

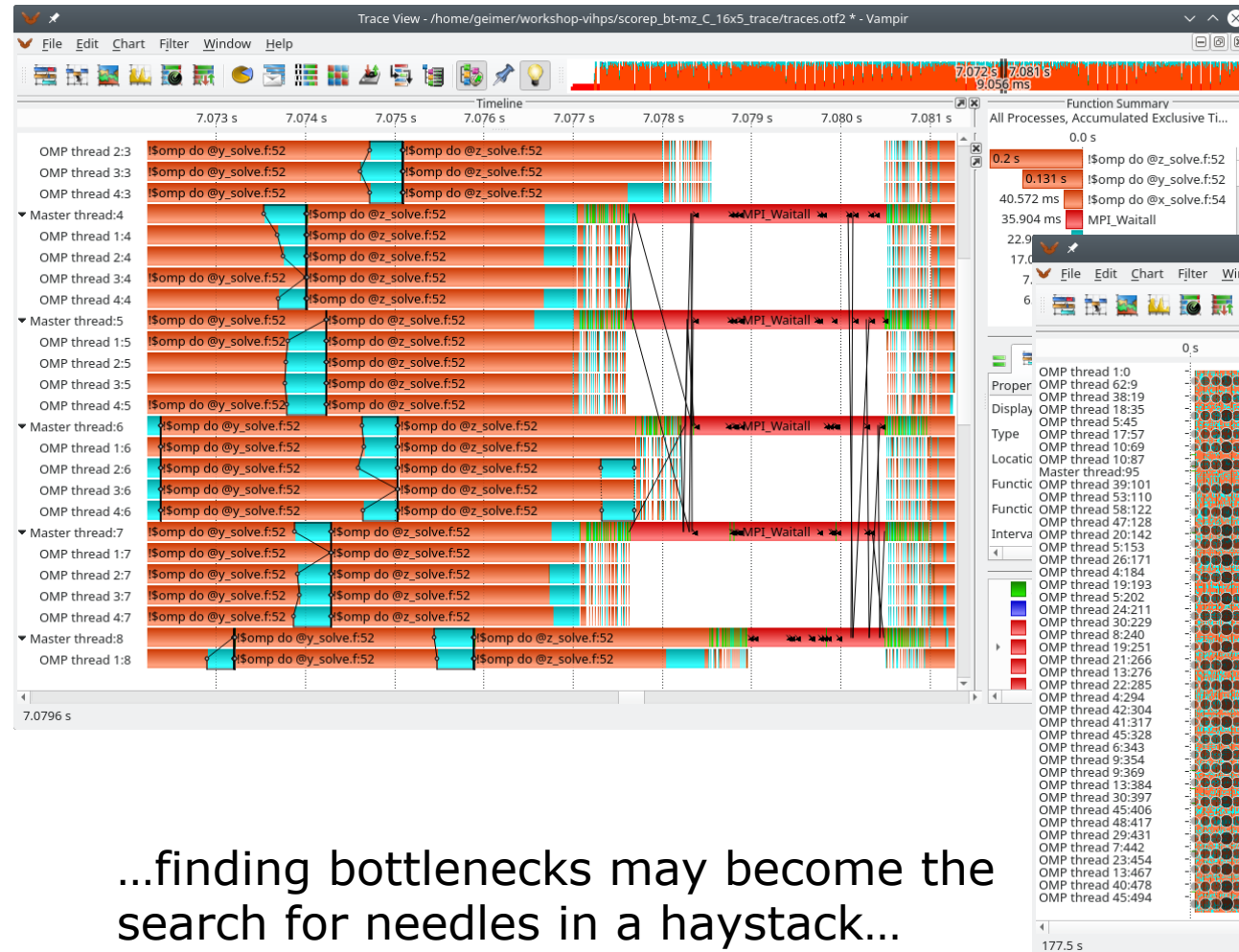
Scalasca Trace Tools

Demo/Hands-on: Automatic trace analysis

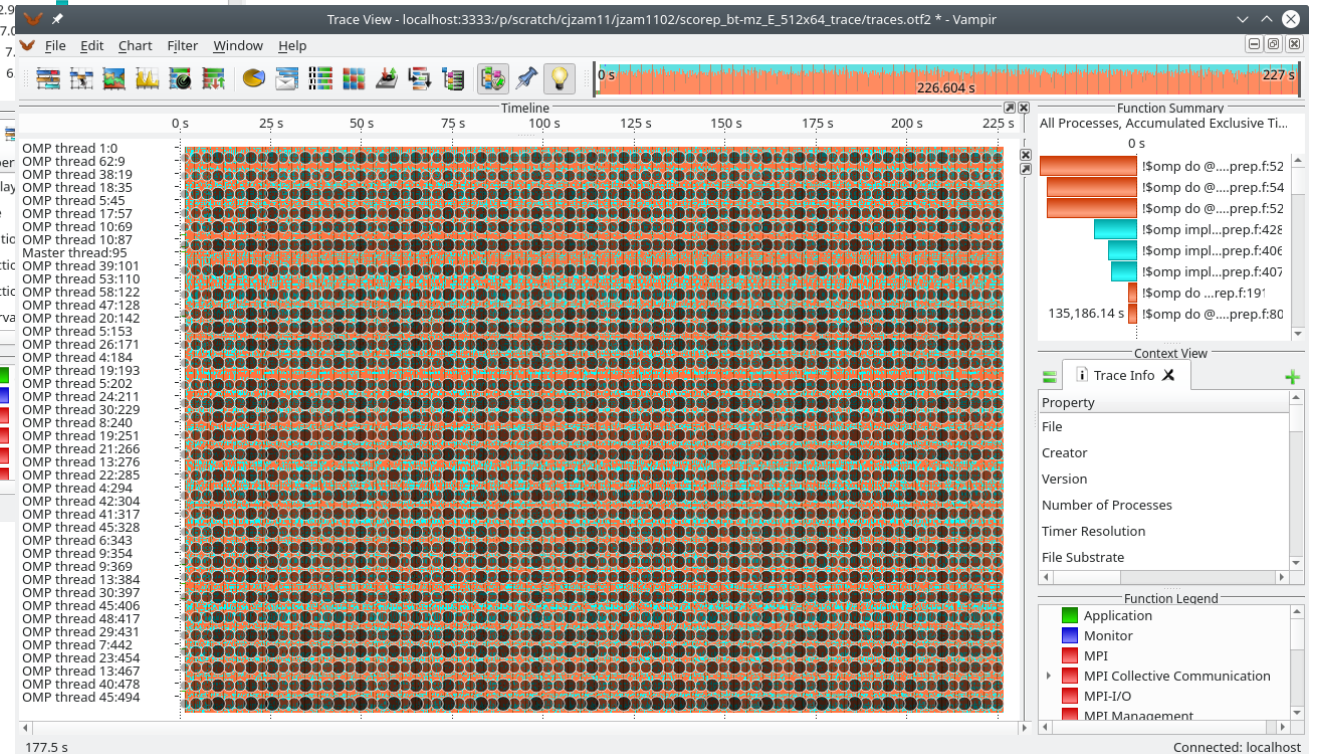
Markus Geimer
Jülich Supercomputing Centre



Motivation



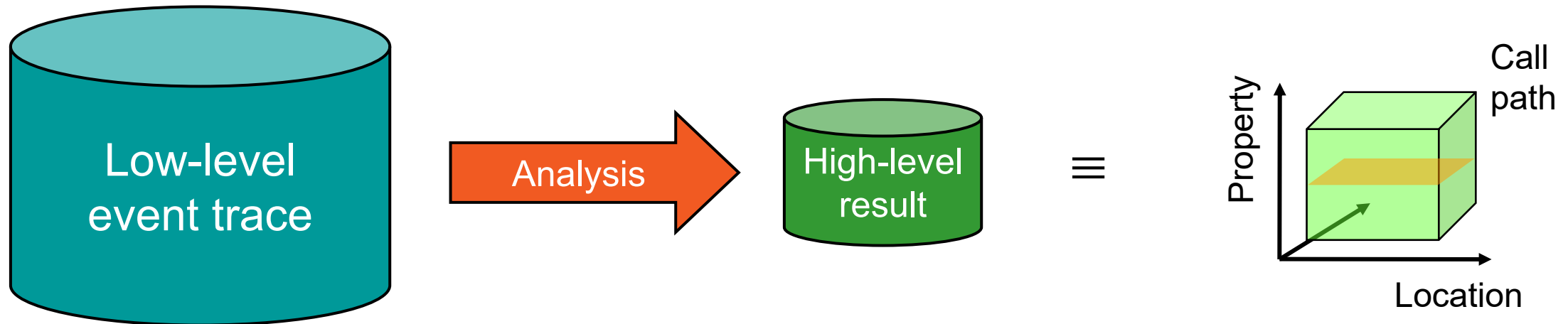
While interactive trace visualization is often intuitive...



