Multi-GPU Multi-Node Implementation of CUDA-aware MPI in GREAPMC

Muhammad Rizwan Ali, Murat Serdar Aygul, Deokjung Lee*
Ulsan National Institute of Science and Technology, UNIST-gil 50, Eonyang-eup, Ulju-gun, Ulsan, 689-798, Korea
**Corresponding author: deokjung@unist.ac.kr*

***Keywords** : GREAPMC, CUDA-aware MPI, Monte Carlo, GPU*

1. Introduction

Due to the stochastic nature of the Monte Carlo (MC) method, a large number of particles must be simulated to achieve accurate results. Furthermore, the inclusion of depletion and multi-physics calculations imposes additional demands on memory and execution time. To address the computational intensity of the MC method, it is essential to incorporate parallel programming capabilities into the MC code. Two widely used parallel programming models for this purpose are the Message Passing Interface (MPI) and OpenMP. MPI is a distributed memory model that facilitates communication between processes running on multiple nodes, making it well-suited for high-performance computing (HPC) applications across clusters. On the other hand, OpenMP is a shared-memory model that simplifies parallel programming within a single multi-core processor or node by enabling threads to access shared memory in a controlled manner. While MPI excels in scalability and flexibility for distributed systems, OpenMP provides ease of use and efficiency for parallelizing tasks on a single machine.

In the context of GPU programming, computations are executed on the GPU, making MPI more suitable than OpenMP for enabling multi-GPU implementations. To meet the goal of reducing simulation time, multi-node, multi-GPU capabilities have been integrated into GREAPMC (**G**pu-optimized **REA**ctor **P**hysics **M**onte **C**arlo), an in-house code developed at UNIST [1]. This paper outlines the methodology employed to enable parallel processing in GREAPMC.

2. Methodology

Multi-GPU capability can be implemented in three distinct ways.

1. Regular MPI
2. CUDA-aware MPI
3. NVIDIA Collective Communications Library (NCCL)

In this work, CUDA-aware MPI has been integrated, with plans to incorporate NCCL in the near future. The distinction between regular MPI and CUDA-aware MPI is thoroughly discussed in [2], but a brief overview is provided here. In regular MPI, MPI functions are limited to accepting pointers to host (CPU) memory. When working in an MPI+CUDA environment, this necessitates staging GPU buffers through host memory, which requires additional `cudaMemcpy` operations. This extra memory copying can potentially hinder performance in multi-GPU setups. In contrast, CUDA-aware MPI allows GPU buffers to be passed directly to MPI functions. The data flow in CUDA-aware MPI is shown in Fig. 1. Pinned memory scheme is used to stage data between GPU and network buffers. Pinned (page-locked) host memory enables efficient PCIe and RDMA transfers by allowing direct memory access, which is essential for high-throughput GPU communication in CUDA-aware MPI.

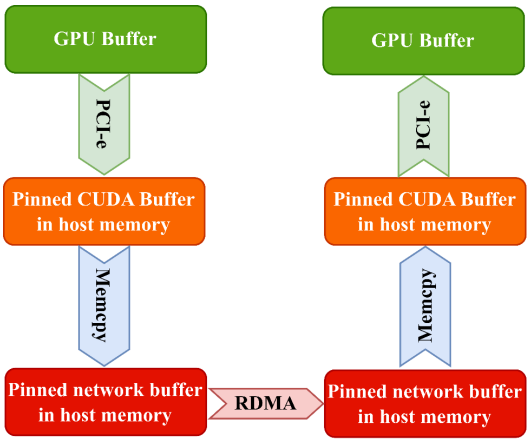

Fig. 1. Flow of data in CUDA-aware MPI.

