Load balancing analysis with OASIS3-MCT

November 2021

1. Introduction

In Coupled General Circulation Models (CGCMs), computations are often performed concurrently in separated executables implementing the different coupled model components. Their respective computing rates in terms of Simulated Year Per Day (SYPD) is a function of their MPI parallelism and the CGCM cannot be load balanced without setting respective appropriate MPI domain decompositions, which is unlikely to happen without a comprehensive work. An analysis of CMIP5 model simulation (Balaji et al., 2017) shows that up to 62% of the allocated resources can be wasted because of this imbalance. Even though this problem has several origins, such as the apparent resource abundance or practical difficulties. we think that, in the perspective of the reduction of the carbon footprint of our activities, the issue of load balancing our coupled configuration is well worth some efforts.

Any proposed solution should be simple and portable enough to fit the community requirements and should be available in the coupler itself. A new version of the OASIS3-MCT load balancing analysis tool is available since OASIS3-MCT_5.0 release (Maisonnave et al., 2021) and is described here. It produces the full timeline of all OASIS3-MCT related

events for each of the allocated resources in a NetCDF file. A Python script is developed to visualise this timeline and its native zoom function facilitates the identification of possible bottlenecks of the coupling (Piacentini & Maisonnave, 2021). More general computing information (simulation time, speed, waiting time, etc.) is also provided for the coupled model and for each component in a text file.

2. Installation

The load balancing analysis functionality can be activated by simply setting to 1 the third number under $NLOGPRT in the *namcouple* configuration file (see the User Guide section 3.2 at https://oasis.cerfacs.fr/en/documentation/).

When activated, this functionality outputs the full timeline of all OASIS3-MCT related events, for any of the allocated resources in one NetCDF file per coupled component, `timeline_XXXX_component.nc` , where `XXX` is the component name (see section 3).

A Python script, `pyLucia.py`, is available in directory `oasis3-mct/util/load_balancing` to visualise results from the NetCDF files . It relies on the following specific libraries:

- `math`
- `sys`
- `os`
- `time`
- `numpy`
- `matplotlib`
- `netCDF4` to read the format of the timeline produced by OASIS ;
- `json` or `yaml` to easily configure the visualization

An appropriate Python3 library and environment is necessary to launch `pyLucia.py`. The 3.7.9 version was tested successfully on a legacy Intel Harpertown desktop, but an older v3 should match.

The load balancing analysis functionality also produces a text file `load_balancing_info.txt` with general computing information (simulation time, speed, waiting time, etc.) for the coupled model and for each component (see section 4).

3. Full timeline of all OASIS3-MCT related events

The load balancing analysis functionality provides for each component[1] the timeline of all operations related to the coupling, so that any simulation slowdown linked to the use of the OASIS3-MCT library can be identified. We call *timeline* a temporal sequence of events occurring during a coupled simulation. The analysis of the timelines, stored in NetCDF files,

[1] OASIS3-MCT gives the possibility to declare independent partitions for subsets of component processes (see section 2.1 of the OASIS3-MCT User Guide). At this development stage, considering the low number of prejudiced users and our wish to keep our algorithm as simple as possible, we prefer not to support the load balancing analysis in that case.

allows to not only identify the waste of resources by components recurrently waiting for their coupling fields but it may also reveal other bottlenecks such as disk access or model internal load imbalance. The full picture of these events makes possible an optimal load balancing, even for the most complex configurations.

3.1. Events

The events measured can be common to all processes and independent of the field exchanges, i.e. (with their flag meaning and flag value in the NetCDF files):

- MPI partitioning definition (`PART`, 8)
- Coupling definition phase, including regridding weight and address computations (`ENDF`, 9)
- Termination (`TERM`, 10)

Events also occur in the time loop and are related to a field exchange:

- sending (`PUT`, 1)
- receiving (`GET`, 2)
- mapping and regridding (`MAP`, 3)
- output to file (`OUT`, 4)
- reading from file (`READ`, 5)
- restart writing for standard coupling fields (`RST`, 6)
- restart writing for coupling fields on which temporal operations are performed, when the coupling period is not complete at the end of the run (`TRN`, 7)

