Scalasca Trace Tools

A scalable trace-based performance analysis toolset

Markus Geimer
Jülich Supercomputing Centre

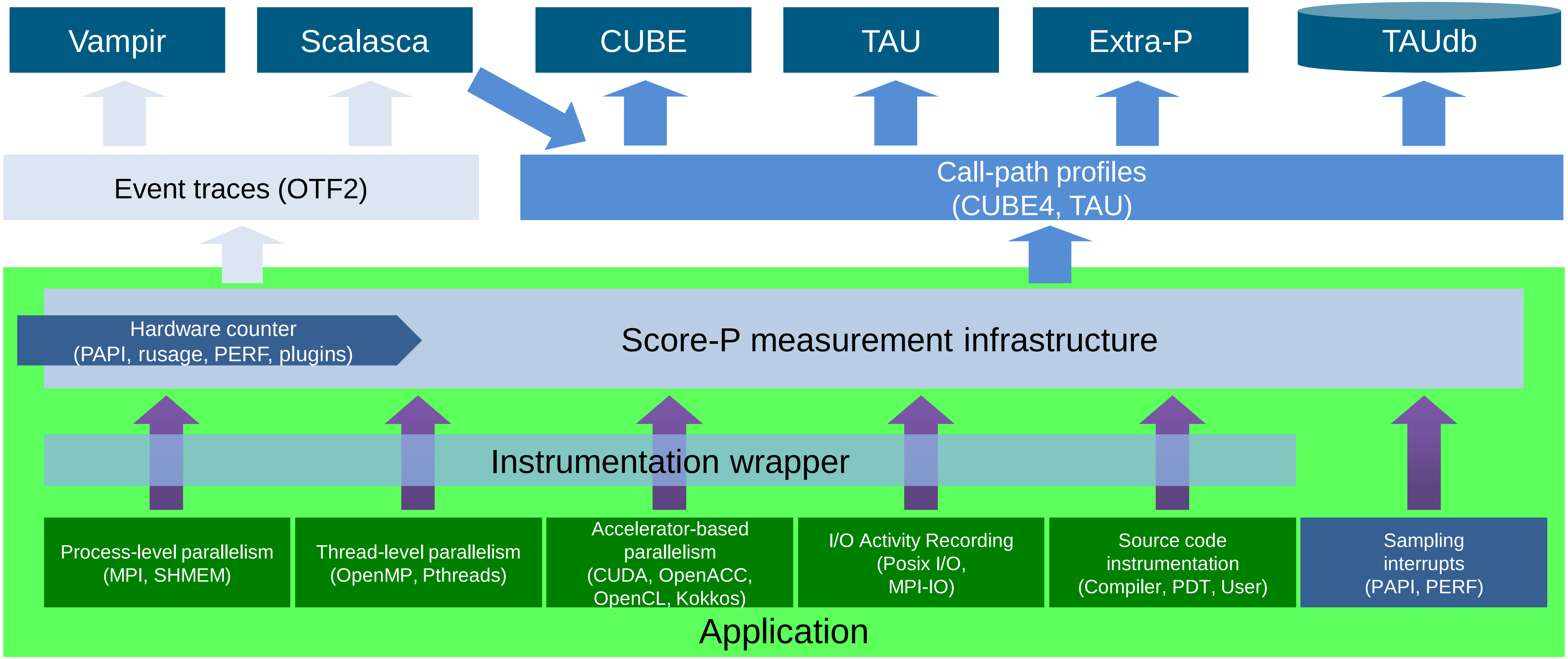
trace tools
scalasca

Scalasca Trace Tools

DOI 10.5281/zenodo.4700519

- **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 (April 2021), coordinated with Score-P v7.0

Score-P Ecosystem Overview



Vampir

Scalasca

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)

