Transactions of the Korean Nuclear Society Spring Meeting
Gyeongju, Korea, May 29-30, 2008

Personal Supercomputing for Monte Carlo Simulation Using a GPU

Jae-Yong Oh, Yang-Hyun Koo, Byung-Ho Lee
Korea Atomic Energy Research Institute, P.O. Box 105, Yuseong, Daejon, 305-600
tylor@kaeri.re.kr

1. Introduction

Since the usability, accessibility, and maintenance of a personal computer (PC) are very good, a PC is a useful computer simulation tool for researchers. It has enough calculation power to simulate a small scale system with the improved performance of a PC's CPU. However, if a system is large or long time scale, we need a cluster computer or supercomputer.

Recently great changes have occurred in the PC calculation environment. A graphic process unit (GPU) on a graphic card, only used to calculate display data, has a superior calculation capability to a PC's CPU as shown in Fig. 1 [1]. This GPU calculation performance is a match for the supercomputer in 2000 [2]. Although it has such a great calculation potential, it is not easy to program a simulation code for GPU due to difficult programming techniques for converting a calculation matrix to a 3D rendering image using graphic APIs [1].

In 2006, NVIDIA provided the Software Development Kit (SDK) for the programming environment for NVIDIA's graphic cards, which is called the Compute Unified Device Architecture (CUDA) [1]. It makes the programming on the GPU easy without knowledge of the graphic APIs.

This paper describes the basic architectures of NVIDIA's GPU and CUDA, and carries out a performance benchmark for the Monte Carlo simulation.

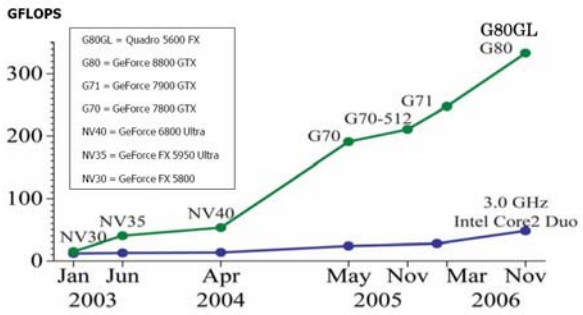

Fig. 1. Floating-Point Operations per Second for the CPU and GPU [1].

2. GPU and CUDA

The GPU on the recent NVIDIA's graphic card is implemented as a set of multiprocessors. Each multiprocessor has a Single Instruction, Multiple Data architecture (SIMD): At any given clock cycle, each processor of the multiprocessor executes the same instruction, but operates on different data [1].

When programmed through CUDA, the GPU is viewed as a compute device capable of executing a very high number of threads in parallel [1]. It operates as a coprocessor to the main CPU, or host. The batch of threads that executes a kernel is organized as a grid of thread blocks as shown in Fig. 2 [1]. Fig.2 also shows several memory types in the GPU.

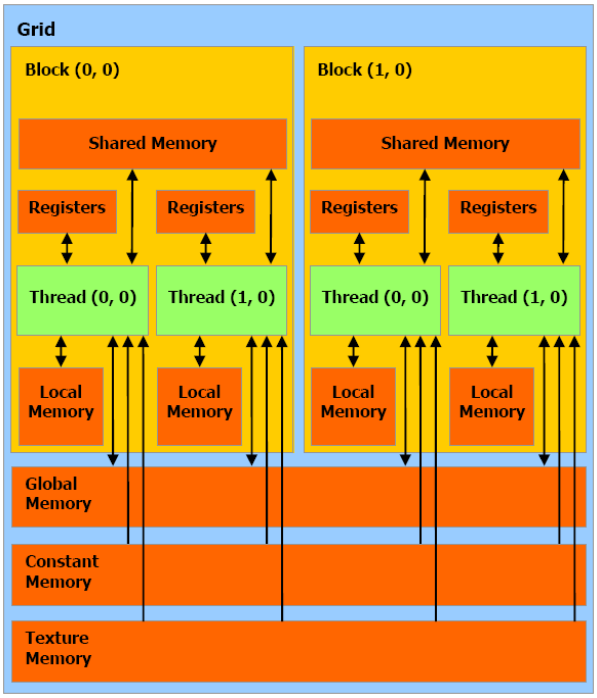

Fig. 2. Memory model [1].

3. Random Number Generation

The Monte Carlo simulation needs a large quantity of random numbers. Usually, a simple random number generator, lrand48 function, is used for the simulation in the PC. The random number generation algorithm for GPU was also proposed [3]. The benchmark shows that both random number generators produce the identical results and GPU is very faster than CPU. For the generation of 1,228,800 random numbers, the lrand48 function takes 27.13 seconds and GPU takes 2.60 seconds, which consists of the calculation time (0.14 seconds) and the data transfer time from GPU's memory to PC's memory (2.41 seconds). Since it reveals that the bottleneck for the performance improvement is the memory bandwidth for a data transfer, the performance can be improved more by maintaining all the data in the GPU's memory.

4. Microstructure Simulation

The microstructure simulation with the Potts model is chosen for the representative example of the Monte Carlo simulation. The Potts model in this paper defines a microstructure as two-dimensional 960×960 triangular lattices with a periodic boundary condition. Each site at the lattice has a state among 64 states, which represents a grain orientation.

All data necessary for the simulation are located in the global memory on the graphic card and only the final results are transferred into the PC's memory for outputs. Each thread on GPU corresponds to each re-orientation attempt and the GPU executes these threads with its stream processes in parallel.

Fig. 1 shows the performance benchmark of the GPU compared with that of Intel Q6700 CPU. As the simulation time increases, the performance increases up to 110 times. This performance enhancement allows a PC to simulate larger systems with longer time which was almost impossible in the PC.

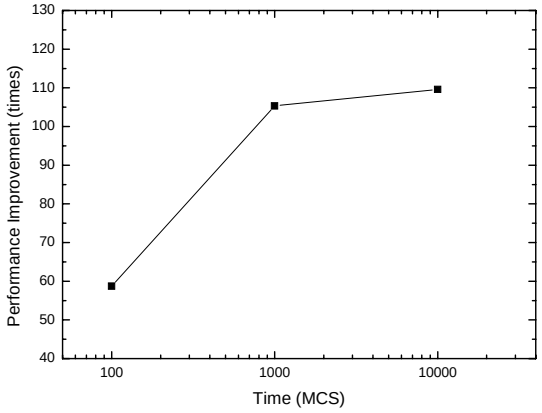

Fig. 3. Performance improvement with the GPU.

5. Conclusion

The performance benchmark shows that the GPU accelerates the Monte Carlo simulation greatly. It allows a PC to simulate larger systems with a longer time which was almost impossible in a PC.

Acknowledgments

The Ministry of Education, Science and Technology (MEST) of the Republic of Korea has sponsored this work through the Mid- and Long-term Nuclear R&D Project.

REFERENCES

[1] NVIDIA, CUDA Programming Guide Version 1.1, 2007.
[2] http://www.top500.org.
[3] J.A. van Meel, A. Arnold, D. Frenkel, S. F. Portegies Zwart and R. G. Belleman, arXiv:0709.3225