THE SCORE-P ECOSYSTEM
TRACING WITH SCORE-P AND SCALASCA

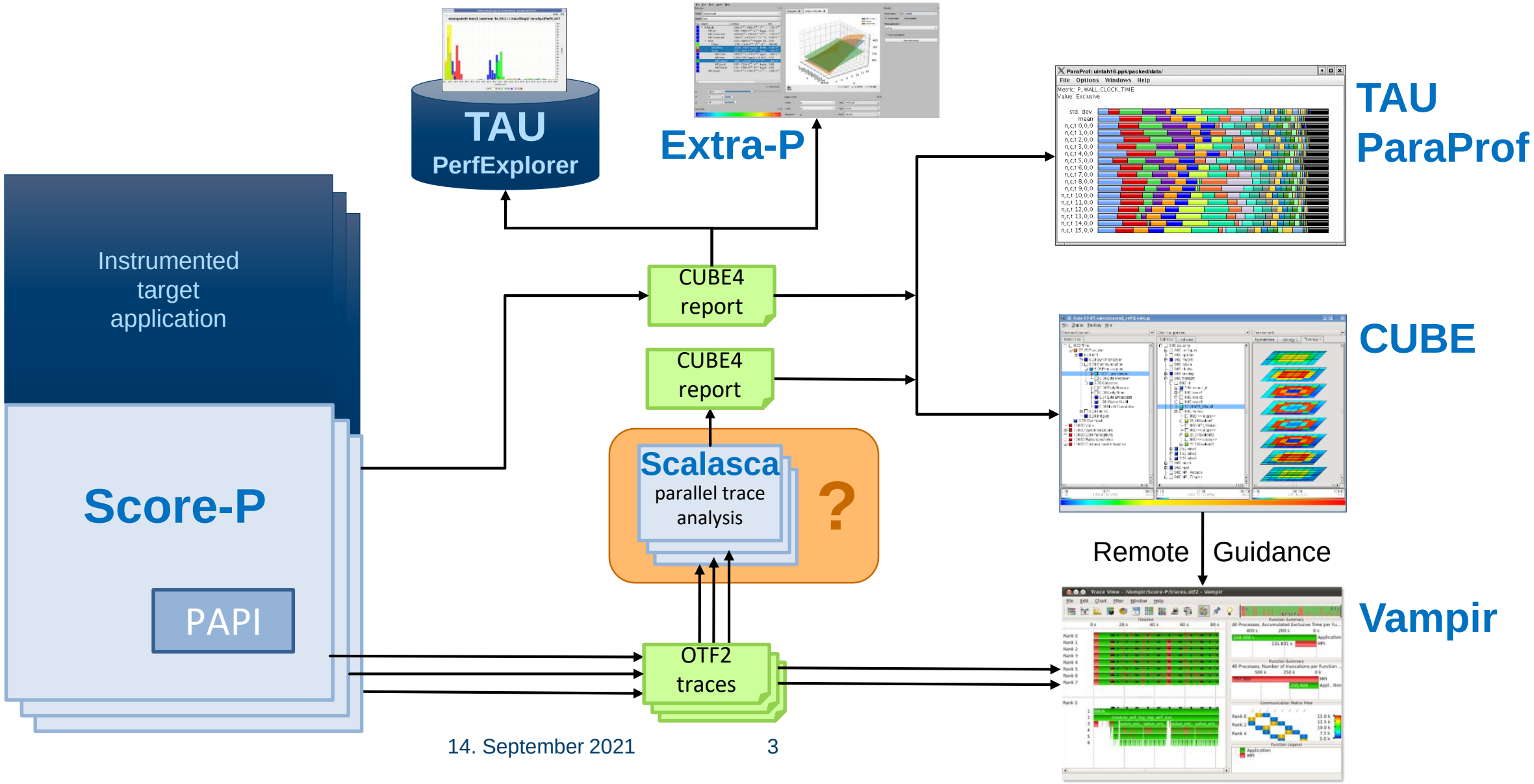
SEP 14, 2018 I BERND MOHR

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

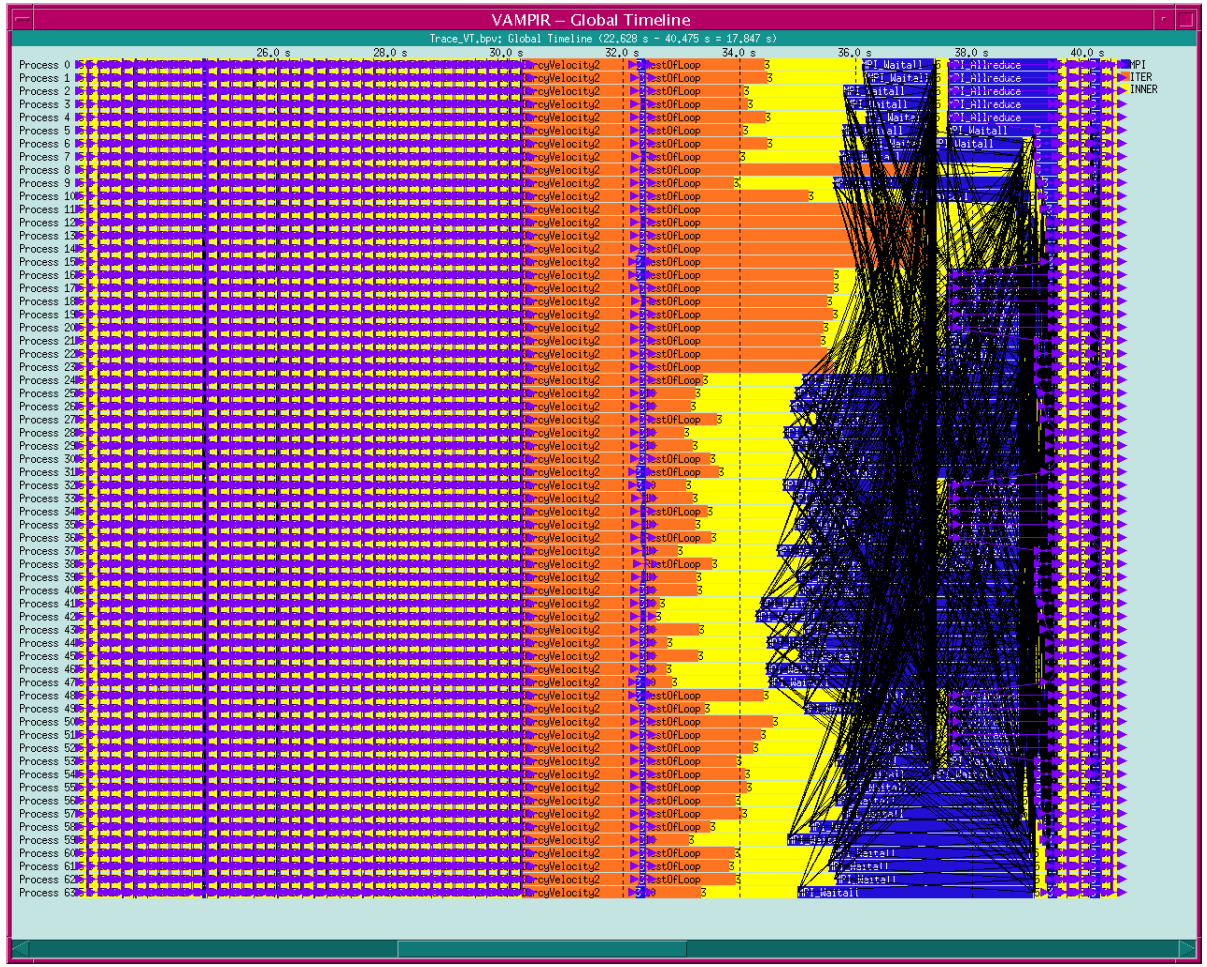
The Big Picture

THE SCORE-P ECOSYSTEM + THE SCALASCA TRACE ANALYZER

Score-P TOOL ECOSYSTEM



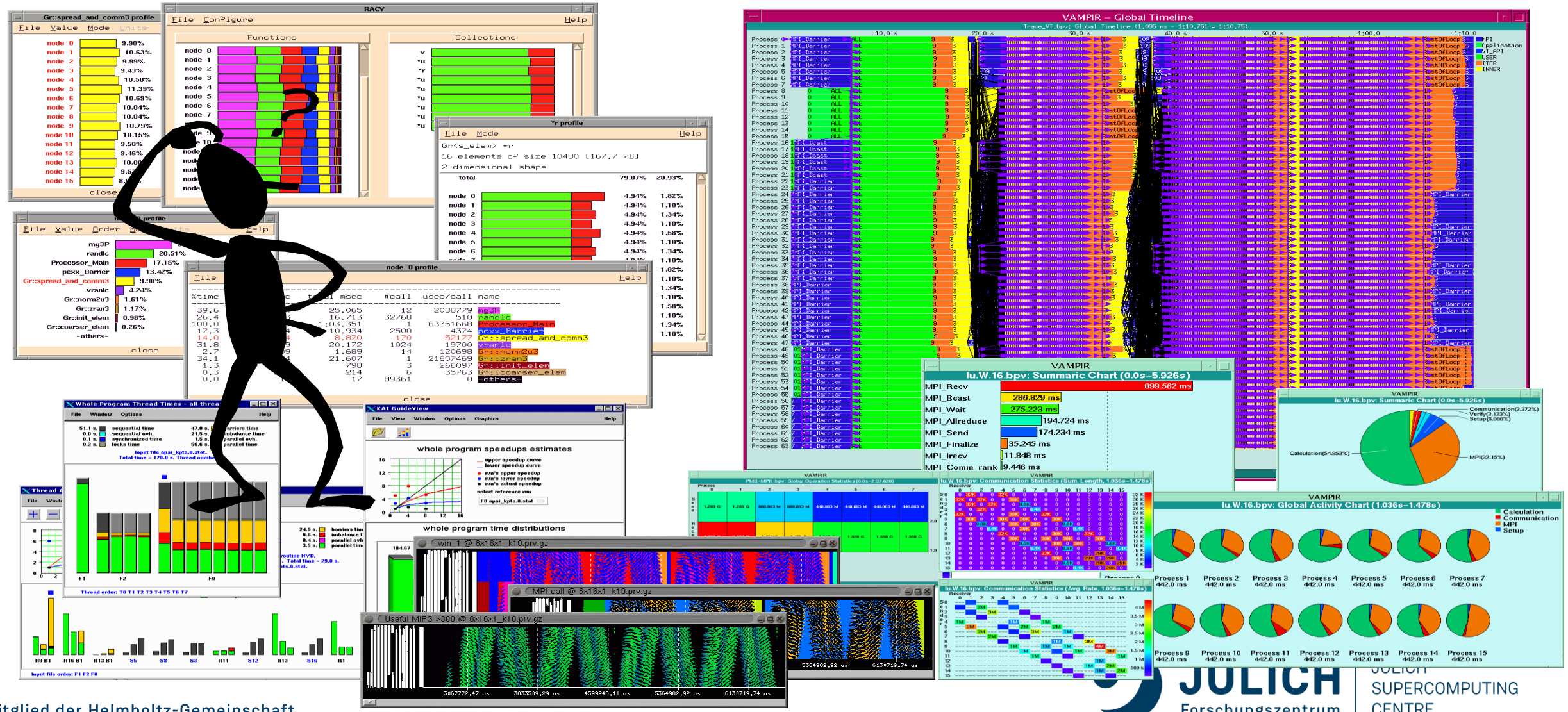
“A PICTURE IS WORTH 1000 WORDS…”



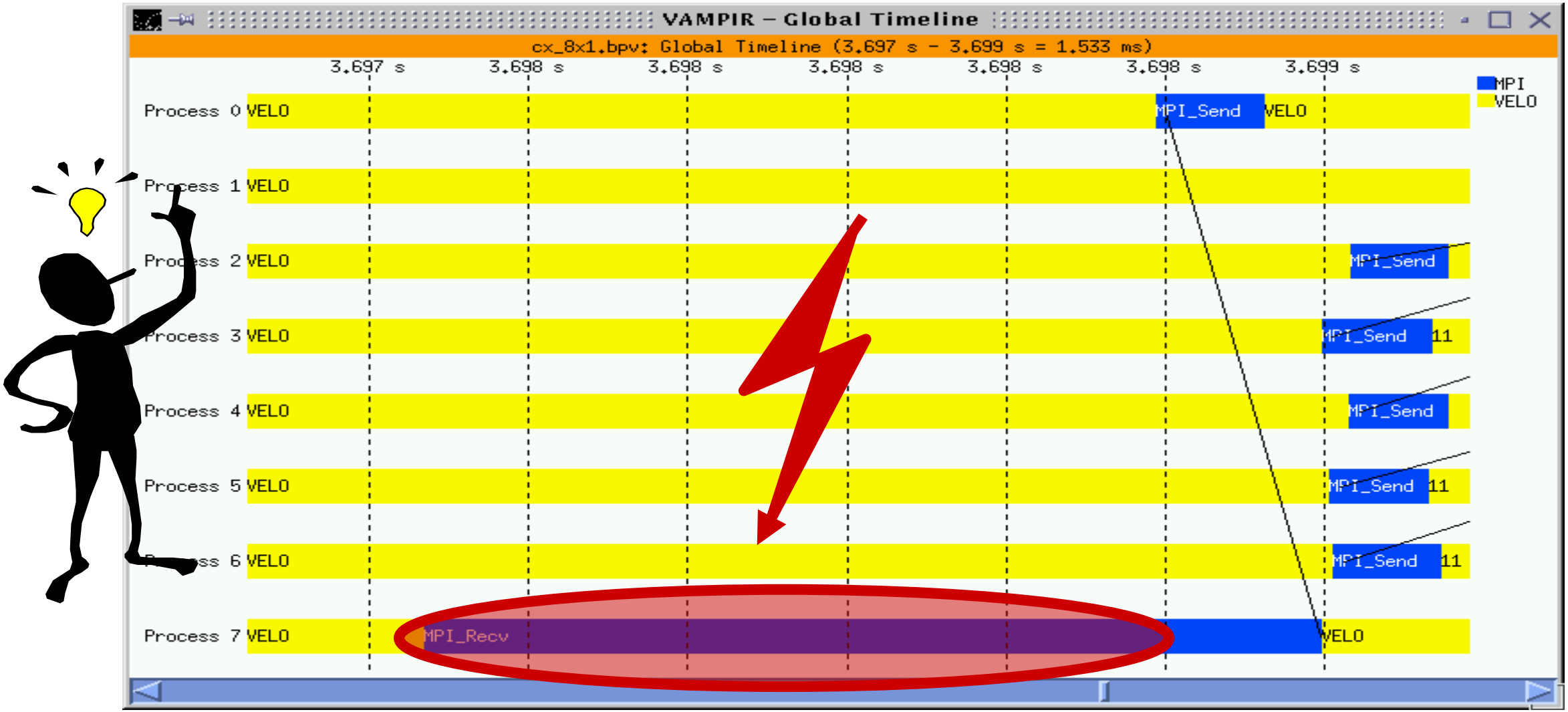
- MPI ring program
- “Real world” example

