

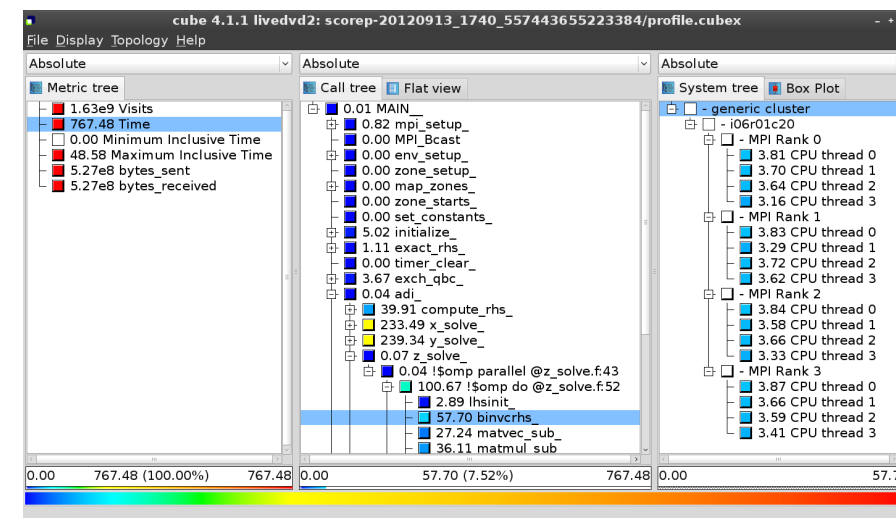
Analysis report examination with Cube

Michael Knobloch
Jülich Supercomputing Centre



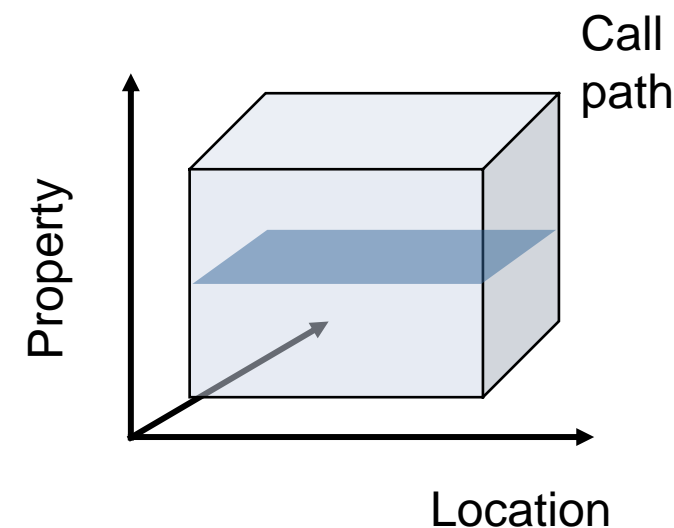
Cube

- Parallel program analysis report exploration tools
 - Libraries for XML+binary report reading & writing
 - Algebra utilities for report processing
 - GUI for interactive analysis exploration
 - Requires Qt4 ≥ 4.6 or Qt 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 v4.4.1 (September 2018)

