7th International Conference on Modelling and Simulation in Nuclear Science and Engineering (7ICMSNSE)
Ottawa Marriott Hotel, Ottawa, Ontario, Canada, October 18-21, 2015

7ICMSNSE-018

Asynchronous Transport Algorithm For Domain Decomposed In JMCT Monte Carlo code*

Gang Li[1], BaoYin Zhang[1], Li Deng[1], Zeyao Mo[1], Ma Yan[1], Rui Li[2], DanHua ShangGuan[1], and YuanGuan Fu[2]

[1] Institute of Applied Physics and Computational Mathematics, Beijing, China
[2] CAEP Software Center for High Performance Numerical Simulation, Beijing, China
lgseven@126.com, zby@iapcm.ac.cn, deng_li@iapcm.ac.cn, zymo@iapcm.ac.cn, ma_yan@iapcm.ac.cn, li_rui@iapcm.ac.cn, sgdh@iapcm.ac.cn, fyg1989@126.com,

Abstract

High-fidelity, large-scale reactor analyses with Monte Carlo can lead to memory overload for a single core processor when it faces the detailed and accurate model of the full-core reactor. Domain decomposed calculation is one of the remedies to solve this problem. In domain decomposition, particles that cross domain boundaries need to be exchanged between the processors. An efficient, robust asynchronous transport algorithm for domain decomposition is introduced in this paper. All point-to-point communications in the algorithm are asynchronous to allow maximal overlap between computational work and communication, and result in increased parallel efficiency. The exchanged particles records are used to exit the loop at the end of the cycle in a criticality calculation when all the particles have finished. Two full-core reactor models, Dayawan reactor and BEAVRS benchmark, are simulated to verify the asynchronous transport algorithm using the Monte Carlo particle transport code JMCT (J Monte Carlo Transport Code).

Keywords: BEAVRS, JMCT, Monte Carlo, domain decomposition, asynchronous transport

1. Introduction

To take advantage of parallel capabilities, most Monte Carlo codes such as MCNP[1] utilize multiple processors by dividing the particles between processors, and synchronizing after every cycle in a criticality calculation or at the end of the simulation for a fixed source calculation[2]. There is no communication between the processors except collecting the tally result. So the parallelism and scalability are very good.

However, this is not an option for high-fidelity, large-scale “real” commercial reactor simulation since a single processor have not enough memory to store all the zones and tallies[3]. For example, the required storage for BEAVRS PWR benchmark is about 1 Terabyte[4]. It is a challenge for storage and calculation methods.

In these cases, the spatial domain must be partitioned and assigned among the processors. This method, known as domain decomposition, is a form of spatial parallelism. It has been applied in the Implicit Monte Carlo (IMC) scheme for thermal radiation transport, such as Milagro code [5-6]. The

* This work is performed under the auspices of National Natural Science Foundation of China (No.61033009, 91118001), the sub-item of special projects of the National Energy Bureau of China (No.2015ZX06002008), the Nuclear Power Development Project of the Science, Technology and Industry for National Defense of China (No.[2012]1523) ,National High Technology Research and Development Program of China(No.2012AA01A309, 2012AA01A303), and the Science Foundation of China Academy of Engineering Physics of China (2014B0202029)

MERCURY code at LLNL, which is a general-purpose Monte Carlo code, implemented combinatorial geometry-based domain de-composition in 2009 [7].

In Monte Carlo simulations using domain decomposition algorithms, processors have to ex-change particles that cross domain boundaries. It is a challenge to reach a high efficiency as well as to keep message communication correct. An asynchronous transport algorithm is described in details in this paper, which is used for domain decomposed particle Monte Carlo in JMCT(J Monte Carlo Transport Code) [8]. The communication efficiency has been considered by overlap between computational work and communication.

2. JCOGIN infrastructure

JMCT is a combinatorial geometry Monte Carlo particle transport code, which has been developed at the Institute of Applied Physics and Computational Mathematics over 30 person-years. It can perform the transport of neutrons and photons, including both multi-group and continuous energy treatments of cross section. It is based on the infrastructure named JCOGIN (J COmbinatorial Geometry Monte Carlo transport INfrastructure). JCOGIN is in charge of storing combinatorial geometry, allocating the memory for tallies and communicating the message for parallel computing, including domain decomposition [9].

