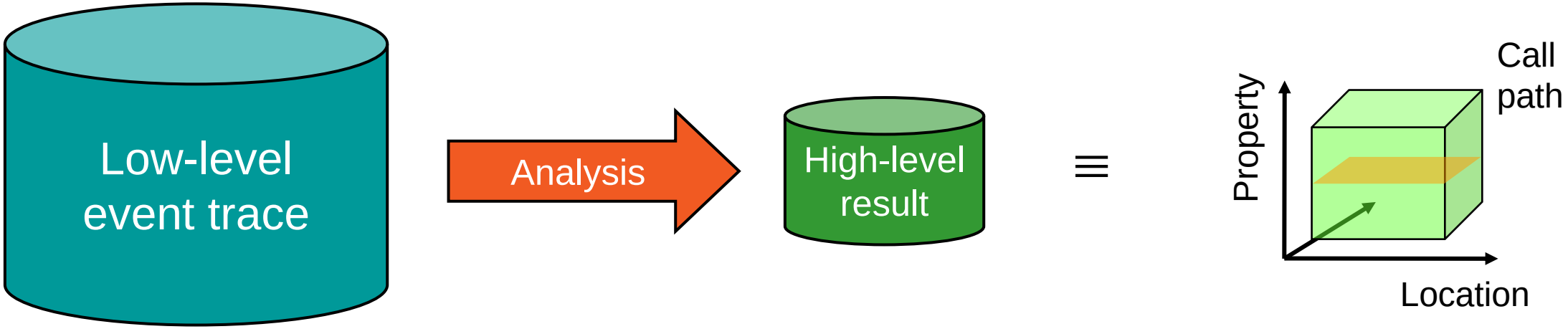
Automatic trace analysis with Scalasca

Christian Feld
Jülich Supercomputing Centre

scalasca

Automatic trace analysis

- 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: 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

  - Technische Universität Darmstadt - Laboratory for Parallel Programming

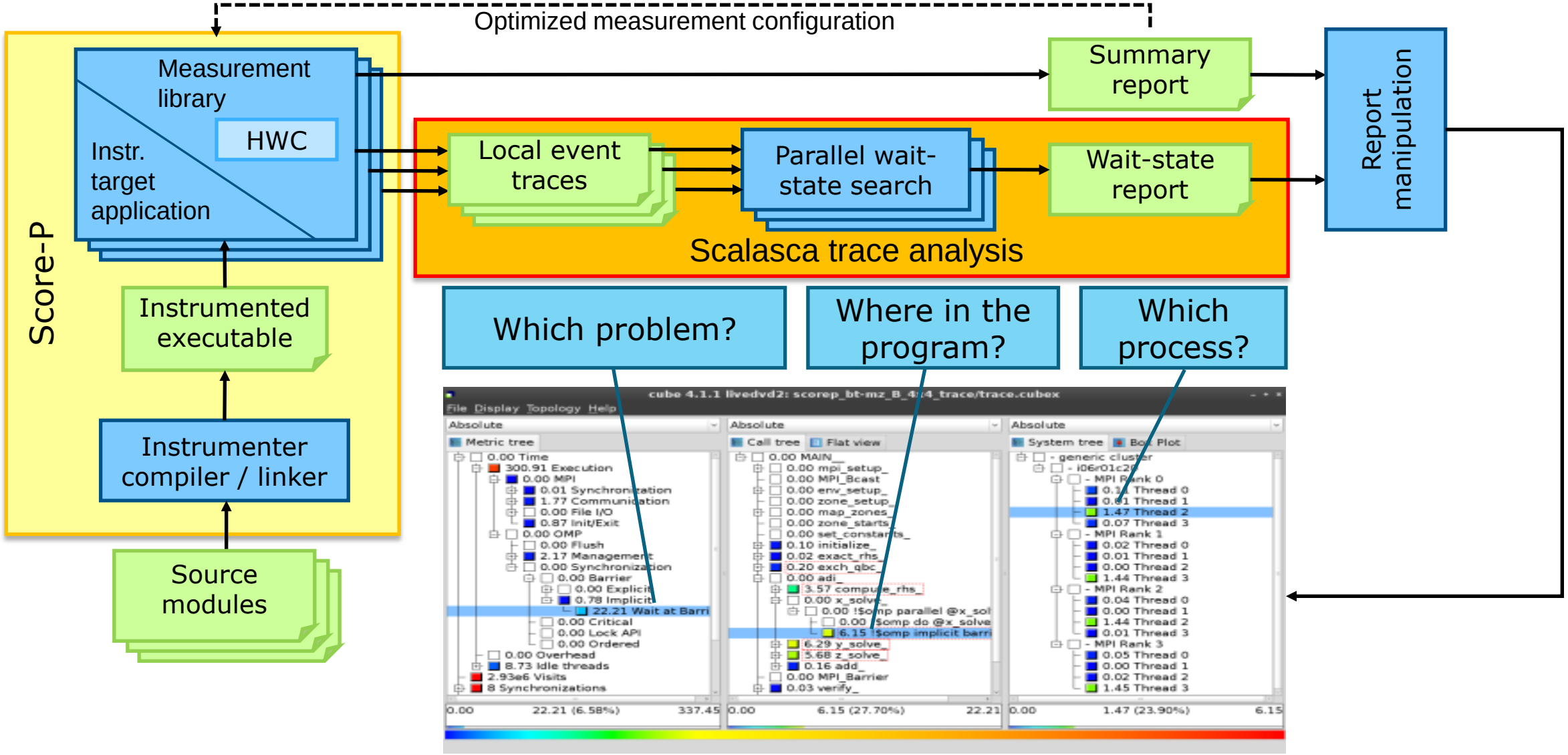
The Scalasca project: Objective

- Development of a **scalable** performance analysis toolset for most popular parallel programming paradigms
- Specifically targeting **large-scale** parallel applications
  - such as those running on K computer, IBM BlueGene or Cray systems with one million or more processes/threads
- Latest release:
  - Scalasca v2.3 coordinated with Score-P v2.0.1 (April 2016)
    - Sampling support, still experimental
    - Improved analysis report postprocessing
    - Documentation improvements
    - Various algorithmic optimizations reducing overall analysis time for traces of multi-threaded applications:
      - Improved memory management.
      - Improved trace preprocessing.
      - Improved timestamp correction.
    - Bug fixes

