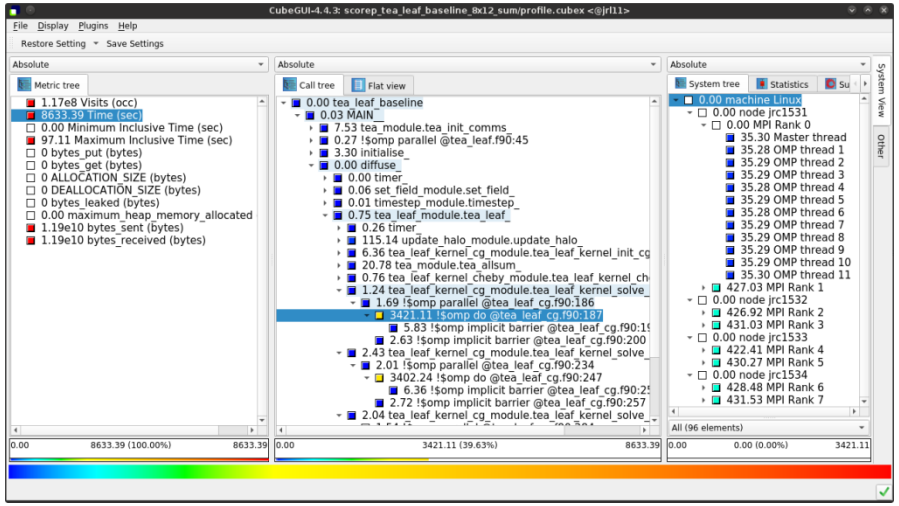
Analysis report examination with Cube

Markus Geimer
Jülich Supercomputing Centre

Cube

CubeLib DOI 10.5281/zenodo.1248078
CubeGUI DOI 10.5281/zenodo.1248087

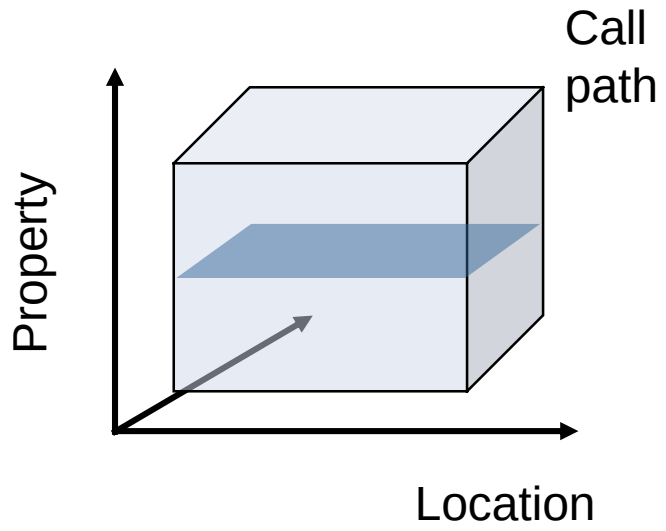
- Parallel program analysis report exploration tools
  - Libraries for XML+binary report reading & writing
  - Algebra utilities for report processing
  - GUI for interactive analysis exploration
    - Requires Qt ≥ 5
- Originally developed as part of the Scalasca toolset
- Now available as a separate component
  - Can be installed independently of Score-P and Scalasca, e.g., on laptop or desktop
  - Latest release: Cube v4.8.2 (September 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

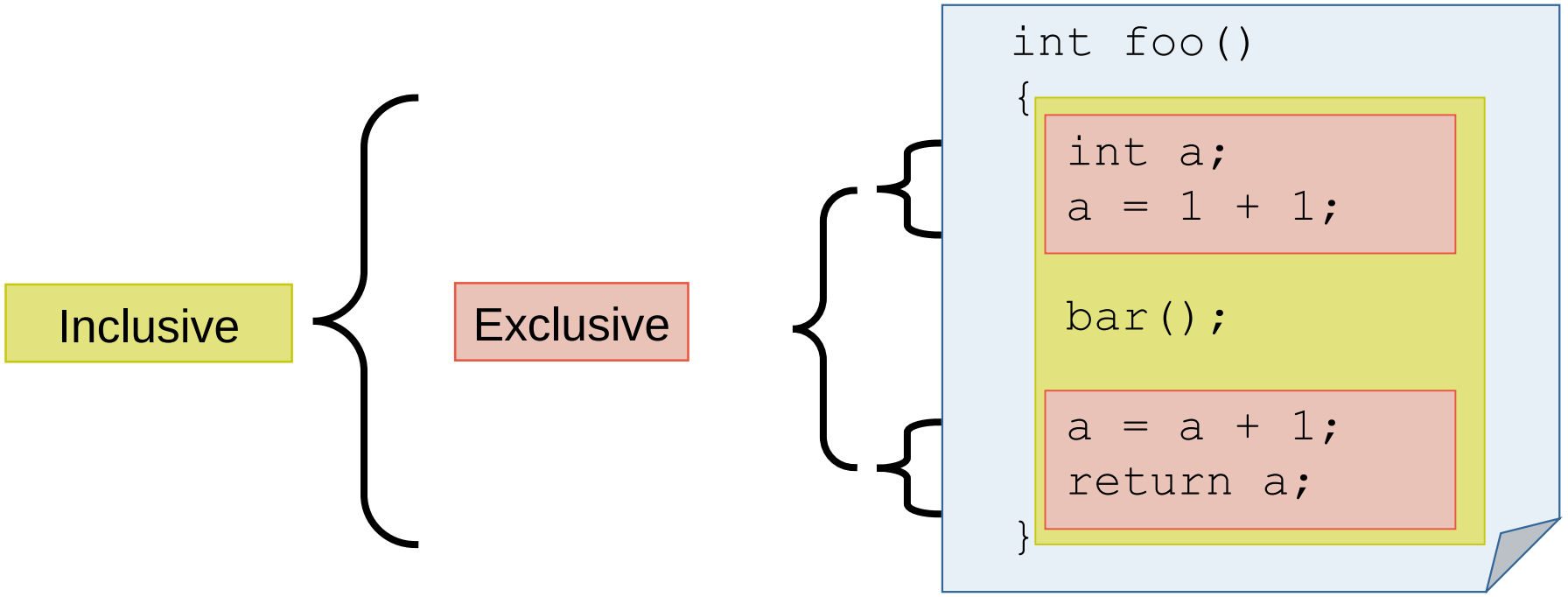
Analysis presentation and exploration

- 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 color: for easy identification of hotspots
  - Inclusive value when closed & exclusive value when expanded
  - Customizable via display modes

