

Analysis report examination with Cube

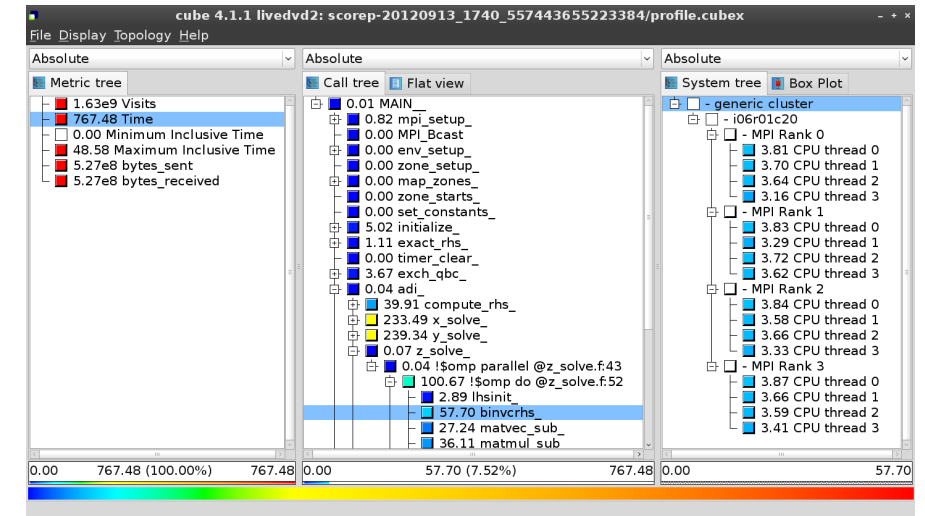
The Scalasca Team
Jülich Supercomputing Centre



Cube

CubeLib DOI 10.5281/zenodo.7737408
CubeGUI DOI 10.5281/zenodo.7737411

- Parallel program analysis report exploration tools
 - Libraries for XML+binary report reading & writing
 - Algebra utilities for report processing
 - GUI for interactive analysis exploration
 - Requires Qt \geq 5
- Originally developed as part of the Scalasca toolset
- Now available as a separate component
 - Can be installed independently of Score-P, e.g., on laptop or desktop
 - Latest release: Cube v4.8.1 (March 2023)



Note: source distribution tarballs for Linux, as well as binary packages provided for Windows & MacOS, from www.scalasca.org website in software/Cube-4x

Cube GUI (archer2)

mailto: scalasca@fz-juelich.de



- Run **remote** (often convenient)
 - start X server (e.g., Xming) locally, or use alternative such as mobaXterm
 - connect to Archer2 with X forwarding enabled
 - **-Y** may be faster but is insecure!
 - load scalasca module and start cube remotely

```
desk$ ssh -X login.archer2.ac.uk
Welcome to ARCHER2...
uan01$ module load scalasca
uan01$ cube ./scorep_sum/profile.cubex
```

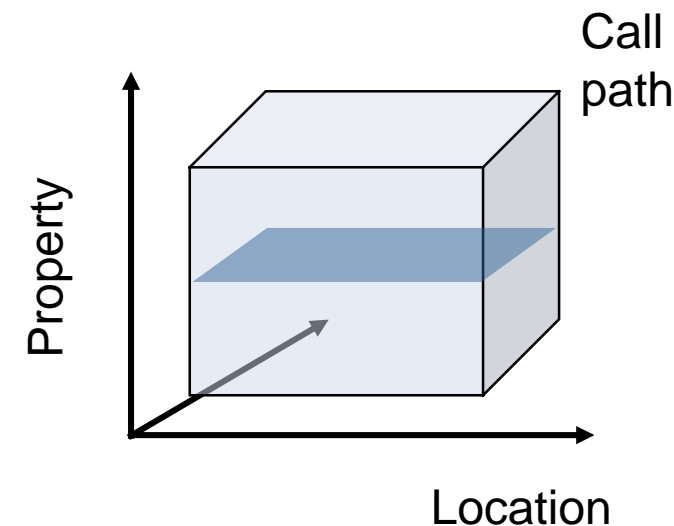
Sample measurements (CUBE files) on Archer2:
/work/y23/shared/tutorial/samples

- Install & run **local** (recommended)
 - 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

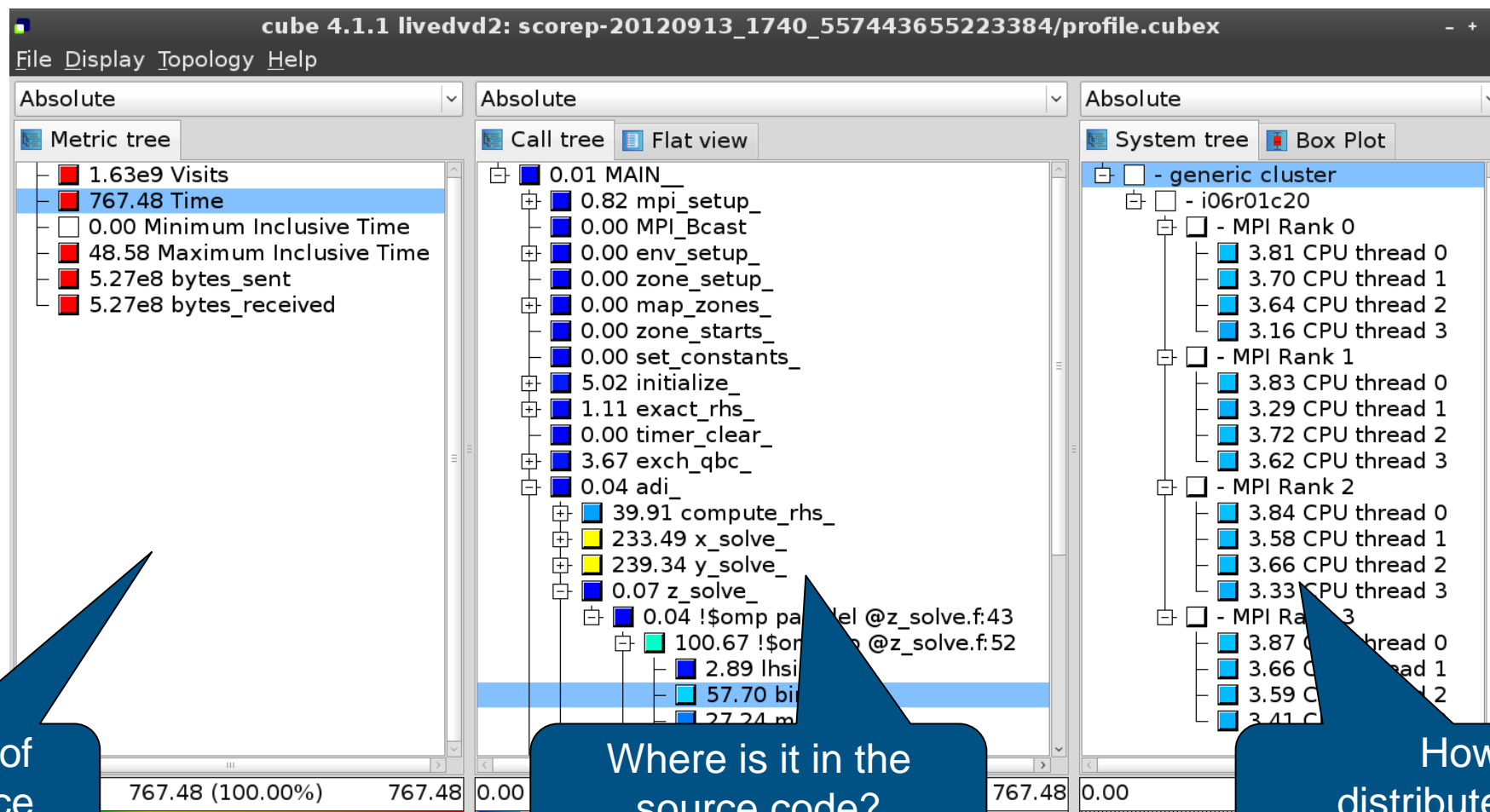
```
desk$ mkdir $HOME/mnt
desk$ sshfs [user@]remote.sys:[dir] $HOME/mnt
desk$ cd $HOME/mnt
desk$ cube ./scorep_sum/profile.cubex
```

Analysis presentation and exploration

- Representation of values (severity matrix) on three hierarchical axes
 - Performance property (metric)
 - Call path (program location)
 - System location (process/thread)
- Three coupled tree browsers
- Cube displays severities
 - *As value*: for precise comparison
 - *As colour*: for easy identification of hotspots
 - *Inclusive* value when closed & *exclusive* value when expanded
 - Customizable via display modes



Analysis presentation



What kind of performance metric?

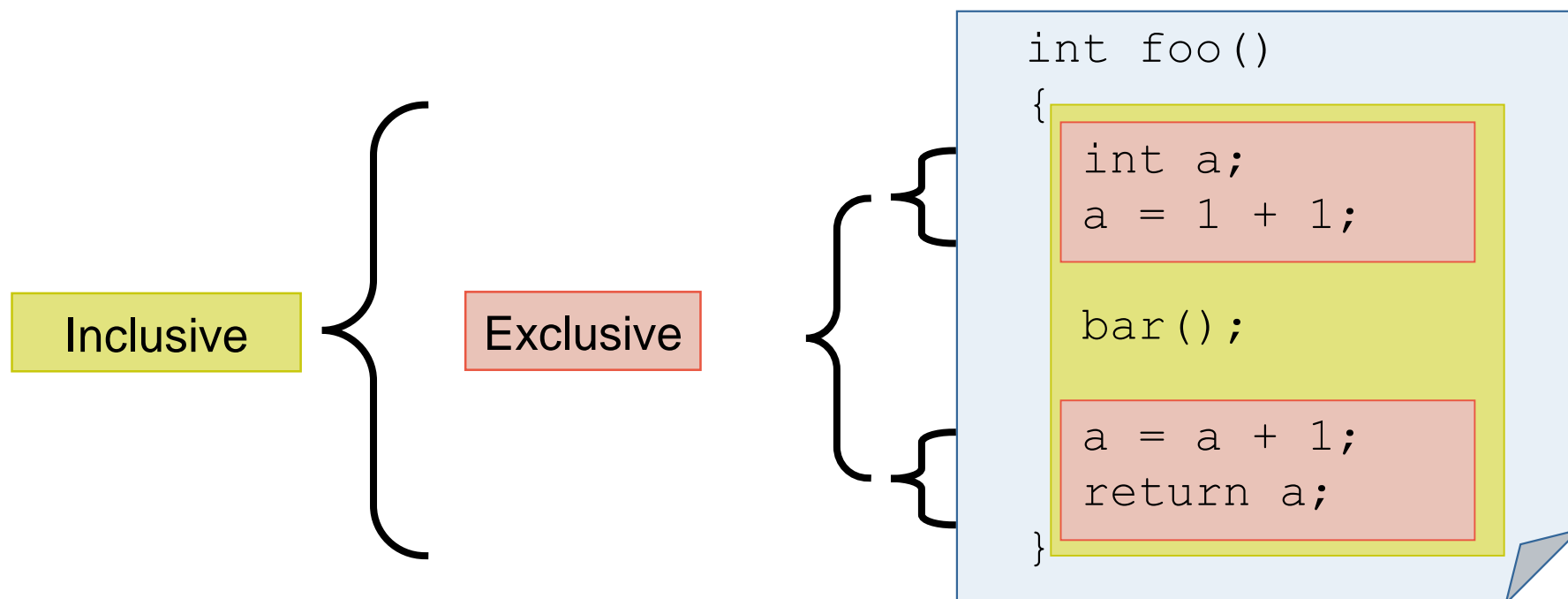
Where is it in the source code?
In what context?

How is it distributed across the processes/threads?

