www.bsc.es

Santiago de Compostela, 18
July 2017

Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

EXCELENCIA
SEVERO
OCHOA

# Optimization of Earth Sciences models

Miguel Castrillo

# Introduction

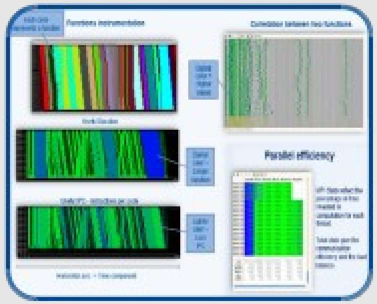
# Barcelona Supercomputing Center

- Created in 2005; more than 400 employees.
- Research, develop and manage information technology.
- Facilitate scientific progress and its application in society.

## Earth Science Department

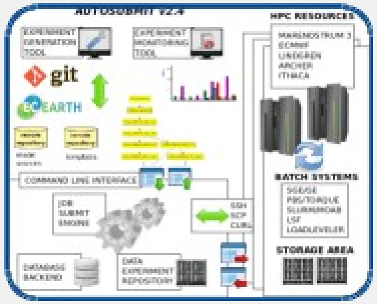
- **Atmospheric composition modelling**
- **Climate prediction modelling**
- **Computational Earth Sciences**
- **Earth Sciences Services**

# Structure



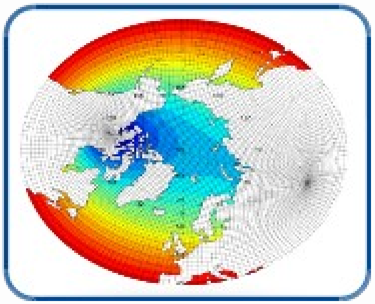
## Performance Team

- Provide HPC Services such as performance analysis
- Apply new computational methods



## Models and Workflows Team

- Development of HPC user-friendly software framework
- Support the development of atmospheric research software



## Data and Diagnostics Team

- Big Data in Earth Sciences
- Provision of data services
- Visualization

# Motivation

- The **necessary refactoring** of numerical codes is receiving lot of attention.
  - Need for computational performance **analysis and optimizations**.
  - Study of **algorithms** suitable for the new generations of HPC platforms.
- Several European **institutions and projects** working together in the same direction (ESiWACE,  EsD's, ETP4HPC…)
- **Clock speeds** not increasing further → Supercomputers growing by adding **parallel processing** units.
- **Compilers** doing great work with **low level** optimizations → **Human decisions** in the development are critical to enable optimizations.
- **Overhead** (extra computation and communication) may not be seen as a problem→ When demand increase (i.e. higher resolutions), a bad implementation will become a **bottleneck** at some point.
- **High Performance Computing is an essential part of Weather and Climate models nowadays.**

# Performance Team

- **Efficient use** of the computational resources
  - Provide **HPC studies** such as performance analysis, indentification of bottlenecks and optimization of parallel applications
- **Research** on new computational methods to apply on Earth Sciences models
  - Study of computational and mathematical **algorithms**.
  - Study of novel **architectures** present in new machines.

# Methodology

- Studying the model

    1. Mathematical study
    2. Computational study
    3. Profiling Study
    4. Introducing optimizations
    5. Reproducibility study

www.bsc.es

# BSC performance tools

# Introduction

- What’s **happening** inside a computer during a model execution?
  - –This question would be really difficult to answer if we didn’t have the proper tools
  - –Older approaches, as timing routines fall short to understand what is really happening

- Using the adequate **performance tools** we can try to find an answer

# From information to knowledge

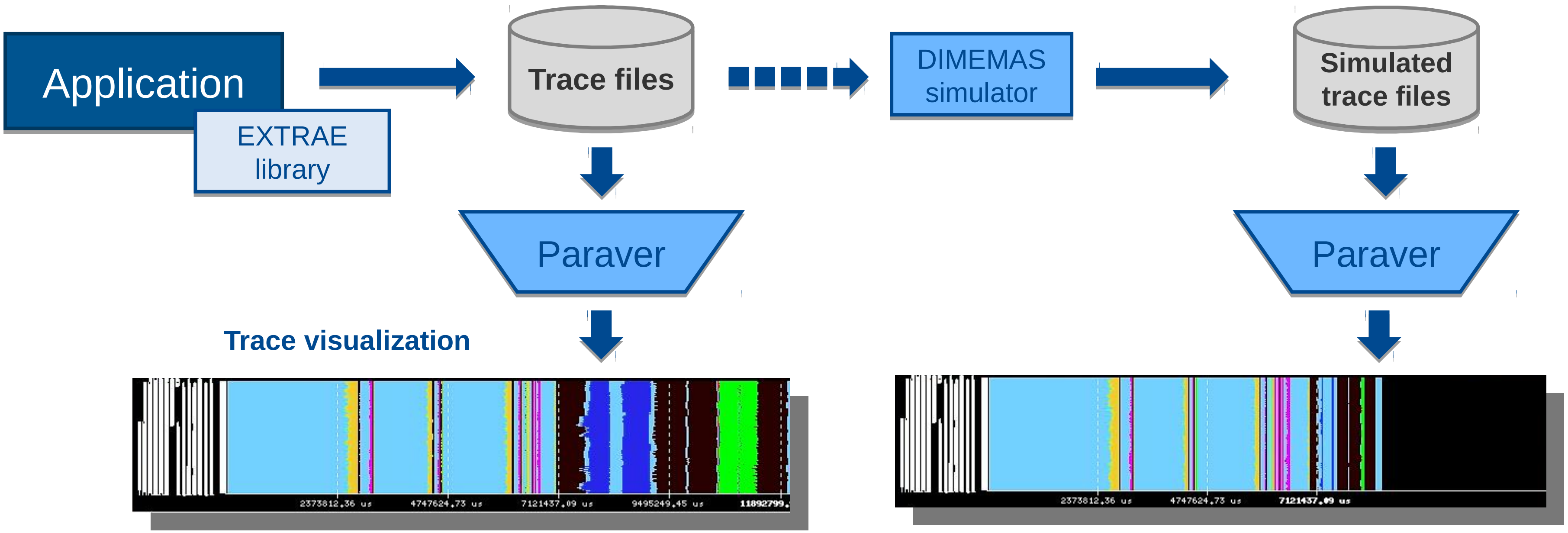
- Having tools able to collect data from every aspect of the application is only part of the equation
- The objective is to **get knowledge** from that data
- Performance tools are intended to use that data to present **useful information**
- It is the expert’s job to analyse that information in order to get knowledge and **take conclusions** about the behaviour of the application

# Trace analysis

- **Trace:**
  - –Event & state history of an application run, for a subsequent analysis.

# BSC tools suite

- Since 1991
- Based on traces
- Open Source: http://www.bsc.es/paraver
- **Extrae**: Package that generates Paraver trace-files for a post-mortem analysis
- **Paraver**: Trace visualization and analysis browser
  - Includes trace manipulation: Filter, cut traces
- **Dimemas**: Message passing simulator



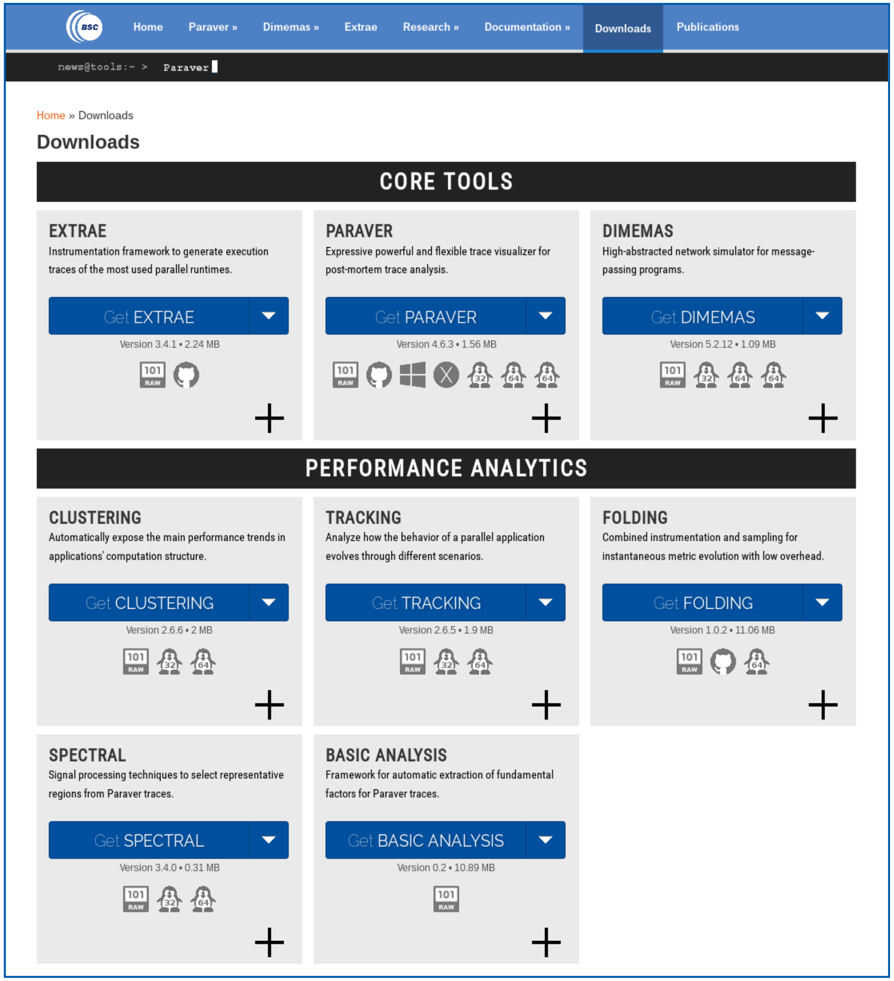
DIMEMAS generated trace. Target = ideal machine

# BSC tools suite

- Download from http://tools.bsc.es

- Extrae
  - Install from sources
    - configure, make, make install
  - Main dependencies
    - MPI
    - libxml2
    - libunwind
    - GNU binutils
    - PAPI

- Paraver & Dimemas
  - Precompiled binaries available



```
news@tools:~ >  Paraver
```

Home » Downloads

Downloads

CORE TOOLS

EXTRAE
Instrumentation framework to generate execution traces of the most used parallel runtimes.

Version 3.4.1 • 2.24 MB

PARAVER
Expressive powerful and flexible trace visualizer for post-mortem trace analysis.

Version 4.6.3 • 1.56 MB

DIMEMAS
High-abstracted network simulator for message-passing programs.

Version 5.2.12 • 1.09 MB

PERFORMANCE ANALYTICS

CLUSTERING
Automatically expose the main performance trends in applications' computation structure.

Version 2.6.6 • 2 MB

TRACKING
Analyze how the behavior of a parallel application evolves through different scenarios.

Version 2.6.5 • 1.9 MB

FOLDING
Combined instrumentation and sampling for instantaneous metric evolution with low overhead.

Version 1.0.2 • 11.06 MB

SPECTRAL
Signal processing techniques to select representative regions from Paraver traces.

Version 3.4.0 • 0.31 MB

BASIC ANALYSIS
Framework for automatic extraction of fundamental factors for Paraver traces.

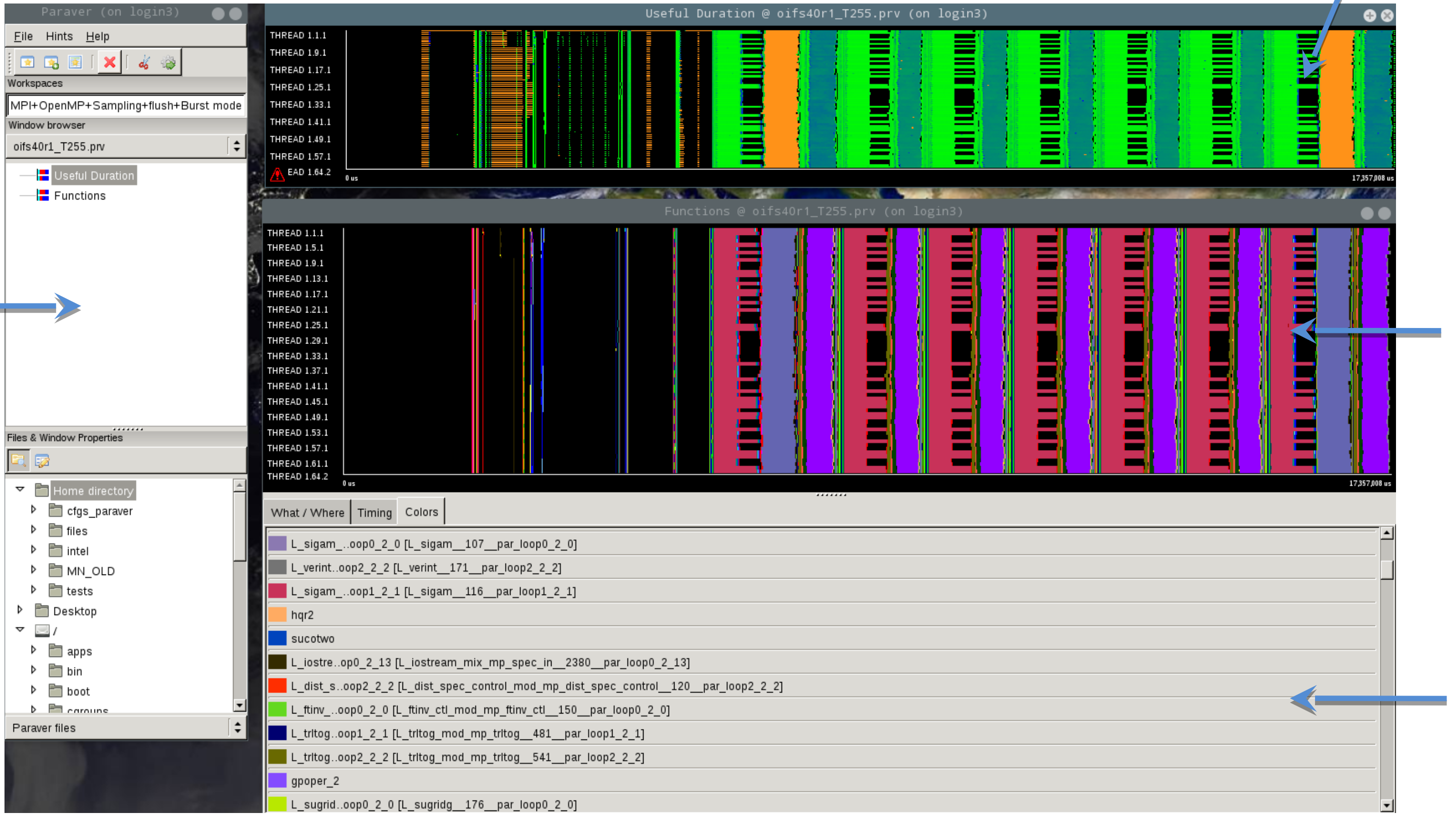
Version 0.2 • 10.89 MB

# BSC tools: Paraver

- Flexible parallel program **visualization** tool based on a GUI.
- From **qualitative global** perception to **deep quantitative** analysis.
- Its power lies on its **flexibility** and **expressive power**.
- Expressive power: Separation between visualization (how to display) and semantic module (value to display):
  - Filter.
  - Semantic functions (categorical, logical, numerical).
  - Multiple levels (thread, task, application).
  - Visualization (tables, timelines).

# BSC tools: Paraver

- Paraver is a very flexible data browser for discovering how an application is running in a parallel environment.



