OASIS3-MCT tutorial

November 2012

All the documentation about the coupler OASIS3-MCT can be found on the OASIS web site at http://oasis.enes.org and in the OASIS3-MCT sources in the oasis3-mct/doc directory.
The current OASIS3-MCT coupler uses internally the Model Coupling Toolkit (MCT) developed by the Argonne National Laboratory (http ://www.mcs.anl.gov/mct) to perform parallel regridding and parallel exchanges of the coupling fields.

I – Tutorial

A. Compiling OASIS3-MCT and running an empty "tutorial" toy model

1. Login with login and password provided and open a terminal session clicking Applications -> System Tools -> Terminal in the top bar. Download the OASIS3-MCT sources using :
   - ➢ mkdir oasis3-mct ; cd oasis3-mct
   - ➢ svn checkout http://oasis3mct.cerfacs.fr/svn/trunk/oasis3-mct .
2. To compile the OASIS3-MCT coupler:
   - Go into directory oasis3-mct/util/make_dir
   - Load the module pgi with the command: "module load pgi". You have to type this command every time you open a new terminal. You need to do load this module to compile with pgi Fortran compiler.
   - Adapt the value of $ARCHDIR in the platform header makefile make.pgi_cerfacs to have the compiled sources in e.g. $(HOME)/oasis3-mct_comp (or another local directory)
   - Adapt the "make.inc" file to include your platform header makefile make.pgi_cerfacs.
   - Type "make realclean –f TopMakefileOasis3" and then "make –f TopMakefileOasis3"
   - The libraries "libmct.a", "libmpeu.a", "libpsmile.MPI1.a" and "libscrip.a" that need to be linked to the models are available in the directory $ARCHDIR/lib
3. To compile the tutorial models:
   - Go into directory oasis3-mct/examples/tutorial
   - Type "make clean; make" (note that the Makefile in this directory automatically includes your header makefile – see the first line in the Makefile)
   - ➢ The executables model1 and model2 are available in the current directory.

4. To run the two tutorial component models:
   - Adapt to your platform and execute the script run_tutorial:
     > ./run_tutorial
     The results of the component models are now in subdirectory $rundir defined in run_tutorial (i.e. ${HOME}/oasis3-mct/examples/tutorial/work_tutorial by default).
   - ➢ In their current form, "model1.F90" and "model2.F90" are not interfaced with the OASIS3-MCT library and do not perform any exchanges. They just run for 6 time steps of 1 hour each without doing anything specific.

B. Interfacing and running the "tutorial" toy model with OASIS3-MCT

After your modifications, the "tutorial" toy model should reproduce the coupling between two simple codes, model1 and model2, with the OASIS3-MCT coupler.

The total run is 6 hours by default. Model1 has a time step of 3600 seconds and runs on a logically-rectangular (182x149) grid. Model2 has a time step of 1800 seconds and runs on a logically-rectangular (96x72) grid.

The coupling exchanges to implement are as follows:

- Every 2 hours, model1 sends the coupling field “FSENDOCN” to model2 that receives it as “FRECVATM”.
- Every 3 hours, model2 sends the coupling field “FSENDATM” to model1 that receives it as “FRECVOCN”.
- In addition, model1 also outputs to a file the field "FSENDOCN" every 2 hours.

1. Have a look at the OASIS3-MCT configuration file "namcouple", located in the directory oasis3-mct/examples/tutorial/data_oasis3, written to perform the exchanges described above. The interpolation specified to transform "FSENDATM" into "FRECVOCN" is the SCRIPR/BILINEAR one (i.e. the weight and addresses file used for the interpolation will be calculated at the first coupling time step with the SCRIP library). Interpolation from "FSENDOCN" into "FRECVATM" will use a pre-existing weight and addresses file (my_remapping_file_bilinear.nc).
2. Modify "model1.F90" and "model2.F90" so to implement the initialisation, definition and declaration calls of OASIS3-MCT library. The lines where to introduce the OASIS3-MCT specific calls are marked with ""!! TOCOMPLETE …". As a first step, suppose that each model will run on only one process. Most of the variables that will be used in the oasis routines calls are already defined at the beginning of the programs. A step-by-step approach is recommended, i.e. validate one-by-one each call implemented by compiling the toy coupled model and checking that the toy model runs fine until the calls.
3. Modify "model1.F90" and "model2.F90" so to implement the coupling exchanges, as described above. In the time step of the models, try first by implementing the reception of the input coupling fields (“oasis_get”) at the beginning of the time step and then, the sending of the output coupling field (“oasis_put”), at the end of the timestep.
4. Try to run the toy coupled model with the script "run_tutorial". In this configuration, where both models call their oasis_get before their oasis_put, a deadlock will occur in the coupled toy model. Find which modifications are needed in the toy to remove this deadlock without changing the order of the oasis_get and oasis_put (see section 2.10.1 of User Guide for more details).

