LUCIA,
load balancing tool
for OASIS coupled systems

Eric Maisonnave, Arnaud Caubel

TR-CMGC-14-63

Table of contents

This document focuses on how to optimize performances of an OASIS3-MCT-based coupled system, allocating to each model (coupled system component) an optimum number of resources (computing cores).

For this purpose, we developed the LUCIA (« Load-balancing Utility and Coupling Implementation Appraisal ») tool, which extracts from results of a coupled simulation all necessary information for a load balancing analysis, i.e. waiting time and calculation time of each system component.

Here will be described how works a coupled system and the various ways to optimize its performances using LUCIA metrics.

The authors appreciated the help of Uwe Fladrich and Martin Evaldsson (SHMI) during the first definition of the key concepts for a load-balancing analysis with OASIS-based systems.

1. Description of an OASIS3-MCT-based coupled system

1.1. What is a «coupled system» ?

We call numerical "**model**" (atmosphere, ocean, etc.) an ensemble of discretized equations which mathematically represent one from the several components of the climate system. A climate "**coupled system**" is the assembly of some of these numerical models, which can be independently developed. Representation of the exchanges occurring between components is called "**coupling**". Exchanged information ("**coupling fields**") is discretized on grids (meshes), which could be different from one model to another. It could be then necessary to interpolate information from source model grid to target model grid. Exchanges between coupled system models are periodic, with "**coupling time step**" periodicity. To be able to perform its own calculations, a model is using information coming at boundaries from another model: at the end of a coupling time step, the target model is waiting the information it needs to resume its calculations. The goal of the present document is to propose a method that allows to reduce as much as possible this waiting time, to speed up the whole coupled simulation duration.

Last version of the OASIS[1] (OASIS3-MCT) library allows to couple climate numerical models , that could have been developed independently. To set up such coupling, the developer has to modify the source codes, calling OASIS library subroutines on a so called "**OASIS interface**". This interface gathers different operations: initialization, MPI partitioning description, coupling field definition, coupling fields send and receive operations and termination. A coupling field can be sent to another model thanks to the OASIS_put subroutine, and received from another model with OASIS_get. Coupling characteristics for each field are defined in a parameter file called "namcouple". The MPI (Message Passing Interface) communication library ensures the exchange of numerical arrays that hold coupling fields.

OASIS interfaced models can be coupled following two different techniques: **sequentially** or **concurrently**. Generally speaking, when a model reaches the beginning of a coupling

[1] Sophie Valcke, Tony Craig, Laure Coquart, 2013: OASIS3-MCT User Guide, OASIS3-MCT 2.0, Technical Report, TR/CMGC/13/17, CERFACS/CNRS SUC URA No 1875, Toulouse, France

time step, it needs results of the other coupled model to resume its own calculations. If the results needed by both models are the results of the previous coupling time step, both model can run at the same time (concurrently). If one of the two models needs results of the current coupling time step, this model can be seen as a subroutine of the other one, and then models have to run sequentially (each model is waiting for the other one to be able to resume its calculations). Be careful that both sequential and concurrent modes can coexist if coupled system includes more than 2 models.

