

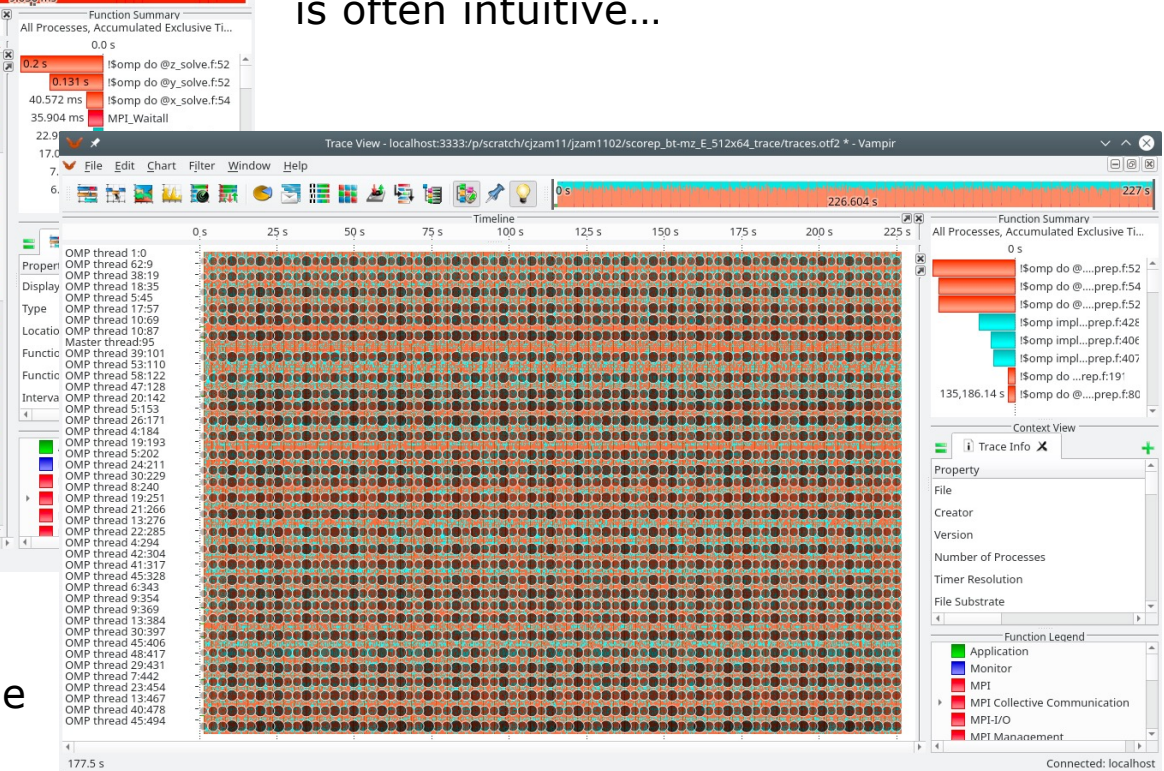
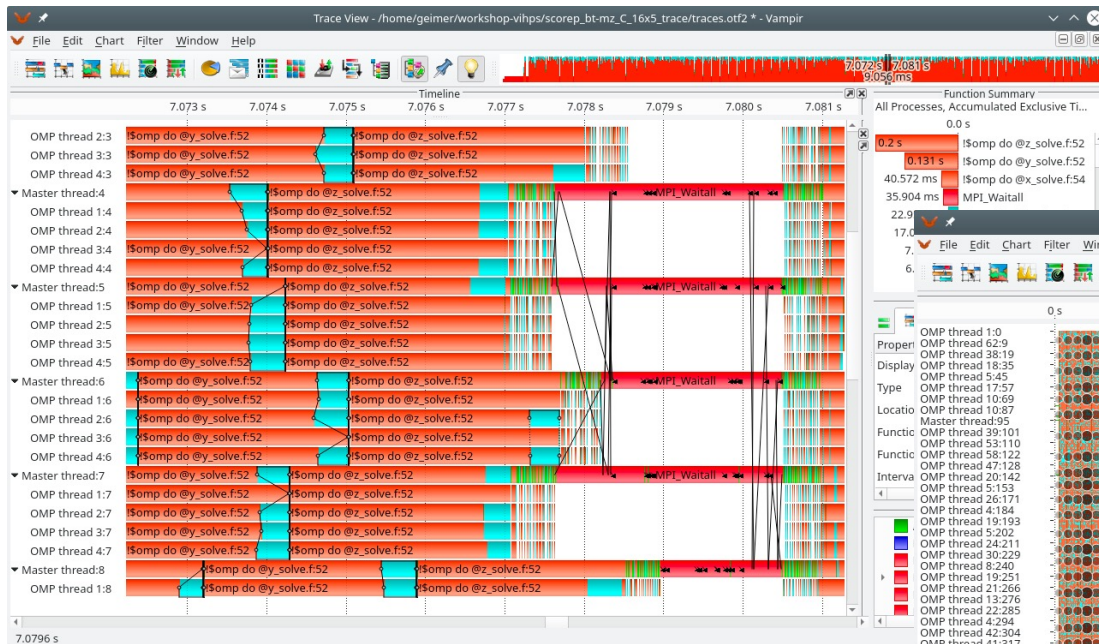
Scalasca Trace Tools

