

Analysis report examination with Cube

Michael Knobloch
Jülich Supercomputing Centre

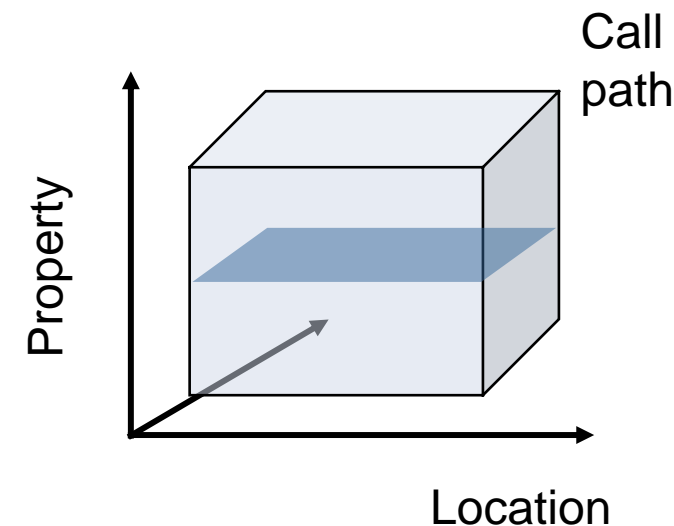


Cube

- 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 4.3.4 (April 2016)

