GPU-based Parallelization of the CUPID code: Two Strategies and Preliminary Results

So Hyun Park*, Ik Kyu Park, Il Jin Kim
Korea Atomic Energy Research Institute, 1045 Daedeok-daro Yuseong-gu Daejeon Korea
**Corresponding author: bibirom10@kaeri.re.kr*

Keywords : CUPID code, GPU-based parallelization, Single-phase turbulence simulation, Computational performance

1. Introduction

Since 2007, the Korea Atomic Energy Research Institute (KAERI) has been developing the CUPID code, a two-phase thermal-hydraulic analysis program, under the national nuclear R&D program. The CUPID code is a versatile solver capable of simulating not only reactor component-scale systems but also computational fluid dynamics (CFD)-scale phenomena. It has been successfully applied to a wide range of problems, such as subchannel-scale reactor core safety analyses and CFD-scale boron mixing simulations within reactor pressure vessels.[1]

Recently, KAERI launched the project to develop a virtual reactor that achieves an experimental-level reliability using high-fidelity simulations. By replacing large-scale experimental campaigns with computational simulations, this project aims to shorten the design cycle and optimize safety margins. The rapid advancement of GPU-based supercomputing technologies has made such large-scale, high-resolution simulations feasible, providing substantial improvements in computational performance.

Furthermore, since the virtual reactor project adopts the PRAGMA code — a GPU-optimized neutron transport solver — it is essential to parallelize the CUPID code on GPUs to enable efficient coupled analyses between the thermal-hydraulic and neutron transport simulations.[2]

In this study, we aim to parallelize the CUPID code for two-phase thermal-hydraulic simulations on GPUs. To achieve efficient and scalable parallelization, we propose two complementary strategies.

Strategy 1. Maintain the existing CUPID data structures and parallelize individual subroutines

Strategy 2: Redesign GPU-optimized data structures and rebuild the solver form the scratch using CUDA C.

2. Strategy 1: Leveraging Existing Data Structures

This strategy parallelizes CUPID while retaining its existing data structures, written primarily in Fortran. The CUPID code adopts a Structure of Arrays (SoA) layout, which is preserved here. GPU parallelization uses a combination of directive-based kernels and manual kernel implementations(Indirective kernel):

Directive kernels method uses the '!$curf kernel do<<<*,*>>> directive, and the existing 'Do loops' are automatically mapped to GPU blocks and threads by the CUDA Fortran compiler. Indirective kernel method explicitly implements custom CUDA kernels to control thread/block mapping and memory access. This parallelization approach offers several advantages: (1) Ease of debugging, (2) Straightforward extension of physical models, (3) Preservation of numerical schemes. However, its disadvantages include limited optimization and reduced code readability.

2.1 Pressure Solver Parallelization

We focused on parallelizing the pressure solver, which accounts for the highest computational cost. CUPID code employs the Incomplete LU-preconditioned Bi-Conjugate Gradient Stabilized (iLU-BiCGStab) algorithm. While vector-vector and vector-matrix operations were parallelized using directive kernels, matrix-vector multiplications were explicitly rewritten with manual kernels to maximize performance.

To further accelerate the iLU preconditioner, we introduces the red-black coloring technique, which decouples data dependencies between neighboring cells. This significantly enhances GPU parallel efficiency.

Benchmarking against a single-core CPU shows a 20x speedup for a mesh with approximately 500,000 cells.

2.2 Governing Equations Parallelization

For most subroutines that compute governing equations, we applied directive kernel method. And it requires CPU-GPU memory transfers.

Although frequent memory copies affect performance, this approach maintains code robustness and compatibility with existing CUPID code.

2.3 Steamtable Parallelization

The steamtable module, responsible for evaluating thermo-physical properties of water, pure steam, and non-condensable gases, plays a critical role in two-phase simulations. Originally, this module performs property evaluation through a single cell-loop with extensive conditional branching(IF-ELSEIF-ELSE), which is inefficient on GPUs. We refactored the steamtable algorithm by separating liquid, steam, NCGs into independent kernels.

2.4 Code Verification

Such GPU-based parallelized pilot code is verified on a 3D k-e turbulence benchmark problems. And it demonstrated excellent agreement with CPU solutions, confirming the correctness of parallel implementation.

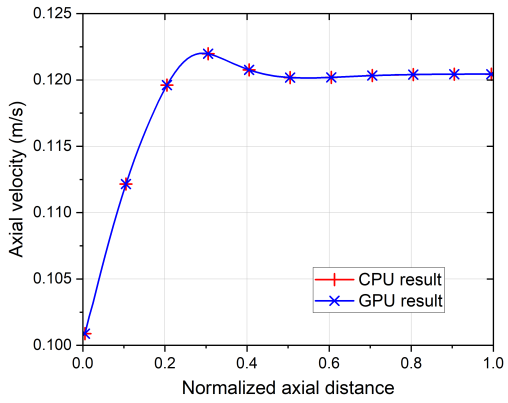

Fig. 1. Comparison of axial velocity with CPU-calculated results

3. Strategy 2: Building GPU-optimized Data Structures

The second strategy builds GPU-optimized data structures and redesigns CUPID from scratch. Unlike Strategy 1, this approach implements manual kernels exclusively.

While the target structure is SoA, the current development stage temporarily uses an Array of Structures (AoS) layout due to code refactoring priorities. The advantages of this strategy include: (1) Reduced memory footprint and GPU-friendly memory access patterns, (2) Extensive use of CUDA libraries(e.g., cuSPARSE, cuBLAS), (3) Greater performance optimization potential. However, since this approach requires a deep understanding on CUPID code and difficult debugging resulting in reconstruction from scratch.

3.1 CUDA Library Utilization for Pressure Solver

We adopted NVIDIA's libraries to parallelize the iLU-BiCGStab solver: Sparse-dense vector operations and sparse matrix-vector multiplications are implemented using cuSPARSE. Vector inner products and vector-vector operations are implemented using cuBLAS. This library-based parallelization achieves high computational efficiency on modern NVIDIA GPUs.

4. Conclusions

We presented two strategies for GPU-based parallelization of the CUPID code to enable high-fidelity two-phase thermal-hydraulic simulations and efficient coupling with the PRAGMA code. Preliminary results lay the groundwork for a virtual reactor framework capable of achieving experimental-level reliability through fully GPU-accelerated simulations.

ACKNOWLEDGEMENT

This research was supported by the National Research Council of Science & Technology(NST) grant by the Korea government (MSIT) (No. GTL24031-100) and the Korea Atomic Energy Research Institute (KAERI) (524520-25).

REFERENCES

[1] Cho, Yun Je, and Han Young Yoon, Numerical Analysis of the ROCOM Boron Dilution Benchmark Experiment Using the CUPID Code, Nuclear Engineering and Design 341, 2019.

[2] N. Choi and H. G. Joo, "Domain decomposition for GPU-Based continuous energy Monte Carlo power reactor calculation," Nuclear Engineering and Technology, vol. 52, pp. 2667-2677, 2020.