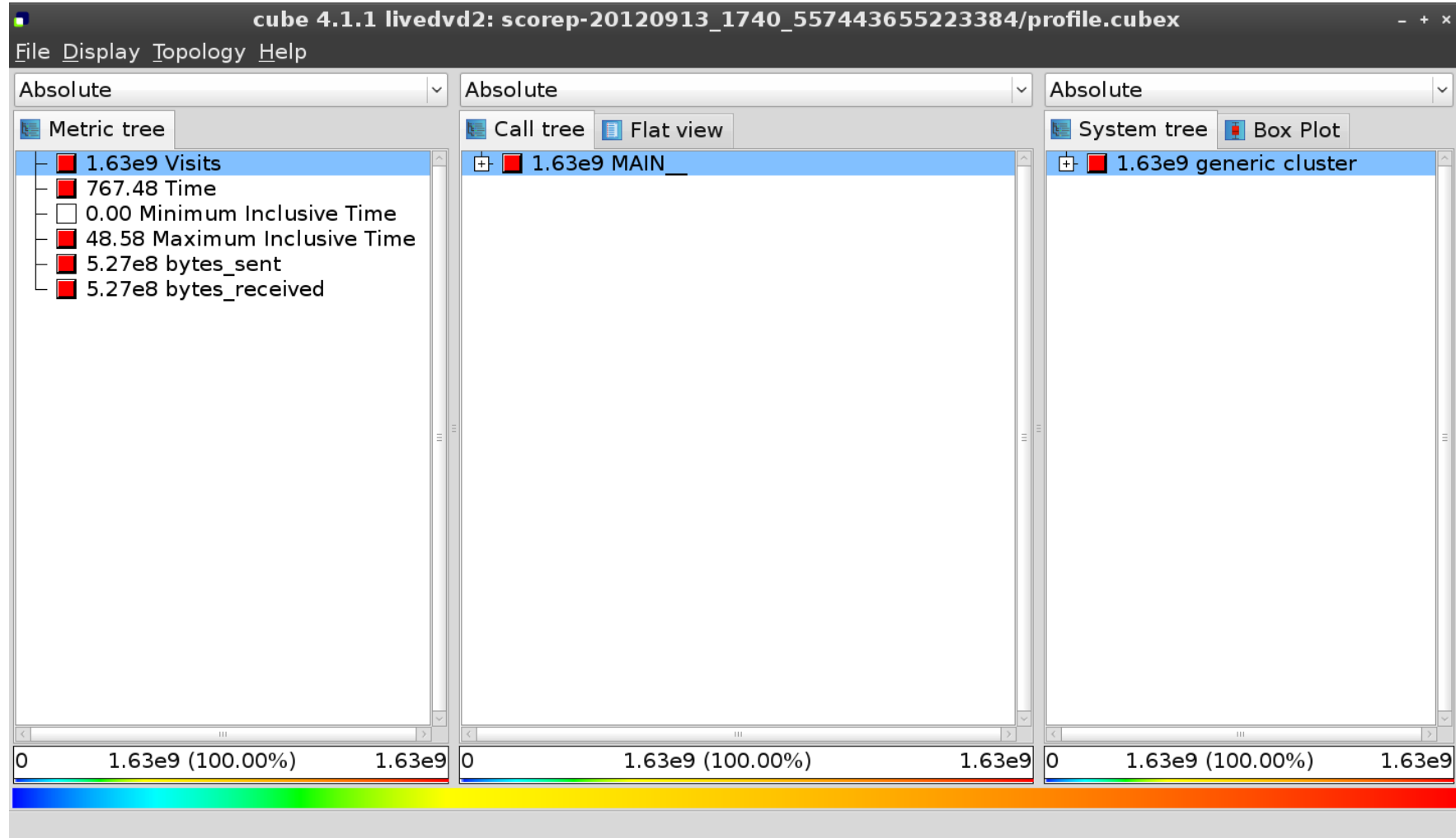
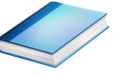
Inclusive vs. exclusive values



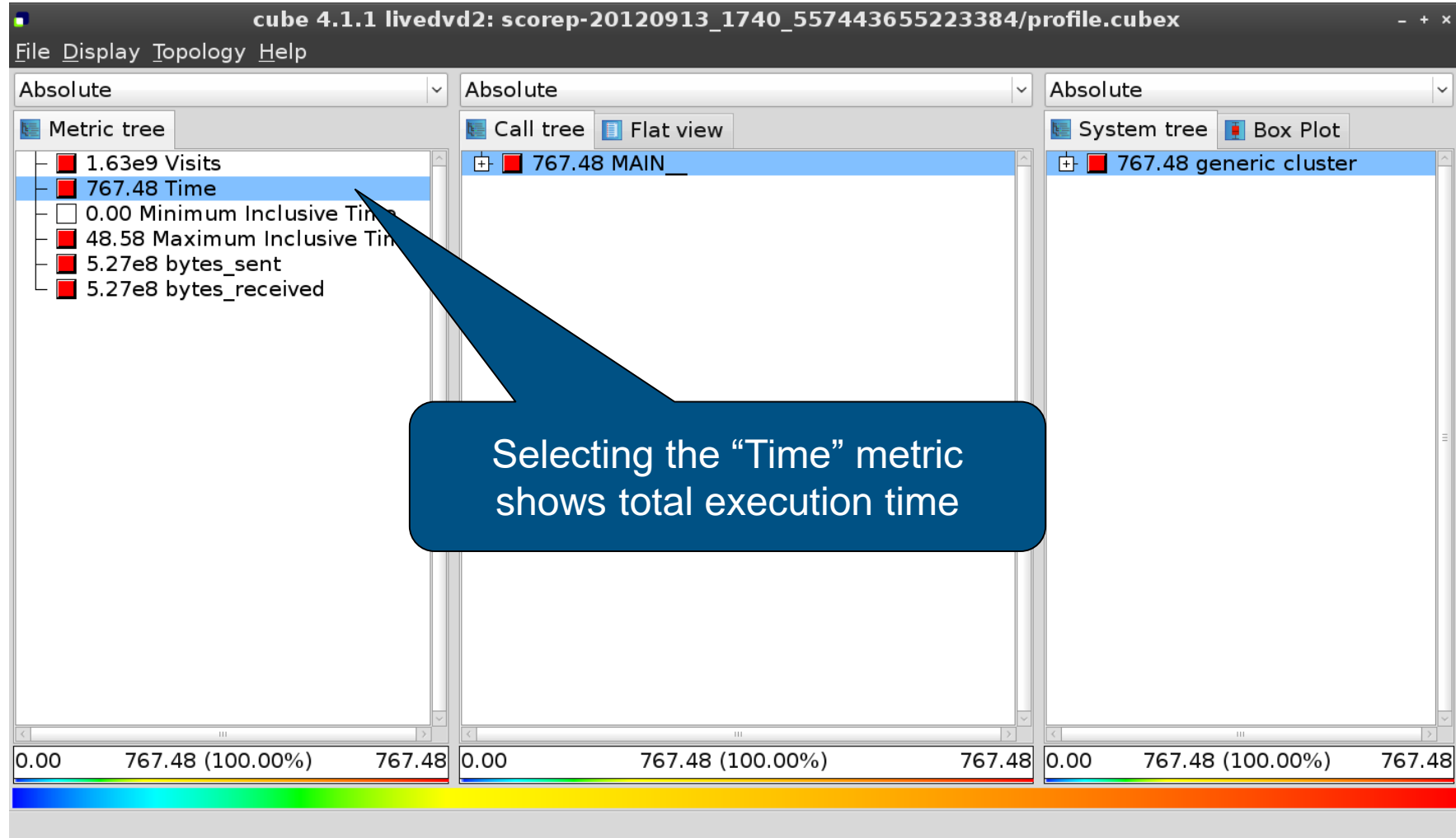
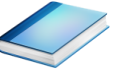
- Inclusive
 - Information of all sub-elements aggregated into single value
- Exclusive
 - Information cannot be subdivided further



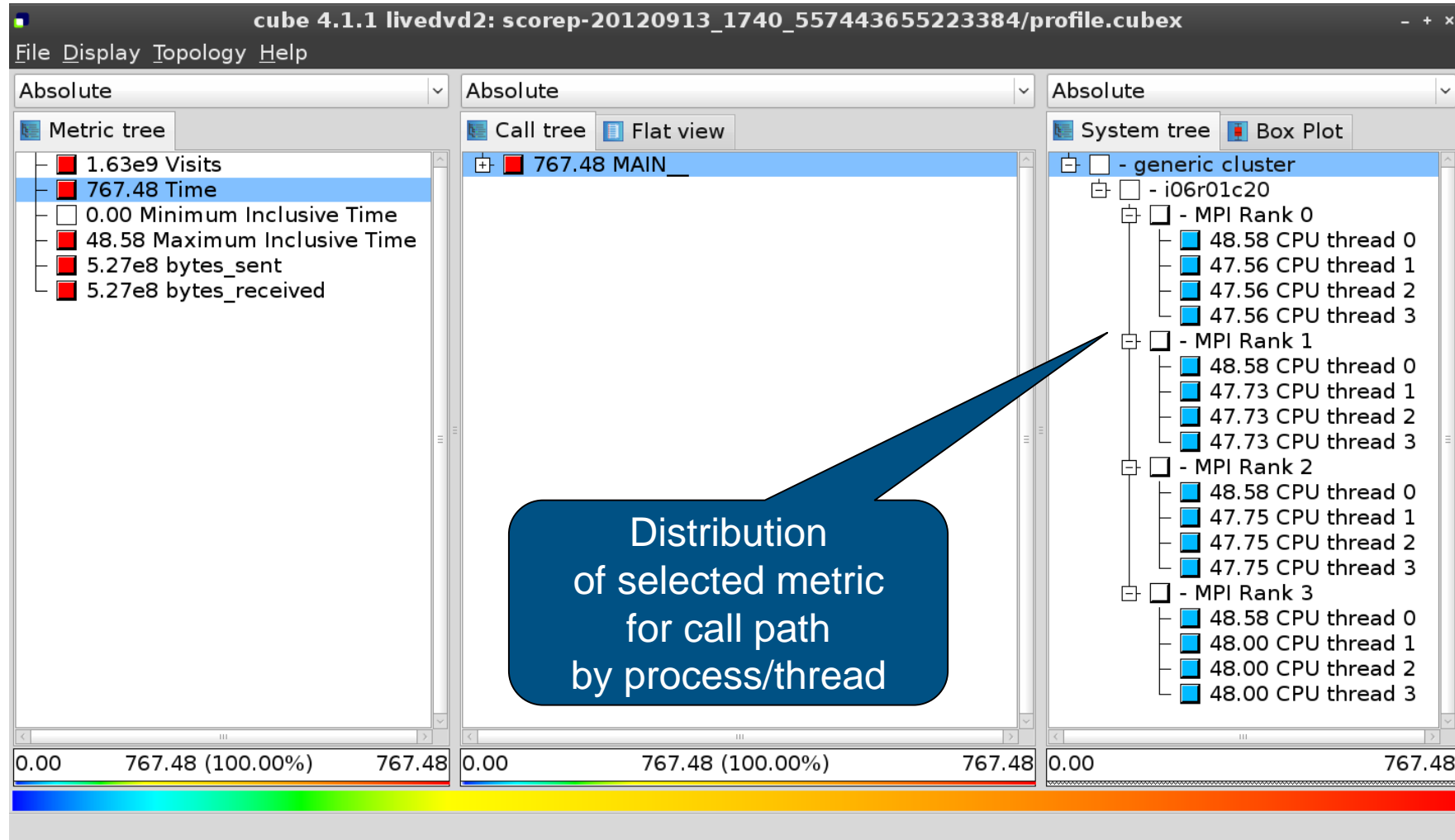
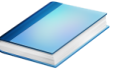
Score-P analysis report exploration (opening view)



