

## Automatic trace analysis with the Scalasca Trace Tools

---

Markus Geimer  
Jülich Supercomputing Centre



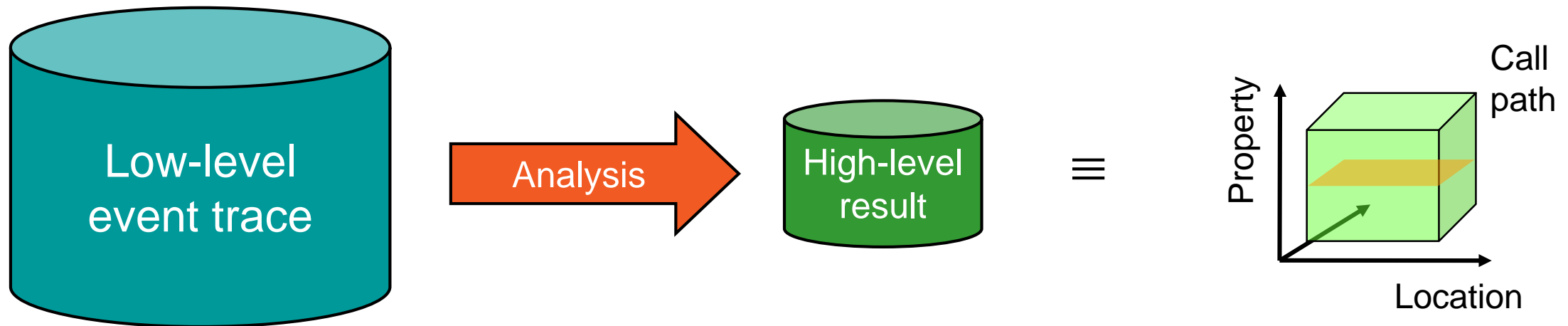
# Scalasca Trace Tools

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, 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 v2.6.1 (Dec 2022)

# Automatic trace analysis

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



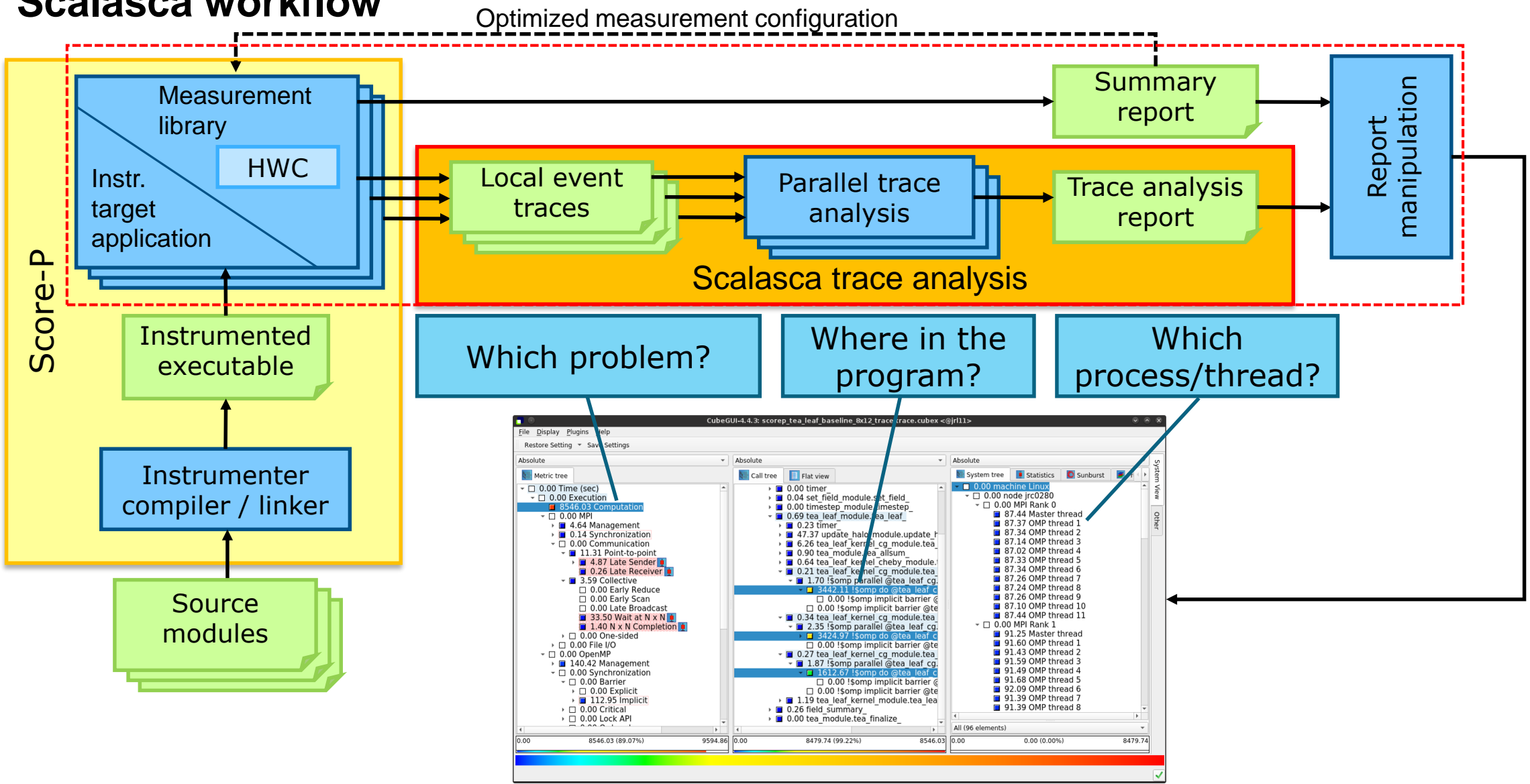
- Guaranteed to cover the entire event trace
- Quicker than manual/visual trace analysis
- Parallel replay analysis exploits available memory & processors to deliver scalability

# Scalasca Trace Tools: Features

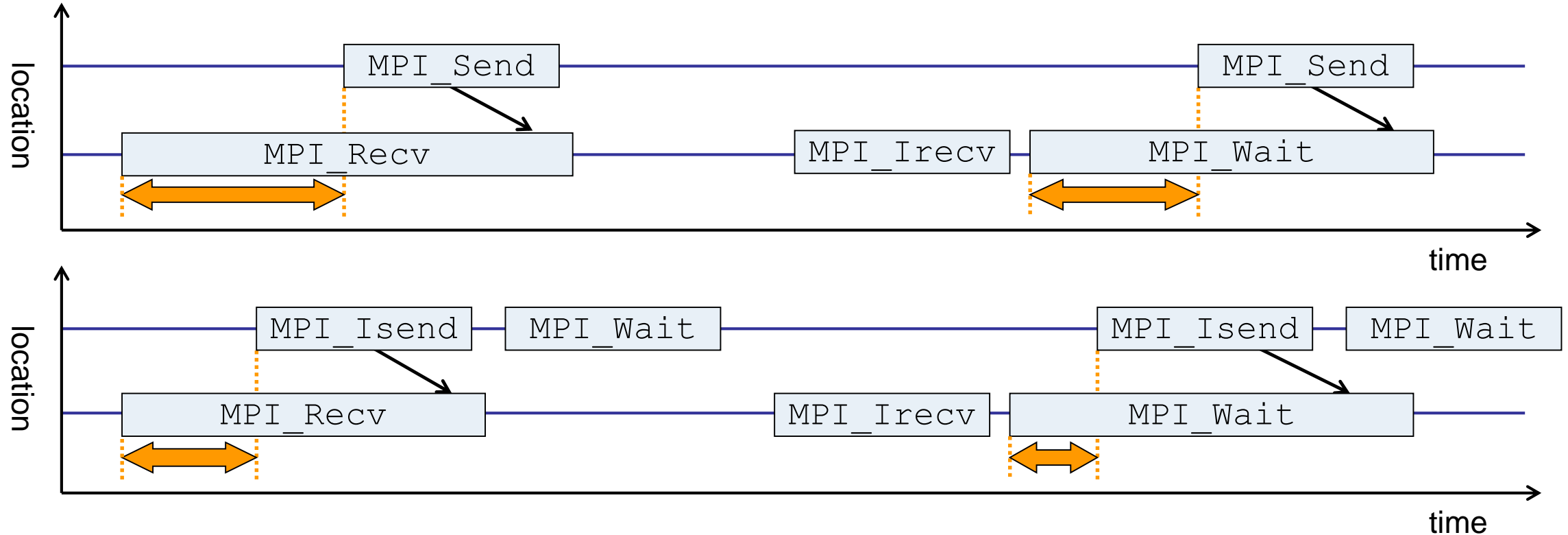
---

- Open source, 3-clause BSD license
- Supports all major HPC platforms
- Uses Score-P instrumenter & measurement libraries
  - Scalasca v2 core package focuses on trace-based analyses
  - Provides convenience commands for measurement, analysis, and post-processing
  - Supports common data formats
    - Reads event traces in OTF2 format
    - Writes analysis reports in CUBE4 format
- Current limitations:
  - Unable to handle traces ...
    - with MPI thread level exceeding `MPI_THREAD_FUNNELED`
    - containing memory events, CUDA/OpenCL device events (kernel, memcpy), SHMEM, or OpenMP nested parallelism
  - PAPI/rusage metrics for trace events are ignored