The geometrical and material data are copied to each MPI rank. The particle data is uniformly distributed among the MPI ranks after assigning a unique seed to each particle. Following each cycle, the ranks must communicate to synchronize fission sites, tally statistics, and accumulate k_{eff} estimators. To ensure efficient MPI implementation in GREAPMC, minimizing communication overhead is critical. Therefore, the Nearest Neighbors Algorithm developed by Romano et al. [3] has been integrated to support multi-GPU capability in GREAPMC. In this algorithm, each rank communicates only with the adjacent ranks. Each rank accumulates the fission sites stochastically which are sampled after the cycle end and redistributed to make the number of source sites equal to the initial number of histories. This results in a uniform distribution of source sites with minimal communication overhead. For GPU kernel execution, each CUDA block was configured with 256 threads. This value was empirically selected based on performance benchmarking to optimize occupancy and memory throughput on GPU side.

3. Results and Discussion

This section presents the MPI speed-up attained by GREAPMC for the OPR-1000 core [4]. Since MPI implementation is only directed towards the acceleration of the code using multiple nodes, the effective multiplication, flux tally, and fission reaction rates essentially remain conserved. GREAPMC uses double precision for critical calculations such as delta-tracking (DTS) and tally accumulation to maintain numerical accuracy. Therefore, no degradation in calculation accuracy was observed due to the use of GPU computation. Core lab at UNIST has a GPU cluster with three nodes and each node has eight GPUs making the total number of GPUs equal to twenty-four. The specifications of the nodes are given in Table I. The node-01 (N1) is a heterogeneous node employing two RTX 3090 with a higher clock speed, more CUDA cores, and a higher SM (Streaming Multiprocessor) count than RTX A5000.

Table I: Node Specifications.

Node	Specification
N1	2 × NVIDIA GeForce RTX 3090 + 6 × NVIDIA RTX A5000
N2 & N3	8 × NVIDIA RTX A5000

The performance, in the form of MPI speedup, of CUDA-aware MPI is computed using the Eq. (1).

$$\text{MPI Speedup} = \frac{\text{Time for } n \text{ GPUs}}{\text{Time for one GPU}} \times 100 \quad (1)$$

This section details the performance evaluation beginning with single-node results. The single-node simulations presented here are conducted using Node 1 (the mixed configuration) and Node 2 (eight RTX A5000s). Following the single-node analysis, multi-node, multi-GPU results are presented to assess the code's scalability as the number of simulated particles per cycle increases. This multi-node analysis examines how effectively the code utilizes the combined resources of multiple nodes and GPUs to handle larger simulations.

3.4 Single-Node, Multi-GPU

Table II presents the MPI speedup comparison using a single node (node 2) with multiple GPUs for two scenarios: ten million particles per cycle and fourteen million particles per cycle. These simulations were performed using the OPR-1000 problem.Fig. 2 illustrates these trends, where the ideal case has a slope of one. The scalability of simulations with more particles per cycle exhibit closer proximity to ideal scaling behavior as the MPI rank count is augmented, in contrast to simulations with fewer particles per cycle.

Table II: MPI Speedup on node 2.

Number of GPUs	MPI Speedup	
	10 million	14 million
1	1.00	1.00
2	1.98	1.98
3	2.94	2.96
4	3.89	3.92
5	4.81	4.87
6	5.74	5.81
7	6.62	6.75
8	7.50	7.64

As shown in the table, increasing the number of particles per cycle improves the MPI speedup. Fig 2. illustrates these trends, where the ideal case has a slope of one. The scalability of simulations with a greater number of particles per cycle exhibits closer proximity to ideal scaling behavior as the MPI rank count is augmented, in contrast to simulations with a lower number of particles per cycle.

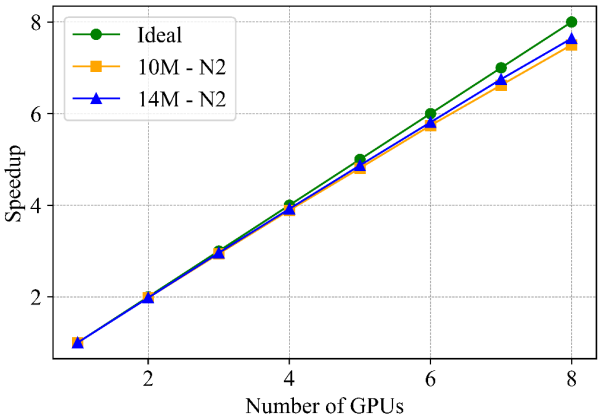

Fig. 2. Speedup against the number of GPUs.

