Interactive visualisation of OASIS coupled models load imbalance

A. Piacentini, E. Maisonnave

TR/CMGC/20/177

Abstract

The OASIS coupling library has recently received improvements to better estimate the load imbalance that a concurrent running of separate executable files can produce. The full timeline of all OASIS related events, for each of the allocated resources, is now available in netCDF format. A Python script is developed to visualise this timeline, thanks to the `matplotlib` *package. Its native zoom function facilitates the identification of possible bottlenecks of the coupling. This document provides a short description of the script functions and a user guide.*

Table of Contents

Abstract....2
1- Tool description....4
  1.1- Principle....4
  1.2- Input....4
  1.3- Implementation....5
2- User guide....5
  2.1- Install....5
  2.2- Parameters....6
    Components....6
    Fields....6
    Rendering....6
    Plots....6
    TimeRange....6
  2.3- Graphical interface....7
Appendix 1: json configuration file example....8
Appendix 2: yaml configuration file example....9
Appendix 3: An OASIS timeline plot using matplotlib....10
References....11

1- Tool description

1.1- Principle

The increasing parallelism of the supercomputers in use in the climate modelling community necessarily imposes to adapt our codes and tools. Any operation that can be processed in parallel must be coded accordingly. In that perspective, a coupler or a coupling framework is a useful tool, that allows the concurrent running of models (coupled components) in coupled systems such as ESMs, e.g. [1]. An efficient use of the allocated computing resources supposes the harmonisation of the components speed. This operation, called load balancing, is often neglected, either because of the apparent resource abundance and practical difficulties. The first barrier must be jumped by people better informed of the resource cost and rarefaction [2,3]. The second issue is addressed for the OASIS coupling library [4] users, since 2014 [5]. Recent improvements [6] lead to produce the full timeline of all OASIS related events, in any of the allocated resources. This timeline provides the *comprehensive* description of most of the operations related to the coupling, so that any simulation slow down in link with the use of the OASIS library can be identified. The huge amount of information displayed requires an efficient visualisation tool [7]. In particular, this visualisation must offer a zoom functionality, to be able to precisely identify interesting events among the mass of the others. For that reason, the `matplotlib` [8] Python tool is preferred to software such as `FERRET` [9].

1.2- Input

If the load balance tracing option is enabled (through `namcouple` parameter), the OASIS library produces timelines in `netCDF` files. We call *timeline* a temporal sequence of events occurring during a coupled simulation. The events can be common to all processes and independent of the field exchanges:

- MPI partitioning description (`PART`)
- Coupling definition phase, including interpolation weights and addresses computations (`ENDF`)
- Termination (`TERM`)

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

- sending (`PUT`)
- receiving (`GET`)
- mapping and interpolation (`MAP`)
- output on file (`OUT`)
- reading on file (`READ`)
- restart writing (`RST`), when partial coupling time step are required (`TRN`)

One timeline file is produced per coupled component. Each file includes variables which describe:

- the clock time when starts and stops any event (nx) on any MPI process (ny) - `timer_strt` & `timer_stop` -,
- the kind of each event – `kind(nx)` -,

- the ID of the exchanged field - `field(nx)` - and
- the ID of the component which exchanges the coupled field - `component(nx)` -

1.3- Implementation

The proposed Python script only relies on three specific libraries:

- `matplotlib`,
- `netCDF4`, to read the format of the timeline produced by OASIS
- `json`, to easily configure the visualisation or
- `yaml`

Each coupling event is plotted with a coloured rectangle. Their x-coordinates are given by the start and stop clock measurement, their y-coordinate by the MPI rank of the process. Only coupling related events are plotted. White areas represent non coupling computing instants, i.e. periods when models are performing their own operations.

The visualisation can by done via the `matplotlib` graphical interface and/or on graphical format file. Several graphical format are available. The PDF vector format also allows zooming over region of interest. To facilitate the printing of the file content, the plot size corresponds to an A4 fold.

A default graphical palette [10] is defined to facilitate the reading to color blind users [11]. However, one could specify its preferred colormap.

To plot the figures takes up to several minutes when tens of million events are involved. If the visualisation interface is not wished by the user (plot in file only), the `X11` server will not be called. This option speeds up the processing, and allows to use the script on terminal only machines. To further speed up the data processing, it can be done on a subset of the event, by defining time boundaries.

