

# Analysis report examination with Cube

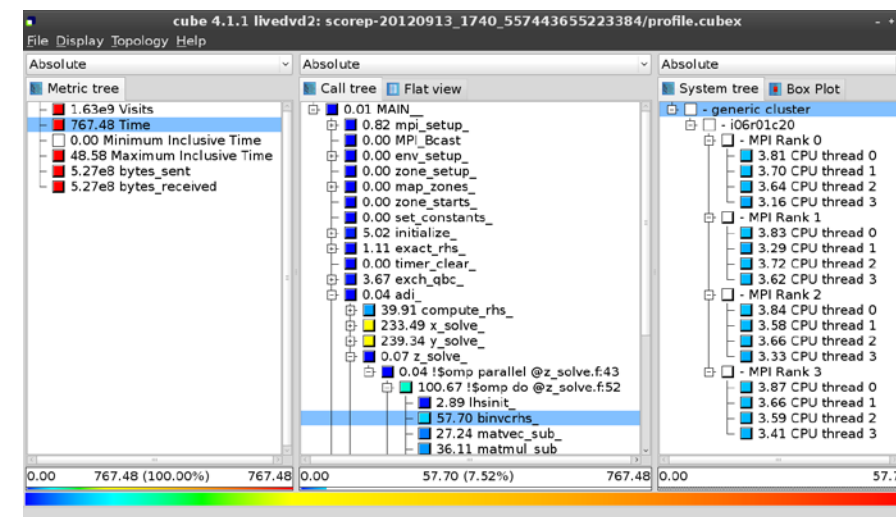
---

David Böhme  
Lawrence Livermore National Lab



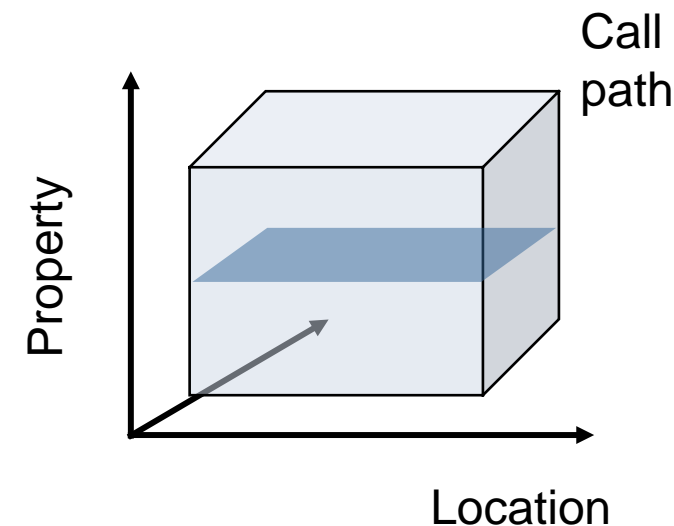
# Cube

- 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 4.3.4 (April 2016)



