Computers & Fluids 88 (2013) 241–249

Contents lists available at ScienceDirect

Computers & Fluids

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

Entropic Lattice Boltzmann Method based high Reynolds number flow simulation using CUDA on GPU

Yu Ye [a], Kenli Li [a,b,*]

[a] College of Information Science and Engineering, Hunan University, Changsha, China
[b] National Supercomputing Center of Changsha, Changsha, China

ARTICLE INFO

Article history:
Received 24 November 2012
Received in revised form 9 July 2013
Accepted 20 August 2013
Available online 27 August 2013

Keywords:
Entropic Lattice Boltzmann Method (ELBM)
Computational fluid dynamics
High Reynolds number
Parallelization
CUDA
GPU

ABSTRACT

Entropic Lattice Boltzmann Method (ELBM) is used for the stable computational simulation of high Reynolds number fluid flows, due to it alleviates the obstacle of numerical instabilities by restoring the second law of thermodynamics (Boltzmann's H-theorem). In general, this stability is gained at the price of some computational overhead, associated with the requirement of adjusting the local relaxation parameter of the standard Lattice Boltzmann Method (LBM) in such a way as to guarantee compliance with H-theorem. In this paper, we present a very efficient implementation strategy for ELBM based high Reynolds number flow simulation on nVIDIA graphics processing unit (GPU) with optimization approaches. Some algorithms for H-α solver on GPU which solve the relaxation adjusting parameter are also proposed in our study. We demonstrate the ELBM-GPU parallel approach for fluid flows simulation which can reduce the computational cost of ELBM implementation and obtain an excellent performance. Meanwhile, we find that the direct approximate method of parameter solution is more efficient than other methods on the whole. The results show that: (1) the whole ELBM-GPU implementation results in average speedups of 3.14 over the single-core ELBM-CPU result; (2) comparison of two types of methods for H-α solver, the direct approximate method can save an average 31.7% of computation time than the iteration method; and (3) the implementation of ELBM on GPU allows us to achieve up to 50% global memory bandwidth utilization ratio.

1. Introduction

With the dramatic advances of high-speed digital computer technology, the interest in computational fluid dynamics (CFD) has increased significantly, especially in the last two decades. It is well known that CFD deals with the numerical solution of the Navier–Stokes (N–S) equations of continuum mechanics. In recent years, the Lattice Boltzmann Method (LBM), which originates from the Lattice Gas Cellular Automata (LGA) method [1], has proved to be an efficient alternative to the numerical solving of N–S equations for simulation of complex fluid systems [2–4]. LBM solves the macroscopic dynamics of fluid flows via a mesoscopic approach based on a simplified version of the Boltzmann kinetic equation (that is the fully discrete equation), this approach is attained by calculating transmission and collision of particle. Owing to the kinetic nature of the equation, the pressure field and the stress tensor are locally available without solving any Poisson problem. Sequentially, a lot of applications on LBM are widespread, which range from hydrodynamics at large Reynolds number to fluid flows at a microscale, porous media and multiphase flows [5–7]. Nevertheless, it has been recognized by many authors that the application of the LBM collision operator brings about numerical instability problems [8,9], especially in conjunction with high Reynolds number flows. Therefore, many researchers constantly seek improvements of the stability properties of lattice kinetic schemes. Fortunately, it has been discussed for some time in the literature that stability of LBM could be improved if the method could be based on an analog of the second law of thermodynamics (Boltzmann H-theorem) [8,10–12]. With the adequately progress in this direction, the Entropic Lattice Boltzmann method (ELBM) [13–15] has been established by constructing the collision integral on the basis of discrete-time H-theorem. ELBM can get stable solution by satisfying Boltzamnn H-theorem, according to the monotonicity and the minimality of the H-function [16]. However, this stability comes at the price of some computational overhead, associated with the requirement of adjusting the local relaxation time of the standard LBM in such a way as to secure compliance with the Boltzmann H-theorem. The main computational overhead in ELBM implementation is to solve non-linear equation for the relaxation adjusting parameter α. In order to avoid such problem, a few optimization strategies to reduce computational overhead of ELBM

* Corresponding author at: College of Information Science and Engineering, Hunan University, Changsha, China Tel.: +86 13036781568.
E-mail address: lkl510@263.net (K. Li).

