New Buildbot test suite for the OASIS3-MCT coupler Fortran source code

23/02/2021

Coquart L., Valcke S., Craig A., Maisonnave E.

CECI, Université de Toulouse, CNRS, CERFACS, Toulouse, France,

TR-CMGC-21-36

The work reported in this document has been done in the framework of the ISENES3 project that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 824084.

Table des matières

1 Introduction

Buildbot (Coquart, L. et al. (2017)) is software used to automatically compile and run the OASIS3-MCT source code (Craig, A. et al. (2017)) when a development is committed on the Git master version of the sources or on an active Git branch. Toy models are used to see if the development introduces any bugs in the source code. To do so the results of the toy models running after the development are compared automatically to the results of a reference state that was verified earlier. The use of Buildbot allows users to detect some problems and solve them before other merging to the master branch. Toy models do not contain any physics. They consist usually of two coupled models, model1 and model2, reproducing some functionalities of the coupler or reproducing algorithms of real coupled models.

The old Buildbot tests suite was developed between 2012 and 2019 using toys testing multiple functionalities and toys from users. The toys were added continuously to the test suite when a development was done on the master of OASIS3-MCT or when a toy was created to reproduce a bug reported by a user. Prior to the implementation of the new Buildbot tool, it was difficult to know what was exactly tested with each toy.

This is the reason why a new Buildbot tests suite was constructed in 2019-2020 based on the **functional toy concept**. Each toy now tests one functionality of the OASIS3-MCT coupler, with one ping (a coupling field is sent in one direction only) or one ping-pong (a coupling field is sent back-and-forth) exchange between model1 and model2 in most of the cases. At any given Git revision of the master all the functionalities of the coupler should be tested. This is why a functional toy is added for each new development included in OASIS3-MCT sources to test and validate the new functionality.

The current tests suite only works with sources of OASIS3-MCT posterior to the release OASIS3-MCT_4.0 (Valcke, S. et al. (2018)).

Compared to what is described in the document (Coquart, L. et al (2017)) where Buildbot 0.8.8 was used, we installed the new version 2.1.0 of Buildbot on fundy, a Linux Fedora Core 26 server at Cerfacs, to perform the new tests.

In the following document, we first present the environment created to construct the new Buildbot tests suite, then we define the general options of the tests suite and we describe quickly each toy: grids used, fields exchanged, mapping and all the tests performed with each toy available in the tests suite as of December 2020. We also describe the comparison of the results to the reference state using Buildbot before we conclude.

2 Environment for the new Buildbot tests suite

Since March 2019, OASIS3-MCT is hosted on the private Cerfacs Gitlab Nitrox server, available to few developers. The sources of the last release OASIS3-MCT_4.0 can be downloaded by the users from the public gitlab.com web site (see How to download the OASIS3-MCT sources on the OASIS web site).

On the private Cerfacs Github Nitrox server, the OASIS3-MCT project is organized in four distinct repositories:
* **OASIS3-MCT/oasis3-mct**: containing the sources of the coupler
* **OASIS3-MCT/oasis3-mct_tests**: containing the Buildbot toys
* **OASIS3-MCT/oasis3-mct_examples-users**: containing some toys to reproduce some user coupled application
* **OASIS3-MCT/oasis3-mct_other**: containing all other environments around OASIS3-MCT that need to be versioned

2.1 Creation of a new buildbot_tests_since_2019 branch in oasis3-mct_tests

The master of oasis3-mct_tests contains all the toys used in the old Buildbot tests suite from 2012 to October 2019. A new branch **buildbot_tests_since_2019** was created from the master to store all the new functional toys created since October 2019 (see Annexe A for more details on the creation of this new branch).

All the scripts written for the tests are contained in the **bench_buildbot_since_2019 directory** of the new **buildbot_tests_since_2019** branch.

The python file **master.cfg** of Buildbot is launched on fundy, where the version 2.1.0 of Buildbot is installed, in /space/coquart/Buildbot_since_2019/oasis3-mct_master (if testing the master) or in /space/coquart/Buildbot_since_2019/oasis3-mct_branch (if testing a new branch).
When OASIS3-MCT sources change on the master or on a branch of Nitrox Git server, the python file **master.cfg** automatically clones these sources and the Buildbot toys, sends them to the tested computers, compiles the sources on the different computers and then launches the tests one after the other. These instructions use the scripts developed and stored in the directory bench_buildbot_since_2019.

Usually, the developments are completed for a given release and the number of toys testing all the functionalities of the release of the coupler stay stable in time.
For the next OASIS3-MCT_5.0 release (Valcke, S. et al. (2020)), planned in December 2021, a new directory will be created **bench_buildbot_5.0** (starting with the content of **bench_buildbot_since_2019**) for the tests done on OASIS3-MCT_5.0 and **bench_buildbot_since_2019** will be kept for the tests done on the master of OASIS3-MCT. It is not planned to create a new branch unless we create another new Buildbot tests suite.

The toy_NLOGPRT was the first toy created for the new Buildbot tests suite. All the other toys presented below were added between October 2019 and December 2020.

3 General options of the new tests suite

3.1 Organization of the data

The resolution of the grids used in the toys is higher than in the previous ones. This is why we decided to store those grids at the same location for all the toys of the new suite.

The files grids.nc, masks.nc and areas.nc are stored in the common directory ***oasis-mct_tests/bench_buildbot_since_2019/common_data_oasis3***.
The remapping files used in the toys are all stored in the common directory ***oasis-mct_tests/bench_buildbot_since_2019/common_rmp_files***. It is in the running script run_buildbot that the remapping files are linked in the working directory corresponding to the toy.

3.2 Grids

In our new toys, we use the following global grids (Coquart, L. et al. (2019)):

* **lmdz**: 96x72x1 points, IPLS/LMD structured logically rectangular low-resolution grid
* **bggd**: 144x143x1 points, IPLS/LMD structured logically rectangular medium resolution grid
* **torc**: 182x149x1 points, IPSL/LOCEAN NEMO ORCA2 structured logically rectangular low-resolution grid
* **nogt**: 294x362x1 points, IPSL/LOCEAN NEMO ORCA1 structured logically rectangular medium resolution grid
* **icos**: 15212x1 points, IPSL/LMD unstructured icosahedral medium resolution grid
* **ssea**: 24572x1 points, Météo-France/CNRM unstructured gaussian reduced medium resolution grid

We also use two structured logically rectangular regional grids:

* **atmt**: 110x110x1 points, atmosphere regional grid
* **toyt**: 110x110x1 points, ocean regional grid

3.3 Toys

Most of the toy models consists of two coupled models model1 and model2, except toy_intracomm consisting in 3 executables model1, model2 and model3. There are also some toys with only one executable model2. We will only describe here the global characteristics of the coupling of model1 and model2.

