International Journal of Engineering & Technology, 8 (1.9) (2019) 230-236

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Openmp and MPI Architectures for Simulating 1D Water Oscillation on Parabolic Domain

P.H. Gunawan*, S. Juliati, M. R. Pahlevi, D. Adytia

School of Computing, Telkom University, Jalan Telekomunikasi No. 1, Terusan Buah Batu Bandung 40257, West Java, Indonesia
**Corresponding author E-mail: phgunawan@telkomuniversity.ac.id*

Abstract

This paper enlightens the simulation of 1D water oscillation on parabolic domain using shallow water equations on multicore processing. Those equation is approached by using finite volume method staggered grid scheme. This scheme is known as a robust scheme for approximating the shallow water equations. Moreover, the scheme is also straightforward to transform into numerical codes. In this paper, the parallel architectures multicore processing OpenMP and MPI are used. The results are shown the CPU time of OpenMP and MPI are better than the serial programming when the number of grids is given more than 200 points. Moreover, the speedup of OpenMP is shown 3 times better than MPI with N_x = 6400 points for both 4 and 8 processors. The maximum of efficiency in the simulation can be achieved around 75% with N_x = 6400 points by 4 cores using OpenMP. However, by 8 cores of processors using OpenMP, the maximum of efficiency is obtained around 60%.

Keywords: *OpenMP, MPI, shallow water, CPU time, multicore process, Oscillation in paraboloid*

1. Introduction

The shallow water equations (SWE) are the system of hyperbolic equations which is also known as the Saint-Venant Equations (Audusse et al., 2004). Some applications of SWE are widely used in fluid flows phenomena. For instance, flooding simulation can be seen in (Stelling and Duinmeijer, 2003), the simulation of sediment change due to water force can be found in references (Gunawan and Lhébrard, 2015; Gunawan et al., 2015), or the simulation of waves in atmosphere can be seen in (Cushman-Roisin and Beckers 2011). The one-dimensional SWE are given as follows

$$\frac{\partial \eta}{\partial t} + \frac{\partial (Hu)}{\partial x} = 0 \tag{1}$$

$$\frac{\partial u}{\partial t} + \frac{\partial (g\eta)}{\partial x} + u\frac{\partial u}{\partial x} = 0 \tag{2}$$

where $\eta(x,t) = H(x,t) - d(x)$ is the elevation water, $H(x,t)$ the positive total water height, $d(x)$ the bathymetry/ topography elevation, $u(x,t)$ the average velocity, g the gravitational force, x and t are space and time respectively (see Fig. 1).

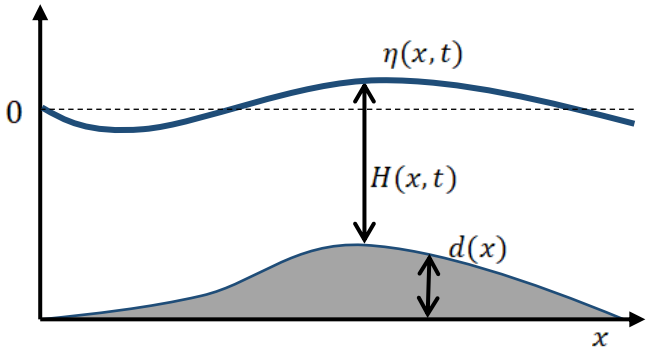

Fig. 1: The SWE configuration.

The equation (1) is called the mass conservation and (2) is known as the momentum conservation. To obtain the numerical solutions of

SWE, some numerical method has been already purposed. For instances, Finite Volume Method (FVM), Finite Element Method (FEM), Spectral Method and ect (see Delestre et al., 2008; Delestre and Marche, 2011; Audusse et al., 2004; Bouchut 2004 for examples). The robust method for approximating SWE is FVM which is proved in mathematical and experiment ways. The eminence of FVM for solving SWE is the capability of method to capture the shock phenomena (see Audusse et al., 2004; Doyen and Gunawan 2014).

