ANT-MOC: Scalable Neutral Particle Transport Using 3D Method of Characteristics on Multi-GPU Systems

Shunde Li*
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
lishunde@cnic.cn

Zongguo Wang*
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
wangzg@cnic.cn

Lingkun Bu*
National Center for Materials Service Safety, University of Science and Technology Beijing
Beijing, China
blk1997@126.com

Jue Wang†
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
wangjue@sccas.cn

Zhikuang Xin
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
xinzhikuang@cnic.cn

Shigang Li
School of Computer Science, Beijing University of Posts and Telecommunications
Beijing, China
shigangli.cs@gmail.com

Yangang Wang
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
wangyg@cnic.cn

Yangde Feng
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
ydfeng@sccas.cn

Peng Shi
National Center for Materials Service Safety, University of Science and Technology Beijing
Beijing, China
pshi@ustb.edu.cn

Yun Hu
China Institute of Atomic Energy
Beijing, China
evanhy@163.com

Xuebin Chi
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
chi@cnic.cn

ABSTRACT

The Method Of Characteristic (MOC) to solve the Neutron Transport Equation (NTE) is the core of full-core simulation for reactors. High resolution is enabled by discretizing the NTE through massive tracks to traverse the 3D reactor geometry. However, the 3D full-core simulation is prohibitively expensive because of the high memory consumption and the severe load imbalance. To deal with these challenges, we develop ANT-MOC[1]. Specifically, we build a performance model for memory footprint, computation and communication, based on which a track management strategy is proposed to overcome the resolution bottlenecks caused by limited GPU memory. Furthermore, we implement a novel multi-level load mapping strategy to ensure load balancing among nodes, GPUs, and CUs. ANT-MOC enables a 3D full-core reactor simulation with 100 billion tracks on 16,000 GPUs, with 70.69% and 89.38% parallel efficiency for strong scalability and weak scalability, respectively.

CCS CONCEPTS

• **Computing methodologies** → **Massively parallel algorithms**;
• **Applied computing** → *Physics*.

KEYWORDS

Neutron particle transport, 3D method of characteristic, Load balancing, Multi-GPUs

*Both authors contributed equally to this research.
†Corresponding author.

[1]The name "ANT-MOC" is inspired by the cooperative transport behavior of ants, which allows them to efficiently exploit resources from their environment.

SC '23, November 12–17, 2023, Denver, CO, USA

ACM ISBN 979-8-4007-0109-2/23/11.
https://doi.org/10.1145/3581784.3607063

ACM Reference Format:
Shunde Li, Zongguo Wang, Lingkun Bu, Jue Wang, Zhikuang Xin, Shigang Li, Yangang Wang, Yangde Feng, Peng Shi, Yun Hu, and Xuebin Chi. 2023. ANT-MOC: Scalable Neutral Particle Transport Using 3D Method of Characteristics on Multi-GPU Systems. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC '23), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3581784.3607063

1 INTRODUCTION

The virtual reactor offers a safer and more cost-effective means for designing and maintaining nuclear reactors. Various research institutions and government agencies have initiated several projects to promote the development of virtual reactors, including VERA [31], NEAMS [4], NURE [5], and CVR [36]. The Neutron Transport Equations (NTE) serves as an important method to calculate various aspects of the reactor core, such as criticality, power distribution and fission rate. Within virtual reactors, the intractable problem is obtaining an exact NTE solution. The Method Of Characteristic (MOC) [1] is an efficient method to solve the NTE because of its geometric solid adaptability and high fidelity.

The MOC solver for the NTE is based on massive characteristic tracks that traverse the geometry of the reactor. Along with tracks, the angular of NTE is discretized into liner tracks [32]. All of the tracks are discretized into segments by reactor structural details such as fuel assemblies, control rods, and other structures. As the reactor core structure becomes more complex, the simulation scale in geometry has evolved from single-assembly to full-core [10]. For more detail and higher accuracy in the radial direction provided by 3D tracks, the simulation scale in tracks has also evolved from 2D to 3D [16] [9]. Recently, 3D full-core simulation has become a long-standing goal of the reactor physics community.

Extending the simulation for 3D full-core is challenging due to the large-scale computation and memory required by tracks and segments. Gunow [12] propose a spatial decomposition method to map large amounts of computation and memory to CPU nodes. The computing capabilities of CPUs remain inadequate for conducting 3D full-core simulations. With the development of heterogeneous computing, more supercomputers adopt accelerators such as GPU for large-scale computing [30]. However, the memory size of GPU often becomes a bottleneck that limits program expansion [37]. In our test case consisting of 100 billion tracks, the memory required for 3D segments is 511.5 times greater than 2D segments case, reaching as high as 132.6 TB. Sciannandrone et al. [26] suggest an alternative approach for CPU solver using chord classification method for axially extruded geometries that only saves segments in 2D. This method allows us to implement the on-the-fly generation of 3D segments in GPU. In addition, the reactor's multi-assembly structure provides significant geometric heterogeneity. State-of-the-art solution methods [33] [3] use fine meshes in the reflective regions and coarse meshes in the central core region to improve accuracy and reduce computing costs. Nonetheless, these approaches lead to an uneven distribution of computational burden among tasks segregated by spatial partitioning. Therefore, two important challenges of 3D full reactor simulation need to be addressed: On the one hand, limited computing power and memory cannot meet the computation requirements of high-resolution simulation. On the other hand, the load imbalance introduced by the decomposition and mapping method severely limits the full performance potential of computers.

To address the above challenges, we develop ANT-MOC, a software with 40,000 lines of code, which solves the 3D full-core NTE using the MOC on GPU clusters. The following contributions are provided:

- We port the track generation, ray tracing, and source computation to the GPU, which enables ANT-MOC to perform end-to-end neutron transport simulations on GPU clusters. A performance model is built to predict the memory usage, computation, and communication traffic of ANT-MOC.
- We propose a track management strategy based on the performance model to alleviate the capacity bottleneck of GPU memory. The regenerated track is minimized by the ratio of temporary tracks and resident tracks.
- Based on the performance model, we propose a three-level load mapping strategy on GPU clusters. Computing resources are fully utilized by mapping tasks of different granularities among nodes, GPUs, and Compute Units (CUs), which refers to SMs.
- We are the first to achieve 100 billion tracks in a full-core 3D pressurized water reactor simulation. The efficiencies of strong scalability and weak scalability reach 70.69% and 89.38% on 16,000 GPUs, respectively.

The rest of this paper is organized as follows: Section 2 introduces MOC solver and compares this work with the current state-of-the-art work. Section 3 presents the whole execution process of ANT-MOC and introduces the GPU kernel and performance model implementation. Section 4 introduces a track management strategy and a three-level load mapping strategy. Section 5 verifies the correctness, performance improvement, and scalability of ANT-MOC.

2 BACKGROUND

2.1 MOC Solver and Spatial Decomposition

MOC is widely used to solve the Boltzmann form of NTE in various neutron transport solution software [2][13][6] [14][18][38]. In ANT-MOC, the S_N method[22] is used to enforce the NTE to be solved for a set of selected directions to which a set of quadrature weights are associated [29]. Then, the spatial discretization and step approximation are applied to geometry to obtain Flat Source Regions (FSR) where the material is homogeneous. Eventually, we obtain the angular flux corresponding to a single track in Equation (1)

$$\psi_{i,m}(x) = e^{-\Sigma x}\psi_{i,m}(0) + E_i Q_i \tag{1}$$

where $\psi_{i,m}(0)$ and $\psi_{i,m}(x)$ are the flux at the entering position and the leaving position of the track, $\sum$ is the cross section for this region, Q_i is the flat source, and E_i is the escape probability for the source. Equation (1) is used to iteratively calculate flux along a track from the starting segment to the ending one.

As shown in Figure 1, the entire flowchart of neutron transport simulation can be divided into four primary procedures: geometry construction, track generation, ray tracing, and transport solving.

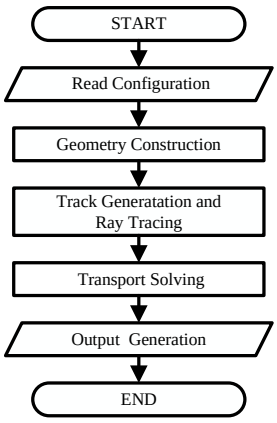

Figure 1: The execution flow of typical MOC programs. Transport solve performs a transport sweep to iterate over all tracks, which is the most time-consuming part of the workflow.

