

HPC CoE POP3

Concept, Method and Praxis

Marc Schlütter, Jülich Supercomputing Centre

HORIZON-EUROHPC-JU-2023-COE

EuroHPC Joint Undertaking

1 January 2024– 31 December 2026

Grant Agreement No 101143931

POP CoE

- A **Centre of Excellence**
  - On **Performance Optimisation and Productivity**
  - Promoting **best practices in parallel programming**
- Providing **FREE Services**
  - Precise understanding of application execution and system behaviour
  - Suggestion/support on how to refactor code in the most productive way
- **Horizontal**
  - Transversal across application areas, platforms, scales
- **For (EuroHPC) academic AND industrial codes and users !**

EuroHPC
Joint Undertaking

Partners

- **Who?**
  - BSC, ES (coordinator)
  - HLRS, DE
  - INESC-ID, PT
  - IT4I, CZ
  - JSC, DE
  - RWTH Aachen, IT Center, DE
  - TERATEC, FR
  - UVSQ, FR

A team with

- Excellence in performance tools and tuning
- Excellence in programming models and practices
- Research and development background AND
  proven commitment in application to real academic and industrial use cases

Motivation

Why?

- Complexity of parallel computers and codes
  - ⇨ Frequent lack of quantified understanding of actual behaviour
  - ⇨ Not clear most productive direction of code refactoring
- Important to maximize efficiency (performance, energy usage) of compute intensive applications and productivity of development efforts

What?

- Parallel programs, mainly MPI & OpenMP
  - Although also OpenACC, OpenCL, CUDA/HIP, …

EuroHPC
Joint Undertaking

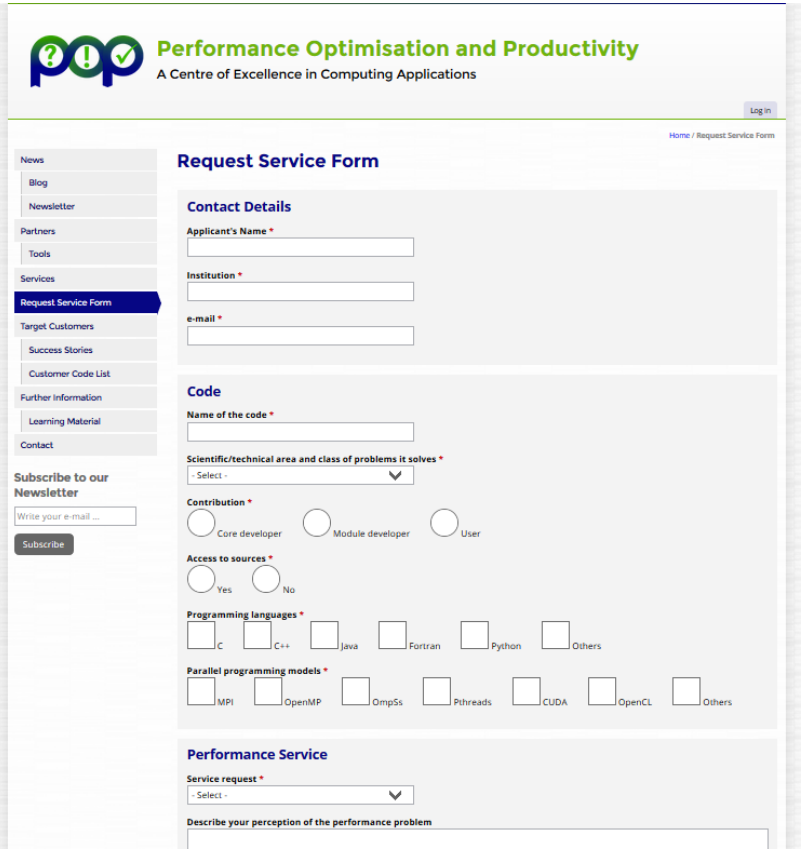
The Process …

When?

January 2024 – December 2026

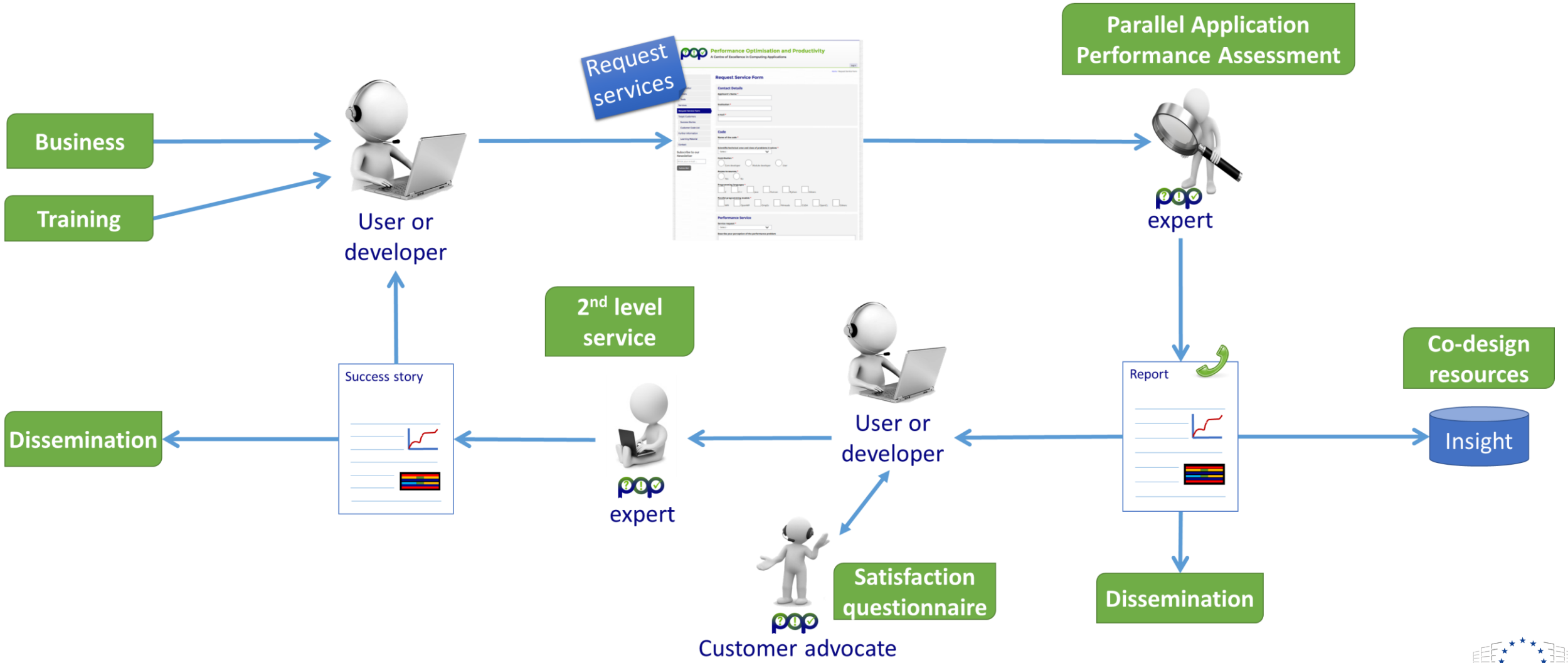
How?

- Apply
  - Fill in short questionnaire describing application and needs https://pop-coe.eu/request-service-form
  - Questions? Ask pop@bsc.es
- Selection/assignment process
- Install tools @ your production machine (local, EuroHPC, …)
- Interactively: Gather data → Analysis → Report

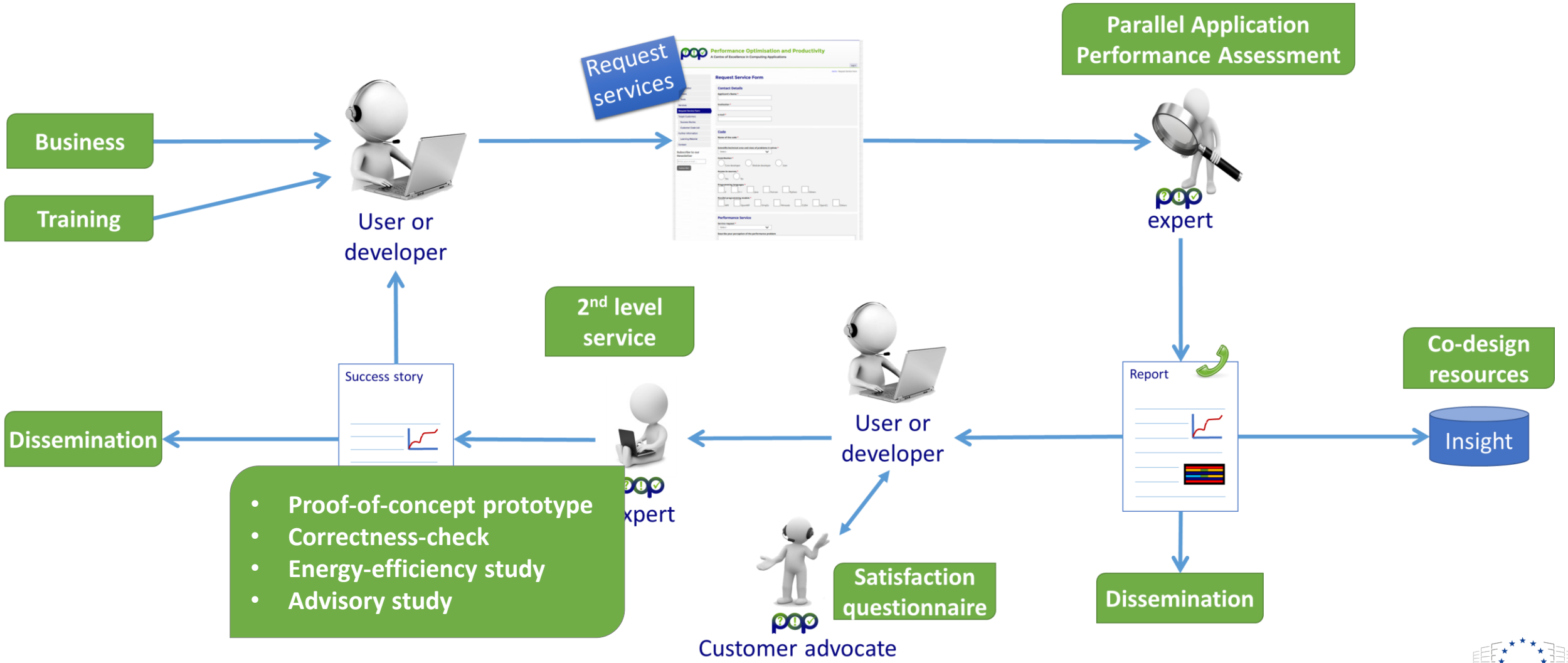


EuroHPC Joint Undertaking

The Process …


Business
Training
User or developer
Request services
Performance Optimisation and Productivity
A Centre of Excellence in Computing Applications
Request Service Form
Contact Details
Code
Performance Service
Parallel Application Performance Assessment
expert
Report
Co-design resources
Insight
Dissemination
Satisfaction questionnaire
Customer advocate
User or developer
2nd level service
expert
Success story
Dissemination

The Process …



EuroHPC Joint Undertaking

Tools

- **Install and use already available measurement & analysis technology**
  - Analysis and predictive capabilities
  - Delivering insight
    - Highly detailed
    - Up to extreme scale
- **Open-source toolsets**
  - **Extrae + Paraver + Dimemas**
  - **Score-P + CUBE + Scalasca**
  - Archer + MUST + OTF-CPT
  - MAQAO
  - CARM
  - MERIC + RADAR
- **Vendor/ISV toolsets**

  (if available at customer site)
  - Vampir
  - Intel tools
  - Nvidia tools
  - Cray/HPE tools
  - Linaro tools

EuroHPC
Joint Undertaking



POP Methodology & Metrics

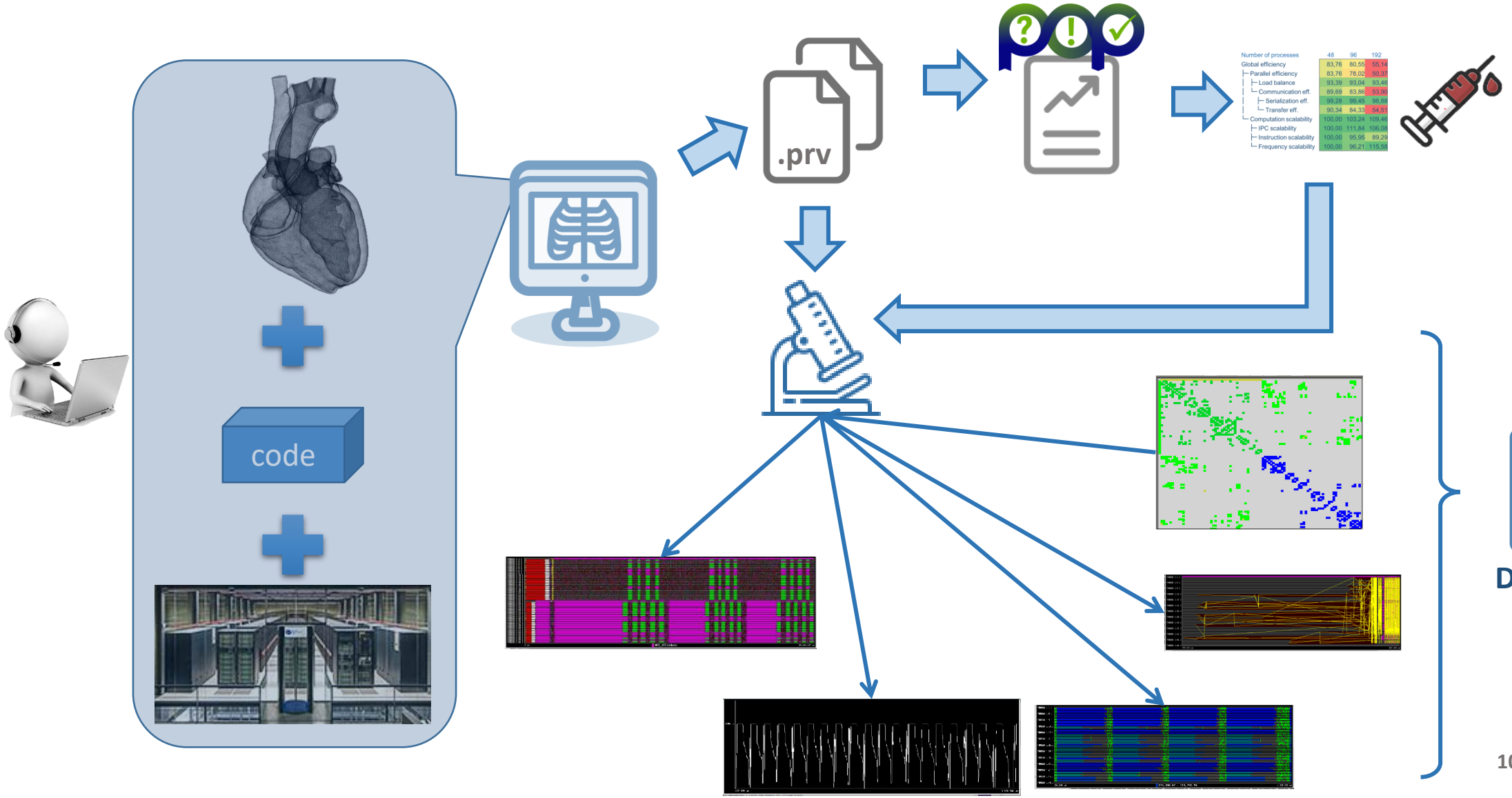
HORIZON-EUROHPC-JU-2023-COE

EuroHPC Joint Undertaking

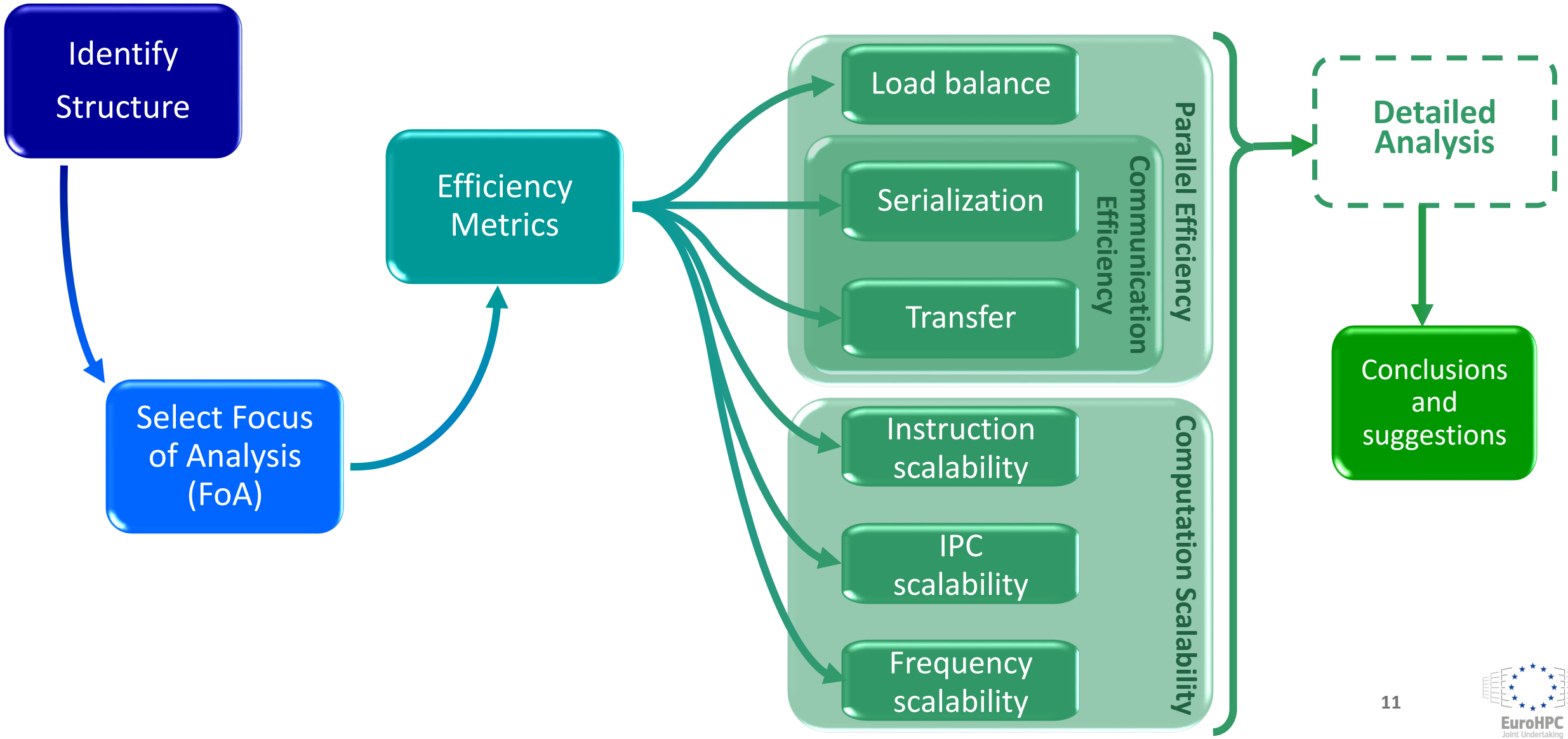
1 January 2024– 31 December 2026

Grant Agreement No 101143931

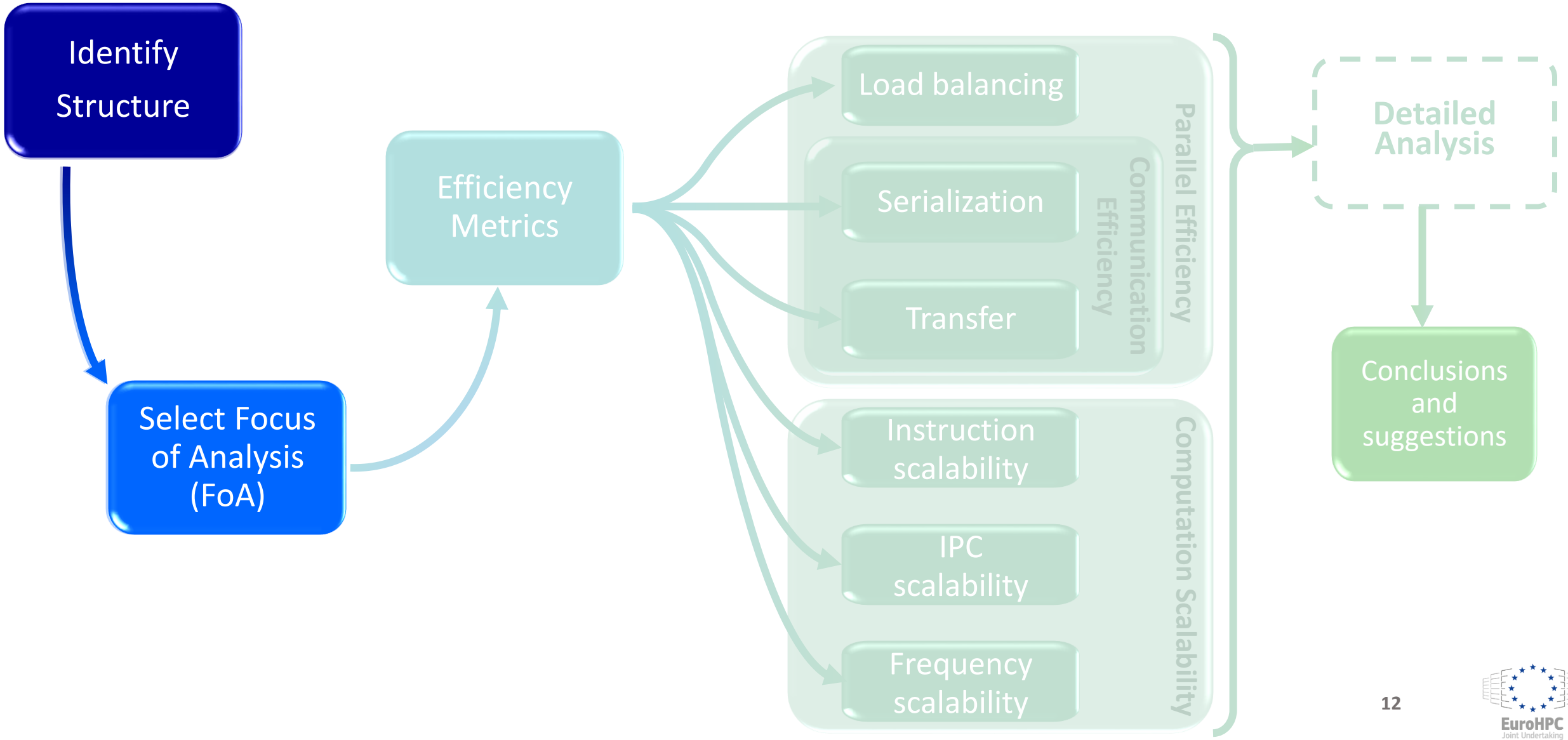
Performance analysis



The journey

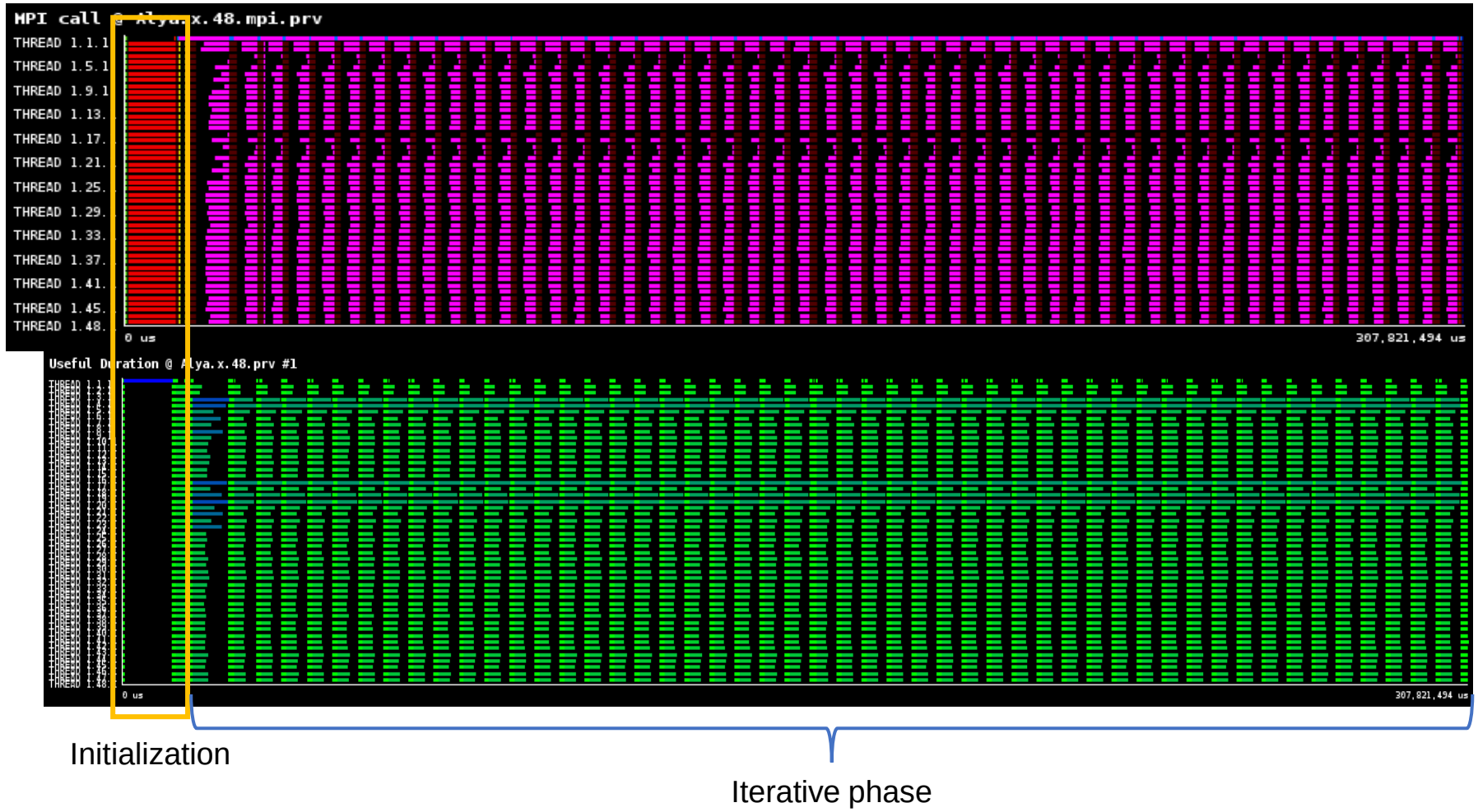

Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balance
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

