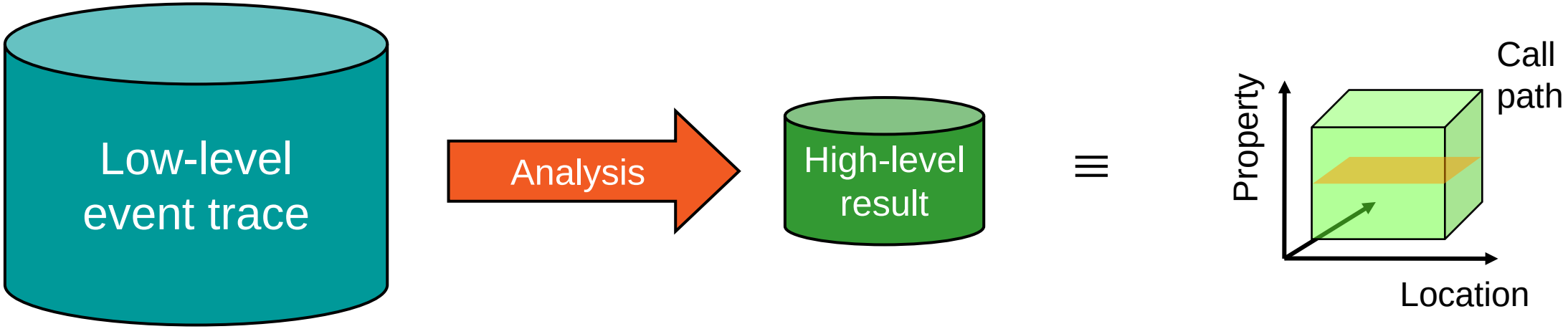
Automatic trace analysis with Scalasca

Dr. Alexandre Otto Strube
Jülich Supercomputing Centre

scalasca

Automatic trace analysis

- Idea
  - Automatic search for patterns of inefficient behaviour
  - Classification of behavior & 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: Overview

- Project started in 2006
  - Initial funding by Helmholtz Initiative & Networking Fund
  - Many follow-up projects
- Follow-up to pioneering KOJAK project (started 1998)
  - Automatic pattern-based trace analysis
- Now joint development of
  - Jülich Supercomputing Centre
  - German Research School for Simulation Sciences
  - Technische Universität Darmstadt - Laboratory for Parallel Programming

