Selected performance assessments using Extrae and Paraver

Marta Garcia-Gasulla, BSC

HORIZON-EUROHPC-JU-2023-COE

EuroHPC Joint Undertaking

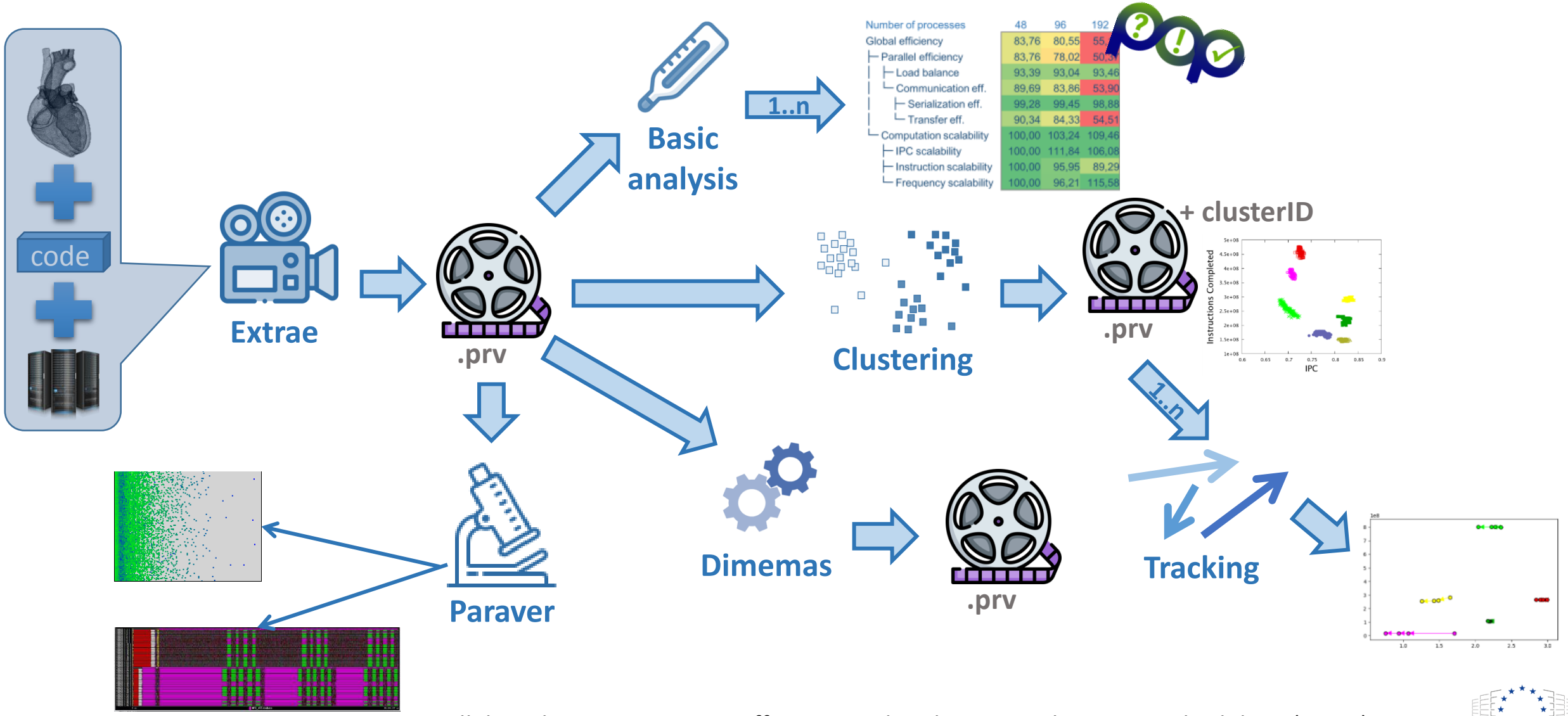
1 January 2024– 31 December 2026

Grant Agreement No 101143931

Content

- The ecosystem of BSC performance tools
  - Extrae
  - Paraver
  - Dimemas
- Show case of a performance analysis using the BSC performance tools

The ecosystem of BSC performance tools



EuroHPC Joint Undertaking

Extrae

Extrae

No need to recompile or relink

- Tracing library transparent to the application
- Platform agnostic
- Parallel programming models:
  - MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, Java, Python
- Hardware counters
  - Through PAPI
- Link to source:
  - Callstack at MPI routines
  - OpenMP outlined routines
  - Selected user functions (Dyninst)
- Periodic sampling
- User friendly API to annotate your code with custom events

How to use Extrae

- Symbol substitution through LD_PRELOAD

```
export LD_PRELOAD=$EXTRAE_HOME/lib/libmpitrace.so
```

- Specific libraries for each runtime and combinations
  - MPI
  - OpenMP
  - OpenMP+MPI
  - …

```
libmpitrace.so      libomptrace.so
libmpitracecf.so    libptmpitrace.so
libmpitracef.so     libptmpitracecf.so
libompitrace.so     libptmpitracef.so
libompitracecf.so   libpttrace.so
libompitracef.so    libseqtrace.so
```

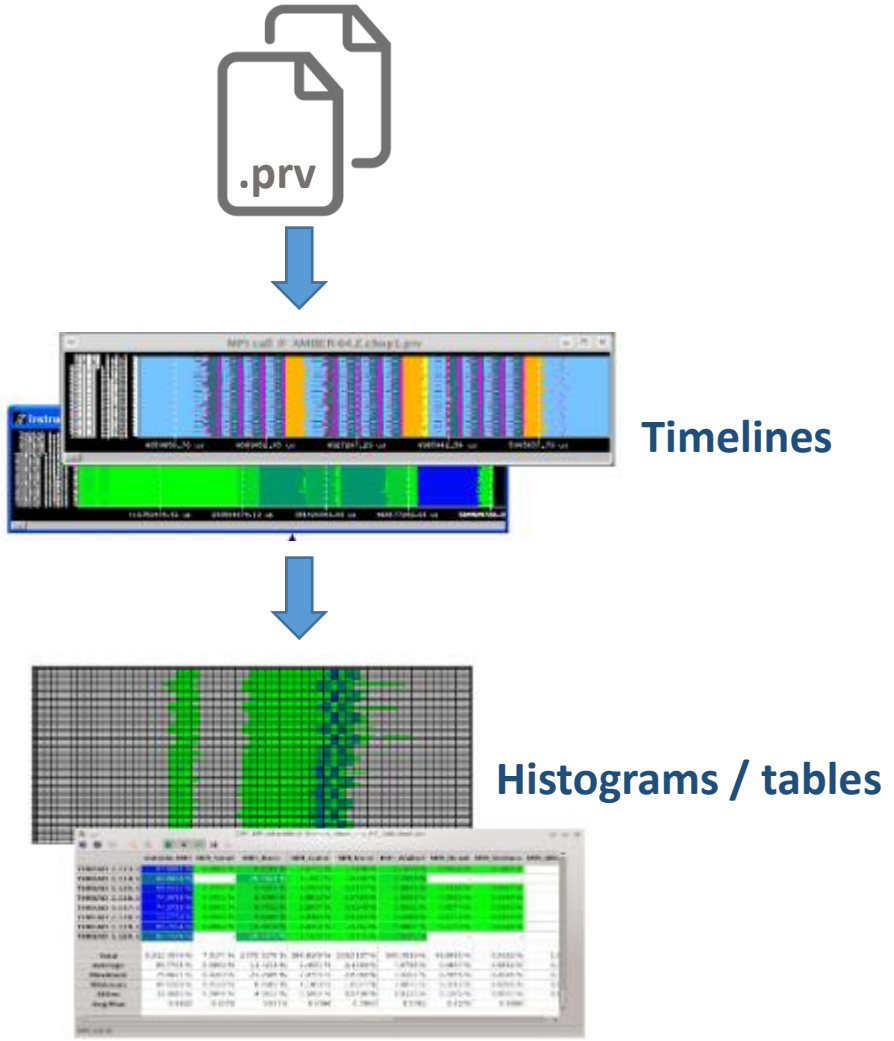
- Detailed configuration in XML file

```
export EXTRAE_CONFIG_FILE=../extrae.xml
```

