ParCFD'2019
31st International Conference on Parallel Computational Fluid Dynamics
May 14-17 2019, Antalya TURKEY

APPLICATION OF D2Q5 PRESSURE BOUNDARY CONDITIONS FOR PARALLEL D2Q9 LATTICE BOLTZMANN METHOD

NEVSAN SENGIL[*] AND FADILE Y. COMEZ[†]

[*] University of Turkish Aeronautical Association
Department of Astronautical Engineering
06790 Ankara, TURKEY
e-mail: nsengil@thk.edu.tr - Web page: https://aero.thk.edu.tr

[†] Middle East Technical University (METU)
Department of Aerospace Engineering
06800 Ankara, TURKEY
e-mail: yudum.comez@metu.edu.tr - Web page: http://ae.metu.edu.tr

Key words: Pressure boundary conditions, Lattice Boltzmann method, PGAS paradigm, coarray Fortran.

Summary. Channel flows with pressure boundary conditions are very common both in industrial applications and scientific researches. In channel flow, a viscous fluid is confined by solid walls. Because of a pressure difference between at the inlet and outlet sections of the channel, fluid keeps moving in from high pressure region to low pressure region. Because of the shear stress, momentum and energy are transferred between fluid and walls. After a while flow becomes steady. In this study, we simulate a 2D channel flow using parallel Lattice Boltzmann method (LBM). LBM method solves the complicated Boltzmann equations on a discretized mesh. This method has some distinct advantages compared to the conventional macro scale methods. LBM is preferred to deal with complicated geometry and multiphase flows. The parallel LBM solver used in this study is developed indigenously using Coarray technique embedded in Fortran programming language. The computational resource is a workstation which consists of 2 Xeon processors with quad cores. Total memory capacity is 4 GB. In this study main structure of the LBM is built on D2Q9 scheme. However on the inlet and outlet D2Q5 scheme is implemented to constitute pressure boundary conditions. Results are compared with Zou/He pressure boundary conditions and analytic solutions.

1 INTRODUCTION

There are many studies in the literature to parallelize the lattice Boltzmann method in order to reduce the solution time. Jonas Tölke implemented the two-dimensional D2Q9 LBM solver on the single-GPU (graphic processing unit) [1]. Christian Obrecht analyzed the D3Q19 lid-driven cavity problem on the six NVIDIA Tesla C1060 GPUs working in parallel [2]. The advent of the newest NVIDIA Fermi architecture GPUs provides direct access and transfers between the GPUs. J. Tölke and C. Obrecht applied message passing interface (MPI) parallelization technique and MPI libraries.

In addition to the conventional techniques such as MPI and OpenMP, the partition global address space (PGAS) paradigm [3] was also implemented to parallelize the lattice Boltzmann method. This new approach is relatively simple for developing and debugging the codes. Some of the programming languages incorporate this new technique are unified parallel C (UPC), Co-array Fortran (CAF) and Java based Titanium. In this study, we used Co-array Fortran to parallelize our sequential lattice Boltzmann solver.

2 PARALLEL LATTICE BOLTZMANN METHOD

2.1 Lattice Boltzmann Method

The lattice Boltzmann method gives velocity distribution function (f_i) by.

$$f_i(x + c_i, t + 1) - f_i(x, t) = \Omega(f_i) \tag{1}$$

Macroscopic fluid density and velocity can be obtained by using distribution function.

$$\rho(x, t) = \sum_i f_i(x, t) \tag{2}$$

$$u(x, t) = \frac{1}{\rho}\sum_i f_i(x, t)\, c_i \tag{3}$$

Furthermore, the equation (1) consists of two parts. These are propagation and collision steps from left to right respectively. The collision part, which is modeled from BGK approach, is given by;

$$\Omega(f_i) = \frac{1}{\tau}\left(f_i^{eq}(x, t) - f_i(x, t)\right) \tag{4}$$

Another part of the lattice Boltzmann equation is propagation (streaming) step which is denoted by $f_i(x + c_i, t + 1)$. In this part, $f_i^{eq}(x, t)$ is the equilibrium distribution function being expressed as the multi-energy Maxwellian model by,

$$f_i^{eq}(\rho, u) = \rho W_i[1 + \frac{c_i.u}{c_s^2} + \frac{(c_i.u)^2}{2c_s^4} - \frac{u.u}{2c_s^2}] \tag{5}$$