In most of the cases, model1 is defined on the nogt grid and model2 is defined on the bggd grid.

Except for the toy_interpolation that explicitly calculates the remapping files with the hybrid OpenMP + MPI SCRIP library included in OASIS3-MCT since version 4.0 (Piacentini, A. et al. (2018)), most of the toys use remapping files pre-calculated and copied in the working directory at the beginning of the run.

Model1 and model2 usually runs 21600 seconds, their coupling period is equal to 7200 seconds and their time step is equal to 3600 seconds.

For all the toys NLOGRT is “1 0” in the OASIS3-MCT configuration file namcouple, except for toy_NLOGPRT (which tests different values of NLOGPRT), toy_mapcons (where NLOGPRT is “20 0”) to be able to get the results from the debug files) and toy_load_balancing (where NLOGPRT is “1 -2” to output load imbalance diagnostics, see Maisonnave, E. et al. (2020)).

The options CHECKIN/CHECKOUT are activated in the namcouple for all the toys. These options calculate the global minimum, the maximum and the sum of the source field/target field. It is a very simple first way to compare the results and validate them using Buildbot.

If the characteristics of a toy are different from what is described above, it will be noted when listing the toy below.

3.4 Decomposition

If the toy runs in parallel, all processes of the toys read their local grid corresponding to their local partition.

In parallel, we run on maximum 7 processes (see section 5 for the platform description), except for toy_interpolation where we also test the hybrid OpenMP+MPI parallelization:

- In monoprocessor, toy_interpolation runs on kraken and belenos on 1 node with 1 processor for each model.
- In parallel on kraken and belenos, toy_interpolation runs on 1 node with 3 processors for model1 and 4 processors for model2, and also on 2 nodes, each model running on 1 node on 1 MPI task with 1 thread or 10 threads.

As already said, in most of the cases, model1 is defined on the nogt grid with a BOX decomposition such that each partition is rectangular and covers all longitudes. Model2 is defined on the bggd grid with an APPLE decomposition: each partition is described as a segment (that in fact matches the BOX rectangular decomposition).

For toy_interpolation, to avoid the multiplication of the tests, the decomposition is also the APPLE one (matching the BOX rectangular one) for both grids for all couples of grids.

The toy model toy_1f1grd_to_2f2gr tests other configurations as described in section 4.11.

3.5 Coupling fields

The coupling fields correspond to a sinusoidal analytical function that depends on the time and the lat-lon coordinates of the model grid, except for toy_mapcons, toy_scalar_coupling and toy_CHECKIN_BLASOLD_BLASNEW_CHECKOUT.

For toy_interpolation, an error of interpolation is calculated (Coquart, L. et al. (2019)).

As already specified, model1 and model2 of the coupled toy models usually perform one ping or one ping-pong of a single coupling field.

4 Listing of the toys with their tests

The different sections of the documentation we refer to below **correspond to the User Guide of the master branch of OASIS3-MCT revision 4467c418**.

4.1 toy_NLOGPRT

Tests the output written in debug files as the function of NLOGPRT value in the namcouple. Tests also the output written in the timer files (see User Guide section 3.2).

One ping-pong between nogt(BOX)-bggd(APPLE), MAPPING/BILINEAR (src bfb).
Only parallel MPI tests.

** NLOGPRT values
- 1 0
- 1 1
- 30 3

4.2 toy_grids_writing

Tests the grid writing routines (see User Guide section 2.2.4).

No coupling fields exchanged. Each model reads its own grid and writes grids.nc, masks.nc and areas.nc.

** Parallelization:
- mono
- para MPI

4.3 toy_mapcons

Tests the CONSERV (Craig, A. (2019)) global options (see User Guide section 4.4).

NLOGPRT is “20 0” to get some coefficients used in the conservation calculation from the debug files for Buildbot.
One ping-pong between nogt(BOX)-bggd(APPLE), MAPPING/BILINEAR (src bfb).
Only parallel MPI tests.

** CONSERV options
- GLOBAL
- GLBPOS
- BASBAL
- BASPOS
- GSSPOS

- BSSPOS

4.4 toy_scalar_coupling

Tests exchanges of scalar values (see User Guide section 2.3.2).

One ping-pong between nogt(BOX)-bggd(APPLE), MAPPING/BILINEAR (src bfb).
10 scalars are sent with a coupling period of 3600 seconds from nogt to bggd, no interpolation.
Only parallel MPI tests.

4.5 toy_interpolation

Tests the remapping of the SCRIP library (see User Guide section 4.3).

This toy was created from the toy located in the examples of the coupler made available to users, ***oasis3-mct/examples/test-interpolation***.

There is a special script to run this toy as the name of the grids and of the remapping are directly read from the namcouple to test as many couple of grids as possible automatically.

One ping from source grid(APPLE) to target grid(APPLE).

** Couple of grids and associated remappings:
- bggd-nogt + nogt-bggd (gauswgt, distwgt, bicu, bili, conserv1st, conserv2nd)
- ssea-nogt (gauswgt, distwgt, bicu, bili, conserv1st) + nogt-ssea (gauswgt, distwgt, bicu, bili, conserv1st, conserv2nd)
- icos-nogt (gauswgt, distwgt, conserv1st) + nogt-icos (gauswgt, distwgt, bicu, bili, conserv1st, conserv2nd)
** Parallelization:
- mono
- para MPI
- para MPI+OpenMP (only on kraken and belenos)
** Interpolations (when possible):
- SCRIPR/DISTWGT
- SCRIPR/GAUSWGT
- SCRIPR/BILINEAR
- SCRIPR/BICUBIC
- SCRIPR/CONSERV (FRACAREA, FRACNNEI, FRACNNTR)
- SCRIPR/CONSERV2D (FRACAREA, FRACNNEI, FRACNNTR)

4.6 toy_CHECKIN_BLASOLD_BLASNEW_CHECKOUT

Tests the pre-processing and post-processing transformations BLASOLD and BLASNEW (see User Guide sections 4.2 and 4.4).

