LONTAR KOMPUTER VOL. 11, NO. 1 APRIL 2020
DOI : 10.24843/LKJITI.2020.v11.i01.p01
Accredited B by RISTEKDIKTI Decree No. 51/E/KPT/2017

p-ISSN 2088-1541
e-ISSN 2541-5832

OpenMP Performance in Numerical Simulation of Dambreak Problem Using Shallow Water Equations

P. H. Gunawan

School of Computing, Telkom University
Jl. Telekomunikasi No. 1, Terusan Buah Batu, Bandung 40257, Indonesia
phgunawan@telkomuniversity.ac.id

Abstract

Numerical simulation of water surface waves is widely used to describe water flow and its impact on human life. For instance, numerical simulation of waves is elaborated to simulate Tsunami as an early warning system. Using a numerical approach, the study of water flow will reduce costs and save time compared with the conventional approach (in the laboratory). Shallow water equations (SWE) is one of the mathematical models which can be used to describe water flow. In the numerical simulation of SWE, the finite volume method is a robust method to approximate SWE. The result of using a numerical approach depends on the number of grids. The high number of grids then the smooth solution can be obtained. However, an increasing number of grids lead to an increase in computational cost. In this paper, parallel computing using the OpenMP platform is given to reduce the computational cost of numerical simulation. In parallel computing performances, Speedup and Efficiency of numerical simulation using 6400 grids points are obtained four times and 51%, respectively. Moreover, by several numbers of cores from 2 to 8, the CPU time of parallel computing is shown decreasing along with the increasing number of computer cores.

Keywords: *Parallel computing, OpenMP, Shallow water equation, Simulation, Numerical*

1. Introduction

Dynamical movement of surface waves can be modeled using the various models. The simple mathematical wave model to describe wave movement dynamically is known as Shallow water equations (SWE). This model is widely used in describing fluid flow problem, such as flow in canal, river, lakes, etc. or it can be used to simulate Tsunami phenomena as an early warning system (see [1, 2, 3] for more detail). Model SWE is a system of hyperbolic equations which consists of two equations (mass and momentum conservation). In one dimension space, SWE is given as follows.

$$\frac{\partial h}{\partial t} + \frac{\partial (hu)}{\partial x} = 0, \tag{1}$$

$$\frac{\partial (hu)}{\partial t} + \frac{\partial \left(hu^2 + \frac{1}{2}gh^2\right)}{\partial x} = 0. \tag{2}$$

where $h(x,t)$ describes water height, $u(x,t)$ describes average velocity, g shows gravitational coefficient, moreover x and t are space and time, respectively.

To solve (1 - 2) numerically, one robust method can be used, which is called the finite volume method (FVM) [4, 5, 6]. FVM is widely used to approximate the hyperbolic type of equations in the numerical problem. Generally, there are two types of approach in FVM, staggered grid and collocated grid model. The detail of these two numerical models can be found in some references [1, 6, 7, 8]. As shown in [6] and [8], FVM collocated and staggered grid model are satisfying mathematical properties of shallow water equations, i.e., preserve positivity of water height, satisfy the well-balanced condition, etc.

Mathematically, a good approximation result depends on the size of space steps or grids. This size is obtained by dividing the domain space into several discrete spaces[9]. Indeed, increasing the number of grids causes high computational cost for approximating (1 - 2). In numerical scheme of (1 - 2), two equations (mass and momentum) will be approximated. Therefore, the process of approximating two equations needs long time execution using a large number of grids.

Here, computational cost can be minimized by applying computer science techniques which is called parallel computing. In this case, computation tasks are optimized using several cores in a single computer. Several references, as in [9, 10, 11, 12] and [13], show the ability of parallel computing for tackling computational cost in the numerical approach. In this paper, the goal of this paper is to implement multi-cores parallel computing in a collocation scheme for SWE. Moreover, the numerical simulation of the dry-wet dam-break problem will be elaborated to investigate the performance of parallel computing.

