Analysis of Parallel Multicore Performance on Sobel Edge Detector

NOOR ELAIZA ABDUL KHALID, SITI ARPAH AHMAD, NOORHAYATI MOHAMED NOOR, AHMAD FIRDAUS AHMAD FADZIL, MOHD NASIR TAIB
Faculty of Computer Science and Mathematics
Faculty of Electrical and Electronic Engineering
University Technology MARA
Shah Alam, Selangor
MALAYSIA
elaiza@tmsk.uitm.edu.my,arpah@tmsk.uitm.edu.my,noorhayati@tmsk.uitm.edu.my,ahmadfirdausfadzil@gmail.com,dr.nasir@ieee.org

Abstract: - This paper presents the parallel multicore Sobel edge algorithm which parallelizes the traditional sequential Sobel edge detection algorithm on a parallel multicore platform. The current advancement of multicore architecture can be utilized by the parallel programming paradigm when focuses on the thread operations. The CPUs/cores provide more processing resource to be used but often not fully utilized to its utmost potential. In this paper, the performance of multicore architectures, combined with the parallel communication software named Message Passing Interface (MPI), on the application of Sobel Edge detector algorithm is implemented on various thread processing is performed and analyzed. The test is being done on ten different images with each image tested in the varying size of 1Kx1K, 2Kx2K, and 3Kx3K pixels. The significant performance increment is discovered due to the fact that the CPU is being fully utilized. This proves that the current hardware is far more underutilized than one would expect.

Key-Words: - Parallel Programming, Sobel Edge Detection, Message Passing Interface (MPI), Multicore.

1 Introduction

Parallel programming or parallel computing is an alternative towards the traditional serial computing which instead of only allowing a single instruction to be executed one at a time, it simultaneously execute multiple instruction at once using multiple computational resources [1]. High-end computing is one area that sought tremendous amount of processing powers which most computer system having difficulties to accomplish. This is where parallel programming plays it role. From solving complex mathematical equations to assisting scientist in research, parallel programming has proven the world it's worth [2].

The advancement of processor technology had produces the multicore computer thus creates a challeges to software engineering communities in utlizing it. One of the way is by applying parallel programming on the platform. Utlizing the multicore by parallel programming is a challenge due to its complex interacting facet of performance based on memory consumption, processor utilization and synchronization and communication costs [2]. Trade-off have to be made among these facets to achieve the goal of better performance and utilization. But in implementing better parallel solution certain conceptual and programming challenges have to be investigated further.

Image processing is one of the areas that sought tremendous processing prowess. The image that required to be processed is often very large but the processing needs to be fast [3]. When such cases happen, the most imminent solution is by adapting for a new and advance hardware that is capable to accommodate the processes, without realizing whether or not the current hardware is fully utilized to its utmost potential [4].

Bearing these issues in mind, the better solution towards solving this problem is by fully utilizing the current multicores hardware, to fulfill its utmost potential and subsequently creating a system that is able to accommodate the massive processing requirement [5]. One way to exploit the multicore

architecture is by parallelization the operation of multiple threads on different cores using parallel communication software named MPI and examined the performance.

This paper aims to analyze the performance of multicore architecture on the application of Sobel Edge detector implemented on various thread processing via the parallel programming paradigm. This paper had been organizes as follows. In section 2, the methodology is described in detail. Section 3 elaborates the results and analysis of the finding. Finally section 4 is the conclusion.

2 Material and Method

2.1 Material

Ten images of size 3Kx3K pixels are used as the initial images. These images are then resized using imaging tools into smaller dimensions of 1Kx1K and 2Kx2K pixels. The images are colored and roughly are scenery, animals and part of car images. The sizes of the imagers are modified to the specified sizes using Adobe Photoshop software.

2.2 Sobel Edge detector

The edge detection algorithm named Sobel, that had been used in this work is a well known and established algorithm for detecting an edge in an image [6]. The algorithm aims to identify points in a digital image at which the image brightness changes sharply or more formally has discontinuities [6]. This algorithm has significant parallelism since it operates at pixel by pixel level. Edge detection is one of the central tasks of the lower levels of image processing which exhibit the need to program in parallel [7]. Sobel edge detector is based on the mathematical equation as given below:

$$|G| = \sqrt[2]{Gx^2 + Gy^2} \qquad (1)$$

where *x* and *y* is the convolution kernels that is in a form of 3x3 mask. Figure 1 illustrates the convolution kernels that usually used for Sobel operator [5].

