

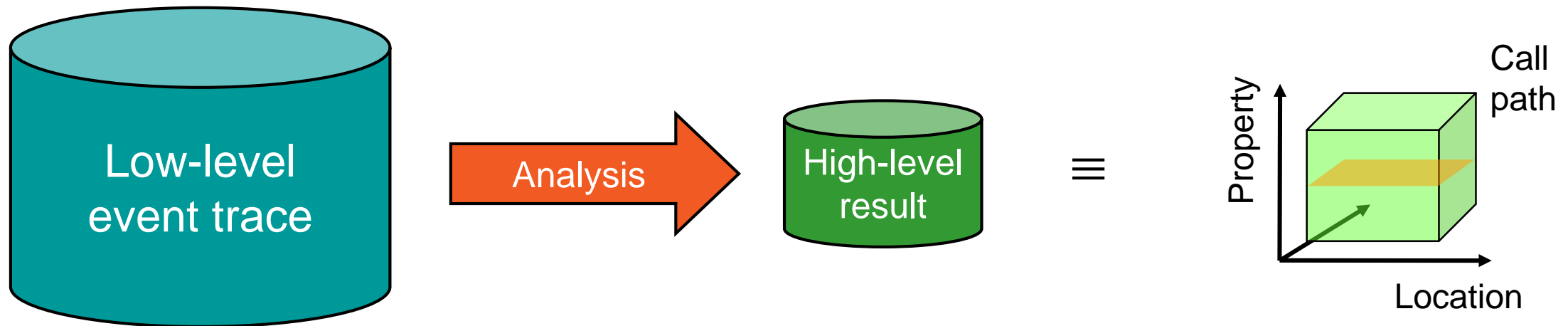
Automatic trace analysis with the Scalasca Trace Tools

Markus Geimer
Jülich Supercomputing Centre



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

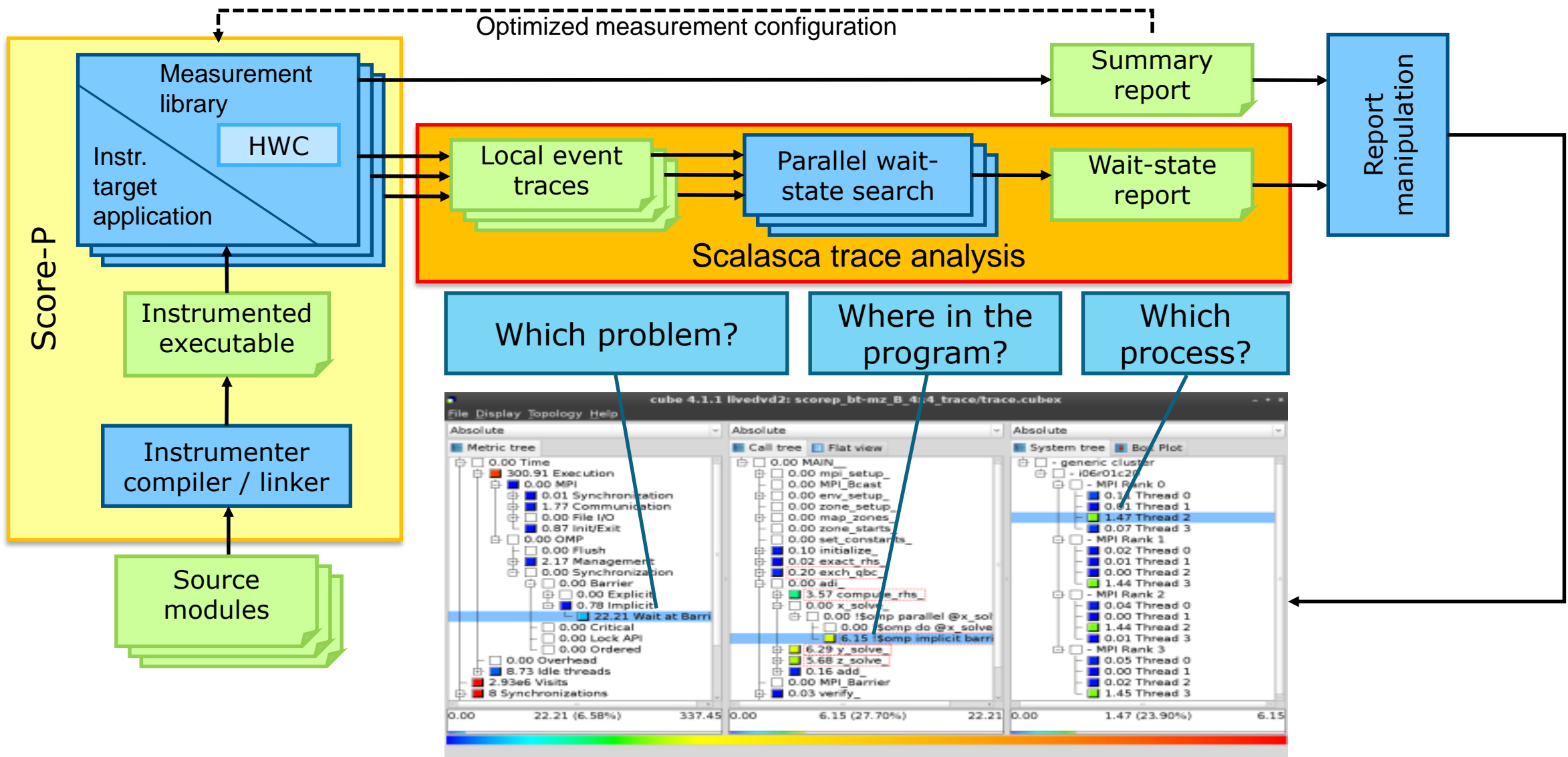
Scalasca Trace Tools: Objective

- Development of a **scalable trace-based** performance analysis toolset for the most popular parallel programming paradigms
 - Current focus: MPI, OpenMP, and POSIX threads
- Specifically targeting large-scale parallel applications
 - Such as those running on IBM Blue Gene or Cray systems with one million or more processes/threads
- Latest release:
 - Scalasca v2.3.1 coordinated with Score-P v2.0.2 (May 2016)

