CONJUGATE HEAT TRANSFER PROBLEMS VIA A LATTICE BOLTZMANN SOLVER ON CPU AND GPU: A PERFORMANCE COMPARISON

Gregorio G. SPINELLI, AND Bayram CELIK

Istanbul Technical University
Department of Astronautical Engineering
34469 Istanbul, TURKEY
e-mail:spinelli15@itu.edu.tr; celikbay@itu.edu.tr

Key words: lattice Boltzmann method, conjugate heat transfer, GPU, CPU

Abstract. We developed a two-dimensional lattice Boltzmann method (LBM) to solve conjugate heat transfer (CHT) problems on GPU. The code solves the energy equations on solid and fluid regions simultaneously without any specific treatment along the solid-fluid interface. Each region is mapped with a uniform Cartesian mesh that consists of D2Q9 lattices. The computations we performed on CPU and GPU showed that the computations on GPU are 12 times faster than the one performed on the CPU. In order to reveal the main reason/s of this discrepancy, we tracked and then evaluated the computational times for the critical stages of the developed solver. Our analyses show that data transfer from device to host costs 95% of the whole computational time. On the other hand, computational time of the collision stage takes 50% of the computational time that is required for the developed lattice Boltzmann solver to run on CPU.

1 INTRODUCTION

The growing demand to solve high complex problems challenges manufacturers to continuously increase floating points operations (FLOPs) per unit time of modern Central Processing Units (CPUs) and Graphics Processing Units (GPUs). Recent GPU cards, such as NVidia TESLA V100, can reach about 7000 gigaFLOPs per unit time, while modern generation Intel Xeon Platinum 8180 are showing performances of about 2200 gigaFLOPs per unit time. Therefore, to execute a high number of parallel calculations on a massive quantity of data, a GPU card is preferred.

The lattice Boltzmann method, introduced by McNamara and Zanetti [1] in 1998, is an explicit method that allows any user to take advantage of massive parallel computation via GPUs [2]. Recently LBM was used to solve several types of flow problems on GPU, such as flows through porous media [3], conjugate heat transfer [2, 4, 5], turbulent flows [6], and etc.

In order to model heat exchange between solid and fluid in contact properly, one needs to solve energy equation on both solid and fluid region [2]. Spinelli et al. [2, 4] and Tarokh et al. [5] proved that LBM is capable of solving CHT problems, showing perfect

agreements with classic FVM results. Spinelli et al. studied the effect of solid to fluid conductivity ratio, Prandtl, and Reynolds numbers on a backward facing step flow over a thick wall.

The objective of this study is to analyze the computational time of the conjugate heat transfer lattice Boltzmann method implementation on GPU. We also compare the computational time of the critical stages of LBM between GPU and CPU versions.

2 LATTICE BOLTZMANN METHOD

The base of LBM is the gas kinetic theory, where the Boltzmann equation describes the evolution of the distribution function for each lattice. From the discrete Boltzmann equation, we obtain two new equations, which are the streaming and the collision stage, as shown in Reference [2]. The collision equation is as below

$$\tilde{f}_i(\vec{r},t) = f_i(\vec{r},t) + \omega_m \left[f_i^{\mathrm{eq}}(\vec{r},t) - f_i(\vec{r},t)\right] \tag{1}$$

In the above equation, $f^{\mathrm{eq}}(\vec{r},t)$ is the equilibrium distribution function, that is computed at the previous time stage. The streaming for each direction i is as follows

$$f_i(\vec{r}+\vec{c}_i\Delta t, t+\Delta t) = \tilde{f}_i(\vec{r},t) \tag{2}$$

In Eq. 1, we use an equilibrium distribution function that is second order accurate in space [7].

$$f_i^{\mathrm{eq}} = w_i\rho\left[1 + \frac{\vec{c}_i\cdot\vec{V}}{c_s^2} + 0.5\frac{(\vec{c}_i\cdot\vec{V})^2}{c_s^4} - 0.5\frac{\vec{V}\cdot\vec{V}}{c_s^2}\right] \tag{3}$$

For each lattice streaming direction, i the particle velocity, c and the weighting factor, w for D2Q9 model, as well as alternative models for 2D applications are described in Reference [7]. Alongside streaming and collision, LBM has two more stages, which are updating the boundary conditions and calculation of macroscopic quantities.

Evaluation of the energy equation follows the same procedure of Reference [2]. We set a new distribution function g as in Eq. 3, where T substitutes the density. Moreover, the collision frequency, ω is related to the thermal diffusivity, α instead of ν.

$$\alpha = \frac{\Delta x^2}{3\Delta t}\left(\omega_e - 0.5\right) \tag{4}$$

3 GPU IMPLEMENTATION

The collision stage, represented by Eq. 1, is coded on GPU as in List. 1. In order to code collision stage on CPU, one would use two `for` loops to swap the matrix that stores distribution function values on the CPU memory. Whereas on GPU, the `if` clause coupled with `blockIdx`, `blockDim`, `threadIdx` variables assures that the thread points to a matrix element within the matrix margins.

Listing 1: A sample code for collision stage on GPU

```
__global__ void Collision_GPU(float *f, float *rho, float *w,
   float *u, float *v, float *cx, float *cy, float omega, int Lx,
   int Ly, int Lz, float cs, float dt)
{
    int k = blockIdx.z * blockDim.z + threadIdx.z;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < Lx && j < Ly && k < Lz) {
        float feq = rho[j][i] * w[k] * (1. + 1./(cs*cs) * (u[j][i
   ] * cx[k] + v[j][i] * cy[k]));
        f[k][j][i] = f[k][j][i] + omega * (feq - f[k][j][i]);
    }
}
```

In this study, we use NVIDIA Quadro K620 as GPU card.

