Status of NEMO Scalability and Inputs from POP Assessment

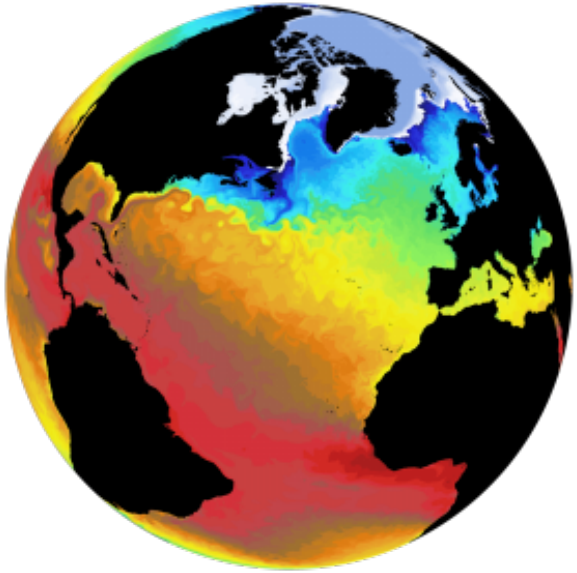
Sebastien Masson
Gaston Irrmann
Jesus Labarta, Marta Garcia, Joan Vinyals Ylla-Catala
Erwan Raffin, David Guibert

PASC21, 5-9 July, Geneva

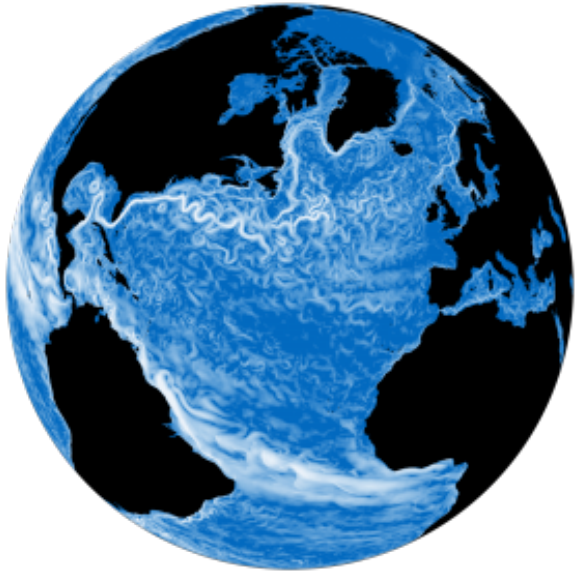
Nucleus for European Modelling of the Ocean

https://www.nemo-ocean.eu/

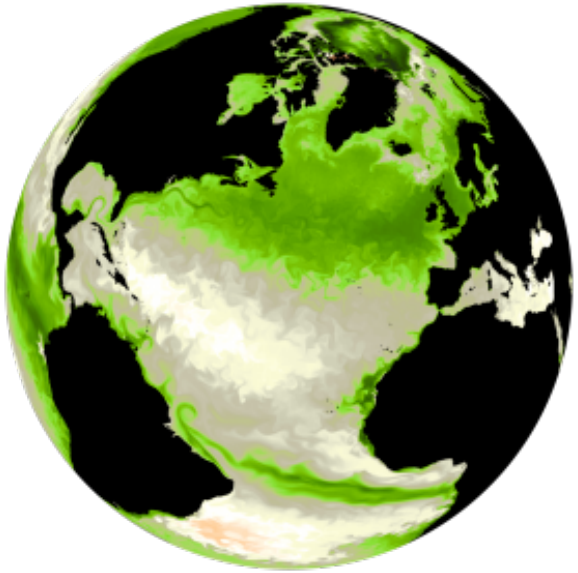
Sea-ice | **ocean dynamics** | Biogeochemistry



temperature
+ sea-ice



Velocity

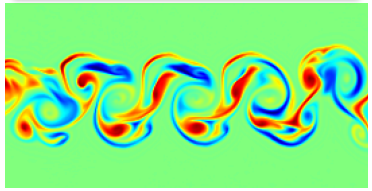


Chlorophyll

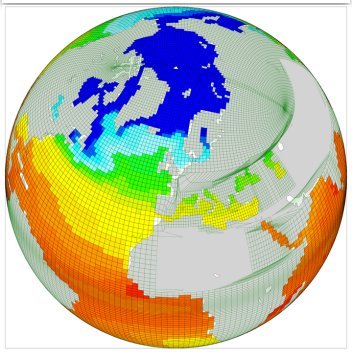
Nucleus for European Modelling of the Ocean

