Using POP Tools: Score-P, Scalasca

Bernd Mohr

EU H2020 Centre of Excellence (CoE)

1 December 2018 – 30 November 2021

Grant Agreement No 824080

What you will learn

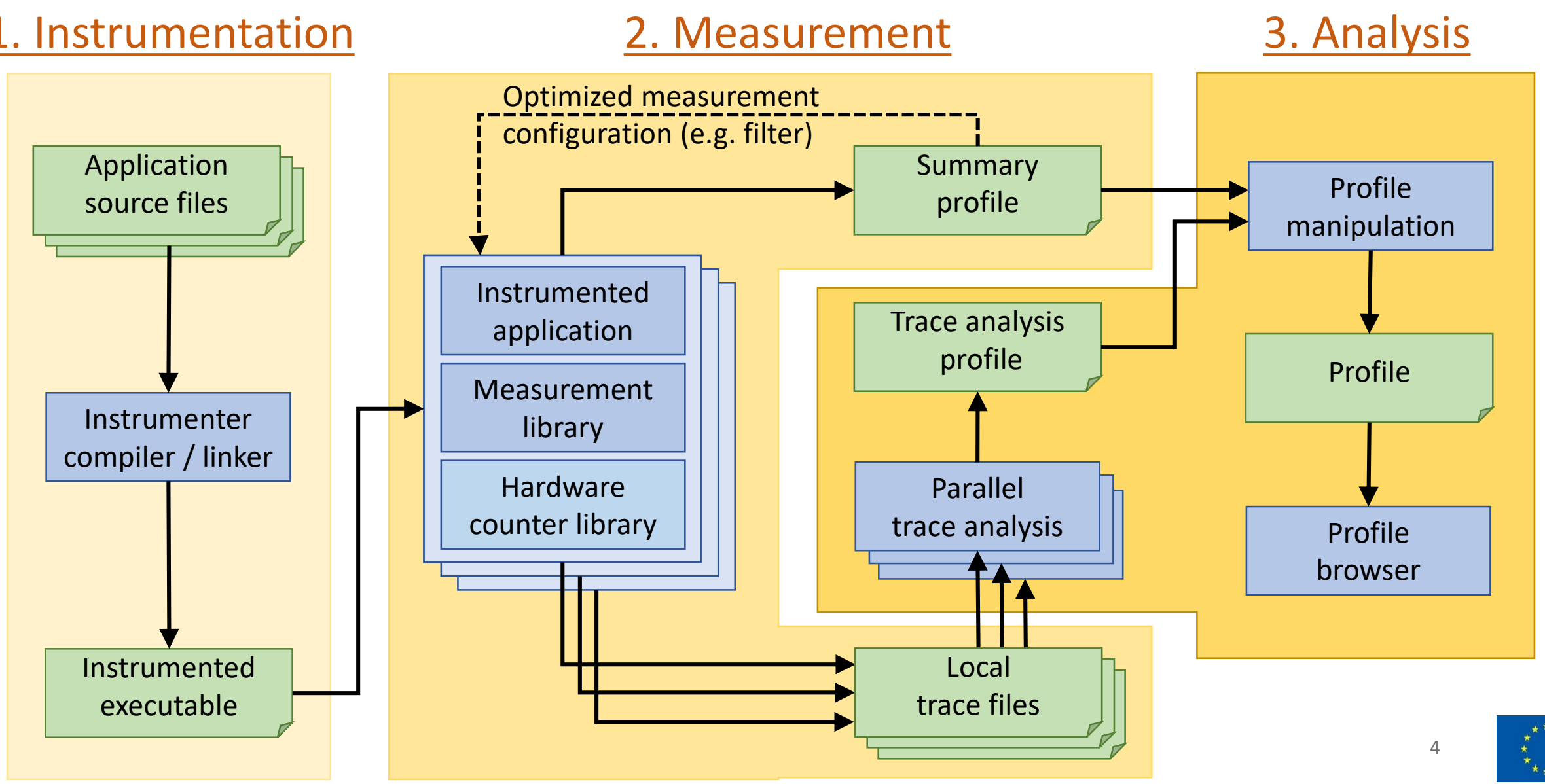
How to instrument HPC application codes and make measurements using the POP tools Score-P and Scalasca