- The results of the tutorial coupled model are now in subdirectory ${HOME}/oasis3-mct/examples/tutorial/work_tutorial. In "work_tutorial", the file "nout.000000", written by one of the master process of the models, contains the information read in the configuration file "namcouple" by all the processes. The OASIS3-MCT debug files debug.root.01 and debug.root.02 are written by the processor of each model. The level of debug information in these files depends on the value of NLOGPRT defined in the namcouple. Here NLOGPRT=10 and the files contain normal diagnostics production, with initial debug info and calling trees of the routines (see the section 3.2 of the User Guide for more details).
- You can visualize the results with ferret and the scripts script_ferret_*.jnl. Type "ferret" and then "go xxx.jnl" where xxx.jnl is the name of the ferret script you want to run. (Note that to run another ferret script, you have to restart ferret.)
- Explain why the number of time steps in the file FSENDATM_model2_02.nc is not the same than in the file FRECVOCN_model1_03.nc.
- Keep your results by renaming /work_tutorial into /work_tutorial_B4_mono

In this configuration, model1 and model2 run with one process each. This is indicated in the launching script "run_tutorial" (see "nproc_exe1" and "nproc_exe2").

C. Running the models in parallel with OASIS3-MCT

To run and couple model1 and model2 in parallel, one has to ensure that the partition definition transferred to OASIS3-MCT via the call to oasis_def_partition is properly done. In model1 and model2, the information to provide as arguments of the oasis_def_partition is calculated by the subroutine decomp_def that you can check in more detail. Then, the number of processes has to be modified in the launching procedure.

- Specify for example "nproc_exe1=3", "nproc_exe2=3" in the script run_tutorial .
- In oasis3-mct/examples/tutorial, execute "run_tutorial", which then launches the models on 3 processes each.
- Keep your results by renaming /work_tutorial into /work_tutorial_C_para
- Visually compare the results with the non-parallel case B4_mono.

OASIS3-MCT supports different parallel decompositions for the models (see the section 2.4 of the User Guide). Tutorial routine "decomp_def.F90" can define the local partition for the BOX or the APPLE decompositions. By default, the APPLE decomposition is used. To test the BOX decomposition, recompile the tutorial models with, in Makefile: "CPPKEYDECOMP_M1=DECOMP_BOX" or "CPPKEYDECOMP_M2=DECOMP_BOX".

II – Test_interpolation

This toy tests the quality of the interpolation between a source grid and a target grid by calculating the error of interpolation on the target grid. There is only one time step and the coupling is performed at t=0. At t=0, model1 sends its coupling field "FSENDANA" to model2.

A. Quality of the conservative interpolation from “torc” to “lmdz”

As specified in the OASIS3-MCT configuration file "data_oasis3/namcouple" and in the toy namelist file "data_oasis3/name_grids.dat", the toy performs a conservative interpolation between “torc” and “lmdz” using the predefined remapping file myfile_torc_to_lmdz_CONSERV_FRACNNEI.nc. The “torc” and “lmdz” grids are defined in the “grids.nc”, “masks.nc” and “areas.nc” OASIS3-MCT grid data files.

To evaluate the quality of the interpolation between the source grid and the target grid an analytical field is defined by model1 on the source grid and then interpolated on the target grid and sent to model2 (or sent to model2 and then interpolated on the target grid) where the error of interpolation is calculated. The error is defined as the absolute value of the difference between the interpolated field and the analytical field calculated on the target grid, divided by the interpolated field (multiplied by 100 to have it in %).

1. Compiling and running test_interpolation
   - Go to the directory "oasis3-mct/examples/test_interpolation"
   - Load the module pgi with the command: "module load pgi".
   - Compile the toy using : "make clean ; make"
   - Adapt to your platform and execute the script run_test_interpolation: "./run_test_interpolation". The results of the component models are now in the $rundir directory (as defined in the script run_test_interpolation).
2. Analysis of the results
   - After running the test_interpolation toy model, the analytical field on the source grid “torc” is available in file $rundir/FSENDANA_model1_01.nc, the interpolated field on the target grid “lmdz” in file $rundir/FRECVANA_model2_01.nc and the error field, as defined above, in file "$rundir/error.nc". You can find the minimum and maximum of the error in the output file $rundir/model2.out_101.
   - ➢ You can visualize the results with ferret and the scripts script_ferret_*.jnl. Type "ferret" and then "go xxx.jnl" where xxx.jnl is the name of the ferret script you want to run. (Note that to run another ferret script, you have to restart ferret.). Why is the error more important at the North pole and near the coasts ?
   - ➢ Keep your results by renaming /work_test_interpolation into /work_test_interpolation_A2_mono

B. Quality of other interpolations

In the tests below, the SCRIP library is used to calculate the weights and addresses remapping file at the first time step t=0.

