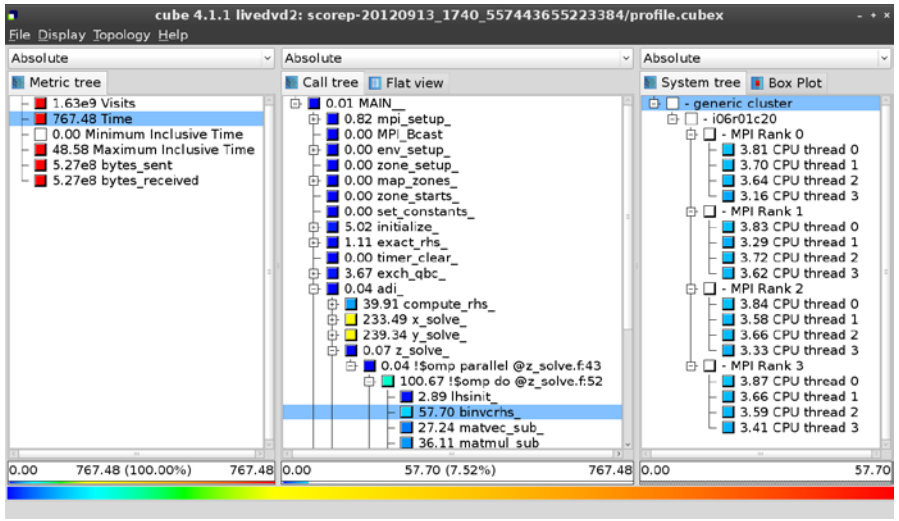
Analysis report examination with Cube

David Böhme
Lawrence Livermore National Lab

cube scalasca

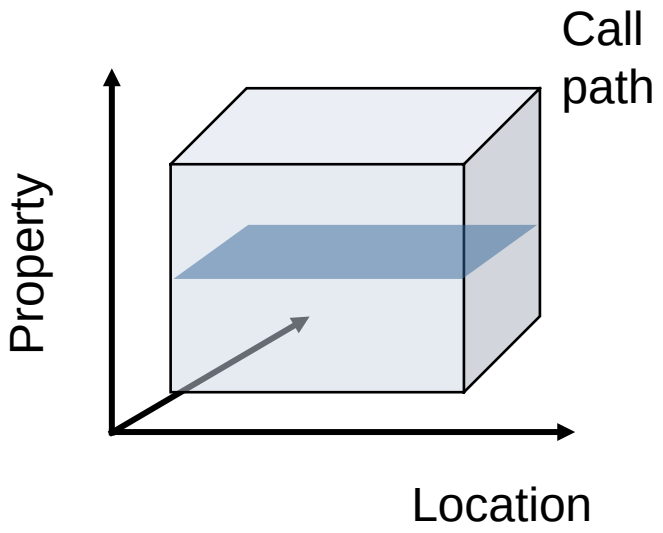
Cube

- 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)

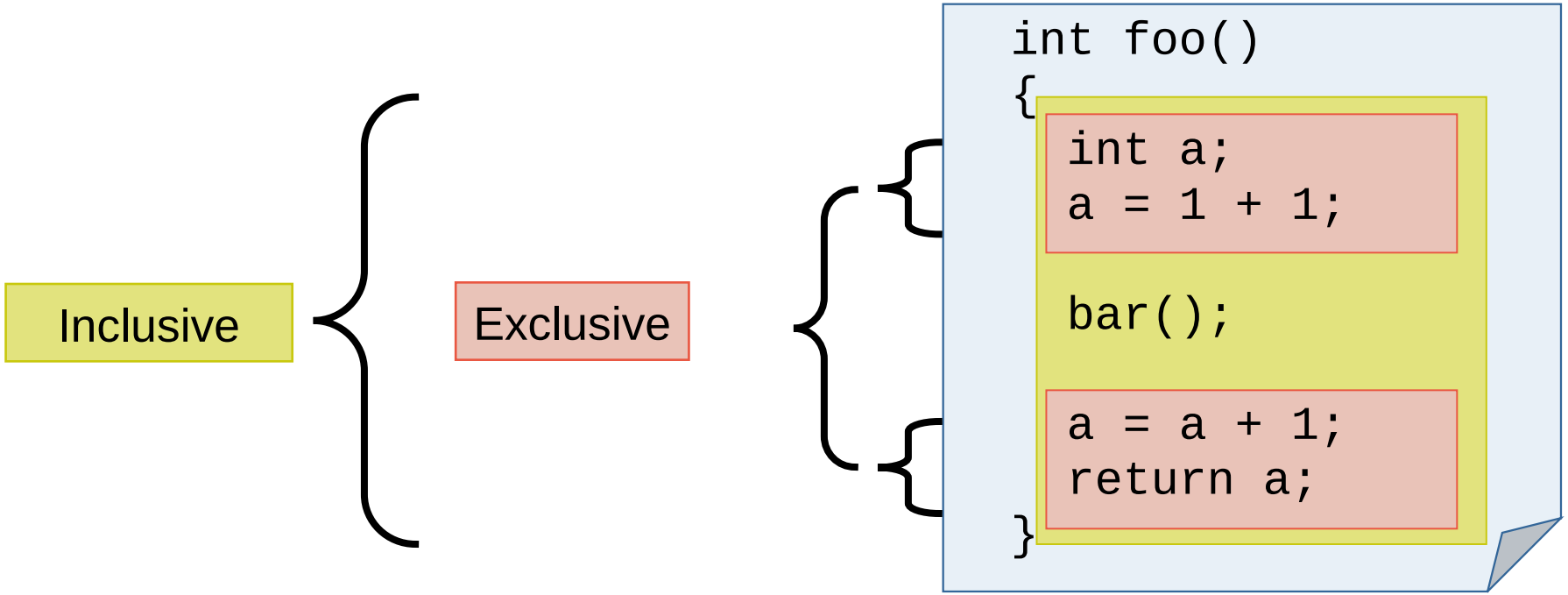


- Three coupled tree browsers

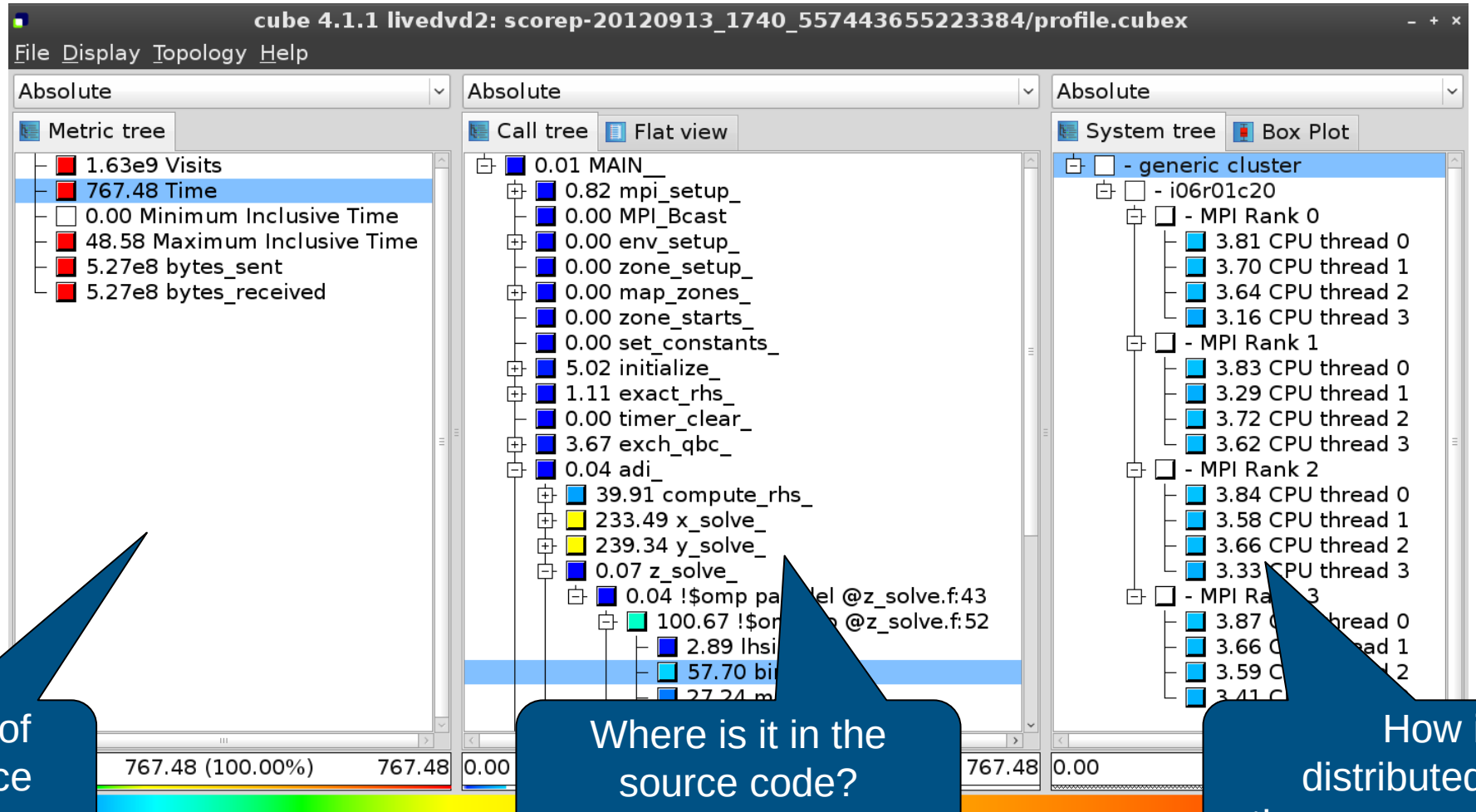
- Cube displays severities
  - As value: for precise comparison
  - As color: for easy identification of hotspots
  - Inclusive value when closed & exclusive value when expanded
  - Customizable via display modes

Inclusive vs. exclusive values

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



Analysis presentation

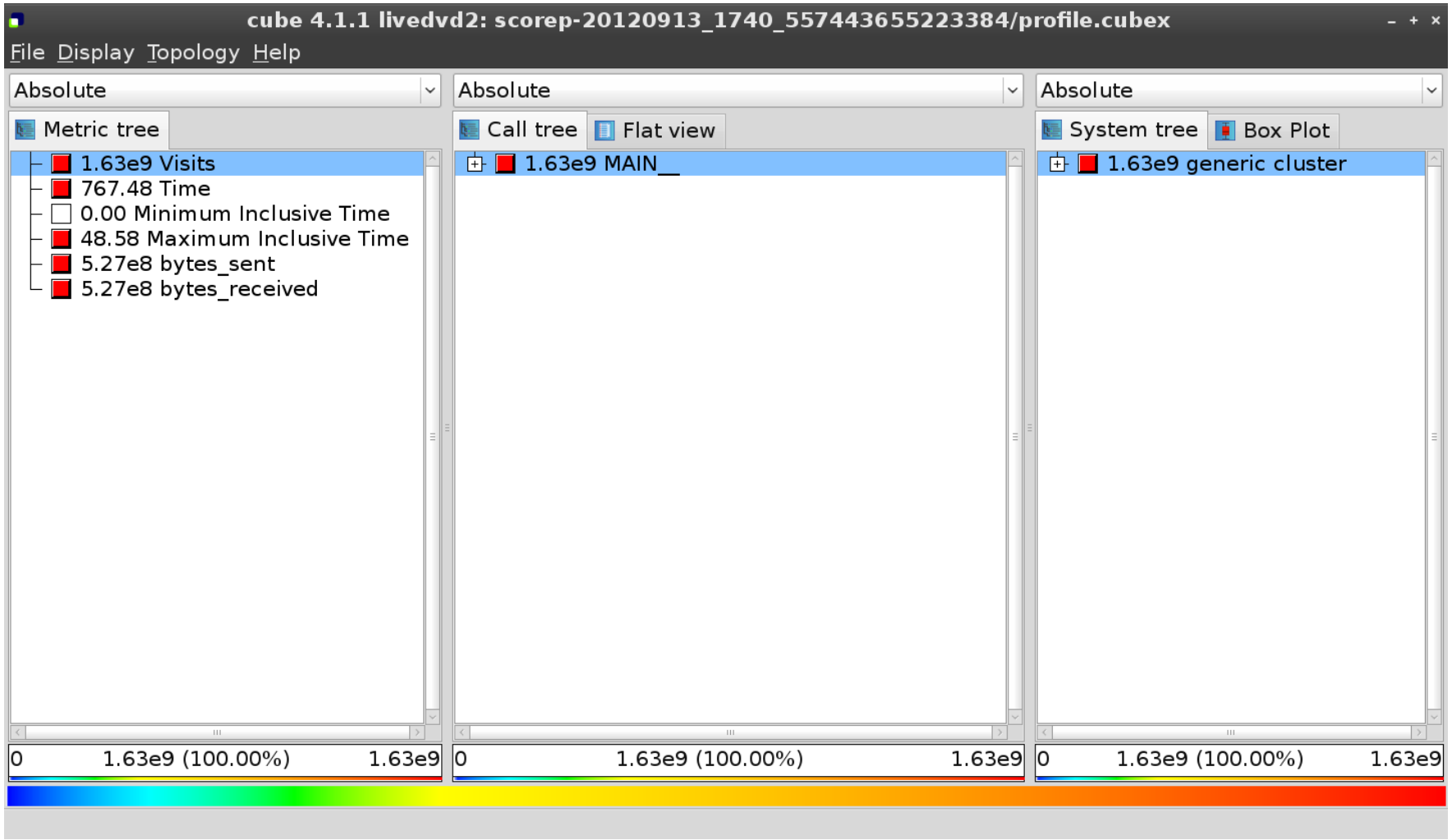


What kind of performance metric?

