Accelerating koblinger's method of compton scattering on GPU

Jing Xie a,b,a*

aSchool of Information, XI'AN University of Finance and Economics, Xi'an, 710100, China
bSchool of Computer, National University of Defense Technology, Changsha 410073, China

Abstract

Graphics Processing Unit (GPU), originally developed for real-time, high-definition 3D graphics in computer games, now provides great faculty in solving scientific applications. The Compton scattering is an important procedure of particle transport. This paper focuses on accelerating the Koblinger's method of Compton scattering on GPU. Koblinger's method is mapped onto the thread execution model of GPU and the massive fine GPU thread mechanism is used in the implementation. The experimental results show NVIDIA M2050 GPU is 8.39-20.36 times faster than Intel Xeon X5670 and X5355 chips.

© 2011 Published by Elsevier Ltd.

Selection and/or peer-review under responsibility of ICAE2011.

Keywords: Koblinger’s method; Compton Scattering; Klein-Nishina formula; GPU

1. Introduction

Monte Carlo (MC) method is widely used to simulate the particle transport. The main advantages of the MC approach are its ability to solve complicated three-dimensional, time-dependent problems of particle transport with arbitrary precision [1]. The simple physics treatment is intended primarily for higher energy photons. The electron positron pair created as a result of pair production is treated for further transport and the photon disappears [2]. The photoelectric effect is regarded as an absorption (without fluorescence), scattering (Compton) is regarded to be on free electrons (without use of form factors), and the highly forward coherent Thomson scattering is ignored [1]:

\[ \sigma_i = \sigma_{pe} + \sigma_{pp} + \sigma_{cs} \]  

where \( \sigma_{pe} \), \( \sigma_{pp} \) and \( \sigma_{cs} \) stand for the cross section of photoelectric effect, pair production and Compton scattering. For the incident energy of particles above 1.5 MeV, the Compton scattering process is sampled exactly by Koblinger's method [3] which will be studied in this paper.

* Corresponding author. Tel. +86 150 2924 7182.. E-mail address: xaxj710@126.com..
The differential cross section for the process is given by the Klein-Nishina (KN) formula [10]:

$$K(\alpha, \mu)d\mu = \frac{\alpha^2}{\alpha + \mu^2 - 1} d\mu$$

(2)

where $r_0$ is the classical electron radius $2.817938 \times 10^{-13}$, $\alpha$ and $\alpha'$ are the incident and final photon energies in units of 0.511 MeV. $\alpha = E / (m c^2)$ where $m$ is the mass of the electron and $c$ is the speed of light, and $\alpha' = \alpha / (1 + \alpha(1 - \mu))$.

The Compton scattering technique was used to study the ground-state electronic properties of several condensed matter systems. Singhal [16] described an experiment which verified the Compton collision formula and the angular dependence of the KN formula. Within the framework of external radiation Compton model, Moderski et al. [17] presented how these effect influence the spectra of blazars for which the production of gamma rays is dominated by Comptonization of external radiation. The Compton scattering process is sampled exactly by Kahn’s method [11] below 1.5 MeV and by Koblinger’s method [3] above 1.5 MeV as analyzed and recommended by Blomquist and Gelbard [12].

The NVIDIA Compute Unified Device Architecture (CUDA) [4] programming model becomes more mature. GPU has been successfully applied to many computation intensive domains, such as random number generation [5], undetermined and determined particle transport [6-9].

As mentioned above, Compton scattering is a basic operation in the MC simulation of particle transport. Xie [13] studied the Kahn’s method of Compton scattering on GPU and gets 3.1-8.51 times speedup. In the MC computation, the PRNG is the basic. James Ticker [7] described a new algorithm, the particle-per-block technique that provides a good match with the underlying GPU multiprocessor hardware design. Heimlich et al. [6] presented a neutron transport simulation by MC method on GPU. The GPU reached a speed up about 125 times when compared to a single-core CPU solution and 14 times, when compared to an 8-core CPU solution. Gong et al. accelerated the PRNG for MCNP on GPU [5] and studied a parallel Monte Carlo benchmark on both GPU [14] and heterogeneous CPU/GPU platform [15].

