International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)
Rio de Janeiro, RJ, Brazil, May 8-12, 2011, on CD-ROM, Latin American Section (LAS) / American Nuclear Society (ANS)
ISBN 978-85-63688-00-2

EVALUATION OF SPEEDUP OF MONTE CARLO CALCULATIONS OF TWO SIMPLE REACTOR PHYSICS PROBLEMS CODED FOR THE GPU/CUDA ENVIRONMENT

Aiping Ding, Tianyu Liu, Chao Liang, Wei Ji, Mark S. Shephard, and X George Xu
Program of Nuclear Engineering and Engineering Physics,
Rensselaer Polytechnic Institute, Troy, NY 12180, USA
xug2@rpi.edu

Forrest B. Brown
XCP-3, Los Alamos National Laboratory
Los Alamos, NM, 87545, USA

ABSTRACT

Monte Carlo simulation is ideally suited for solving Boltzmann neutron transport equation in inhomogeneous media. However, routine applications require the computation time to be reduced to hours and even minutes in a desktop system. The interest in adopting GPUs for Monte Carlo acceleration is rapidly mounting, fueled partially by the parallelism afforded by the latest GPU technologies and the challenge to perform full-size reactor core analysis on a routine basis. In this study, Monte Carlo codes for a fixed-source neutron transport problem and an eigenvalue/criticality problem were developed for CPU and GPU environments, respectively, to evaluate issues associated with computational speedup afforded by the use of GPUs. The results suggest that a speedup factor of 30 in Monte Carlo radiation transport of neutrons is within reach using the state-of-the-art GPU technologies. However, for the eigenvalue/criticality problem, the speedup was 8.5. In comparison, for a task of voxelizing unstructured mesh geometry that is more parallel in nature, the speedup of 45 was obtained. It was observed that, to date, most attempts to adopt GPUs for Monte Carlo acceleration were based on naïve implementations and have not yielded the level of anticipated gains. Successful implementation of Monte Carlo schemes for GPUs will likely require the development of an entirely new code. Given the prediction that future-generation GPU products will likely bring exponentially improved computing power and performances, innovative hardware and software solutions may make it possible to achieve full-core Monte Carlo calculation within one hour using a desktop computer system in a few years.

Key Words: Monte Carlo, GPU, CUDA, neutron transport, acceleration, reactor physics.

1. INTRODUCTION

Monte Carlo simulation provides the most accurate solution for the Boltzmann neutron transport equation in inhomogeneous, 3D media such as a nuclear reactor assembly. However, the large computation time that is required to obtain results of acceptable precision has been a roadblock to wider and more routine applications in the design and analysis of nuclear reactor systems. The so-called “Kord Smith Challenge” was first proposed in an invited lecture at the 2003 ANS Mathematics & Computation Conference to bring attention to the urgent role of Monte Carlo simulation in nuclear reactor design and analysis [1]. Smith predicted that a full-core Monte Carlo calculation would take 5,000 hours on a 2-GHz PC and, based on Moore’s law, could not be performed in less than 1 hour until the year 2030 [1]. Martin re-analyzed the topic at the 2007 ANS Mathematics & Computation Conference and concluded that it would actually be 2019 before such a full reactor core calculation could be accomplished in 1 hour making use of a

predicted 1500-core processor at that time [2]. To monitor the Monte Carlo computational performances, a benchmark test model of a simplified Pressured Water Reactor (PWR) core assembly was subsequently proposed by Hoogenboom and Martin in 2009 [3] and later revised by Hoogenboom et al in 2010 [4].

Smith or Martin did not explicitly consider a number of likely improvements in computer hardware such as graphics processing units (GPUs) — a technology recently emerged with impressive capabilities in parallelization and data communication. Brown observes, during an invited lecture for the Monte Carlo 2010 Conference in Tokyo, that the existing production Monte Carlo codes such as MCNP were not optimized for large-scale parallel computing with the GPU platform and that it was prudent to develop an entirely new code that utilizes such features as vectorization [5]. A number of studies reported in 2010 have identified hardware and software issues in developing Monte Carlo applications for the GPUs [6-9]. In particular, the radiation treatment planning community has been thrilled by the possibility of performing minutes-level Monte Carlo dose calculations in a busy clinic setting. However, the reported results on GPU speedup in these studies have not met expectations and it has become clear to the authors of these studies that the implementation of Monte Carlo schemes on GPU was not straightforward [6, 8].