0045-7930/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compfluid.2013.08.005

implementation are required with using both numerical methods [17] and parallel computation [18].

Graphics Processing Unit (GPU) is a massively multi-threaded architecture, it is firstly used for graphical computation and now non-graphical computing to accelerate [19]. The most attractive feature of the GPU is its inherent parallelism, which comes from multiple SIMD processing units. It is because of this characteristic of GPU that it can be adopted properly for LBM simulator on GPU [20]. Particularly, using nVIDIA's Compute Unified Device Architecture (CUDA) extension of the C language to employ GPU for LBM simulation has been shown to provide well speedups and Lattice node Updates Per Second (LUPS) over general-purpose CPU versions [21–23]. Thus, we have been made recently several attempts to implement ELBM on nVIDIA GPU using CUDA. Additionally, to calculate the relaxation adjusting parameter α of ELBM implementation is proven to be very suitable for parallel solution.

In this paper, we intend to provide the implementation of a very efficient strategy for ELBM based high Reynolds number simulation on nVIDIA GPU with optimization approaches. Some algorithms for H-α solver on GPU are also presented and achieved. These implementation are made possible by the use of the nVIDIA CUDA C language programming environment, with some optimization principles more than those ones used in previous work. These principles including maximum of occupancy, utilization of shared memory adequately, sufficient coalesced global memory access and thread divergence rareness, lead us to implement ELBM simulation over a GPU-nVIDIA GeForce card achieving high performance and near 3× speedups over general purpose CPU. Meanwhile, we compare with the total computation cost and the parameter calculation time ratio of several methods, it is found that the direct method of parameter solution is most efficient among such methods which are referred in the paper.

2. The Entropic Lattice Boltzmann Method

It is generally known that there are a lot of literatures about the standard LBM description. For brevity, LBM is based on a threefold discrete form of Lattice Boltzmann equation's time, space and velocity. Velocity space reduces to a finite set of well chosen velocities $\{\boldsymbol{e}_i | i = 0 \cdots N-1\}$, where N is the number of direction of velocity and $\boldsymbol{e}_0 = 0$. Then the evolution equation becomes

$$f_i(\boldsymbol{x}+\boldsymbol{e}_i\delta t, t+\delta t) = f_i(\boldsymbol{x},t) + \theta(f_i^{eq}(\boldsymbol{x},t) - f_i(\boldsymbol{x},t)), \quad i = 0 \cdots N-1 \tag{1}$$

with $f_i(\boldsymbol{x},t) = f(\boldsymbol{x},\boldsymbol{v},t)|_{\boldsymbol{v}=\boldsymbol{e}_i}$ representing the probability of finding a particle at lattice site $\boldsymbol{x}$ at time t, moving along the lattice direction defined by the discrete speed $\boldsymbol{e}_i$. f_i^{eq} is an approximation of the Maxwell–Boltzmann equilibrium distribution function at low Mach numbers and is solely based on hydrodynamic variables, i.e. fluid density ρ and flow velocity $\boldsymbol{u}$, which are given as $\rho = \sum_{i=0}^{N-1} f_i$, $\rho\boldsymbol{u} = \sum_{i=0}^{N-1} f_i \boldsymbol{e}_i$. The symbol θ is called relaxation parameter which equals $\frac{1}{\tau}$ with τ representing a typical single timescale.

Here we focus on the description of ELBM main idea as well as the primary distinction between ELBM and LBM.