3.2. Timeline file

One timeline file is produced in one NetCDF file per coupled component, `timeline_XXXX_component.nc` , where `XXX` is the component name. Each file describes:

- the clock time when any event (`nx`) on any MPI process (`ny`) starts and stops `timer_strt` and `timer_stop` ;
- the kind of each event, `kind(nx)` (see their flag value in the paragraph above);
- the ID of the exchanged field, `field(nx)`;
- the ID of the component exchanging the coupled field, `component(nx).`

3.3. Python script configuration with json or yaml file

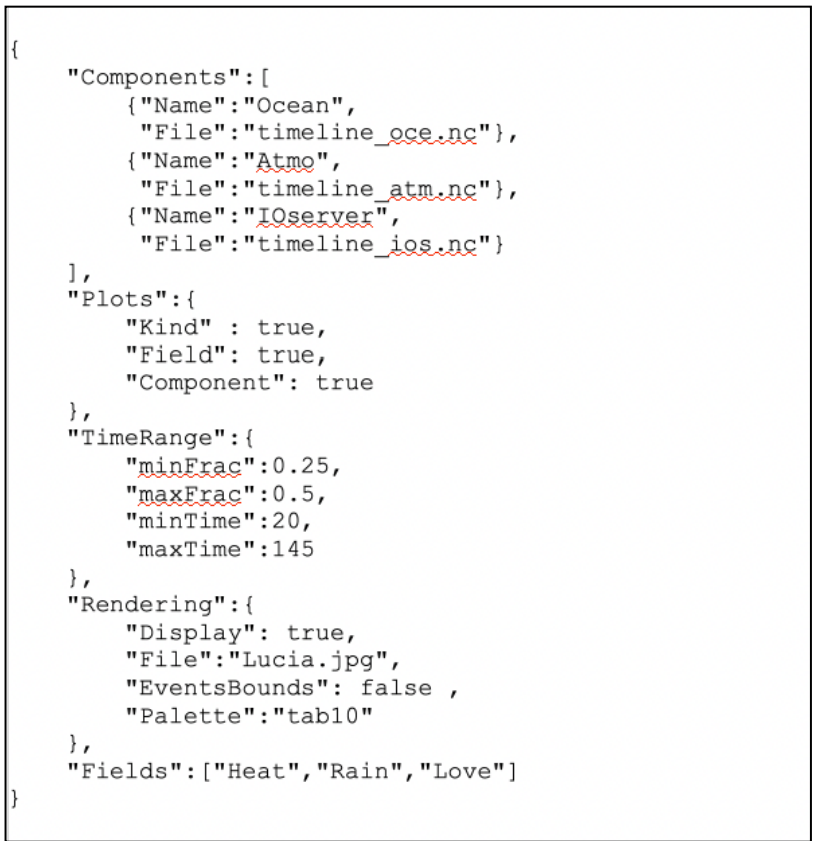
To run the Python script, `pyLucia.py`, you have to indicate as argument the name of the configuration `json` (or `yaml`) parameter file, e.g.:

```
> pyLucia.py lucia.yaml
```

