The Scalasca Scalable Parallel Performance Analysis Toolset - For POP Assessments and Beyond

Bernd Mohr

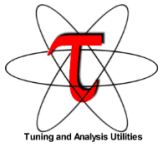
EU H2020 Centre of Excellence (CoE)

1 December 2018 – 30 November 2021

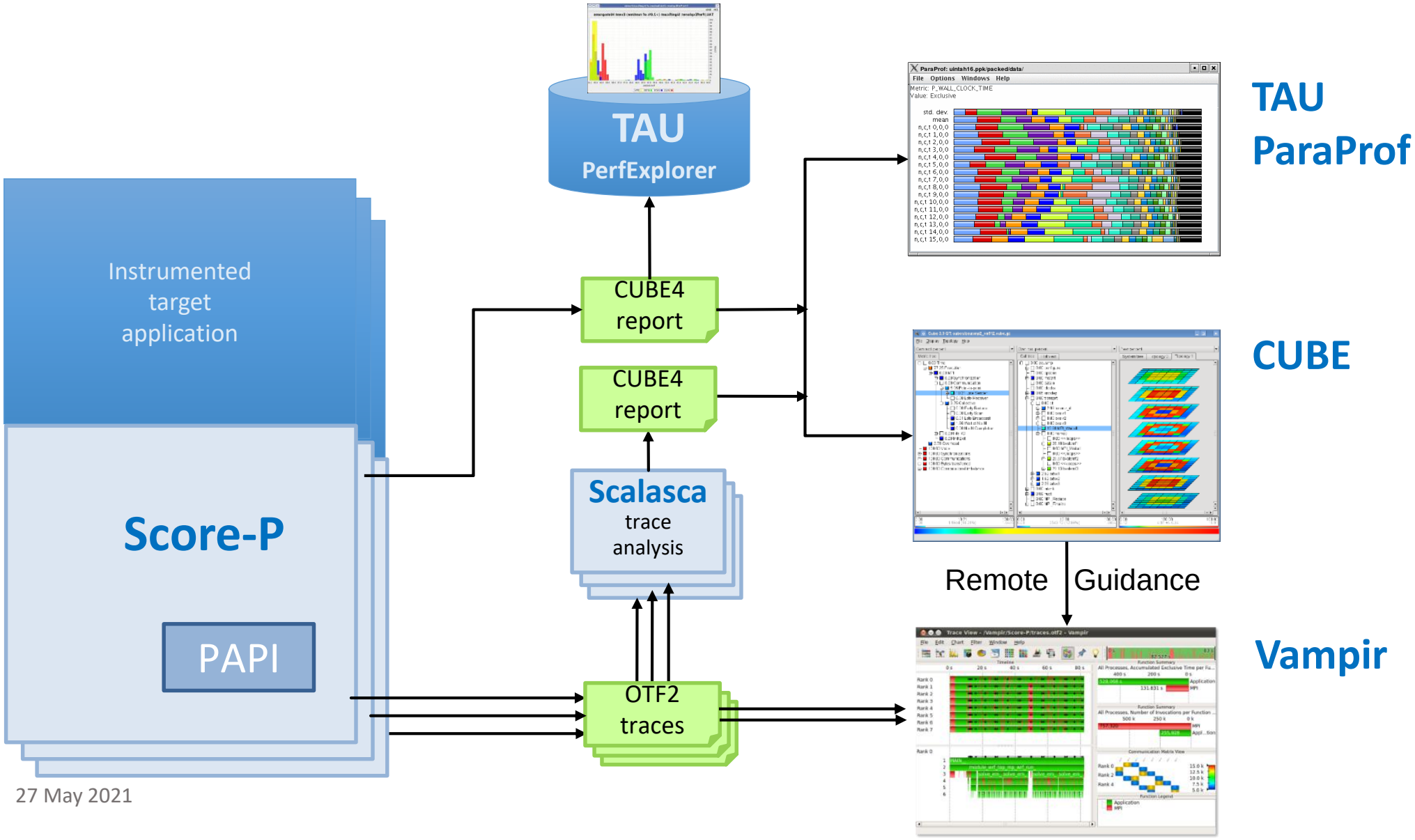
Grant Agreement No 824080

The Score-P Tool Ecosystem

The Context



Score-P Tool Ecosystem

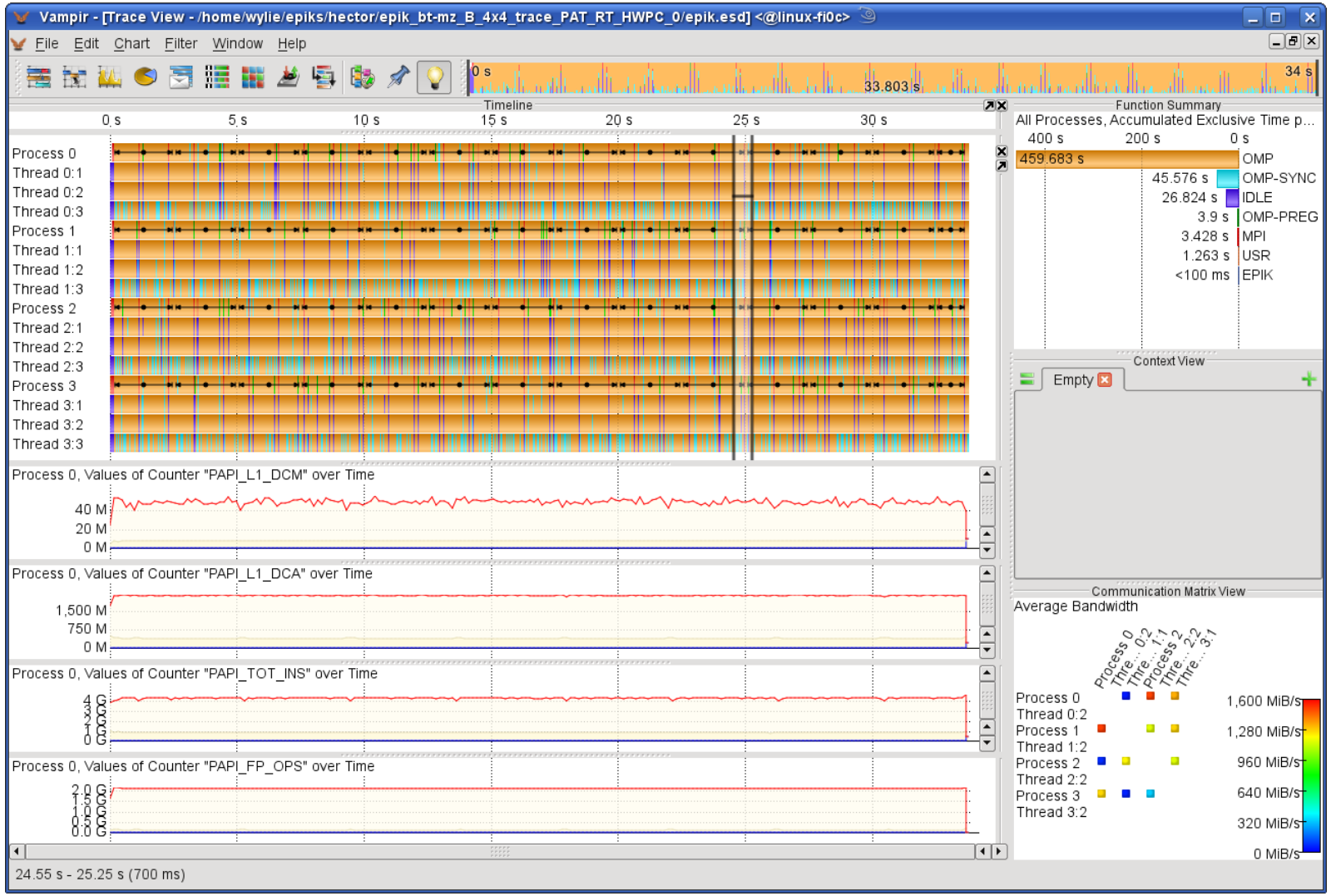


Score-P Performance Instrumentation and Measurement

https://score-p.org/

- Provide typical base functionality for HPC performance tools
- C, C++, Fortran, Python support
- **Instrumentation** (various methods)
  - Multi-process paradigms (MPI, SHMEM)
  - Thread-parallel paradigms (OpenMP, POSIX threads)
  - Accelerator-based paradigms (OpenACC, CUDA, OpenCL, Kokkos)
  - Additional execution information (HW+SW counter, I/O, memory, …)
  - **In any combination!**
- Flexible **measurement** without re-compilation:
  - Basic and advanced **profile** generation (⇨ CUBE4 format)
  - Event **trace** recording (⇨ OTF2 format)
- Highly scalable measurement

