New & Forthcoming Titles

Home > New & Forthcoming Titles

Communications in Computer and Information Science

Barbosa, S., **Chen**, P., **Cuzzocrea**, A., **Du**, X., **Filipe**, J., **Kara**, O., **Kotenko**, I., **Sivalingam**, K.M., **Slezak**, D., **Washio**, T., **Yang**, X. (Eds.)

ISSN: 1865-0929

FOR AUTHORS AND EDITORS

- CCIS Forthcoming Proceedings List
- Author and Editor Instructions
- Copyright Form

SERVICES FOR THE SERIES

- Contacts
- Download Product Flyer
- OCS - Online Conference Service
- CCIS volumes at SpringerLink

Contents

Invited Paper

Fuzzy Logic and Control System

Genetic Algorithm and Heuristic Approaches

Artificial Intelligence and Machine Learning

Similarity-Based Models

Classification and Clustering Techniques

Intelligent Data Processing

Feature Extraction

Image Recognition

Visualization Techniques

Intelligent Network

Cloud and Parallel Computing

Strategic Planning

Intelligent Applications

Intelligent Systems for Enterprise, Government and Society

Fast and Efficient Parallel Computations Using a Cluster of Workstations to Simulate Flood Flows

Sudi Mungkasi[1] and J. B. Budi Darmawan[2]

[1] Department of Mathematics, Faculty of Science and Technology,
Sanata Dharma University, Mrican, Tromol Pos 29, Yogyakarta 55002, Indonesia
sudi@usd.ac.id

[2] Department of Informatics, Faculty of Science and Technology,
Sanata Dharma University, Mrican, Tromol Pos 29, Yogyakarta 55002, Indonesia
b.darmawan@usd.ac.id

Abstract. A strategy is proposed for fast and efficient parallel computations using a cluster of workstations to simulate flood flows. We implement ANUGA software to conduct flood simulations. ANUGA solves the two-dimensional shallow water equations using a finite volume method. ANUGA can be run in either sequential or parallel. We focus our work on parallel computations. Our computation scenarios are implemented on a cluster of workstations. Parallel computation results are assessed based on execution time and efficiency. The strategy that we propose gives a significant advantage when we want to get fast and efficient computations using a cluster of workstations.

Keywords: ANUGA software, flood flow, parallel programming.

1 Introduction

Flood flow simulations are needed for an early warning system before the real event. The system warns the community when there in an indication of an upcoming flood. To decide whether a warning should be issued or not, simulations of floods should be conducted beforehand.

Fast and efficient flood simulations are always desired in addition to accurate results [1–3]. In terms of computation, we need to balance between speed, efficiency, and accuracy of simulations. Accuracy of simulations has been discussed thoroughly in a number of references, such as Guinot [5], LeVeque [8] and Mungkasi [11]. In contrast, speed and efficiency of flood flow simulations still need to be enhanced. This paper limits the discussion to speed and efficiency of flood flow simulations using ANUGA software.

To gain a high speed computation, running a simulation sequentially is not the best option. Therefore, we run simulations in parallel. Speed and efficiency are considered and analysed in order to assess parallel computation results. This paper gives a strategy to get fast and efficient parallel computations using a cluster of workstations to simulate flood flows.

The rest of this paper is organised as follows. We recall the mathematical model governing water flows in Section 2. The model is solved numerically in Section 3.

© Springer-Verlag Berlin Heidelberg 2015

R. Intan et al. (Eds.): ICSIIT 2015, CCIS 516, pp. 469–477, 2015.
DOI: 10.1007/978-3-662-46742-8_43

The numerical method is implemented in parallel into ANUGA software, as described in Section 4. We provide computational results and discussion about the results in Section 5. Finally, we draw some concluding remarks in Section 6.

2 Governing Equations

In this section we recall the mathematical model governing shallow water flows. We refer to Mungkasi and Roberts [10] for the description of the model. More detailed explanation of the shallow water equations can be found in other work, such as [5, 15, 17].