In order to complete this paper, in Section 2 a brief introduction of FVM collocated scheme with HLLC flux for SWE. In Section 3, the parallel algorithm of numerical scheme is given. The numerical results and parallel performances are provided in Section 4. The conclusion of this paper is shown in Section 5.

2. Numerical Scheme

For simplicity, SWE (1 - 2) can be rewritten in the following compact form,

$$U_t + F(U)_x = 0 \tag{3}$$

where

$$U = (h, hu)^T, \tag{4}$$

$$F(U) = \left(hu, hu^2 + \frac{1}{2}gh^2\right)^T. \tag{5}$$

In FVM, the spatial and time domain is discretized into several control volumes. For instance, in Figure 1 a control volume V_k is given at point k. This control volume is defined on $(x_{k-1/2}, x_{k+1/2}) \times (t^n, t^{n+1})$. Consider computational domain of simulation is $\Omega = [0, L] \times [0, T]$, then the following discrete properties can be defined as,

- point $x_k = k \times \Delta x$ with the space step $\Delta x = L/N_x$ and $k \in \mathcal{M} = \{0, 1 \cdots, N_x\}$,
- point $t^n = n \times \Delta t$ with $\Delta t = T/N_t$ and $n \in \mathcal{T} = \{0, 1, \cdots\}$,

where N_x and N_t are the number of discrete points of spatial and time, respectively.

Let's $U_k^n, k \in \mathbb{Z}, n \in \mathbb{N}$ be a discrete value of solution SWE (3), then it can written as

$$U_k^n :\approx \int_{V_k} U(x, t^n) \quad \mathrm{d}x, \quad \forall k \in \mathcal{M}, n \in \mathcal{T}. \tag{6}$$

Therefore in FVM collocated scheme, the discretization of SWE is given as

$$\frac{U_k^{n+1} - U_k^n}{\Delta t} + \frac{F_{k+\frac{1}{2}}^n - F_{k-\frac{1}{2}}^n}{\Delta x} = 0, \quad \forall k \in \mathcal{M}, n \in \mathcal{T} \tag{7}$$

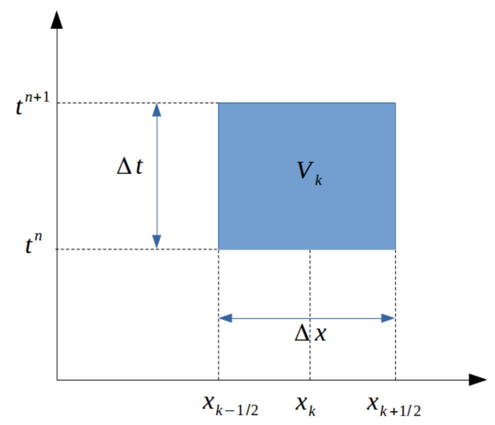


Figure 1. The visualization of control volume in FVM.

where flux $F_{i\pm\frac{1}{2}}$ will be approximated using the numerical flux which called HLLE (Harten, Lax, van Leer and Einfeld) and given as

$$F^n_{k+\frac{1}{2}} = \mathfrak{F}(U^n_k, U^n_{k+1}) = a_1 F(U^n_{k+1}) + a_2 F(U^n_k) - a_3(U^n_{k+1} - U^n_k), \tag{8}$$

where $F(U_k)$ is numerical flux function (5). Meanwhile, coefficients a_1 and a_2 are given as follows,

$$a_1 = \frac{\min(\lambda_2, 0) - \min(\lambda_1, 0)}{\lambda_2 - \lambda_1} \quad a_2 = 1 - a_1 \quad a_3 = \frac{\lambda_2|\lambda_1| - \lambda_1|\lambda_2|}{2(\lambda_2 - \lambda_1)} \tag{9}$$

The coefficients λ_1 and λ_2 can be obtained in some references, for instance, see [4, 14, 15]. Thus the discretization (7) can be rewritten as

$$U^{n+1}_k = U^n_k - \frac{\Delta t}{\Delta x}\left(\mathfrak{F}(U^n_k, U^n_{k+1}) - \mathfrak{F}(U^n_{k-1}, U^n_k)\right), \quad \forall k \in \mathcal{M}, n \in \mathcal{T} \tag{10}$$

Note that numerical form (10) is under stability condition, which is given by the following condition

$$\frac{\Delta t}{\nu} \leq \frac{\Delta x}{\max\limits_k \left(|u_k| + \sqrt{gh_k}\right)} \tag{11}$$

with $0 < \nu \leq 1$ is called Courant number.