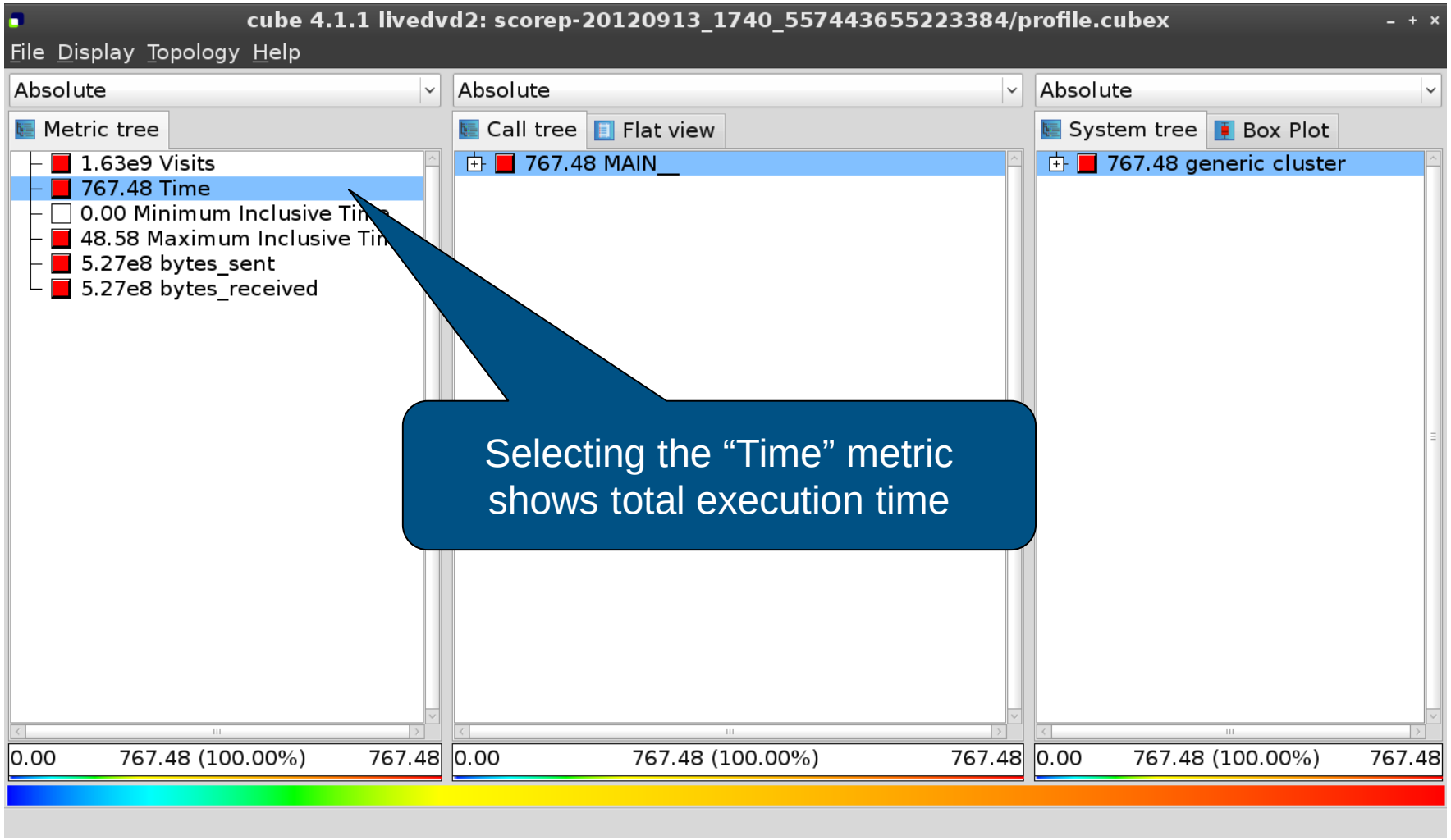
Where is it in the source code? In what context?

How is it distributed across the processes/threads?

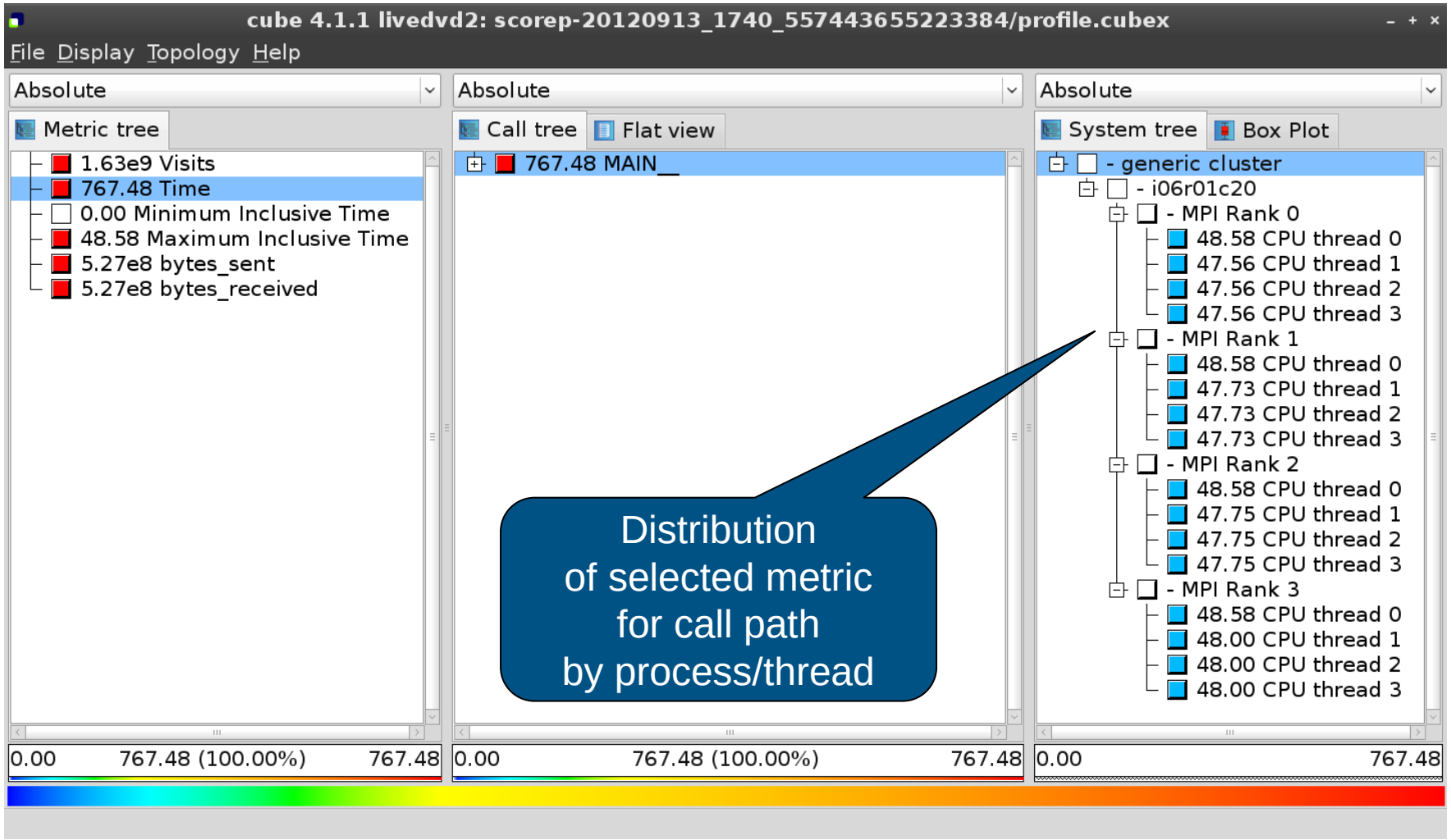
Score-P analysis report exploration (opening view)



