Parallel computations using a cluster of workstations to simulate elasticity problems

2016 J. Phys.: Conf. Ser. 776 012081

(http://iopscience.iop.org/1742-6596/776/1/012081)

Parallel computations using a cluster of workstations to simulate elasticity problems

J. B. B. Darmawan[1] and S. Mungkasi[2]

[1]Department of Informatics, Faculty of Science and Technology,
Sanata Dharma University, Yogyakarta, Indonesia
[2]Department of Mathematics, Faculty of Science and Technology, Sanata Dharma University,
Yogyakarta, Indonesia

E-mail: b.darmawan@usd.ac.id

Abstract. Computational physics has played important roles in real world problems. This paper is within the applied computational physics area. The aim of this study is to observe the performance of parallel computations using a cluster of workstations (COW) to simulate elasticity problems. Parallel computations with the COW configuration are conducted using the Message Passing Interface (MPI) standard. In parallel computations with COW, we consider five scenarios with twenty simulations. In addition to the execution time, efficiency is used to evaluate programming algorithm scenarios. Sequential and parallel programming performances are evaluated based on their execution time and efficiency. Results show that the one-dimensional elasticity equations are not appropriate to be solved in parallel with MPI_Send and MPI_Recv technique in the MPI standard, because the total amount of time to exchange data is considered more dominant compared with the total amount of time to conduct the basic elasticity computation.

1. Introduction

One of mathematics models describing elastic wave propagation in solid media is the elasticity partial differential equation model. This model can be applied in the real world, such as propagation of earthquakes, acoustic waves and waves on an elastic medium. This paper focuses on wave propagation in solid media.

The elasticity model is solved numerically. It may be solved analytically, but it is very difficult to find an explicit form of its exact solution. Therefore, this paper deals with elasticity problems for their numerical solutions.

Usually it is difficult for a single computer to gain high speed computation. A cluster of workstations (COW) often offers economical and efficient approach to addressing this issue. Parallel programming using COW has been proven to improve the speed of computation to simulate flood flows using ANUGA software [1]. Therefore, a parallel programming might speed up the computation of elasticity simulations, compared to the sequential programming. To analyze the results of the parallel computing, we use speed and efficiency variables. This paper investigates if we obtain high speed and efficiency in parallel computations using a cluster of workstation to solve elasticity problems.

This paper is organized as follows. We recall the mathematical model governing elasticity problems and present the numerical scheme to solve the model in Section 2. In Section 3, the numerical method

is implemented in parallel to solve elasticity problems. We provide computational results and discussion in Section 4. Finally, we draw some concluding remarks in Section 5.

2. Mathematical model and numerical scheme

To clarify which mathematical model and numerical scheme used in our work, we recall them in this section.

The mathematical model governing the elasticity problems are [2]

$$\varepsilon_t - u_x = 0, \tag{1}$$

$$u_t - \sigma(\varepsilon)_x = 0. \tag{2}$$

Here t is time variable and x is the space (position) variable, $\varepsilon = \varepsilon(x,t)$ is the strain, $u = u(x,t)$ is the velocity, and $\sigma(\varepsilon)$ is the stress. We consider the space domain $0 \le x \le 100$. We assume that the initial condition is

$$\varepsilon(x,0) = 0, \tag{3}$$

$$u(x,0) = 0, \tag{4}$$

for all x. The boundary condition is

$$\varepsilon(0,t) = 0, \tag{5}$$

$$u(0,t) = \begin{cases} -0.4\left(1+\cos\left(\dfrac{(t-30)\pi}{30}\right)\right) & \text{if} \quad t \le 60, \\ 0 & \text{if} \quad t > 60, \end{cases} \tag{6}$$

$$\varepsilon(100,t) = 0, \tag{7}$$

$$u(100,t) = 0. \tag{8}$$

Note that equations (1)-(2) are conservation laws in the form

$$q_t + f(q)_x = 0, \tag{9}$$

where $q = q(x,t)$ is the conserved quantity and $f(q)$ is the flux function. Conservation laws can be solved using finite volume numerical method.

The fully discrete finite volume scheme for equation (5) is

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(F_{i+1/2}^n - F_{i-1/2}^n), \tag{10}$$

where Q_i^n is the average approximation of the quantity at the i-th cell at the n-th time step, and $F_{i+1/2}^n$ is the average approximation of the flux function at the $(i+1/2)$ interface from the n-th time step to the $(n+1)$-th time step. All approximations are in the finite volume sense. In this paper we use the Lax-Friedrichs flux formulation. All quantities are assumed to have SI units. Numerical computation is conducted in such a way that the numerical method is stable.

