PARALLEL PROCESSING FOR INTEGRATION PROBLEM USING HIGH PERFORMANCE COMPUTING WITH OPENMPI : CASE STUDY ON THE NUMBER OF INTEGRATION INTERVAL

Teguh Bharata Adji¹), Muhammad Aditya Pradana²), Sujoko Sumaryono³)
Electrical Engineering & Information Technology Department
Engineering Faculty, Gadjah Mada University
Yogyakarta, Indonesia
E-mail : adji@mti.ugm.ac.id¹) adit.pradana@mti.gadjahmada.edu ²)

Abstract
Along with the increasing need to solve various problems of numerical computation effectively and efficiently, the need for a computer system with high computing capability has increased. Computer systems with high computing capability (high performance computing) offers the ability to integrate the resources of multiple computers to solve a problem of numerical computing. This computer system called computer cluster. The cluster must have the ability to perform computing process by using parallel computing mechanism called message passing. In this study, the implementation of message passing mechanism on a computer cluster is done by using Open Message passing Interface (OpenMPI) application.
This study aims to analyze the performance of computer cluster using MPI mechanism in handling the process in parallel computing based on the execution time, speedup, and the efficiency. Parallel computing processes will be executed with OpenMPI application to solve the problems of numerical integration using trapezoidal method.
The research method used in this study is by implementing OpenMPI on a Linux-based cluster system and then analyzing the performance of the system in dealing with parallel computing process to solve the numerical integration problems by increasing the number of intervals used in a numerical integration problem.
The results of this study shows that by increasing number of intervals, the sequential execution time is initially faster than parallel execution time, although finally reduced significantly with the increase of number of integration interval. Instead, the parallel execution time continues to increase rapidly exceeding the sequential execution time.
Keyword : high performance computing, cluster, message passing, parallel computing, numerical integration, OpenMPI, execution time, speedup, efficiency.

I. INTRODUCTION
The increasingly rapid development of technology bring in a variety development of computation concepts. Computation concept was originally a computational process using a single processor. To increase the speed of computing time, the use of multiple processors to handle the computing process is introduced. This concept is known as parallel computing.
The parallel computing process requires a computer with High Performance Computing (HPC) ability to handle effectively and efficiently. This is because the HPC has the ability to fit with the concept of parallel computing itself, such as the ability to integrate resources (processors, memories, disks) of several computers in completing tasks simultaneously (in parallel). This system is called computer cluster. The cluster must have the ability to perform the computing process in parallel with a parallel computing mechanism called message passing or generally known as Message Passing Interface (MPI). Recently most of MPI implementations on a computer cluster uses Open Message Passing Interface (OpenMPI) application.
Cluster ability with MPI mechanism in handling the process of parallel computing, computational and communication processes that occur during the process will be tested to resolve the case of numerical integration. Therefore, the design of a system with high computing capability-based cluster that can perform parallel computing processes to resolve cases and apply the MPI mechanism of the numerical integration is needed. The results of cluster performance will be measured based on execution time, speedup, and efficiency.

2. BASIC THEORY
2.1 Message Passing Interface (MPI)
MPI is independent language of communication protocol used for parallel programs on a computer cluster [8]. MPI is a standard in parallel computing
based on message passing that is widely used in parallel programming environments because of its independent. One of the MPI implementation in a cluster computer networks is the OpenMPI application. There are at least two basic functions or operations to handle messages in MPI, i.e. message delivery operation \texttt{send()} and messages receive operation \texttt{recv()} as shown in Figure 1. With these two commands, a sender process is able to send messages to a receiver and then the receiver process will call a function to retrieve messages that is sent to it. The communication model of send and receive is called point-to-point communication. Through these two basic commands, further communication model that involves more than one process at a time can be developed [14].

Figure 1. Send and receive operations between processes [14]

There is another kind of messages delivery for a group of processes in MPI. This function/operation can be used to sync and send data from one processor to all other processors (broadcast) and to add/sum data from several different processes (reduce). This kind of communication model is called as a collective communication model as shown in Figure 2.

Figure 2. Broadcast operation between processes

2.2 Execution Time

There are many parameters that can be used to measure the performance of parallel system, such as parallel computing speedup, efficiency and execution time [4]. This research will discuss about computation performance that is affected by the cluster architecture as well as the programming code especially for the numerical integration problem. In this study, the program execution time is calculated based on the difference between recorded start time to recorded finish time that is defined in the program. In the sequential program, the program execution time is determined by the computational time, whereas in the parallel program the execution time is the sum of computation time and communication time. In some cases, communication time will dominate the overall execution time. As a result, the parallel execution time will increase [13]. The communications factors that appear on the parallel solutions are usually not seen in sequential solutions and are considered as overheads. Elapsed time of this overhead is highly dependent on the characteristics of the program. Therefore parallel program design must be arranged in such a way to minimize the time overhead, e.g. overlapping computing time with communication time [6].