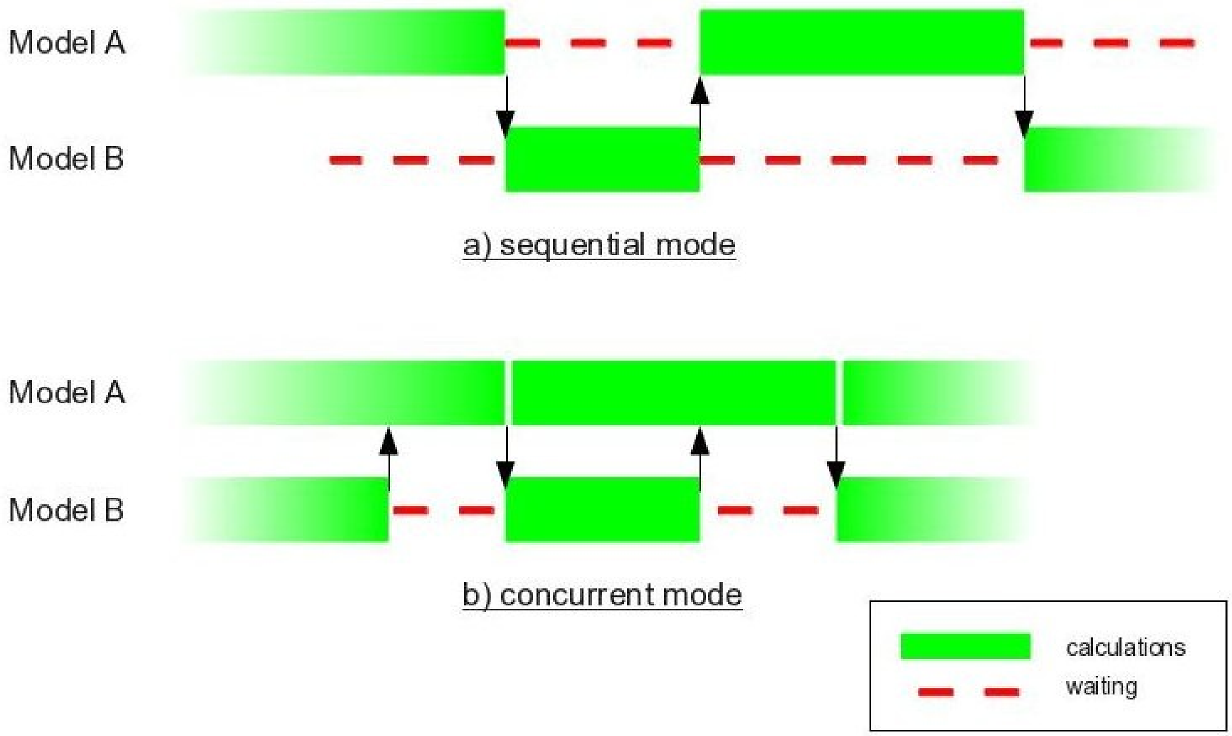

Fig 1 : Sequential, concurrent mode,
on a coupled system including 2 models

1.2. Coupling sequence

Model provides to OASIS coupled fields devoted to be used by one (or more) other(s) model(s). Discrete (gridded) values of those variables can be different on source and target grids: in this case, an interpolation is necessary to map the information from source model to target model. OASIS library, linked to the models, performs this interpolation operation :

- on source model, before sending the information to the target model
- on target model, which interpolates the coupled fields (received on the source grid) onto its own grid

OASIS parameter file (“namcouple”) has to be modified, for each coupling field, to specify on which side interpolation will take place (keywork “src” for source or “dst” for target, “MAPPING” option).

For a given model, a coupling time step can be decomposed as follows (not necessarily in the same order):

- the model performs its own calculations
- the model receives coupled variables from other models (via MPI communications done in Oasis_get subroutines)
- the model sends to other models the coupled variables it calculated (via MPI communications done in Oasis_put subroutines)
- the model performs interpolations if necessary before sending (“src”) or after receiving (“dst”) coupling fields.

1.3. LUCIA performance analysis tool

For each model, the LUCIA tool can measure time spent in each phase listed above. Clock time is measured during simulation, saved in log files and finally post-processed to provide clear and concise information. Clock time measures (via MPI_Wtime function) are done in OASIS routines before and after each coupling field exchange (send and receive operations) and before and after each interpolation, for each MPI process (if involved in coupling) of each model. A clock synchronization control is done at beginning, to detect a possible clock shift between nodes involved in the coupling. Initialization and termination phases are excluded from the measurements.

LUCIA is available in OASIS3-MCT from version 3.0 (or on trunk version, from revision 934, available on demand). Enabling of log file writing is done by setting keyword $NLOGPRT second argument to -1, in “namcouple” file. Log file names are «lucia.MM.PPPPPP», with MM: model ID in coupled system, PPPPPP: MPI rank in local model communicator.

When simulation stops[2], “lucia” script must be launched from the directory where log files were produced. This script calls a FORTRAN program (previously compiled with -c option of “lucia” script), which reads log files, process and displays on standard output (and info.dat ASCII file, that could feed gnuplot[3] software for a graphical output) the following quantities :

- ***En*** : time spent by model (n) sending and receiving MPI messages (we are talking about MPI messages involved in coupling field exchanges, excluding model internal parallel communications). More precisely, *En* measures the time spent between the beginning and the end of a message sending or receiving. This time encompasses every **communication** time. Since OASIS uses non blocking send (MPI_WAITALL + MPI_ISEND), the sending time is the time necessary to write messages into MPI buffer[4]. The receiving time encompasses the time spent to read messages in MPI buffer and the possible **load unbalance**[5] time between models: a model can have to wait for the other one ends its calculations before being able to send the

[2] LUCIA analysis can be performed during simulation, but, of course, results will only be based on coupling time steps already performed by the simulation at this time.

