Recent Advances in the Development of the GPU-Enabled Code STREAM3D-GPU

Siarhei Dzianisau [a], Fathurrahman Setiawan [a], Deokjung Lee [a, b *]
[a]*Department of Nuclear Engineering, Ulsan National Institute of Science and Technology, 50 UNIST-gil, Ulsan, 44919*
[b]*Advanced Nuclear Technology and Services, 406-21 Jonga-ro, Jung-gu, Ulsan, 44429, Republic of Korea*
**Corresponding author: deokjung@unist.ac.kr*

Keywords: neutron transport, CMFD, GPU, OpenACC

1. Introduction

The method of characteristics (MOC) has established itself as a powerful approach for solving neutron transport equations, balancing high accuracy with computational efficiency. MOC-based codes such as nTRACER [1], OpenMOC [2], and MPACT [3] have emerged as viable alternatives to Monte Carlo methods, offering deterministic solutions while significantly reducing computational costs. However, large-scale reactor simulations using MOC remain demanding, particularly in full-core 3D modeling. With the increasing adoption of Graphics Processing Unit (GPU) acceleration in scientific computing, MOC solvers have become prime candidates for performance optimization. Notably, nTRACER successfully integrated GPU acceleration into its 2D/1D MOC solver and extended it to depletion calculations [4][5], demonstrating substantial speedups. These advancements have underscored the potential of GPU-based transport solvers and motivated further development in this area.

Expanding on this progress, we focus on GPU acceleration for our in-house code, STREAM, which employs a 2D/3D Diamond-Difference MOC solver [6]. This method enhances accuracy and stability compared to 2D/1D MOC approaches while optimizing memory and computational resource usage. Other attempts at GPU acceleration for similar solvers exist [7], but they often lack a direct and fair comparison with CPU implementations, making it difficult to quantify the true impact of GPU offloading. In this work, we report on the development status of the GPU-enabled version of STREAM3D. Compared to our previous work [8], we incorporate a GPU-accelerated Coarse Mesh Finite Difference (CMFD) solver to further enhance computational performance. By systematically comparing CPU and GPU implementations, we provide a comprehensive assessment of speedup and accuracy. Our study presents steady-state and depletion results, focusing on key reactor parameters and pin-wise power distributions.

Building on previous developments in GPU acceleration, we aim to address several key challenges in STREAM3D-GPU. First, we ensure that our GPU implementation maintains numerical accuracy and consistency with the reference CPU-based STREAM3D solver. Second, we optimize the performance of both transport and CMFD solvers to maximize the computational benefits of GPU acceleration. Finally, we present a detailed evaluation of our approach through benchmark problems, comparing results against traditional CPU-based methods and assessing the overall impact on computational efficiency.

The remainder of this paper is as follows: Section 2 describes the methodology and implementation details of the GPU-accelerated CMFD solver, presents the results of steady-state simulations using a large pressurized water reactor depletion problem, and summarizes the findings in a discussion. Section 3 provides conclusion remarks and topics for future work.

2. Methods and Results

2.1 GPU-enabled CMFD in STREAM3D-GPU

Previous work on developing a GPU-enabled MOC solver has been reported earlier [8]. In this study, we introduce a GPU-enabled CMFD module. CMFD is used as the primary acceleration technology in STREAM and has been proven to be very efficient at improving the convergence speed of the main MOC solver [6]. CMFD consists of two levels: assembly-wise CMFD, and pin-wise CMFD. For both levels, neutron energy group and space condensation are required, which is one of the primary consumers in the overall process. One of the key changes in the GPU-enabled CMFD is the introduction of Jacobi iteration for the linear system solver. In STREAM3D, the CMFD solver employs Gauss-Seidel iterative methods with a Red-Black planar scheme for parallelization, which does not allow sufficient GPU parallelization.

Before starting the CMFD offloading to GPU, we observed that it was not accelerated much in other GPU-enabled codes [5]. We associate this with a few possible factors: small problem size, which may amplify the time required to launch GPU kernels and update memory; reliance on double precision (DP), which could penalize GPU-enabled codes due to an insufficient number of DP cores in modern consumer-grade or even workstation-level GPUs. Regardless of the exact reasons, we attempted to mitigate possible challenges for porting our CMFD code to GPU by following a newly developed scheme shown in Fig. 1. To ensure the maximal utilization of CUDA cores, we converted all CMFD-related data to single precision.