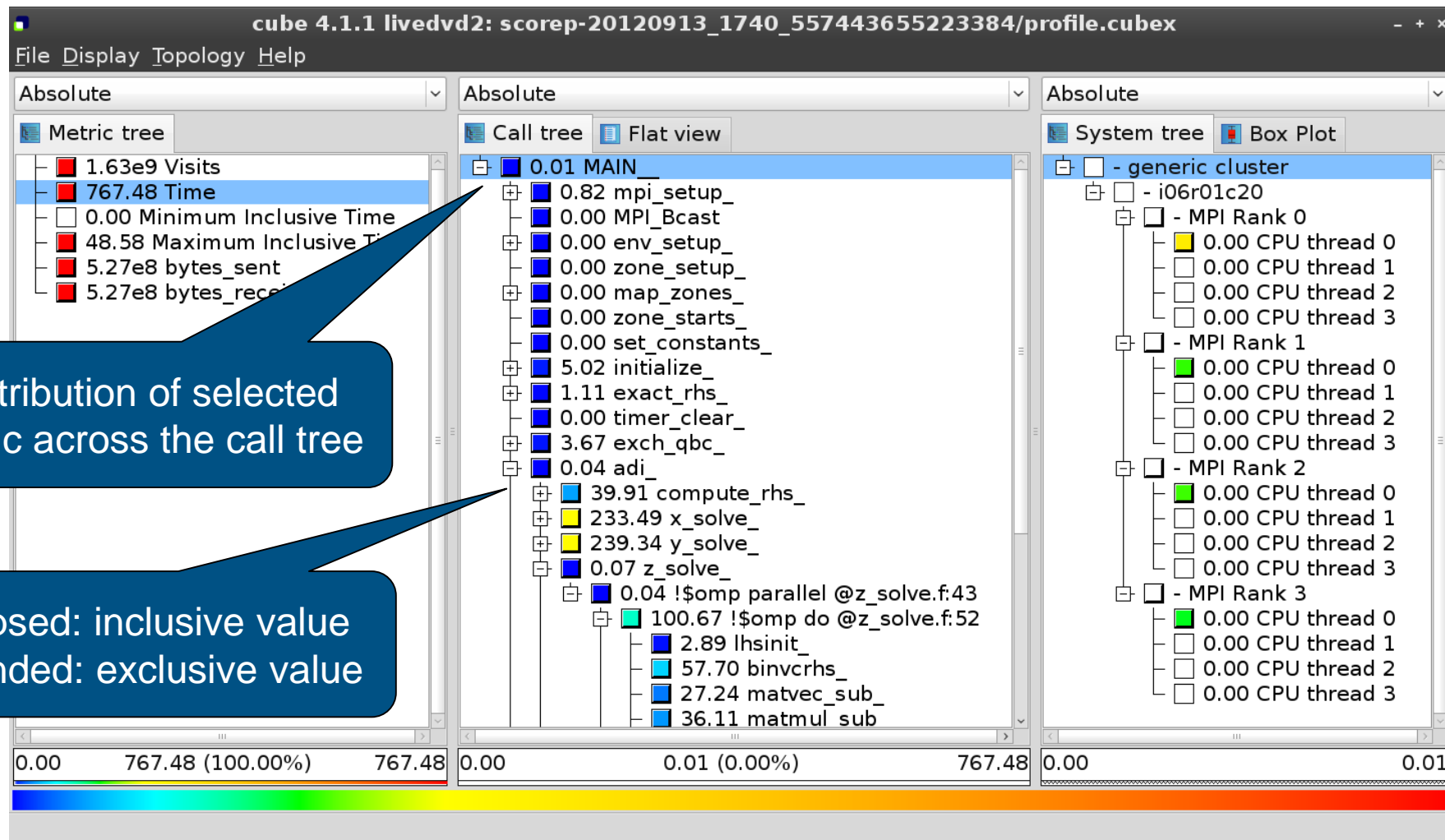
Metric selection



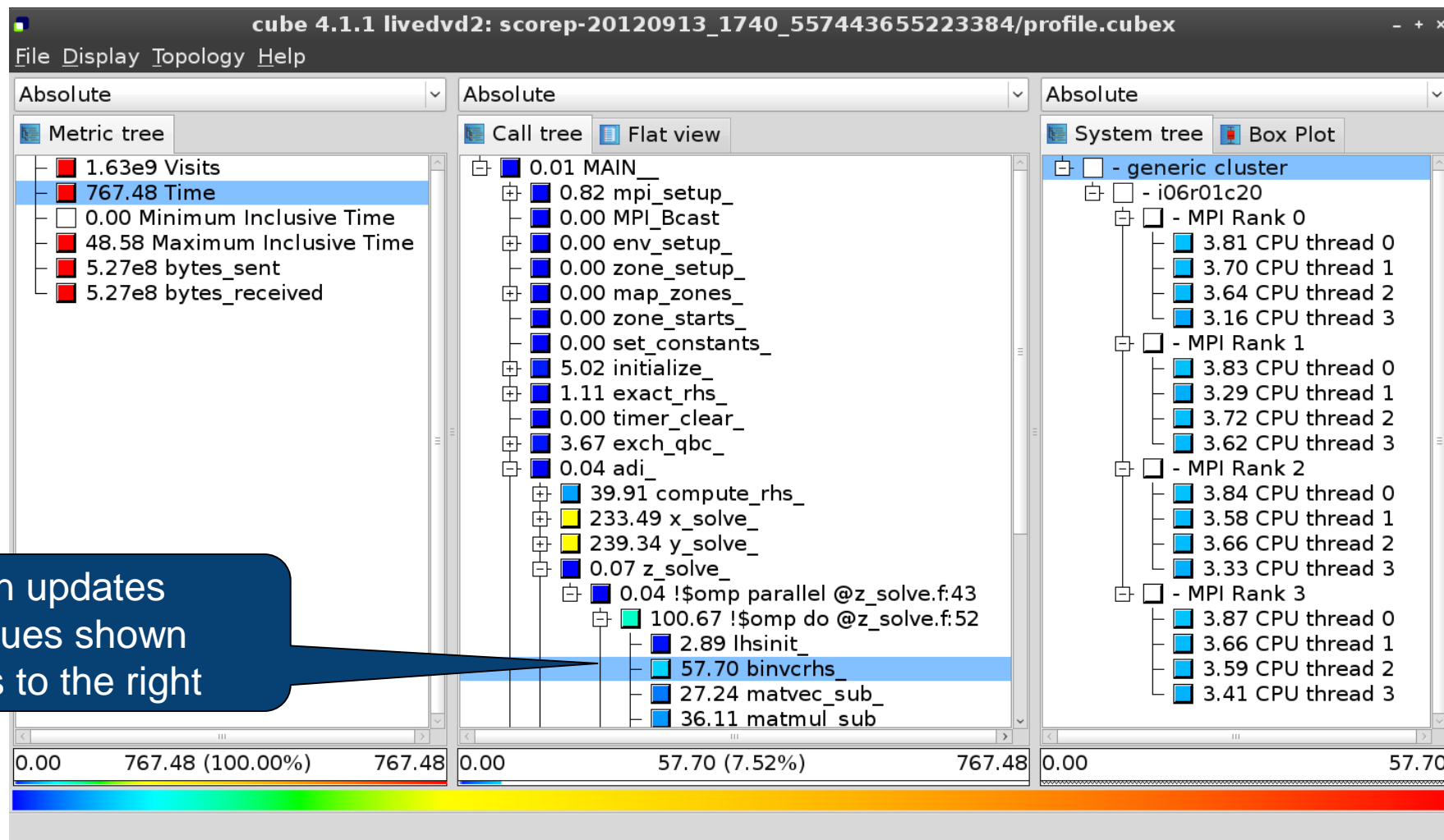
Expanding the system tree



Expanding the call tree



Selecting a call path



Source-code view via context menu

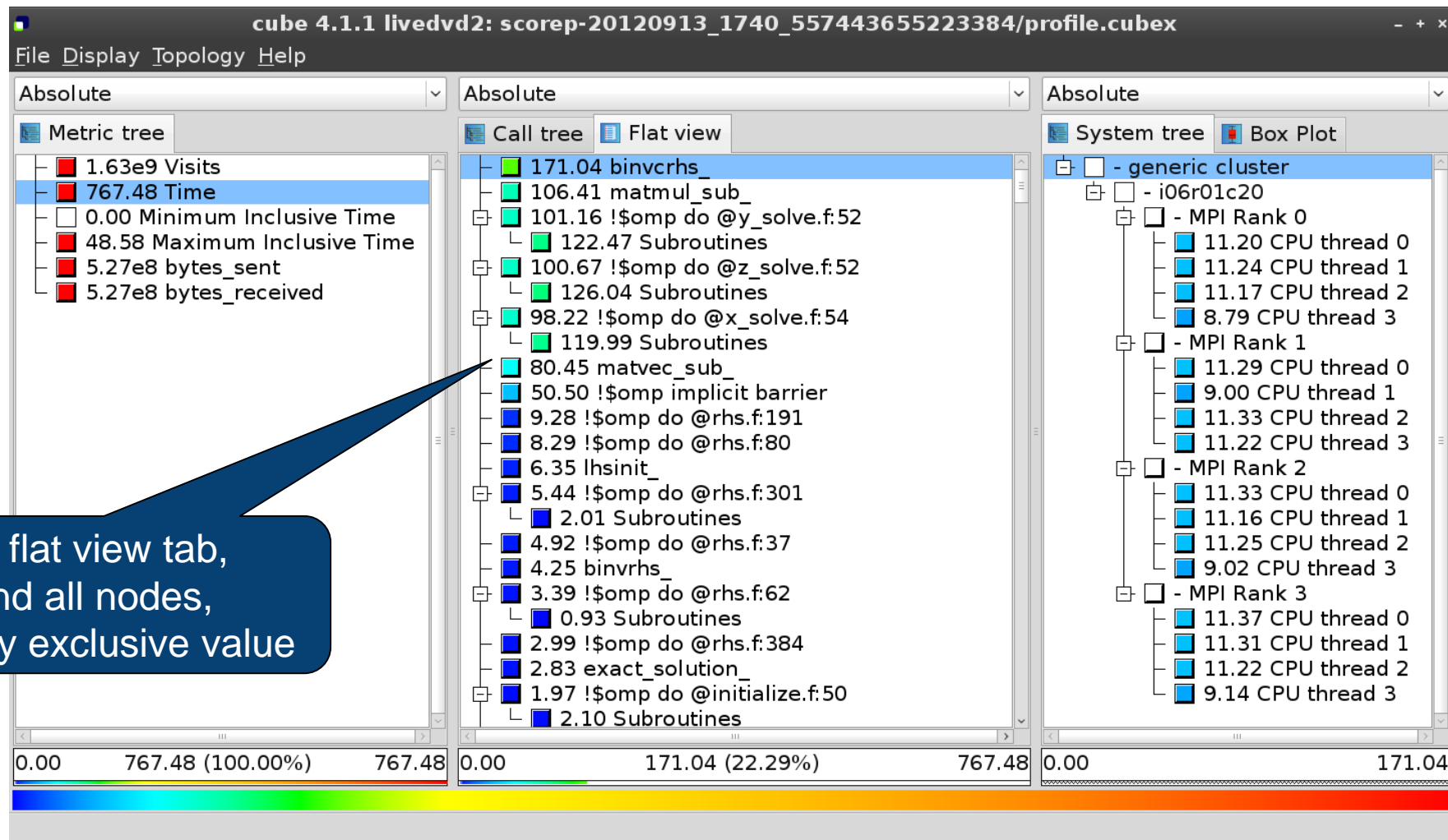


The screenshot displays the 'cube 4.1.1 livedvd2: scorep-20120913_1740_557443655223384/profile.cubex' application. It features three main panels: 'Metric tree', 'Call tree', and 'System tree'. The 'Call tree' panel is active, showing a hierarchical view of function calls. A context menu is open over the 'binvcrhs' node, with 'Source code' selected. A blue callout box points to the 'binvcrhs' node with the text 'Right-click opens context menu'. The status bar at the bottom indicates the current view shows the source code of the clicked item.

Right-click opens context menu

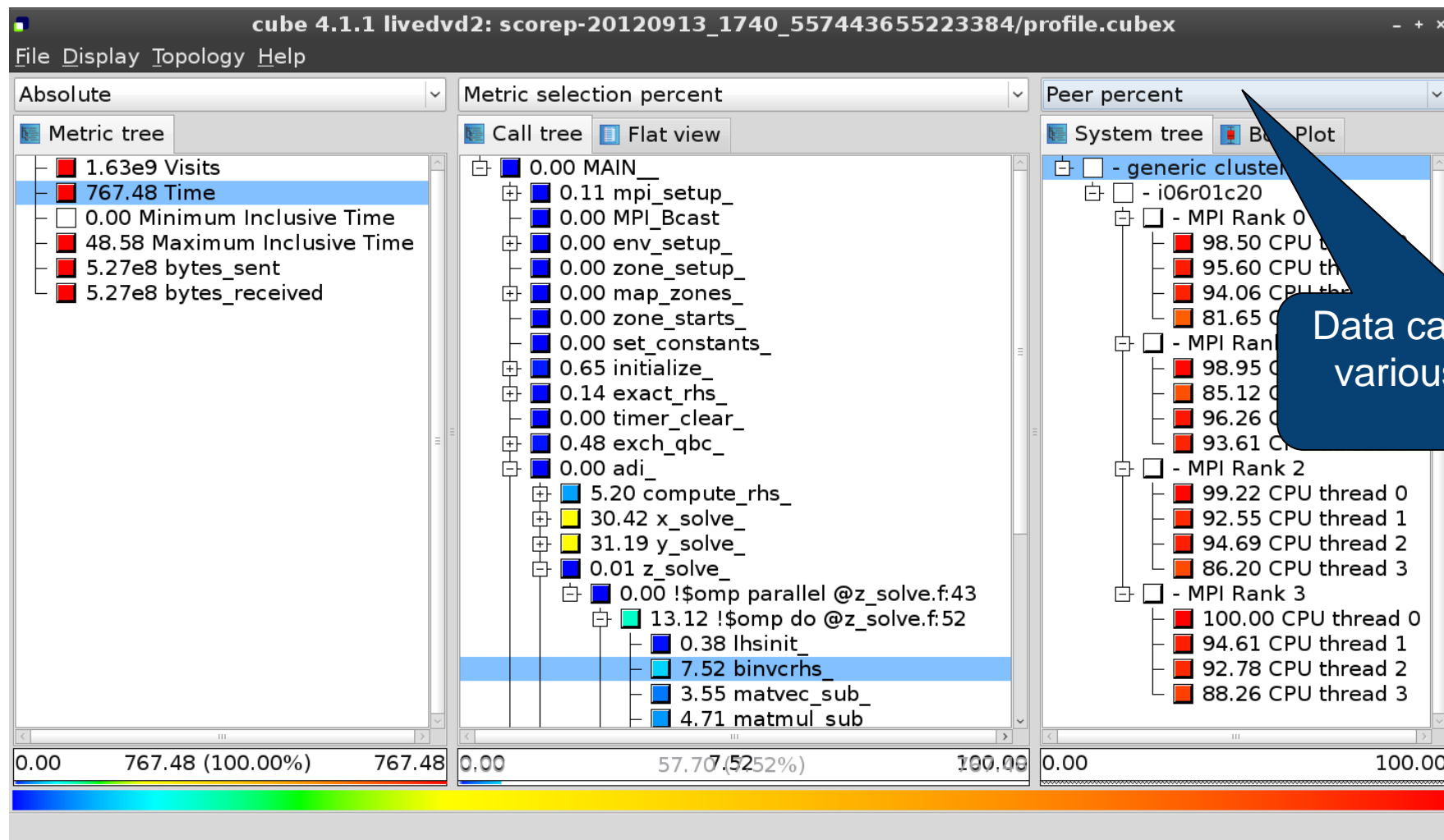
Shows the source code of the clicked item

