

# Analysis report examination with Cube

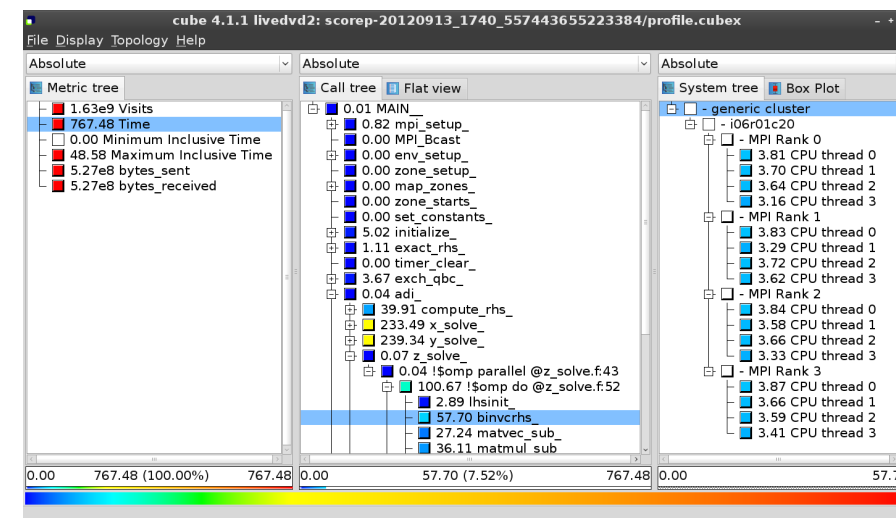
---

Marc Schlütter  
Jülich Supercomputing Centre



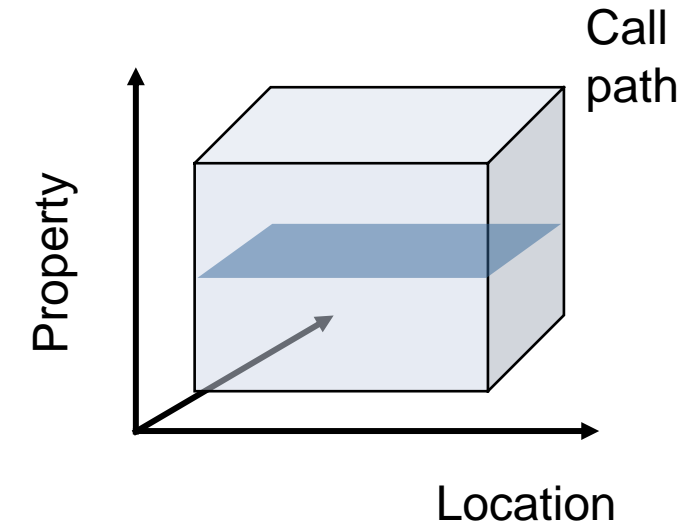
# Cube

- 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  $\geq 4.6$  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)

