

VI-HPS

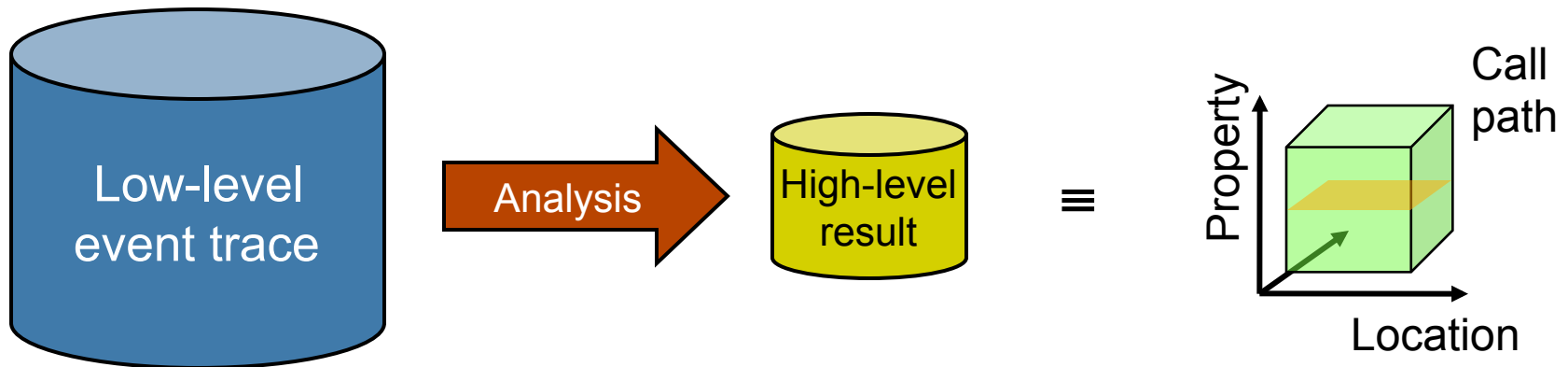


Automatic trace analysis with Scalasca

Markus Geimer, Brian Wylie, David Böhme
Jülich Supercomputing Centre

scalasca 

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



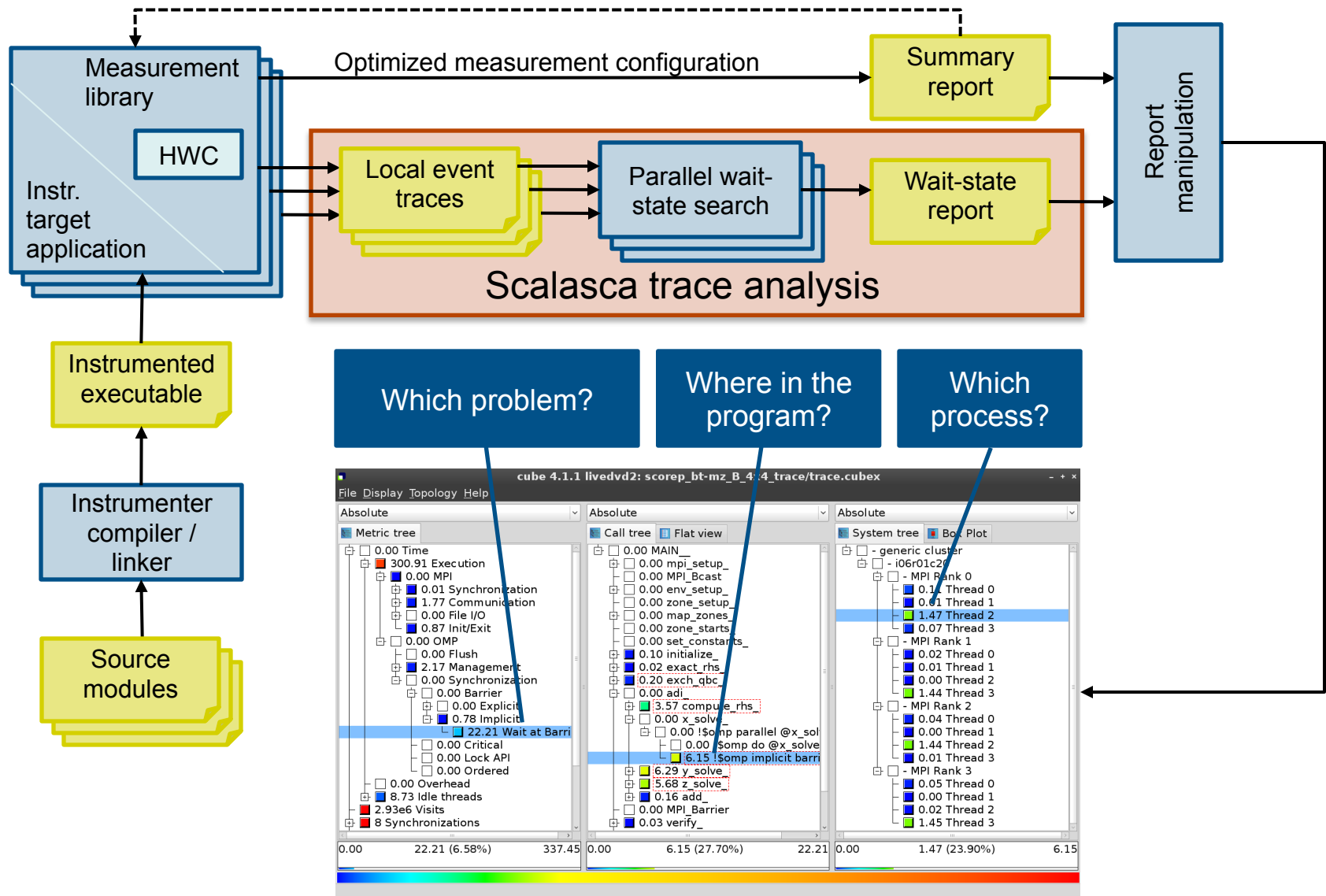
- Guaranteed to cover the entire event trace
- Quicker than manual/visual trace analysis
- Parallel replay analysis exploits available memory & processors to deliver scalability