The vector form of the two-dimensional shallow water equations is

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x + \mathbf{g}(\mathbf{q})_y = \mathbf{s}\,. \tag{1}$$

Here $\mathbf{q} = [h\ uh\ vh]^T$ is the vector consisting of water depth h, x-momentum uh, and y-momentum vh. Variables u and v represent velocities in the x- and y-direction respectively. Functions $\mathbf{f}(\mathbf{q})$ and $\mathbf{g}(\mathbf{q})$ are fluxes in the x- and y-direction respectively, where

$$\mathbf{f}(\mathbf{q}) = \begin{bmatrix} uh \\ u^2h + \frac{1}{2}gh^2 \\ uvh \end{bmatrix} \quad \text{and} \quad \mathbf{g}(\mathbf{q}) = \begin{bmatrix} vh \\ vuh \\ v^2h + \frac{1}{2}gh^2 \end{bmatrix}\,. \tag{2}$$

The right hand side $\mathbf{s}$ of (1) is the source term given by

$$\mathbf{s} = \begin{bmatrix} 0 \\ -gh\,(z_x + S_{fx}) \\ -gh\,(z_y + S_{fy}) \end{bmatrix} \tag{3}$$

where $z(x, y)$ is the bed topography, and $S_f = \sqrt{S_{fx}^2 + S_{fy}^2}$ is the bed friction following Manning's resistance law

$$S_{fx} = \frac{u\eta^2\sqrt{u^2+v^2}}{h^{4/3}} \quad \text{and} \quad S_{fy} = \frac{v\eta^2\sqrt{u^2+v^2}}{h^{4/3}}. \tag{4}$$

The constant η is the Manning resistance coefficient. The notation g represents the acceleration due to gravity.

We integrate (1) over an arbitrary closed and connected spatial domain Ω having boundary Γ and apply the Gauss divergence theorem to the flux terms to get the integral form

$$\frac{\partial}{\partial t}\int_{\Omega} \mathbf{q}\,d\Omega + \oint_{\Gamma} \mathbf{F}\cdot\mathbf{n}\,d\Gamma = \int_{\Omega} \mathbf{s}\,d\Omega\,. \tag{5}$$

Here $\mathbf{F} = [\mathbf{f}(\mathbf{q})\ \mathbf{g}(\mathbf{q})]^T$ is the flux vector, $\mathbf{n} = [\cos(\theta)\ \sin(\theta)]^T$ is the outward normal vector of the boundary, and θ is the angle between $\mathbf{n}$ and the x-direction.

The shallow water wave equations admit the rotational invariance property, which implies [17]

$$\mathbf{F} \cdot \mathbf{n} = \mathbf{T}^{-1}\mathbf{f}(\mathbf{T}\mathbf{q}) \tag{6}$$

where $\mathbf{T}$ is the transformation matrix

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} . \tag{7}$$

Therefore, equation (5) can now be written as

$$\frac{\partial}{\partial t}\int_{\Omega} \mathbf{q}\, d\Omega + \oint_{\Gamma} \mathbf{T}^{-1}\mathbf{f}(\mathbf{T}\mathbf{q})\, d\Gamma = \int_{\Omega} \mathbf{s}\, d\Omega\,. \tag{8}$$

This integral form (8) of the shallow water equations helps in the construction of a numerical method that we discuss in the next section.

3 Numerical Method

We solve the shallow water equations numerically using a finite volume method. The spatial domain is discretised into a finite number of triangular cells.

The semi-discrete scheme of the finite volume method corresponding to equation (8) is

$$\frac{d\mathbf{q}_i}{dt} + \frac{1}{A_i} \sum_{j\in\{0,1,2\}} \mathbf{H}_{ij}\, l_{ij} = \mathbf{s}_i\,. \tag{9}$$

Here $\mathbf{q}_i$ is the vector of conserved quantities averaged over the ith cell, A_i is the area of the ith triangle (cell), $\mathbf{s}_i$ is the source term associated with the ith cell, $\mathbf{H}_{ij}$ is the outward normal flux of material across the ij edge, and l_{ij} is the length of the ij edge. The ij edge is the interface between the ith and jth cells. Equation (9) is implemented on each triangular cell. The summation in sigma notation of equation (9) goes for $j \in \{0, 1, 2\}$, as each cell has three edges. Readers interested in the algorithm to compute the numerical flux $\mathbf{H}_{ij}$ should consult the work of Roberts et al. [10, 15].

The finite volume method (9) has been implemented into ANUGA software. ANUGA is a free and open source software developed by the Australian National University (ANU) and Geoscience Australia (GA). It can be downloaded from its official website `http://anuga.anu.edu.au`. It is available to be installed on Windows and Linux machines. ANUGA can be run in either sequential or parallel. It has been widely used for simulating flood inundations [6, 7, 12–14].