The journey


Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Identify structure

- Usually use “MPI calls” view or “Useful Duration”



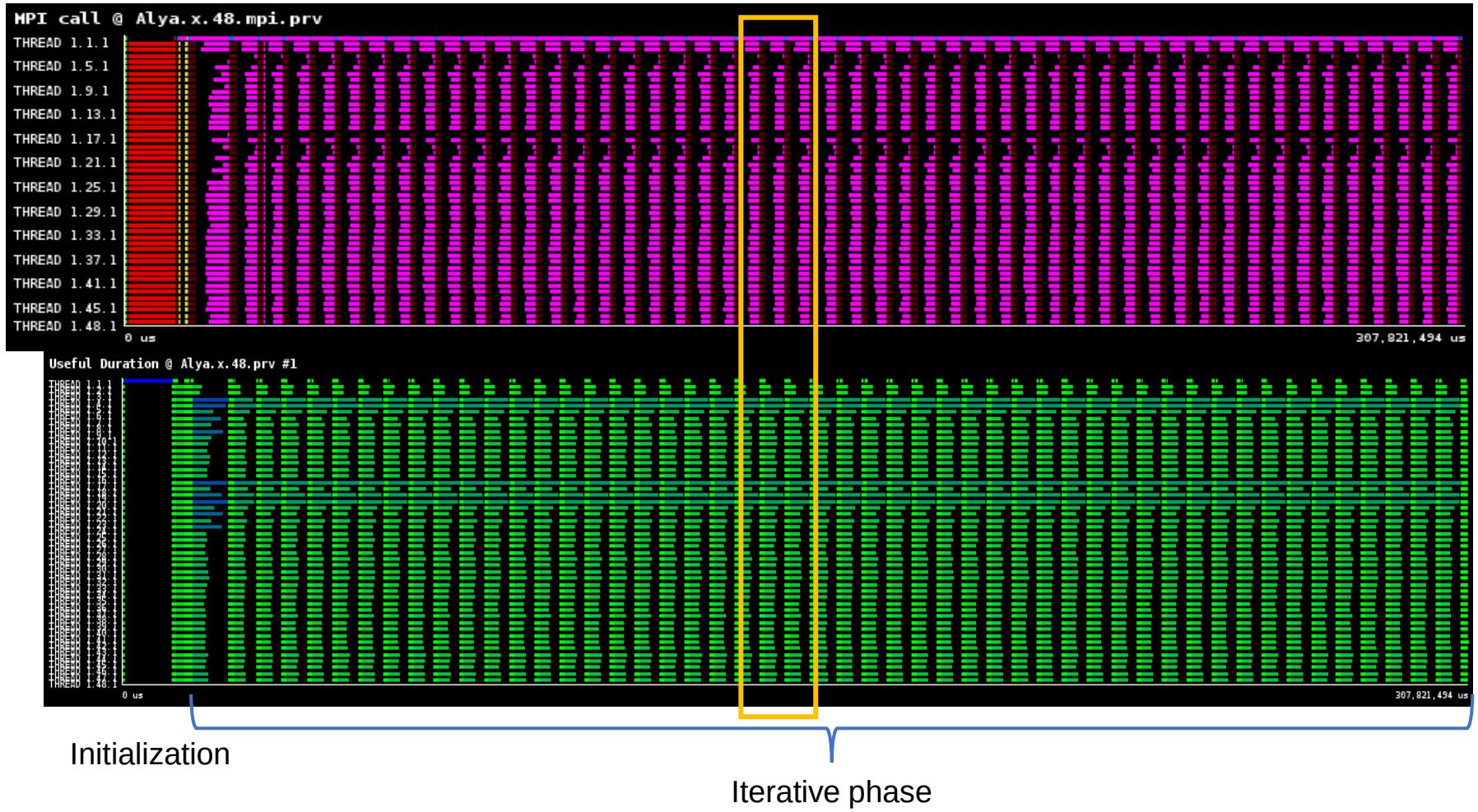
- Objectives:
  - Understand general structure
  - Identify phases
  - Detect iterative pattern
  - Granularity

- Clear iterative pattern
- With an initialization phase
- All iterations are similar
- We can select a few to analyze

EuroHPC
Joint Undertaking

Select Focus of Analysis (FoA)

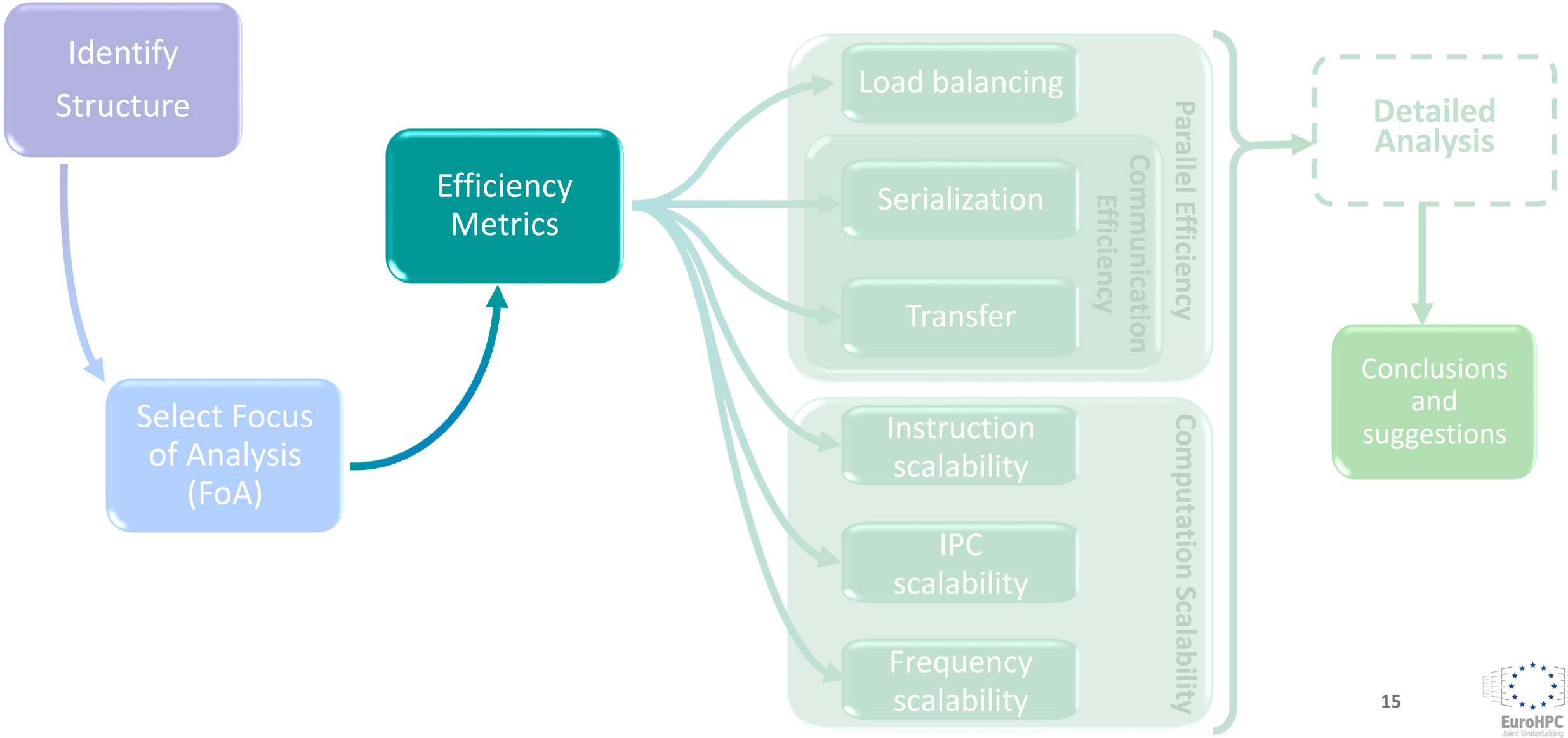
- Usually use “MPI calls” view or “Useful Duration”



Objectives:

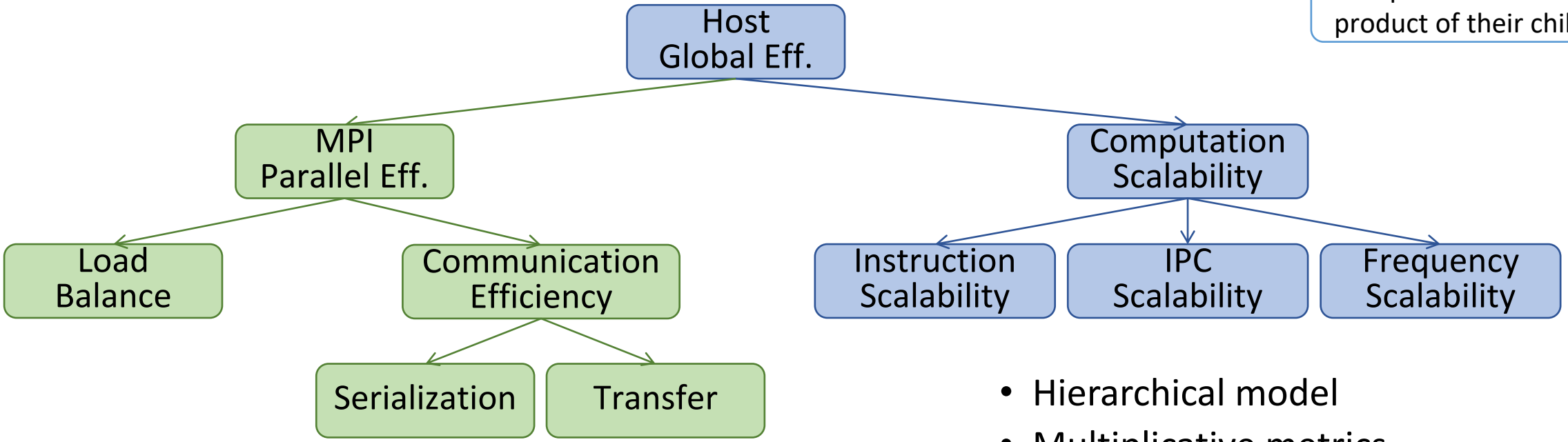
- Select the region we want to analyze
- Not a correct answer, depends on the context of the analysis
- For the same trace we may select two different FoA to perform two different studies with two different objectives.

The journey


Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

The Efficiency Metrics

All parent metrics are the product of their child metrics



- Hierarchical model
- Multiplicative metrics
- 1 to 100% scale
- Two kind of metrics:
  - Efficiency metrics
    - Absolute metrics
  - Scalability metrics
    - Relative to a base case
    - 100% for the base case

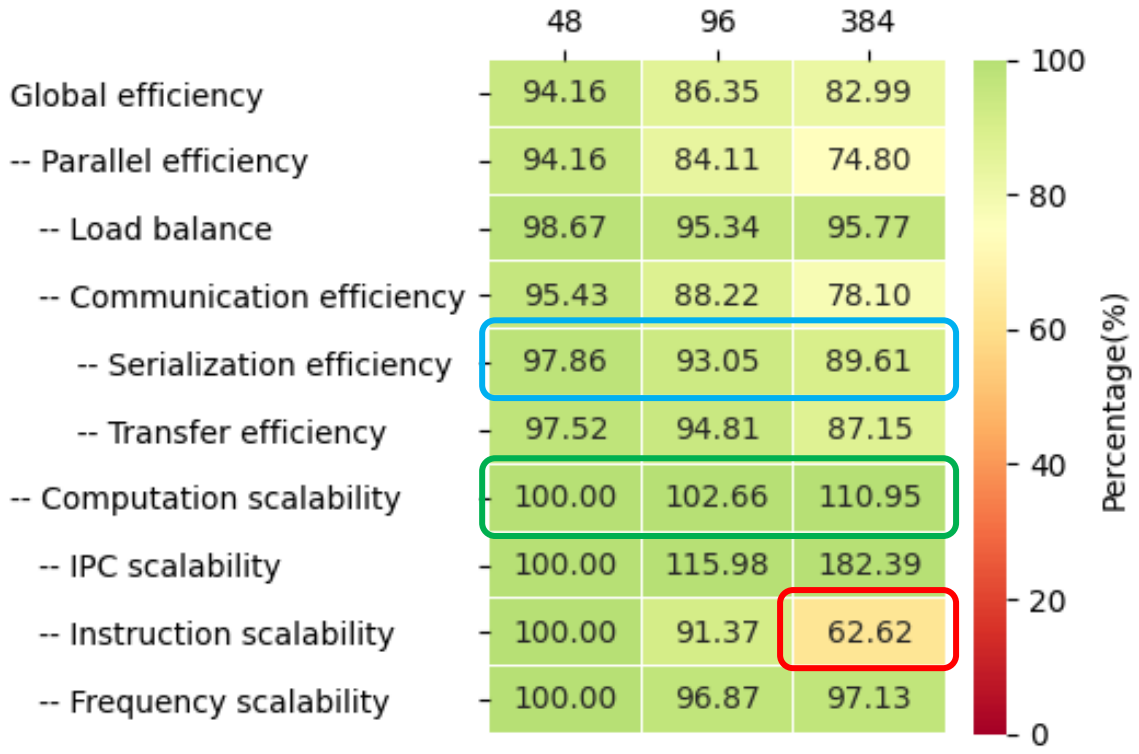
The Efficiency Metrics

- What they are NOT
  - The end of the journey
- What they are…
  - … a general mechanism to describe the fundamental concepts of parallelism
  - … a hint that tells where to look
  - … a way to quantify efficiency loss
  - … a fair comparison between different…
    - … code versions
    - … architectures
    - … core counts (scalability)
    - … applications
- For each metric we are going to see:
  - What it quantifies
  - How it is computed
    - Formula
    - Graphical
  - Interpretation
  - Where to look next

EuroHPC
Joint Undertaking

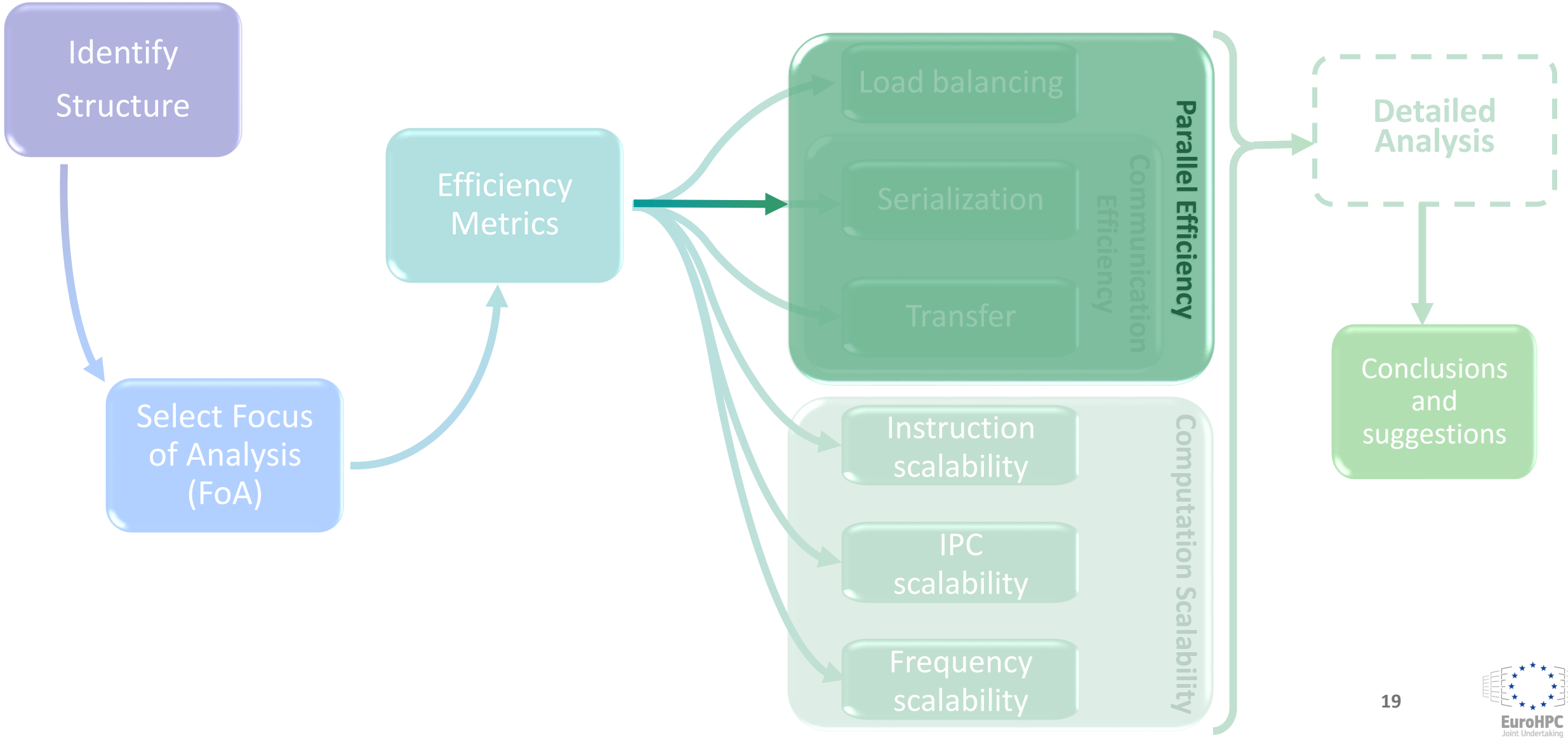
The Efficiency Metrics

- Usually shown in a table
  - Rows: Metrics
  - Columns: Different traces
- With colored cells as a heat map
- What to look for?
  - Low values
  - Trend
  - High values



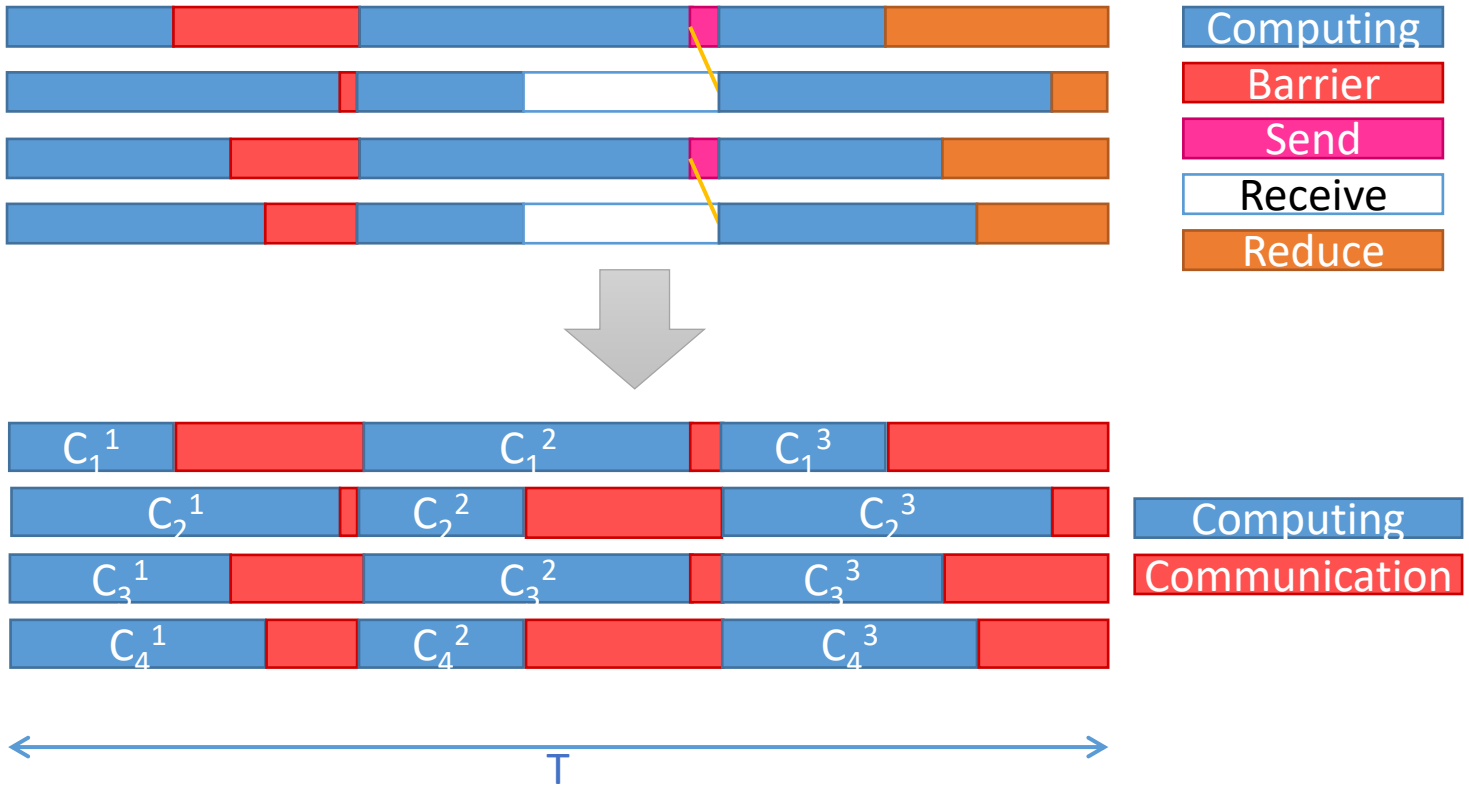
EuroHPC
Joint Undertaking

The journey


Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Some semantics first

- The state of a processes is simplified to two values:
  - Useful == Computing
  - Not useful == Otherwise



- T = Elapsed time
- P = Number of processes
- c_i = Compute time of process i
  - $c_i = \sum_{j=1}^{n} c_i^j$
- C = Total compute time
  - $C = \sum_{i=1}^{P} c_i$

EuroHPC Joint Undertaking

Parallel Efficiency

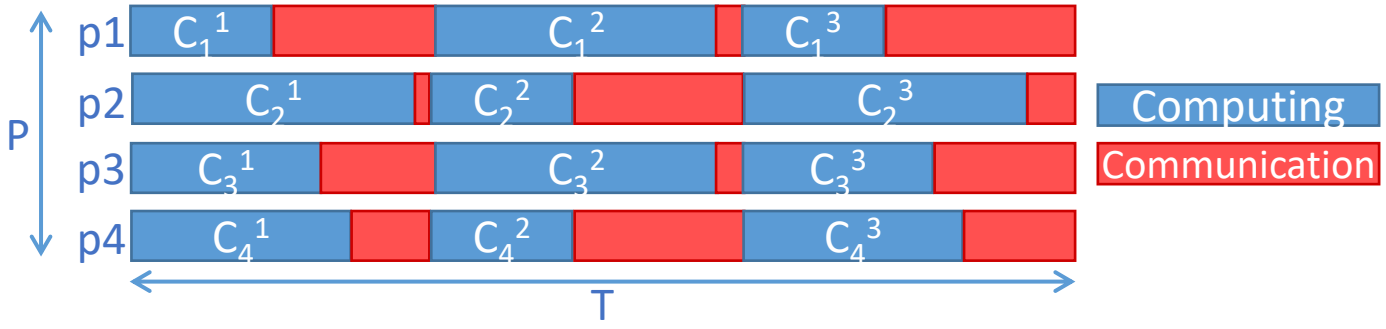
Quantifies: The extent to which all resources in the system are kept active doing useful work

How it is computed: Ratio between time used to do useful computation and consumed cpu time

Parallel Efficiency = / (+)

How it is computed:

$$Parallel\ Efficiency = \frac{\sum_{i=1}^{P} c_i}{P * T}$$



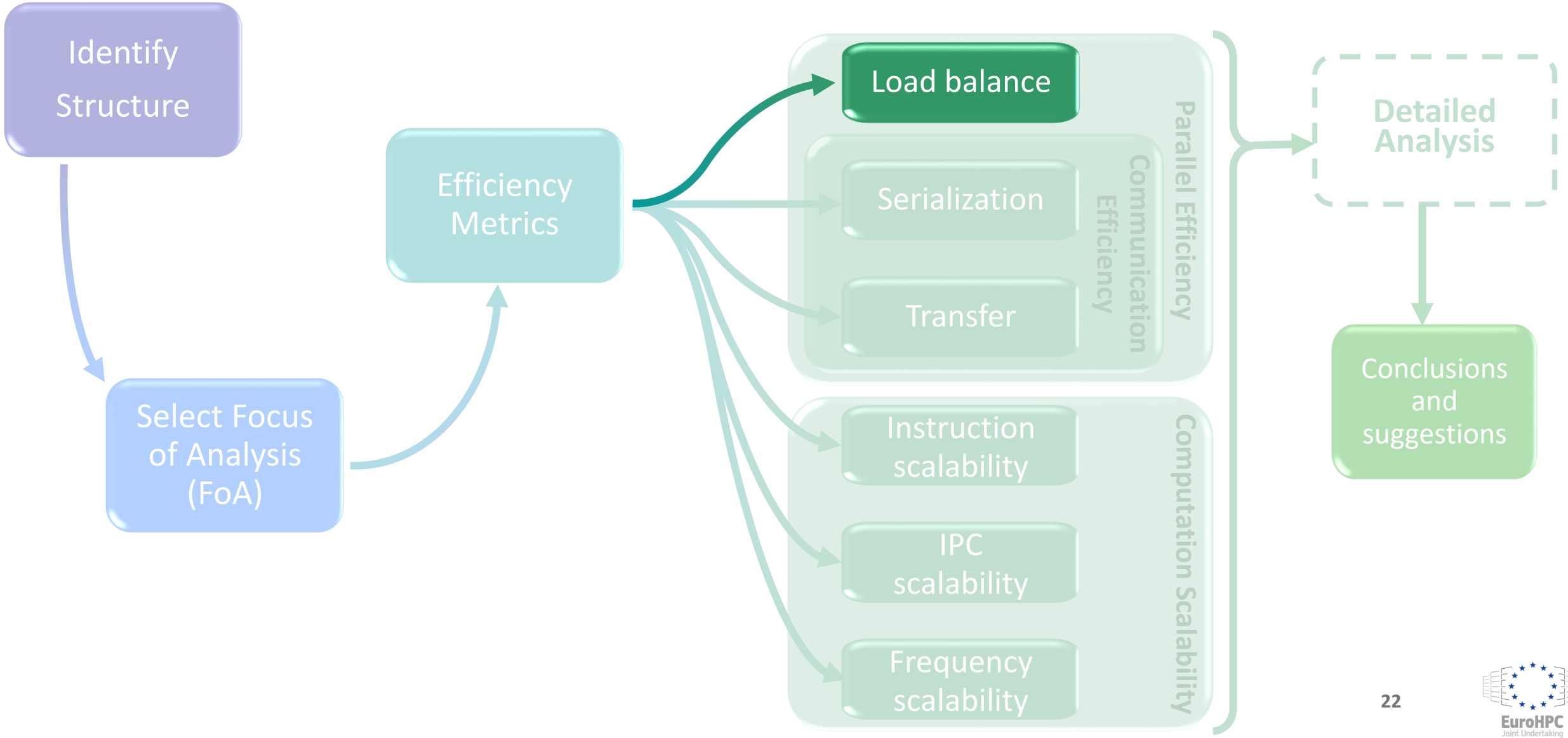
Interpretation: A low value indicates that a low fraction of the time consumed is used to do useful computation.

