Preprocess static subdomain decomposition in practical cases of 2D unsteady hydraulic simulation

A. Lacasta*, P. García-Navarro*, J. Burguete**, J. Murillo*,

Abstract

Explicit finite volume methods are frequently used and widely accepted in hydraulic models based on the shallow water approximation. The main drawback of the approach is the time step size limit imposed by the Courant-Friedrichs-Lewy numerical stability constraint. This leads to excessively long computational times in large scale cases of practical interest. At the same time, the accuracy of the numerical results is associated to the use of fine computational meshes able to achieve enough spatial resolution. Taking into account that hydraulic modellers do not have access, in general, to large computational facilities, suitable and useful parallelization techniques are required. Furthermore, if High Performance Computing facilities are used, it is usually necessary to provide an estimation of the requirements of computational load to cover the length of the simulation. In this work the suitability of a preprocess static subdomain decomposition is explored and presented as a promising strategy to improve the efficiency of 2D unsteady shallow water computational models over dry bed in medium scale computational facilities and, at the same time, is useful to provide a preprocess computational time estimation if large scale computational facilities are going to be used.

Keywords: Parallelization, Computational hydraulics, Unsteady flow, Static Domain Decomposition, Dry/wet boundaries

*Fluid Mechanics, Universidad Zaragoza
**Soil and Water, EEAD, CSIC
Email addresses: alacasta@unizar.es (A. Lacasta), pigar@unizar.es (P. García-Navarro), burguete@eead.csic.es (J. Burguete), Javier.Murillo@unizar.es (J. Murillo)

1. Introduction

Parallelization strategies offer the possibility to improve the compromise between stability and efficiency in numerical methods with restrictions in the time step size. Also, in general, explicit methods used to simulate unsteady inundation flows do not require to perform calculations in all the cells but only in those covered by water. As the simulation progresses, the number of involved computational cells changes. When using a well balanced pre-process subdomain decomposition, the initial optimal partition becomes bad balanced as the wave propagation advances. On the other hand, as the number of partitions grows, the relative weight of this unbalance decreases. Some authors have proposed different ways to perform this kind of implementation on structured meshes [12] whereas others [14] have performed the partition on unstructured meshes using METIS to stablish the partitions according to the number of wet/dry cells. In the present work the impact of the number of cells based the partition strategy is analyzed.

2. Mathematical model/Governing equations

The two-dimensional shallow water equations, which represent mass and momentum conservation in a plane, can be obtained by depth-averaging the Navier-Stokes equations. Neglecting diffusion of momentum due to viscosity and turbulence, wind effects and the Coriolis term, they form a system of equations:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{U})}{\partial y} = \mathbf{S}(\mathbf{U}, \mathbf{x}, \mathbf{y}) \tag{1}$$

where

$$\mathbf{U} = (h,\ q_x,\ q_y)^T \tag{2}$$

are the conserved variables with h representing the water depth, q_x and q_y, with (u, v) the depth averaged components of the velocity vector $\mathbf{u}$ along the (x, y) coordinates respectively. The fluxes of these variables are given by:

$$\mathbf{F} = \left(q_x,\ \frac{q_y^2}{h} + \frac{1}{2}gh^2,\ \frac{q_x q_y}{h}\right)^T, \qquad \mathbf{G} = \left(q_y,\ \frac{q_x q_y}{h},\ \frac{q_y^2}{h} + \frac{1}{2}gh^2\right)^T \tag{3}$$

where g is the acceleration of the gravity. The source terms of the system are split in two kind of terms. The bed slope and friction source terms of the momentum equations:

$$\mathbf{S} = (0,\ gh(S_{ox} - S_{fx}),\ gh(S_{oy} - S_{fy}))^T \tag{4}$$

where the bed slopes of the bottom level z are

$$S_{ox} = -\frac{\partial z}{\partial x}, \qquad S_{oy} = -\frac{\partial z}{\partial y} \tag{5}$$

and the friction losses are written in terms of the Manning's roughness coefficient n:

$$S_{fx} = \frac{n^2 u\sqrt{u^2+v^2}}{h^{4/3}}, \qquad S_{fy} = \frac{n^2 v\sqrt{u^2+v^2}}{h^{4/3}} \tag{6}$$

System (1) is time dependent, non linear, and contains source terms. Under the hypothesis of dominant advection it can be classified and numerically dealt with as belonging to the family of hyperbolic systems. The mathematical properties of (1) include the existence of a Jacobian matrix, $\mathbf{J_n}$, of the flux normal to a direction given by the unit vector $\mathbf{n}$, $\mathbf{En}$, with $\mathbf{E} = \mathbf{F}n_x + \mathbf{G}n_y$, defined as

$$\mathbf{J_n} = \frac{\partial \mathbf{En}}{\partial \mathbf{U}} = \frac{\partial \mathbf{F}}{\partial \mathbf{U}} n_x + \frac{\partial \mathbf{G}}{\partial \mathbf{U}} n_y \tag{7}$$

This Jacobian can be used to form the basis of the upwind numerical discretization that will be outlined in next section.

