EPJ Web of Conferences **302**, 04009 (2024) https://doi.org/10.1051/epjconf/202430204009
SNA + MC 2024

Adaptive Memory Saving Strategy in Shielding Calculation of RMC

Yingzhe Hu[1,*], *Shanfang* Huang[1], *Zeguang* Li[1], *Zhaoyuan* Liu[1], *Kan* Wang[1], and *Tao* Ma[1]

[1]Tsinghua University

Abstract. RMC(Reactor Monte Carlo) is a Monte Carlo simulation software used in reactor design and has been verified by different benchmarks. The former data structure and algorithm have been optimized for criticality and burn-up calculation making it inconvenient to be implemented into shielding simulation and variance reduction. In order to expand its availability in shielding calculation, several optimizations have been made in RMC code design to elevate efficiency in variance reduction. Considering parallel computation has been implemented in Monte Carlo simulation for long time, in this paper we proposed an adaptive memory saving strategy specifically designed for boosting parallel variance reduction calculation in RMC. We further found that this strategy can be very effective in certain condition without sacrificing any calculation performance.

1 Introduction

Parallel computation has been implemented in Monte Carlo Calculation for quite long and has already been recognized as a necessary part of reactor simulation. However, the parallel data structure and algorithm design is quite varied for different problem types. In RMC parallel computation has been implemented for long and its ability has been verified by a lot of benchmarks[2]. Considering current parallel ability was designed mainly for criticality and burn-up calculation, it is reasonable to optimize parallel data structure and algorithm in shielding calculation, especially in simulation of deep-penetration problems. In this paper we will first show the parallel program model in RMC and then introduce two new techniques we have already developed in RMC code for memory saving in some problems with extremely large weight window parameter input. These two techniques are developed with MPI and OpenMP respectively, serving for different computer hardware environments and computation needs. They are all later verified using different models, showing that they are all practical optimization for industrial simulation.

2 Parallel Model in RMC

In RMC multiple parallel models has been implemented in particle transportation, especially MPI and OpenMP. They are designed for different targets and serve for multiple problems. MPI parallelization was originally introduced in RMC as an important component

*e-mail: hyz20@mails.tsinghua.edu.cn

for particle transportation acceleration and OpenMP parallelization was first implemented in RMC for memory sharing because of the large consumption from cross section tally of burn-up calculation. These two techniques are both proven effective. Later both were combined in RMC for the industrial simulation of reactor cores and burn-up problems.[2] Now in RMC MPI and OpenMP can used separately and can also be used in combination.

2.1 MPI Parallel Model

In RMC code, MPI parallelization is used in many simulation problems including criticality, burnup and fixed source calculation[2]. It's developed mainly for accelerating particle transportation and it's been proven effective with multiple benchmarks including HBR-2[3].

For a more clear definition, MPI parallel structure is shown in figure 1 below. As important parts of the code are all copied by each process, every single process has its own copy of the geometry and material information of the model to be simulated. In this parallel model the tally data and particle transportation run-time data are different between processes. After the simulation runs to the batch end, RMC can use the MPI communication mechanism to transport data between different processes and gather necessary data from each process to get the total statistical results. Last these data was processed by the main process and be output in the file for user to check.

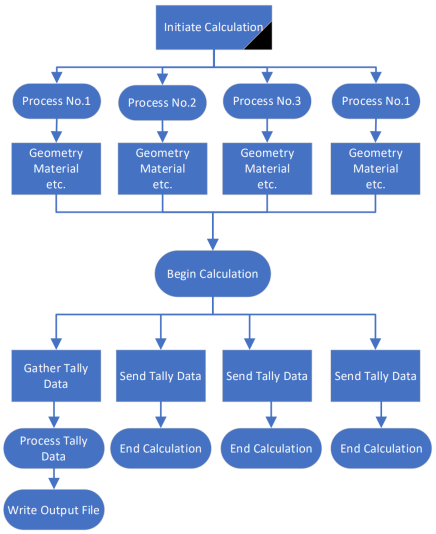

Figure 1. MPI Parallel Model in RMC code

However, this parallel model will duplicate the memory allocation among the processes, including particle, nuclear data and tally information. In certain condition with extraordinarily large memory consumption this parallel model will cause unacceptable memory cost even on the super computers with multiple nodes. In this case some modification must be made so memory cost can be controlled.

