Determining parallel application execution efficiency & scaling using the POP methodology

Marta García-Gasulla & Sandra Mendez (Barcelona Supercomputing Center)
Anke Visser & Brian Wylie (Jülich Supercomputing Centre)

HORIZON-EUROHPC-JU-2023-COE

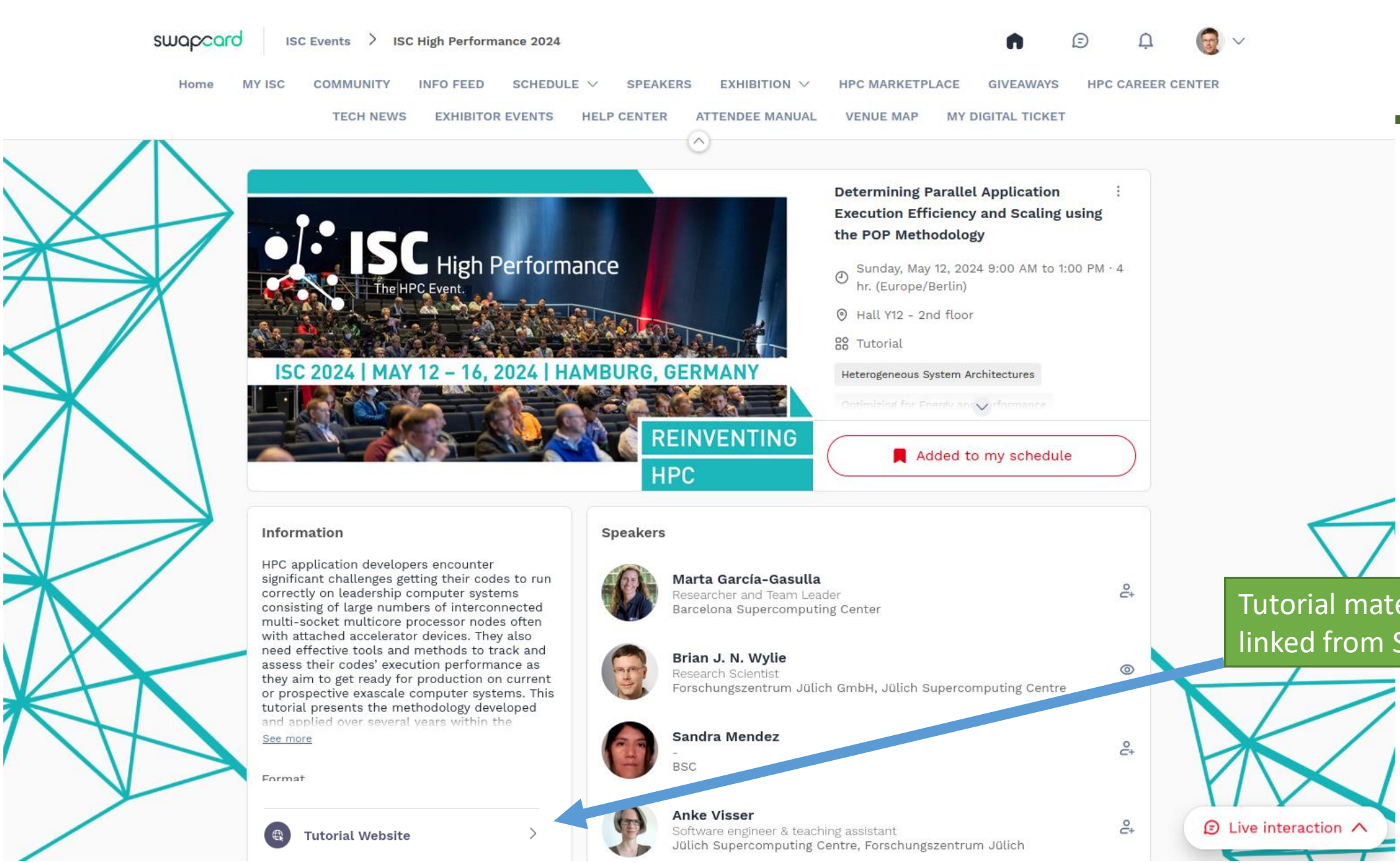
1 January 2024– 31 December 2026

Grant Agreement No 101143931

Logistics

- [120] Presentations with demonstrations of tools
- (break)
- [90] Hands-on exercises using tools with provided measurements

- Download material for hands-on exercises from SwapCard page
  - https://fz-juelich.sciebo.de/s/ku8yg5uAlssdWso
- Folder with slides & measurements for analysis:
  - ISC24_tutorial_guide.pdf (preparation instructions)
  - BTMZ & SMXV (Scalasca/CUBE profiles)
  - lulesh (Paraver/Extrae traces)



Tutorial materials website linked from Swapcard page

Agenda

- Welcome/Introduction
  - [15] POP Centre of Excellence goals, services and tools
  - [30] POP methodology and scaling/efficiency metrics
- Review of representative performance assessments
  - [30] Assessments using JSC tools
  - [30] Assessments using BSC tools
- Setup for hands-on exercises
  - [15] Installing Paraver & Scalasca/CUBE GUIs
- (break)
- Analyzing provided measurements
  - [75] Demonstrations & hands-on exercises
- Review and conclusion

Agenda for hands-on session

- Hands-on exercises analyzing provided measurements
  - Paraver
    - Determining a suitable focus of analysis (FOA) from event traces
    - Determining efficiencies for the FOA
    - In-depth examination
      - Clustering
  - Scalasca/CUBE
    - Determining a suitable focus of analysis (FOA) from profiles
    - Determining efficiencies for the FOA
    - In-depth examination
      - Critical path and delay analysis
- Review and conclusion

Selected performance assessments using Scalasca/Score-P/CUBE

Brian Wylie (Jülich Supercomputing Centre)

HORIZON-EUROHPC-JU-2023-COE

EuroHPC Joint Undertaking

1 January 2024– 31 December 2026

Grant Agreement No 101143931

Scalasca

DOI 10.5281/zenodo.7440854

- Collection of trace-based performance tools
  - Specifically designed for large-scale systems
  - Features automatic trace analyzer providing wait-state, critical-path & delay analysis
  - Supports MPI, OpenMP, POSIX threads, and hybrid MPI+OpenMP/Pthreads
  - Uses Score-P instrumentation & measurement infrastructure and CUBE analysis report infrastructure
- Available under 3-clause BSD open-source license

- Documentation & sources:
  - https://www.scalasca.org
- Contact:
  - mailto: scalasca@fz-juelich.de

trace tools scalasca

Score-P

DOI 10.5281/zenodo.10822140

- Infrastructure for instrumentation and performance measurements
- Instrumented application can be used to produce several results:
  - Call-path profiling: CUBE4 data format used for data exchange
  - Event-based tracing: OTF2 data format used for data exchange

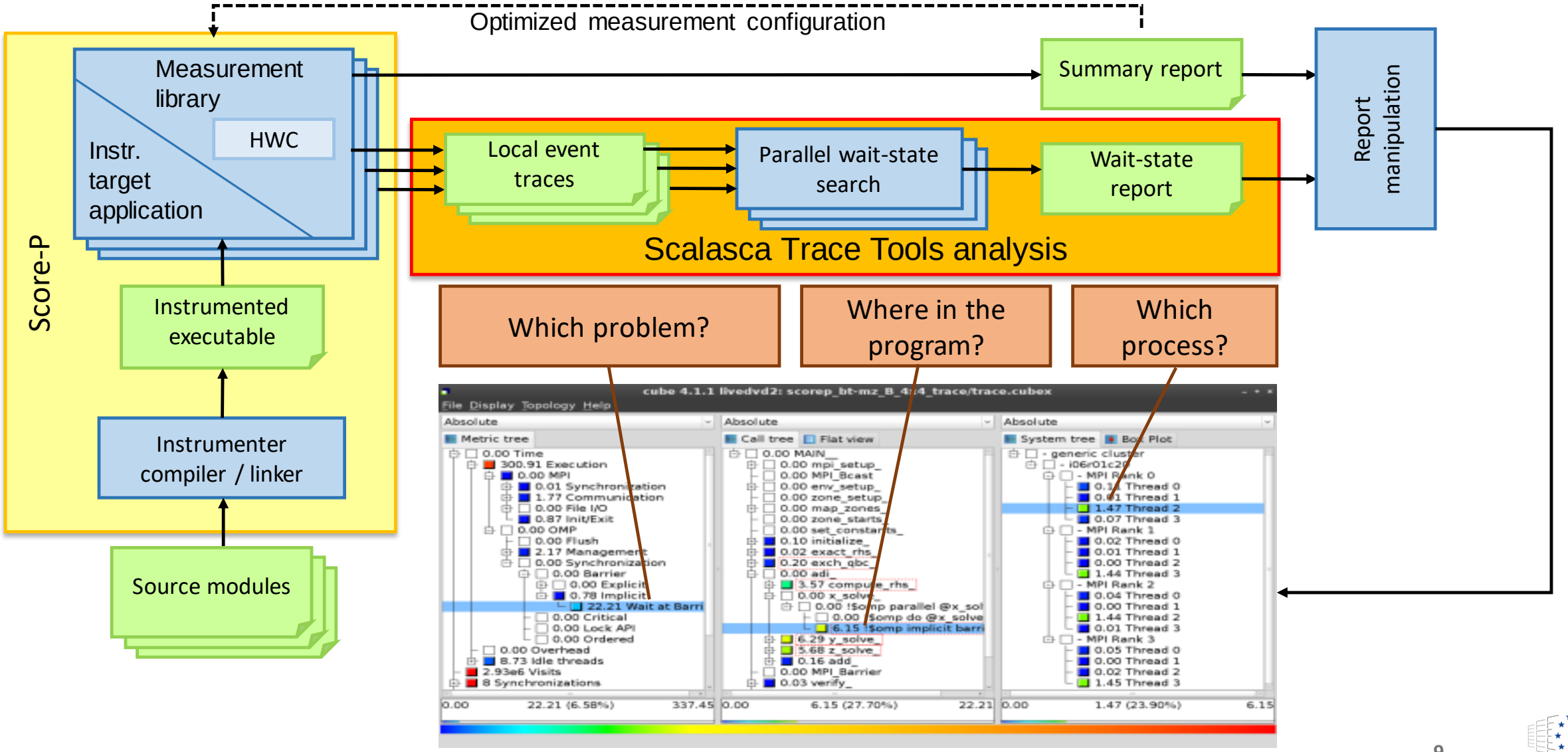
- Supported parallel paradigms:
  - Multi-process: MPI, SHMEM
  - Thread-parallel: OpenMP, Pthreads
  - Accelerator-based: CUDA, HIP, OpenCL, OpenACC, Kokkos

- Open Source; portable and scalable to all major HPC systems
- Initial project funded by BMBF
- Further developed in multiple third-party funded projects

- Documentation & sources: https://www.score-p.org

EuroHPC
Joint Undertaking

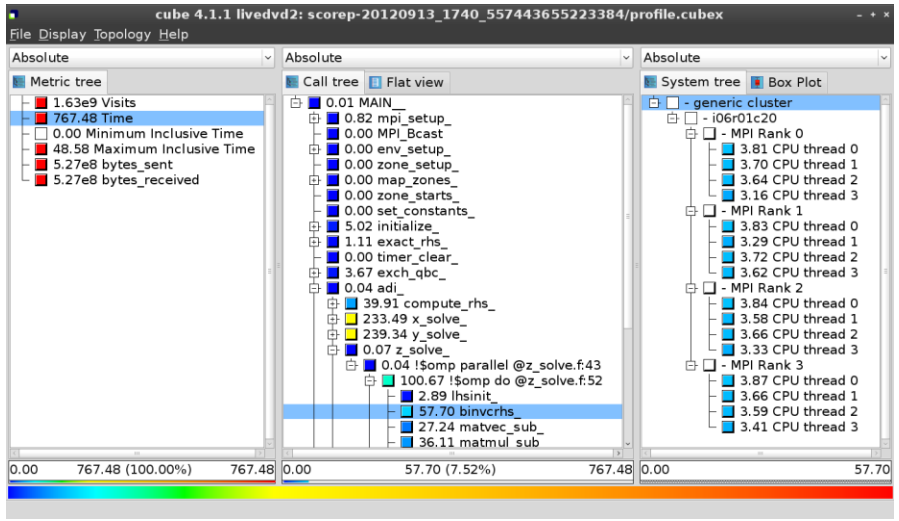
Scalasca workflow



Cube

DOI 10.5281/zenodo.8345207

- Parallel program analysis report exploration tools
  - Libraries for Cube report reading & writing
  - Algebra utilities for report processing
  - GUI for interactive analysis exploration
- Available under 3-clause BSD open-source license
- Documentation & sources:
  - http://www.scalasca.org
- User guide also part of installation:
  - <prefix>/share/doc/CubeGuide.pdf
- Contact:
  - mailto: scalasca@fz-juelich.de



Cube GUI options

- Run ***remote*** *(often convenient)*
  - start X server (e.g., Xming) locally or use mobaXterm or VNC
  - connect with X forwarding enabled
    - **-Y** may be faster but is insecure!
  - load cube module and start cube remotely

- Install & run ***local***
  - install Cube GUI locally on desktop
    - binary packages available for macOS & Windows and externally provided by OpenHPC and various Linux distributions
    - source package available for Linux, requires Qt
      - configure/build/install manually or use your favourite framework (e.g. Spack or EasyBuild)
  - copy .cubex file (or entire scorep directory) to desktop from remote system
    OR locally mount remote filesystem
  - start cube locally

https://www.**scalasca.org**/scalasca/software/cube-4.x/download.html

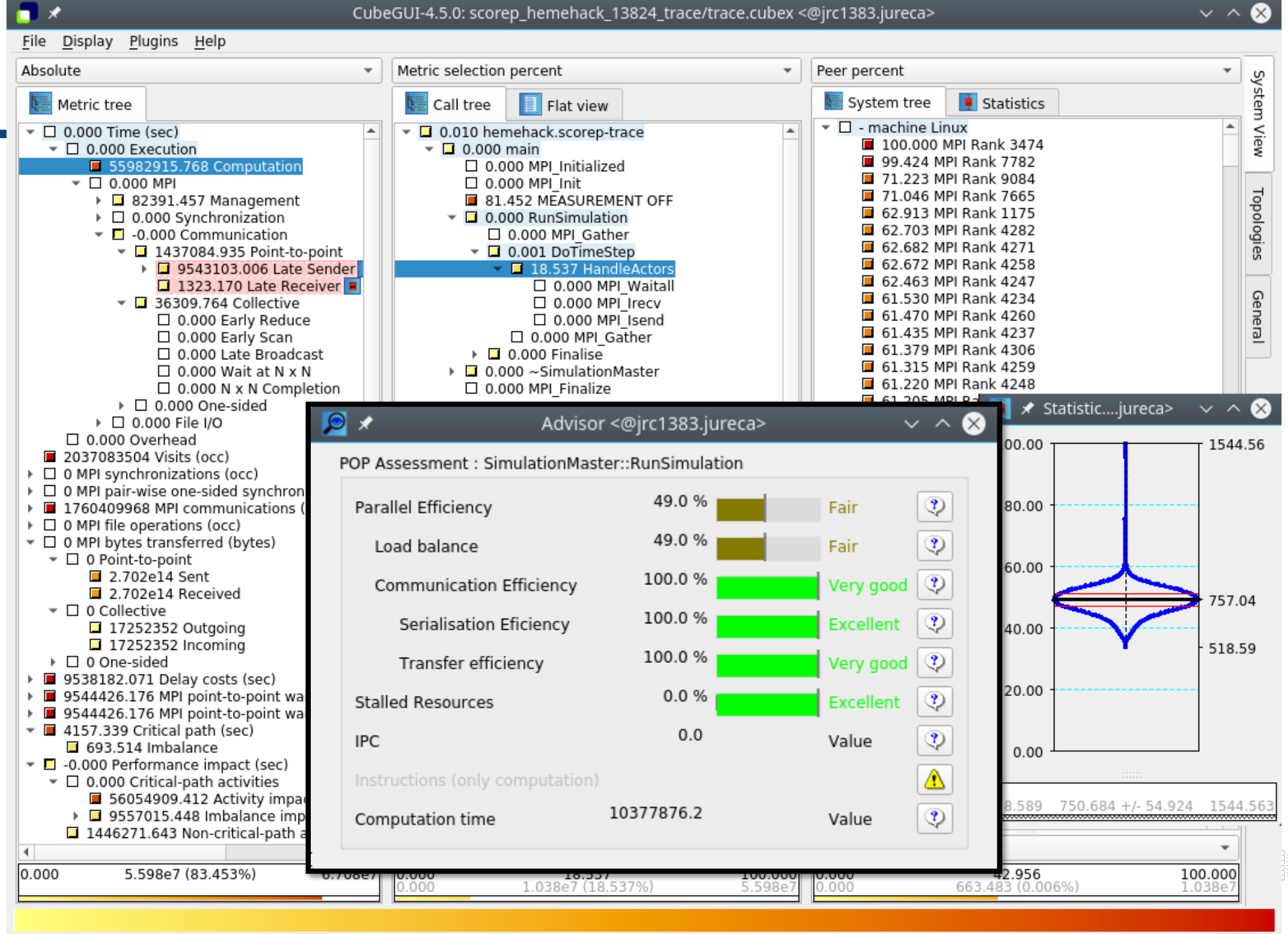
mailto: **scalasca@fz-juelich.de**

EuroHPC
Joint Undertaking

cube scalasca

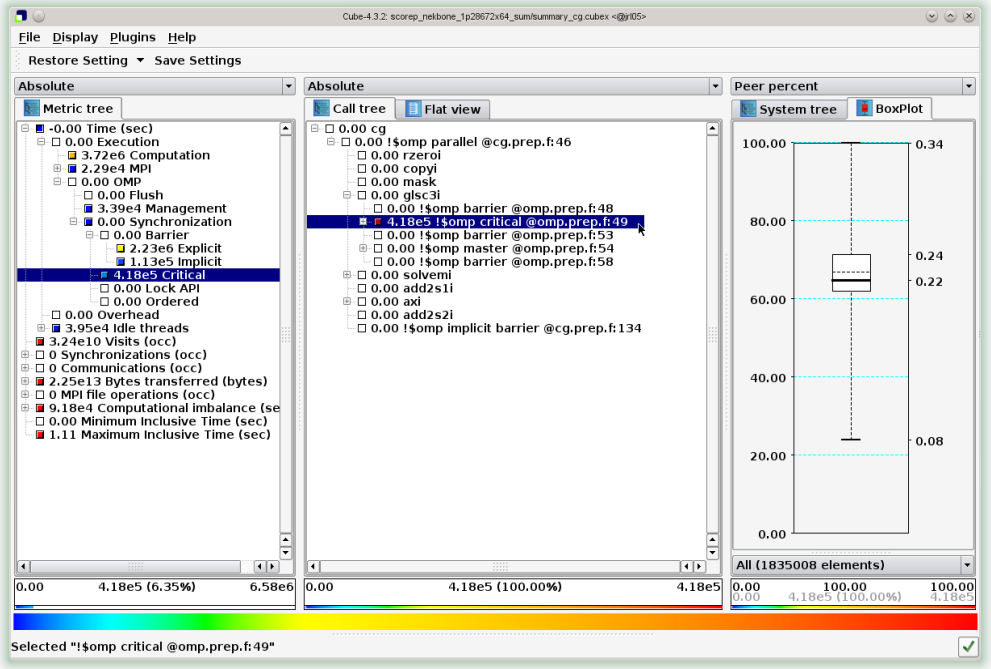
Advisor assessment

Assessment of execution efficiency factors using POP model

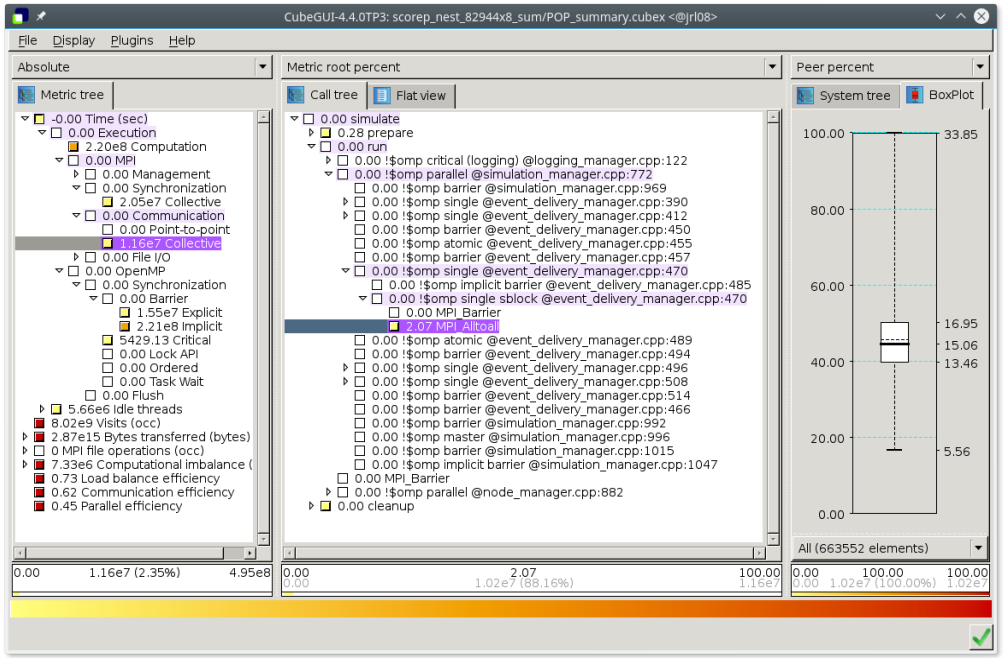


Scalasca Exascale Readiness

- Largest experiment
  - Application: Nekbone
  - System: *JUQUEEN* IBM BG/Q
  - 28,672 x 64 = 1,835,008 threads



- Largest experiment by user
  - Application: NEST
  - System: *K* computer
  - 82,944 x 8 = 663,552 threads



EuroHPC Joint Undertaking

Using Scalasca/Score-P/CUBE

Brian Wylie (Jülich Supercomputing Centre)

HORIZON-EUROHPC-JU-2023-COE

EuroHPC Joint Undertaking

1 January 2024– 31 December 2026

Grant Agreement No 101143931

Alternate approach

- Scalasca/Score-P/CUBE analysis based on sets of execution call-paths (call-graph subtrees) rather than execution time intervals
  - generally all instances aggregated together (automatically)
    - facilitates scalability,
    - but therefore can't readily distinguish individual (or ranges of consecutive) instances
- Comprehensive parallel execution traces are often prohibitively large
  - too many events to collect/analyse, requiring too much memory
  - may also be subject to considerable measurement distortion
- Necessary to balance content/expressiveness and cost
  - typically an iterative process

Steps

- Prepare an initial “rough” (summary) measurement
- Determine an appropriate Focus of Analysis (FOA)
- Refine instrumentation/measurement configuration for chosen FOA
  - Apply judicious filtering
  - Add manual annotations and measurement control
- Collect execution trace & summary profiles with sets of HWCs
- Use CUBE Advisor to acquire efficiency metrics for FOA
- Explore other metrics as directed by efficiency metrics

Determining a Focus of Analysis (FOA)

- Default of “main” will include one-off initialization (MPI_Init) and finalization (MPI_Finalize) as well as initial file reading, etc.
  - Fine if these are only small proportions of overall execution
- Cleaner if a specific call-path can be identified which avoids the one-off parts amortised in long production runs
  - where solver/timestep/iterations occur
    - expected to be (largely) homogeneous, representative of typically longer executions
  - if necessary, multiple call-paths can be combined
    - however, should avoid disjoint call-paths
  - may be annotated as a specific Score-P region for convenience
  - allows shorter execution measurements to still be representative
    - proportionally reduces size of execution traces

Familiarity

- Application developers likely know the most relevant parts of their code to focus on
  - may depend on the particular context of analysis
    - comparing a new algorithm or implementation
    - addressing file I/O, etc.
- Generally, analysts won't know and will need to determine this
  - perhaps based on guidance from application developers

One or more FOA

- Often one FOA is sufficient
  - provided it is a good representation of the dominant parallel execution phase
  - simplifies subsequent analyses
- but sometimes beneficial to have several
  - perhaps to include initialization and/or file reading/writing as comparison
  - where performance/scaling characteristics are very different
- may therefore want/need to explore/evaluate several candidates

- Timeline visualization of an execution trace can help identify the key execution phase(s) and associated call-paths

Which comes first: profile or trace?

- Obtaining an execution trace is often prohibitively expensive
  - requiring instrumentation/measurement to be suitably configured
- Generally need initial profile to refine instrumentation/measurement
  - “scoring” of profiles provides good estimate of total trace size and data to be collected by each process/thread
    - “score” based on number of times each event encountered (i.e. number of visits)
  - allows measurement buffers to be sized appropriately
    - to avoid intermediate flushes of measurement buffers to files on disk storage
  - identifies small frequent events with disproportionate overheads that don’t add value to measurement (and may well distort it significantly)
    - these should be filtered during measurement
    - or preferably avoided when instrumenting

Value determination

- Main value is from communication and synchronization events
  - typically MPI message-passing and/or OpenMP worksharing, tasking, offload
    - but also SHMEM, pthreads, OpenACC, OpenCL, CUDA, ROCm/HIP, etc.
  - represent additional parallel execution costs
- and the regions which provide execution context for them

- Purely computational regions are relatively low value for analysis
  - might all be ignored, or retain only a few key large ones
  - often fine to retain most, provided the small frequent ones are avoided
    - equivalent to them being “inlined” into their parent regions

Example performance assessments

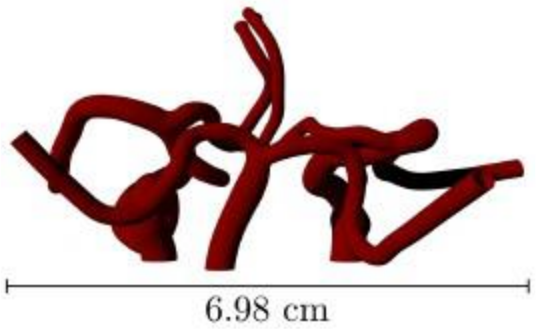
- HemeLB (MPI) on *SuperMUC-NG*
  - also previously assessed on *ARCHER* Cray XC30 & *Blue Waters* Cray XE6

- SPECFEM3D (MPI+CUDA) on *Leonardo-B*
  - MPI version previously assessed on *Joloit-Curie*

HemeLB (SuperMUC-NG)

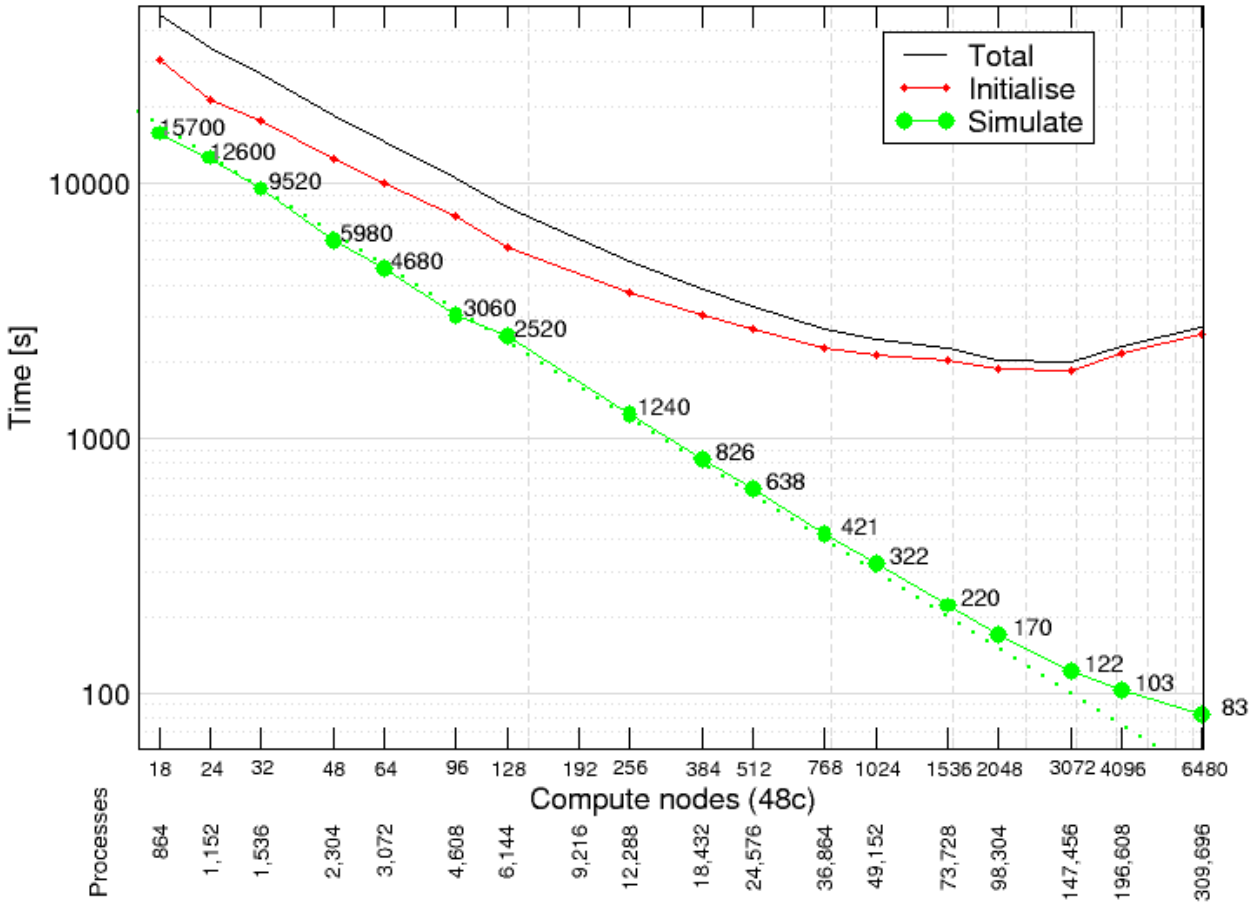
DOI 10.5281/zenodo.4105743

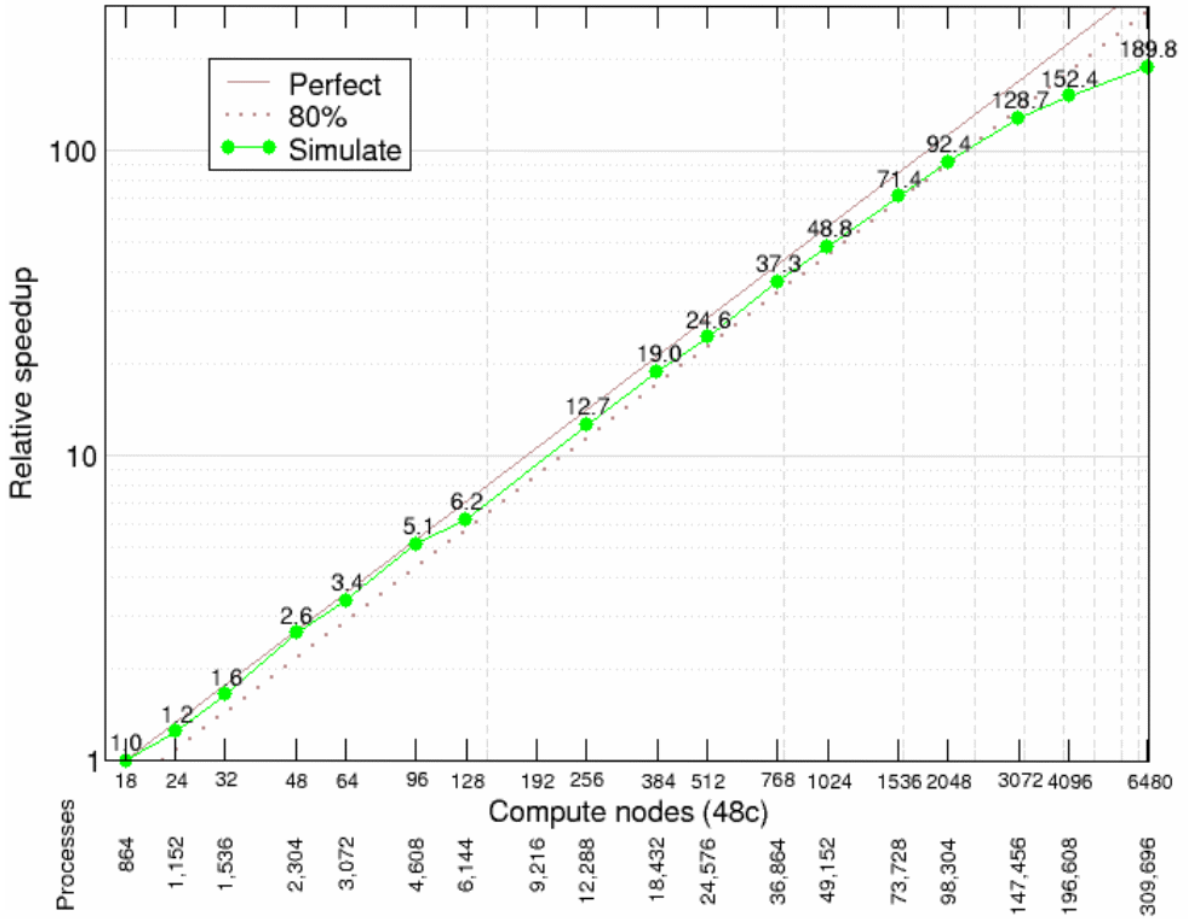
- 3D macroscopic blood flow in human arterial system developed by UC London (UK)
  - lattice-Boltzmann method tracking fluid particles on a lattice grid with complex boundary conditions
  - exascale flagship application of EU H2020 HPC Centre of Excellence for Computational Biomedicine (CompBioMed)
- 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

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]

HemeLB@SNG strong scaling efficiency

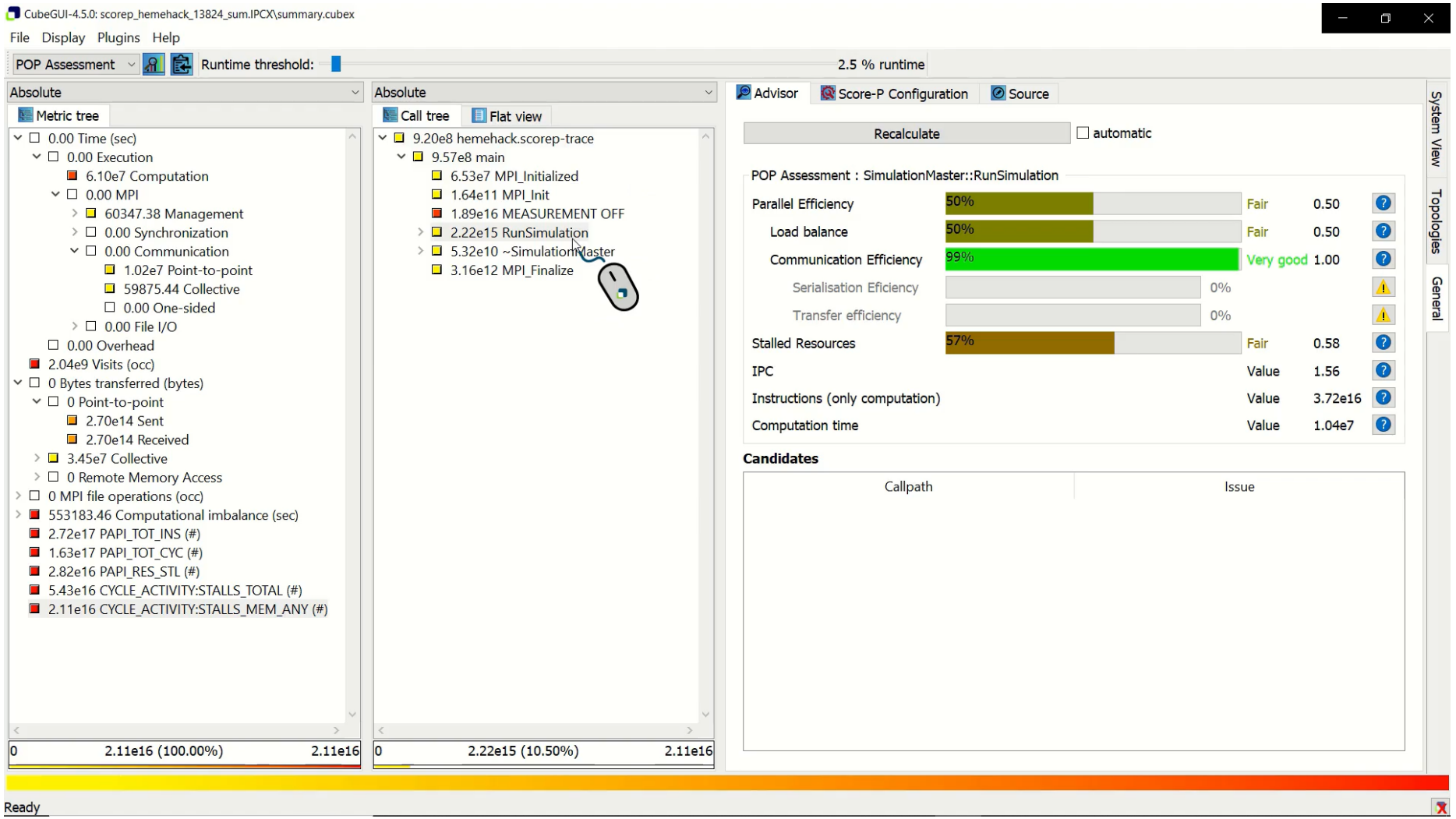
Compute nodes	24	32	48	64	96	128	192	256	384	512	768	1024	1536	2048	3072	4096	6452
Processes	**1152**	**1536**	**2304**	**3072**	**4608**	**6144**	**9216**	**12288**	**18432**	**24576**	**36864**	**49152**	**73728**	**98304**	**147456**	**196608**	**309696**
Global scaling efficiency	**0.79**	**0.79**	**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

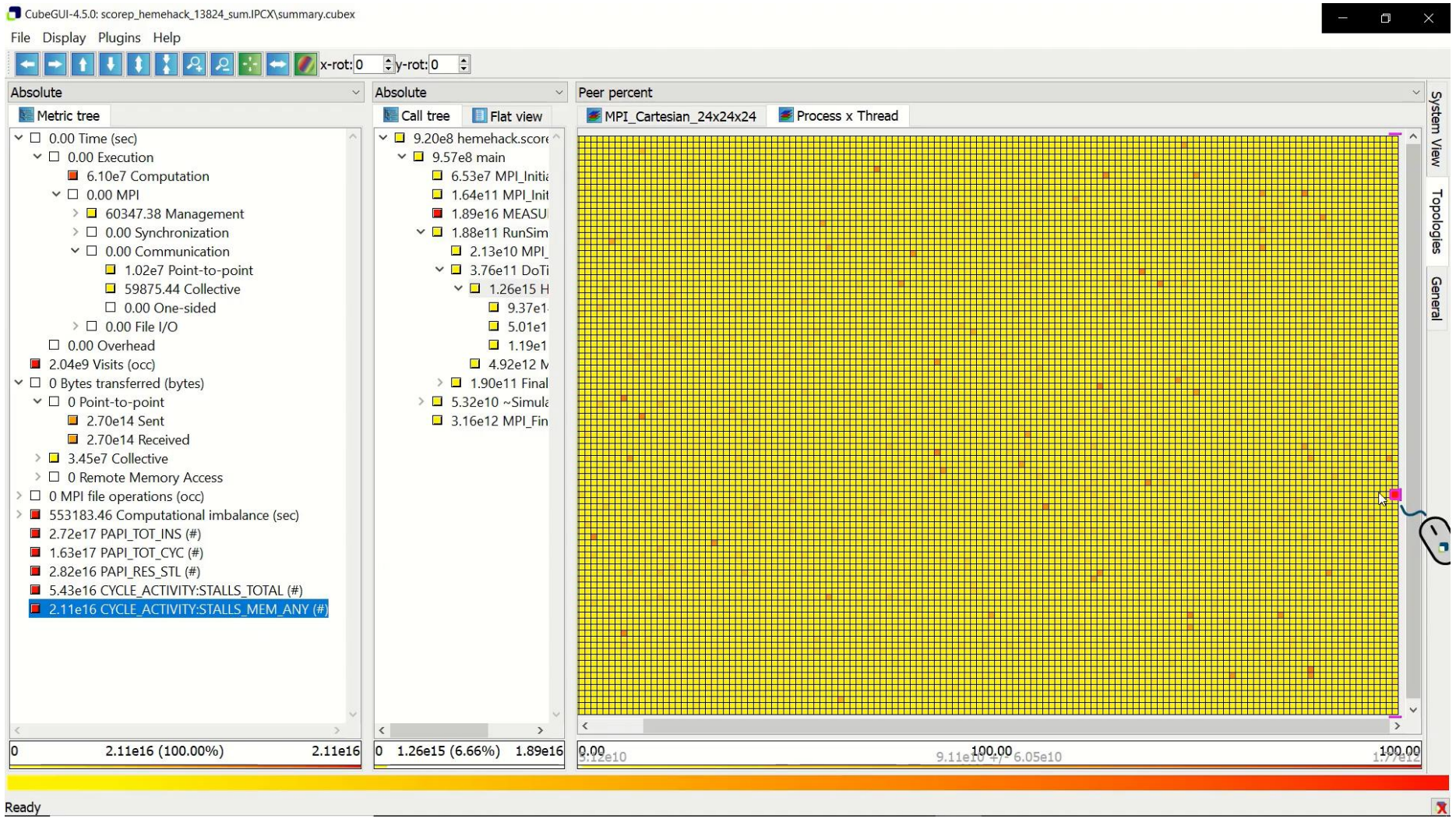
- Parallel efficiency deteriorating following Load balance efficiency
  - Communication efficiency excellent throughout
- Computation scaling (relative to 1152 processes) very good except at largest scale
  - Degradation of Instructions scaling partially compensated by improving IPC scaling

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

Advisor: POP efficiency assessment



Topological presentation



Example performance assessments

- HemeLB on *SuperMUC-NG* (MPI)
  - also previously assessed on *ARCHER* Cray XC30 & *Blue Waters* Cray XE6

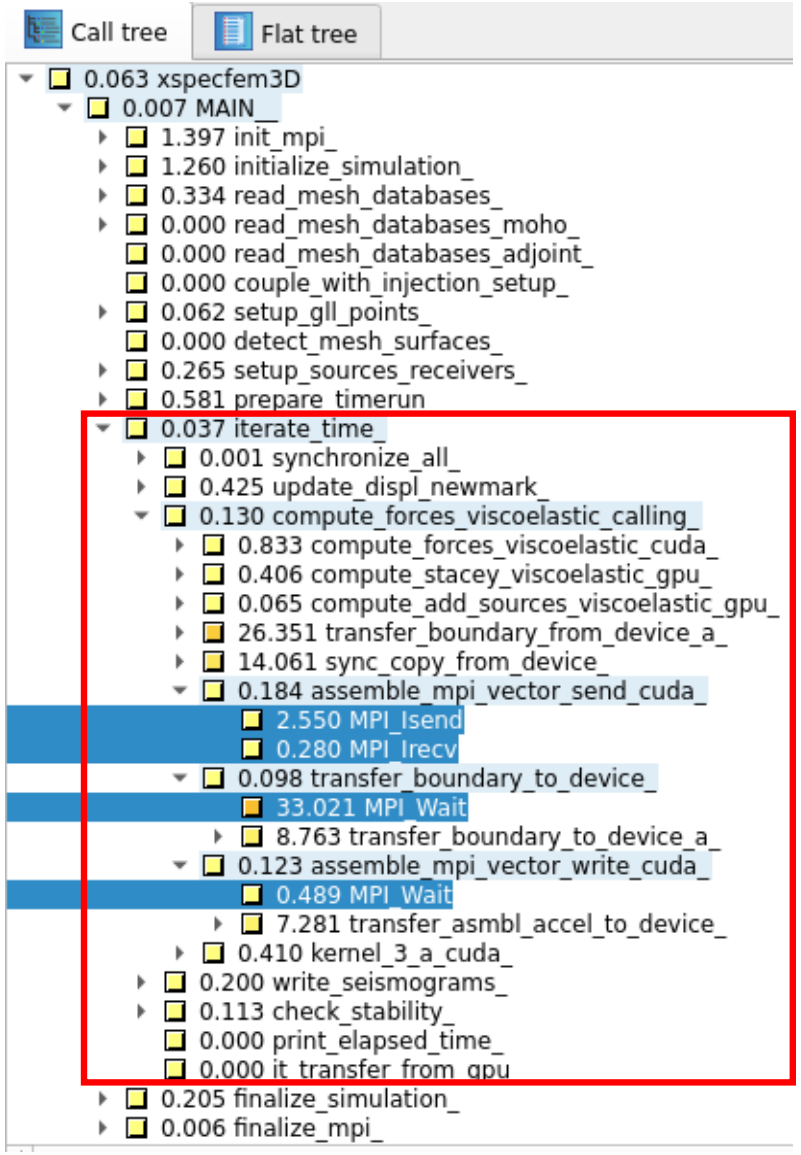
- SPECFEM3D on *Leonardo-B* (MPI+CUDA)
  - also previously assessed on *Joloit-Curie*

SPECFEM3D (Leonardo-B)

- 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*

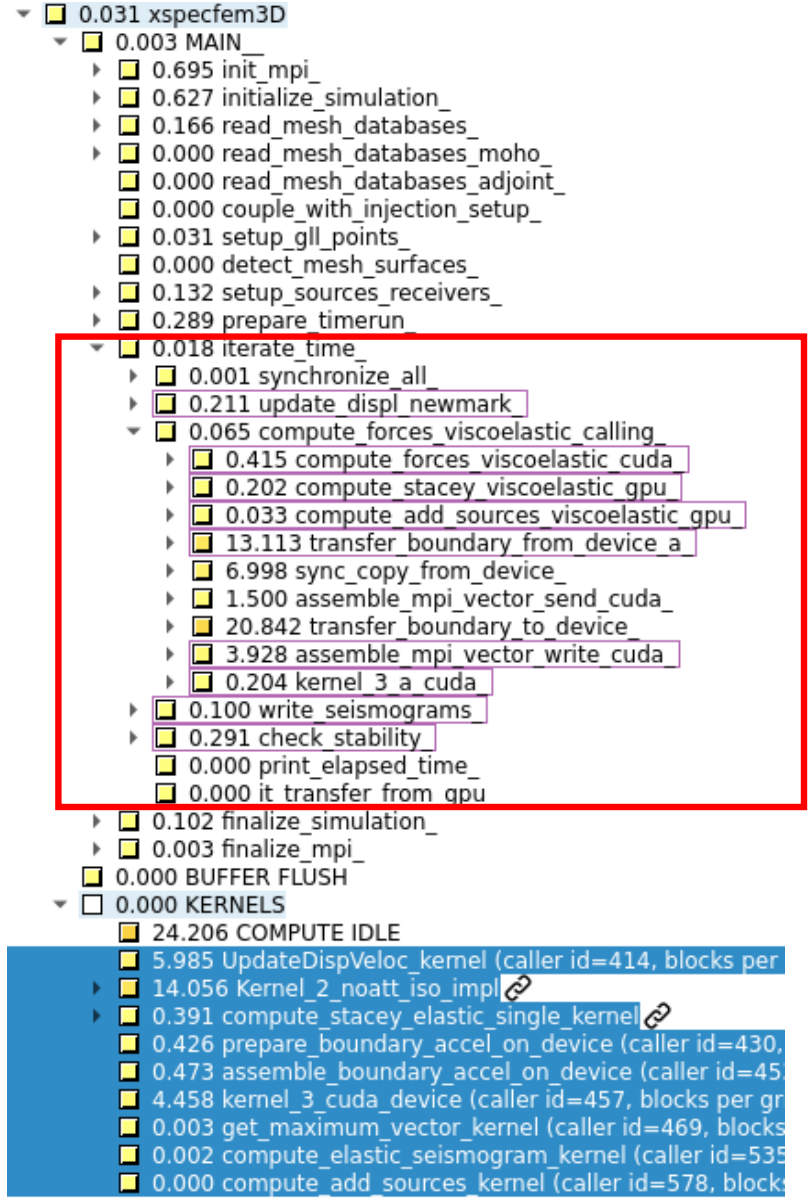
ChEESE

Execution call-tree & Focus of Analysis



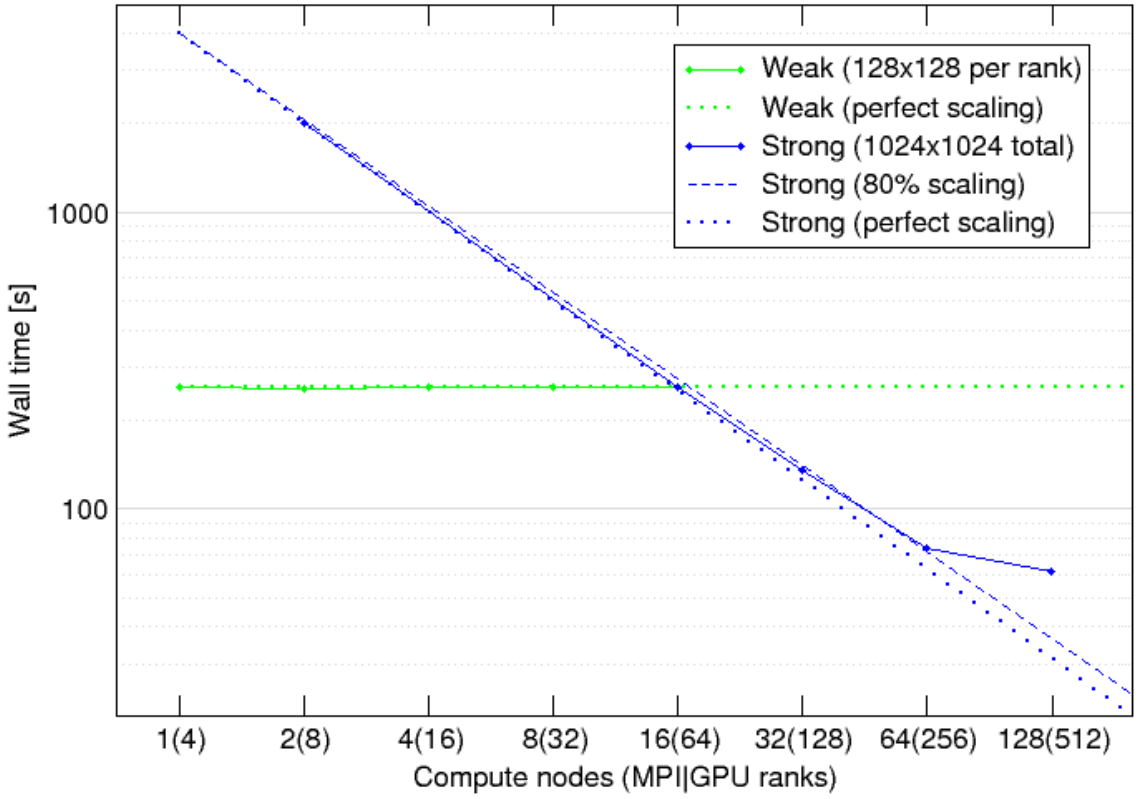
- Structure
  - setup/initialise (amortised in full run)
    - read (w/o MPI), MPI_Bcast, MPI_Reduce, etc
  - solver (*iterate_time*)
    - 48000 timesteps
      - non-blocking point-to-point communication for boundary exchange with 2D neighbours
      - data transfer to/from associated GPU device and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

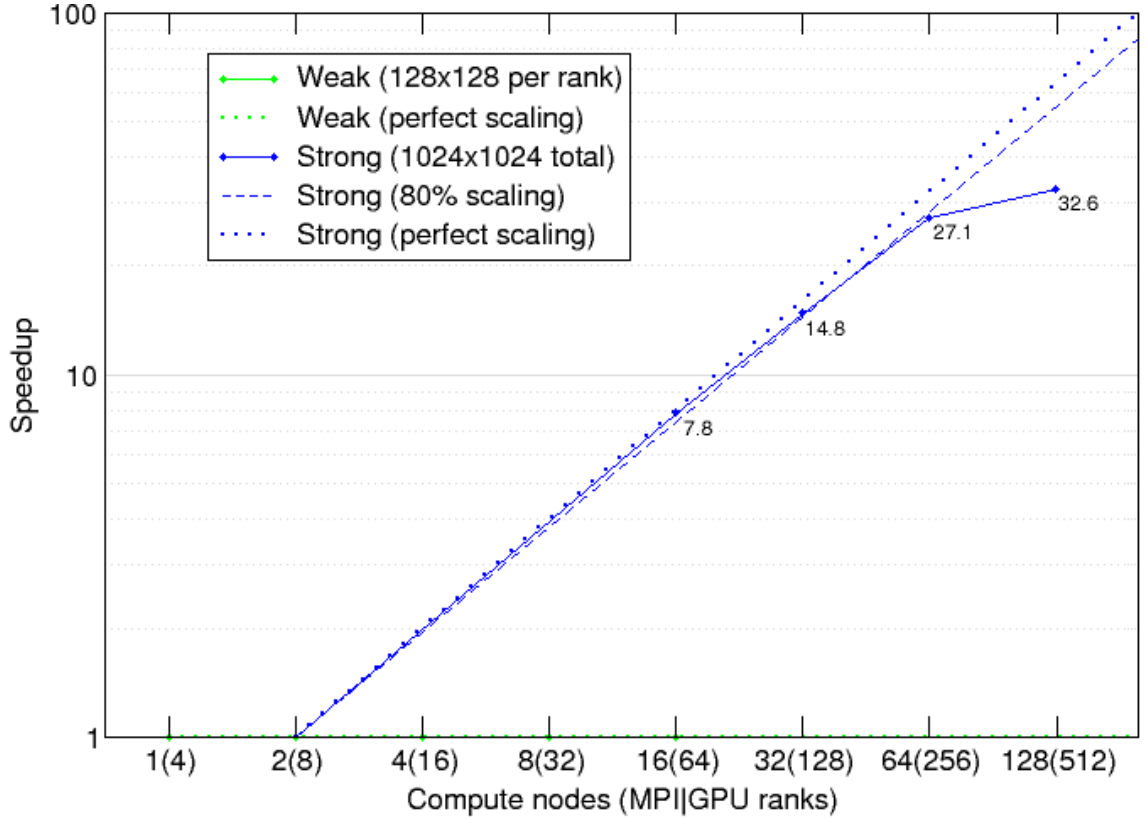
Execution call-tree & kernels



- Structure
  - solver (*iterate_time*)
    - contains all of the CUDA kernel executions
    - 48000 timesteps
      - seven of nine kernels executed by all ranks
      - characteristics oft 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*

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?

Efficiency model (strong scaling)

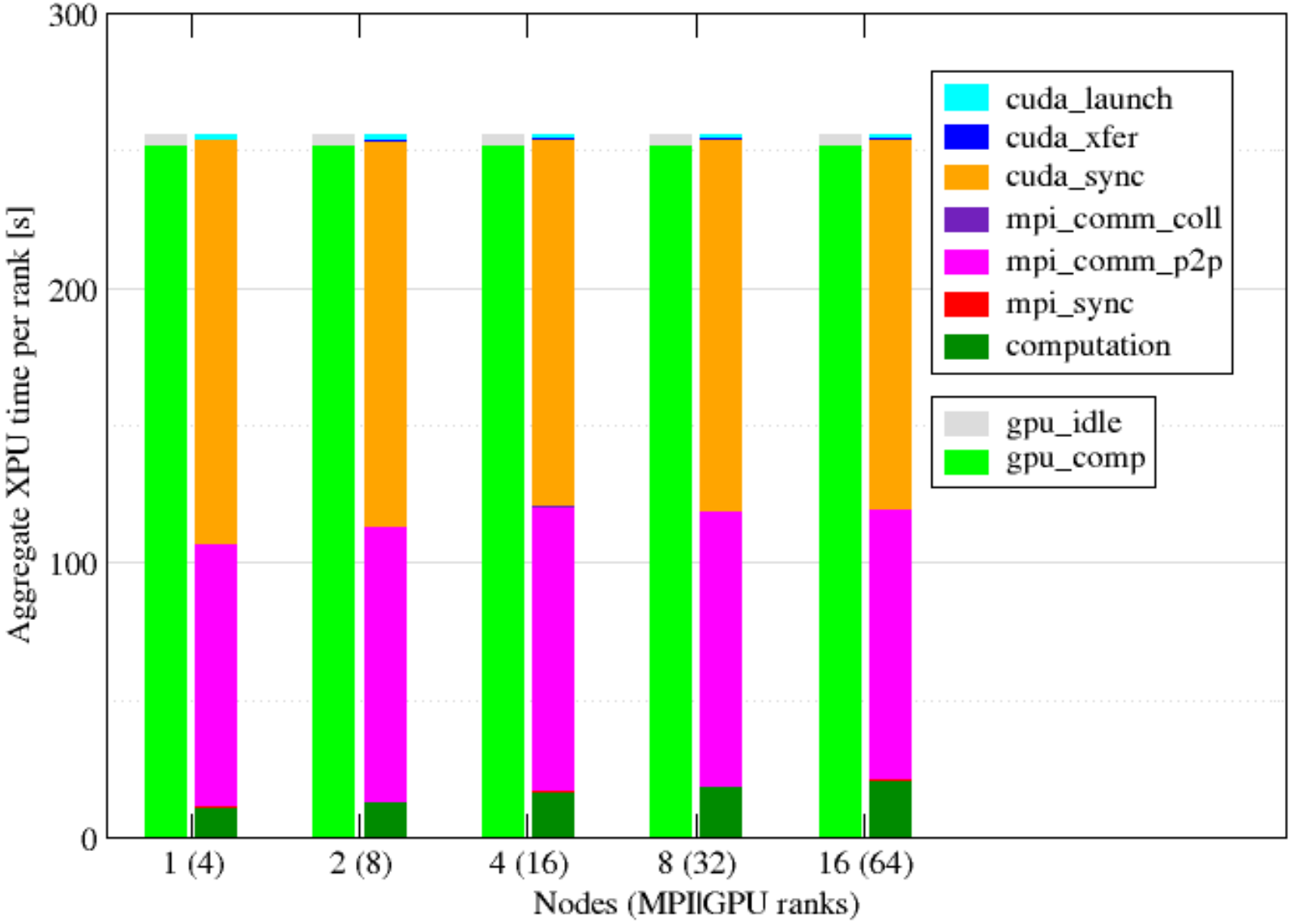
	Problem size	1024x1024	1024x1024	1024x1024	1024x1024	1024x1024
	MPI|GPU ranks	**8**	**64**	**128**	**256**	**512**
	Wall time [s]	**2001.806**	**255.948**	**135.478**	**73.846**	**61.400**
XPU	Global scaling efficiency	0.515	0.504	0.476	0.437	0.263
	- Computation time scaling	*1.000*	0.943	0.861	0.835	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

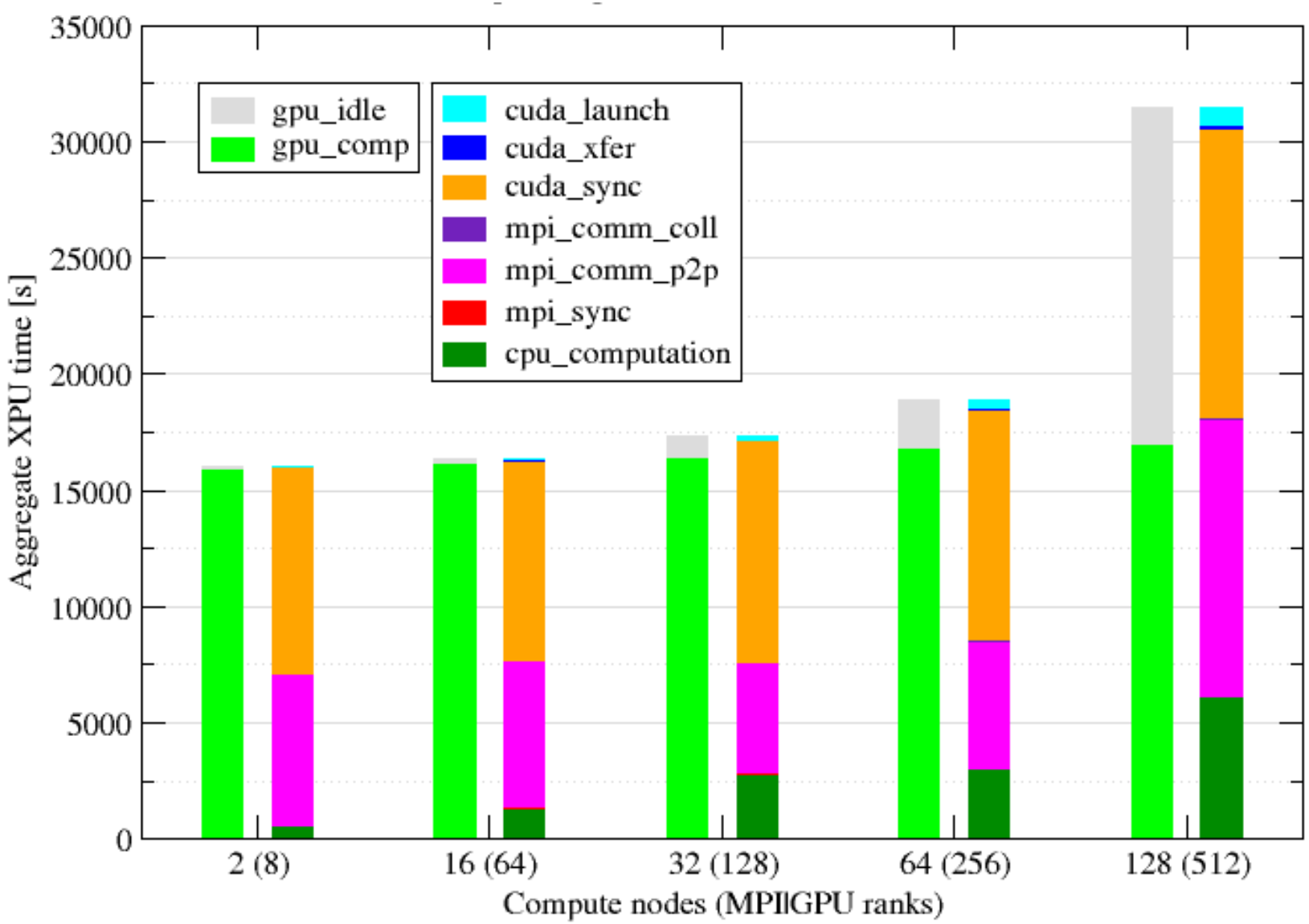
- Good GPU weak scaling efficiency to 128 GPUs (excellent load balance)
- Very poor CPU efficiency?
- Moderate XPU (GPU+CPU) efficiency?

Weak scaling (128x128 per rank)



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

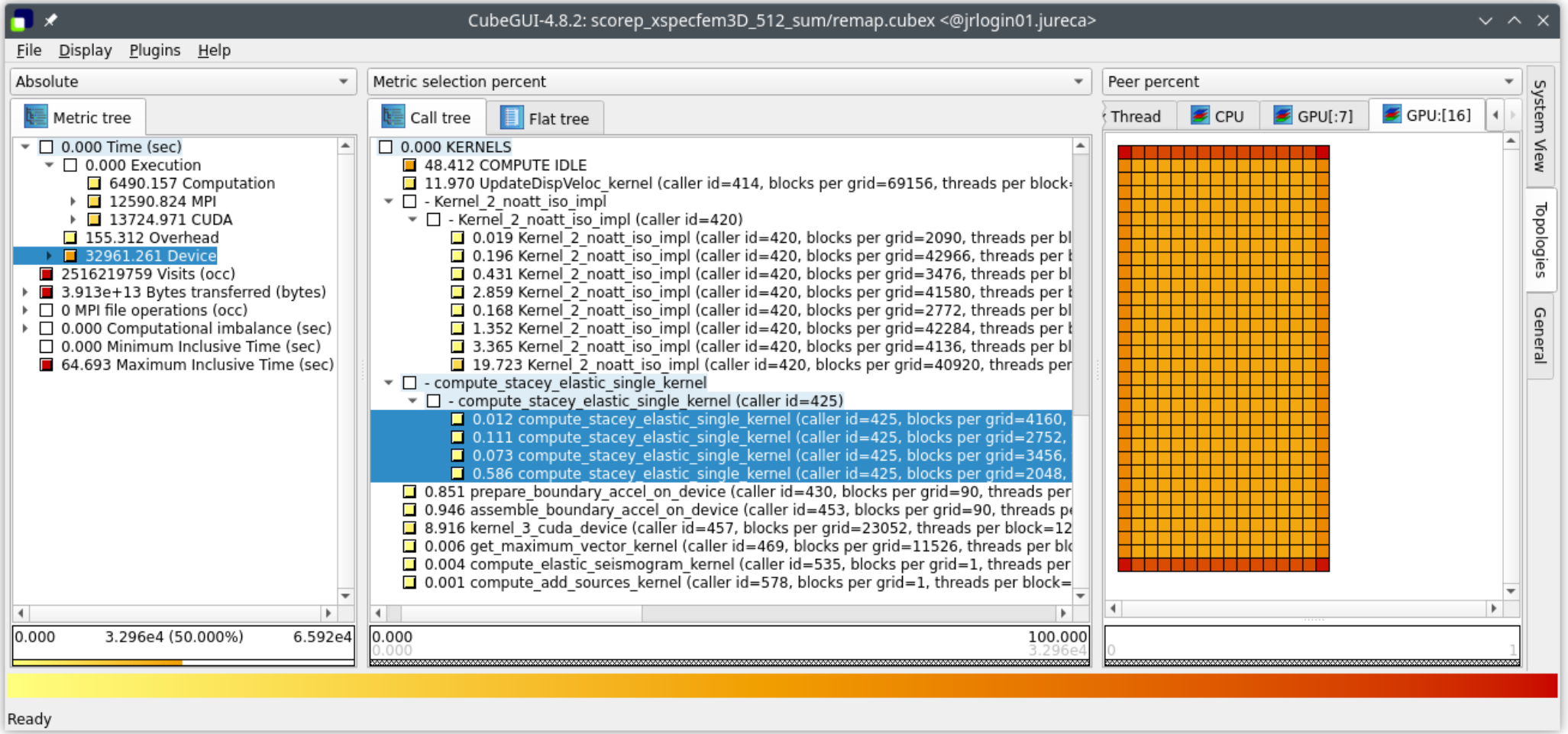
Strong scaling (1024x1024 total)



- 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

Topological inhomogeneities

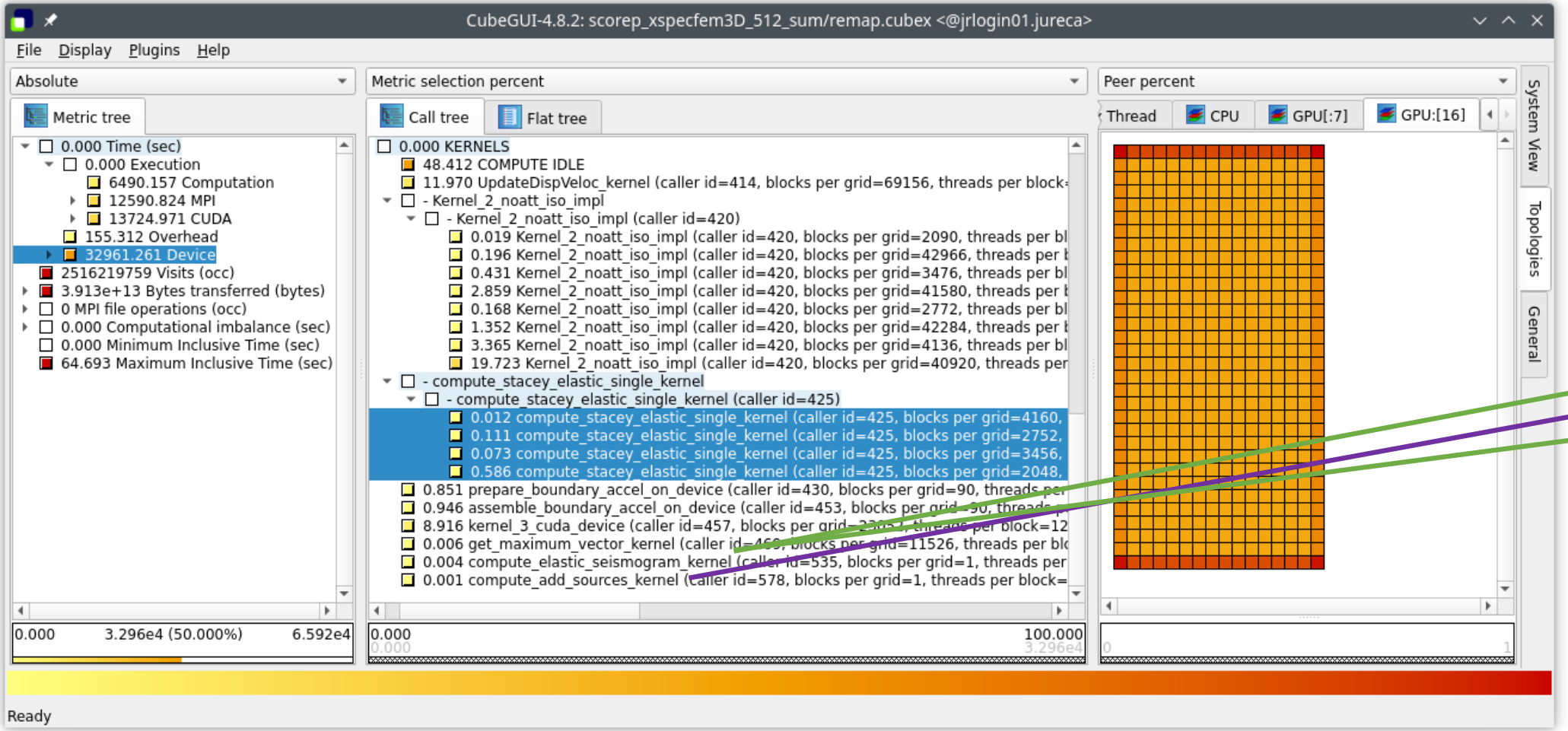
Kernel variant executed (characteristics and corresponding execution time) varies according to position in 2D grid: four corners, upper/lower & left/right edges, interior