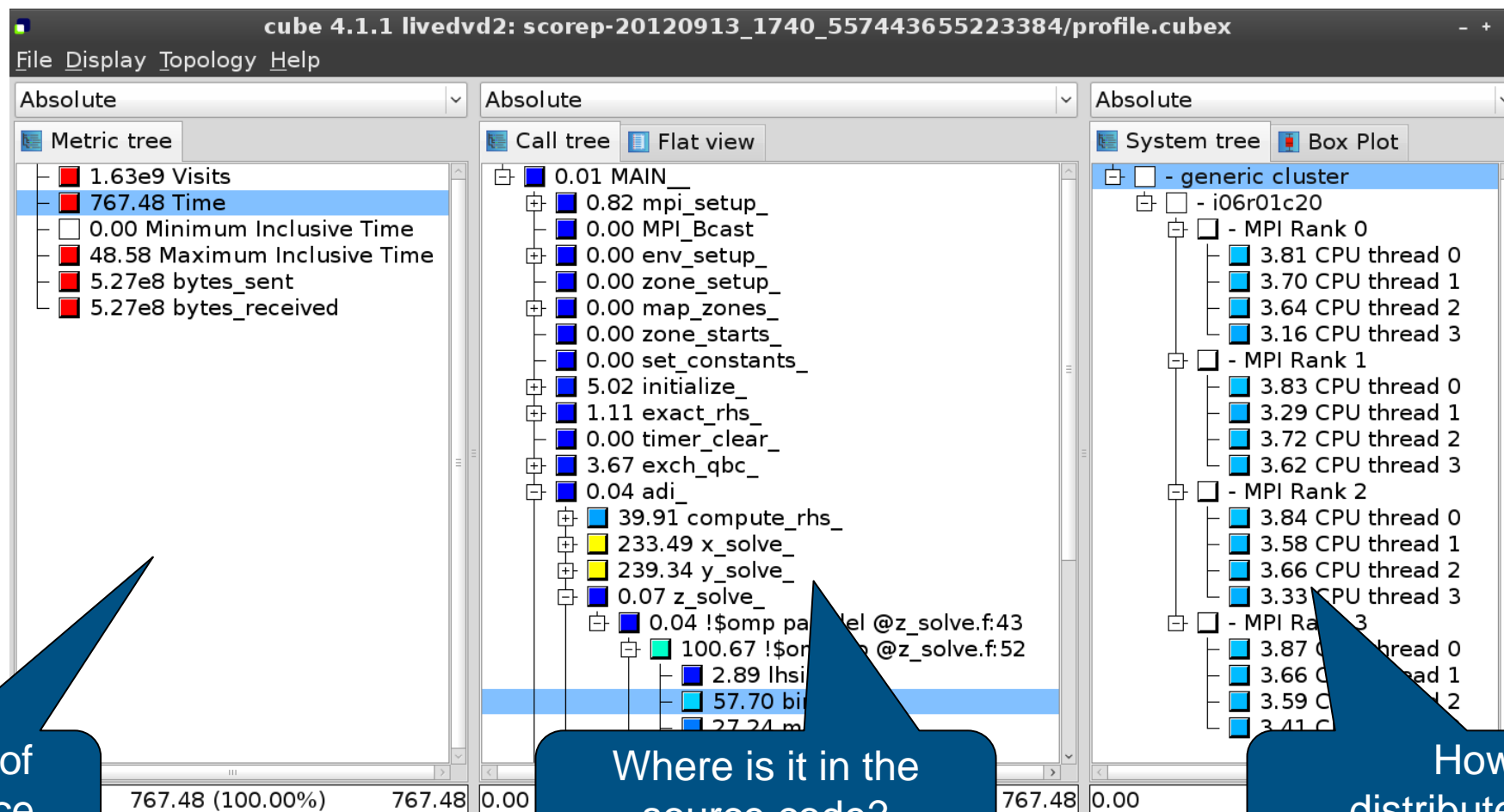


# Analysis presentation and exploration

- 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



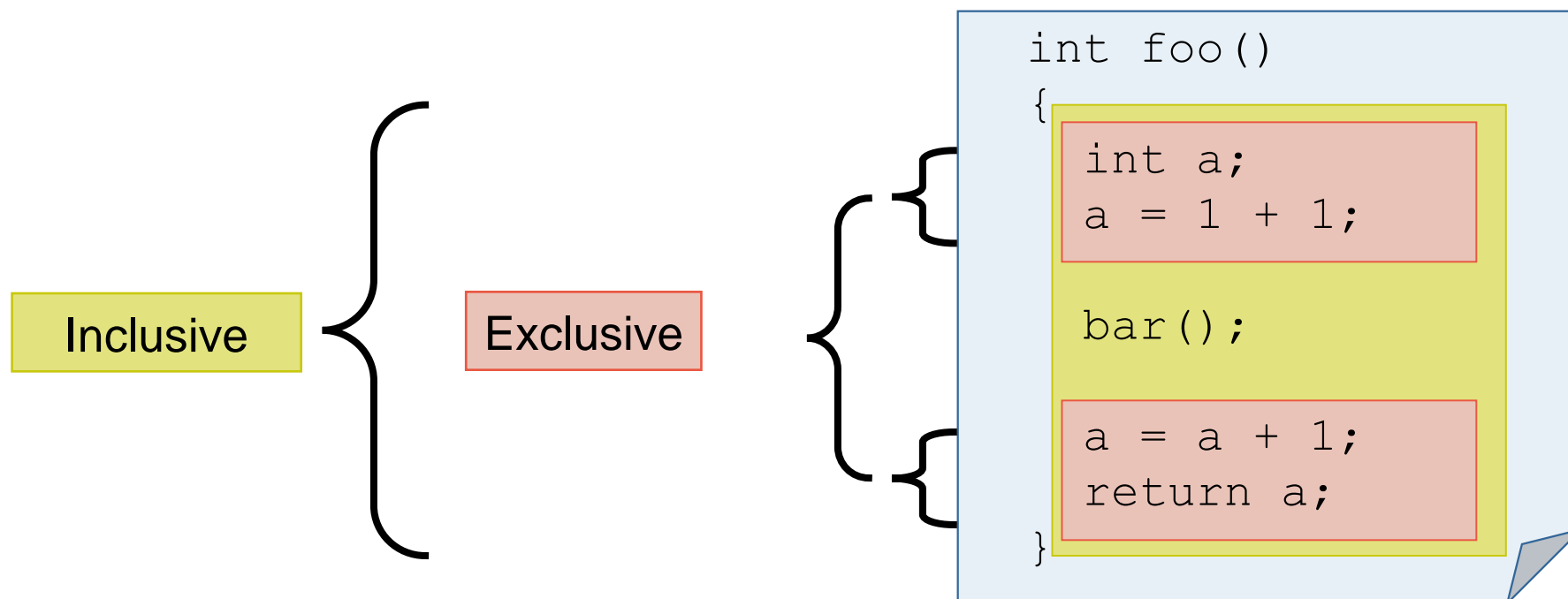
What kind of performance metric?

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

How is it distributed across the processes/threads?