Interactive Event Trace Analysis

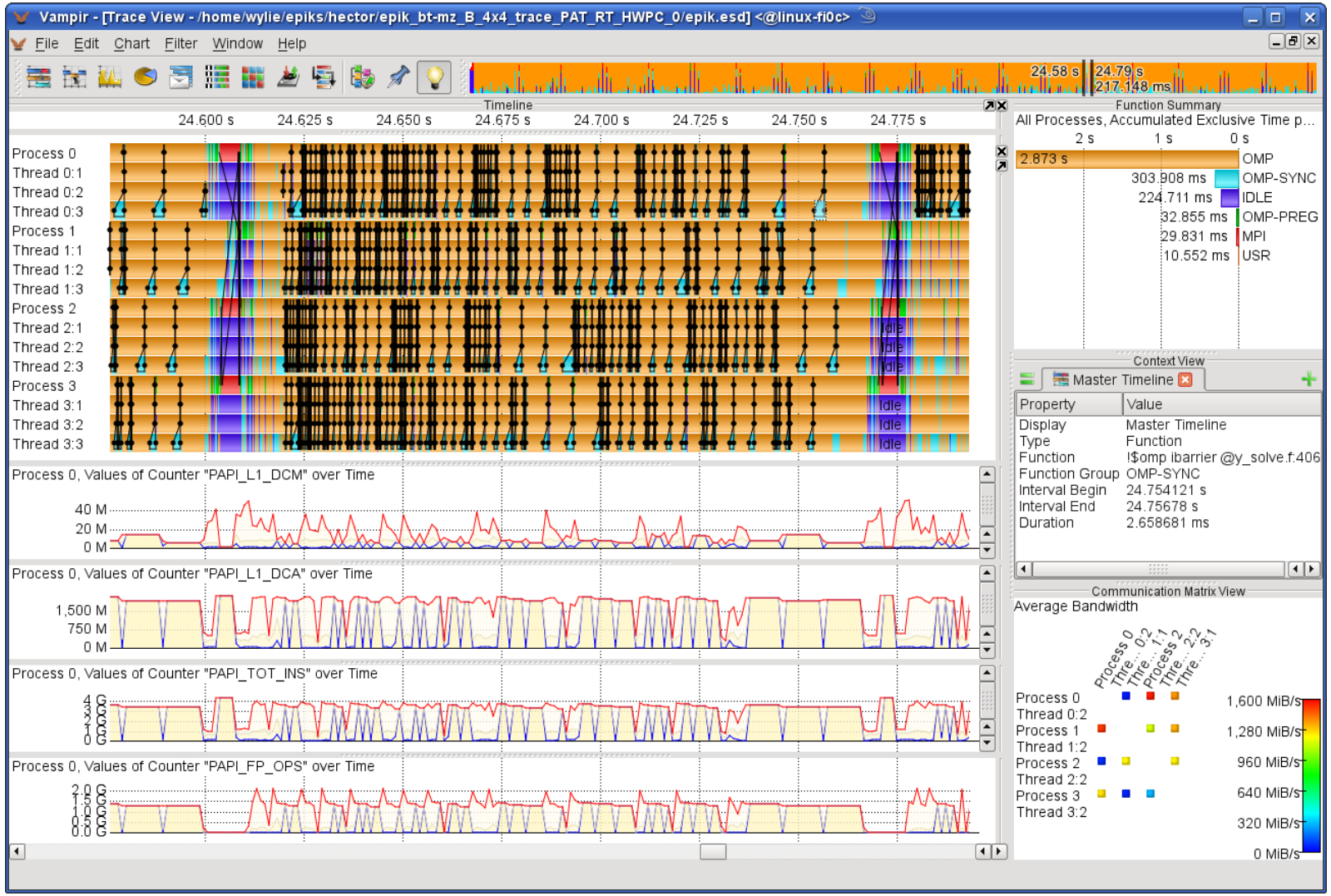


https://vampir.eu/

Visual presentation of dynamic runtime behaviour

- Event timeline chart for states & interactions of processes/threads
- Communication statistics, summaries & more

Interactive Event Trace Analysis



https://vampir.eu/

Interactive browsing, zooming, selecting

- Linked displays & statistics adapt to selected time interval

Trace formats

- OTF (VampirTrace)
- OTF2 (Score-P)
- EPIK (Scalasca1)

TAU Performance System®

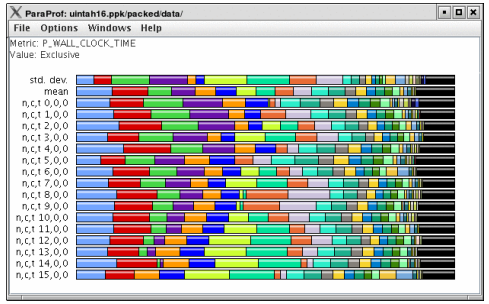
Tuning and Analysis Utilities

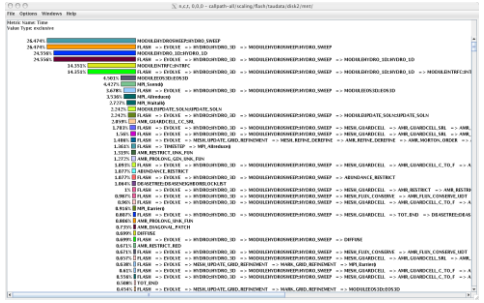
UNIVERSITY OF OREGON

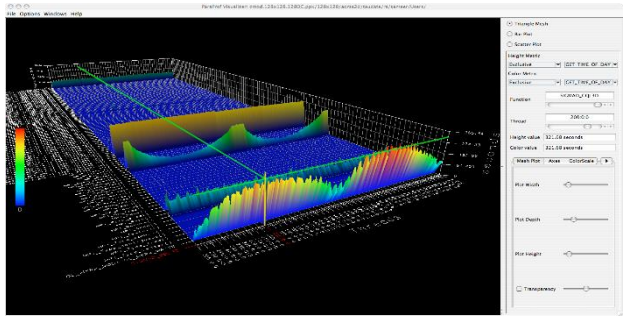
http://tau.uoregon.edu/

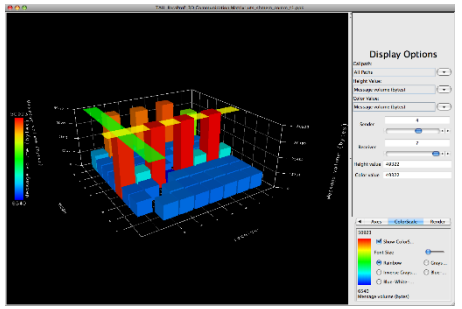
- **Very portable tool set** for instrumentation, measurement and analysis of parallel multi-threaded applications

- Supports
  - **Various profiling modes and tracing**
  - Various forms of **code instrumentation + sampling**
  - C, C++, Fortran, Java, Python
  - Multi-process paradigms (MPI, SHMEM, GPI, ARMCI)
  - Thread-parallel paradigms (OpenMP, POSIX threads)
  - Accelerator-based paradigms (OpenACC, CUDA, OpenCL, HIP/ROCm, OneAPI, Kokkos)





