathbf{x}, t) - \mathbf{m}_j^{(\mathrm{eq})}(\mathbf{x}, t)\right] + \delta_t F_i' \tag{7}$$

where F_i' is the forcing term in velocity space, and is given by

$$\mathbf{F}' = \mathbf{M}^{-1}\left(\mathbf{I} - \frac{\mathbf{S}}{2}\right)\mathbf{M}\tilde{\mathbf{F}} \tag{8}$$

and the term $\mathbf{M}\tilde{\mathbf{F}}$ is given by

$$\mathbf{M}\tilde{\mathbf{F}} = \left[0,\ 6\mathbf{u}\cdot\mathbf{F},\ -6\mathbf{u}\cdot\mathbf{F},\ F_x,\ -F_x,\ F_y,\ -F_y, 2(u_xF_x - u_yF_y),\ (u_xF_y + u_yF_x)\right]^T \tag{9}$$

where $\mathbf{F} = g\beta_T(T - T_0)\hat{\mathbf{y}}$. The density variation $\delta\rho$ and velocity $\mathbf{u}$ are obtained from

$$\delta\rho = \sum_{i=0}^{8} f_i, \quad \mathbf{u} = \sum_{i=0}^{8} \mathbf{e}_i f_i + \frac{1}{2}\mathbf{F} \tag{10}$$

To solve Eq. (6c), the evolution equation of temperature distribution function based on D2Q5 lattice model can be written as

$$g_i(\mathbf{x} + \mathbf{e}_i\delta_t, t + \delta_t) - g_i(\mathbf{x}, t) = -(\mathbf{N}^{-1}\mathbf{Q})_{ij}\left[\mathbf{n}_j(\mathbf{x}, t) - \mathbf{n}_j^{(\mathrm{eq})}(\mathbf{x}, t)\right] \tag{11}$$

where g_i is the temperature distribution function.

The equilibrium moments $\mathbf{n}^{(\mathrm{eq})}$ is given by

$$\mathbf{n}^{(\mathrm{eq})} = [T,\ uT,\ \nu T,\ a_T T,\ 0]^T \tag{12}$$

where a_T is a constant determined by the thermal diffusivity as

$$a_T = 20\sqrt{3}\kappa - 4 \tag{13}$$

The relaxation matrix is given by $\mathbf{Q} = \mathrm{diag}(0, q_T, q_T, q_e, q_\nu)$, where the relaxation parameters are given as $q_T = 3 - \sqrt{3}$ and $q_e = q_\nu = 4\sqrt{3} - 6$.

The temperature T is obtained from

$$T = \sum_{i=0}^{4} g_i \tag{14}$$

2.3. LB model for fluid flow, heat transfer and mass transfer

In addition to temperature variation, concentration variation will also cause density variation, thus resulting in buoyancy effect. Analogs to the Boussinesq approximation, the concentration can also be treated as a passive scalar, and its influence to the velocity field is through the buoyancy term [25]. Then, the governing equations can be written as

$$\nabla \cdot \mathbf{u} = 0 \tag{15a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho_0}\nabla p + \nu \nabla^2 \mathbf{u} + g\beta_T (T - T_0)\hat{\mathbf{y}} + g\beta_s (C - C_0)\hat{\mathbf{y}} \tag{15b}$$

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \kappa \nabla^2 T \tag{15c}$$

$$\frac{\partial C}{\partial t} + \mathbf{u} \cdot \nabla C = D \nabla^2 C \tag{15d}$$

where C is the concentration, β_s is the concentration expansion coefficient, and D is the mass diffusivity. ρ_0, T_0 and C_0 are reference density, temperature, and concentration, respectively.

In LB method, to solve Eqs. (15a) and (15b), the evolution equation of density distribution function is the same as Eq. (7), and $\mathbf{F}$ in Eq. (9) is given as $\mathbf{F} = g\beta_T (T - T_0)\hat{\mathbf{y}} + g\beta_s (C - C_0)\hat{\mathbf{y}}$; to solve Eq. (15c), the evolution equation of temperature distribution function is the same as Eq. (11); to solve Eq. (15d), the evolution equation of concentration distribution function based on D2Q5 lattice model can be written as

$$h_i(\mathbf{x} + \mathbf{e}_i \delta_t, t + \delta_t) - h_i(\mathbf{x}, t) = -(\mathbf{N}^{-1}\mathbf{Q})_{ij}\left[\mathbf{n}_j(\mathbf{x}, t) - \mathbf{n}_j^{(\mathrm{eq})}(\mathbf{x}, t)\right] \tag{16}$$

where h_i is the concentration distribution function. Here, the equilibrium moments $\mathbf{n}^{(\mathrm{eq})}$ is given by

$$\mathbf{n}^{(\mathrm{eq})} = [C,\ uC,\ vC,\ a_s C,\ 0]^T \tag{17}$$

where a_s is a constant determined by the mass diffusivity as

$$a_s = 2\sqrt{3}D - 4 \tag{18}$$

The relaxation matrix is given by $\mathbf{Q} = \mathrm{diag}(0, q_C, q_C, q_e, q_v)$, where the relaxation parameters are given as $q_C = 3 - \sqrt{3}$ and $q_e = q_v = 4\sqrt{3} - 6$.

The temperature C is obtained from

$$C = \sum_{i=0}^{4} h_i \tag{19}$$

3. OpenACC implementation and optimization

3.1. GPU programming using OpenACC

CPUs are optimized for low latency access to cached data, while GPUs are optimized for data parallelism and hide latency with intensive computations. In OpenACC parallel execution model, both computation and data are offloaded from the CPU host to the GPU accelerator, so that the combined tremendous computing power and superior memory bandwidth of GPU can be utilized.

To ensure the correctness of the code employing OpenACC acceleration, an incremental approach to accelerate the code should be taken. The first step is to assess the numerical algorithm and estimate the running time each subroutine takes. In LB method, the standard implementation consists of four subroutines: collision, streaming, bounce-back, and calculating macro variables as illustrated in Fig. 1. Among them, the collision subroutine takes the most computational time due to extensive updates of distribution function along nine directions in two-dimensional simulations or nineteen directions in three-dimensional simulations at a fluid node; in the streaming subroutine, the information of distribution function at a node is simply transferred to its neighboring, and there is no tedious mathematical computations; the bounce-back subroutine only works at the boundary nodes to mimic various boundary conditions; although the subroutine of calculating macro variables also takes effect at fluid nodes in the whole computational domain, its computational cost is much lower compared to the collision subroutine due to much simpler mathematical manipulations. The second step is to use OpenACC *parallel loop* directive to parallelize computational intensive loops. Thanks to the locality nature of LB method, the subroutines of collision and calculating macro variables completely use local node information. Although the streaming subroutine involves data transfer, the data dependence problem can be avoided by using two distribution functions for data swapping. The third step is to optimize data locality and minimize data transfer between CPU host and GPU accelerator through PCIe. After initializing the macroscopic field information (such as velocity $\mathbf{u}$, density ρ, temperature T and concentration C) and the mesoscopic distribution functions (such as density distribution function f, temperature distribution function g and concentration distribution function h), they can be copied into GPU memory space, and only copied back to CPU until the end of iterations. This can be achieved by the data clauses including *copy*, *copyin*, *copyout*, and *create*. Otherwise, at each iterative step, all the data will be transferred between CPUs and GPUs by default, which may consume more time than the computation itself.

