Environmental Modelling & Software 73 (2015) 201–217

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Scalable parallel implementation for 3D semi-implicit hydrodynamic models of shallow waters

Mancia Anguita [a, d, *], Mario Acosta [a, b, d], F. Javier Fernández-Baldomero [a], Francisco J. Rueda [b, c]

[a] Dept. of Computer Architecture, University of Granada, Spain
[b] Water Research Institute, University of Granada, Spain
[c] Dept. of Civil Engineering, University of Granada, Spain
[d] Research Center for Information and Communications Technologies, University of Granada (CITIC), Spain

A R T I C L E I N F O

Article history:
Received 6 February 2015
Received in revised form
3 July 2015
Accepted 14 August 2015
Available online 28 August 2015

Keywords:
Three-dimension shallow water
Scalable parallel implementation
Semi-implicit integration
Pipeline processing
Domain-decomposition
OpenMP/MPI implementation
Nesting implementation

A B S T R A C T

This work presents a parallel implementation for 3D semi-implicit hydrodynamic models of shallow waters that scales in low-cost clusters of computers. The scalability of semi-implicit hydrodynamic models is limited due to the need of all-to-one/one-to-all communications at each simulation time-step as it is here shown. These communications are avoided taking advantage of a nesting implementation, which resolves, in addition to the model at the original grid resolution (nested), a model with a lower grid resolution (parent). Nesting implementations are normally used to simulate both global and local processes with less memory and execution time by using as nested domain just the area where local processes occur while the parent model simulates the complete domain; but here, it is used to improve scalability. A two-level processing structure is proposed for the parallel implementation: pipeline plus domain-decomposition. The resulting parallel implementation with two-level structure scales with a slope near one.

1. Introduction

Table 1 summarizes software packages with parallel implementations that are used to simulate three-dimensional ***shallow waters*** (i.e., in these models, a hydrostatic approximation is assumed for the vertical equation of motion). The difficulty to obtain a parallel implementation and its scalability (i.e. the time improvement as new computing resources are added) depend on the *time-discretization scheme* used for solving the 3D governing equations: explicit, (semi-)implicit, or splitting (column 3 in Table 1). *Explicit* schemes require higher computational time due to the limitation of the integration time-step to the time a surface (external) gravity wave takes to travel between two adjacent horizontal grid points: this limitation is referred to as Courant or CFL (Courant-Friedrichs Lewy) stability condition for gravity waves. In order to reduce computational time, allowing the use of higher time-steps while retaining free-surface effects, splitting and semi-implicit methods are preferred, but these methods are more difficult to parallelize. *Semi-implicit* approaches (Casulli and Cheng, 1992) avoid the time-step limitation due to CFL condition by treating implicitly the gravity-wave terms in the model equations, while other terms are treated explicitly, so that the time-step can be increased. Fully implicit implementations for three-dimensional shallow waters (3D-SW) equations are avoided due to the requirements in computational time and memory; implicit schemes require solving a coupled system of nonlinear equations for velocity and surface elevation over the entire domain each time-step. In shallow water modeling with a semi-implicit scheme (used for example in Si3D, Table 1), the solutions for surface elevation and velocity are uncoupled, a system of linear equations over the entire domain is solved at each time-step for surface elevation, and velocities are obtained explicitly using the computed surface elevations. The coefficient matrix for this system is symmetric and positive-definite so that the equations can be efficiently resolved using an iterative technique. The preferred iterative solver is PCG (Preconditioner Conjugate Gradient) due to its efficiency (see, for example, Golub and Van Loan, 2012; and Saad, 2003). For its part,

* Corresponding author. Dept. of Computer Architecture, University of Granada, Spain.
E-mail address: manguita@ugr.es (M. Anguita).

http://dx.doi.org/10.1016/j.envsoft.2015.08.014
1364-8152/© 2015 Elsevier Ltd. All rights reserved.

Table 1
Several software packages used to simulate 3D-SW and several parallel implementations proposed for them.

Soft.	References	Time integration	Nesting implementation	Parallel implementation	Parallel programming paradigm	Parallel structure
EFDC	EPA, 2002; Hamrick, 1992	Implicit splitting (PCG solver recommended)		O'Donncha et al., 2014	MPI	Domain-decomposition
MOM	Griffies et al., 2008	Explicit splitting	Herzfeld and Andrewartha, 2012	Griffies et al., 2008	FMS[a] (MPI)	Domain-decomposition
				Beare and Stevens, 1997	PVM[b]	Domain-decomposition
POM	Blumberg and Mellor, 1987	Explicit splitting	Giunta et al., 2007 (using the nesting of RSL[c] interface)	Jordi and Wang, 2012	MPI	Domain-decomposition
				Giunta et al., 2007	RSL[c] (MPI)	Domain-decomposition
POP	Smith et al., 2010; Dukowicz and Smith, 1994	Implicit splitting (PCG solver recommended)		Smith et al., 2010	Hybrid OpenMP-MPI	Domain-decomposition
ROMS	Shchepetkin and McWilliams, 2005	Explicit splitting	Debreu et al., 2012; Penven et al., 2006 (using multi-grid of AGRIF[d] interface)	Wang et al., 2005	MPI	Domain-decomposition
Si3D	Smith, 2006	Semi-implicit (PCG solver recommended)	Acosta et al., 2015	Acosta et al., 2010	Hybrid OpenMP-MPI	Domain-decomposition