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

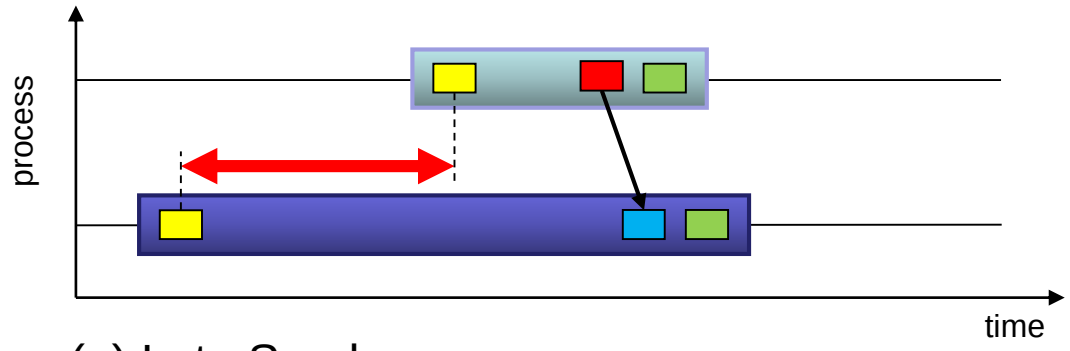
“WHAT ABOUT 1000’S OF PICTURES?” (WITH 100’S OF MENU OPTIONS)



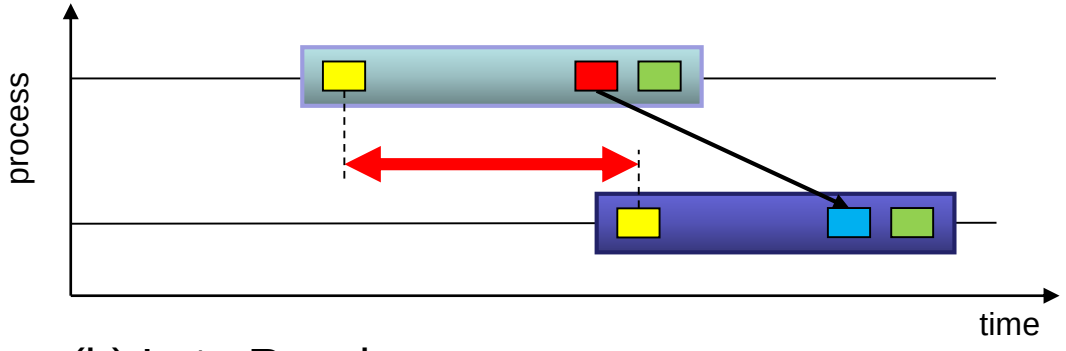
EXAMPLE AUTOMATIC ANALYSIS: LATE SENDER



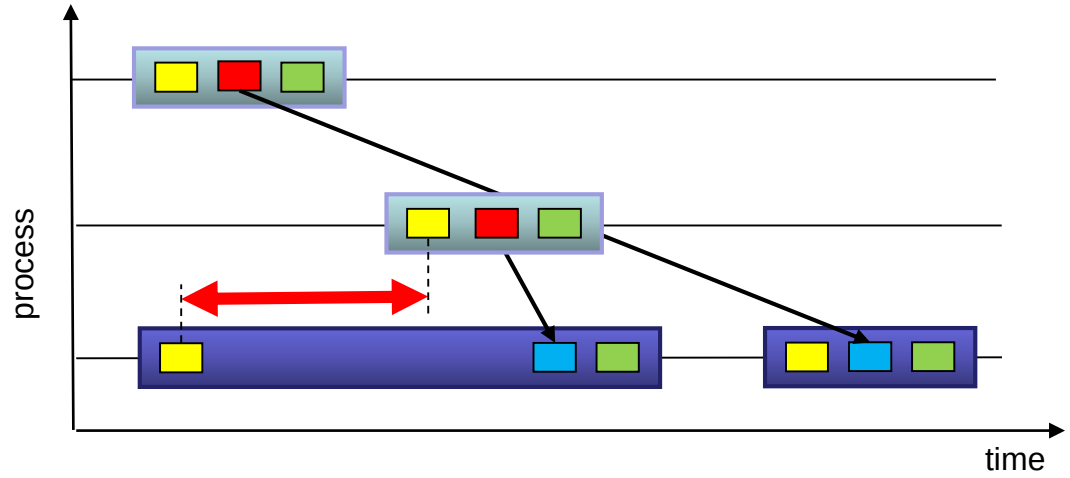
EXAMPLE MPI WAIT STATES



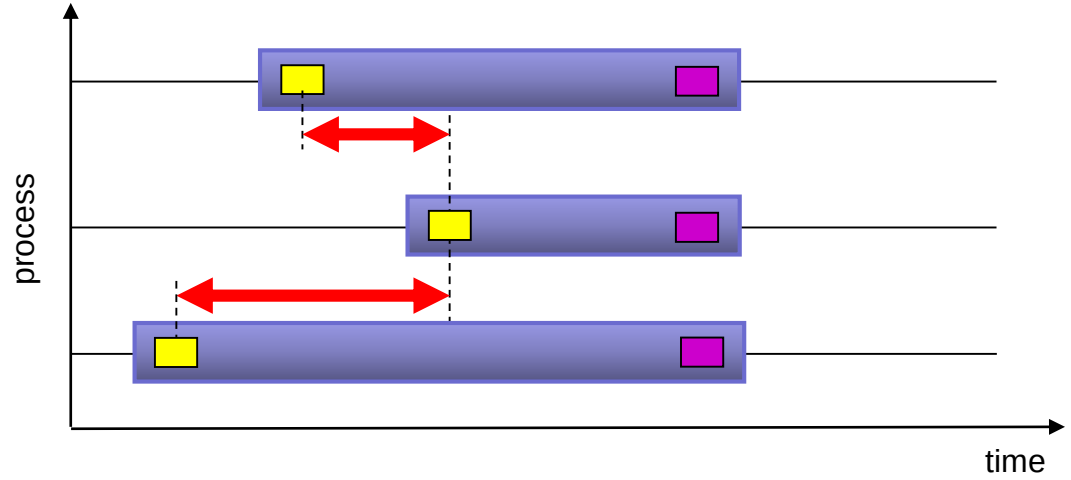
(a) Late Sender



(b) Late Receiver



(c) Late Sender / Wrong Order



(d) Wait at N x N

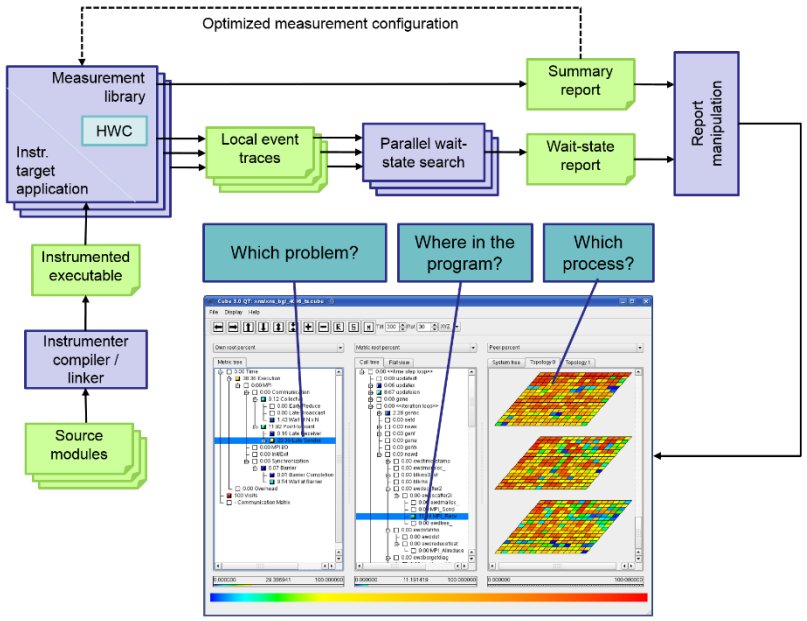
ENTER EXIT SEND RECV COLLEXIT

SCALASCA

http://www.scalasca.org/

- Scalable Analysis of Large Scale Applications

- Approach
  - **Instrument** C, C++, and Fortran parallel applications (**with Score-P**)
  - Option 1: **scalable call-path profiling**
  - Option 2: **scalable event trace analysis**
    - **Collect** event traces
    - **Process trace in parallel**
      - Wait-state analysis
      - Delay and root-cause analysis
      - Critical path analysis
    - **Categorize and rank** results



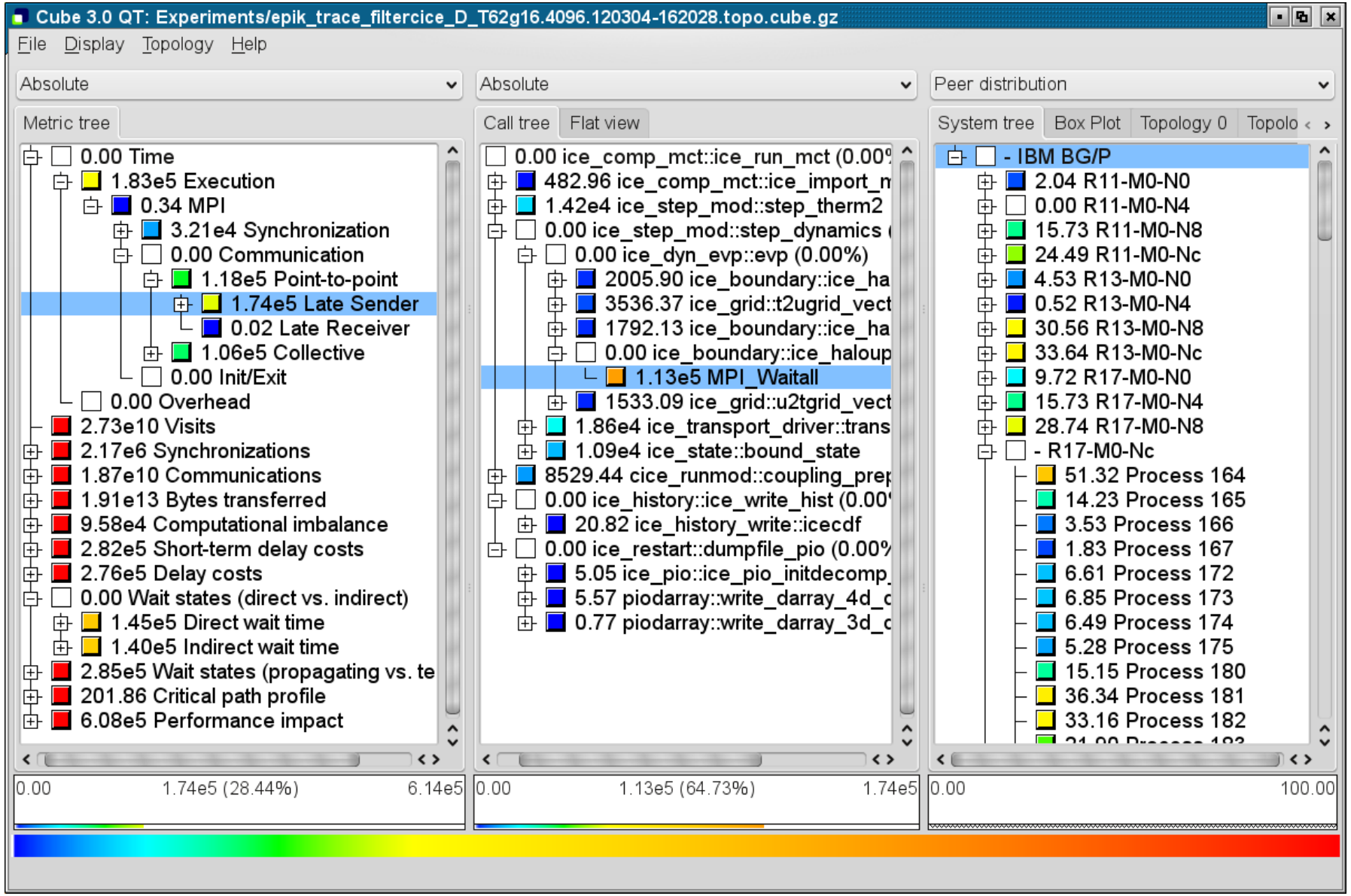
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

SCALASCA EXAMPLE: CESM SEA ICE MODULE

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Late Sender Analysis

- Finds waiting at MPI_Waitall() inside ice boundary halo update
- Shows distribution of imbalance across system and ranks

