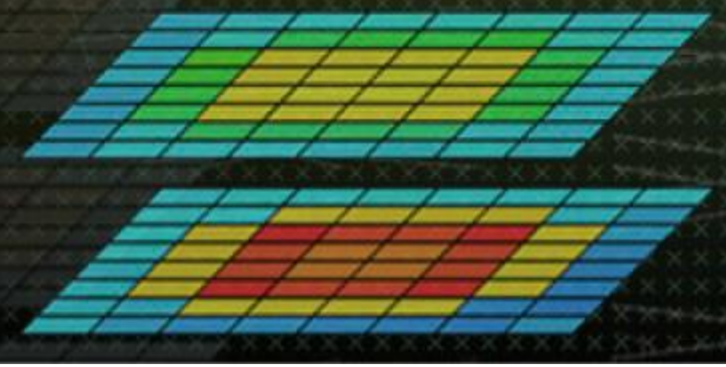
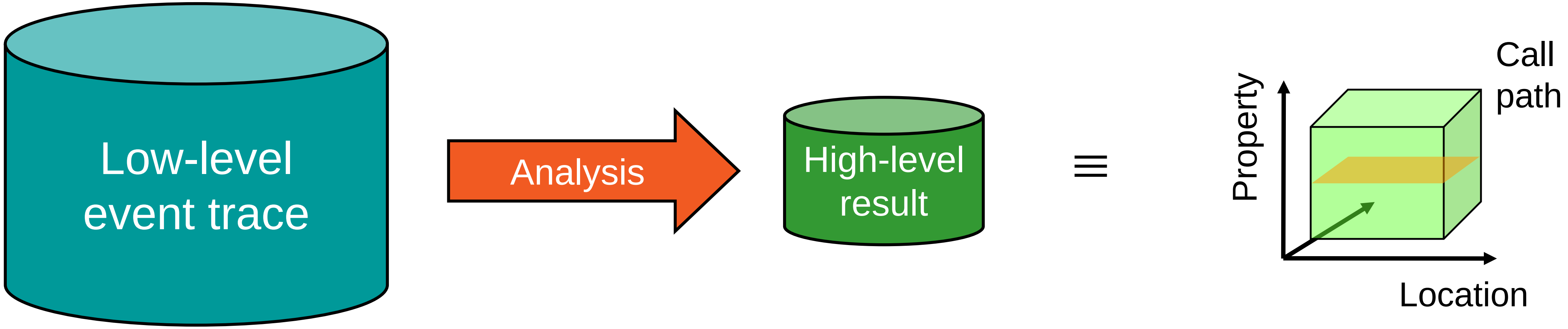


Automatic trace analysis with the Scalasca Trace Tools

Brian Wylie
Jülich Supercomputing Centre

Automatic trace analysis

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

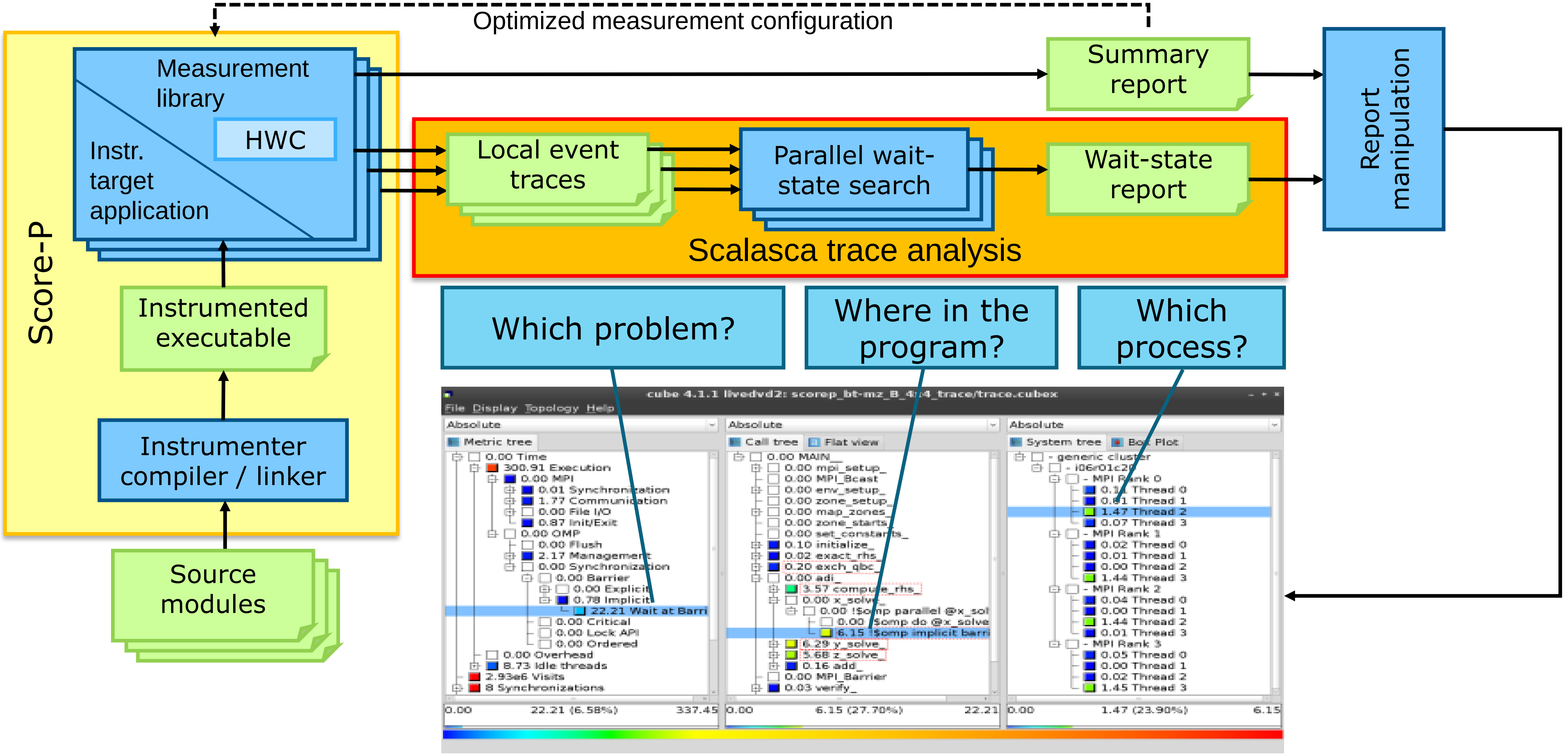
- Specifically targeting large-scale parallel applications
  - Such as those running on IBM Blue Gene or Cray systems with one million or more processes/threads

- Latest release:
  - Scalasca v2.5 coordinated with Score-P v5.0 (March 2019) also works with Score-P v6.0

Scalasca Trace Tools features

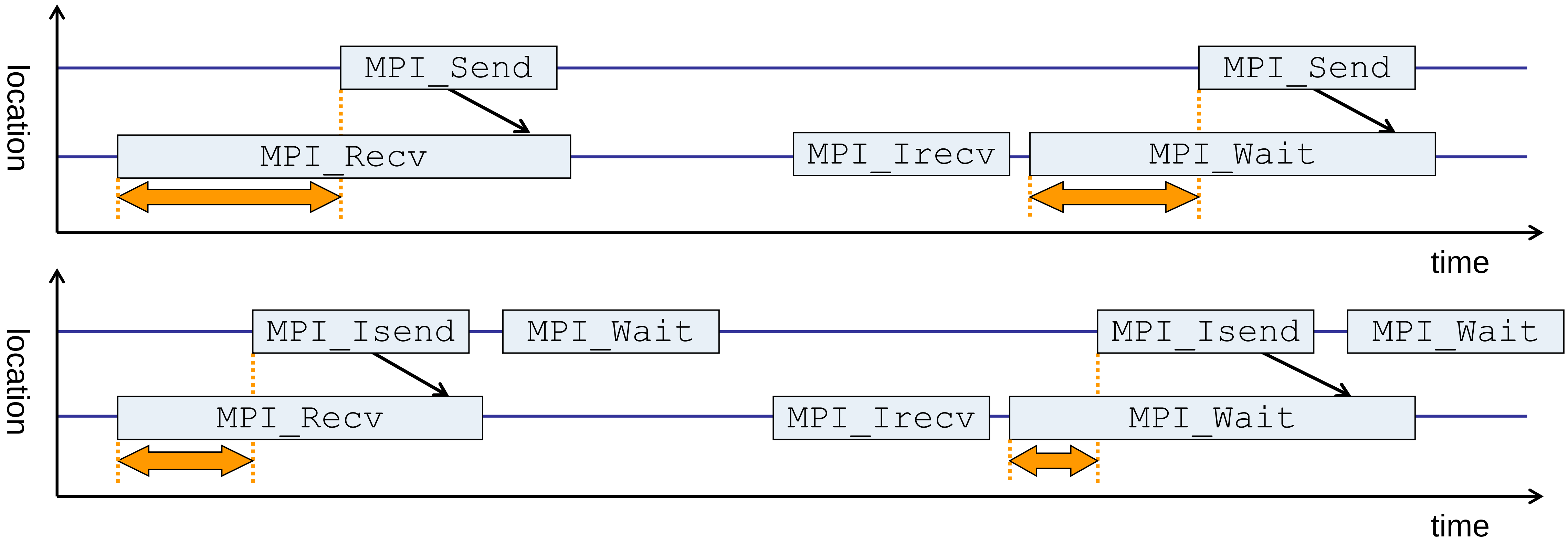
- Open source, 3-clause BSD license
- Fairly portable
  - IBM Blue Gene, Cray XT/XE/XK/XC, SGI Altix, Fujitsu FX10/100 & K computer, Linux clusters (x86, Power, ARM), Intel Xeon Phi, ...
- Uses Score-P instrumenter & measurement libraries
  - Scalasca v2 core package focuses on trace-based analyses
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

