Case Study for Performance-Portability of Lattice Boltzmann Kernels

1st Geng Liu
Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, US
gliu@anl.gov

2nd Joseph Insley
Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, US
insley@anl.gov

3rd Saumil Patel
Computational Science
Argonne National Laboratory
Lemont, IL, US
spatel@anl.gov

4th Silvio Rizzi
Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, US
srizzi@alcf.anl.gov

5th Victor Mateevitsi
Leadership Computing Facility
Argonne National Laboratory
Lemont, IL, US
vmateevitsi@anl.gov

6nd Amanda Randles
Biomedical Engineeering
Duke University
Durham, NC, US
arandles@duke.edu

***Abstract*—In this work, we study the performance-portability of offloaded lattice Boltzmann kernels and the trade-off between portability and efficiency. The study is based on a proxy application for the lattice Boltzmann method (LBM). The performance portability programming framework of Kokkos (with CUDA or SYCL backend) is used and compared with programming models of native CUDA and native SYCL. The Kokkos library supports the mainstream GPU products in the market. The performance of the code can vary with accelerating models, number of GPUs, scale of the problem, propagation patterns and architectures. Both Kokkos library and CUDA toolkit are studied on the supercomputer of ThetaGPU (Argonne Leadership Computing Facility). It is found that Kokkos (CUDA) has almost the same performance as native CUDA. The automatic data and kernel management in Kokkos may sacrifice the efficiency, but the parallelization parameters can also be tuned by Kokkos to optimize the performances.**

***Index Terms*—performance-portability, lattice Boltzmann method, Kokkos, CUDA.**

I. Introduction

The lattice Boltzmann method (LBM) has become increasingly popular in simulating fluids and is widely used in many industrial, engineering and environmental processes. LBM [1] is an alternative approach to solve the Navier-Stokes equations (NSE) in the low-Mach number regime and its governing equation can be derived from the Boltzmann equation after discretizing the phase space with constant microscopic lattice velocities. Based on the discrete velocity model (DVM) proposed by Qian et al. [2], LBM solves the fluid field by iteratively applying collision and propagation algorithms to the particle probability distribution functions (PDFs). One major drive behind the use of LBM in the CFD community is the ease of parallelization.

The portability of the lattice Boltzmann codes needs to be considered to support different architectures. In this work, the GPU performance comparison of different models is investigated. This will help users of LBM or other similar stencil applications to find a better accelerating tool without sacrificing the performance-portability. Therefore it is especially important to compare the performances of those portable models with changeable backends. To assess the predicted performance of applications using the lattice Boltzmann or other similar stencil methods, a lattice Boltzmann proxy application, developed by John Gounley from Oak Ridge National Laboratory, is employed. The proxy application implements the Bhatnagar–Gross–Krook collision model with standard lattices (D3Q19, D3Q27, and D3Q27) for 3D simulations and three common lattice Boltzmann propagation patterns (AA, AB push and AB pull). Additionally, to focus on the performance of complex geometries (such as common use cases of vascular geometries or porous media) where a non-direct addressing scheme may be advantageous, the proxy app uses a standard indirect addressing scheme.

A specific focus of the proxy application is on performance-portability, in order to be informative for future work to support different GPU architectures. To this end, the proxy app has been developed to include memory configuration and compute kernels using native CUDA, native SYCL and Kokkos. Kokkos is a C++ library that aims at unifying different low-level parallel programming models such as OpenMP, CUDA, SYCL, HIP, etc. [3], [4] This framework can support building cross-platform applications and is said to achieve performance-portability with a single codebase. The arrays managed by Kokkos are in the form of Views. The Kokkos Views can be constructed on devices by simply specifying the memory space according to selected backend. Kokkos kernels are also executed in specified execution space. A performance portability programming framework like Kokkos not only manages the memory and execution spaces, but also controls the parallelization by automatically tuning the execution parameters. The trade-off between portability and efficiency is therefore related to whether the kernels are computationally intensive, and whether the memory accessing is indirect and complicated. The performances of the three

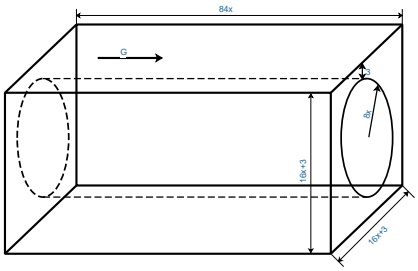


Fig. 1: Geometry setup of the test problem.