In JCOGIN, geometry element is solid, such as spheres, ellipsoids, rectangles, cylinders, cones, for instance. Complex solid can be made from several simple solids via using Boolean operators: intersection, union, and complement. It is called a cell when a solid is filled with materials. All the cells are associated with a tree structure in the memory. Every node on the tree corresponds to a cell, and it means that the cell has other cells inside it when its corresponding node has daughters. Domain decomposition is to divide the whole tree into many sub-trees and assign them on different processors. The processor can only track particles and tally on the corresponding domain of the sub-tree. The particles will be sent to another processor when they cross the domain boundaries.

3. Asynchronous algorithm

In domain decomposed particle Monte Carlo, two sets of data must be communicated between the processors. The nearest neighbours must exchange particles that cross domain boundaries. A global communication operation must also be performed so that all the processors know when all the other processors are finished tracking all the particles.

3.1 Algorithm flow

```
for the other processor
|   post nonblocking receive for one integer: the number of particles (MPI_Irecv)
if master processor
|   post nonblocking receive for the exchanged particles record from all slaves(MPI_Irecv)
else
|   post nonblocking receive for finished message from master(MPI_Irecv)
while not finished
|   if each M particles are simulated
|   |   if other processes send particles (MPI_Test)
```

```
|  |  |  receive the particles, and store them in the received particles set
|  if successfully get a particle from the received particles set or sources
|  |  track the particle until it dies or crosses the domain boundaries
|  |  if the particle crosses the domain boundaries
|  |  |  buffer the information of the particle, including the current random number
|  |  |  if the buffer is full
|  |  |  |  send the particle buffer (MPI_Send)
|  else
|  |  send any partially full particle buffers(MPI_Send)
|  |  if master processor
|  |  |  wait for particles or the exchanged particles record (MPI_Waitany)
|  |  |  |  if particles
|  |  |  |  |  continue track particles
|  |  |  |  if the exchanged particles record
|  |  |  |  |  check the exchanged particles record, and send the finished flag if the result is  |
|  |  |  |  correct. (MPI_Send)
|  |  if slave processor
|  |  |  send the exchanged particles record to the master processor (MPI_Send)
|  |  |  wait for particles or finished flag (MPI_Waitany)
|  |  |  |  if particles
|  |  |  |  |  continue track particles
|  |  |  |  if finished flag
|  |  |  |  |  finish the simulation
Cancel all outstanding nonblocking receives
```

3.2 The exchanged particles record and the received particles set

Each processor saves the total numbers of particles sent to other processes, and the total numbers of particles received from other processors. It is called the exchanged particles record, which is very important to exit the asynchronous transport after every cycle in a criticality calculation. As a processor can exchange particles with any other processors in general， the length of the ex-changed particles record is 2(P-1), where P is the number of processors.

The received particles are stored in a particle set, with the name of the received particles set. The processor gets particles and tracks them one by one from the received particles set. When there is no particle in the set, the processor picks particles from the local source.

3.3 Sending particles

As particles move through the domain of the processor A, they may hit domain boundaries and need to be sent to another processor, for example B. In order to reduce the frequency of communication, particles are buffered. When the buffer is full, all of the N particles in it are sent together. Processor A sends the number of particles to processor B, and waits for the reply. The particles are not sent until the reply is received. After that, processor A updates the exchanged particles record and empties the buffer.

During the time waiting for the reply, processor A need to check if there are any particles sent in from other processors, and received particles if so, avoiding the deadlock that processors are waiting for the replies of each other.

The number N can be adjusted according to memory size. The frequency of sending particles will increase if N is very small. The particles will be backlogged in the buffer, while other processors maybe have no particles to track if N is very big.

3.4 Receiving particles

In order to reduce the time of processor A on waiting for the reply when it is sending particles, processor B should make the reply as soon as possible. However, processor B is busy tracking the local active particles. In the algorithm, processor B checks whether any particles have been sent after tracking every M local particles. When it received the number of the particles from processor A, it sends a reply to processor A immediately, and makes a point-to-point communi-cation with processor A. After receiving the particles, processor B puts them into the received particles set, and updates the exchanged particles record.