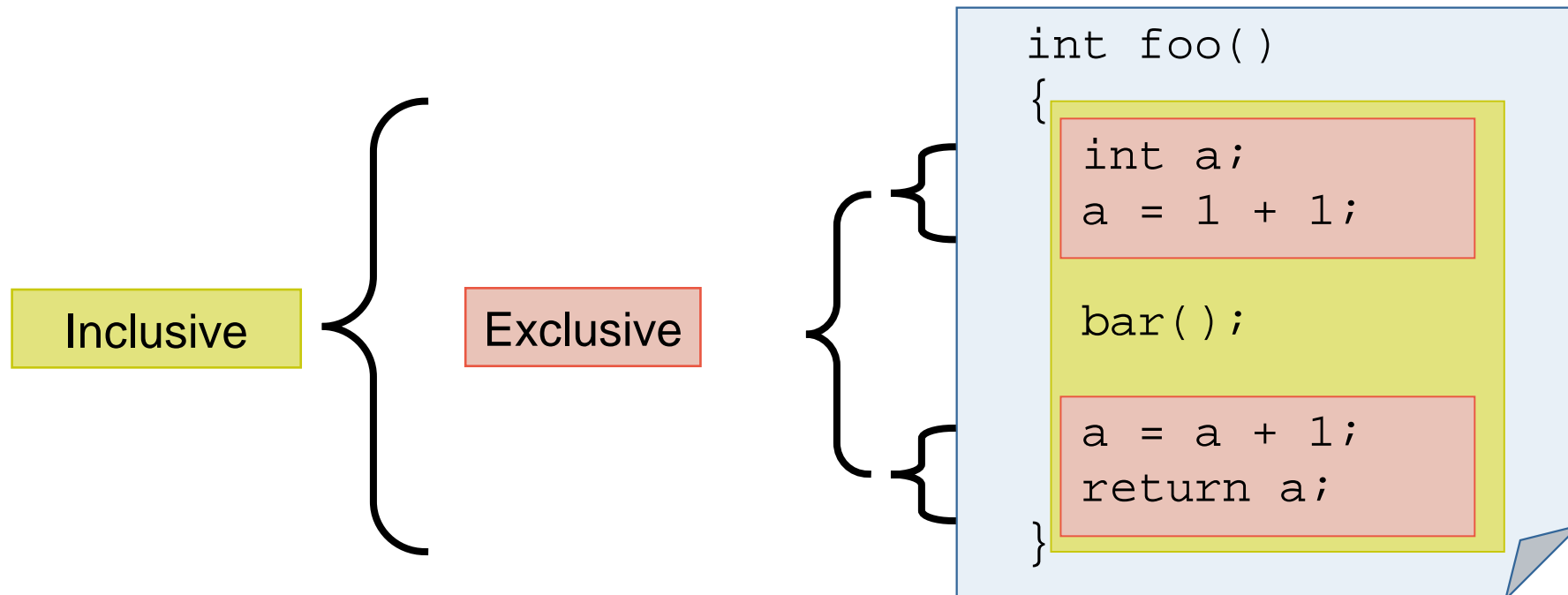
# Analysis presentation and exploration

- 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

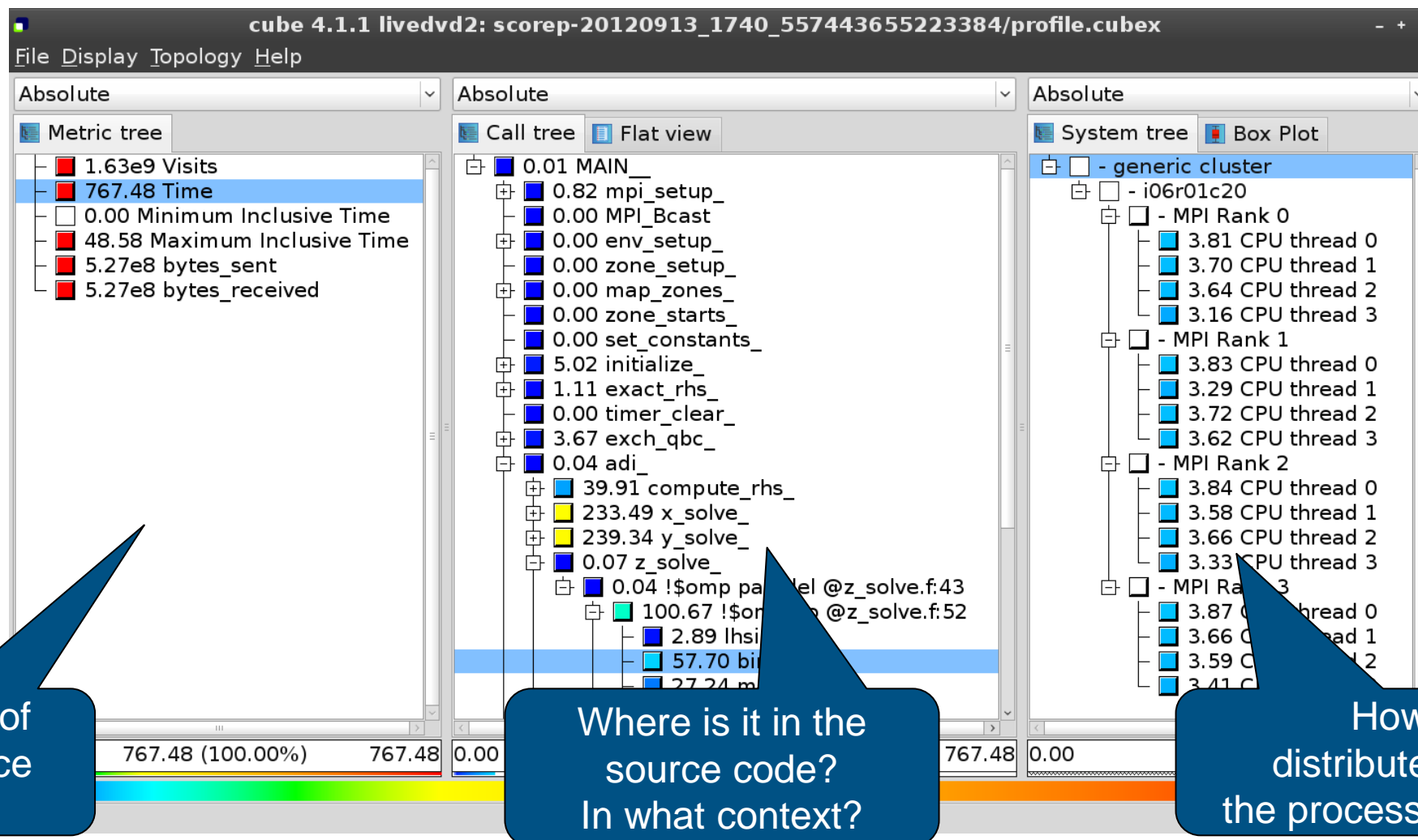


# Inclusive vs. exclusive values

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

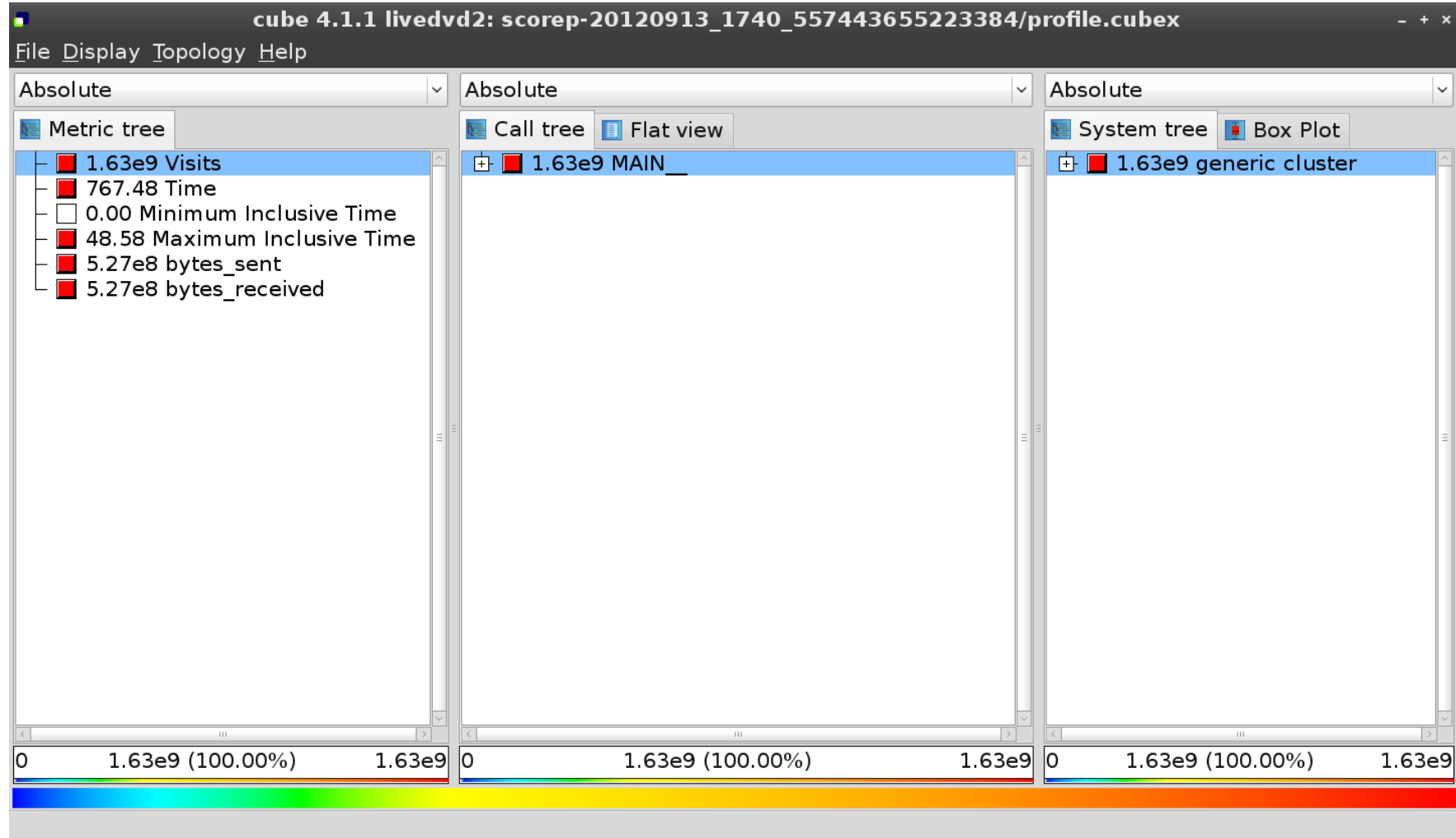


# Analysis presentation

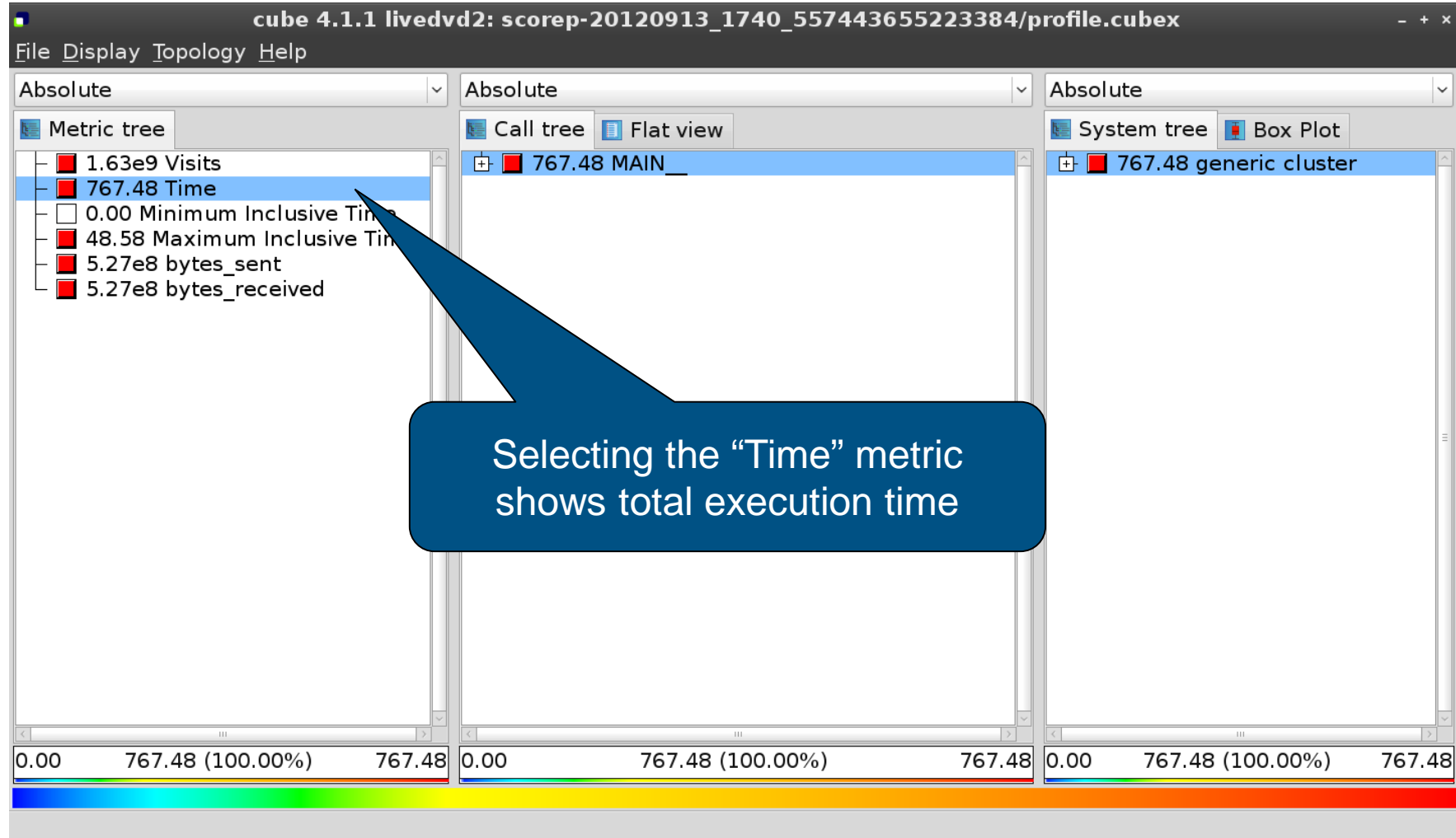




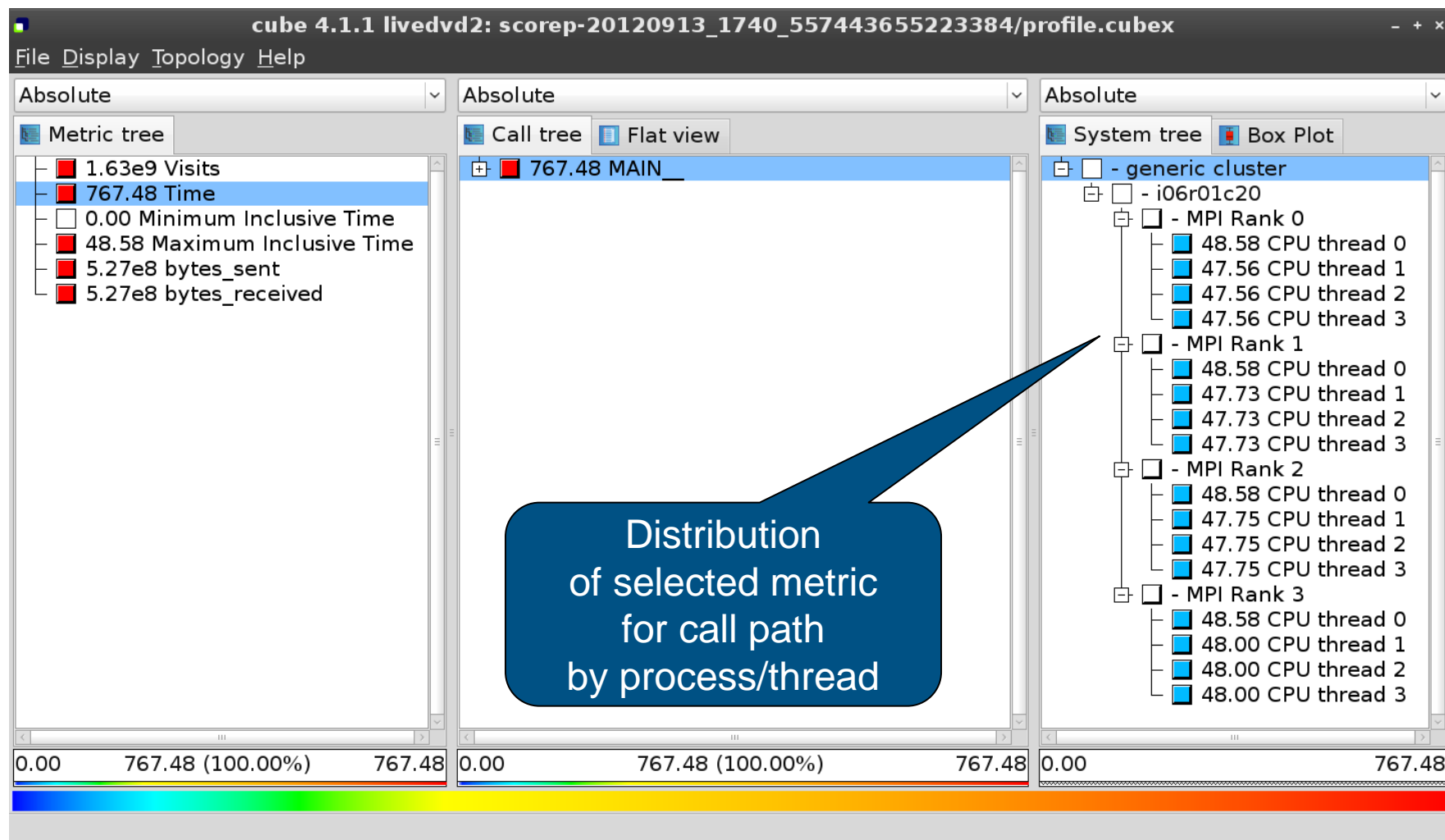
# Score-P analysis report exploration (opening view)