Scalasca workflow



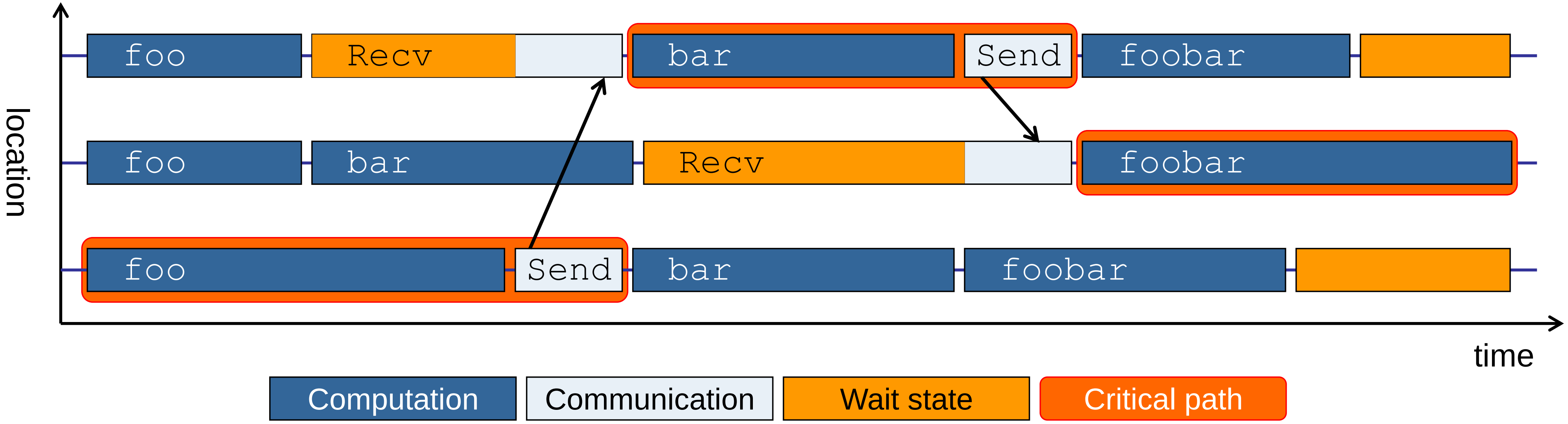
```
cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex

Metric tree                      Call tree                              System tree
0.00 Time                        0.00 MAIN__                            - generic cluster
  300.91 Execution                 0.00 mpi_setup_                        - i06r01c20
    0.00 MPI                       0.00 MPI_Bcast                           - MPI Rank 0
      0.01 Synchronization         0.00 env_setup_                            0.11 Thread 0
      1.77 Communication           0.00 zone_setup_                           0.01 Thread 1
      0.00 File I/O                0.00 map_zones_                            1.47 Thread 2
      0.87 Init/Exit               0.00 zone_starts_                          0.07 Thread 3
    0.00 OMP                       0.00 set_constants_                      - MPI Rank 1
      0.00 Flush                   0.10 initialize_                           0.02 Thread 0
      2.17 Management              0.02 exact_rhs_                            0.01 Thread 1
      0.00 Synchronization         0.20 exch_qbc_                             0.00 Thread 2
        0.00 Barrier               0.00 adi_                                  1.44 Thread 3
          0.00 Explicit              3.57 compute_rhs_                      - MPI Rank 2
          0.78 Implicit              0.00 x_solve_                            0.04 Thread 0
            22.21 Wait at Barri        0.00 !$omp parallel @x_sol             0.00 Thread 1
        0.00 Critical                    0.00 !$omp do @x_solve               1.44 Thread 2
        0.00 Lock API                    6.15 !$omp implicit barri            0.01 Thread 3
        0.00 Ordered                 6.29 y_solve_                          - MPI Rank 3
    0.00 Overhead                    5.68 z_solve_                            0.05 Thread 0
    8.73 Idle threads                0.16 add_                                0.00 Thread 1
  2.93e6 Visits                    0.00 MPI_Barrier                           0.02 Thread 2
  8 Synchronizations               0.03 verify_                               1.45 Thread 3

0.00   22.21 (6.58%)   337.45   0.00   6.15 (27.70%)   22.21   0.00   1.47 (23.90%)   6.15
```

Example: “*Late Sender*” wait state



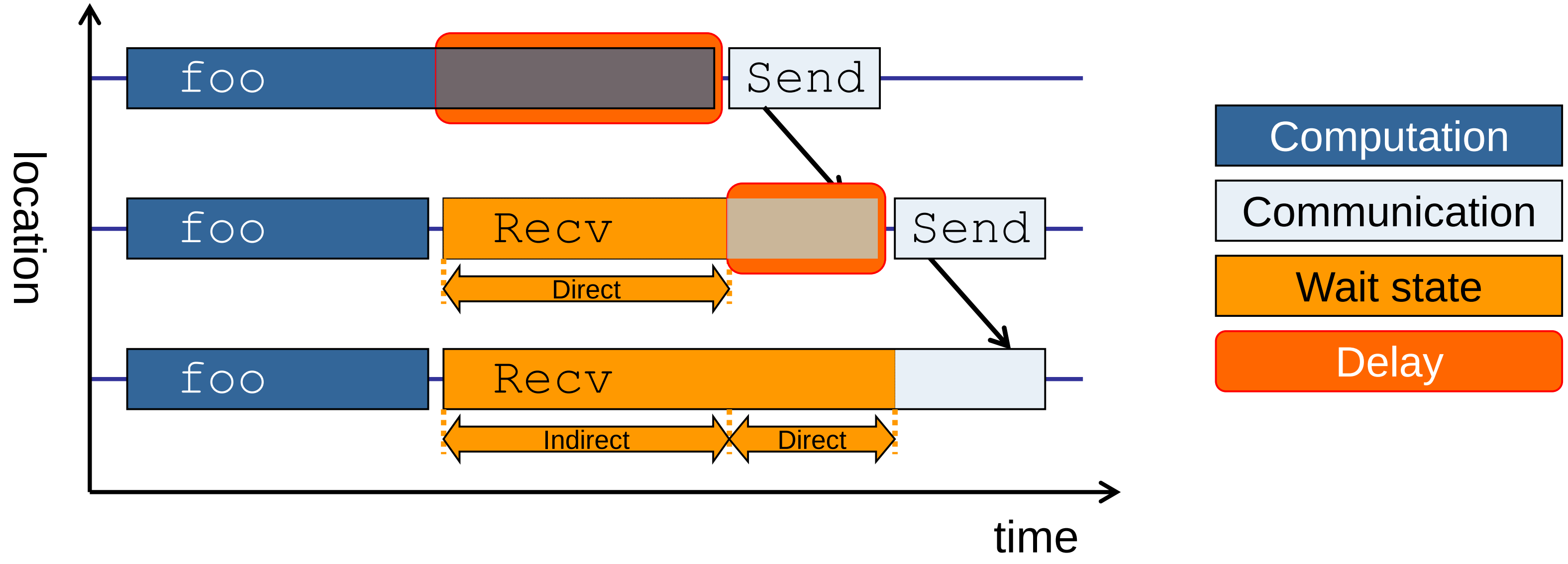
- Waiting time caused by a blocking receive operation posted earlier than the corresponding send
- Applies to blocking as well as non-blocking communication

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

trace tools
scalasca

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 2.5
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 2.5: 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

Scalasca convenience command: square / scalasca -examine

```
% square
Scalasca 2.5: 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)

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)

Recap: Local installation (Hawk)

- Select appropriate environment

```
% module load gcc mpt
```

- Latest/recent versions of all VI-HPS tools not yet installed system-wide
  - Add extra module path
  - Required for each shell session

```
% module use /zhome/academic/HLRS/hlrs/hpcoft28/spack/modulefiles7/lmod/linux-centos8-x86_64/Core
% module load scalasca scorep cube
```

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

```
% cd $WORK/NPB3.3-MZ-MPI
```

BT-MZ summary measurement collection...