Flat profile view

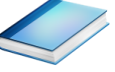


Select flat view tab,
expand all nodes,
and sort by exclusive value

Alternative display modes

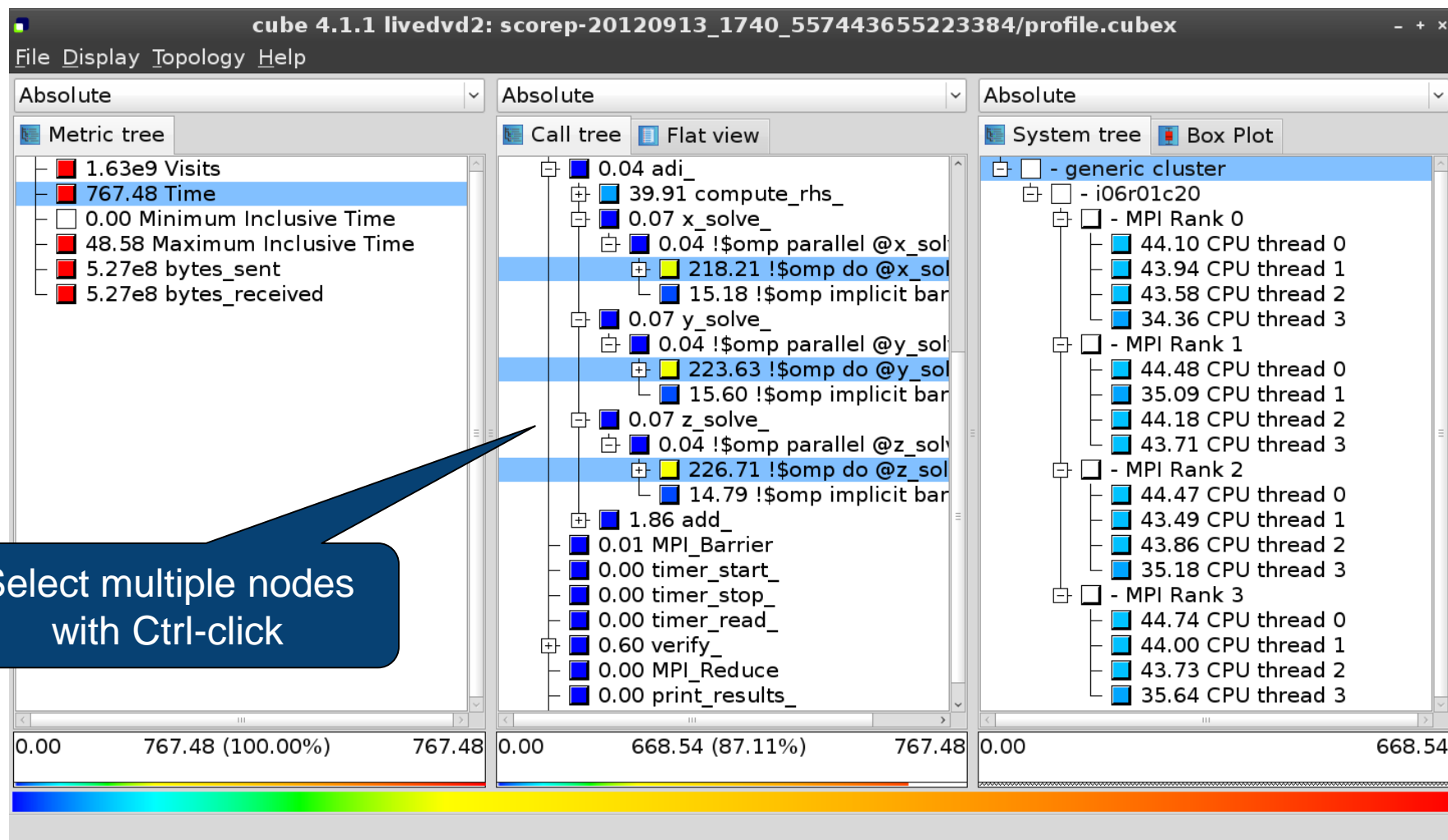


Important display modes



- Absolute
 - Absolute value shown in seconds/bytes/counts
- Selection percent
 - Value shown as percentage w.r.t. the selected node
"on the left" (metric/call path)
- Peer percent (system tree only)
 - Value shown as percentage relative to the maximum peer value

Multiple selection



Context-sensitive help



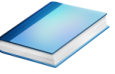
The screenshot displays the cube 4.1.1 GUI with a context-sensitive help menu open over the 'Metric tree' panel. The menu includes options like 'Getting started', 'Mouse and keyboard control', 'What's This?' (highlighted), and 'About'. A blue callout box points to the 'What's This?' option with the text: 'Context-sensitive help available for all GUI items'. The main window shows a hierarchical view of performance metrics, including 'Visits', 'Time', and various computational steps like 'compute_rhs_', 'solve_', and 'verify_'. A 'System tree' panel on the right shows a hierarchy of MPI ranks and CPU threads. A status bar at the bottom indicates 'Change into help mode for display components'.

Derived metrics



- Derived metrics are defined using CubePL expressions, e.g.:
`metric::time(i)/metric::visits(e)`
- Values of derived metrics are not stored, but calculated on-the-fly
- Types of derived metrics:
 - Prederived: evaluation of the CubePL expression is performed before aggregation
 - Postderived: evaluation of the CubePL expression is performed after aggregation
- Examples:
 - “Average execution time”: Postderived metric with expression
`metric::time(i)/metric::visits(e)`
 - “Number of FLOP per second”: Postderived metric with expression
`metric::FLOP()/metric::time()`

Derived metrics in Cube GUI



Collection of derived metrics

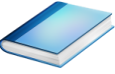
Parameters of the derived metric

CubePL expression

The screenshot shows the Cube GUI interface with a dialog box for creating a new derived metric. The dialog is titled "Create new metric as a child of metric". It has several fields and sections:

- Select metric from collection:** Average execution time (kenobi)
- Derived metric type:** Postderived metric
- Display name:** Average visit time
- Unique name:** avg_visit_time
- Data type:** DOUBLE
- Unit of measurement:** sec
- URL:** (empty)
- Description:** Calculates average time of region execution per visit. Autor is Michael Knobloch.
- Calculation:** A section with a checked checkbox and a text area containing the CubePL expression: `metric::time()/metric::visits(e)`
- Buttons:** "Create metric" and "Cancel"
- Footer:** "Share this metric with SCALASCA group" and a green checkmark icon.

Example: FLOPS based on PAPI_FP_OPS and time



Cube-4.3.1: scorep_8x4_sum/profile.cubex (on froggy1)

File Display Plugins Help
Restore Setting Save Settings

Edit metric FLOPS (on froggy1)

Select metric from collection: --- please select ---

Derived metric type: Postderived metric

Display name: FLOPS

Unique name: flops

Data type: DOUBLE

Unit of measurement:

URL:

Description:

Calculation Calculation Init Aggregation "+" Aggregation "-"

`metric::PAPI_FP_OPS()/metric::time()`

Edit metric Cancel

Share this metric with SCALASCA group

Absolute Metric tree

- 1.17e7 Visits (occ)
- 1148.49 Time (sec)
- 0.00 Minimum Inclusive Time (sec)
- 41.57 Maximum Inclusive Time (...)
- 0 bytes_put (bytes)
- 0 bytes_get (bytes)
- 5.75e12 PAPI_TOT_INS (#)
- 2.69e12 PAPI_TOT_CYC (#)
- 2.12e12 PAPI_FP_OPS (#)
- 3.12e9 bytes_sent (bytes)
- 3.12e9 bytes_received (bytes)
- 1.84e9 FLOPS**

Absolute Call tree Flat view

- 3.17e5 MAIN_
 - 7.04e5 mpi_setup_
 - 6.34e4 MPI_Bcast
 - 2.05e5 env_setup_
 - 7.39e5 zone_setup_
 - 9.31e5 map_zones_
 - 9.39e4 zone_starts_
 - 6.16e5 set_constants_
 - 5.91e8 initialize_
 - 0.00 exact_rhs_
 - 145.62 !\$omp parallel @exac...
 - 2.54e4 !\$omp do @exact_r...
 - 9.65e8 !\$omp do @exact_r...**
 - 9.62e8 !\$omp do @exact_r...
 - 8.14e8 !\$omp do @exact_r...
 - 1.21e5 !\$omp do @exact_r...
 - 0.00 !\$omp implicit barrier...
 - 6.23e4 exch_qbc_
 - 1.94e9 adi_
 - 2.19e5 MPI_Barrier
 - 1.92e9 <<bt_iter>> (200 itera...
 - 1.98e8 verify_
 - 1.05e5 MPI_Reduce

Absolute System tree Barplot Heatmap

 - machine Linux
 - node frog6
 - MPI Rank 0
 - 1.17e9 Master thread
 - 9.43e8 OMP thread 1
 - 9.47e8 OMP thread 2
 - 9.47e8 OMP thread 3
 - MPI Rank 1
 - 1.17e9 Master thread
 - 9.87e8 OMP thread 1
 - 9.68e8 OMP thread 2
 - 9.72e8 OMP thread 3
 - MPI Rank 2
 - 1.10e9 Master thread
 - 8.97e8 OMP thread 1
 - 8.77e8 OMP thread 2
 - 8.76e8 OMP thread 3
 - MPI Rank 3
 - 1.09e9 Master thread
 - 9.06e8 OMP thread 1
 - 9.04e8 OMP thread 2
 - 9.02e8 OMP thread 3

All (32 elements)

0.00 1.84e9 (100.00%) 1.84e9 0.00 9.65e8 (-0.00%) -12858016489314434.00 0.00... -179769313486231570814527423731704356798070...

Selected "\$!omp do @exact_rhs.f:46"

Iteration profiling

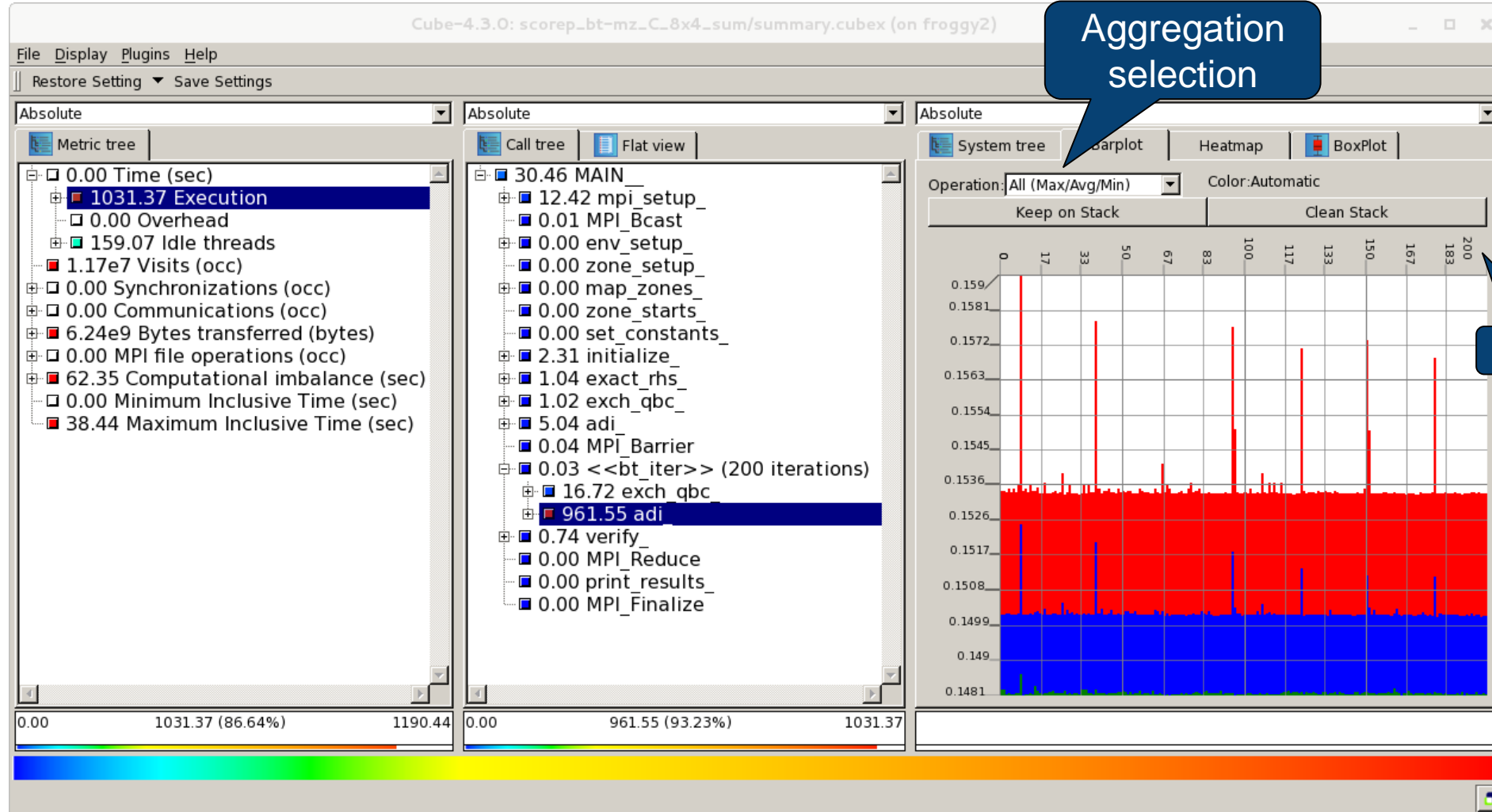
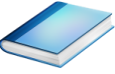