1. Do base run of HPC application first
2. Instrument HPC application with Score-P
3. Perform profile measurement
    A. Optimize measurement configuration for filtering and tracing
4. Perform trace measurement and analyze it with Scalasca

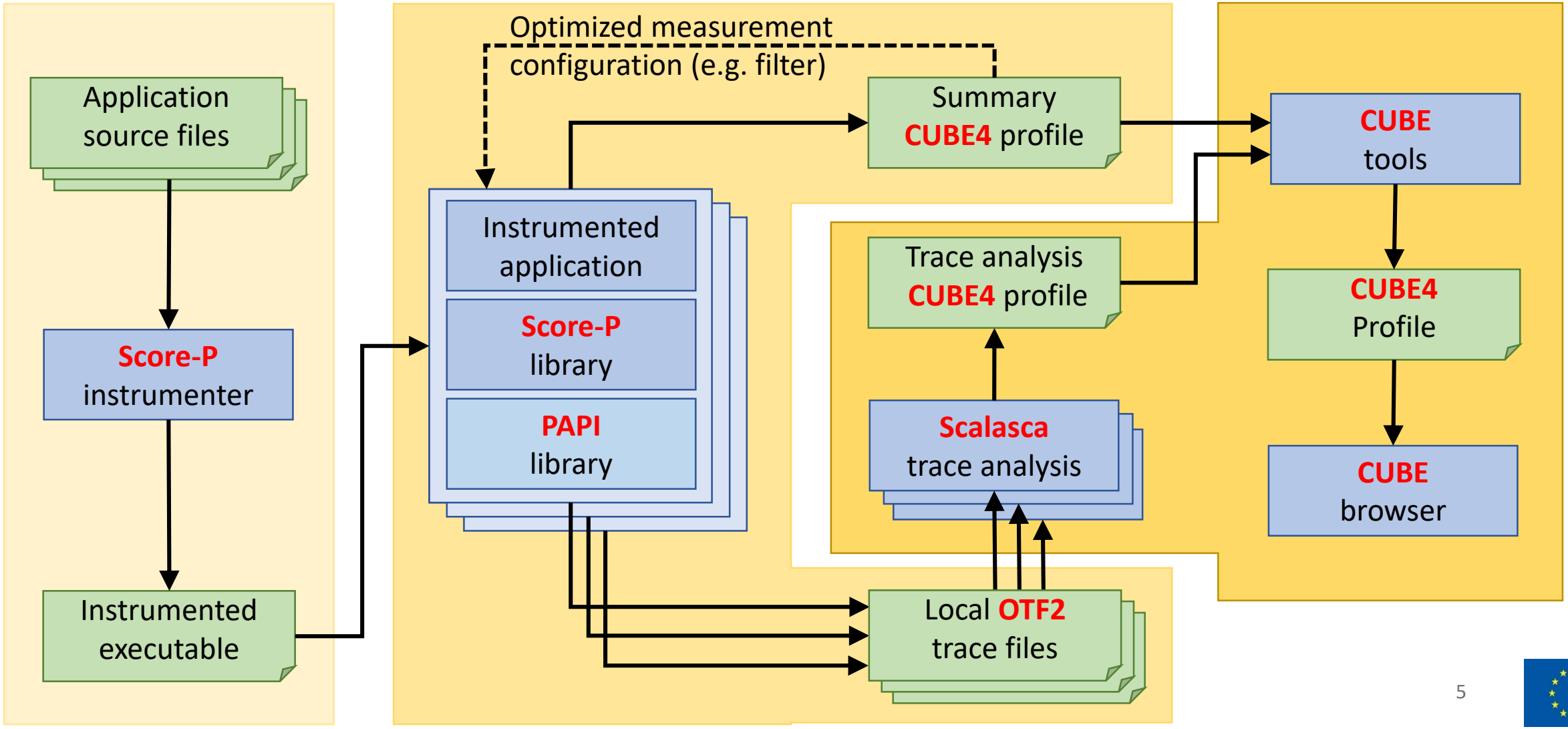
Prerequisites

- Access to HPC cluster or Linux workstation/laptop
  - Including development software (Compiler, MPI library)
- Access to the source code of the application to analyze
- POP tools Score-P and Scalasca installed
  - Make sure "`bin`" subdirectories of tool installation prefixes are in `$PATH`

