Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Computational coupling cost evaluation

Challenges and solutions for the new generation of Earth System Models

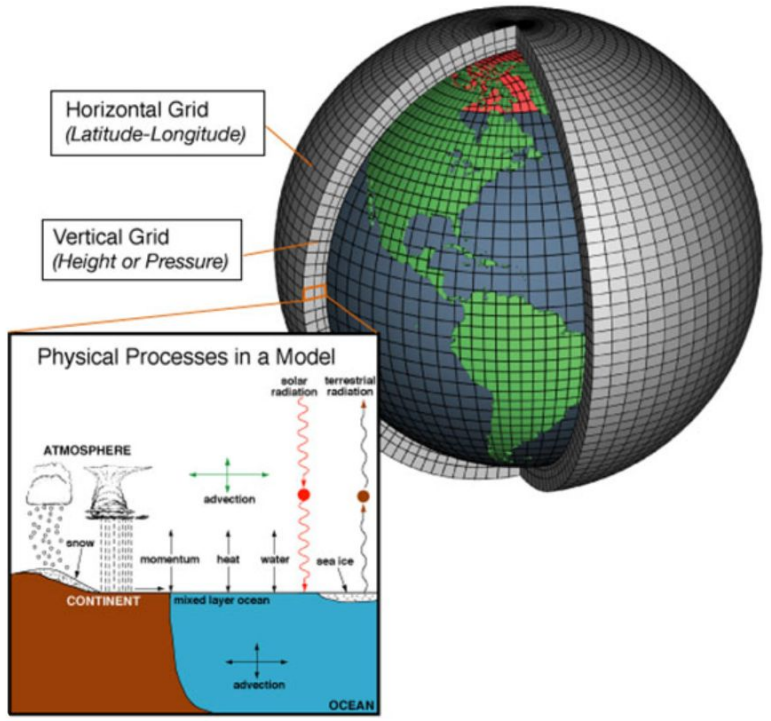
Mario C. Acosta Sergi Palomas

22-09-2020

Context

An Earth System Model (ESM) in an example of couple model application

- Multi-physics simulations based on two or more computing applications
- Involves several interacting components simulating the atmosphere, oceans, land, sea ice…
- Coupling data must be interpolated (regridded) subject to different constraints such as conservation
- Include data transfer among sub-domains of one component and/or components
- Coordination among components is needed and faster components could wait to others for synchronization of exchanged fields.

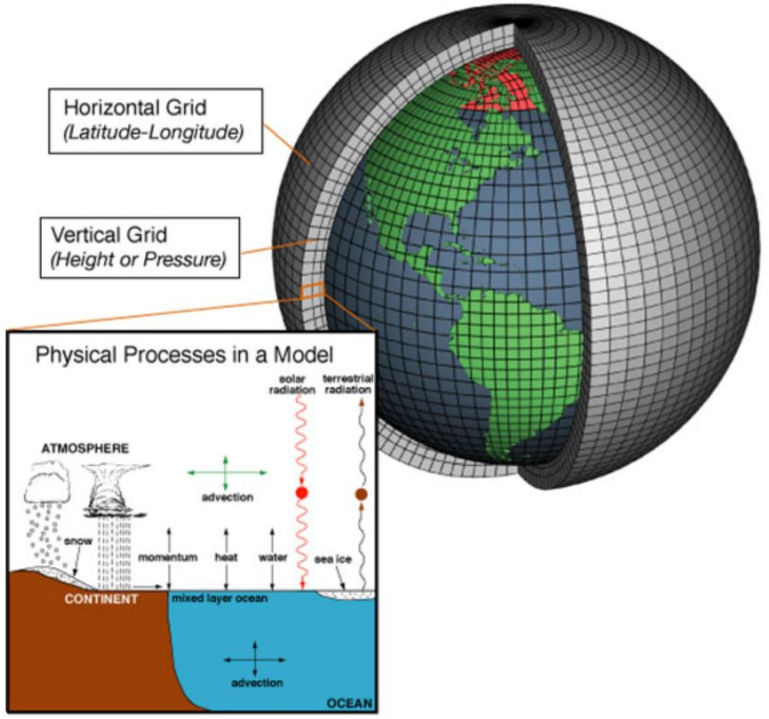


http://celebrating200years.noaa.gov/breakthroughs/climate_model/welcome.html

Context

An Earth System Model (ESM) in an example of couple model application

- Multi-physics simulations based on two or more computing applications
- Involves several interacting components simulating the atmosphere, oceans, land, sea ice…
- **Coupling data must be interpolated (regridded) subject to different constraints such as conservation**
- **Include data transfer among sub-domains of one component and/or components**
- **Coordination among components is needed and faster components could wait to others for synchronization of exchanged fields.**



http://celebrating200years.noaa.gov/breakthroughs/climate_model/welcome.html

Context: Computational point of view

An Earth System Model (ESM) in an example of couple model application

- **Coupling data must be interpolated (regridded) subject to different constraints such as conservation**
  - Extra computation is needed for the interpolations.
  - Some constraints could affect to the performance of the model itself. For example, the conservation could require serialization techniques, reducing the parallel efficiency.
- **Include data transfer among sub-domains of one component and/or components**
  - Communications between components are needed depending on the paradigm used (MPI communications, synchronization points…)
- **Coordination among components is needed and faster components could wait to others for synchronization of exchanged fields.**
  - Depend on the set up. Although many times is negligible, some waiting time could be impossible to avoid and achieve the minimum waiting time could not be trivial.

Context: Computational point of view

An Earth System Model (ESM) in an example of couple model application

- **Coupling data must be interpolated (regridded) subject to different constraints such as conservation**
  - Extra computation is needed for the interpolations.
  - Some constraints could affect to the performance of the model itself. For example, the conservation could require serialization techniques, reducing the parallel efficiency.
- **Include data transfer among sub-domains of one component and/or components**
  - Communications between components are needed depending on the paradigm used (MPI communications, synchronization points…)
- **Coordination among components is needed and faster components could wait to others for synchronization of exchanged fields.**
  - Depend on the set up. Although many times is negligible, some waiting time could be impossible to avoid and achieve the minimum waiting time could not be trivial.

- Are our models/workflow ready to evaluate the computational efficiency of the coupled implementation?

Context: Computational point of view

An Earth System Model (ESM) in an example of couple model application

- **Coupling data must be interpolated (regridded) subject to different constraints such as conservation**
  - Extra computation is needed for the interpolations.
  - Some constraints could affect to the performance of the model itself. For example, the conservation could require serialization techniques, reducing the parallel efficiency.
- **Include data transfer among sub-domains of one component and/or components**
  - Communications between components are needed depending on the paradigm used (MPI communications, synchronization points…)
- **Coordination among components is needed and faster components could wait to others for synchronization of exchanged fields.**
  - Depend on the set up. Although many times is negligible, some waiting time could be impossible to avoid and achieve the minimum waiting time could not be trivial.

- Are our models/workflow ready to evaluate the computational efficiency of the coupled implementation?
- Are we 100% that the set-up chosen is the best one in terms of efficiency?

Context: Computational point of view

An Earth System Model (ESM) in an example of couple model application

- **Coupling data must be interpolated (regridded) subject to different constraints such as conservation**
  - Extra computation is needed for the interpolations.
  - Some constraints could affect to the performance of the model itself. For example, the conservation could require serialization techniques, reducing the parallel efficiency.
- **Include data transfer among sub-domains of one component and/or components**
  - Communications between components are needed depending on the paradigm used (MPI communications, synchronization points…)
- **Coordination among components is needed and faster components could wait to others for synchronization of exchanged fields.**
  - Depend on the set up. Although many times is negligible, some waiting time could be impossible to avoid and achieve the minimum waiting time could not be trivial.

- We are moving to more complex ESMs with very complex set-ups…
  - Are we sure that for all those new situations the methodology for analysis is still valid?

Context: Computational point of view

An Earth System Model (ESM) in an example of couple model application

- **Coupling data must be interpolated (regridded) subject to different constraints such as conservation**
  - Extra computation is needed for the interpolations.
  - Some constraints could affect to the performance of the model itself. For example, the conservation could require serialization techniques, reducing the parallel efficiency.
- **Include data transfer among sub-domains of one component and/or components**
  - Communications between components are needed depending on the paradigm used (MPI communications, synchronization points…)
- **Coordination among components is needed and faster components could wait to others for synchronization of exchanged fields.**
  - Depend on the set up. Although many times is negligible, some waiting time could be impossible to avoid and achieve the minimum waiting time could not be trivial.

- We are moving to more complex ESMs with very complex set-ups…
  - Are we sure that for all those new situations the methodology for analysis is still valid?
  - Are we sure that it is something that we want to do manually?

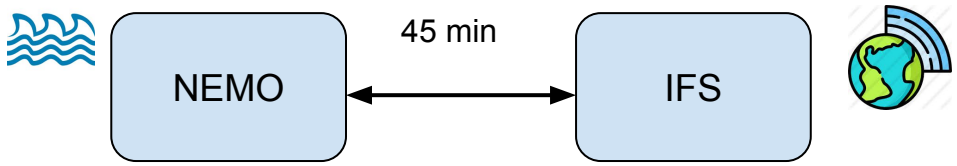
Main goals

- Define a new **common set of performance metrics** to evaluate coupled models → Lucia/OASIS3-MCT
- **Reduce the execution cost** of each component and an Earth System Model (ESM) as a whole
- Establish a **method to balance** any given experiment (models used, coupling settings, resolution, output, etc.)

Context

In EC-Earth we are currently using up to 6 components. Therefore, the complexity to use the computing resources effectively has also grown.

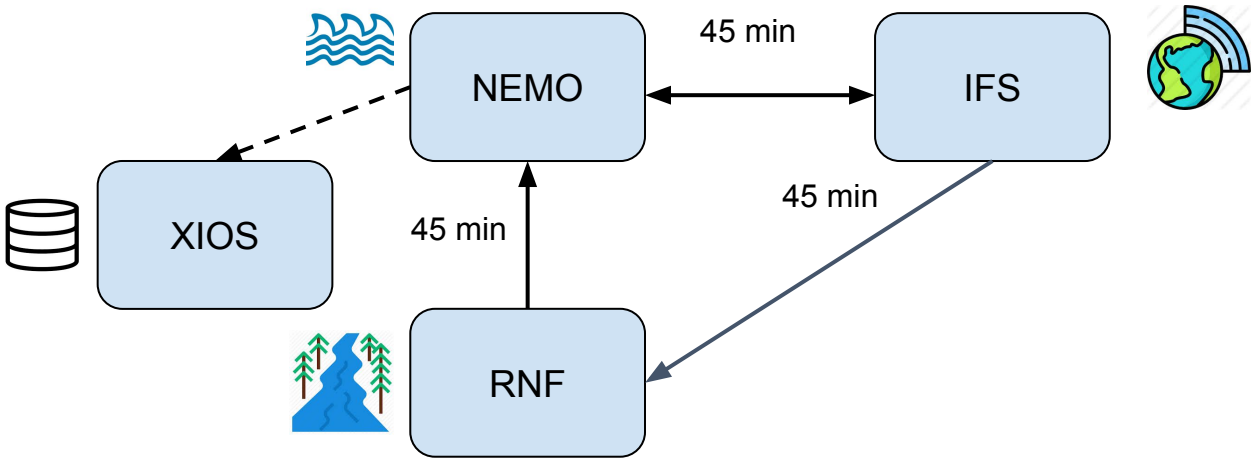
- Communication: Increase in the number of fields to exchange (over 200 every 6h)
- Computation: More interpolations which add an overhead, including some conservatives.
- Coordination: 6 components using different time-step and time for exchanging, all of them are blocking exchanges



Context

In EC-Earth we are currently using up to 6 components. Therefore, the complexity to use the computing resources effectively has also grown.

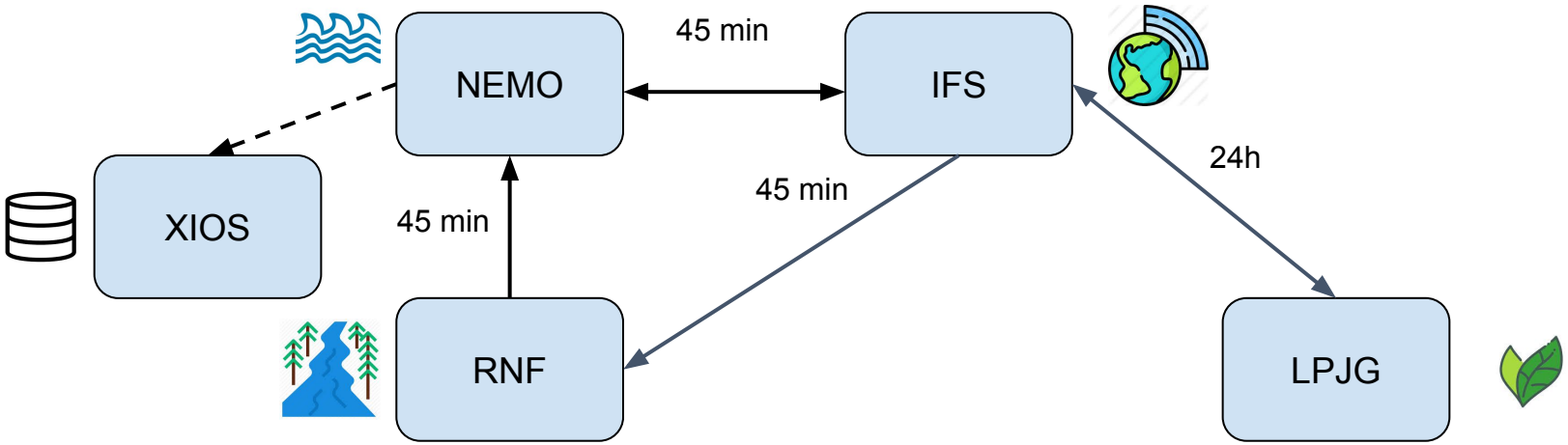
- Communication: Increase in the number of fields to exchange (over 200 every 6h)
- Computation: More interpolations which add an overhead, including some conservatives.
- Coordination: 6 components using different time-step and time for exchanging, all of them are blocking exchanges



Context

In EC-Earth we are currently using up to 6 components. Therefore, the complexity to use the computing resources effectively has also grown.