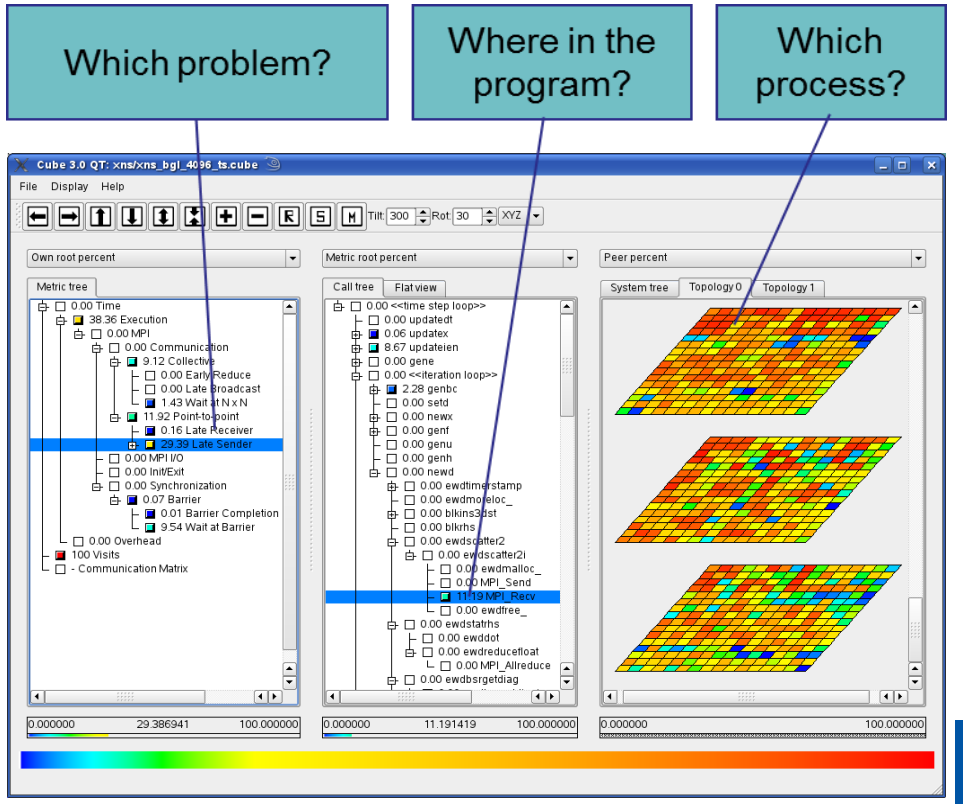




- Scalable Analysis of Large Scale Applications

https://scalasca.org/

- 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

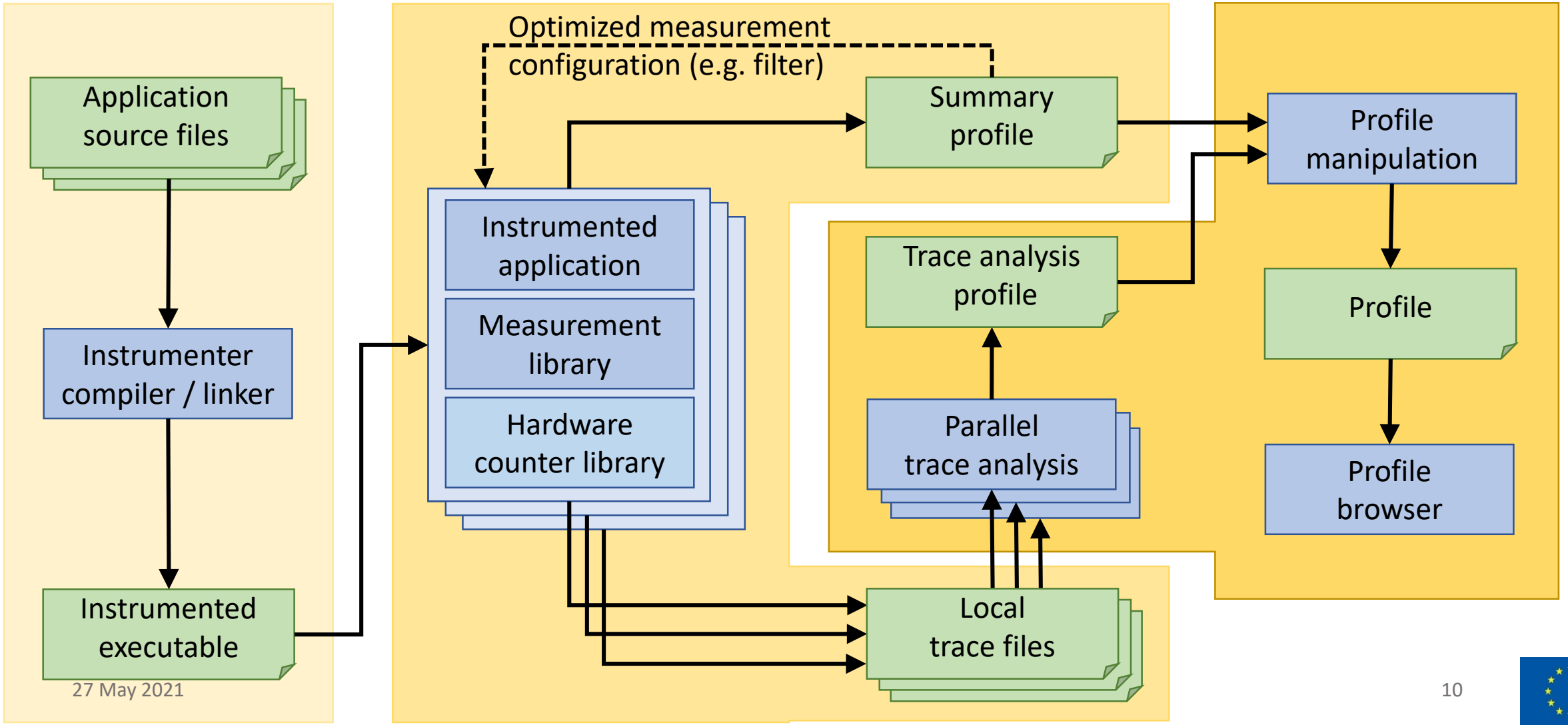


CUBE

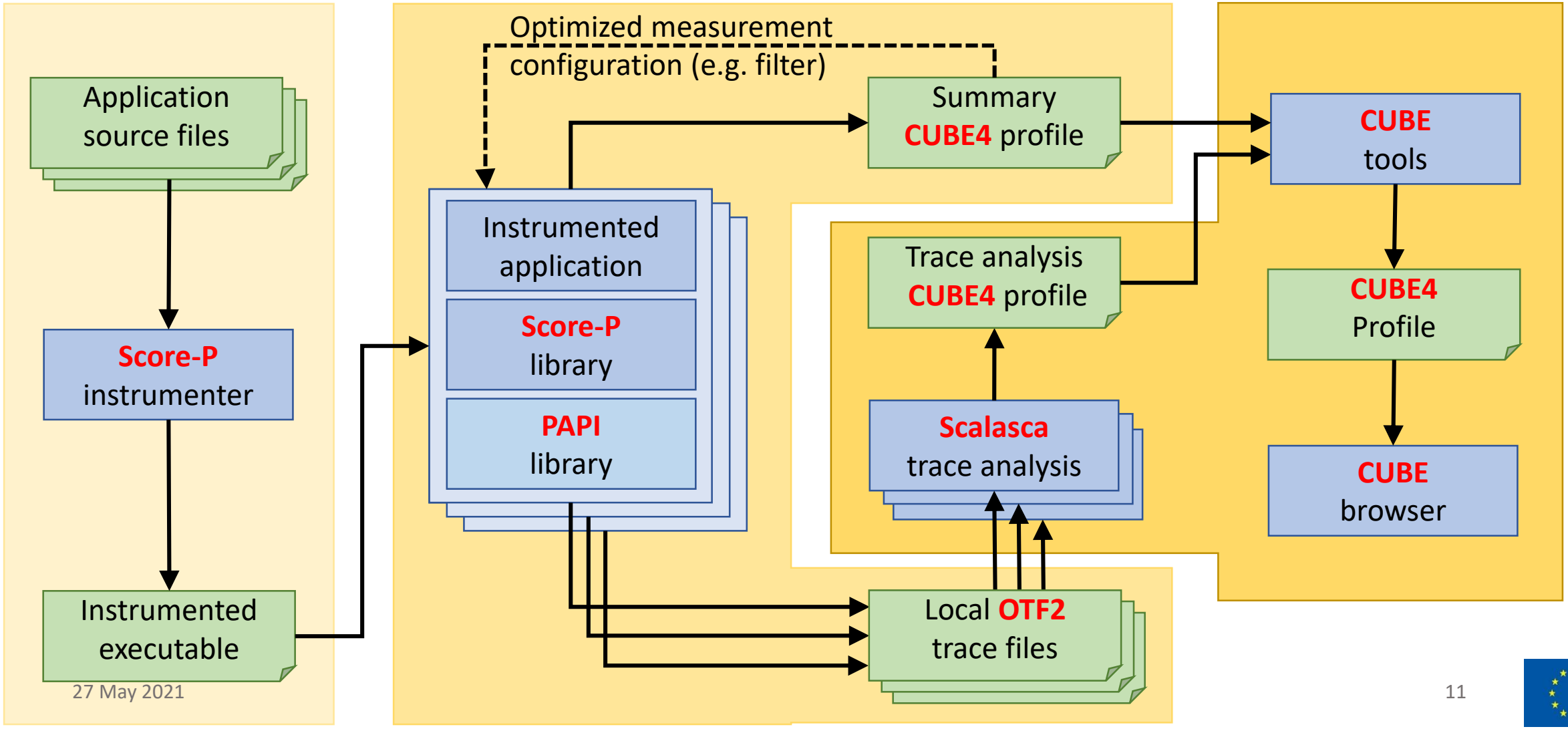
The Scalasca Workflow

The Basics

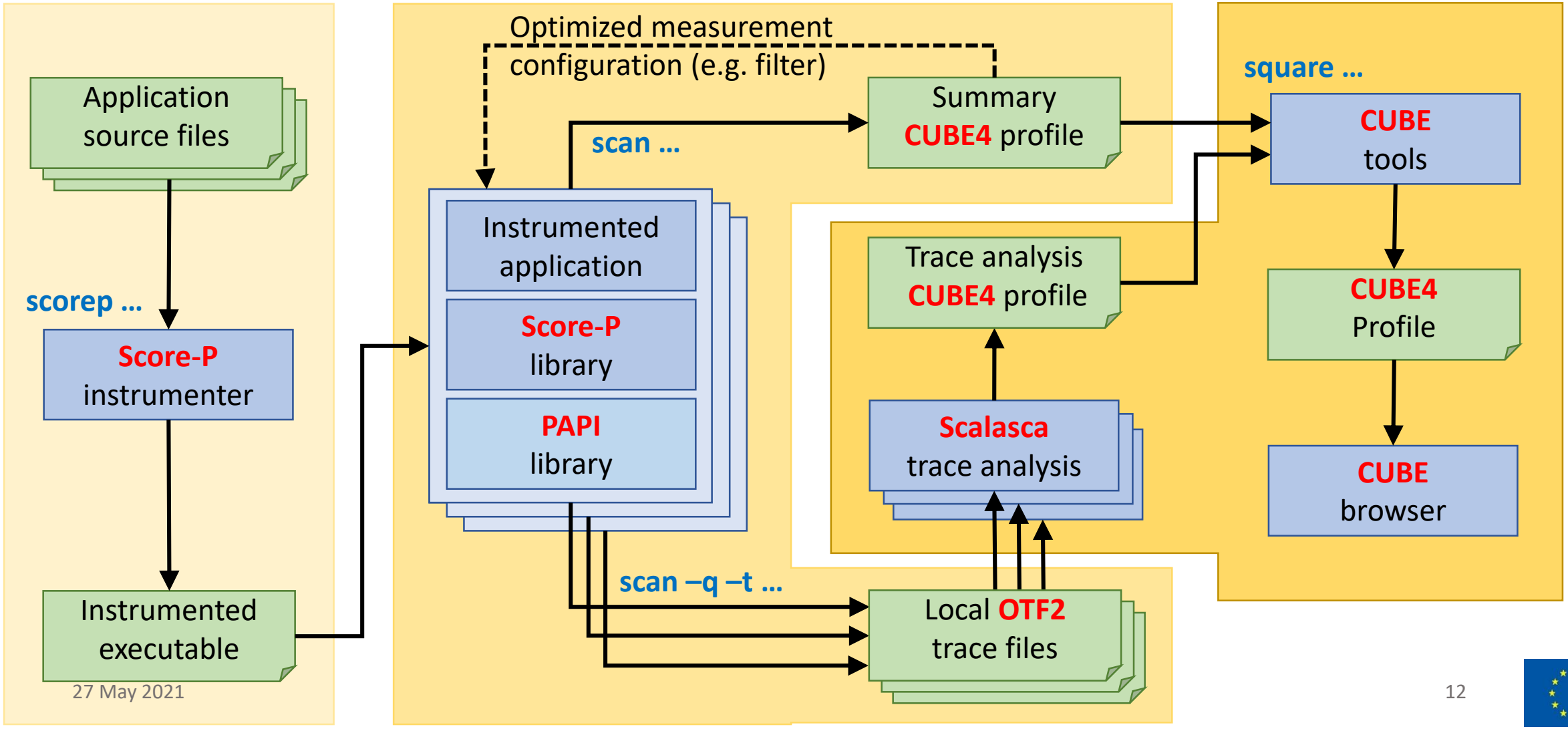
scalasca Performance Analysis Workflow



scalasca Performance Analysis Workflow



scalasca Performance Analysis Workflow



POP Assessments with Scalasca

This is the way

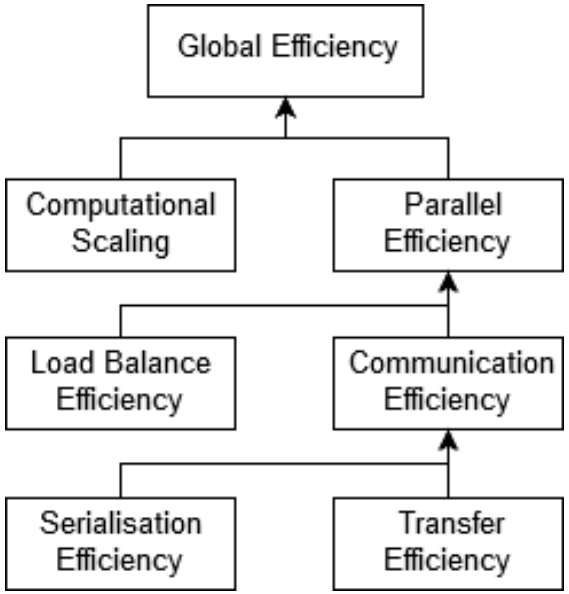
Recap: POP Metrics

https://pop-coe.eu/further-information/learning-material

- **Original (POP1) Metrics**
  - Article explaining the POP Standard Metrics for Parallel Performance Analysis
  - Presentation summarizing the POP Standard Metrics for Parallel Performance Analysis
- **New (POP2) Hybrid Metrics**
  - Introduction explaining the POP2 Standard Metrics for Performance Analysis of Hybrid Parallel Applications
  - Cheat sheet for Additive Hybrid Metrics
  - Cheat sheet for Multiplicative Hybrid Metrics