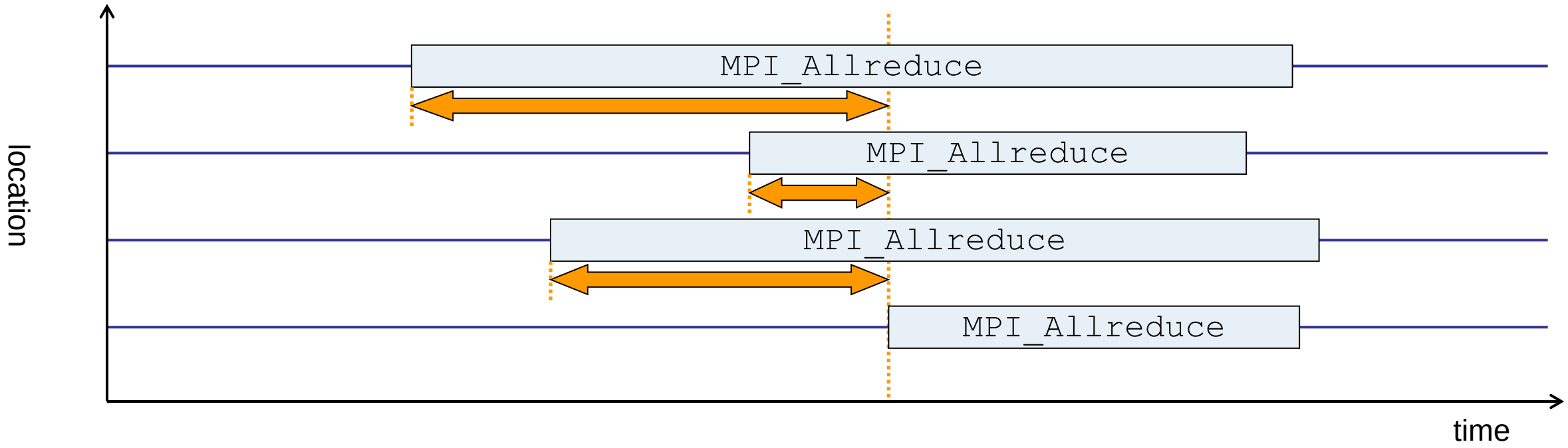
Scalasca 2.2 features

- 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

Scalasca workflow

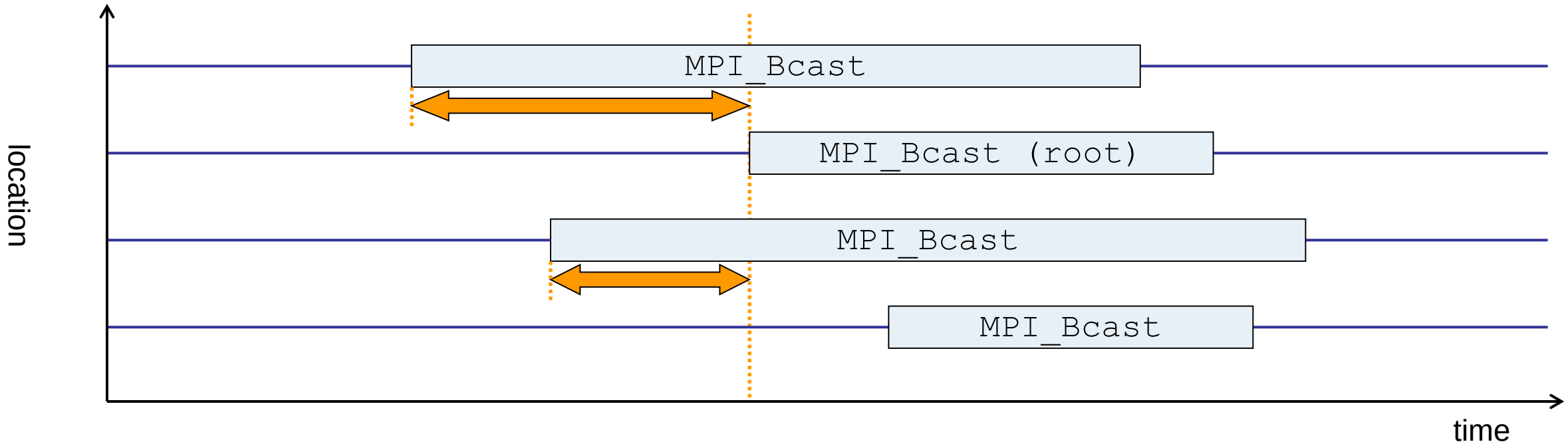


Example: Wait at NxN



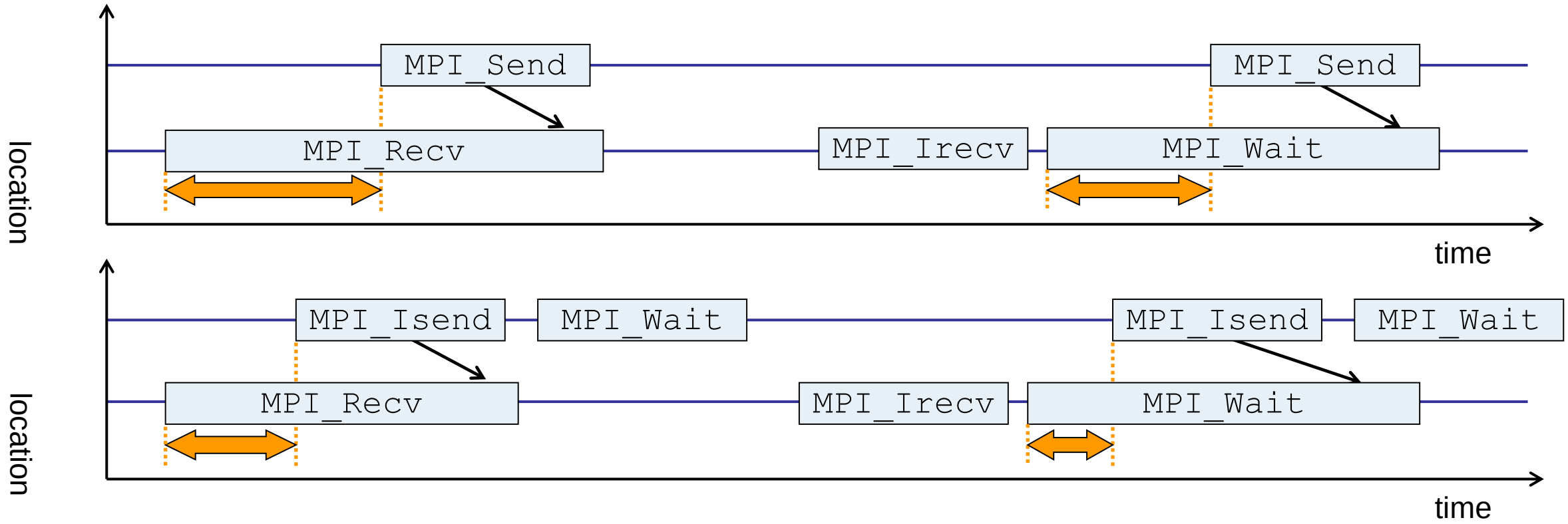
- Time spent waiting in front of synchronizing collective operation until the last process reaches the operation
- Applies to: MPI_Allgather, MPI_Allgatherv, MPI_Alltoall, MPI_Reduce_scatter, MPI_Reduce_scatter_block, MPI_Allreduce