In FVM, there are two approaches to obtain the numerical solution. First approach is called FVM collocated scheme (Delestre et al. (2008); LeVeque (2002); Toro (2013)). The scheme allocates the all unknown variables in one full grid point. Then the numerical fluxes are approximated in half grid points. Meanwhile, the second approach is called FVM staggered scheme (Stelling and Duinmeijer, 2003; Doyen and Gunawan 2014). The scheme allocates the velocity variable in the half grid points and the other variables are stored in the full grid points. The FVM staggered scheme calculates the fluxes in upwind ways depend on the velocity direction.

The numerical solution of SWE can be approximated well by huge number of grid points. As concomitant of giving the large number of discrete points, the computational cost to compute solution will increase. Thus the idea to reduce the computational cost by parallel processing is available in some references (Gunawan, 2016; De la Asunción et al., 2016; De La Asunción et al., 2011; Brodtkorb et al., 2012 and Castillo et al., 2013). This idea is simple, dividing the grid points into some block tasks which is contained some parts of discrete points. Then, the computational procedures for mass and momentum balances can be done for each blocks by different processors.

In this paper, the OpenMP and MPI multicores architecture will be elaborated for simulating 1D water oscillation in paraboloid. The numerical scheme of FVM staggered will be used due to the robustness of the scheme for simulating dry-wet bed simulations (Stelling and Duinmeijer, 2003; Doyen and Gunawan 2014). Moreover, the performance of each parallel platforms will be elaborated in order to see the capability of each platforms for the proposed problem. Thus this paper is organized as follows, in the next sections, the numerical scheme of FVM staggered scheme is presented. After that, about the architecture of parallel platforms OpenMP and MPI are given. Additionally, numerical simulation and parallel performance for each platforms are elaborated. Finally, the conclusion is shown in the last section of paper.

2. Numerical Scheme

Consider the control volume for staggered grid scheme in the discrete space point k on the domain $\Omega = [-L, L]$ is given in Fig. 2. There are two control volumes in staggered grid scheme, for mass conservation is presented in full grid and momentum conservation is presented half grid. Full grid is defined on $(x_{k-1/2},\ x_{k+1/2})$ and half grid is defined on $(x_{k-1},\ x_k)$. The space is discretized by a natural number N_x such that $x_k = -L + (k \times \Delta x)$ with the space step $\Delta x = (L + L)/N_x$ and $k \in \mathcal{M} = \{0, 1, 2, \cdots, N_x\}$. Similarly, the time is discretized by a time step Δt thus the discrete space t^n can be defined as $t^n := n \times \Delta t$ with $n \in \tau = \{0, 1, 2, \cdots\}$.

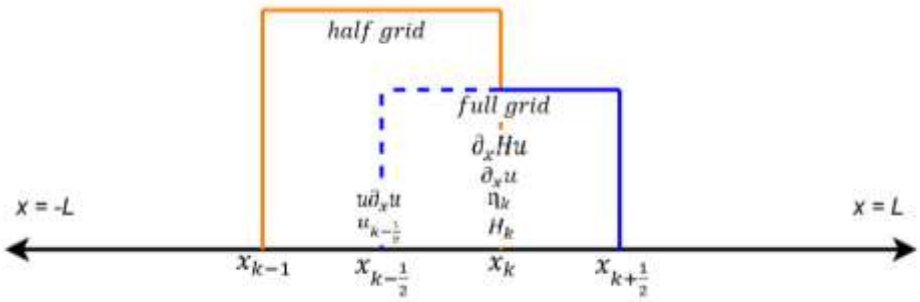

Fig. 2: The control volume of staggered grid scheme.