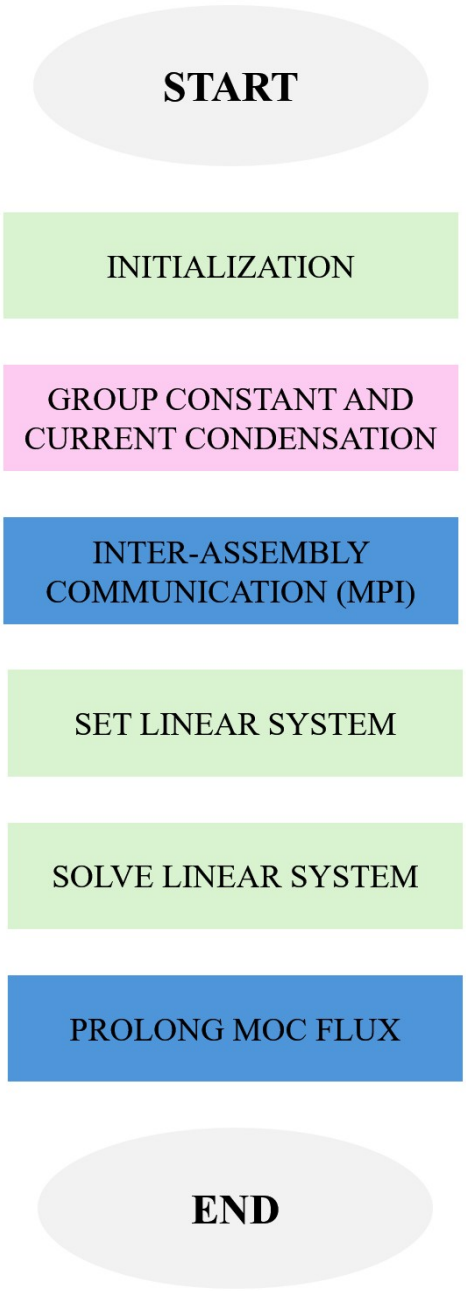

Fig. 1. GPU-enabled CMFD implementation scheme.

In Fig. 1, green boxes are fully executed in GPU, whereas blue boxes utilize CPU. The pink box uses both CPU and GPU code.

2.2 Depletion Results

For a depletion simulation, we used an OPR-1000 octant-core problem [9] with thermal-hydraulic feedback, and we modeled 30 burnup steps for both code cases. The first case is our previous version of STREAM3D-GPU, in which the MOC solver was offloaded to GPU [8]. In that version, the CMFD solver was executed using CPU only, and was based on the CPU-optimized scheme. In the current study, we call it the reference since it was verified against a CPU version. The second case is the updated STREAM3D-GPU code, with the CMFD improvements presented in this study. For brevity, we call it the candidate code in this study. The speedup that was achieved in the candidate version compared to the reference version is available in Fig. 2, and the comparison of core-wise pin powers for both versions is shown in Fig. 3-5.

We used the same single node for both versions, meaning the hardware and the software were identical. The node comprises 64 CPU cores, 8 NVIDIA RTX A5000 24 GB cards, and 2 TB of DDR-4 memory. We opt for a single-node calculation since it is the most consumer-appealing option. Using multiple nodes often implies investing in a much larger, power—and space-demanding, and eventually costly system. In both cases, GPUs communicated through the host side MPI due to the large size of the angular flux arrays that were to be synchronized between different MPI domains. OpenMPI in NVIDIA HPC SDK supports direct communication between GPUs via CUDA-aware MPI, but it was not used in STREAM3D-GPU.

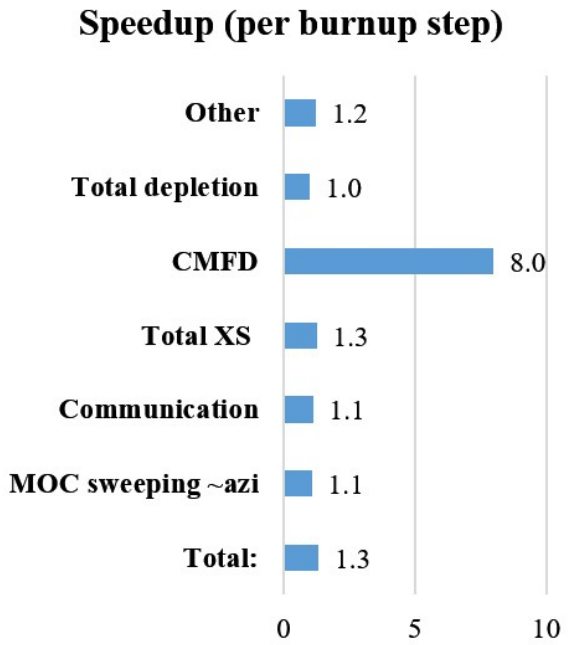

Fig. 2. 3D OPR-1000 octant-core, average speedup between the reference and the candidate STREAM3D-GPU code per burnup step.