Automatic trace analysis



Motivation

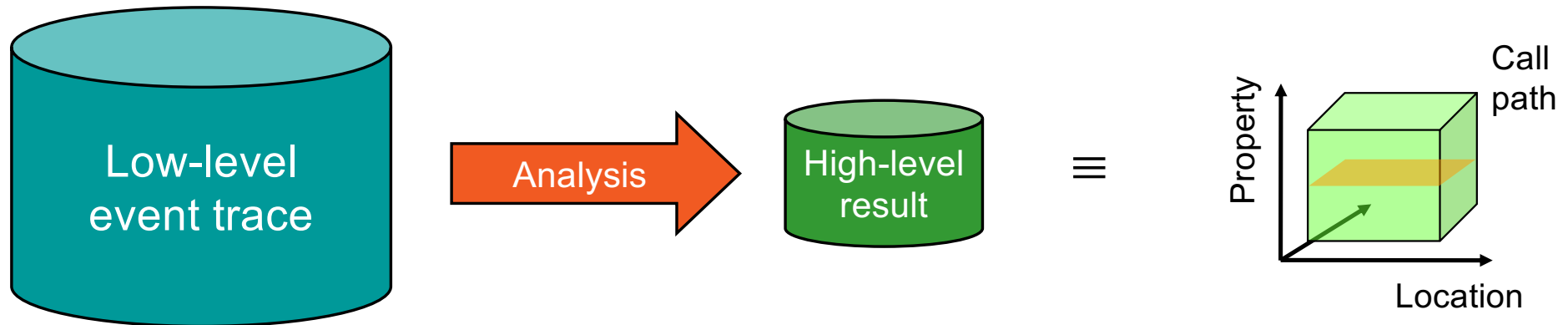
While interactive trace visualization is often intuitive...



