

# Automatic trace analysis with the Scalasca Trace Tools

---

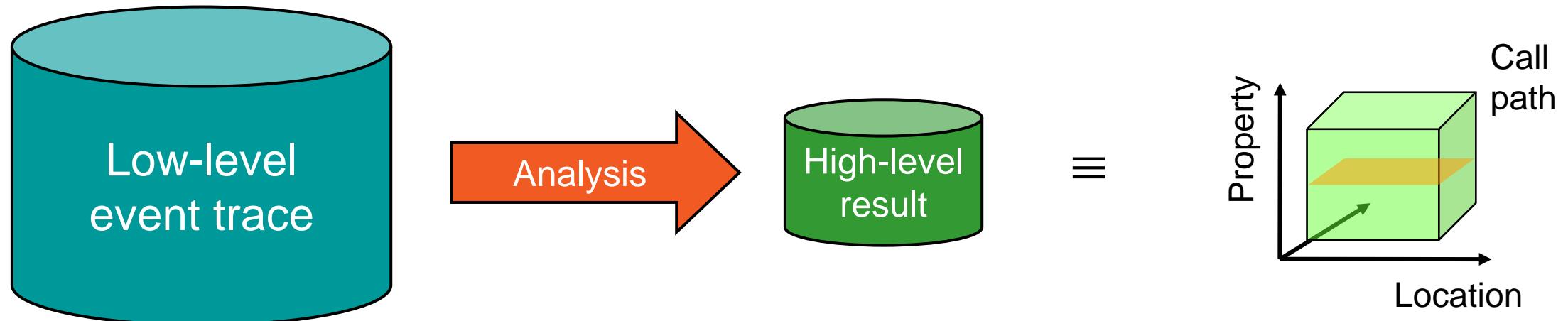
Brian Wylie  
Jülich Supercomputing Centre

trace tools   
scalasca

# 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

---

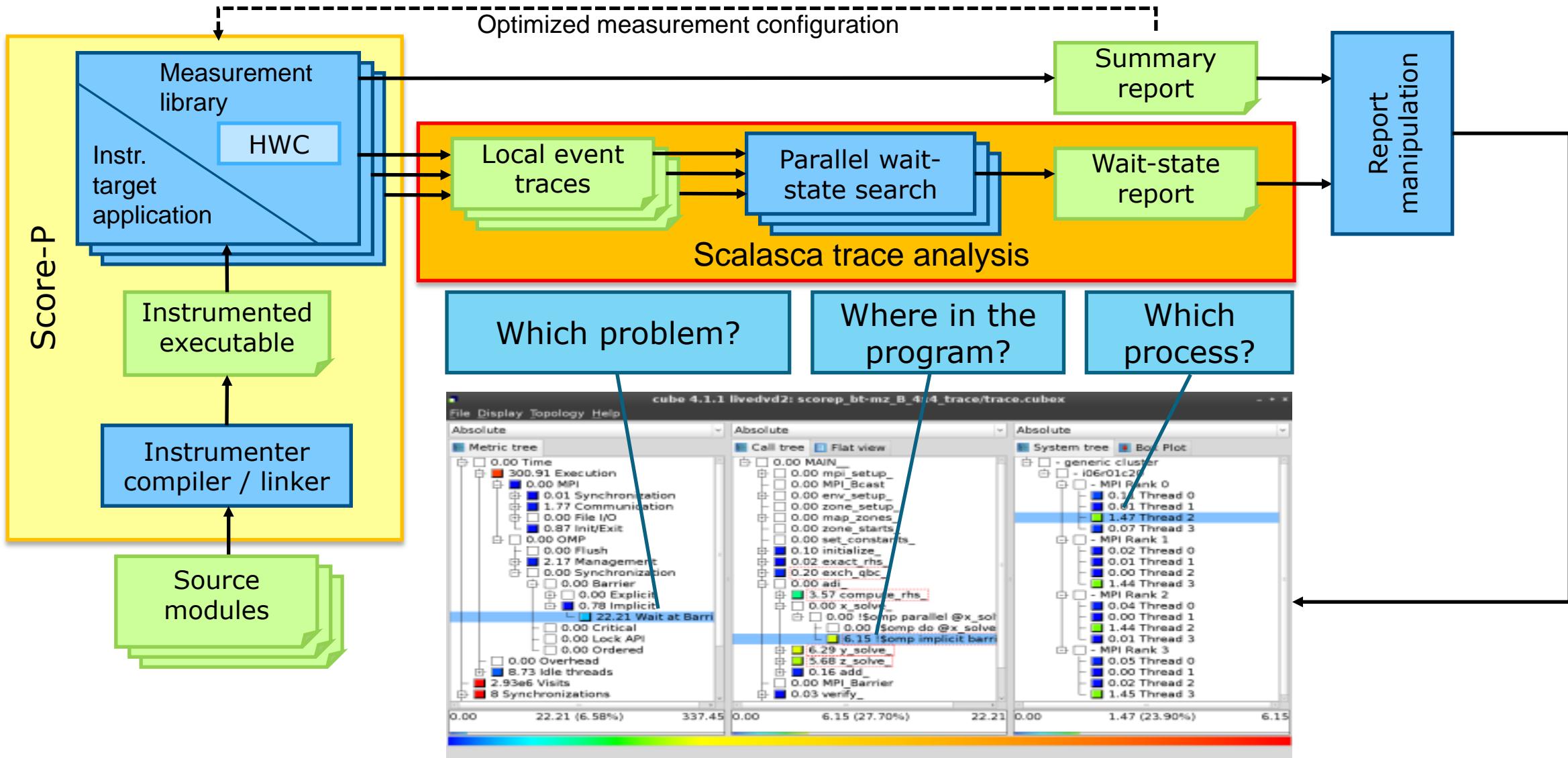
- Development of a **scalable trace-based** performance analysis toolset for the most popular parallel programming paradigms
  - Current focus: MPI, OpenMP, and POSIX threads
- Specifically targeting large-scale parallel applications
  - Such as those running on IBM Blue Gene or Cray systems with one million or more processes/threads
- Latest release:
  - Scalasca v2.5 coordinated with Score-P v5.0 (March 2019)