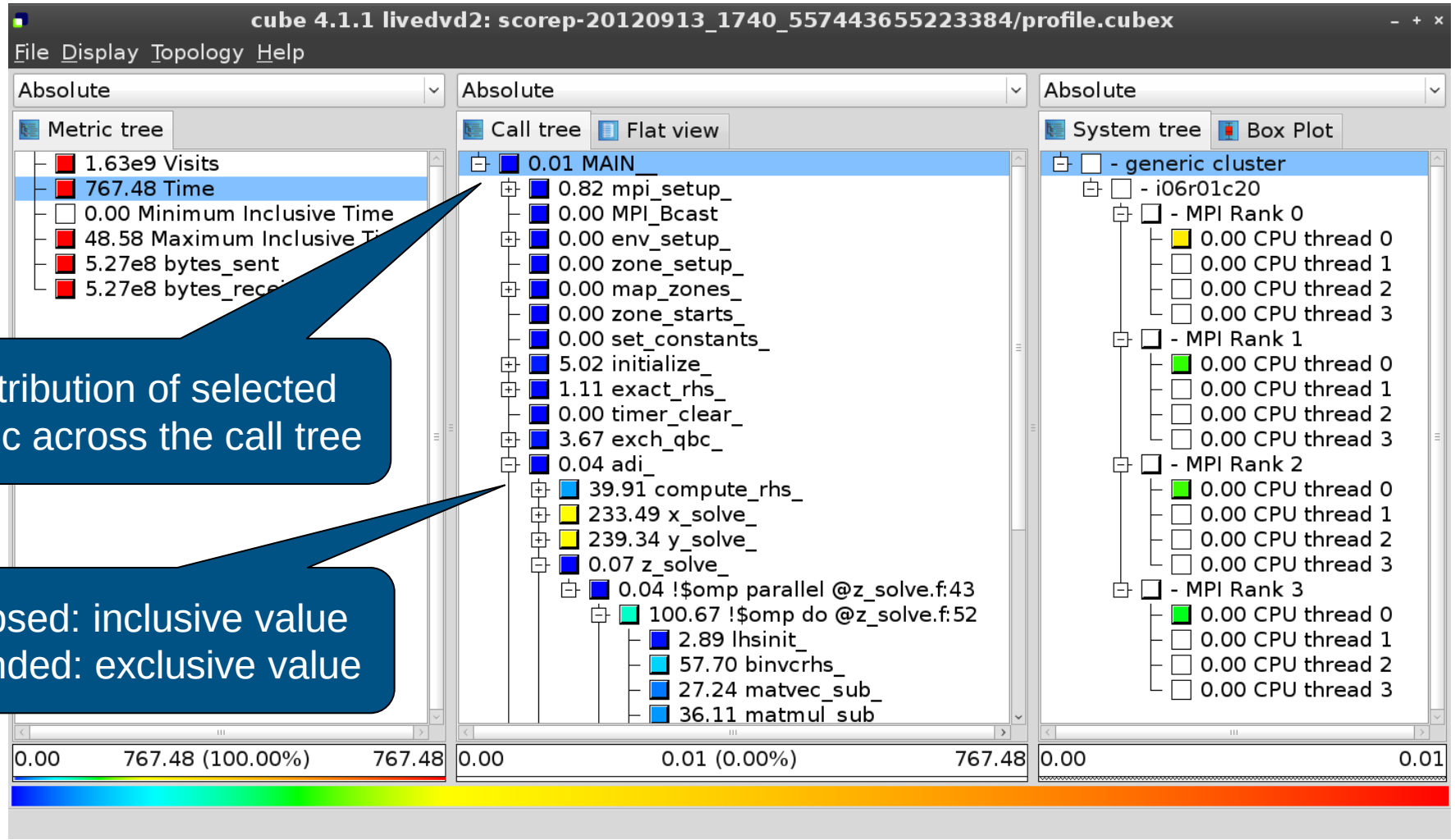
Metric selection



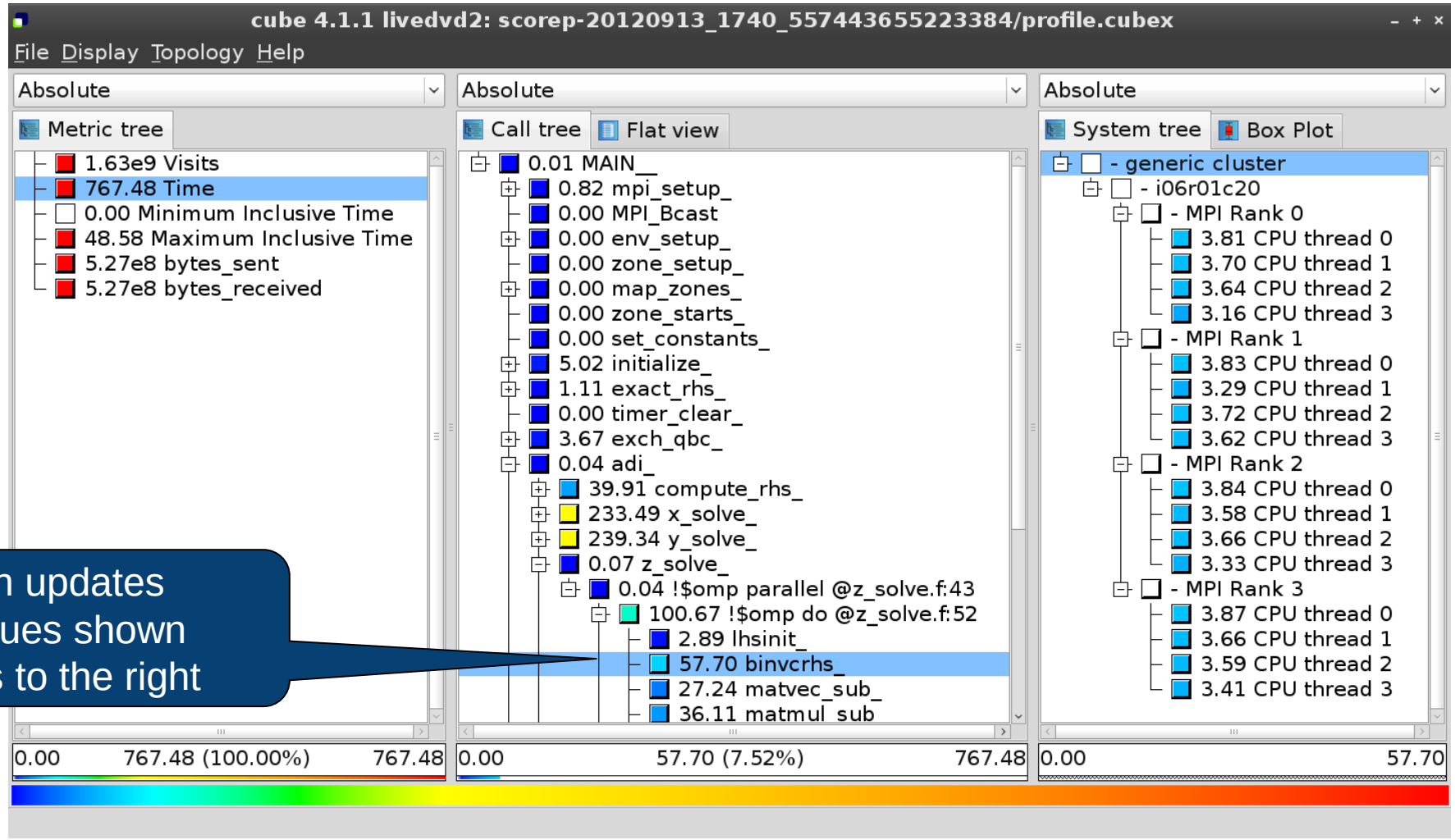
Expanding the system tree



Expanding the call tree

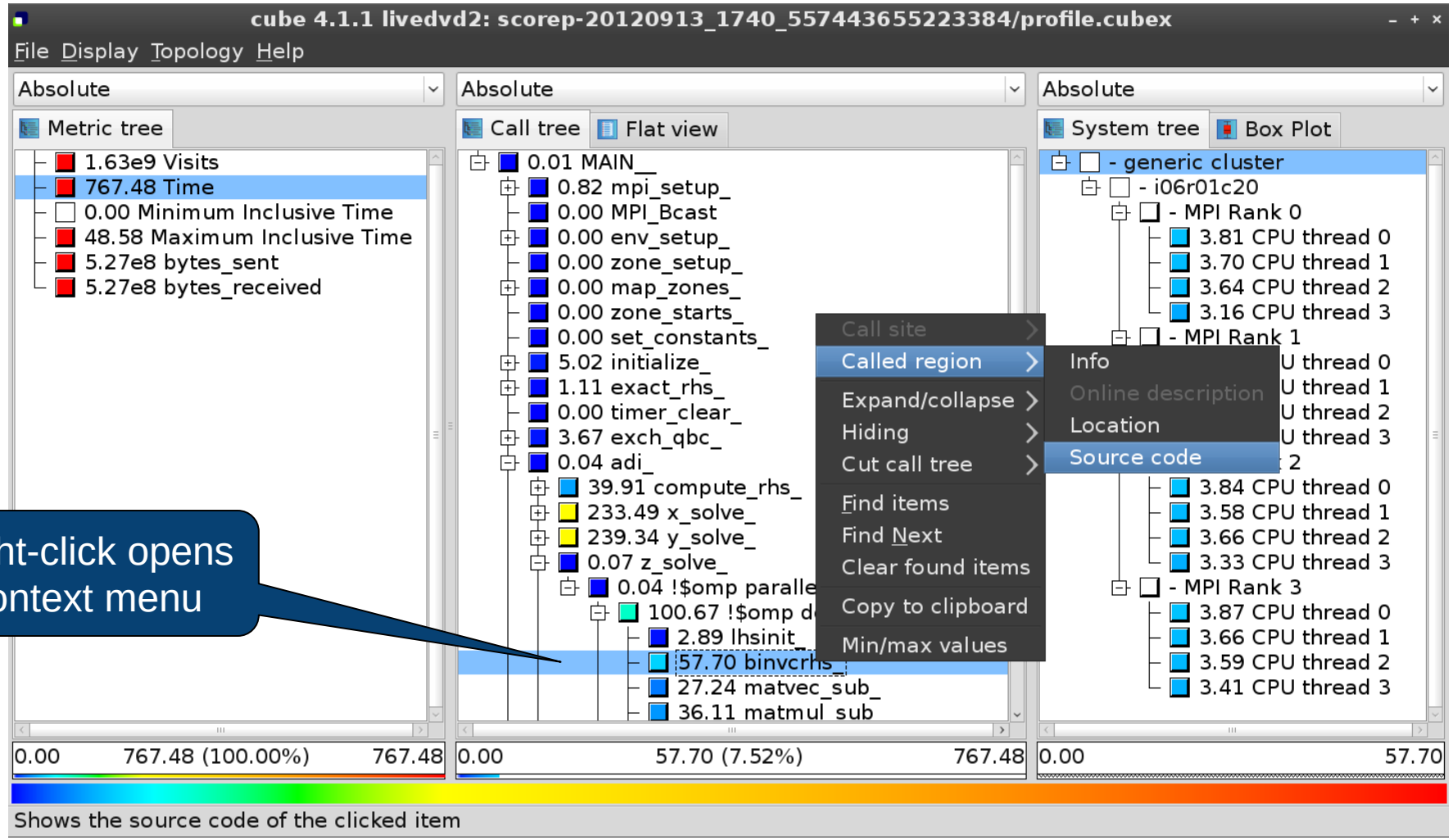


Selecting a call path



Selection updates metric values shown in columns to the right

Source-code view via context menu



Right-click opens context menu

Source-code view

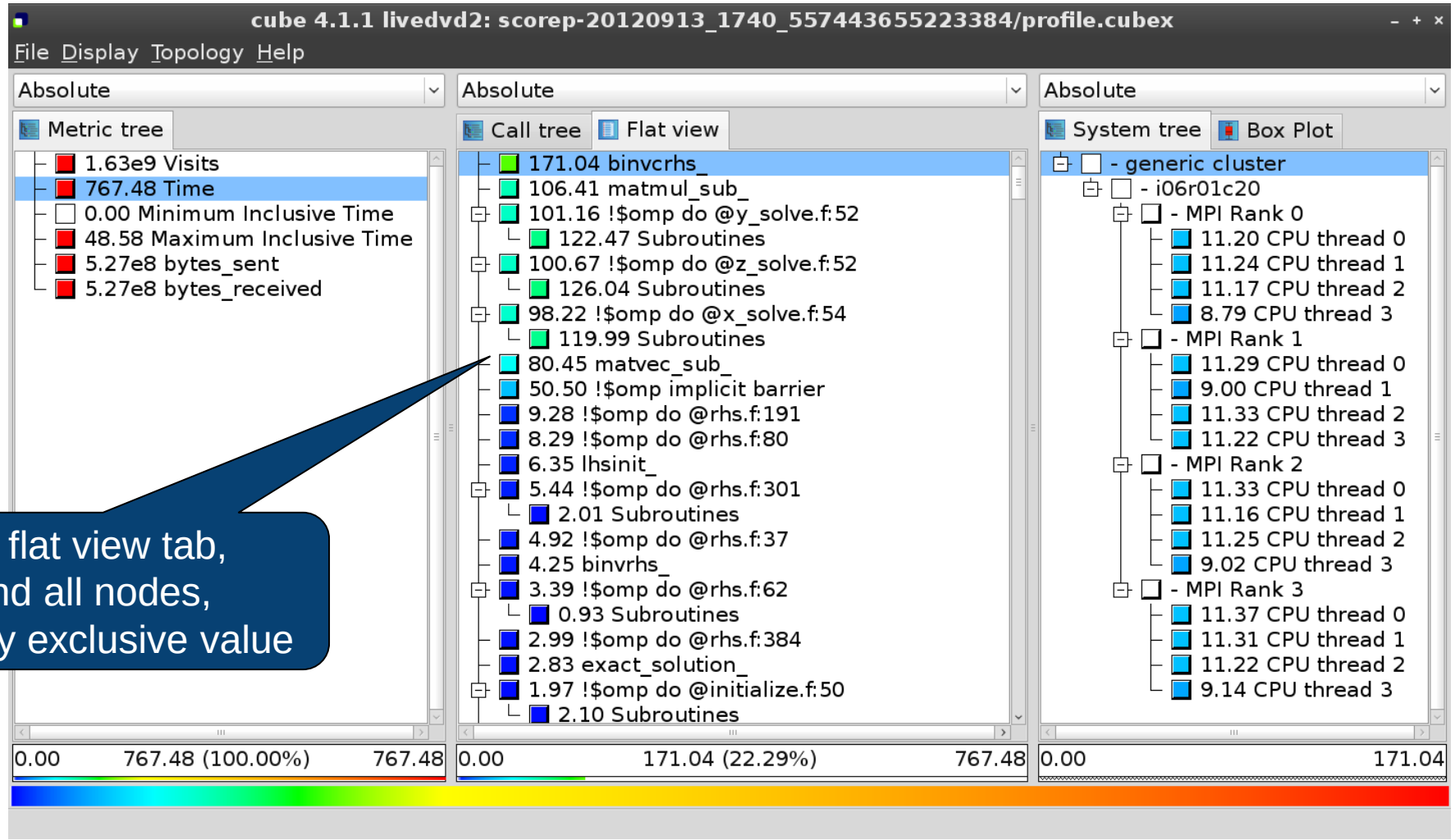
/home/geimer/Projects/Tests/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f

```
      subroutine binvcrhs( 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

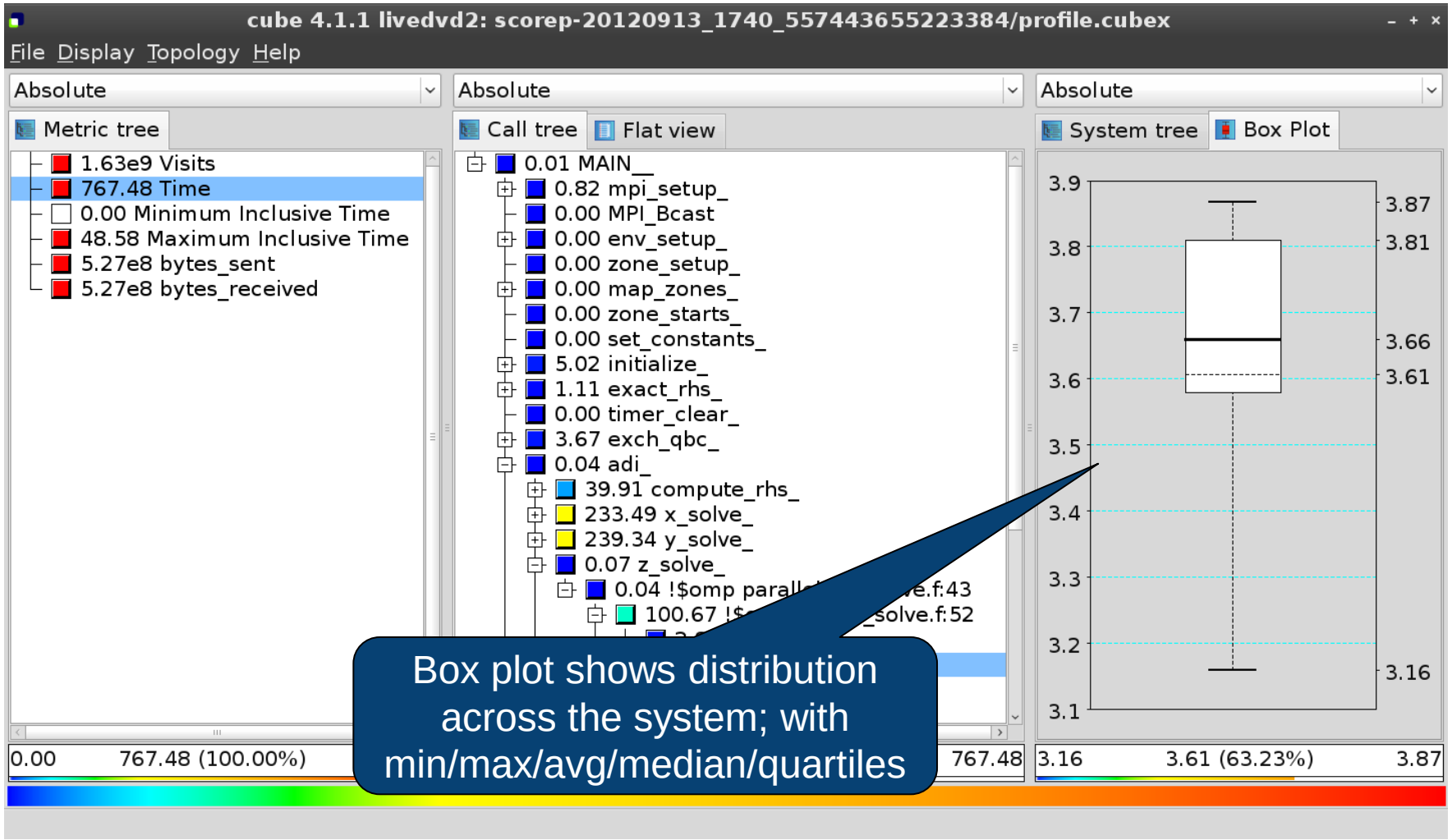
Note:
This feature depends on file and line number information provided by the instrumentation, i.e., it may not always be available

Flat profile view

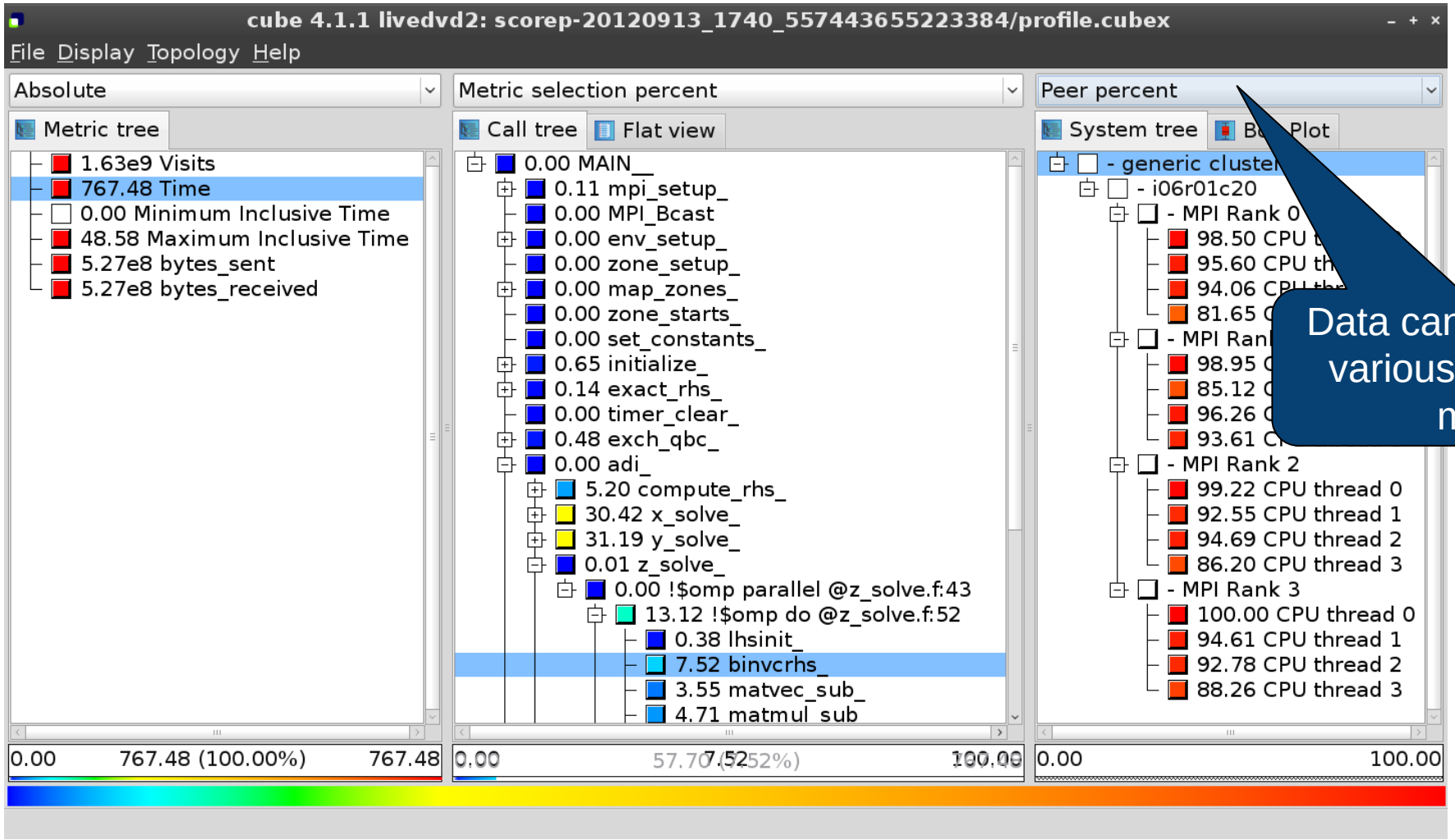


Select flat view tab,
expand all nodes,
and sort by exclusive value

Box plot view



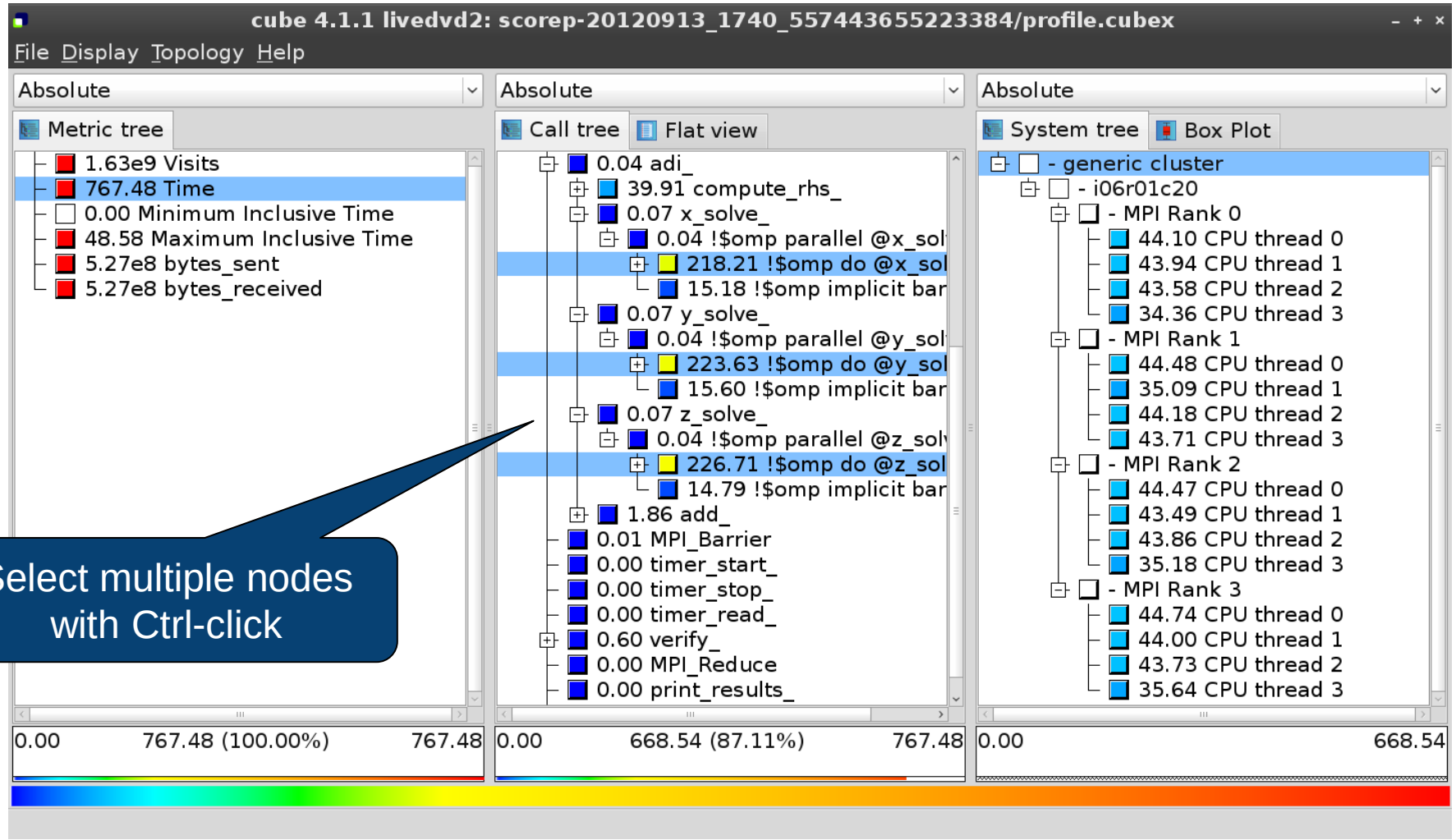
Alternative display modes



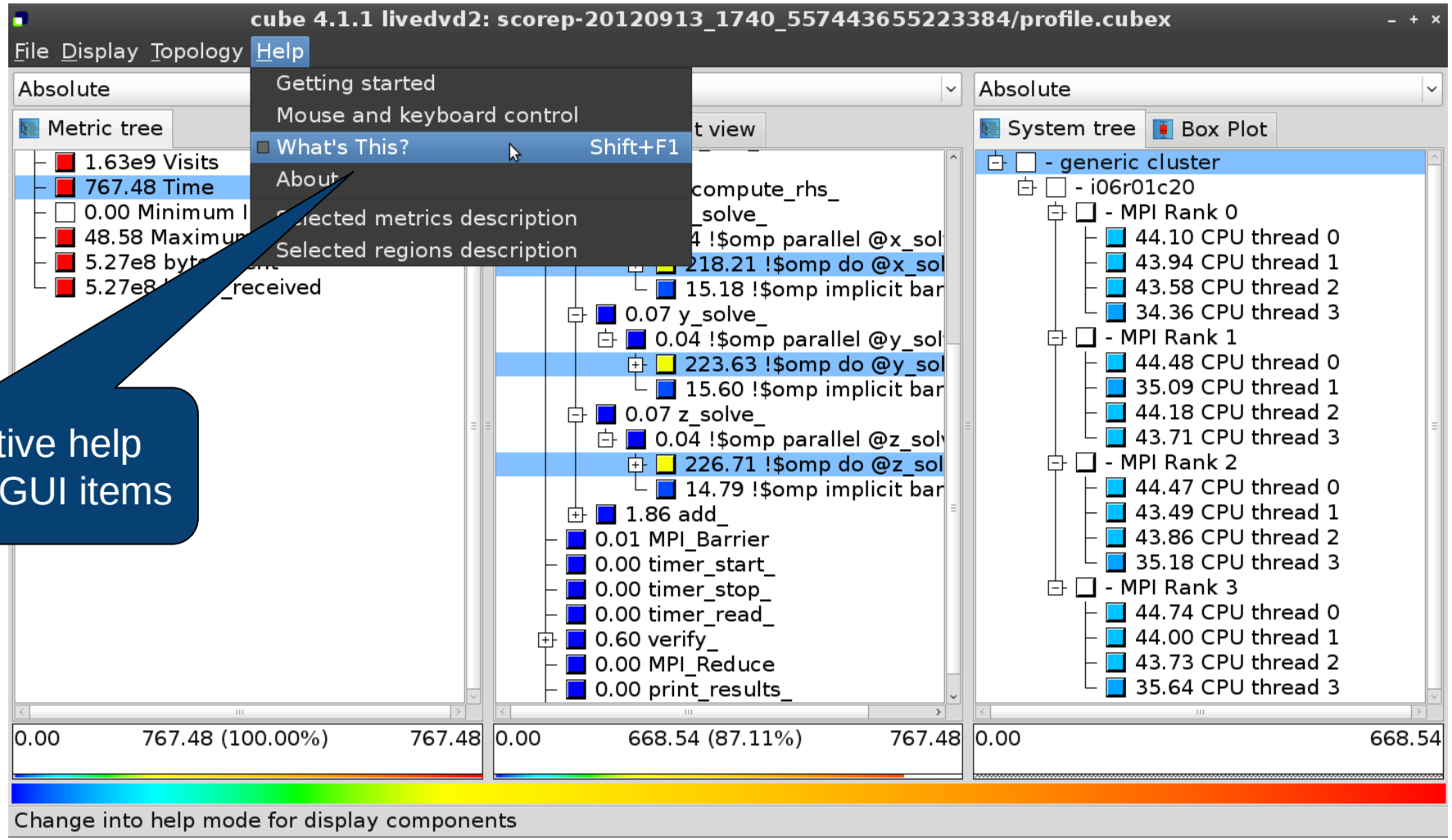
Important display modes

- 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



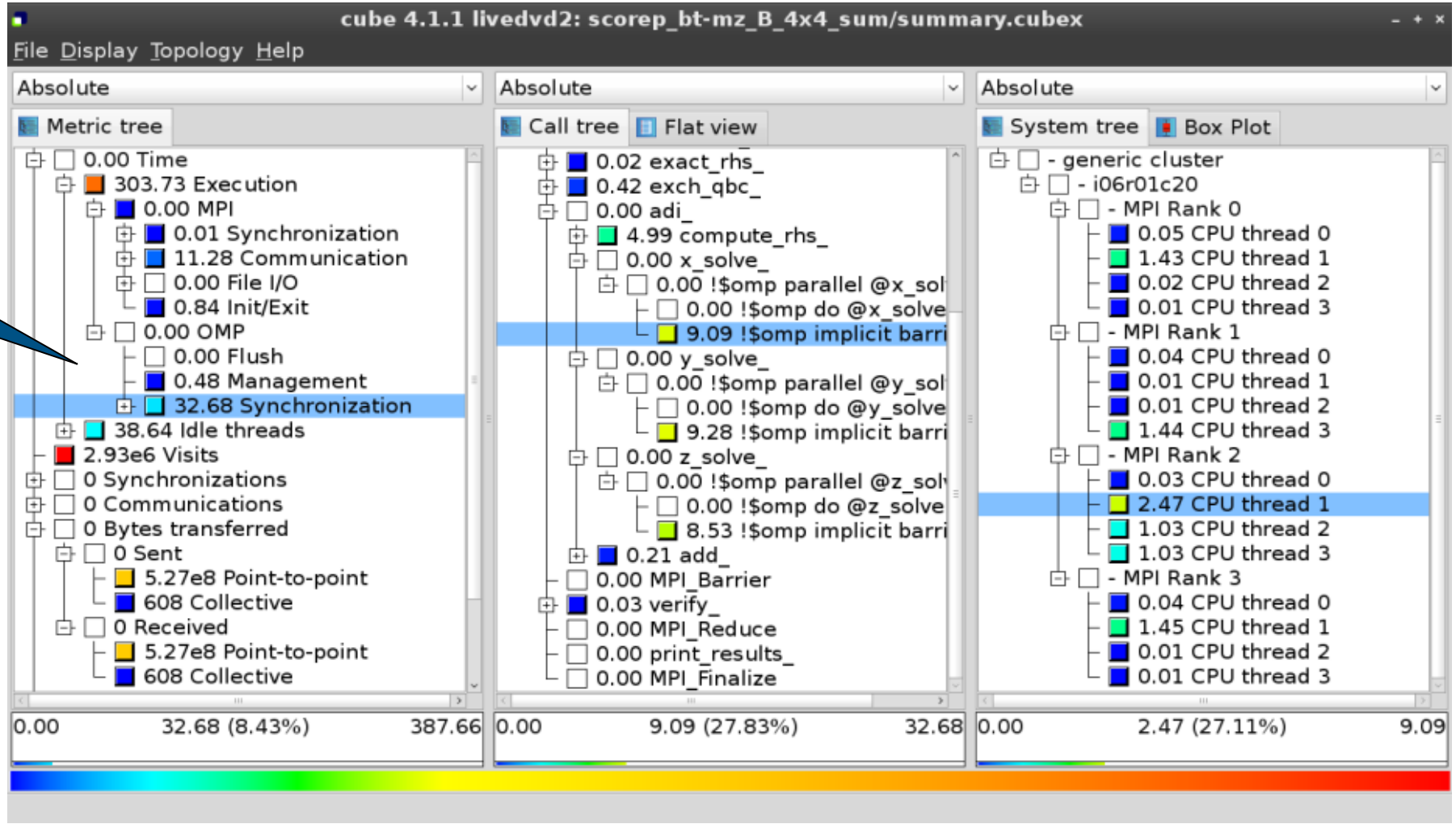
Context-sensitive help



Context-sensitive help available for all GUI items

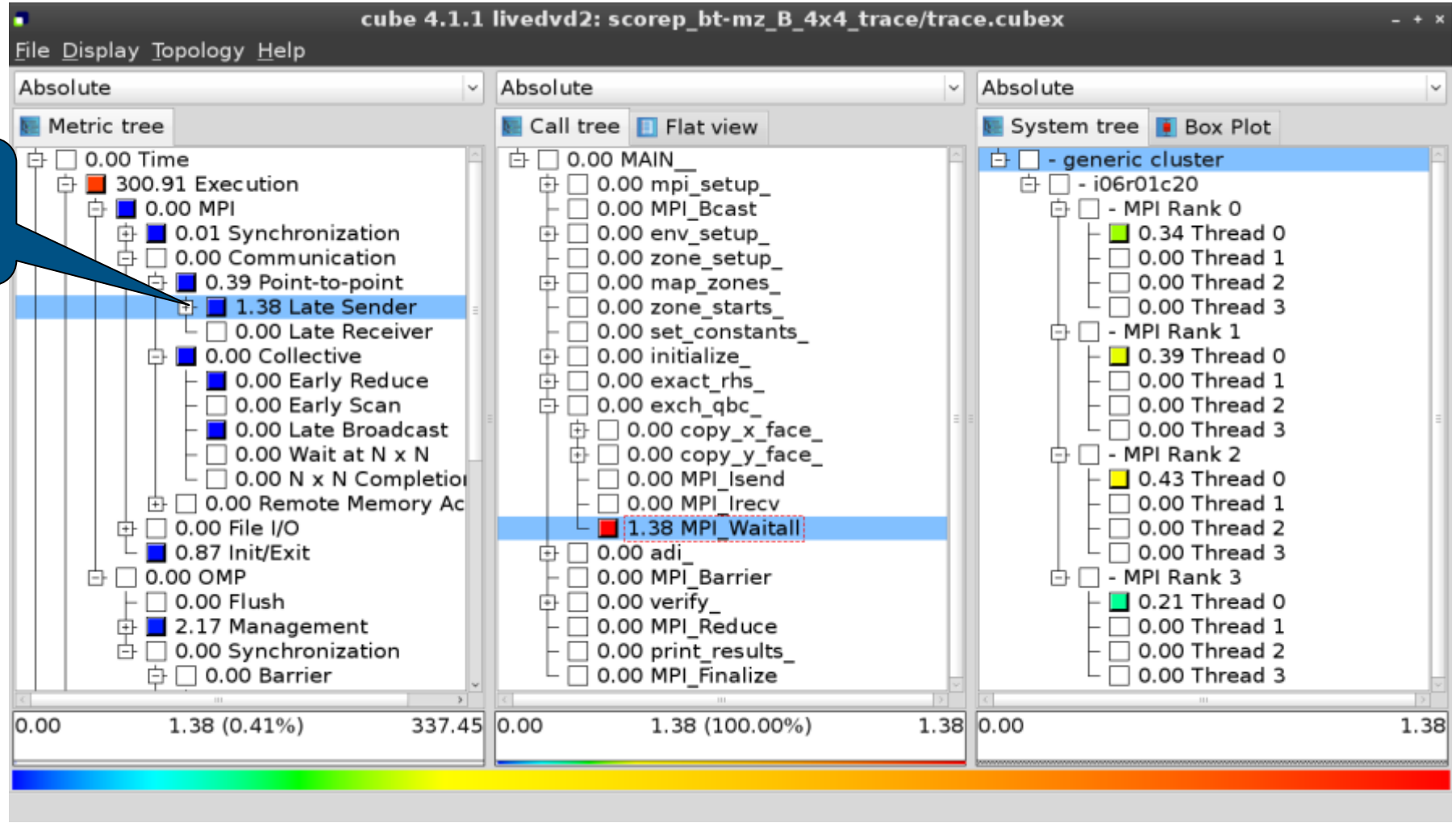
