UNIVERSITI PUTRA MALAYSIA

IMPLEMENTATION OF MPI IN PYTHON AND COMPARISON WITH OTHER PARALLEL PROGRAMMING TECHNIQUES

ASMALIZA ZULKIFLI

FSKTM 2015 30

IMPLEMENTATION OF MPI IN PYTHON AND COMPARISON WITH OTHER PARALLEL PROGRAMMING TECHNIQUES

By

ASMALIZA ZULKIFLI

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirements for the Master of Computer Science

JUNE 2015

DEDICATIONS

This thesis is dedicated to:

My beloved late parents, Hj. Zulkifli Hj. Zakaria and Hjh. Senik Hj. Ahmad,

May Ar-Rahman Allah SWT bless both of them forever till Jannah.

My dear brothers, sisters and family.

Thank you very much for the support, encouragement and constant love. You're my inspiration!

Also to my respected supervisor, Dr. Nor Asilah Wati Abdul Hamid.

Your encouragement, support, advice and guidance are an outline to my achievement.

Finally, thank you to all my friends and colleagues,

Thank you for the support and encouragement.

ABSTRACT

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the Master of Computer Science.

IMPLEMENTATION OF MPI IN PYTHON AND COMPARISON WITH OTHER PARALLEL PROGRAMMING TECHNIQUES

By

ASMALIZA ZULKIFLI

Supervisor: Dr. Nor Asilah Wati Abdul Hamid
Faculty: Faculty of Computer Science and Information Technology

Abstract. High performance computing becomes more important in many areas by provide fast, reliable and cost effective solutions in many applications. Availability of multi-core system in various platforms ranging from desktop computer to supercomputer enable parallelism to be exploiting by users. Parallel programming provides access to users to optimize resources by applying multi-threading or multi-processing techniques in application development. One of the popular approaches in parallel programming is message passing which is widely used in both distributed and shared memory architecture.

Python is a powerful open source programming language that popular among scientific computing committee. It provides flexibility and space for skilled users to create their own environment, and appeal beginners with its object-oriented programming. Python also support parallel programming by adapting message passing paradigm into its language. Few of its MPI implementations are pyMPI, Pypar, MPI for Python (mpi4py) and pypvm. MPI for Python is complies with MPI-2 specification and can be used with other Python modules such as NumPy and Cython to exploit multiple processors.

This project will provide detail analysis of current implementation of message passing paradigm in Python and compare it with other popular parallel programming technique. Therefore, it aims to produce a good reference to users especially beginners in developing parallel applications.

ABSTRAK

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah sarjana Sains Komputer.

IMPLEMENTATION OF MPI IN PYTHON AND COMPARISON WITH OTHER PARALLEL PROGRAMMING TECHNIQUES

Oleh

ASMALIZA ZULKIFLI

Penyelia: Dr. Nor Asilah Wati Abdul Hamid
Fakulti: Fakulti Sains Komputer dan Teknologi Maklumat

Abstrak. Pengkomputeran berprestasi tinggi menjadi lebih penting dalam pelbagai bidang dengan menyediakan penyelesaian yang cepat, boleh dipercayai dan kos efektif dalam pelbagai aplikasi. Kewujudan sistem multi-core dalam pelbagai platform yang terdiri daripada komputer desktop untuk superkomputer membolehkan keselarian dieksploitasi oleh pengguna. Pengaturcaraan selari menyediakan akses kepada pengguna untuk mengoptimumkan sumber dengan menggunakan multi-threading atau multi-pemprosesan teknik dalam pembangunan aplikasi. Salah satu pendekatan yang popular dalam pengaturcaraan selari adalah penghantaran mesej yang digunakan secara meluas di kedua-dua seni bina memori; teragih dan perkongsian.

Python adalah bahasa pengaturcaraan daripada sumber terbuka yang berkuasa dan popular di dalam bidang pengkomputeran saintifik. Ia menyediakan fleksibiliti dan ruang untuk pengguna mahir dalam mewujudkan persekitaran pengaturcaraan mereka sendiri, selain menyakinkan pengguna kurang mahir dengan pengaturcaraan berorientasi objek itu. Python juga menyokong pengaturcaraan selari dengan menyesuaikan paradigm penghantaran mesej ke dalam bahasanya. Beberapa pelaksanaan MPI di dalam persekitaran Python adalah pyMPI, Pypar, MPI for Python (mpi4py) dan pypvm. MPI for Python mematuhi spesifikasi MPI-2 dan boleh digunakan dengan modul Python yang lain seperti NumPy dan Cython untuk mengeksploitasi multi-pemprosesan.

