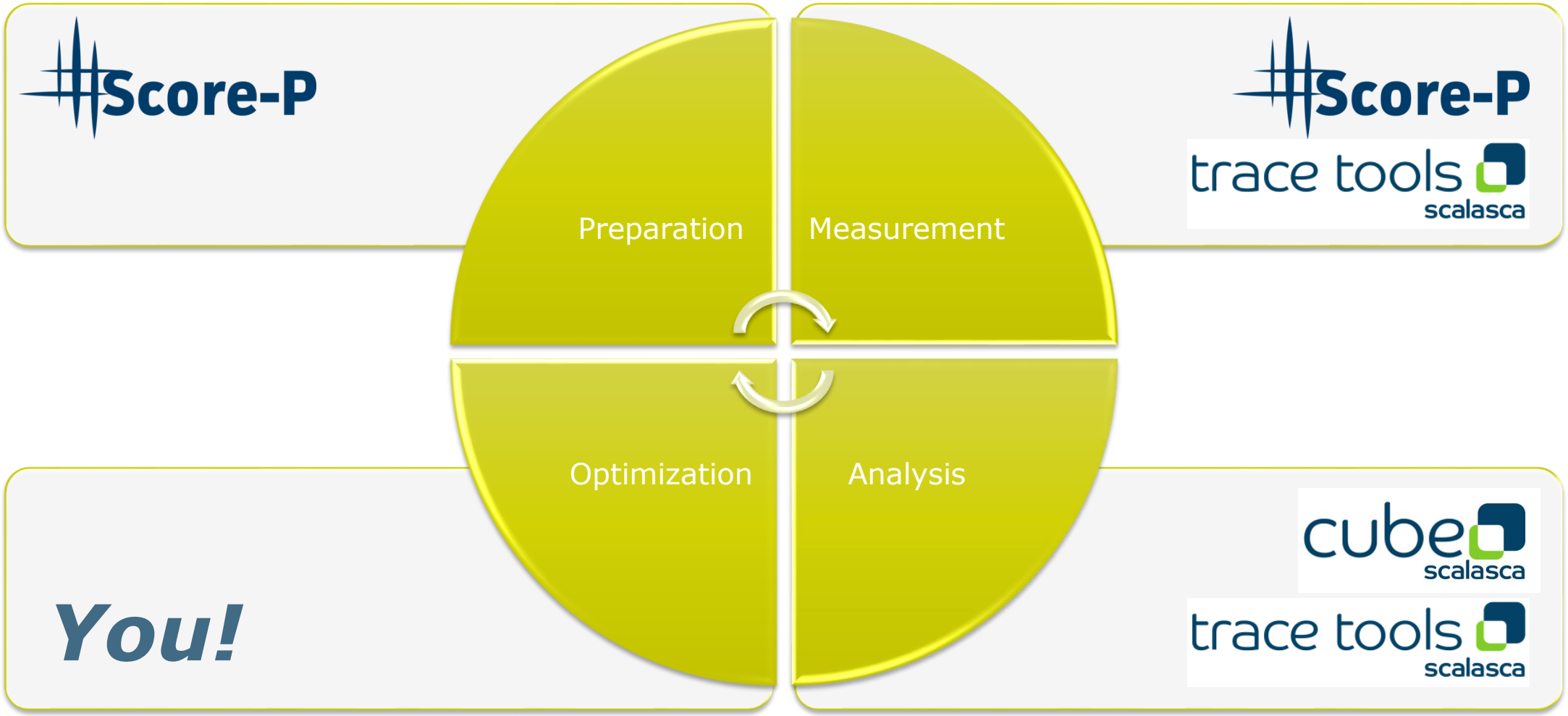
Parallel Performance Analysis using Scalasca/Score-P/CUBE toolset on Karolina (CPU & GPU)