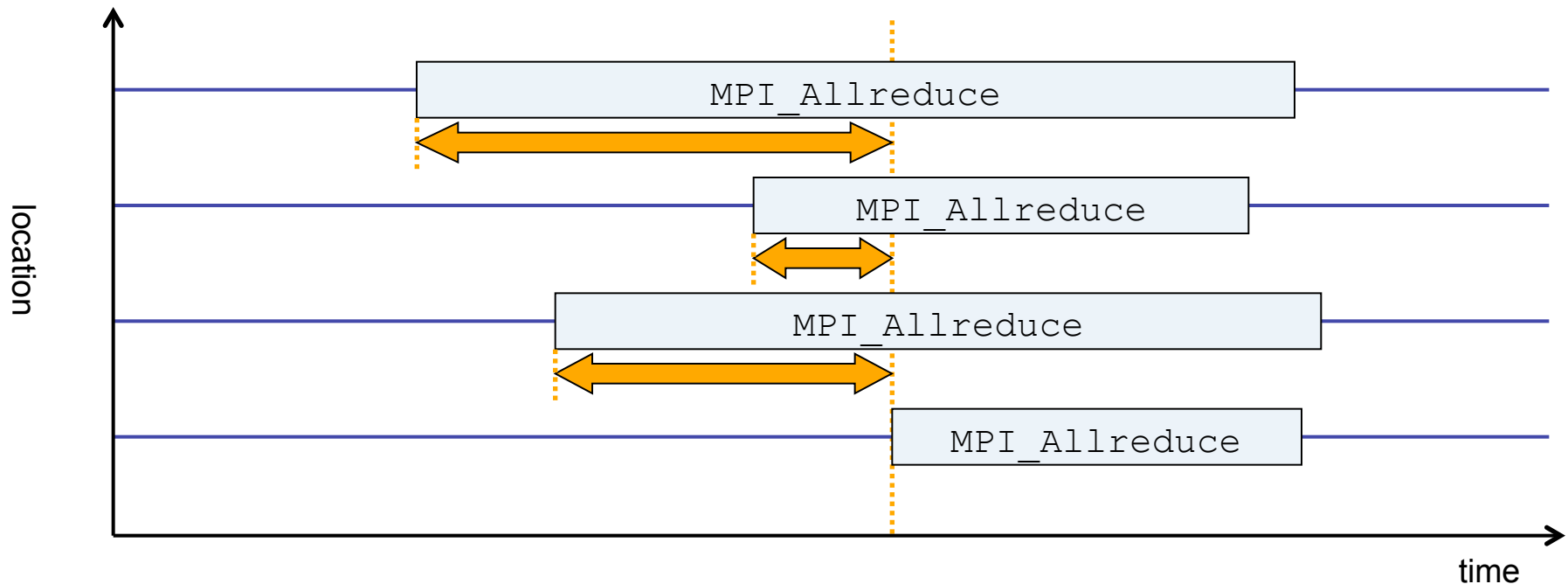
- Project started in 2006
 - Initial funding by Helmholtz Initiative & Networking Fund
 - Many follow-up projects
- Follow-up to pioneering KOJAK project (started 1998)
 - Automatic pattern-based trace analysis
- Now joint development of
 - Jülich Supercomputing Centre
 - German Research School for Simulation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