-1	0	1
-2	0	2
-1	0	1

0°

-1	-2	-1
0	0	0
1	1	1

90°

Figure 1: Convolution Kernel

Despite the simple appearance of the formula which looks fairly simple to calculate, in terms of programming it involves a huge number of iterations within the program in to finish the operation.

2.2 Parallel Architecture and Software Design

Parallel manner is more complicated. There are two ways of parallelization, data or task parallelism. This work used data parallelism as a digital image can be split into several parts to be processed by two processor's of duo and quad core.

2.1.2 Hardware and software

The multicore processors specification used in this work are duo core and quad core and is described in Table 1.

Table 1: Hardware Used

Component	Description	
# of Processor Cores	2 (Duo)	4 (Quad)
Processor	Intel® Core™ Duo E7500	Intel® Xeon® E5420
RAM	3.46 GB DDR-2 RAM	3.46 GB DDR-2 RAM

The software required to perform the parallel process are Windows XP, Microsoft visual studio,. Net Framework. MPICH 2 parallel communication software and CPU Monitoring software for performance measure as shown in Figure 2.

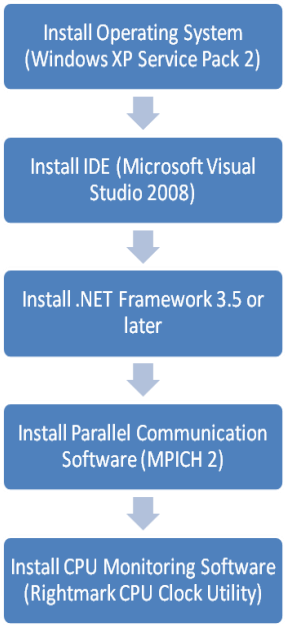

Figure 2: Sofware Installation Flow

ISBN: 978-1-61804-019-0

The parallel communication software used in this project is called the message passing interface or MPI. MPICH 2 is the software that enables message passing in parallel system.

The main role that the message passing interface (MPI) plays in this project is to execute programs in multiple threads, thus enabling all the central processing unit (CPU) to utilize each and every single core in order to accommodate the multiple threads execution. In this paper, we used 2 threads, 4 threads, 6 threads, 8 threads and 10 threads.

2.1.2 Data Parallelization Process

There are two ways of parallelization, data or task parallelism. Data parallelism looks to be the simplest and feasible solution in this case, as a digital image can be split into several parts to be processed by different processor's core. The sequential versus parallel process design are depicted in the figure 2.

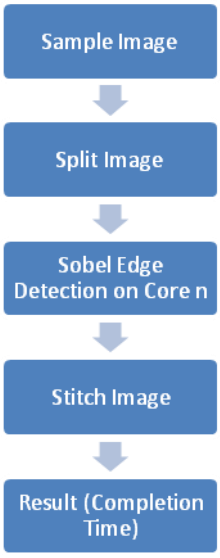

Figure 3: Data parallelization Model

The data parallelism is achieved by splitting a single image into 2,4,6,8 or 10 parts which correspond to 2,4,6,8 and 10 threads used. Example of image splitting into two part is depicted in Figure 4. After splitting the image, both parts of the image will undergo Sobel edge detector algorithm. Then each of the partition is executed in individual threads on the different core. Figure 3 illustrates how the split image looks after the Sobel edge detection algorithm is applied and how the parts are stitched back together upon the completion of the operation