Scalasca's post-processed summary analysis report

Split base metrics into more specific metrics



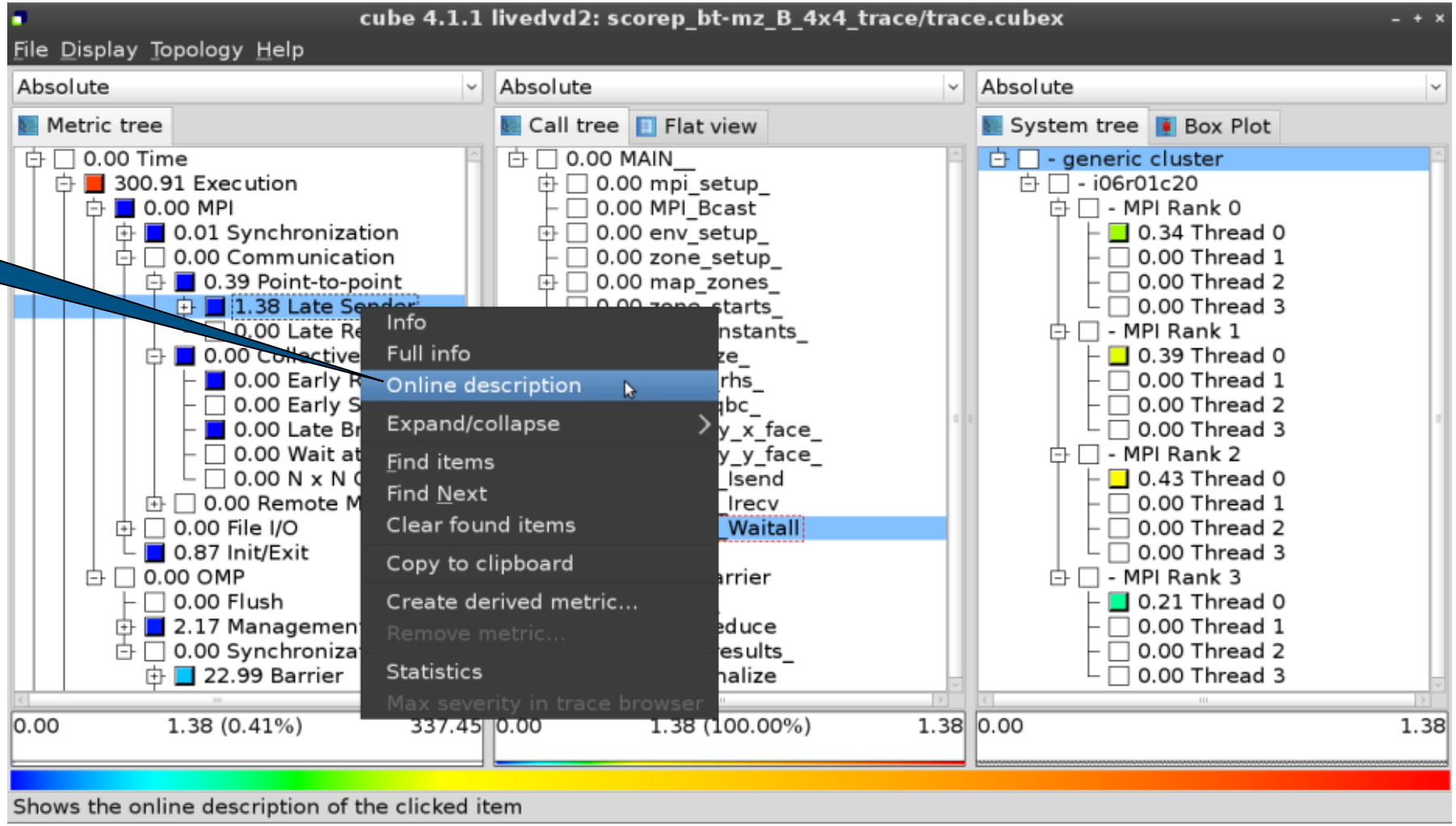
Scalasca's post-processed trace analysis report

Additional trace-based metrics in metric hierarchy



Online metric description

Access online metric description via context menu



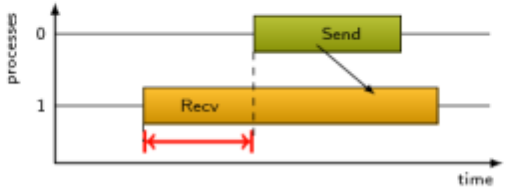
Online metric description

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:

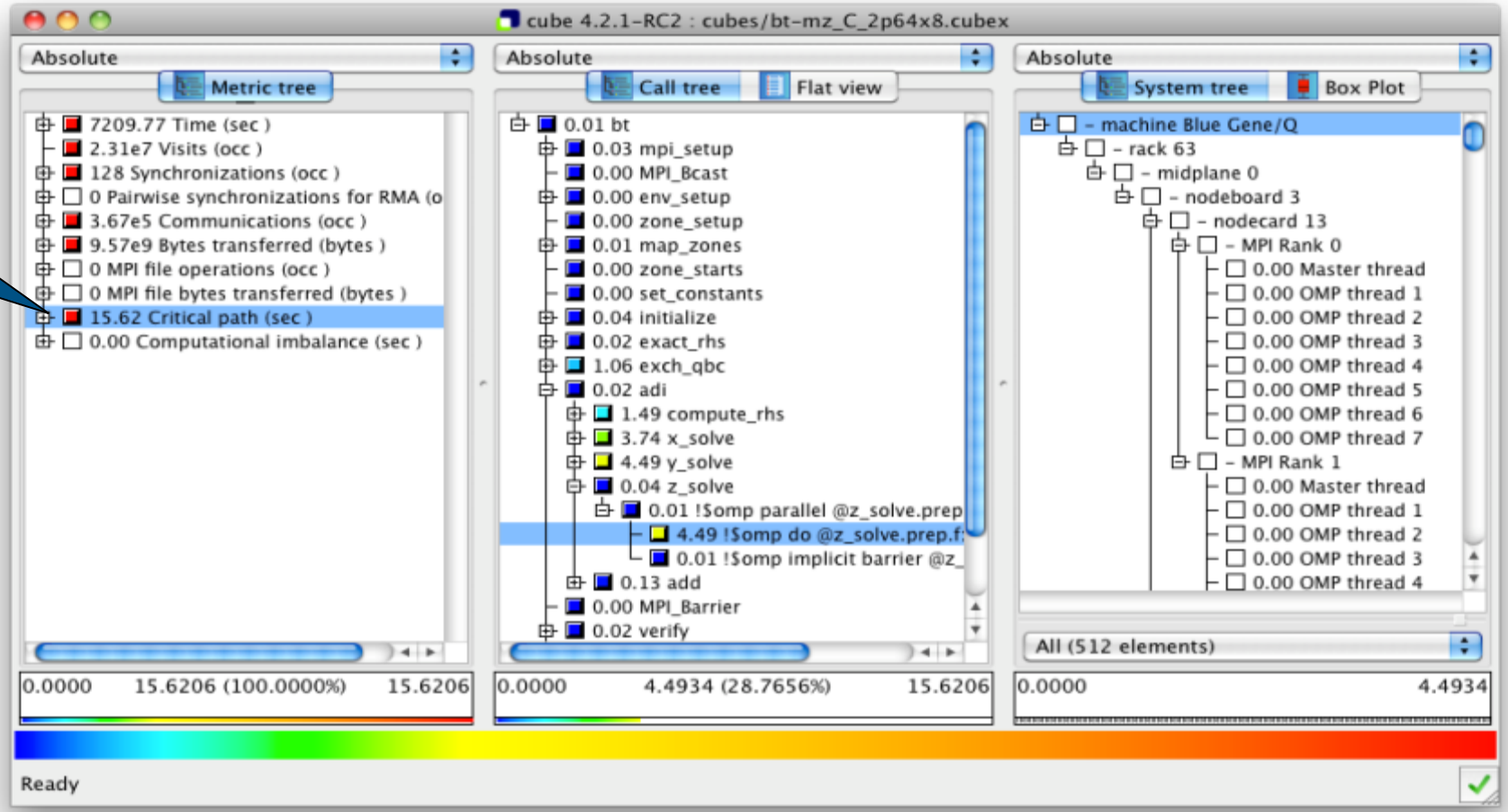
MPI Point-to-point Communication Time

Children:

Close

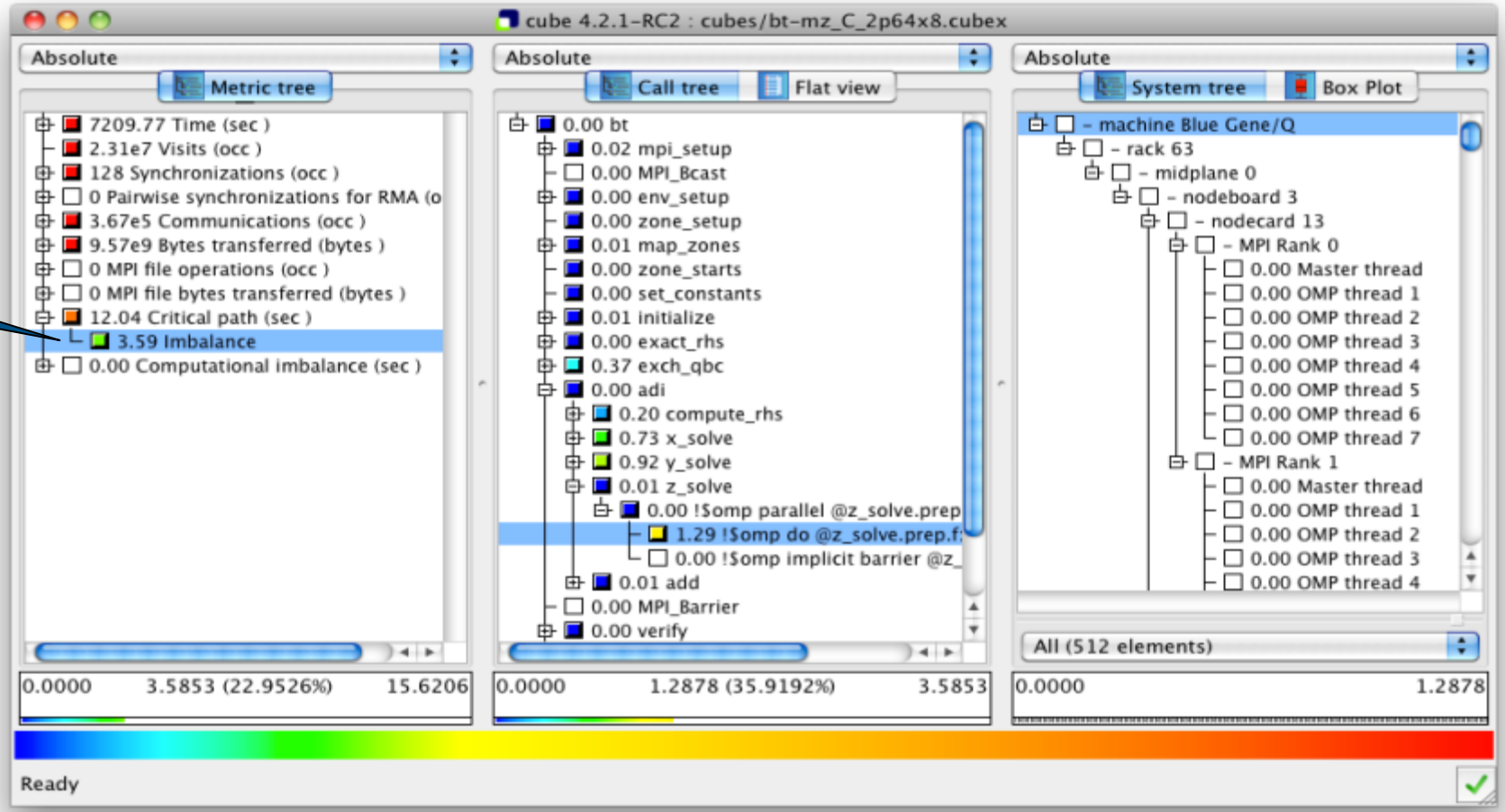
Scalasca critical-path analysis