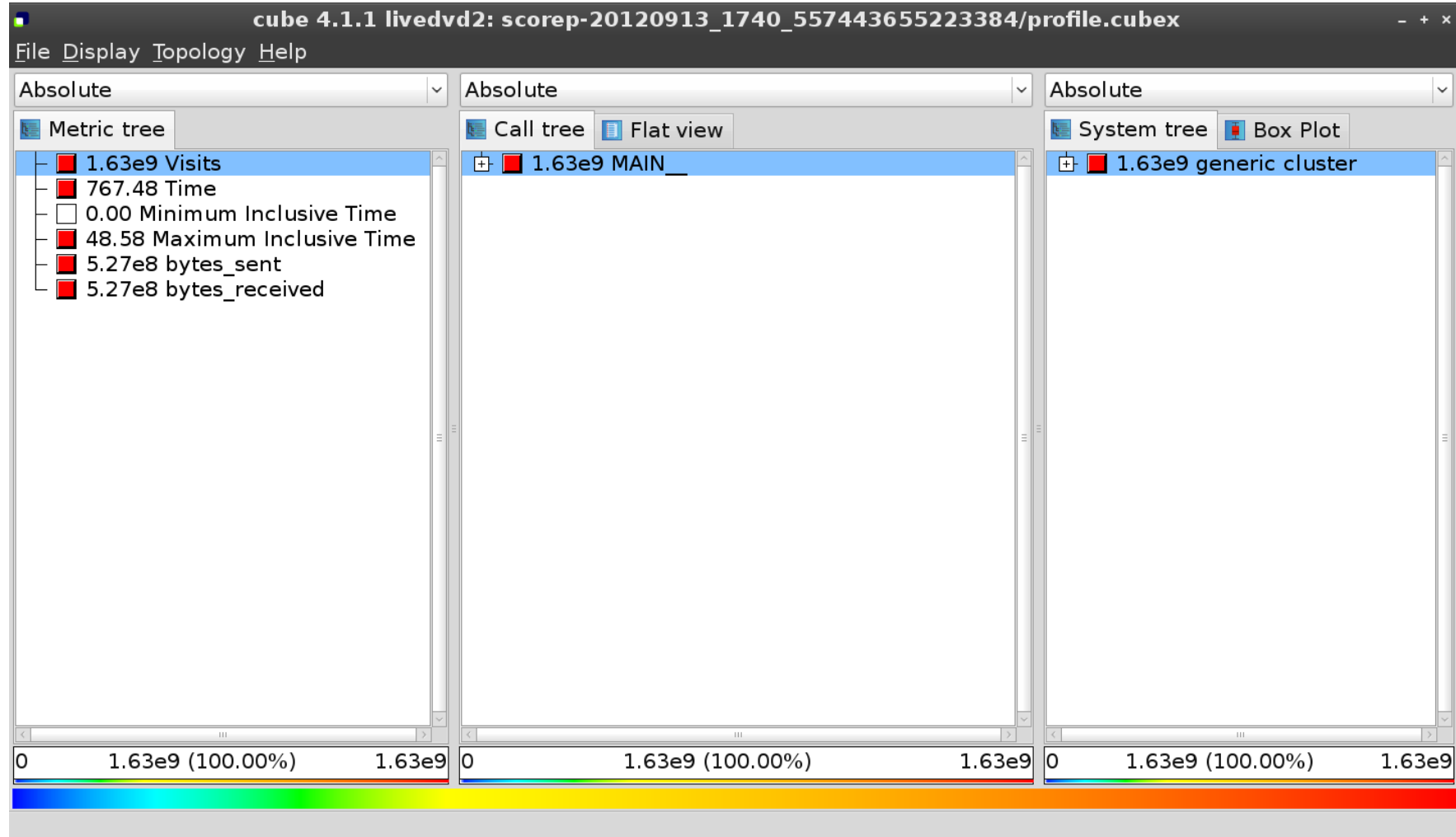
# Inclusive vs. exclusive values

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

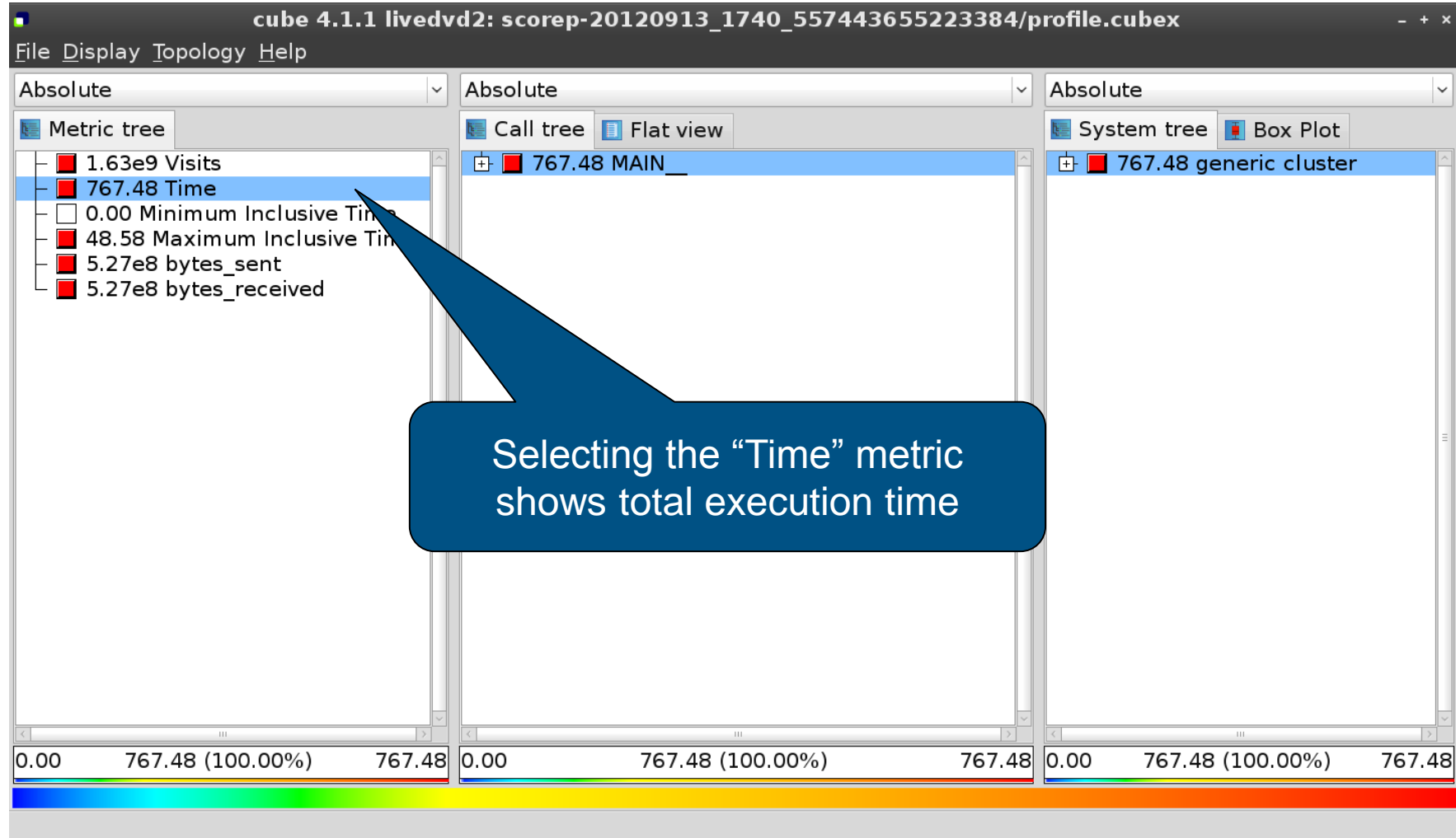




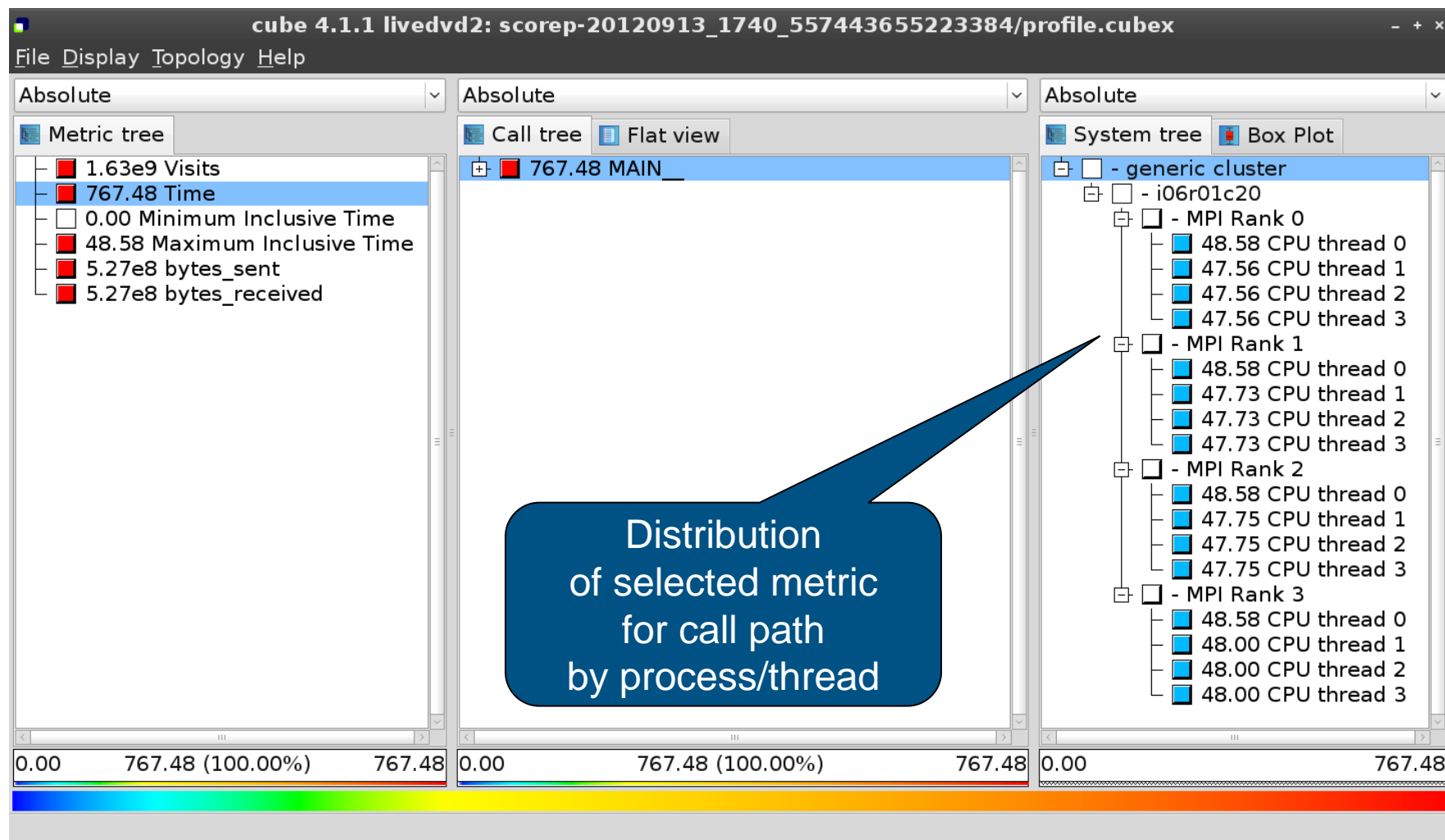
# Score-P analysis report exploration (opening view)