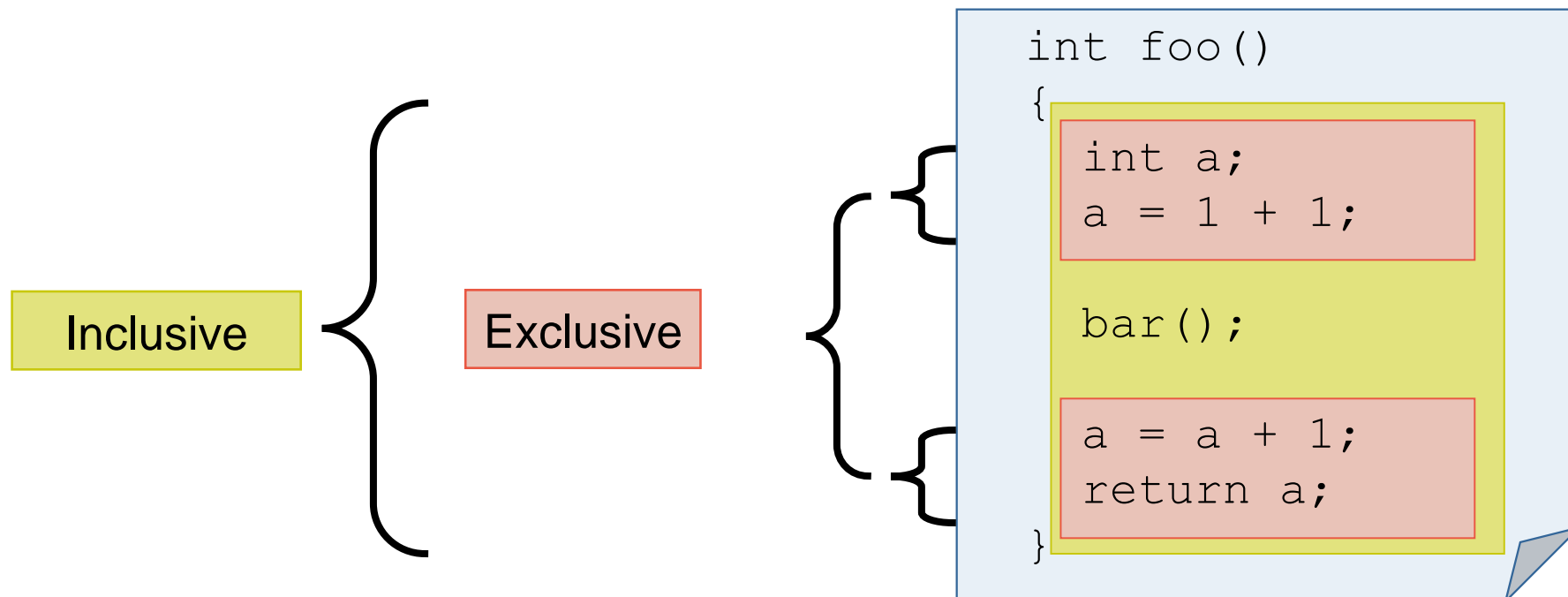
Analysis presentation and exploration

- 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

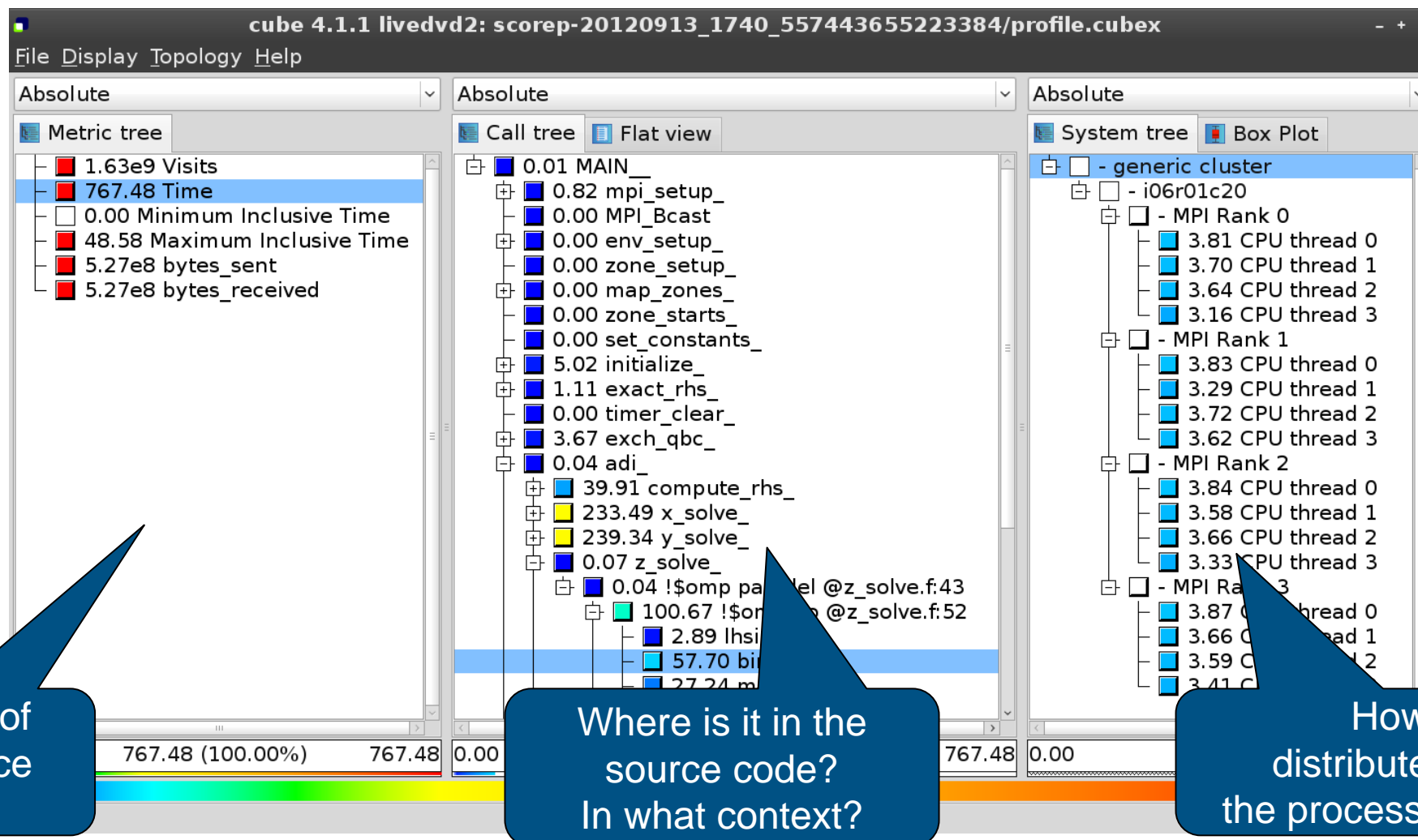


Inclusive vs. exclusive values

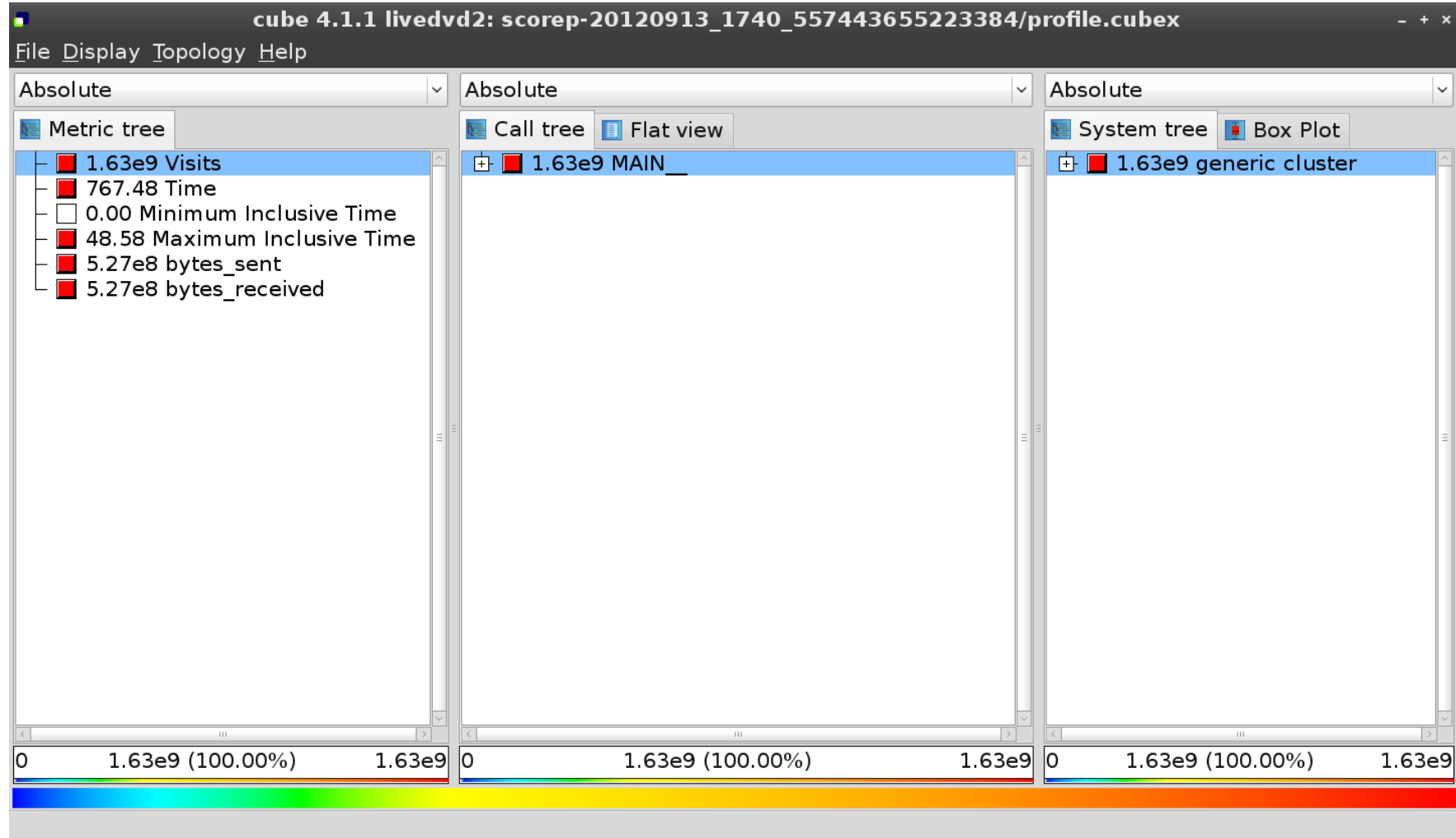
- Inclusive
 - Information of all sub-elements aggregated into single value
- Exclusive
 - Information cannot be subdivided further



