GPU Acceleration of 3D MOC Solver in STREAM3D Using OpenACC

Siarhei Dzianisau[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: MOC, GPU, OpenACC, CUDA.

1. Introduction

Method of characteristics (MOC) has become a prominent way of finding a deterministic solution for a neutron transport equation. Codes that utilize MOC in some form, such as nTRACER [1], OpenMOC [2], MPACT [3], and others, offer a competitive alternative to high-fidelity Monte-Carlo codes while consuming less memory and, often, less computational resources.

Given the recent advancements in graphics card (GPU) acceleration of time-consuming and resource-demanding applications in various fields of science and engineering, MOC codes became a desired candidate for a similar performance boost. Notable success was achieved by the nTRACER developers who ported the 2D/1D MOC solver to GPU [4], followed by porting other time-consuming modules such as the depletion module [5]. These results pushed others, including us, to develop efficient algorithms for porting our codes to GPU.

In our case, we are interested in enabling GPU acceleration for our in-house code STREAM [6]. This code, unlike the previously mentioned references, is based on a 2D/3D Diamond-Difference MOC solver. The advantages of this solver include higher accuracy and stability compared to 2D/1D methods, while saving substantial amounts of system memory and computing resources. Unsurprisingly, other researchers already attempted to develop a GPU-enabled code utilizing our in-house 2D/3D MOC method [7]. However, this work lacks information on the actual speedup offered by adding a GPU into the system. This is because a CPU version of the code needs to be developed for such a comparison, which is not done in this and some other cases of publicly presented GPU-enabled codes.

In this work, we are aiming to share our approach in offloading the most time-consuming part of our code STREAM to GPU using an OpenACC framework. Similar to [4] and [5], we have a CPU-optimized version of the code already implemented, which means we can demonstrate the actual comparison between the CPU and the GPU versions that utilize the same methodology, thus making it a fair comparison in terms of showing the GPU speedup. The methodology of our approach is briefly described in section 2, while the test problem results are covered through sections 3 and 4. Lastly, the conclusion remarks are presented in section 5.

2. Methodology

The first published result of our work on the GPU enabling of our code STREAM [8] was based on the same code structure as the CPU version, thus offering identical results. However, that approach not only was impractical due to the lack of GPU-specific optimizations, but also was problematic in terms of problem scalability. The largest problem presented there was the 3D C5G7 problem, which only required 8 Gb of system memory, thus making it easy to fit into a limited GPU memory. However, the real-world problems could consume hundreds of Gb of system memory, thus making them more challenging to be properly offloaded to GPU.

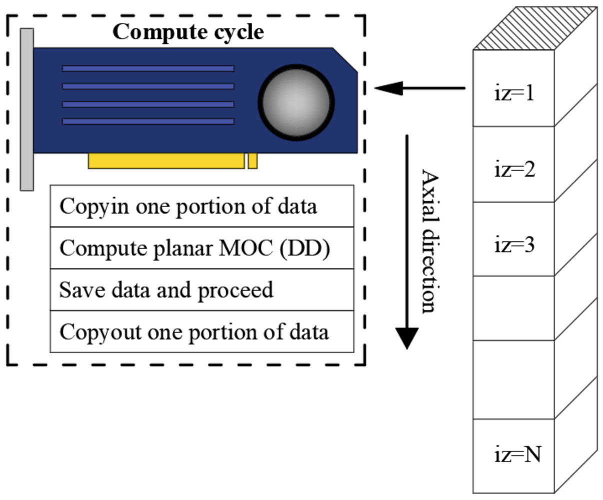


Fig. 1. Axial decomposition strategy for offloading the 2D/3D MOC solver to GPU.

In our current version, we have introduced a novel decomposition approach that efficiently reduces the occupied GPU memory for any reactor geometry size. The approach is schematically shown in Fig. 1 and is based on the original code structure inherited from [6] and [8], while utilizing the sequential nature of the axial loop. As a result, for each plane, a portion of minimally required data is transferred from the host side (also known as the CPU side) to the device side (known as the GPU side) before the planar sweep. After the sweep is finished, some data stays on the device, while the axially dependent data is copied back to the host for storage and future use. The essential data to be copied in and copied out every step includes the incoming and outgoing angular fluxes, and neutron current for CMFD acceleration.

3. Description of test problems

To determine the performance boost offered by our GPU-enabled MOC solver, two problems were chosen. The first problem is the 3D C5G7 benchmark problem in quarter-core symmetry [8][9]. It uses only 7 neutron energy groups and does not have any multi-physics feedback. The second problem is a conventional OPR-1000 3D model [10] simulated in octant-core format to save the system memory. The chosen model was assembled using gadolinia (Gd) fuel assemblies (FA) shown in stripe colors in Fig. 2, and non-Gd FAs shown in solid colors. Unlike the C5G7 problem, the OPR-1000 model was simulated with thermal-hydraulic feedback and using all 72 neutron energy groups.

