Journal of Applied Fluid Mechanics, Vol. 10, No. 1, pp. 55-67, 2017.
Available online at www.jafmonline.net, ISSN 1735-3572, EISSN 1735-3645.
DOI: 10.18869/acadpub.jafm.73.238.26431

A New Approach to Reduce Memory Consumption in Lattice Boltzmann Method on GPU

M. Sheida[1†], M. Taeibi-Rahni[1] and V. Esfahanian[2]

[1] *Sharif University of Technology, Tehran, Iran*
[2] *University of Tehran, Tehran, Iran*

†*Corresponding Author Email: sheida_mojtaba@ae.sharif.edu*

(Received April 20, 2015; accepted September 2, 2016)

ABSTRACT

Several efforts have been performed to improve LBM defects related to its computational performance. In this work, a new algorithm has been introduced to reduce memory consumption. In the past, most LBM developers have not paid enough attention to retain LBM simplicity in their modified version, while it has been one of the main concerns in developing of the present algorithm. Note, there is also a deficiency in our new algorithm. Besides the memory reduction, because of high memory call back from the main memory, some computational efficiency reduction occurs. To overcome this difficulty, an optimization approach has been introduced, which has recovered this efficiency to the original two-steps two-lattice LBM. This is accomplished by a trade-off between memory reduction and computational performance. To keep a suitable computational efficiency, memory reduction has reached to about 33% in D2Q9 and 42% in D3Q19. In addition, this approach has been implemented on graphical processing unit (GPU) as well. In regard to onboard memory limitation in GPU, the advantage of this new algorithm is enhanced even more (39% in D2Q9 and 45% in D3Q19). Note, because of higher memory bandwidth in GPU, computational performance of our new algorithm using GPU is better than CPU.

Keywords: Memory reduction; Optimization; Computational performance; Lattice boltzmann method (LBM).

NOMENCLATURE

d	dimensions
e	velocity element
f	fluid packet
i	fluid packet Index
q	total lattice cites
Re	Reynolds Number
$\vec{V}$	velocity
ρ	density
temp	temporary index
new	updated Index
t	time

1. INTRODUCTION

Recently, Lattice Boltzmann Method (LBM) has become an appropriate commonly used technique in fluid flow simulations (Higuera and Jimenez 1989, Higuera *et al.* 1989, Chen *et al.* 1992, Qian *et al.* 1992, Succi 2001, Di Rienzo *et al.* 2012). LBM has emerged from lattice-gas automata in late 1980s (Frisch *et al.* 1986, McNamara and Zanetti 1988, Succi 2001, Mattila *et al.* 2007), which was itself derived from Boltzmann equation. LBM has been applied to many thermo-fluid flow problems, such as laminar, turbulent, complex geometries, thermal, multi-component and immiscible fluids, and multiphase flows (Dawson *et al.* 1993, Shan and Chen 1993, Bailey *et al.* 2009, Rahmati *et al.* 2014).

LBM Simulation is performed in both two- and three-dimensions, where like conventional CFD approach, each lattice node discretize the media to fluid or solid nodes and the fluid packets propagate through the lattice in discrete time steps. At each lattice point, the packets collide with each other, while they are restricted locally and only depend on data from the neighboring nodes. Note, the spatial locality of LBM to data makes it a good candidate for parallelization.

In addition, simplicity in coding, easy handling of

complex geometries, and straightforward incorporation of microscopic interactions make LBM a favorable method to investigate complex fluid flows (Mattila *et al.* 2007).

Nevertheless, there are some restrictions in using LBM, e.g., boundary conditions; Mach number limitations; memory restrictions; and memory bandwidth limitations.

In LBM, the computational grid exchanges information of each node with its neighbors. Since there are many packets (with regard to the particle distribution functions), a high memory bandwidth is usually needed, which causes some hardware limitations.

Improvement of LBM computational efficiency is still an open research. Inefficiency in computations, which causes low convergence rates (Geller *et al.* 2006, Mattila *et al.* 2007) is one of LBM restrictions. There have been efforts to make this better by grid refinement (Filippova and Hänel 1998, Mattila *et al.* 2007), improved algorithms for explicit time-marching implementations (Massaioli and Amati 2002, Pohl *et al.* 2003, Mattila *et al.* 2007), and code optimization (Pohl *et al.* 2003, Velivelli and Bryden 2004, Wellein *et al.* 2006, Mattila *et al.* 2007).

Note, several types of LBM have been implemented. Generally, they are tagged in DdQq, where d is dimension (2 or 3) and q denotes the number of particle distribution functions (PDFs) (Wittmann *et al.* 2013). On the other hand, LBM contains two distinct steps: streaming and collision. In streaming step, data are coupled to and from adjacent lattice nodes, while in collision step, data are usually independent of the underlying lattice type and computations are performed in this step (Mattila *et al.* 2008).

Allocation of nodes in memory can be determined either by direct addressing of a full grid ("simple index arithmetic") or by indirect addressing (Pan *et al.* 2004, Zeiser *et al.* 2009, Wittmann *et al.* 2013) ("an index array which holds the full adjacency information of all nodes, the q−1 neighbors per node")(Wittmann *et al*. 2013).

On behalf of computational inefficiency, high memory requirement is another bottleneck in using LBM. Allocation of data can be optimized for either streaming or collision steps. To optimize for streaming step, the structure of arrays (SoA) data layout is used, where each direction of the discrete velocities is stored in an individual array. While, to optimize for collision step, array of structures (AoS) data layout is used. In AoS, only one array stores the PDFs node-wise. Note, this type of allocation of PDFs in memory causes better cache utilization (Wellein *et al.* 2006, Wittmann *et al.* 2013).

The differences between basic algorithms are related to their treatment of this data dependence. So far, for implementation of LBM, several algorithms have been identified, e.g., the Lagrangian, compressed grid (shift), swap, two-lattice, and two-step algorithm (Massaioli and Amati 2002, Pohl *et al.* 2003, Mattila *et al.* 2007). Of course, each of these has its own advantages and disadvantages.

Note, in the basic two-step algorithm, collision and streaming steps are handled separately, while in one-step algorithm, collision and streaming steps are combined to one step. To avoid data dependencies, in both of these algorithms two lattices are used.

Lagrangian approach was presented by Massaioli and Amati (Massaioli and Amati 2002). They compared it with two-step algorithm, in which slight improvement in performance was observed.

To reduce memory traffic within the system, the compressed grid (shift) algorithm was presented by Pohl *et al.* to reduce high memory consumption of two-lattice algorithm by combining collision and streaming steps. They compared their new algorithm with the two-lattice algorithm (Pohl *et al.* 2003) and observed that both methods have the same performance with almost half the memory requirements in shift algorithm. The effect of data layouts on computational performance was studied by Wellein *et al.* (Wellein *et al.* 2006). They found that data layout had an important role in achieving high performance.

On the other hand, the swap algorithm was proposed for implementation of LBM by Guo, Zhao *et al.* 2004. They compared the computational performance and memory consumption of the four main algorithms, namly, shift, swap, two-lattice, and two-step with a collision-optimized data layout.

Another study was performed by Mattila *et al.* (Mattila *et al.* 2007). They compared implementations of the two-step, one-step, compressed grid, swap, and Langrangian algorithms with different addressing, e.g., direct, semi-direct, and indirect.

