MOHID parallelization upgrade following a domain decomposition approach

HIDROMOD
modelação em engenharia

Document		
Classification: Public	**Distribution** NVIST	**Observations** n/a
Title MOHID parallelization following a domain decomposition approach		
Keywords MOHID, parallelization, domain decomposition, MPI		
Entity that produced the document Hidromod, Modelação em Engenharia, Lda Rua Rui Teles Palhinha, 4, 1º, Leião, 2740-278 Porto Salvo Tel: 214211373 – Fax: 214211272 Email: hidromod@hidromod.com		Certificação de Qualidade ISO 9001:2008
Client Consellería de Medio Ambiente, Territorio e Infraestruturas Rúa Roma, nº 6. 15707 Santiago de Compostela. A Coruña. Teléfono: (+34) 881 999 654		meteogalicia
Authors Paulo Chambel Leitão		**Title** Civil Eng., PhD Environment
Verification Ricardo Miranda José Chambel Leitão		**Title** Mechanical Eng., Master Environment Civil Eng., Doutor Mech Eng.
Production date 28/1/2014	**Page number** 33	**Reference number** Version 2

HIDROMOD
modelação em engenharia

Index

HIDROMOD
modelação em engenharia

Figure Index

1 Introduction

In this document the MOHID parallelization upgrade done by Hidromod in the framework of the EnergyMare project is described. The work was sub-contracted by MeteoGalicia, one of EnergyMare partners. This upgrade had a global objective of increasing the speedup factor associated with MOHID modelling system's parallelization. A specific goal was to increase the computational efficiency of the operational MOHID hydrodynamic model used by MeteoGalicia. This upgrade will allow running larger model domains with the same horizontal discretization. This way MeteoGalicia could merge the 4 operational MOHID models run for each Ria in a single model covering the entire Rias maintaining the horizontal and vertical discretization. This methodology will allow simulating Rias interaction. Additionally, with this new upgrade MeteoGalicia will be able to run higher resolution models and consequently enhance its capability to accurately identify areas with high potential to generate marine current energy and fulfil one of the EnergyMare goals.

Currently MOHID has two main parallelization approaches implemented: (1) one where each MOHID model/sub-model runs in a different process and the communication between processes is done using MPI; (2) In the other approach each “do loop” is parallelized using OpenMP directives; MPI allocates processes and OpenMP allocates threads.

Along this document “process”, “thread” and “core” concepts will be used. The first two are used when the discussion is focused in the software (MPI – processes, OpenMP - threads) and third one (core) in the hardware (e.g. speedup quantification). To avoid confusions, it is important to remember that a thread is a single line of commands that are getting processed; each application has at least one thread, most have multiples. In computing, a process is an instance of a computer program that is

being executed. It contains the program code and its current activity. Depending on the operating system (OS), a process may be made up of multiple threads of execution that execute instructions concurrently. A core is the physical hardware that works on the thread. In general a processor can only work on one thread per core, CPUs with hyper threading can work on up to two threads per core.

The upgrade done in MOHID in the framework of the EnergyMare project aimed to minimize one of the major limitations of the "one process – one model" approach that is the workload distribution between cores. This limitation was minimized with the domain decomposition concept implementation. The idea is to allow MOHID to associate to one model more than one process. By default, MOHID assumes "one model – one process" but in the new version the user has the possibility to associate to one model more than one process. In this approach a model is broken-down in sub-domains that communicate two-way along sub-domains interfaces.

The future goal is to test MOHID implementations following a hybrid programming parallelization approach where MPI (domain decomposition) and OpenMP (do loop parallelization using OpenMP) are used simultaneous. This type of approach is considered to be a consolidated approach and is expected to be one of the near future trends in high performance computing[1]. However in the long term (a decade or more from now) programming languages may support enough native parallelism to obviate the need for OpenMP directives. To better understand the framework behind the strategy described in this report the reader should read the follow article http://www.hpcwire.com/2013/12/03/compilers-accelerated-programming/.

In this document, the methodology is firstly presented, followed by a results chapter where the increase of performance due to the new upgrade is quantified for schematic

[1] Michael Wolfe, at PGI, writes about programming standards for the next generation of HPC systems in HPCWire http://www.hpcwire.com/2013/12/03/compilers-accelerated-programming/

and realistic MOHID implementations. Finally conclusions are presented. In this chapter implementation limitations and future development guidelines are also addressed.

2 Methodology

In the first phase of this chapter, an analysis to the parallelization methodologies available in the MOHID standard version is presented. Finally the upgrade done based in the domain decomposition concept is described.

Present MOHID parallelization approach

The present version of MOHID has two main parallelization approaches implemented:

- One model – one process (Figure 1) using MPI. This means if MOHID is run with sub-models the user can allocate each model to a different process. MPICH is the MPI implementation used (http://www.mpich.org/).
- Parallelization of each "do loop" using OpenMP (http://openmp.org/) directives.

