Journal of Petroleum Science and Engineering 156 (2017) 546–552

Contents lists available at ScienceDirect

Journal of Petroleum Science and Engineering

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

GPU-accelerated volumetric lattice Boltzmann method for porous media flow

Senyou An [a,b], Huidan(Whitney) Yu [b,*], Jun Yao [a,**]

[a] School of Petroleum Engineering, China University of Petroleum, Qingdao, 266580, China
[b] Department of Mechanical and Energy Engineering, Indiana University-Purdue University, Indianapolis, IN, 46202, USA

ARTICLE INFO

Keywords:
Porous media
GPU parallel
Volumetric lattice Boltzmann method
Digital core
Petroleum

ABSTRACT

The volumetric lattice Boltzmann method (VLBM) has been recently developed and validated for dealing with flows in complex geometries. To reveal the intricate and arbitrary porous media skeleton, VLBM categorizes the computational domain into fluid, solid, and boundary cells by introducing a volumetric parameter $P(\vec{x})$, through which the lattice Boltzmann equations are self-regularized. As a result, the no-slip bounce-back boundary condition at the inter walls is integrated in the streaming term. Since its data structure is aligned and kernel pattern is clear, VLBM is ideally suited for GPU parallelization. Using the $P(\vec{x})$ in the streaming operation, branch diverse can be effectively decreased. In this paper, we use several optimization methods, such as memory arrangement and kernel design, to maximize the performance of parallelization for VLBM. As an application, we simulated petroleum flow in a digital sandstone with two resolutions, 256^3 and $256^2 \times 512$, and evaluated its permeability. The best parallel performance reaches 808.7 MLUPS (Million Lattice Updates Per Second), which is 1421.3-times speedup compared with the serial computation with allocated memory.

1. Introduction

The lattice Boltzmann method (LBM) (Benzi et al., 1992; Chen et al., 1992) has become a popular alternative to traditional Navier-Stoke (NS) equation solvers (e.g. finite element method, finite volume method), especially for incompressible and time-dependent flow (Chen and Doolen, 1998; Aidun and Clausen, 2010). As a heritage from cellular automaton, the LBM simulates fluid dynamics via prescribed discrete kinetic equations for time evolution of discrete particle density distribution functions due to molecular interaction. The macroscopic properties such as velocity, pressure and wall shear stress, are the macro-scale reflection of particle distribution. Mathematically, the incompressible lattice Boltzmann equation can recover NS equations based on BGK(Bhatnagar-Gross-Krook) collision approximation (Bhatnagar et al., 1954) and Chapman-Enskog technique (Chapman and Cowling, 1970) to the second-order accuracy in both space and time (Chen and Doolen, 1998).

In the traditional LBM, fluid particles are set on the node points and the computational domain is usually characterilized by 0 and 1, representing fluid and solid. The particle distribution functions represent the corresponding cell's density layout linked with momentum distribution. In the iterative evolution, particles' collisions occur in host cells, and then the particles stream to adjust grids along their velocity directions. When a lattice cell is cut by arbitrarily curved boundaries, either a fluid or solid node has to be determined via the volume fraction of solid. Such a treatment may alter the real flow domain, which can be significantly in accurate in porous media flow. To improve the accuracy, the computational resolution must be very fine, causing demandingly high computation cost. Unstructured mesh may ease the computation demand (Peng et al., 1999; Li et al., 2005), but it is not applicable for porous media flow because there will be difficulties in small isolated areas or extreme tip points in random porous structure (Aavatsmark et al., 1998). The unstructured mesh may also weaken those strong points of structured LBM in GPU parallelization (Qian et al., 1995; Feichtinger et al., 2011). Recently, Yu, et al. developed a mass-conserved volumetric lattice Boltzmann method (VLBM) (Yu et al., 2014), in which fluid particles are uniformly distributed in each lattice cell. The computational domain are categorized through fluid, solid, and boundary cells by introducing a volumetric parameter $P(\vec{x})$, defining the solid fraction in each cell. The volumatic lattice Boltzmann equations are self-regulated through $P(\vec{x})$. In VLBM, the no-slip bounce-back boundary condition at the inter walls is integrated in the streaming term. The introduction of volumatic