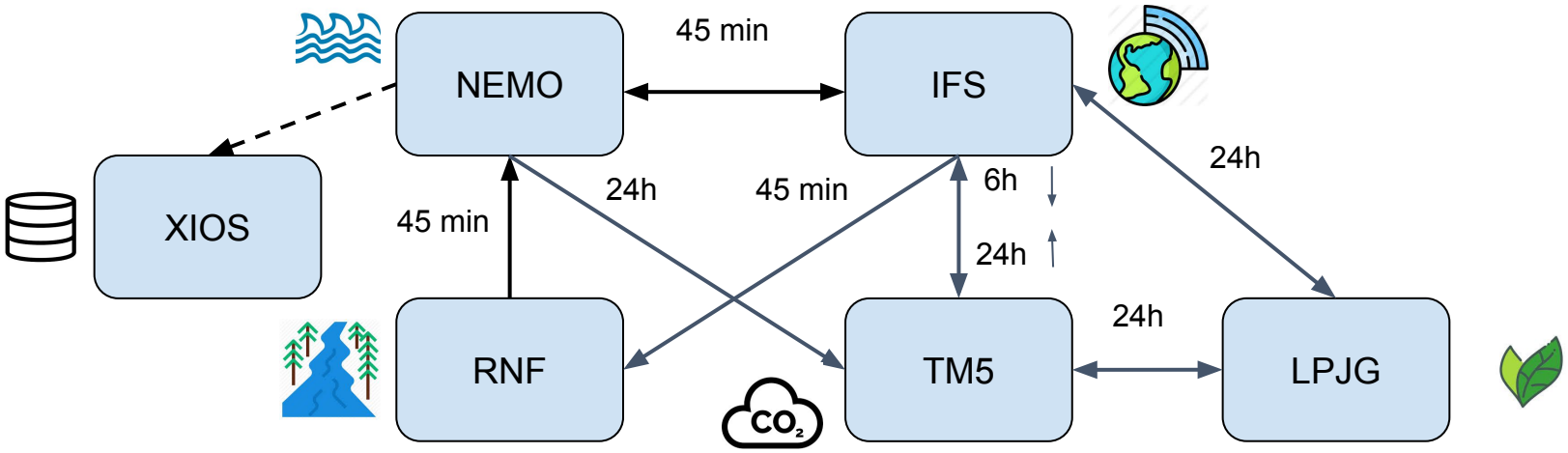
- Communication: Increase in the number of fields to exchange (over 200 every 6h)
- Computation: More interpolations which add an overhead, including some conservatives.
- Coordination: 6 components using different time-step and time for exchanging, all of them are blocking exchanges



Context

In EC-Earth we are currently using up to 6 components. Therefore, the complexity to use the computing resources effectively has also grown.

- Communication: Increase in the number of fields to exchange (over 200 every 6h)
- Computation: More interpolations which add an overhead, including some conservatives.
- Coordination: 6 components using different time-step and time for exchanging, all of them are blocking exchanges



Where each component has its own:

- Scalability (efficiency) curves
- Resolution
- Output config
- Parameters that affect the performance of the component (e.g online diagnostics)
- Irregular timestep lengths

LUCIA

Lucia has been the standard post-processing tool used to get performance metrics when using the **OASIS-MCT** coupler.

- Developed by CERFACS in 2014

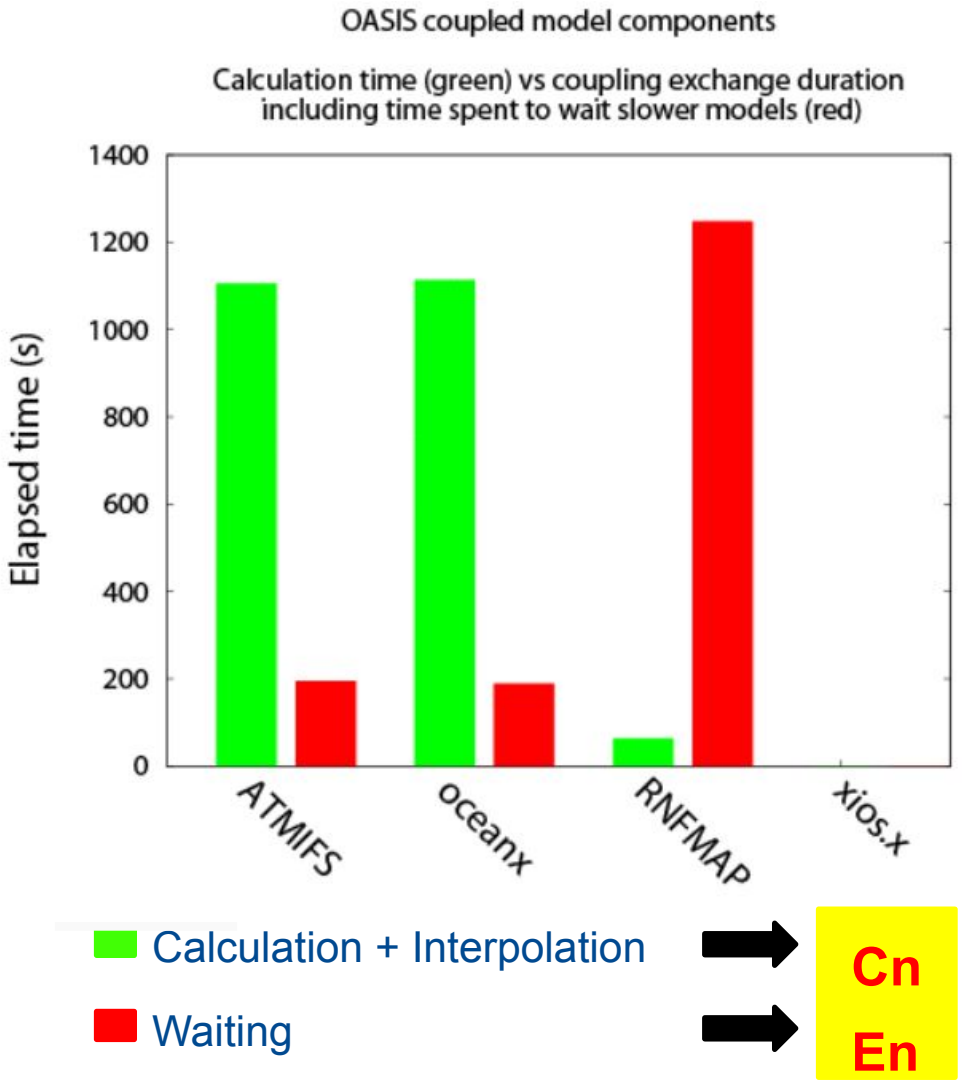
- Provides 3 metrics for each component:

    - En: Waiting time (MPI_Get)
    - Cn: Model computation + Interpolation time
    - Jitter: time separating the first and the last process when starts a coupling event (Min/Max)

LUCIA

Lucia has been the standard post processing tool used to get performance metrics when using the OASIS-MCT coupler.

```
Load balance analysis

Component -         Calculations    -     Waiting time (s) - # cpl step :

xios.x                      0.00                    0.00            0
oceanx                    183.64                  176.46          989
ATMIFS                    358.69                    0.18          989
RNFMAP                     16.77                  342.11          989

Additional informations
Component -  OASIS mean interpolation time -    Jitter  (s):
xios.x                    0.00                      0.00
oceanx                    0.35                      5.76
ATMIFS                    2.55                      4.50
RNFMAP                   13.25                      0.00
```



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Metrics

Currently

For each component:

- Cn: Time spent in calculations (model + interpolations)
- En: Time waiting
- Jitter: Imbalance of the processes (max-min time to start a send or receive)

Method: minimize the En (normally, trying to make all components run at the same SYPD)

Objective

For each component:

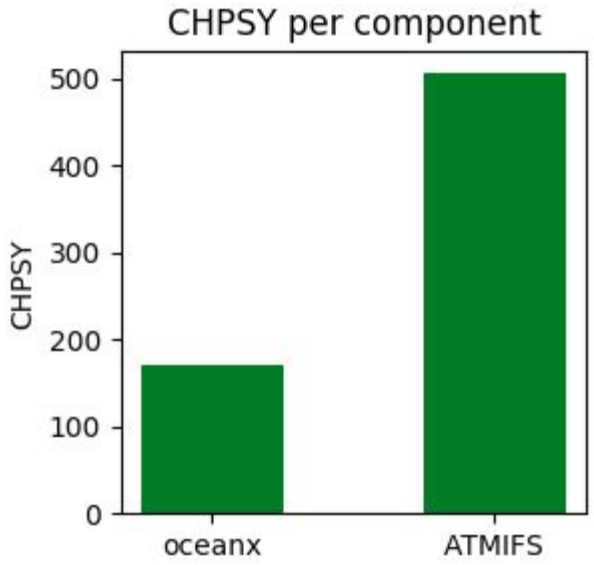
- SYPD and CHPSY
- Coupling cost (Balaji et al.)
- Waiting, Sending, Interpolation and component time
- Visualization of the execution pattern per coupling step
- Dependencies between components

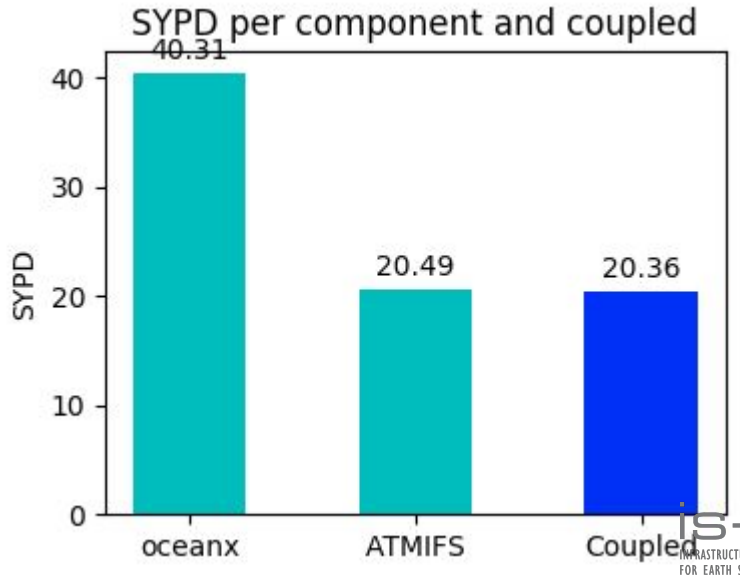
Method: Minimize the waiting time while having the best number of processes for each component (Efficiency/TTS)

TTS: Time to solution

Lucia lite

The SYPD and CHPSY. Useful to get an idea about how well each component is running respect to the resources given.

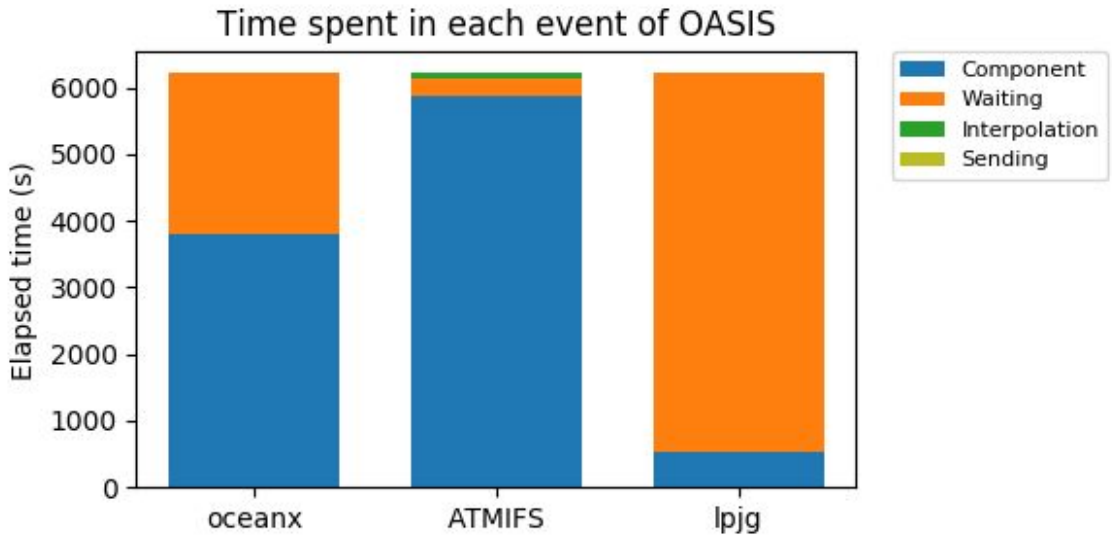




Lucia lite

The SYPD and CHPSY. Useful to get an idea about how well each component is running respect to the resources given.