# Scalasca workflow

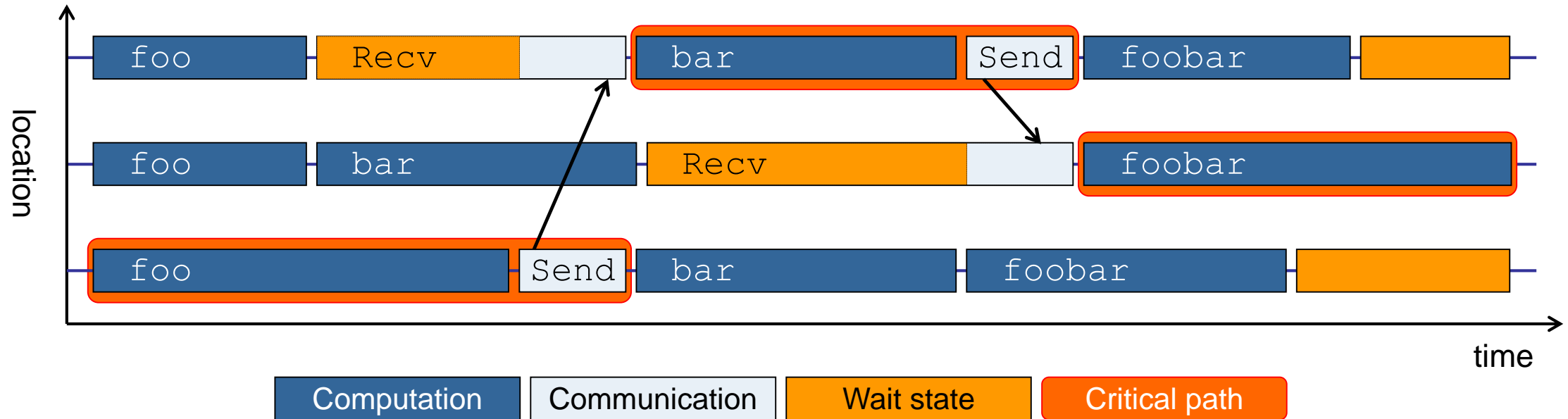


## Example: “Late Sender” wait state



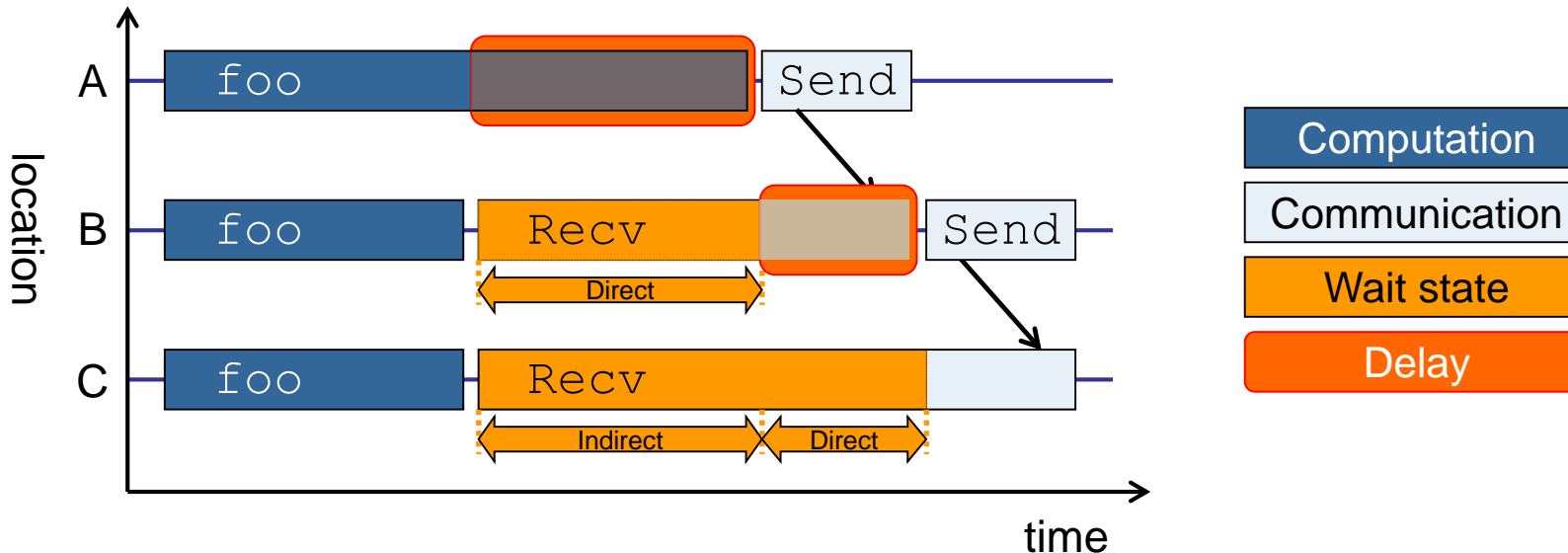
- Waiting time caused by a blocking receive operation posted earlier than the corresponding send
- Applies to blocking as well as non-blocking communication

## Example: Critical path



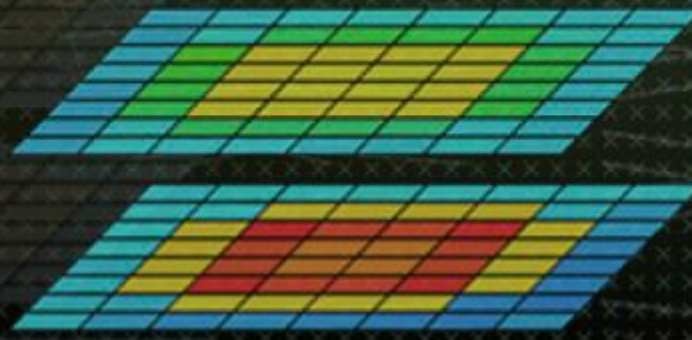
- Shows call paths and processes/threads that are responsible for the program's wall-clock runtime
- Identifies good optimization candidates and parallelization bottlenecks

## 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*





## Hands-on: NPB-MZ-MPI / BT

---

trace tools   
scalasca

## Recap: Setup for exercises

---

- Connect to your account on one of the training systems

- Barnard (TUD): 

```
% ssh -Y <account>@login[1-4].barnard.hpc.tu-dresden.de
```
- Claix-2023 (RWTH): 

```
% ssh -Y <account>@login23-[1-4].hpc.itc.rwth-aachen.de
```

- Set account and default environment via helper script

- Barnard (TUD): 

```
% source /projects/p_nhr_vihps/setup.sh
```
- Claix-2023 (RWTH): 

```
% source /home/hpc/vihps-tw44/setup.sh
```

- Change to directory containing BT-MZ sources

- Existing instrumented executable can be reused

```
% cd $VIHPS_WORKSPACE  
% cd hands-on/score-p
```

## Demo: BT-MZ summary measurement collection...

```
% cp jobscript/[barnard|claix-2023]/bt-mz.sbatch .
% vim bt-mz.sbatch
#SBATCH -J filter          <= change this to "scalasca"

#export SCOREP_ENABLE_PROFILING=true
#export SCOREP_ENABLE_TRACING=true
export SCOREP_FILTERING_FILE=initial_scorep.filter
#export SCOREP_TOTAL_MEMORY=
# change NOTES as desired to reflect measurement settings
NOTES=summary
export SCOREP_EXPERIMENT_DIRECTORY=...

# Scalasca settings
NEXUS="scan -s"
#SCAN_ANALYZE_OPTS="--time-correct"

$NEXUS srun -n $SLURM_NTASKS [...] $EXE
```

```
% sbatch bt-mz.sbatch
```

- Change to the top-level directory and edit the job script

### Hint:

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

## scan: 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)

## Demo: BT-MZ summary measurement

---

```
S=C=A=N: Scalasca 2.6.1 runtime summarization
S=C=A=N: scalasca/scorep-4-12-summary experiment archive
S=C=A=N: Fri Feb 23 11:54:48 2024: Collect start
srun ... bin.scorep/bt-mz_C.4

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

Number of zones:  8 x  8
Iterations: 200    dt:  0.000100
Number of active processes:      4

[... More application output ...]

S=C=A=N: Fri Feb 23 11:55:09 2024: Collect done (status=0) 21s
S=C=A=N: scalasca/scorep-4-12-summary complete.
```

- Run the application using the Scalasca measurement collection & analysis nexus prefixed to launch command
- Creates experiment directory:  
scorep-4-12-summary

## Demo: BT-MZ summary analysis report examination

---

- Score summary analysis report

```
% square -s scalasca/scorep-4-12-summary  
INFO: Post-processing runtime summarization report (profile.cubex)...  
INFO: Score report written to scalasca/scorep-4-12-summary/scorep.score
```

- Post-processing and interactive exploration with Cube

```
% square scalasca/scorep-4-12-summary  
INFO: Displaying scalasca/scorep-4-12-summary/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

## BT-MZ trace measurement collection...

```
% vim bt-mz.sbatch
#SBATCH -J scalasca

#export SCOREP_ENABLE_PROFILING=true
#export SCOREP_ENABLE_TRACING=true
export SCOREP_FILTERING_FILE=initial_scorep.filter
export SCOREP_TOTAL_MEMORY=348MB
# change NOTES as desired to reflect measurement settings
NOTES=trace
export SCOREP_EXPERIMENT_DIRECTORY=...

# Scalasca settings
NEXUS="scan -t"
SCAN_ANALYZE_OPTS="--time-correct"

$NEXUS srun -n $SLURM_NTASKS [...] $EXE
```

```
% sbatch bt-mz.sbatch
```

- Change to the top-level directory and edit the job script
- Add "-t" to 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: Fri Feb 23 12:49:25 2024: Collect start
srun ... bin.scorep/bt-mz_C.4

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

Number of zones: 8 x 8
Iterations: 200 dt: 0.000100
Number of active processes: 4

[... More application output ...]

S=C=A=N: Fri Feb 23 12:49:45 2024: Collect done (status=0) 20s
```

- Starts measurement with collection of trace files ...