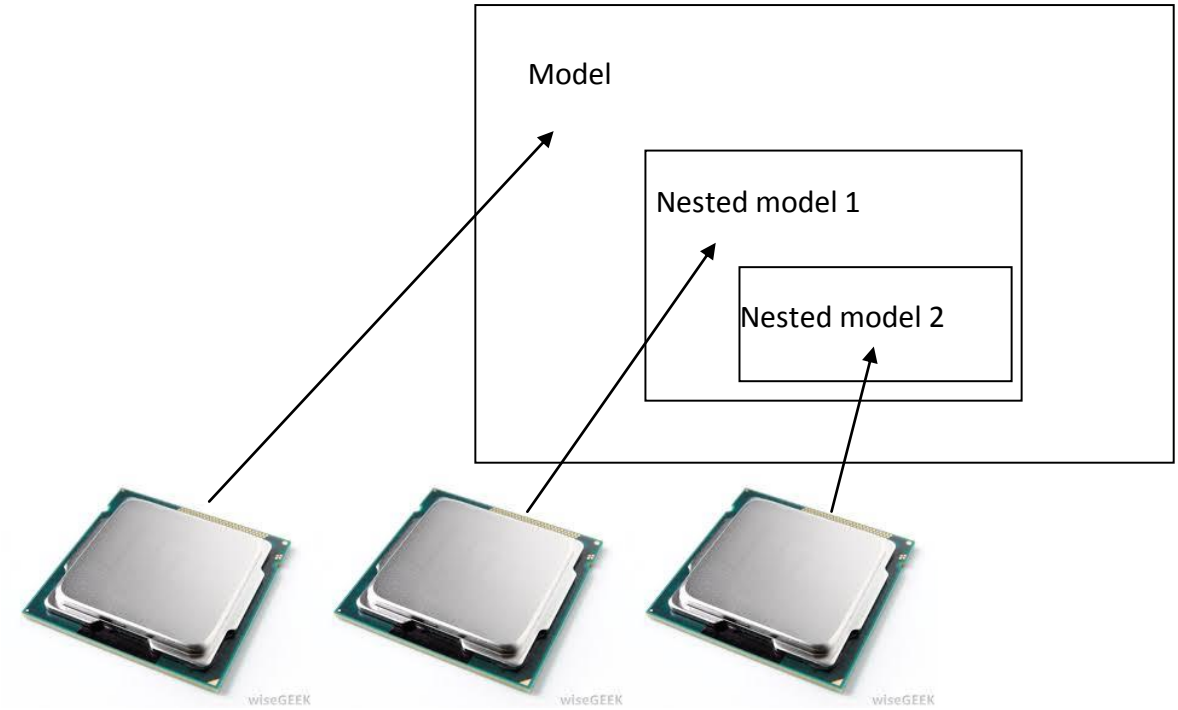


Figure 1 : Diagram showing the workload distribution for the MOHID parallelization methodology "one model – one process".

The MPI based approach has not been used for many years due to an error occurring when "father's" and nested models' time step was not the same. This error is now corrected. Also in the framework of this upgrade, an error in the nesting process between models with different vertical discretizations was identified and corrected.

The positive and negative points of the one model – one process using MPI approach are:

- Positive:
    - Communication between processes is done in the highest levels of the software structure. MOHID allows each model to be "confined" in only one object. This approach takes the most advantage of the object oriented approach followed in the MOHID programming. The main consequence is that the coding is contained in specific modules (e.g. main, model, hydrodynamic) and far away from the more active coding

areas. These areas are the ones changed more often due to constant upgrades to add new functionalities;

- Because this methodology was implemented using MPI this means that it can be used in hardware architectures with distributed memory (e.g. clusters);

- Negative:

  - Lack of flexibility in the workload distribution between cores. The parallelization is restricted to MOHID implementations with nested models. Even in the implementations with several nested models the only way to balance workload by core in a uniform way is if users define models with a spatial and temporal discretization such that the computation effort by model is equal. This can be very difficult, particularly for MOHID beginners and even for experienced ones once it is very time consuming;

  - In MPI the communication between processes needs to be coded;

  - Parallelization by itself changes the programming paradigm. Programming in MOHID with MPI is very demanding. The level of abstraction increases a lot. A simple “do loop” with MPI communication can be quite demanding (e.g. nesting communication).

The positive and negative points of parallelization of each “do loop” using OpenMP directives:

- Positive:

  - The level of abstraction for “do loops” parallelization is very easy;

- The communication between threads is done automatically when OpenMP is used. The OpenMP directives allows a more easy parallelization of the code at the “do loop” level;
- More efficient in the workload distribution by core than the “one model – one process” old methodology.

- Negative:
  - It is done in a very low level of software structure. Parallelization at higher levels tend to be more efficient because decreases the number of parallelization instructions;
  - The parallelization at do loops level generates a lot of extra source code focus only in parallelization;
  - Only the new Fortran compilers allow efficient debugging of parallel code (e.g. Intel 12).

Domain decomposition implementation

The upgrade done in MOHID aimed to minimize the major limitation of the “one process – one model” approach that is the workload distribution between cores. This goal was achieved implementing the domain decomposition concept.

Instead of implementing the domain decomposition from scratch, the use of the software structure already present in MOHID to run several nested models was decided. In MOHID, when in the compilation phase, the preprocessor definition “_USE_MPI” is add each model is run as an independent process. In the new upgrade this option is still valid but the user can additionally break down any model by several processes. This way the user has more control over the computational workload distribution between cores. This break down is done in the **tree.dat** input file where all

models and nested models are listed. If the user wants to break down a specific model, he only needs to add in front of a specific model ": " and the number of processes that he wants to allocate to a specific model (e.g. " : 4" – in this case the user wants to decompose a model in 4 sub-domains). In Figure 2 a **tree.dat** example is presented where the user wants to run the simulation in 8 processes. In this example one process is allocated to the "father model" and the remaining 7 processes to the nested model (or son model).

```
3  +..\..\RiasBaixasA_MPI_8_cores\exe
4  ++..\..\RiasBaixasA_MPI_8_cores\Arousa\exe : 7
5
```

Figure 2 : Example of a tree.dat file where the domain decomposition is turn on.

This new feature is only active if MOHID is compiled with the preprocessor definition "_USE_MPI". It is recommended to compile MOHID in 64 bits because it has the following advantages, when compared to the 32 bits option:

- It is 10-30% faster;
- It allocates ~30% less RAM;
- It does not have any RAM limitations is opposition to the 32 bits that have strong limitations for RAM allocations higher that 2 Gb.

If the 64 bits windows version the extra preprocessor definition "_NO_NETCDF" must be add because the netcdf dll for 64 bits are not compiled yet. For compiling MOHID in NIX platforms, the steps described in MOHID Wiki must be followed: http://www.mohid.com/wiki/index.php?title=*NIX_platforms.

To execute the MOHID using MPI it is necessary to install in the computer the MPICH (www.mpich.org) or similar. These installers have also the MPI libraries necessary to the compilation procedure. The easiest way to run Mohid using MPI is with the following command line:

>mpiexec -np 2 -localonly M:\aplica\MPI\FATHER\exe\MohidWater.MPI.exe

In the above example, MohidWater is running with 2 processes, using only local machine's resources.

To run MohidWater using multiple computers use something like this:

>mpiexec -hosts 2 matrix 1 poseidon 2 -map M:\\matrix\projects M:\aplica\MPI\FATHER\exe\MohidWater.MPI.exe

Both machines are required to have the same executable. Here an explicit map is made so that both machines know where the executable is. M: must be a mapped drive and mpiexec must be evoked from the mount point. -map will create the mount in slave machines and unmounts it at the end of the run.

Each sub-domain writes its output in separate files. Therefore, when the model ends, a post-processor tool that glues the sub-domain's outputs in single files must be executed in the directory where the input file "tree.dat" is located. The post-processor tool is named DomainDecompositionConsolidation.exe and it is available (like MOHID source code) via http://mohid.codeplex.com/ (source code directory: software\domaindecompositionconsolidation) developed in collaboration with Maretec-IST.

Decomposition methodology

By default, MOHID breaks down a model domain assuming sub-domains with the same number of lines (Figure 3).