In the geometry construction phase, the geometry Constructive Solid Geometry (CSG) method [2] [24] is used to construct the 3D geometry. The CSG method is an intuitive and convenient modeling method widely used in mainstream reactor simulation programs. Then the geometry is divided into FSR by the SPH homogenization method[35]. The geometry construction phase requires less computation and memory footprint.

In the track generation and ray tracing phase, tracks and segments are generated based on the geometric structure [27]. ANT-MOC can make use of the On-The-Fly (OTF) method [11] or the Chord Classification Method (CCM) [26] for axial track generation. After laying tracks on the axial surface, a group of 3D tracks distributed in the Z-axis space is expanded. 3D Tracks are divided into segments by FSR. The amount of computation is related to the number of track segments due to the iteration scheme of MOC. All 3D tracks are stored along with additional parameters on radial sections and could be restored during transport solving, which enables us to adopt a track management strategy on 2D tracks with the OTF method.

In the transport solving phase, the FSR's flux energy distribution is first calculated by transmission scanning and source updating. Each transmission scanning integrates the flux of each energy group along each orbit from the previous iteration. At the same time, each FSR's fission source and absorption are updated by counting the new flux contribution of tracks. Fission source and absorption are used to calculate K_{eff}. In reactor physics, K_{eff} refers to the effective neutron multiplication factor of the core. It is one of the key physical parameters for evaluating nuclear reactors. These steps are repeated iteratively until the source converges in each region.

There are many techniques for decomposing the workload to multi-domain[19][41]. Spatial decomposition [21] is the most straightforward and efficient decomposition technique, which has been extensively used in various MOC programs [28] [13] [20] [25]. It requires only near-neighbor communications for angular flux along truncated tracks. Accordingly, dividing geometry into spatial subdomains keeps a large granularity and low overhead. The implementation of spatial decomposition is strongly related to the laydown of tracks since the tracks are supposed to be connected at spatial subdomain boundaries. The solution is affected by the number and size of spatial subdomains. A subdomain only updates its incoming angular flux at the end of a source computation, which is similar to a Point Jacobi iteration [9]. There might be differences between the fission rates produced with and without the spatial decomposition, while the normalized fission rates are usually the same.

2.2 Current State of the Art and Challenges

Over the years, several neutron transport-solving software have emerged after undergoing numerous developments and improvements. They all employ MOC to solve neutron transport problems. However, most programs do not directly employ 3D MOC because of the memory and computational capabilities limitations. These programs more commonly use 2D/1D coupled characteristic line calculations or solely use MOC to calculate the homogenized components of the subassembly.

We give an overview of some typical MOC software. These software involve of DeCART [13], NECP-X [6], nTRACEE [14], MPACT [18], ThorMOC [38], OpenMOC [2], and ANT-MOC. Cho [7] originally proposes a 2D/1D method [40] for solving the 3D NTE. The method couples radial 2D equations [39] with axial 1D equations. Due to its significant reduction in computational complexity compared to direct 3D solving, the method has been widely adopted. However, in the 2D/1D method, transverse leakage may result in a negative total source and computational instability. In contrast, the 3D method can effectively handle axial leakage problems. The 3D MOC/DD method [38] is based on axial linear approximations of angular flux and source. It decomposes the 3D equation into a series of axial 2D equations coupled by surface angular flux.

Table 1: Comparison of typical MOC software

Name	Solver type	Decomposition method	Hardware support
DeCART	2D+1D	None	CPU
NECP-X	2D+1D	track	CPU
MPACT	2D+1D	None	CPU
nTRACER	2D+1D	None	single-GPU
OpenMOC-2D	2D	Spatial	single-GPU
OpenMOC-3D	3D	Spatial	CPU
ThorMOC	3D MOC/DD	Angle	Single-GPU
ANT-MOC	3D	Spatial+Angle+Track	Multi-GPU

In Table 1, OpenMOC stands out as the most functional software. It is equipped with a 2D GPUsolver and can perform 3D computation on CPU multi-node. The goal of many neutron transport-solving software developments is to achieve an end-to-end, high-resolution, three-dimensional neutron transport solution for direct full-core simulation using GPUs.There are many practical obstacles to realizing this goal:

(1) The amount of computation required for a direct 3D neutron transport solution is approximately 1000 times greater than

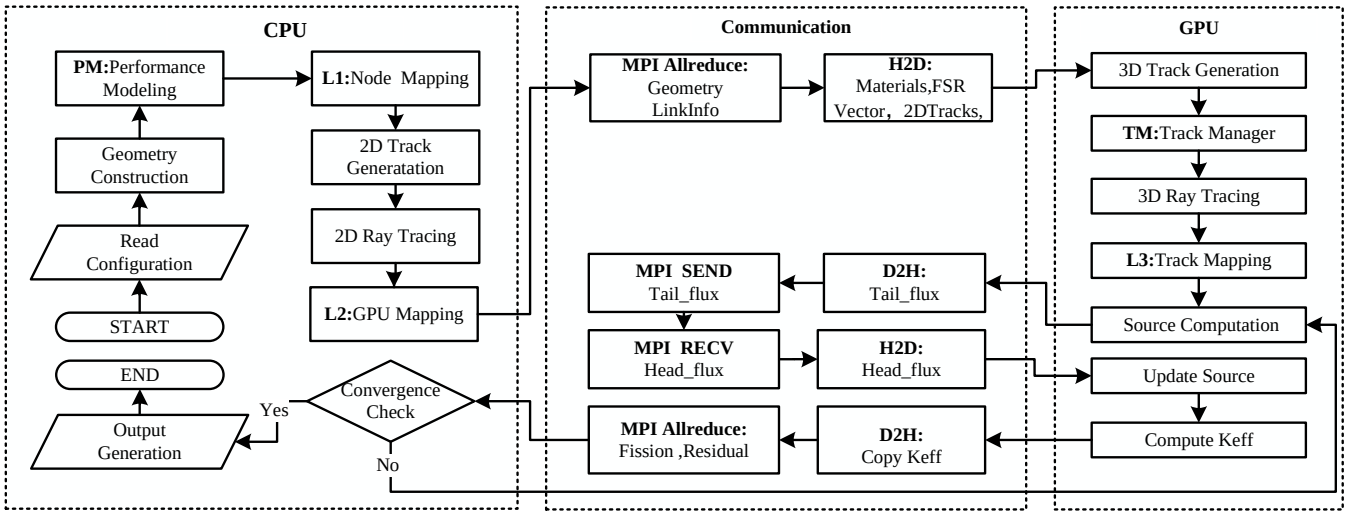

Figure 2: The execution flow of parallel version of ANT-MOC

that of 2D solution [19]. Furthermore, the increase in the scale of tracks requires a faster-than-linear increase in the computational count.

(2) High-resolution neutron transport simulation places higher demands on memory footprint, and the limited memory space exacerbates this issue. It is a significant problem to balance efficient heterogeneous computing with memory footprint.

(3) The division of a simulation task may result in load imbalance. This is often caused by the heterogeneous nature of reactor structures [34] and the parallel spatial decomposition method [8] of the software. As a result, such load imbalance can extend across the different levels of supercomputer hardware, including nodes, GPUs, and CUs, thereby creating significant obstacles for efficient and scalable programs.

To the best of our knowledge, ANT-MOC is the first direct 3D neutron transport software on GPU cluster. ANT-MOC can significantly accelerate neutron transport simulations compared to CPU-based codes. A strategy for managing tracks is proposed to alleviate the bottleneck of memory footprint. Furthermore, a three-level load mapping strategy partitions geometry, angles, and tracks into nodes, GPUs, and CUs, respectively. These methodologies effectively boost hardware utilization and improve parallel scalability.

3 IMPLEMENTATION

This section gives the overview of ANT-MOC and its parallel implementation in detail. Furthermore, a performance model has been established to predict memory, computation, and communication efficiency use of ANT-MOC.

3.1 Overview of ANT-MOC

ANT-MOC is equipped with a GPU solver and supports cross-node calculation via MPI. The execution flow of parallel version of ANT-MOC is shown in Figure 2. For the convenience of description, we divide the execution flow into five stages corresponding to Figure 1.

(1) Read Configuration: ANT-MOC reads a configuration file containing spatial decomposition parameters and track-generation-related parameters.

