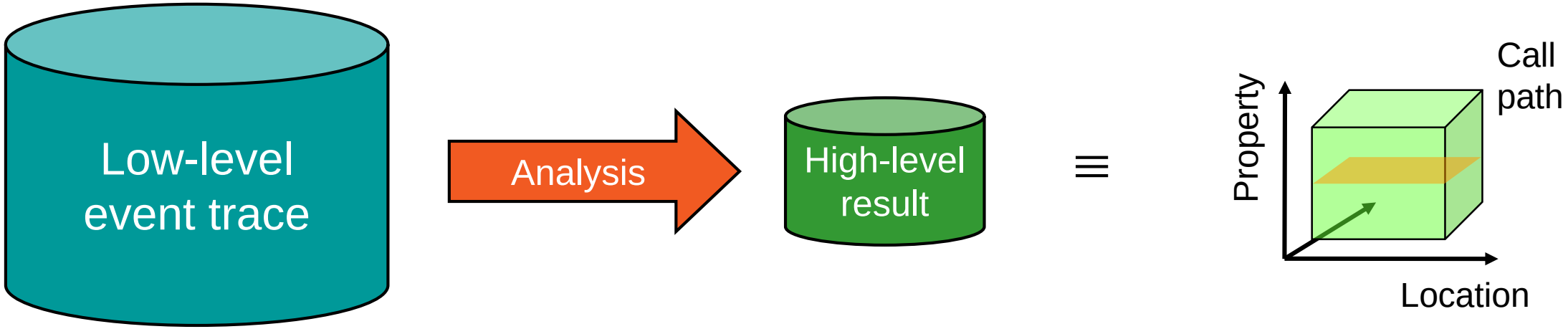
Automatic trace analysis with the Scalasca Trace Tools

Marc-André Hermanns
RWTH Aachen University

trace tools
scalasca

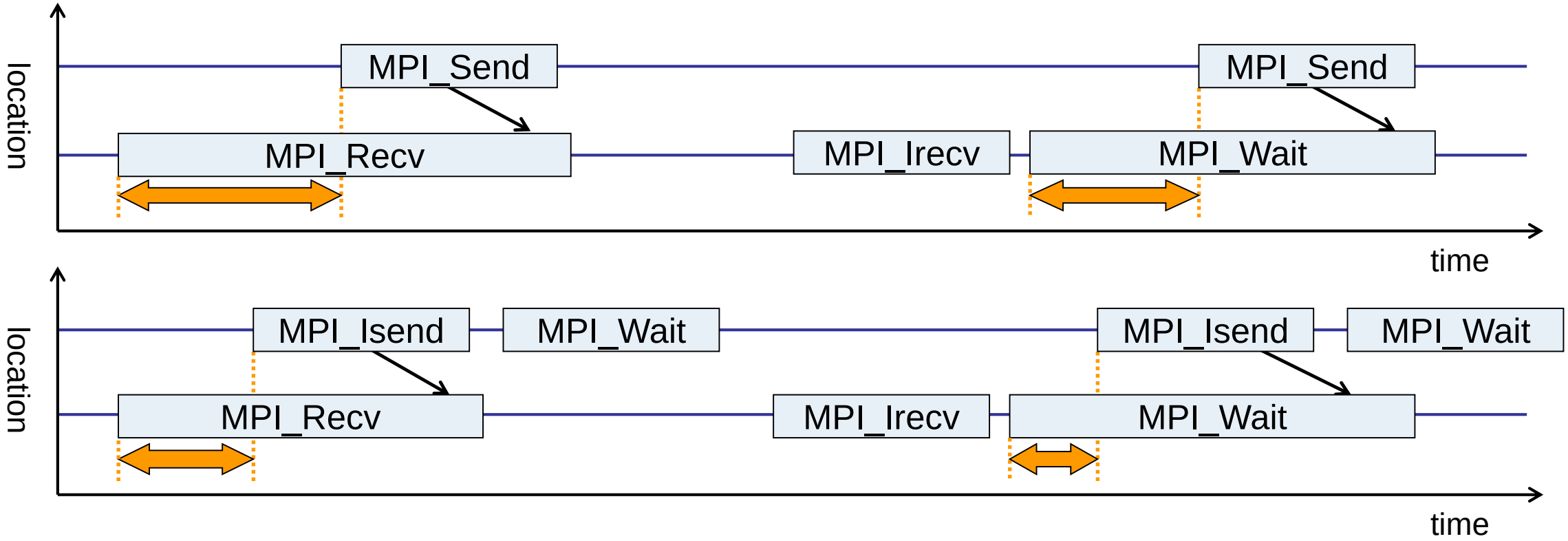
Automatic trace analysis

- Idea
  - Automatic search for patterns of inefficient behaviour
  - Classification of behaviour & quantification of significance
  - Identification of delays as root causes of inefficiencies



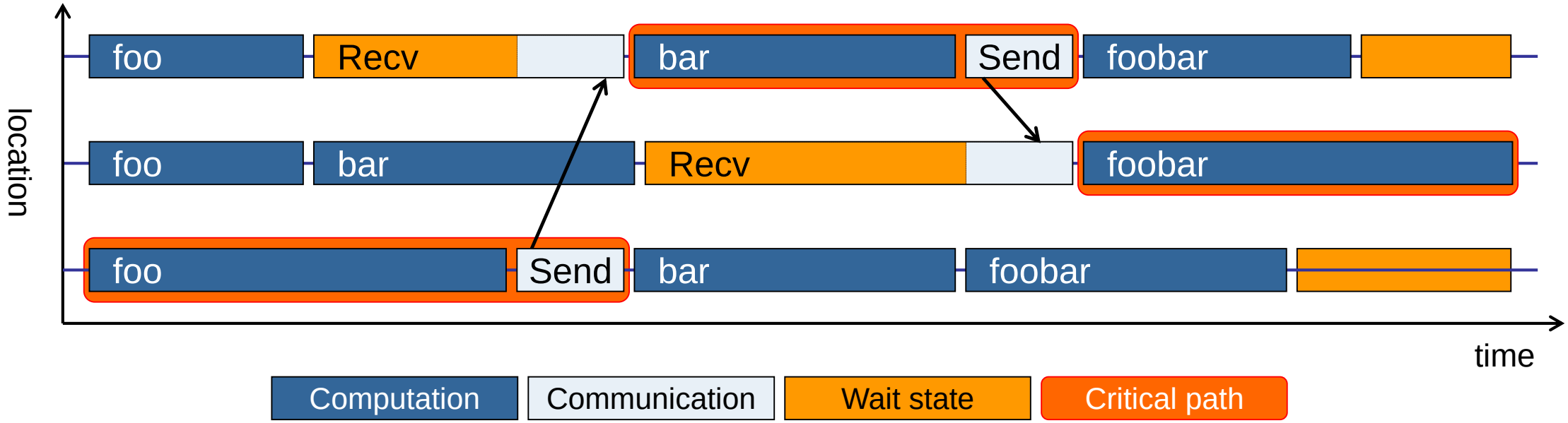
  - Guaranteed to cover the entire event trace
  - Quicker than manual/visual trace analysis
  - Parallel replay analysis exploits available memory & processors to deliver scalability

Example: "*Late Sender*" wait state



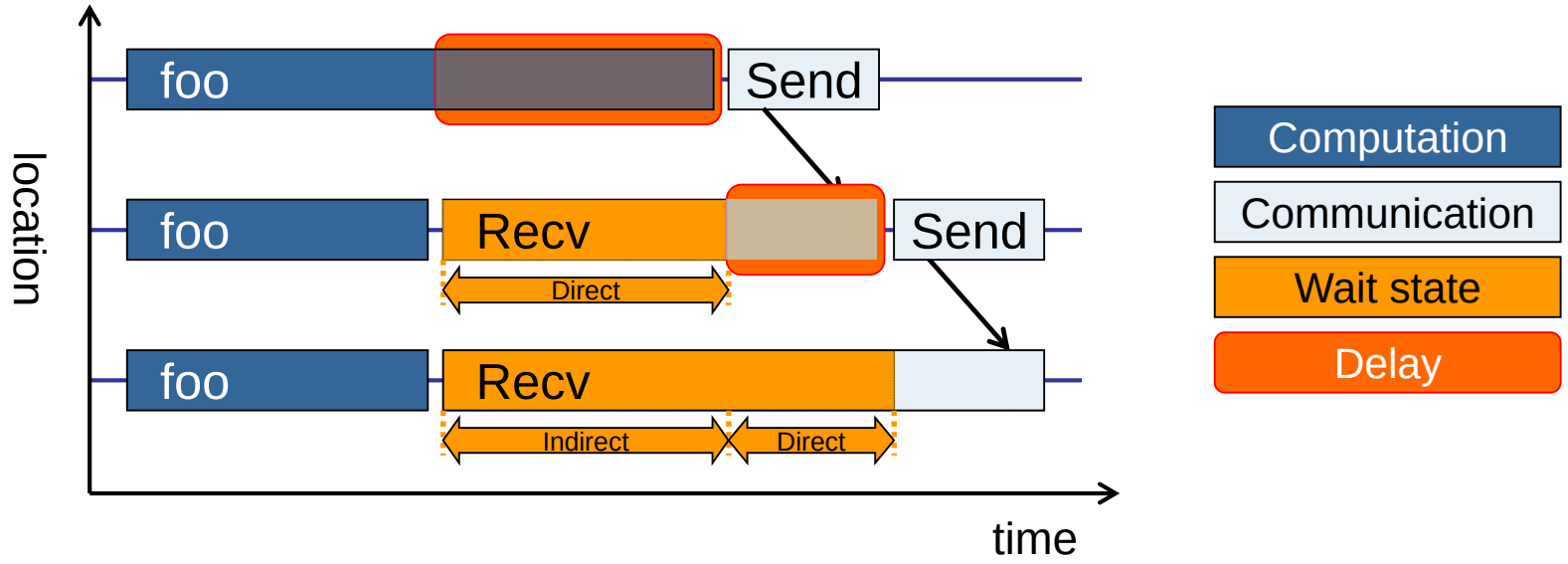
- Waiting time caused by a blocking receive operation posted earlier than the corresponding send
- Applies to blocking as well as non-blocking communication

Example: Critical path



- Shows call paths and processes/threads that are responsible for the program's wall-clock runtime
- Identifies good optimization candidates and parallelization bottlenecks

Example: Root-cause analysis



- Classifies wait states into direct and indirect (i.e., caused by other wait states)
- Identifies *delays* (excess computation/communication) as root causes of wait states
- Attributes wait states as *delay costs*

Demo / Hands-on: TeaLeaf case study (performed by Markus Geimer)

Checking resource requirements

```
% scorep-score -r scorep-20191202_1521_12347518278737660

Estimated aggregate size of event trace:                   5GB
Estimated requirements for largest trace buffer (max_buf): 663MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       687MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=687MB to avoid intermediate flushes
 or reduce requirements using USR regions filters.)

flt     type  max_buf[B]      visits time[s] time[%] time/visit[us]  region
         ALL 694,583,945 117,797,019 8089.01   100.0          68.67  ALL
         OMP 657,597,924 110,518,776 7992.14    98.8          72.31  OMP
         COM  17,360,018   4,696,860   33.47     0.4           7.13  COM
         MPI  16,624,782   1,657,880   61.72     0.8          37.23  MPI
         USR   3,002,610     923,495    1.68     0.0           1.81  USR
      SCOREP          41           8    0.00     0.0         322.11  SCOREP
```

