POP methodology: Identifying Performance Bottlenecks through Efficiency Metrics

Judit Giménez (BSC)

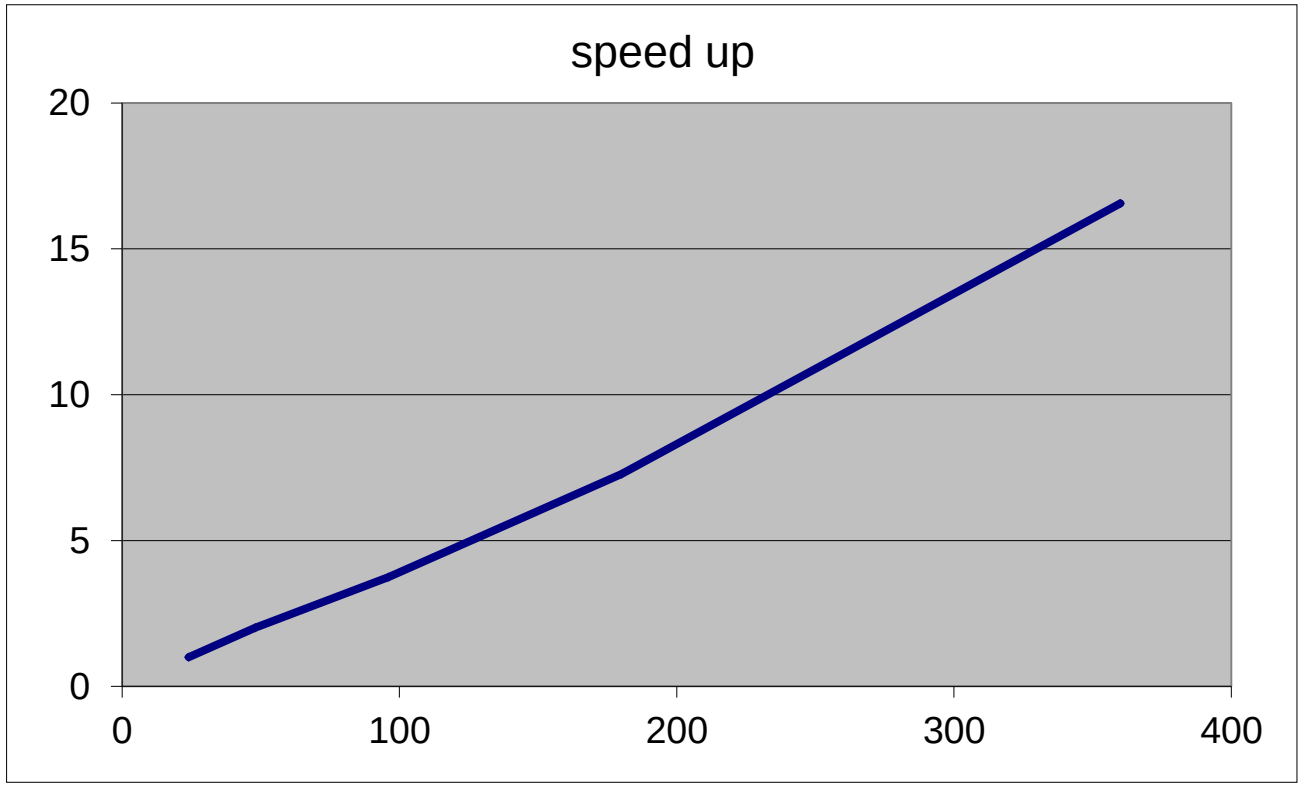
HORIZON-EUROHPC-JU-2023-COE

1 January 2024– 31 December 2026

Grant Agreement No 101143931

Should we be happy…

…with a perfect scaling?



- Strong scaling
- Unbalanced simulation

- What about the reference case?

Measuring the efficiency of a run

- Model the application run on a given platform (app + SW stack + HW)
- Run allocation: using a set of resources during a time interval

Parallel run **efficiency** = %useful
useful = time spent with the computations (outside parallel runtime)

- Reasons to be inside MPI parallel runtime (potential bottlenecks)
  - **Data transfer**
  - But also
    - **Work unbalance**
    - **Temporal unbalances and serializations**

Measuring the scaling efficiency

- Scaling of the parallel efficiency (and its components)
  - Is increasing/reducing the % useful time?
  - If it is reduced, which factor increases its weight?

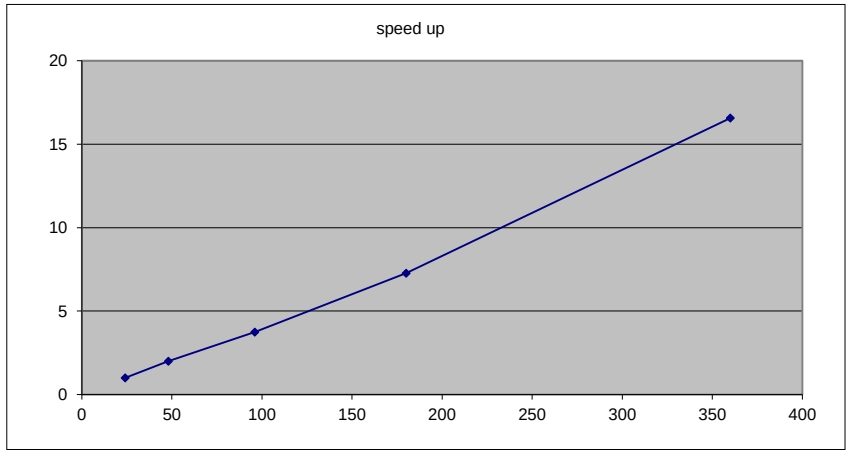
- Scaling of the **computations**
  - There is code replication (increase of **instructions**)?
  - Does the scale affect the **IPC**?
    - Strong scaling vs. weak scaling

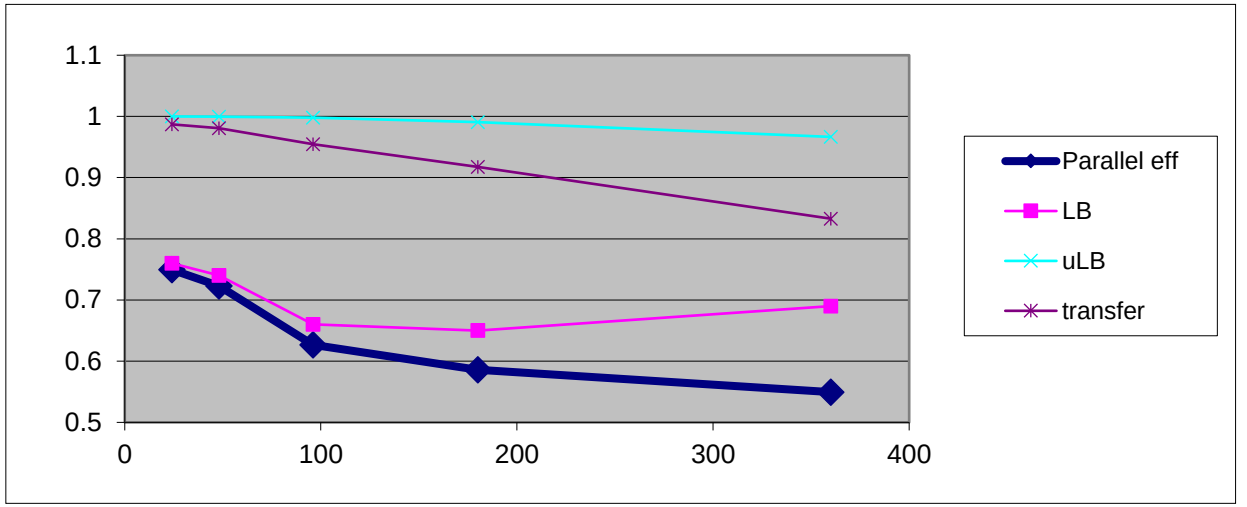
All factors can be expressed as a percentage or a positive number