One ping from nogt(BOX) to bggd(APPLE), MAPPING/BILINEAR (src bfb).

** Parallelisation:
- mono
- para MPI
** Transformations (in addition to the MAPPING/BILINEAR):
- CHECKIN
- CHECKIN + BLASOLD
- CHECKIN + BLASOLD + BLASNEW
- CHECKIN + BLASOLD + BLASNEW + CHECKOUT

4.7 toy_MAPPING_options

Tests all combinations of the mapping options MAPLOC, MAPSTRATEGY, NMAPDEC (see User Guide section 4.3).

One ping from nogt(BOX) to bggd(APPLE), MAPPING/BILINEAR.
Only parallel MPI tests.
$NMATXRD=orig (indicates the method used to read the remapping file)
$NUNITNO=100,150

** MAPLOC:
- src
- dst
** MAPSTRATEGY:
- bfb
- sum
- opt
** NMAPDEC:
- decomp_wghtfile
- decomp_1d

4.8 toy_NMATXRD_options

Tests the NMATXRD options, which corresponds to the way the remapping files are read (see User Guide section 3.2)

One ping from nogt(BOX) to bggd(APPLE), BILINEAR/MAPPING.
Only parallel MPI tests.
$MAPLOC=src, $MAPSTRATEGY=bfb, $NMAPDEC=decomp_1d. XXX

** NMATXRD
- orig
- ceg

4.9 toy_mixed_SP_DP

Tests the exchanges between model1 compiled in single precision and model2 compiled in double precision.
One ping-pong nogt(BOX)-bggd(APPLE), MAPPING/BILINEAR (src bfb).
Only monoprocessor tests.

4.10 toy_identical_grids

Tests the exchanges between identical grids in model1 and model2.

One ping from bggd(APPLE) to bggd(APPLE), no remapping.
Only monoprocessor tests.

4.11 toy_1f1grd_to_2f2gr

Tests one source field defined on one grid sent to two target fields defined on two different target grids.

Multiple pings from nogt of model1 to lmdz and bggd of model2, MAPPING/BILINEAR (src bfb).

* nogt(BOX) in model1 – lmdz(APPLE) and bggd(APPLE) in model2
** Parallelization:
- mono
- para MPI

* nogt(APPLE) in model1 – lmdz(BOX) and bggd(BOX) in model2
** Parallelization:
- mono
- para MPI

4.12 toy_auxiliary_routines

Tests all the auxiliary routines (see User Guide section 2.2.9).

One ping nogt(BOX)-bggd(APPLE), MAPPING/BILINEAR (src bfb).
Only monoprocessor tests.

** Routines tested:
- oasis_abort
- oasis_get_debug
- oasis_set_debug
- oasis_get_intercomm
- oasis_get_intracomm
- oasis_put_inquire

- oasis_write_restart
- oasis_get_ncpl, oasis_get_freqs

4.13 toy_multiple_fields_one_communication

Tests multiple coupling fields sent via a single communication (see User Guide section 2.2.7).

Multiple pings from nogt(BOX) to bgg(APPLE).

** Parallelization:
- mono
- para MPI

** Coupling fields and transformations:
- one multiple coupling field (2 grouped fields), (MAPPING/BILINEAR (src bfb) + ACCUMUL) + one single coupling field (SCRIPR/BILINEAR+AVERAGE), without LAG
- one multiple coupling field (2 grouped fields), (MAPPING/BILINEAR (src bfb) + ACCUMUL) + one single coupling field (SCRIPR/BILINEAR+AVERAGE), with LAG (with all the coupling fields in the same restart)

4.14 toy_time_transformations

Tests the time transformations (see User Guide section 4.1).

One ping from nogt(BOX) to bggd(APPLE), MAPPING/BILINEAR (src bfb).
Only monoprocessor tests.
The time step of the models is equal to 3600 seconds.
The runtime is equal to 57600 seconds with a coupling period equal to 14400 seconds.

** Transformations:
- ACCUMUL
- AVERAGE
- T_MIN
- T_MAX

** Time simulated:
- 0 to (57600-3600)

** LAG tests:
- without LAG
- with LAG

4.15 toy_restart_ACCUMUL_1_NOLAG

Tests the LOCTRANS restart for the ACCUMUL time transformations without LAG (doc section 2.3.3).

One ping from nogt(BOX) to bggd(APPLE), MAPPING/BILINEAR (src bfb), ACCUMUL.

Only monoprocessor tests.
Without LAG.
The time step of the models is equal to 3600 seconds.
The runtime is equal to 43200 seconds with a coupling period equal to 14400 seconds.

** Time simulated:
- 0 to (43200-3600)

4.16 toy_restart_ACCUMUL_1_LAG

Tests the LOCTRANS restart for the ACCUMUL time transformation with LAG (see User Guide section 2.3.3).

One ping from nogt(BOX) to bggd(APPLE), MAPPING/BILINEAR (src bfb), ACCUMUL.
Only monoprocessor tests.
With LAG.
The time step of the models is equal to 3600 seconds.
The runtime is equal to 43200 seconds with a coupling period equal to 14400 seconds.

** Time simulated:
- 0 to (43200-3600)

4.17 toy_restart_ACCUMUL_2_NOLAG

Tests the LOCTRANS restart for the ACCUMUL time transformation without LAG using the LOCTRANS restart file of toy_restart_ACCUMUL_1_NOLAG (see User Guide section 2.3.3).

One ping from nogt(BOX) to bggd(APPLE), MAPPING/BILINEAR (src bfb).
Only monoprocessor tests.
Without LAG.

** Time simulated:
- 0 to (14400-3600)

4.18 toy_restart_ACCUMUL_2_LAG

Tests the LOCTRANS restart for the ACCUMUL time transformation with LAG using the restart file of toy_restart_ACCUMUL_1_LAG (see User Guide section 2.3.3).

One ping from nogt(BOX) to bggd(APPLE), MAPPING/BILINEAR (src bfb).
Only monoprocessor tests.
Without LAG.

