Accelerating Lattice Boltzmann method with C++ standard language parallel algorithm

Ziheng Yuan
Graduate school of Engineering
The University of Tokyo
Tokyo, Japan
zihengyuan22@g.ecc.u-tokyo.ac.jp

Takashi Shimokawabe
Information Technology Center
The University of Tokyo
Tokyo, Japan
shimokawabe@cc.u-tokyo.ac.jp

GPUs have been widely used to accelerate computation in recent years. From physics simulations to neural networks. Problems with high computation or data density could be solved efficiently by relying on the high parallel computation performance of the GPU. Although GPU can solve problems efficiently, writing parallel programs and running them correctly and efficiently on GPUs remains a challenging issue, especially when the problems are very complex. Typically, parallel programs can be written using a dedicated language, such as CUDA or OpenCL. In this case, since the program needs to prioritize meeting the hardware requirements, the obtained program will not be entirely focused on expressing the actual algorithm. Also, dedicated languages can be limited by hardware constraints, which greatly limits the portability and long-term maintainability of the code.

Since C++17, execution policies have been introduced into the C++ standard library, allowing programmers to efficiently incorporate parallelism into their programs through STL functions that support these policies. The compiler can automatically parallelize the execution of functions specified by the execution policies. Additionally, dynamic memory allocation enables the compiler to obscure the instructions for CPU-GPU communication, allowing programmers to focus more on expressing the algorithm rather than manually managing hardware threads and memory resources, thereby simplifying the complexity of writing parallel programs. Although the actual implementation of C++ standard code relies on other existing parallel languages, such as OpenACC or OpenMP, resulting in slight performance losses, the strong influence of the C++ standard ensures that parallel code written in C++ will be supported by mainstream compilers in the future and can execute on cross-platform, with acceptable performance loss.

The immersed boundary-lattice Boltzmann method (IB-LBM) is used in this experiment to solve the moving boundary flow problem. The IB-LBM combines the Boltzmann method with the immersed boundary method. In this issue, LBM is suitable for parallelization due to its uniform grid distribution and simple kernel algorithm. IBM, on the other hand, can solve boundary problems without changing the grid size, and thus can also be parallelized. In this experiment, the code using IB-LBM to solve 3D propeller simulation was selected as the experimental code.

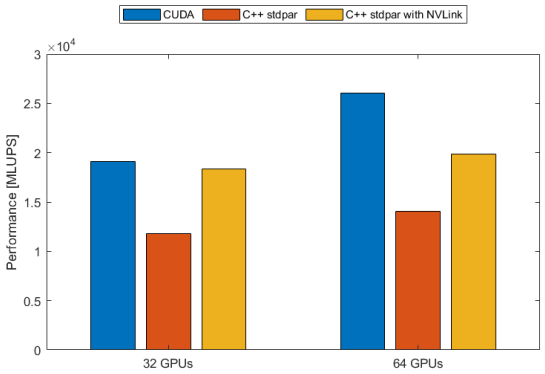

Figure 1: Comparison of CUDA and C++ stdpar Performance

The experiment platform is Wisteria-aquarius supercomputer provided by The University of Tokyo. GPU model is NVIDIA A100. Each computation node contains 8 GPUs. The first issue involves applying standard C++ to the IB-LBM simulation code and running programs on GPUs. Another important issue discussed in this paper is how to use available resources to support C++ standard algorithms for solving large-scale problems. Currently, C++ does not offer a solution to the multi-GPU communication problem. When using C++ standard algorithms for acceleration, it is necessary to supplement some of the functions that C++ cannot realize with the help of existing specialized languages. In this experiment, the LBM and IBM is parallelized by the C++ standard algorithm, and the need for multi-GPU communication is satisfied with the help of CUDA-aware MPI, as well as the need for atomic operations with the help of CUDA. CUDA-aware MPI can enable data to communicate through NVLink, achieving GPU-GPU direct communication. Compared with the direct use of MPI for GPU to GPU communication, the use of NVLink reduce the time consumption of communication by 78.5% in 32 GPUs case and 60.3% in 64 GPUs case. The performance comparation is shown in Figure 1.

REFERENCES

[1] Latt J, Coreixas C, Beny J (2021) Cross-platform programming model for many-core lattice Boltzmann simulations. PLoS ONE 16(4): e0250306. https://doi.org/10.1371/journal.pone.0250306