To characterize the parallel performance of LB implementation, we adopt the metrics of million lattice updates per second (MLUPS), which is defined as

$$\mathrm{MLUPS} = \frac{\mathrm{meshsize} \times \mathrm{iterationsteps}}{\mathrm{runningtime} \times 10^6} \tag{20}$$

For a given mesh size, we fixed the iteration steps as 20,000, and measure the running time. Less running time results in higher MLUPS, which also means the better parallel performance. All the simulations in the present work use double-precision floating point arithmetic, which ensures the simulation accuracy.

Three benchmark problems, namely, lid driven cavity problem [26,27], buoyancy driven cavity problem [28,29], and double diffusive cavity problem [30,31] are adopted to investigate the parallel computing performance of OpenACC based LB simulation involving multiphysics processes. Fig. 2 shows the computing performance of utilizing OpenACC with the programming effort mentioned

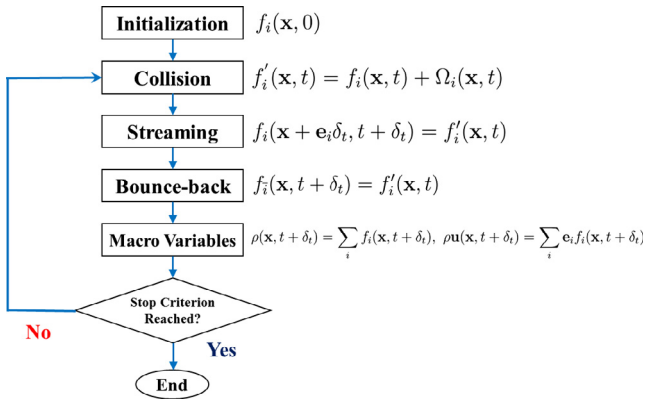
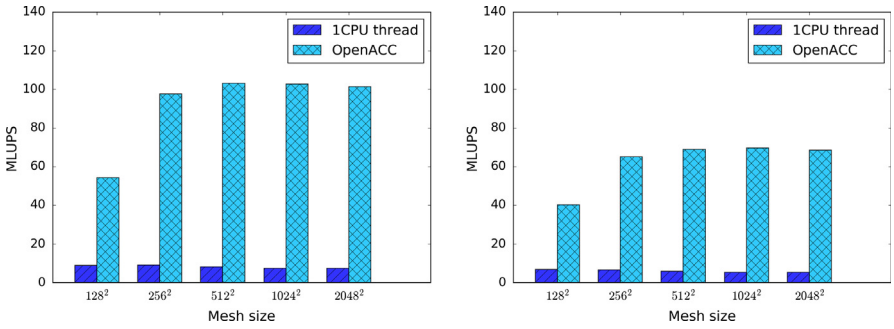
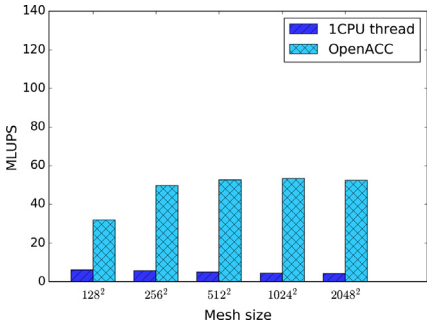


Fig. 1. Schematic illustration of LB flow chat.



(a) Lid driven cavity problem

(b) Buoyancy driven cavity problem



(c) Double diffusive cavity problem

Fig. 2. Performance comparisons between CPU and basic OpenACC implementation.

above, and around 12–13 times speedup can be obtained comparing with the original code running on CPU. It is interesting to notice that the computing performance is much higher for the isothermal fluid flow problem than that for the coupled fluid flow, heat and mass transfer problem. This is because LB algorithm is memory bandwidth limited algorithm, and the usage of more global arrays to store distribution functions will appear as a limiting step. Specifically, we need three distribution functions f, g, and h to store the information for flow field, temperature field, and concentration field in the double diffusive cavity problem. In contrast, only one distribution function f is needed in the lid driven cavity problem. It should also be pointed out that, this is the most basic effort of porting to GPU platform, and we can further boost the parallel computing performance by applying the improvement strategies mentioned in Sections 3.2, 3.3 and 3.4.

3.2. Optimize data layout

Usually, in CPU based LB implementation, the data layout of distribution function belongs to array of structure (AoS) because CPUs can make use of cache memory. For example, the distribution function $f_i(\mathbf{x}, t)$ is stored with index $(i + 9 \times x + 9 \times Nx \times y)$ for the D2Q9 lattice model, as illustrated in Fig. 3. However, as GPUs adopt single instruction multiple threads (SIMT) execution model, the data layout should be changed to structure of array (SoA) to meet the requirement of coalescing memory access in OpenACC based LB implementation. For the D2Q9 lattice model, the distribution function $f_i(\mathbf{x}, t)$ is then stored with index $(x + Nx \times y + Nx \times Ny \times i)$, so that parallel threads running the same instruction can access consecutive locations in memory. Using Fortran programming language, this index formula implementation means changing array $f(i, x, y)$ to array $f(x, y, i)$; while the array $f[y][x][i]$ is changed to array $f[i][y][x]$ using C programming language. Similar index formula for the D3Q19 lattice model can be straightforwardly obtained.

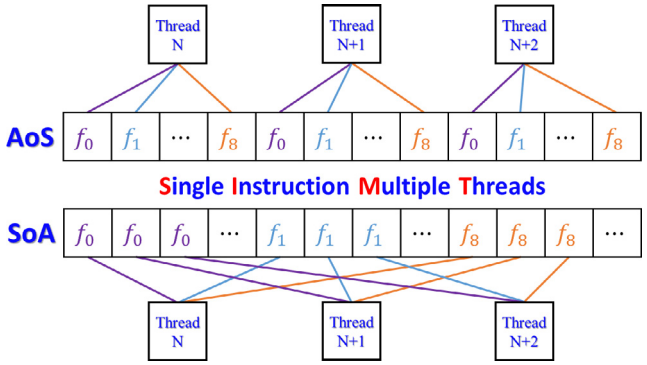
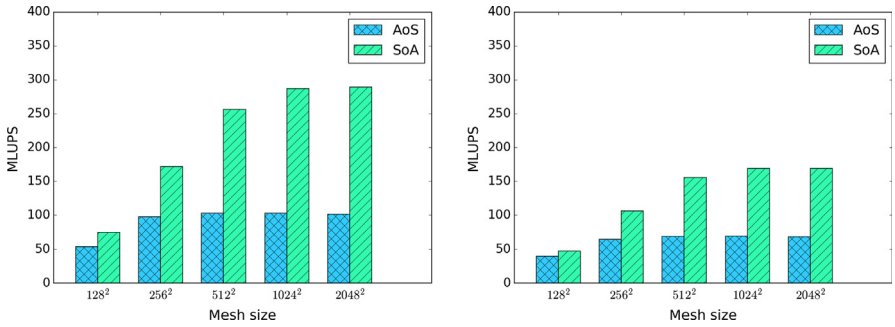


Fig. 3. Schematic illustration of AoS and SoA data layout.

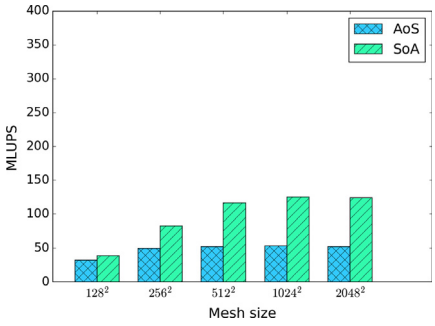
Fig. 4 compares the parallel performance of OpenACC accelerated LB implementation adopting AoS and SoA data layout. A speedup of 2.37–2.85X can be obtained after changing the data layout, which also means around 29–38 times speedup can be obtained comparing with the original code running on CPU. It is encouraging to see the effort of optimizing data layout, which practically means just changing the index of an array for distribution function and requires little programming effort, rewards nearly a threefold increase in computing performance.