# Metric selection

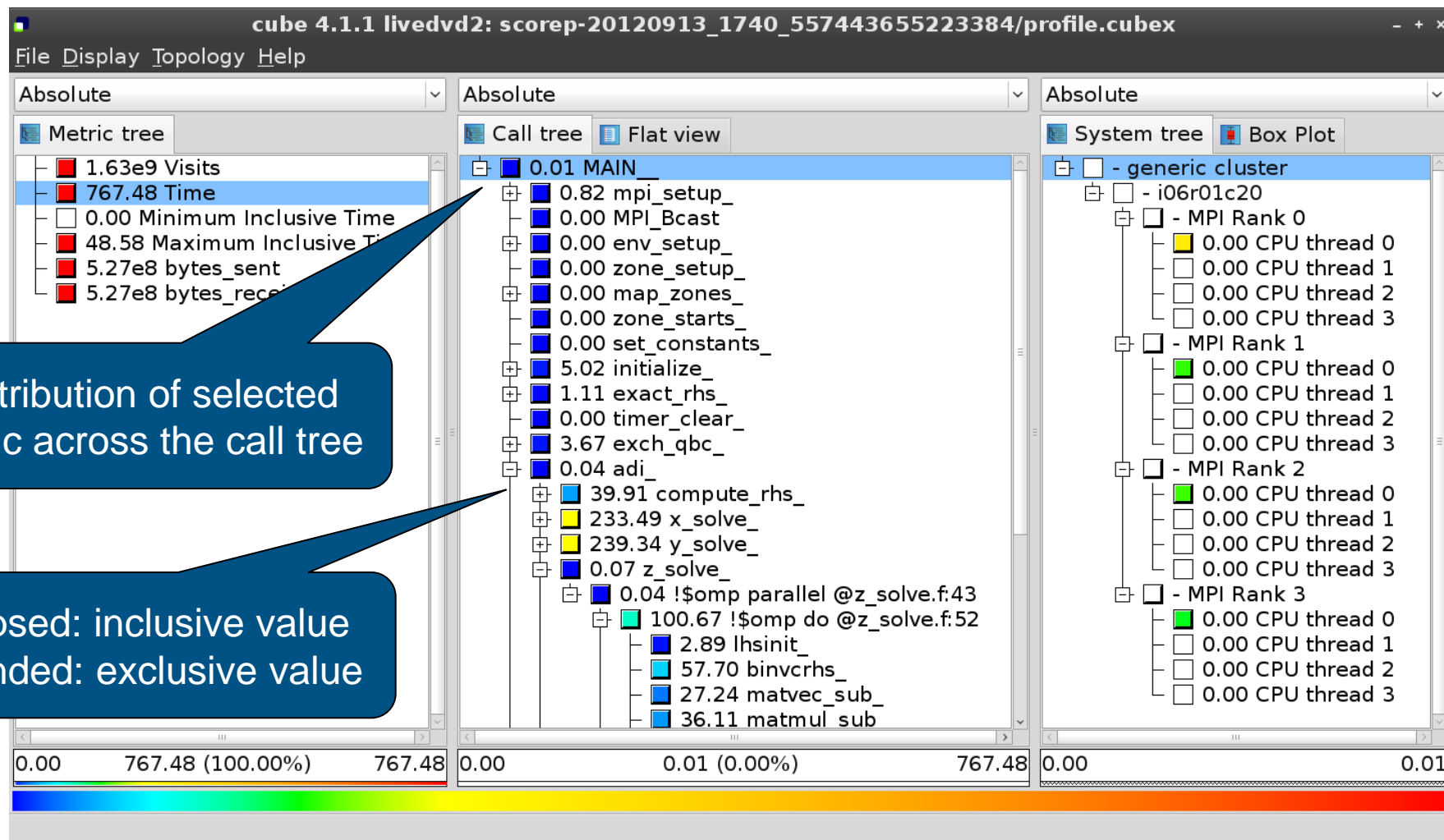


# Expanding the system tree

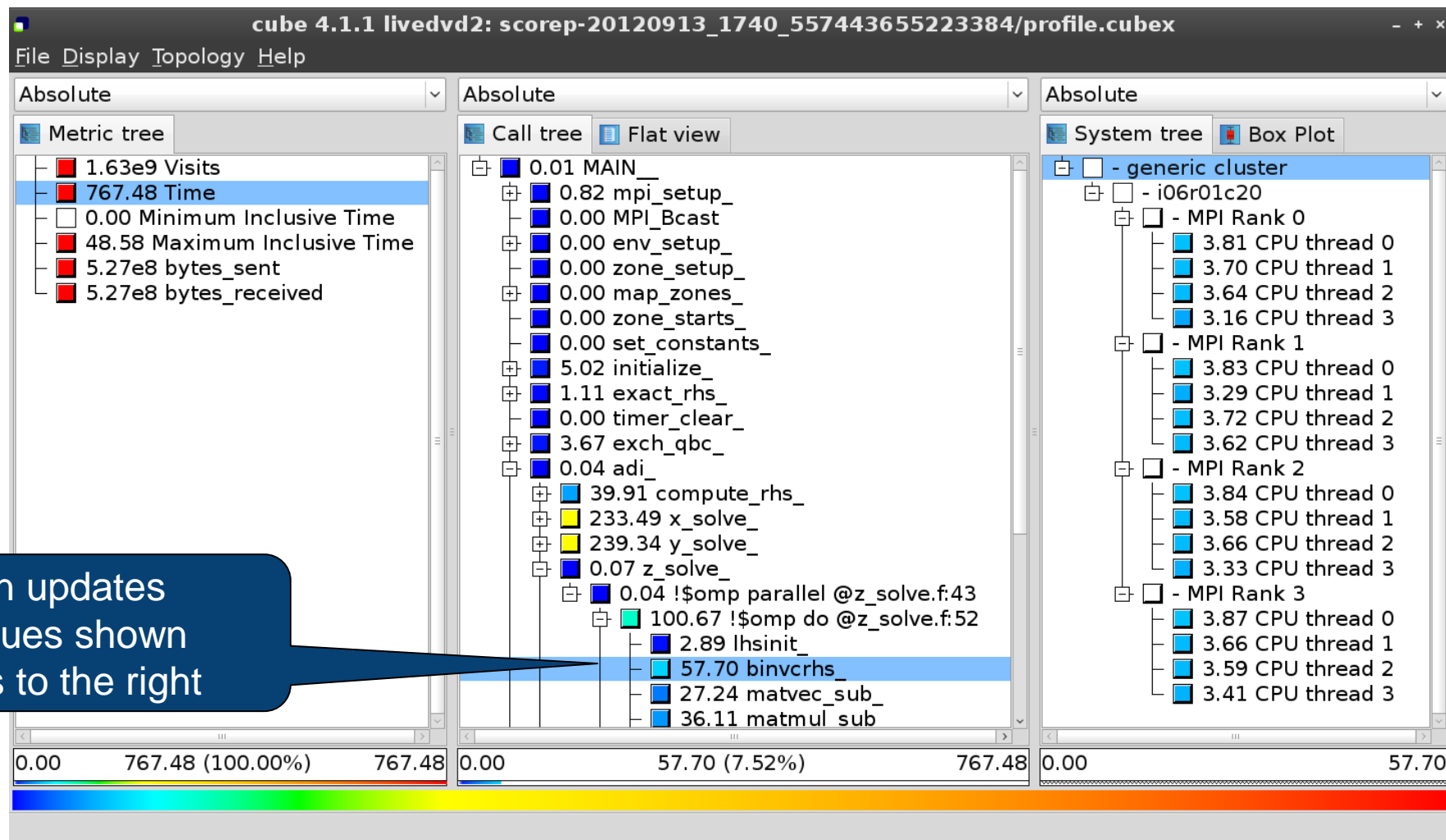




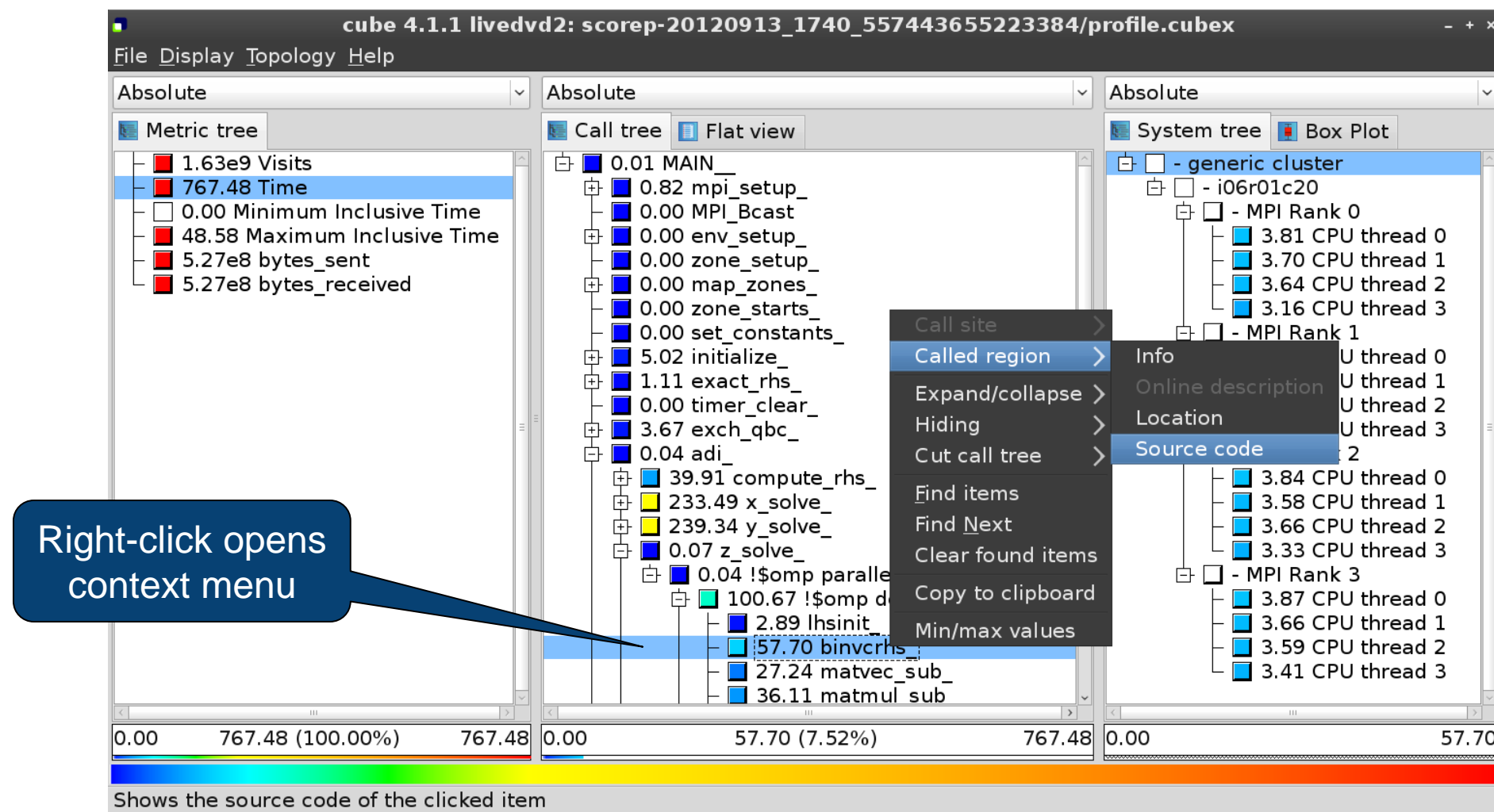
# Expanding the call tree



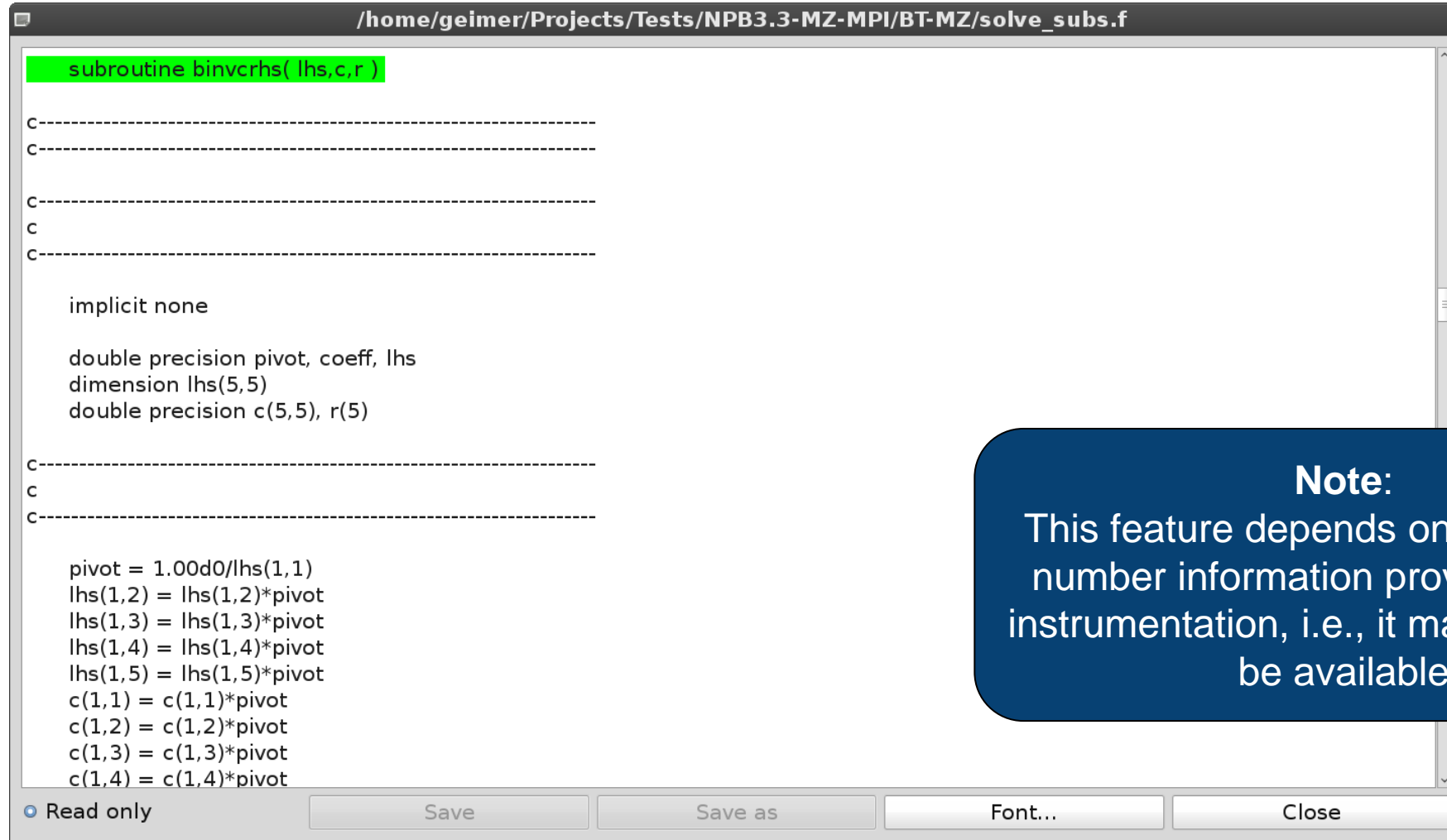
# Selecting a call path



# Source-code view via context menu



# Source-code view



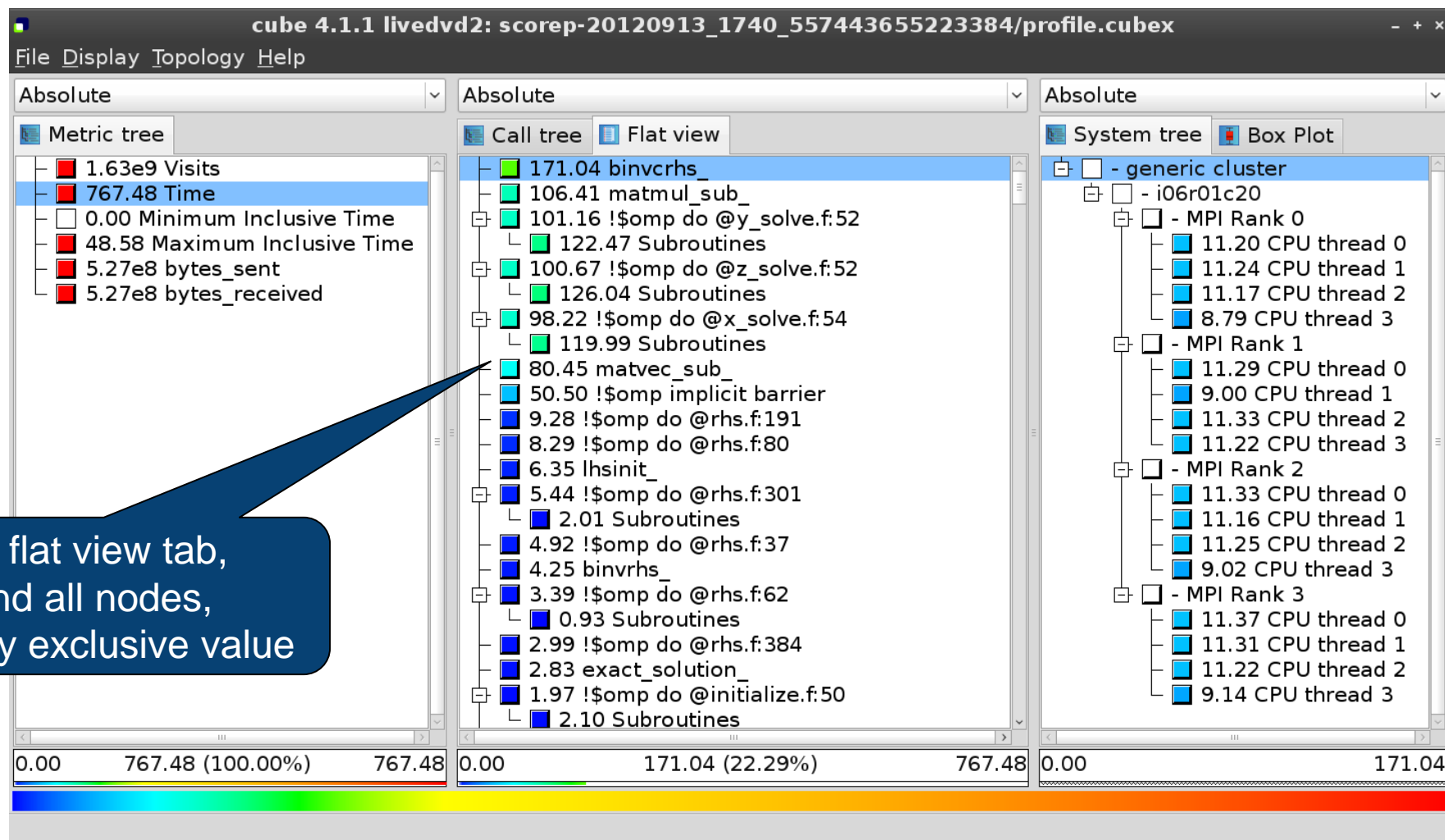
```
subroutine binvrhs( lhs,c,r )  
  
C-----  
C-----  
  
C-----  
C  
C-----  
  
implicit none  
  
double precision pivot, coeff, lhs  