Brian Wylie
Jülich Supercomputing Centre

IT4I, 4-6 September 2024

Performance engineering workflow



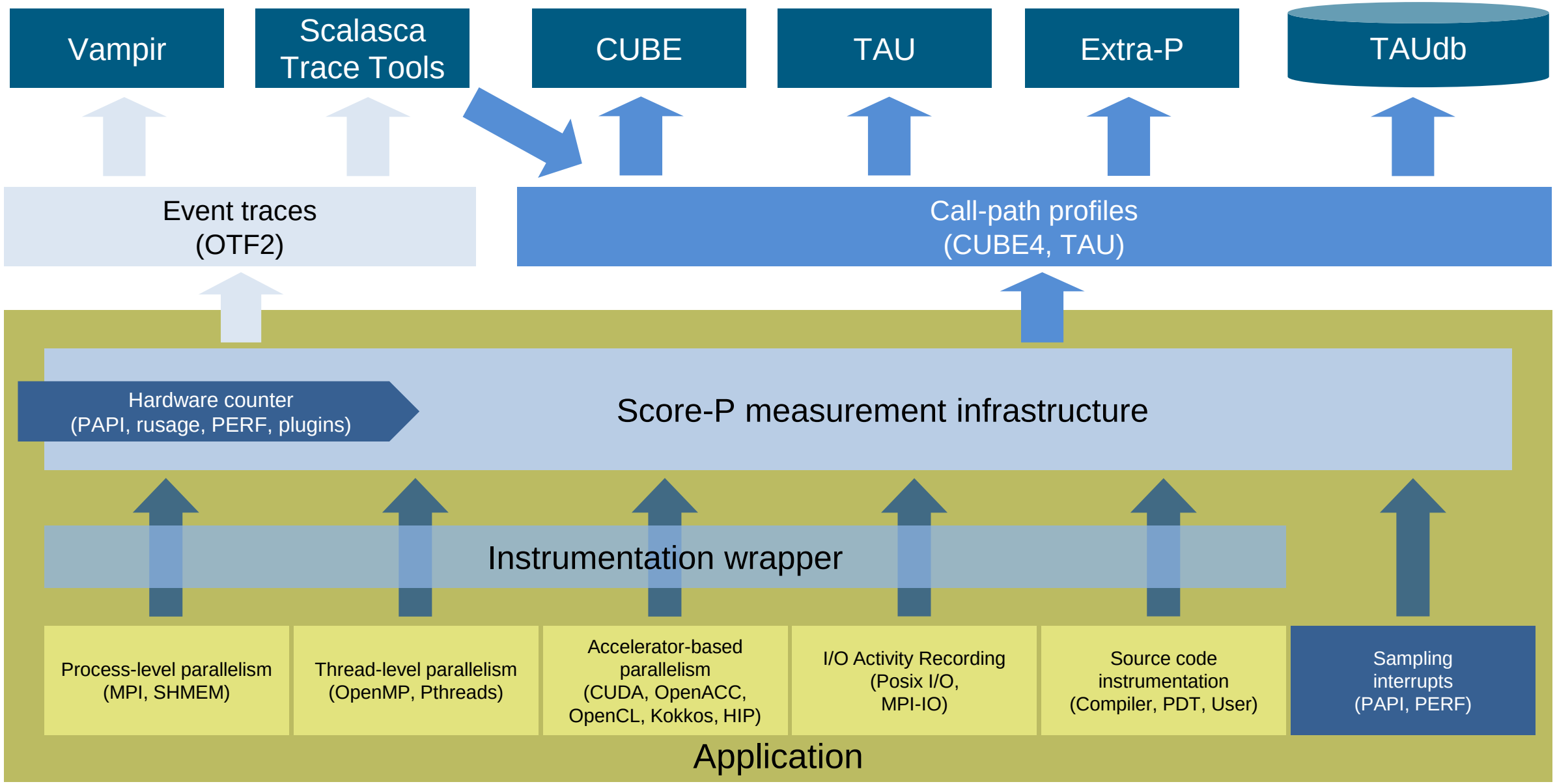
Score-P

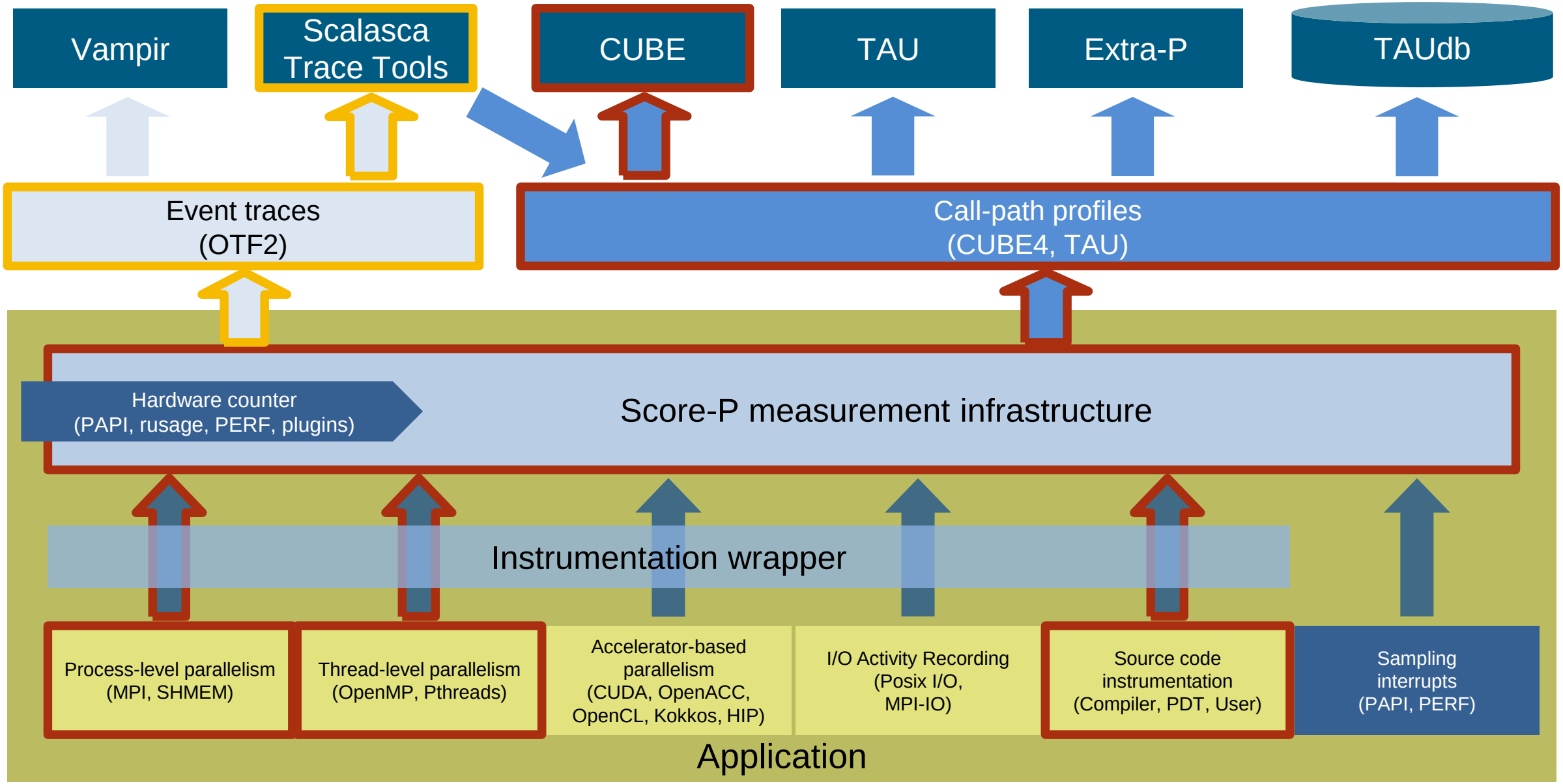
DOI 10.5281/zenodo.1240731

- Infrastructure for instrumentation and performance measurements
- Instrumented application can be used to produce several results:
  - Call-path profiling: CUBE4 data format used for data exchange
  - Event-based tracing: OTF2 data format used for data exchange
- Supported parallel paradigms:
  - Multi-process: MPI, SHMEM
  - Thread-parallel: OpenMP, POSIX threads
  - Accelerator-based: CUDA, HIP, OpenCL, OpenACC

- Initial project funded by BMBF
- Close collaboration with PRIMA project funded by DOE
- Further developed in multiple 3rd-party funded projects

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung


Vampir
Scalasca Trace Tools
CUBE
TAU
Extra-P
TAUdb
Event traces (OTF2)
Call-path profiles (CUBE4, TAU)
Hardware counter (PAPI, rusage, PERF, plugins)
Score-P measurement infrastructure
Instrumentation wrapper
Process-level parallelism (MPI, SHMEM)
Thread-level parallelism (OpenMP, Pthreads)
Accelerator-based parallelism (CUDA, OpenACC, OpenCL, Kokkos, HIP)
I/O Activity Recording (Posix I/O, MPI-IO)
Source code instrumentation (Compiler, PDT, User)
Sampling interrupts (PAPI, PERF)
Application


Vampir
Scalasca Trace Tools
CUBE
TAU
Extra-P
TAUdb
Event traces (OTF2)
Call-path profiles (CUBE4, TAU)
Hardware counter (PAPI, rusage, PERF, plugins)
Score-P measurement infrastructure
Instrumentation wrapper
Process-level parallelism (MPI, SHMEM)
Thread-level parallelism (OpenMP, Pthreads)
Accelerator-based parallelism (CUDA, OpenACC, OpenCL, Kokkos, HIP)
I/O Activity Recording (Posix I/O, MPI-IO)
Source code instrumentation (Compiler, PDT, User)
Sampling interrupts (PAPI, PERF)
Application