4 Parallel Programming

In this section we compare flood simulation results from several scenarios of parallel computations. We implement MPI Personal Computer cluster for parallel programming with ANUGA. Parallel implementation of ANUGA is described in [19].

We are interested in cluster computations using PCs interconnected by network technology. These computations have great advantages, such as high performance for low price, system scalability, and rapid adjustment to new technological advances [16]. The availability of Gigabit Ethernet supports those advantages. Gigabit Ethernet has speed up the communication performance of PC cluster using MPI than ever before such as Fast Ethernet, Myrinet and Scalable Coherent Interface [9]. One of the most popular communication protocols for PC cluster to support parallel programming is Message Passing Interface (MPI). MPI is a library specification for message-passing in the network [20] and one of its implementations is MPICH. MPICH is a high performance and widely portable implementation of the MPI standard (see [21] for more details about MPICH).

ANUGA is developed based on Python programming language. It supports parallel programming with MPI using pypar as the interface between MPI and Python. OPENMPI and MPICH2 are supported by pypar [19]. We use MPICH2 for our parallel computations as the implementation of MPI.

Foster [4] proposed efficiency to be a metric to evaluate parallel algorithm performance. They [4] noted that execution time varied with problem size, but efficiency is independent of problem size. However execution time is also an important issue in computations. Therefore, in this paper we consider the execution time as well as the efficiency to evaluate our parallel algorithm scenarios.

The relative speedup S_{relative} is formulated as

$$S_{\text{relative}} = \frac{T_1}{T_p}\,, \tag{10}$$

where T_1 is the execution time on one processor and T_p is the time on P processors. The relative speedup means the factor by which execution time is reduced on P processors. Relative efficiency E_{relative} is defined as

$$E_{\text{relative}} = \frac{T_1}{PT_p}\,. \tag{11}$$

Note that equation (10) is equivalent to $S_{\text{relative}} = PE_{\text{relative}}$. This means that the speedup is known if and only if the efficiency is known.

5 Numerical Results

In this section we present main results of our research.

For a preliminary illustration, we present simulation results of the Cairn tsunami flood inundations. The numerical setting is available in the current ANUGA software. Representatives of simulation results are given in Figures 1 and 2. Figure 1 shows the initial condition of the Cairn coastal region in Australia before an incoming flow from the ocean exists. At time $t > 0$ we assume that there is a fixed incoming flow from the ocean, which results in flood inundations around the coastal region. This is illustrated in Figure 2 showing the situation at time $t = 5000$ seconds simulated real time.

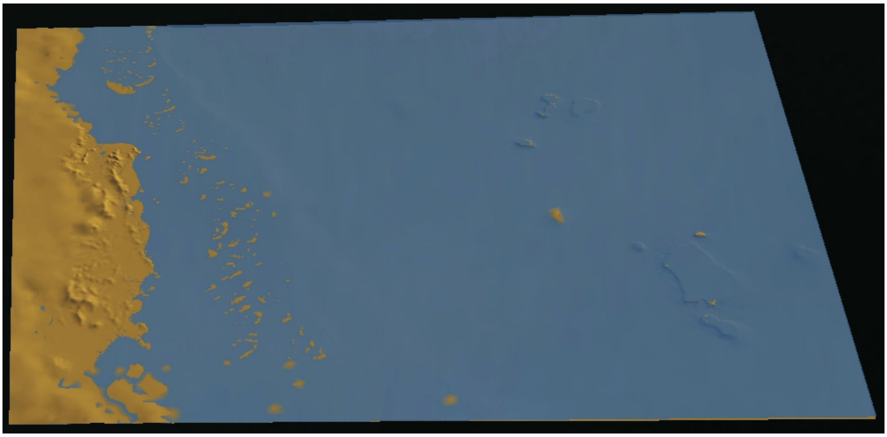

Fig. 1. Initial condition of the Cairn coastal region. Before the simulation starts, water is at rest.