or

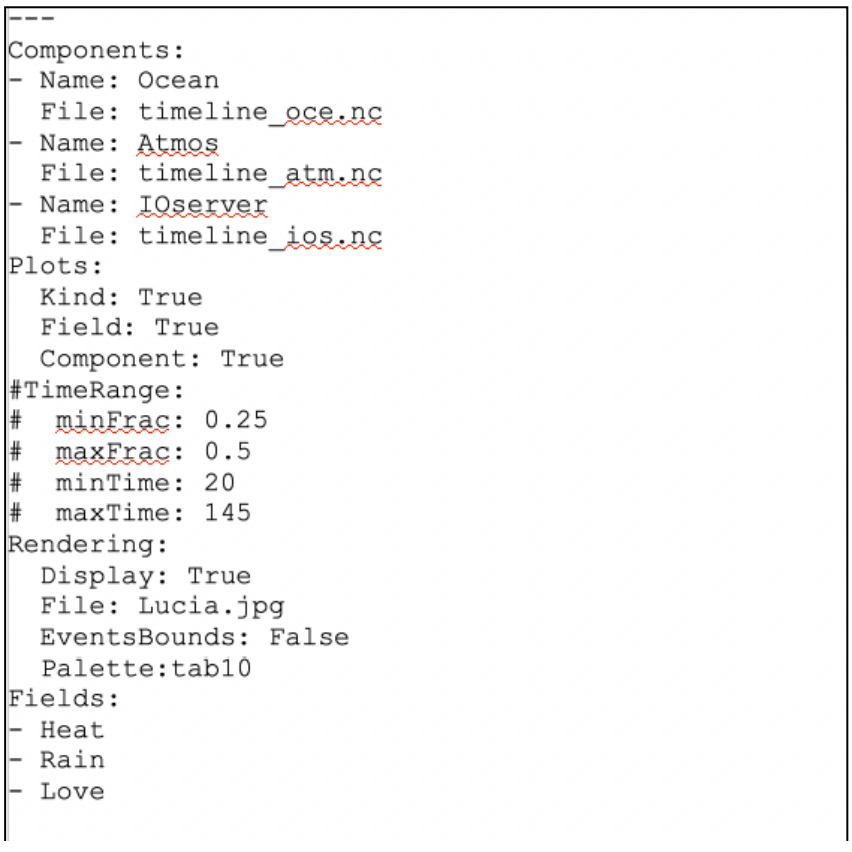
```
> pyLucia.py lucia.json
```

The `yaml` or `json` configuration files, see Figures 1 and 2 respectively, include several mandatory or optional objects. They define the visualization workflow and information that appears in the resulting graphs to label (or design) the plotted timeline.



```
{
    "Components":[
        {"Name":"Ocean",
         "File":"timeline_oce.nc"},
        {"Name":"Atmo",
         "File":"timeline_atm.nc"},
        {"Name":"IOserver",
         "File":"timeline_ios.nc"}
    ],
    "Plots":{
        "Kind" : true,
        "Field": true,
        "Component": true
    },
    "TimeRange":{
        "minFrac":0.25,
        "maxFrac":0.5,
        "minTime":20,
        "maxTime":145
    },
    "Rendering":{
        "Display": true,
        "File":"Lucia.jpg",
        "EventsBounds": false ,
        "Palette":"tab10"
    },
    "Fields":["Heat","Rain","Love"]
}
```

Figure 1 - Example of a `json` configuration file for the Python script `pyLucia.py`



```
---
Components:
- Name: Ocean
  File: timeline_oce.nc
- Name: Atmos
  File: timeline_atm.nc
- Name: IOserver
  File: timeline_ios.nc
Plots:
  Kind: True
  Field: True
  Component: True
#TimeRange:
#  minFrac: 0.25
#  maxFrac: 0.5
#  minTime: 20
#  maxTime: 145
Rendering:
  Display: True
  File: Lucia.jpg
  EventsBounds: False
  Palette:tab10
Fields:
- Heat
- Rain
- Love
```

Figure 2 - Example of a `yaml` configuration file for the Python script `pyLucia.py`

3.4. Components

The files describing the timelines for each component must be specified. A model name must be added to fully describe the input information. User must declare one sub-object `(Name, File)` per component.

3.5. Plots

Up to three graphics can be displayed in the same plot depending on the value of `Kind`, `Field` or `Component` sub-object. With Kind set to True, a graph showing the type of event (`PUT, GET, MAP, OUT, READ, RST, TRN, PART, ENDF, TERM)` for each event is output (see the upper graph in Figure 3). With `Field` set to `True`, a graph showing the coupling field implied in the event is output (see the middle graph in Figure 3). With `Component` set to `True`, a graph showing the other component implied the event is output (see the lower graph in Figure 3).

3.6. TimeRange

To reduce the timeline along the x-axis (time), a fraction (`minFrac/maxFrac`) or a time window in second (`minTime,maxTime`) of the full timeline can be defined. If both fractions and time bounds are prescribed, only fractions are taken into account.

3.7. Rendering

To visualise the timeline through the matplotlib GUI visualisation tool, the `Display` sub-object must be set to true. In case of saving the graphics in a file, the `File` sub-object must be set to the name of the output file; extension in the name defines the file format. Joint visualisation on the screen and saving in file is possible. As an option, boundaries of the event rectangles can be plotted; this option (`EventsBounds` sub-object) can clarify the event sequence when several events with same `Kind/Field/Component` follow one another. The `Palette` option allows to choose among the `matplotlib` built-in colormaps.