# Scalasca Trace Tools features

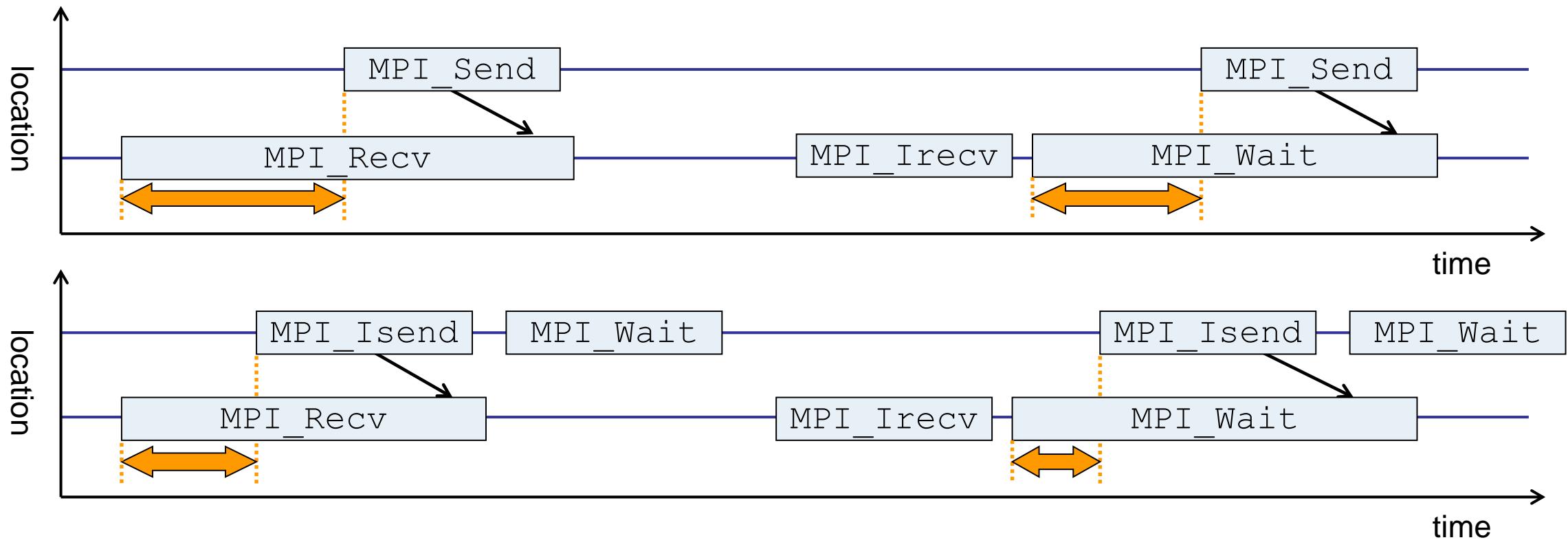
---

- 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 CUDA or SHMEM events, or OpenMP nested parallelism
  - PAPI/rusage metrics for trace events are ignored

# Scalasca workflow

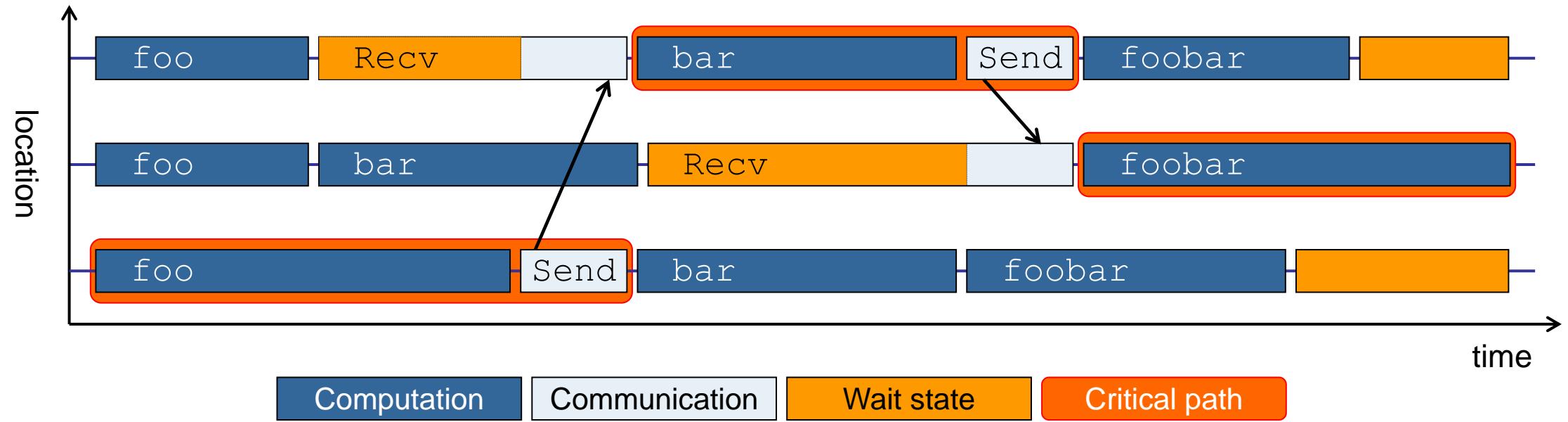


## Example: “*Late Sender*” wait state



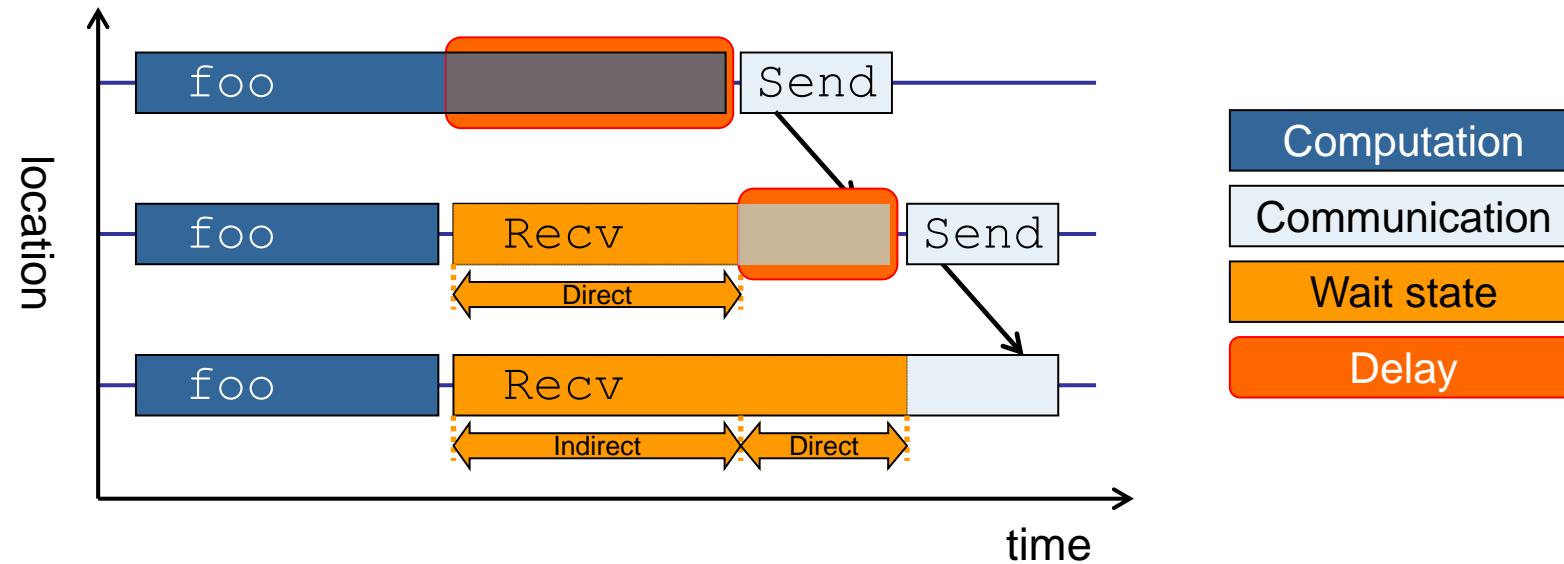
- Waiting time caused by a blocking receive operation posted earlier than the corresponding send
- Applies to blocking as well as non-blocking communication

