

Automatic trace analysis with the Scalasca Trace Tools

Ilya Zhukov
Jülich Supercomputing Centre



Scalasca Trace Tools

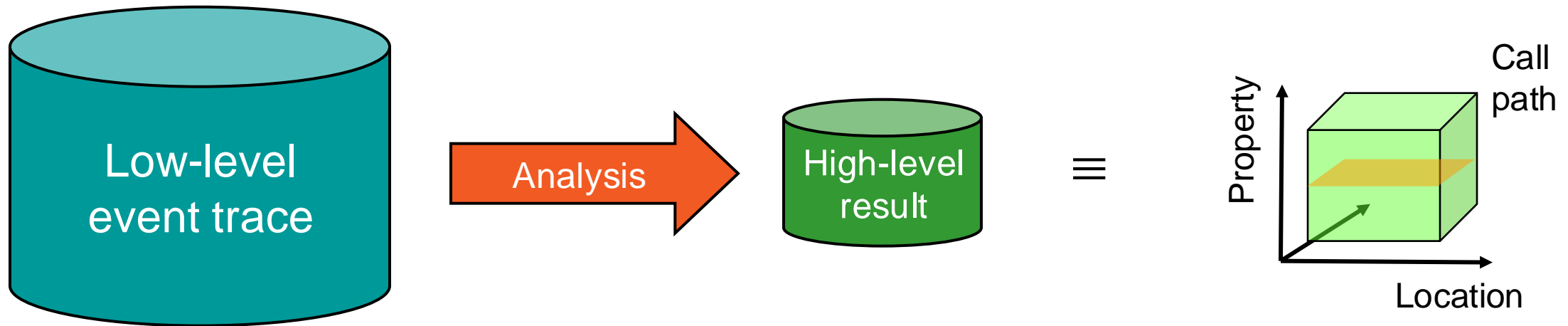
DOI [10.5281/zenodo.4103922](https://doi.org/10.5281/zenodo.4103922)

- **Scalable trace-based** performance analysis toolset for the most popular parallel programming paradigms
 - Current focus: MPI, OpenMP, and (to a limited extend) 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 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

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

Hands-on: CloverLeaf MPI+OpenACC



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 examination & analysis

Recap: Setup for exercises

- Connect to your account on Leonardo (with X11-forwarding)

```
% ssh -X <yourid>@login.leonardo.cineca.it
```

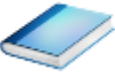
- Set tools for a particular software stack (NVHPC + OpenMPI) via helper script

```
% module load nvhpc/23.1 openmpi/4.1.4--nvhpc--23.1-cuda-11.8  
% source /leonardo/pub/userexternal/bwylie00/tools/nvomp/setup.sh
```

- Change to directory containing CloverLeaf_OpenACC sources
 - Existing instrumented executable can be reused

```
% cd /leonardo_work/tra24_epicure/<your_user_name>  
% cd CloverLeaf_OpenACC
```

CloverLeaf_OpenACC summary analysis report examination



- Score summary analysis report

```
% square -s scorep_clover_leaf_8_sum  
INFO: Post-processing runtime summarization result (profile.cubex)...  
INFO: Score report written to ./scorep_clover_leaf_8_sum/scorep.score
```

- Post-processing and interactive exploration with Cube

```
% square scorep_clover_leaf_8_sum  
INFO: Displaying ./scorep_clover_leaf_8_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

CloverLeaf_OpenACC trace measurement collection...



```
% cd bin.scorep
% cp ../jobscripts/scalasca.sbatch .
% cat scalasca.sbatch

#export SCOREP_ENABLE_TRACING=true
export SCOREP_FILTERING_FILE=../config/scorep.filter
export SCOREP_OPENACC_ENABLE=regions,wait,enqueue
export SCOREP_CUDA_ENABLE=runtime
export SCOREP_TOTAL_MEMORY=17MB
export SCAN_ANALYZE_OPTS=--time-correct

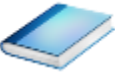
scalasca -analyze -t srun ./clover_leaf
```

```
% sbatch scalasca.sbatch
```

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

- Submit the job

CloverLeaf_OpenACC trace measurement ... collection



```
S=C=A=N: Scalasca 2.6.1 trace collection and analysis  
S=C=A=N: Wed Oct 25 14:58:52 2023: Collect start  
srun ./clover_leaf
```

```
Clover version      1.300
```

```
[... More application output ...]
```

```
S=C=A=N: Wed Oct 25 14:59:33 2023: Collect done (status=0) 41s
```

- Starts measurement with collection of trace files ...

CloverLeaf_OpenACC trace measurement ... analysis



```
...
S=C=A=N: Wed Oct 25 14:59:33 2023: Analyze start
srun scout.mpi --time-correct ./scorep_clover_leaf_8_trace/traces.otf2

SCOUT    (Scalasca 2.6.1)

Analyzing experiment archive ./scorep_clover_leaf_8_trace/traces.otf2

Opening experiment archive ... done (0.009s).
Reading definition data    ... done (0.008s).
Reading event trace data  ... done (0.695s).
Preprocessing              ... done (0.721s).
Timestamp correction       ... done (0.711s).
Analyzing trace data       ... done (10.108s).
Writing analysis report    ... done (0.137s).

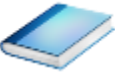
Max. memory usage          : 426.422MB

      # passes              : 1
      # violated            : 0

Total processing time      : 12.489s
S=C=A=N: Wed Oct 25 14:59:49 2023: Analyze done (status=0) 16s
```

- Continues with automatic (parallel) analysis of trace files

CloverLeaf_OpenACC trace analysis report exploration



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

```
% square scorep_clover_leaf_8_trace
INFO: Post-processing runtime summarization report (profile.cubex)...
INFO: Post-processing trace analysis report (scout.cubex)...
INFO: Displaying ./scorep_clover_leaf_8_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