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

---

scalasca

Scalasca command – One command for (almost) everything

```
% scalasca
Scalasca 2.3
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 2.3: 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 2.3: 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

```
% square
Scalasca 2.3: 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)

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

Setup environment

- Ensure `scalasca` found on PATH
  - on login nodes for experiment examination & post-processing:

```
% source /home/hpc/a2c06/lu23bud/LRZ-VIHPSTW21/tools/source-me.scorep-2.0.1.mpt.sh
% which scalasca
/home/hpc/a2c06/lu23bud/LRZ-VIHPSTW21/tools/scalasca/REL-2.3/intel-mpi.mpt/bin/scalasca
```

  - on compute nodes within jobscripts:

```
# add tools to PATH
source /home/hpc/a2c06/lu23bud/LRZ-VIHPSTW21/tools/source-me.scorep-2.0.1.mpt.sh
# alternatively add just scalasca
export PATH=/home/hpc/a2c06/lu23bud/LRZ-VIHPSTW21/tools/scalasca/REL-2.3/intel-mpi.mpt/bin:$PATH
```

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

BT-MZ summary measurement collection...

```
% cd bin.scorep
% cp ../jobscript/lrz_uv2_mpt/scalasca2.mpt.sbatch .
% vi scalasca2.mpt.sbatch
 [...]
export OMP_NUM_THREADS=4
CLASS=B
NPROCS=4
EXE=./bt-mz_$CLASS.$NPROCS

export SCOREP_TIMER=clock_gettime
export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_FP_INS
#export SCOREP_TOTAL_MEMORY=66M

export SCAN_MPI_LAUNCHER=srun_ps
NEXUS="scalasca -analyze -s"
$NEXUS srun_ps "-n $NPROCS -t $OMP_NUM_THREADS" $EXE
```

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

default ‘s’ summary profile collection)

```
% sbatch ./scalasca2.mpt.sbatch
```

- Submit the job

BT-MZ summary measurement

```
S=C=A=N: Scalasca 2.3 runtime summarization
S=C=A=N: ./scorep_bt-mz_B_4x4_sum experiment archive
S=C=A=N: Tue Apr 19 07:52:11 2016: Collect start
/lrz/sys/tools/slurm_utils/bin/srun_ps -n 4 -t 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:     8

 [... More application output ...]

S=C=A=N: 19 07:53:25 2016: Collect done (status=0) 74s
S=C=A=N: ./scorep_bt-mz_B_4x4_sum complete.
```

- Run the application using the Scalasca measurement collection & analysis nexus prefixed to launch command
- Creates experiment directory:
  ./scorep_bt-mz_B_4x4_sum

BT-MZ summary analysis report examination

- Score summary analysis report

```
% square -s  scorep_bt-mz_B_4x4_sum
INFO: Post-processing runtime summarization result...
INFO: Score report written to ./scorep_bt-mz_B_8x4_sum/scorep.score
```

- Post-processing and interactive exploration with CUBE

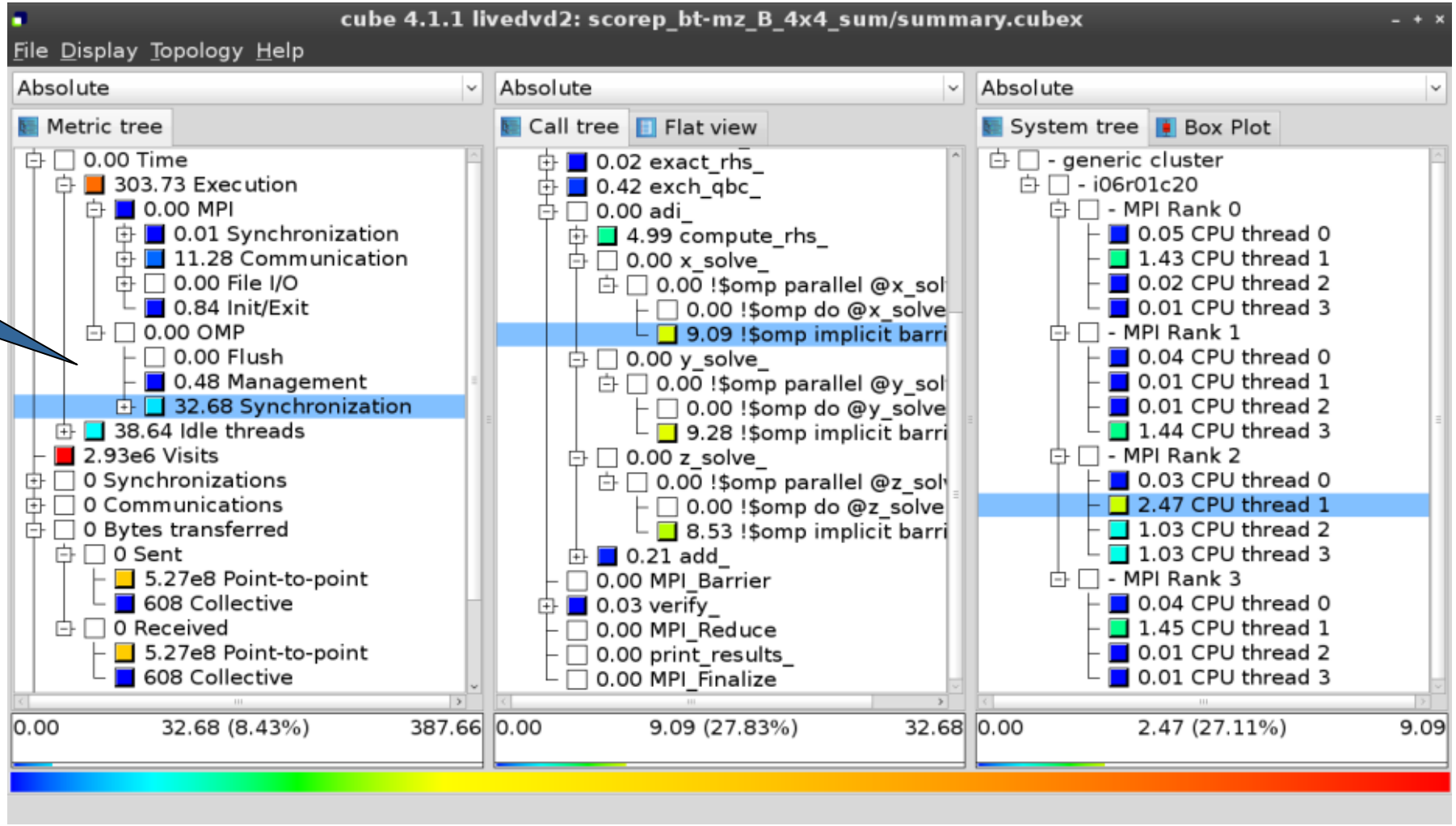
```
% square  scorep_bt-mz_B_4x4_sum
INFO: Displaying ./scorep_bt-mz_B_4x4_sum/summary.cubex...

                [GUI showing summary analysis report]
```