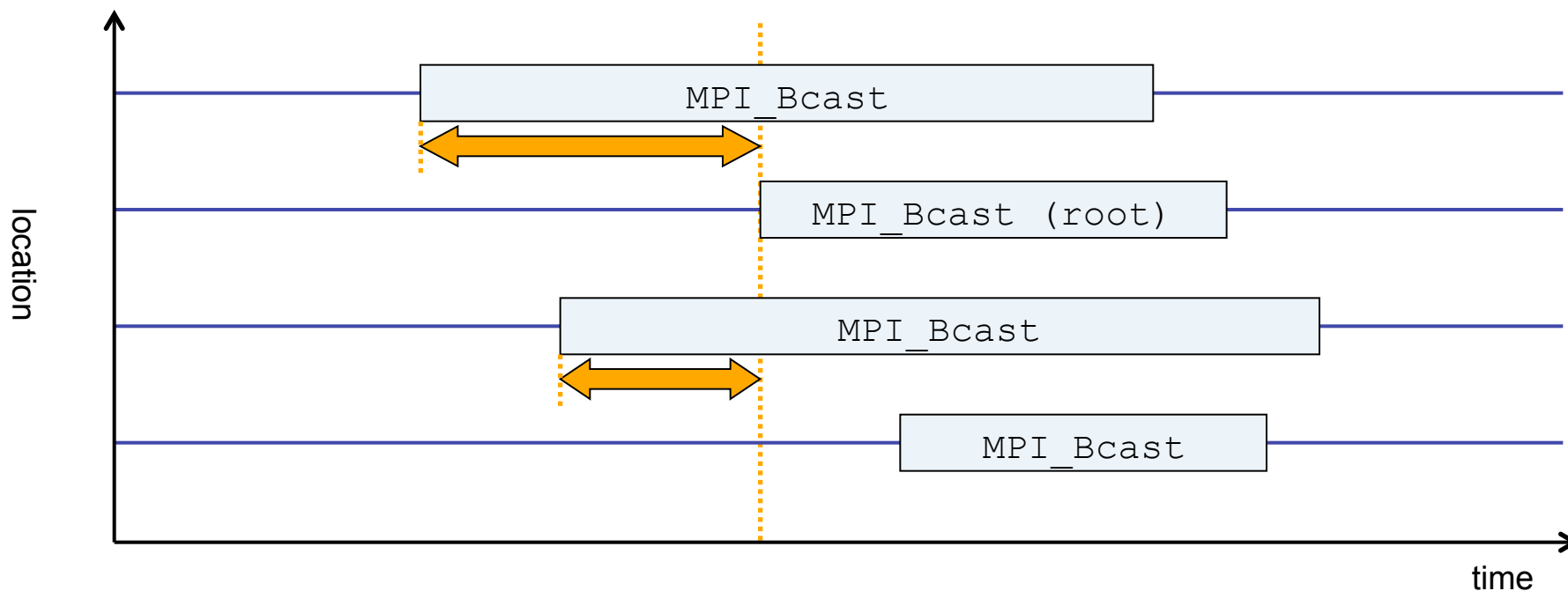
- Development of a **scalable** performance analysis toolset for most popular parallel programming paradigms
- Specifically targeting **large-scale** parallel applications
 - such as those running on IBM BlueGene or Cray XT systems with one million or more processes/threads
- Latest release:
 - Scalasca v2.0 with Score-P support (August 2013)
 - Scalasca v2.1 coordinated with Score-P v1.3 (planned 2014Q1)
 - initial support for Fujitsu FX10 & K computer
 - improved support for Scalasca v1.x (EPIK/ELG) trace analysis
 - additional trace analysis of critical path
 - robustness and performance improvements

- Open source, New BSD license
- Fairly portable
 - IBM Blue Gene, IBM SP & blade clusters, Cray XT, SGI Altix, Fujitsu FX10 & K computer, Solaris & Linux clusters, ...
- Uses Score-P instrumenter & measurement libraries
 - Scalasca 2.0 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 Pthreads, or OpenMP nested parallelism and tasking
 - PAPI/rusage metrics for trace events are ignored

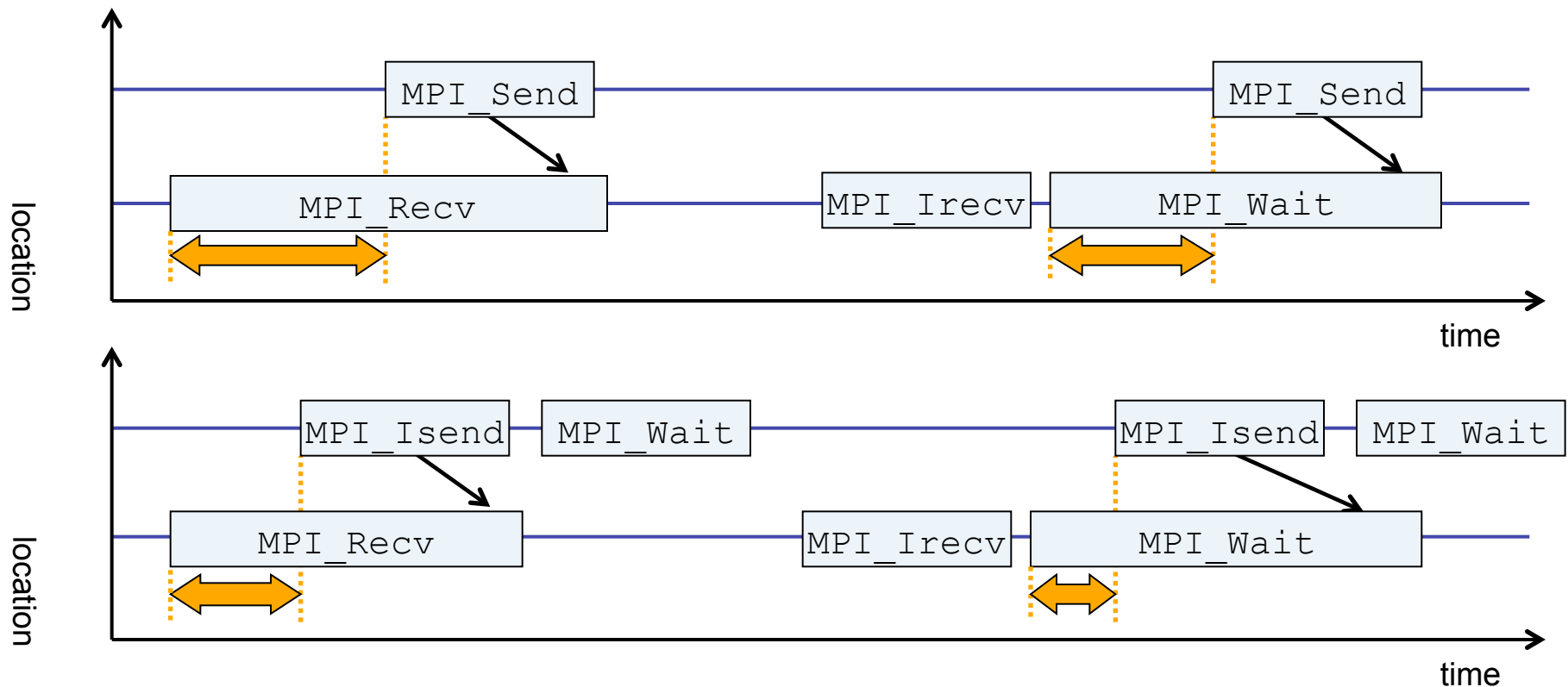




- Time spent waiting in front of synchronizing collective operation until the last process reaches the operation
- Applies to: MPI_Allgather, MPI_Allgatherv, MPI_Alltoall, MPI_Reduce_scatter, MPI_Reduce_scatter_block, MPI_Allreduce