* Corresponding author.
** Corresponding author.
E-mail addresses: whyu@iupui.edu (H. Yu), rcogfr_upc@126.com (J. Yao).

http://dx.doi.org/10.1016/j.petrol.2017.06.031
Received 3 May 2017; Received in revised form 13 June 2017; Accepted 14 June 2017
Available online 17 June 2017
0920-4105/Published by Elsevier B.V.

representation makes an enormous contribution to accurately describe complex boundaries with no compromise of fine resolution, which is great for porous media flow.

The VLBM is ideally suitable for GPU parallelization. Previously, Wang, et al. parallelized VLBM for simulating blood flow in human arteries (Wang et al., 2015). The performance was about 100 MLUPS (Million Lattice node Updates Per Second). This parallel scheme ended up with 10 MLUPS for crude oil flow in digital stone, implying a need of more advanced parallel schemes. In this work, we parallelize the VLBM by highly memory-efficient technologies, including memory arrangement and kernel structure design. First, we have adopted the modified tiling algorithm from Tran N P, et al. (Tran et al., 2015) to minimize the effect of the uncoalesced accesses. Second, we develop a new allocation method for CPU memory to guarantee enough continued space for reading geometry data. Third, we optimize the parallelization by removing branch divergences, arranging the register usage and combining streaming and collision into a kernel to maximize the acceleration of computation. To demonstrate the application, we simulate porous media flow in a digital core by using the GPU-accelerated VLBM.

The remainder of the paper is organized as follows. Section 2 introduces the mathematical formulation of VLBM. The high-efficient GPU parallelization for VLBM is presented in Sec.3. An application about crude oil flow in porous media is studied in Sec.4. Finally, Sec.5 provides a summary discussion and concludes the paper.

2. Volumetric lattice Boltzmnn method

The VLBM has been introduced in the previous paper in detail (Yu et al., 2014). Herein, we just cite the foundational concepts and equations for better comprehension of parallel computation. VLBM is designed to accurately and conveniently deal with boundaries by defining solid occupation function $P(\vec{x})$. Just like Fig. 1, extracted for vertical direction of the D3Q19 model, $P(\vec{x}) = 0$ is used to represent fluid domain, and the shaded (solid) zone means $P(\vec{x}) = 1$. When the cell is crossed by the boundary line, it will have partial fluid and partial solid. Correspondingly, the solid occupation function will be between 0 and 1. So the arbitrary structure can be described with less grids compared with traditional LBM.

The fluid particles are sited in lattice cells, as opposed to distributed on the nodes in the conventional LBM. To reflect the effect of Pvalue $P(\vec{x})$, the distribution function is defined as $n_i(\vec{x}, t) = f_i(\vec{x}, t)\cdot P(\vec{x})$ with velocity $\vec{e}_i$ occupying a lattice cell $\vec{x}_i$ at time t and it deals with the time evolution of the particle distribution function analogy to LBE.

$$n_i(\vec{x} + \vec{e}_i\Delta t, t + \Delta t) = n_i(\vec{x}, t) + \Omega_i(\vec{x}, t) \tag{1}$$

where $\Omega_i(\vec{x}, t)$ is a collision term due to the molecule particles' motion and the momentum exchange between them. The most common calculation for this part is the BGK model with a single-scale relaxation time τ, as Eq. (2). And $i = 0, 1, 2, \cdots, b$ represents predefined directions of molecular motion.

$$\Omega_i(\vec{x}, t) = -\frac{1}{\tau}[n_i(\vec{x}, t) - n_i^{eq}(\vec{x}, t)] \tag{2}$$

where the $n_i^{eq}(\vec{x}, t)$ is the equilibrium function, which is formed as:

$$n_i^{eq}(\vec{x}, t) = N(\vec{x}, t)\omega_i\left[1 + \frac{\vec{e}_i\cdot\vec{u}}{c_s^2} + \frac{(\vec{e}_i\cdot\vec{u})^2}{2c_s^4} + \frac{(\vec{u})^2}{2c_s^2}\right] \tag{3}$$

where $N(\vec{x}, t) = \sum n_i(\vec{x}, t)$ is density at the current cell, ω_i is weight fraction of ith velocity direction, and c_s is speed velocity.