## Example: Critical path

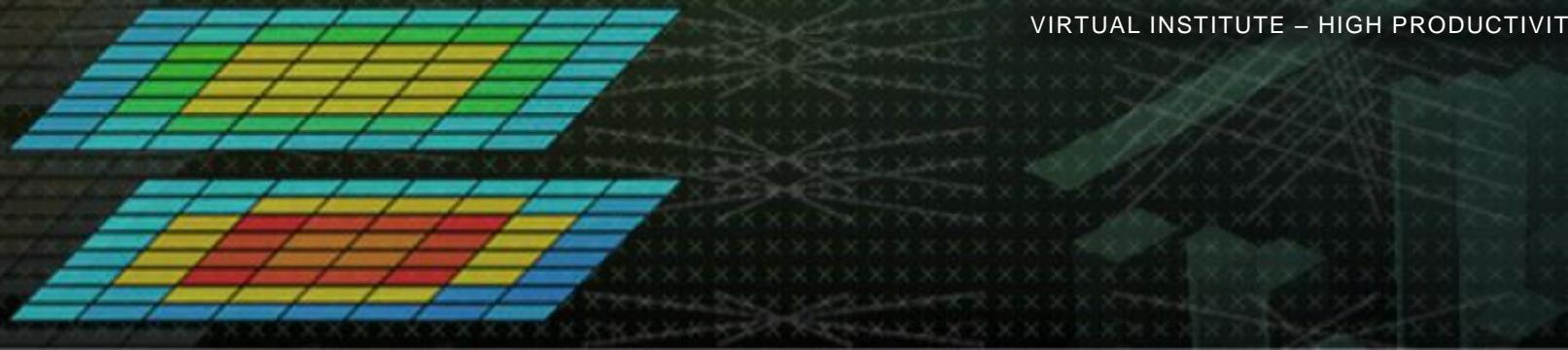


- Shows call paths and processes/threads that are responsible for the program's wall-clock runtime
- Identifies good optimization candidates and parallelization bottlenecks

## 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 2.4
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 compatibility command: skin / scalasca -instrument

```
% skin
```

```
Scalasca 2.4: 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
  - **Deprecated! Use Score-P instrumenter directly.**

# Scalasca convenience command: scan / scalasca -analyze

```
% scan
Scalasca 2.4: 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) 1998-2018 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 / scalasca -examine

```
% square
```

```
Scalasca 2.4: 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 (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)

# Setup environment

---

- Remember to source provided shell code snippet to add local tool installations to \$PATH

```
% source ~tg828282/Tutorial/vihps-intel.sh
```

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

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

## BT-MZ summary measurement collection...

```
% cd bin.scorep
% cp ..../jobscript/stampede2/scalasca.sbatch .
% vi scalasca.sbatch

# Score-P measurement configuration
export SCOREP_FILTERING_FILE=..../config/scorep.filt
#export SCOREP_TOTAL_MEMORY=50M
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC

# Scalasca configuration
export SCAN_ANALYZE_OPTS="--time-correct"

# Run the application using Scalasca nexus
scalasca -analyze ibrun ./bt-mz_{CLASS}.{PROCS}
```

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

```
% sbatch scalasca.sbatch
```

- Submit the job

# BT-MZ summary measurement

```
S=C=A=N: Scalasca 2.4 runtime summarization
S=C=A=N: ./scorep_bt-mz_C_32x4_sum experiment archive
S=C=A=N: Mon Aug 21 07:52:03 2017: Collect start
ibrun ./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 Aug 21 07:52:36 2017: Collect done (status=0) 33s
S=C=A=N: ./scorep_bt-mz_C_32x4_sum complete.
```

- Run the application using the Scalasca measurement collection & analysis nexus prefixed to launch command
- Creates experiment directory:  
scorep\_bt-mz\_C\_32x4\_sum

# BT-MZ summary analysis report examination

- Score summary analysis report

```
% square -s scorep_bt-mz_C_32x4_sum  
INFO: Post-processing runtime summarization result...  
INFO: Score report written to ./scorep_bt-mz_C_32x4_sum/scorep.score
```

**Hint:**

Copy ‘summary.cubex’ to Live-DVD environment using ‘scp’ to improve responsiveness of GUI

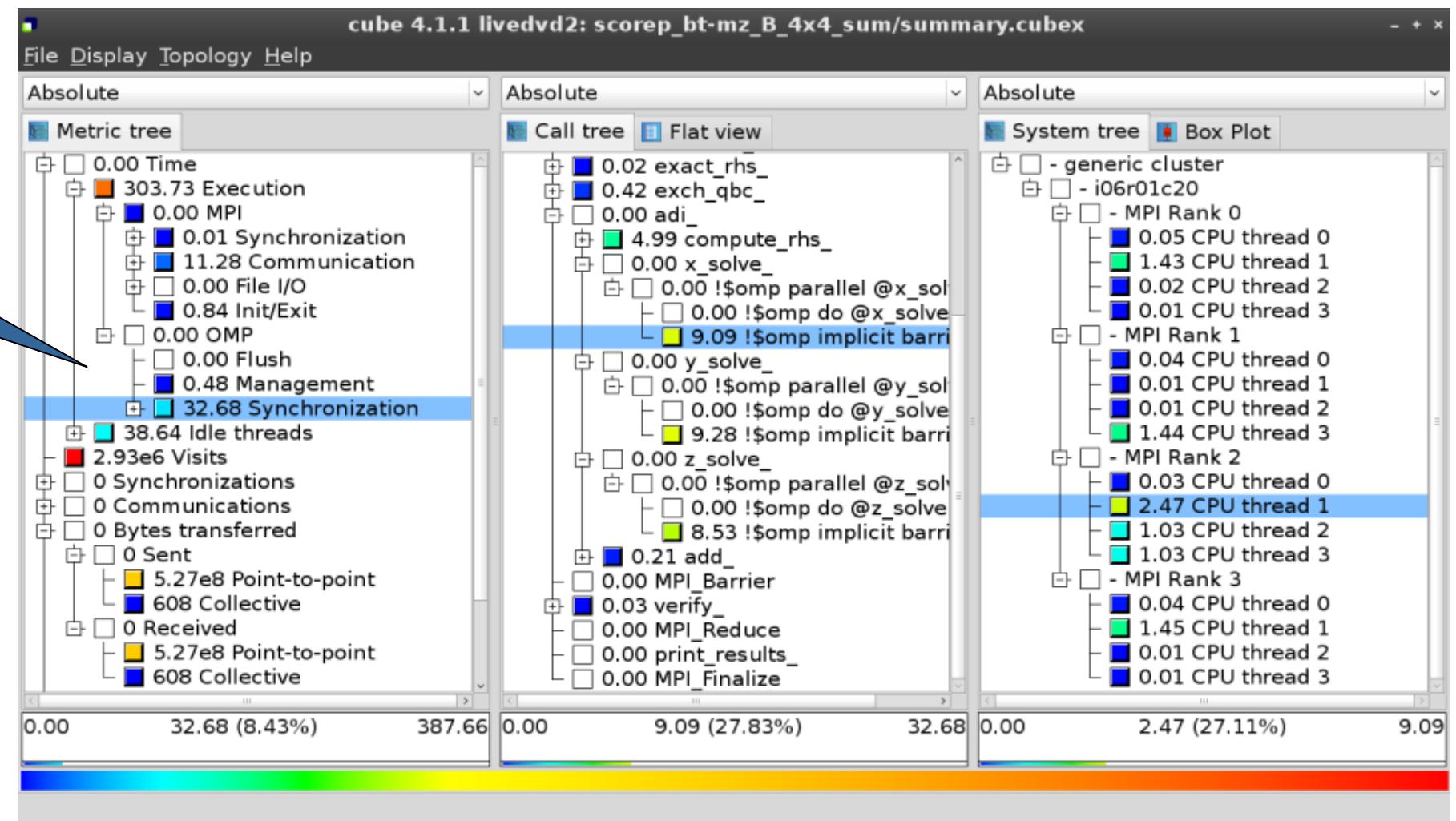
- Post-processing and interactive exploration with Cube