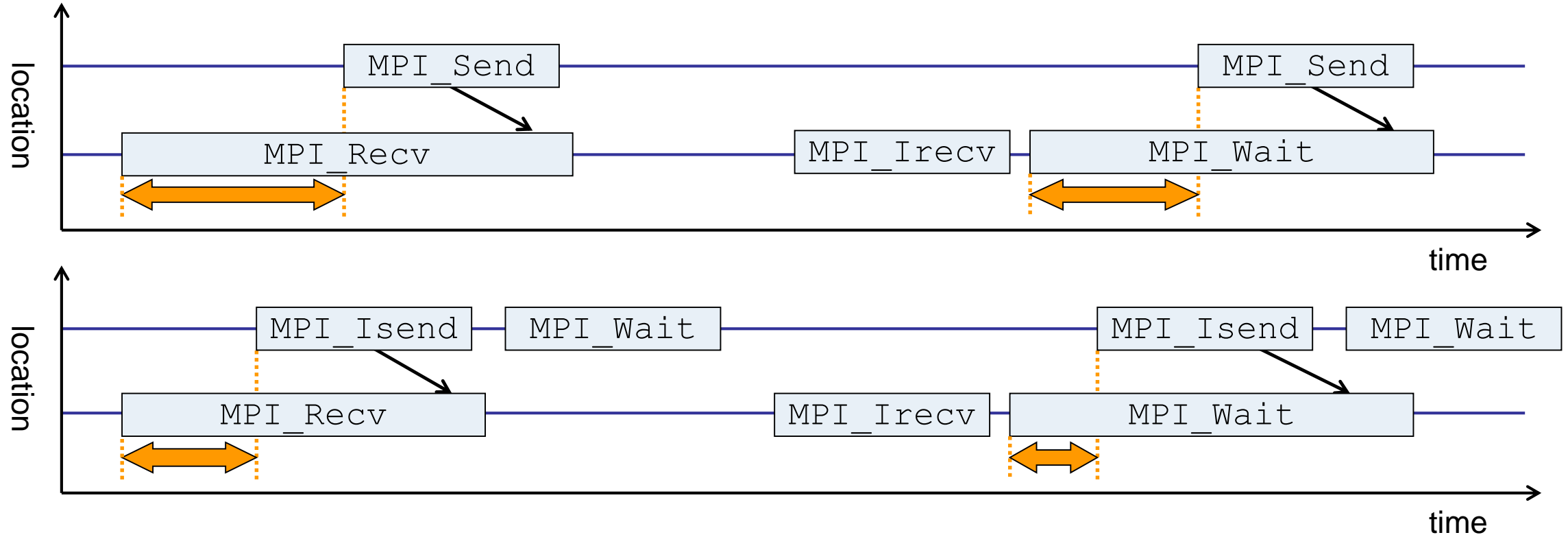
Scalasca Trace Tools features

- Open source, 3-clause BSD license
- Fairly portable
 - IBM Blue Gene, Cray XT/XE/XK/XC, SGI Altix, Fujitsu FX10/100 & K computer, Linux clusters (x86, Power, ARM), Intel Xeon Phi, ...
- Uses Score-P instrumenter & measurement libraries
 - Scalasca v2 core package focuses on trace-based analyses
 - 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 CUDA or SHMEM events, or OpenMP nested parallelism
 - PAPI/rusage metrics for trace events are ignored

