Performance comparison analysis library communication cluster system using merge sort

D A R Wulandari¹, M E Ramadhan²
¹Information System of Department, University of Jember, Jember, Indonesia
²Mechanical engineering of Department, University of Jember, Jember, Indonesia

E-mail: edowarditsdesain@gmail.com; diah.retnaniw@gmail.com

Abstract. Begins by using a single processor, to increase the speed of computing time, the use of multi-processor was introduced. The second paradigm is known as parallel computing, example cluster. The cluster must have the communication protocol for processing, one of it is message passing Interface (MPI). MPI have many library, both of them OPENMPI and MPICH2. Performance of the cluster machine depend on suitable between performance characters of library communication and characters of the problem so this study aims to analyze the comparative performances libraries in handling parallel computing process. The case study in this research are MPICH2 and OpenMPI. This case research execute sorting’s problem to know the performance of cluster system. The sorting problem use mergesort method. The research method is by implementing OpenMPI and MPICH2 on a Linux-based cluster by using five computer virtual then analyze the performance of the system by different scenario tests and three parameters for to know the performance of MPICH2 and OpenMPI. These performances are execution time, speedup and efficiency. The results of this study showed that the addition of each data size makes OpenMPI and MPICH2 have an average speed-up and efficiency tend to increase but at a large data size decreases. increased data size doesn't necessarily increased speed up and efficiency but only execution time example in 100000 data size. OpenMPI has a execution time greater than MPICH2 example in 1000 data size average execution time with MPICH2 is 0.009721 and OpenMPI is 0.003895 OpenMPI can customize communication needs.

1. Introduction

The technology grows rapidly and new computation paradigms are also being developed. Started by computation process with single processor in the past, recently the usage of multi-processor computers for parallel computing is very common, in order to speed up the computation period. To carry out various kinds of parallel computing, then there is a need for parallel machines infrastructures that consist of a number of computers connected by network able to solve certain problem in parallel.

There are two types of parallel computers, which are Shared Memory Multiprocessor and Distributed Memory Multicomputer. Cluster system belongs to Distributed Memory Multicomputer, which is two or more computers or nodes that are connected into a single integrated system. In a previous study found that the distributed computing make many motivation for implementation because can solve many large problem from large data sets using more than one computers that to be a single integrated resource system. The main point in cluster system is how to the resource sharing all members by efficiently for handling the large problem or data. Each cluster members have each function. there are master and slave. a master node manages the resources needed for the job, divides
the job into parts of the job then assigns the job to each slave. and slave node do task from master node. when the resource is available the task from master gets executed. master give a job to a available of members. when there is no available members, a task goes into a work queue. [1]. The most important part of cluster computer is middleware application that is capable to integrate all computer members within the cluster so that they can work together. The main task of middleware is to communicate and synchronize computers within that particular cluster. One famous cluster's middleware is MPI (Message Passing Interface), which is an independent language of communication protocol in parallel programming. Many MPI libraries have strengths and weaknesses in each implementations [2]. There are numerous libraries in MPI, some of them are GridMPI, OpenMPI, MPICH-Madeleine, MPICH-G2, MPICH-VMI, MPICH2, LAM/MPI [2] [3]. all library of MPI have different process management but the purpose is integrated more than one computers or processors to be a single system with parallel process and establishing a portable, efficient, and flexible standard for message passing that used for writing message passing programs from node to node[4] [5].

There are several available libraries that facilitate the implementation of MPI, such as MPICH2 and openMPI. OpenMPI is proven to have the ability to adjust communication management efficiently, based on the characteristics of existing infrastructure [6]. The testing of cluster system performance can be conducted by implementing certain computer program, and in this research was parallel mergesort. Parallel mergesort is a modification of the old sequential mergesort, in which it supports parallel sorting. Parallel mergesort algorithm is relatively easy to set up since it has divide and conquer method, which can work in parallel [7].

In a previous study found that the distributed computing especially cluster make many motivation for implementation. Implementation using PC-cluster. In this research study about performance analysis communication middleware. Communication middleware in cluster system is very important because the cluster system needs communication tools to work together between many computers to be single integrated system resource. This research compare between MPICH2 and OpenMPI. In a previous study speed-up and efficiency can be parameters of performance when both were solving similar computation process was conducted so this research use speed-up and efficiency to know the parameters both MPICH2 and OpenMPI[8]. In this case, a sorting problem was used. Mergesort was implemented in this research for solving sorting problem. It was used since they are the fastest existing algorithms and is capable of working in parallel. Cluster system was established using virtualization so that it could undertake parallel computing process more efficiently. MPICH2 and OpenMPI were chosen among many other available algorithms for their advantages among their competitors, in which both of them are more well-known and are more commonly utilized.