3. Finite Volume Model

To introduce the finite volume scheme used in this work, (1) is integrated in a volume or grid cell Ω:

$$\frac{\partial}{\partial t}\int_\Omega \mathbf{U} d\Omega + \int_\Omega (\overrightarrow{\nabla}\mathbf{E}) d\Omega = \int_\Omega \mathbf{S} d\Omega \tag{8}$$

It is assumed that the third integral can be reformulated as

$$\int_\Omega \mathbf{S} d\Omega = \oint_{\partial\Omega} (\mathbf{Tn}) dl \tag{9}$$

where $\mathbf{T}$ is a suitable numerical source matrix. This enables the following formulation

$$\frac{\partial}{\partial t}\int_\Omega \mathbf{U} d\Omega + \oint_{\partial\Omega} \mathbf{En} dl = \oint_{\partial\Omega} \mathbf{Tn} dl \tag{10}$$

When the domain is sub-divided in cells Ω_i (see Figure 1), using a mesh fixed in time, (10) can also be applied to each cell. Assuming a first order in space approach equation (10) reduces to [11]

$$\frac{(\mathbf{U}_i^{n+1} - \mathbf{U}_i^n)}{\Delta t} A_i + \sum_{k=1}^{NE} (\delta\mathbf{E} - \mathbf{T})_k \mathbf{n}_k l_k = 0 \tag{11}$$

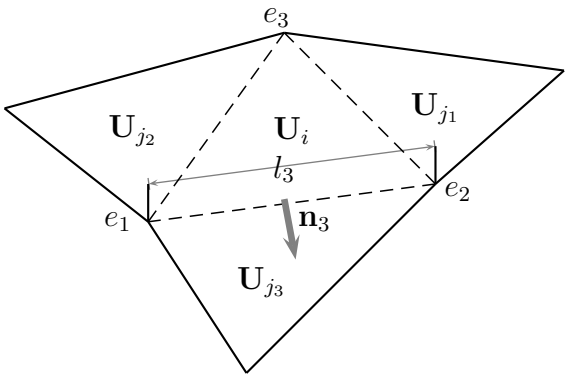


Figure 1: Cell parameters.

where A_i is the cell area, $\delta\mathbf{E} = \mathbf{E}_j - \mathbf{E}_i$, with $\mathbf{E}_i$ and $\mathbf{E}_j$ the values of the function $\mathbf{E}$ at the neighbour cells i and j respectively, $\mathbf{n}_k = (n_x, n_y)$, the outward unit normal vector to the cell edge k, l_k is the corresponding edge length, NE is the number of edges in the cell.

The exact value of the flux and sources at each cell edge $(\delta\mathbf{E} - \mathbf{T})_k\mathbf{n}_k$ in (11) is unknown, and it is approximated by means of a Godunov type approximate solver [3], that depends on the values of the variables at both sides of the cell edge. One option is the Roe's approximate solution [13]. This approximation is based on the local linearization of the flux $\mathbf{E}$, by means of an approximated Jacobian matrix $\widetilde{\mathbf{J}}_{\mathbf{n},k}$. This approximated Jacobian matrix provides a set of three real eigenvalues $\widetilde{\lambda}_k^m$ and eigenvectors $\widetilde{\mathbf{e}}_k^m$.

The problem is reduced to a 1D Riemann problem projected onto the direction $\mathbf{n}$ at each cell edge. The hyperbolic nature of the equations allows to express the flux and source term as a linear combination with the following form [10]

$$(\delta\mathbf{E} - \mathbf{T})_k\mathbf{n}_k = \sum_{m=1}^{3}(\widetilde{\lambda}\alpha\theta\widetilde{\mathbf{e}})_k^m \tag{12}$$

The $\widetilde{\lambda}$ eigenvalues express the direction of propagation of the information. They are usefull to provide a correct updating of the conserved variables. The explicit first order upwind scheme for system (11) gets the form:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \Delta t \sum_{k=1}^{NE} \mathbf{\Psi}_{i,k}^n \tag{13}$$

and is expressed as the sum of each edge contribution, $\mathbf{\Psi}_{i,k}$

$$\mathbf{\Psi}_{i,k} = \sum_{m=1}^{3} (\widetilde{\lambda}^{-} \alpha \theta \widetilde{\mathbf{e}})_k^m l_k / A_i \tag{14}$$

with $\widetilde{\lambda}^{-} = \frac{1}{2}(\widetilde{\lambda} - |\widetilde{\lambda}|)$. It is worth mentioning that special care must be considered when defining the linearization coeficients in (14) to avoid unphsical results, as negative values of water depth, when updating the conserved variables [10].

The scheme as in (13) is explicit and, therefore, conditionally stable. The dynamic time step in the stability region must be computed following [10]

$$\Delta t = \min \left\{ \Delta t_k \right\}_{k=1,Nedge}, \qquad \Delta t_k = \left\{ \frac{A_{\min,k}}{max \left[|\widetilde{\lambda}_k^m| \right] l_k} \right\} \tag{15}$$

with $A_{\min,k} = \min(A_i, A_j)$ and $Nedge$ the number of cell edges in the domain.