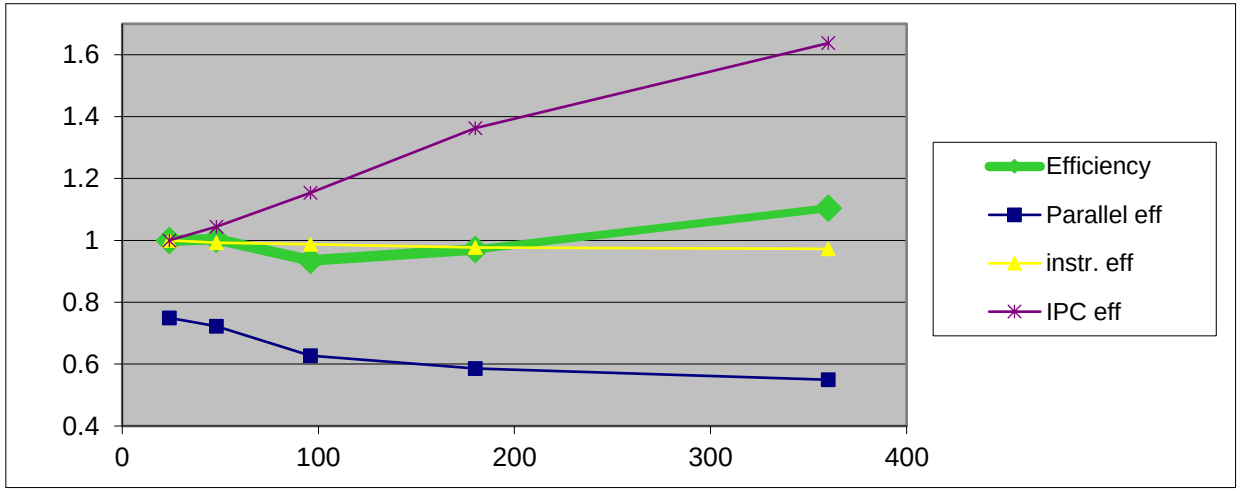
- always the higher the better

Why is my code scaling?