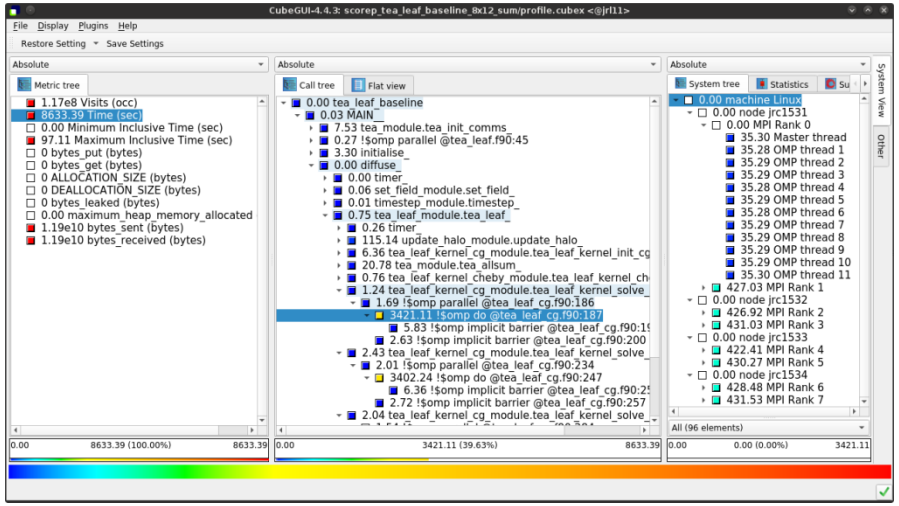
Score-P features

- Open source: 3-clause BSD license
  - Commitment to joint long-term cooperation
  - Development based on meritocratic governance model
  - Open for contributions and new partners
- Portability: supports all major HPC platforms
- Scalability: successful measurements with >1M threads
- Functionality:
  - Generation of call-path profiles and event traces (supporting highly scalable I/O)
  - Using direct instrumentation and sampling
  - Flexible measurement configuration without re-compilation
  - Recording of time, visits, communication data, hardware counters
  - Support for MPI, SHMEM, OpenMP, Pthreads, CUDA, HIP, OpenCL, OpenACC and valid combinations
- Latest release: Score-P 8.4 (Mar 2024)

cube scalasca

CubeLib DOI 10.5281/zenodo.1248078
CubeGUI DOI 10.5281/zenodo.1248087

- Parallel program analysis report exploration tools
  - Libraries for XML+binary report reading & writing
  - Algebra utilities for report processing
  - GUI for interactive analysis exploration
    - Requires Qt ≥ 5
- Originally developed as part of the Scalasca toolset
- Now available as separate components
  - Can be installed independently of Score-P and Scalasca, e.g., on laptop/desktop
  - Latest releases: Cube v4.8.2 (Sep 2023)



Note: source distribution tarballs for Linux, as well as binary packages provided for Linux, Windows & MacOS, from **www.scalasca.org** website in Software/Cube 4.x

Cube GUI (karolina)

mailto: **scalasca@fz-juelich.de**

- Run ***remote*** *(often convenient)*
  - start X server (e.g., Xming) locally, or use alternative such as mobaXterm
  - connect to Karolina with X forwarding enabled
    - **-Y** may be faster but is insecure!
  - load module and start cube remotely

```
desk$ ssh -X login.karolina.it4i.cz
Welcome to Karolina…
karolina$ module load CubeGUI
karolina$ cube ./scorep_sum/profile.cubex
```

Sample measurements (CUBE files) on Karolina:
/mnt/proj2/dd-24-88/jsc/samples

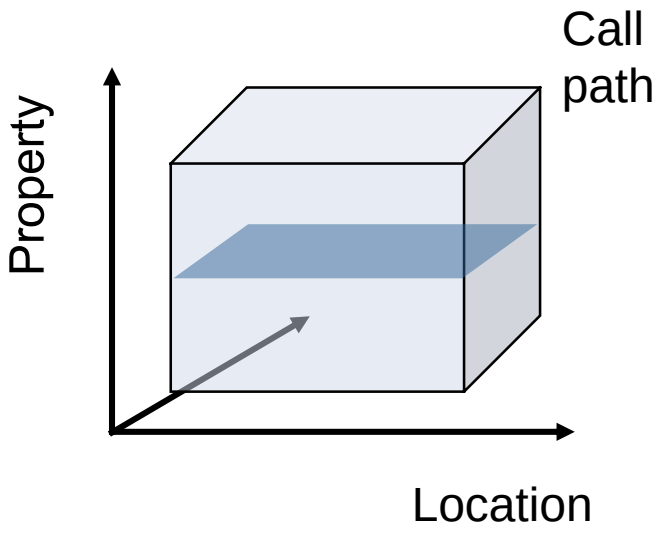
- Install & run ***local*** *(recommended)*
  - install Cube GUI locally on desktop
    - binary packages available for MacOS & Windows and externally provided by OpenHPC and various Linux distributions
    - source package available for Linux, requires Qt
      - configure/build/install manually or use your favourite framework (e.g. Spack or EasyBuild)
  - copy .cubex file (or entire scorep directory) to desktop from remote system
    OR locally mount remote filesystem
  - start cube locally

```
desk$ mkdir $HOME/mnt
desk$ sshfs [user@]remote.sys:[dir] $HOME/mnt
desk$ cd $HOME/mnt
desk$ cube ./scorep_sum/profile.cubex
```