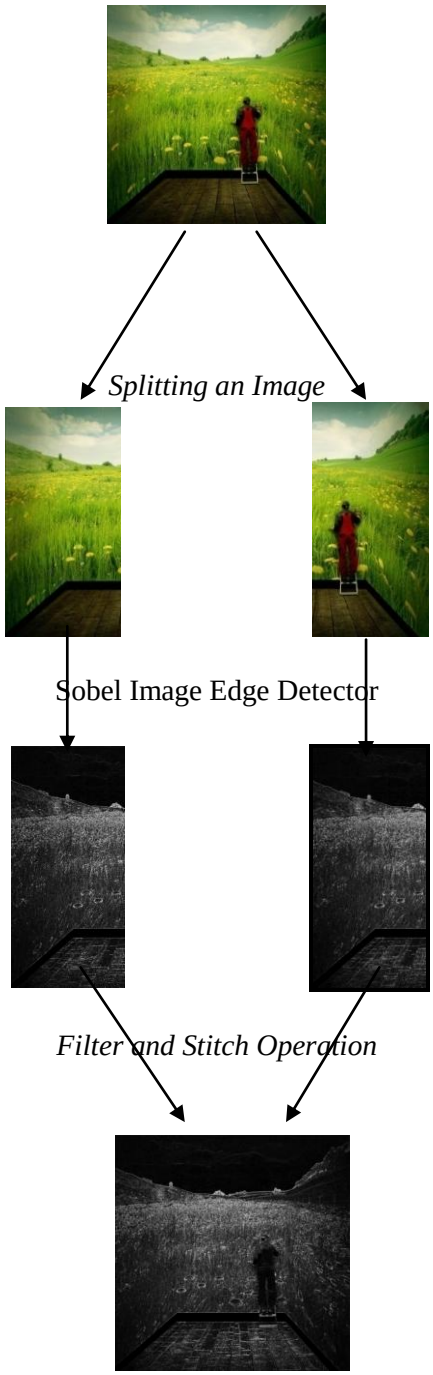

Figure 4: Data splitting and stitching architecture

.

2.1.3 Performance Evaluation Method

The evaluation of the parallel execution performance is measured with respect to speedup, performance improvement and efficiency with reference to the time taken for both sequential and parallel processing [3].

Speedup measures how much a parallel algorithm is faster than a corresponding sequential algorithm. The speedup calculation is based on Equation 1;

$$\text{Speedup} = \frac{\text{sequential(time)}}{\text{parallel(time)}} \quad (1)$$

The performance improvement depicts measurements relative improvement that the parallel system has over the sequential process. This performance is measured based on Equation 2;

$$\text{Performance Improvement} = \frac{\text{sequential(time)} - \text{parallel(time)}}{\text{sequential(time)}} \quad (2)$$

Efficiency is used to estimate how well-utilized the processors are in solving the problem, compared to how much effort is wasted in communication and synchronization. As for efficiency, the calculation is based on Equation 3;

$$\text{Efficiency} = \frac{\text{sequential(time)}}{\text{no of processor x parallel (time)}} \quad (3)$$

3 Results and Discussion

The results discussion are divided into the performance of sequential process and parallel processes in multicore processor

3.2 Sequential processing Speed

Figure 2.1 shows the time taken to process the ten images of 1Kx1K, 2Kx2K and 3Kx 3K .

Table 2: Sequential Result on Intel XEON E5420 and Intel CORE 2 Duo E7500

	1Kx1K		2Kx2K		3Kx3K	
	Quad Core	Duo Core	Quad Core	Duo Core	Quad Core	Duo Core
1	3.52	3.20	13.39	11.94	29.05	25.98
2	3.44	3.10	13.14	11.70	29.02	25.87
3	3.42	3.06	13.03	11.63	28.53	25.44
4	3.39	3.04	12.93	11.56	28.12	25.26
5	3.51	3.17	13.25	11.85	28.91	25.79
6	3.46	3.10	13.19	11.76	29.25	26.17
7	3.48	3.12	13.34	11.92	29.42	26.36
8	3.48	3.11	13.29	11.88	29.46	26.29
9	3.39	3.05	12.92	11.66	28.78	25.78
10	3.54	3.14	13.48	12.07	30.07	26.68

As expected the larger the image the higher the processing time. It is found that the quad core require more time to process Sobel edge detector compared to the Duo core. This could be explained by the overhead time needed to manage the duo core is lesser than the Quad core.

3.1 Parallel Results

The Parallel results are discussed based on speedup, performance improvements, efficiency and *CPU Utilization Factor Using the Algorithm* in performing the algorithm.

3.1.1 Speedup

The speedup results between the utilization of 2, 4, 6, 8 and 10 thread between Quad and Duo CPU/core can be observed in Figure 6. There speedup values for almost all the images processed with duo core are close to two which indicates reasonably good performance whereas the quad core shows the speedup values of around 3.5 thus showing slightly less efficient use of all the processors. Two threads process work quite efficiently with the Duo core which reduces slightly with the increased in threads used. However as the number of thread increases, the more efficient the Quad core performance with the exception of ten thread. Thus this indicates that two thread is best used for Duo core and eight threads are suitable for Quad core.