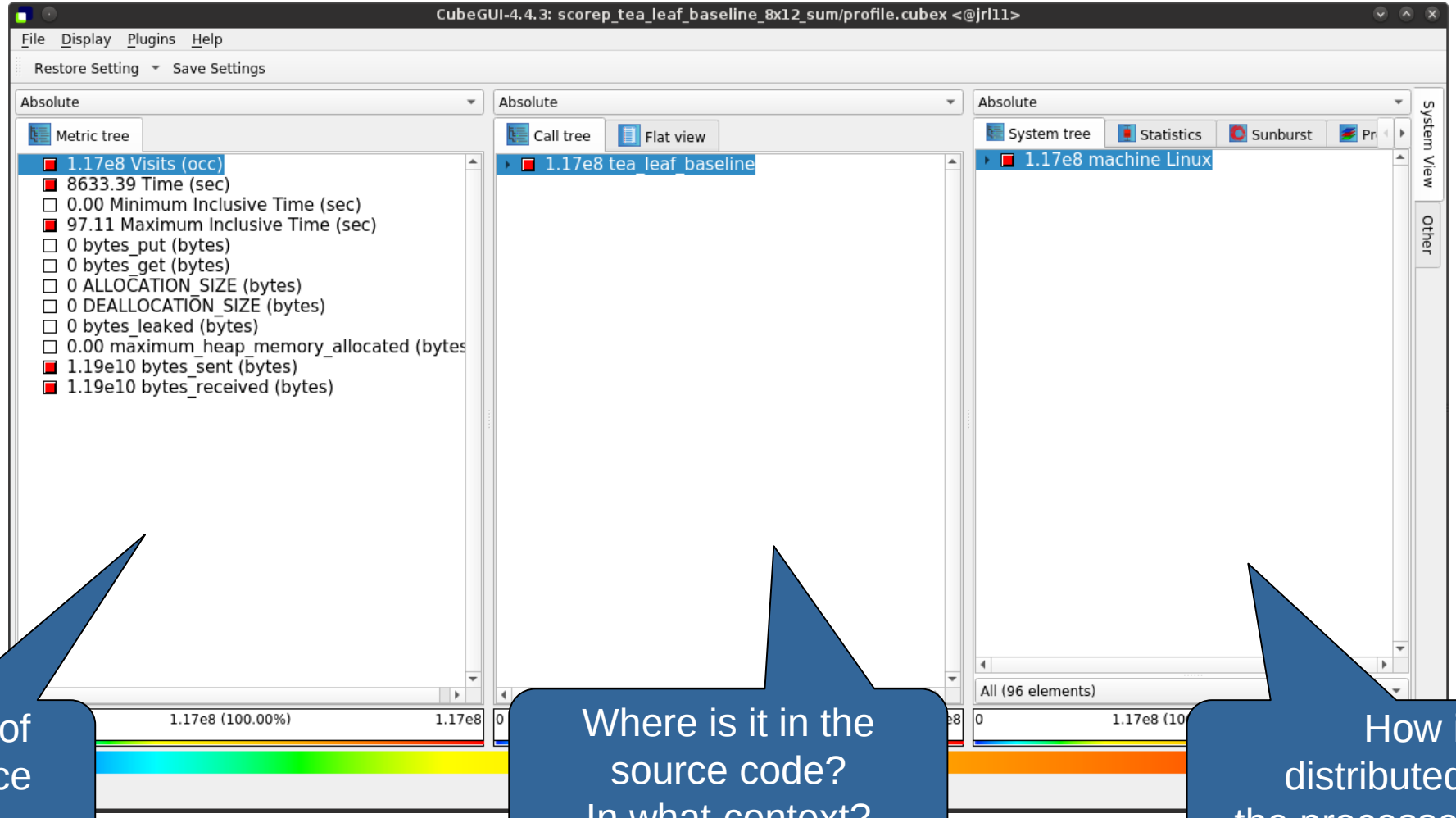


Inclusive vs. exclusive values

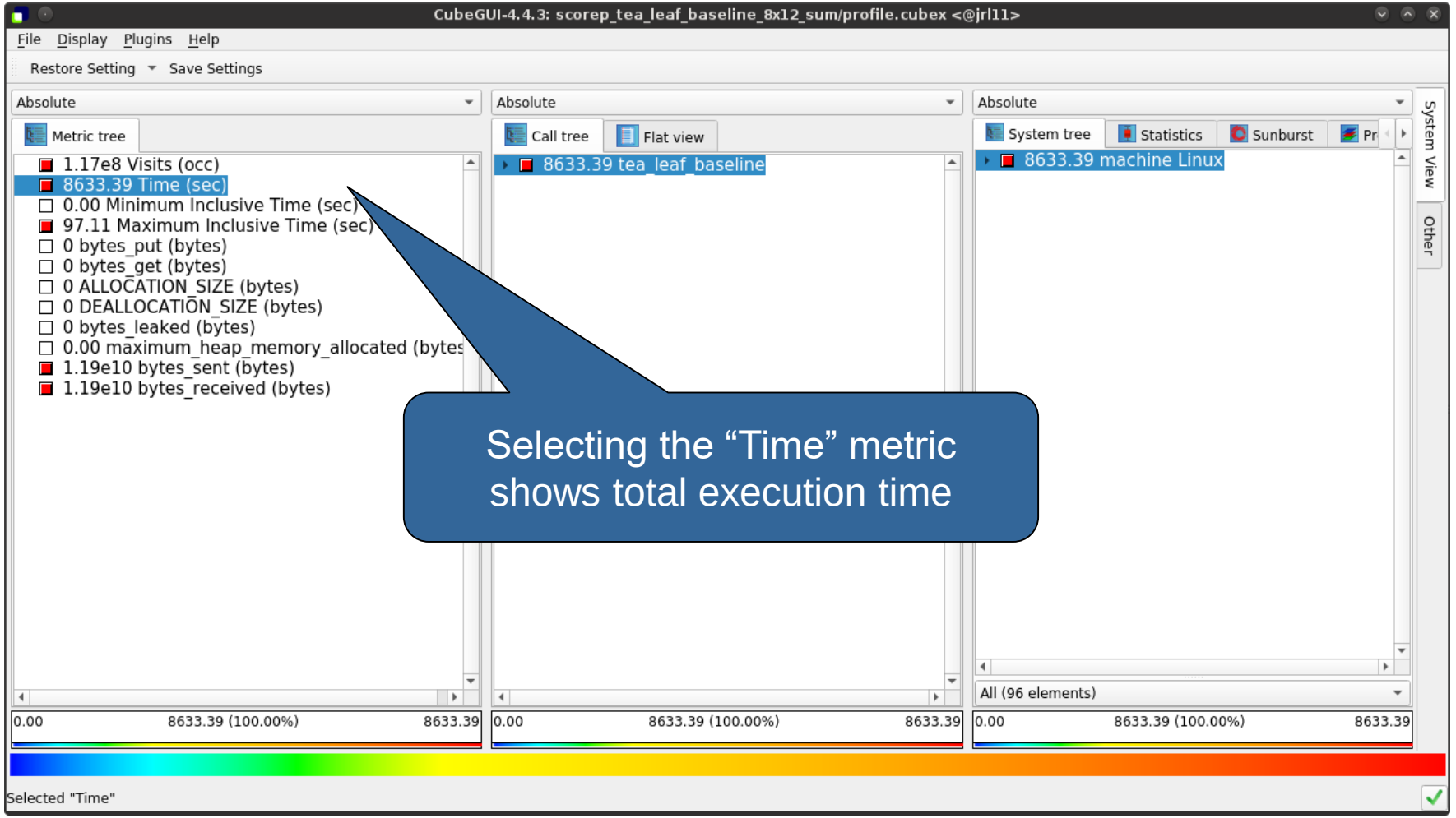
- Inclusive
  - Information of all sub-elements aggregated into single value
- Exclusive
  - Information cannot be subdivided further



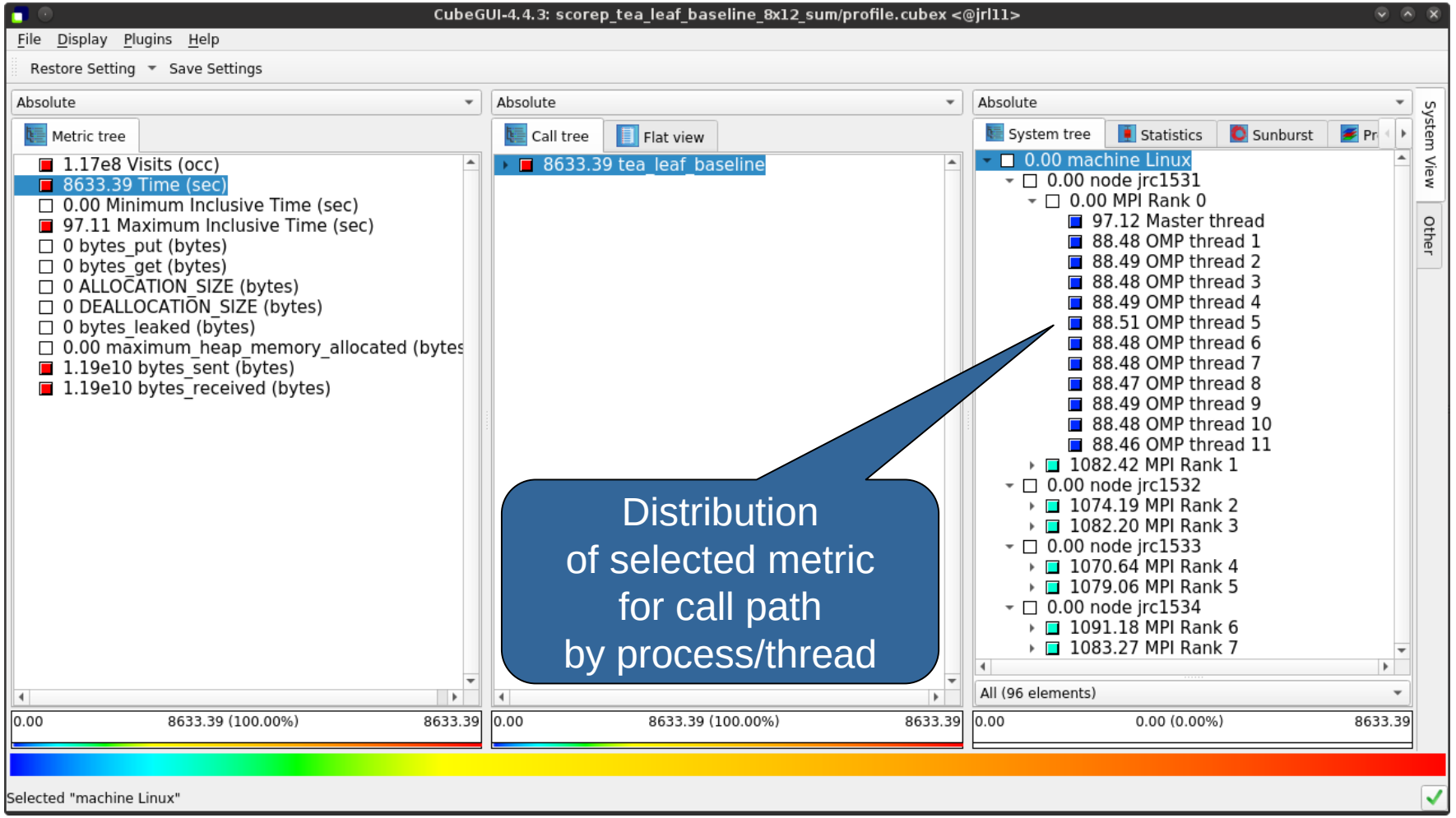
Score-P analysis report exploration (opening view)