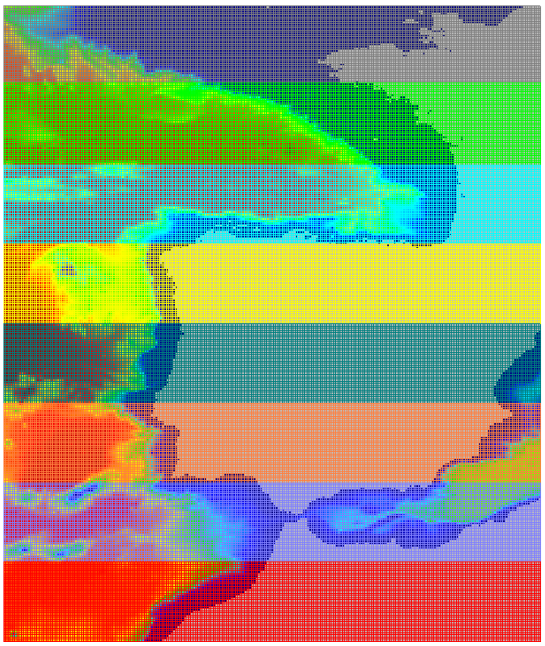


Figure 3 : Default domain decomposition done by MOHID.

However, the user can specify a different domain breakdown (Figure 4) via a specific input file. The name of the input file is defined in the bathymetry input using the keyword D_DECOMP.

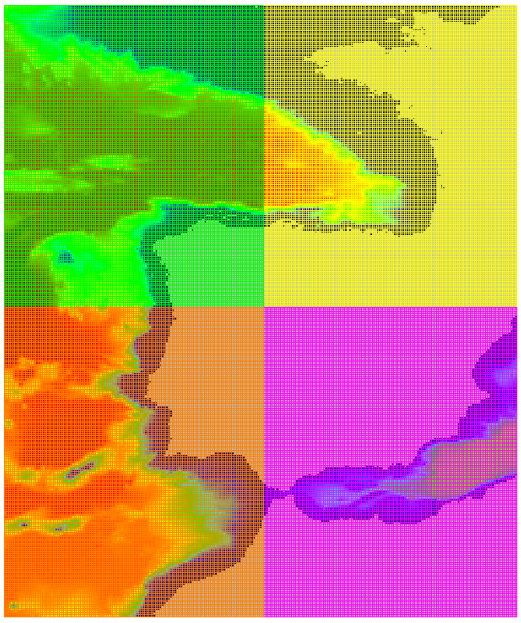


Figure 4 : Possible domain decomposition specified by the user.

In this new input file, the user needs to define, for each sub-domain ID (MPI_ID), the limits of each boundary (ILB_IUB, JLB_JUB) and the number of boundaries (INTERFACES_NUMBER). Additionally the user needs to specify for each boundary the adjacent sub-domains (Figure 5). Boundaries are divided in two types: along lines (<BeginInterfaceWE>/<EndInterfaceWE>) and along columns (<BeginInterfaceSN>/<EndInterfaceSN>).

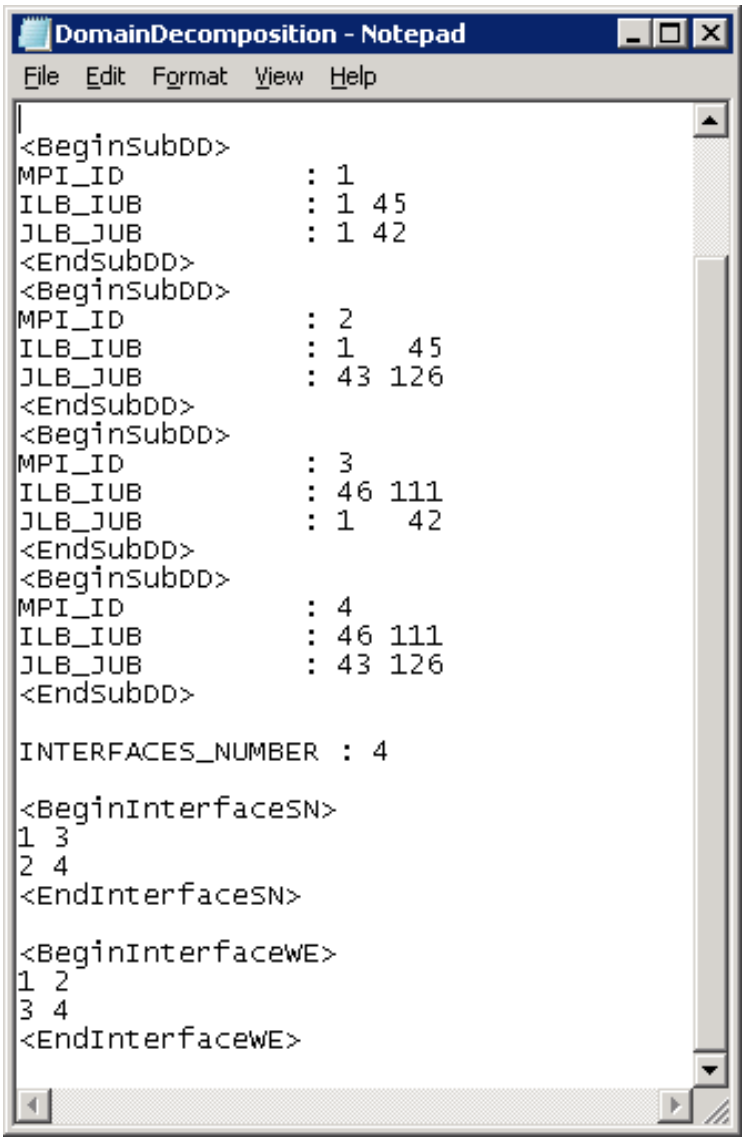


Figure 5 : Example of a domain decomposition procedure specified by the user.

The numeration of the sub-domains' ID (MP_ID) follows the order of the models list in the **tree.dat** file. In the case presented in Figure 2, the MPI_ID of the first model is 0 and for the 7 sub-domains of the second model it goes from 1 to 7. By default the order of these ID's between sub-domains follows the order of the line's indexes (Figure 6). If the decomposition procedure is specified via input file, the order is the one chosen by the user.