Demo: TeaLeaf MPI+OpenMP case study



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

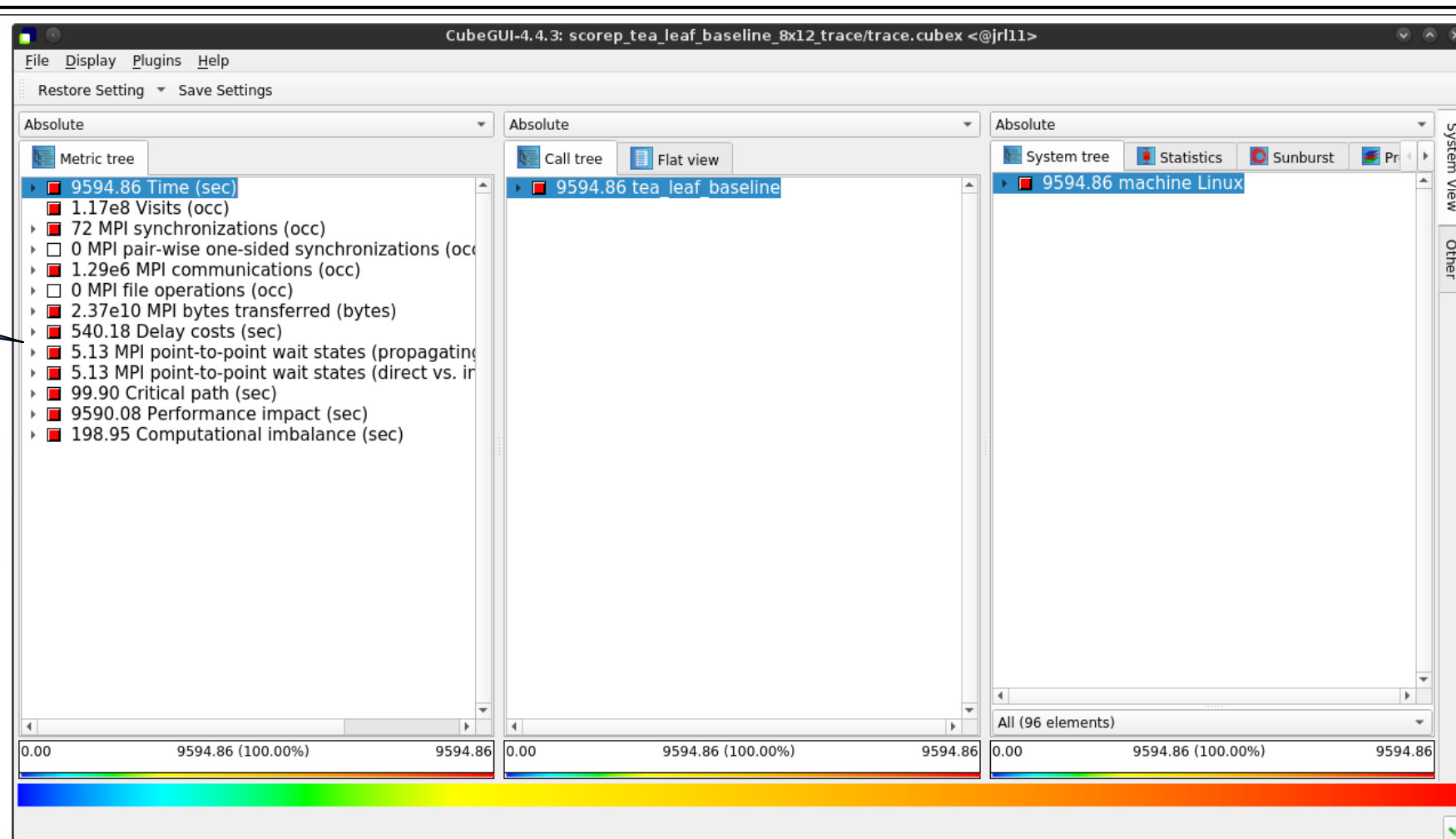


```
% cube scorep_tea_leaf_baseline_8x12_trace/trace.cubex  
[GUI showing post-processed trace analysis report]
```

Scalasca analysis report exploration (opening view)



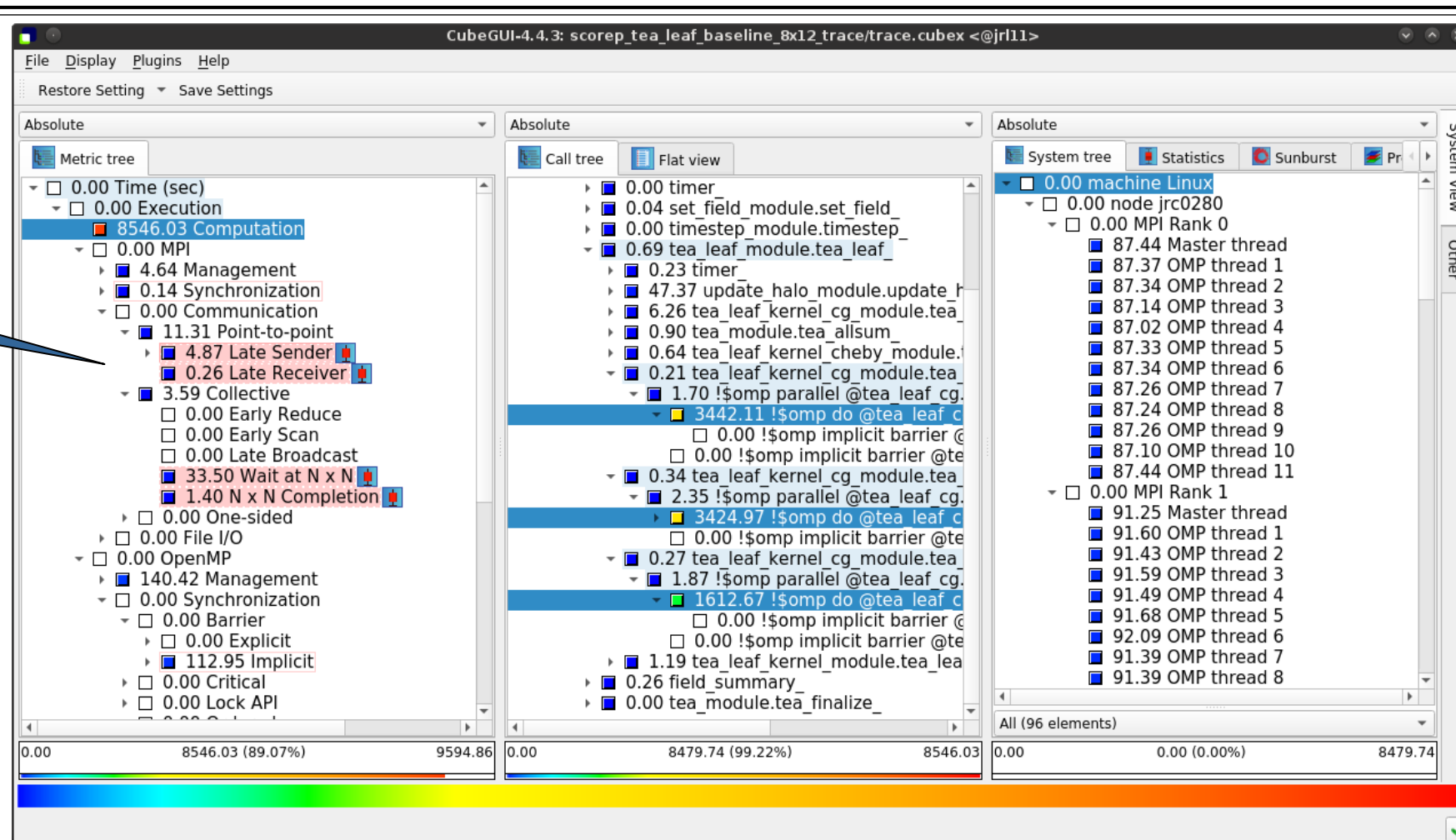
Additional top-level metrics produced by the trace analysis...