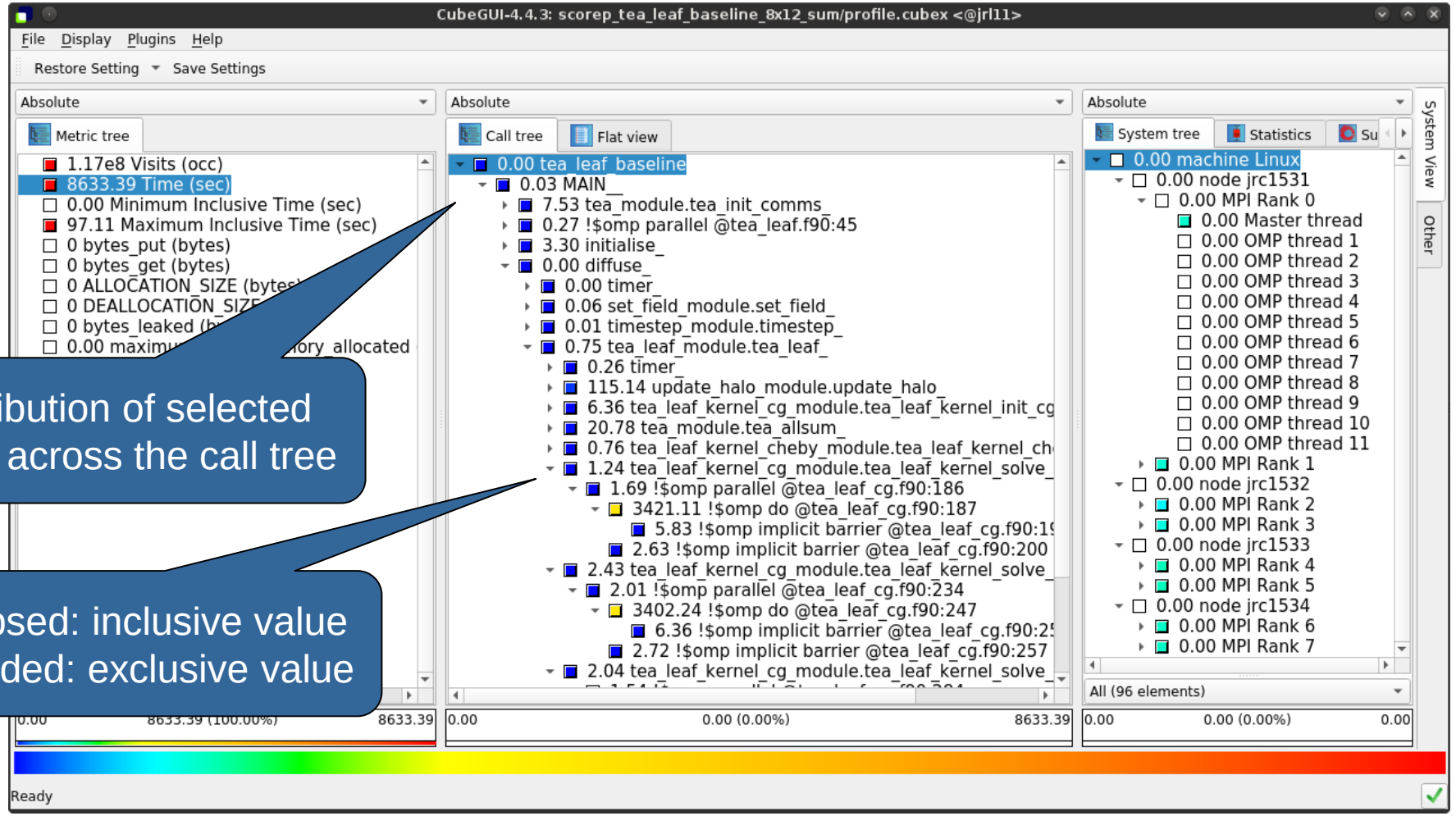
Metric selection



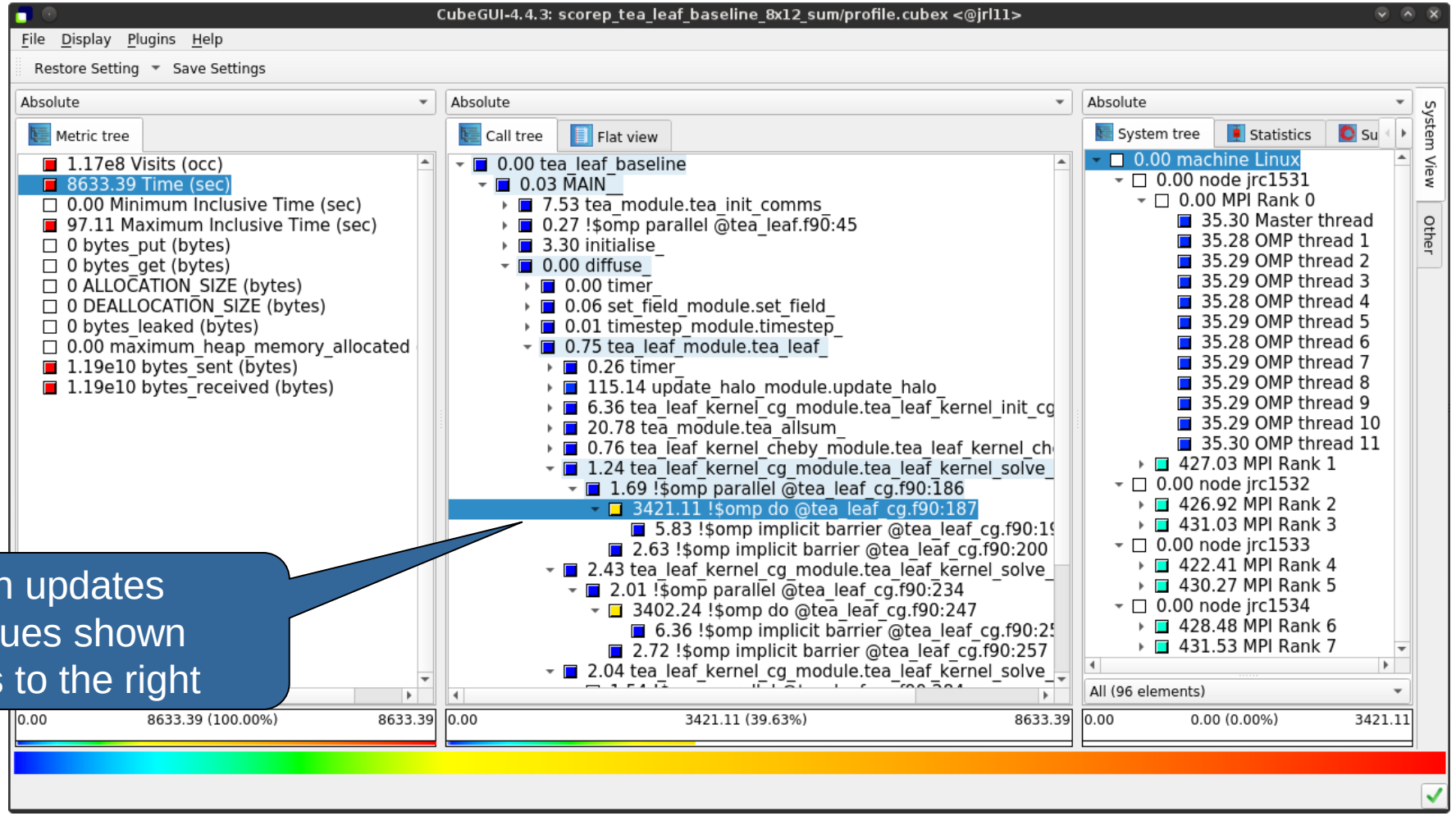
Expanding the system tree



Expanding the call tree

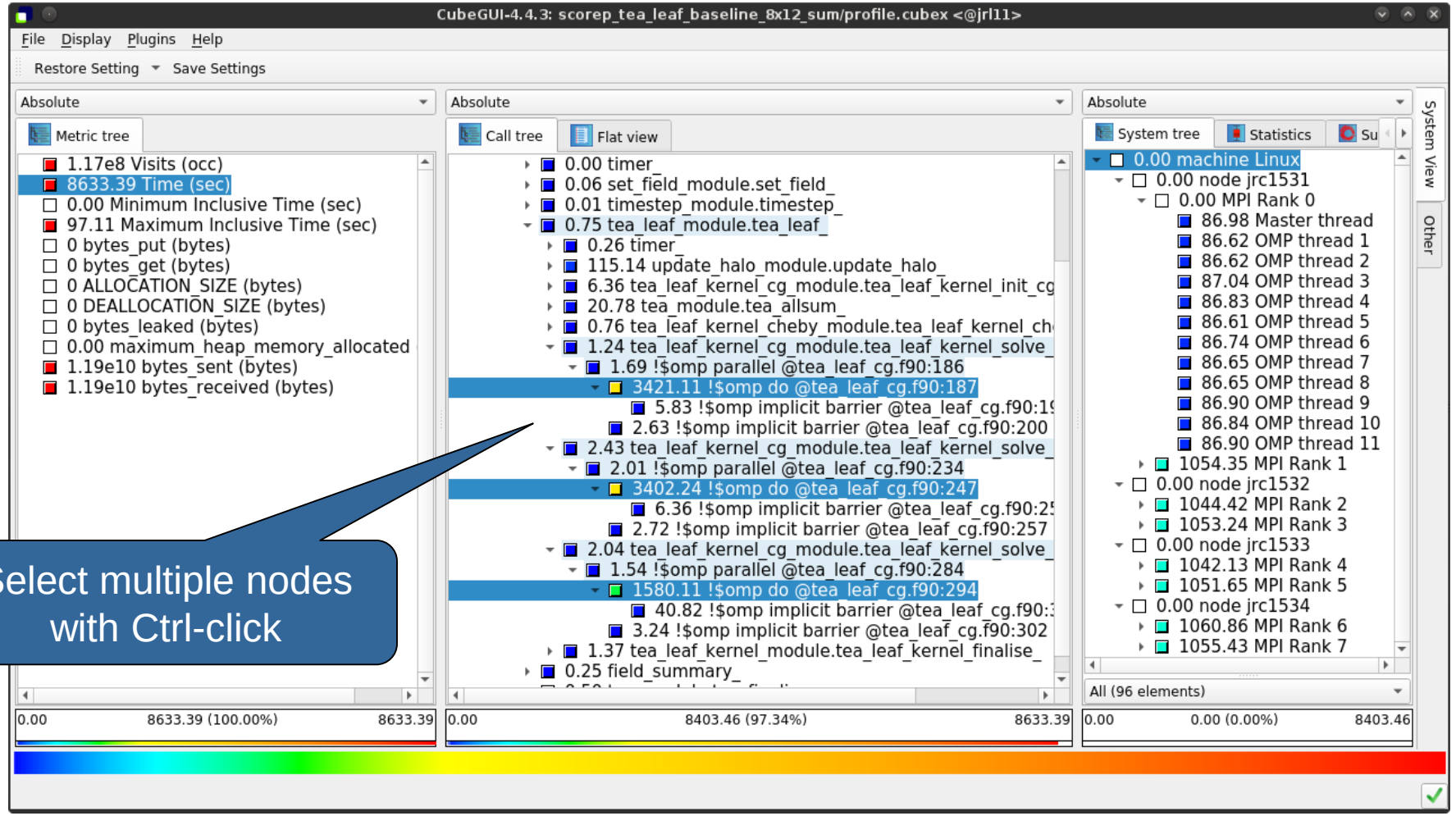


Selecting a call path

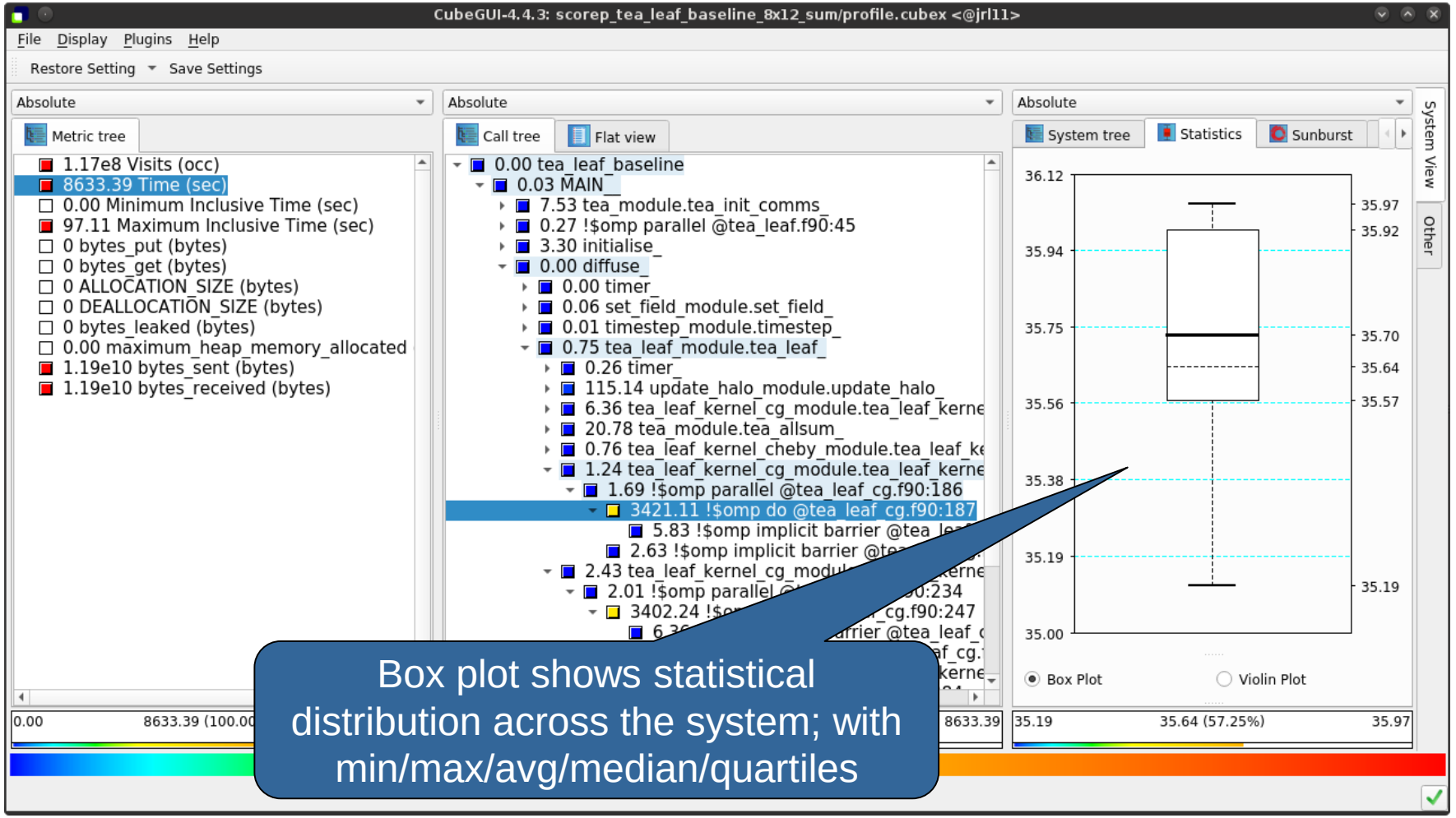