Where to look next?

- Look at its child metrics

EuroHPC Joint Undertaking

The journey


Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balance
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Load Balance

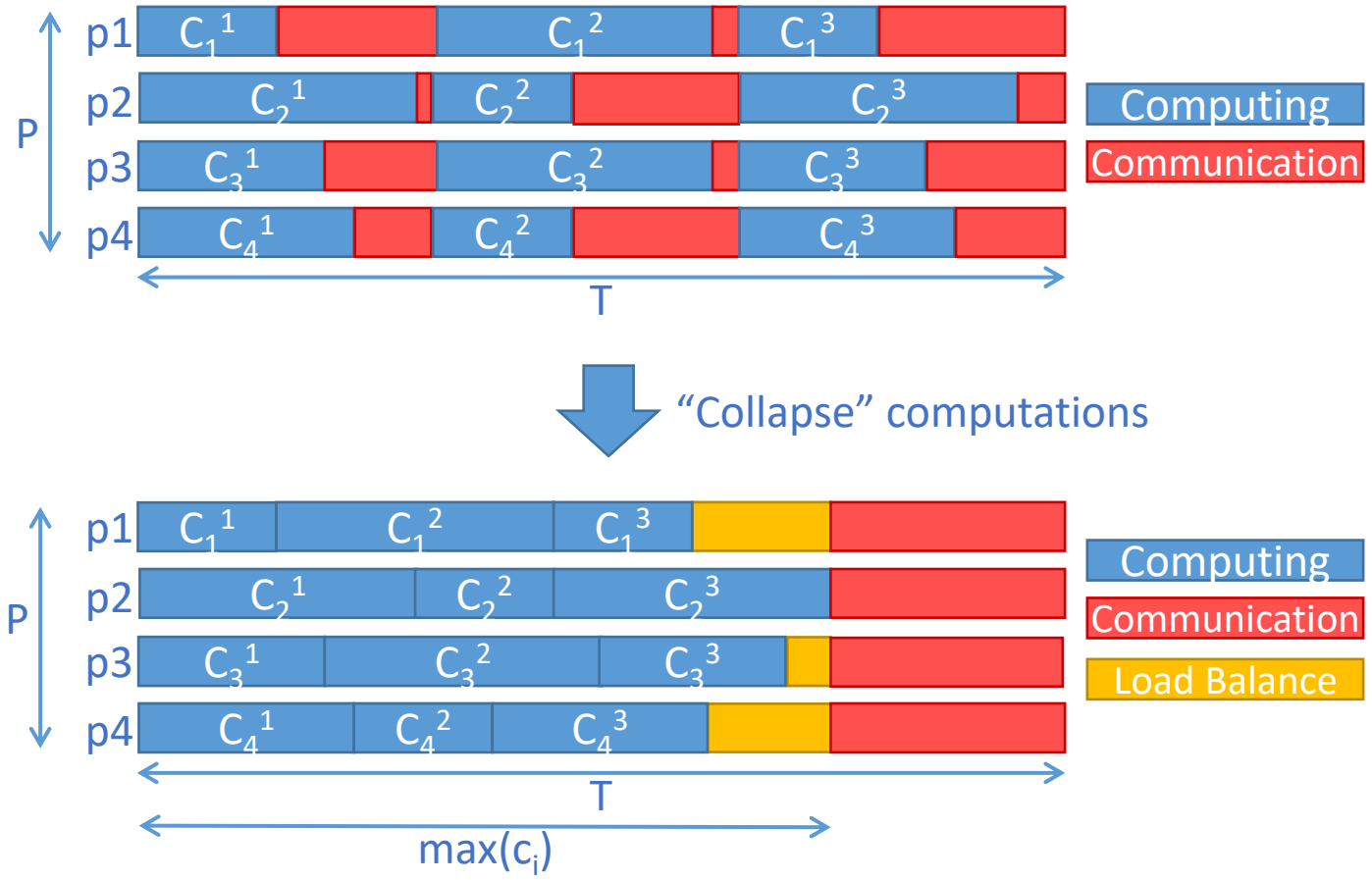
Quantifies: The efficiency loss due to the global distribution of work among processes.

How it is computed: Ratio between time used to do useful computation and the useful computation of the most loaded process multiplied by the number of processes

Load Balance = ■ / (■ + ■)

How it is computed:

$$Load\ Balance = \frac{\sum_{i=1}^{P} C_i}{max^{P}{}_{i=1}(c_i) * P}$$

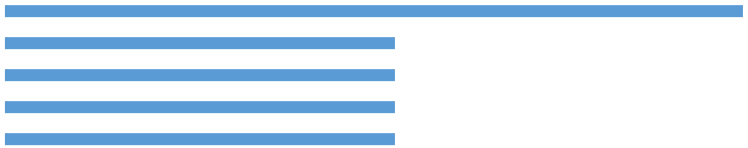


EuroHPC Joint Undertaking

Load Balance

Interpretation: A low value in this metric indicates that more highly loaded processes keep other processes idle for a significant amount of time.

- A single highly loaded process will make this metric report a low value



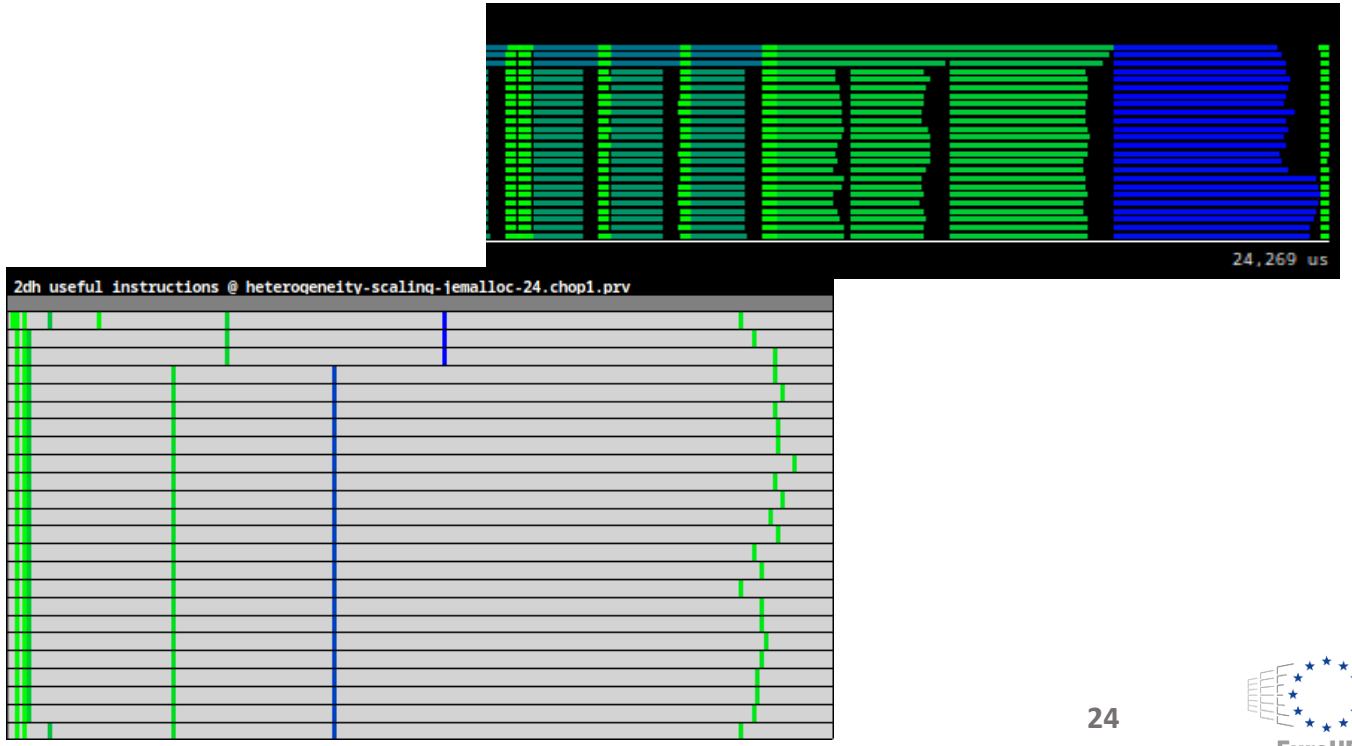
$$LB = \frac{2+1+1+1+1}{2*5} = \frac{6}{10} = 0.6$$

- A single low loaded process won't have an effect on this metric



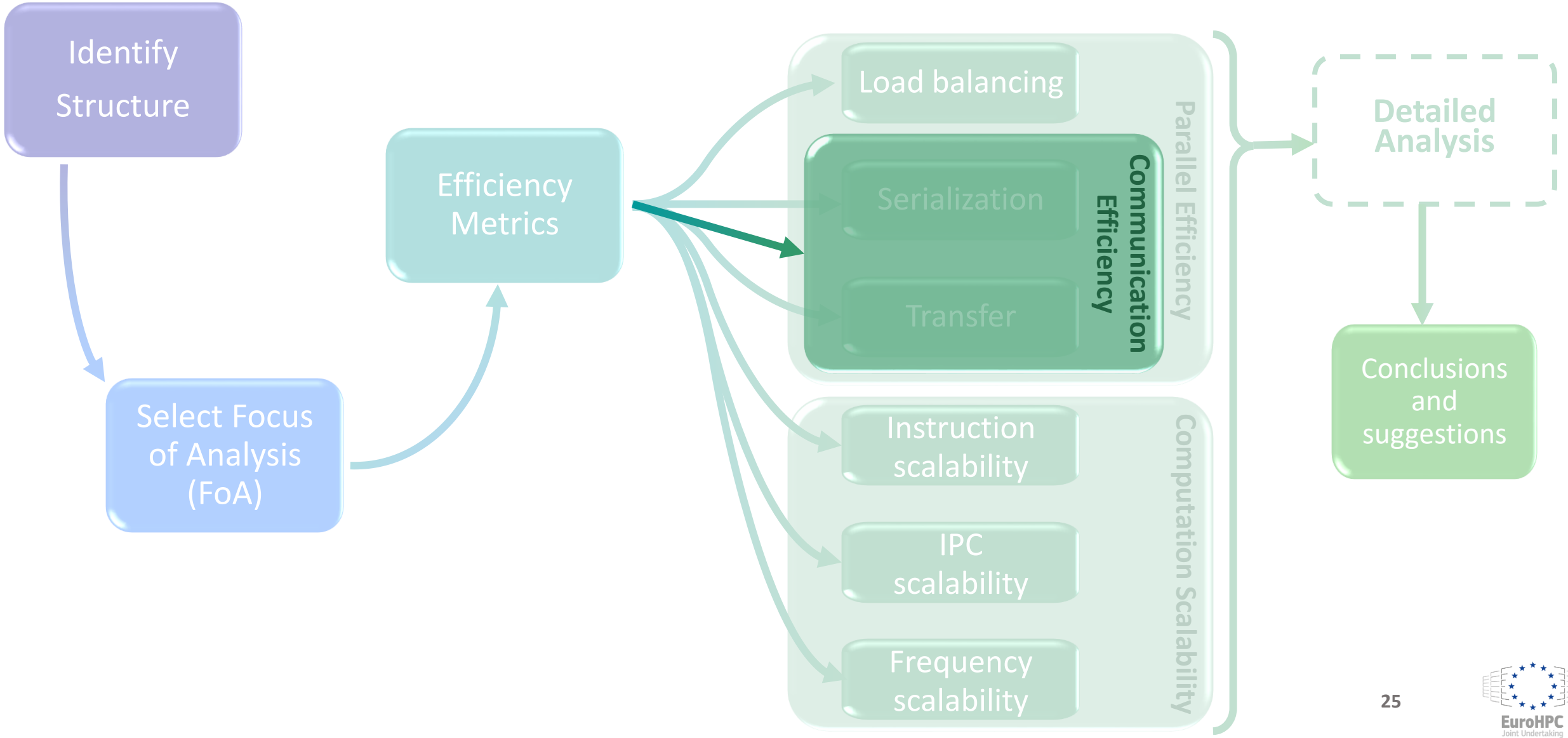
$$LB = \frac{2+2+2+2+1}{2*5} = \frac{9}{10} = 0.9$$

- What to look next?
  - We try to understand the cause of the Load Imbalance
  - In general can have 3 sources
    - Number of instructions → Histogram of useful instructions
    - IPC → Histogram of IPC
    - Frequency → Histogram of cycles per us



EuroHPC
Joint Undertaking

The journey


Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Communication Efficiency
Serialization
Transfer
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Communication Efficiency

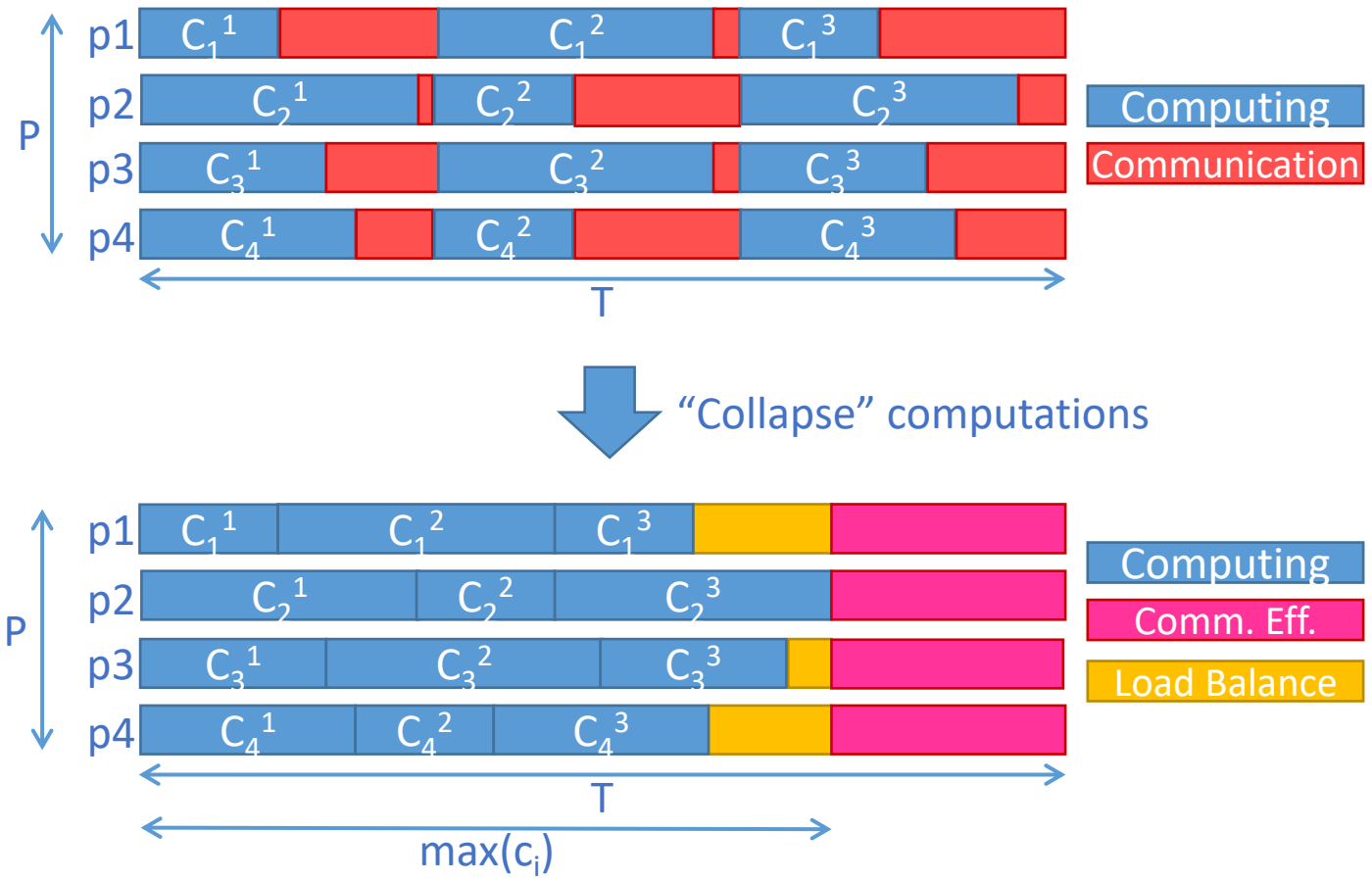
Quantifies: The efficiency loss due to the communication of data. Be it due to synchronizations between processes or to the overhead introduced by the communication itself. Excluding time loss due to global load imbalance

How it is computed: Ratio between the useful computation time of the most loaded processes and the total elapsed time

Communication Eff.= (+) / (+ +)

How it is computed:

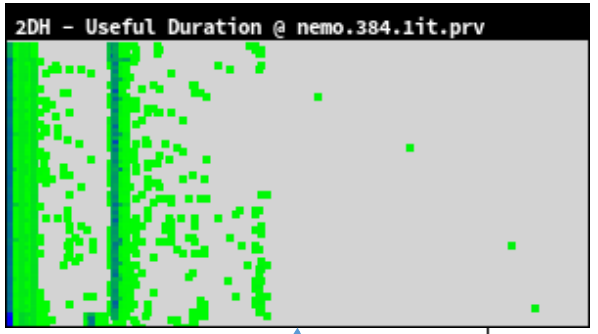
$$Communication\ Eff. = \frac{max^{P}{}_{i=1}(c_i)}{T}$$

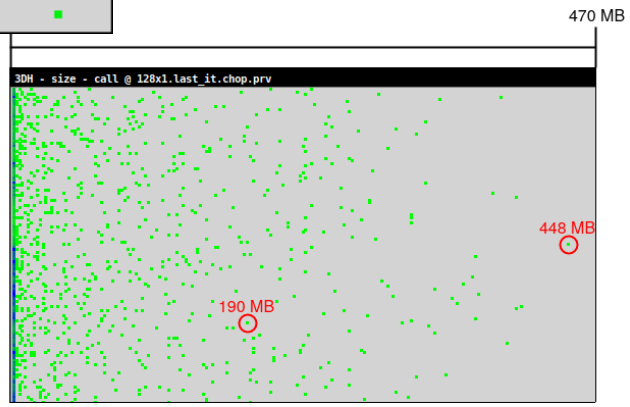


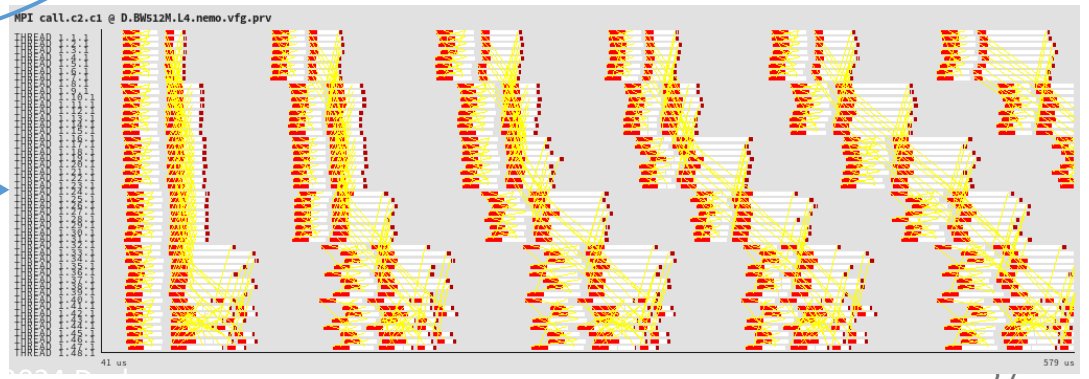
Communication Efficiency

Interpretation: A low value in this metric indicates that the interaction between processes is impacting the performance.

- This metric can report good values in codes where a profile reports significant time in MPI and that would be reported by a bad load balance efficiency
- What to look next?
- If possible child metrics
- If not…
  - How many MPI calls are made?
    - Histogram of MPI calls
  - How often? Granularity of computations
    - Useful duration
  - How much data is sent?
    - Bytes sent per MPI call
  - Which are the semantics of the communication?
    - Chains of dependences?
      - MPI calls

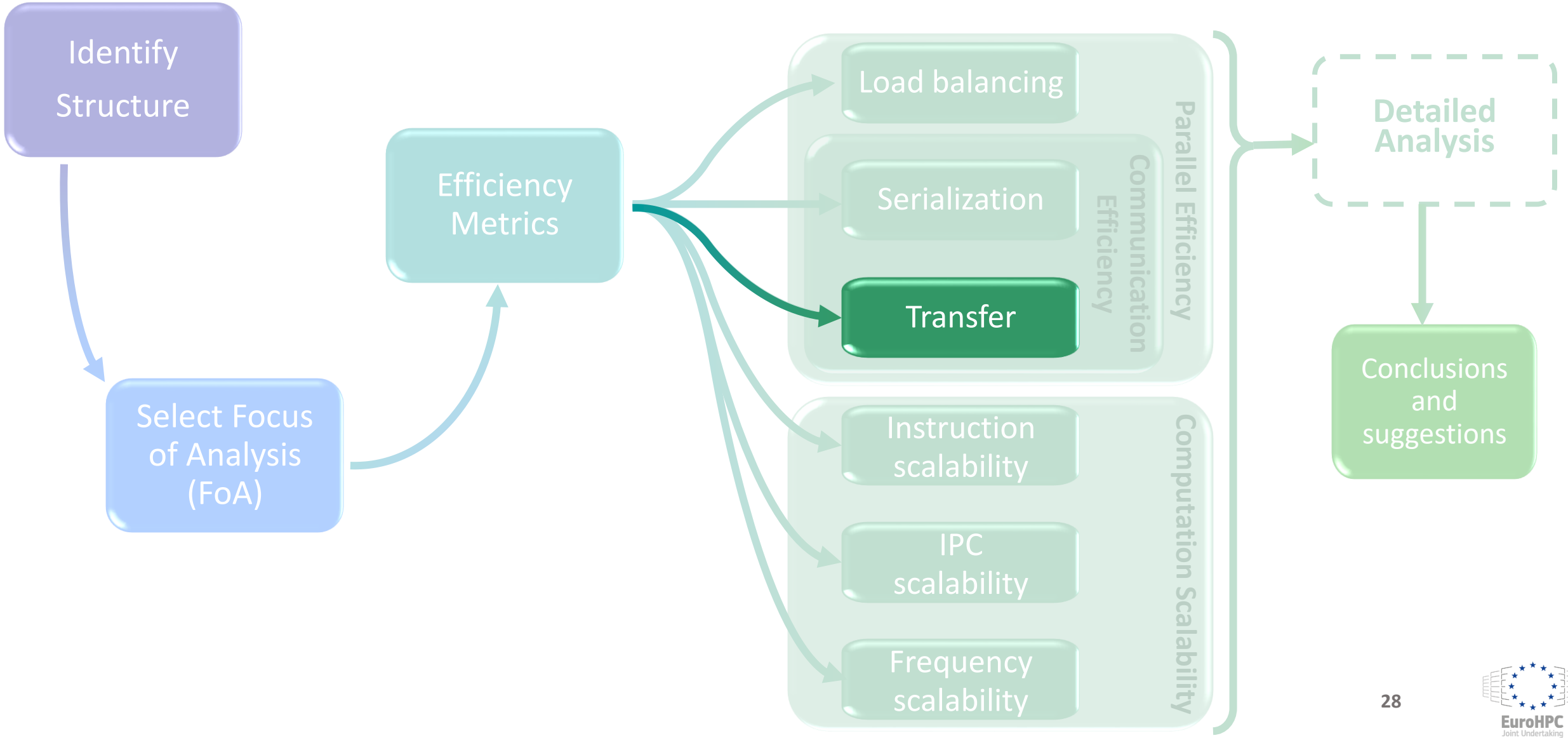






EuroHPC Joint Undertaking

The journey


Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Transfer Efficiency

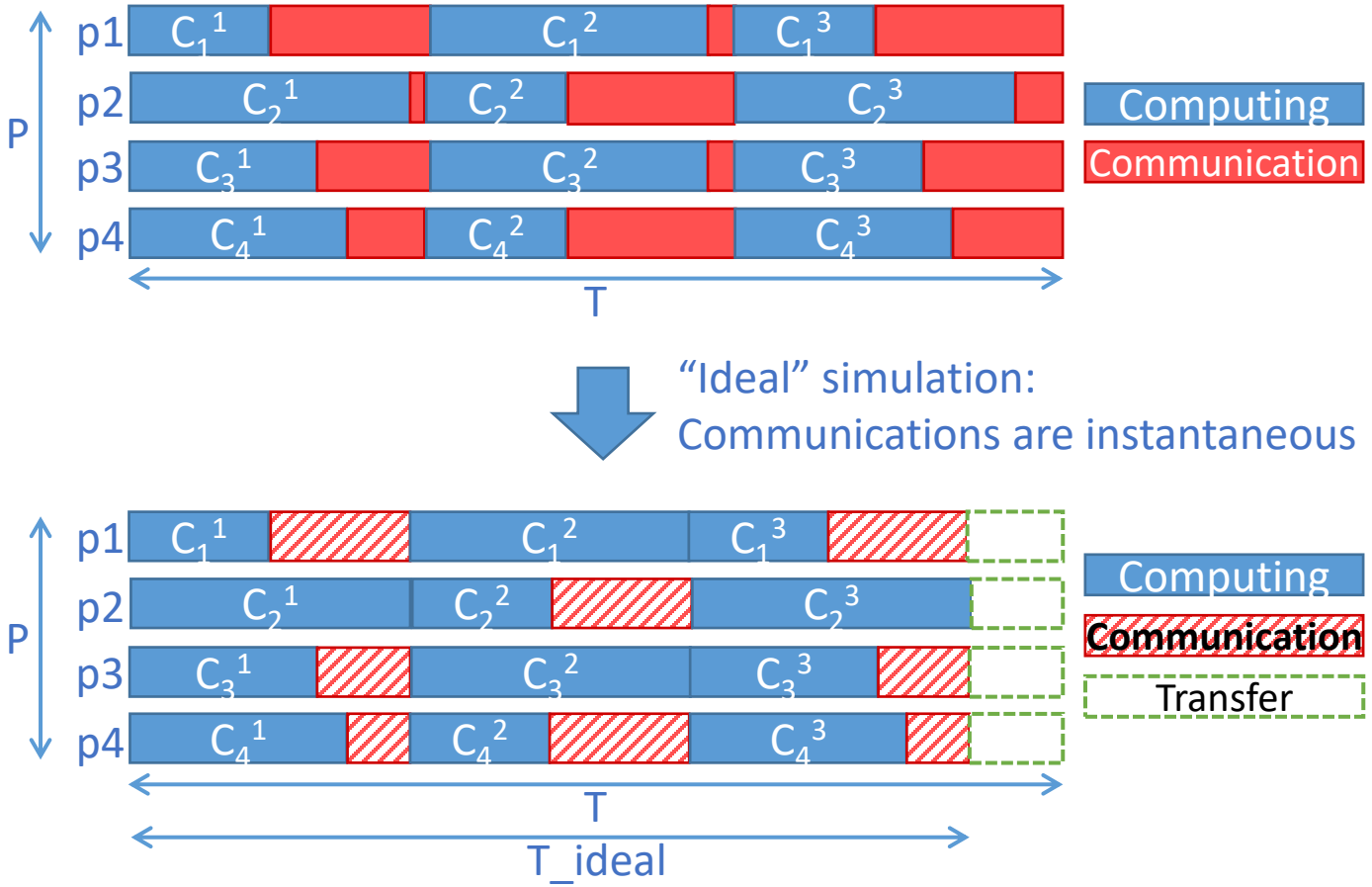
Quantifies: Efficiency loss related to the non instantaneous nature of communication mechanisms. Includes time to transmit the data over the physical channel and the overhead in the libraries.

How it is computed: Ratio between elapsed time in the ideal simulation and the elapsed time in the real execution

Transfer Eff.= (+) / (+ +)

How it is computed:

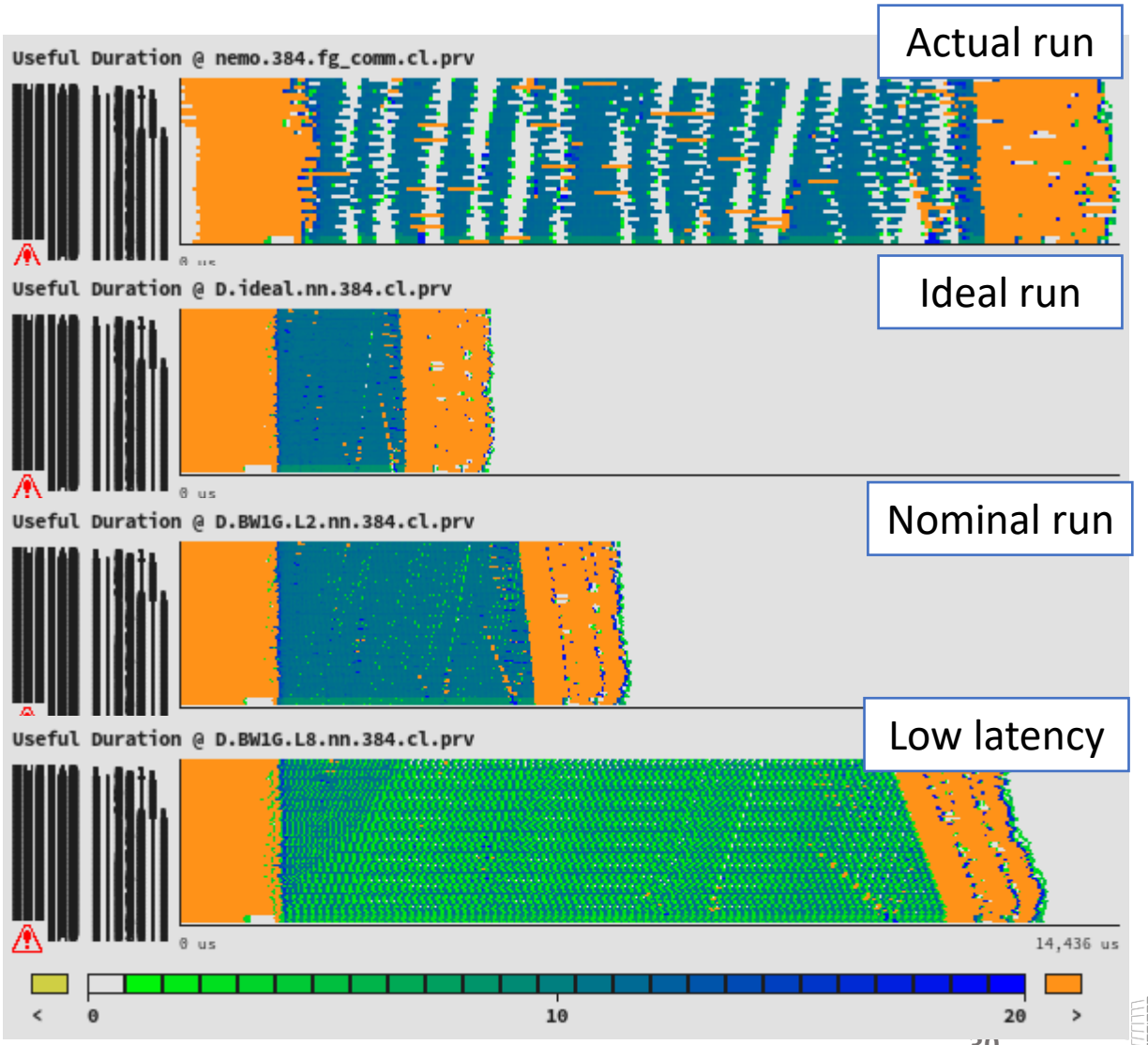
$$Transfer\ Eff. = \frac{T_{ideal}}{T}$$



Transfer Efficiency

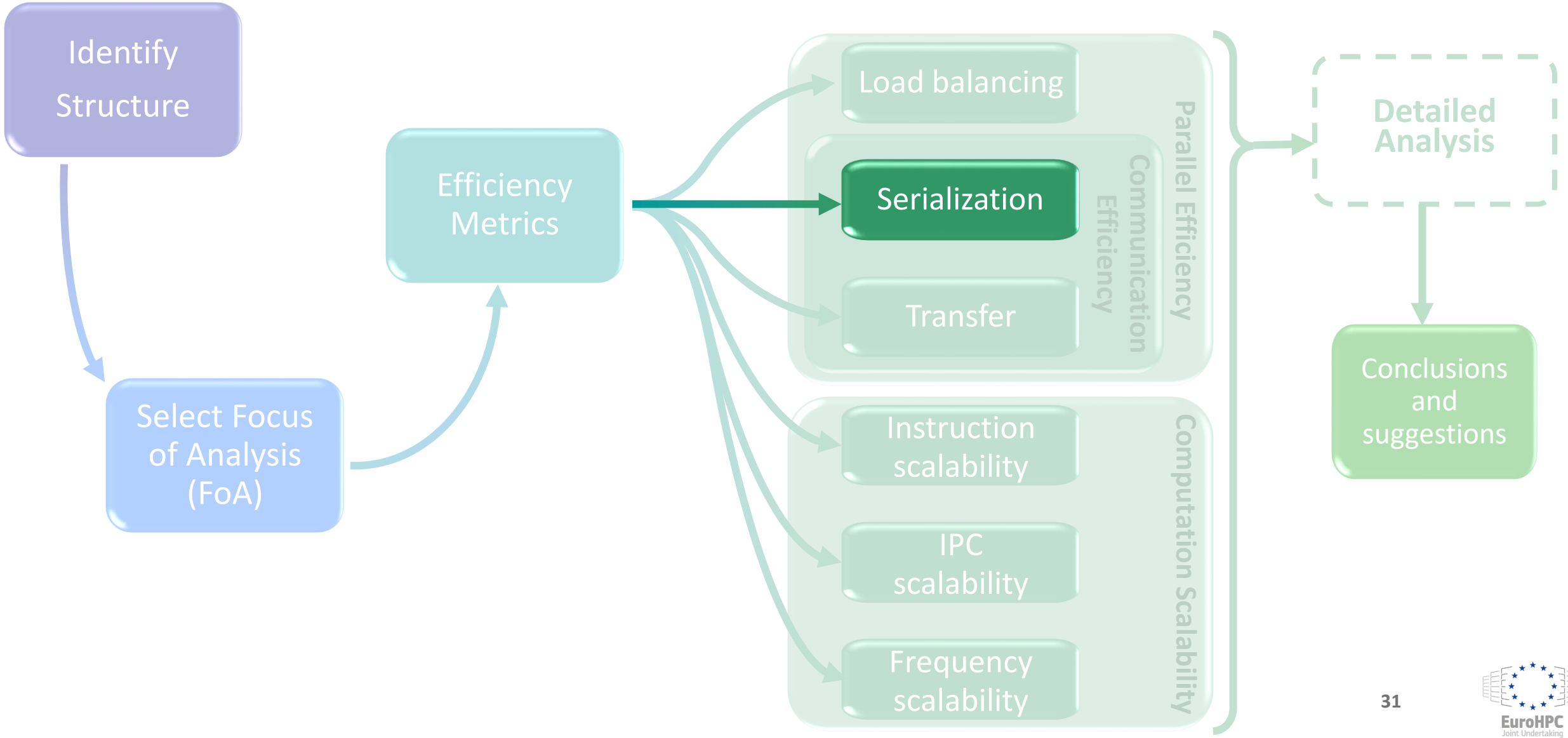
Interpretation: Low values indicate that the execution is suffering from a high overhead of the runtime or a poor latency or bandwidth of the network.

- What to look next?
  - Determine if the transfer problem is Bandwidth or Latency
    - Note that in Latency we include the overhead of the communication library
    - Use Simulations of Dimemas
      - Different BW and latency



EuroHPC Joint Undertaking

The journey


Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Serialization Efficiency

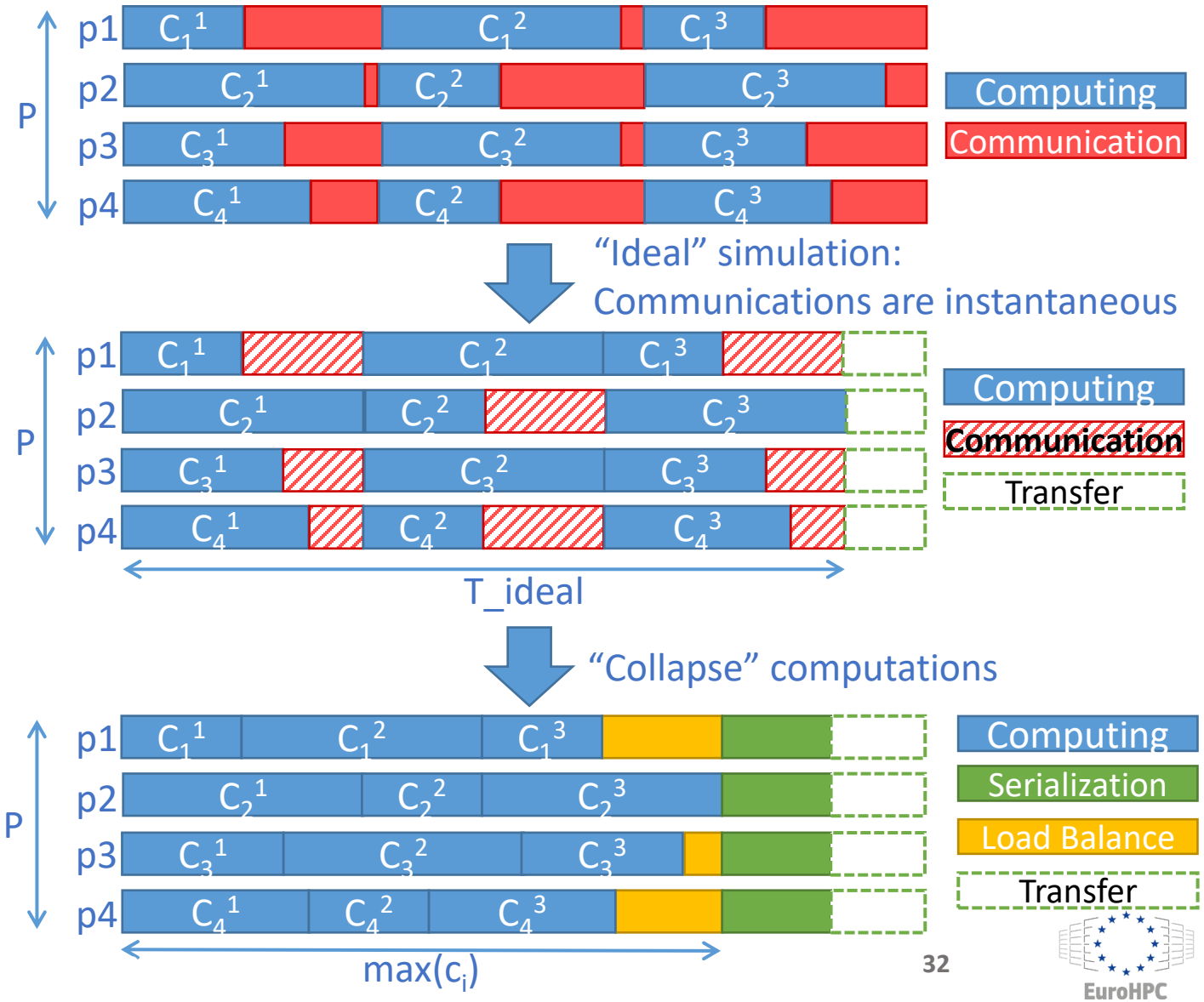
Quantifies: Inefficiencies caused by circular dependences or non-uniform imbalances

How it is computed: Ratio between useful computation of most loaded process and elapsed time in the ideal simulation

Serialization Eff.= (+) / (+)

How it is computed:

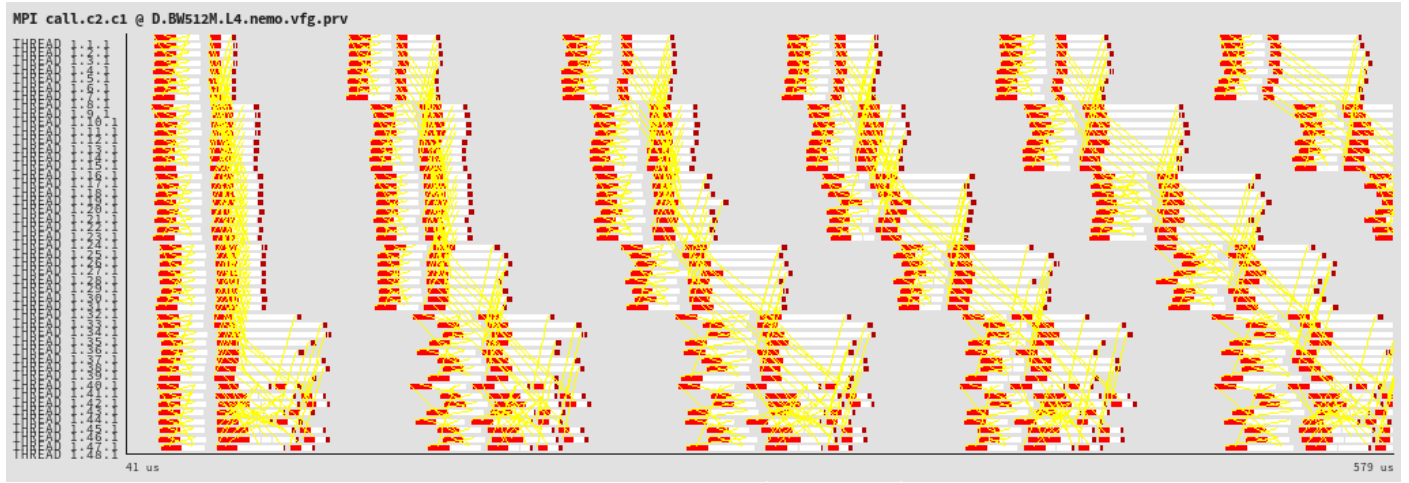
$$Serialization\ Eff. = \frac{max^{P}{}_{i=1}(c_i)}{T_{ideal}}$$



Serialization Efficiency

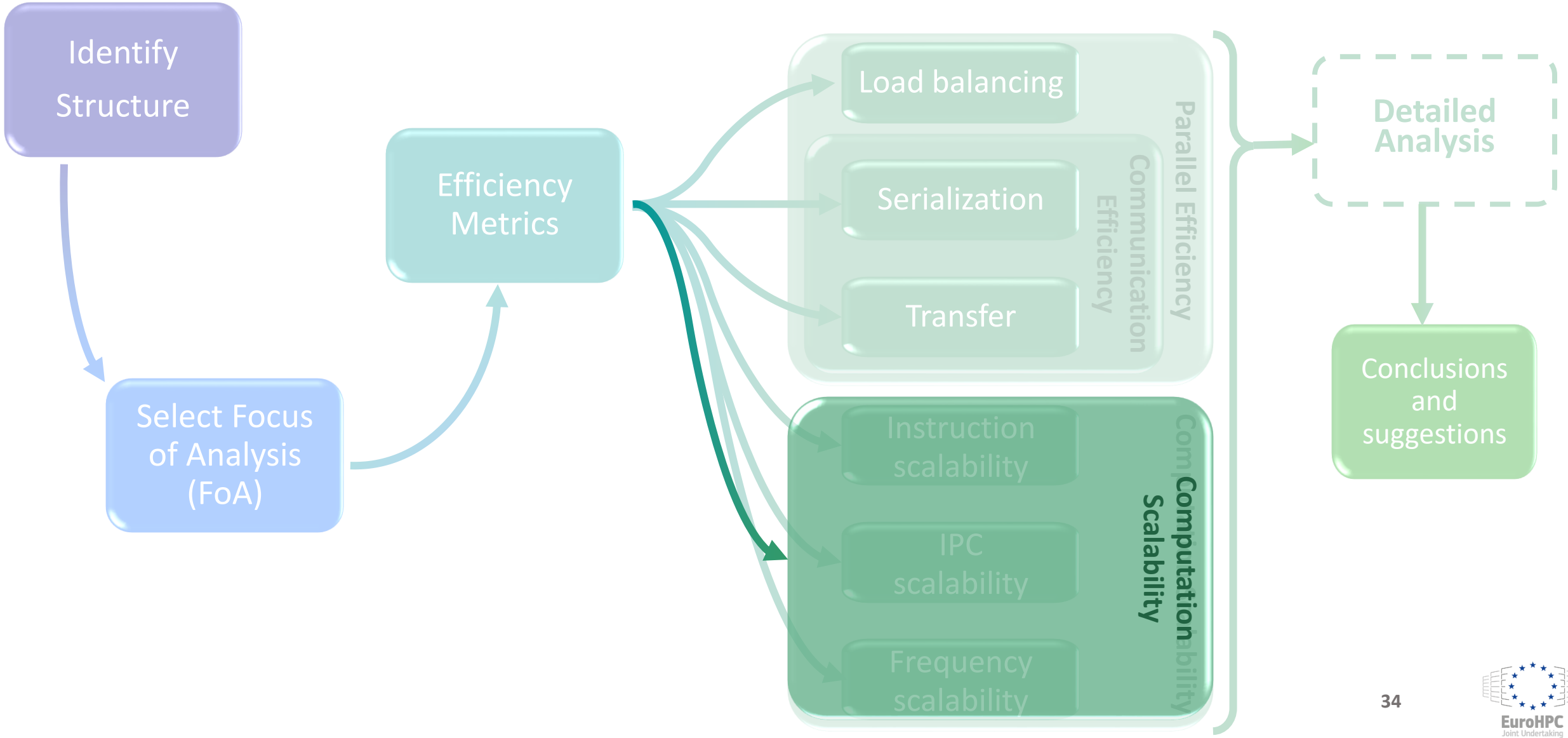
Interpretation: A low value indicates the existence of circular dependences. They can be caused by actual algorithmic serialization, irregularities in the load of processes during the execution or noise.

- What to look next?
  - Discard noise
    - Cycles per us
  - Try to identify causes for circular waits.
  - Understand semantics of the communication
    - MPI Calls



EuroHPC
Joint Undertaking

The journey


Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Computation Scalability

Quantifies: How the time spent computing scales with respect to the reference case.

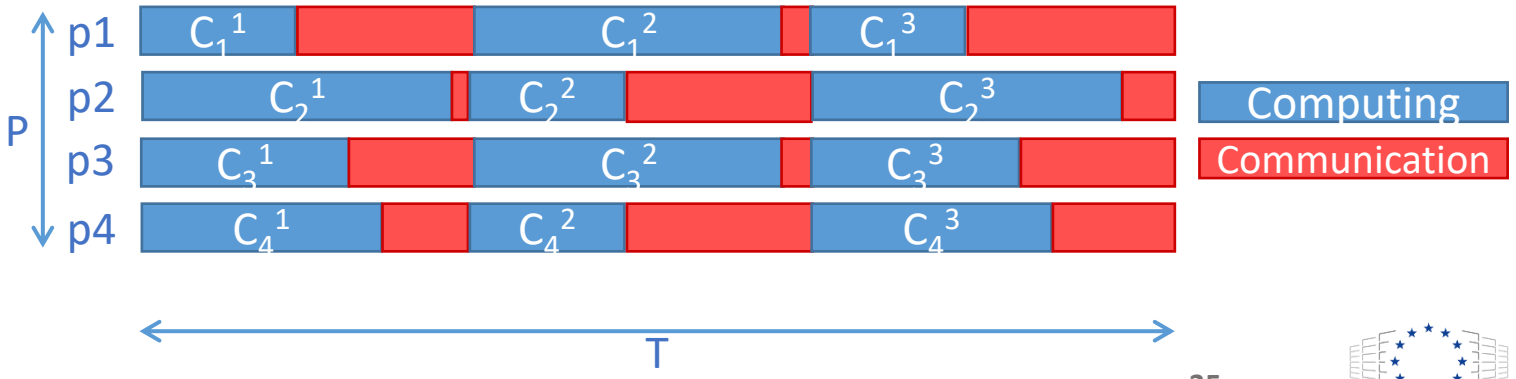
How it is computed: Ratio between time spent on useful computation in the reference case and the time spent on useful computation on current run.

How it is computed:

$$Computation\ Sc. = \frac{C_{ref}}{C_{current}}$$

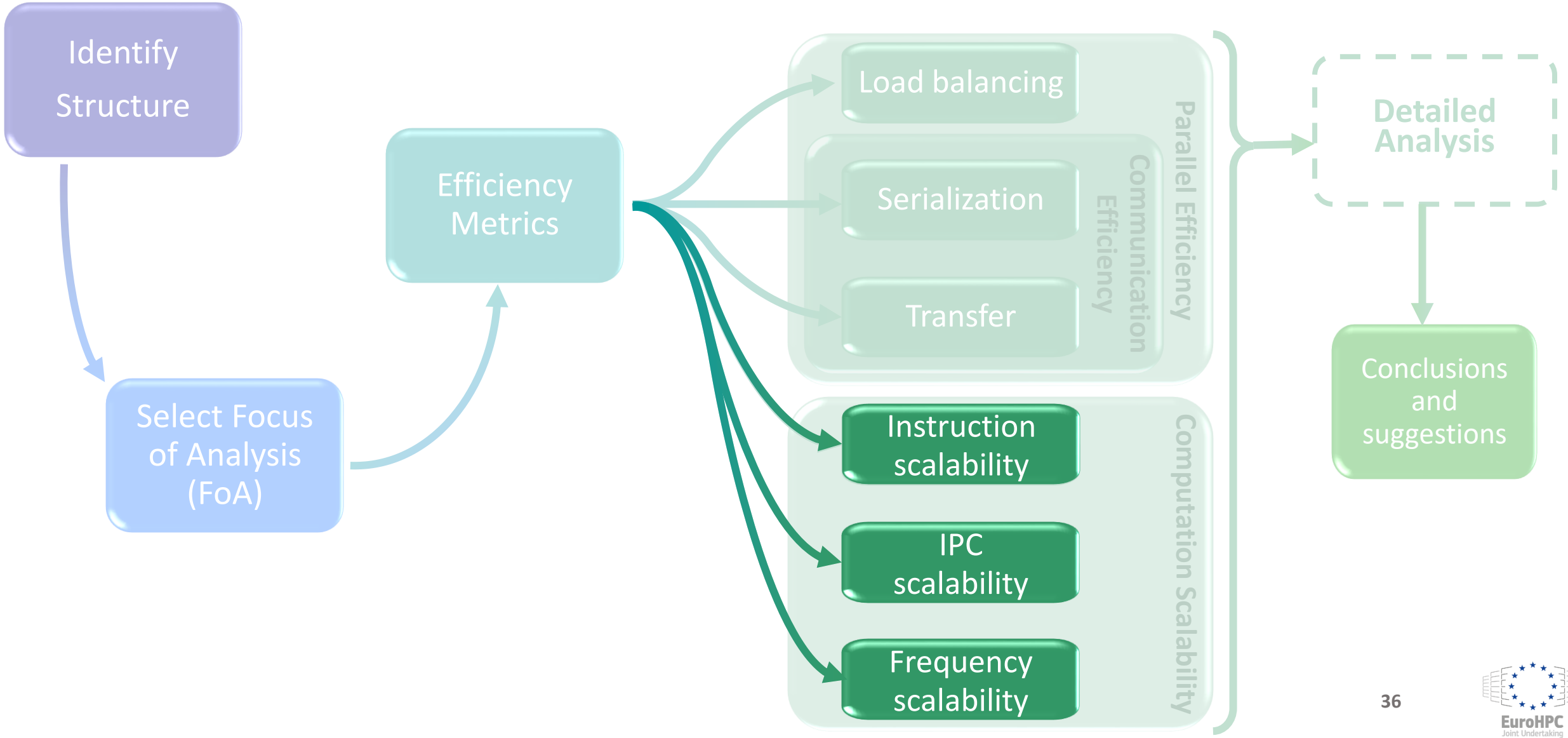
Interpretation: A low value indicates time grows per core count. Ideally the total compute time to solve a problem should be constant, independent of core count (in strong scaling).

- Relative metric, based on a reference case.
  - C_{ref} = Useful Computation of the reference run
  - $C_{current}$ = Useful computation current run
- 100% for the reference case
- Can be weak or strong scaling
  - Strong scaling for all the formulas presented
    - In strong scaling we assume total compute time and total number of instructions should remain constant as we increase the number of processes
  - In an analogous way weak scaling can be computed
- High level metric composed by three child metrics based on:
  - T = # instr. / (IPC * freq)



EuroHPC Joint Undertaking

The journey


Identify Structure
Select Focus of Analysis (FoA)
Efficiency Metrics
Load balancing
Serialization
Transfer
Communication Efficiency
Parallel Efficiency
Instruction scalability
IPC scalability
Frequency scalability
Computation Scalability
Detailed Analysis
Conclusions and suggestions

Instructions, IPC, Frequency Scalability



Quantifies: How the number of instructions scales with respect to the reference case.

How it is computed:

$$Instructions\ Sc. = \frac{I_{ref}}{I_{current}}$$

Interpretation: A value less than 100 indicates that the total number of instructions to solve the problem grows with core count, which ideally should not be the case.
May be caused by code replication.

Quantifies: How the IPC scales with respect to the reference case.

How it is computed:

$$IPC\ Sc. = \frac{IPC_{current}}{IPC_{ref}}$$

Interpretation: A value less than 100 indicates that the IPC for the specific core count is worse that of the reference case. May be caused by different locality, contention on hardware resources, ...

A value above 100 indicates a higher IPC this can be produced by cache effects for example.

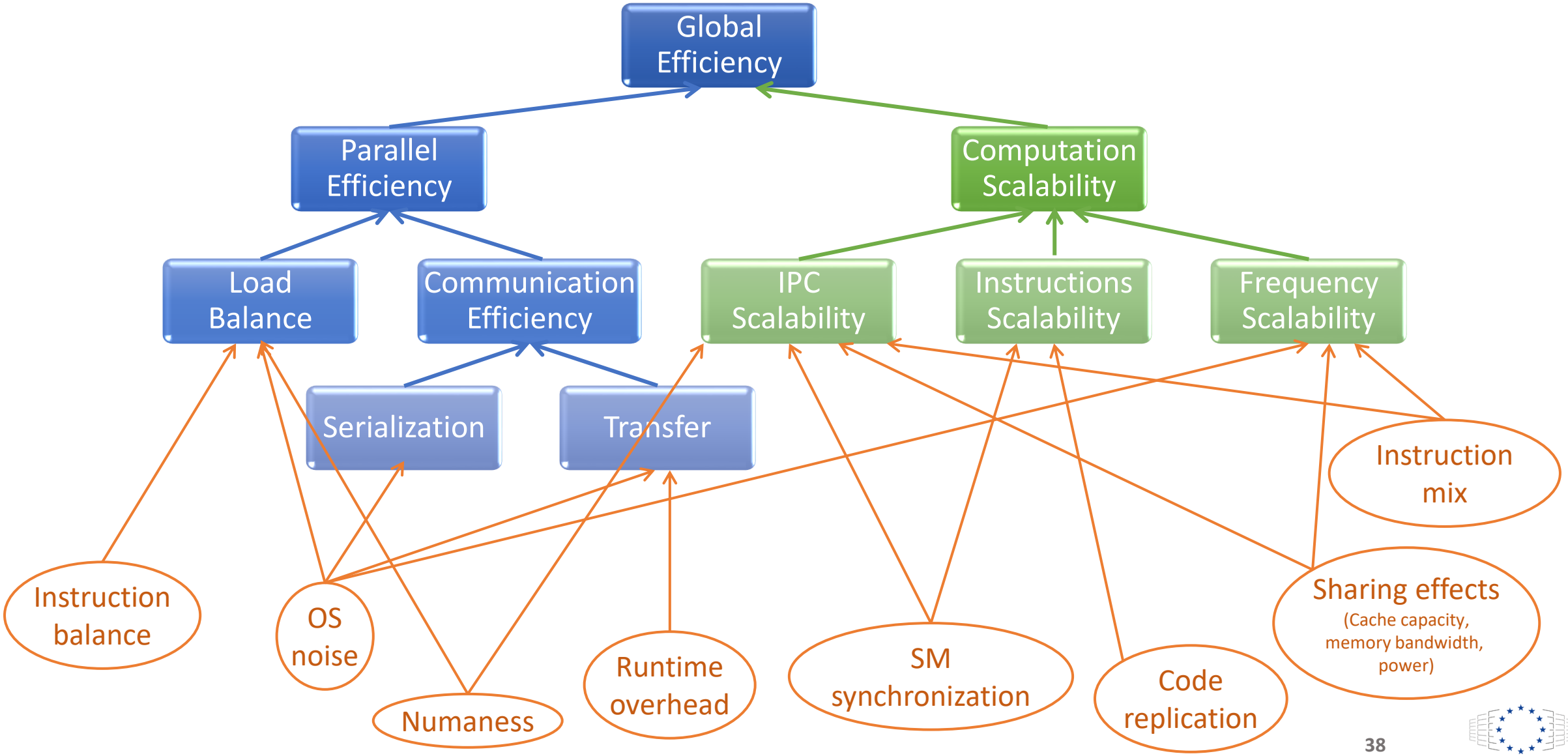
Quantifies: How the frequency scales with respect to the reference case.

How it is computed:

$$Frequency\ Sc. = \frac{Freq_{current}}{Freq_{ref}}$$

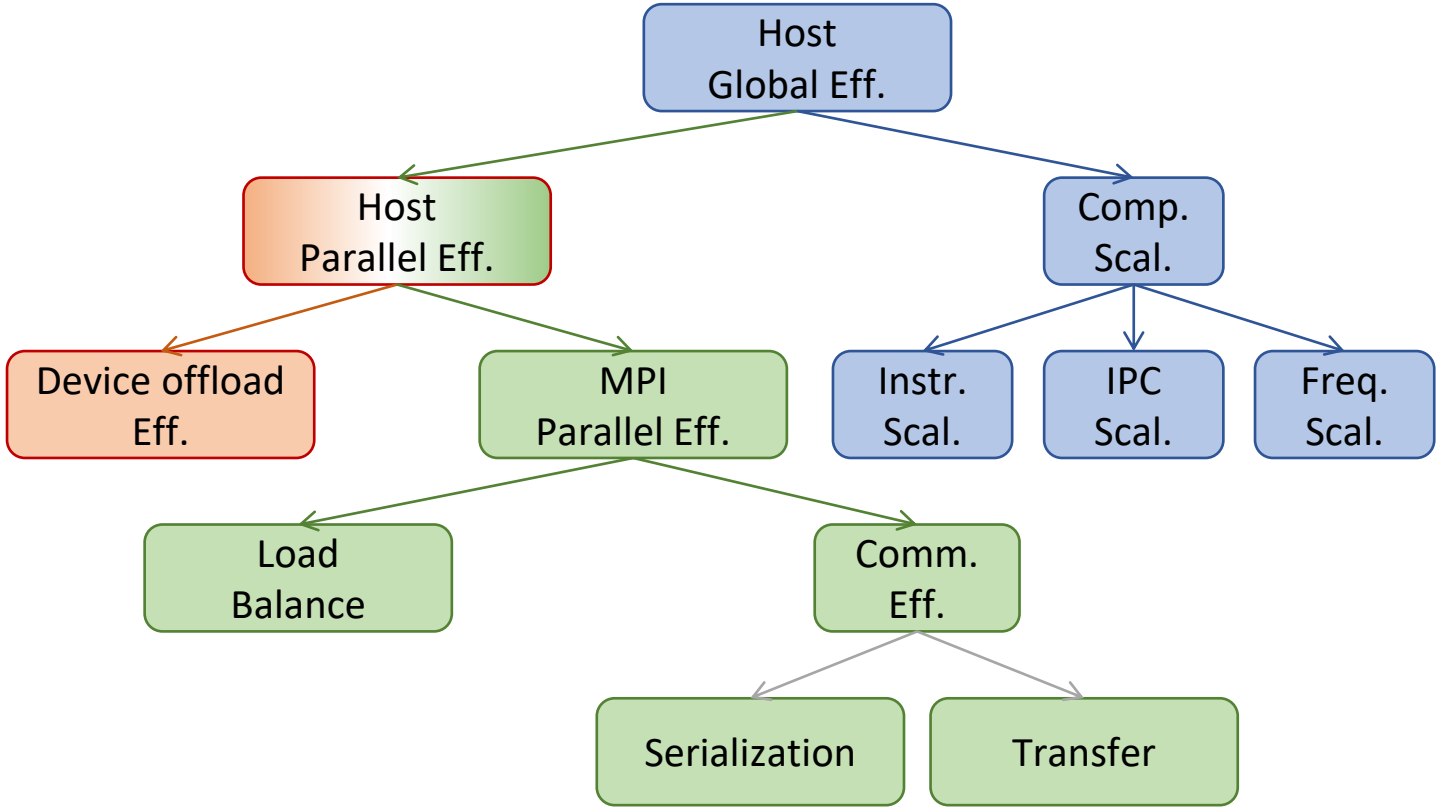
Interpretation: A value less than 100 indicates that the frequency is lower than the reference frequency. This may be caused by "preemptions", power management measures, ...

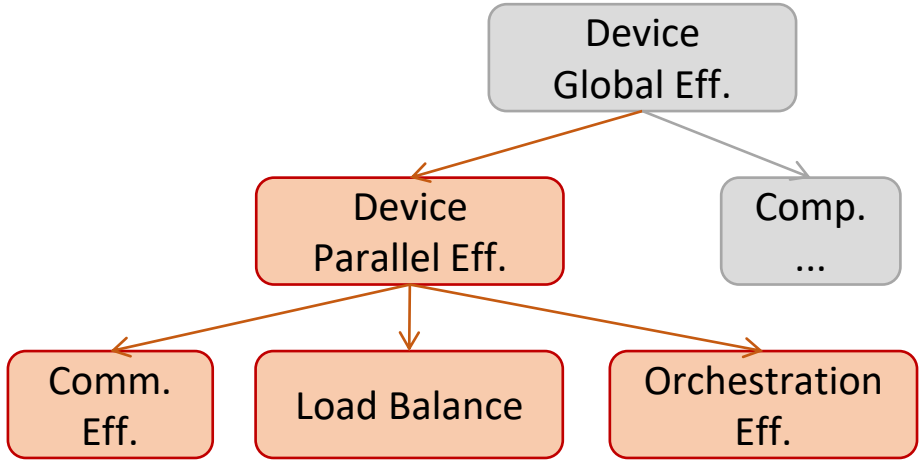
Factors and causes



GPU metrics

All parent metrics are the product of their child metrics





Efficiency metrics

Scalability metrics

GPU metrics

EuroHPC
Joint Undertaking



Selected performance assessments

HORIZON-EUROHPC-JU-2023-COE

EuroHPC Joint Undertaking

1 January 2024– 31 December 2026

Grant Agreement No 101143931

SPECFEM3D (Leonardo-B)

https://co-design.pop-coe.eu/reports/POP3-AR-002-SPECFEM3D.html

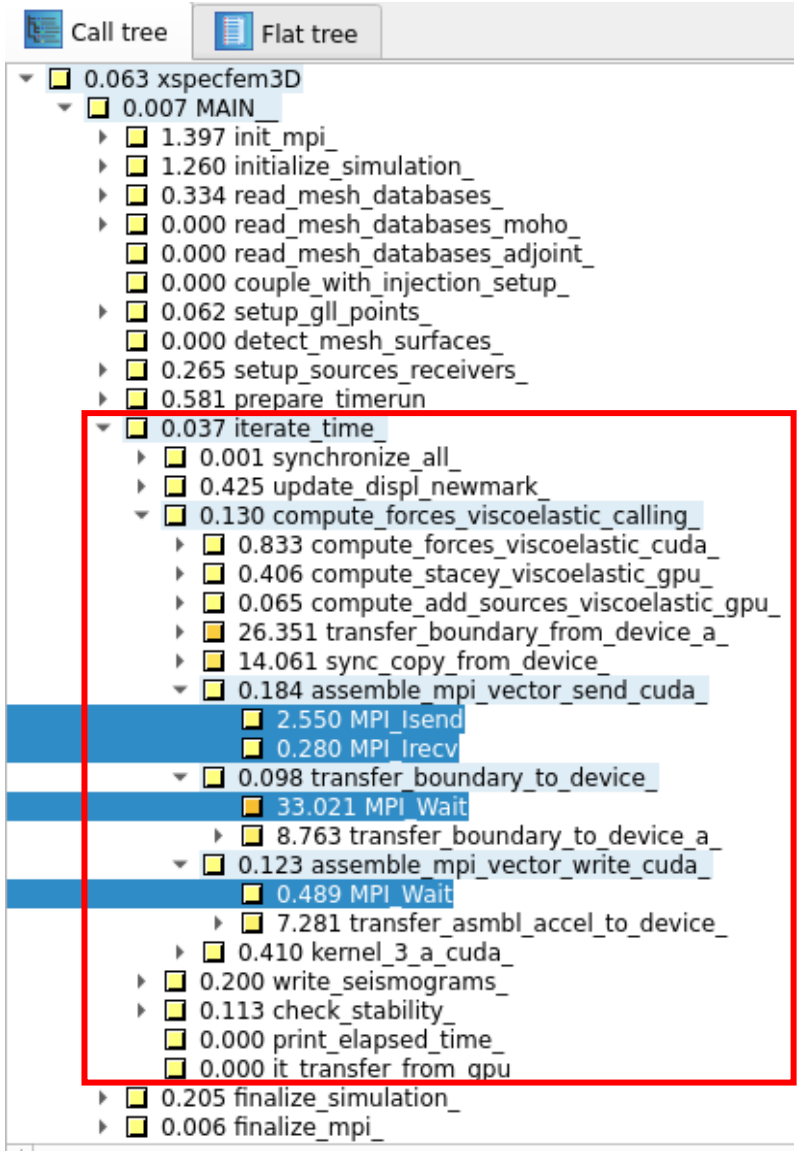
DOI 10.5281/zenodo.13643996

- SPECFEM3D
  - Software package for simulation of seismic wave propagation based on the spectral-element method
  - Assessment for HPC CoE for Exascale in Solid Earth (ChEESE)
  - Version 4.0.0 (release)
- Fortran90 (and some C) parallelized with MPI & CUDA: one MPI process per GPU
  - Intel oneAPI 2023.0.0 compilers and 2021.7.1 MPI libraries (not GPU-Aware)
- Testcase: 1 source in elastic domain; 4 seismic receiver stations
  - 48000 solver timesteps with intermediate writing disabled
  - weak scaling (22x 128x128 = 360,448 elastic elements per rank)
  - strong scaling (22x 1024x1024 = 23,068,672 elastic elements total)
- Executed on *Leonardo-Booster* Atos Bull Sequana XH21355 (CINECA)
  - 2345 compute nodes with 32-core Intel Xeon Platinum 8358 ('IceLake') CPUs @ 2.6 GHz and quad Nvidia A100 ('Ampere') GPUs [64GB]
  - Nvidia Mellanox HDR DragonFly++ interconnection network
- Measurements with Scalasca/2.6.1 using Score-P/8.3
- Focus of analysis (FOA): xspecfem3D/*iterate_time*

ChEESE

EuroHPC Joint Undertaking

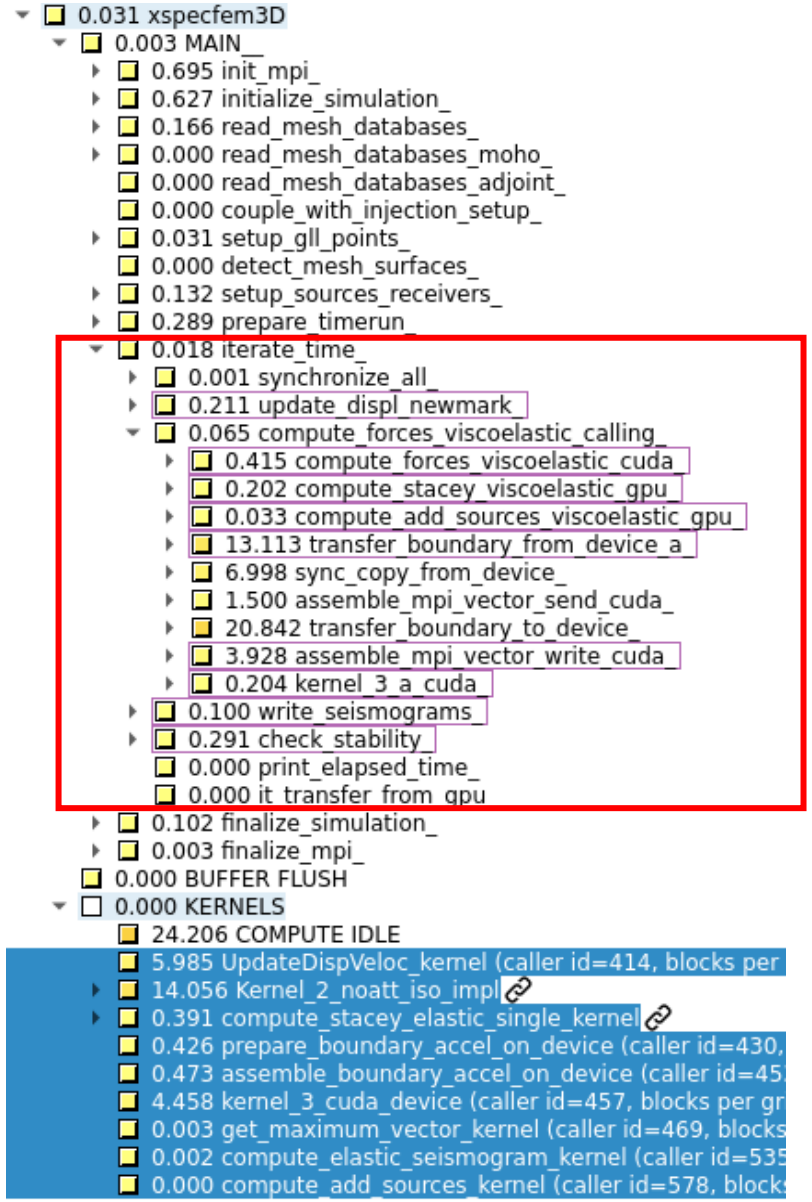
Execution call-tree & Focus of Analysis



- Structure
  - setup/initialise (amortised in full run)
    - read (w/o MPI), MPI_Bcast, MPI_Reduce, etc
  - solver (*iterate_time*)
    - 48000 timesteps
      - non-blocking point-to-point communication for boundary exchange with 2D neighbours
      - data transfer to/from associated GPU device and corresponding stream synchronization
    - summary output every 500 steps
      - collective MPI_Reduce
  - *write_seismograms* executed only once
- Focus of Analysis selected for assessment: *iterate_time*