Scalasca workflow

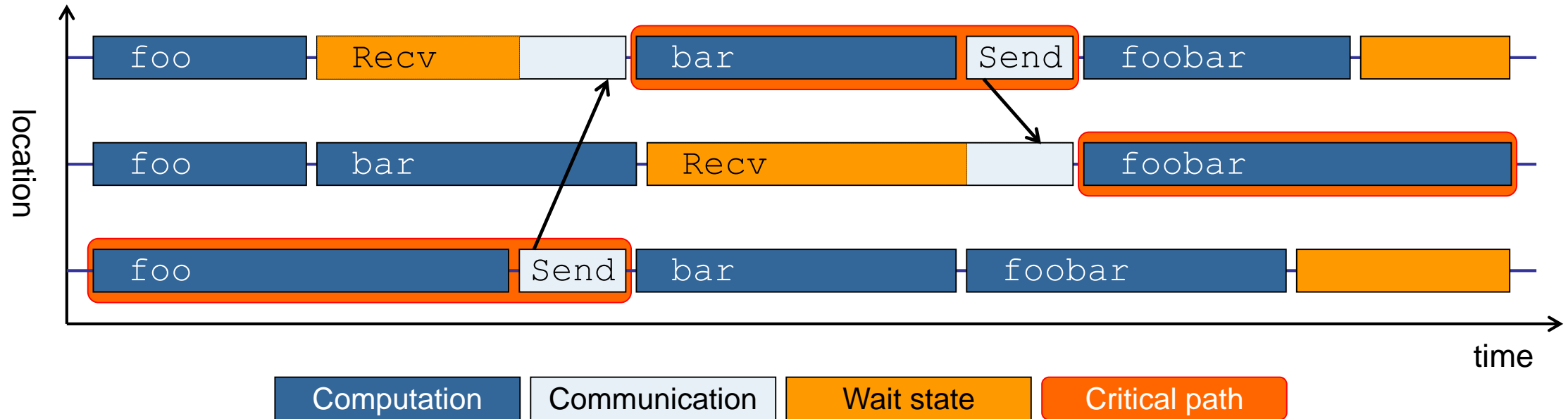


Example: “Late Sender” wait state



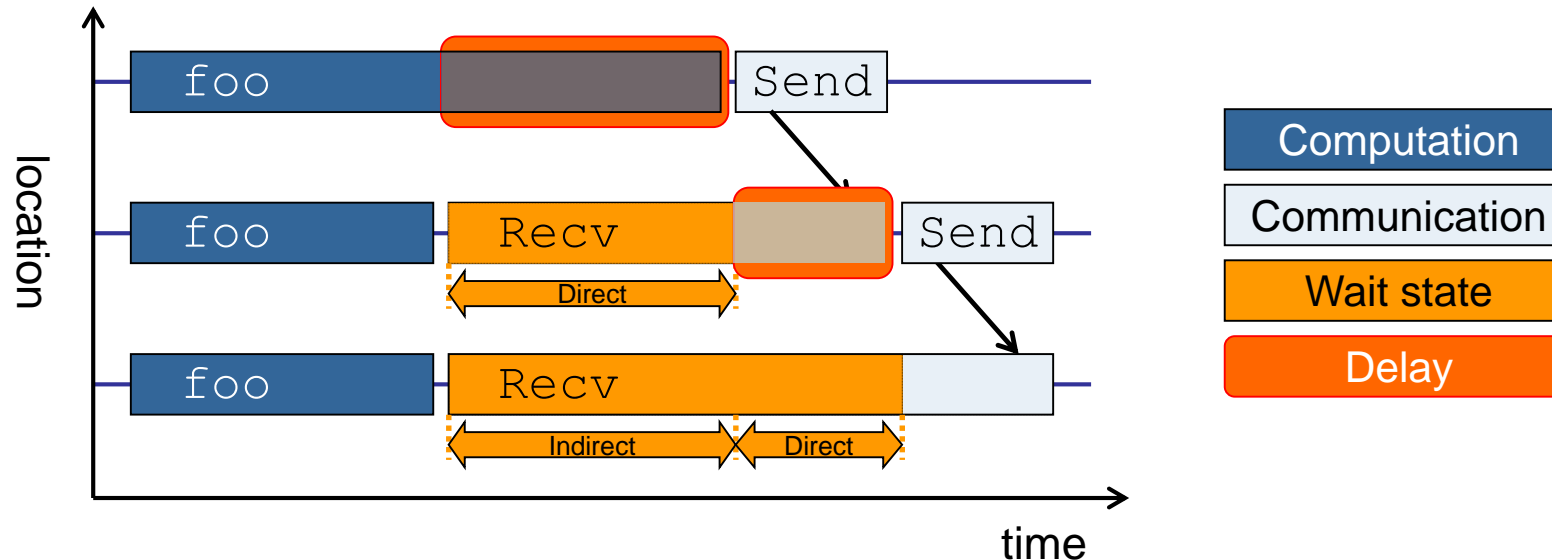
- Waiting time caused by a blocking receive operation posted earlier than the corresponding send
- Applies to blocking as well as non-blocking communication

Example: Critical path

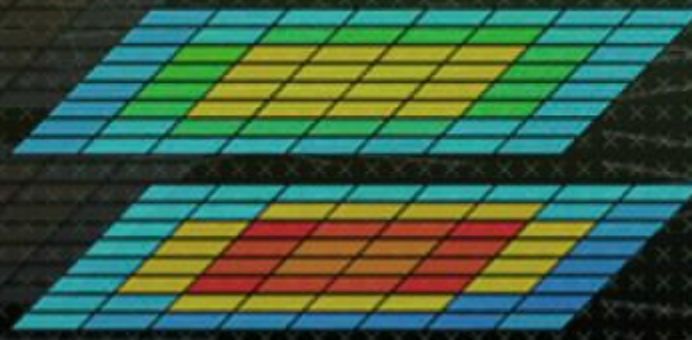


- Shows call paths and processes/threads that are responsible for the program's wall-clock runtime
- Identifies good optimization candidates and parallelization bottlenecks

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



Local setup

- Load environment modules
 - Required for each shell session

```
% module load scorep/3.0-bullxmpi_mlx-intel16  
% module load scalasca/2.3.1-bullxmpi_mlx-intel16
```

- Important:
 - Some Scalasca commands have a run-time dependency on Score-P
 - Thus, make sure to also have the Score-P module loaded when using Scalasca

Scalasca command – One command for (almost) everything

```
% scalasca
Scalasca 2.3.1
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 compatibility command: skin / scalasca -instrument

```
% skin
Scalasca 2.3.1: application instrumenter (using Score-P instrumenter)
usage: skin [-v] [-comp] [-pdt] [-pomp] [-user] [--*] <compile-or-link-command>
  -comp={all|none|...}: routines to be instrumented by compiler [default: all]
                        (... custom instrumentation specification depends on compiler)
  -pdt:  process source files with PDT/TAU instrumenter
  -pomp: process source files for POMP directives
  -user: enable EPIK user instrumentation API macros in source code
  -v:    enable verbose commentary when instrumenting

  --*:   options to pass to Score-P instrumenter
```

- Scalasca application instrumenter
 - Provides compatibility with Scalasca 1.x
 - **Deprecated! Use Score-P instrumenter directly.**

Scalasca convenience command: scan / scalasca -analyze

```
% scan
Scalasca 2.3.1: 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.
-m metrics    : Metric specification for measurement.
```

- Scalasca measurement collection & analysis nexus

Scalasca advanced command: scout - Scalasca automatic trace analyzer

```
% scout.hyb --help
SCOUT Copyright (c) 1998-2016 Forschungszentrum Juelich GmbH
      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

Scalasca convenience command: square / scalasca -examine

```
% square
Scalasca 2.3.1: analysis report explorer
usage: square [-v] [-s] [-f filtfiler] [-F] <experiment archive | cube file>
  -c <none | quick | full> : Level of sanity checks for newly created reports
  -F                        : Force remapping of already existing reports
  -f filtfiler              : Use specified filter file when doing scoring
  -s                        : Skip display and output textual score report
  -v                        : Enable verbose mode
  -n                        : Do not include idle thread metric
```

- Scalasca analysis report explorer (Cube)

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)

BT-MZ summary measurement collection...

```
% cd bin.scorep
% cp ../jobscript/mistral/scalasca.sbatch .
% vim scalasca.sbatch

[...]
```

```
export SCOREP_FILTERING_FILE=scorep.filt
#export SCOREP_TOTAL_MEMORY=222M
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC

# Scalasca configuration
export SCAN_ANALYZE_OPTS="--time-correct"
```

```
scalasca -analyze srun ./bt-mz_${CLASS}.${PROCS}
```

```
% sbatch scalasca.sbatch
```