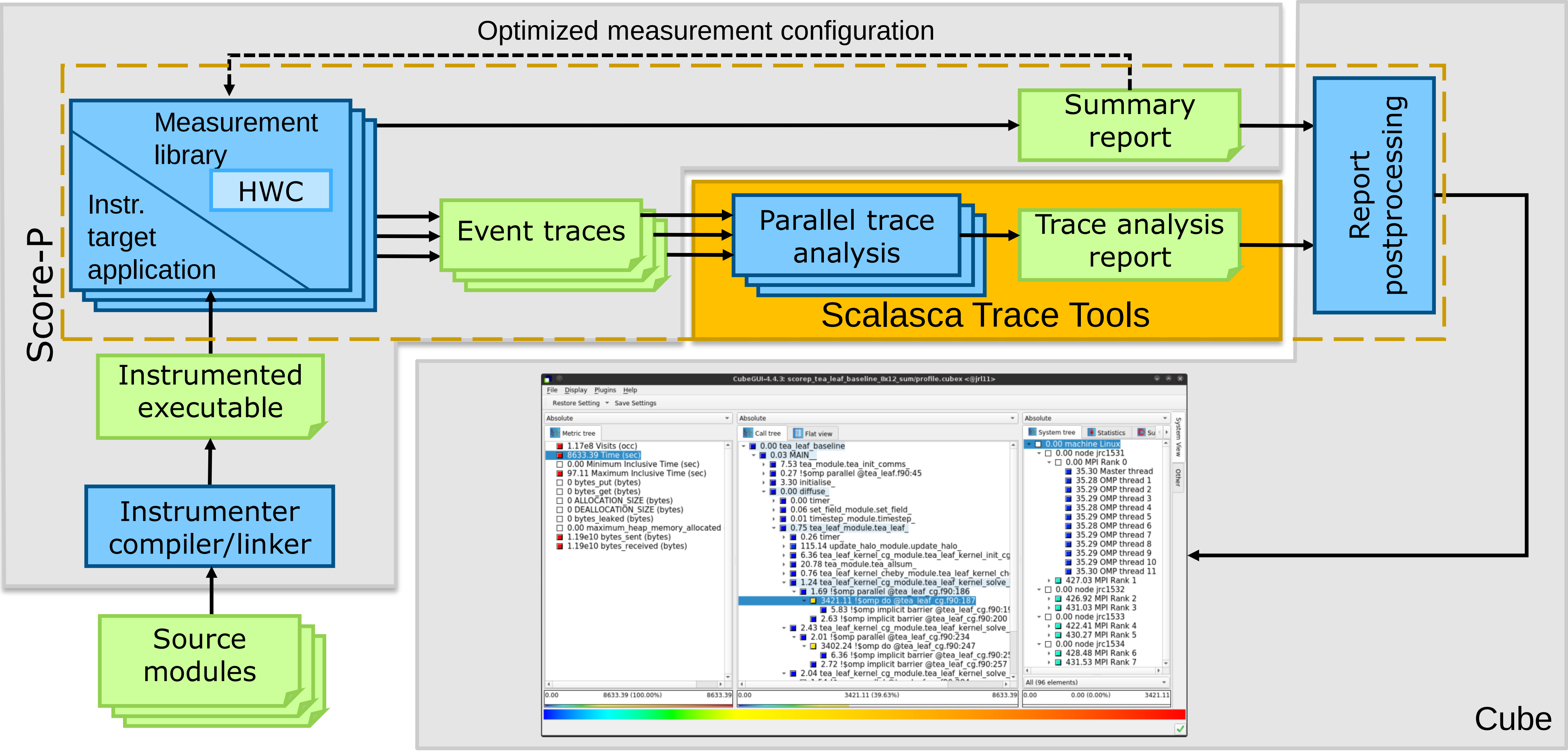
I/O Activity Recording (Posix I/O, MPI-IO)

Source code instrumentation (Compiler, PDT, User)

Sampling interrupts (PAPI, PERF)

Application

Scalasca Trace Tools Workflow



Optimized measurement configuration

Score-P

Measurement library

HWC

Instr. target application

Summary report

Event traces

Parallel trace analysis

Trace analysis report

Scalasca Trace Tools

Report postprocessing

Instrumented executable

Instrumenter compiler/linker

Source modules

Cube

CubeGUI-4.4.3: scorep_tea_leaf_baseline_8x12_sum/profile.cubex <@jrl11>

File Display Plugins Help

Restore Setting ▾ Save Settings

Absolute ▾

Metric tree

```
🟥 1.17e8 Visits (occ)
🟥 8633.39 Time (sec)
□ 0.00 Minimum Inclusive Time (sec)
🟥 97.11 Maximum Inclusive Time (sec)
□ 0 bytes_put (bytes)
□ 0 bytes_get (bytes)
□ 0 ALLOCATION_SIZE (bytes)
□ 0 DEALLOCATION_SIZE (bytes)
□ 0 bytes_leaked (bytes)
□ 0.00 maximum_heap_memory_allocated
🟥 1.19e10 bytes_sent (bytes)
🟥 1.19e10 bytes_received (bytes)
```