2. Methods

The analysis of computation process within this research was conducted by comparing the performance of MPICH2 and OpenMPI in solving a sorting problem. In order to carry out the research, the following steps should be conducted:

1. Study related papers about OpenMPI and MPICH2 clustering system, sorting program using mergesort and quicksort methods. and measurement analysis. This literature study was undertaken by utilizing various resources, such as books, journal papers, and electronic documents on the Internet.
2. Design and implement cluster processes, which were conducted in the following orders:
   a. All virtual computers were installed with distro Linux Ubuntu desktop v.10.04 – Lucid Lynx operating system. This research use 5 virtual computer for master and slave (slave is member names of this cluster members). Virtual computer from IT as a service cloud computing was used because the cost-efficient of infrastructure [9].
   b. All of computers has configured in network connection without configured because using virtual computer from IAAS cloud system, Figure 1.
c. Identification all node by defining the IP address and hostname of each node. Master must know about IP address all of member in system for giving their task and each member must know IP address their master for getting their task from master.

d. Configuring SSH can communicate and conduct data exchange to all nodes without having to provide authentication process. In the implementation master node generates a public key which is a random number. This public key will then be sent to the SSH folder on the slave node. Once the public key of the master node is sent into the SSH folder, the public key is converted into an authorized key so that when the master node performs an SSH connection on the slave node, SSH on the slave node will check whether the computer that will access it has been registered in the authorized keys slave node. If indeed the computer that will access it is listed, then the master node can access without having to enter the slave node password first.

e. Configuring NFS. In this cluster system, the directory to be shared is the data directory, in this directory is used to store the files used for the test sample. NFS serves to perform a shared directory that will be used for file sharing.

f. MPICH2 and openMPI were installed and configured in five virtual computers. Hence, there are one master computer and four slave computers.

g. Installation and configuration of Build Essential for running and executing parallel sorting program using C language program.

3. Implementing cluster system and testing analysis.

![Cluster system design](image)

**Figure 1.** Cluster system design; (a) private cloud computing architecture, (b) focus of this research on cluster system design, (c) notebook as remote controller

The cluster system consisted of five homogeneous virtual computers provided by cloud service. One of those five virtual computers was dedicated as master node and the rest four computers were used as slave nodes. Each node was installed with open source operating system Linux Ubuntu 10.04 LTS (Lucid Lynx). From existing five virtual nodes, it was then possible to establish two cluster system, which were Cluster A implementing MPICH2 library and Cluster B implementing openMPI library. Figure 3.1 describes the architectural design of the cluster system implemented in this research. openMPI library was chosen among other MPI libraries because it is one of most famous libraries in parallel computing and because of its good performance. Both libraries has its own unique characteristics, syntaxes, and advantages that are worth comparing.

4. Cluster system performance test was conducted by executing computation program on cluster system with the following scenario:

The purpose of cluster system performance testing was to understand the performance comparison between MPICH2 and openMPI in terms of speed-up and efficiency. Random number sorting problem was implemented in C programming language to test system performance. Two sorting methods used in this research was mergesort algorithms. Number of tested data was configured to be varied, ranging from 100, 1000, 10,000 to 100,000 data size. The data size is number random.
5. Resource monitoring system using HTOP tool.
6. Comparing the result of parallel computing using MPICH2 and that of openMPI, and then deriving conclusion.

Merge sort is used for testing the performance of this cluster system. Mergesort is a sorting algorithm that uses the divide and conquer approach. In a parallel process, the first uncompleted list is divided into two sublists by a single processor, then the sublists are sent to another processor. Each processor handles one sublist. Each sublist is subdivided into two smaller sections, the sublist division is done until the sublist is unbreakable or subdivided, until there is only a number in a sublist. Then each sublist is merged into a new sublist with two numbers then reassembled into each sublist of four numbers until all the numbers on the sublist are combined into one ordered list, Figure 2.

