Check, debug and relaunch simulation and post-processing jobs

This section describes the monitoring tools, the tools to identify and solve problems, and the tools to monitor and restart the post processing jobs if needed.

Table of Content

1. Check status of your simulations

1.0.1. System tools

The batch manager at each computing center provides tools to check the status of your jobs. For example to know if the job is on queue, running or suspended.

1.0.1.1. TGCC

You can use `ccc_mstat` on Irene. To see the available options and useful scripts, see Working on Irene.

1.0.1.2. IDRIS

You can use `squeue` on Jean Zay. To see the available options and useful scripts, see Working on Jean Zay.

1.0.2. run.card

When the simulation has started, the file `run.card` is created by libIGCM using the template `run.card.init`. `run.card` contains information of the current run period and the previous periods already finished. This file is updated at each run period by libIGCM. You can find here information of the time consumption of each period. The status of the job is set to `OnQueue`, `Running`, `Completed` or `Fatal`.

1.1. End of simulation

Once your simulation is finished you will receive an email saying that the simulation was `Completed` or that it `Failed` and two files will be created in the working directory of your experiment:

- run.card
- Script_Output_JobName

A `Debug` directory is created if the Job failed during the simulation part (in other words when the job is executing calculs in modeles). This directory contains diagnostic text files for each configuration components. It won't be created if the job reaches the time limit and is stopped by the batch scheduler.

If the crash is not properly handeld by libIGCM, you will find a lot of files in `$RUN_DIR`. You can find the path of this directory in `Script_Output_JobName`, search for RUN_DIR.

```
IGCM_sys_MkdirWork : /scratch_of_computer/login/RUN_DIR/Job_number/***/
IGCM_sys_Cd : /scratch_of_computer/login/RUN_DIR/Job_number/***/
```

If the simulation was successfully completed output files will be stored in the following directory:

- `STORE/IGCM_OUT/TagName/[SpaceName]/[ExperimentName]/JobName` at TGCC and IDRIS if you are running a DEVT or PROD simulation
- `SCRATCH/IGCM_OUT/TagName/[SpaceName]/[ExperimentName]/JobName` at TGCC and IDRIS if you are running a TEST simulation

in case of a DEVT or PROD simulation, you will find the following subdirectories:

- `RESTART` = tar of the restart files for all model components and with the pack frequency
- `DEBUG` = tar of the debug text files for all model components
- `ATM`
- `CPL`
- `ICE`
- `OCE`
- `SRF`
- `SBG`
- `Out` = run log files
- `Exe` = executables used for the run
- `ATM/Output`, `CPL/Output`, etc... = NetCDF output of the model components

in case of a TEST simulation, you will find the following subdirectories:

- `ATM`
- `CPL`
- `ICE`
- `OCE`
- `SRF`
- `SBG`
- `Out` = run log files
- `Exe` = executables used for the run
- `ATM/Output`, `CPL/Output`, etc... = NetCDF output files of the model components
- `ATM/Restart`, `CPL/Restart`, etc... = NetCDF restart files of the model components
- `ATM/Debug`, `CPL/Debug`, etc... = text output files of the model components

1.2. Diagnostic tools : Checker

1.2.1. TimeSeries_Checker

The `TimeSeries_Checker.job` can be used in diagnostic mode to check if all time series have been successfully created. Read more further below.

1.2.2. SE_Checker

See further below.

---

2. Analyzing the Job output : Script_Output

Reminder --> This file contains three parts:

- prepare parameters files, and copying the input files
- running the model
- post processing

These three parts are defined as follows:

```
########################################
#       ANOTHER GREAT SIMULATION       #
########################################

1st part (prepare parameters files, copying the input files)

########################################
#      DIR BEFORE RUN EXECUTION        #
########################################

2nd part (running the model)

########################################
#       DIR AFTER RUN EXECUTION        #
########################################

3rd part (post processing)
```

A few common bugs are listed below:

- if the file ends before the second part, possible reasons can be:
  - you didn't delete the existing `run.card` file in case you wanted to overwrite the simulation;
  - you didn't specify `OnQueue`in the `run.card` file in case you wanted to continue the simulation;
  - one of the input files was missing (e.g. it doesn't exist, the machine has a problem,...);

  - the frequencies (`PackFrequency` ...) do not match `PeriodLength`.
  - `PeriodLength` do not match with `DateBegin` and `DateEnd`
- if the file ends in the middle of the second part, it's most likely because you didn't request enough memory or CPU time.
- if the file ends in the third part, it could be caused by:
  - an error during the execution;
  - a problem while copying the output;
  - a problem when starting the post processing jobs.

If the following message is displayed in the second part of the file, it's because there was a problem during the execution:

```
========================================================================
EXECUTION of : /usr/bin/time ccc_mprun -E-K1  -f ./run_file
Return code of executable : 1
IGCM_debug_Exit :  EXECUTABLE

!!!!!!!!!!!!!!!!!!!!!!!!
!!   ERROR TRIGGERED    !!
!!   EXIT FLAG SET      !!
!------------------------!

0 - IGCM_debug_Exit (_0_)
IGCM_sys_Mkdir : /path_of_your_simulation/Debug
IGCM_sys_Cp : out_execution /path_of_your_simulation/Debug/JobName_PeriodDateBegin_PeriodDateEnd_out_execution_error
========================================================================
```

In this case you need to explore the Debug directory.
If the following message is displayed :

```
========================================================================
EXECUTION of : mpirun -f ./run_file > out_run_file 2>&1
========================================================================
```

If there is a message indicating that the `restartphy.n` file doesn't exist it means that the model simulation was completed but before the end date of your simulation. If this happens you must refer to the output log of each model of your simulation. For example, the output file of the ocean model is stored on the file server under this name:

```
IGCM_sys_Put_Out : ocean.output xxxxxxxx/OCE/Debug/xxxxxxxx_ocean.output
```

you can retrieve them in the RUN_DIR directory of your simulation.

In general, if your simulation stops you can look for the keyword "IGCM_debug_CallStack" in this file. This keyword will come after a line explaining the error you are experiencing.

```
Example :

--Debug1--> IGCM_comp_Update

IGCM_debug_Exit :  IGCM_comp_Update missing executable create_etat0_limit.e

!!!!!!!!!!!!!!!!!!!!!!!!
!! IGCM_debug_CallStack !!
!------------------------!
```

3. Debug

Your problem could come from a programming error. To find it you can use the text output of the model components located in the Debug subdirectories. Your problem could be caused by the computing environment. This problem is not always easy to identify. It is therefore important to perform benchmark simulations to learn about the usual behavior of a successfully completed simulation.

3.1. The Debug directory

If the simulation failed due to abnormal exit from the executable, a `Debug` directory is created in the working directory. It contains output text files of all model components for your configuration. You should read them to look for errors.

- `xxx_out_gcm.e.err` and `xxx_out_gcm.e.out`(or `xxx_out_lmdz.e.err` and `xxx_out_lmdz.e.out`) --> lmdz text output
- `xxx_out_orchidee` --> orchidee text output
- `xxx_ocean.output` --> nemo text output
- `xxx_inca.out` --> inca text output
- `xxx_out_xios.e.err` and `xxx_out_xios.e.out` --> xios log and error message
- `xxx_run.def` --> lmdz parameter files
- `xxx_gcm.def` --> lmdz parameter files
- `xxx_traceur.def` --> lmdz parameter files
- `xxx_physiq.def` --> lmdz parameter files
- `xxx_orchidee.def` --> orchidee parameter files
- `xxx_****.xml` --> all xml files use for this simulation

In models logs files (out_lmdz.x.err, out_lmdz.x.out, out_orchidee, ...) you will find output for each process of the simulation. You can look at the end of the first process to find main error message.

Your best friend is : `grep -i error * ; grep -i 'e r r o r' *ocean.output`

3.2. Programming error

Please, take the time to read and analyze modifications you have done in the code. Nobody codes perfectly.