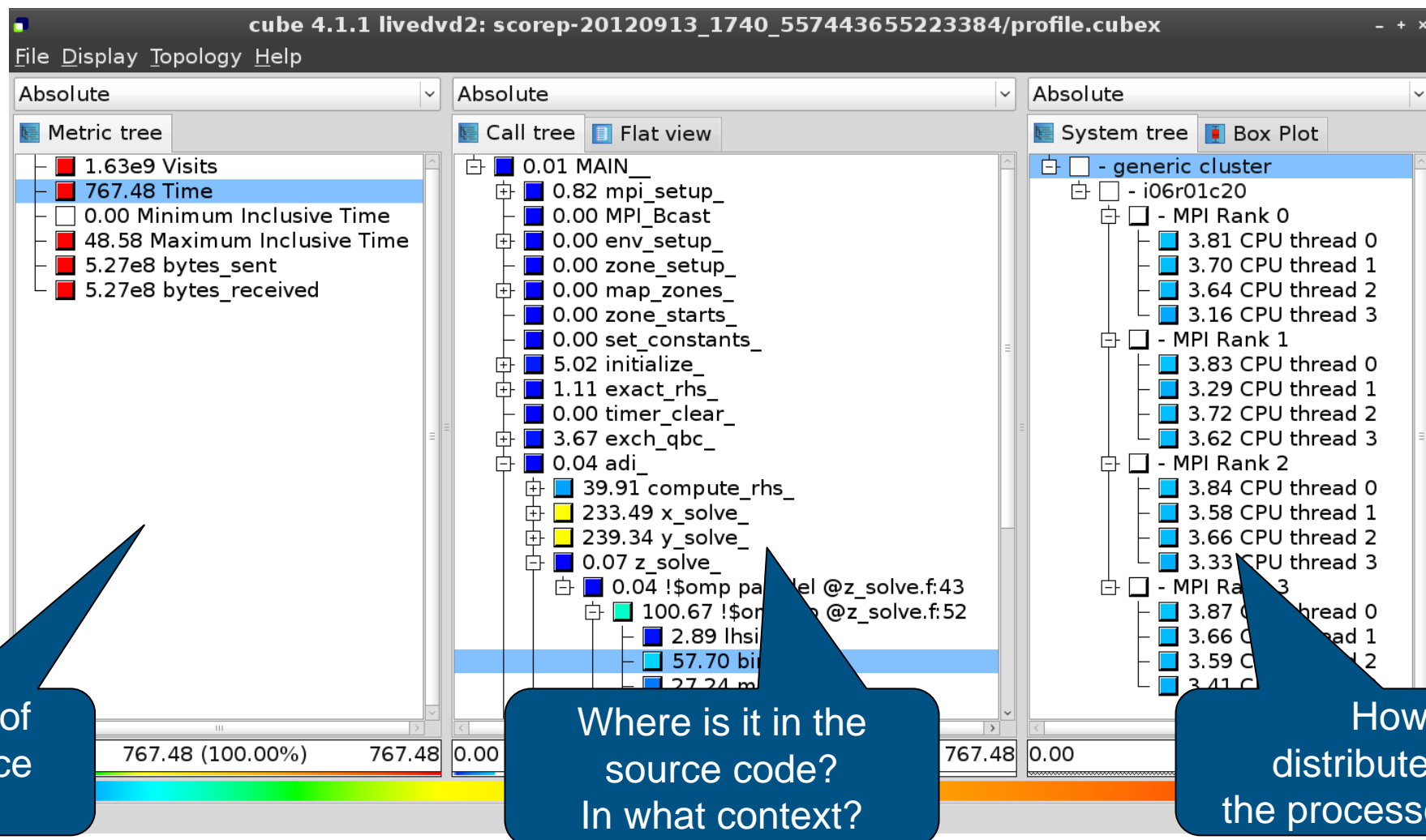


Analysis presentation and exploration

- 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

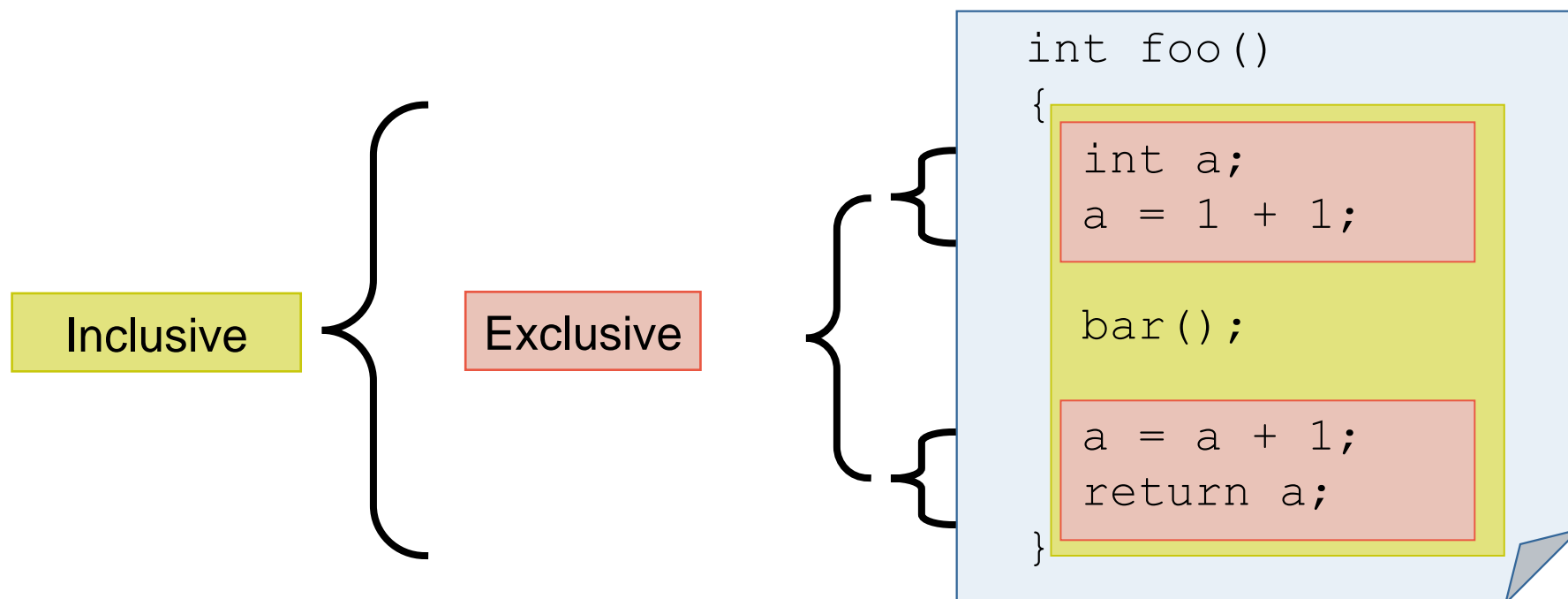


Analysis presentation

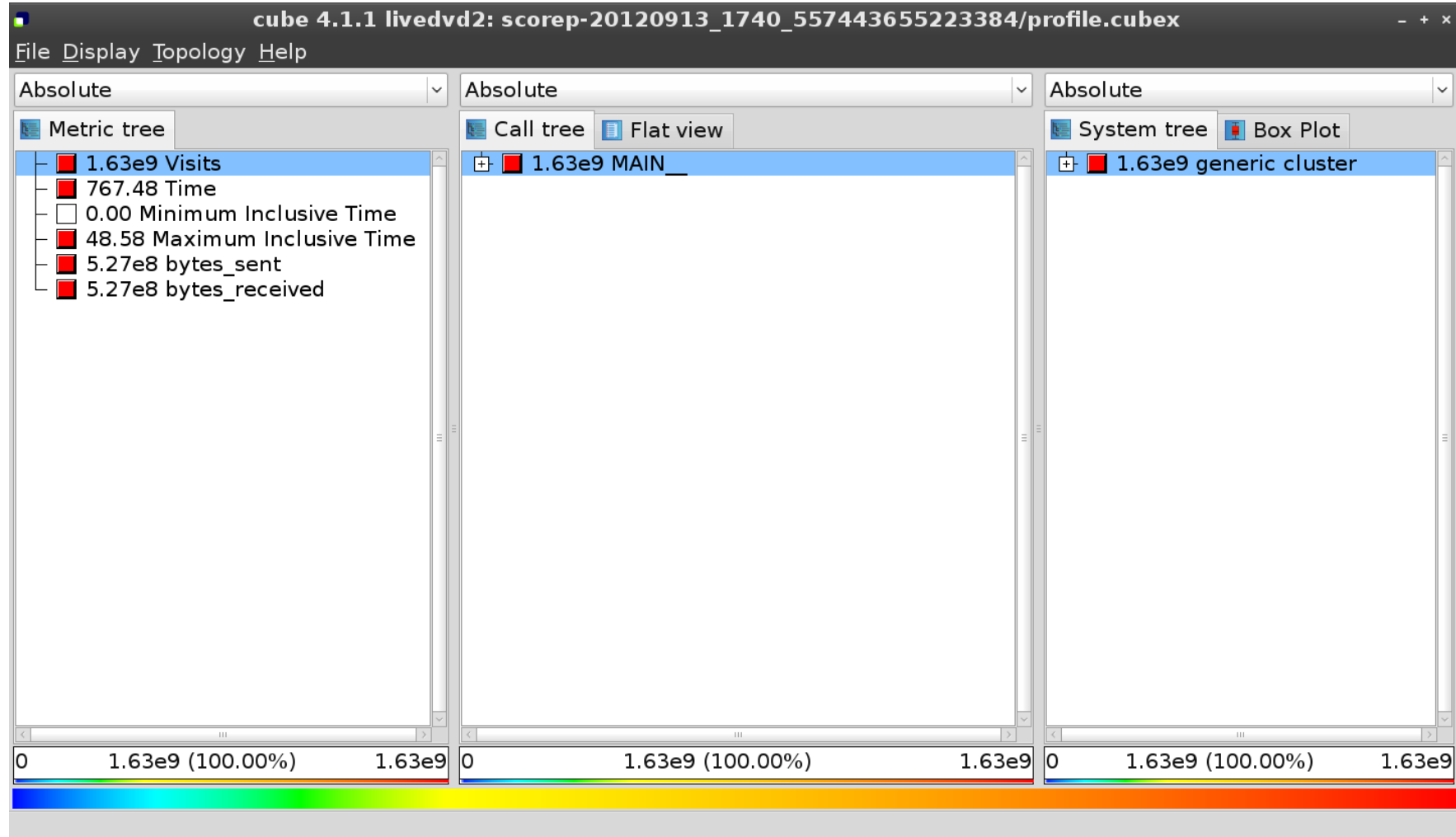


Inclusive vs. exclusive values

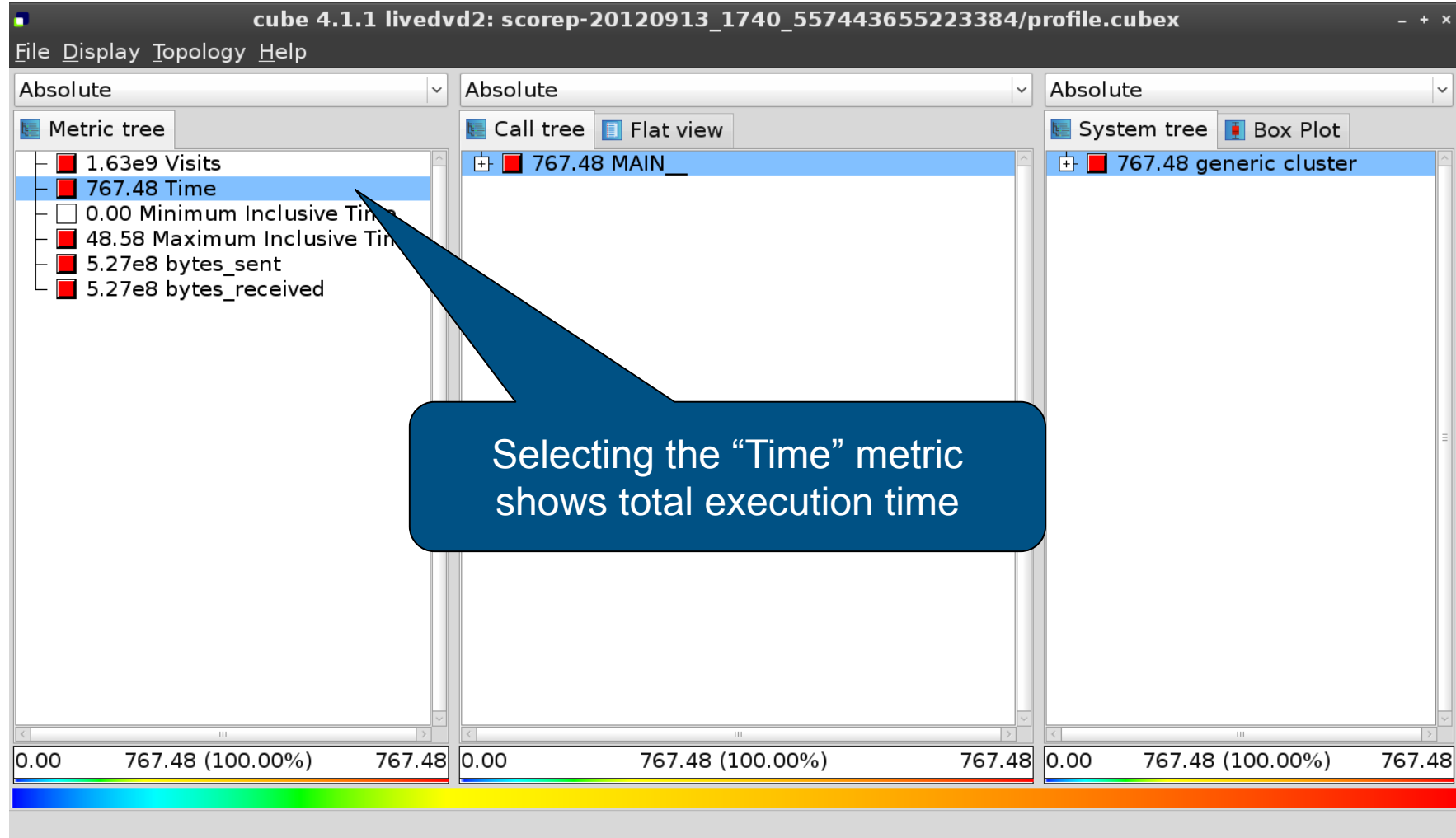
- Inclusive
 - Information of all sub-elements aggregated into single value
- Exclusive
 - Information cannot be subdivided further