Checking resource requirements

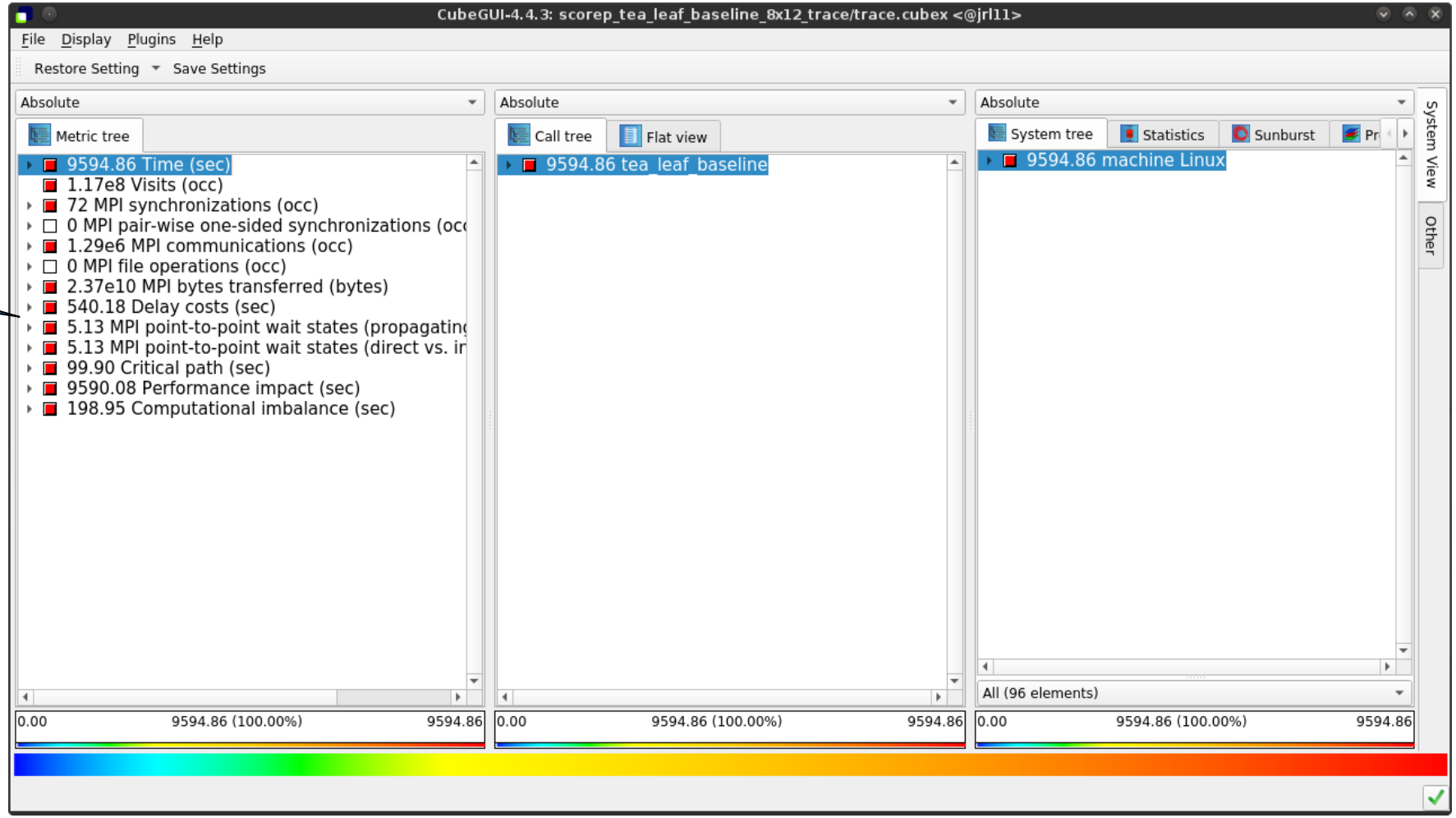
```
% scorep-score -r scorep-20191202_1521_12347518278737660 -c 2

Estimated aggregate size of event trace:                   12GB
Estimated requirements for largest trace buffer (max_buf): 1559MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       1583MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=1583MB to avoid intermediate flushes
 or reduce requirements using USR regions filters.)

flt     type    max_buf[B]      visits time[s] time[%] time/visit[us]  region
         ALL 1,634,686,051 117,797,019 8089.01   100.0          68.67  ALL
         OMP 1,537,910,256 110,518,776 7992.14    98.8          72.31  OMP
         COM    56,753,905   4,696,860   33.47     0.4           7.13  COM
         MPI    30,210,240   1,657,880   61.72     0.8          37.23  MPI
         USR     9,816,225     923,495    1.68     0.0           1.81  USR
      SCOREP           100           8    0.00     0.0         322.11  SCOREP
```

TeaLeaf Scalasca job script

```
% cat tealeaf_scalasca.job
[…]
module load DEVELOP DEV-TOOLS scorep otf2 scalasca

# Local trace buffer space without HWC measurement
export SCOREP_TOTAL_MEMORY=670MB

# With HWC measurement enabled, more buffer space is needed
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC
#export SCOREP_TOTAL_MEMORY=1600MB

#export SCALASCA_OPTIONS="-s"
#export SCALASCA_OPTIONS="-t"
export SCALASCA_OPTIONS="-q -t"

# Specific options for the Scalasca Parallel Analyzer
export SCAN_ANALYZE_OPTS="--time-correct"

scalasca -analyze ${SCALASCA_OPTIONS} $MPIEXEC $FLAGS_MPI_BATCH ./tea_leaf
```

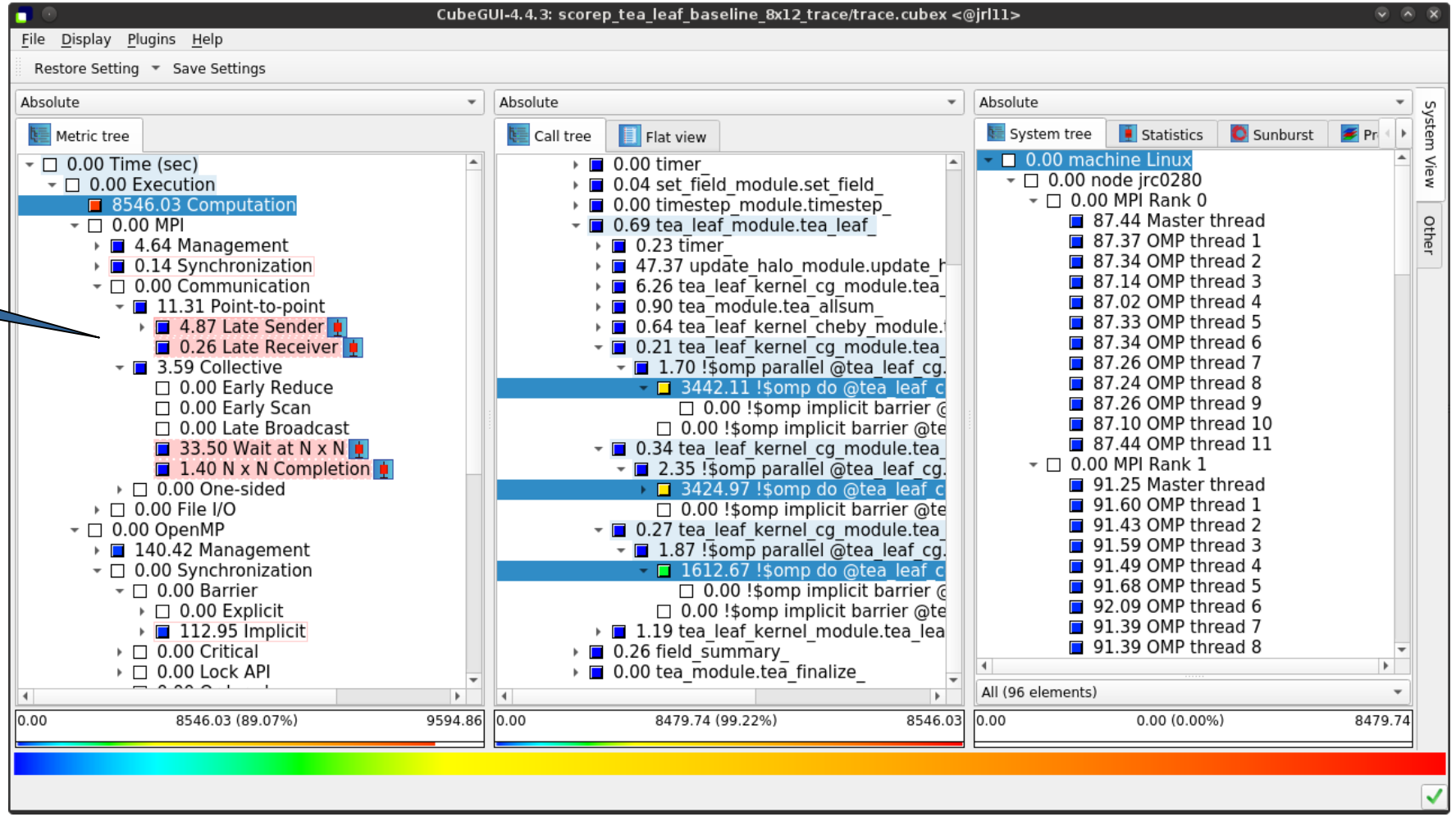
Scalasca analysis report exploration (opening view)