Selection updates metric values shown in columns to the right

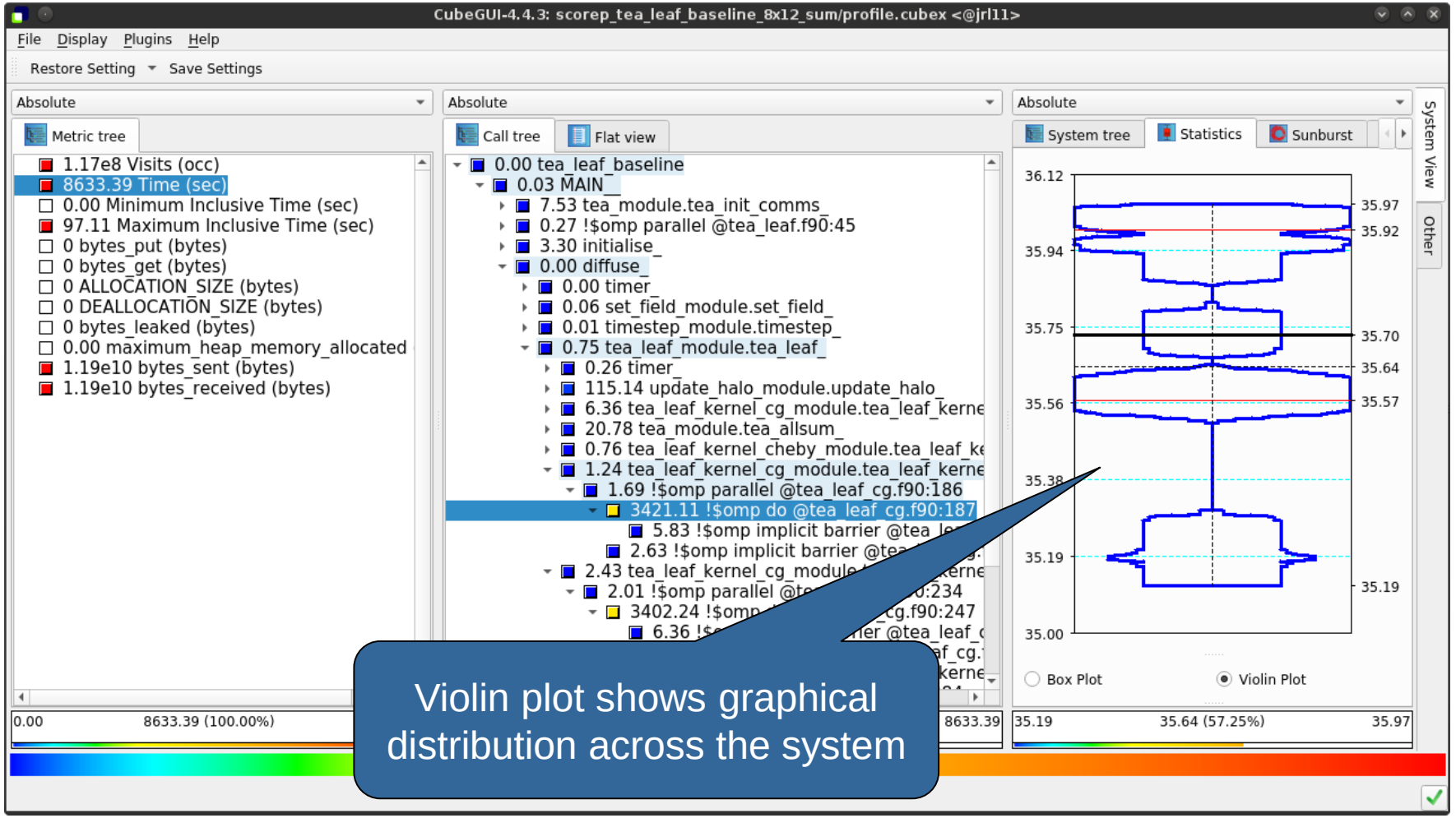
Multiple selection



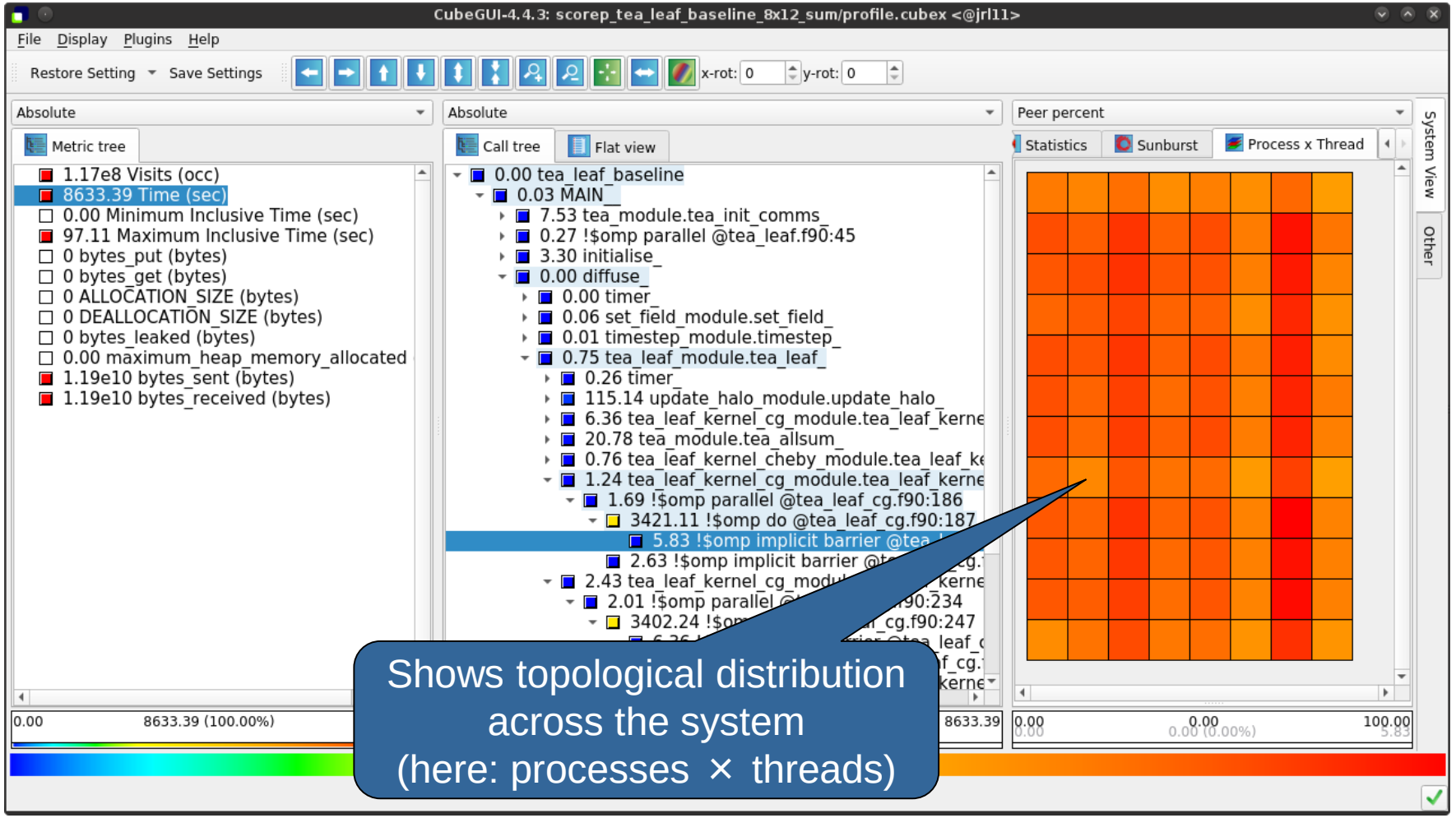
Box plot view



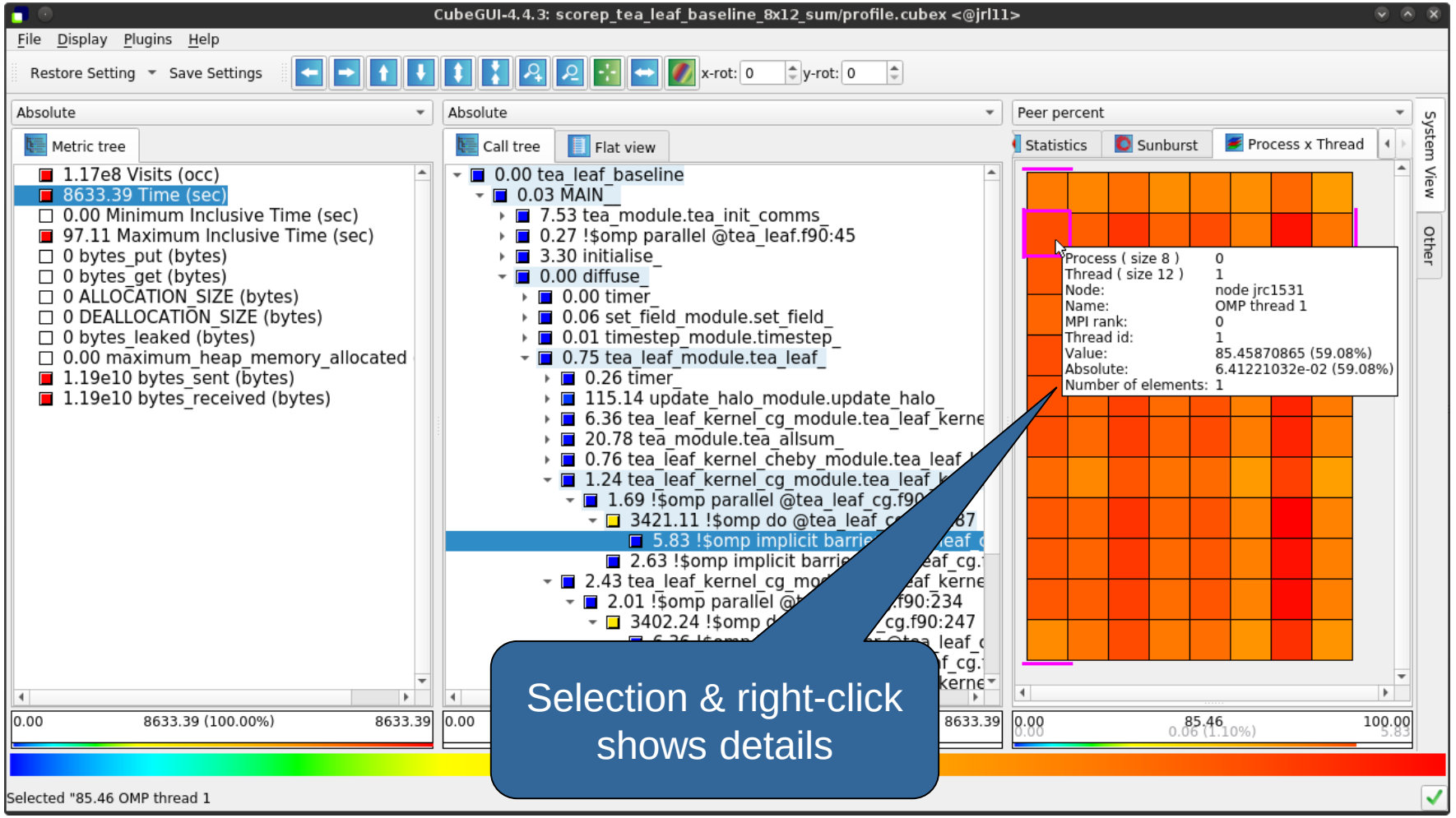
Violin plot view



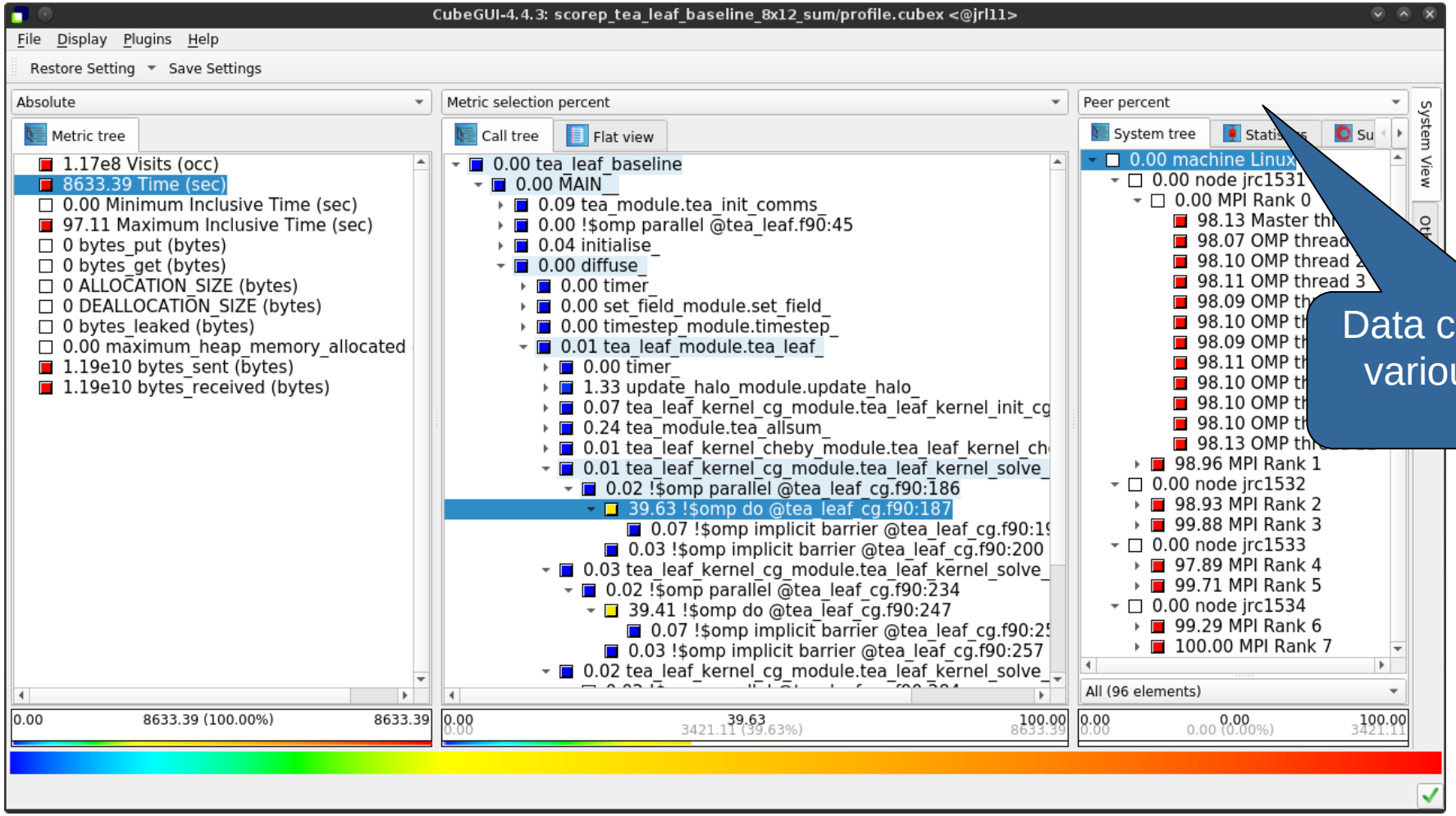
Topology view



Topology view (cont.)