The time spent for component calculations and for each coupling event (Send, Receive and Interpolation)

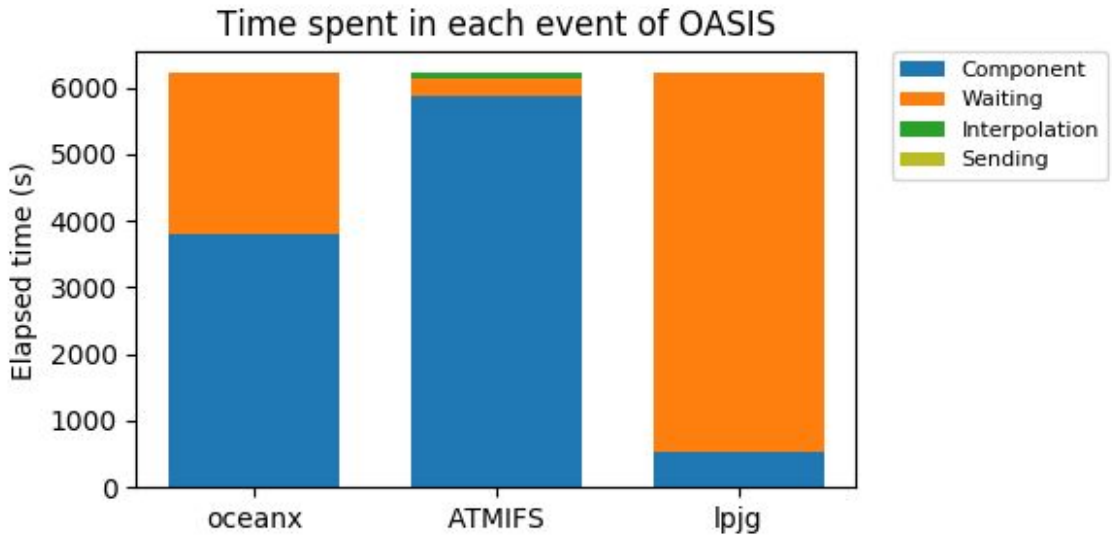


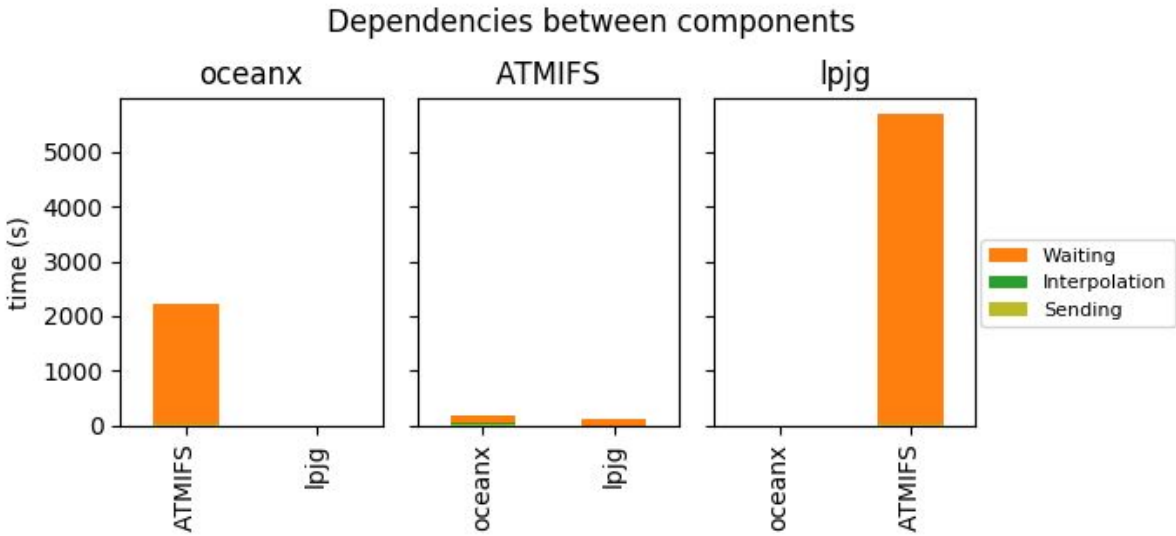
Lucia lite

The SYPD and CHPSY. Useful to get an idea of how well each component is doing with the resources given.

The time spent for component calculations and for each coupling event (Send, Receive and Interpolation)

The dependencies between the components involved in the coupling.



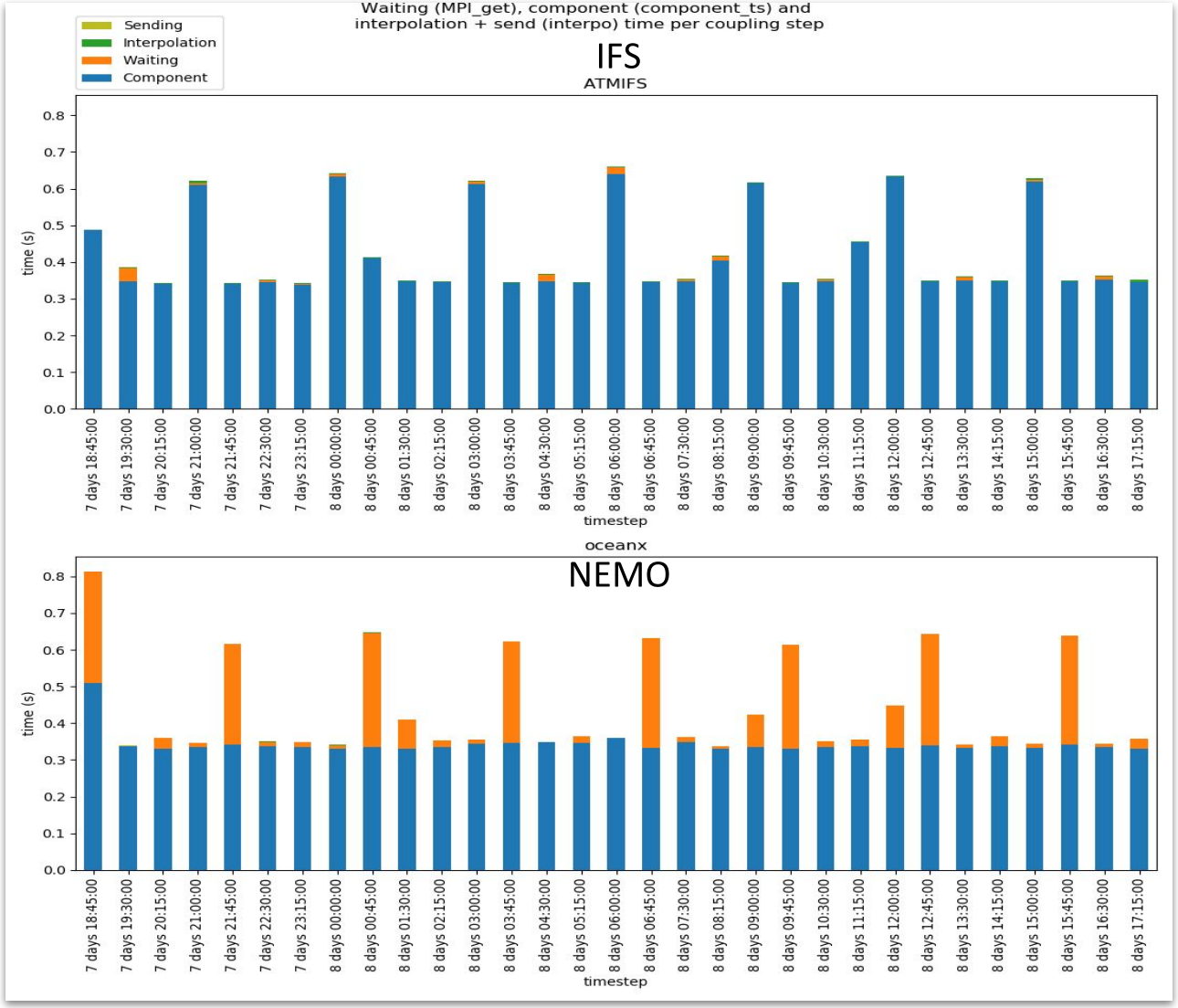


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Lucia lite

Information per coupling step, where:

- A coupling interval (CI) is the interval between consecutive coupling regions of a model
- Waiting/Sending/Interpolation is the time spent in all MPI_Get/MPI_Put/Interpolate routines during a CI
- Component time is the time in a CI which outside Oasis routines



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Lucia lite: Coupling Cost

The **coupling cost (C)** is the standard metric (CPMIP, Balaji et al, 2017) to measure the percentage of the overhead caused by coupling. This includes the waiting time due to unbalanced components and cost of the coupling algorithm itself.

$$C \equiv \frac{T_M P_M - \sum_c T_C P_C}{T_M P_M}$$

$\mathbf{T_M}$ and $\mathbf{P_M}$ are the runtime and number of resources used for the whole model

And $\mathbf{T_C}$ and $\mathbf{P_C}$ for each component

Additionally, we can separate the coupling cost per component. This partial coupling cost (PCC) with the following formula:

$$PCC \equiv \frac{T_{Ccpl} P_C}{T_M P_M}$$

$\mathbf{T_{Ccpl}}$ is the total time spent in coupling by a component (i.e. waiting + send + interpolation time)

LUCIA Lite: Balancing components

Each component has its own scalability (and efficiency) curve. As we add more of them, achieving a configuration where all of them are balanced gets more difficult

Irregular timesteps make impossible to achieve a perfect balance between components and it is not possible to detect with accumulative or average metrics.

Different coupling frequencies make things even harder

Every time we add a new component or the set-up change...

Brief coupling case studies

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

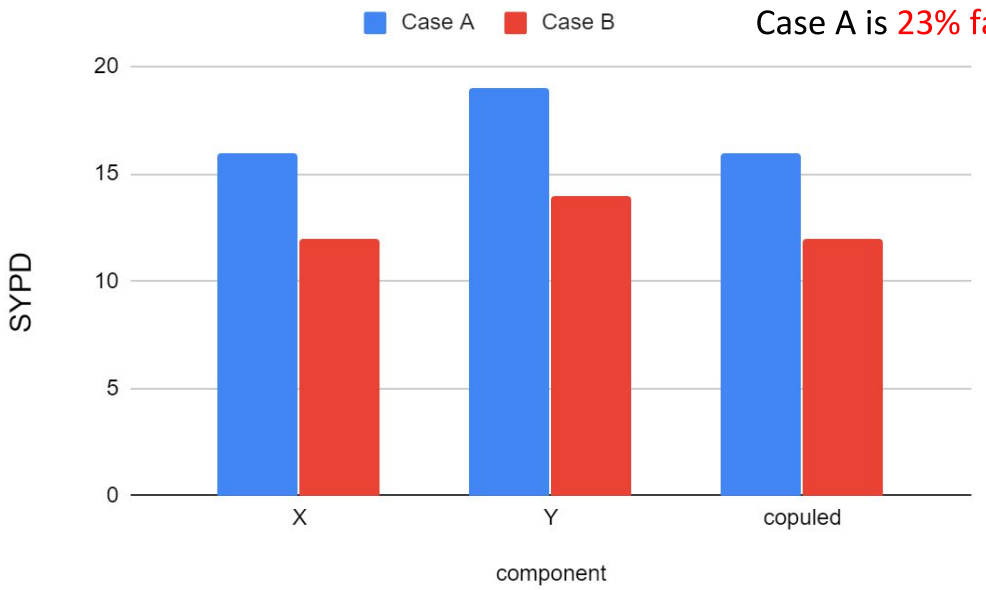
Brief coupling case studies

SYPD

CHPSY

SYPD

These metrics provide important information about the performance of each component (CMIP). They are used to choose the appropriate number of resources to use.

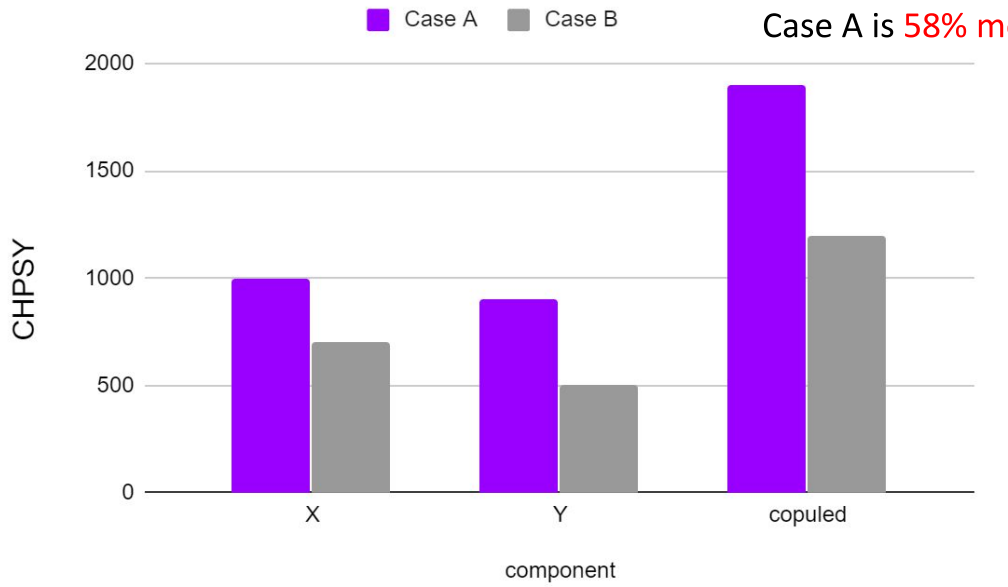


Case A is 23% faster

Is it preferable configuration A or B?

CHPSY

These metrics provide important information about the performance of each component (CMIP). They are used to choose the appropriate number of resources to use.



Case A is 58% more costly

Is it preferable configuration A or B?