Additional top-level metrics produced by the trace analysis…



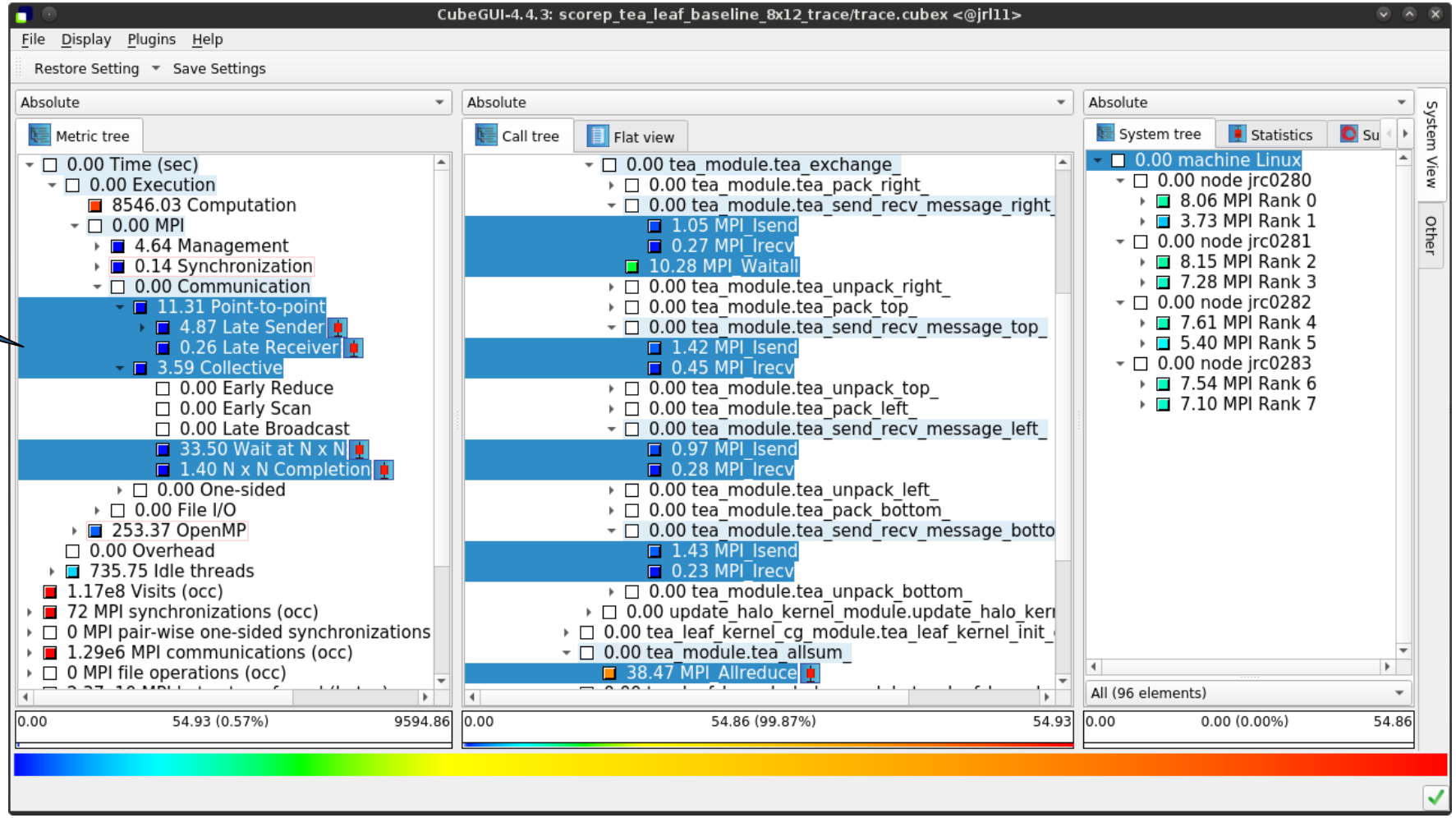
Scalasca wait-state metrics

…plus additional wait-state metrics as part of the “Time” hierarchy



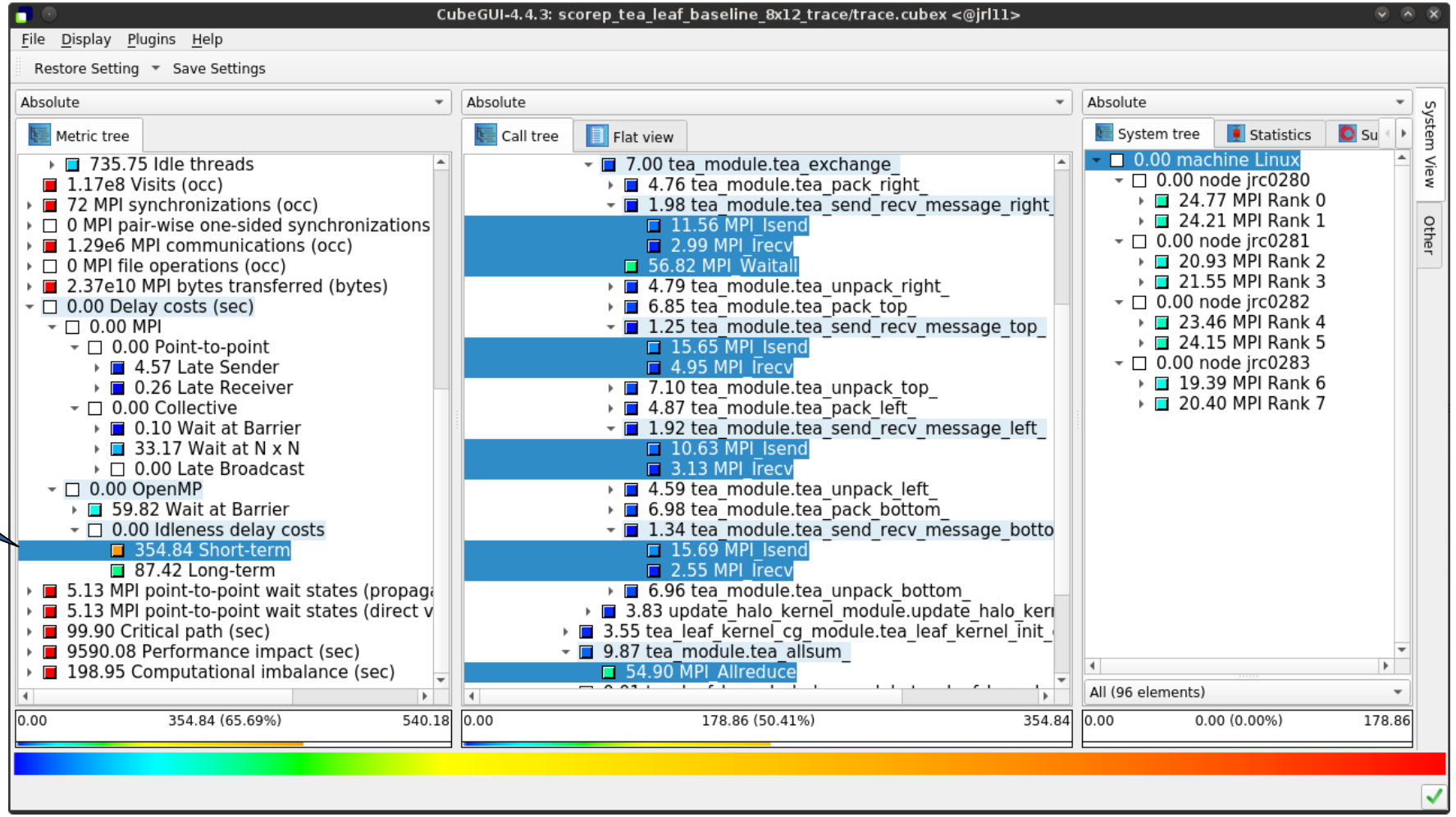
TeaLeaf Scalasca report analysis (I)

While MPI communication time and wait states are small (~0.6% of the total execution time)…



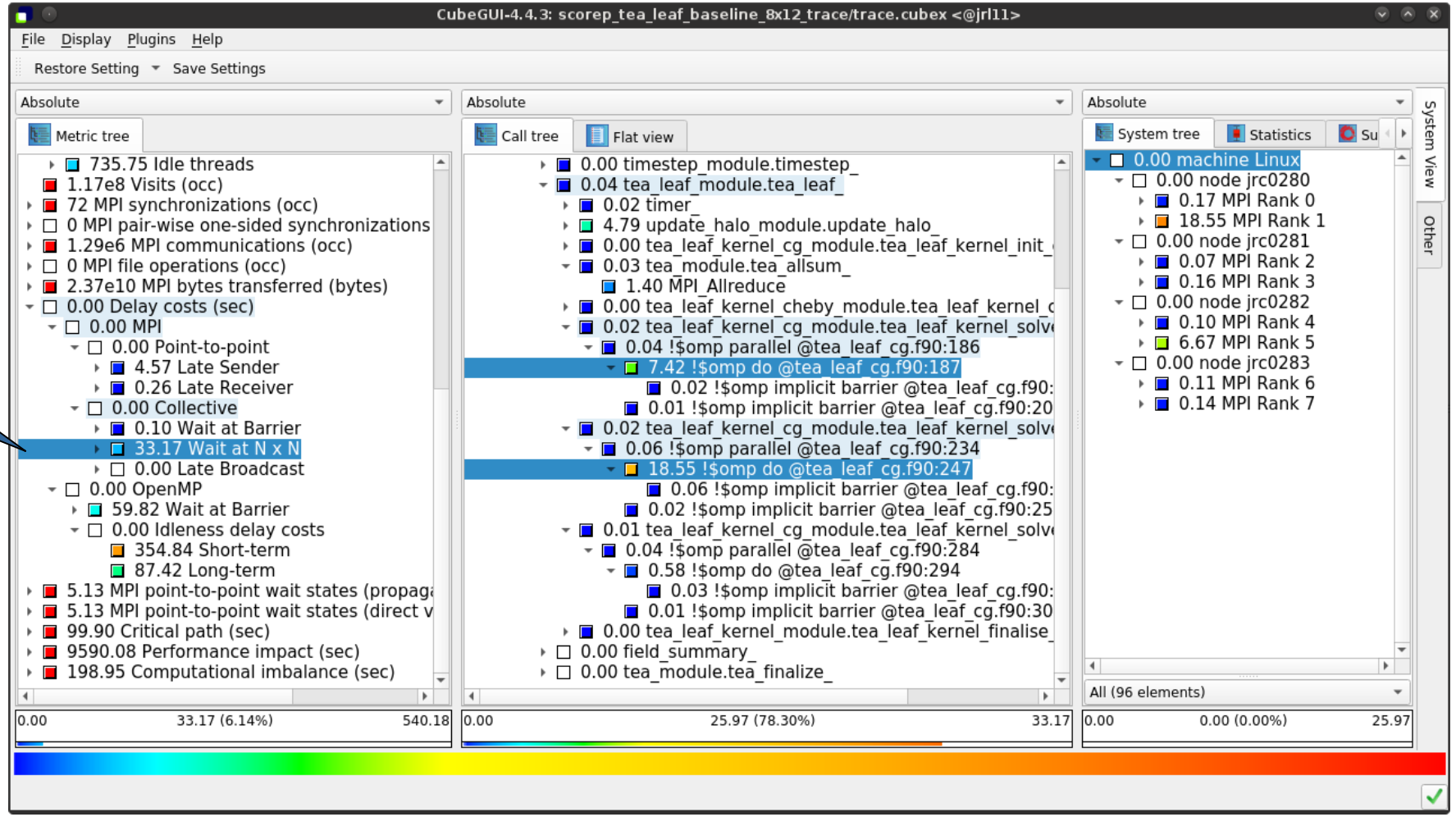
TeaLeaf Scalasca report analysis (II)