```
% square scorep_bt-mz_C_32x4_sum  
INFO: Displaying ./scorep_bt-mz_C_32x4_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...

```
% cd bin.scorep
% cp ..../jobscript/stampede2/scalasca.sbatch .
% vi scalasca.sbatch

# Score-P measurement configuration
export SCOREP_FILTERING_FILE=../config/scorep.filt
export SCOREP_TOTAL_MEMORY=50M
export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC

# Scalasca configuration
export SCAN_ANALYZE_OPTS="--time-correct"

# Run the application using Scalasca nexus
scalasca -analyze -t ibrun ./bt-mz_${CLASS}.${PROCS}
```

```
% sbatch scalasca.sbatch
```

- Change to directory with the executable and edit the job script
- Add “-t” to the scalasca -analyze command

- Submit the job

## BT-MZ trace measurement ... collection

```
S=C=A=N: Scalasca 2.4 trace collection and analysis
S=C=A=N: Mon Aug 21 07:58:54 2017: Collect start
ibrun ./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 Aug 21 07:59:29 2017: Collect done (status=0) 35s
```

- Starts measurement with collection of trace files ...

# BT-MZ trace measurement ... analysis

```
S=C=A=N: Mon Aug 21 07:59:30 2017: Analyze start
ibrun scout.hyb ./scorep_bt-mz_C_32x4_trace/traces.otf2

Analyzing experiment archive ./scorep_bt-mz_C_32x4_trace/traces.otf2

Opening experiment archive ... done (0.040s).
Reading definition data ... done (0.127s).
Reading event trace data ... done (0.726s).
Preprocessing ... done (0.311s).
Timestamp correction ... done (0.556s).
Analyzing trace data ... done (15.144s).
Writing analysis report ... done (0.738s).

Total processing time      : 17.754s
S=C=A=N: Mon Aug 21 07:59:50 2017: Analyze done (status=0) 20s
```