# Metric selection



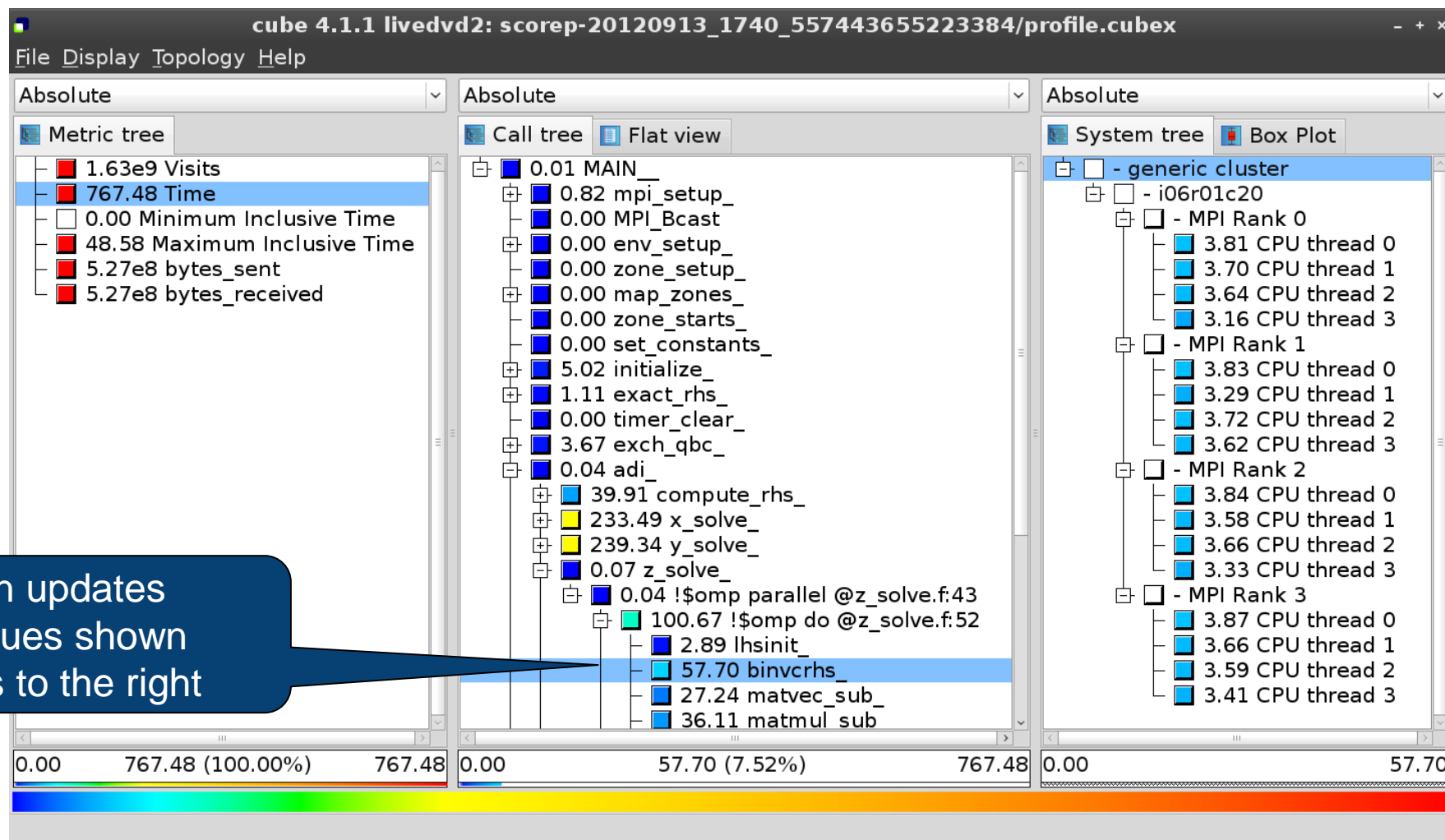
# Expanding the system tree



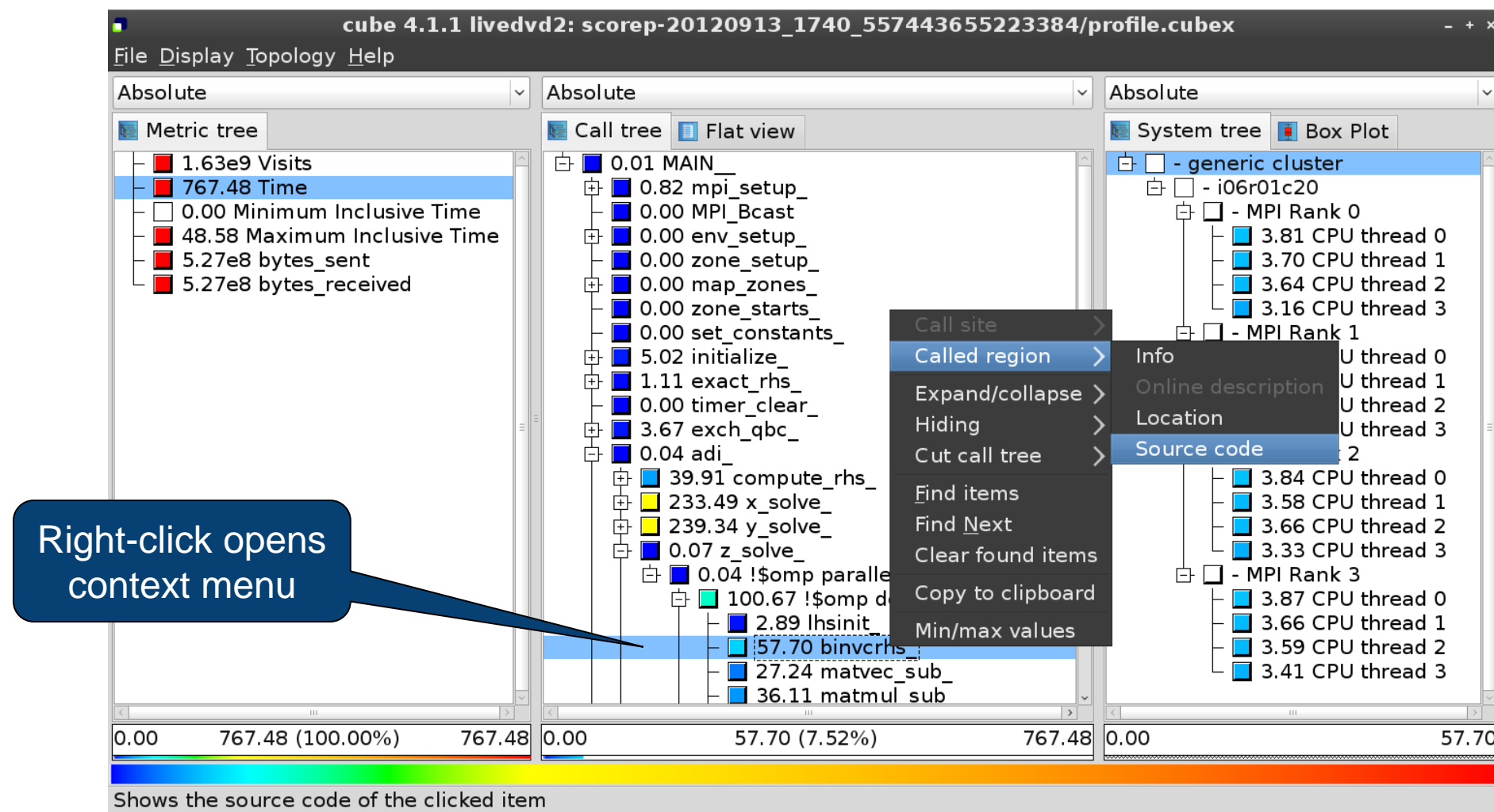




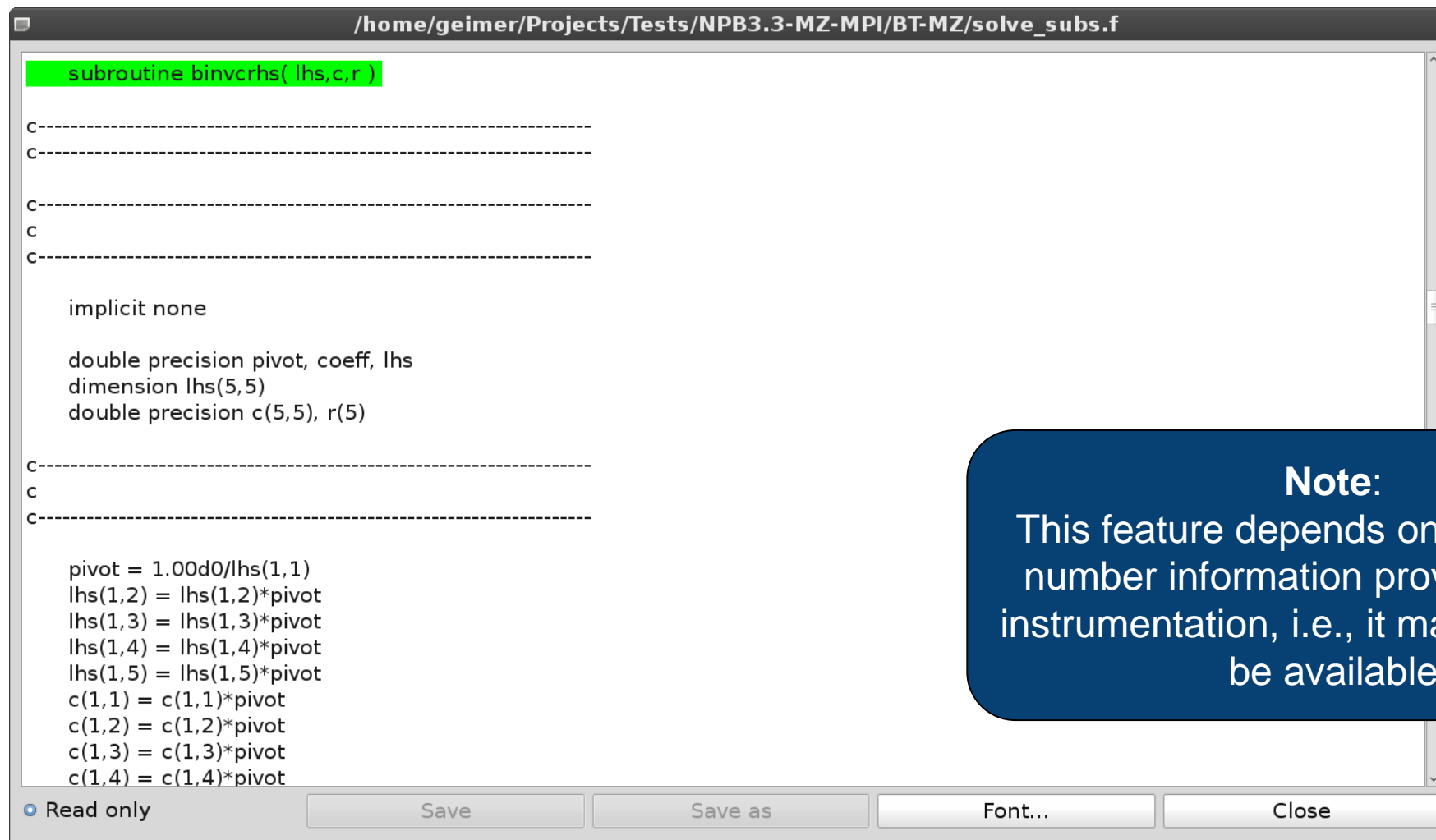
## Selecting a call path



## Source-code view via context menu



## Source-code view



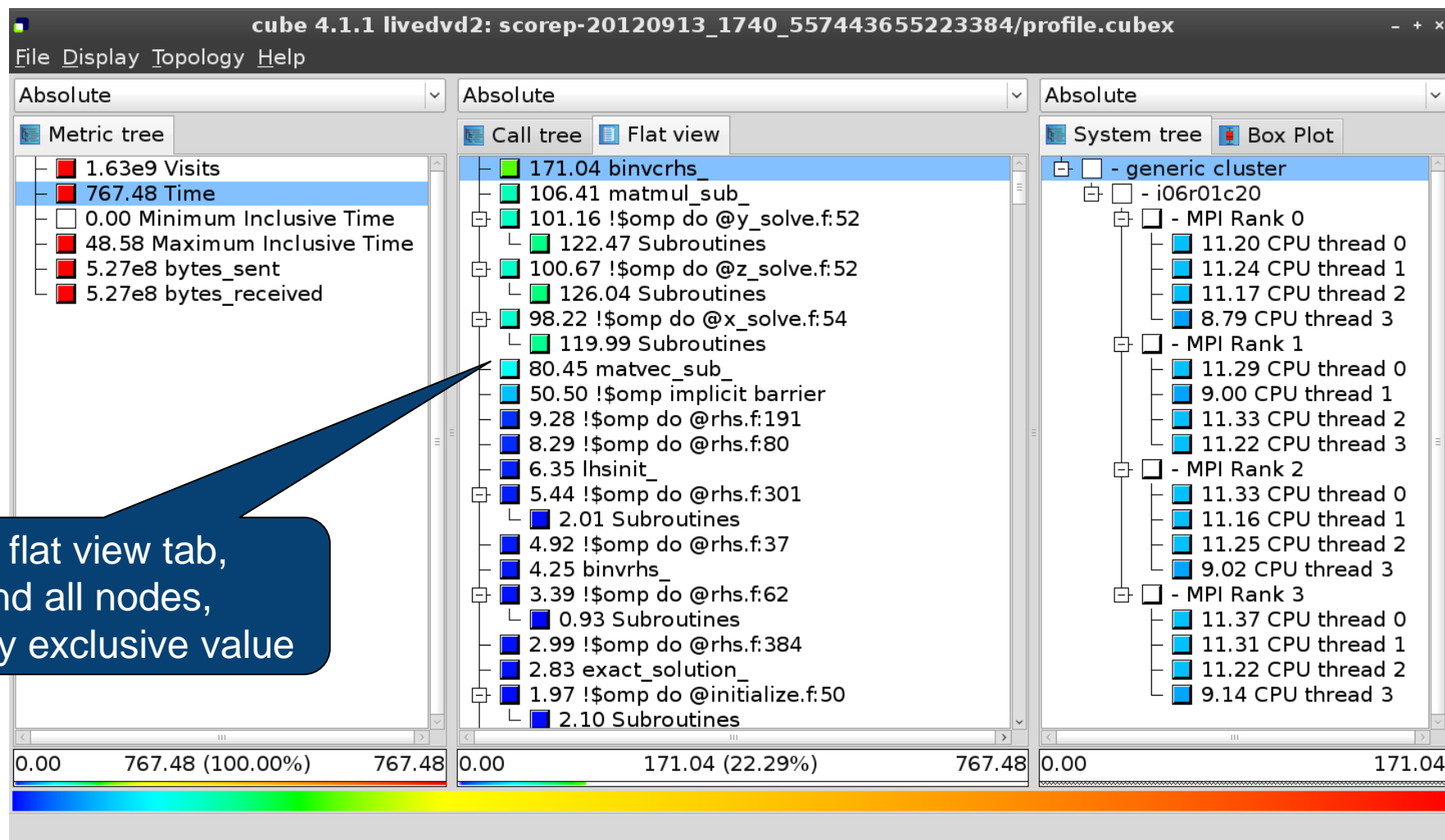
```
subroutine binvrhs( lhs,c,r )  
  
C-----  
C-----  
  
C-----  
C  
C-----  
  
implicit none  
  
double precision pivot, coeff, lhs  
dimension lhs(5,5)  
double precision c(5,5), r(5)  
  
C-----  
C  
C-----  
  
pivot = 1.00d0/lhs(1,1)  
lhs(1,2) = lhs(1,2)*pivot  
lhs(1,3) = lhs(1,3)*pivot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

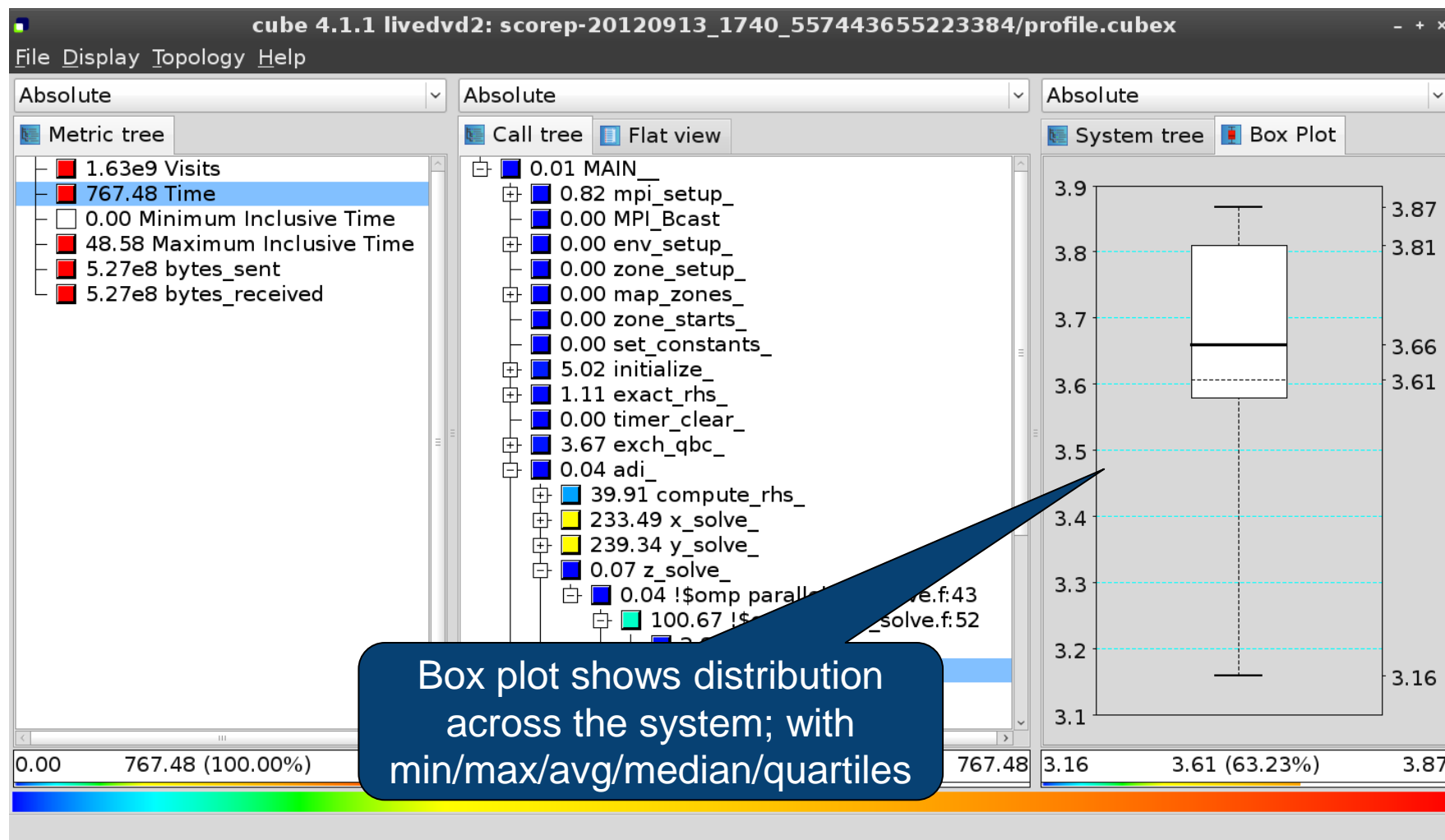
This feature depends on file and line number information provided by the instrumentation, i.e., it may not always be available

# Flat profile view

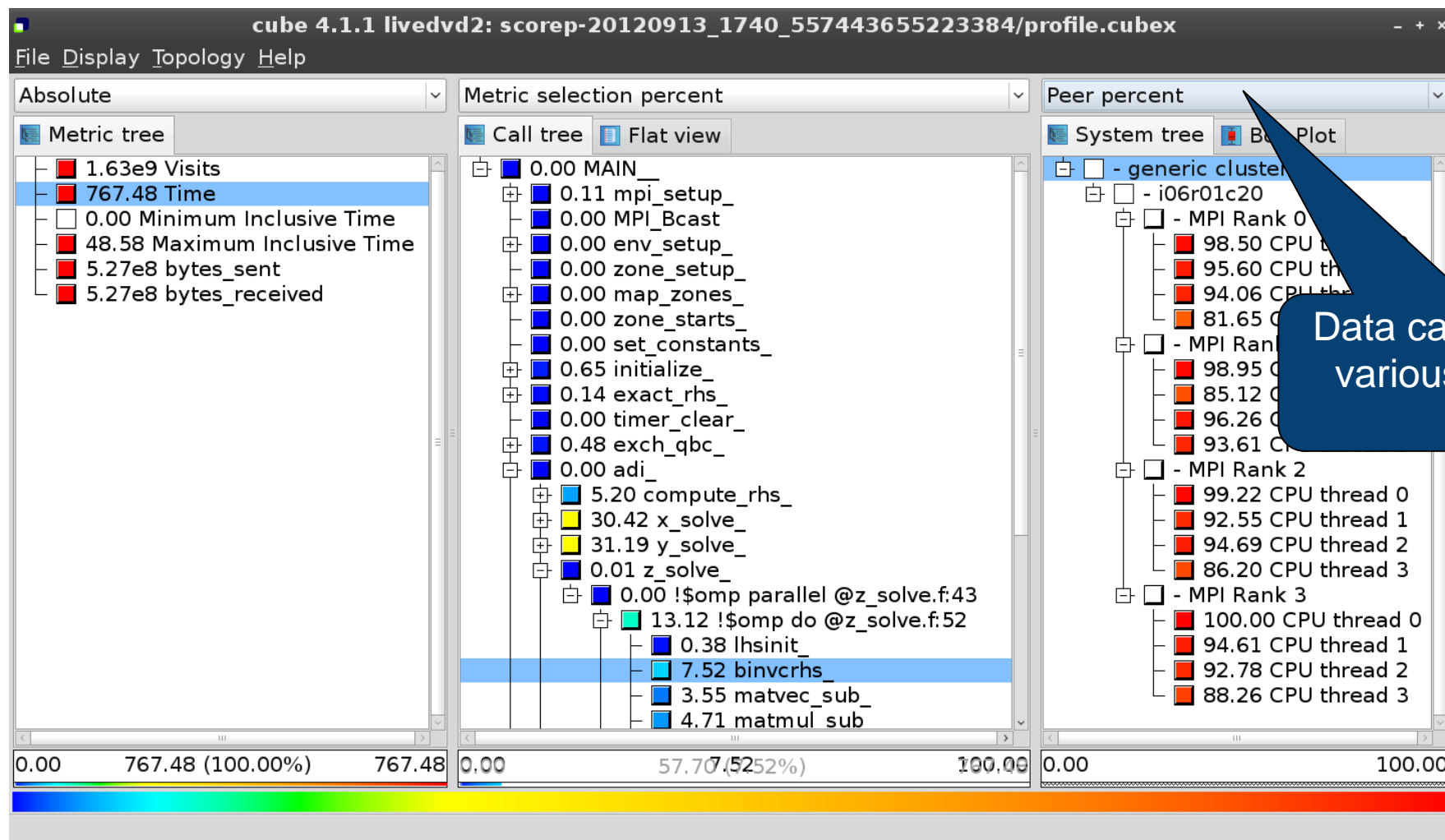




## Box plot view



# Alternative display modes

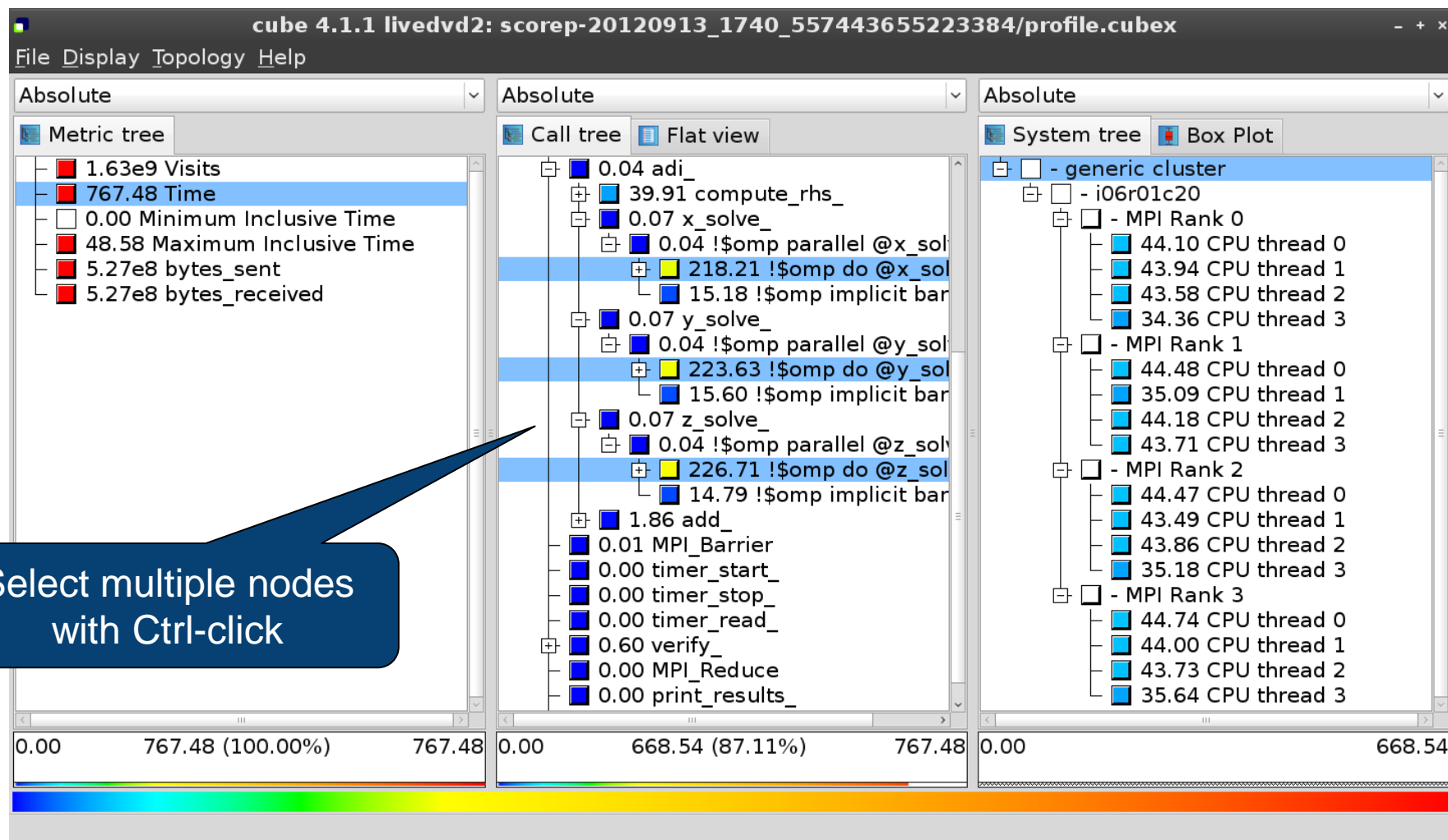


# Important display modes

---