## BT-MZ trace measurement ... analysis

```
...
S=C=A=N: Fri Feb 23 12:49:45 2024: Analyze start
  srun [...] scout.hyb --time-correct \
>   scalasca/scorep-4-12-trace/traces.otf2

SCOUT   (Scalasca 2.6.1)

Analyzing experiment archive scalasca/scorep-4-12-trace/traces.otf2

Opening experiment archive ... done (0.002s).
Reading definition data    ... done (0.002s).
Reading event trace data  ... done (1.117s).
Preprocessing              ... done (0.729s).
Timestamp correction       ... done (4.370s).
Analyzing trace data       ... done (23.496s).
Writing analysis report    ... done (0.284s).

Max. memory usage          : 3048.270MB

      # passes              : 1
      # violated            : 0

Total processing time      : 30.087s
S=C=A=N: Fri Feb 23 12:50:21 2024: Analyze done (status=0) 36s
```

- Continues with automatic (parallel) analysis of trace files

## BT-MZ trace analysis report exploration

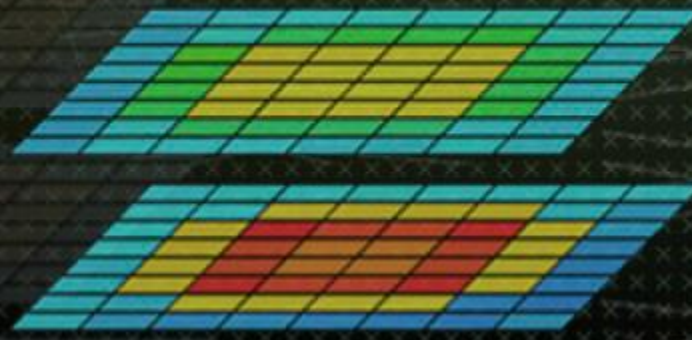
---

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

```
% square scalasca/scorep-4-12-trace  
INFO: Post-processing runtime summarization report (profile.cubex)...  
INFO: Post-processing trace analysis report (scout.cubex)...  
INFO: Displaying scalasca/scorep-4-12-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



## Case study: TeaLeaf MPI+OpenMP

---



## Case study: TeaLeaf MPI+OpenMP

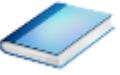
---

- HPC mini-app developed by the UK Mini-App Consortium
  - Solves the linear 2D heat conduction equation on a spatially decomposed regular grid using a 5 point stencil with implicit solvers
  - Part of the Mantevo 3.0 suite
  - Available on GitHub: <https://uk-mac.github.io/TeaLeaf/>
- Measurements of TeaLeaf reference v1.0 taken on Jureca cluster @ JSC
  - Using Intel 19.0.3 compilers, Intel MPI 2019.3, Score-P 5.0, and Scalasca 2.5
  - Run configuration
    - 8 MPI ranks with 12 OpenMP threads each
    - Distributed across 4 compute nodes (2 ranks per node)
    - Test problem "5": 4000 × 4000 cells, CG solver

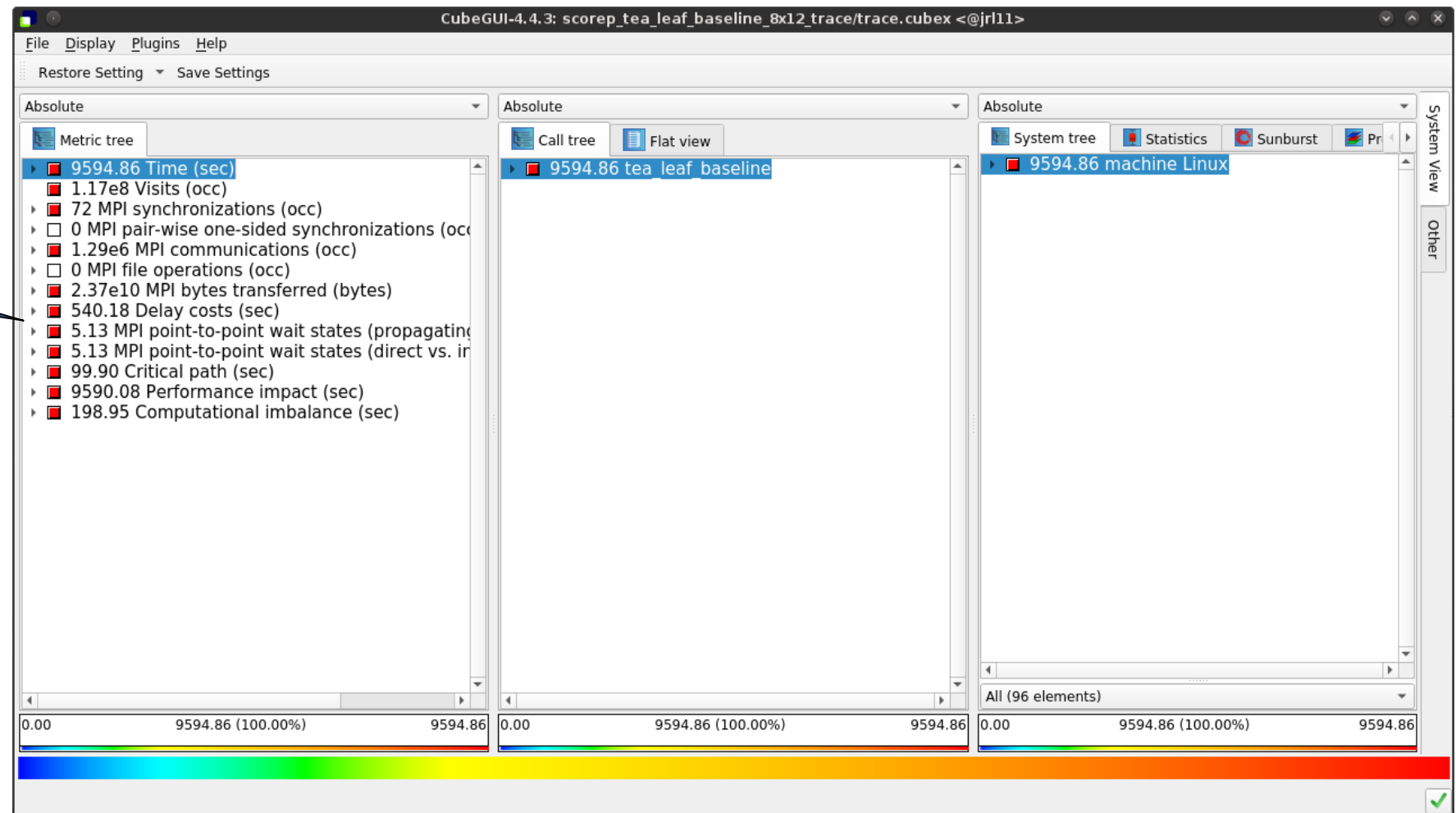