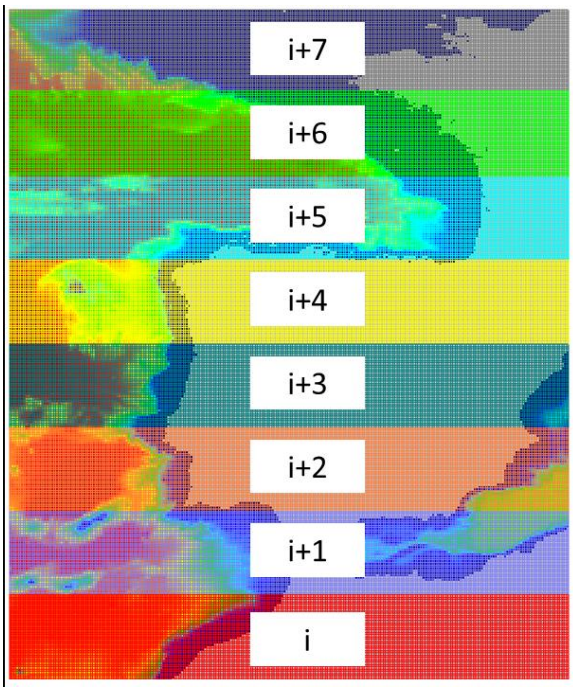


Figure 6 : Default sub-domain's ID order.

Additionally the user can specify the dimension of the halo area. By default this dimension is 2 cells (HALOPOINTS keyword) to cope with second order advection methods and the biharmonic filter. This default value implies that in adjacent sub-domains the number of lines (or columns depending from the boundary orientation) that are overlapped is 4. However, if a first order advection method is used and the biharmonic filter is disconnected the MOHID user can specify only 1 HALOPOINTS reducing to 2 the number of overlapping lines (or columns).

Changes to MOHID projects

Discharges and Time Series location must be given in coordinates using the following keywords:

COORD_X : -9.5556

COORD_Y : 42.3455

The location using the cells indexes (keywords I_CELL : line and J_CELL : columns) cannot be used when a domain decomposition approach is used. Additionally the keyword IGNORE_ON must be set to true (IGNORE_ON : 1) in the discharge input file. This way if a discharge is outside of the sub-domain the model does not stop and ignores it. The necessary change, for example, in the discarges_x.dat file of the original Arousa project run in operational mode by MeteoGalicia are presented below.

```
IGNORE_ON                   : 1

<begindischarge>
NAME                        : Ulla
DESCRIPTION                 : Automatic Data value
!I_CELL                      : 110
!J_CELL                      : 124
COORD_X                     : -8.723866
COORD_Y                     : 42.67763
K_CELL                      : 34
ALTERNATIVE_LOCATIONS       : 0
DISCHARGE UNIFORM           : 1
```

Specific changes to MeteoGalicia implementations

The level 1 model of MeteoGalicia implementation had a 3600 s time step when the ROMS solution hdf5 file had a time step of 900 s. The time step of level 1 should be 900 s and not 3600 s.

In the new version of MOHID, in the input file atmosphere_x.dat for the data block of property “precipitation”, it is necessary to add the following keyword

“ACCUMULATE_VALUES : 1”. The specified units are “mm” so based in the input data (in mm) the MOHID model needs compute the flux in time (e.g. mm/h).

The “RibaixasA” project, operationally run by MeteoGalicia, was changed following the specifications presented above. The new project can be downloaded from https://www.dropbox.com/s/r8a1yl28yrmd8yk/RiasBaixasA.rar.

3 Results

The MOHID upgrade was tested extensively so far for two implementations:

- schematic case entitled “Freshwater Square”;
- operational implementation done by MeteoGalicia for Ria de Arousa.

The goal of the tests was to check the increase of speed of the new update versus the number of cores relative to a MOHID executable with no parallelization. Additionally the same tests were made for the standard version of MOHID where the parallelization is done at the “Do loops” level. The idea was to check the improvement of this new upgrade relative to the present parallelization option available to MOHID users.

The testing has focus in the MOHID executable for windows in single precision for 32 bits. This is the MOHID executable configuration more disseminated among MOHID users. Additionally tests were done for 64 bits version.

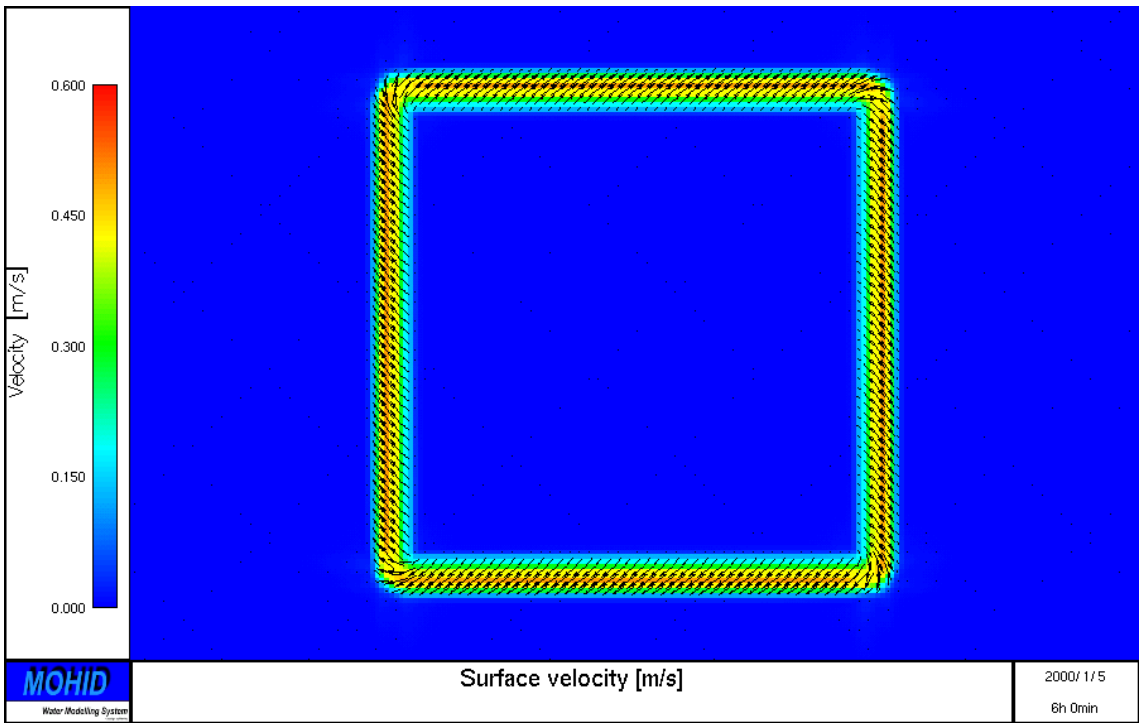
Freshwater Square

The “Freshwater Square” (schematic case) consists in a model with constant depth of 1.000 m and no land points. The domain is a square and has 1.000 km per 1.000 km. This implementation only has one model level (no nesting). The spatial horizontal discretization is 500x500 cells with a spatial step of 2 km. Vertically a 10 layer sigma discretization with equal thickness was assumed. The only water property simulated was salinity. A k-ε model is used to simulate vertical turbulence and in horizontal a constant horizontal viscosity was defined.

