Evaluation of CUDA for Multi-Energy X-ray Image Reconstruction

Dae-Young Park and Seung Eun Lee*
Dept. of Electronic Engineering
Seoul National University of Science and Technology, Seoul, Korea
*e-mail: seung.lee@seoultech.ac.kr

Abstract. High-performance computing capability enabled the development of a multi-energy X-ray imaging system which requires image reconstruction from multi-energy images. Usually, parallel processing of CUDA is faster than CPU. However, the CUDA could not be taking advantage depending on kind of program. In this paper, we investigate the performance of the CUDA for parallel processing in order to evaluate the potential use of it for the multi-energy x-ray image reconstruction.

Keywords: X-ray imaging; GPGPU (General Purpose Graphics Processing Unit); Single Photon Counting; High-performance computing; Image reconstruction

1 Introduction

A type of X-ray imaging based on photon counting, instead of charge integrating has been introduced offering improved image quality compared to the former devices [1]. Single Photon Counting (SPC) detector provides an accurate representation of the beam hardening effect by assigning the same weight to the all the detected photons, leading to a higher but more correct expression of the beam hardening effect [2]. Measuring low energy photons individually by counting the number of phonons with various thresholds increases the image contrast.

In a SPC X-ray system, the digital pixel sensor is required to (a) acquire the charge of each individual event sensed corresponding its pixel, (b) classify the total energy of each incoming charge packet into one of selectable energy bins, and (c) count every charge by energy. Then, the counter values for each energy bin are forwarded to the processing unit for image reconstruction. Finally, reconstructed images are overlaid by using visualization software. This requires high-performance computing in image reconstruction step. We have evaluated the performance of CUDA compared to the CPU, by using non-weighted adding function [3]. In this paper, we evaluate the computing time of the weighted addition which is the mandatory function in the SPC X-ray image reconstruction.

* Corresponding author, e-mail: seung.lee@seoultech.ac.kr, Tel: +82-2-970-9021

© Elsevier, 2012
2  Backgrounds

In recent years, developers have been finding effective methods improving processing speed in the image processing area. NVIDIA’s CUDA (Compute Unified Device Architecture) provides a massively parallel processing. Thanks to the improved processing performance, scientists and researchers are using the CUDA for image and video processing, medical engineering, computational biology and chemistry, fluid dynamics simulation, ray tracing and much more [4].

GPUs (Graphics Processing Units) are increasingly developed using parallel architecture. The GPU has moved from being used solely for graphical tasks to the whole process of computation. GPGPU (General-Purpose computing on Graphics Processing Units) is extensively used for signal processing, using the GPU instead of CPU for floating point operation[5].

![Fig 1. Sequential Execution with CUDA](image)

Figure 1 shows the CUDA execution model where a host operates serial processing. In case of repeatedly operating an algorithm, the data to be processed transfer from host(CPU) using serial processing to device (GPU, CUDA) using the parallel processing by calling kernel function to start parallel processing on the device (CUDA). The kernel function makes the grid which is composed of many blocks of threads on the device. A thread is a sequence of instructions that can be executed on different data units in parallel. The data completed by the process on device are returned to host for serial operation. Such a sequence is repeated by the process until the program is completed. The CUDA is expected to offer improving performance by the parallel process rather than the serial process.

3  Performance Evaluation

3.1  Experimental Introduction and Expectation

In this paper, we implement that the three vector matrices are added in each of the CPU and CUDA for comparing computation speed. In order to measure the processing speed on CPU and CUDA, the experiments are performed in various
environments. 1) We make three vectors (e.g. \(a_{ij}, b_{ij}, c_{ij}\)) composing of per each 64×64, 128×128, 640×480, 800×600, 1024×768, 1280×1280 to measure the processing speed according to the size of the matrix. 2) It is separated by thread 512 per block and thread 1024 per block to measure processing speed according to the number of threads. 3) Elements of the three vectors are weighed (e.g. \(w_1, w_2, w_3\)) to measure processing speed according to the complexity of the operation. 4) We program functions which are composed of two types: the first type is non-weight adding function (1) and the second type is weighted adding function (2) as follows:

\[
F_0_{ij} = a_{ij} + b_{ij} + c_{ij}. \quad (1)
\]
\[
F_1_{ij} = w_1 a_{ij} + w_2 b_{ij} + w_3 c_{ij} \quad (2)
\]