  - In-depth explanation of the Additive Hybrid Metrics
  - Webinar Identifying Performance Bottlenecks in Hybrid MPI + OpenMP Software

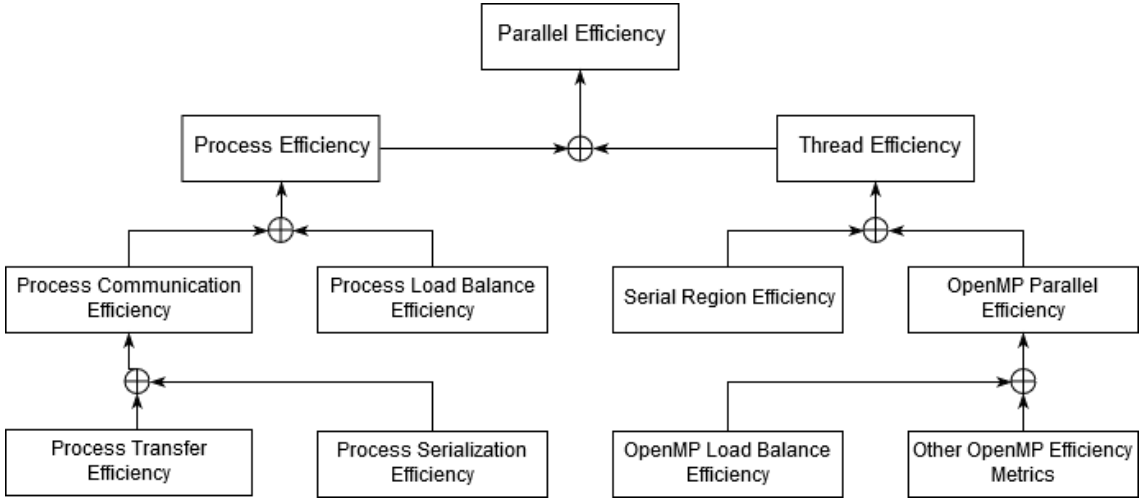
Recap: POP Phase1 (MPI) Metrics

- The following metrics are used in a POP Performance Audit:
- Global Efficiency (GE): GE = PE * CompE
  - Parallel Efficiency (PE): PE = LB * CommE
    - **Load Balance** Efficiency (LB): LB = avg(CT)/max(CT)
    - **Communication** Efficiency (CommE): CommE = SerE * TE
      - Serialization Efficiency (SerE):
        SerE = max (CT / TT on ideal network)
      - Transfer Efficiency (TE): TE = TT on ideal network / TT
  - **Computation** Scaling (CompS)
    - Computed out of IPC Scaling and Instruction Scaling
    - For strong scaling: ideal scaling -> efficiency of 1.0
-

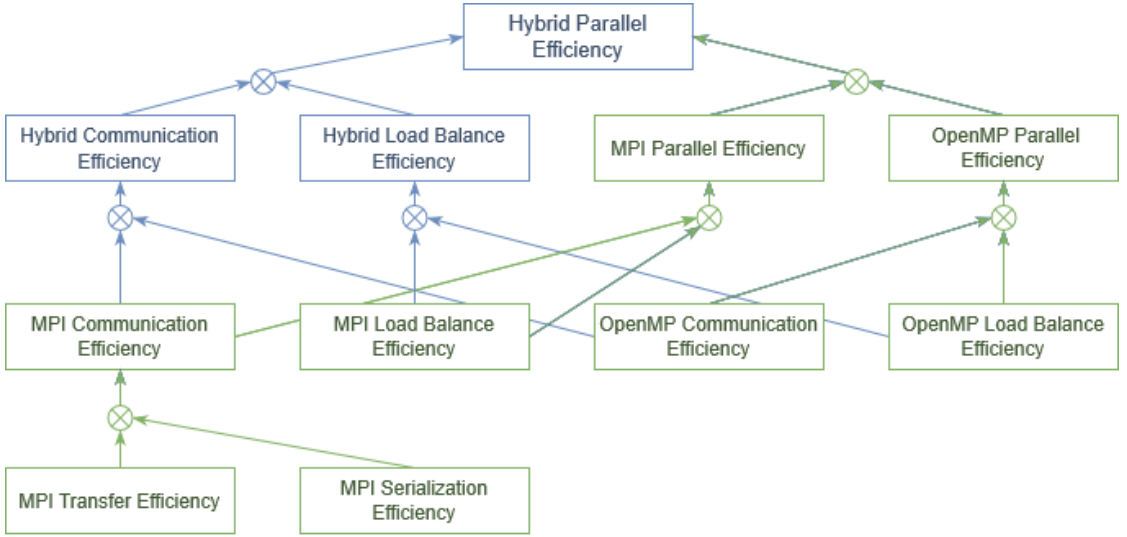


CT = Computational time
TT = Total time

Recap: POP Phase2 (Hybrid) Metrics



Additive Version



Multiplicative Version

POP Metrics + Scalasca

1. Instrument application and setup measurement parameters (e.g. filtering)
   - `scorep <comp+link+cmds>`
   - `scan <exec+cmd> …`