The initial condition for salinity is a constant value in all the domain of 36 psu except in a centre square (200 km x 200 km) where salinity is 35.9 psu in the entire water column.

A 2 cells halo region was used to cope with the second order advection numerical scheme usually used in ocean implementations. The domains' decomposition was done dividing the original domain only along the lines (X direction or West-East). This means all domains have always the same number of columns.

This model is run for a period of 1 day with a time step of 90 s. After an initial spin up, due to the geostrophic adjustment a clock-wise rotation is established at the surface along the isohalines being the maximum velocity located where the salinity gradient is also maximum (upper panel - Figure 7). Near the bottom a similar circulation is established but with counter-clockwise rotation (lower panel - Figure 7). The circulation is forced by the radial density gradient. At the surface the only force that is able to balance the sea level gradient (barotropic force) is the coriolis force. At mid water the baroclinic force balances the barotropic force (null velocity) but near the bottom the baroclinic force overpowers the barotropic force and the rotation is inverted.



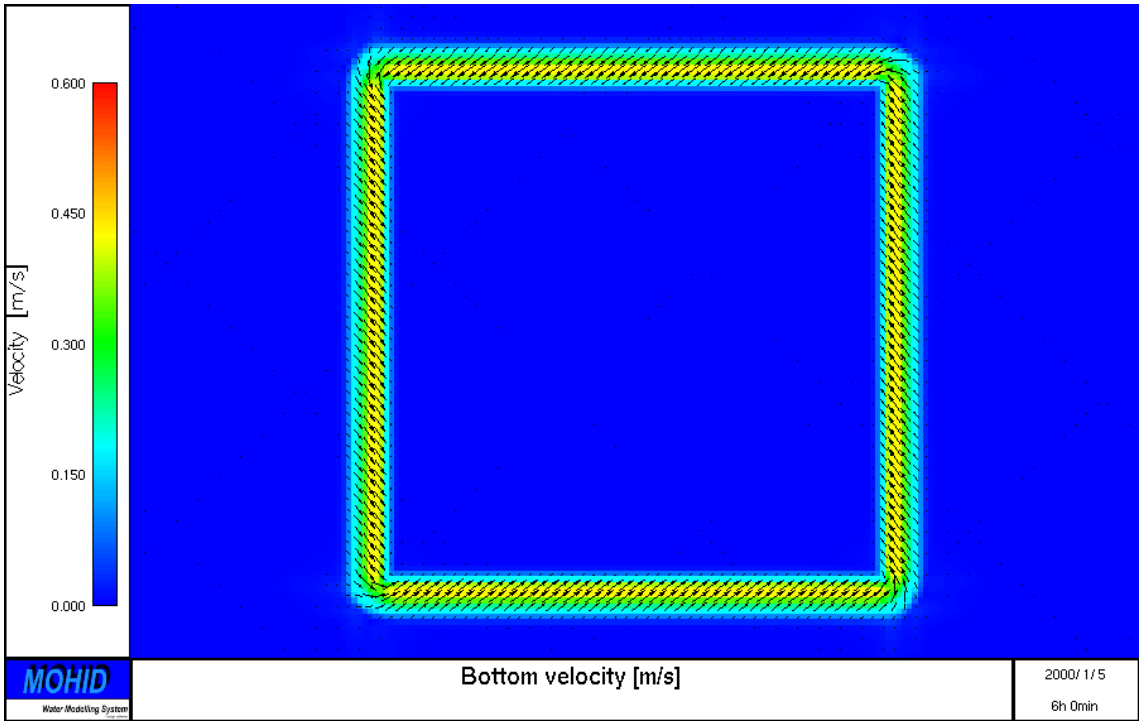


Figure 7 : "Freshwater square" surface velocity when the equilibrium state is reached. Upper panel – velocity field at the surface, bottom panel – velocity field near the bottom.

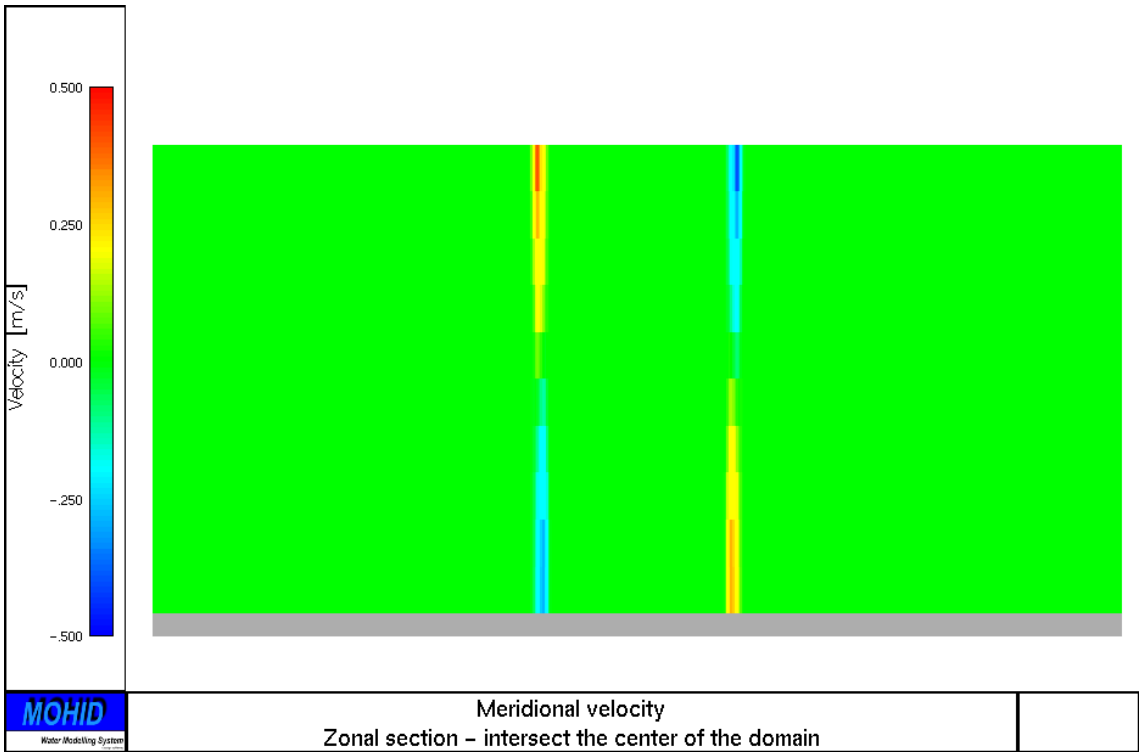


Figure 8 : Zonal section in the domain center of the meridional velocity. The positive velocity is a positive meridional velocity. Near the surface it is possible to see the signal of the clockwise circulation and in the bottom the anticlockwise circulation associated to salinity gradient.

