PERFORMANCE EVALUATION OF STENCIL ON MULTI-CORE COMPUTER

MUSTAFA SALEH MAHDI AL-KHAFFAF

FSKTM 2019 28

PERFORMANCE EVALUATION OF STENCIL ON MULTI-CORE COMPUTER

By

MUSTAFA SALEH MAHDI AL-KHAFFAF

Thesis submitted to the School of Graduate Student, Universiti Putra Malaysia,

in Fulfillment of the Requirement for the Degree of Master of Computer Science

2019

COPYRIGHT

DEDICATION

This Thesis is dedicated to:

The sake of Allah. My Creator and my Master.

My great teacher and messenger, and beloved supervisor Assoc. Prof. Dr. Nor Asilah Wati Abdul Hamid (May Allah bless and grant her).

Who taught us the purpose of life.

My beloved Parents,

My Brother and Sisters,

And all my friends,

For

Their Endless Patience and Support

ABSTRACT

Abstract of this thesis is presented to the Senate of Universiti Putra Malaysia, in fulfillment of the requirement for the degree of Master of Computer Science

PERFORMANCE EVALUATION OF STENCIL ON MULTI-CORE COMPUTER

By

MUSTAFA SALEH MAHDI AL-KHAFFAF

Chair: Assoc. Prof. Dr. Nor Asilah Wati Abdul Hamid

Faculty: Computer Science and Information Technology

High Performance Computing (HPC) can be defined as the practice of combining computing power to attain higher level of performance, aiding one to solve complex tasks in various sectors, namely engineering, science and business efficiently and faster, compared to what a normal computer or workstation might offer. As the number of HPC users grows, various parallel programming models are also developed to fulfil the specific goals and needs of each user. However, with the availability of multiple parallel programming models to be chosen from, users will face with another challenge, on how to choose the best model that meets the specific requirements. Thus, the current work has performed a comparative study on MPI, OpenMP, Threading Building Blocks (TBB), and POSIX threads (Pthreads) as well a hybrid paradigms (MPI+OpenMP and MPI+Pthreads) in compute-intensive problem and in multi-core environment, to provide program developers and potential researchers with the information on the models that fit their goals best. The performance of the four selected parallel programming models has been measured through the speedup, execution time and also

efficiency. Besides that, the current study has applied the stencil computation as a benchmark application.

ABSTRAK

Abstrak tesis yang dikemukakan Senat Universiti Putra Malaysia sebagai memenuhi

Keperluan untuk Ijazah Komputer Sains

PENILAIAN PRESTASI STENCIL KE ATAS KOMPUTER MULTI-CORE

Oleh

MUSTAFA SALEH MAHDI AL-KHAFFAF

Pengerusi: Assoc. Prof. Dr. Nor Asilah Wati Abdul Hamid

Fakulti: Computer Science and Information Technology

High Performance Computing (HPC) ditakrifkan sebagai penggabungan kuasa pengkomputeran untuk mencapai tahap prestasi yang lebih tinggi, dan bertujuan untuk membantu seseorang untuk menyelesaikan tugas kompleks dalam pelbagai sektor, seperti kejuruteraan, sains, dan perniagaan dengan lebih cekap dan pantas, berbanding dengan prestasi yang ditawarkan oleh komputer biasa atau stesen kerja. Oleh kerana bilangan pengguna HPC semakin meningkat, pelbagai model *parallel programming* telah dibina untuk memenuhi matlamat dan keperluan spesifik setiap pengguna. Walau bagaimanapun, dengan adanya pelbagai pilihan model *parallel programming*, pengguna akan berhadapan dengan cabaran untuk memilih model terbaik yang memenuhi keperluan khusus mereka. Oleh itu, kajian semasa telah melakukan kajian perbandingan di antara MPI, OpenMP, Threading Building Blocks (TBB), dan POSIX thread (Pthreads), serta paradigma hibrid (MPI + OpenMP dan MPI + Pthreads), dalam mengira masalah intensif dan di dalam *multi-core environments*, untuk menawarkan pemaju program dan penyelidik dengan maklumat mengenai model yang sesuai

dengan matlamat dan keperluan khusus mereka. Prestasi empat model *parallel programming* yang dipilih telah diukur melalui kelajuan, masa pelaksanaan, dan juga kecekapan setiap model. Selain itu, kajian semasa telah menggunakan pengiraan stensil sebagai aplikasi penanda aras.

