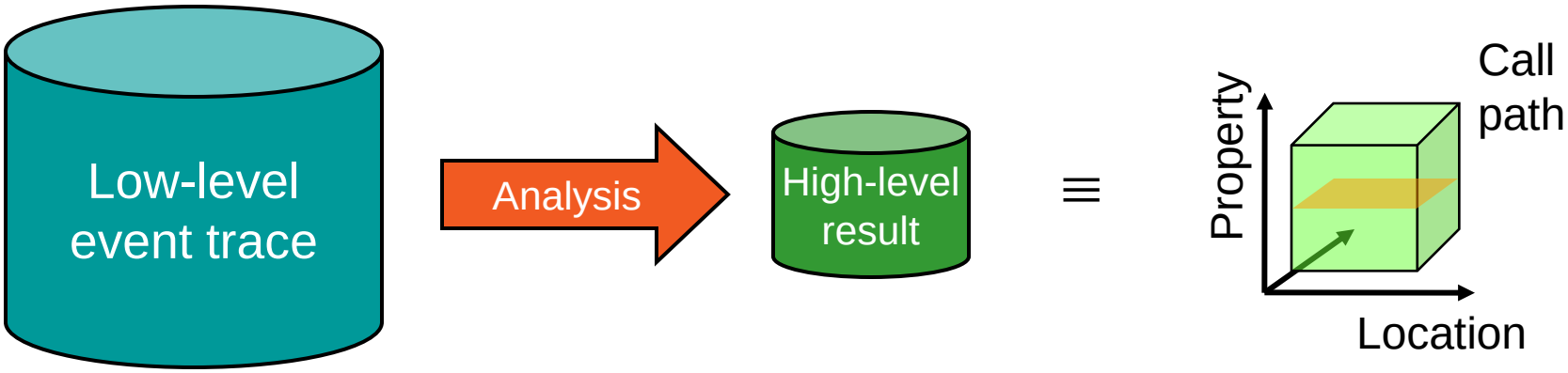
Scalasca

Brian Wylie
Jülich Supercomputing Centre

scalasca

- Idea
  - Automatic search for patterns of inefficient behaviour
  - Classification of behaviour & quantification of significance



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

The Scalasca project

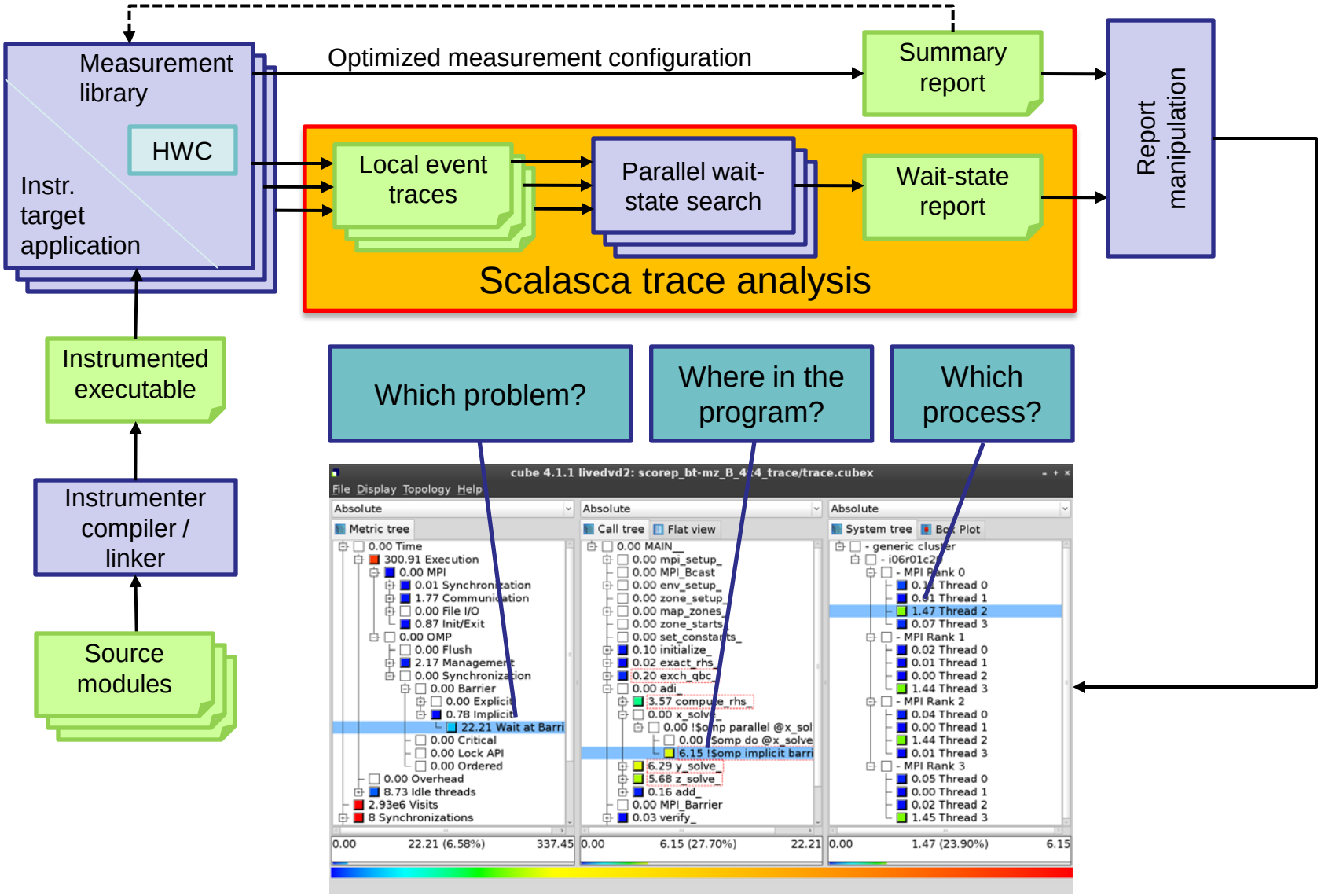
- Project started in 2006
  - Follow-up to pioneering KOJAK project (started 1998)
- Joint development of
  - Jülich Supercomputing Centre
  - German Research School for Simulation Sciences

- Development of a **scalable** performance analysis toolset for most popular parallel programming paradigms
- Specifically targeting **large-scale** parallel applications
  - such as those running on IBM BlueGene or Cray systems with one million or more processes/threads
- Latest release:
  - Scalasca v2.1 (August 2014)