3. Parallel Architecture

Parallel computing can is a computational procedure that is to compute several tasks of computation simultaneously. This type of computing can be done by a single computer with multi-cores or multiple computers. One popular platform in multi-cores parallel computing is called OpenMP (Open Multi-Processing). This platform is a shared memory multiprocessing programming type and can be used in several programming languages like C/C++, Fortran, etc.

For example, in [9], parallel computing using the OpenMP platform is shown success to reduce computational time for solving the 1D heat equation. Moreover, OpenMP is shown as simple and straightforward in application. The performance of OpenMP depends on the specification of the computer. In this paper, two measurements of parallel performance metrics will be elaborated.

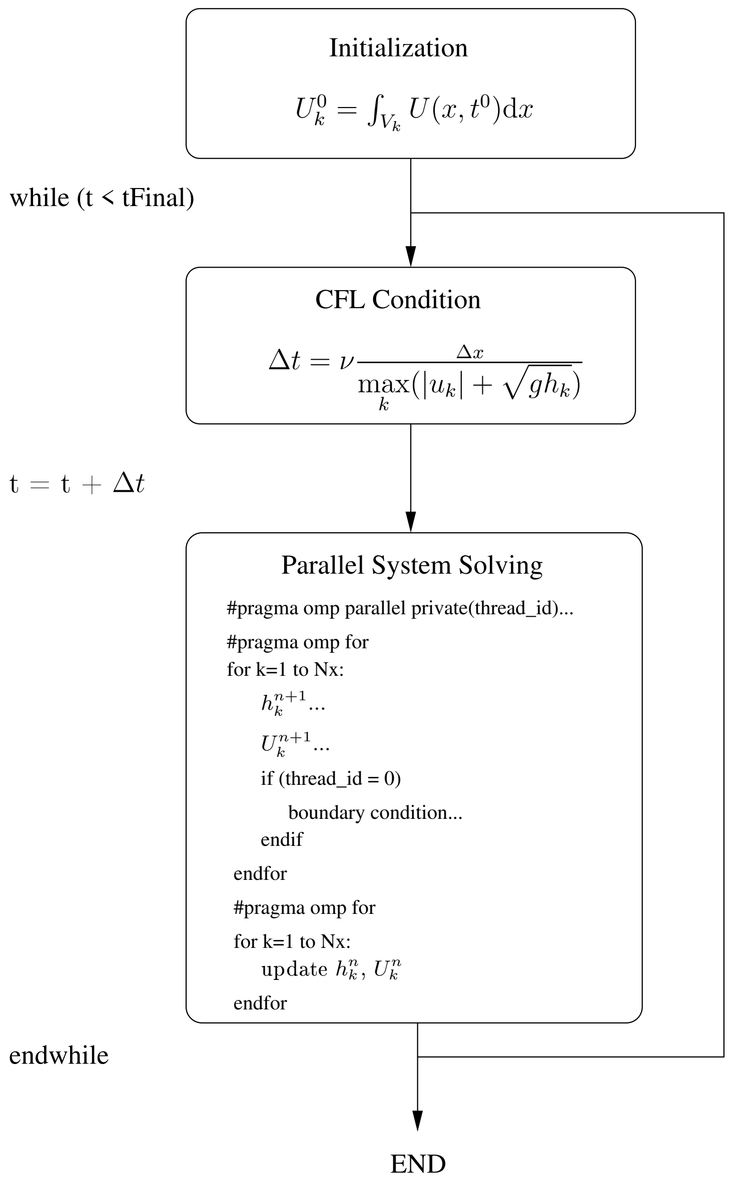
Here speedup and efficiency metrics will be given. The speedup can be obtained by

