CPU AND GPU PERFORMANCE ANALYSIS ON 2D MATRIX OPERATION

Kwek Benny Kurniawan[1], YB. Dwi Setianto[2]

[1,2]Program Studi Teknik Informatika Fakultas Ilmu Komputer, Universitas Katholik Soegijapranata

[2]setianto@unika.ac.id

Abstract

GPU or Graphic Processing Unit can be used on many platforms in general GPUs are used for rendering graphics but now GPUs are general purpose parallel processors with support for easily accessible programming interfaces and industry standard languages such as C, Python and Fortran. In this study, the authors will compare CPU and GPU for completing some matrix calculation. To compare between CPU and GPU, the authors have done some testing to observe the use of Processing Unit, memory and computing time to complete matrix calculations by changing matrix sizes and dimensions. The results of tests that have been done shows asynchronous GPU is faster than sequential. Furthermore, thread for GPU needs to be adjusted to achieve efficiency in GPU load.

Keywords: CUDA, GPU, CPU, Parallel

Introduction

GPU or Graphic Processing Unit can be used in many platforms such as smartphones, game consoles, computing and workstations, GPUs are generally used for rendering graphics but now GPUs are general purpose parallel computing platform and programming model that leverages the parallel compute engine in NVIDIA GPUs to solve many complex computational problems in a more efficient way than on a CPU [1].

Compute Unified Device Architecture (CUDA) is a soft- ware platform for massively parallel high-performance computing [1], [2]. By using CUDA, CUDA will facilitate the programmer because it is supported by the use of language c. CUDA is usually used for graphical programming such as digital and video image processing such as image segmentation, image quality change, pattern recognition that can be applied to real-world needs [3]. This project will compare the effect of matrix dimension length to matrix operation and give analysis result used two platform such as CUDA and Java.

Research Method

This project will compare CPU and GPU performance. Author used two processors and one GPU. GPU platform that is used in this project is CUDA, and CPU platform that used in this project is Java.

To compare CPU and GPU Author will try to test GPU and CPU with the different case such as:

1. The effect of matrix dimension with time computation in CPU and GPU. (in CPU will test in different processing unit)

2. The effect of matrix dimension with processing unit usage in CPU and GPU (in % and GPU used 2 Thread to compare [2], [4])
3. The effect of matrix dimension with memory usage in CPU and GPU (in % and GPU used global memory [1], [5], [6])
4. The effect of time on the number of matrix elements in performing different tasks on GPU.

The effect of matrix dimension with time computation, processing unit usage and memory usage in this project tested 4 types of operation matrices with several dimensions of the matrices, and test performance used software monitor GPU and CPU usage memory and time computation in several times and write the analyze compare CPU and GPU used graph.

Results and Analysis

To determine the effect of matrix dimension on computation time, memory usage, use of PU and the influence of matrix length, program testing is matrix, addition and flipping matrix. Testing has been done 3x to get best result and every operation is enhanced with long dimension of 500x500, 1000x1000, 5000x5000 and 10000x10000 tested to 2 processors is dual core E2140 and i5 4460, GPU using 1050Ti.

Below is the results of multiplication effect studies, additions and flipping matrices on different matrix dimensions and explain the results.