🟪 L_sigam_..oop0_2_0 [L_sigam__107__par_loop0_2_0]
⬛ L_verint..oop2_2_2 [L_verint__171__par_loop2_2_2]
🟥 L_sigam_..oop1_2_1 [L_sigam__116__par_loop1_2_1]
🟧 hqr2
🟦 sucotwo
🟫 L_iostre..op0_2_13 [L_iostream_mix_mp_spec_in__2380__par_loop0_2_13]
🟥 L_dist_s..oop2_2_2 [L_dist_spec_control_mod_mp_dist_spec_control__120__par_loop2_2_2]
🟩 L_ftinv_..oop0_2_0 [L_ftinv_ctl_mod_mp_ftinv_ctl__150__par_loop0_2_0]
🟦 L_trltog..oop1_2_1 [L_trltog_mod_mp_trltog__481__par_loop1_2_1]
🟫 L_trltog..oop2_2_2 [L_trltog_mod_mp_trltog__541__par_loop2_2_2]
🟪 gpoper_2
🟩 L_sugrid..oop0_2_0 [L_sugridg__176__par_loop0_2_0]

Useful duration (communication vs computation)

Paraver Control Window

OpenMP regions

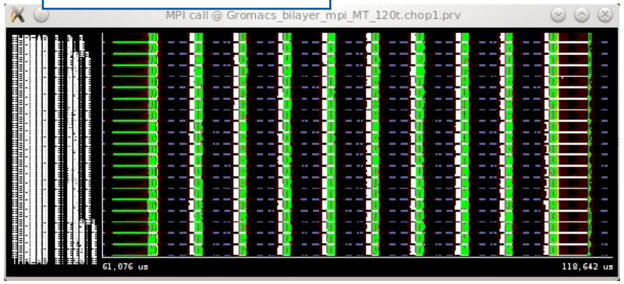
Name of the OpenMP regions

General view of an openIFS T255 trace with Paraver

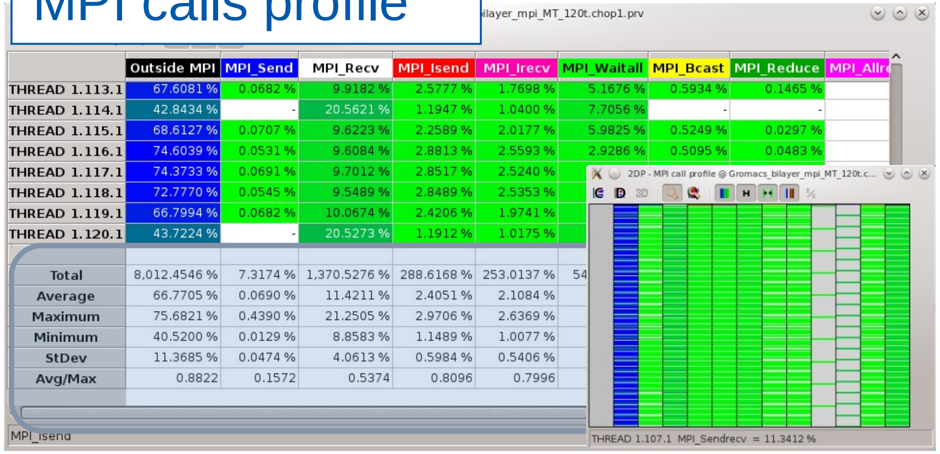
# BSC tools: Paraver

- From timelines to tables

MPI calls



MPI calls profile

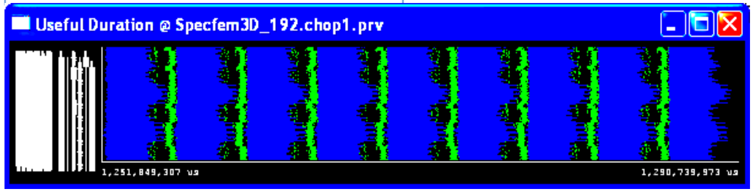

ilayer_mpi_MT_120t.chop1.prv
2DP - MPI call profile @ Gromacs_bilayer_mpi_MT_120t.c...
3D
H

| | Outside MPI | MPI_Send | MPI_Recv | MPI_Isend | MPI_Irecv | MPI_Waitall | MPI_Bcast | MPI_Reduce | MPI_Allre |
|---|---|---|---|---|---|---|---|---|---|
| THREAD 1.113.1 | 67.6081 % | 0.0682 % | 9.9182 % | 2.5777 % | 1.7698 % | 5.1676 % | 0.5934 % | 0.1465 % | |
| THREAD 1.114.1 | 42.8434 % | - | 20.5621 % | 1.1947 % | 1.0400 % | 7.7056 % | - | - | |
| THREAD 1.115.1 | 68.6127 % | 0.0707 % | 9.6223 % | 2.2589 % | 2.0177 % | 5.9825 % | 0.5249 % | 0.0297 % | |
| THREAD 1.116.1 | 74.6039 % | 0.0531 % | 9.6084 % | 2.8813 % | 2.5593 % | 2.9286 % | 0.5095 % | 0.0483 % | |
| THREAD 1.117.1 | 74.3733 % | 0.0691 % | 9.7012 % | 2.8517 % | 2.5240 % | | | | |
| THREAD 1.118.1 | 72.7770 % | 0.0545 % | 9.5489 % | 2.8489 % | 2.5353 % | | | | |
| THREAD 1.119.1 | 66.7994 % | 0.0682 % | 10.0674 % | 2.4206 % | 1.9741 % | | | | |
| THREAD 1.120.1 | 43.7224 % | - | 20.5273 % | 1.1912 % | 1.0175 % | | | | |
| Total | 8,012.4546 % | 7.3174 % | 1,370.5276 % | 288.6168 % | 253.0137 % | 54 | | | |
| Average | 66.7705 % | 0.0690 % | 11.4211 % | 2.4051 % | 2.1084 % | | | | |
| Maximum | 75.6821 % | 0.4390 % | 21.2505 % | 2.9706 % | 2.6369 % | | | | |
| Minimum | 40.5200 % | 0.0129 % | 8.8583 % | 1.1489 % | 1.0077 % | | | | |
| StDev | 11.3685 % | 0.0474 % | 4.0613 % | 0.5984 % | 0.5406 % | | | | |
| Avg/Max | 0.8822 | 0.1572 | 0.5374 | 0.8096 | 0.7996 | | | | |

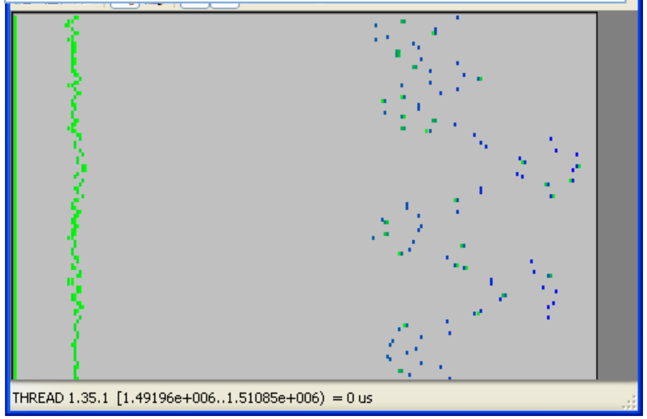
MPI_Isend

THREAD 1.107.1 MPI_Sendrecv = 11.3412 %

Useful Duration



Histogram Useful Duration

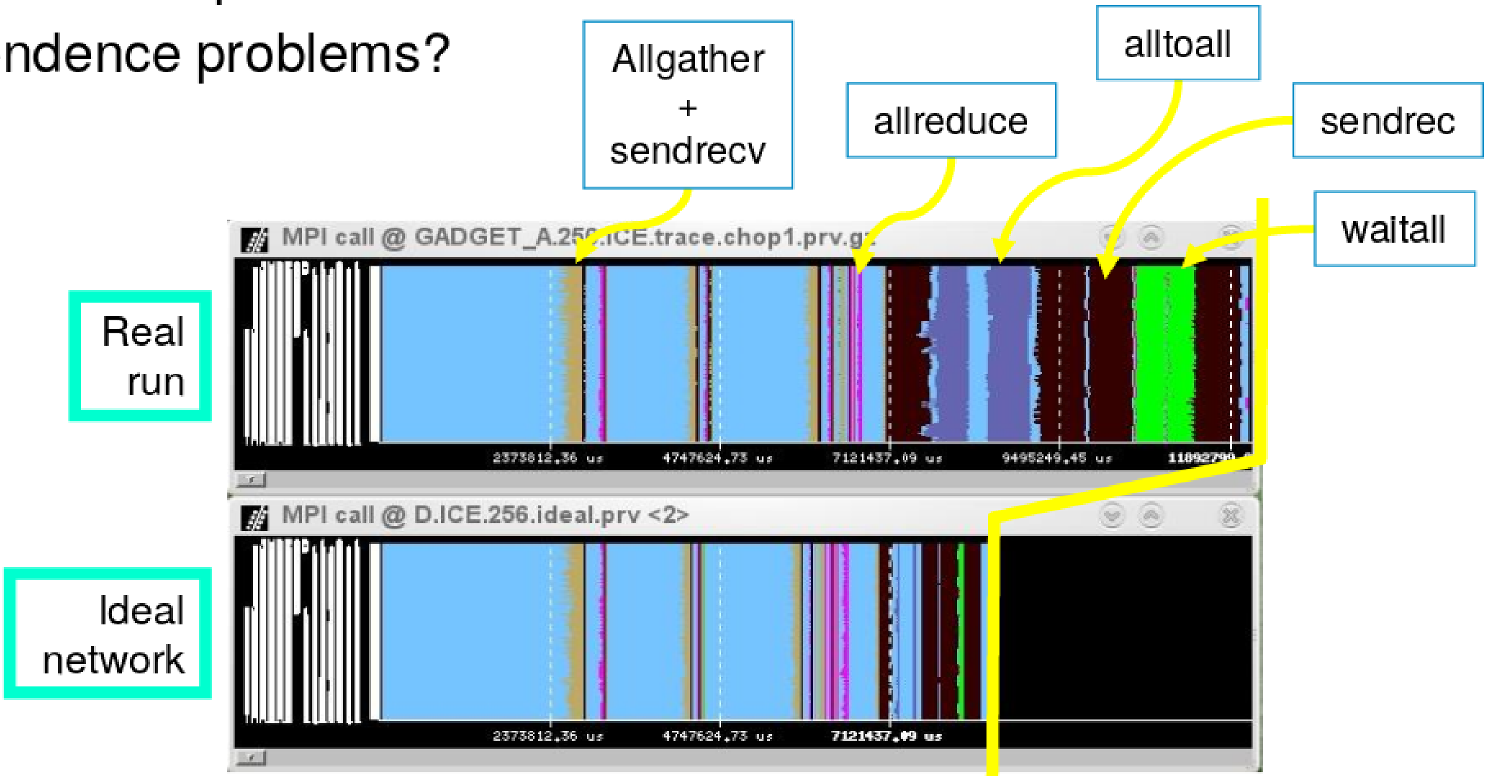


THREAD 1.35.1 [1.49196e+006..1.51085e+006) = 0 us

# BSC Tools: Dimemas

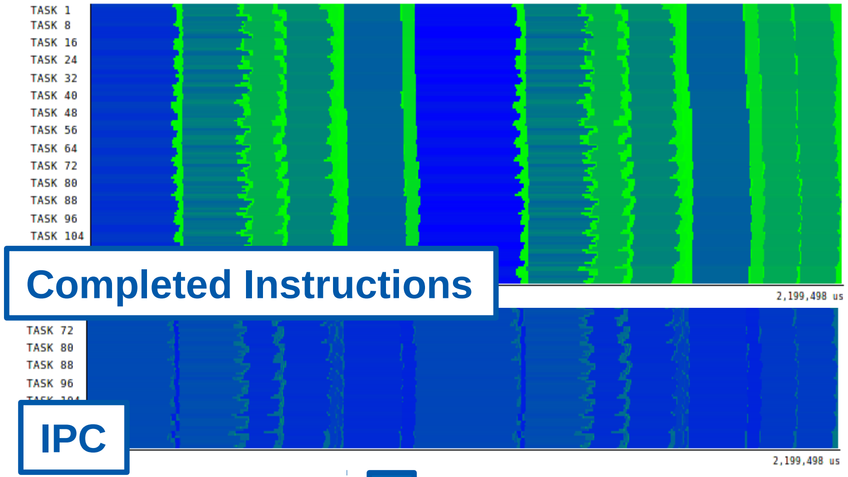
The impossible machine: BW = ∞, L = 0

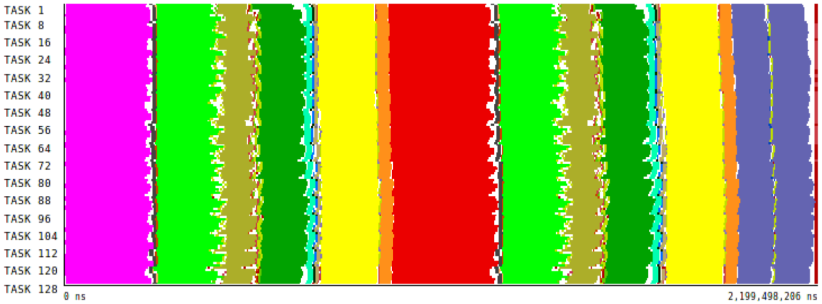
- Actually describes/characterizes intrinsic application behavior
  - Load balance problems?
  - Dependence problems?

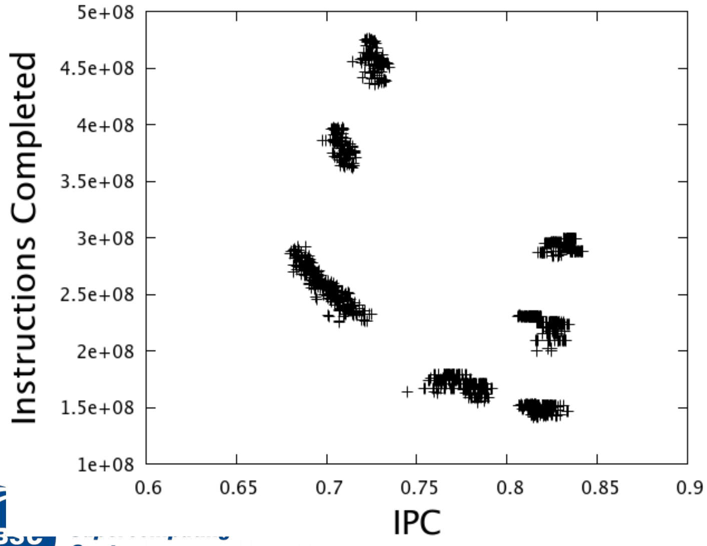


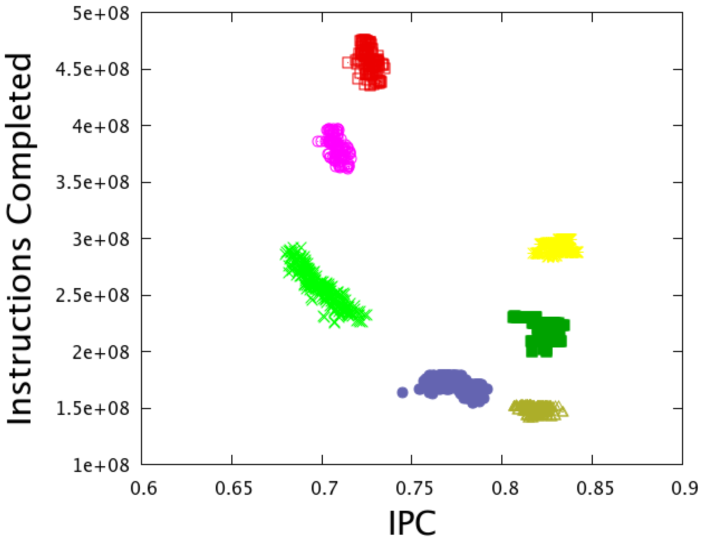
Impact on practical machines?

# BSC Tools: Clustering





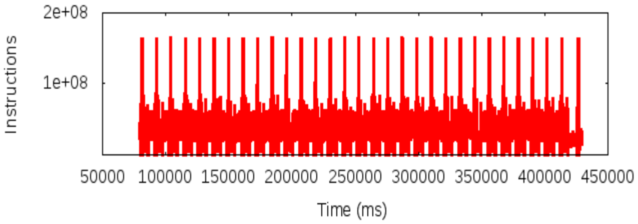




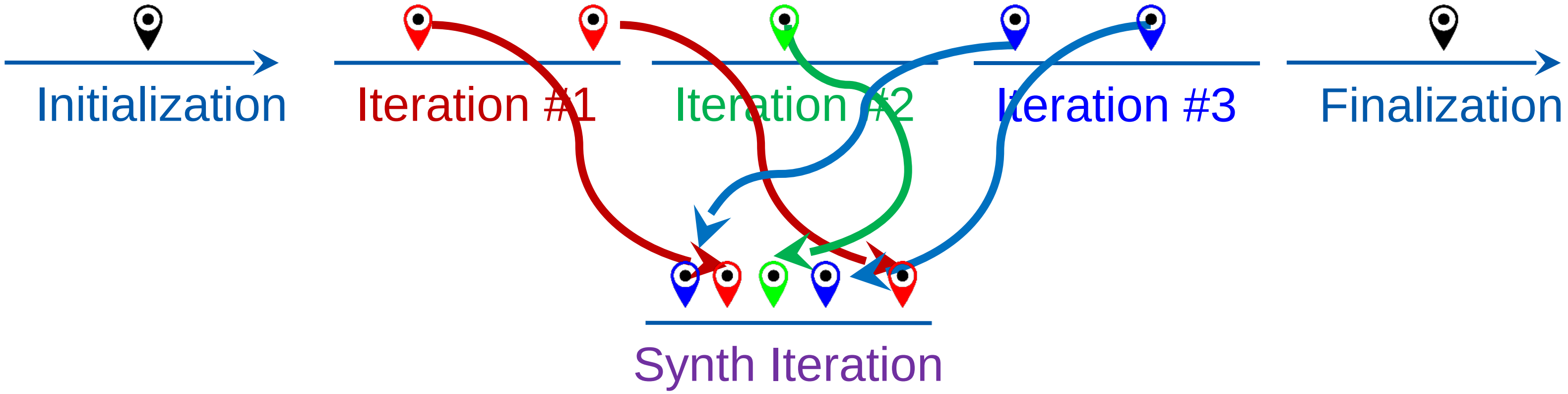
Automatic Detection of Parallel Applications Computation Phases (IPDPS 2009)