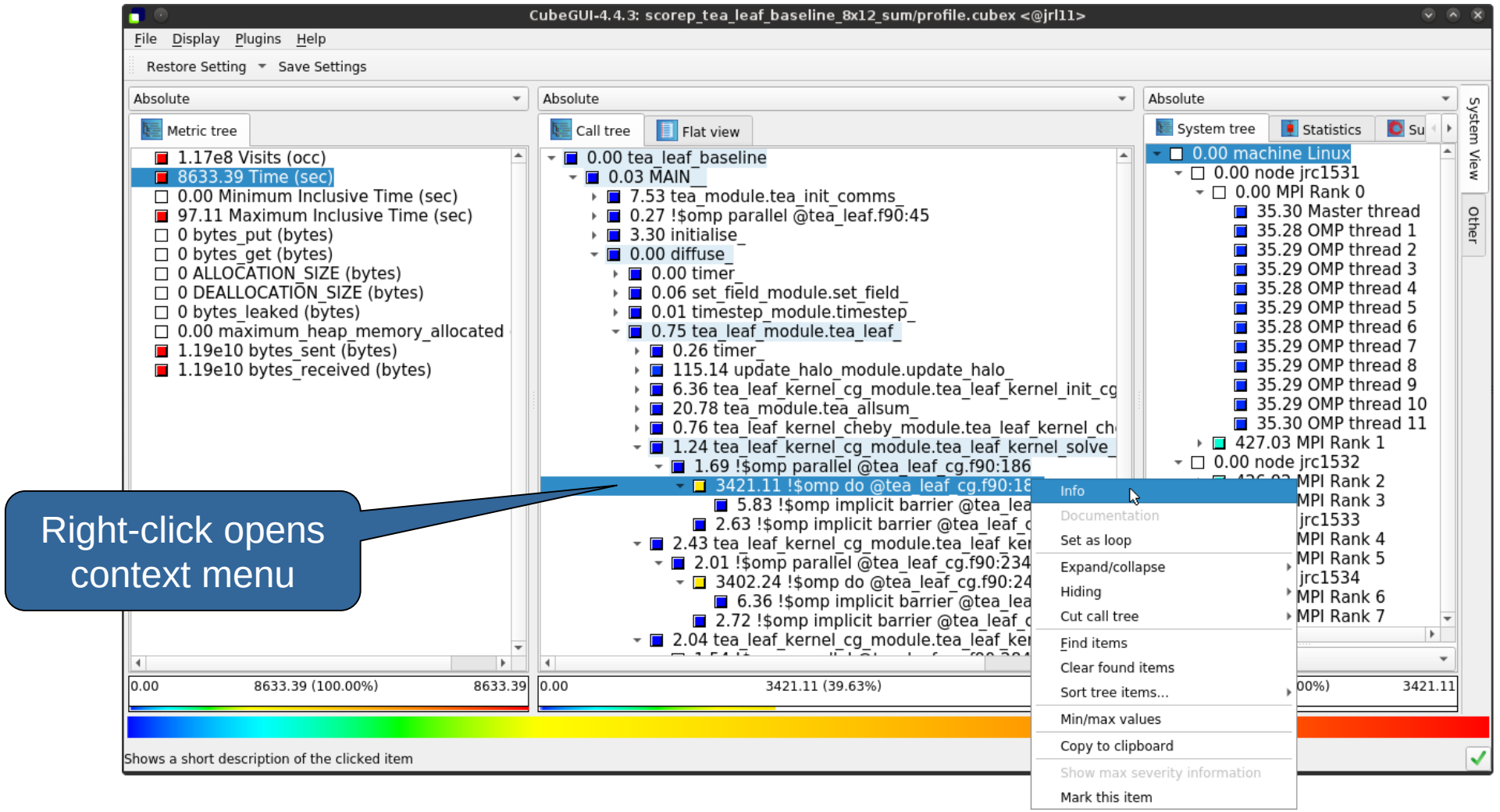
Alternative display modes



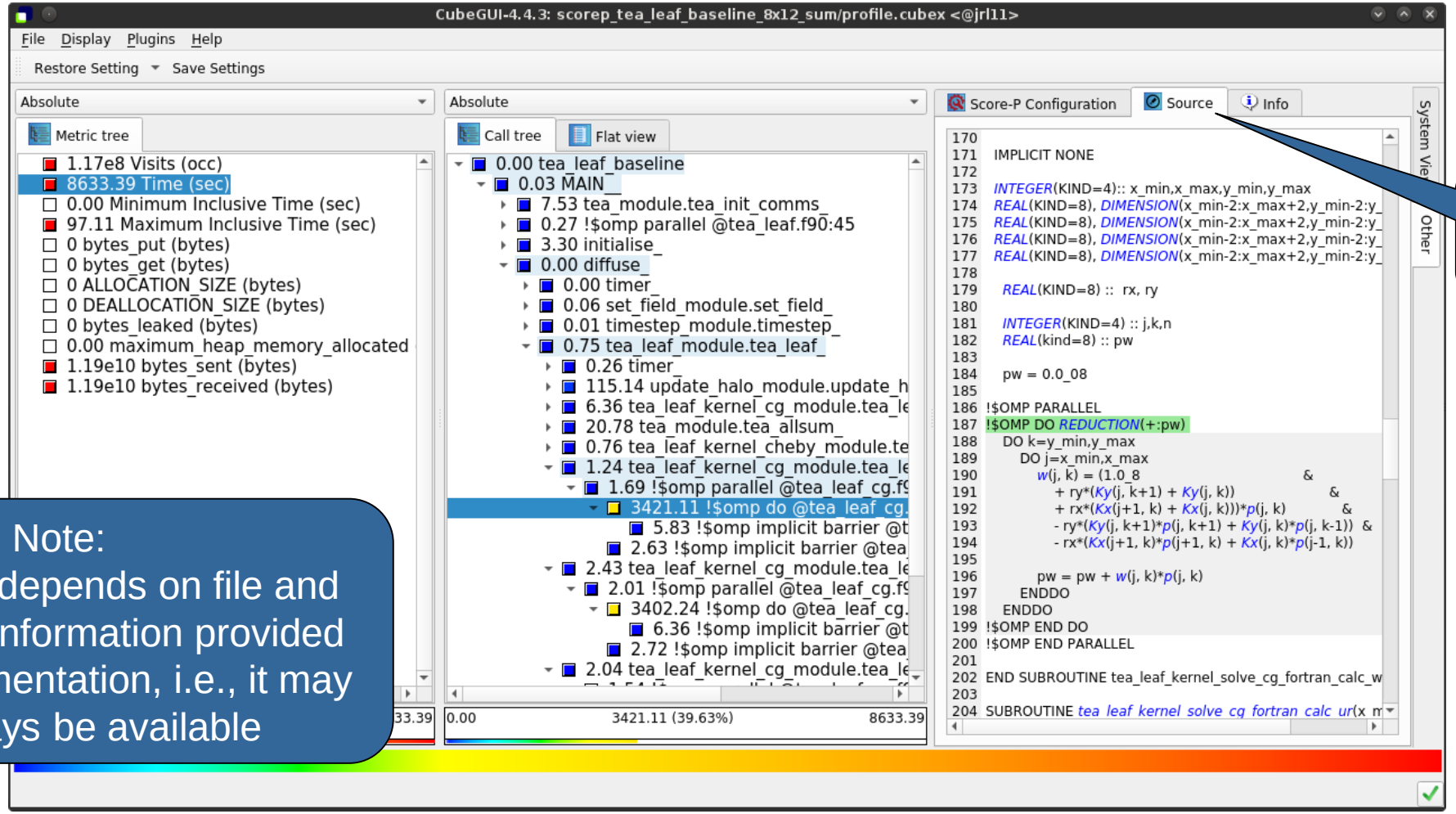
Important display modes

- 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

Source-code view via context menu



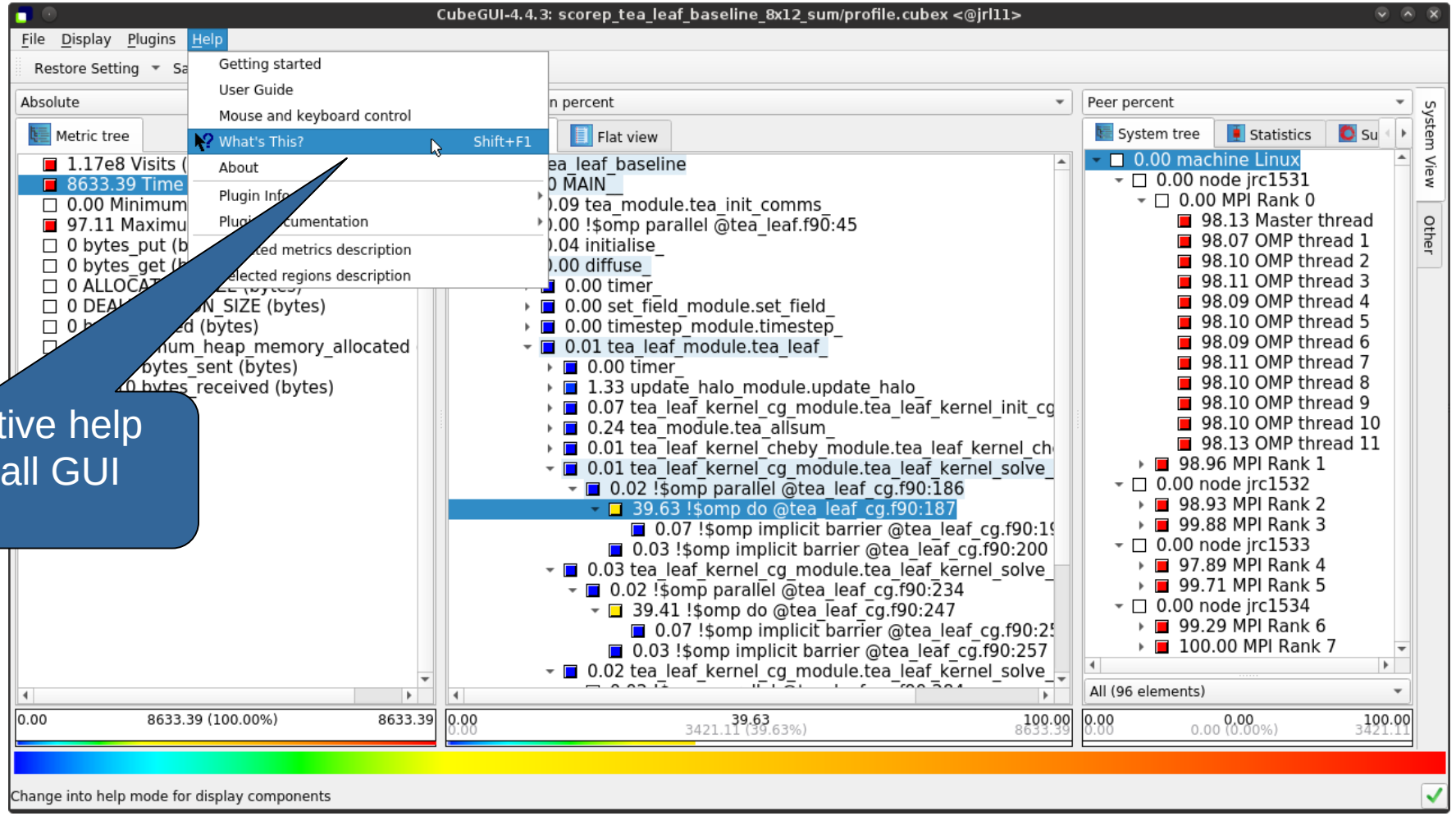
Source-code view



Select “Source” tab

Note:
This feature depends on file and line number information provided by the instrumentation, i.e., it may not always be available

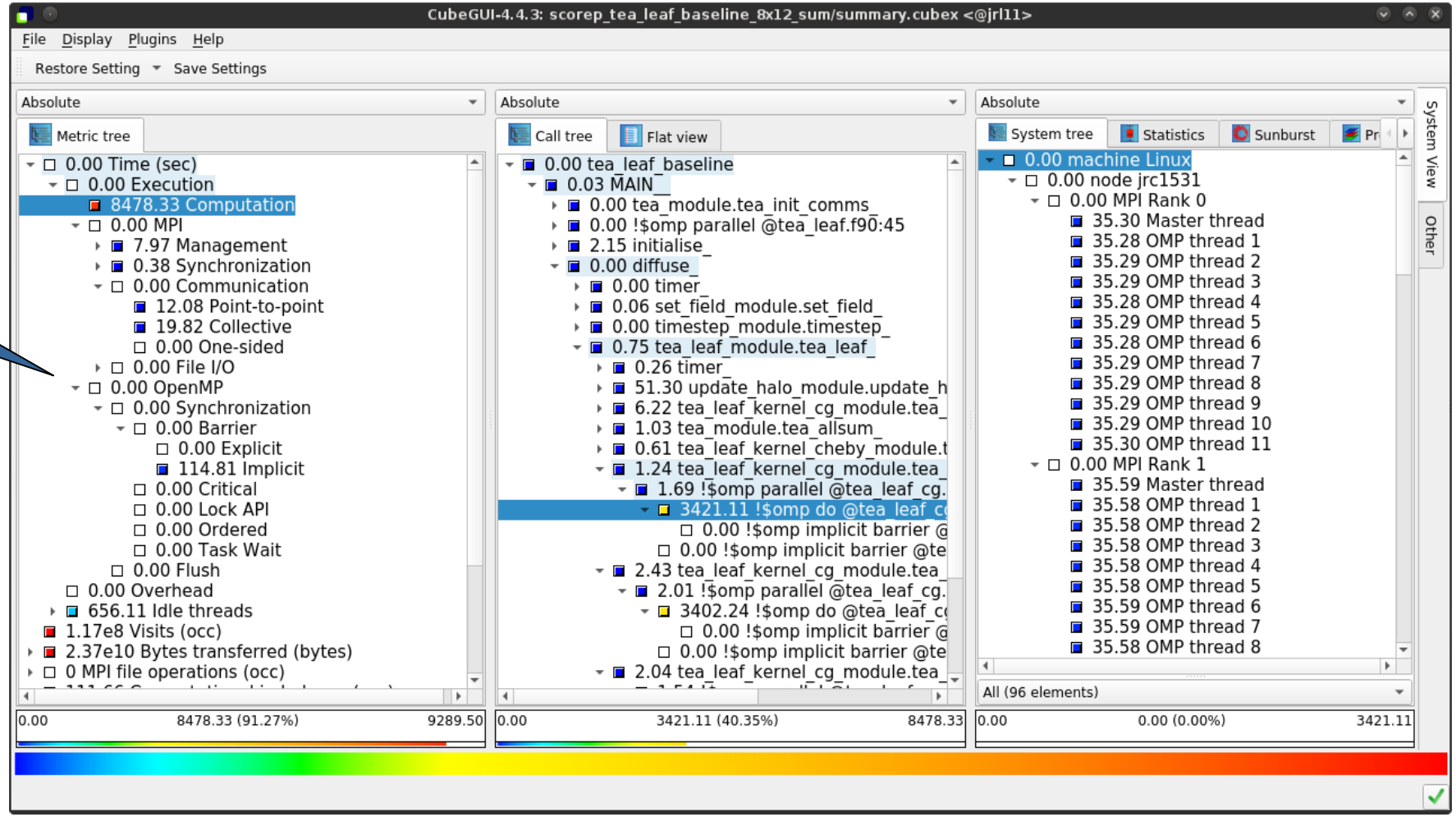
Context-sensitive help



Context-sensitive help available for all GUI items

Post-processed summary analysis report

Split base metrics into more specific metrics



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:
  - https://www.scalasca.org
- User guide also part of installation:
  - <prefix>/share/doc/cubegui/CubeUserGuide.pdf
- Contact:
  - mailto: scalasca@fz-juelich.de

Case study: TeaLeaf

TeaLeaf

- HPC mini-app developed by the UK Mini-App Consortium
  - Solves the linear 2D heat conduction equation on a spatially decomposed regular grid using a 5 point stencil with implicit solvers
  - Part of the Mantevo 3.0 suite
  - Available on GitHub: http://uk-mac.github.io/TeaLeaf/

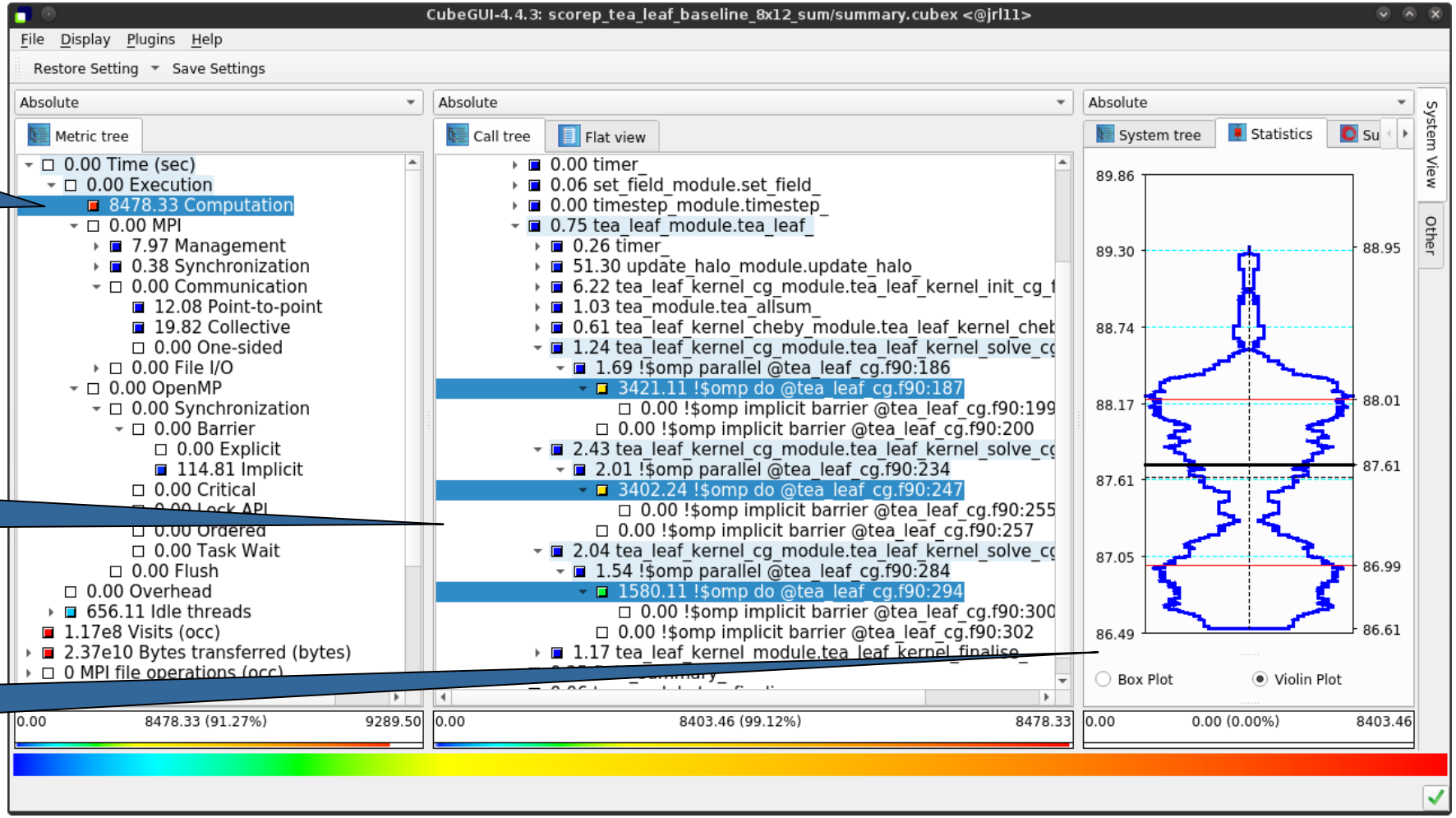
- Measurements of TeaLeaf reference v1.0 taken on (previous) Jureca cluster @ JSC
  - Using Intel 19.0.3 compilers, Intel MPI 2019.3, and Score-P 5.0
  - Run configuration
    - 8 MPI ranks with 12 OpenMP threads each
    - Distributed across 4 compute nodes (2 ranks per node)
    - Test problem “5”: 4000 × 4000 cells, CG solver

