Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

Load-balanced parallel architectures for 2-D water quality model PARATUNA-WQ on OpenMP

Wai Kiat Tan [1, 2, *], Hock Lye Koh [1], Su Yean Teh [2]

[1]*School of Mathematical Sciences, Sunway University, Jalan Universiti, Bandar Sunway, 47500 Selangor, Malaysia*
[2]*School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Pulau Pinang, Malaysia*
[3]*Jeffrey Sachs Center on Sustainable Development (JSC), Sunway University, Jalan Universiti, 47500 Bandar Sunway, Selangor, Malaysia*

A R T I C L E I N F O

Article history:
Received 15 March 2017
Received in revised form
10 August 2017
Accepted 15 September 2017

Keywords:
PARATUNA-WQ
Water quality simulation model
Environmental sustainability
OpenMP
High performance computing

A B S T R A C T

Because of the potential speedup, parallel algorithms have recently been developed for improving serial applications in ocean and coastal hydrodynamics and water quality simulations. Developing a parallel program, however, is a difficult task that requires special and expensive processing resources. Motivated by the potential benefits of parallelization, this paper develops a load-balanced parallel architecture on OpenMP to improve on an in-house serial two-dimensional water quality simulation model to a parallel application named PARATUNA-WQ. Analysis of the performance of speedup is discussed to justify the use of parallel architecture in water quality simulation model. Speedup achieved by PARATUNA-WQ is close to the maximum theoretical speedup predicted by the Amdahl Law. Further enhancement for application to very large computational domain consisting of 25 million computational nodes is possible by integrating MPI architecture into the framework of OpenMP, the result of which will be reported in a subsequent paper.

1. Introduction

Because of the potential speedup, parallel algorithms have recently been developed for improving serial applications in ocean and coastal hydrodynamics and water quality simulations. Writing a parallel program is a difficult task that requires special and expensive processing resources. Motivated by the potential benefit of parallelization, this paper develops a load-balanced parallel application named PARATUNA-WQ on OpenMP to improve on an in-house serial two-dimensional water quality simulation model TUNA-WQ. Fast urban development along coastal regions worldwide to accommodate growing populations has led to increasingly severe pollution in the coastal water. An example would be the Mekong River and its Delta, flowing for some 4000 km from the head water in Tibet to the estuary in Ho Chi Minh City. Enhancement of water quality in the Mekong could benefit from fruitful research in robust mathematical simulation models. These models are useful for assessing the cost-effectiveness of competing treatment technology and control measures. Such mathematical model simulations lead to the spatial-temporal distributions of selected key water quality parameters to permit optimal choice of management options. We have earlier developed a serial water quality simulation model code-named TUNA-WQ to simulate the hydrodynamics and pollutant transport in offshore, coastal and estuarine systems based upon serial architectures (Chong et al., 2016). The main code TUNA is designed to simulate the two-dimensional vertically integrated hydrodynamic flows subject to tidal influence, initial and boundary conditions. Other forcing terms include an abrupt vertical uplift of ocean seabed that generates a tsunami (Koh et al., 2009; Teh et al., 2009). For realistic and accurate simulation of tsunami, it is essential to consider large computational domains with high spatial resolutions in the near shore coastal regions. This requirement gives rise to a scenario with a large number of computational nodes, frequently exceeding 10 million nodes, resulting in long computational time in excess of hours or even days. Hence, it is necessary to speed up the computational time in order to render the model robust. One approach is to utilize the power of

* Corresponding Author.
Email Address: waikiatt@sunway.edu.my (W. K. Tan)

https://doi.org/10.21833/ijaas.2017.012.019
2313-626X/© 2017 The Authors. Published by IASE.

parallel architecture involving simultaneous run of many processors. Similarly, the modelling of transport of pollutants with long half-life such as radioactive isotopes transported over a large oceanic space would require long computational time. Likewise, simulation of the aquatic ecosystems in the Mekong over the annual cycles would require several days of runtime. Parallel architecture may be used to speed up the computing time. In parallel computing, it is essential to provide an evenly distributed load balance of computational tasks among the participating processors (Dongarra et al., 2003; Liu and Liu, 2003). Good load balancing and effective traffic control are the keys to efficient parallelization (Wang et al., 2012). In the following section, we provide a brief overview of parallelization of four serial water quality simulation models for improving their performance.