- 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_32x4_trace
INFO: Post-processing runtime summarization result...
INFO: Post-processing trace analysis report...
INFO: Displaying ./scorep_bt-mz_C_32x4_trace/trace.cubex...
```

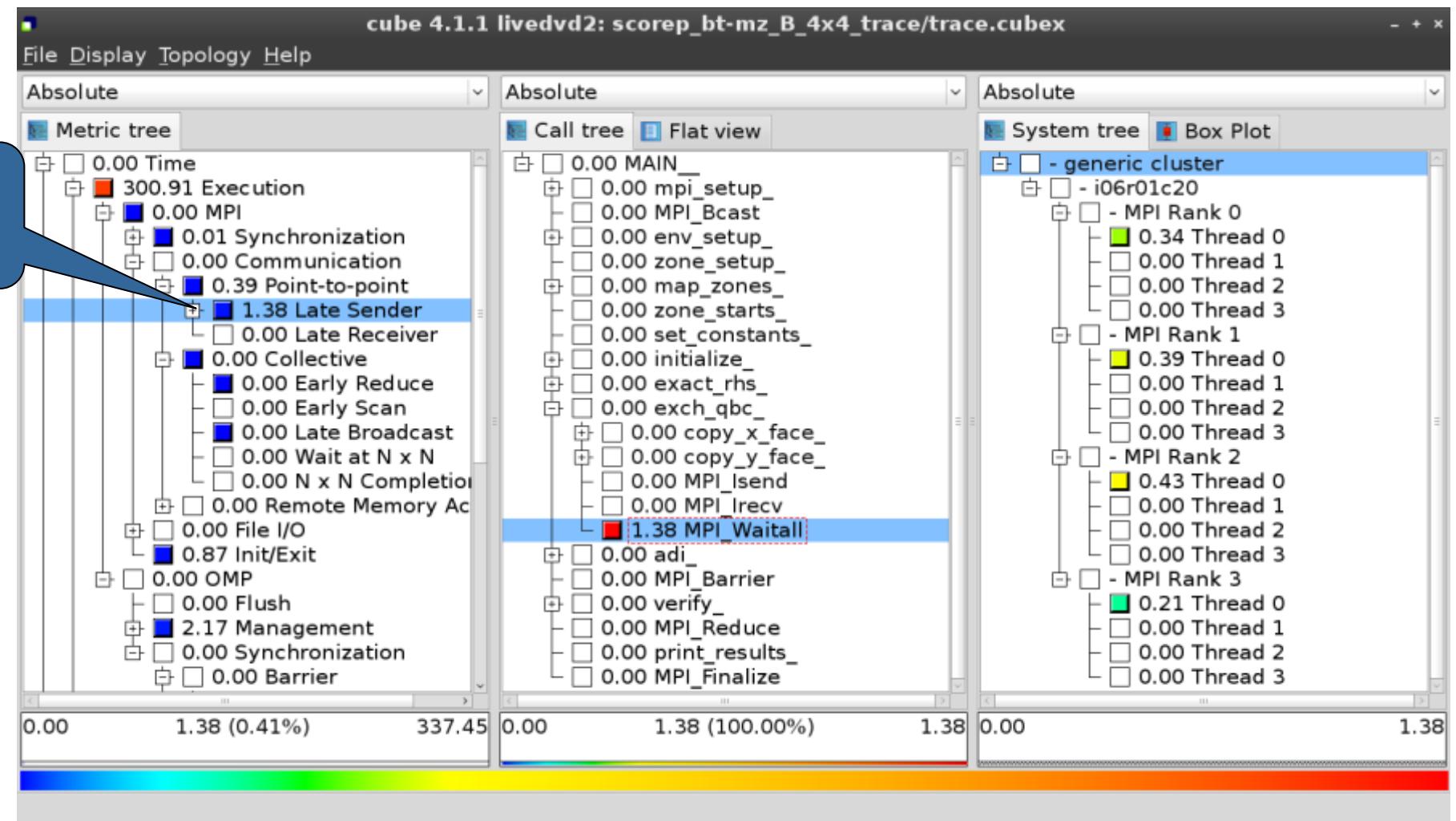
[GUI showing trace analysis report]

## Hint:

Run ‘square -s’ first and then copy ‘trace.cubex’ to Live-DVD environment using ‘scp’ to improve responsiveness of GUI