The tests were done in two computers with the following hardware configurations:

- Intel® Xeon® CPU E5420 2 processors 2.53GHz with 4 processors (total 8 cores), 12 Gb of RAM and 64 bit operating system.

- Intel(R) Xeon(R) E5-4620 2.20GHz 4 processors (32 cores), 64 Gb of RAM, Windows Server 2012 R2 Standard

For the first computer the tests done are resumed in Figure 9. For the "Freshwater Square" schematic case with 4 cores the domain decomposition using MPI option is able to have a speed up of 3 while the do loop parallelization using OpenMP has a speedup slighter above 2. For 8 cores the first option as speedup of almost 5 while the second option has a speed of almost 3. For this case, and for 8 cores, this means the new upgrade is able to almost double the performance of the standard parallelization

methodology of MOHID. Using the MOHID 64 bits version a slight increase in the speedup factor for the domain decomposition using MPI functionality is achieved. For example, in 8 cores the MOHID 32 bits version has a speedup factor of 4.7 and the 64 bits version a speedup factor of 5.3.

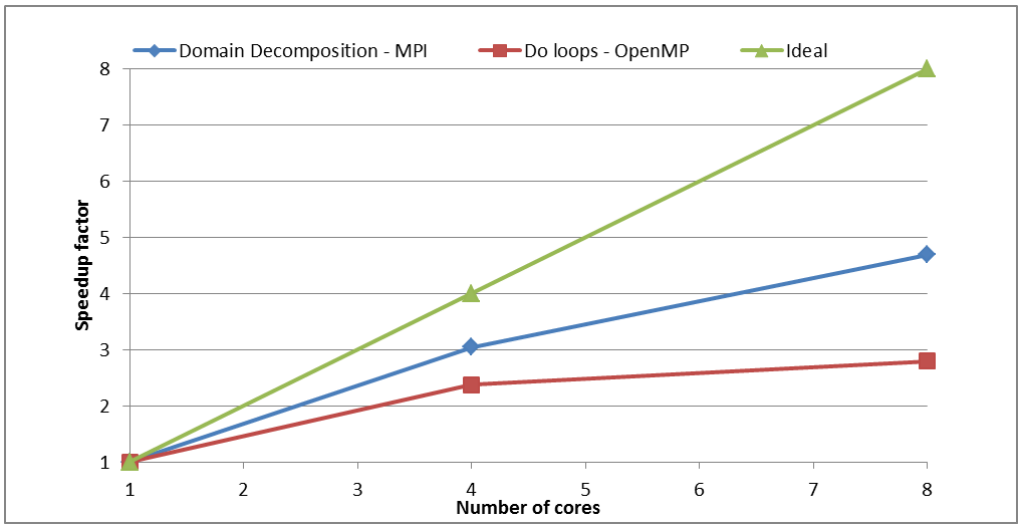


Figure 9 : Speedup factor for MOHID in 32 bits run in 8 cores (2.53GHz) computer. Green line – ideal speedup, blue line – speedup with the domain decomposition option implemented using MPI. Red line – speedup using the do loops parallelization with OpenMP option.

For the 32 cores computer, the domain decomposition MPI implementation reveals a speedup factor with a constant growth rate until 12 cores (speedup factor of 6 – see Figure 10Figure 10). The speedup factor still growths from 12 to 24 cores but with a lower rate. Above 24 cores the speedup factor starts to decrease. **For this MOHID implementation (55x500x10) the ideal number of cores to be used is ~12. Above this number the speedup factor increment is residual.**

This schematic case can be downloaded from https://www.dropbox.com/s/29ztms06668m3n1/FreshWaterSquare.rar ready to be run in a windows platform.

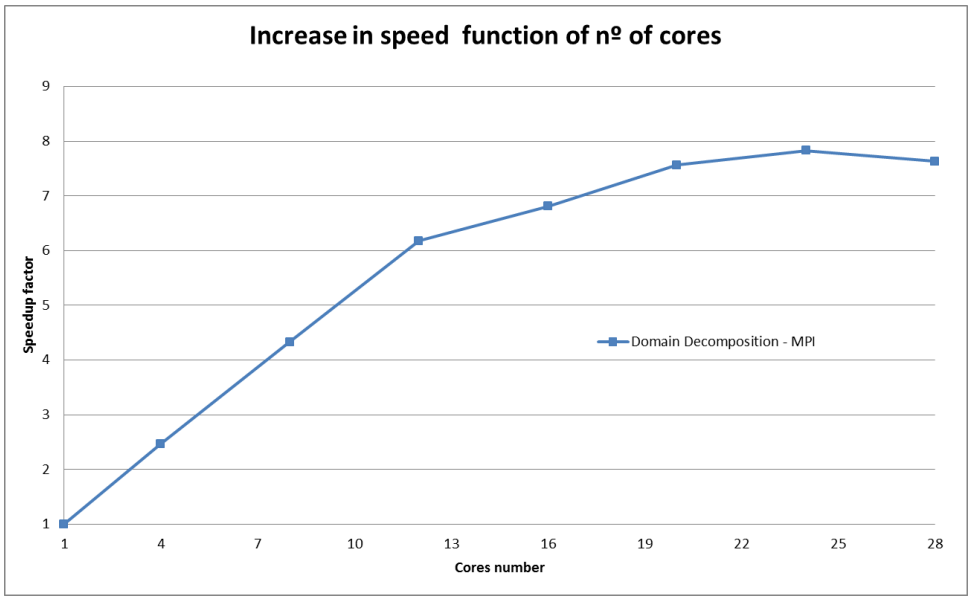


Figure 10 : Speedup factor for MOHID in 32 bits run in 32 cores (2.2GHz) computer. Blue line - domain decomposition option implemented using MPI.

Ria de Arousa operational implementation

MeteoGalicia presently runs MOHID in operational mode for several Rias along the Galicia coast. These models are nested in off-line with a ROMS implementation for the entire Galicia coast (

Figure 11).

The new upgrade of MOHID allows for this case a speedup factor of 4 for 8 cores (Figure 12). In this case the speedup factor increase using domain decomposition, relative to the do loops parallelization, is lower than the previous schematic case. In this case, for 8 cores the increase is only 30% (Figure 12) while in the previous case was 50% (Figure 9) for the same hardware. The possible reasons for this difference are listed below:

- The cells number of the Arousa (cells = 126x111x34) implementation is 5 times smaller than the schematic case (cells = 500x500x10). The domain decomposition speedup factor efficiency is proportional to the number of computation cells. The bigger the number of the computation cells the closer is the speedup factor to the ideal speedup limit;

- The domain decomposition implementation was integrated in the original implementation of MPI that allocates at least one model to one process. This means that one core is allocated to the first level that only reads every 15 minutes the ROMS solution from an hdf5 file. This core is underused along the simulation. The other 7 cores (7 sub-domains resulting from the domain decomposition approach) are used intensively to simulate the Arousa model hydrodynamics. This means that there is a slight unbalance of the computation effort by the available cores.