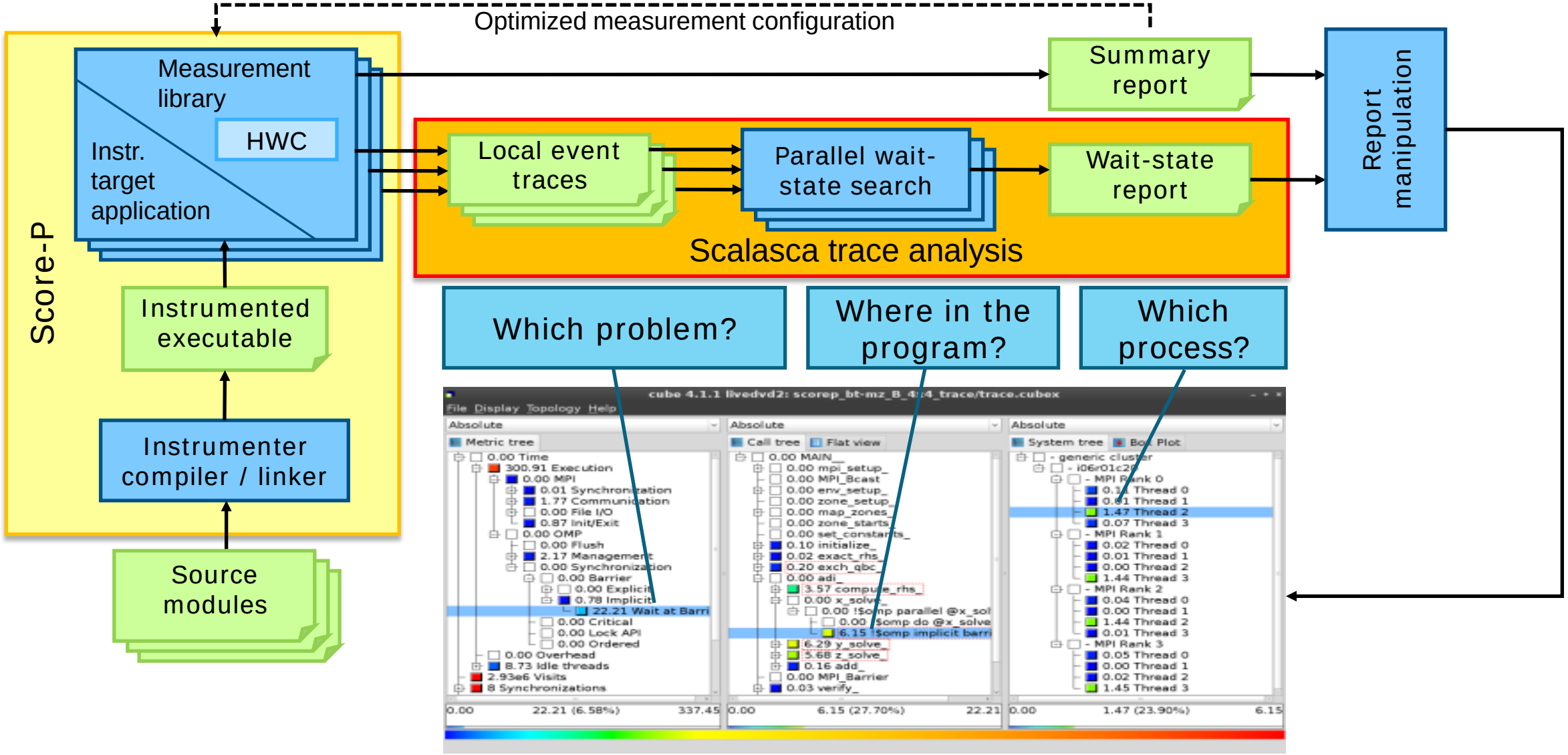
Scalasca 2.2 features

- Open source, New BSD license
- Fairly portable
  - IBM Blue Gene, Cray XT/XE/XK/XC, SGI Altix, Fujitsu FX10/100 & K computer, Linux clusters, Intel Xeon Phi (native MIC) ...
- Uses Score-P instrumenter & measurement libraries
  - Scalasca 2 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 containing CUDA or SHMEM events, or OpenMP nested parallelism
  - PAPI/rusage metrics for trace events are ignored

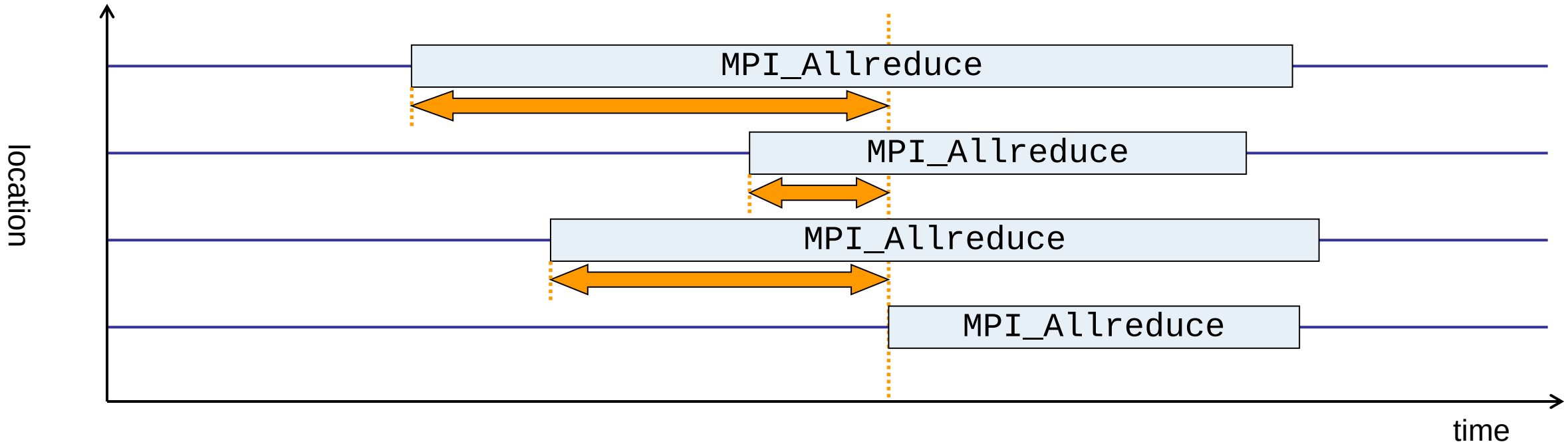
The Scalasca project: Objective

- 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 v2.2 coordinated with Score-P v1.4 (January 2015)
    - initial support for Intel Xeon Phi (native mode only)
    - full support for traces in SIONlib format (if configured for OTF2)
    - basic support for POSIX threads and OpenMP tasking
    - added lock contention and root-cause/delay analysis
  - Scalasca v2.2.1 coordinated with Score-P 1.4.1 (May 2015)
    - bug-fixes and optimisations