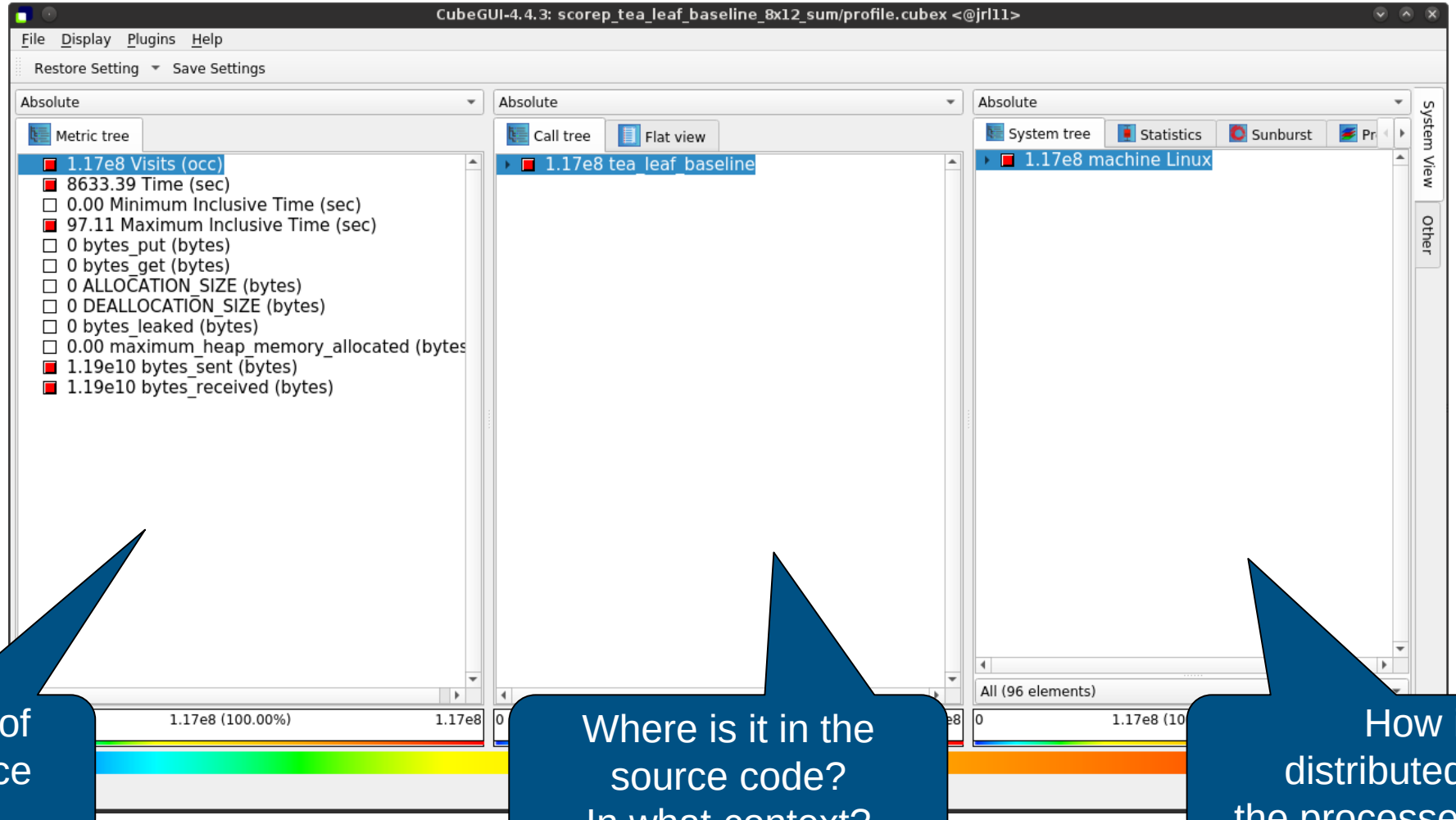
https://www.**scalasca.org**/scalasca/software/cube-4.x/download.html

Analysis presentation and exploration

- Representation of values (severity matrix) on three hierarchical axes
  - Performance property (metric)
  - Call path (program location)
  - System location (process/thread)
- Three coupled tree browsers
- Cube displays severities
  - As *value*: for precise comparison
  - As *colour*: for easy identification of hotspots
  - *Inclusive* value when closed & *exclusive* value when expanded
  - Customizable via display modes



Plain summary analysis report (opening view)