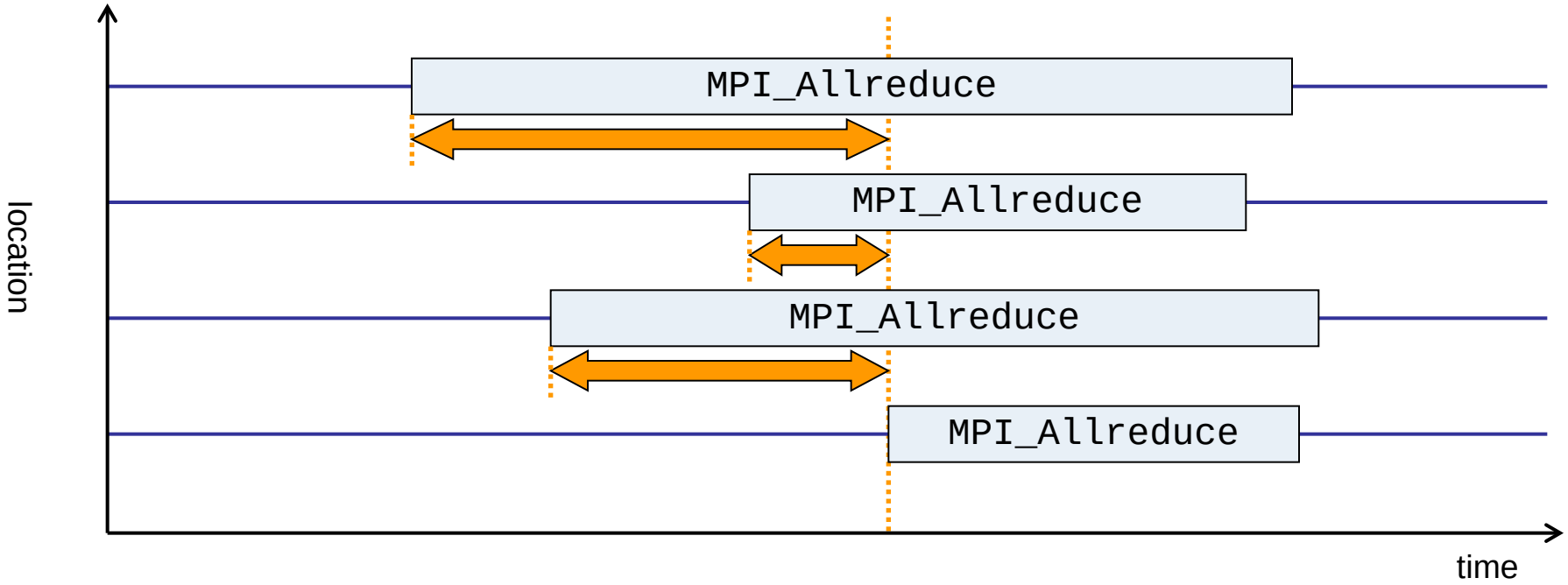
Scalasca 2.1 features

- Open source, BSD 3-clause license
- Fairly portable
  - IBM Blue Gene, IBM SP & blade clusters, Cray XT/XE/XK/XC, SGI Altix, Solaris & Linux clusters, Fujitsu FX10 & K computer, ...
- Uses Score-P instrumenter & measurement libraries
  - Scalasca 2.1 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:
  - No support for nested OpenMP parallelism and tasking
  - Unable to handle OTF2 traces containing CUDA events
  - PAPI & rusage metrics for trace events are ignored

Scalasca workflow

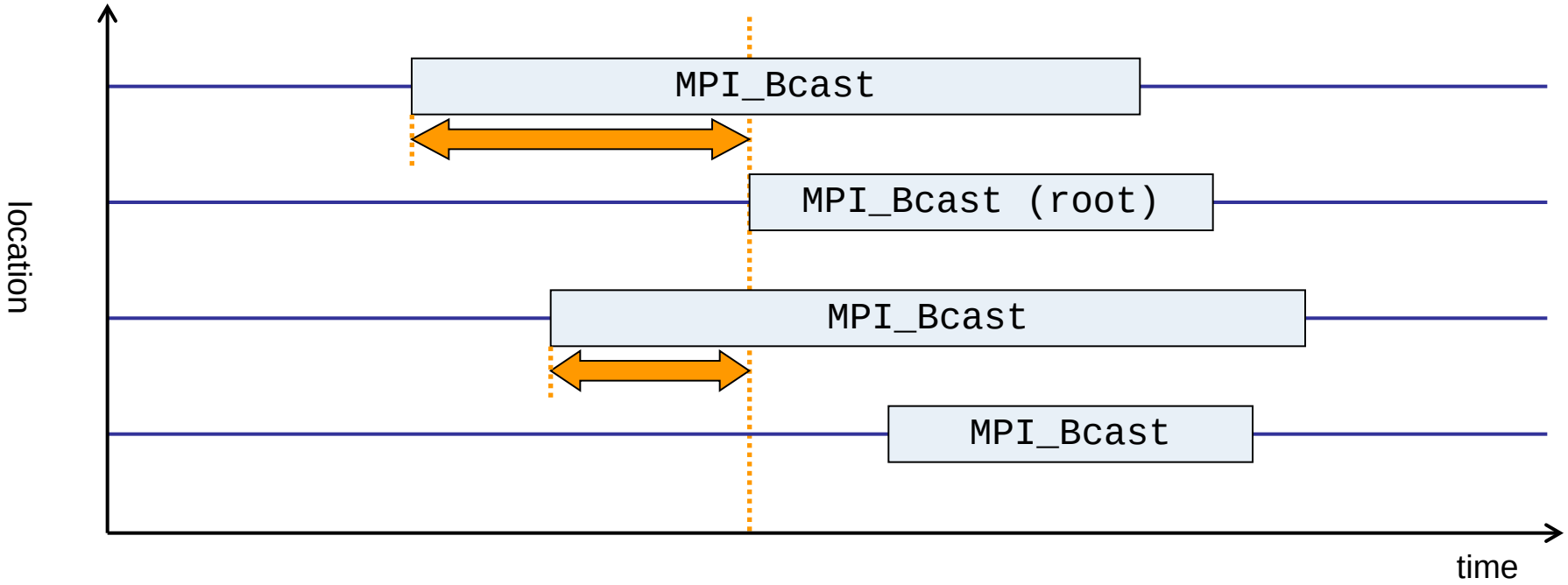


Example: Wait at NxN



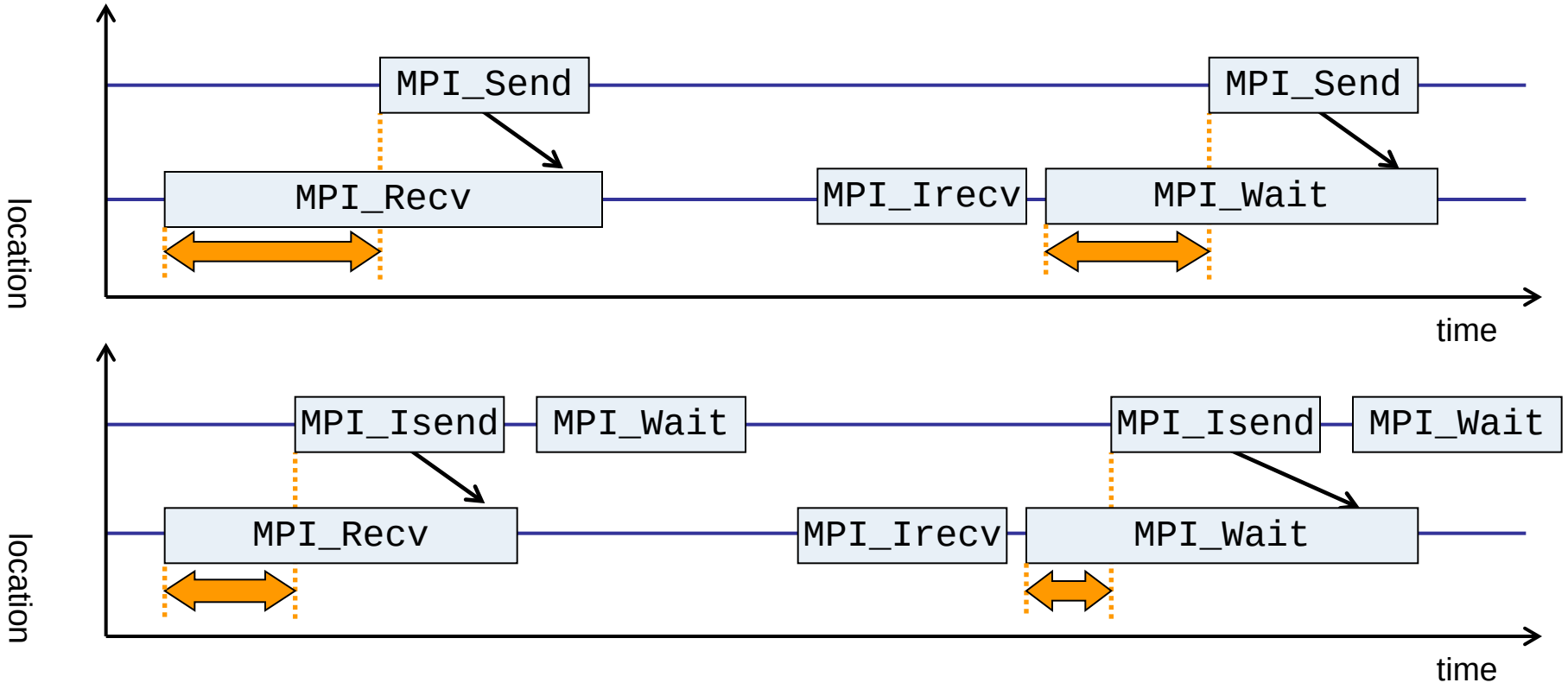
- Time spent waiting in front of synchronizing collective operation until the last process reaches the operation
- Applies to: MPI_Allgather, MPI_Allgatherv, MPI_Alltoall, MPI_Reduce_scatter, MPI_Reduce_scatter_block, MPI_Allreduce