Scalasca wait-state metrics



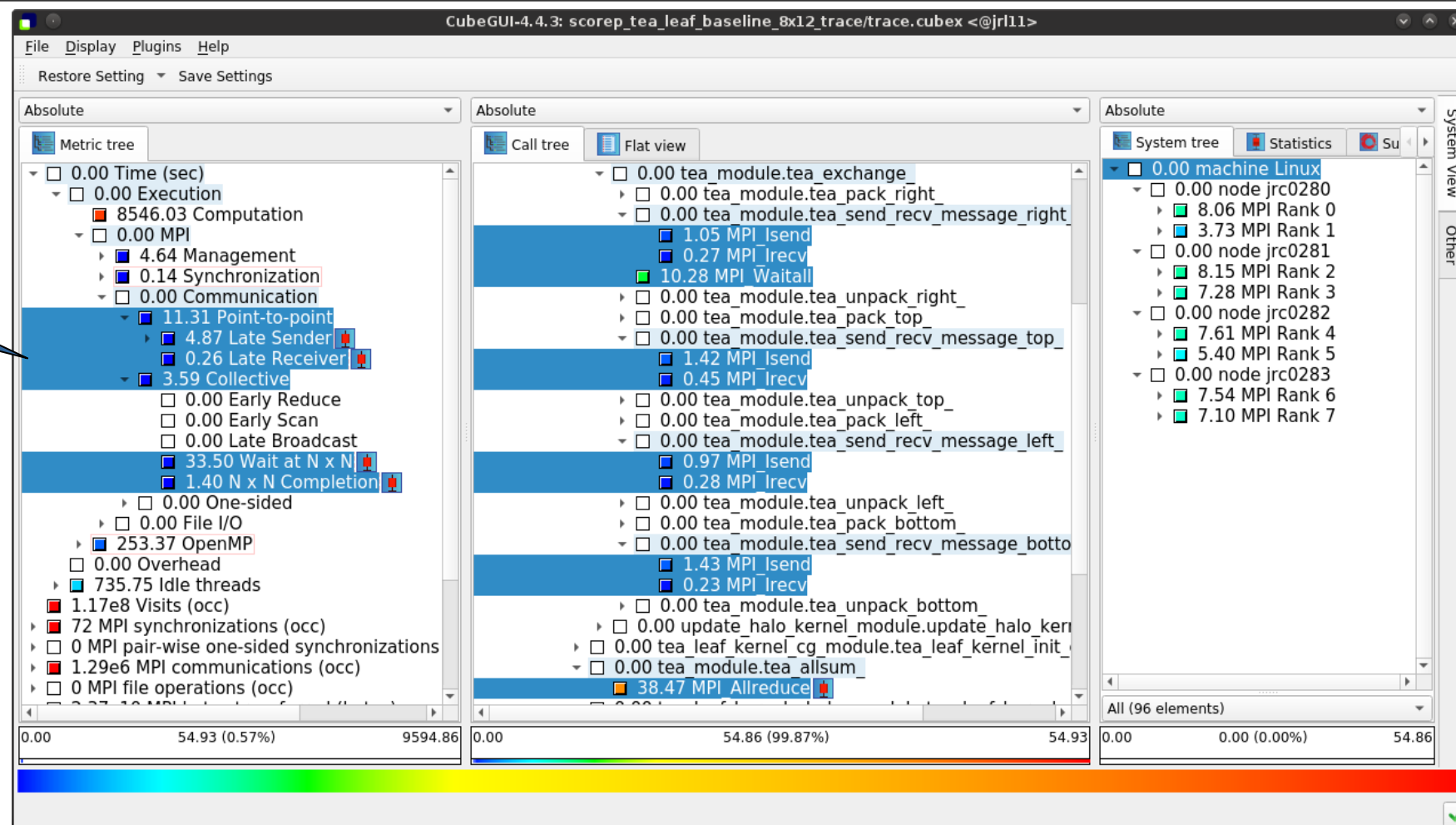
...plus additional wait-state metrics as part of the “Time” hierarchy



TeaLeaf Scalasca report analysis (I)



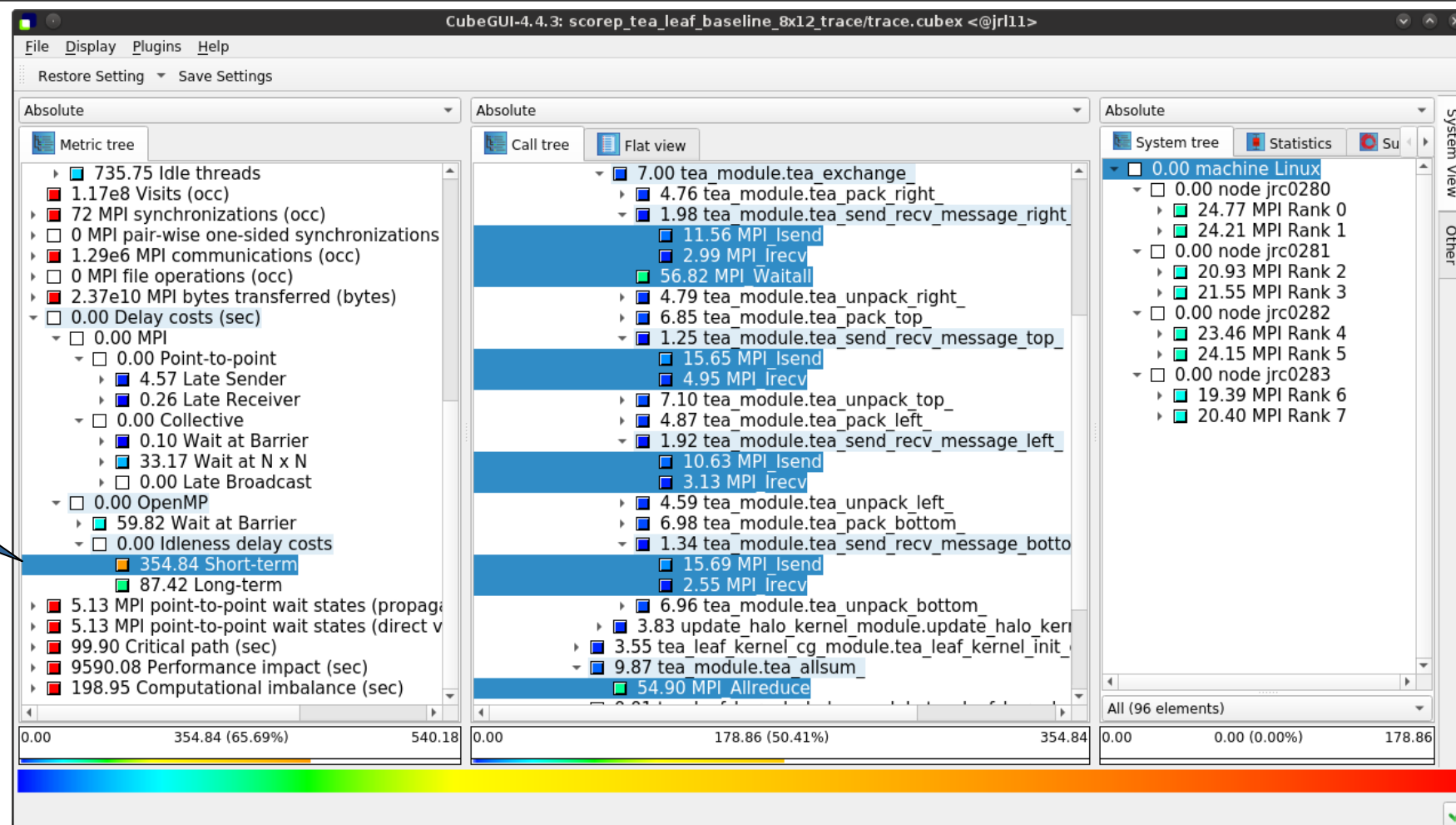
While MPI communication time and wait states are small (~0.6% of the total execution time)...



TeaLeaf Scalasca report analysis (II)