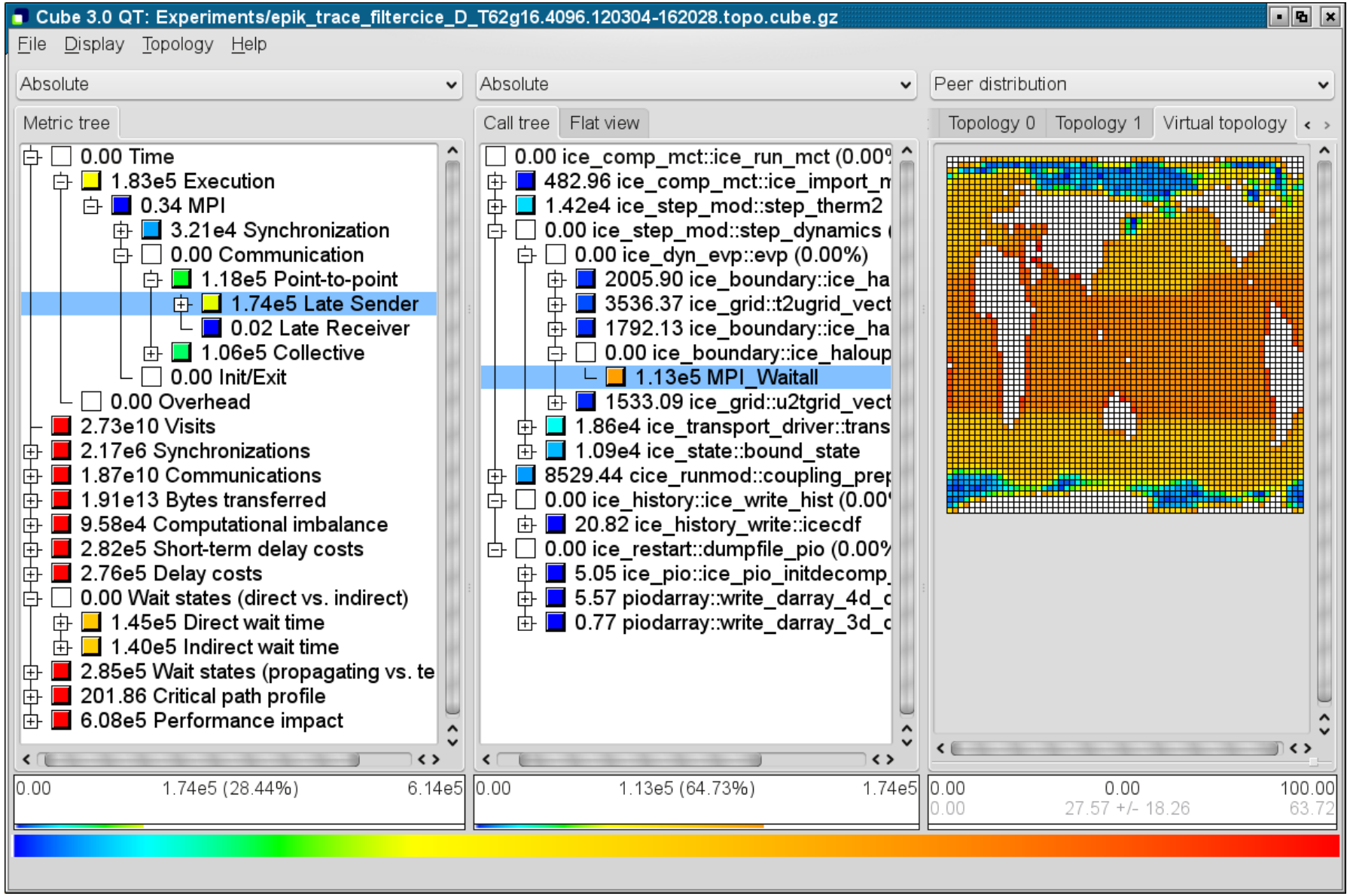


SCALASCA EXAMPLE: CESM SEA ICE MODULE

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

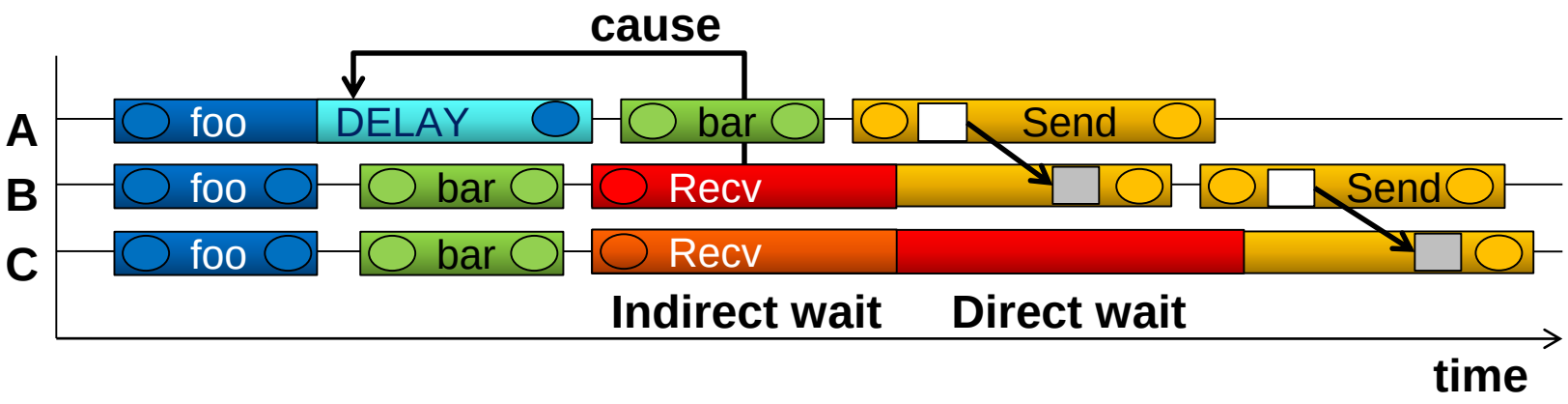
Late Sender Analysis + Application Topology

- Shows distribution of imbalance over topology
- MPI topologies are automatically captured



SCALASCA ROOT CAUSE ANALYSIS

- **Root-cause analysis**
  - Wait states typically caused by load or communication imbalances earlier in the program
  - Waiting time can also propagate (e.g., indirect waiting time)
  - Enhanced performance analysis to find the root cause of wait states
- **Approach**
  - Distinguish between direct and indirect waiting time
  - Identify call path/process combinations delaying other processes and causing first order waiting time
  - Identify original **delay**

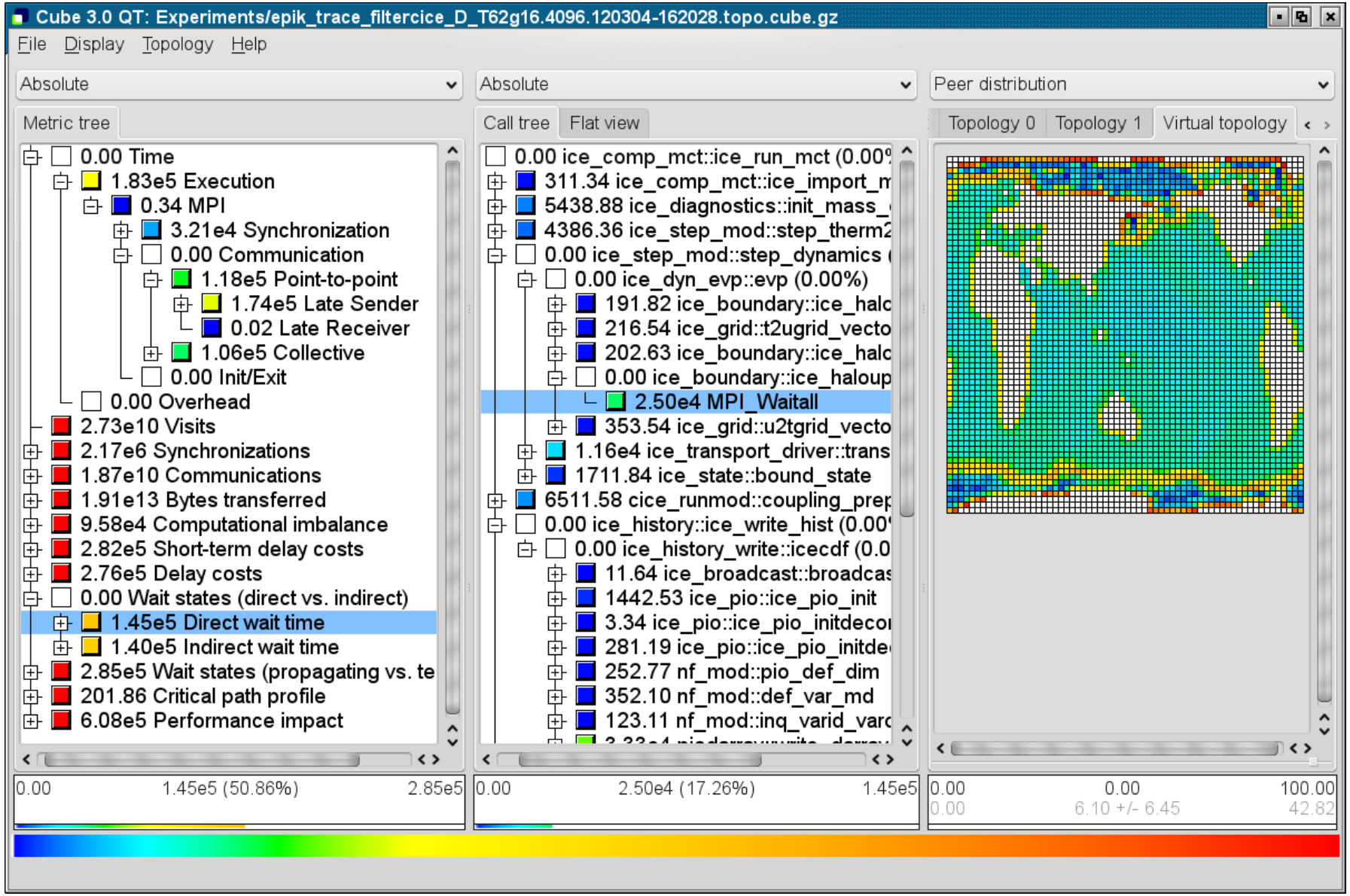


SCALASCA EXAMPLE: CESM SEA ICE MODULE

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Direct Wait Time Analysis

- Direct wait caused by ranks processing areas near the north and south ice borders