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

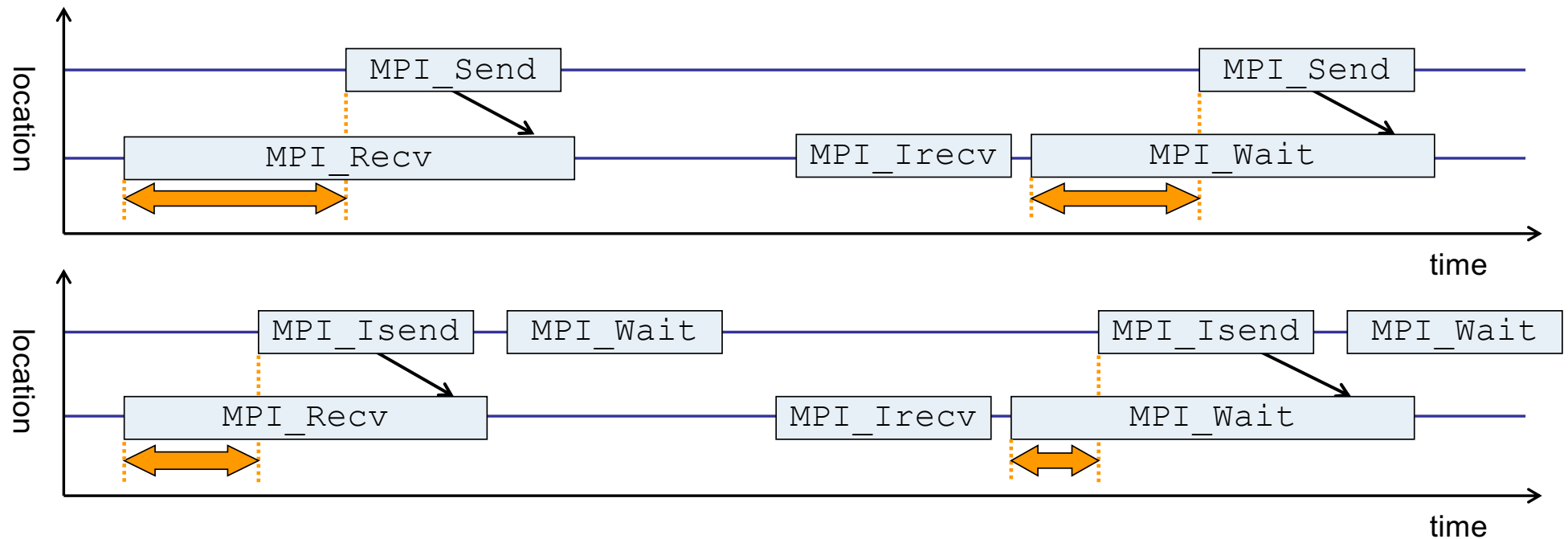
Idea: Automatic trace analysis

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



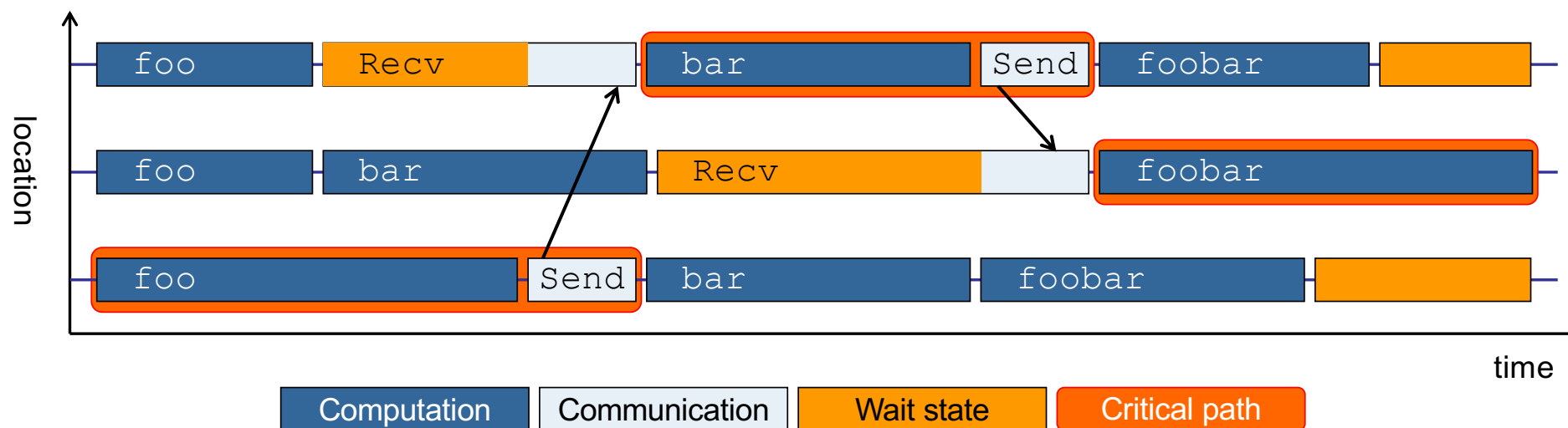
- Guaranteed to cover the entire event trace
- Quicker than manual/visual trace analysis
- Parallel replay analysis exploits available memory & processors to deliver scalability