Paraver

Paraver

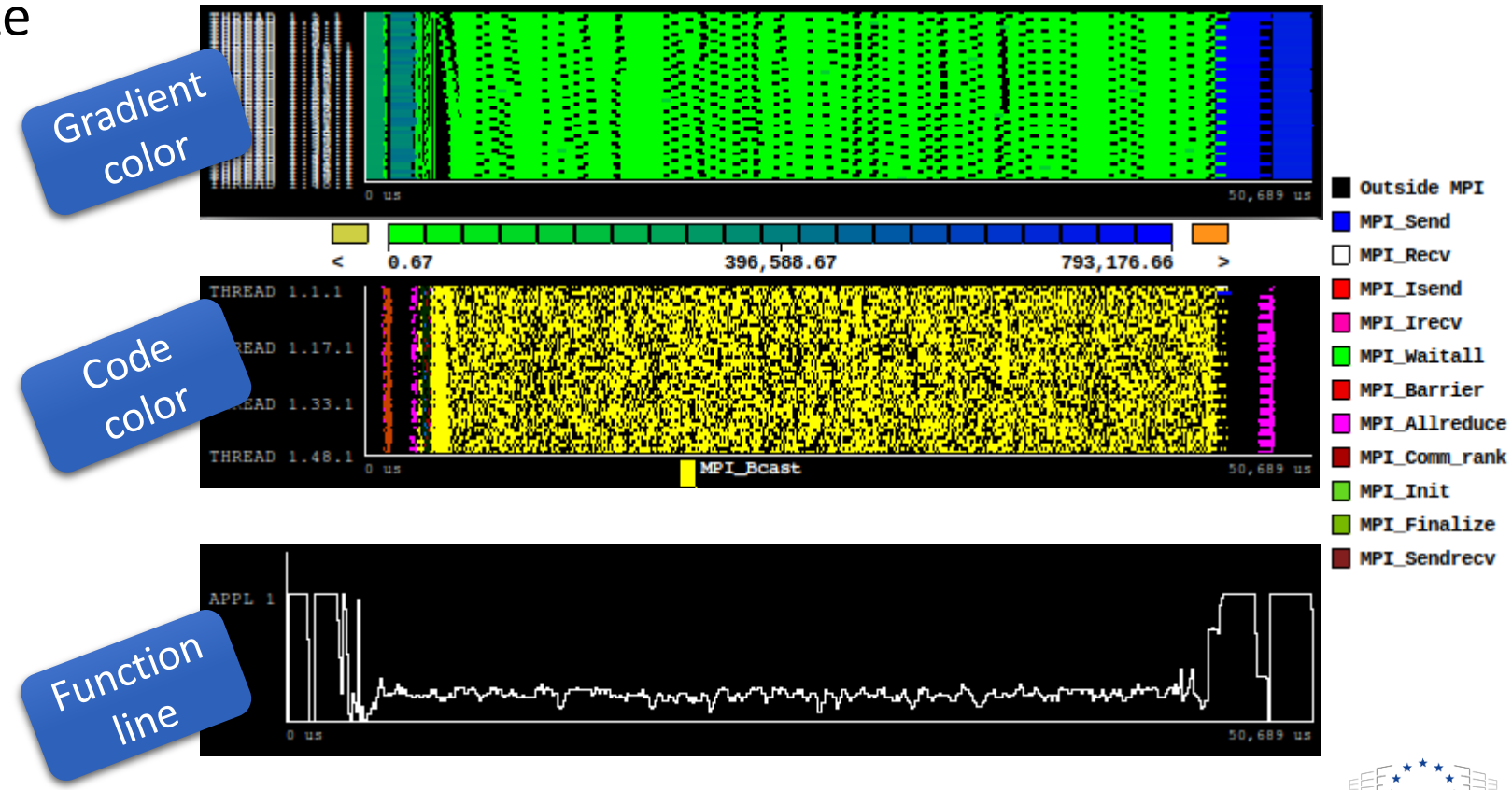
- (Performance) Data Browser
  - Any kind of timestamped data
  - Trace Visualization and analysis
  - Trace manipulation
  - Flexible
    - No pre-assumed semantics
    - Fully programmable
  - 2 Kind of views:
    - Timeline
    - Histograms, 2D and 3D tables
  - Multiple loaded traces
    - Allow comparative analysis