![Diagram of Merge Sort]

**Figure 2.** How To Work Parallel Merge Sort

In this research use three parameters. They are time execution, speed-up and efficiency for testing performance analysis [10]. Speed-up of a program \( S_p \) is defined as the time it takes a program to execute in serial using one processor or computer \( t_s \) divided by the time it takes to execute in parallel using many processors or many computer \( t_p \). The formula for speedup is in equation (1).

\[
S_p = \frac{t_s}{t_p} \tag{1}
\]

Another metric to measure the performance of a parallel algorithm is efficiency. Efficiency is speed-up divide processors that this system use. The formula for efficiency is in equation (2).

\[
E = \frac{S_p}{p} \tag{2}
\]

3. Result and Discussion

Cluster system capability testing in computing process on both cluster systems using execution time, speed-up parameters, and efficiency. Cluster testing is done by running some sample array sorting program which contains integer numbers in four scenarios. The mergesort method is chosen because it uses the divide and conquer method approach, including the fastest and stable sorting method. In another research Divide and Conquer has been designed to work in machines with multiple processors. This test is by sorting random numbers of integer types with intervals of 100 to 100000 numbers or data size. This test will be done using several scenarios. That is using one (sequential) until five processors or computer (multiprocessors). The selection of data between 100 to 100000 random numbers is intended to show the significant difference in execution time. This scenario aims to see the effect of changing the number of nodes with the large amount of data on the execution time, and know what computing program is appropriate so that the performance of both libraries can work optimally. This research have four scenario. The scenario are executing the random number with interval 100,
1000, 10000, and 100000 data size by using sequential and parallel between two until five nodes. The purpose of this testing is performance of Open MPI and MPICH2 with three parameters. That is speed-up efficiency and execution time. The result of this testing are:

1. Execution time base on number of processor

![Graphical comparison OpenMPI and MPICH2 on average execution time for mergesort algorithm](image)

Comparison OpenMPI and MPICH2 on average execution time for merge sort at fig 3, show that The sistem cluster have good performance in data size more than 10000 random number because when the increasing processor execution time more faster then sequential, but have bad performance when the data size less than 1000 random number. When the data size less than 1000 random number the sequential performance more better then many processor. In sequential computing applications the resulting process time is computational time, which is the time needed to calculate the steps of computing. In parallel computing applications, in addition to determining computational time also needs to take into account the time required for communication in sending messages to the parallel application. Increasing processor can retarded execution time when the data size less than 1000 random number. This incident caused all of processor have communication time for solve sorting number. Communication time be used for sharing data, division of task each processor. OpenMPI has a execution time greater than MPICH2 example in 1000 data size average execution time WITH MPICH2 is 0.009721 and OpenMPI is 0.003895 OpenMPI can customize communication needs. OpenMPI faster then MPICH2 at 100 until 1000 data size but at 10000 until 100000 data size OpenMPI and MPICH2 have the same performance. This incident caused merge sort method have many communication time for divide and merge list of random number. This is in accordance with previous research that OpenMPI is able to adjust the communication settings efficiently in accordance with the characteristics of existing infrastructure [6], because in the openMPI architecture there is an MCA as a layer component that provides management services for all other layers [11].

2. Speed-up and efficiency
Figure 4. Graphical comparison OpenMPI and MPICH2 on average execution time for mergesort algorithm

Comparison OpenMPI and MPICH2 on speed-up for merge sort at fig 4, show that OpenMPI and MPICH2 have almost the same performance at more than 10000 data size. The speed-up value of MPICH2 is slightly smaller than that of openMPI. However, as the number of random number increased as large as 100,000, the speed-up value of MPICH2 significantly improved, compared to that of OpenMPI. The speed-up values on both libraries decreased as there were addition of more processors.

3. efficiency

Figure 5. Graphical comparison OpenMPI and MPICH2 on average execution time for mergesort algorithm

Figure 5 described efficiency comparison on both libraries based on the number of nodes. Firstly, the average efficiency rate on program was calculated when using openMPI library. Then, similar process conducted under MPICH2 was also done. The results of those to processes were then compared. Figure 5 showed that efficiency of MPICH2 was less than that of openMPI. Efficiency rates were also decreasing on both libraries as the number of processors increased. The unideal speed-up and efficiency was due to the existence of overhead in parallel system. For example, additional computation which was only required to parallel computing, communication among processors, and synchronization process. Such phenomenon worked on all parallel systems and the speed-up and efficiency trends followed Amdahl's Law. However, speed-up and efficiency would also increase as the data size also getting larger, and that was in parallel with Gustafson's Law. While the addition of processors would decrease efficiency rate, the increase of data size would improve the efficiency [12].