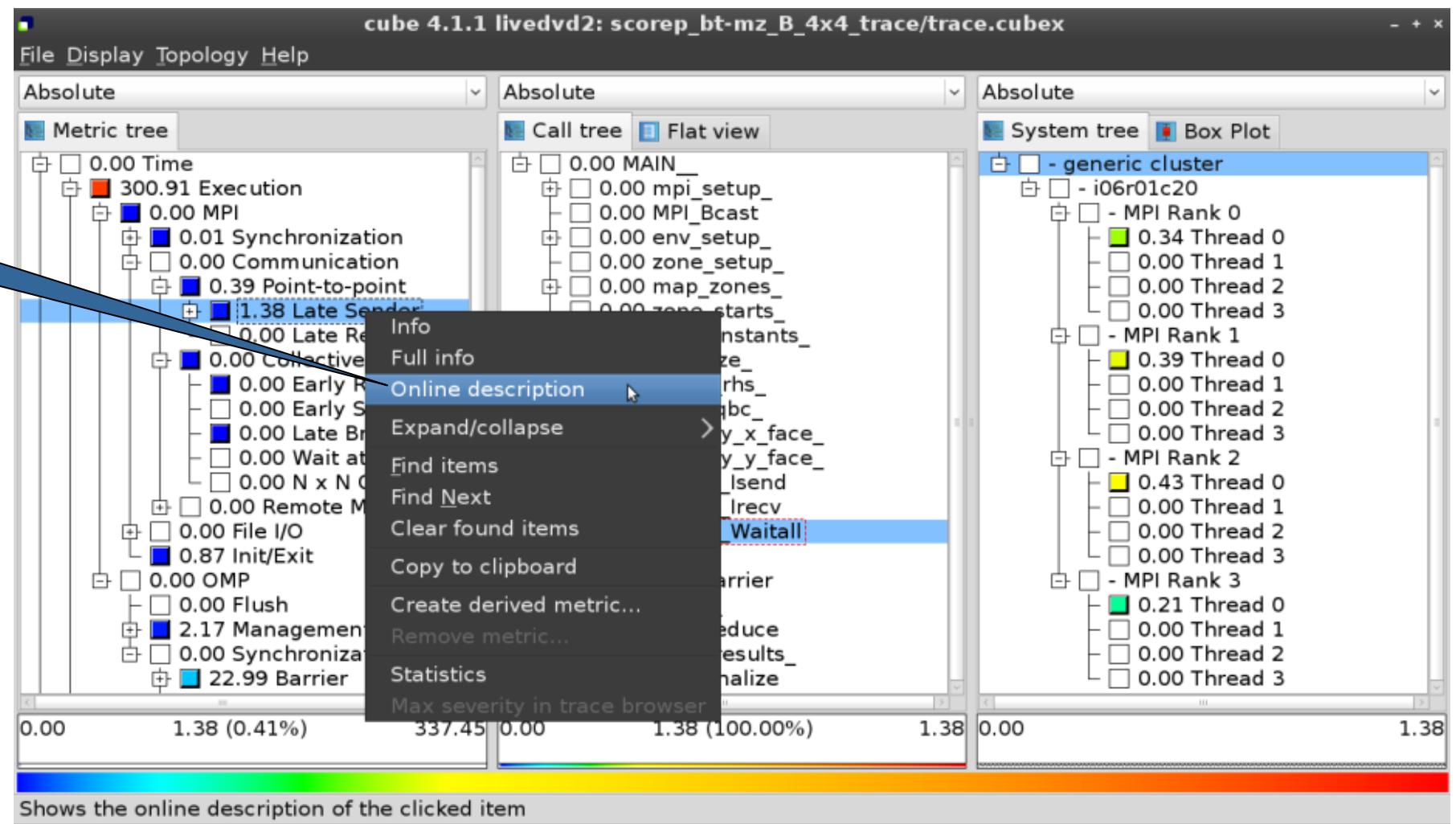
# Post-processed trace analysis report

Additional trace-based metrics in metric hierarchy



# Online metric description

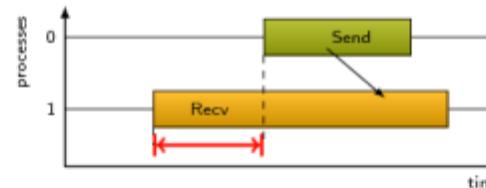
Access online metric description via context menu



# Online metric description

**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 a 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.

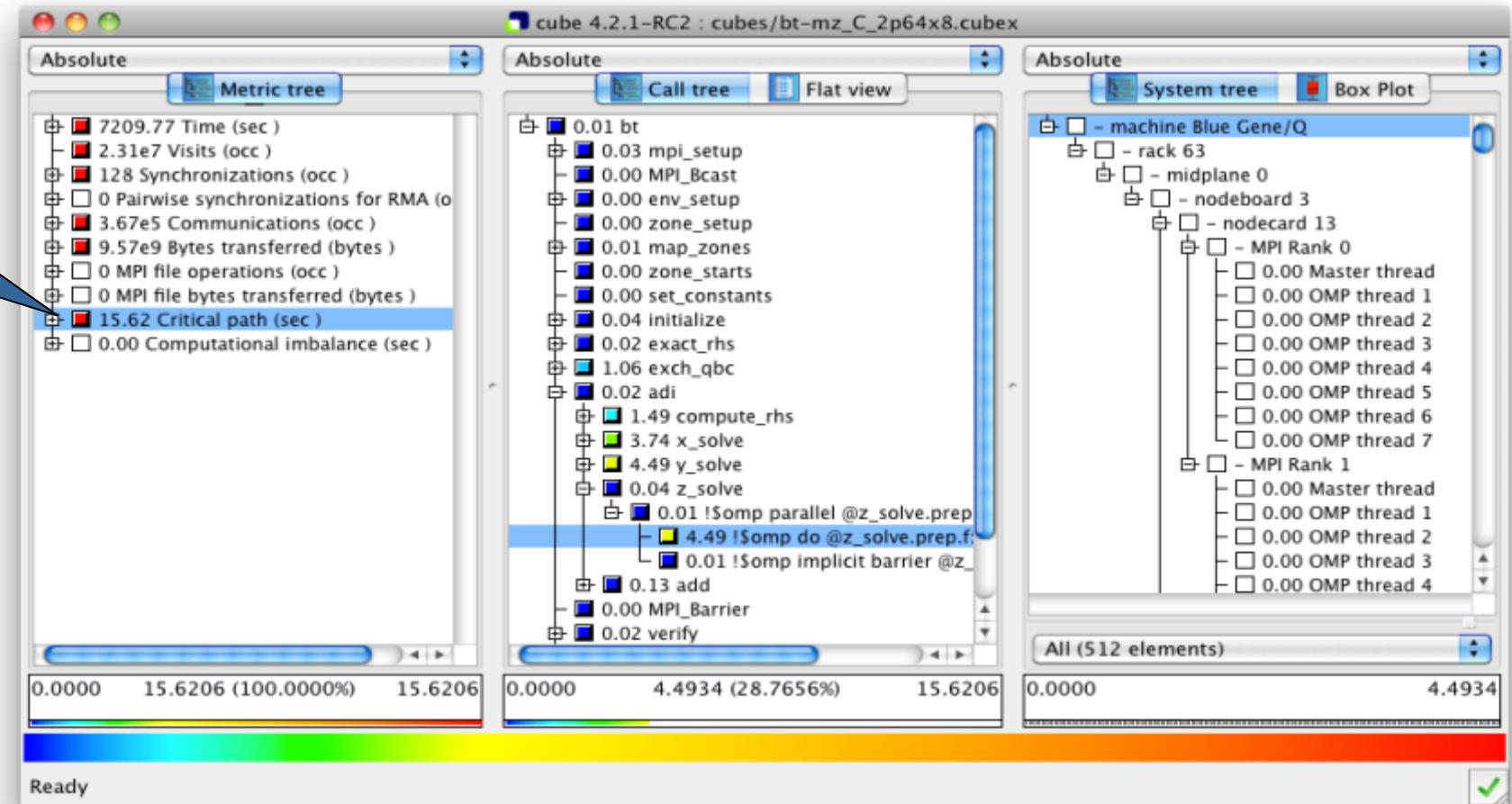