[3] http://www.gnuplot.info/

[4] MPI_WAITALL function is called before MPI_ISEND, which means that a message is sent only if the previous one has been received

[5] Load balancing concept is applied to models (a model can be faster or slower than another) and not, as usual in parallel programming, to processes of the same MPI executable (an MPI process is faster or slower than another).

requested information. This waiting time is often one order of magnitude bigger than MPI communication times. That is why *En* can be called **waiting time**. If model is MPI-parallel, *En* measures the time spent between the latest send and receive operations for all model MPI processes (and for all coupling fields). These extreme values are preferred to mean values to include jitter into calculation times (see next paragraph).

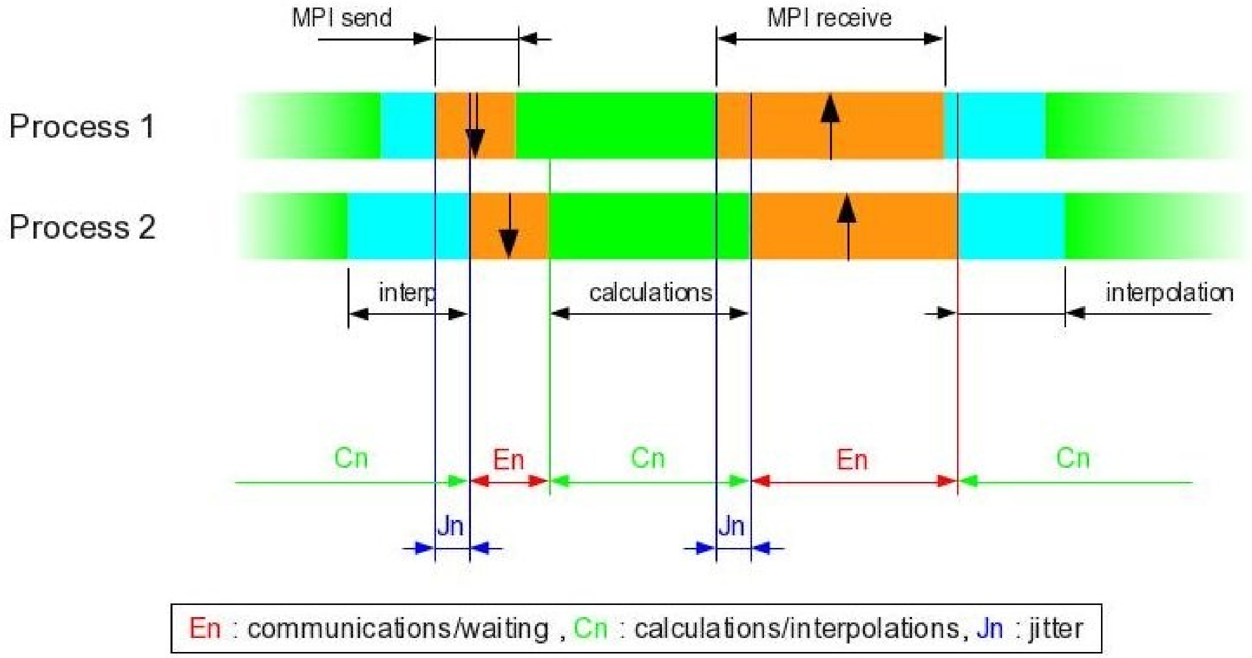

Figure 2 : En, Cn et Jn (jitter) definition, during send/receive operations, for 2 processes of an MPI parallel model (n)

- ***Cn* : the** time spent by model (n) to perform its own **calculations** and OASIS **interpolations**. This time is the complement to *En* time: *Cn +En* sum must be equal to the total simulation time used for the analysis[6]. *Cn* includes model calculation times but also, when model is parallel, the possible model internal unbalancing (so called **jitter**). Jitter is the adjustment time needed to wait the moment where all MPI processes are able to send or receive a coupling variable. This time can proceed from model itself (some calculations are more costly on some MPI sub-domains than on others) or more probably from OASIS interpolations because some processes could handle more sub-domains than others during interpolations (which increases load unbalance between processes and, then, jitter). Moreover, OASIS interpolation is the last operation before jitter measures: synchronization probably occurs before, that previously integrates jitter coming from other model calculations.

  We chose to include this jitter time in *Cn*, even though it is strongly dependent to coupling optimization, then directly linked to *En*.

Besides *En* et *Cn* quantities, LUCIA provides values of jitter and mean (over all model

[6] First of first two and last coupling steps are excluded from the measurement, to not taking into account possible biases due to initialization and ending phases.

processes) of OASIS interpolation times, for each model.