multiple applications

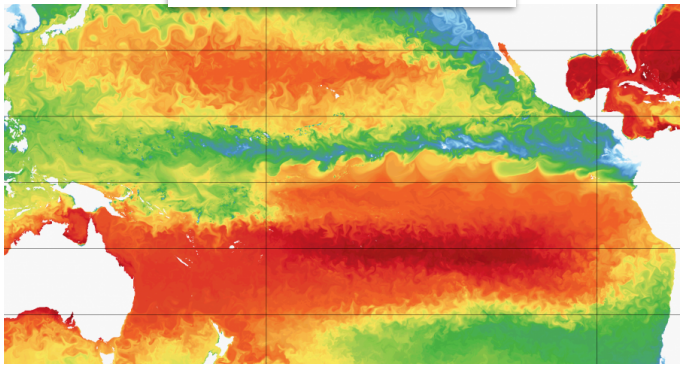
teaching



climate change



operational oceanography



laptop

small configuration
long integration

big configuration
short integration

=> multiple constrains

Nucleus for European Modelling of the Ocean

35 years of development by physicists

=> readability

=> sustainability

European consortium : 5 partners

development coordination (yearly work plan)
quality control

=> slow/heavy integration process

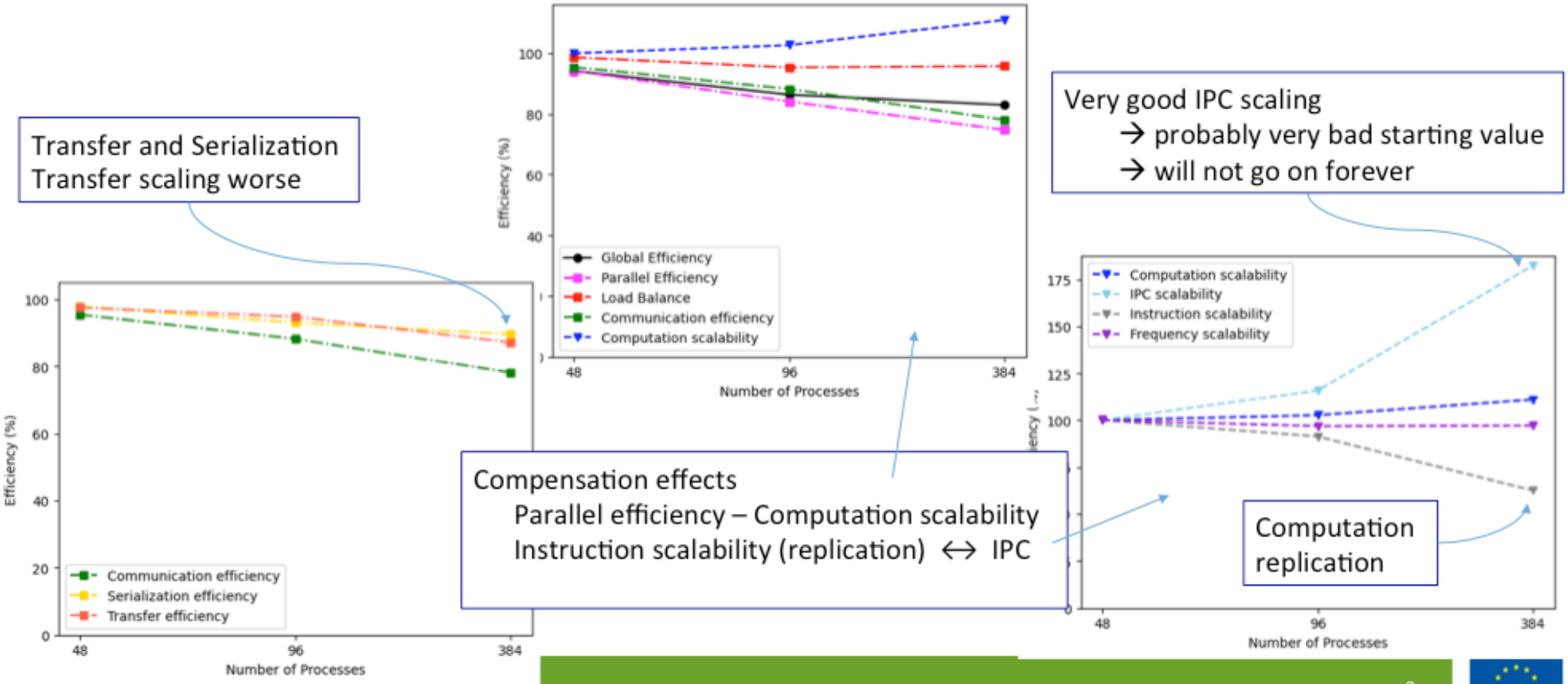
Detailed knowledge of the code

… but empirical knowledge of HPC

=> need of profiling, experts and quantitative information

=> provide a benchmark corresponding to a realistic configuration but without I/O