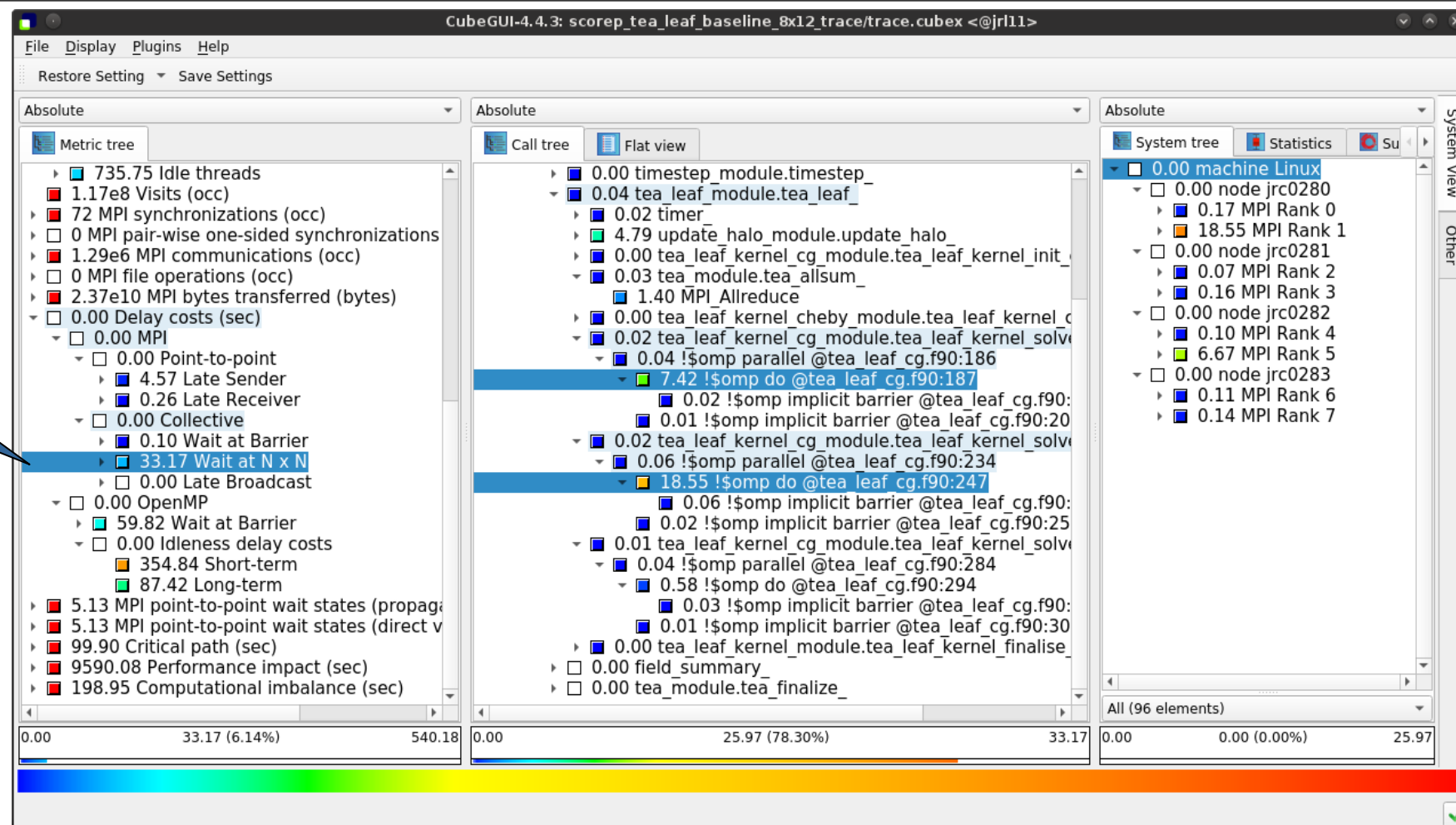
...they directly cause a significant amount of the OpenMP thread idleness



TeaLeaf Scalasca report analysis (III)

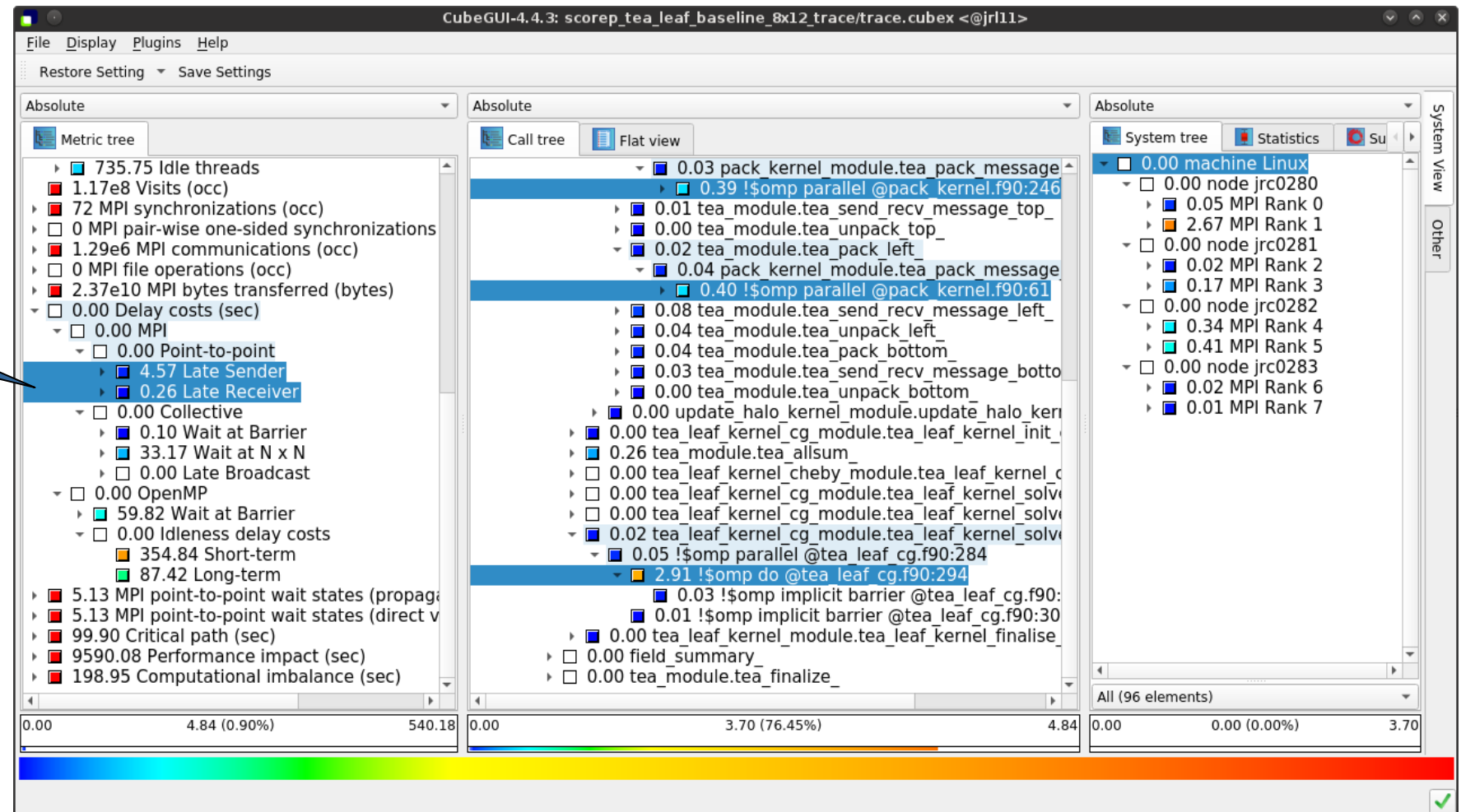


The “Wait at NxN” collective wait states are mostly caused by the first 2 OpenMP `do` loops of the solver (on ranks 5 & 1, resp.)...



TeaLeaf Scalasca report analysis (IV)