0.00 8633.39 (100.00%) 8633.39

Absolute ▾

Call tree | Flat view

```
▾ 🟦 0.00 tea_leaf_baseline
  ▾ 🟦 0.03 MAIN__
    ▸ 🟦 7.53 tea_module.tea_init_comms_
    ▸ 🟦 0.27 !$omp parallel @tea_leaf.f90:45
    ▸ 🟦 3.30 initialise_
    ▾ 🟦 0.00 diffuse_
      ▸ 🟦 0.00 timer_
      ▸ 🟦 0.06 set_field_module.set_field_
      ▸ 🟦 0.01 timestep_module.timestep_
      ▾ 🟦 0.75 tea_leaf_module.tea_leaf_
        ▸ 🟦 0.26 timer_
        ▸ 🟦 115.14 update_halo_module.update_halo_
        ▸ 🟦 6.36 tea_leaf_kernel_cg_module.tea_leaf_kernel_init_cg
        ▸ 🟦 20.78 tea_module.tea_allsum_
        ▸ 🟦 0.76 tea_leaf_kernel_cheby_module.tea_leaf_kernel_ch
        ▾ 🟦 1.24 tea_leaf_kernel_cg_module.tea_leaf_kernel_solve_
          ▾ 🟦 1.69 !$omp parallel @tea_leaf_cg.f90:186
            ▾ 🟨 3421.11 !$omp do @tea_leaf_cg.f90:187
                🟦 5.83 !$omp implicit barrier @tea_leaf_cg.f90:19
              🟦 2.63 !$omp implicit barrier @tea_leaf_cg.f90:200
        ▾ 🟦 2.43 tea_leaf_kernel_cg_module.tea_leaf_kernel_solve_
          ▾ 🟦 2.01 !$omp parallel @tea_leaf_cg.f90:234
            ▾ 🟨 3402.24 !$omp do @tea_leaf_cg.f90:247
                🟦 6.36 !$omp implicit barrier @tea_leaf_cg.f90:25
              🟦 2.72 !$omp implicit barrier @tea_leaf_cg.f90:257
        ▾ 🟦 2.04 tea_leaf_kernel_cg_module.tea_leaf_kernel_solve_
```

0.00 3421.11 (39.63%) 8633.39

Absolute ▾

System tree | Statistics | Su

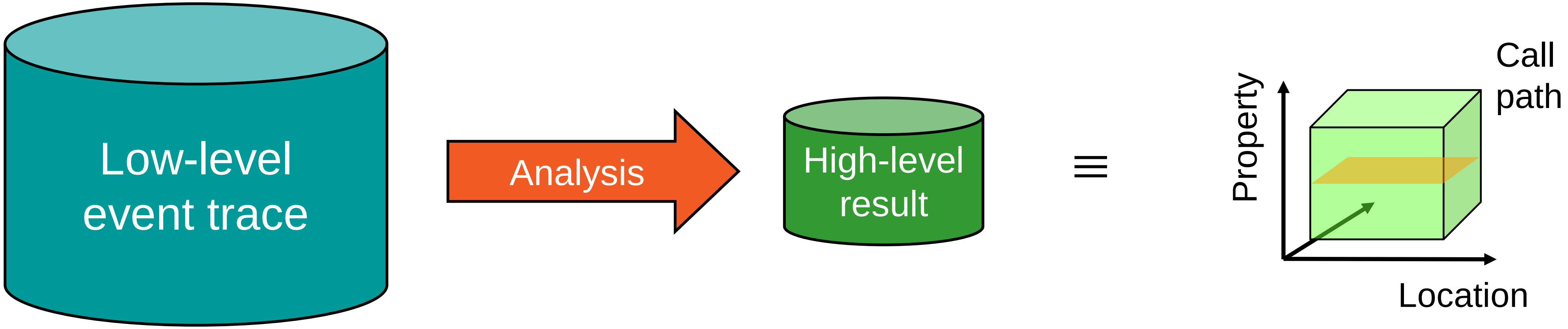
```
▾ □ 0.00 machine Linux
  ▾ □ 0.00 node jrc1531
    ▾ □ 0.00 MPI Rank 0
        🟦 35.30 Master thread
        🟦 35.28 OMP thread 1
        🟦 35.29 OMP thread 2
        🟦 35.29 OMP thread 3
        🟦 35.28 OMP thread 4
        🟦 35.29 OMP thread 5
        🟦 35.28 OMP thread 6
        🟦 35.29 OMP thread 7
        🟦 35.29 OMP thread 8
        🟦 35.29 OMP thread 9
        🟦 35.29 OMP thread 10
        🟦 35.30 OMP thread 11
    ▸ 🟩 427.03 MPI Rank 1
  ▾ □ 0.00 node jrc1532
    ▸ 🟩 426.92 MPI Rank 2
    ▸ 🟩 431.03 MPI Rank 3
  ▾ □ 0.00 node jrc1533
    ▸ 🟩 422.41 MPI Rank 4
    ▸ 🟩 430.27 MPI Rank 5
  ▾ □ 0.00 node jrc1534
    ▸ 🟩 428.48 MPI Rank 6
    ▸ 🟩 431.53 MPI Rank 7
```