3.8. Fields

Coupling fields are identified in OASIS3-MCT by numerical or alphanumerical IDs. For a non-ambiguous naming, it is required to explicitly provide a name for each coupling field in this object. Fields must be named following the *namcouple* sequence.

4. Graphical output

When the `File` sub-object of `Rendering` in the `json/yaml` file is set to the name of an output file, the plots are saved in this file. If the `Display` sub-object is set to true, a call to the `matplotlib.pyplot.show` command at the end of the script opens an interactive window. An example of timeline visualisation with this GUI is shown in Figure 3.

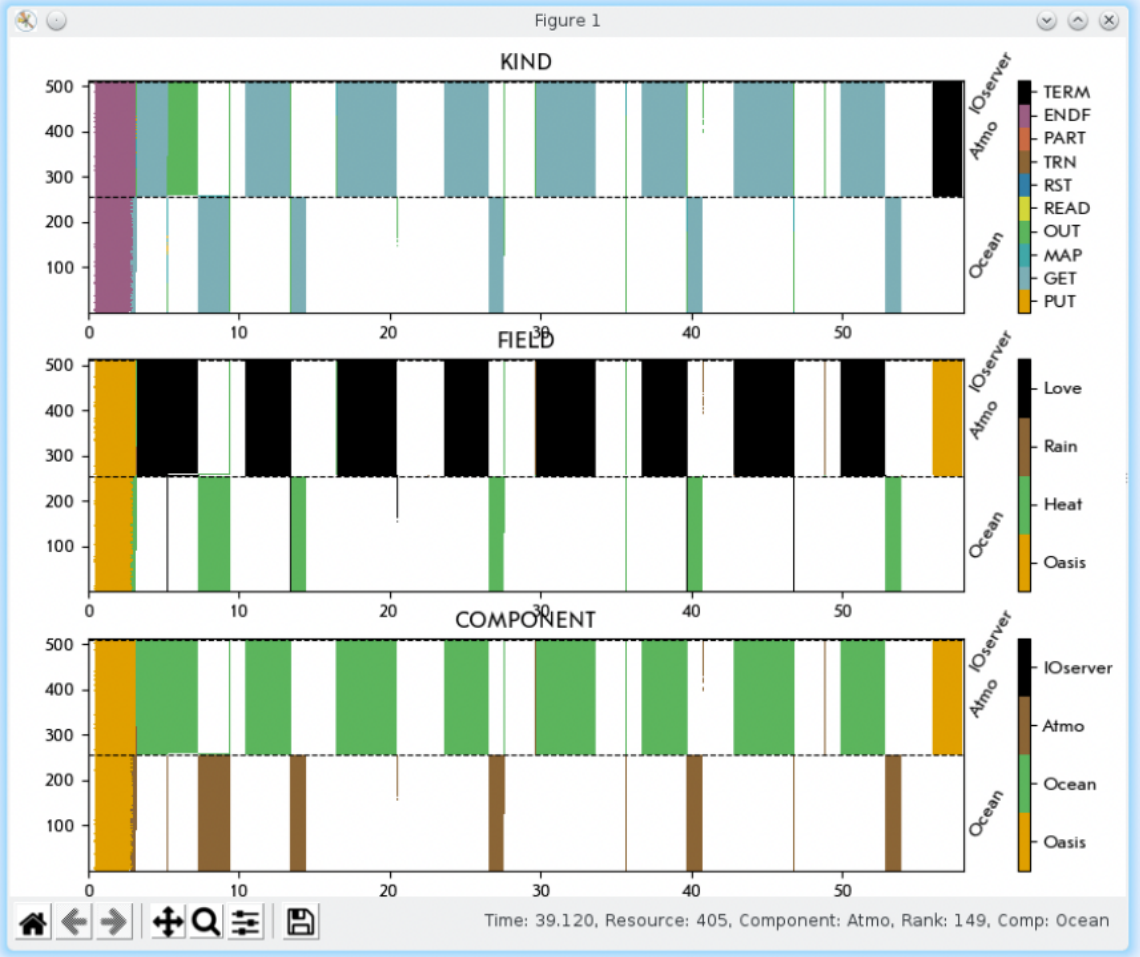


Figure 3 - Example of the graphs interactively displayed when `Display` sub-object is set to true. The type of event for each event is plotted in the upper graph (`Kind`). The coupling field implied in the event is plotted in the middle graph (`Field`). The other component implied the event is plotted in the lower graph (`Component`). The x axis represents the elapsed time and the y axis represents the MPI rank of all component processes (in the global communicator).