In the present work, however, a new algorithm has been introduced, which approximately halfs the memory requirement in LBM, while preserves the simplicity of LBM. This approach is more applicable for GPU, because of onboard memory restriction in GPU by less performance reduction.

2. LBM

In 1986, Frisch *et al.* introduced LBM, based on lattice-gas cellular automata (LGCA) to simulate a real fluid flow. But It can, also be directly derived from Boltzmann equation (Abe 1997, He and Luo 1997, Philippi *et al.* 2006, Shan *et al.* 2006). Nowadays, it has become a very acceptable method in many research and industrial flows.

In Boltzmann equation, a set of fluid particle populations is used. These populations follow a Maxwellian distribution function. The number of

populations has been defined by Hermite–Gauss integration. By applying the Chapman–Enskog expansion, the incompressible Navier–Stokes equations can be recovered from Boltzmann equation in the low Mach number limit (Zhang and Seaton 1994, Coppens and Froment 1995, Alvarado *et al.* 1997).

Finite difference scheme has been mostly used to solve the space and time parts of lattice Boltzmann equation (LBE). In this regard, both time and space phases are discretized and time is advanced in lattice-Boltzmann simulation, where each time step is divided into two parts: collision step, where momentum is exchanged between the fluid packets at each node and streaming step, where the fluid packets are transferred to the neighbor nodes along their paths.

In contrast to conventional CFD methods, in each cell, LBM uses a set of particle distribution functions (PDF) to describe the fluid flow. A PDF is defined as the expected value of particles in a volume located at the lattice position X with the lattice velocity e_i. Computationally, LBM is based on a uniform grid of square/cubic (2D/3D) cells, which are updated in each time step, using an information exchange with nearest neighbor cells. Structurally, this is equivalent to an explicit time stepping for a regular finite difference scheme. For LBM, the lattice velocities determine the finite difference stencil, where index i represents an entry in the stencil.

In LBM, solution is started by an initial configuration, which is done using initial macroscopic values. On each time step, the fluid packets move from their lattice nodes to the neighboring nodes according to their velocities. When all fluid packets are transferred to lattice nodes, the fluid packets of each lattice node collide together and change their velocities. The average motion of the particles describes the macroscopic behavior of the flow.

By an appropriate collision term, using LBE, the Navier–Stokes equations with second order accuracy (Chen and Doolen 1998, Succi 2001) in the macroscopic limit can be satisfied.

There are many collision operators, in which the single-relaxation-time approximation is the simplest one, where in kinetic theory, it is known as the BGK approximation.

In addition, besides the main collision approximation, there are several collision rules, where three of them are used in most simulations, namely: 1) standard single-relaxation, which is used for simulating fluid flows, 2) a bounce-back collision rule, which is used for simulating solid-fluid boundaries (Coppens and Froment 1995), and 3) a collision rule, which is used for simulating pressure-controlled boundary conditions.

During the streaming step, almost all fluid packets, except the stationary ones (usually it is with zero index), go from each node and propagates to the corresponding incoming fluid packet locations in the neighboring nodes. Then, these incoming fluid packets and the zero index fluid packet are used in the next collision step.

2.1 Standard LBM

There is a standard implementation of LBM, which is known as two step-two lattice method. In this method, collision and streaming are literally as two separate steps and two lattices, denoted by A and B, where A stores the distribution values of the nodes (often denoted as f_i) and B stores the post-collision, pre-streaming, values (often denoted f_i^{temp}).

In each time step, after performing the collision step, the new computed PDF's are stored in lattice B at the same index position. In the streaming step, the following PDFs in lattice B are replaced with the neighbor nodes in lattice A. Note, streaming step has not any computational task and it just contains the data replacement. By performing this step, current time step is finished and is ready for the next time step.

Because of several transfer of data to and from memory at each time step and since two arrays of data is needed to store, the memory bandwidth intensive and high memory consumption are two major restrictions of this algorithm. On the other hand, there are two ways in Implementations of LBM namely, "Push" and "Pull". In "push", collision step is executed before streaming step, and in "pull", it is vice versa. A suitable scheme for most implementations is "pull". Note, LBM implementation contains several sections, data storage model, data layouts, and addressing schemes.

To accomplish the streaming step, accessing the distribution values of the neighboring nodes is needed. In this regard, there are three addressing schemes namely: direct, semi-indirect, and indirect, where the easiest one to implement is direct addressing. the explicit time marching is considered by direct address scheme. In this scheme, the neighboring lattice nodes are explicitly known through the structure of the lattice, via array indexing.

In direct addressing, both V and P are accessed through enumeration numbers provided by the enumeration function. This implies that in vector V, memory must be allocated also for the distribution values of the solid nodes.

2.2 New Algorithm

In order to reduce memory consumption, a new algorithm was developed, which approximately halves memory requirements. Although some other algorithms have been presented before, keeping the simplicity of LBM implementation has also been considered in this new algorithm.

In regard to our new algorithm, simplicity, memory efficiency, and bandwidth reduction of the standard LBM algorithm have been easily modifiable.

Generally, in implementation of this algorithm, the original two steps have been retained, while each streaming and collision steps have been done for each fluid packages, individually. The flowcharts of the standard two-step and the new algorithm are shown in Fig. 1.

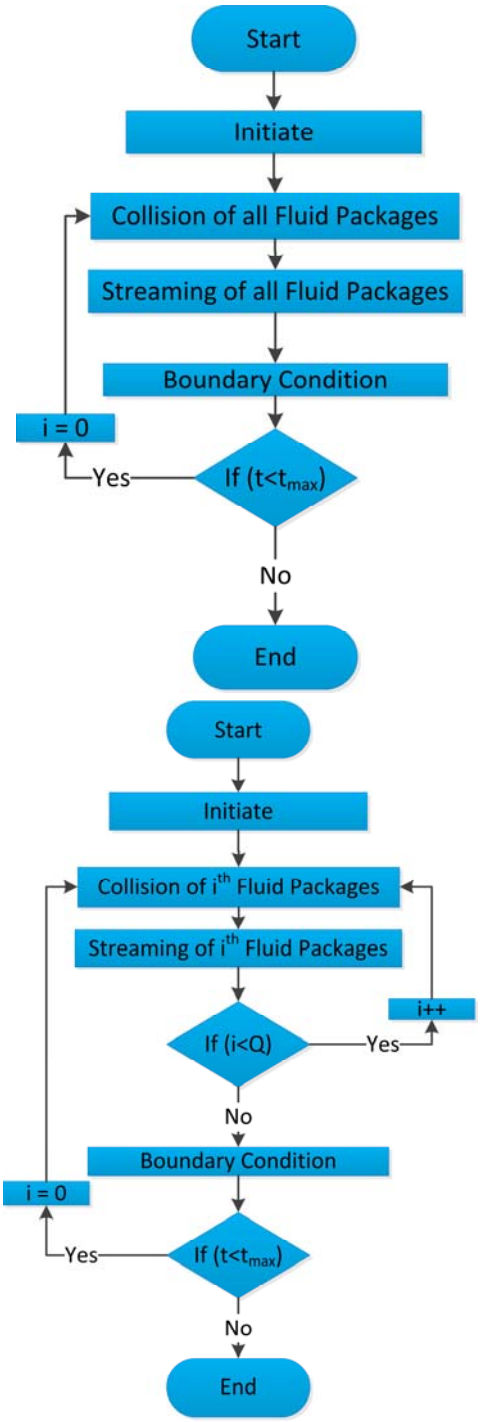


Fig. 1. Flowcharts of the standard (up) and the new algorithms (down).