4. Machine Description

The machine where the computation has been made is called *Caesaraugusta* and it is a member of *Spanish Supercomputing Network*. Its main capabilities are the distributed memory paralellism paradigm using MPI standard and is briefly described in Table 1.

Technical Characteristics	
Processor	512 processors PowerPC 970FX 2.2 GHz
Performance	2994.04 GFlops
Memory	1 TB RAM Memory
Network	Interconnection networks Myrinet
O.S.	Linux
Total	256 nodes

Table 1: Caesaraugusta specifications.

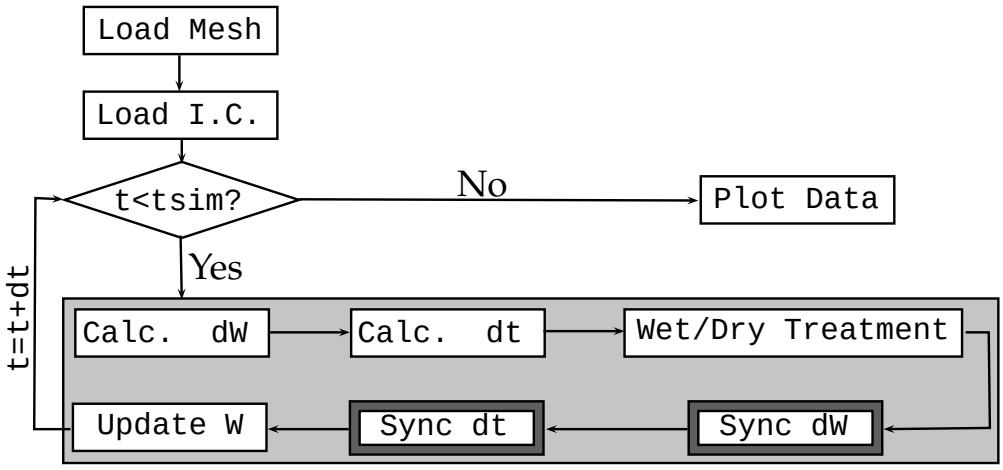


Figure 2: Algorithm scheme with new operations highlighted

Algorithm 1 Syncronize ΔU (Send and Receive is applied to *Delta*U of neighbouring cells)

Require: Domain has been decomposed in $p \geq 2$

1: **if** $mpi_{rank} = 1$ **then**
2: receive from 2 and send to 2
3: **else**
4: **if** $mpi_{rank} = p$ **then**
5: send to $p-1$ and receive from $p-1$
6: **else**
7: **if** $mpi_{rank} MOD(2) = 0$ **then**
8: send to $mpi_{rank}-1$, receive from $mpi_{rank}-1$,receive from $mpi_{rank}+1$ send to $mpi_{rank}+1$
9: **else**
10: receive from $mpi_{rank}+1$, send to $mpi_{rank}+1$, send to $mpi_{rank}-1$ and receive from $mpi_{rank}-1$
11: **end if**
12: **end if**
13: **end if**

5. Paralellization Strategy

Parallelization on distributed memory machines requires a special way to

Algorithm 2 Syncronize Δt

Require: Domain has been decomposed in $p \geq 2$

if $mpi_{rank} = 1$ **then**
  $DT(0) = DT_0$
  for l=1..p-1 **do**
    Receive $DT(mpi_{rank})$ from node l
  end for
  To find $\min \delta t$ as $(dt = MIN(DT(0..p-1))$
  for i=1..p-1 **do**
    Send dt to node i
  end for
else
  Find $\min \delta t$ of my domain
  Send dt_{rank}
  Receive dt_{min}
end if

decompose the domain. In 2D problems such as the one solved in the present work, it is possible to choose between 1D domain decomposition (along a preferential direction) or 2D domain decomposition (no preferential direction). In either of them, the schematic representation of the procedure to modify the variable updating given by (13) and the dynamic time step calculation given by (15) in every subdomain is displayed in Figure 2. The modifications introduced by the parallelization are highlighted in the flowchart showing the necessity to provide global information to all the subdomains in every time step. The particular form to perform this will be detailed later.

On the other hand, in unsteady problems like the flooding of dry regions, it is possible to use constant in time 1D or 2D decompositions (static) or the dynamic version of both of them that change the domain distribution along the simulation as the flooding progresses and the number of computational cells involved in the calculation grows. If the dynamic decomposition is chosen, the computational time required to balance the load among subdomains becomes critical as the effective computational domain changes along the simulation. The static decomposition, on the other hand, does not require an estimation of the computational load each time step as this option does not offer an optimal distribution. In the present work, the suitability of the static decomposition strategy for unsteady problems is explored.