4 RESULTS

We solved the well-known benchmark problem of a backward facing step duct over a thick wall both on GPU and on CPU. The details of the problem are the same as in Reference [2], where we assign three different constant solid to fluid thermal conductivity ratios which are 10, 50 and 100.

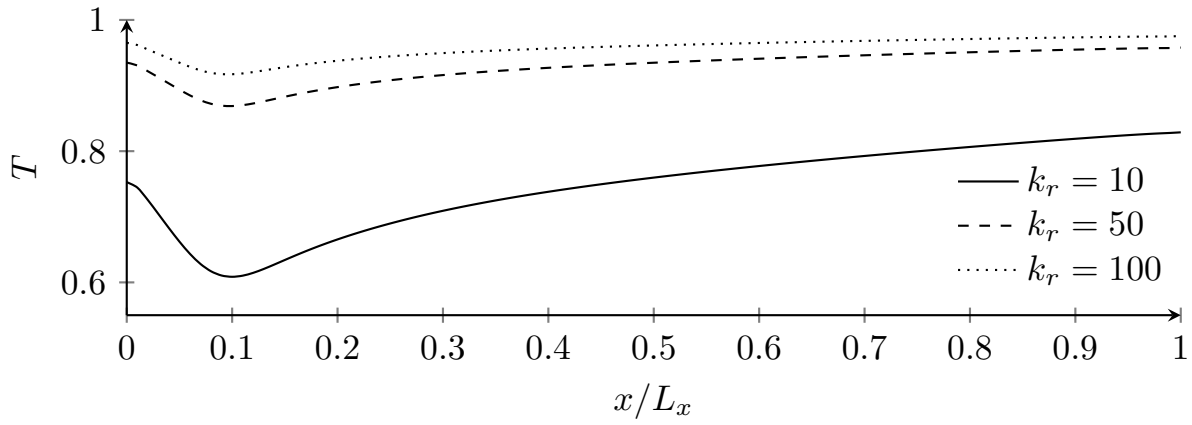

Figure 1: Interface temperature distributions for the backward facing step flow at Re = 200, for $k_r = 10, 50, 100$

Fig. 1 shows the interface temperature distributions for Re = 200, which are obtained with three different thermal conductivity ratios (10, 50 and 100). Our results shows that on average the GPU spends $0.109s$ to complete one cycle iteration, while the CPU computes one iteration in $1.206s$. Tab. 1 contains the computational time for the critical stages of the method. Particularly, the GPU version spends 95% of its execution time by transferring data from device to host. The most dispendious stage for CPU is collision,

which takes about 50% of the total computational time. Whereas the GPU spends 0.079% of its computational time to compute the collision stage. In conclusion, our CHT LBM implementation on GPU is twelve times faster than its CPU version.

Table 1: GPU and CPU execution time comparison in percentage

$\%t$	Streaming	Collision	BC	ρ, V, T	D-D Data Tra.	H-D Data Tra.	Other
CPU	$1.586 \cdot 10^{1}$	$5.011 \cdot 10^{1}$	$3.142 \cdot 10^{-2}$	$3.360 \cdot 10^{1}$	$4.708 \cdot 10^{-2}$	—	$3.515 \cdot 10^{-1}$
GPU	$6.246 \cdot 10^{-3}$	$7.924 \cdot 10^{-3}$	$5.204 \cdot 10^{-2}$	$6.061 \cdot 10^{-3}$	$1.003 \cdot 10^{-2}$	$9.560 \cdot 10^{1}$	$4.318 \cdot 10^{-1}$

5 CONCLUSIONS

We compared the GPU and CPU computational costs of the algorithm by solving the well-known backward facing step flow over a thick wall on a uniform 2D Cartesian mesh. We observed that the collision process is the most dispendious for CPU in terms of computational cost, while for GPU the update of boundary conditions process is the slowest. The results show that running the CHT LBM code on GPU accelerates the solver twelve times.

ACKNOWLEDGMENTS

Gregorio Gerardo Spinelli would like to thank TUBITAK for the Scholarship Program for International Students (Code 2215).

REFERENCES

[1] McNamara, G. R. and Zanetti, G. Use of the Boltzmann Equation to Simulate Lattice-Gas Automata. *Phys. Rev. Lett.* (1988) **61**:2332–2335.

[2] Celik, B. and Spinelli, G. G. Conjugate Heat Transfer Computations Via Lattice Boltzmann Method. *Journal of Aeronautics and Space Technologies* (2017) **10**:49–59.

[3] McClure, J. E., Prinstand, J. F., and Miller, C. T. Comparison of CPU and GPU Implementations of the Lattice Boltzmann Method. In Proceedings of *XVIII International Conference on Water Resources* (2010).

[4] Spinelli, G. G. and Celik, B. Conjugate heat transfer computations with lattice Boltzmann method and comparisons its results with finite volume method. In Proceedings of *IX Ankara International Aerospace Conference* (2017).

[5] Tarokh, A., Mohamad, A. A., and Jiang, L. Simulation of Conjugate Heat Transfer Using the Lattice Boltzmann Method. *Numerical Heat Transfer, Part A: Applications* (2013) **63**:159–178.

[6] Yu, H., Luo, L., and Girimaji, S. S. LES of turbulent square jet flow using an MRT lattice Boltzmann model. *Computers & Fluids* (2006) **35**:957–965.

[7] Perumal, D. A. and Dass, A. K. A Review on the development of lattice Boltzmann computation of macro fluid flows and heat transfer. *Alexandria Engineering Journal* (2015) **54**:955–971.