TeaLeaf summary report analysis (I)

91% of the execution time is computation…

…almost entirely spent in 3 OpenMP `do` loops…

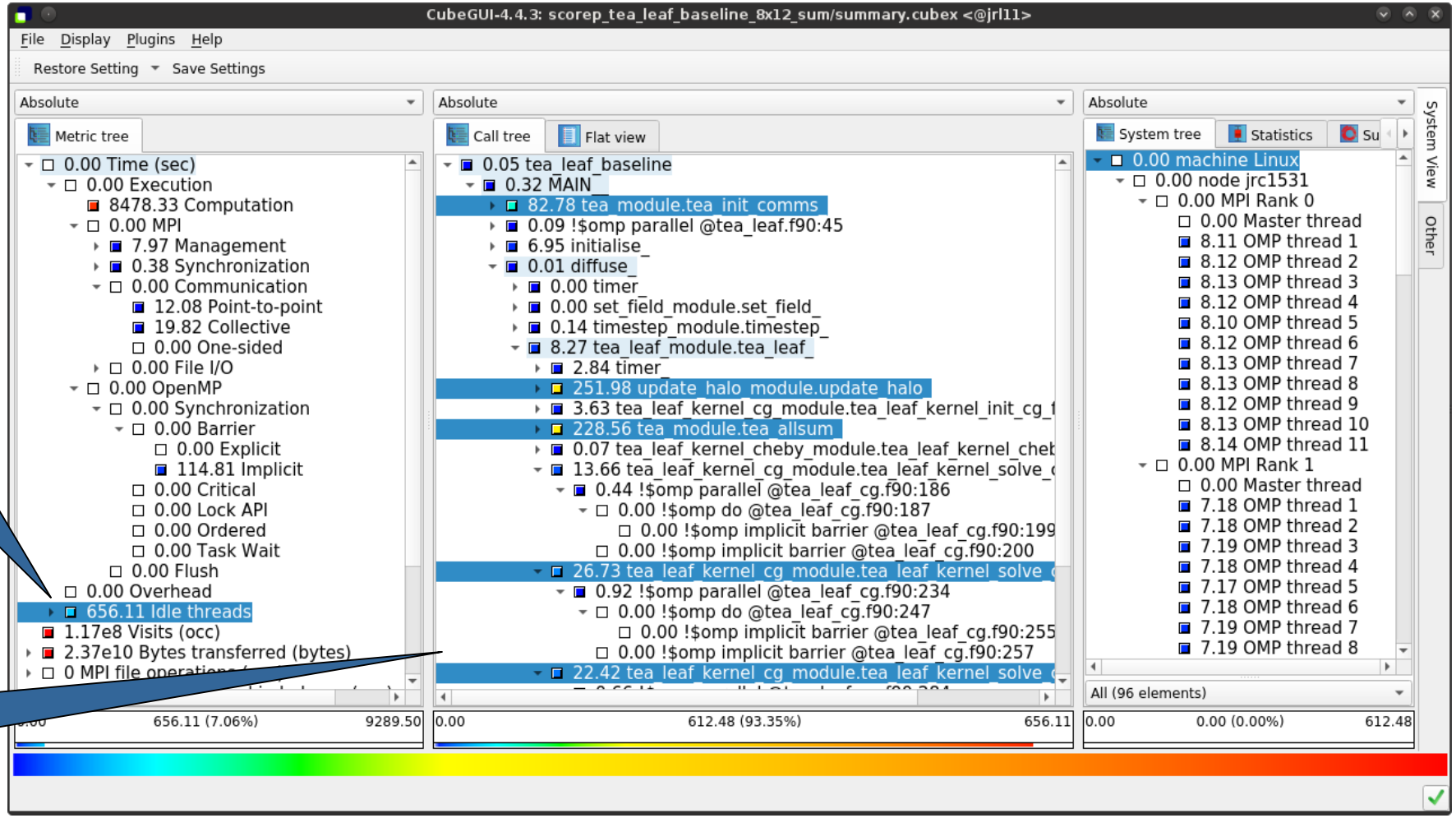
…with a slight imbalance across ranks & threads



TeaLeaf summary report analysis (II)

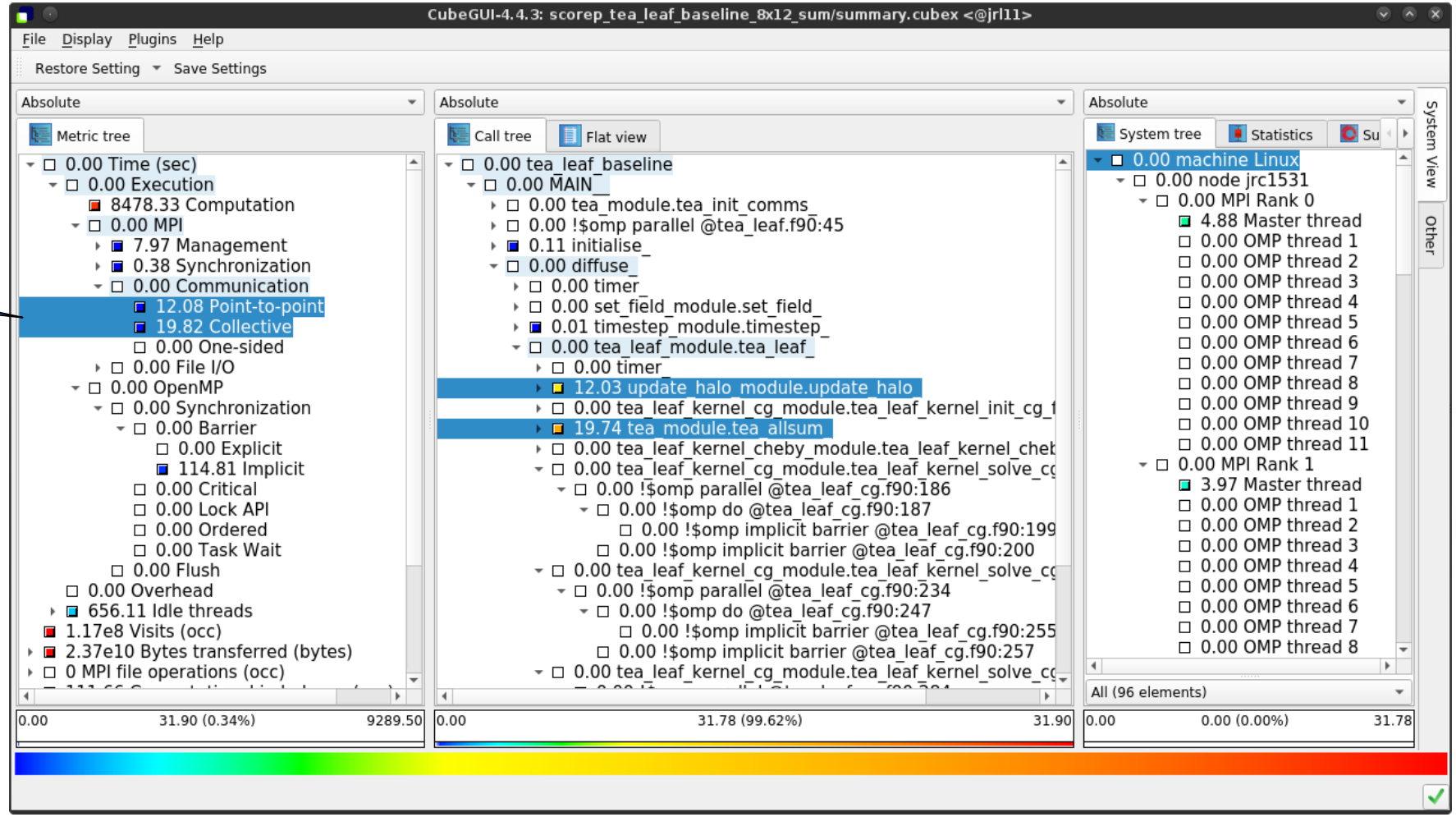
7% of the execution time are lost due to idle threads…

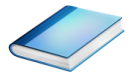
…in non-OpenMP parallelized code regions



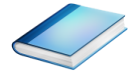
TeaLeaf summary report analysis (III)

MPI communication time is negligible (0.34%); communication is only on the master threads (MPI_THREAD_FUNNELED)



Reference material 

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

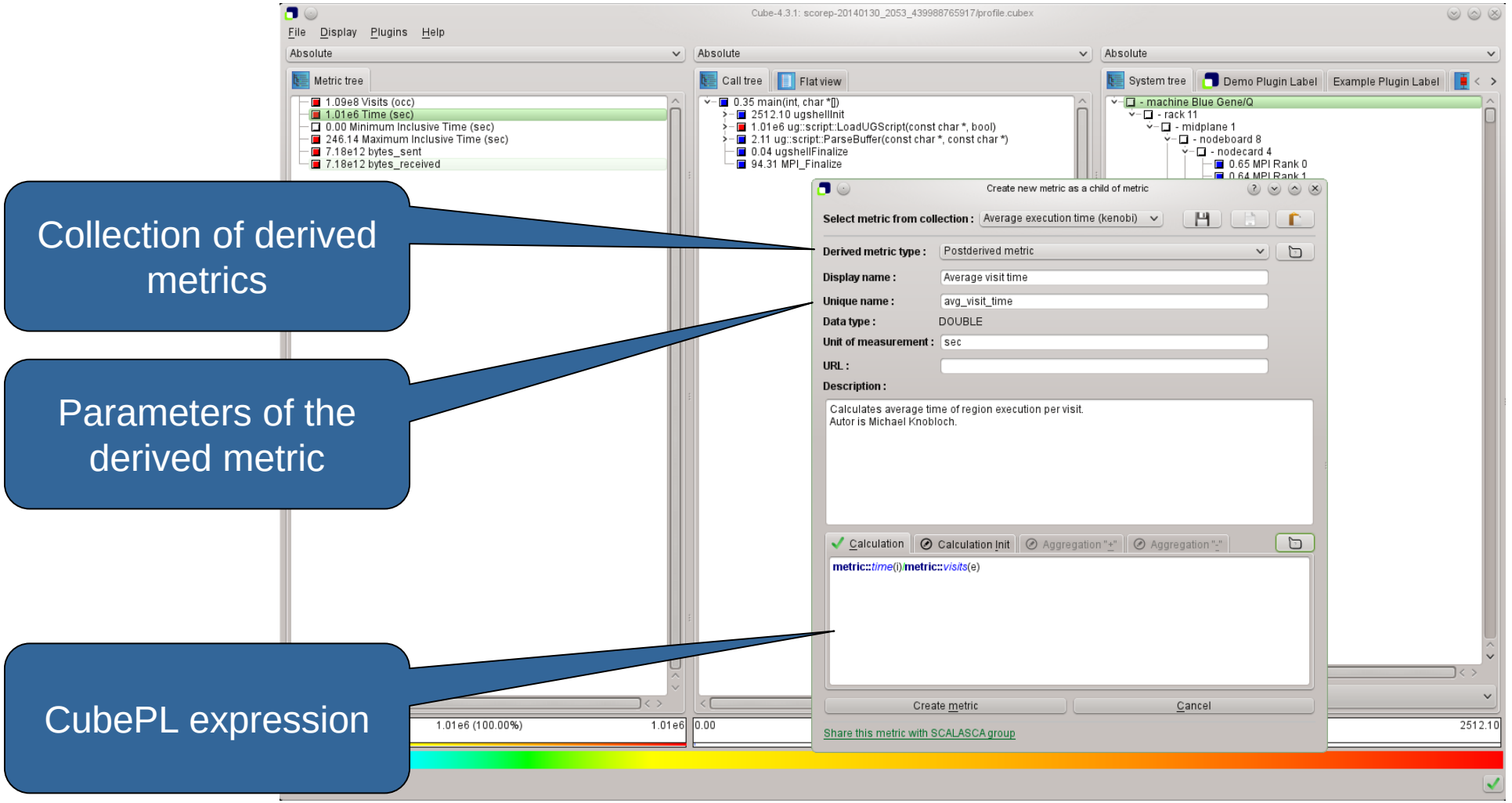
- Examples:
  - "Average execution time": Postderived metric with expression

    metric::time(i)/metric::visits(e)