**Parent:**  
[MPI Point-to-point Communication Time](#)

**Children:**

[Close](#)

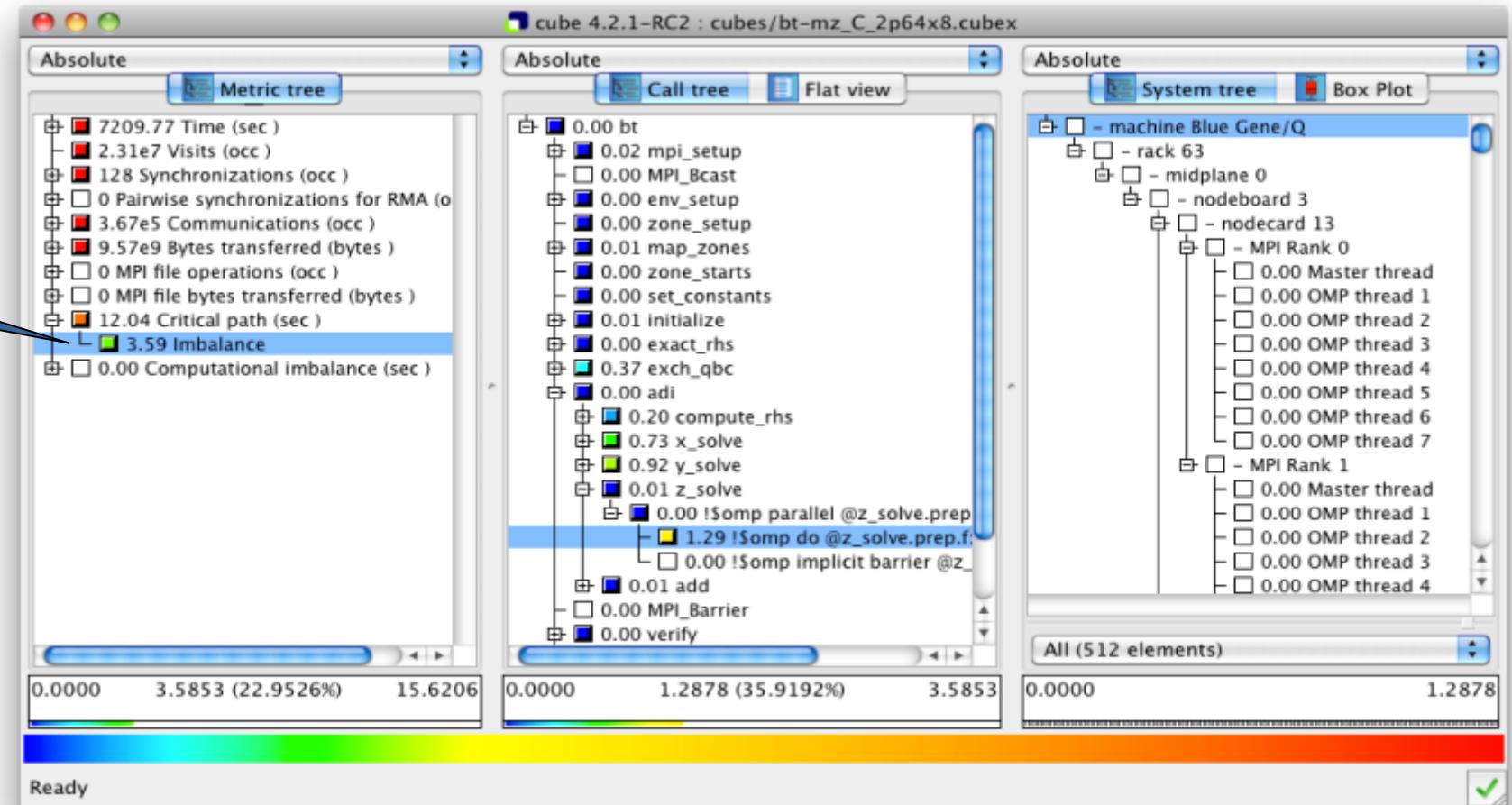
# Critical-path analysis

Critical-path profile shows wall-clock time impact

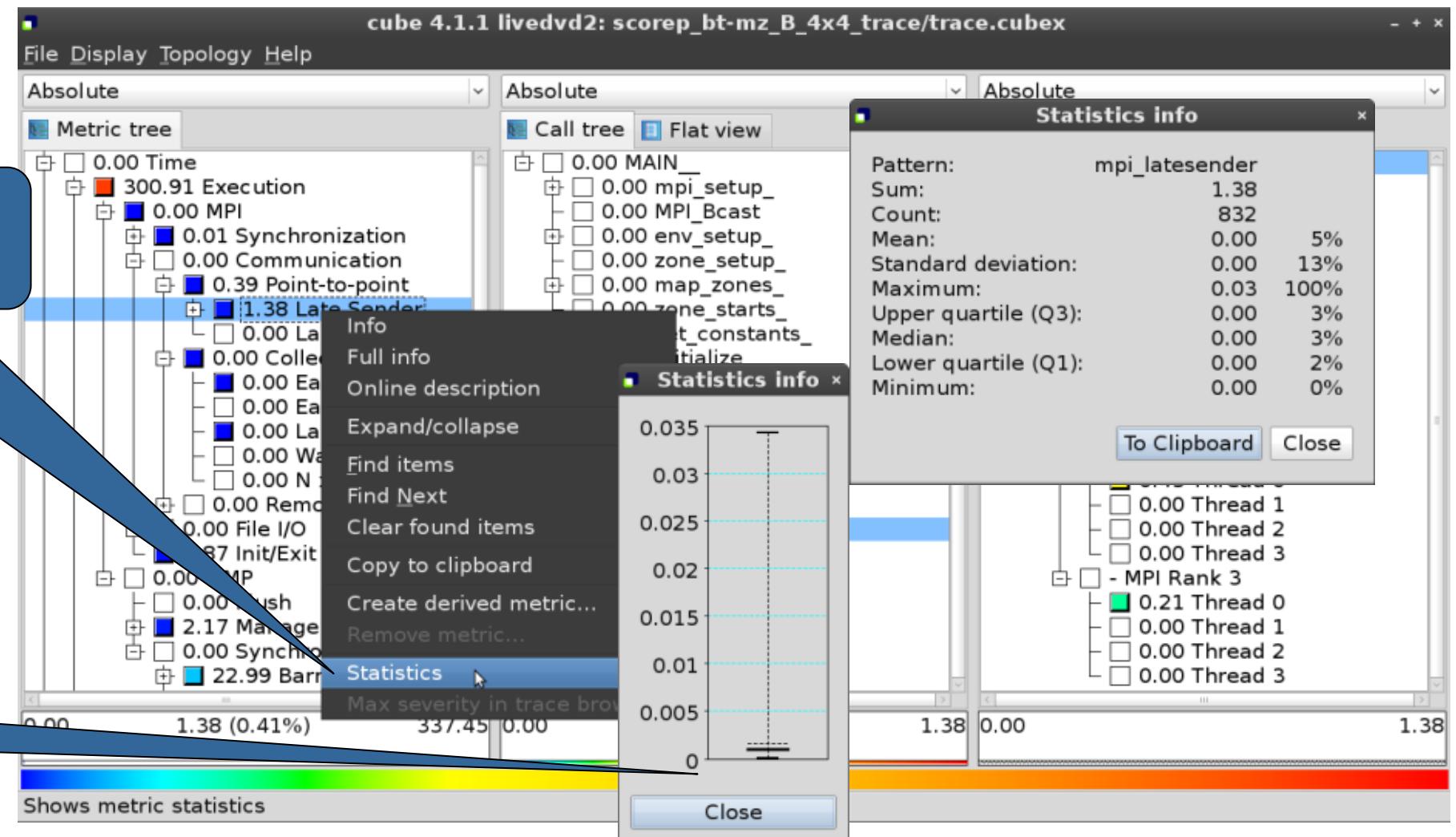


# Critical-path analysis

Critical-path imbalance highlights inefficient parallelism

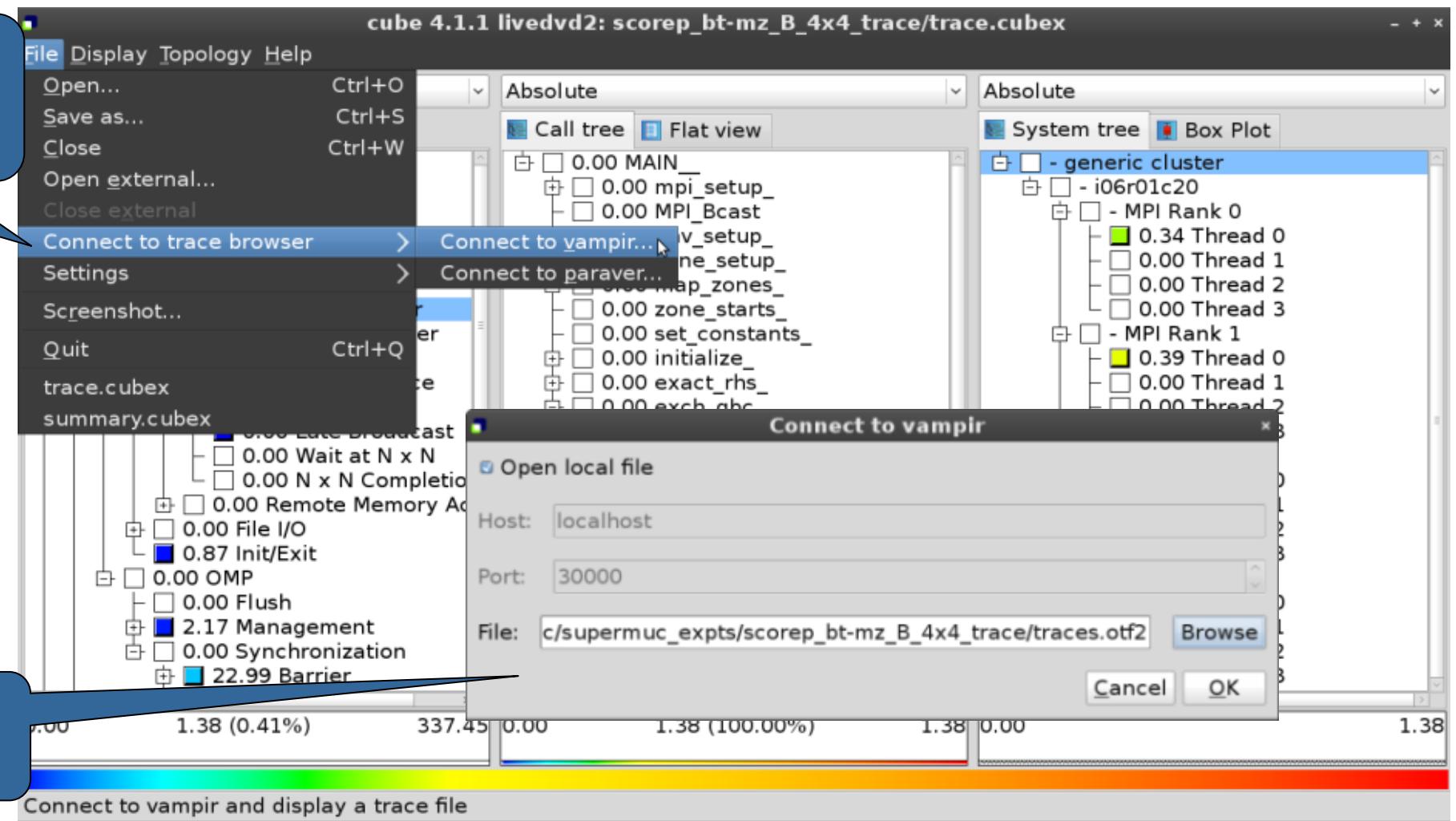