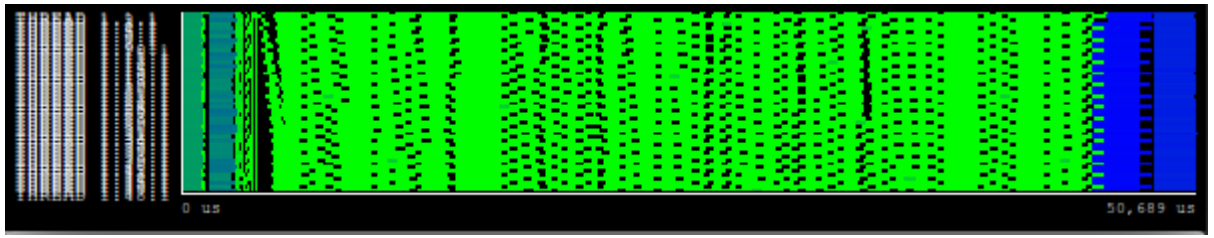
Timelines

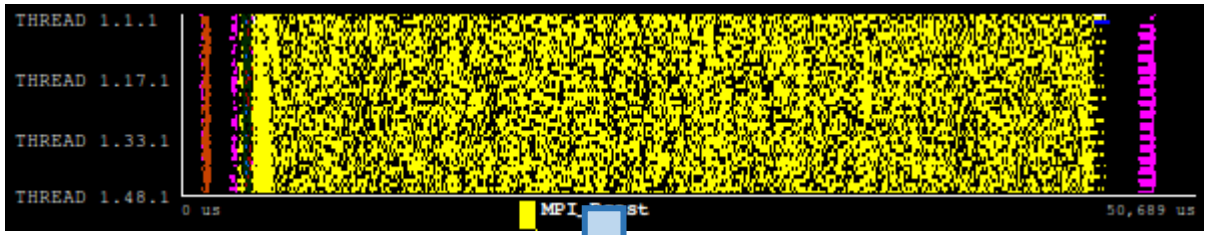
Separate the *What* from the *How*

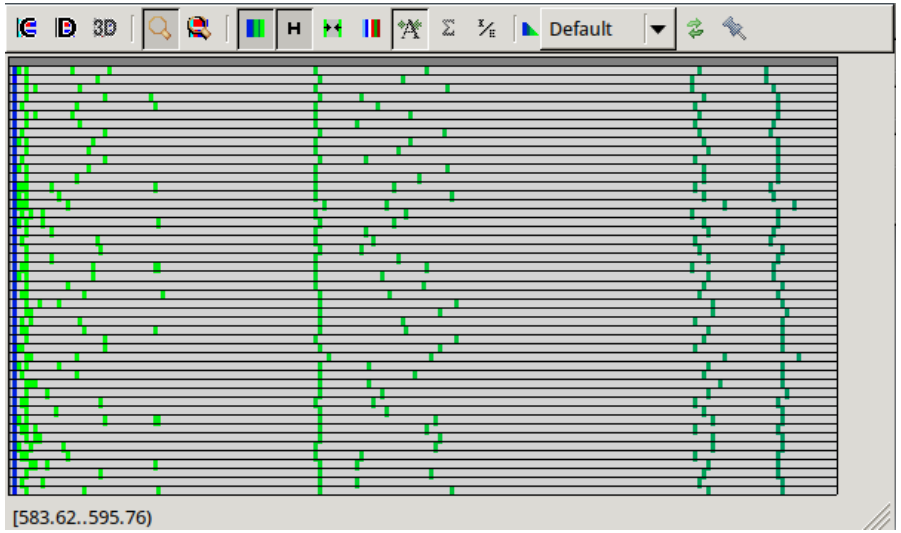
- *What* -> Semantics
  - Which event to visualize
  - Which value to use
- How -> Visual representation



Histograms



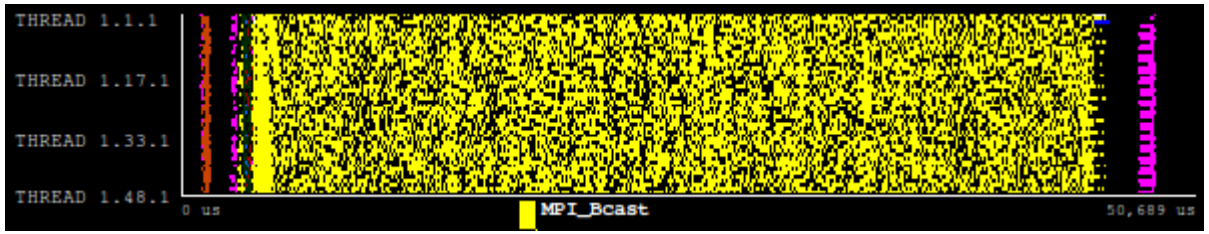




	Outside MPI	MPI_Send	MPI_Recv	MPI_Bcast	MPI_Allreduce	MPI_Comm_rank	MPI_Comm_size
THREAD 1.1.1	23.74 %	0.26 %	1.40 %	72.80 %	0.48 %	0.03 %	0.03 %
THREAD 1.2.1	24.55 %	0.19 %	0.23 %	72.50 %	0.77 %	0.03 %	0.03 %
THREAD 1.3.1	22.63 %	1.47 %	0.34 %	73.42 %	0.38 %	0.03 %	0.03 %
THREAD 1.4.1	23.79 %	-	-	72.27 %	2.29 %	0.03 %	0.03 %
THREAD 1.5.1	23.47 %	-	-	73.83 %	1.06 %	0.03 %	0.02 %
THREAD 1.6.1	23.34 %	-	-	72.99 %	2.03 %	0.03 %	0.03 %
THREAD 1.7.1	22.74 %	-	-	73.19 %	2.36 %	0.04 %	0.03 %
THREAD 1.8.1	23.58 %	-	-	74.02 %	0.69 %	0.03 %	0.03 %
THREAD 1.9.1	22.66 %	-	-	73.59 %	2.05 %	0.04 %	0.03 %
THREAD 1.10.1	22.93 %	-	-	73.34 %	2.10 %	0.04 %	0.02 %
THREAD 1.11.1	23.02 %	-	-	74.22 %	1.11 %	0.03 %	0.03 %
THREAD 1.12.1	23.94 %	-	-	72.64 %	1.78 %	0.03 %	0.03 %
THREAD 1.13.1	23.04 %	-	-	73.28 %	1.99 %	0.04 %	0.03 %
THREAD 1.14.1	24.67 %	-	-	72.64 %	0.99 %	0.03 %	0.03 %
THREAD 1.15.1	22.88 %	-	-	73.61 %	1.80 %	0.03 %	0.03 %
THREAD 1.16.1	22.96 %	-	-	73.46 %	1.98 %	0.04 %	0.03 %
THREAD 1.17.1	22.96 %	-	-	74.58 %	0.88 %	0.03 %	0.03 %
THREAD 1.18.1	23.72 %	-	-	72.61 %	2.09 %	0.03 %	0.03 %
THREAD 1.19.1	23.33 %	-	-	72.71 %	2.28 %	0.04 %	0.03 %
THREAD 1.20.1	22.98 %	-	-	74.26 %	1.09 %	0.03 %	0.03 %
THREAD 1.21.1	23.03 %	-	-	73.04 %	2.27 %	0.03 %	0.03 %
THREAD 1.22.1	22.96 %	-	-	73.35 %	2.06 %	0.04 %	0.02 %