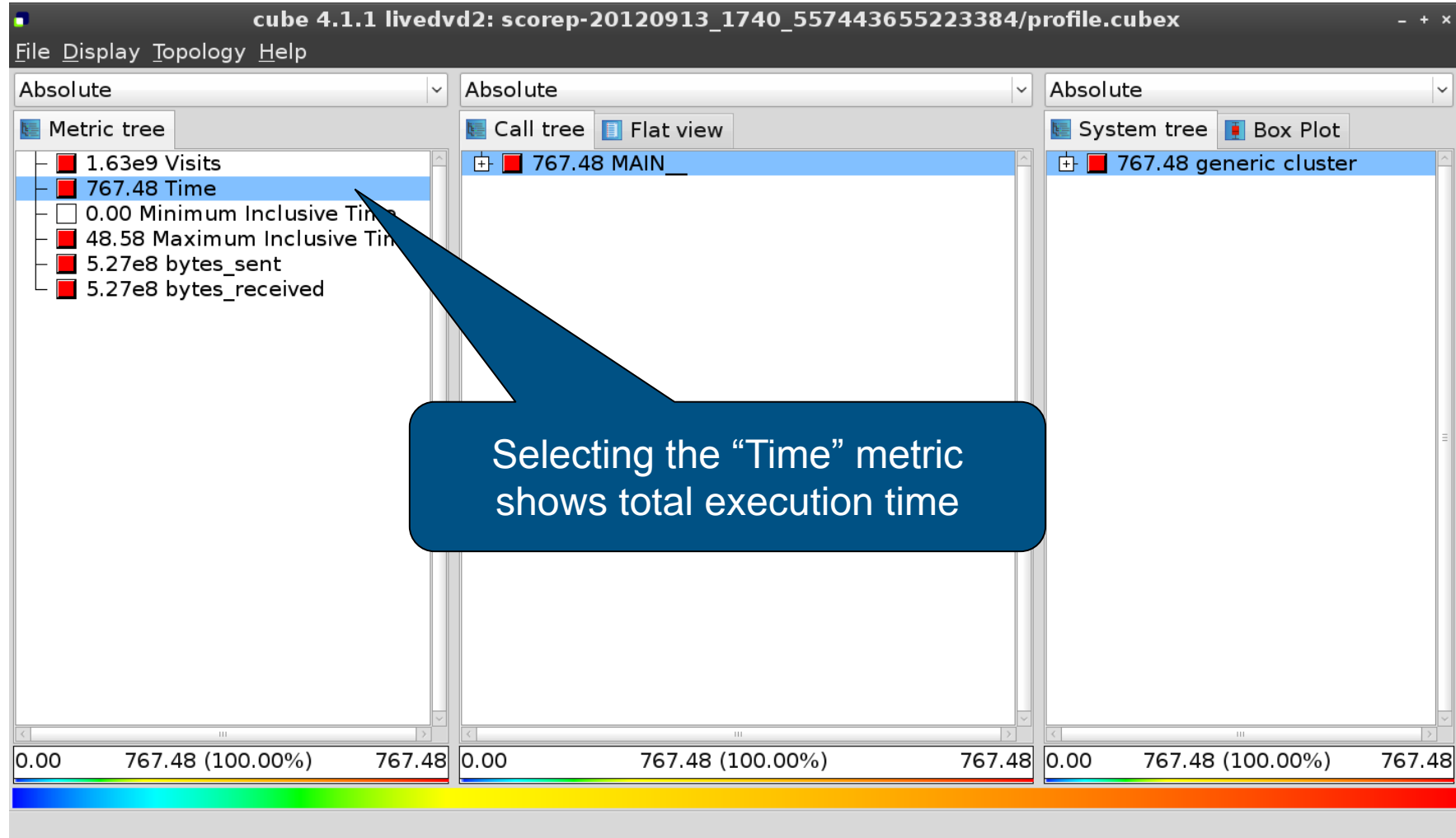
Analysis presentation



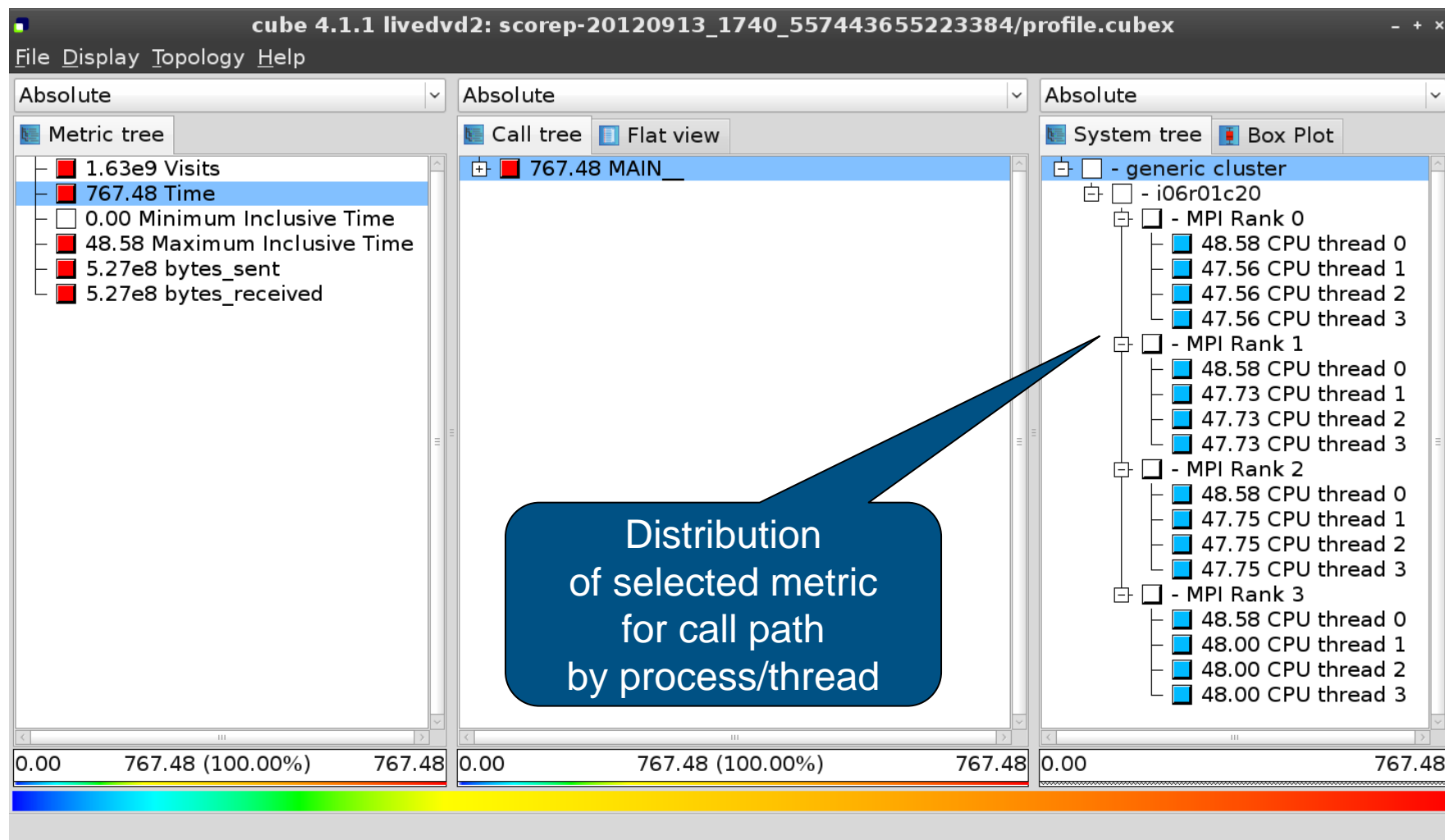
Score-P analysis report exploration (opening view)