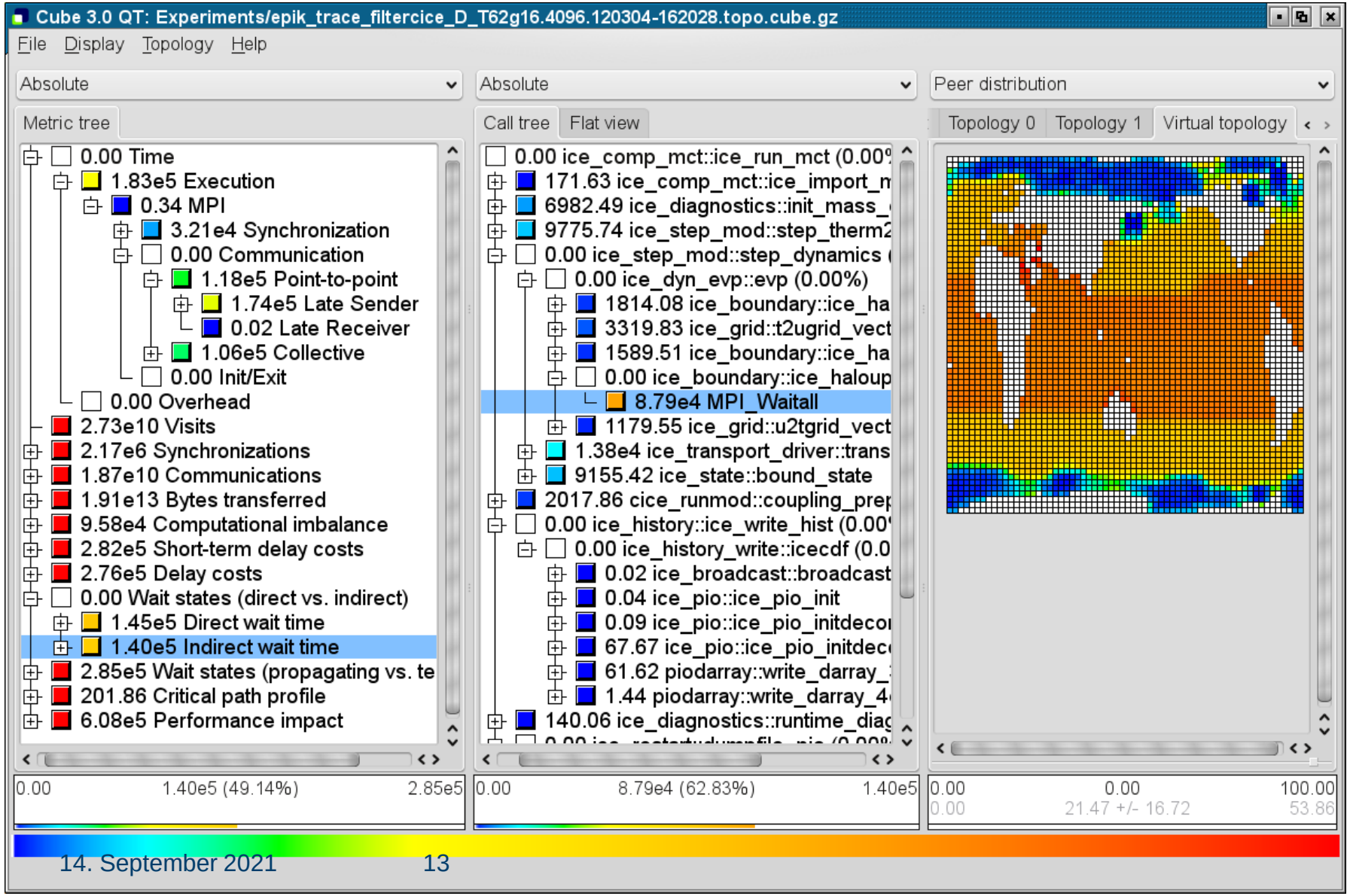


SCALASCA EXAMPLE: CESM SEA ICE MODULE

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Indirect Wait Time Analysis

- Indirect waits occurs for ranks processing warmer areas

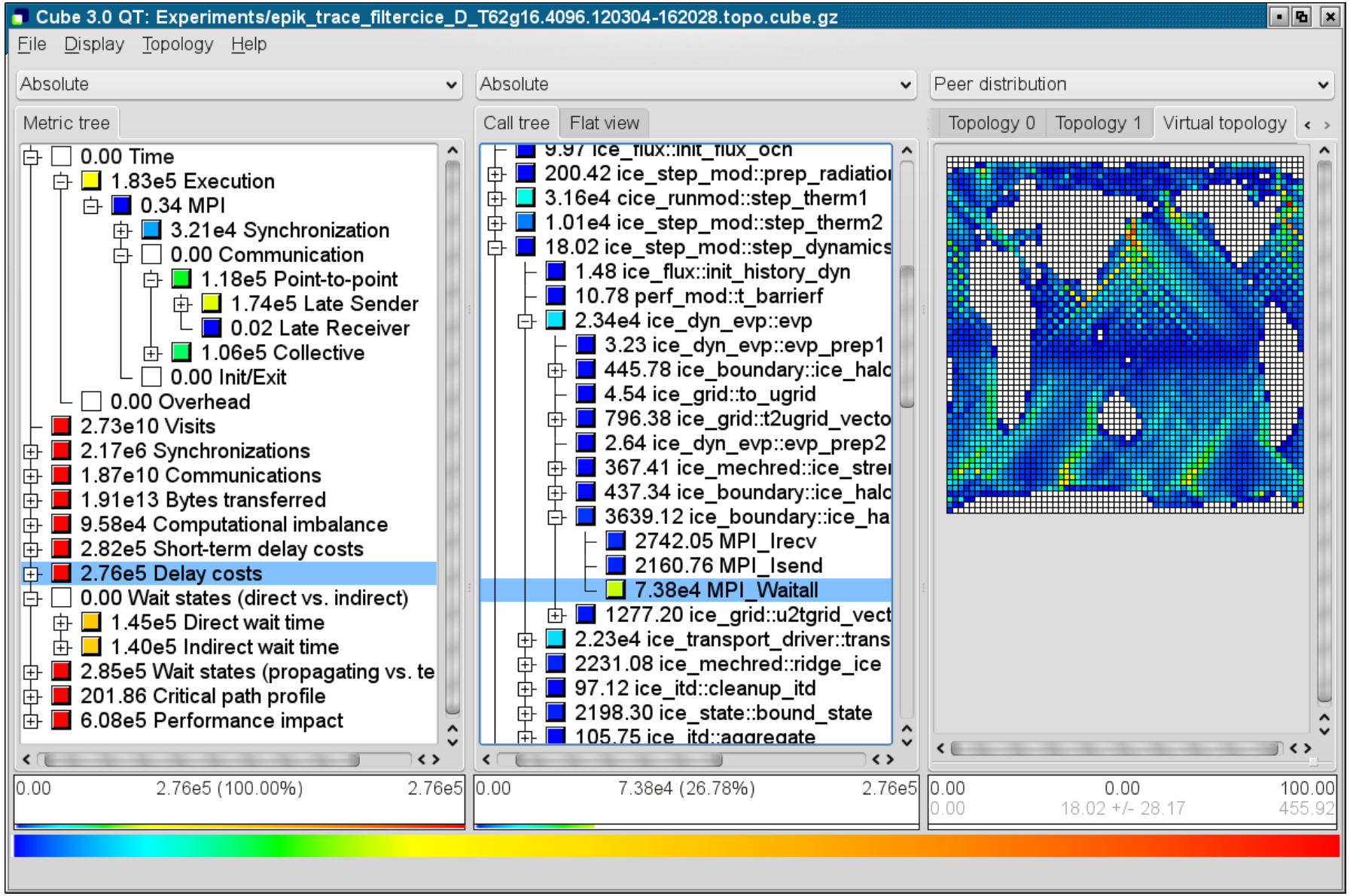


SCALASCA EXAMPLE: CESM SEA ICE MODULE

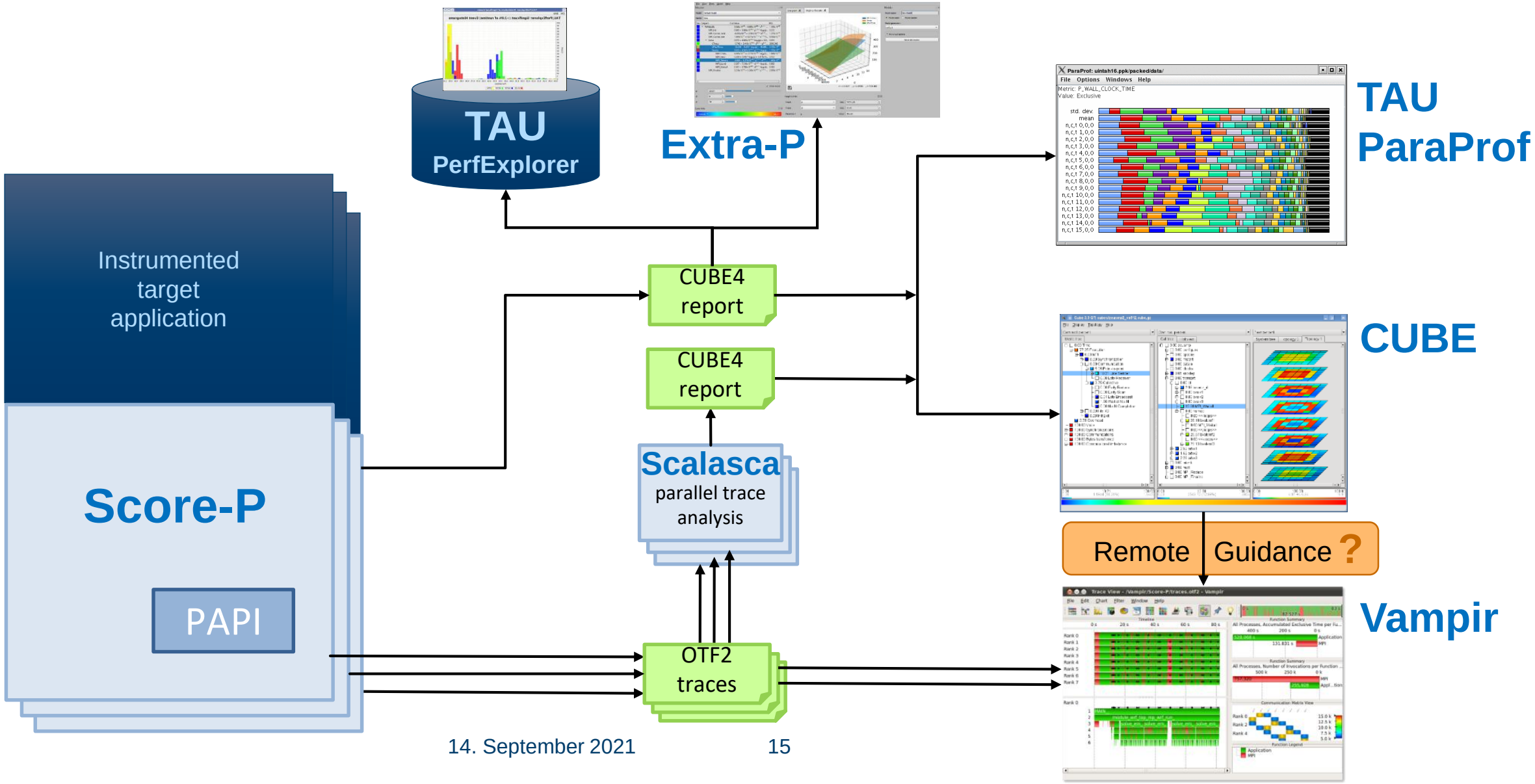
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Delay Costs Analysis

- Delays **NOT** caused on ranks processing ice!



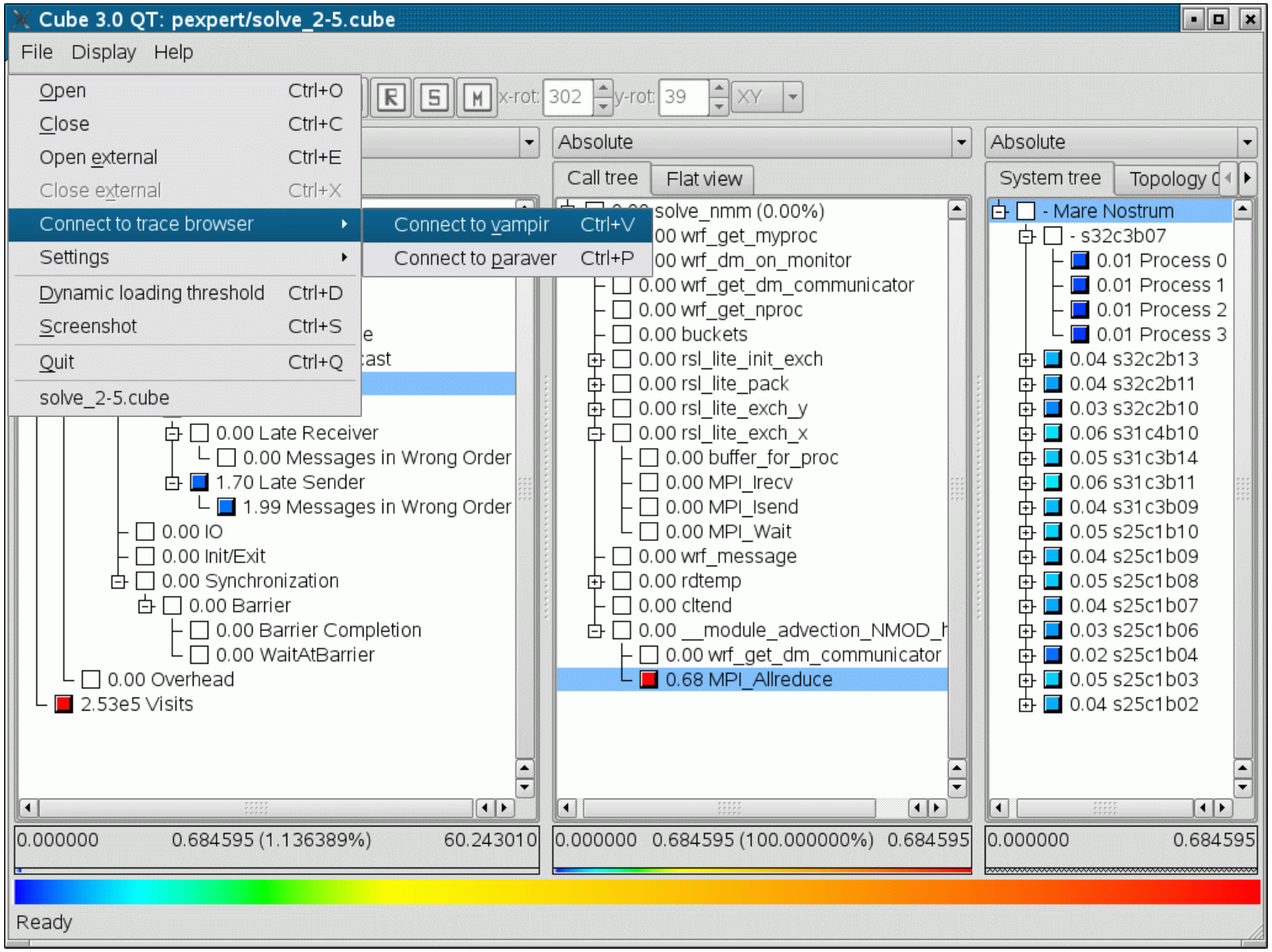
Score-P TOOL ECOSYSTEM



SCALASCA ⇨ VAMPIR INTEGRATION

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

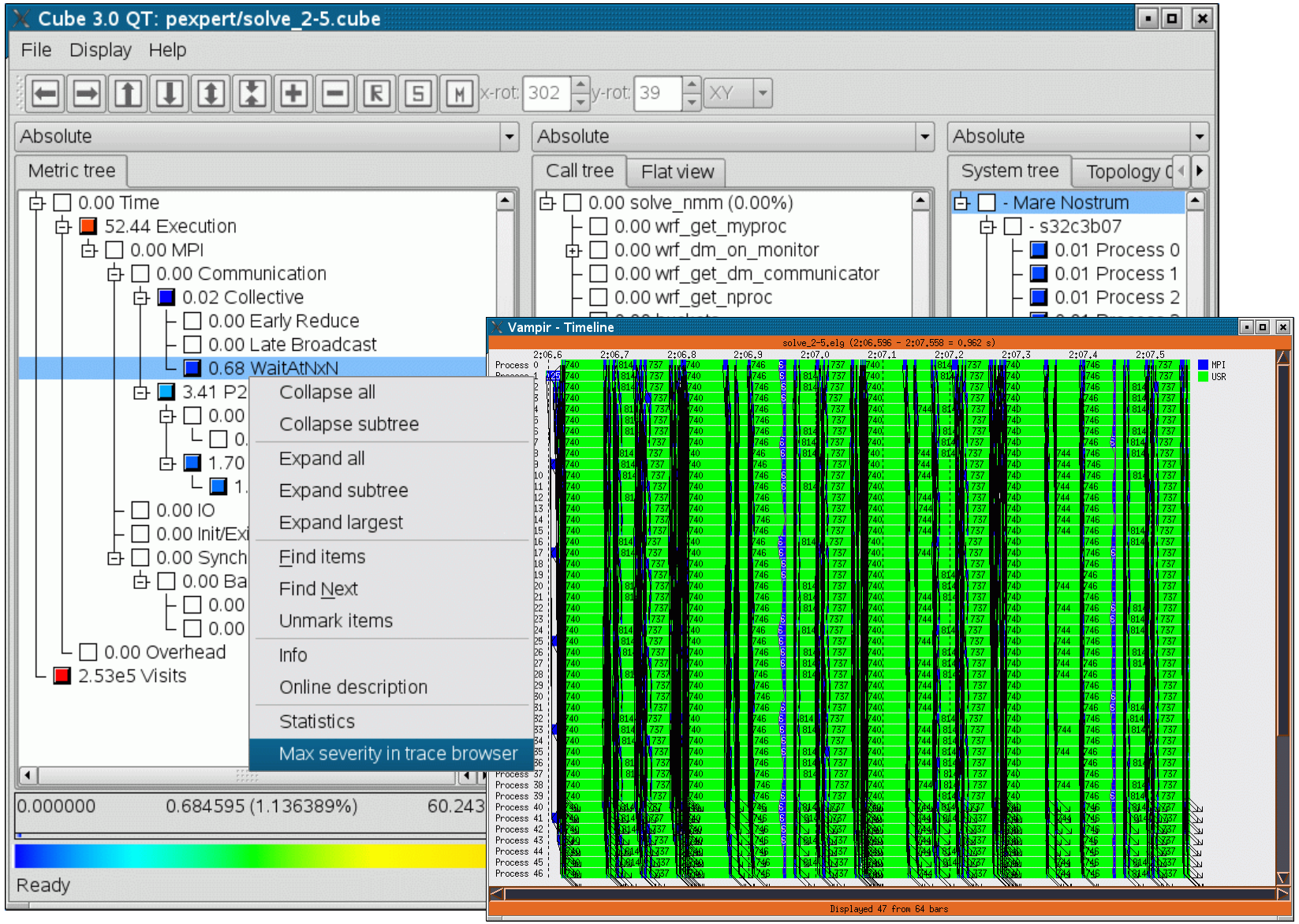
1. Connect to Vampir
   - Loads underlying trace