dimension lhs(5,5)  
double precision c(5,5), r(5)  
  
C-----  
C  
C-----  
  
pivot = 1.00d0/lhs(1,1)  
lhs(1,2) = lhs(1,2)*pivot  
lhs(1,3) = lhs(1,3)*pivot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

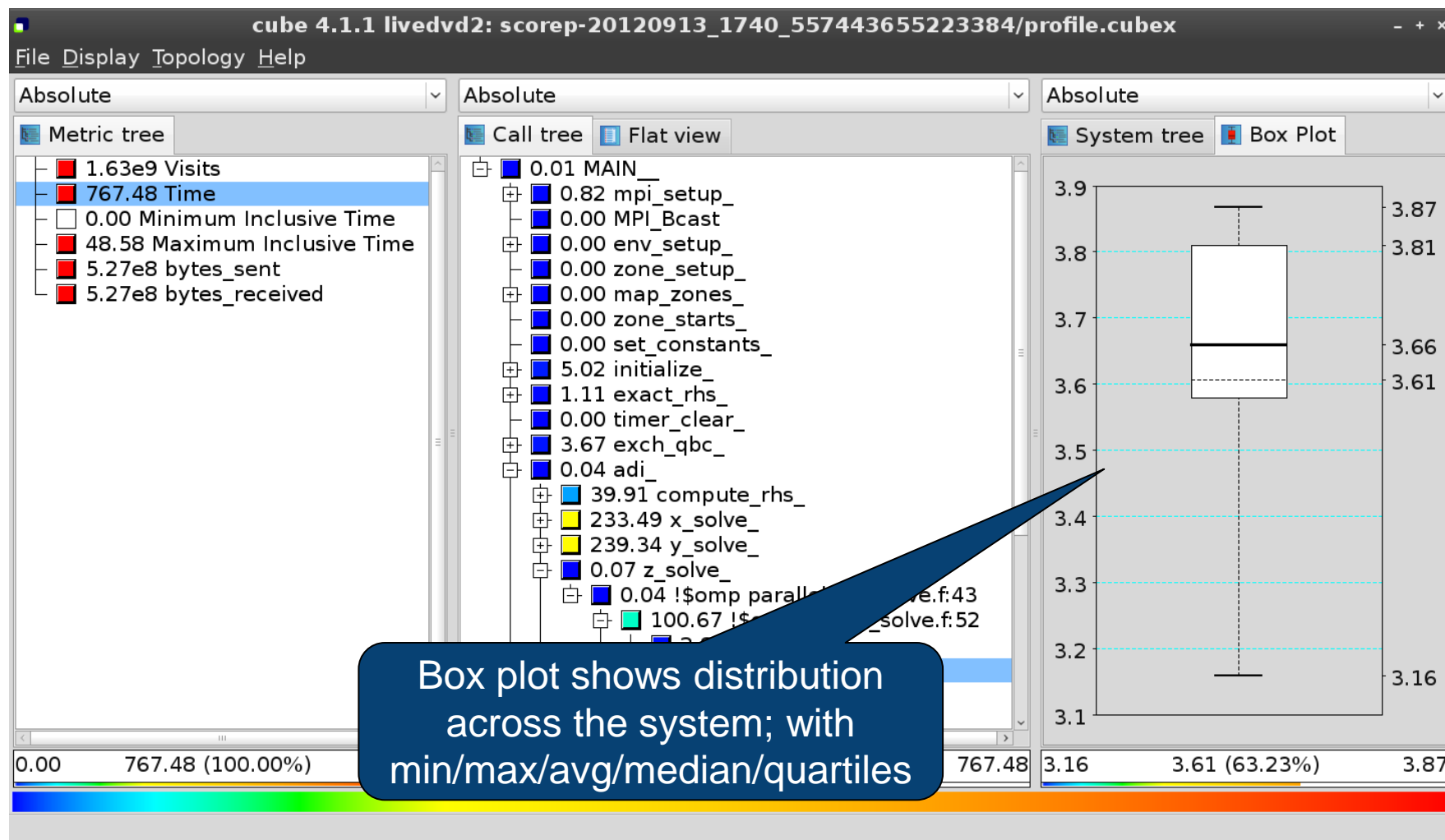
This feature depends on file and line number information provided by the instrumentation, i.e., it may not always be available

# Flat profile view

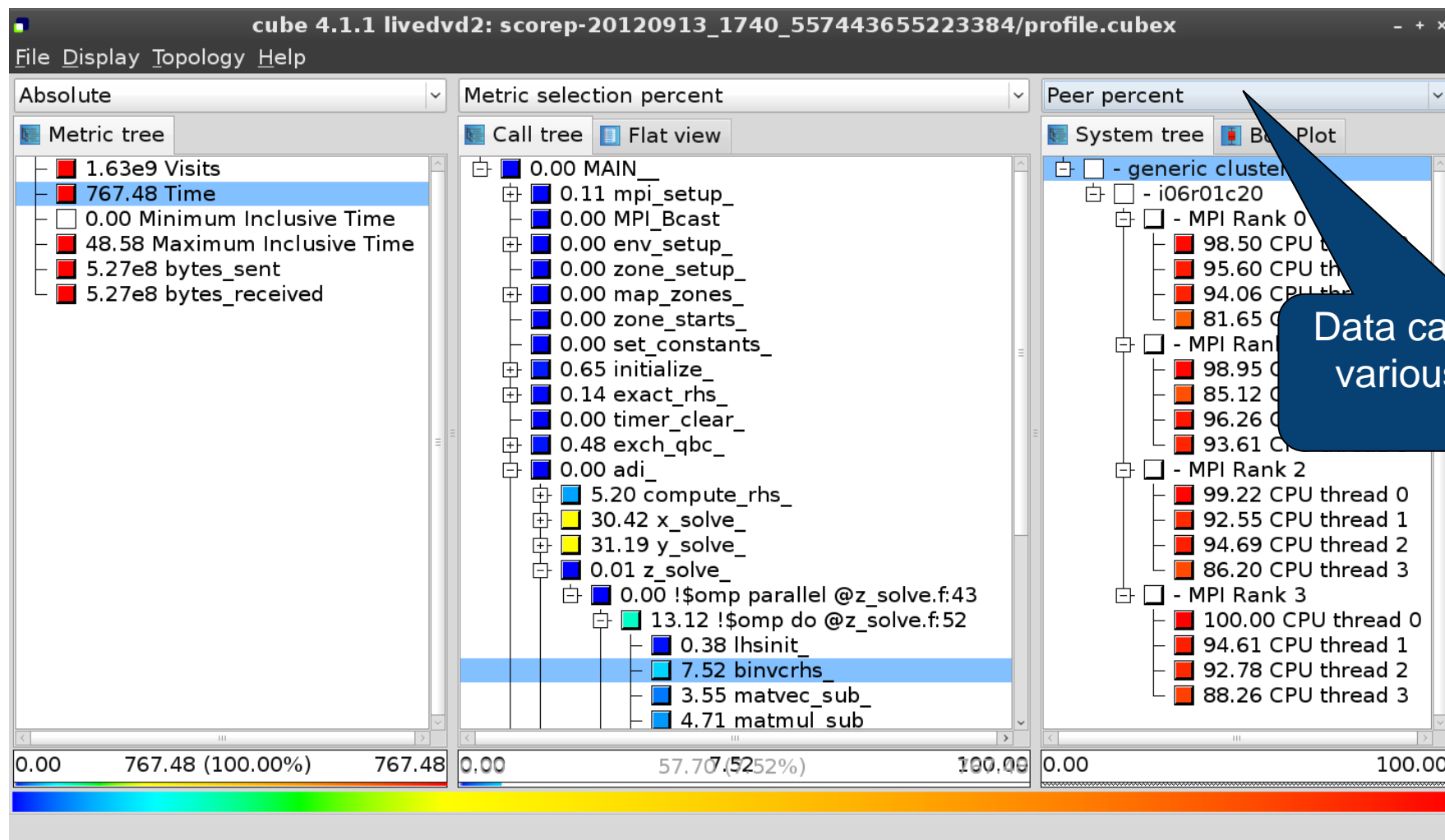