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

Post-processed summary analysis report

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/lrz_uv2_mpt/scalasca2.mpt.sbatch .
% vi scalasca2.mpt.sbatch

 [...]

export OMP_NUM_THREADS=4
CLASS=B
NPROCS=4
EXE=./bt-mz_$CLASS.$NPROCS

export SCOREP_TIMER=clock_gettime
export SCOREP_FILTERING_FILE=../config/scorep.filt
export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_FP_INS
export SCOREP_TOTAL_MEMORY=66M

export SCAN_MPI_LAUNCHER=srun_ps
NEXUS="scalasca -analyze -t"
$NEXUS srun_ps "-n $NPROCS -t $OMP_NUM_THREADS" $EXE

% sbatch ./scalasca2.mpi.sbatch
```

- Change to directory with executable and edit job script

use ‘-t’ option for trace collection & analysis

- Submit the job

BT-MZ trace measurement … collection

```
S=C=A=N: Scalasca 2.3 trace collection and analysis
S=C=A=N: Fri Sep 20 15:09:59 2013: Collect start
srun_ps -n 4 -t 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: Tue Apr 19 08:09:51 2016: Collect done (status=0) 77s
```

- Starts measurement with collection of trace files …

BT-MZ trace measurement … analysis

```
S=C=A=N: Tue Apr 19 08:09:52 2016: Analyze start
srun_ps -n 4 -t 4 scout.hyb ./scorep_bt-mz_B_4x4_trace/traces.otf2

Analyzing experiment archive
                  ./scorep_bt-mz_B_`4x4_trace/traces.otf2

Analyzing experiment archive
         ./scorep_bt-mz_B_4x4_trace/traces.otf2

Opening experiment archive ... done (0.003s).
Reading definition data    ... done (0.004s).
Reading event trace data   ... done (2.821s).
Preprocessing              ... done (0.558s).
Analyzing trace data       ... done (28.628s).
Writing analysis report    ... done (0.670s).


Total processing time     : 32.802s
S=C=A=N: Tue Apr 19 08:10:34 2016: Analyze done (status=0) 42s
```

- Continues with automatic (parallel) analysis of trace files

BT-MZ trace analysis report exploration

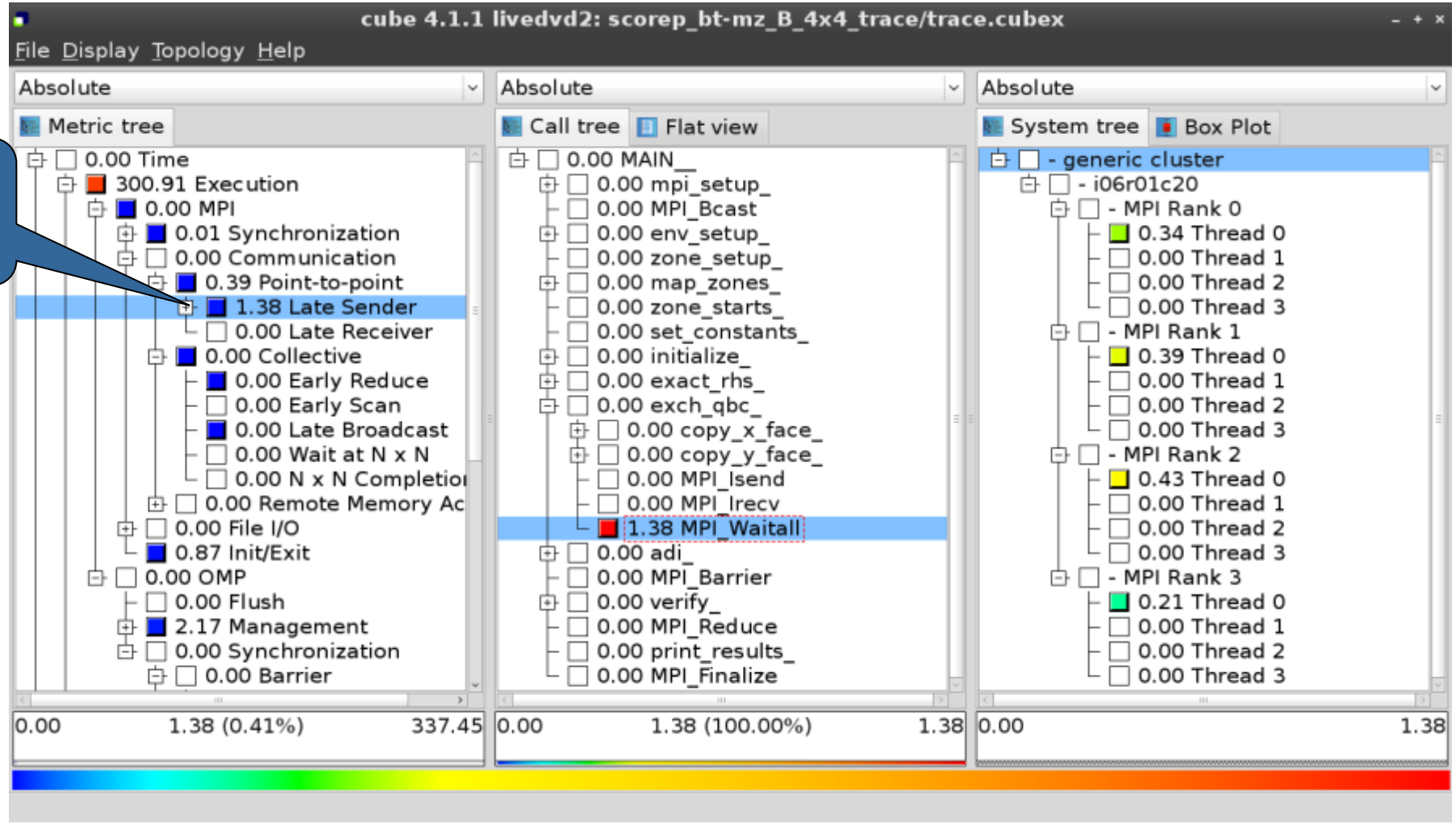
- Produces trace analysis report in experiment directory containing trace-based wait-state metrics

```
% square  scorep_bt-mz_B_4x4_trace
INFO: Post-processing runtime summarization result...
INFO: Post-processing trace analysis report...
INFO: Displaying ./scorep_bt-mz_B_4x4_trace/trace.cubex...