(2) Geometry Construction: The geometry is divided into several small sub-geometries. After division, the performance model covered in Section 3.3 can estimate memory, computation, and communication. Several sub-geometries are mapped to different nodes using the strategy in Section 4.2.1.

(3) Track Generation and ray tracing: Each node generates 2D tracks for its assigned sub-geometry and maps the tracks to the GPU based on the policy in section 4.2.2. Then, the connection information between geometries is synchronized through MPI. Materials, FSR vector, and 2D track data are then copied to GPU memory. The GPU first carries out 3D track generation, after which the track manager covered in Section 4.1 divides them into two categories: resident tracks and temporary tracks. Then, 3D ray tracing is conducted on the resident tracks. The ray tracing of temporary tracks is performed within the Source Computation process.

(4) Transport solving: The Track Mapping is carried out using the policy outlined in section 4.2.3. Once the source computation is completed, the tail fluxes of tracks are transmitted through the adjacent domains of MPI between different nodes and receive flux as new head flux and update source and K_{eff}. Iteration continues until the flux residuals value is below a certain threshold.

(5) Output Generation: If the Iteration converges and the transport is solved, ANT-MOC will output the FSR fission rate data to a file.

3.2 Parallel Implementation

ANT-MOC divides tasks into different granularity. Spatial decomposition is carried out based on geometric structure which is evenly divided into multiple cuboid sub-geometries arranged in 3D space.

The decomposed sub-geometry only exchanges flux with neighbors. In order to achieve the load balance mapping covered in 4.2.2, it is necessary to map multiple sub-geometries to a single GPU. We implement a geometry fusion method that merges multiple geometries into a fusion-geometry. To be specific, the modular ray tracing method[19] ensures that each sub-geometry has identical dimensions and distribution of tracks. Nevertheless, distinct sub-geometries contain discrete information, such as neighboring geometries, distribution of FSRs and distribution of materials. To address this issue, an additional dimension is added to store information on subdomains that are fused into one fusion-geometry. Upon completion of the kernel of the transport solution, track fluxes are transferred between GPUs via DMA within the same node. Subsequently, the track flux is transferred to adjacent fusion-geometry in other nodes.

We port the transport solver to the GPU. Relevant codes are implemented through CUDA and integrated with the ROCM platform-suitable code snippets. We can use hipify-perl automatic code conversion scripts for easy ROCM platform code conversion. Therefore, the GPU solver can support both NVIDIA and AMD hardware devices. On the right side of Figure 2, three algorithms of 3D track generation, 3D ray tracing, and source computation are the most time-consuming and account for 70% of the computational workload. Considering the number of tracks is large enough to allow large-scale parallelism, we map 3D tracks to GPU threads to perform these algorithms.

Algorithm 1: 3D track generation on GPU

Input: d_track_3D, num_3D_tracks
Output: 3D tracks, segments, or flux
1 Calculate thread ID: tid = threadIdx.x + blockIdx.x * blockDim.x;
2 **while** $tid <$ `num_3D_tracks` **do**
3 Execute track processing workflow;
4 $tid \leftarrow tid + blockDim.x * gridDim.x$;

Algorithm 1 presents a high-level summary of how the 3D tracks are operated on a GPU. The loop hierarchies of the three algorithms are identical to one another, as they all require a sweep of 3D tracks. The algorithm takes two inputs: d_track_3D and num_3D_tracks. The algorithm outputs consist of any 3D tracks generated by track generation, segments calculated by ray tracing, or flux computed by source computation. In line 1, each thread calculates its unique thread ID and processes a specific 3D track. In line 2, the while loop ensures that the available threads process all 3D tracks. The track processing workflow is executed for each track. The thread ID is then updated. The three algorithms differ in function: 3D track generation aims to get 3D tracks through 2D tracks; 3D ray tracing aims to generate 3D track segments; source computation aims to calculate flux along each track.

3.2.1 3D Track Generation. The generation of 3D tracks is based on 2D tracks, and a set of 3D tracks are stacked on top of a 2D track. The mapping relationships between 2D tracks and 3D tracks are pre-computed on the CPU and copied to GPU memory. The 3D tracks are traversed using Algorithm 1, and the processing flow for each track is illustrated in Figure 3(a). In a given stack of 3D tracks, all tracks share the same position, including the azimuthal angle and x-y plane of the corresponding 2D track. A 3D track index is proposed by leveraging both 2D track chain and 2D track stack indexes, which allows us direct access to linking tracks. Finally, the linking information and indexing information of the 3D tracks is stored in GPU memory.

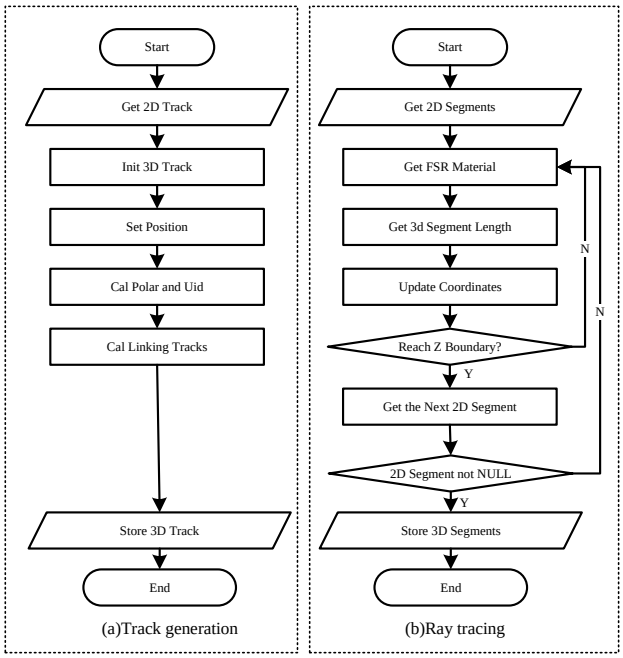

Figure 3: Flow chart of 3D track generation and ray tracing

3.2.2 3D Ray Tracing. 3D ray tracing is executed to trace along 2D base tracks and get 3D segments. The processing flow of each 3D track is shown in Figure 3(b). The generation process of 3D Segments involves two levels of loops. The outer loop iterates through all the required 2D segments, while the inner loop traces each 2D segment. Within the inner loop, the FSR-related information for the starting coordinate is obtained first, and then the length of the 3D Segment is calculated and the coordinate is updated. At the same time, it is checked whether the coordinate crosses the Z-axis boundary. If so, the loop is exited; otherwise, the above steps are repeated based on the new coordinate.

3.2.3 Source Computation. The transport solver traverses all energy groups. We first calculate the flux for each energy group by traversing a specific track in the forward direction. The scalar flux is calculated and accumulated in each segment into the corresponding FSR. The remaining flux is saved for transmission to adjacent regions. We then transfer the boundary angle fluxes to the outgoing track. Following the above procedures, the segments in the reverse direction are performed by the same process. The one-to-many relationship between FSR and tracks results in multiple linear read/write scalar fluxes in parallel transport solving. We put the FSR-associated fluxes into shared memory and use atomic

operations for accessing. By appropriately organizing the order of variables to align data, we can save memory access time and memory consumption.