In contrast to the standard algorithm, in our new method, collision step of the i^{th} fluid package (q) and then its streaming step were performed. These steps loop over all fluid packages. In addition, boundary conditions were implemented after finishing collision-streaming of all fluid packages. As seen, there is not any special consideration in this step. By ending this step, current time step becomes complete and the next time step begins.

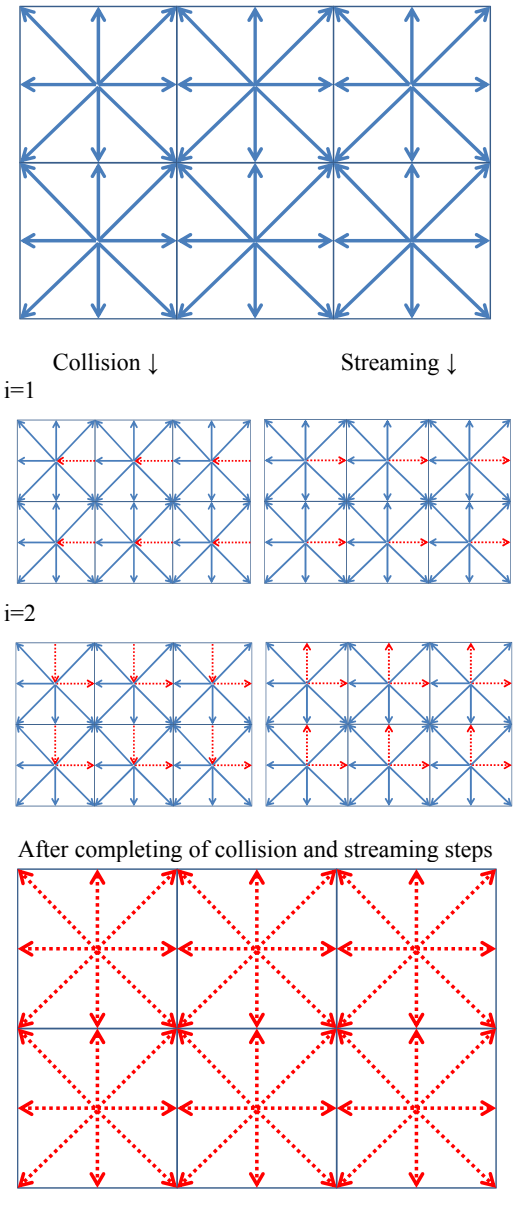


Fig. 2. Collision-streaming consequence of our new algorithm: initial or previous time step (top); collision-streaming of each cites (middle); the end of current time step (bottom).

The collision-streaming consequence is shown in Fig. 2. By performing the collision of each fluid package, it would be streamed in its direction. It could be performed because of the independency of fluid packages in their own directions. Therefore, coupling of collision-streaming for each fluid package would be allowed without any interruption in operation of the original LBM, which would indeed lead to the same results at each time step.

Briefly, the steps of our new algorithm in each time

step are as follow:

1. collision of the ith fluid package,
2. streaming of the ith fluid package,
3. performing steps 1 and 2 for all cites of fluid packages, and
4. implementing of the boundary conditions.

Thus, the collision-streaming in this method are completely independent, so that decoupling of fluid packages from each other is allowed in each time step.

To find the new macroscopic variable for the next time step (during streaming of the fluid package), they are calculated and stored for the next time step, as:

$$\rho = \sum_{i=0}^{q} f_i \tag{1}$$

$$\overrightarrow{V} = \sum_{i=0}^{q} \vec{e}_i . f_i \tag{2}$$

Depending on the dimension and the total number of lattice cites (q), $\vec{e}_i$ is defined for D2Q9 as:

$e_x = \{0, 1, 0, -1, 0, 1, -1, -1, 1\}$,
$e_y = \{0, 0, 1, 0, -1, 1, 1, -1, -1\}$ and, for D3Q19 as:

$$e_\alpha = \begin{cases} (0,0,0) & \alpha = 0 \\ (\pm1,0,0)\mathrm{c}, (0,\pm1,0)\mathrm{c}, (0,0,\pm1)\mathrm{c} & \alpha = 1 \sim 6 \\ (\pm1,\pm1,0)\mathrm{c}, (0,\pm1,\pm1)\mathrm{c}, (\pm1,0,\pm1)\mathrm{c} & \alpha = 7 \sim 18 \end{cases}$$

$\sum$ operator could be discretized to multi-steps, which is performed after streaming and updating of each *f*. Therefore, by discretizing the macroscopic variables, the independency of calculation and updating of fluid packages are individually retained. Before beginning of collision and streaming steps Thus, during collision and streaming of each fluid package, the new streamed fluid package is summed (Eqs. 1 and 2). When all fluid packages have been finished, the fluid properties for the next time step are ready as well.

Also, as the boundary conditions are implemented after collision and streaming steps, the changes of macroscopic values at the boundary nodes are inevitable. So, as a straight treatment, the macroscopic values at some boundary conditions (e.g., inlet) should be recalculated and updated.

In addition, in the new algorithm just a spare variable is used to store f_i^{temp} to replace in f_i^{new}, which in D2Q9 lead to memory reduction of 45% and in D3Q19 lead to memory reduction about 47%, instead of exactly half.

It should be noted that except for the flow properties, if there are several distribution functions, the same temporary variables could be used. This leads to more memory reduction. Thus, our approach could be used for distribution functions of other properties, such as temperature, spices concentration, as well.

3. RESULT AND DISCUSSION

3.1 Benchmark Simulations

To check if our new algorithm is working correctly, its related results were compared to those of the standard two-step algorithm. In this regard, several benchmark problems, i.e., 2-D and 3-D flows with different Re numbers, were considered. In the 2-D cases, a standard lid driven cavity flow and flow around a cylinder were considered, while in the 3-D cases, a cubic cavity and flow around a sphere were considered. To ensure that both of these algorithms have the same result, they were implemented in a code with two zones, where the new algorithm was dedicated in a zone and the standard two-step used in another one. Also, to couple the data of the two algorithms at intersection of the zones, an interface boundary condition was developed, which performs similar to a streaming step.

Note, to compare the computational performance of both algorithms, use of MLUPS (lattice updated per second in million) is a suitable choice. Higher MLUPS, for the same domain size, means better computational performance. Clearly, it shows less computational time of the solution. MLUPS is monitored for several grid resolutions to do performance analysis. The hardware used here was a E8400 core 2 Duo by 3 GHz CPU, which only one of them was used.

2-D Lid-Driven Cavity flow

As a basic fluid dynamic issue, a lid driven cavity flow was solved with several grid sizes. Fig. 3 shows the related physical domain and its boundary conditions. The unit lid driven velocity was converted to LBM velocity, using lattice length (L) and relaxation factor (Ω) to retrieve the same Re in both physical and LBM domains.

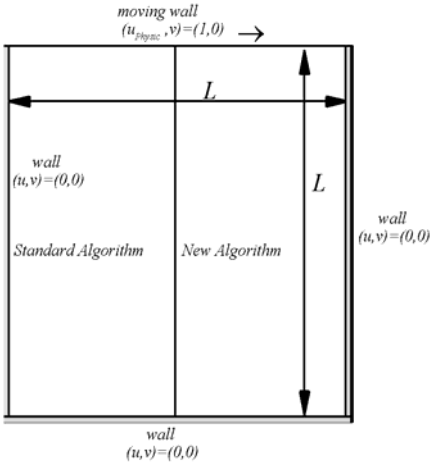


Fig. 3. Physical domain and boundary conditions of a 2-D lid driven cavity flow.