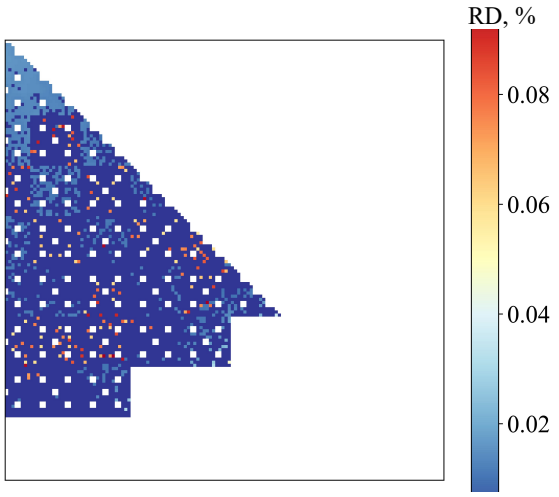

Fig. 3. 3D OPR-1000 octant-core, 0.0 GWD/MTU, relative difference (RD, %) for pin power between the reference and the candidate STREAM3D-GPU code.

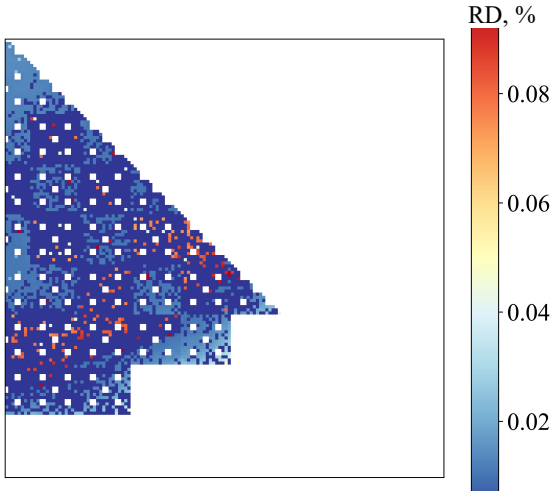

Fig. 4. 3D OPR-1000 octant-core, 1.5 GWD/MTU, relative difference (RD, %) for pin power between the reference and the candidate STREAM3D-GPU code.

As shown in Fig. 2, the newly updated version of STREAM3D-GPU demonstrates a reasonable speedup compared to the previous code version. Accuracy-wise, STREAM3D-GPU does not display unusual deviations from the reference, which could be observed in Fig. 3-5. The k_{eff} and pin powers stayed within the expected margin for a drastically modified code executed on GPU. For all burnup steps, the difference in k_{eff} did not exceed 10 pcm, and the difference in each pin power did not exceed 0.09%. Such deviations are considered normal considering the changes introduced. When the code is converted to GPU, the order of operations changes, which affects the values of processed and accumulated data, especially as the CMFD module was converted to single precision.

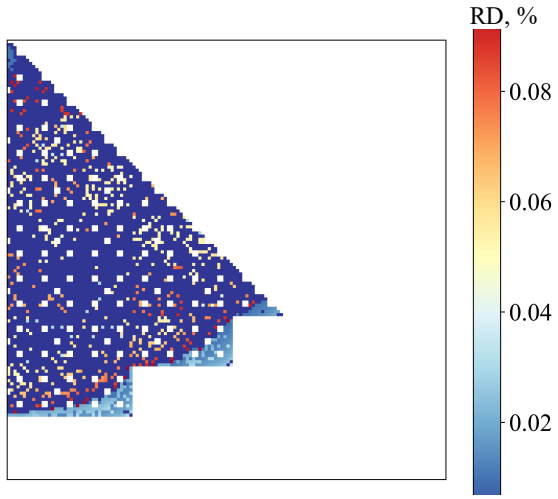

Fig. 5. 3D OPR-1000 octant-core, 1.5 GWD/MTU, relative difference (RD, %) for pin burnup between the reference and the candidate STREAM3D-GPU code.