** Time simulated:
- 0 to (14400-3600)

4.19 toy_bundle

Tests the exchanges of one 3D bundle coupling field (see User Guide 2.2.7).

One ping-pong between nogt(BOX)-bggd(APPLE), MAPPING/BILINEAR (src bfb).

** Parallelization:
- mono
- para MPI
** Coupling fields:
- one bundle with 5 levels in each model

4.20 toy_configuration_components_A

Tests the configuration A described in the picture 2.1 of the OASIS3-MCT documentation, see Appendix B below.

Multiple ping pong between nogt(BOX)-bggd(APPLE), MAPPING/BILINEAR (src bfb).
Only parallel MPI tests.

** Configuration:
- exchange A: 2 executables, model1 and model2, running concurrently on separate sets of MPI tasks. Model 1 has one component comp1m1 defined on nogt (proc 0-1). Model2 has two components comp2m2, defined on bggd (proc 2-6), and comp4m2 (proc 7). Comp4m2 does not couple
** Coupling fields:
- one bundle field (5 levels) + one multiple field (2 grouped fields) + 1 single field, in both models

4.21 toy_configuration_components_B

Tests the configuration B described in the picture 2.1 of the OASIS3-MCT documentation, see Appendix B below.

This toy contains only one executable model2.
Multiple ping pong between bggd(APPLE)-lmdz(APPLE), MAPPING/BILINEAR (src bfb).
Only parallel MPI tests.

** Configuration:
- exchange B: 2 concurrently components in model2, comp2m2 defined on bggd(APPLE) (proc 0-1) and comp3m2 defined on lmdz(APPLE) (proc 2-3). Comp4m2 (proc 4) does not couple
** Coupling fields:
- one bundle field (5 levels) + one multiple field (2 grouped fields) + 1 single field, on each grid bggd and lmdz

4.22 toy_configuration_components_C

Tests the configuration C described in the picture 2.1 of the OASIS3-MCT documentation, see Appendix B below.

This toy contains only one executable model2.
Multiple ping pong between lmdz(APPLE)-icos(SERIAL), MAPPING/DISTWGT (src bfb).
Only parallel MPI tests.

* Configuration:
- exchange C: 3 components in one executable: comp2m2, comp3m2 and comp4m2. Comp2m2, defined on bggd (proc 0-1), and comp4m2 (proc 5) do not couple. Component comp3m2 has two concurrent sub-components, one running on lmdz (proc 2-3) and one running on icos (proc 4)
* Coupling fields:
- one bundle field (5 levels) + one multiple field (2 grouped fields) + 1 single field, on each grid lmdz and icos

4.23 toy_configuration_components_CGH

Tests configurations C, G, and H described in the picture 2.1 of the OASIS3-MCT documentation, see Appendix B below.

This toy contains only one executable model2.
Only parallel MPI tests.

* Configuration:
- Exchange C, G, H: 3 components in one executable: comp2m2, comp3m2, comp4m2. Comp2m2, defined on bggd (APPLE) (proc 0-1) and comp4m2 (proc 5) do not couple. Component comp3m2 has two concurrent sub-components, one running on lmdz(APPLE) (proc 2-3) and one running on icos(SERIAL) (proc 4). It has also one sub-component running sequentially on ssea(APPLE) (proc 2-4), compared to the two other ones
* Coupling fields:
- one ping pong between lmdz(APPLE) and icos(SERIAL), MAPPING/DISTWGT (src bfb)
- multiple ping pong between lmdz(APPLE) and ssea(APPLE) of one bundle field (5 levels) + one multiple field (2 grouped fields) + 1 single, MAPPING/BILINEAR (src bfb)
- one ping pong between icos(SERIAL) and ssea(APPLE), MAPPING/DISTWGT (src bfb)

4.24 toy_configuration_components_ABCGH

Tests configurations A, B, C, G, and H described in the picture 2.1 of the OASIS3-MCT documentation, see Appendix B below.

This toy contains two executables.
Only parallel MPI tests.

** Configuration:

- Exchange A, B, C, G, H: 2 executables model1 and model2, running concurrently on separate sets of MPI tasks. Model 1 has one component comp1m1 defined on nogt (proc 0-1). Model2 has three components comp2m2, defined on bggd (proc 2-3), comp3m2 defined on lmdz (proc 4-5), icos (proc 6), ssea (proc 4-6), and comp4m2 (proc 7). Comp4m2 does not couple
* * Coupling fields:
- ***model1-comp1m1 to model2-comp2m2***: multiple ping pong between nogt(BOX) and bggd(APPLE) of one bundle field (5 levels) + one multiple field (2 grouped fields) + 1 single coupling field, MAPPING/BILINEAR (src bfb)
- model2-comp2m2 et model2-comp2m3: one ping pong from bggd(APPLE) to lmdz(APPLE), MAPPING/BILINEAR (src bfb)
- model2-comp2m3: one ping pong between lmdz(APPLE) and icos(SERIAL), MAPPING/DISTWGT (src bfb)
- ***model2-comp2m3***: multiple ping pong between lmdz(APPLE) and ssea(APPLE) of one bundle field (5 levels) + one multiple field (2 grouped fields) + 1 single coupling field, MAPPING/BILINEAR (src bfb)
- ***model2-comp2m3***: one ping pong between icos(SERIAL) and ssea(APPLE), MAPPING/DISTWGT (src bfb)

4.25 toy_load_balancing

This toy tests the new diagnostic of load imbalance in coupled models (see Maisonnave, E. et al. (2020)).

One ping pong between nogt(BOX)-nogt(APPLE),MAPPING/BILINEAR (src bfb).
Only parallel MPI tests.

** NLOGPRT:
- (1, -2)

4.26 toy_gaussianreducedgrid

Tests the APPLE decomposition by entire number of latitudes for a gaussian reduced grid (see User Guide section 2.2.3).

One ping pong between nogt(BOX)-ssea(APPLE), MAPPING/BILINEAR (src bfb).

** Parallelization:
- mono
- para MPI

4.27 toy_grids_regional_to_regional

Tests one ping pong between two regional structured grids.

One ping pong between toyt(BOX)-atmt(BOX), MAPPING/DISTWGT (src bfb).