Example: “Late Sender” wait state



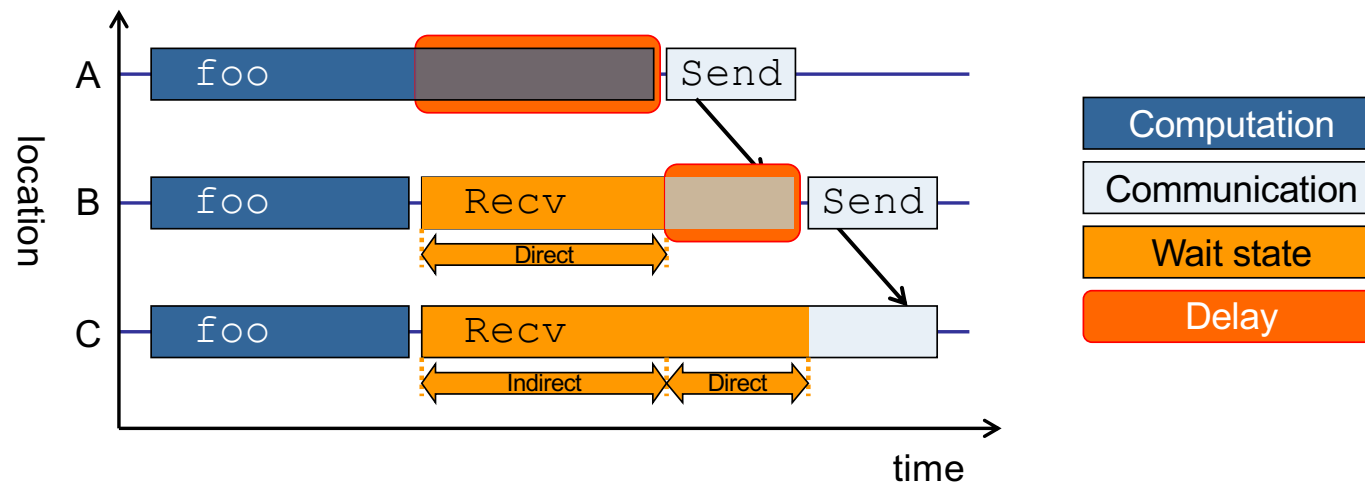
- Waiting time caused by a blocking receive operation posted earlier than the corresponding send
- Applies to blocking as well as non-blocking communication

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)