3.3 Performance Model

Table 2: Performance related parameter

Comments	Shorthand
The number of azimuth angles	N_{num}
The spacing of azimuth angles	S_{azim}
The number of polar angles	P_{num}
The spacing of polar angles	S_{polar}
The number of 2d tracks	N_{2D}
The number of 2d segments	N_{2Dseg}
The number of 3d tracks	N_{3D}
The number of 3d segments	N_{3Dseg}
The number of FSRs	N_{FSR}

During the simulation of reactors by using ANT-MOC, the performance of memory footprint, computation, and communication is usually affected by the initial input parameters. Therefore, it is necessary to identify the relationships between performance and input parameters. The input parameters and their derivations are listed in Table 2. The first four parameters are the initial inputs, and others are derived from inputs. We present the relationships for these parameters in Equation (2), (3), and (4), and we further explore how these parameters impose their influences on memory, computation, and communication performance in Equation (5), (6), and (7).

$$N_{2D} = \sum_{a=1}^{\frac{N_{num}}{2}} f(a) \tag{2}$$

The number of 2d tracks N_{2D} is calculated by Equation (2) where N_{num} is the angle number of vector tracks, and the angle number of scalar tracks is $N_{num}/2$. The function f adopts the track-laying algorithm, which returns the number of tracks passing through the x-y plane at specific angles.

$$N_{3D} = \sum_{i=1}^{N_{2D}} \sum_{p=1}^{P_{num}} g(a, i, p) \tag{3}$$

Each 2D track expands into a set of 3D tracks. Each 3D track must be divided into P_{num} distinct polar angles. The number of 3D tracks is calculated by Equation (3), where g is a function to calculate the number of 3D tracks with the same polar angle and 2D base track. If the FSRs are determined, a linear relationship between the number of segments and tracks is presented in Equation (4).

$$\begin{aligned} N_{2Dseg} &= \frac{B_{2Dseg}}{B_{2D}} * N_{2D} \\ N_{3Dseg} &= \frac{B_{3Dseg}}{B_{3D}} * N_{3D} \end{aligned} \tag{4}$$

Equation (4) is used to estimate the number of segments. The coefficients are derived from a small test case. Where B_{2Dseg} and B_{2D} respectively denote the number of segments and tracks in a small sample. When the scale of rays is relatively dense, the number of segments increases proportionally with the number of tracks. Therefore, we can use a small sample to predict the segment count for a larger sample. The same principle is applied in the 3D case as well. When the geometry is constructed, the number of FSRs is determined by the subdivision of the components, and it remains fixed throughout the simulation.

Table 3: Percentage of memory footprint for main variables

Item	Percent
2D_tracks	0.02%
3D_tracks	0.71%
2D_segments	3.41%
3D_segments	93.31%
Track_fluxs	1.85%
Others	0.69%
All	100%

We have examined the memory footprint of the main vectors during the ANT-MOC calculation, and the results are displayed in Table 3. 3D and 2D segments occupy is up to 97% of the total memory, and this proportion increases with an increase in the number of tracks. The maximum memory footprint during the simulation can be calculated by Equation (5). All variables in Table 3 are vectors, and each item of them is constructed by a structure type. S is the mapping function from the size of a vector to its memory occupation. F is the memory occupation of the constants or fixed-size vectors we considered in our code.

$$\begin{aligned} memory = F &+ S(N_{2D}) + S(N_{3D}) \\ &+ S(N_{2Dseg}) + S(N_{3Dseg}) + S(N_{FSR}) \end{aligned} \tag{5}$$

According to the 3D ray tracing and source computation algorithm, we can infer that the number of segments determines the computation workload of the solver. The computation workload is corrected linearly with the number of segments in the source iterative. The computation workload can be evaluated by Equation (6).

$$computation \overset{\propto}{\sim} N_{3Dseg} \tag{6}$$

The processes use Buffered Synchronous algorithm[20] to synchronize the 3D track flux with their neighbors. Each 3D track maintains the flux of two directions of tracks and multiple energy groups. Single precision is used for flux memory. The communication traffic can be calculated by Equation (7).

$$communication = N_{3D} * 2 * num\_group * 4 \tag{7}$$

where the constant 2 represents the two directions of flux, num_group represents the number of energy groups, and the constant 4 represents the memory width of a float in bytes. The unit of communication is bytes.

The performance model can make ANT-MOC more efficient. For example, the communication and computation model can be used to design a load-mapping strategy and a track management strategy. The memory model can give reasonable memory estimation and avoid memory overflow. In Section 4.1 and 4.2, we apply the performance model to develop the strategy of track management and load mapping of ANT-MOC.

4 OPTIMIZATION

We have proposed two optimization strategies to achieve scalable 3D neutron transport solving. One is the track management strategy to trade off memory and computation, and the other is a three-level load mapping strategy suitable for supercomputer to reduce idle time.

4.1 Track management strategy

As the memory requirement of 3D segments is tens to hundreds of times that of 2D segments, the vast memory requirement is prohibitive[19]. It poses a more significant challenge to the limited memory of the GPU. The OTF method[11] saves memory by extending 3D tracks in real-time with 2D tracks. ANT-MOC implements 3D track generation and 3D ray tracing based on the OTF method on GPUs. This ensures that 3D segments are generated and immediately used in real-time on GPUs. The OTF track generation method significantly saves memory footprint but introduces a significant amount of computation.

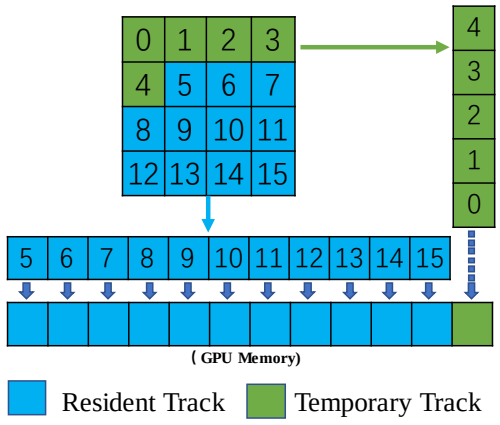

Figure 4: Track management strategy

In order to maximize GPU memory utilization and minimize redundant track generation, we have developed a track management strategy as illustrated in Figure 4. This approach categorizes tracks into resident and temporary tracks. Resident tracks, which are stored in the GPU memory, contain segments that do not need to be expanded during transmission resolution. Temporary track segments are discarded once the transmission solution has been executed. When selecting resident tracks, we rank them based on the number of segments calculated by Equation (4), with preference given to those with more segments in order to reduce the number of load operations during ray tracing. Furthermore, we have implemented a fused ray tracing and source computation kernel to minimize both kernel switching overhead and memory copy.

4.2 Three-level load mapping strategy

As demonstrated by Equation (4), the number of segments reflects the computational cost when solving transport problems. However, due to non-uniform grid partitioning in Geometry Construction, sub-geometries resulting from spatial decomposition may have different execution times during Transport Solving. This can cause delays in communication synchronization, leading to faster processes becoming idle and wasting computing resources. Moreover, variations in 3D track lengths lead to an imbalance in thread workload inside GPUs. To address these issues, we propose a three-level load-mapping strategy that matches the hardware architecture of supercomputers. Figure 5 illustrates how different partitioning granularities are mapped onto the hardware. The load-mapping strategies for each level are described as follow.

4.2.1 L1. The first-level load-mapping strategy is designed to decompose and evenly map the geometry to multiple compute nodes.As shown in Figure 5(1), a 3D spatial geometry representing the actual reactor structure is first decomposed into 8 sub-geometry with a grid size of 2x2x2. Next, we estimate the computational load of each sub-geometry using the performance model in Equation (4) and construct a graph data structure with the computational load as the nodes and the connections between sub-geometries as the edges. After processing by the graph partitioning software such as ParMETIS[15], these 8 sub-geometries are organized into 4 groups and assigned to 4 distinct nodes. We chose a grid size of 2x2x2 for ease of illustration in demonstrating the spatial decomposition. In practice, the number of sub-geometry resulting from spatial decomposition is usually about tenfold the number of nodes. It is worth noting that the value of 10 is determined based on empirical observations[23]. On the one hand, selecting a value that is too low might hamper the potential load-mapping gains. On the other hand, opting for a value that is excessively large would result in convoluted graph structures and failure to achieve the ideal partitioning effectiveness. The determination of the specific value is worthy of detailed investigation and analysis.

4.2.2 L2. The second-level load-mapping strategy is designed to decompose and evenly map the sub-geometry group to multiple GPUs. As shown in Figure 5(2), a sub-geometry group is merged into a fusion-geometry, which is then decomposed and assigned to different GPUs based on the azimuthal angle of 2D tracks. The tracks are then copied to the respective GPUs for parallel processing. Since the number of azimuthal angles is defined as a multiple of 4, the number of GPUs per node is usually an even number, This partitioning approach helps to achieve reasonable parallelism. Furthermore, this method also enables efficient transfer of track flux between adjacent sub-geometries within each GPU.

4.2.3 L3. The third-level load-mapping strategy is designed to sort and evenly map the 3D tracks in a single GPU to CUs. As shown in Figure 5(3), the tracks are first arranged in descending order according to the number of segments sorted by segment_num. After the sorting of tracks, each CU is mapped with tracks in a sequential manner. Then, the remaining tracks are mapped to each CU using the same sequence until all tracks have been mapped to a CU. Once the tracks are sorted and mapped onto different CUs, each CU will perform unique computational branches based on the resident and temporary tracks defined by the track management algorithm in section 4.1. Finally, the source will be updated.