…they directly cause a significant amount of the OpenMP thread idleness



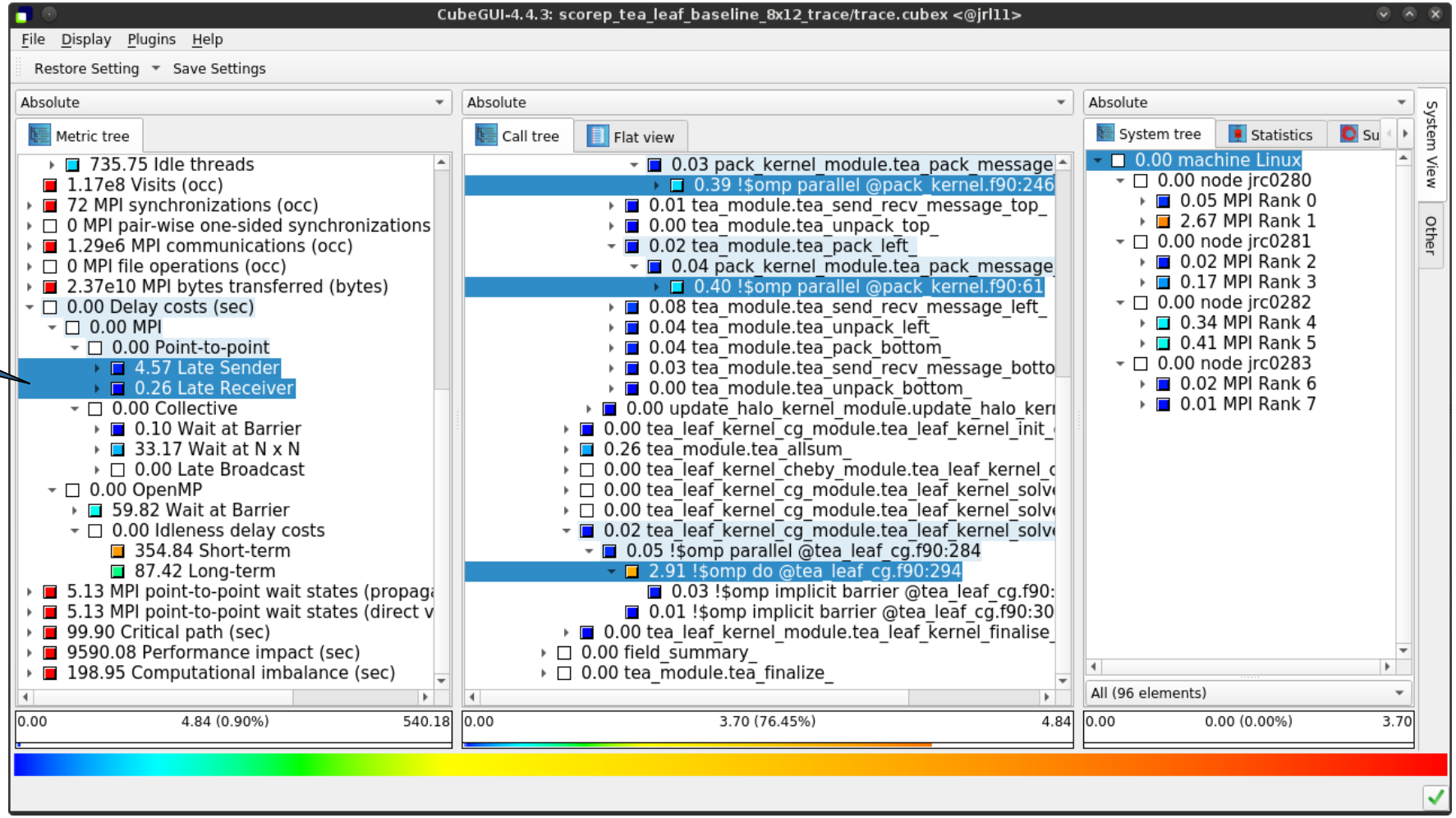
TeaLeaf Scalasca report analysis (III)

The “Wait at NxN” collective wait states are mostly caused by the first 2 OpenMP `do` loops of the solver (on ranks 5 & 1, resp.)…



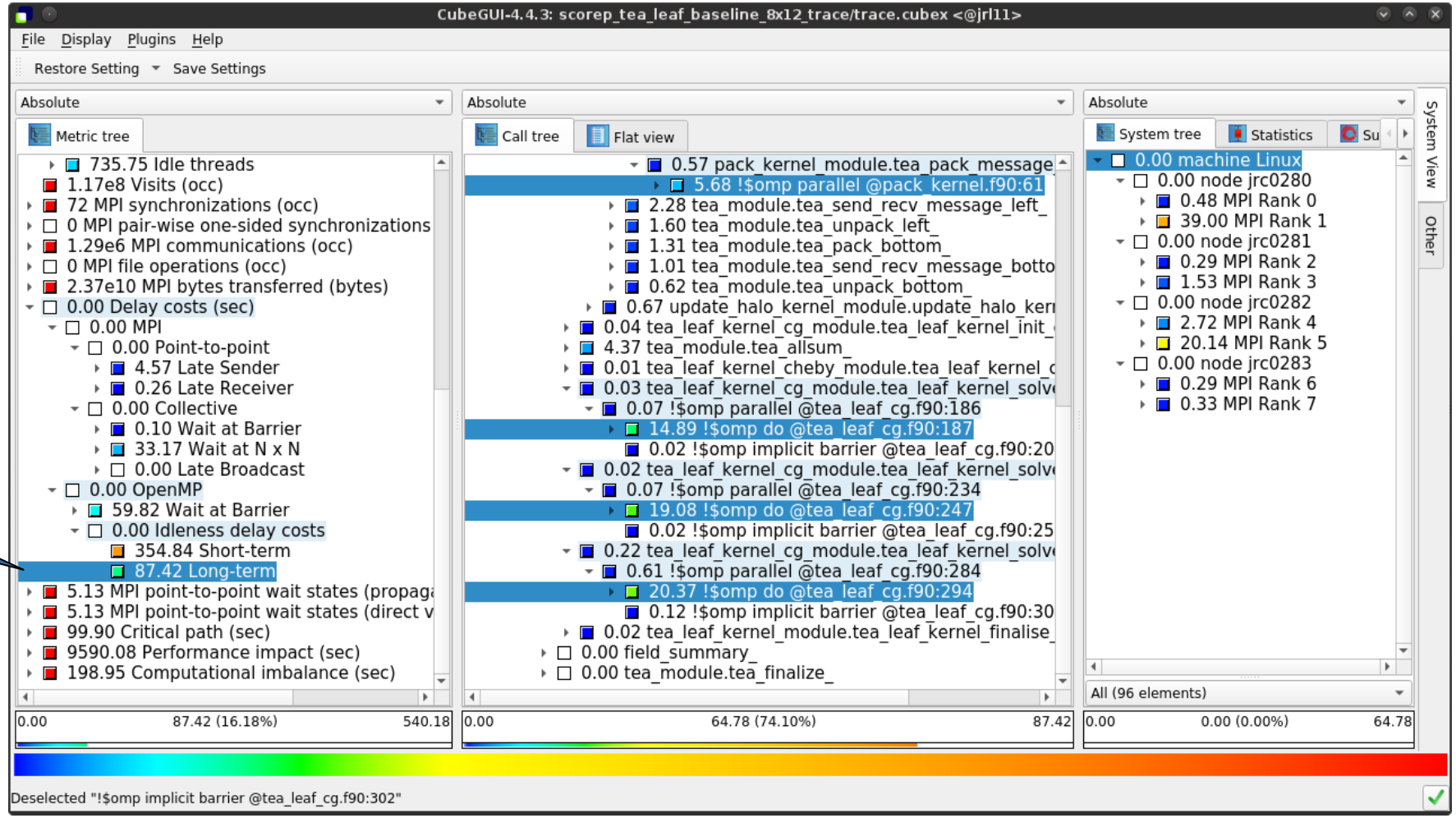
TeaLeaf Scalasca report analysis (IV)

…while the MPI point-to-point wait states are caused by the 3rd solver `do` loop (on rank 1) and two loops in the halo exchange



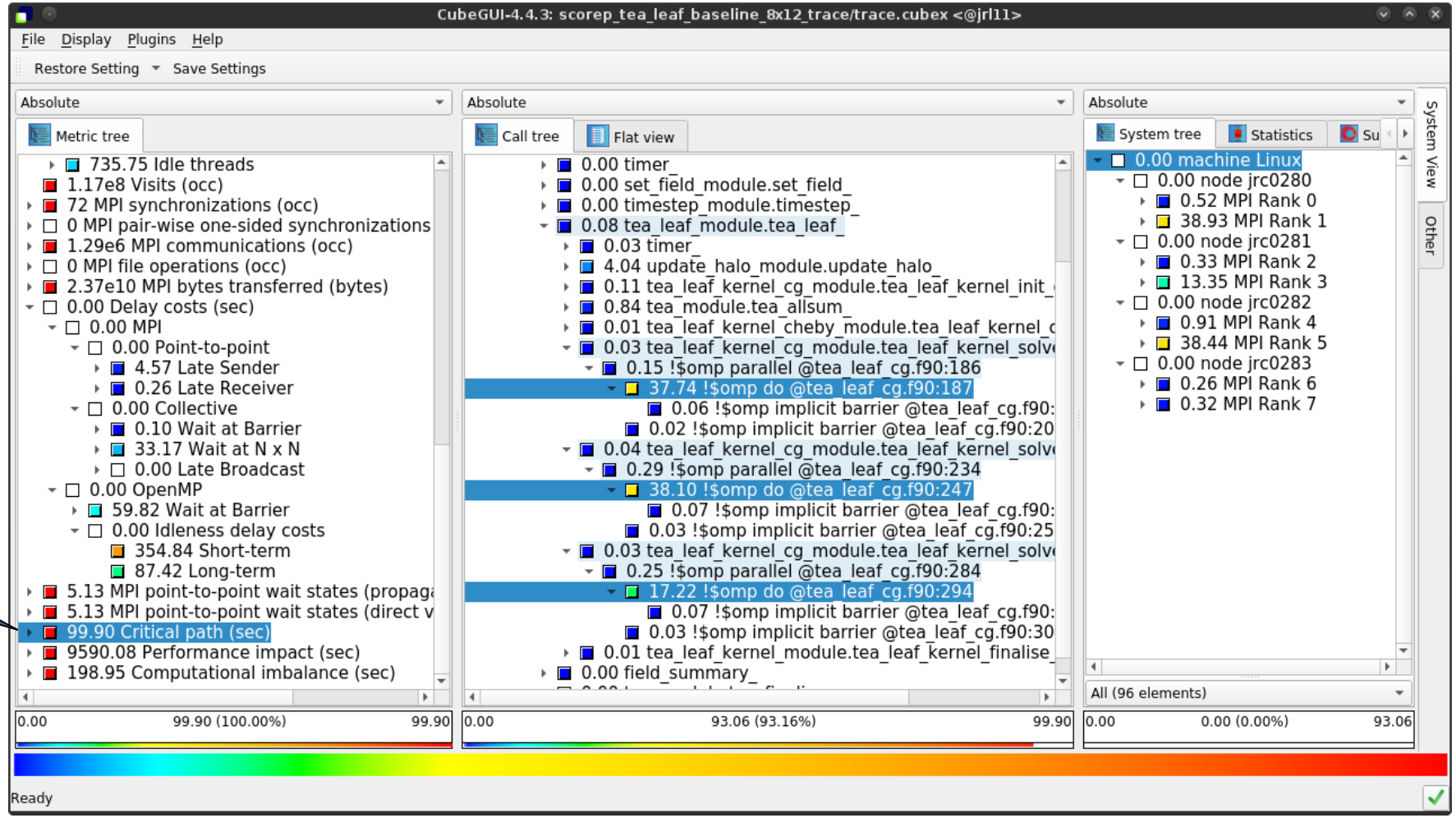
TeaLeaf Scalasca report analysis (V)

Various OpenMP `do` loops (incl. the solver loops) also cause OpenMP thread idleness on other ranks via propagation