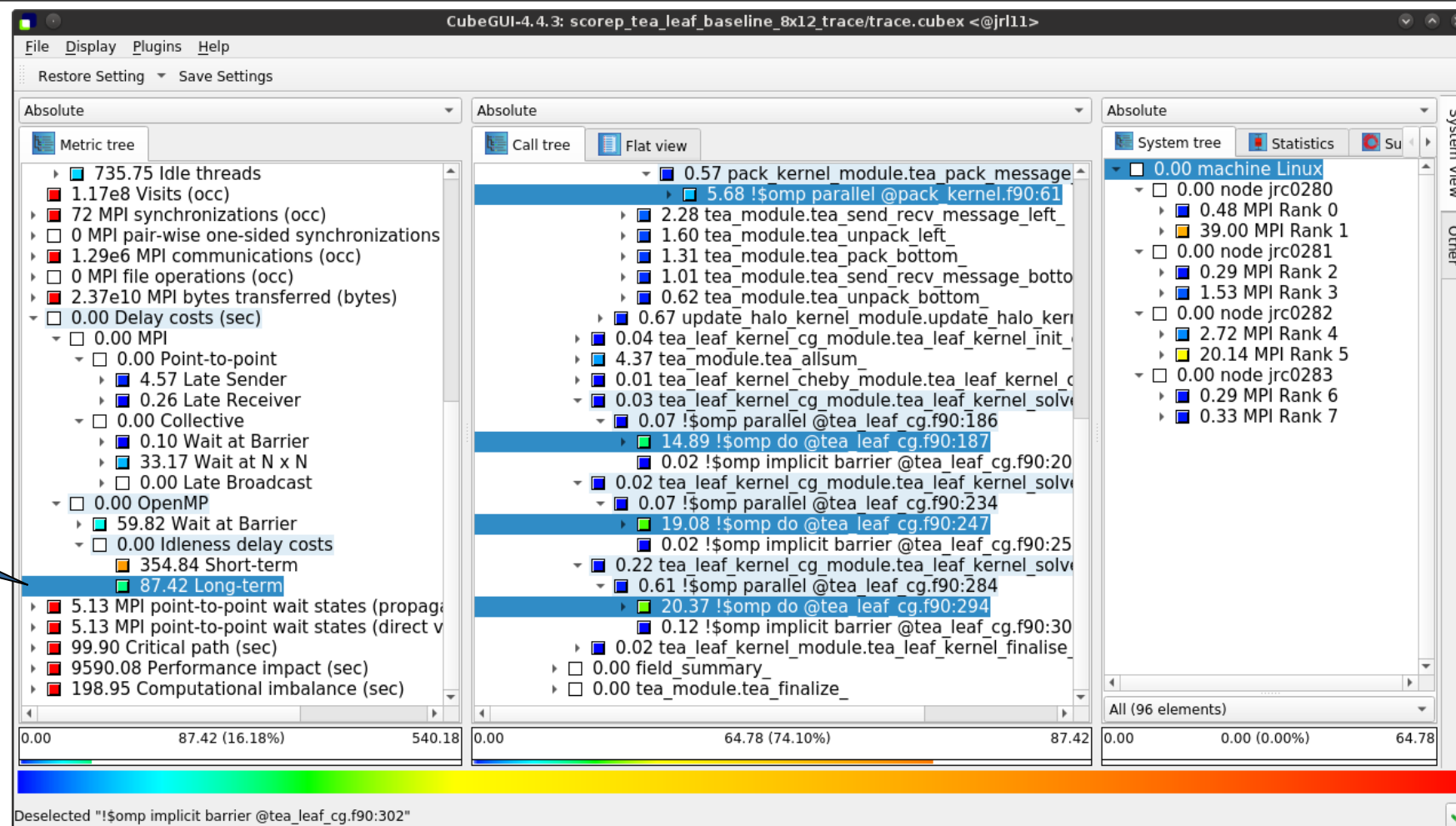
...while the MPI point-to-point wait states are caused by the 3rd solver do loop (on rank 1) and two loops in the halo exchange



TeaLeaf Scalasca report analysis (V)



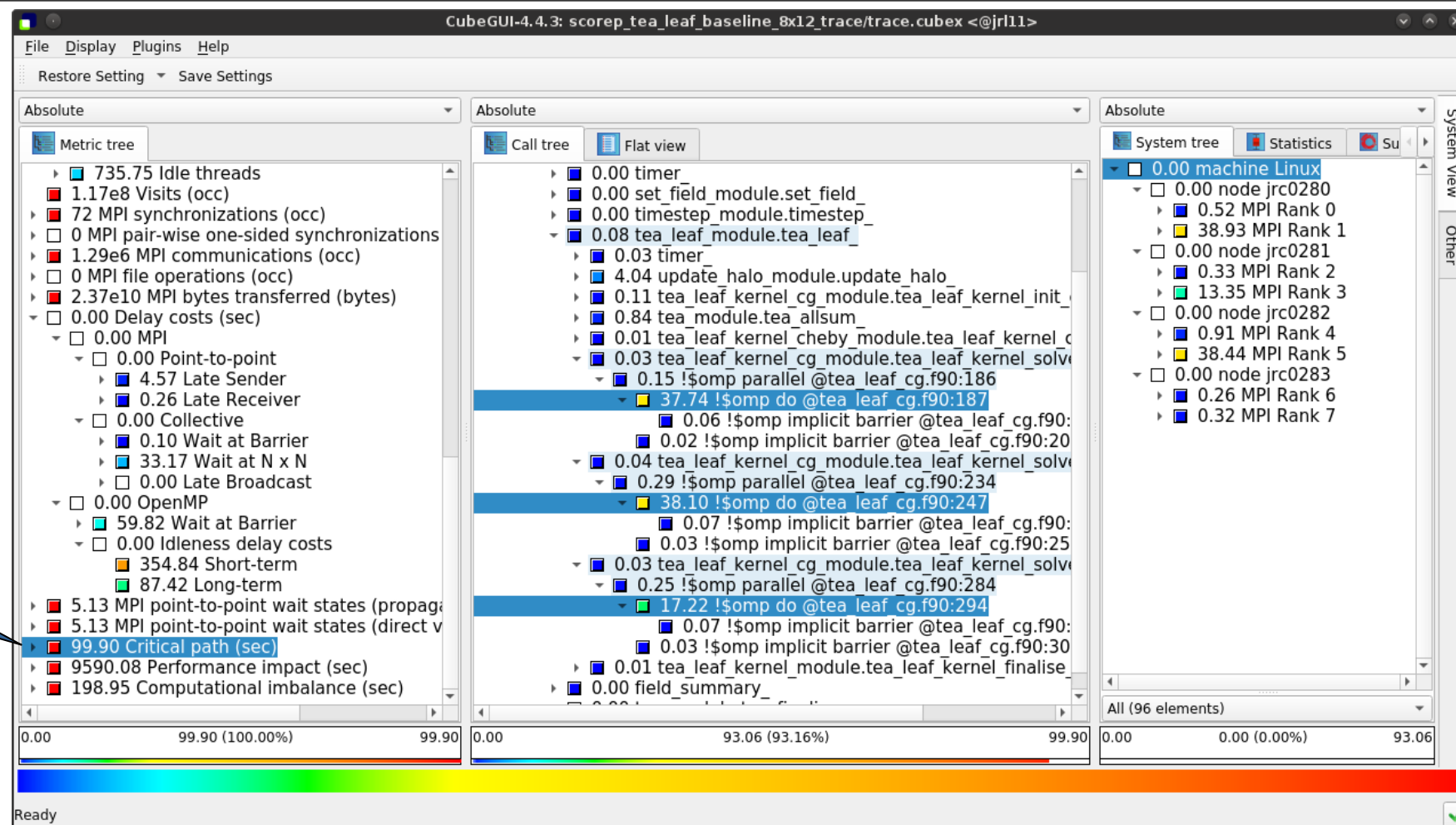
Various OpenMP `do` loops (incl. the solver loops) also cause OpenMP thread idleness on other ranks via propagation



TeaLeaf Scalasca report analysis (VI)