2.1. The main idea of ELBM

In order to overcome the inability to attain low viscosity, especially when the flow velocity or spatial gradients are large, ELBM was developed by Karlin et al. [11] and expanded by several researchers recently. They attempt to enhance stability via the H-function, through entropy functions whose local equilibrium are suitable to recover the N–S equations. ELBM solves the kinetic equation Eq. (1) where the relaxation parameter θ is computed with the purpose of satisfying the monotonicity of H-function and H-theorem.

Based on the discussion in this paper is adopting discrete schema, the discrete H-function is given as follows

$$H(\boldsymbol{f}) = \sum_{i=0}^{N-1} f_i \ln\left(\frac{f_i}{\omega_i}\right) \tag{2}$$

where ω_i are the weights associated with the ith particle speed $\boldsymbol{e}_i$ and the definition of f_i, N are the same as above. Beyond that, in D spatial dimensions, the ith particle distribution function $f_i(\boldsymbol{x},t)$ has the following rule:

$$f_i(\boldsymbol{x},t) = \omega_i (2\pi T_0)^{\frac{D}{2}} exp^{\frac{\boldsymbol{v}_i^2}{2T_0}} F(\boldsymbol{x},\boldsymbol{v},t) \tag{3}$$

where $F(\boldsymbol{x},\boldsymbol{v},t)$ is the one-particle distribution function, having position vector $\boldsymbol{x}$, continuous velocity vector $\boldsymbol{v}$ and observation time t, T_0 is the reference temperature.

2.2. The primary distinction between ELBM and LBM

Firstly, the corresponding local equilibrium distribution function is derived not only from the expansion of Maxwell–Boltzmann distribution but also from the minimization of H-function under the constrain of local conservation laws. The explicit expression of f_i^{eq} which stands for the local velocity equilibrium distribution function in ith direction has the following form [24]:

$$f_i^{eq} = \rho\omega_i \prod_{j=1}^{D} (2 - \sqrt{1+u_j^2}) \left(\frac{2u_j + \sqrt{1+3u_j^2}}{1-u_j}\right)^{\frac{v_{ij}}{v}} \tag{4}$$

where j is the index of the spatial direction, u_j is the component of macroscopic velocity in jth direction.

Secondly, the relaxation parameter of standard LBM is constant in general, but the relaxation parameter θ of ELBM is locally adjusted in such a way that the monotonicity of the H-function is satisfied. The monotonicity constrain on the H-function is imposed through a two-steps geometric procedure. In the first step, the H-function remains constant when populations are changed in the direction of the bare collision, $\boldsymbol{\Delta} = \boldsymbol{f}^{eq} - \boldsymbol{f}$(as given by Karlin et al. in [10]); In the second step, dissipation is introduced and the magnitude of the H-function decreases. This delivers the following effective relaxation frequency [16]:

$$\theta = \alpha\beta \tag{5}$$

where

$$\beta = \frac{\delta t}{2\tau + \delta t} \tag{6}$$

Here $\tau > 0$ is the relaxation time related to the kinematic viscosity $v = \tau c_s^2$, where c_s is the speed of sound of the model and δt is the discrete time step.

Furthermore, in Eq. (5), the parameter $\alpha > 0$ which adjusts locally the relaxation time, as dictated by compliance with the H-theorem is obtained by solving the following non-linear equation [10,11].

$$H(\boldsymbol{f}) = H(\boldsymbol{f} + \alpha\boldsymbol{\Delta}) \tag{7}$$

In fact, Eq. (7) is also called the entropy estimate, which means the entropy in the postrelaxation state $\boldsymbol{f} + \alpha\boldsymbol{\Delta}$ is equal to the entropy of the precollision state $\boldsymbol{f}$. It is crucial to remark [17] that when $\boldsymbol{f} \to \boldsymbol{f}^{eq}$, the solution $\alpha \to 2$ and Eq. (1) recover the stand LBM.

As the end of this section, we have the necessity to mention Reynolds number. Normally, Reynolds number is defined as following equation: