Preliminary Performance Assessment of GPU Acceleration Module in nTRACER

Namjae Choi, Junsu Kang, Han Gyu Joo*
Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, 08826, Korea
**Corresponding author: joohan@snu.ac.kr*

1. Introduction

There have been increasing demands on more precise simulation of nuclear reactors, which raised interests on the massive parallelization of reactor analysis methods. There have been two approaches for achieving massive parallelism so far. First approach is to use thousands of CPU processors paired with high speed interconnects and apply fine-grain domain decomposition. MPACT [1] first demonstrated massive parallelization on TITAN supercomputer, and DeCART [2] and STREAM [3] also presented the application of similar approaches. Second approach is to exploit the massive processing power of graphics processing units (GPU) which are at the forefront of heterogeneous computing. OpenMOC [4] proved the high efficiency of ray tracing calculation on GPUs, and nTRACER [5, 6, 7] has also developed algorithms for GPUs.

The two approaches have pros and cons. CPU-based massive parallelization is usually easier to implement on existing codes, while migrating the codes to GPUs often requires rewriting many if not all parts of the codes. Also, GPUs are less versatile than CPUs, so dedicated tuning works are necessary to achieve high performance on GPUs. But the former is often not feasible in terms of economics. Even though there exist such systems, occupying large number of CPUs would be unavailable since the system will be shared with others. However, GPUs are power-efficient and cost-effective, and can provide massive computing power with only a few units.

This paper introduces GPU parallelization strategies of nTRACER and its performance. The strategies used in each component of nTRACER – ray tracing, CMFD acceleration, and axial sweep – are introduced briefly along with the bibliographies where the details can be found. Also, performance assessments with 2D and 3D core problems are presented. It is expected to facilitate the utilization of GPUs in reactor physics, which, we believe, will be the mainstream in the future.

2. Theoretical Backgrounds

In this paper, only a brief introduction of the methods is presented. Full details of the implementations will be explained in a compiled journal paper, or can be found in part from [4, 5, 6, 7, 8].

2.1 Ray Tracing

A GPU-accelerated ray tracing algorithm for isotropic scattering cases was first proposed by OpenMOC [4] and further extended by nTRACER [5] to anisotropic scattering cases with more consideration on utilization of CPU – GPU concurrency and mixed precision. The sweep algorithm for isotropic scattering case used in nTRACER is illustrated in **Figure 1**. Fine-grain thread-level parallelism is applied to group sweeps, and coarse-grain warp-level parallelism is exposed to ray sweeps. The GPU threading scheme is conceptually illustrated in **Figure 2**. The colors indicate rotational rays.

```
FOR G DO Group Block Sweep
  FOR r PARALLEL DO Rotational Ray Sweep
    FOR l ∈ r DO Pin Ray Sweep
      FOR g ∈ G PARALLEL DO Group Sweep
        FOR p DO Polar Angle Sweep
          Accumulate pin incoming current on register
        END DO
        Atomically accumulate pin incoming current
      END DO
      FOR s ∈ l DO Ray Segment Sweep
        FOR g ∈ G PARALLEL DO Group Sweep
          FOR p DO Polar Angle Sweep
            Save outgoing track angular flux on cache
            Accumulate angular flux change on register
          END DO
        Atomically accumulate region scalar flux
        END PARALLEL DO
      END DO
      FOR g ∈ G PARALLEL DO Group Sweep
        FOR p DO Polar Angle Sweep
          Accumulate pin outgoing current on register
        END DO
        Atomically accumulate pin outgoing current
      END PARALLEL DO
    END DO
  END PARALLEL DO
END DO
```

Figure 1. Parallel ray tracing algorithm.

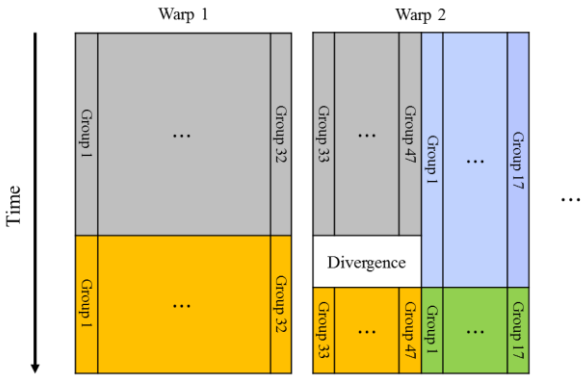

Figure 2. Ray tracing threading scheme.