Critical-path profile shows wall-clock time impact



Scalasca critical-path analysis

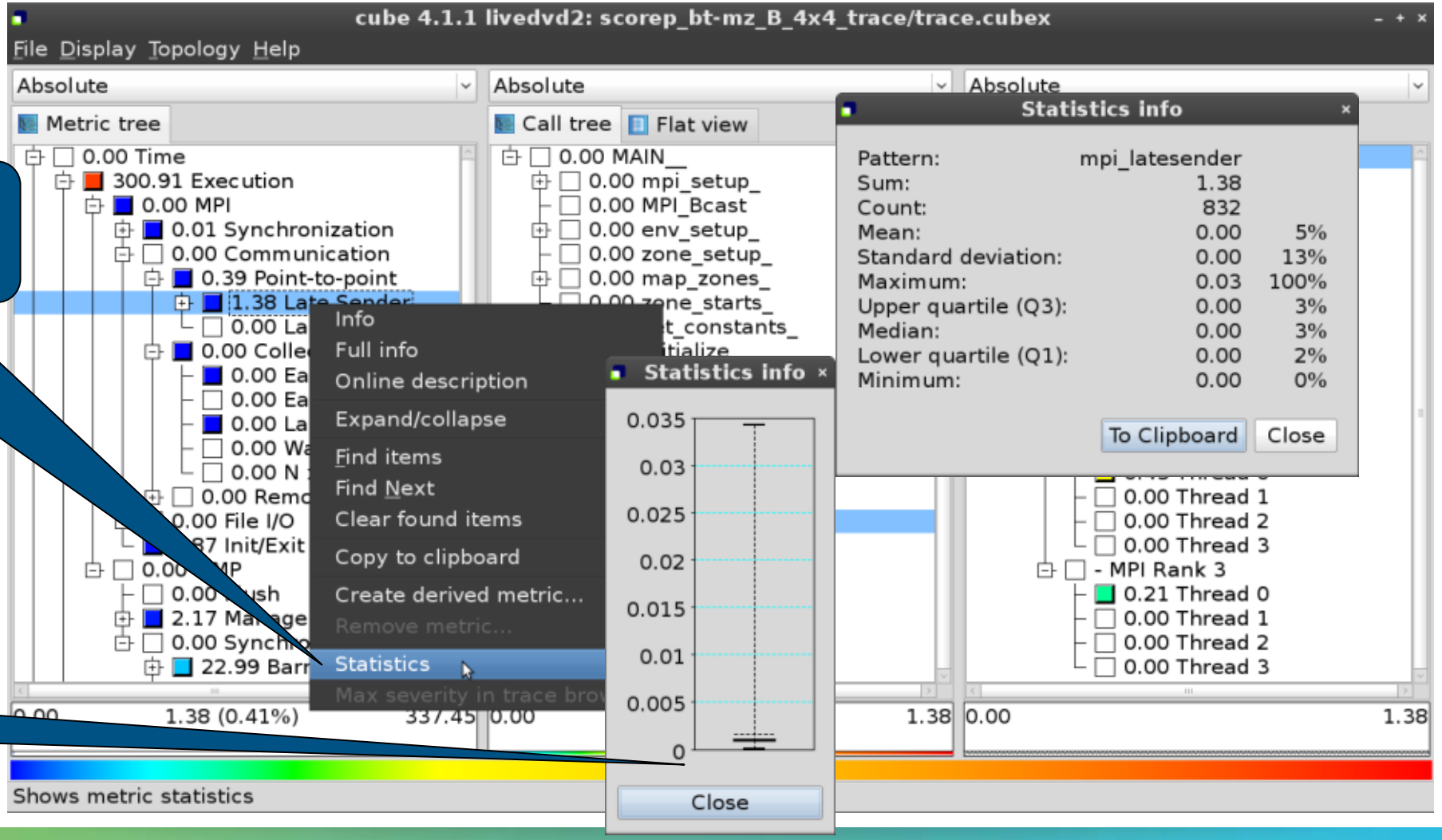
Critical-path imbalance highlights inefficient parallelism



Scalasca pattern instance statistics

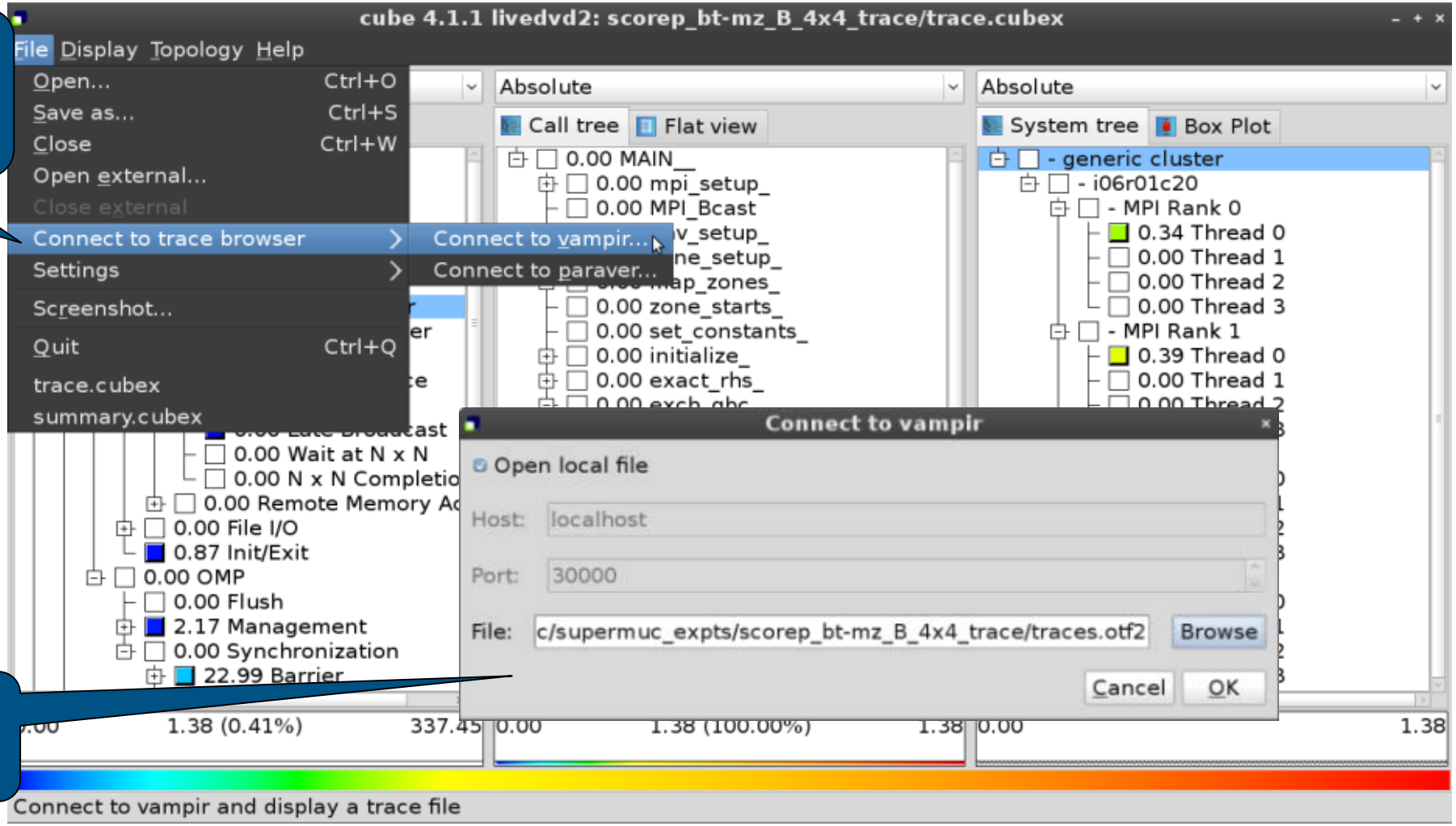
Access pattern instance statistics via context menu

Click to get statistics details



Connect to Vampir trace browser

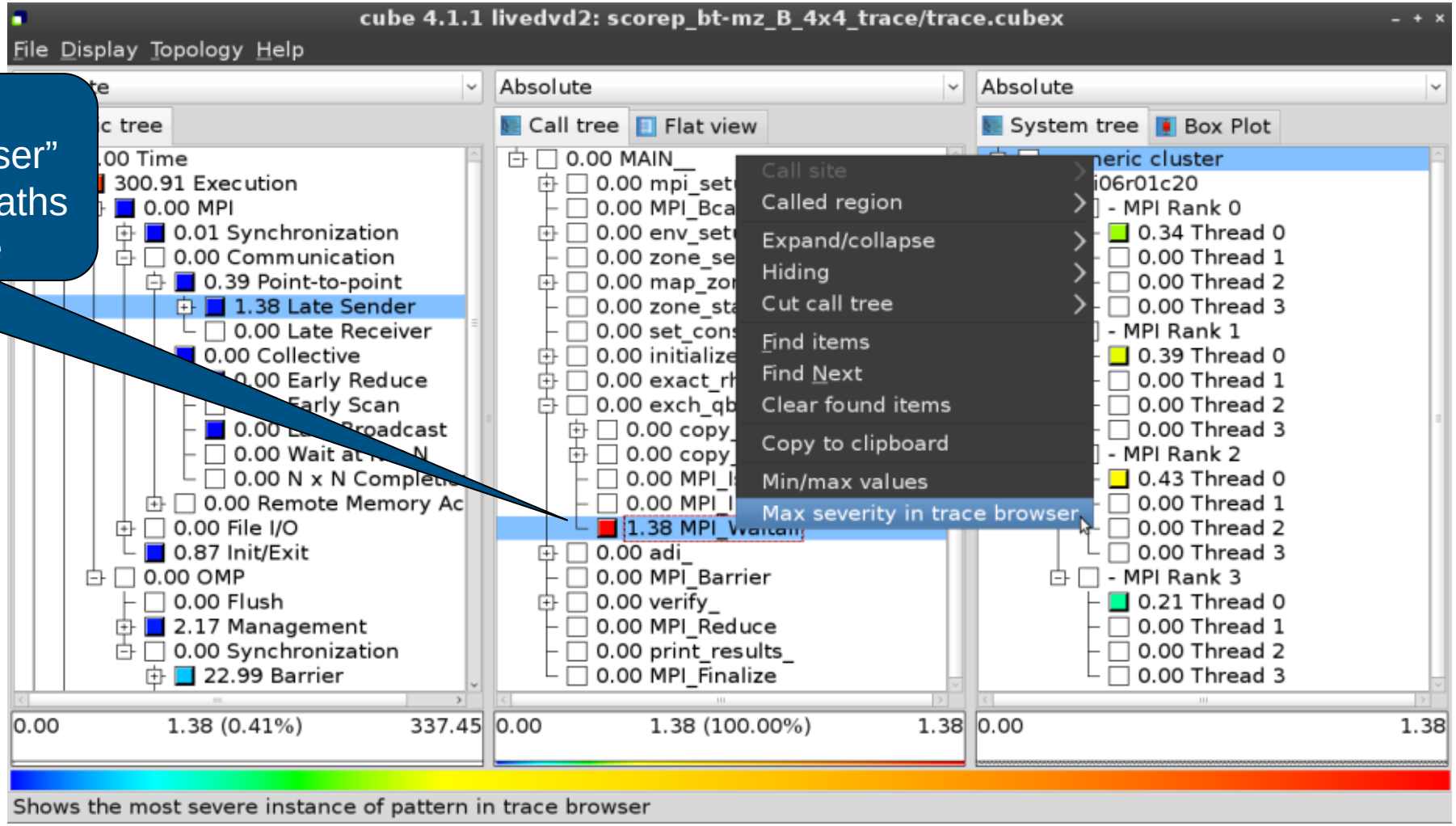
To investigate most severe pattern instances, connect to a trace browser…



…and select trace file from the experiment directory

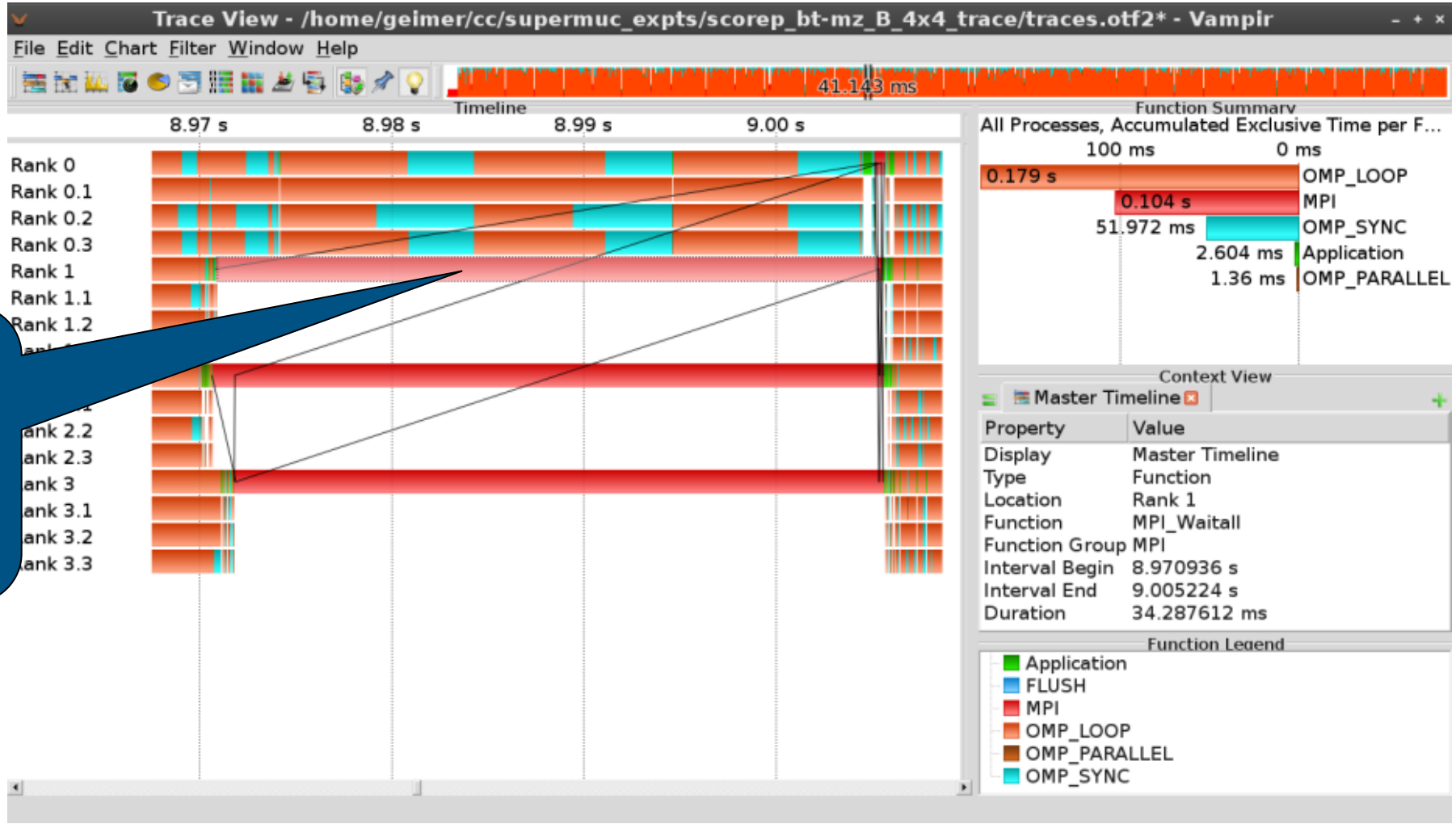
Show most severe pattern instances