All (96 elements) ▾

0.00 0.00 (0.00%) 3421.11

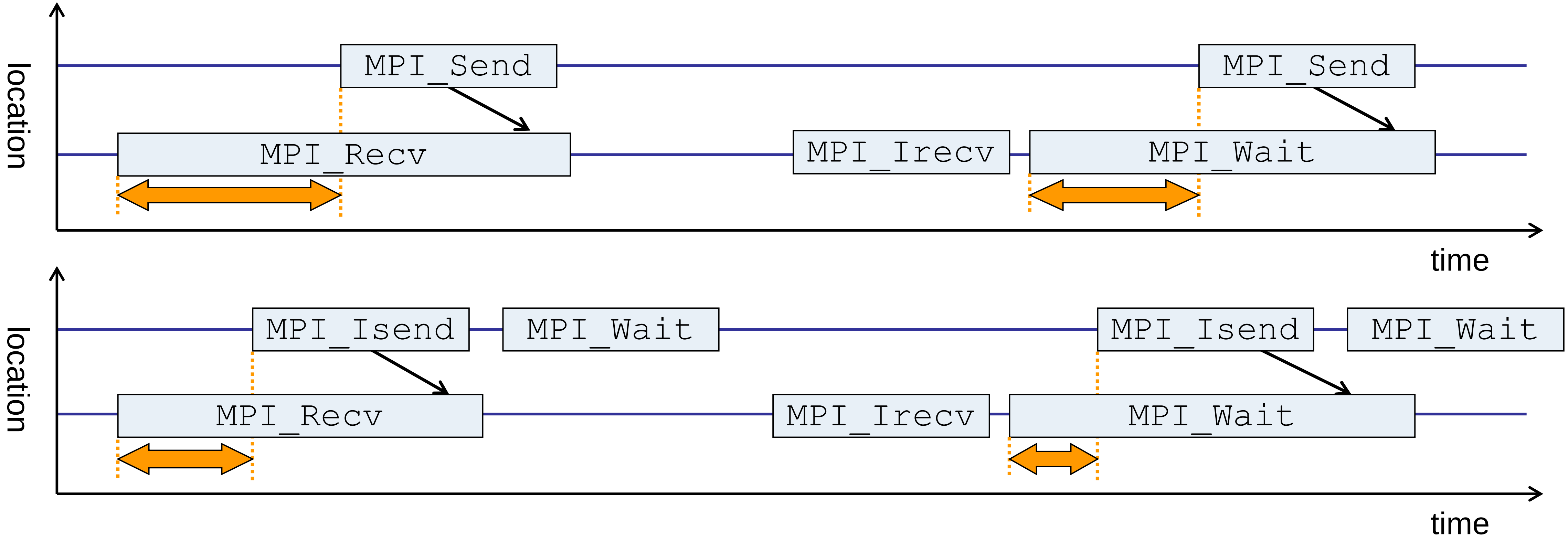
Automatic Trace Analysis

- Idea
  - Automatic search for patterns of inefficient behavior
  - Classification of behavior & quantification of significance
  - Identification of delays as root causes of inefficiencies



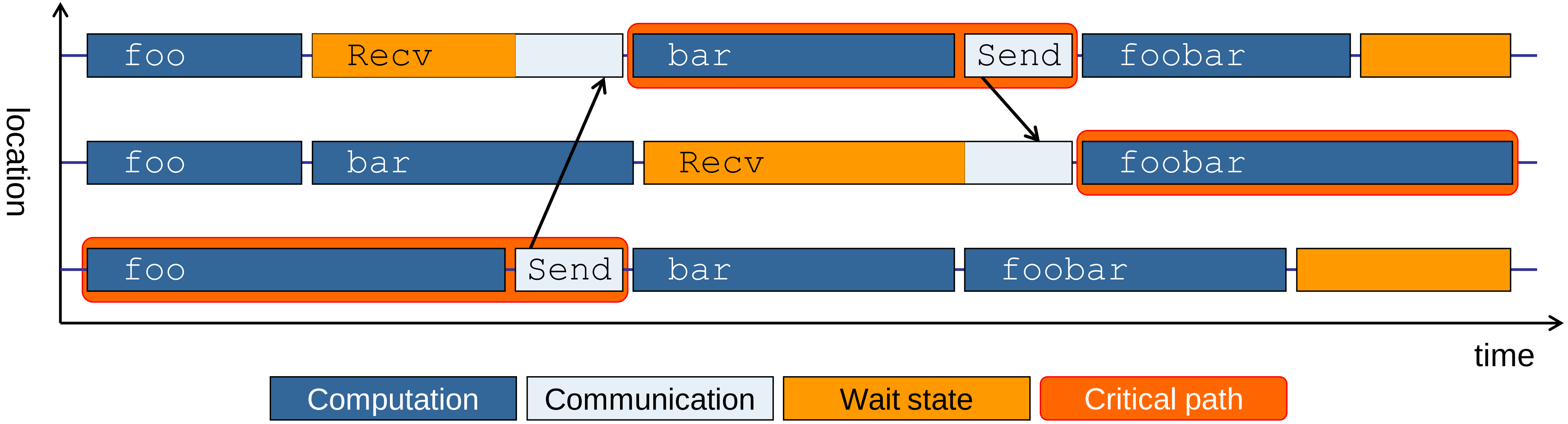
  - Guaranteed to cover the entire event trace
  - Quicker than manual/visual trace analysis
  - Parallel replay analysis exploits available memory & processors to deliver scalability