Figure 3a gives an example of graphical output: *Cn* (green) and *En* (red) are represented for the PULSATION[7] coupled system, composed by WRF atmosphere and NEMO ocean models, on concurrent mode coupling. One can notice that model with the slowest calculations (NEMO, 195 s) is actually not waiting at all. It is the consequence that WRF model, faster (130s), is providing its coupling fields before NEMO asks for them. In this case, WRF resources could be re-allocated to speed up NEMO (the slowest model) and, since this speed is equal to the slowest model speed, the whole system.

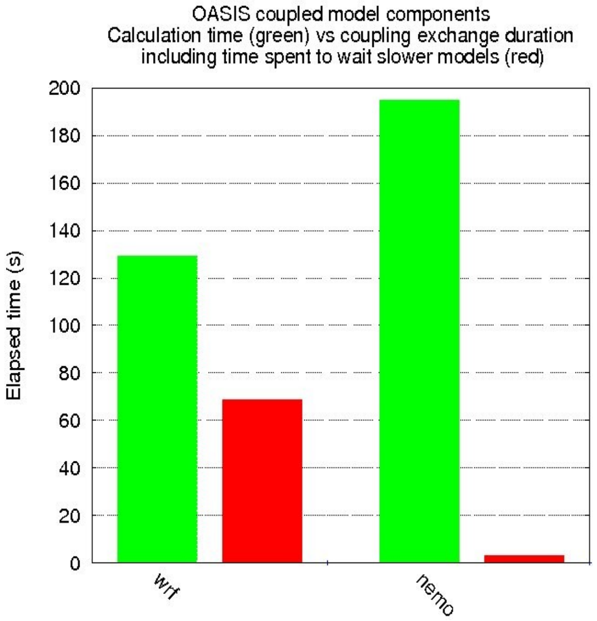

Figure 3a: LUCIA load balancing analysis of WRF-NEMO coupled system, concurrent mode

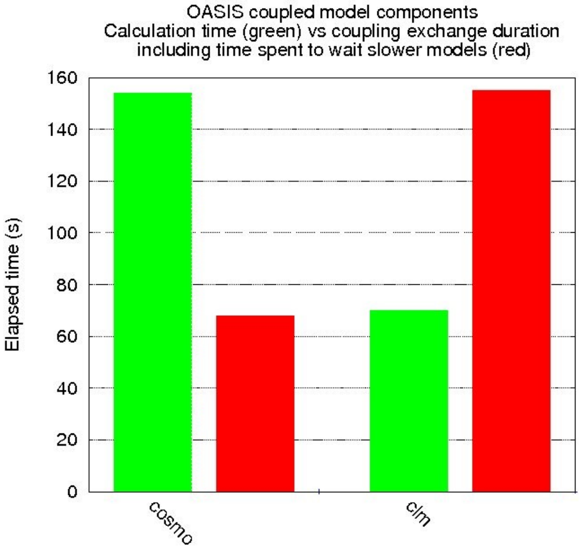

Figure 3b: LUCIA load balancing analysis of COSMO-CLM coupled system, sequential mode

On figure 3b,COSMO atmosphere and CLM (CESM) land surface models are coupled on sequential mode. In this case:

$$C_1 = E_2 \text{ ,}$$
$$C_2 = E_1$$

because, at coupling time step t, one of the two models (CLM) is waiting time step t calculation results of the other one (COSMO), see figure 1.

2. Analysis and optimization

To optimally allocate resources (CPU cores) to each model (component) of a coupled system is the main purpose of the LUCIA analysis tool. Analysis and optimization can be

[7] Masson, S., Hourdin, C., Benshila, R., Maisonnave, E., Meurdesoif, Y., Mazauric, C., Samson, G., Colas, F., Madec, G., Bourdallé-Badie, R., Valcke, S., Coquart, L., 2012: Tropical Channel NEMO-OASIS-WRF Coupled simulations at very high resolution , 11.4. 13th WRF Users' Workshop – 25-29 June 2012, Boulder, CO

done following two kind of strategies, depending whether your goal is:

- for a given resource number, performing the coupled system simulation as fast as possible.
- using system components at its optimum scalability (or parallel efficiency) and allocating as much resources as necessary for that.

The two corresponding methods are described in this document, for both sequential and concurrent modes. To simplify our explanation, we will assume that only 2 models are included in the coupled system, but the analysis can be done with any number of component.