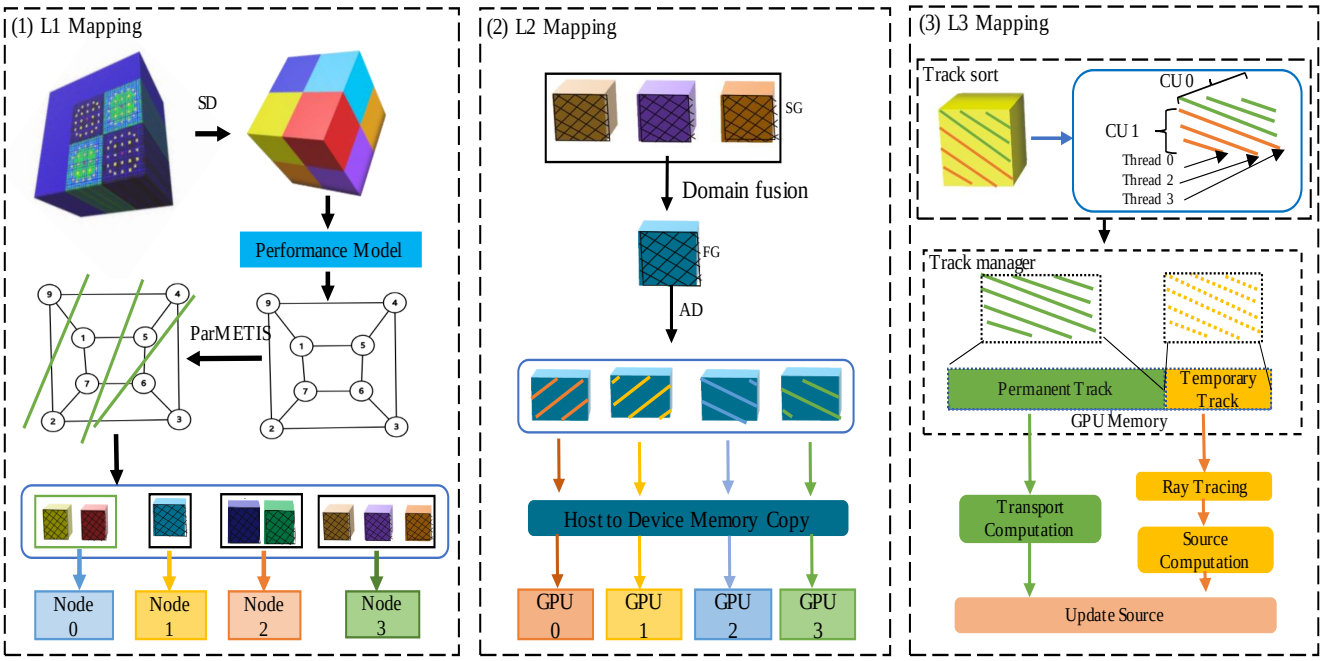

Figure 5: Three-level load mapping strategy.First-level (L1) load-mapping is node-awared, second-level(L2) load mapping is GPU-awared, and third-level (L2) load mapping is CU-awared.

5 EVALUATION

The OECD/NEA has published the C5G7 3D extension benchmark for simulation performance tests of various 3D neutron transport codes. In this section, we use ANT-MOC code to solve the C5G7 3D extension benchmark to evaluate its correctness and efficiency. Figure 6 shows the radial arrangement of the C5G7, which is a real-world case with strong inhomogeneity used to validate ANT-MOC. The real-world case consists of two UO_2 assemblies, two MOX assemblies, and five reflector assemblies. The UO_2 and MOX assemblies have 17x17 pin cells, while the reflector assemblies are homogeneous. Each pin cell is square with a side length of 1.26 cm and a pin radius of 0.54 cm. The UO_2 assemblies are composed of a fission chamber, UO_2 fuel cells, and guide tubes. Besides a fission chamber and guide tubes, MOX assemblies are composed of MOX fuel cells with three different enrichment of U235(4.3%, 7.0%, and 8.7%).

The experiments are carried out on an AMD GPU cluster which has more than 4,000 computing nodes. Each node includes one 32-core AMD Zen-based processor, four AMD Instinct MI60 GPUs, and 128 GB host memory. The processor has four Core Complex Dies, each configured as a NUMA node. Each GPU has 64 CUs and 16 GB global memory. The high-speed computing network utilizes the HDR InfiniBand network, with a network speed of 200Gbps between computing nodes.

5.1 Correctness Validation

To verify the correctness of ANT-MOC, a comparison between ANT-MOC and OpenMOC is conducted. The same experimental parameters are shown in Table 4. The reasonable axial and radial

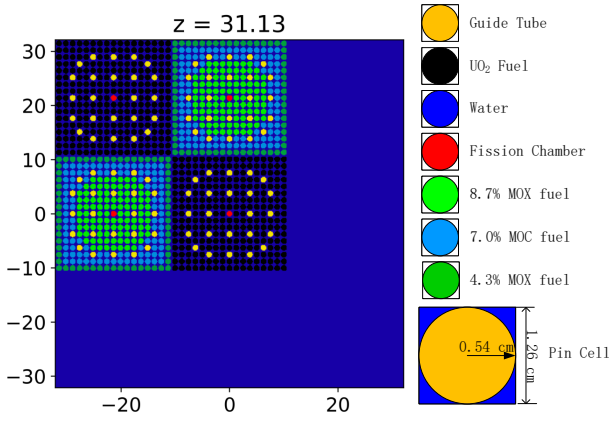

Figure 6: Horizontal section of the C5G7 3D extension benchmark.

track laying parameters are set to ensure that each FSR has tracks passing through. The geometry model is divided into 2×2×2 sub-geometries in 3D space, and these eight sub-geometries are mapped to eight GPUs cross nodes in ANT-MOC, while they are mapped to eight CPU cores across nodes in OpenMOC.

The two software are compared in terms of assembly pin-wise fission rate and K_{eff}. In this test, the K_{eff} values are always consistent during the convergence of the two software. Fission rate distribution can provide important information on the behavior of the reactor, such as the power density and neutron flux distribution.

Table 4: Experimental parameters of ANT-MOC and OpenMOC

Item	Parameters
Size of geometry	64.26^3 cm^3
Number of assemblies	3x3
Number of azimuthal angles	4
Number of polar angles	4
Radial track spacing	0.5
Axial track spacing	0.1
Number of sub-geometry	2x2x2

The fission rate distribution is conducted by ParaView[17], and its results are shown in Figure 7. The figure illustrates the distribution of the fission rate calculated for the C5G7 benchmark. It demonstrates that the components at the central position exhibit a high fission rate, gradually decreasing towards the periphery. The relative error of the assembly pin-wise fission rate between ANT-MOC and OpenMOC are all zero, it can be concluded that ANT-MOC has the same simulation accuracy with OpenMOC. We also have done a comparison of the execution time of ANT-MOC (1 GPU) compared with OpenMOC-3D(8 CPU cores) and ANT-MOC achieves up to 428 times performance improvement.

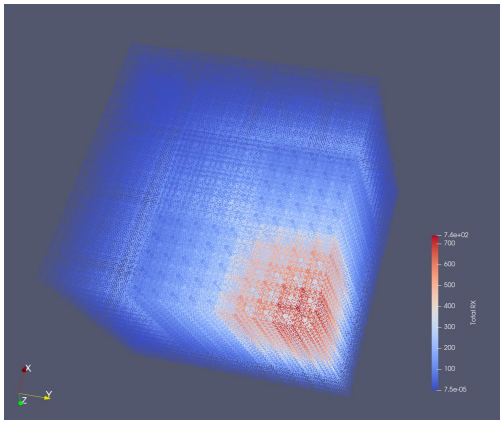

Figure 7: Visualization of the fission rate distribution.

5.2 Performance Model Validation

As outlined in Equation (5)-(7), communication traffic is a quantifiable metric that is directly related to the number of 3D tracks. Conversely, memory and computation requirements are determined by the number of segments, which is an estimated value. Therefore, the accuracy of the performance model can be evaluated by comparing the difference between the estimated segment number and that of the actual value. A set of experiments with different numbers of tracks are used to test the performance, and the results are as shown in Figure 8. The x-axis represents the number of tracks, while the primary axis displays the segment number and the secondary axis presents the relative error value (eff).