According to Figure four and five, it was clearly seen that as the number of data ranged from 100 up to 10,000, the speed-up and efficiency rate increased as the data grew larger. However, when the random number were as large as 100,000 data, both speed-up and efficiency rare got smaller. Therefore, it was then known that the maximum speed-up and efficiency rate were on 10,000 data size, for all possible test cases scenario, from using two nodes up to five nodes. After conducting this research, it was proven that increasing the size of problem by executing larger data was not necessarily improving speed-up and efficiency. According to Amdal's Law, after some certain limit, speed-up and efficiency would decrease because the existence of larger communication time overhead compared to that of computation time on parallel algorithm. It was also possible due to the unpredicted computation task, imbalance of tasks distribution among processors, the existence of excessive sequential code, or the existence of pure sequential code. In some algorithms, the existence of might reduce the average speed-up time. As the data grew, the execution time both on serial and parallel processing would also increase. As the data grew as large as 100,000, the speed-up decreased because execution time on serial and parallel were getting slower, and therefore the difference between the two were smaller.
4. Conclusion
In this paper discuss about performance between two library communication of MPI to handle task in cluster system. There are MPICH2 AND OpenMPI. The task is sorting number 100 until 100000 data size using mergesort method. Performance parameters use speed-up, efficiency and execution time. The result are when the small data size execute in sequential processing, the performance is better than executing in cluster system using more than one processors / computers. When the data size are large the performance of cluster system more faster than sequential process using one computer or one processor. Execution time of openMPI tends to be faster than that of MPICH2 for small size data, because programs conduct more communication process. However, execution time of MPICH2 tends to be faster as the data gets larger in size, since the increase of computation process reduces the communication effect. The values of openMPI's efficiency rate and speed-up tend to be greater than that of MPICH2 on small size data, ranging from 100 to 10,000 data. In contrary, as the number of data increases, such as 100,000, the values of efficiency rate and speed-up of MPICH2 are better than that of openMPI. Within such scenario, the speed-up difference is 3.1429 and the efficiency rate difference is 0.754. From the result found that implementation library communication of middleware communication have affect in performance of cluster system. Many library communication of middleware communication is in cluster system, but we must to know about suitability between characters of the task and characters of library communication from communication time parameters for optimization of cluster system. Cluster system need communication time to share the task in all members but do not spend long time to communicate between all members because it make decreasing performance, the execution time will be slower, so we need discuss about that in further research.

Acknowledgments
This paper is a part of my research when i was in student of master degree that I developed again. I would like to express my gratitude to my advisors and my team of cluster system for supporting this research. This paper could not be written to its fullest without them, who served as well as one who challenged and encouraged me throughout my time spent studying. they would have never accepted anything less than my best efforts, and for that, I thank them for all support and togetherness they give when we did this research.

Reference
[1] Kahanwal B and Singh T P 2012 The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle International Journal of Latest Research in Science and Technology 1(2) 183-187
[2] Hoefler T, Squyres J M, Mehl T, Mietke F and Rehm W 2005 Implementing a Hardware-based Barrier in Open MPI In Proceedings of 2005 KiCC Workshop, Chemnitzer Informatik Berichte
[3] Hablot L, Glück O, Mignot J C, Genaud S and Primet P V B 2007 Comparison and tuning of MPI implementations in a grid context Université de Lyon INRIA LIP (Laboratoire de l’Informatique du Parallélisme)
[4] Desai N, Lusk A, Bradshaw R and Lusk E 2005 MPISH: A parallel shell for MPI programs. In Proceedings of the 1st Workshop on System Management Tools for Large-Scale Parallel Systems (IPDPS ’05) Denver Colorado USA
[5] Gaurav D, Arora S and Gupta P Effect of parallelization, execution time and inter process communication on sorting techniques using Message Passing Interface International Journal of Computer Applications 101 (5) 0975 – 8887
[6] Graham R L, Shipman G M, Barrett B W, Castain R H, Bosilca G and Lumsdaine A 2006 Open MPI: A High-Performance Heterogeneous MPI (USA: University of Tennessee)
[7] Marszałek Z 2017 Parallelization of Modified Merge Sort Algorithm Symmetry 9(9) 1-18
[8] Eager D L, Zoharjan J and Lazowska E D 1989 Speed-up Versus Efficiency in Parallel Sistems IEEE Transactions On Komputers 38 (3)
[9] Shawish A and Salama M 2014 Cloud Computing: Paradigms and Technologies Studies in Computational Intelligence 495 39-67
[10] ALECU F 2007 performance analysis of parallel algorithms journal of applied quantitative methods 2 (1)
[11] Graham R L, Woodall T S and Squyres J M 2005 Open MPI: A Flexible High Performance MPI Proceedings 6th Annual International Conference on Parallel Processing and Applied Mathematics
[12] Mathew J, Vijayakumar R 2011 The Performance of Parallel Algorithms by Amdahl’s Law, Gustafson’s Trend International Journal of Computer Science and Information Technologies 2 (6) 2796-2799