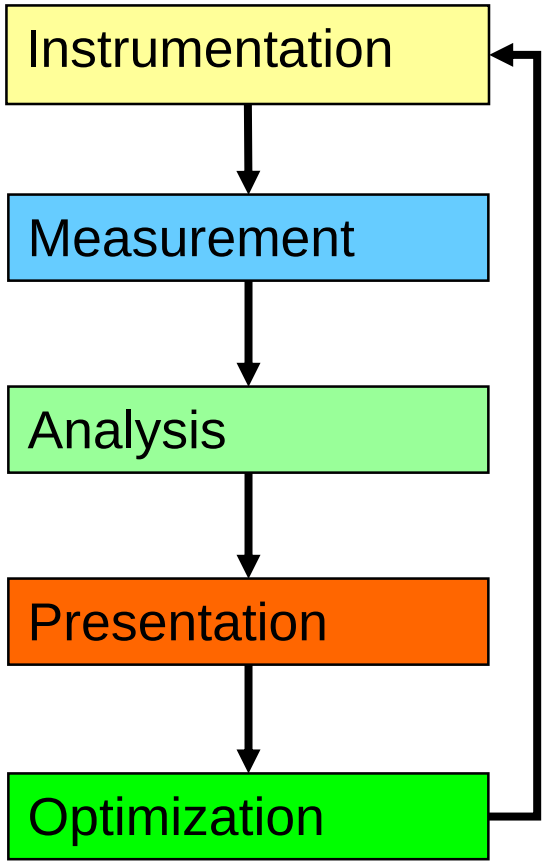
THE SCORE-P ECOSYSTEM
PROFILING WITH SCORE-P AND CUBE

SEP 7, 2018 I BERND MOHR

Background

PARALLEL PERFORMANCE TOOLS 101

PERFORMANCE MEASUREMENT CYCLE



- Insertion of extra code (probes, hooks) into application
- Collection of data relevant to performance analysis
- Calculation of metrics, identification of performance problems
- Transformation of the results into representation that can be easily understood by a human user
- Elimination of performance problems (**Left to User!**)

PERFORMANCE MEASUREMENT

Two dimensions

When performance measurement is triggered

- **External trigger** (asynchronous)
  - **Sampling**
    - Trigger: Timer interrupt OR Hardware counters overflow
- **Internal trigger** (synchronous)
  - Code **instrumentation** (automatic or manual)

How performance data is recorded

- **Profile**
  - Summation of events over time
- **Trace**
  - Sequence of events over time

MEASUREMENT METHODS: PROFILING

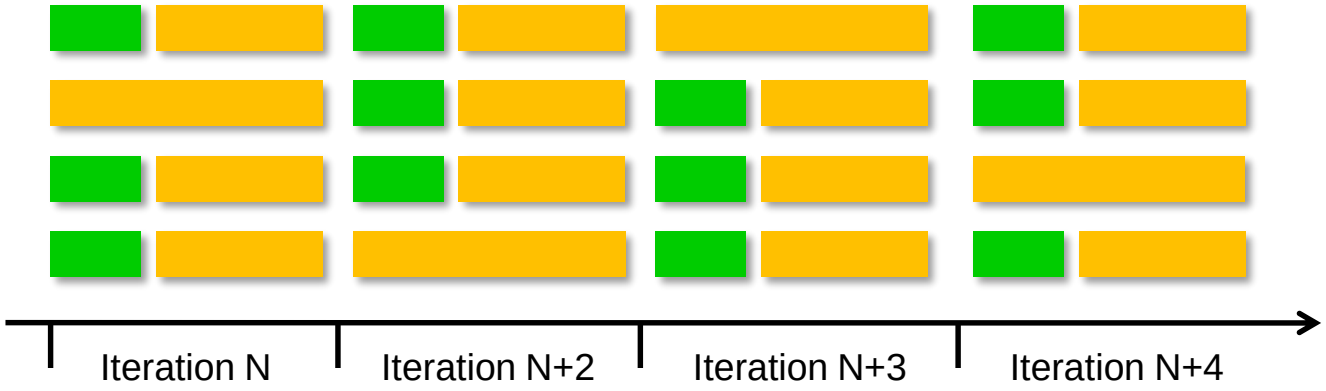
- Recording of **aggregated information**
  - Time
  - Counts
    - Calls
    - Hardware counters
- **about program and system entities**
  - Functions, call sites, loops, basic blocks, …
  - Processes, threads
- **Statistical information**
  - Min, max, mean and total number of values

Advantages
+ **Works also for long-running programs**

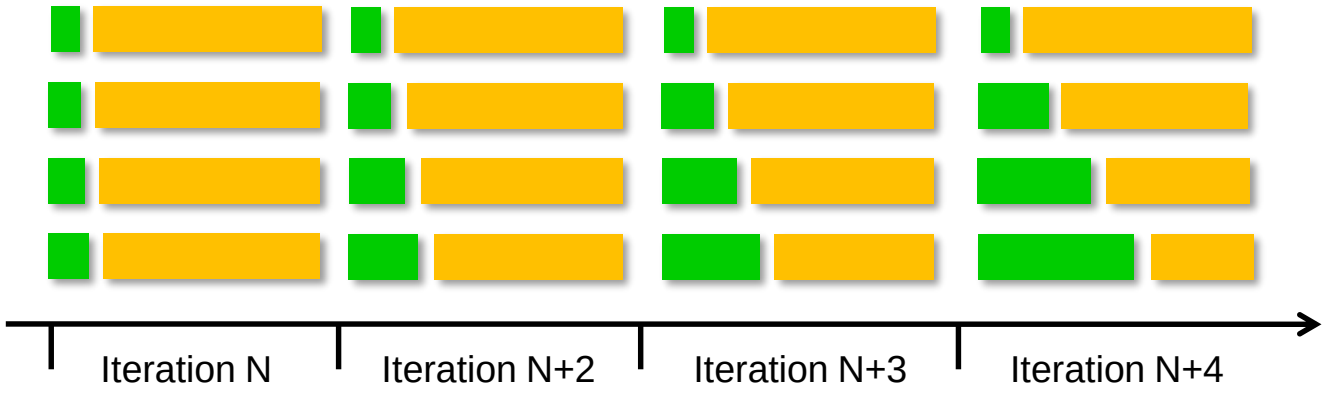
Disadvantages
– **Variations over time get lost**

PROFILING: ISSUES RELATED TO "AVERAGING"

- Moving bottleneck across processors can "average out" imbalances



- Imbalance changes over time ⇨ problem worse for short runs!