- 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



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

VI-HPS



Hands-on: NPB-MZ-MPI / BT

scalasca

- One command for (almost) everything...

```
% scalasca
Scalasca 2.1-alpha3
Toolset for scalable performance analysis of large-scale applications
usage: scalasca [-v][-n][c] {action}
    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

-v, --verbose          enable verbose commentary
-n, --dry-run          show actions without taking them
-c, --show-config      show configuration 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 application instrumenter

```
% skin  
Scalasca 2.1-alpha3: 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
```

- Provides compatibility with Scalasca 1.x
- Recommended: use Score-P instrumenter directly

- Scalasca measurement collection & analysis nexus

```
% scan
Scalasca 2.1-alpha3: 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 filtfiler : File specifying measurement filter.
    -l lockfile  : File that blocks start of measurement.
    -m metrics   : Metric specification for measurement.
```

- 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 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 analysis report explorer

```
% square
Scalasca 2.1-alpha3: analysis report explorer
usage: square [-v] [-s] [-f filtfiler] [-F] <experiment archive
              | cube file>
-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
```


- **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)

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

```
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_W_4x4_sum
% OMP_NUM_THREADS=4 scan mpiexec -np 4 ./bt-mz_W.4
S=C=A=N: Scalasca 2.1-alpha3 runtime summarization
S=C=A=N: ./scorep_bt-mz_W_4x4_sum experiment archive
S=C=A=N: Thu Sep 13 18:05:17 2012: 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 Sep 13 18:05:39 2012: 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`

- Score summary analysis report

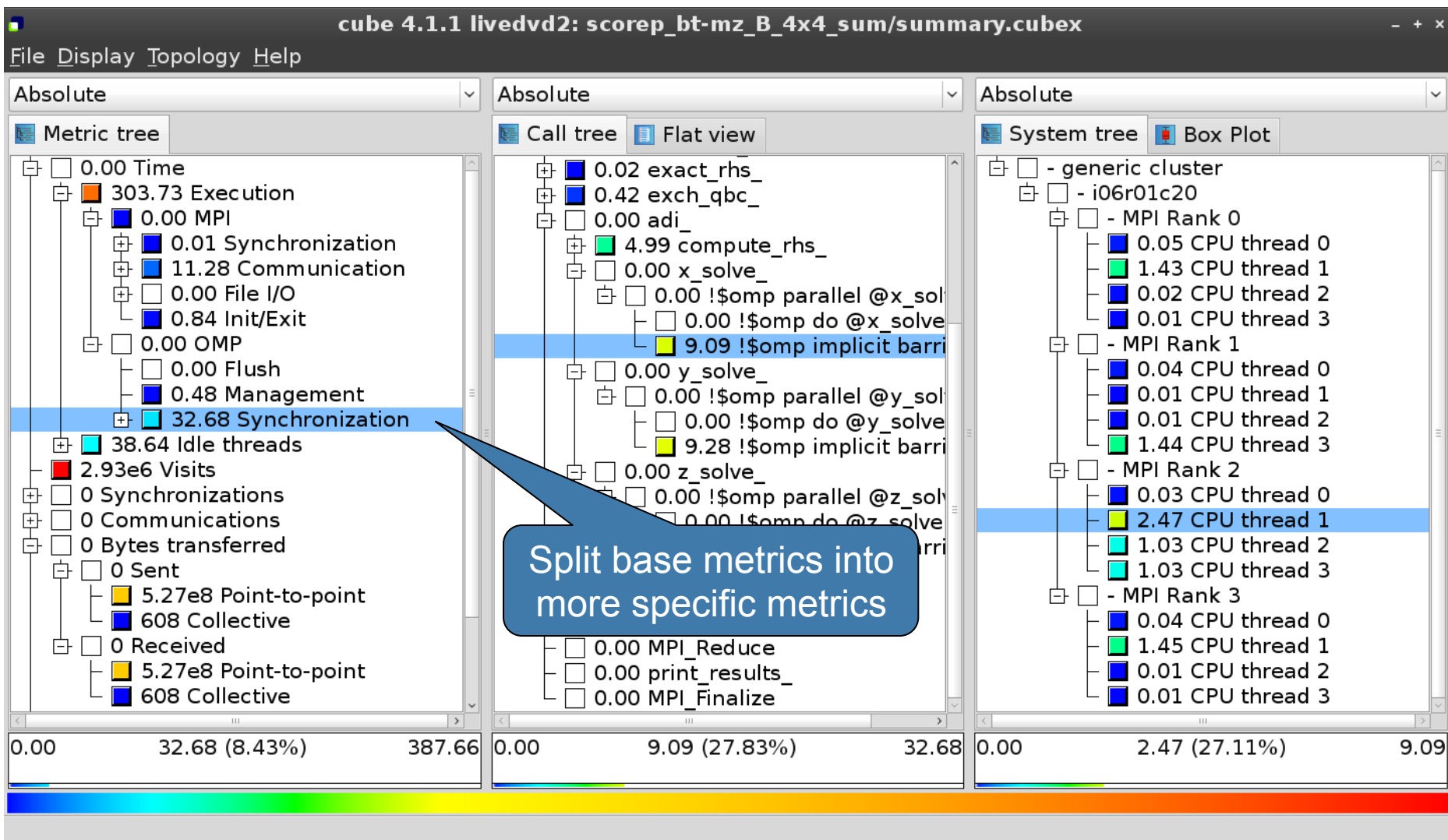
```
% square -s scorep_bt-mz_W_4x4_sum  
INFO: Post-processing runtime summarization result...  
INFO: Score report written to ./scorep_bt-mz_W_4x4_sum/scorep.score
```