- 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 the shortcut "Shift+F1"), and "About". The "Metric tree" on the left lists various metrics, with "767.48 Time" selected. The central "System tree" shows a hierarchical view of the application's execution, with "compute\_rhs\_" selected. The right panel shows a "Box Plot" of the "System tree" data. The bottom status bar displays three progress bars: "0.00 767.48 (100.00%) 767.48", "0.00 668.54 (87.11%) 767.48", and "0.00 668.54". A blue callout box on the left contains the text "Context-sensitive help available for all GUI items".

File Display Topology Help

Absolute

Metric tree

- 1.63e9 Visits
- 767.48 Time
- 0.00 Minimum I
- 48.58 Maximum
- 5.27e8 byt
- 5.27e8
- \_received

Getting started

Mouse and keyboard control

What's This? Shift+F1

About

Selected metrics description

Selected regions description

compute\_rhs\_

\_solve\_

4 !\$omp parallel @x\_sol

218.21 !\$omp do @x\_sol

15.18 !\$omp implicit bar

0.07 y\_solve\_

0.04 !\$omp parallel @y\_sol

223.63 !\$omp do @y\_sol

15.60 !\$omp implicit bar

0.07 z\_solve\_

0.04 !\$omp parallel @z\_sol

226.71 !\$omp do @z\_sol

14.79 !\$omp implicit bar

1.86 add\_

0.01 MPI\_Barrier

0.00 timer\_start\_

0.00 timer\_stop\_

0.00 timer\_read\_

0.60 verify\_

0.00 MPI\_Reduce

0.00 print\_results\_

Absolute