# BSC Tools: Folding

- HPC / Scientific applications
  - Repetitive nature



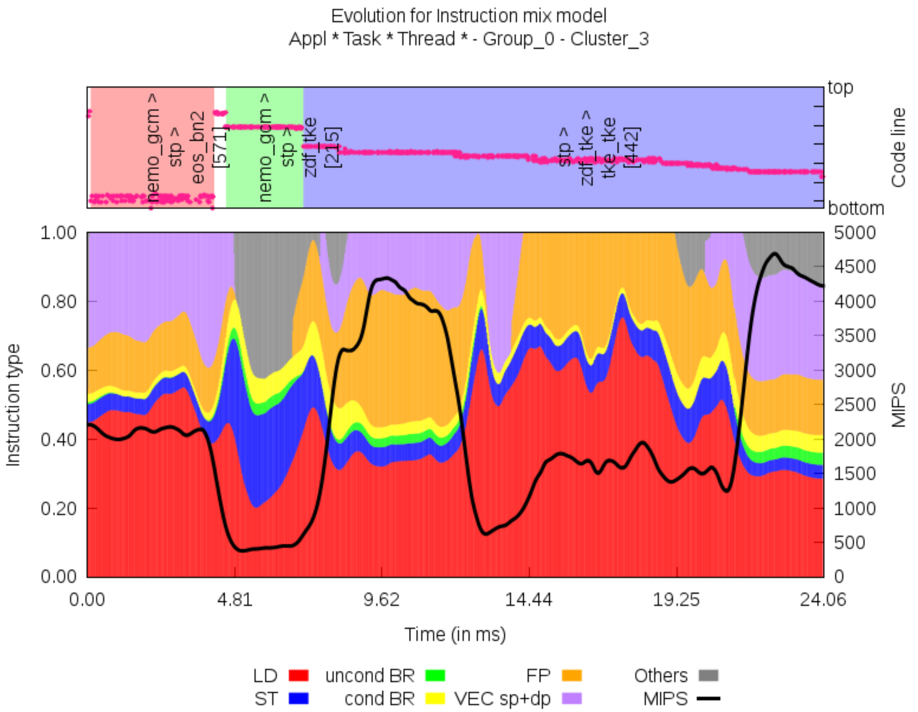
- **Instantaneous metrics** with **minimum overhead**
  - Combine instrumentation and sampling
    - Instrumentation delimits regions (routines, loops, …)
    - Sampling  exposes progression within a region
  - Captures performance counters and call-stack references

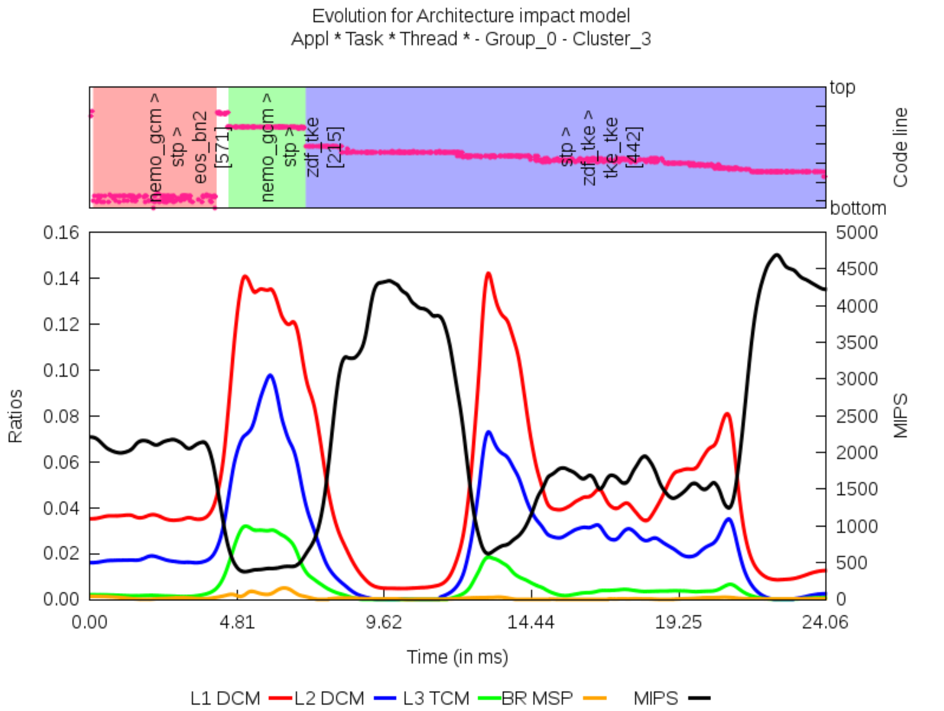


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# BSC Tools: Folding

- The first performance decrease coincides with a lot of store instructions but also other not categorized instructions, and the second with an increase of load and vector operations.
- The minimum value in the MIPS plot line coincides with a peak of the data cache misses ratio.





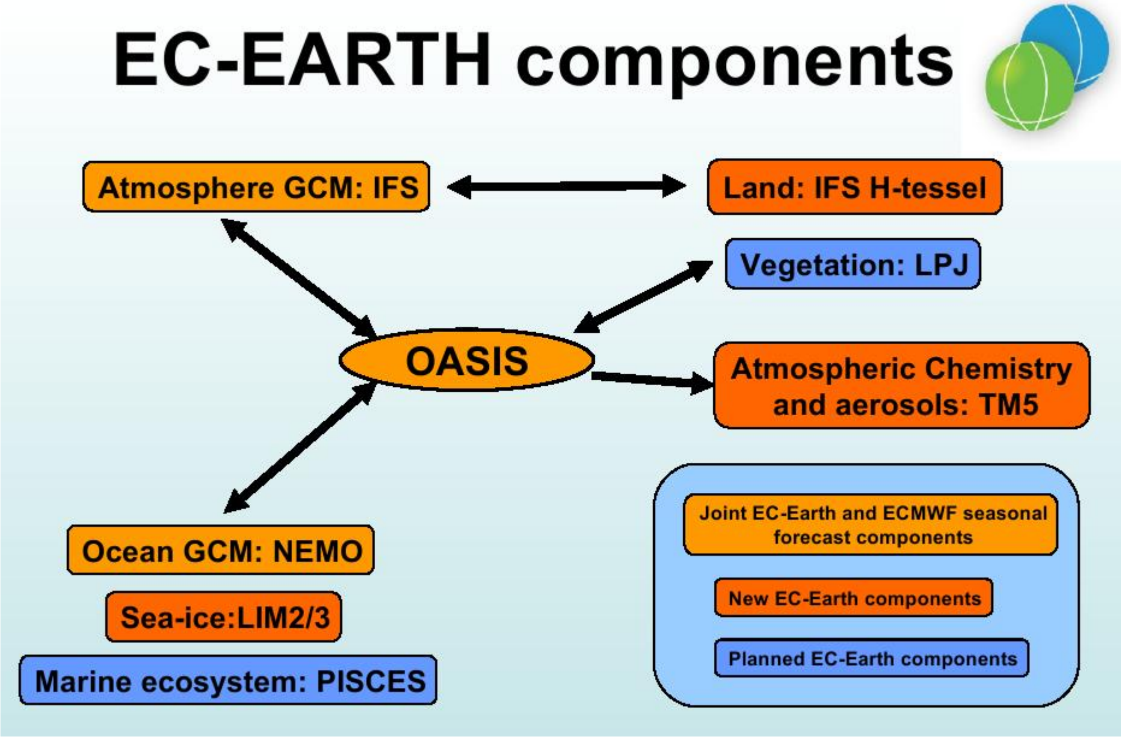
# The EC-Earth model

# EC-Earth

- Earth System Model
- Reliable in-house predictions of global climate change
- Part of a Europe-wide consortium
- Being used in large European projects
  - EMBRACE
  - EUPORIAS
  - IS-ENES
  - SPECS
  - EsiWACE
  - PRIMAVERA
- 3.1 version → IFS + NEMO-LIM + OASIS

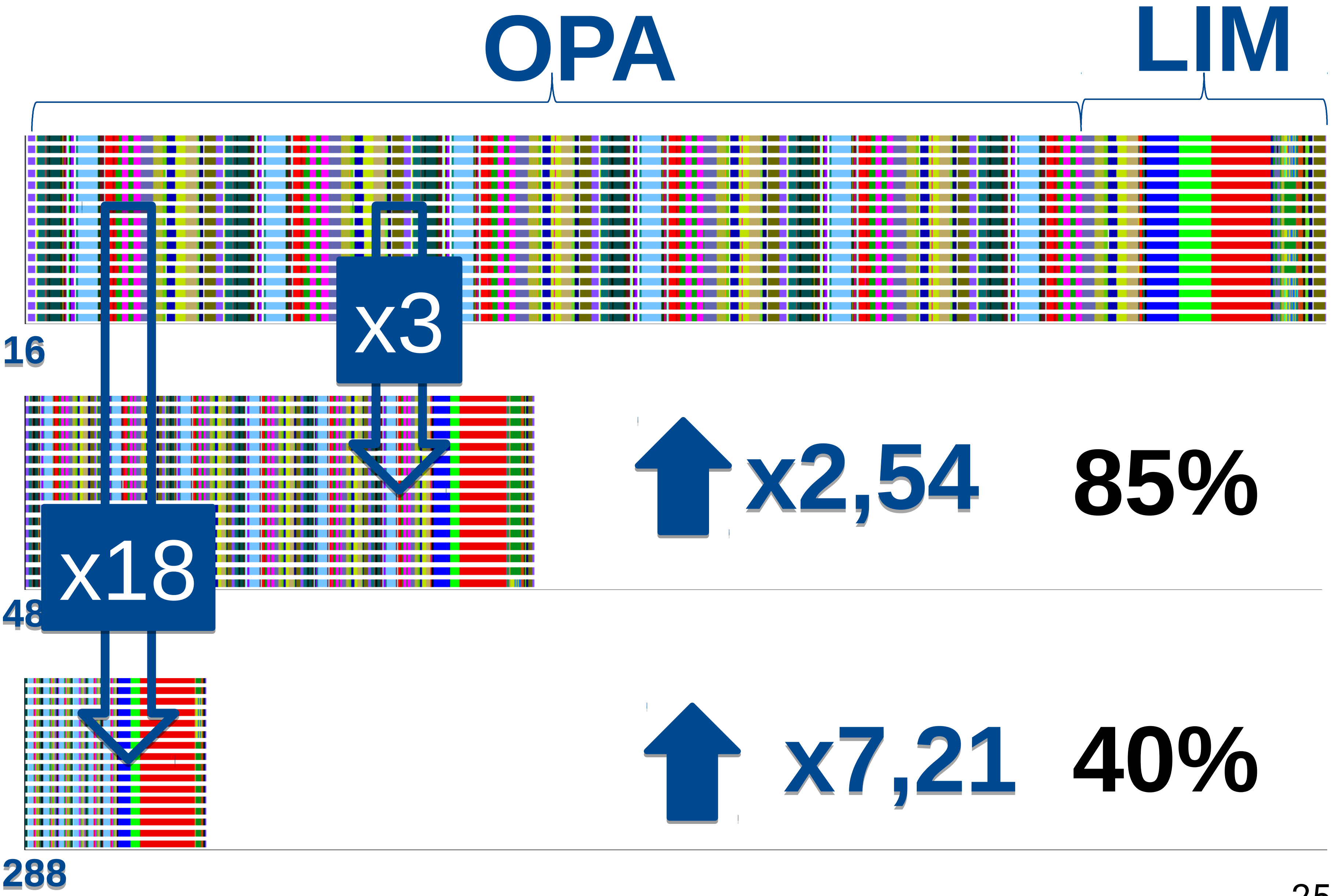
# EC-Earth v3.2

- EC-Earth 3.2
  - Integrated Forecasting System (IFS 36r4) as atmosphere model
  - Nucleus for European Modelling of the Ocean (NEMO 3.6) as ocean model
  - OASIS3-MCT coupler
  - Louvain-la-Neuve sea-Ice Model 3 (LIM3) as sea ice model