The dicretization of shallow water equations (1) in staggered grid scheme is given as

$$\frac{\eta_k^{n+1} - \eta_k^n}{\Delta t} + \frac{q_{k+\frac{1}{2}}^n - q_{k-\frac{1}{2}}^n}{\Delta x} = 0, \qquad \forall k \in \mathcal{M}, \qquad n \in \tau \tag{3}$$

Then, the flux $q_{k\pm\frac{1}{2}}^n$ will be approached by $q_{k\pm\frac{1}{2}}^n = \widehat{H}_{k\pm\frac{1}{2}}^n u_{k\pm\frac{1}{2}}^n$ with

$$\widehat{\boldsymbol{H}}_{\boldsymbol{k-\frac{1}{2}}}^{\boldsymbol{n}} = \begin{cases} \boldsymbol{H_{k-1}^n}, & \textbf{if} \quad \boldsymbol{u_{k-\frac{1}{2}}^n > 0} \\ \boldsymbol{H_k^n}, & \textbf{otherwise} \end{cases}.$$

This equation approximate the water depth H at half grid depend on the sign of velocity, although variable H only defined at full grid. For the positive velocity, the flow is assumed flowing from left to the right and vice versa. Thus, if the velocity $u_{k-\frac{1}{2}}^n > 0$, then the information of H should be conducted from grid $k-1$ or otherwise be taken from grid k.

In this paper, Neumann boundary conditions $q_{-\frac{1}{2}}^n = q_{\frac{1}{2}}^n$ and $q_{Nx+\frac{1}{2}}^n = q_{Nx-\frac{1}{2}}^n$ are used. The interesting part of FVM staggered scheme is how to approximate the advection equation $u(\partial u/\partial x)$. In paper of (Stelling and Duinmeijer, 2003) the advection is approximated by

$$\frac{\partial\left(\frac{1}{2}u^2\right)}{\partial x} \approx \frac{1}{\bar{H}_{k+\frac{1}{2}}^{n+1}}\left[\frac{\left(\bar{q}_{k+\frac{1}{2}}^n u_{k+\frac{1}{2}}^n - \bar{q}_{k-\frac{1}{2}}^n u_{k-\frac{1}{2}}^n\right) - u_{k+\frac{1}{2}}^n\left(\bar{q}_{k+\frac{1}{2}}^n - \bar{q}_{k-\frac{1}{2}}^n\right)}{\Delta x}\right] \tag{4}$$

where $\bar{H}_{k+\frac{1}{2}}^{n+1} = (H_k^{n+1} + H_{k+1}^{n+1})/2$ is the average of new water height at half point $x + \frac{1}{2}$. The dry wet procedure in FVM staggered is very simple, if $\bar{H}_{k+\frac{1}{2}}^{n} = 0$ then there are no water flow at point $x + \frac{1}{2}$, thus $\frac{\partial\left(\frac{1}{2}u^2\right)}{\partial x}$=0 the scheme forced the velocity to zero. Finally, the discretization of momentum balance (2) is given as follows

$$\frac{u_{k+\frac{1}{2}}^{n+1} - u_{k+\frac{1}{2}}^{n}}{\Delta t} + g\frac{\eta_{k+1}^{n} - \eta_k^{n}}{\Delta x} + \frac{\partial\left(\frac{1}{2}u^2\right)}{\partial x} = 0 \tag{5}$$

This scheme is called semi-implicit scheme staggered scheme since the new water height is used in (5). The stability of this scheme is given as

$$\Delta t \leq \nu \frac{\Delta x}{\max\limits_{i\in\mathcal{M}}\left(\frac{\left|q_{k+1/2} + q_{k-1/2}\right|}{2h_k^n} + \sqrt{gh_k^n}\right)} \tag{6}$$

where $0 < \nu \leq 1$ is the Courant number (see (Doyen and Gunawan 2014) for more detail).