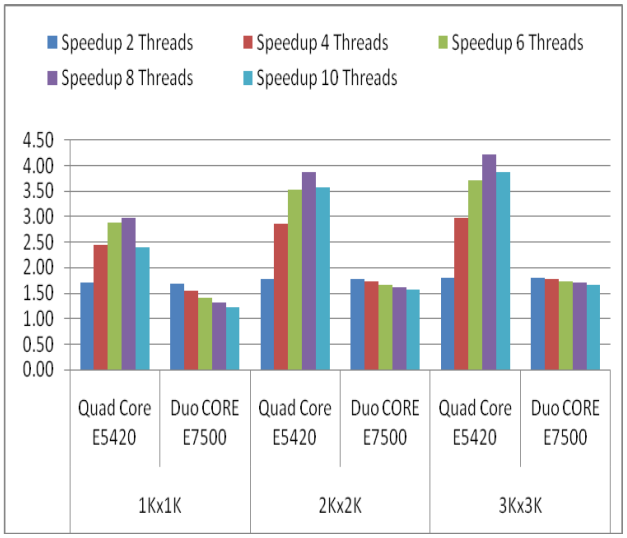

Figure 6: Histogram of Comparison for Speedup of Different threads

3.1.2. Performance improvement Index

The performance improvement index discusses the

performance of multicore and multithreading process. The histogram of the performance improvement index of Duo and Quad core are depicted in Figure 7. The performance significantly improved for duo core from image with 1Kx1K to 2KX2K, however it does not indicate much improvement between the 2Kx2K and 3Kx3K. It is also found that there is an average of about 0.45 for the duo cluster compared to the quad core process which is mostly above 0.70 for images of 2Kx2k and 3Kx3K. The performance of the Duo core reduces for all case as the number of thread increases however the performance increases steadily with the exception of 10 threads for the Quad core. This indicates the best performance for Duo core is two threads and while eight thread is best used for Quad core compared to the sequential process.

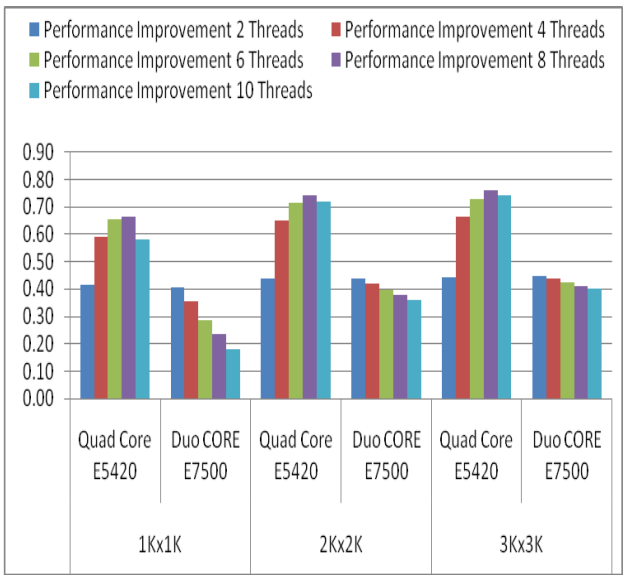

Figure 7: Histogram of Comparison for Performance of Different threads

3.1.3 Efficiency

Efficency of utilization of the processors are shown in Figure 8. The results shows higher efficency for Duo core compared to the Quad core. The larger the number of thread, the lower the efficiency. The reduction is much more significant for the Duo core. Efficiency is highest with the two thread for both Duo and Quad core This is true for all the image sizes. But the larger the image the higher the efficiency.

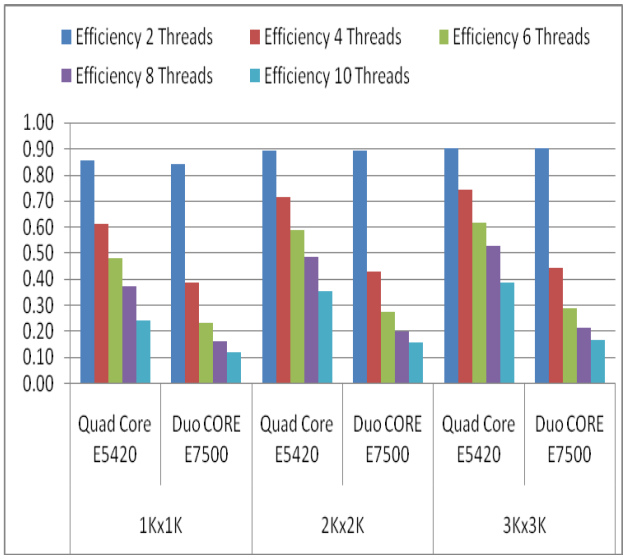

Figure 8: Histogram of Comparison for Efficiency of Different threads

This research is about utilizing used PC in organization by demonstrating its usability in parallel implementation of image processing algorithm. Sobel edge detection algorithm had been successfully implemented in sequential and parallel manner. Basically both duo and quad have similar trends for all the method of performance such as speedup, performance improvement and the efficiency. Generally the performance measurement methods increases from 1Kx1K to 2Kx2K but reduces at 3Kx3K images. However, the parallel execution remains to outperform the sequential execution as indicated by the positive measurement for speedup and performance improvement but not in terms of efficiency.