ACKNOWLEDGEMENT

To my Lord Allah Almighty, I am thankful for the blessings and virtues, and for reconciling, strength, patience, courage, and determination he gave me to complete this work to the fullest, Alhamdulillah.

I would like to extend my gratitude to **Assoc. Prof. Dr. Nor Asilah Wati Abdul Hamid**, for her supervision, advice, and guidance from the very early stage of this project as well as giving me extraordinary experiences throughout the work. Above all and the most needed, she provided me with unflinching encouragement and support in various ways. Moreover, also like to thank my Assessor, **Assoc. Prof. Dr. Rohaya Latip** for his suggestions and comments through the whole process of developing my thesis.

My warmest gratitude goes to all of my family members, especially **my Father and my Mother** who always believed in me, gave me all the possible support, and being patient with me for years, providing me with everything, just to make me focus on my goals. I am also thankful for **my friends** around me for their support and concern about my study, and their willingness to provide me with any support I need.

Finally, I must extend my sincere thanks to the **Ministry of Higher Education**. Nonetheless, my gratitude to the **Malaysian people** in general for their perfect hospitability in their green land during my study period.

APPROVAL

This thesis was submitted to the Faculty of Computer Science and Information Technology of Universiti Putra Malaysia and has been accepted as partial fulfillment of the requirement for the degree of Master of Computer Science.

The members of the Supervisory Committee were as follows:

Supervisor: Assoc. Prof. Dr. Nor Asilah Wati Abdul Hamid

Department of Communication Technology and Network

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

Date and Signature: _____

Assessor: Assoc. Prof. Dr. Rohaya Latip

Department of Communication Technology and Network

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

Date and Signature: _____

DECLARATION

I declare that the thesis is my original work, except for the quotation and citations, which have been duly, acknowledge. I also declare that it has not been previously, and is not concurrently, submitted for any other degree at Universiti Putra Malaysia or any other institution.

Signature: _____

Name and Matric No: **MUSTAFA SALEH MAHDI AL-KHAFFAF (GS48123)**

Date: ----------

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

HPC	High Performance Computing
CPU	Central Processing Unit
GPU	Graphics Processing Unit
MPI(SHM)	Message Passing Interface Shared Memory
Pthread	POSIX Threads
OpenMP	Open Multi-Processing
TBB	Threading Building Blocks
API	Application Programming Interface

CHAPTER 1

INTRODUCTION

1.1 Background

The term High Performance Computing (HPC) carries the meaning of the practice of accumulating computing power in order to obtain greater level of performance compared to what a normal computer or workstation could give, especially when it comes to solving complex tasks in various sectors, namely engineering, science and business (Ashraf, Eassa, Albeshri, & Algarni, 2018). In addition, HPC is also referred in two other terms, namely supercomputing and parallel computing. The chief concept in HPC is that, rather than employing a single compute which will take 100 hours to complete a task, the same task could be solved in 1 hour by employing 100 computers at the same time. While a single node in supercomputer might not be more powerful in comparison to a single compute, but it will when all the resources are connected with each other (Ashraf et al., 2018). HPC systems enable high communication bandwidth, but with low level of message intermission and failure rates on computer nodes. In the recent years, HPC has been extensively employed due to its powerful computational performance compared to machines with single-processing. This is due to the homogeneity of its multi-core engineering, that consists of a number of interchangeable processor cooperating to complete complex jobs (Albalawi, Thulasiraman, & Thulasiram, 2013). The idea of parallel engineering was first discovered around 1960 together with transistors, which was used instead of tubes and other restricted machineries. These transistors help the processors to become smaller and easier to manage. The first generation of microprocessors were later introduced in the early 70s.