$$S(p) = \frac{T_1}{T_p}, \tag{12}$$

Where T_1 and T_p are CPU time for serial and parallel, respectively. Where p describes the number of cores that are used for computing. Meanwhile, the efficiency of parallel computing can be computed as

$$E(p) = \frac{S(p)}{p} \times 100\%. \tag{13}$$

In this paper, the numerical method (7) will be computed in parallel computing. Therefore the numerical algorithm is given for simplicity. A numerical algorithm for computing (7) in parallel can be seen in Figure 2.


Initialization
$U_k^0 = \int_{V_k} U(x, t^0)\mathrm{d}x$
while (t < tFinal)
CFL Condition
$\Delta t = \nu \frac{\Delta x}{\max_k(|u_k| + \sqrt{gh_k})}$
$\mathrm{t} = \mathrm{t} + \Delta t$
Parallel System Solving
endwhile
END

```
#pragma omp parallel private(thread_id)...
#pragma omp for
for k=1 to Nx:
    $h_k^{n+1}$...
    $U_k^{n+1}$...
    if (thread_id = 0)
        boundary condition...
    endif
endfor
#pragma omp for
for k=1 to Nx:
    update $h_k^n$, $U_k^n$
endfor
```

Figure 2. A numerical algorithm for solving (7) in parallel.

Here, the numerical algorithm in parallel is given in two areas, in serial and parallel computing. As shown in Figure 2, serial computing can be done in the initialization process of U_k^0 and in defining CFL condition. Since these two processes are not fit in parallel computing. Meanwhile, parallel computing with OpenMP can be started in the inner loop stage, which is to compute (7) by defining the water height and velocity variable. Note that, the numerical algorithm in serial is similar to the Figure 2, where OpenMP is not applied in the parallel area.

4. Numerical Results and Parallel performances

To obtain results of numerical simulation and parallel implementation, the following specification of the computer is given in Table 1.

Table 1. The computer specifications for numerical simulation and parallel implementation

Name	Type
Operating System	Centos 6.5
Processors	AMD 2 socket @4 cores
RAM	8 GB

4.1. Numerical Simulation of Dry and Wet Dambreak

Dambreak problem is very popular in numerical simulation of SWE. This problem produces shock phenomena, which is a big challenge for the numerical scheme to tackle discontinuity solution [14]. Here two problems are given in dry-wet bed of dambreak. The following initial configuration of dambreak in dry bed problem in the spatial domain $[0, 1]$ is given as follows

$$h(x,0) = \begin{cases} 0, & \text{if } x \geq 0.5 \\ 1, & \text{otherwise} \end{cases}, \tag{14}$$

$$h(x,0)u(x,0) = 0. \tag{15}$$

Meanwhile, for wet dambreak problem is shown as

$$h(x,0) = \begin{cases} 0.2, & \text{if } x \geq 0.5 \\ 1, & \text{otherwise} \end{cases}, \tag{16}$$

$$h(x,0)u(x,0) = 0. \tag{17}$$

The difference between dry and wet bed simulation is located on the right side of the dam wall (in this case at $x = 0.5$). Numerical results of dambreak simulation with h and u profile are shown in Figure 3.