- Show time dependent behavior by “unrolling” iterations
- Preparations:
 - Mark loop body by using Score-P instrumentation API in your source code

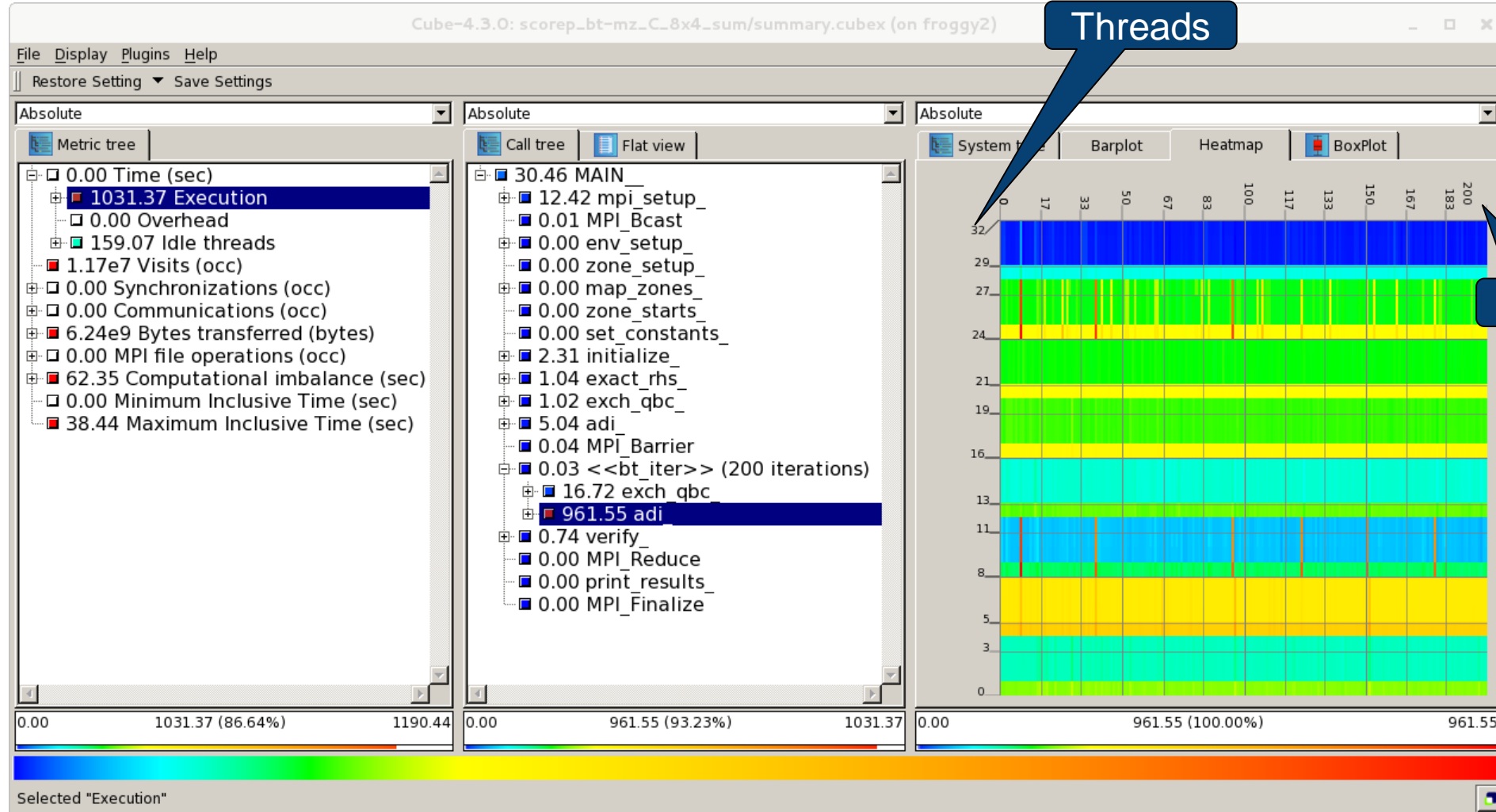
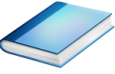
```
SCOREP_USER_REGION_DEFINE( scorep_bt_loop )  
SCOREP_USER_REGION_BEGIN( scorep_bt_loop, "<<bt_iter>>", SCOREP_USER_REGION_TYPE_DYNAMIC )  
SCOREP_USER_REGION_END( scorep_bt_loop )
```

- Result in the Cube profile:
 - Iterations shown as separate call trees
 - Useful for checking results for specific iterations
 - or
 - Select your user-instrumented region and mark it as loop
 - Choose “Hide iterations”
 - View the Barplot statistics or the (thread x iterations) Heatmap

Iteration profiling: Barplot



Iteration profiling: Heatmap



CUBE algebra utilities

- Extracting solver sub-tree from analysis report

```
% cube_cut -r '<<ITERATION>>' scorep_bt-mz_C_16x8_sum/profile.cubex  
Writing cut.cubex... done.
```

- Calculating difference of two reports

```
% cube_diff scorep_bt-mz_C_16x8_sum/profile.cubex cut.cubex  
Writing diff.cubex... done.
```

- Additional utilities for merging, calculating mean, etc.
- Default output of `cube_utility` is a new report `utility.cubex`
- Further utilities for report scoring & statistics
- Run utility with `-h` (or no arguments) for brief usage info

Square sneak preview

- Scalasca provides **square** to facilitate analysis report exploration
 - square = scalasca -examine [OPTIONS] (./scorep_expt_sum | ./profile.cubex)
- Processes intermediate .cubex files produced by Score-P and Scout
 - profile.cubex -> summary.cubex
 - scout.cubex -> trace.cubex
- and (optionally) starts CUBE GUI with the post-processed file
 - containing additional derived metrics and metric hierarchies



Cube: Further information

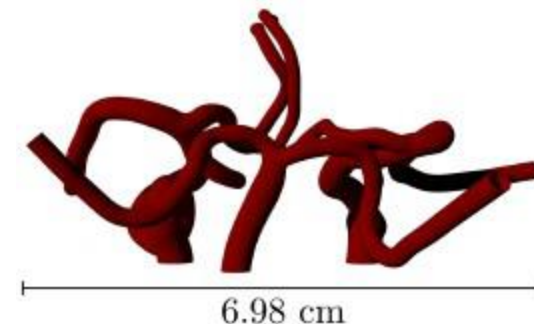
- Parallel program analysis report exploration tools
 - Libraries for Cube report reading & writing
 - Algebra utilities for report processing
 - GUI for interactive analysis exploration
- Available under 3-clause BSD open-source license
- Documentation & sources:
 - <http://www.scalasca.org>
- User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



Score-P/CUBE case study HemeLB

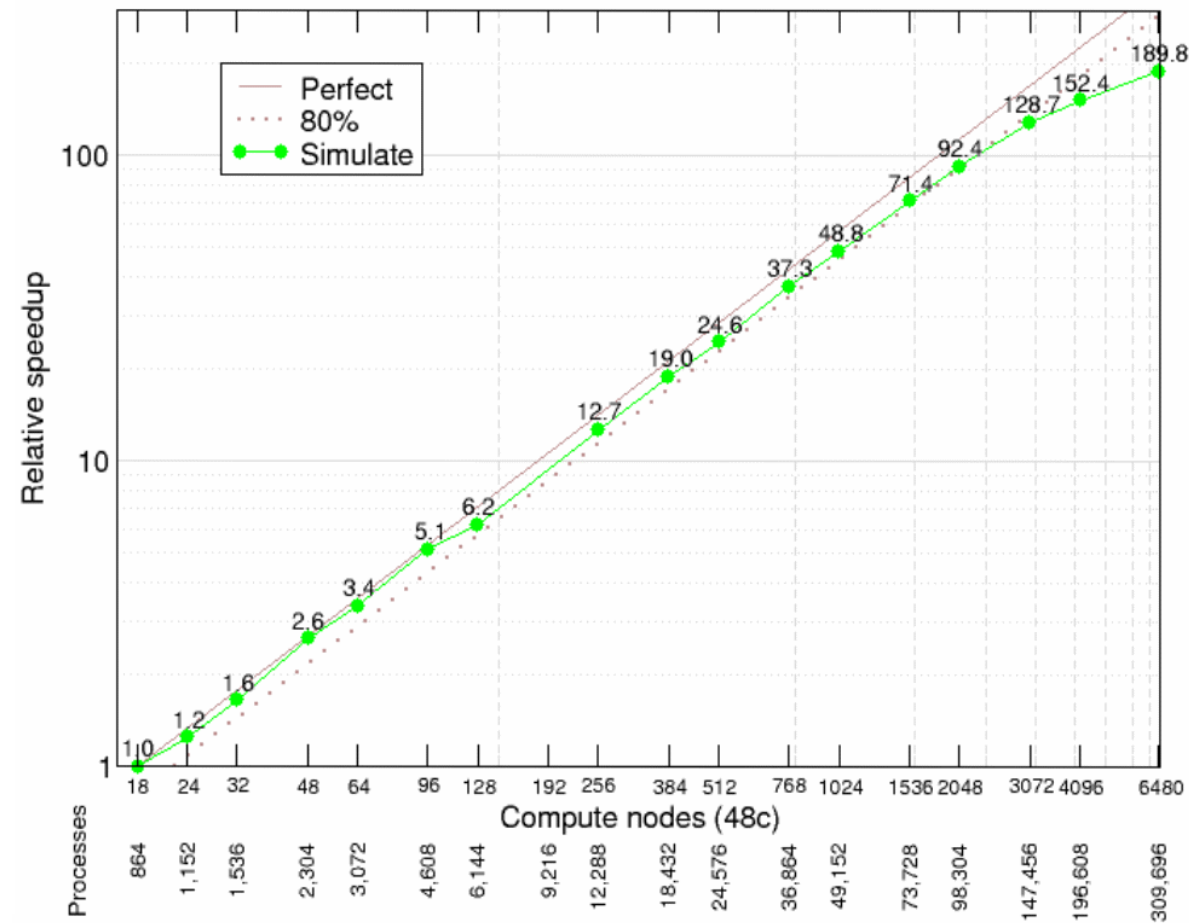
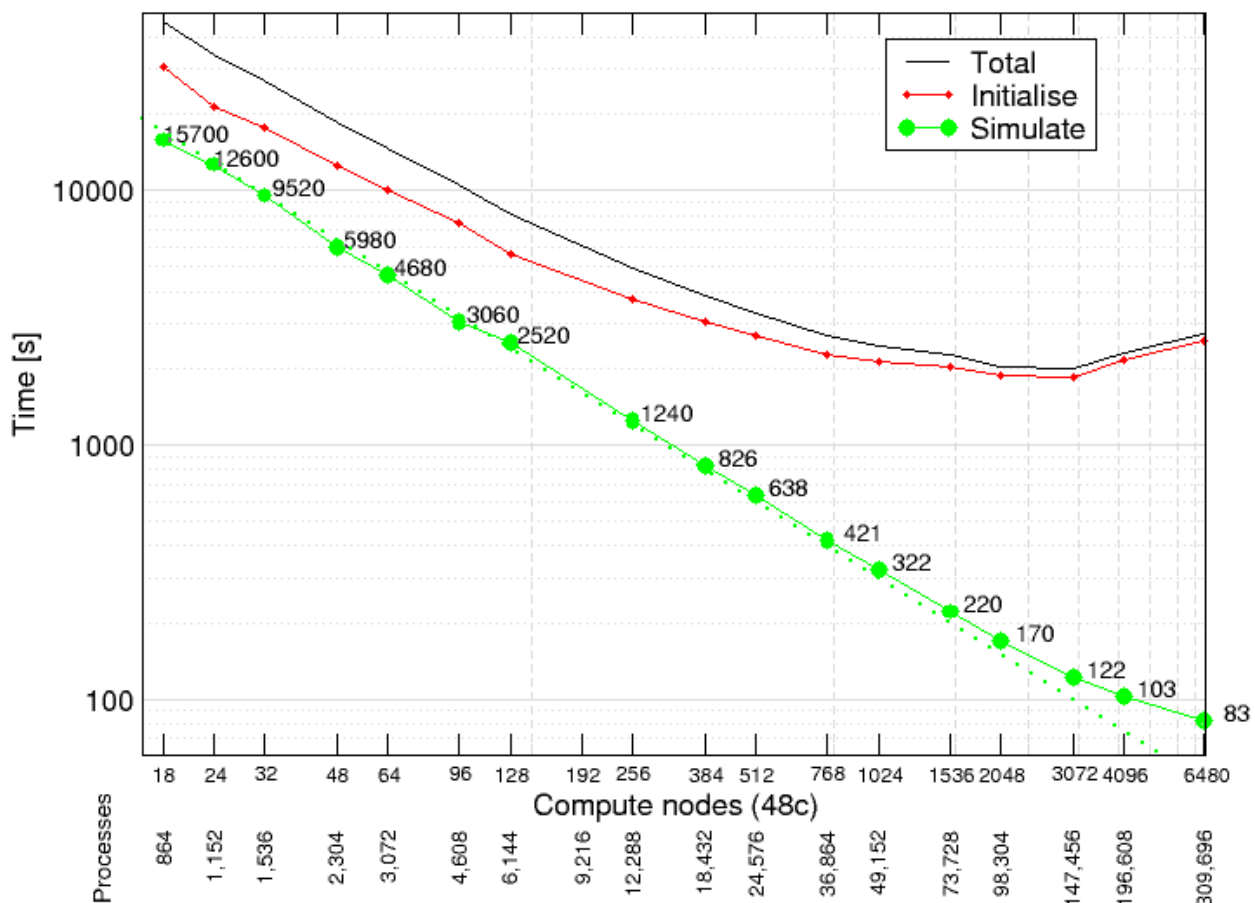
HemeLB (SuperMUC-NG: no GPUs)