- Change to directory with the executable and edit the job script

- Submit the job

BT-MZ summary measurement

```
S=C=A=N: Scalasca 2.3.1 runtime summarization
S=C=A=N: ./scorep_bt-mz_B_4x12_sum experiment archive
S=C=A=N: Fri Oct 21 10:37:53 2016: Collect start
srun ./bt-mz_B.4

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

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

[... More application output ...]

S=C=A=N: Fri Oct 21 10:38:08 2016: Collect done (status=0) 15s
S=C=A=N: ./scorep_bt-mz_B_4x12_sum complete.
```

- Run the application using the Scalasca measurement collection & analysis nexus prefixed to launch command
- Creates experiment directory:
./scorep_bt-mz_B_4x12_sum

BT-MZ summary analysis report examination

- Score summary analysis report

```
% square -s scorep_bt-mz_B_4x12_sum  
INFO: Post-processing runtime summarization result...  
INFO: Score report written to ./scorep_bt-mz_B_4x12_sum/scorep.score
```

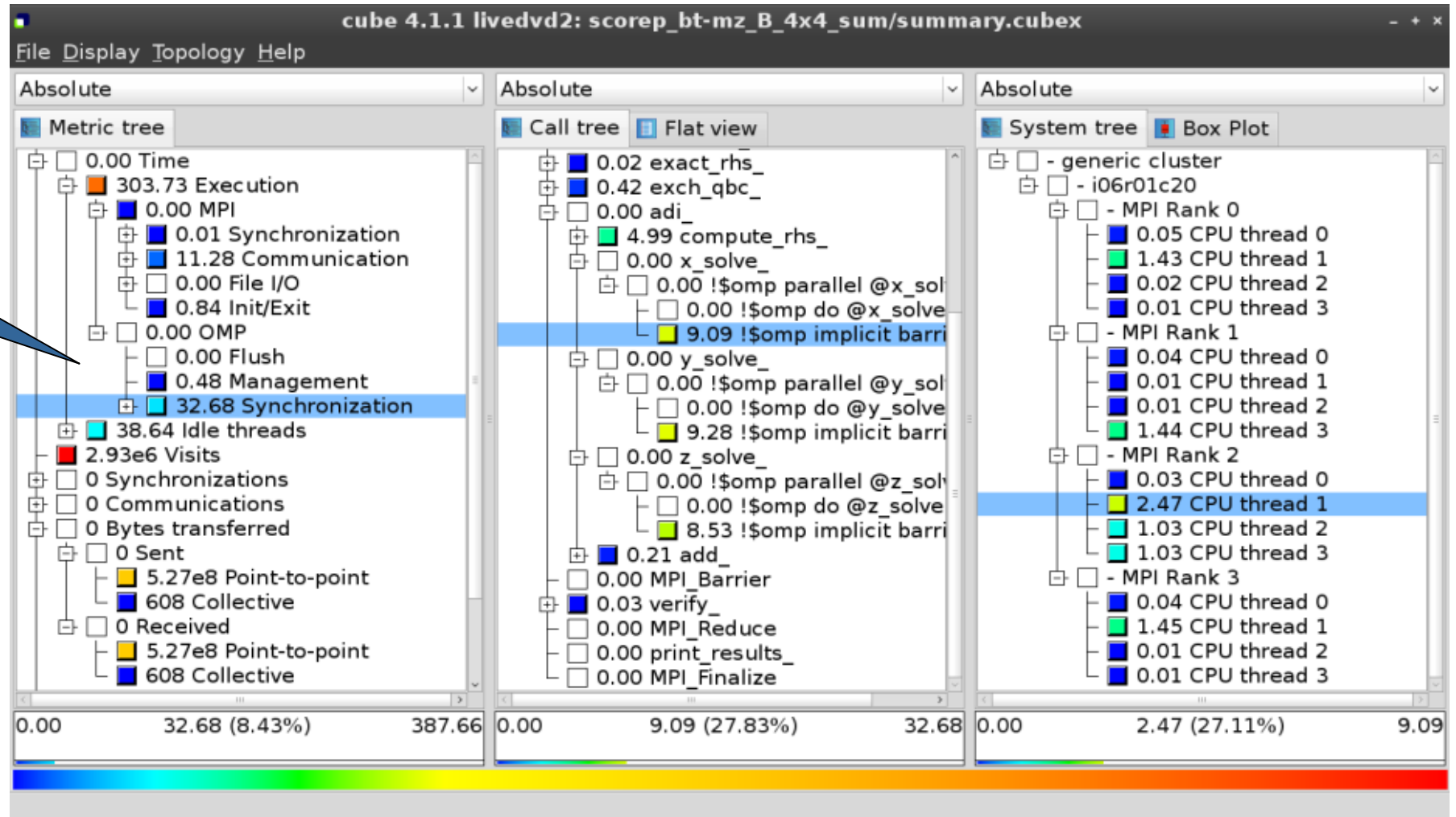
- Post-processing and interactive exploration with Cube

```
% square scorep_bt-mz_B_4x12_sum  
INFO: Displaying ./scorep_bt-mz_B_4x12_sum/summary.cubex...  
  
[GUI showing summary analysis report]
```

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

Post-processed summary analysis report

Split base metrics into more specific metrics



BT-MZ trace measurement collection...

```
% cd bin.scorep
% cp ../jobscript/mistral/scalasca.sbatch .
% vim scalasca.sbatch

[...]

export SCOREP_FILTERING_FILE=scorep.filt
export SCOREP_TOTAL_MEMORY=222M
export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC

# Scalasca configuration
export SCAN_ANALYZE_OPTS="-time-correct"

scalasca -analyze -t srun ./bt-mz_${CLASS}.${PROCS}
```

```
% sbatch scalasca.sbatch
```

- Change to directory with the executable and edit the job script
- Add "-t" to the scalasca -analyze command

- Submit the job

BT-MZ trace measurement ... collection

```
S=C=A=N: Scalasca 2.3.1 trace collection and analysis
S=C=A=N: Fri Oct 21 10:06:15 2016: Collect start
srun ./bt-mz_B.4

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

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

[... More application output ...]

S=C=A=N: Fri Oct 21 10:06:15 2016: Collect done (status=0) 16s
```

- Starts measurement with collection of trace files ...