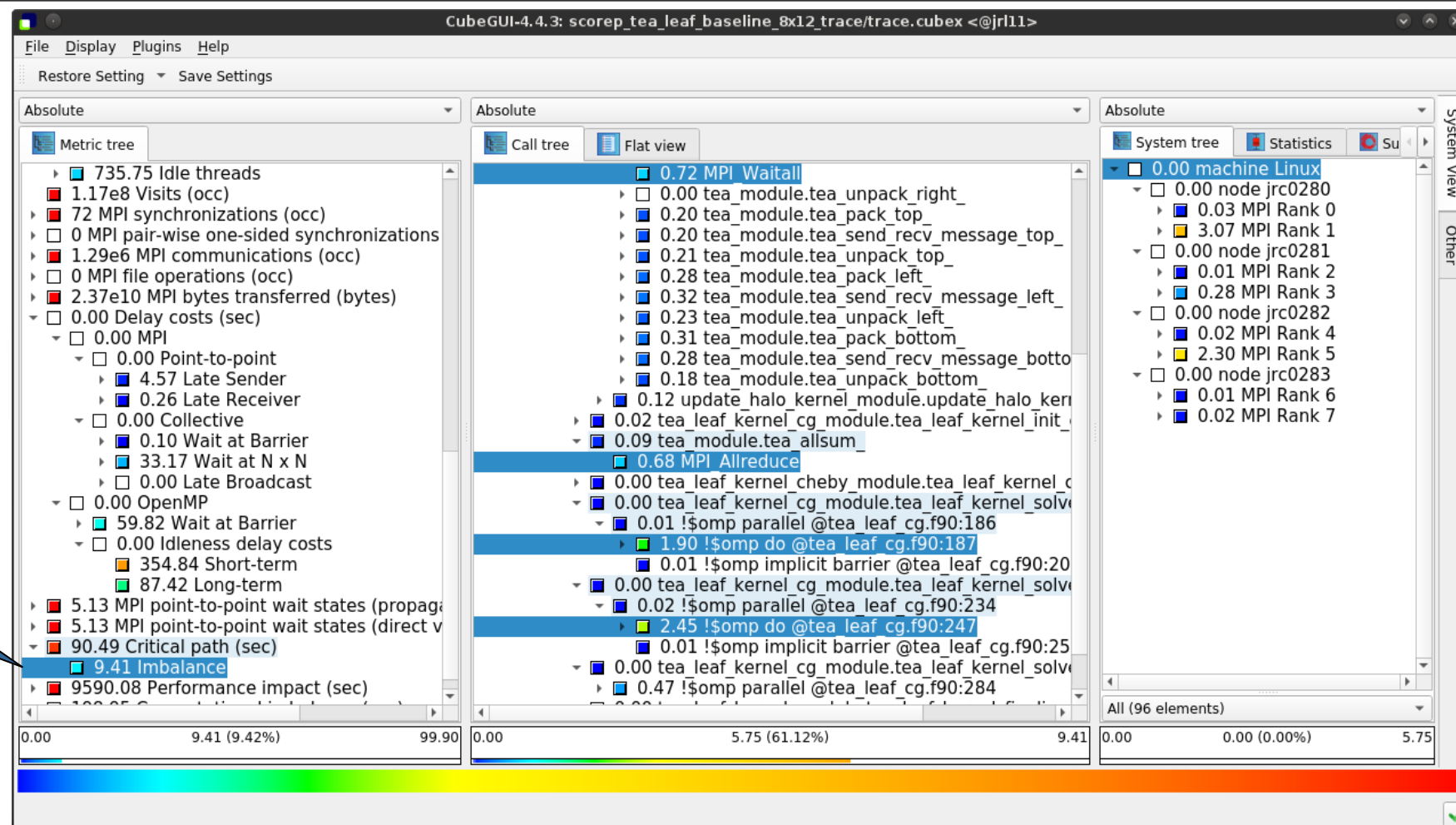
The Critical Path also highlights the three solver loops...

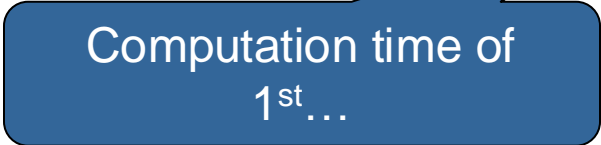


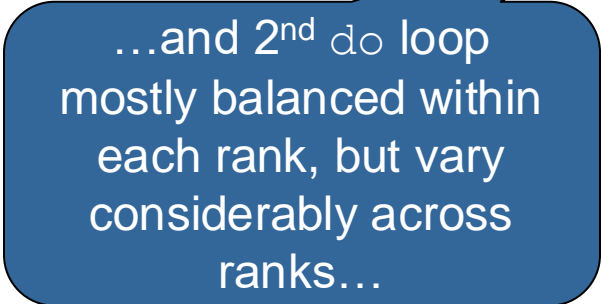
TeaLeaf Scalasca report analysis (VII)

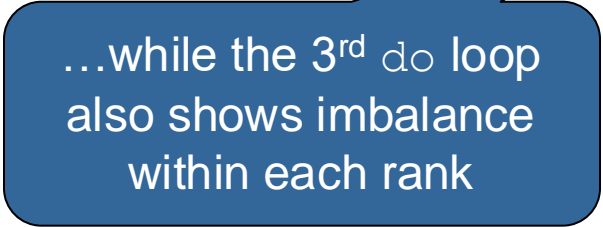


...with imbalance (time on critical path above average) mostly in the first two loops and MPI communication









TeaLeaf 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, OpenACC, POSIX threads, and hybrid MPI+OpenMP/OpenACC/Pthreads
 - host-side events only (no device/offloaded events) from OpenMP, OpenACC, OpenCL, HIP, CUDA
- 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

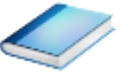


```
% scalasca
Scalasca 2.6.1
Toolset for scalable performance analysis of large-scale parallel applications
usage: scalasca [OPTION]... ACTION <argument>...
  1. prepare application objects and executable for measurement:
     scalasca -instrument <compile-or-link-command> # skin (using scorep)
  2. 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

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)

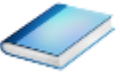
Scalasca convenience command: square / scalasca -examine