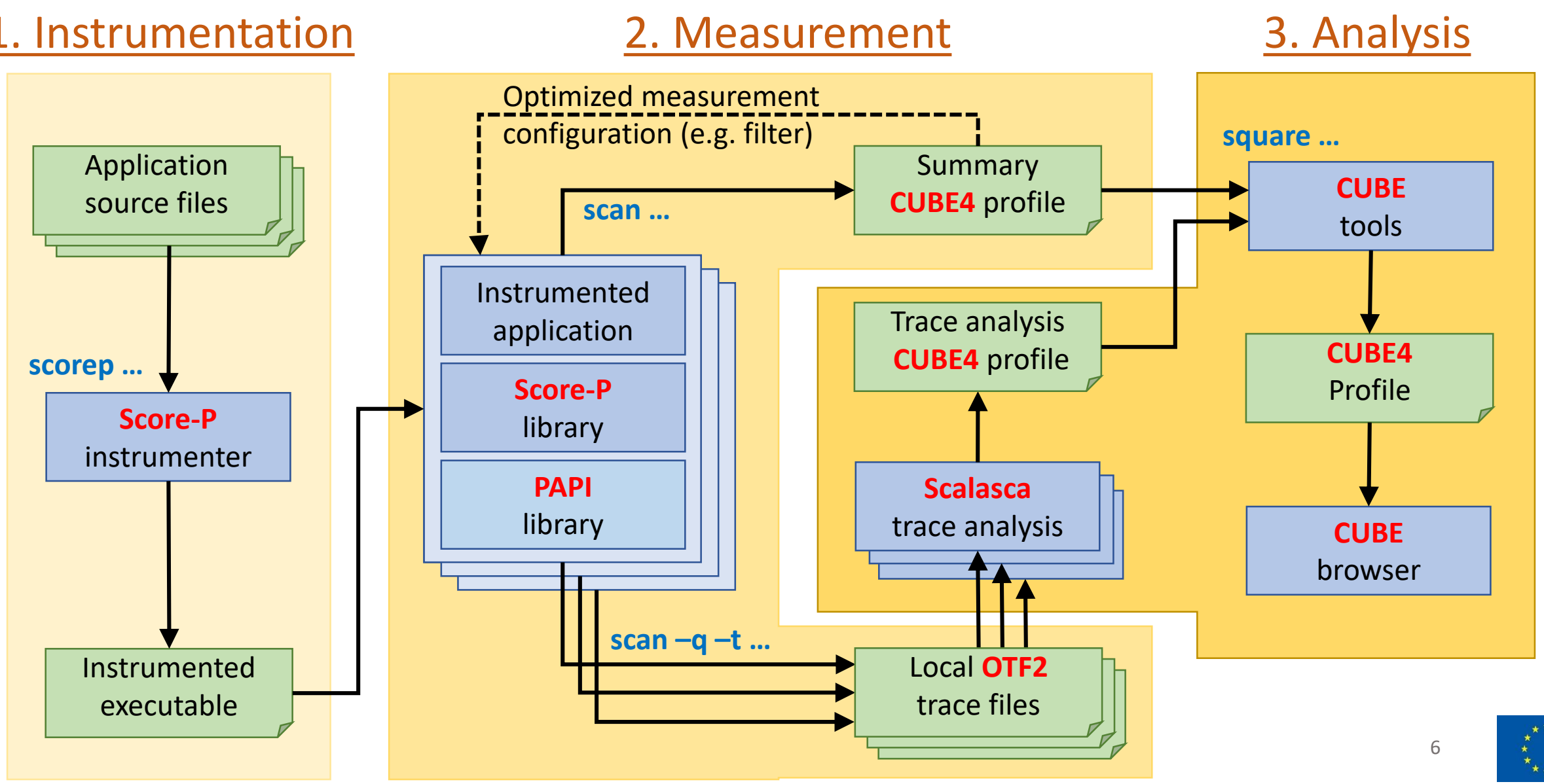
Performance Analysis Workflow



Performance Analysis Workflow