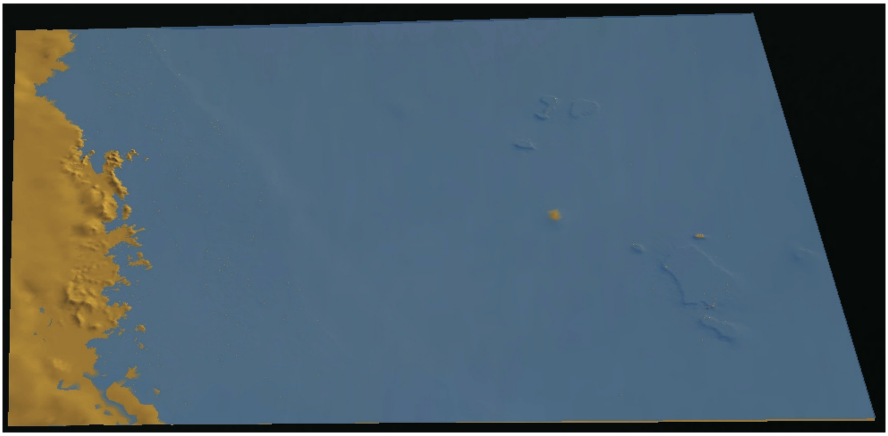

Fig. 2. Simulation result after 5000 seconds of a fixed incoming water flow from the ocean boundary. Flood inundation occurs at the Cairn coastal region.

Now we focus on our main results of parallel computations for flood flows. Simulations are conducted in a cluster of workstations in Sanata Dharma University. The cluster consists of 16 PCs with i3 processors, that is one PC has one i3 processor. Note that an i3 processor has 2 cores, and an i3 processor has 4 threads. When we simulate flood flows for our research, all 16 PCs are turned on with Linux operating system. For the network communication we use a one Gigabit Ethernet. In addition, for the RAM, we have four gigabytes (4 GB RAM) in each PC. We refer to the work of Foster [4] and Yang et al. [18] for concepts of parallel programming.

We have conducted 20 simulations for running ANUGA software in order to simulate flood flows. One simulation is run in squential. The others are run in parallel. We consider scenarios of 1, 2, 4, 8, and 16 PCs for our computations. Using MPICH we can distribute the computation processes in parallel. For each scenario, we consider 1, 2, 3, and 4 processes on each PC.

For a flood flow problem we record the total time for each simulation in Table 1. We see that running the simulation into either 3 processes or 4 processes does not make the computation faster than just 1 or 2 processes. This is reasonable because an i3 processor has only 2 cores even though it has 4 threads. The number of cores is more crucial in computations. The total time for all simulations are viewed in Figure 3.

Table 1. Total time for a simulation scenario using 1 Gbps

Number of PCs	1 process	2 processes	3 processes	4 processes
1	2828.41	2255.50	2455.60	2505.65
2	1653.14	1311.79	1370.35	1457.07
4	1060.57	819.40	806.83	785.04
8	710.20	495.09	578.32	550.70
16	431.46	383.81	472.92	543.72

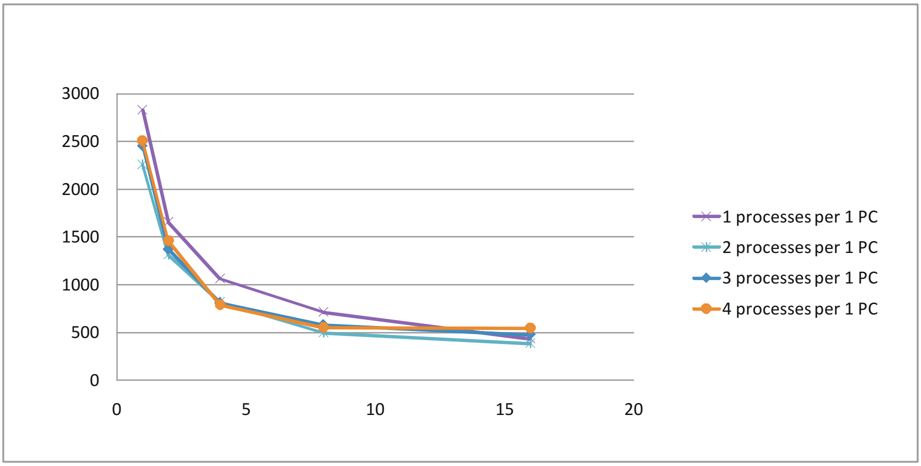

Fig. 3. Total time elapsed in a simulation scenario for several computational settings. The horizontal axis is the number of PCs. The vertical axis is the total time used for computations.