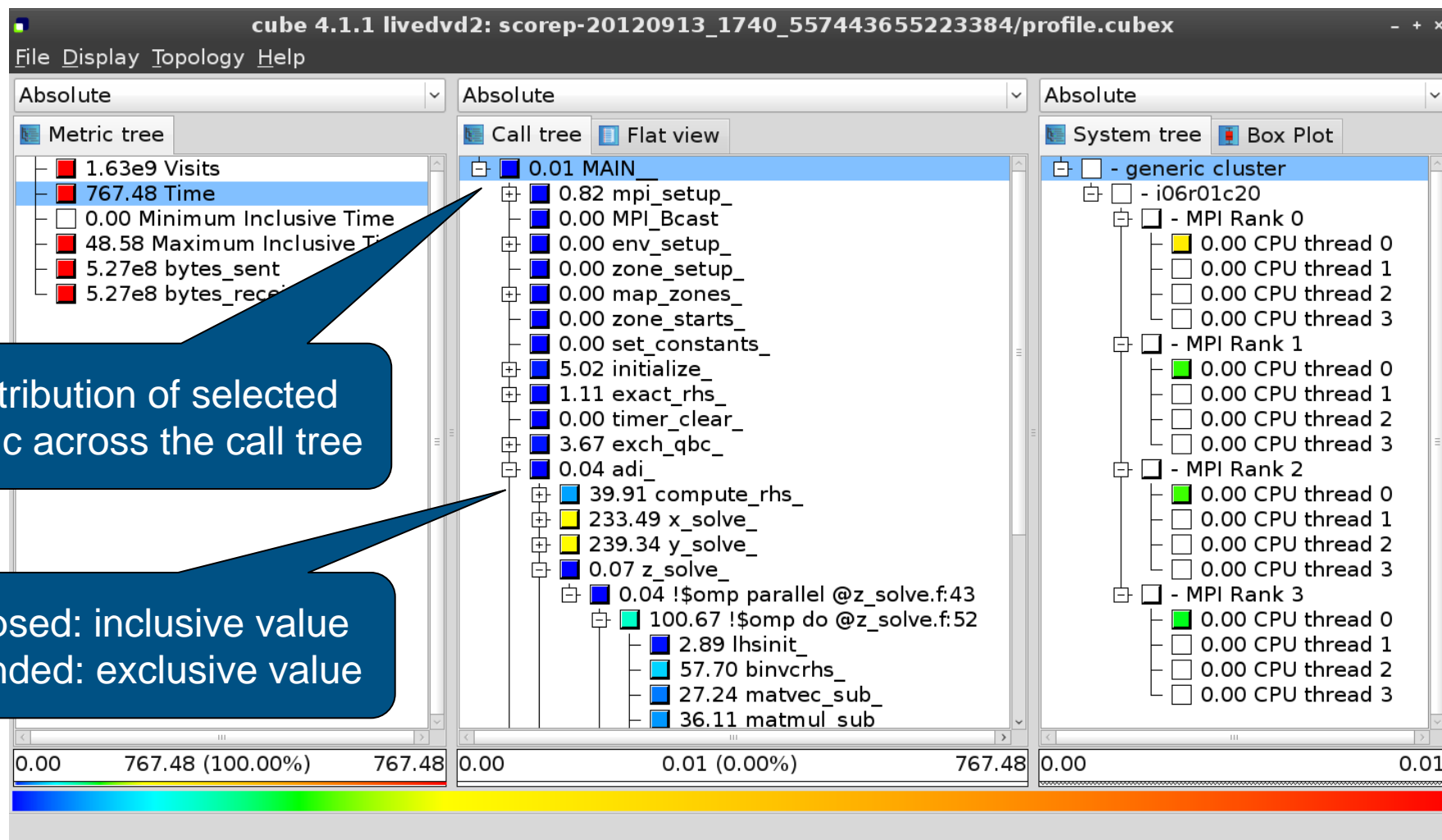
Metric selection



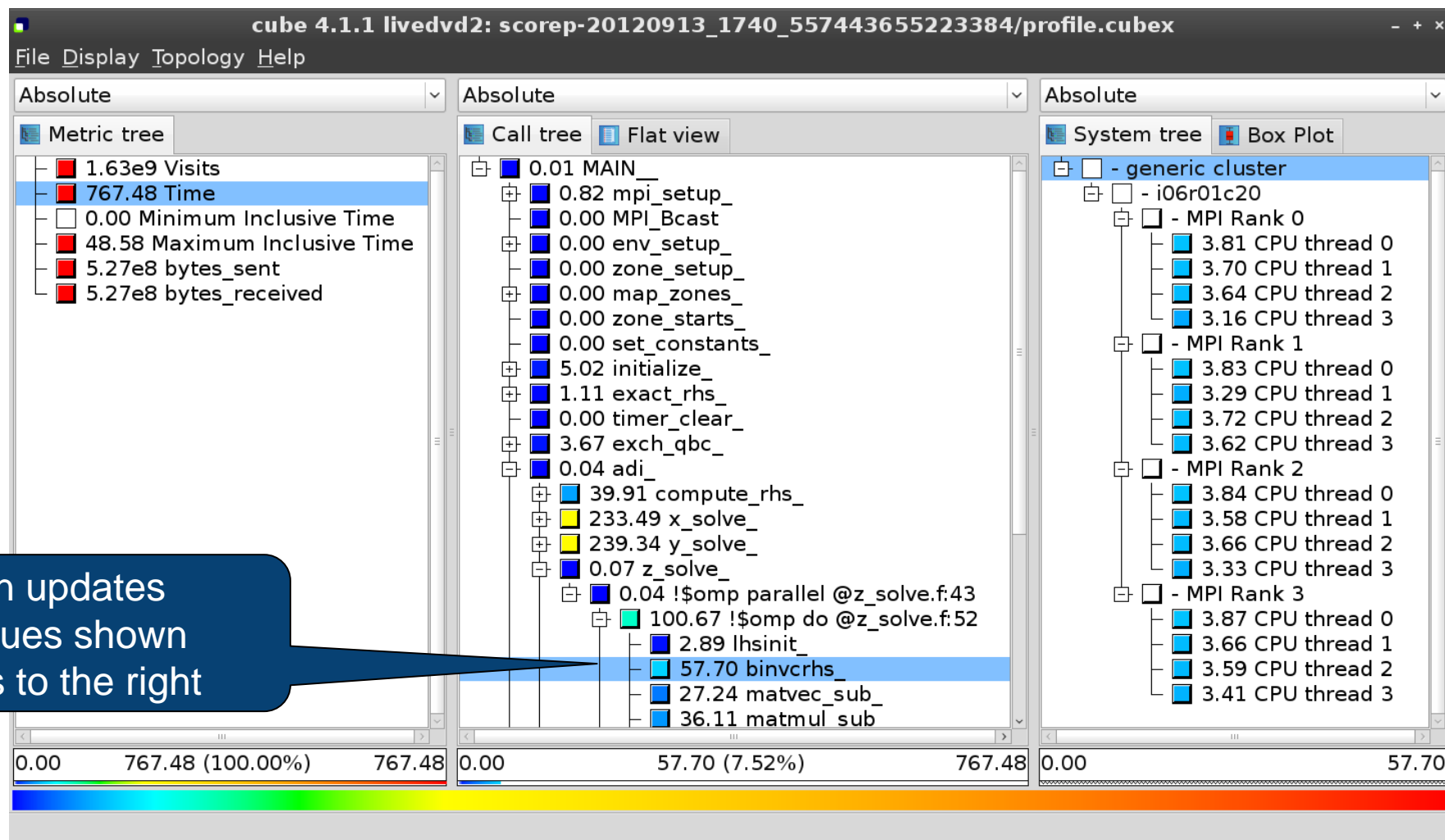
Expanding the system tree



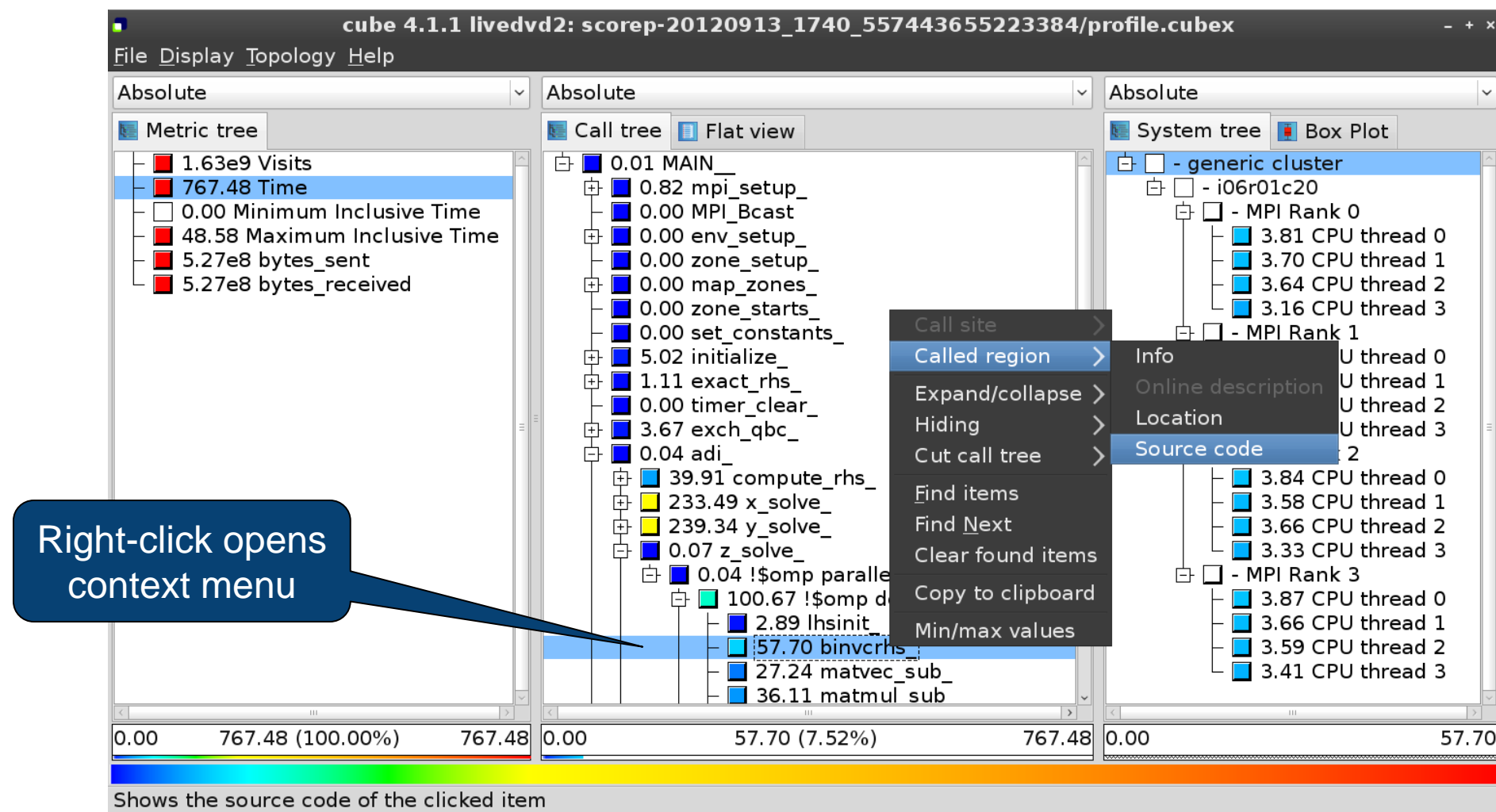
Expanding the call tree



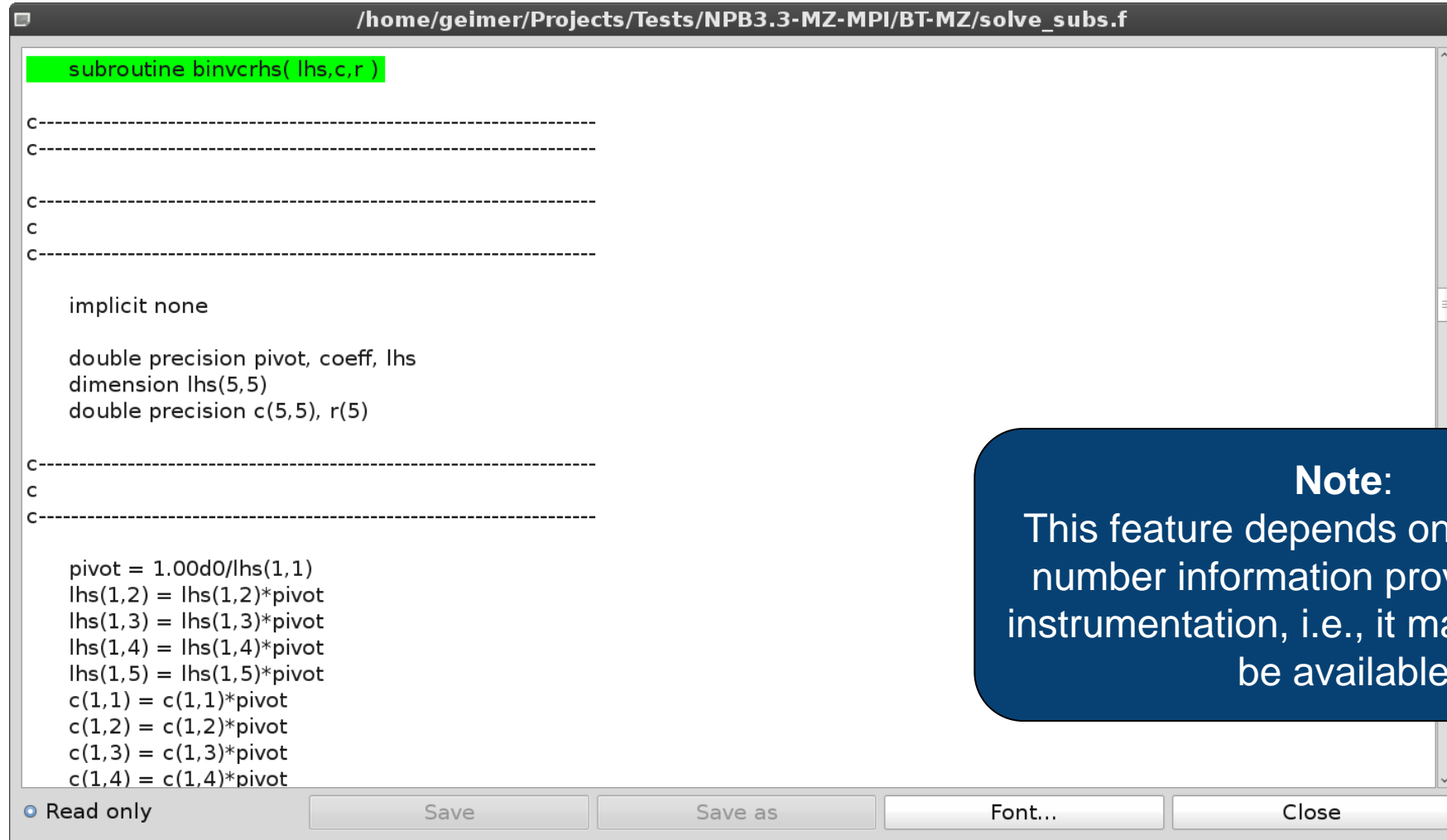
Selecting a call path



Source-code view via context menu



Source-code view



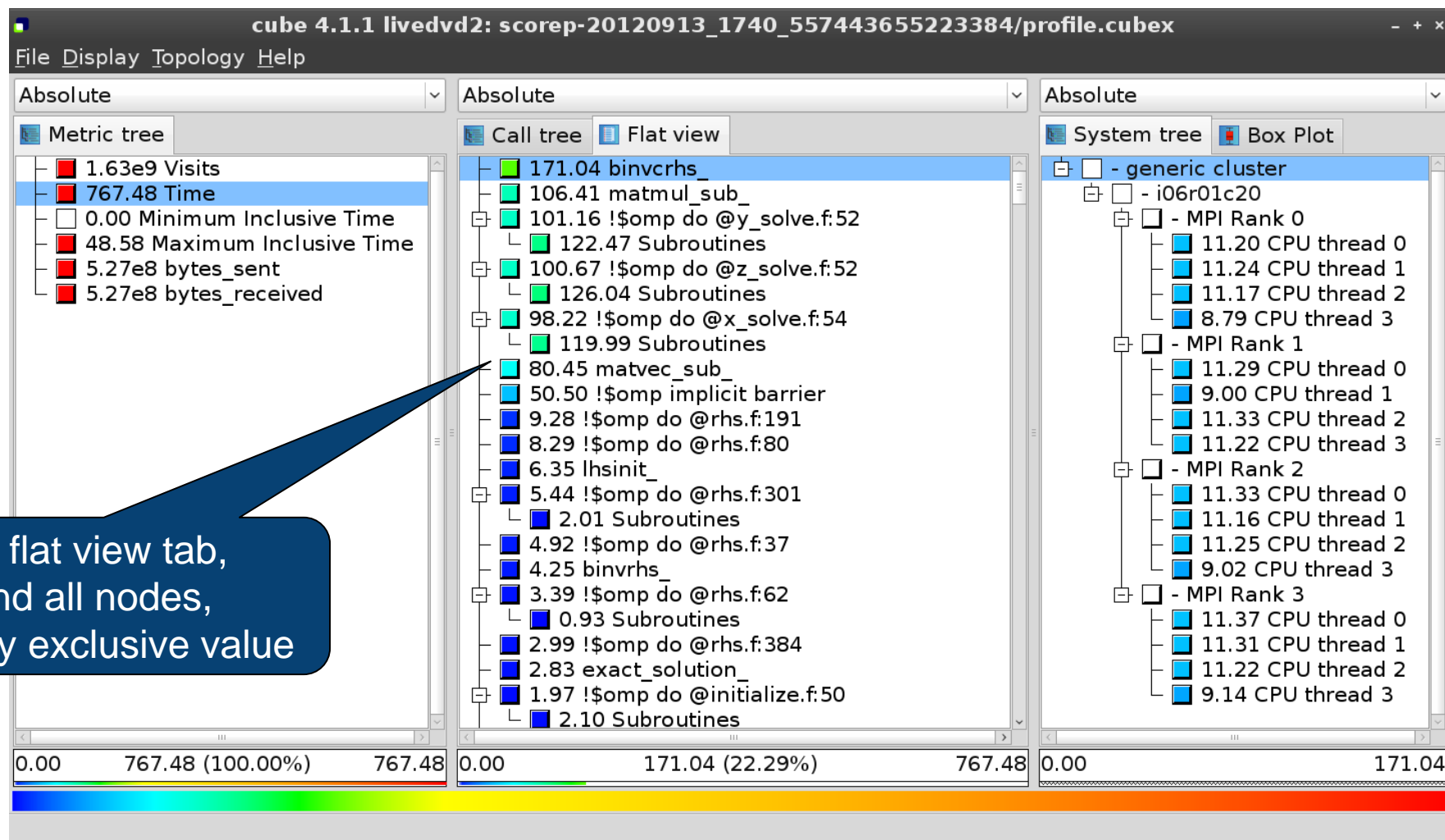
```
subroutine binvrhs( lhs,c,r )  
  
C-----  
C-----  
  
C-----  
C  
C-----  
  
implicit none  
  
double precision pivot, coeff, lhs  
dimension lhs(5,5)  
double precision c(5,5), r(5)  
  
C-----  
C  
C-----  
  
pivot = 1.00d0/lhs(1,1)  
lhs(1,2) = lhs(1,2)*pivot  
lhs(1,3) = lhs(1,3)*pivot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

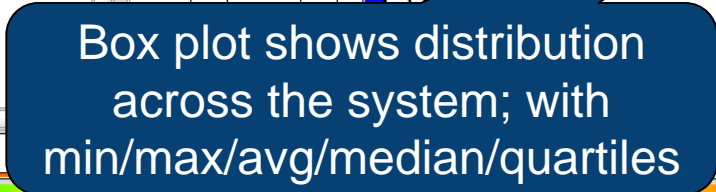
☒ Read only Save Save as Font... Close

Note:

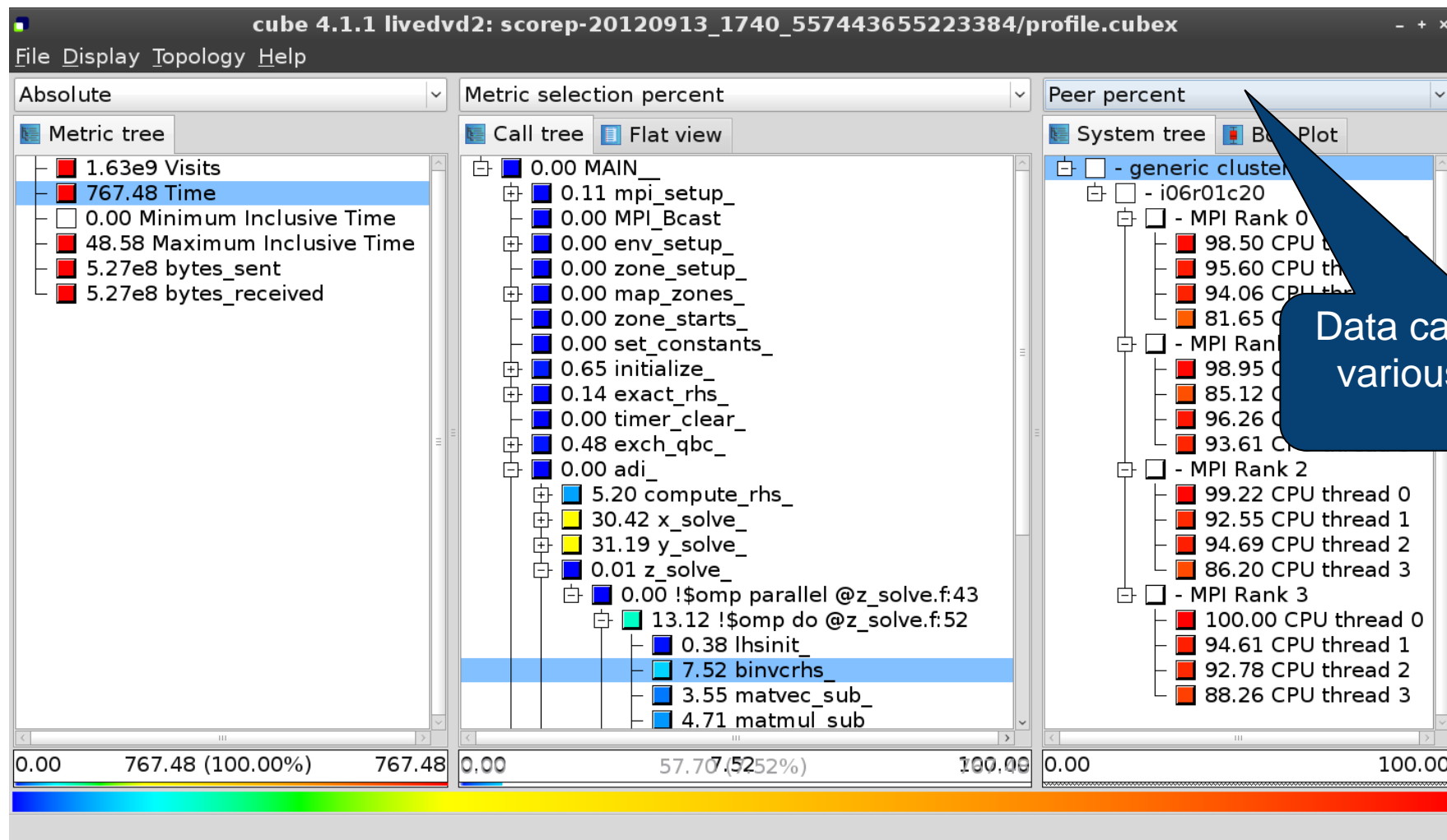
This feature depends on file and line number information provided by the instrumentation, i.e., it may not always be available