# NEMO model optimization

# NEMO model scalability

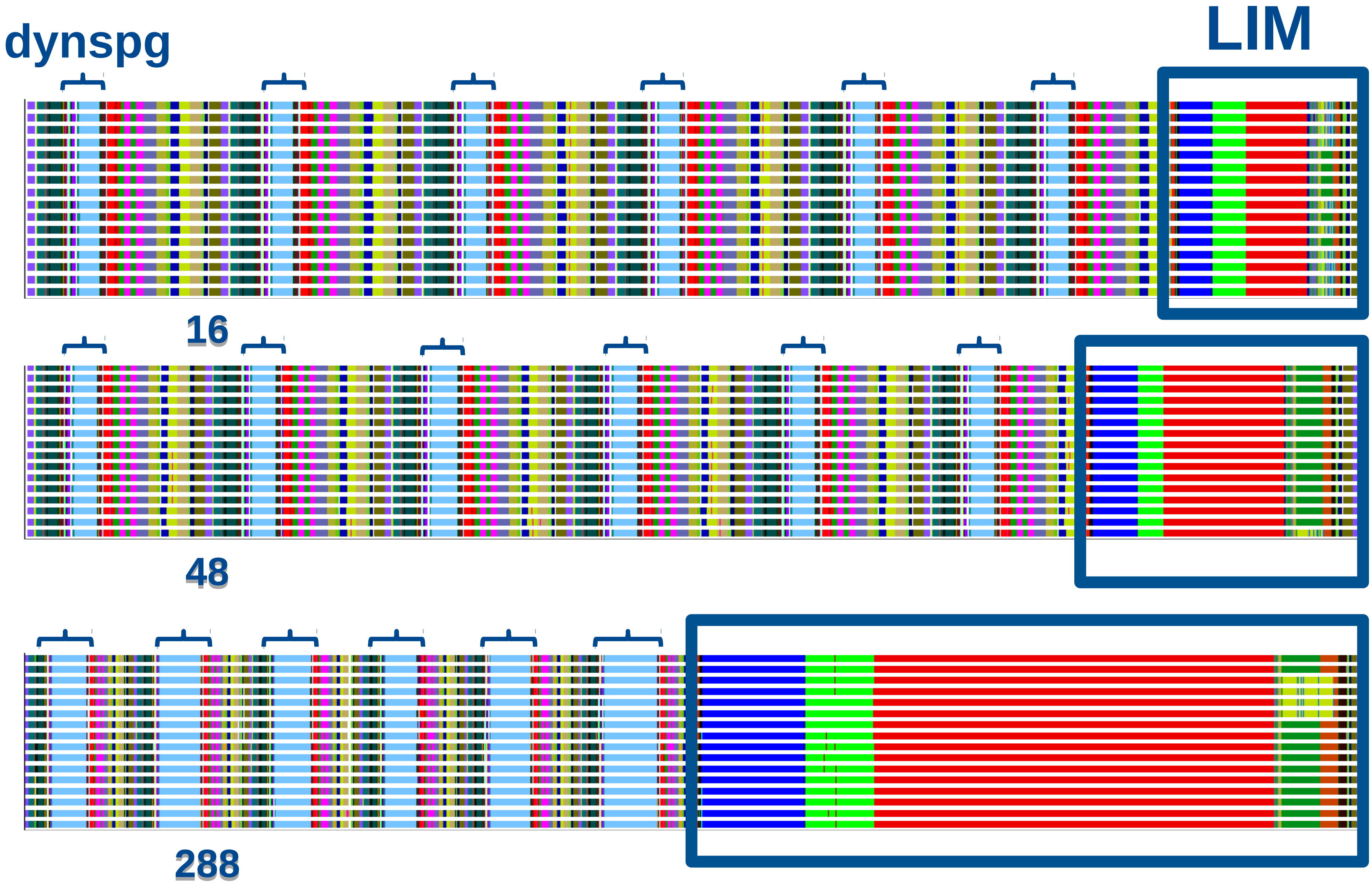


x2,54 85%

x7,21 40%

*Timelines have the same duration

# LIM and dynspg as bottlenecks



dynspg

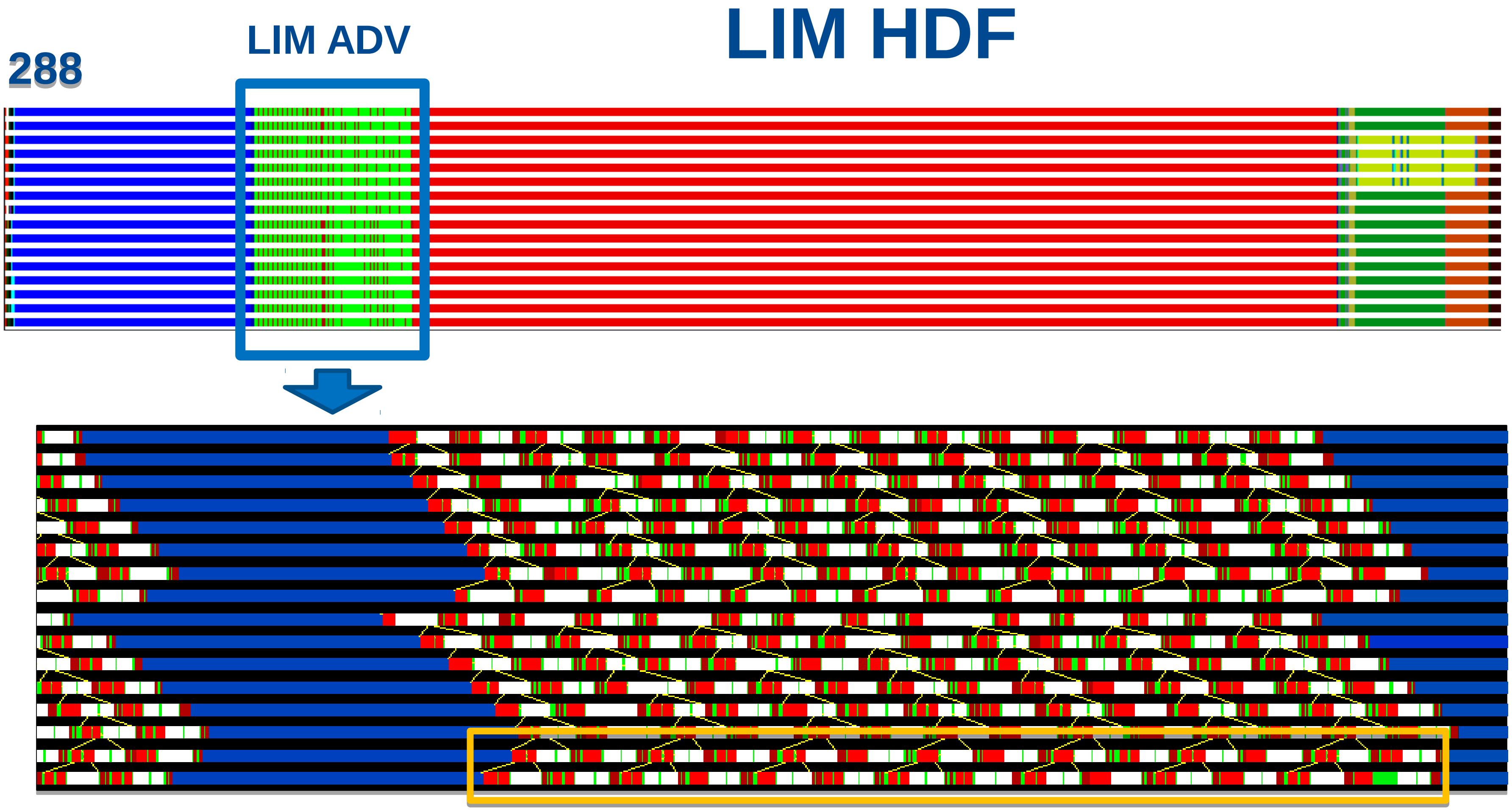
LIM

16

48

288

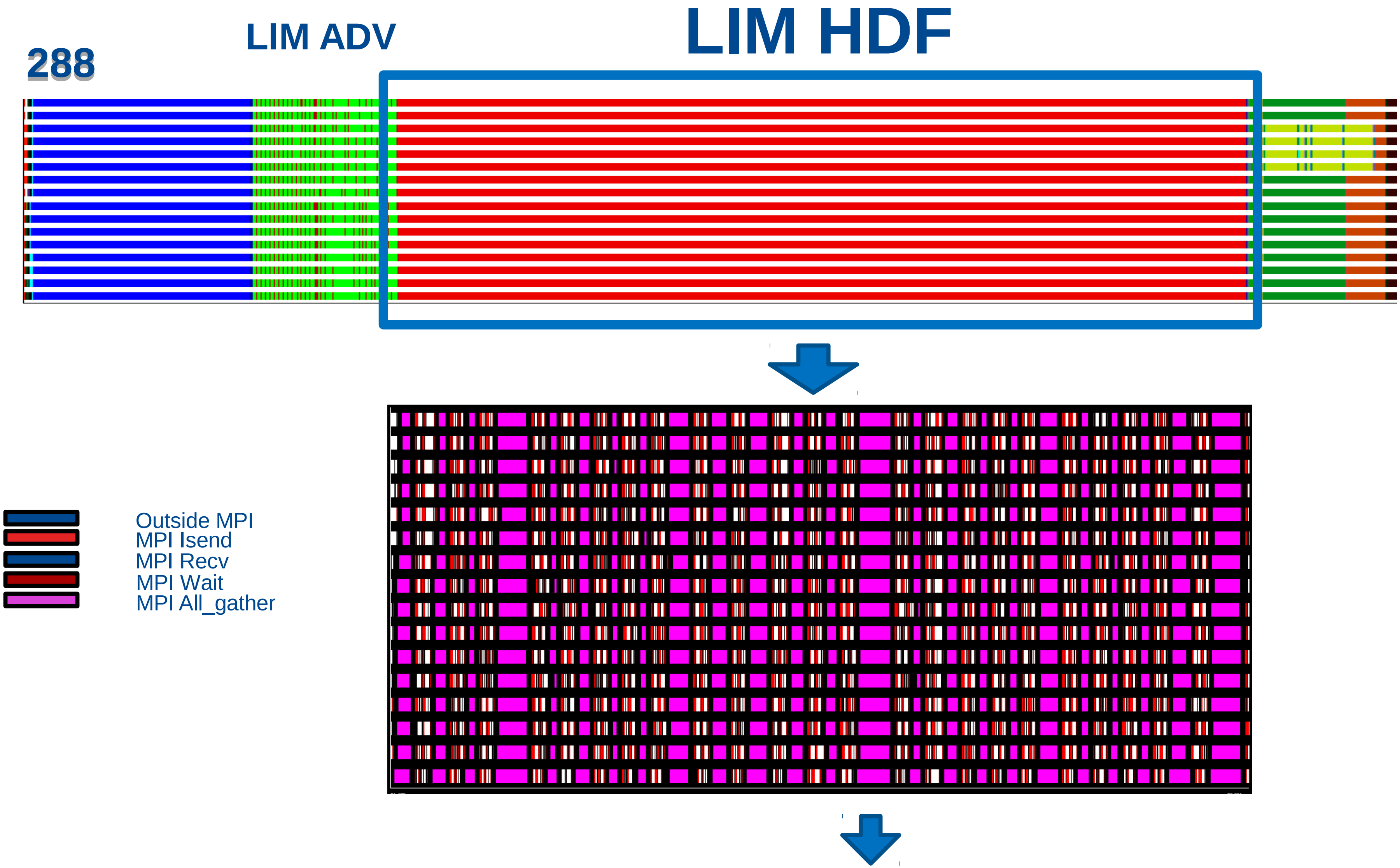
# Sea ice advection



🟦 Outside MPI
🟥 MPI Isend
⬜ MPI Recv
🟥 MPI Wait

7 border interchanges

# Sea ice horizontal diffusion



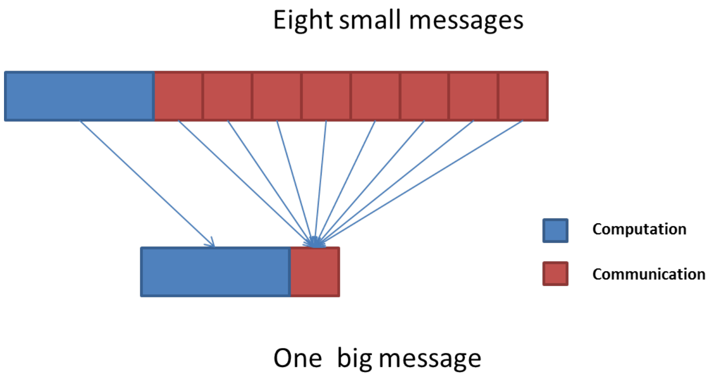
Only **20%** of the time invested on **computation**

Global Communication at **every** loop **iteration** → **60%** of the time

# Optimizations

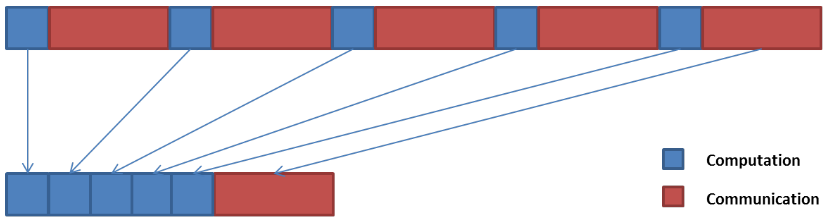
## MPI message packing

Taking in account that NEMO is really sensitive to latency, messages aggregation is the best way to reduce the time invested in communications. Therefore, consecutive messages have been packed wherever the computational dependencies allow to do so.



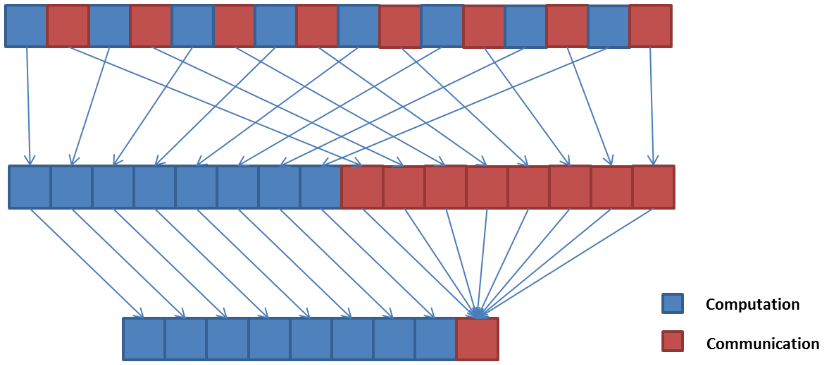
## Convergence check reduction

Some routines use collective communications to perform a convergence check in iterative solvers. The cost of this verifications is really high, reaching a 66% of the time. Wherever the model allowed it, we reduced the frequency of this verifications in order to increase parallel efficiency.



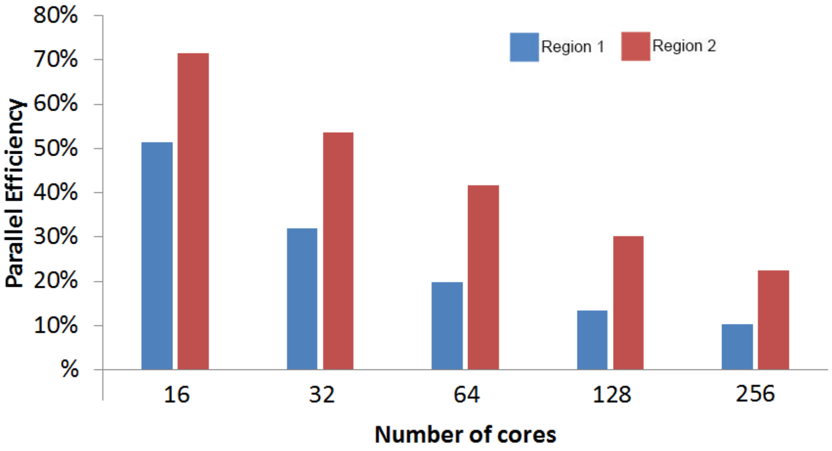
## Reordering

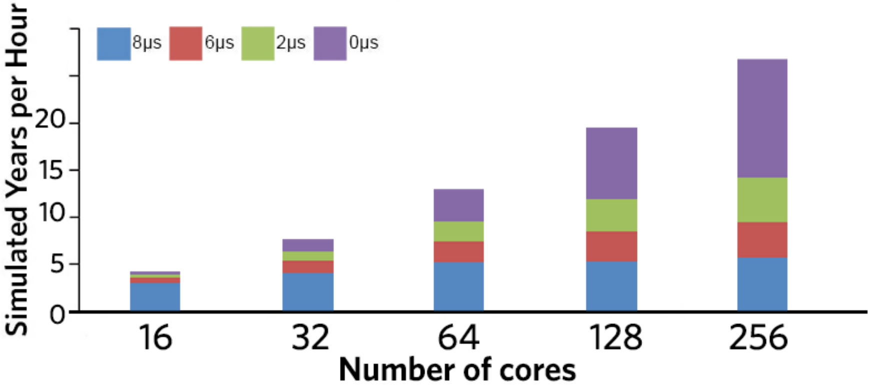
In order to apply the message packing optimization to as many routines as it was possible, it was necessary to rearrange some computation and communication regions, taking into account the dependencies between them , to reduce the number of messages. This way it was possible to compute (and communicate) up to 41 variables at the same time, resulting in a dramatic reduction of the granularity.



# NEMO performance analysis

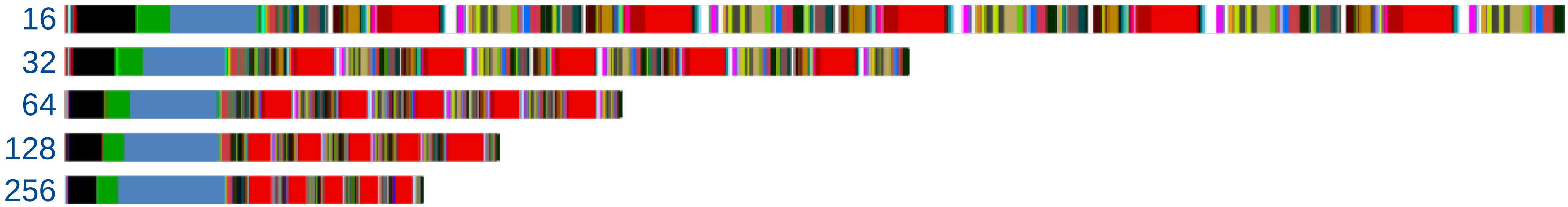
- However, communications are the main performance problem. Even in the 16-core case parallel efficiency is really bad.
- The figure at the right shows how sensitive the model is to network latency.
- Communications efficiency drops much more faster than computational.