            [GUI showing trace analysis report]
```

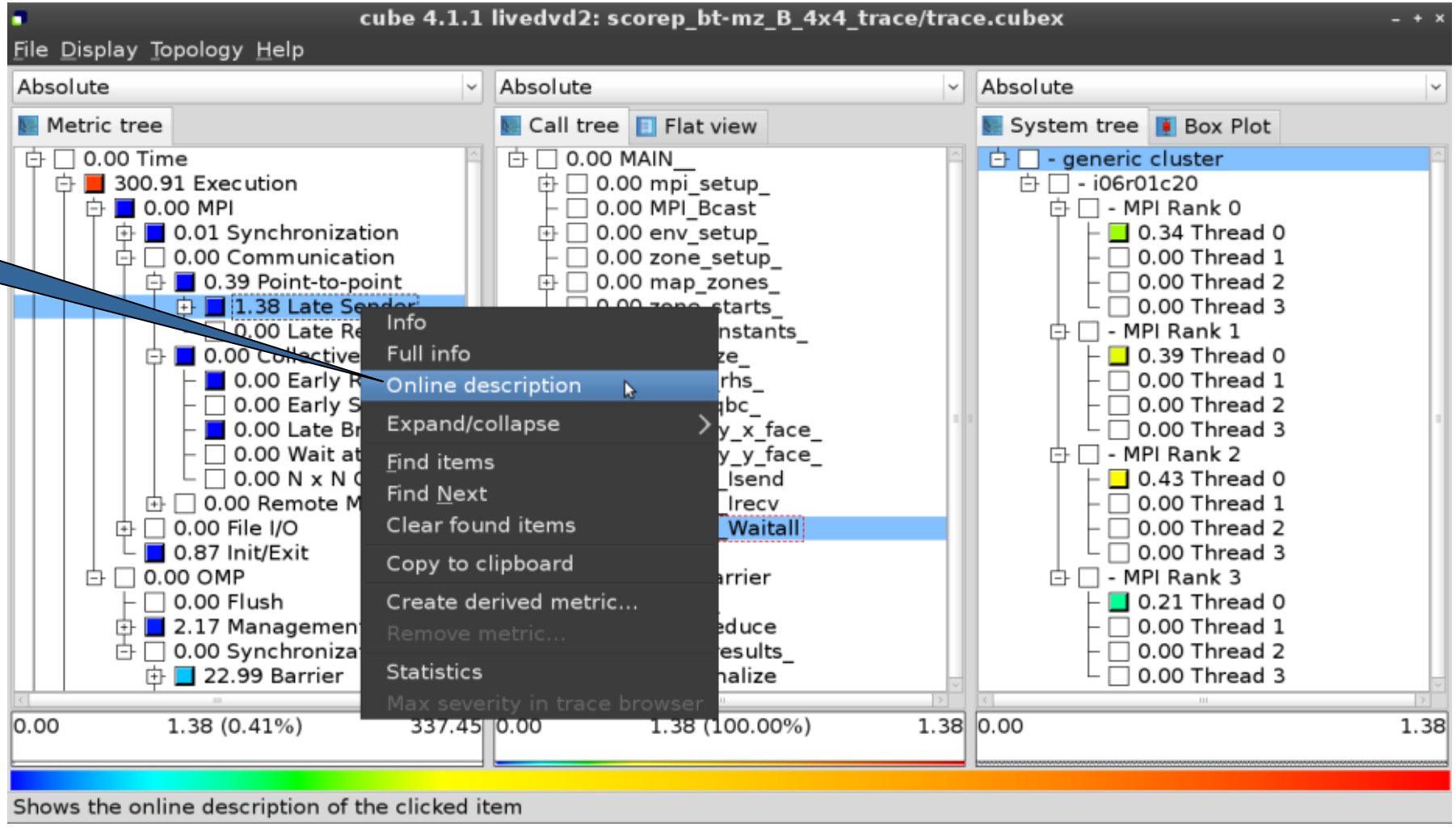
Post-processed trace analysis report

Additional trace-based metrics in metric hierarchy



Online metric description

Access online metric description via context menu



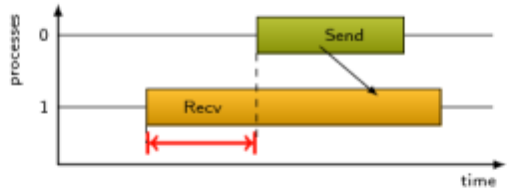
Online metric description

