XVIII International Conference on Water Resources
CMWR 2010
J. Carrera (Ed)
©CIMNE, Barcelona, 2010

COMPARISON OF CPU AND GPU IMPLEMENTATIONS OF THE LATTICE BOLTZMANN METHOD

James .E. McClure[*], Jan F. Prins[†] and Cass T. Miller[*]

[*] Department of Environmental Sciences and Engineering
CB 7431, 148 Rosenau Hall
School of Public Health
University of North Carolina
Chapel Hill, North Carolina 27599-7431

[†]Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina 27599-7431

Key words: Lattice Boltzmann Methods, Permeability Estimation, Graphics Processor, GPU, Multi-relaxation time, MRT

Summary. The lattice Boltzmann Method (LBM) has become a standard tool for estimating porous medium permeabilities from image data and numerically generated packings. We consider implementations of the single-relaxation time BGK scheme for single phase flow as well as a more computationally intensive multi-relaxation time (MRT) scheme. Results demonstrate that a considerable performance increase is achieved by implementing on graphics processing unit (GPU) for both methods. The MRT scheme is shown to provide a more efficient means for permeability estimation on GPU relative to the BGK approach. The increased accuracy of the MRT scheme allows accurate permeability measurements to be obtained at lower resolutions, more than offsetting the increased computational cost associated with MRT.

1 INTRODUCTION

Utilization of graphics processing units (GPUs) for high-performance computing can demonstrate significant performance benefits for memory-bandwidth limited applications [1, 2]. Many LBM schemes rely on local information only, a computational structure which is ideally suited to GPUs. Relative to CPU implementations, GPU implementations of the lattice Boltzmann (LBM) often achieve performance increases of an order of magnitude.

Single component LBM solve the Navier-Stokes equations by exploiting the relationship between kinetic and continuum theory. Such schemes offer significant advantages over numerical Navier-Stokes solutions when massively parallel implementations are required or where complex boundary conditions must be accommodated. These requirements have resulted in a proliferation in the use of LBM for porous medium applications, where large domains and complex solid boundary conditions are requisite. The LBM has become a standard tool for estimation of porous medium permeabilities [3]. Multi-relaxation time (MRT) schemes provide much more accurate permeability estimates compared to the less computationally-intensive BGK scheme[4]. In this work, we compare the computational behavior of the MRT and BGK LBM on CPU and GPU. These results motivate the use of GPUs for permeability estimation in porous media. We consider the tradeoff between increased accuracy of MRT and its increased computational cost relative to the BGK scheme.

2 METHODS

2.1 The Lattice Boltzmann Method

In this work,we two commonly used implementations of the LBM for modeling single phase flow, both of which use a three-dimensional, nineteen velocity vector (D3Q19) structure. The computational domain is defined by a rectangular prism with sides of length N_x, N_y and N_z. The domain is discretized by regularly spaced lattice nodes $\boldsymbol{x}_i$ where $i = 0, 1, \ldots, N-1$, $N = N_xN_yN_z$. Our implementations utilize a three-dimensional nineteen velocity (D3Q19) model:

$$\boldsymbol{\xi}_q = \begin{cases} \{0,0,0\}^T & \text{for } q = 0 \\ \{\pm1,0,0\}^T, \{0,\pm1,0\}^T, \{0,0,\pm1\}^T & \text{for } q = 1,2,\ldots,6 \\ \{\pm1,\pm1,0\}^T, \{\pm1,0,\pm1\}^T, \{0,\pm1,\pm1\}^T & \text{for } q = 7,8,\ldots,18 \end{cases} \tag{1}$$