3.3. Minimize memory access frequency

Accessing data in the GPU memory space may take longer time than the actual computations on these data. Thus, it is essential to minimize the frequency of accesses to global memory. In LB method, the frequency of global memory accesses can be reduced by fusing the collision and streaming subroutines. Based on the standard LB implementation described in Section 3.1, there is one read-access and one write-access in each of the two subrouti-



(a) Lid driven cavity problem (b) Buoyancy driven cavity problem



(c) Double diffusive cavity problem

Fig. 4. Performance comparisons between AoS and SoA data layout.

nes to update the distribution function. An alternative implementation is fusing the collision and streaming subroutines, then only one read-access and one write-access is needed in total. Specifically, at odd time steps, the distribution function f is read from the memory and the post-collision distribution function f' is wrote to the memory. At even time steps, the procedure is reversed. In practical implementation, we suggest to follow the flow chat shown in Fig. 5, which can avoid handling the algorithm flow control to distinguish odd or even iterations using *if* statement.

Fig. 6 compares the parallel performance between native two-steps and fusion one-step implementations for the update of distribution function. A speedup of 1.34–1.42X can be obtained after reducing memory access frequency, which also means around 39–54 times speedup can be obtained comparing with the original code running on CPU. One should keep in mind that the results in Figs. 2, 4 and 6 represent an incremental improvement in the implementation, and the left side of the bar in Figs. 4 and 6 are the same as the right side of the bar in Figs. 2 and 4, respectively.

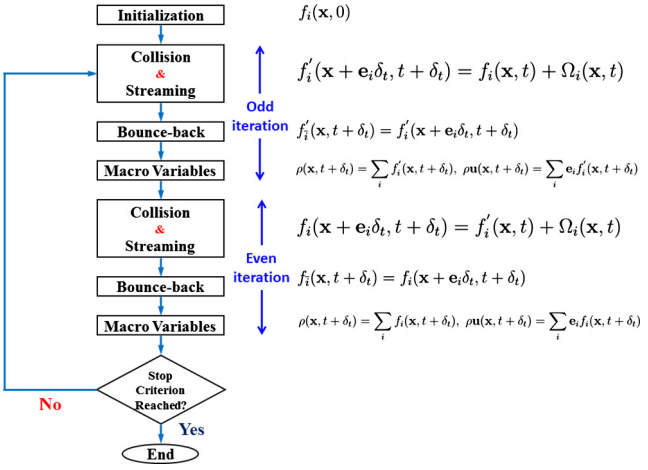
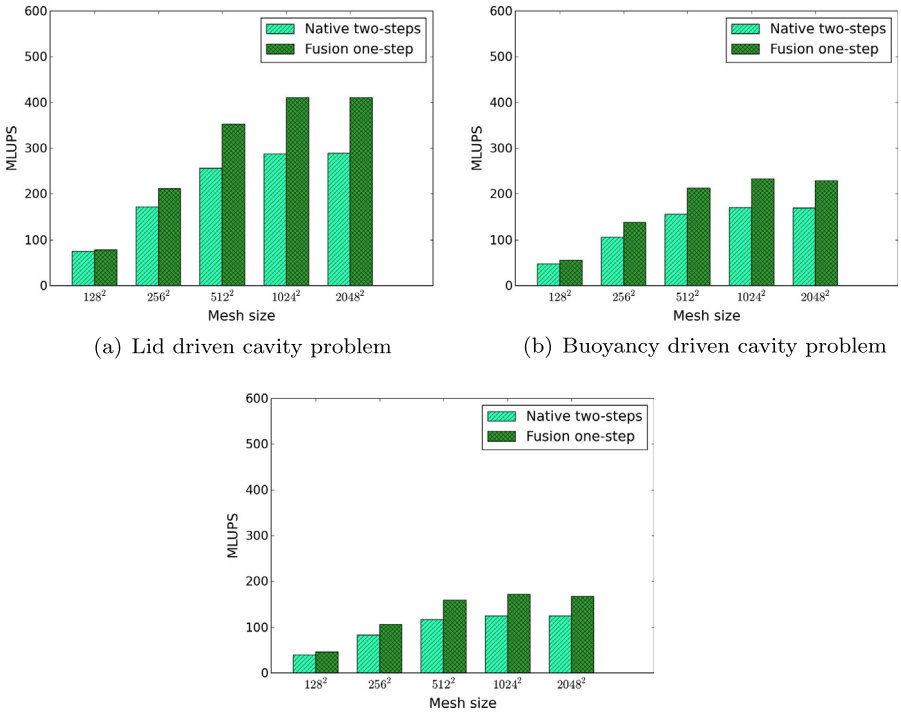


Fig. 5. Schematic illustration of LB flow chat by fusing collision and streaming subroutine.

It is also interesting to notice from the above performance comparison that MLUPS is higher when the mesh size is above 1000^2, as shown in Fig. 7 which consolidates the above-mentioned improvement strategies. This is because there is extensive computational load with a larger mesh size and GPUs can hide latency with intensive computations. In addition, the speedup for double diffusive cavity problem degrades a little compared with that for lid driven cavity problem, which is due to the increase of complexity of the numerical algorithm.

3.4. Adjust the number of gangs and vector length

In addition to *loop* directive to parallelize computational intensive loops, OpenACC execution model gives more detailed control over three levels of parallelism via *gang*, *worker*, and *vector* clauses. Gang parallelism is the highest level, and gangs work independently of each other and may not synchronize; vector parallelism has the finest granularity, indicating the data elements that operated on with the same instruction; worker parallelism sits between gang and vector levels. To further boost the parallel performance of the code based on OpenACC acceleration, the programmers may specify the number of gangs and vector length to fit a target architecture. Since the present numerical tests are performed on NVIDIA® Tesla® K20c GPU accelerator, which prefers the number of gangs and vector length to be a multiple of 32, Fig. 8 gives the parallel performance by varying the number of gangs between 2^7 and 2^{11}, and vector length between 2^7 and 2^{10}. Here, we tested five different mesh sizes with the same total computational loads, but with different aspect ratios including 1:16, 1:4, 1:1, 4:1, and 16:1. Results show that an optimal choice for the number of gangs equals 2048, and the vector length equals 1024; in addition, the aspect ratio of the computational domain has minor effects on the parallel performance.



(a) Lid driven cavity problem (b) Buoyancy driven cavity problem

(c) Double diffusive cavity problem

Fig. 6. Performance comparisons between native two-steps and fusion one-step implementation.

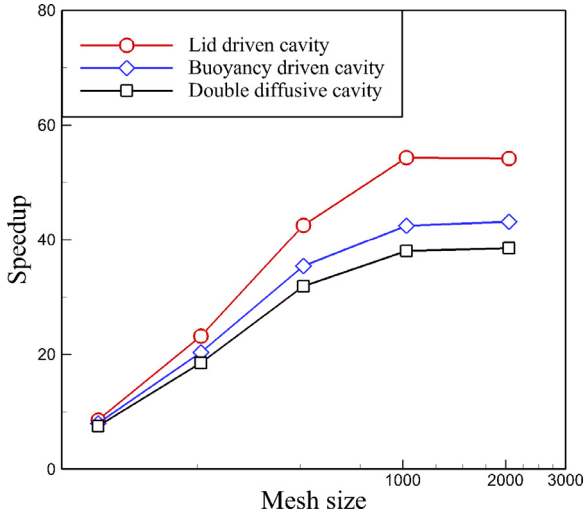


Fig. 7. Correlation of speedup versus mesh size for three benchmark problems.