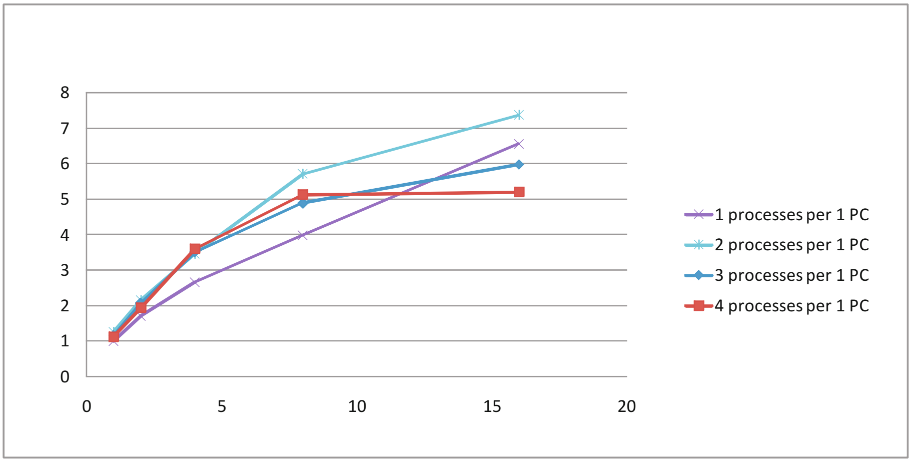

Fig. 4. Speed up obtained from parallel computations for a simulation scenario. The horizontal axis is the number of PCs. The vertical axis is the speed up.

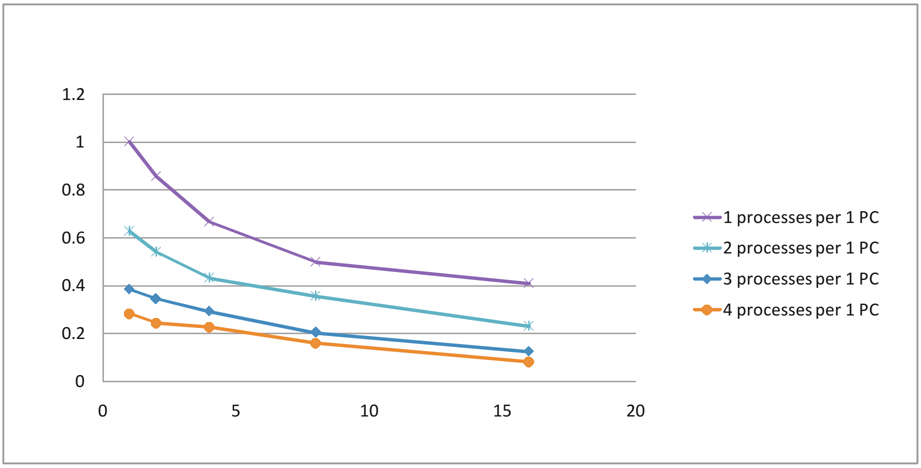

Fig. 5. Efficiency of parallel computations for a simulation scenario. The horizontal axis is the number of PCs. The vertical axis is the efficiency.

For the speed up we observe that more number of PCs leads to faster computation, which is reasonable. This is shown in Figure 4. However more number of PCs makes the computation less efficient, as shown in Figure 5. This can be understood as follows: physically the resources are wasted if the number of PCs is too many, and mathematically the divisor in equation (11) gets larger when P increases, which makes the efficiency low. From the speed and efficiency, we recommend that we use only one core in each i3 machine. Again, using all two cores in each i3 machine results in less efficient computation than our recommendation. We believe that this is due to some background tasks or other processes. One of the background tasks is the operating system turned on for each machine (note that one of the two cores in each i3 machine has run for the operating system already). Another task is the MPI process distributing the computations in parallel.

6 Conclusion

We have simulated several scenarios of parallel computations for flood flows. We have obtained a strategy for fast and efficient parallel computations using a cluster of workstations consisting of PCs with i3 processors. The strategy is to use only one core in each i3 machine in order to get fast and efficient computations. Note that employing all cores in each i3 machine leads to faster computations but less efficient. For future direction, we will investigate some strategies for fast and efficient parallel computations using other types of clusters.