When using a 1D static decomposition, the existence of a preferential flow direction is implicit and the the good balance of number of cells among partitions is required. In that case, the communication scheme is as it appears in Figure 4 and is detailed in Algorithm 1. On the left hand side of Figure 4, the scheme used to exchange information about the variable updating is said to follow a *red/black update* where the operation is first applied to "red" elements and then to "black". To organize the communication, the algorithm is split in 4 sequential and ordered steps. Independently of the number of partitions in the domain, in two steps the "red" elements are sending to the "black" elements. In the other two steps, the "red" elements receive and the "black" send. The right hand side of Figure 4 shows the *reduction* pattern used to share the information concerning the time step size and is detailed in Algorithm 2. First, each subdomain sends its time step size to one of them (previously stated) where the global minimum size is calculated; then, this value is resent to all of the subdomains in order to share the global minimum time step. MPI functions Senc/Recv are designed oriented to block the execution until the function has not ended [8]. In order to synchronize all the tasks, Algorithm 1 and Algorithm 2 make this function calls ensuring that, every time-step, the functions have sent/received the neccessary data to continue with the calculation.

The most important advantage of this strategy is the possibility of stablishing an approximation of the duration of the simulation and of the computational requirements according to the time cost desired. In other words, when we have enough information about the decomposition in terms of number of cells per subdomain and the largest number of cell edges (walls) shared between all subdomains the simulation cost can be bounded as follows:

$$t_X \leq n_{steps}((t_x \max(n_{cell,i})) + (t_c\alpha \max(n_{walls,i,j}))) \tag{16}$$

where t_x is the calculation time per cell and $t_c\alpha$ is the time of communication per wall.

Values of t_x and $t_c\alpha$ are obtained through regression analysis. A preliminar test case with 4 partitions with a growing number of cells involved was run for that purpose. The computaional cost associated to one time step was measured as a growing function of the number of cells involved. This is plotted on the left side of Figure 3.

Taking into account that we are looking for the fitting of

$$t_X \leq n_{steps}((t_x \max(n_{cell,i})) + (t_c\alpha \max(n_{walls,i,j}))) \tag{17}$$

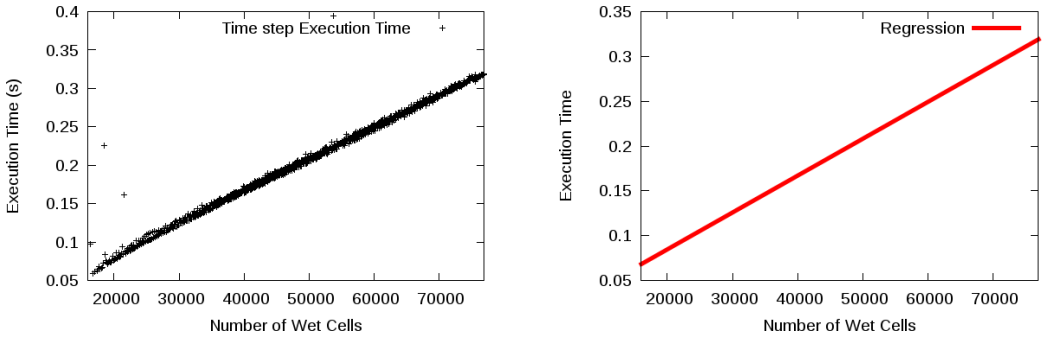


Figure 3: Left: Measured execution time required for one time step as a function of the number of wet cells in a case with 4 subdomains. Right: Linear regression.

it can be rewritten as a linear function,

$$t_X \leq n_{steps}((4.11272E-06 \max(n_{cell,i})) + (0.00242021)) \tag{18}$$

where the second factor is a constant term independent from the number of cells involved in each time step and adjustable to each simulation. In a test case with 40 shared walls, a value of $t_c\alpha = 5.9028E-5$ was obtained.

Quantifying $t_x = 4.11272 \cdot 10^{-6}$ and $t_c\alpha = 5.9028 \cdot 10^{-5}$ for our machine:

$$t_X \leq n_{steps}((4.11272 \cdot 10^{-6} \max(n_{cell,i})) + (5.9028 \cdot 10^{-5} \max(n_{walls,i,j}))) \tag{19}$$

This value is actually an upper bound since all the cells in the subdomains are not always wet ($h > 0$).

6. Results

6.1. Test case 1: Dam break over flat and frictionless bed

The first test case is concerned with the simulation of a dam break flow (Figure 7) over flat and frictionless bed in a rectangular domain divided by a solid wall except in a narrow central region where a gate is assumed. A discontinuity in the water surface between 5m and 0m together with zero velocity are assumed as initial conditions. After the gate removal, the flow develops in a simultaneous shock wave advancing over the lower depth reservoir and a rarefaction wave smoothly progressing into the higher level side.