```
% cd bin.scorep
% cp ../jobscript/hawk/scalasca.pbs .
% cat scalasca.pbs

# Score-P measurement configuration
export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC
#export SCOREP_METRIC_RUSAGE=ru_stime
#export SCOREP_METRIC_RUSAGE_PER_PROCESS=ru_maxrss
#export SCOREP_TOTAL_MEMORY=71M

# Scalasca measurement & analysis configuration
module load scalasca
export SCAN_TARGET=$EXE

# Run the application using Scalasca collection & analysis nexus
scan -s   mpirun -np $NPROCS   omplace -nt $OMP_NUM_THREADS   $EXE
```

- Change to directory with the Score-P instrumented executable and edit the job script

Hint:
scan = scalasca -analyze
-s = profile/summary (def)

```
% qsub -q R_tw scalasca.pbs
```

- Submit the job

BT-MZ summary measurement

```
S=C=A=N: Scalasca 2.5 runtime summarization
S=C=A=N: ./scorep_bt-mz_C_16x8_sum experiment archive
S=C=A=N: Thu Apr 11 13:25:35 2019: Collect start
mpirun -np 16 ./bt-mz_C.16

 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:    16

 [... More application output ...]

S=C=A=N: Thu Apr 11 13:25:50 2019: Collect done (status=0) 15s
S=C=A=N: ./scorep_bt-mz_C_16x8_sum complete.
```

- Run the application using the Scalasca measurement collection & analysis nexus prefixed to launch command
- Creates experiment directory: scorep_bt-mz_C_16x8_sum

BT-MZ summary analysis report examination

- Score summary analysis report

```
% square -s  scorep_bt-mz_C_16x8_sum
INFO: Post-processing runtime summarization result...
INFO: Score report written to ./scorep_bt-mz_C_16x8_sum/scorep.score
```

- Post-processing and interactive exploration with Cube

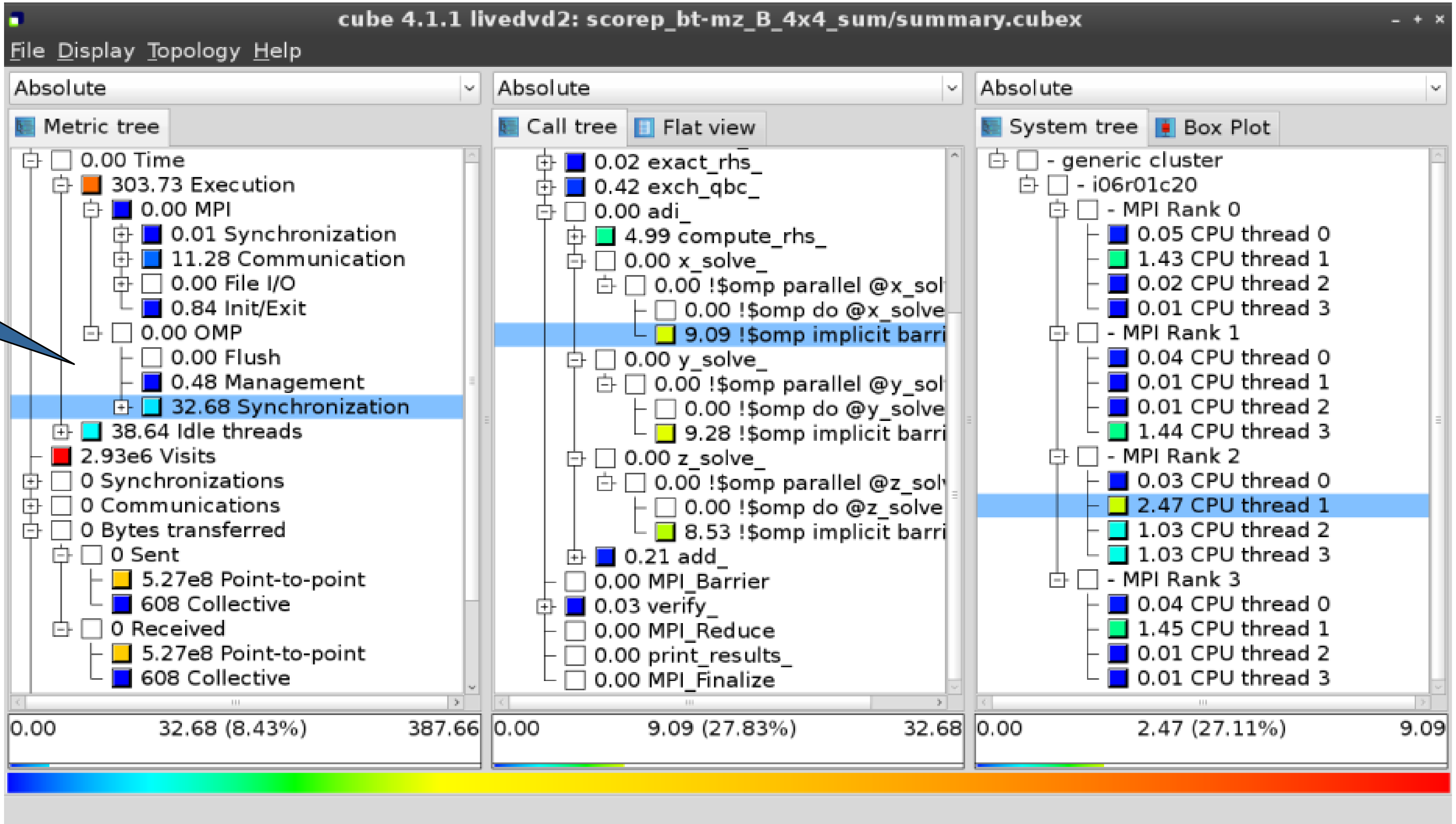
```
% square  scorep_bt-mz_C_16x8_sum
INFO: Displaying ./scorep_bt-mz_C_16x8_sum/summary.cubex...

                    [GUI showing summary analysis report]
```

Hint:
Copy ‘profile.cubex’ to local system (laptop) using ‘scp’ to improve responsiveness of GUI

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

Post-processed summary analysis report

Split base metrics into more specific metrics


cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_sum/summary.cubex
File Display Topology Help
Absolute
Metric tree
Absolute
Call tree
Flat view
Absolute
System tree
Box Plot

```
0.00 Time
  303.73 Execution
    0.00 MPI
      0.01 Synchronization
      11.28 Communication
      0.00 File I/O
      0.84 Init/Exit
    0.00 OMP
      0.00 Flush
      0.48 Management
      32.68 Synchronization
  38.64 Idle threads
2.93e6 Visits
0 Synchronizations
0 Communications
0 Bytes transferred
  0 Sent
    5.27e8 Point-to-point
    608 Collective
  0 Received
    5.27e8 Point-to-point
    608 Collective
```

```
0.02 exact_rhs_
0.42 exch_qbc_
0.00 adi_
  4.99 compute_rhs_
  0.00 x_solve_
    0.00 !$omp parallel @x_sol
      0.00 !$omp do @x_solve
      9.09 !$omp implicit barri
  0.00 y_solve_
    0.00 !$omp parallel @y_sol
      0.00 !$omp do @y_solve
      9.28 !$omp implicit barri
  0.00 z_solve_
    0.00 !$omp parallel @z_sol
      0.00 !$omp do @z_solve
      8.53 !$omp implicit barri
  0.21 add_
0.00 MPI_Barrier
0.03 verify_
0.00 MPI_Reduce
0.00 print_results_
0.00 MPI_Finalize
```

```
- generic cluster
  - i06r01c20
    - MPI Rank 0
      0.05 CPU thread 0
      1.43 CPU thread 1
      0.02 CPU thread 2
      0.01 CPU thread 3
    - MPI Rank 1
      0.04 CPU thread 0
      0.01 CPU thread 1
      0.01 CPU thread 2
      1.44 CPU thread 3
    - MPI Rank 2
      0.03 CPU thread 0
      2.47 CPU thread 1
      1.03 CPU thread 2
      1.03 CPU thread 3
    - MPI Rank 3
      0.04 CPU thread 0
      1.45 CPU thread 1
      0.01 CPU thread 2
      0.01 CPU thread 3
```

```
0.00    32.68 (8.43%)    387.66
0.00    9.09 (27.83%)    32.68
0.00    2.47 (27.11%)    9.09
```

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/hawk/scalasca.pbs .
% vim scalasca.pbs

# Score-P measurement configuration
export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC
#export SCOREP_METRIC_RUSAGE=ru_stime
export SCOREP_TOTAL_MEMORY=71M

# Scalasca measurement & analysis configuration
module load scalasca
export SCAN_ANALYZE_OPTS=“--time-correct”
export SCAN_TARGET=$EXE

# Run the application using Scalasca collection & analysis nexus
scan -t   mpirun -np $NPROCS   omplace -nt $OMP_NUM_THREADS   $EXE
```

```
% qsub -q R_tw scalasca.pbs
```