MEASUREMENT METHODS: TRACING

- Recording **information about** significant points (**events**) during execution of the program
  - Enter/leave a code region (function, loop, …)
  - Send/receive a message ...
- Save information in **event record**
  - Timestamp, location ID, event type
  - plus event specific information
- **Event trace** := stream of event records sorted by time

⇨ Abstract execution model on level of defined events

Advantages

+ **Can be used to reconstruct the dynamic behavior**
+ **Profiles can be calculated out of trace data**

Disadvantages

– **HUGE trace files**
– **Can only be used for short durations or small configurations**

EVENT TRACING

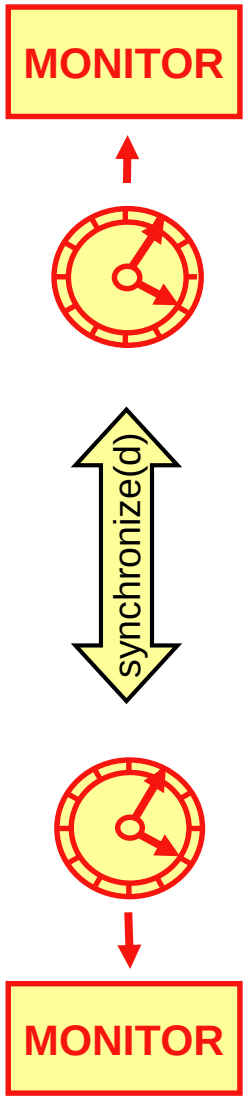
Process A

```
void foo() {
  trc_enter("foo");
  ...
  trc_send(B);
  send(B, tag, buf);
  ...
  trc_exit("foo");
}
```

instrument

Process B

```
void bar() {
  trc_enter("bar");
  ...
  recv(A, tag, buf);
  trc_recv(A);
  ...
  trc_exit("bar");
}
```



Local trace A

...		
58	ENTER	1
62	SEND	B
64	EXIT	1
...		

1	foo
...	

Local trace B

...		
60	ENTER	1
68	RECV	A
69	EXIT	1
...		

1	bar
...	

Global trace

...			
58	A	ENTER	1
60	B	ENTER	2
62	A	SEND	B
64	A	EXIT	1
68	B	RECV	A
69	B	EXIT	2
...			

merge

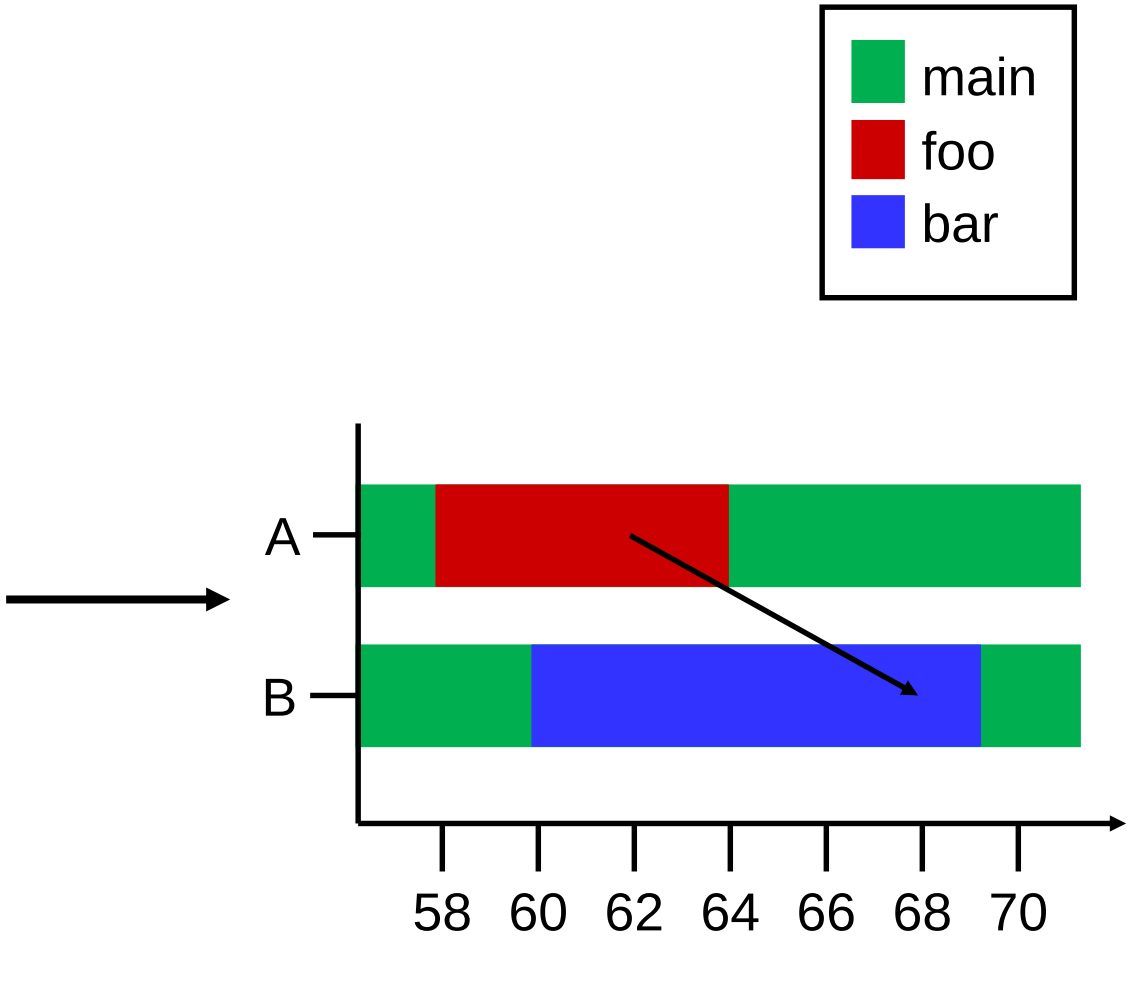
unify

1	foo
2	bar
...	

EVENT TRACING: “TIMELINE” VISUALIZATION

1	foo
2	bar
3	...

...			
58	A	ENTER	1
60	B	ENTER	2
62	A	SEND	B
64	A	EXIT	1
68	B	RECV	A
69	B	EXIT	2
...			



NO SINGLE SOLUTION IS SUFFICIENT!