The accelerated part in entire MOC sweep process is only the ray tracing part. Cross section reconstruction and source update are performed on CPUs with double precision arithmetic, and the ray tracing is performed on GPUs with single precision. To enable concurrent run of CPUs and GPUs, the energy groups are partitioned into blocks; that is, mixed precision strategy based on task parallelism is employed. The procedure is described in **Figure 3**.

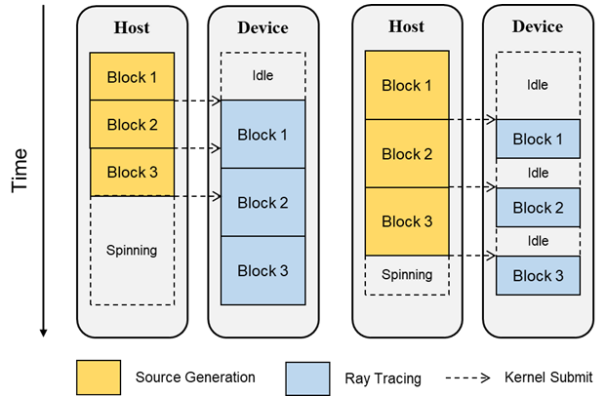

Figure 3. CPU – GPU concurrency diagram.

2.2 CMFD Acceleration

Assessment of linear system solvers and investigation of the best-suited algorithms on GPU architectures were done through the previous researches of our group [6, 7]. The conclusions of the researches were as follows:

(1) Sliced ELLPACK (SELL) matrix format is better than Compressed Sparse Row (CSR) format for highly structured migration matrices.
(2) For Krylov methods, Sparse Approximate Inverse (SPAI) type preconditioner outperforms Incomplete LU (ILU) type preconditioner.
(3) By neglecting the scattering terms of the migration matrix in SPAI preconditioning, preconditioner set-up time reduces significantly, while increase in the number of inner iterations remains marginal.
(4) Block Successive Over-relaxation (BSOR) with red-black ordering shows better performance than BiCGSTAB.
(5) Iterative refinement technique performs well within the CMFD power iteration framework.

We used linear algebra libraries in CUDA toolkit for CMFD solver implementation to ensure high degree of optimization. However, due to the lack of support for SELL format in the libraries, current implementation of GPU-based CMFD solver uses CSR format.

2.3 Axial Solver

nTRACER has established an axial transport solver for stability enhancement [8]. It employs subplane concept axially instead of resorting to conventional polynomial expansion. The iteration scheme is depicted in **Figure 4** and explained below.

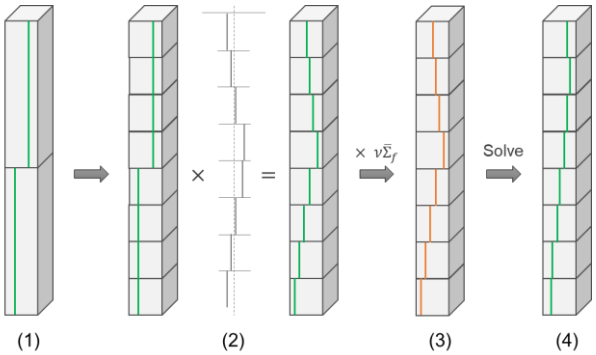

Figure 4. Axial sweeping scheme.

(1) Through the CMFD power iteration, coarse mesh average flux is obtained.
(2) The coarse mesh average flux is fed to subsequent fine meshes. Then, it is multiplied by the flux form function obtained from previous axial sweep.
(3) Using the reconstructed fine mesh flux, fine mesh fission source is evaluated.
(4) Perform fixed source iteration with axial sweeper a few times and obtain new flux form function.