The main aim of parallel computing is to improve the performance and efficiency; thus every step of parallelization process, namely assignment, decomposition, mapping and synchronization have significant roles in achieving this (Culler, Singh, & Gupta, 1999). The development of parallel computing includes both data centers and supercomputers, and also any devices which operate through CPU or GPU processing unit. As the need for parallel computing grows, the number of processors also increase steadily to meet the demands (Navarro, Hitschfeld-Kahler, & Mateu, 2014). There are various parallel programming models that have been introduced to aid the developers, programmers and researchers, but, the most widely used models are MPI, OpenMP, Threading Building Blocks (TBB),hybrid (MPI/OpenMP and MPI/Pthreads), and POSIX threads (Pthreads). However, with the huge number of selections on parallel programming models comes another obstacle, namely on which model is best to fulfil the specific goal or tasks based on their level of performance. This problem will be discussed further in the following section.

1.2 Problem statement

After the discovery of the correlation between the wall of the chip dissipations with the increase of clock speed in the semiconductor industry technology, the Moore's law was introduced into the add processor cores. In parallel to this discovery, the number of processor cores mounted on a single chip has been increased by the manufacturers. While these multi-core processors are fundamental for many program developers, the question on how to maximize the performance of the multi-core platforms in HPC has been constantly discussed (Chou & Chen, 2016). To produce efficient parallel programs, the program developers need to firstly understand the basis of multi-core platforms, particularly on the characteristics of the hardware. In response to that, a number of parallel programming models were introduced (Diaz, Munoz-Caro, & Nino, 2012; Kasim, March, Zhang, & See, 2008).

The main goals of parallel programming are to firstly acquire high performance from the application, and secondly, to solve complex problems which require heavy load of processing and immense resources. It also aims to reduce the execution time which could be achieved through either increasing the system speed-up, or maximizing the parallel application development, or both. As the memory to process data are accessible by a huge number of threads, synchronization is also the key element in parallel programming to prevent starvation. In the programming effort, but also hides the details of the hardware. Through the advancement of software technology, the distributed and parallel applications continues to progress, thus heighten the ability to reach the hardware's theoretical performance peak (Kang, Lee, & Lee, 2015; Noaje, Krajecki, & Jaillet, 2010).

However, in line with the growing number of parallel programming models and their various distinctive features, developers and programmers will face with the question of which parallel programming model is the most suitable for the implementation of computation-intensive on multi-core system with the most efficient performance (Michailidis & Margaritis, 2016; Salehian, Liu, & Yan, 2017)). Thus, the current research will examine four parallel programming models namely (MPI Shared Memory, OpenMP, POSIX threads (Pthreads), and Threading Building Blocks (TBB) plus hybrid models (MPI+OpenMP and MPI+Pthreads). In order to provide the developers, programmers and potential researchers with a clearer picture of the most suitable parallel programming model to be implemented, a comparative study has been done on the selected five models.

1.3 Objectives

The objective of this research is:

To analyze the performance of four parallel programming models which includes OpenMP, MPI SHM, TBB and Pthreads with stencil as the benchmark application on a multi-core shared memory system.

1.4 Project Scope

The current research will be limited based on the following aspects:

1. A critical review of the four selected parallel programming models which includes OpenMP, MPI SHM, TBB, and Pthreads on multi-core shared memory systems.

2. The experiment for each model will be implemented using stencil computation to measure and analyze the performance of the five selected models.