3. Parallel Architecture

There are two different architectures in multicore parallel processing. First architecture is known as the shared memory architecture. This architecture permits each processors to use the main memory for communicating of data. Moreover, programming shared memory architecture incriminates the programming codes in shared memory which can be executed by each processors. OpenMP is an example of platform shared memory architecture. OpenMP is a simple instruction to divide a problem into some subproblems which are called threads. OpenMP can be integrated in several programming language such as C/C++ and Fortran.

Second architecture is known as the distributed memory architecture. In this architecture, programming codes should be delivered privately to each processors. Thus each processor will execute the code using each own local memory. Then, final result for each processors should be distributed to the master processor in order to gather all results. The example platform of this architecture is MPI (Message Passing Interface). Similar to OpenMP, MPI can be integrated to programming languages C/C++ and Fortran.

The performance of parallel programming can be conducted by two measurements which are called speedup and efficiency. The speedup can be obtained by

$$S(p) = \frac{T_1}{T_p} \tag{7}$$

where $S(p)$ is speedup by using parallel programming using p processor, T_1 the CPU time by using single processor, and T_p CPU time by using p processor. Therefore, efficiency can be computed by

$$E = \frac{S(p)}{p} \times 100\% \tag{8}$$

In order to use OpenMP and MPI parallel architecture for solving shallow water equations (1-2) using FVM staggered scheme (3-5), the diagram of block process should be constructed for each architectures. The diagram of block process using OpenMP and MPI can be seen in Fig. 3 and 4 respectively. In Fig. 3, the diagram shows two domains of computation, in serial and parallel using OpenMP. In serial domain, the declaration of all variables and looping time command are located. Meanwhile, the computation of mass and momentum balances which are consist of several loops will be executed in parallel domain.

The programming code of OpenMP is simple and straightforward, for instance in C/C++, the instruction only add syntax "#pragma omp parallel for" above the looping codes. And then, the single core will automatically divide the loop into several block process and then ordered other cores to execute the blocks.

Similar to the block process of OpenMP, the block processor of MPI in Fig. 4 has two domains of computation. The difference part from block process of OpenMP is located on looping time. In MPI, the looping time is designed to be executed in each processors. The master processor will update the local time variable for every time steps, and then will send it to the others processors. Thus the other processors will be executed time loop similar to the master.

In this process, each processors need communication to each other for transferring data (initial or results). Here, MPI needs commands "MPI_Send()" and "MPI_Recv()" to send and receive data respectively. Note that, using MPI needs more communication time than the shared memory architecture. Thus, MPI could not guarantee produces best execution time.