Here c_s is the velocity of sound and W_i is the weight of lattice velocities. f_i^{eq} function is given by D2Q9 lattice model which consists of two dimensions and nine velocities ; $W_{i=0} = \frac{4}{9}$, $W_{i=1,2,3,4} = \frac{1}{9}$, $W_{i=5,6,7,8} = \frac{1}{36}$ are weight values for the model,

$$f_i^{eq} = \rho W_i[1 + 3c_i.u + \frac{9(c_i.u)^2}{2} - \frac{3(u.u)}{2}] \quad (6)$$

2.2 Co-array Fortran

The coarray programming bases on two primary issues how the code is parallelized ,work distribution and data distribution. The coarray Fortran adopts the Simple Program Multiple Data (SPMD) execution model.To control the communication mechanism, the partitioned global address space (PGAS) language is utilized. The PGAS model in CAF requires additional co-dimensions, so within the scope of the coarray syntax, there are two intrinsic functions that give information about the images in a multi-codimension grid. One of these is "num_images ()" pointing to the number of processors which may be the same number of physical processors, or it may be less, or it may be more, the second fundamental function is "this_image ()" that shows which processor is considered. The data distribution evolves that the local data are executed by each image and attainability of the local array is pointed by asterisks in square bracket [*]. Without square bracket the code can be run as independent. Furthermore,in order to sync a subsequent or preceding, read or write access of the same data from different processes, a synchronization of the processes has a great deal of importance on the coarrays. Consequently, we implement the coarray Fortran for parallelization, because it requires less line of codes. Additionally, understanding and maintaining the code is easier with the CAF.

2.3 Parallelization of the LBM solver

The LBM parallelization is based on domain decomposition (DD) technique. We divided the computational domain to subdomains. The number of the subdomains and processors are equal. These subdomains consist of columns of grids called strip as in Figure 1. Therefore, in the propagation (streaming) step, only the neighbor strips send and fetch data.

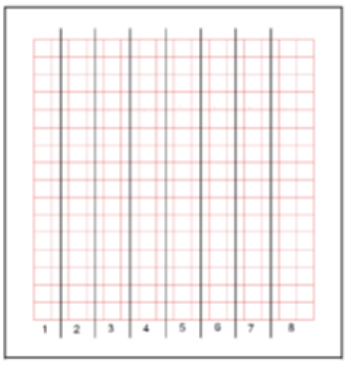


Figure 1: Grid partitioning for 8 cores

4 CHANNEL FLOW

4.1 Channel Geometry

In the current study plane Poiseuille flow is chosen as a test case for the application of new boundary condition. In a plane Poiseuille flow, a viscous fluid flows along two parallel plates with the help of a pressure difference as in Figure 2. The pressure (p) and density (ρ) in D2Q9 model are dependent variables and relation between them given as $p=\rho/3$ in D2Q9 model. According to this relation, fluid density changes in Poiseuille flow even if fluid is assumed incompressible. In the current study top and bottom boundaries are kept stationary and the west and east boundaries are kept opened.

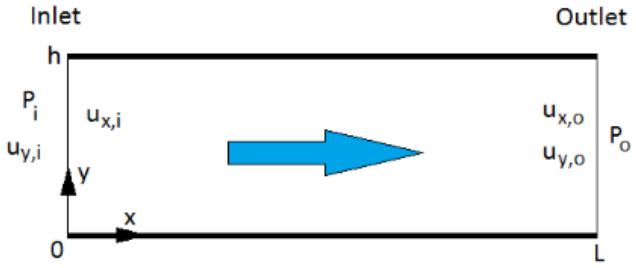


Figure 2: Plane Pouiseuille flow with pressure boundary conditions

4.2 Boundary Conditions

D2Q9 and D2Q5 LBM models are summarized in terms of densities and velocities in Figure 3.