EuroHPC
Joint Undertaking

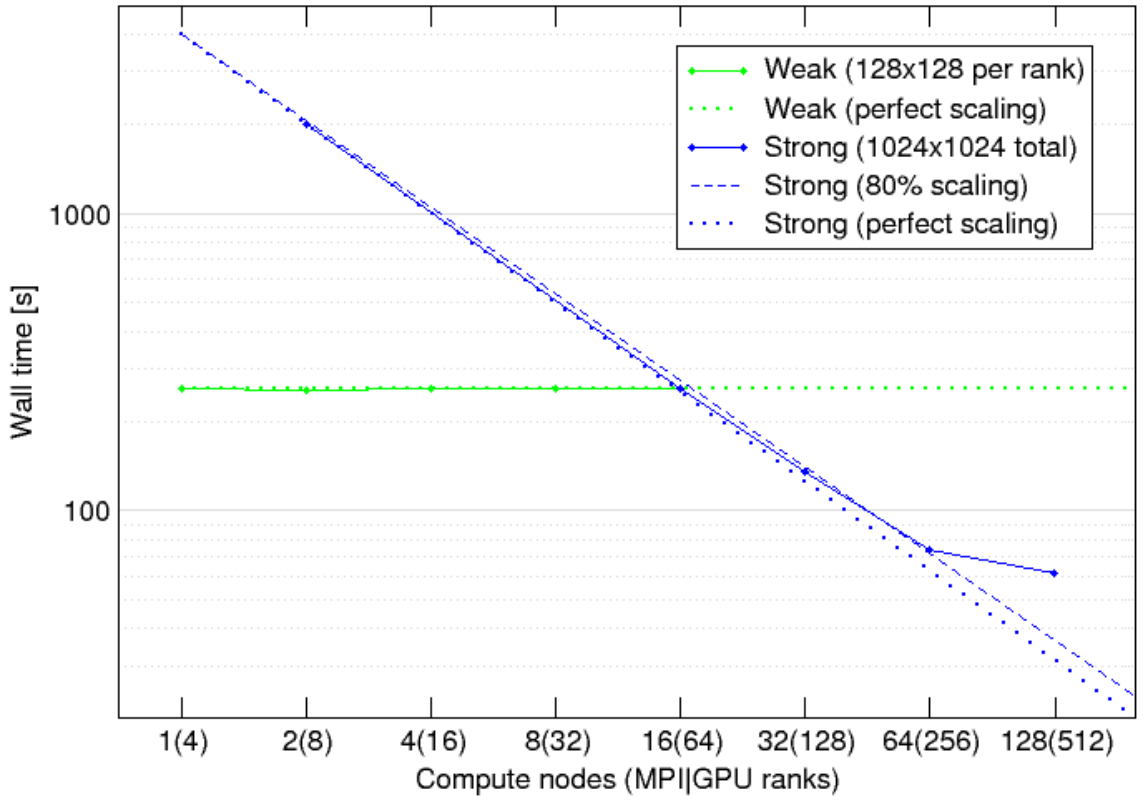
Execution call-tree & kernels

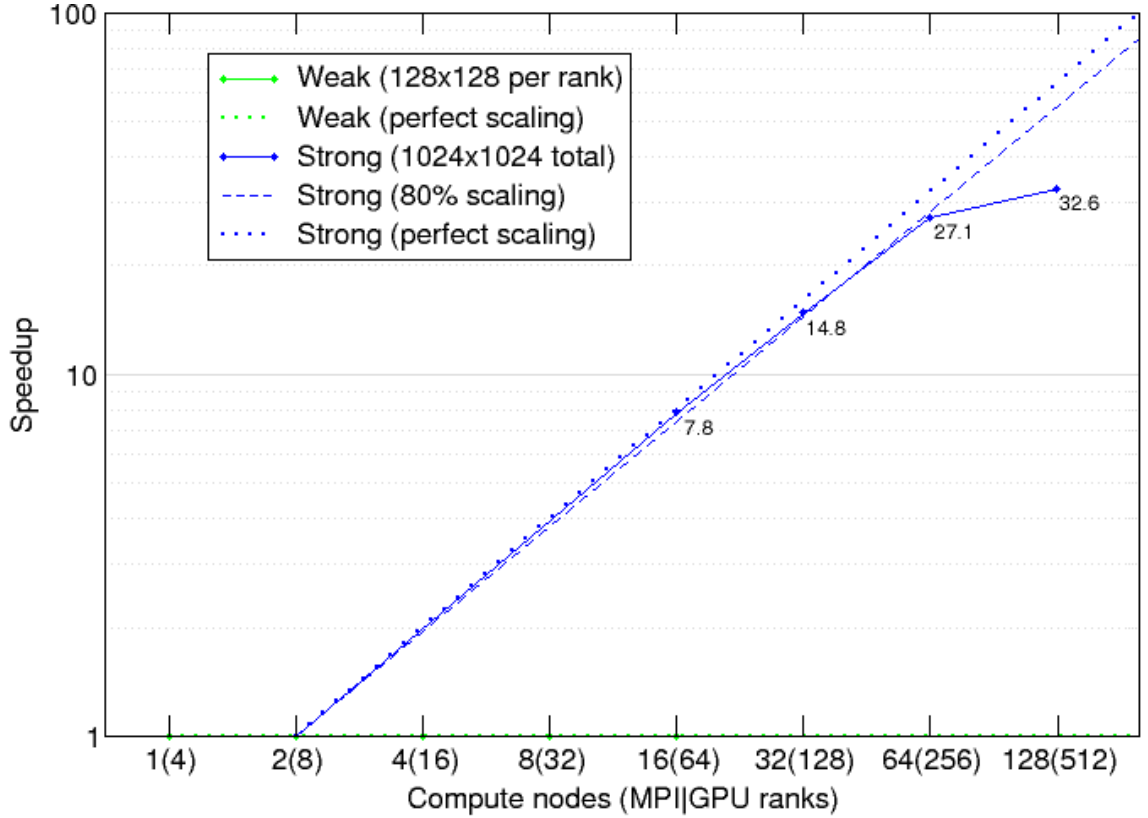


- Structure
  - solver (*iterate_time*)
    - contains all of the CUDA kernel executions
      - GPU devices idle during init/finalize
    - 48000 timesteps
      - seven of nine kernels executed by all ranks
      - characteristics often determined by position in 2D grid
    - *compute_add_sources_kernel* only executed by a single GPU (rank 243 of 512)
    - *compute_elastic_seismogram* only by 4 nearby GPUs (ranks 241, 245, 273, 277 of 512)
- Focus of Analysis selected for assessment: *iterate_time*

EuroHPC Joint Undertaking

Scaling & speed-up





- xspecfem3d FOA *iterate_time* on Leonardo-Booster
- Excellent weak scaling (expected to continue to higher node counts)
- Very good strong scaling (above 80% of perfect) to around 64 compute nodes (256 GPUs)

Efficiency model (weak scaling)

	Problem size	256x256	512x256	512x512	1024x512	1024x1024
	MPI|GPU ranks	**4**	**8**	**16**	**32**	**64**
	Wall time [s]	**255.873**	**255.462**	**256.035**	**255.898**	**255.948**
XPU	Global scaling efficiency	0.514	0.516	0.514	0.514	0.514
	- Computation time scaling	*1.000*	0.992	0.977	0.972	0.963
	- Parallel efficiency	0.514	0.520	0.526	0.529	0.534
	- - Load balance efficiency	0.522	0.526	0.533	0.536	0.541
	- - Orchestration efficiency	0.986	0.988	0.988	0.988	0.988
GPU	Global scaling efficiency	0.986	0.987	0.985	0.986	0.986
	- Computation time scaling	*1.000*	1.000	1.000	1.000	1.000
	- Parallel efficiency	0.986	0.987	0.985	0.986	0.986
	- - Load balance efficiency	1.000	0.999	0.997	0.998	0.998
	- - Orchestration efficiency	0.986	0.988	0.988	0.988	0.988

- Orchestration efficiency
  - MPI communication & CUDA management
  - aka Communication efficiency

Key:
1.10
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00

- Excellent GPU weak scaling efficiency
- Very poor CPU efficiency?
- Moderate XPU (GPU+CPU) efficiency?

EuroHPC Joint Undertaking

Efficiency model (strong scaling)

	Problem size	1024x1024	1024x1024	1024x1024	1024x1024	1024x1024
	MPI|GPU ranks	**8**	**64**	**128**	**256**	**512**
	Wall time [s]	**2001.806**	**255.948**	**135.478**	**73.846**	**61.400**
XPU	Global scaling efficiency	0.515	0.504	0.476	0.437	0.263
	- Computation time scaling	*1.000*	0.943	0.861	0.835	0.715
	- Parallel efficiency	0.515	0.534	0.553	0.523	0.367
	- - Load balance efficiency	0.518	0.541	0.583	0.586	0.671
	- - Orchestration efficiency	0.995	0.988	0.948	0.892	0.547
GPU	Global scaling efficiency	0.995	0.973	0.919	0.843	0.507
	- Computation time scaling	*1.000*	0.987	0.972	0.950	0.937
	- Parallel efficiency	0.995	0.986	0.945	0.887	0.541
	- - Load balance efficiency	1.000	0.998	0.996	0.994	0.989
	- - Orchestration efficiency	0.995	0.988	0.948	0.892	0.547

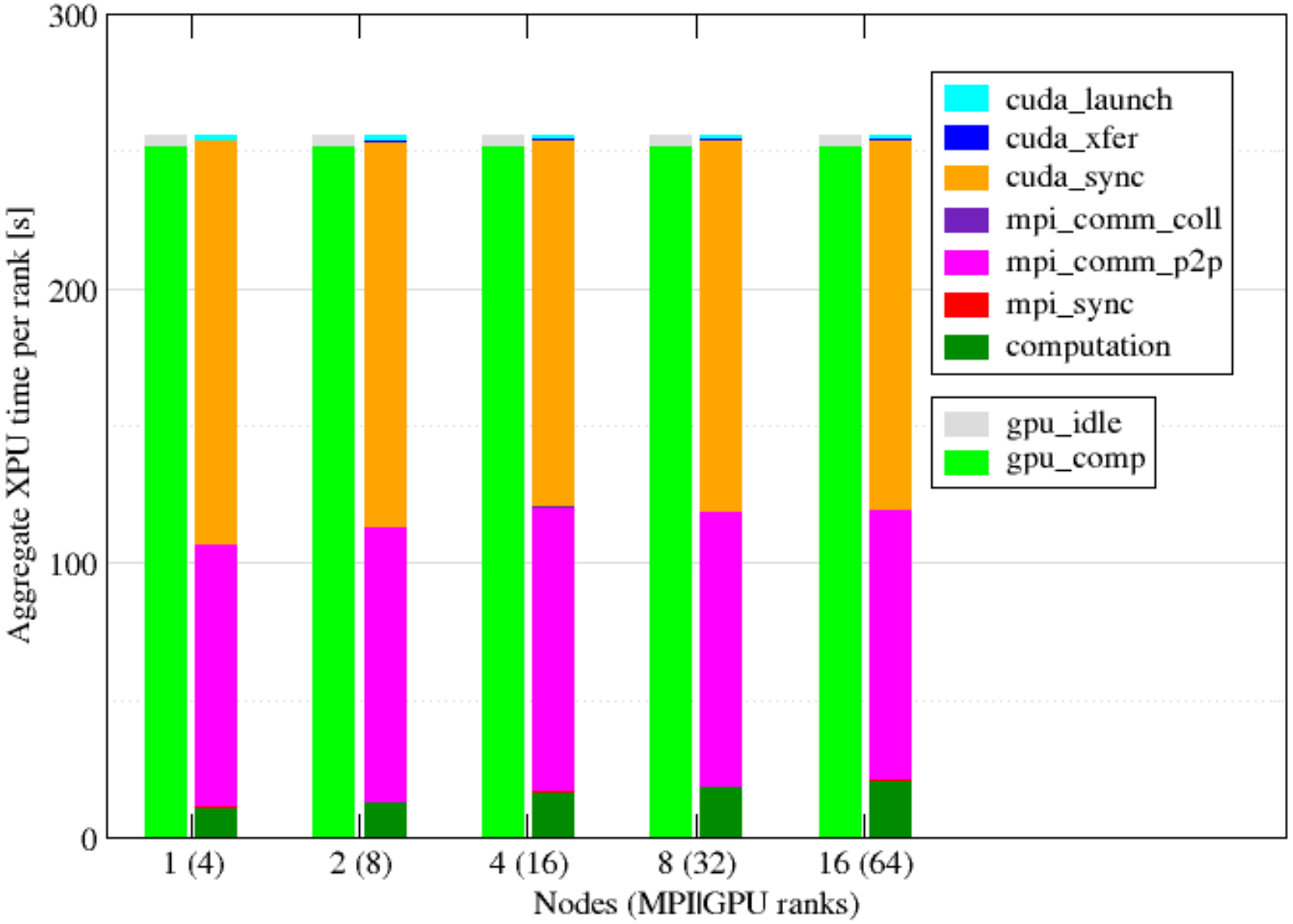
- Orchestration efficiency
  - MPI communication & CUDA management
  - aka Communication efficiency

Key:
1.10
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00

- Good GPU weak scaling efficiency to 128 GPUs (excellent load balance)
- Very poor CPU efficiency?
- Moderate XPU (GPU+CPU) efficiency?

EuroHPC
Joint Undertaking

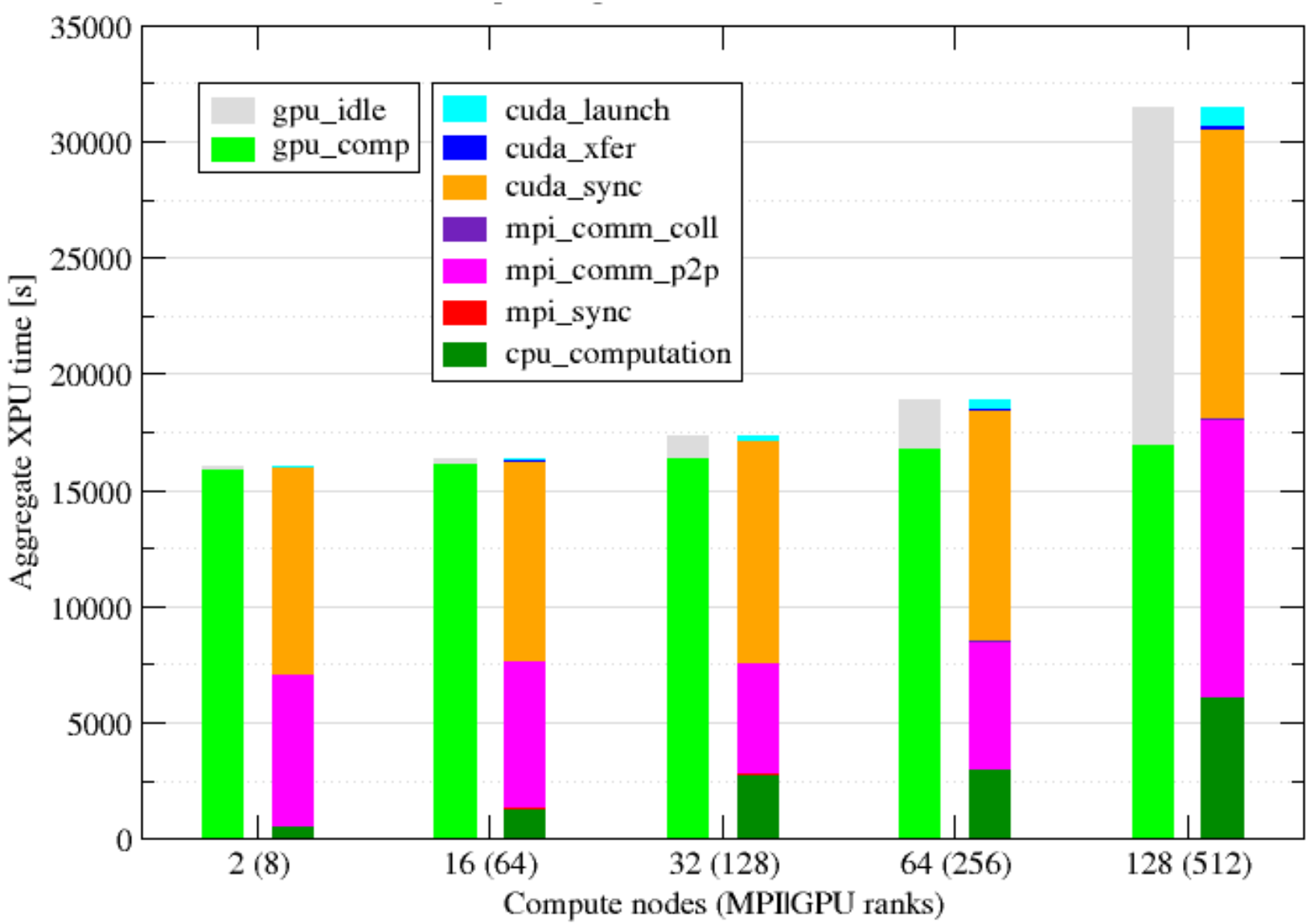
Weak scaling (128x128 per rank)



- Excellent weak scaling
- Little GPU idle time
- MPI communication effectively overlapped with GPU kernel computation

EuroHPC
Joint Undertaking

Strong scaling (1024x1024 total)

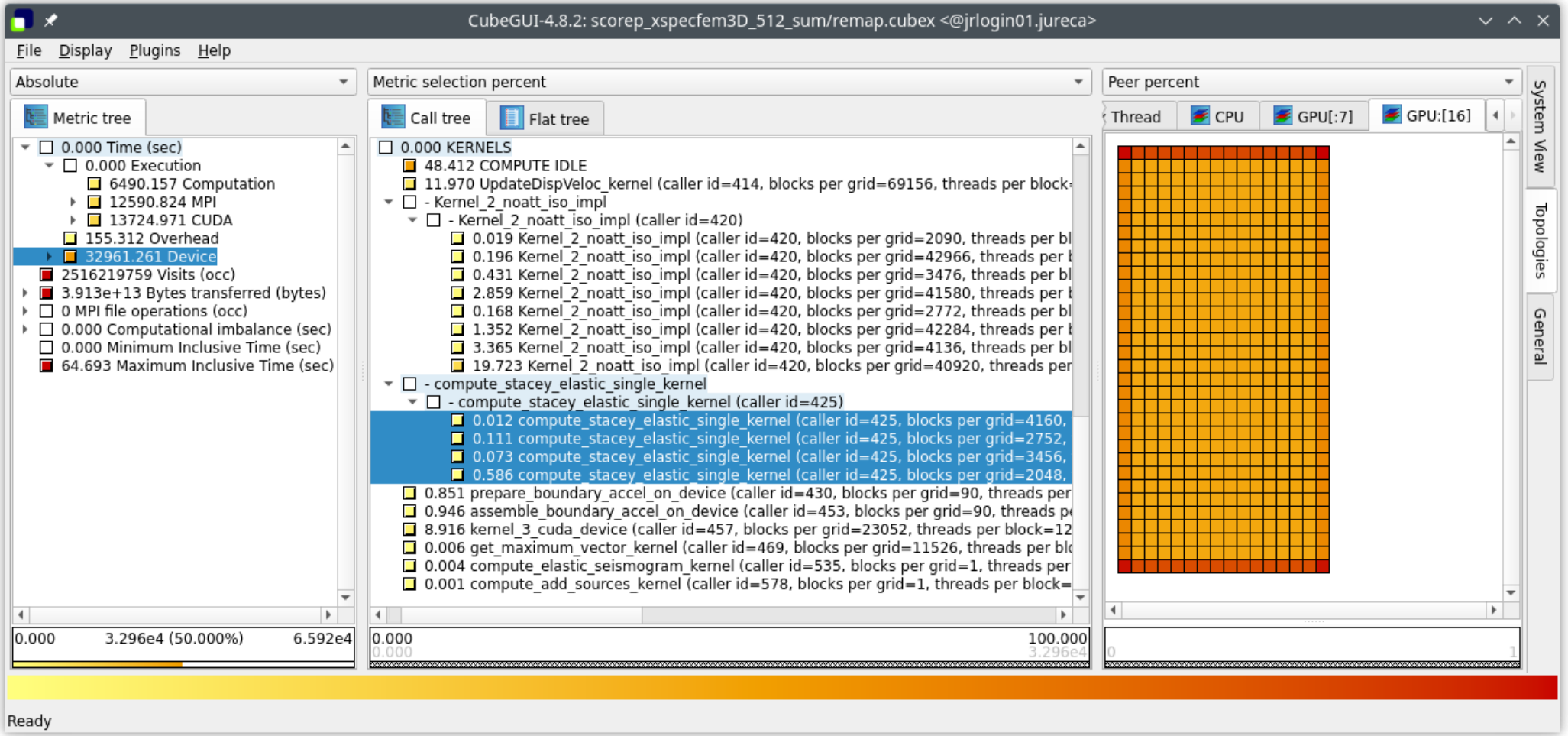


- Good scaling to 256 GPUs (64 nodes)
- GPU computation time slowly grows progressively
- GPU idle time grows for 256 & particularly 512 GPUs
- CPU computation time grows substantially
  - sync_copy_from_device & transfer_boundary_to_device_a
- For 512 GPUs, growing MPI communication no longer fully overlapped with GPU kernel computation

EuroHPC Joint Undertaking

Topological inhomogeneities

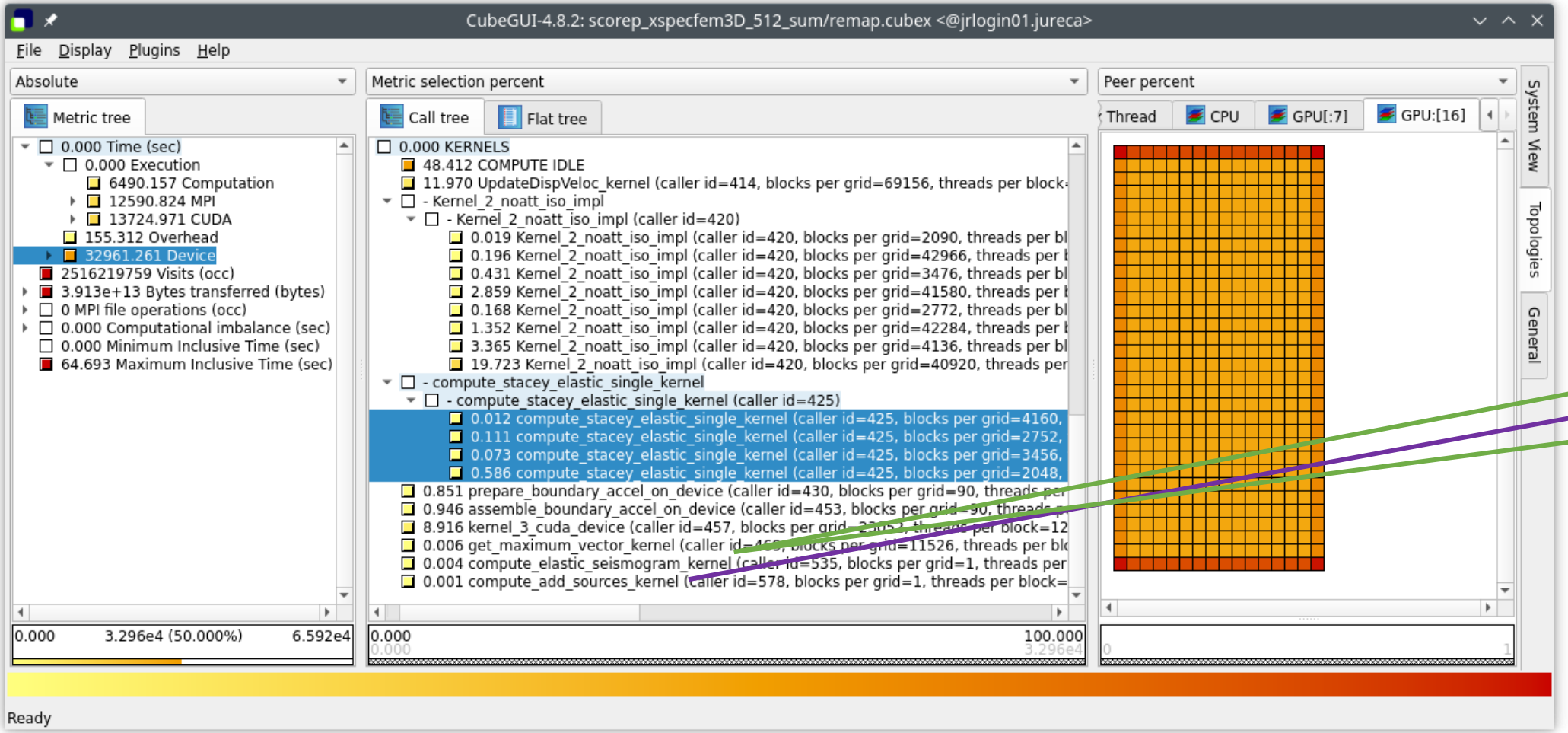
Kernel variant executed (characteristics and corresponding execution time) varies according to position in 2D grid: four corners, upper/lower & left/right edges, interior