⇨ **Combination of methods, techniques and tools needed**

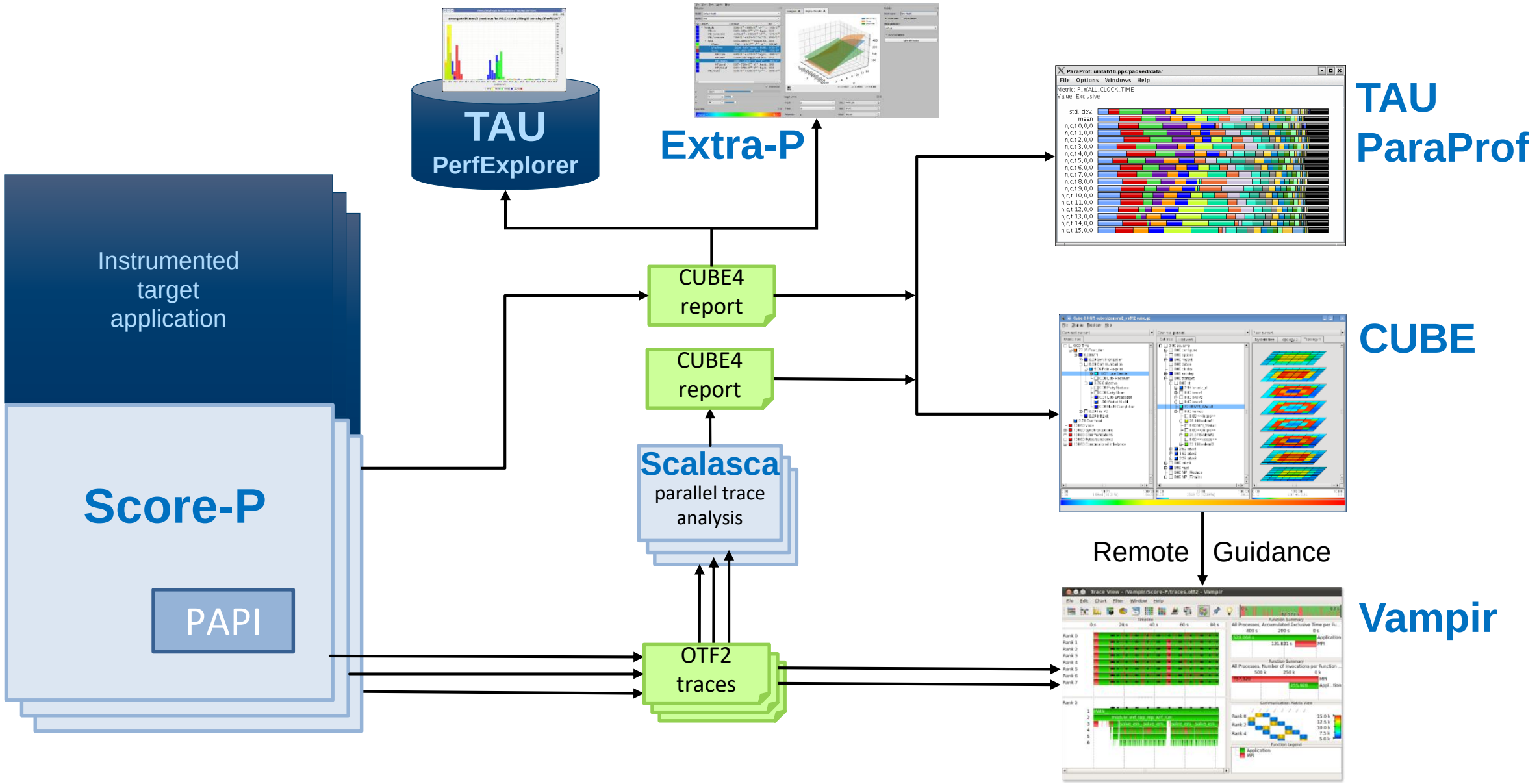
- Instrumentation
  - Source code / binary, static / dynamic, manual / automatic
- Measurement
  - Internal / external trigger, profiling / tracing
- Analysis
  - Statistics, Visualization, Automatic, Data mining, …

The Big Picture

THE SCORE-P ECOSYSTEM

Score-P TOOL ECOSYSTEM

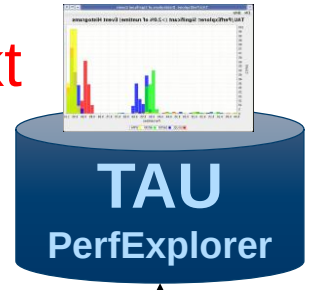
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

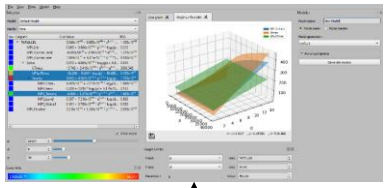


Score-P TOOL ECOSYSTEM

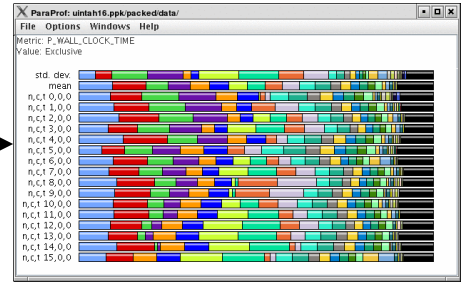
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