2.3 Speedup

Speedup is also called acceleration factor, estimates how much speed of multiple processors will increase in solving a problem [14]. These estimates can be calculated using the best solution that can be performed by a single processor, which is the best sequential algorithm on a single processor system as compared to parallel algorithms on a multiprocessor (cluster systems) that are being observed. Speedup ($S_{pp}$) is defined as follows:
In theoretical analysis, speedup can be derived based on the number of computational steps:

\[
S_p = \frac{\text{time process using } n \text{ single processor system (with the best algorithm)}}{\text{time process using } p \text{ processor}}
\]

\[
S_p = \frac{\text{number of computational steps using a single processor}}{\text{number of computational steps using } p \text{ processor}}
\]

2.4 Efficiency
Efficiency is useful to calculate the amount of time needed by the processor to perform computation. Efficiency (E) is defined as follows:

\[
E = \frac{\text{time execution using a single processor}}{\text{time execution using multi processor } \times p \text{ processor}}
\]

or can expressed by:

\[
E = \frac{S_p}{p} \times 100\%
\]

Speedup and efficiency is a function of data size and number of processors. Speedup generally will not increase linearly with the increasing number of processors, but tends to reach a saturation point. In other words, efficiency will decrease if the number of processors increases. This decrease occurs because of overhead in parallel systems, for example, the additional computation required in parallel systems, communication between the processor, and synchronization process. This applies to all parallel systems [6].

2.5 Numerical Integration
Numerical integration problems used in this study is the integral of the exponential function, with the following formula:

\[
\int_{-2}^{2} e^{2x} dx
\]

The numerical integration problems has a lower limit \((x_0)\) value equal to -2 and the upper limit value \((x_n)\) equal to 2. The number of intervals \((n)\) used is 4. Therefore, when

\[
h = \frac{x_n - x_0}{n}
\]

then the length of each interval \((h)\) is 1, as illustrated in Figure 3.

![Figure 3. Graphic of the numerical integration problems](image)

Having the lower limit value \((x_0)\), upper limit \((x_n)\), interval \((n)\), and length of each interval \((h)\), then the numerical integration can be solved as follows:
The cluster is said to be homogeneous because all head node and ten others are used as compute nodes. A cluster system to be used is a homogeneous cluster so the result becomes:

\[ \int_{-2}^{2} x^2 \, dx = 0, \quad (4.3) \]

or can be written as:

\[ \int_{-2}^{2} x^2 \, dx = 0, \quad (4.4) \]

so the result becomes:

\[ \int_{-2}^{2} x^2 \, dx = 0, \quad (4.4) \]

If the value of \( h \) changed to 0.5, the result becomes:

\[ \int_{-2}^{2} x^2 \, dx = 3, \quad (4.4) \]

The actual numerical integration value is 40.97148. It can be seen that when \( h \) value is 1, the error incurred amounted to 50.92%, while when \( h \) value is 0.5, the error will be 13.57%. From the illustration above then it can be concluded that it takes a small value of \( h \) to obtain an accuracy approximations value of the desired integral. This suggests that the \( n \) value used should be large. The greater the value of \( n \) is used, the better integration approximation value resulted in.

3. SYSTEM DESIGN
Cluster system to be used is a homogeneous cluster system based on Linux and is made up of eleven units of computers with one computer is used as a head node and ten others are used as compute nodes. The cluster is said to be homogeneous because all nodes in the cluster have identical hardware specifications.

Cluster built in this study is a dedicated cluster for parallel computing. To build a dedicated cluster for parallel computing, specific requirements and solutions is needed as shown in Table 1. The architecture of the cluster system in this study can be seen in Figure 4.

### Table 1. Requirements and solutions in building Cluster

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>The ability to distribute computing processes</td>
<td>using OpenMPI</td>
</tr>
<tr>
<td>File system that is accessible by all nodes</td>
<td>using OpenMPI</td>
</tr>
<tr>
<td>The mechanism of communication between nodes without the authentication process</td>
<td>using OpenMPI</td>
</tr>
</tbody>
</table>

Figure 4. The cluster system design architecture
4. ANALYSIS AND RESULTS

4.1 Number of integration intervals scalability test scenario

In this scenario, the stand alone computer will execute integration program in sequential process, then the results will be compared with cluster system that work in parallel by performing number of integration intervals scalability. The aim is to see how the computational loads affect the cluster performance in terms of execution time, speedup, and efficiency. The execution time resulted in from the stand alone computer system and cluster can be seen in Table 2.