SCALASCA ⇨ VAMPIR INTEGRATION

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

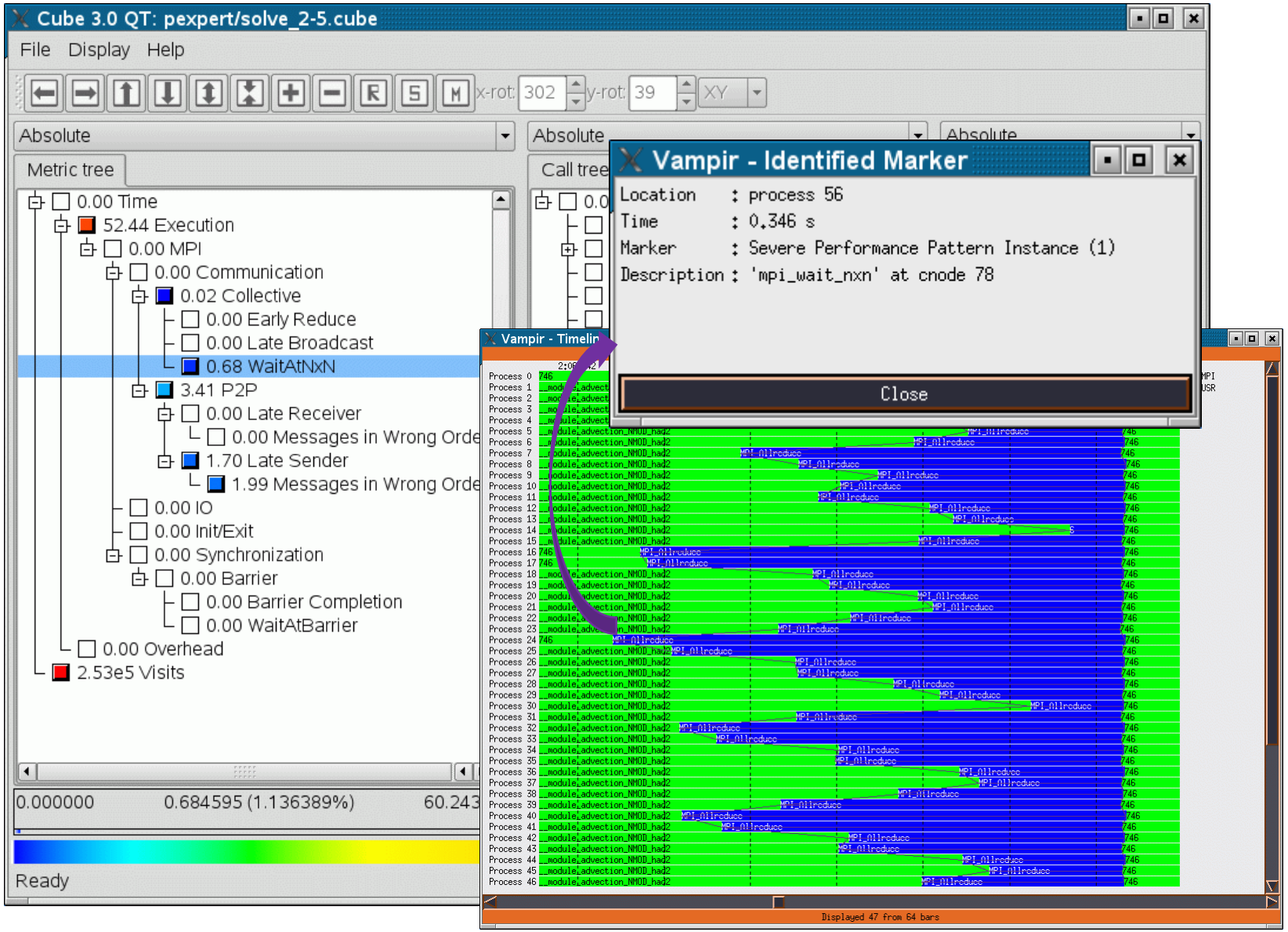
1. Connect to Vampir
   - Loads underlying trace
2. Use context menu
   - Max severity
   - Zooms to corresponding view



SCALASCA ⇨ VAMPIR INTEGRATION

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

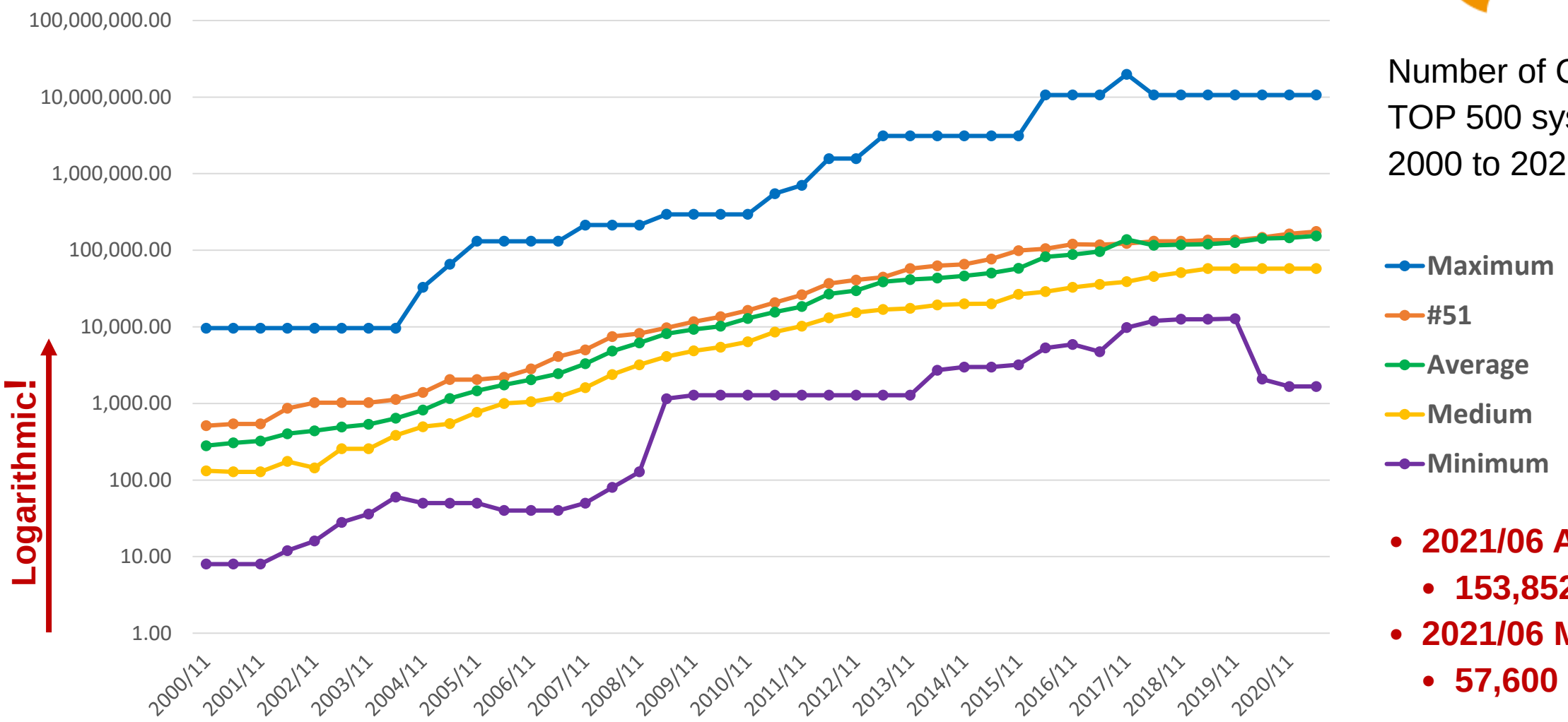
1. Connect to Vampir
   - Loads underlying trace
2. Use context menu
   - Max severity
   - Zooms to corresponding view
3. Use extensive Vampir features to investigate further



To infinity and beyond

EXTREME CONCURRENCY

TYPICAL HPC SYSTEM SIZE (NO. OF CORES)

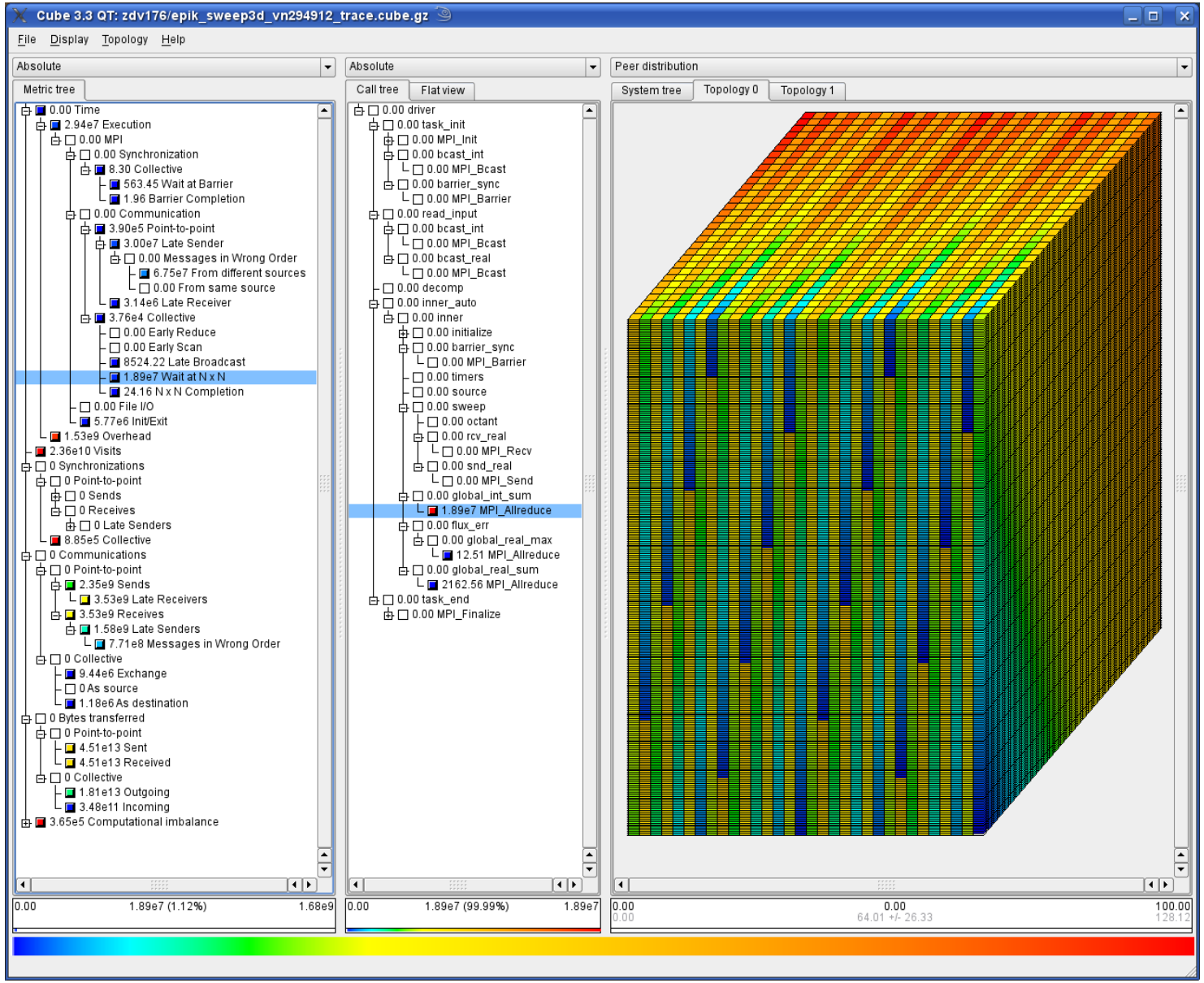


Number of Cores
TOP 500 systems
2000 to 2021

- **2021/06 Avg:**
  - **153,852**
- **2021/06 Median:**
  - **57,600**

SCALASCA TRACE ANALYSIS SWEEP3D@294,912 BGP

- 10 min sweep3D runtime
- 11 sec analysis
- 4 min trace data write/read (576 files)
- 7.6 TB buffered trace data
- 510 billion events