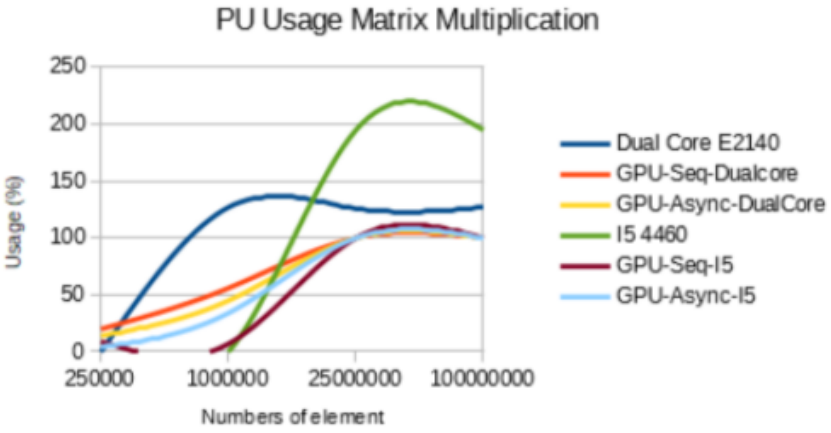

Figure 1: Processing Unit Matrix Multiplication

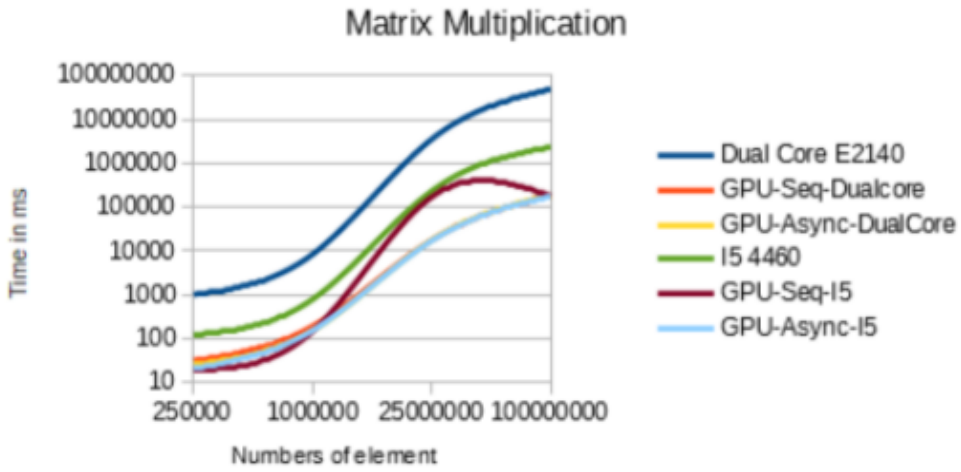

Figure 2: Time Computation Matrix Multiplication

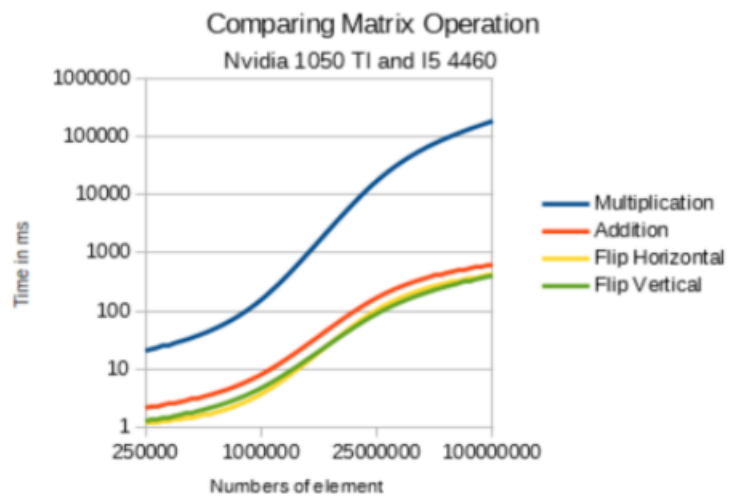

Figure 3: Comparing Matrix Operation

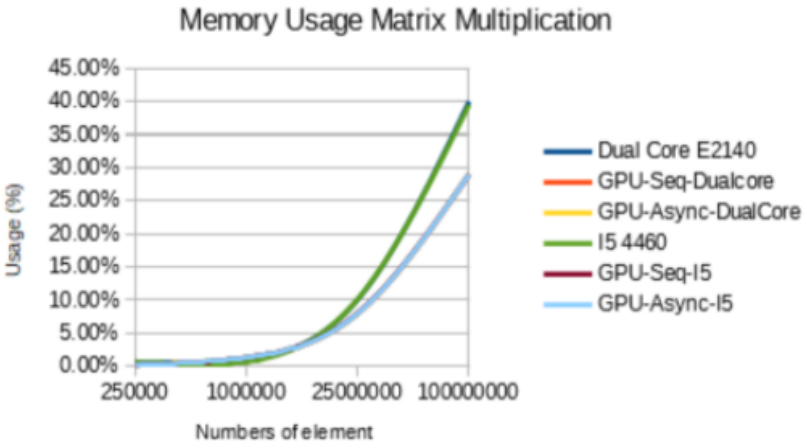

Figure 4: Memory Usage Matrix Multiplication

The graph above shows the multiplication of the computing time matrix on the GPU and CPU has a significant difference. Processor usage tends to be lower than GPU CPU. Memory usage looks quite stable with many spikes and the comparison results show linear results for each given job.

Conclusion

1. The effect of matrix dimension with time computation in CPU and GPU (in CPU will test in different processing unit).

In this work after doing experiment with matrix multiplication, addition and flip (horizontal and vertical) with different matrix size from 500x500, 1000x1000, 5000x5000 and 10000x10000 and monitoring time computation. Time computation showing that GPU matrix multiplication take faster than CPU matrix implementation for data more than 20m above.

2. The effect of matrix dimension with processing unit usage in CPU and GPU (in %). In this work after doing experiment with matrix multiplication , addition and flip (horizontal and vertical) with different matrix size from 500x500, 1000x1000, 5000x5000 and 10000x10000 and monitoring processing unit usage. CPU utilization classified near 200% but in GPU utilization showing 100% but it is can be lower by maximum thread use in because in this project only using 2 Thread to solve.

3. The effect of matrix dimension with memory usage in CPU and GPU (in %). In this work after doing experiment with matrix multiplication , addition and flip (horizontal and vertical) with different matrix size from 500x500, 1000x1000, 5000x5000 and 10000x10000 and monitoring memory usage. Memory to use is directly proportional and look linear.

4. Effect of time on the number of matrix elements in performing different tasks on the GPU. From the test results above shows the same GPU but in use in different PU generate different computational time.

References

[1] NVIDIA, “CUDA C Programming Guide,” no. December, 2016.

[2] T. R. Halfhill, “Parallel Processing with CUDA,” Microprocessor Report, pp. 1–8, 2008.

[3] B. Kurniawan, T. B. Adji, and N. A. Setiawan, “Analisis Perbandingan Komputasi GPU dengan CUDA dan Komputasi CPU untuk Image dan Video Processing,” Seminar Nasional Aplikasi Teknologi Informasi (SNATI), vol. 1, no. 1, pp. 25–31, 2015.

[4] V. Volkov, “Better performance at lower occupancy,” Proceedings of the GPU Technology Conference, pp. 1–75, 2010.

[5] Khoirudin and J. Shun-Liang, “GPU application in CUDA memory,” Advanced Computing: An International Journal, vol. 6, no. 2, pp. 1–10, 2015, doi: 10.5121/acij.2015.6201.

[6] N. Corporation, “NVIDIA CUDA Architecture,” Compute, no. April, 2009.NVIDIA, “CUDA C Programming Guide,” no. December, 2016.