Since the sweeper employs method of characteristics, the GPU acceleration algorithm is essentially same with **Figure 1**, except for the definition of the rays. And as well, the sweeping is done in single precision. The cross sections are supplied by double precision on CPUs, but source update is done on GPU with single precision.

2.4 Distributed Memory Parallelism

Basically, the GPU calculation module employs planar domain decomposition strategy; each GPU is assigned to an MOC plane. However, usual GPU clusters come out in the form that multiple GPUs are embedded in a shared memory node. Thus, each MPI process is bound to a GPU and multiple processes are launched in each node, as illustrated in **Figure 5**. Local rank of a process inside the shared memory node is found by defining a local communicator using MPI_COMM_TYPE_SPLIT routine with parameter MPI_COMM_TYPE_SHARED as the argument.

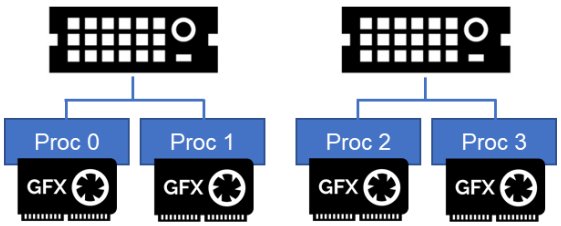

Figure 5. Distributed memory parallelization topology.

3. Results and Discussion

Runtime and accuracy comparison between CPU-based and GPU-accelerated solvers for APR1400 2D and 3D core cases are presented. The CPU-based solver is the default Gauss-Seidel solver in nTRACER. For CMFD calculation on CPU, Intel Math Kernel Library was used. The specifications of the clusters used for calculations are listed in **Table 1**. Top is the *Soochiro 3* CPU cluster in SNU, and bottom is a newly constructed GPU cluster equipped with commercial GPUs that are specialized for single precision arithmetic.

Table 1. Cluster specifications.

# of Nodes	27
CPU	2 × Intel Xeon E5-2640 v3 16 Cores, 2.8 GHz (Boost)
Memory	8 × 16GB DDR4 RAM
Interconnect	2Gbps Ethernet
Compiler	Intel Fortran 14.0.3

# of Nodes	4
CPU	2 × Intel Xeon E5-2630 v4 20 Cores, 2.4 GHz (Boost)
GPU	4 × NVIDIA GeForce GTX 1080
Memory	8 × 16GB DDR4 RAM
Interconnect	1Gbps Ethernet
Compiler	PGI Fortran 17.10

The nTRACER model of APR1400 [9] was reduced to half-3D with 16 MOC planes due to the restriction of the current GPU cluster. Thus, half-3D case requires 16 nodes or 16 GPUs. In addition, 2D core was solved for the reference to analyze the overheads coming from the 3D extension. 2D case uses single node or single GPU.

3.1 2D Quarter Core

The problem contains 47 energy groups, 20,032 pins, 1,683,860 flat source regions, and 259,628,613 ray segments. 4 polar angles were used. 4 MOC outers were required to converge, and each MOC outer consists of 4 transport sweeps; 2 sweeps on entire 47 groups, and the other 2 sweeps on 24 thermal groups. **Table 2** contains the calculation time of each component and the speedup ratio. The eigenvalues matched exactly (1.00289).

Table 2. 2D core calculation time summary.

Calculation	CPU	GPU	Speedup
Subgroup (Ray Tracing)	257.4s (253.1s)	13.2s (6.9s)	19.5 (36.7)
MOC	1218.4s	95.1s	12.8
CMFD (Ax = b)	31.0s (24.2s)	27.1s (4.5s)	1.1 (5.4)
Total	1506.8s	135.4s	11.1

The ray tracing time in MOC cannot be separated due to the concurrency, but its high efficiency can be seen through the speedup ratio of ray tracing calculation in subgroup calculation. In the meantime, the speedup in CMFD is marginal, though the speedup in linear system solving time is significant. The rest of the CMFD time is composed of cell homogenization and linear system set-up which are performed on CPU. The problem is that the generation of a CSR matrix is difficult to parallelize and is currently being done with single thread. Also, the CPU solver employs group-wise solution in which the matrix of each group can be set-up in parallel, while the GPU solver employs whole-group solution which uses a single large matrix. Slower CPU clock in GPU cluster and the PGI compiler being less efficient than the Intel compiler also serve as the causes.