** Parallelization:
- mono
- para MPI

4.28 toy_create_couplcomm

Tests the possibility of coupling a sub-component only over a subset of processes, see picture 2.2 of the OASIS3-MCT documentation, see Appendix B below.

Toy created from toy_configuration_components_C, adding two processors not coupling to comp3m2.

4.29 toy_NTHRESH_STHRESH

Tests the reading of NTHRESH and STHRESH in the namcouple for the SCRIP/CONSERV remapping (see User Guide section 4.3).
NTHRESH/STHRESH is the value of the northern/southern latitude threshold in radians where conservative area computation switches from linear boundaries in longitude and latitude at the equator to a Lambert equivalent azimuthal projection toward the north/south pole.

One ping from nogt(BOX) to bggd(APPLE)

** Parallelization:
- mono
- para MPI
** Interpolation:
- SCRIPR/CONSERV (default in SCRIP: NTHRESH=2.0   STHRESH=-2.0; no Lambert projection)
- SCRIPR/CONSERV: NTHRESH=1.45   STHRESH=-1.45

4.30 toy_intracomm

Tests oasis_get_intercomm, oasis_get_intracomm and new oasis_get_multi_intracomm routines (see User Guide 2.2.9).

3 executables: model1 (nogt), model2 (bggd), model3 (bggd).
One ping from nogt(BOX) to bggd(APPLE), MAPPING/BILINEAR (src bfb).
One ping from bggd(APPLE) to bggd(APPLE), no interpolation.

* Parallelization:
- mono
- paraMPI

5 Computing platforms

At the time of writing, the new Buildbot tests suite runs on fundy, stiff, kraken and belenos described below:

* Fundy is a Linux Fedora Core 26 computer of Cerfacs with 4 processors, and the Buildbot tests run using gfortran (based on gcc version 7.3.1 20180130) + openmpi (mpirun version 2.0.2). At the moment, there are 222 tests done on this computer.
* Stiff is also a Linux Fedora Core 26 computer of Cerfacs with 4 processors, and the Buildbot tests run using pgi (version 18.7) + openmpi (mpirun version 2.1.2). At the moment, there are 222 tests done on this computer.
* Kraken is a Lenovo super-computer at Cerfacs and the Buildbot tests run using intel (version 18.0.1.163) + intelmpi (version 2018.1.163). One node contains 36 cores. At the moment, there are 326 tests done on this computer.
* Belenos is an AMD super-computer at Météo-France (newer than kraken) and the Buildbot tests run using intel (version 2018.5.274) + intelmpi (version 2018.5.274). One node contains 128 cores. At the moment, there are 326 tests done on this computer.

The number of tests is different from a machine to another as we run the toy models with different levels of parallelization on the different platforms; as already detailed above:

- On stiff and fundy, we only run in monoprocess or with multiple processors (7), even for toy_interpolation.
- On kraken and belenos, we perform tests on 1 node in monoprocess or in parallel (7 processors) and we also perform tests on 2 nodes, each model running on 1 node using 1 MPI task and either 1 thread or 10 threads, to test the hybrid parallelization OpenMP+MPI for toy_interpolation.

6 Results

6.1 Results of the tests for Buildbot

Each test is running in a directory defined using the name of the toy, the namcouple tested, the Makefile tested, the number of nodes and threads (equals to 1 except for toy_interpolation on kraken and belenos), “model1” with the number of processors used to run it, “model2” with the number of processors used to run it and “model3 with the number of processors used to run it. It is possible to run a maximum of 3 executables. If one or two executables are not used their number of processors is equal to 0.

All the parameters of each test are described in a file named arbitrarily case1.txt to case999.txt, for monoprocessor tests, and in a file named case10001.txt to case10999.txt, for the corresponding parallel tests. For toy_interpolation the configuration files testing the OpenMP+MPI parallelization are named interp55.txt to interp106.txt for the 1 thread case and interp10055.txt to interp.10106.txt for the 10 threads case.

In each file, the name of the toy, the namcouple used, the Makefile used, the number of processes for each executable model1, model2 and model3 (or the number of nodes, number of MPI tasks and number of threads for toy_interpolation) are specified. Then there is the number of fields to be verified by Buildbot followed by their NetCDF debug file’s name.

On the contrary of what was done in the old Buildbot suite, there is only one script to launch to run the toy, create the output text files and verify the results with Buildbot, independently of the number of executables (one, two or three). Some special scripts were written for toy_interpolation as the name of the couple of grids are read in the namcouple as well as the remapping under testing. There is also the script to manage error detection after the run.

All these files are located in ***oasis-mct_tests/bench_buildbot_since_2019*** of the new ***buildbot_tests_since_2019*** branch.

As for the old Buildbot tests suite, a reference state is first created and verified. In each reference repository of each test validated, there are multiple ***text files*** created from the output files of the OASIS3-MCT coupler (nout file, debug files printed thanks to NLOPRT=1, debug coupling files, restarts if LAG>0 and/or if LOCTRANS operations) and the model outputs, using ***the script output_files_oa3-mct_buildbot***. The script output_files_oas3-mct_buildbot uses principally grep and ncdump commands.

The text files created at the end of each run allow Buildbot to validate the new development.   One text file validates that the job ran successfully. One text file validates the min IO unit number and the max IO unit number read in the nout.000000 file. One text file validates the diagnostics on the coupling fields written in the debug files thanks to the use of the CHECKIN and CHECKOUT options in the namcouple. One text file validates the minima and maxima of the coupling fields sent and received in the toy models. All the debug coupling fields, stored in NetCDF files, and described in the configuration files case*.txt or interp*.txt, are stored into a text file thanks to a ncdump command to be able to verify their values afterwards. If the toy runs in monoprocessor and in parallel, one text file is created for the parallel case to validate the parallelization by comparison to the monoprocessor case. Some special verifications are done for toy_grids_writing, as we must verify that the grids read by each model are the same that the one written in areas.nc, masks.nc and grids.nc. It is also the case for the toys toy_restart_ACCUMUL* where the output coupling fields, the LOCTRANS restart (LAG=0), or the restart (LAG>0) fields must be compared to the ones obtained with toy_time_transformations for LOCTRANS=ACCUMUL. For toy_load_balancing we verify the debug files and if the timer files were created but we do not verify the consistency of the load balancing results as we run toy models.

