GPU/CUDA-Ready Parallel Monte Carlo Codes for Reactor Analysis and Other Applications

Tianyu Liu, Lin Su, Aiping Ding, Wei Ji, Christopher Carothers and X. George Xu
Rensselaer Polytechnic Institute (RPI),
110 Eighth Street, Troy, NY 12180, USA
xug2@rpi.edu

Forrest Brown
Los Alamos National Lab (LANL)

INTRODUCTION

Monte Carlo simulation is widely used in reactor analysis and medical physics for its high accuracy. The time-consuming process of achieving a desired precision can be significantly facilitated by parallel computing methodologies. The exascale High Performance Computing (HPC) architecture, which is expected to arrive within this decade has the potential to further accelerate the Monte Carlo calculations. However, the ever-increasing parallelism is facing at least two challenges. One is the fact that the power consumption increases with the Floating Point Operations Per Second (FLOPS). The other is a lack of software for the emerging hardware. The first issue is addressed by seeking more energy efficient system, such as Graphics Processing Unit (GPU)-accelerated heterogeneous systems. For example, Tianhe-1A and TSUBMAME 2.0, which---ranked as the 2nd and 5th most powerful computers in the world by November 2011---consists of thousands of NVIDIA GPUs. This paper addresses the need to meet the second challenge, which is to develop a Monte Carlo code that can be effectively implemented on GPUs.

DESCRIPTION OF THE ACTUAL WORK

We developed a GPU-based Monte Carlo code for nuclear reactor analysis and medical physics. The code was tested on the state-of-the-art NVIDIA Tesla M2090, which is based on the latest Fermi compute architecture. The Tesla M2090 features up to 512 CUDA cores organized in 16 Streaming Multiprocessors (SM) with 32 cores for each SM. It has a high peak double-precision floating point performance of 665 Giga FLOPS and a high memory bandwidth of 177 Giga bytes per second. The code can be executed on both CPU and GPU platforms, which allows users to make performance comparison. The CPU programs were written in C and tested on a desktop with Intel Xeon six-core 2.8 GHz CPU and 9 GB RAM. The GPU programs were written in CUDA C 4.0 and tested on the headless (i.e. computation-dedicated) Tesla M2090.

Radiation transport simulation for neutron, photon and proton are incorporated in this code. Geometry and physics specifications are stated below.

Neutron Simulations

The effective multiplication factor, also known as eigenvalue, was calculated under two typical configurations: a bare spherical core system and a binary slab system. The 3-D bare spherical core is made of homogeneous fissile material, and the 1-D binary slab system of fuel and moderator that are distributed alternately. Our physical model adopts one group approximation and includes elastic scattering, fission and capture reactions. Fission and capture were simulated using non-analogue method. Weight-window, which is one of the variance reduction techniques, was used to guarantee all neutrons have proper weight values.

The GPU-based program was developed according to the CUDA software model, which is the scalable programming model, and the hardware model, which is Fermi compute architecture. These considerations correspond to two important undertakings in designing a GPU program, namely, thread assignment and memory allocation.

Firstly, CUDA programming model offers a thread hierarchy. At the top level of the hierarchy is grid, which represents the entirety of parallel units. The grid can be decomposed into multiple equally-shaped blocks, each of which is executed on one single SM. The block further consists of many threads that cooperate with each other. This thread hierarchy is suitable for Monte Carlo eigenvalue calculation which contains two loops. The outer one iterates over all fission source batches and was performed in sequential fashion whereas the inner one iterates over all histories in a certain batch and was performed in parallel on the GPU. In performing the inner loop, all the histories were evenly assigned to the blocks of a grid, and further to the threads of each block. It has been noted that the Monte Carlo algorithm requires several intermediate data be frequently transferred between CPU and GPU, including new eigenvalue estimates, source distribution, source weights, etc.