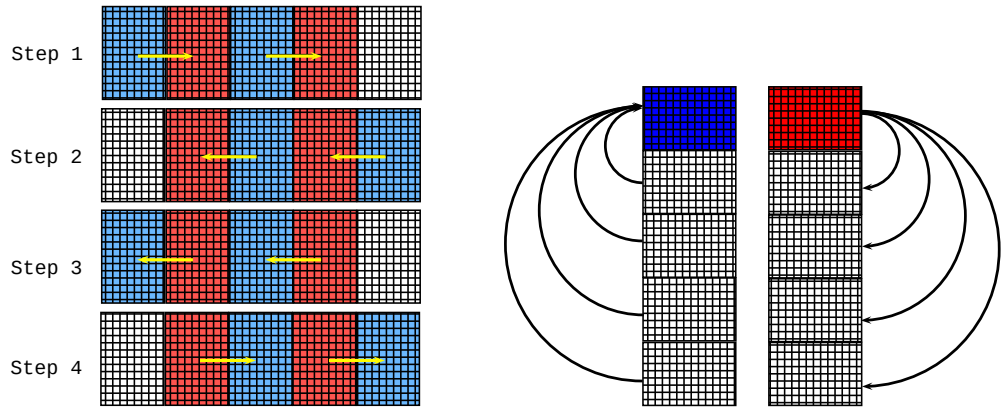


Figure 4: Left: Communication scheme for the conservative variables U. Right: Comunication scheme for Δt.

For the simulation all the boundaries were assumed as solid walls and 300,000 grid cells were used. The grid is shown on the left part of Figure (5) together with information concerning the size of the cells. The right part of Figure (5) displays an example of the partition in 8 subdomains. These are parallel to the gate due to the symmetry of the test case. The amount of grid cells used allowed for a large number of partitions and, in particular, up to 128 nodes were used in the calculation.

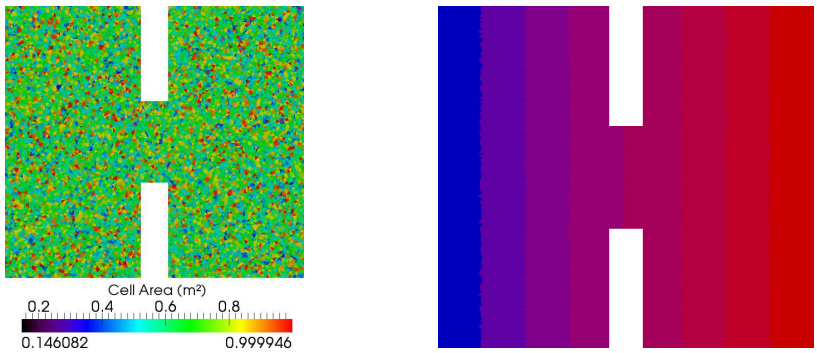
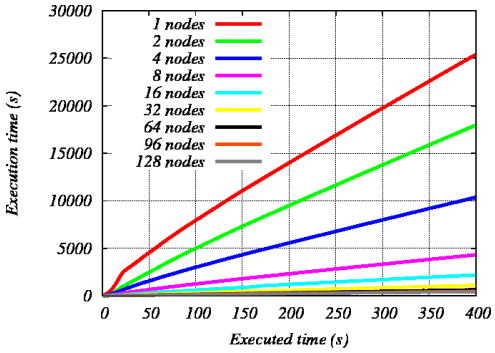
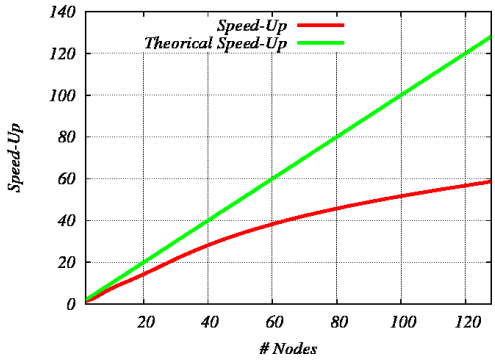


Figure 5: Test case 1. left: Geometry of the problem. Right: Partition in 8 subdomains

Figure 6(a) and Table 2 shows that the computational time decreases as



(a) Computational time versus real time



(b) Speed-up in 400 s.

Figure 6: Computational time (a), and speed-ud (b) in Test case 1 using *Caesaraugusta*

the number of computers increases. However, Figure 6(b) displays that the gain is not linear; this is due to the constant time spent in communication.

These cases are worst cases because every domain is wet all the time and hence is the effective calculation domain. On most cases, this situation is variable as unsteady problems make variable the domain. This test-case shows the better performance it can be reached as this case does not treat the wet/dry problem .

Table 2: Performance measurement for test case 1

Performance Measurement Techniques	Number of Processors						
	2	4	8	16	32	64	128
Speed-Up	1.41	2.45	5.87	11.54	23.02	29.93	58.65
Efficiency	0.70	0.61	0.73	0.72	0.71	0.62	0.45

6.2. Test case 2: Inundation flow in a river

As a second test case of more complexity and interest in practical applications of Hydraulic Engineering, a realistic inundation event in a reach of the Ebro river (Spain) has been used. In this case, the simulation has been run for 38,000 s. The topography is represented on Figure (8). The left part of