Case A if we apply an **strict** TTS

Case B is better otherwise

Brief coupling case studies

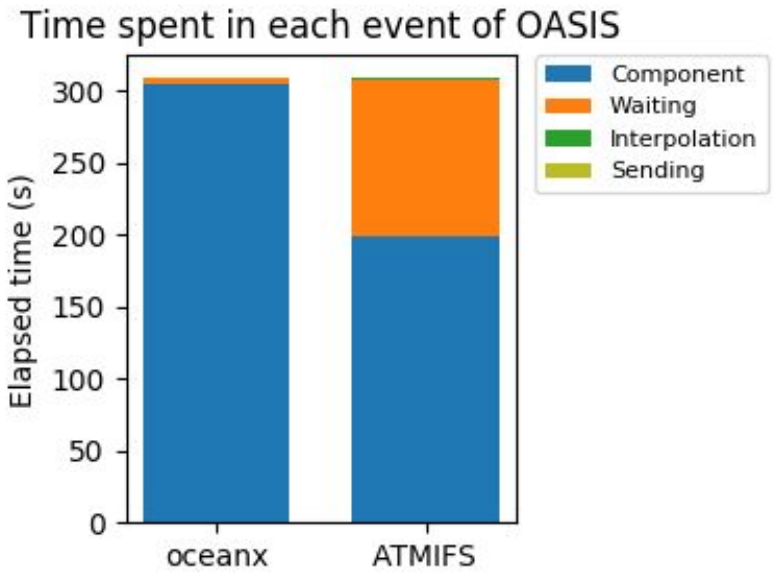
Coupling cost

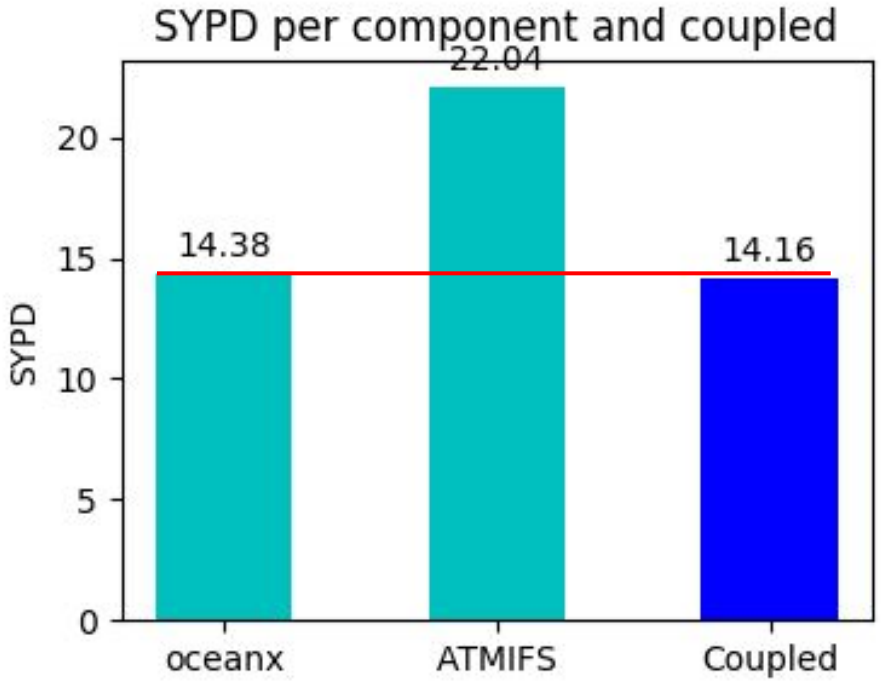
Waiting + Sending + Interpolation

The pattern

Balancing IFS and NEMO

- IFS is waiting for NEMO ⟶ NEMO is slower
- Coupled execution as slow as the slowest component (NEMO)

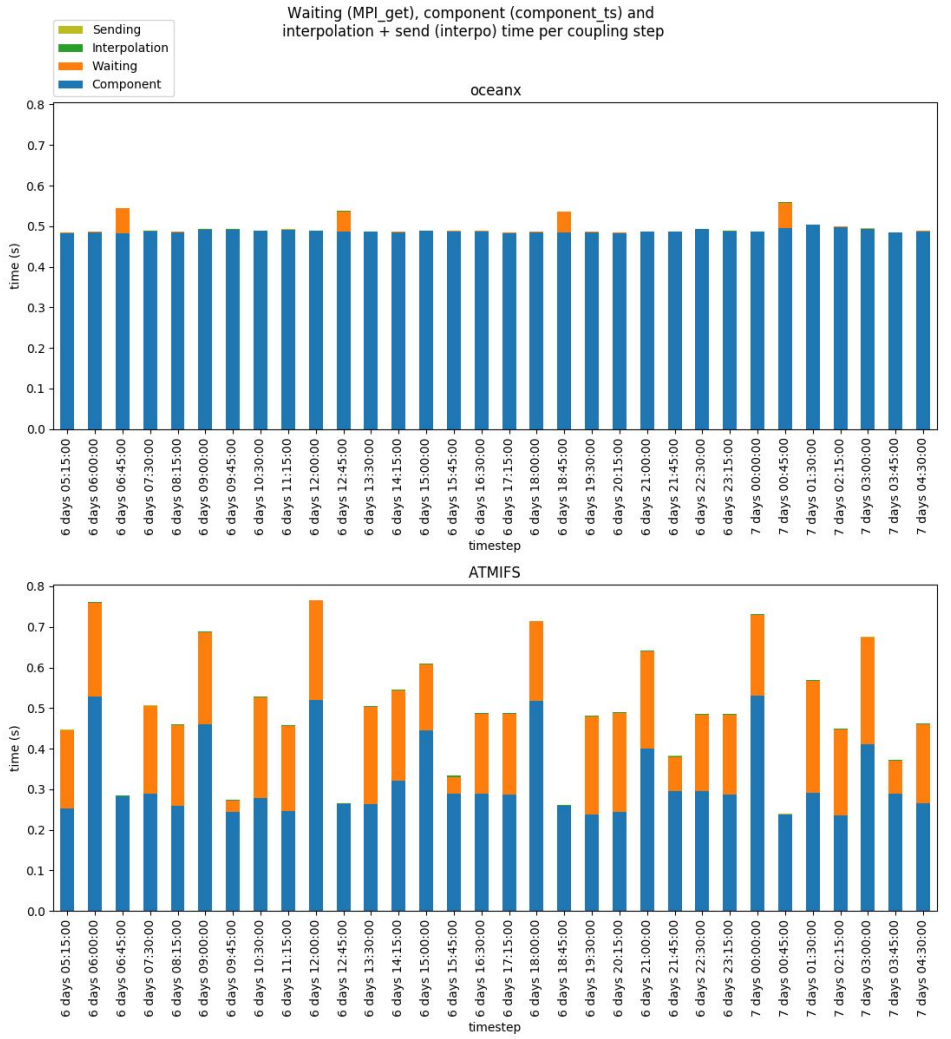




Balancing IFS and NEMO

- IFS is waiting for NEMO → NEMO is slower

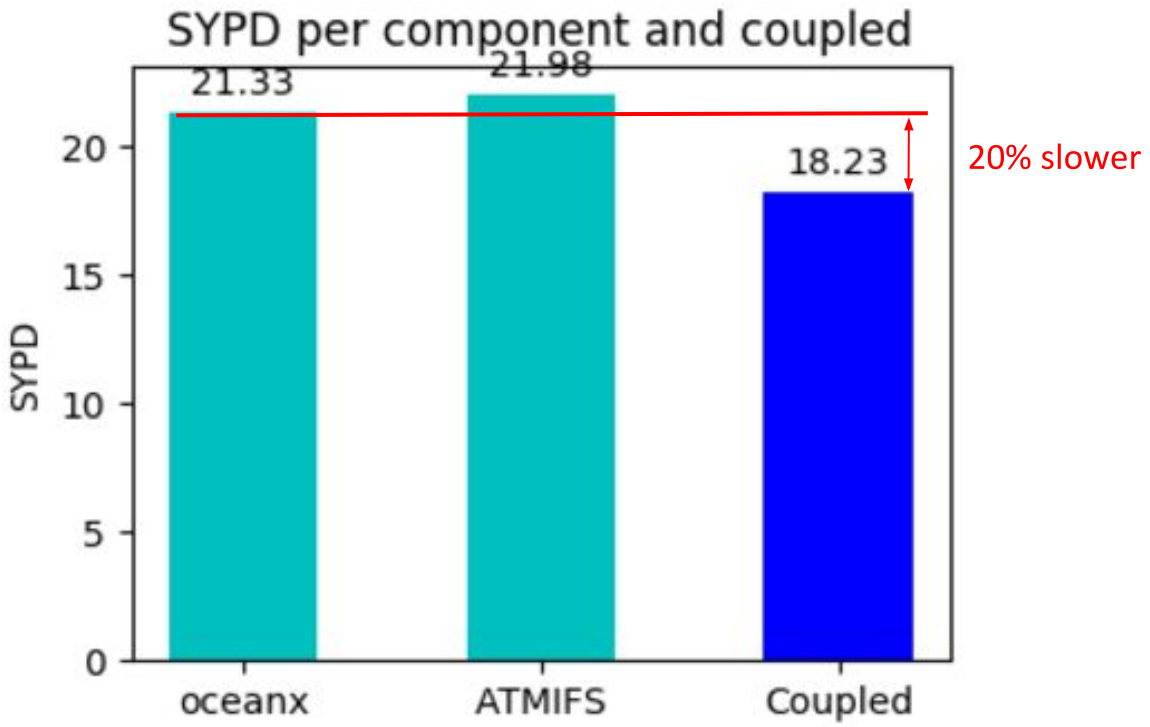
- We give more resources to NEMO until they reach the same speed



Balancing IFS and NEMO

- Both models run at the same speed

- The **coupled SYPD** is significantly **worse** than the speed at which the components are running.

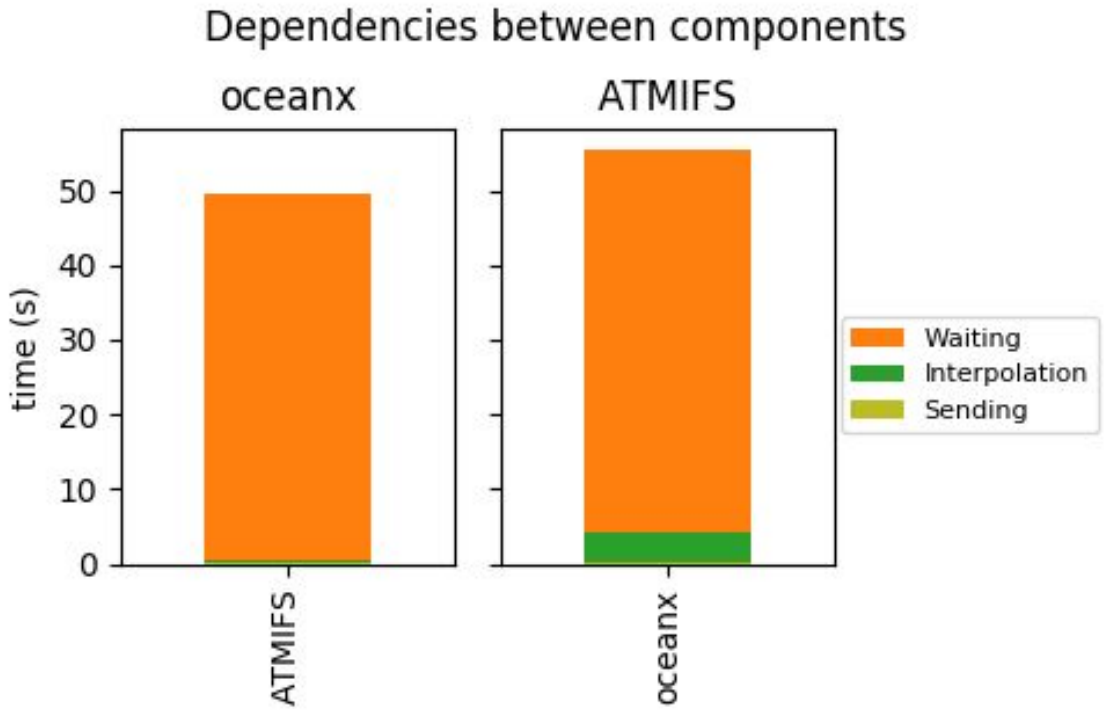


Balancing IFS and NEMO

- Both models have a similar SYPD

- The **coupled SYPD** is significantly **worse** than the speed at which the components are running.