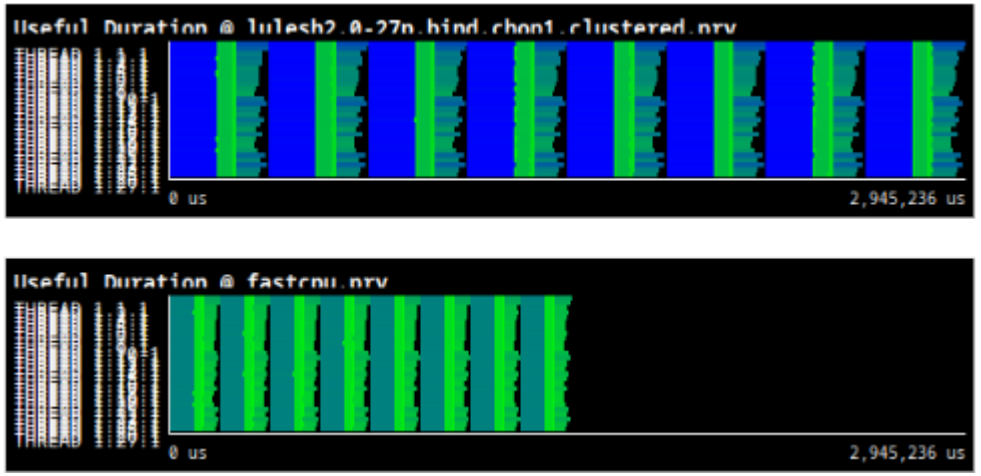
Trace Manipulation

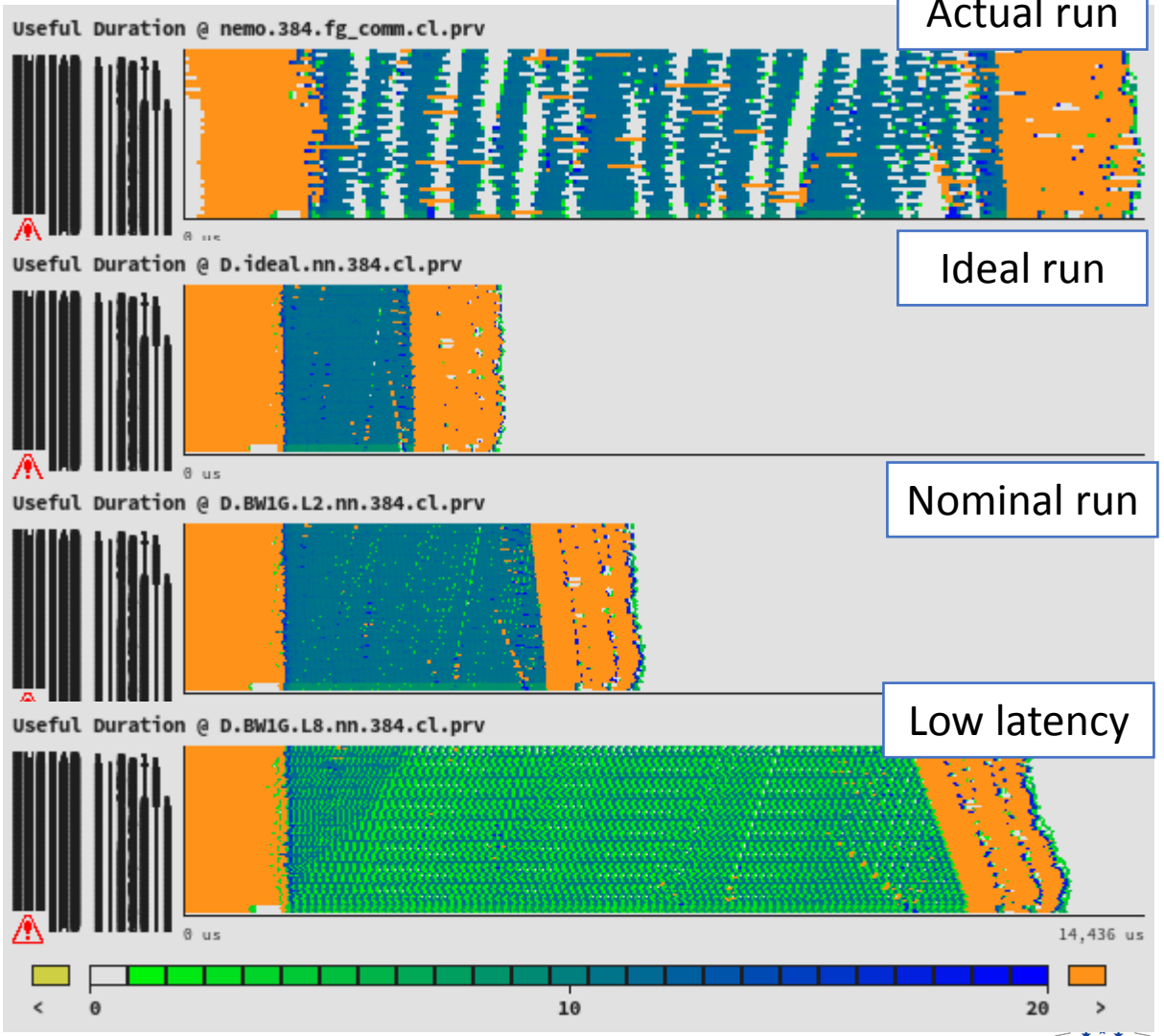
- Traces can be manipulated
  - Chop: cut in time or processes
  - Filter: Subset of records based on…
    - … duration, type, value…
  - The output is still a Paraver trace



Dimemas

- Change parameters from the machine:
  - Network: Latency, bandwidth
  - CPU frequency





Actual run

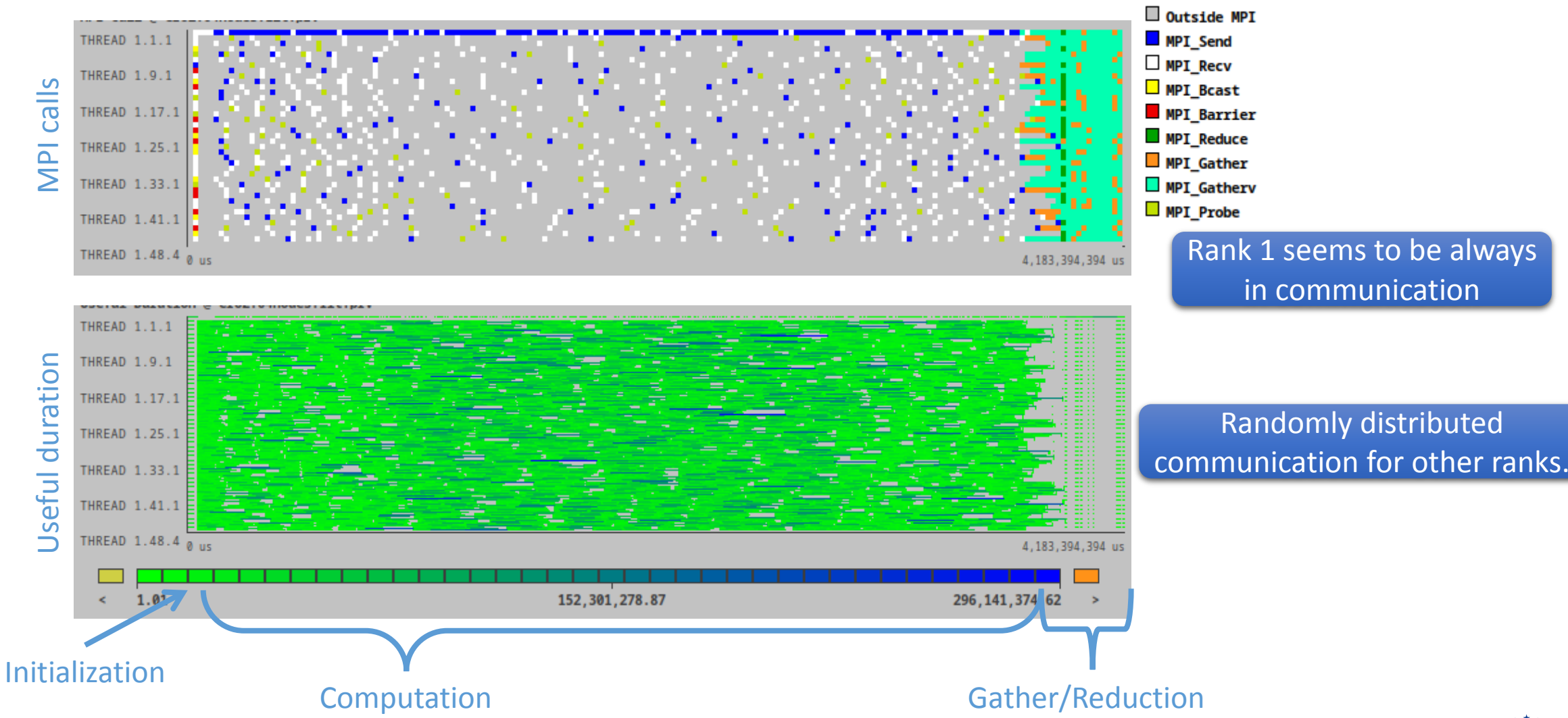
Ideal run

Nominal run

Low latency

Performance analysis of a hybrid code with BSC tools

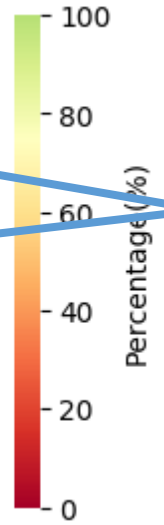
Structure and Focus of Analysis (FoA)