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



Investigate most severe instance in Vampir

Vampir will automatically zoom to the worst instance in multiple steps (i.e., undo zoom provides more context)



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

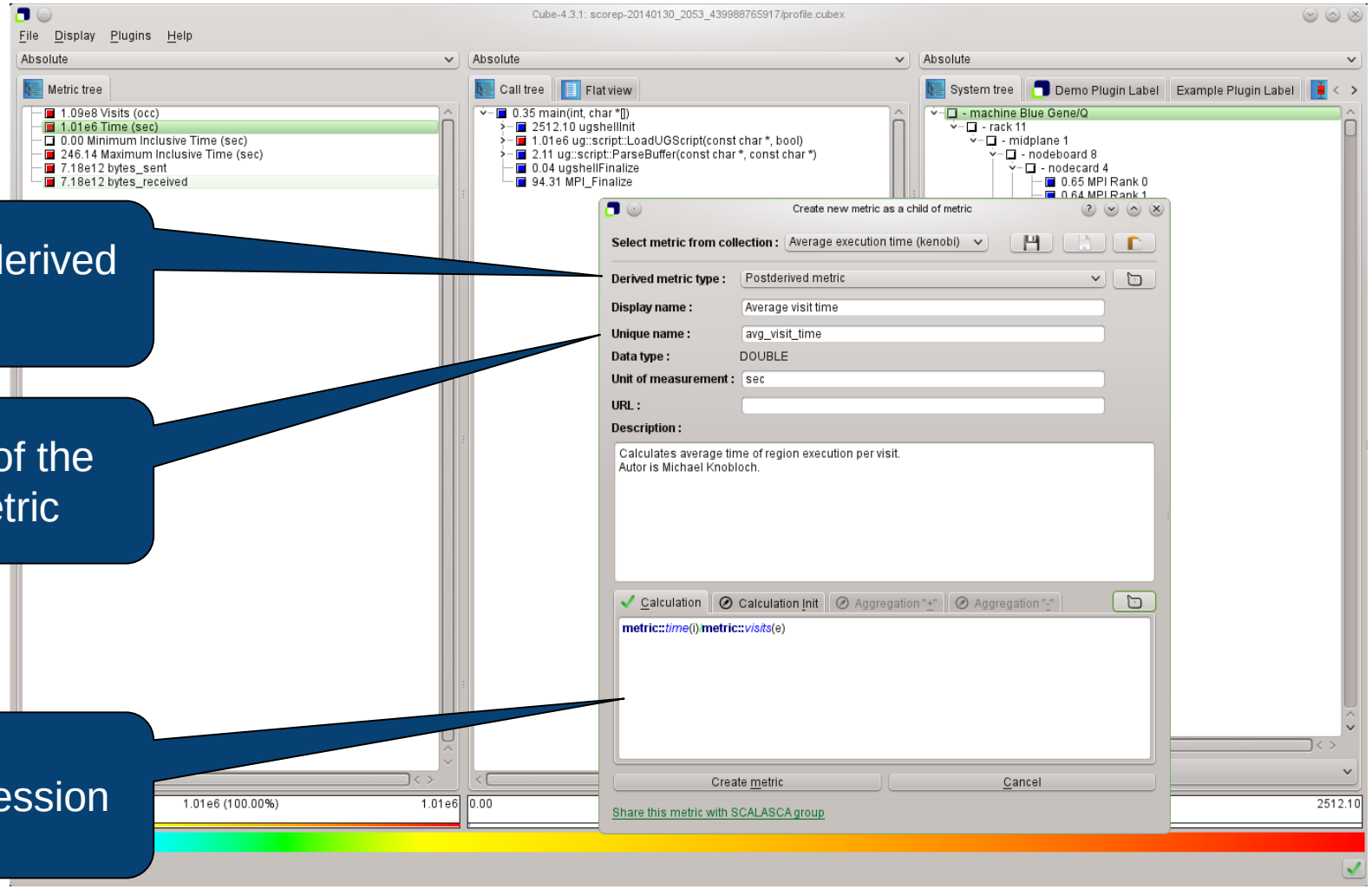
- Examples:
  - "Average execution time": Postderived metric with expression

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

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

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

Derived metrics in Cube GUI

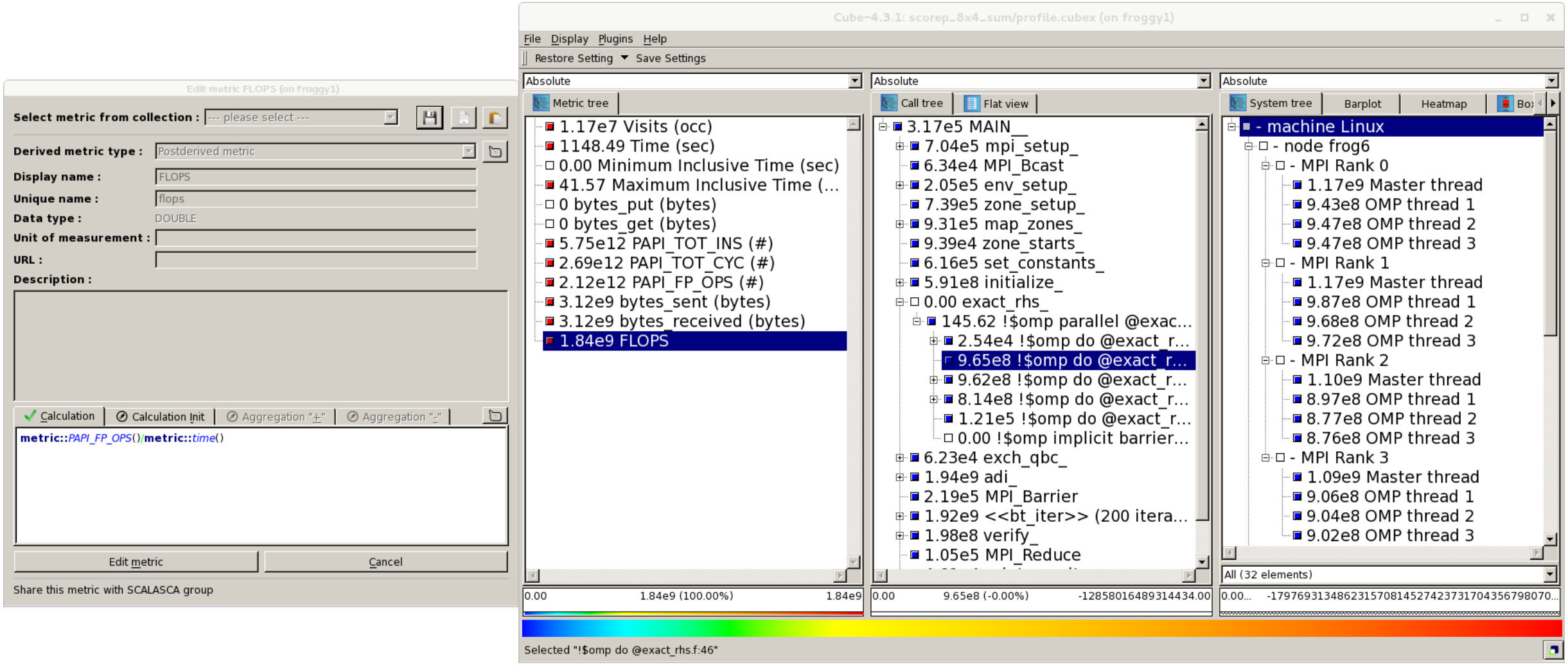