- 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



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

- Load modules

```
% module load UNITE
UNITE loaded
% module load scorep/1.2.3
scorep/1.2.3 loaded
% module load scalasca/2.1a3-intel13-impi
scalasca/2.1a3-intel13-impi loaded
% module load cube/4.2.3rc1-intel
cube/4.2.3rc1 loaded
```

- Change to directory containing NPB BT-MZ sources
- Existing instrumented binary in bin.scorep/ can be reused

- Change to executable directory and edit job script

```
% cd bin.scorep
% cp ../jobscript/mds/run_scan_tracing.ll .
% vim run_scan_tracing.ll

[...]
```

```
module load UNITE scalasca/2.1-alpha3

export SCOREP_FILTERING_FILE=../config/scorep.filt
export SCOREP_TOTAL_MEMORY=50M
#export SCOREP_METRIC_PAPI=PAPI_FP_OPS

scan -t mpirun -np ${LOADL_TOTAL_TASKS} ./bt-mz_B.4
```

- Submit the job

```
% llsubmit run_scan_tracing.ll
```

- Continues with automatic (parallel) analysis of trace files

```
S=C=A=N: Mon Apr 7 11:07:39 2014: Analyze start
[...]
Analyzing experiment archive ./scorep_bt-mz_B_4x4_trace/traces.otf2

Opening experiment archive ... done (0.006s).
Reading definition data ... done (0.010s).
Reading event trace data ... done (0.285s).
Preprocessing ... done (0.823s).
Analyzing trace data ...
  Wait-state detection (fwd) (1/5) ... done (0.682s).
  Wait-state detection (bwd) (2/5) ... done (0.045s).
  Synchpoint exchange (3/5) ... done (0.161s).
  Critical-path & delay analysis (4/5) ... done (0.201s).
done (1.182s).
Writing analysis report ... done (0.078s).

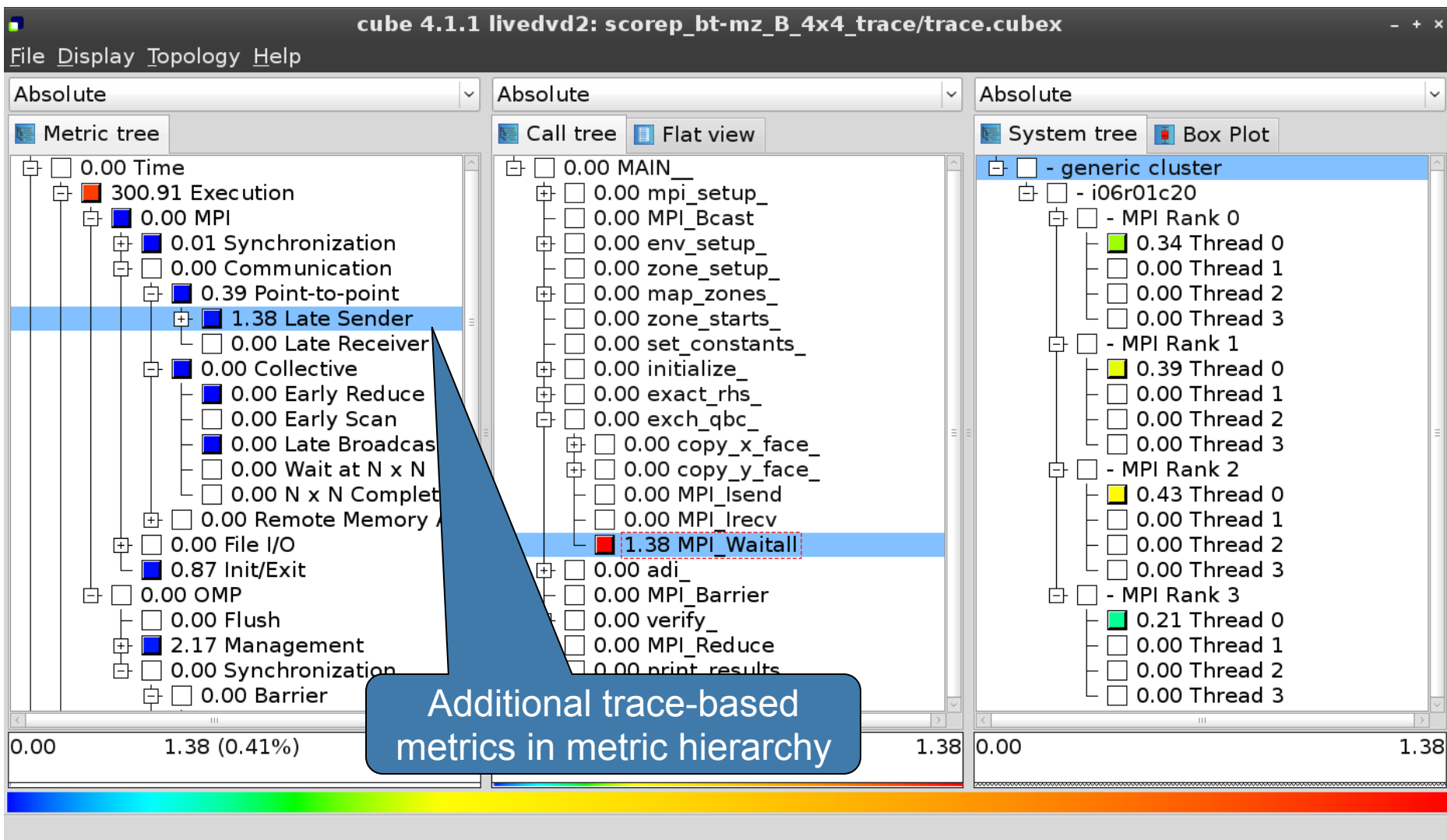
Max. memory usage : 227.945MB

Total processing time : 2.963s
S=C=A=N: Mon Apr 7 11:07:44 2014: Analyze done (status=0) 5s
```