Flat profile view



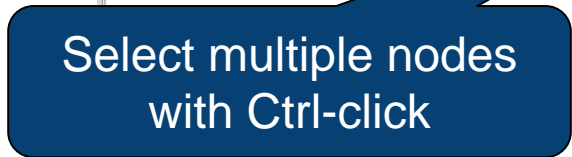


Alternative display modes



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



Context-sensitive help

The screenshot displays the 'cube 4.1.1' application window. The 'Help' menu is open, showing options: 'Getting started', 'Mouse and keyboard control', 'What's This?' (highlighted with a mouse cursor), and 'About'. The 'What's This?' option has a 'Shift+F1' shortcut. A blue callout bubble points to the 'What's This?' menu item with the text: 'Context-sensitive help available for all GUI items'.

The application window shows three main panels:

- Metric tree (Left):** Displays a list of metrics. The '767.48 Time' metric is selected and highlighted in blue. Below the list, a progress bar shows '0.00 767.48 (100.00%) 767.48'.
- System tree (Right):** Displays a hierarchical tree of system components. The 'generic cluster' is selected, and its sub-components are listed, including 'i06r01c20', 'MPI Rank 0', 'MPI Rank 1', 'MPI Rank 2', and 'MPI Rank 3'. Each rank lists its CPU threads and their respective times. Below the tree, a progress bar shows '0.00 668.54 (87.11%) 767.48'.
- Bottom Panel:** A color-coded progress bar and a status bar that reads '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:

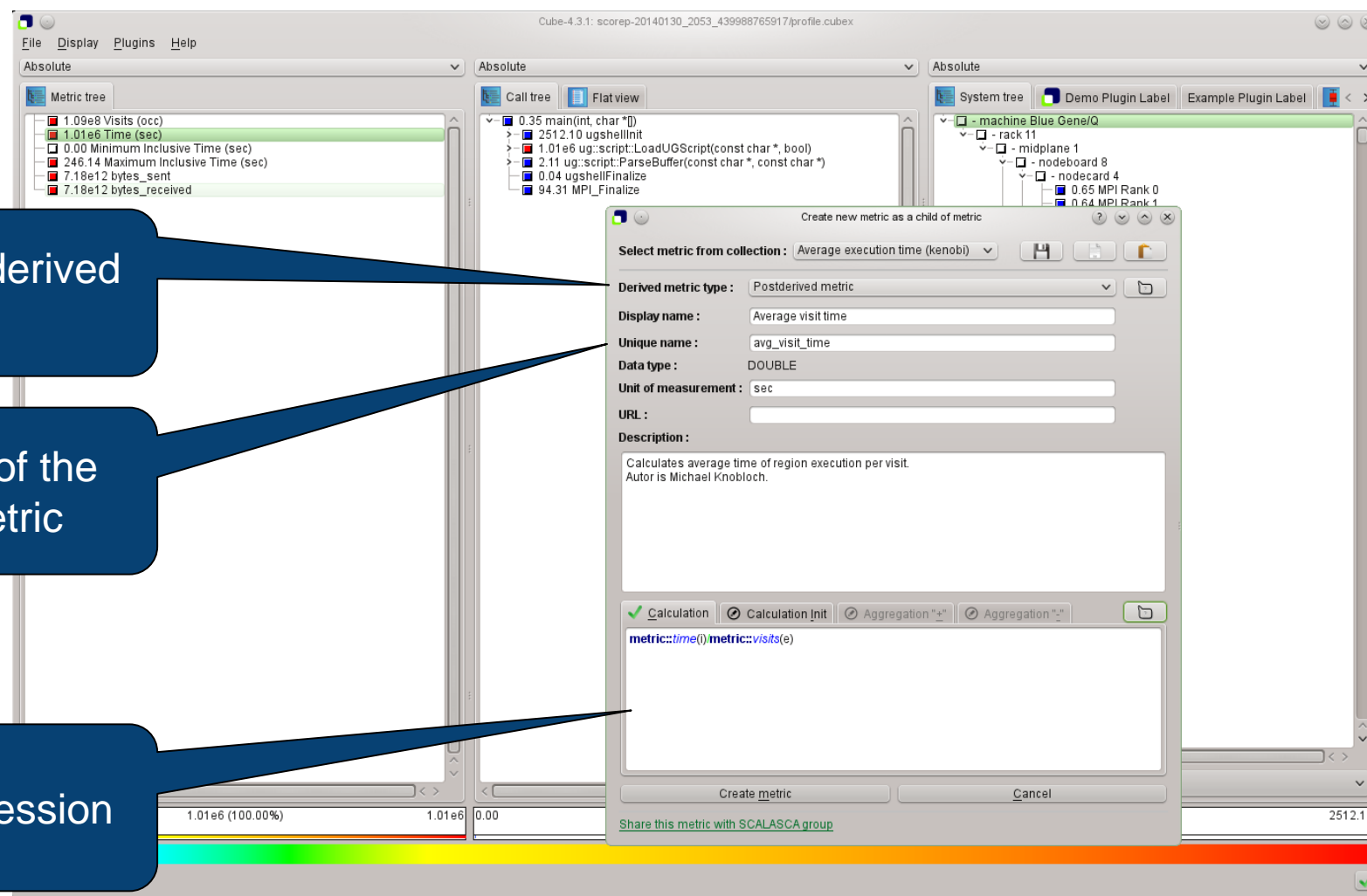
- “Average execution time”: Postderived metric with expression

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

- “Number of FLOP per second”: Postderived metric with expression

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

Derived metrics in Cube GUI



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 three main panels:

- Left Panel (Edit metric FLOPS):** This panel allows for the configuration of a derived metric. It includes fields for "Select metric from collection", "Derived metric type" (set to "Postderived metric"), "Display name" (set to "FLOPS"), "Unique name" (set to "flops"), "Data type" (set to "DOUBLE"), "Unit of measurement", "URL", and a "Description" field. At the bottom, there is a "Calculation" section with a formula: `metric::PAPI_FP_OPS()/metric::time()`. Buttons for "Edit metric" and "Cancel" are also present.
- Middle Panel (Metric tree):** This panel shows a hierarchical tree of metrics. The "Absolute" view is selected. The tree includes metrics such as "1.17e7 Visits (occ)", "1148.49 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" (highlighted in blue).
- Right Panel (System tree):** This panel shows a hierarchical tree of system components. The "Absolute" view is selected. The tree includes components such as "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..." (highlighted in blue), "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".

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_B_mic15p30x4_sum/profile.cubex  
Writing cut.cubex... done.
```