Collection of derived metrics

Parameters of the derived metric

CubePL expression

Example: FLOPS based on PAPI_FP_OPS and time



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

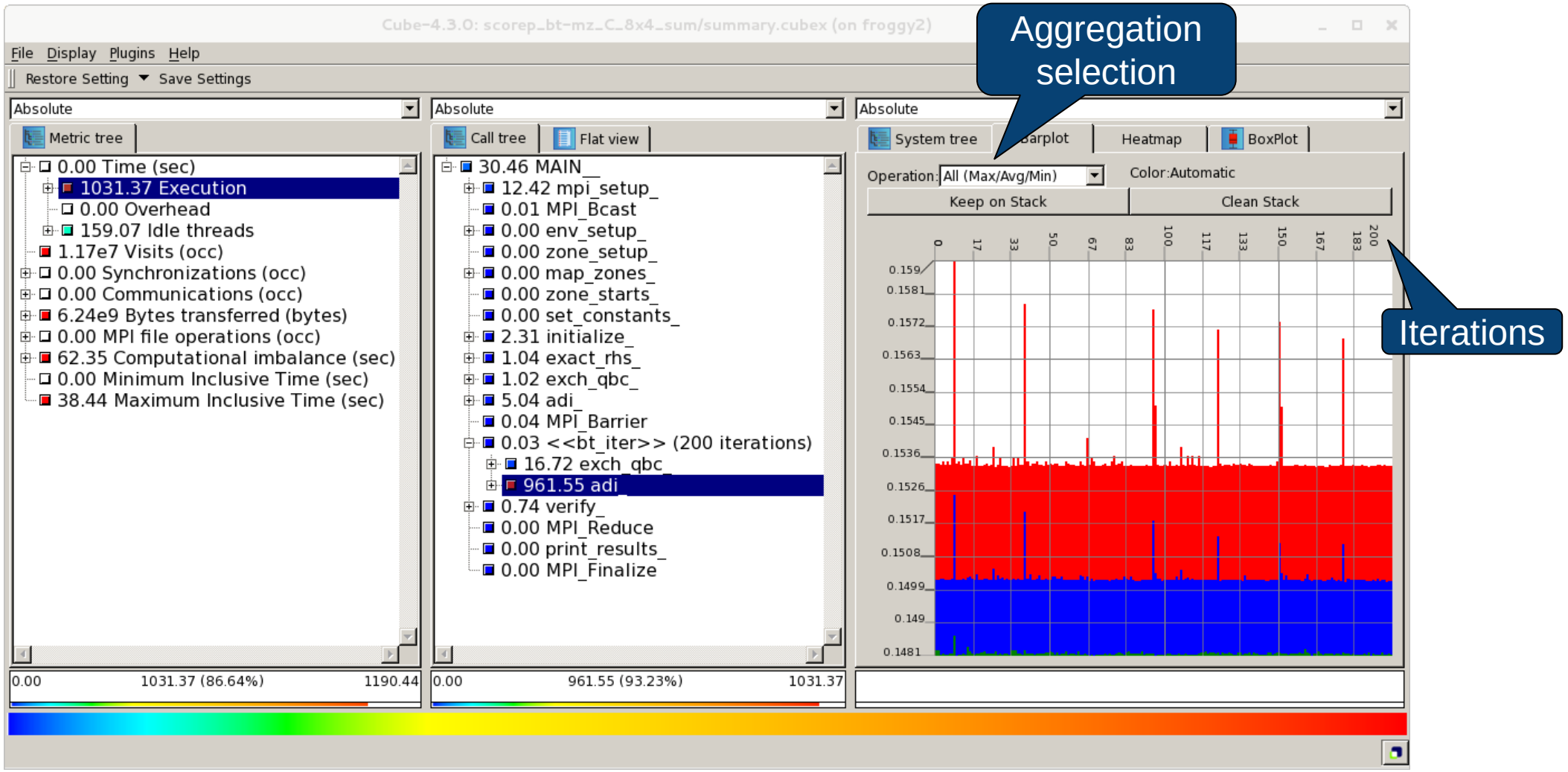
```
SCOREP_USER_REGION_DEFINE( scorep_bt_loop )
SCOREP_USER_REGION_BEGIN( scorep_bt_loop, "<<bt_iter>>", SCOREP_USER_REGION_TYPE_DYNAMIC )
SCOREP_USER_REGION_END( scorep_bt_loop )
```

- Result in the Cube profile:
  - Iterations shown as separate call trees
  - ➢ Useful for checking results for specific iterations

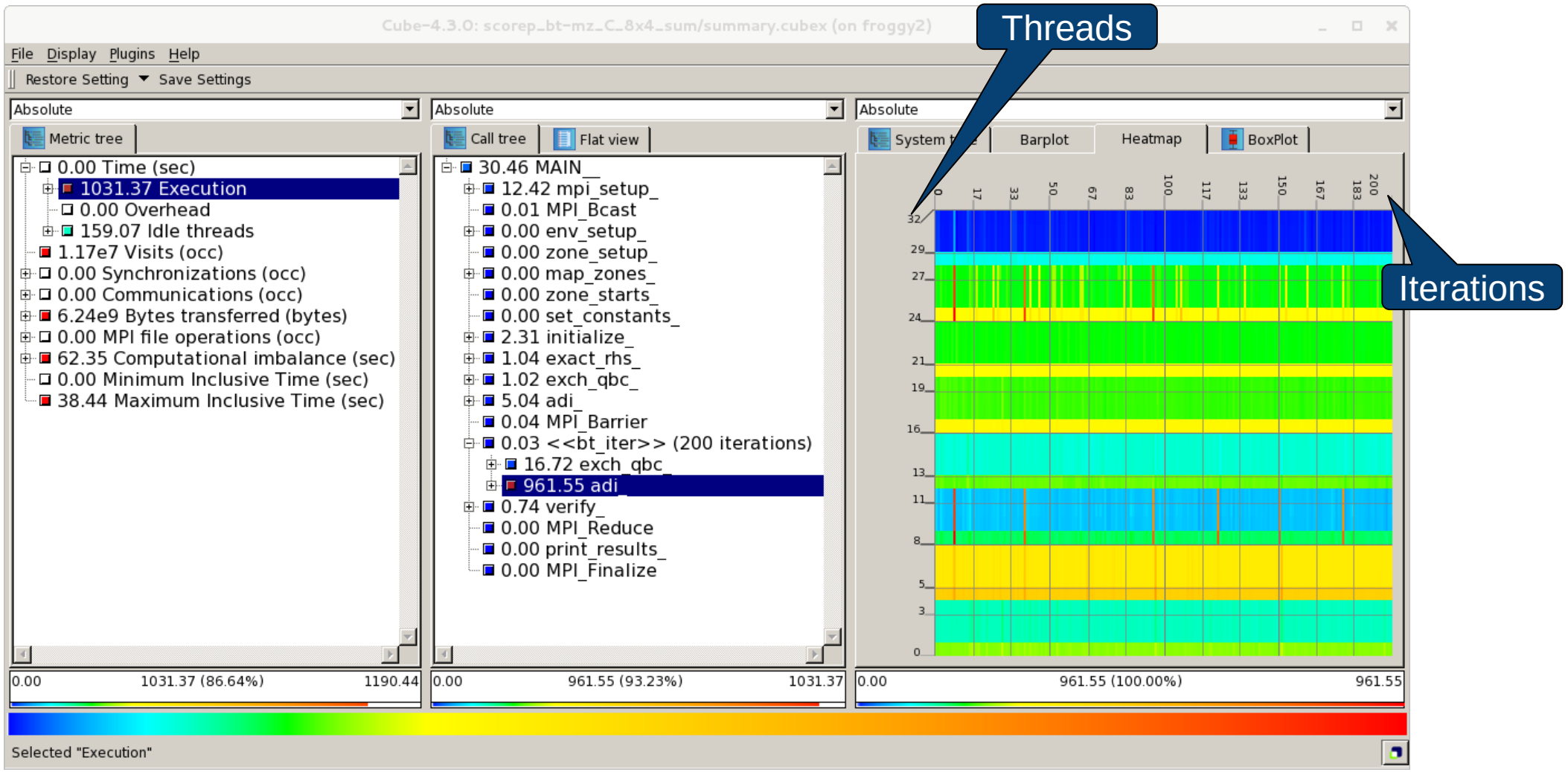
or

  - 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