2.1 Model load balancing, optimization for a given number of resources

To set a certain number of resources for the whole coupled system is the starting point of this first method. This number may be the maximum number of core available on the machine, or the optimal number that allows to perform short tests quickly, at implementation stage, or whole simulation at production stage (depending on the machine batch configuration strategy).

For sequential mode, there is no need of model load balancing for system performance optimization: since one model after the other is performing its calculations (see figure 1), the goal is simply to fully reduce each model calculation times, using method described at chapter 2.2.a. We will focus here to the concurrent mode only.

To start, a first simulation has to be launched with any number of resources. Then, we check *Cn* and *En* information that LUCIA provides (Cn_1 and En_1 for model 1, Cn_2 and En_2 for model 2).

If model 1 is the fastest, we get:

$$Cn_1 < Cn_2$$
$$En_2 = 0$$
$$En_1 > 0$$

In this case, we have to fully decrease En_1 waiting time, since model 1 is waiting data coming from the slowest model: this leads to speed up the slowest model. If total number of cores is constant, it also means that the fastest model has to be slow down. To do this, we have two choices:

- to reallocate CPU cores of the slowest model that were previously used by the fastest model. This operation must be iterated until

  $Cn_1 \sim Cn_2$ and
  En_1 et En_2 as small as possible

  The final coupled system configuration, regarding load balancing and resources usage, is also the fastest.

- to assign OASIS interpolation processing to the fastest model (see chapter 1.2), if it is not already the case. This strategy has poorer expected gain, since interpolation times are small[8] compared to the other computation times of the model. However, the interpolation can affect model jitter and then increase, in the modified model, its total restitution time.

Special cases

Namcouple option: EXPOUT

It seems better not to perform LUCIA analysis if “EXPOUT” mode (coupling fields exchanged at each coupling time step are output in NETCDF files) is enabled: file writing time is then added to *Cn* model calculation time. OASIS3-MCT file writing is not parallel, which means that master process only writes in output files the gathered arrays. This single-process-only operation has a non negligible cost, including jitter due to model parallelism perturbation.

Namcouple option: OUTPUT

With this option, the OASIS library can be used as a NETCDF output writing library. LUCIA cannot process coupling fields if not received by a model. Consequently, namcouple option “OUTPUT” is not compatible with a LUCIA analysis.

Non uniform values of coupling time step

LUCIA can process exchanges between models even though coupling fields of a given model are exchanged with different coupling time steps.

Coupled system of more than 2 components (with or without IO server)

LUCIA computes *En* and *Cn* values for a coupled system of 2 or more than 2 components. Be careful that coupling frequencies of exchanged fields of a given model can be different following which other model is involved in the coupling. Figure 4 shows an example involving 3 models and 1 IO server (xios.x, which is not involved in OASIS coupling). ARPEGE/NEMO coupling frequency for all exchanged fields is equal to 3 hours. For all other exchanges (ARPEGE/TRIP and TRIP/ARPEGE), it is equal to 1 day. For a 4 day long simulation, LUCIA analysis is performed over 3.75 days in ARPEGE and NEMO, but over 3 days only in TRIP, which explains that, in this particular example, *En*+*Cn* is not the same for all *n* system components.

[8] Additional information provided by LUCIA analysis (mean over all MPI model processes) gives a good idea of how much this time can be

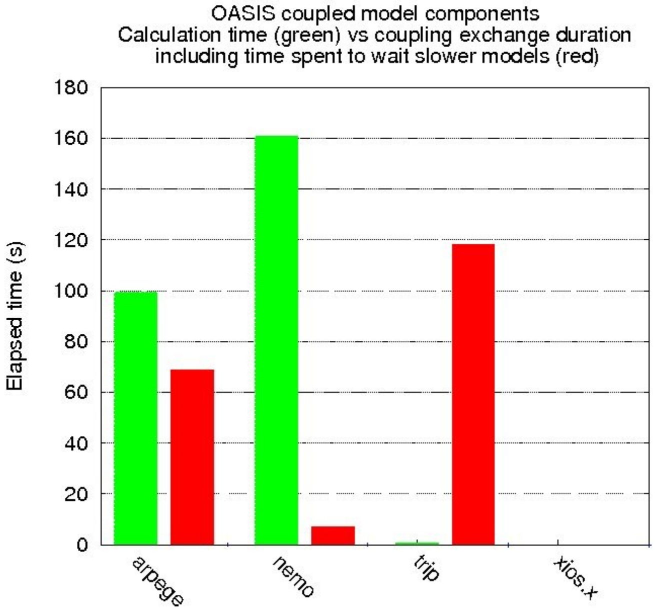

Figure 4 : LUCIA load balancing analysis for ARPEGE-TRIP-NEMO-XIOS coupled system

2.2 Optimal resource allocation (model speed or scalability/parallel efficiency)

While the first method leads to an optimal configuration for a given number of resources, this second approach gives the number of resources necessary to speed up the coupled system as much as possible (1st case) or reach the optimal scalability (or parallel efficiency) for each model (2nd case).