5 Okt





Extra-P
19 Okt



TAU
ParaProf
5 Okt

Instrumented
target
application

Score-P
7. Sep

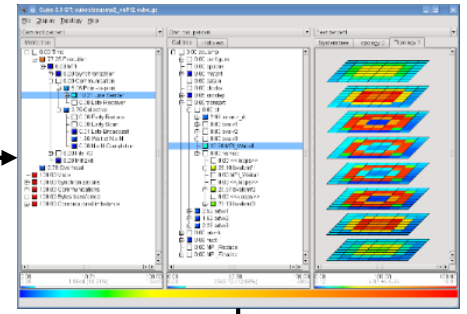
PAPI

CUBE4
report

CUBE4
report

Scalasca
parallel trace
analysis

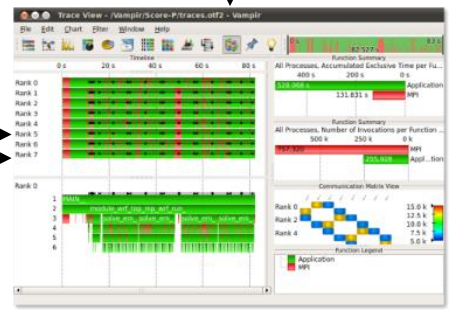
14. Sep



CUBE
7. Sep

Remote | Guidance

OTF2
traces



Vampir
21. Sep

How To

PARALLEL APPLICATION PROFILE MEASUREMENT WITH SCORE-P

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

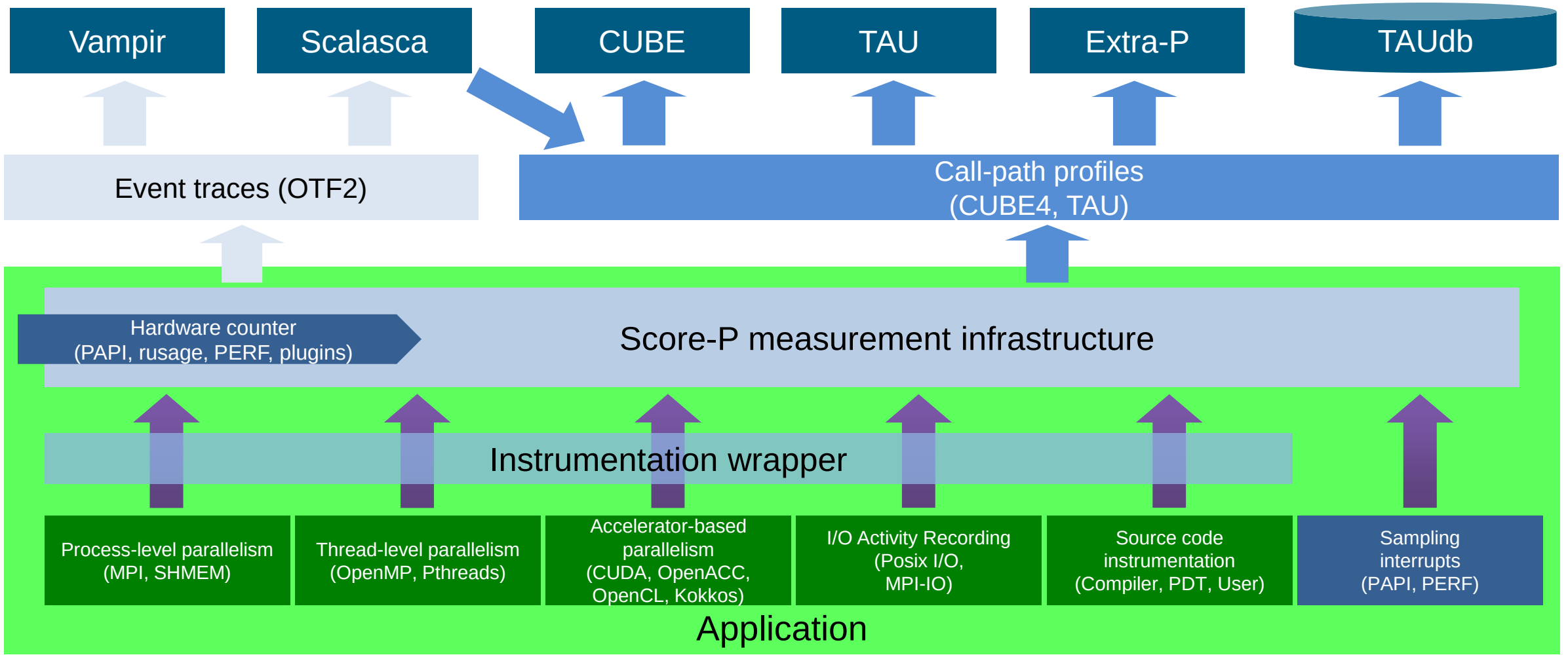
- Community-developed open-source
- Replaced tool-specific instrumentation and measurement components of partners
- http://www.score-p.org

JÜLICH
Forschungszentrum
JÜLICH
SUPERCOMPUTING
CENTRE

Score-P FUNCTIONALITY

- Provide typical functionality for HPC performance tools
- **Instrumentation** (various methods)
  - Multi-process paradigms (MPI, SHMEM)
  - Thread-parallel paradigms (OpenMP, POSIX threads)
  - Accelerator-based paradigms (OpenACC, CUDA, OpenCL. Kokkos)
  - **In any combination!**
- Flexible **measurement** without re-compilation:
  - Basic and advanced **profile** generation (⇨ CUBE4 format)
  - Event **trace** recording (⇨ OTF2 format)
- Highly scalable I/O functionality
- Support all fundamental concepts of partner's tools

Score-P ARCHITECURE



BASIC INSTRUMENTATION OF PARALLEL PROGRAMS

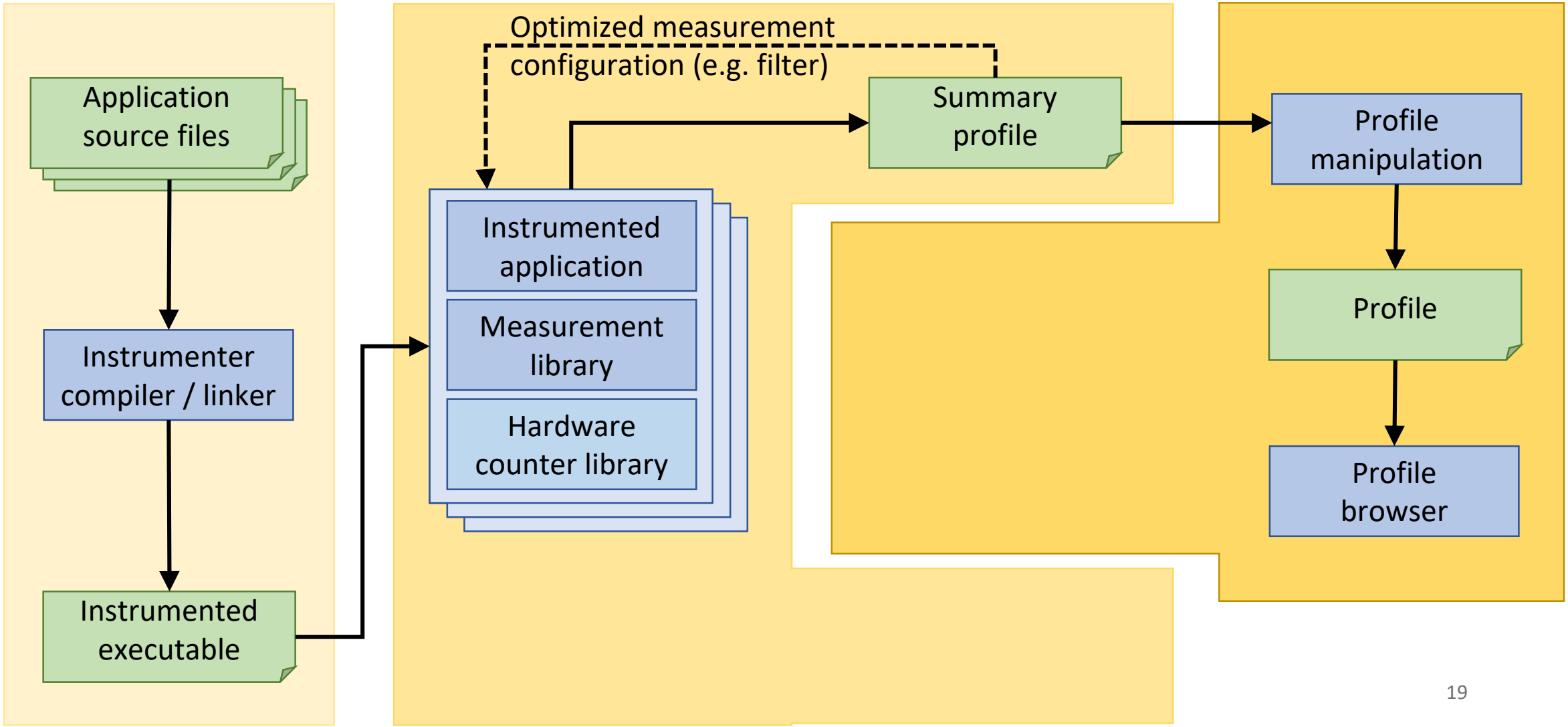
- User code
  - **Automatic instrumentation of functions via compiler switches**
    - Can be optimized via a filter specification
  - Manually with instrumentation API (arbitrary regions)

- **MPI via PMPI wrapper functions**

- **OpenMP via OPARI source-code instrumenter**
  - Will be replaced by OMPT interface adapter in the future