- Change to directory with the Score-P instrumented executable and edit the job script

- Add “-t” to the `scan` command
- Submit the job

BT-MZ trace measurement ... collection

```
S=C=A=N: Scalasca 2.5 trace collection and analysis
S=C=A=N: Thu Apr 11 13:35:31 2019: Collect start
mpirun -n 16  omplace -nt 8  ./bt-mz_C.16

 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:    16

 [... More application output ...]

S=C=A=N: Thu Apr 11 13:35:48 2019: Collect done (status=0) 17s
```

- Starts measurement with collection of trace files …

BT-MZ trace measurement ... analysis

```
…
S=C=A=N: Thu Apr 11 13:35:48 2019: Analyze start
 mpirun -np 16  omplace -nt 8  scout.hyb --time-correct \
>    ./scorep_bt-mz_C_16x8_trace/traces.otf2

SCOUT   (Scalasca 2.5)

Analyzing experiment archive ./scorep_bt-mz_C_16x8_trace/traces.otf2

Opening experiment archive ... done (0.022s).
Reading definition data    ... done (0.005s).
Reading event trace data   ... done (0.365s).
Preprocessing              ... done (1.621s).
Timestamp correction       ... done (3.048s).
Analyzing trace data       ... done (26.146s).
Writing analysis report    ... done (0.343s).

Max. memory usage         : 519.883MB

        # passes        : 1
        # violated      : 0

Total processing time     : 31.604s
S=C=A=N: Thu Apr 11 13:36:26 2019: Analyze done (status=0) 38s
```

- Continues with automatic (parallel) analysis of trace files

BT-MZ trace analysis report exploration

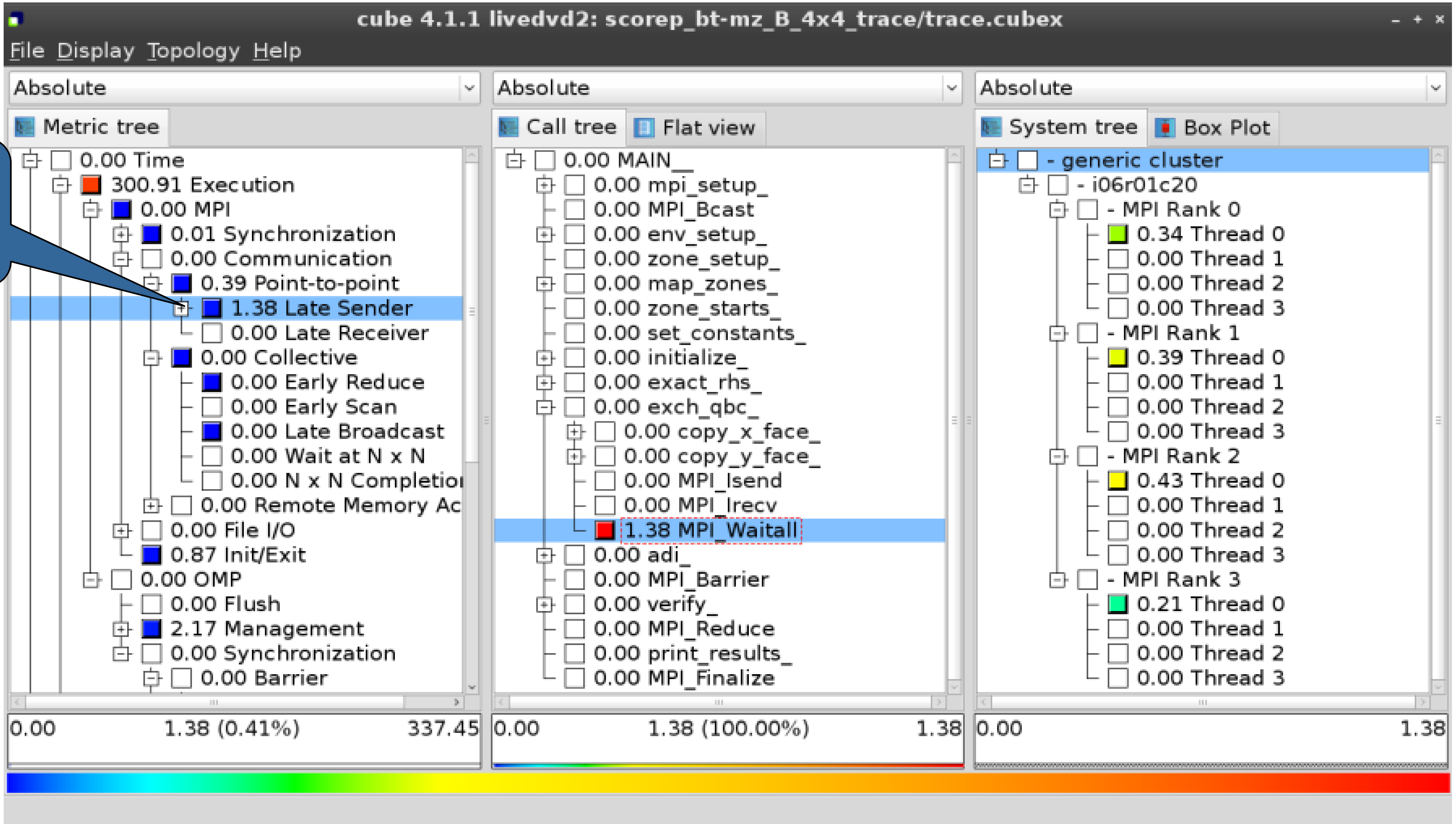
- Produces trace analysis report in the experiment directory containing trace-based wait-state metrics

```
% square  scorep_bt-mz_C_16x8_trace
INFO: Post-processing runtime summarization result...
INFO: Post-processing trace analysis report...
INFO: Displaying ./scorep_bt-mz_C_16x8_trace/trace.cubex...

                [GUI showing trace analysis report]
```

Hint:
Run ‘square -s’ first and then copy ‘trace.cubex’ to local system (laptop) using ‘scp’ to improve responsiveness of GUI

Post-processed trace analysis report

Additional trace-based metrics in metric hierarchy



cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex

```
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 Late Broadcast
          0.00 Wait at N x N
          0.00 N x N Completion
        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
        0.00 Barrier
```

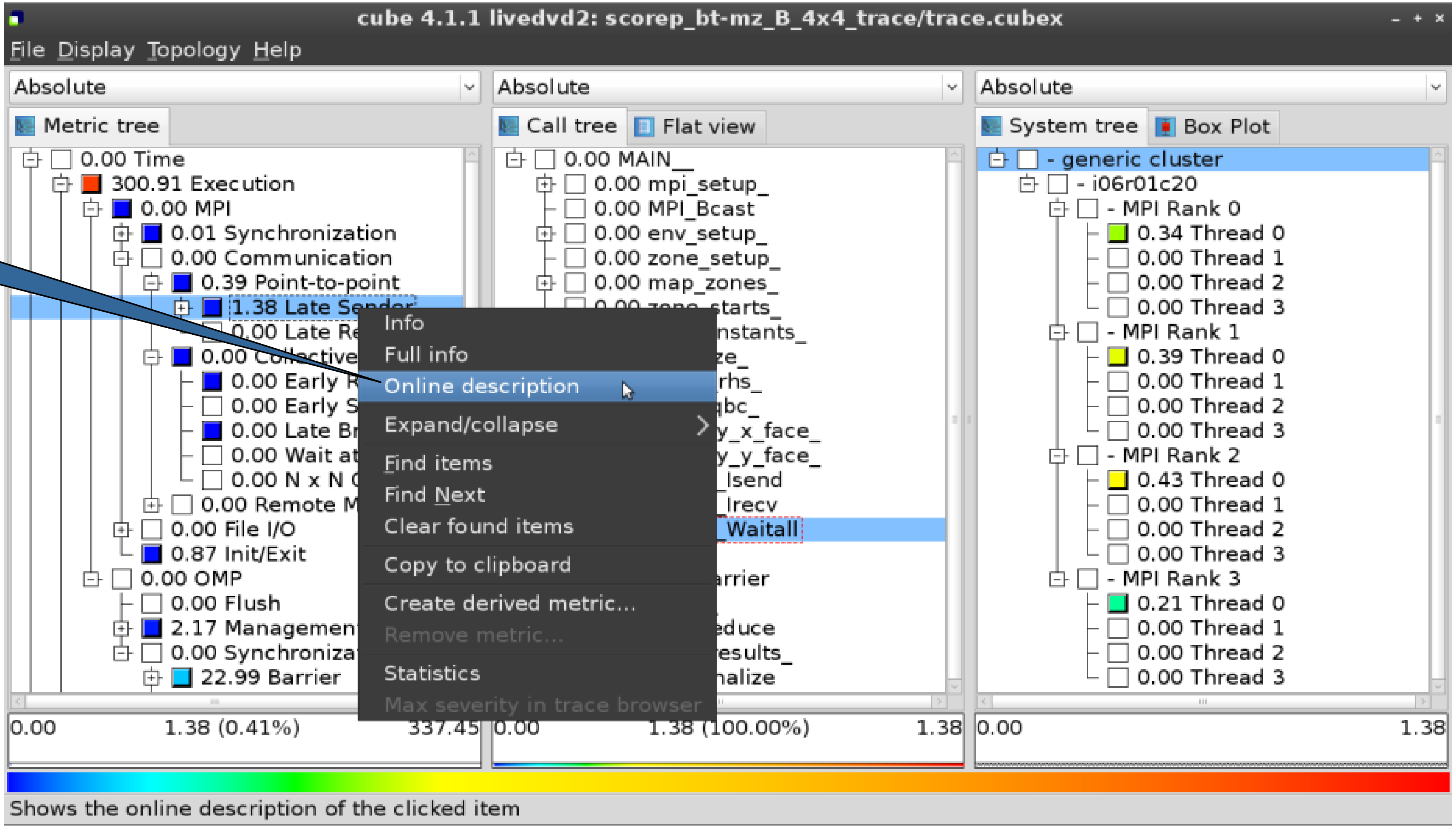
```
0.00 MAIN__
  0.00 mpi_setup_
  0.00 MPI_Bcast
  0.00 env_setup_
  0.00 zone_setup_
  0.00 map_zones_
  0.00 zone_starts_
  0.00 set_constants_
  0.00 initialize_
  0.00 exact_rhs_
  0.00 exch_qbc_
    0.00 copy_x_face_
    0.00 copy_y_face_
    0.00 MPI_Isend
    0.00 MPI_Irecv
    1.38 MPI_Waitall
  0.00 adi_
  0.00 MPI_Barrier
  0.00 verify_
  0.00 MPI_Reduce
  0.00 print_results_
  0.00 MPI_Finalize
```

```
- 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
    - MPI Rank 2
      0.43 Thread 0
      0.00 Thread 1
      0.00 Thread 2
      0.00 Thread 3
    - MPI Rank 3
      0.21 Thread 0
      0.00 Thread 1
      0.00 Thread 2
      0.00 Thread 3
```