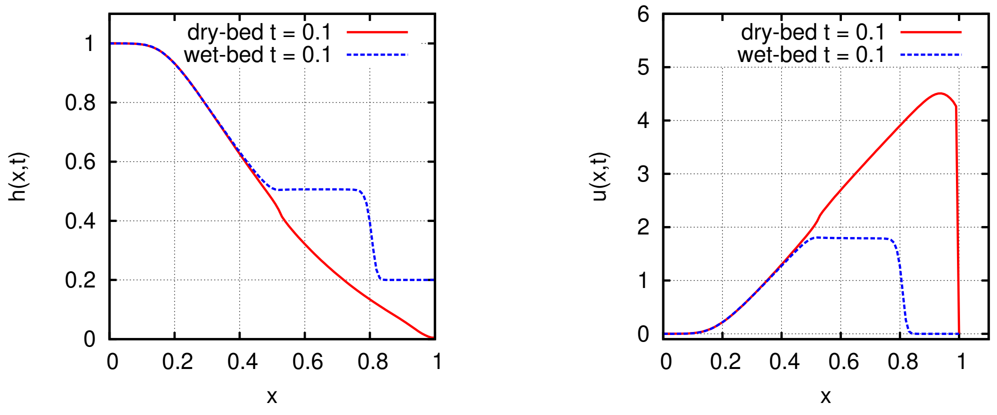


Figure 3. Numerical simulation of dry and wet bed at simulation time $T = 0.1$ s.

As can be shown in Figure 3, the results of numerical simulation of the dry-wet bed are well elaborated. These results are similar to the analytical solution of dry-bed dam-break simulation by SWASHES software, which can be found in [16]. Here in Figure 3 (left), the water height profile for the wet-bed produces shock near $x = 0.8$ due to different energy of different water height. This phenomenon is satisfying Rankine-Hugoniot relation in mathematical observation [14].

4.2. Parallel Implementation

In this section, the performance of OpenMP for simulating dambreak problems is given. First, the comparison of CPU time for both numerical simulations (wet and dry dam-break) can be seen in Figure 4. Moreover, serial and parallel of CPU time are shown for both problems. Here, several numbers of grid size are elaborated to see OpenMP performance, in this case $N_x \in \{200, 400, 800, 1600, 3200, 64000\}$.

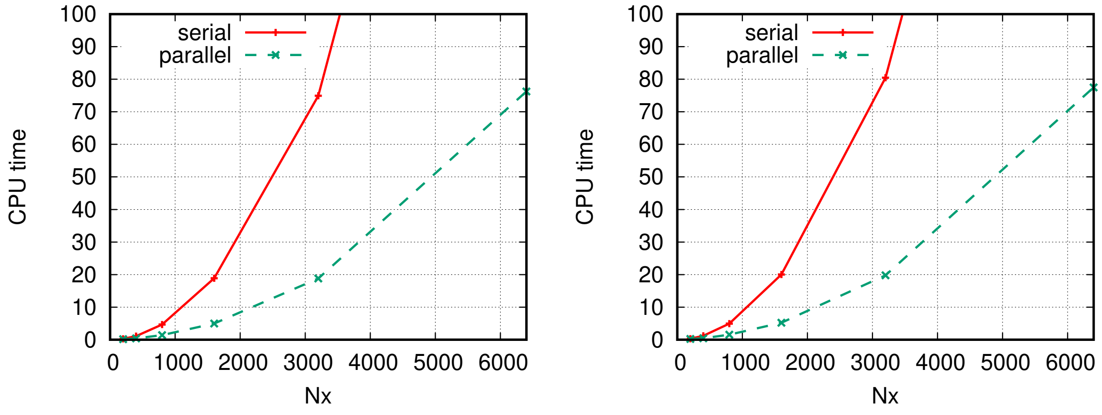


Figure 4. Performance result: CPU time in serial and parallel for dry (left) and wet (right) dam-break.

Here in parallel implementation, the number of the processor for computing is eight cores. From Figure 4, a similar profile of CPU time can be seen for both numerical simulations. However, it can be seen that for both problems, similar CPU time in serial computing with grids number $N_x = 3600$ and in parallel computing with $N_x = 64000$ can be seen. This can be observed that the OpenMP platform is successfully applied, and it can reduce the computational cost of serial code.