Projek ini akan menyediakan analisis terperinci daripada pelaksanaan semasa paradigm penghantaran mesej dalam persekitaran Python dan melaksanakan perbandingan dengan teknik pengaturcaraan selari popular yang lain. Oleh itu, ia bertujuan untuk menghasilkan satu rujukan yang baik kepada pengguna terutama pengaturcara muda dan kurang mahir dalam membangunkan aplikasi selari.

ACKNOWLEDGEMENTS

Alhamdulillah, all praises to Allah S.W.T for the strength and blessing in completing this thesis.

I would like to express my deepest appreciation to my supervisor, Dr. Nor Asilah Wati Abdul Hamid, for the patience and truthful guidance and unlimited confidence in me. I would also like to thank her for being an open person and for encouraging and helping me to shape my interest and ideas. Her guidance helped me much throughout the entire journey of this research.

My sincere thanks and gratitude also dedicated to the Dean and members of FSKTM for their endless support. Their care and support help me overcome setbacks and stay focused on my study. I'm highly appreciate their believed in me.

My greatest appreciation and friendship goes to my friends, especially those who spend sweet and sweat together at Universiti Putra Malaysia. Thank you very much friends.

APPROVAL

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Master Computer Science. The members of the Supervisory Committee were as follows:

Nor Asilah Wati Abdul Hamid, PhD
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Supervisor)

Mohamed Othman, PhD
Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Accessor)

Date: 24 June 2015

DECLARATION

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

Signature : _____

Name and Matric No : **ASMALIZA ZULKIFLI (GS36927)**

Date : _____

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

HPC	High Performance Computing
LOC	Line of codes
MPI	Message Passing Interface
mpi4py	MPI for Python
MPMD	Multiple Program Multiple Data
MS	Matrix size
np	Number of processors
OpenMP	Open Specification for Multi-Processing
PVM	Parallel Virtual Machine
SPMD	Single Program Multiple Data

1 INTRODUCTION

1.1 Background

High performance computing (HPC) has growing to become more important in many critical sectors such as finance, medical, engineering, and research in providing fast execution of large computational complexity of specific jobs that require powerful computing resources to obtain the results. It's proven to become most effective solutions in many architecture platforms such as cluster, grid and cloud computing.

HPC not only benefit large organizations with powerful supercomputers installed at their infrastructure, but also can accelerate Small and Medium Enterprises (SME) to be more productive and improve the company credibility in target market. Thinkplay.tv, a UK company that using HPC resources to render big files for image processing and Moleculomics, a Swansea-based pharmaceutical company that perform analysis of genetic information for produce new drug; are two examples of SME company that successfully gain benefit from HPC technology (Shainer, 2014).

Most of commercial workstations and home desktops have multi-core processors installed to improve the performance and expedite the computational tasks. Tianhe-2, also known as Milky Way-2 is a top supercomputer in the world since June 2013, has 3,120,000 cores distributed in 16,000 computer nodes with Intel Ivy Bridge Xeon processors and most advance Xeon Phi chips installed (Top500). It achieves 33.86 petaflop per second in its benchmarking test. In recent article by Laskow (2015), stated that US Department of Energy has announced to invest USD200 million to build new supercomputer Aurora by 2018 with less power consumption than its competitors but can achieve exa-scale. The advent of multi-core computers has bringing parallel programming to become an essential tool to fully utilize the advantage of multi-core hardware by harvesting computer processing power.

Multi-threading and multi-processing are two common concepts used by a programmer to build a parallel program. Multi-threading refers to shared memory architecture paradigm where master thread forks a collection of tasks to slave threads that shared same memory location. OpenMP (Open specification for Multi-Processing) is de facto standard that using shared memory architecture principle. Meanwhile, multi-processing refers to distributed memory architecture paradigm where a collection of tasks been

assigned to designated processors and communication overhead between processors possibly occur. Message-passing has proven to be an effective computational model specifically for distributed memory architectures. Message Passing Interface (MPI) and Parallel Virtual Machine (PVM) are two popular approaches of message passing techniques, even though MPI implementation are widely been used since it support point-to-point and collective communication and offer high performance, scalability, and portability (Hafeez et al., 2011).