Each velocity vector $\boldsymbol{\xi}_q$ is associated with a distribution f_q/ The fluid density and velocity are obtained from these distributions:

$$\rho = \sum_{q=0}^{Q-1} f_q, \tag{2}$$

$$\mathbf{u} = \frac{1}{\rho}\sum_{q=0}^{Q-1} \boldsymbol{\xi}_q f_q. \tag{3}$$

Since density and velocity can be obtained directly from the distributions, a numerical solution for f_q implies a solution for ρ and $\mathbf{u}$. The solution for the LBM is provided by solving the lattice Boltzmann equation (LBE), which may be generally expressed in the form:

$$f_q(\boldsymbol{x}_i + \boldsymbol{\xi}_q, t+1) - f_q(\boldsymbol{x}_i, t) = \mathcal{J}_q, \tag{4}$$

where $\mathcal{J}_q$ is a collision operator which accounts for changes in f_q due to the intermolecular collisions. Solution of Eq. 4 is usually accomplished in two steps, referred to as streaming:

$$f_q^*(\boldsymbol{x}_i + \boldsymbol{\xi}_q, t+1) = f_q(\boldsymbol{x}_i, t), \tag{5}$$

and collision:

$$f_q(\boldsymbol{x}_i, t+1) = f_q^*(\boldsymbol{x}_i, t+1) + \mathcal{J}_q^*. \tag{6}$$

Permeability estimates are obtained by assuming the macroscopic flow obeys Darcy's law:

$$-\frac{\partial p}{\partial x} + \rho g = \frac{\rho \eta v}{\kappa}, \tag{7}$$

where p is the pressure, ρg is the external force, η is the fluid viscosity, v is the mass averaged velocity over the domain and κ is the permeability.

2.1.1 Single-Component BGK Model

The BGK model assumes that the distributions relax at a constant rate toward equilibrium values f_q^{eq} prescribed by the Maxwellian distribution. The collision term for the LBGK model is:

$$\mathcal{J}_q = \frac{1}{\tau}\Big(f_q^{eq} - f_q\Big). \tag{8}$$

The relaxation rate is specified by the parameter τ, which relates to the fluid viscosity:

$$\eta = \tfrac{1}{3}\Big(\tau - \tfrac{1}{2}\Big). \tag{9}$$

The equilibrium distributions approximate the Maxwellian distribution, For the D3Q19 model, they take the form:

$$f_q^{eq} = w_q \rho \Big[1 + 3(\boldsymbol{\xi}_q \cdot \mathbf{u}) + \tfrac{9}{2}(\boldsymbol{\xi}_q \cdot \mathbf{u})^2 - \tfrac{3}{2}(\mathbf{u} \cdot \mathbf{u})\Big] \tag{10}$$

where $w_0 = \frac{1}{3}$, $w_{1,\ldots,6} = \frac{1}{18}$ and $w_{7,\ldots,18} = \frac{1}{36}$. This choice of equilibrium conditions ensures that mass and momentum will be conserved.

2.1.2 Single-Component Multi-Relaxation Time Model

In the BGK approximation to the collision term given by Eq. 8, all non-conserved hydrodynamic modes relax toward equilibrium at the same rate. Multi-relaxation time schemes are constructed so that different hydrodynamic modes may relax at different rates. In vector space, many physically significant hydrodynamic modes are associated with linear combinations of the distributions f_q^w:

$$\hat{f}_m = \sum_{q=0}^{Q-1} \alpha_{m,q} f_q, \tag{11}$$

where $\alpha_{m,q}$ represents a set of constant coefficients associated with a particular mode m. Since we may specify Q independent, linear combinations of f_q, coefficients must be defined for $m = 0, 1, 2, \ldots, Q-1$. The coefficients must be chosen carefully in order to ensure that the moments correspond with physical modes which are hydrodynamically significant. The coefficients $\alpha_{m,q}$ are obtained by applying a Gram-Schmidt orthogonalization to polynomials of the discrete velocities [5]. The resulting set of moments include density, momentum and kinetic energy modes, as well as modes associated with elements of the stress tensor. The relaxation process is carried out in moment space, with each mode relaxing at its own rate specified by λ_m:

$$\mathcal{J}_q = \sum_{m=0}^{Q-1} \alpha^*_{q,m} \lambda_m \Big(\hat{f}_m^{eq,} - \hat{f}_m \Big). \tag{12}$$