```
% cube scorep_tea_leaf_baseline_8x12_trace/trace.cubex  
[GUI showing post-processed trace analysis report]
```

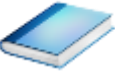
# Scalasca analysis report exploration (opening view)



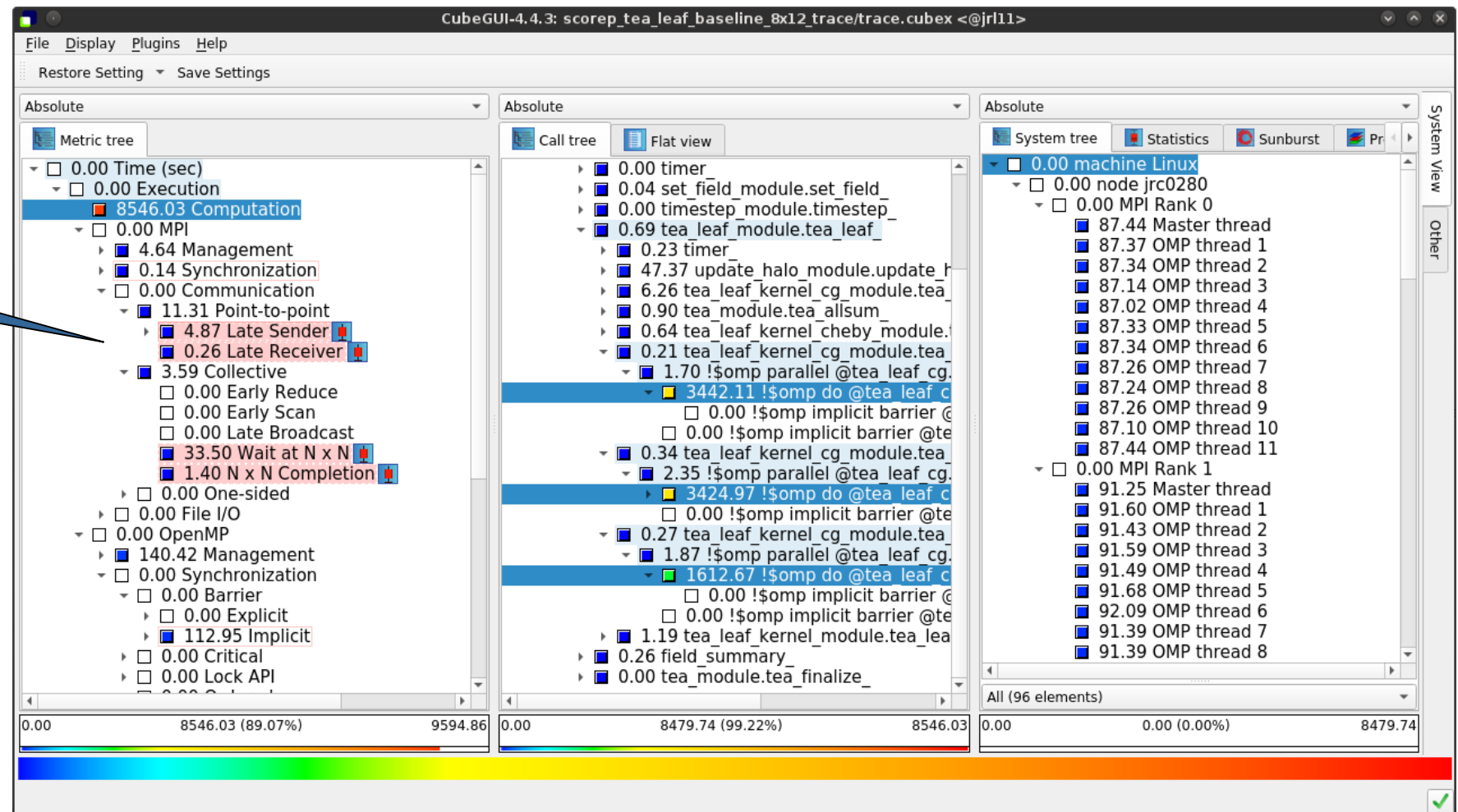
Additional top-level metrics produced by the trace analysis...



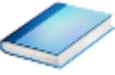
# Scalasca wait-state metrics



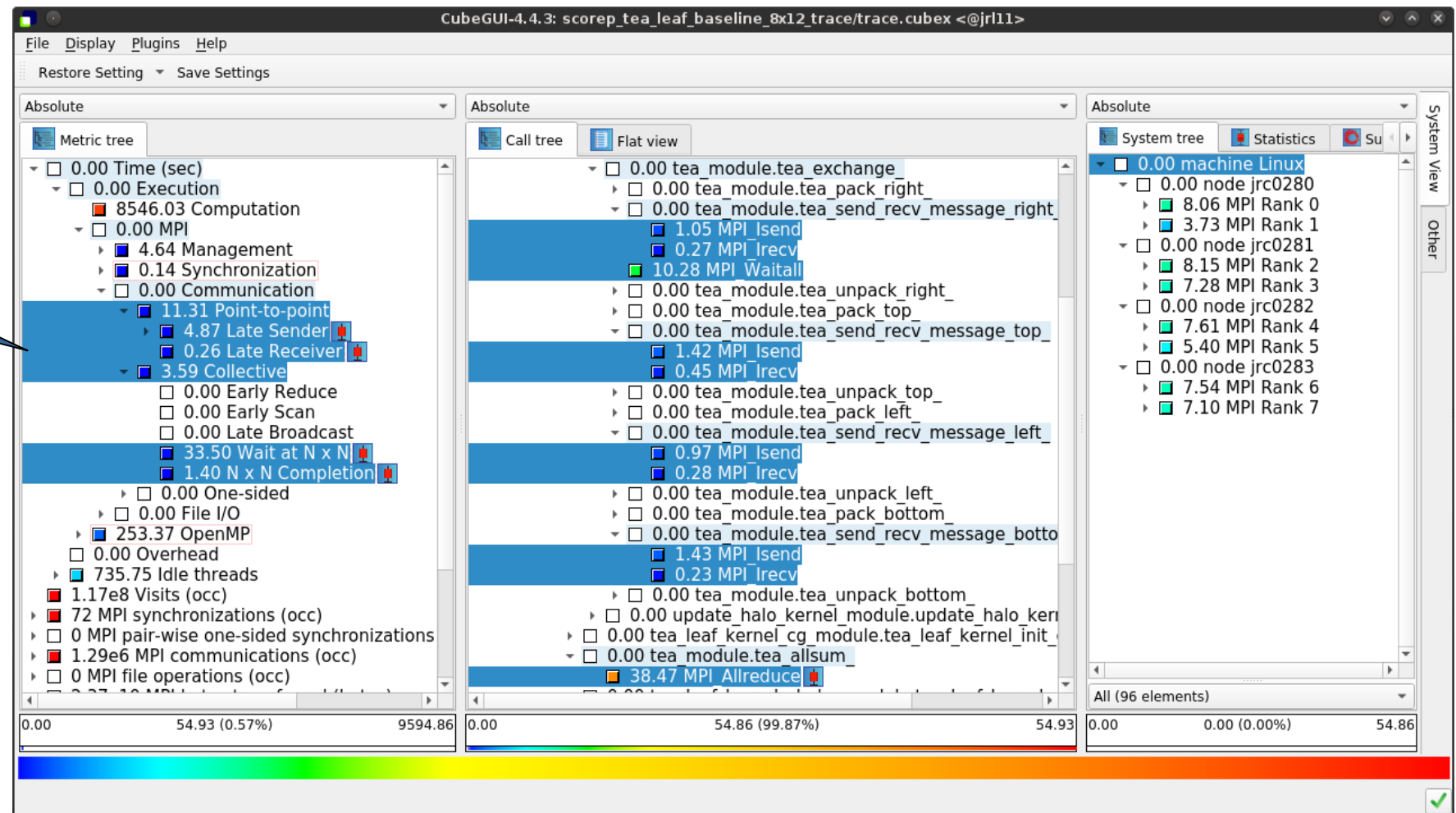
...plus additional wait-state metrics as part of the “Time” hierarchy



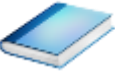
# TeaLeaf Scalasca report analysis (I)



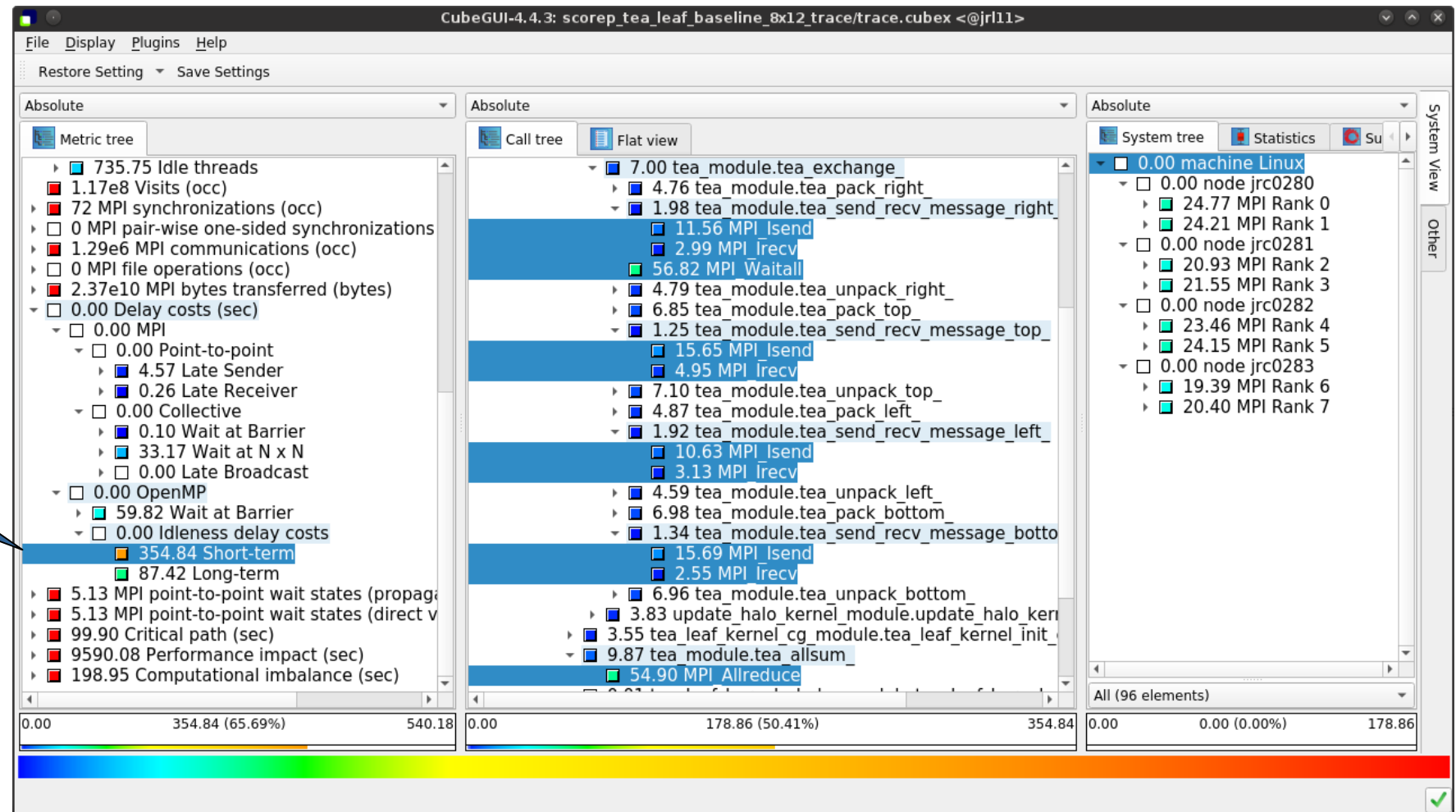
While MPI communication time and wait states are small (~0.6% of the total execution time)...



# TeaLeaf Scalasca report analysis (II)

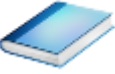


...they directly cause a significant amount of the OpenMP thread idleness