- This could be due to the cost of interpolations, but as we see, they have a minor impact in the execution Instead, both models are waiting to each other



Balancing IFS and NEMO

Both components have to wait each other due to irregular timesteps in IFS

IFS timestep computation is extended every 3 hours due to the calculation of the radiation. NEMO has to wait for IFS every time this happens

Impossible to detect this irregularities in the past. Giving information per coupling step can provide a deeper insight to the users and allow better configurations to be used

Optimizing NEMO and IFS balance

The **partial coupling cost (PCC)** shows how the **coupling cost (C)** is splitted among the components used:

- PCC (IFS) = 13.49
- PCC (NEMO) = 3.34

- C = 16.83

This means that the **cost** of the coupling (mainly waiting time) on IFS is the main waste of resources in this configuration. This is due to the number of processes used in IFS (528), much greater than the ones used for NEMO (144)

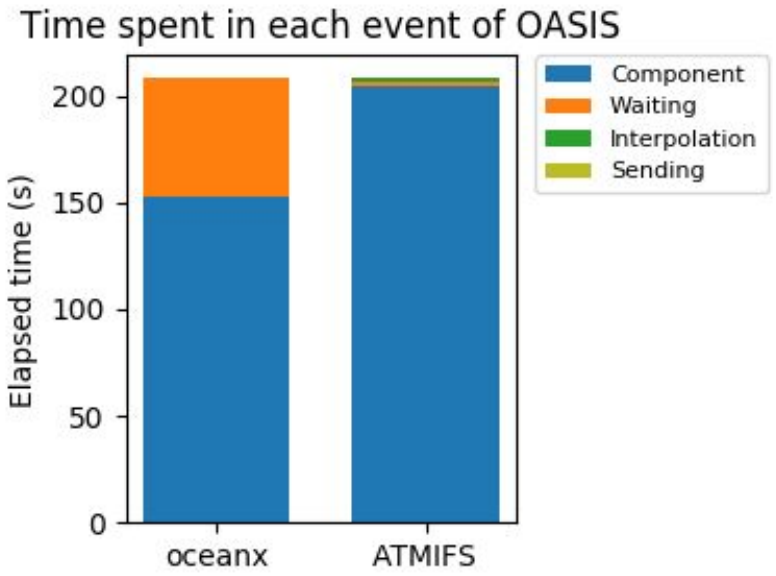
To reduce the coupling cost, we reduce the waiting time of IFS → **increase the speed of NEMO**

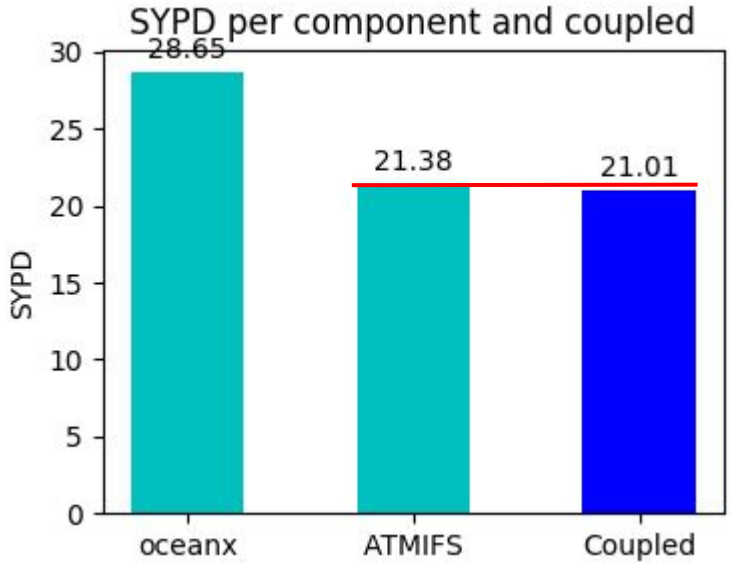
```
oceanx
Number of processes: 144
Number of coupling steps: 593
Coupled component execution time: 242.503
Coupled component cost (coupled time (h) * num_proc): 9.700
Component useful execution time: 204.700 (84.41%)
Component coupling time: 37.803 (15.59%)
Component waiting time: 37.523
Component interpolation time: 0.236

oceanx partial coupling cost: 3.340%
oceanx SYPD: 21.393
oceanx CHPSY: 161.547

-----------------------------------

ATMIFS
Number of processes: 528
Number of coupling steps: 593
Coupled component execution time: 242.503
Coupled component cost (coupled time (h) * num_proc): 35.567
Component useful execution time: 200.855 (82.83%)
Component coupling time: 41.648 (17.17%)
Component waiting time: 32.033
Component interpolation time: 9.508

ATMIFS partial coupling cost: 13.494%
ATMIFS SYPD: 21.803
ATMIFS CHPSY: 581.212

-----------------------------------


Total coupling cost: 16.83%
Coupled SYPD: 18.06
Coupled CHPSY: 893.11
```

Optimizing NEMO and IFS balance

- NEMO is waiting for IFS → IFS is slower

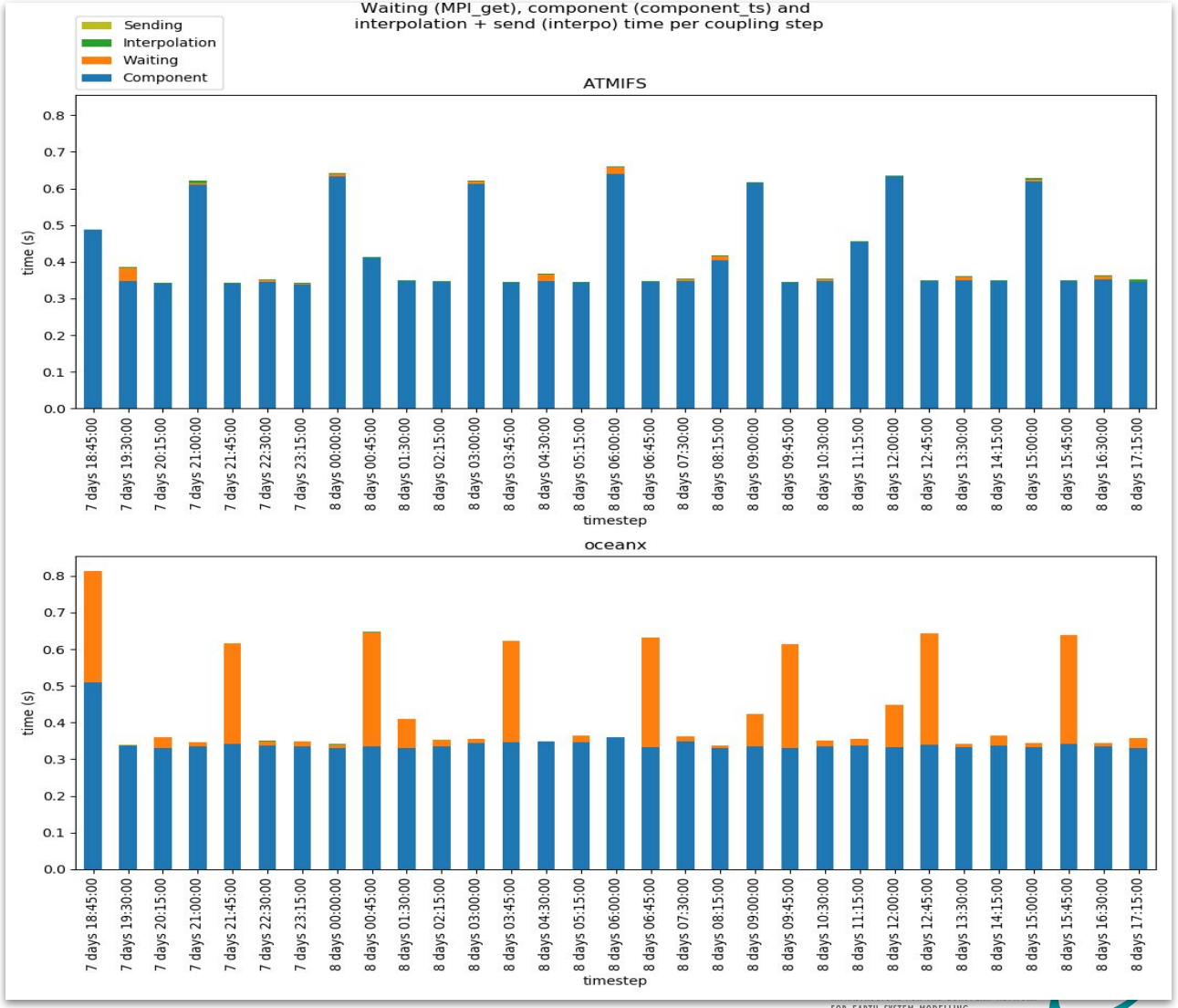
- Coupled execution as slow as the slowest component (IFS)





Optimizing NEMO and IFS balance

- IFS waiting time is negligible
- NEMO waiting time mainly during the radiation phase of IFS (every 3h)
- NEMO coupling cost is less since it uses fewer processes



Optimizing NEMO and IFS balance

Before analysis

```
Coupled SYPD: 14.161
Coupled CHPSY: 1132.124
Coupling cost: 31.040%
```

IFS faster than NEMO

Optimizing NEMO and IFS balance

Before analysis

```
Coupled SYPD: 14.161
Coupled CHPSY: 1132.124
Coupling cost: 31.040%
```

IFS faster than NEMO

+27.5 %
- 21.1 %
- 45.7 %

Balanced

```
Coupled SYPD: 18.058
Coupled CHPSY: 893.110
Coupling cost: 16.835%
```

IFS and NEMO at the same speed

Optimizing NEMO and IFS balance

Before analysis

```
Coupled SYPD: 14.161
Coupled CHPSY: 1132.124
Coupling cost: 31.040%
```

IFS faster than NEMO

+27.5 %
- 21.1 %
- 45.7 %

Balanced

```
Coupled SYPD: 18.058
Coupled CHPSY: 893.110
Coupling cost: 16.835%
```

IFS and NEMO at the same speed

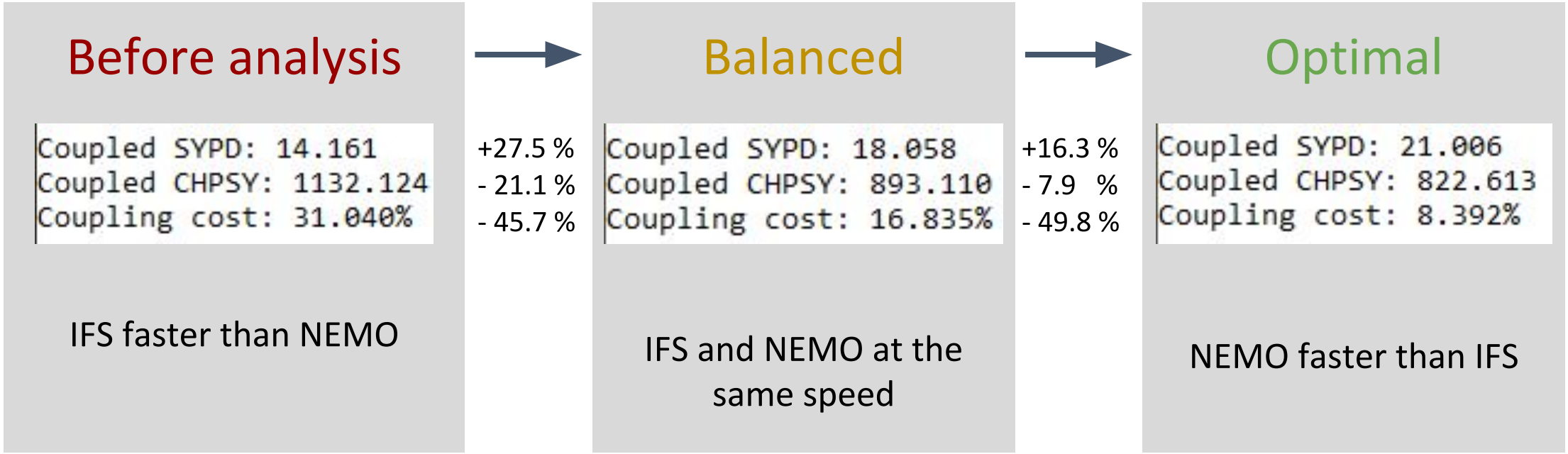
+16.3 %
- 7.9 %
- 49.8 %

Optimal

```
Coupled SYPD: 21.006
Coupled CHPSY: 822.613
Coupling cost: 8.392%
```

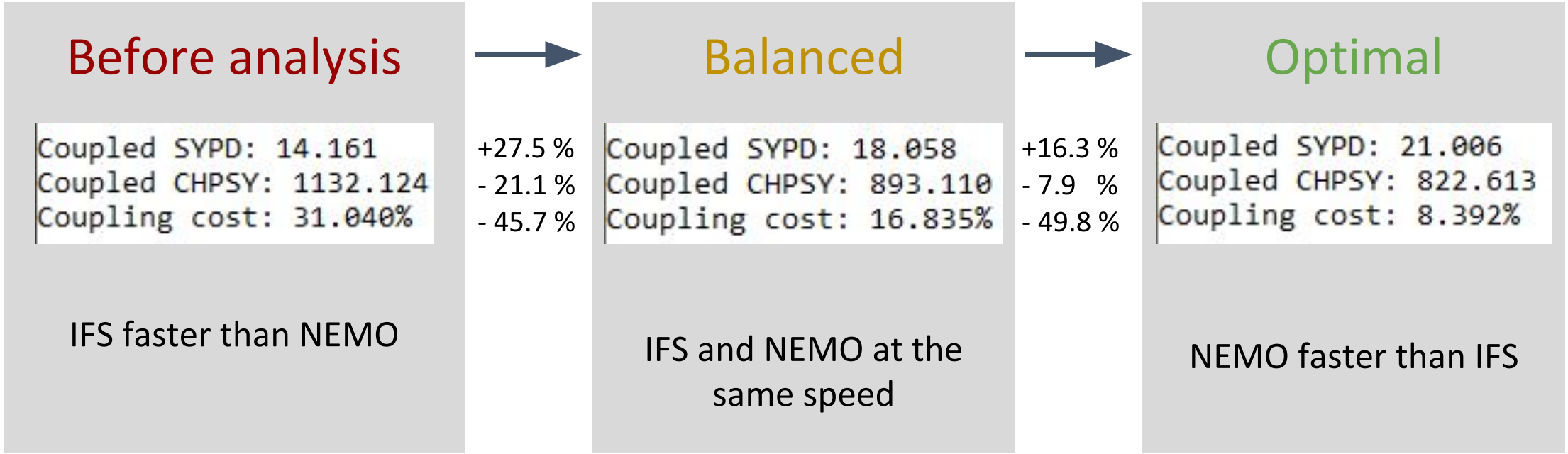
NEMO faster than IFS

Optimizing NEMO and IFS balance



Intuitive methods can fail to achieve **the best solution**

Optimizing NEMO and IFS balance



Intuitive methods can fail to achieve the **best solution**

It is necessary to have a **tool** which provides the required **performance metrics** to make this analysis possible