Example: “*Late Sender*” Wait State



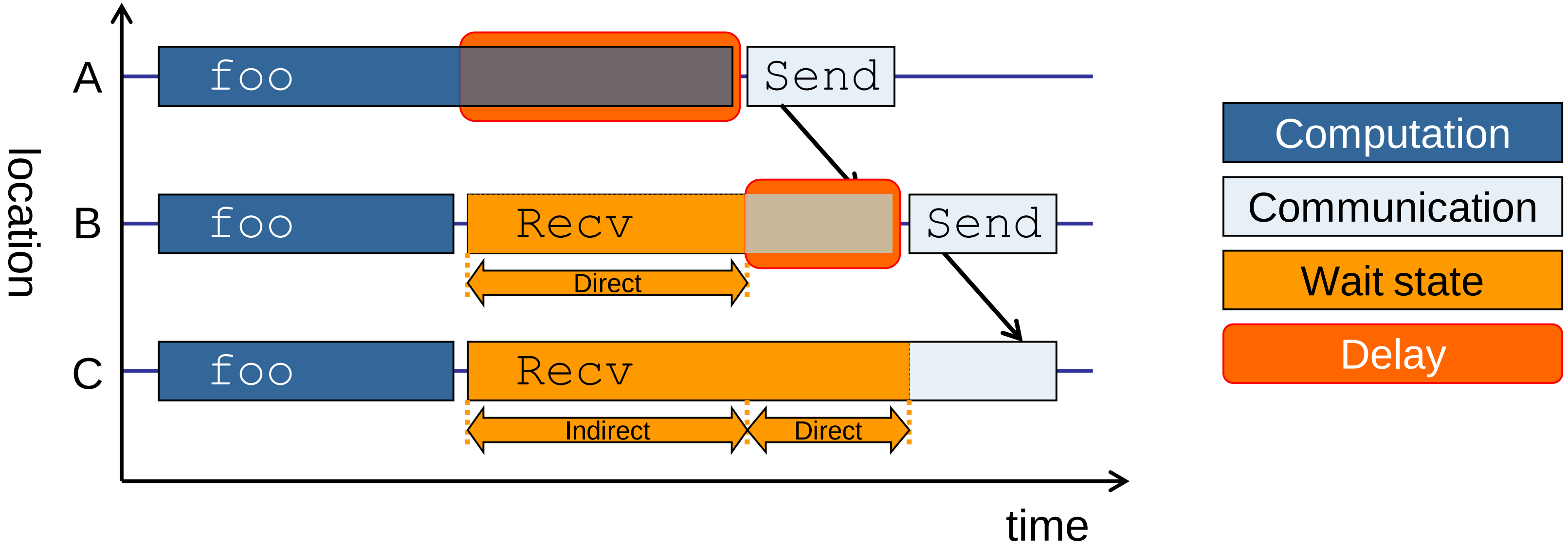
- Waiting time caused by a blocking receive operation posted earlier than the corresponding send
- Applies to blocking as well as non-blocking communication

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*

Objectives for DEEP-SEA

- Extend analysis capabilities
  - Add support for analyzing communication on MPI inter-communicators
- Improve MSA-awareness of analysis
  - Distinguish inter- and intra-module wait states
- Improve MSA-/MPMD-awareness of convenience commands
  - Support automatic invocation of trace analyzer in more cases

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:
  - https://www.scalasca.org
- Contact:
  - mailto: scalasca@fz-juelich.de

trace tools
scalasca

Questions?