Example: Late Broadcast



- Waiting times if the destination processes of a collective 1-to-N operation enter the operation earlier than the source process (root)
- Applies to: MPI_Bcast, MPI_Scatter, MPI_Scatterv

Example: Late Sender



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

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

scalasca

Scalasca command

- One command for (almost) everything…

```
% scalasca
Scalasca 2.1
Toolset for scalable performance analysis of large-scale 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

   -c, --show-config  show configuration and exit
   -h, --help         show this help and exit
   -n, --dry-run      show actions without taking them
       --quickref     show quick reference guide and exit
   -v, --verbose      enable verbose commentary
   -V, --version      show version information and exit
```

Scalasca compatibility command: *skin*

- Scalasca application instrumenter

```
% skin
Scalasca 2.1: application instrumenter using scorep
usage: skin [-v] [–comp] [-pdt] [-pomp] [-user] <compile-or-link-cmd>
   -comp={all|none|...}: routines to be instrumented by compiler
          (... custom instrumentation specification for compiler)
   -pdt:  process source files with PDT 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
```

  - Deprecated command
    - Provides compatibility with Scalasca 1.x
    - Prints corresponding Score-P instrumenter command
    - Helps in transitioning existing configurations
  - Recommended: use Score-P instrumenter directly

Scalasca convenience command: *scan*

- Scalasca measurement collection & analysis nexus

```
% scan
Scalasca 2.1: 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.
```

Scalasca advanced command: *scout*

- Scalasca automatic trace analyzer

```
% mpiexec –np 1 scout.hyb --help
SCOUT   Copyright (c) 1998-2014 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
  --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 analysis report explorer

```
% square
Scalasca 2.1: analysis report explorer
usage: square [-v] [-s] [-f filtfile] [-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 filtfile         : Use specified filter file when doing scoring
   -s                  : Skip display and output textual score report
   -v                  : Enable verbose mode
```

Automatic measurement configuration

- **scan** configures Score-P measurement by setting some environment variables automatically
  - 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...

- Change to directory with executable and edit job script

```
% cd bin.scorep
% cp ../jobscript/dlr/scan.pbs .
% vi scan.pbs

 [...]

module load scalasca

# Scalasca2/Score-P configuration
#export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_TOTAL_MEMORY=31M

NEXUS="scalasca -analyze -f ../config/scorep.filt"
$NEXUS  mpiexec -np $NPROCS  $EXE
```

- Submit the job

```
% qsub scan.pbs
```


BT-MZ summary measurement

- Run the application using the Scalasca measurement collection & analysis nexus prefixed to launch command

```
% export SCOREP_FILTERING_FILE=../config/scorep.filt
% OMP_NUM_THREADS=4 scan mpiexec –np 4 ./bt-mz_W.4
S=C=A=N: Scalasca 2.1 runtime summarization
S=C=A=N: ./scorep_bt-mz_W_4x4_sum experiment archive
S=C=A=N: Thu Jun 12 18:05:17 2014: Collect start
mpiexec –np 4 ./bt-mz_W.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: Thu Jun 12 18:05:39 2014: Collect done (status=0) 22s
S=C=A=N: ./scorep_bt-mz_W_4x4_sum complete.
```

- Creates experiment directory ./scorep_bt-mz_W_4x4_sum

BT-MZ summary analysis report examination

- Score summary analysis report

```
% square -s  scorep_bt-mz_W_4x4_sum
INFO: Post-processing runtime summarization result...
INFO: Score report written to ./scorep_bt-mz_W_4x4_sum/scorep.score
```

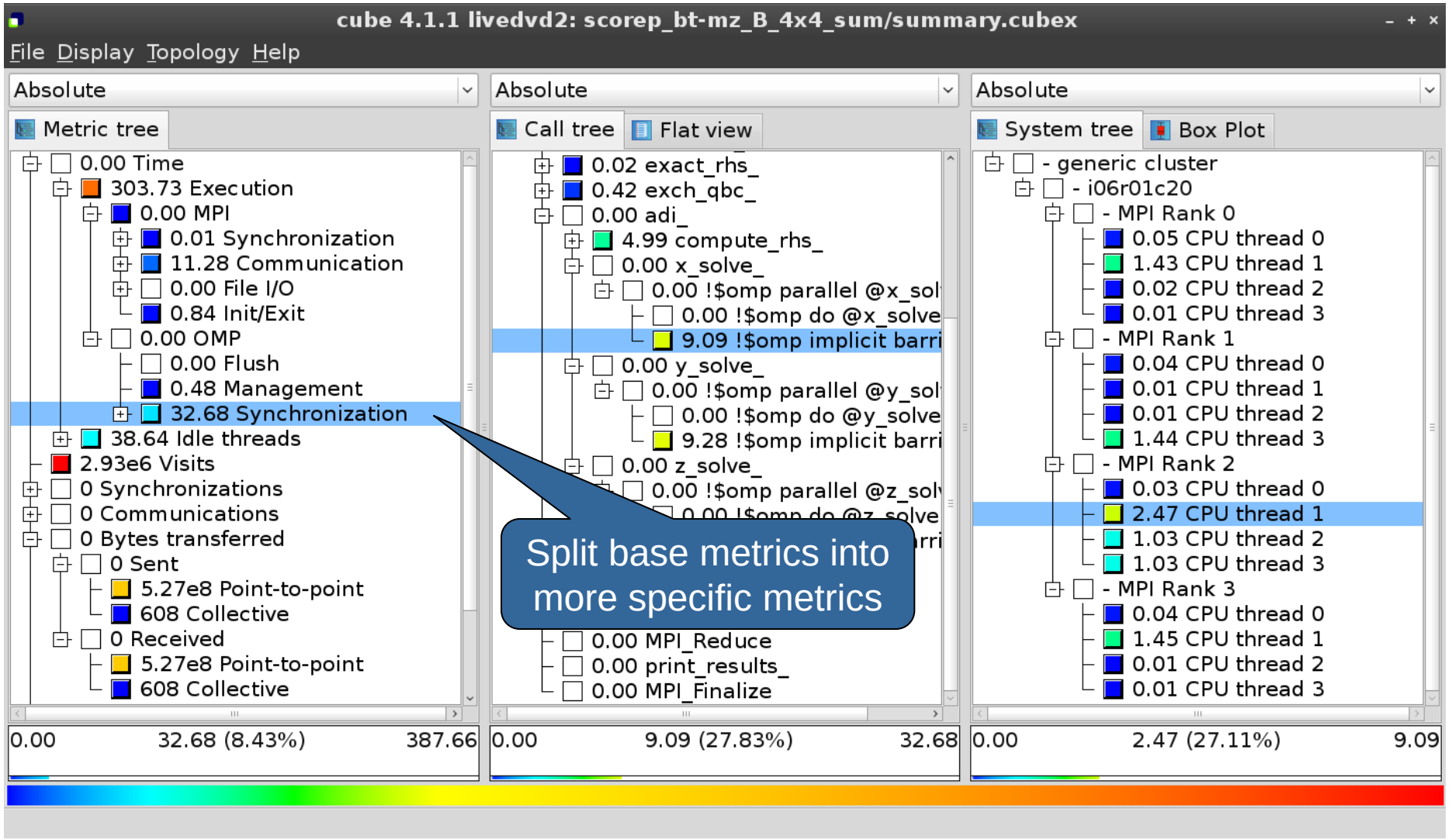
- Post-processing and interactive exploration with CUBE