Scalasca workflow

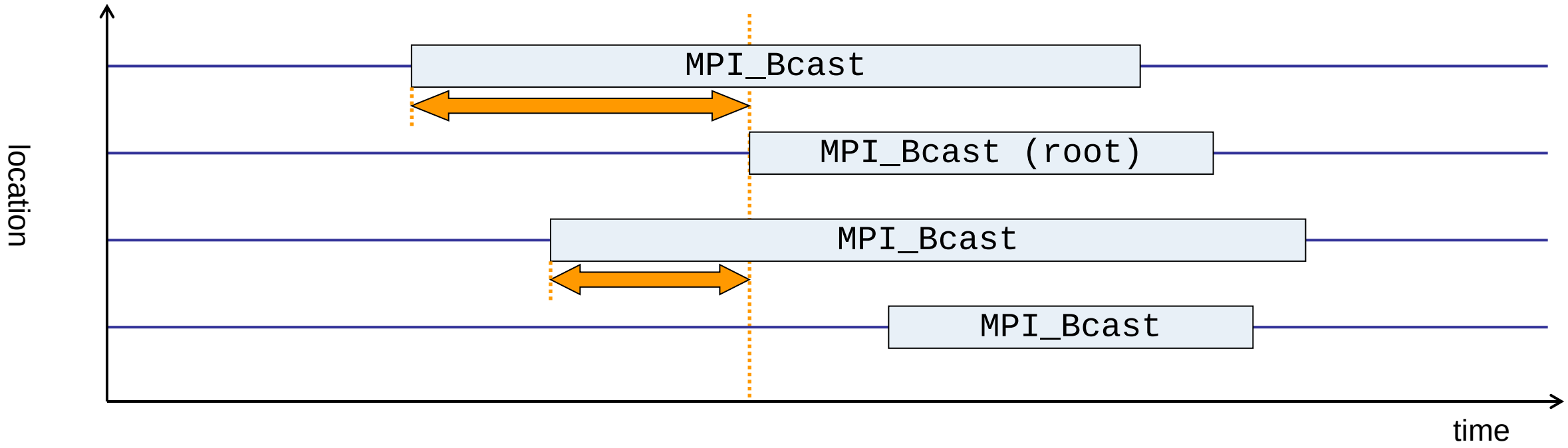


Example: Wait at NxN



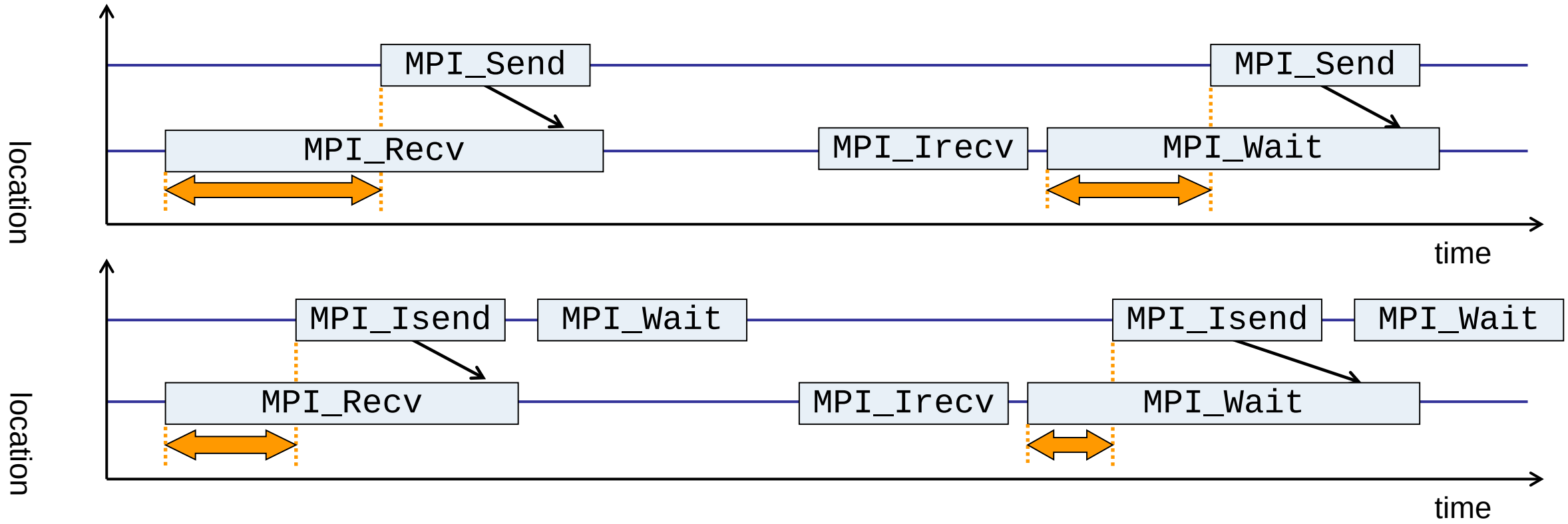
- Time spent waiting in front of synchronizing collective operation until the last process reaches the operation
- Applies to: MPI_Allgather, MPI_Allgatherv, MPI_Alltoall, MPI_Reduce_scatter, MPI_Reduce_scatter_block, MPI_Allreduce

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:
NPB-MZ-MPI / BT

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

Scalasca command – One command for (almost) everything

```
% scalasca
Scalasca 2.2
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
   -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

```
% skin
Scalasca 2.2: 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
  - Recommended: use Score-P instrumenter directly

Scalasca convenience command: scan

```
% scan
Scalasca 2.2: 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) 1998-2015 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

```
% square
Scalasca 2.2: 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
   -n                       : Do not include idle thread metric
```

- Scalasca analysis report explorer

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)

Setup environment

- Load module

```
% module load intel impi scorep scalasca cube
```

- Change to directory containing NPB3.3-MZ-MPI sources
- Existing instrumented executable in bin.scorep/ directory can be reused

BT-MZ summary measurement collection...

```
% cd bin.scorep
% cp ../jobscript/leftraru/scalasca-profile.sbatch .
% vi scalasca-profile.sbatch

 [...]

CLASS=C
NPROCS=32
EXE=./bt-mz_$CLASS.$NPROCS

#export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_TOTAL_MEMORY=78M

scalasca -analyze –s srun $EXE
```

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

```
$ sbatch ./scalasca-trace.sbatch
```

- Submit the job

BT-MZ summary measurement

.ERR FILE:

```
S=C=A=N: Scalasca 2.2.2 runtime summarization
Using SCAN_MPI_RANKS=32 processes!
S=C=A=N: ./scorep_bt-mz_C_32xO_sum experiment archive
S=C=A=N: Mon Oct 26 13:23:52 2015: Collect start
/usr/bin/srun ./bt-mz_C.32
(Here is the execution)
S=C=A=N: Mon Oct 26 13:26:14 2015: Collect done (status=0) 142s
S=C=A=N: ./scorep_bt-mz_C_32xO_sum complete.
```

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

.OUT FILE:

```
NAS Parallel Benchmarks NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
Number of zones:  16 x  16
Iterations: 200    dt:   0.000100
Number of active processes:    32
Use the default load factors with threads
Total number of threads:     32  (  1.0 threads/process)
[...] More application output
```

- Creates experiment directory:
  ./scorep_bt-mz_C_32xO_sum

BT-MZ summary analysis report examination

- Score summary analysis report

```
$ square -s  scorep_bt-mz_C_32x0_sum
INFO: Post-processing runtime summarization result...
INFO: Score report written to ./scorep_bt-mz_C_32x0_sum/scorep.score
```

- Post-processing and interactive exploration with CUBE

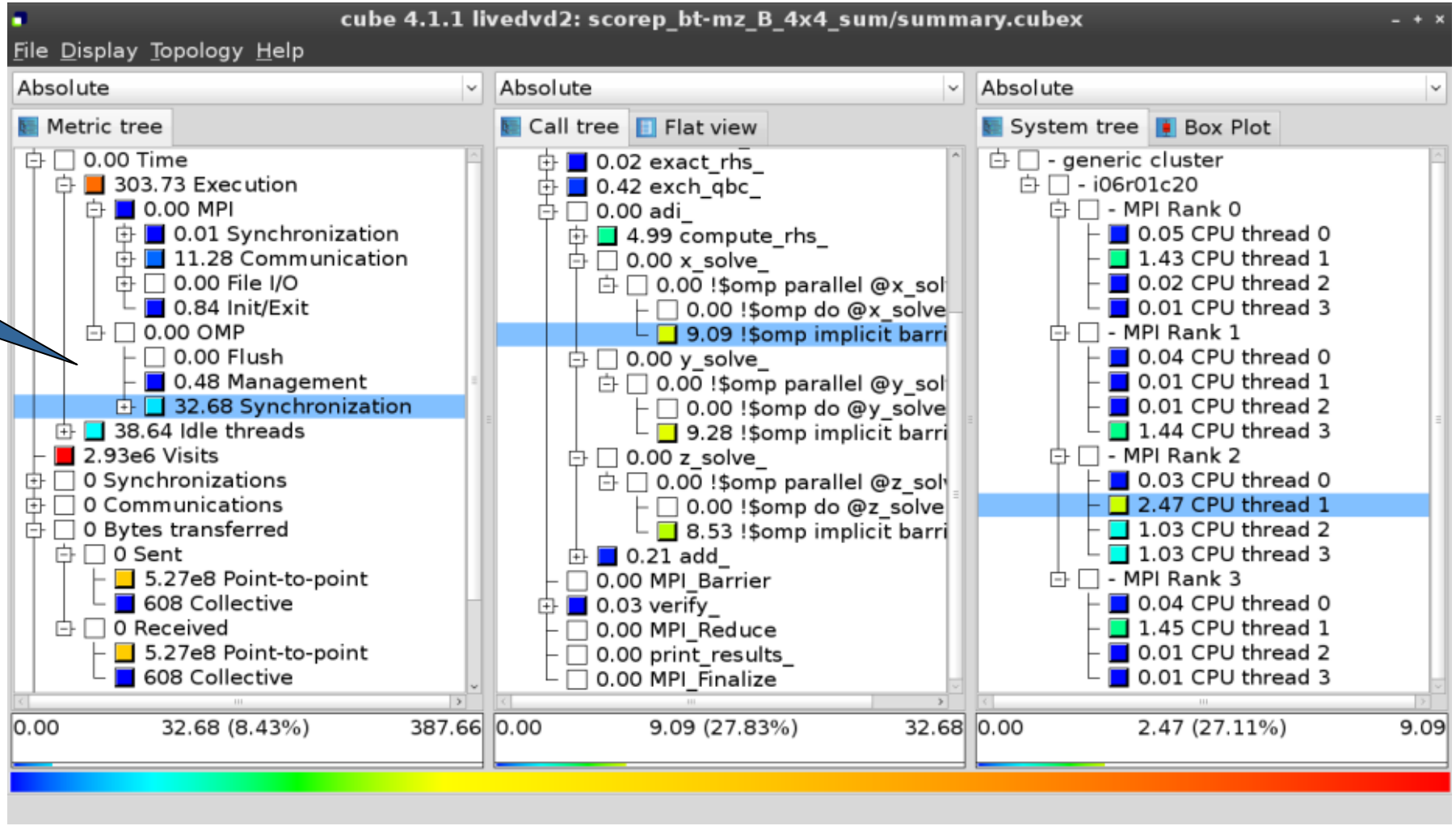
```
$ square  scorep_bt-mz_C_32x0_sum
INFO: Displaying ./scorep_bt-mz_C_32x0_sum/summary.cubex...

                [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



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...

```
% cd bin.scorep
% cp ../jobscript/leftraru/scalasca-trace.sbatch .
% vi scalasca-trace.sbatch

 [...]

CLASS=C
NPROCS=32
EXE=./bt-mz_$CLASS.$NPROCS

export SCOREP_FILTERING_FILE=../config/scorep.filt

export SCOREP_TOTAL_MEMORY=168M
export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC


scalasca -analyze -t srun $EXE



$ sbatch ./scalasca-trace.sbatch
```

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

BT-MZ trace measurement ... collection

```
S=C=A=N: Mon Oct 26 16:18:56 2015: Collect start
/usr/bin/srun ./bt-mz_C.32
NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
Number of zones:  16 x  16
Iterations: 200    dt:   0.000100
Number of active processes:    32

 [... More application output ...]

S=C=A=N: Mon Oct 26 16:20:59 2015: Collect done (status=0) 123s
```

- Starts measurement with collection of trace files ...