...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



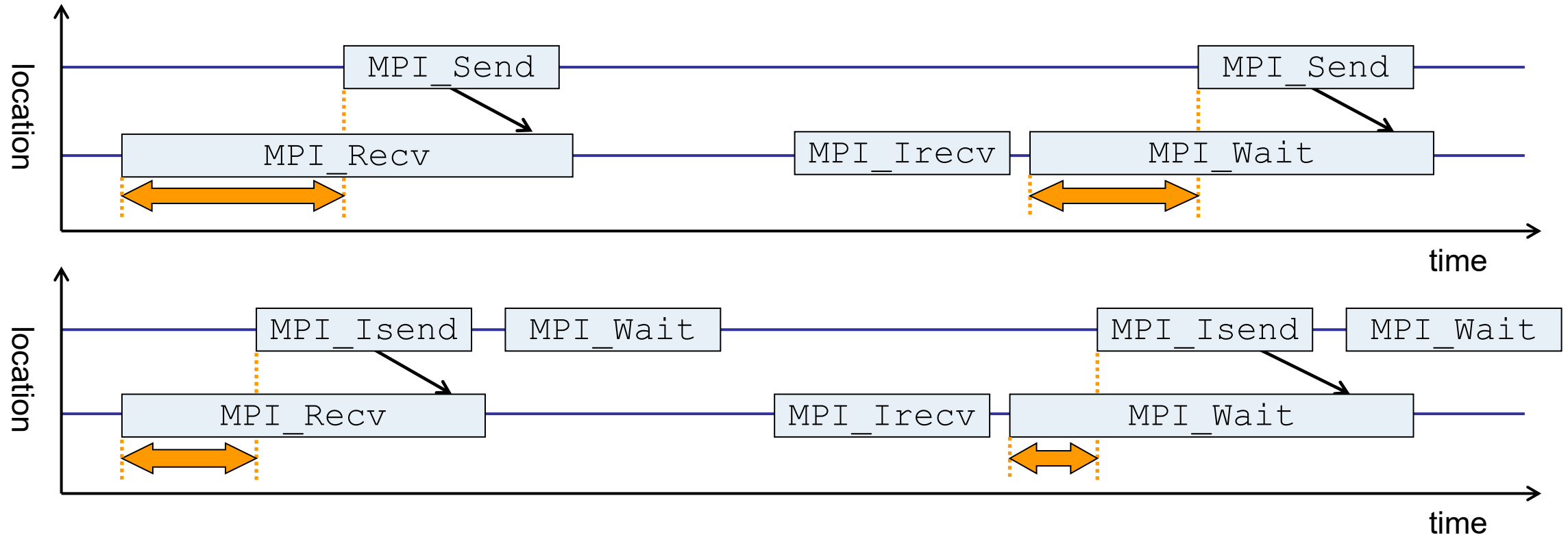
- Guaranteed to cover the entire event trace
- Quicker than manual/visual trace analysis
- Parallel replay analysis exploits available memory & processors to deliver scalability

