Journal of Advanced Computing Research Vol. 4, Issue 2 (2019) 4-6

Parallel Processing Applications - A Case Study on Distributed Memory Programming with MPI

Mayasarah binti Maslizan [#1], Mohamed Faidz Mohamed Said [#2]

[#] *Faculty of Computer & Mathematical Sciences, Universiti Teknologi MARA*
70300 Seremban, Negeri Sembilan, MALAYSIA

[1] sarahmaya95@gmail.com

[2] faidzms@ieee.org

***Abstract*—One of the types of parallel programming is distributed memory programming. It refers to a multiprocessor computer system. Each processor has its own private memory and not permitted to be accessed by any of the other tasks. Frequent and rapid data exchanges between the tasks are needed for most of the distributed memory programs. MPI is a communication process for programming parallel computers. The objective of the MPI is to be broadly used for writing message passing programs by generating a flexible, accessible, and efficient benchmark for message passing. MPI remains the vital model used in high-performance computing today. In this research paper, there are a few applications that use distributed memory programming and MPI with different types of models. Besides, the research paper also attempts to show the usage of the distributed memory programming with MPI. Results are then stated and the conclusion is obtained.**

Keywords: **distributed memory, message passing interface**

I. Introduction

Instead of one large memory pool all processors access, a distributed memory system is a multiprocessor system which has a segment of memory for each processor. Such distributed memory machines are called message-passing systems because two processors must pass data directly to one another [1]. Another dissimilarity is that because each processor has its own memory, the same program could run on a workstation network, where each workstation has one processor and its own memory [2]. Message passing interface known as MPI which runs in parallel is the best common process which obviously delivers data from one computer to another. If each computer has its own memory and is not shared with the others, all data exchange has to occur through specific procedures. MPI is not a new programming language. It is a collection of functions and macros, or a c that can be used in C programs [3]. Majority of the MPI programs are based on Single Program Multiple Data known as SPMD. This means that the input data makes each copy computes different things and executes the same number of processes. According to [4], Message Passing Interface (MPI) is used to draw the tear film maps by equating the formula. Next, MPICH-G is a grid-enabled implementation of the MPI, it has been developed to let a user runs MPI programs across several computers at various sites using the similar commands that would be used on a parallel computer [5]. EpiFast has been proposed and master slave computation model has been applied by the researcher to allow scalability on distributed memory systems [6]. Lastly, paper [7] shows the use of the MPI to describe the tactics and application details by parallelizing the SPIDER software package on distributed-memory parallel computers. This paper is organized as follows. Background is described in Section II. Section III state the problem statement based on this research. Section IV show the methods used in this research and Section V defines the results of the testing carried out. Finally, the conclusion is given in Section VI.

II. Background

A distributed memory system refers to an architecture where each CPU has its own memory space, which cannot be directly accessed by other CPUs (Figure 1). Distributed memory systems are most mutual in wide-ranging supercomputers, which consist of multiple computation nodes connected by network interfaces [8]. A distributed memory environment can also be followed on a single workstation, where the same physical memory is logically separated. The benefit of a distributed memory system is that huge computational resources can be used to unravel problems [9]. The weakness is that the network interfaces are very slow and communication to access data on other nodes is accompanied by a latency which can be a major factor in the execution of parallel code [10].

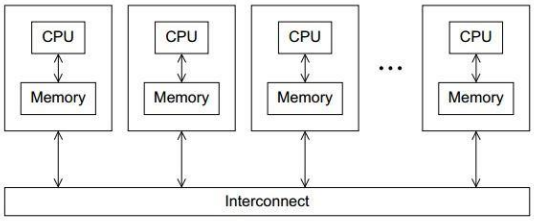

Figure 1. A distributed memory system [2]

MPI known as Message Passing Interface, is a collection of procedures that can be used to create parallel programs in C or Fortran77. MPI has been developed in 1993-1994 [11]. It enables Single Program Multiple Data (SPMD) parallel computing by passing the messages. In the library, there are over 100 procedures provided. MPI is invented to allow the users create programs that can run effectively on most parallel architectures [12]. It is supported on virtually all High-

Performance Computing (HPC) platforms [13]. The final version for the draft standard became accessible in May of 1994. MPI can also support distributed program execution on heterogeneous hardware.