MPI has been implemented widely across various programming languages; either or low-level programming languages such as C/C++, C#, Java, Fortran, (Hafeez et al., 2011), or high-level programming languages in scientific computing likes MATLAB, IDL, Octave, Scilab and Python due to its neutral architecture and independency to any specific programming language. It provides a set of library routines that hides all the low-level details, simplifying development and maintaining portability, without jeopardizing the performance (Dalcin et al., 2011).

1.2 Problem Statement

Majority big organizations have adapted HPC technologies in their workloads to ensure faster computational resources, increase productivity and minimize operational cost. Availability of HPC resources across computing domains (cluster, grid and cloud) becomes more reliable and cost effective to appeal stakeholders to turn into parallel programming. Besides that, commercial desktop computers also equipped with multi-core CPUs to enhance the performance by enabling parallelism in executing tasks.

A survey conducted by Desktop Engineering shows only 17% of CAE/CAD users using HPC technologies in facilitating their routine works. One of the reasons is lacking of skilled staff to manage an expanded HPC computing environment (Desktop Engineering, 2011). Complexity of parallel programming can demotivate programmers to write and debug parallel codes especially in low level programming.

Python is a high level programming language that widely used in scientific community that use clean and simple syntax which enable skilled programmers to create their computing environment. It is a powerful language that offers advantage in writing high-level program of large-scale, massively parallel scientific applications and driving simulations in parallel architectures (Beazley et al., 1997; Hinsen, 2007). Advantages

offer by MPI and Python can be combine to offer better parallel programming technique to ease and guide user especially beginner to explore about multi-core processing principle.

This research will provide critical review of current MPI implementation in Python (MPI for Python (mpi4py)) with the standard of message passing technique (OpenMPI) and multi-threading approach (OpenMP).

1.3 Objectives

The objectives of this research as follow:

i. To provide in-depth analysis about current implementation of Python as a parallel programming technique,
ii. To provide comparative analysis of current implementation of mpi4py with other parallel programming language standards; particularly OpenMP and OpenMPI
iii. To evaluate and analyze the performance of parallel programming languages; OpenMP, OpenMPI, and mpi4py

1.4 Project Scope

This research will focus on two (2) aspects, which are:

i. Critical review of MPI implementation in Python
   Provide explanation and analysis about current MPI implementation in Python. It will highlight the difference, similarity, advantage and disadvantage of its implementation.

ii. Performance analysis of selected MPI Python with other parallel programming techniques
   Provide real measurement (experiment) of each implementation to evaluate and analyze the performance of each parallel programming technique by using matrix multiplication as a testing subject.

1.5 Thesis Organization

This thesis is organized as follows:

Chapter 1, Background of the research, problem statement, objective, scope, research methodology and thesis organization are described in an appropriate manner.

Chapter 2, Literature review of the research and related works are discussed in this chapter.

Chapter 3, Described the methodology research, parameter, algorithm and performance metric.

Chapter 4, Discussed about findings result of performance analysis by including graphs and tables. Each of the findings has been analyses and discussed accordingly.

Chapter 5, Described the conclusion of the thesis and proposed the future work of this thesis.

REFERENCES

Ahmad F., Pitch P. and Xin Y. (2008). A Study of Process Arrival Patterns for MPI Collective Operations. Int J Parallel Prog (2008) 36:543–570

Ajkunic, E., Fat, H., Omerovic, E., Talic, K., & Nosovic, N. (2012). A Comparison of Five Parallel Programming Models for C ++, 1780–1784.

Antonio G.I. (2014). mpi4py HPC Python, TACC

Barrett P., Hunter J., Greenfield P. (2004). Matplotlib – a Portable Python Plotting Package, Astronomical Data Analysis Software & Systems XIV, 2004.

Dalcin L. (2011). MPI for Python, Python for Parallel Scientific Computing

Dalcin, L., Paz, R., Kler, P. a., & Cosimo, A. (2011). Parallel Distributed Computing Using Python. Advances in Water Resources, 34(9), 1124–1139.

Dalcin, L., Paz, R., Storti, M., D'Elía, J. (2008). MPI for Python: Performance Improvements and MPI-2 Extensions. J. Parallel Distrib. Comput. 68 (2008) 655 – 662.