BT-MZ trace measurement ... analysis

```
S=C=A=N: Fri Oct 21 10:06:16 2016: Analyze start
srun scout.hyb ./scorep_bt-mz_B_4x12_trace/traces.otf2

Analyzing experiment archive
./scorep_bt-mz_B_4x12_trace/traces.otf2

Opening experiment archive ... done (0.065s).
Reading definition data ... done (0.191s).
Reading event trace data ... done (0.839s).
Preprocessing ... done (0.287s).
Timestamp correction ... done (0.686s).
Analyzing trace data ... done (8.884s).
Writing analysis report ... done (0.219s).

Total processing time: 11.444s
S=C=A=N: Fri Oct 21 10:06:28 2016: Analyze done (status=0) 12s
```

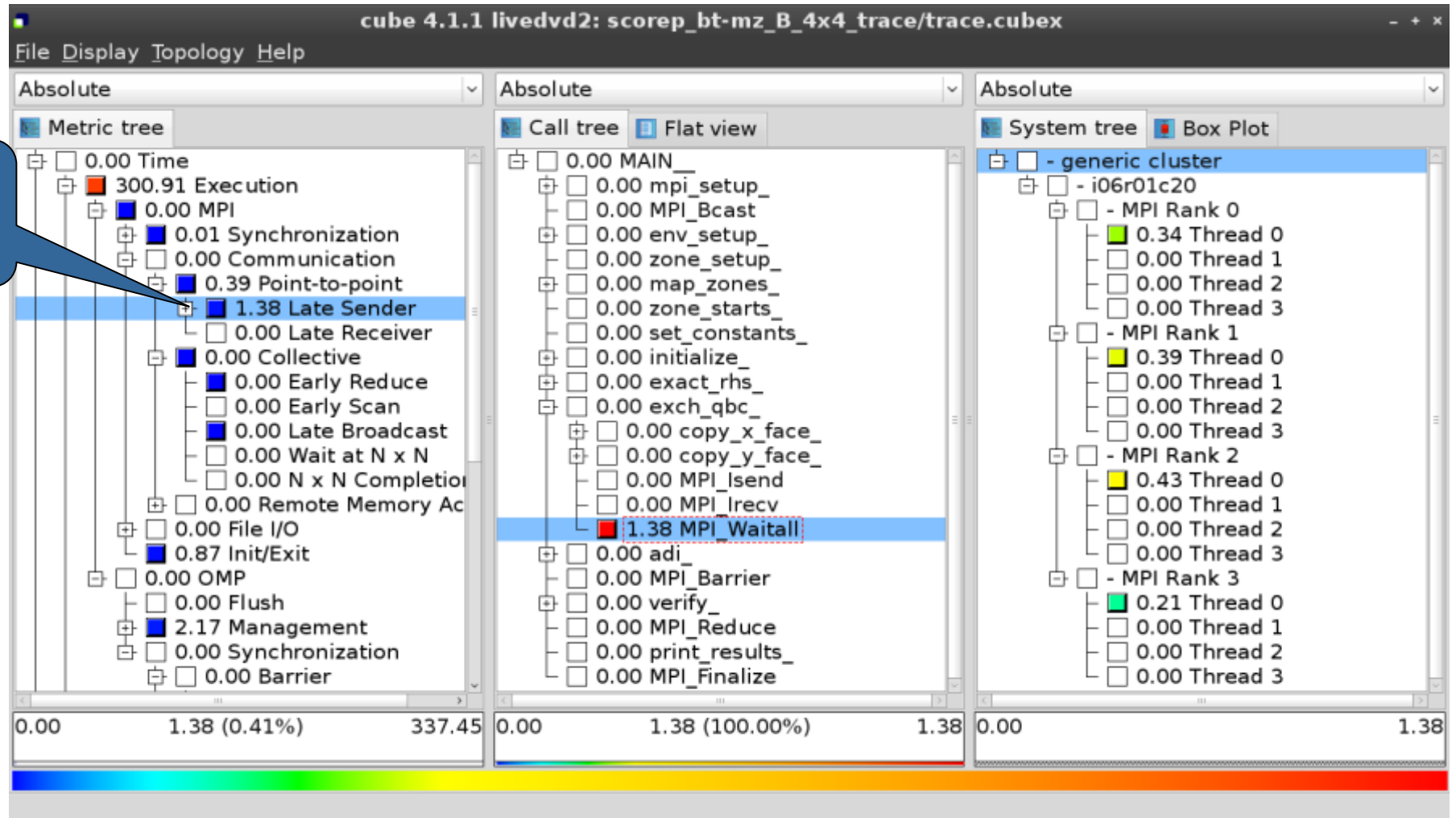
- Continues with automatic (parallel) analysis of trace files

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_4x12_trace  
INFO: Post-processing runtime summarization result...  
INFO: Post-processing trace analysis report...  
INFO: Displaying ./scorep_bt-mz_B_4x12_trace/trace.cubex...  
  