The number M should be adjusted according to N. Processor A will wait for a long time if M is relatively too big. While processor B will waste the checking time if M is relatively too small.

3.5 Waiting for finished flag

When there are no local active particles, including the received particles and source particles, the processor sends any partially full particle buffers.

Slave processor sends the exchanged particles record to master processor, and then enters a state waiting for finished flag or new particles from other processors. As a result it will go back to track particles if new particles arrive, or it will exit the cycle if finished flag arrive.

Master processor receives and checks the exchanged particles records, and determines whether to finish the cycle. If the checking result is true, it sends finished flag to slave processors and exit the cycle. If the checking result is false, it enters a state waiting for the exchanged particles re-cord or new particles from other processors. As a result it will go back to track particles if new particles arrive, or it will check the records again if the exchanged particles records get updated.

3.6 The key for finishing the cycle

Master processor checks the exchanged particles records of all processors, including itself. For each two processors, such as processor A and B, master processor makes sure if the number of the particles that processor A sent to processor B, which is got from processor A, is equal to the number of the particles that processor B received form processor A, which is got from processor B. Only when all the equations meet among the processors, the checking result is true. Master processor sends finished flag.

If any equation fails to meet, it means one processor is still tracking particles, yet its newest exchanged particles record is not sent to master processor. So the simulation can't stop. Master processor has to wait the new exchanged particles record and check them again.

The time complexity of checking the records is $O(P^2)$, while P is the number of parallel processors used for domain decomposition.

3.7 Combination with domain replication

The parallel efficiency of domain decomposition is much lower than that of domain replication as there is much message communication during the simulations. So the combination of two parallelisms is provided to improve parallel efficiency. Assuming $P_1 * P_2$ processors in all, P_1 copies of the whole geometry are made firstly, and each copy is assigned $1/P_1$ of the total particles. Then each copy is decomposed into P_2 domains, and each processor got one domain to track the particles. Only the domains decomposed from the same copy can exchange particles to each other.

It is suggested that P_2 should be minimized in the case of sufficient memory. So the extra work of master processor for checking the exchanged particles records ($O(P_2^2)$) is usually small, and it is of little effect on load balancing.

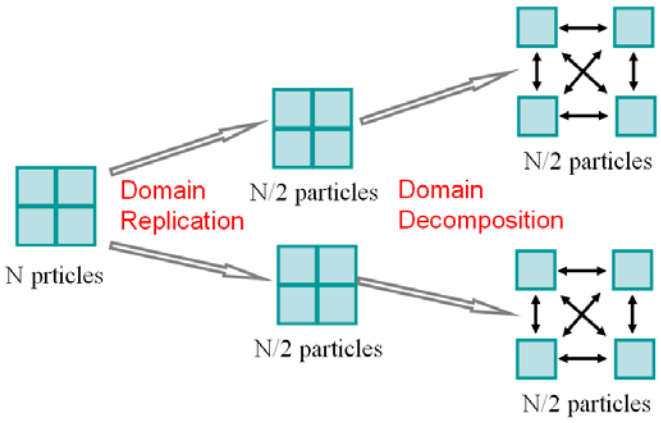


Figure 1. Example of domain decomposition and domain replication are used in combination. Firstly the whole geometry is cloned by two copies, and each copy is assigned half of the total particles. Secondly each is decomposed into 4 domains. There are 2×4=8 processors in all, and each processor is in charge of a domain. Only the domains from the same copy can exchange the particles to each other.