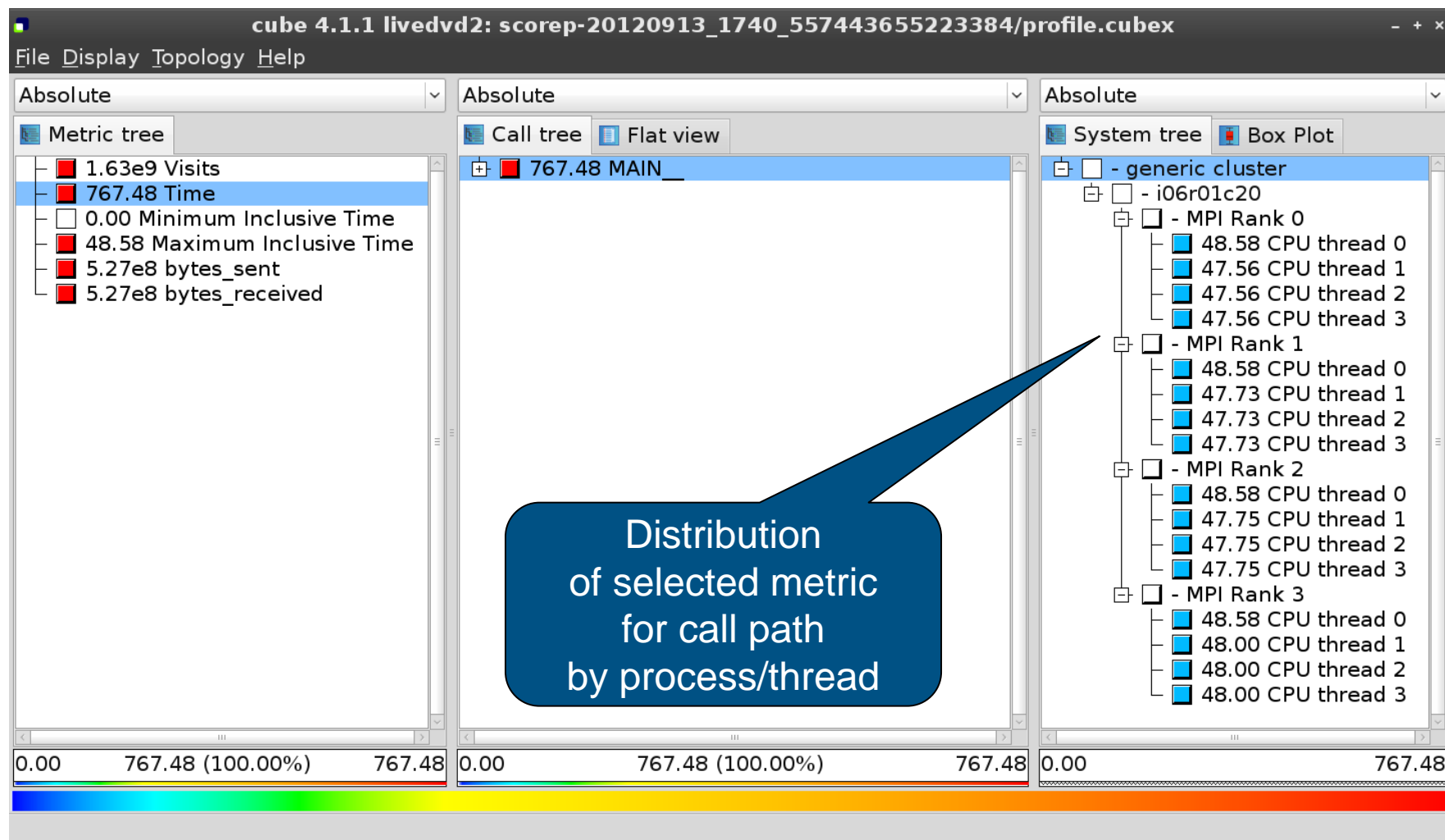
Score-P analysis report exploration (opening view)