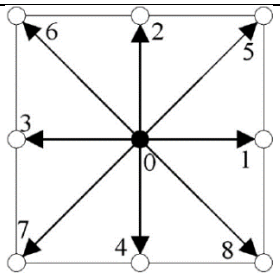
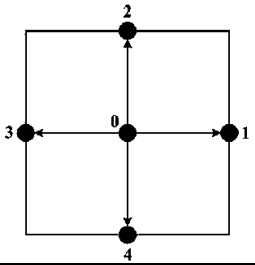
D2Q9 Model	D2Q5 Model
	
$\rho = f_0^9 + f_1^9 + f_2^9 + f_3^9 + f_4^9 + f_5^9 + f_6^9 + f_7^9 + f_8^9$ $\rho u_x = f_1^9 + f_5^9 + f_8^9 - f_3^9 - f_6^9 - f_7^9$ $\rho u_y = f_2^9 + f_5^9 + f_6^9 - f_4^9 - f_7^9 - f_8^9$	$\rho = f_0^5 + f_1^5 + f_2^5 + f_3^5 + f_4^5$ $\rho u_x = f_1^5 - f_3^5$ $\rho u_y = f_2^5 - f_4^5$

Figure 3: Comparision of the D2Q9 and D2Q5 LBM models

5 SIMULATION RESULTS

After 100000 iteration and for Kn=10 the LBM results from both Zou-He [4] and new approach is shown in Figure 4.

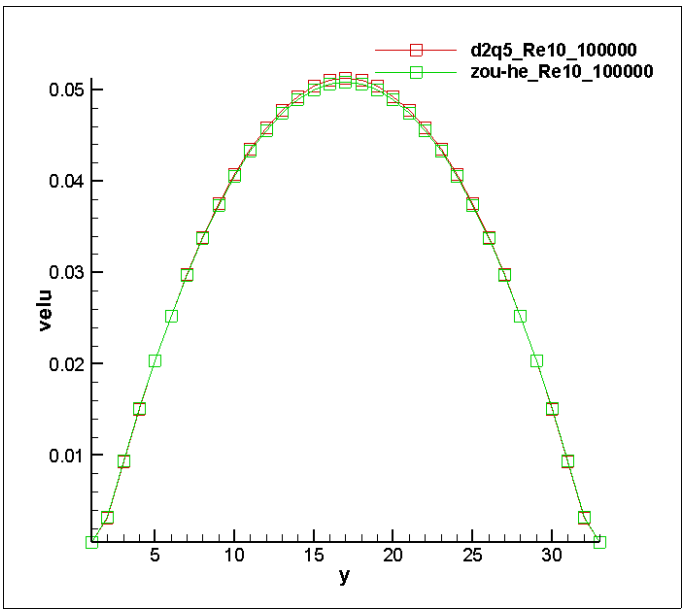


Figure 4: Comparision of the Zou-He and proposed boundary conditions

6 RESULTS

We observed that the velocity profile and pressure distribution through channel obtained with new boundary conditions are perfectly match with both Zou-He and analytical results.

REFERENCES

[1] J. Tölke, Implementation of a lattice Boltzmann kernel using the Compute Unified Device Architecture developed by NVIDIA, Computing and Visualization in Science. 2010;29-39.

[2] C. Obrecht, F. Kuznik, B. Tourancheau, and J.J Roux, Multi-GPU implementation of the lattice Boltzmann method, Computers and Mathematics with Applications, 2011; 10.1016.

[3] M.Metcalf, J.Reid, M.Cohen , Modern Fortran explained, Oxford University Press. 2013.

[4] Zou Q. and He X., On pressure and velocity boundary conditions for the lattice Boltzmann BGK model, Phys. of Fluids, 1997, 9, (6).