3.3. Unknown error

In this case, it's possible to relaunch the main job to run again the last period.

If the simulation stopped before coming to the end due to an error, it is possible to relaunch the latest period after eventual modifications. The simulation will then read run.card to know where to start and the simulation will continue until the end (if the problem was solved).

To relaunch manually you first need to be sure that no files have been stored for the same period. In libIGCM there are 2 scripts that help you do this cleaning up :

- The error occurred before the packs have been created:

```
path/to/libIGCM/clean_PeriodLength.job
```

- The error occurred after the packs were created:

```
path/to/libIGCM/clean_latestPackperiod.job [CCYY]
# CCYY = year up to which you are deleting everything (this year included). By default, it's the current year in run.ca
```

3.4. Use of RUN_DIR directory to run without libIGCM infrastructure

Sometimes, that could be useful (and more effective) to have all the information of the run in the same directory : that allows you to run directly into the RUN_DIR directory, using a Job_debug to be launched. To activate this debug functionality (available from libIGCM rev 1576) :

- modify the main computing Job

```
DRYRUN=4
```

- go into RUN_DIR directory and submit the Job, as indicated in the message at the end of Script_Output of the main Job.

```
##########################################
#    DEBUG PHASE : CREATION OF RUN_DIR    #
```

```
##########################################

You are in development or debug phase
You can run directly into the running directory which is here
/ccc/scratch/cont003/gencmip6/p86caub/RUN_DIR/7485895_33135/DEBUG-LIBIGCM.02.33135
Inside the run directory you will find a Job_debug_DEBUG-LIBIGCM.02
to be used to launch the run as follows :
ccc_msub Job_debug_DEBUG-LIBIGCM.02
```

- Note that it is needed to generate a new Job (via ./ins_job command) in case of update of libIGCM to benefit from this new functionality.

4. Start or restart post processing jobs

You can run post processing jobs once the main job is finished (for example if the post processing job was deactivated in config.card or if you encountered a bug).

You can:

1. Work in a dedicated directory located in the experiment directory (e.g. `PATH_MODIPSL/config/IPSLCM5A/ST11/POST_REDO`). **Best choice.**
2. Work directly in the experiment directory (which looks like `PATH_MODIPSL/config/IPSLCM5A/ST11/`). Possible option but not recommended.

For the first option (recommended) you have to copy (or link `ln -s`) the files and directories `config.card` `POST`, `COMP` and run.card

```
cd $PATH_MODIPSL/config/IPSLCM5A/ST11
mkdir -p POST_REDO
cd POST_REDO/
cp -pr ../COMP  .
cp -pr ../POST  .
cp -pr ../config.card  .
cp -pr ../run.card .
```

For more informations about the post-processing, see Running simulation and post-processing

4.1. Restart REBUILD

Most of configurations no longer has a Rebuild step, due to the use of parallel I/O (done with XIOS). In most of the case, you can go to the next step. The rebuild step is sill use in some ORCHIDEE configurations

- Copy the `libIGCM/rebuild_fromWorkdir.job` file to the experiment directory or to the dedicated directory;
- Edit it:

```
StandAlone=true

libIGCM=                    # Points to the libIGCM directory of the experiment

PeriodDateBegin=            # beginning date of the last serie to be "rebuilded"

NbRebuildDir=               # Number of directories in the series to be "rebuilded"
                           # until the PeriodDateBegin


REBUILD_DIR=                # Path for the backup of files waiting to be reconstructed
                           # (looking like $SCRATCHDIR/IGCM_OUT/.../JobName/REBUILD or $SCRATCHDIR/TagName/JobName/REBU
                           # if RebuildFromArchive=NONE)

MASTER=${MASTER:=irene|jeanzay} # Select the computing machine : MASTER=irene for example
```

- Submit the job:

```
ccc_msub rebuild_fromWorkdir.job                # TGCC

sbatch rebuild_fromWorkdir.job                  # IDRIS
```

The rebuild job submits `pack_output.job` automatically.