The inverse transformation coefficients $\alpha^*_{q,m}$ map the moments back to distribution space, and represent the matrix inverse of the transformation coefficients $\alpha_{m,q}$. The equilibrium moments $\hat{f}_m^{eq,}$ are functions of the local density ρ and momentum $\mathbf{j}$. The relaxation parameters are chosen to minimize viscosity dependence on permeability:

$$\lambda_1 = \lambda_2 = \lambda_9 = \lambda_{10} = \lambda_{11} = \lambda_{12} = \lambda_{13} = \lambda_{14} = \lambda_{15} = \lambda_A = \frac{1}{\tau} \tag{13}$$

$$\lambda_4 = \lambda_6 = \lambda_8 = \lambda_{16} = \lambda_{17} = \lambda_{18} = \lambda_B = \frac{8(2-\lambda_A)}{8-\lambda_A} \tag{14}$$

2.1.3 GPU implementation of the LBM

Each GPU consists a very large number of cores capable of handling computations. The GPU is utilized most efficiently when each core is able to execute computations independently from other cores. CUDA is a programming language designed with

GPU applications in mind. Computations are divided among the GPU cores by decomposing the computational domain into a grid of threadblocks which may be executed in any order. This ensures code which scales proportional to the number of cores, while imposing algorithmic constraints on the programmer. Unlike CPU applications, which can take advantage of specialized computations to achieve optimal performance, CUDA relies on computational kernels executed by each thread.

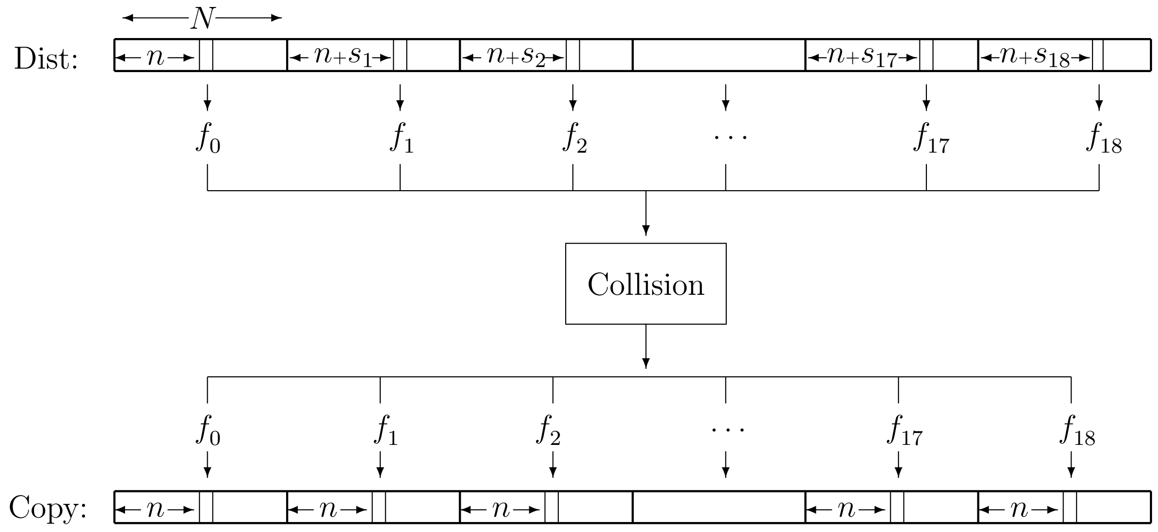


Figure 1: Schematic illustration of array structure and kernel execution in CUDA.

The structure of the kernel is illustrated by Fig. 1. The nineteen distributions are packed into a single array which is indexed such that Dist$[N*q+i] = f_q(\boldsymbol{x}_i, t)$. In order to fuse the streaming and collision operations into a single memory access, the two-array streaming (TAS) algorithm was used. Streaming is accomplished by reading registers from the array using a set of offsets s_q to pull distributions from the adjacent lattice nodes. The streaming offsets are defined by:

$$s_q = \boldsymbol{\xi}_q \cdot \left\{ \begin{array}{c} 1 \\ N_x \\ N_x N_y \end{array} \right\}. \tag{15}$$

The registers, denoted by $f_0, f_1, \ldots f_{18}$ in Fig. 1, are used to carry out the collision step. For BGK, additional registers must be allocated to store the density ρ and velocity $\mathbf{u}$. These moments are included in the nineteen additional registers required by MRT. At the implementation level, only two relaxation parameters λ_A and λ_B

are passed to the MRT kernel in CUDA. After the collision process has been applied, $f_0, f_1, \ldots f_{18}$, are written a copy of the distribution array, indexed in the same way.