1.5 Thesis Organization

The thesis is divided into six chapters, a brief description for each as follows: Chapter 1, description for in an appropriate manner Research area, problem statement, objective, project scope, and thesis organization. Chapter 2, introduces an overview of the selected four parallel programming models, the benchmark that has been employed in this study, literature review of the research and related works. Chapter 3, describes the methodology research that has been conducted, parameters, algorithm and performance, and metrics that have been exploited in the experiment. Chapter 4, provides the implementation of stencil computation using the four selected parallel programming models. Chapter 5, discusses the results of performance analysis, shows the behavior of each parallel programming model that is picked by this study,

and provides the core findings that are observed during the analysis. Chapter 6, supplies a conclusion of the study and the future work.

REFERENCES

Albalawi, E., Thulasiraman, P., & Thulasiram, R. (2013). Task Level Parallelization of All Pair Shortest Path Algorithm in OpenMP 3.0. In *2nd International Conference on Advances in Computer Science and Engineering (CSE 2013)* (pp. 1–2).

Asaadi, H., Khaldi, D., & Chapman, B. (2016). A Comparative Survey of the HPC and Big Data Paradigms: Analysis and Experiments. In *2016 IEEE International Conference on Cluster Computing* (pp. 423–432). IEEE. https://doi.org/10.1109/CLUSTER.2016.21

Asaduzzaman, A., Sibai, F. N., & El-Sayed, H. (2013). Performance and power comparisons of MPI Vs Pthread implementations on multicore systems. In *2013 9th International Conference on Innovations in Information Technology (IIT)* (pp. 1–6). IEEE. https://doi.org/10.1109/Innovations.2013.6544384

Ashraf, M. U., Eassa, F. A., Albeshri, A. A., & Algarni, A. (2018). Performance and Power Efficient Massive Parallel Computational model for HPC Heterogeneous Exascale Systems. *IEEE Access*, *6*, 1–1. https://doi.org/10.1109/ACCESS.2018.2823299

Brickner, R. G., Kennedy, K., Mellor-Crummey, J., & Roth, G. H. (1997). *Compiling Stencils in High Performance Fortran*.

Chou, C.-Y., & Chen, K.-T. (2016). Performance Evaluations of Different Parallel Programming Paradigms for Pennes Bioheat Equations and Navier-Stokes Equations. In *2016 International Computer Symposium (ICS)* (pp. 503–508). Chiayi, Taiwan: IEEE.

Christgau, S., Spazier, J., & Schnor, B. (2015). A Performance and Scalability Analysis of the Tsunami Simulation EasyWave for Different Multi-Core Architectures and Programming Models. *University Potsdam, Institute of Computer Science, Potsdam, Germany, Tech. Rep. TR-2015-01*.

Christgau, S., Spazier, J., Schnor, B., Hammitzsch, M., Babeyko, A., & Waechter, J. (2014). A comparison of CUDA and OpenACC: accelerating the tsunami simulation easywave. In *Architecture of Computing Systems (ARCS), 2014 Workshop Proceedings* (pp. 1–5). VDE.

Coulouris, J. D. G. (2015). *Distributed Systems: Concepts and Design* (5th ed.). New York, NY, USA: Addison Wesley.

Culler, D. E., Singh, J. P., & Gupta, A. (1999). *Parallel computer architecture: a hardware/software approach*. San Francisco: Morgan Kaufmann Publishers.

Diaz, J., Munoz-Caro, C., & Nino, A. (2012). A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era. *IEEE Transactions on Parallel and Distributed Systems*, *23*(8), 1369–1386. https://doi.org/10.1109/TPDS.2011.308

Ajkunic, E., Fatkic, H., Omerovic, E., Talic, K., & Nosovic, N. (2012). A Comparison of Five Parallel Programming Models for C++. In *MIPRO, 2012 Proceedings of the 35th International Convention* (pp. 1780–1784). Retrieved from https://www.researchgate.net/profile/Hana_Fatkic/publication/261424700_A_comparison_of_five_parallel_programming_models_for_C/links/5607df6808ae5e8e3f3a3a1d.pdf

Guerrera, D., Maffia, A., & Burkhart, H. (2018). Reproducible stencil compiler benchmarks using prova! *Future Generation Computer Systems*. https://doi.org/10.1016/j.future.2018.05.023

Holewinski, J., Pouchet, L.-N., & Sadayappan, P. (2012). High-performance code generation for stencil computations on GPU architectures. In *Proceedings of the 26th ACM international conference on Supercomputing* (pp. 311–320). ACM.