To verify the accuracy of the new algorithm, our results were compared to some benchmark studies in lid driven cavity flow. Fig. 4 shows the comparative plot of our new algorithm at Re 100, which confirms the accuracy of the new method very well.

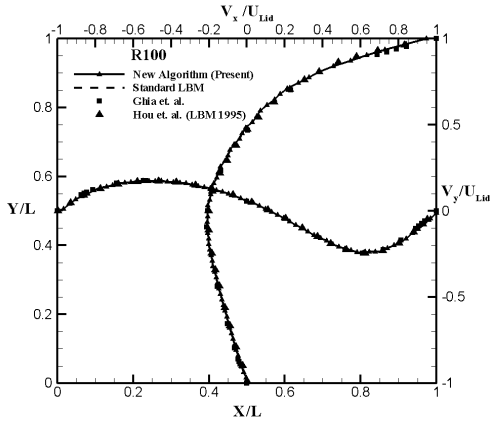
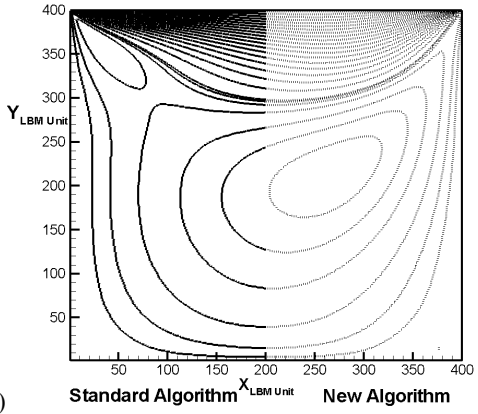
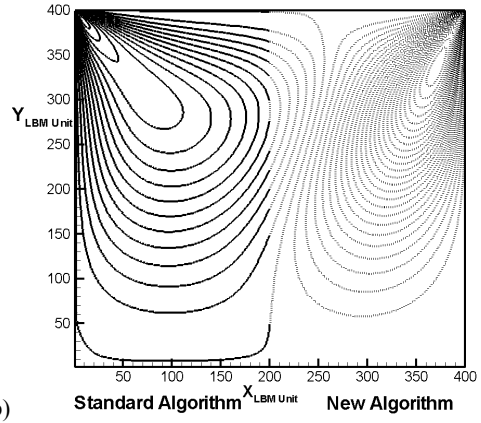


Fig. 4. X and Y-components of the velocity in the mid-line of a lid driven cavity at Re=100.



(a)



(b)

Fig. 5. Contours of X (a) and Y-components (b) of the velocity for a lid driven cavity at Re=100 with a 400x400 grid, using the standard and the new LBM algorithms.

Figure 5 shows the contours of velocity components in a 400x400 gird size for both algorithms. Note, there is not any discontinuity at the interfaces. Because of no interpolation/extrapolation at intersection boundaries, both algorithms have the same results (see also Fig. 5). This could also be observed in streamline contours of Fig. 6.

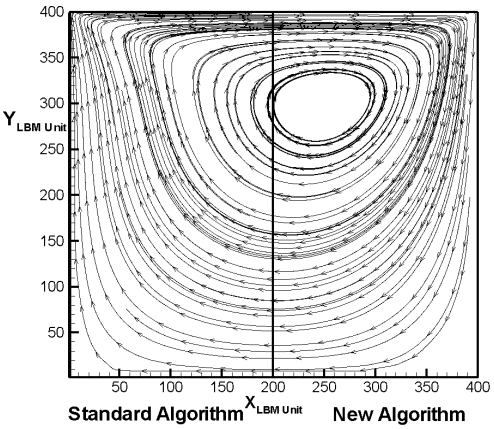


Fig. 6. Streamlines of a lid driven cavity at Re=100, using the standard and the new algorithms.

To do a Re study, numerical simulation was carried out using both the standard LBM and the new algorithms for Re of 400, 1000, 5000, 7500, and 10000. Steady state solution was obtained, except for the last case, since bifurcation takes place somewhere between Re 7500 and 10,000 (Hou *et al.* 1994). As Hou *et al.* (1994) presented, the results for Re 10,000 oscillate between a series of different configurations when using the standard LBM. It is one of the weaknesses of the standard LBM to capture turbulence phenomena by increasing turbulence intensity at higher Re's.

In addition, to simulate an unsteady incompressible flow, some additional conditions and methods, such as non-equilibrium extrapolation, multi-relaxation time scheme, combining LBM with a subgrid model, etc. are used to eliminate artificial compressible effects (Xiao-Yang *et al.* 2004, Zhen-Hua *et al.* 2006).

The comparative streamlines of the standard LBM and our new algorithm at Re 400, 1000, 5000 and 7500 are shown in Fig. 7. In our new method, the basic discretization of lattice Boltzmann equation (LBE) has not been changed and no changes in the obtained results were expected (Figs. 7-8). In our new approach, collision and propagation steps are performed serially and sequentially, which does not influence the final values of the distribution functions.

The accuracy of the new algorithm is the same as that of the standard LBM and the result adapted very well to other benchmark studies at different Re's (Fig. 8). By refining the grid sizes, the standard LBM can resolve low Re turbulent flow field very well. While at high Re, weakness of the standard LBM (and the present algorithm indeed) is more

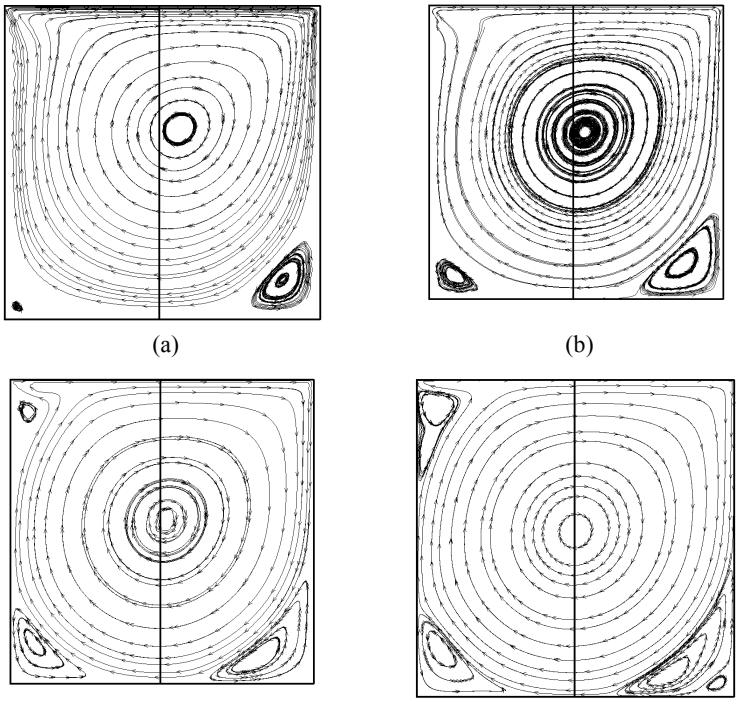


Fig. 7. Streamlines for a lid driven cavity at (a) Re=400, (b) Re=1000, (c) Re=5000 and (d) Re=7500 using the standard and the new algorithms.