2. Overview of parallelization

Firstly, TELEMAC is an integrated modelling suite for simulating offshore, coastal and estuarine systems, subject to free-surface flows, including flooding and drying processes. It has been undergoing continuous enhancement by the French EDF for the past 20 years. The TELEMAC suite includes (a) TELEMAC-2D for solving the depth-integrated shallow water equations to simulate the coastal environment when the horizontal length scale of the flow is greater than the vertical scale, and (b) TELEMAC-3D where the full Navier–Stokes equations are solved (Hervouet, 2007).

TELEMAC-2D has been used for simulating tidal currents off the coast of Brittany in France in the vicinity of a renewable energy marine turbine farm to estimate available energy. TELEMAC-3D has been used for assessing the effects of fresh water discharges on the salinity distribution in a coastal lagoon Berre Lagoon in southern France (Moulinec et al., 2011).

The authors implemented parallel architecture on the original serial TELEMAC codes to optimize performance made available by High Performance Computing (HPC) on several high-end platforms. Secondly, a parallel numerical simulation model CONDIFP was developed for the analysis of depth-averaged convection–diffusion problems on the Caltech Center for Advanced Computing Research Intel Touchstone Delta System, using up to 512 computational processors with an aggregate peak speed of 38.4 gigaflops and 19.5 gigabytes of memory. The original serial CONDIFP encountered the problem with uneven load distributions (Pirozzi, 1997).

A subsequent parallel improvement overcomes this uneven load allocation problem implemented on the Intel Paragon XP/S Model L38 Platform to illustrate the parallel versatility and reliability (Pirozzi and Zicarelli, 2000). Thirdly, a serial estuarine hydrodynamic and sediment transport model has been parallelized by Message Passing Interface (MPI) method based on domain decomposition techniques on IBM/SP2 machine using High Performance Fortran HPF FORTRAN 90 code and applied to study tidal currents and sediment transports in the Belgian coast (Yu et al., 1998). Fourthly, MPI method based on domain decomposition techniques has also been used to parallelize serial estuarine hydrodynamic and sediment transport model SW2D code. The parallel model PARSW2D is applied to analyze tidal currents and sediment transports in the estuary and offshore area of Zhejiang in China with good performance, reducing computational time by a factor of 30 to less than one day of computation time for simulation of annual sediment transport by tidal currents (Wenlong et al., 2014). Other achievements have been reported in the parallelization of two-dimensional shallow water depth-integrated hydrodynamic and sediment transport model (Wang and Zhang 2009; Yang and Cai, 2011).

3. Serial TUNA-WQ

TUNA-WQ is a serial two dimensional vertically-integrated water quality model that simulates tidal dynamics and pollutant transport to study the ecological and water quality scenarios in an aquatic environment. TUNA-WQ consists of two modules, namely hydrodynamics model TUNA and transport model WQ. The TUNA module solves the two-dimensional shallow water equations, as shown in Eqs. 1-3, using an explicit staggered finite difference method, while WQ module solves the transport equation, as shown in Eq. 4.

$$\frac{\partial \eta}{\partial t} + H\frac{\partial U}{\partial x} + H\frac{\partial V}{\partial y} = 0 \tag{1}$$

$$\frac{\partial U}{\partial t} + H\frac{\partial (U^2)}{\partial x} + H\frac{\partial (UV)}{\partial y} + g\frac{\partial \eta}{\partial x} + \frac{gn^2}{H^{4/3}}U\sqrt{U^2+V^2} = 0 \tag{2}$$

$$\frac{\partial V}{\partial t} + H\frac{\partial (V^2)}{\partial y} + H\frac{\partial (UV)}{\partial x} + g\frac{\partial \eta}{\partial y} + \frac{gn^2}{H^{4/3}}V\sqrt{U^2+V^2} = 0 \tag{3}$$

$$\frac{\partial S}{\partial t} = E_x\frac{\partial^2 S}{\partial x^2} + E_y - \frac{\partial^2 S}{\partial y^2} - U\frac{\partial S}{\partial x} - V\frac{\partial S}{\partial y} - \alpha S + W \tag{4}$$

Here, η (m) is water elevation above mean sea level, H (m) is mean water depth, U (m/s) and V (m/s) are velocity components in x- and y-directions, g (m/s^2) is gravitational acceleration, n (s/m$^{1/3}$) is Manning's friction coefficient, S (kg/m^3) is concentration of a substance, E_x (m^2/s) and E_y (m^2/s) are dispersions in x- and y- directions, α (s^{-1}) is decay rate of the substance and W (kg/s) is loading rate.