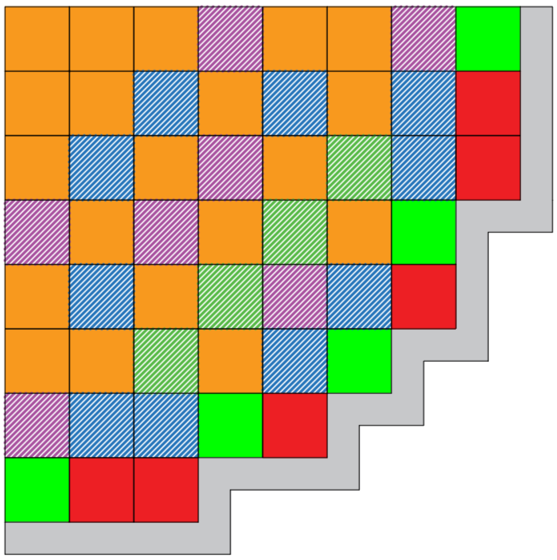


Fig. 2. Loading pattern of the modeled reactor in a quarter-core format (solid colors – UO_2 pins only; stripe colors – UO_2 pins and Gd pins; grey frame – radial reflector).

An important difference between the chosen models and other GPU-enabled code models such as [5] and [7] is the axial mesh size. In STREAM, the axial mesh size is 3 cm, resulting in 197 axial planes for the OPR-1000 problem, and 64 axial planes for C5G7 problem, which is higher than in the mentioned publications. This choice adds extra fidelity to the code, while the downside is the higher system memory requirement.

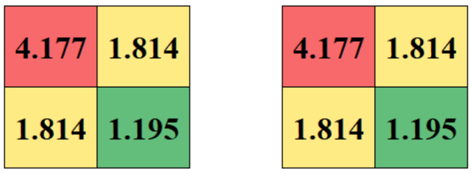
4. Results and discussion

The presented results were obtained using the following configurations. For the CPU reference, a 64-core system was used (2 AMD EPYC 7543 32-core CPUs). For the GPU result, 8 NVIDIA RTX A5000 were used for the MOC solver, while the remaining parts of the code were accelerated using the same 64 CPU cores.

First, the 3D C5G7 benchmark execution time for the CPU reference and the GPU-enabled code is provided in Table 1. The 2D FA powers shown in Fig. 3 are non-distinguishable, while the effective multiplication factor (k_{eff}) is identical for both versions.

Table 1. C5G7 performance

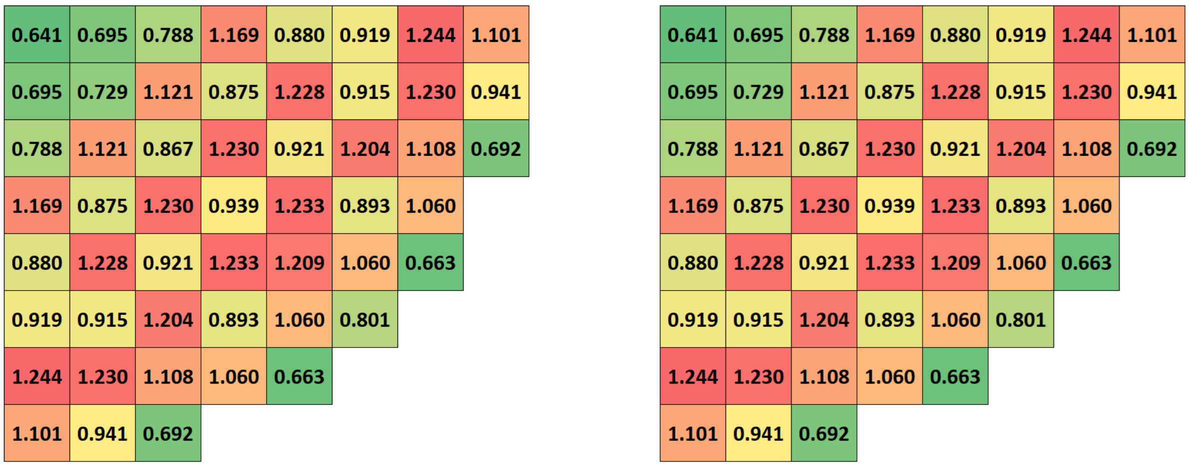
Metric	CPU	GPU	Speedup
MOC Solver, sec	1,368	34	40.23
Total execution time, sec	1,704	100	17.04



4.177	1.814
1.814	1.195

4.177	1.814
1.814	1.195

Fig. 3. C5G7 2D FA power distribution for the CPU reference (left) and the GPU-enabled code (right).