4. RESULTS

4.1 Dayawan PWR of China

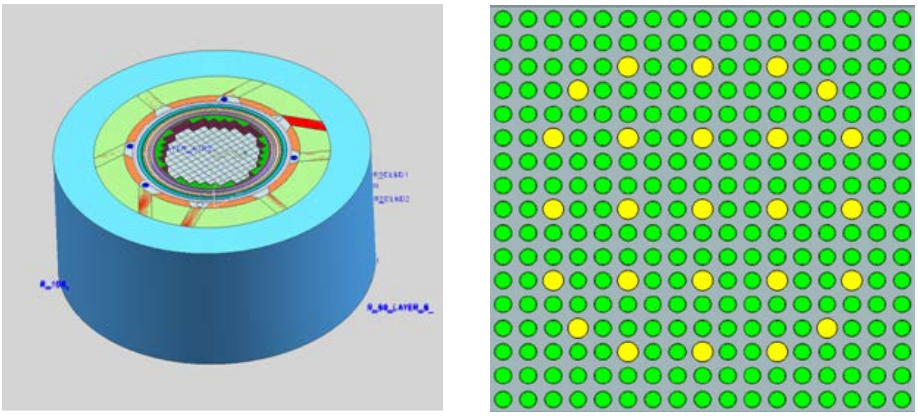


Figure 2. The Dayawan PWR model and cross-sectional views of an assembly

The PWR model from the Dayawan nuclear power station is employed to test the asynchronous transport algorithm. There are 157 assemblies in the model. Each assembly is composed of 17 × 17 pins, with 264 for fuel rod and 25 for control rod, and each rod contain 2 layers in radius. As there is no control rod in this model, the cells for control rod are empty. In the axial direction each assembly is divided into 16 segments. There are 2.11 million cells in this model.

The main purpose of the domain decomposition is to reduce the memory requirement. Figure 3 shows the memory usages of different number of domain decompositions for the Dayawan PWR model that neutron flux is tallied on all the cells with 10 energy bins. We can see the memory usage decreases rapidly with the number of domain decomposition increasing. Once the number of domain decomposition is doubled, the memory decreases nearly half.

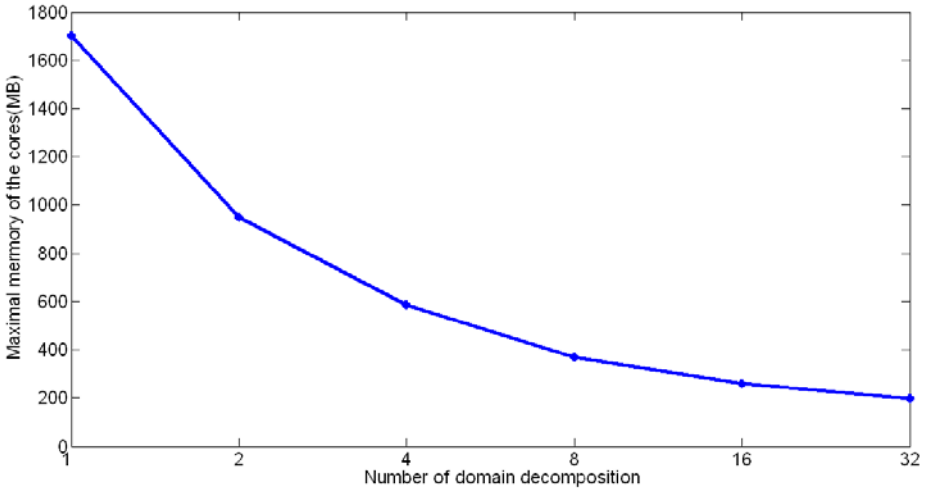
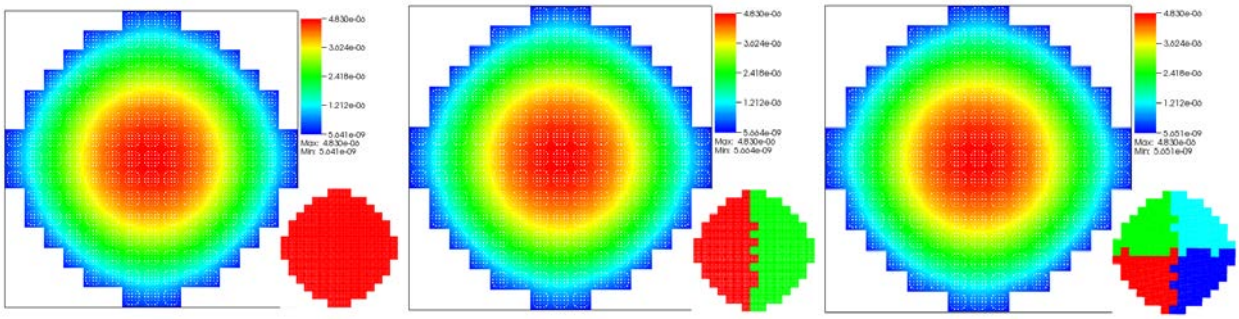