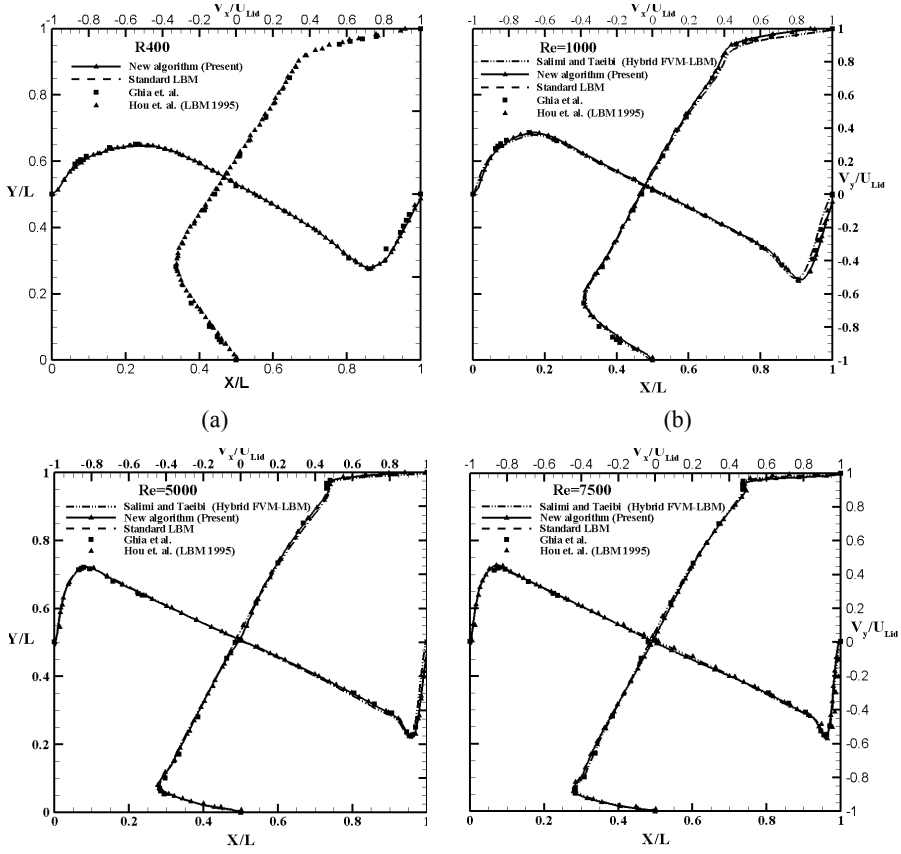


Fig. 8. Comparison of the velocity components of the new algorithm to others at mid-line of the cavity at (a) Re=400, (b) Re=1000, (c) Re=5000 and (d) Re=7500.

visible. This natural weakness refers to compressible effect of LBM, which leads to an undesirable error in the numerical simulation. Some efforts have been proposed to reduce or eliminate such errors.

Since in the new algorithm collision and streaming steps for each cite are serialized, to model the turbulent flow with our new algorithm, may need further work.

Table 1 shows the performance analysis of several grid sizes in MLUPS, mentioned earlier. The best obtained performance for 3 grid sizes was about 76% of the standard method in a 1000x1000 grid.

Table 1 Computational performance comparison of the standard and the new algorithms in MLUPS with different grid sizes for 2D Lid-Driven Cavity

Methods / Grid Size	Standard Algorithm	New Algorithm
400x400	2.63	1.92
500x500	2.66	1.97
1000x1000	2.73	2.08

Flow Around a Cylinder

As another 2D basic flow problem, flow around a cylinder was simulated. A cylinder with diameter "D" was placed at the middle of height "H" (=5D) and a diameter distance from the inlet in a domain with length "L" (=20D). Velocity inlet boundary condition was used in the front side of the domain (left side) and outlet boundary condition was used at the end of the domain (right side). In the top and bottom of the domain, periodic boundary condition was used. While, on the surface of the cylinder, wall boundary condition was implemented. The physical domain and the boundary conditions are shown in Fig. 9.

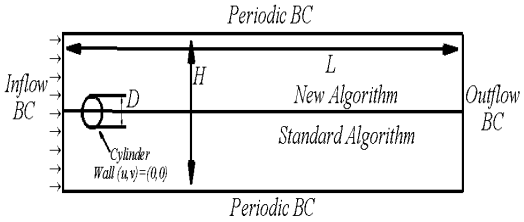


Fig. 9. Physical domain of flow around a cylinder.

The results show the effectiveness of the presented method. Velocity contours (Fig. 10) and streamlines (Fig. 11) do not show any discontinuity, which confirms correctness of the new algorithm.

Figures 12 and 13 show the X and Y components of the velocity at different locations, verifying the equality of the results for both new and standard algorithms.

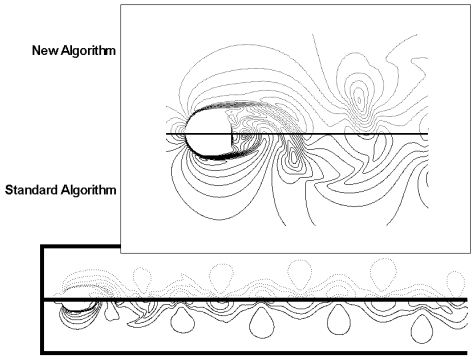


Fig. 10. Simulation of Velocity contours around a cylinder at Re=500, using the standard (bottom) and the new algorithms (top).

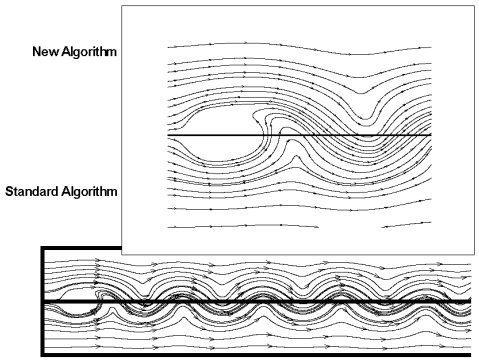


Fig. 11. Streamlines around a cylinder at Re=500, using the standard (bottom) and the new algorithms (top).

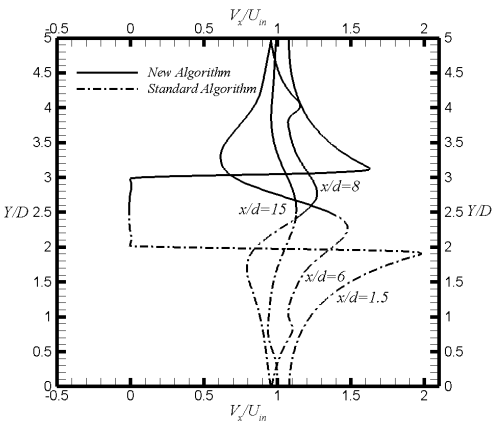


Fig. 12. X –component of the velocity in different sections of the flow around a cylinder at Re=500, with a 1280x320 grid, using the standard and the new algorithms.

Table 2 shows the performance analysis for several grid sizes. The performance for 4 grid sizes approximately was the same in most of them and was about 0.65 of standard method.