System tree Box Plot

- generic cluster
  - i06r01c20
    - MPI Rank 0
      - 44.10 CPU thread 0
      - 43.94 CPU thread 1
      - 43.58 CPU thread 2
      - 34.36 CPU thread 3
    - MPI Rank 1
      - 44.48 CPU thread 0
      - 35.09 CPU thread 1
      - 44.18 CPU thread 2
      - 43.71 CPU thread 3
    - MPI Rank 2
      - 44.47 CPU thread 0
      - 43.49 CPU thread 1
      - 43.86 CPU thread 2
      - 35.18 CPU thread 3
    - MPI Rank 3
      - 44.74 CPU thread 0
      - 44.00 CPU thread 1
      - 43.73 CPU thread 2
      - 35.64 CPU thread 3

0.00 767.48 (100.00%) 767.48

0.00 668.54 (87.11%) 767.48

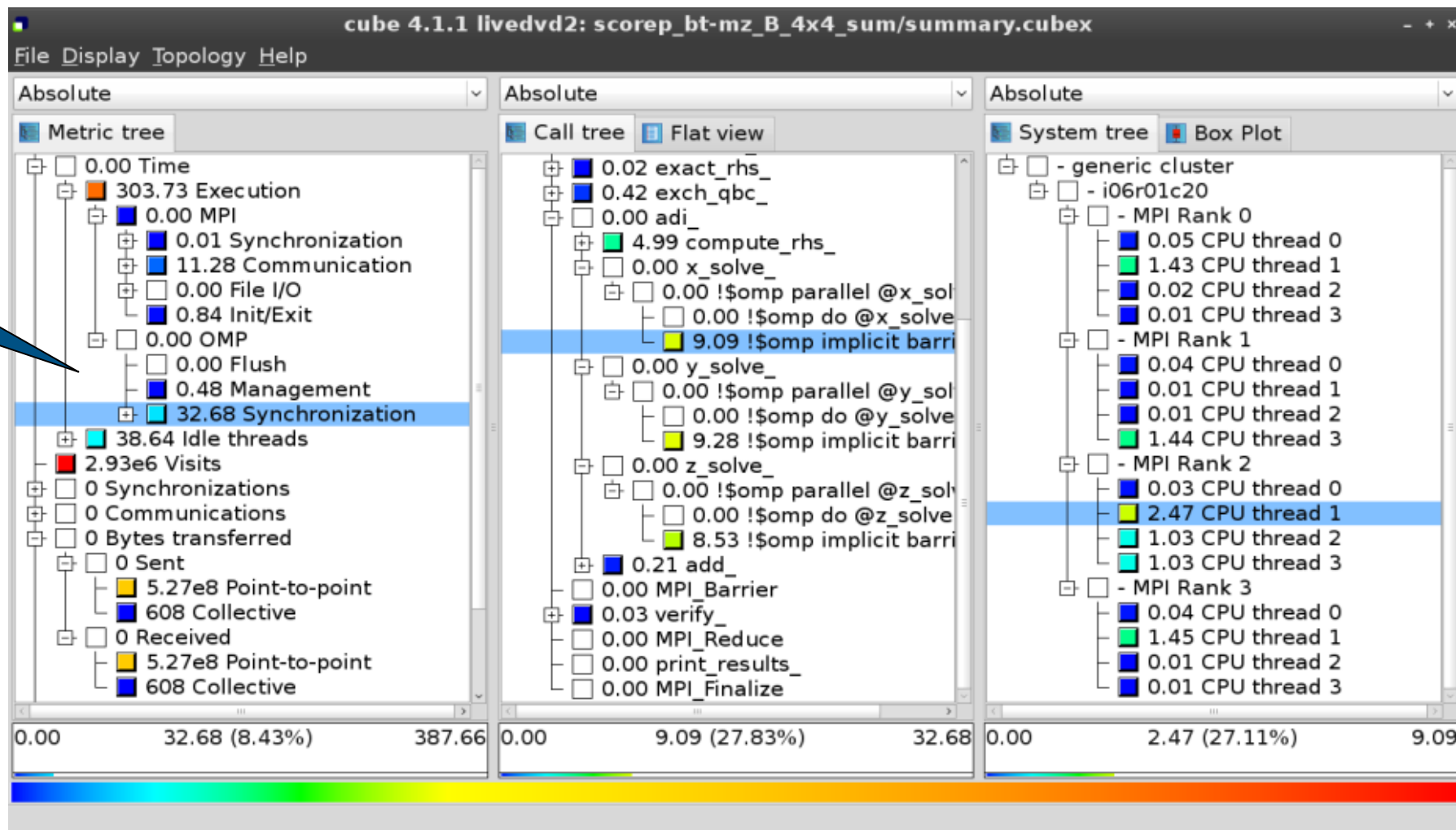
0.00 668.54

Change into help mode for display components

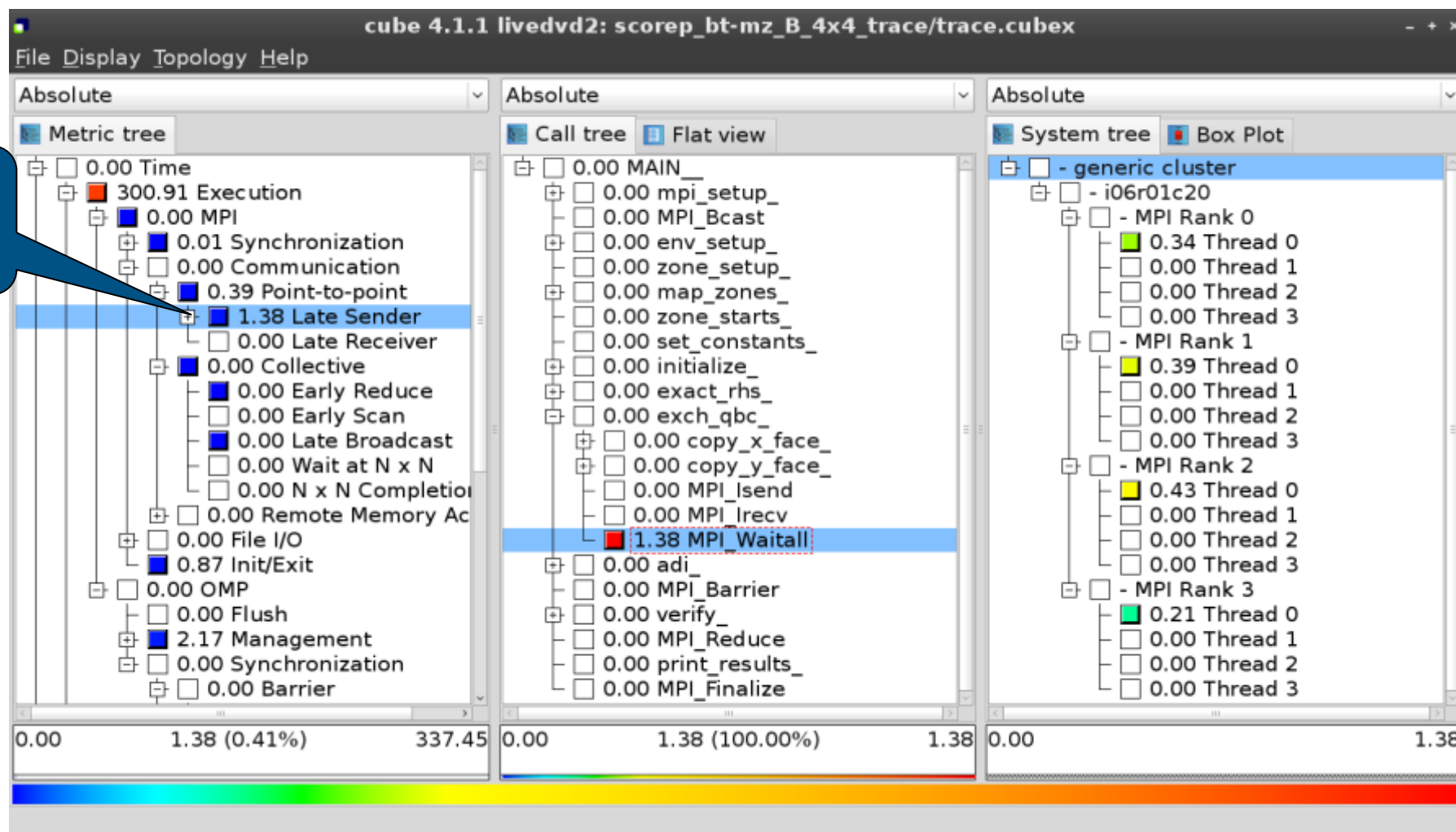


# Scalasca's post-processed summary analysis report

Split base metrics into more specific metrics

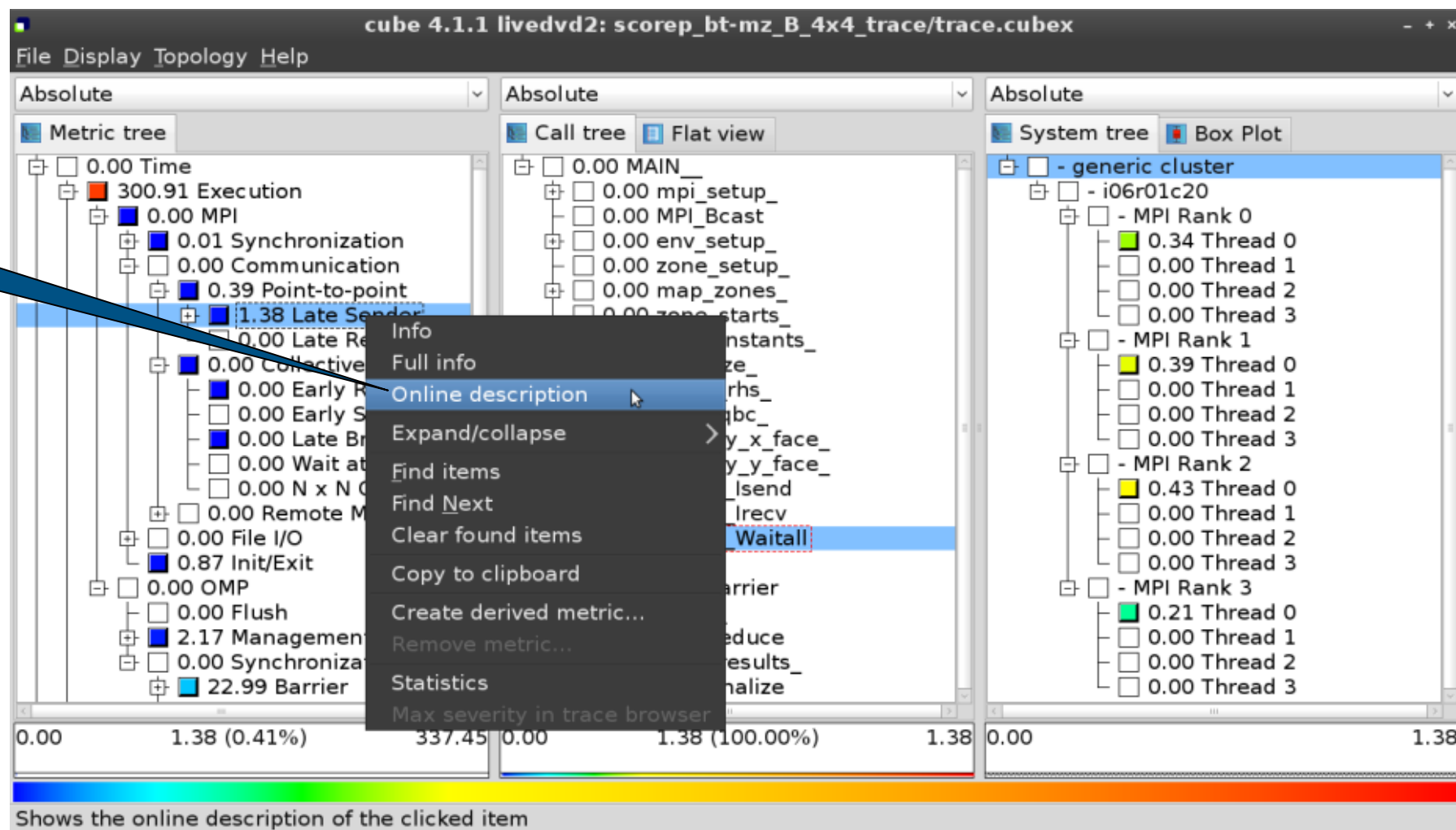


# Scalasca's post-processed trace analysis report



# Online metric description

Access online metric description via context menu

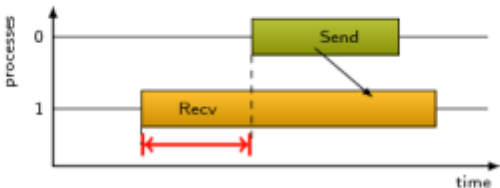