III. Problem Statement

According to [14], message passing performance depends on the time to transmit data through the network, also known as latency and the amount of data that can be passed in a given amount of time, also recognized as bandwidth. The challenges to use the distributed memory programming with MPI in the system are programs may not measure on lower-performance networks and more programming changes are involved to go from serial to parallel version. It can also be difficult to fix. Lastly, the communication network between the nodes make the performance limited.

IV. Methodology

In this research paper, the main focus is on how to implement the distributed memory programming with MPI in the suitable system or model. Based on [7], single-particle reconstruction was implemented in system for processing image data from electron microscopy and related field known as SPIDER by reviewing the basic algorithmic ingredients and the chances for their parallelization. The arrangement of MPI-enabled SPIDER on two main processes in a structure refinement procedure which are alignment and 3-D reconstruction has been discussed and the details was implemented in it. As a final point, the amount of trials faced in the progress of the MPI version of SPIDER has been considered and some possible techniques to amend the present implementation are suggested.

V. Result

This section gives a sequence of outcomes achieved for using distributed memory programming with MPI. The two most time-consuming SPIDER operations which are the multi-reference alignment and 3-D reconstruction operations by using MPI was successfully parallelized [7]. Figure 2 below shows three different computing platforms on the wall-clock time consumed by the parallelized SPIDER multi-reference alignment. The alignment was implemented on the TFIID data set while the alignment command was executed using various numbers of processors to calculate the parallel scalability of the alignment computation.

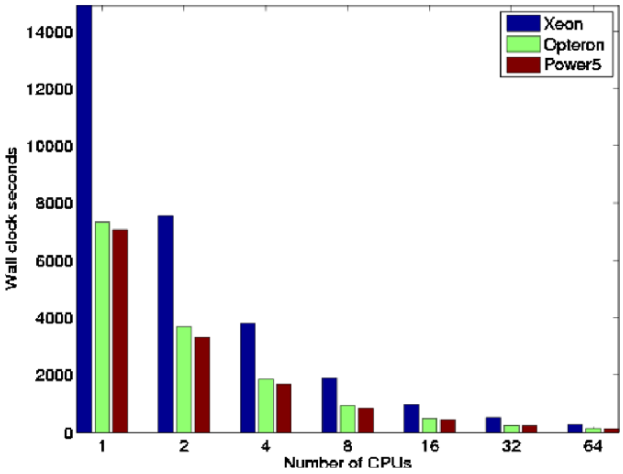

Figure 2. Multi-Reference Alignment [7]

The second outcome is displayed in Figure 3. The graph below represents the performance of the parallel SIRT reconstruction algorithm (BP RP) with the three different platforms used to compute a 3-D reconstruction on the TFIID dataset.

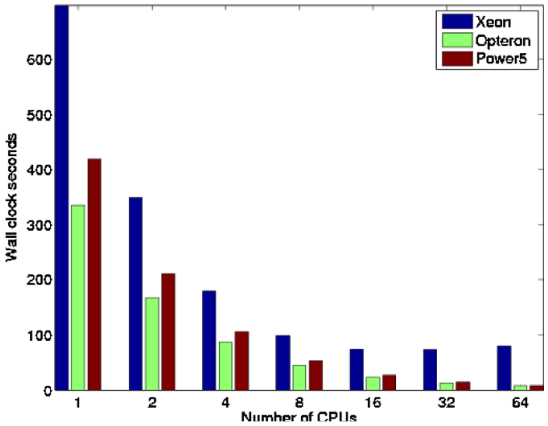

Figure 3. 3-D reconstruction operations [7]

VI. Conclusion

MPI or the Message-Passing Interface is a collection of functions that can be called from C, C++, or FORTRAN programs. A correspondent is a group of methods that can deliver messages to each other. The single-program multiple data or SPMD approach has been used by the numerous parallel programs. There are a few purposes for using MPI. Firstly, MPI is the only message passing library that can be considered a standard. It is supported on virtually all High-Performance Computing (HPC) platforms. Basically, it has substituted all previous message passing libraries. Secondly, it is not necessary to modify the source code when porting application to a various platform that supports the MPI standard. Lastly, vendor implementations should be able to exploit native hardware features to optimize performance. Any implementation is free to develop optimized algorithms. Based on the papers reviewed, there will be numerous applications that can use the distributed memory programming

with MPI in the future because it has been amazingly successful in various kinds of fields so far.