2.2 OpenMP Parallel Model

In order to reduce the memory cost, OpenMP was introduced in RMC. It was mostly used in burn-up calculation and has shown the capability of dealing with large memory cost.

In simulation of benchmark with large number of burn-up region, RMC code with OpenMP successfully reduced the memory cost and completed the calculation task.

In RMC code, OpenMP thread level parallel is organized as figure 2 below. As is shown, with OpenMP thread parallelization, many components of RMC are shared between threads thus reduced the memory consumption. In burn-up calculation with a large number of burning region, memory of cross section tally was shared between threads[2].

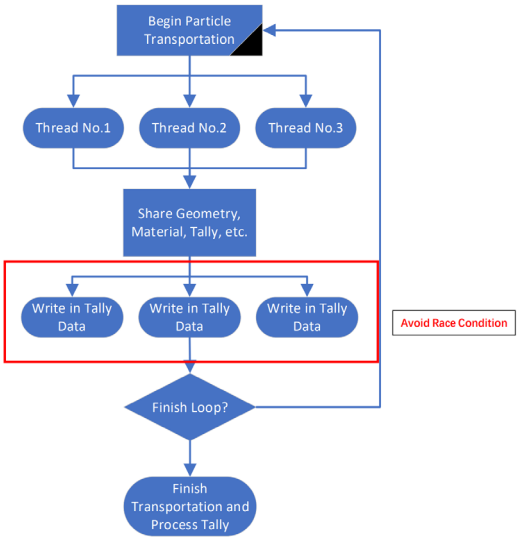

Figure 2. OpenMP Parallel Model in RMC code

OpenMP thread parallelization can easily solve the memory cost of duplicate data structure, but in order to avoid race condition in particle transportation simulation, some performance must be sacrificed. This is very common in Monte Carlo program[4]. In order to balance the memory cost and the performance, we developed an adaptive memory saving strategy in RMC, combining MPI and OpenMP with some new feature of MPI parallelization.

3 Shielding Calculation in RMC

As RMC is very famous in nuclear reactor core design, it is also important for the users to perform shielding calculation. In RMC code this kind of problem is mainly performed by fixed source component.

3.1 Code Design of Shielding Calculation in RMC

In RMC shielding calculation as well as fixed source calculation has been optimized and we used Venus-II benchmark to verify its ability.In RMC, fixed source part share the same particle transport code with criticality calculation but has the different posterior process to do particle splitting and tally processing. As is shown in figure 3 below, RMC code has a special optimization for variance reduction in fixed source calculation.

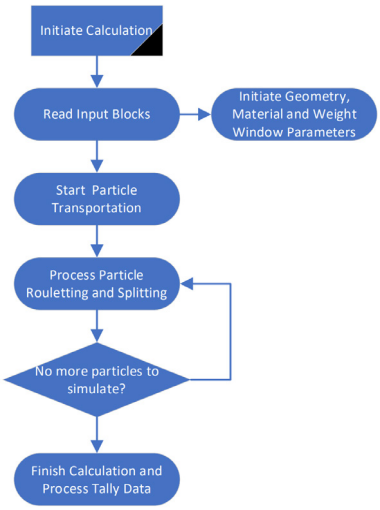

Figure 3. Shielding Calculation in RMC

3.2 Dilemma of deep-penetration calculation

For deep penetration problem, it is usually inevitable to use variance reduction code with weight window to lower the statistical error.For some input with extremely large weight window input,it was previously inconvenient to perform parallel computation using fixed source code in RMC. It's because MPI cannot reduce memory cost when the number of process increases and the memory of a node of a supercomputer is limited. For example, to run an simulation with a weight window input file in size of 5 GB with 100 MPI processes parallelization, 100*5GB memory will be consumed simply by the weight window parameters from the input file. It is also very time consuming for the processes to read all the data into memory with I/O interface. This make it almost impossible to run simulation with large weight window input file especially on the PC, or even the supercomputer with multiple nodes.

