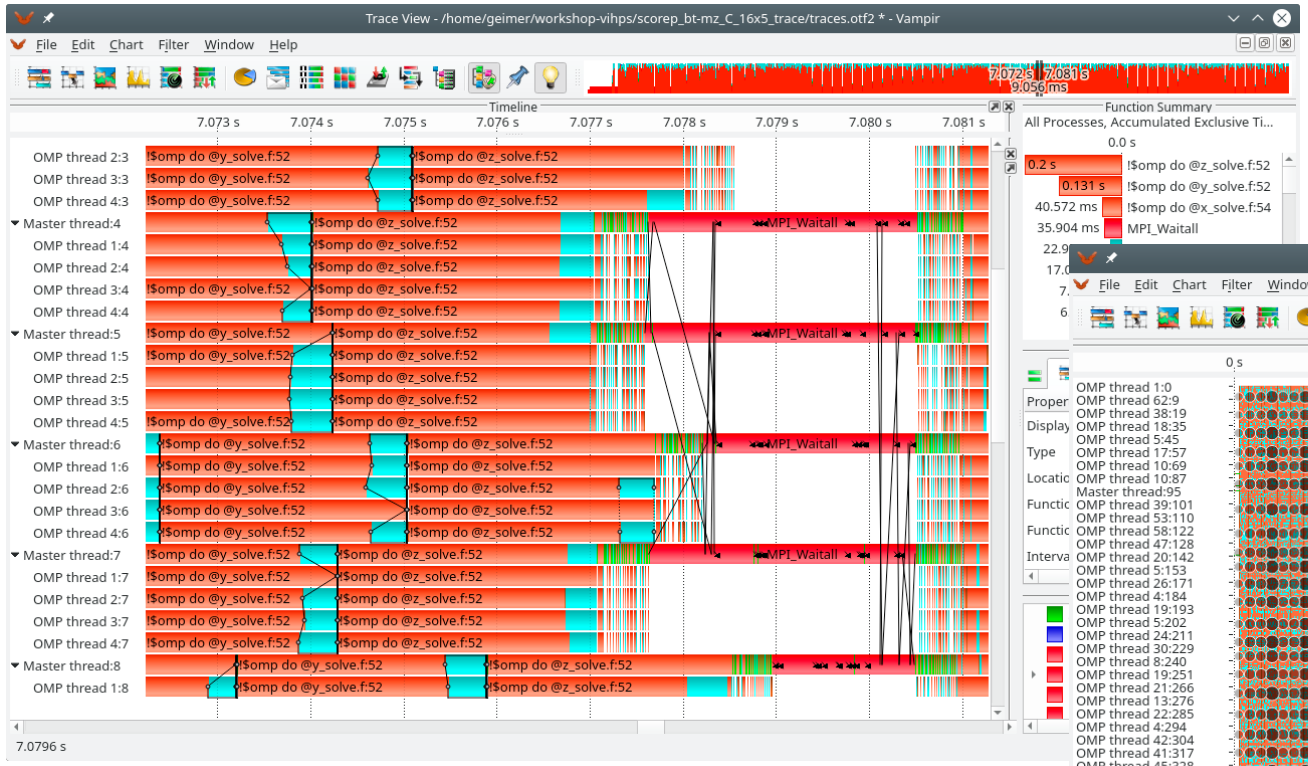
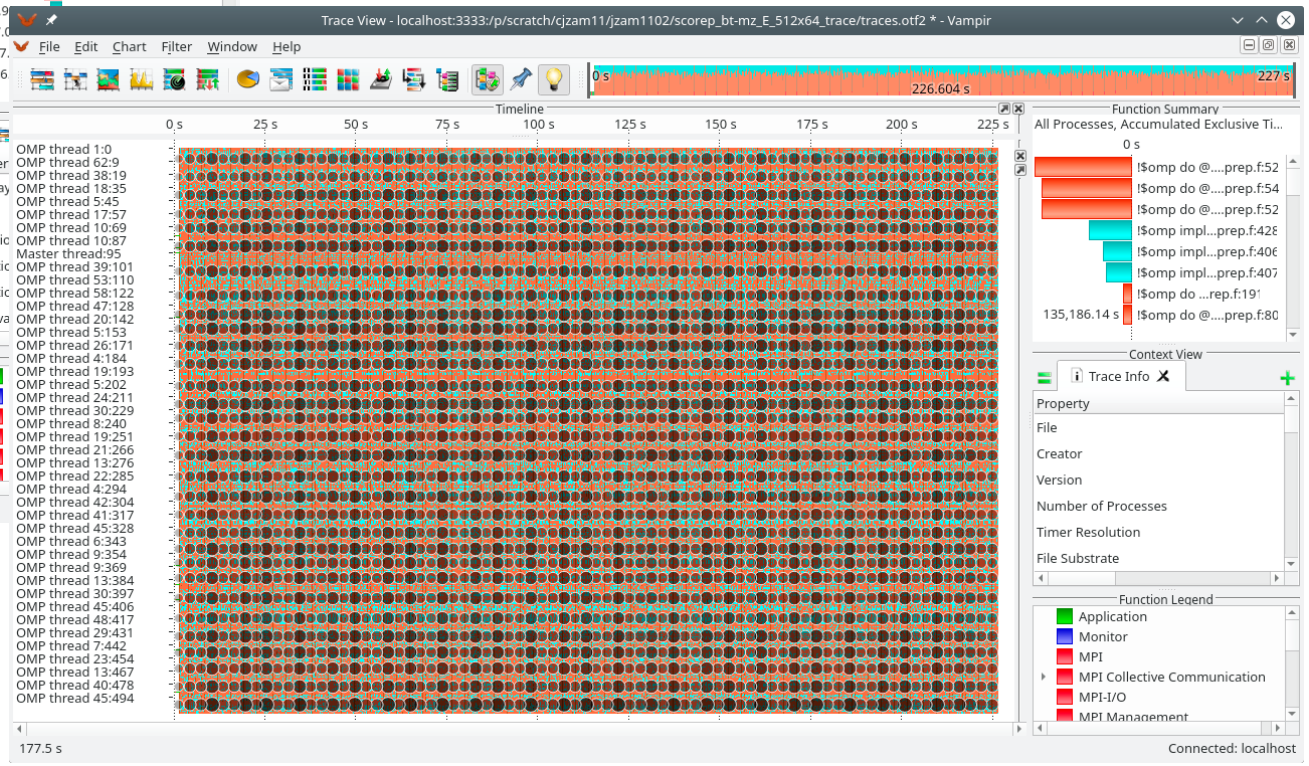
Scalasca Trace Tools

Demo/Hands-on: Automatic trace analysis

Motivation



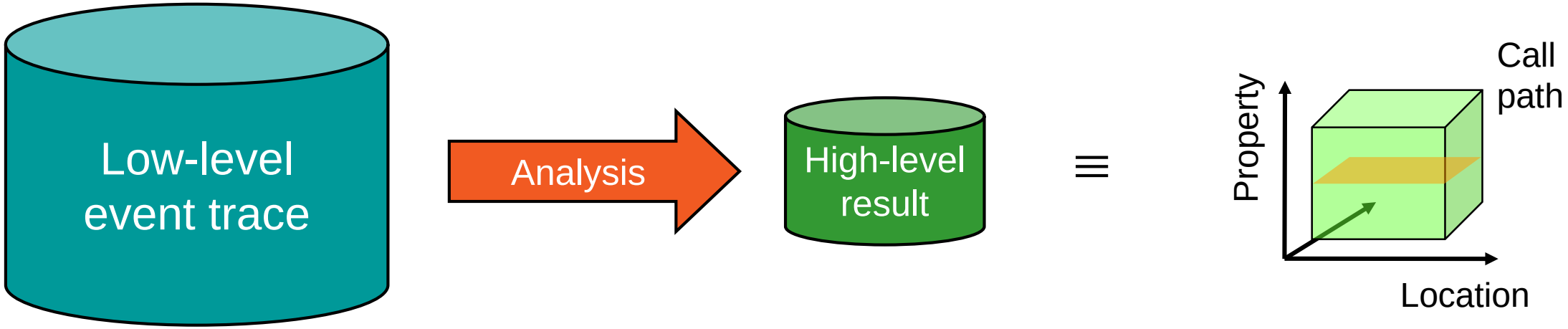
While interactive trace visualization is often intuitive…