To avoid confusion, streaming and collision should be separated in the calculation. After collision, temporary array "postcollision" is defined as $n'_i(\vec{x}, t)$, as the following equation:

$$n'_i(\vec{x}, t) = n_i(\vec{x}, t) + \Omega_i(\vec{x}, t) \tag{4}$$

Streaming means that the fluid particles move from the current cell to neighboring cells. The key point for VLBM is the streaming part, as it considers the fluid volume fraction of boundary cells. Only a specific fluid can stream to these cells, meaning the other part will be bounced back to the host cell. As illustrated in Fig. 1, the 11, 5 and 12 direction velocities contain two parts, particles streaming from previous cells (black solid arrow) $[1- P(\vec{x}, t)]n'_i(\vec{x}- \vec{e}_i\Delta t, t)$ and bounce-back fluid from downwind boundary cells, $P(\vec{x}+ \vec{e}_{i\star}\Delta t, t)n'_{i\star}(\vec{x}_i, t)$, as shown in following equation:

$$n''_i(\vec{x}, t + \Delta t) = [1 - P(\vec{x}, t)]n'_i(\vec{x} - \vec{e}_i\Delta t, t) + P(\vec{x} + \vec{e}_{i\star}\Delta t, t)n'_{i\star}(\vec{x}_i, t) \tag{5}$$

where $i^\star$ corresponds to the opposite velocity direction with i direction, meaning $\vec{e}_{i^*} = -\vec{e}_i$. This modified streaming process ensures that particles are advected or reflected to their appropriate places in the fluid domain, but does not introduce any extra mass.

The resulting density, velocity and pressure are obtained as follows:

$$\rho(\vec{x}, t) = \sum n_i(\vec{x}, t)/[1 - P(\vec{x}, t)] \tag{6}$$

$$\vec{u}(\vec{x}, t) = \sum \vec{e}_i n_i(\vec{x}, t) \Big/ \sum n_i(\vec{x}, t) \tag{7}$$

$$p(\vec{x}, t) - p_0 = c_s^2[\rho(\vec{x}, t) - \rho_0] \tag{8}$$

where p_0 and ρ_0 are original reference pressure and density, respectively.

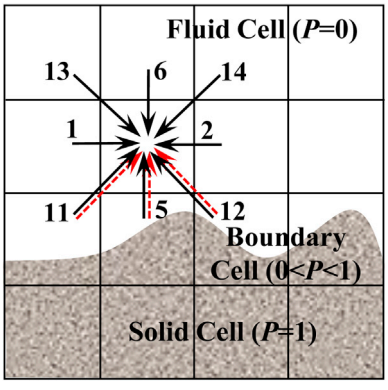

Fig. 1. Illustration of volumetric representation of cells using solid ratio P: $P = 0$ (fluid), $P = 1$ (solid), $0 < P < 1$ (boundary).

3. GPU parallelization for VLBM

One of the most important strengths of LBM is ideally suitable for GPU parallel. In this section, we introduce how to realize memory efficient parallelization for D3Q19-based VLBM and optimize it to high speed, including dynamic allocation, coalesced global memory, register arrangement, and kernel structure design.

3.1. Dynamic allocation for CPU

The limits of CPU static zone memory and GPU device memory are essential factors to block acceleration and improvement of model size, especially when the big data file exists in the data transformation between host and device. For porous media flow, structure information and hydrodynamic parameters (Pvalue, density, velocity and pressure), are essential arrays to be defined for output in CPU host. Distribution function arrays are applied, initialized and updated in GPU directly. Assuming the model size is $512 \times 512 \times 512$ and all the data is float type,