Figure 3. The memory usages of different numbers of domain decomposition

Criticality calculations and total neutron flux on all the fuel cells are simulated by the JMCT code, run with 100 cycles of 409.6 million particles each with 20 inactive cycles. Three kinds of combinations of domain decomposition and domain replication are done on 4096 CPU cores, with 1 domain ×4096 domain replications, 2 domain decompositions on assemblies ×2048 domain replications and 4 domain decompositions on assemblies ×1024 domain replications. Flux results of all fuel cells in the middle of XOY plane is shown in Figure 4. The flux results of different combinations of domain decomposition and domain replication are almost the same.

In case (a), there is only one domain and 557MB memory is used for each processor. In case (b), there are two domains in total and the two groups of all processors utilize 332MB and 345 MB memory respectively. In case (c), there are four domains in total and the four groups of all processors consume 224MB, 232MB, 230MB and 224MB respectively.



(a) (b) (c)

Figure 4. Flux results of all fuel cells in the middle of XOY plane. (a) 1 domain×4096 domain replications (b) 2 domain decompositions×2048 domain replications (c) 4 domain decompositions×1024 domain replications

Figure 5 shows the parallel efficiency for the three kinds of parallel modes, 1 domain, 2 domain decompositions and 4 domain decompositions, from 128 cores to 4096 cores. Each calculation in Figure 5 is with 50 cycles, and 50000 neutrons for each core. There are three curves in Figure 5, and each is relatively stable with the number of the cores, indicating that the scalability of combinations of domain decomposition and domain replication is very good. The '1 domain' parallel mode is just traditional particle parallel, and its parallel efficiency is greatest of all. The parallel efficiency of 2 domain decompositions is about 98% of it and 95% for 4 domain decompositions. So the time increased in domain decomposition for particle exchange is very small.

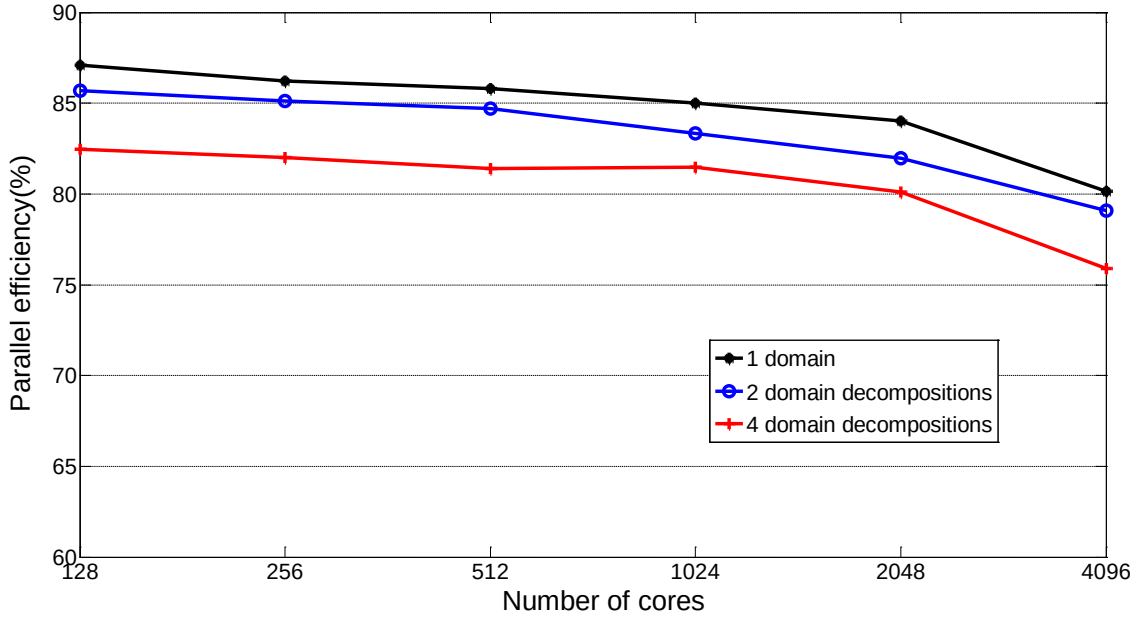


Figure 5. The parallel efficiency for three kinds of parallel modes from 128 cores to 4096 cores.