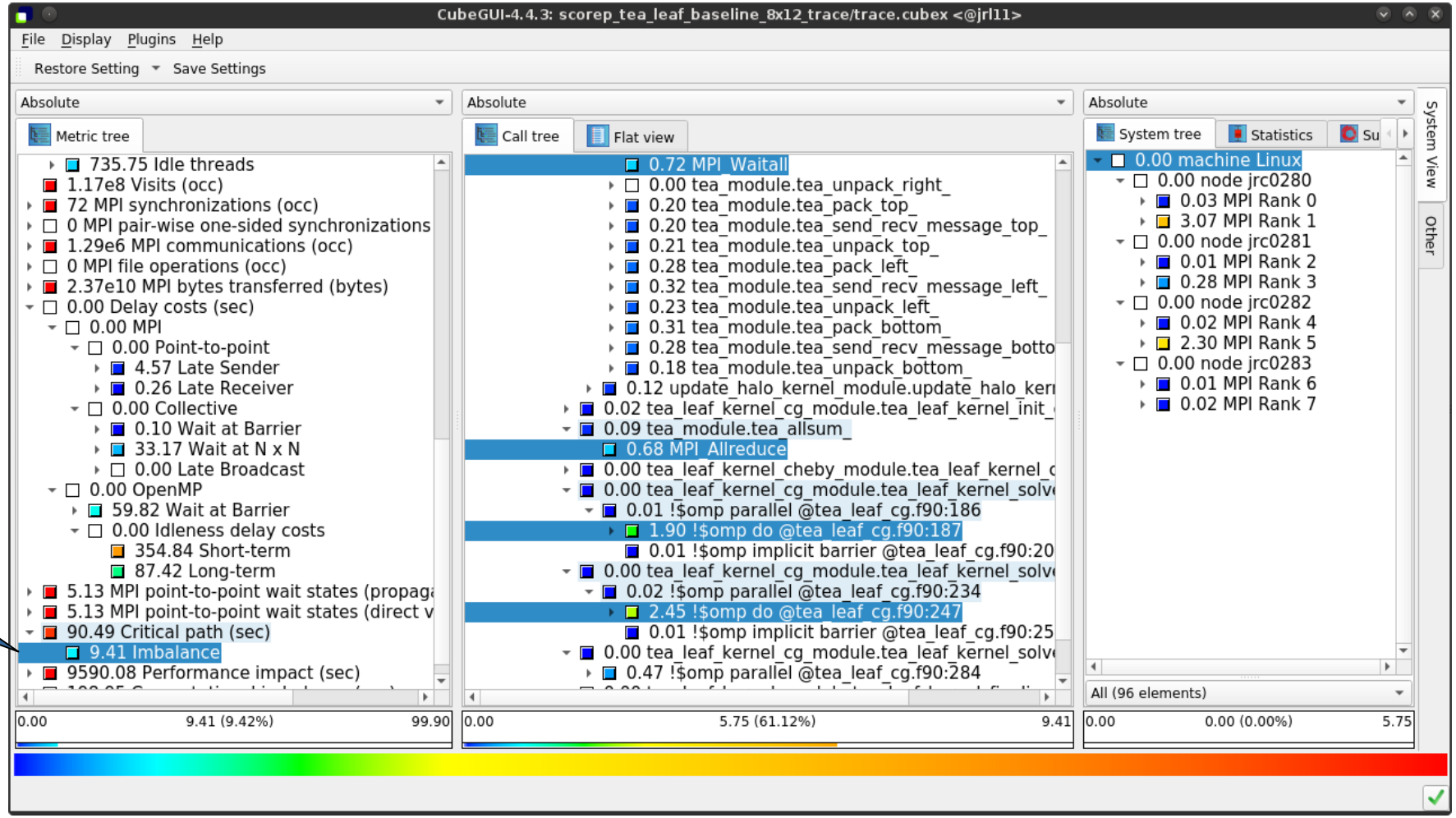
TeaLeaf Scalasca report analysis (VI)

The Critical Path also highlights the three solver loops…



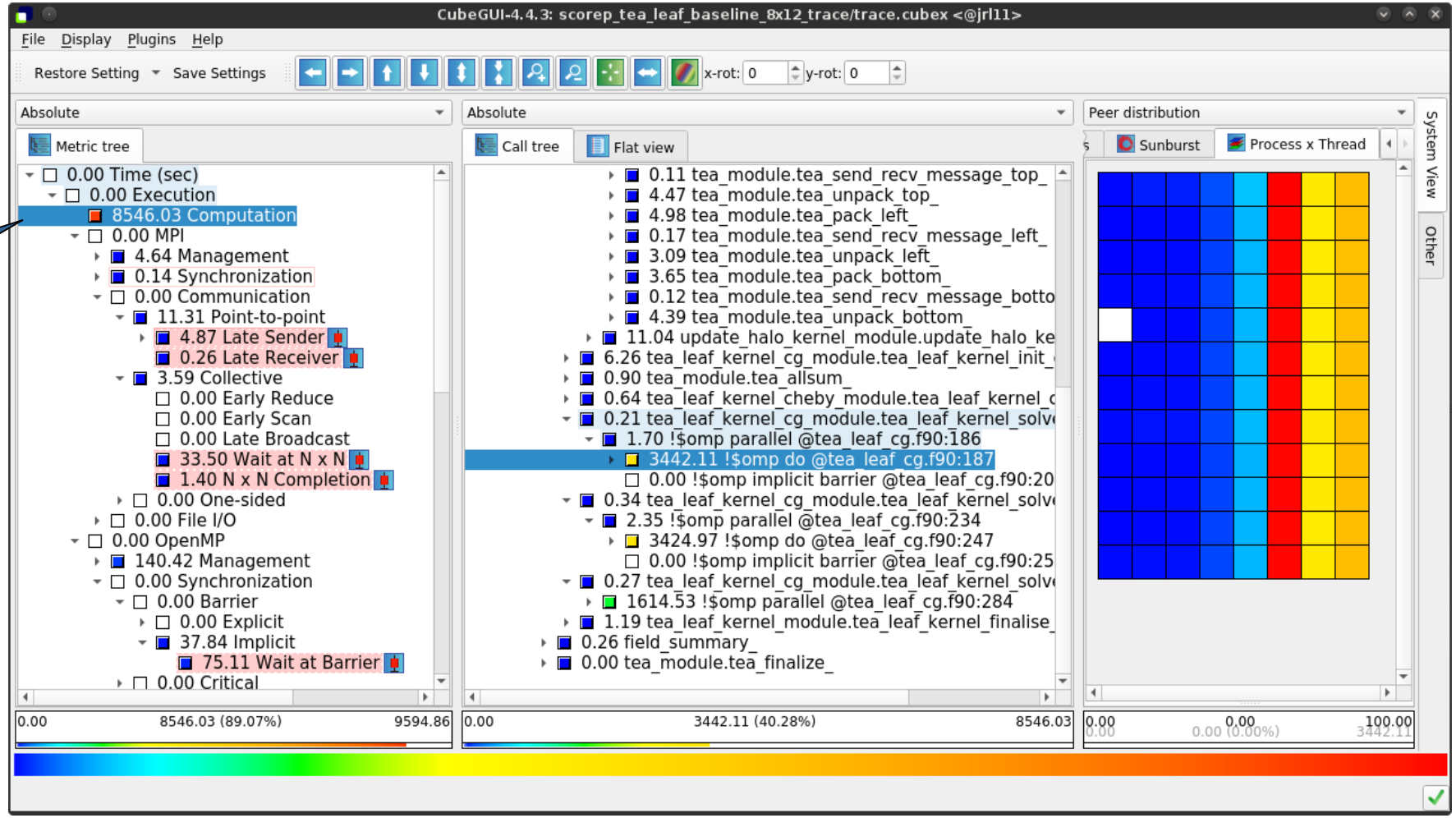
TeaLeaf Scalasca report analysis (VII)

…with imbalance (time on critical path above average) mostly in the first two loops and MPI communication



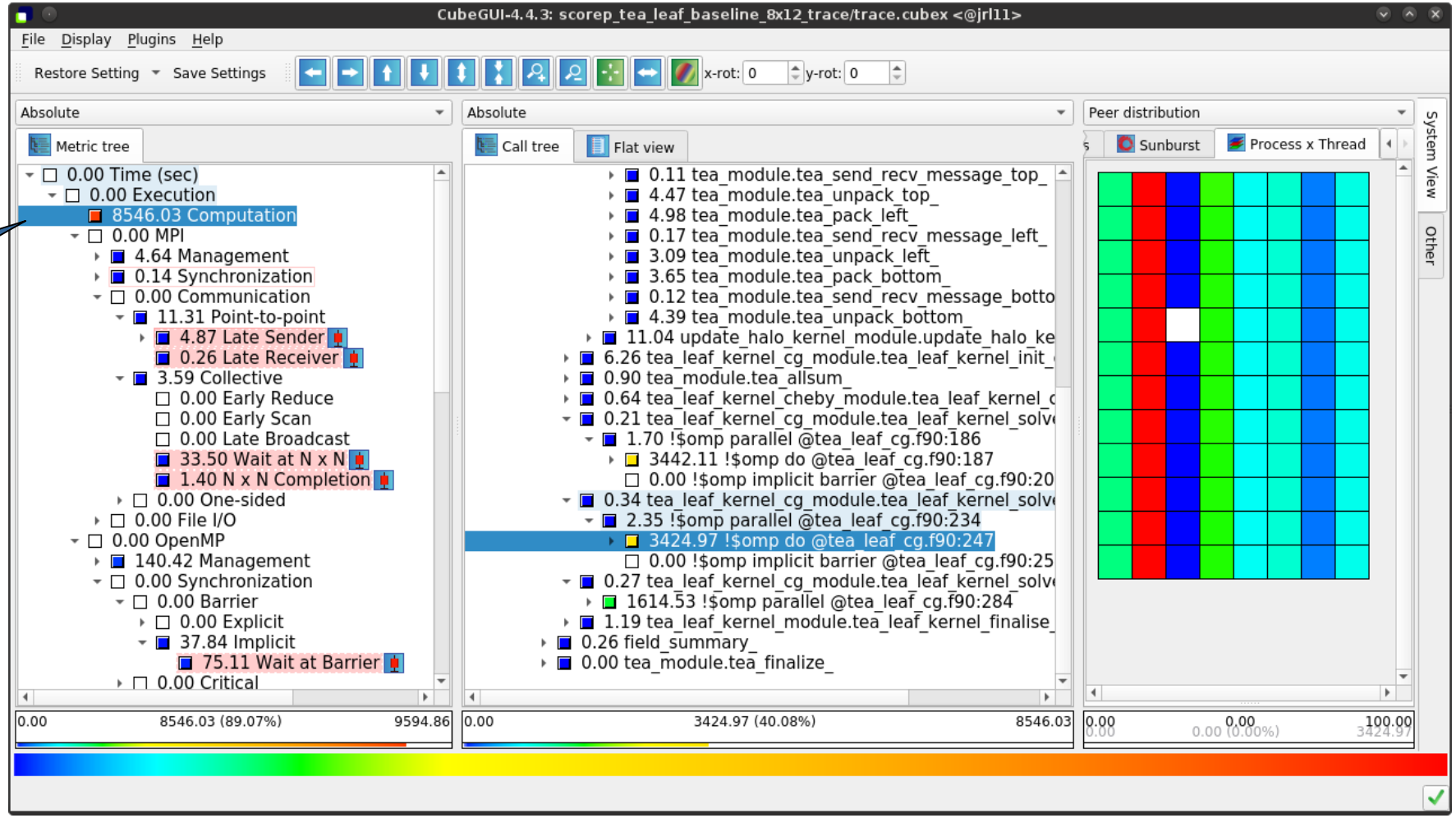
TeaLeaf Scalasca report analysis (VIII)