4.2. Restart Pack_output

In case you haven't done it yet, copy `config.card` `COMP` `POST` and `run.card` (post process only part of the simulation) in the `POST_REDO/` directory.

> Note: you need to do this part in case the pack_output was not submitted by the rebuild job (if used), or if you encountered a bug.

- Copy the `libIGCM/pack_output.job` file to the experiment directory or to the dedicated directory;
- Edit it :

```
libIGCM=${libIGCM:=::modipsl::/libIGCM}         # path of the libIGCM library

MASTER=${MASTER:=irene|jeanzay}                     # machine on which you work

DateBegin=${DateBegin:=20000101}                # start date of the period to be packed

DateEnd=${DateEnd:=20691231}                    # end date of the period to be packed

PeriodPack=${PeriodPack:=10Y}                   # pack frequency
```

- Submit the job:

```
ccc_msub pack_output.job                        # TGCC

sbatch pack_output.job                          # IDRIS
```

`create_ts.job` and `create_se.job` are submitted automatically.

4.3. Restart Pack_restart or Pack_debug

For more informations about the "Pack" : see Concatenation of "PACK" outputs.

In case you haven't done it yet, copy `config.card` `COMP` `POST` and `run.card` (post process only part of the simulation) in the `POST_REDO/` directory.

- Copy the `libIGCM/pack_debug.job` and `libIGCM/pack_restart.job` files to the experiment directory or to the dedicated directory;
- Edit them :

```
libIGCM=${libIGCM:=::modipsl::/libIGCM}         # path of the libIGCM library

MASTER=${MASTER:=irene|jeanzay}                     # machine on which you work

DateBegin=${DateBegin:=20000101}                # start date of the period to be packed

DateEnd=${DateEnd:=20691231}                    # end date of the period to be packed

PeriodPack=${PeriodPack:=10Y}                   # pack frequency
```

- Submit the two jobs:

```
ccc_msub pack_debug.job ; ccc_msub pack_restart.job      # TGCC
```

```
sbatch pack_debug.job ; sbatch pack_restart.job          # IDRIS
```

4.4. Restart the Time Series with TimeSeries_checker.job

In case you haven't done it yet, copy `config.card` `COMP` `POST` and `run.card` (post process only part of the simulation) in the `POST_REDO/` directory.

- Copy the `libIGCM/TimeSeries_Checker.job` file to the experiment directory or to the dedicated directory besides COMP and config.card as minimum
- You do not need to modify the `TimeSeries_Checker.job`. By default it takes the variables from config.card.
- Check the value of TimeSeriesCompleted in run.card. For a complete check set :

```
TimeSeriesCompleted=
```

Run the `TimeSeries_Checker.job` in interactive mode. It will call the missing create_ts jobs :

```
./TimeSeries_Checker.job
```

or alternatively, in ksh :

```
./TimeSeries_Checker.job 2>&1 | tee TSC_OUT     # Create log file

grep Batch TSC_OUT                              # find all the submitted jobs
```

Answer n to the following question to only see what is missing and answer y to submit all missing create_ts.job :

```
"Run for real (y/n)"
```

4.5. Restart the Seasonal Mean calculations

In case you haven't done it yet, copy `config.card` `COMP` `POST` and `run.card` (post process only part of the simulation) in the `POST_REDO/` directory.

There are two methods:

4.5.1. SE_Checker.job (recommended method)

- Copy the `libIGCM/SE_Checker.job` file to the experiment directory or to the dedicated directory
  Run `SE_checker.job` in interactive mode. This will call the create_se jobs:

```
./SE_Checker.job
```

or alternatively, in ksh :

```
# Create logfile:
./SE_Checker.job 2>&1 | tee SE_OUT
# Find all started jobs :
grep Batch SE_OUT
```

4.5.2. Restart create_se.job

- Copy the `libIGCM/create_se.job` file to the experiment directory or to the dedicated directory;
- Edit it:

```
StandAlone=true
```

```
libIGCM=                                    # path of the libIGCM library

PeriodDateEnd=                              # end date of the decade to be processed
```

- Submit the job:

```
ccc_msub create_se.job                      # TGCC

sbatch create_se.job                        # IDRIS
```

4.6. Restart the monitoring with monitoring.job

Transfer `config.card`, `COMP`, `POST`, and `run.card` (post process part of the simulation only) in the `POST_REDO/` directory if you have not done so yet.