PERFORMANCE ANALYSIS WORKFLOW

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE



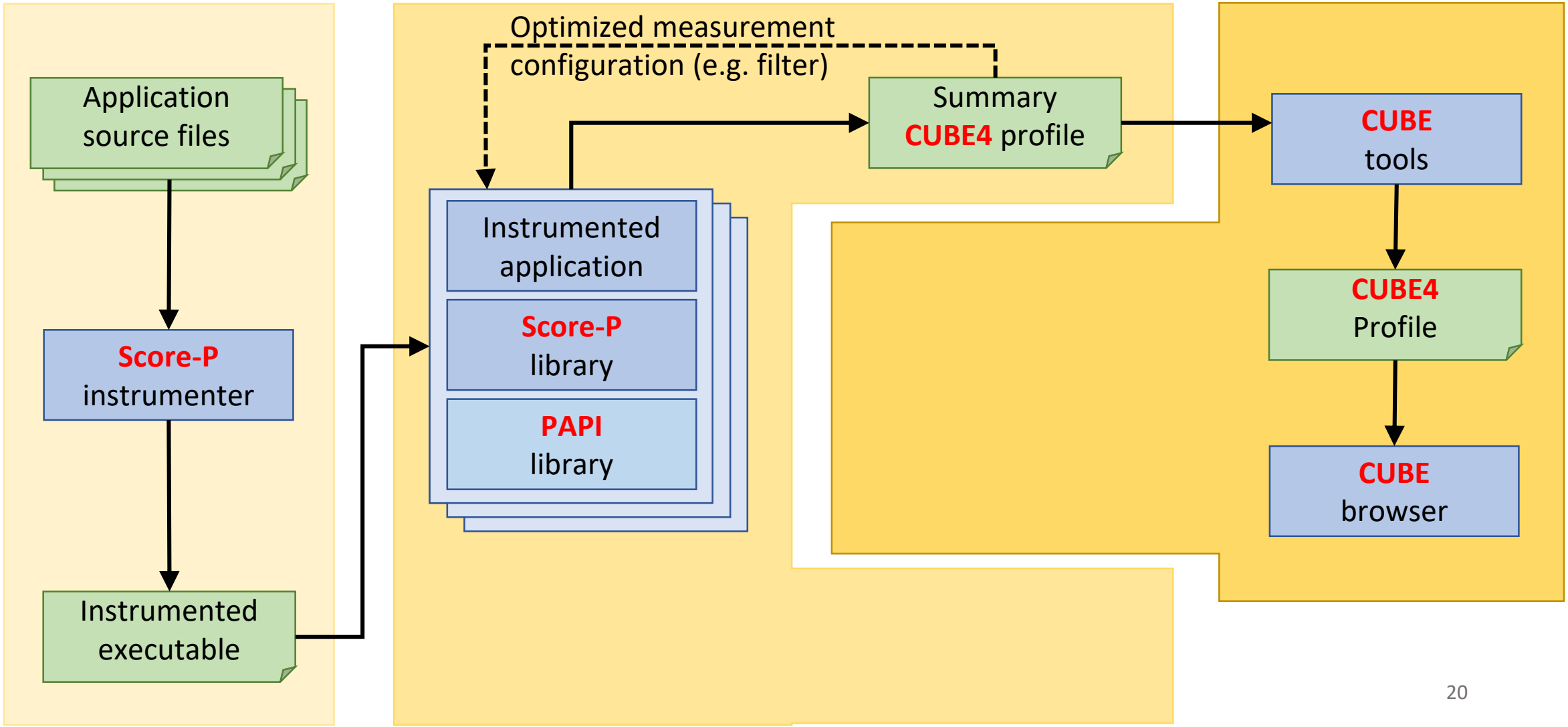
PERFORMANCE ANALYSIS WORKFLOW

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

1. Instrumentation

2. Measurement

3. Analysis



DEMO

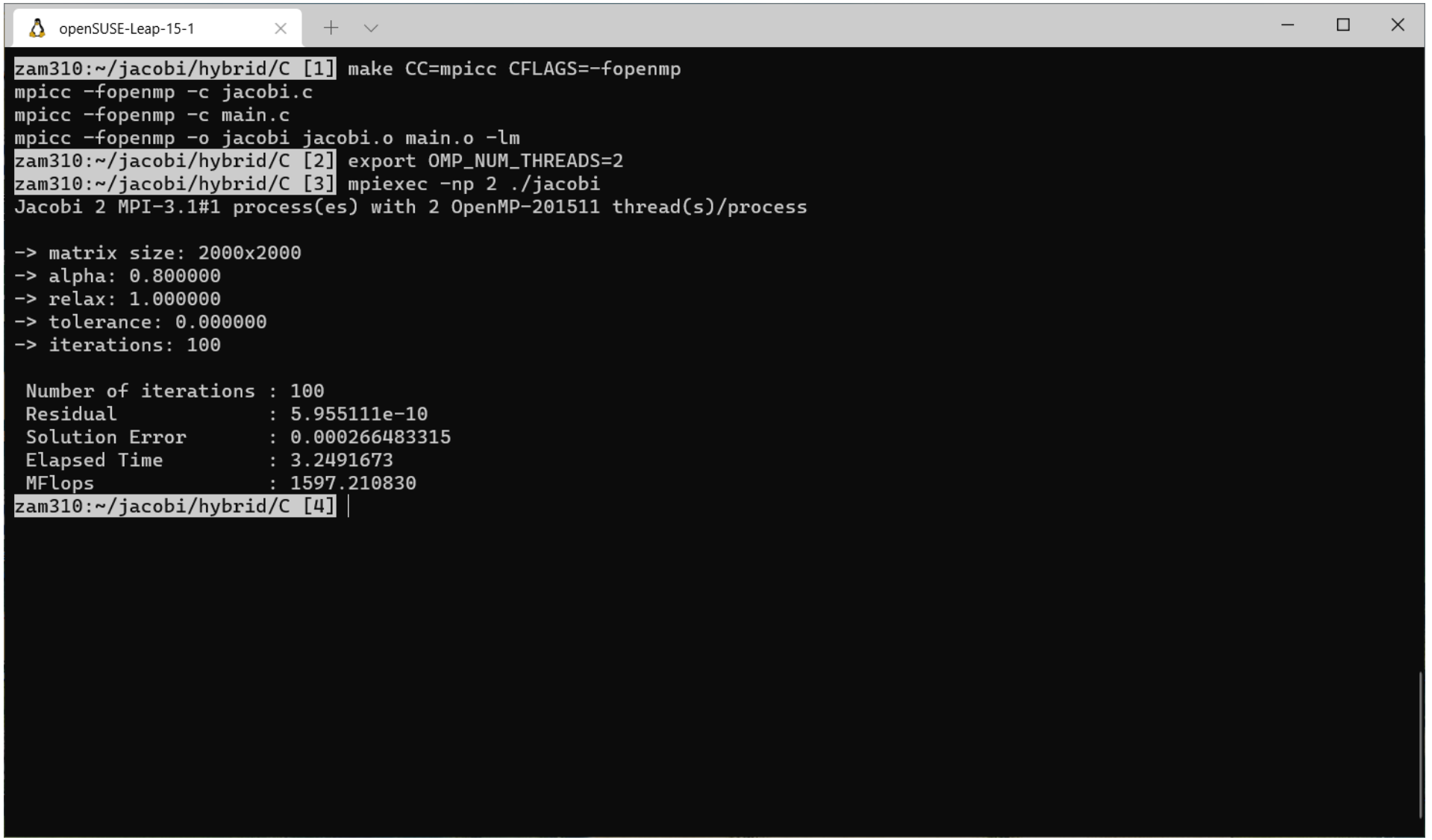
- Measurement of simple Jacobi solver
  - Solves Poisson equation on rectangular grid assuming
    - Uniform discretization in each direction
    - Dirichlect boundary conditions