As a summary, Table 1 gives the maximize parallel performance of OpenACC accelerated LB algorithm implementations by adopting the above-mentioned improvements. The mesh size is 2048×2048, and MLUPS for GPU implementations are chosen after adjusting the number of gangs and vector length on NVIDIA® Tesla® K20c accelerator. A speedup around 49–64 times can be achieved compared with the serial implementations.

4. Numerical benchmark

4.1. Lid driven cavity problem

The flow domain is a unit square, with the top wall moving at a constant velocity (U_0), and the other three walls are stationary. The dimensionless number characterize the system is the Reynolds number (Re), which is defined as

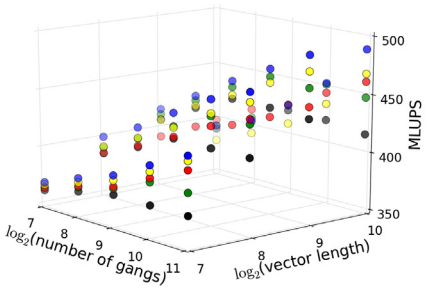
$$Re = \frac{U_0 L_0}{\nu} \tag{21}$$

where U_0 is the characteristic velocity and L_0 is the characteristic length of the system. In this work, we provide results for steady states with $Re = 5000$ and 7500. The criterion for reaching steady state is given by

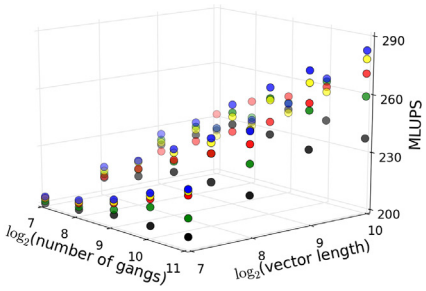
$$\frac{\sum_i \|\mathbf{u}(x_i, t + 2000\delta_t) - \mathbf{u}(x_i, t)\|_2}{\sum_i \|\mathbf{u}(x_i, t)\|_2} < 10^{-9} \tag{22}$$

where $\|\mathbf{u}\|_2$ denotes L^2 norm of $\mathbf{u}$. The characteristic velocity is chosen as $U_0 = 0.1$. Fig. 9 shows stream-function, vorticity and pressure fields obtained on grid 2049^2, which qualitatively agrees with previous study. We also provide benchmark results on the extreme of stream-function ($|\psi|_{\max}$), and the vorticity ($|\omega|$) at the same position for both the primary vortex and the lower-right secondary vortex, as shown in Table 2. Existing data from Ghia et al. [26] using vorticity-stream function formulation together with coupled strongly implicit multigrid method; Bruneau et al. [27] using finite differences discretization and on a multigrid solver with a cell-by-cell relaxation procedure; and Zhu et al. [32] using discrete unified gas kinetic scheme are also provided as comparison. In addition, the asymptotic values f_∞ are used as the reference values to compute the relative error, which are then used to estimate the order of accuracy n for LB simulation.

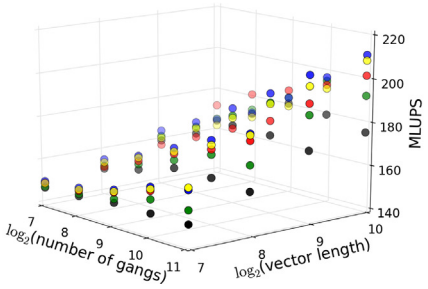
It should be noted that there are controversy results on the Reynolds number when the first Hopf bifurcation occurs: Ghia et al. [26] provided steady results up to $Re = 10,000$, while Bruneau et al. [27] obtained the first Hopf bifurcation at Reynolds number close to $Re = 8000$. Our data shows at Re = 10,000 the solution cannot converge to steady states, thus, steady solution of Reynolds number up to $Re = 7500$ are provided as benchmark. It is also worth mentioning that the simulation accuracy can be further



(a) Lid driven cavity problem



(b) Buoyancy driven cavity problem



(c) Double diffusive cavity problem

Fig. 8. Performance comparisons between various number of gangs and vector length. Five different mesh sizes are tested: red circles represent mesh size 512×8192; yellow circles represent mesh size 1024×4096; blue circles represent mesh size 2048×2048; green circles represent mesh size 4096×1024; black circles represent mesh size 8192×512. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

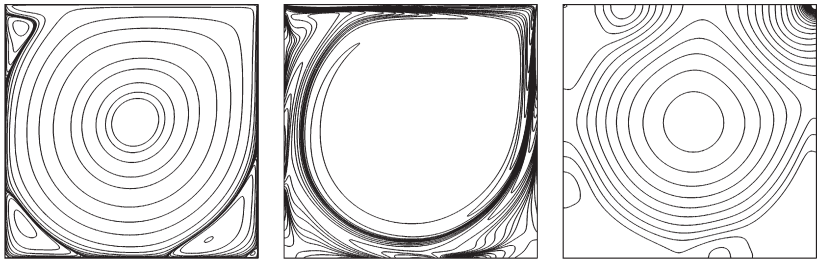
Table 1
Performance comparison between CPU and GPU based LB algorithm implementation.

	CPU (MLUPS)	GPU (MLUPS)	Speedup
Lid driven cavity problem	7.6	488.7	64X
Buoyancy driven cavity problem	5.3	282.6	53X
Double diffusive cavity problem	4.3	210.8	49X

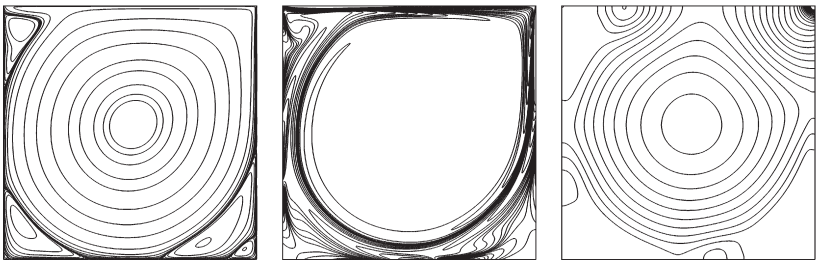
improved by adopting adaptive mesh refinement method [33,34], however, using non-uniform meshes may spoil the parallel efficiency of LB algorithm and the trade-offs deserve further investigations.

4.2. Buoyancy driven cavity problem

The flow domain is a unit square, with left and right vertical walls maintained at a constant high temperature and low temperature, respectively; and the top and bottom horizontal walls are adiabatic. All four walls are stationary. The dimensionless numbers



(a) $\mathrm{Re} = 5000$



(b) $\mathrm{Re} = 7500$

Fig. 9. From left to right: steady solution of stream-function, vorticity and pressure fields on grid 2049×2049.

Table 2
Benchmark solutions of the primary vortex and lower-right secondary vortex.