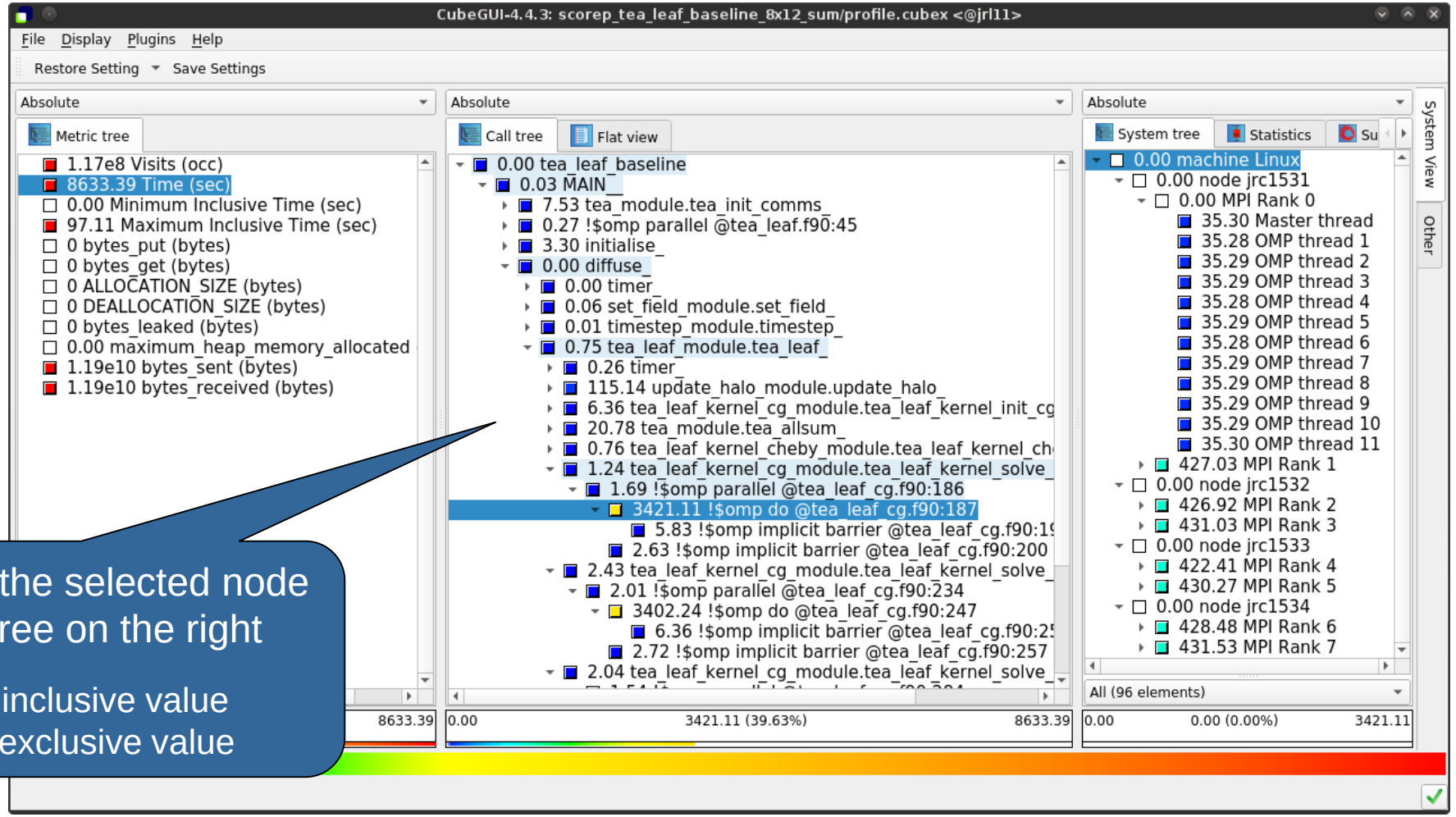


What kind of performance metric?

Where is it in the source code? In what context?

How is it distributed across the processes/threads?

Plain summary analysis report (expanded call tree/system tree)