Performance Analysis Workflow

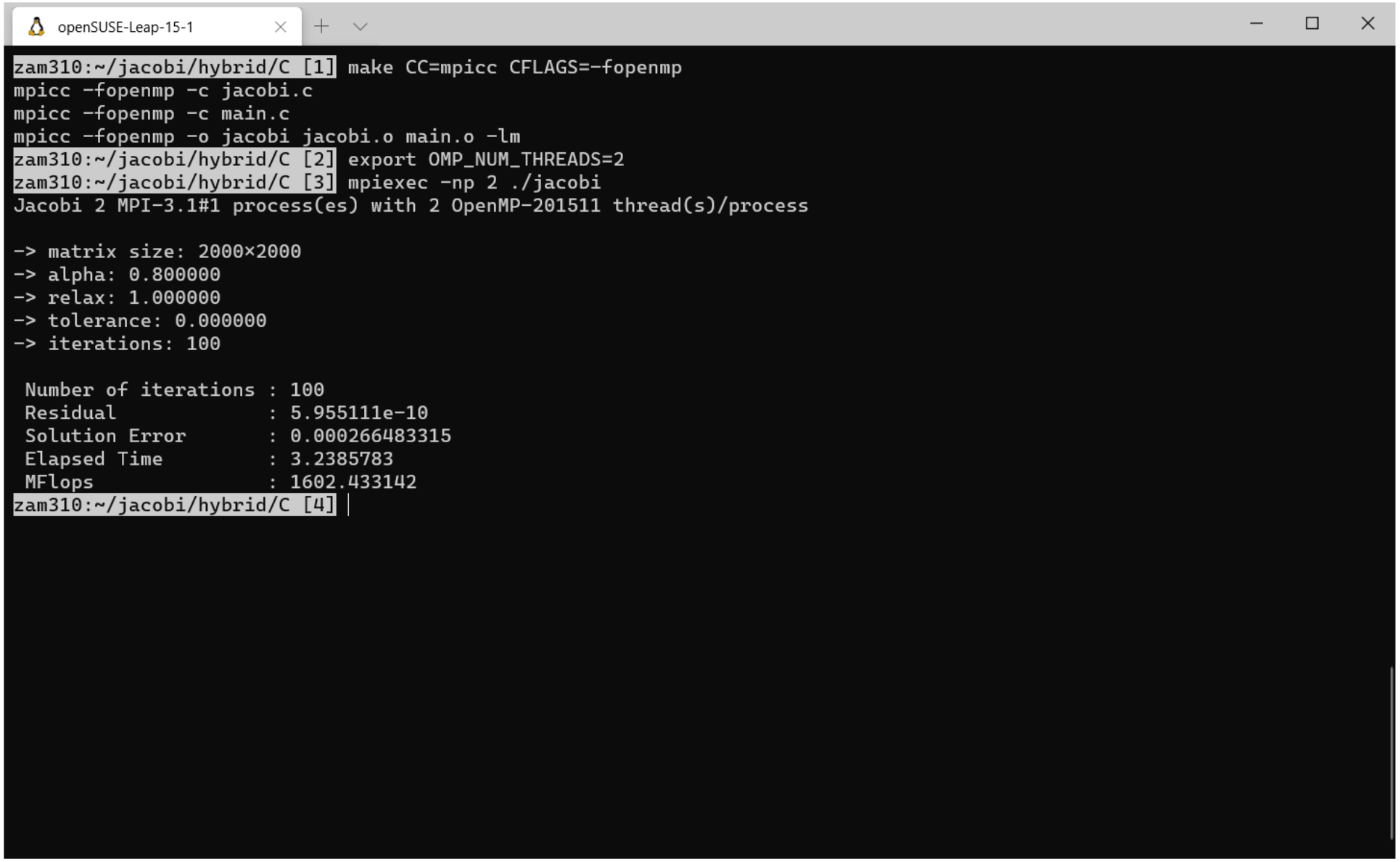


Demo

- 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)