Efficiency metrics

Scaling number of processes (48 → 384)
Keeping constant number of threads (4)

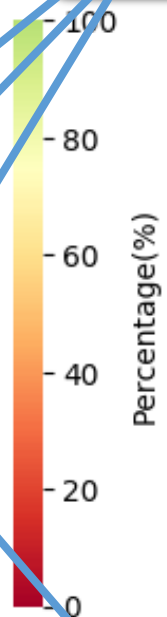
	192(48x4)[1]	384(96x4)[2]	768(192x4)[3]	1536(384x4)[4]
Global efficiency	80.20	70.95	57.81	39.51
-- Parallel efficiency	80.20	70.59	58.33	40.43
-- Load balance	91.52	86.73	76.55	62.27
-- Communication efficiency	87.64	81.39	76.21	64.93
-- Computation scalability	100.00	100.52	99.10	97.73
-- IPC scalability	100.00	99.94	100.23	100.35
-- Instruction scalability	100.00	100.58	98.91	97.67
-- Frequency scalability	100.00	100.00	99.96	99.71



Main factors affecting scalability:
- Load Balance
- Communication efficiency

Who to blame MPI or OpenMP?

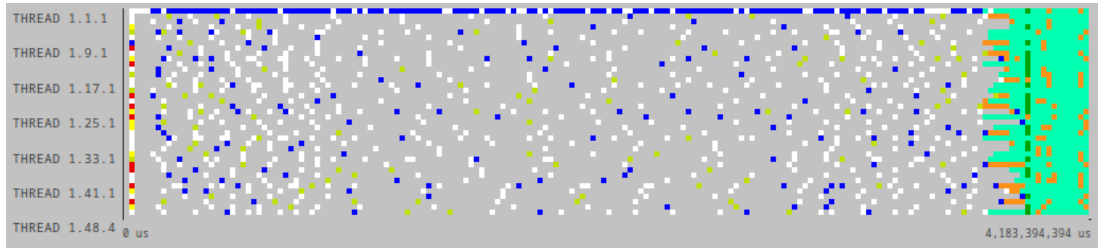
	192(48x4)[1]	384(96x4)[2]	768(192x4)[3]	1536(384x4)[4]
-- Hybrid Parallel efficiency	80.20	70.59	58.33	40.43
-- MPI Parallel efficiency	88.95	79.11	66.42	47.71
-- MPI Load balance	95.46	90.80	84.69	68.30
-- MPI Communication efficiency	93.18	87.13	78.42	69.86
-- Serialization efficiency	93.20	87.18	78.53	70.03
-- Transfer efficiency	99.98	99.94	99.86	99.75
-- OpenMP Parallel efficiency	90.17	89.23	87.83	84.74
-- OpenMP Load Balance	95.87	95.53	90.39	91.17
-- OpenMP Communication efficiency	94.05	93.41	97.17	92.94



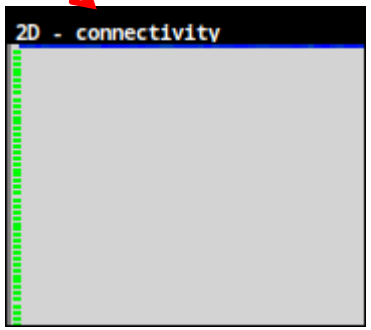
Very low MPI parallel efficiency due to Load Balance and Serialization