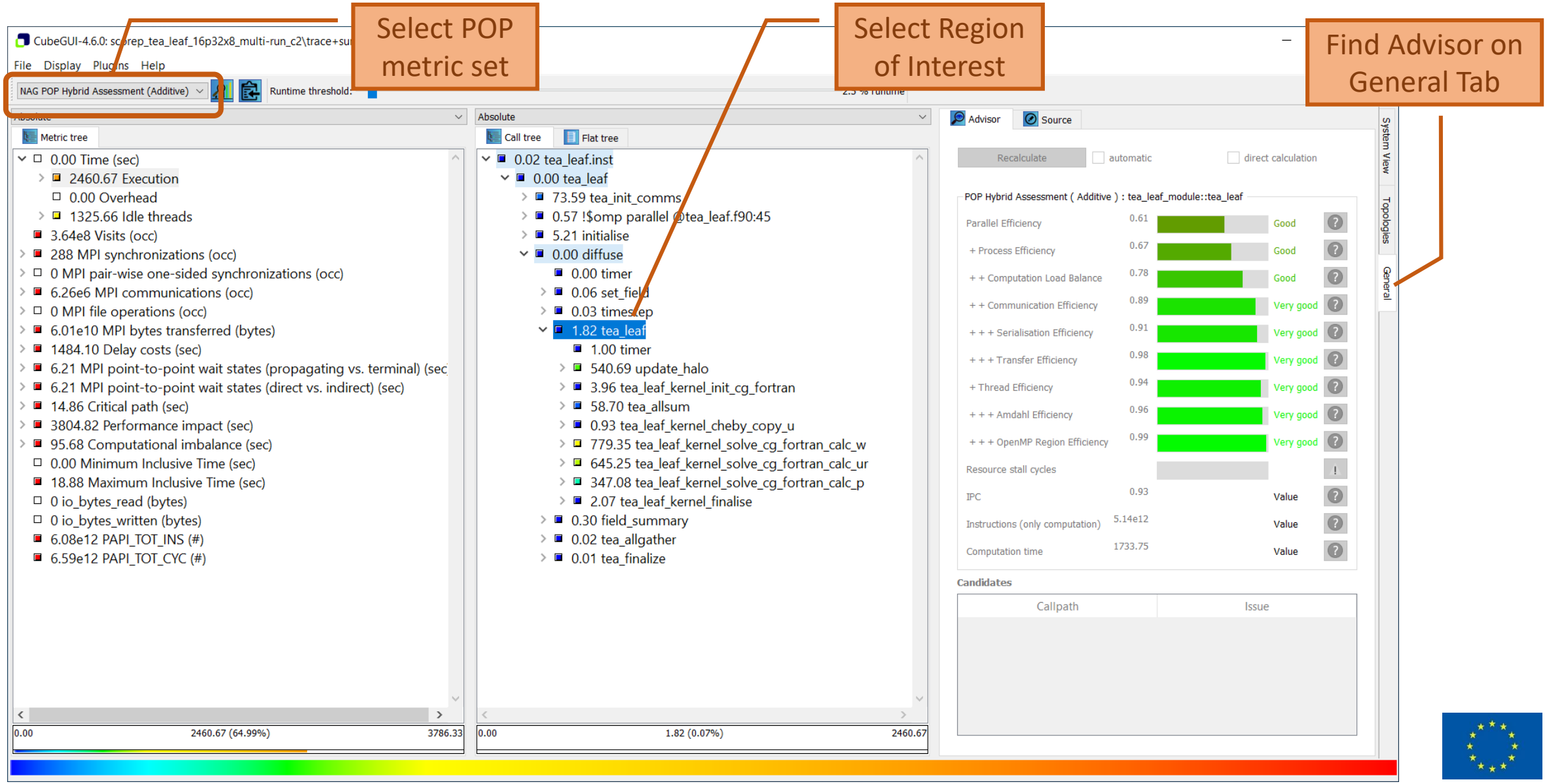
for scaling, repeat

2. For parallel efficiency: perform trace measurement and analysis
3. For computational scaling: perform profile measurement with suitable HW counters
   - `scan -P pop <exec+cmd>`
4. Merge profile and trace measurement
5. Post-process measurement
6. Analyze POP metrics with Cube Advisor
   - `square <measurement+archive>`

Requires
- Scalasca V2.6
- Cube 4.6

!