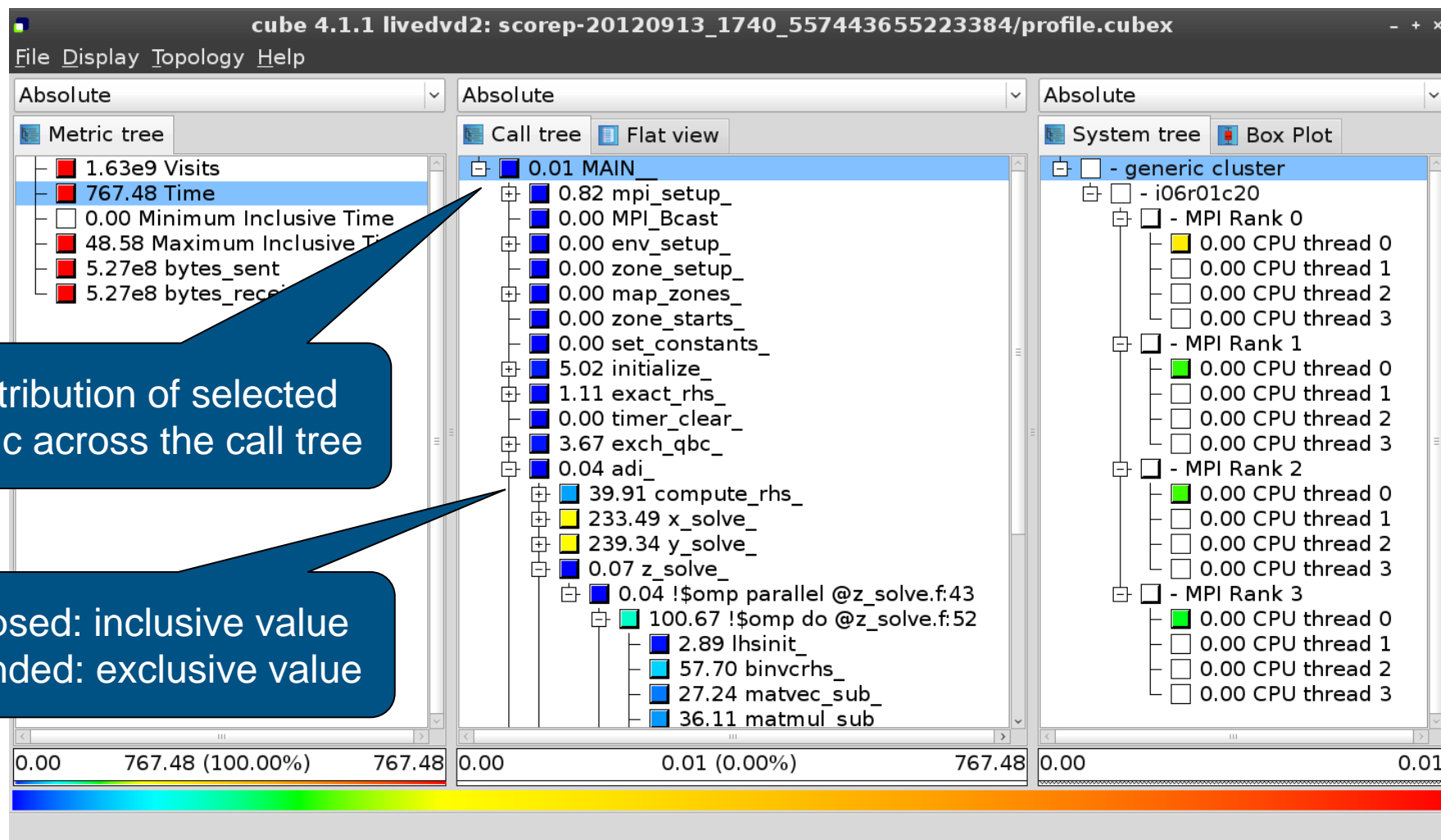
Metric selection



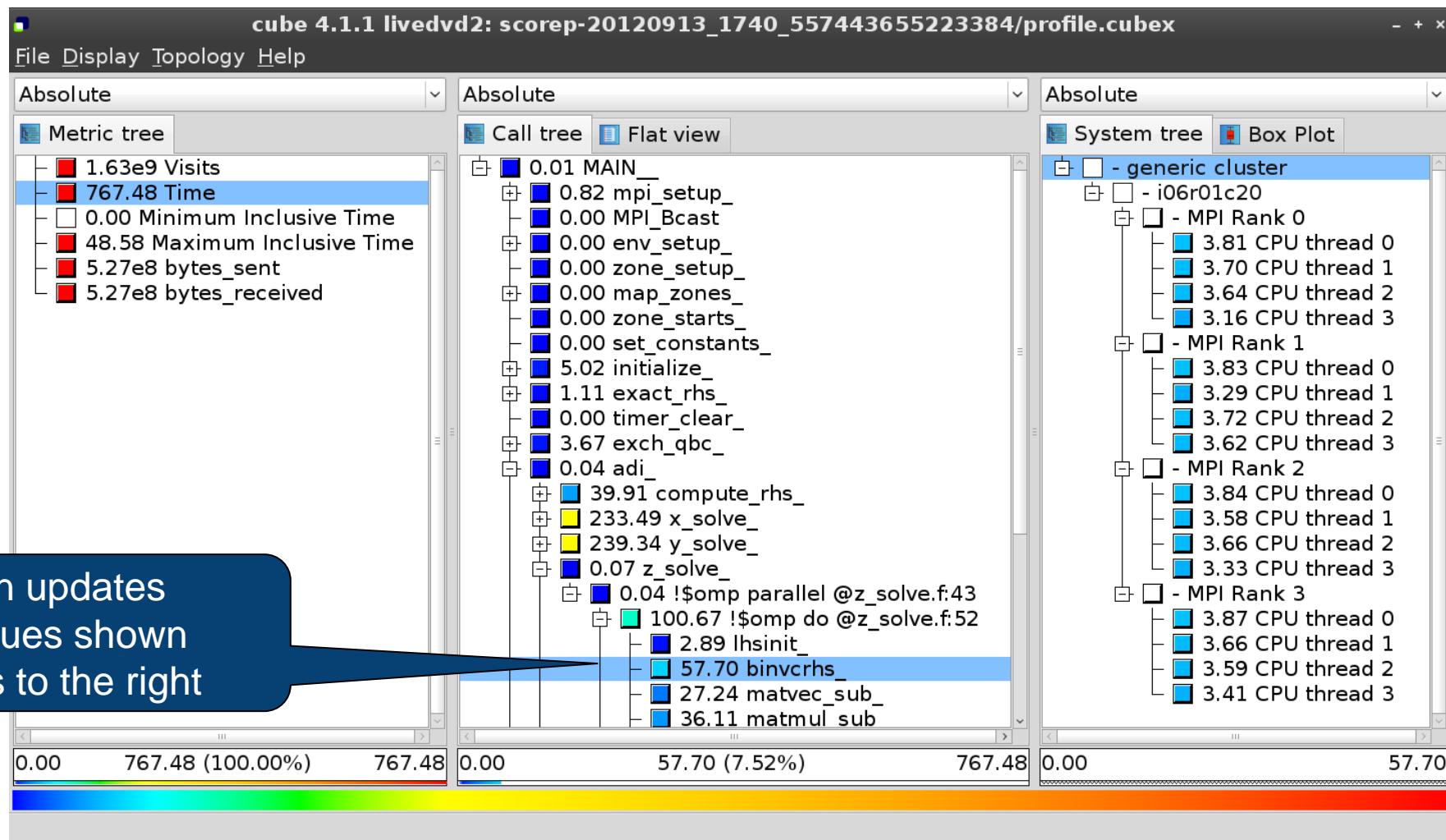
Expanding the system tree



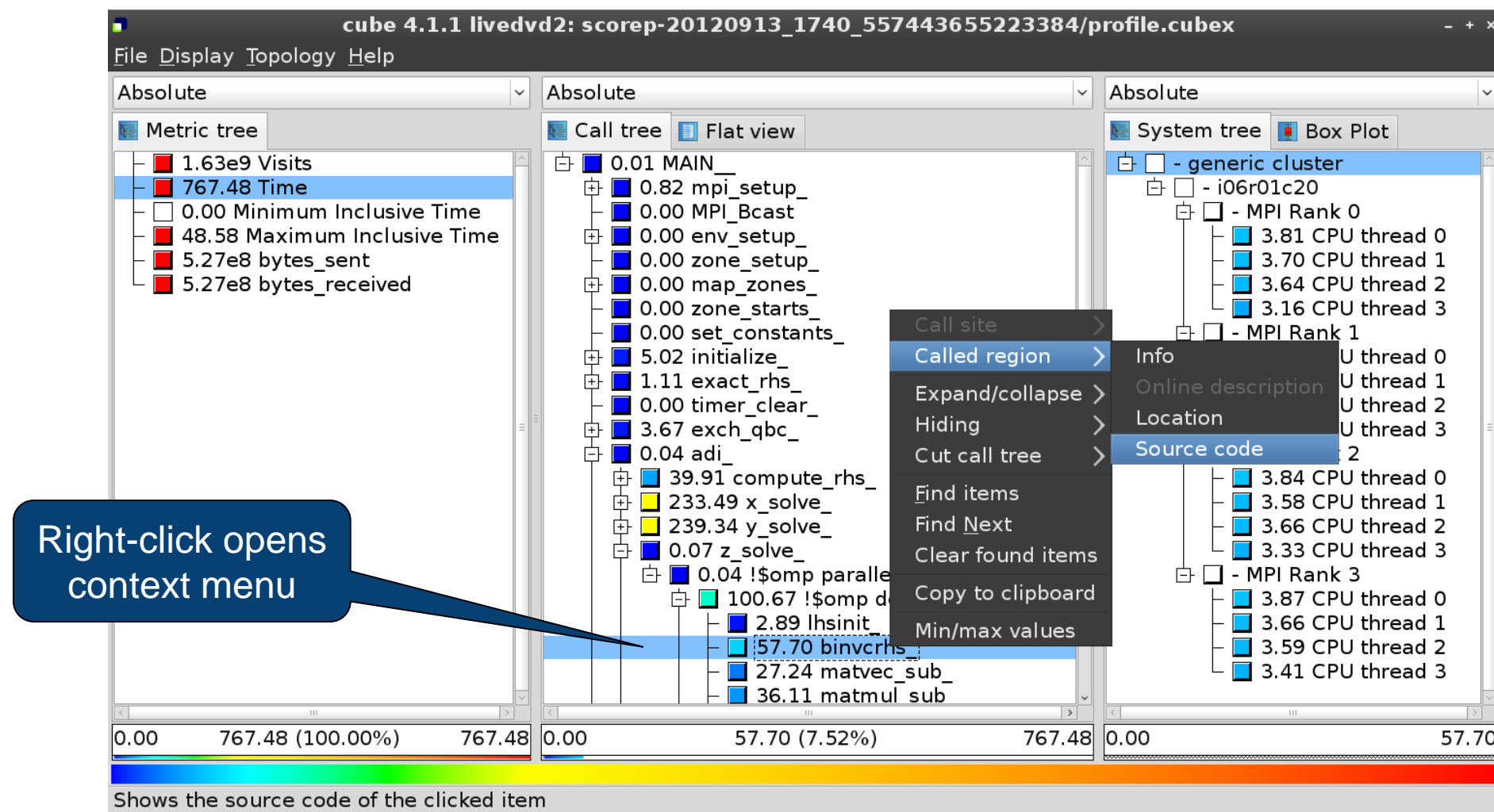
Expanding the call tree



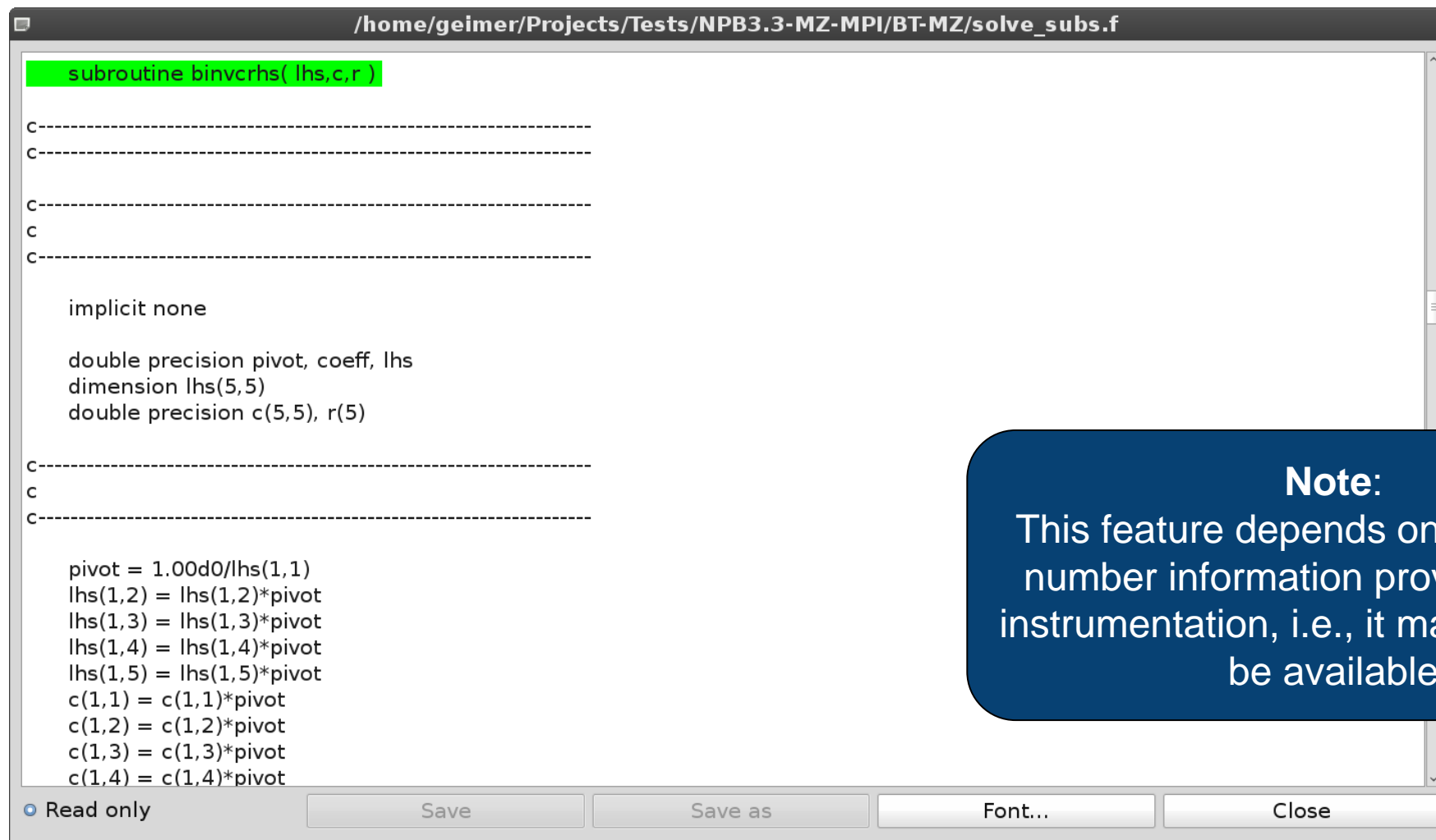
Selecting a call path



Source-code view via context menu



Source-code view



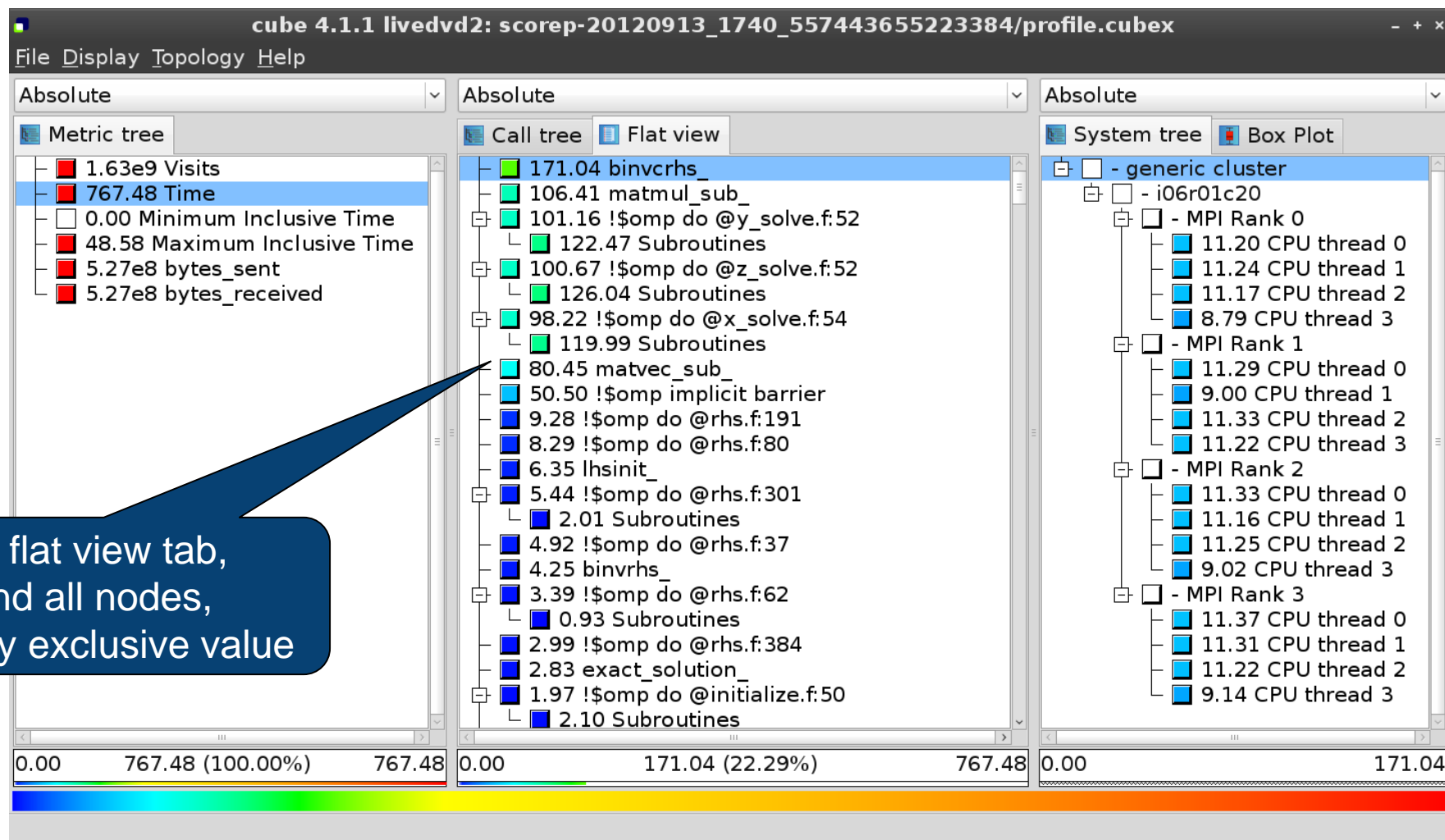
```
subroutine binvrhs( lhs,c,r )  
  
C-----  
C-----  
  
C-----  
C  
C-----  
  