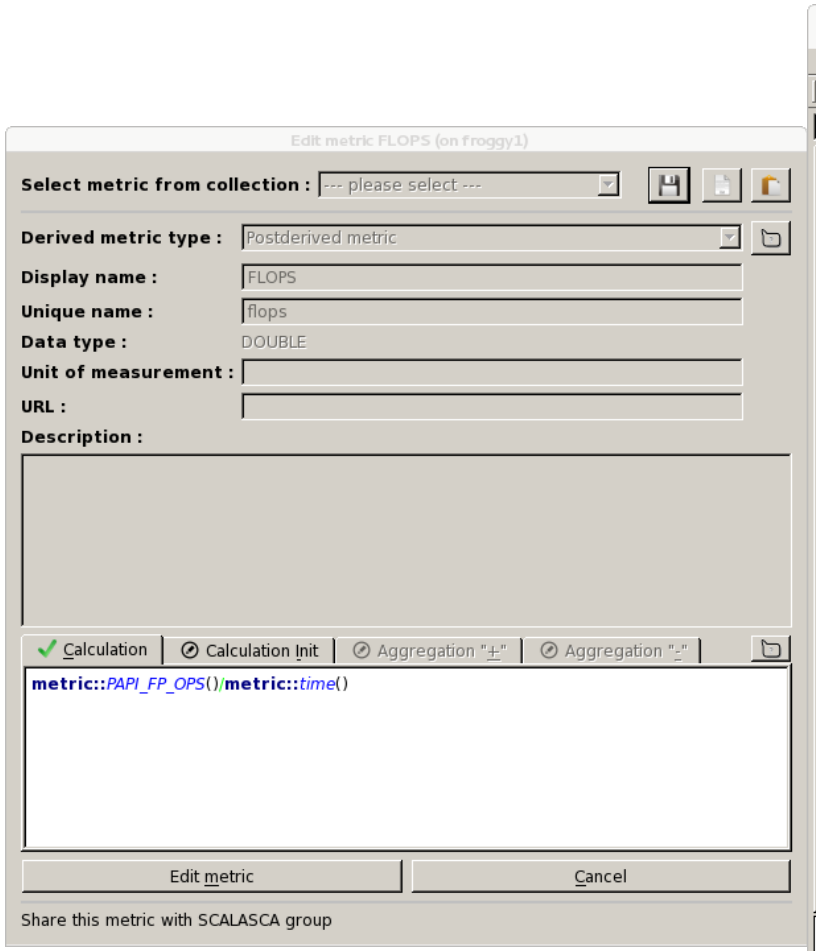
  - "Number of FLOP per second": Postderived metric with expression

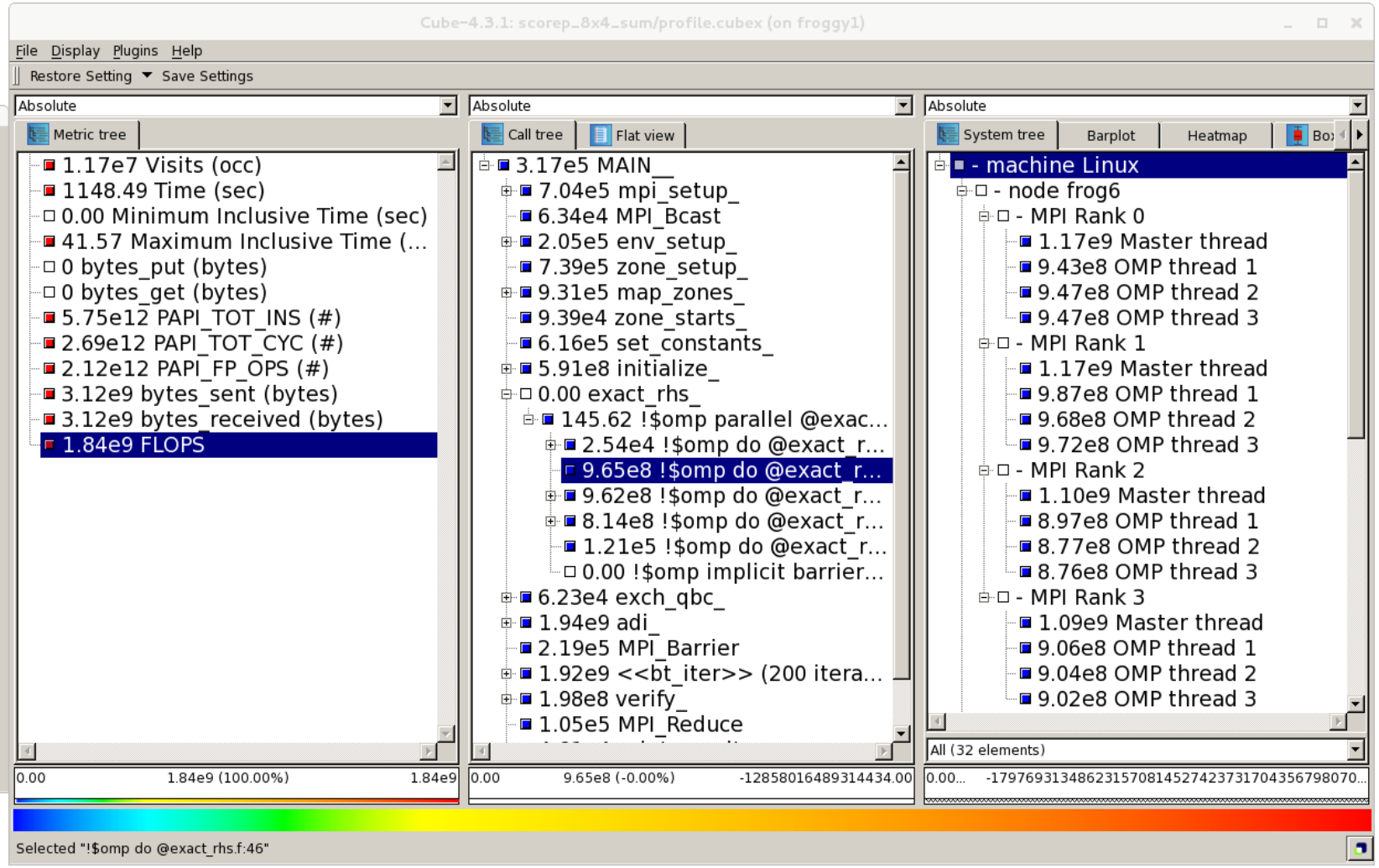
    metric::FLOP()/metric::time()

Derived metrics in Cube GUI

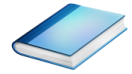


Example: FLOPS based on PAPI_FP_OPS and time





CUBE algebra utilities



- Extracting solver sub-tree from analysis report

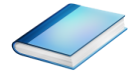
```
% cube_cut  -r '<<ITERATION>>'  scorep_bt-mz_C_32x4_sum/profile.cubex
Writing cut.cubex... done.
```

- Calculating difference of two reports

```
% cube_diff  scorep_bt-mz_C_32x4_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

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

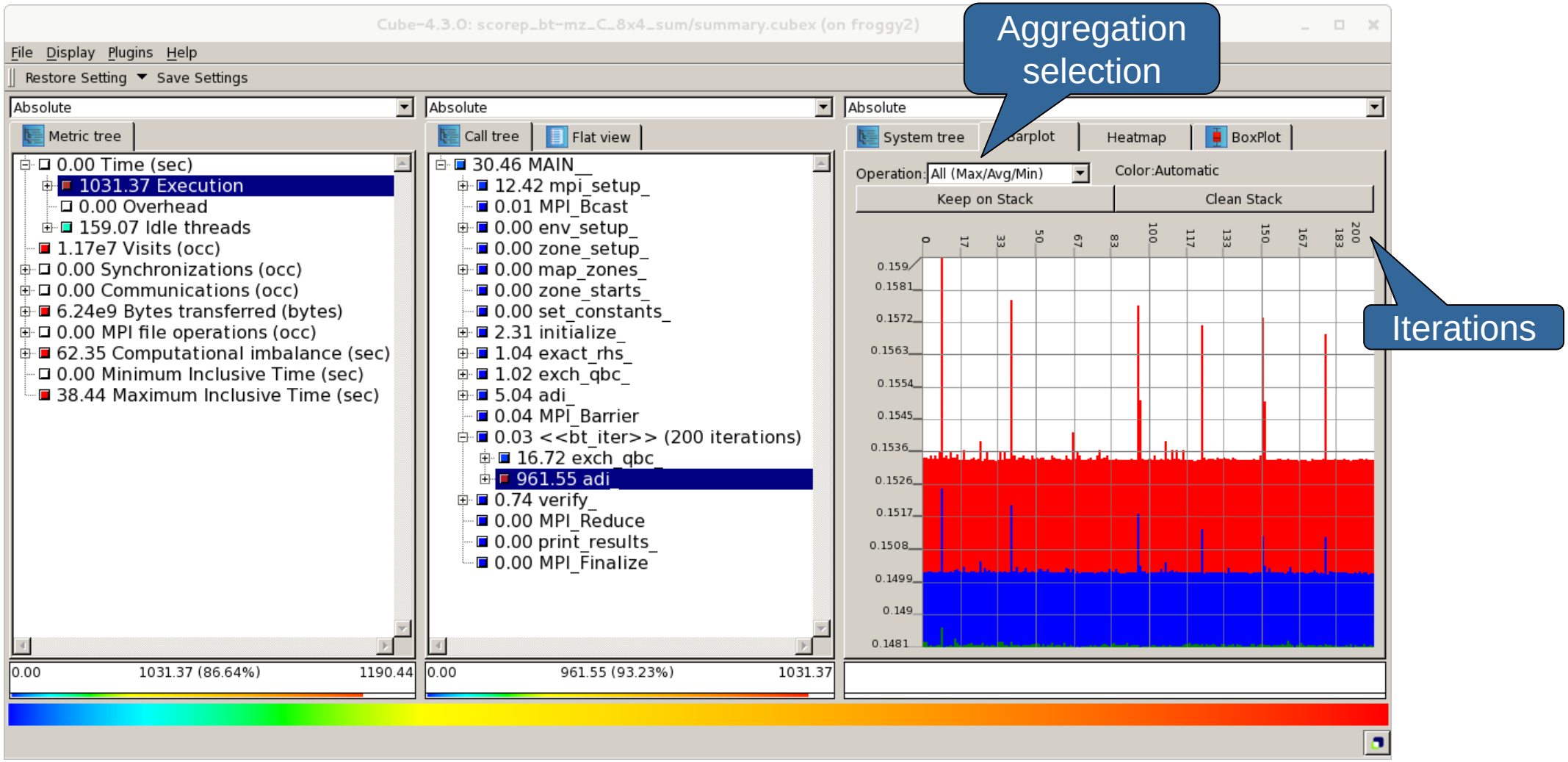
➢Useful for checking results for specific iterations

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