Computation time of 1st…



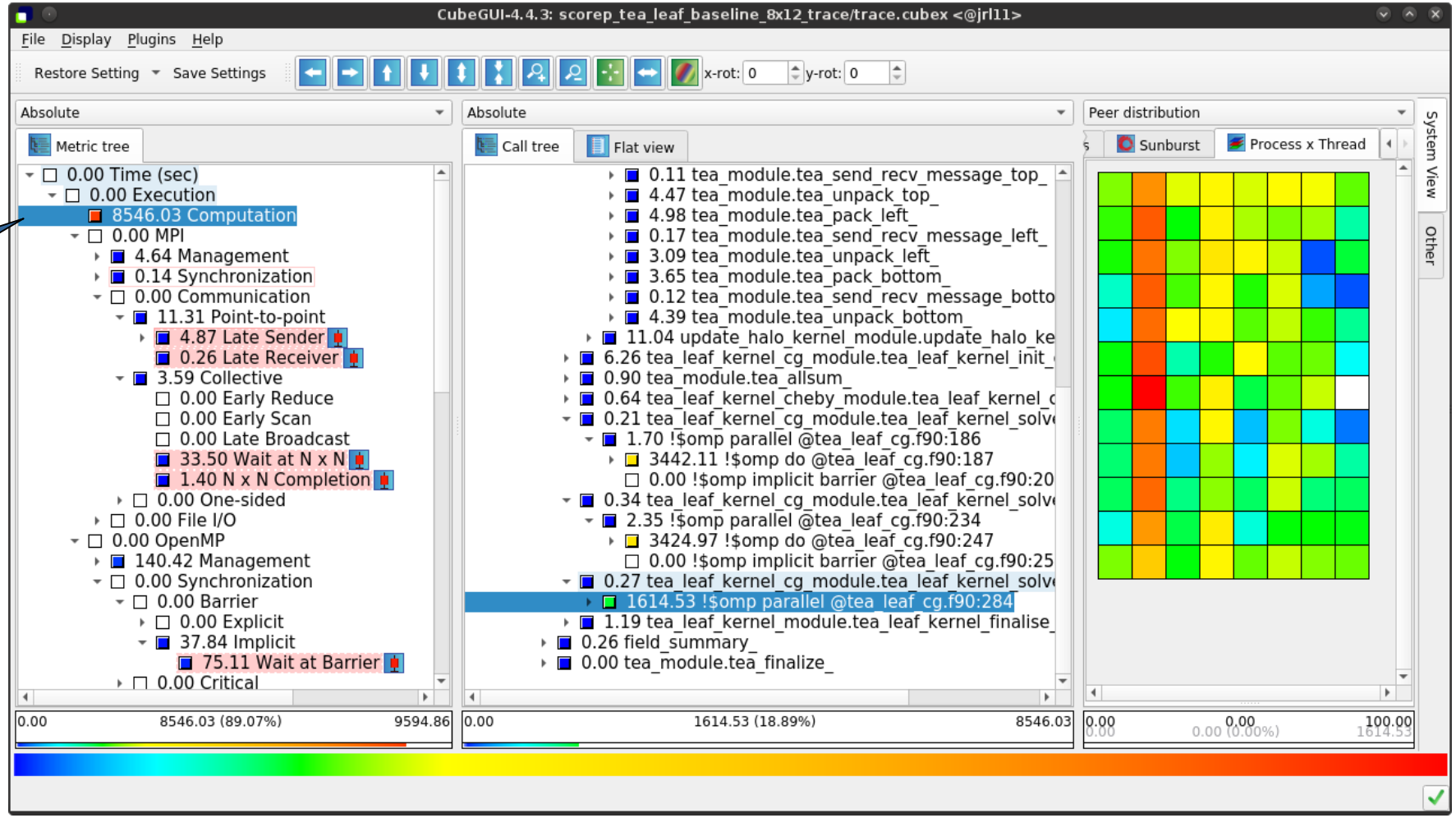
TeaLeaf Scalasca report analysis (IX)

…and 2nd `do` loop mostly balanced within each rank, but vary considerably across ranks…



TeaLeaf Scalasca report analysis (X)

…while the 3rd do loop also shows imbalance within each rank



TeaLeaf analysis summary

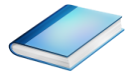
- The first two OpenMP do loops of the solver are well balanced within a rank, but are imbalanced across ranks
  - ➔ Requires a global load balancing strategy
- The third OpenMP do loop, however, is imbalanced within ranks,
  - causing direct "Wait at OpenMP Barrier" wait states,
  - which cause indirect MPI point-to-point wait states,
  - which in turn cause OpenMP thread idleness
  - ➔ Low-hanging fruit

- Adding a `SCHEDULE(guided)` clause reduced
  - the MPI point-to-point wait states by ~66%
  - the MPI collective wait states by ~50%
  - the OpenMP "Wait at Barrier" wait states by ~55%
  - the OpenMP thread idleness by ~11%
  - ➔ **Overall runtime (wall-clock) reduction by ~5%**

Scalasca Trace Tools: Further information

- Collection of trace-based performance tools
  - Specifically designed for large-scale systems
  - Features an automatic trace analyzer providing wait-state, critical-path, and delay analysis
  - Supports MPI, OpenMP, POSIX threads, and hybrid MPI+OpenMP/Pthreads
- Available under 3-clause BSD open-source license
- Documentation & sources:
  - http://www.scalasca.org
- Contact:
  - mailto: scalasca@fz-juelich.de

trace tools scalasca

Reference material 

---

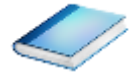
trace tools
scalasca

Scalasca command – One command for (almost) everything

```
% scalasca
Scalasca 2.5
Toolset for scalable performance analysis of large-scale parallel applications
usage: scalasca [OPTION]... ACTION <argument>...
    1. prepare application objects and executable for measurement:
       scalasca -instrument <compile-or-link-command> # skin (using scorep)
    2. run application under control of measurement system:
       scalasca -analyze <application-launch-command> # scan
    3. interactively explore measurement analysis report:
       scalasca -examine <experiment-archive|report>  # square

Options:
   -c, --show-config     show configuration summary and exit
   -h, --help            show this help and exit
   -n, --dry-run         show actions without taking them
       --quickref        show quick reference guide and exit
       --remap-specfile  show path to remapper specification file and exit
   -v, --verbose         enable verbose commentary
   -V, --version         show version information and exit
```

- The ‘`scalasca -instrument`’ command is deprecated and only provided for backwards compatibility with Scalasca 1.x., recommended: use Score-P instrumenter directly

Scalasca compatibility command: skin / scalasca -instrument



```
% skin
Scalasca 2.5: application instrumenter (using Score-P instrumenter)
usage: skin [-v] [-comp] [-pdt] [-pomp] [-user] [--*] <compile-or-link-command>
  -comp={all|none|...}: routines to be instrumented by compiler [default: all]
                        (... custom instrumentation specification depends on compiler)
  -pdt:  process source files with PDT/TAU instrumenter
  -pomp: process source files for POMP directives
  -user: enable EPIK user instrumentation API macros in source code
  -v:    enable verbose commentary when instrumenting

  --*:   options to pass to Score-P instrumenter
```

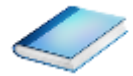
- Scalasca application instrumenter
  - Provides compatibility with Scalasca 1.x
  - **Deprecated! Use Score-P instrumenter directly.**

Scalasca convenience command: scan / scalasca -analyze

```
% scan
Scalasca 2.5: measurement collection & analysis nexus
usage: scan {options} [launchcmd [launchargs]] target [targetargs]
       where {options} may include:
  -h    Help       : show this brief usage message and exit.
  -v    Verbose    : increase verbosity.
  -n    Preview    : show command(s) to be launched but don't execute.
  -q    Quiescent  : execution with neither summarization nor tracing.
  -s    Summary    : enable runtime summarization. [Default]
  -t    Tracing    : enable trace collection and analysis.
  -a    Analyze    : skip measurement to (re-)analyze an existing trace.
  -e exptdir       : Experiment archive to generate and/or analyze.
                     (overrides default experiment archive title)
  -f filtfile      : File specifying measurement filter.
  -l lockfile      : File that blocks start of measurement.
  -R #runs         : Specify the number of measurement runs per config.
  -M cfgfile       : Specify a config file for a multi-run measurement.
```

- Scalasca measurement collection & analysis nexus

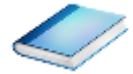
Scalasca advanced command: scout - Scalasca automatic trace analyzer



```
% scout.hyb --help
SCOUT   (Scalasca 2.5)
Copyright (c) 1998-2019 Forschungszentrum Juelich GmbH
Copyright (c) 2009-2014 German Research School for Simulation Sciences GmbH

Usage: <launchcmd> scout.hyb [OPTION]... <ANCHORFILE | EPIK DIRECTORY>
Options:
  --statistics       Enables instance tracking and statistics [default]
  --no-statistics    Disables instance tracking and statistics
  --critical-path    Enables critical-path analysis [default]
  --no-critical-path Disables critical-path analysis
  --rootcause        Enables root-cause analysis [default]
  --no-rootcause     Disables root-cause analysis
  --single-pass      Single-pass forward analysis only
  --time-correct     Enables enhanced timestamp correction
  --no-time-correct  Disables enhanced timestamp correction [default]
  --verbose, -v      Increase verbosity
  --help             Display this information and exit
```

- Provided in serial (.ser), OpenMP (.omp), MPI (.mpi) and MPI+OpenMP (.hyb) variants

Scalasca advanced command: clc_synchronize

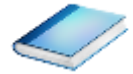


- Scalasca trace event timestamp consistency correction

```
Usage: <launchcmd> clc_synchronize.hyb <ANCHORFILE | EPIK_DIRECTORY>
```

  - Provided in MPI (.mpi) and MPI+OpenMP (.hyb) variants
  - Takes as input a trace experiment archive where the events may have timestamp inconsistencies
    - E.g., multi-node measurements on systems without adequately synchronized clocks on each compute node
  - Generates a new experiment archive (always called ./clc_sync) containing a trace with event timestamp inconsistencies resolved
    - E.g., suitable for detailed examination with a time-line visualizer

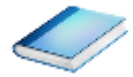
Scalasca convenience command: square / scalasca -examine