3 RESULTS

Performance of BGK and MRT LBM was evaluated for dense lattice sizes ranging from $N = 16^3$ to $N = 192^3$ on GPU and $N = 8^3$ to $N = 128^3$ on CPU. Nvidia Quadro FX 5600 cards were used to perform GPU calculations, whereas CPU calculations were performed on a 3.0 gHz Intel Clovertown processor. A highly optimized swap implementation was used to execute the streaming step on the CPU, shown to be far superior to TAS for CPU [6].

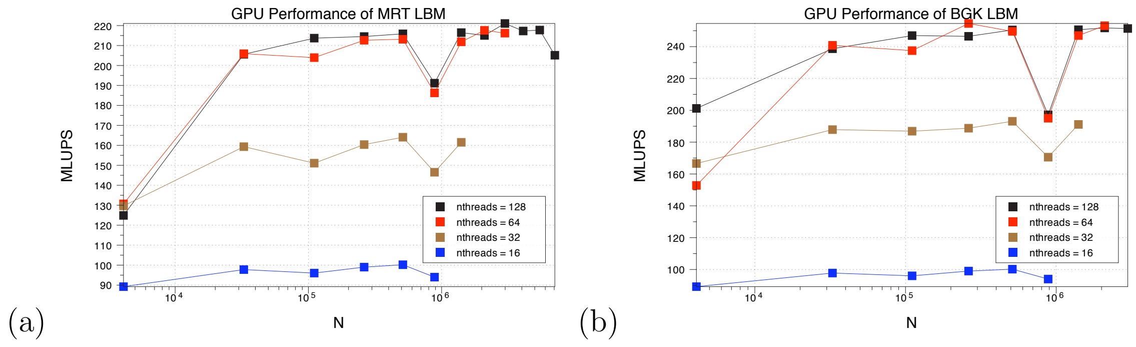
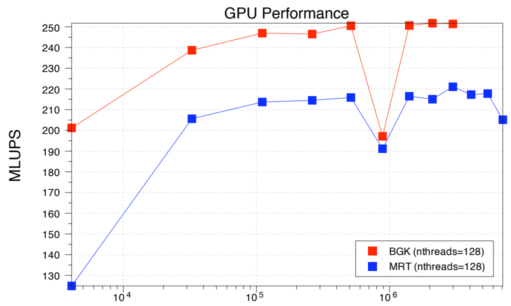


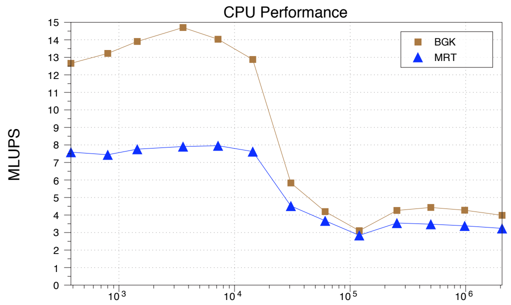
Figure 2: GPU performance of the LBM for various thread counts for (a) BGK (b) MRT.

Fig. 2 shows the performance of BGK and MRT implementations as a function of domain size with a range of thread counts. Performance is evaluated in terms of million-lattice-updates-per-second (MLUPS). Performance of the LBM on GPU depends on the number of threads executing at a time. Optimal results are achieved with the number of threads equal to 64 or 128. GPU performance is very sensitive to coalescence of the read and write operations performed for each thread-block. For certain domain sizes, lack of coalescence can lead to a significant drop in the number of MLPUS observed. The number of MLUPS observed for BGK shows excellent agreement with other results published for BGK LBM on GPU [1]. Results for MRT LBM demonstrate that the MRT approach is competitive with BGK in terms of MLUPS in spite of increased computations. This is consistent with the observation that the LBM is primarily limited by memory bandwidth, such that more computationally intensive methods may be accommodated easily.