2. Implement Koblinger's method on GPU

The PRNG is the basic of Monte Carlo simulation. The original serial implementation on CPU and the parallel implementation on GPU can refer to paper [6], which gives the details of the PRNG for MCNP on GPU.

The implementation of Koblinger’s method on GPU is shown in Algorithm 1. The __device__ qualifier declares a function that is executed on the device and callable from the device only. The input parameter $E_{in}$ stands for the incident energy. $E_{in}$ ranges is bigger than 3 for the energy in units of 0.511 MeV. In the real application, it needs to call the device function many times. When it comes to the parallel implementation, we should ensure not reuse the random numbers. Assuming thread1 is independent from thread2. Thread1 samples the Compton scattering with the random number set $S_1=\{r_i, r_i+1, \ldots , r_i+i0\}$, where $\{r\}$ is the ordinal random numbers. Thread2 uses random number set $S_2=\{r_j, r_j+1, \ldots , r_j+j0\}$. The seeds $r_i, r_j$ is determined by the jump function of the PRNG [5]. The property $S_1 \cap S_2 = \emptyset$ should be achieved, where $\emptyset$ is the empty set.

3. Results and discussions

The experimental platforms consist of three platforms (M2050, X5670 and X5355). M2050 consists of one NVIDIA M2050 GPU (448 CUDA cores, 1.15 GHz) and CUDA NVCC compiler version 3.2. X5670 consists of one Intel Xeon X5670 CPU (six cores, 2.93 GHz), Intel Fortran compiler version 11.1 and MPICH2 version 1.3.2p1. X5355 consists of one Intel Xeon X5355 CPU (four cores, 2.67 GHz), Intel Fortran compiler version 10.1 and MPICH2 version 1.0.7.
3.1. Experimental results

![Runtime vs Problem Size](image)

The impact of the size of CUDA thread block is shown in Fig. 1. The problem size is the number of random numbers used in the experiment. The BASE is 44001280, which is the product of 4297, 4 and 256. An optimizing option with CUDA programming model is the thread block size, which means how many GPU threads running in a GPU thread block. The execution with 128 threads per thread block outperforms that with other thread block size. For problem size 16, the runtime of 128 threads per thread block is only 0.43 seconds.
GPU threads running in a GPU thread block. The execution with 128 threads per thread block outperforms that with other thread block size. For problem size 16, the runtime of 128 threads per thread block is only 0.43 seconds.

![Graph showing performance comparisons among M2050, X5670 and X5355](image)

The performance comparisons among M2050, X5670 and X5355 are shown in Fig. 2. The CPU implementations run with all cores of the CPU chips X5670 and X5355. The performance speedup between M2050 and X5670 is up to 8.39. M2050 is 20.36 times faster than X5355. X5670 is 1.62 times faster than X5355.

3.2. Discussions

The MC simulations are regarded as inherently parallel computation which is very suitable for GPU architecture. But the M2050 is only 8.39 times faster the high end six cores X5670 for Koblinger’s method. The speedup of sampling Compton scattering with Kahn’s method is only 3.10. The three branches in sampling Compton scattering with Kahn’s method decrease the performance on GPU. Because Koblinger’s method is much more straightforward than Kahn’s method, GPU is much more suitable for Koblinger’s method than Kahn’s method.

4. Conclusions

Choosing the right algorithm and efficiently mapping it to the underlying hardware has always been important in high-performance computing. In this paper, sampling Compton scattering by Koblinger’s method is accelerated on GPU. It has been implemented with the CUDA programming model. The advantage of Koblinger’s method on GPU has demonstrated on NVIDIA M2050 GPU compared with two Intel CPUs.

Acknowledgements

This research work is supported by the Science Research Plan in XI’AN University of Finace and Economics (Program No.10XCK29), by the National Natural Science Foundation of China under grant No.60970033, also by 973 Program of China under grant No.61312701001.

References