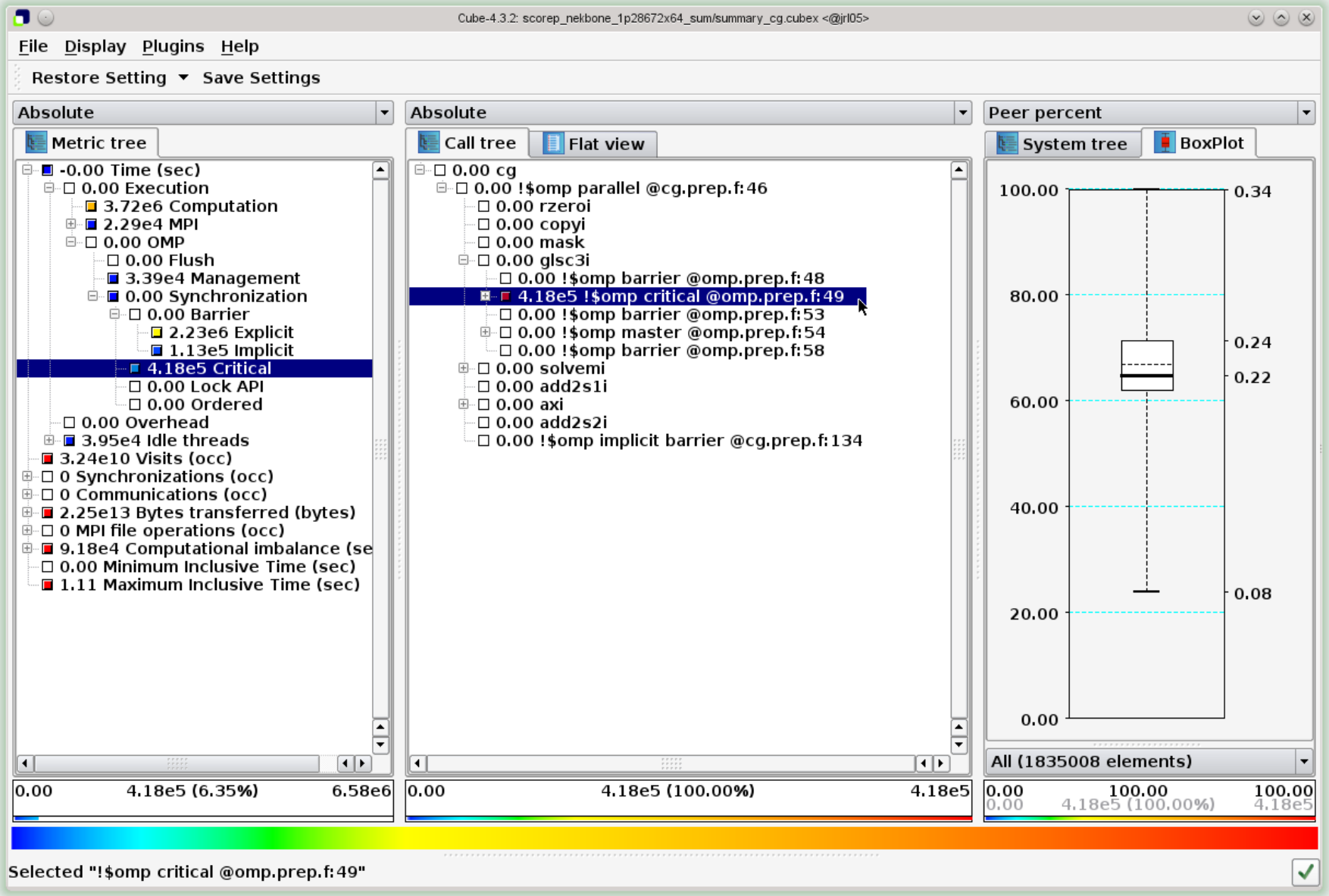
B. J. N. Wylie, M. Geimer, B. Mohr, D. Böhme, Z.Szebenyi, F. Wolf: Large-scale performance analysis of Sweep3D with the Scalasca toolset. Parallel Processing Letters, 20(4):397-414, 2010.

SCALASCA: 1,835,008 THREADS TEST CASE

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

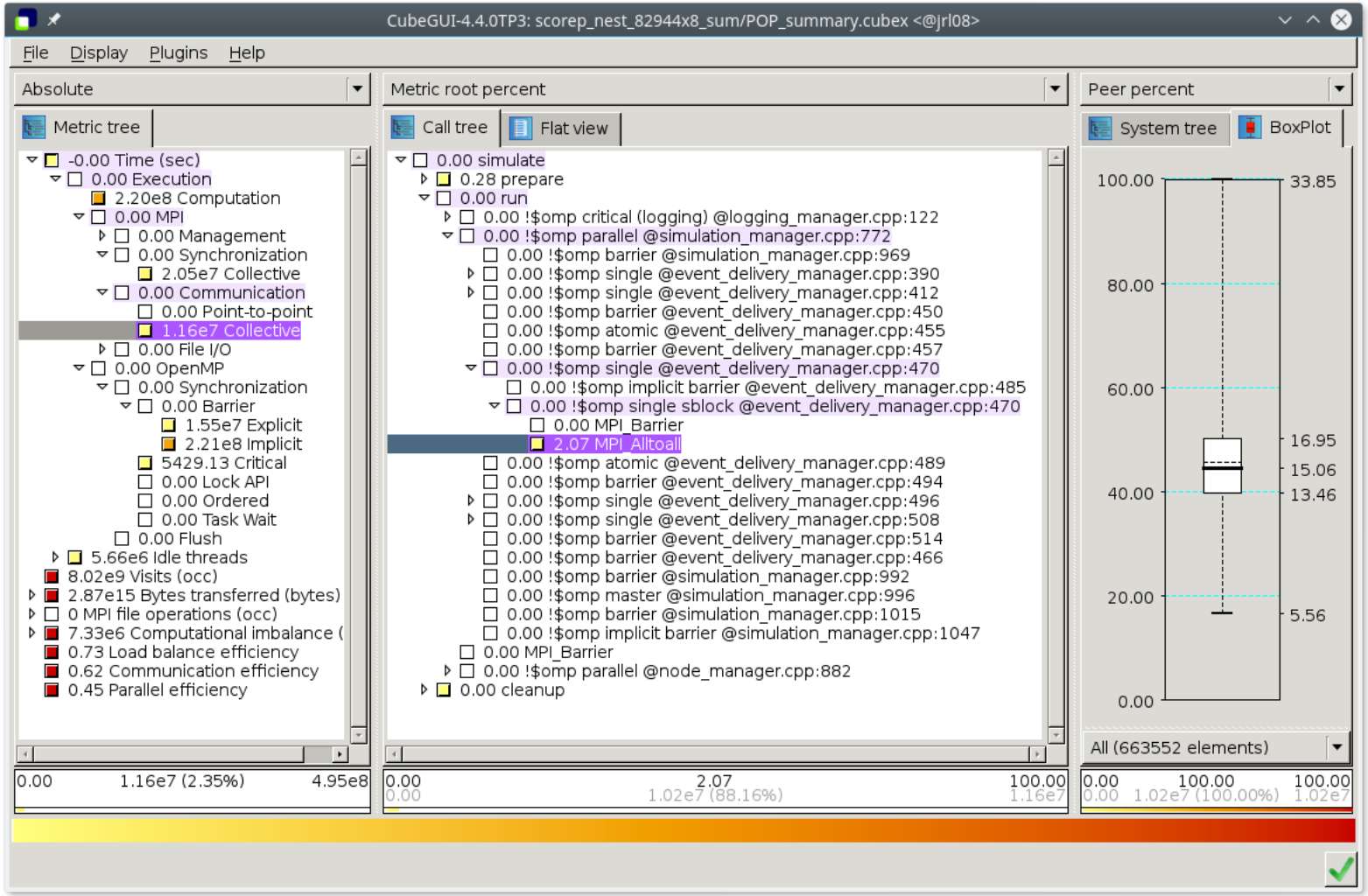
- Nekbone
- CORAL benchmark
- JuQueen experiment
- 28,672 x 64 = 1,835,008 threads

- Load imbalance at OpenMP critical section



SCALASCA: USER ANALYSIS OF NEST ON K COMPUTER

- Jülich nest :: neural network simulator code
- Measurement of **full system K computer run**
  - 82,944 nodes
  - 663,552 threads
- Performance analyst
  - Itaru Kitayama (RIKEN)
- Analysis of MPI and OpenMP communication and synchronization at large scale



JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

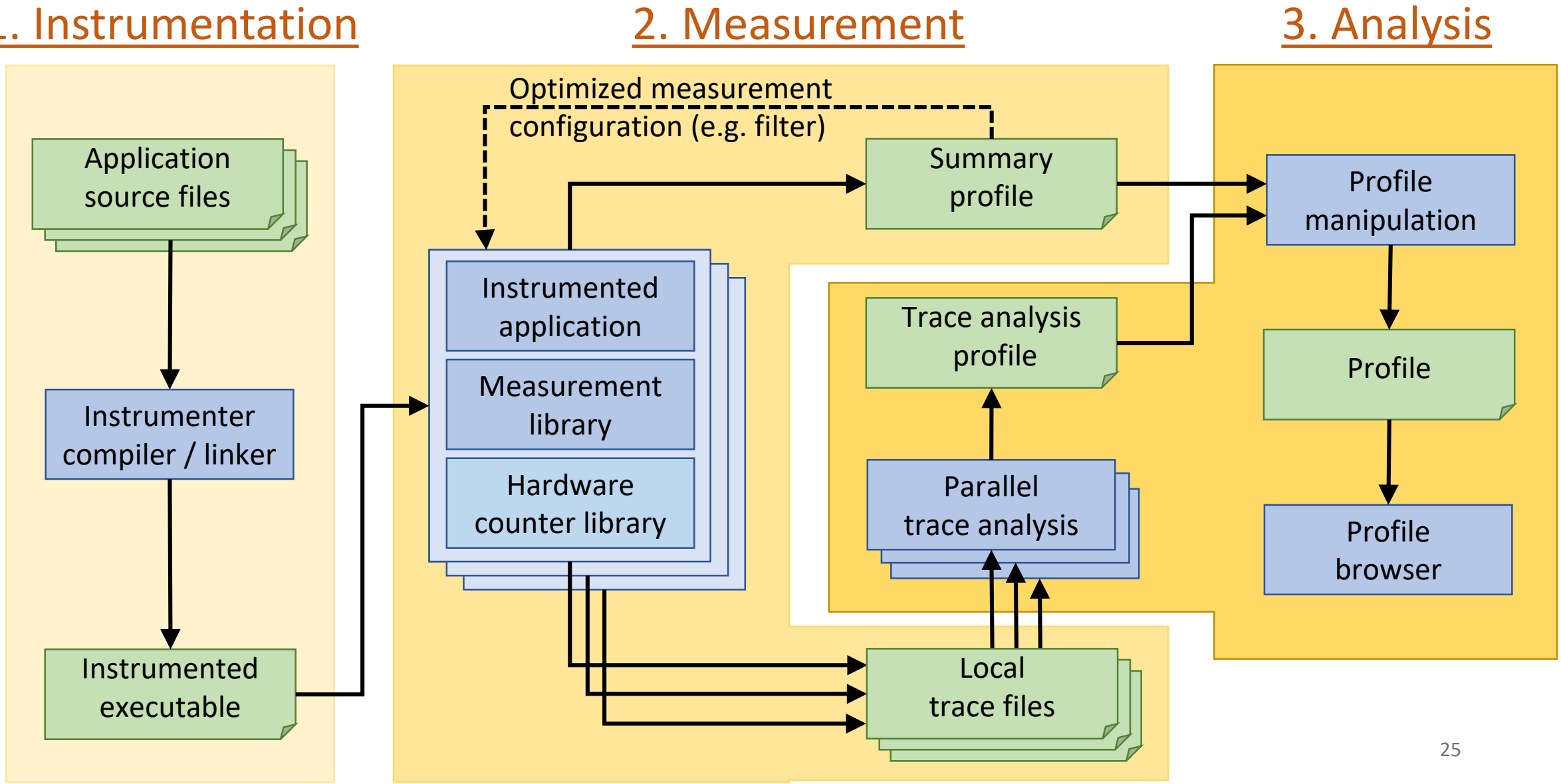
How To

PARALLEL APPLICATION PROFILE MEASUREMENT WITH SCORE-P

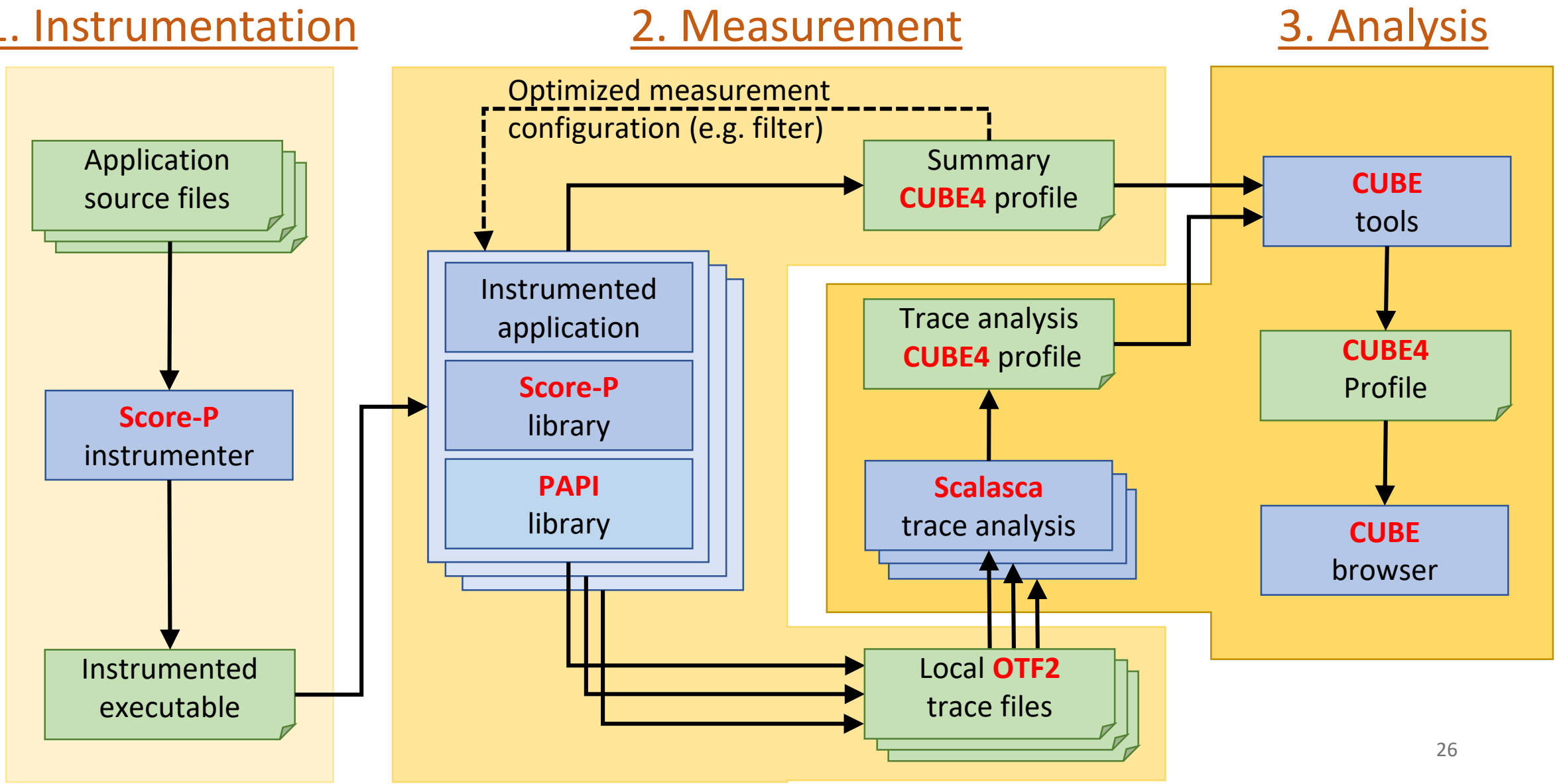
JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