BT-MZ trace measurement ... analysis

```
S=C=A=N: Mon Oct 26 18:12:54 2015: Analyze start
/usr/bin/srun -n 32 /home/apps/scalasca/2.2.2/bin/scout.hyb \
./scorep_bt-mz_C_32xO_trace/traces.otf2
SCOUT   Copyright (c) 1998-2015 Forschungszentrum Juelich GmbH
        Copyright (c) 2009-2014 German Research School for Simulation
                                Sciences GmbH

Analyzing experiment archive ./scorep_bt-mz_C_32xO_trace/traces.otf2

Opening experiment archive ... done (0.033s).
Reading definition data    ... done (0.030s).
Reading event trace data   ... done (0.111s).
Preprocessing              ... done (0.072s).
Analyzing trace data       ... done (2.707s).
Writing analysis report    ... done (0.222s).

Max. memory usage         : 61.559MB

Total processing time     : 3.223s
S=C=A=N: Mon Oct 26 18:12:58 2015: Analyze done (status=0) 4s
```

- Continues with automatic (parallel) analysis of trace files

  scout.log

BT-MZ trace analysis report exploration

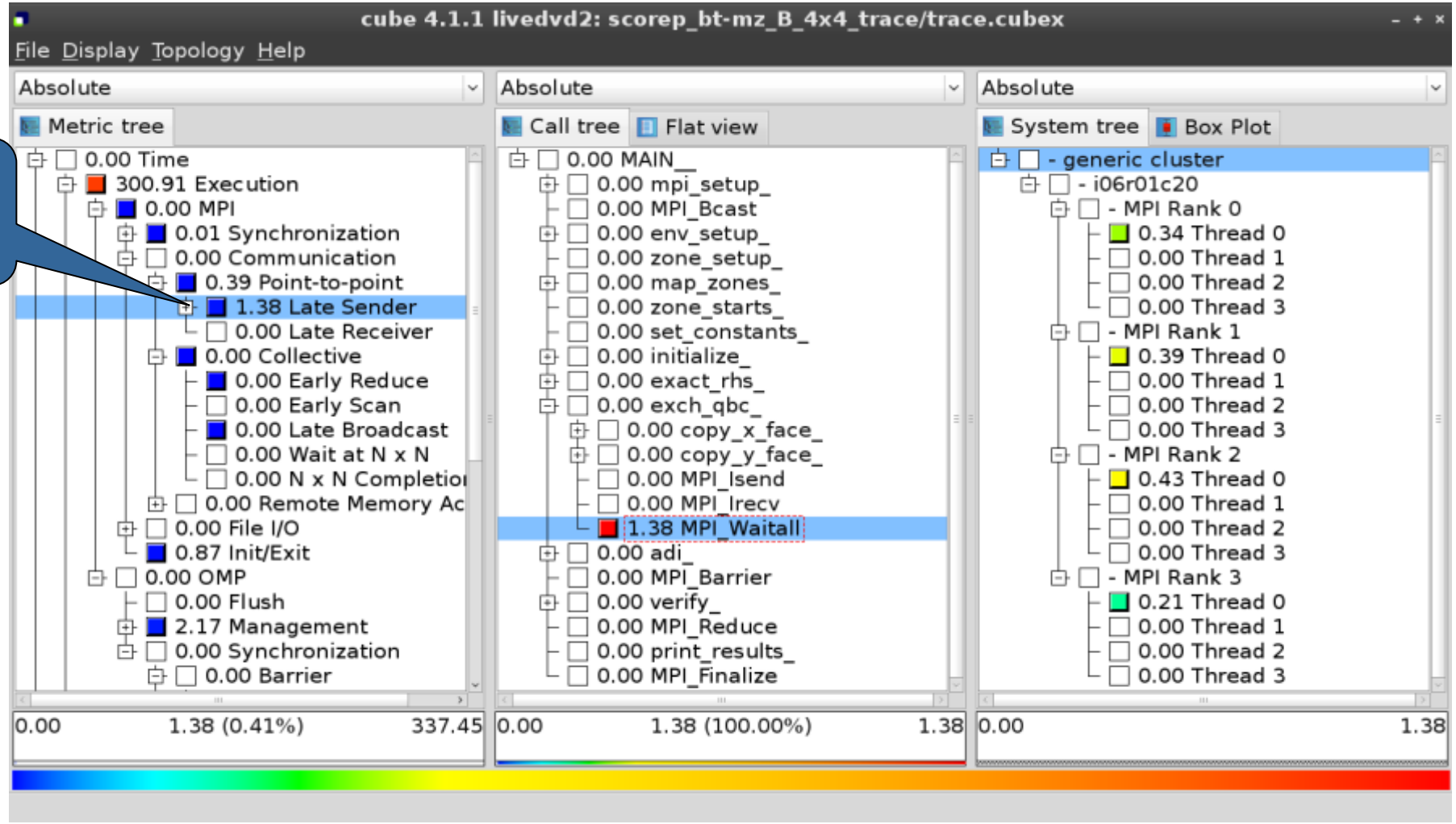
- Produces trace analysis report in experiment directory containing trace-based wait-state metrics

```
% square  scorep_bt-mz_C_32x0_trace
INFO: Post-processing runtime summarization result...
INFO: Post-processing trace analysis report...
INFO: Displaying ./scorep_bt-mz_C_32x0_trace/trace.cubex...

            [GUI showing trace analysis report]
```