implicit none  
  
double precision pivot, coeff, lhs  
dimension lhs(5,5)  
double precision c(5,5), r(5)  
  
C-----  
C  
C-----  
  
pivot = 1.00d0/lhs(1,1)  
lhs(1,2) = lhs(1,2)*pivot  
lhs(1,3) = lhs(1,3)*pivot  
lhs(1,4) = lhs(1,4)*pivot  
lhs(1,5) = lhs(1,5)*pivot  
c(1,1) = c(1,1)*pivot  
c(1,2) = c(1,2)*pivot  
c(1,3) = c(1,3)*pivot  
c(1,4) = c(1,4)*pivot
```

☒ Read only Save Save as Font... Close

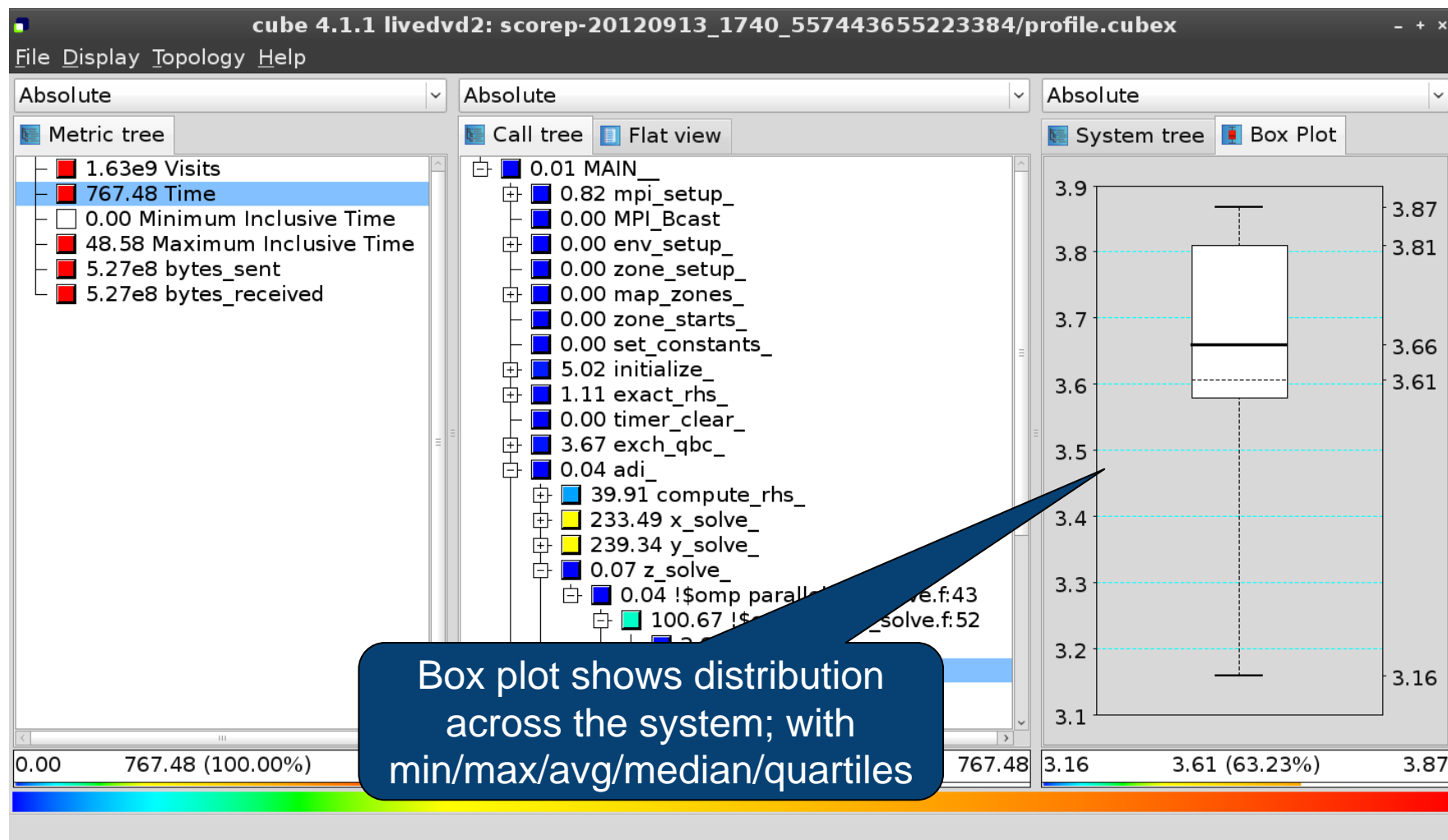
Note:

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

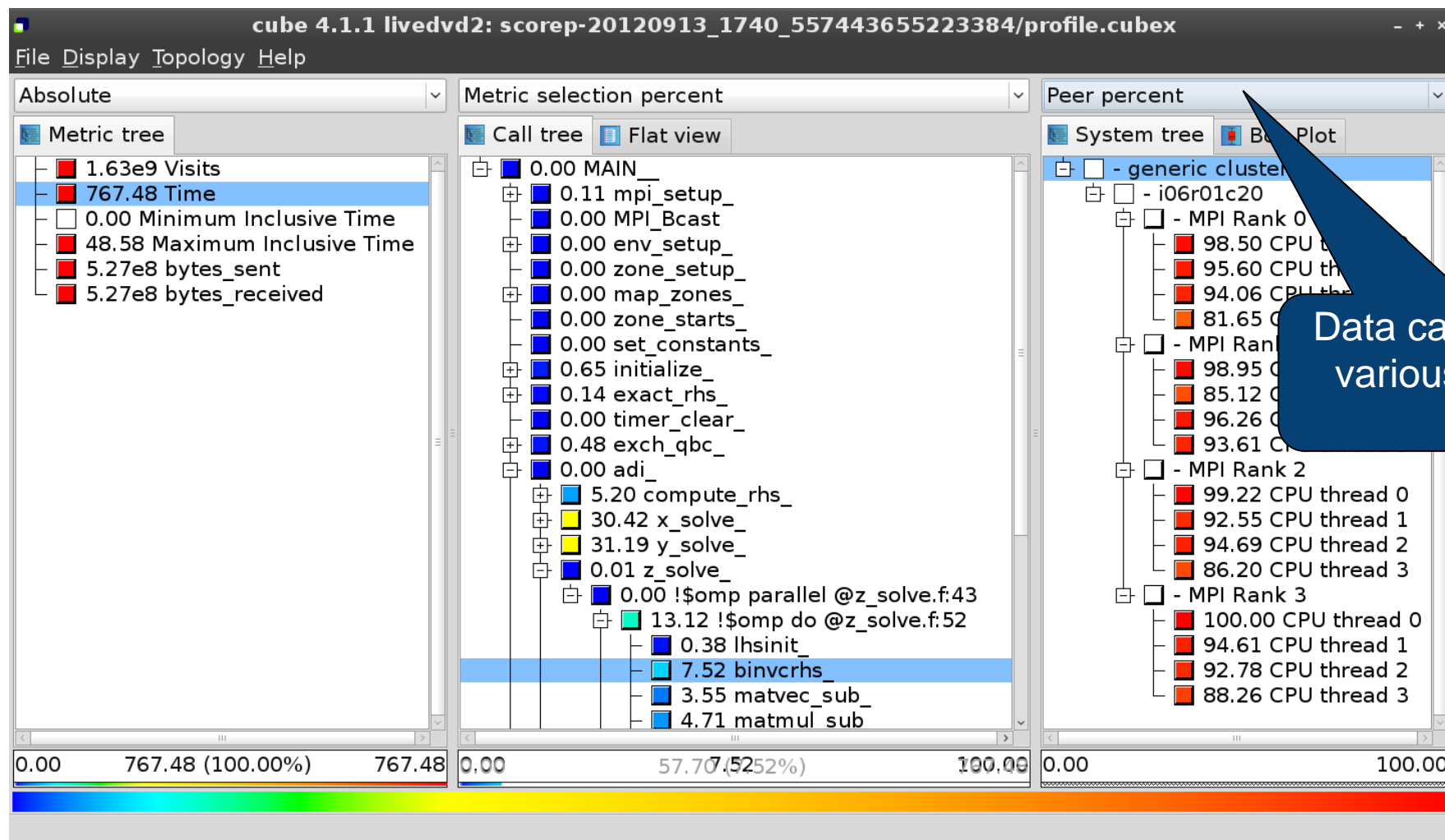
Flat profile view



Box plot view



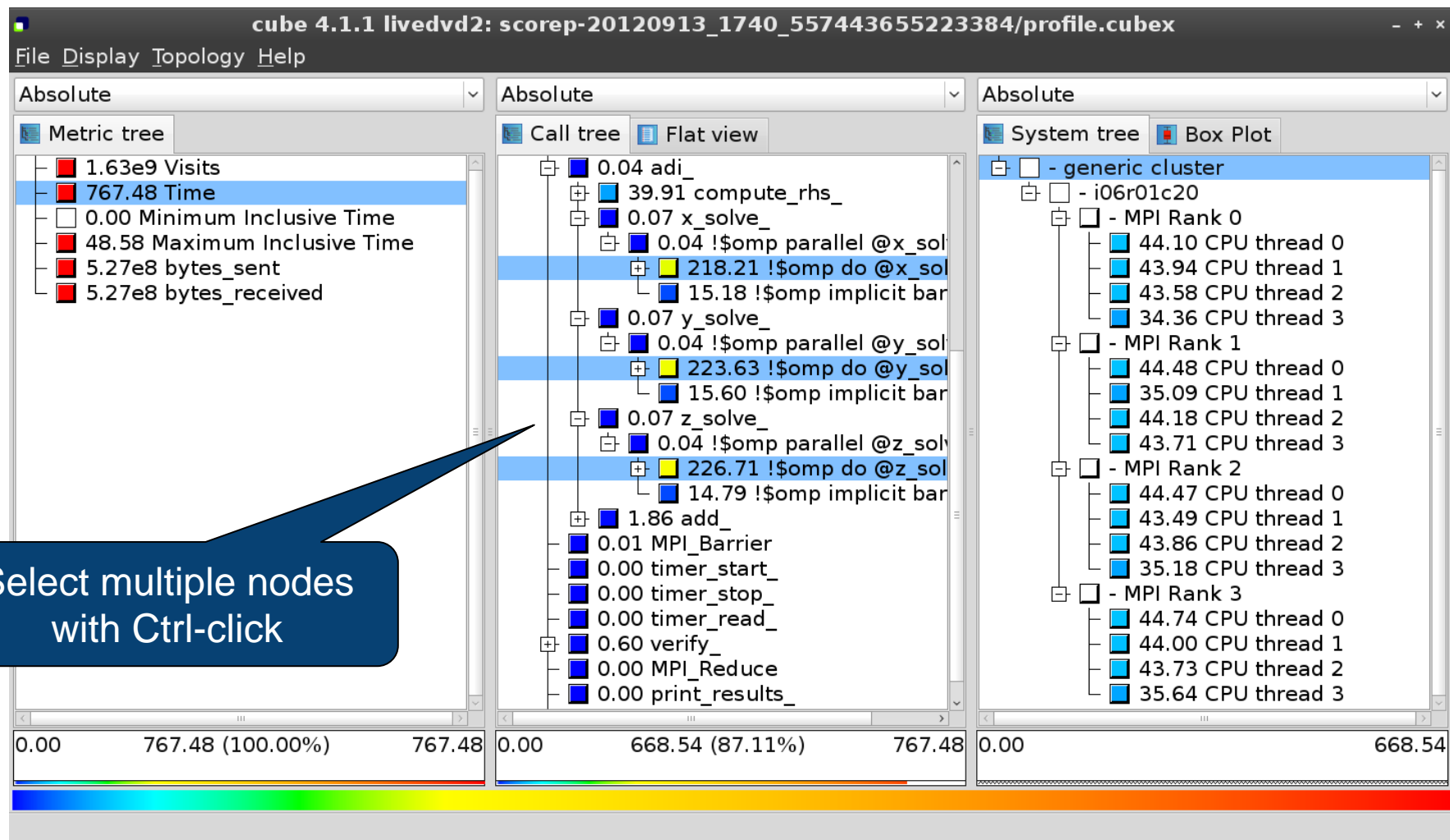
Alternative display modes



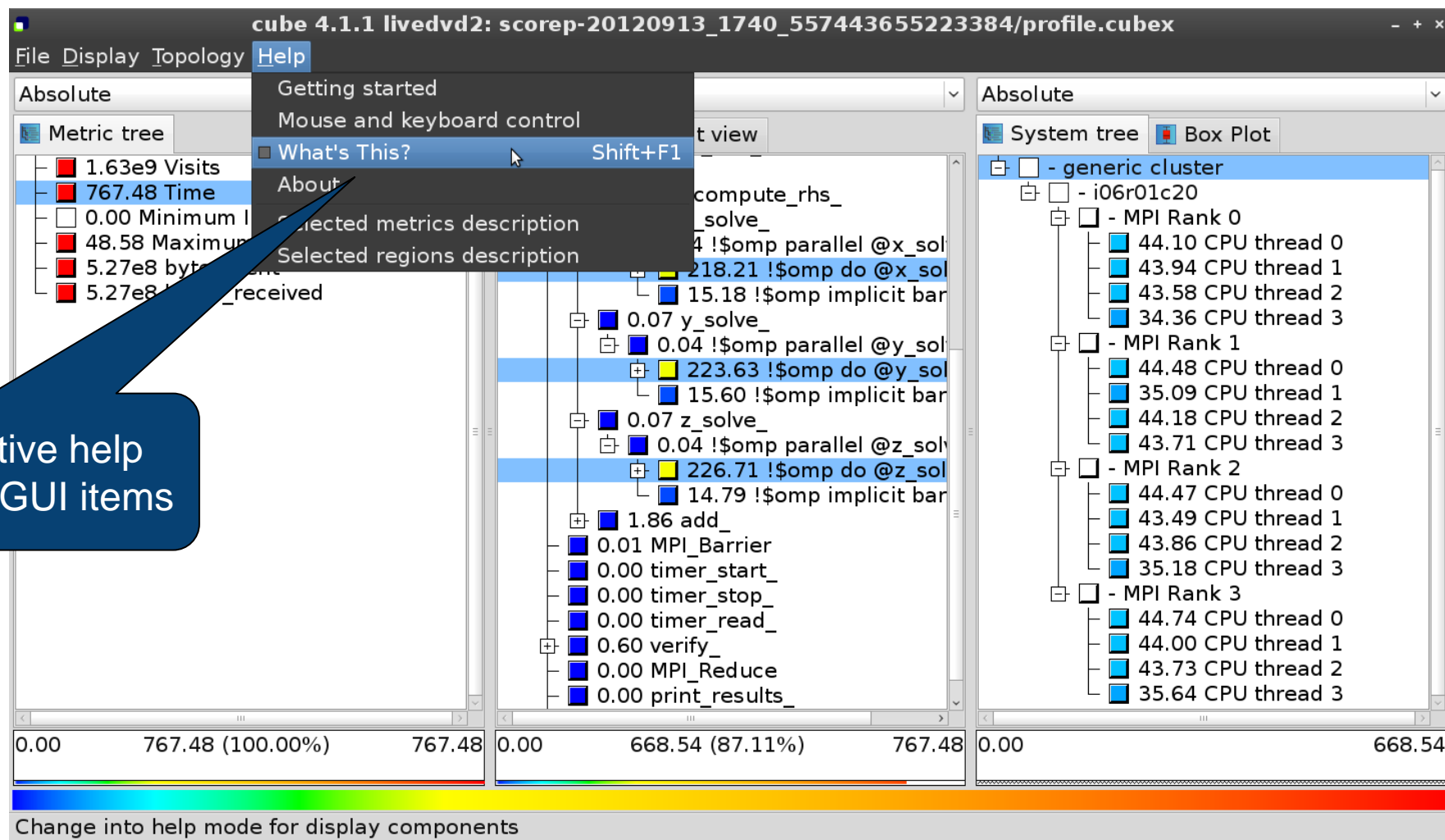
Important display modes

- 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



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:

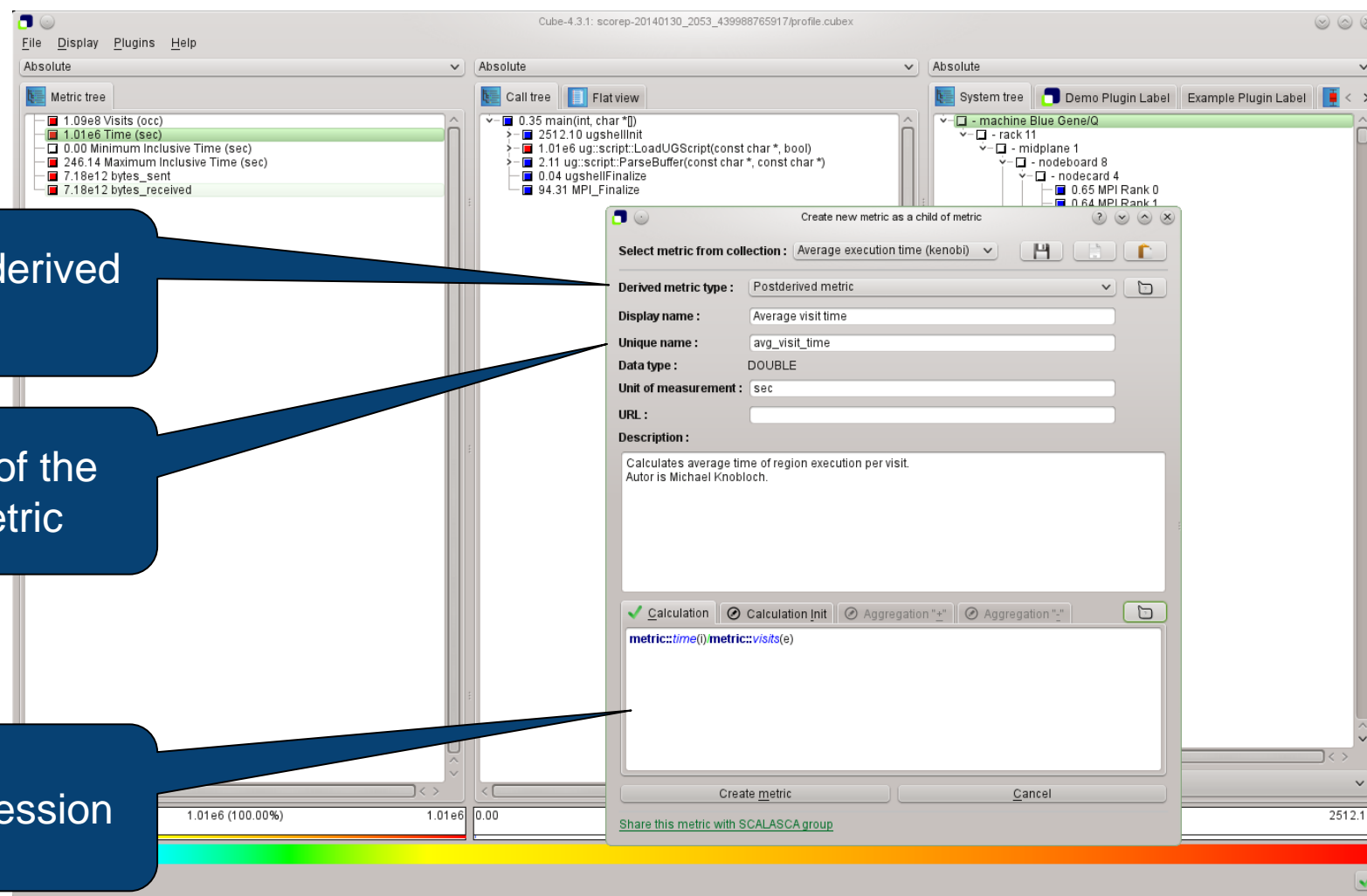
- “Average execution time”: Postderived metric with expression

`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

Example: FLOPS based on PAPI_FP_OPS and time

The screenshot displays the Cube-4.3.1 software interface, which is used for analyzing performance metrics. It is divided into several panes:

- Left Pane (Edit metric FLOPS):** This pane allows for configuring a derived metric. The 'Select metric from collection' dropdown is set to '--- please select ---'. The 'Derived metric type' is 'Postderived metric'. The 'Display name' is 'FLOPS', the 'Unique name' is 'flops', and the 'Data type' is 'DOUBLE'. The 'Unit of measurement' is empty. The 'URL' is empty. The 'Description' field is empty. At the bottom, there is a 'Calculation' section with a formula: `metric::PAPI_FP_OPS()/metric::time()`. Buttons for 'Edit metric' and 'Cancel' are present.
- Metric tree (Absolute):** This pane shows a list of metrics. The 'FLOPS' metric is highlighted in blue. The list includes: 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), and 1.84e9 FLOPS.
- Call tree (Absolute):** This pane shows a hierarchical view of the metrics. The 'MAIN' metric is highlighted in blue. The tree includes: 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_, and 1.05e5 MPI_Reduce.
- System tree (Absolute):** This pane shows a hierarchical view of the system. The 'machine Linux' metric is highlighted in blue. The tree includes: - 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, and 9.02e8 OMP thread 3.