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

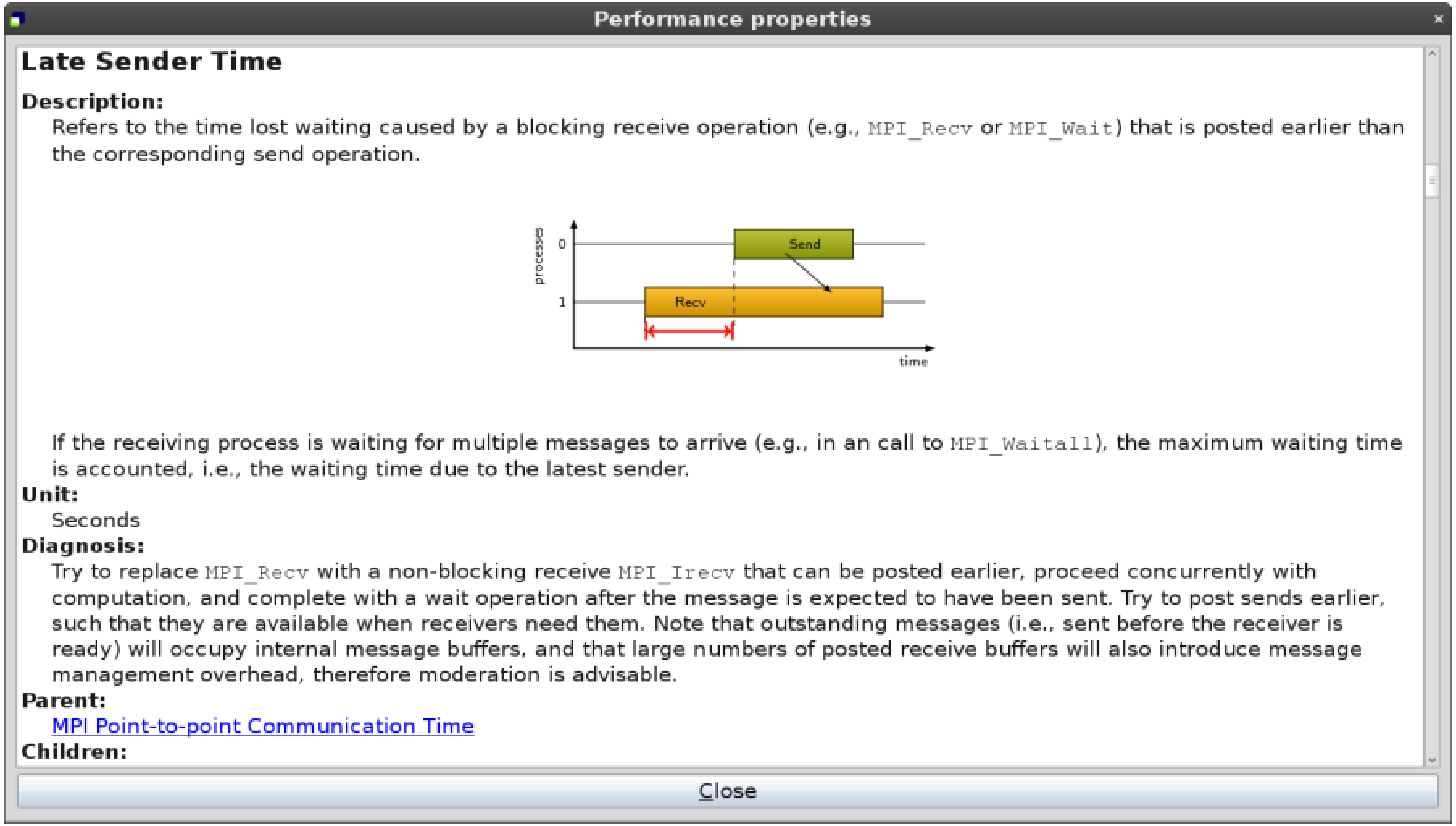
Online metric description

Access online metric description via context menu



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.

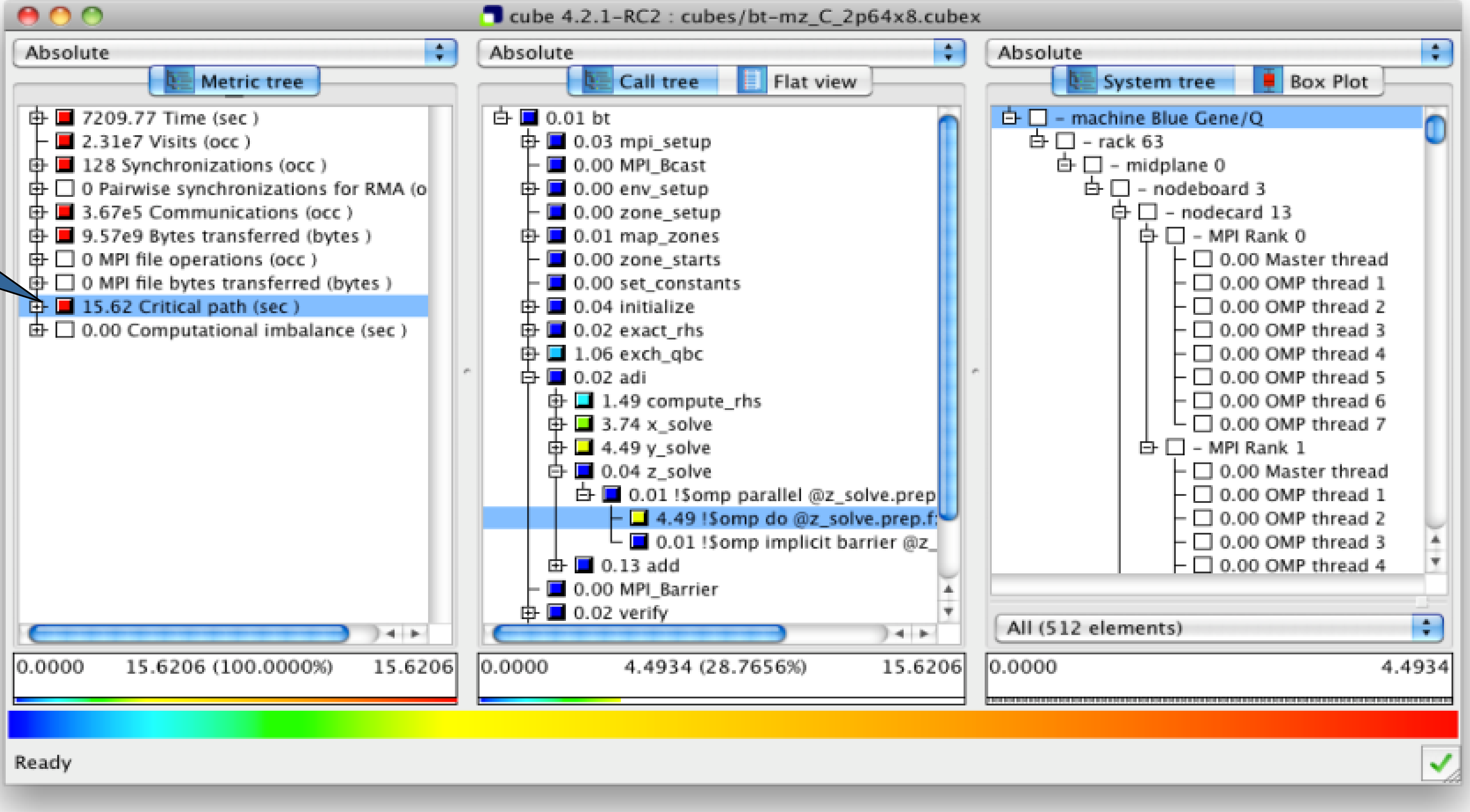
Parent:

MPI Point-to-point Communication Time

Children:

Critical-path analysis

Critical-path profile shows wall-clock time impact



cube 4.2.1-RC2 : cubes/bt-mz_C_2p64x8.cubex

Absolute — Metric tree

```
⊞ 7209.77 Time (sec )
  2.31e7 Visits (occ )
⊞ 128 Synchronizations (occ )
⊞ 0 Pairwise synchronizations for RMA (o
⊞ 3.67e5 Communications (occ )
⊞ 9.57e9 Bytes transferred (bytes )
⊞ 0 MPI file operations (occ )
⊞ 0 MPI file bytes transferred (bytes )
⊞ 15.62 Critical path (sec )
⊞ 0.00 Computational imbalance (sec )
```

0.0000 15.6206 (100.0000%) 15.6206

Absolute — Call tree | Flat view

```
⊟ 0.01 bt
   ⊞ 0.03 mpi_setup
     0.00 MPI_Bcast
   ⊞ 0.00 env_setup
     0.00 zone_setup
   ⊞ 0.01 map_zones
     0.00 zone_starts
     0.00 set_constants
   ⊞ 0.04 initialize
   ⊞ 0.02 exact_rhs
   ⊞ 1.06 exch_qbc
   ⊟ 0.02 adi
      ⊞ 1.49 compute_rhs
      ⊞ 3.74 x_solve
      ⊞ 4.49 y_solve
      ⊟ 0.04 z_solve
         ⊟ 0.01 !$omp parallel @z_solve.prep
            4.49 !$omp do @z_solve.prep.f
            0.01 !$omp implicit barrier @z_
      ⊞ 0.13 add
     0.00 MPI_Barrier
   ⊞ 0.02 verify
```

0.0000 4.4934 (28.7656%) 15.6206

Absolute — System tree | Box Plot

```
⊟ - machine Blue Gene/Q
   ⊟ - rack 63
      ⊟ - midplane 0
         ⊟ - nodeboard 3
            ⊟ - nodecard 13
               ⊟ - MPI Rank 0
                  0.00 Master thread
                  0.00 OMP thread 1
                  0.00 OMP thread 2
                  0.00 OMP thread 3
                  0.00 OMP thread 4
                  0.00 OMP thread 5
                  0.00 OMP thread 6
                  0.00 OMP thread 7
               ⊟ - MPI Rank 1
                  0.00 Master thread
                  0.00 OMP thread 1
                  0.00 OMP thread 2
                  0.00 OMP thread 3
                  0.00 OMP thread 4
```

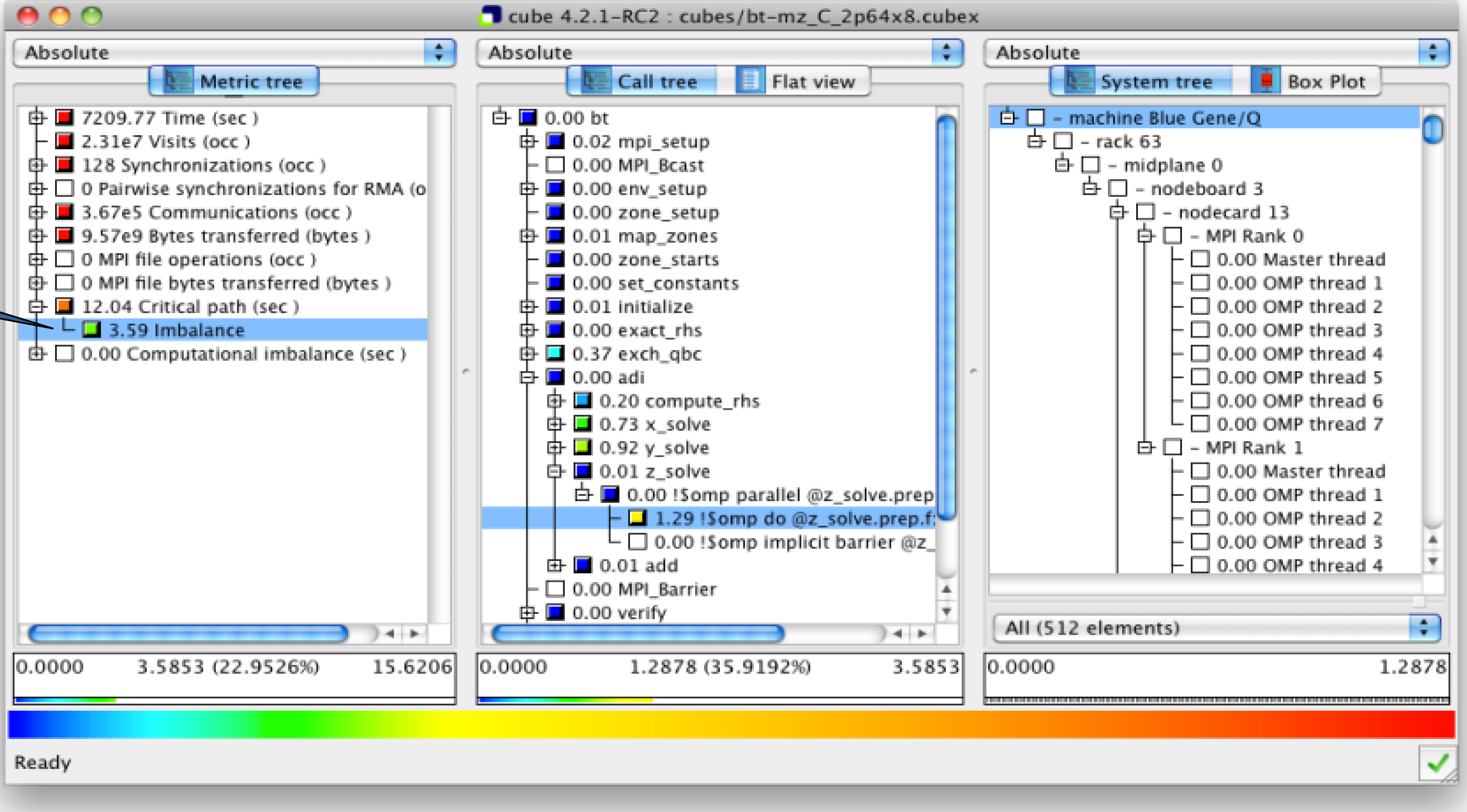
All (512 elements)

0.0000 4.4934

Ready

Critical-path analysis

Critical-path imbalance highlights inefficient parallelism



cube 4.2.1-RC2 : cubes/bt-mz_C_2p64x8.cubex

Absolute | Metric tree

```
7209.77 Time (sec )
2.31e7 Visits (occ )
128 Synchronizations (occ )
0 Pairwise synchronizations for RMA (o
3.67e5 Communications (occ )
9.57e9 Bytes transferred (bytes )
0 MPI file operations (occ )
0 MPI file bytes transferred (bytes )
12.04 Critical path (sec )
  3.59 Imbalance
0.00 Computational imbalance (sec )
```

0.0000   3.5853 (22.9526%)   15.6206

Absolute | Call tree | Flat view

```
0.00 bt
  0.02 mpi_setup
  0.00 MPI_Bcast
  0.00 env_setup
  0.00 zone_setup
  0.01 map_zones
  0.00 zone_starts
  0.00 set_constants
  0.01 initialize
  0.00 exact_rhs
  0.37 exch_qbc
  0.00 adi
    0.20 compute_rhs
    0.73 x_solve
    0.92 y_solve
    0.01 z_solve
      0.00 !$omp parallel @z_solve.prep
        1.29 !$omp do @z_solve.prep.f
        0.00 !$omp implicit barrier @z_
    0.01 add
  0.00 MPI_Barrier
  0.00 verify
```