3-D Cavity Flow

The physical domain contains two rectangular cubes, in which the fluid flow is solved, using the standard method in one zone and our new method in another one (Fig. 14). Flow is solved for Re 1000

in a 64x64x64 grid. The velocity contours are shown in Fig. 15. The X-component of velocity at the middle of the domain in X-direction and different levels in Y-direction are shown in Fig. 16. Both Figs. 15 and 16 prove the correctness of our new algorithm.

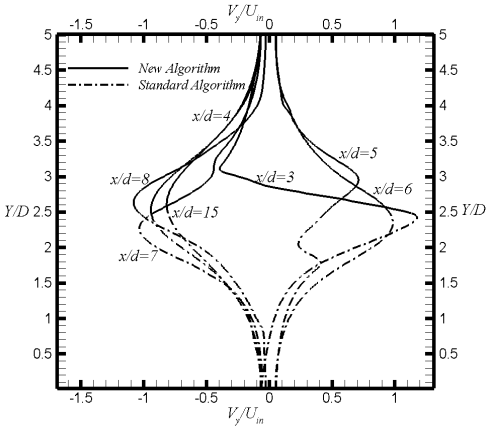


Fig. 13. Y-component of the velocity in different sections of the flow around a cylinder at Re=500, with a 1280x320 grid, using the standard and the new algorithms.

Table 2 Computational performance comparison of the standard and the new algorithms in MLUPS with different grid sizes for flow around a cylinder

Methods / Grid Size	Standard Algorithm	New Algorithm
640x160	3.106	2.06
960x240	3.178	2.073
1280x320	3.21	2.079
1600x640	3.22	2.080

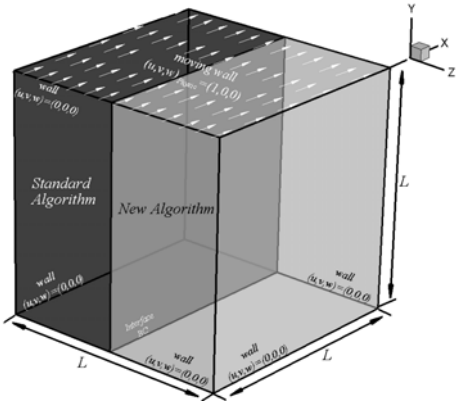


Fig. 14. Physical domain for the 3-D cavity flow.

To prove the authenticity and the accuracy of the new algorithm in 3D, the obtained results have been compared to other methods such as pseudo spectral method of Ku et. al. (Ku *et al.* 1987), FVM of Babu et. al. (Babu and Korpela 1994), Hybrid FVM-LBM of Salimi and Taeibi-Rahni (Salimi and Taeibi-Rahni 2015). Figure 17 shows the comparative velocity plots in the mid-line section at X and Y directions, which verify the accuracy and correctness of the new algorithm.

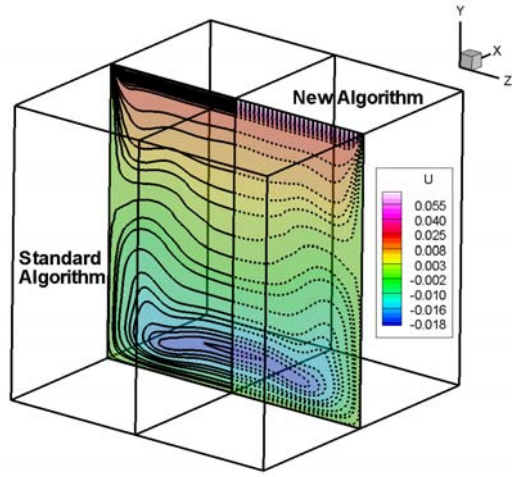


Fig. 15. Contours of the X-component of velocity for a 3D cavity flow at Re=1000, using a 64X64X64 grid.

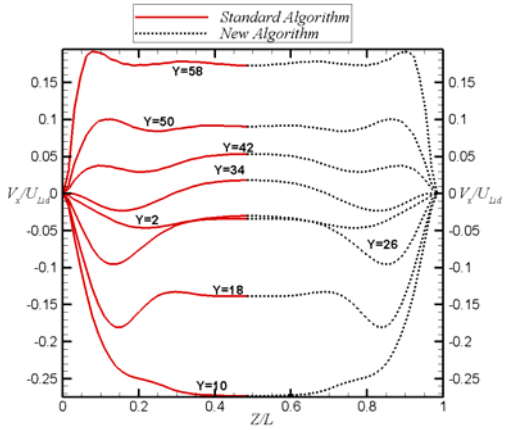


Fig. 16. The X-component of velocity for a 3D cavity flow at Re=1000, using a 64X64X64grid in the mid-plane of X (X/L=0.5) at different levels in Y direction.

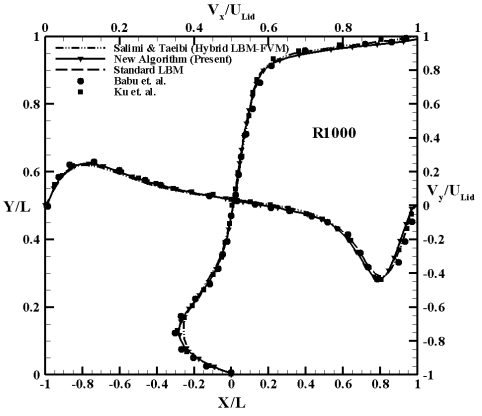


Fig. 17. Comparison of the velocity components of our new algorithm to others at mid-line of the cavity at Re=1000.

Table 3 shows the performance analysis for two algorithms. This problem has been solved for two different grid sizes to compare computational performance in MLUPS. Approximately, the performance was about 0.73 of standard method.

Table 3. Computational performance comparison of the standard and the new algorithms in MLUPS on different grid sizes for 3-D cavity flow.

Methods / Grid Size	Standard Algorithm	New Algorithm
64x64x64	1.32	0.938
128x128x128	1.38	1.012

Flow Around a Sphere

In flow around a sphere, computational domain was divided into two halves: one half for standard method and another half for the new method (Fig. 18). A sphere with diameter "D" was placed at the center of height "H" (=2D) and width "W" (=2D) with a half diameter distance from the inlet in a domain with length "L" (=4D). Velocity Inlet boundary condition was used in the front side of the domain and outlet boundary condition was used at the end of the domain. At the side panels of the domain, periodic boundary condition was used. While, on the surface of the sphere, wall boundary condition was used. The detailed of the physical domain and the boundary conditions are shown in Fig. 18.

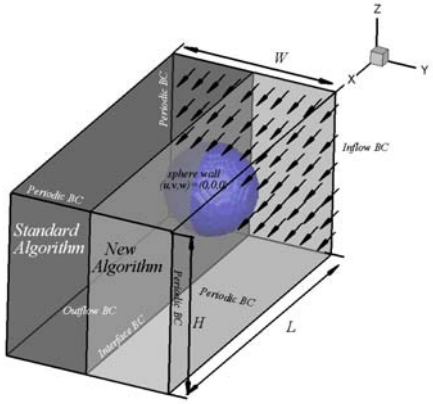


Fig. 18. The physical domain of flow around a sphere.

Figure 19 shows the flow streamline around a sphere at Re 20 and 200, obtained from two methods. As expected, there was not any discontinuity at interface of the computational domain. This continuity has been seen of velocity contours of Fig. 20.

No differences between 2-D and 3-D in performance have been observed. This is due to high computational cost of streaming and collision part of LBM method (with respect to other parts, e.g., boundary conditions).