3.2 Half-3D Quarter Core

Table 3 shows the computing time result of the 3D case, with all the conditions same as the 2D case. The axial solver condition is as follows: 0.5cm subplane mesh, 5 polar angles, 10 iterations per sweep, and 2 sweeps per CMFD. The eigenvalues were same as well (0.99894).

Table 3. 3D core calculation time summary.

Calculation	CPU	GPU	Speedup
Subgroup	275.9s	17.5s	15.8
MOC	1561.7s	118.0s	13.2
CMFD (Initialize) (Ax = b)	57.7s (10.9s) (27.8s)	53.5s (25.8s) (13.5s)	1.1 (0.4) (2.1)
Axial (Kernel)	197.6s (124.1s)	63.6s (15.7s)	3.1 (7.9)
Total	2092.9s	252.6s	8.3

The overall acceleration efficiency in 3D calculation was reduced, majorly due to the hardware restrictions. In 2D, all CPU cores were used by 1 GPU, but in 3D, they are shared by 4 GPUs. It results in more time spent for MOC source calculation and CMFD homogenization which are done on CPU side. Also, about 1/3 of CMFD and 3/4 of axial solver time is spent for communication, which is caused by the use of slow Ethernet network. The detrimental factors apart from the hardware are: 1) axial sweep occurs in the middle of the CMFD power iteration, which requires regenerating the linear system, and 2) CMFD routines are under-optimized. Be that as it may, the overall speedup ratio is considerable, given the amount of computation resources used in each case.

Next, the validity of mixed precision strategy was examined. The solutions obtained at full convergence (9 MOC outers) were used. **Figure 6** shows the integrated pin power difference between double precision CPU solver and mixed precision GPU solver, and **Figure 7** shows the integrated axial power difference. It can be

seen that the difference in the power only occurs at the last significant digit (10^{-4}), though the degree of error is slightly larger in the axial power than in the pin power. Therefore, we conclude that the mixed precision scheme is valid and also effective.

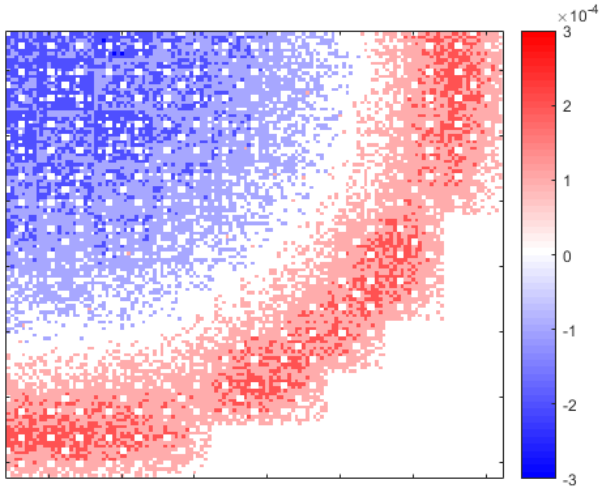

Figure 6. Absolute difference of integrated pin power.

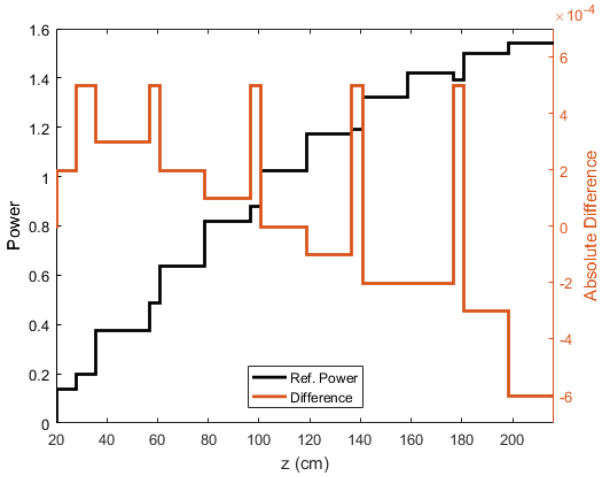

Figure 7. Absolute difference of integrated axial power.