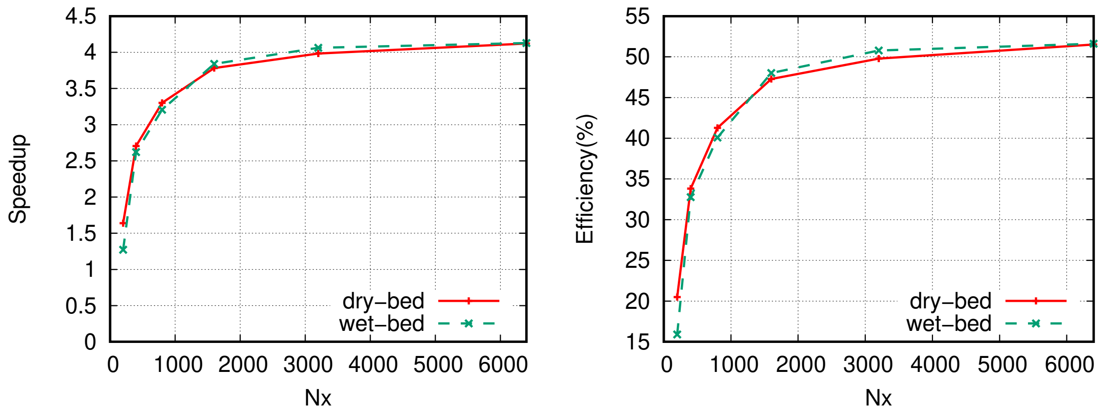


Figure 5. Performance result: Speedup (left) and Efficiency (right) of dry-wet dam-break.

Another parallel performance metrics, speedup, and efficiency are shown in Figure 5. These performance metrics are used to see how fast and efficient parallel computing in reducing the computational cost. As shown in Figure 5 (left), the speedup of parallel computing for both problems is reaching four times of serial computing. Moreover, since eight cores are used in this experiment, then the efficiency of parallel computing is approximately 51%, which is shown in Figure 5 (right). This means that only 51% of the average computational cost in serial code can be reduced. Since as we can see in the numerical algorithm of parallel (see Figure 2 for more detail), not all areas of computation can be parallelized. Some areas are still shown in serial computation. These performances are obtained from equations (12) and (13).

p-ISSN 2088-1541
e-ISSN 2541-5832

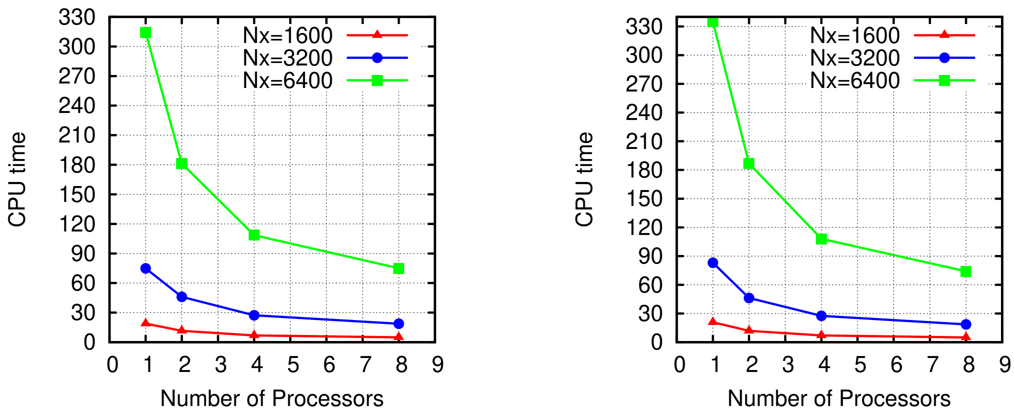


Figure 6. The CPU time for some numbers of core in dry (left) and wet (right) bed dam-break.

For another addition, numerical simulation of parallel computing sung several numbers of cores (2, 3, 4, 8) are also elaborated. The results in dry wet dam-break problems can be seen in Figure 6. As shown in Figure 6, the increasing number of cores from 2 to 8, resulting in decreasing of CPU time. Indeed, the increasing number of cores causing some tasks are executed faster than using the low number of cores. And this result is shown for both problems. Indeed from Figure 6, an increasing number of cores into large numbers could not guarantee CPU time is always decreasing since efficiency factor becomes an obstacle in multicore parallel programming.