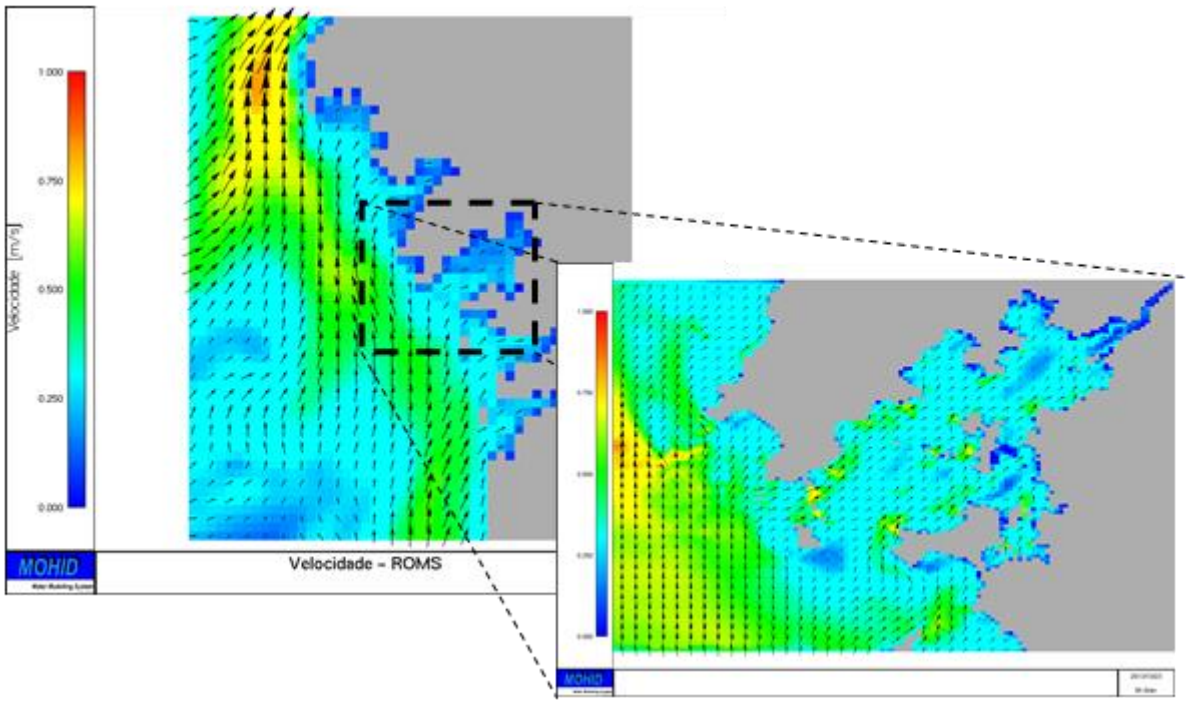


Figure 11 : Surface velocity: Left panel ROMS output used to define the open boundary of the MOHID models implementation (Level 1). Right panel MOHID model implemented for Ria Arousa (Level 2).

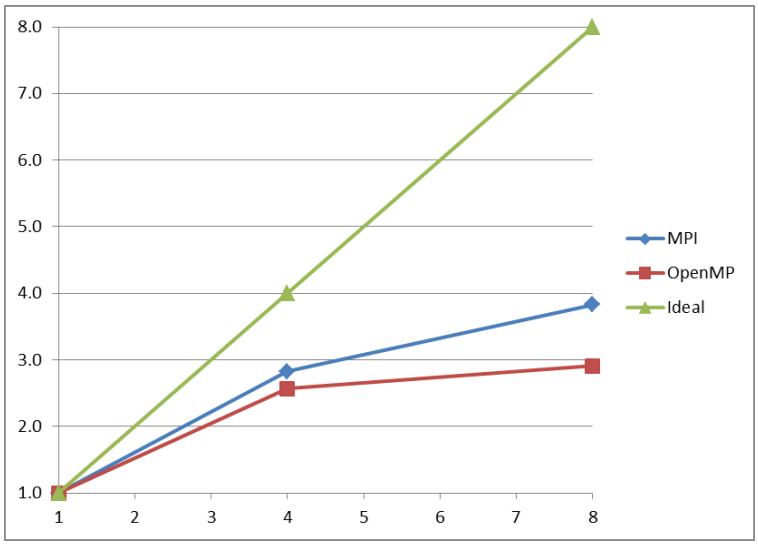


Figure 12 : Speedup factor for MOHID in 32 bits run in 8 cores (2.5G3Hz) computer. Green line – ideal speedup, blue line – speedup with the domain decomposition option implemented using MPI. Red line – speedup using the do loops parallelization with OpenMP option.

4 Common errors and limitations

Land mapping hdf5 name

The error " HDF5ReadWindowI4_2D - ModuleHDF5 - ERR20" or "HDF5ReadWindowI4_3D - ModuleHDF5 - ERR20" can occur when the domain decomposition/MPI functionality is used in MOHID projects with input files (initial or boundary conditions) in hdf5 format. When the domain decomposition / MPI option is ON each subdomain only read a subsection (sub-window) of the hdf5 boundary/initialization input files. In this procedure the Field4D module is used. This module is able to construct all the modules related with the spatial discretization just reading the hdf5, for example, 2D and 3D mapping. The Field4D module associates by default the follow names to the land mapping:

- 2D mapping : *WaterPoints* and *WaterPoints2D*
- 3D mapping : *WaterPoints3D*

When a different name is used from the three already presented the errors presented above occur. This error is related with the use of a nonstandard name in an hdf5 for the land mapping field. In this case it is necessary to specify the land mapping name using the keyword MASK_GRID in every block you are defining an hdf5 input file. For example, if the name “LandCells” is used in the hdf5 file you need to define something like :

```
<beginproperty>
…
FILENAME : WindVelocity.hdf5
MASK_GRID : LandCells
…
<endproperty>
```

In other cases where the user need to specify the name of the land mapping is when the HDF5 file have two land mapping field, for example, “WaterPoints” and “WaterPoints3D”.