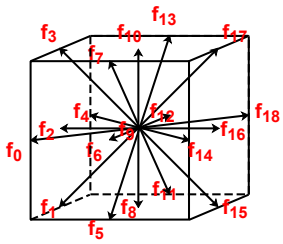


Fig. 2: D3Q19 stencil.

propagation patterns will show whether it is worth applying the performance portable framework to similar stencil applications.

II. Implementation Details

In LBM Proxy App, the test problem is a 3D pressure driven cylinder channel flow. The geometry can be depicted by Fig. 1. The axial length of the cylinder is 84x and radius is 8x, where x is a scale factor. The domain is periodic in the axial direction. Although the cylinder channel is located in a box, only the fluid points inside the cylinder participate in the computation. Nodal bounce back [5] is applied to the channel wall points. The fluid in the channel is driven by an evenly distributed body force $\mathbf{G}$.

The lattice Boltzmann governing equation with BGK collision operator [6] can be expressed by

$$\text{collision:} \quad f_\alpha^* = f_\alpha - \omega(f_\alpha - f_\alpha^{\text{eq}}) + (1 - \frac{\omega}{2})F_\alpha, \tag{1}$$

$$\text{propagation:} \quad f_\alpha(\mathbf{x}, t) = f_\alpha^*(\mathbf{x} - \mathbf{e}_\alpha \delta_t, t - \delta_t), \tag{2}$$

where f_α is the PDF and f_α^* is the post-collision PDF, ω is the relaxation frequency and F_α is the external force term. The PDF defined at position $\mathbf{x}$ and time t moves at a velocity of $\mathbf{e}_\alpha$ per δ_t time increment. The equilibrium PDF f_α^{eq} can be expressed by

$$f_\alpha^{eq} = t_\alpha \rho \left[1 + \frac{\mathbf{e}_\alpha \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{e}_\alpha \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u} \cdot \mathbf{u}}{2c_s^2} \right], \tag{3}$$

where t_α is the corresponding weight, $c_s = \sqrt{1/3}$ is the speed of sound. And the force term can be expressed by

$$F_\alpha = (\mathbf{e}_\alpha - \mathbf{u}) \cdot \mathbf{G} + (\mathbf{e}_\alpha \cdot \mathbf{u})(\mathbf{e}_\alpha \cdot \mathbf{G}). \tag{4}$$

The fluid density ρ and fluid velocity $\mathbf{u}$ can be recovered by taking the moments of PDFs:

$$\rho = \sum_\alpha f_\alpha, \tag{5}$$

$$\mathbf{u} = \frac{1}{\rho} \left(\frac{\mathbf{G}}{2} + \sum_\alpha \mathbf{e}_\alpha f_\alpha \right). \tag{6}$$

α ranges from 0 to 18 because the DVM used in this paper is D3Q19 (Fig. 2).

III. Performance Results

The performances are evaluated by Mega Fluid Lattice Updates per Second (MFLUPS). This value m can be expressed by:

$$m = \frac{\text{lattice points in the whole domain} \times \text{ iterations}}{\text{computing time} \times 10^6}. \tag{7}$$

The performance data for different propagation patterns and different programming models are shown in Fig. 3, where the scale factor x is 16 and the implementation is on a single node (8 NVIDIA A100 GPUs with 40 GB memory space) of ThetaGPU. The factor x is chosen to make full use of the device memory.

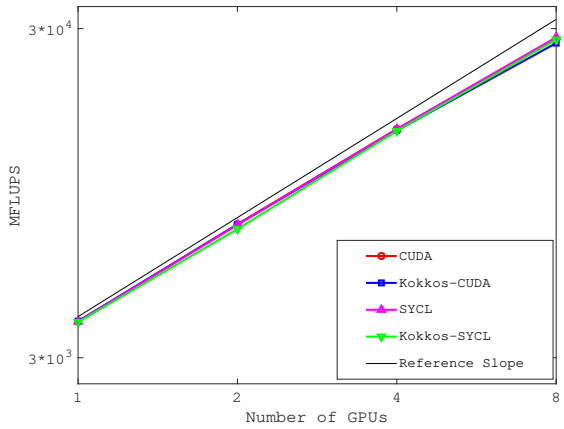
From the results we can see that AA pattern is in general faster than AB patterns. Within the same propagation pattern, the performances of Kokkos with CUDA backend, native CUDA and native SYCL are almost identical. The performance of Kokkos with SYCL backend has similar behavior for the AA pattern and AB pull pattern, but is not as good for AB push pattern on NVIDIA devices. A possible explanation is that the AB push pattern has a more complicated memory accessing mechanism, which makes the Kokkos optimization for SYCL backend work unexpectedly. The performance of Kokkos with SYCL backend for AB push pattern can be as minimal as 70% of native CUDA performance.