4. Conclusion

GPU acceleration capability has been implemented in nTRACER. The modules are still under development, so there exist yet under-optimized routines, such as the host set-up routines of CMFD solver. However, with the significant rates of speedup in hotspots, total runtime reduction was remarkable as well. Also, the validity of mixed precision techniques were verified, which enables the use of commercial GPUs in scientific calculation.

Still, there exist limitations in 3D calculation, though they are mostly contributed by the hardware. First is the slowdown of host routines that comes from sharing the CPU resources. It is inevitable in the systems where multiple GPUs are contained in a single computing node, and the only way to resolve this problem is to optimize the CPU routines and try to offload more calculations on GPUs. Second is the communication load in CMFD and axial calculation. It can be somehow mitigated by optimizing CMFD algorithms in the way that minimizes communications such as applying local/global iteration strategy. However, the best way to overcome it is to use high performance networks such as Infiniband.

Remaining works are as follows: 1) incorporation of thermal-hydraulics feedback calculation to perform HFP calculations, 2) incorporation of depletion calculation to perform core follow calculations, and 3) CMFD routine optimizations. Finishing the tasks will make it available to complete a whole-core cycle depletion within 2 hours on a moderate-sized GPU cluster.

ACKNLOGEMENTS

This research is supported by National Research Foundation of Korea (NRF) Grant No. 2016M3C4A7952631 (Realization of Massive Parallel High Fidelity Virtual Reactor).

REFERENCES

[1] B. Kochunas, T. Downar, Z. Liu, “Parallel 3-D Method of Characteristics in MPACT,” International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, Sun Valley, Idaho, USA, May 5 – 9 (2013).
[2] J. Y. Cho, S. Yuk, “Massive Parallel Computation for an Efficient Whole Core Transport Calculation,” Transactions of the Korean Nuclear Society Spring Meeting, Jeju, Korea, May 17 – 18 (2018).
[3] S. Choi, D. Lee, “Efficient Parallelization Strategy of STREAM for Three-dimensional Whole-core Neutron Transport Calculation,” Transactions of the Korean Nuclear Society Spring Meeting, Jeju, Korea, May 17 – 18 (2018).
[4] W. Boyd, K. Smith, B. Forget, “A Massively Parallel Method of Characteristics Neutral Particle Transport Code for GPUs,” International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, Sun Valley, Idaho, USA, May 5 – 9 (2013).
[5] N. Choi, J. Kang, H. G. Joo, “Massively Parallel Method of Characteristics Neutron Transport Calculation with Anisotropic Scattering Treatment on GPUs,” International Conference on High Performance Computing in Asia-Pacific Region, Tokyo, Japan, Jan 28 – 31 (2018).
[6] N. Choi, J. Kang, H. G. Joo, “Performance Comparison of Linear System Solvers for CMFD Acceleration on GPU Architectures,” Transactions of the Korean Nuclear Society Spring Meeting, Jeju, Korea, May 17 – 18 (2018).
[7] J. Kang, H. G. Joo, “GPU-based Parallel Krylov Linear System Solvers for CMFD Calculation in nTRACER,” Transactions of the Korean Nuclear Society Spring Meeting, Jeju, Korea, May 17 – 18 (2018).
[8] N. Choi, J. Kang, H. G. Joo, “Stability Enhancement of Planar Transport Solution Based Whole-core Calculation Employing Augmented Axial Method of Characteristics,” Annals of Nuclear Energy. *(Submitted for publication.)*
[9] H. Hong, H. G. Joo, “Analysis of the APR1400 PWR Initial Core with the nTRACER Direct Whole Core Calculation Code and the McCARD Monte Carlo Code,” Transactions of the Korean Nuclear Society Spring Meeting, Jeju, Korea, May 18-19 (2017).