2- User guide

2.1- Install

An appropriate Python3 library and environment is necessary to launch our script. The 3.7.9 version was tested successfully on a legacy Intel Harpertown desktop, but an older v3 should match. The previously cited libraries[1] are also mandatory. You have to indicate as argument the name of the `json` (or `yaml`) parameter file described in the next chapter :

```
> pyLucia.py lucia.yaml
```
or
```
> pyLucia.py lucia.json
```

after defining the appropriate paths.

1 `NetCDF4 , json , sys, os , time , numpy, math, matplotlib`

2.2- Parameters

The `json` configuration file (an example is given in Appendix 1) includes several mandatory or optional objects. They define the workflow of our visualisation, and information that is not included in the OASIS `netCDF` input file but must appear in the visualisation to label (or design) the plotted timeline. Optionally, the same information can be described in a `yaml` format configuration file (see Appendix 2 for example)

Components

The files describing the timelines of the coupled system must be named. A model name must be added to fully describe the input information. User must declare one sub-object (`file,name`) per component.

Fields

Coupling field are identified in OASIS by numerical or alphanumerical IDs. For a non ambiguous naming, it is required to the user to explicitly provide the coupling field names in this `json/yaml` object. Fields must be named following the OASIS `namcouple` sequence.

Rendering

One may choose to visualise the timeline through the `matplotlib` GUI visualisation tool. In this case, the `Display` sub-object must be set to true. In case of plotting the graphics in file, the `File` sub-object value must be set to the name of the output file. Extension in the name defines the file format. Joint visualisation at screen and writing on 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 of the same kind/field/counterpart_component follow one another. The `Palette` option allows to choose among the `matplotlib` built-in colormaps.

Plots

Up to three graphic can be displayed in the same plot. For that, the user will set to true the `Kind`, `Field` or `Component` sub-object, depending on the `netCDF` variable(s) it would like to see.

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.

2.3- Graphical interface

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 1. 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 or component). The local communicator MPI rank and the name of the component are also shown.