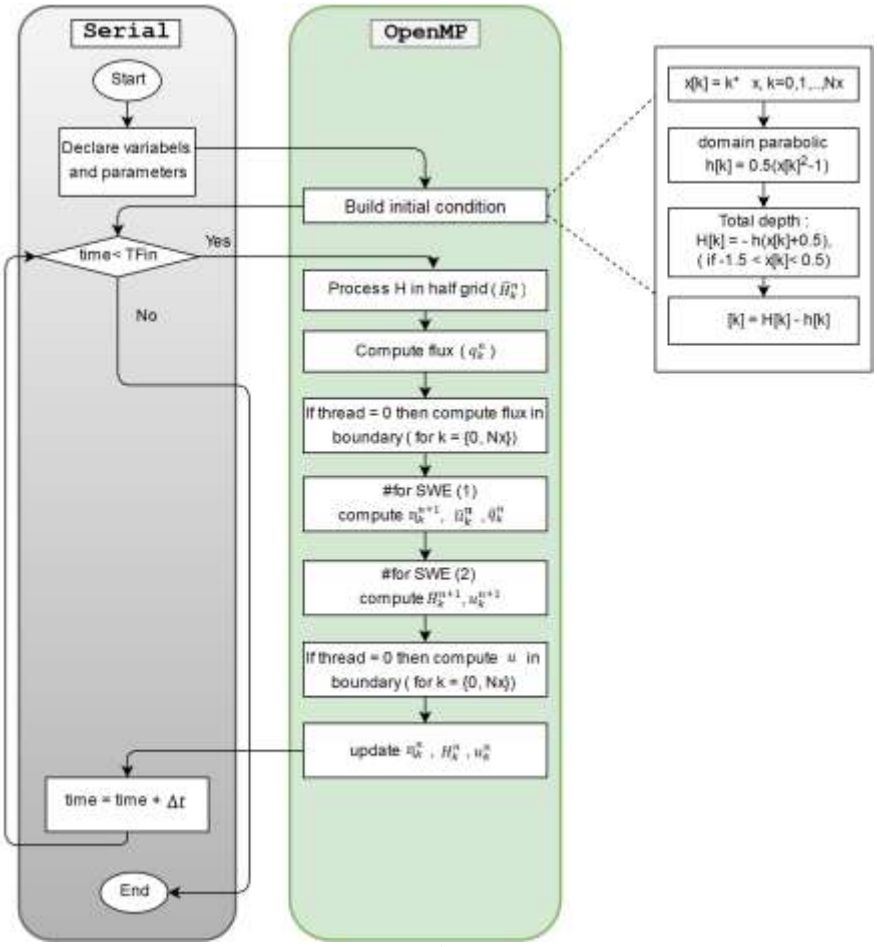

Fig. 3: Block process of OpenMP architecture

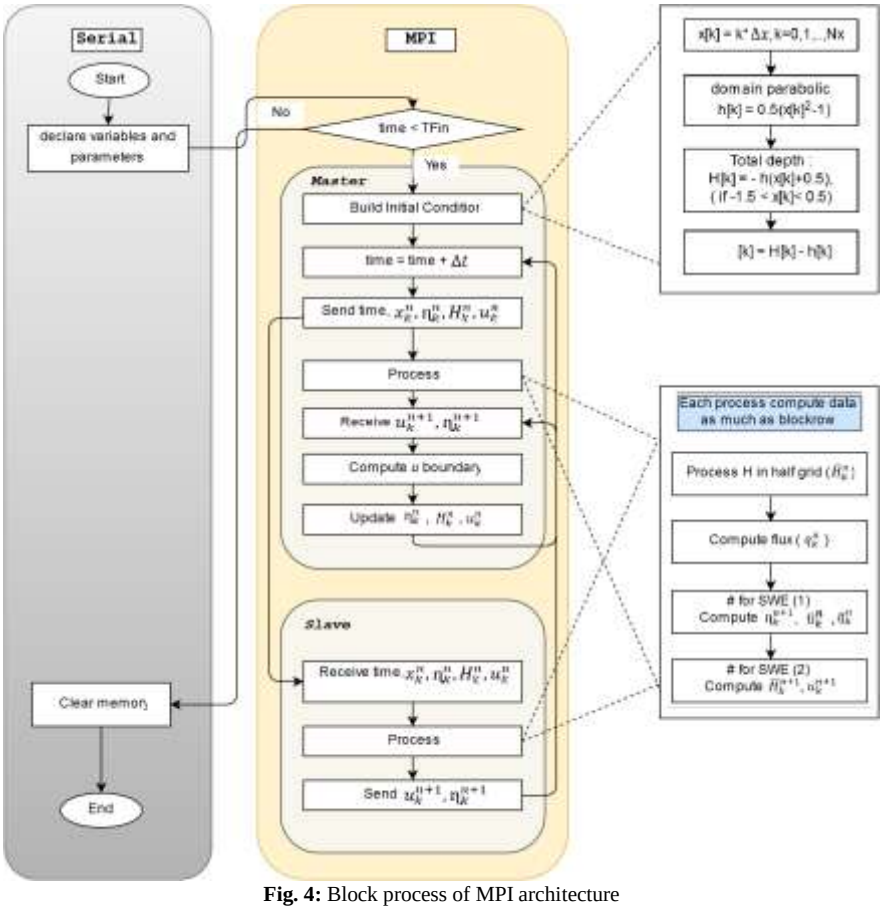

Fig. 4: Block process of MPI architecture

4. Numerical simulation and parallel performance

To obtain the numerical simulation and parallel performance, the following computer specifications are used.