```
% square  scorep_bt-mz_W_4x4_sum
INFO: Displaying ./scorep_bt-mz_W_4x4_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



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

BT-MZ trace measurement collection...

- Change to directory with executable and edit job script

```
% cd bin.scorep
% cp ../jobscript/dlr/scan.pbs .
% vi scan.pbs

 [...]

module load scalasca/2.1

# Scalasca2/Score-P configuration
export SCOREP_FILTERING_FILE=../config/scorep.filt
export SCOREP_TOTAL_MEMORY=31M

NEXUS="scalasca -analyze -t"
$NEXUS  mpiexec -np $NPROCS  $EXE
```

- Submit the job

```
% qsub scan.pbs
```


BT-MZ trace measurement collection...

- Re-run the application using Scalasca nexus with **“-t”** flag

```
% export SCOREP_FILTERING_FILE=../config/scorep.filt
% OMP_NUM_THREADS=4 scan -t mpiexec –np 4 ./bt-mz_W.4
S=C=A=N: Scalasca 2.1 trace collection and analysis
S=C=A=N: ./scorep_bt-mz_W_4x4_trace experiment archive
S=C=A=N: Thu Jun 12 18:05:39 2014: Collect start
mpiexec –np 4 ./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: Thu Jun 12 18:05:58 2014: Collect done (status=0) 19s
 [... continued ...]
```


BT-MZ trace measurement ... analysis

- Continues with automatic (parallel) analysis of trace files

```
S=C=A=N: Thu Jun 12 18:05:58 2014: Analyze start
mpiexec –np 4 scout.hyb ./scorep_bt-mz_W_4x4_trace/traces.otf2
SCOUT   Copyright (c) 1998-2012 Forschungszentrum Juelich GmbH
        Copyright (c) 2009-2012 German Research School for Simulation
                                Sciences GmbH

Analyzing experiment archive ./scorep_bt-mz_W_4x4_trace/traces.otf2

Opening experiment archive ... done (0.002s).
Reading definition data    ... done (0.004s).
Reading event trace data   ... done (0.130s).
Preprocessing              ... done (0.259s).
Analyzing trace data       ...
  Wait-state detection (fwd)      (1/4) ... done (0.575s).
  Wait-state detection (bwd)      (2/4) ... done (0.138s).
  Synchpoint exchange             (3/4) ... done (0.358s).
  Critical-path analysis          (4/4) ... done (0.288s).
done (1.360s).
Writing analysis report    ... done (0.121s).

Total processing time     : 1.924s
S=C=A=N: Thu Jun 12 18:06:00 2014: Analyze done (status=0) 2s
```