- 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 unused]
 - 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 (500 μ s) simulation of cerebrovascular "circle of Willis" geometry
 - 6.4 μ m 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 of FOA *RunSimulation*

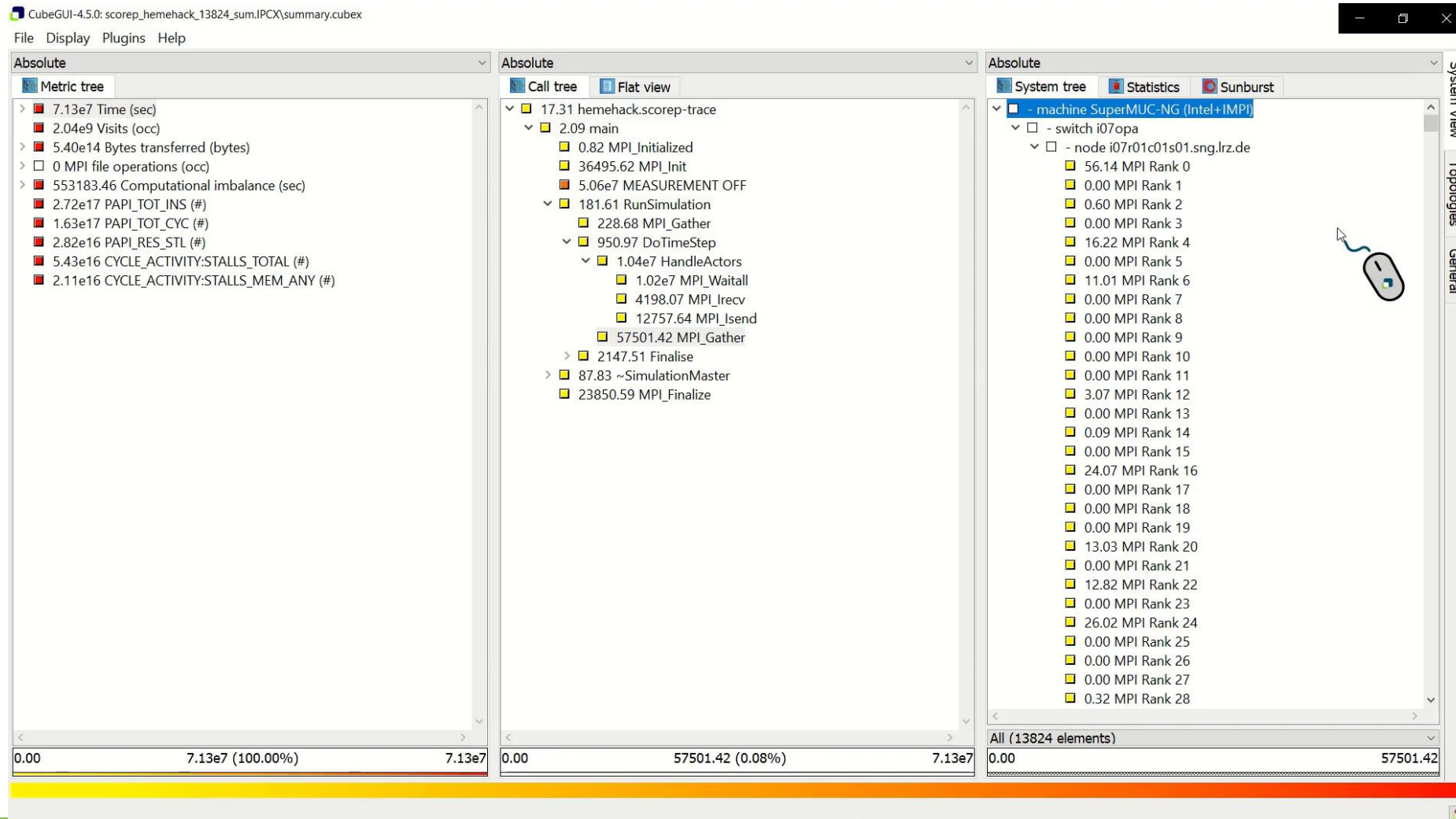


[Execution of 9,216 processes on 192 compute nodes not possible due to insufficient compute nodes with adequate memory in 'fat' partition (768 GiB vs. regular 96 GiB node memory)]

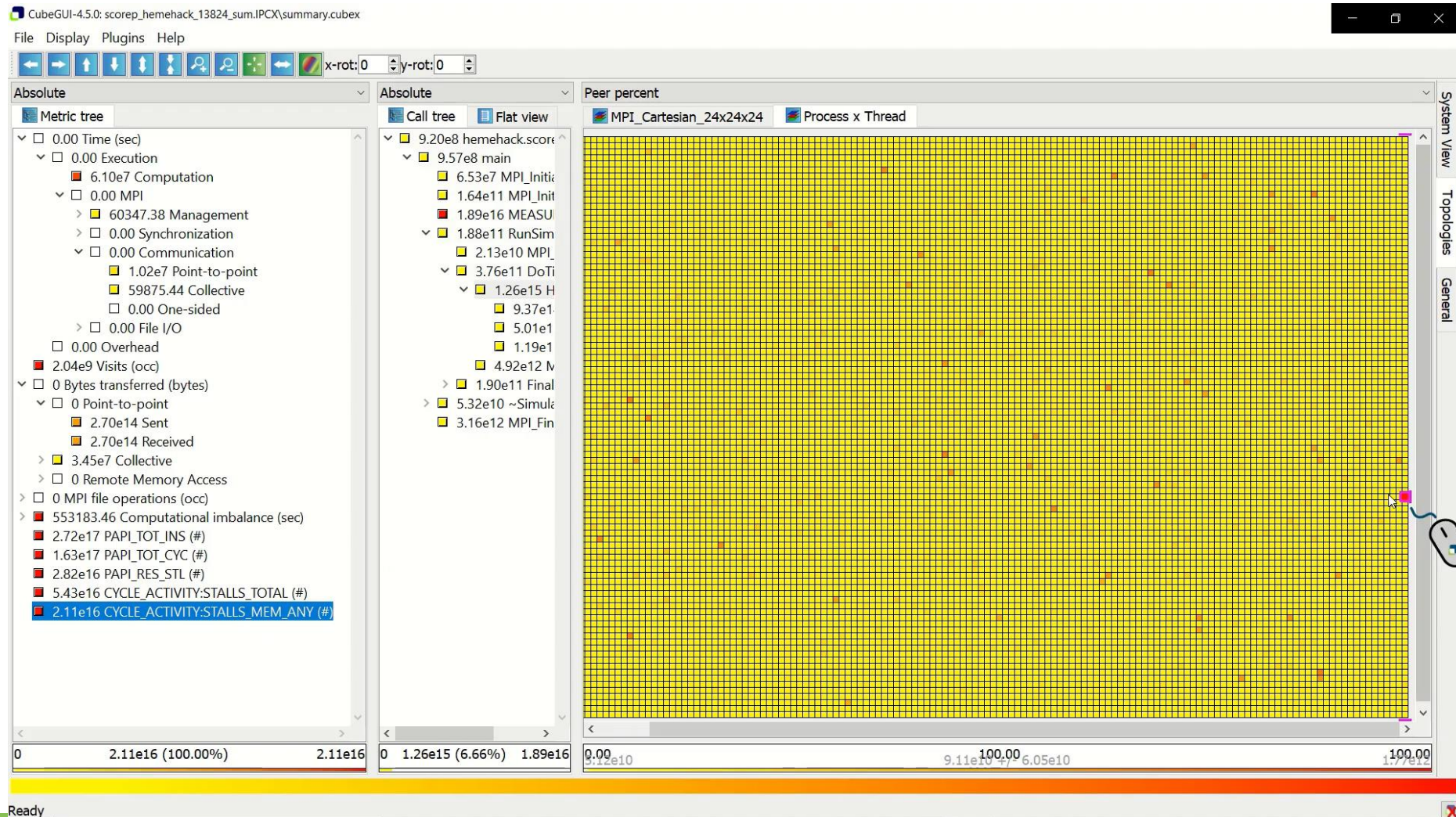
Initial tree presentation: Time of MPI_Gather per MPI process



DOI 10.5281/zenodo.4080701



Topological presentation: STALLS_MEM_ANY for HandleActors



Advisor: POP efficiency assessment for RunSimulation



CubeGUI-4.5.0: scorep_hemehack_13824_sum.IPCX\summary.cubex

File Display Plugins Help

POP Assessment Runtime threshold: 2.5 % runtime

Absolute

Metric tree

- 0.00 Time (sec)
 - 0.00 Execution
 - 6.10e7 Computation
 - 0.00 MPI
 - 60347.38 Management
 - 0.00 Synchronization
 - 0.00 Communication
 - 1.02e7 Point-to-point
 - 59875.44 Collective
 - 0.00 One-sided
 - 0.00 File I/O
 - 0.00 Overhead
 - 2.04e9 Visits (occ)
 - 0 Bytes transferred (bytes)
 - 0 Point-to-point
 - 2.70e14 Sent
 - 2.70e14 Received
 - 3.45e7 Collective
 - 0 Remote Memory Access
 - 0 MPI file operations (occ)
 - 553183.46 Computational imbalance (sec)
 - 2.72e17 PAPI_TOT_INS (#)
 - 1.63e17 PAPI_TOT_CYC (#)
 - 2.82e16 PAPI_RES_STL (#)
 - 5.43e16 CYCLE_ACTIVITY:STALLS_TOTAL (#)
 - 2.11e16 CYCLE_ACTIVITY:STALLS_MEM_ANY (#)

0 2.11e16 (100.00%) 2.11e16

Absolute

Call tree Flat view

- 9.20e8 hemehack.scorep-trace
 - 9.57e8 main
 - 6.53e7 MPI_Initialized
 - 1.64e11 MPI_Init
 - 1.89e16 MEASUREMENT OFF
 - 2.22e15 RunSimulation
 - 5.32e10 ~SimulationMaster
 - 3.16e12 MPI_Finalize

0 2.22e15 (10.50%) 2.11e16

Advisor Score-P Configuration Source

Recalculate automatic

POP Assessment : SimulationMaster::RunSimulation

| | | | | |
|---------------------------------|-----|-----------|---------|---|
| Parallel Efficiency | 50% | Fair | 0.50 | ? |
| Load balance | 50% | Fair | 0.50 | ? |
| Communication Efficiency | 99% | Very good | 1.00 | ? |
| Serialisation Efficiency | 0% | | | ! |
| Transfer efficiency | 0% | | | ! |
| Stalled Resources | 57% | Fair | 0.58 | ? |
| IPC | | Value | 1.56 | ? |
| Instructions (only computation) | | Value | 3.72e16 | ? |
| Computation time | | Value | 1.04e7 | ? |