Table 1: The Computer Specifications

Name	Type
Operating System	Centos 6.5
Processors	Intel (R) Xeon(R) CPU E5-2670 v3 @2.30Ghz
RAM	8 GB
CPU(s)	48

This computer is a part of supercomputer in HPC laboratory Ilmu Komputasi, School of Computing, Telkom University.

Numerical simulation

In this section, the numerical simulation of oscillation in paraboloid is presented. The initial configurations of simulation is given as follows

$$d(x) = 0.5\,x^2 - x + 0.5,$$

$$H(x,0) = \begin{cases} -0.5\,d(x+0.5), & \text{if} \quad -1.5 \le x \le 0.5 \\ 0, & otherwise \end{cases},$$

$$u(x,0) = 0.$$

The result of this simulation are shown in Fig. 5. In Fig. 5 (Left), the water level $\eta(x,t)$ at various final time numbers are given. The solid line is shown the initial of water level, and the other lines show water level at each final time numbers. Moreover, the topography of paraboloid domain is shown by the grey area. Fig. 5 (right) shows the ability of the numerical scheme can handle the dry-wet simulation. The accuracy of the numerical solution of this problem compared to the analytical solution using staggered scheme can be found in (Doyen and Gunawan, 2014).

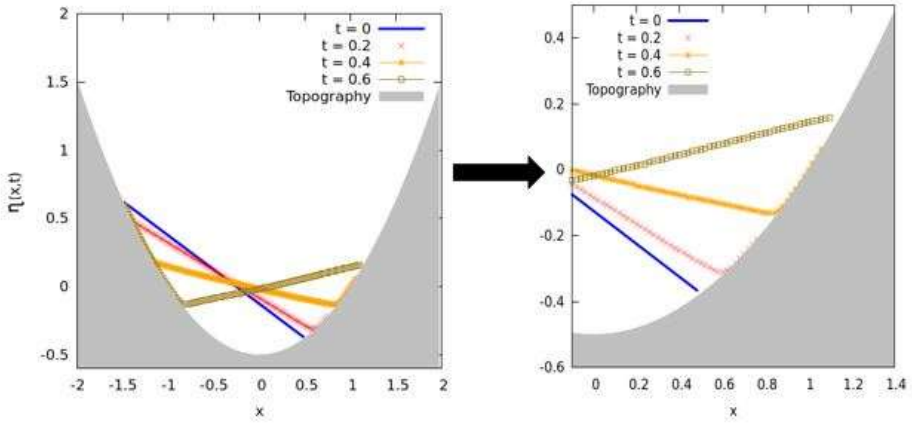

Fig. 5: (Left) Water level η(x,t)at various final time numbers. (Right) The zoom part of left domain.

In next section, the performance of parallel processing for this simulation will be elaborated. The performance will be conducted by examined the CPU time for various grid numbers.

Parallel performance

In this section, the performance of OpenMP and MPI will be elaborated. First, the CPU time for OpenMP and MPI from the simulation using final time 15 seconds will be shown.

Table 2: The CPU time

Number of grid	Serial (s)	CPU time (s) of OpenMP		CPU time (s) of MPI	
		4 cores	8 cores	4 cores	8 cores
200	4.7016	9.1220	11.8547	6.37066	10.2485
400	18.5854	21.1377	19.5061	15.3937	27.1289
800	73.9065	48.4992	48.8219	46.8245	66.4478
1600	295.5170	131.1910	111.1010	147.1210	196.5140
3200	1183.0300	431.2220	311.4390	881.1690	614.9810
6400	4765.9300	1578.8800	999.0920	2939.8800	3674.9700

Table 2 shows the comparison of CPU time in serial parallel using 4 and 8 cores for each platforms. Form the Table 2, the CPU time in serial for number of grids 200 and 400 is smaller than the CPU time in parallel. This means the parallel processing is not efficient since the number of grid points is small, thus using single core is already enough. However, using the large number of points (from 800 to 6400 points), the CPU time in parallel is better than the serial.