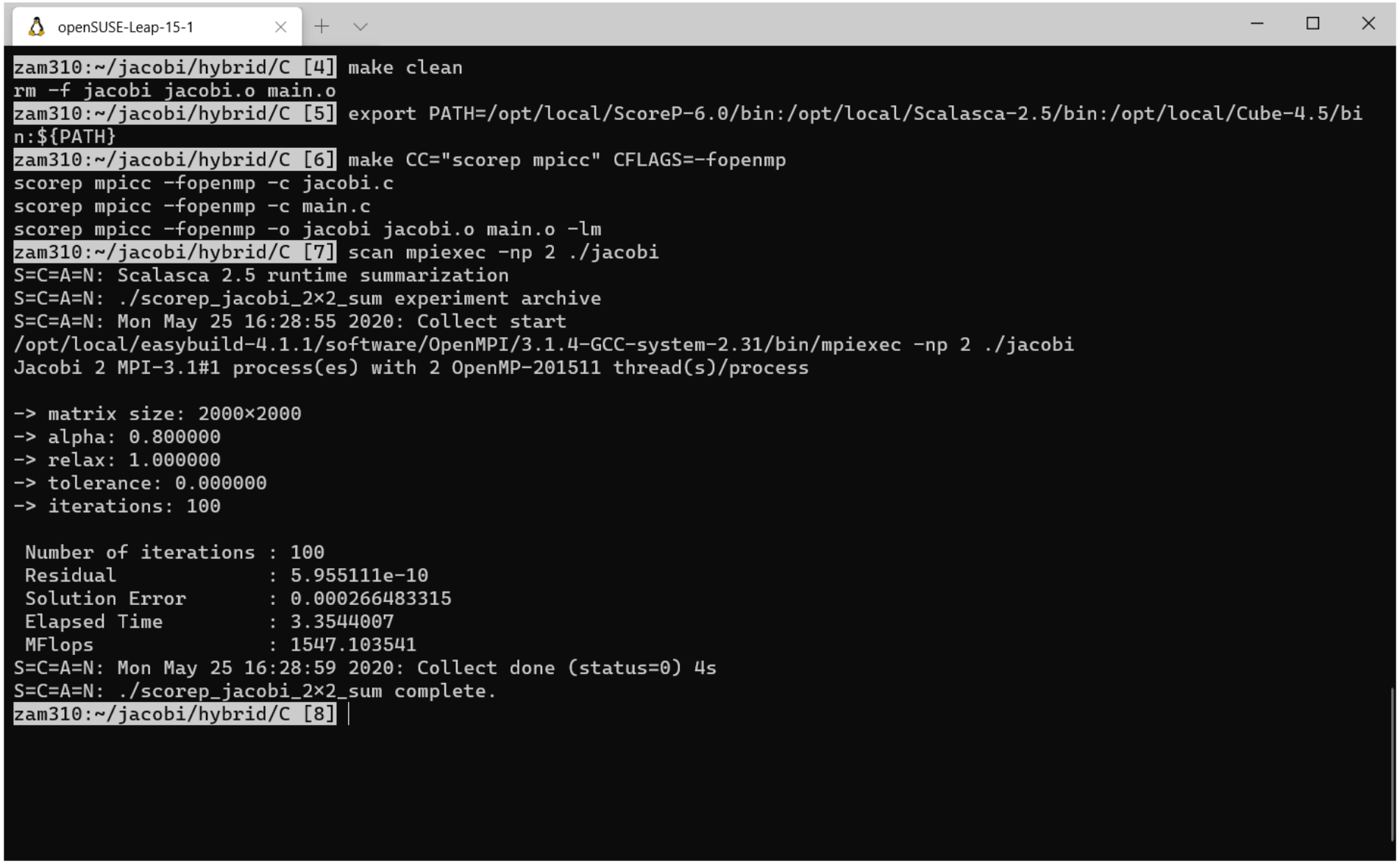
Demo: Base Run of Application



Notes

- Compile application
- Execute application with 2 threads on 2 processes
- Write down execution time for later comparison

Demo: Instrument + Profile



Notes

- Make sure tools are in $PATH
- Instrument: prepend scorep
- Measure profile: prepend scan

- Compare execution time to check overhead

Demo: Optimize Measurement Configuration (Score)