REFERENCES

[1] J. A. Izaac and J. B. Wang, "pyCTQW: A continuous-time quantum walk simulator on distributed memory computers," *Computer Physics Communications*, vol. 186, pp. 81-92, 2015.

[2] S. Jin, Y. Chen, D. Wu, R. Diao, and Z. H. Huang, "Implementation of Parallel Dynamic Simulation on Shared-Memory vs. Distributed-Memory Environments," *IFAC-PapersOnLine*, vol. 48, no. 30, pp. 221-226, 2015.

[3] P. Leggett, S. Johnson, and M. Cross, "CAPLib—a 'thin layer'message passing library to support computational mechanics codes on distributed memory parallel systems," *Advances in Engineering Software*, vol. 32, no. 1, pp. 61-83, 2001.

[4] J. González-Domínguez, B. Remeseiro, and M. J. Martín, "Parallel definition of tear film maps on distributed-memory clusters for the support of dry eye diagnosis," *Computer Methods and Programs in Biomedicine*, vol. 139, pp. 51-60, 2017.

[5] I. Foster and N. T. Karonis, "A grid-enabled MPI: Message passing in heterogeneous distributed computing systems," in *Proceedings of the 1998 ACM/IEEE conference on Supercomputing*, 1998: IEEE Computer Society, pp. 1-11.

[6] K. R. Bisset, J. Chen, X. Feng, V. Kumar, and M. V. Marathe, "EpiFast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems," in *Proceedings of the 23rd international conference on Supercomputing*, 2009: ACM, pp. 430-439.

[7] C. Yang *et al.*, "The parallelization of SPIDER on distributed-memory computers using MPI," *Journal of structural biology*, vol. 157, no. 1, pp. 240-249, 2007.

[8] B. Thomaszewski and W. Blochinger, "Physically based simulation of cloth on distributed memory architectures," *Parallel Computing*, vol. 33, no. 6, pp. 377-390, 2007.

[9] M. Alvioli and R. L. Baum, "Parallelization of the TRIGRS model for rainfall-induced landslides using the message passing interface," *Environmental Modelling & Software*, vol. 81, pp. 122-135, 2016.

[10] M. Cole, "Bringing skeletons out of the closet: a pragmatic manifesto for skeletal parallel programming," *Parallel computing*, vol. 30, no. 3, pp. 389-406, 2004.

[11] P. B. Hansen, "An evaluation of the message-passing interface," *ACM Sigplan Notices*, vol. 33, no. 3, pp. 65-72, 1998.

[12] H. Sivaraj and G. Gopalakrishnan, "Random walk based heuristic algorithms for distributed memory model checking," *Electronic Notes in Theoretical Computer Science*, vol. 89, no. 1, pp. 51-67, 2003.

[13] E. G. Pinho and F. H. de Carvalho, "An object-oriented parallel programming language for distributed-memory parallel computing platforms," *Science of Computer Programming*, vol. 80, pp. 65-90, 2014.

[14] D. Feng, A. N. Chernikov, and N. P. Chrisochoides, "A Hybrid Parallel Delaunay Image-to-Mesh Conversion Algorithm Scalable on Distributed-Memory Clusters," *Procedia Engineering*, vol. 163, pp. 59-71, 2016.

[15] Maya, "https://www.youtube.com/watch?v=zGVGfmcudgE&rel=0." [Online]. Available: https://www.youtube.com/watch?v=zGVGfmcudgE&rel=0.