[a] FMS (Flexible Modelling System, Balaji, 2002). It provides an interface to MPI and to SHMEM (library of Cray).
[b] PVM (Parallel Virtual Machine). Popular tool in the 90s for message-passing programming based on a library of functions. The experience in PVM helped to develop MPI, current de-facto standard based on a library of functions for message-passing programming.
[c] RSL (Runtime System Library, Michalakes, 2000). It provides an interface able to define levels of grids and to parallelize the grid levels (domains) over the same set of processors, where each one has a piece of every domain. It uses MPI.
[d] AGRIF (Adaptive Grid Refinement in Fortran, Debreu et al., 2008). It provides an interface to define levels of grids.

splitting methods (Blumberg and Mellor, 1987) separate the 3D governing equations into the so called external or barotropic mode, a 2D model for the depth-averaged flow (associated with the fast moving waves), and the internal or baroclinic mode, a 3D model for the vertical structure of flow (slower moving waves). The coupling of these internal and external modes is required. This splitting allows different time-steps for the 2D and 3D models, enabling the use of explicit integration with a short time-step that satisfies the CFL condition for the fast-moving surface waves and with a longer time-step for the 3D model. However, the problem of coupling the external and internal modes with different time-steps comes up in this case. Several variations of splitting methods are used. Usually, the *internal mode* uses an explicit scheme except for the vertical diffusion terms, which are usually treated implicitly for stability reasons. *External mode* can be either explicit (explicit splitting methods), or it can be implicit or semi-implicit (implicit splitting methods). *Explicit splitting methods* (used for example in ROMS, MOM4.0 and later MOM releases, and POM in Table 1) avoid the need to solve a system of equations over the entire domain at each external mode time-step, simplifying their numerical and parallel implementation, while *implicit splitting methods* (e.g. EFDC, POP in Table 1) make external and internal modes coupling easier allowing the same time-step for both modes. Discussions about the accuracy of (semi-)implicit schemes and (explicit and implicit) splitting methods can be found for example in Smith, 2006, and Dukowicz and Smith, 1994. Solving an equation system to obtain surface elevation over the entire domain each time-step makes hydrodynamic models with (semi-)implicit (splitting and non-splitting) schemes more difficult to parallelize than explicit splitting and fully explicit schemes. Note that the smaller computational load (due to the larger time-step) also impairs the scalability of these schemes compared to explicit schemes, i.e. the task of parallelizing explicit schemes is more rewarding for the programmer.

Parallel implementations of hydrodynamic models require *interchange* collective communications each time step, no matter what time integration method is used. (Semi-)implicit approaches (splitting or non-splitting) additionally require *all-to-one/one-to-all* collective communications. To be more precise, if a parallel PCG solver is used to obtain surface elevation, all-to-all reduction communications (i.e. all-to-one reduction plus one-to-all broadcast communications) are required at each solver iteration (see for example Nesterov, 2010) and, if a sequential PCG solver within the parallel code is used, a couple of all-to-one gather and one-to-all scatter communications are required (see for example Acosta et al., 2010; O'Donncha et al., 2014). A parallel PCG adds both interchange and all-to-one/one-to-all collective communications at each solver iteration independently of the preconditioner used (the CG algorithm and customary preconditioners can be seen, for example, in Golub and Van Loan, 2012; and Saad, 2003), and they cannot be eliminated. A reduction in the number of individual all-to-all communications has been obtained in D'Azevedo et al., 1999 by a rearrangement of the CG computation that allows combining the two all-to-all original reductions into a single all-to-all communication with two reductions. As it is here shown for low-cost clusters, the all-to-one/one-to-all collective communications limit the scalability of (semi-)implicit (splitting and non-splitting) schemes. The lack of scalability is a disadvantage of these schemes (also stated in Griffies et al., 2000 and Weller et al., 2013). The implementation here proposed for parallel hydrodynamic models allows to avoid the all-to-one/one-to-all communications that limit scalability without relinquishing a semi-implicit implementation (splitting or non-splitting) or a PCG solver. The approach proposed to obtain a scalable implementation uses an ***online nesting implementation*** to eliminate the all-to-one/one-to-all communications (Section 2.2 clarifies the difference between *online* and *offline* implementations).

Nesting implementations are used in hydrodynamic models with structured grids to allow simulating both base-scale (global) processes and regional (local) processes reducing both the memory and run-time requirements because they avoid the simulation of the entire basin in the high resolution required to simulate local processes (Fig. 1). Online nesting implementations (Table 1, 4th column) have been proposed for instance for MOM (Herzfeld and Andrewartha, 2012), POM (Giunta et al., 2007), ROMS (Debreu