Table III outlines the MPI speedup for the same simulations on Node 1. A comparison of Table III with Table II reveals superior performance from node 1. The performance enhancement can be attributed to the presence of two high performant RTX 3090 GPU cards on node 1.

Table III: MPI Speedup on node 1.

Number of GPUs	MPI Speedup	
	10 million	14 million
1	1.00	1.00
2	1.97	2.00
3	2.95	2.99
4	3.89	3.95
5	4.87	4.96
6	5.73	5.88
7	6.73	6.85
8	7.55	7.82

Fig 3. illustrates the achieved MPI speedup. The MPI speedup gap between ten million particles per cycle and

fourteen million particles per cycle has increased here compared to simulations on node 2.

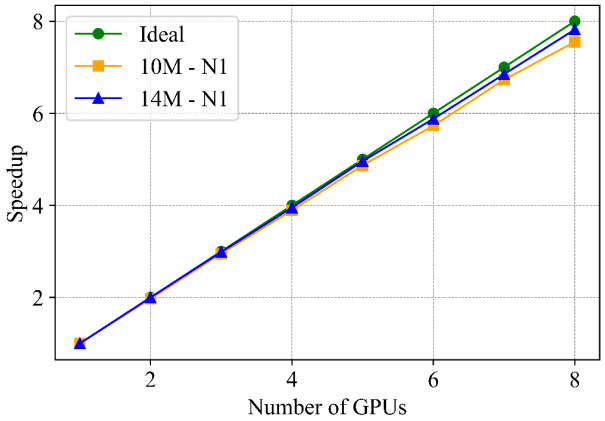

Fig. 3. MPI speedup for simulations on node 1 of the GPU cluster.

3.5 Multi-Node, Multi-GPU

Fig.4 illustrates the MPI speedup achieved when scaling the simulation from 1 GPU to 24 GPUs across 3 nodes (24 GPU cards). The dashed line represents ideal linear scaling, meaning the speedup would equal the number of GPUs if there were no overhead. Both the 10M and 14M particle curves closely follow a linear trend, though they fall slightly below the ideal due to communication overhead and other inefficiencies. Notably, the 14M curve remains nearer to the ideal line as the GPU count increases, indicating that a larger number of particles per cycle helps to better offset the communication costs. In simple terms, higher particle counts lead to more efficient use of the GPUs, yielding performance that more closely approximates ideal scaling.

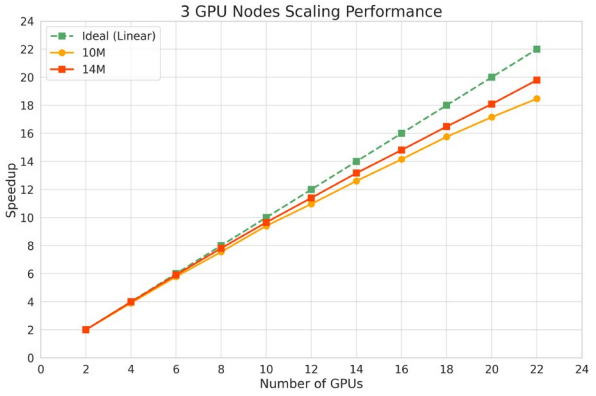

Fig. 4. The plot of extended scaling performance in 3 GPU nodes (total of 24 GPUs)

3.6 Multi-Node, Multi-GPU

In this configuration, the OPR1000 core was run using all three nodes on the GPU cluster (containing 24 GPUs). The particles per cycle varied from 24 million to 240 million. The tracking rates are computed and plotted using the active cycle's average execution time. Table IV gives tracking rates in million neutrons per second using all 24 GPUs.