Every time a text file is created for a verification with Buildbot, a ***text diff file*** is also created with ***the script output_files_oa3-mct_buildbot***, comparing the actual result of the run to the reference state. When creating the first reference state, the text diff files have a size equals to 0 but as all the results are validated by hand in this case. When adding new toys corresponding to new developments, the toys are first validated by hand before integrating the Buildbot tests suite.

Buildbot (installed on fundy) listens to the GIT server and when a modification is done in the master or a branch, the tests are launched on fundy, stiff, kraken and belenos. The extraction of the sources of the OASIS3-MCT coupler and the Buildbot toys are done automatically as well as the compilation of OASIS3-MCT on the different computers. Then all the tests are run one after the other. In the analysis repository, all the text files described above are created at the end of each test and the verification of the results is done by checking **the size of the text diff files** also created at the end of each test.

If the size of the diff files is equal to 0, which means that there is no difference between the current tests and the reference state, the output is green in the graphical Buildbot interface; otherwise the output is marked red, the tests stop, and it is necessary to open the corresponding text diff file with a size different from 0 to see where the differences come from (characters, numbers), see Appendix C below.

7 Conclusions

This report presents the development of a new Buildbot tests suite for the OASIS3-MCT coupler Fortran sources, between October 2019 and December 2020. Compared to the old one (Coquart, L. et al. (2017)), each toy model tests now one functionality and the results are easier to interpret and errors easier to catch.

After describing the new environment created for the new tests and the general options of the Buildbot toys, we described quickly the toys and the different tests they perform.

Buildbot now runs on fundy, stiff, kraken and belenos. The Buildbot tool has helped to find a number of bugs during the development of the OASIS3-MCT coupler.

Some toys are still missing (for example toy_icos_parallel to test the decomposition of an icosahedrical grid or the toy SPOC created to performed online trainings for users) and will be added soon. Other toys will be added if needed to test and validate new developments in the master until the release OASIS3-MCT_5.0 in December 2021.

There is the development of a python interface in OASIS3-MCT at the moment. Some toys were created to test the python API, as for the Fortran sources described in this document. The toys created for the pyoasis part of OASIS3-MCT should be added to the new Buildbot tests suite in 2021.

8 Bibliography

Valcke, S., Craig, A. and Coquart, L. (2020), OASIS3-MCT development plan, CECI, Université de Toulouse, CNRS, CERFACS, Toulouse, France - TR-CMGC-20-164, Technical report

Maisonnave, E., Coquart, L. and Piacentini, A. (2020), A better diagnostic of the load imbalance in OASIS based coupled systems, CECI, Université de Toulouse, CNRS, CERFACS, Toulouse, France - TR-CMGC-20-176, Technical report

Craig, A. (2019) 9, GSSPOS and BSSPOS options for the global conservation in OASIS3-MCT, CECI, Université de Toulouse, CNRS, CERFACS, Toulouse, France - TR-CMGC-19-128, Technical report

Coquart, L. and Valcke, S. (2019) ESMF v7.1.0r remapping between two structured grids and between two unstructured grids and one structured grid, CECI, Université de Toulouse, CNRS, CERFACS, Technical report

Valcke, S., Craig, A. and Coquart, L. (2018) OASIS3-MCT User Guide, OASIS3-MCT4.0, CECI, Université de Toulouse, CNRS, CERFACS - TR-CMGC-18-77, Technical report

Piacentini, A., Maisonnave, E., Jonville, G., Coquart, L. and Valcke, S. (2018), A parallel SCRIP interpolation library for OASIS, CECI, Université de Toulouse, CNRS, CERFACS - WN-CMGC-18-34, Toulouse, France , Working note

Craig, A., Valcke, S. and Coquart, L. (2017) Development and performance of a new version of the OASIS coupler, OASIS3-MCT 3.0, Geoscientific Model Development, 10, pp. 3297-3308, doi: 10.5194/gmd-10-3297-2017

Coquart, L., d'Ast, I. and Valcke, S. (2017) Buildbot : Le logiciel utilisé pour compiler et tester automatiquement les développements réalisés dans le coupleur OASIS3-MCT, UMR 5318 CECI, CERFACS/CNRS, TR-CMGC-17-85, Technical report

9 Appendix A

Creation of the new buildbot_tests_since_2019 branch on Tioman, Linux Fedora Core 26 at Cerfacs computer

```
* cd /space/coquart
* git clone git@nitrox.cerfacs.fr:globc/OASIS3-MCT/oasis3-mct_tests.git
* cd /space/coquart/oasis3-mct_tests
* git checkout -b buildbot_tests_since_2019
* git push --set-upstream origin buildbot_tests_since_2019
        Total 0 (delta 0), reused 0 (delta 0)
        remote:
        remote: To create a merge request for buildbot_tests_since_2019, visit:
        remote:   https://nitrox.cerfacs.fr/globc/OASIS3-MCT/oasis3-
        mct_tests/merge_requests/new?merge_request%5Bsource_branch%5D=buildbot_te
        sts_since_2019
        remote:
        To nitrox.cerfacs.fr:globc/OASIS3-MCT/oasis3-mct_tests.git
         * [new branch]      buildbot_tests_since_2019 -> buildbot_tests_since_2019
        Branch buildbot_tests_since_2019 set up to track remote branch
        buildbot_tests_since_2019 from origin.
```