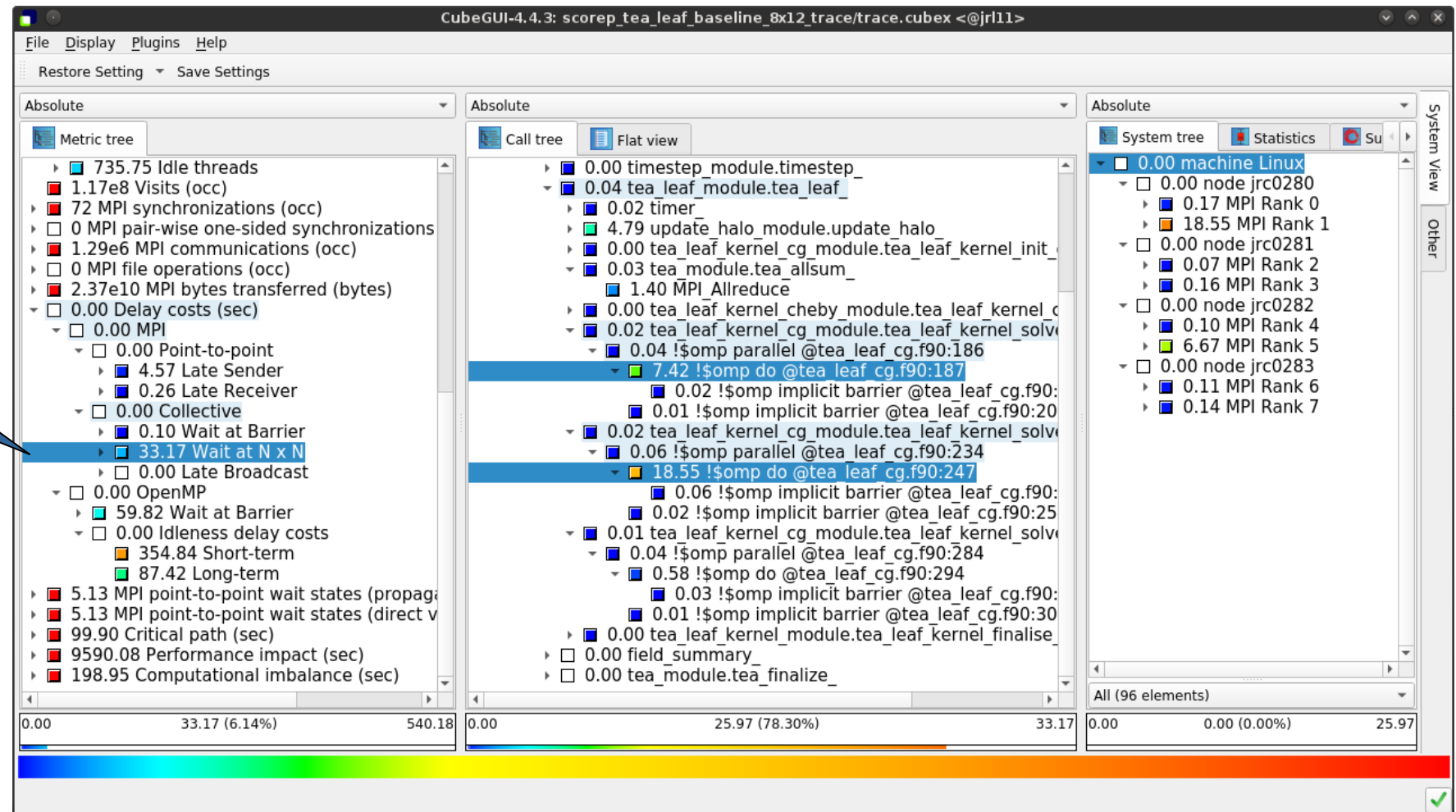




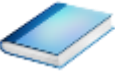
# TeaLeaf Scalasca report analysis (III)



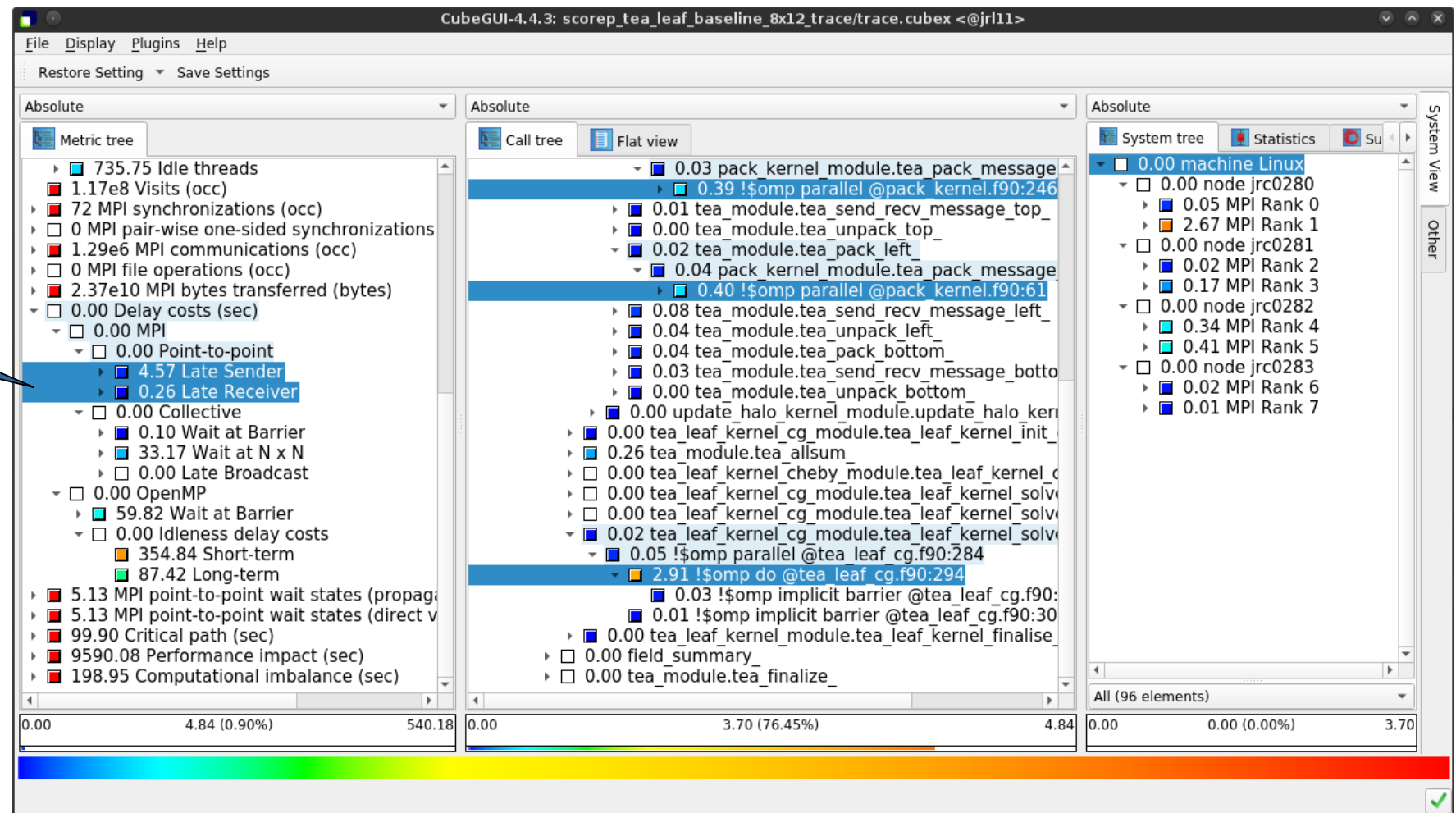
The “Wait at NxN” collective wait states are mostly caused by the first 2 OpenMP `do` loops of the solver (on ranks 5 & 1, resp.)...



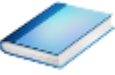
# TeaLeaf Scalasca report analysis (IV)



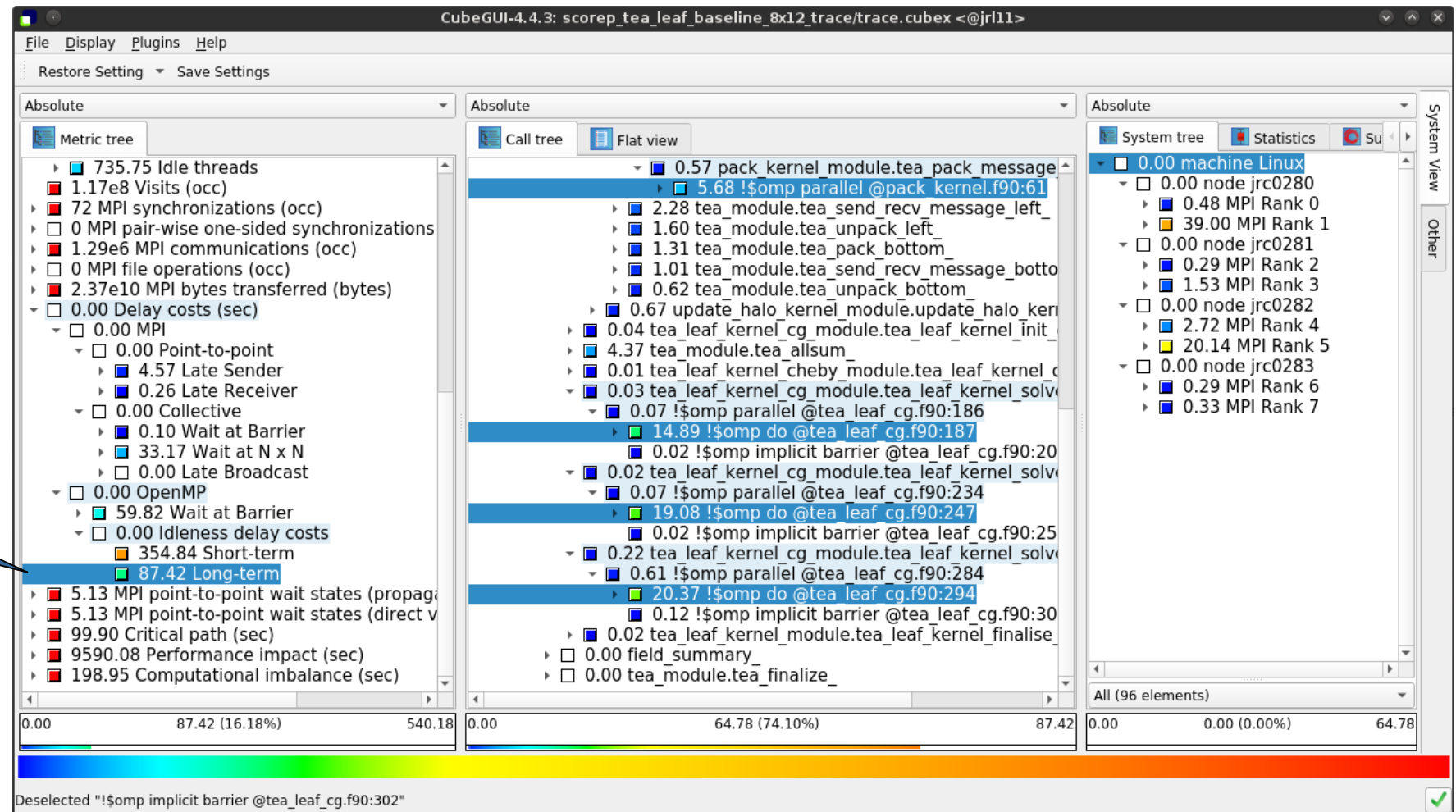
...while the MPI point-to-point wait states are caused by the 3<sup>rd</sup> solver do loop (on rank 1) and two loops in the halo exchange



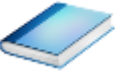
# TeaLeaf Scalasca report analysis (V)



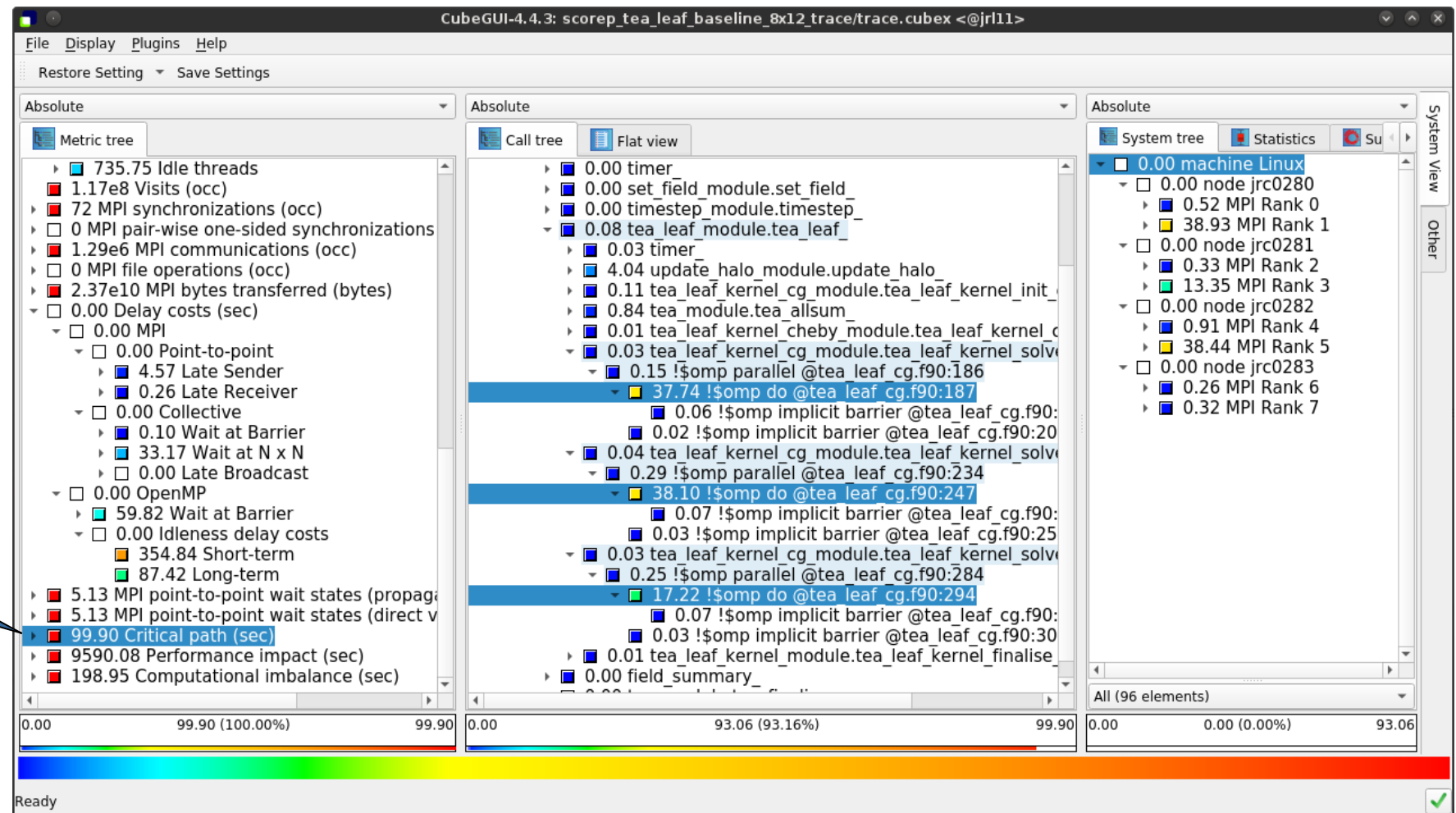
Various OpenMP `do` loops (incl. the solver loops) also cause OpenMP thread idleness on other ranks via propagation



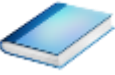