IV. Future Work

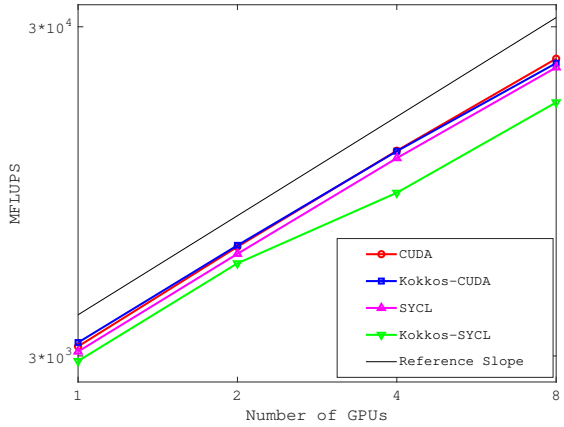
Roofline plots and other profiling data will be collected to further explain the behaviors of the tested programming models on multiple platforms. Native SYCL and Kokkos (SYCL), Kokkos (OpenMP Target) are also going to be applied to the proxy application on non-NVIDIA devices. The profiling data and the comparisons will contribute to migrating stencil applications, whose algorithms are similar to LBM, to different parallel computing platforms.

Acknowledgment

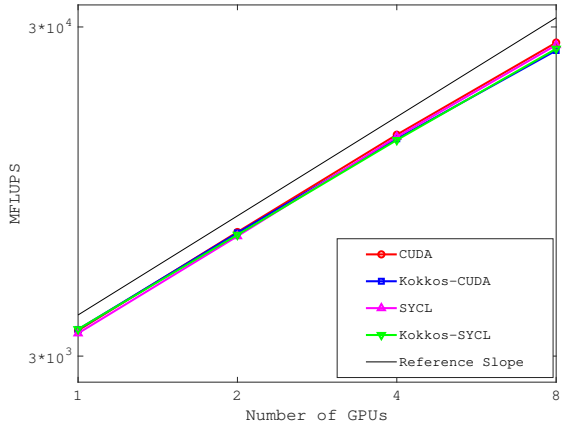
This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. . This research is also a part of the Early Science Program of Argonne (AESP). The authors would also like to thank John Gounley from Oak Ridge Leadership Computing Facility for providing the base code of the proxy app.



(a) Performances for AA propagation pattern.



(b) Performances for AB push propagation pattern.



(c) Performances for AB pull propagation patterns.

Fig. 3: Performance comparison of native CUDA, native SYCL and Kokkos with CUDA/SYCL backend on ThetaGPU.

REFERENCES

[1] X. He and L.-S. Luo, "Theory of the lattice boltzmann method: From the boltzmann equation to the lattice boltzmann equation," *Phys. Rev. E*, vol. 56, pp. 6811–6817, 1997.

[2] Y. H. Qian, D. d'Humieres, and P. Lallemand, "Lattice BGK Models for Navier-Stokes Equation," *Europhys. Lett.*, vol. 17, no. 6, pp. 479–484, 1992.

[3] H. C. Edwards, C. R. Trott, and D. Sunderland, "Kokkos: Enabling manycore performance portability through polymorphic memory access patterns," *J. Parallel Distributed Comput.*, vol. 74, pp. 3202–3216, 2014.

[4] K. Takahashi, W. Watanakeesuntorn, K. Ichikawa, J. Park, R. Takano, J. Haga, G. Sugihara, and G. Pao, "kedm: A performance-portable implementation of empirical dynamic modeling using kokkos," 05 2021.

[5] C. Chang, C.-H. Liu, and C.-A. Lin, "Boundary conditions for lattice boltzmann simulations with complex geometry flows," *Computers & Mathematics with Applications*, vol. 58, no. 5, pp. 940–949, 2009. Mesoscopic Methods in Engineering and Science.

[6] P. Bhatnagar and E. G. M. Krook, "A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems," *Phys. Rev.*, vol. 94, no. 3, pp. 511–525, 1954.