This paper analyzes the potentials and challenges in adopting GPU hardware and software environment for reactor physics problems. A computational experiment involving two simple reactor physics problems is performed to evaluate the speedup potentials of various GPU implementation strategies. Future GPU technologies are discussed in the context of addressing the needs for full-size reactor core Monte Carlo analysis.

2. MATERIALS AND METHODS

2.1. Consideration Of Two Reactor Physics Problems

Two types of radiation transport problems are normally investigated in reactor design and analysis: fixed-source problems and eigenvalue/criticality problems. In fixed-source problems, a neutron beam is assumed to be incident on the system of interest and the neutron reflection/transmission rates are calculated. Such problems are typical in radiation shielding design and neutron slowing down computation. In eigenvalue/criticality problems, the system is composed of a fissile material that is subjected to chain reactions. The multiplication factor and fission source distribution are calculated for the purposes of reactor core design and analysis.

In the fixed-source problem, a unidirectional neutron beam penetrates a heterogeneous system, composed of a pure scattering background material and 20 fixed-size pure absorbers. Without losing generality, a 1D slab geometry configuration with one-group radiation transport computation was considered. The absorbers are uniformly distributed in the system forming a lattice structure — a simplified condition similar to that of fuel pins at resonance neutron energies in the water moderator in Light Water Reactors. The neutron random walk process is simulated using both analog and non-analog techniques. Collision/path length estimators were employed in flux and criticality predictions. Reflection/transmission rates and flux distribution at a fine spatial resolution were calculated. For accurate comparison in terms of speed, statistical

errors were kept within 1% of relative standard deviation in each tallied quantity for both central processing unit (CPU) and GPU.

In the eigenvalue/criticality problem, a homogeneous fission system was used for algorithm comparison in calculating multiplication factor and fission source distribution. Assuming a uniformly distributed fission neutron source in the system at the first batch, a total of 200 generations of chain reactions were simulated including the first 50 generations that were treated inactive to ensure the fission source convergence. Multiplication factors were calculated using both collision and path length estimators for each generation. The fission source distribution in each generation was also tallied and used for monitoring the source convergence. Statistical errors were obtained within 1% of relative standard deviation.

2.2. The GPU Programming Environment

The GPU hardware used in this study is NVIDIA Tesla™ C2050 computing processor. Based on the new NVIDIA "Fermi" architecture, the Tesla™ C2050 with 448 scalar streaming processor (SP) cores (14 streaming multiprocessors (SM) × 32 SPs/SM) and 3-GB memory was designed for solving large-scale computing problems more efficiently. The commercial goal of Tesla™ C2050 was to make supercomputing available to a single desktop and, as such, the system supports the full use of C++ programming language. Tesla™ C2050 contains a newly available Error-Correcting Code (ECC) memory technique, meeting a critical requirement for large-scale computing accuracy. The hardware boasts a memory bandwidth of 144GB/s per GPU and delivers up to 515 gigaFLOPS of double-precision peak performance [10].

NVIDIA has developed the Computer Unified Device Architecture (CUDA) as the parallel computing engine for its GPUs. A production Monte Carlo code contains complex nuclear physics features and was not written for the CUDA environment [5]. Using CUDA, the programmer is able to allocate variables to the GPU memory, identify the thread and block index, and run the codes on the GPUs by launching the kernel codes executed in parallel by a grid of threads on the GPUs [11]. Threads on Tesla™ C2050 are grouped into blocks with a maximum of 1024 threads per block and the maximum number of resident threads per SM is 1536. Due to various hardware considerations, each block is partitioned into many warps with 32 threads per warp. All threads in the same warp execute the same instruction, called single-instruction, multiple-thread (SIMT) [11]. SIMT works the best when all threads within a warp follow the same control flow path.