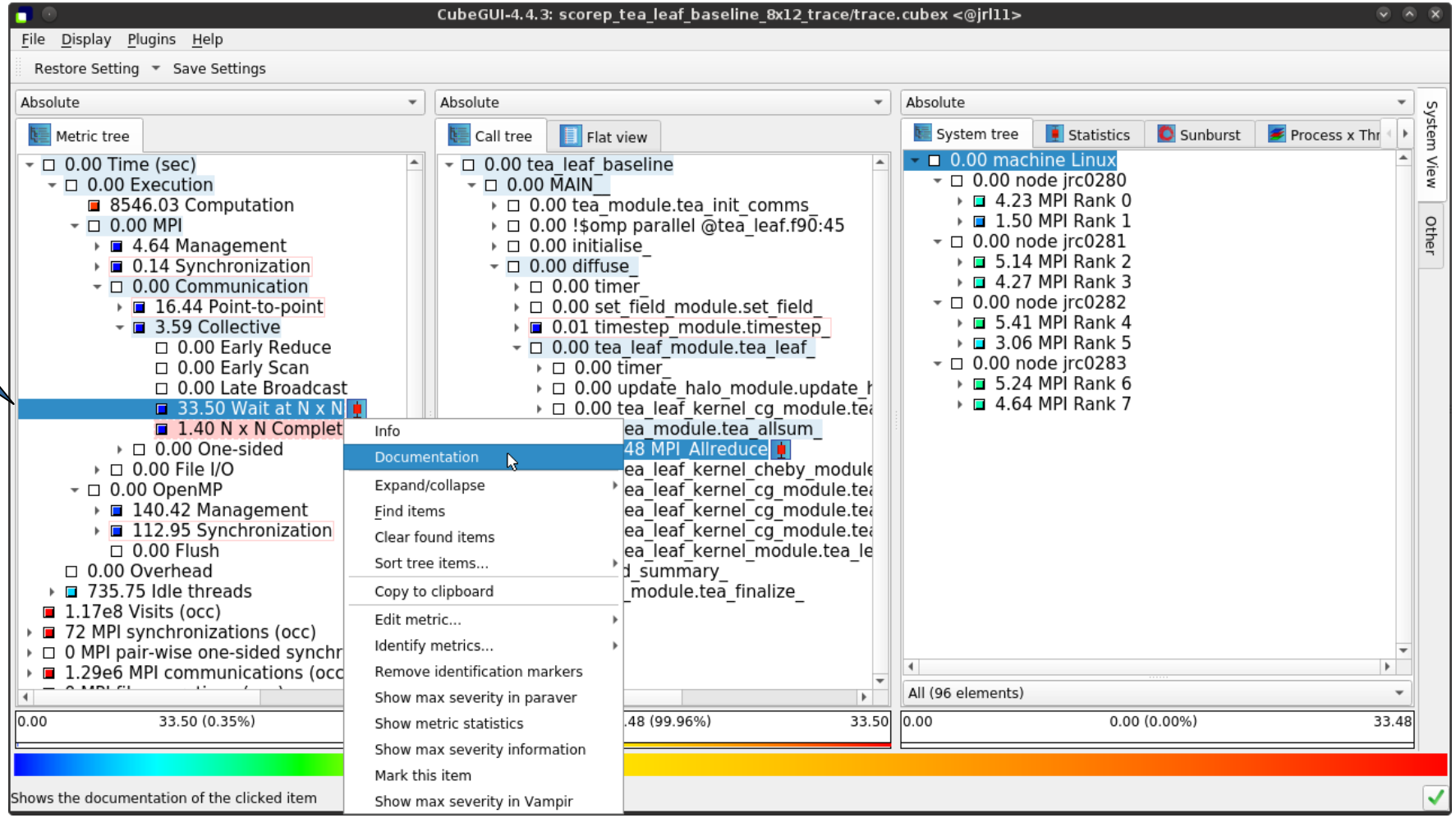
```
% square
Scalasca 2.5: analysis report explorer
usage: square [OPTIONS] <experiment archive | cube file>
   -c <none | quick | full> : Level of sanity checks for newly created reports
   -F                       : Force remapping of already existing reports
   -f filtfile              : Use specified filter file when doing scoring (-s)
   -s                       : Skip display and output textual score report
   -v                       : Enable verbose mode
   -n                       : Do not include idle thread metric
   -S <mean | merge>        : Aggregation method for summarization results of
                              each configuration (default: merge)
   -T <mean | merge>        : Aggregation method for trace analysis results of
                              each configuration (default: merge)
   -A                       : Post-process every step of a multi-run experiment
```

- Scalasca analysis report explorer (Cube)

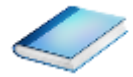
Online metric description



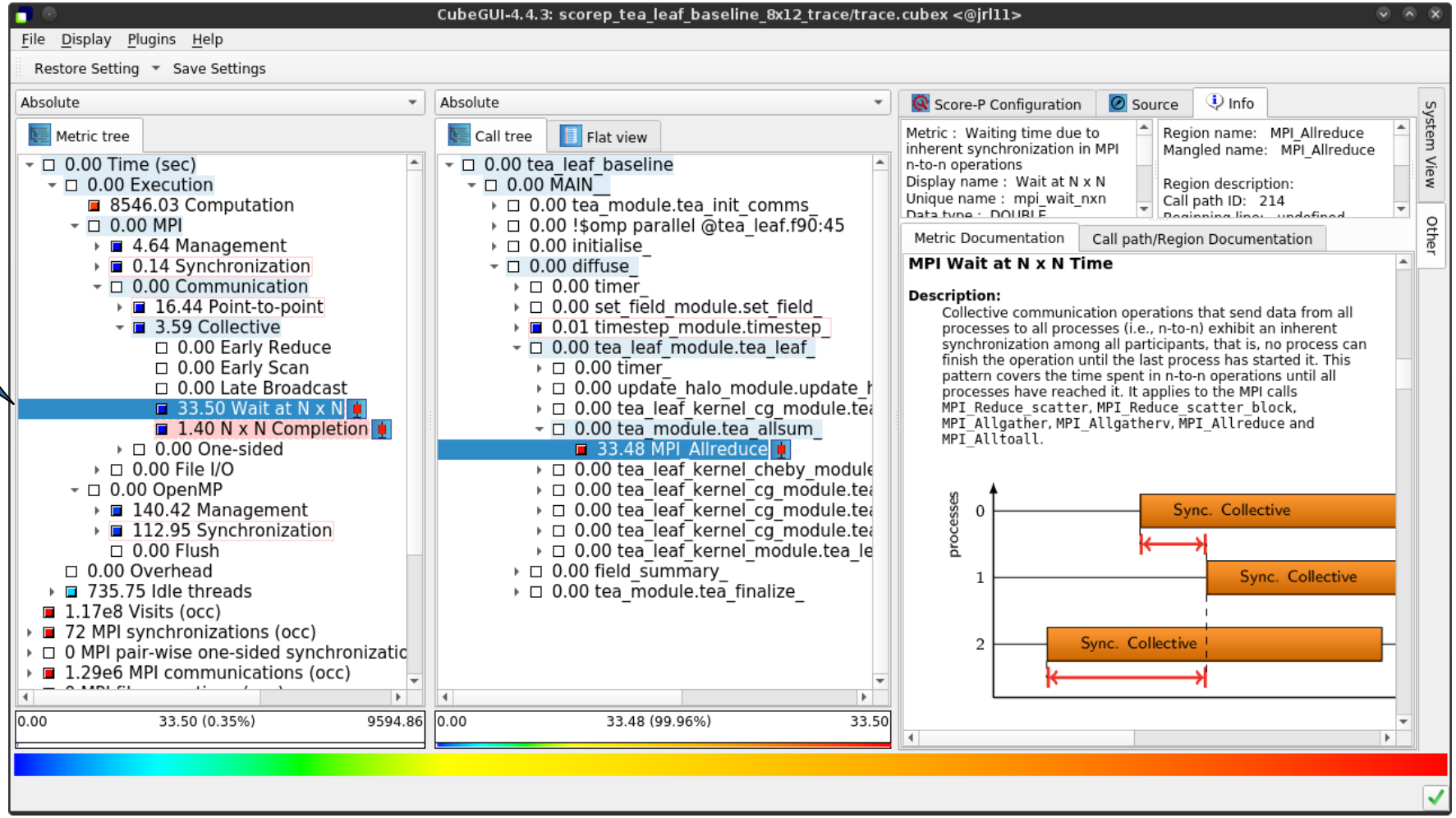
Access online metric description via context menu (right-click)



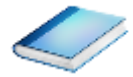
Online metric description (cont.)



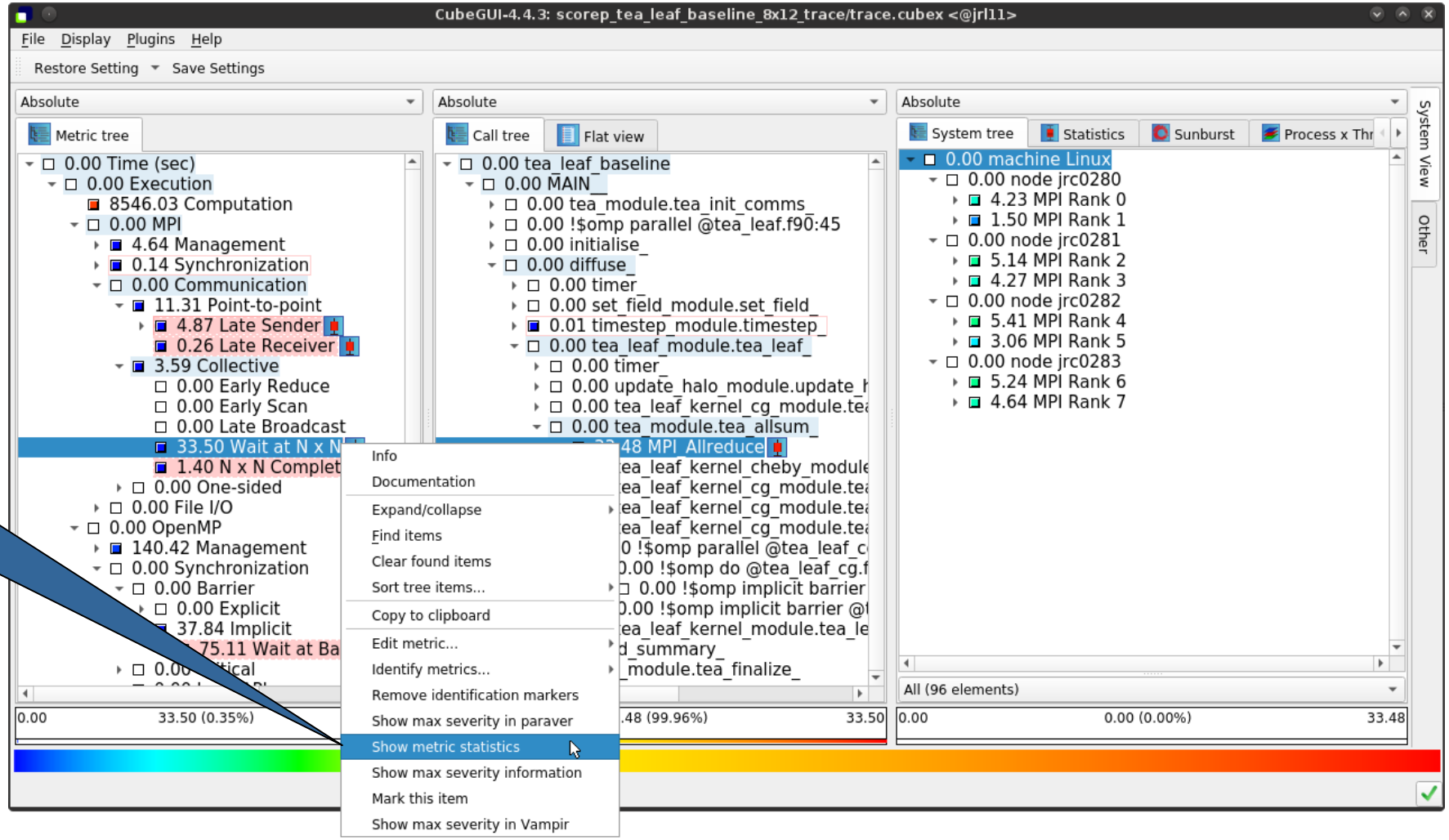
Selection of different metric automatically updates description