- 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

DEMO: BASE RUN OF APPLICATION

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

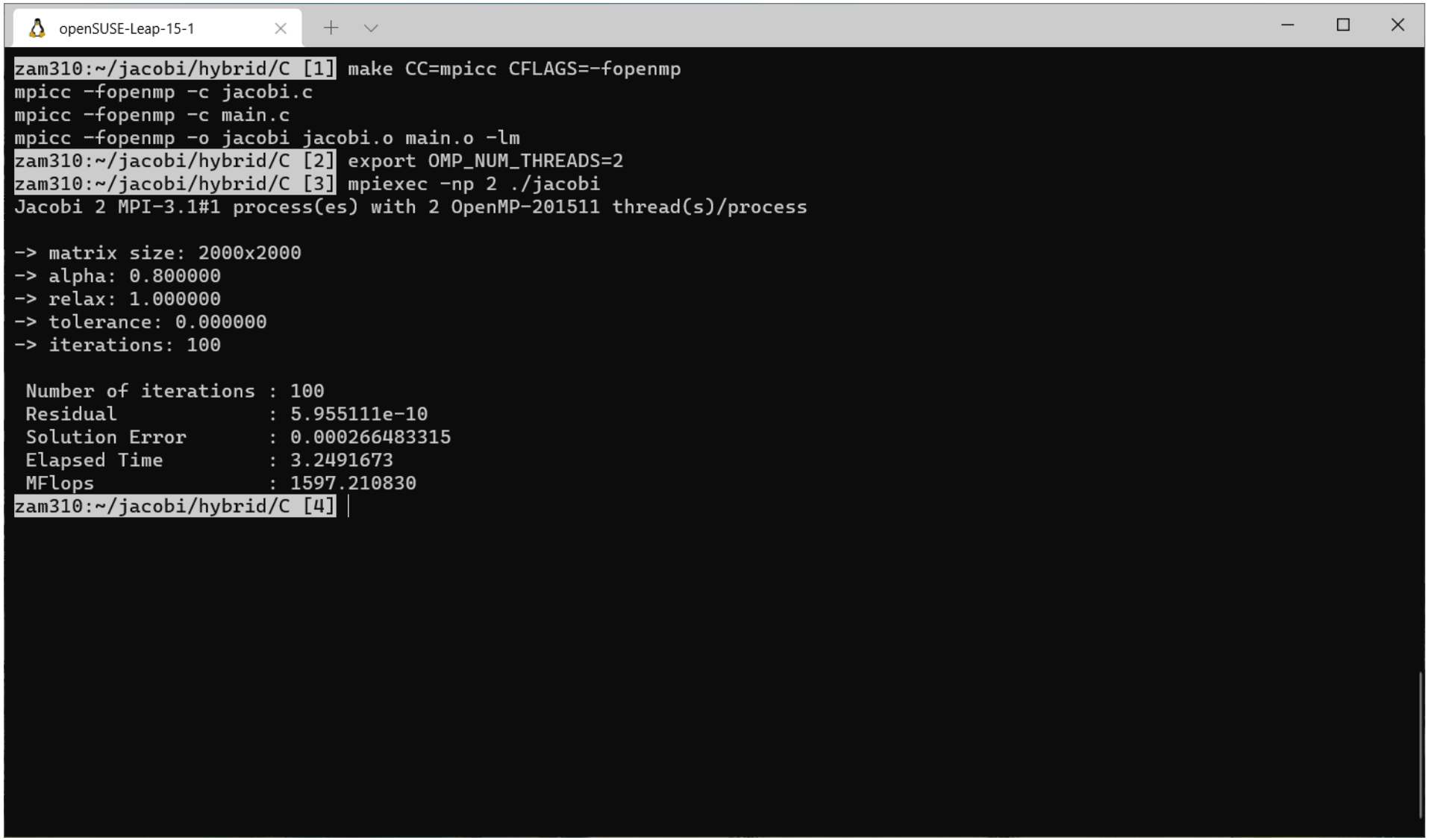


Notes

- 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: BASE RUN OF APPLICATION

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE



Notes

- Make sure tools are in $PATH
- Instrument: prepend scorep
- Measure profile: run instrumented application
- Compare execution time to check overhead
- Profile in scorep-xxx subdirectory

DEMO: OPTIMIZE MEASUREMENT CONFIGURATION (SCOREP-SCORE)

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

openSUSE-Leap-15-1

```
zam310:~/jacobi/hybrid/C [10] scorep-score -r scorep-20210906_1511_11469445778398O0/profile.cubex

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   14.38   100.0        3745.70  ALL
         OMP     61,078  2,812   13.64    94.8        4849.76  OMP
         MPI     27,440    812    0.66     4.6         807.85  MPI
         COM      2,756    212    0.09     0.6         423.99  COM
      SCOREP         41      2    0.00     0.0          24.80  SCOREP
         USR         26      2    0.00     0.0          20.30  USR

         OMP     17,400    400    0.00     0.0           2.15  !$omp parallel @jacobi.c:61
         OMP     17,400    400    0.00     0.0           1.64  !$omp parallel @jacobi.c:148
         MPI      8,900    200    0.00     0.0           8.60  MPI_Irecv
         MPI      8,900    200    0.00     0.0           5.11  MPI_Isend
         MPI      6,800    200    0.39     2.7        1937.67  MPI_Allreduce
         OMP      5,200    400    0.00     0.0           2.06  !$omp implicit barrier @jacobi.c:80
         OMP      5,200    400    9.92    69.0       24801.17  !$omp for @jacobi.c:64
         OMP      5,200    400    0.97     6.8        2428.47  !$omp implicit barrier @jacobi.c:79
         OMP      5,200    400    2.27    15.8        5686.02  !$omp for @jacobi.c:148
         OMP      5,200    400    0.29     2.0         722.15  !$omp implicit barrier @jacobi.c:155
         MPI      2,600    200    0.09     0.6         439.76  MPI_Waitall
         COM      2,600    200    0.00     0.0          15.36  ExchangeJacobiMpiData
         OMP        174      4    0.00     0.0           6.62  !$omp parallel @main.c:161
         MPI         68      2    0.00     0.0         132.30  MPI_Bcast
         MPI         68      2    0.00     0.0         261.50  MPI_Reduce
         OMP         52      4    0.18     1.2       44827.72  !$omp for @main.c:161
         OMP         52      4    0.00     0.0         183.28  !$omp implicit barrier @main.c:177
```