Scalasca Trace Tools

DOI [10.5281/zenodo.4103922](https://doi.org/10.5281/zenodo.4103922)

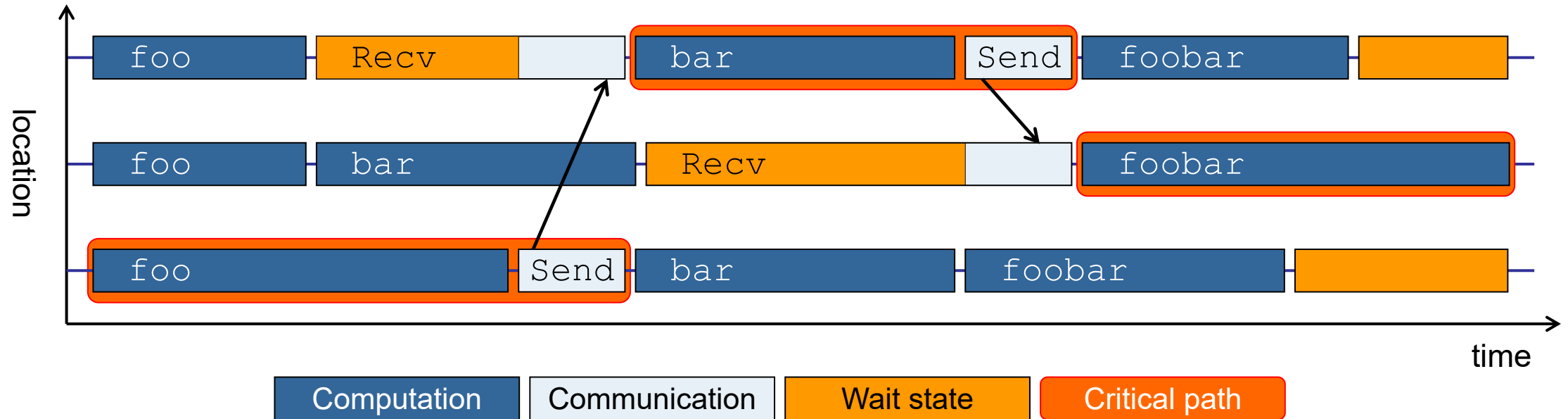
- **Scalable trace-based** performance analysis toolset for the most popular parallel programming paradigms
 - Current focus: MPI, OpenMP, and (to a limited extent) POSIX threads
 - Analysis of traces including *only host-side events* from applications using CUDA, ROCm, OpenCL, or OpenACC (also in combination with MPI and/or OpenMP) is possible, but results need to be interpreted with some care
- Specifically targeting large-scale parallel applications
 - Demonstrated scalability up to 1.8 million parallel threads
 - Of course also works at small/medium scale
- Latest release:
 - Scalasca Trace Tools v2.6.2 (April 2025)