Efficiency model

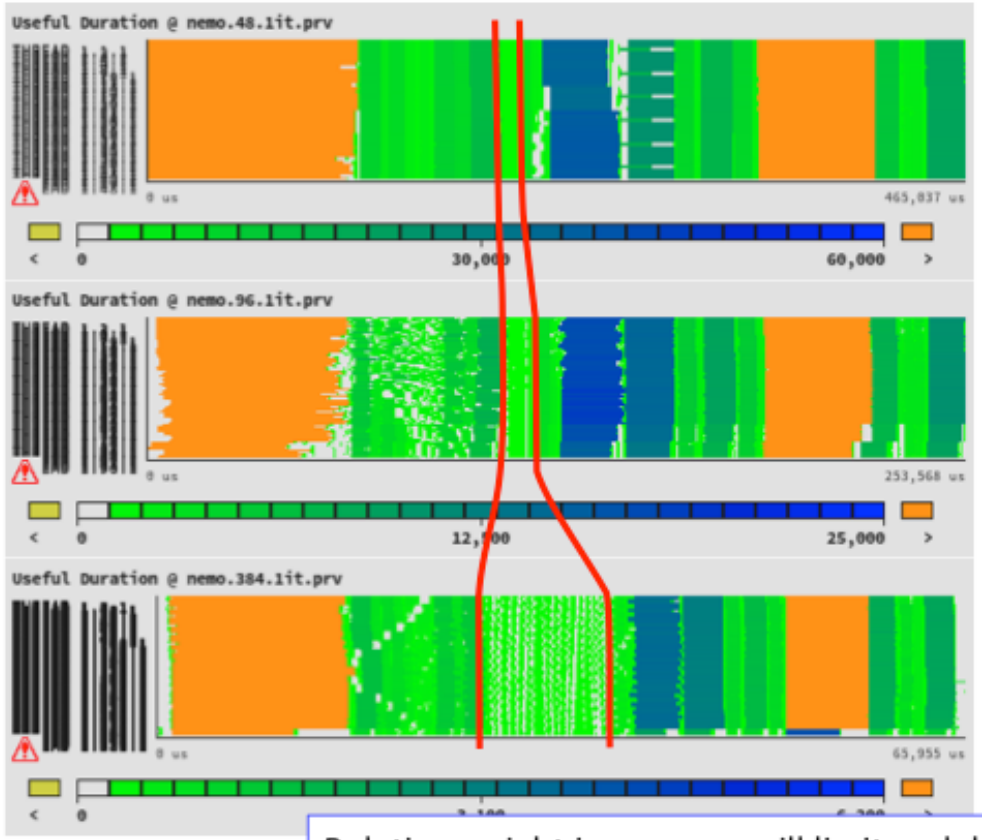


NEMO is memory bandwidth:
=> efficiency is good because of “bad starting value”

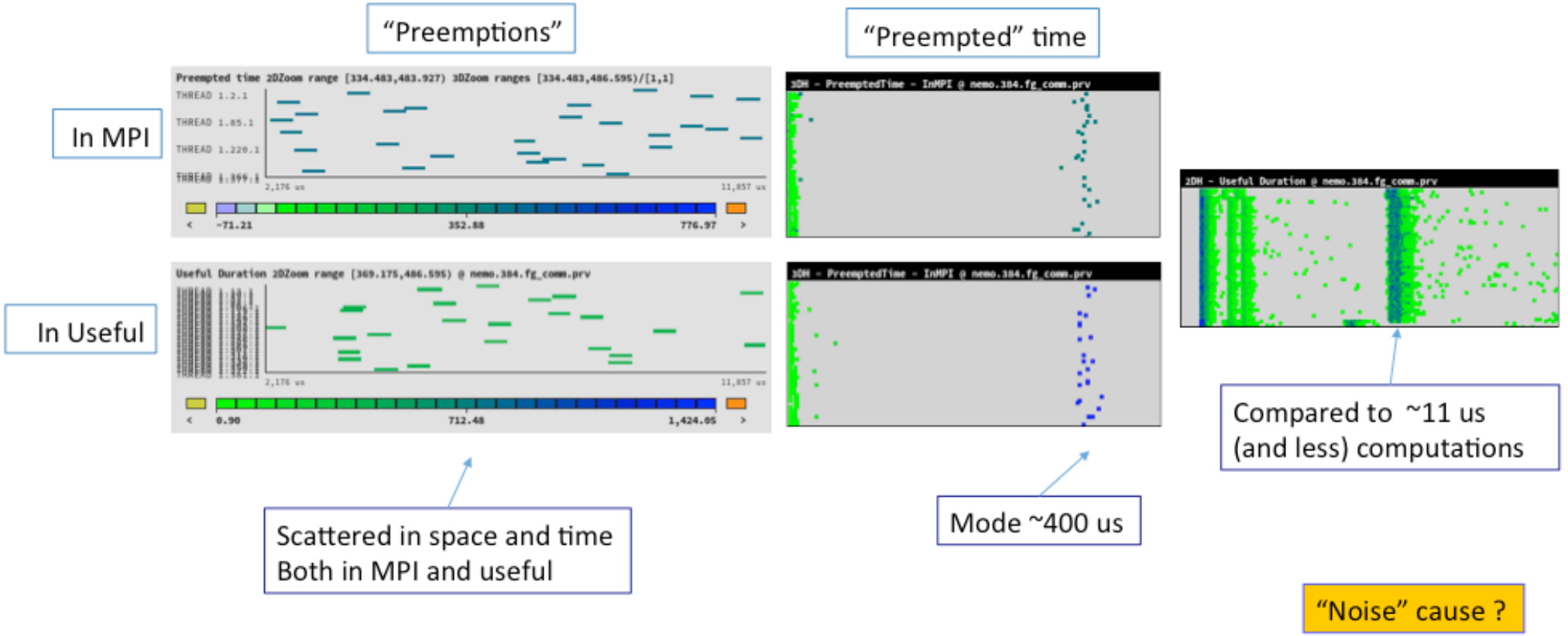
Many aspects pointed out by extrae/paraver analyses…