Notes

- Optimize measurement config: scoring with scorep-score
- Potential need for filtering
  → see user guides

DEMO: MEASUREMENT CONFIGURATION

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

```
openSUSE-Leap-15-1
zam310:~/jacobi/hybrid/C [12] scorep-info config-vars | more
SCOREP_ENABLE_PROFILING
  Description: Enable profiling
         Type: Boolean
      Default: true

SCOREP_ENABLE_TRACING
  Description: Enable tracing
         Type: Boolean
      Default: false

SCOREP_VERBOSE
  Description: Be verbose
         Type: Boolean
      Default: false

SCOREP_TOTAL_MEMORY
  Description: Total memory in bytes per process to be consumed by the
               measurement system
         Type: Number with size suffixes
      Default: 16000k

SCOREP_PAGE_SIZE
  Description: Memory page size in bytes
         Type: Number with size suffixes
      Default: 8k

SCOREP_EXPERIMENT_DIRECTORY
  Description: Name of the experiment directory as child of the current working
               directory
         Type: Path
      Default: ""
```

Notes

- Measurement configuration via environment variables
- “scorep-info configvars” command lists all variables

SCORE-P: FURTHER USEFUL INFORMATION

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

- 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)
    - Using Score-P with cmake
    - Wrapping call to 3rd party libraries
    - Analyzing application memory usage
    - Analyzing CUDA/OpenCL/OpenACC applications
    - Hardware performance counters
    - User instrumentation API