3. Method for parallel computations

Conceptually, distributed processing such as COW is not different from distributed memory MIDP according to Flynn's classification, but the communication between processors is slower as they are connected such as by Ethernet, Fiber Channel or ATM [3, 4]. We implement Message Passing Interface (MPI) on Personal Computer cluster to realize COW for parallel programming with the C programming language.

We apply cluster computations using PC's interconnected by network technology. These computations have great advantages, such as high performance for system scalability, rapid adjustment to new technological and low price [5]. The availability of Gigabit Ethernet, which has speed up the communication performance of PC cluster using MPI than ever before (such as Fast Ethernet, Myrinet and Scalable Coherent Interface), supports those advantages [6].

MPI is one of the most popular communication protocols for PC cluster to support parallel programming. It is a library specification for message-passing in the network [7] and one of its implementations is MPICH which is a high performance and widely portable implementation of the MPI standard [8]. As the implementation of MPI, we use MPICH2 which supports the C programming language.

As execution time varied with problem size, Foster [9] proposed efficiency which is independent of problem size to be a metric to evaluate parallel algorithm performance. However, execution time is also an important issue in computations. Therefore, the execution time as well as the efficiency are considered in this paper to evaluate our parallel algorithm scenarios.

The relative speed up S_{relative} is formulated as

$$S_{\text{relative}} = \frac{T_1}{T_{\text{p}}}, \tag{11}$$

where T_1 is the execution time on one processor and T_{p} is the time on p processors. The relative speed up means the factor by which execution time is reduced on p processors. Relative efficiency E_{relative} is defined as

$$E_{\text{relative}} = \frac{T_1}{p \cdot T_{\text{p}}}. \tag{12}$$

From equations (11)-(12), the relative speed up is clearly related to the relative efficiency.

4. Computational results

In this section we present main results of our research. Computations are conducted in a cluster of workstations in Sanata Dharma University. The cluster consists of 16 PC's, one PC has one i3 processor. Note that an i3 processor has 2 cores and 4 threads. All 16 PCs run on Linux operating system. For network communication we use a one Gigabit Ethernet for each PC connected with switch hub one Gigabit. In addition, we have four gigabytes RAM in each PC. We refer to the work of Foster [9] and Yang *et al.* [10] for concepts of parallel programming.

We have conducted 20 simulations for running program in order to solve the elasticity problem mentioned in Section 2. One simulation is run in sequential and the others are run in parallel. We use scenarios of 1, 2, 3, 4, 8, and 16 PCs for our computations. For each scenario, we consider 1, 2, 3, and 4 processes on each PC.

For a basic elasticity problem we record the total time for each simulation in Table 1. As shown in Figure 1, for the total time we observe that more number of processes leads to slower computation. These anomalies occurred, because the more number of processes needs the more amount of time to distribute data from the master process to several worker processes and to collect data from several worker processes to the master process in the MPI_Send and MPI_Recv techniques in the MPI standard. The total amount of time to exchange data is considered more dominant compared with the total amount of time to conduct elasticity computations with the elasticity model for all processes.

Table 1. Total time for an elasticity simulation scenario (in seconds).

Number of PCs	1 process	2 processes	3 processes	4 processes
1	0.8330	0.8680	1.2655	1.4515
2	20.9475	33.4225	46.0460	58.4080
4	49.5995	86.6760	123.4045	160.9290
8	101.1100	186.9505	273.4295	360.1995
16	200.6775	385.0150	570.1225	755.7885