# TeaLeaf Scalasca report analysis (VI)



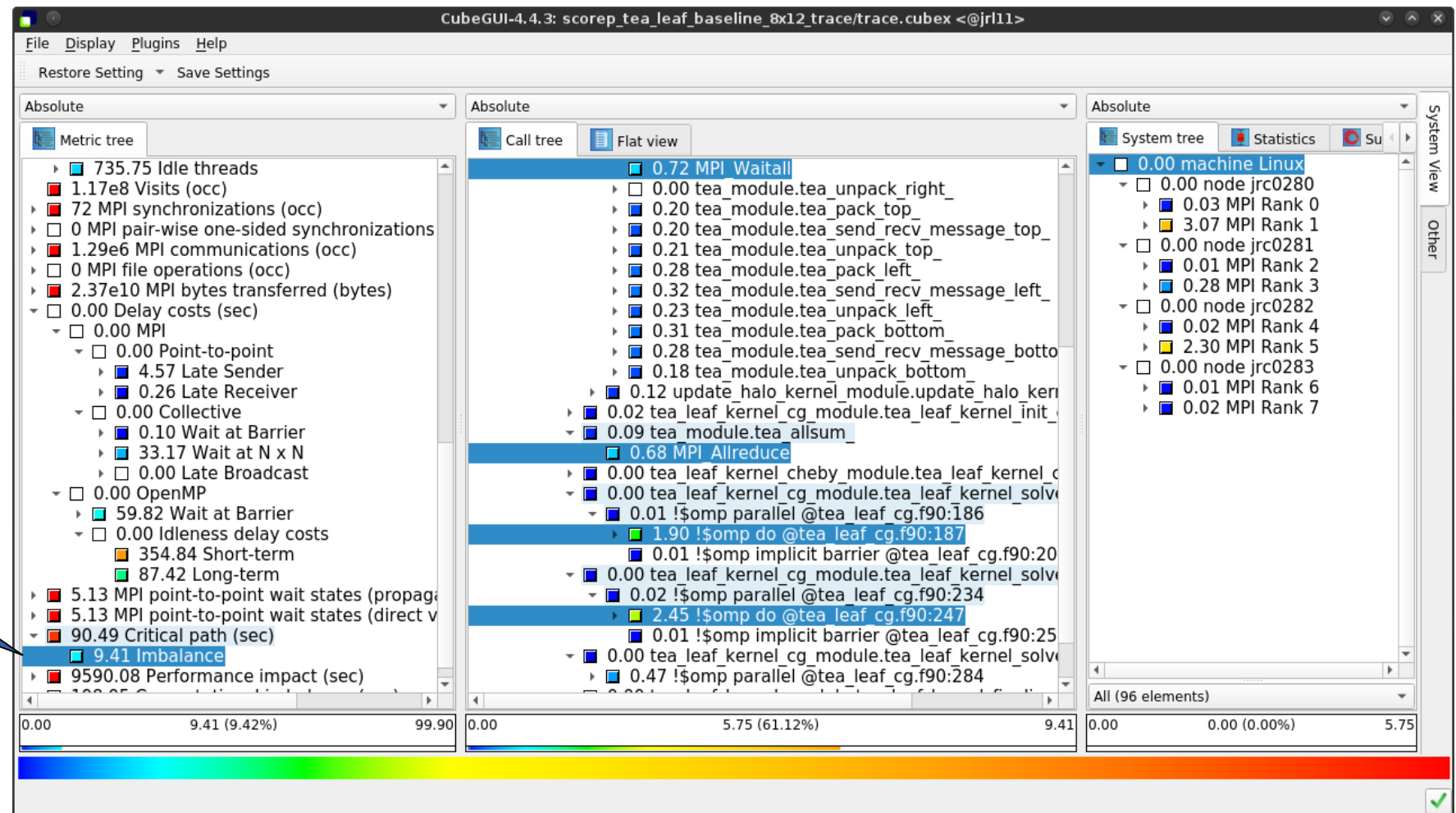
The Critical Path also highlights the three solver loops...



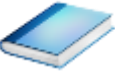
# TeaLeaf Scalasca report analysis (VII)



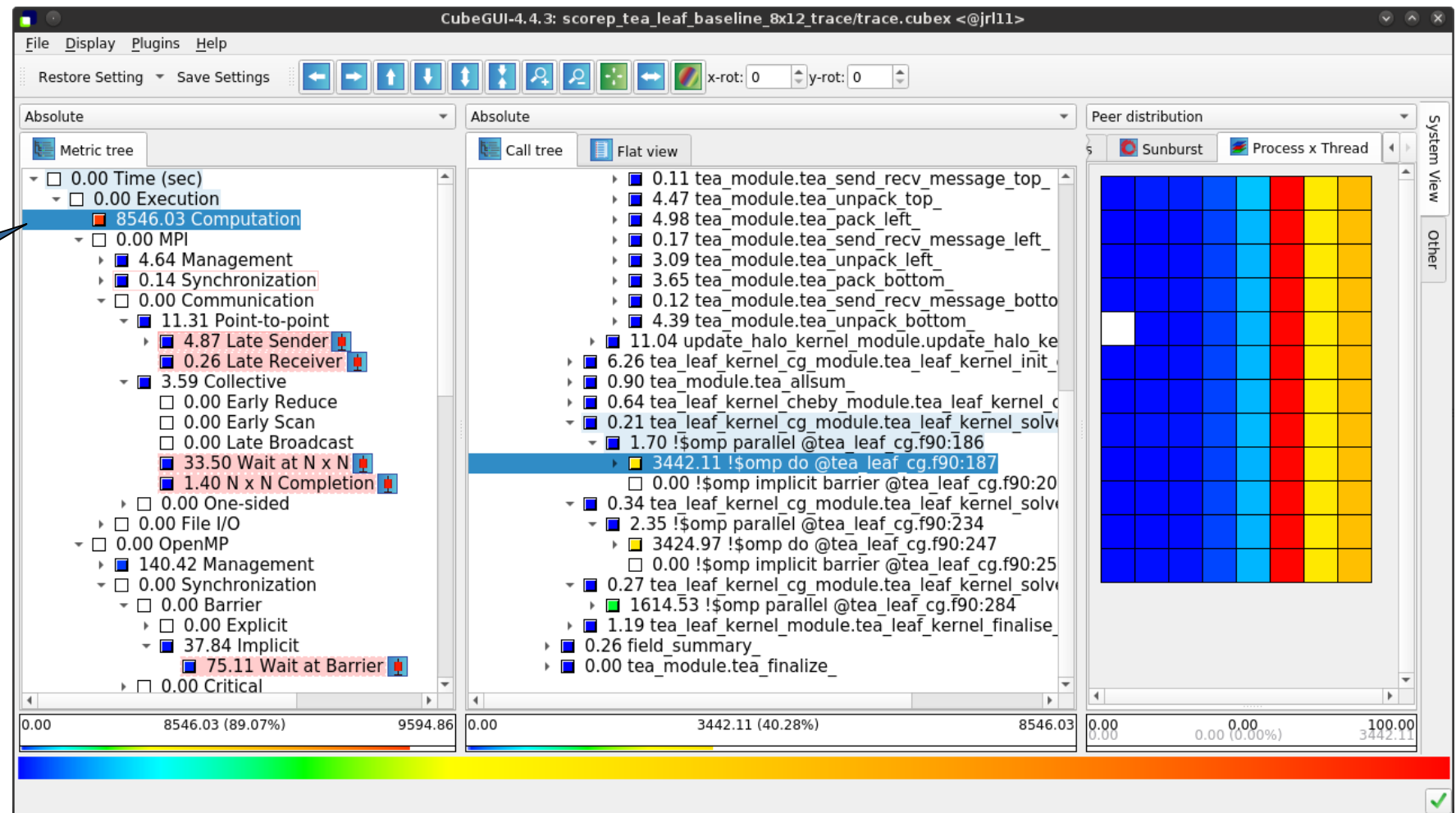
...with imbalance (time on critical path above average) mostly in the first two loops and MPI communication



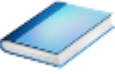
# TeaLeaf Scalasca report analysis (VIII)



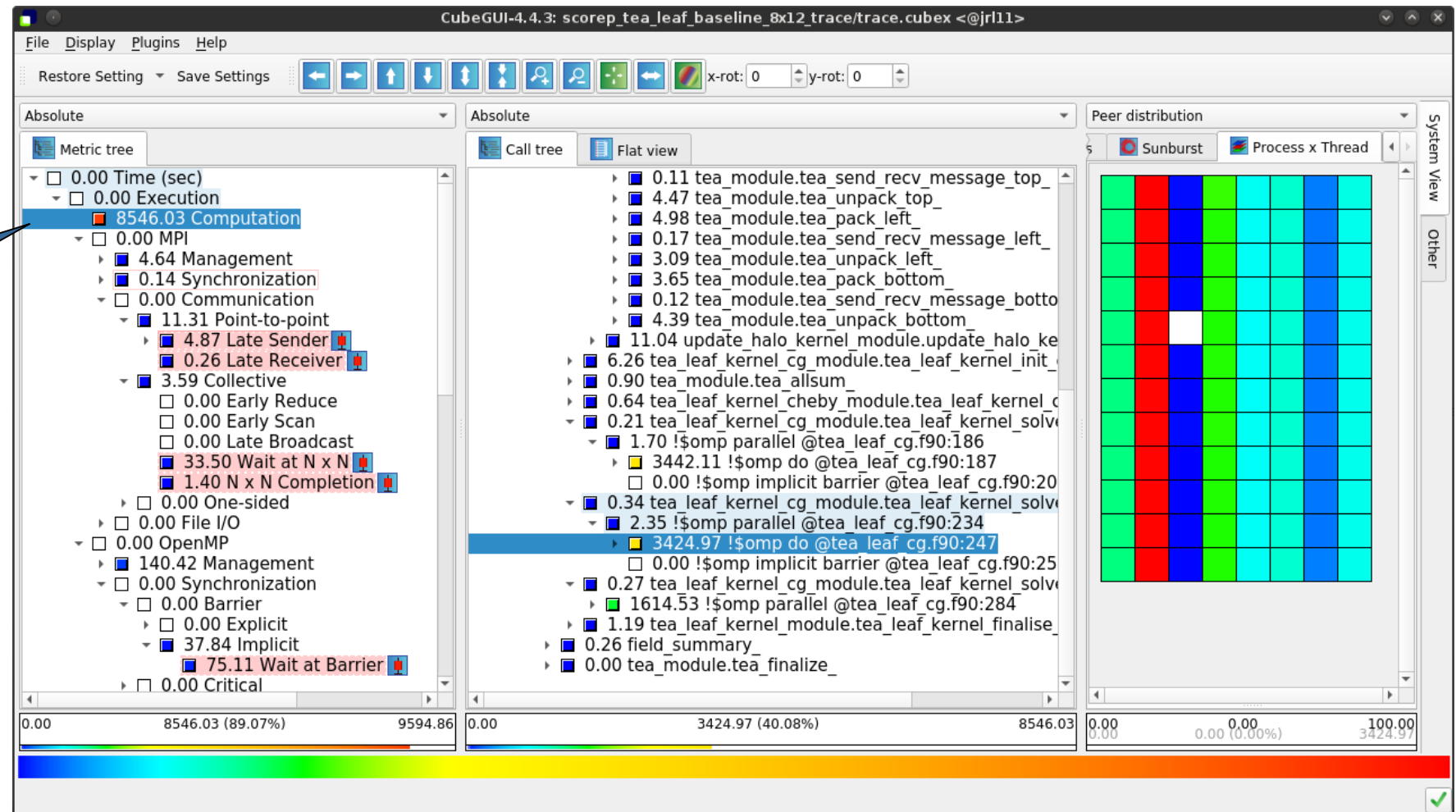
Computation time of  
1<sup>st</sup>...



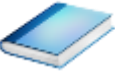
# TeaLeaf Scalasca report analysis (IX)



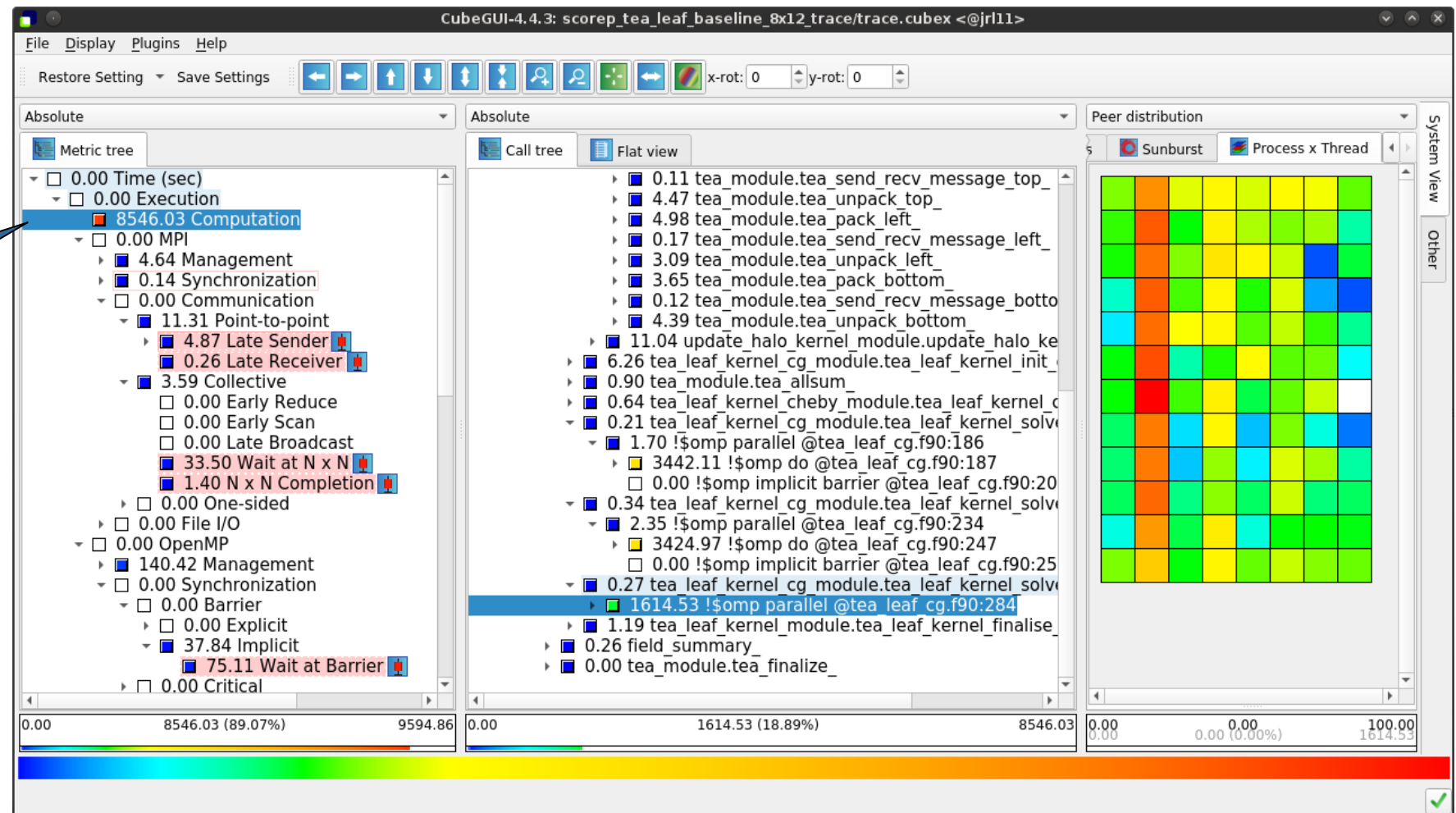
...and 2<sup>nd</sup> do loop mostly balanced within each rank, but vary considerably across ranks...



# TeaLeaf Scalasca report analysis (X)



...while the 3<sup>rd</sup> do loop also shows imbalance within each rank