For this study, CPU-based Monte Carlo neutron transport and eigenvalue/criticality codes were written in C++ and compiled with Microsoft Visual studio 2008. The CPU codes were run on a Windows Vista desktop with an Intel Xeon® E5507 Quad-core 2.26-GHz with 6-GB memory. The GPU-based codes were optimized with 256 threads per block and 48 warps per SM to keep GPU fully occupied. Such baseline Monte Carlo calculational experiments were designed to evaluate the GPU acceleration afforded by the GPUs.

3. RESULTS

After running both the benchmark CPU and GPU codes for 10^7 particles, the maximum speedup by adopting the GPUs is found to be 29 for the neutron transport problem and 8.5 for the

eigenvalue/criticality problem. It was observed that although a CUDA kernel code can run correctly on the CUDA device, its execution speed and performance can vary greatly depending on the resource constraints of the device and the threads' diverge during the execution [11]. As there is no real equivalence between GPU SMs or SPs and CPU cores, it is difficult to accurately estimate the theoretical maximum speedup of the code on GPUs. In the general Monte Carlo simulation, each neutron particle is statistically independent of the others and can be assigned to a separate thread. But its tracking and collision simulation may undergo a different path from the others resulting in a situation where the threads within a warp may take different control flow paths. In such situation, multiple passes (for example, involving the "if - else" statements) will be needed with some runtime and performance penalties [11].

The eigenvalue/criticality problem involves more "if-then-else" constructs in the kernel code to track each particle's path. For GPU, the SIMT only works well when all threads within a warp follow the same control flow path. When threads within a warp take different control flow paths, for example, our if-then-else construct, some threads execute the then parts and others execute the else part, the SIMT execution style no longer works well. In such situations, the execution of the warp will require multiple passes through these divergent paths. One pass will be needed for those threads that follow the then part and another pass for those that follow the else part. These passes are sequential to each other, thus adding to the execution time.

It should be noted that effective algorithms and coding methods for GPUs are very similar to the methods for vector computing in the 1980-90s [12,13]. Rearrangement of the Monte Carlo algorithm at a high level, using event-based logic and queues of particles awaiting common operations, proved successful for vector computers. Such methods could be adopted for the current work, to eliminate the need for multiple passes and improve the efficiency.

It is found that GPUs perform well for applications that exhibit a high level of parallelism, with the threads following the same control path. To demonstrate this, we also investigated a GPU-based code previously developed to voxelize geometries defined by unstructured triangular meshes, such as those used to describe the RPI Adult Male and Female phantoms representing adult workers [14]. Currently, production Monte Carlo codes in use cannot directly run radiation transport in such advanced and anatomically accurate phantoms, thus the time-consuming voxelization has to be performed so a production Monte Carlo code such as the MCNP5 can be used. The GPU-based voxelization algorithm was developed using parity-counting and ray-stabbing methods [15] for mesh surfaces and the voxel resolution is specified by a user.

During the voxelization process, each voxel conversion process involved the same operation: ray-stabbing number is counted on GPU and the parity-counting method was run on CPU to detect if this voxel was inside the mesh surface. All the threads followed the same control flow paths and no thread divergence existed. For this highly parallel process, the speedup by using the GPUs was found in this study to be a factor of 45.5. The speedup experimental results for double-precision floating point calculations are summarized in Table1.