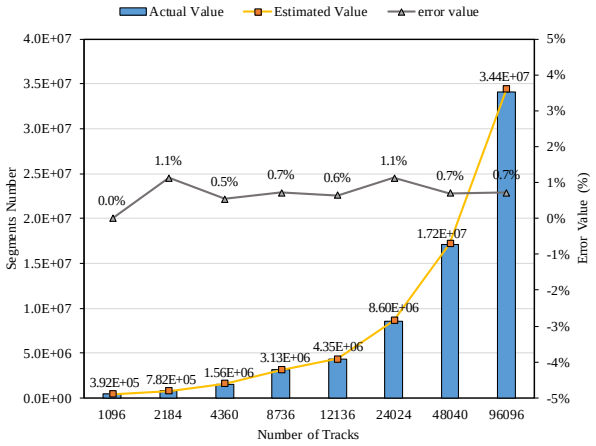

Figure 8: Comparison between predicted and measured values of the number of segments.

Figure 8 shows that the estimated and measured values of the segments are closely matched, and the relative error between the two fluctuates within 1.1%. Therefore, we conclude that the estimation of segments is accurate and indirectly proves the accuracy of the estimation of memory and computation.

5.3 Computation and Memory Evaluation

There are three methods for storing tracks and segments in memory. The EXP method stores all tracks explicitly, resulting in faster computation without the need for recomputation. The OTF method overcomes the memory bottleneck of EXP, while the Manager method strikes a balance between OTF and EXP, achieving faster calculations while efficiently utilizing memory. To evaluate the performance and suitability of the three methods, we conducted experiments using five different number of tracks. Specifically, we measured the average solver runtime over 10 transport iterations and recorded memory utilization prior to the start of transport execution. We set the memory threshold for resident tracks at 6.144GB using a greedy approach, which is subject to variation depending on the hardware used.

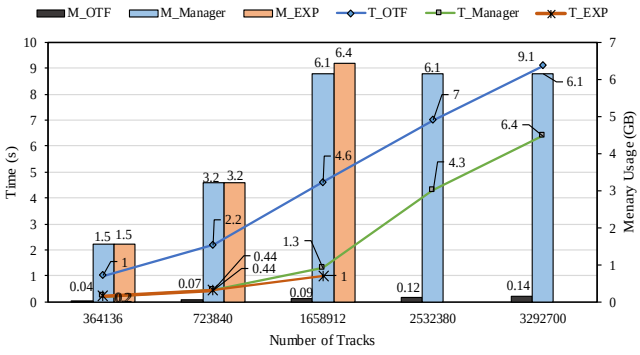

Figure 9: Memory and time comparison of three methods.

The time and memory requirements of all three methods are displayed in figure 9, with time represented by a line graph and memory by a bar chart. The explicit method (EXP) serves as the benchmark. The horizontal axis represents the track scale of the test case, the primary axis represents the solver runtime, and the secondary axis shows memory utilization. The M_OTF represents the peak memory usage of the example using OTF, and T_OTF indicates its execution time. Based on the performance comparison presented in Figure 9, we can observe that the EXP provides the fastest solution speed, but its effectiveness is limited by GPU memory as the scale of the problem increases. On the other hand, the OTF method significantly reduces memory consumption through its real-time track generation technology, but it introduces a track generation kernel that is five times larger than the source computation kernel. Finally, the track management strategy (Manager) balances the benefits of both methods to enable high-resolution simulations and minimize the computational overhead introduced by OTF. As a result, it delivers a 30% performance improvement for OTF simulations.

5.4 Load Balance Evaluation

In order to evaluate the load balancing approach for parallel tasks, we define the load uniformity index for MAX load/Avg load, which is a value greater than 1. The closer the value is to 1, the more balanced the load mapping will be. Figure 10 shows the comparison between the load imbalance degree of the baseline and the three-level load mapping strategy. The baseline is the partitioning method adopted by OpenMOC and is also mentioned as No balance. The x-axis is the amount of GPU, and the y-axis is the load uniformity index.

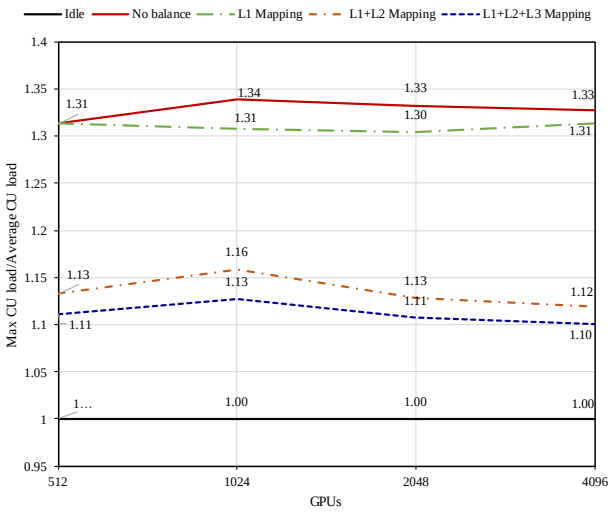

Figure 10: Comparison of multi-level load mapping strategy.

According to the experimental results, the L1 load mapping method reduces the load by 5%. The L2 load mapping method reduces the load by 53%. The L3 load mapping method reduces the load by 8%. The limited gains observed from the implementation of L3 can be attributed to the fact that L2 partitioning had already facilitated a relatively high level of load balancing across multiple GPUs in a single node. Nevertheless, L3 still resulted in a significant increase in performance for each individual GPU. Furthermore, load mapping at the sub-geometry granularity of L1 determines the maximum potential improvement in load balancing, and enhancing the mapping strategy of L1 is crucial for achieving further performance gains in this area.

5.5 Scalability

In this test, we took a high-resolution simulation with a scale up to 100 billion tracks and trillion segments, which has never been achieved before in neutron transport solution software. Scalability tests were also conducted for the three-stage load mapping strategy to evaluate the impact of the optimizations on scalability.

We evaluated the scalability of ANT-MOC on the AMD GPU cluster. We utilized 1000 GPUs (250 nodes) as the baseline throughout our testing process. On average, each GPU was respectively allocated 54,581,544 and 5,124,596 tracks for strong and weak scalability tests. We employed the C5G7 3D extension benchmark during our evaluation, and the number of GPUs used in the test was from 1000 to 16000. During the experiments conducted on the weak scalability design, we varied the number of segments, which fluctuated within 1% error.

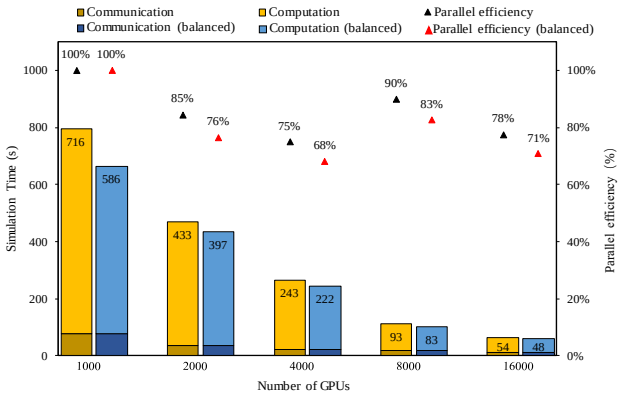

Figure 11: Strong scalability results.

Figure 11 shows the strong scalability test results of ANT-MOC. for 8000 GPUs, the parallel efficiency is observed to increase. In this case, all tracks are resident because GPU memory is enough. This reduces the necessary computations for ray tracing, especially when the single GPU task size decreases. In addition, the figure portrays the scalability achieved by implementing load-balancing strategies, which significantly reduce computation time while preserving high levels of parallel efficiency. Even in the largest test case, load balancing still provides a performance improvement of up to 12%. However, The addition of load mapping strategies also resulted in a reduction of parallel efficiency. This decrease in parallel efficiency occurred due to the reduced degree of load imbalance resulting from the decrease in single-processor loads during strong scalability testing. This reduction limited the potential for optimizing load

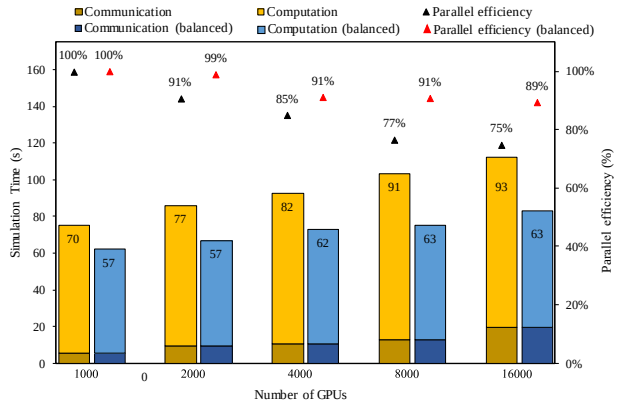

Figure 12: Weak scalability results.

mapping strategies, leading to a decrease in the effectiveness of these strategies. After incorporating all optimization capabilities into ANT-MOC, the parallel efficiency of strong scalability testing reached 71% on 16,000 GPUs, indicating that ANT-MOC possesses performance potential on even larger-scale hardware.