[GUI showing trace analysis report]
```

Post-processed trace analysis report



Online metric description

Access online metric description via context menu

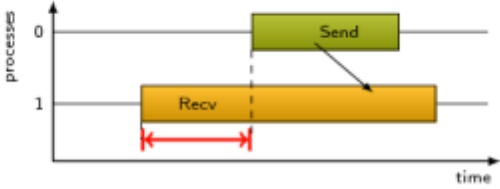
The screenshot displays the 'cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex' application. It features three main panels: 'Metric tree', 'Call tree', and 'System tree'. The 'Metric tree' panel shows a hierarchical view of metrics, with '1.38 Late Send' selected. A context menu is open over this item, listing options such as 'Info', 'Full info', 'Online description', 'Expand/collapse', 'Find items', 'Find Next', 'Clear found items', 'Copy to clipboard', 'Create derived metric...', 'Remove metric...', and 'Statistics'. The 'Online description' option is highlighted. The 'Call tree' panel shows a call stack with 'Waitall' highlighted. The 'System tree' panel shows a system tree with various threads and MPI ranks. A color bar at the bottom indicates the severity of the metrics, ranging from blue (low) to red (high). The status bar at the bottom shows 'Shows the online description of the clicked item'.

Online metric description

Performance properties

Late Sender Time

Description:
Refers to the time lost waiting caused by a blocking receive operation (e.g., `MPI_Recv` or `MPI_Wait`) that is posted earlier than the corresponding send operation.



If the receiving process is waiting for multiple messages to arrive (e.g., in an call to `MPI_Waitall`), the maximum waiting time is accounted, i.e., the waiting time due to the latest sender.

Unit:
Seconds

Diagnosis:
Try to replace `MPI_Recv` with a non-blocking receive `MPI_Irecv` that can be posted earlier, proceed concurrently with computation, and complete with a wait operation after the message is expected to have been sent. Try to post sends earlier, such that they are available when receivers need them. Note that outstanding messages (i.e., sent before the receiver is ready) will occupy internal message buffers, and that large numbers of posted receive buffers will also introduce message management overhead, therefore moderation is advisable.

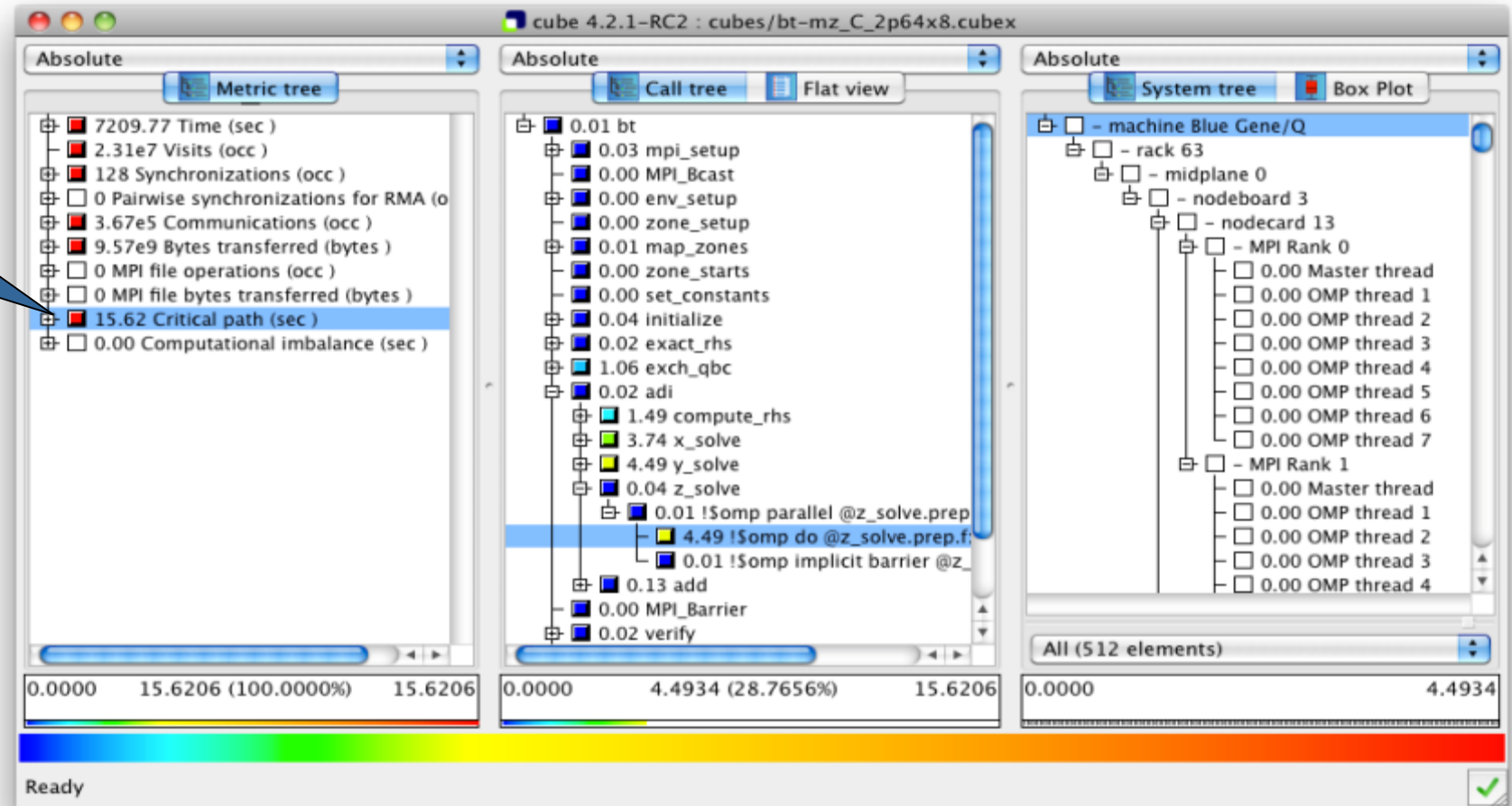
Parent:
[MPI Point-to-point Communication Time](#)

Children:

Close

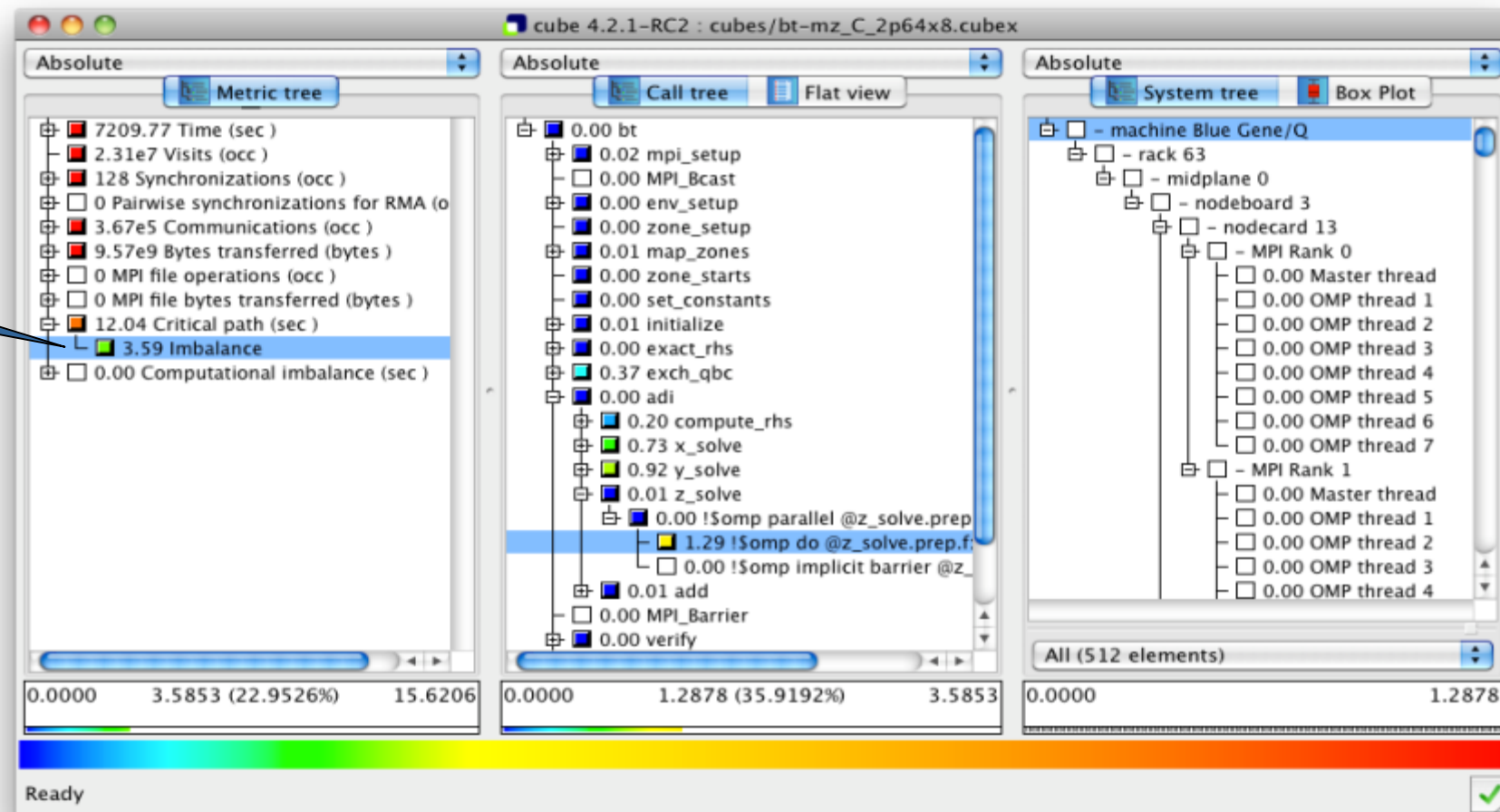
Critical-path analysis

Critical-path profile shows wall-clock time impact



Critical-path analysis