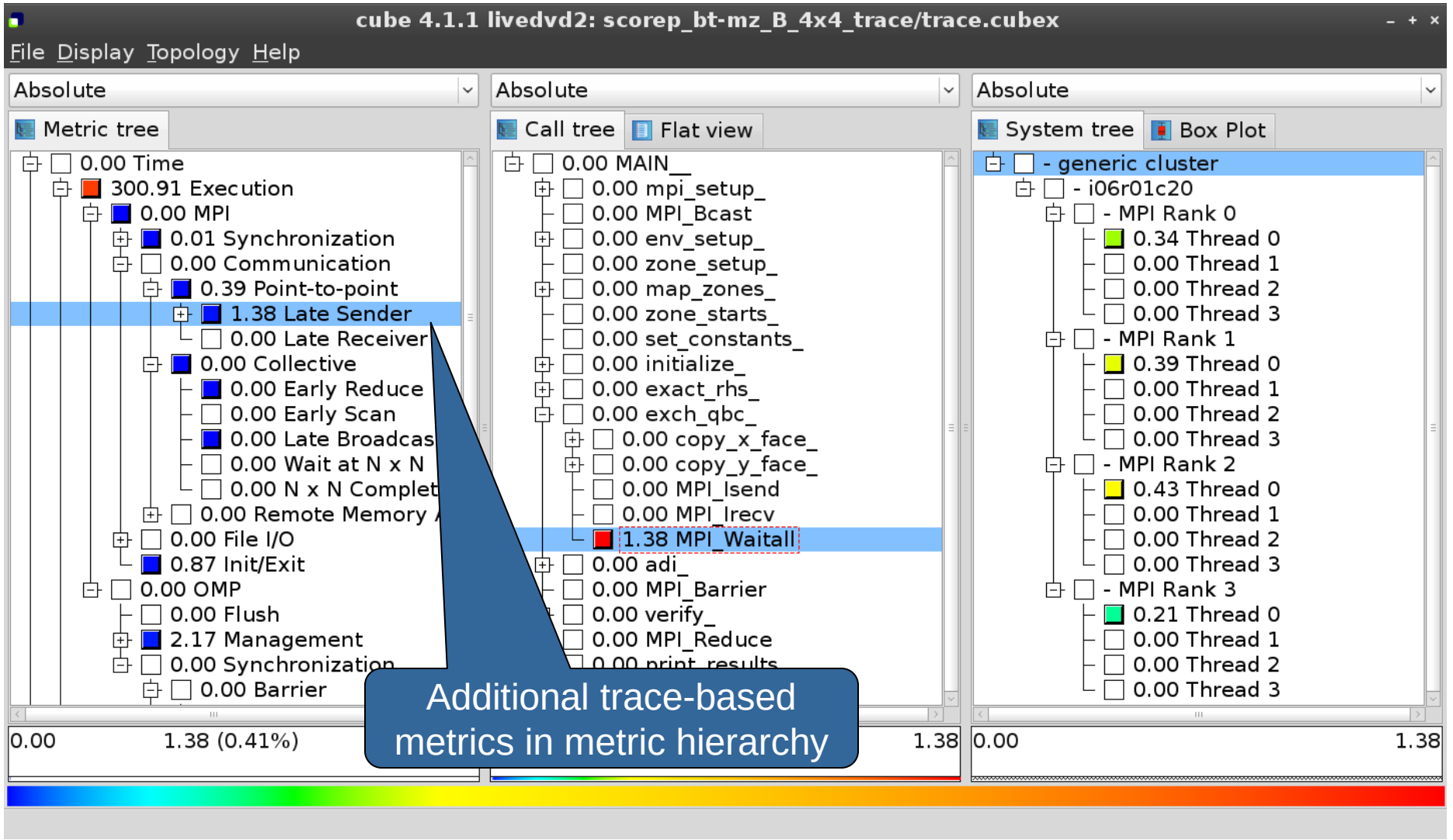

BT-MZ trace analysis report exploration

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

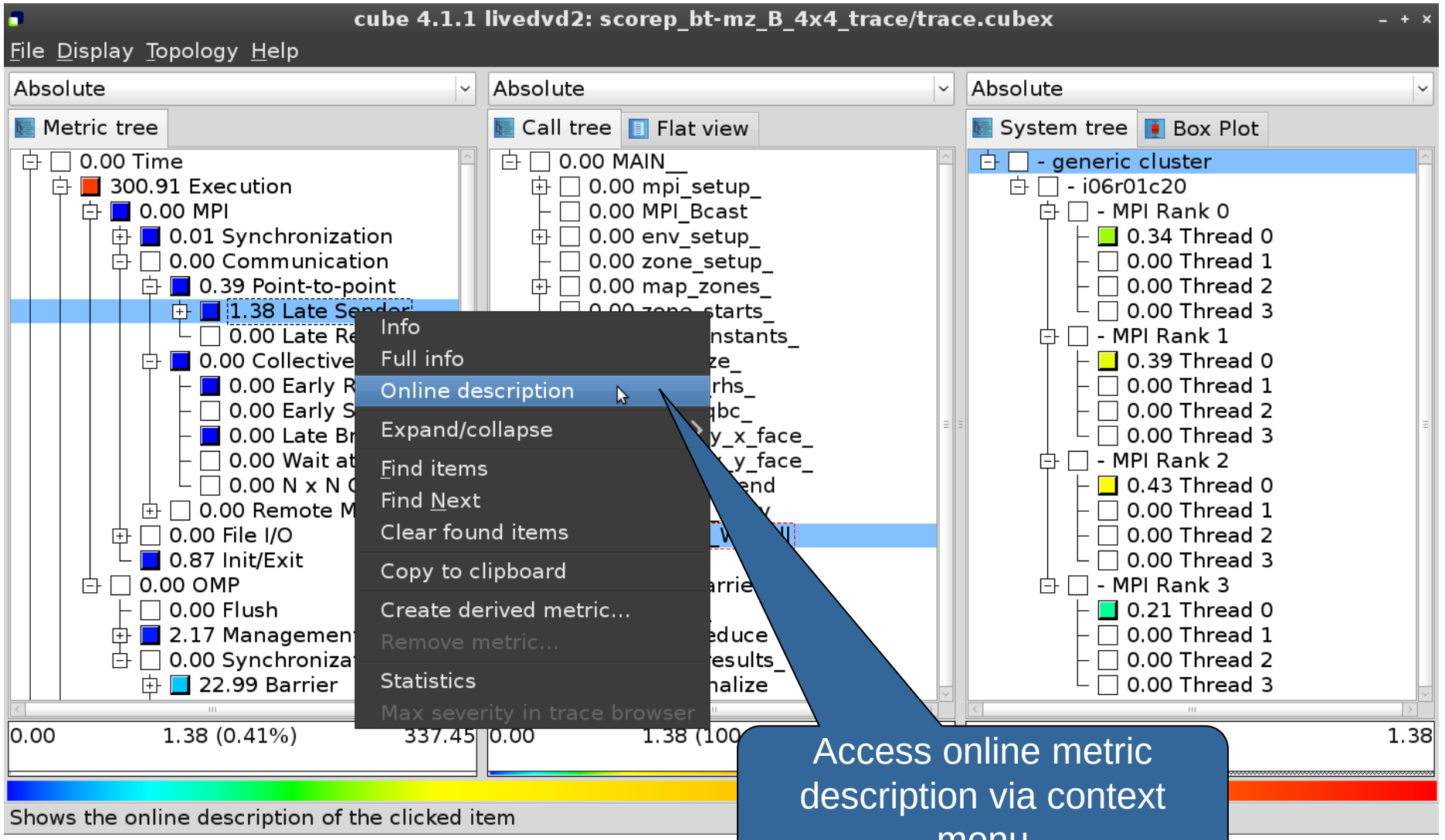
```
% square  scorep_bt-mz_W_4x4_trace
INFO: Post-processing runtime summarization result...
INFO: Post-processing trace analysis report...
INFO: Displaying ./scorep_bt-mz_W_4x4_trace/trace.cubex...

                [GUI showing trace analysis report]
```


Post-processed trace analysis report



Online metric description



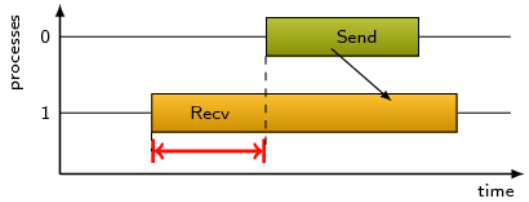
Online metric description

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.