Figure 12 shows the weak scalability test results. As observed in weak scalability testing, The increase in computation time with scale results from implementing the spatial decomposition method for task partitioning in ANT-MOC, which generates additional grids and thereby contributes to an increase in computational complexity. Despite our attempts to adjust the number of segments proportionally, the increase in grids still exacerbated the degree of load imbalance. However, In tests where a load mapping strategy was configured, the reduction in parallel efficiency caused by spatial decomposition was alleviated. After implementing all optimization capabilities of ANT-MOC, the weak scalability parallel efficiency reached 89% on 16,000 GPUs with 174.66 billion tracks. This finding suggests that ANT-MOC is capable of handling large-scale problems.

6 CONCLUSION

In this work, we develop a software named ANT-MOC to solve the 3D full-core NTE using the MOC on GPU clusters. We present a three-level load mapping strategy which adapted to the hardware structure to alleviate the severe load imbalance in ANT-MOC. The tasks are divided into different granularities and mapped to different hardware levels. To reduce the bottleneck caused by the GPU memory for large-scale simulations, we port the main kernel for transport calculation and implement a real-time track generation method on the GPU. Furthermore, we minimize the "time for space" cost through a track management strategy. For the first time, we reach the simulation scale of billions of characteristic lines and achieve good scalability even on up to 16000 GPUs. Moreover, The wide occurrence of load imbalance in parallel software often leads to a waste of energy and money. It is worth investigating and analyzing this phenomenon further. The multi-level load mapping strategy proposed in this paper for neutron transport software has the potential to be further discussed and promoted for its universality, which could help alleviate the waste caused by load imbalance.

ACKNOWLEDGMENTS

This research was supported by the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No.XDB0500000), the Youth Innovation Promotion Association CAS (2021167), the GHfund B(ghfund202302028732) and the Fundamental Research Funds for the Central Universities. The numerical calculations in this study were carried out on the ORISE Supercomputer. We gratefully acknowledge the support of the China Institute of Atomic Energy provided models for this work.

REFERENCES

[1] JR Askew. 1972. *A characteristics formulation of the neutron transport equation in complicated geometries.* Technical Report. United Kingdom Atomic Energy Authority.
[2] William Boyd, Samuel Shaner, Lulu Li, Benoit Forget, and Kord Smith. 2014. The OpenMOC method of characteristics neutral particle transport code. *Annals of Nuclear Energy* 68 (2014), 43–52.
[3] William Boyd, Andrew Siegel, Shuo He, Benoit Forget, and Kord Smith. 2016. Parallel performance results for the OpenMOC neutron transport code on multicore platforms. *The International Journal of High Performance Computing Applications* 30, 3 (2016), 360–375.
[4] K Bradley. 2013. *NEAMS: the nuclear energy advanced modeling and simulation program.* Technical Report. Argonne National Lab.(ANL), Argonne, IL (United States).
[5] Christian Chauliac, José-Maria Aragonés, Dominique Bestion, Dan Gabriel Cacuci, Nicolas Crouzet, Frank-Peter Weiss, and Martin A Zimmermann. 2011. NURESIM–A European simulation platform for nuclear reactor safety: Multi-scale and multi-physics calculations, sensitivity and uncertainty analysis. *Nuclear Engineering and Design* 241, 9 (2011), 3416–3426.
[6] Jun Chen, Zhouyu Liu, Chen Zhao, Qingming He, Tiejun Zu, Liangzhi Cao, and Hongchun Wu. 2018. A new high-fidelity neutronics code NECP-X. *Annals of Nuclear Energy* 116 (2018), 417–428.
[7] JY Cho. 2002. Three-dimensional heterogeneous whole core transport calculation employing planar MOC solutions. *Trans. Am. Nucl. Soc.* 87 (2002), 234.
[8] Charbel Farhat. 1988. A simple and efficient automatic FEM domain decomposer. *Computers & Structures* 28, 5 (1988), 579–602.
[9] Geoffrey Gunow, Benoit Forget, and Kord Smith. 2019. Full core 3D simulation of the BEAVRS benchmark with OpenMOC. *Annals of Nuclear Energy* 134 (2019), 299–304.
[10] Geoffrey Gunow, Samuel Shaner, William Boyd, Benoit Forget, and Kord Smith. 2017. *Accuracy and performance of 3D MOC for full-core PWR problems.* Korean Nuclear Society - KNS, Korea, Republic of.
[11] Geoggrey Gunow, Samuel Shanner, Benoit Forget, and Kord Smith. 2016. Reducing 3D MOC storage requirements with axial on-the-fly ray tracing. *Physics of Reactors 2016 (PHYSOR 2016)* (2016).
[12] Geoffrey Alexander Gunow. 2018. *Full core 3D neutron transport simulation using the method of characteristics with linear sources.* Ph. D. Dissertation. Massachusetts Institute of Technology.
[13] Cho Jin Young, Joo Han Gyu, Kim Ha Yong, and Chang Moon-Hee. 2003. Parallelization of a three-dimensional whole core transport code DeCART. (Jul 2003).
[14] Yeon Sang Jung, Cheon Bo Shim, Chang Hyun Lim, and Han Gyu Joo. 2013. Practical numerical reactor employing direct whole core neutron transport and subchannel thermal/hydraulic solvers. *Annals of Nuclear Energy* 62 (2013), 357–374.
[15] George Karypis, Kirk Schloegel, and Vipin Kumar. 1997. Parmetis: Parallel graph partitioning and sparse matrix ordering library. (1997).
[16] Blake W Kelley, Benjamin Collins, and Edward W Larsen. 2013. *2D/1D approximations to the 3D neutron transport equation. II: Numerical comparisons.* Technical Report. American Nuclear Society, 555 North Kensington Avenue, La Grange Park, IL ....
[17] Kitware Inc. 1999. ParaView. https://www.paraview.org.
[18] Brendan Kochunas, Benjamin Collins, Dan Jabaay, Thomas J Downar, and William R Martin. 2013. *Overview of development and design of MPACT: Michigan parallel characteristics transport code.* Technical Report. American Nuclear Society, 555 North Kensington Avenue, La Grange Park, IL ....
[19] Brendan Matthew Kochunas. 2013. *A Hybrid Parallel Algorithm for the 3-D Method of Characteristics Solution of the Boltzmann Transport Equation on High*

Performance Compute Clusters. Thesis. http://deepblue.lib.umich.edu/handle/2027.42/100072 Accepted: 2013-09-24T16:03:54Z.

[20] Brendan Matthew Kochunas. 2013. *A Hybrid Parallel Algorithm for the 3-D Method of Characteristics Solution of the Boltzmann Transport Equation on High Performance Compute Clusters*. Ph. D. Dissertation.

[21] Shinya Kosaka and Etsuro Saji. 2000. Transport theory calculation for a heterogeneous multi-assembly problem by characteristics method with direct neutron path linking technique. *Journal of nuclear science and technology* 37, 12 (2000), 1015–1023.

[22] Elmer Eugene Lewis and Warren F Miller. 1984. Computational methods of neutron transport. (1984).

[23] Michael J Quinn. 2003. Parallel programming. *TMH CSE* 526 (2003), 105.

[24] Paul K Romano, Nicholas E Horelik, Bryan R Herman, Adam G Nelson, Benoit Forget, and Kord Smith. 2015. OpenMC: A state-of-the-art Monte Carlo code for research and development. *Annals of Nuclear Energy* 82 (2015), 90–97.