16x32 grid

Topological inhomogeneities

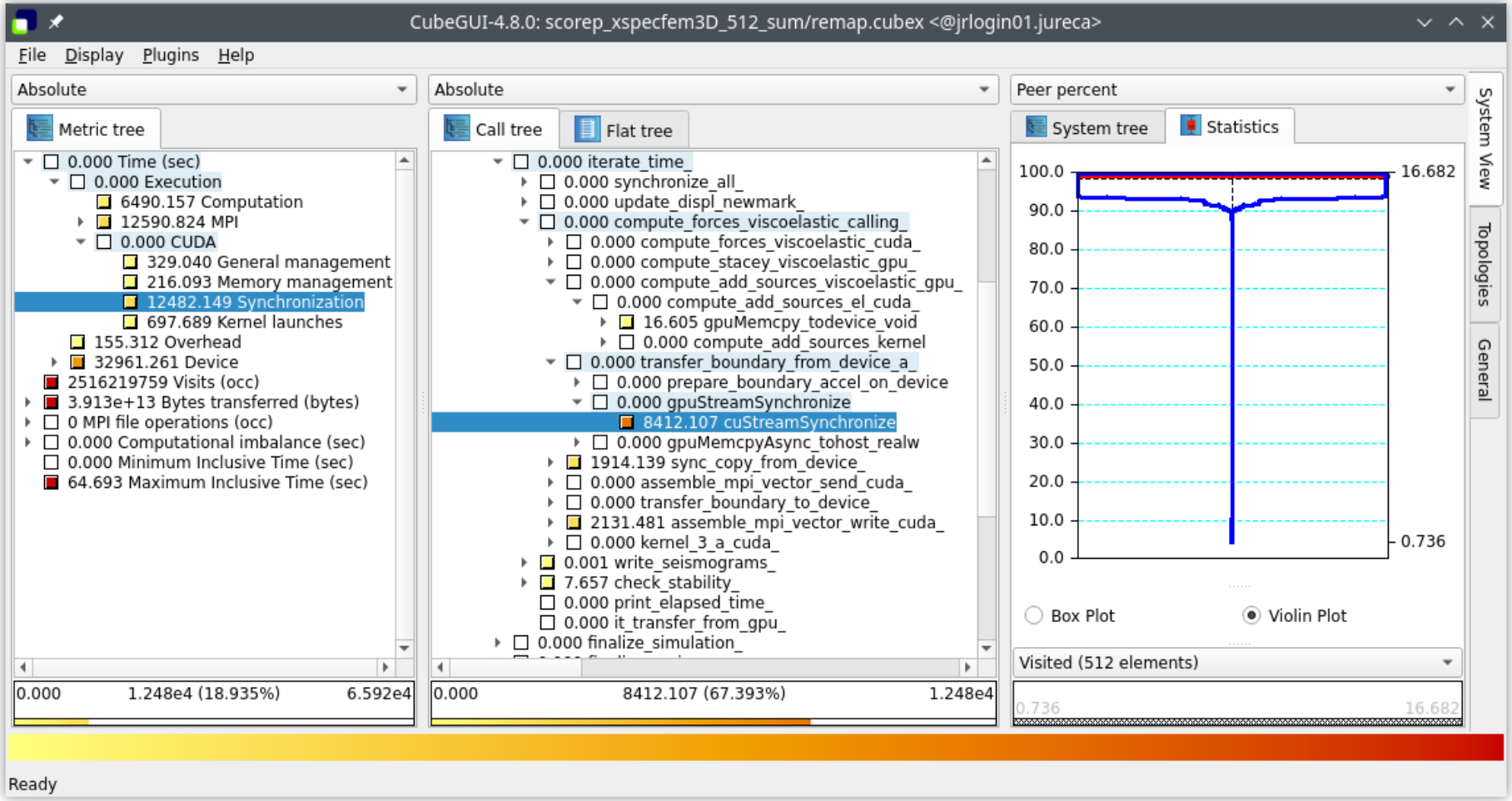
Kernel variant executed (characteristics and corresponding execution time) varies according to position in 2D grid: four corners, upper/lower & left/right edges, interior

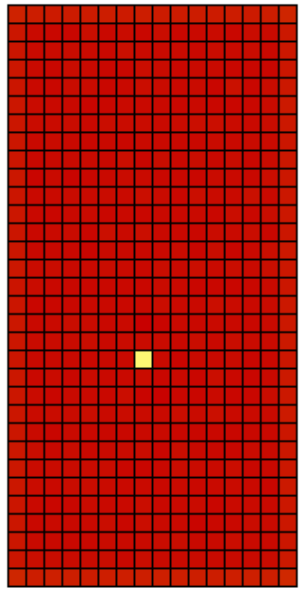


16x32 grid

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

Hands-on: Analyzing BT-MZ with Scalasca/CUBE

Anke Visser (Jülich Supercomputing Centre)

EU H2020 Centre of Excellence (CoE)

1 October 2015 – 31 March 2018
1 December 2018 – 30 November 2021

Grant Agreement No 676553 and No 824080

NPB-MZ-MPI Suite / BT-MZ

- The NAS Parallel Benchmark suite (MPI+OpenMP version)
  - Available from:

    http://www.nas.nasa.gov/Software/NPB
  - 3 benchmarks configurable for various sizes & classes
- BT-MZ (Block Triangular solver, multizone version)
  - solves discretized version of unsteady, compressible Navier-Stokes equations in three spatial dimensions
  - Implemented in 20 or so Fortran90 source modules
  - 200 time-steps on a regular 3-dimensional grid timed for benchmark figure
- Built & run on JUWELS-Cluster (dual 24-core Intel Skylake nodes)
  - Intel compiler + MPI, instrumentation by Score-P

Ex: Identify suitable Focus of Analysis

- Initial execution measurement configuration
  - Class=B benchmark executable run on a single compute node
  - 12 MPI processes each with 4 OpenMP threads

- Initial (default) summary measurement:
  - BT-MZ / scorep_bt-mz_B_12x4_sum.def / summary.cubex

- [Filtered summary measurement: scorep_bt-mz_B_12x4_sum.filt]
- [Filtered trace measurement: scorep_bt-mz_B_12x4_trace]

Ex: Determining FOA efficiency

- Instrumentation to annotate FOA
- Revised execution measurement configuration
  - Class=C benchmark executable run on a single compute node
  - 24 MPI processes each with 4 OpenMP threads
- Combined summary(HWC)+trace analysis measurements:
  - BT-MZ / bt-mz_C_24x4 / bt-mz_C_N1p24c1_trace+summary.cubex
  - BT-MZ / bt-mz_C_24x4 / bt-mz_C_N1p24c4_trace+summary.cubex
- Compare efficiencies of the two execution configurations
- Examination of additional analysis metrics

Determining parallel application execution efficiency & scaling using the POP methodology

Marta Garcia-Gasulla & Sandra Mendez (Barcelona Supercomputing Centre)
Anke Visser & Brian Wylie (Jülich Supercomputing Centre)

HORIZON-EUROHPC-JU-2023-COE

EuroHPC Joint Undertaking

1 January 2024– 31 December 2026

Grant Agreement No 101143931

Summary

POP CoE

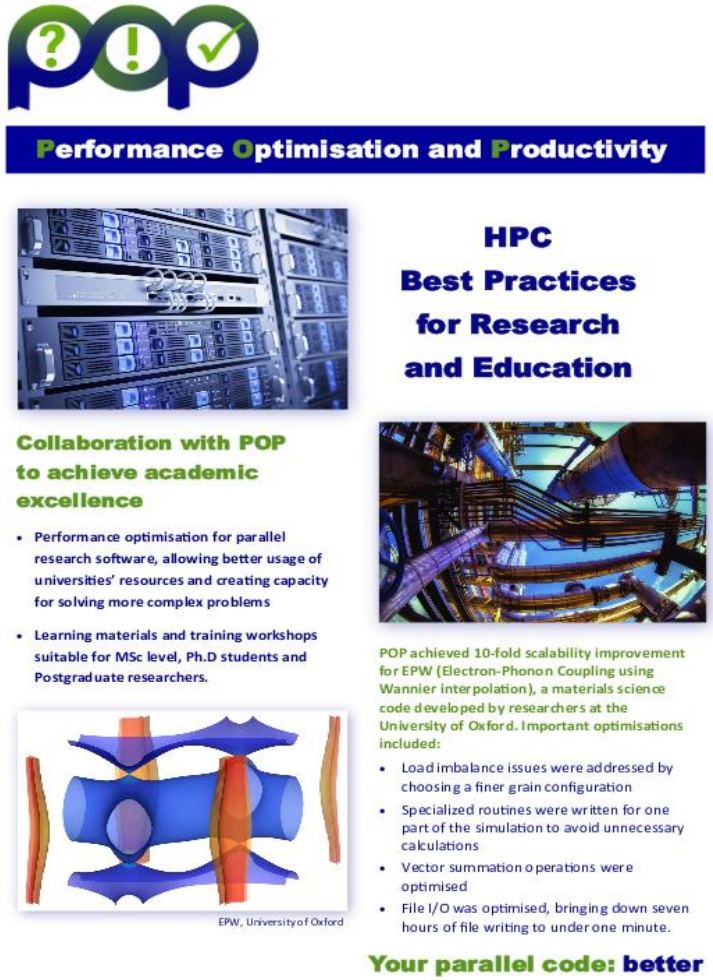
- Promotes **best practices in parallel programming**
- Encourages a systematic approach to performance optimization
- Facilitates and invests in training HPC experts

POP Performance Metrics

- Build a quantitative picture of application behaviour
- Allow quick diagnosis of performance problems in parallel codes
- Identify strategic directions for code refactoring
- So far metrics for **MPI**, **OpenMP** and **hybrid** (OpenMP + MPI) codes

POP works

- Across application domains, platforms, scales
- With European academic, government and industrial customers including code developers, code users, HPC service providers and vendors
  - To apply for a POP service go to https://pop-coe.eu/services



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.