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:

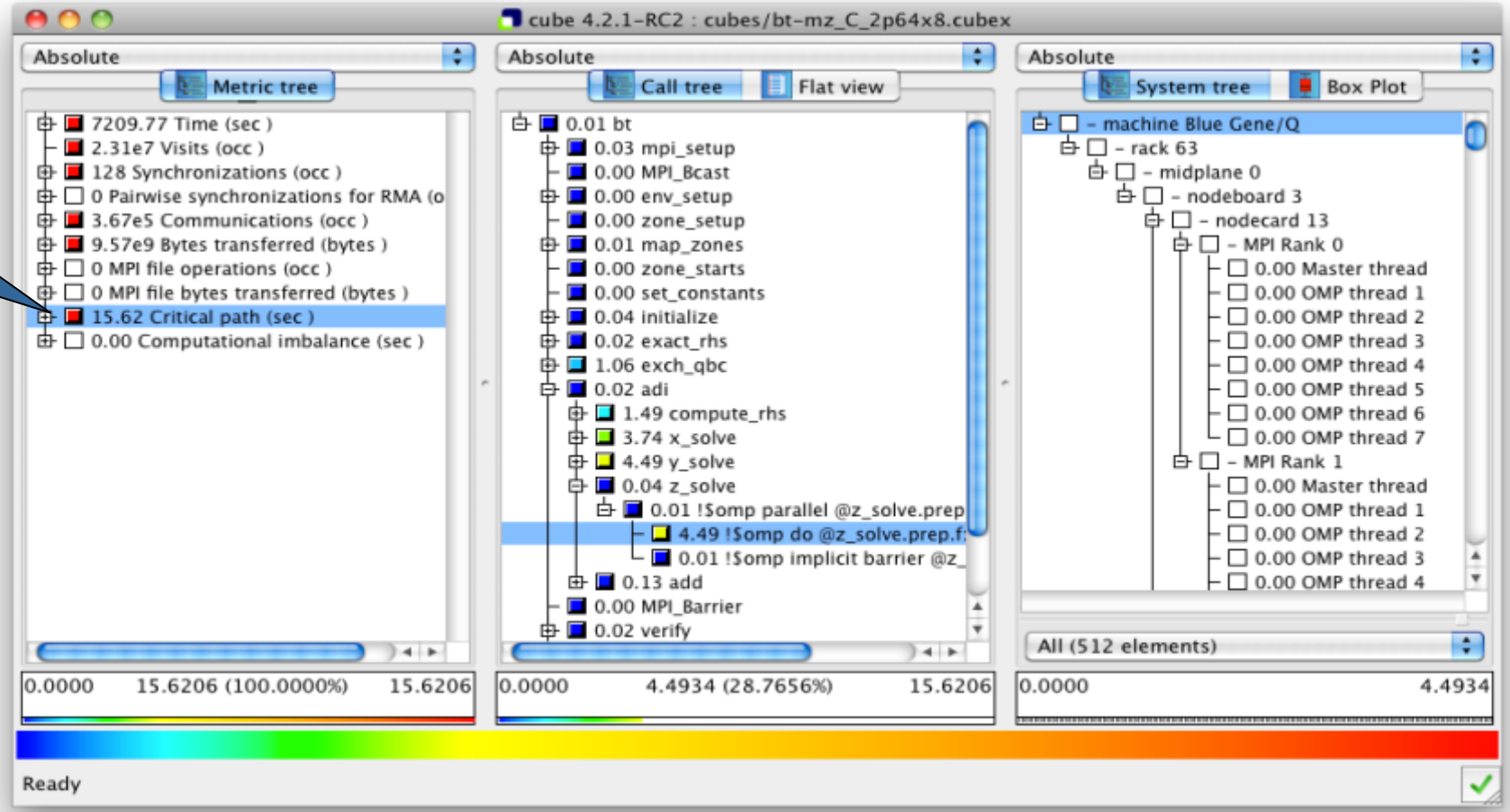
MPI Point-to-point Communication Time

Children:

Close

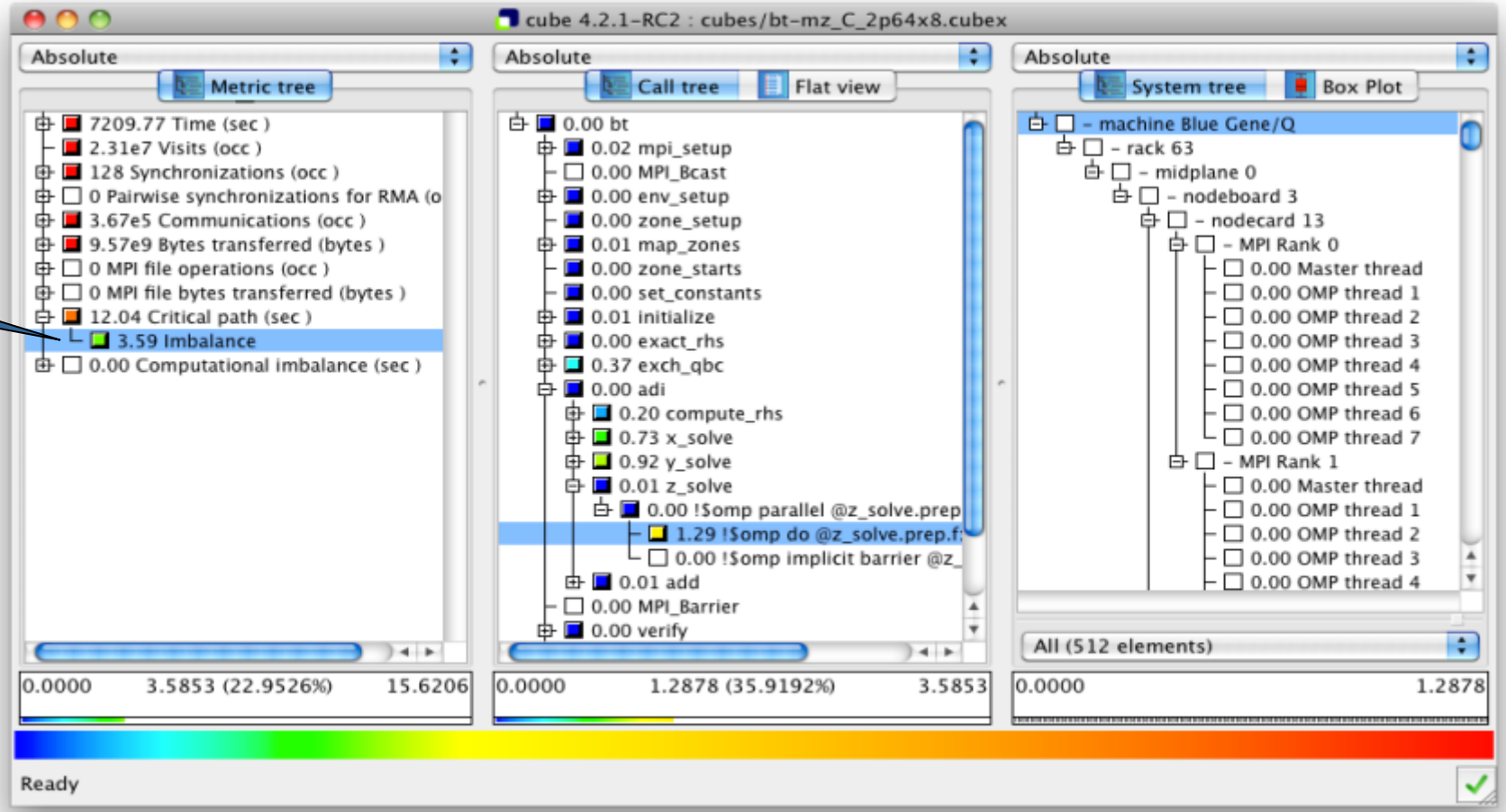
Critical-path analysis

Critical-path profile shows wall-clock time impact



Critical-path analysis

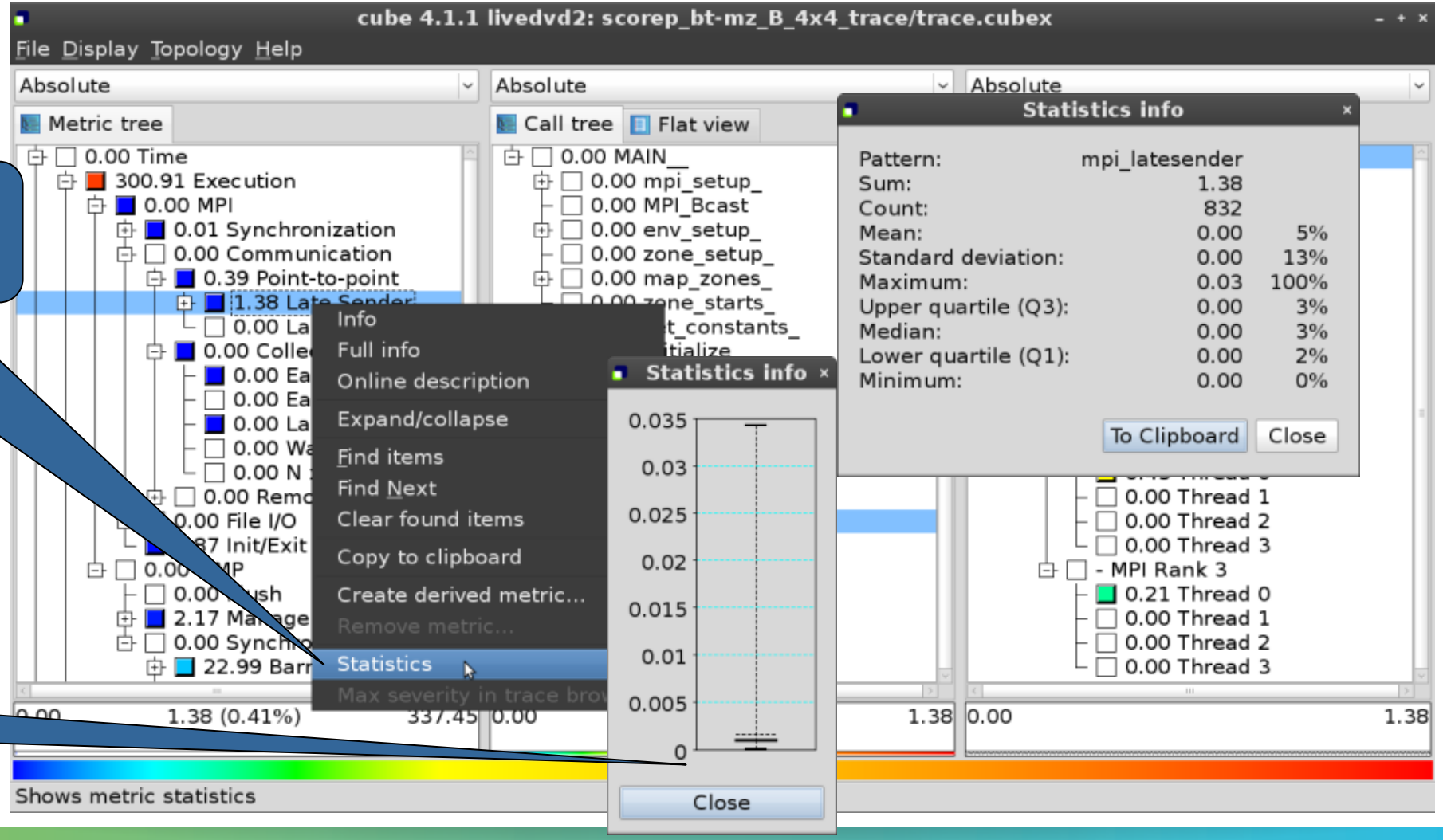