0.641	0.695	0.788	1.169	0.880	0.919	1.244	1.101
0.695	0.729	1.121	0.875	1.228	0.915	1.230	0.941
0.788	1.121	0.867	1.230	0.921	1.204	1.108	0.692
1.169	0.875	1.230	0.939	1.233	0.893	1.060	
0.880	1.228	0.921	1.233	1.209	1.060	0.663	
0.919	0.915	1.204	0.893	1.060	0.801		
1.244	1.230	1.108	1.060	0.663			
1.101	0.941	0.692					

0.641	0.695	0.788	1.169	0.880	0.919	1.244	1.101
0.695	0.729	1.121	0.875	1.228	0.915	1.230	0.941
0.788	1.121	0.867	1.230	0.921	1.204	1.108	0.692
1.169	0.875	1.230	0.939	1.233	0.893	1.060	
0.880	1.228	0.921	1.233	1.209	1.060	0.663	
0.919	0.915	1.204	0.893	1.060	0.801		
1.244	1.230	1.108	1.060	0.663			
1.101	0.941	0.692					

Fig. 4. OPR-1000 2D FA power distribution for the CPU reference (left) and the GPU-enabled code (right).

For the OPR-1000 model, the corresponding results are presented in Table 2 and Fig. 4. Similar to the C5G7 problem, the k_{eff} value of the GPU version exactly matched the CPU reference result.

Table 2. OPR-1000 octant-core performance

Metric	CPU	GPU	Speedup
MOC Solver, sec	61,535	3,995	15.40
Total execution time, sec	212,416	11,201	18.96

All in all, the presented results show noticeable improvement of the C5G7 model calculation time and indicate the memory efficiency of the newly developed GPU code. The OPR-1000 octant-core problem requires around 1 Tb of system memory for storing all the necessary data, which could impose issues for GPU code development due to comparably tiny memory capacity of consumer-grade GPUs (in our case – 24 Gb per card). However, the data splitting method allowed our code to reliably fit inside the GPU memory regardless of the core geometry size.

The reason for drastic reduction of the total execution time in Table 2 is the communication time between radially decomposed regions. In the GPU code, each MPI process is assigned a separate GPU. In the CPU code, there could be very few OpenMP-enabled CPU cores per MPI process. This could create a load imbalance between different radial domains and increase the waiting times for exchanging the incoming and outgoing flux information. The solution for the CPU side is to employ more CPU cores, which is typically done. However, this is beyond the scope of the presented study because our goal is to compare a single node performance using the same hardware resources, compiler, and working environment.

5. Conclusions

In this study, fresh results on the development of GPU-enabled version of 2D/3D MOC code STREAM are presented. To overcome the challenges of fitting a large commercial PWR problem into a small GPU on-board memory, a new method of axial decomposition was introduced and underwent testing. Two common problems were used for the testing purposes, one is the C5G7 problem, and the other is an OPR-1000 octant-core problem that is using all 72 energy groups with multi-physics feedback.

Testing was performed using a single computing node consisting of 64 CPU cores and 8 GPU cards. Offloading the 2D/3D MOC solver of STREAM onto GPUs showed more than 15 times speedup compared to the CPU result. Overall, the total execution time of the code was also drastically reduced, partially due to using only a single computing node for evaluation. Notably, the results of calculations stayed identical between the CPU and the GPU versions due to using the same methodology and directly comparing the CPU version of the code to the GPU version of the code. This offers important insights on the actual speedup that could be achieved by enabling the GPU acceleration in an already existing CPU-optimized code.

Future plans for the GPU-enabled version of STREAM include offloading other computationally expensive modules to GPU and comparing the codes using multiple CPU and GPU nodes.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019M2D2A1A03058371).

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, M. F. Khandaq, T. Q. Tran, D. Lee, On the GPU Acceleration of a 3D MOC Code STREAM Using OpenACC, Proceedings of American Nuclear Society Mathematics & Computation 2023, Niagara Falls, Ontario, Canada, August 13–17, 2023.
[9] E. E. Lewis, M. A. Smith, N. Tsoulfanidis, G. Palmiotti, T. A. Taiwo and R. N. Blomquist, Benchmark Specification for Deterministic 2-D/3-D MOX fuel assembly Transport Calculations without Spatial Homogenization (C5G7MOX), NEA/NSC/DOC, 2001.
[10] 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.