- Calculating difference of two reports

```
% cube_diff scorep_bt-mz_B_mic15p30x4_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

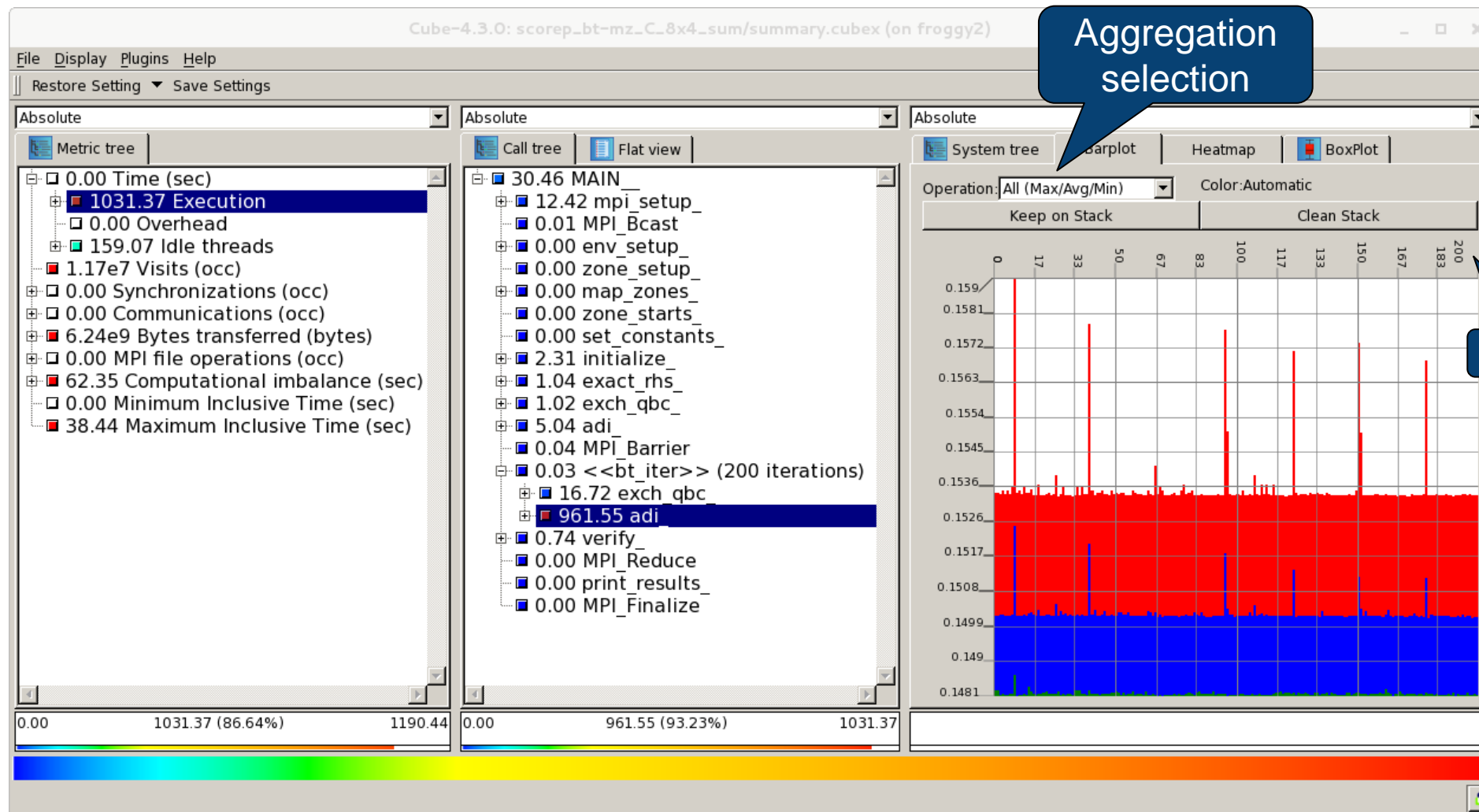
Iteration profiling

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

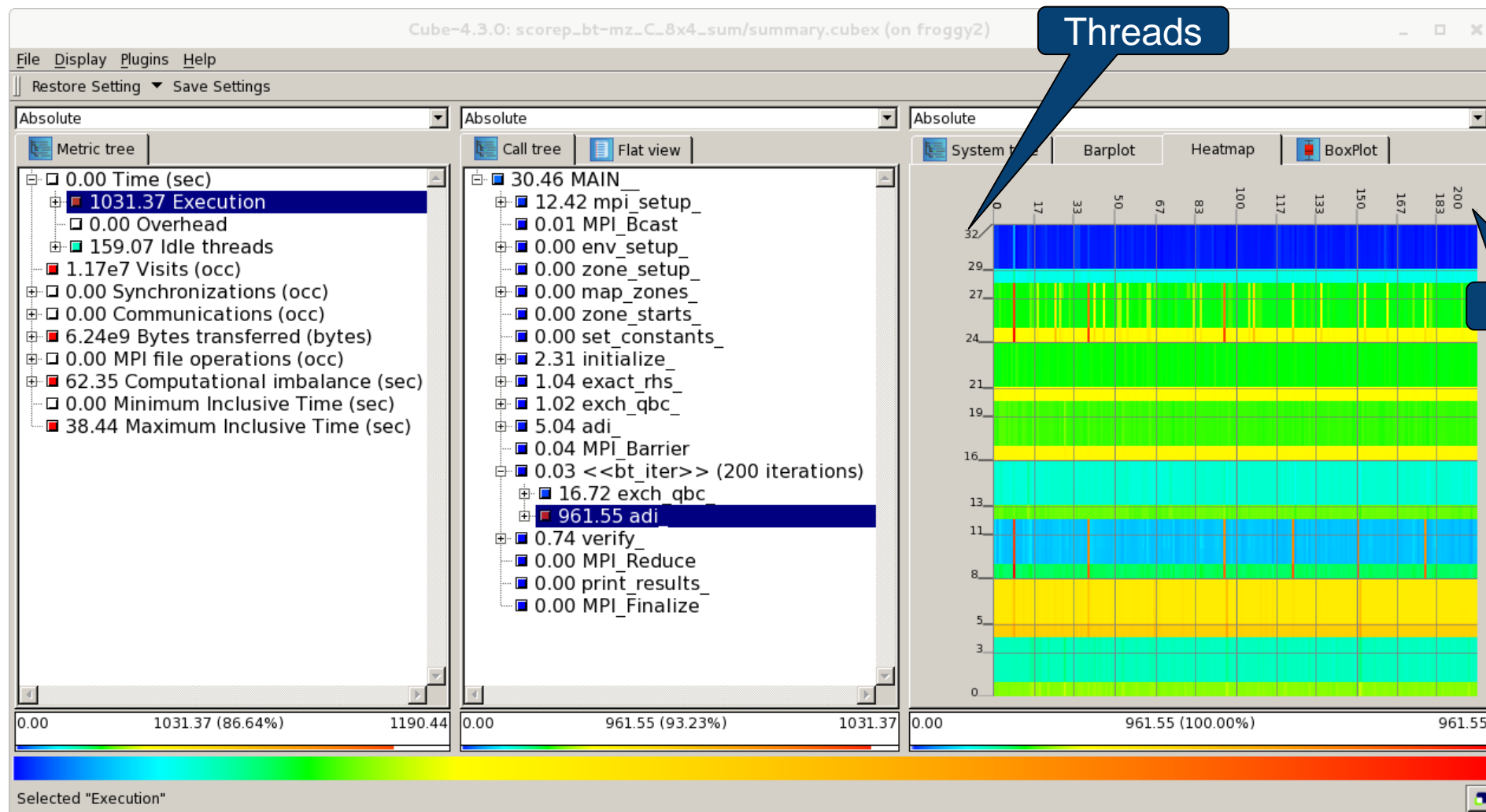
```
SCOREP_USER_REGION_DEFINE( scorep_bt_loop )  
SCOREP_USER_REGION_BEGIN( scorep_bt_loop, "<<bt_iter>>", SCOREP_USER_REGION_TYPE_DYNAMIC )  
SCOREP_USER_REGION_END( scorep_bt_loop )
```

- 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 XML 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