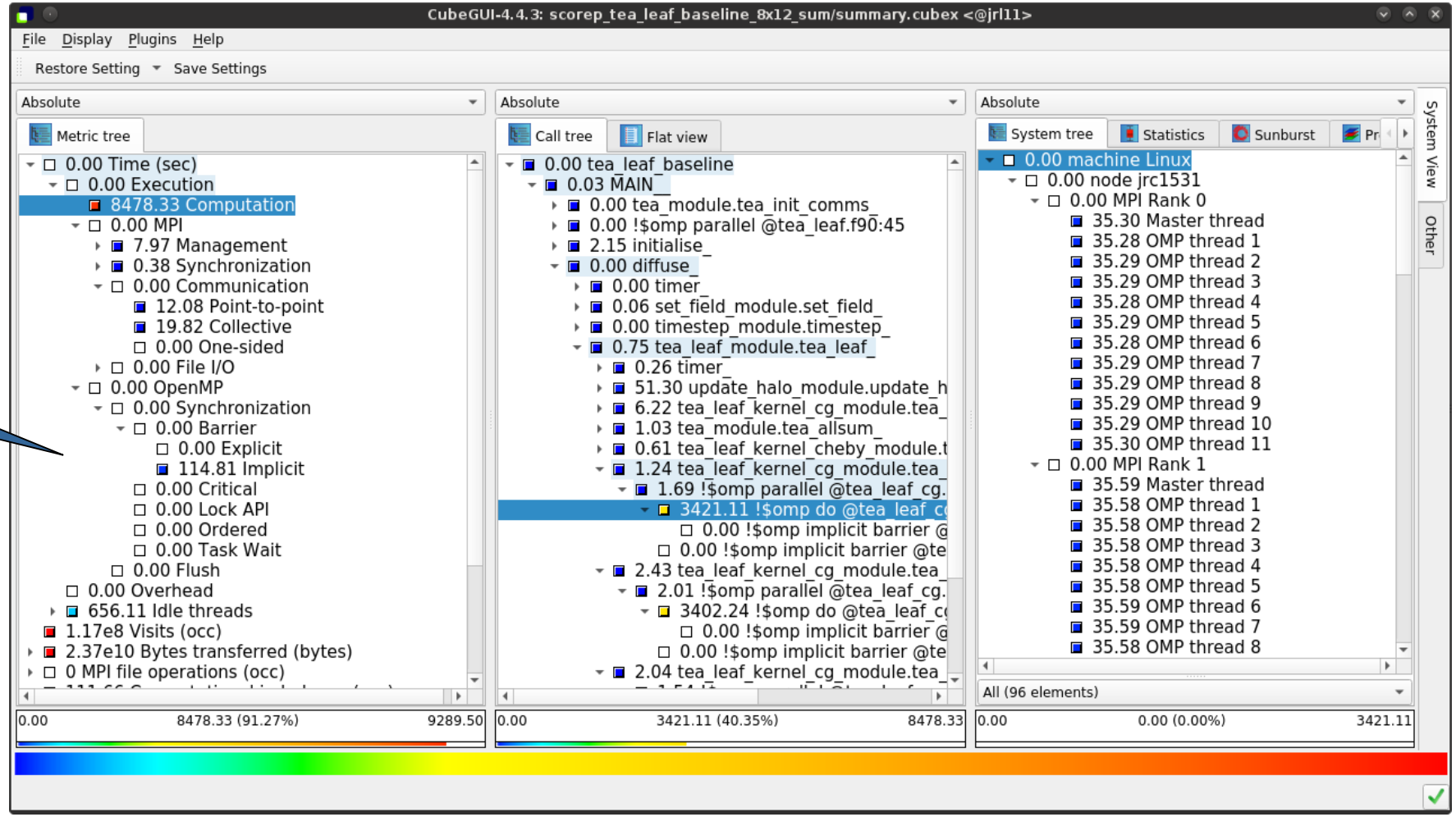


Distribution of the selected node across the tree on the right

Collapsed: inclusive value
Expanded: exclusive value

Post-processed summary analysis report (Scalasca)

Split base metrics from plain report into hierarchy of more specific metrics



trace tools scalasca

DOI 10.5281/zenodo.4103922

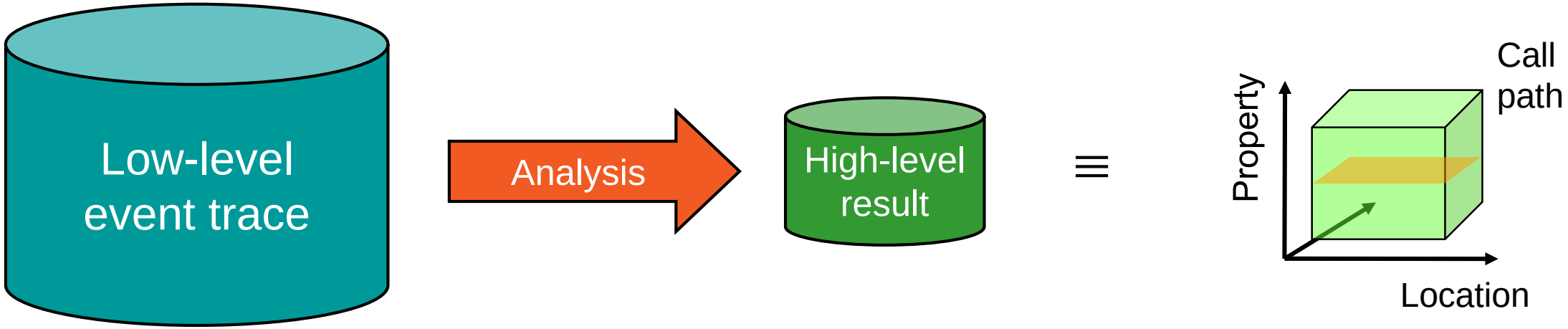
- **Scalable trace-based** performance analysis toolset for the most popular parallel programming paradigms
  - Current focus: MPI, OpenMP, and (to a limited extend) POSIX threads
  - Analysis of traces including only host-side events from applications using CUDA, OpenCL, or OpenACC (also in combination with MPI and/or OpenMP) is possible, but results need to be interpreted with some care

- Specifically targeting large-scale parallel applications
  - Demonstrated scalability up to 1.8 million parallel threads
  - Of course also works at small/medium scale

- Latest release:
  - Scalasca Trace Tools v2.6.1 (Dec 2022)

Automatic trace analysis

- Idea
  - Automatic search for patterns of inefficient behavior
  - 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