The belonging of resources to each component is delimited by dashed black lines. Zooms and movements in plots are possible via push buttons. The zoomed figure can be saved into graphical format files. The mouse cursor is configured to display its position (time/resource number) and the name associated to the designated rectangle (`Kind/Field/Component`). The local communicator MPI rank and the name of the components are also shown.

5. General computing information in text file

The load balancing analysis functionality also produces a text file `load_balancing_info.txt` with general computing information (simulation time, speed, waiting time, etc.) for the coupled model and for each component. An example of this file is illustrated at Figure 4. In simple cases, this global information can help to allocate resources in a balanced way.

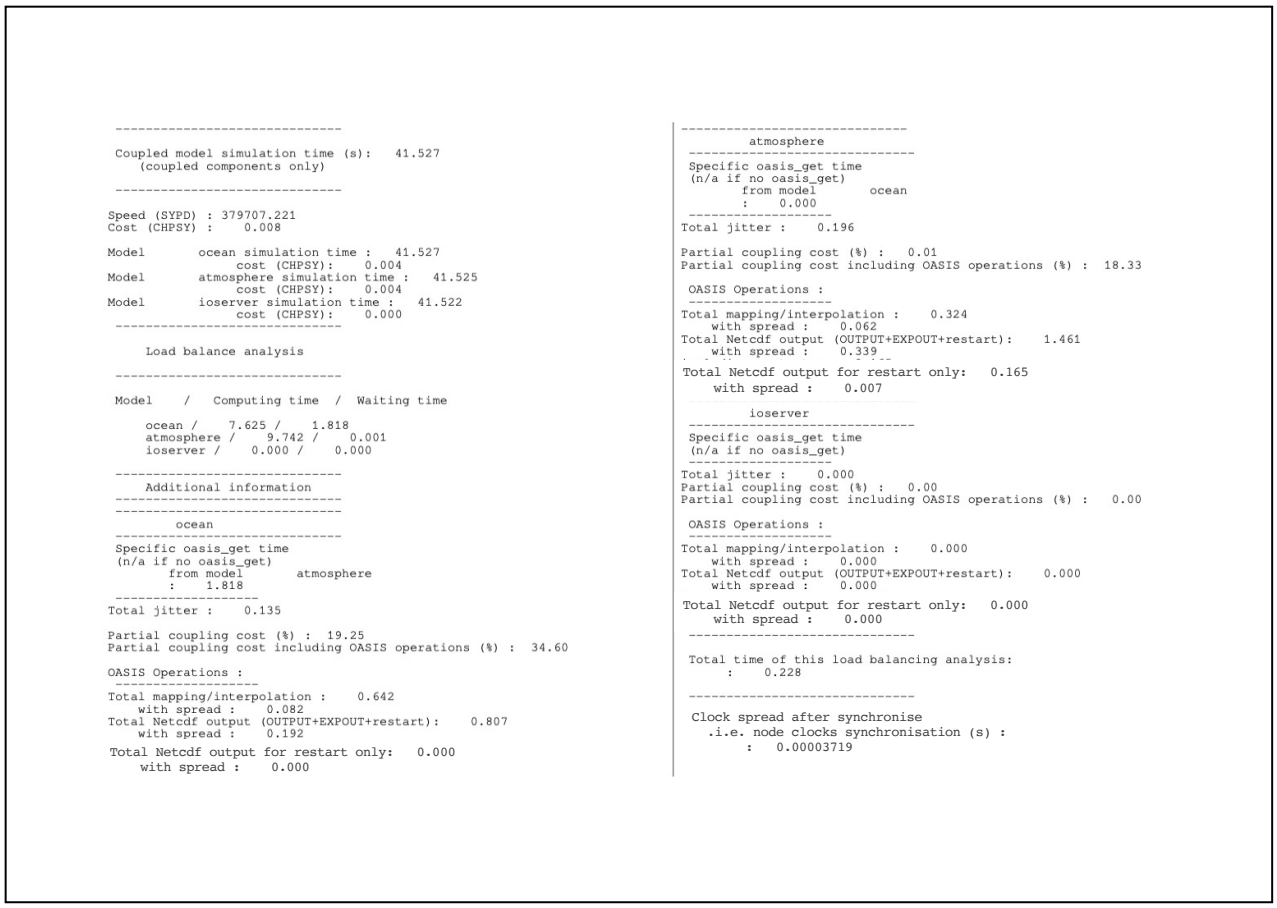


Figure 4 - Example of a `load_balancing_info.txt` file providing general computing information

The definition of the numbers delivered by this analysis are the following:

- the total time of the simulation, from the beginning of the first process calling `oasis_init`, to the end of the last process calling `oasis_terminate` in seconds over all processes of the coupled system ("`Coupled model simulation time (s) :`")
- the speed in Simulated Year per Day (SYPD) of the coupled system ("`Speed (SYPD) :`")
- the cost in Core Hours Per Simulated Year (CHPSY) of the coupled system ("`Cost (CHPSY) :`")
- the total time in seconds and corresponding cost in CHPSY per component (e.g "`ocean simulation time :`" and "`cost (CHPSY):`")

Then under "`Load balance analysis`", this file provides for each component the computing time and the waiting time (e.g. "`ocean / 7.625 / 1.818 `"). The waiting time in seconds is the time spent while receiving and sending[2] coupling fields. The computing time is defined as the difference between the total loop time and the waiting time, where the total

[2] The time spent while sending a coupling field is usually negligible as the oasis_put are non-blocking.

loop time is not the total time but the time spent in the coupling time loop only, excluding the first and the last coupling time steps[3].

Additional information is provided for each component under the line "`Additional information`"; this information has to be taken with care as numbers that may appear illogical may be produced in special cases. In case of doubt, the detailed information from the timeline of events (see sections 3 and 4) should be taken as the reference.

For each component, the total time spent in receiving the fields (averaged over all component processes) is provided ("`Specific oasis_get time`") per corresponding source components ("`from model atmosphere` "). The total of these oasis_get times should be very close to the component waiting time (as the time spent waiting to send a coupling field is usually negligible).

For each component, the "`Total jitter`" is also provided. This total jitter time is defined as the total for all send and receive operations of the time difference between the latest end and the earliest start of the operation among all component processes. The jitter time for one specific operation for a component running on 4 processes is illustrated on Figure 5.