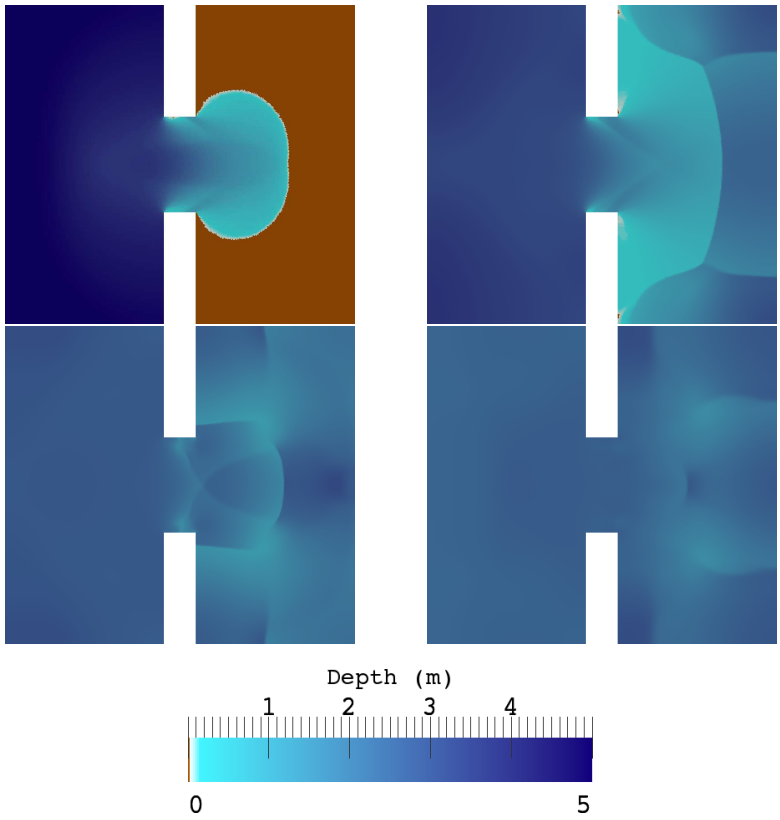


Figure 7: Evolution of water depth in t=7.5s [Top Left], t=25s [Top Right], t=32.5s [Bottom Left], t=50s [Bottom Right] in Test case 2

Figure (9) shows the grid of 41,000 cells and an example of its decomposition in 8 subdomains. The mesh cell size distribution is detailed on the right part of Figure (9). This case was decomposed up to 24 subdomains and a 1D criterion analogous to the one chosen in the previous example was followed due to the existence of a preferential direction in the flow. The inlet hydrograph is a linear curve where $Q(t = 0) = 0m^3/s, Q(t = 35,000) = 2,000m^3/s$.
The numerical results of the simulation are plotted in Figure (10) in the form of six snapshots of the water depth contour map at six different times.

In this case, the gain achieved by means of the static subdomain decomposition presented is better than the theorical one. This is due to the increasing number of cells involved in the calculation as time progresses. The graph of the computational time versus real time has been plotted in Figure (11) (a)

and the speed-up achieved at time $t = 35,000s$ is shown both in Figure (11) (b) and Table 3. It is worth noting that the algorithm used to include new cells requires high interaction with the main memory, and this access misses sometimes in cache memory [6]. Each miss implies time spent accessing to main memory. The paralellization in subdomains somehow implies also parelellizing these accesses therefore decreasing the number of misses in cache memory.

This case is closer to realistic simulations, where the flooding area is increasing (or decreasing). In order to make the minimum number of calculations, to reduce the number of cells wich it is going to make calculations in, is desired.

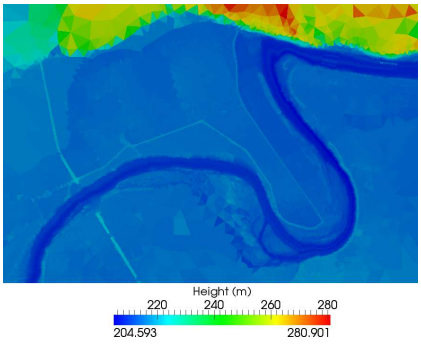


Figure 8: Test case 2. Topography of the problem.

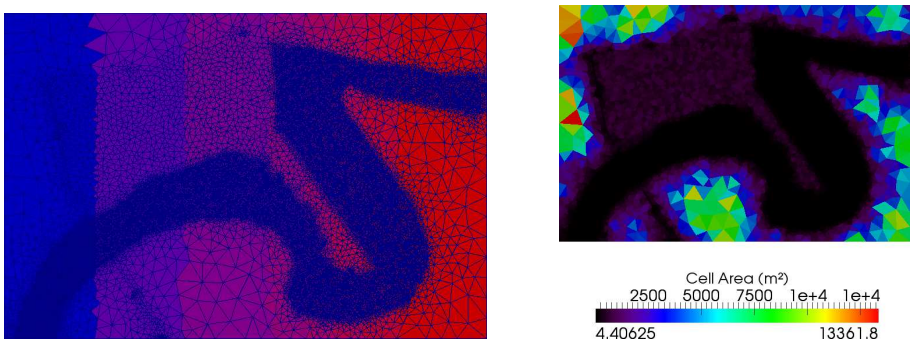


Figure 9: Test case 2. Left: Computational grid used and example of partition in 8 subdomains. Right: Distribution of cell sizes.