2.3 Discussion

One important observation could be made upon observing the Fig. 2 result. Despite using the same code in the modules other than the CMFD module, most of them consumed less time, while the depletion time stayed the same. The reason for that is the newly developed GPU-enabled CMFD module. CMFD is regarded as an efficient acceleration technique for the MOC solver, which means it can reduce the number of MOC iterations before reaching convergence. Since we modified the CMFD solution scheme, we got a minor improvement of convergence for some of observed problems. This results in having less MOC iterations per burnup step, and, consequently, a reduction of the time for the involved modules by 10-20%. Since the number of depletion steps did not change, the depletion module consumed the same amount of time.

The average execution time per burnup step for the tested OPR-1000 octant-core was just 20 minutes while using one GPU node with 8 GPU cards. While it is expected that adding more computing nodes would reduce this time even further, there are compiler-related challenges to overcome due to nvfortran being noticeably slower for the CPU parts of the code than, for example, gfortran. As a result, while the GPU code is designed to be linearly scalable, the CPU routines could display worse improvements in performance. Further work is required to determine the compiler-specific CPU bottlenecks.

The effect of developing the GPU-enabled CMFD code was expected to be less than it was achieved in this study. We attribute the positive changes with several factors: converting the data to single precision, redesigning the CMFD module to make it more

optimized for GPU (changing the Gauss-Seidel scheme to Jacobi scheme).

3. Conclusions

This study presents an updated version of our GPU-enabled deterministic neutron transport code STREAM3D-GPU. Compared to previous versions, a GPU version of the CMFD solver was developed and incorporated into the code. The testing results demonstrate a noticeable speedup of the solver compared to a 64-core CPU version, whereas the accuracy of the code was not compromised and stayed on par with the original code. A single node calculation for OPR-1000 octant-core depletion problem was observed to take only 20 minutes per burnup step.

Further work on STREAM3D-GPU includes porting the cross-section processing routines and depletion routines to GPU.

Acknowledgment

This work was supported by the Innovative Small Modular Reactor Development Agency grant funded by the Korea Government (MOTIE) (No.RS-2024-00407975).

REFERENCES

[1] Y. S. Jung, C. B. Shim, C. H. Lim, H. G. Joo, Practical Numerical Reactor Employing Direct Whole Core Neutron Transport and Subchannel Thermal/Hydraulic Solvers, Annals of Nuclear Energy, Vol. 62, Pages 357-374, 2013.

[2] W. Boyd, S. Shaner, et al., The OpenMOC Method of Characteristics Neutral Particle Transport Code, Annals of Nuclear Energy, Vol. 68, Pages 43-52, 2014.

[3] B. Collins, S. Stimpson, et al., Stability and Accuracy of 3D Neutron Transport Simulations using the 2D/1D Method in MPACT, Journal of Computational Physics, Vol. 326, Pages 612-628, 2016.

[4] N. Choi, J. Kang, H. G. Lee, H. G. Joo, Practical Acceleration of Direct Whole-Core Calculation Employing Graphics Processing Units, Progress in Nuclear Energy,Vol. 133, 103631, 2021.

[5] H. G. Lee, K. M. Kim, H. G. Joo, Development of Scalable GPU-Based Direct Whole-Core Depletion Calculation Methods, Progress in Nuclear Energy, Vol. 165, 104928, 2023.

[6] S. Choi, D. Lee, Three-Dimensional Method of Characteristics/Diamond-Difference Transport Analysis Method in STREAM for Whole-core Neutron Transport Calculation, Computer Physics Communications, Vol. 260, 107332, 2021.

[7] A. Zhang, M. Dai, et.al., Development of A GPU-Based Three-Dimensional Neutron Transport Code, Annals of Nuclear Energy, Vol. 174, 109156, 2022.

[8] S. Dzianisau, D. Lee, "GPU Acceleration of 3D MOC Solver in STREAM3D Using OpenACC," KNS Spring Meeting, Jeju, South Korea, May 9-10, 2024.

[9] H. Kim, A. Cherezov, et. al., Multi-Cycle Analysis of OPR1000 Using Multi-Physics Coupled Codes of RAST-K, CTF and FRAPCON, Proceedings of American Nuclear Society Mathematics & Computation 2019, Oregon USA, August 25-29, 2019.