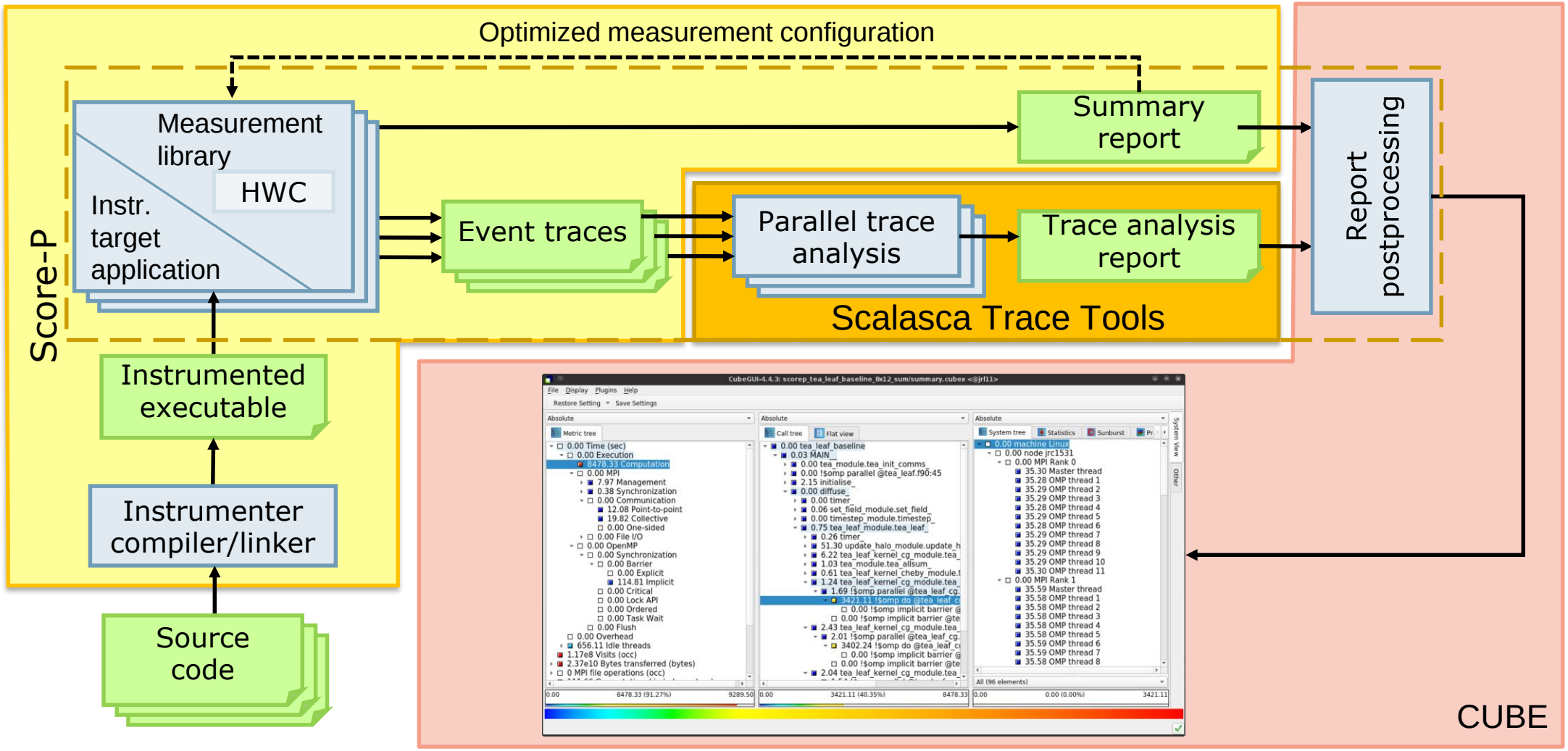
Scalasca Trace Tools features

- Open source: 3-clause BSD license
- Portability: supports all major HPC platforms
- Scalability: successful analyses with >1M threads

- Uses Score-P instrumenter & measurement libraries
  - Scalasca v2 core package focuses on trace-based analyses
  - Provides convenience commands for measurement, analysis, and postprocessing
  - 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 memory events, CUDA/HIP/OpenCL device events (kernel, memcpy), SHMEM, or OpenMP nested parallelism
  - PAPI/rusage metrics for trace events are ignored

Putting it all together



Outline

Day 0: (Wednesday 4 September)
- Example POP assessments using Scalasca toolset

Day 1: (Thursday 5 September)
- Instrumentation & measurement with Score-P
- Execution profile analysis examination with CUBE
- Refinement via scoring & measurement filtering

Day 2: (Friday 6 September)
- Automated trace collection & analysis with Scalasca
- Interactive trace analysis with Vampir
- GPU/custom measurements & analyses

Morning sessions (10:45-12:00):
- Presentation / demonstration of tools using examples on Karolina
  - universal CPU partition & accelerated GPU partition

Afternoon sessions (13:00-16:45):
- Guided assistance to apply tools to your own application code(s) or provided examples