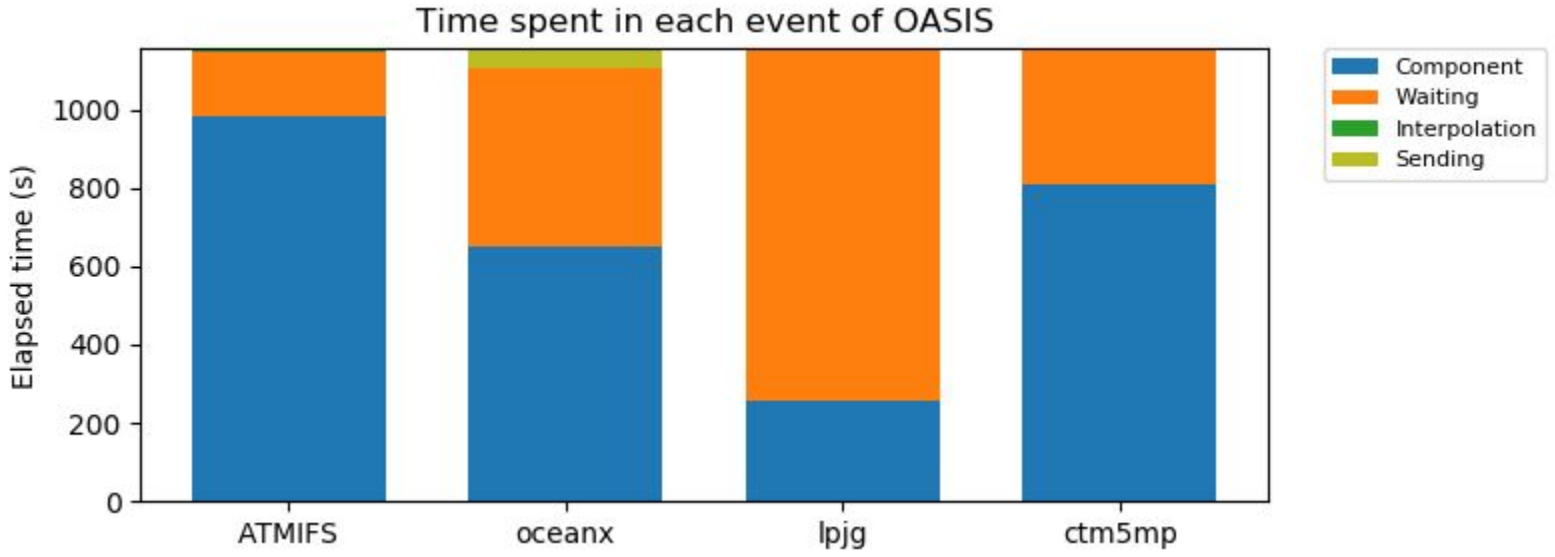
Brief coupling case studies

Coupling cost

Ignoring components

Dependencies

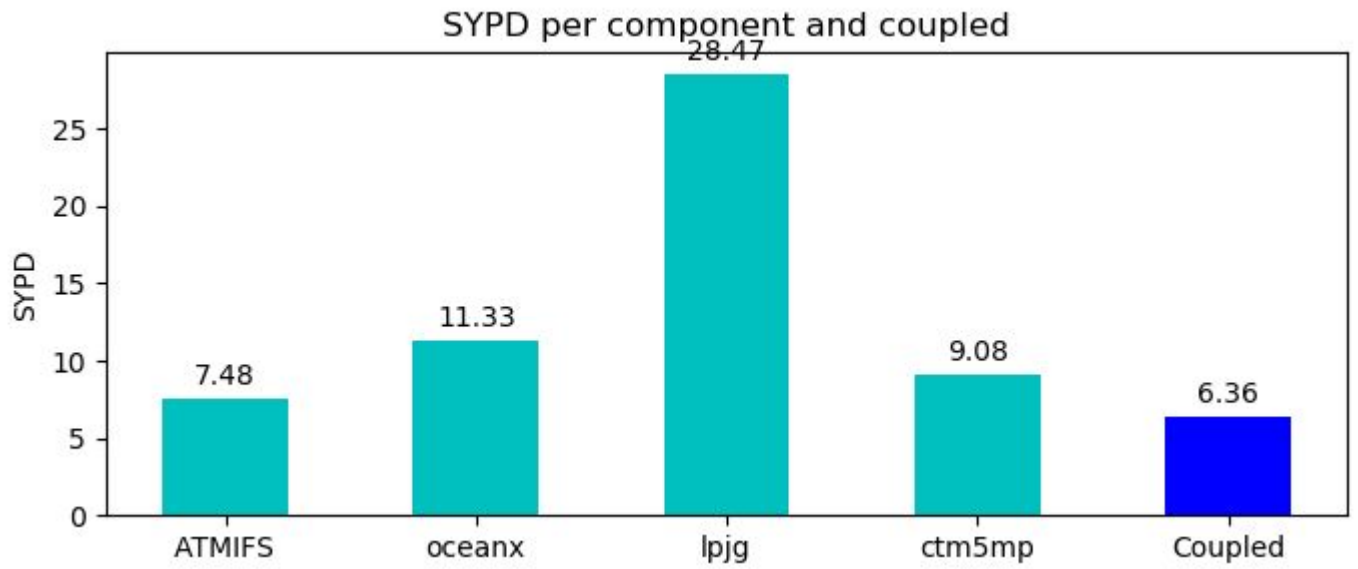
IFS + NEMO + TM5 + LPJG



- IFS and TM5 are the slowest components. Still, they have an overhead due to waiting time
- NEMO is almost half of the time waiting
- LPJG is the fastest component

IFS + NEMO + TM5 + LPJG

Firstly, we check which of the components is adding more to the coupling cost (i.e. acting as a bottleneck) using the partial coupling cost (PCC)



PCC

IFS → 3.01

NEMO → 8.77

LPJG → 15.53

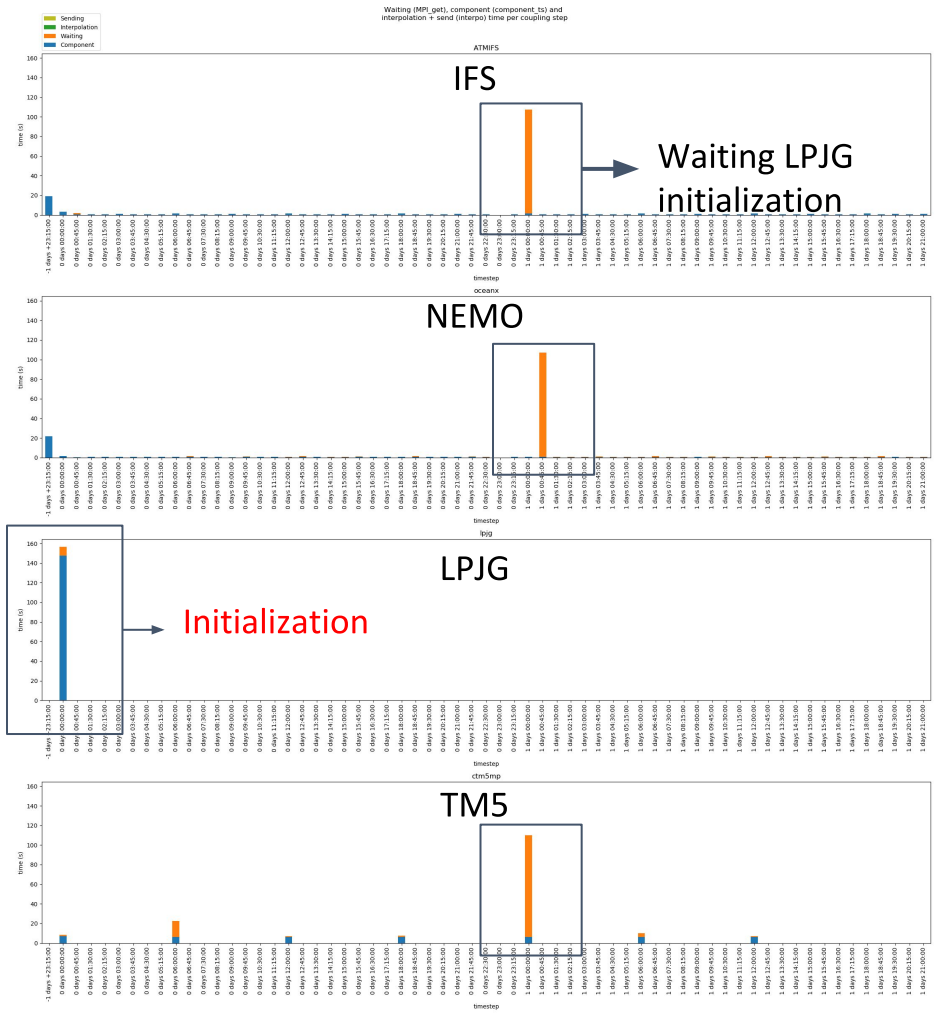
TM5 → 5.99

C = 33.3%

IFS + NEMO + TM5 + LPJG

However, after having a look to the pattern, we observe that LPJG calculation time is due to a heavy initialization (around 2 min) which has a direct impact on all components which have to wait

To avoid getting biased results due to initialization and finalization (and since we can't change how long this initialization takes) we choose to ignore this phase



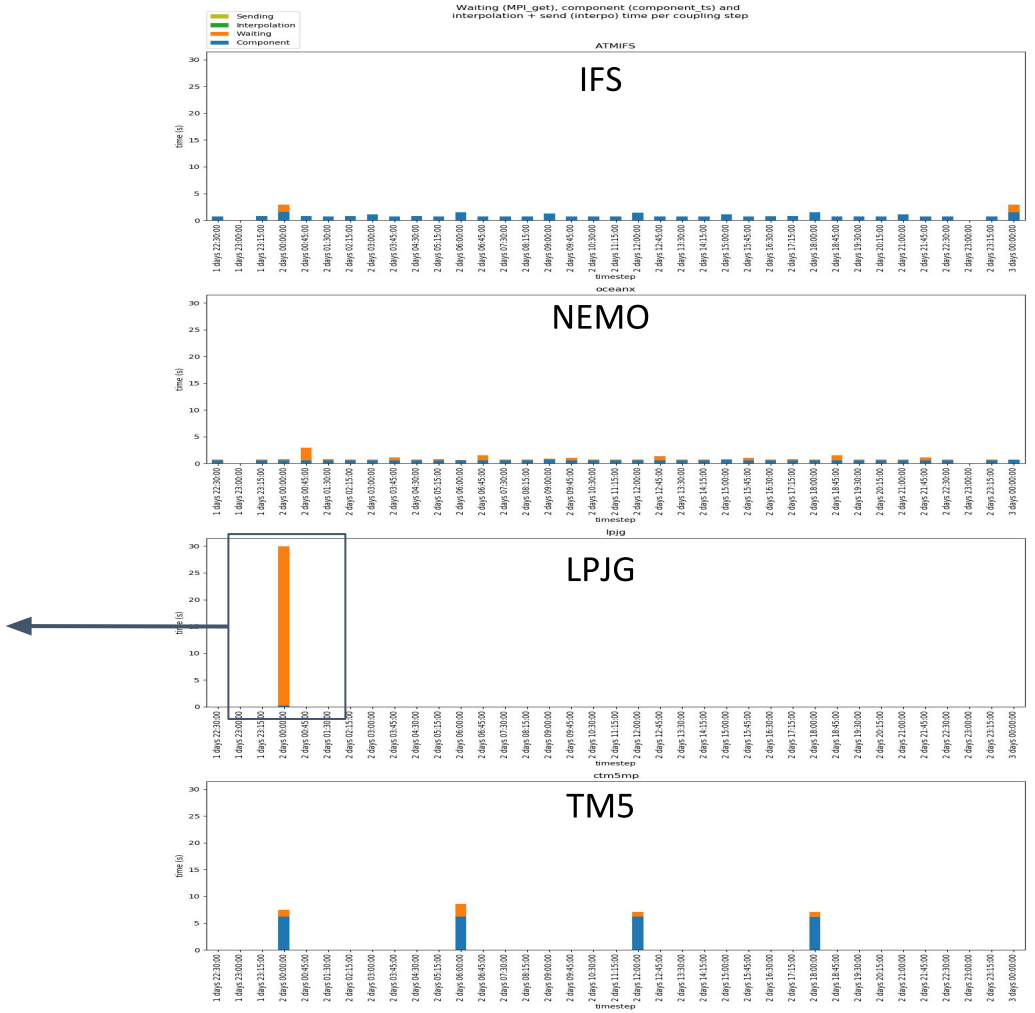
IFS + NEMO + TM5 + LPJG

However, after having a look to the pattern, we observe that LPJG has a heavy initialization (around 2 min) which has a direct impact on all components