…finding bottlenecks may become the search for needles in a haystack…

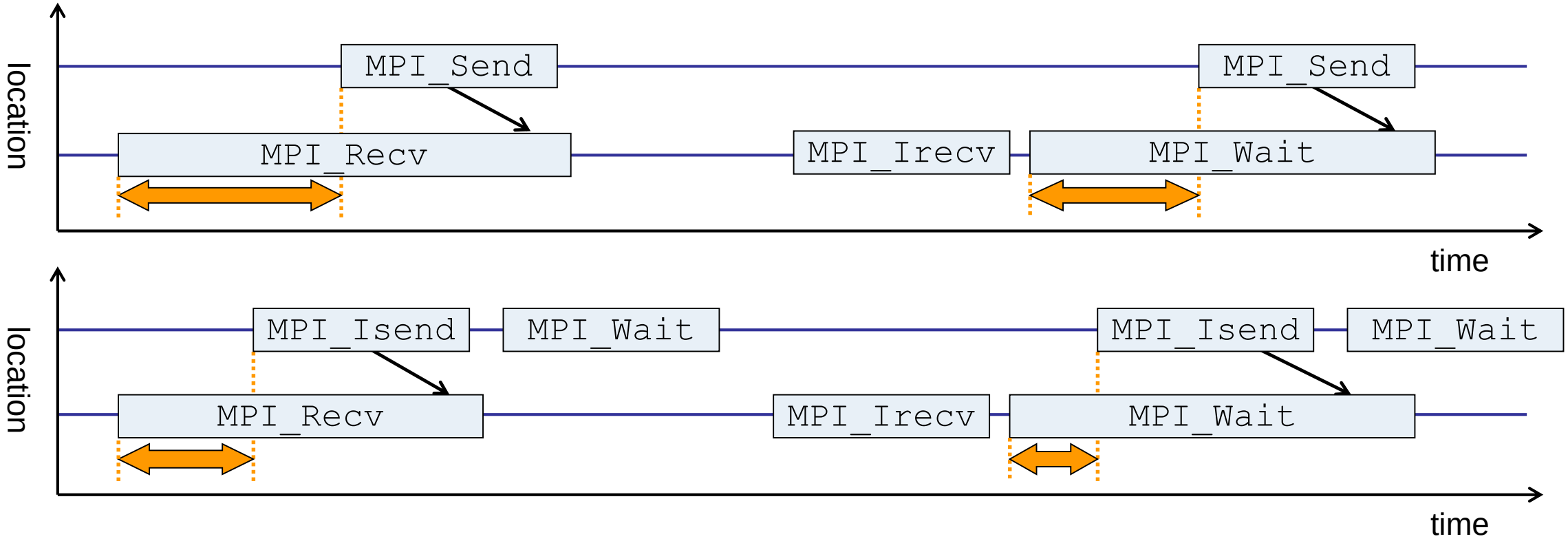
Idea: Automatic trace analysis

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



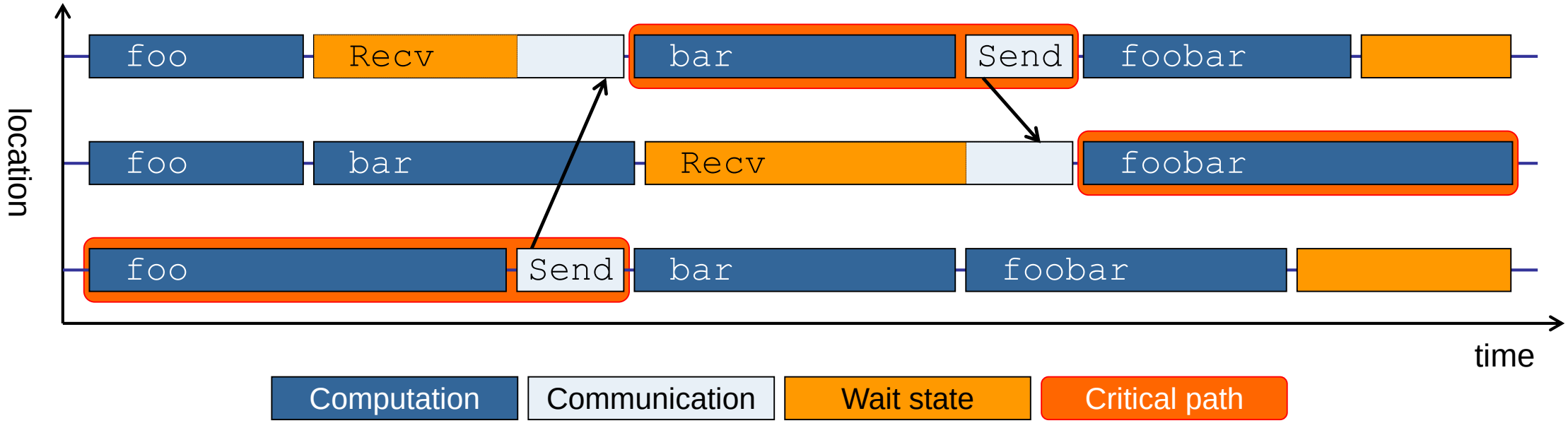
- Guaranteed to cover the entire event trace
- Quicker than manual/visual trace analysis
- Parallel replay analysis exploits available memory & processors to deliver scalability