Jin, H., Jespersen, D., Mehrotra, P., Biswas, R., Huang, L., & Chapman, B. (2011). High performance computing using MPI and OpenMP on multi-core parallel systems. *Parallel Computing*, *37*(9), 562–575. https://doi.org/10.1016/j.parco.2011.02.002

Kang, S. J., Lee, S. Y., & Lee, K. M. (2015). Performance Comparison of OpenMP, MPI, and MapReduce in Practical Problems. *Advances in Multimedia*, *2015*, 1–9. https://doi.org/10.1155/2015/575687

Kim, C. G., & Seo, Y.-H. (2016). Parallel JPEG Color Conversion on Multi-Core Processor. *International Journal of Multimedia and Ubiquitous Engineering*, *11*, 9–16. http://dx.doi.org/1014257/ijmue.2016.11.2.02

Krpic, Z., Martinovic, G., & Crnkovic, I. (2012). Green HPC: MPI vs. OpenMP on a shared memory system. In *MIPRO, 2012 Proceedings of the 35th International Convention* (pp. 246–250). IEEE.

Leist, A., & Gilman, A. (2014). Comparative Analysis of Parallel Programming Models for C++. In *The Ninth International Multi-Conference on Computing in the Global Information Technology*.

Luecke, G., Weiss, O., Kraeva, M., & Hoekstra, J. C. J. (2010). Performance Analysis of Pure MPI Versus MPI+OpenMP for Jacobi Iteration and a 3D FFT on the Cray XT5.

Memeti, S., Li, L., Pllana, S., Kołodziej, J., & Kessler, C. (2017). Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: Programming Productivity, Performance, and Energy Consumption (pp. 1–6). ACM Press. https://doi.org/10.1145/3110355.3110356

Michailidis, P. D., & Margaritis, K. G. (2016). Scientific computations on multi-core systems using different programming frameworks. *Applied Numerical Mathematics*, *104*, 62–80. https://doi.org/10.1016/j.apnum.2014.12.008

Navarro, C. A., Hitschfeld-Kahler, N., & Mateu, L. (2014). A Survey on Parallel Computing and its Applications in Data-Parallel Problems Using GPU Architectures. *Communications in Computational Physics*, *15*(2), 285–329. https://doi.org/10.4208/cicp.110113.010813a

Noaje, G., Krajecki, M., & Jaillet, C. (2010). MultiGPU computing using MPI or OpenMP. In *Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing* (pp. 347–354). https://doi.org/10.1109/ICCP.2010.5606414

Salehian, S., Liu, J., & Yan, Y. (2017). Comparison of Threading Programming Models. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (pp. 766–774). IEEE. https://doi.org/10.1109/IPDPSW.2017.141

Schäfer, A., & Fey, D. (2011). High performance stencil code algorithms for GPGPUs. *Procedia Computer Science*, *4*, 2027–2036.

Sharma, M., & Soni, P. (2014). Comparative Study of Parallel Programming Models to Compute Complex Algorithm. *International Journal of Computer Applications*, *96*(19), 9–12. https://doi.org/10.5120/16900-6961

Silven, M. (2014). *Evaluation and Comparison of Programming Frameworks for Shared Memory Multicore Systems*.

Tousimojarad, A., & Vanderbauwhede, W. (2014). Comparison of Three Popular Parallel Programming Models on the Intel Xeon Phi. In *Euro-Par 2014: Parallel Processing Workshops* (pp. 314–325). Springer, Cham. https://doi.org/10.1007/978-3-319-14313-2_27