1. Bilinear interpolation between “torc” and “lmdz” using the SCRIP library
   - Modify the configuration file "data_oasis3/namcouple" to perform a bilinear interpolation using explicitly the SCRIP library available in OASI3-MCT to calculate the interpolation weights and addresses (see the section 4.3 of the User Guide for more details).
   - Run again the toy test_interpolation executing the script "./run_test_interpolation".
   - Keep your results by renaming /work_test_interpolation into /work_test_interpolation_B1_mono. You can see that a new remapping file has been created rmp_torc_to_lmdz_BILINEAR.nc.
   - ➢ Visually compare the results with the case A2_mono.

2. Bilinear interpolation from “lmdz” to “torc” using the SCRIP library

- Exchange the names of the source grid and the target grid in the namelist file "data_oasis3/name_grids.dat" and put the correct grid dimensions and periodicity in the "data_oasis3/namcouple". Note that the script run_test_interpolation automatically changes the strings “src_name” and “tgt_name” for respectively the source and target acronyms indicated in the “name_grids.dat” .
- Run again the toy test_interpolation executing the script "./run_test_interpolation".
- Keep your results by renaming /work_test_interpolation into /work_test_interpolation_B2_mono.

➢ Analyse the error of interpolation (adapt the scale in the file script_ferret_error.jnl to better visualise the error if necessary).

C. Time statistics: comparison of time using src or dst options of the "MAPPING" transformation

It is possible to get time performance informations with OASIS3-MCT thanks to the variable "TIMER_Debug" defined in the module "mod_oasis_timer.F90" (see the section 6.3.2 of the User Guide for more details).

1. Modification of the sources of "OASIS3-MCT":

- Go to the directory $(HOME)/oasis3-mct/lib/psmile/src. In "mod_oasis_timer.F90" put "TIMER_Debug=1". With this value, time statistics will be calculated over all the processors of each model and each master processor will write a timer file, "model1.timers_0000" and "model2.timers_0000", containing the time statistic results.
- Go back to the directory "oasis3-mct/examples/test_interpolation".
- Compile the toy and the libraries using : "make clean ; make" (note that the toy Makefile automatically recompiles OASIS3-MCT libraries if needed).

2. Time performance for the "MAPPING" transformation using the "src" option **(**see section 4.3 of the User Guide):

- Modify the configuration file "data_oasis3/namcouple" to test the performance of the bilinear interpolation from “lmdz” to “torc” with the pre-existing remapping file rmp_lmdz_to_torc_BILINEAR.nc just created, using the "src" option (in this case, the interpolated field is interpolated from the source grid to the target grid on the source model process(es) and then sent to the target model process(es)). Do not forget to copy the rmp_lmdz_to_torc_BILINEAR.nc from the previous working directory /work_test_interpolation_B2_mono to the /data_oasis3 directory; as the script run_test_interpolation copies all *.nc files from the data_oasis3 directory to the working directory, it will then be available for use by OASIS3-MCT.
- Run the test_interpolation toy model with for example 4 processes per model and keep your results in /work_test_interpolation_C2_para_src .

➢ Look into the files "model1.timers_0000" and "model2.timers_0000". The information given in these files are explained in the section 6.3.2 of the User Guide.

3. Time performance for the "MAPPING" transformation using the "dst" option:

- Test again the "MAPPING" transformation between “lmdz” and “torc” with the remapping file rmp_lmdz_to_torc_BILINEAR.nc in parallel with 4 processes per model

using the "dst" option. In this case, the field is interpolated from the source to the target grid on the target processes.

- Keep your results respectively in /work_test_interpolation_C3_para_dst.
- ➢ Compare the time statistics to the results obtained with the "src" option. In which case and in which file is there a “pmap_001” or a “gmap_001” timer?

D. Other optimization options for the "MAPPING" transformation : bfb and sum

By default, the "bfb" option is activated (see the section 4.3 of the User Guide for more details). It forces the mapping to be done in a bit for bit manner and the results are independent of the domain decomposition. The "sum" option forces the mapping to be done using partial sums. The source values belonging to one particular process and used in the calculation of one target point is locally partially summed before the global sum, involving possible other source values from other processes, is done. The partial sums will result in non bit reproducible results between simulations done with different number of processes. Finally, the "opt" option allows the code to choose either "bfb" or "sum" based on which might be faster.

If the models run on one processor the results are of course independent of the option "bfb" and "sum" (or "opt").