Example: "*Late Sender*" wait state



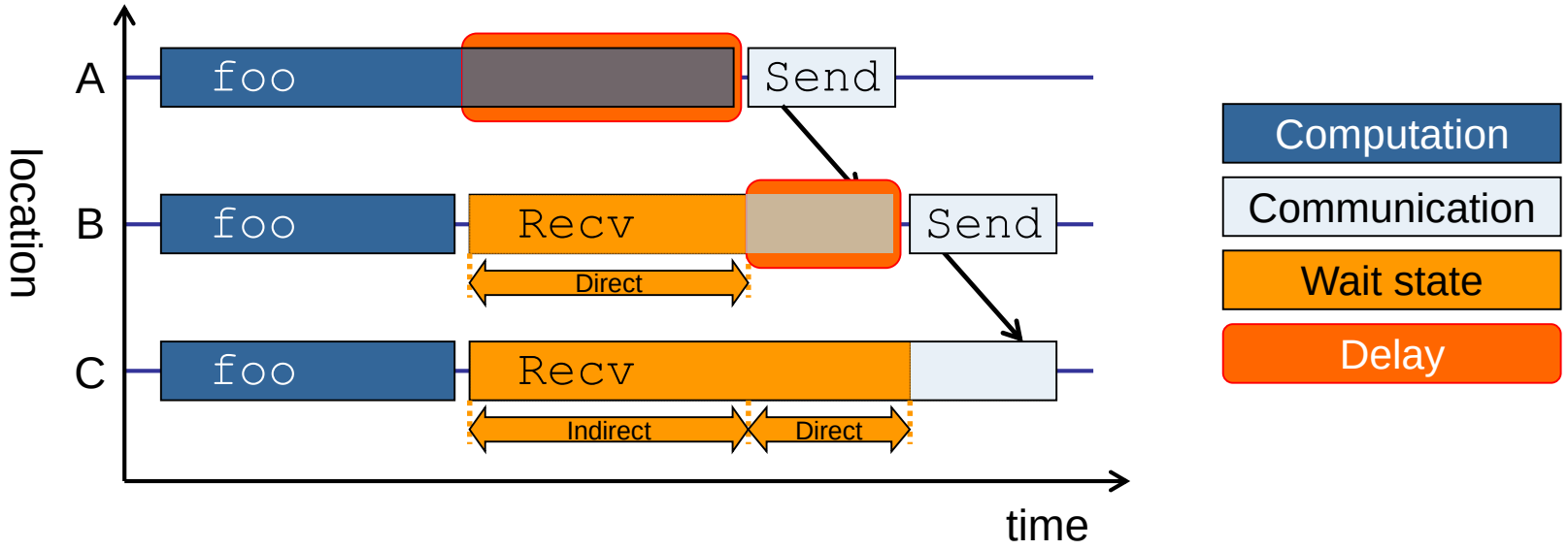
- Waiting time caused by a blocking receive operation posted earlier than the corresponding send
- Applies to blocking as well as non-blocking communication

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*

Scalasca command – One command for (almost) everything

```
% scalasca
Scalasca 2.6.1
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 convenience command: scan / scalasca -analyze

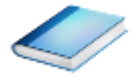
```
% scan
Scalasca 2.6.1: measurement collection & analysis nexus
usage: scan {options} [launchcmd [launchargs]] target [targetargs]
       where {options} may include:
  -h    Help       : show this brief usage message and exit.
  -v    Verbose    : increase verbosity.
  -n    Preview    : show command(s) 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

Automatic measurement configuration

- scan configures Score-P measurement by automatically setting some environment variables and exporting them
  - E.g., experiment title, profiling/tracing mode, filter file, …
  - Precedence order:
    - Command-line arguments
    - Environment variables already set
    - Automatically determined values
- Also, scan includes consistency checks and prevents corrupting existing experiment directories
- For tracing experiments, after trace collection completes then automatic parallel trace analysis is initiated
  - Uses identical launch configuration to that used for measurement (i.e., the same allocated compute resources)

Scalasca advanced command: scout - Scalasca automatic trace analyzer



```
% scout.hyb --help
SCOUT   (Scalasca 2.6.1)
Copyright(c) 1998-2022 Forschungszentrum Juelich GmbH
Copyright(c) 2014-2021 RWTH Aachen University
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 convenience command: square / scalasca -examine

```
% square
Scalasca 2.6.1: 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)

Toolchain, Score-P, and Scalasca modules (DINE)

- Select modules for the Intel + IntelMPI tool chain

```
% module load  intel_comp/2020-update2 intel_mpi/2020-update2
```

- Load Score-P, Scalasca and Cube modules
  - Score-P & Scalasca installations are toolchain specific!

```
% module load  scorep/8.4 scalasca/2.6.1 cube/4.8.2
```

BT-MZ summary measurement collection...

```
% cd bin.scorep
% cp ../jobscript/dine/scan.sbatch .
% cat scan.sbatch
...
# set up environment
module purge
module load intel_comp/2020-update2 intel_mpi/2020-update2
module load scalasca/2.6.1 scorep/8.4

# measurement configuration
export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_TOTAL_MEMORY=100M
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC,...
#export SCAN_ANALYZE_OPTS="—time-correct"

set -x
export OMP_NUM_THREADS=6
scan -s mpiexec -np 8 ./bt-mz_C.8
```

- Change to directory with the Score-P instrumented executable and edit the job script

Hint:
scan = scalasca -analyze
-s = profile/summary (def)

```
% sbatch scan.sbatch
```

- Submit the job

BT-MZ summary measurement

```
S=C=A=N: Scalasca 2.6.1 runtime summarization
S=C=A=N: ./scorep_bt-mz_C_8x6_sum experiment archive
S=C=A=N: Sat Apr 20 10:11:19 2024: Collect start
mpiexec ./bt-mz_C.8

 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) -
    BT-MZ MPI+OpenMP Benchmark

 Number of zones:  16 x  16
 Iterations: 200    dt:   0.000100
 Number of active processes:     8

 [... More application output ...]

S=C=A=N: Sat Apr 20 10:11:39 2024: Collect done (status=0) 20s
S=C=A=N: ./scorep_bt-mz_C_8x6_sum complete.
```

- Run the application using the Scalasca measurement collection & analysis nexus prefixed to launch command
- Creates experiment directory:
  scorep_bt-mz_C_8x6_sum

BT-MZ summary analysis report examination

- Score summary analysis report

```
% square -s  scorep_bt-mz_C_8x6_sum
INFO: Post-processing runtime summarization report (profile.cubex)...
INFO: Score report written to ./scorep_bt-mz_C_8x6_sum/scorep.score
```

- Post-processing and interactive exploration with Cube

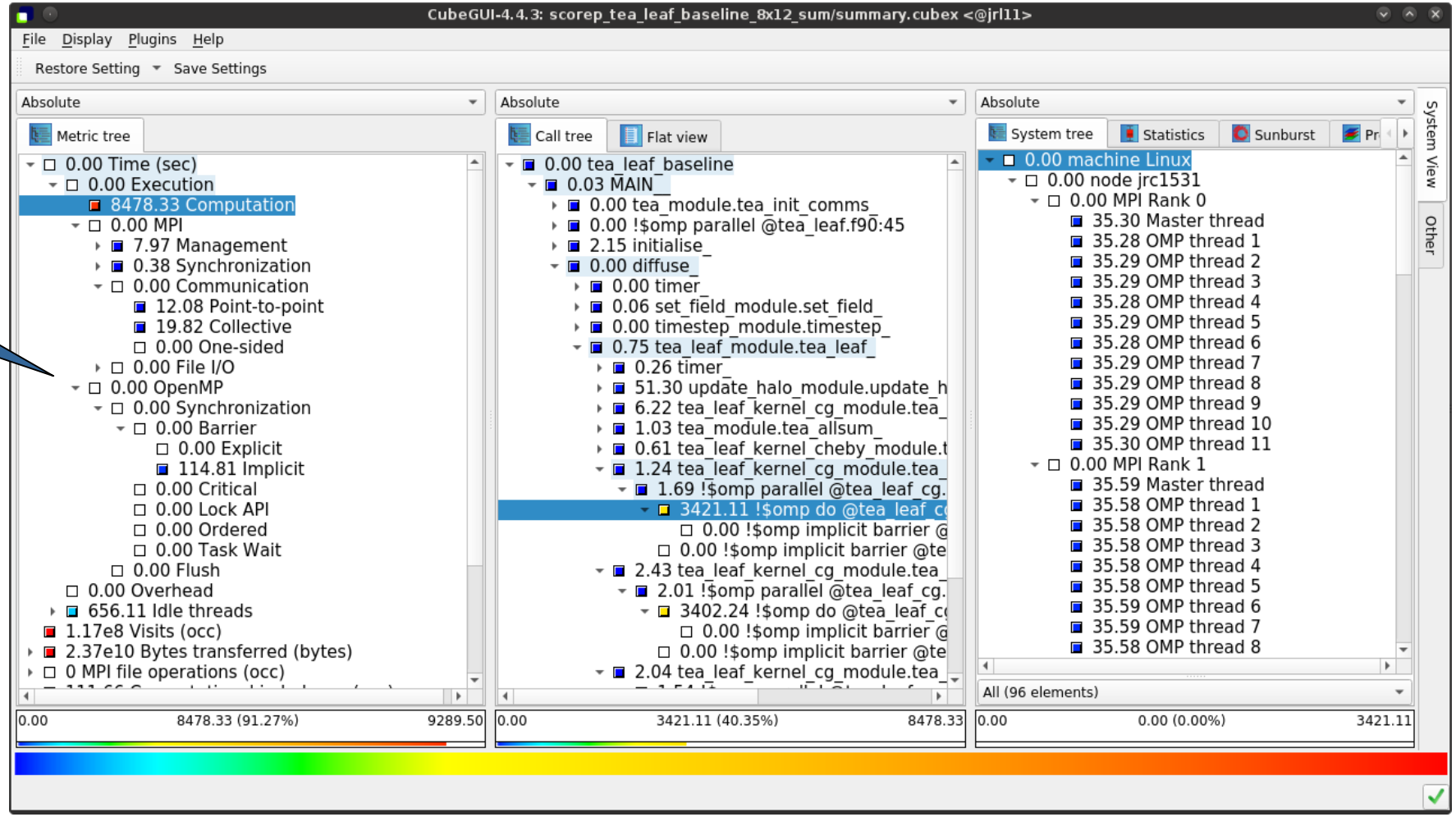
```
% square  scorep_bt-mz_C_8x6_sum
INFO: Displaying ./scorep_bt-mz_C_8x6_sum/summary.cubex...
                    [GUI showing summary analysis report]
```