## Box plot view



# Alternative display modes

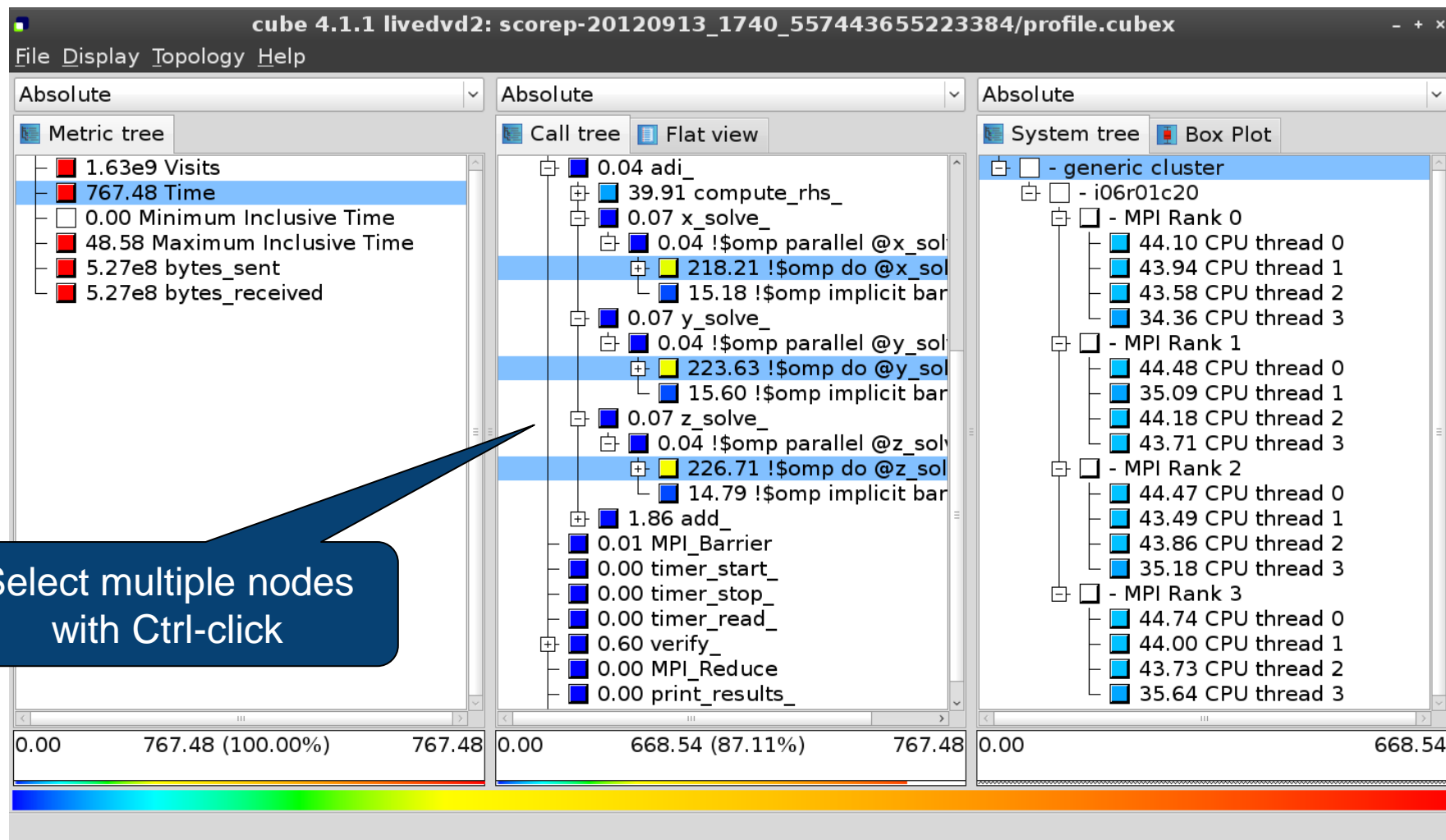


# Important display modes

---

- 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

The screenshot displays the cube 4.1.1 GUI with the title bar 'cube 4.1.1 livedvd2: scorep-20120913\_1740\_557443655223384/profile.cubex'. The 'Help' menu is open, showing options: 'Getting started', 'Mouse and keyboard control', 'What's This?' (highlighted with a mouse cursor and 'Shift+F1'), and 'About'. The 'Metric tree' on the left lists metrics like '1.63e9 Visits', '767.48 Time', and '0.00 Minimum I'. The central 'System tree' shows a hierarchy of MPI ranks and CPU threads. The bottom status bar shows three progress bars with values like '0.00 767.48 (100.00%) 767.48' and '0.00 668.54 (87.11%) 767.48'. A blue callout bubble points to the 'What's This?' menu item.

Context-sensitive help available for all GUI items

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()`**

# 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 )
```