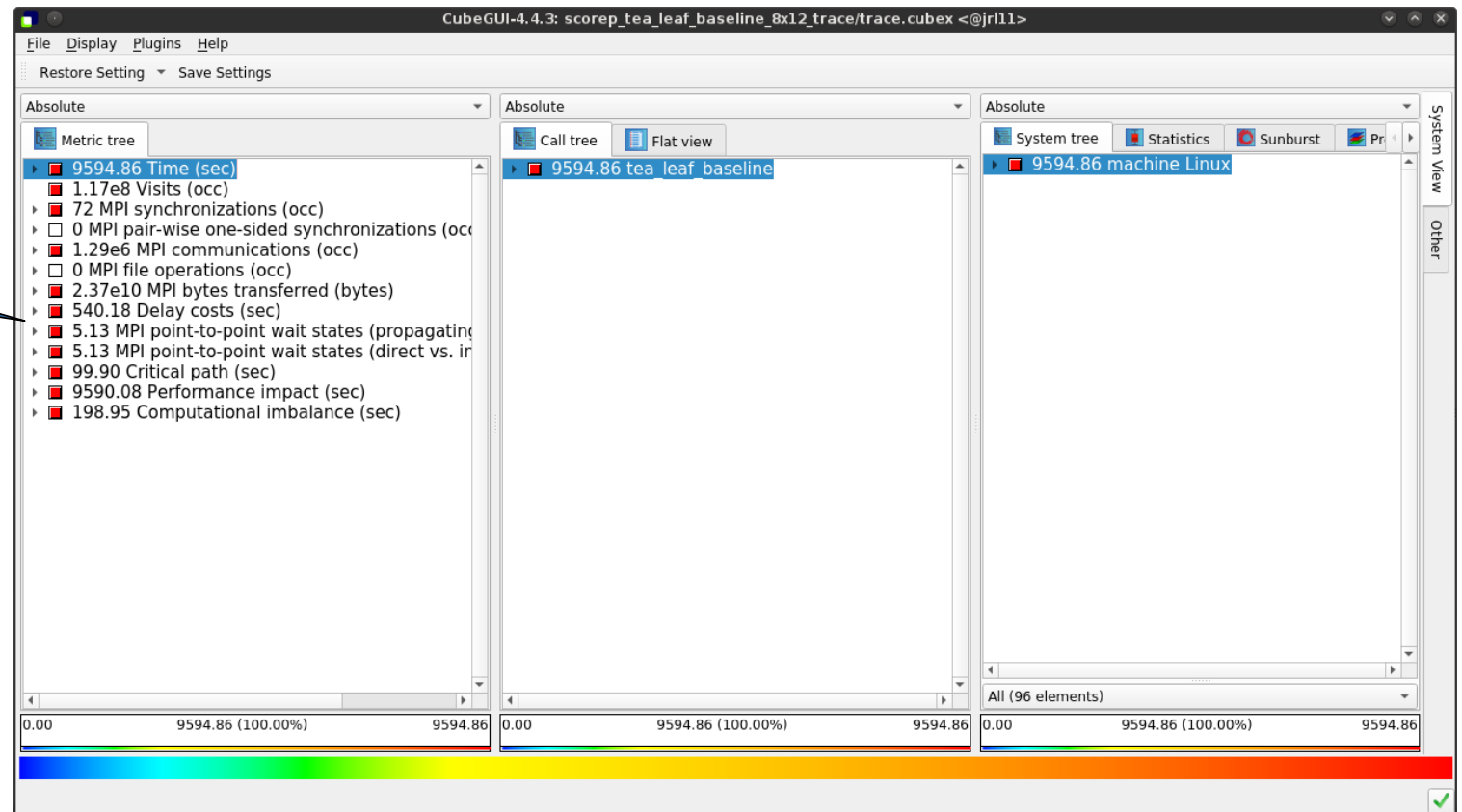
Hands-On

- Single link to the hands-on exercises
 - **<https://go.fzj.de/tw45>**
- Let's collect traces for BT-MZ and analyze them with Scalasca
 - Follow "Scalasca trace analysis" in the hands-on
 - Examine results with CUBE