Candidates

| Callpath | Issue |
|----------|-------|
| | |

Ready

HemeLB (JUWELS-Volta)

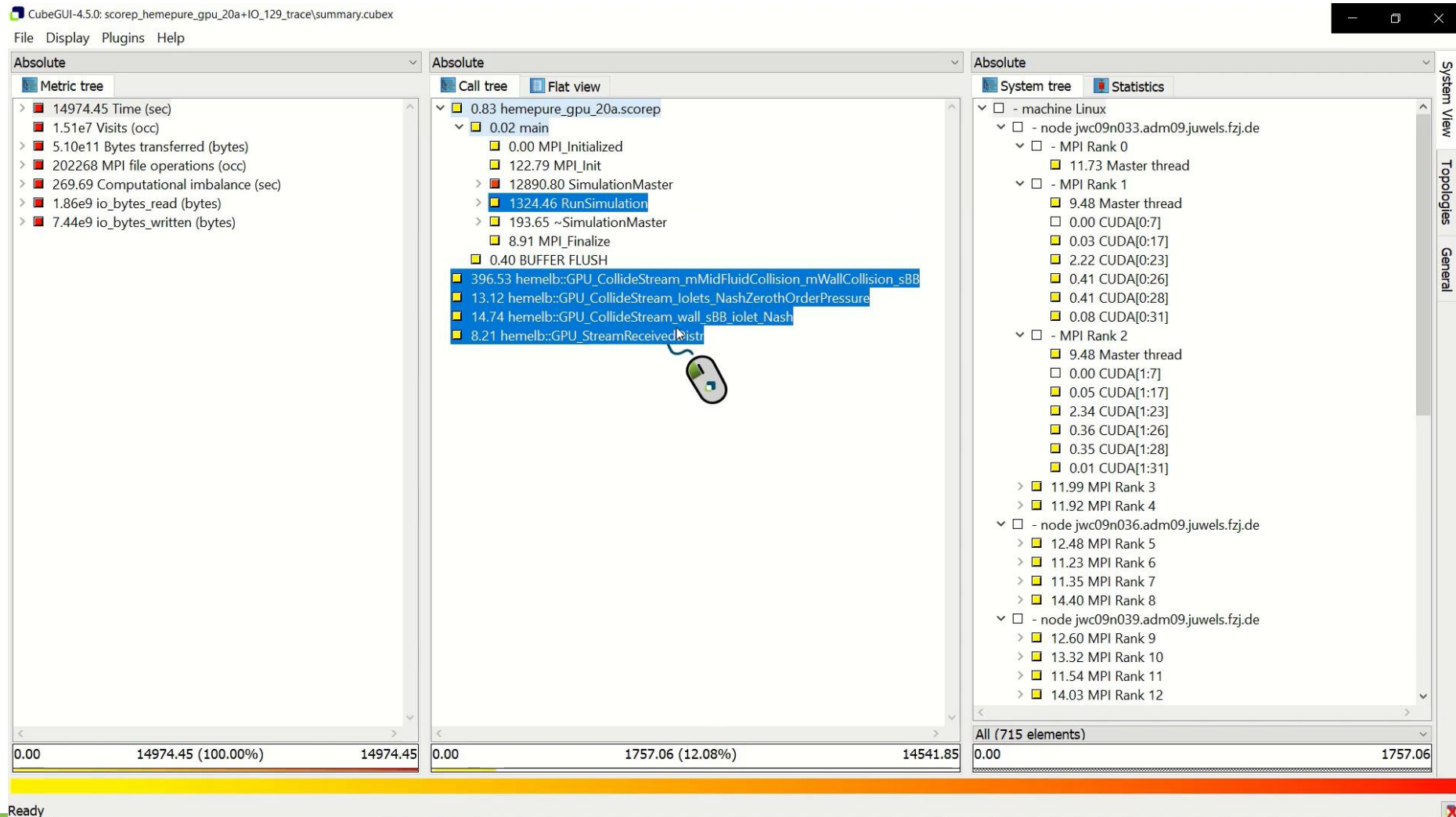
- 3D macroscopic blood flow in human arterial system developed by UC London (UK)
 - lattice-Boltzmann method tracking fluid particles on a lattice grid with complex boundary conditions
 - exascale flagship application of EU H2020 HPC Centre of Excellence for Computational Biomedicine
 - HemeLB open-source 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 100 μ s) 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
- \Rightarrow Identification & quantification of impact of load balance and its variation**



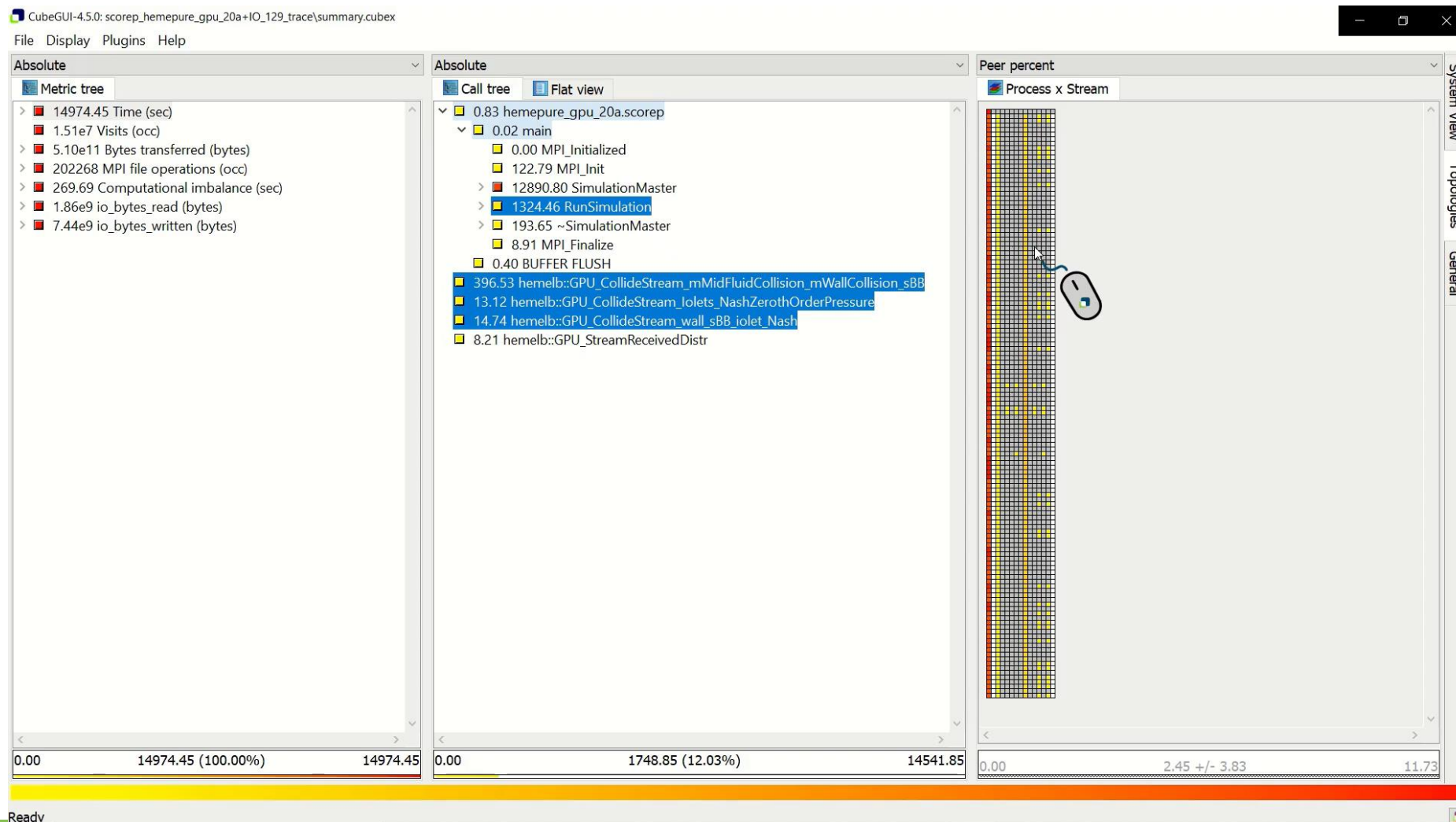
Tree: Time for asynch. CUDA kernels on separate CUDA streams