Critical-path imbalance highlights inefficient parallelism



Pattern instance statistics

The screenshot displays the 'cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex' application. The main window is divided into several panes: 'Metric tree' on the left, 'Call tree' and 'Flat view' in the center, and a 'Statistics info' dialog box on the right. A context menu is open over the '1.38 Late Sender' metric in the 'Metric tree'. A blue callout bubble points to the 'Statistics' option in the context menu. Another blue callout bubble points to the 'Statistics info' dialog box. The 'Statistics info' dialog box shows the following data:

Pattern:	mpi_latesender
Sum:	1.38
Count:	832
Mean:	0.00 5%
Standard deviation:	0.00 13%
Maximum:	0.03 100%
Upper quartile (Q3):	0.00 3%
Median:	0.00 3%
Lower quartile (Q1):	0.00 2%
Minimum:	0.00 0%

The 'Statistics info' dialog box also includes a 'To Clipboard' button and a 'Close' button. The 'Metric tree' shows a hierarchical view of metrics, with '1.38 Late Sender' selected. The 'Call tree' shows a hierarchical view of calls, with '0.00 MAIN_' selected. The 'Flat view' shows a flat list of metrics, with '0.00 Thread 0' selected. The 'Statistics info' dialog box also includes a 'Close' button.

Access pattern instance statistics via context menu

Click to get statistics details

Connect to Vampir trace browser

To investigate most severe pattern instances, connect to a trace browser...

The screenshot shows the Vampir trace browser interface. The main window displays a call tree and a system tree. The call tree shows a hierarchy of operations with their durations and percentages. The system tree shows the hardware configuration and MPI ranks. A dialog box titled "Connect to vampir" is open, allowing the user to select a trace file. The dialog has fields for Host (localhost), Port (30000), and File (c:/supermuc_expts/scorep_bt-mz_B_4x4_trace/traces.otf2). The "Open local file" checkbox is checked. The "Connect to vampir" button is highlighted.

File Display Topology Help

Open... Ctrl+O

Save as... Ctrl+S

Close Ctrl+W

Open external...

Close external

Connect to trace browser > Connect to vampir...

Settings > Connect to paraver...

Screenshot...

Quit Ctrl+Q

trace.cubex

summary.cubex

Absolute

Absolute

Call tree Flat view

System tree Box Plot

- generic cluster

- i06r01c20

- MPI Rank 0

0.34 Thread 0

0.00 Thread 1