Desktop Engineering. (2011). 5 Easy Steps to a High Performance Cluster.

Graham E. F., George B., Jelena P.G, Thara A. and Jack J. D. (2007). Tuned: An Open Mpi Collective Communications Component. Distributed and Parallel Systems (2007)

Graham, R.L., et al. (2006). OpenMPI: A High-Performance, Heterogeneous MPI. Proceedings of IEEE International Conference on Cluster Computing, pp. 1–9

Hafeez, M., Asghar, S., & Malik, U. A. (2011). Survey of MPI Implementations, 206–220.

Heien, E. M., Takata, Y., Hagihara, K., & Kornafeld, A. (2009). PyMW - A Python Module for Desktop Grid and Volunteer Computing. 2009 IEEE International Symposium on Parallel & Distributed Processing, 1–7.

Hinsen, B. K. (2007). Parallel Scripting with Python.

Jakimovska, D., Jakimovski, G., Tentov, A., & Bojchev, D. (2012). Performance Estimation of Parallel Processing Techniques on Various Platforms. 2012 20th Telecommunications Forum, TELFOR 2012 - Proceedings, 1409–1412.

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.

Jones E., Oliphant T., Peterson P., et al. (2001-2007). SciPy: Open Source Scientific Tools for Python. <http://www.scipy.org/_>

Laskow S. (2015). The Race to Build The World's Greatest Supercomputer.

Lee, K. M., Song, T. H., Yoon, S. H., Kwon, K. H., & Jeon, J. W. (2011). OpenMP Parallel Programming Using Dual-Core Embedded System. Control, Automation and Systems (ICCAS), 2011 11th International Conference on, 762–766.

Matthias K., Thomas R. and Gudula R. (2006). Optimizing MPI Collective Communication by Orthogonal Structures. Cluster Comput (2006) 9:257–279

Miller P. (2000-2013). pyMPI project page. <http://pympi.sourceforge.net/>.

Moritz B. (2013). Different Approaches to the Numerical Solution of the 3D Poisson Equation Implemented in Python, Computing (2013)

Nielsen O. (2002-2014). Pypar project page. <http://code.google.com/p/pypar/>.

Oliphant T. (2005-2010). NumPy: numerical Python. <http://numpy.scipy.org/>.

Pérez F. (2001-2007). IPython: an Enhanced Python Shell. <http://ipython.scipy.org/_>

Pérez F. and Granger B. (2007). IPython: a System for Interactive Scientific Computing, Comput. Sci. Eng. 9 (3) (2007) 21–29.

Schatz, F., Koschnicke, S., Paulsen, N., Starke, C., & Schimmler, M. (2011). MPI Performance Analysis of Amazon EC2 Cloud Services for High Performance Computing, 371–381.

Shainer, G. (2014). High Performance Computing for All (Yes, You Too…). HPC Source. A New Dawn: Bringing HPC to the Enterprise. 13-15.

Stefano M. and Paolo B. (2010). High-Performance Parallel Computations Using Python as High-Level Language, Euro-Par 2010 Workshops

Tokhi, M. O., Hossain M. A. Hossain, Shaheed M. H. (2003). Parallel Computing for Real-time Signal Processing and Control, Advanced Textbooks in Control and Signal Processing , 2003

Top #1 Systems. (2015) <http://www.top500.org/featured/top-systems/>

Tu B., Fan J., Zan J. and Zhao X. (2012). Performance analysis and optimization of MPI collective operations on multi-core clusters. J Supercomput (2012) 60:141–162

Volodymyr T. and Anatoly S. (2014). Efficiency of Parallel Large-Scale Two-Layered MLP Training on Many-Core System. ICNNAI 2014, CCIS 440, pp. 201–210, 2014.

Wikipedia. Matrix multiplication. Access on June 2015

Wikipedia. Speedup. Access on June 2015

Wilbers, I., Langtangen, H.P., Ødegard, A. (2009). Using Cython to Speed up Numerical Python Programs. Proceedings of MekIT 2009, pp. 495–512. NTNU, Tapir

Yaakoub E.K. (2012). HPC Python Tutorial: Introduction to mpi4py, TACC

Yang X.J., Du J. and Wang Z. (2011). An Effective Speedup Metric For Measuring Productivity In Large-Scale Parallel Computer Systems, Journal Supercomputing (2011)

Yosi B.A. (2012). Multicore Programming Using the ParC Language