openSUSE-Leap-15-1

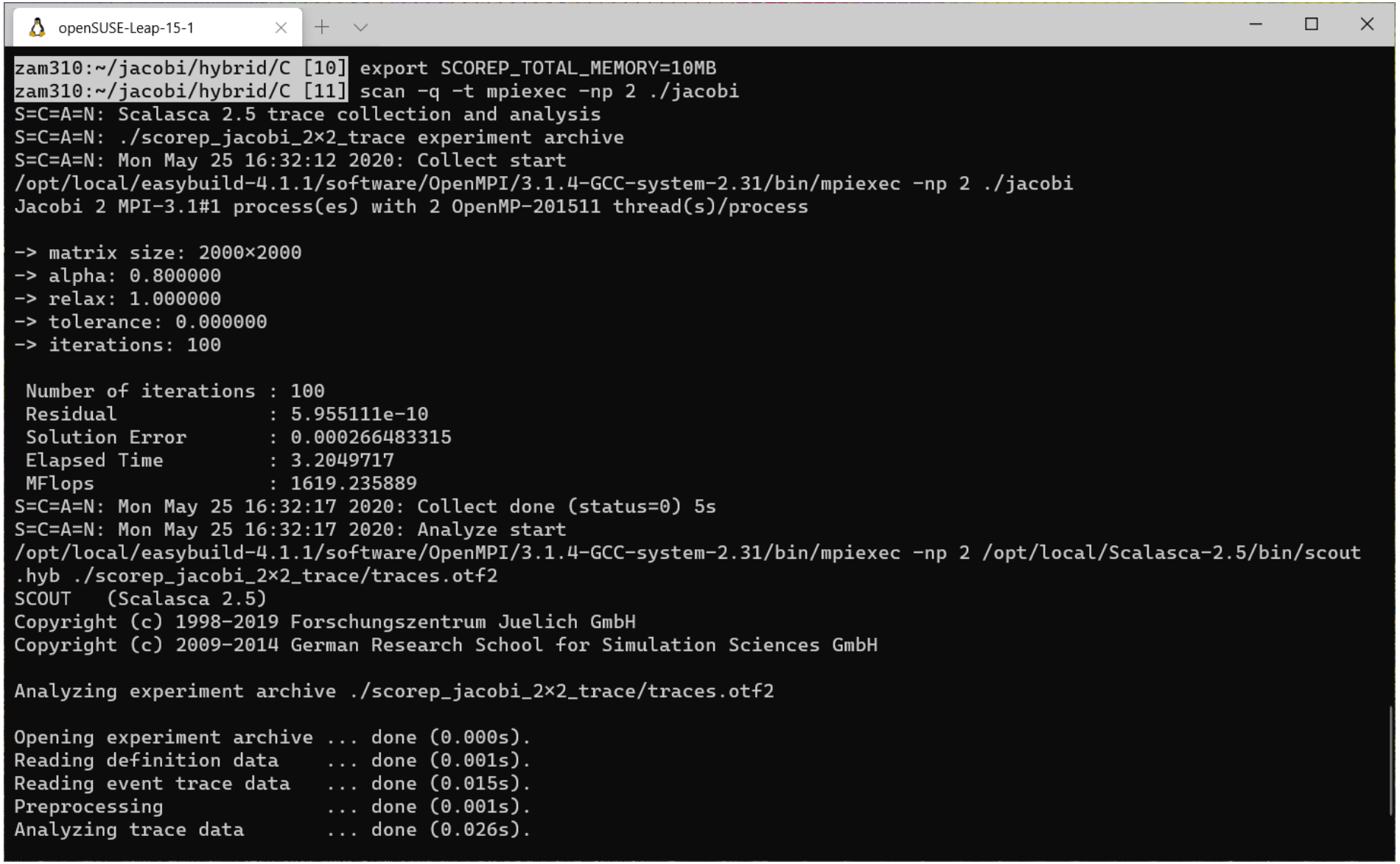
```
zam310:~/jacobi/hybrid/C [8] square -s ./scorep_jacobi_2x2_sum/
INFO: Post-processing runtime summarization report (profile.cubex)...
/opt/local/ScoreP-6.0/bin/scorep-score -r ./scorep_jacobi_2x2_sum/profile.cubex > ./scorep_jacobi_2x2_sum/scorep.scor
e
INFO: Score report written to ./scorep_jacobi_2x2_sum/scorep.score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



Notes

- Measure trace: prepend scan
  - –q: profile off
  - –t: trace on
- After trace measurement, Scalasca trace analyzer runs automatically

Further Useful Information

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

- Scalasca documentation
  - A full workflow example
- Score-P documentation
  - Performance Analysis Workflow Using Score-P
- Slides from 33rd 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

Where to get help

Tool	Support Email
Score-P	support@score-p.org
Scalasca, Cube	scalasca@fz-juelich.de

Performance Optimisation and Productivity

A Centre of Excellence in HPC

Contact:

- https://www.pop-coe.eu
- pop@bsc.es
- @POP_HPC
- youtube.com/POPHPC

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676553 and 824080.