CG-POP mpi2s1D - 180x120



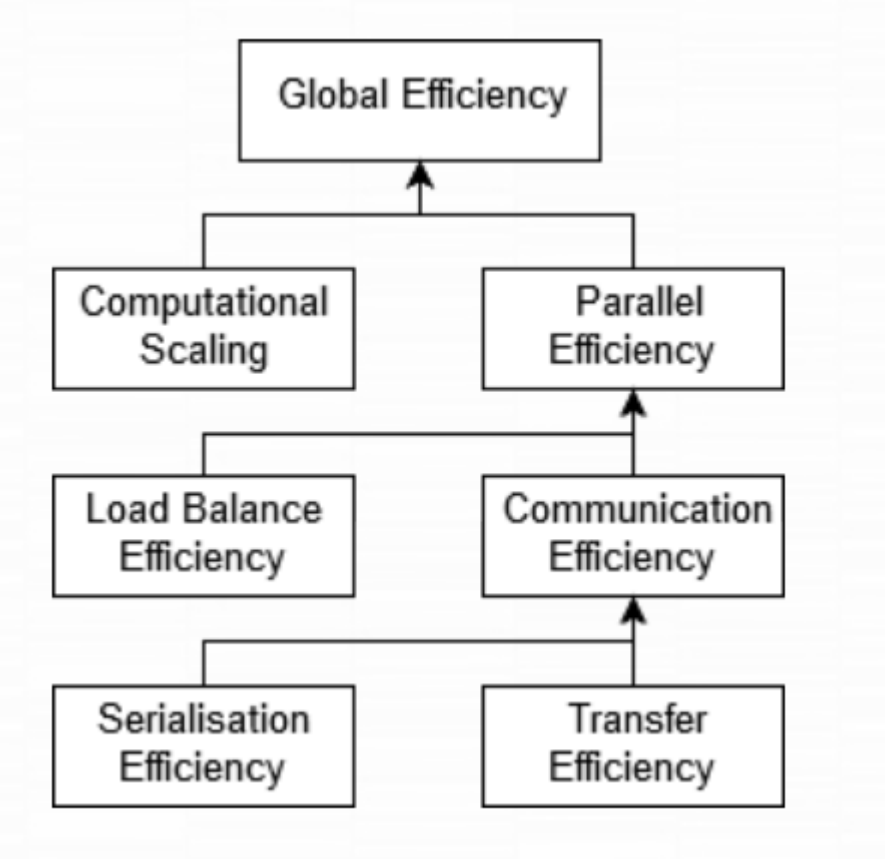




POP's efficiency metrics

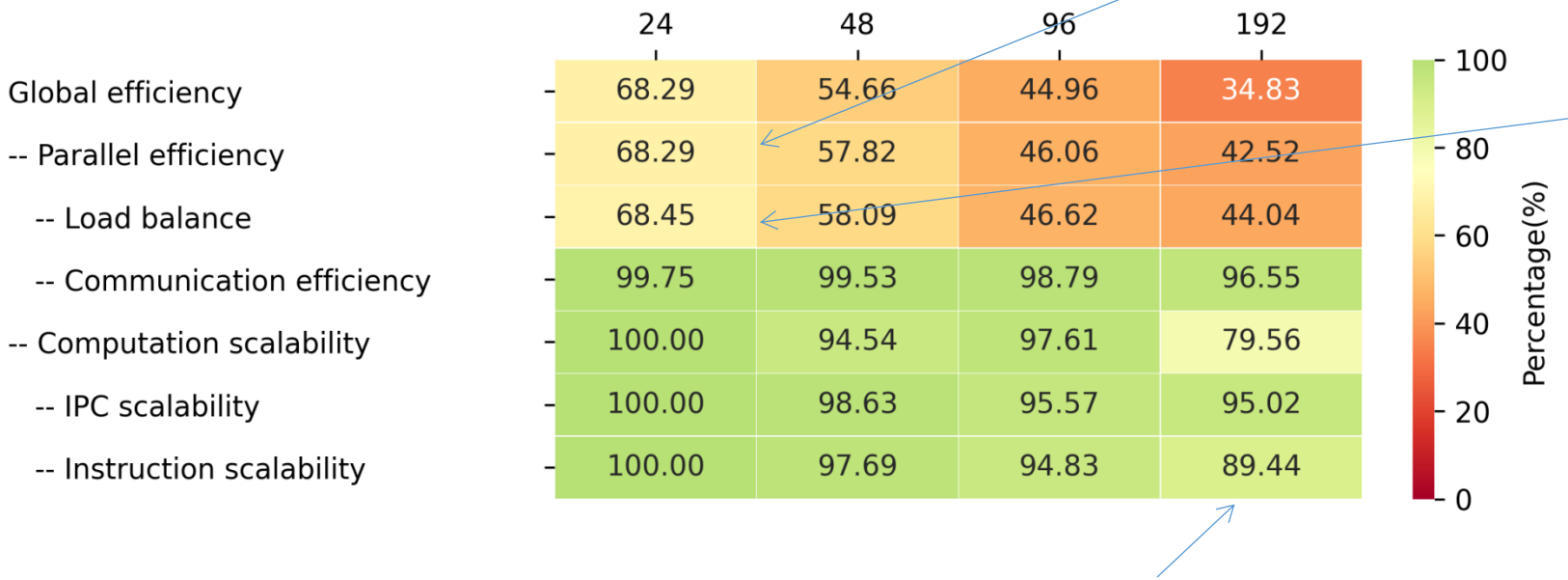
- The POP methodology defines hierarchies of metrics (pop-coe.eu/sites/default/files/pop_files/metrics.pdf)

- Global Efficiency (GE)
  - Parallel Efficiency (PE)
    - Load Balance Efficiency (LB)
    - Communication Efficiency (CommE)
      - Serialization Efficiency (SerE)
      - Transfer Efficiency (TE)
  - Computation Efficiency (CompE)
    - IPC Scaling
    - Instruction Scaling
    - Frequency Scaling



Efficiency model analysis – examples

- Looking at the reported efficiencies:
  - Above 0.8 may be considered good
  - Execution vs. Scaling
  - These numbers will drive the analysis