TUNA-WQ was originally developed with Fortran 77, but upgraded to Fortran 90 with its pseudocode displayed in Fig. 1. Upon execution, an input file is required to initialize constant parameters and loop controls with user defined values. This is followed by dynamic allocations of array for all dependent variables to setup the computational domain (i.e. number of computational nodes). The simulation will then begin, after computing the initial conditions, up to a user defined number of iterations N. In each iteration, η in Eq. 1 is first computed using the U and V from the previous iteration. After computing the boundary conditions, Eqs. 2 and 3 are solved by using the η computed from the current iteration. Eq. 4 is then solved for S using the U and V computed from the current iteration. At the end of every iteration, outputs such as concentration and velocity field can be stored at a user defined interval.

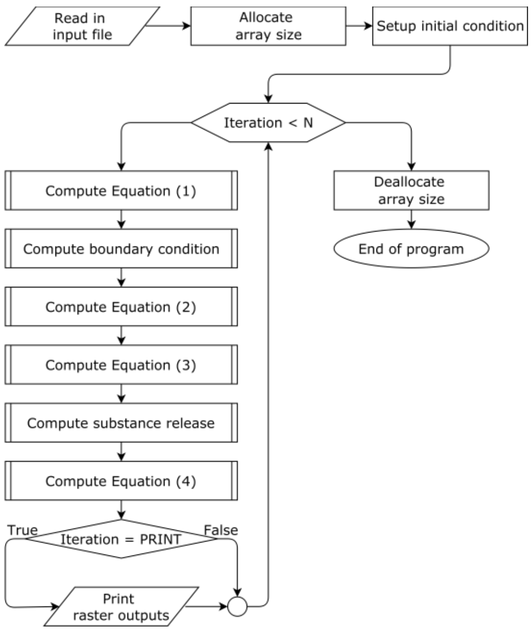

Fig. 1: Pseudocode of TUNA-WQ

4. Runtime analysis of serial TUNA-WQ

We first present the computational runtime analysis for the serial TUNA-WQ. For this purpose, a computational domain consisting of a channel with dimension of 20 km by 10 km in x- and y- directions respectively and a mean water depth of 10 m chosen. A tidal wave of amplitude of 1.0 m and wave period of 12.42 hours flowing in west-east direction in the open channel is simulated with a substance being released at the center of the domain. The simulation setup is summarized in Table 1.

Table 1: Computational domain and simulation setup

Nodes	Δx	Δt	Iteration N	$T = N \times \Delta t$
201×101	100 m	1 s	172800	48 hours

TUNA-WQ is compiled using Intel Visual Fortran (IVF) Compiler 16.0 and the simulation is performed in Windows 10 Pro with Intel Core i5-4460 Processor (6M Cache, 3.2 GHz). The total runtime for this simulation is 43.875 s. Three simulation snapshots of concentration and velocity field at time t = 38, 41 and 44 hours are displayed in Fig. 2.

In addition, an application of software performance analysis, namely Intel VTune Amplifier, is utilized to perform a profiling on TUNA-WQ simulation to investigate time spent in each computational subroutine. In this simulation, the computation of transport Eq. 4 is the most time-consuming component, taking up 48.02% of the total runtime. This is followed by the computations of Eqs. 2 and 3, taking up 20.63% and 19.48% of the total runtime, respectively, as displayed in Fig. 3. On the other hand, the computations of boundary conditions and substance release have the least impact among the computational subroutines. Note that, the sum of time spent in computational subroutine shown in Fig. 3 is 99.31%, whereas the remaining 0.69% of total runtime is taken up by other subroutines.