Critical-path imbalance highlights inefficient parallelism



Pattern instance statistics

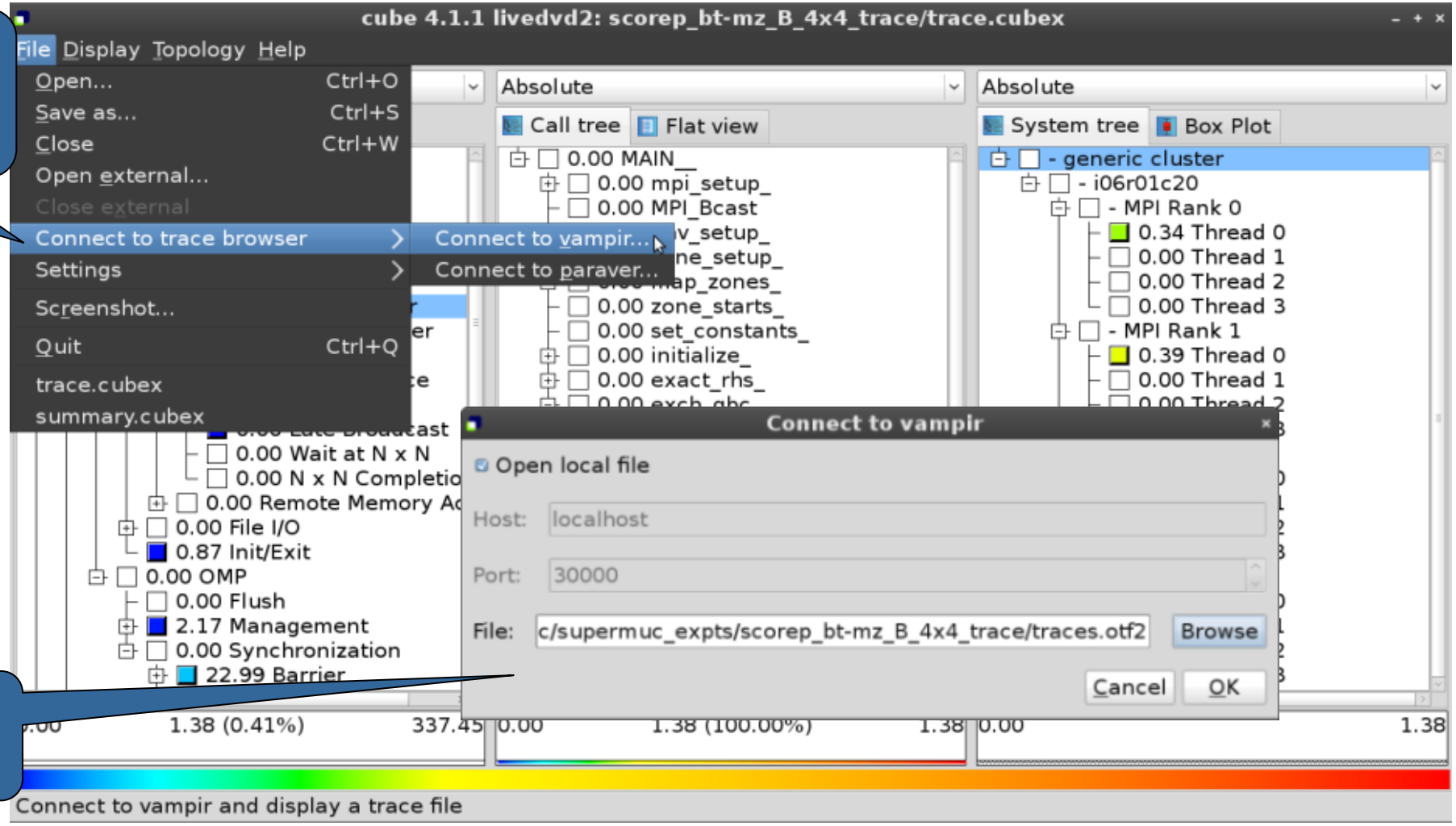
Access pattern instance statistics via context menu

Click to get statistics details



Connect to Vampir trace browser

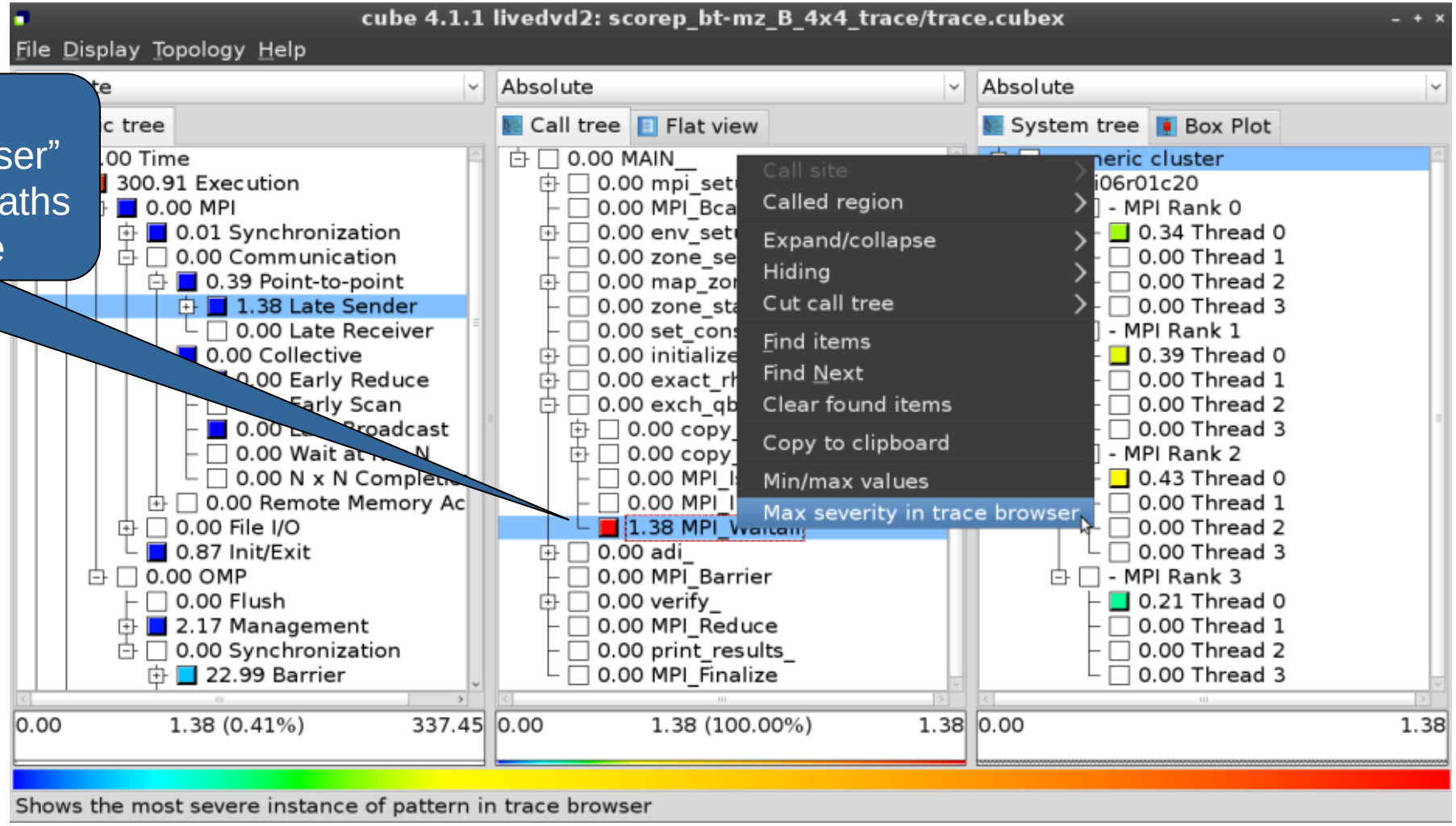
To investigate most severe pattern instances, connect to a trace browser…