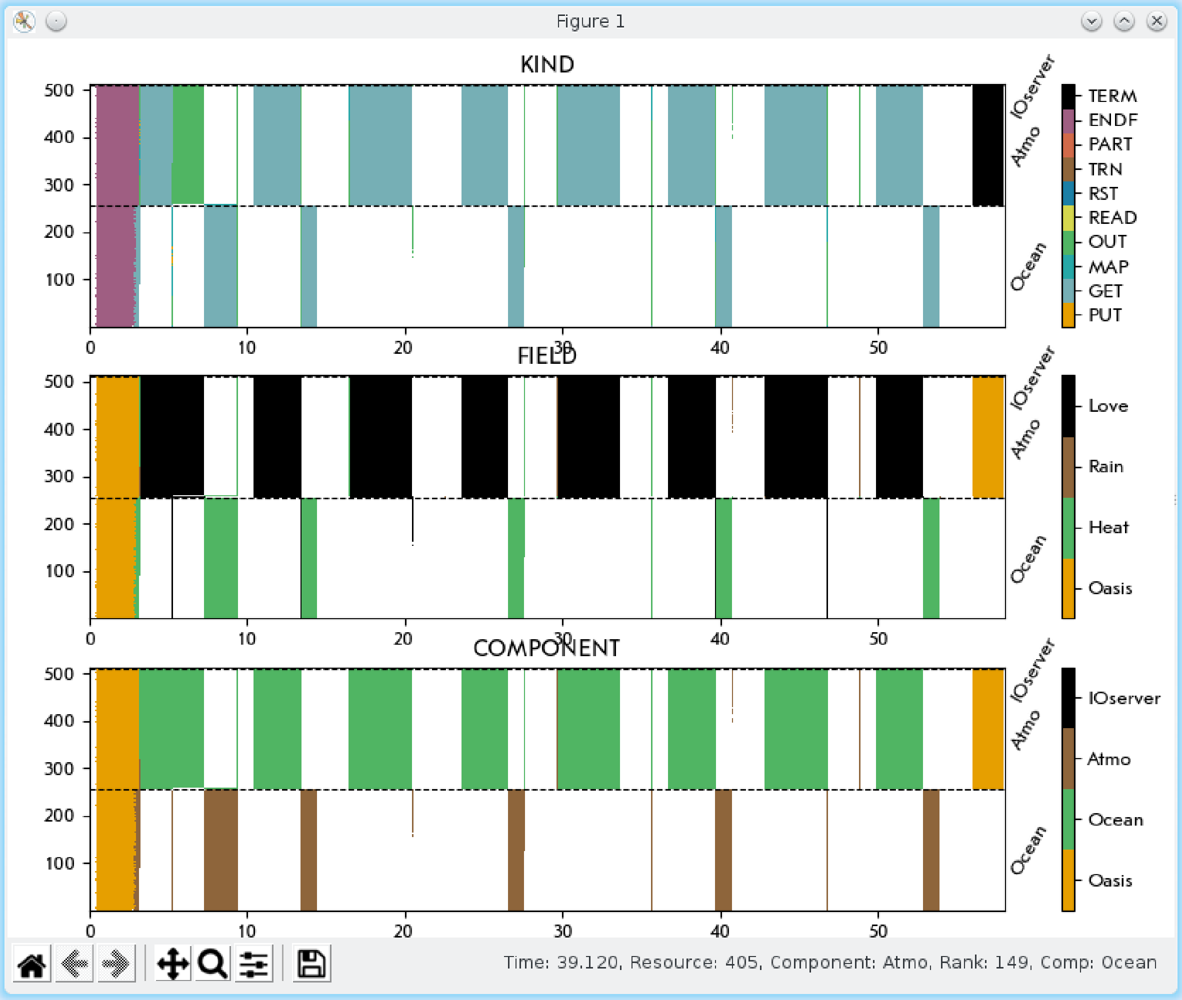


Figure 1: Example of GUI visualisation of kind, coupling field and component counterpart in timelines of three coupled components

Acknowledgments:
This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824084 (IS-ENES-3). whereby.com provided the browser-based video meeting tool.

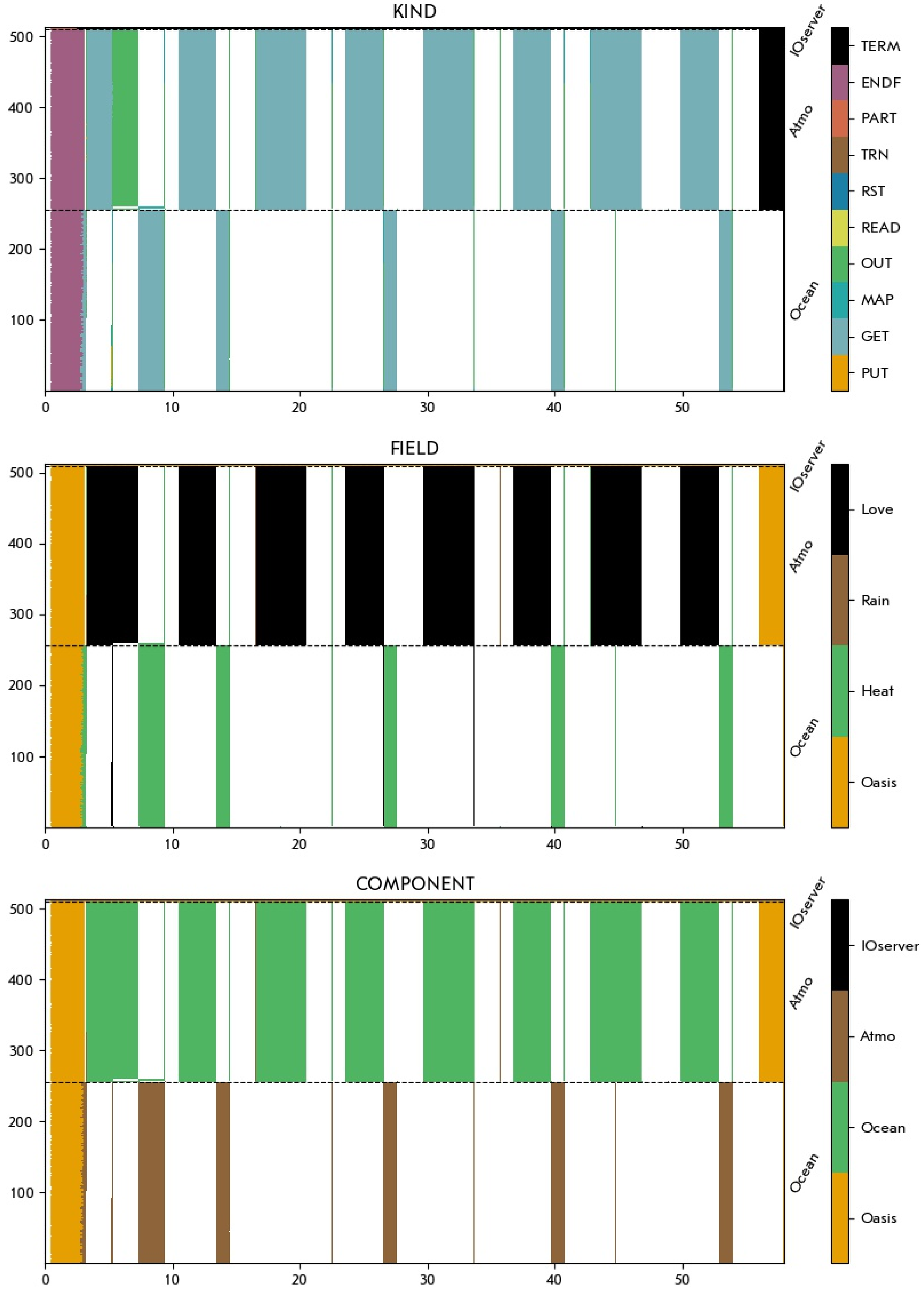
Appendix 1: `json` configuration file example

```
{
    "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"]
}
```