1. "MAPPING" transformation using the "sum" option :
   - Modify the configuration file data_oasis3/namcouple in order to perform a bilinear interpolation from “lmdz” to “torc” with the pre-existing remapping file rmp_lmdz_to_torc_BILINEAR.nc using the ("src" + "sum") options.
   - Run the toy in parallel, always on 4 processes for each model. Keep your results in /work_test_interpolation_D1_para_src_sum.
   - ➢ Compare the interpolated field, the error and the time statistics results with the case in /work_test_interpolation_C2_para_src, corresponding to the case with only "src" option i.e. with “bfb” by default). Verify that the latter gives indeed bit reproducible results compared to the mono case and that this is not the case with “sum” option. Do you see any performance differences between the two cases?
2. "MAPPING" transformation using the "opt" option :
   - Modify the configuration file "data_oasis3/namcouple" in order to perform a bilinear interpolation from “lmdz” to “torc” with the pre-existing remapping file rmp_lmdz_to_torc_BILINEAR.nc using the "src" + "opt" options.
   - Run the toy model in parallel and keep your results in /work_test_interpolation_D2_para_src_opt
   - ➢ Compare the interpolated field, the error and the time statistics results with the case C2_para_src and D1_para_src_sum. Which strategy was chosen in this case ("sum" or "bfb") ?

E. Test of the quality of the bilinear interpolation between the participant's grids.

1. Creation of the grids.nc, masks.nc and areas.nc files, needed to calculated the weights and addresses mapping files, with the grid definitions of the participant's models.

The NCL program "create_aux_files.ncl" located in oasis3-mct/examples/test_interpolation gives an example of how one would create files grids.nc, masks.nc and areas.nc from the two files "data_oasis3/grid_model1.nc" and "data_oasis3/grid_model2.nc" containing respectively the ORCA2T and the LMDz grids. This program can be used as a base to build you own grids.nc, masks.nc and areas.nc files starting from your model grid files. All the documentation about NCL can be found on the NCAR web site: http://www.ncl.ucar.edu/

2. Quality of the bilinear interpolation between "grd1" and "grd2" of the participants:
   - Put the name of your source "grd1" and target "grd2" grids in the namelist data file "data_oasis3/name_grids.dat" and change the dimensions of the grids in the configuration file "data_oasis3/namcouple".
   - Modify the configuration file "data_oasis3/namcouple" in order to perform a bilinear interpolation from the source "grid1" to the target "grid2" using the SCRIP library.
   - Specify "nproc_exe1=1", "nproc_exe2=1" in the script run_test_interpolation. Execute "./run_test_interpolation".
   - Keep your results in /work_test_interpolation_E2_mono
   - ➢ Analyse the quality of the interpolation by plotting the interpolated field and the error of interpolation.

3. Quality of the bilinear interpolation between "grd2" and "grd1" of the participants:
   - Modify the name of your source and target grid in the namelist data file "data_oasis3/ name_grids.dat" and change the dimensions of the grids in the configuration file "data_oasis3/namcouple".
   - Execute "./run_test_interpolation".
   - Keep your results in /work_test_interpolation_E3_mono
   - ➢ Analyse the quality of the interpolation by plotting the interpolated field and the error of interpolation.

III – Compilation and running of Oasis3-mct and Tutorial on the remote computer of the participants

Before interfacing the real models with OASIS3-MCT, it is necessary to verify that the coupler and the toy tutorial compile and run on the remote computers of the participants.

1. Send OASIS3-MCT to the participant's remote computer:
   - Tar and zip the repository oasis3-mct:
   - ➢ cd $(HOME) ; "tar -cvf oasis3-mct.tar oasis3-mct" ; "gzip oasis3-mct.tar"
   - Send the file to your remote computer using "ssh":
   - ➢ scp oasis3-mct.tar $user@remote_computer:/remote_directory

2. Compile the OASIS3-MCT coupler:
   - Untar and uzip oasis3-mct.tar.gz
   - ➢ "gunzip oasis3-mct.tar.gz" ; "tar -xvf oasis3-mct.tar"
   - Go into directory "oasis3-mct/util/make_dir". Create a header makefile make.myplatform to compile OASIS3-MCT on your machine. Adapt also make.inc.
   - Type "make realclean –f TopMakefileOasis3" and then "make –f TopMakefileOasis3"

3. Compile and run tutorial :

- Go into "oasis3-mct/examples/tutorial"
- Type "make clean" and then "make" to compile the toy.
- Adapt the script "run_tutorial" to run on your remote platform. Execute "./run_tutorial".
- Keep your results in /work_tutorial_C

IV - Interfacing of OASIS3-MCT in participant's real models

The models and OASIS3-MCT must be compiled with the same compilers on the same platform.

V - Compilation and running of the coupled models