…and select trace file from the experiment directory

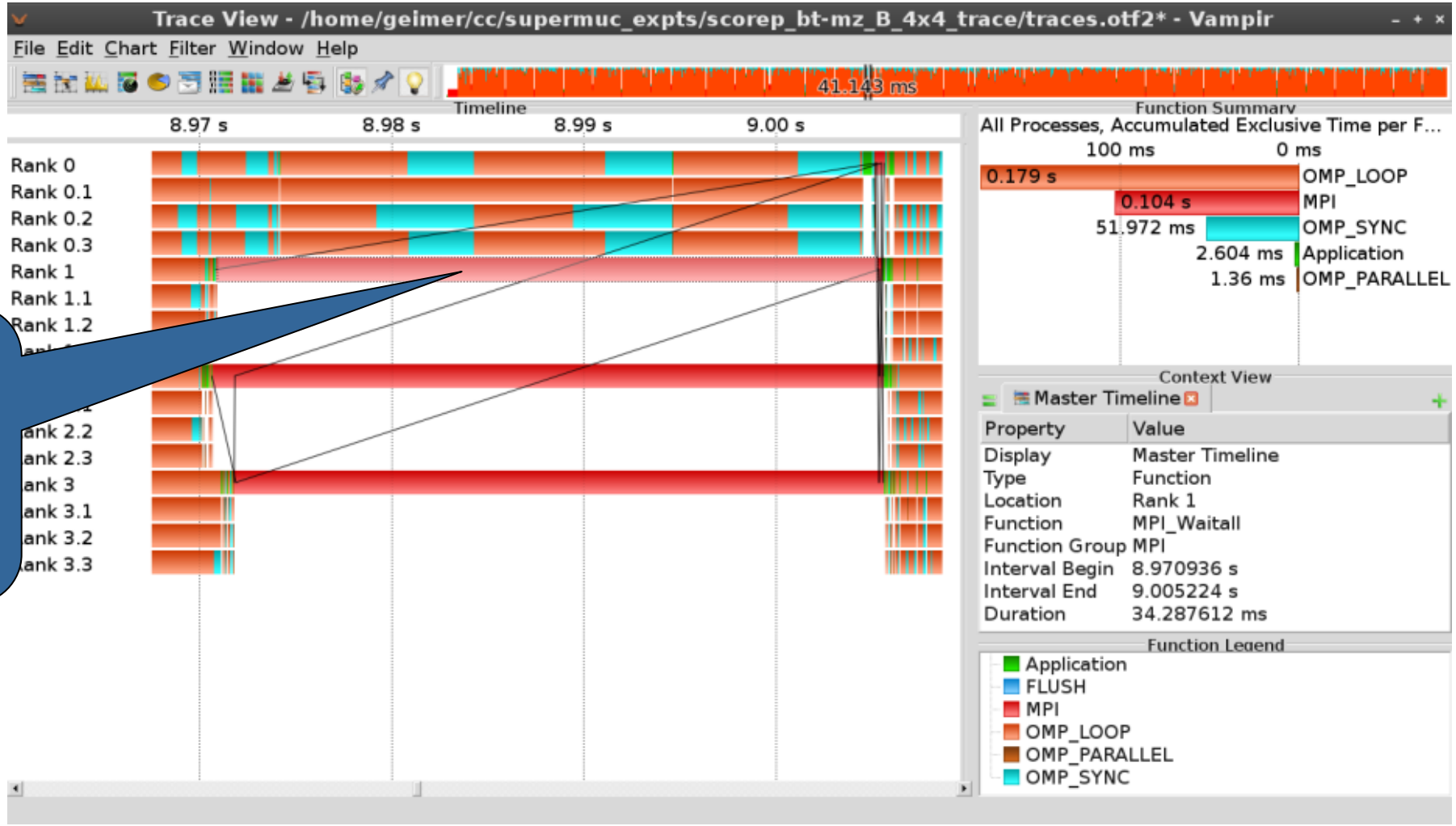
Show most severe pattern instances

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



Investigate most severe instance in Vampir

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



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