0.0000   1.2878 (35.9192%)   3.5853

Absolute | System tree | Box Plot

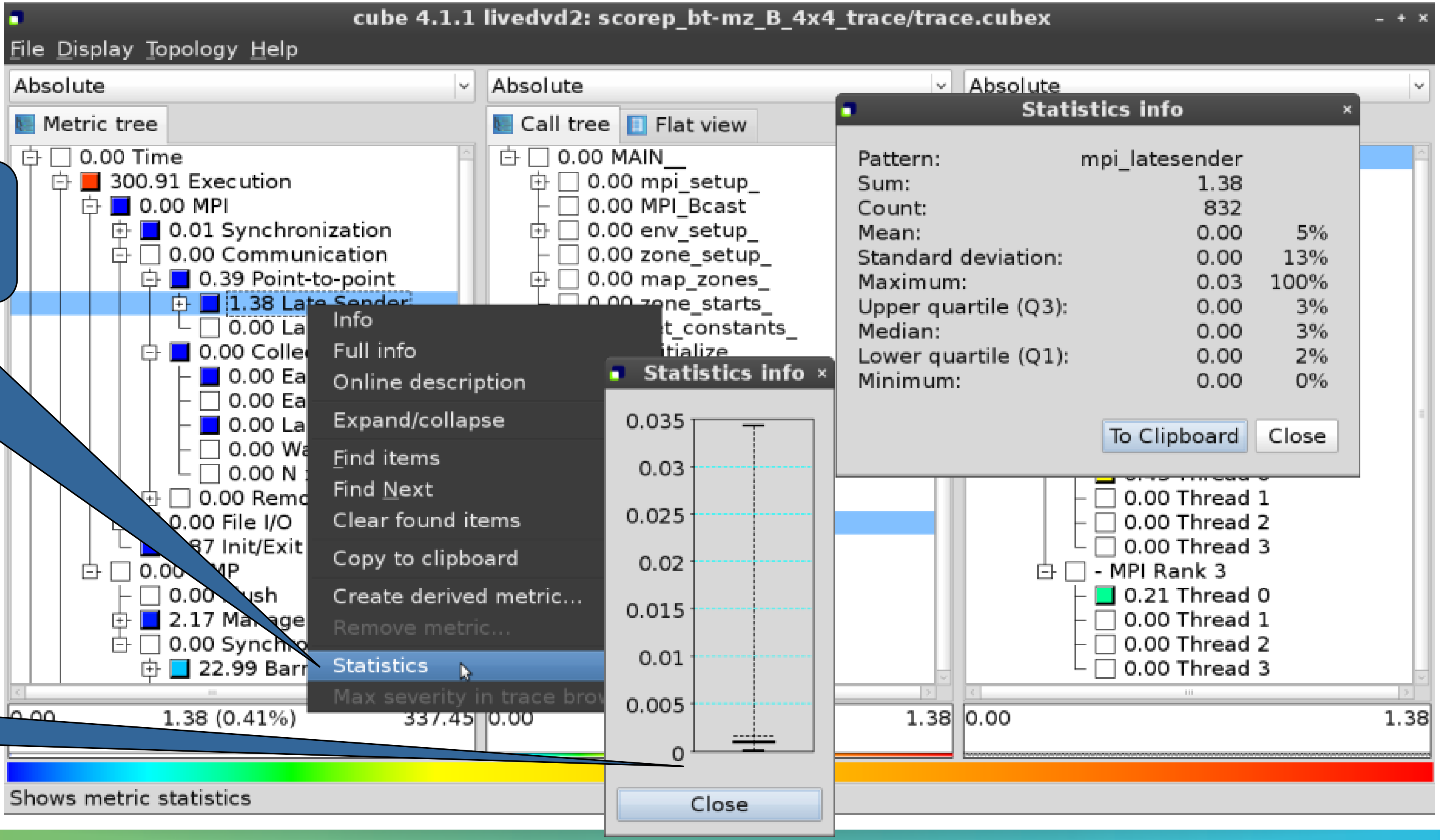
```
- machine Blue Gene/Q
  - rack 63
    - midplane 0
      - nodeboard 3
        - nodecard 13
          - MPI Rank 0
            0.00 Master thread
            0.00 OMP thread 1
            0.00 OMP thread 2
            0.00 OMP thread 3
            0.00 OMP thread 4
            0.00 OMP thread 5
            0.00 OMP thread 6
            0.00 OMP thread 7
          - MPI Rank 1
            0.00 Master thread
            0.00 OMP thread 1
            0.00 OMP thread 2
            0.00 OMP thread 3
            0.00 OMP thread 4
```

All (512 elements)

0.0000   1.2878

Ready

Pattern instance statistics



Statistics info

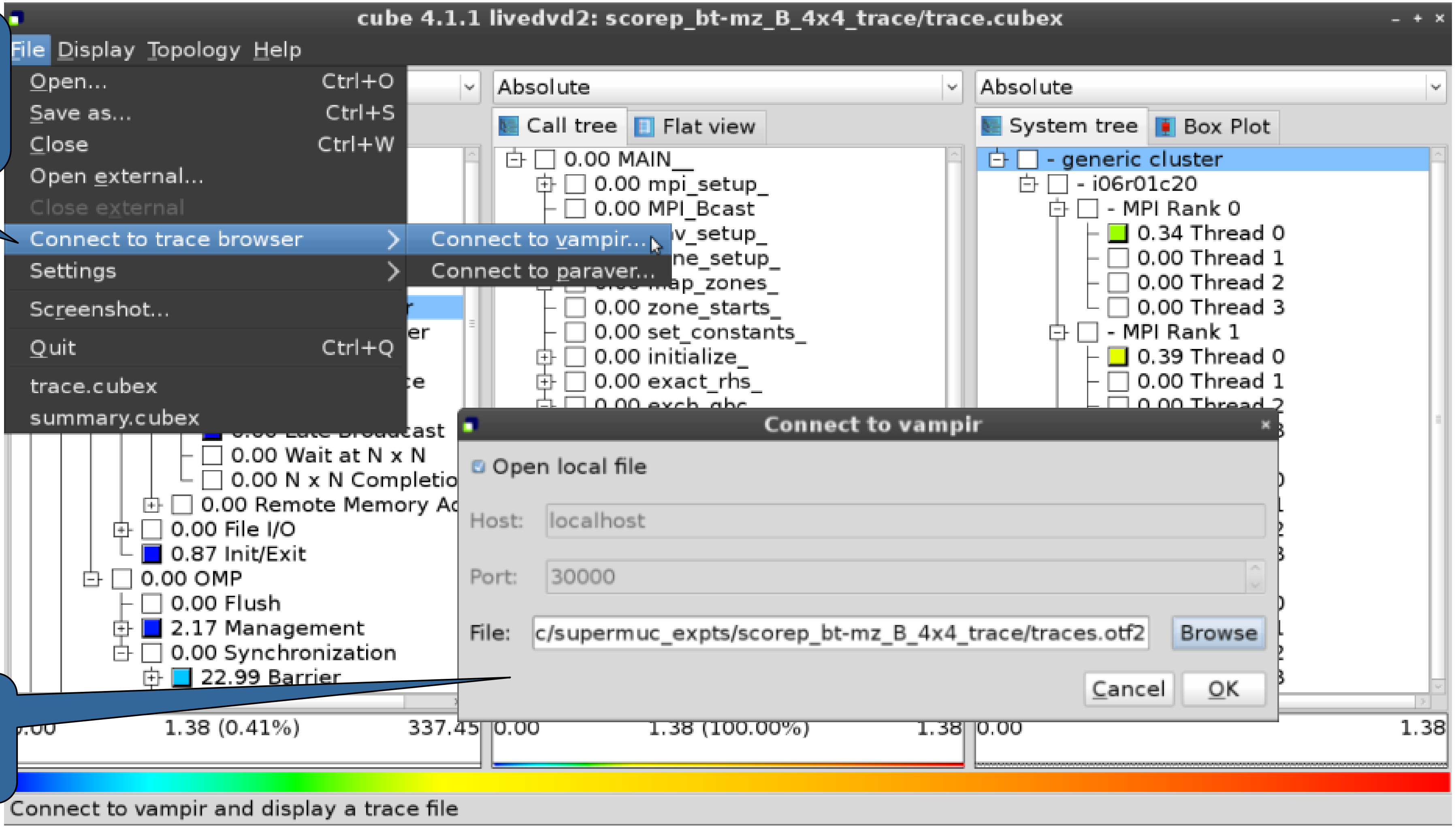
Pattern:	mpi_latesender	
Sum:	1.38	
Count:	832	
Mean:	0.00	5%
Standard deviation:	0.00	13%
Maximum:	0.03	100%
Upper quartile (Q3):	0.00	3%
Median:	0.00	3%
Lower quartile (Q1):	0.00	2%
Minimum:	0.00	0%