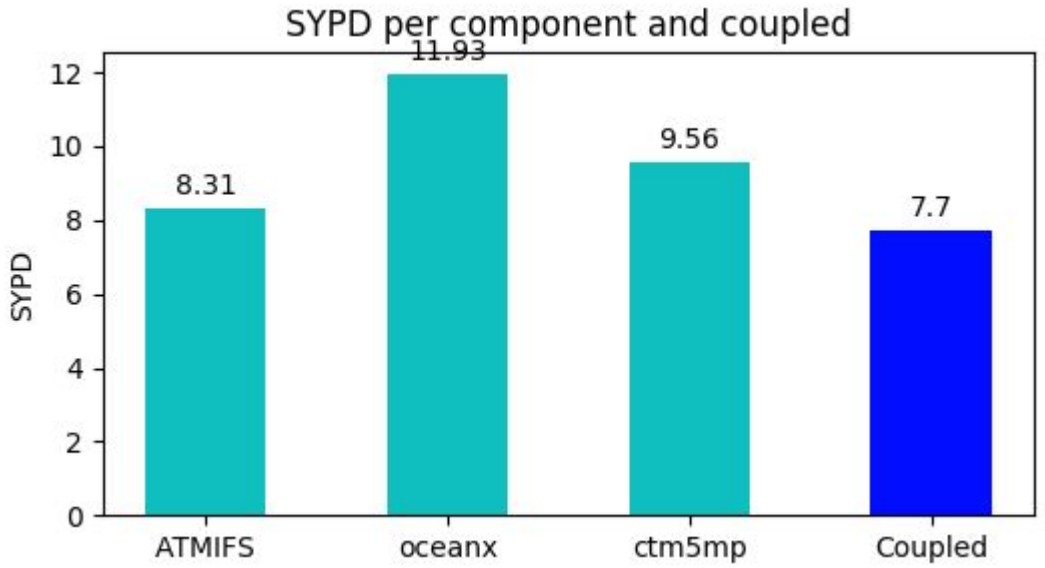
To avoid getting biased results due to initialization and finalization (and since we can't change how long this initialization takes) we choose to ignore this phase

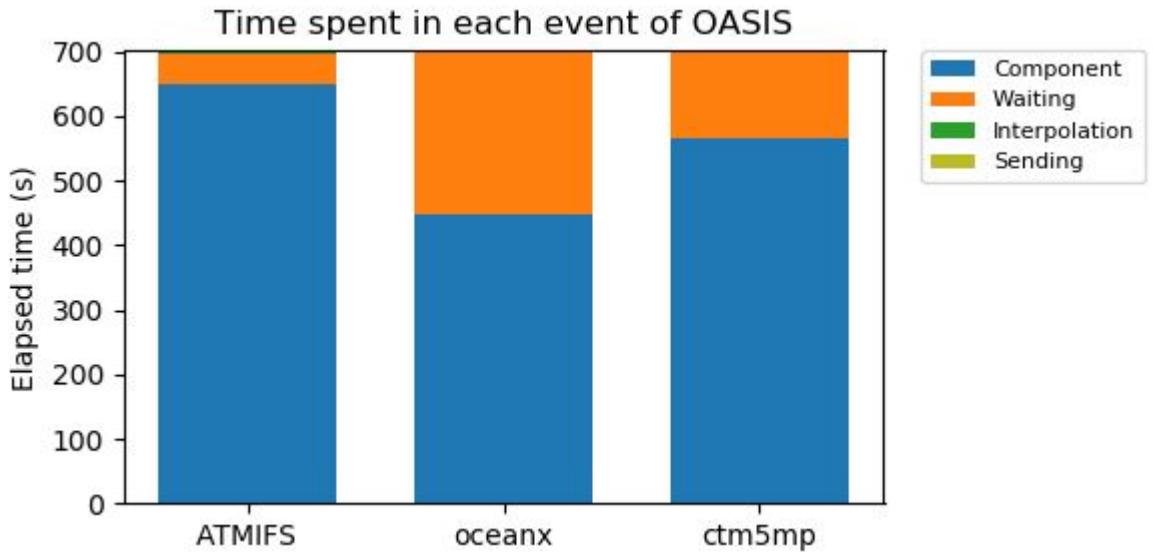
Additionally, LPJG only communicates every 24 hours and most of the time is waiting for the other components

We reduced the number of cores to the minimum and to ignore this component for the balancing study since it has little direct impact to the coupling



IFS + NEMO + TM5 + LPJG





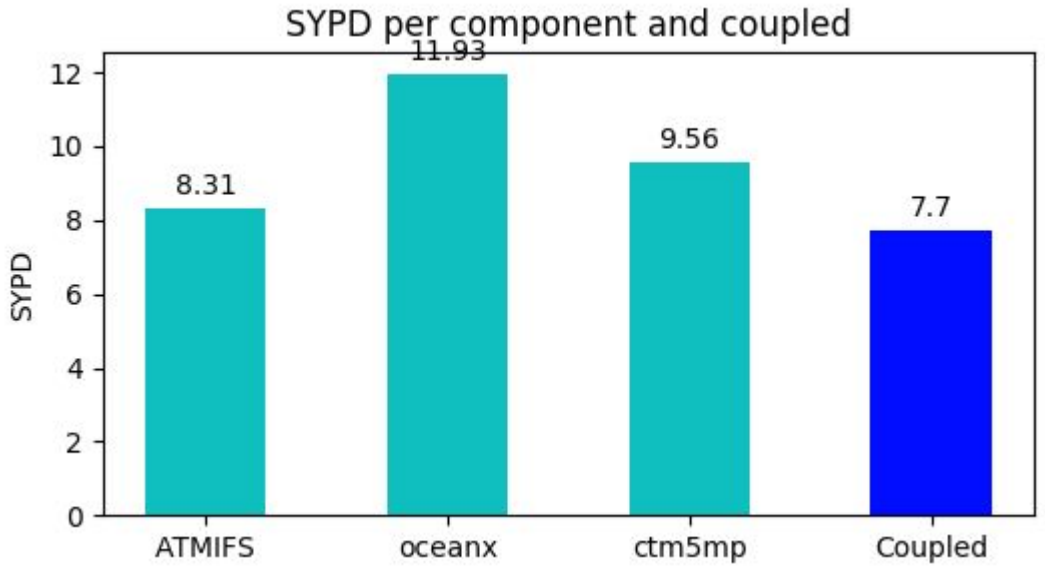
We take a look again to the PCC:

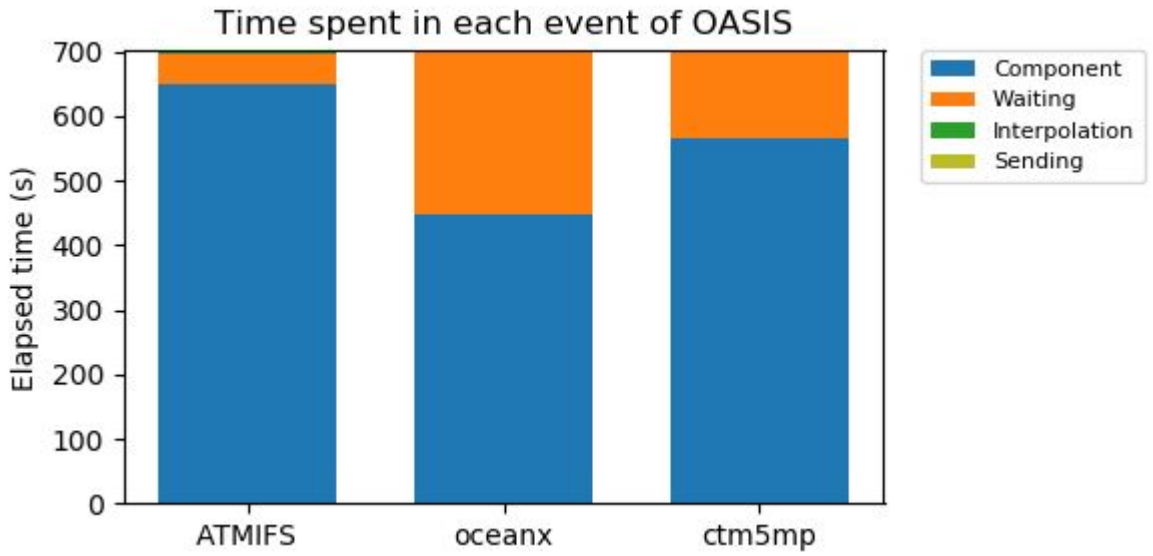
- IFS = 2.48
- NEMO = 11.84
- TM5 = 6.49

} C = 20.81%

We want to reduce NEMO waiting time

IFS + NEMO + TM5 + LPJG





We take a look again to the PCC:

- IFS = 2.48
- NEMO = 11.84
- TM5 = 6.49

C = 20.81%

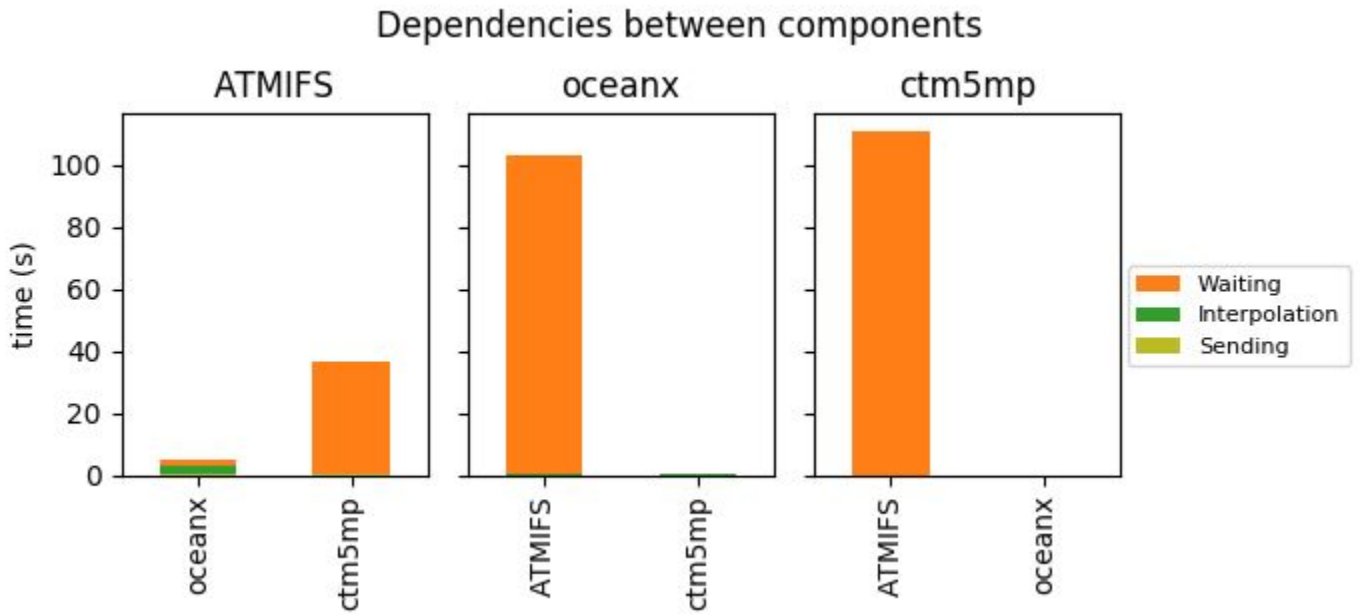
We want to reduce NEMO waiting time

How can we know if we have to give more resources to IFS or to TM5? We need to know the **dependencies between components**

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

is-enes
INFRASTRUCTURE FOR THE EUROPEAN NETWORK FOR EARTH SYSTEM MODELLING

IFS + NEMO + TM5 + LPJG

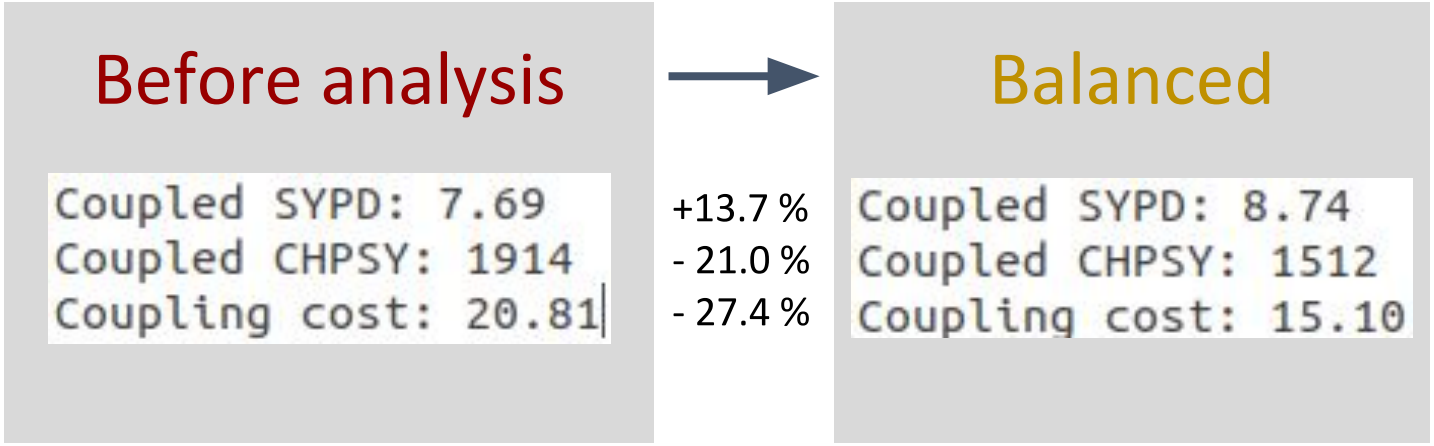


Once we use more than 2 components, it is impossible to know from which component a waiting time is coming from. Key information when deciding to which component we have to increase or reduce resources.