=> focus on the MPI part

Structure

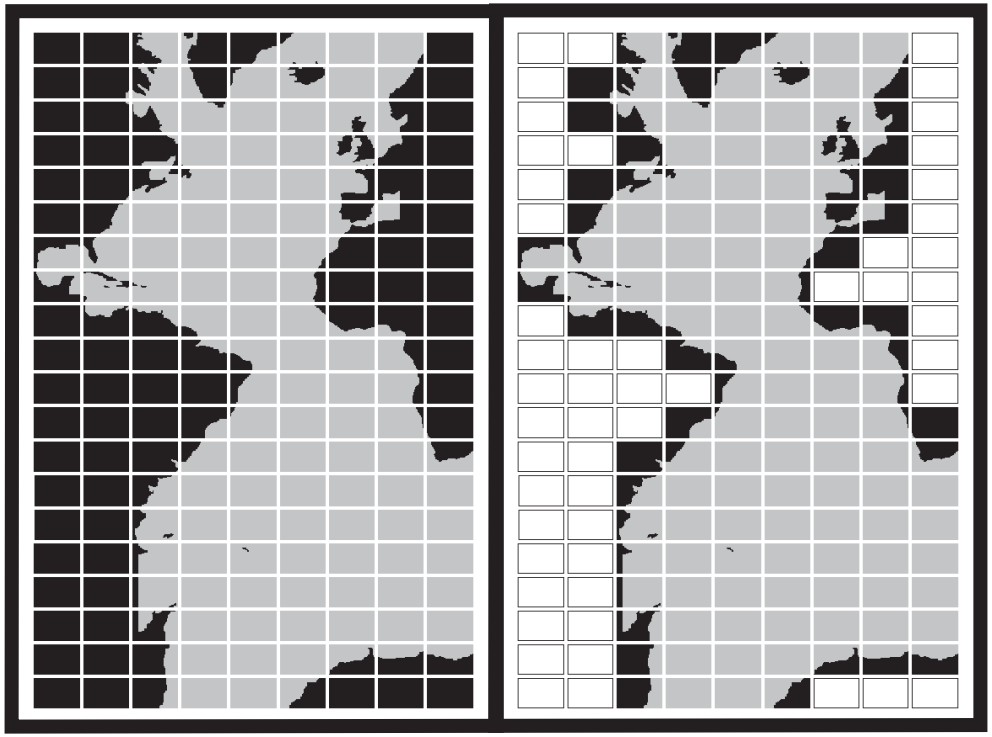


Close look at noise

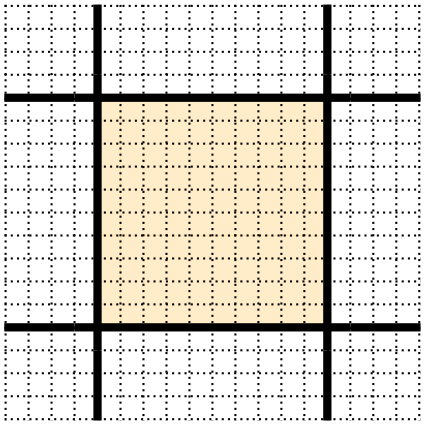


"Noise" is a "common feature":
=> Cant fight noise, learn to live with it…

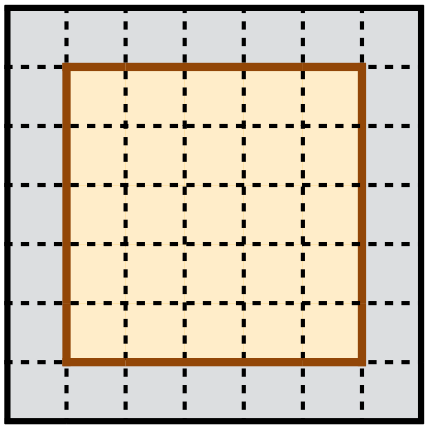
MPI domain decomposition



(1) Cut the domain

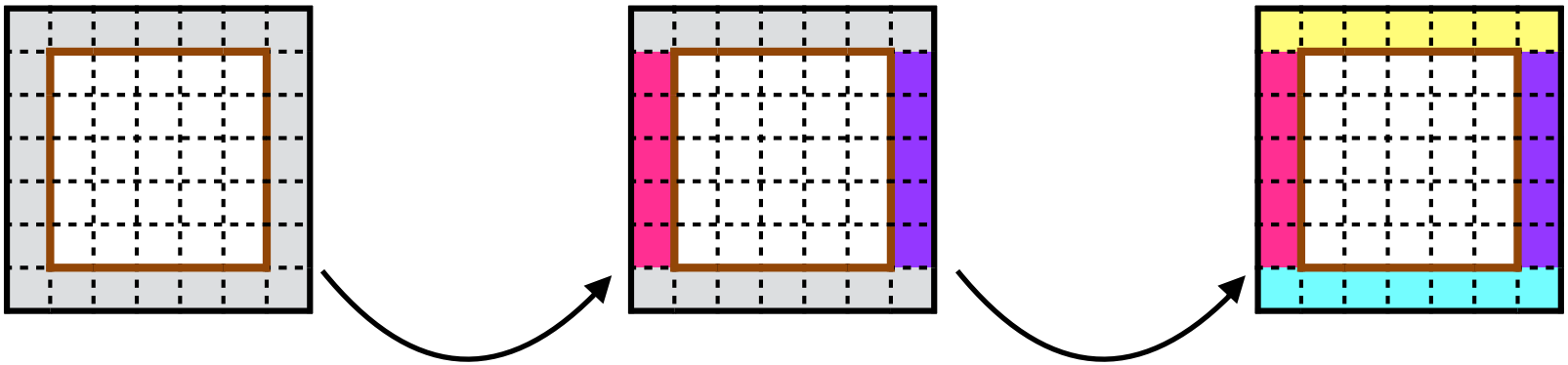


(2) add halos



Fill MPI halos : point-to-point

1) initialization
allocate MPI buffers



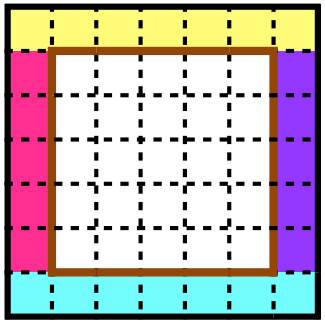
2 communication phases

2) East-West communications:
fill E-W MPI buffers
2 MPI_Isend
2 MPI_Recv
fill E-W halos

3) North-South communications:
fill N-S MPI buffer
2 MPI_Isend
2 MPI_Recv
fill N-S halos