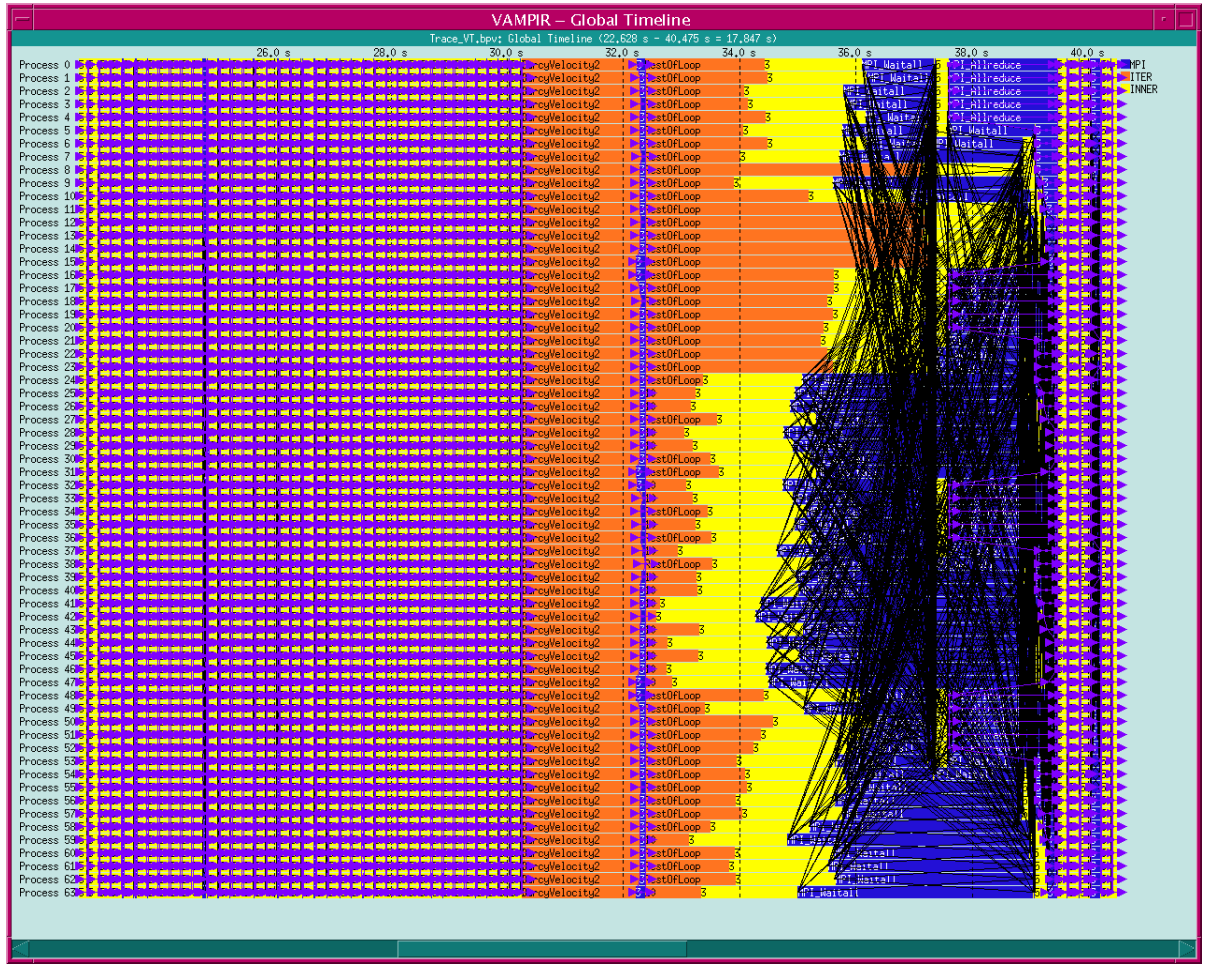
POP Metrics + Scalasca



(Detailed) Performance Analysis with Scalasca

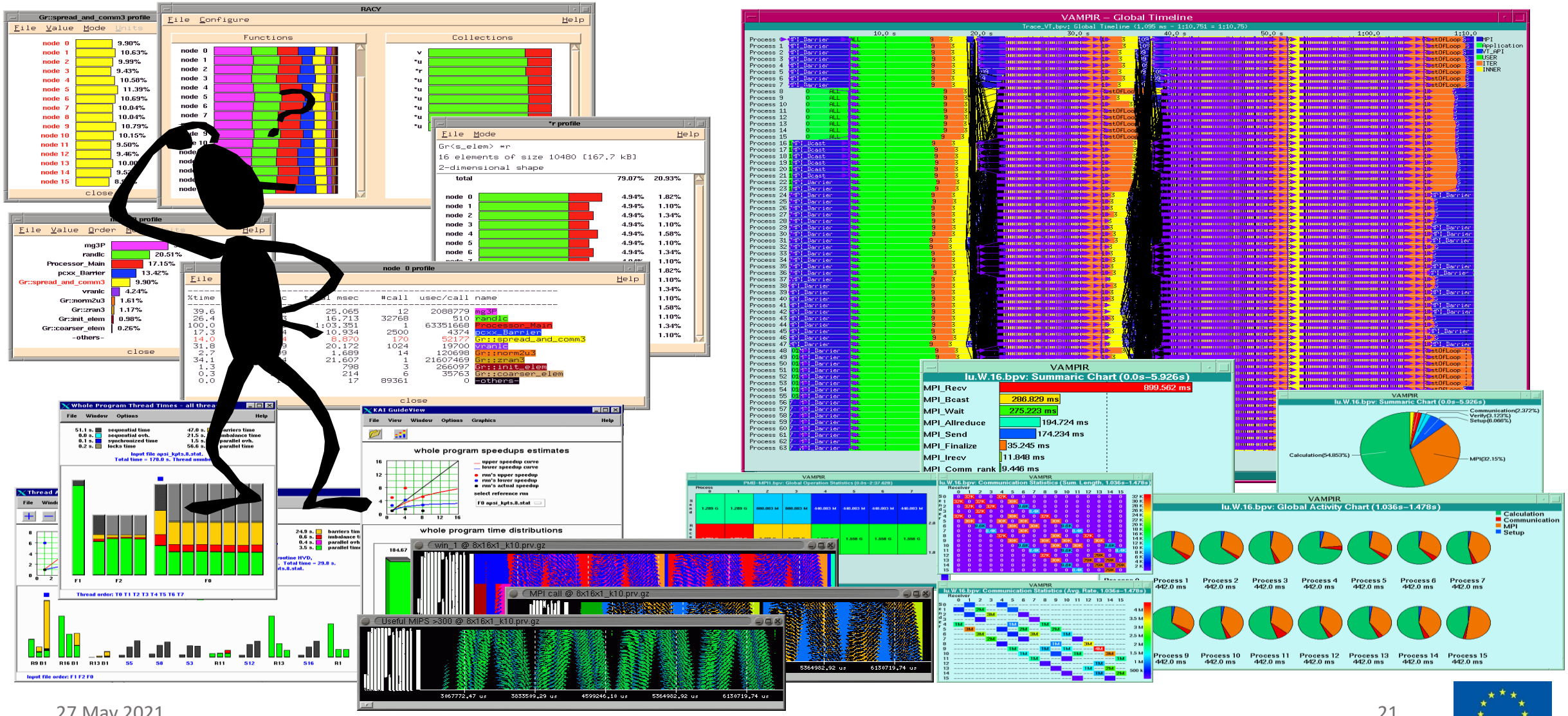
Finding the cause(s)

“A picture is worth 1000 words…”

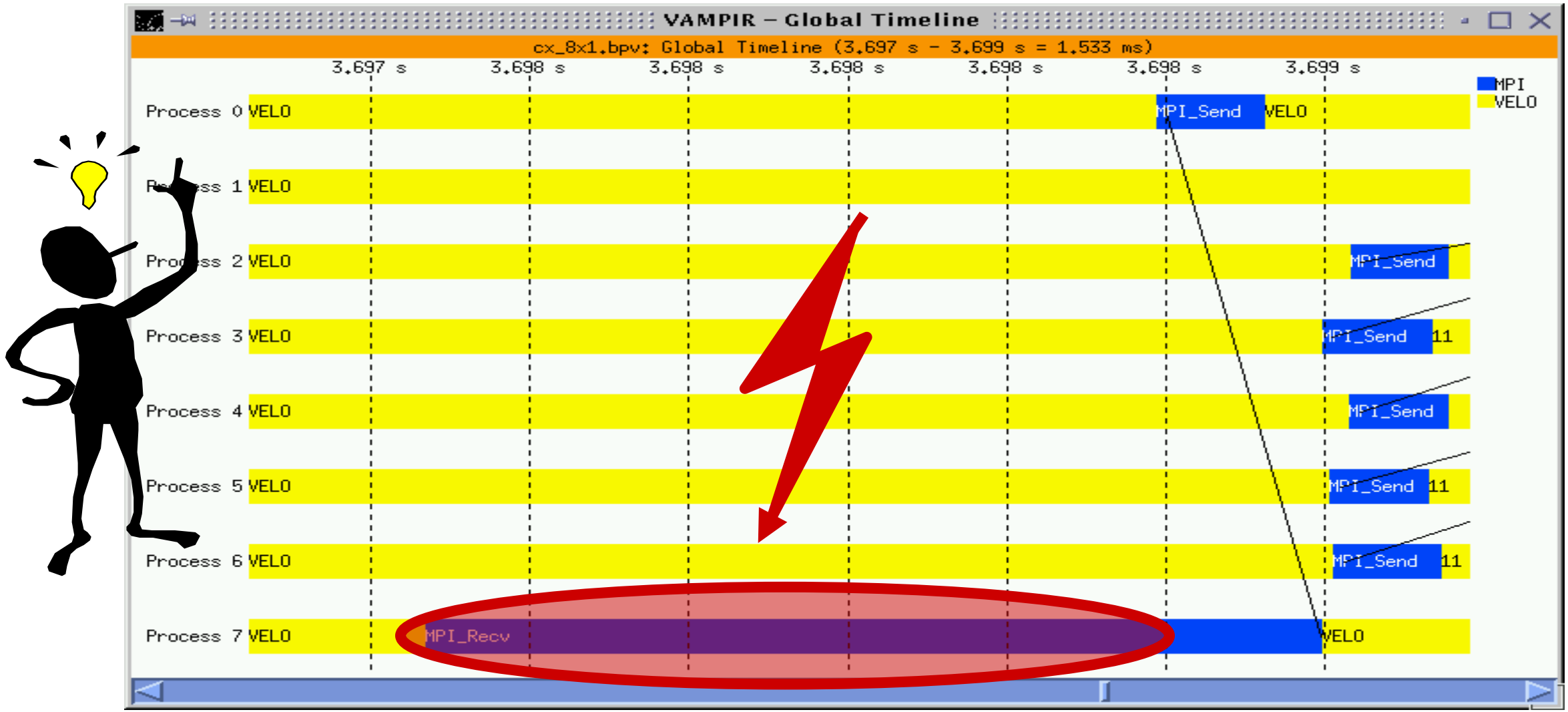


- MPI ring program
- “Real world” example

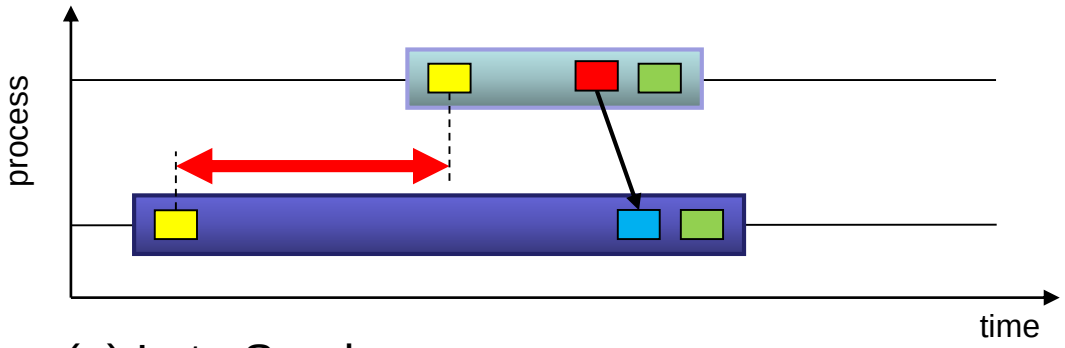
“What about 1000’s of pictures?” (with 100’s of menu options)



