Performance Comparison of Kokkos-Based and CUDA/OpenACC Lattice Boltzmann Solvers

Jipeng Su
The University of Tokyo

Takashi Shimokawabe
The University of Tokyo

Ziheng Yuan
The University of Tokyo

1. Introduction

People see a mixture of CPUs, GPUs, and specialized accelerators, and each vendor provides its own programming ecosystem, such as CUDA, OpenMP, or vendor-specific toolchains. As a result, scientific codes written for one platform often require significant rewriting to run efficiently on another system. This creates a serious sustainability problem for long-lived simulation codes.
This motivates the need for a performance-portable programming model, and Kokkos is designed to satisfy this requirement.

2. Lattice Boltzmann method

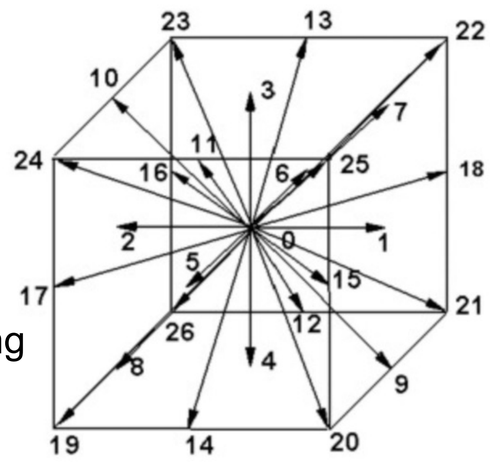
Lattice Boltzmann method is a computational method for simulating the fluid dynamics. Streaming and collision processes from the fluid particles are calculated via distribution functions.

$$f_i(x + e_i\Delta t, t + \Delta t) = f_i(x, t) + \Omega_i(f(x, t))$$

$$\Omega_i(f(x, t)) = -\frac{1}{\tau}(f_i(x, t) - f_i^{eq}(x, t))$$

Each iteration consists of two steps:
• Collision: local relaxation toward equilibrium
• Streaming: propagation to neighboring lattice nodes

From a performance perspective:
• Collision is compute-bound
• Streaming is memory-access intensive



D_3Q_{27} model

3. Kokkos

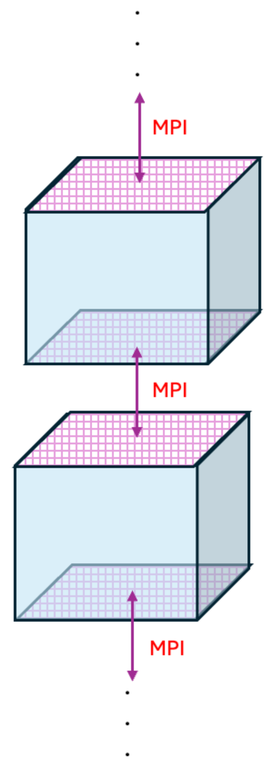
Kokkos aims to achieve performance portability through several key ideas.
First, it supports multiple processors from different vendors. Backends like CUDA, OpenMP or Serial can be enabled for different targets.
Second, it provides an intuitive abstraction for parallelism and data arrangement. A typical array used by Kokkos is called **View**, which can be used to store one or more dimensions.
Third, it allows layout optimization depending on the target hardware.

4. Method

4.1 MPI communications

• In the 3D LBM model:
GPU nodes share the halo layer of x-y plane through MPI communication

• This allows to exchange the boundary information
to update the boundary values in each GPU node

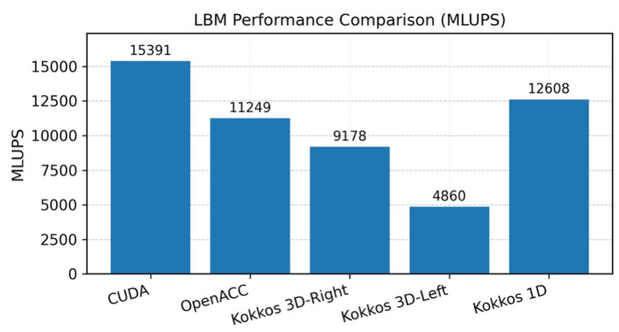


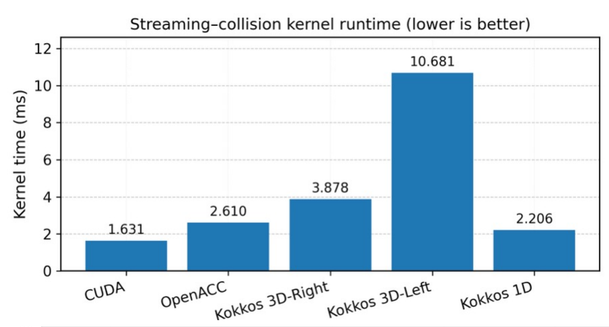
4.2 Kokkos 1D and 3D cases

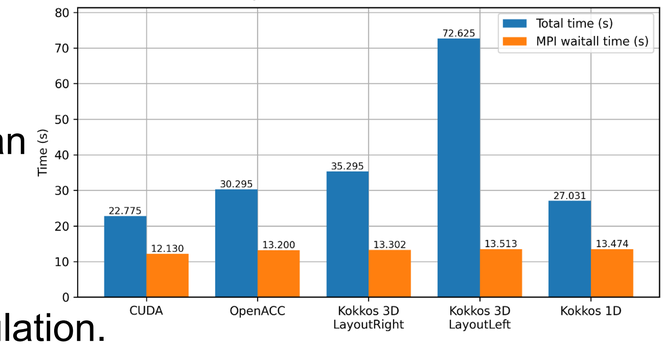
• For Kokkos 1D View case, Kokkos RangePolicy is used, and the index will be computed manually.

• For Kokkos 3D View case, Kokkos MDRangePolicy is used, and it will automatically support the multi-dimensional computation

5. Experiment Result







Generally speaking, Kokkos 1D shows best performance which can catch up with CUDA. But both multi-dimensional kokkos version show drawbacks here on the performance for the 3D LBM simulation.

communication time is similar across all implementations.
This indicates that the performance differences mainly come from the streaming–collision kernel, rather than MPI overhead.
Therefore, kernel-level memory efficiency is the dominant factor.