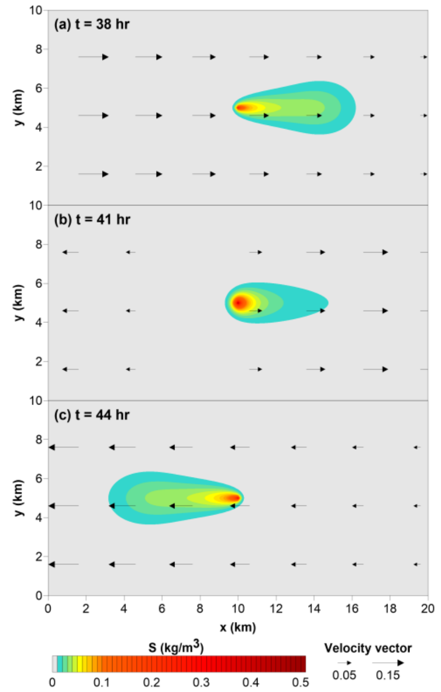

Fig. 2: TUNA-WQ simulation snapshots

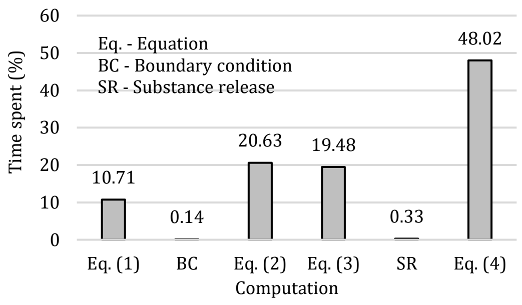

Fig. 3: Time spent in TUNA-WQ computational subroutines

5. OpenMP parallelization in PARATUNA-WQ

In this section, we parallelize the computational subroutines in TUNA-WQ by means of OpenMP (Open Multi-Processing) implementation. OpenMP is an implementation of multi-threading, a method of parallelization whereby a master thread (with thread ID 0) forks out a specified number of slave threads (with thread ID > 0) and divides a task allocation among them. The threads, including master and slave, then run concurrently within a parallel region. Upon exiting the parallel region, all slave threads will join back into the master thread, resulting in serial process to continue until the end of the program.

The profiling results for TUNA-WQ previously obtained via VTune Amplifier indicated that the computation subroutines within the loop of ITERATION < N takes up 99.31% of runtime. Therefore, we construct a parallel region to accommodate the loop using !$OMP PARALLEL directive with data attribute clauses *default*, *shared* and *private*, as displayed in Fig. 4. The clause *default* (*none*) is used to ensure that each variable in the parallel region is declared using either *shared* or *private* clause for careful implementation. In PARATUNA-WQ, all dynamic allocated array variables and constant parameters are declared as *shared*, allowing the data to be visible and accessible by all threads simultaneously. On the other hand, all dummy variables, loop counters and dummy loop counters are declared as *private*, allowing each thread to have a local copy and later use it in !$OMP DO SCHEDULE directive. Note that, a subroutine CPU_TIME() and OpenMP runtime subroutine OMP_GET_WTIME() are called before and after the execution of parallel region to record the elapsed CPUs time and elapsed wall clock time for investigating the performance of parallelized PARATUNA-WQ.

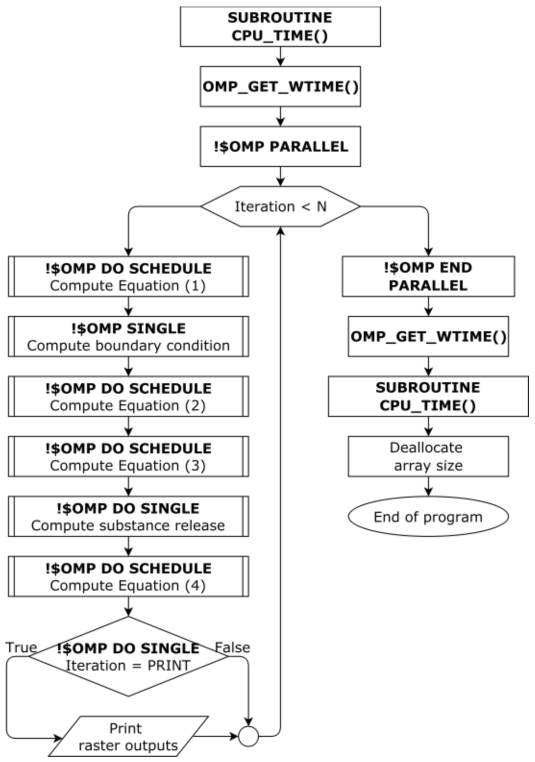
SUBROUTINE CPU_TIME()
OMP_GET_WTIME()
!$OMP PARALLEL
Iteration < N
!$OMP DO SCHEDULE Compute Equation (1)
!$OMP SINGLE Compute boundary condition
!$OMP DO SCHEDULE Compute Equation (2)
!$OMP DO SCHEDULE Compute Equation (3)
!$OMP DO SINGLE Compute substance release
!$OMP DO SCHEDULE Compute Equation (4)
True
!$OMP DO SINGLE Iteration = PRINT
False
Print raster outputs
!$OMP END PARALLEL
OMP_GET_WTIME()
SUBROUTINE CPU_TIME()
Deallocate array size
End of program