3.2 Experimental setup

The experiments were run on a CPU Intel i7 2600K 3.4 GHz Quad-Core, L1 Cache 4 x 32KB / 2 x 32KB, 8GB DDR3 1.333GHz of main memory, and PCI-E 2.0. The GPU is NVIDIA GeForce GTX 460 SE with global memory 993 Mbytes and GPU clock speed 1.46GHz, L2 cache size 524288bytes, maximum number of threads per block are 1024. It has 6 multiprocessors, each multiprocessor has 48 CUDA cores, resulting in total of 288 CUDA cores. Windows 7 professional is used as an Operation system. Developer tool is Microsoft Visual C++ 2008. Table 1 summarizes the experimental setup.

<table>
<thead>
<tr>
<th></th>
<th>CPU(host)</th>
<th>GPU_CUDA(device)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model name</td>
<td>Intel i7 2600K</td>
<td>NVIDIA GeForce GTX 460 SE</td>
</tr>
<tr>
<td>Clock speed</td>
<td>3.4 GHz</td>
<td>1.46GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>8GB DDR3(RAM)</td>
<td>993 Mbytes(global memory)</td>
</tr>
<tr>
<td>Operation System</td>
<td>Windows 7 Professional K</td>
<td></td>
</tr>
<tr>
<td>Developer tool</td>
<td>Microsoft Visual C++ 2008</td>
<td></td>
</tr>
</tbody>
</table>

3.3 Experimental Result and Discussion

Figure 2 shows CPU and CUDA processing time without data transfer time between the host and the device. When vector matrix size is 64×64, CUDA is slower than CPU. When vector matrix size is 128×128, the data operated in CPU and CUDA are processed as similar processing time. After 640×480 of vector matrix size, the CUDA processing time is almost constant but the CPU processing time noticeably increase. The processing time of the CPU increases for the larger matrix, however CUDA’s processing time is little affected by the matrix size.
The thread number per block does not affect to the processing time. Although thread numbers per block are increased, the processing time does not decrease noticeably because of the small problem size compared to the core number of CUDA.

In order to evaluate the performance benefit of CUDA, more complex operation as shown in equation (2) was applied to CPU and CUDA. The operation is base function in which we are interest for the x-ray image reconstruction.

Figure 3 shows the processing time for the weighted addition. The complicated operation increases the processing time of CPU. However, CUDA processing time is no great difference compared to Fig. 2. From the experimental results, we found that parallel processing on CUDA could outperform the CPU for the complicated problems and adopting CUDA for x-ray image reconstruction is reasonable.

![Fig 2. Computing Time of the non-weighted addition varying the resolution](image1)

![Fig 3. Computing Time of the weighted addition varying the resolution](image2)

4 Conclusion

In this paper, we investigated the performance of the CUDA for parallel processing with two functions in order to evaluate the potential use of it for our x-ray image reconstruction. In the case of only processing time of the CUDA without the data transfer time, CUDA outperforms CPU for the larger problems. In this experiment, the CUDA processing time is nearly constant. However, the CPU processing time is
steadily increased according to the matrix size. Assuming that matrixes are the multiple emerged images, adopting CUDA for the image reconstruction step is the feasible option in SPC x-ray imaging system design.

**Acknowledgments.** This work was financially supported in part by Seoul National University of Science and Technology and the Korean Ministry of Knowledge Economy and KEIT under contract No. 10038695.

**References**