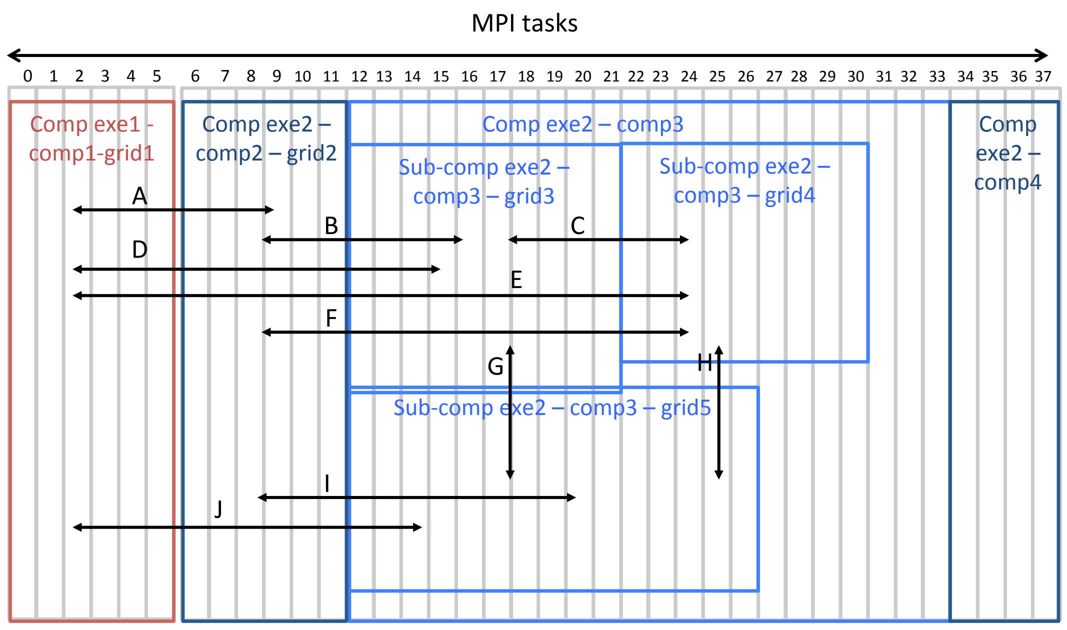
Updating of the new buildbot_tests_since_2019 branch on Tioman

```
* cd /space/coquart
* rm -rf oasis3-mct_tests
* git clone git@nitrox.cerfacs.fr:globc/OASIS3-MCT/oasis3-mct_tests.git
* cd /space/coquart/oasis3-mct_tests
* git checkout -b buildbot_tests_since_2019
* git rm -r bench_buildbot bench_buildbot_old_4.0 HR_tutorial mapchk maphot
no_pes_coupling pes_coupling pes_coupling_3.0 tc3a test_1bin_concurrent
test_1bin_ocnice test_1bin_sequential test_1fto2f_2models test_3D_remap_file
test_bundles test_eric_3bin_io_2grids test_gaussian_reduced test_grouped_fields
test_identical_grids test_prism_oasis testr4r8 test_simple_multiple_fields
test_simple_options test_smasson_zooms test_writing_grids toy_auxiliary_routines
toy_eric_echam_cosmo_cottbus toy_eric_pulsation toyhadgem3_UKC
* git gui

* mkdir bench_buildbot_since_2019
* mkdir toy_NLOGPRT
* git gui
```

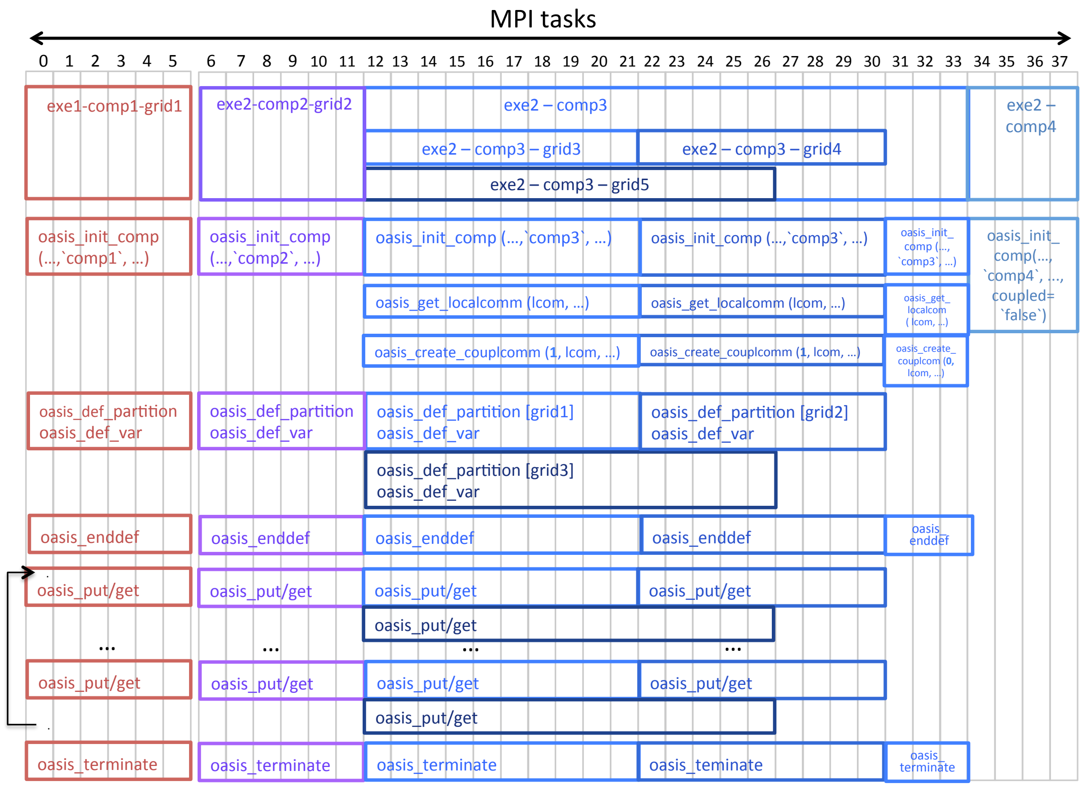
10 Appendix B

Picture 2.1 of the OASIS3-MCT documentation



The different configuration of components supported by OASIS3-MCT_3.0. Two executables `exe1` and `exe2` are running concurrently on separate sets of MPI tasks (0-5 for `exe1` and 6-37 for `exe2`). Executable `exe1` includes only one component `comp1` that has coupling fields defined on only one grid `grid1` (decomposed on all its 6 tasks). Executable `exe2` includes 3 components, `comp2`, `comp3`, and `comp4` running concurrently respectively on tasks 6-11, 12-33 and 34-37. Component `comp2` participates in the coupling with fields defined on only one coupling grid `grid2` (decomposed on all its 5 tasks) while `comp4` does not participate at all in the coupling. Component `comp3` has 3 sub-components, respectively exchanging coupling fields defined on `grid3` (tasks 12-21), `grid4` (tasks 22-30) and `grid5` (tasks 12-26, therefore overlaping with both `grid3` and `grid4`); finally, `comp3` has 3 tasks (31-33) not involved in the coupling. Sub-components exe2-comp3-grid3 and exe2-comp3-grid5, or sub-components exe2-comp3-grid4 and exe2-comp3-grid5 are examples of coupling between sub-components running sequentially on overlapping sets of tasks.