Fig. 4: Implementation of OpenMP directives in TUNA-WQ

To parallelize the computations within the loop of ITERATION < N, the !$OMP DO SCHEDULE directive is implemented on the computations of Eqs. 1-4. The directive !$OMP DO SCHEDULE is a work-sharing construct that 'chops' the number of iterations in *chunk* and later assigns the *chunk* to threads according to the defined scheduling clause. In this parallelization of PARATUNQ-WQ, the *dynamic* scheduling clause is implemented. In *dynamic* scheduling clause, the number of iterations is 'chopped' using a specified integer of chunk which is assigned to each thread. Once a particular thread finishes its chunk, it returns to get another chunk from the loop that is left. In this paper, *chunk* = 10 is used in dynamic scheduling clause. Furthermore, !$OMP SINGLE directive is implemented in the computations of boundary conditions and substance release, because these computational subroutines do not consume much simulation runtime.

The simulation described above is repeated for five times using both serial and parallel TUNA-WQ to compare the total runtime, as summarized in Table 2. The parallel implementation of OpenMP in PARATUNA-WQ reduces the runtime from 43.8587 s in TUNA-WQ to 18.6556 s in PARATUNA-WQ, resulting in a speedup factor of 2.35, with four CPUs. In addition, Intel VTune Amplifier is utilized to monitor the number of CPUs simultaneously utilized throughout the simulation. As observed in Fig. 5, the parallel process (more than 1 active CPUs) in TUNA-WQ takes up 12.49 s, while the serial process (1 active CPU) takes up 4.49 s, indicating 67% of the simulation runtime is taken up in parallel process.

Table 2: Comparison of total runtime between serial TUNA-WQ and parallel PARATUNA-WQ

Simulation	Total runtime (s)		Speedup (T_s/T_p)
	TUNA-WQ (T_s)	PARATUNA-WQ (T_p)	
1	43.8750	18.7292	2.34
2	43.9062	18.8110	2.33
3	43.5906	18.8011	2.32
4	44.0625	18.4854	2.38
5	43.8594	18.4511	2.38
Average	43.8587	18.6556	2.35

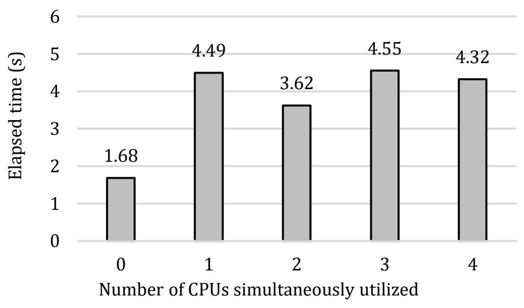

Fig. 5: Number of CPUs simultaneously utilized

6. Amdahl law

Amdahl's law states that the maximum speed up in parallelizing a serial algorithm is limited asymptotically by the serial fraction f of the code, as given in Eq. 5, in which m is the number of processors (Kathavate and Srinath, 2014). The serial fraction f of the code is the component that is not possible to parallelize. With increasing values of m, theoretical speedup reaches the limiting asymptotic value of $1/f$. For OpenMP PARATUNA-WQ, f is approximately 0.2, implying the limiting theoretical speedup of 5.0 when m becomes very large. In OpenMP implementation of PARATUNA-WQ, m is 4, giving theoretical speedup = 2.5, compared to the actual speedup of 2.35. A higher theoretical speedup of 4 can be obtained with 16 processors. Further increase in speedup is no longer significant beyond 16 processors. It should be noted that Amdahl's law assumes that the percentage of serial code f is independent of the problem size, which is not necessarily true, as overhead and synchronization tends to decrease with increasing computational nodes. Hence, the fraction f of time spent on executing serial code decreases with increase in problem size, for example increase in number of computational nodes (Padua, 2011). With very large number of computational nodes, the theoretical speedup is given by Eq. 6, known as the Gustafson Law. Fig. 6 display the theoretical speedup given by the Amdahl Law as a function of m, with three different values of f. Fig. 7 shows the comparison of speedup predicted by the Amdahl Law and Gustafson Law, indicating significant deviation between the two theoretical estimates.