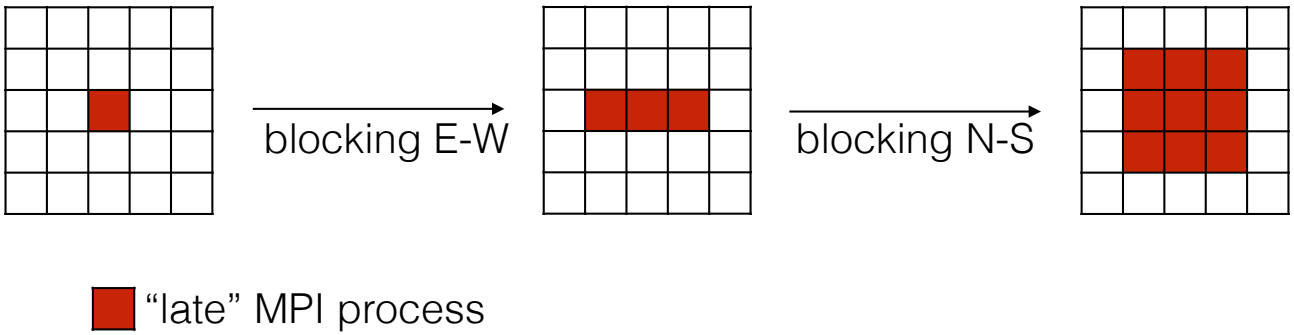
4) finalization
4 MPI_Wait
deallocate MPI buffers

communicating with 4 neighbors
blocking receive
fixed order: E-W next N-S



works in a perfect world with a perfect synchronism…

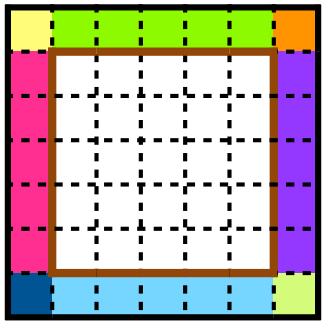
but significant “external perturbations/noise”



=> propagation of any perturbation…

=> suppress blocking and fixed order communications

do communication as soon as you can



independent communications
=> need 8 communications (sides+corners)

allocate MPI buffers
8 fill send buffers + MPI_Isend
While the **8** Recv are not done:
    8 tests with MPI_Prove: ready to Recv?
        if true: MPI_Recv + fill halos
8 MPI_Wait
dellocate MPI buffers

New solution is worse than the original one…

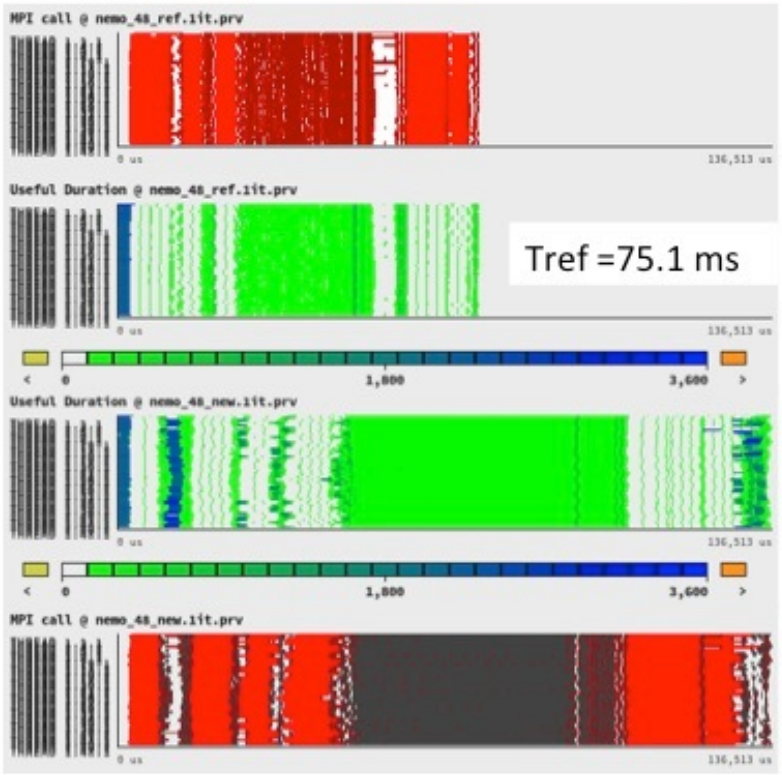
Why?

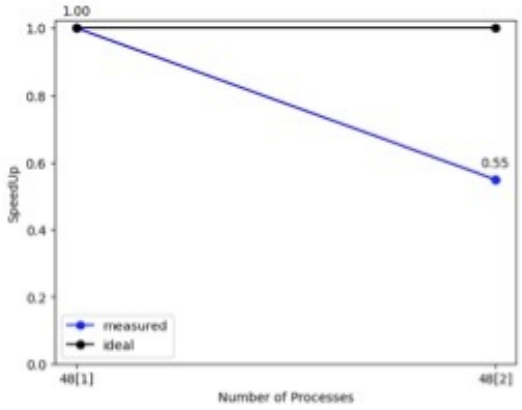
8 instead of 4 neighbors?
wrong coding?