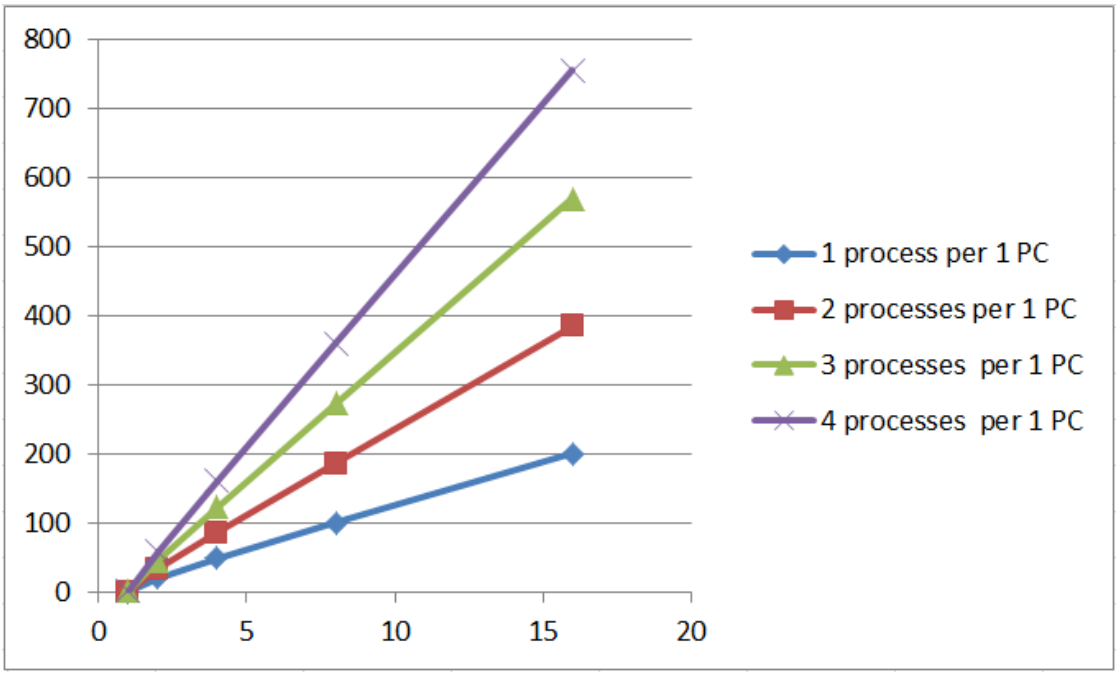

Figure 1. Total time elapsed in a elasticity simulation scenario for several computational settings. The horizontal axis is the number of PCs. The vertical axis is the total time used for computations.

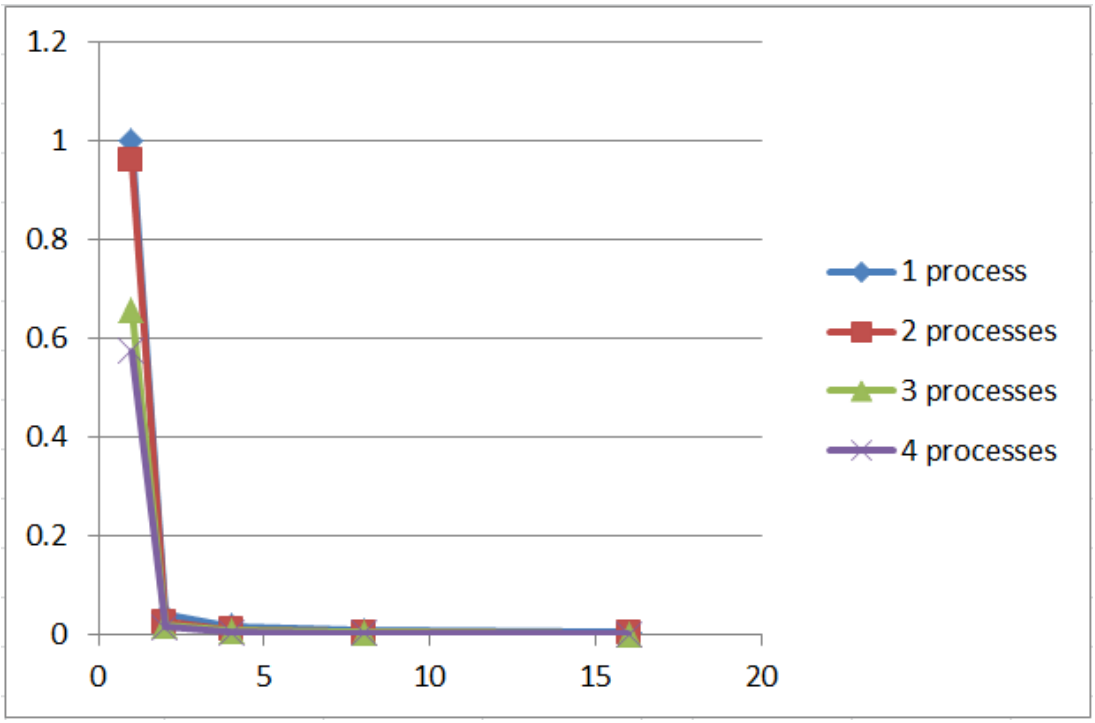

Figure 2. Speed up of parallel computation for an elasticity simulation scenario. The horizontal axis is the number of PCs. The vertical axis is the total time used for computations.

