Performance Evaluation of the GPU-Parallelized CUPID Code for Large-Scale Reactor Thermal-Hydraulic Simulation

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, Large-scale reactor simulation

1. Introduction

The Korea Atomic Energy Research Institute (KAERI) has been developing the CUPID code for high-fidelity two-phase thermal-hydraulic analysis.[1] To realize the "virtual reactor" concept, which requires experimental-level reliability, accelerating the CUPID code via GPU computing is essential. Previous studies established the foundational framework for GPU-based parallelization and explored two strategies: maintaining existing structures with directive-based parallelization (Strategy 1) and rebuilding the solver from scratch using CUDA C (Strategy 2).

As an intermediate milestone of Strategy 1, a version capable of simulating single-phase turbulent flows was successfully implemented and officially named CUPID-G. This study presents a comprehensive performance evaluation of CUPID-G, in which all governing equations, the steam table, and the pressure solver have been parallelized for GPU execution. Moving beyond preliminary tests, we demonstrate its computational capabilities by applying it to a large-scale i-SMR prototype simulation, focusing on the practical turnaround time and speedup achieved in a GPU computing environment.

2. Methodology

CUPID-G was parallelized using a hybrid approach within the CUDA Fortran environment, combining directive-based kernels with external CUDA kernels for specialized optimizations.[2]

Pressure Solver Accelerating: The pressure solver is the most computationally intensive component of CUPID, primarily due to the iterative nature of the iLU-preconditioned BiCGStab algorithm. To achieve high performance on GPUs, three key enhancements were implemented: (1) Red-Black coloring for parallel preconditioning, [3] (2) padded sparse matrix storage to optimize memory throughput, and (3) a hybrid kernel strategy that utilizes external CUDA kernels for core numerical operations.

Integrated Governing Equations: All governing equations for the two-fluid, three-field model were parallelized using CUDA Fortran directive kernels. To maximize parallel efficiency and minimize the overhead of CPU-GPU memory transfers, the execution flow was restructured to reside primarily on the GPU.

Steam Table Refactoring: The property calculation module, which previously suffered from heavy conditional branching, was refactored into independent kernels for liquid, steam, and non-condensable gases (NCGs). This refactoring strategy improves warp efficiency and resource utilization on the GPU by reducing thread divergence.

3. Performance Evaluation

The performance of CUPID-G was evaluated using a large-scale reactor simulation and compared against a high-performance CPU node (configured with 1 core and 40 cores).

Large-scale Simulation: This study utilizes a high-resolution, 10-million-cell mesh for a quarter-core transient simulation of an i-SMR prototype, representing a real-scale engineering application. The evaluation focused on single-phase turbulent flow within the lower plenum, covering the domain from the pump inlet to the core outlet, including complex internal structures such as the flow skirt and core inlet pipes.

Hardware Specification: GPU computations were performed on a single NVIDIA A100, while CPU computations were conducted on an Intel® Xeon® Gold 6348. The CPU performance was measured in two configurations: a single core (1 CPU) and a full 40-core node (1 node).

3.1. 1 CPU core vs. 1 GPU card

The overall simulation achieved a speedup of approximately 48.7x. While most subroutines exhibited significant acceleration, the iLU

factorization—previously the primary bottleneck—delivered a speedup exceeding 100x. This confirms the successful implementation of the Red-Black coloring method. Excluding the factorization step, the iLU-BiCGStab solver itself achieved a speedup of over 70x, demonstrating the high potential of the proposed GPU strategy for sparse matrix operations.

3.2.40 CPU cores vs. 1 GPU card

The overall speedup was approximately 1.4x, indicating that a single GPU can outperform a fully utilized CPU node. While the performance of the governing equation routines was comparable to the 40-core CPU run, the iLU factorization and the pure iLU-BiCGStab solver achieved speedups of 2.4x and 2.8x, respectively. Since the solver accounts for a substantial fraction of the total runtime in the original CUPID code, these improvements translated into a superior overall performance for the single GPU.

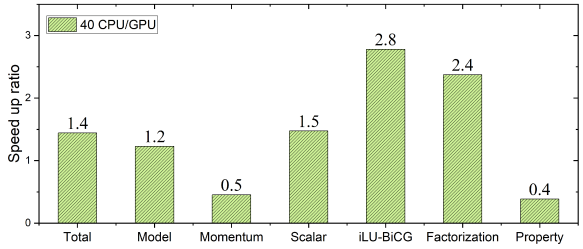

Fig. 1. Comparison of Subroutine-level performance between 40-cores CPU and single GPU computations.

4. Conclusions

This study evaluated the computational performance of CUPID-G against both single-core and 40-core CPU configurations for large-scale reactor thermal-hydraulics. By integrating the governing equations, steam tables refactoring, and an optimized pressure solver into a unified GPU-parallelized framework, we achieved a nearly 50-fold speedup over a single CPU core and a 1.4x improvement over a standard 40-core computing node. The successful execution of a 10-million-cell i-SMR prototype simulation proves that CUPID-G is capable of handling real-scale engineering problems with high efficiency. Future work will extend this framework to multi-GPU configurations and focus on parallelizing full two-phase models with phase change for next-generation virtual reactor development.

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) (524520-26).

REFERENCES

[1] J. J. Jeong, H. Y. Yoon, I. K. Park, and H. K. Cho, “The CUPID Code Development and Assessment Strategy,” Nuclear Engineering and Technology, Vol.42(6), pp.636-655 (2010).
[2] Ruetsch, Gregory, and Massimiliano Fatica. CUDA Fortran for scientists and engineers: best practices for efficient CUDA Fortran programming. Elsevier, 2024.
[3] Cheng, Lei, et al. “GPU-accelerated Krylov subspace methods for fast fluid dynamics simulation of indoor/outdoor airflow.” Building and Environment (2025): 113876.