## TeaLeaf analysis summary

---

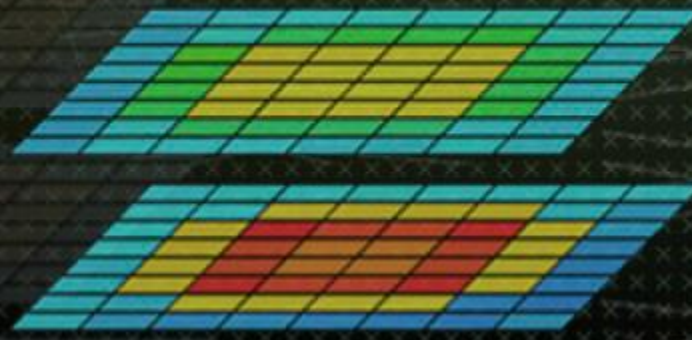
- The first two OpenMP do loops of the solver are well balanced within a rank, but are imbalanced across ranks
  - Requires a global load balancing strategy
- The third OpenMP do loop, however, is imbalanced within ranks,
  - causing direct “Wait at OpenMP Barrier” wait states,
  - which cause indirect 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:
  - <https://www.scalasca.org>
- Contact:
  - mailto: [scalasca@fz-juelich.de](mailto:scalasca@fz-juelich.de)





## Reference material

---



# 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 will be removed in the next major release  
⇒ use Score-P instrumenter directly

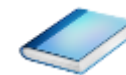
# Scalasca convenience command: scan / scalasca -analyze