Example: “Late Sender” wait state



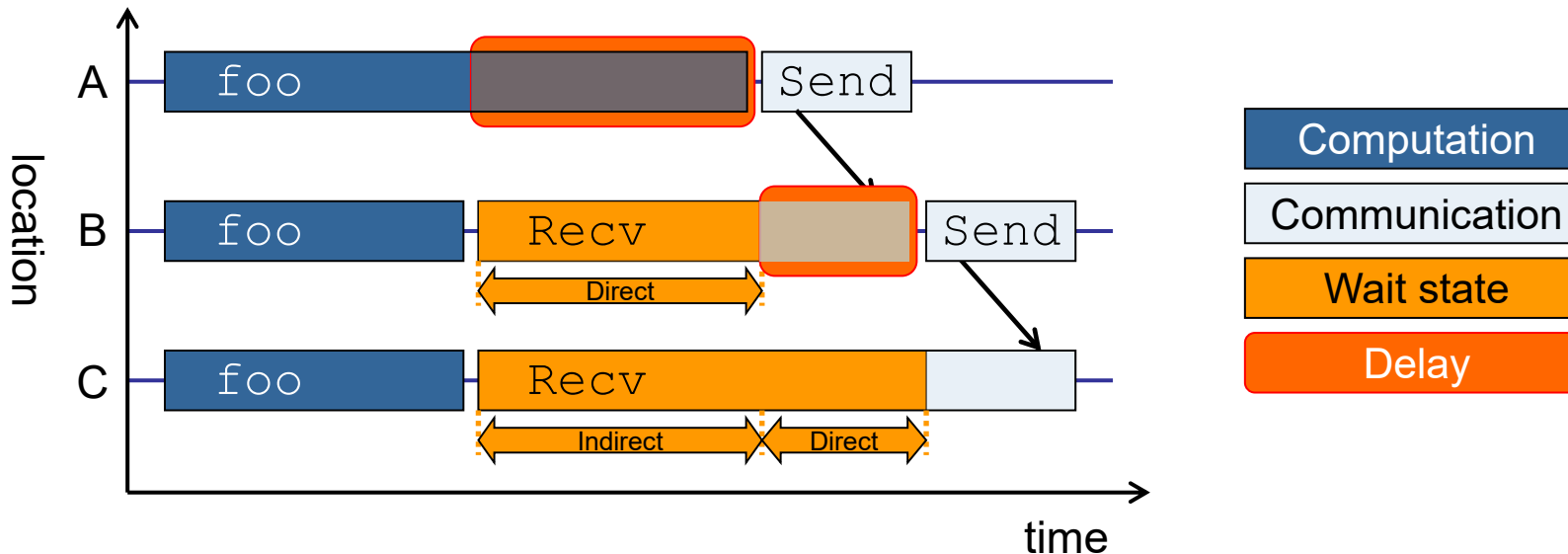
- Waiting time caused by a blocking receive operation posted earlier than the corresponding send
- Applies to blocking as well as non-blocking communication

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 2.6.2
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, use Score-P instrumenter directly!

Scalasca convenience command: scan / scalasca -analyze

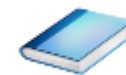
```
% scan
Scalasca 2.6.2: measurement collection & analysis nexus
usage: scan {options} [launchcmd [launchargs]] target [targetargs]
      where {options} may include:
-h      Help          : show this brief usage message and exit.
-v      Verbose       : increase verbosity.
-n      Preview       : 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.
-P      preset        : Specify a preset for a multi-run measurement, e.g., 'pop'.
-L      : List available multi-run presets.
-D      cfgfile       : Check a multi-run config file for validity and dump...
```

▪ 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 2.6.2)
Copyright(c) 1998-2025 Forschungszentrum Juelich GmbH
Copyright(c) 2014-2022 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 2.6.2: 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
  -I                       : Ignore structural sanity checks and force aggregation
                          of measurements in a multi-run experiment
  -x <scorep-score opt>   : Pass options to scorep-score
```

▪ Scalasca analysis report explorer (Cube)

Compiler, MPI, Score-P and Scalasca modules

- Select modules for the GNU + Open MPI tool chain

```
% module load gnu_comp/14.1.0 openmpi/5.0.3  
% module load scorep/9.4 scalasca/2.6.2 cube/4.9.1
```

- Copy tutorial sources to your WORK directory

```
% export WORK=/cosma/apps/do009/$USER  
% cd $WORK  
% tar xf /cosma/apps/do009/shared/Scalasca/NPB3.4-MZ-MPI.tar.gz  
% cd NPB3.4-MZ-MPI
```

BT-MZ summary measurement collection...

```
% cd bin.scorep
% cp ../jobscript/bluefield1/scan.sbatch .
% cat scan.sbatch
...
# set up environment
module purge
module load gnu_comp openmpi
module load scalasca

# measurement configuration
export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_TOTAL_MEMORY=100M
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC,...

set -x
export OMP_NUM_THREADS=6
scan -s mpiexec ./bt-mz_B.x
```

```
% sbatch scan.sbatch
```

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

Hint:

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

- Submit the job

BT-MZ summary measurement

```
S=C=A=N: Scalasca 2.6.2 runtime summarization
S=C=A=N: ./scorep_bt-mz_B_4x6_sum experiment archive
S=C=A=N: Wed Apr 22 10:36:53 2026: Collect start
mpiexec ./bt-mz_B.x

  NAS Parallel Benchmarks (NPB3.4-MZ MPI+OpenMP) - \
>   BT-MZ Benchmark

Number of zones:      8 x      8
Iterations: 200      dt:    0.000300
Number of active processes:      4

[... More application output ...]