Nesting and automatic domain decomposition

There is a limitation when the user wants to use the domain decomposition approach in a model and associated nested models (Figure 13). If the user chooses the automatic decomposition the “father” sub-domain of all nested models associated will be the model sub-domain to be allocated in last or the upper sub-domain (higher indexes lines). For example, if I have model with 100x100 cells and I decomposed it in two sub-domains the one that automaticly will be considered the father model of associated sub-models will be model will start in line 51 and ends in line 100.

This limitation can be workaround if the user specifies manually the domain decomposition mapping. The user only needs to guarantee that the domain with highest ID is the father model.

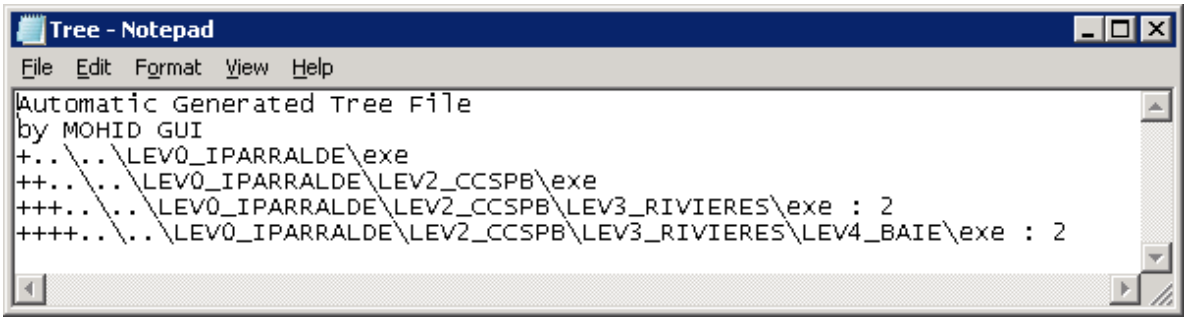


Figure 13 – Tree.dat example where the user wants to use the domain decomposition approach in a model (Lev_Rivieres) and associated nested model (Lev4_Baie).

Father/nesting vs domain decomposition

The present version assumes if a “father” model is decomposed the associated nested models are linked with only one sub-domain (Figure 14).

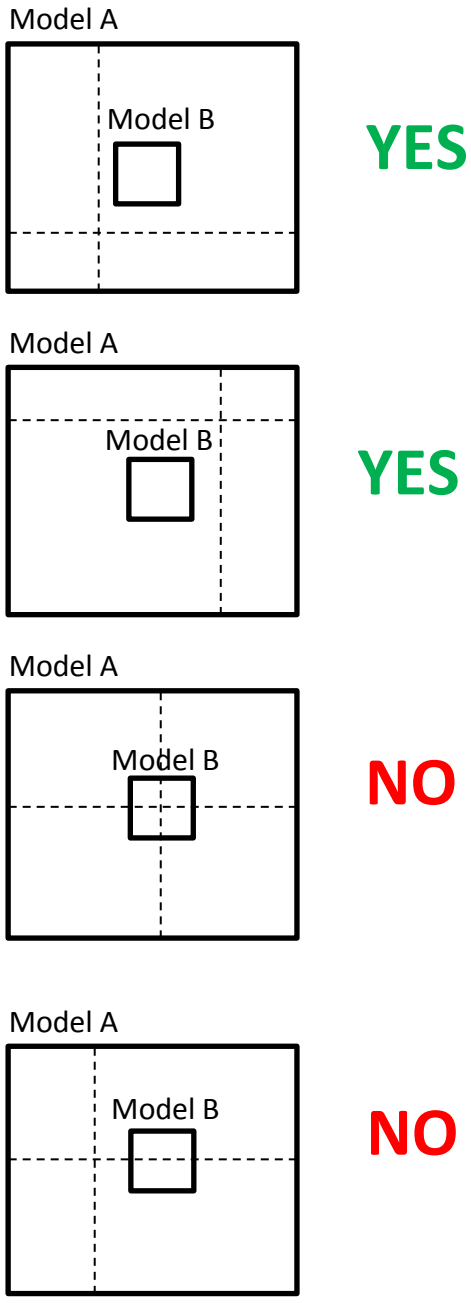


Figure 14 – Four scenarios of a Domain decomposition of “father model” (model A) in 4 sub-domains with a nested model (model B): top two – valid and bottom two - invalid.

Problem with the SUBMODEL_EXTRAPOLATE option

When a model is break down in very small sub-domains the option SUBMODEL_EXTRAPOLATE (horizontal extrapolation of the “father solution” velocity field using the nearest neighbour approach) start to have problems, for example, the model can stop with the follow error:

"ExtraPol3DNearestCell_8 - ModuleFunctions - ERR10"

When this error happens the user should replace the SUBMODEL_EXTRAPOLATE (SUBMODEL_EXTRAPOLATE : 1) option by the MISSING_NULL (MISSING_NULL : 1). This last option assumes null values in the horizontal extrapolation process of the “father solution” velocity field.

5 Conclusions

The new MOHID parallelization upgrade (based in the domain decomposition concept) allows a speedup factor of ~0.5 X cores number for 3D baroclinc implementations with 0.5 to 2.5 million computational cells. This speedup factor is valid from 2 to 13 cores. Above this limit the speedup increase is residual and above 24 cores there is even a decrease of the speedup. A speedup factor of 8 (24 cores) was the maximum speedup achieved for ~2.5 million cells 3D baroclinic implementation for a computer with 32 cores (2.2GHz). An efficiency increase is expected for larger grids and for implementations simulating more hydrodynamic or biogeochemical processes (e.g. primary production).

The speedup increase due to the new upgrade, relative to the do loop parallelization with OpenMP directives, is approximately 30% (for 0.5 millions computational cells) to 50% (2.5 millions computational cells) for 8 cores. Globally, the new method speedup factor, relative to the old method, grows with the number of cores and with the number of computation cells.

This new update opens the door for testing MOHID in clusters. Another important future step will be to test strategies of using the “One process – One model” (MPI) together with the do loop parallelization (OpenMP directives).

Operational implementations of the MOHID model done is several points of the globe are usually for grids of the order of 200x200x50. Hidromod’s conviction is that with this new upgrade it is possible to start implementing models with 1 to 2 orders of magnitude more computational cells. In the case of the Galicia coast, it is recommended that MeteoGalicia implements a model at least 10 times bigger than the one already being run for Ria de Arousa. The old grid is 111x126x34 and, for the present hardware of MeteoGalicia, it is possible to start testing a computational grid of

1000x200x34. A huge step forward would be to start testing MOHID in a high performance computing environment like the one managed by CESGA. It would be interesting to analyse the MOHID performance, for example, running an implementation of 2000x2000x50 in hardware's facilities with a number of nodes of the order of 1000.