Re	Ref.	Grid	$\lvert\psi\rvert_{\max}$	$\lvert\omega\rvert$	$\lvert\psi\rvert_{\max}\times 10^3$	$\lvert\omega\rvert$
5000	Ghia [26]	257^2	0.118966	1.86016	3.08358	2.66354
	Bruneau [27]	2048^2	0.12197	1.9327	3.0706	2.7244
	Zhu [32]	512^2	0.122387	1.945667	3.090044	2.777576
	Present	513^2	0.119942	1.90664	3.00970	2.66394
	Present	1025^2	0.121113	1.92401	3.04203	2.73440
	Present	2049^2	0.121676	1.93239	3.05795	2.74009
		∞	0.121864	1.93519	3.06327	2.74199
		n	1.68	1.68	1.67	2.68
7500	Ghia [26]	257^2	0.119976	1.87987	3.28484	3.49312
	Present	513^2	0.119562	1.88478	3.15172	3.15388
	Present	1025^2	0.121023	1.90645	3.19014	3.21385
	Present	2049^2	0.121715	1.91682	3.20898	3.22024
		∞	0.121946	1.92028	3.21527	3.22237
		n	1.69	1.68	1.67	2.50

characterize the system is the thermal Rayleigh number (Ra_T) and Prandtl number (Pr), which are defined as

$$Ra_T = \frac{g\beta_T \Delta T L_0^3}{\nu\kappa}, \quad Pr = \frac{\nu}{\kappa} \tag{23}$$

where L_0 is the characteristic length of the system. In this work, we provide results for steady states with $Ra_T = 10^7$, and 10^8. The Prandtl number is fixed as $Pr = 0.71$. The criterion for reaching steady state is given by

$$\begin{aligned} &\frac{\sum_i \|\mathbf{u}(x_i, t+2000\delta_t) - \mathbf{u}(x_i,t)\|_2}{\sum_i \|\mathbf{u}(x_i,t)\|_2} < 10^{-9}, \\ &\frac{\sum_i |T(x_i, t+2000\delta_t) - T(x_i,t)|}{\sum_i |T(x_i,t)|} < 10^{-7} \end{aligned} \tag{24}$$

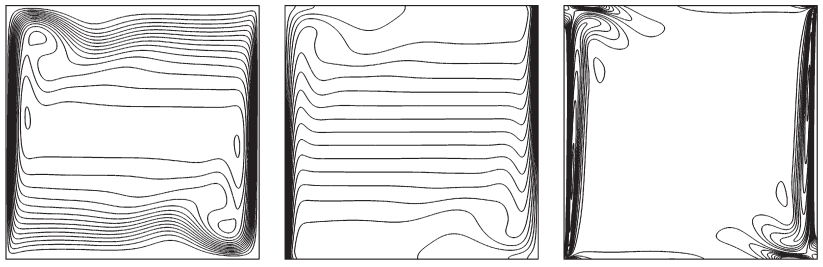
where $\|\mathbf{u}\|_2$ denotes L^2 norm of $\mathbf{u}$. Fig. 10 shows stream-function, temperature and vorticity fields obtained on grid 2049^2, which qualitatively agrees with previous study.

We also provide benchmark results on the stream-function at the cavity center $|\psi|_{\text{mid}}$, the maximum horizontal velocity $u_{\max}$ and its position y along the vertical mid-plane, the maximum vertical velocity $v_{\max}$ and its position x along the horizontal mid-plane, as shown in Table 3. Existing data from Quéré [28] using pseudo-spectral Chebyshev algorithm; and Contrino et al. [29] using multiple-relaxation-time LB method are also provided as comparison.

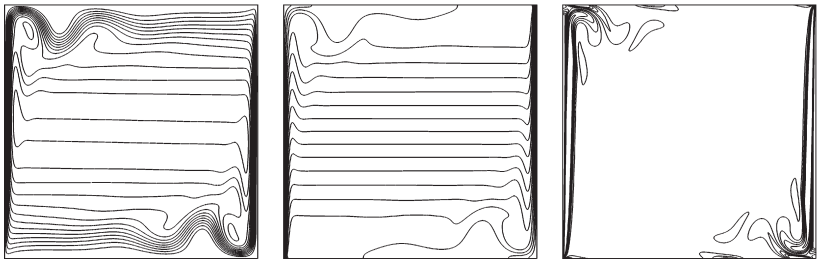
In addition to the benchmark result for hydrodynamic intensities, we also provide benchmark results for Nusselt numbers, including the volume average Nusselt number $\langle Nu\rangle$, the average Nusselt number along the hot wall Nu_0, the average Nusselt number along the vertical mid-plane $Nu_{1/2}$, the maximum local Nusselt number along the hot wall $Nu_{\max}$ and its location y, and the minimum local Nusselt number along the hot wall $Nu_{\min}$, as shown in Table 4.

4.3. Double diffusive cavity problem

The flow domain is a unit square, with left vertical walls maintained at a constant high temperature and low concentration; and right vertical walls maintained at a constant low temperature and



(a) $\mathrm{Ra}_T = 10^7$



(b) $\mathrm{Ra}_T = 10^8$

Fig. 10. From left to right: steady solution of stream-function, temperature and vorticity fields on grid 2049×2049.

Table 3
Benchmark solutions of the hydrodynamic intensities.

Ra_T	Ref.	Grid	$\lvert\psi\rvert_{\text{mid}}$	$u_{\max}$	y	$v_{\max}$	x
10^7	Quéré [28]	128^2	29.3617	148.5954	0.879	699.1795	0.021
	Contrino [29]	2043^2	29.3653	148.5852	0.8793	699.3224	0.0213
	Present	513^2	29.37105	148.31041	0.88012	698.70233	0.02047
	Present	1025^2	29.36462	148.52300	0.87951	699.25729	0.02098
	Present	2049^2	29.36267	148.57189	0.87921	699.33944	0.02123
		∞	29.36202	148.58821	0.87911	699.36685	0.02131
		n	1.90	2.05	1.67	2.30	1.67
10^8	Quéré [28]	128^2	52.32	321.9	0.928	2222.	0.012
	Contrino [29]	2043^2	52.3508	321.9063	0.9279	2222.3279	0.0120
	Present	513^2	52.37456	310.43507	0.91910	2219.03321	0.01267
	Present	1025^2	52.33688	318.34510	0.92634	2222.31042	0.01220
	Present	2049^2	52.32652	320.98831	0.92753	2223.46603	0.01196
		∞	52.32306	321.87064	0.92793	2223.85179	0.01188
		n	1.95	1.85	2.24	1.82	1.65

Table 4
Benchmark solutions of the Nusselt numbers.

Ra_T	Ref.	Grid	$\langle Nu\rangle$	Nu_0	$Nu_{1/2}$	$Nu_{\max}$	$Nu_{\min}$
10^7	Quéré [28]	128^2	–	16.523	16.523	39.39	1.366
	Contrino [29]	2043^2	16.5231	16.5230	16.5231	39.3950	1.3659
	Present	513^2	16.53658	16.53243	16.53766	39.91395	1.39477
	Present	1025^2	16.52673	16.52345	16.52710	39.48729	1.37206
	Present	2049^2	16.52414	16.52229	16.52428	39.40215	1.36731
		∞	16.52328	16.52190	16.52334	39.37374	1.36572
		n	1.97	2.39	1.97	2.13	2.10
10^8	Quéré [28]	128^2	–	30.225	30.225	87.24	1.919
	Contrino [29]	2043^2	30.2251	30.2241	30.2251	87.2454	1.9195
	Present	513^2	30.30100	30.31372	30.30588	93.88534	2.12199
	Present	1025^2	30.24443	30.24051	30.24571	88.61612	1.97001
	Present	2049^2	30.23013	30.22650	30.23050	87.49182	1.93027
		∞	30.22536	30.22183	30.22542	87.11660	1.91701
		n	1.99	2.15	1.99	2.09	1.98

high concentration; and the top and bottom horizontal walls are adiabatic and impenetrable. All four walls are stationary. The dimensionless numbers characterize the system is the thermal Rayleigh number (Ra_T), solute Rayleigh number (Ra_s), Prandtl number (Pr), and Lewis number (Le), which are defined as

$$Ra_T = \frac{g\beta_T \Delta T L_0^3}{\nu\kappa}, \quad Ra_s = \frac{g\beta_s \Delta C L_0^3}{\nu D}, \quad Pr = \frac{\nu}{\kappa}, \quad Le = \frac{\kappa}{D} \tag{25}$$

where L_0 is the characteristic length of the system. In this work, we provide results for steady states with $Ra_s = 5\times10^6, 5\times10^7$, and 10^8 for $Ra_T = 10^7$; $Ra_s = 5\times10^7$ and 2×10^8 for $Ra_T = 10^8$. The Prandtl number is fixed as $Pr = 0.71$, and the Lewis number is fixed as $Le = 1$. The criterion for reaching steady state is given by