$$speedup_{Amdahl} = \frac{1}{f+(1-f)/m} \quad (5)$$

$$speedup_{Gustafson} = m - f(m-1) \quad (6)$$

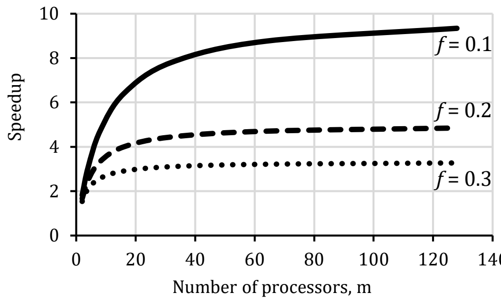

Fig. 6: Amdahl law

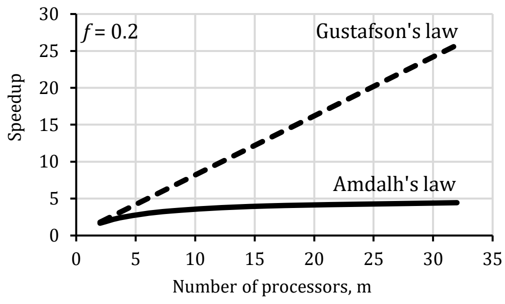

Fig. 7: Gustafson law versus Amdahl law for $f = 0.2$

7. Discussions

Running the MPI applications with several MPI processes but using only one single processor machine gives a good pre-indication about the speedup behavior of MPI applications, before running them on real powerful cluster machines or on an expensive parallel system. This approach was applied to reasonably predict the speedup of MPI applications that solve the wave equation and prime numbers generator problems using multiple physical processors (El-Nashar, 2011). In an upcoming paper, we will report our research findings on the performance of PARATUNA-RP that is used to simulate tsunami run-up heights and inundation distances along coastal beaches. PARATUNA-RP parallel architecture is based upon integrating MPI architecture into the framework of OpenMP. Parallelizing serial applications requires dedicated and expensive processing resources, prompting a debate between the overhead cost of parallelization and the benefit of speedup. The nature of the problem (for example the size of computational domain) is one of the most important factors that affect the parallel speedup. If the problem can be divided into independent subdomains with minimal communication-other than to split up the domain and to combine the final results-then this is a great parallelization opportunity, with the parallel applications exhibiting an almost linear speed up (Padua, 2011).

Parallel applications on the OpenMP programming platform has been shown to achieve significant speedup compared to the serial algorithm. For example, the parallel algorithm for the simple matrix multiplication on the OpenMP performs better than the sequential algorithm, with a maximum speedup of 1.96 achieved with only two processors, which is almost equal to the theoretical speedup of 2.00 with $m = 2$ and $f = 0.0$, by the Amdahl Law. However, the improvement in performance gained by the use of multi-processors depends very much on the algorithms used and their implementation. In particular, possible gains are limited by the fraction f of the algorithm that can be run in parallel simultaneously on multi-processors, an effect that is described by the Amdahl's law (Kathavate and Srinath, 2014). The parallelized application using OpenMP can further be fine-tuned and improved by reducing the value of f using the facilities provided by the Intel Vtune Amplifier tool. The parallel algorithms using OpenMP interface has been used for computing the solution of dense system of linear equations, for computing the value of Pi and for analyzing the speedup of parallel algorithms on multi-processor system. The numerical experiments show that the parallel algorithms achieve good performance in terms of speedup compared to the serial application (Sharma and Gupta, 2012).

8. Conclusion

This paper has presented a successful parallel implementation of a two dimensional vertically-integrated water quality application PARATUNA-WQ on OpenMP. It is noted that the value of f will decrease and the speedup will increase with increasing size of computational domain. Hence parallelization of serial tsunami model TUNA on large computational domain will achieve speedup closer to the theoretical speedup if many multi-processors are used. As noted earlier, an optimal number of processors should be around 16, in which case the speedup is close to the asymptotic upper bound given by $1/f$.

The next upcoming paper will address the implementation of parallel algorithm on existing serial TUNA, with the hope of achieving a speedup of at least 10. With a computational domain consisting of 25 million nodes, existing serial TUNA took 7 days runtime. The parallel PARATUNA, however, will take less than one day of runtime to simulate. We hope to implement PARATUNA-WQ for studying the Mekong aquatic ecosystems in the future.