For both cases, user tests a range of possible number of core for each model, and LUCIA provides their corresponding *Cn* times: scalability or parallel efficiency can be deduced from those times measurements and curves plotted (one per model). For example, from a small initial number of resources, one can double several times this number to reach the **scalability limit** of each model: this limit is reached when model restitution time does not decrease while resource number allocated still increases. In figure 5 (logarithmic x-axis), scalability curves of NEMO and WRF models, components of ocean-atmosphere coupled system, are plotted. NEMO model reaches this limit around 2,000 CPU cores, WRF model around 16,000 CPU cores.

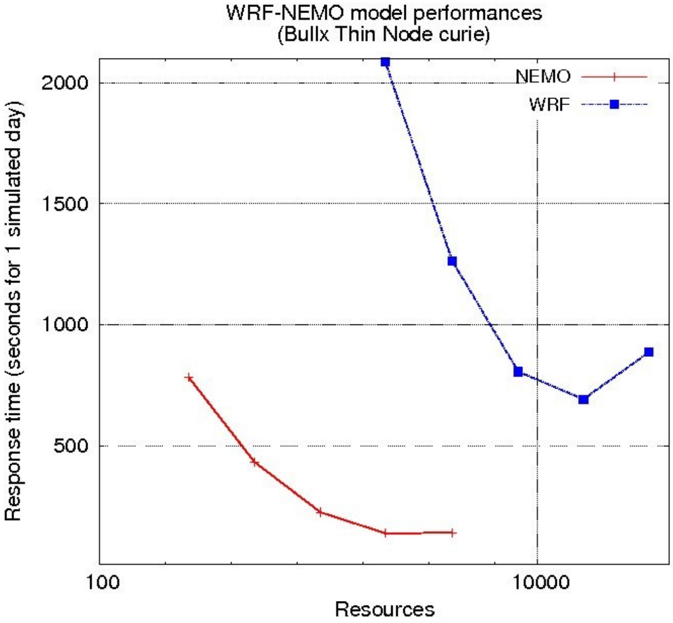

Figure 5 : Model scalabilities of WRF and NEMO model as part of a coupled system

Please notice that it is as part of a particular coupled system that model performances are measured here. These scalabilities can differ from the one measured for a stand alone model, or the one measured for the same model but included in another coupled system. Our method has two advantages: (i) all coupled system model scalabilities are measured at the same time and (ii) the measured scalability is the exact scalability each model exhibits during coupled system production phase.

<u>2.2.a. Speed</u>

For sequential mode (each model waits results for other model calculations), it is possible, for each model independently, to determine from their respective scalability curves how to set the fastest configuration, and then deduce the total resources needed for the whole system.

For concurrent mode, the maximum speed of coupled system will be approximately the same than the “slowest” model one. When all model scalability limits are determined, the “slowest” model is the model which has the longest restitution time at scalability limit. We allocate to the model as much resources as we need to reach this limit. Then, we allocate to the other model of the coupled system as much resources as necessary to go as fast as the “slowest” model (or slightly faster[9]), considering that a higher number of resource is useless since it cannot increase the whole system speed. With this load balancing set up, the coupled system is pushed at its maximum speed, using no more than the needed number of resources for each of its components.

On figure 5, we show for example that WRF model (at scalability limit of 16,000 cores) is slower than NEMO model (at scalability limit of 2,000 cores). To go at the same speed

[9] When two components of a coupled system are performing a coupled time step at the same speed, coupling operations at the end of the coupling step can interfere together and lead to a slight slow down, so that the coupled system speed is slightly smaller than model speeds as measured in a load unbalanced coupled configuration (where one component goes faster than the others)

than WRF, NEMO needs about 512 CPU cores. The optimum resource distribution will be 16,000 cores for WRF and 512 cores for NEMO.

2.2.b. Scalability / Parallel efficiency

It quite often happens that to work at scalability limit leads to a waste of resources. Because, most of the time, the model is far from the perfect scalability when this limit is reached: even with a limited number of resource, allocating more resources does not lead to a proportional speed up of the simulation, but much less. Consequently, for resource saving reasons, one could consider that it does not worth to push the model at scalability limit, even though coupled system speed is slightly reduced.