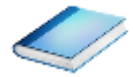
Metric statistics



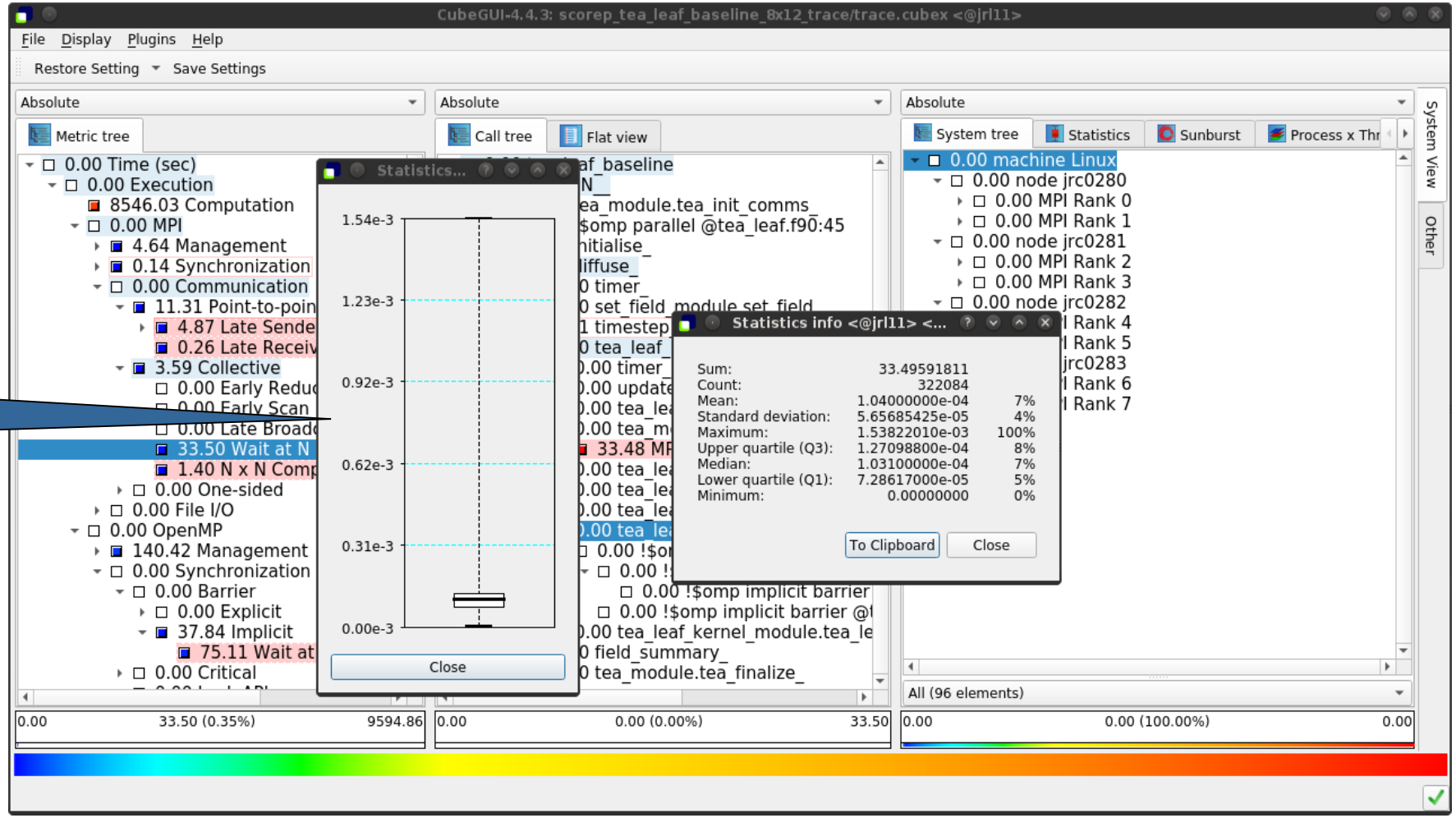
Access metric statistics for metrics marked with box plot icon from context menu



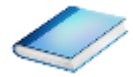
Metric statistics (cont.)



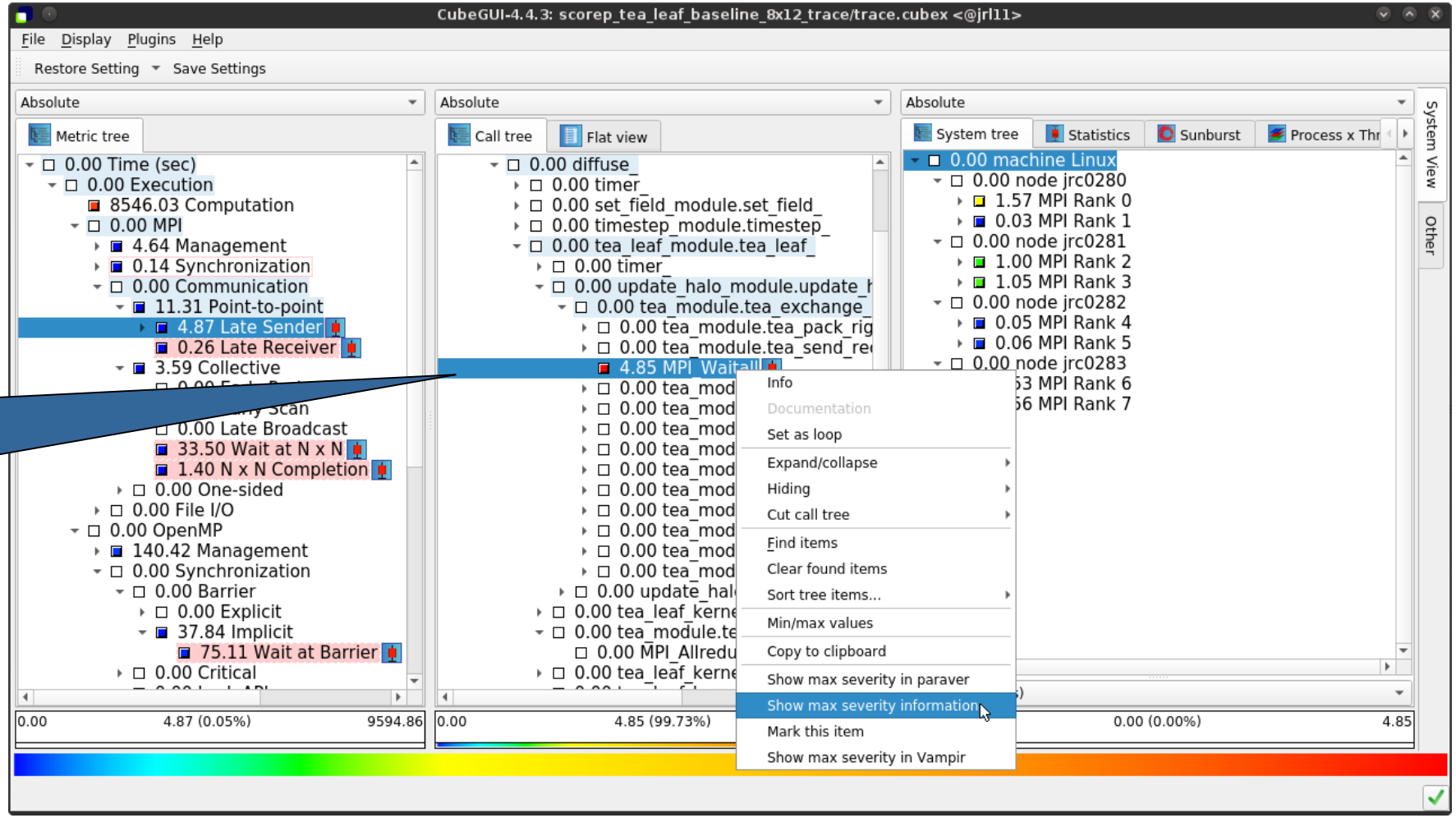
Shows instance statistics box plot, click to get details



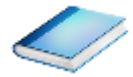
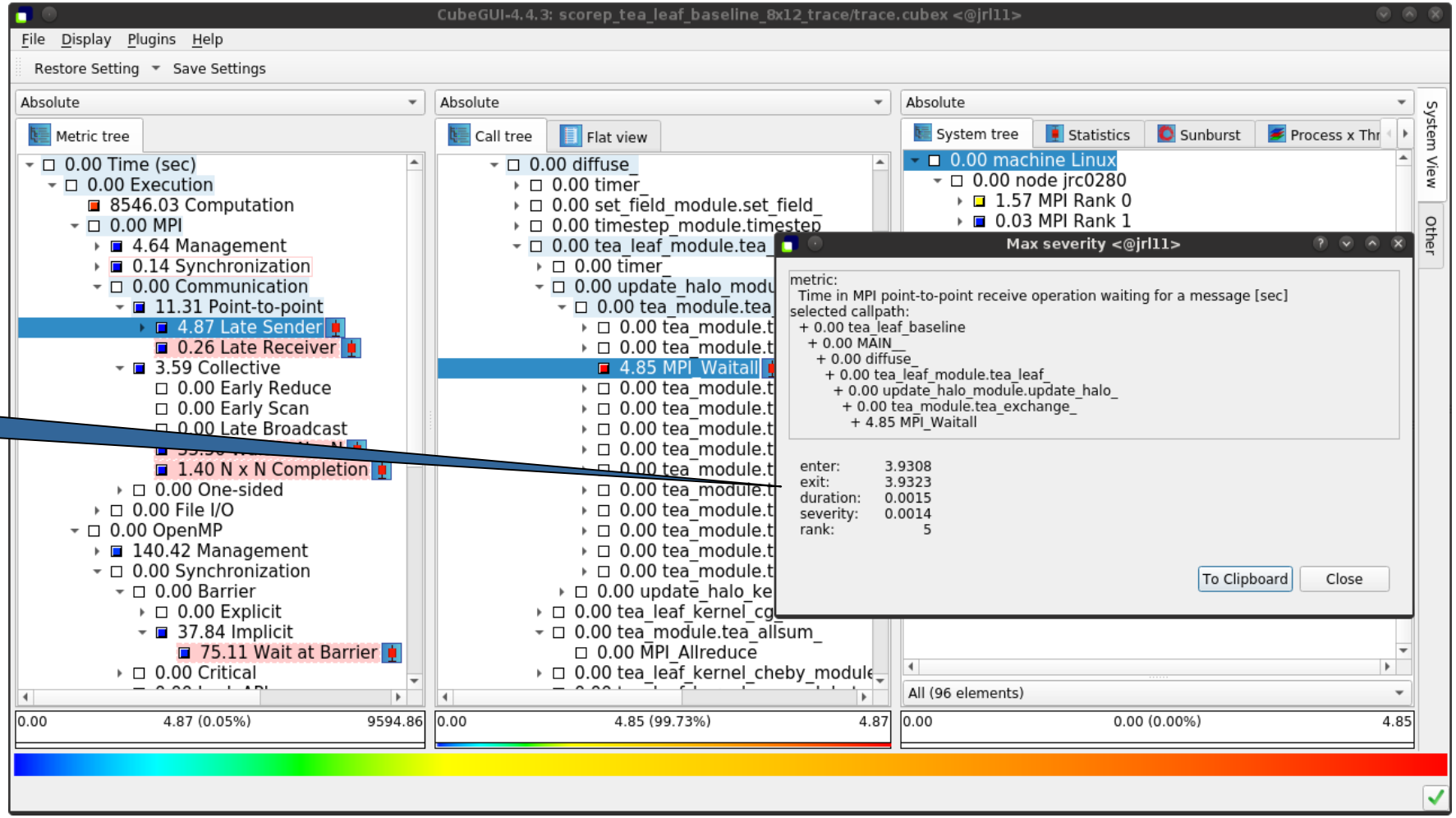
Metric instance statistics



Access most-severe instance information for call paths marked with box plot icon via context menu



Metric instance statistics (cont.)



Shows instance details