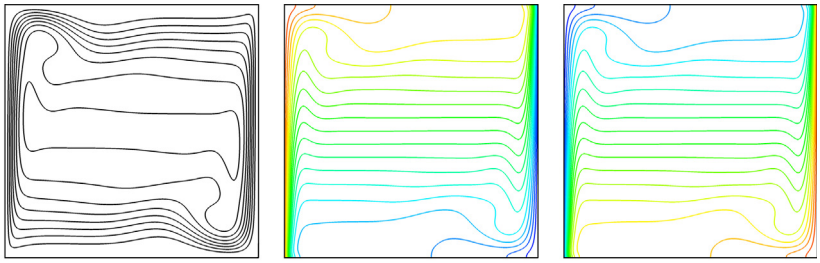
$$\begin{aligned}
&\frac{\sum_i \|\mathbf{u}(x_i, t+2000\delta_t) - \mathbf{u}(x_i,t)\|_2}{\sum_i \|\mathbf{u}(x_i,t)\|_2} < 10^{-9},\\
&\frac{\sum_i |T(x_i, t+2000\delta_t) - T(x_i,t)|}{\sum_i |T(x_i,t)|} < 10^{-7},\\
&\frac{\sum_i |C(x_i, t+2000\delta_t) - C(x_i,t)|}{\sum_i |C(x_i,t)|} < 10^{-7}
\end{aligned} \tag{26}$$

where $\|\mathbf{u}\|_2$ denotes L^2 norm of $\mathbf{u}$. Figs. 11 and 12 shows stream-function, temperature and concentration fields obtained on grid 2049^2 for $Ra_T = 10^7$ and $Ra_T = 10^8$, respectively.

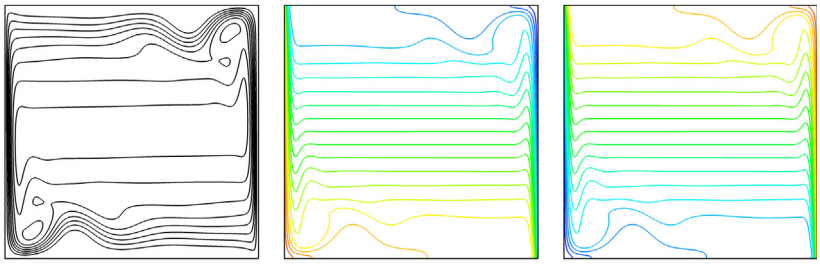
We also provide benchmark results for Sherwood numbers, including the volume average Sherwood number $\langle Sh\rangle$, the average Sherwood number along the low concentration wall Sh_0, the average Sherwood number along the vertical mid-plane $Sh_{1/2}$, the maximum local Sherwood number along the low concentration wall $Sh_{\max}$, and the minimum local Sherwood number along the low concentration wall $Sh_{\min}$, as shown in Tables 5 and 6. Existing data from Beghein et al. [30] using finite volume method with SIMPLER algorithm; and Hu et al. [31] using lattice Boltzmann flux scheme are also provided as a comparison. It is worth mentioning that previous benchmark results on the double diffusive cavity problem is quite limited, e.g., only the average Sherwood number along the low concentration wall Sh_0 is available; and there is no results are available for Ra_T up to 10^8. The present quantitative results will certainly enrich our knowledge on the double diffusive cavity problem, and we encourage follow up works to provide validation using other advanced numerical techniques.

5. Conclusion

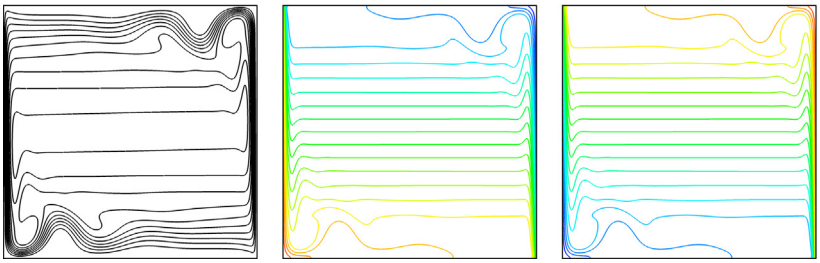
In this work, we have presented improved implementation of OpenACC to accelerate LB simulations involving fluid flow, heat and mass transfer. The results demonstrate that using OpenACC allows a speed up around 50–60 times for multiphysics LB simulation. The applications of OpenACC accelerated LB simulation to two-dimensional lid driven cavity problem, buoyancy driven cavity problem, and double diffusive cavity problem using fine grid provide benchmark quality results. The extensions to much more memory consuming three-dimensional implementation will be pursued in future work.



(a) $\mathrm{Ra}_s = 5 \times 10^6$

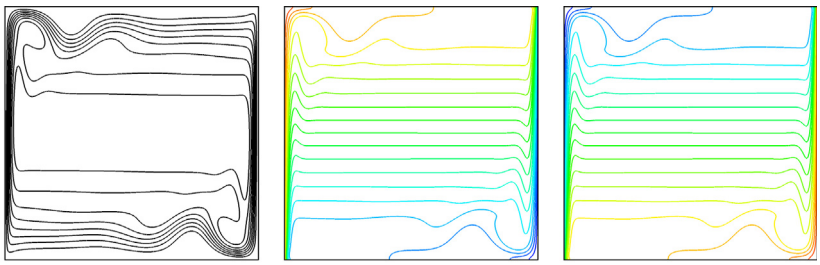


(b) $\mathrm{Ra}_s = 5 \times 10^7$

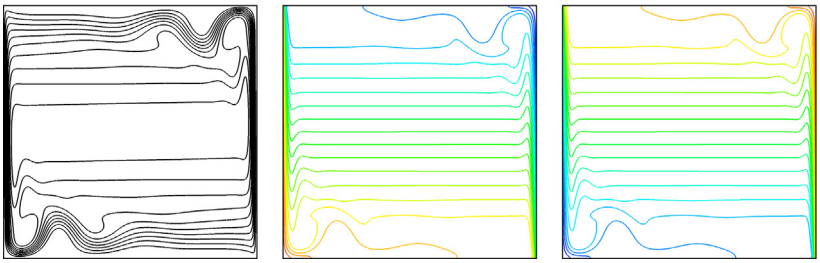


(c) $\mathrm{Ra}_s = 10^8$

Fig. 11. From left to right: steady solution of stream-function, temperature and concentration fields on grid 2049×2049 ($Ra_T = 10^7$).



(a) $\mathrm{Ra}_s = 5 \times 10^7$



(b) $\mathrm{Ra}_s = 2 \times 10^8$

Fig. 12. From left to right: steady solution of stream-function, temperature and concentration fields on grid 2049×2049 ($Ra_T = 10^8$).