S=C=A=N: Wed Apr 22 10:37:28 2026: Collect done (status=0) 35s
S=C=A=N: ./scorep_bt-mz_B_4x6_sum complete.
```

- Run the application using the Scalasca measurement collection & analysis nexus prefixed to launch command
- Creates experiment directory:
scorep_bt-mz_B_4x6_sum

BT-MZ summary analysis report examination

- Score summary analysis report

```
% square -s scorep_bt-mz_B_4x6_sum  
INFO: Post-processing runtime summarization report (profile.cubex)...  
INFO: Score report written to ./scorep_bt-mz_B_4x6_sum/scorep.score
```

- Post-processing and interactive exploration with Cube

```
% square scorep_bt-mz_B_4x6_sum  
INFO: Displaying ./scorep_bt-mz_B_4x6_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

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/bluefield1/scan.sbatch .
% vim scan.sbatch
...
# set up environment
module purge
module load gnu_comp openmpi
module load scalasca

# measurement configuration
export SCOREP_FILTERING_FILE=../config/scorep.filt
export SCOREP_TOTAL_MEMORY=100M
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC,...

set -x
export OMP_NUM_THREADS=6
scan -t mpiexec ./bt-mz_B.x
```

```
% 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 2.6.2 trace collection and analysis
S=C=A=N: Wed Apr 22 13:23:26 2026: Collect start
mpiexec ./bt-mz_B.x

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

Number of zones:    8 x    8
Iterations: 200      dt:    0.000300
Number of active processes:    4

[... More application output ...]

S=C=A=N: Wed Apr 22 13:24:01 2026: Collect done (status=0) 35s
```

- Starts measurement with collection of trace files ...

BT-MZ trace measurement ... analysis

```
...
S=C=A=N: Wed Apr 22 13:26:01 2026: Analyze start
mpiexec scout.hyb ./scorep_bt-mz_B_4x6_trace/traces.otf2

SCOUT (Scalasca 2.6.2)

Analyzing experiment archive ./scorep_bt-mz_B_4x6_trace/traces.otf2

Opening experiment archive ... done (0.004s).
Reading definition data ... done (0.002s).
Reading event trace data ... done (0.651s).
Preprocessing ... done (3.554s).
Analyzing trace data ... done (59.708s).
Writing analysis report ... done (0.251s).

Max. memory usage : 486.344MB

Total processing time : 64.272s
S=C=A=N: Wed Apr 22 13:25:06 2026: Analyze done (status=0) 65s
```

- 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_B_4x6_trace
INFO: Post-processing runtime summarization report (profile.cubex)...
INFO: Post-processing trace analysis report (scout.cubex)...
INFO: Displaying ./scorep_bt-mz_B_4x6_trace/trace.cubex...

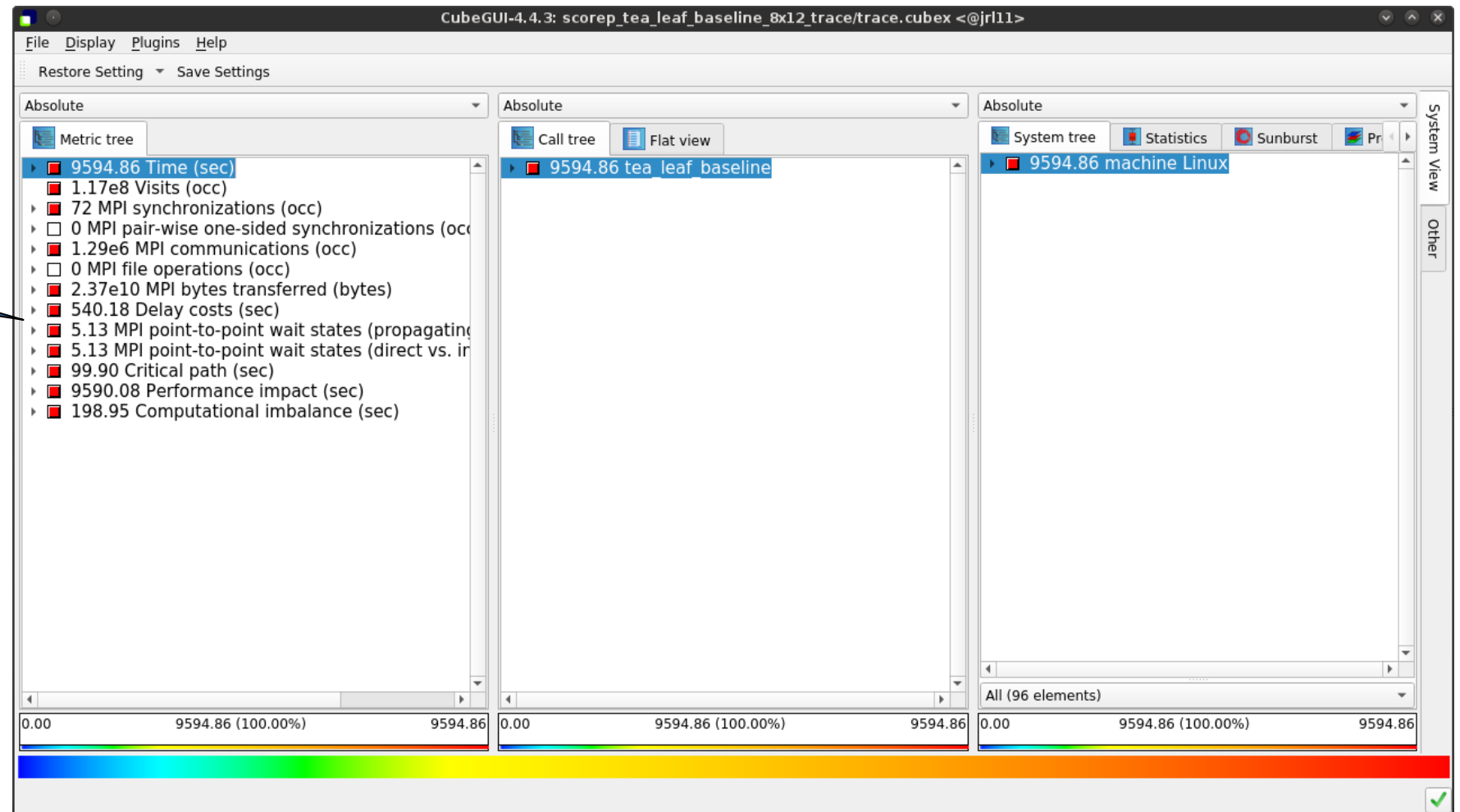
      [GUI showing trace analysis report]
```