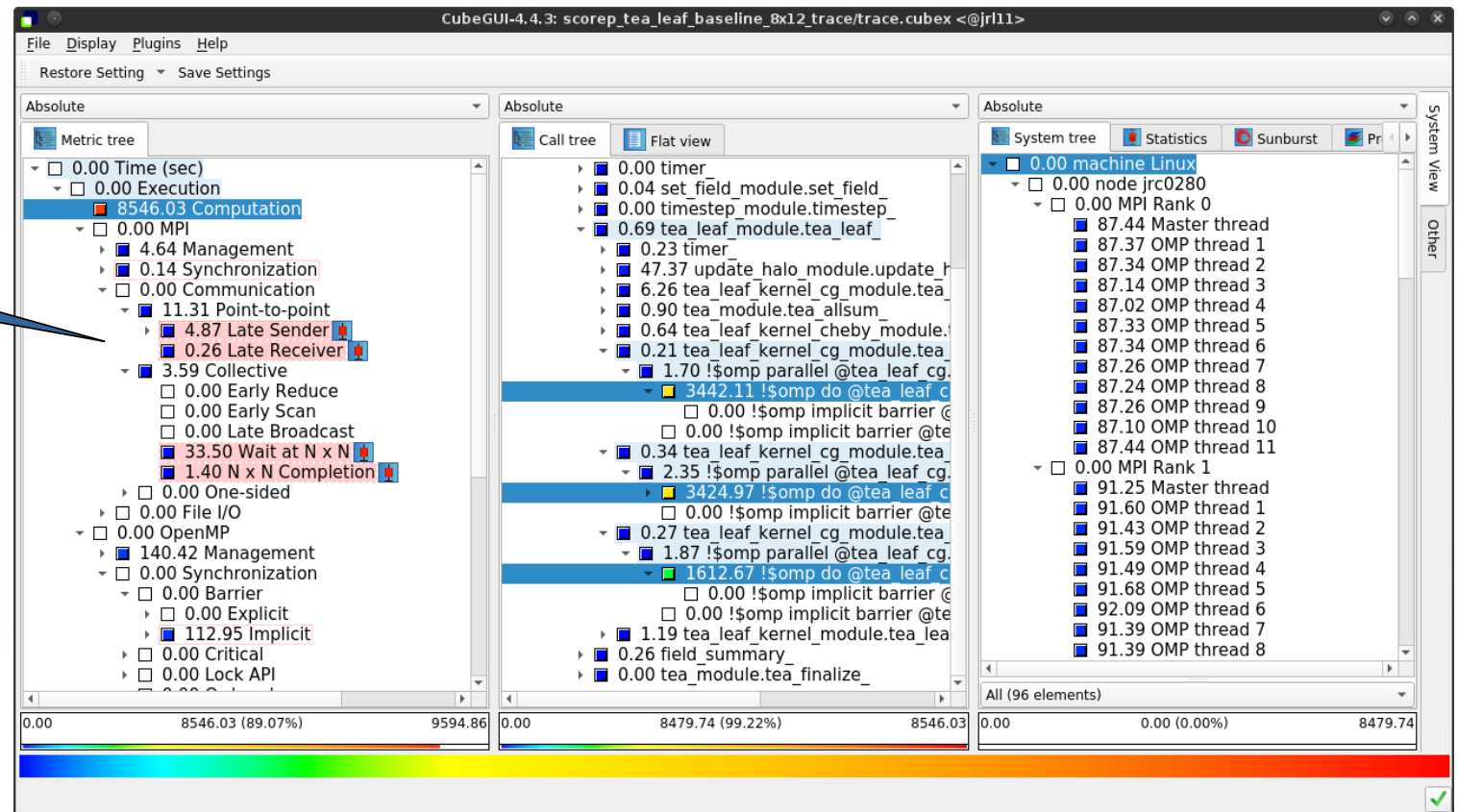
Scalasca analysis report exploration (opening view)

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

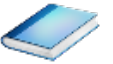


Scalasca wait-state metrics

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



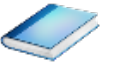
Online metric description



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

The screenshot shows the CubeGUI-4.4.3 interface with a performance analysis window titled 'scorep_tea_leaf_baseline_8x12_trace/trace.cubex <@jrl11>'. The window is divided into three main panes: 'Metric tree', 'Call tree', and 'System tree'. The 'Metric tree' pane on the left shows a hierarchical view of performance metrics. A context menu is open over the '33.50 Wait at N x N' metric, with the 'Documentation' option selected. The 'Call tree' pane in the middle shows a detailed view of the selected metric, including sub-metrics like 'timer' and 'update_halo_module.update_halo'. The 'System tree' pane on the right shows a view of the system components, including nodes and MPI ranks. At the bottom of the window, there are several data rows and a color-coded bar representing the distribution of metrics.