Table 5
Benchmark solutions of the Sherwood numbers ($Ra_T = 10^7$).

Ra_s	Ref	Grid	$\langle Sh \rangle$	Sh_0	$Sh_{1/2}$	Sh_{max}	Sh_{min}
5×10^6	Beghein [30]	45^2	–	−13.6	–	–	–
	Hu [31]	200^2	–	−13.64	–	–	–
	Present	513^2	−13.72797	−13.72392	−13.72898	−1.23026	−30.92539
	Present	1025^2	−13.72223	−13.71955	−13.72258	−1.23235	−30.89116
	Present	2049^2	−13.72068	−13.71922	−13.72082	−1.23314	−30.89247
		∞	−13.72016	−13.71911	−13.72023	−1.23340	−30.89291
		n	1.96	2.73	1.95	1.79	3.10
5×10^7	Beghein [30]	45^2	–	−23.7	–	–	–
	Hu [31]	200^2	–	−23.85	–	–	–
	Present	513^2	−23.85876	−23.85898	−23.86024	−1.66060	−64.37414
	Present	1025^2	−23.83032	−23.82603	−23.83076	−1.67243	−63.75814
	Present	2049^2	−23.82303	−23.82018	−23.82318	−1.67586	−63.64995
		∞	−23.82060	−23.81823	−23.82065	−1.67701	−63.61385
		n	1.99	2.19	1.99	1.92	2.20
10^8	Present	513^2	−29.48261	−29.49357	−29.48709	−1.86018	−85.86658
	Present	1025^2	−29.43027	−29.42622	−29.43143	−1.88097	−84.43427
	Present	2049^2	−29.41701	−29.41348	−29.41735	−1.88674	−84.15782
		∞	−29.41258	−29.40923	−29.41265	−1.88867	−84.06556
		n	1.99	2.16	1.99	1.95	2.15

Table 6
Benchmark solutions of the Sherwood numbers ($Ra_T = 10^8$).

Ra_s	Ref	Grid	$\langle Sh \rangle$	Sh_0	$Sh_{1/2}$	Sh_{max}	Sh_{min}
5×10^7	Present	513^2	−25.29378	−25.29600	−25.29527	−1.71943	−69.42192
	Present	1025^2	−25.26009	−25.25574	−25.26062	−1.72925	−68.82583
	Present	2049^2	−25.25149	−25.24842	−25.25167	−1.73218	−68.72866
		∞	−25.24862	−25.24598	−25.24868	−1.73316	−68.69624
		n	1.99	2.18	1.98	1.91	2.24
2×10^8	Present	513^2	−30.30208	−30.31536	−30.30706	−1.88324	−89.30962
	Present	1025^2	−30.24448	−30.24056	−30.24577	−1.90870	−87.60812
	Present	2049^2	−30.23013	−30.22653	−30.23050	−1.91558	−87.25699
		∞	−30.22534	−30.22185	−30.22540	−1.91788	−87.13981
		n	2.00	2.16	2.00	1.96	2.11

Acknowledgements

The authors thank Professor Wei Shyy for his useful discussion. The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. 623313).

Appendix A. D2Qn lattice model

A.1. D2Q9 lattice model

For the two-dimensional D2Q9 lattice model, $\mathbf{e}_i$ can be given as

$$[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6, \mathbf{e}_7, \mathbf{e}_8] = c \begin{bmatrix} 0 & 1 & -1 & 0 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & -1 & 1 & 1 & -1 & -1 \end{bmatrix} \tag{27}$$

$\mathbf{M}$ is a 9×9 orthogonal transformation matrix given by [35]

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{28}$$

A.2. D2Q5 lattice model

For the two-dimensional D2Q5 lattice model, $\mathbf{e}_i$ can be given as

$$[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4] = c \begin{bmatrix} 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \tag{29}$$

$\mathbf{N}$ is a 5×5 orthogonal transformation matrix given by [36]

$$\mathbf{N}=\begin{bmatrix}1&1&1&1&1\\0&1&0&-1&0\\0&0&1&0&-1\\-4&1&1&1&1\\0&1&-1&1&-1\end{bmatrix} \tag{30}$$

Appendix B. Dimensional scaling

B.1. Nondimensionalization model for fluid flow

With the scalings

$$\mathbf{x}/L_0\to\mathbf{x}^*,\quad t\Big/\left(\frac{L_0}{U_0}\right)\to t^*,\quad \mathbf{u}/U_0\to\mathbf{u}^*,\\ p\Big/\left(\frac{\rho_0U_0^2}{L_0^3}\right)\to p^*,\quad \rho\Big/\left(\frac{\rho_0}{L_0^3}\right)\to\rho^* \tag{31}$$

then, Eqs. (1a) and (1b) can be rewritten in dimensionless form as

$$\nabla\cdot\mathbf{u}^*=0 \tag{32a}$$

$$\frac{\partial\mathbf{u}^*}{\partial t^*}+\mathbf{u}^*\cdot\nabla\mathbf{u}^*=-\nabla p^*+\frac{1}{Re}\nabla^2\mathbf{u}^* \tag{32b}$$

B.2. Nondimensionalization model for fluid flow and heat transfer

With the scalings

$$\mathbf{x}/L_0\to\mathbf{x}^*,\quad t\Big/\left(\frac{L_0^2}{\kappa}\right)\to t^*,\quad \mathbf{u}\Big/\left(\frac{\kappa}{L_0}\right)\to\mathbf{u}^*,\\ p\Big/\left(\frac{\rho_0\kappa^2}{L_0^2}\right)\to p^*,\quad (T-T_0)/\Delta T\to T^* \tag{33}$$

then, Eqs. 6a, 6b and 6c can be rewritten in dimensionless form as

$$\nabla\cdot\mathbf{u}^*=0 \tag{34a}$$

$$\frac{\partial\mathbf{u}^*}{\partial t^*}+\mathbf{u}^*\cdot\nabla\mathbf{u}^*=-\nabla p^*+Pr\nabla^2\mathbf{u}^*+Ra_TPrT^*\tilde{\mathbf{y}} \tag{34b}$$

$$\frac{\partial T^*}{\partial t^*}+\mathbf{u}^*\cdot\nabla T^*=\nabla^2T^* \tag{34c}$$

B.3. Nondimensionalization model for fluid flow, heat transfer and mass transfer

With the scalings

$$\mathbf{x}/L_0\to\mathbf{x}^*,\ t\Big/\left(\frac{L_0^2}{\kappa}\right)\to t^*,\ \mathbf{u}\Big/\left(\frac{\kappa}{L_0}\right)\to\mathbf{u}^*,\ p\Big/\left(\frac{\rho_0\kappa^2}{L_0^2}\right)\to p^*,\\ (T-T_0)/\Delta T\to T^*,\ (C-C_0)/\Delta C\to C^* \tag{35}$$

then, Eqs. (15a)–(15c) and (15d) can be rewritten in dimensionless form as

$$\nabla\cdot\mathbf{u}^*=0 \tag{36a}$$

$$\frac{\partial\mathbf{u}^*}{\partial t^*}+\mathbf{u}^*\cdot\nabla\mathbf{u}^*=-\nabla p^*+Pr\nabla^2\mathbf{u}^*+Ra_TPrT^*\tilde{\mathbf{y}}+Ra_s\frac{Pr}{Le}C^*\tilde{\mathbf{y}} \tag{36b}$$

$$\frac{\partial T^*}{\partial t^*}+\mathbf{u}^*\cdot\nabla T^*=\nabla^2T^* \tag{36c}$$

$$\frac{\partial C^*}{\partial t^*}+\mathbf{u}^*\cdot\nabla C^*=\frac{1}{Le}\nabla^2C^* \tag{36d}$$

References

[1] K. Mattila, T. Puurtinen, J. Hyväluoma, R. Surmas, M. Myllys, T. Turpeinen, F. Robertsen, J. Westerholm, J. Timonen, J. Comput. Sci. 12 (2016) 62–76, http://dx.doi.org/10.1016/j.jocs.2015.11.013.