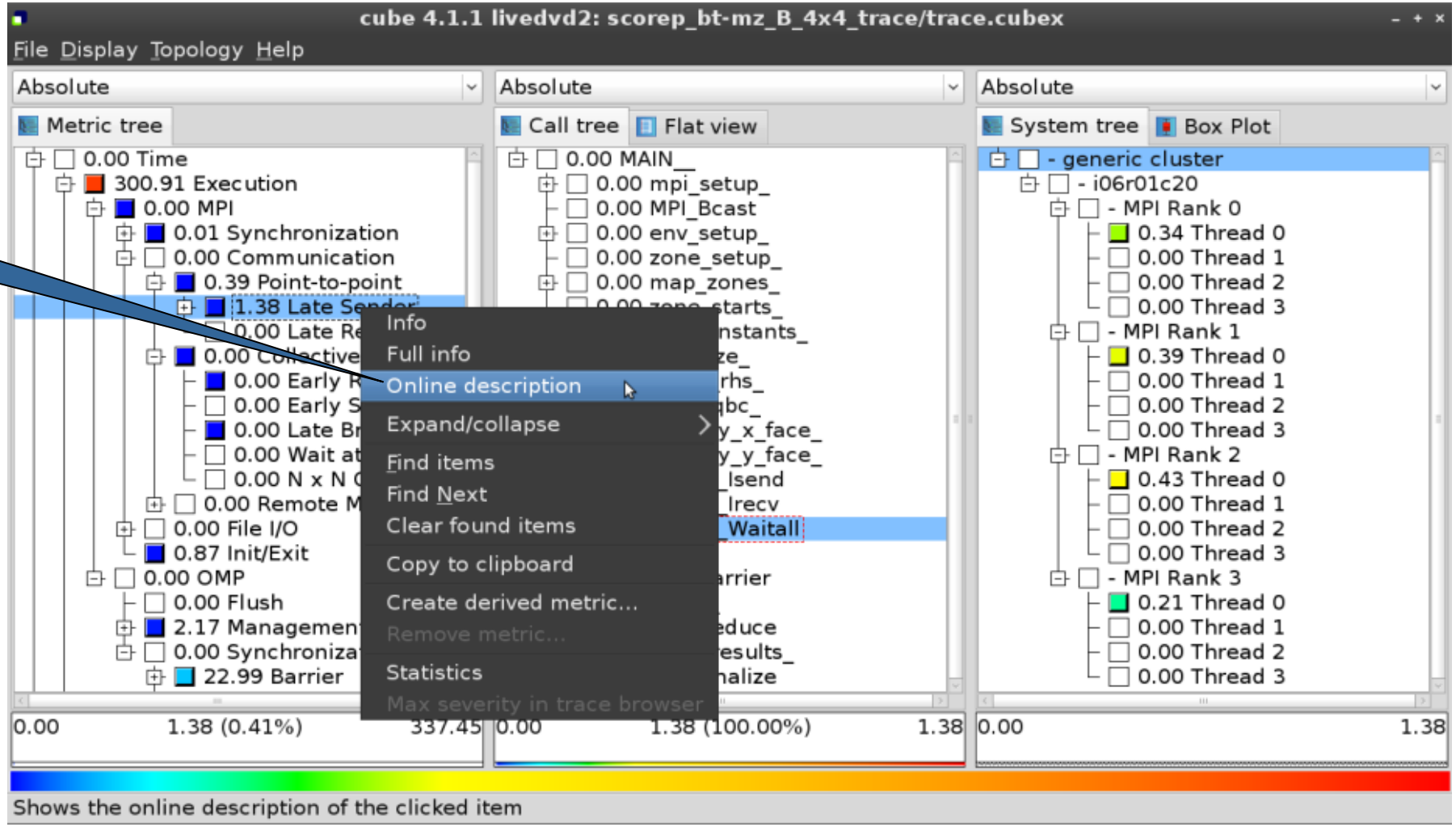
Post-processed trace analysis report

Additional trace-based metrics in metric hierarchy



Online metric description

Access online metric description via context menu



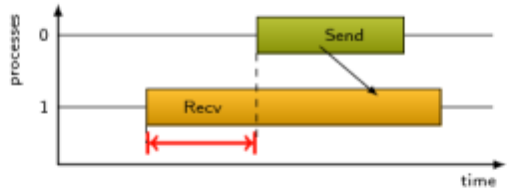
Online metric description

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:

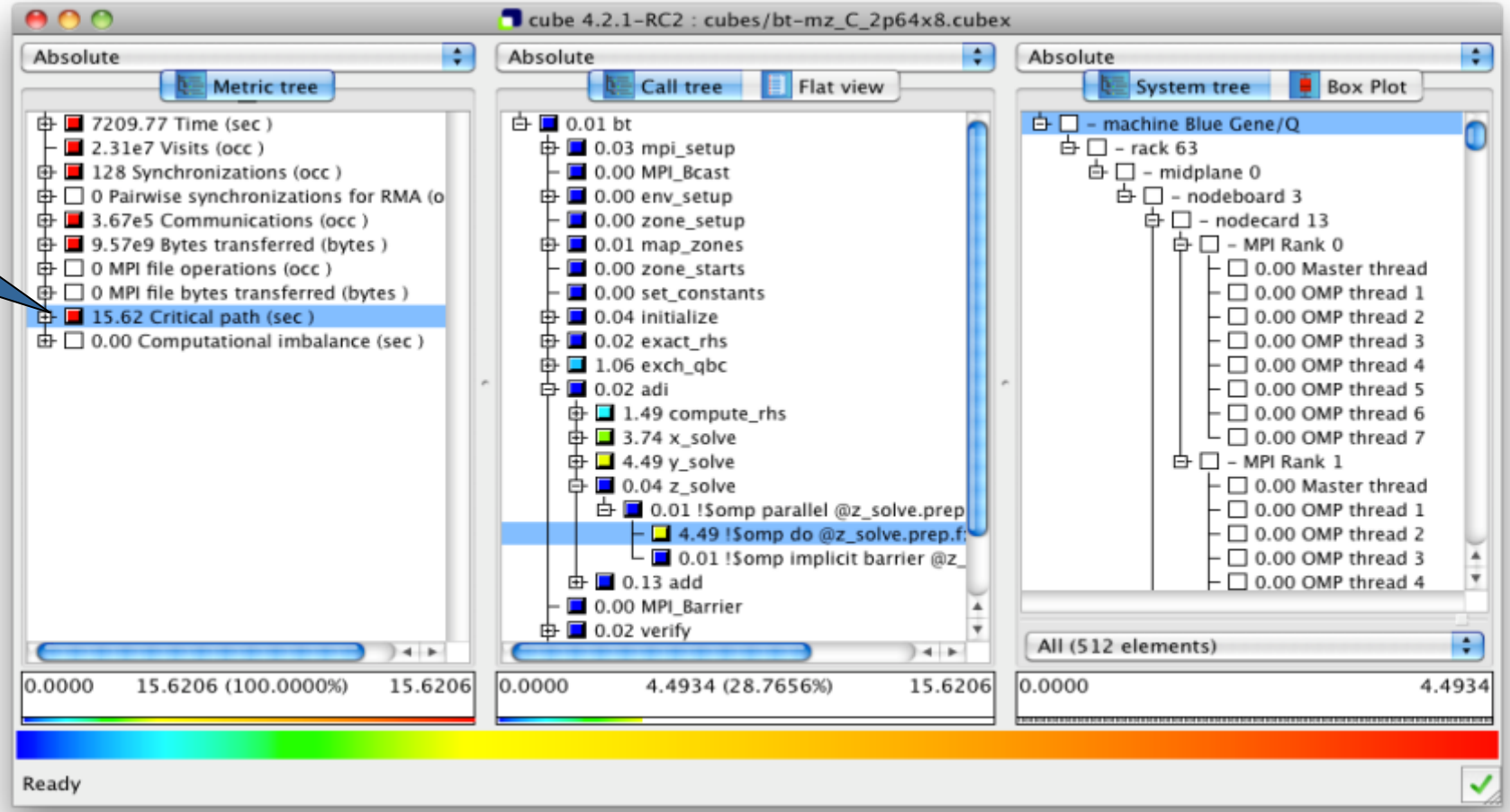
MPI Point-to-point Communication Time

Children:

Close

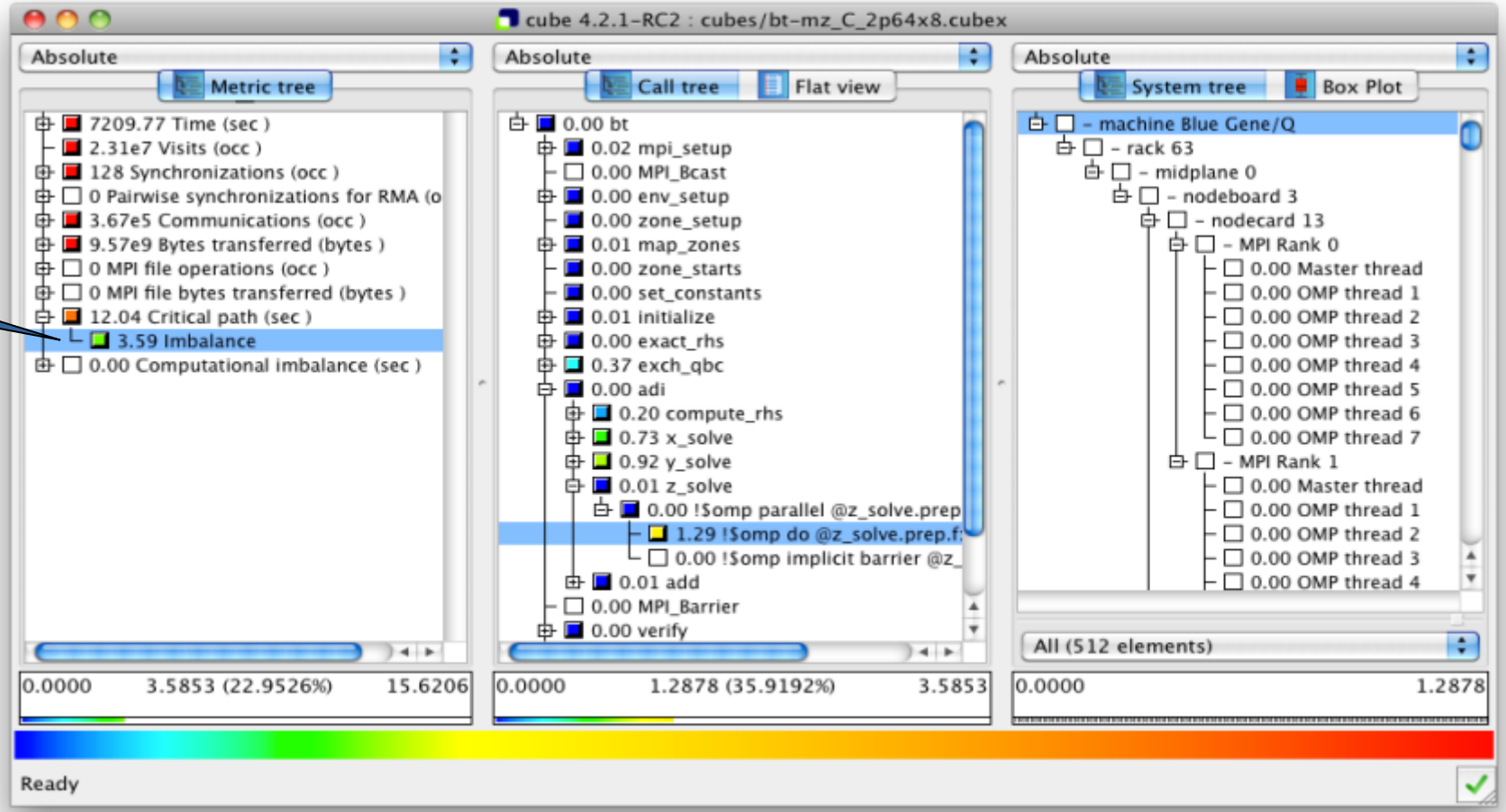
Critical-path analysis

Critical-path profile shows wall-clock time impact



Critical-path analysis

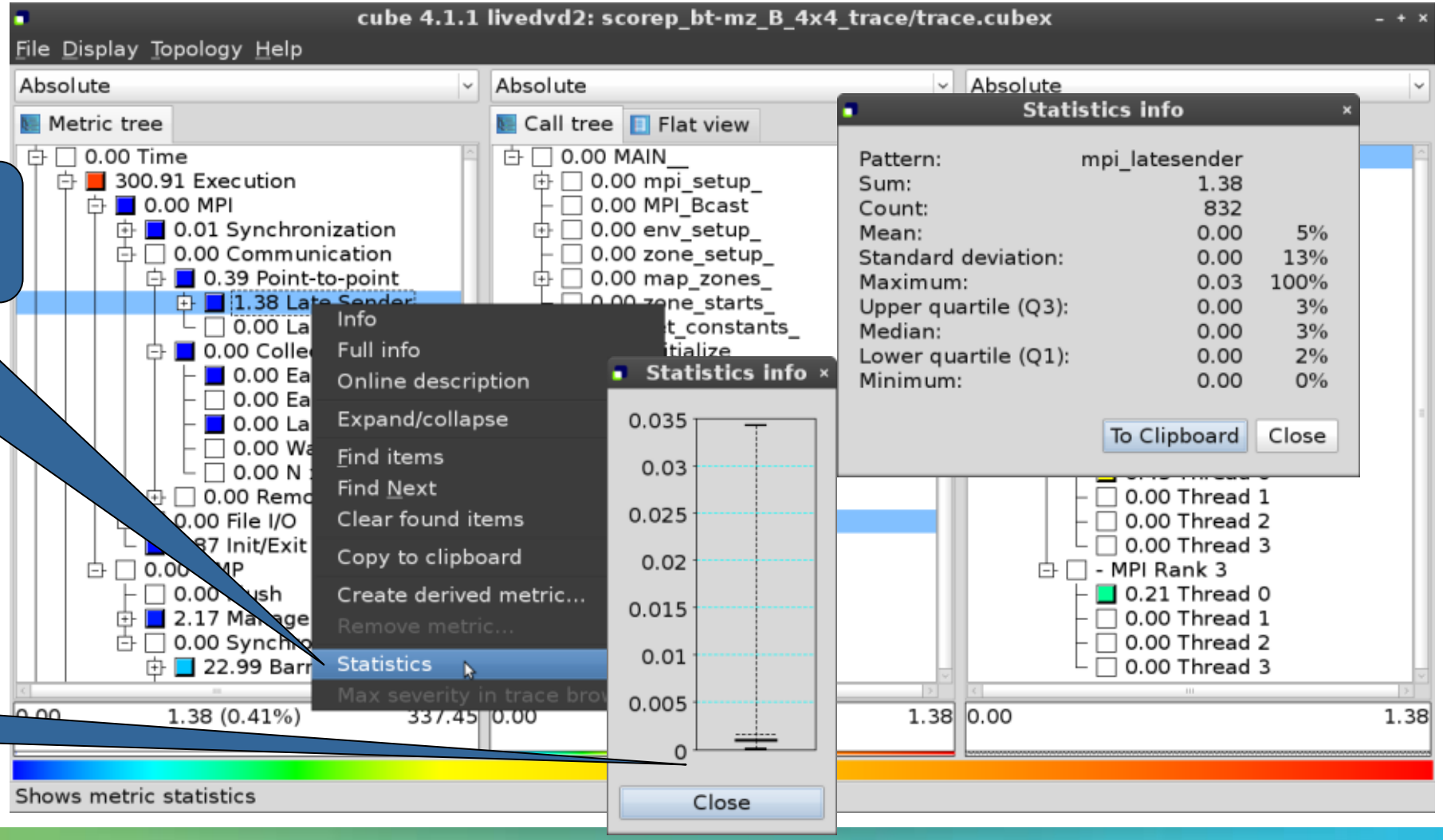
Critical-path imbalance highlights inefficient parallelism



Pattern instance statistics

Access pattern instance statistics via context menu

Click to get statistics details



Further information

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

scalasca