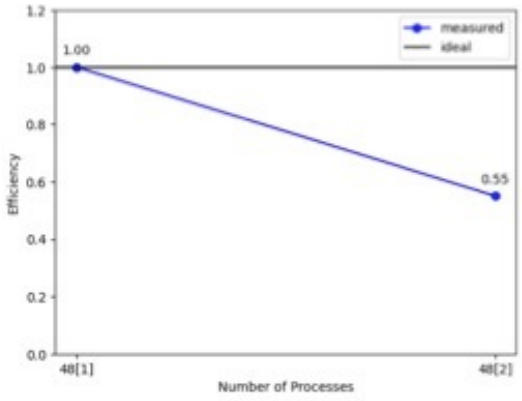
Scaling

reference

new

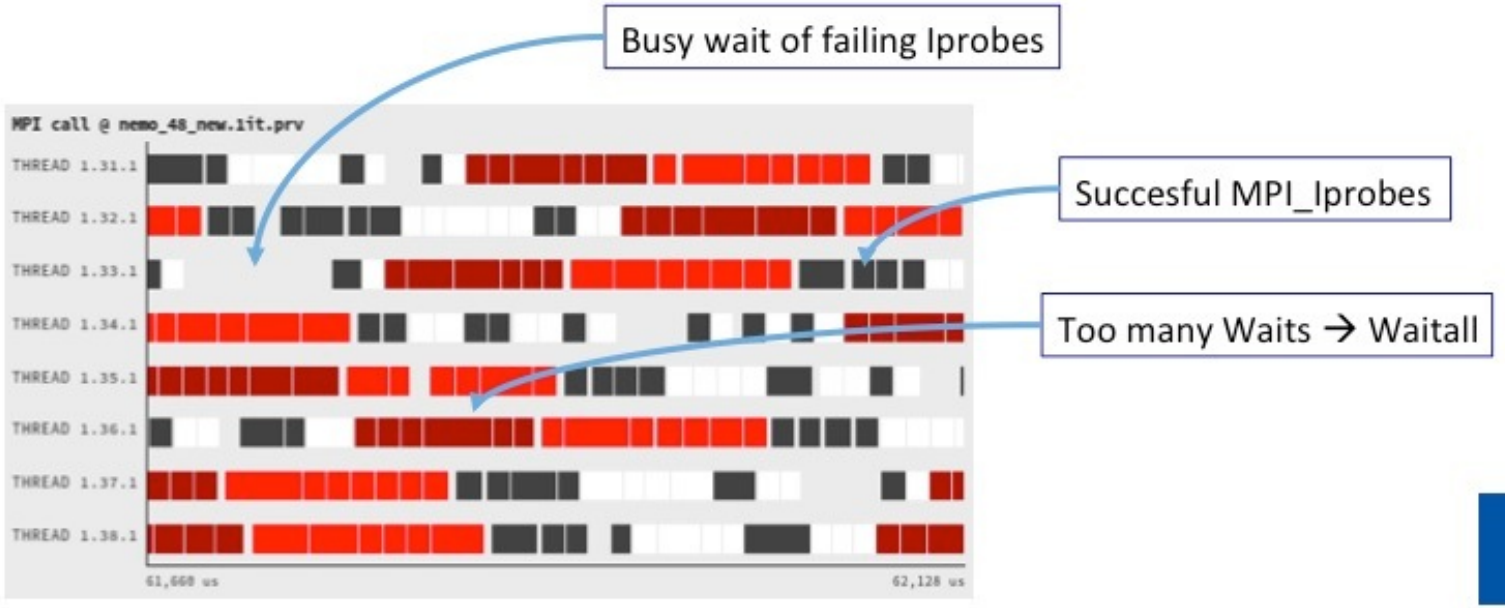






MPI calls

- Motivation for Iprobes ??
  - Overhead !!