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…



cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex

0.00 MAIN__
0.00 mpi_setup_
0.00 MPI_Bcast
0.00 zone_starts_
0.00 set_constants_
0.00 initialize_
0.00 exact_rhs_

- 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 Wait at N x N
0.00 N x N Completio
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

Connect to vampir

☑ Open local file

Host: localhost

Port: 30000

File: c/supermuc_expts/scorep_bt-mz_B_4x4_trace/traces.otf2

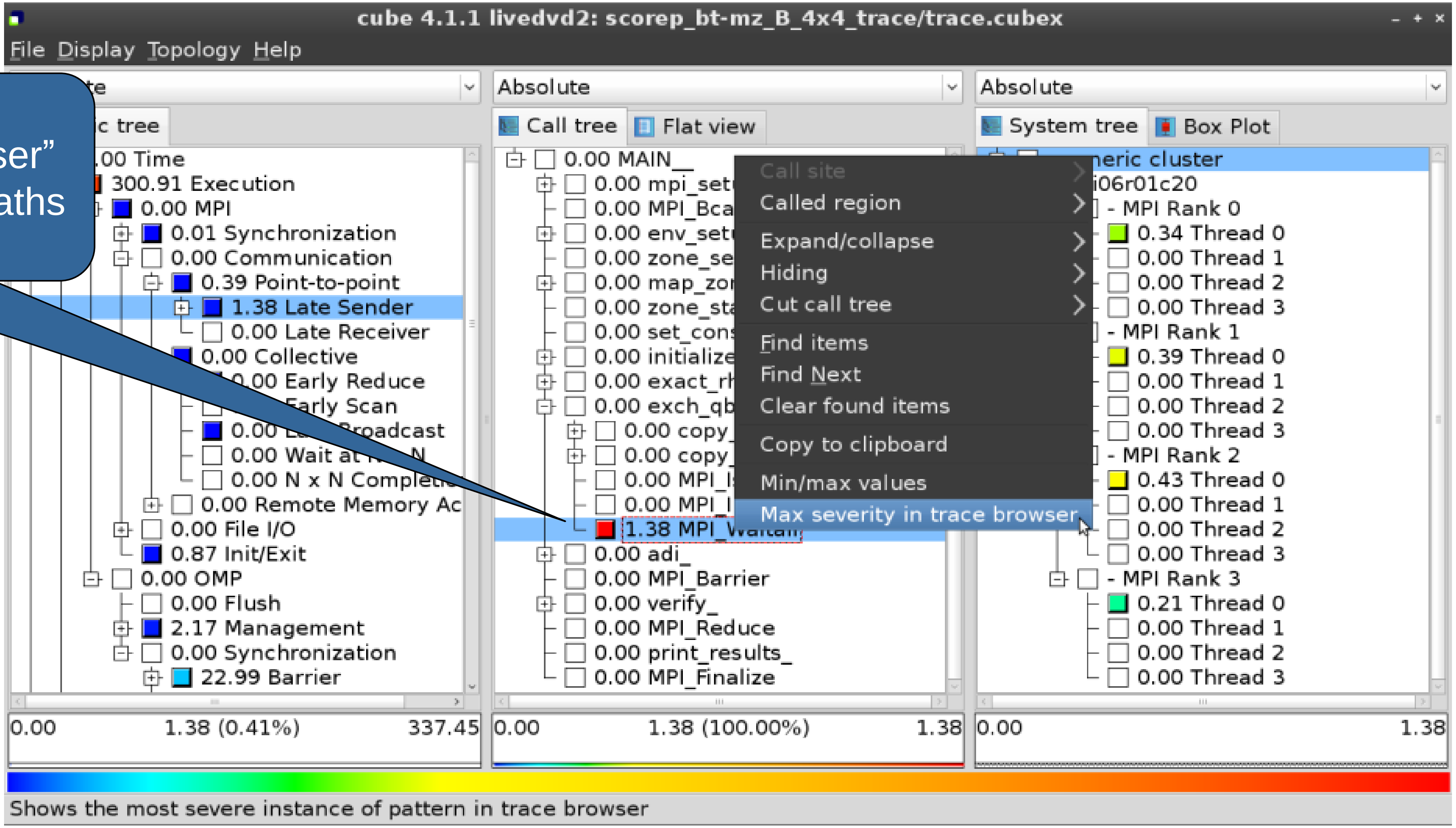
0.00 1.38 (0.41%) 337.45 0.00 1.38 (100.00%) 1.38 0.00 1.38

Connect to vampir and display a trace file

…and select trace file from the experiment directory

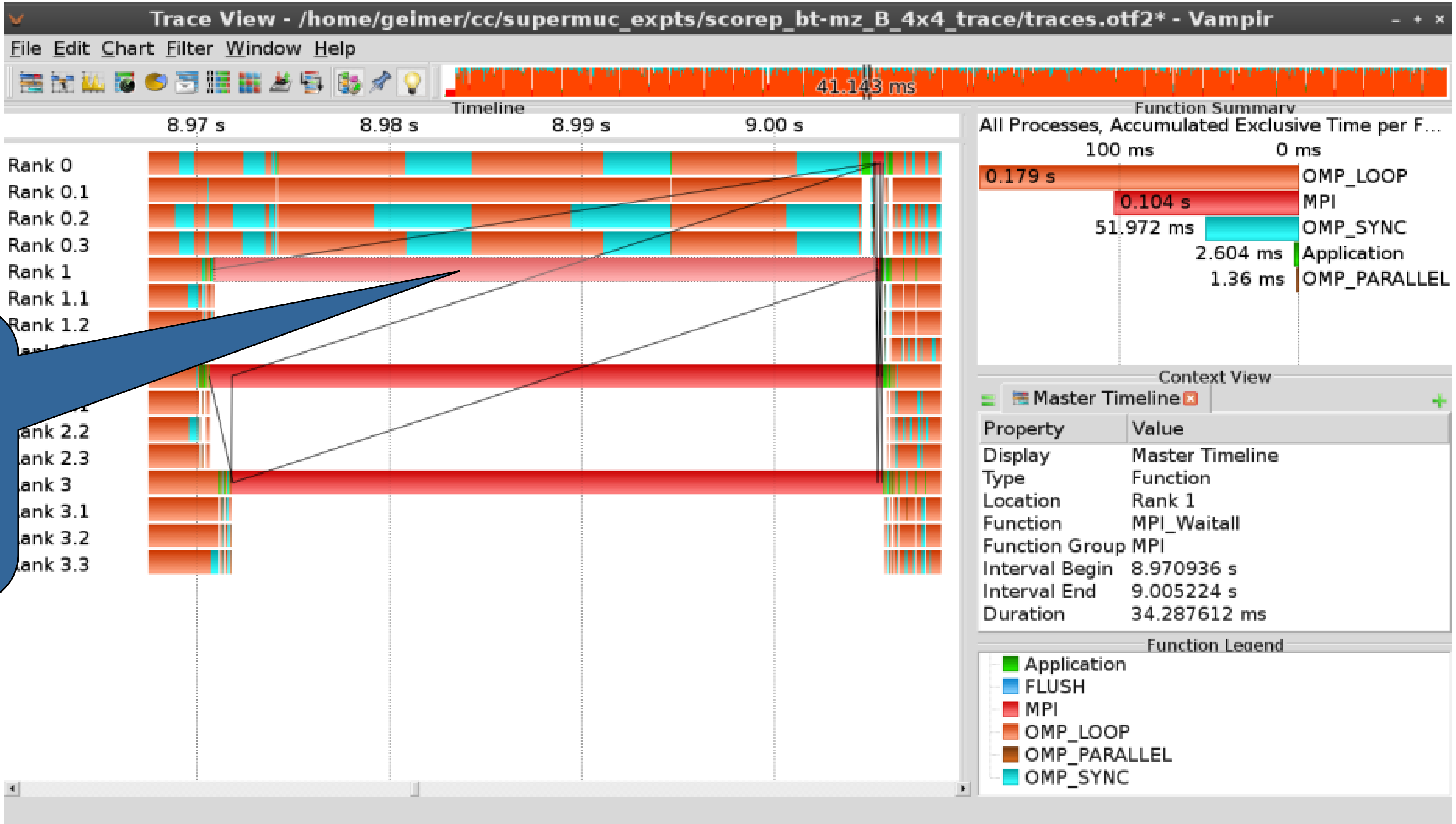
Show most severe pattern instances

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



Shows the most severe instance of pattern in trace browser

Investigate most severe instance in Vampir



Property	Value
Display	Master Timeline
Type	Function
Location	Rank 1
Function	MPI_Waitall
Function Group	MPI
Interval Begin	8.970936 s
Interval End	9.005224 s
Duration	34.287612 ms

Function Legend

- 🟩 Application
- 🟦 FLUSH
- 🟥 MPI
- 🟧 OMP_LOOP
- 🟫 OMP_PARALLEL
- 🟦 OMP_SYNC

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