Comparison of GPU and CPU implementations verify that GPU performance is an order of magnitude faster for lattice sizes $N \geq 32^3$, as shown in Fig. 3 (a) and



(a)



(b)

Figure 3: Performance for MRT and BGK schemes for (a) GPU (b) CPU

(b). On CPU, a significant performance dropoff is observed once the size of the distribution arrays exceed the cache size. Due to the fixed overhead associated with kernel execution, GPU is more advantageous for larger domain sizes, which favors porous medium applications. For a dense lattice, the GPU implementation represents a factor of nearly fifty times more MLUPS relative to the CPU. This performance increase is attributed to the higher memory bandwidth of the GPU as well as the fact that the GPU calculations are performed in float rather than double. Algorithm differences impose slightly different memory demands for the two schemes.

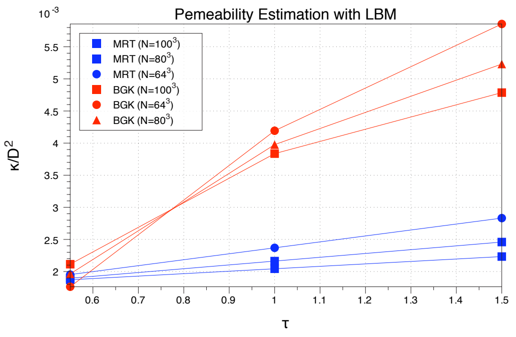


Figure 4: Dimensionless permeability estimates obtained via LBM for a range of relaxation parameters τ. The length scale D is the number of pixels per sphere radius.

The impact of method choice on permeability estimation was investigated using a random sphere packing of 125 equally sized spheres in a cubic domain. The porosity of the packing was 0.37642. Permeability results were obtained via LBM by discretizing the packing to obtain domains of size 64^3, 80^3 and 100^3. The results are shown in Fig. 4 for both BGK and MRT at each discretization level. Permeability estimates demonstrate that the computational cost of the MRT scheme is more than compensated for by the increased accuracy of the scheme. Results demonstrate that larger domains must be considered to mitigate the strong viscosity dependence observed for the BGK approach. The cost associated with simulation of larger domain sizes far exceeds the increased computational expense per iteration associated with MRT.

4 CONCLUSIONS

Implementation of BGK and MRT lattice Boltzmann schemes demonstrate the advantages of the GPU; performance on GPU is fifty times faster than CPU. Additional registers are required for the MRT scheme, but this does not inhibit its application on GPU. As with CPU implementation, execution of the streaming step is the primary performance bottleneck on GPU. Viscosity dependence of BGK permeability estimates implies that much larger domain sizes must be considered to achieve reliable permeability estimates. MRT offers significant benefits even for the current generation of GPUs, where register space is at a premium.

REFERENCES

[1] A. Kaufman, Z. Fan, K. Petkov, Implementing the lattice Boltzmann model on commodity graphics hardware, JOURNAL OF STATISTICAL MECHANICS-THEORY AND EXPERIMENTdoi:10.1088/1742-5468/2009/06/P06016.

[2] Z. Fan, F. Qiu, A. E. Kaufman, Zippy: A framework for computation and visualization on a GPU cluster, COMPUTER GRAPHICS FORUM 27 (2) (2008) 341–350.

[3] C. Pan, M. Hilpert, C. T. Miller, Pore-scale modeling of saturated permeabilities in random sphere packings, Physical Review E 64 (6) (2001) 9.

[4] C. Pan, L.-S. Luo, C. T. Miller, An evaluation of lattice Boltzmann schemes for porous medium flow simulation, Computers & Fluids 35 (8–9) (2006) 898–909.

[5] D. d'Humiéres, I. Ginzburg, M. Krafczyk, P. Lallemand, L. S. Luo, Multiple-relaxation-time lattice Boltzmann models in three dimensions, Philosophical Transactions of the Royal Society of London Series A-Mathematical Physical and Engineering Sciences 360 (2002) 437–451.

[6] K. Mattila, J. Hyvaeluoma, J. Timonen, T. Rossi, Comparison of implementations of the lattice–Boltzmann method, Computers & Mathematics with Applications 55 (7) (2008) 1514–1524.