# Online metric description

**Performance properties**

**Late Sender Time**

**Description:**  
Refers to the time lost waiting caused by a blocking receive operation (e.g., `MPI_Recv` or `MPI_Wait`) that is posted earlier than the corresponding send operation.



If the receiving process is waiting for multiple messages to arrive (e.g., in an call to `MPI_Waitall`), the maximum waiting time is accounted, i.e., the waiting time due to the latest sender.

**Unit:**  
Seconds

**Diagnosis:**  
Try to replace `MPI_Recv` with a non-blocking receive `MPI_Irecv` that can be posted earlier, proceed concurrently with computation, and complete with a wait operation after the message is expected to have been sent. Try to post sends earlier, such that they are available when receivers need them. Note that outstanding messages (i.e., sent before the receiver is ready) will occupy internal message buffers, and that large numbers of posted receive buffers will also introduce message management overhead, therefore moderation is advisable.

**Parent:**  
[MPI Point-to-point Communication Time](#)

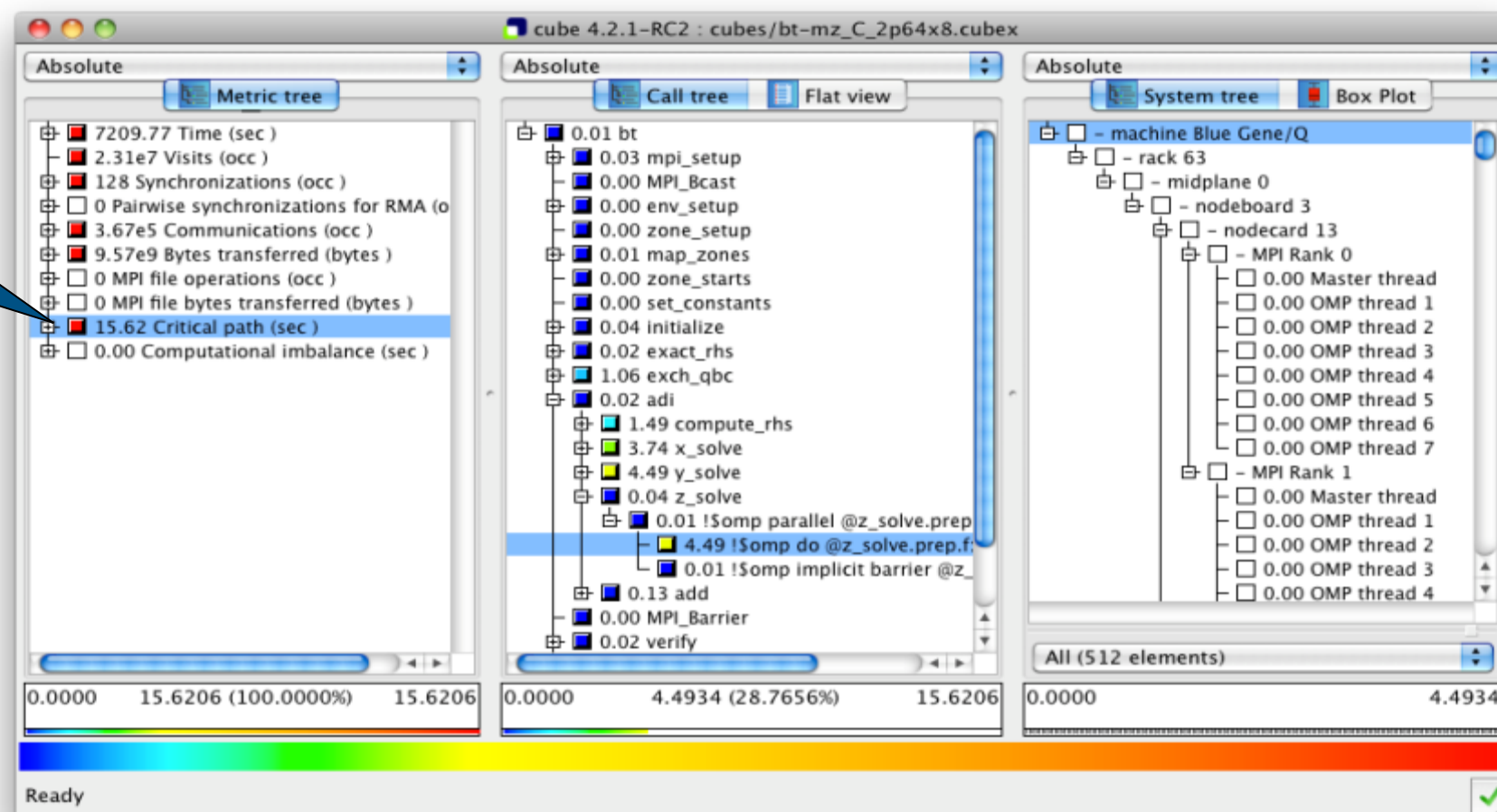
**Children:**

Close



# Scalasca critical-path analysis

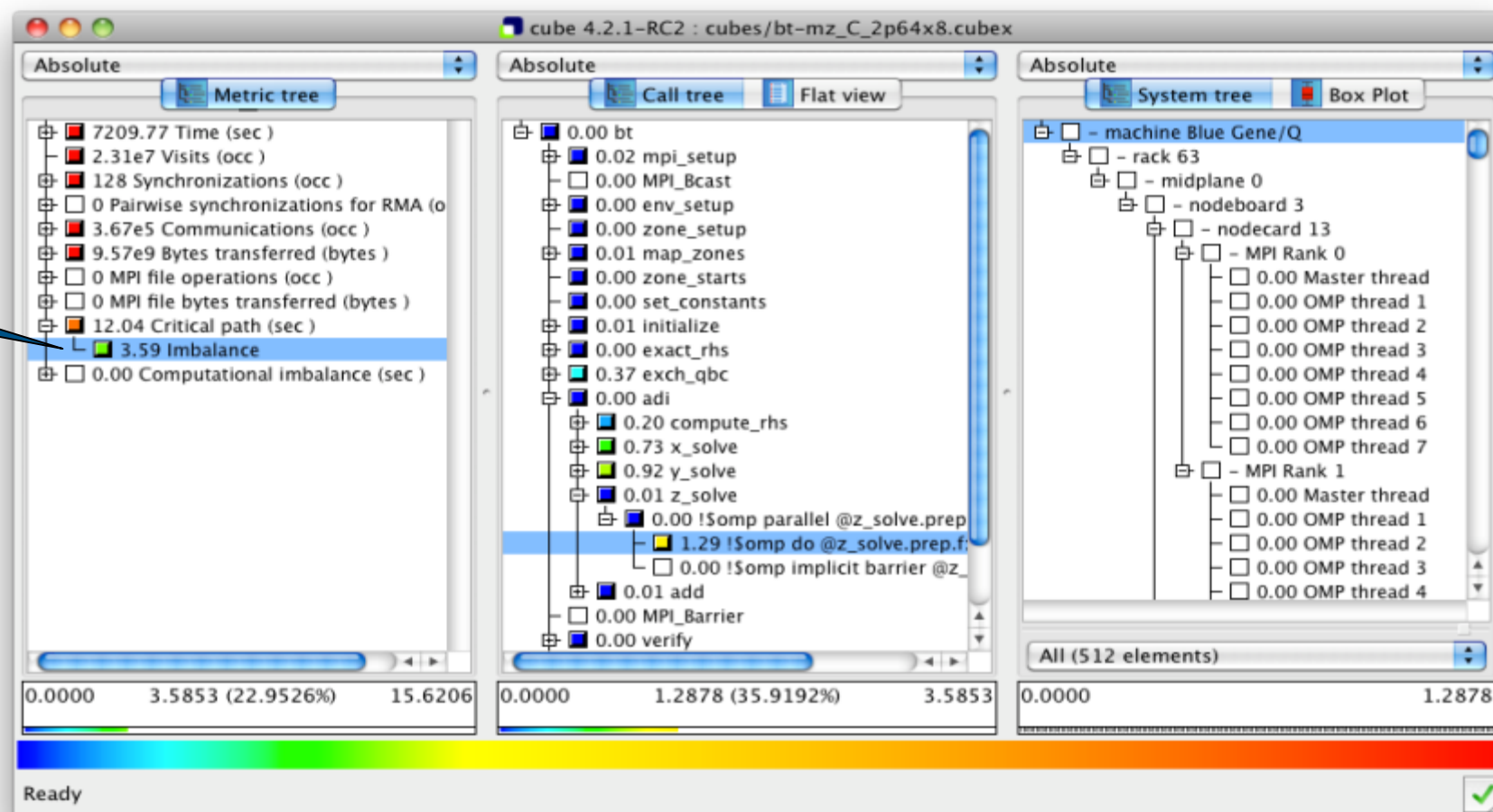
Critical-path profile shows  
wall-clock time impact