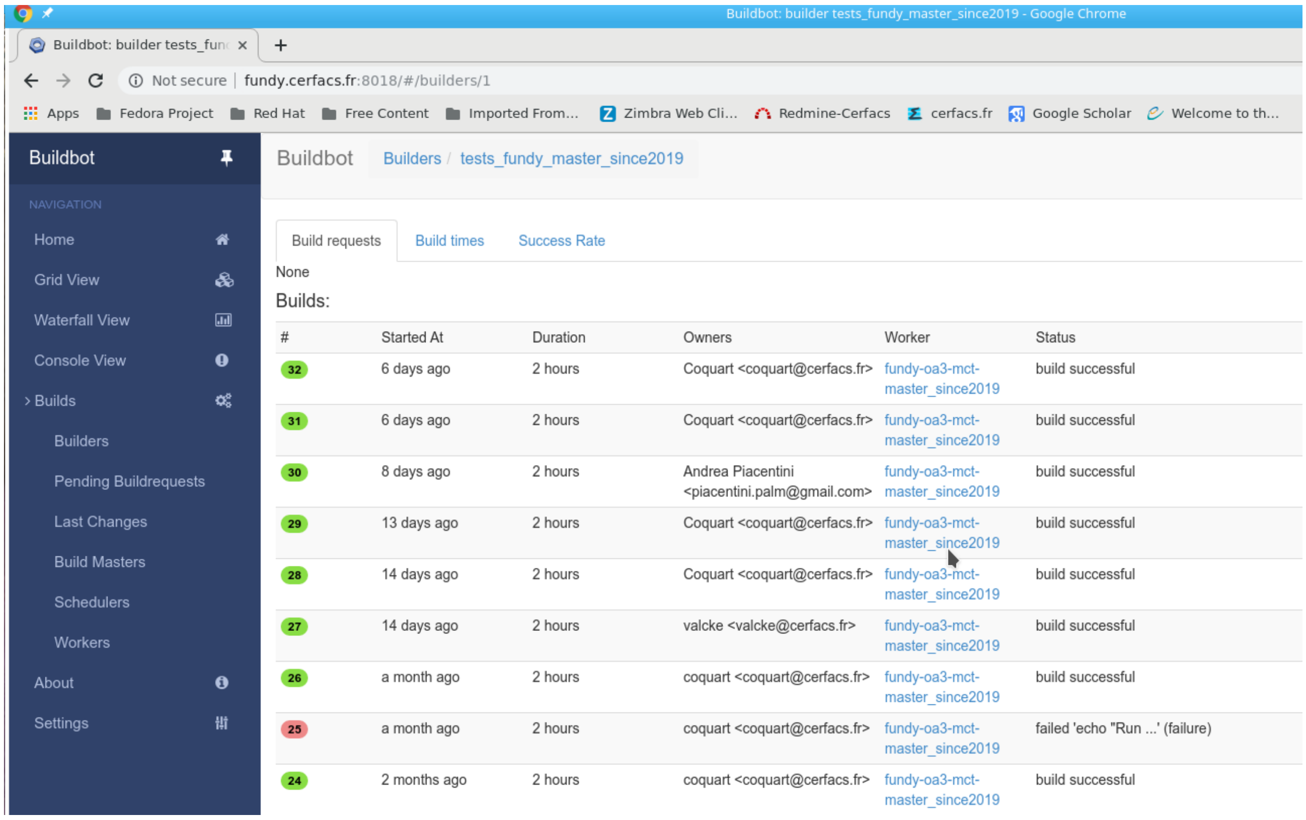
Picture 2.2 of the OASIS3-MCT documentation



The sequence of OASIS3-MCT calls that have to be implemented in the codes so to allow the configuration of components described on figure 2.1. Each MPI tasks has to call `oasis_init_comp` once with the name of its component as 2^{nd} argument. As none of `comp4` tasks is participating to the coupling, `comp4` tasks calls `oasis_init_comp` with `coupled=.false.` as 4^{th} argument and does not call any other OASIS3-MCT routine. As some of `comp3` tasks are participating to the coupling, all `comp3` tasks have to call `oasis_init_comp`, `oasis_get_localcomm`, `oasis_create_couplcomm`, `oasis_enddef` and `oasis_terminate` (these are the only routine to be called by `comp3` tasks 31-33 not participating to the coupling). To initialise the coupling exchanges, the tasks of a sub-component holding a field decomposed on a specific grid have to call the `oasis_def_partition` to express the decomposition of the grid, `oasis_def_var` to declare the coupling field and `oasis_enddef`. Finally, the tasks of a sub-component exchanging coupling fields have to call `oasis_put` and `oasis_get` accordingly.

11 Appendix C

Example of Buildbot results on fundy computer



None

Builds:

#	Started At	Duration	Owners	Worker	Status
32	6 days ago	2 hours	Coquart <coquart@cerfacs.fr>	fundy-oa3-mct-master_since2019	build successful
31	6 days ago	2 hours	Coquart <coquart@cerfacs.fr>	fundy-oa3-mct-master_since2019	build successful
30	8 days ago	2 hours	Andrea Piacentini <piacentini.palm@gmail.com>	fundy-oa3-mct-master_since2019	build successful
29	13 days ago	2 hours	Coquart <coquart@cerfacs.fr>	fundy-oa3-mct-master_since2019	build successful
28	14 days ago	2 hours	Coquart <coquart@cerfacs.fr>	fundy-oa3-mct-master_since2019	build successful
27	14 days ago	2 hours	valcke <valcke@cerfacs.fr>	fundy-oa3-mct-master_since2019	build successful
26	a month ago	2 hours	coquart <coquart@cerfacs.fr>	fundy-oa3-mct-master_since2019	build successful
25	a month ago	2 hours	coquart <coquart@cerfacs.fr>	fundy-oa3-mct-master_since2019	failed 'echo "Run ...' (failure)
24	2 months ago	2 hours	coquart <coquart@cerfacs.fr>	fundy-oa3-mct-master_since2019	build successful