Hint:
Copy ‘summary.cubex’ to local system (laptop) using ‘scp’ to improve responsiveness of GUI

- The post-processing derives additional metrics and generates a structured metric hierarchy

Post-processed summary analysis report

Split base metrics into more specific metrics



Performance analysis steps

- 0.0 Reference preparation for validation

- 1.0 Program instrumentation
- 1.1 Summary measurement collection
- 1.2 Summary analysis report examination

- 2.0 Summary experiment scoring
- 2.1 Summary measurement collection with filtering
- 2.2 Filtered summary analysis report examination

- 3.0 Event trace collection
- 3.1 Event trace analysis & report examination

BT-MZ trace measurement collection...

```
% cd bin.scorep
% cp ../jobscript/dine/scan.sbatch .
% vim scan.sbatch
...
# set up environment
module purge
module load intel_comp/2020-update2 intel_mpi/2020-update2
module load scalasca/2.6.1 scorep/8.4

# measurement configuration
export SCOREP_FILTERING_FILE=../config/scorep.filt
export SCOREP_TOTAL_MEMORY=100M
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC,...
export SCAN_ANALYZE_OPTS="—time-correct"

set -x
export OMP_NUM_THREADS=6
scan -t mpiexec -np 8 ./bt-mz_C.8
```

```
% sbatch scan.sbatch
```

- Change to directory with the Score-P instrumented executable and edit the job script
- Use "-t" with the `scan` command
- Submit the job

BT-MZ trace measurement … collection

```
S=C=A=N: Scalasca 2.6.1 trace collection and analysis
S=C=A=N: Sat Apr 20 10:16:41 2024: Collect start
mpiexec  ./bt-mz_C.8

 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP \
>Benchmark

 Number of zones:  16 x  16
 Iterations: 200    dt:   0.000100
 Number of active processes:     8

 [... More application output ...]

S=C=A=N: Sat Apr 20 10:17:03 2024: Collect done (status=0) 22s
```

- Starts measurement with collection of trace files …

BT-MZ trace measurement … analysis

```
…
S=C=A=N: Sat Apr 20 10:17:03 2024: Analyze start
mpiexec scout.hyb ./scorep_bt-mz_C_8x6_trace/traces.otf2

SCOUT   (Scalasca 2.6.1)

Analyzing experiment archive ./scorep_bt-mz_C_8x6_trace/traces.otf2

Opening experiment archive ... done (0.007s).
Reading definition data    ... done (0.009s).
Reading event trace data   ... done (0.209s).
Preprocessing              ... done (0.565s).
Timestamp correction       ... done (1.316s).
Analyzing trace data       ... done (15.125s).
Writing analysis report    ... done (0.227s).

Max. memory usage         : 996.031MB

Total processing time     : 17.508s
S=C=A=N: Sat Apr 20 10:17:22 2024: Analyze done (status=0) 19s
```

- Continues with automatic (parallel) analysis of trace files

BT-MZ trace analysis report exploration

- Produces trace analysis report in the experiment directory containing trace-based wait-state metrics

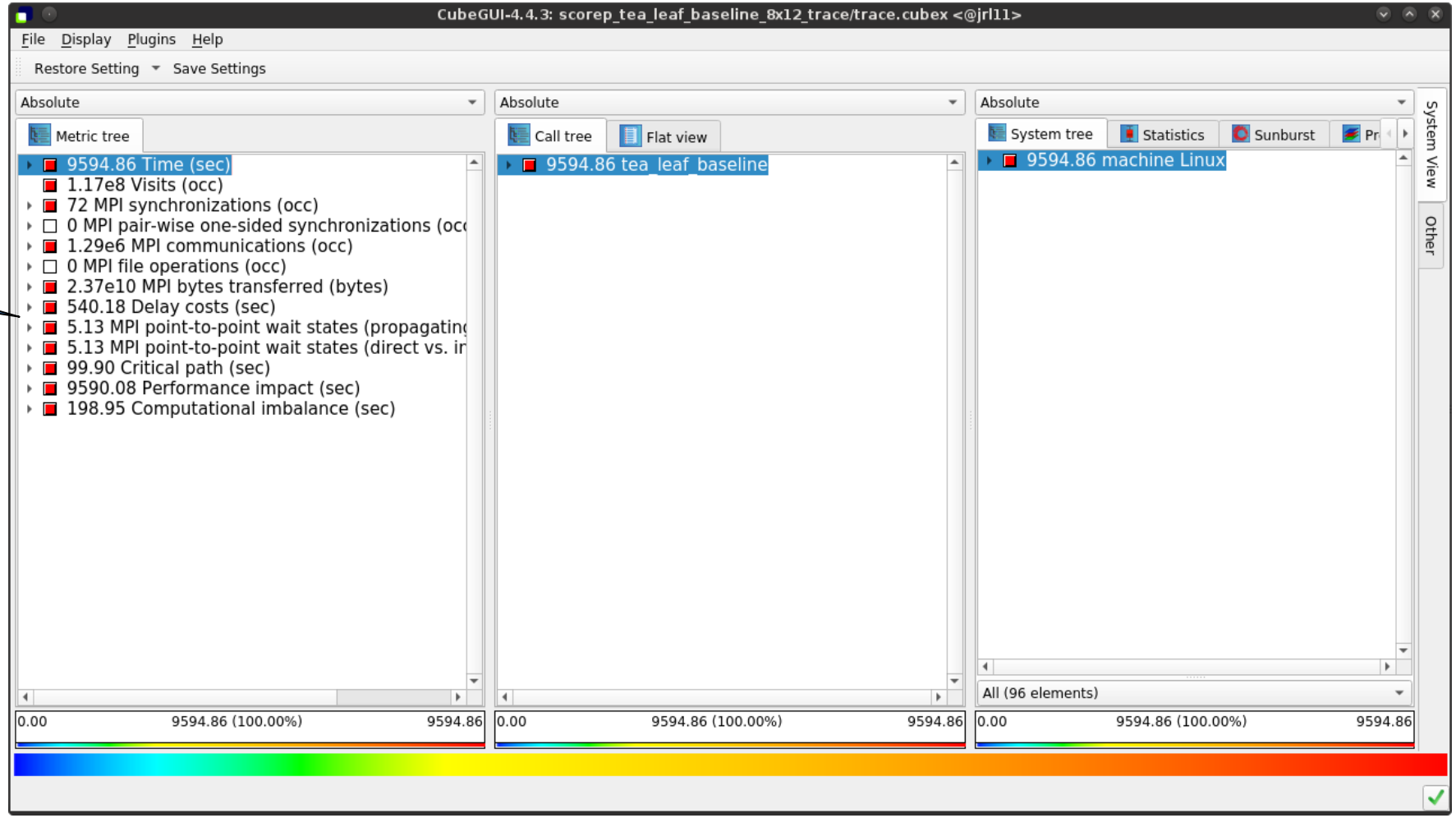
```
% square  scorep_bt-mz_C_8x6_trace
INFO: Post-processing runtime summarization report (profile.cubex)...
INFO: Post-processing trace analysis report (scout.cubex)...
INFO: Displaying ./scorep_bt-mz_C_8x6_trace/trace.cubex...

                [GUI showing trace analysis report]
```