- 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



The screenshot displays the 'cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex' application interface. It features three main panels: 'Metric tree', 'Call tree', and 'System tree', all set to 'Absolute' view. The 'Metric tree' panel on the left shows a hierarchical list of metrics, with '1.38 Late Sender' 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...', 'Statistics', and 'Max severity in trace browser'. The 'Online description' option is highlighted. The 'Call tree' panel in the middle shows a list of function calls, and the 'System tree' panel on the right shows a hierarchy of system components and threads. A blue callout box with a pointer indicates that the 'Online description' can be accessed via the context menu.

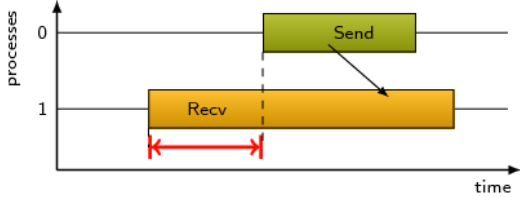
Access online metric description via context menu



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.

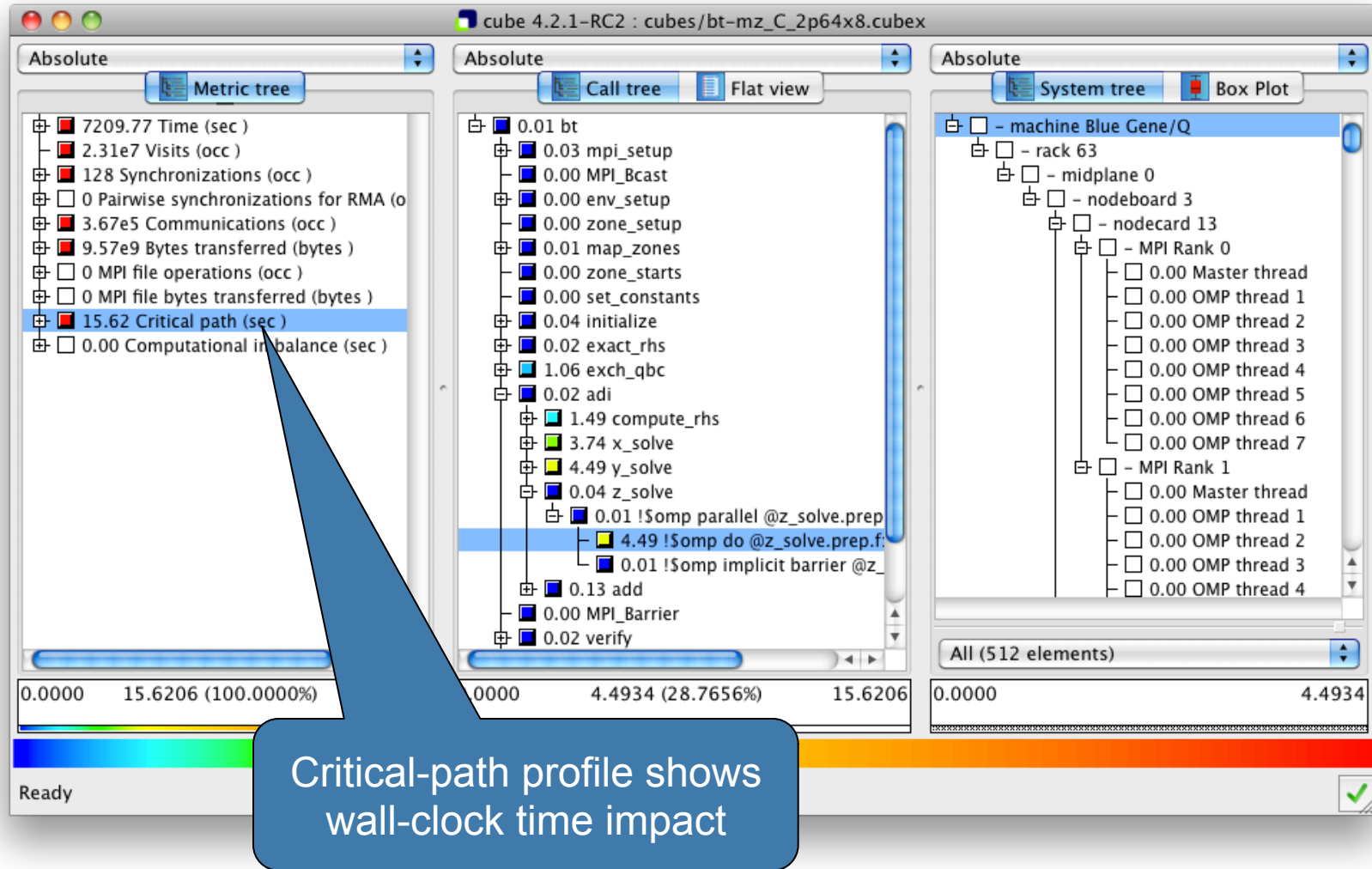
Unit:
Seconds

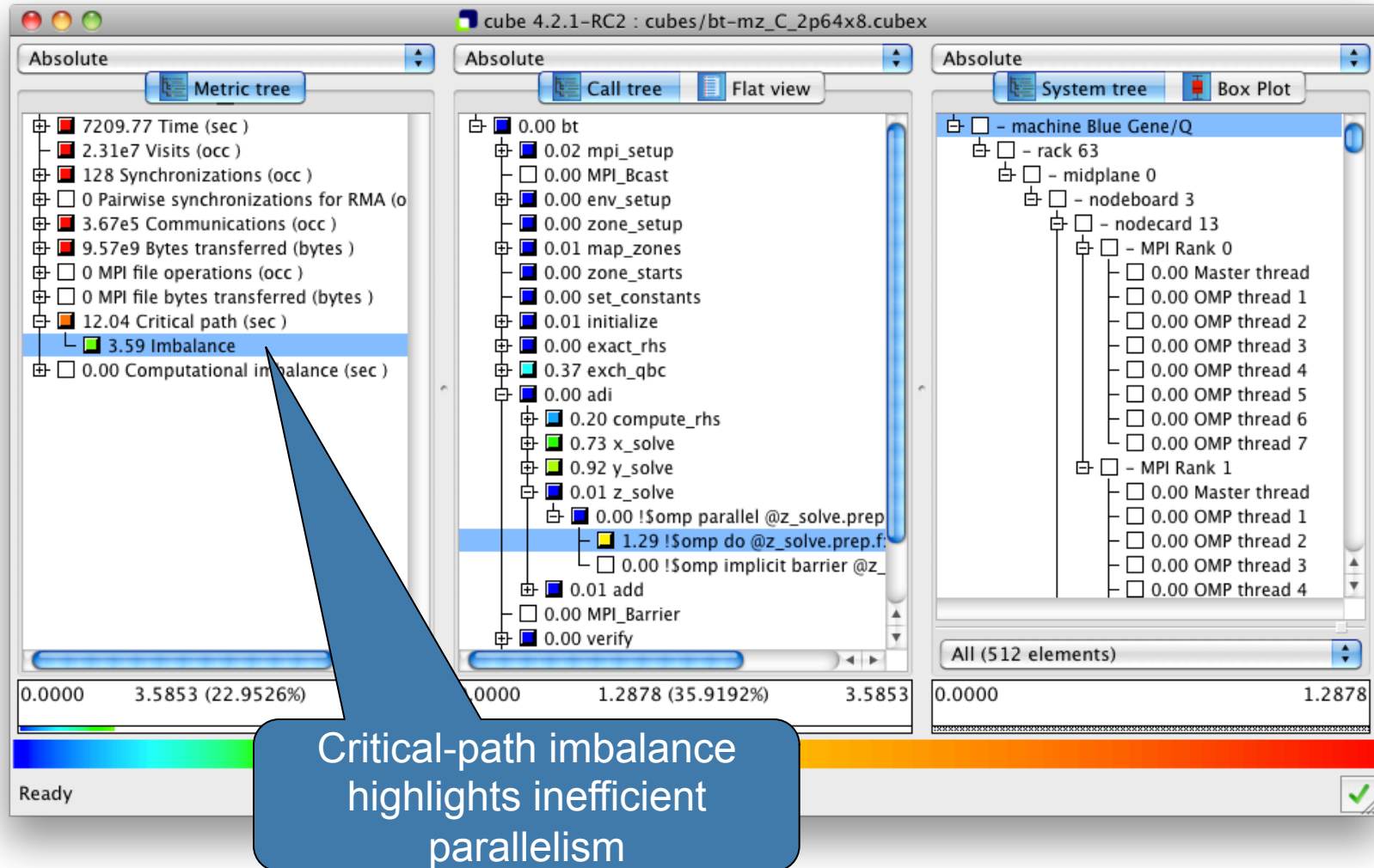
Diagnosis:
Try to replace `MPI_Recv` with a non-blocking receive `MPI_Irecv` that can be posted earlier, proceed concurrently with computation, and complete with a wait operation after the message is expected to have been sent. Try to post sends earlier, such that they are available when receivers need 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





The screenshot displays the VI-HPS interface with the file `cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex` open. The 'Metric tree' on the left shows a hierarchy of metrics, with '1.38 Late Sender' selected. A context menu is open over this metric, 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...', 'Statistics', and 'Max severity in trace browser'. The 'Statistics' option is highlighted. A callout bubble points to the 'Statistics' option with the text 'Access pattern instance statistics via context menu'. Another callout bubble points to the 'Statistics info' window with the text 'Click to get statistics details'. The 'Statistics info' window shows the following data for the 'mpi_latesender' pattern:

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%

The 'Statistics info' window also includes a 'To Clipboard' button and a 'Close' button. The 'Statistics' window shows a bar chart with a y-axis ranging from 0 to 0.035 and a single bar at 0.00. The 'Metric tree' at the bottom shows a color-coded bar for '1.38 (0.41%)' and a status bar indicating 'Shows metric statistics'.