From Table 2, the speedup (7) and efficiency (8) performances of OpenMP and MPI using 4 and 8 cores can be achieved. The speedup and efficiency profiles are shown in Figs. 6 and 7 respectively. In Fig. 6, the speedup of OpenMP is better than MPI in general. However, the speedup using MPI shown slightly better in number of grids 200 until 800 using 4 cores (see Fig. 6 left). The MPI fails to get the good speedup performance due to the time for passing variables in processors is dominant than the computing time. Thus the speedup of MPI is shown decreasing after using 1600 and 3200 points for 4 and 8 cores respectively.

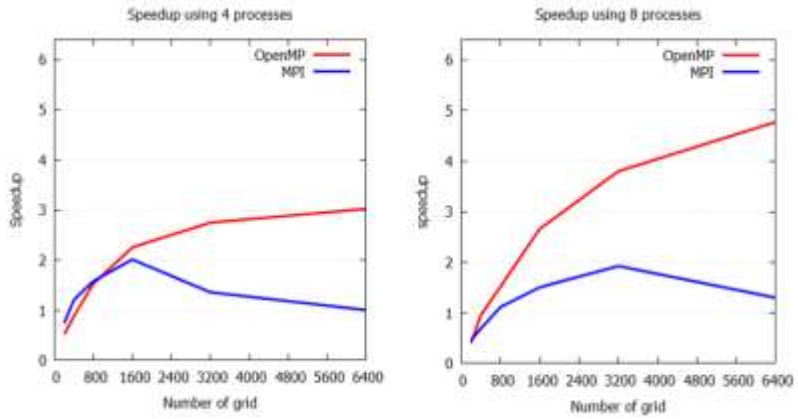

Fig. 6: Speedup graph of OpenMP and MPI using 4 (left) and 8 (right) cores.

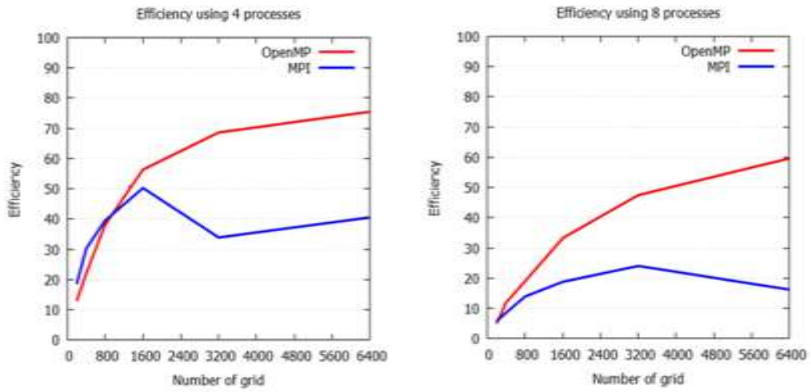

Fig. 7: Efficiency of OpenMP and MPI using 4 (left) and 8 (right) cores.

The efficiency profiles in Fig. 7 using 4 and 8 cores are given. The efficiency of parallel processing using 4 cores is better than 8 cores in both platforms. This means that the increasing the number of cores no guarantee to get better efficiency in the simulation.