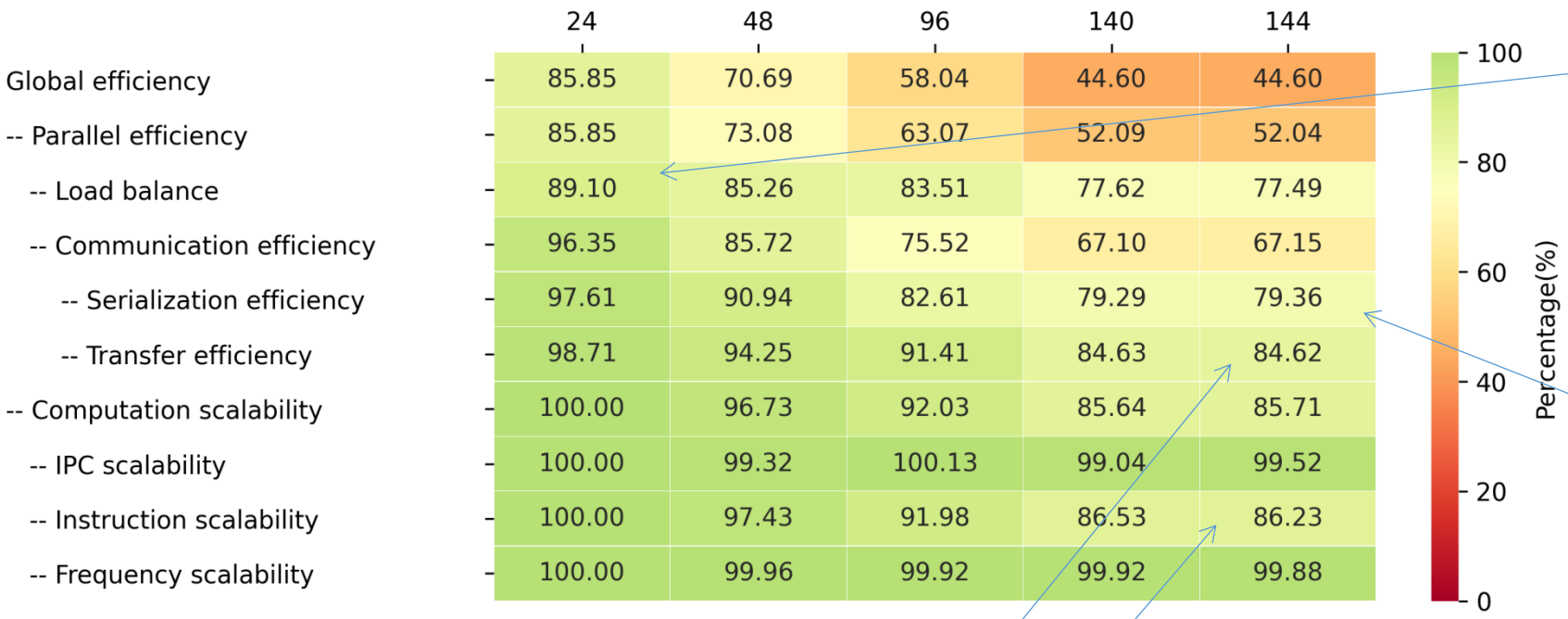


Poor parallel efficiency for the lowest core count

Lower value for the smallest core count and main factor that limits scalability

Big drop – code owner hints to explain it

Efficiency model analysis – examples



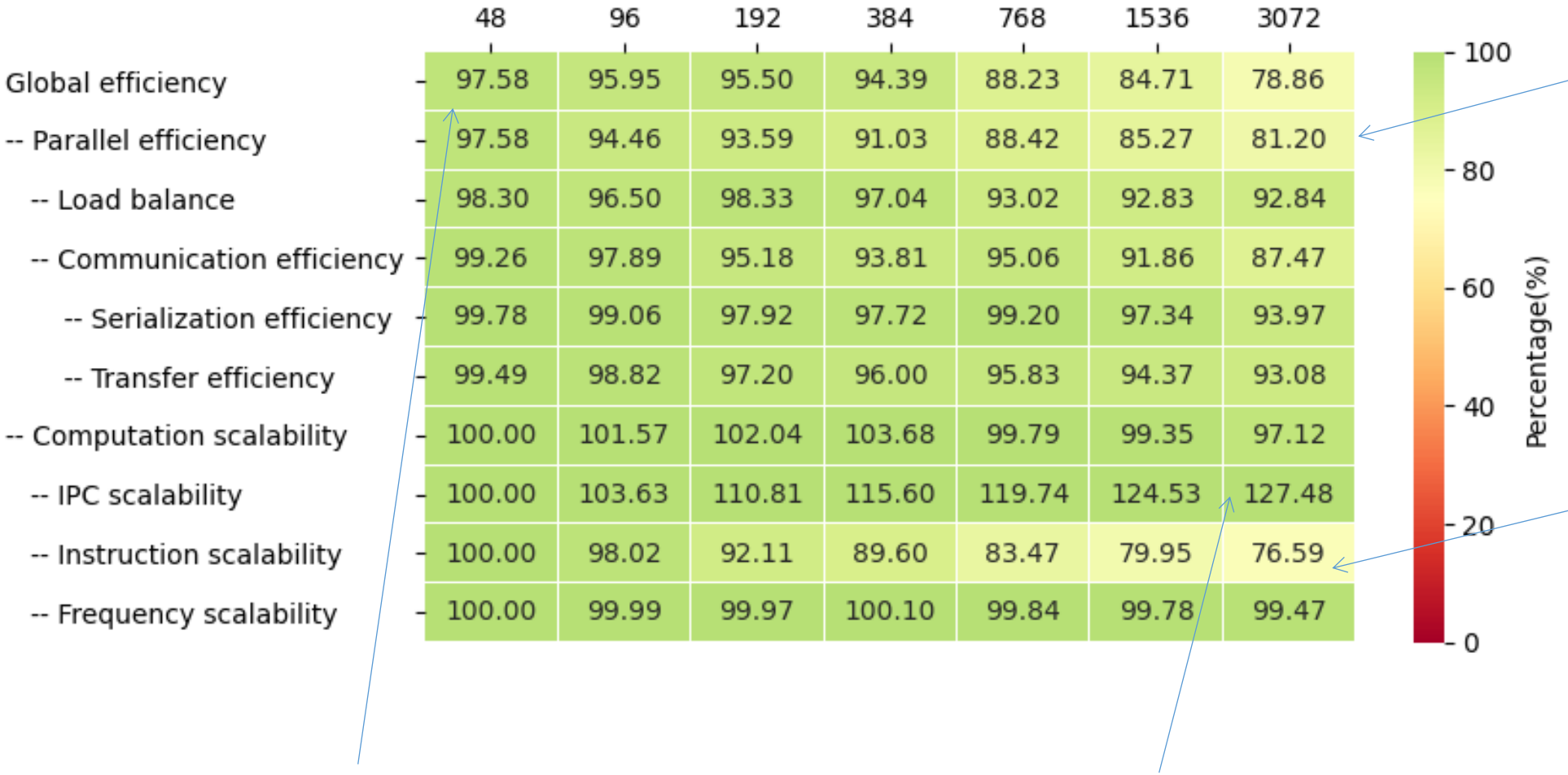
Lower value for the smallest core count

Main factor that limits scalability (biggest drop)

Other factors limiting scalability

- Data transfer
- Code replication

Efficiency model analysis – examples



	48	96	192	384	768	1536	3072
Global efficiency	97.58	95.95	95.50	94.39	88.23	84.71	78.86
-- Parallel efficiency	97.58	94.46	93.59	91.03	88.42	85.27	81.20
-- Load balance	98.30	96.50	98.33	97.04	93.02	92.83	92.84
-- Communication efficiency	99.26	97.89	95.18	93.81	95.06	91.86	87.47
-- Serialization efficiency	99.78	99.06	97.92	97.72	99.20	97.34	93.97
-- Transfer efficiency	99.49	98.82	97.20	96.00	95.83	94.37	93.08
-- Computation scalability	100.00	101.57	102.04	103.68	99.79	99.35	97.12
-- IPC scalability	100.00	103.63	110.81	115.60	119.74	124.53	127.48
-- Instruction scalability	100.00	98.02	92.11	89.60	83.47	79.95	76.59
-- Frequency scalability	100.00	99.99	99.97	100.10	99.84	99.78	99.47

Smooth degradation of parallel eff. with similar contribution of LB, Ser, Transfer

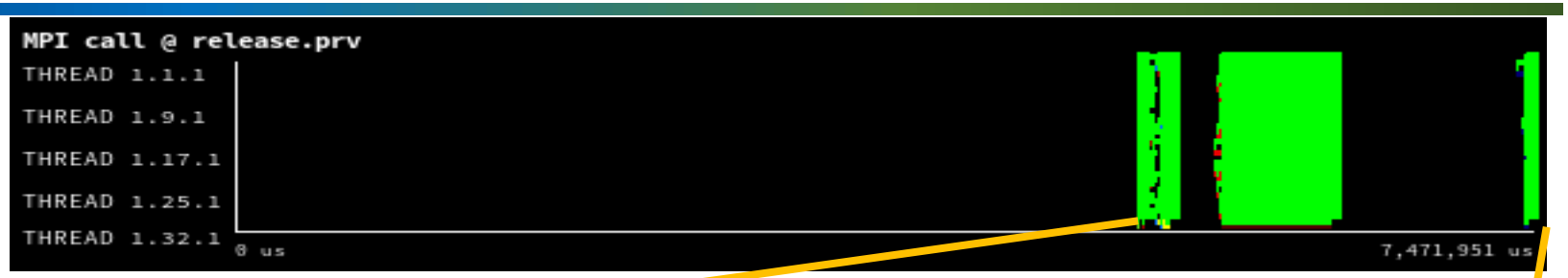
Main factor that will limit scalability (IPC cannot grow forever)