Table IV: Tracking rate in multi-node, multi-GPU setting.

Particle per cycle (million)	Tracking rate (Mn/s)
24	15.635
30	15.775
36	15.987
42	16.156
50	16.32
70	16.487
90	16.592
100	16.623
120	16.631
140	16.662
160	16.698
180	16.705
200	16.745
240	16.923

From Fig. 5, GPUs demonstrate superior performance when processing a larger number of particles. For instance, in the 24 million particle simulation, each GPU handles one million particles, while in the 240 million particle simulation, each GPU processes ten million particles. Further increasing the particle count is expected to enhance the tracking rate further, as evidenced by the upward trend in the plot beyond the 200 million particle mark. This observation aligns with the previously discussed improved parallel efficiency for higher particle counts per cycle, collectively indicating excellent scalability for the number of particles.

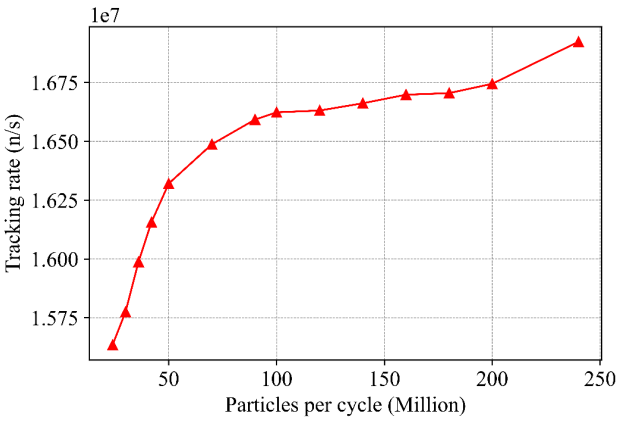

Fig. 5. The plot of tracking rate as the number of particles per cycle increases.

4. Conclusions

This work demonstrates the successful implementation of CUDA-aware MPI in GREAPMC, enabling efficient multi-node, multi-GPU capabilities to accelerate neutron transport calculations. By leveraging CUDA-aware MPI, the code minimizes data transfer overhead and simplifies GPU buffer management, achieving near-ideal scaling behavior as the number of simulated particles per cycle increases. Multi-node scalability tests underscored the code's ability to

efficiently utilize 24 GPUs across three nodes, with tracking rates improving significantly for larger particle counts. The observed trends highlight the importance of balancing computational load and communication overhead, particularly when scaling to distributed systems. Future work will focus on adding depletion capability to GREAPMC.

ACKNOWLEDGEMENTS

This research has been supported by project L20S089000, funded by Korea Hydro & Nuclear Power Co. Ltd.

REFERENCES

[1] M. R. Ali, M. S. Aygul, and D. Lee, "Enhancing PWR Monte Carlo Simulations with GREAPMC: A GPU-Accelerated Approach," in *International Conference on Physics of Reactors (PHYSOR 2024)*, San Francisco, CA, USA, pp. 2174–2183. doi: doi.org/10.13182/PHYSOR24-43582.

[2] J. Kraus, "An Introduction to CUDA-Aware MPI." Accessed: Oct. 26, 2024. [Online]. Available: https://developer.nvidia.com/blog/introduction-cuda-aware-mpi/

[3] P. K. Romano and B. Forget, "Parallel Fission Bank Algorithms in Monte Carlo Criticality Calculations," *Nuclear Science and Engineering*, vol. 170, no. 2, pp. 125–135, Feb. 2012, doi: 10.13182/NSE10-98.

[4] S. Choi and D. Lee, "Preliminary Results for OPR1000/APR1400 Whole-core Analysis with Neutron Transport Code STREAM," presented at the M&C, Oregon, USA, Aug. 2019.