Online metric description (cont.)

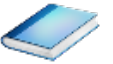


Selection of different metric automatically updates description

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

- Metric tree (Left):** A hierarchical tree of metrics. The 'Wait at N x N' metric is selected and highlighted in blue. Below it, 'MPI Allreduce' is also highlighted.
- Call tree (Center):** A tree view showing the call path for the selected metric. The 'MPI Allreduce' node is highlighted in blue.
- Metric Description (Right):** A detailed view of the 'MPI Wait at N x N Time' metric. It includes:
 - Region name:** MPI Allreduce
 - Mangled name:** MPI_Allreduce
 - Call path ID:** 214
 - 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.
 - Diagram:** A Gantt-style chart showing three processes (0, 1, 2) on the y-axis. Each process has a horizontal bar representing its execution. Red double-headed arrows indicate the synchronization period for each process, labeled 'Sync. Collective'.

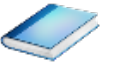
Metric statistics



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

The screenshot shows 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 performance metrics. A context menu is open over a metric labeled '33.50 Wait at N x N', with the 'Show metric statistics' option highlighted. The 'Call tree' panel in the middle shows a detailed view of the selected metric's execution path. The 'System tree' panel on the right shows a view of the system components, including nodes and MPI ranks. A status bar at the bottom of the interface displays 'All (96 elements)' and a progress indicator.

Metric statistics (cont.)



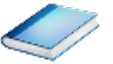
Shows instance statistics box plot, click to get details

The screenshot shows the CubeGUI interface with the following components:

- Metric tree (left):** A hierarchical tree of metrics. The 'Wait at N' metric is highlighted in blue, with a value of 33.50.
- Box plot (center):** A box plot for the 'Wait at N' metric. The y-axis ranges from 0.00e-3 to 1.54e-3. The plot shows a median around 0.31e-3, with whiskers extending to approximately 0.62e-3 and 1.23e-3.
- Statistics info window (right):** A window titled 'Statistics info' displaying the following data:

Sum:	33.49591811
Count:	322084
Mean:	1.04000000e-04 7%
Standard deviation:	5.65685425e-05 4%
Maximum:	1.53822010e-03 100%
Upper quartile (Q3):	1.27098800e-04 8%
Median:	1.03100000e-04 7%
Lower quartile (Q1):	7.28617000e-05 5%
Minimum:	0.00000000 0%
- System tree (right):** A tree view showing the system hierarchy, including 'machine Linux', 'node jrc0280', and 'MPI Rank 0-7'.
- Bottom bar:** A horizontal bar with a color gradient from blue to red, showing values for different metrics.

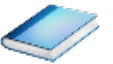
Metric instance statistics



The screenshot displays the CubeGUI-4.4.3 interface with three main panels: 'Metric tree', 'Call tree', and 'System tree'. The 'Metric tree' on the left shows a hierarchy of metrics, with several items marked with box plot icons indicating severity. The 'Call tree' in the center shows a detailed view of a selected metric, with a context menu open over it. The 'System tree' on the right shows the overall system structure. A blue callout bubble points to the 'Late Sender' metric in the 'Metric tree'.

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

Metric instance statistics (cont.)



Shows instance details

The screenshot shows the CubeGUI interface with three main panels: 'Metric tree', 'Call tree', and 'System tree'. The 'Metric tree' on the left shows a hierarchy of metrics, with '4.87 Late Sender' selected. The 'Call tree' in the center shows the call path for the selected metric, with '4.85 MPI Waitall' selected. The 'System tree' on the right shows the system hierarchy, with '0.03 MPI Rank 1' selected. A pop-up window titled 'Max severity <@jrl11>' displays the following details:

```

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
    
```

At the bottom of the interface, there are three summary rows for the selected metrics:

0.00	4.87 (0.05%)	9594.86
0.00	4.85 (99.73%)	4.87
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