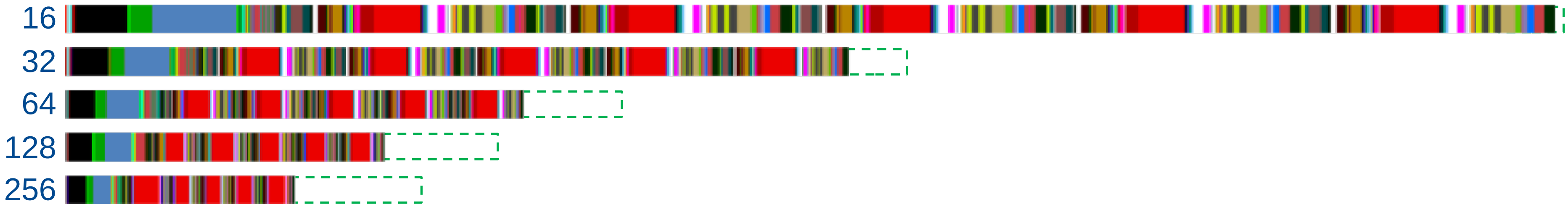


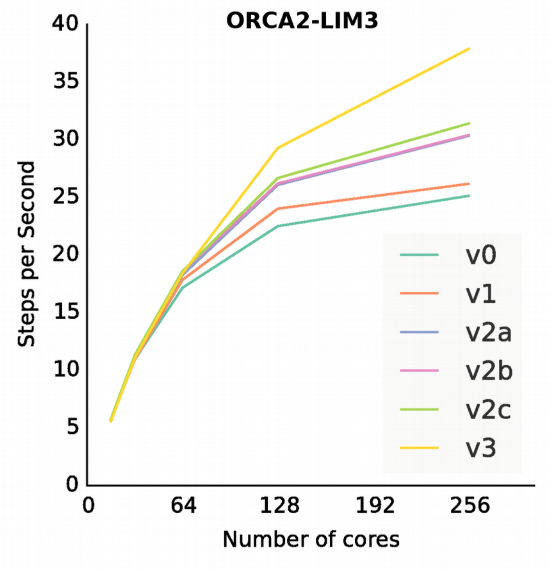
# NEMO 3.6 optimization: Results

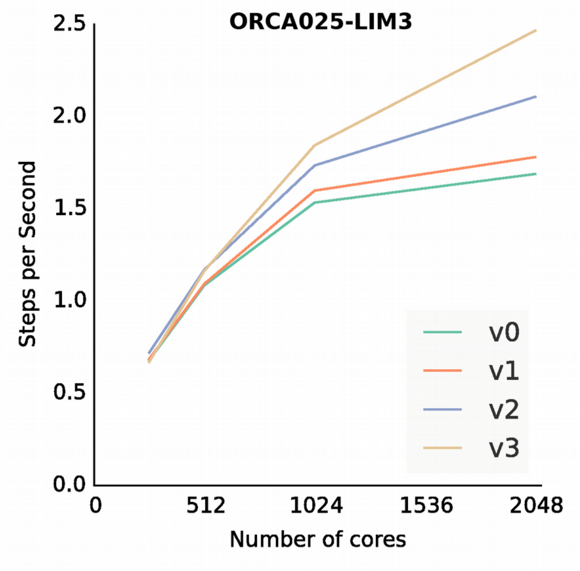
## Original code



## Optimized code



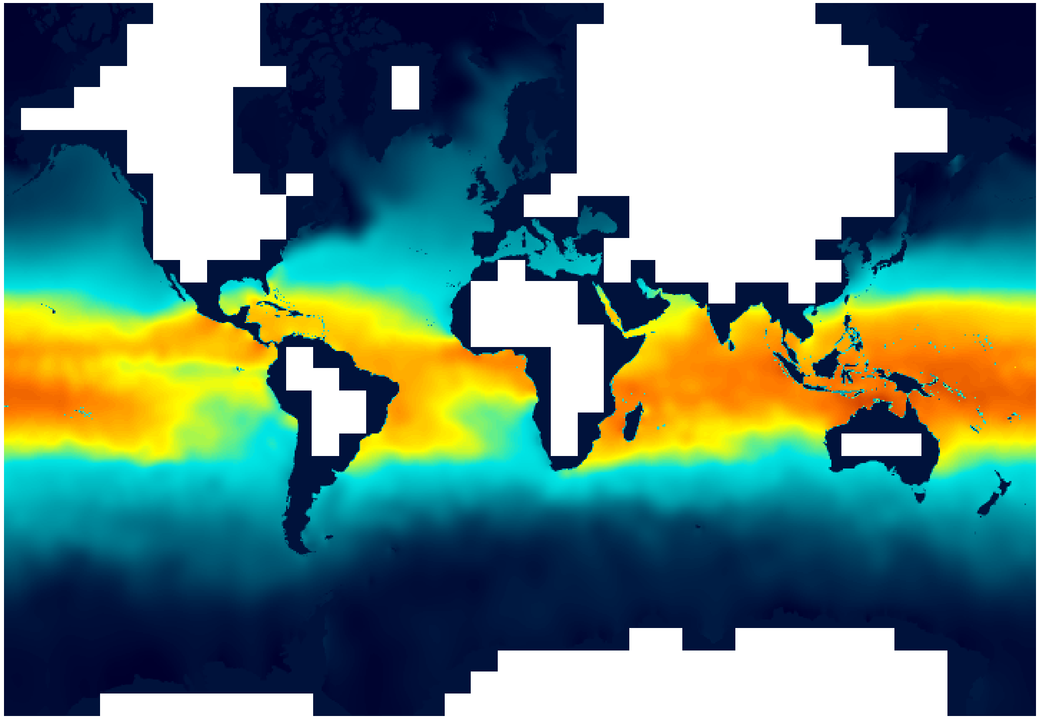




V0 → Original
V1 → Message packing
V2 → Conv. Check reduction
V3 → Reordering

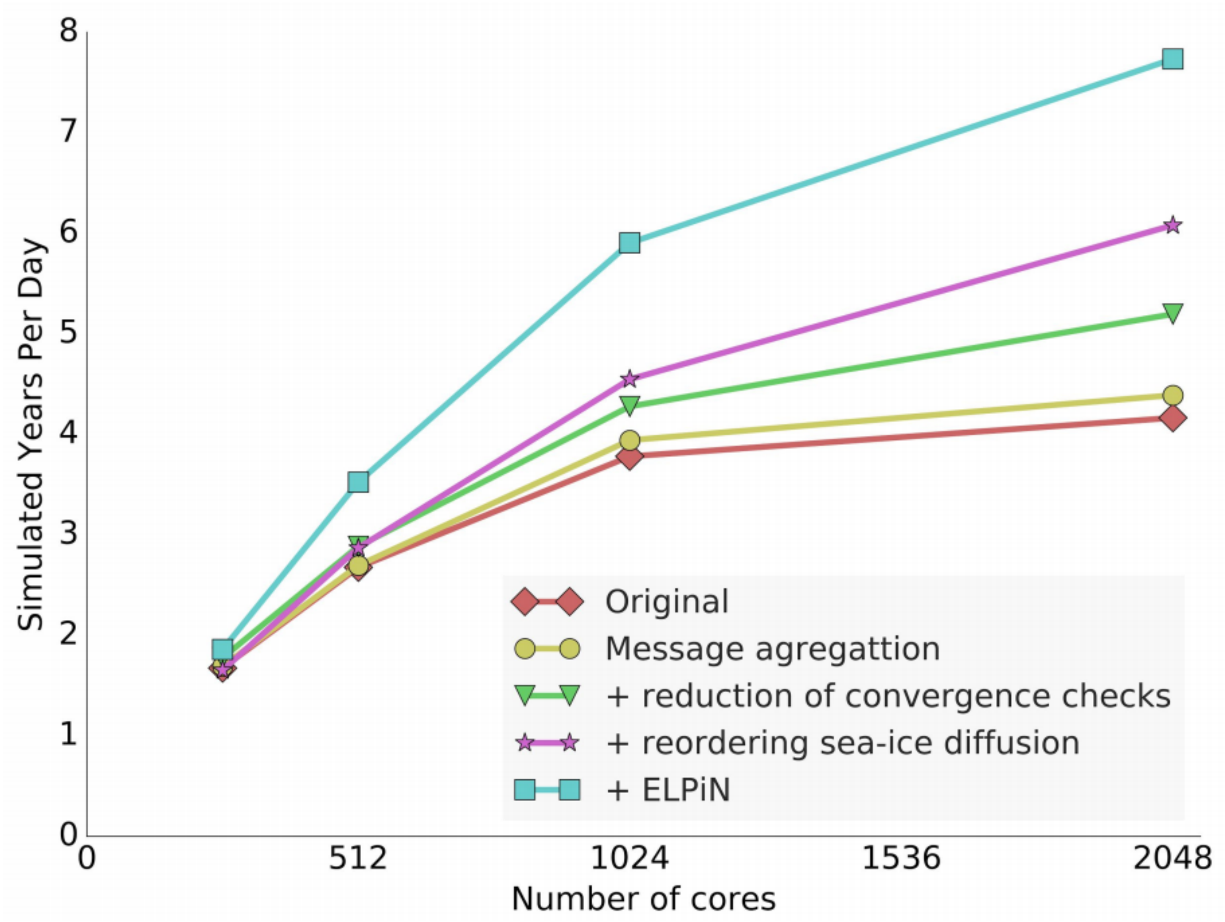
# ELPiN

- A tool that allows to find proper namelist parameters to exclude land-only processes in NEMO simulations
- NEMO decomposes automatically the domain:
  - Computes and communicates in land-only processes and then discards the result → waste of resources



- ORCA025 domain decomposed in 1287 sub-domains
- 312 are land-only and therefore removed (24% of the total grid)

# ELPiN

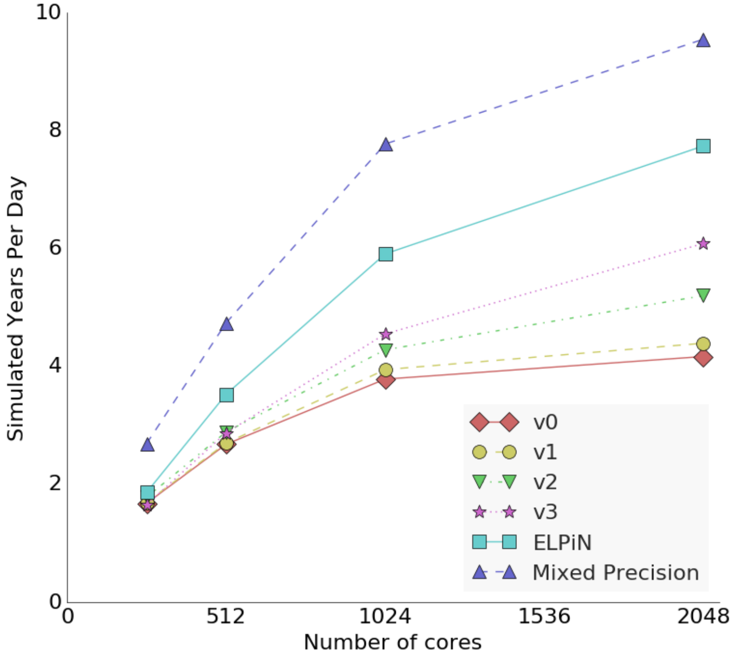


*Impact of optimizations done on the NEMO model for an ORCA025-LIM3 simulation*

# Use of single precision in NEMO

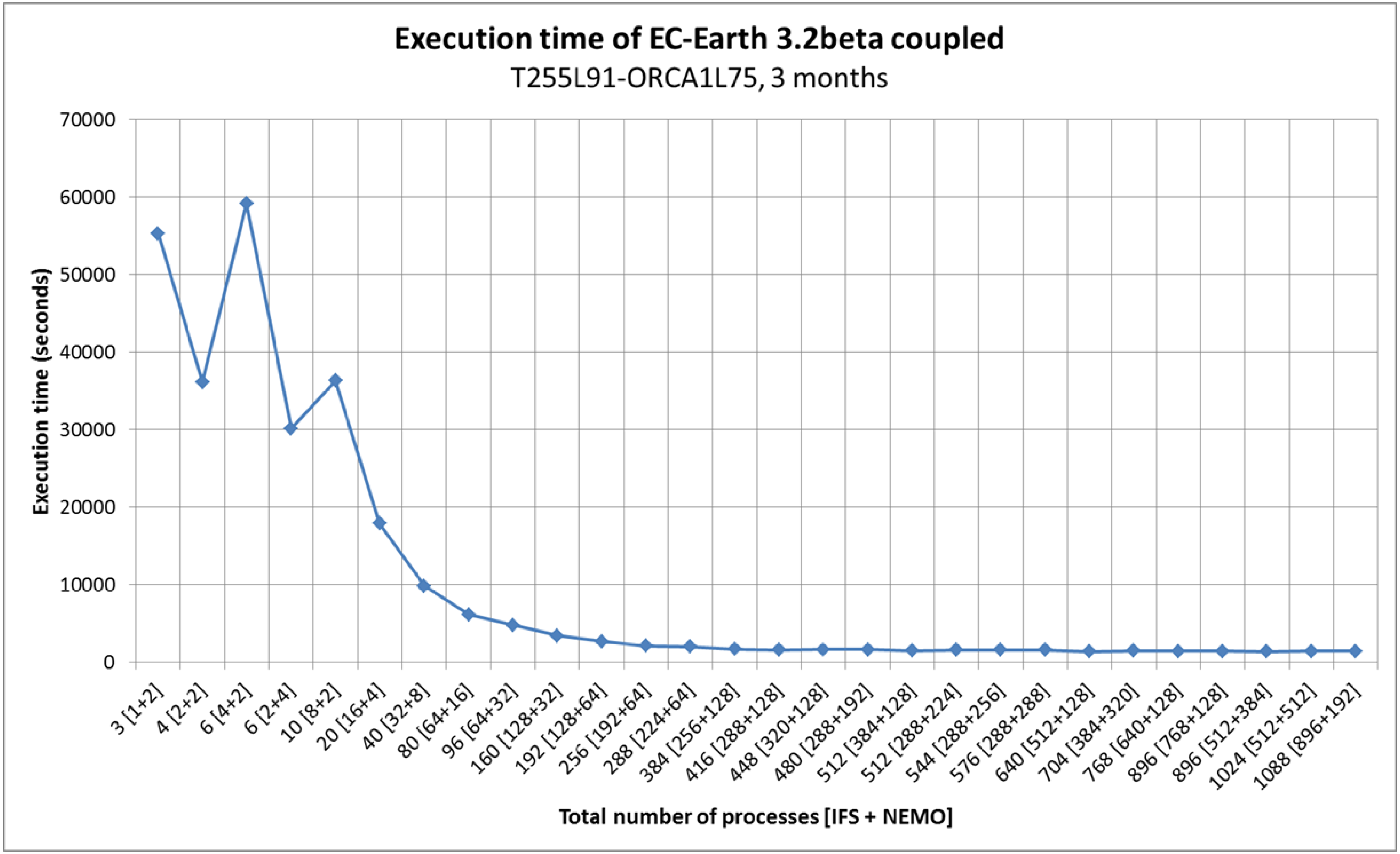
- BSC began an exploratory study to know which impact in computational performance may have a reduction of the precision in NEMO.
- First results show a 40% improvement on ORCA025-LIM3, by only introducing mixed precision in the ocean side.
- Further studies may determine which parts of the code are tolerant to a reduction in precision.

*“At present, there is no other measure within our reach that could have a greater impact on performance.”*



# Optimization of the EC-Earth coupled model

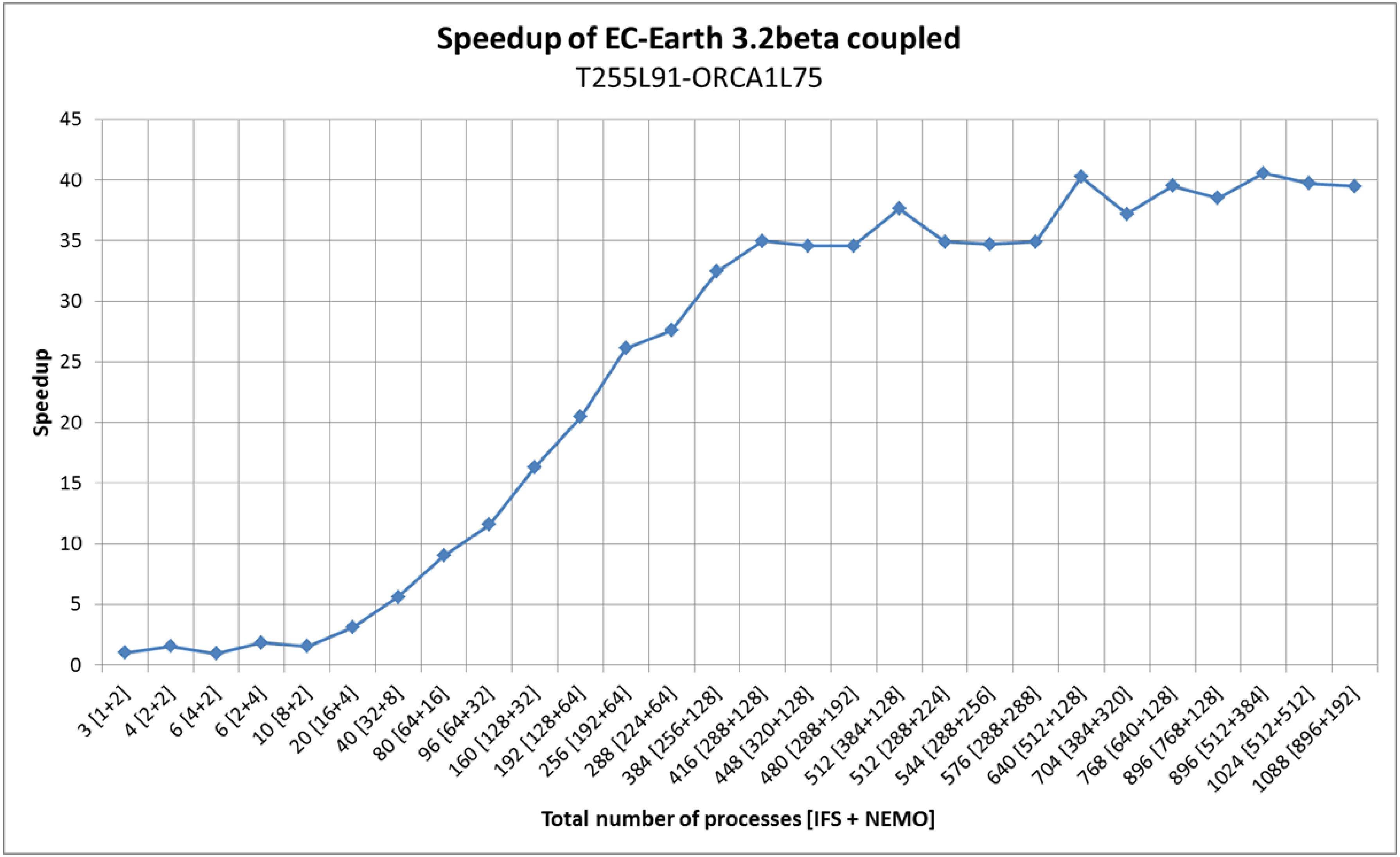
# EC-Earth scalability



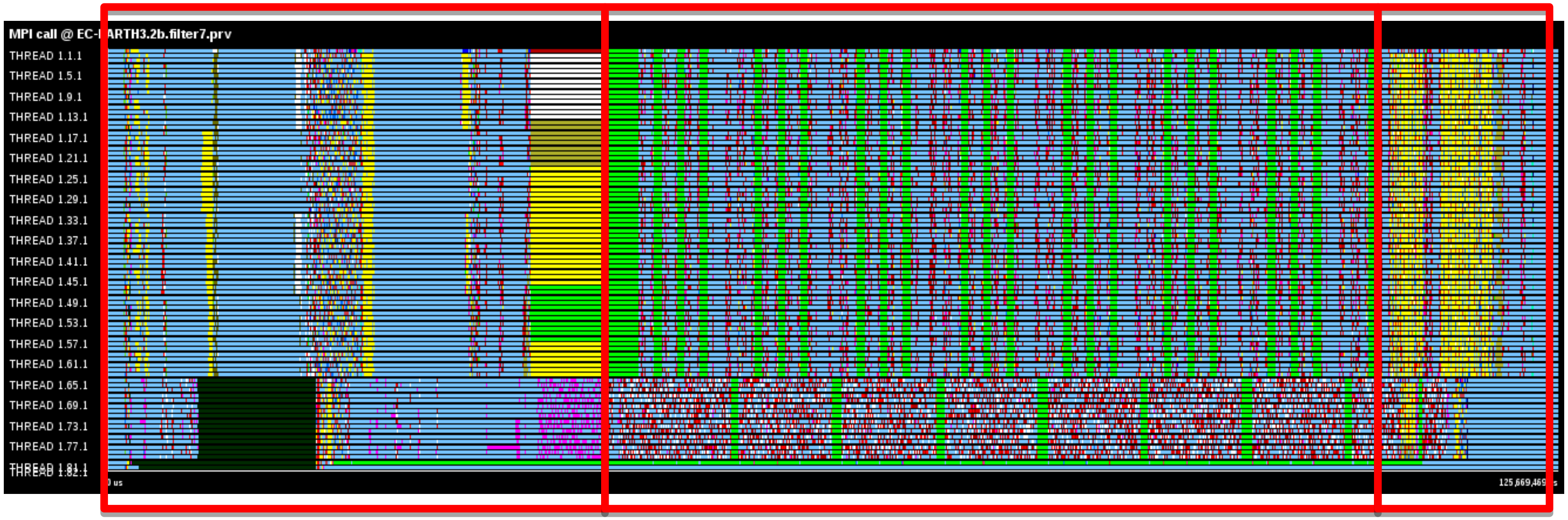
**Execution time of EC-Earth 3.2beta coupled**