```
% 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

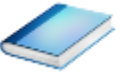


```
% scout.hyb --help
SCOUT      (Scalasca 2.6.1)
Copyright (c) 1998-2022 Forschungszentrum Juelich GmbH
Copyright (c) 2014-2021 RWTH Aachen University
Copyright (c) 2009-2014 German Research School for Simulation Sciences GmbH

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)

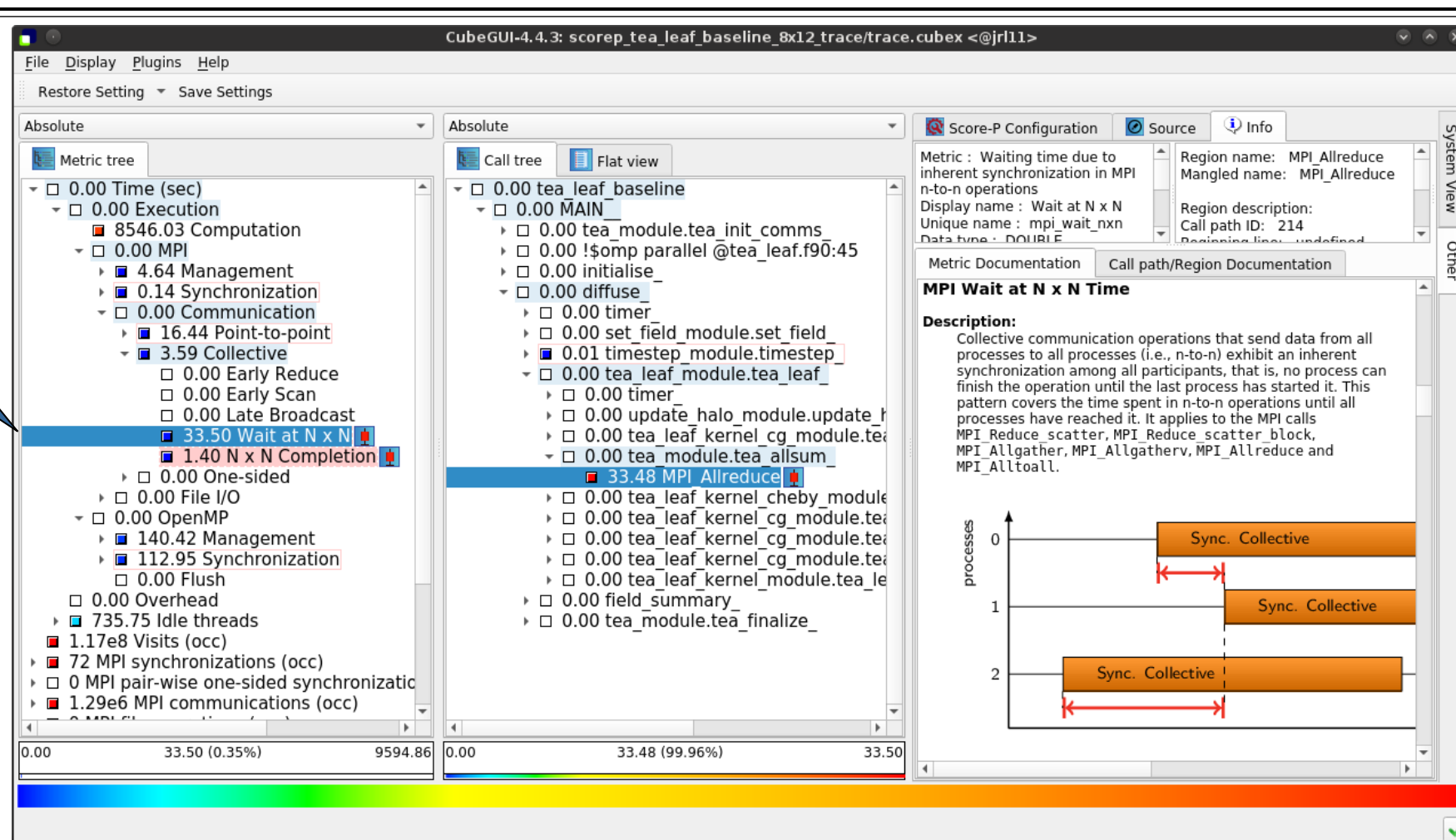
The screenshot displays the CubeGUI-4.4.3 interface for the 'scorep_tea_leaf_baseline_8x12_trace/trace.cubex' file. The 'Metric tree' panel on the left shows a hierarchical view of metrics. A context menu is open over the '33.50 Wait at N x N' item, with the 'Documentation' option selected. The 'Call tree' panel in the center shows the execution flow, and the 'System tree' panel on the right shows the system configuration. The bottom status bar indicates the current metric value and its percentage of the total.

Metric	Value	Percentage
33.50 Wait at N x N	33.50	0.35%

Online metric description (cont.)



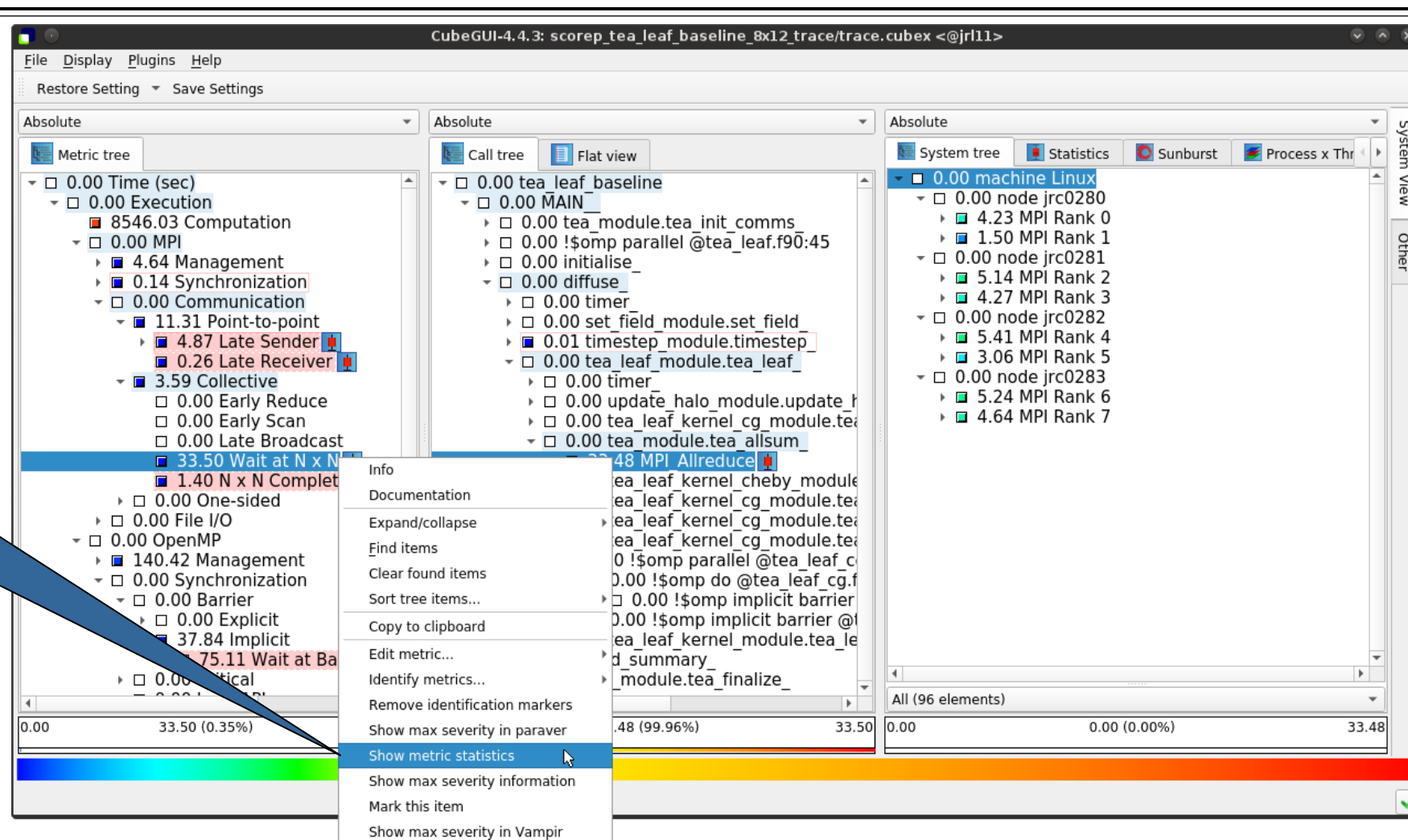
Selection of different metric automatically updates description



Metric statistics



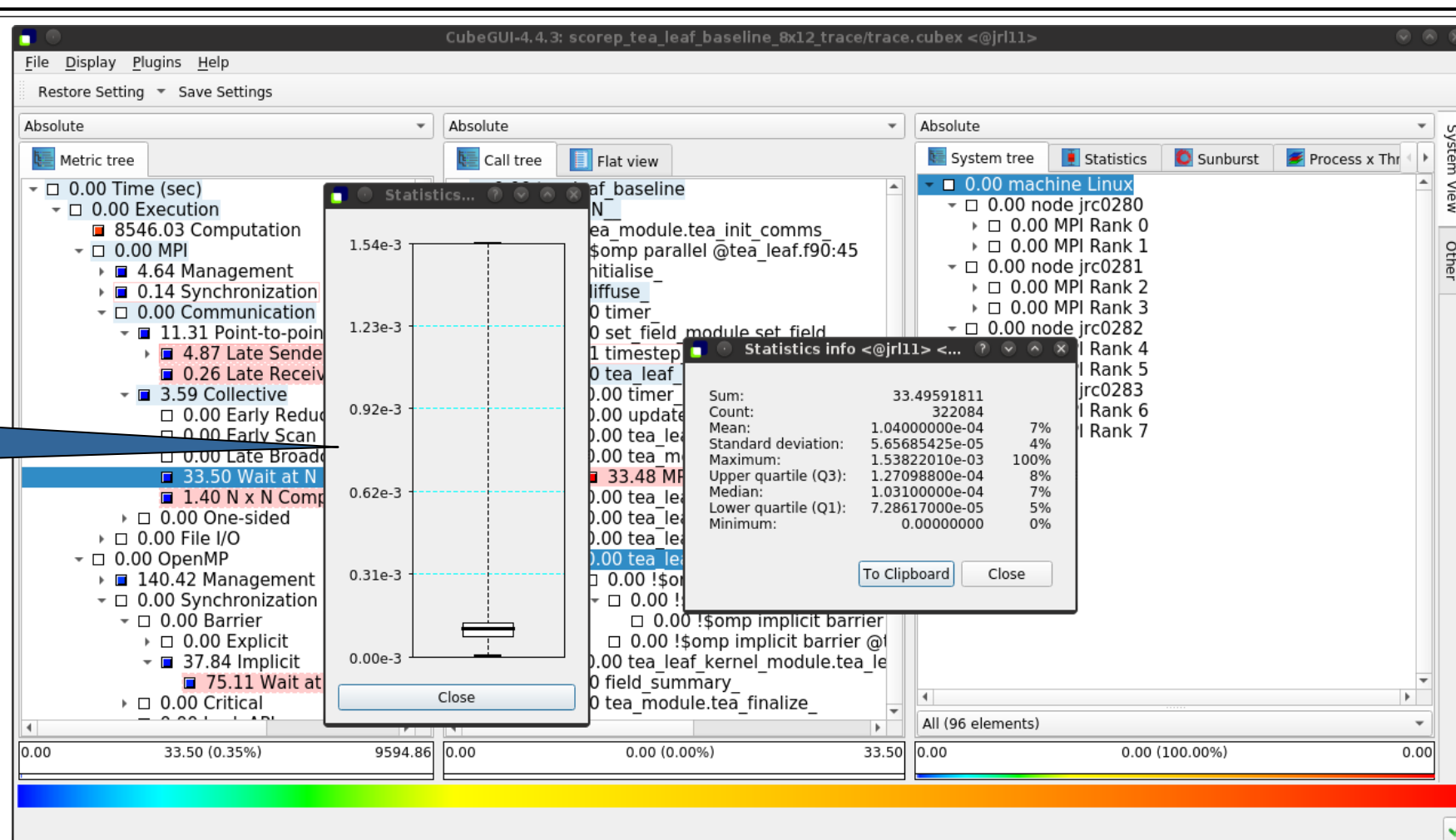
Access metric statistics for metrics marked with box plot icon from context menu



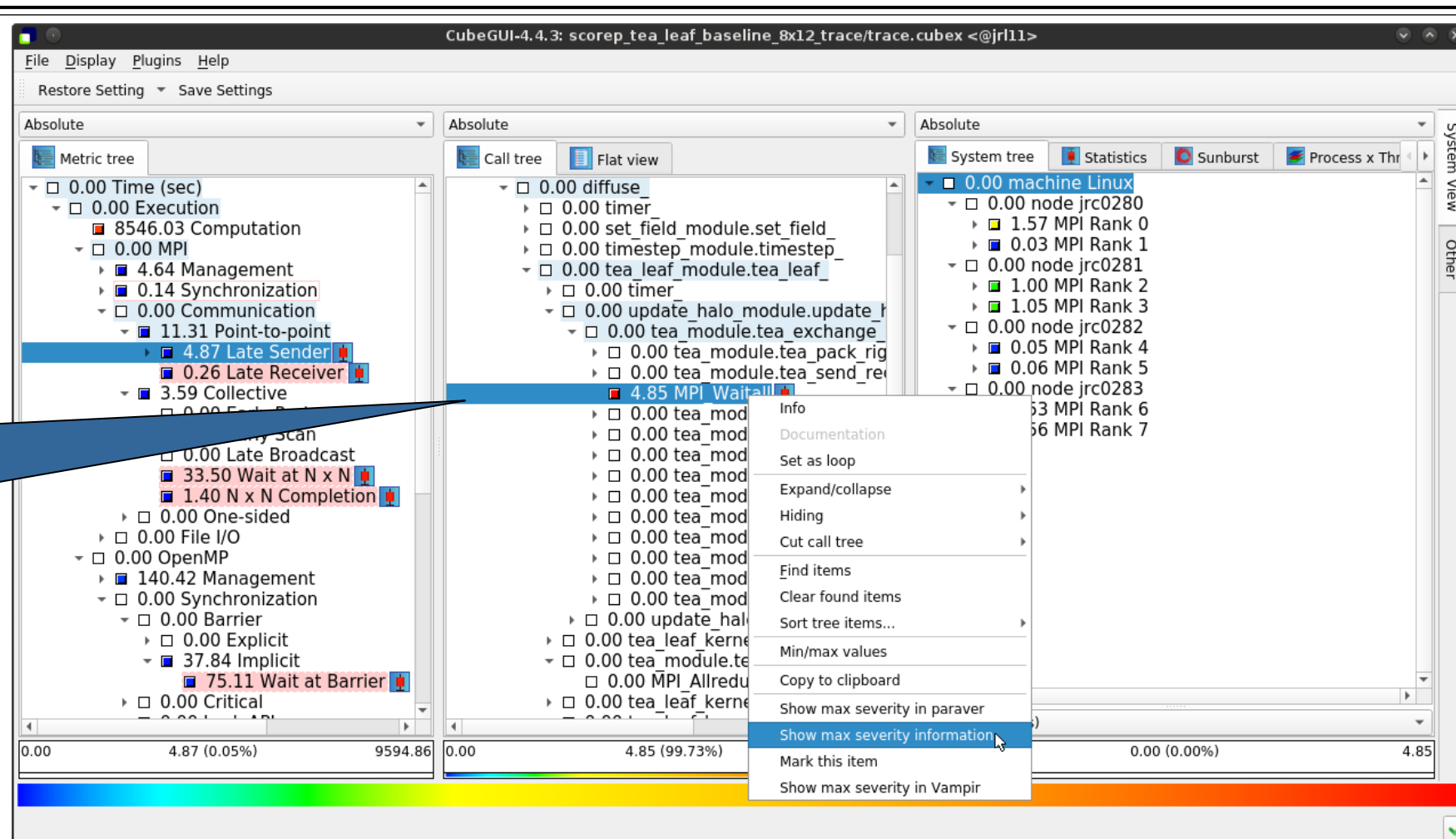
Metric statistics (cont.)



Shows instance statistics box plot, click to get details



Metric instance statistics