OpenMP better parallel efficiency, still a trend to decrease

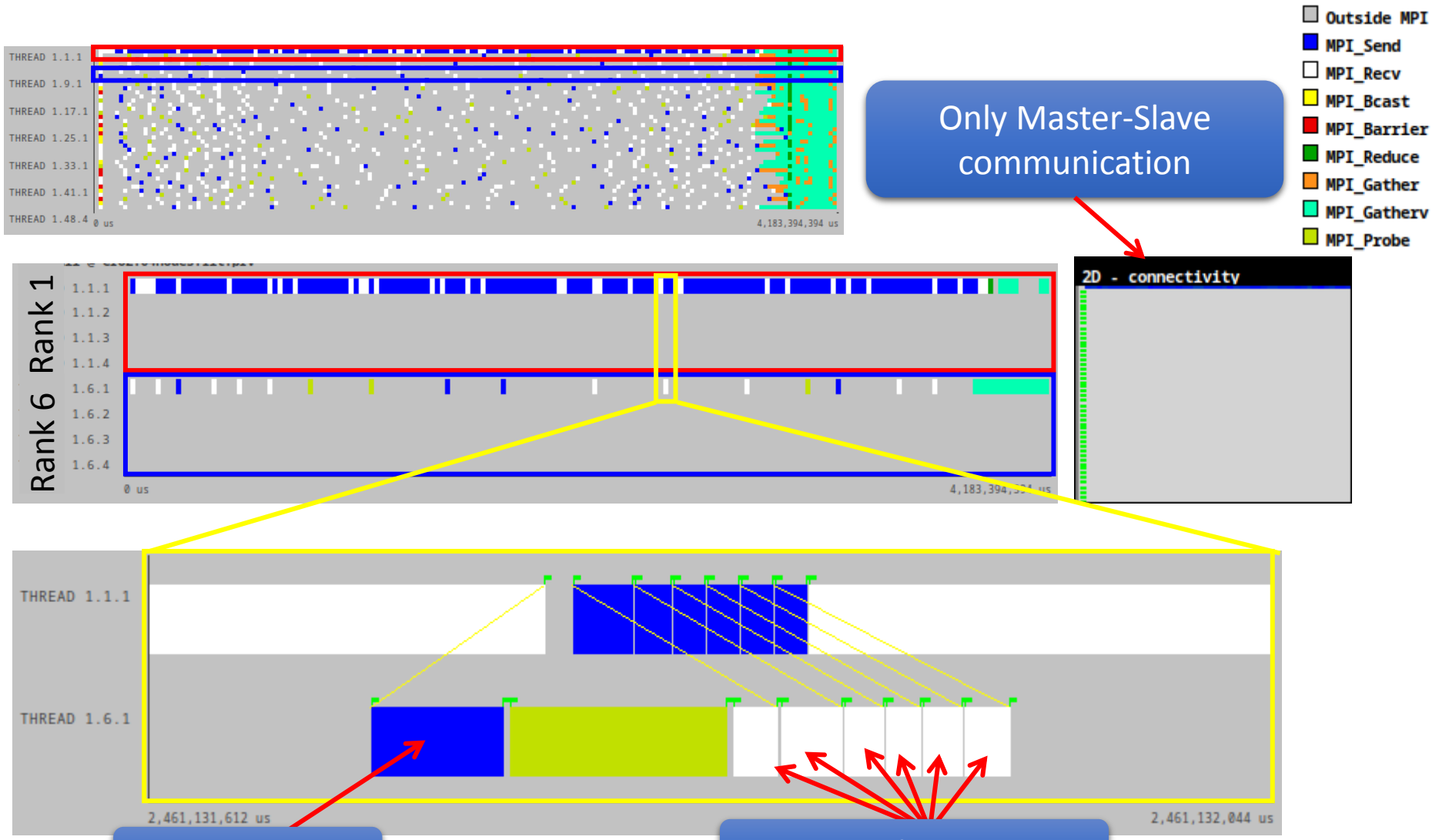
Communication pattern



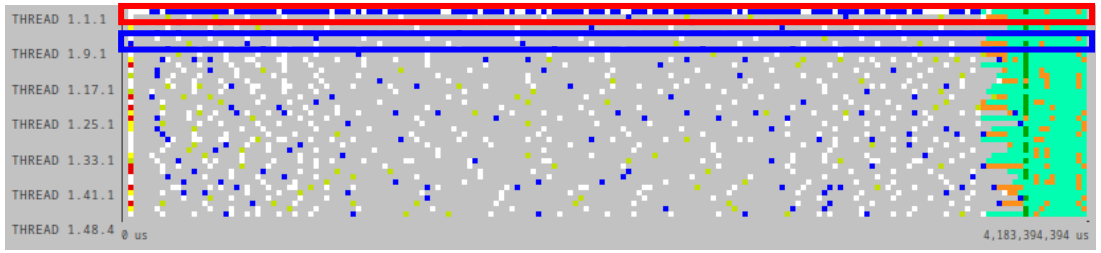
Only Master-Slave communication

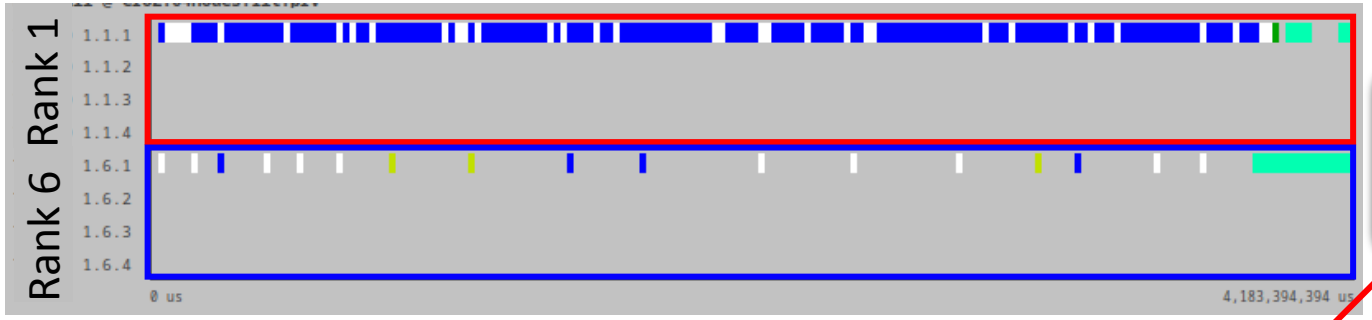


Communication pattern

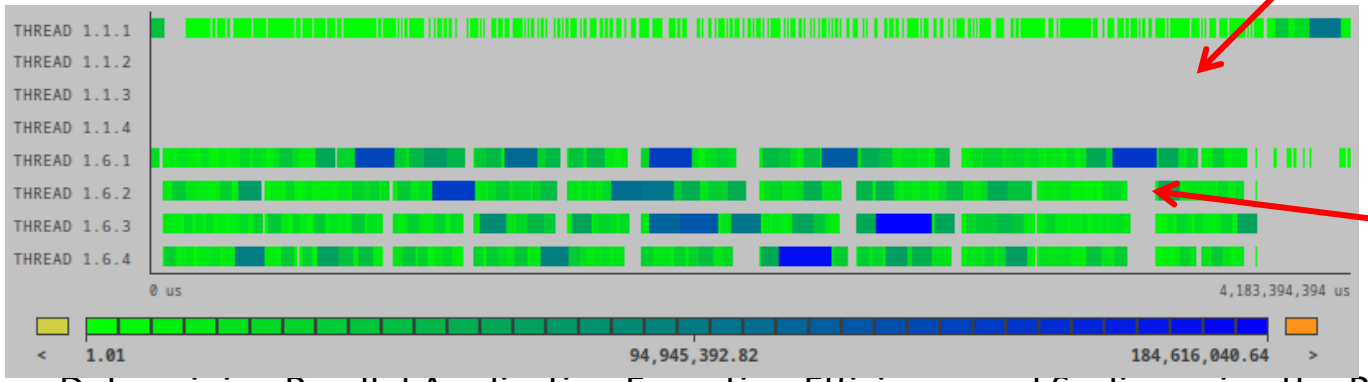


Communication pattern





The threads of the master process don't perform useful computation

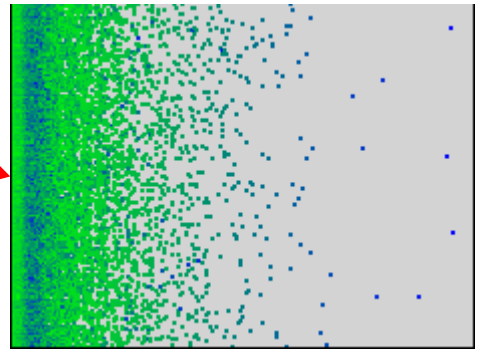


Worker rank do useful computation at all threads

Gaps happen when one thread has longer useful computation.