Appendix 2: `yaml` configuration file example


```yaml
---
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
```

Appendix 3: An OASIS timeline plot using `matplotlib`



References

[1] Séférian, R., Nabat, P., Michou, M., Saint-Martin, D., Voldoire, A., Colin, J., Decharme, B., Delire, C., Berthet, S., Chevallier, M. and Sénési, S., 2019: Evaluation of CNRM Earth System Model, CNRM-ESM2-1: Role of Earth System Processes in Present-Day and Future Climate. *Journal of Advances in Modeling Earth Systems*, *11*(12), pp.4182-4227

[2] Berthoud, F., Bzeznik, B., Gibelin, N., Laurens, M., Bonamy, C., et al., 2020: Estimation de l'empreinte carbone d'une heure.coeur de calcul. (in french), Rapport de recherche; UGA – Université Grenoble Alpes; CNRS; INP Grenoble; INRIA. 2020. hal-02549565v4

[3] Thompson, N. and Spanuth, S., 2018: The decline of computers as a general purpose technology: why deep learning and the end of Moore's Law are fragmenting computing, *Available at SSRN 3287769*

[4] Craig A., Valcke S., 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

[5] Maisonnave, E., Caubel, A., 2014: LUCIA, load balancing tool for OASIS coupled systems, Technical Report, TR/CMGC/14/63, SUC au CERFACS, URA CERFACS/CNRS No1875, France

[6] Maisonnave, E., Coquart, L. and Piacentini, A., 2020: A better diagnostic of the load imbalance in OASIS based coupled systems, TR/CMGC/20/176, SUC au CERFACS, URA CERFACS/CNRS No1875, France

[7] Guilyardi, E., 1996: The Vairmer Experiment Manager. User's Guide and Reference Manual, Technical Report, TR/CMGC/96/20, 102pp, CERFACS, France

[8] Hunter, J. D., 2007: Matplotlib: A 2D Graphics Environment, Computing in *Science & Engineering*, vol. 9, no. 3, pp. 90-95

[9] http://ferret.pmel.noaa.gov/Ferret

[10] Goedhart, J., 2019: Material related to the blog "Dataviz with Flying Colors". *Zenodo*. http://doi.org/10.5281/zenodo.3381072

[11] Ichihara, Y. G., Okabe, M., Iga, K., Tanaka, Y., Musha, K., & Ito, K., 2008: Color universal design: the selection of four easily distinguishable colors for all color vision types. In *Color Imaging XIII: Processing, Hardcopy, and Applications* (Vol. 6807, p. 68070O). International Society for Optics and Photonics