PERFORMANCE ANALYSIS WORKFLOW

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE



PERFORMANCE ANALYSIS WORKFLOW



PERFORMANCE ANALYSIS WORKFLOW (FOR TRACING)

* part of Scalasca

- **Instrument ⇨ `scorep`**
  - Invoke compiler instrumentation including filter specification handling
  - Preprocess source files with opari (iff OpenMP)
  - Link MPI + I/O + … wrapper libraries and measurement core libraries
- **Measure and analyze traces ⇨ `scan*`**
  - Set necessary environment, e.g. to enable tracing, …
  - Make sure parallel execution command passes environment
  - Run instrumented application
  - In the same (batch) job, run scalasca trace analyzer with same number ranks/threads
- **Analyze results ⇨ `square*`**
  - Post-process cube result file (e.g. adding derived metrics and metrics hierarchy)
  - View cube file

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

SCORE-P / SCALASCA MEASUREMENT DATA

- All measured data and meta information stored in experiment directory
  - `scorep_<executable>_<size>_[sum|trace]`
  - Example: `scorep_tea_leaf_2p8x12_trace`

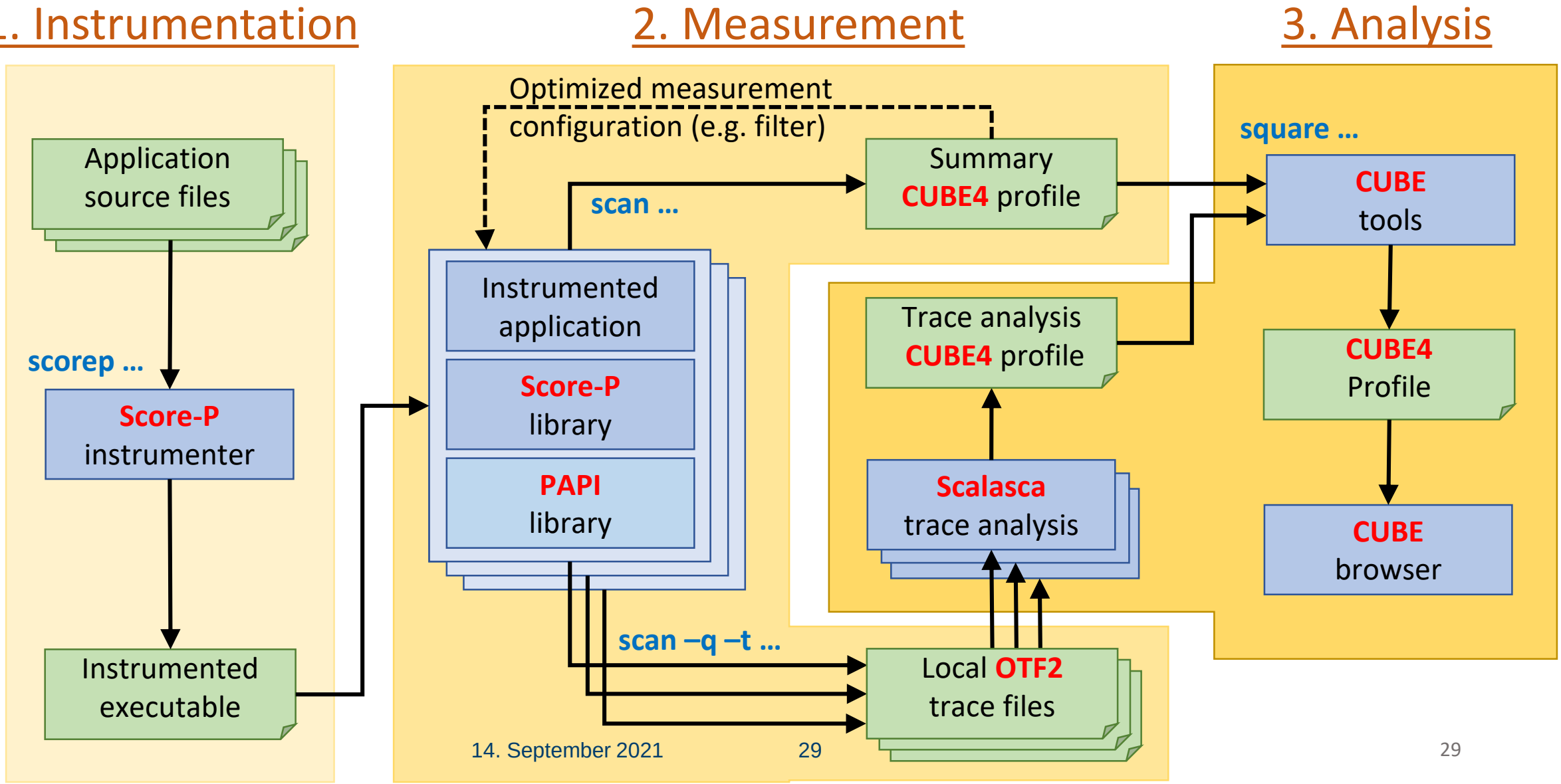
  - Contents
    - profile.cubex — Score-P profile measurement
    - summary.cubex — Post-processed profile measurement
    - scout.cubex — Scalasca trace analyzer result
    - trace.cubex — Post-processed trace analyzer result

    - Post-processed Cube files include
      - Additional derived metrics
      - Enhanced metrics hierarchy

JÜLICH Forschungszentrum JÜLICH SUPERCOMPUTING CENTRE

PERFORMANCE ANALYSIS WORKFLOW

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE



DEMO

- Measurement of simple Jacobi solver
  - Solves Poisson equation on rectangular grid assuming
    - Uniform discretization in each direction
    - Dirichlect boundary conditions

- Available in multiple variants
  - C, C++ or Fortran source code
  - MPI, OpenMP, or hybrid (MPI+OpenMP)

- Example code: https://pop-coe.eu/sites/default/files/pop_files/jacobi-example.zip

JÜLICH Forschungszentrum JÜLICH SUPERCOMPUTING CENTRE

DEMO: BASE RUN OF APPLICATION

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

openSUSE-Leap-15-1

```
zam310:~/jacobi/hybrid/C [1] make CC=mpicc CFLAGS=-fopenmp
mpicc -fopenmp -c jacobi.c
mpicc -fopenmp -c main.c
mpicc -fopenmp -o jacobi jacobi.o main.o -lm
zam310:~/jacobi/hybrid/C [2] export OMP_NUM_THREADS=2
zam310:~/jacobi/hybrid/C [3] mpiexec -np 2 ./jacobi
Jacobi 2 MPI-3.1#1 process(es) with 2 OpenMP-201511 thread(s)/process

-> matrix size: 2000×2000
-> alpha: 0.800000
-> relax: 1.000000
-> tolerance: 0.000000
-> iterations: 100

 Number of iterations : 100
 Residual             : 5.955111e-10
 Solution Error       : 0.000266483315
 Elapsed Time         : 3.2385783
 MFlops               : 1602.433142
zam310:~/jacobi/hybrid/C [4]
```

Notes

- Compile application
- Execute application with 2 threads on 2 processes
- Write down execution time for later comparison
- Pro Tip: run multiple times to check variability

DEMO: INSTRUMENT + PROFILE

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

openSUSE-Leap-15-1

```
zam310:~/jacobi/hybrid/C [4] make clean
rm -f jacobi jacobi.o main.o
zam310:~/jacobi/hybrid/C [5] export PATH=/opt/local/ScoreP-6.0/bin:/opt/local/Scalasca-2.5/bin:/opt/local/Cube-4.5/bi
n:${PATH}
zam310:~/jacobi/hybrid/C [6] make CC="scorep mpicc" CFLAGS=-fopenmp
scorep mpicc -fopenmp -c jacobi.c
scorep mpicc -fopenmp -c main.c
scorep mpicc -fopenmp -o jacobi jacobi.o main.o -lm
zam310:~/jacobi/hybrid/C [7] scan mpiexec -np 2 ./jacobi
S=C=A=N: Scalasca 2.5 runtime summarization
S=C=A=N: ./scorep_jacobi_2x2_sum experiment archive
S=C=A=N: Mon May 25 16:28:55 2020: Collect start
/opt/local/easybuild-4.1.1/software/OpenMPI/3.1.4-GCC-system-2.31/bin/mpiexec -np 2 ./jacobi
Jacobi 2 MPI-3.1#1 process(es) with 2 OpenMP-201511 thread(s)/process

-> matrix size: 2000×2000
-> alpha: 0.800000
-> relax: 1.000000
-> tolerance: 0.000000
-> iterations: 100

 Number of iterations : 100
 Residual             : 5.955111e-10
 Solution Error       : 0.000266483315
 Elapsed Time         : 3.3544007
 MFlops               : 1547.103541
S=C=A=N: Mon May 25 16:28:59 2020: Collect done (status=0) 4s
S=C=A=N: ./scorep_jacobi_2x2_sum complete.
zam310:~/jacobi/hybrid/C [8]
```

Notes

- Make sure tools are in $PATH
- Instrument: prepend `scorep`
- Measure profile: prepend `scan`
- Compare execution time to check overhead
- Profile in `scorep_APP_SIZE_sum` subdirectory

DEMO: OPTIMIZE MEASUREMENT CONFIGURATION (SCORING)

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

openSUSE-Leap-15-1

```
zam310:~/jacobi/hybrid/C [8] square -s ./scorep_jacobi_2x2_sum/
INFO: Post-processing runtime summarization report (profile.cubex)...
/opt/local/ScoreP-6.0/bin/scorep-score -r ./scorep_jacobi_2x2_sum/profile.cubex > ./scorep_jacobi_2x2_sum/scorep.scor
e
INFO: Score report written to ./scorep_jacobi_2x2_sum/scorep.score
zam310:~/jacobi/hybrid/C [9] head -25 ./scorep_jacobi_2x2_sum/scorep.score

Estimated aggregate size of event trace:                   179kB
Estimated requirements for largest trace buffer (max_buf): 90kB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       7MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=7MB to avoid intermediate flushes
 or reduce requirements using USR regions filters.)

flt     type max_buf[B] visits time[s] time[%] time/visit[us]  region
         ALL     91,341  3,840   13.40   100.0        3488.63  ALL
         OMP     61,078  2,812   12.76    95.2        4536.48  OMP
         MPI     27,440    812    0.55     4.1         683.27  MPI
         COM      2,756    212    0.08     0.6         400.31  COM
      SCOREP         41      2    0.00     0.0          18.35  SCOREP
         USR         26      2    0.00     0.0          17.05  USR

         OMP     17,400    400    0.00     0.0           2.09  !$omp parallel @jacobi.c:61
         OMP     17,400    400    0.00     0.0           1.55  !$omp parallel @jacobi.c:148
         MPI      8,900    200    0.00     0.0           7.43  MPI_Irecv
         MPI      8,900    200    0.00     0.0           4.51  MPI_Isend
         MPI      6,800    200    0.33     2.5        1663.23  MPI_Allreduce
         OMP      5,200    400    0.96     7.1        2389.23  !$omp implicit barrier @jacobi.c:79
         OMP      5,200    400    2.09    15.6        5223.62  !$omp for @jacobi.c:148
         OMP      5,200    400    0.22     1.6         538.58  !$omp implicit barrier @jacobi.c:155
         OMP      5,200    400    9.31    69.5       23269.84  !$omp for @jacobi.c:64
         OMP      5,200    400    0.00     0.0           1.73  !$omp implicit barrier @jacobi.c:80
zam310:~/jacobi/hybrid/C [10]
```

Notes

- Optimize measurement config: scoring with `square -s`
- Also does post-processing
- Potential need for filtering → see user guides
- Set `SCOREP_TOTAL_MEMORY`

DEMO: TRACE + ANALYZE

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

openSUSE-Leap-15-1

```
zam310:~/jacobi/hybrid/C [10] export SCOREP_TOTAL_MEMORY=10MB
zam310:~/jacobi/hybrid/C [11] scan -q -t mpiexec -np 2 ./jacobi
S=C=A=N: Scalasca 2.5 trace collection and analysis
S=C=A=N: ./scorep_jacobi_2×2_trace experiment archive
S=C=A=N: Mon May 25 16:32:12 2020: Collect start
/opt/local/easybuild-4.1.1/software/OpenMPI/3.1.4-GCC-system-2.31/bin/mpiexec -np 2 ./jacobi
Jacobi 2 MPI-3.1#1 process(es) with 2 OpenMP-201511 thread(s)/process

-> matrix size: 2000×2000
-> alpha: 0.800000
-> relax: 1.000000
-> tolerance: 0.000000
-> iterations: 100

 Number of iterations : 100
 Residual             : 5.955111e-10
 Solution Error       : 0.000266483315
 Elapsed Time         : 3.2049717
 MFlops               : 1619.235889
S=C=A=N: Mon May 25 16:32:17 2020: Collect done (status=0) 5s
S=C=A=N: Mon May 25 16:32:17 2020: Analyze start
/opt/local/easybuild-4.1.1/software/OpenMPI/3.1.4-GCC-system-2.31/bin/mpiexec -np 2 /opt/local/Scalasca-2.5/bin/scout
.hyb ./scorep_jacobi_2×2_trace/traces.otf2
SCOUT   (Scalasca 2.5)
Copyright (c) 1998-2019 Forschungszentrum Juelich GmbH
Copyright (c) 2009-2014 German Research School for Simulation Sciences GmbH

Analyzing experiment archive ./scorep_jacobi_2×2_trace/traces.otf2

Opening experiment archive ... done (0.000s).
Reading definition data    ... done (0.001s).
Reading event trace data   ... done (0.015s).
Preprocessing              ... done (0.001s).
Analyzing trace data       ... done (0.026s).
```