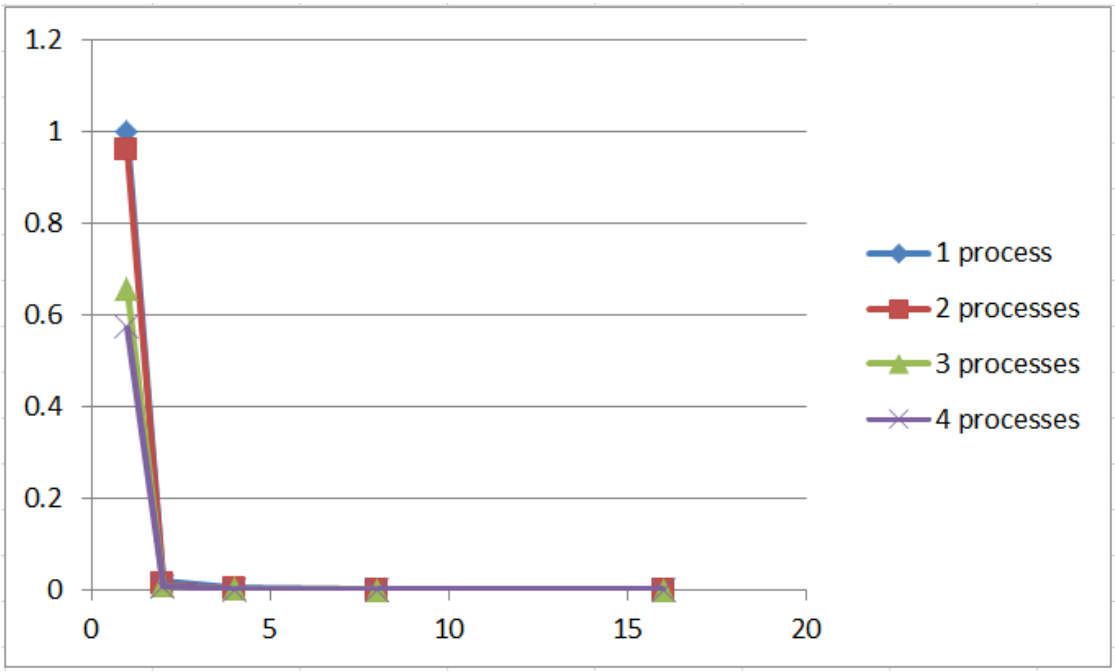

Figure 3. Efficiency obtained from parallel computations for an elasticity simulation scenario. The horizontal axis is the number of PCs. The vertical axis is the total time used for computations.

Figure 2 and Figure 3, for the speed up and the efficiency respectively, show that they get worse, since the number of processes is more than one. This is reasonable because the total time needed for computations also gets worse.

5. Conclusion

We have simulated several scenarios of parallel computations for elasticity problems. We obtain that the one-dimensional elasticity equations are not appropriate to be solved using parallel programming with MPI_Send and MPI_Recv technique in the MPI standard, because the total amount of time to exchange data is considered more dominant compared with the total amount of time to conduct the basic elasticity computation using the finite volume method. Without loss of generality, we recommend that if the total time needed in the data exchange is larger than the total time needed in the basic computations, then the sequential computation may be better to implement than the parallel.

6. Acknowledgments

This work was financially supported by the Institute for Research and Community Services of Sanata Dharma University (LPPM USD). The LPPM USD internal research grant year 2016 is gratefully acknowledged by both authors.

7. References

[1] Mungkasi S and Darmawan J B B 2015 *CCIS* **516** 469
[2] LeVeque R J 2002 *Int. J. Numer. Meth. Fluids* **40** 93
[3] Flynn M J 1972 *IEEE Trans. Comput.* **C-21** 948
[4] van der Steen A J and Dongarra J 2002 *Massive Comp.* **4** 791
[5] Sterling T 2001 *Beowulf Cluster Computing with Linux* (Cambridge: MIT Press)
[6] Mache J 1999 *Proc. 1st IEEE Computer Society Int. Workshop on Cluster Computing* pp 36-42
[7] The Message Passing Interface (MPI) standard http://www.mcs.anl.gov/research/projects/mpi
[8] MPICH http://www.mpich.org
[9] Foster I 1995 *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering* (Boston: Addison-Wesley)
[10] Yang J, Shen W and Yan G 2009 *Proc. 1st Int. Workshop on Education Technology and Computer Science* pp 895-898