Acknowledgements. The authors thank Professor Stephen Gwyn Roberts at Mathematical Sciences Institute, The Australian National University, Canberra, Australia for stimulating discussions. The first author also thanks The Australian National University and SMERU Research Institute for the Indonesia Project research grant.

References

1. Baldock, T.E., Barnes, M.P., Guard, P.A., Hie, T., Hanslow, D., Ranasinghe, R., Gray, D., Nielsen, O.: Modelling Tsunami Inundation on Coastlines with Characteristic Form. In: 16th Australasian Fluid Mechanics Conference, pp. 939–942. AFMS (2007)
2. Cousins, W.J., Power, W.L., Destegul, U.Z., King, A.B., Trevethick, R., Blong, R., Weir, B., Miliauskas, B.: Earthquake and Tsunami Losses from Major Earthquakes Affecting the Wellington Region. In: 2009 New Zealand Society for Earthquake Engineering Conference, Paper Number 24. NZSEE (2009)
3. Davies, G.: A Well-Balanced Discretization for a Shallow Water Inundation Model. In: 2011 International Congress on Modelling and Simulation (MODSIM), pp. 2824–2830. MSSANZ (2011)
4. Foster, I.: Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison-Wesley, Boston, Massachusetts (1995)
5. Guinot, V.: Wave Propagation in Fluids: Models and Numerical Techniques. ISTE, London - Wiley, Hoboken (2008)
6. Jakeman, J.D., Bartzis, N., Nielsen, O., Roberts, S.: Inundation Modelling of the December 2004 Indian Ocean Tsunami. In: 2007 International Congress on Modelling and Simulation (MODSIM), pp. 1667–1673. MSSANZ (2007)
7. Jakeman, J.D., Nielsen, O.M., Putten, K.V., Mleczko, R., Burbidge, D., Horspool, N.: Towards Spatially Distributed Quantitative Assessment of Tsunami Inundation Models. Ocean Dynamics 60, 1115–1138 (2010)

8. LeVeque, R.J.: Finite Volume Methods for Hyperbolic Problems. Cambridge University Press, Cambridge (2002)
9. Mache, J.: An Assessment of Gigabit Ethernet as Cluster Interconnect. In: 1st IEEE Computer Society International Workshop on Cluster Computing. IEEE (1999)
10. Mungkasi, S., Roberts, S.G.: A Finite Volume Method for Shallow Water Flows on Triangular Computational Grids. In: 2011 International Conference on Advanced Computer Science and Information System (ICACSIS), pp. 79–84. IEEE (2011)
11. Mungkasi, S.: A Study of Well-Balanced Finite Volume Methods and Refinement Indicators for the Shallow Water Equations. PhD Thesis, Australian National University, Canberra, Australia (2012)
12. Mungkasi, S., Roberts, S.G.: Predictions on Arrival Times of Water of the St. Francis Dam Break Flood Using ANUGA. In: 2013 International Congress on Modelling and Simulation (MODSIM), pp. 304–309. MSSANZ (2013)
13. Nielsen, O., Sexton, J., Gray, D., Bartzis, N.: Modelling Answers Tsunami Questions. AusGeo News 83, 1–5. Geoscience Australia (2006)
14. Roberts, S.G., Nielsen, O.M., Jakeman, J.: Simulation of Tsunami and Flash Floods. In: Modeling, Simulation and Optimization of Complex Processes, pp. 489–498. Springer, Berlin (2008)
15. Roberts, S., Nielsen, O., Gray, D., Sexton, J.: ANUGA User Manual. Geoscience Australia and Australian National University, Canberra (2010)
16. Sterling, T.: Beowulf Cluster Computing with Linux. MIT Press, Cambridge, Massachusetts (2001)
17. Tan, W.Y.: Shallow Water Hydrodynamics. Elsevier, Amsterdam (1992)
18. Yang, J., Shen, W., Yan, G.: Construct COW Based on MPICH in Linux Environment. In: 1st International Workshop on Education Technology and Computer Science. IEEE (2009)
19. AnugaParallel:
`https://anuga.anu.edu.au/wiki/AnugaParallel`
20. Message Passing Interface (MPI) Standard:
`http://www.mcs.anl.gov/research/projects/mpi`
21. MPICH High-Performance Portable MPI:
`http://www.mpich.org`