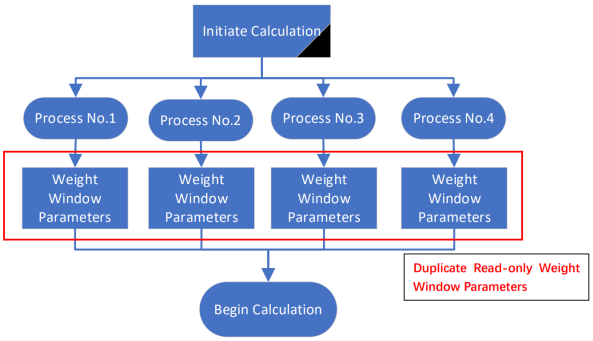

Figure 4. Duplicate Memory of Weight Window Parameters

4 Adaptive Memory Saving Strategy

In shielding calculation of RMC, some updates have been made to solve the memory and performance problem listed above. We introduce an adaptive memory saving strategy to help parallel program to share necessary information between processes or threads. This strategy solves the problem mentioned above and showed acceptable result in verification.

4.1 Remote Memory Access of MPI

In MPI-2, Remote Memory Access (RMA) was introduced to solve the memory consumption problem of MPI interface.[5] This feature allows processes in a parallel program to directly access and modify memory on remote processes without explicit message passing. In detail, it means a process can access the memory of another process without requiring explicit coordination from the target process.

Considering that the weight window parameter inputs are in fact read-only, we can use RMA in MPI parallel process to avoid race condition which is possible in thread parallelization. Thus We used this feature to share read-only weight window parameters between processes.

In RMC code, we use the MPI library to implement the function we proposed in this paper, the specific version is mpich 3.1 and the operation system is Ubuntu 18.04.

4.2 Memory Saving Strategy using RMA

In RMC code, MPI parallel using RMA is implemented in shielding calculation and when variance reduction components of different processes are using weight window parameters, they are all reading the same bulk of memory which was allocated in initialization using RMA. In this case the memory cost is reduce to only one copy of weight window parameter. Compared to the previous memory cost, the implementation of RMA successfully solved the problem of large memory consumption of large number of weight window parameters.

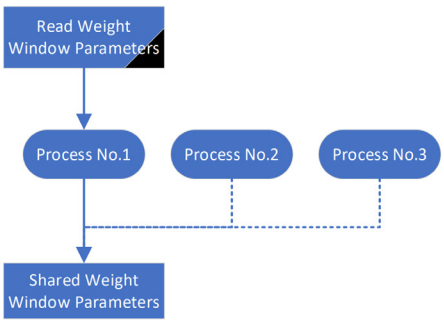

Figure 5. Share Weight Window Parameters using RMA

4.3 OpenMP Parallel Development in Fixed Source Code

In RMC code OpenMP parallelization is also implemented in shielding calculation. With this method, a paralleled computation with a higher rate of memory sharing is developed. In our development, we make it possible that all types of particles can be simulated in

shielding code of RMC and all the necessary components of calculation is shared between threads. In order to avoid race condition some synchronization and lock implementation must be included in code and result in delay of thread calculation which decreases the performance.

4.4 Structure of Adaptive Memory Saving Strategy

Adaptive Memory Saving Strategy is based on the multi-type parallelization of transportation code where all the input-data and running-time data can be paralleled based on actual needs. The structure design is shown in 6. This Figure shows how important data is paralleled in the calculation. The MPI parallelization is used for multiple particles to be simulated simultaneously while the RMA is used for weight window parameters input sharing and the OpenMP code is used for some large running-time data to be shared between threads in order to reduce memory cost.

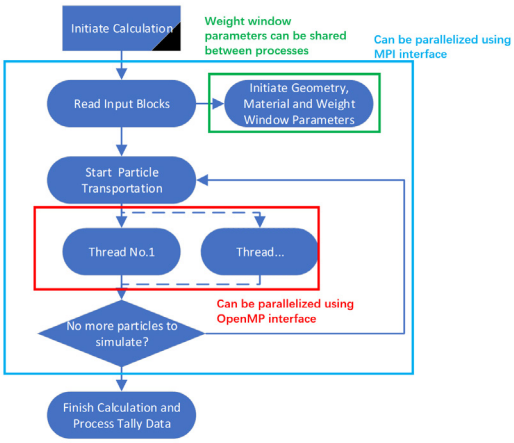

Figure 6. Structure of Adaptive Memory Saving Strategy

5 Verification

In this section, the result of VENUS-II simulation with a large number of weight window parameters. A weight window parameter input in size of 5 GB is used to do variance reduction. For this problem, we performed three kinds of simulation condition: 1) multiple processes without RMA 2) multiple processes with RMA 3) multiple processes along with OpenMP parallelization but without RMA. All the simulations were run with a personal computer that has a AMD Ryzen 9 3950X 16-Core Processor and 32GB memory.