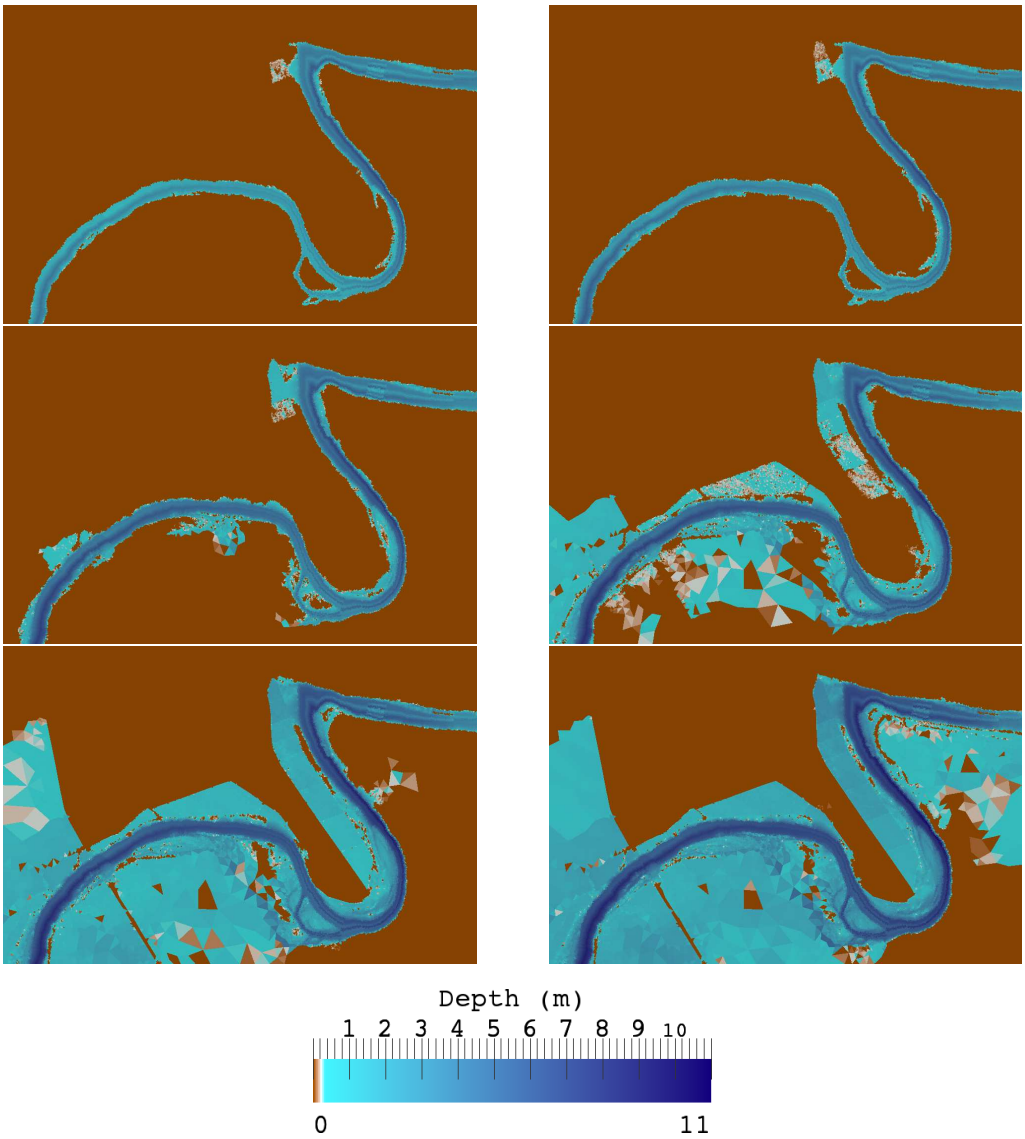
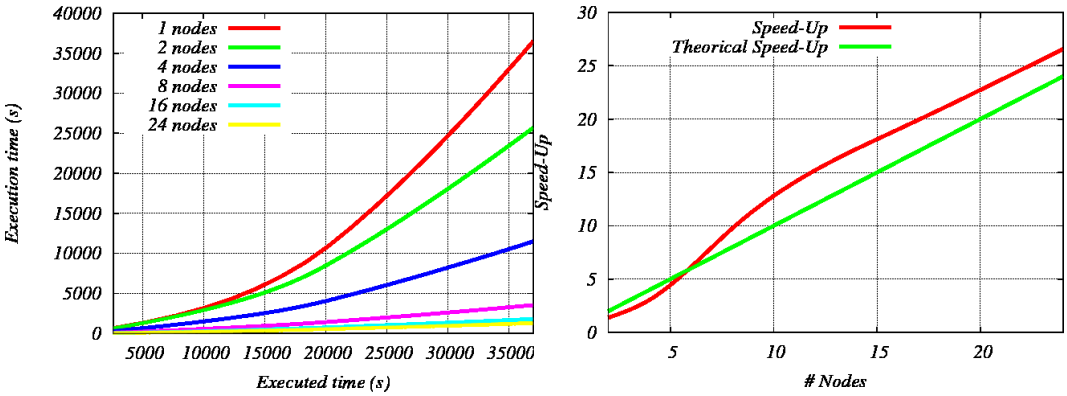


Figure 10: Evolution of water depth in t=0s [Top Left], t=7,000s [Top Right], t=14,000s [Middle Left], t=21,000s. [Middle Right], t=28,000s. [Bottom Left] and t=35,000s [Bottom Right] in Test case 2 with input hidrograph Q(t)=17.5t m^3/s

7. Conclusions

In this work, the suitability of a static subdomain decomposition strategy for realistic unsteady 2D shallow water flows with and without wet/dry fronts



(a) Computational time versus real time

(b) Speed-up in 36,500 s.