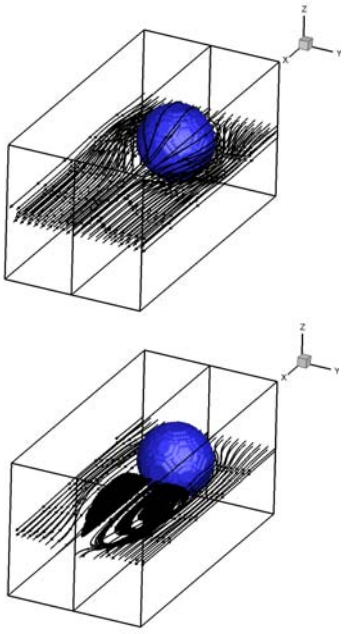


Fig. 19. Streamline of flow around a sphere at Re=20 (up), Re=200 (down).

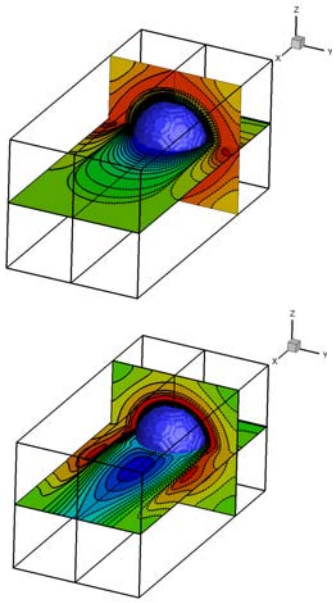


Fig. 20. Contours of X-component of velocity at Re=20 (up), Re=200 (down).

3.2 Algorithm Optimization

At the first step to optimize the presented algorithm, instead of a temporary array variable, more temporary variable could be used. Therefore, more fluid packages could be transferred in each collision-streaming step and the rate of call back memory reduces, as the number of temporary arrays increase. Figure 21 shows the optimized flowchart of the new algorithm with "n" temporary arrays.

This optimization approach is a trade-off between the standard and the new algorithms. By increasing the temporary arrays, method would be similar to standard algorithm and by reducing the number of temporary arrays to one, computational

Table 4 Computational performance comparison analysis.

	MLUPS	Performance	Memory Reduction
Standard Two-Step	3.276 1.56		NA (D2Q9) NA (D3Q19)
New Algorithm*	2.28 1.08	70% 69%	~45%(D2Q9) ~48%(D3Q19)
Optimized #1**	2.75 1.30	84% 83%	~39%(D2Q9) ~45%(D3Q19)
Optimized #2***	3.1 1.47	95% 94%	~33%(D2Q9) ~42%(D3Q19)
*	"n" = 1; New Algorithm with one temporary array.		
**	"n" = 2; New Algorithm with two temporary arrays.		
***	"n" = 3; New Algorithm with three temporary arrays.		

performance reduces, memory conservation is improved, and memory usage approximately reduces to a half. Thus, as a conclusion, these two algorithms (standard and optimized) would be a compromise between computational performance and memory reduction.

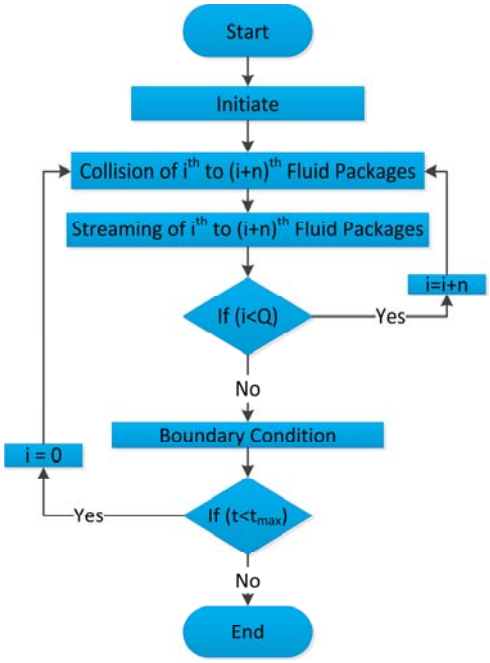


Fig. 21. Optimized flowchart of the new algorithm, using 'n' temporary arrays.

By increasing the number of temporary arrays to two or three fluid packages, the computational performance improvement is quite acceptable. This optimization approach leads to the improvement of performance. Table 4 shows the performance of the algorithm in one, two, and three fluid packages in each collision-streaming step, compared to the standard two-step algorithm. In three temporary arrays, computational performance is improved to about 95% of standard algorithm (94% in D3Q19) and memory reduction is decreased from 45 to 33% (48 to 42% in D3Q19).

3.3 Implementation of GPU

As mentioned before, at each lattice point, the fluid packets collide with each other and are restricted locally, depending on data from neighboring nodes. The spatial locality of LBM to data makes it a good candidate for parallelization. For many reasons, such as cost, memory bandwidth, energy consumption, etc., the best platform for parallel processing of LBM is GPU.

As a regular treatment to reduce ideal processing part, which arises from data transfer form PC main memory to GPU's onboard memory and vice versa, all of the computational domain is loaded on GPU memory. As GPU's onboard memory is limited, the new algorithm would be more useful to simulate a larger physical domain. This limitation is more highlighted, when considering heat and mass transfers.

Here, two algorithms have been implemented on GPU by CUDA compiler. The obtained GPU speed-up levels (with respect to CPU in MLUPS) are shown in Table 5. In all cases of optimization, speed-up level on 3-D domain is more than 2-D, which is because of higher memory bandwidth on GPU. Also, higher memory bandwidth causes to recover computational performance just by two temporary arrays, in compare with run on ordinary processors (instead of three temporary arrays). So, the new algorithm on GPU recovers computational performance inefficiency besides reduction in memory consumption.

4. CONCLUSION

As presented in this paper, the new algorithm reduces memory consumption to nearly a half and some computational performance reductions. To improve the computational performance, an approach was presented which leads to recover this deficiency by one or two extra temporary array data. Comparison of two different simulations at

Table 5 Computational performance comparison of two algorithms analysis

	CPU (MLUPS)	Tesla C2050 (MLUPS)	Speed-Up
Standard Two-Step	3.276 1.56	140 73	~ 42 (D2Q9) ~ 46 (D3Q19)
New Algorithm*	2.28 1.08	109 55	~ 47 (D2Q9) ~ 53 (D3Q19)
Optimized #1**	2.75 1.30	132 72	~ 48 (D2Q9) ~ 55 (D3Q19)
Optimized #2***	3.1 1.47	154 81	~ 50 (D2Q9) ~ 56 (D3Q19)
*	"n" = 1; New Algorithm with one temporary array.		
**	"n" = 2; New Algorithm with two temporary arrays.		
***	"n" = 3; New Algorithm with three temporary arrays.		

various Re numbers (in both two- and three-dimensions) were shown that the algorithm give the same result by less memory and approximately the same computational performance with respect to standard two steps algorithm. Implementation of this approach is very simple and quiet efficient in memory performance. In spite of retaining the simplicity in the new algorithm, GPU was simply implemented. Note, higher bandwidth in GPU is very helpful to gain more memory reduction via less computational performance loss.

The optimization approach has been a compromise between the new and the standard algorithms, whereby it increments the number of temporary array, or memory bandwidth, it would be very similar to standard algorithm and by decrement the number of temporary arrays to one it would be just like new algorithm. So, it is a tradeoff between memory consumption and computational performance. Also, Note, that by performing collision and streaming steps of each cite individually in the new algorithm, modeling some developed features of the standard LBM (such as turbulent modeling) may be very difficult and would need further investigation.