Very good efficiency with the smallest core count

Improvement on IPC compensates increase of instructions

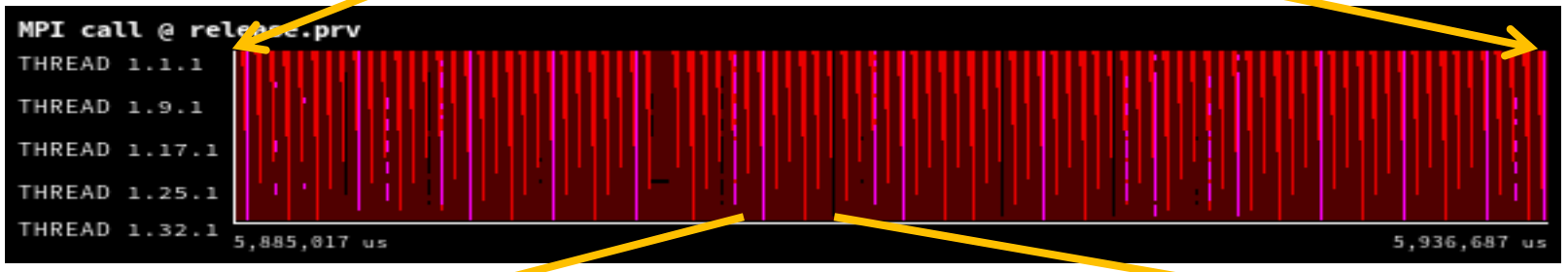
Selecting FoA and understanding why

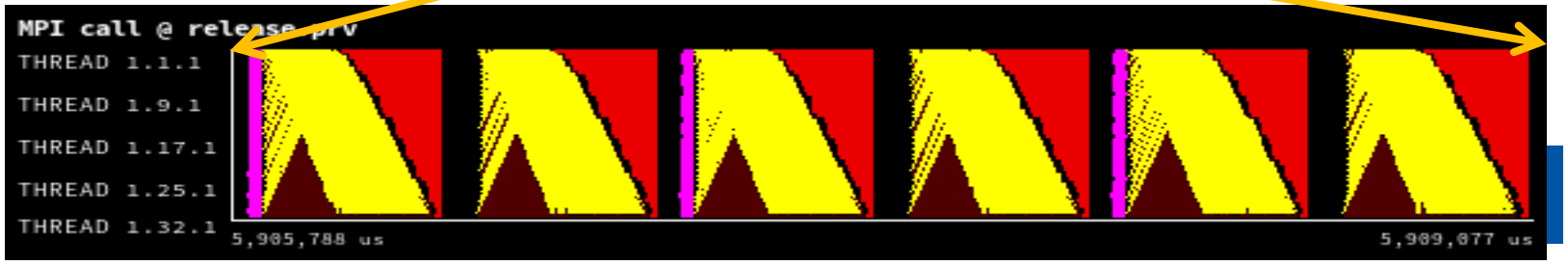
Parallel efficiency 93.28
Communication eff. 93.84



Parallel efficiency 77.93
Communication eff. 79.79

Parallel efficiency 28.84
Communication eff. 30.42





Understanding why – LB efficiency

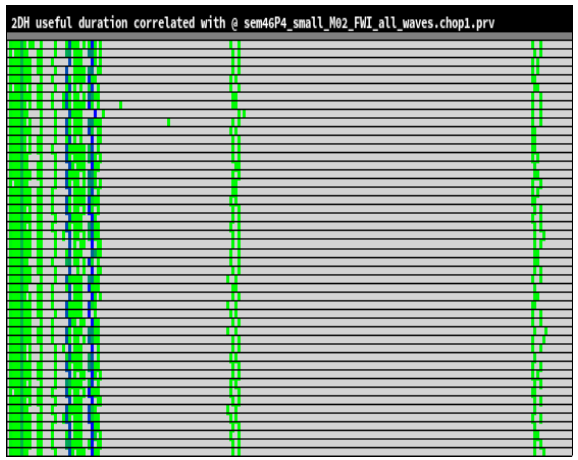
Ex 1: OpenMP

	Running
THREAD 1.1.1	98.65 %
THREAD 1.1.2	57.08 %
Total	155.74 %
Average	77.87 %
Maximum	98.65 %
Minimum	57.08 %
StDev	20.78 %
Avg/Max	0.79

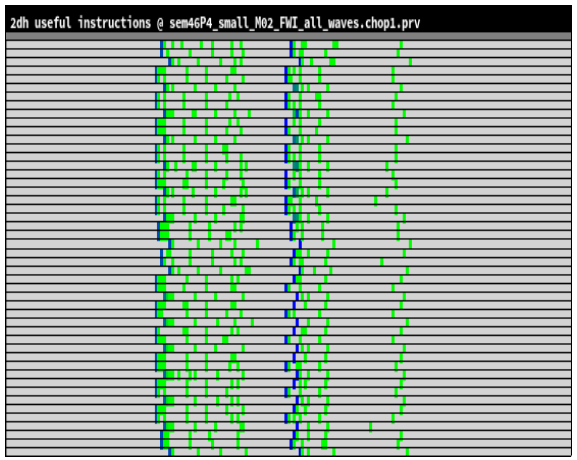
	Running
THREAD 1.1.1	99.27 %
THREAD 1.1.2	26.60 %
THREAD 1.1.3	26.76 %
THREAD 1.1.4	26.30 %
THREAD 1.1.5	26.53 %
THREAD 1.1.6	26.81 %
THREAD 1.1.7	26.42 %
THREAD 1.1.8	26.41 %
Total	285.09 %
Average	35.64 %
Maximum	99.27 %
Minimum	26.30 %
StDev	24.05 %
Avg/Max	0.36

Regions not parallelized with OpenMP

Ex 2: MPI



duration



instructions

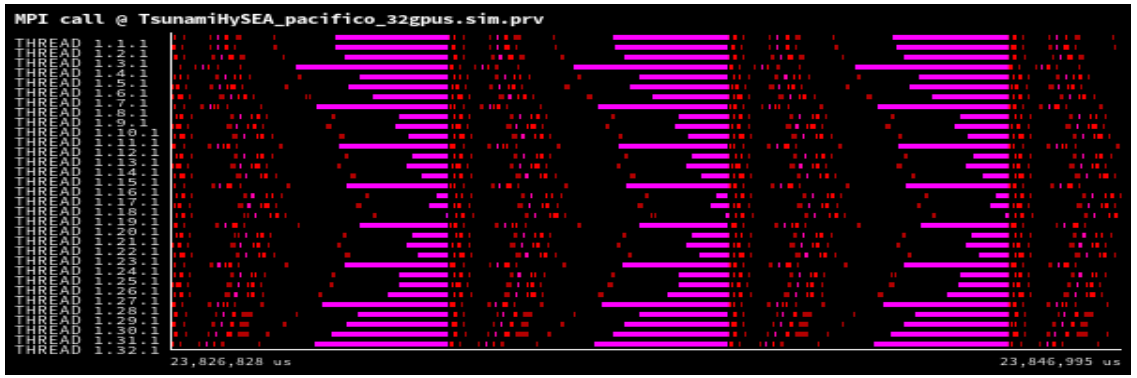
Structured instructions unbalance

Understanding why – transfer eff.