Notes

- Measure trace: prepend `scan`
  - –q: profile off
  - –t: trace on
- After trace measurement, Scalasca trace analyzer runs automatically
- Traces and enhanced profile in `scorep_APP_SIZE_trace` subdirectory

DEMO

- TeaLeaf Reference V1.0
- HPC mini-app developed by the UK Mini-App Consortium
  - Solves the linear 2D heat conduction equation on a spatially decomposed regular grid using a 5 point stencil with implicit solvers
  - https://github.com/UK-MAC/TeaLeaf_ref/archive/v1.0.tar.gz
- Measurements performed on Jusuf cluster @ JSC
  - Run configuration: 32 MPI ranks with 8 OpenMP threads each (across 2 nodes)
    - Test problem “5”: 4000 × 4000 cells, CG solver
- https://pop-coe.eu/sites/default/files/pop_files/scorep_tea_leaf_16p32x8_multi-run_c2.zip

JÜLICH Forschungszentrum JÜLICH SUPERCOMPUTING CENTRE

DEMO

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

File Display Plugins Help

Absolute | Metric selection percent | Metric selection percent

Metric tree | Call tree | Flat view | System tree | Statistics | Sunburst

- 0.00 Time (sec)
  - 0.00 Execution
    - 8544.65 Computation
    - 0.00 MPI
      - 3.44 Management
      - 0.09 Synchronization
      - 0.00 Communication
        - - Point-to-point
        - - Collective
        - 0.00 One-sided
      - 0.00 File I/O
    - 518.41 OpenMP
  - 0.00 Overhead
  - 856.84 Idle threads
- 1.18e8 Visits (occ)
- 72 MPI synchronizations (occ)
- 0 MPI pair-wise one-sided synchronizations (occ)
- 1.29e6 MPI communications (occ)
- 0 MPI file operations (occ)
- 2.37e10 MPI bytes transferred (bytes)
- 1028.61 Delay costs (sec)
- 4.05 MPI point-to-point wait states (propagating vs. terminal)
- 4.05 MPI point-to-point wait states (direct vs. indirect) (sec)
- 103.99 Critical path (sec)
- 9982.93 Performance impact (sec)
- 264.86 Computational imbalance (sec)
- 0.00 Minimum Inclusive Time (sec)
- 103.90 Maximum Inclusive Time (sec)
- 0 ALLOCATION_SIZE (bytes)

0.00 | 8544.65 (100.00%) | 8544.65

- 0.00 tea_leaf.inst
  - 0.00 tea_leaf
    - 0.00 tea_init_comms
    - 0.00 !$omp parallel @tea_leaf.f90:45
    - 0.02 initialise
    - 0.00 diffuse
      - 0.00 timer
      - 0.00 set_field
      - 0.00 timestep
      - 0.04 tea_leaf
        - 0.02 timer
        - 2.57 update_halo
        - 0.08 tea_leaf_kernel_init_cg_fortran
        - 0.02 tea_allsum
        - 0.01 tea_leaf_kernel_cheby_copy_u
        - 0.01 tea_leaf_kernel_solve_cg_fortran_calc_w
          - 39.02 !$omp parallel @tea_leaf_cg.f90:186
        - 0.01 tea_leaf_kernel_solve_cg_fortran_calc_ur
          - 38.89 !$omp parallel @tea_leaf_cg.f90:234
        - 0.01 tea_leaf_kernel_solve_cg_fortran_calc_p
          - 19.29 !$omp parallel @tea_leaf_cg.f90:284
        - 0.01 tea_leaf_kernel_finalise
      - 0.00 field_summary
      - 0.00 tea_allgather
      - 0.00 tea_finalize

0.00 | 97.19 | 100.00
0.00 | 8304.91 (97.19%) | 8544.65

- machine Linux
  - node jrc0036
    - 12.07 MPI Rank 0
    - 12.14 MPI Rank 1
  - node jrc0037
    - 12.07 MPI Rank 2
    - 12.19 MPI Rank 3
  - node jrc0038
    - 12.12 MPI Rank 4
    - 12.22 MPI Rank 5
  - node jrc0039
    - 12.24 MPI Rank 6
    - 12.13 MPI Rank 7

All (96 elements)

0.00 | 100.00
0.00 | 0.00 (0.00%) | 8304.91

System View | Topologies | Other

FURTHER USEFUL INFORMATION

Extended and more detailed example based on NAS Parallel Benchmark (NPB) BT-MZ

- Scalasca documentation
  - A full workflow example
- Score-P documentation
  - Performance Analysis Workflow Using Score-P
- Slides from 40th VI-HPS Tuning Workshop
  - Score-P instrumentation & measurement toolset
  - Score-P analysis scoring & measurement filtering
  - Score-P specialized instrumentation and measurement (Advanced)
  - Scalasca automated trace analysis

JÜLICH Forschungszentrum JÜLICH SUPERCOMPUTING CENTRE

Get Started

TOOL SOFTWARE INSTALLATION

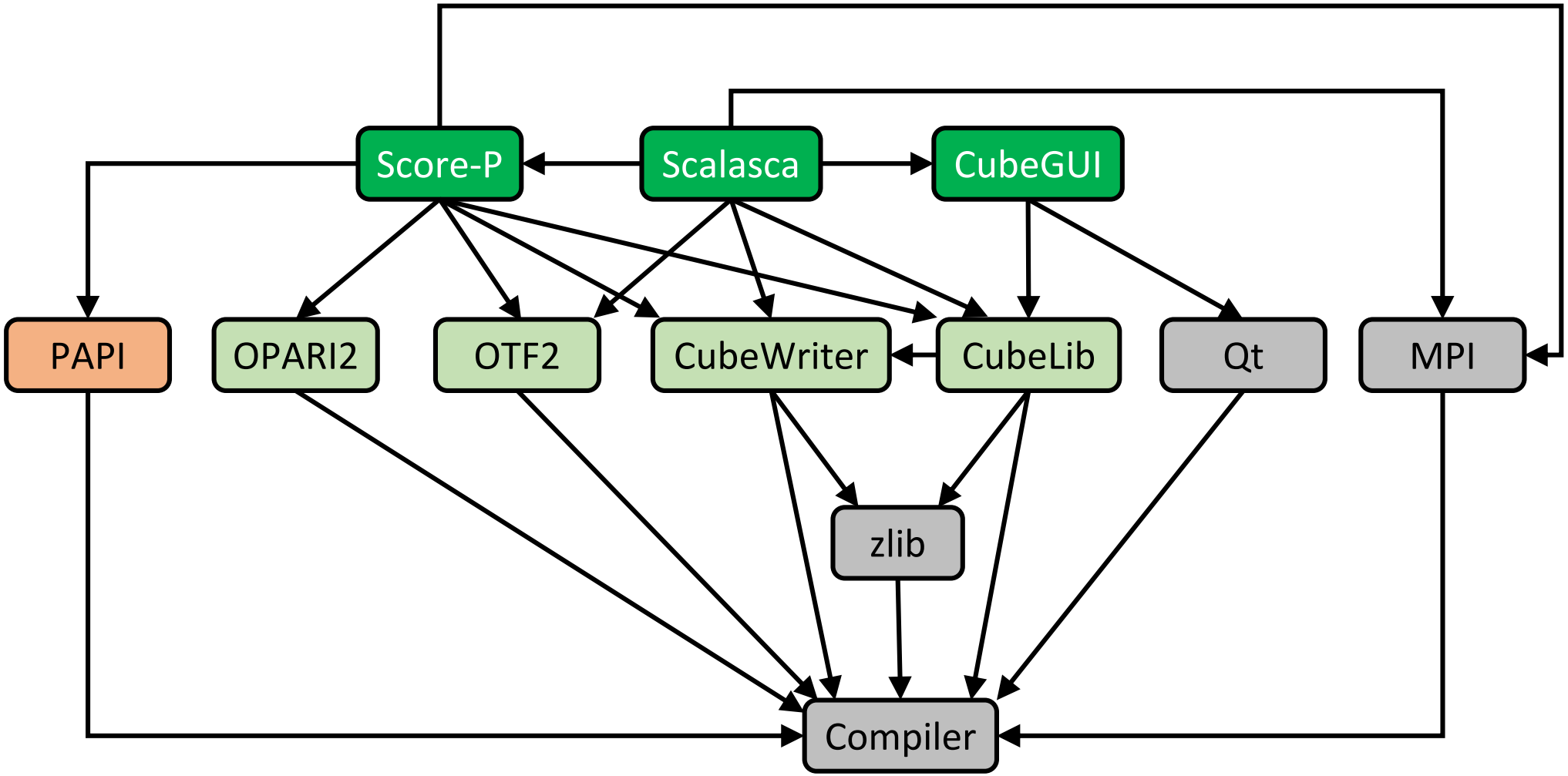
PREREQUISITES

- Basic understanding of Linux shell commands
- Access to HPC cluster or Linux workstation/laptop

- **Development software**
  - Compiler suite (C, C++, Fortran)
    - GCC, Intel, IBM XL, PGI, …
  - MPI library
    - OpenMPI, MPICH, Intel, …
  - Cube only: QT4 (≥ 4.6.0) or Qt5
  - make

JÜLICH Forschungszentrum JÜLICH SUPERCOMPUTING CENTRE

PACKAGE DEPENDENCIES



MANUAL INSTALLATION

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Install Cube
- Download
- Unpack
- Configure
- Compile
- Install

cube scalasca

Install Score-P
- Download, …

Score-P

Install Scalasca
- Download, …

scalasca

Personal Desktop / Laptop

Remote HPC Cluster

Local HPC Cluster / Desktop / Laptop

Pro Tip: Order important so already installed subcomponents can be re-used

MANUAL INSTALLATION: cube scalasca

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

- Download latest version
  - http://www.scalasca.org/scalasca/software/cube-4.x/
  - CubeBundle 4.6
- Unpack

```
% tar zxf CubeBundle-4.6.tar.gz && cd CubeBundle-4.6
```

- Configure

```
% ./configure --prefix=/opt/local/Cube-4.6
```

or
`$HOME/tools/Cube-4.6`

- Compile

```
% make
```

- Install

```
% make install
```

- **Pro Tip:** Website also provides binary installers for Windows and MAC OS

MANUAL INSTALLATION: Score-P

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

- Download latest version
  - http://www.score-p.org (scroll down to “Download section”)
  - Score-P 7.1
- Make sure to re-use Cube subcomponents (Local Scenario only)

```
% export PATH=/opt/local/Cube-4.6/bin:${PATH}
```

- Unpack

```
% tar zxf scorep-7.1.tar.gz && cd scorep-7.1
```

- Configure

```
% ./configure --prefix=/opt/local/ScoreP-7.1
```

- Compile

```
% make
```

- Install

```
% make install
```

MANUAL INSTALLATION: Score-P (2)

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

- **Note:** Score-P is Compiler and MPI dependent
  - Needs to be installed for every Compiler/MPI combination

- Advanced configuration
  - Specify compiler suite other than GCC

```
% ./configure ... --with-nocross-compiler-suite=(ibm|intel|pgi)
```

  - Specify MPI library if NOT auto-detected or more than one MPI library available

```
% ./configure ... --with-mpi=(intel3|mpich3|openmpi3|…)
```

  - Specify PAPI component if NOT auto-detected (needed for POP metrics calculation)

```
% ./configure ... --with-papi-header=<path-to-papi.h>
                  --with-papi-lib=<path-to-libpapi.*>
```

- See installation guide on how to configure support for additional programming models (CUDA, OpenCL, SHMEM, OpenACC)

MANUAL INSTALLATION: scalasca

- Download latest version
  - https://www.scalasca.org/scalasca/software/scalasca-2.x/
  - Scalasca 2.5
- Make sure to re-use Score-P subcomponents (e.g. OTF2)

```
% export PATH=/opt/local/ScoreP-6.0/bin:${PATH}
```

- Unpack

```
% tar zxf scalasca-2.5.tar.gz && cd scalasca-2.5
```

- Configure

```
% ./configure --prefix=/opt/local/Scalasca-2.5
```

- Compile

```
% make
```

- Install

```
% make install
```

MANUAL INSTALLATION: scalasca (2)

- **Note:** Scalasca is Compiler and MPI dependent
  - Needs to be installed for every Compiler/MPI combination

- Advanced configuration
  - Specify compiler suite other than GCC

```
% ./configure ... --with-nocross-compiler-suite=(ibm|intel|pgi)
```

  - Specify MPI library if NOT auto-detected or more than one MPI library available

```
% ./configure ... --with-mpi=(intel3|mpich3|openmpi3|…)
```


JÜLICH Forschungszentrum JÜLICH SUPERCOMPUTING CENTRE

AUTOMATED INSTALLATION

- Use IF
  - You need to install tools for multiple Compiler/MPI library combinations
  - You need to install also other HPC tools, libraries, and applications
  - You easily want to maintain and update a HPC software stack

- HPC package managers
  - EasyBuild (UGhent, source)
    - https://easybuild.readthedocs.io/
  - Spack (LLNL, source)
    - https://spack.io/
  - OpenHPC (Linux Foundation, binary)
    - https://openhpc.community/

JÜLICH Forschungszentrum JÜLICH SUPERCOMPUTING CENTRE

AUTOMATED INSTALLATION

- EasyBuild (<tool> = Scalasca|Score-P|CubeGUI)
  - Search for suitable easyconfigs: `eb -S <tool>`
  - Copy best matching easyconfig and adapt desired version and toolchain
  - Install: `eb <tool>-<version>-<toolchain>.eb`

- Spack (<tool> = scalasca|scorep|cube)
  - Install: `spack install <tool>@<version> %<compiler> ^<mpi-library>`

- OpenHPC (<tool> = scalasca|scorep)
  - Use Linux package manager (zypper, yum, …) to install suitable RPM
  - E.g. `zypper install <tool>-<compiler>-<mpi>-<ohpcversion>-<arch>.rpm`

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

QUESTIONS?

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

scalasca

- http://www.scalasca.org
- scalasca@fz-juelich.de

Score-P
Scalable performance measurement infrastructure for parallel codes

- http://www.score-p.org
- support@score-p.org