5. FUTURE WORK

Besides working on optimization to increase the computational performance and reduction in memory consumption, some supplementary tasks, such as implementing more complicated boundary conditions are on schedule.

REFERENCES

Abe, T. (1997). Derivation of the lattice Boltzmann method by means of the discrete ordinate method for the Boltzmann equation. *Journal of Computational Physics* 131(1), 241-246.

Alvarado, V., H. T. Davis and L. Scriven (1997). Effects of pore-level reaction on dispersion in porous media. *Chemical engineering science* 52(17), 2865-2881.

Babu, V. and S. A. Korpela (1994). Numerical solution of the incompressible three-dimensional Navier-Stokes equations. *Computers and fluids* 23(5), 675-691.

Bailey, P., J. Myre, S. D. Walsh, D. J. Lilja and M. O. Saar (2009). Accelerating lattice Boltzmann fluid flow simulations using graphics processors. *Parallel Processing, 2009. ICPP'09. International Conference on, IEEE.*

Chen, H., S. Chen and W. H. Matthaeus (1992). Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method. *Physical Review A* 45(8), R 5339.

Chen, S. and G. D. Doolen (1998). Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics* 30(1), 329-364.

Coppens, M. O. and G. F. Froment (1995). Diffusion and reaction in a fractal catalyst pore—II. Diffusion and first-order reaction. *Chemical engineering science* 50(6), 1027-1039.

Dawson, S. P., S. Chen and G. Doolen (1993). Lattice Boltzmann computations for reaction diffusion equations. *The Journal of Chemical Physics* 98(2), 1514-1523.

Di R., A. F., P. Asinari, E. Chiavazzo, N. Prasianakis and J. Mantzaras (2012). Lattice Boltzmann model for reactive flow simulations. *EPL (Europhysics Letters)* 98(3), 34001.

Filippova, O. and D. Hänel (1998). Grid refinement for lattice-BGK models. *Journal of Computational Physics* 147(1), 219-228.

Frisch, U., B. Hasslacher and Y. Pomeau (1986).

Lattice-gas automata for the Navier-Stokes equation. *Phys. Rev. Lett* 56(14), 1505.

Geller, S., M. Krafczyk, J. Tölke, S. Turek and J. Hron (2006). Benchmark computations based on lattice-Boltzmann, finite element and finite volume methods for laminar flows. *Computers and Fluids* 35(8), 888-897.

He, X. and L. S. Luo (1997). Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation. *Physical Review E* 56(6), 6811.

Higuera, F. and J. Jimenez (1989). Boltzmann approach to lattice gas simulations. *EPL (Europhysics Letters)* 9(7), 663.

Higuera, F., S. Succi and R. Benzi (1989). Lattice gas dynamics with enhanced collisions. *EPL (Europhysics Letters)* 9(4), 345.

Hou, S., Q. Zou, S. Chen, G. D. Doolen, andA. C. Cogley (1994). Simulation of cavity flow by the lattice Boltzmann method. *arXiv preprint comp-gas/9401003*.

Ku, H. C., R. S. Hirsh, andT. D. Taylor (1987). A pseudospectral method for solution of the three-dimensional incompressible Navier-Stokes equations. *Journal of Computational Physics*, 70(2): 439-462.

Massaioli, F. and G. Amati (2002). *A*chieving high performance in a LBM code using OpenMP. *The Fourth European Workshop on OpenMP*, Mattila, K., J. Hyväluoma, T. Rossi, M. Aspnäs and J. Westerholm (2007). An efficient swap algorithm for the lattice Boltzmann method. *Computer Physics Communications* 176(3), 200-210.

Mattila, K., J. Hyväluoma, J. Timonen and T. Rossi (2008). Comparison of implementations of the lattice-Boltzmann method. *Computers and Mathematics with Applications* 55(7), 1514-1524.

McNamara, G. R. and G. Zanetti (1988). Use of the Boltzmann equation to simulate lattice-gas automata. *Physical Review Letters* 61(20), 2332.

Pan, C., J. F. Prins and C. T. Miller (2004). A high-performance lattice Boltzmann implementation to model flow in porous media. *Computer Physics Communications* 158(2), 89-105.

Philippi, P. C., L. A. Hegele Jr, L. O. Dos Santos and R. Surmas (2006). From the continuous to the lattice Boltzmann equation: The discretization problem and thermal models. *Physical Review E* 73(5), 056702.

Pohl, T., M. Kowarschik, J. Wilke, K. Iglberger and U. Rüde (2003). Optimization and profiling of the cache performance of parallel lattice Boltzmann codes. *Parallel Processing Letters* 13(04), 549-560.

Qian, Y., D. d'Humières and P. Lallemand (1992). Lattice BGK models for Navier-Stokes equation. *EPL (Europhysics Letters)* 17(6), 479.

Rahmati, A., M. Ashrafizaadeh and E. Shirani (2014). A Multi-Relaxation-Time Lattice Boltzmann Method on Non-Uniform Grids for Large Eddy Simulation of Rayleigh-Bénard Convection Using Two Sub-Grid Scale Models. *Journal of Applied Fluid Mechanics* 7(1), 89-102.

Salimi, M. and M. Taeibi-Rahni (2015). New lifting relations for estimating LBM distribution functions from corresponding macroscopic quantities, based on equilibrium and non-equilibrium moments. *Journal of Computational Physics* 302, 155-175.

Shan, X. and H. Chen (1993). Lattice Boltzmann model for simulating flows with multiple phases and components. *Physical Review E* 47(3), 1815.

Shan, X., X. F. Yuan and H. Chen (2006). Kinetic theory representation of hydrodynamics: a way beyond the Navier–Stokes equation. *Journal of Fluid Mechanics* 550, 413-441.

Succi, S. (2001). *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Clarendon, Oxford.

Velivelli, A. and K. M. Bryden (2004). A cache efficient implementation of the lattice Boltzmann method for the two-dimensional diffusion equation. *Concurrency and Computation: Practice and Experience* 16(14), 1415-1432.

Wellein, G., T. Zeiser, G. Hager, andS. Donath (2006). On the single processor performance of simple lattice Boltzmann kernels. *Computers and Fluids* 35(8), 910-919.

Wittmann, M., T. Zeiser, G. Hager and G. Wellein (2013). Comparison of different propagation steps for lattice Boltzmann methods. *Computers and Mathematics with Applications* 65(6), 924-935.

Xiao-Yang, Z., S. Bao-Chang, and W. Neng-Chao (2004). Numerical simulation of LBGK model for high Reynolds number flow. *Chinese Physics* 13(5), 712.

Zeiser, T., G. Hager and G. Wellein (2009). Benchmark analysis and application results for lattice Boltzmann simulations on NEC SX vector and Intel Nehalem systems. *Parallel Processing Letters* 19(04), 491-511.

Zhang, L. and N. Seaton (1994). The application of continuum equations to diffusion and reaction in pore networks. *Chemical Engineering Science* 49(1), 41-50.

Zhen-Hua, C., S. Bao-Chang and Z. Lin (2006). Simulating high Reynolds number flow in two-dimensional lid-driven cavity by multi-relaxation-time lattice Boltzmann method. *Chinese Physics* 15(8), 1855.