[2] M.M. Waldrop, Nature News 530 (2016) 144, http://dx.doi.org/10.1038/530144a.

[3] P. Cheng, X. Quan, S. Gong, X. Liu, L. Yang, Adv. Heat Transfer 46 (2014) 187–248, http://dx.doi.org/10.1016/bs.aiht.2014.08.004.

[4] Q. Li, K. Luo, X. Li, Phys. Rev. E 87 (2013) 053301, http://dx.doi.org/10.1103/PhysRevE.87.053301.

[5] A. Xu, T. Zhao, L. An, L. Shi, Int. J. Heat Fluid Flow 56 (2015) 261–271, http://dx.doi.org/10.1016/j.ijheatfluidflow.2015.08.001.

[6] Q. Li, K. Luo, Q. Kang, Y. He, Q. Chen, Q. Liu, Prog. Energy Combust. Sci. 52 (2016) 62–105, http://dx.doi.org/10.1016/j.pecs.2015.10.001.

[7] H. Huang, X. Yang, M. Krafczyk, X.-Y. Lu, J. Fluid Mech. 692 (2012) 369–394, http://dx.doi.org/10.1017/jfm.2011.519.

[8] H. Huang, X. Yang, X.-Y. Lu, Phys. Fluids (1994-present) 26 (2014) 053302, http://dx.doi.org/10.1063/1.4874606.

[9] X. Yang, H. Huang, X. Lu, Phys. Rev. E 92 (2015) 063009, http://dx.doi.org/10.1103/PhysRevE.92.063009.

[10] A. Xu, T. Zhao, L. Shi, X. Yan, Int. J. Heat Fluid Flow 62 (2016) 560–567, http://dx.doi.org/10.1016/j.ijheatfluidflow.2016.08.001.

[11] Z. Chai, B. Shi, J. Lu, Z. Guo, Comput. Fluids 39 (2010) 2069–2077, http://dx.doi.org/10.1016/j.compfluid.2010.07.012.

[12] Z. Chai, C. Huang, B. Shi, Z. Guo, Int. J. Heat Mass Transf. 98 (2016) 687–696, http://dx.doi.org/10.1016/j.ijheatmasstransfer.2016.03.065.

[13] W.-Z. Fang, L. Chen, J.-J. Gou, W.-Q. Tao, Int. J. Heat Mass Transf. 92 (2016) 120–130, http://dx.doi.org/10.1016/j.ijheatmasstransfer.2015.08.071.

[14] J. Tölke, M. Krafczyk, Int. J. Comput. Fluid Dynam. 22 (2008) 443–456, http://dx.doi.org/10.1080/10618560802238275.

[15] N. Delbosc, J.L. Summers, A. Khan, N. Kapur, C.J. Noakes, Comput. Math. Appl. 67 (2014) 462–475, http://dx.doi.org/10.1016/j.camwa.2013.10.002.

[16] L.-S. Lin, H.-W. Chang, C.-A. Lin, Comput. Fluids 80 (2013) 381–387, http://dx.doi.org/10.1016/j.compfluid.2012.01.018.

[17] H.-W. Chang, P.-Y. Hong, L.-S. Lin, C.-A. Lin, Comput. Fluids 88 (2013) 866–871, http://dx.doi.org/10.1016/j.compfluid.2013.08.019.

[18] C. Huang, B. Shi, N. He, Z. Chai, Adv. Appl. Math. Mech. 7 (2015) 1–12, http://dx.doi.org/10.4208/aamm.2014.m468.

[19] C. Huang, B. Shi, Z. Guo, Z. Chai, Commun. Comput. Phys. 17 (2015) 960–974, http://dx.doi.org/10.4208/cicp.2014.m342.

[20] M. Januszewski, M. Kostur, Comput. Phys. Commun. 185 (2014) 2350–2368, http://dx.doi.org/10.1016/j.cpc.2014.04.018.

[21] E. Calore, J. Kraus, S.F. Schifano, R. Tripiccione, in: European Conference on Parallel Processing, Springer, pp. 613–624. http://dx.doi.org/10.1007/978-3-662-48096-0_47.

[22] L.-S. Luo, W. Liao, X. Chen, Y. Peng, W. Zhang, et al., Phys. Rev. E 83 (2011) 056710, http://dx.doi.org/10.1103/PhysRevE.83.056710.

[23] B.-F. Wang, D.-J. Ma, C. Chen, D.-J. Sun, J. Fluid Mech. 711 (2012) 27, http://dx.doi.org/10.1017/jfm.2012.360.

[24] S.-N. Xia, Z.-H. Wan, S. Liu, Q. Wang, D.-J. Sun, J. Fluid Mech. 798 (2016) 628–642, http://dx.doi.org/10.1017/jfm.2016.338.

[25] W. Shyy, M.-H. Chen, Phys. Fluids A: Fluid Dynam. (1989–1993) 3 (1991) 2592–2607, http://dx.doi.org/10.1063/1.858200.

[26] U. Ghia, K.N. Ghia, C. Shin, J. Comput. Phys. 48 (1982) 387–411, http://dx.doi.org/10.1016/0021-9991(82)90058-4.

[27] C.-H. Bruneau, M. Saad, Comput. Fluids 35 (2006) 326–348, http://dx.doi.org/10.1016/j.compfluid.2004.12.004.

[28] P. Le Quéré, Comput. Fluids 20 (1991) 29–41, http://dx.doi.org/10.1016/0045-7930(91)90025-D.

[29] D. Contrino, P. Lallemand, P. Asinari, L.-S. Luo, J. Comput. Phys. 275 (2014) 257–272, http://dx.doi.org/10.1016/j.jcp.2014.06.047.

[30] C. Beghein, F. Haghighat, F. Allard, Int. J. Heat Mass Transf. 35 (1992) 833–846, http://dx.doi.org/10.1016/0017-9310(92)90251-M.

[31] Y. Hu, D. Li, S. Shu, X. Niu, Comput. Math. Appl. 72 (2016) 48–63, http://dx.doi.org/10.1016/j.camwa.2016.04.032.

[32] L. Zhu, P. Wang, Z. Guo, J. Comput. Phys. 333 (2017) 227–246, http://dx.doi.org/10.1016/j.jcp.2016.11.051.

[33] C.-K. Kuan, J. Sim, W. Shyy, Acta. Mech. Sin. 28 (2012) 999–1021, http://dx.doi.org/10.1007/s10409-012-0126-3.

[34] C.-K. Kuan, K.-L. Pan, W. Shyy, J. Fluid Mech. 759 (2014) 104–133, http://dx.doi.org/10.1017/jfm.2014.558.

[35] P. Lallemand, L.-S. Luo, Phys. Rev. E 61 (2000) 6546, http://dx.doi.org/10.1103/PhysRevE.61.6546.

[36] J. Wang, D. Wang, P. Lallemand, L.-S. Luo, Comput. Math. Appl. 65 (2013) 262–286, http://dx.doi.org/10.1016/j.camwa.2012.07.001.