The screenshot displays the CubeGUI-4.4.3 interface, which is used for analyzing performance metrics. The interface is divided into three main panels: Metric tree, Call tree, and System tree. The Metric tree panel shows a hierarchy of metrics, with '4.87 Late Sender' highlighted. The Call tree panel shows a hierarchy of call paths, with '4.85 MPI Waitall' highlighted. The System tree panel shows a hierarchy of system components, with '0.03 MPI Rank 1' highlighted. A pop-up window titled 'Max severity <@jrl11>' displays the selected call path and its severity.

Metric tree:

- 0.00 Time (sec)
 - 0.00 Execution
 - 8546.03 Computation
 - 0.00 MPI
 - 4.64 Management
 - 0.14 Synchronization
 - 0.00 Communication
 - 11.31 Point-to-point
 - 4.87 Late Sender
 - 0.26 Late Receiver
 - 3.59 Collective
 - 0.00 Early Reduce
 - 0.00 Early Scan
 - 0.00 Late Broadcast
 - 1.40 N x N Completion
 - 0.00 One-sided
 - 0.00 File I/O
 - 0.00 OpenMP
 - 140.42 Management
 - 0.00 Synchronization
 - 0.00 Barrier
 - 0.00 Explicit
 - 37.84 Implicit
 - 75.11 Wait at Barrier
 - 0.00 Critical

Call tree:

 - 0.00 diffuse
 - 0.00 timer
 - 0.00 set_field_module.set_field
 - 0.00 timestep_module.timestep
 - 0.00 tea_leaf_module.tea
 - 0.00 timer
 - 0.00 update_halo_module
 - 0.00 tea_module.tea
 - 0.00 MPI Waitall

System tree:

 - 0.00 machine Linux
 - 0.00 node jrc0280
 - 1.57 MPI Rank 0
 - 0.03 MPI Rank 1

Max severity <@jrl11>

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

To Clipboard Close