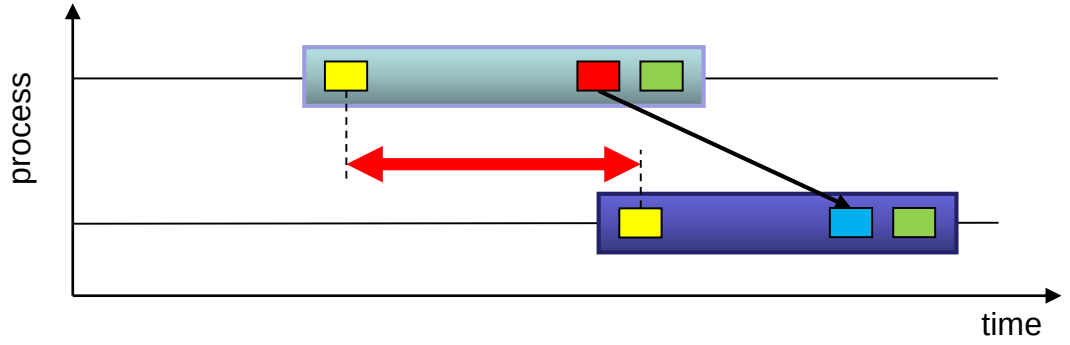
Example Automatic Analysis: Late Sender



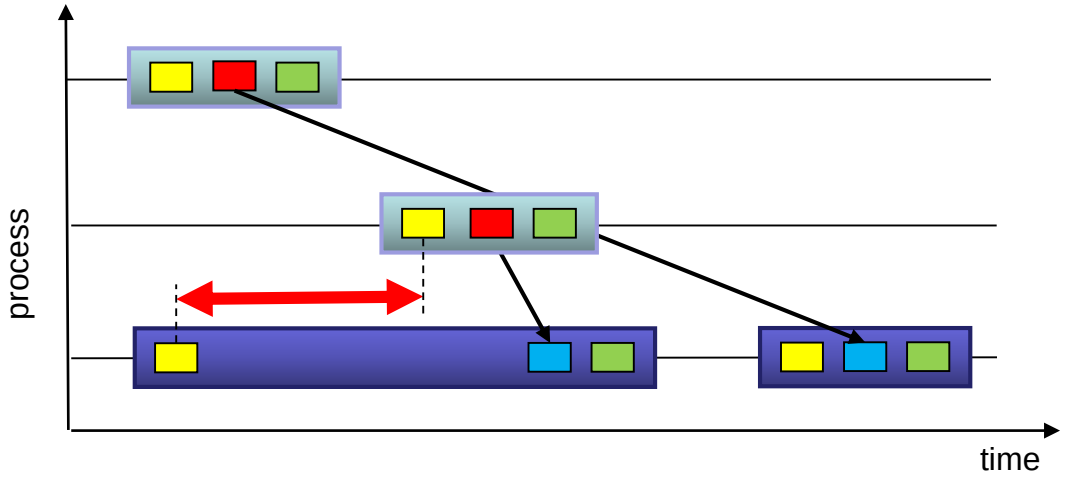
Example MPI Wait States



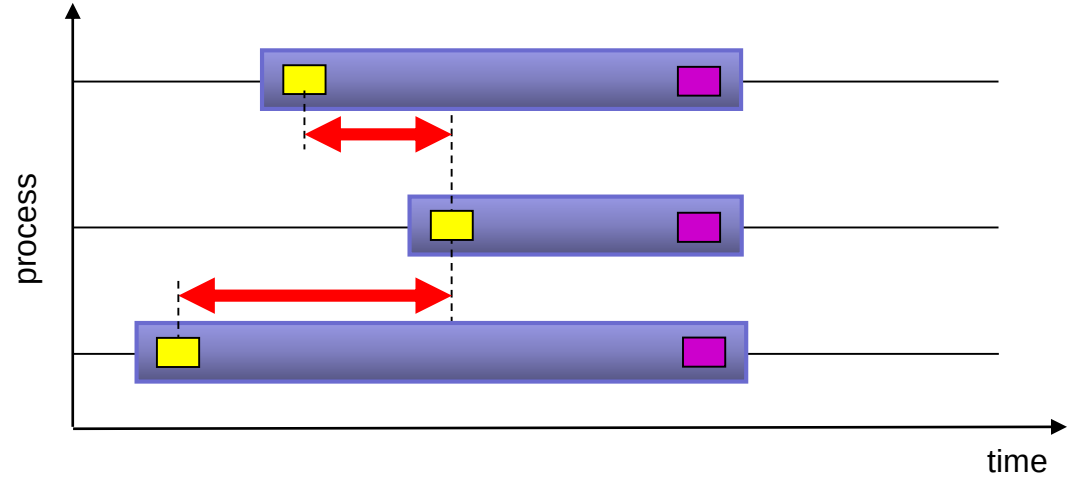
(a) Late Sender



(b) Late Receiver



(c) Late Sender / Wrong Order



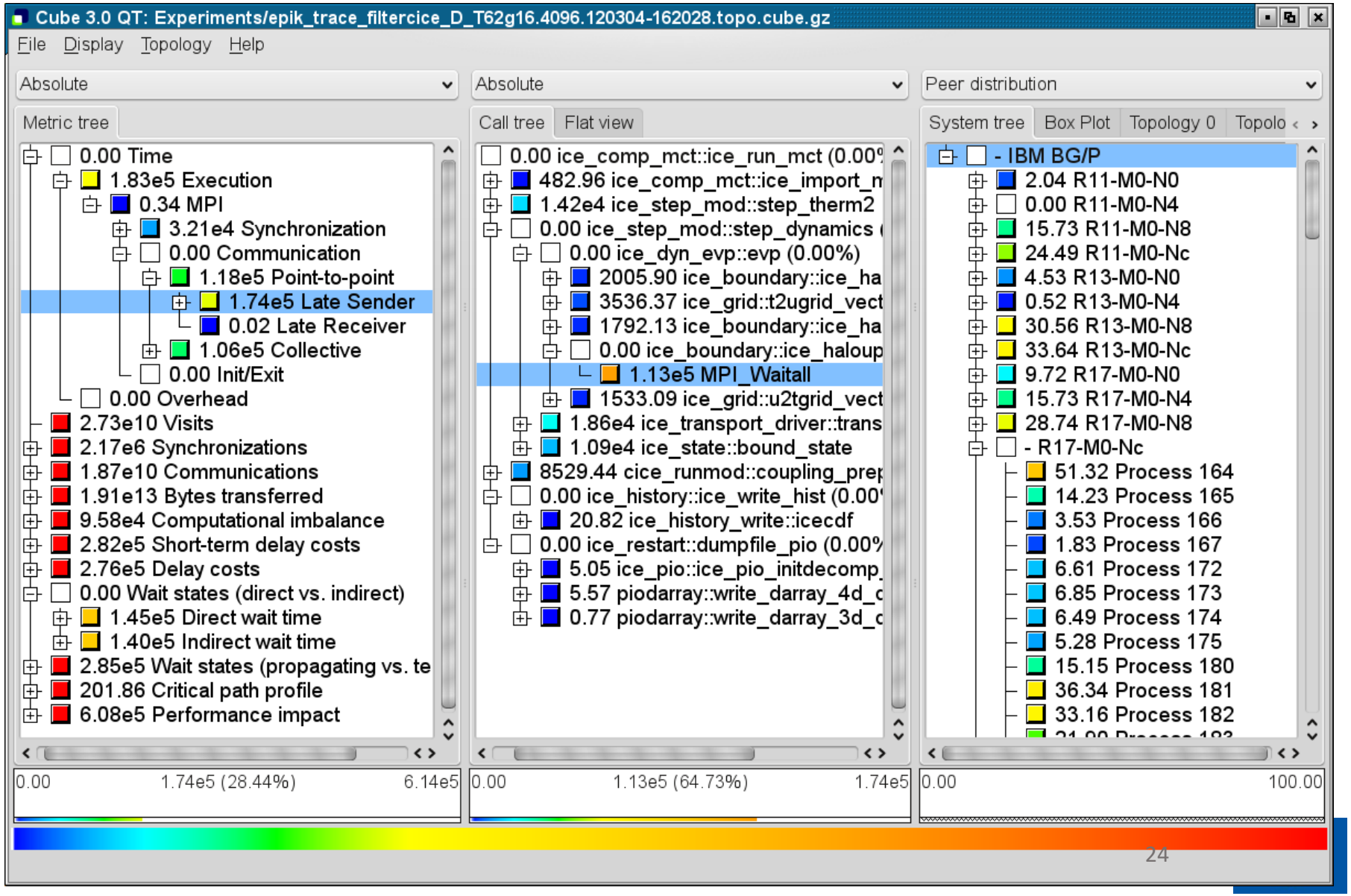
(d) Wait at N x N

ENTER EXIT SEND RECV COLLEXIT

Scalasca Example: CESM Sea Ice Module

Late Sender Analysis

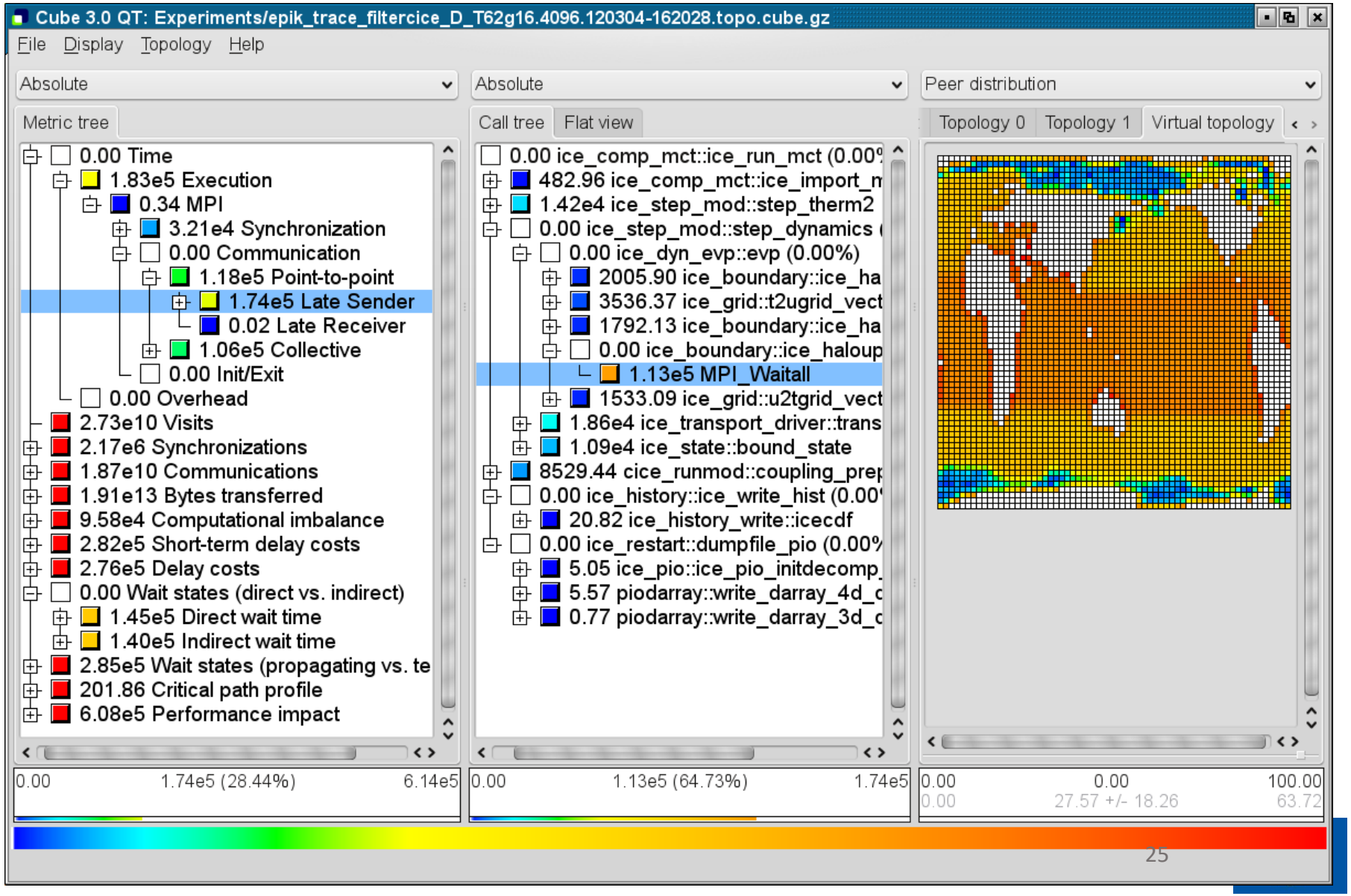
- Finds waiting at MPI_Waitall() inside ice boundary halo update
- Shows distribution of imbalance across system and ranks



Scalasca Example: CESM Sea Ice Module

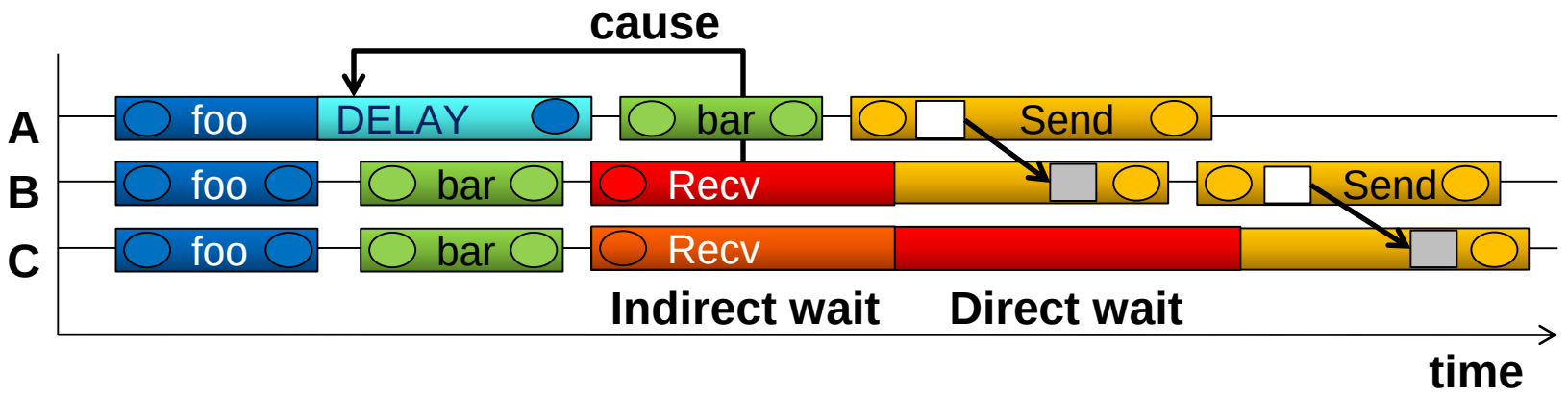