Ex 1: **Load Balance 76%** Serialization 99%, **Transfer 84%**

Real run

Ideal network



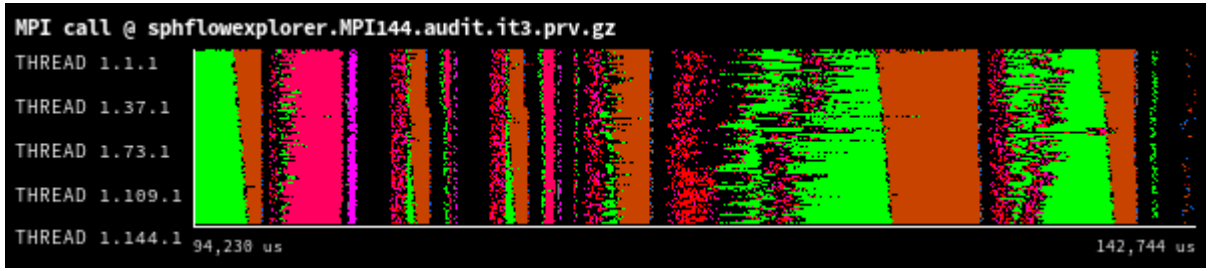
With instantaneous communications

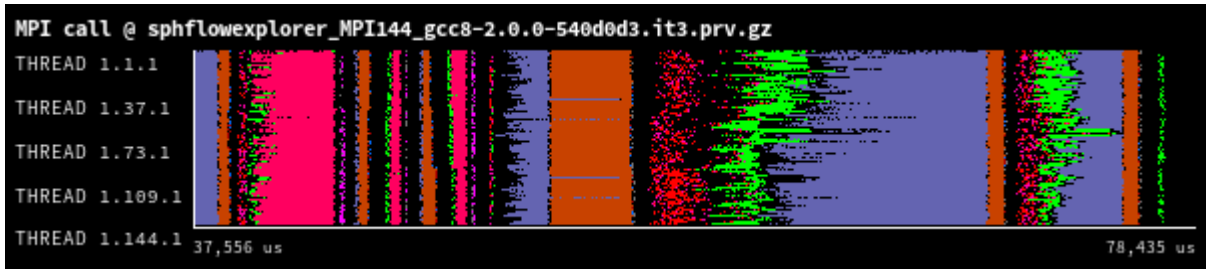
- Most of the point to point calls (red) almost disappear → data transferred in point to point is limited by network resources
- The allreduce (pink) increases a bit → absorbs previous load imbalance

Understanding why – serialization eff.

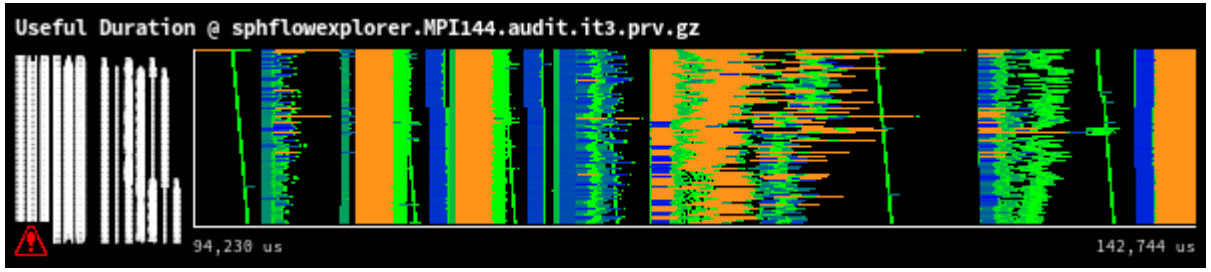
Ex 1: Audited version (left) vs. improved version (right)

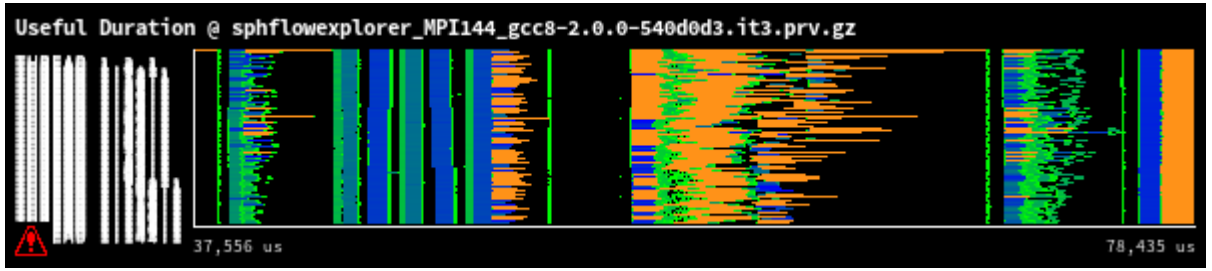
MPI calls





Useful duration

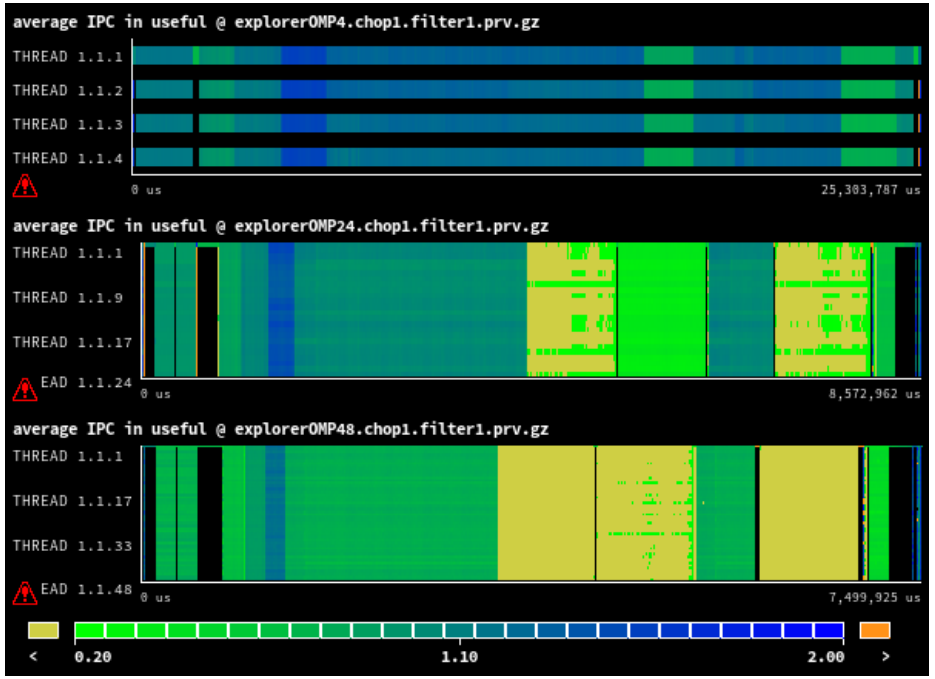




Serialization detected as stepped computations → running these phases in parallel obtain a reduction of 22% in the region.