16x32 grid

Topological inhomogeneities

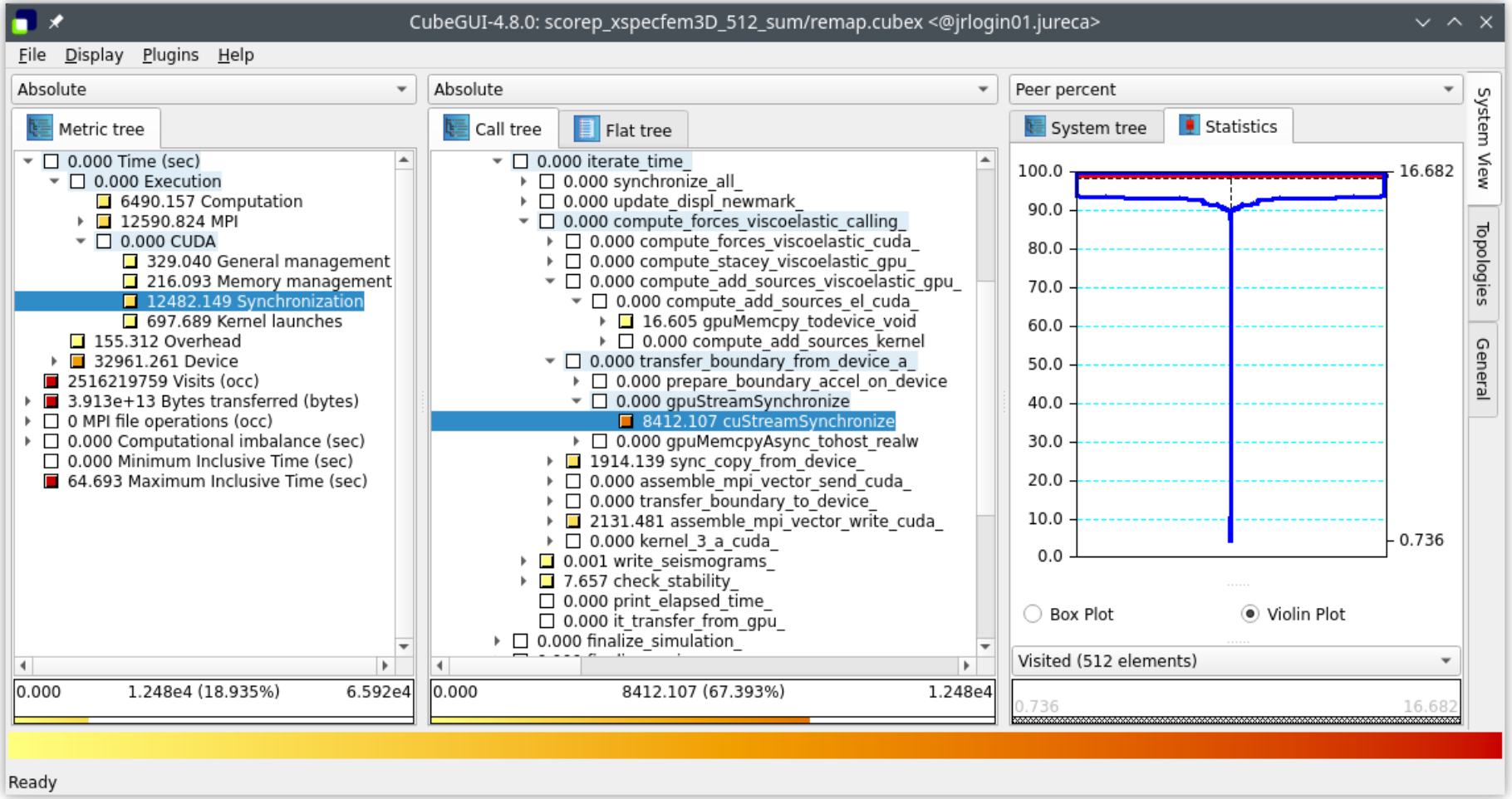
Kernel variant executed (characteristics and corresponding execution time) varies according to position in 2D grid: four corners, upper/lower & left/right edges, interior

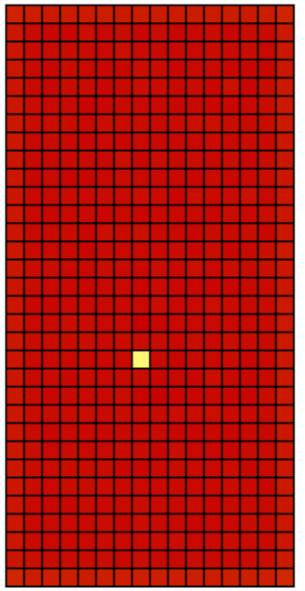


16x32 grid

GPU computation imbalance

36% of CPU execution time is CUDA synchronization, 67% of which is 16s within *transfer_boundary_from_device_a* following *compute_add_sources_kernel* that's only executed by a single GPU (source rank 243 of 512)





Summary of observations

- *iterate_time* (solver) chosen as focus of analysis
  - negligible time for initialization/finalization
- Excellent weak scaling up to 16 nodes (64 GPUs) and likely beyond
  - Computation very well balanced over GPUs; Excellent GPU utilization
  - MPI P2P communication time grows with scale, but effectively overlapped with GPU computation kernels
- Good strong scaling speedup up to 64 nodes (256 GPUs)
  - Computation remains very well balanced over GPUs
  - Orchestration efficiency progressively diminishes
    - *compute_add_sources_kernel* execution by a single GPU seems the main origin
  - MPI P2P communication time grows significantly, becomes no longer fully overlapped with GPU computation kernels

EuroHPC
Joint Undertaking

Performance Optimisation and Productivity

A Centre of Excellence in HPC

Contact:

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

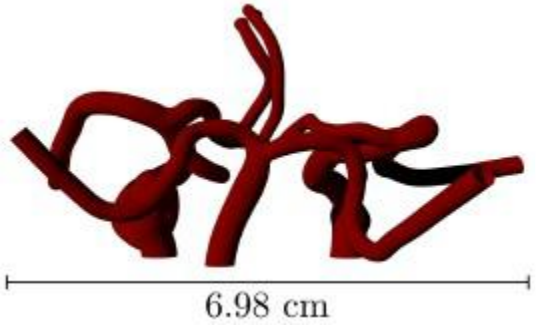
This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101143931. The JU receives support from the European Union's Horizon Europe research and innovation programme and Spain, Germany, France, Portugal and the Czech Republic.

HemeLB (SuperMUC-NG)

https://co-design.pop-coe.eu/reports/POP2-AR-041-HemeLB.html

DOI 10.5281/zenodo.4105743
DOI 10.5281/zenodo.4275232

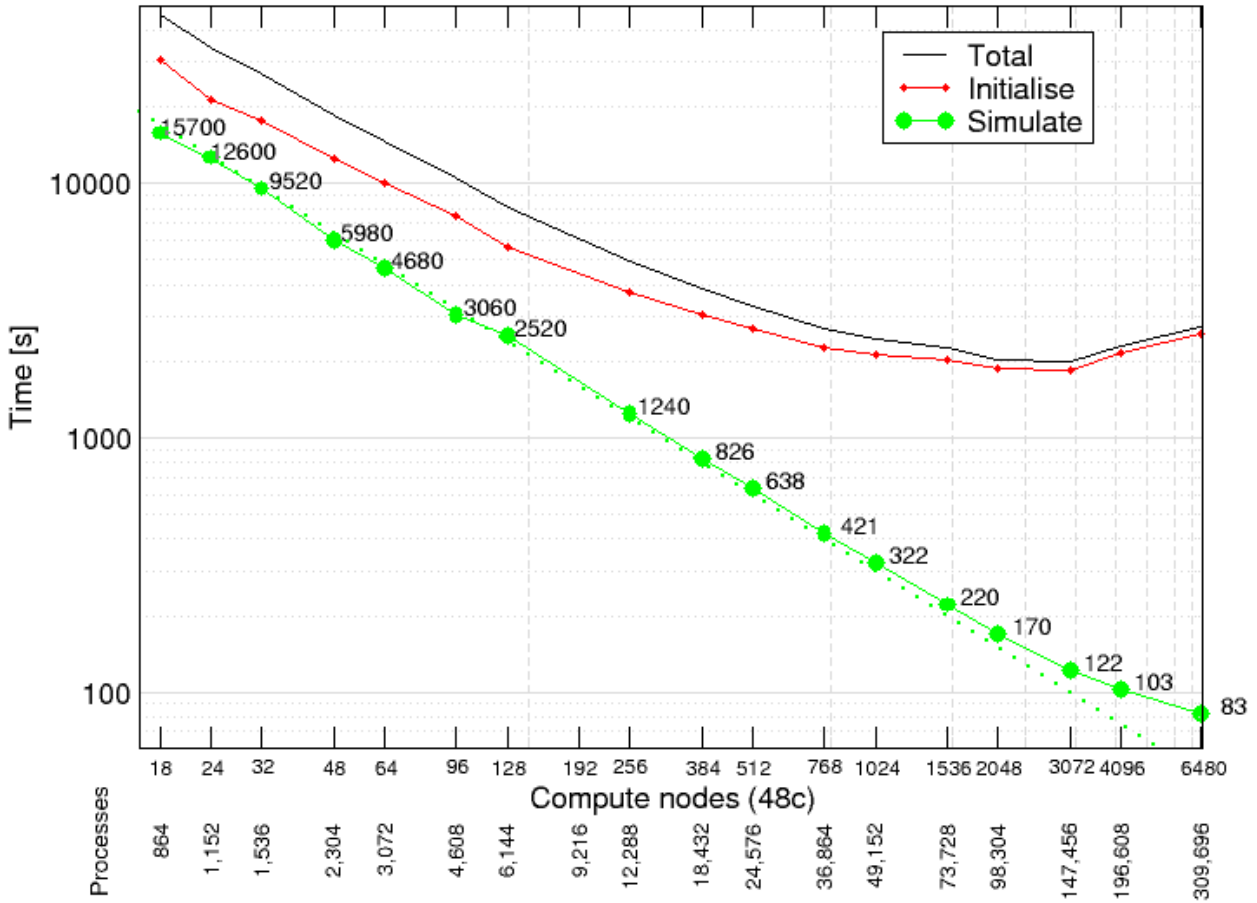
- 3D macroscopic blood flow in human arterial system developed by UC London (UK)
  - lattice-Boltzmann method tracking fluid particles on a lattice grid with complex boundary conditions
  - exascale flagship application of EU H2020 HPC Centre of Excellence for Computational Biomedicine (CompBioMed)
- HemeLB open-source code and test case: www.hemelb.org
  - C++ parallelized with MPI
    - Intel Studio 2019u4 compiler and MPI library (v19.0.4.243)
    - configured with 2 ‘reader’ processes (intermediate MPI file writing disabled)
    - MPI-3 shared-memory model employed within compute nodes to reduce memory requirements when distributing lattice blocks from reader processes
  - Focus of analysis 5,000 time-step (500µs) simulation of cerebrovascular “circle of Willis” geometry
    - 6.4µm lattice resolution (21.15 GiB): 10,154,448,502 lattice sites
- Executed on *SuperMUC-NG* Lenovo ThinkSystem SD650 (LRZ):
  - 2x 24-core Intel Xeon Platinum 8174 (‘Skylake’) @ 3.1GHz
  - 48 MPI processes/node, 6452 (of 6480) compute nodes: 309,696 MPI processes
  - 190x speed-up from 864 cores: 80% scaling efficiency to over 100,000 cores

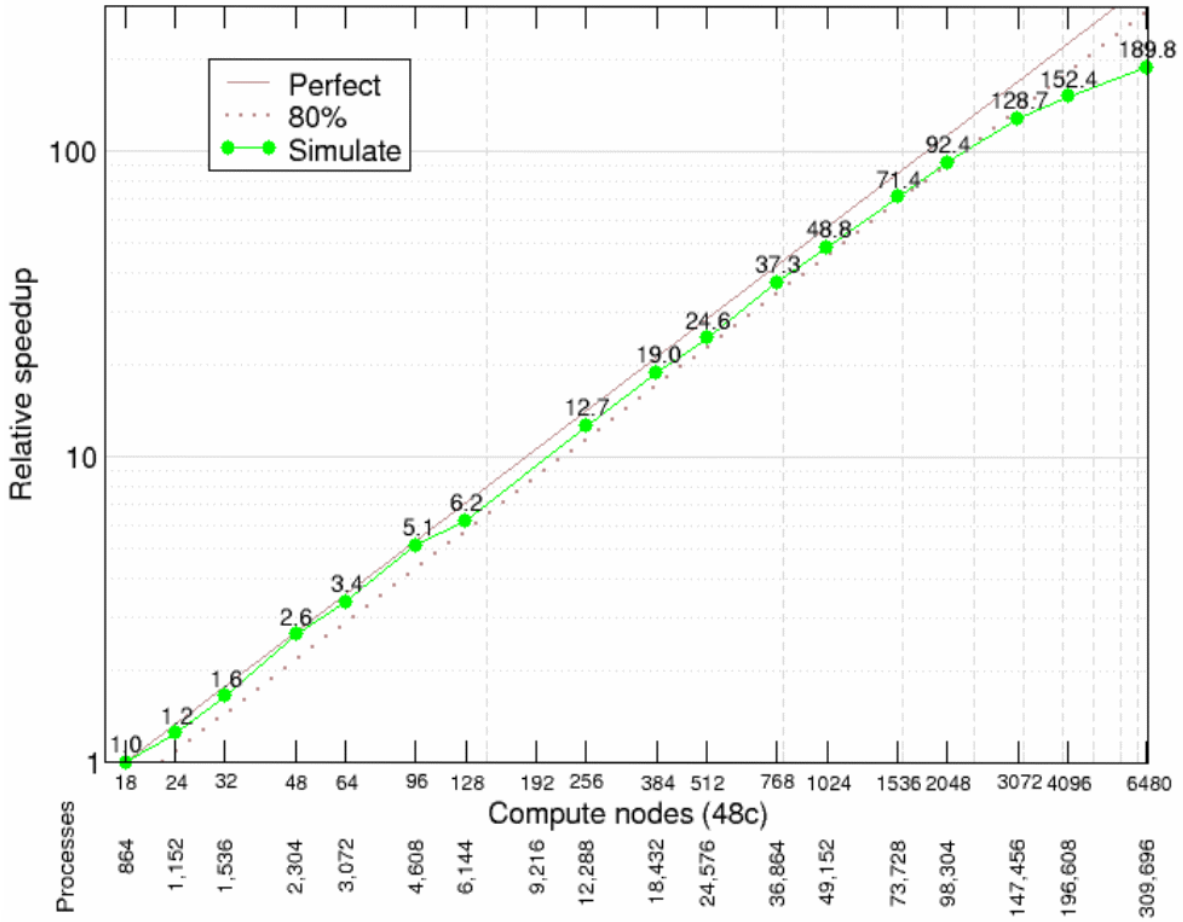


⇒ Identification & quantification of impact of load balance and its variation

EuroHPC Joint Undertaking

HemeLB@SNG strong scaling





[Execution of 9,216 processes on 192 compute nodes not possible due to insufficient compute nodes with adequate memory in 'fat' partition (768 GiB vs. regular 96 GiB node memory]

EuroHPC
Joint Undertaking

HemeLB@SNG strong scaling efficiency

Compute nodes	24	32	48	64	96	128	192	256	384	512	768	1024	1536	2048	3072	4096	6452
Processes	**1152**	**1536**	**2304**	**3072**	**4608**	**6144**	**9216**	**12288**	**18432**	**24576**	**36864**	**49152**	**73728**	**98304**	**147456**	**196608**	**309696**
Global scaling efficiency	**0.79**	**0.79**	**0.84**	**0.80**	**0.82**	**0.75**		**0.73**	**0.72**	**0.73**	**0.74**	**0.68**	**0.68**	**0.65**	**0.62**	**0.57**	**0.45**
- Parallel efficiency	0.79	0.80	0.87	0.83	0.86	0.80		0.75	0.74	0.74	0.77	0.71	0.72	0.70	0.72	0.70	0.73
- - Load balance efficiency	0.79	0.80	0.88	0.84	0.86	0.80		0.75	0.74	0.75	0.78	0.72	0.74	0.72	0.74	0.73	0.80
- - Communication efficiency	1.00	1.00	1.00	1.00	1.00	1.00		1.00	1.00	0.99	0.99	0.99	0.98	0.98	0.97	0.96	0.92
- Computation scaling	*1.00*	0.99	0.96	0.96	0.95	0.93		0.98	0.98	0.98	0.96	0.96	0.94	0.93	0.87	0.81	0.61
- - Instructions scaling	*1.00*	1.00	1.00	1.00	1.00	1.00		1.00	1.00	1.00	0.99	0.97	0.94	0.89	0.79	0.67	0.45
- - IPC scaling	*1.00*	0.99	0.96	0.96	0.95	0.93		0.98	0.98	0.99	0.98	0.99	1.00	1.04	1.11	1.21	1.36
IPC	1.411	1.395	1.353	1.355	1.342	1.316		1.377	1.387	1.396	1.383	1.390	1.417	1.473	1.566	1.704	1.919
											Key:	<0.65	<0.75	<0.85	<0.95	<1.00	>1.00

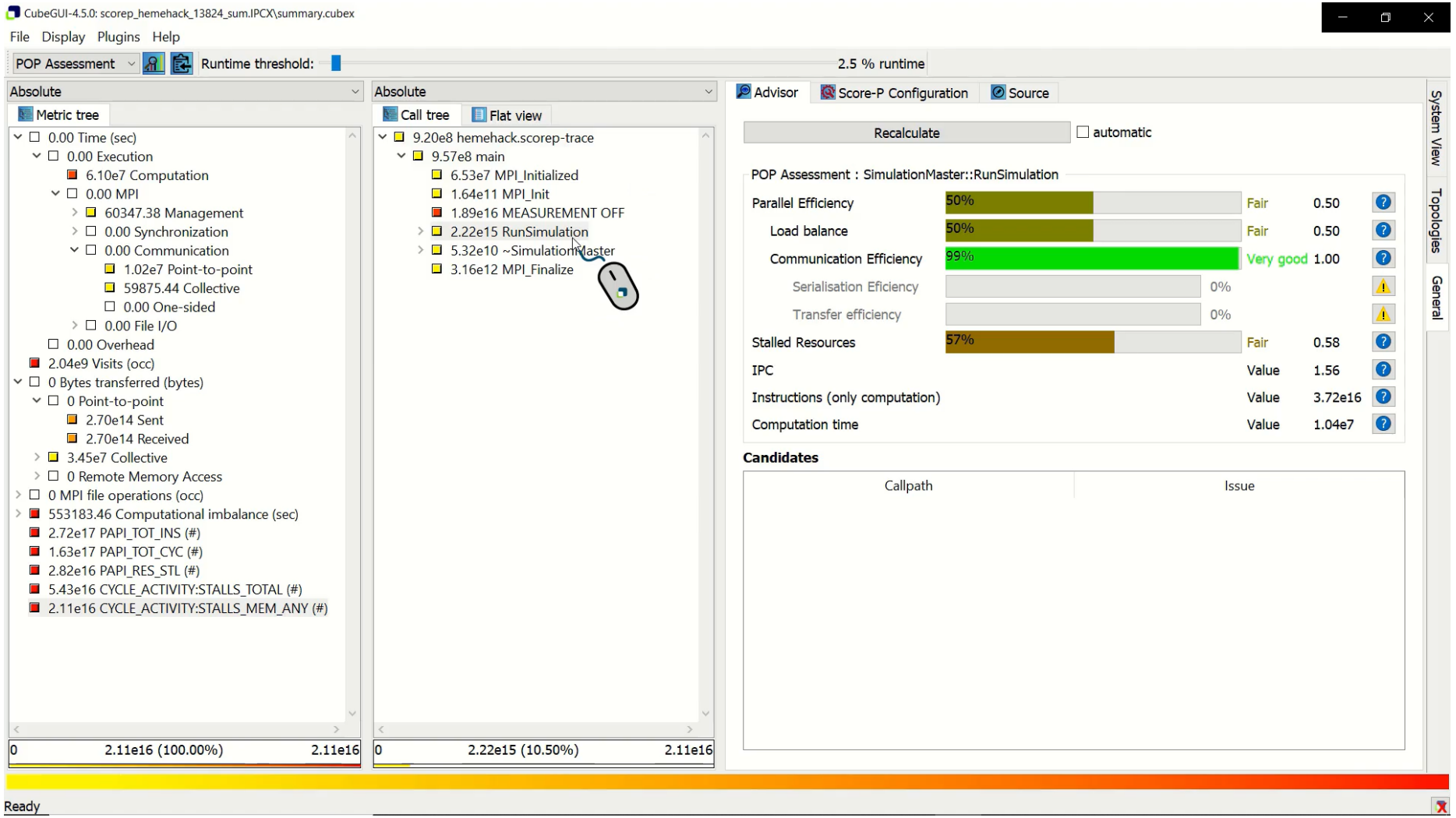
Global scaling efficiency fairly good around 80%, before degrading at larger scales

- Parallel efficiency deteriorating following Load balance efficiency
  - Communication efficiency excellent throughout
- Computation scaling (relative to 1152 processes) very good except at largest scale
  - Degradation of Instructions scaling partially compensated by improving IPC scaling

[POP CoE scaling efficiency model: www.pop-coe.eu]

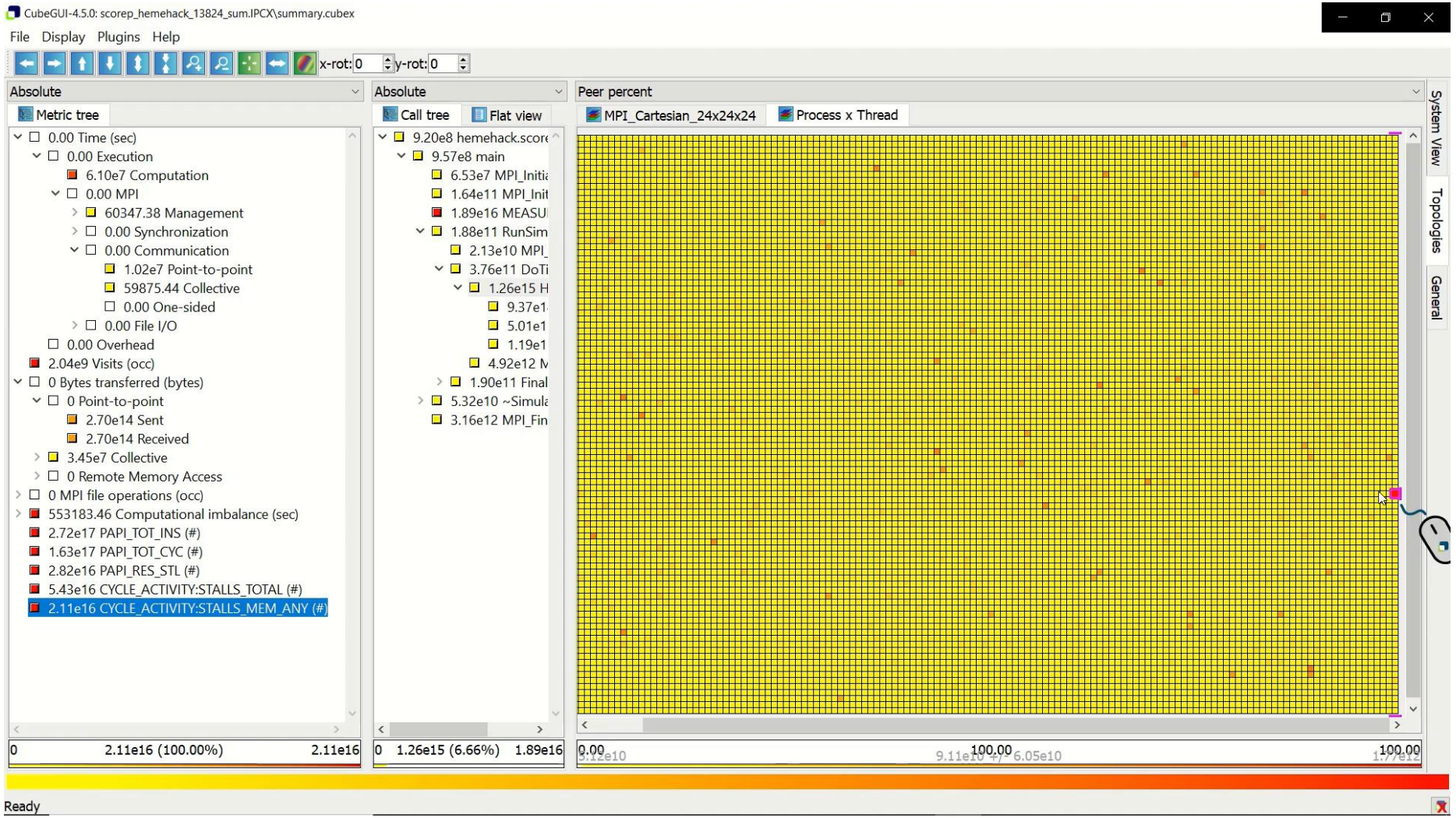
EuroHPC Joint Undertaking

Advisor: POP efficiency assessment



EuroHPC
Joint Undertaking

Topological presentation



HemeLB_GPU (JUWELS-Volta)

https://co-design.pop-coe.eu/reports/POP2-AR-065-HemeLB_GPU.html

DOI 10.5281/zenodo.4117942
DOI 10.5281/zenodo.4081080

- 3D macroscopic blood flow in human arterial system developed by UC London (UK)
  - lattice-Boltzmann method tracking fluid particles on a lattice grid with complex boundary conditions
  - exascale flagship application of EU H2020 HPC Centre of Excellence for Computational Biomedicine
- HemeLB open-source code and test case: www.hemelb.org
  - C++ parallelized with MPI + CUDA (in development)
    - GCC/8.3.0 compiler, CUDA/10.1.105 and ParaStationMPI/5.4 library
    - configured with 2 ‘reader’ processes and intermediate MPI file writing
    - rank 0 ‘monitor’ process doesn’t participate in simulation