In this case, NEMO is waiting for IFS. Therefore, if we give more processes to IFS, we will improve the usage of the resources. What's more, TM5 is also waiting for IFS. We can solve both issues by speeding up IFS

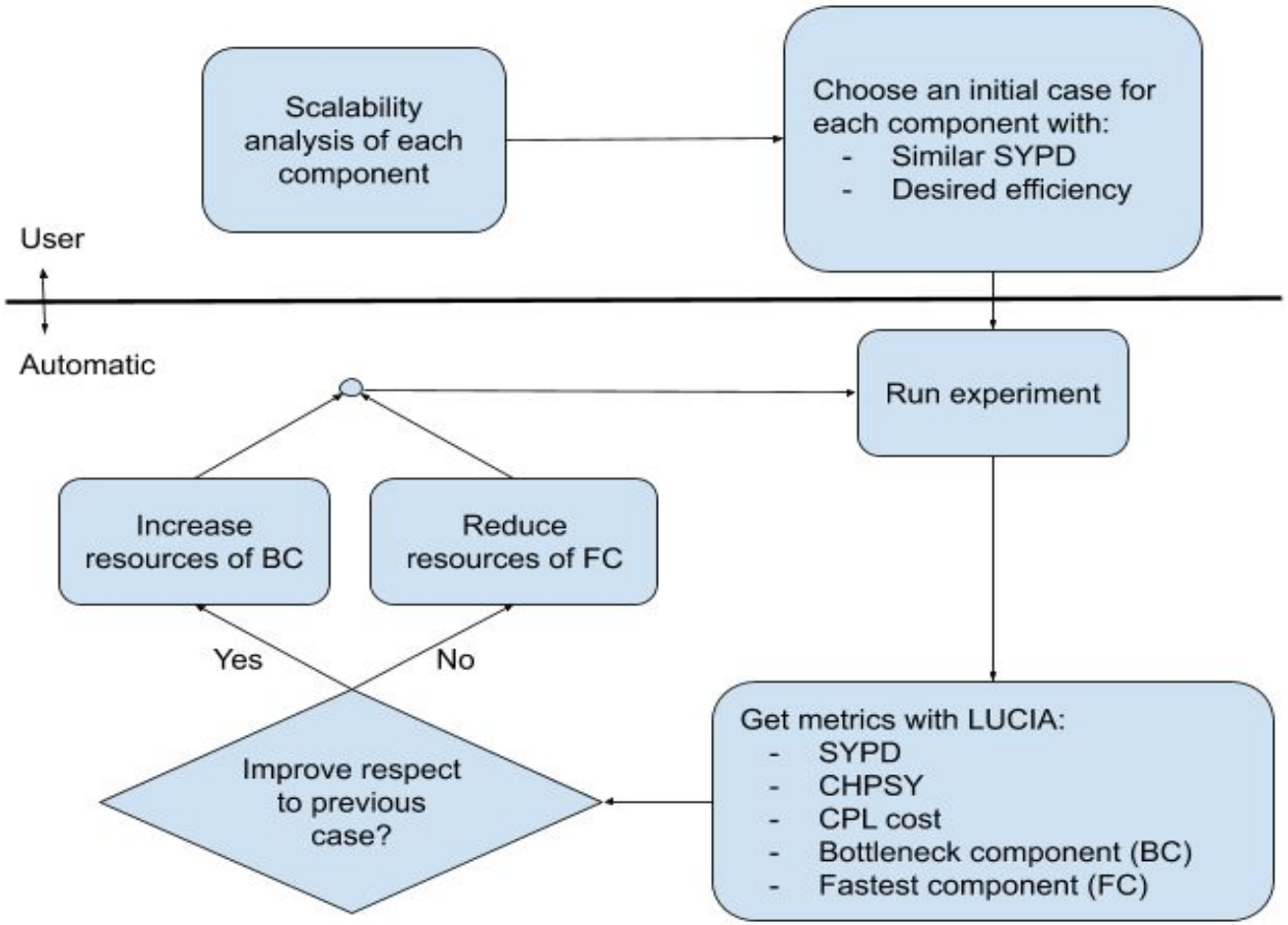
IFS + NEMO + TM5 + LPJG

- LPJG was only an issue during the initialization phase. We have reduced the number of resources used by this component to the minimum
- The PCC showed that we had to reduce the waiting time on NEMO
- The dependencies revealed that we had to improve IFS execution time. This was also applicable to reduce TM5 PCC



To find the best solution, we have to keep iterating this process **manually** until reaching the **optimal solution**

Automatic methodology



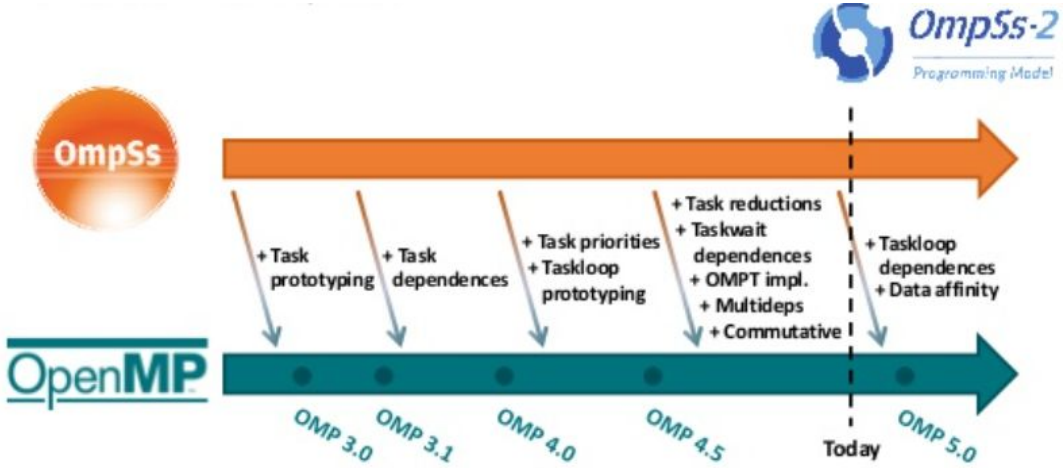
The result will be a coupled configuration which:

- **Minimizes the waiting time to the minimum (CPL cost).**
- **The method will iterate until the solution desired by the user is satisfied (such TTS or Energy to Solution).**

The future: a new dynamical approach

MPI-OpenMP interoperability: Taskifying MPI calls

- Overlap/out of order execution
  - Computation-communication
  - Communication-communication
  - Phases/iterations
- Provide laxity for communications
  - Tolerance poorer communications
- Migrate/aggregate load balance issues
  - Flexibility for Dynamic Load Balance (it could used as a library)



This approach will be tested in IFS for DEEP-SEA project

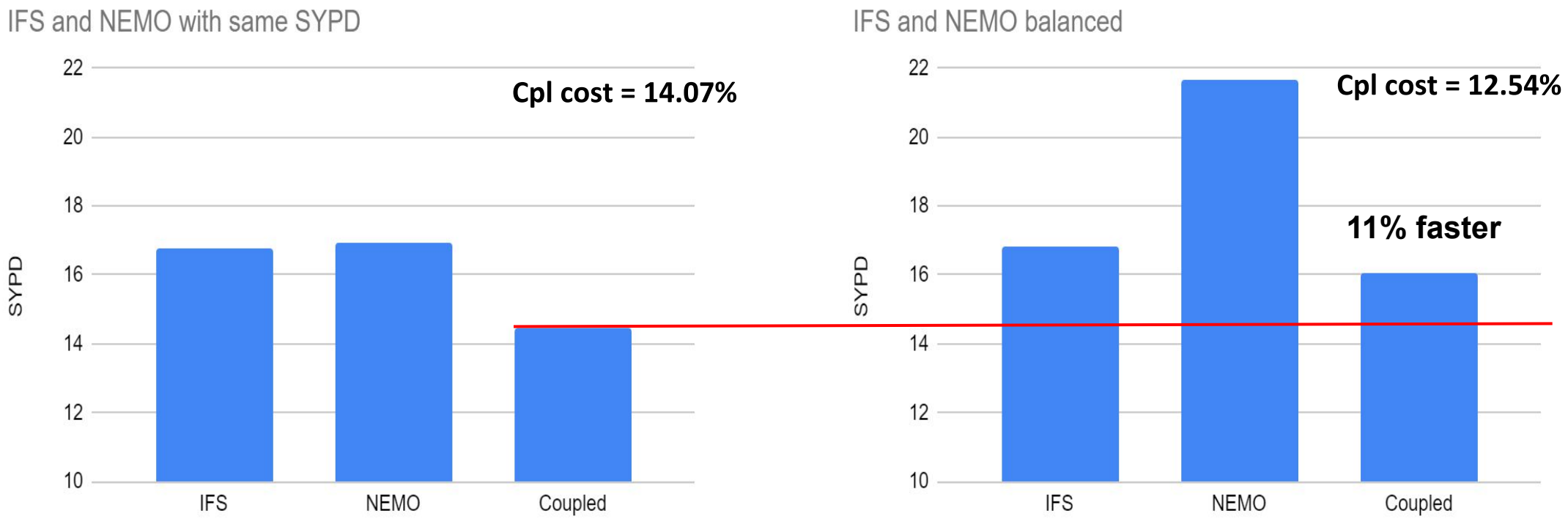
- Improving the efficient use of the computational resources inside a computing node
- Extend OpenMP with new directives to support asynchronous parallelism and heterogeneity
- Creating data-dependencies of the calculation and communication tasks and improving the performance automatically

Thank you

mario.acosta@bsc.es sergi.palomas@bsc.es

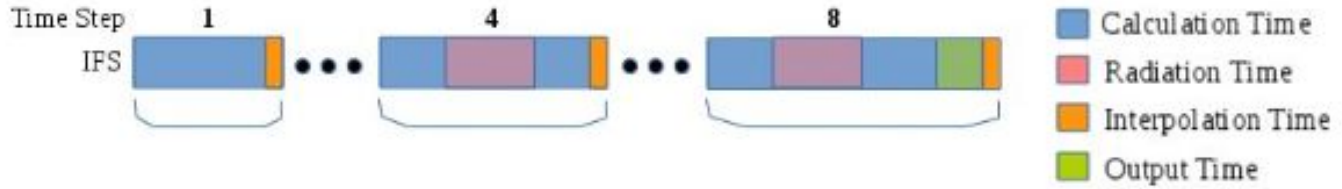
Coupling pattern

A simplistic approach as using the same SYPD for each component can fail

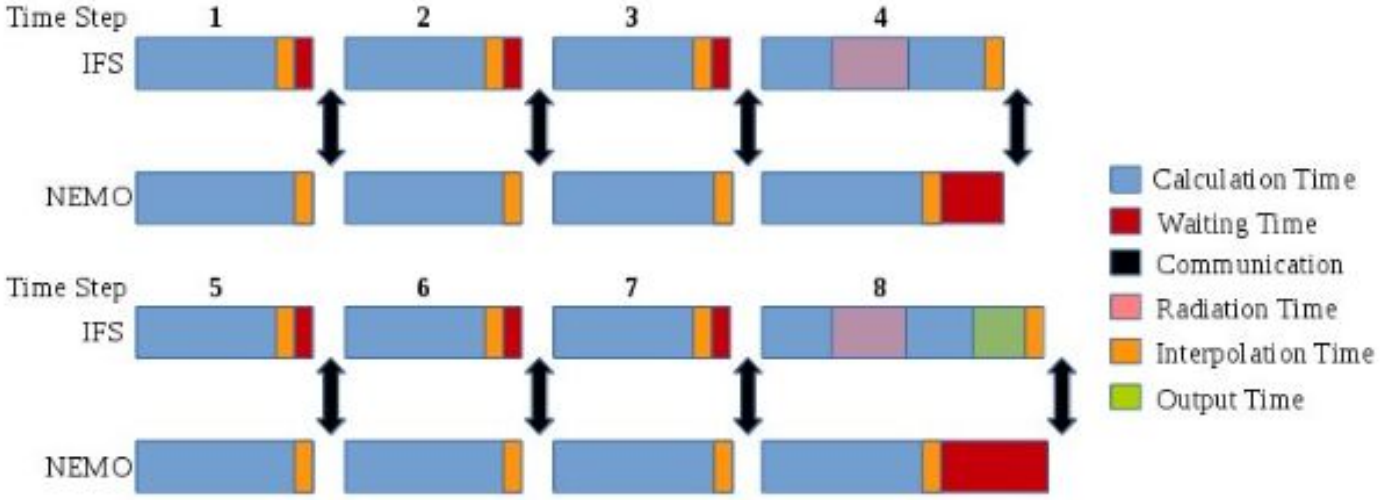


Coupling pattern

- IFS execution time step is not regular



  - Six per eight time steps are regular
  - One per four time steps calculates radiation
  - One per eight time steps produces outputs



Context

- The necessary refactoring of numerical codes is given a lot of attention and is stirring a number of discussions.
  - Computational performance analysis and new optimizations are needed for actual numerical models.
  - Study new algorithms for the new generation of high performance platforms (path to exascale).
- Several European institutions and projects working together on the same direction (ESiWACE2, IS-ENES3, ETP4HPC…)

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

is-enes
INFRASTRUCTURE FOR THE EUROPEAN NETWORK FOR EARTH SYSTEM MODELLING

Context

- **ISENES3 H2020 project** performs a complete computational and energy performance analysis of CMIP6 models using the CPMIP metrics
  - A set of metrics thought to study the computational performance of climate models.
- The task will move beyond traditional metrics and expand the database with a systematic collection of performance results from IS-ENES3 and external partners.

We have four goals through this task: **(in collaboration BSC-CERFACS-CMCC)**

- Collect and define new metrics and generate analyses to reduce the cost of each component independently and an Earth System Model as a whole
- The coupling results will differentiate load-balancing, component-interaction and single component issues
- **Integrate metrics in community tools (improve LUCIA/OASIS3-MCT)**
- Prove how to use the new set of metrics for analysis and improve the performance of the model