How To

PROFILE ANALYSIS WITH CUBE

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

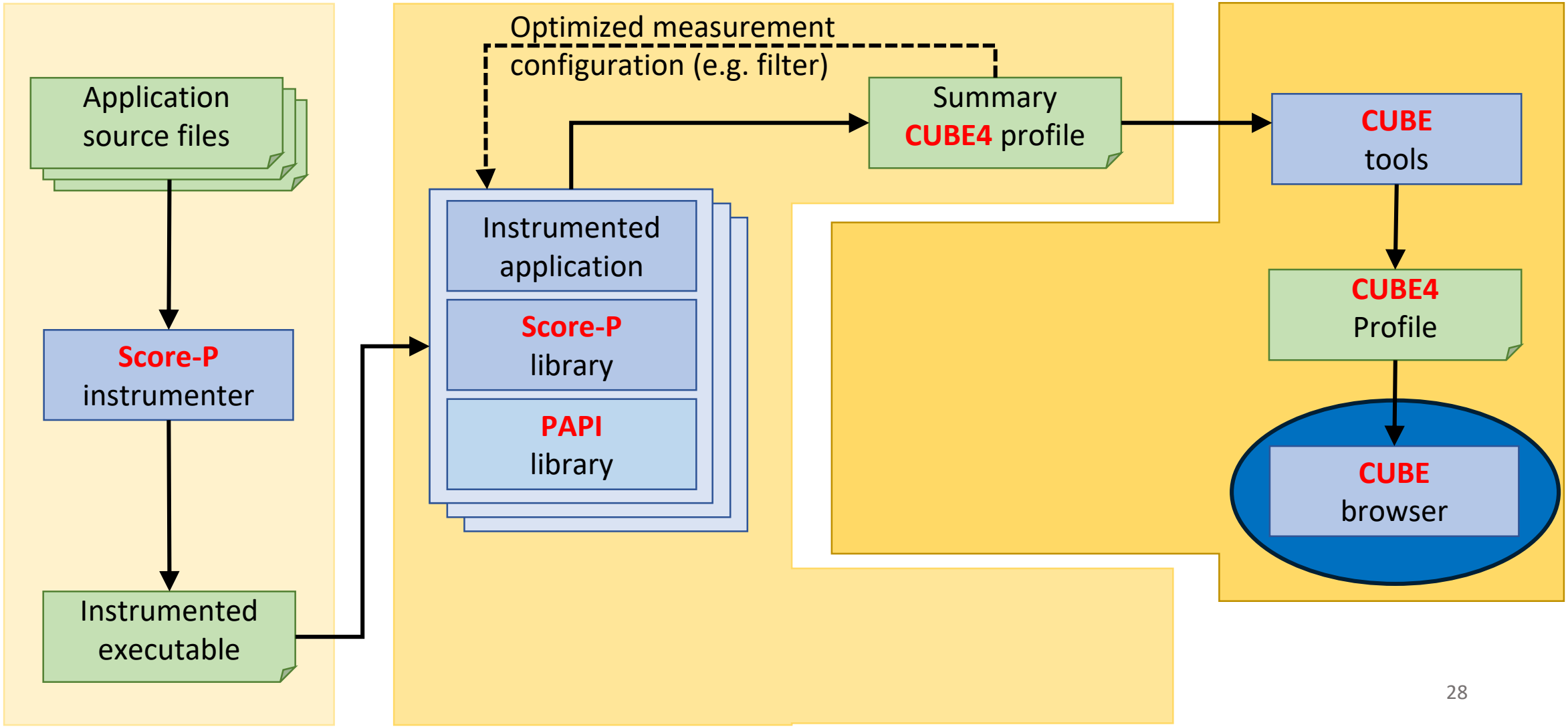
PERFORMANCE ANALYSIS WORKFLOW

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

1. Instrumentation

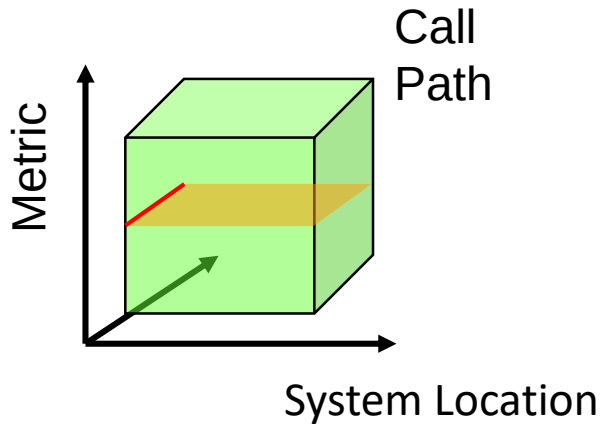
2. Measurement

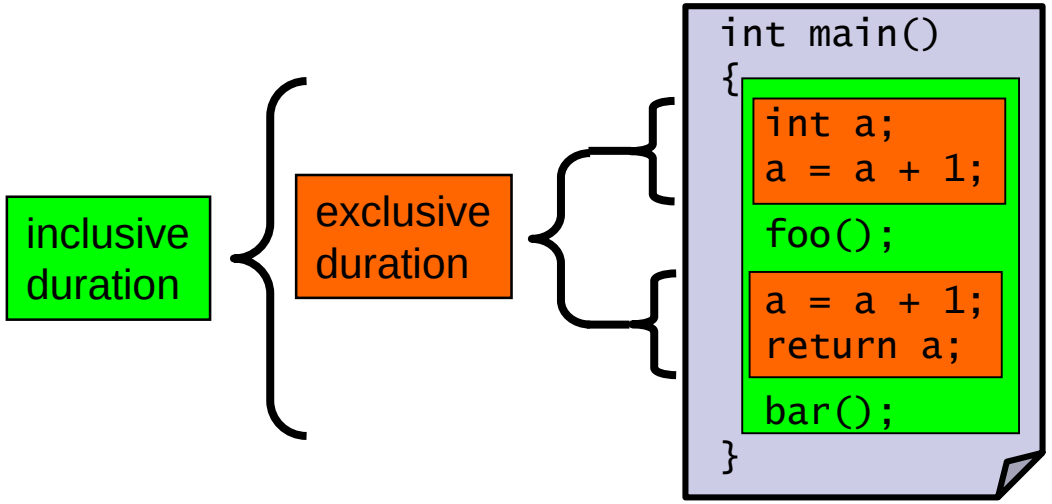
3. Analysis



CUBE DATA

- Measured values organized in 3D block (“Cube”) along **three hierarchical axes**
  - Metrics (general → specific)
  - Call tree
  - System location (machine → node → process → thread)
- Displayed as **three coupled tree browsers**
  - Each node displays metric value
    - As color: for easy identification of bottlenecks
    - As number: for precise comparison
  - Displayed metric value depends on state
    - Collapsed (inclusive value)
    - Expanded (exclusive value)





CUBE BASIC COMMANDS

- **Starting Cube GUI:**

```
% cube <cube_file>
```

1 Window
2 Commands
3 Panes

- Basic GUI commands
  - **Expand / Collapse tree nodes**
    - Chooses level of granularity
    - Use context menu to expand / collapse whole (sub)trees
  - **Select tree nodes**
    - Shows distribution of metric value in tree to the right
    - Use Ctrl+Click to select multiple nodes

DEMO

- TeaLeaf Reference V1.0

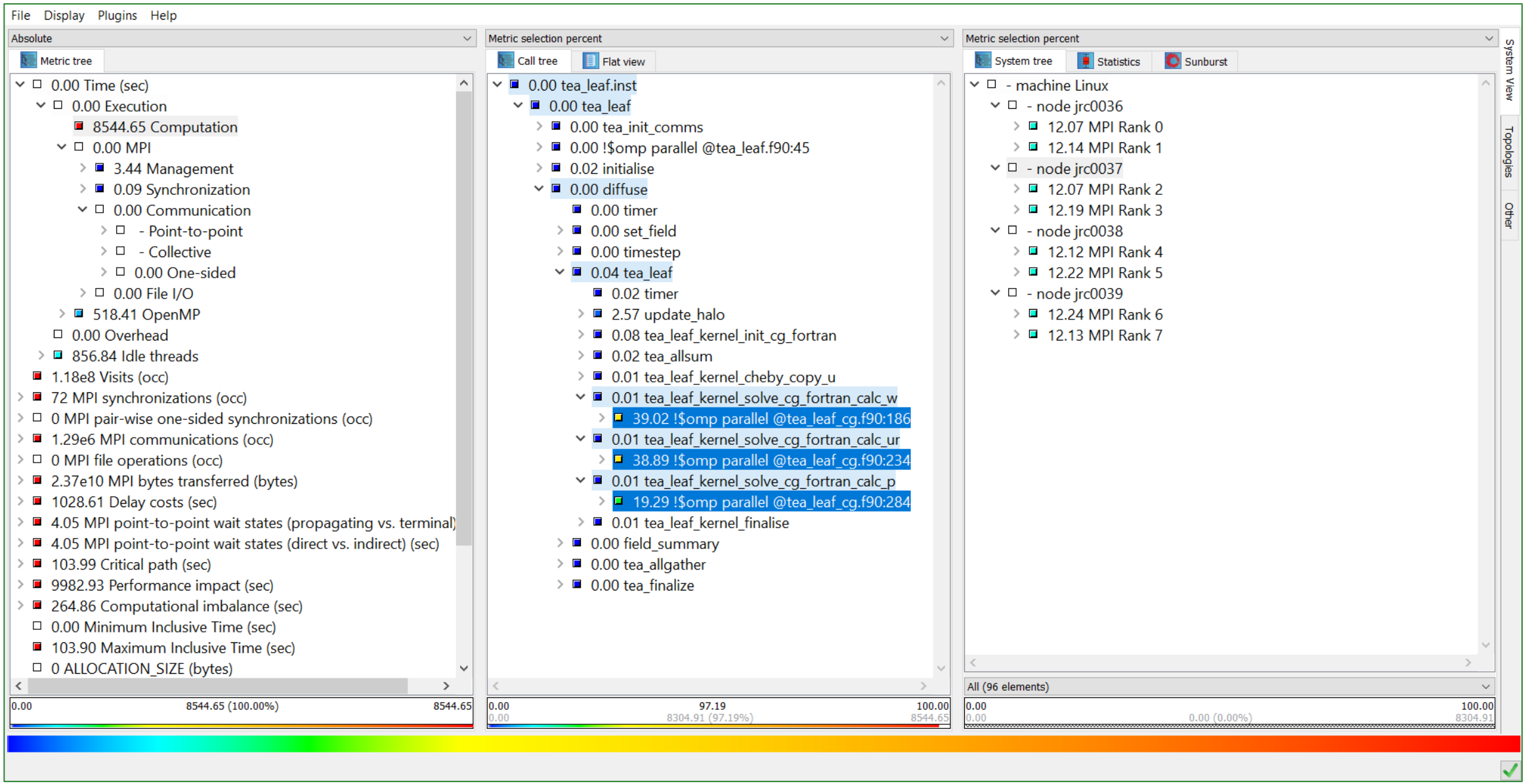
- HPC mini-app developed by the UK Mini-App Consortium
  - Solves the linear 2D heat conduction equation on a spatially decomposed regular grid using a 5 point stencil with implicit solvers
  - https://github.com/UK-MAC/TeaLeaf_ref/archive/v1.0.tar.gz

- 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

DEMO

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE



USEFUL CUBE COMMAND LINE TOOLS

- **cube_diff:** Computes the difference between the data of two cube files
- **cube_merge:** Combine multiple cube files into one
- **cube_derive:** Add user-defined derived metric
- **cube_cut:** Cut sub calltrees out out of / from cube data
- **cube_stat:** Print basic statistics in text-form
- **cube_dump:** Export data from cube file in various forms (human, gnuplot, csv, R)

CUBE: FURTHER USEFUL INFORMATION

- Slides from 40th VI-HPS Tuning Workshop
  - CUBE profile explorer
- User Guide (HTML | PDF)
- Cube Derived metrics guide (HTML | PDF)
- Cube Command line tools guide (HTML | PDF)

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