Hint:
Run ‘square -s’ first and then copy ‘trace.cubex’ to local system (laptop) using ‘scp’ to improve responsiveness of GUI

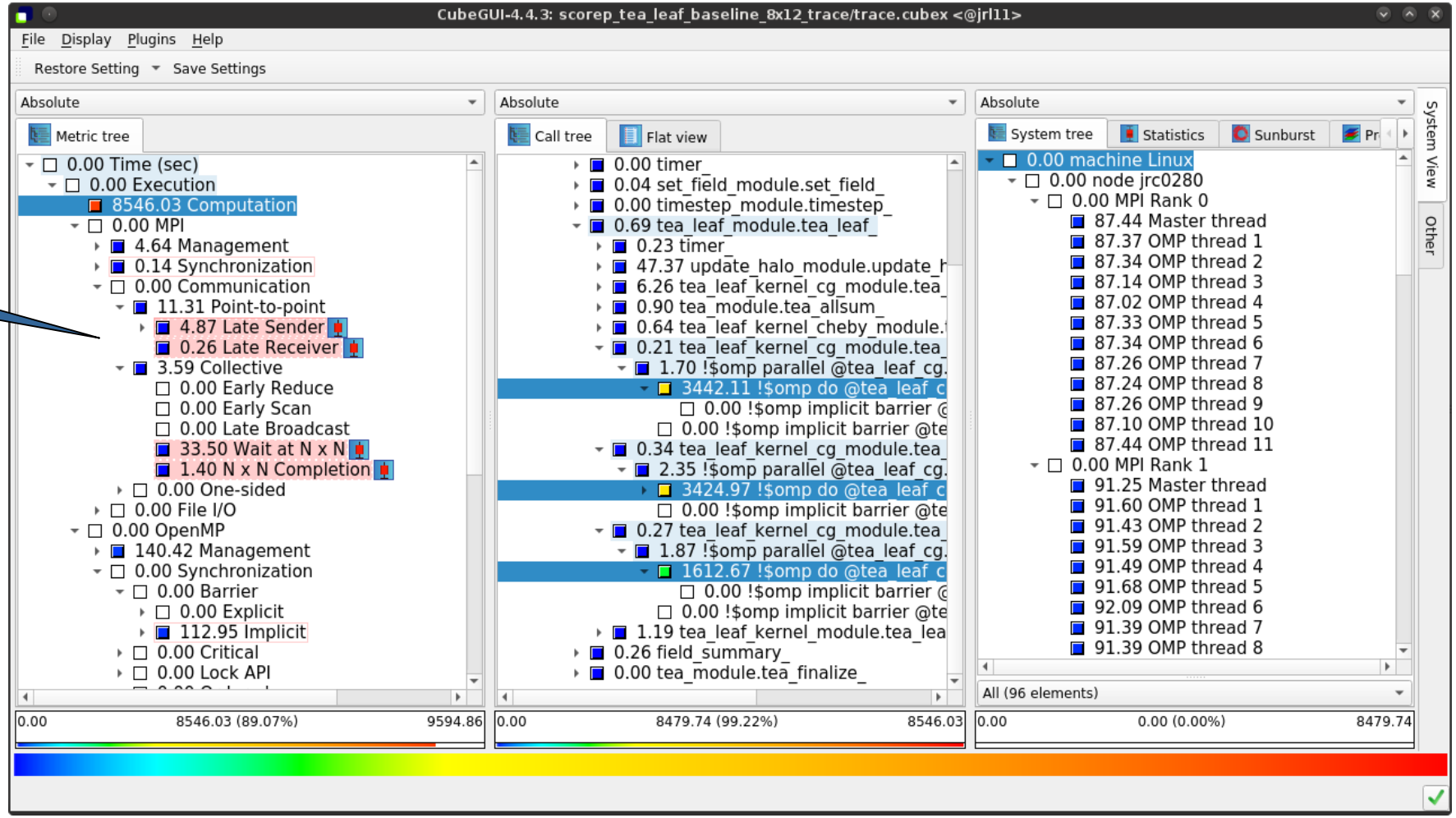
Scalasca analysis report exploration (opening view)