5. Conclusion

Two parallel architectures OpenMP and MPI have been implemented for simulating the 1D shallow water oscillation in paraboloid domain. The finite volume method staggered scheme has been used to approximate the solution of shallow water equations. The results show the CPU time of OpenMP and MPI are better than the serial programming when the number of grids is more than 200 points. Moreover, the speedup of OpenMP is shown 3 times better than MPI with $N_x = 6400$ points for both 4 and 8 processors. The maximum of efficiency in the simulation can be achieved around 75% with $N_x = 6400$ points by 4 cores using OpenMP. However, by 8 cores of processors using OpenMP, the maximum of efficiency is obtained around 60%. This paper is expected to show the important of parallel processing in the numerical area. Specially, in numerical method for fluid dynamics which needs the high computation in a huge discrete points.

References

[1] Audusse, E., Bouchut, F., Bristeau, M.-O., Klein, R., and Perthame, B. (2004). A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. SIAM Journal on Scientific Computing, 25(6):2050– 2065.

[2] Bouchut, F. (2004). Nonlinear stability of finite Volume Methods for hyperbolic conservation laws: And Well-Balanced schemes for sources. Frontiers in Mathematics. Birkhäuser Verlag, Basel.

[3] Brodtkorb, A. R., Sætra, M. L., and Altinakar, M. (2012). Efficient shallow water simulations on gpus: Implementation, visualization, verification, and validation. Computers & Fluids, 55:1–12.

[4] Castillo, D., Ferreiro, A. M., García-Rodríguez, J. A., & Vázquez, C. (2013). Numerical methods to solve PDE models for pricing business companies in different regimes and implementation in GPUs. Applied Mathematics and Computation, 219(24), 11233-11257.

[5] Cushman-Roisin, B. and Beckers, J.-M. (2011). Introduction to geophysical fluid dynamics: physical and numerical aspects, volume 101. Academic Press.

[6] De la Asunción, M., Castro, M. J., Mantas, J. M., & Ortega, S. (2016). Numerical simulation of tsunamis generated by landslides on multiple GPUs. Advances in Engineering Software, 99, 59-72.

[7] De La Asunción, M., Mantas, J. M., & Castro, M. J. (2011). Simulation of one-layer shallow water systems on multicore and CUDA architectures. The Journal of Supercomputing, 58(2), 206-214.

[8] Delestre, O., Cordier, S., James, F., and Darboux, F. (2008). Simulation of rain-water overland-flow. In 12th International Conference on Hyperbolic Problems, volume 67, pages 537–546. American Mathematical Society.
[9] Delestre, O., & Lagrée, P. Y. (2013). A well-balanced finite volume scheme for blood flow simulation. International Journal for Numerical Methods in Fluids, 72(2), 177-205.
[10] Delestre, O. and Marche, F. (2011). A numerical scheme for a viscous shallow water model with friction. Journal of Scientific Computing, 48(1-3):41–51.
[11] Doyen, D. and Gunawan, P. H. (2014). An explicit staggered finite volume scheme for the shallow water equations. In Finite Volumes for Complex Applications VII-Methods and Theoretical Aspects, pages 227–235. Springer.
[12] Gunawan, P. H. (2016). Scientific parallel computing for 1d heat diffusion problem based on openmp. In Information and Communication Technology (ICoICT), 2016 4th International Conference on, pages 1–5. IEEE.
[13] Gunawan, P. H., and Lhébrard, X. (2015). Hydrostatic relaxation scheme for the 1D shallow water-Exner equations in bedload transport. Computers & Fluids, 121, 44-50.
[14] Gunawan, P. H., Eymard, R., and Pudjaprasetya, S. R. (2015). Staggered scheme for the Exner–shallow water equations. Computational Geosciences, 19(6), 1197-1206.
[15] LeVeque, R. J. (2002). Finite volume methods for hyperbolic problems, volume 31. Cambridge university press.
[16] Stelling, G. S. and Duinmeijer, S. A. (2003). A staggered conservative scheme for every froude number in rapidly varied shallow water flows. International Journal for Numerical Methods in Fluids, 43(12):1329–1354.
[17] Toro, E. F. (2013). Riemann solvers and numerical methods for fluid dynamics: a practical introduction. Springer Science & Business Media.