0.00 Thread 2

0.00 Thread 3

- MPI Rank 1

0.39 Thread 0

0.00 Thread 1

0.00 Thread 2

0.00 Thread 3

0.00 Late Broadcast

0.00 Wait at N x N

0.00 N x N Completion

0.00 Remote Memory Access

0.00 File I/O

0.87 Init/Exit

0.00 OMP

0.00 Flush

2.17 Management

0.00 Synchronization

22.99 Barrier

1.38 (0.41%) 337.45 0.00 1.38 (100.00%) 1.38 0.00 1.38

Connect to vampir and display a trace file

...and select trace file from the experiment directory

Show most severe pattern instances

The screenshot displays the 'cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex' application. It features three main panels: a left sidebar with a hierarchical tree view, a central 'Call tree' panel, and a right 'System tree' panel. A context menu is open over a call path labeled '1.38 MPI_Waitany' in the central panel, which is highlighted with a red border. The menu options include 'Call site', 'Called region', 'Expand/collapse', 'Hiding', 'Cut call tree', 'Find items', 'Find Next', 'Clear found items', 'Copy to clipboard', 'Min/max values', and 'Max severity in trace browser'. A blue callout box on the left contains the text: 'Select "Max severity in trace browser" from context menu of call paths marked with a red frame'. At the bottom of the interface, a color scale bar is visible, and a caption reads: 'Shows the most severe instance of pattern in trace browser'.

Select
"Max severity in trace browser"
from context menu of call paths
marked with a red frame

cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex
File Display Topology Help

0.00 Time
300.91 Execution
0.00 MPI
0.01 Synchronization
0.00 Communication
0.39 Point-to-point
1.38 Late Sender
0.00 Late Receiver
0.00 Collective
0.00 Early Reduce
0.00 Early Scan
0.00 Broadcast
0.00 Wait at N
0.00 N x N Complete
0.00 Remote Memory Ac
0.00 File I/O
0.87 Init/Exit
0.00 OMP
0.00 Flush
2.17 Management
0.00 Synchronization
22.99 Barrier

0.00 1.38 (0.41%) 337.45

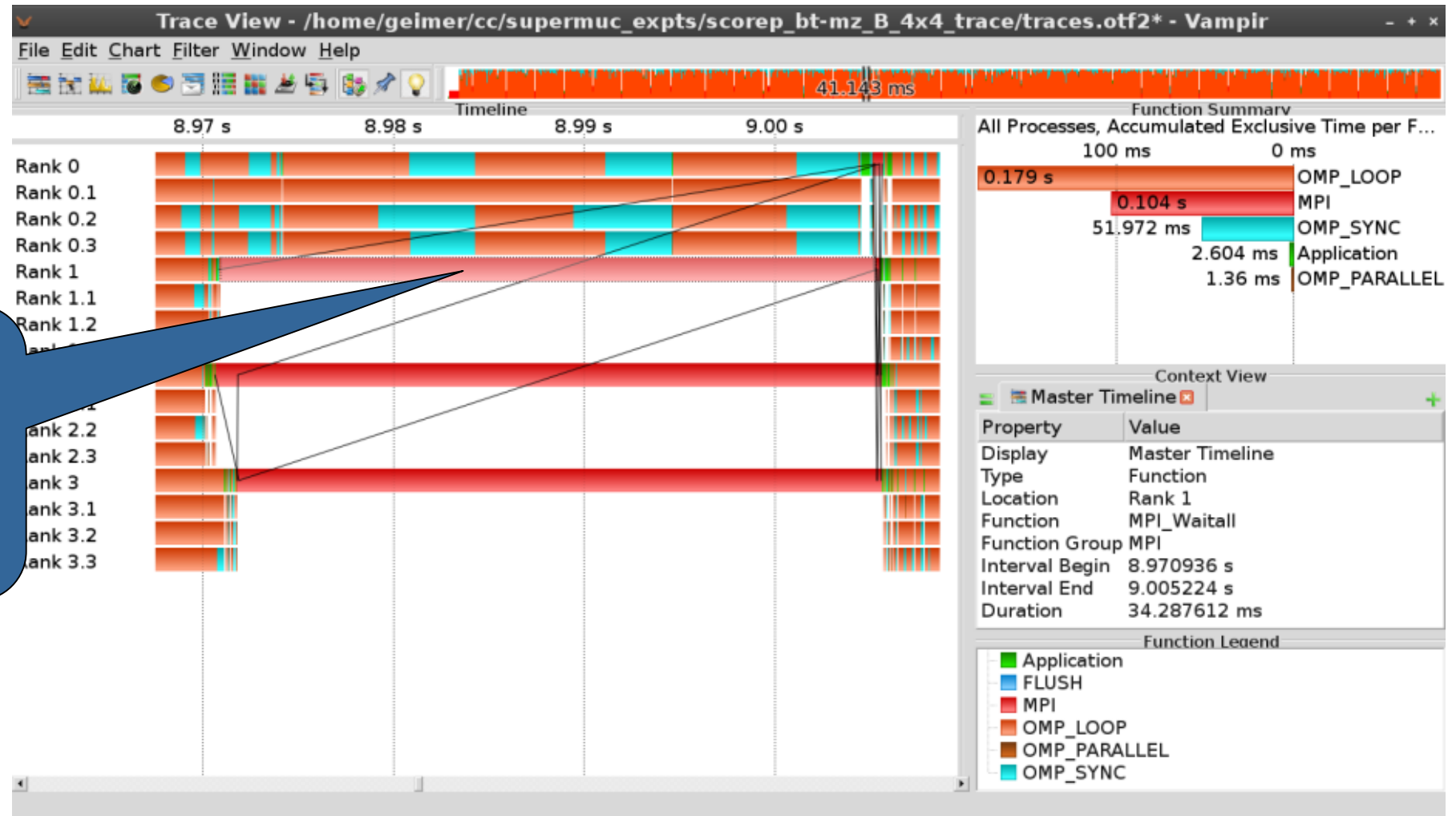
0.00 1.38 (100.00%) 1.38

0.00 1.38

Shows the most severe instance of pattern in trace browser

Investigate most severe instance in Vampir

Vampir will automatically zoom to the worst instance in multiple steps (i.e., undo zoom provides more context)



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:
 - <http://www.scalasca.org>
- Contact:
 - [mailto: scalasca@fz-juelich.de](mailto:scalasca@fz-juelich.de)