Additional top-level metrics produced by the trace analysis…

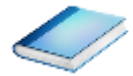


Scalasca wait-state metrics

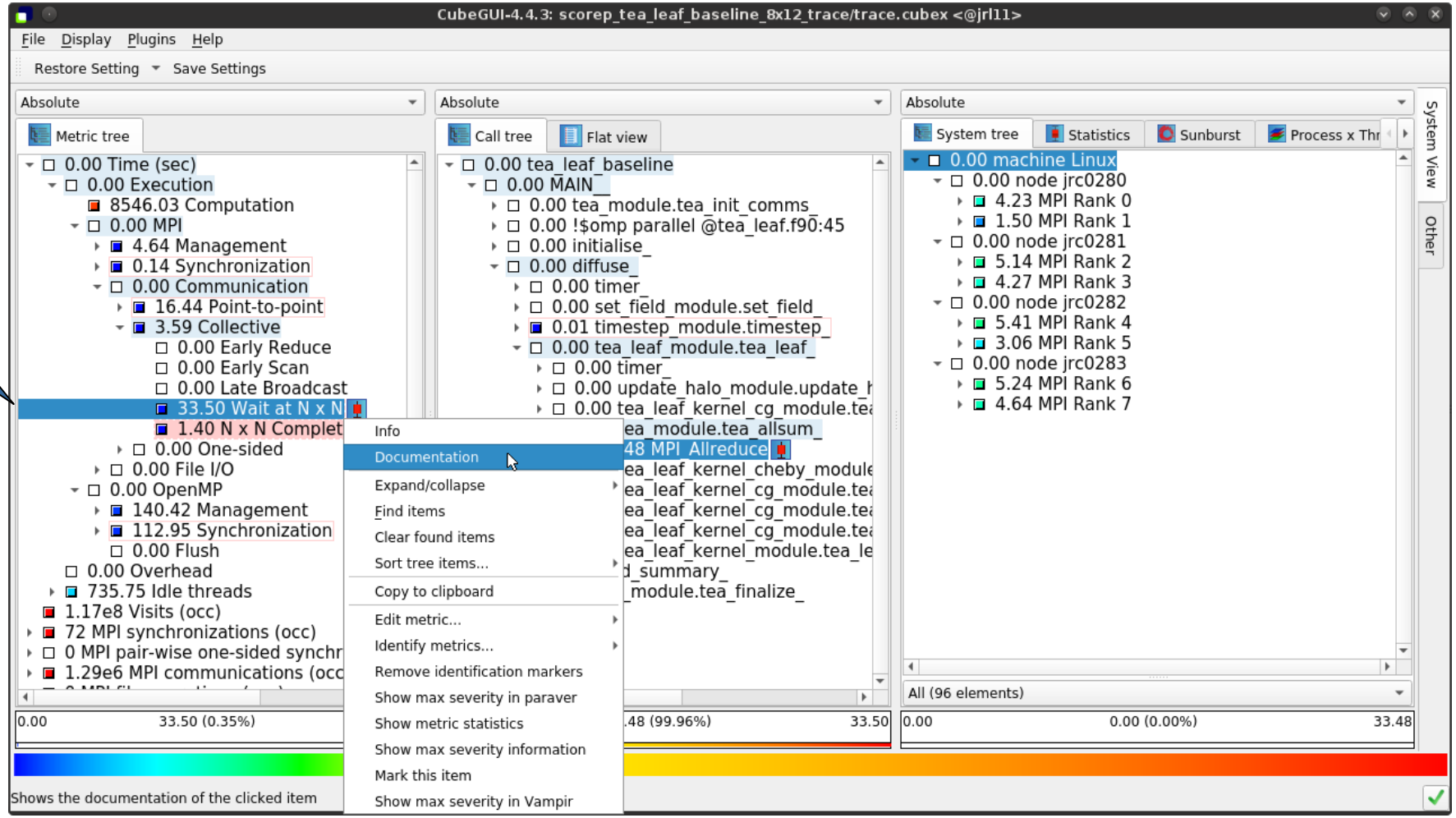
…plus additional wait-state metrics as part of the “Time” hierarchy



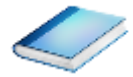
Online metric description



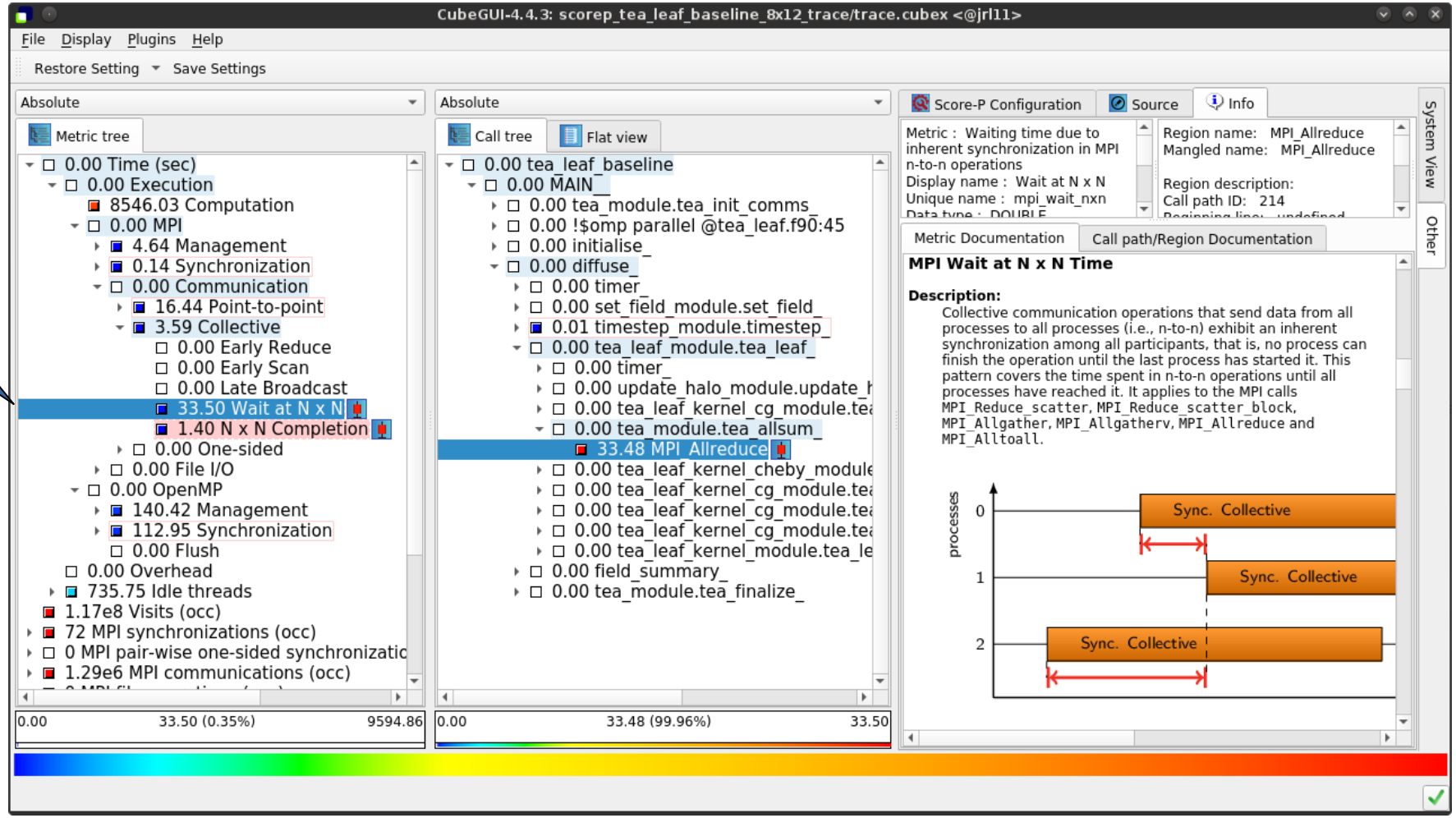
Access online metric description via context menu (right-click)



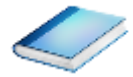
Online metric description (cont.)