T255L91-ORCA1L75, 3 months

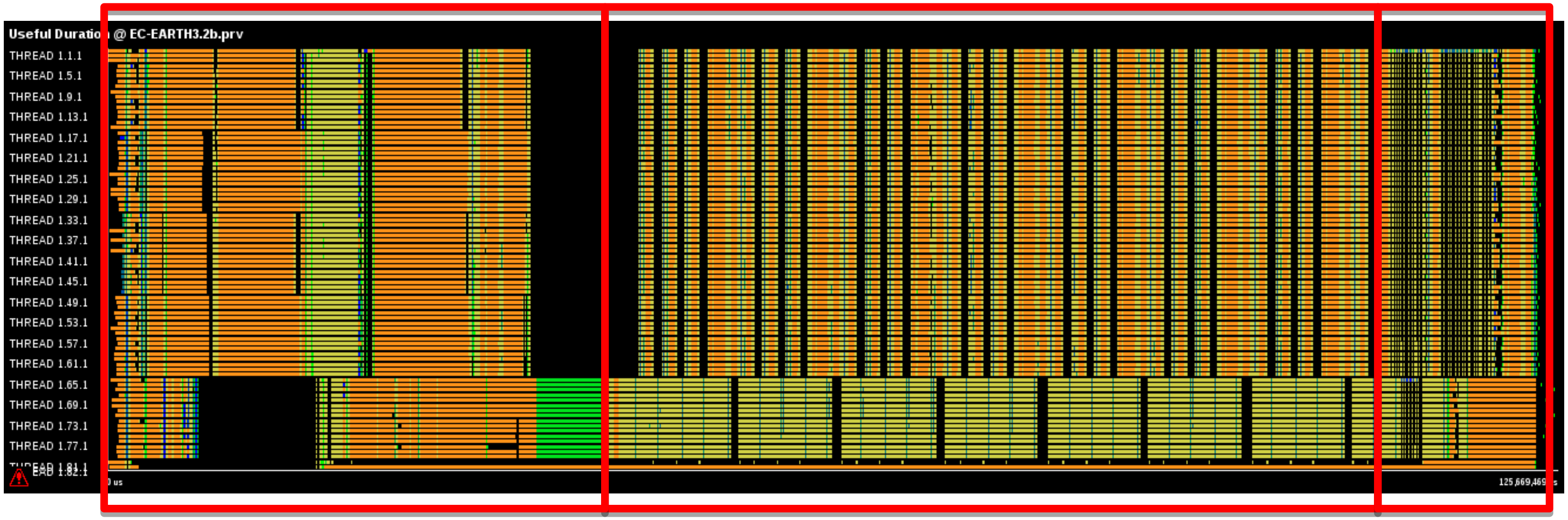
# EC-Earth scalability



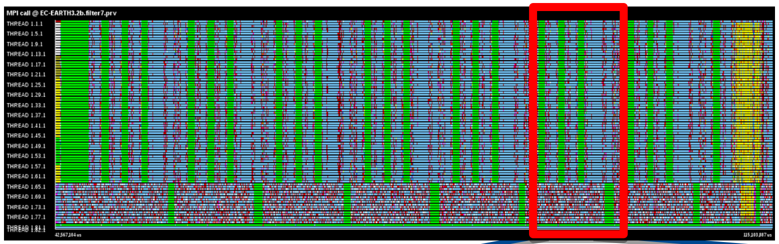
# EC-Earth trace analysis


MPI call @ EC-EARTH3.2b.filter7.prv

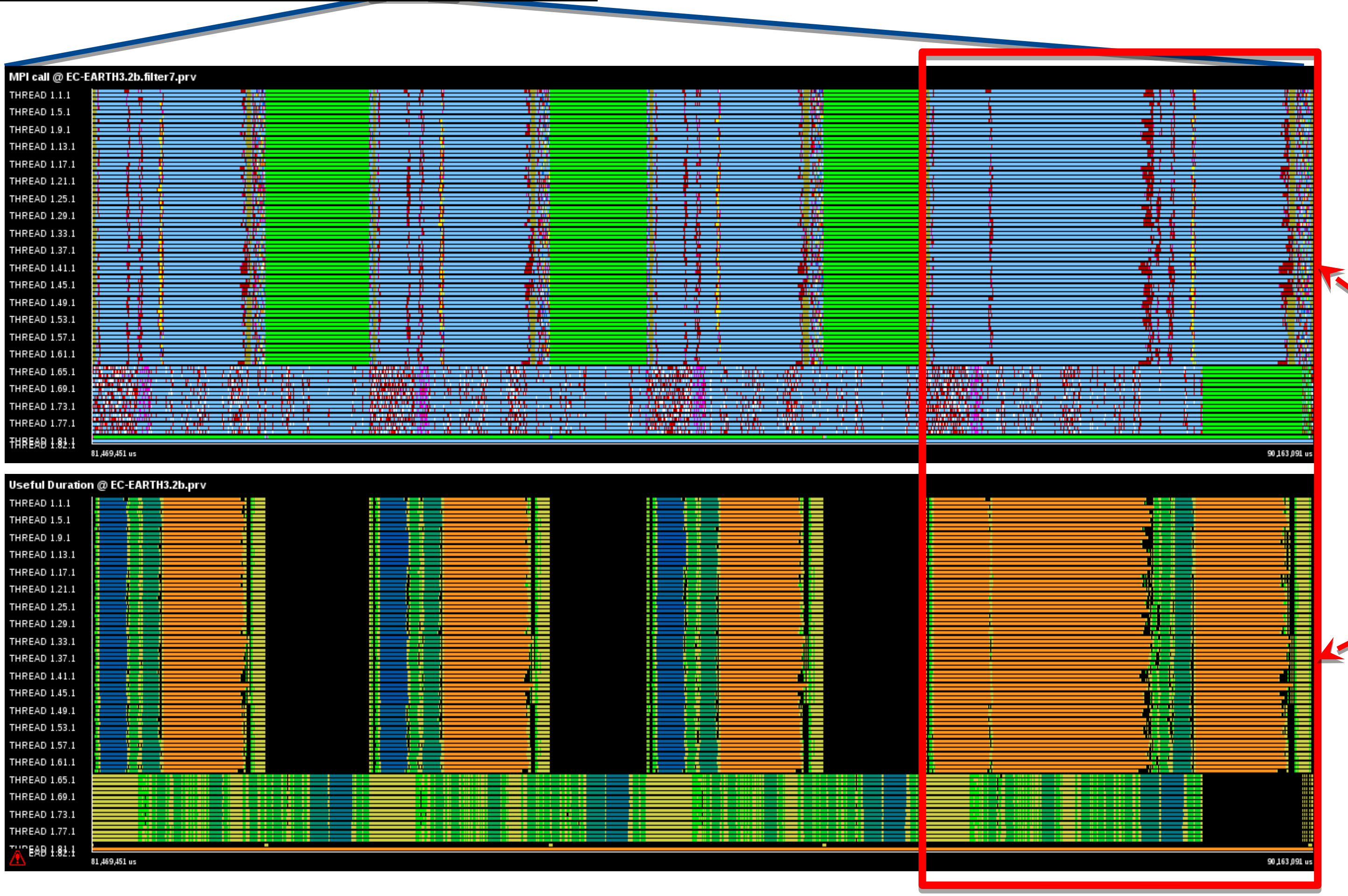
Initialization Time steps Finalization


Useful Duration @ EC-EARTH3.2b.prv

# EC-Earth trace analysis



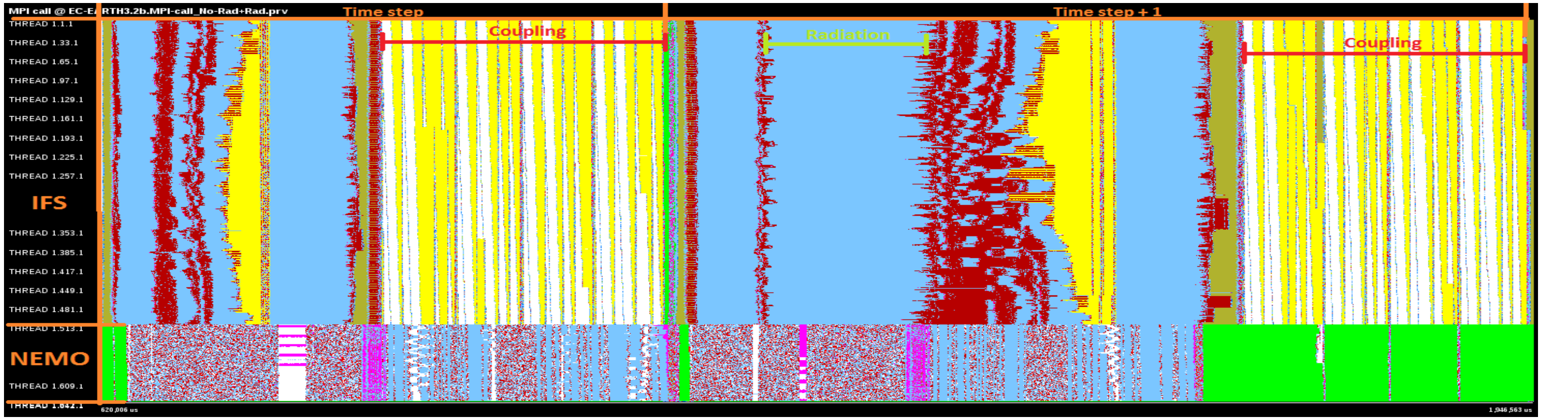
1 of every 4 time steps, IFS executes radiation routines, where NEMO has to wait



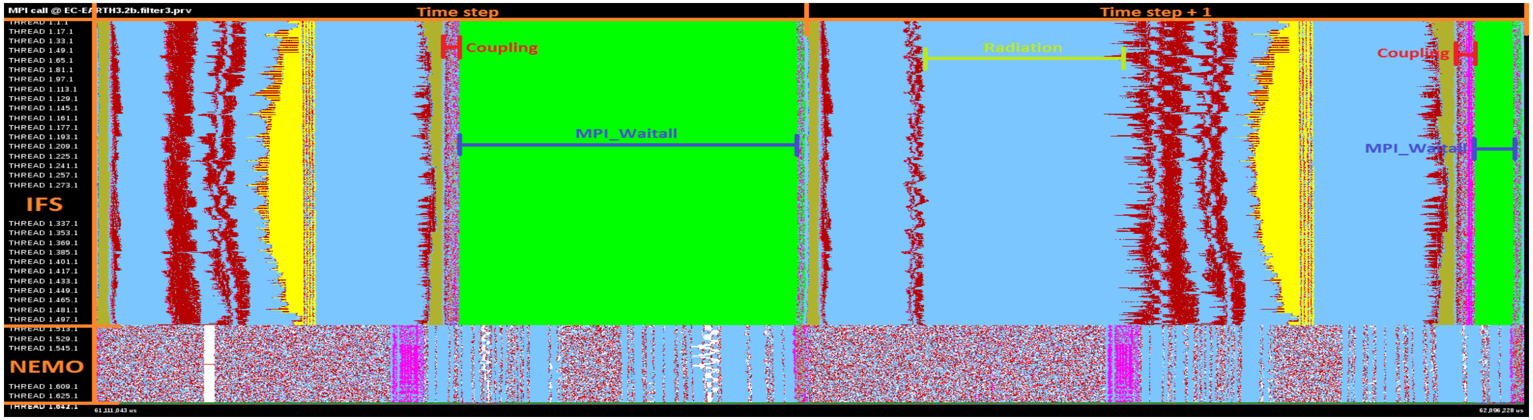
Time step with radiation

# EC-Earth coupling optimizations

Collaboration with the EC-Earth Technical Working Group to improve the model execution



Success case: coupling field gathering and OPT option of OASIS coupler for global conservative transformations



Coupling process improved up to 90%

Optimizations included in trunk EC-Earth 3.2.2, substantially benefiting CMIP6 simulations

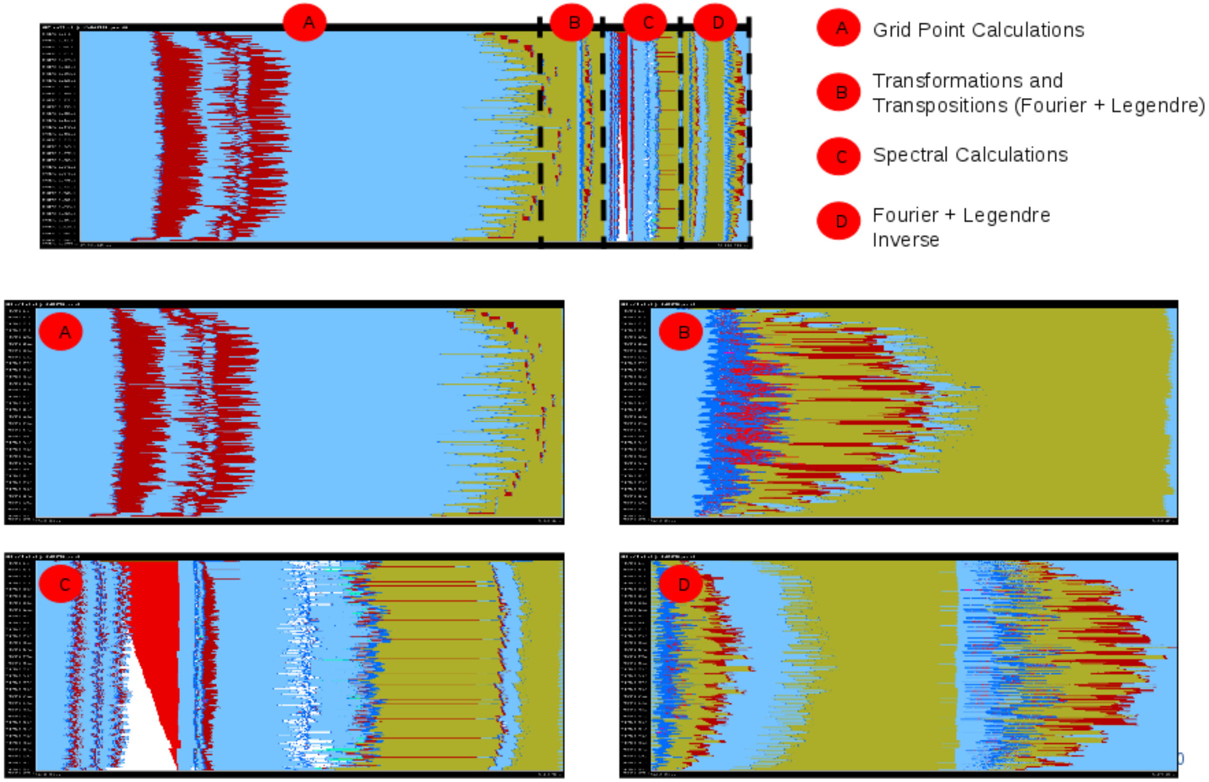
# ECMWF collaboration

- BSC is collaborating with the Research department at ECMWF to improve the computational performance of IFS/OpenIFS models.

- Key activities:
  - Contribution for the next official release of IFS to use the BSC tools (Extrae, Paraver…).
  - IFS/OpenIFS performance analysis and optimization.
  - IFS-NEMO coupling comparison:
    - Independent components (via OASIS) vs single binary.

# IFS performance analysis

A profiling analysis of IFS43r1 and OpenIFS40r1 was done using BSC Tools. These analysis can be useful to highlight which parts of the code could be improved in the future.