Table 2. Execution time with the scalability of the number of integration intervals

<table>
<thead>
<tr>
<th>Number of Interval (n)</th>
<th>Sequential (sec.)</th>
<th>Parallel (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.000013</td>
<td>0.003158</td>
</tr>
<tr>
<td>8</td>
<td>0.000017</td>
<td>0.003177</td>
</tr>
<tr>
<td>16</td>
<td>0.000017</td>
<td>0.003184</td>
</tr>
<tr>
<td>32</td>
<td>0.000034</td>
<td>0.003198</td>
</tr>
<tr>
<td>64</td>
<td>0.000057</td>
<td>0.003208</td>
</tr>
<tr>
<td>128</td>
<td>0.000103</td>
<td>0.003215</td>
</tr>
<tr>
<td>256</td>
<td>0.000107</td>
<td>0.003200</td>
</tr>
<tr>
<td>512</td>
<td>0.000382</td>
<td>0.003254</td>
</tr>
<tr>
<td>1024</td>
<td>0.000742</td>
<td>0.003250</td>
</tr>
<tr>
<td>2048</td>
<td>0.001477</td>
<td>0.003319</td>
</tr>
<tr>
<td>4096</td>
<td>0.003132</td>
<td>0.003448</td>
</tr>
<tr>
<td>8192</td>
<td>0.003132</td>
<td>0.003448</td>
</tr>
<tr>
<td>16384</td>
<td>0.013820</td>
<td>0.004028</td>
</tr>
<tr>
<td>32768</td>
<td>0.024542</td>
<td>0.005032</td>
</tr>
<tr>
<td>65536</td>
<td>0.051966</td>
<td>0.007421</td>
</tr>
<tr>
<td>131072</td>
<td>0.105479</td>
<td>0.011706</td>
</tr>
</tbody>
</table>

The increasing number of intervals that are used cause a decrease in sequential execution time generated. Table 2 shows the sequential execution time decreases significantly to 0.1 seconds when the number of intervals used is worth 8192, while the parallel execution time is constantly increasing with the average increase in execution time of 0.0005 seconds.

Figure 5 shows the comparison of execution time generated by a stand alone computer and cluster system with the scalability of the number of intervals.

![Graph of sequential and parallel execution time comparison with scalability of the number of integration intervals](image)

Test result shows that the sequential execution time resulted is still faster than the parallel execution time at the beginning of the program execution process. But with the increasing number of intervals used, the sequential execution time has decreased significantly. On the other side, the acceleration resulted in from parallel execution time is increase exceeds the sequential execution time.

The increasing amount of these intervals showed that the running computational load on both systems is also increasing. The greater the number of intervals, the greater the computational load that must be executed. It can be clearly seen that the parallel execution time is faster than the sequential execution time, when the value of intervals used are more than 16,384.

Table 3. The results of the numerical integration approximation values and the errors resulted in
This is due to the time needed to perform the process of message passing communication and to deliver tasks from the head node to each compute node which was proportional to the computational load in the cluster system. In the mean time, the stand alone computer was not able to handle large computational load in series with quicker process.

The results of this comparison showed that the benefit of parallel execution using the cluster system is clearly visible when running large computational load. It was proven by the results of parallel execution time that was faster than the sequential execution time.

The number of intervals used in integration calculation will affect the resulted integration approximation value. The greater the number of intervals are used, the better numerical integration approximation value resulted in because of smaller error. The results of this numerical integration approximation value achieves a converged state when the number of intervals is more than 65536. The results of the numerical integration approximation value based on the number of intervals used are listed in Table 3.

5. CONCLUSIONS

The conclusions that can be drawn from this research are:

a. The use of a small number of integration intervals in the parallel execution time generated by the cluster system is slower than sequential execution time, although number of nodes increment is done in the parallel execution.

b. With the use of high number of integration intervals, parallel execution time is faster as the number of nodes increase.

c. With the increasing number of intervals used, the sequential execution time that was initially faster than parallel execution time will reduce significantly. On the other hand, the acceleration resulted in by parallel execution time increases and exceeds the sequential execution time.

d. The number of integration intervals will affect the resulted integration approximation value.

e. The number of integration intervals will affect the resulted integration approximation value. The greater the number of intervals are used, the better numerical integration approximation value resulted in because of smaller error.

REFERENCES