Table 1: GPU speedup evaluation results

Case	Execution Time T_{CPU} (minutes)	Execution Time T_{GPU} (minutes)	Speed-up factor T_{CPU}/T_{GPU}
Neutron transport problem	~0.496	~0.017	~29.2
eigenvalue/criticality problem	4.25	~0.5	~8.5
▲ Voxelization	~2380.4	~ 52.3	~ 45.5

4. DISCUSSION

4.1. GPU Speedup Studies By Other Groups

A number of studies by other groups have identified hardware and software issues in developing Monte Carlo applications for the GPUs [6-9]. Using a desktop system of an Intel Core i7-920 CPU, 6-GB of DDR3 RAM, and an NVIDIA GTX 275 GPU with 896-MB of GPU RAM, Nelson and Ivanov [6] reported a speedup of 11 for a Monte Carlo neutron transport problem involving double-precision floating point calculations. Although they concluded that the GPUs were useful, several limitations in the GPU technologies were identified including the maximum memory of only 4GB, a lack of ECC support, and a lack of software optimization that probably caused the relatively small speedup. Tickner [7] reported a new Monte Carlo code for the GPU/CUDA environment and summarized a number of innovative approaches. The radiation treatment planning community has been quite excited about the possibility of performing minutes-level Monte Carlo calculations [8,9]. Jia et al [8] developed a GPU-based Monte Carlo coupled electron–photon transport code for radiation treatment dose calculations. Their speedup experiment involved phantoms with a water–lung–water or water–bone–water slab geometry, and a 20-MeV monoenergetic electron point source or a 6 MV photon point source. Speed-up factors ranging from 5.0 to 6.6 times were observed using an NVIDIA Tesla^{TM} C1060 GPU (1.3 GHz) card against an Intel Xeon CPU processor (2.27 GHz). This group observed limitations associated with SMIP architecture, clock speed and complex memory allocation schemes. The results on GPU speedup in these two studies are somewhat disappointing in that such level of speedup were already feasible using CPU clusters, albeit at a much greater hardware cost.

The results found from our experiment suggest that a speedup of up to 30 in Monte Carlo radiation transport of neutrons is within reach using the state-of-the-art GPU technologies. For a task of voxelizing unstructured mesh geometry that is more parallel in nature, the speedup of 45 is easily obtained. It has become clear to us that the implementation of Monte Carlo schemes that are optimized for GPU/CUDA will require a significant amount of effort, even for an application-specific Monte Carlo code. A clear approach is yet to be defined about how to take full advantage of the parallelism afforded by the GPU without suffering from any performance penalty. Obviously, this topic will be intensively investigated.

4.2. GPU Technologies In The Near Future

Despite challenges in reaching a "tipping point" in GPU-based Monte Carlo speedups, one cannot help feeling excited about what is on the horizon in terms of GPU parallel computing technologies. In 2010, the fastest six-core CPU processor had a theoretical peak performance of 107.55 gigaFLOPS (Intel Core i7 980 XE) in double precision calculations. GPUs are considerably more powerful, with NVIDIA Tesla C2050 GPUs performing around 515 gigaFLOPS. Figure 1 compares the CPU and GPU double precision performances, showing that the GPUs will continue to out-perform the CPUs in the foreseeable future [10,11].

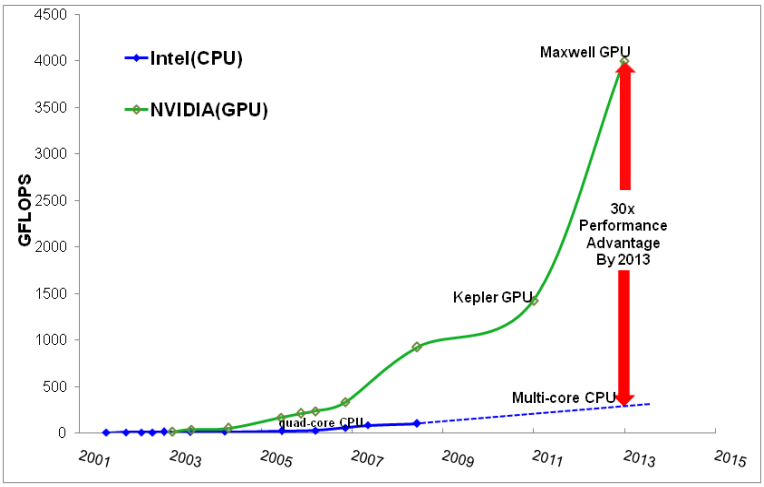

Figure 1. Comparison of CPU and GPU performances [11]

In the benchmark test model of a simplified PWR full-core model [3, 4], the number of fuel pins is about 300 while the number of fuel assemblies is around 200. With 100 different nuclides for which the reaction rate is to be calculated for 10 depletion regions and 100 axial segments for each pin, the total number of computational tallies is 6 billion. To achieve a statistical uncertainty in each local power region of 1% or less, there are clear challenges for Monte Carlo calculations in CPU time and memory.