Figure 11: Computational time (a), and speed-ud (b) in Test case 2 using *Caesaraugusta*

Table 3: Performance measurement for test case 2

Performance Measurement Techniques	Number of Processors				
	2	4	8	16	24
Speed-Up	1.38	3.06	9.76	19.01	26.55
Efficiency	0.69	0.76	1.22	1.18	1.10

has been analyzed. The parallelization technique has been coupled to a previously developed explicit upwind finite volume model already tested. The decomposition of the domain into subdomains has been made part of the preprocess of the simulation so that the user can choose the number of processors to reach the simulation time beforehand. Beside this, HPC infraestructures need an estimation of the CPU time to allocate users in their queues and the proposed strategy offers a very close approach of that time.

From the computational point of view, the performance offered by this kind of decompositions is good since the actual calculation time decreases as the number of subdomains increases.

It is important to note that the paralellization does not only imply the

distribution of the calculation steps among different processors but also the distribution of the memory access. This is the main reason why the achieved speed-up is non-linear. Present day computer architectures are paying special attention to this factor, making bigger cache sizes and developing new levels for them with the only purpose of decreasing the misses associated to main memory access.

The proposed parallelization strategy is a simple technique easy to adapt to existing explicit finite volume models. The performance obtained in terms of computational time reduction for realistic inundation flow problems over complex topographies is encouraging.

References

[1] Chapman, B., Jost, G., Pas, R. v. d., 2007. Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation). The MIT Press.

[2] Garcia-Navarro, P., Vazquez-Cendon, M. E., 2000. On numerical treatment of the source terms in the shallow water equations. Computers and Fluids 29 (8), 951 – 979.

[3] Godunov, S., 1959. A difference method for numerical calculation of discontinuous solutinsof the equatins of hydrodynamics. Mat. Sb. 47(89) (3), 271–306.

[4] Gropp, W., Thakur, R., Lusk, E., 1999. Using MPI-2: Advanced Features of the Message Passing Interface, 2nd Edition. MIT Press, Cambridge, MA, USA.

[5] Health, M. T., 1997. Scientific Computing. McGraw Hill.

[6] Helmbold, D. P., McDowell, C. E., apr 1990. Modelling speedup (n) greater than n. Parallel and Distributed Systems, IEEE Transactions on 1 (2), 250–256.

[7] Karp, A. H., Flatt, H. P., May 1990. Measuring parallel processor performance. Commun. ACM 33, 539–543.
URL `http://doi.acm.org/10.1145/78607.78614`

[8] Message Passing Interface Forum, September 2009. MPI: A Message-Passing Interface Standard, Version 2.2. High Performance Computing Center Stuttgart (HLRS).

[9] Murillo, J., Burguete, J., García-Navarro, P., 2008. Analysis of a second-order upwind method for the simulation of solute transport in 2d shallow water flow. International Journal of Numerical Methods in Fluids 56, 661–686.

[10] Murillo, J., García-Navarro, P., June 2010. Weak solutions for partial differential equations with source terms: Application to the shallow water equations. J. Comput. Phys. 229, 4327–4368.
URL http://dx.doi.org/10.1016/j.jcp.2010.02.016

[11] Murillo, J., García-Navarro, P., Burguete, J., Brufau, P., 2007. The influence of source terms on stability, accuracy and conservation in two-dimensional shallow flow simulation using triangular finite volumes. International Journal for Numerical Methods in Fluids 54 (5), 543–590.

[12] Neal, J. C., Fewtrell, T. J., Bates, P. D., Wright, N. G., 2010. A comparison of three parallelisation methods for 2d flood inundation models. Environmental Modelling & Software 25 (4), 398 – 411.
URL http://www.sciencedirect.com/science/article/pii/S1364815209002965

[13] Roe, P. L., August 1997. Approximate riemann solvers, parameter vectors, and difference schemes. J. Comput. Phys. 135, 250–258.
URL http://dl.acm.org/citation.cfm?id=260709.260742

[14] Sanders, B. F., Schubert, J. E., Detwiler, R. L., 2010. Parbrezo: A parallel, unstructured grid, godunov-type, shallow-water code for high-resolution flood inundation modeling at the regional scale. Advances in Water Resources 33 (12), 1456 – 1467.
URL http://www.sciencedirect.com/science/article/pii/S0309170810001429