4.2 BEAVRS PWR of MIT

The MIT Computational Reactor Physics Group has released specifications for a full core PWR Monte Carlo performance benchmark BEAVRS[3]. There are 193 fuel assemblies utilizing three unique enrichments (3.1 w/o, 2.4 w/o and 1.6 w/o U235). Each fuel assembly contains a 17×17 array of pins with 264 fuel rods per assembly. The remaining 25 locations within the assembly are occupied by burnable absorber rods, guide tubes, or instrument tubes, depending on location within the core. Control rod positions are as specified for the conditions being analyzed.

The JMCT model was generated using the JLAMT tool [10]. Specification (Version 1.0.1 of the benchmark [4]) are followed exactly as no approximations are necessary in the construction of the MC21 model to match the specification.

Axially, there are 398 fuel cells in all from 36.0070 cm (bottom of active fuel) to 401.767 cm (top of active fuel) in each fuel rod, as shown in Figure 6. Because of the grid spacers, fuel cells are not the same high: the highest is 1.069 cm, and the shortest is 0.91 cm. There are 25.22 million cells in this model.

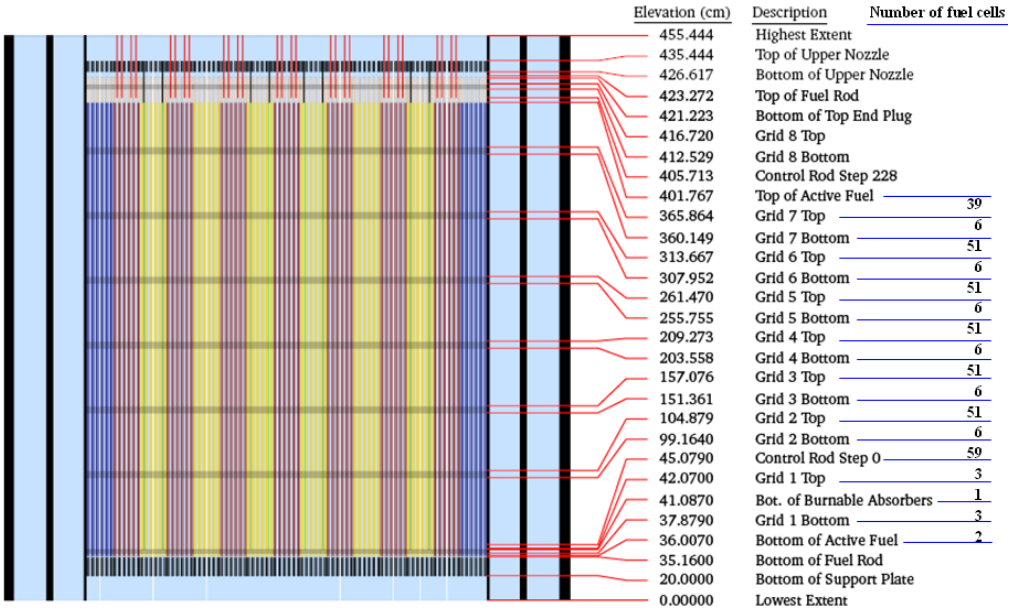


Figure 6. Scale view of axial cross section and number of fuel cells for axial planes in the active fuel region

Criticality calculation is simulated by the JMCT code, run with 1000 cycles of 81.92 million particles each with 400 inactive cycles. As the memory usage of BEAVRS model, about 14GB, is too large for a core, the domain replication can't make the calculation. Combination of domain decomposition and domain replication is done on 4096 CPU cores, with 8 domain decompositions on assemblies × 512 domain replications, as shown in Figure 7. The 8 groups of all processors consume 1.9GB, 1.9GB, 1.9GB, 1.9GB, 2.0GB, 2.0GB, 2.1GB and 2.1GB memory respectively.