To more easily find this new limit, we better use scalability derivative (parallel efficiency). Instead of restitution time *T* versus resources number *R*, we plot parallel efficiency *E* as:

$$E = (T_1 * R_1) / (R * T)$$

with T_1 restitution time for a minimal number of resource R_1 (1, ideally) required by that model 1.

By convention, a model with no more resources than the ones needed to exhibit a parallel efficiency of more than 1/2 is considered as acceptable.

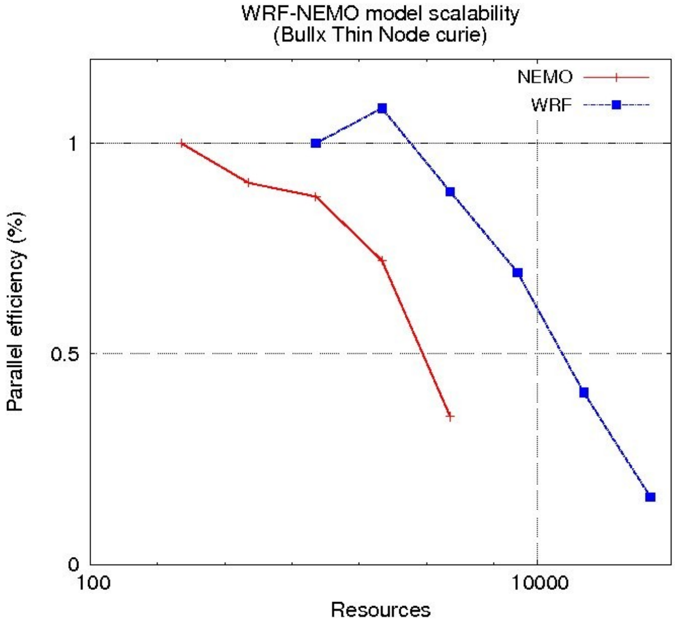

Figure 6 : WRF and NEMO model parallel efficiency on coupled mode

When coupling is sequential, scalability (or parallel efficiency) curves are used to choose, for each model independently, how much resources are needed to reach the appropriate levels of scalability (or parallel efficiency).

When coupling is concurrent, instead of scalability limit, parallel efficiency is considered (for example ½). When all model parallel efficiency limits are determined, the “slowest” model is the model which has the longest restitution time at parallel efficiency limit. Resources for each model are attributed accordingly to this first number.

On figure 6, we show for example parallel efficiencies of WRF/NEMO coupled system.Value equal to ½ is reached for the slowest model (WRF) after 10,000 CPU cores. To run NEMO at the same speed, 256 CPU cores are necessary (this information can be deduced from figure 5). The optimum resource distribution is 10,000 cores for WRF and 256 cores for NEMO. Coupled system total resources are smaller than on 2.2.a paragraph case (38%), for a slow down of only 7%.

3. Conclusion

The LUCIA tool was developed to measure differential performances within an OASIS-based coupled system. Thanks to it, it is possible to measure how much time each system component is spending doing its own calculation and how much time it is waiting for information coming from the other components.

LUCIA cannot replace more comprehensive profiling tools (such as « vampirtrace », « paraver », « vtune », etc.). Those tools, exhaustively analyzing MPI communications of the whole system (OASIS communications + models parallel communications) are able to more precisely identify slow down that can occur on a given system. However, LUCIA gathers some advantages compared to these tools :

- ease-of-use: coupled models do not have to be instrumented at compiling stage; measurements are done through a simple namelist modification
- robustness: MPI MPMD mode, mandatory for OASIS, is not always supported by standard profiling tools
- succinctness: only 4 quantities, integrated over all model processes, are provided for user analysis

LUCIA helps you to optimize coupled system performances, saving computational time and/or resources. The analysis results could also be used for more advanced optimizations, like process mapping on selected machine cores or nodes.

4. Example

Figure 7 shows an example of the use of LUCIA to study the gain of OpenMP parallelisation on IPSL atmospheric model LMDZ. LMDZ is hybrid parallelized (MPI and OpenMP) and we test the benefit of the use of OpenMP in the IPSL Earth System model IPSLCM6, i.e LMDZ atmospheric model coupled with NEMO as oceanic component. IPSLCM6 configuration is tested at LMD144x142x59(LMDZ)-ORCA2(NEMO) resolution. Thanks to graphical format provided by LUCIA, we can see easily that the activation of OpenMP allows to reduce the computing time of LMDZ component and, since NEMO computing time remains unchanged, the waiting time of NEMO component in this coupled system.