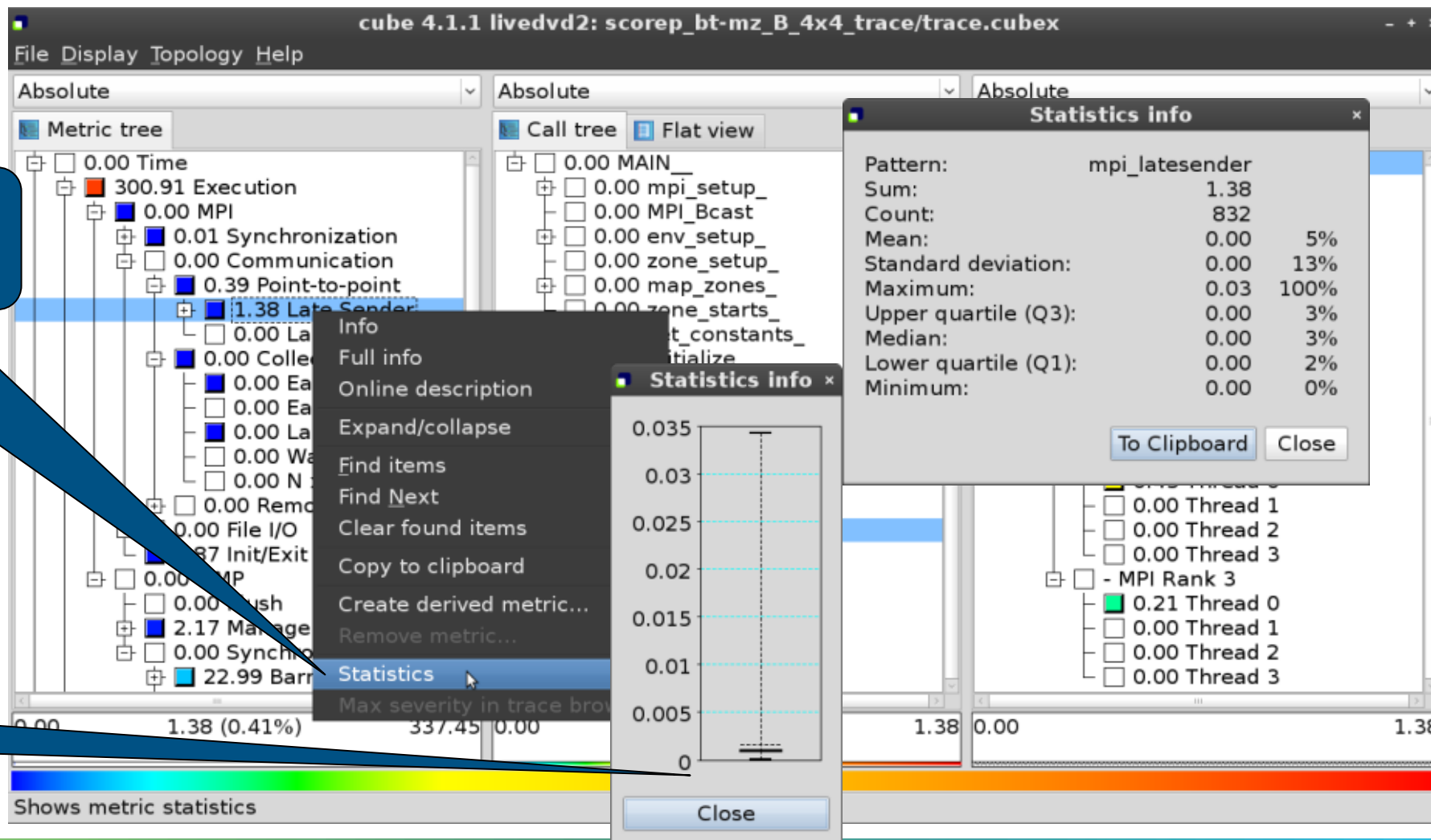


# Scalasca critical-path analysis

Critical-path imbalance  
highlights inefficient  
parallelism

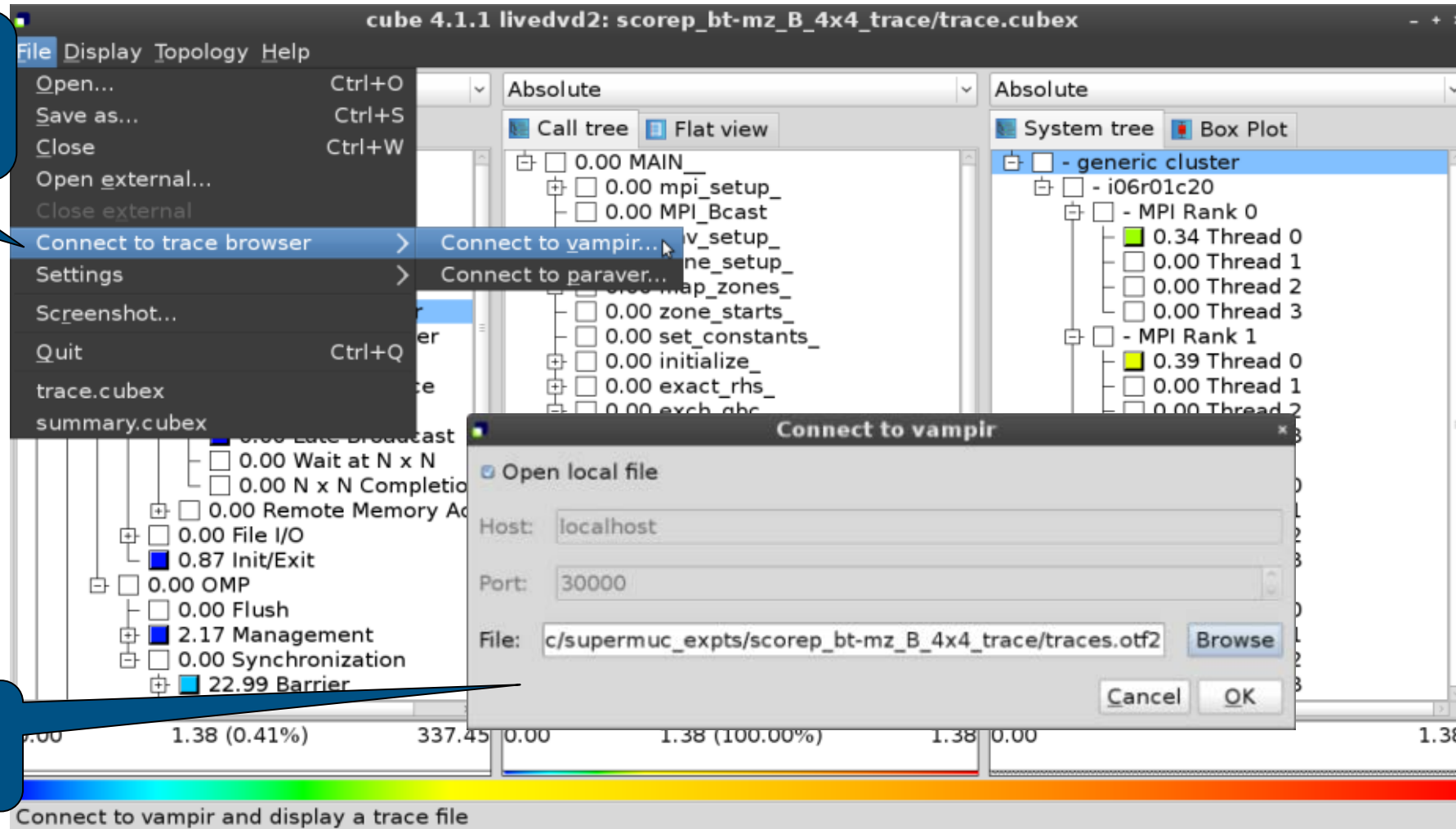


# Scalasca pattern instance statistics



## Connect to Vampir trace browser

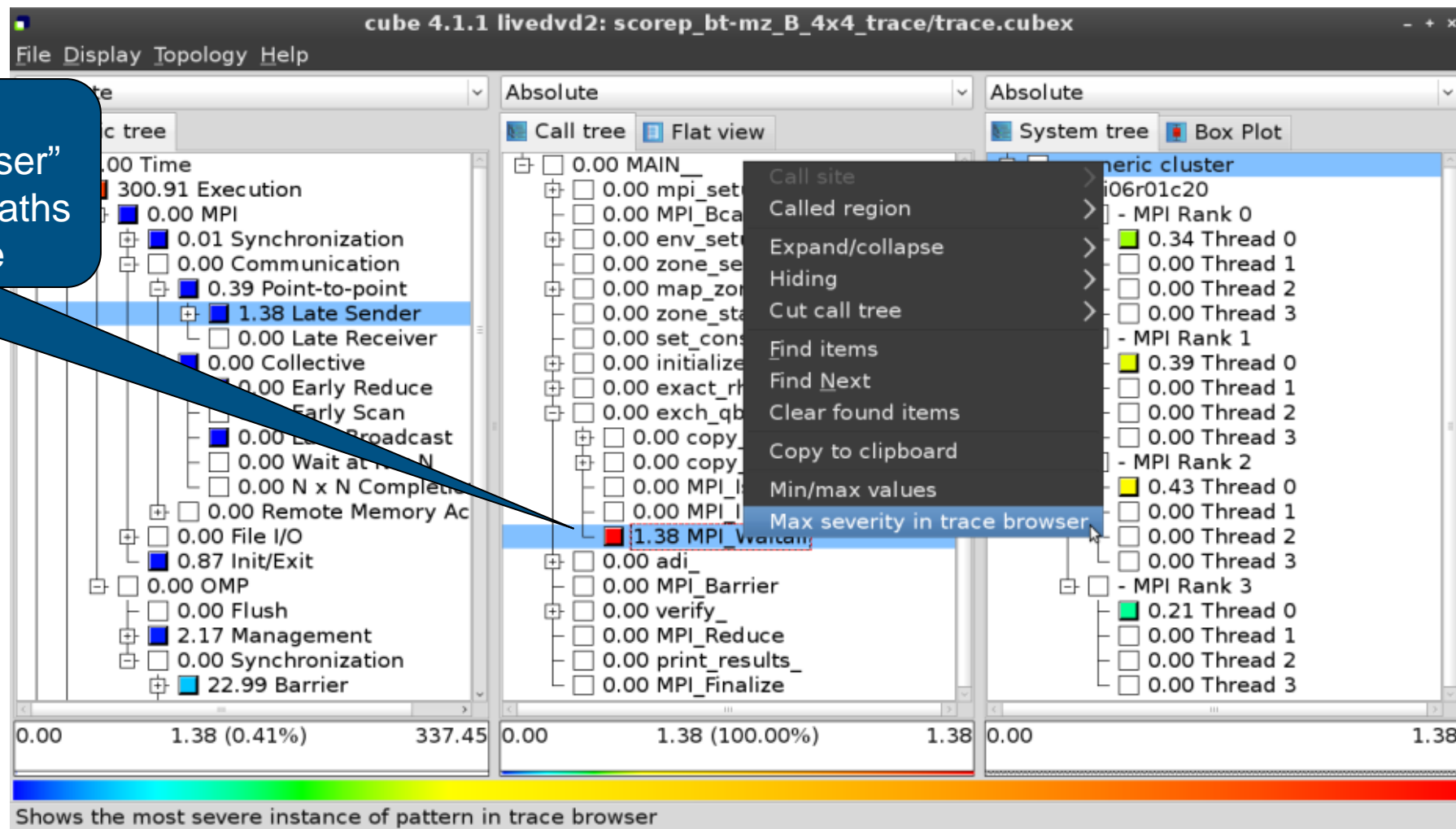
To investigate most severe pattern instances, connect to a trace browser...



...and select trace file from the experiment directory

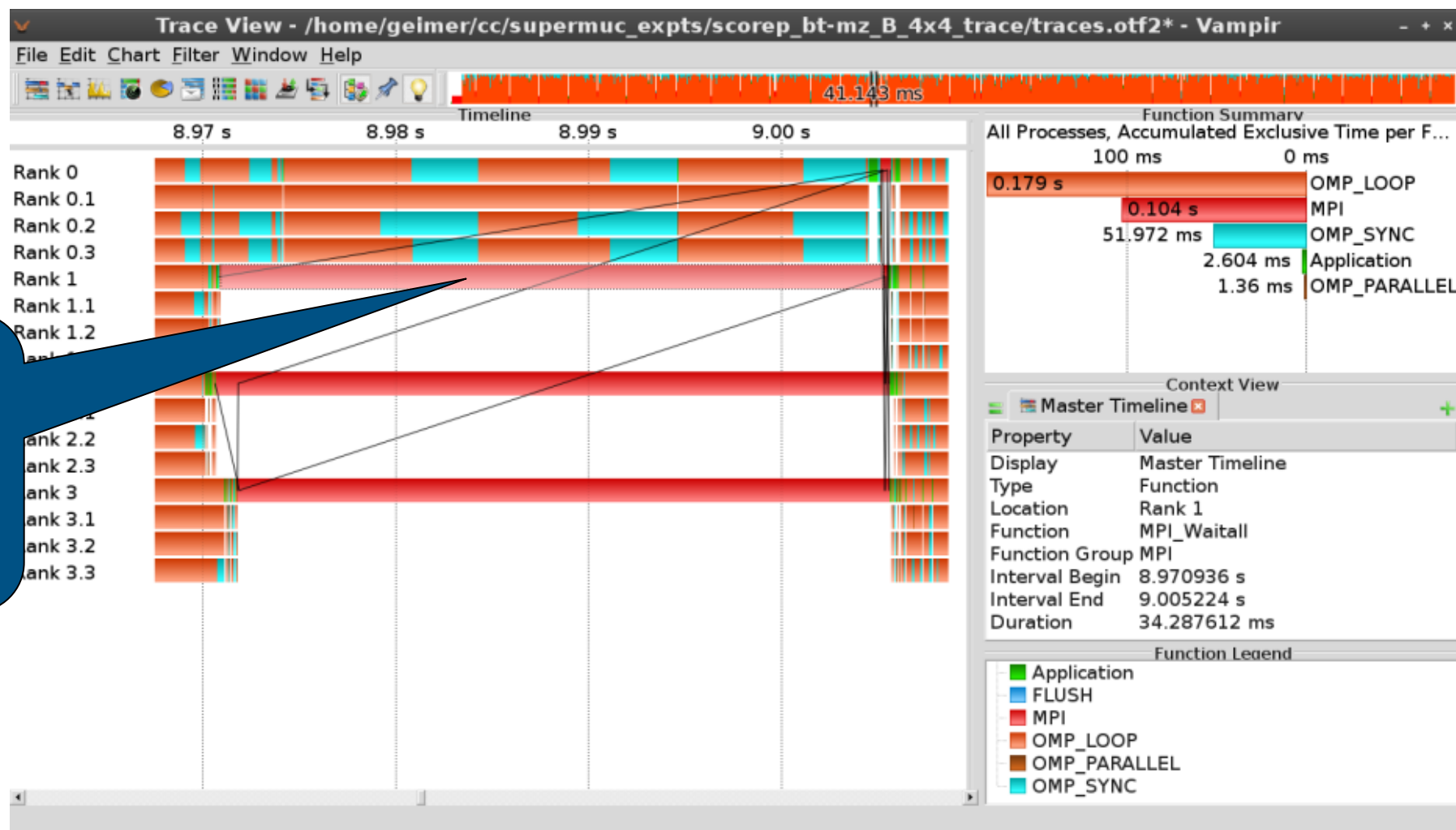
## Show most severe pattern instances

Select  
“Max severity in trace browser”  
from context menu of call paths  
marked with a red frame





# Investigate most severe instance in Vampir





## 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

1.01e6 (100.00%) 1.01e6 0.00

2512.10

# Example: FLOPS based on PAPI\_FP\_OPS and time

The screenshot displays the Cube-4.3.1 performance analysis tool interface, showing the calculation of FLOPS based on PAPI\_FP\_OPS and time.

**Left Panel: 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: `metric::PAPI_FP_OPS()/metric::time()`
- Buttons: Edit metric, Cancel
- Share this metric with SCALASCA group