Acknowledgment

TWK and TSY thank the International Centre for Theoretical Physics (ICTP) in Trieste, Italy for fellowships to participate in several high performance computing (HPC) Workshops, Schools and Symposiums. Financial support provided by Grant #203/PMATHS/6730101 is gratefully acknowledged. TWK and KHL thank Sunway University Business School for supporting this research publication.

References

Chong MSL, Teh SY, and Koh HL (2016). TUNA-WQ simulation of suspended sediments transport. In: Salleh S, Aris NA, Bahar A, Zainuddin ZM, Maan N, Lee MH, and Yusof YM (Eds.), American Institute of Physics (AIP) Conference Proceedings, AIP Publishing, 1750(1). http://dx.doi.org/10.1063/1.4954550

Dongarra J, Fox G, Kennedy K, White A, Foster I, Gropp W, and Torczon L (2003). Sourcebook of parallel computing. Morgan Kaufmann Publishers, Burlington, USA.

El-Nashar AI (2011). To parallelize or not to parallelize, speed up issue. International Journal of Distributed and Parallel Systems, 2(2): 14-28.

Hervouet JM (2007). Hydrodynamics of free surface flows: Modelling with the finite element method. John Wiley & Sons, Hoboken, USA.

Kathavate S and Srinath NK (2014). Efficiency of parallel algorithms on multi core systems using openmp. International Journal of Advanced Research in Computer and Communication Engineering, 3(10): 8237-8241.

Koh HL, Teh SY, Liu PLF, Ismail AIM, and Lee HL (2009). Simulation of Andaman 2004 Tsunami for assessing impact on Malaysia. Journal of Asian Earth Sciences, 36(1): 74-83.

Liu GR and Liu MB (2003). Smoothed particle hydrodynamics: A meshfree particle method. World Scientific, Singapore, Singapore.

Moulinec C, Denis C, Pham CT, Rouge D, Hervouet JM, Razafindrakoto E, Barber RW, Emerson DR, and Gu XJ (2011). TELEMAC: An efficient hydrodynamics suite for massively parallel architectures. Computers and Fluids, 51(1): 30-34.

Padua D (2011). Encyclopedia of parallel computing. Springer Science and Business Media, Berlin, Germany.

Pirozzi MA (1997). Numerical simulation of fluid dynamic problems on distributed memory parallel computers. Concurrency and Computation: Practice and Experience, 9(10): 989-998.

Pirozzi MA and Zicarelli M (2000). Environmental modeling on massively parallel computers. Environmental Modeling and Software, 15(5): 489-496.

Sharma SK and Gupta K (2012). Performance analysis of parallel algorithms on multi-core system using openmp. International Journal of Computer Sciences, Engineering and Information Technology, 2(5): 55-64.

Teh SY, Koh HL, Liu PLF, Izani AMI, and Lee HL (2009). Analytical and numerical simulation of tsunami mitigation by mangroves in Penang, Malaysia. Journal of Asian Earth Sciences, 36(1): 38-46.

Wang D, Long H, and Wang Z (2012). A load-balanced parallel algorithm of smooth particle hydrodynamics. In the 16th International Conference on Internet Computing for Science and Engineering (ICICSE'12), IEEE, Henan, China, 193-197. https://doi.org/10.1109/ICICSE.2012.62

Wang J and Zhang M (2009). Parallel computer technology study on 2D numerical model of flow and sediment in rivers. Journal of Waterway and Harbor, 30: 222-225.

Wenlong C, Yingbiao S, Xiuguang W, Zhiyong L, and Rongsheng W (2014). Parallel computer technology study on hydrodynamic and sediment transport mathematical model in estuaries based on MPI. In the 13th International Conference on Distributed Computing and Applications to Business, Engineering and Science (DCABES'14), IEEE, Xian Ning, China: 42-45. https://doi.org/10.1109/DCABES.2014.12

Yang C and Cai XC (2011). A parallel well-balanced finite volume method for shallow water equations with topography on the cubed-sphere. Journal of Computational and Applied Mathematics, 235(18): 5357-5366.

Yu CS, Berlamont J, Embrechts H, and Roose D (1998). Modeling coastal sediment transport on a parallel computer. Physics and Chemistry of the Earth, 23(5): 497-504.