```
% 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.
-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
                       : the processed configuration for comparison.
```

- Scalasca measurement collection & analysis nexus

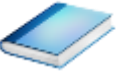
# Scalasca convenience command: square / scalasca -examine



```
% 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
  -c <number>              : Consider number of counters when doing scoring (-s)
  -F                       : Force remapping of already existing reports
  -f filtfiler             : 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 option(s) to scorep-score
```

## ▪ Scalasca analysis report explorer (Cube)

# Scalasca advanced command: scout - Scalasca automatic trace analyzer

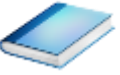


```
% scout.hyb --help
SCOUT      (Scalasca 2.6.1)
Copyright (c) 1998-2022 Forschungszentrum Juelich GmbH
Copyright (c) 2014-2021 RWTH Aachen University
Copyright (c) 2009-2014 German Research School for Simulation Sciences GmbH

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 advanced command: `clc_synchronize`



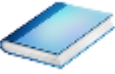
- Scalasca trace event timestamp consistency correction

```
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



# Online metric description



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

The screenshot shows 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:

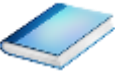
- Metric tree (left):** A hierarchical tree of metrics. The "Wait at N x N" metric is selected, and a context menu is open over it. The menu options include: Info, Documentation (highlighted), Expand/collapse, Find items, Clear found items, Sort tree items..., Copy to clipboard, Edit metric..., Identify metrics..., Remove identification markers, Show max severity in paraver, Show metric statistics, Show max severity information, Mark this item, and Show max severity in Vampir.
- Call tree (middle):** A tree view showing the execution flow of the application. The "diffuse\_" metric is selected.
- System tree (right):** A tree view showing the system hierarchy, including nodes and MPI ranks.

At the bottom of the interface, there is a table showing the performance of the selected metric across different components:

Component	Value	Percentage
0.00	33.50	(0.35%)
0.00	33.50	(99.96%)
0.00	0.00	(0.00%)
0.00	33.48	

A color bar at the bottom indicates the severity of the metrics, ranging from blue (low) to red (high).

# 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)

0.00 33.50 (0.35%) 9594.86

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\_
                                    - 0.00

0.00 33.48 (99.96%) 33.50

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\_Allgather, MPI\_Allreduce and MPI\_Alltoall.

processes

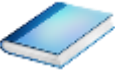
0 Sync. Collective

1 Sync. Collective

2 Sync. Collective

System View Other

# Metric statistics



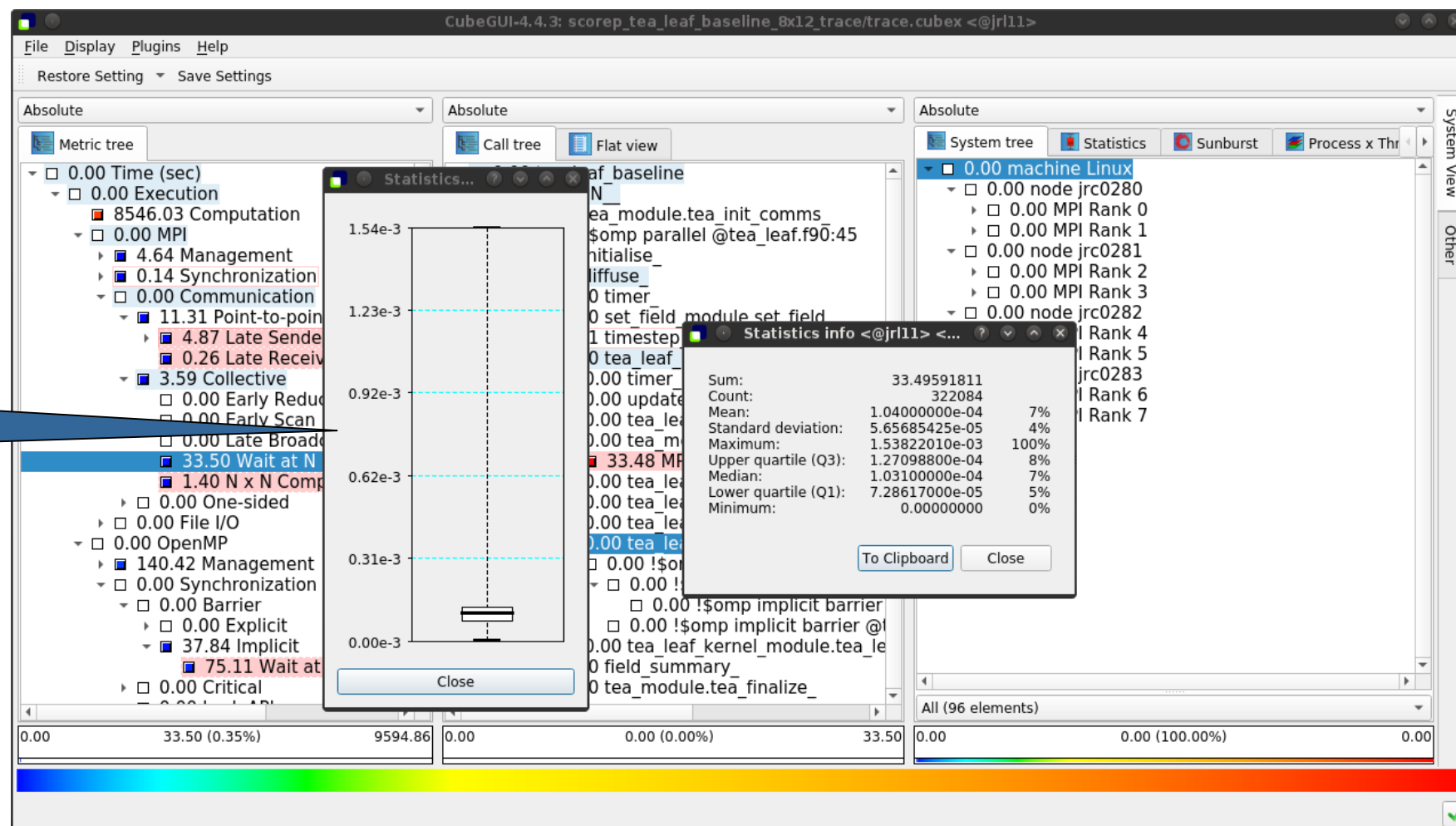
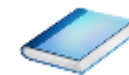
Access metric statistics for metrics marked with box plot icon from context menu

The screenshot displays the CubeGUI-4.4.3 interface for a trace file named 'scorep\_tea\_leaf\_baseline\_8x12\_trace/trace.cubex'. The interface is divided into several panels:

- Metric tree (left):** Shows a hierarchical view of metrics. The 'Wait at N x N' metric is highlighted, and a context menu is open over it. The menu includes options like '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' (highlighted), 'Show max severity information', 'Mark this item', and 'Show max severity in Vampir'.
- Call tree (middle):** Shows a detailed view of the selected metric, including sub-metrics like 'tea\_leaf\_kernel\_cheby\_module', 'tea\_leaf\_kernel\_cg\_module', and 'tea\_leaf\_kernel\_module'.
- System tree (right):** Shows a view of the system components, including nodes like 'jrc0280', 'jrc0281', 'jrc0282', 'jrc0283', and 'jrc0284', each with associated MPI ranks and their respective metrics.

The 'Show metric statistics' option is highlighted in the context menu, indicating that the user is accessing the detailed statistics for the selected metric.

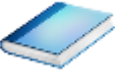
# Metric statistics (cont.)



Shows instance statistics box plot, click to get details



# Metric instance statistics (cont.)



Shows instance details

0.00 4.87 (0.05%) 9594.86

0.00 4.85 (99.73%) 4.87

0.00 0.00 (0.00%) 4.85