Hint:

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

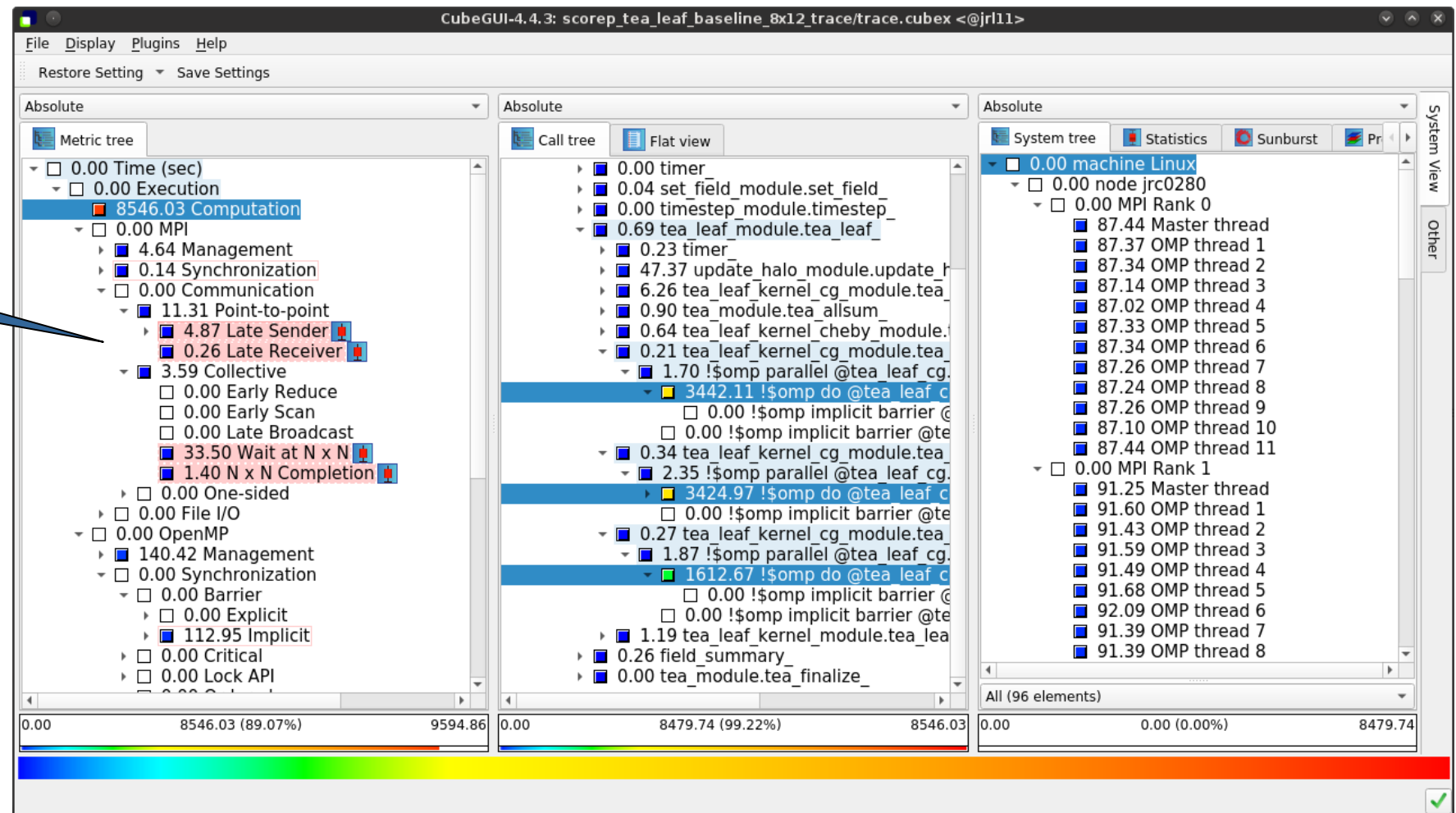
Scalasca analysis report exploration (opening view)

Additional top-level metrics produced by the trace analysis...

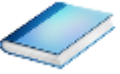


Scalasca wait-state metrics

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



Online metric description



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

The screenshot displays the CubeGUI-4.4.3 interface with the title bar "scorep_tea_leaf_baseline_8x12_trace/trace.cubex <@jr11>". The interface is divided into several panels:

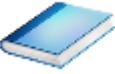
- Metric tree (left):** Shows a hierarchical view of metrics. The "Wait at N x N" metric is selected and highlighted in blue. A context menu is open over it, with "Documentation" selected.
- Call tree (middle):** Shows a detailed view of the selected metric's call tree. The "48 MPI Allreduce" metric is highlighted in blue.
- System tree (right):** Shows a view of the system tree with various nodes and their associated metrics.