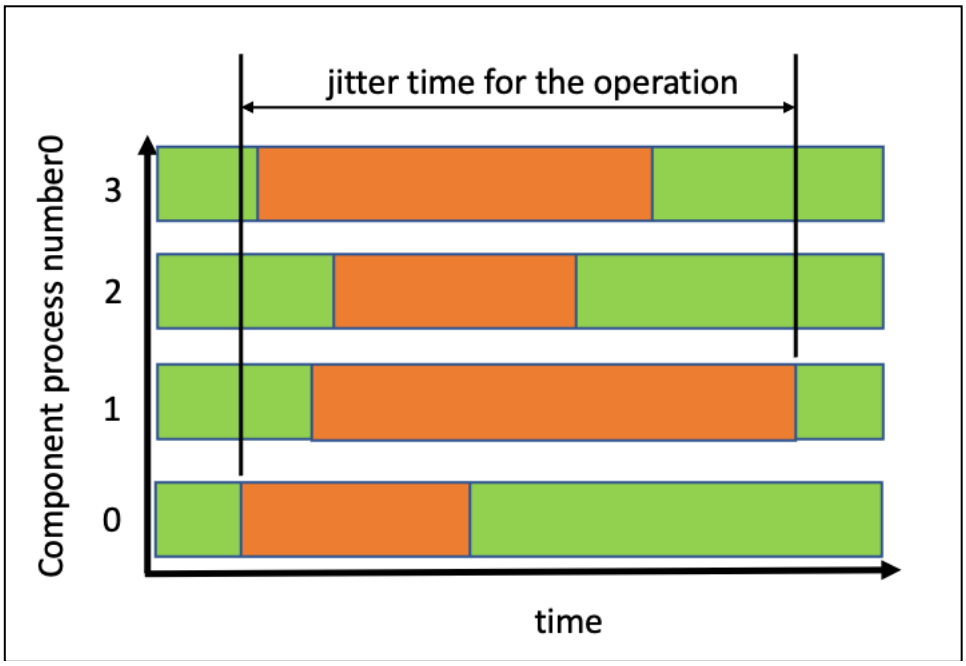


Figure 5 – Jitter time for one specific operation (in orange) for a component running on 4 processes.

The "`partial coupling cost`", which is the ratio (in %) of the component waiting time compared to the total loop time, is also provided for each component, as well as the "`Partial coupling cost including OASIS operations`", which is the ratio (in %) of the component waiting time plus all other OASIS3-MCT runtime coupling field operations (output, input, mapping/interpolation, restarts writing) compared to the total loop time.

[3] The first and the last coupling time steps are excluded from the total loop time as these time steps often involve specific operations, like restart writing or reading; including them would distort the extrapolation of the time needed for longer simulations.

The time spent during OASIS3-MCT runtime coupling field operations is also detailed per component, with averaged values across component processes and corresponding total jitter when starting the operations ("`with spread`"), as follows:

- the total time for mapping/interpolation operations (s) ("`Total mapping/interpolation :`")
- the total time for NetCDF output launched according to the *namcouple* options (OUTPUT, EXPOUT) or during restart writing (s) ("`Total Netcdf output (OUTPUT+EXPOUT+restart) :`")
- the total time for NetCDF restart writing only ("`Total Netcdf output for restart only:`")

Finally, two details are provided to check the performance and the validity of the load balancing analysis itself:

- the duration of the load balancing synthesis phase itself (s) ("`Total time of this load balancing analysis:`");
- the spread of time clocks measured after an MPI_barrier call, to check the coherence of the clock between the computing nodes (s) ("`Clock spread after synchronise`").

6. Conclusions

OASIS3-MCT load balancing analysis functionality delivers at low cost a set of summarized quantities on the computing characteristics of a coupled model that can help to better understand the origin of any coupling extra cost. It also provides the timeline, for all processes involved in the coupling, of every coupling event, such as coupling field sending or receiving, but also coupling initialization and termination, regridding, file reading or writing. These timelines are available in NetCDF files and are easily accessible thanks to a Python visualization script.

To be able to keep the implementation as simple as possible (~ 700 lines), the analysis is limited to a component- (and not a partition-) per-basis. Storing the analysis requires additional memory on the master process of each component approximately equal to the size of a single precision array containing the number of coupling events times the number of MPI processes. This means that users may have to restrain the analysis to relatively short simulations or parallel models with a relatively small number of processes to avoid memory overflow.

7. References

Balaji, V., Maisonnave, E., Zadeh, N., Lawrence, B. N., Biercamp, J., Fladrich, U., Aloisio, G., Benson, R., Caubel, A., Durachta, J., Foujols, M.-A., Lister, G., Mocavero, S., Underwood, S., and Wright, G., 2017: CPMIP: Measurements of Real Computational Performance of Earth System Models in CMIP6 , Geosci. Model Dev., 46, 19-34, doi:10.5194/gmd-10-19-2017

Maisonnave, E., L. Coquart, and A. Piacentini: A better diagnostic of the load imbalance in OASIS based coupled systems Technical Report, TR/CMGC/20/176, CERFACS, Toulouse, France, 2020 https://oasis.cerfacs.fr/wp-content/uploads/sites/114/2021/08/GLOBC_TR_Maisonnave-load-balancing_2020.pdf

Piacentini, A. and E. Maisonnave: Interactive visualisation of OASIS coupled models load imbalance Technical Report, TR/CMGC/20/177, CERFACS, Toulouse, France, 2020 https://oasis.cerfacs.fr/wp-content/uploads/sites/114/2021/08/GLOBC_TR_Piacentini_Interactive_visualisation_of_OASIS_2020.pdf