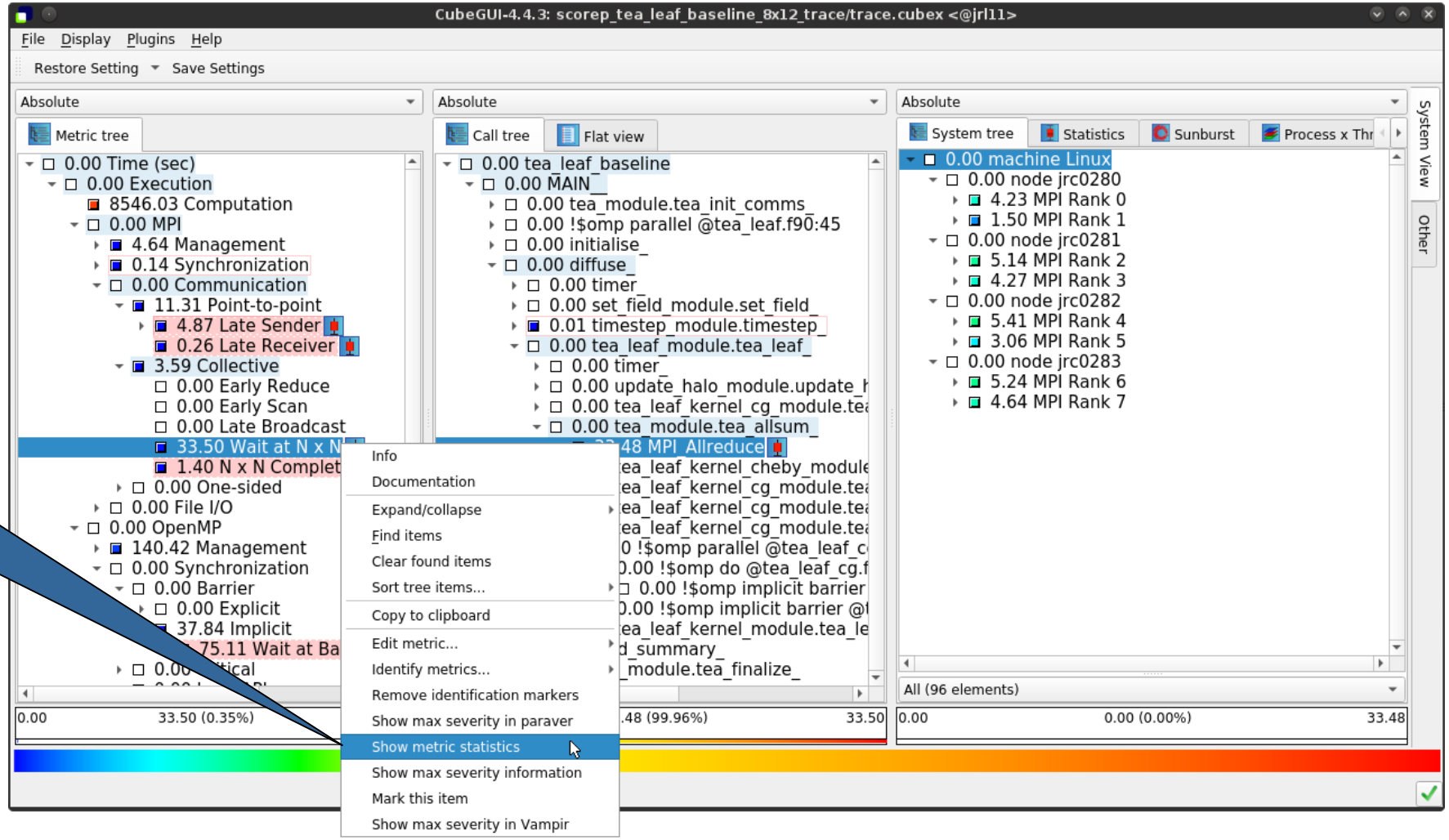
Selection of different metric automatically updates description



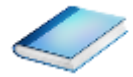
Metric statistics



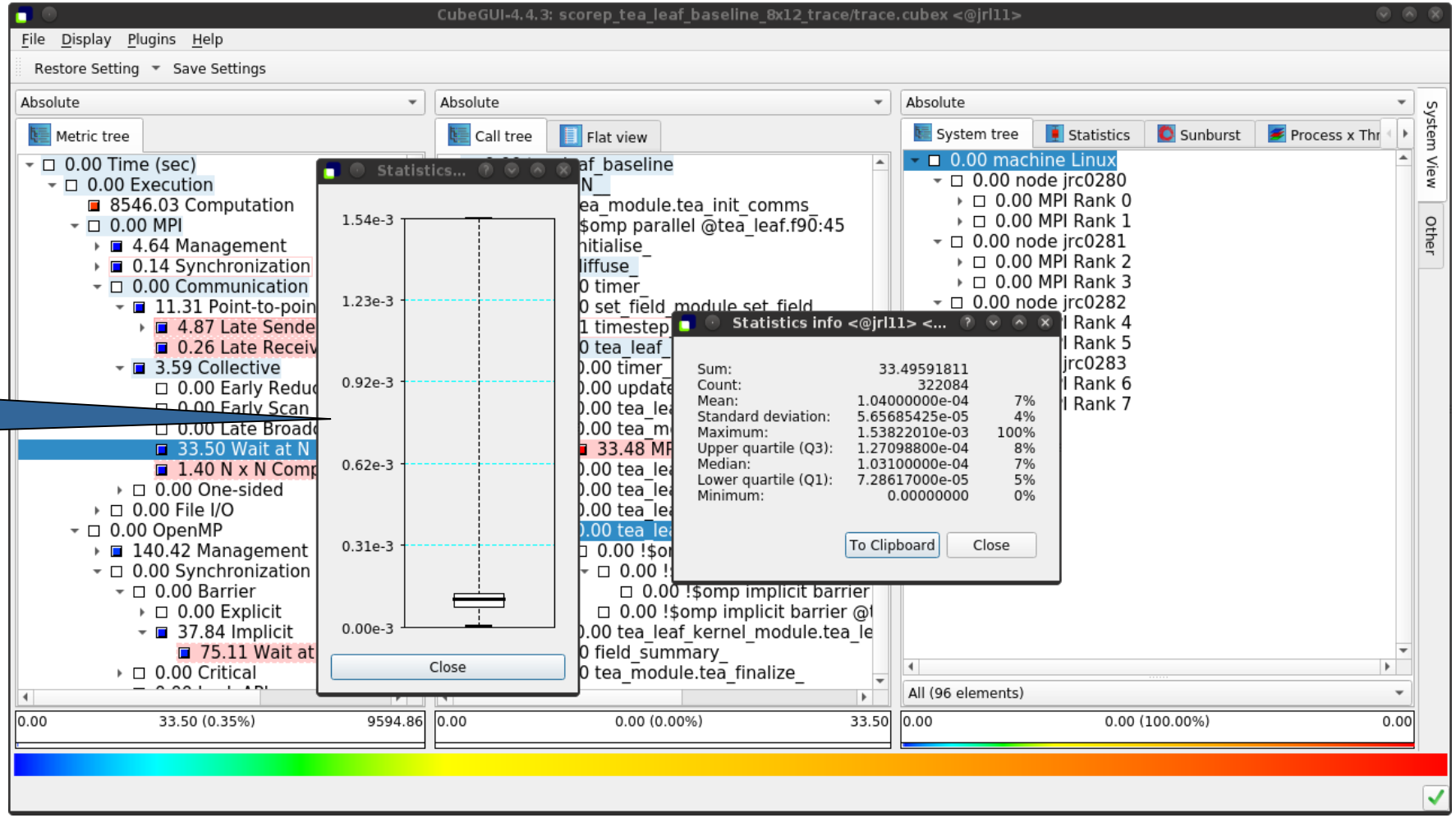
Access metric statistics for metrics marked with box plot icon from context menu



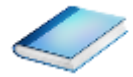
Metric statistics (cont.)