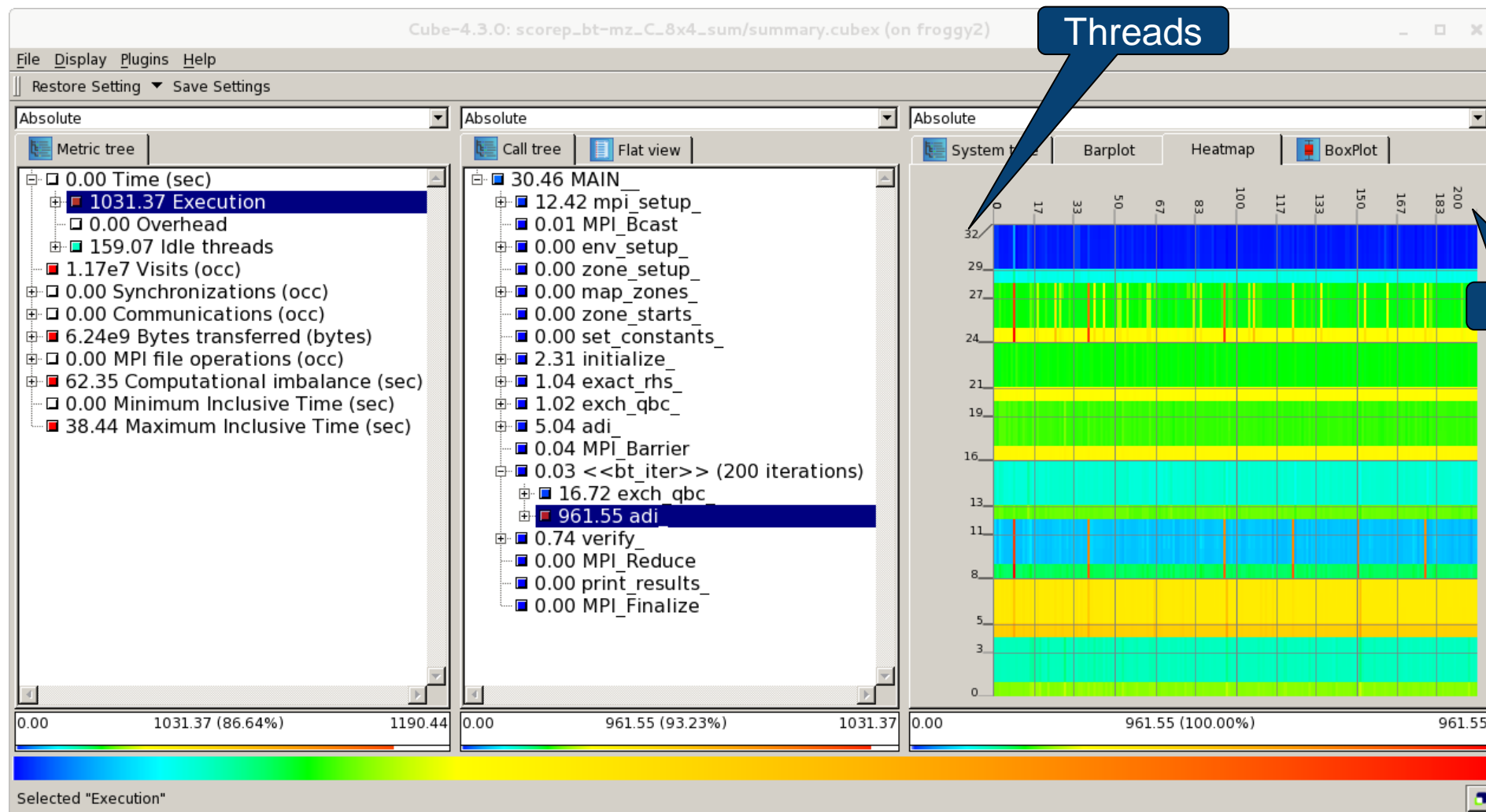
- Result in the Cube profile:
  - Iterations shown as separate call trees
    - Useful for checking results for specific iterations
  - or
  - Select your user-instrumented region and mark it as loop
  - Choose “Hide iterations”
    - View the Barplot statistics or the (thread x iterations) Heatmap

The screenshot displays the Cube-4.3.0 application interface, which is titled "Cube-4.3.0: scorep\_bt-mz\_C\_8x4\_sum/summary.cubex (on froggy2)". The interface is divided into several panels:

- Left Panel (Metric tree):** Shows a hierarchical tree of performance metrics. The "Execution" metric is highlighted, showing a value of 1031.37. Other metrics include Overhead (0.00), Idle threads (159.07), Visits (occ) (1.17e7), Synchronizations (occ) (0.00), Communications (occ) (0.00), Bytes transferred (bytes) (6.24e9), MPI file operations (occ) (0.00), Computational imbalance (sec) (62.35), Minimum Inclusive Time (sec) (0.00), and Maximum Inclusive Time (sec) (38.44).
- Middle Panel (Call tree / Flat view):** Shows a detailed breakdown of the execution time. The "MAIN" function is highlighted, showing a total time of 30.46. Sub-functions include mpi\_setup\_ (12.42), MPI\_Bcast\_ (0.01), env\_setup\_ (0.00), zone\_setup\_ (0.00), map\_zones\_ (0.00), zone\_starts\_ (0.00), set\_constants\_ (0.00), initialize\_ (2.31), exact\_rhs\_ (1.04), exch\_qbc\_ (1.02), adi\_ (5.04), MPI\_Barrier\_ (0.04), <<bt\_iter>> (200 iterations) (0.03), exch\_qbc\_ (16.72), adi\_ (961.55), verify\_ (0.74), MPI\_Reduce\_ (0.00), print\_results\_ (0.00), and MPI\_Finalize\_ (0.00).
- Right Panel (System tree / Barplot / Heatmap / BoxPlot):** Shows a box plot of the performance data. The x-axis is labeled "Iterations" and ranges from 0 to 200. The y-axis shows values from 0.1481 to 0.159. The plot displays a distribution of values for each iteration, with a red line indicating the maximum value and a blue line indicating the minimum value. The plot is titled "Aggregation selection" and "Iterations".

The bottom of the application shows a color-coded progress bar and a status bar with the following values: 0.00, 1031.37 (86.64%), 1190.44, 0.00, 961.55 (93.23%), 1031.37.

# 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](mailto:scalasca@fz-juelich.de)