At the bottom of the interface, there is a status bar showing the selected metric: 'Selected !\$omp do @exact_rhs.f:46'.

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:

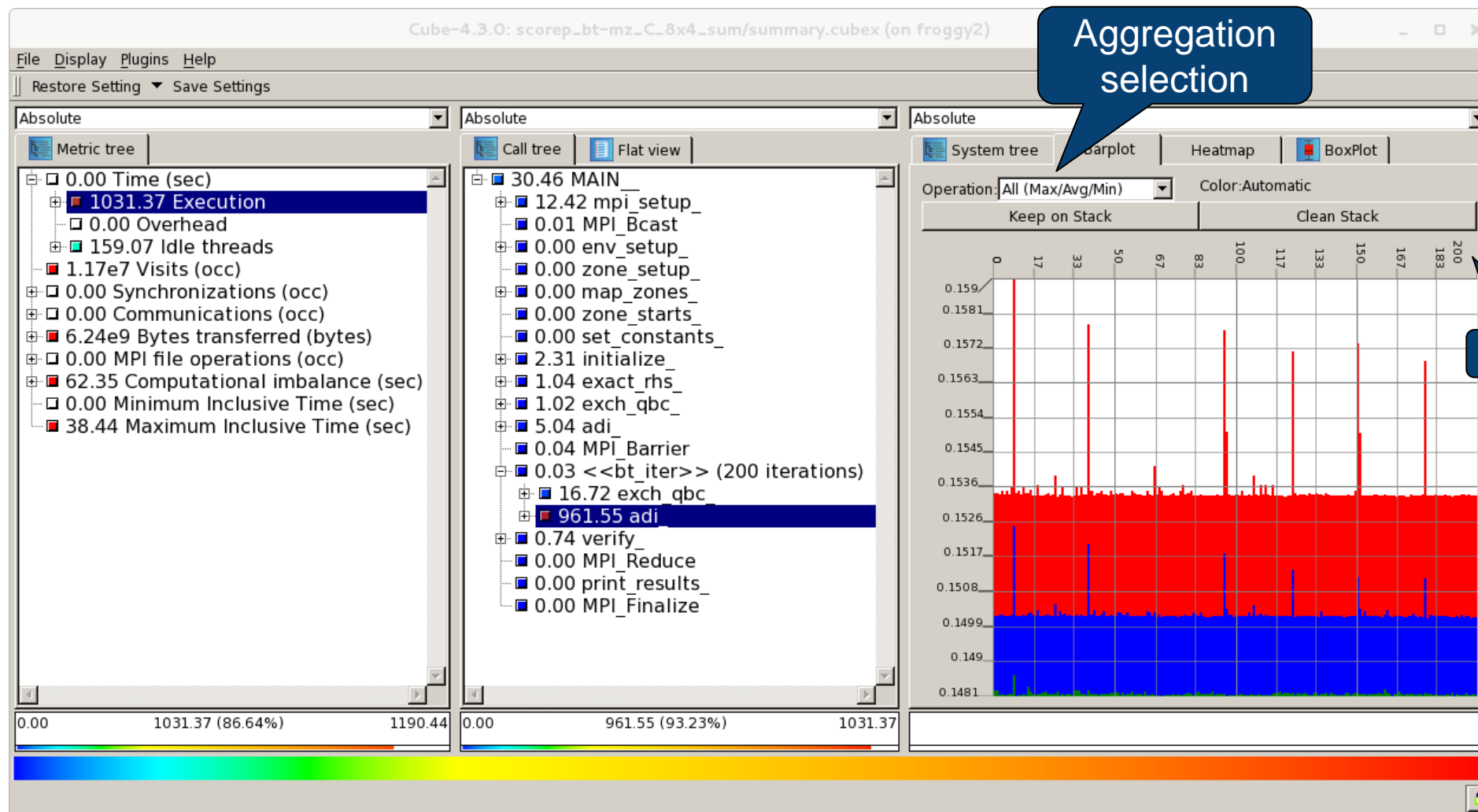
- Mark loop body by using Score-P instrumentation API in your source code

```
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 )
```

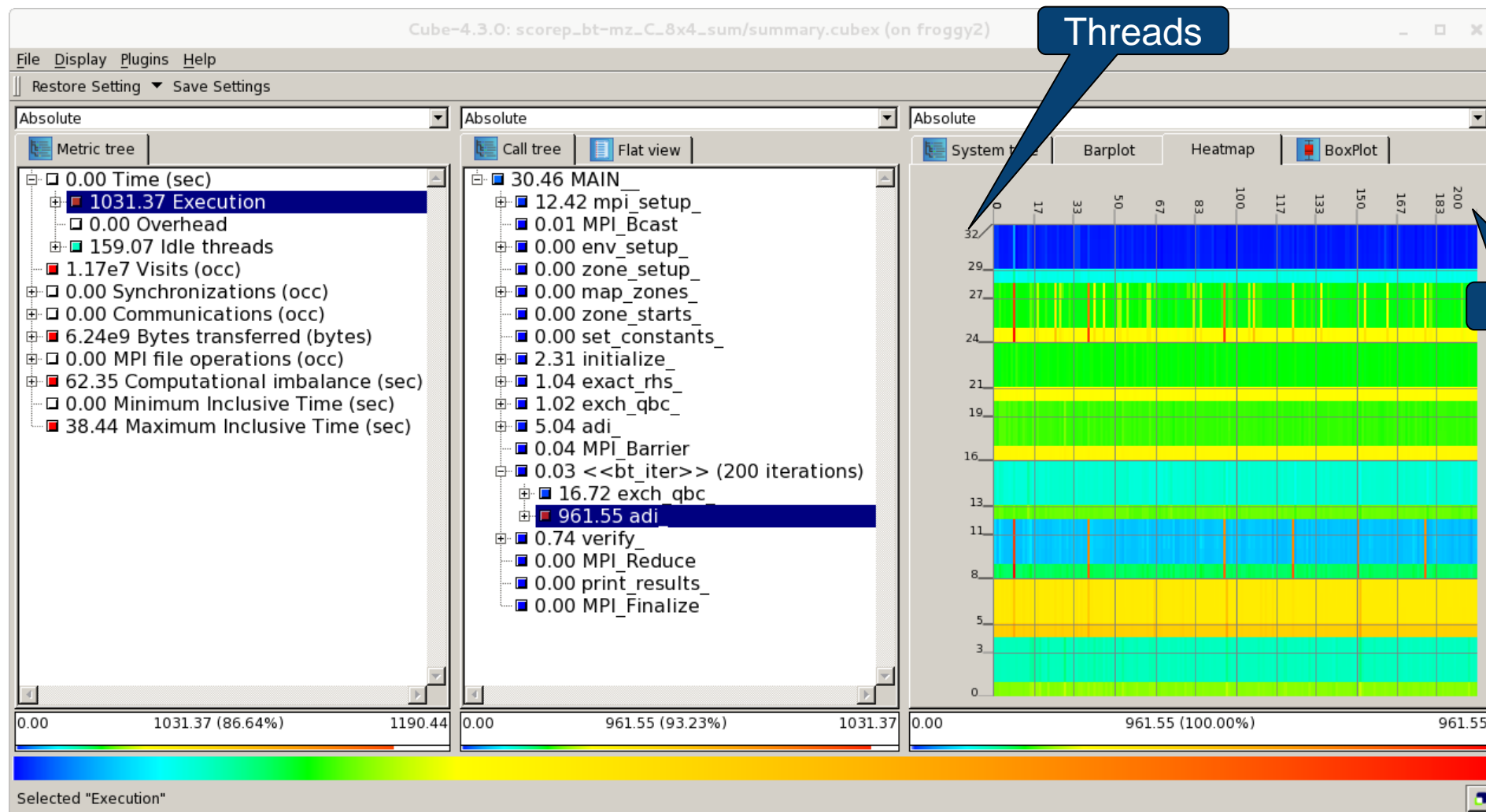
- Result in the Cube profile:

- 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



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:
 - ``cube-config --cube-dir`/share/doc/CubeGuide.pdf`
- Contact:
 - mailto: scalasca@fz-juelich.de