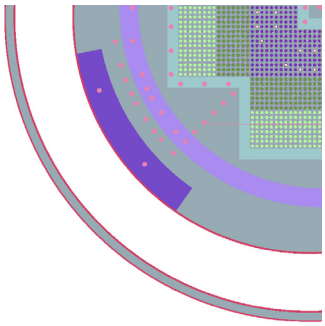

Figure 7. Cross Section of Venus-II benchmark

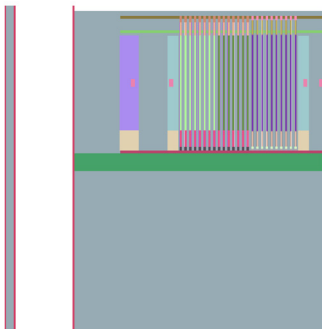

Figure 8. Longitudinal Section of Venus-II benchmark

The results are listed in table 1 below. MPI-2processes means that the corresponding simulation was run with 2 processes in parallel. MPI-6processes means that calculation was run with 6 processes in parallel. OpenMP-10threads means 10 threads performed the calculation and AMSS(Adaptive Memory Saving Strategy) is the method proposed in this paper. AMSS-10 had 10 processes to perform the calculation at the same time.

Table 1. Calculation Time and Memory Cost

Results	Time	Memory Cost
Single Process	60.4min	4.2GB
MPI-2processes	35.2min	8.4GB
MPI-6processes	17.6min	25.2GB
OpenMP-10threads	15.8min	4.2GB
AMSS-2processes	35.2min	4.2GB
AMSS-6processes	17.6min	4.2GB
AMSS-10processes	8.6min	4.2GB

By comparing the simulation time and result and the memory cost, we can find out the advantage of adaptive memory saving strategy in boosting large paralleled Monte Carlo simulation. These parallel strategies all worked as we expected.

As we can see, the effiency of the MPI parallelization has been verified and the calculation time consumption has been reduced. However, the memory cost increases linearly with respect to the number of process.

The efficiency of the calculation with 10 OpenMP threads has scaled badly because of the implementation of tallies of RMC hasn't been fully optimized for OpenMP and the race condition happens fequently during the calculation.

Compared with the MPI and OpenMP parallelization, the RMSS reduced the memory cost with smaller performance sacrifice. As for scaling result, AMSS also has the advantages that it keeps the same scaling efficiency as MPI parallelization as the weight window input is read-only so it isn't affected by the race condition. This method has been verified using the VENUS-II benchmark and is specifically suitable for the calculation of shieding problem with large weight window input on relatively small clusters.

In general, the Adaptive Memory Saving Strategy(AMSS) has shown the same parallel efficiency as the MPI parallelization but with much smaller memory cost. The OpenMP parallelization provided the lowest memory cost in most conditions but it also suffered from the current parallelization design, which will be further optimized in the future.

6 Conclusion

Adaptive Memory Saving Strategy in RMC successfully solved the memory cost problem of deep-penetration calculation in shielding transportation code. With this implementation RMC can be applied in full-reactor shielding calculation with large weight window input and can run on supercomputer cluster for industrial use.

References

[1] Chen et al, ANNALS OF NUCLEAR ENERGY **Volume 147**, (2020)

[2] Wang et al, ANNALS OF NUCLEAR ENERGY **Volume 82**, 121-129 (2015)

[3] Wang et al. Progress on RMC: a Monte Carlo neutron transport code for reactor analysis,International Conference on Mathematic and Computational Method, M&C 2011

[4] Romano et al. ANNALS OF NUCLEAR ENERGY **Volume 51**, 274-281 (2013)

[5] Jenks et al. INTERNATIONAL JOURNAL OF PARALLEL PROGRAMMING **Volume 25**, 281-304 (1997)