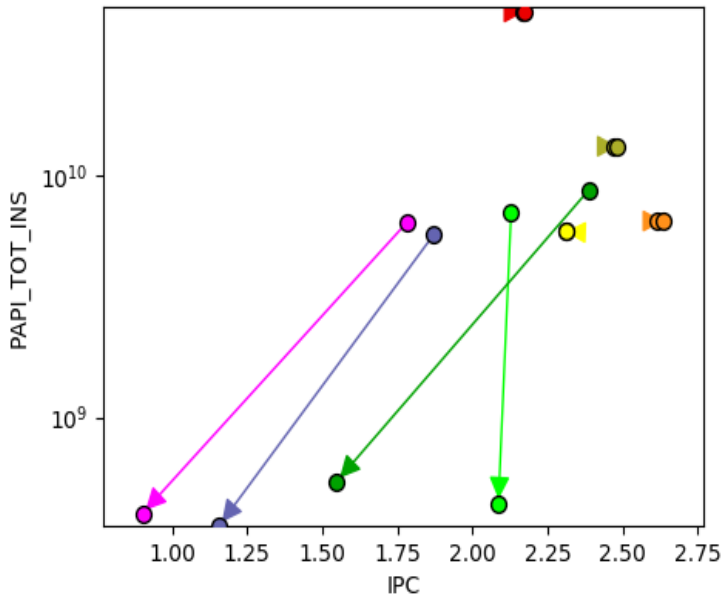
Understanding why – IPC scaling

Ex 1: OpenMP



IPC reduction is on almost all the phases

Ex 2: MPI



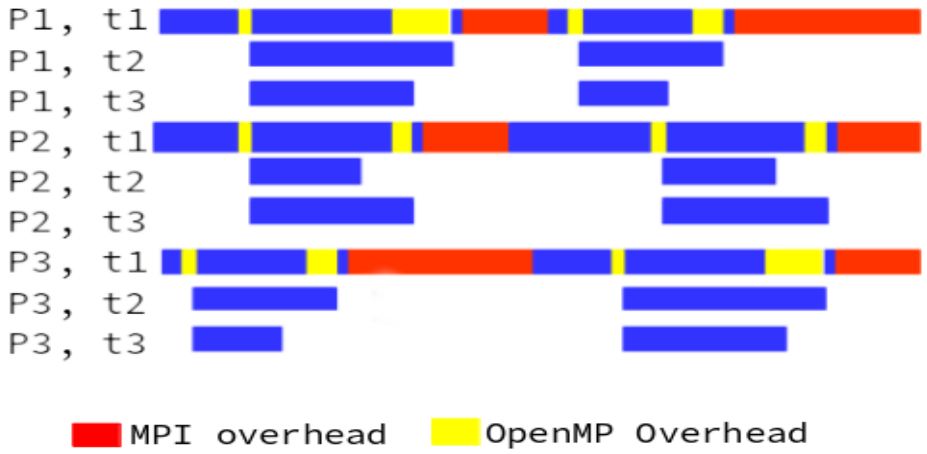
Good one: light green region

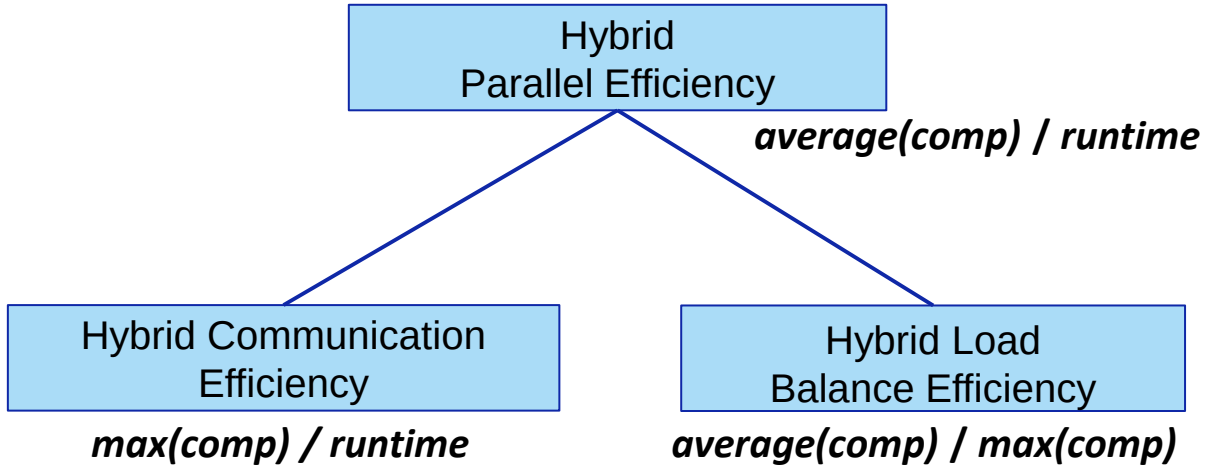
3 regions suffer IPC degradation (arrows to the left)

[4 regions have code replication]

Efficiency model extensions

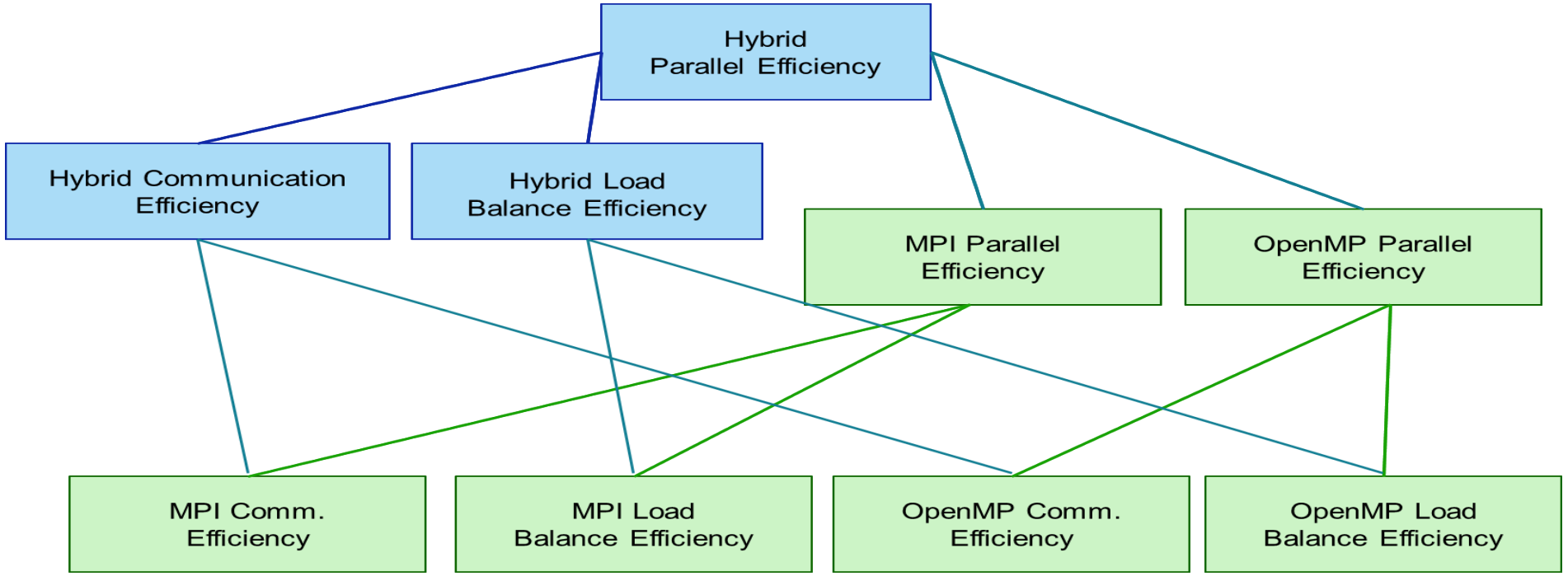
Applying the original model to hybrid codes





- Efficiency computed as percentage of time outside the two parallel runtimes
- Useful as a first step to distinguish between Load Balance and Communication. But how to dig down?
  - Efficiencies are mixing inefficiencies from MPI and OpenMP → Need to distribute the blame between the two programming models

What do we want?