Secondly, frequent data transfer incurs additional latency. To mitigate this issue, a special type of CPU physical memory called page-locked zero-copy memory was used as opposed to GPU global memory. This memory has the convenience and advantage of Direct Memory Access (DMA) by the GPU. In addition, per-

thread local memory was used to store neutrons resulting from splitting technique whereby a neutron with large weight is decomposed into several neutrons with smaller weight. Besides, global memory was used to store a counter which indicates how many fission neutrons have been generated. Further, a combination of per-thread register, per-block shared memory and page-locked zero-copy memory were used to store temporary data for eigenvalues calculation.

Photon Simulations

Two cases of special clinical importance were simulated. One is X-ray imaging simulation, and the other is associated radiation dose calculation. VIP-MAN, a voxelized computation human phantom was used to represent a patient receiving a routine X-ray chest screening. The phantom consists of 147×86×470 cubic voxels, each with a side of 4mm. A square parallel x-ray beam following a continuous spectrum with 120kVp was selected as the source and placed in front of the phantom. Three types of photo-atomic interactions were simulated: photoelectric effect, Compton scattering and Rayleigh scattering. The energy of the secondary electrons was assumed to be locally deposited. Kahn's sampling technique was adopted to accelerate rejection sampling. The raw microscopic cross-section data were obtained from ENDF-VI, and all the macroscopic cross-sections, dependent upon energy and elemental composition were calculated on the spot.

Thread assignment for GPU in case of photon is straightforward: a fixed number of histories were assigned to each thread and were run sequentially on the GPU, and 256 threads were assigned to each block of grid. In memory allocation, the VIP-MAN phantom data and the relatively large microscopic cross-section data were placed in global memory, while all the other constant arrays used to calculate the macroscopic cross-sections were stored in constant memory for fast access.

Proton Simulations

As a preliminary study of charged particle, proton transport in a homogeneous water phantom was simulated. Dose in different penetration depth was calculated. Class I condensed method was used for the transport. Energy loss straggling and multiple-scattering deflection were included. Thread assignment and memory allocation are similar to the photon case.

RESULTS

In the eigenvalue calculation, the history of initial batch was set to be 16384, and a total of 1000 batches with 200 inactive batches were simulated. The speedup factor, defined as the ratio of the number of generations simulated per second by the GPU to that by the CPU was calculated.

For the spherical core system, the computing speed (number of histories simulated per second) is 5.52×10^5 and 3.11×10^6 for CPU and GPU respectively, indicating a speedup factor of 5.6. For the binary slab system, the computing speed is 5.67×10^5 and 7.17×10^6, meaning a speedup of 12.7. In addition, compared to device global memory, use of host page-locked zero-copy memory increases GPU speed by a factor of 1.2.

Test of photon and proton codes shows that CPU has a computing speed of 2.49×10^4 and 1.32×10^4, whereas GPU has a speed of 2.66×10^5 and 7.48×10^5, resulting in a speedup of 10.7 and 56.5 for photon and proton simulations respectively.

This paper shows the feasibility of accelerating Monte Carlo simulation by energy-efficient GPUs and the necessity of developing new software to accommodate the hardware change. Future work includes creating a multi-GPU system composed of up to16 Tesla M2090 GPUs, which is a slice of a larger system such as Titan at Oak Ridge National Lab, and exploring the possibility of achieving further speedup in Monte Carlo calculation.

REFERENCES

1. http://www.nvidia.com/docs/IO/105880/DS-Tesla-M-Class-Aug11.pdf (2012).
2. NVIDIA, *NVIDIA CUDA C Programming Guide Version 4.1*, Chapter 2,3,4, (2011).
3. T. LIU, et al., "Preliminary tests of a 3D Monte Carlo neutron transport code for reactor analysis using NVIDIA C2050 GPU card and CUDA environment", *Proc. PHYSOR 2012*, Knoxville, Tennessee, April 15-20, 2012, American Nuclear Society (Accepted) (2012).
4. A. DING, et al., "Evaluation of speedup of Monte Carlo calculations of simple reactor physics problems coded for the GPU/CUDA environment", *International Conference on Mathematics and Computational Methods applied to Nuclear Science and Engineering 2011*, Rio de Janeiro, Brazil, May 8-12, 2011, American Nuclear Society (2011).