A Grid Point Calculations

B Transformations and Transpositions (Fourier + Legendre)

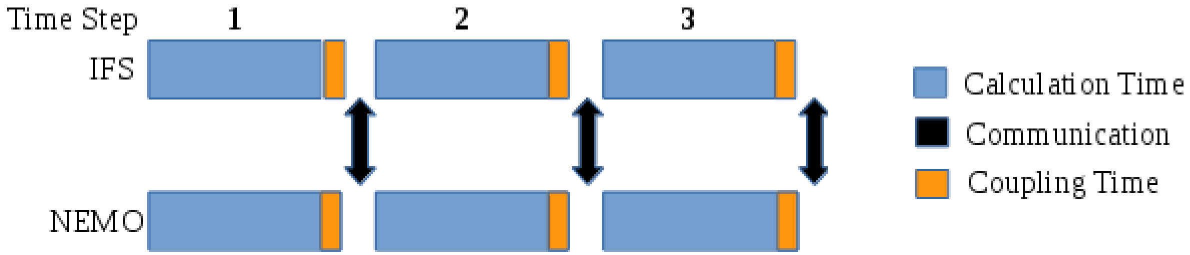
C Spectral Calculations

D Fourier + Legendre Inverse

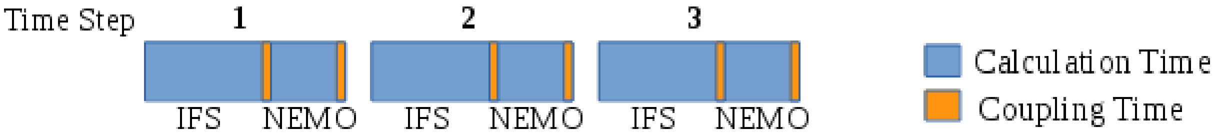
Example of the IFS profiling study using Paraver for each phase of IFS43r1

# EC-Earth coupling

- BSC and ECMWF will work together to evaluate the advantages and disadvantages of coupling the ocean component (NEMO) to IFS as:



Independent components



Using a single binary

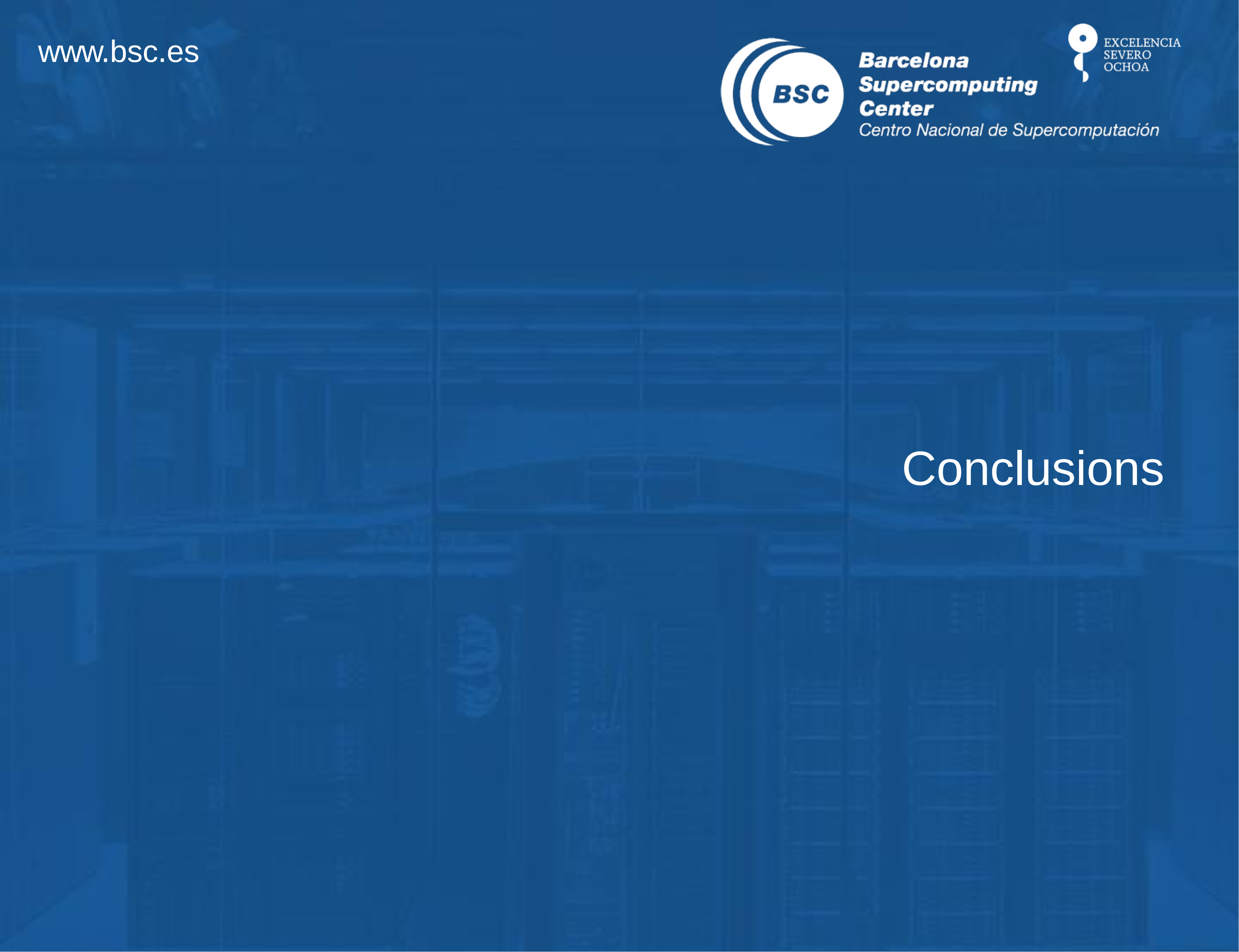
# OpenIFS

- Open version of the ECMWF model
  - Integrated Forecasting System (IFS)
  - Single column model (SCM)
  - Offline-surface model (OSM)

- Currently working with v-1 operational version (40r1)

- Hybrid parallelization (MPI+OpenMP)

- Next plans
  - Run a 1km global configuration (T7999) in ESiWACE CoE
  - Port to OmpSs
  - Apply Dynamic Load Balancing library (DLB)
  - Adding XIOS support



# Conclusions

# Conclusions

- **Trace analysis** can guide the users in **understanding** their code’s behaviour and efficiency

- **Performance tools** help in finding specific code parts that should be improved and which is the cause of the performance degradation

- A precise analysis and prediction can generate ideas that direct the **restructuring** of the application in the most productive way

- **Little changes** in the configuration can significantly **improve the performance**

- These tools provide the information but its **user’s task** to get conclusions from the different metrics

www.bsc.es

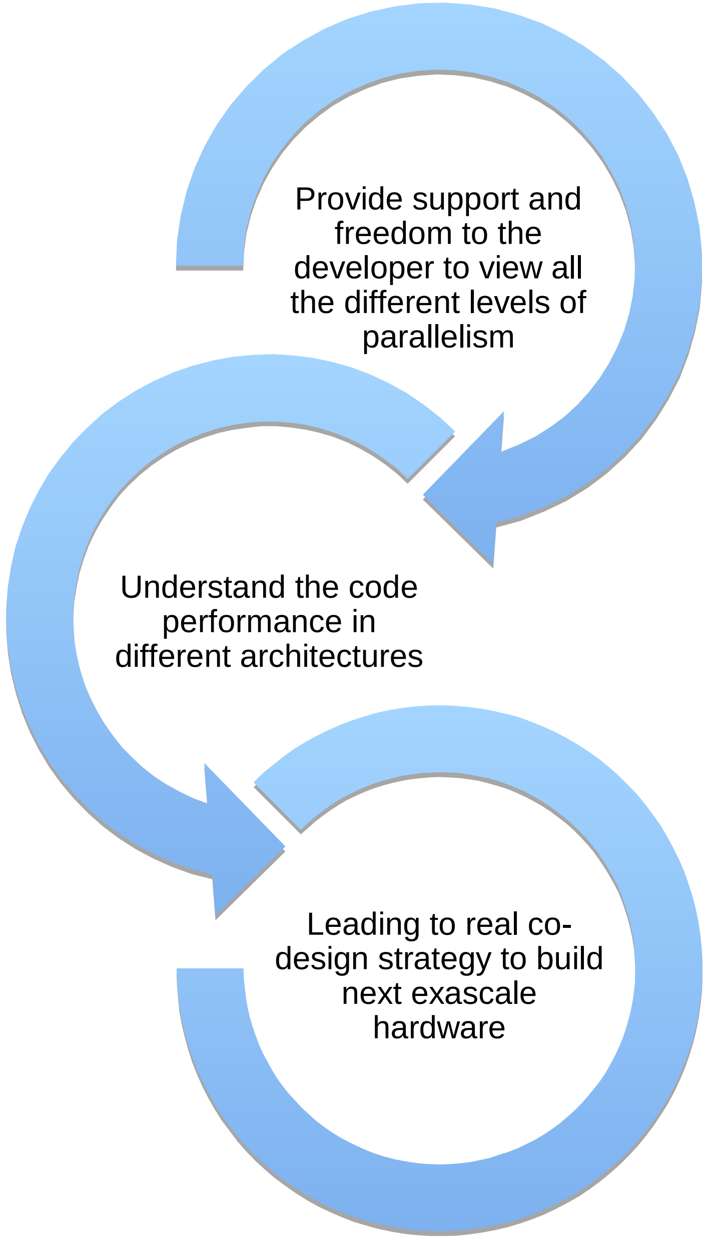
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

# Thank you!

For further information please contact
miguel.castrillo@bsc.es

# Introduction



Provide support and freedom to the developer to view all the different levels of parallelism

Understand the code performance in different architectures

Leading to real co-design strategy to build next exascale hardware

# Methodology

1. Mathematical study
   - Some methods could be better than others
     - Discretization used (explicit, implicit, semi-implicit…)
     - Parallel adaptation (solvers, preconditioners…)
   - How to implement new algorithms for new architectures

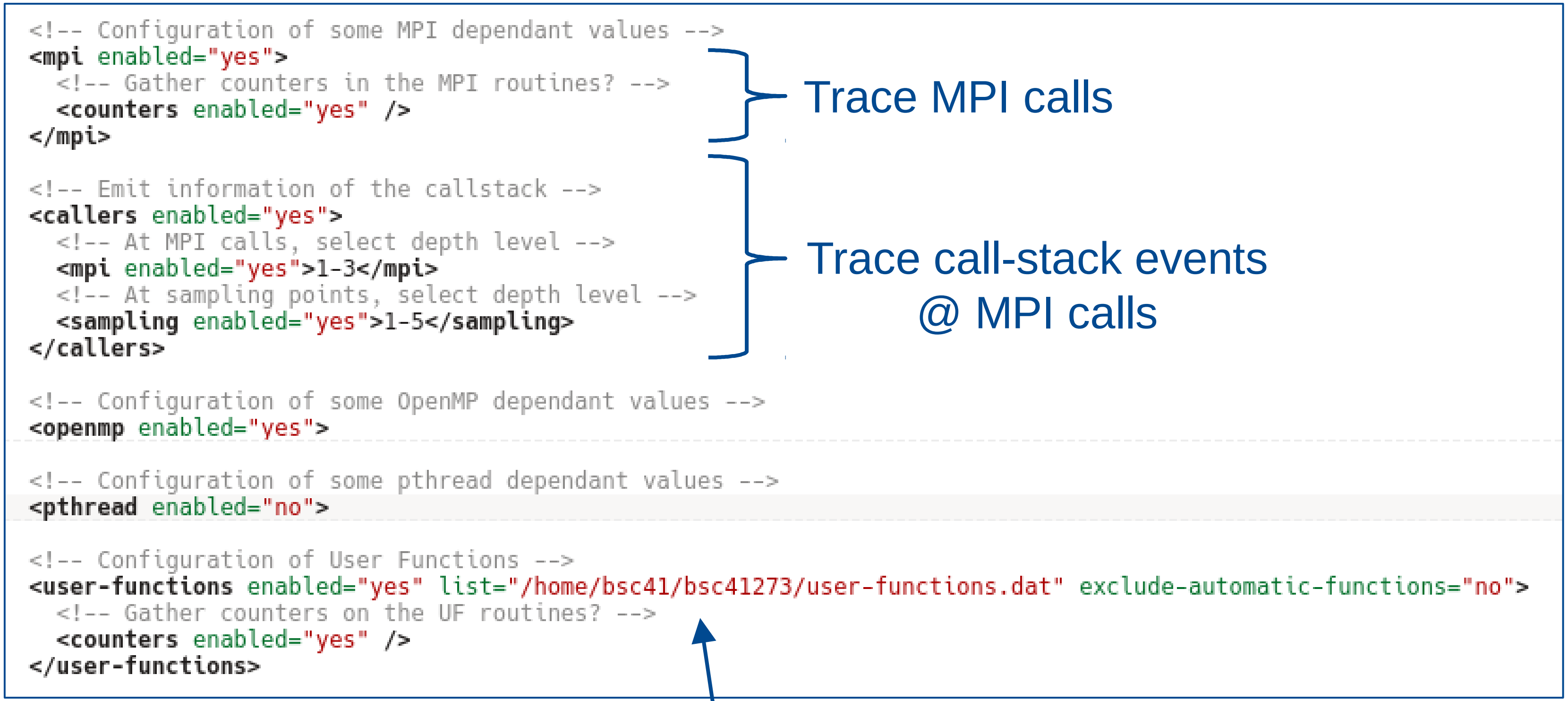
2. Computational study
   - Achieve load balance among components
   - Reduce overhead introduced by parallel applications
   - Ensure the computational algorithm takes advantage of the architecture

3. Profiling Study
   - General profiling
   - Profiling applied to Earth System Models

# Methodology

1. Introducing optimizations
   - Improvement of the mathematical and/or computational algorithm
     - Apply scientific methods which are found in the literature
     - Improve the method using a new approach
   - Revolution: Create a new (and better) algorithm taking into account the research line followed
2. Reproducibility study
   - Evaluate if the accuracy and reproducibility of the model is similar using or not the optimizations proposed
   - Take into account the nature of climate models
     - How to evaluate, in parallel executions, if the differences between runs are significant or not.

# Extrae configuration: extrae.xml



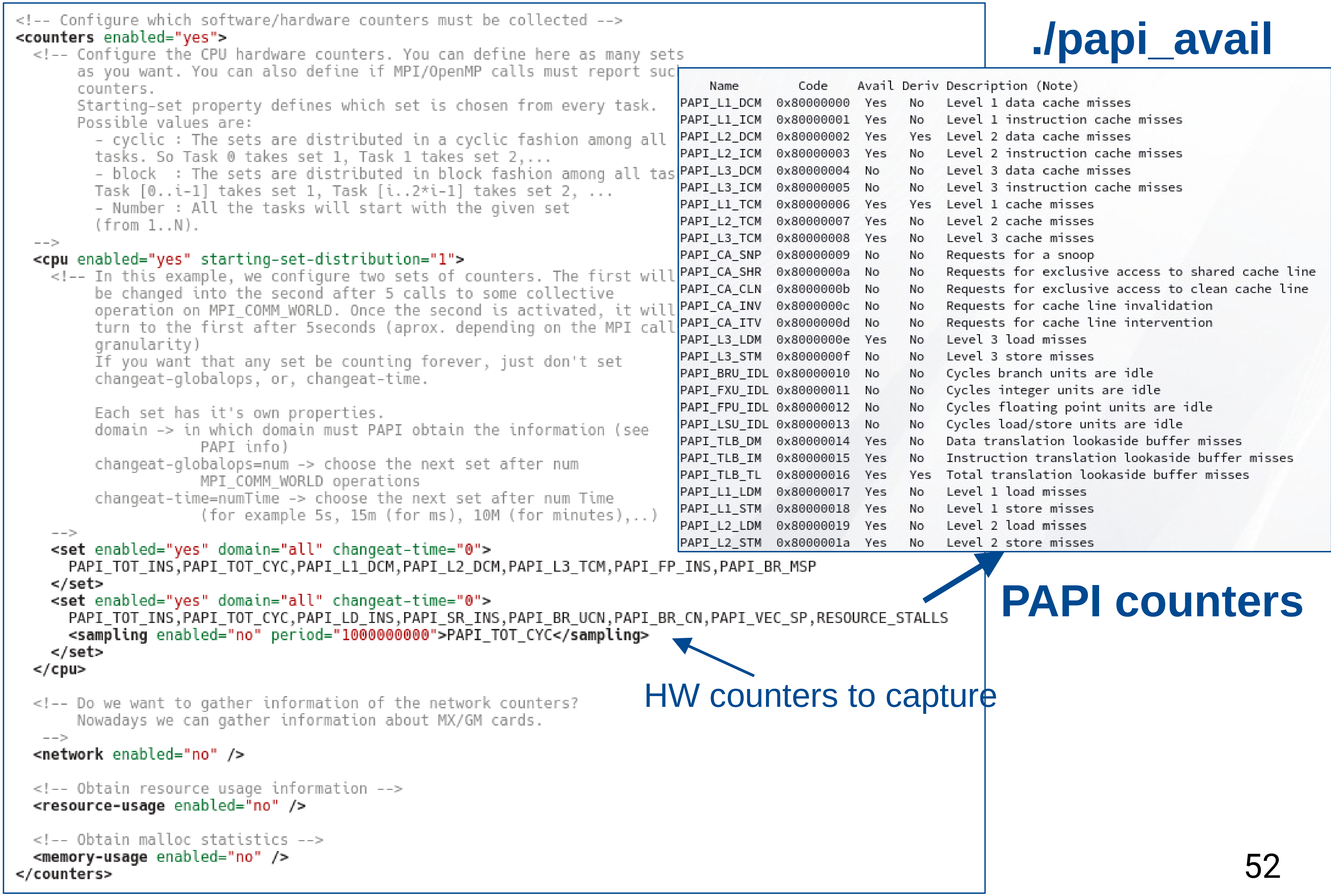
```
<!-- Configuration of some MPI dependant values -->
<mpi enabled="yes">
  <!-- Gather counters in the MPI routines? -->
  <counters enabled="yes" />
</mpi>

<!-- Emit information of the callstack -->
<callers enabled="yes">
  <!-- At MPI calls, select depth level -->
  <mpi enabled="yes">1-3</mpi>
  <!-- At sampling points, select depth level -->
  <sampling enabled="yes">1-5</sampling>
</callers>

<!-- Configuration of some OpenMP dependant values -->
<openmp enabled="yes">

<!-- Configuration of some pthread dependant values -->
<pthread enabled="no">

<!-- Configuration of User Functions -->
<user-functions enabled="yes" list="/home/bsc41/bsc41273/user-functions.dat" exclude-automatic-functions="no">
  <!-- Gather counters on the UF routines? -->
  <counters enabled="yes" />
</user-functions>
```