The screenshot shows the 'cube 4.1.1' application window. The title bar reads 'cube 4.1.1 livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex'. The 'File' menu is open, showing options like 'Open...', 'Save as...', 'Close', 'Open external...', 'Close external', 'Connect to trace browser', 'Settings', 'Screenshot...', 'Quit', 'trace.cubex', and 'summary.cubex'. The 'Connect to trace browser' option is highlighted, and a submenu is visible with 'Connect to vampir...' and 'Connect to paraver...'. The 'Connect to vampir' dialog box is open, showing 'Open local file' checked, 'Host: localhost', 'Port: 30000', and 'File: c:/supermuc_expts/scorep_bt-mz_B_4x4_trace/traces.otf2'. The 'Browse' button is highlighted. The background shows a call tree and system tree view. A status bar at the bottom reads 'Connect to vampir and display a trace file'.

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

...and select trace file from the experiment directory



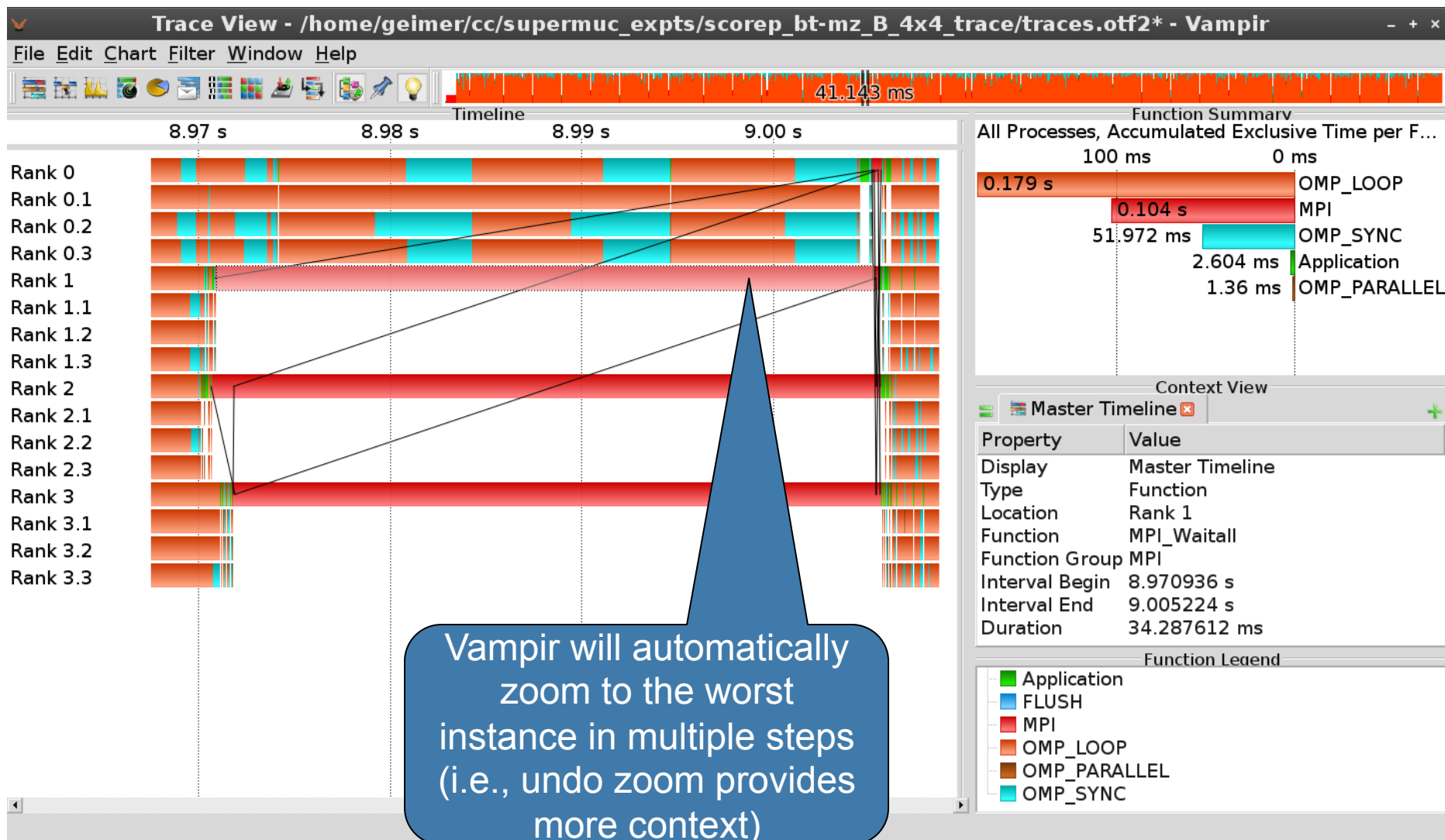
Show most severe pattern instances

The screenshot displays the cube 4.1.1 interface for the file `livedvd2: scorep_bt-mz_B_4x4_trace/trace.cubex`. It features three main panels:

- Metric tree:** Shows a hierarchical view of performance metrics. The 'Late Sender' metric is highlighted with a blue background and a red frame, indicating its severity.
- Call tree:** Displays a call graph. The '1.38 MPI_Wait' entry is highlighted with a blue background and a red frame. A context menu is open over this entry, showing options like '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' (which is highlighted).
- System tree:** Shows a system tree view with a 'Box Plot' option.

A blue callout box points to the '1.38 MPI_Wait' entry in the Call tree, containing the text: "Select 'Max severity in trace browser' from context menu of call paths marked with a red frame".

Investigate most severe instance in Vampir



Scalable performance analysis of large-scale parallel applications

- 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