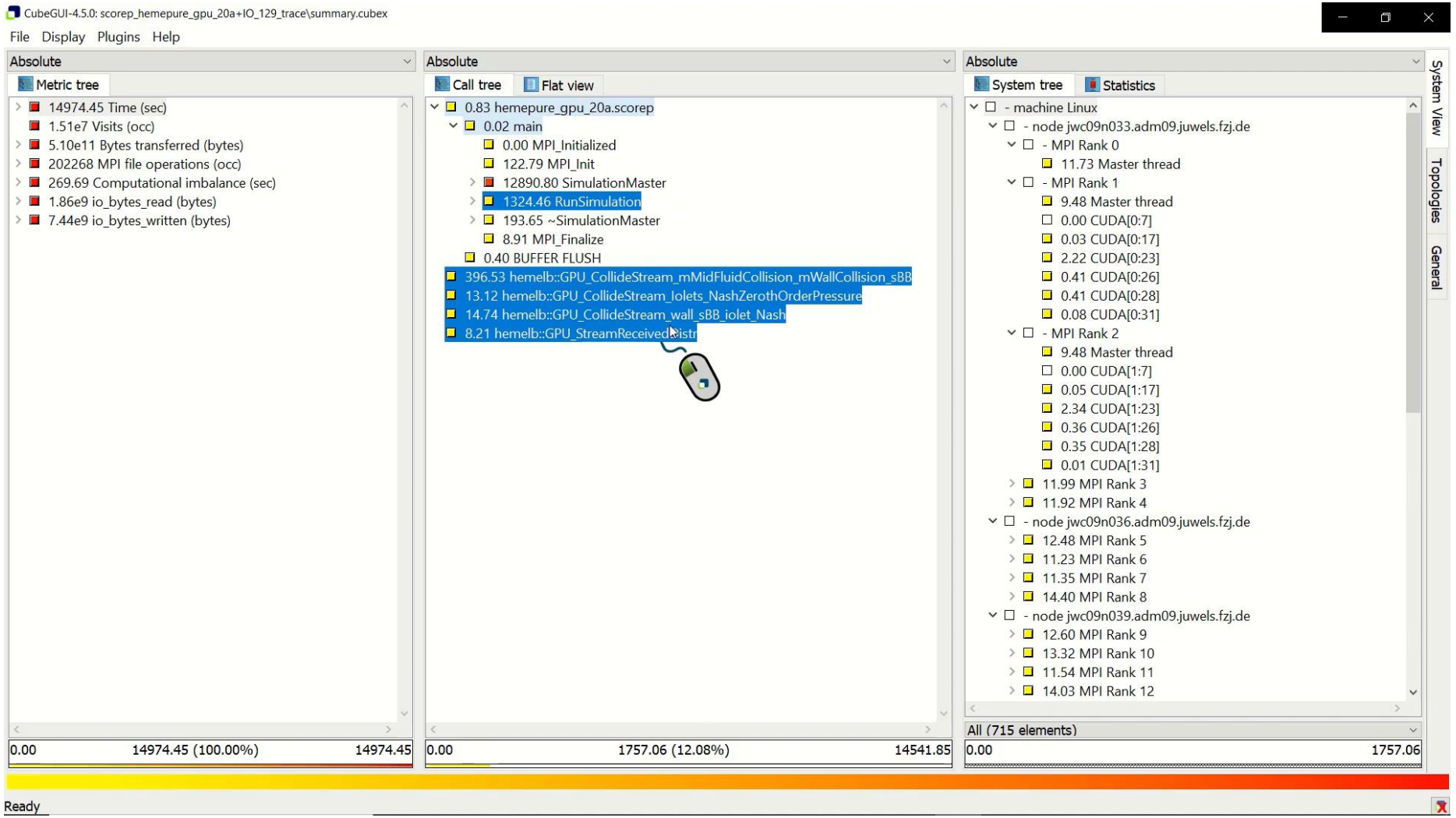


  - Focus of analysis 2,000 time-step (each 100µs) simulation of CBM2019_Arteries_patched geometry
    - 1.78 GiB: 66,401,494 lattice sites, 1+38 iolets
- Executed on *JUWELS-Volta* (@JSC):
  - 2x 20-core Intel Xeon Platinum 8168 (‘Skylake’) CPUs + 4 Nvidia V100 ‘Volta’ GPUs
  - 4* MPI processes/node (one per GPU), 32 (of 56) compute nodes: 129 MPI processes

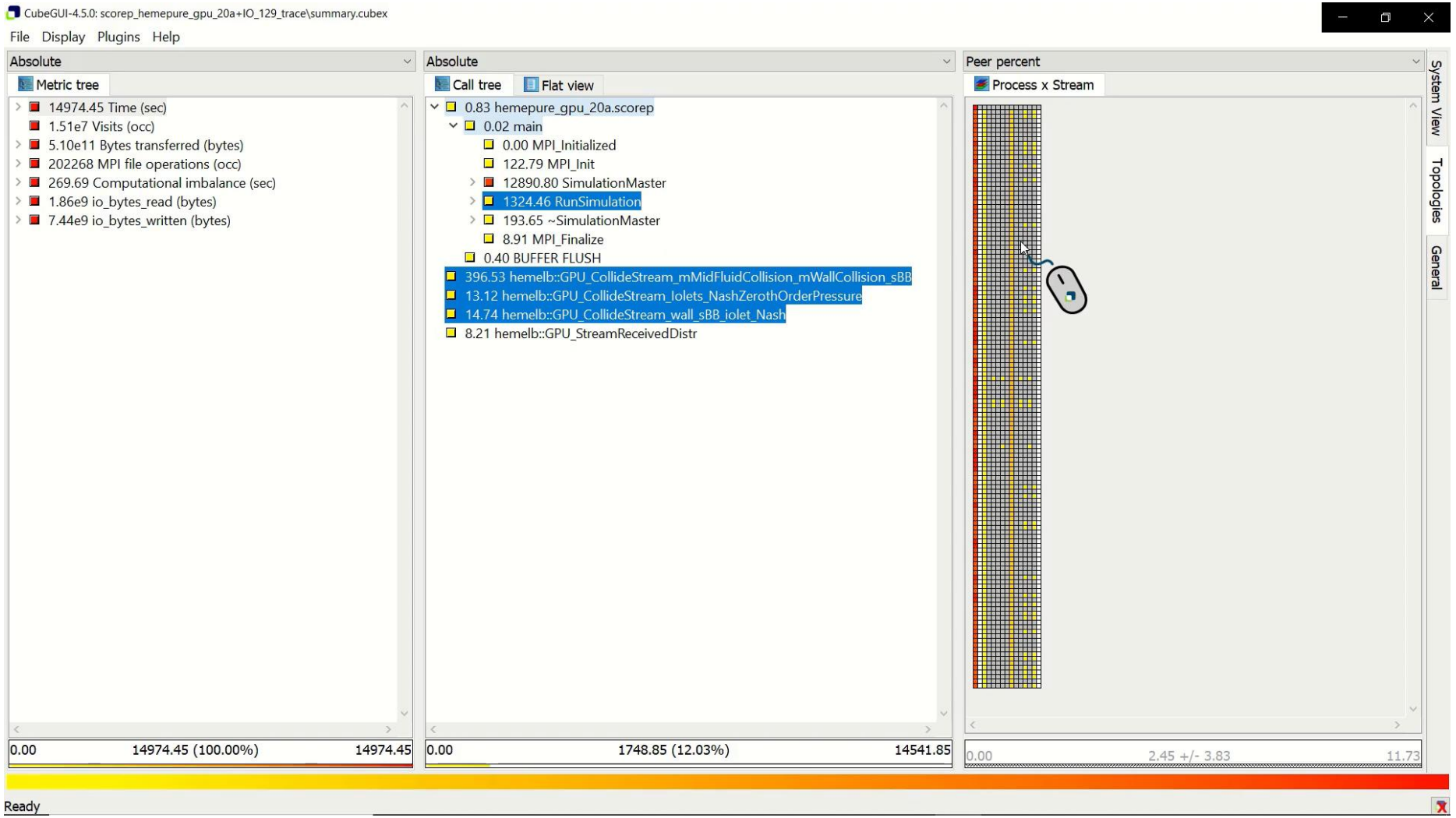
⇒ Identification & quantification of impact of CPU/GPU load balance and its variation

EuroHPC Joint Undertaking

Time for asynch. CUDA kernels

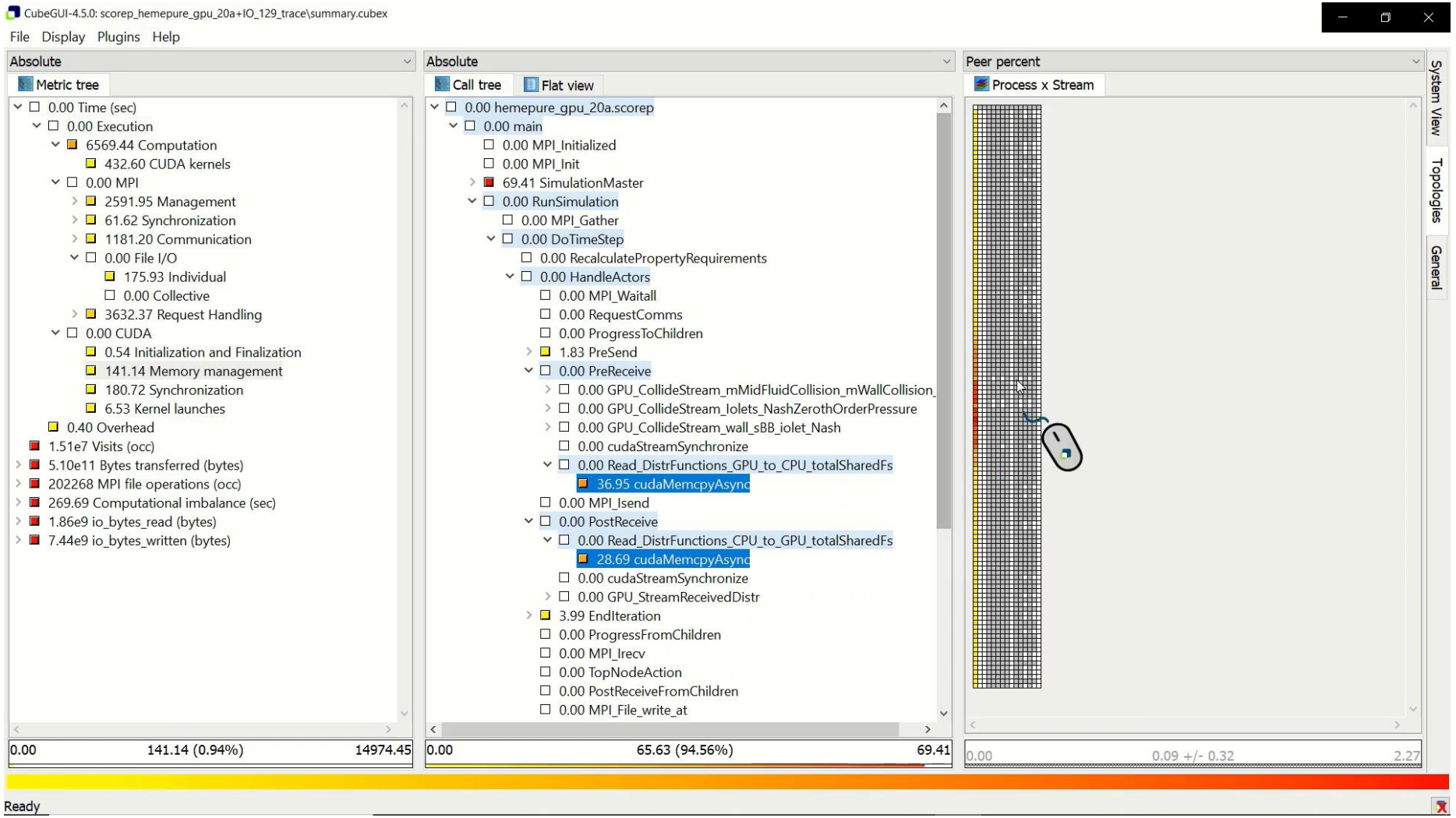


Time for asynch. CUDA kernels



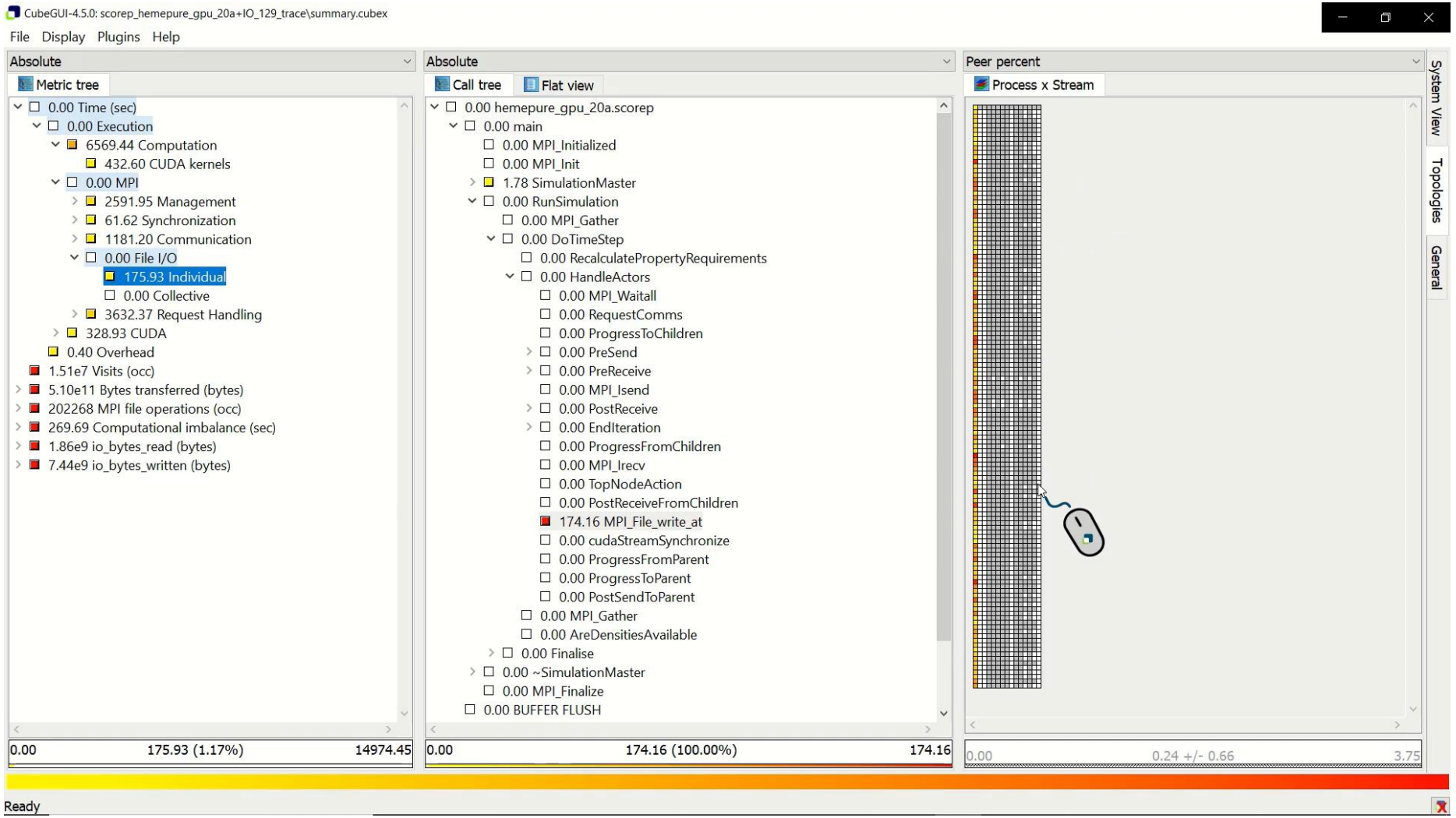
EuroHPC
Joint Undertaking

Time for CUDA asynch. memory copies

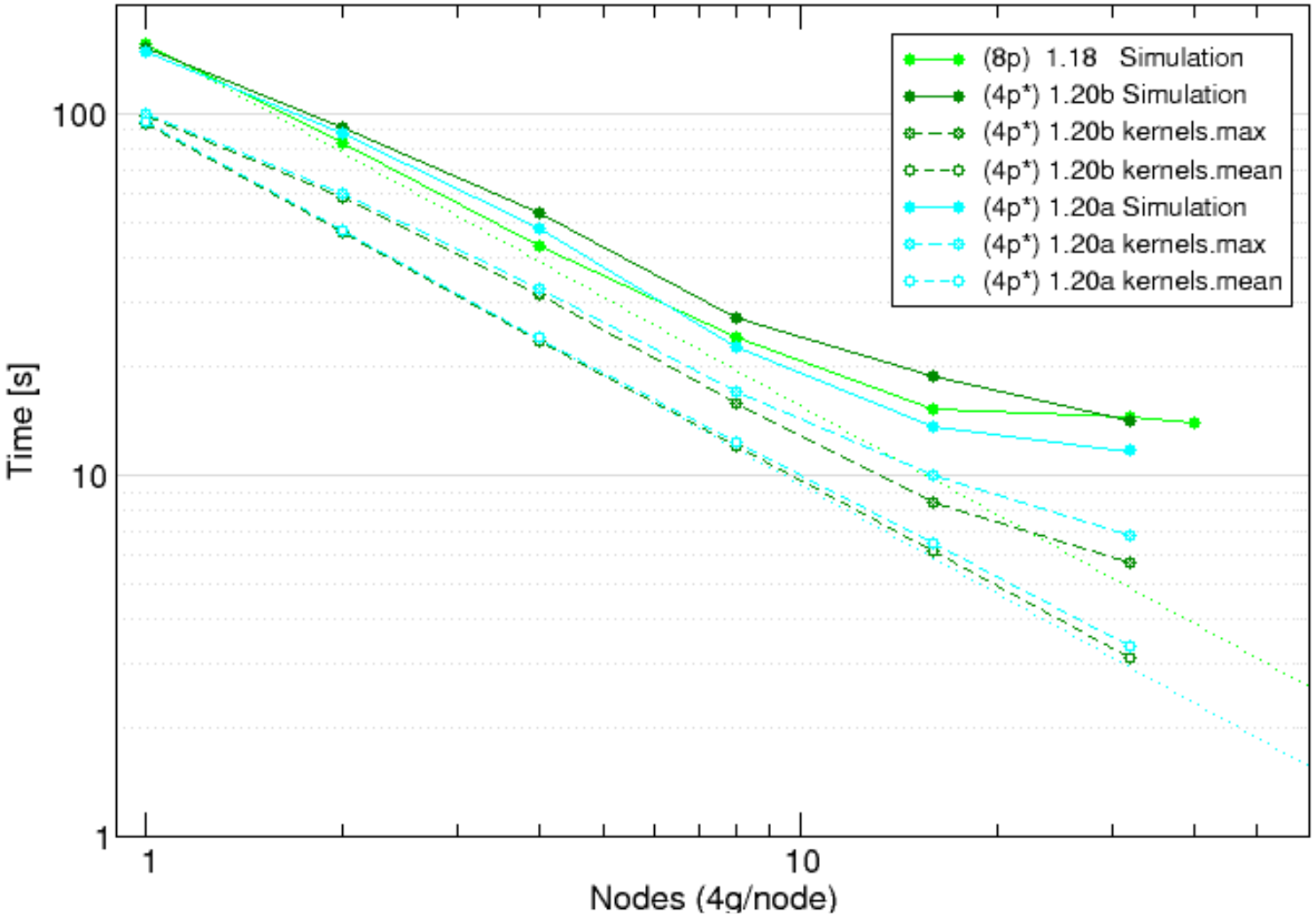


EuroHPC
Joint Undertaking

Time for MPI file writing on CPU



HemeLB@JUWELS-Volta strong scaling

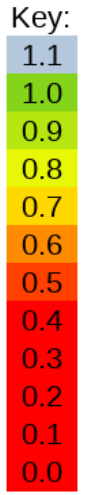


- Reference execution with 8ppn
  - multiple processes offloading GPU kernels generally unproductive
- Comparison of versions (4ppn)
  - v1.20a generally better
- Synchronous MPI file writing is the primary bottleneck
- CUDA kernels on GPUs
  - less than half of Simulation time (therefore GPUs mostly idle)
  - total kernel time scales very well (0.93 scaling efficiency)
  - load balance deteriorates (0.95 for single node, 0.50 for 32 nodes)

EuroHPC Joint Undertaking

HemeLB@JUWELS-V scaling efficiency

	1n 5p	2n 9p	4n 17p	8n 33p	16n 65p	32n 129p
Simulation time [s]	147.87	88.38	48.13	22.66	13.68	11.67
Global scaling efficiency	0.64	0.53	0.49	0.52	0.43	0.25
– Parallel efficiency	0.64	0.53	0.50	0.54	0.47	0.29
– – Load balance efficiency (GPU)	0.95	0.78	0.73	0.73	0.65	0.50
– – Communication efficiency (GPU)	0.67	0.68	0.68	0.75	0.73	0.58
– Computation scaling (GPU)	1.00	1.00	0.99	0.96	0.92	0.87



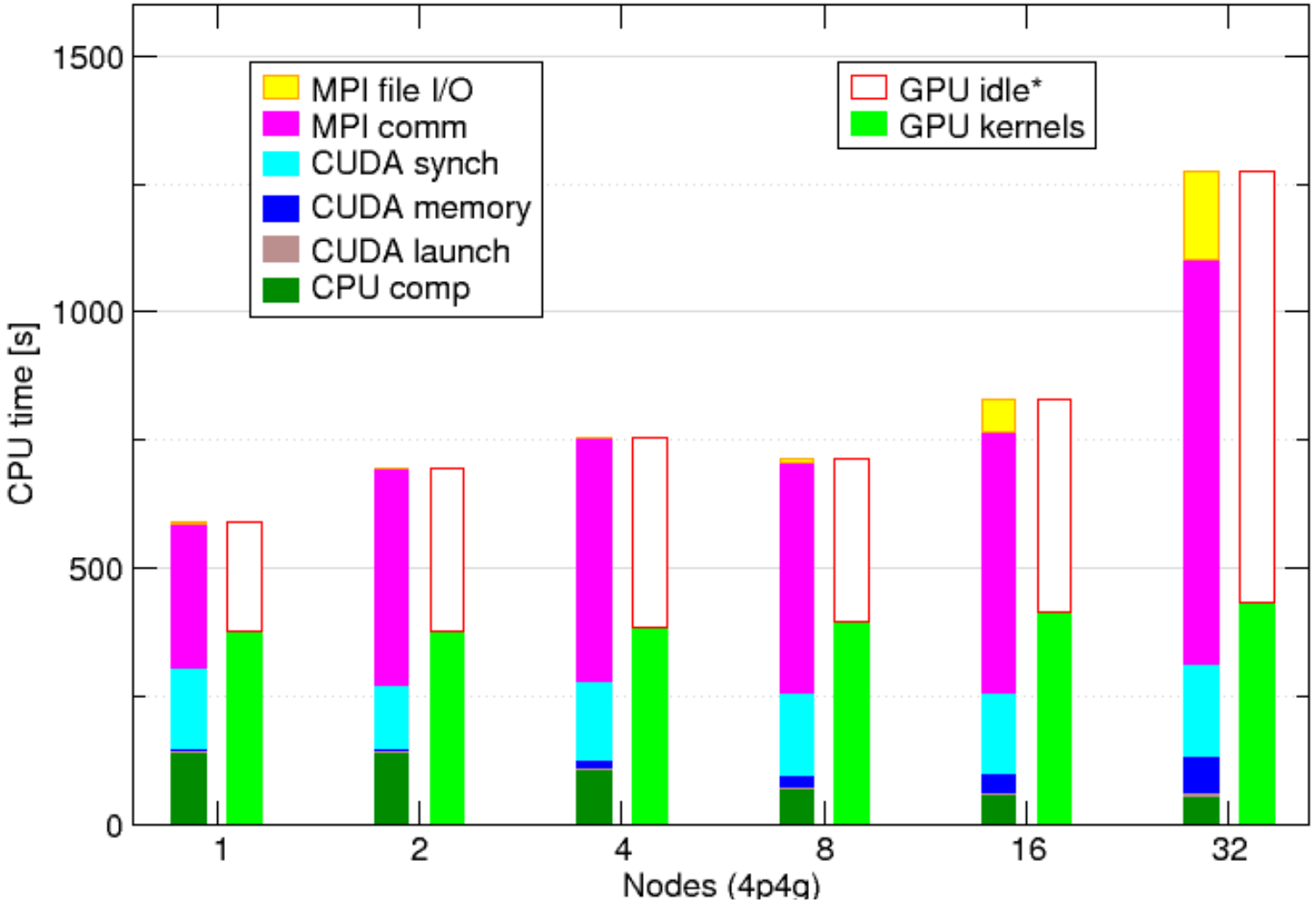
Only considering GPUs (ignoring all CPU cores, 90% of which are completely unused)

- Single (quad-GPU) node already suffers significant communication inefficiency
  - includes MPI file writing, but doesn't degrade much as additional nodes are included
- Load balance of GPUs deteriorates progressively
- GPU computation scaling remains reasonably good

[POP CoE scaling efficiency model: www.pop-coe.eu]

EuroHPC
Joint Undertaking

HemeLB@JUWELS-V strong scaling



- CPU+GPU time breakdown
- CUDA kernels on GPUs
  - less than half of Simulation time (therefore GPUs mostly idle)
  - total kernel time scales very well (0.87 scaling efficiency)
- MPI processes on CPUs
  - computation time decreases
  - CUDA synchronization time fairly constant, but time for memory management increases somewhat
  - MPI communication time dominates, with much more time for file writing with 16+ nodes

EuroHPC
Joint Undertaking