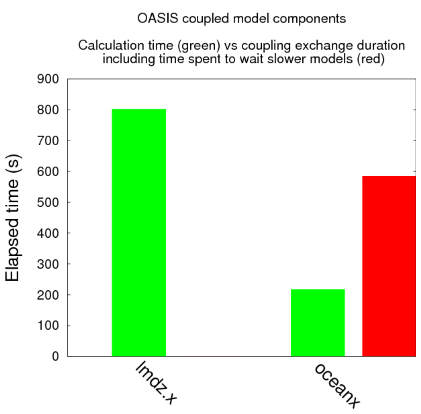	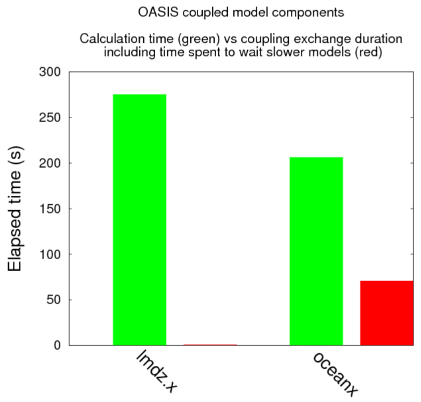	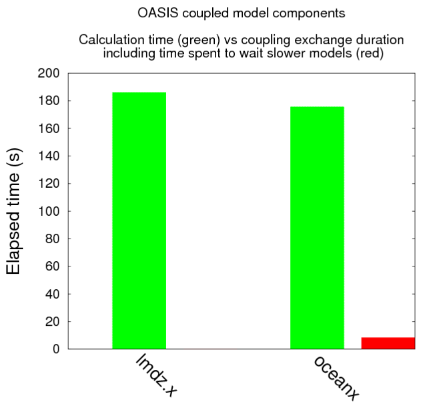
Figure 7a *47 LMDZ MPI x* ***1 OMP thread*** *4 NEMO MPI*	*Figure 7b* *47 LMDZ MPI x* ***4 OMP threads*** *4 NEMO MPI*	*Figure 7c* *47 LMDZ MPI x* ***8 OMP threads*** *4 NEMO MPI*

Appendix: Instructions for use

```
=======================================================

This tool was developpped by Eric Maisonnave (CERFACS),
Uwe Fladrich, Martin Evaldsson (SMHI) and Arnaud Caubel (IPSL)
to perform an analysis of the coupled components load balance

        v1.0 :              12/2013

=======================================================

1. Compilation
--------------

In $(OASIS_DIR)/oasis3-mct/util/lucia, compile main-lucia.F90 :
* compile using the command : lucia -c
* if your compiler is not automatically detected,
  specify your compiler by modifying "F90=my_compiler" in
  "lucia" script file
* executable file "lucia.exe" is created in the same directory


2. Simulation set up
--------------------

Before launching your OASIS coupled model, modify your "namcouple"
file: the second number on the line below $NLOGPRT must be set to
-1. This option enables the production of OASIS-LUCIA log files,
named "lucia.MM.PPPPP", with "MM" executable number and "PPPPP"
MPI process number in local communicator. It is not possible to
produce timer log files at the same time that lucia log files.


3. Post processing
------------------

In your results directory (where executable and model output are
located), post-treat the files produced by OASIS-LUCIA log files:
$(OASIS_DIR)/oasis3-mct/util/lucia/lucia

This command will post-treat the OASIS-LUCIA log files located in
the directory using lucia.exe


4. Analysis
-----------

Several information related to the coupled simulation are provided
on standard output:
```

* Name and number of "lucia.MM.PPPPP" processed
For performance reasons (ASCII file reading), LUCIA do not process all OASIS-LUCIA log files, but only a subset, displayed below the comment line "Computed log files for model MM"

* Coupling field names + model exchanging them, ordered by exchange date
This information is displayed below the comment line "Exchanged fields (based on first exchange)"

* Load balance
For each model, LUCIA gives the total time spent during calculations, the total time spent to wait information from OASIS and the number of coupling time step used to calculate those values.
Time is in seconds. Information is displayed below the comment line "Component -   Calculations   -   Waiting time (s) - # cpl step "

* Additional information
LUCIA also provides, for each model, the total time spent to perform OASIS interpolations during simulation, and total process jitter measured at each OASIS send/receive steps.
This information is displayed below the comment line "Additional informations".

Load balance information is also provided on graphical format (using gnuplot, if available) in oasis_balance.eps file.