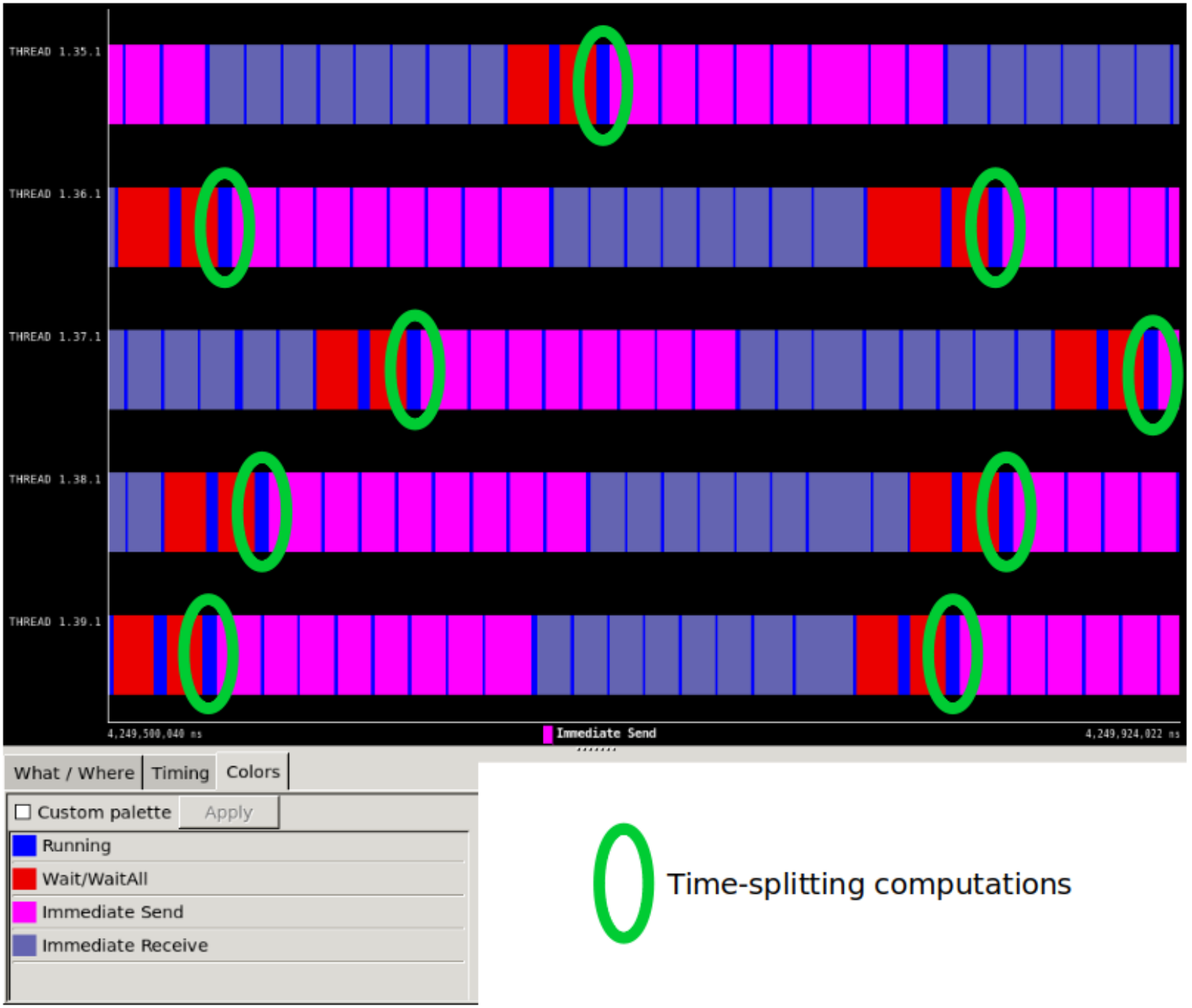


suppress MPI_Prove
MPI_Wait -> MPI_Waitall

allocate MPI buffers
8 fill buffer + MPI_Isend
8 MPI_Irecv
1 MPI_Waitall (Irecv)
fill halos
1 MPI_Waitall (Isend)
dellocate MPI buffers

small buffers:
keep in memory

1 MPI_Waitall
(Isend previous call)
8 fill buffer + MPI_Isend
8 MPI_Irecv
1 MPI_Waitall (Irecv)
fill halos


THREAD 1.35.1
THREAD 1.36.1
THREAD 1.37.1
THREAD 1.38.1
THREAD 1.39.1
4,249,500,040 ns
Immediate Send
4,249,924,022 ns
What / Where
Timing
Colors
Custom palette
Apply
Running
Wait/WaitAll
Immediate Send
Immediate Receive
Time-splitting computations

Use MPI
Neighbourhood
Collectives

initialization:
define communication
graph

fill buffers
1 MPI_Neighbor_alltoallv
fill halos

Use MPI
persistent calls

initialization:
define communication
graph
allocate buffers

fill buffers
1 MPI_Startall
1 MPI_Waitall
fill halos

Develop a specific configuration:

- contains only the critical part of the time step (high frequency MPI calls)
- perfect load balance
- 10x10 or 30x30 points in MPI subdomains
- 10 000 time steps

Light Timing:

- No profiling,
- just a simple MPI_Wtime to mesure 1 time step duration

Test 5 communication patterns:

- 4 EW-NS + waitall
- 4 EW-NS + waitall at next call
- 8 Isend + Irecv + waitall at next call
- 1 MPI Neighbourhood Collectives
- 1 MPI Persistent calls + waitall

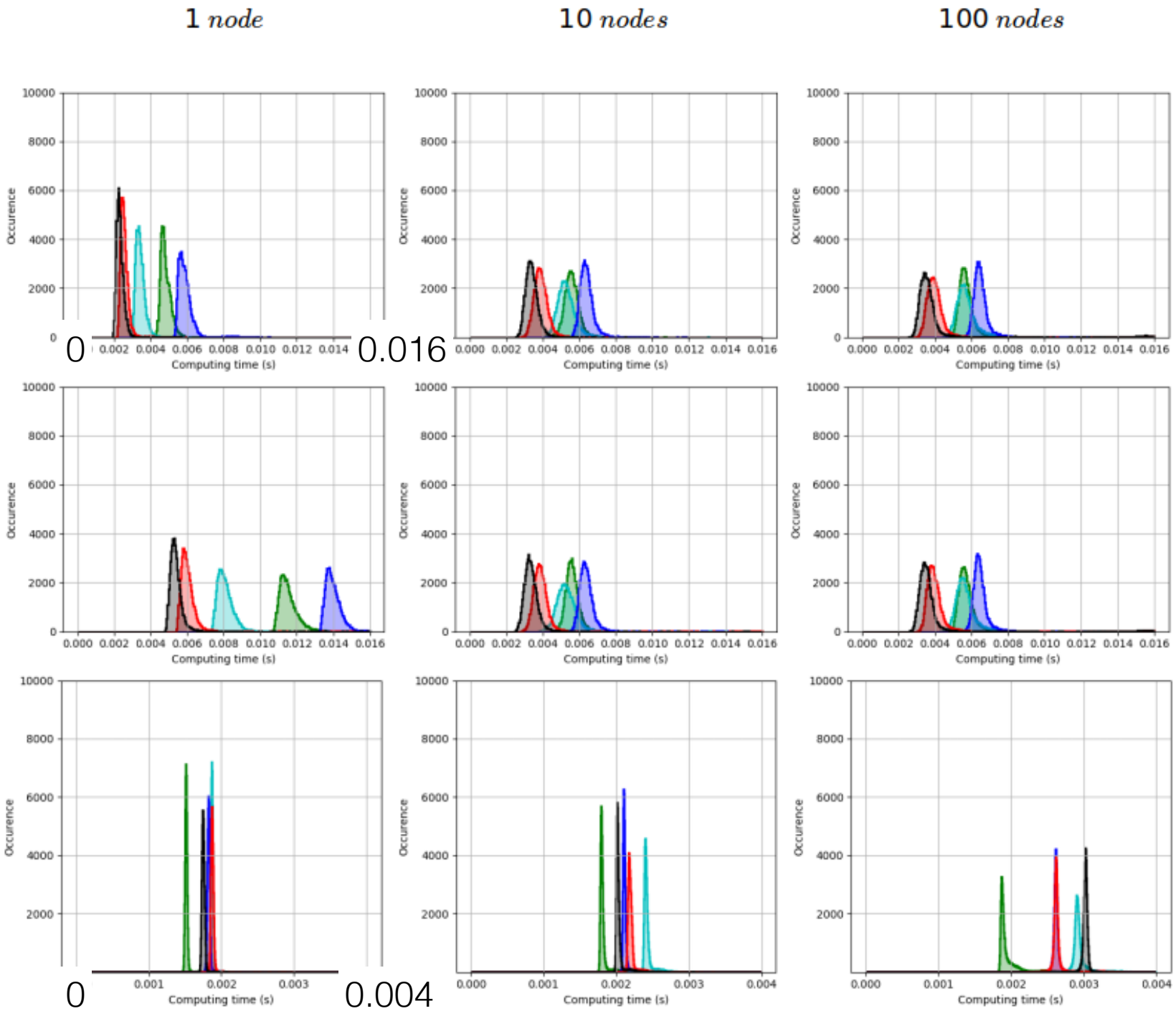
1 job on 1, 10 or 100 nodes:

repeat 5 times: (5 communication patterns) x (2 MPI subdomain size)

=> 50 runs/job

2 machines : Irene (48 cores/nodes) and Jean-Zay (40 cores/nodes)

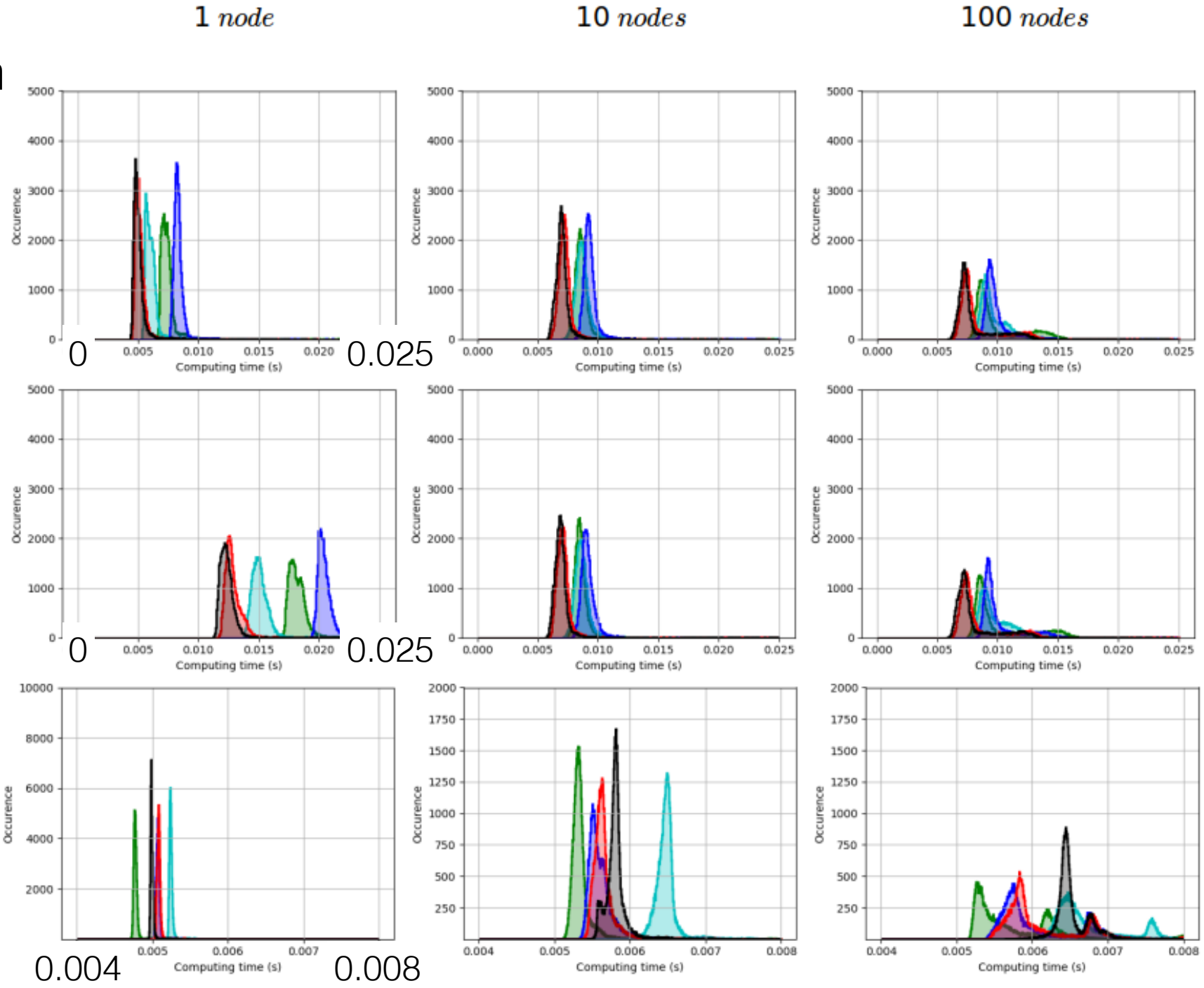
10x10 MPI subdomain



4 EW-NS + waitall 4 EW-NS + waitall at next call 8 Isend + Irecv + waitall at next call

1 MPI Neighbourhood Collectives 1 MPI Persistent calls + waitall

30x30 MPI subdomain



4 EW-NS + waitall 4 EW-NS + waitall at next call 8 Isend + Irecv + waitall at next call

1 MPI Neighbourhood Collectives 1 MPI Persistent calls + waitall

Conclusion

 and the
extrae/paraver analyses are an incredible tool
extremely useful for developers

but HPC is not an easy task!

difficulties to find a solution that works in all cases…
difficulties to find a solution that is not too heavy

Still a lot of work!