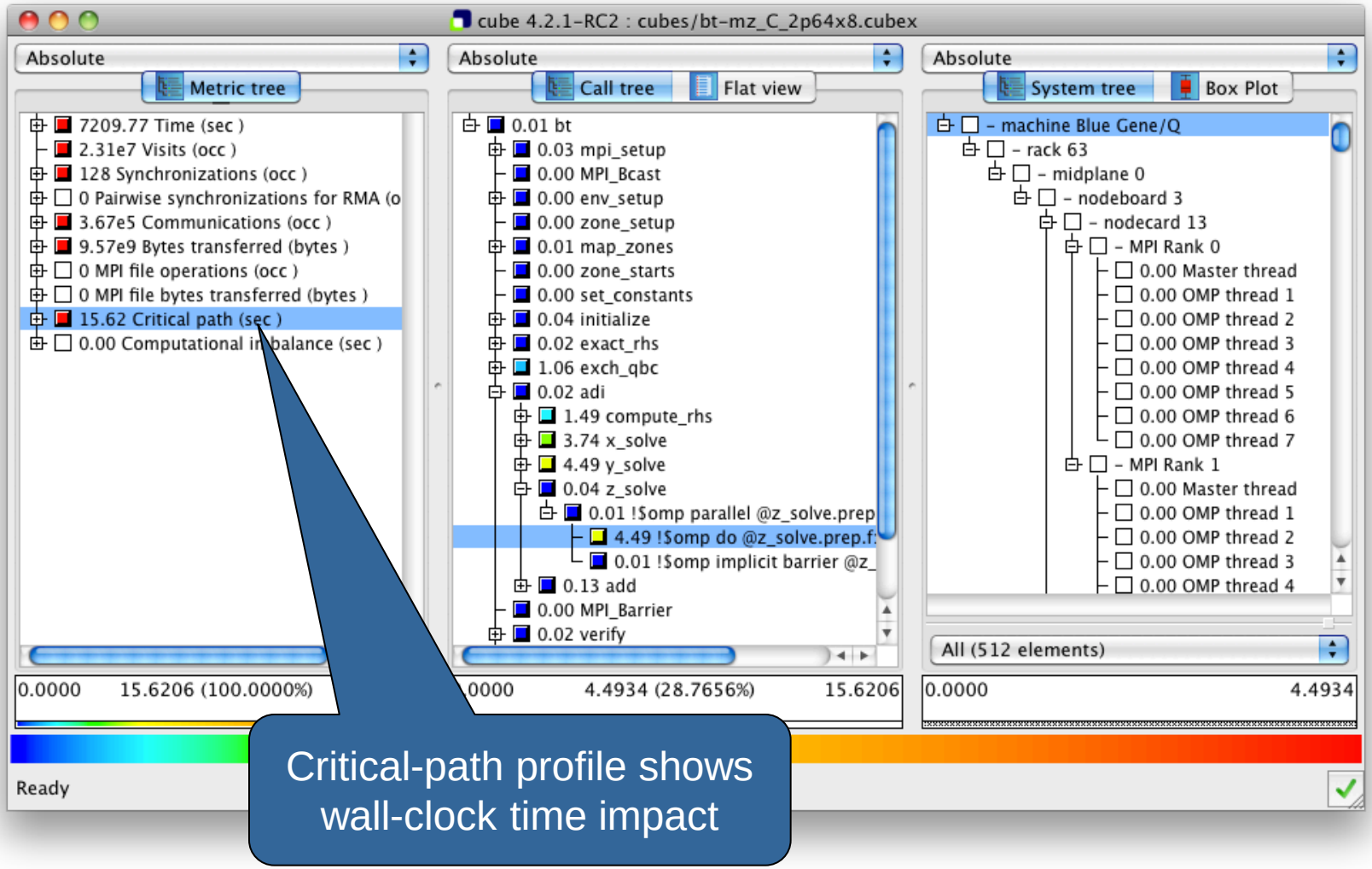
Parent:

MPI Point-to-point Communication Time

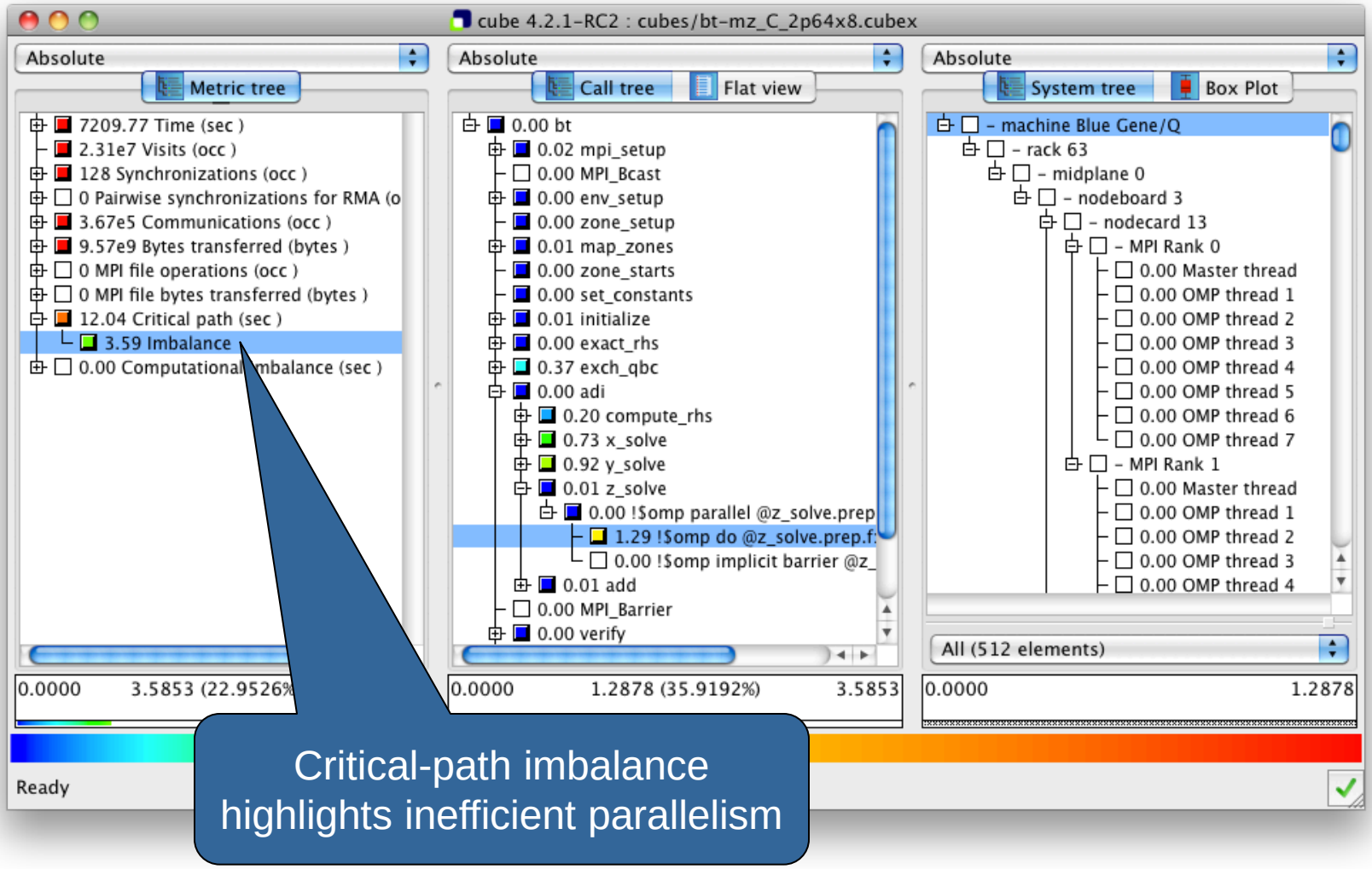
Children:

Close

Critical-path analysis

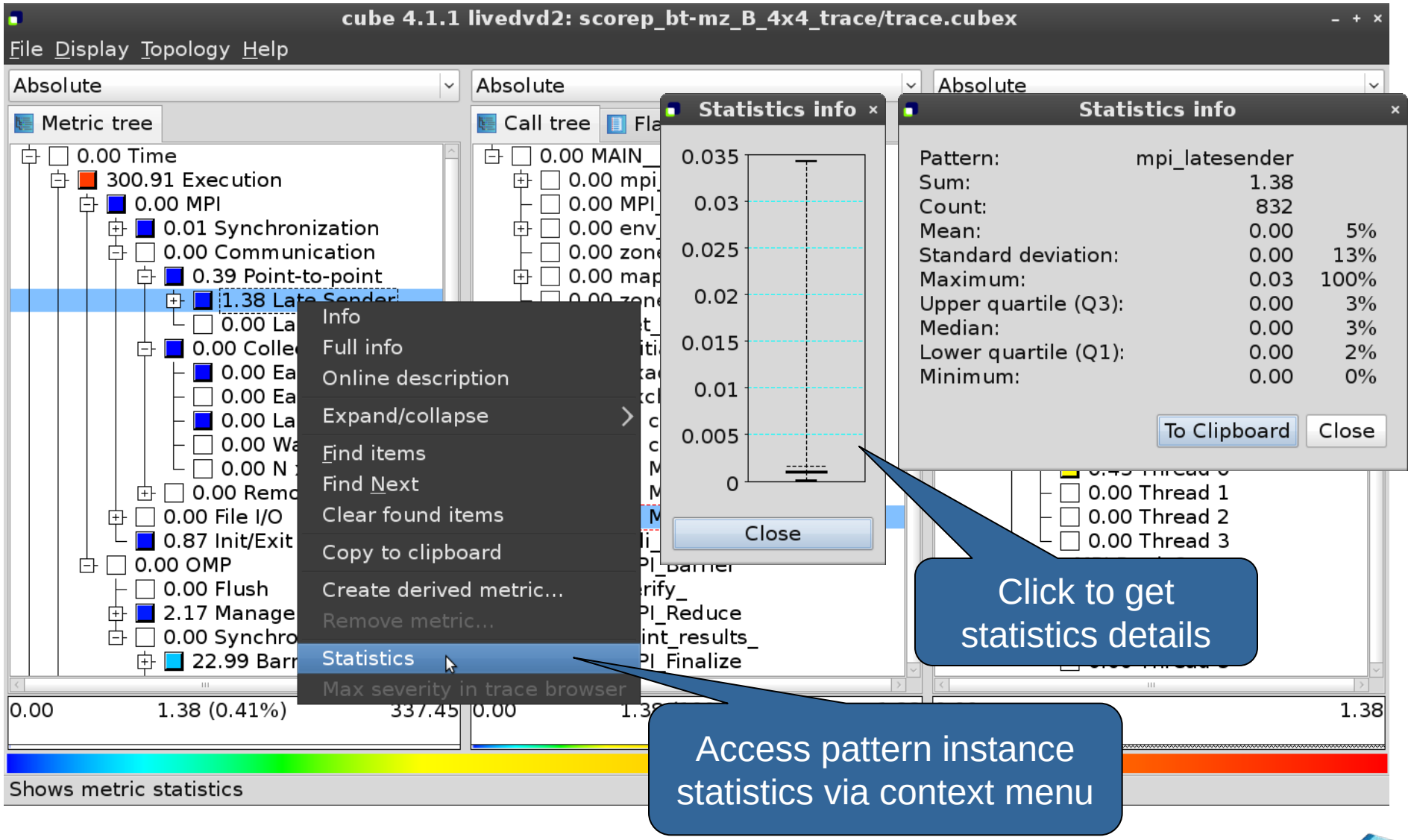


Critical-path analysis

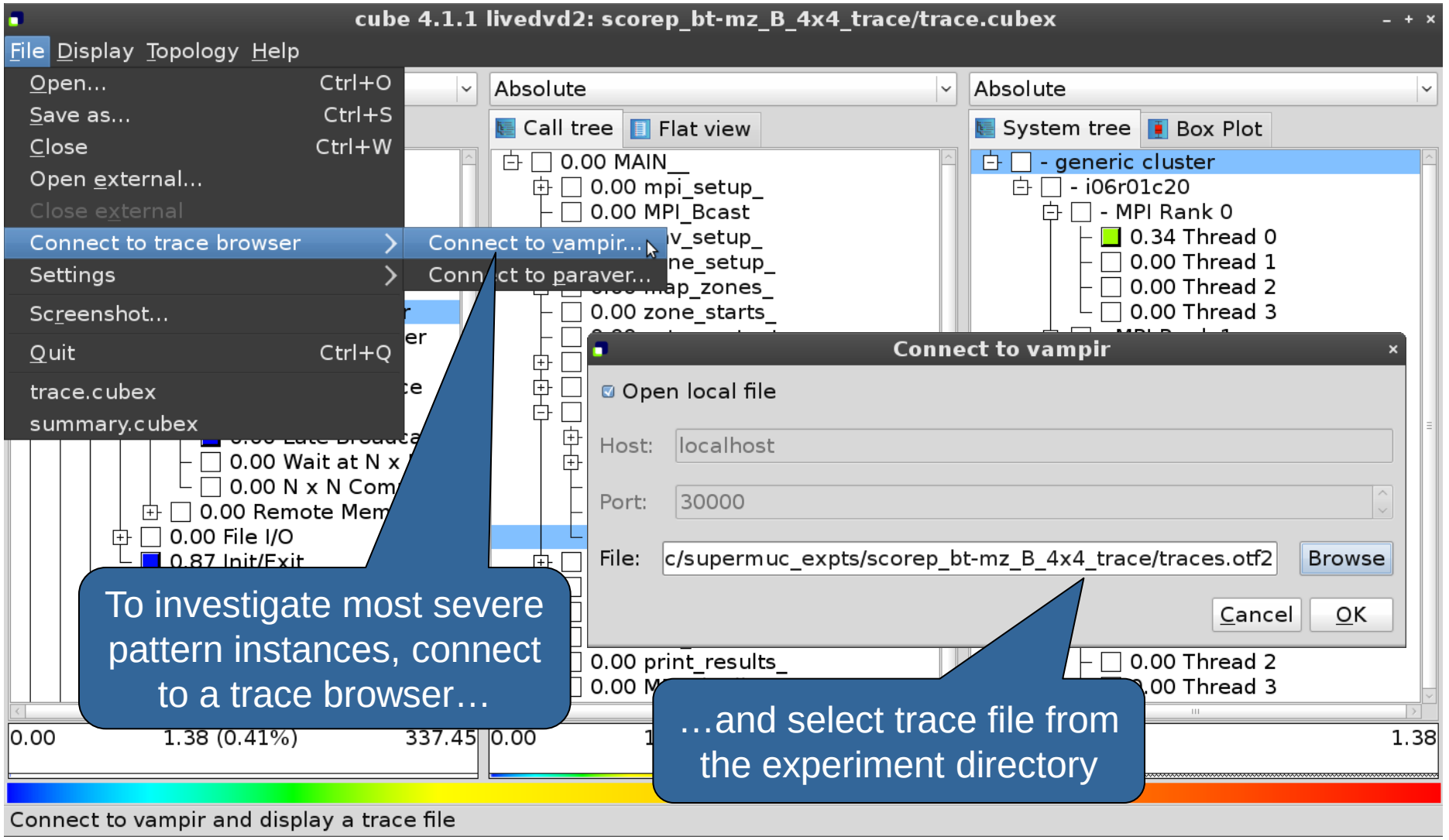


Critical-path imbalance highlights inefficient parallelism

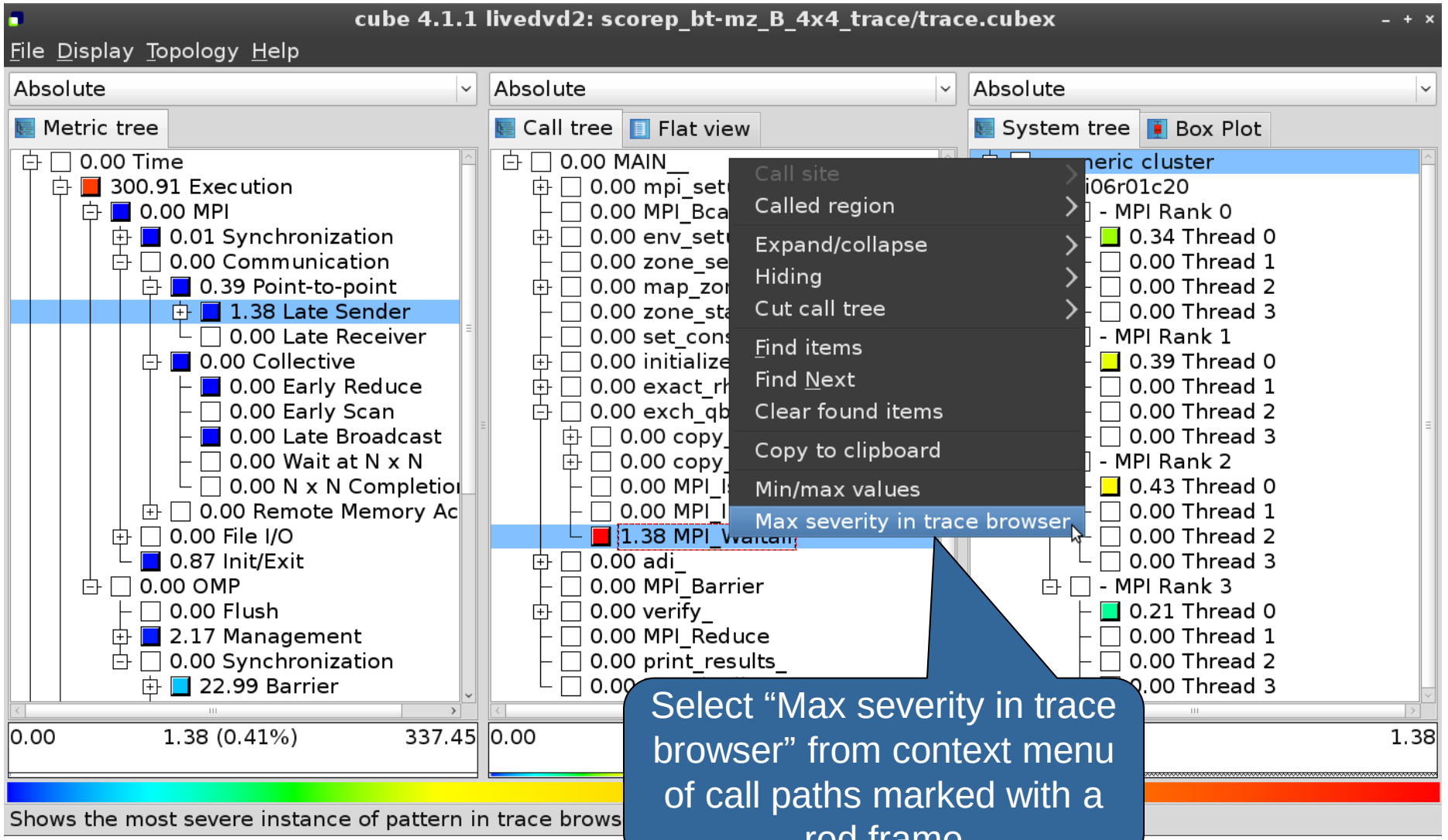
Pattern instance statistics



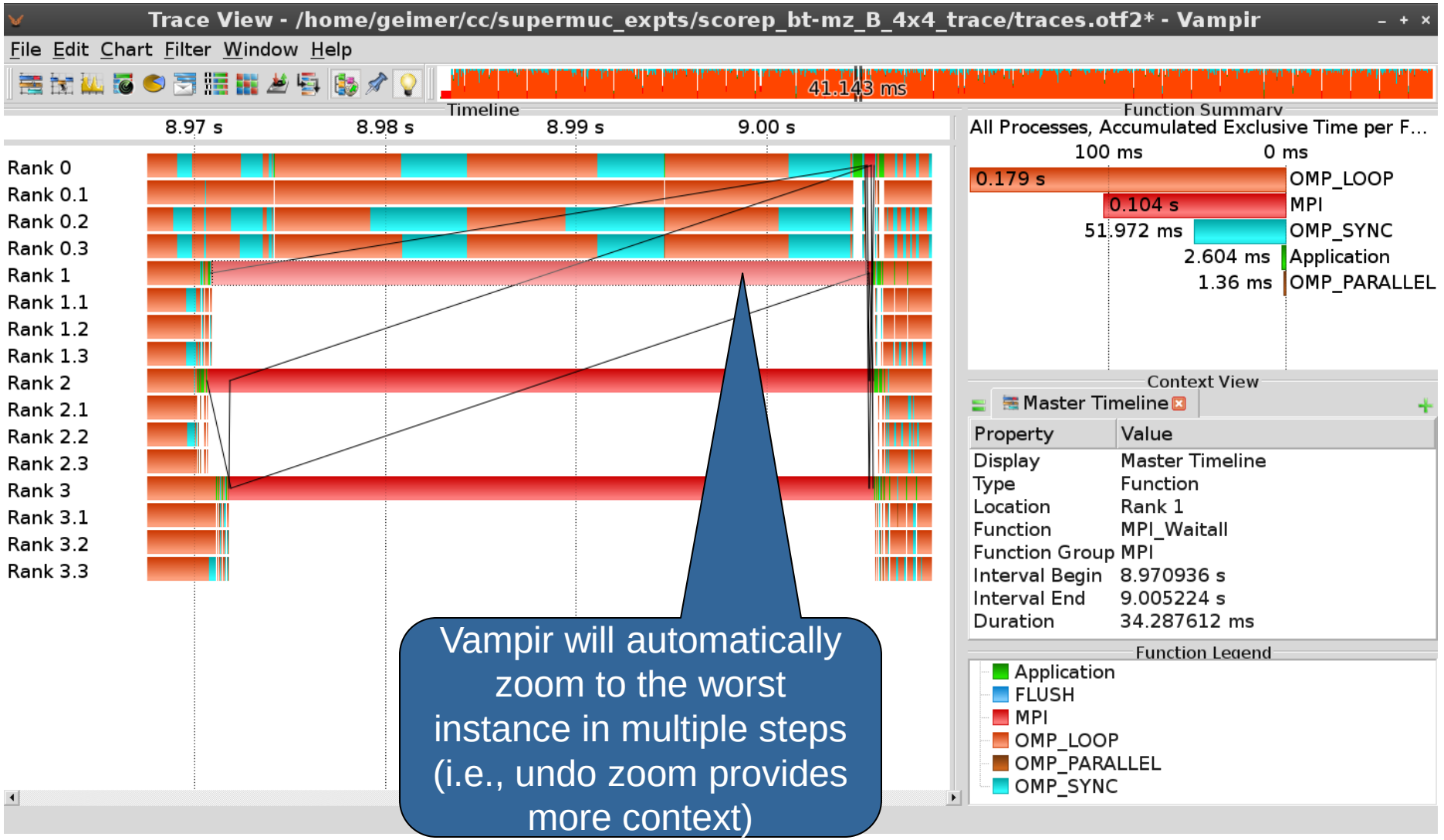
Connect to Vampir trace browser



Show most severe pattern instances



Investigate most severe instance in Vampir



Scalable performance **a**nalysis of **la**rge-**sc**ale parallel **a**pplications

- toolset for scalable performance measurement & analysis of MPI, OpenMP & hybrid parallel applications
- supporting most popular HPC computer systems
- available under New BSD open-source license
- sources, documentation & publications:
  - http://www.scalasca.org
  - mailto: scalasca@fz-juelich.de