3..1.4 CPU Utilization Factor Using the Algorithm

The main objective of this paper is to fully utilize the CPU to its utmost potential. Therefore, CPU utilization needs to be monitored via the task manager to determine whether or not it met the criteria intended. Table 3 describes the CPU utilization factor when executing the sequential algorithm on both CPUs respectively.

Table 3: Sequential CPU Utilization Percentage

Processor e	CPU Utilization
Quad Core	15%
Duo Core	50%

ISBN: 978-1-61804-019-0

As suggested by the result above, the CPU utilization factor is nowhere near its full capabilities. The first CPU wasted 85% of the CPU's full capabilities while the second CPU wasted around 50%. Table 4 describes the CPU utilization percentage for each CPU.

Table 4: Parallel CPU Utilization Percentage

Processor	CPU Utilization			
	2 trds	4 trds	6 thrs	8 trds
Quad Core	25%	50%	75%	100%
Duo Core	50%	100%	100%	100%

The CPU utilization factor in a way, compliment the performance result earlier. The efficiency of the first processor to execute 8 concurrent threads at the same time is evidence as the CPU is only fully utilized when executing at this thread setup.

The reason why the second processor failed to emulate the first processor's result is because it already fully utilized the CPU when executing only 2 threads. More threads won't make thing any faster as the resource is already fully utilized.

Conclusion and recommendation

It is certainly evidence that parallel multicore Sobel algorithm improves the performance of the traditional sequential Sobel algorithm by fully utilize the CPU to its utmost potential. Parallel processing performs better than sequential processing in terms of speed but with a trade off with the performance and the efficiency of utilizing the processors individually. However with the increasing amount of data sizes to be process, multicore provides a welcome alternative for fast processing. This research provide a gateway to identify suitable methods to process large data fast. This initial research can invoke the use of multicore in clustering environment. It also provides some knowledge in balancing the utilization of single core and multicore processors in heterogenous cluster environment. This work is not limited to image processing methods only

Acknowledgment

The authors acknowledge with gratitude to Research Management Institute (RMI), UiTM and financial support from E-Science Fund (06-01-01-SF0306) from the Ministry of Science, Technology and Innovation (MOSTI),Malaysia.

References:

[1] B. Barney, *Introduction to Parallel Computing.* Retrieved from Lawrence Livermore National Laboratory: https://computing.llnl.gov/tutorials/parallel_comp/, 2010

[2] C .Lin and L.Snyder,"Principles of Parallel programming", Pearson International, 2009

[3] N. Haron, R. Ami, I. A.Aziz, L. T. Jung and S. R.. Shukri, Parallelization of Edge Detection Algorithm using MPI on Beowulf Cluster. *Innovations in Computing Sciences and Software Engineering* . 2010

[4] C. Szydlowski, Multithreaded Technology & Multicore Processors. *Dr. Dobb's Journal,* May 2005.

[5] S.Akhter and J.Roberts, Multi-Core Programming: Increasing Performance through Software Multi-threading, 2006

[6] R.C. Gonzales and R.E Woods," Digital Image Processing", Pearson Education International, 2002.

[7] B.Allen, M.Wilkinson Parallel Programming, Techniques and Applications Using Networked Workstations and Parallel Computers, Pearson,2005.

[8] Z.Guo,W.Xu and Z. Chai, Image Edge Detection Based on FPGA. *2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science* . 2010

[9] R. L.Rosas, A. D.Luca and F. B. Santillan (). SIMD Architecture for Image Segmentation using Sobel Operators Implemented in FPGA Technology. *2nd International Conference on Electrical and Electronics Engineering (ICEEE) and XI Conference on Electrical Engineering (CIE 2005),* 2005

[10] C.Szydlowski, Multithreaded Technology & Multicore Processors. *Dr. Dobb's Journal, May 2005* .