The context menu for the "Wait at N x N" metric includes the following options:

- Info
- Documentation
- Expand/collapse
- Find items
- Clear found items
- Sort tree items...
- Copy to clipboard
- Edit metric...
- Identify metrics...
- Remove identification markers
- Show max severity in paraver
- Show metric statistics
- Show max severity information
- Mark this item
- Show max severity in Vampir

The bottom of the interface shows a color bar and a status bar with the text "Shows the documentation of the clicked item".

Online metric description (cont.)



Selection of different metric automatically updates description

CubeGUI-4.4.3: scorep_tea_leaf_baseline_8x12_trace/trace.cubex <@jr11>

File Display Plugins Help

Restore Setting Save Settings

Absolute

Metric tree

- 0.00 Time (sec)
 - 0.00 Execution
 - 8546.03 Computation
 - 0.00 MPI
 - 4.64 Management
 - 0.14 Synchronization
 - 0.00 Communication
 - 16.44 Point-to-point
 - 3.59 Collective
 - 0.00 Early Reduce
 - 0.00 Early Scan
 - 0.00 Late Broadcast
 - 33.50 Wait at N x N
 - 1.40 N x N Completion
 - 0.00 One-sided
 - 0.00 File I/O
 - 0.00 OpenMP
 - 140.42 Management
 - 112.95 Synchronization
 - 0.00 Flush
 - 0.00 Overhead
 - 735.75 Idle threads
 - 1.17e8 Visits (occ)
 - 72 MPI synchronizations (occ)
 - 0 MPI pair-wise one-sided synchronizations (occ)
 - 1.29e6 MPI communications (occ)

Absolute

Call tree Flat view

- 0.00 tea_leaf_baseline
 - 0.00 MAIN_
 - 0.00 tea_module.tea_init_comms
 - 0.00 !\$omp parallel @tea_leaf.f90:45
 - 0.00 initialise_
 - 0.00 diffuse_
 - 0.00 timer_
 - 0.00 set_field_module.set_field
 - 0.01 timestep_module.timestep_
 - 0.00 tea_leaf_module.tea_leaf_
 - 0.00 timer_
 - 0.00 update_halo_module.update_halo_
 - 0.00 tea_leaf_kernel_cg_module.tea_
 - 0.00 tea_module.tea_allsum_
 - 33.48 MPI_Allreduce_
 - 0.00 tea_leaf_kernel_cheby_module
 - 0.00 tea_leaf_kernel_cg_module.tea_
 - 0.00 tea_leaf_kernel_cg_module.tea_
 - 0.00 tea_leaf_kernel_cg_module.tea_
 - 0.00 tea_leaf_kernel_module.tea_
 - 0.00 field_summary_
 - 0.00 tea_module.tea_finalize_

Score-P Configuration Source Info

Metric : Waiting time due to inherent synchronization in MPI n-to-n operations
 Display name : Wait at N x N
 Unique name : mpi_wait_nxn
 Data type : DOUBLE

Region name: MPI_Allreduce
 Mangled name: MPI_Allreduce
 Region description:
 Call path ID: 214
 Beginning line: undefined

Metric Documentation Call path/Region Documentation

MPI Wait at N x N Time

Description:
 Collective communication operations that send data from all processes to all processes (i.e., n-to-n) exhibit an inherent synchronization among all participants, that is, no process can finish the operation until the last process has started it. This pattern covers the time spent in n-to-n operations until all processes have reached it. It applies to the MPI calls MPI_Reduce_scatter, MPI_Reduce_scatter_block, MPI_Allgather, MPI_Allgatherv, MPI_Allreduce and MPI_Alltoall.

processes

0

1

2

Sync. Collective

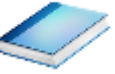
Sync. Collective

Sync. Collective

0.00 33.50 (0.35%) 9594.86

0.00 33.48 (99.96%) 33.50

Metric statistics



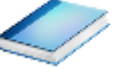
Access metric statistics for metrics marked with box plot icon from context menu

The screenshot displays the CubeGUI-4.4.3 interface with three main panels: Metric tree, Call tree, and System tree. The Metric tree panel on the left shows a hierarchical view of metrics, with '33.50 Wait at N x N' selected. A context menu is open over this metric, listing various actions. The 'Show metric statistics' option is highlighted. The Call tree panel in the middle shows the execution flow, and the System tree panel on the right shows the system hierarchy. A bar chart at the bottom of the Metric tree panel shows the distribution of the selected metric.