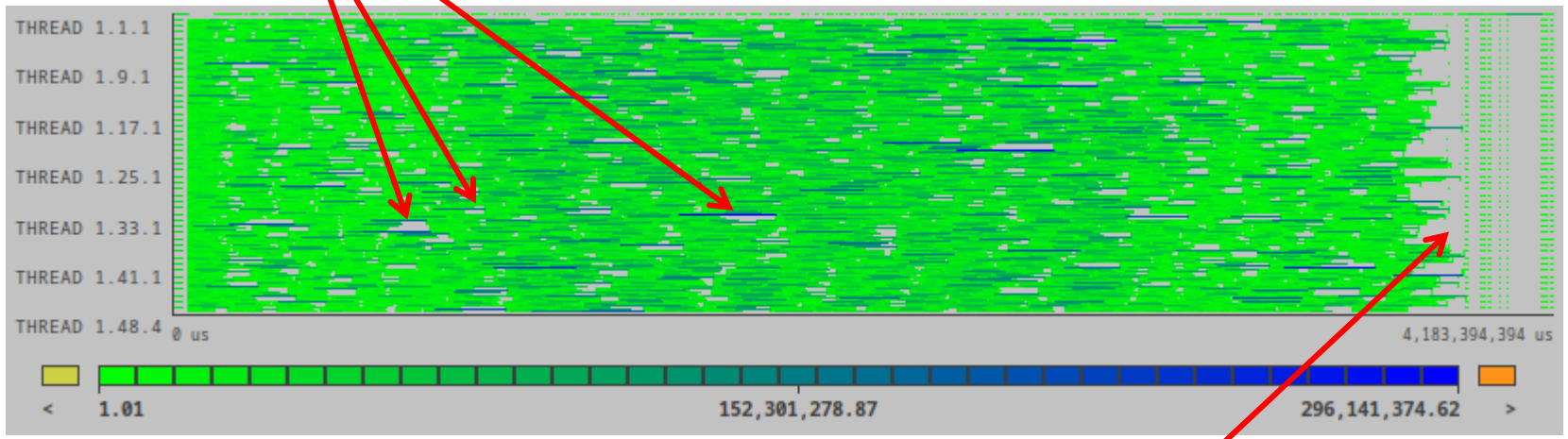
Load Imbalance

High variability in useful duration



Long Bursts distributed among all processes

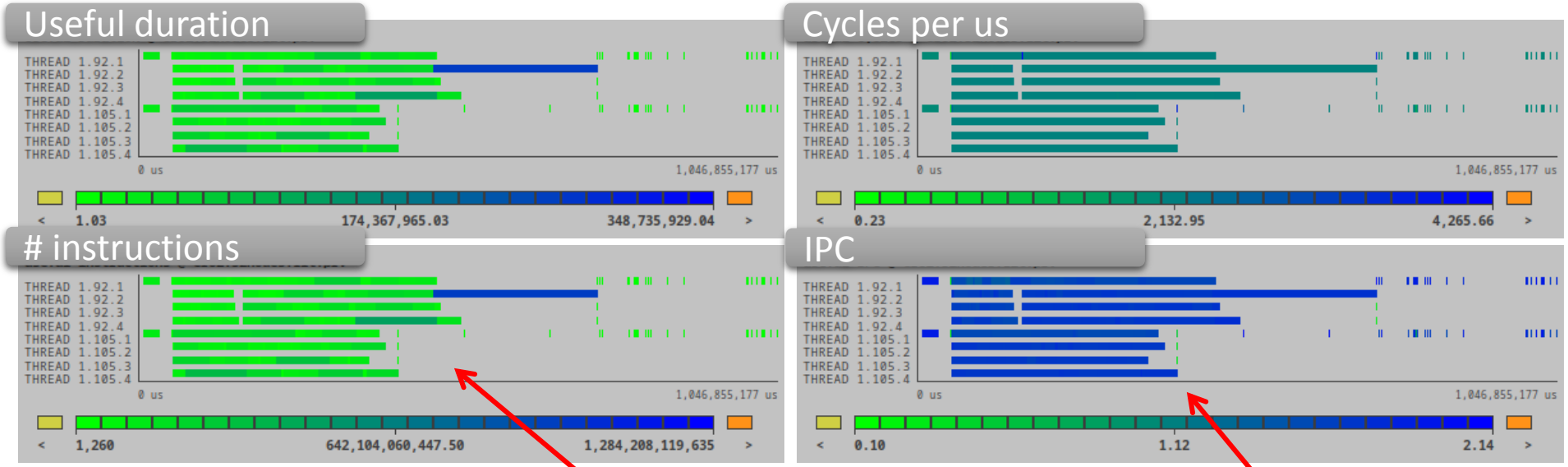
Several, not useful gaps randomly distributed



Long Bursts appear more in the “center”

Load Imbalance at the end

The source of the Load imbalance

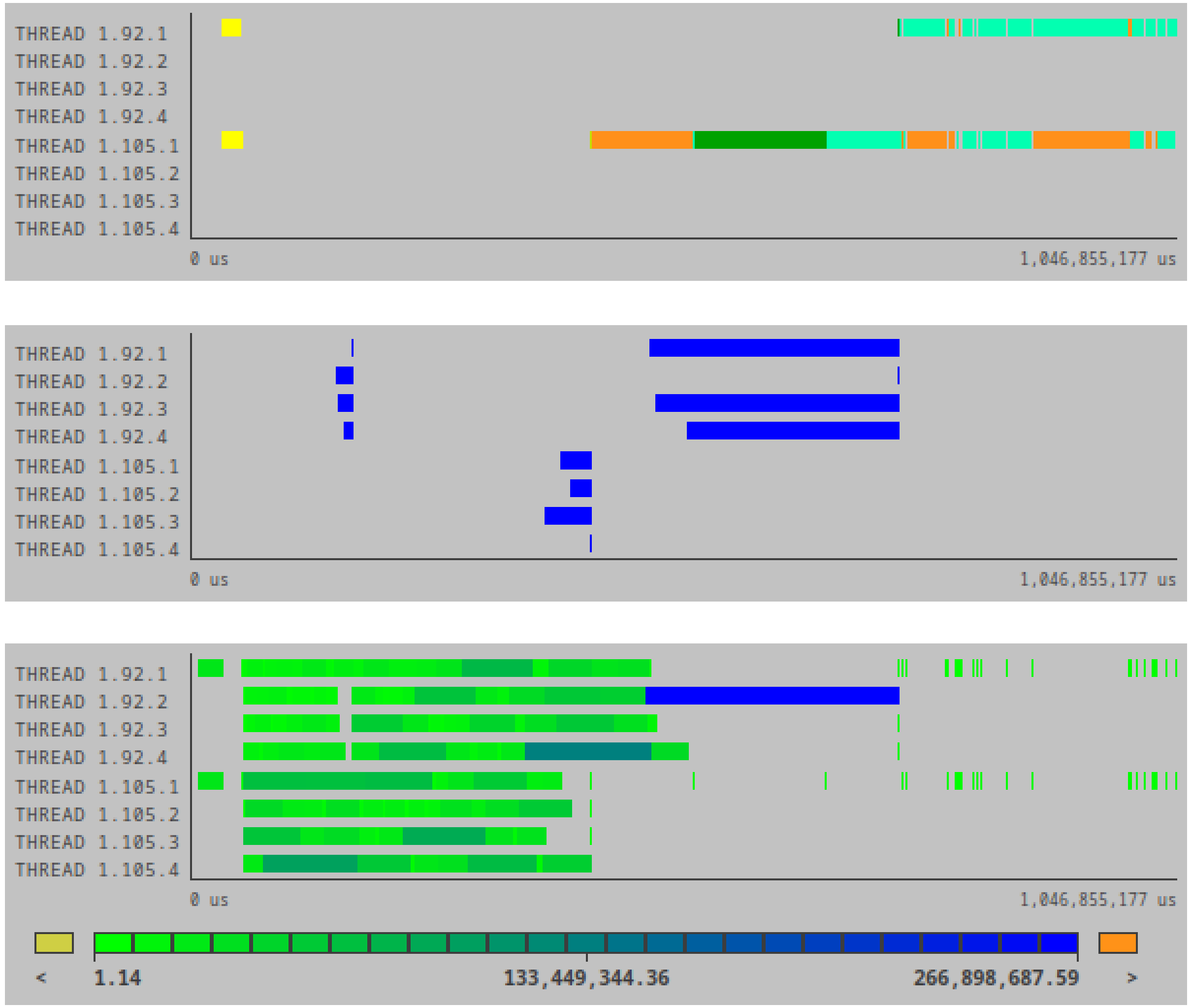


The amount of instructions correlates with the burst duration.