The high performance computing industry is moving toward a hybrid computer model, where GPUs and CPUs are integrated to perform general purpose computing tasks. As parallel processors, GPUs excel at tackling large number of neutron transport histories in a Monte Carlo code that can be calculated simultaneously. As sequential processors, CPUs are adept at more sequential tasks. The world's most powerful supercomputer — the Tianhe-1 system — achieves a performance level of 2.507 petaFLOPS by harvesting the power of 14,336 CPUs and 7,168 Tesla™ M2050 GPUs. With a budget of $30,000, it seems to be feasible in the year of 2013/2014 to construct a hybrid CPU/GPU system to address the Monte Carlo computational needs for the benchmark full-size PWR core model. The future-generation NVIDIA product, Maxwell — the successor to Kepler — is intended for the high performance computing market at an affordable price. According to NVIDIA GPU roadmap revealed at NVIDIA's GPU technology conference 2010, Maxwell will have 10 to 12 times the performances per watt compared to the Fermi and bring a sixteen-fold increase on parallel GPU computing based on its new 22nm fabrication process. By 2013/2014, the 8 Maxwell GPUs integrated in one system are

expected to deliver 32 teraFLOPS in double precision calculations at much reduced cost and power consumption than the current models. Each streaming processor within the GPU can directly access the geometry and nuclear data. This avoids the use of Message Passing Interface (MPI) protocols for CPU clusters reported recently [16, 17]. Using optimized particle parallelism and compressive sensing data sampling/processing schemes [18], a miniaturized desktop supercomputer of hybrid CPU/GPU structure can, in principle, achieve a possible speed-up factor of ~1000. Therefore, it it seems to be feasible to perform a full-size PWR core analysis in less than an hour in a desktop system by 2014, provided that an optimized Monte Carlo code is available.

The alternative method to developing next generation high performance, low power, parallel computers is to employ high core count nodes in which operating system overheads are effectively eliminated on the cores dedicated to compute operations. The primary example of this type is IBM Blue Gene/Q currently targeted for massively parallel supercomputing [19]. However, the same approach could be taken in the development of desktop systems with fewer high core count nodes. High threat count parallelism will be central to the ability to effectively take advantage of potential systems of that type. The SIMT parallelism methods being used in the codes being developed in this paper are ideally suited to meet that need.

5. CONCLUSION

There is little doubt that GPU/CUDA technologies will facilitate our on-going effort in using the Monte Carlo methods for routine full-size reactor core analysis. However, naïve implementations by several groups so far have only resulted in less than 30x speedups. To take full advantage of GPU technologies, many challenges related to the hardware and software must be carefully understood and addressed. Furthermore, it must be kept in mind that some of the deficiencies and constrains in existing GPUs will likely be mitigated in future-generation products, leading to exponentially improved and perhaps unexpected computing power when compared with its CPU counterparts. For this reason, the reactor physics community shall be forward thinking and be ready to take advantage of what will be made available by the parallel computer industry in the next 5 years.

ACKNOWLEDGMENTS

Support from NVIDIA including the Tesla™ C2050 card is appreciated.

REFERENCES

1. K. Smith, “Reactor Core Methods,” Invited lecture at the *M&C 2003 International Conference*, Gatlinburg, TN, USA, April 6-10 (2003).
2. W. R. Martin, “Advances in Monte Carlo Methods for Global Reactor Analysis,” Invited lecture at the *M&C 2007 International Conference*, Monterey, CA, USA, April 15-19 (2007).
3. J. E. Hoogenboom, W. R. Martin, “A Proposal For A Benchmark To Monitor The Performance Of Detailed Monte Carlo Calculation Of Power Densities In A Full Size Reactor Core,” *International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)*, Saratoga Springs, New York, May 3-7 (2009).