- Global load balance and communication concepts can be mapped to any parallel programming paradigm
- The efficiencies at hybrid level collapse the contributions from the two programming models

A very simple case

Duration of the computing regions

Hybrid efficiencies [%]
- Parallel efficiency 47.07
- Load Balance 52.09
- Communication 90.37

- From the MPI point of view, OpenMP runtime is as useful as computation

MPI calls

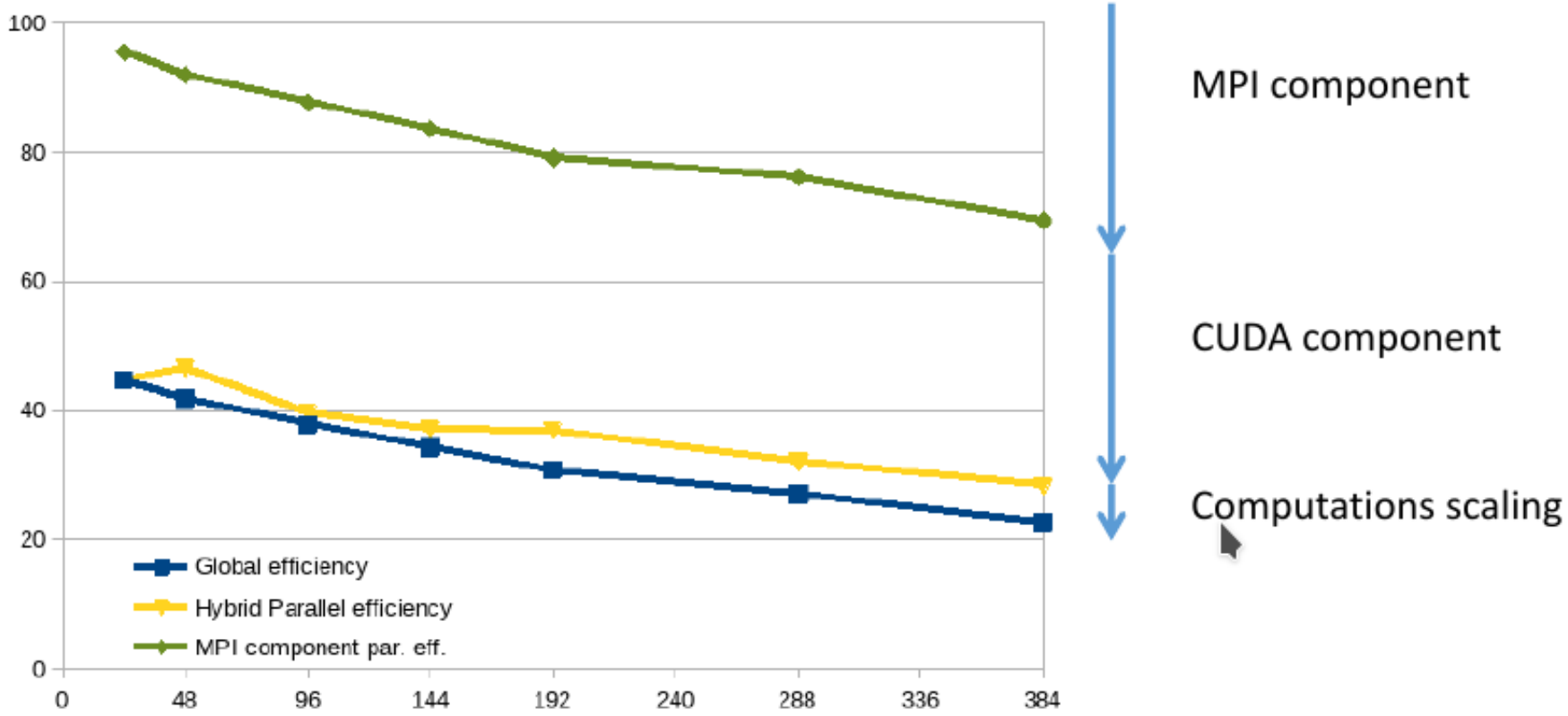
MPI efficiencies [%]
- Parallel efficiency 95.97
- Load Balance 98.87
- Communication 97.06

- Whatever cannot be blamed on MPI is caused by OpenMP

OpenMP parallel functions

OpenMP efficiencies [%]
- Parallel efficiency 49.05
- Load Balance 52.69
- Communication 93.11

Same approach applied to MPI+CUDA

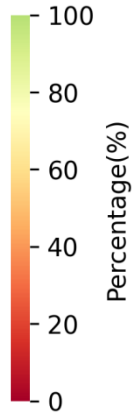


- Bigger impact from CUDA component with a similar contribution in all the scales
- MPI contribution increases with the scale, but still lower than CUDA

A different approach for MPI+CUDA

- Some codes use efficiently the GPUs but intentionally do not use the CPUs → **lower metrics focusing on the GPUs only**

	1	4	8	16	24	32	40	48	56	64
-- Global Parallel efficiency	47.43	47.19	46.80	46.27	45.73	44.48	43.61	43.30	42.34	42.04
-- Global Load balance	50.00	49.91	49.97	49.91	49.92	49.83	49.59	49.64	49.59	49.40
-- GPUs Global efficiency	94.85	94.08	92.94	91.03	89.33	85.99	83.83	82.77	80.17	79.00
-- GPUs Parallel efficiency	94.85	94.33	93.53	92.39	91.23	88.67	86.87	86.18	84.19	83.52
-- GPUs Load Balance	100.00	99.79	99.86	99.65	99.59	99.33	98.77	98.80	98.62	98.15
-- GPUs Communication efficiency	94.85	94.53	93.66	92.71	91.61	89.27	87.95	87.23	85.37	85.09
-- GPUs Computation scalability	100.00	99.73	99.37	98.53	97.92	96.98	96.50	96.05	95.22	94.59



- Considering all the resources allocated, the efficiencies are around 40-50% because the work is concentrated in the GPUs (reported as unbalance). The small degradation in the parallel efficiency is also observed in the GPUs metrics.
- The GPUs global efficiency reports a degradation with the scale that is related mainly with the communication efficiency but also with the computation scalability. The load balance between GPUs is ok for all the scales.

How to calculate the metrics?

- From Extrae/Paraver data
  - **BSC's Basic Analysis** module metrics (tools.bsc.es/downloads)
- From Score-P traces
  - **Since CUBE version 4.5** (www.scalasca.org/software/cube-4.x/download.html)
- This training will show you how to use them!

Performance Optimisation and Productivity

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 Union's Horizon 2020 research and innovation programme under grant agreement No 676553 and 824080.