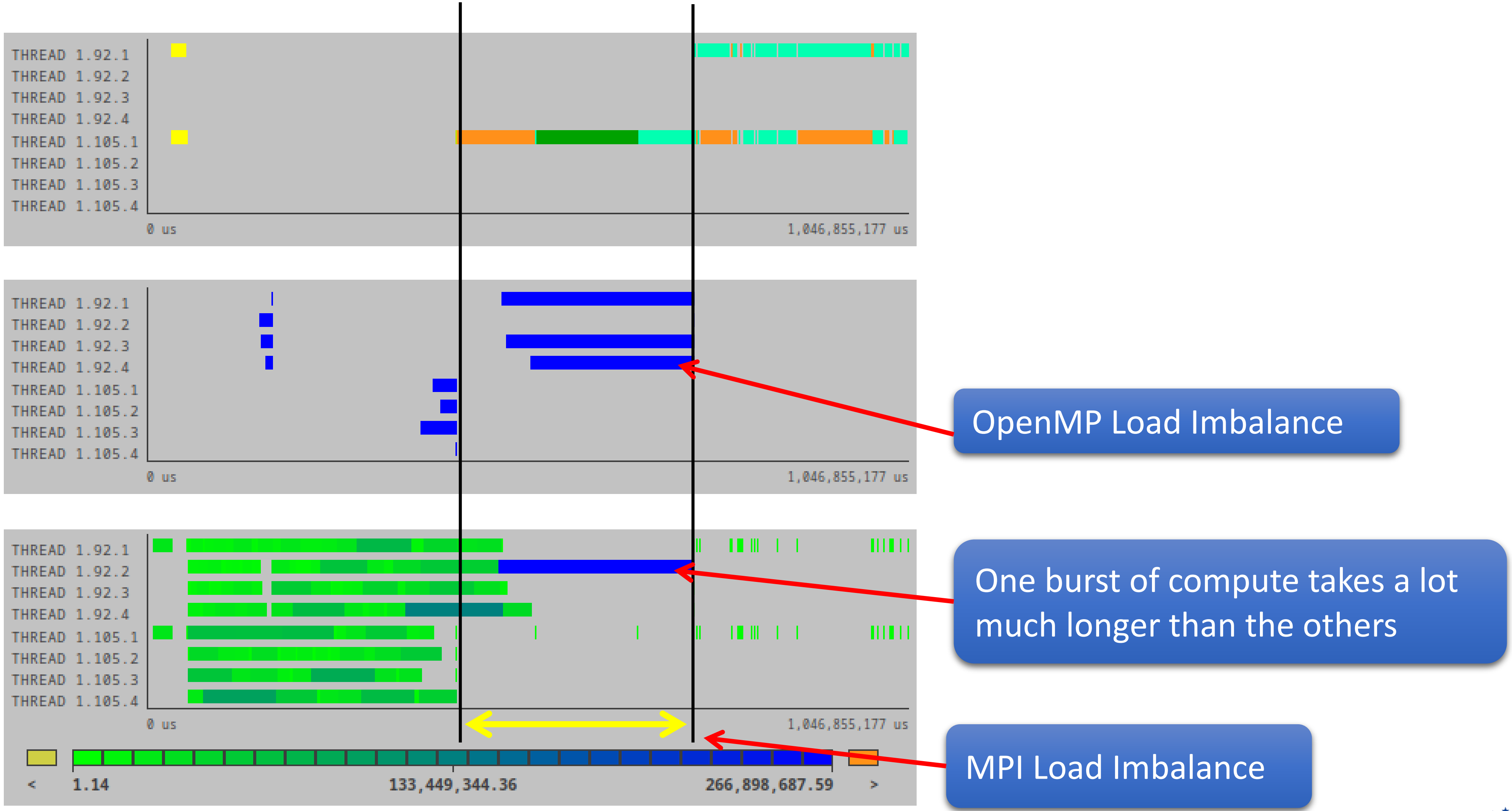
Load balance seems to come due to more instructions. Hence more code execution.

On the other hand, the cycles and IPC are stable for most bursts.

Load imbalance in detail

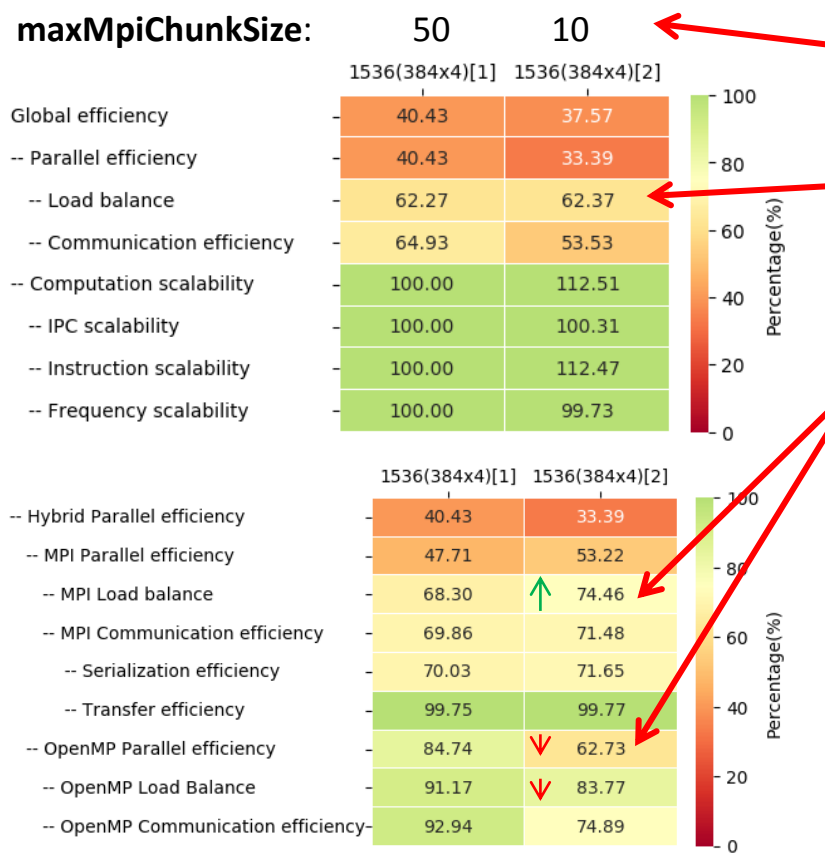


Load imbalance in detail



Load imbalance

Application parameter to set size of chunk sent. Reduce it to improve MPI Load imbalance



MPI grain reduction

Overall Worst Load Balance

We tradeoff the OpenMP Load Balance for the MPI

Conclusions

- Main issue caused by unpredictable long chunks of computation
  - Affects both MPI and OpenMP
- Suggestions:
  - OpenMP: Avoid synchronization between threads when asking for more work to the master
  - MPI and OpenMP: Implement a “guided like” distribution of work by the master process. Bigger chunks at the beginning of the execution smaller at the end
  - MPI: “Kill” long computations at the end of the execution if there is no more work to do.
- After the implementation of the optimizations the execution in 16 nodes was 3x times faster

Performance Optimisation and Productivity 3

A Centre of Excellence in HPC

Contact:

- https://www.pop-coe.eu
- pop@bsc.es
- @POP_HPC
- youtube.com/POPHPC

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101143931. The JU receives support from the European Union's Horizon Europe research and innovation programme and Spain, Germany, France, Portugal and the Czech Republic.