Trace MPI calls

Trace call-stack events
@ MPI calls

Trace user functions (from list)

# Extrae configuration: extrae.xml



```
<!-- Configure which software/hardware counters must be collected -->
<counters enabled="yes">
  <!-- Configure the CPU hardware counters. You can define here as many sets
       as you want. You can also define if MPI/OpenMP calls must report suc
       counters.
       Starting-set property defines which set is chosen from every task.
       Possible values are:
         - cyclic : The sets are distributed in a cyclic fashion among all
         tasks. So Task 0 takes set 1, Task 1 takes set 2,...
         - block  : The sets are distributed in block fashion among all tas
         Task [0..i-1] takes set 1, Task [i..2*i-1] takes set 2, ...
         - Number : All the tasks will start with the given set
         (from 1..N).
  -->
  <cpu enabled="yes" starting-set-distribution="1">
    <!-- In this example, we configure two sets of counters. The first will
         be changed into the second after 5 calls to some collective
         operation on MPI_COMM_WORLD. Once the second is activated, it will
         turn to the first after 5seconds (aprox. depending on the MPI call
         granularity)
         If you want that any set be counting forever, just don't set
         changeat-globalops, or, changeat-time.

         Each set has it's own properties.
         domain -> in which domain must PAPI obtain the information (see
                     PAPI info)
         changeat-globalops=num -> choose the next set after num
                     MPI_COMM_WORLD operations
         changeat-time=numTime -> choose the next set after num Time
                     (for example 5s, 15m (for ms), 10M (for minutes),..)
    -->
    <set enabled="yes" domain="all" changeat-time="0">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L1_DCM,PAPI_L2_DCM,PAPI_L3_TCM,PAPI_FP_INS,PAPI_BR_MSP
    </set>
    <set enabled="yes" domain="all" changeat-time="0">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_LD_INS,PAPI_SR_INS,PAPI_BR_UCN,PAPI_BR_CN,PAPI_VEC_SP,RESOURCE_STALLS
      <sampling enabled="no" period="1000000000">PAPI_TOT_CYC</sampling>
    </set>
  </cpu>

  <!-- Do we want to gather information of the network counters?
       Nowadays we can gather information about MX/GM cards.
   -->
  <network enabled="no" />

  <!-- Obtain resource usage information -->
  <resource-usage enabled="no" />

  <!-- Obtain malloc statistics -->
  <memory-usage enabled="no" />
</counters>
```

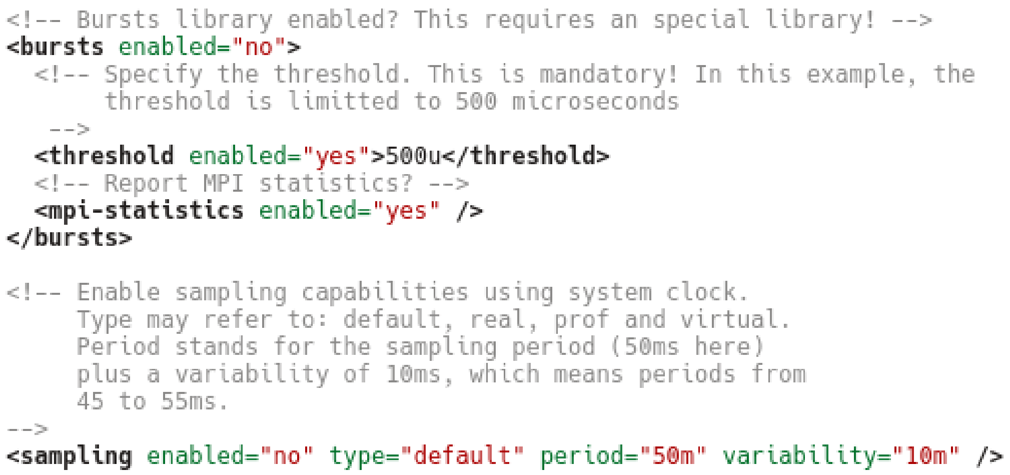
./papi_avail

```
    Name         Code      Avail Deriv Description (Note)
PAPI_L1_DCM  0x80000000  Yes   No   Level 1 data cache misses
PAPI_L1_ICM  0x80000001  Yes   No   Level 1 instruction cache misses
PAPI_L2_DCM  0x80000002  Yes   Yes  Level 2 data cache misses
PAPI_L2_ICM  0x80000003  Yes   No   Level 2 instruction cache misses
PAPI_L3_DCM  0x80000004  No    No   Level 3 data cache misses
PAPI_L3_ICM  0x80000005  No    No   Level 3 instruction cache misses
PAPI_L1_TCM  0x80000006  Yes   Yes  Level 1 cache misses
PAPI_L2_TCM  0x80000007  Yes   No   Level 2 cache misses
PAPI_L3_TCM  0x80000008  Yes   No   Level 3 cache misses
PAPI_CA_SNP  0x80000009  No    No   Requests for a snoop
PAPI_CA_SHR  0x8000000a  No    No   Requests for exclusive access to shared cache line
PAPI_CA_CLN  0x8000000b  No    No   Requests for exclusive access to clean cache line
PAPI_CA_INV  0x8000000c  No    No   Requests for cache line invalidation
PAPI_CA_ITV  0x8000000d  No    No   Requests for cache line intervention
PAPI_L3_LDM  0x8000000e  Yes   No   Level 3 load misses
PAPI_L3_STM  0x8000000f  No    No   Level 3 store misses
PAPI_BRU_IDL 0x80000010  No    No   Cycles branch units are idle
PAPI_FXU_IDL 0x80000011  No    No   Cycles integer units are idle
PAPI_FPU_IDL 0x80000012  No    No   Cycles floating point units are idle
PAPI_LSU_IDL 0x80000013  No    No   Cycles load/store units are idle
PAPI_TLB_DM  0x80000014  Yes   No   Data translation lookaside buffer misses
PAPI_TLB_IM  0x80000015  Yes   No   Instruction translation lookaside buffer misses
PAPI_TLB_TL  0x80000016  Yes   Yes  Total translation lookaside buffer misses
PAPI_L1_LDM  0x80000017  Yes   No   Level 1 load misses
PAPI_L1_STM  0x80000018  Yes   No   Level 1 store misses
PAPI_L2_LDM  0x80000019  Yes   No   Level 2 load misses
PAPI_L2_STM  0x8000001a  Yes   No   Level 2 store misses
```

PAPI counters

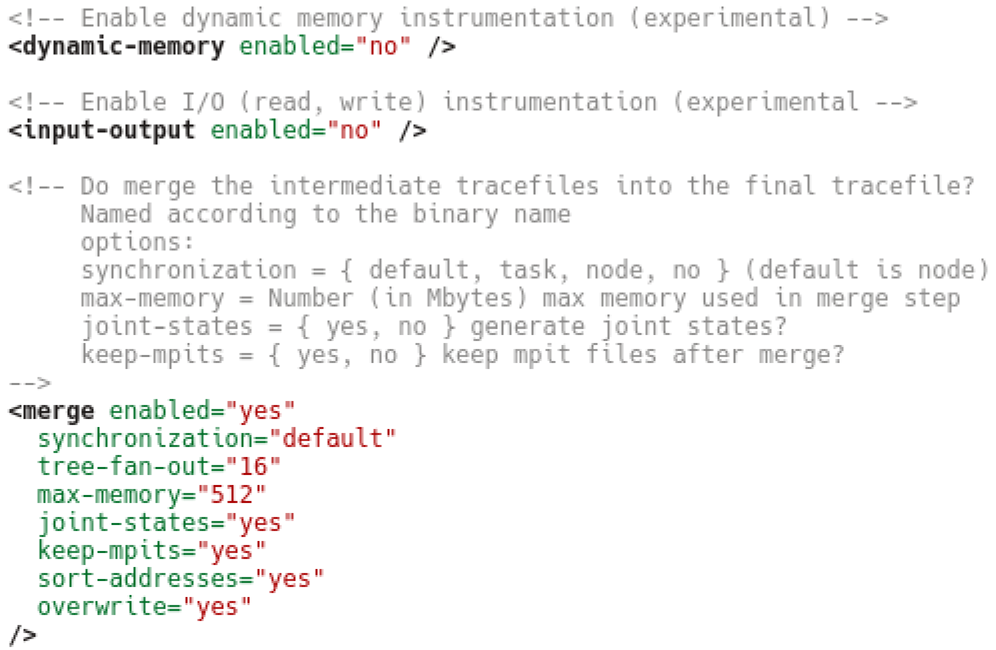
HW counters to capture

# Extrae configuration: extrae.xml



```
<!-- Bursts library enabled? This requires an special library! -->
<bursts enabled="no">
  <!-- Specify the threshold. This is mandatory! In this example, the
       threshold is limitted to 500 microseconds
   -->
  <threshold enabled="yes">500u</threshold>
  <!-- Report MPI statistics? -->
  <mpi-statistics enabled="yes" />
</bursts>

<!-- Enable sampling capabilities using system clock.
     Type may refer to: default, real, prof and virtual.
     Period stands for the sampling period (50ms here)
     plus a variability of 10ms, which means periods from
     45 to 55ms.
-->
<sampling enabled="no" type="default" period="50m" variability="10m" />
```

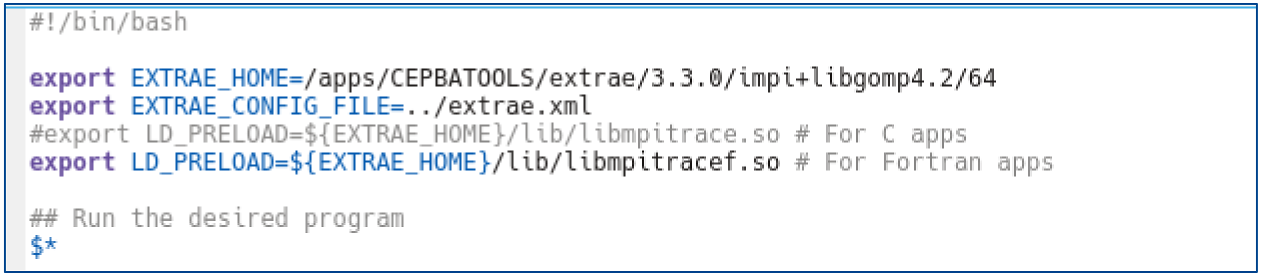
Enable sampling



```
<!-- Enable dynamic memory instrumentation (experimental) -->
<dynamic-memory enabled="no" />

<!-- Enable I/O (read, write) instrumentation (experimental -->
<input-output enabled="no" />

<!-- Do merge the intermediate tracefiles into the final tracefile?
     Named according to the binary name
     options:
     synchronization = { default, task, node, no } (default is node)
     max-memory = Number (in Mbytes) max memory used in merge step
     joint-states = { yes, no } generate joint states?
     keep-mpits = { yes, no } keep mpit files after merge?
-->
<merge enabled="yes"
  synchronization="default"
  tree-fan-out="16"
  max-memory="512"
  joint-states="yes"
  keep-mpits="yes"
  sort-addresses="yes"
  overwrite="yes"
/>
```

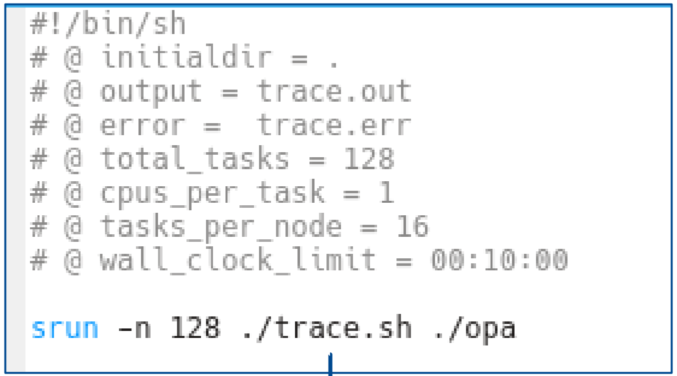
# Tracing

## Extrae wrapper: sets environment and loads required library



```
#!/bin/bash

export EXTRAE_HOME=/apps/CEPBATOOLS/extrae/3.3.0/impi+libgomp4.2/64
export EXTRAE_CONFIG_FILE=../extrae.xml
#export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitrace.so # For C apps
export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitracef.so # For Fortran apps

## Run the desired program
$*
```

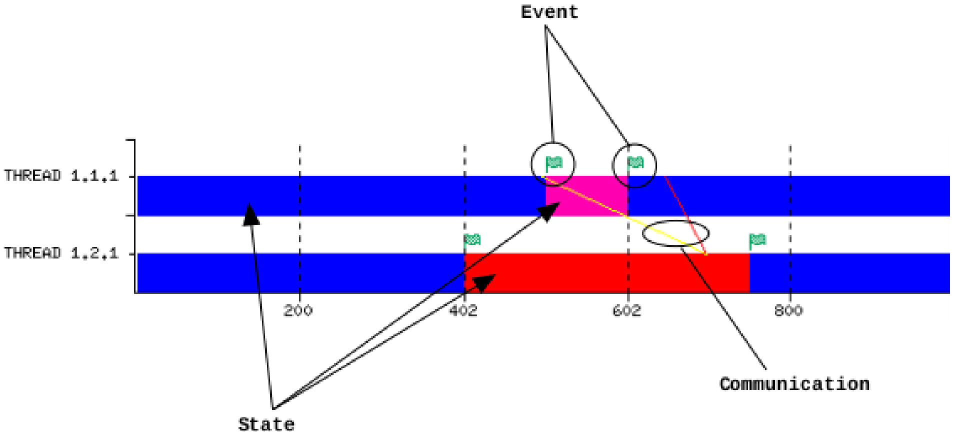
## Add the wrapper to your batch script



```
#!/bin/sh
# @ initialdir = .
# @ output = trace.out
# @ error =  trace.err
# @ total_tasks = 128
# @ cpus_per_task = 1
# @ tasks_per_node = 16
# @ wall_clock_limit = 00:10:00

srun -n 128 ./trace.sh ./opa
```

# Trace analysis

- **Paraver traces:** made up from records (timestamp + event or activity) of three different kind:
  - **State records:** intervals of thread status, i.e, waiting in a barrier (either MPI or OpenMP), waiting for a message, computing...
  - **Event records:** punctual event occurred in a given timestamp, as entry & exit points of user functions, MPI routines, OpenMP parallel regions...
  - **Communication records:** relationship between two objects, as communication between two processes (MPI), task movement among threads (OpenMP/OmpSs) or memory transfers (CUDA/OpenCL).



# Trace analysis

- Paraver traces are composed by **three files** (one ASCII trace file + two metadata optional files):
  - ASCII trace file (**.prv**): defines the objects structure and contains **a list of all the trace records.**
  - Paraver configuration file (**.pcf**): defines **labels and colors associated to states and events**.
  - Names configuration file (**.row**): defines the **row labels** that will be displayed in the application.

```
#Paraver (22/05/01 at
16:20):1021312:2(16,16):1:2(1:1,1:2)
1:1:1:1:1:0:100:4
1:2:1:2:1:0:200:4
1:1:1:1:1:100:300:1
1:1:1:1:1:200:500:4
3:1:1:1:1:300:325:2:1:2:1:200:330:10:3000
2:1:1:1:1:300:60000000:1
.
.
.
.
.
```

Trace file (.prv)

# Trace analysis

| | Average values | CLAIX |
|---|---|---|
| Event | 150-200 ns | 140 ns |
| Event + PAPI | 750 ns – 1 us | 600 ns |
| Event + callstack (1 level) | 600 ns | 690 ns |
| Event + callstack (6 levels) | 1.9 us | 2.6 us |