5. Conclusion

Parallel computing performances for simulating dry-wet dam-break problem using OpenMP and shallow water equation have been done. Two numerical simulations of the dam-break problem also have been well elaborated. Here, OpenMP is shown satisfying to reduce CPU time in several numbers of grid in simulation. Speedup of simulation using parallel computing is shown able to reach four times of serial computing. Moreover, the efficiency of numerical simulation using eight cores is obtained approximately 51%, with the number of the grid is $N_x = 6400$.

References

[1] G. S. Stelling and S. A. Duinmeijer, "A staggered conservative scheme for every froude number in rapidly varied shallow water flows," *International Journal for Numerical Methods in Fluids*, vol. 43, no. 12, pp. 1329–1354, 2003.

[2] B. Cushman-Roisin and J.-M. Beckers, *Introduction to geophysical fluid dynamics: physical and numerical aspects*. Academic Press, 2011, vol. 101.

[3] O. Delestre and P.-Y. Lagrée, "A well-balanced finite volume scheme for blood flow simulation," *International Journal for Numerical Methods in Fluids*, vol. 72, no. 2, pp. 177–205, 2013.

[4] O. Delestre, S. Cordier, F. James, and F. Darboux, "Simulation of rain-water overland-flow," in *12th International Conference on Hyperbolic Problems*, vol. 67. American Mathematical Society, 2008, pp. 537–546.

[5] O. Delestre and F. Marche, "A numerical scheme for a viscous shallow water model with friction," *Journal of Scientific Computing*, vol. 48, no. 1-3, pp. 41–51, 2011.

[6] E. Audusse, F. Bouchut, M.-O. Bristeau, R. Klein, and B. Perthame, "A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows," *SIAM Journal on Scientific Computing*, vol. 25, no. 6, pp. 2050–2065, 2004.

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

[8] D. Doyen and P. H. Gunawan, “An explicit staggered finite volume scheme for the shallow water equations,” in *Finite Volumes for Complex Applications VII-Methods and Theoretical Aspects*. Springer, 2014, pp. 227–235.

[9] P. H. Gunawan, “Scientific parallel computing for 1d heat diffusion problem based on openmp,” in *Information and Communication Technology (ICoICT), 2016 4th International Conference on*. IEEE, 2016, pp. 1–5.

[10] M. de la Asunción, M. Castro, J. Mantas, and S. Ortega, “Numerical simulation of tsunamis generated by landslides on multiple gpus,” *Advances in Engineering Software*, vol. 99, pp. 59–72, 2016.

[11] M. De La Asunción, J. M. Mantas, and M. J. Castro, “Simulation of one-layer shallow water systems on multicore and cuda architectures,” *The Journal of Supercomputing*, vol. 58, no. 2, pp. 206–214, 2011.

[12] A. R. Brodtkorb, M. L. Sætra, and M. Altinakar, “Efficient shallow water simulations on gpus: Implementation, visualization, verification, and validation,” *Computers & Fluids*, vol. 55, pp. 1–12, 2012.

[13] D. Castillo, A. Ferreiro, J. A. García-Rodríguez, and C. Vázquez, “Numerical methods to solve pde models for pricing business companies in different regimes and implementation in gpus,” *Applied Mathematics and Computation*, vol. 219, no. 24, pp. 11 233–11 257, 2013.

[14] R. J. LeVeque, *Finite volume methods for hyperbolic problems*. Cambridge university press, 2002, vol. 31.

[15] E. F. Toro, *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

[16] O. Delestre, C. Lucas, P.-A. Ksinant, F. Darboux, C. Laguerre, T.-N. Vo, F. James, S. Cordier *et al.*, “Swashes: a compilation of shallow water analytic solutions for hydraulic and environmental studies,” *International Journal for Numerical Methods in Fluids*, vol. 72, no. 3, pp. 269–300, 2013.