- Copy the `libIGCM/monitoring.job` file to the experiment directory or to the dedicated directory;
  You do not need to modify the `monitoring.job`. By default it takes the variables from config.card.
  - If ORCA_version=eORCA1.2 in config.card, then change to RESOL_OCE=ORCA1 in monitoring.job
- Submit the job:

```
ccc_msub monitoring.job                     # TGCC

sbatch monitoring.job                       # IDRIS
```

4.7. Restart the atlas figures creation with `create_se.job`

Transfer `config.card`, `COMP`, `POST`, and `run.card` (post process part of the simulation only) in the `POST_REDO/` directory if you have not done so yet.

- Copy the `libIGCM/create_se.job` file to the experiment directory or to the dedicated directory;
- Edit it and indicate the period (yyyymmdd) for which you need to redo the atlas. For example.

```
#
DateBegin=19600101
#
PeriodDateEnd=19691231
```

- Submit the job:

```
ccc_msub create_se.job                      # TGCC

sbatch create_se.job                        # IDRIS
```

5. Optimization with Lucia

IPSLCM coupled model runs three executables (atmosphere, ocean and IO server) that use three separate sets of computing cores. The number of cores attributed to each one should be choose such as the execution times of each executable are as close as possible, to reduce the waiting time.

LUCIA is a tool implemented in OASIS that measure execution and waiting times of each executable, and helps to tune the number of execution cores for each model.

5.1. LUCIA documentation

■http://www.cerfacs.fr/oa4web/papers_oasis/lucia_documentation.pdf

5.2. Using LUCIA

First install and run a coupled model. Then performs some modifications.

5.2.1. Get a version of OASIS with LUCIA

```
cd modipsl : mv oasis-mct oasis-mct_orig
cp -rf $CCCHOME/../../igcmg/igcmg/Tools/oasis3-mct_lucia oasis3-mct
cd config/IPSLCM6 ; gmake clean ; gmake
```

5.2.2. Update DRIVER/oasis.driver (example for Irene)

```
Index: oasis.driver
===================================================================
--- oasis.driver        (revision 3545)
+++ oasis.driver        (working copy)
@@ -117,6 +117,7 @@
    #   To be changed
    #   On Irene
+     ~igcmg/Tools/irene/lucia/lucia
    #   To be changed
    #   On Jean Zay
    #   $HOME/../../psl/rpsl035/LUCIA/lucia
    fi
```

5.2.3. Update COMP/oasis.card

```
Index: oasis.card
===================================================================
--- oasis.card  (revision 3545)
+++ oasis.card  (working copy)
@@ -4,7 +4,7 @@
[UserChoices]
OutputMode=n
FreqCoupling=5400
-Lucia=n
+Lucia=y
```

5.2.4. Run the model

5.2.4.1. Script_Output will contains some additionnal information

```
  Component -           Computation -       Waiting time (s) - done on tstep nb

 LMDZ          1385.77 ( +/-   6.69 )            7.58 ( +/- 6.14 )  362
 oceanx        1319.37 ( +/-  18.68 )           85.41 ( +/- 18.70 )  362
 xios.x           0.00 ( +/-   0.00 )            0.00 ( +/- 0.00 )    0

 New analysis

 Component -         Calculations   -     Waiting time (s) - done on tstep nb:

 LMDZ                     1379.55                 6.45       362
 oceanx                   1300.79                85.21     362
 xios.x                      0.00 0.00           0
```

5.2.5. LUCIA will also procuce a graphic that you will find in :

```
IGCM_OUT/${TagName}/${SpaceName}/${ExperimentName}/${JobName}/CPL/Debug/${JobName}_*******_oasis_balance.eps
```