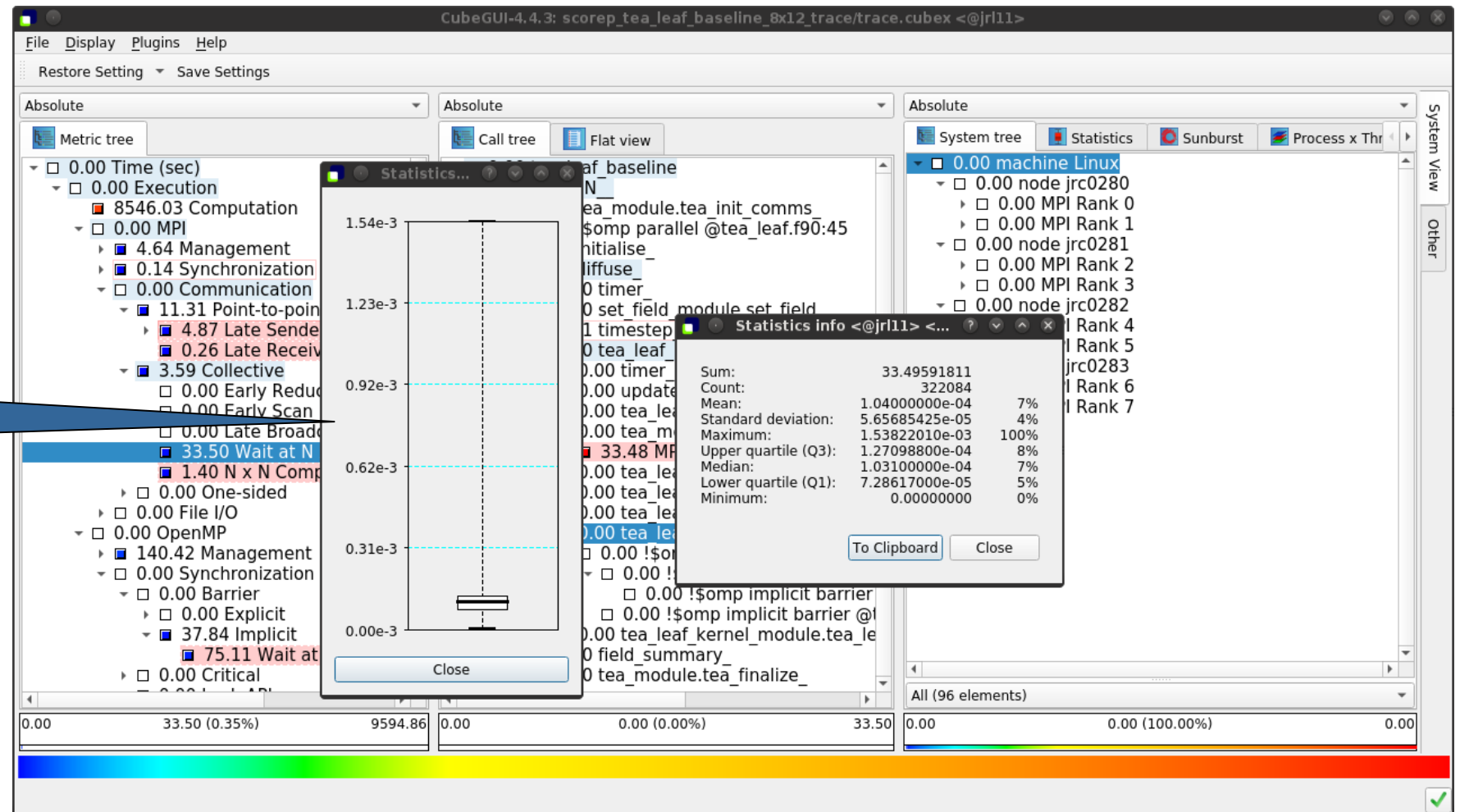
Context menu options:

- Info
- Documentation
- Expand/collapse
- Find items
- Clear found items
- Sort tree items...
- Copy to clipboard
- Edit metric...
- Identify metrics...
- Remove identification markers
- Show max severity in paraver
- Show metric statistics
- Show max severity information
- Mark this item
- Show max severity in Vampir

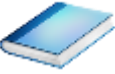
Metric statistics (cont.)



Shows instance statistics box plot, click to get details



Metric instance statistics (cont.)



The screenshot shows the CubeGUI-4.4.3 interface with the following panels:

- Metric tree:** A hierarchical view of metrics. The 'Late Sender' metric is highlighted with a blue callout box. Other metrics include 'Late Receiver' (0.26), 'MPI Waitall' (4.85), and 'Wait at Barrier' (75.11).
- Call tree:** A view showing the call paths for the selected metric. The 'MPI Waitall' metric is selected, showing its call path.
- System tree:** A view showing the system tree structure, including 'machine Linux', 'node jrc0280', and 'MPI Rank 0'.

A callout box titled 'Shows instance details' points to the 'Late Sender' metric. The details window shows the following information:

```
metric:
Time in MPI point-to-point receive operation waiting for a message [sec]
selected callpath:
+ 0.00 tea_leaf_baseline
+ 0.00 MAIN_
+ 0.00 diffuse_
+ 0.00 tea_leaf_module.tea_leaf_
+ 0.00 update_halo_module.update_halo_
+ 0.00 tea_module.tea_exchange_
+ 4.85 MPI_Waitall

enter:      3.9308
exit:       3.9323
duration:   0.0015
severity:   0.0014
rank:       5
```

The bottom of the interface shows a color bar and a table of metrics:

| Metric | Value | Percentage |
|---------|-------|------------|
| 0.00 | 4.87 | (0.05%) |
| 9594.86 | 0.00 | |
| 0.00 | 4.85 | (99.73%) |
| 4.87 | 0.00 | |
| 0.00 | 0.00 | (0.00%) |
| 4.85 | | |

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