AMR Framework to Realize Effective High-Resolution Simulations on Multiple GPUs

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

Naoyuki Onodera
Center for Computational Science and e-Systems,
Japan Atomic Energy Agency
Chiba, Japan

Recently grid-based physical simulations with GPU require effective methods to adapt grid resolution to certain sensitive regions of simulations. An adaptive mesh refinement (AMR) method is one of the effective methods to compute certain local regions that demand higher accuracy with higher resolution. To develop the applications adopting AMR effectively with maintaining high performance on multiple GPUs, we are developing a block-based AMR framework for stencil applications [3].

The proposed AMR framework is designed to provide highly-productive programming environment for stencil applications with adapting grid resolution to certain sensitive regions of simulations. The proposed AMR framework is originally based on our existing high-productivity framework for stencil applications on Cartesian grid [1]. By extending this original framework and adding the AMR data structure with halo exchange functions and mesh refinement mechanisms, we construct this AMR framework. This framework is intended to execute the user program on NVIDIA's GPU and x86 CPU, which is implemented in CUDA and C++. The programmer simply describes a C++11 lambda that updates a grid point, which is applied to the entire grids with various resolution over a tree-based AMR data structure effectively.

The framework can locally change the resolution of the grids for arbitrary regions in the time integration loop of applications. The entire computational domain is divided into a large number of small uniform grid blocks. Since each grid block is a uniform grid, the computation on these blocks can be solved with the conventional stencil calculations. This strategy may be effective for performance improvement because GPU can often derive high performance when accessing contiguous memory. By using this framework, the programmer just writes the stencil calculations for a uniform grid with a single resolution. The framework can apply this user-written functions to a large number of grid blocks with various resolutions simultaneously. In addition, the framework provides some C++ classes to realize other processes required for AMR, such as mesh refinement, exchanging data in halo regions between grid blocks with different resolutions, and data migration to maintain load balance.

In halo exchange with multiple GPUs, in order to utilize the simple memory access pattern with avoiding performance degradation, the framework exploits the temporal blocking (TB) method for locality improvement. The multiple time steps can be advanced in each GPU independently of the others without communication in the TB. By using the countdown-based TB proposed in our previous research [2], the framework can apply the TB to the user code without changing the structure of time integration loop.

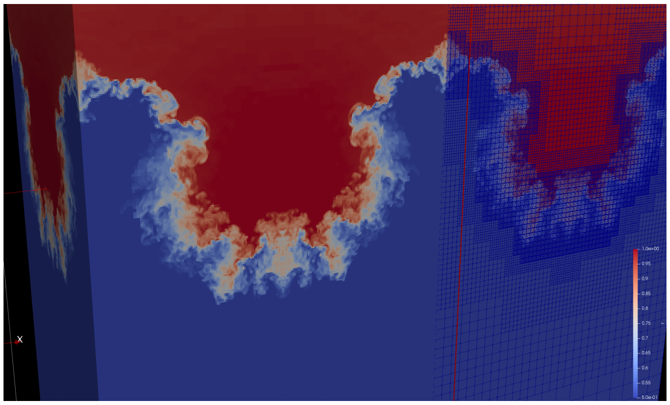

Figure 1: A snapshot of density distribution results obtained by the simulation of 3D compressible flow. The boundary lines of the grid blocks are also shown in part.

Figure 1 shows a snapshot of computational results of the Rayleigh-Taylor instability obtained by 3D compressible flow computation written by this AMR framework. By applying the AMR method to fluid simulation, we have succeeded in simulating with a fine structure around the interface of two fluids. With our proposed framework, we have conducted performance studies of the framework-based compressible flow simulation on a single GPU and using multiple GPUs on TSUBAME 3.0. The framework-based compressible flow simulation has achieved to reduce the computational time to less than 15% with 10% of memory footprint compared to the equivalent computation running on the fine uniform grid. The good weak scaling is obtained using 288 GPUs of TSUBAME 3.0 with the efficiency reaching 84%.

REFERENCES

[1] Takashi Shimokawabe, Takayuki Aoki, and Naoyuki Onodera. 2014. High-Productivity Framework on GPU-Rich Supercomputers for Operational Weather Prediction Code ASUCA. In *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 251–261. https://doi.org/10.1109/SC.2014.26

[2] Takashi Shimokawabe, Toshio Endo, Naoyuki Onodera, and Takayuki Aoki. 2017. A Stencil Framework to Realize Large-Scale Computations Beyond Device Memory Capacity on GPU Supercomputers. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 525–529. https://doi.org/10.1109/CLUSTER.2017.97

[3] Takashi Shimokawabe and Naoyuki Onodera. 2019. A High-Productivity Framework for Adaptive Mesh Refinement on Multiple GPUs. In *Computational Science – ICCS 2019*, João M F Rodrigues, Pedro J S Cardoso, Jânio Monteiro, Roberto Lam, Valeria V Krzhizhanovskaya, Michael H Lees, Jack J Dongarra, and Peter M A Sloot (Eds.). Springer International Publishing, Cham, 281–294. https://doi.org/10.1007/978-3-030-22734-0_21