**Right Panel: Cube-4.3.1: scorep\_8x4\_sum/profile.cubex (on froggy1)**

The right panel shows the hierarchical metric tree with three views: Metric tree, Call tree, and Flat view. The Metric tree view is selected, showing the following metrics:

- 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

The Call tree view shows the following metrics:

- 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

The System tree view shows the following metrics:

- 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

The bottom of the right panel shows a color bar and 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

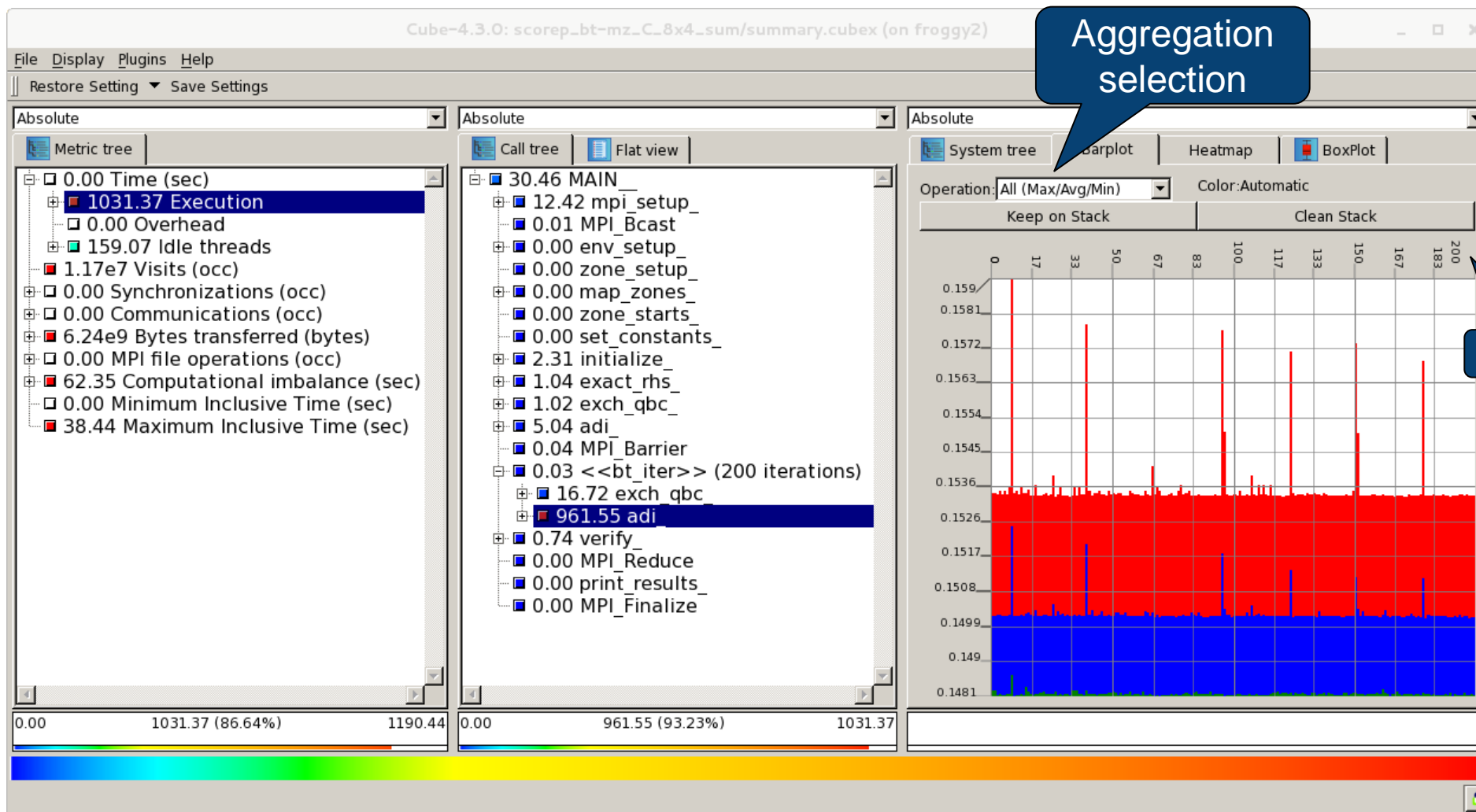
```
SCOREP_USER_REGION_DEFINE( scorep_bt_loop )  
SCOREP_USER_REGION_BEGIN( scorep_bt_loop, "<<bt_iter>>", SCOREP_USER_REGION_TYPE_DYNAMIC )  
SCOREP_USER_REGION_END( scorep_bt_loop )
```

- Result in the Cube profile:

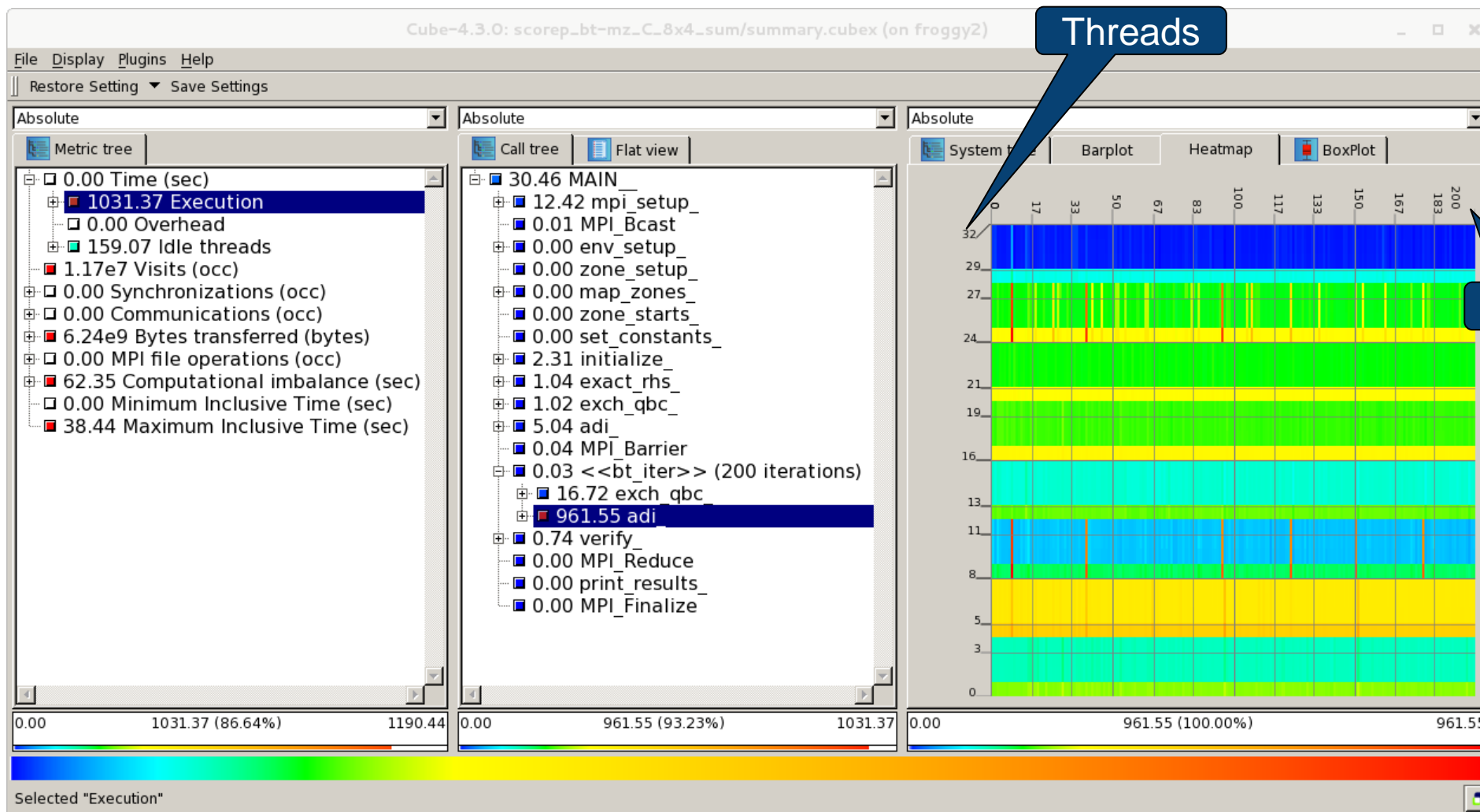
- Iterations shown as separate call trees
    - Useful for checking results for specific iterations
  - or
  - Select your user-instrumented region and mark it as loop
  - Choose “Hide iterations”
    - View the Barplot statistics or the (thread x iterations) Heatmap



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