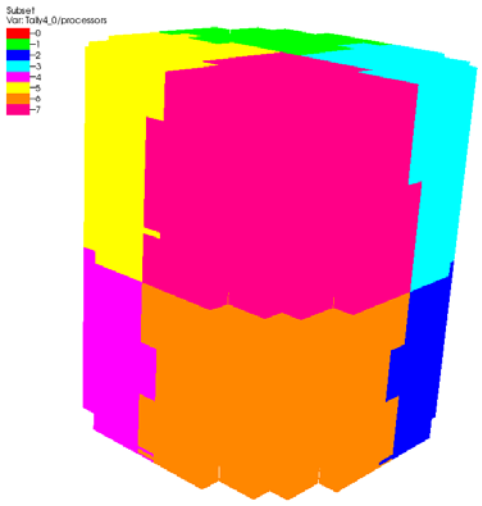
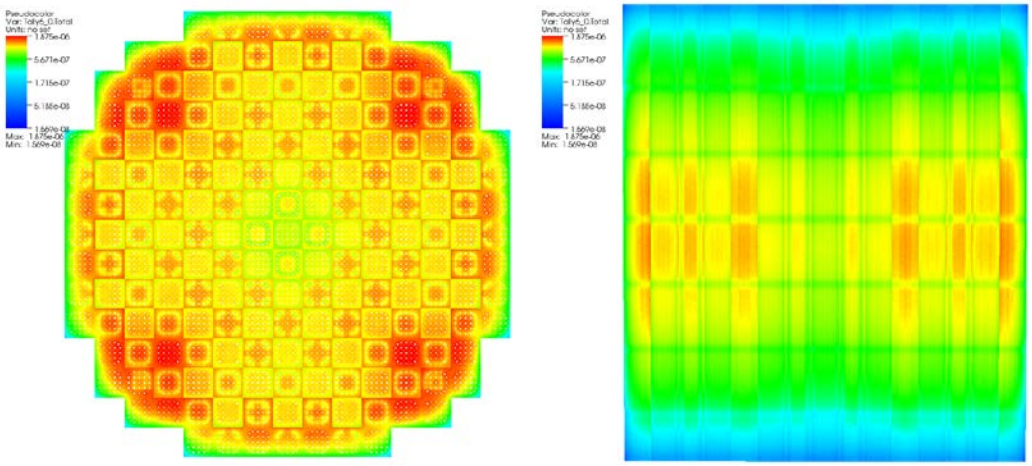


Figure 7. Results of 8 domain decomposition: the core is divided into two parts for each dimension. The domains generated automatically. When an Assembly is on the divided plane half to half, it will choose one of the two domains stochastically. So the domain boundaries are irregular.

Energy depositions averaged over a cell for all the fuel cells are got. Figure 8 shows the result of radial cross section and axial cross section in the middle of the core. Figure 8(a) looks like the results of MC21 code [11]. Figure 8(b) shows the effect of grid spacers is modeled accurately in JMCT. The 3D picture of the energy deposition results of 1/8 reactor core is shown in Figure 9.



(a) radial cross section in the middle of the core (b) axial cross section in the middle of the core