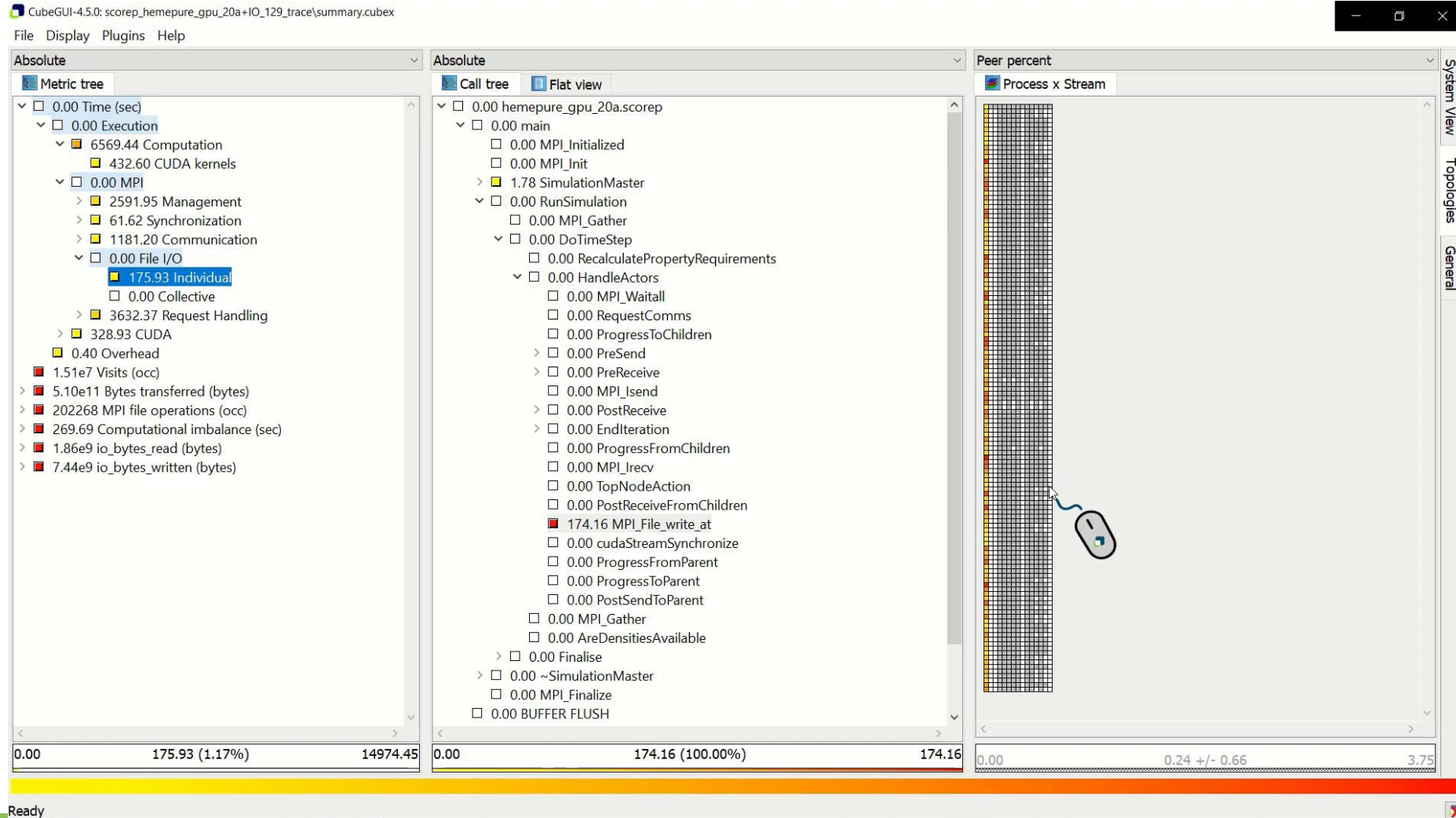
DOI 10.5281/zenodo.4081080



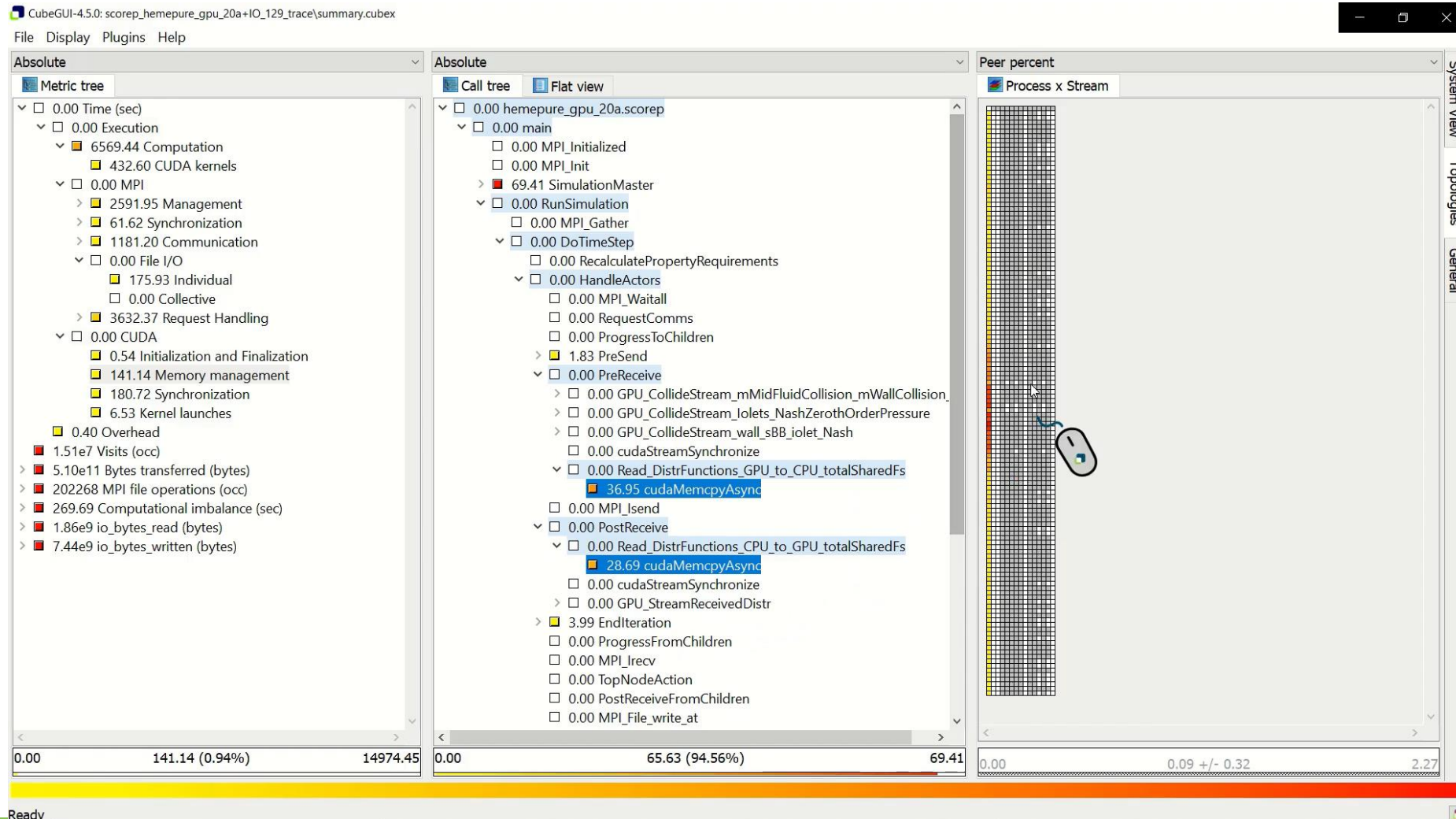
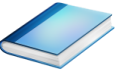
Topo: Time for asynch. CUDA kernels on separate CUDA streams



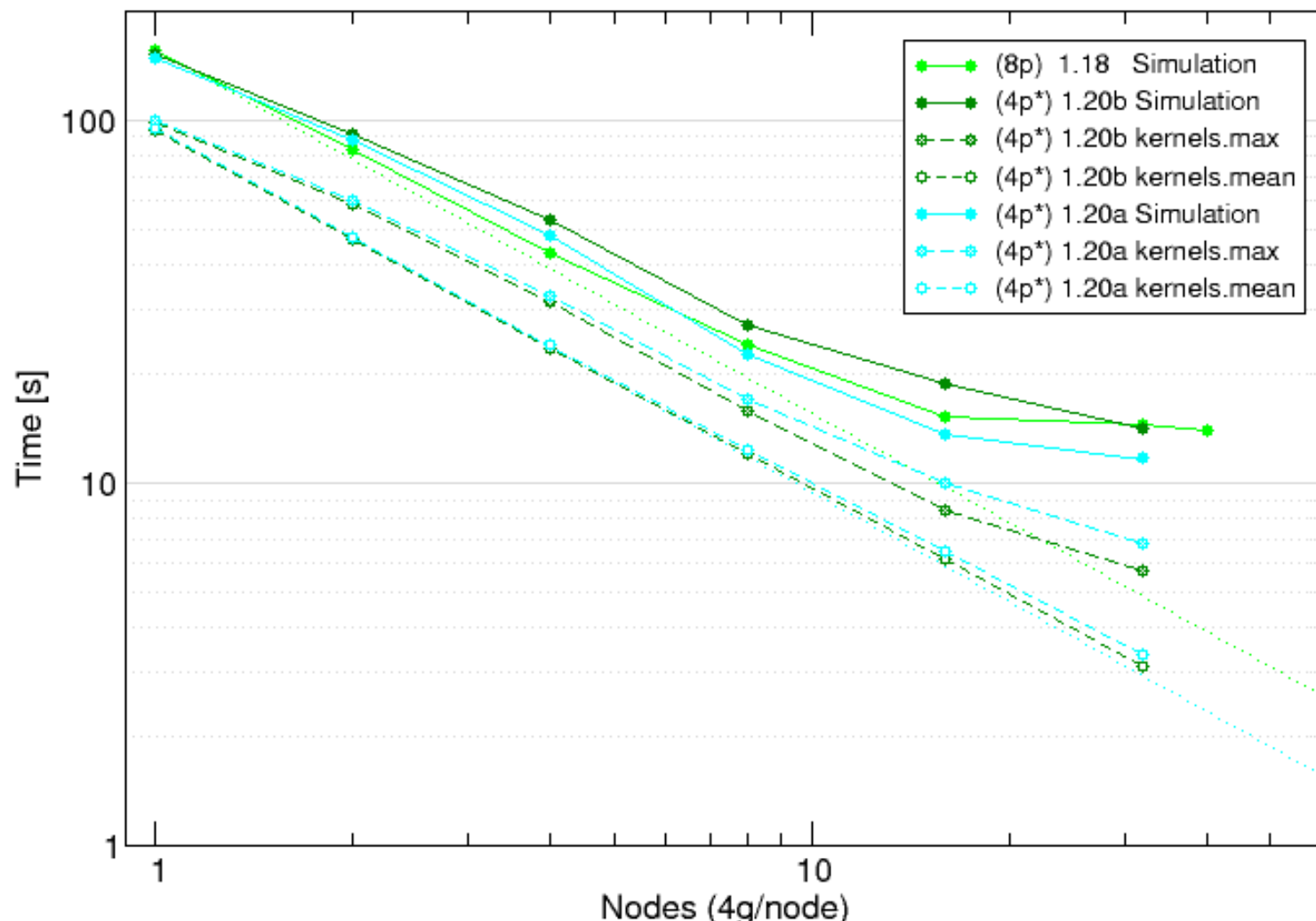
Topo: Time for MPI file writing on CPU varies per MPI process



Topo: Time for CUDA asynchronous memory copies is imbalanced



HemeLB@JUWELS-Volta strong scaling of FOA *RunSimulation*



- 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)

HemeLB@JUWELS/Volta strong scaling efficiency of *RunSimulation*

| | 1n 5p | 2n 9p | 4n 17p | 8n 33p | 16n 65p | 32n 129p | Key: |
|------------------------------------|--------|-------|--------|--------|---------|----------|------|
| Simulation time [s] | 147.87 | 88.38 | 48.13 | 22.66 | 13.68 | 11.67 | 1.1 |
| Global scaling efficiency | 0.64 | 0.53 | 0.49 | 0.52 | 0.43 | 0.25 | 1.0 |
| – Parallel efficiency | 0.64 | 0.53 | 0.50 | 0.54 | 0.47 | 0.29 | 0.9 |
| – – Load balance efficiency (GPU) | 0.95 | 0.78 | 0.73 | 0.73 | 0.65 | 0.50 | 0.8 |
| – – Communication efficiency (GPU) | 0.67 | 0.68 | 0.68 | 0.75 | 0.73 | 0.58 | 0.7 |
| – Computation scaling (GPU) | 1.00 | 1.00 | 0.99 | 0.96 | 0.92 | 0.87 | 0.6 |
| | | | | | | | 0.5 |
| | | | | | | | 0.4 |
| | | | | | | | 0.3 |
| | | | | | | | 0.2 |
| | | | | | | | 0.1 |
| | | | | | | | 0.0 |

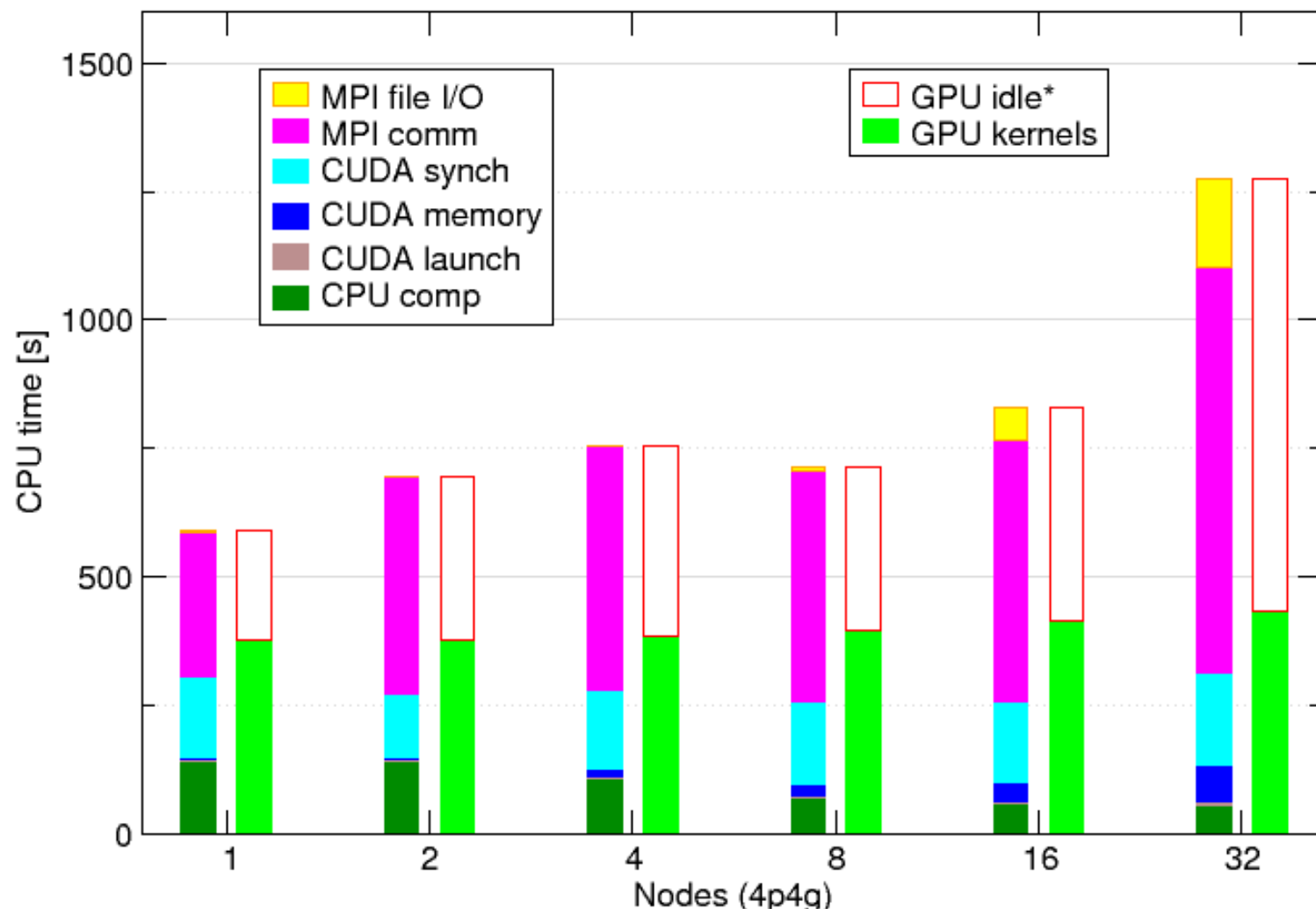
Only considering GPUs (ignoring all CPU cores, 90% of which are completely unused)

- 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]



HemeLB@JUWELS-Volta strong scaling of FOA *RunSimulation*



- 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