[25] Didier Schneider, F Dolci, F Gabriel, J-M Palau, M Guillo, and B Pothet. 2016. APOLLO3® CEA/DEN deterministic multi-purpose code for reactor physics analysis. In *PHYSOR 2016–Unifying Theory and Experiments in the 21st Century*.

[26] Daniele Sciannandrone, Simone Santandrea, and Richard Sanchez. 2016. Optimized tracking strategies for step MOC calculations in extruded 3D axial geometries. *Annals of Nuclear Energy* 87 (2016), 49–60.

[27] Samuel Shaner, Geoffrey Gunow, Benoit Forget, and Kord Smith. 2015. Theoretical Analysis of Track Generation in 3d Method of Characteristics. *Prof. Forget via Chris Sherratt* (April 2015). https://dspace.mit.edu/handle/1721.1/108661 Accepted: 2017-05-04T14:49:17Z ISBN: 9781510808041 Publisher: American Nuclear Society.

[28] Peitao Song, Zhijian Zhang, Liang Liang, Qian Zhang, and Qiang Zhao. 2019. Implementation and performance analysis of the massively parallel method of characteristics based on GPU. *Annals of Nuclear Energy* 131 (2019), 257–272.

[29] Xiao Tang, Qing Li, Xiaoming Chai, Xiaolan Tu, Wenbin Wu, and Kan Wang. 2017. Efficient procedure for radial MOC and axial SN coupled 3D neutron transport calculation. In *Proc. Int. Conf. Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C2017)*. 16–20.

[30] Top500. 2022. TOP500 Lists. http://www.top500.org/lists/

[31] John A Turner, Kevin Clarno, Matt Sieger, Roscoe Bartlett, Benjamin Collins, Roger Pawlowski, Rodney Schmidt, and Randall Summers. 2016. The virtual environment for reactor applications (VERA): design and architecture. *J. Comput. Phys.* 326 (2016), 544–568.

[32] GJ Wu and R Roy. 2003. A new characteristics algorithm for 3D transport calculations. *Annals of Nuclear Energy* 30, 1 (2003), 1–16.

[33] Wenbin Wu, Yingrui Yu, Qi Luo, Dong Yao, Qing Li, and Xiaoming Chai. 2020. Verification of the 3D capability of OpenMOC with the C5G7 3D extension benchmark. *Annals of Nuclear Energy* 140 (June 2020), 107293. https://doi.org/10.1016/j.anucene.2019.107293

[34] Wenbin Wu, Yingrui Yu, Qi Luo, Dong Yao, Qing Li, and Xiaoming Chai. 2020. Verification of the 3D capability of OpenMOC with the C5G7 3D extension benchmark. *Annals of Nuclear Energy* 140 (June 2020), 107293. https://doi.org/10.1016/j.anucene.2019.107293

[35] Akio Yamamoto, Masahiro Tatsumi, Yasunori Kitamura, and Yoshihiro Yamane. 2004. Improvement of the SPH method for pin-by-pin core calculations. *Journal of nuclear science and technology* 41, 12 (2004), 1155–1165.

[36] Wen YANG, Changjun HU, Tiancai LIU, An WANG, and Mingyu WU. 2019. Research progress of China virtual reactor (CVR1. 0). *Atomic Energy Science and Technology* 53, 10 (2019), 1821.

[37] Ao Zhang, Ming Dai, Maosong Cheng, Jianhui Wu, and Jingen Chen. 2022. Development of a GPU-based three-dimensional neutron transport code. *Annals of Nuclear Energy* 174 (Sept. 2022), 109156. https://doi.org/10.1016/j.anucene.2022.109156

[38] Ao Zhang, Ming Dai, Maosong Cheng, Jianhui Wu, and Jingen Chen. 2022. Development of a GPU-based three-dimensional neutron transport code. *Annals of Nuclear Energy* 174 (2022), 109156.

[39] Hongbo Zhang, Hongchun Wu, and Liangzhi Cao. 2011. An acceleration technique for 2D MOC based on Krylov subspace and domain decomposition methods. *Annals of Nuclear Energy* 38, 12 (Dec. 2011), 2742–2751. https://doi.org/10.1016/j.anucene.2011.08.015

[40] H. Zhang, Y. Zheng, H. Wu, and L. Cao. 2013. *A 2D/1D coupling neutron transport method based on the matrix MOC and NEM methods*. Technical Report. American Nuclear Society - ANS; La Grange Park (United States). https://www.osti.gov/biblio/22212728

[41] Jingchao Zheng, Zhiqiang Wang, Zeyi Xie, Xingjie Peng, Chen Zhao, and Wenbin Wu. 2023. Parallel Communication Optimization Based on Graph Partition for Hexagonal Neutron Transport Simulation Using MOC Method. *Energies* 16, 6 (Jan. 2023), 2823. https://doi.org/10.3390/en16062823 Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

Appendix: Artifact Description/Artifact Evaluation

ARTIFACT DOI

10.5281/zenodo.8084790

ARTIFACT IDENTIFICATION

In this appendix, We described the technical details for the validation of ANT-MOC. ANT-MOC supports full core neutron transport solution on GPU cluster. This article respectively describes GPU kernels, performance model, track management strategy, and multi-level load mapping strategy of ANT-MOC. In addition to the verification of the above modules, a whole-core 3D pressurized water reactor simulation was used to verify the scalability of ANT-MOC.

The module selection and the parameter setting are read through a specific configuration file. The time or storage indicators are given in the run log.

ANT-MOC is available as open source on
https://github.com/lsder/ANT-MOC.

REPRODUCIBILITY OF EXPERIMENTS

Experimental tasks are submitted via slurm scripts. The relevant experiment construction scripts are stored in the benchmark path, and users can compile and reproduce them separately.

An example of a slurm script is shown below.

```
$ cat run/slurm.job
NTASKS=8
LOGFILE=c5g7-$NTASKS-%j.log
ERRFILE="c5g7-$NTASKS-%j.err"
JOBNAME="MOC"
echo "TASK MOC C5G7 TEST START NTASK=$NTASKS DOMAIN={$D1,$D2,$D3}"
sbatch « END
#!/bin/bash
#SBATCH -J $JOBNAME
#SBATCH -o $LOGFILE
#SBATCH -e $ERRFILE
#SBATCH -p normal
##SBATCH -c 8
#SBATCH –gres=dcu:4
#SBATCH -n 8
mpirun –oversubscribe -n $NTASKS ../build/run/newmoc –config="config.yaml"
END
```

Where NTASKS is the number of GPUs to be called, and in config.yaml we need to adjust the number of domain_decomposition to be consistent with NTASKS.

We add several different sets of profiles in the benchmark path to verify the experiments covered in this paper, where you can submit tasks by” sh slurm.job ” for verification.

According to the log fragments generated by the program, the execution time and storage usage of each stage of the software can be analyzed through the log file.

ARTIFACT DEPENDENCIES REQUIREMENTS

The experiments are carried out on an AMD GPU cluster that has more than 4,000 computing nodes. Each node includes one 32-core AMD Zen-based processor, four AMD Instinct MI60 GPUs, and 128 GB host memory. The processor has four Core Complex Dies, each configured as a NUMA node. Each GPU has 64 CUs and 16 GB of global memory. The high-speed computing network utilizes the HDR InfiniBand network, with a network speed of 200Gbps between computing nodes. The operating system is NFS Server 3.2. The slurm version is 20.11.8-1.4.1-20220823. The version of Module greater than 3.2.10 is recommended.

we need to load the environment by sourcing the shell " env.sh".

```
$ cat env.sh
module purge
module load compiler/cmake/3.24.1
module load compiler/rocm/3.9.1
module load compiler/devtoolset/7.3.1
module load mpi/openmpi/4.0.4/gcc-7.3.1
```

The source compilation environment shown as follows:

```
Currently Loaded Modulefiles:
1) compiler/rocm/3.9.1 3) compiler/devtoolset/7.3.1
2) compiler/cmake/3.23.1 4) mpi/hpcx/2.4.1/intel-2017.5.239
```

The model and data for the experiment are fixed in the code of benchmarks.

ARTIFACT INSTALLATION DEPLOYMENT PROCESS

After environment initialization is complete, the software is built by command :

```
git clone https://github.com/lsder/ANT-MOC.git
$cd ANT-MOC
$ mkdir build && cd build && cmake ../ && make -j 24
```

After building, we can copy the newmoc executable file to the benchmark path. the experiments can be run using job scripts in the benchmark.