Figure 8. Energy depositions averaged over a cell for all the fuel cells

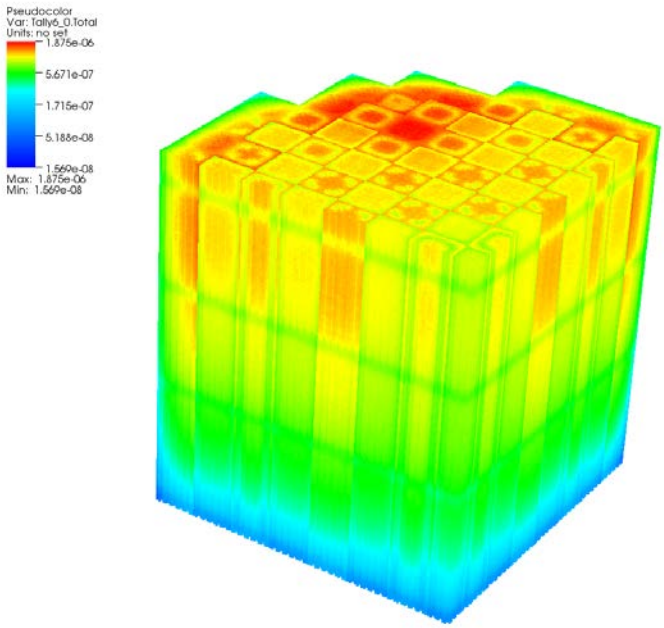


Figure 9. The 3D picture of the energy deposition results of 1/8 reactor core

5. Conclusion

The asynchronous transport algorithm for domain decomposed particle Monte Carlo in JMCT is described in details and two full-core PWR models, Dayawan reactor and BEAVRS benchmark, are simulated to verify the feasibility and correctness of the algorithm.

The domain decomposition parallelism are used when the geometry and tallies of the reactor model is much beyond the capacity of the memory. And it can also be used in combination with the domain replication.

The decomposition of the whole model is performed only according to the geometry. But in fact there are many different materials in the same geometry, which will take different time to track the particles. So the dynamic load balancing is needed to adjust the domain decomposition. This is our work in the near future.

6. References

[1] X-5 Monte Carlo Team, “MCNP — A General Monte Carlo N Particle Transport Code, Version 5-Volume I: Overview and theory,” LA-UR-03-1987, Los Alamos National Laboratory (2005)

[2] B.T. Mervin, S.W. Mosher, T.M. Evans et al, “Variance estimation in domain decomposed Monte Carlo eigenvalue calculations,” PHYSOR 2012 Knoxville, Tennessee, USA, April 15-20 (2012)

[3] F. Brown, B. Kiedrowski, D. Brown, “Advanced Monte Carlo for Reactor Physics Core Analysis,” PHYSOR2012, Knoxville, Tennessee, USA, April 15-20(2012)

[4] N.E. Horelik, B.R. Herman, B. Forget, and K.S. Smith, “Benchmark for Evaluation and Validation of Reactor Simulations (BEAVRS),” Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, Sun Valley, Idaho, May 5 – 9, on CD-ROM (2013)

[5] T.A. Brunner, T.J. Urbatsch, T.M. Evans, N.A. Gentile. “Comparison of four parallel algorithms for domain decomposed implicit Monte Carlo”, Journal of Computational Physics, 212(2) pp. 527-539(2006)

[6] T.A. Brunner, P.S. Brantley, “An efficient, robust, domain-decomposition algorithm for particle Monte Carlo”, Journal of Computational Physics, 228 pp. 3882-3890(2009)

[7] M. O’Brien, K. Joy, R. Procassini, et al. “Domain Decomposition of a Constructive Solid Geometry Monte Carlo Transport Code,” M&C+SNA, NY, USA, May 3-7, (2009)

[8] Gang Li, Baoyin Zhang, Li Deng, et al. “Combinatorial Geometry Domain Decomposition Strategies for Monte Carlo Simulations”, Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, Sun Valley, Idaho, May 5-9, on CD-ROM (2013)

[9] Baoyin Zhang, Gang Li, Li Deng, et al. “JCOGIN: A Parallel Programming Infrastructure for Monte Carlo Particle Transport”, PHYSOR 2014 - The Role of Reactor Physics Toward a Sustainable Future, The Westin Miyako, Kyoto, Japan, September 28 -October 3, on CD-ROM (2014)

[10] MA Yan, FU Yuanguang, LI Shu, "Development of Modeling Tools for pin-by-pin Precise Reactor Simulaion", Atomic Energy Science and Technology, Vol.47(Suppl.),pp:443-447, (2013)

[11] Daniel J. Kelly, III and Brian N. Aviles, Bryan R. Herman, “MC21 Analysis of the MIT PWR Benchmark: Hot Zero Power Results”, Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, Sun Valley, Idaho, May 5-9, on CD-ROM (2013)