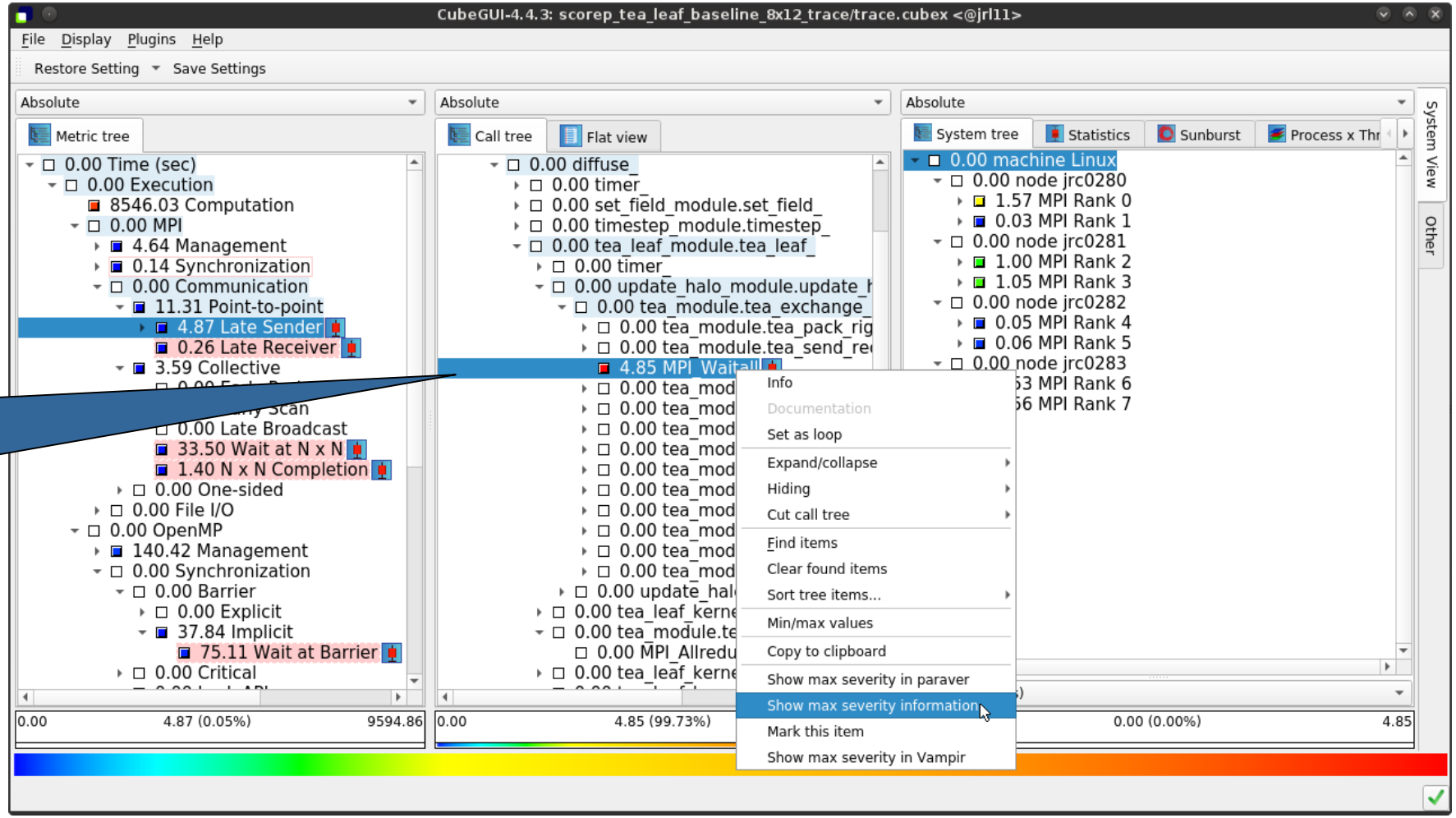
Shows instance statistics box plot, click to get details



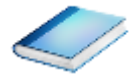
Metric instance statistics



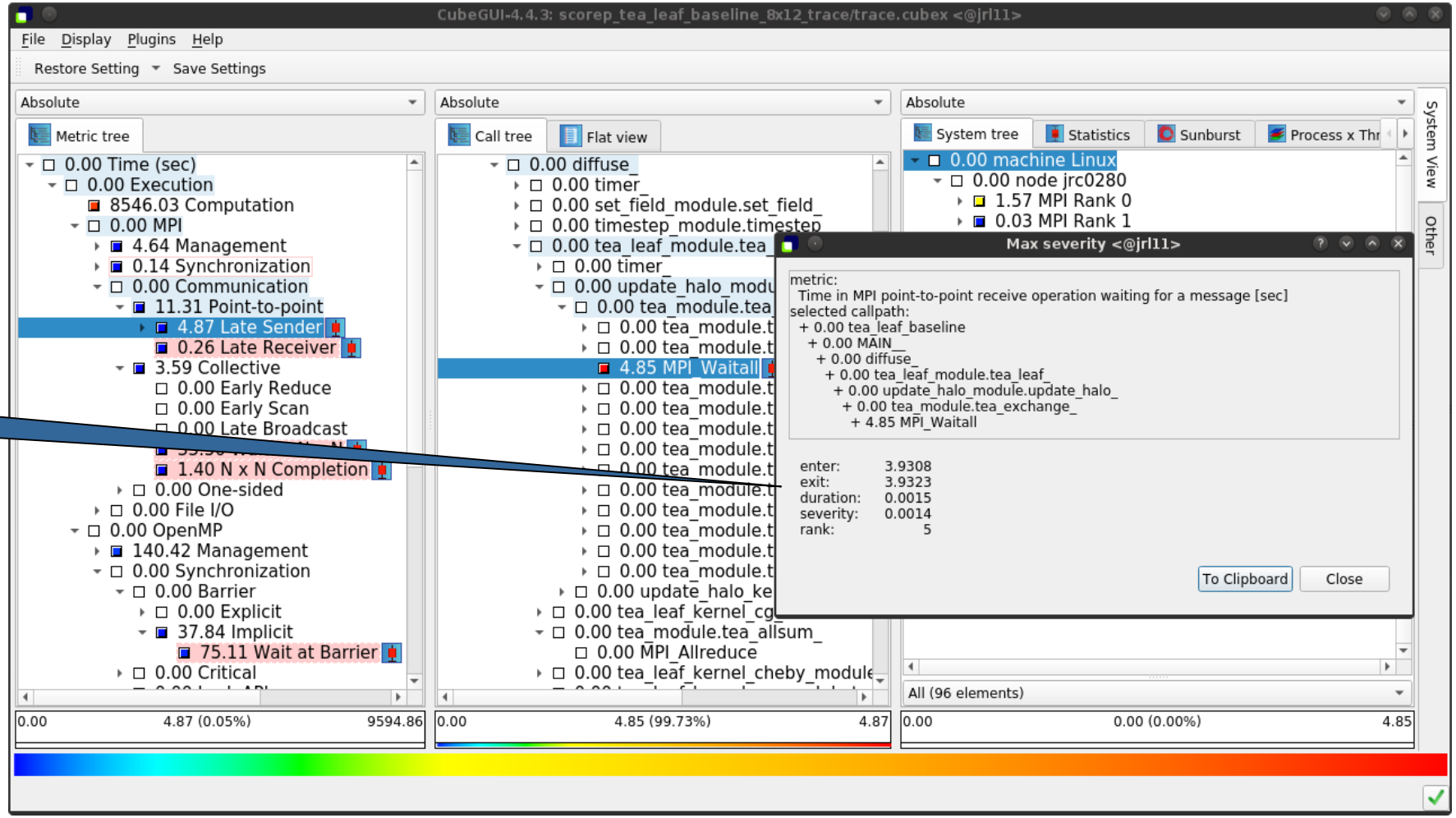
Access most-severe instance information for call paths marked with box plot icon via context menu



Metric instance statistics (cont.)



Shows instance details



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:
  - https://www.scalasca.org
- Contact:
  - mailto: scalasca@fz-juelich.de

trace tools scalasca