# Pattern instance statistics



# Connect to Vampir trace browser

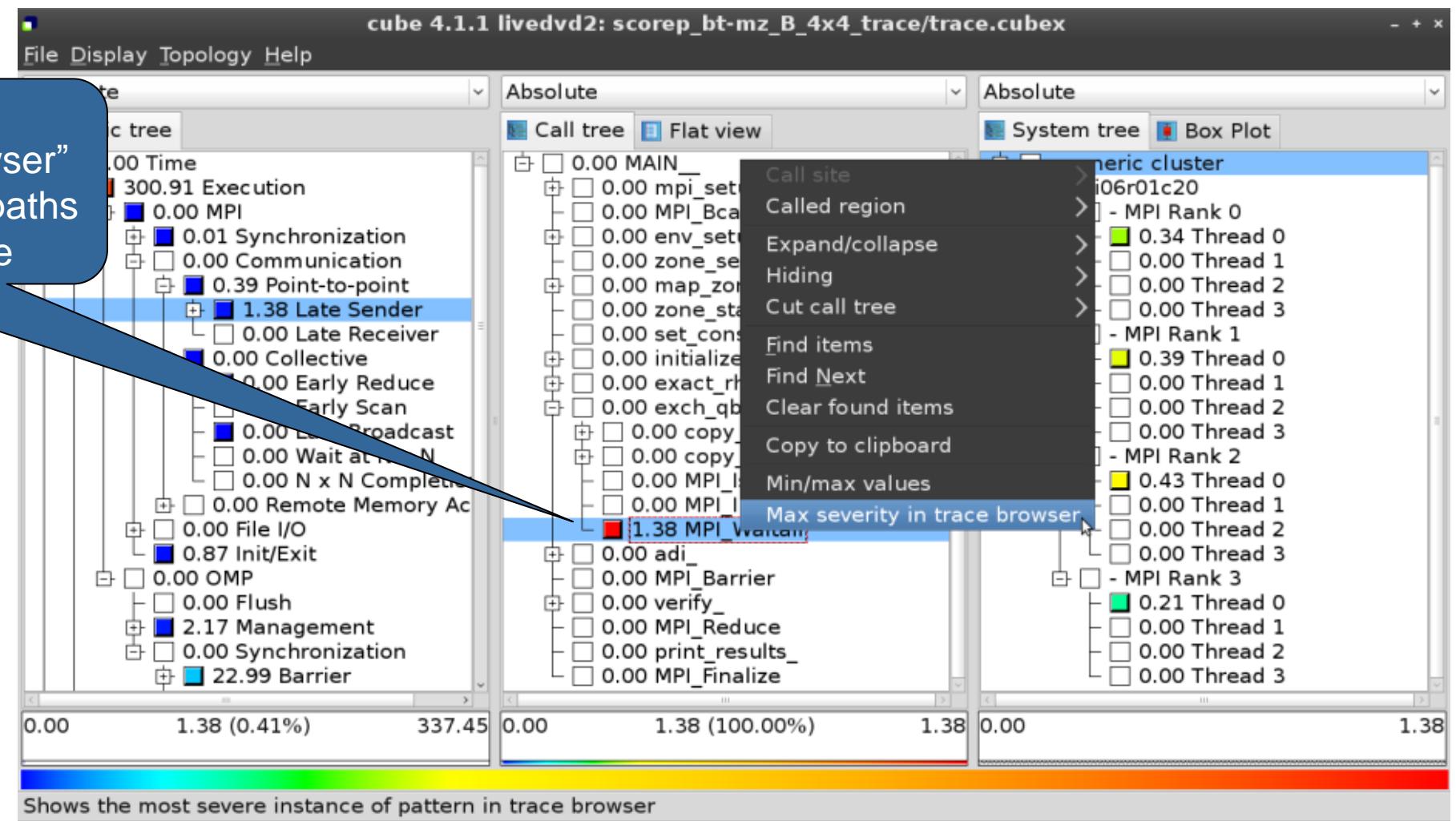
To investigate most severe pattern instances, connect to a trace browser...



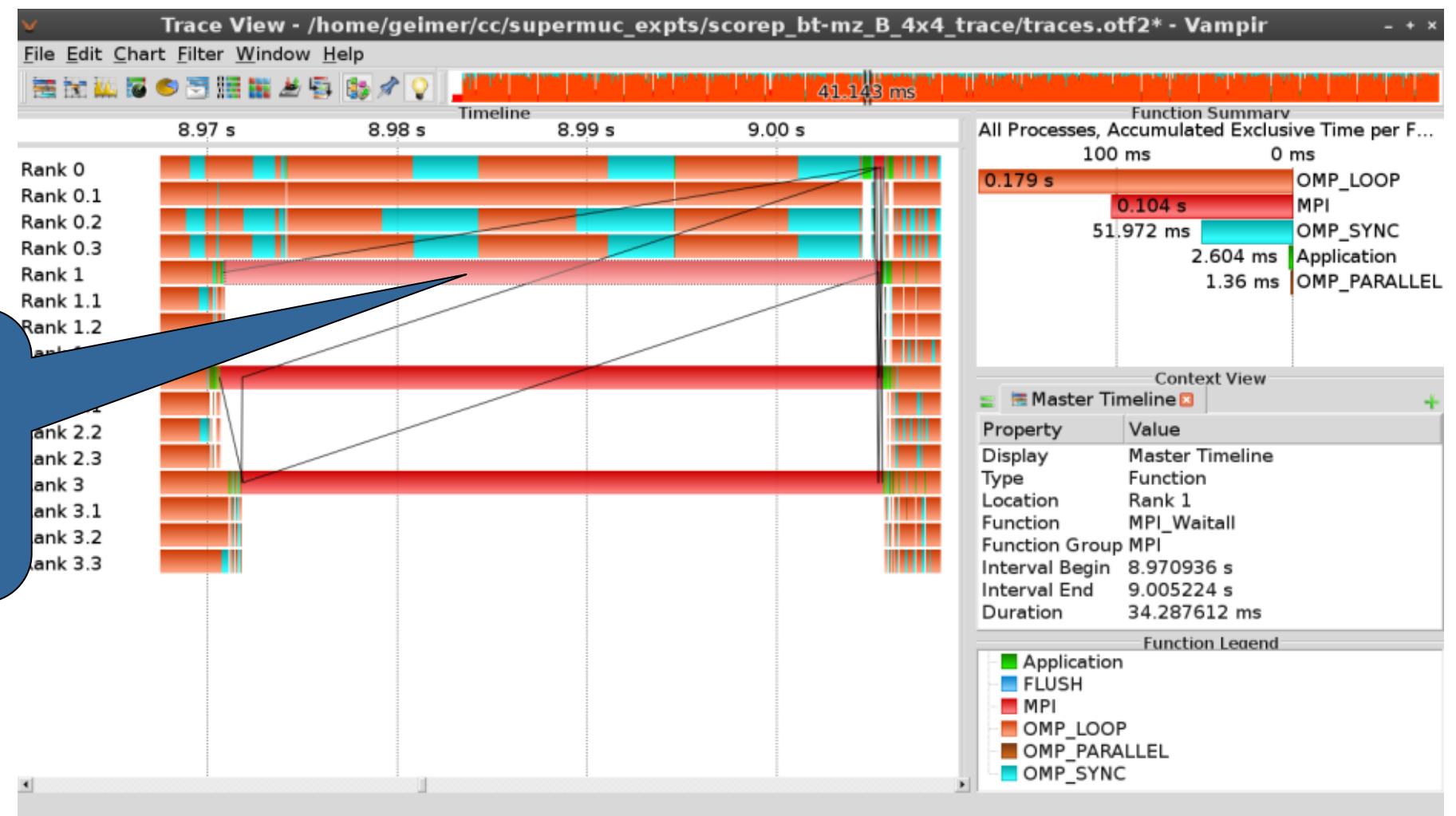
...and select trace file from the experiment directory

# 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



# 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](mailto:scalasca@fz-juelich.de)