4. J. E. Hoogenboom, W. R. Martin, B. Petrovic, “Monte Carlo Performance Benchmark For Detailed Power Density Calculation In A Full Size Reactor Core, Benchmark specifications, Revision 1.1, June 2010” http://www.oecd-nea.org/dbprog/documents/MonteCarlobenchmarkguideline_004.pdf (2010).
5. F. Brown, “Recent Advances and Future Prospects for Monte Carlo,” *Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2010 (SNA + MC2010)*, Tokyo, Japan October 17-21 (2010).
6. A. G. Nelson, K. N. Ivanov, “Monte Carlo Methods For Neutron Transport On Graphics Processing Units Using Cuda” *PHYSOR 2010 – Advances in Reactor Physics to Power the Nuclear Renaissance*, Pittsburgh, Pennsylvania, USA, May 9-14 (2010).
7. J. Tickner, “Monte Carlo simulation of X-ray and gamma-ray photon transport on a graphics-processing unit,” *Computer Physics Communications*, **181 (11)**, pp.1821-1832 (2010).
8. X. Jia, X. Gu, J. Sempau, D. Choi, A. Majumdar, S. B. Jiang, “Development of a GPU-based Monte Carlo dose calculation code for coupled electron–photon transport,” *Phys. Med. Biol.* **55,** pp.3077–3086 (2010).
9. P. P. Yepes, D. Mirkovic, P.J. Taddei, “A GPU implementation of a track-repeating algorithm for proton radiotherapy dose calculations,” *Phys. Med. Biol*. **55,** pp.7107–7120 (2010).
10. http://www.nvidia.com/object/product_tesla_C2050_C2070_us.html (2010).
11. David B. Kirk, Wen-mei W. Hwu, “Programming Massively Parallel Processors: A Hands-on Approach,” Morgan Kaufmann (2010).
12. W. R. Martin and F. B. Brown, "Present Status of Vectorized Monte Carlo for Particle Transport Analysis," *International Journal of Supercomputer Applications*, Vol. 1, No. 2, 11-32 (June 1987).
13. F. B. Brown and W. R. Martin, "Monte Carlo Methods for Vector Computers," *J. Progress in Nuclear Energy*, Vol. 14, No. 3, 269-299 (1984).
14. Y.H. Na, B. Zhang, J. Zhang, P.F. Caracappa, X.G. Xu, “Deformable Adult Human Phantoms for Radiation Protection Dosimetry: Anthropometric data representing size distributions of adult worker populations and software algorithms,” *Phys. Med. Biol*. **55**, pp.3789-3811 (2010).
15. F. Nooruddin, G. Turk, “Simplification and Repair of polygonal models using volumetric techniques”. *IEEE Trans. On Visualization and Computer Graphics*, **9(2)**, pp.191-205 (2003).
16. P. K. Romano, B. Forget, F. Brown, “Towards Scalable Parallelism in Monte Carlo Particle Transport Codes Using Remote Memory Access,” *Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2010 (SNA + MC2010)*, Tokyo, Japan, October 17-21 (2010).
17. D. J. Kelly, T. M. Sutton, T. H. Trumbull, P. S. Dobreff, “MC21 Monte Carlo Analysis Of The Hoogenboom-Martin Full-Core PWR Benchmark Problem,” *PHYSOR 2010 – Advances in Reactor Physics to Power the Nuclear Renaissance*, Pittsburgh, Pennsylvania, USA, May 9-14 (2010).
18. M. Mille, L. Su, B. Yazici, X. G. Xu, “Opportunities and Challenges in Applying the Compressive Sensing Framework to Nuclear Science and Engineering,” *2011 International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)*, Rio de Janeiro, RJ, Brazil, (2011).
19. The Register. IBM uncloaks 20 petaflops Blue Gene/Q super. http://www.theregister.co.uk/2010/11/22/ibm_blue_gene_q_super/ (2010).