Late Sender Analysis + Application Topology

- Shows distribution of imbalance over topology
- MPI topologies are automatically captured
- Also: topology Process x Threads



Scalasca Root Cause Analysis

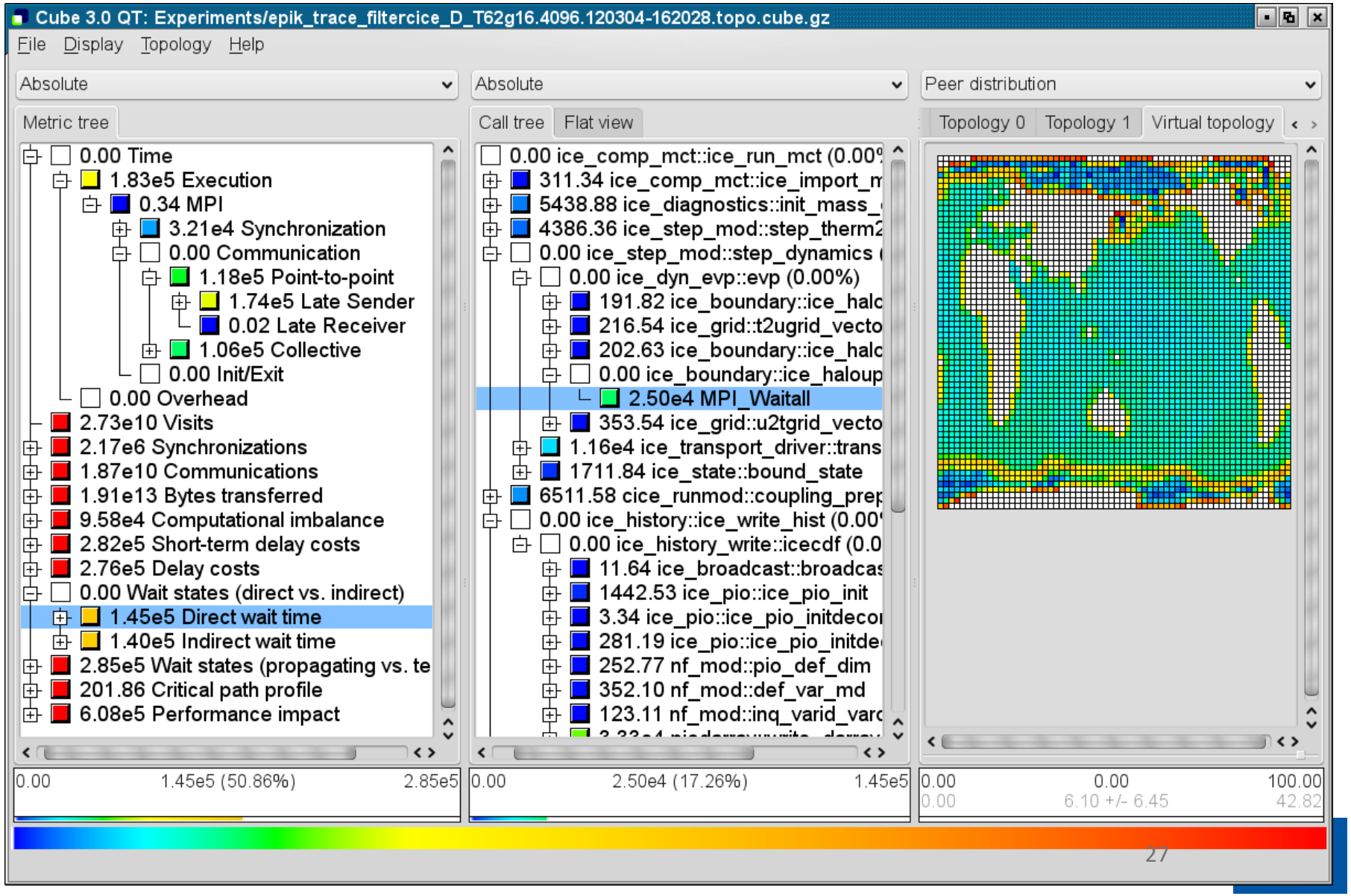
- **Root-cause analysis**
  - Wait states typically caused by load or communication imbalances earlier in the program
  - Waiting time can also propagate (e.g., indirect waiting time)
  - Enhanced performance analysis to find the root cause of wait states
- **Approach**
  - Distinguish between direct 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

Direct Wait Time Analysis

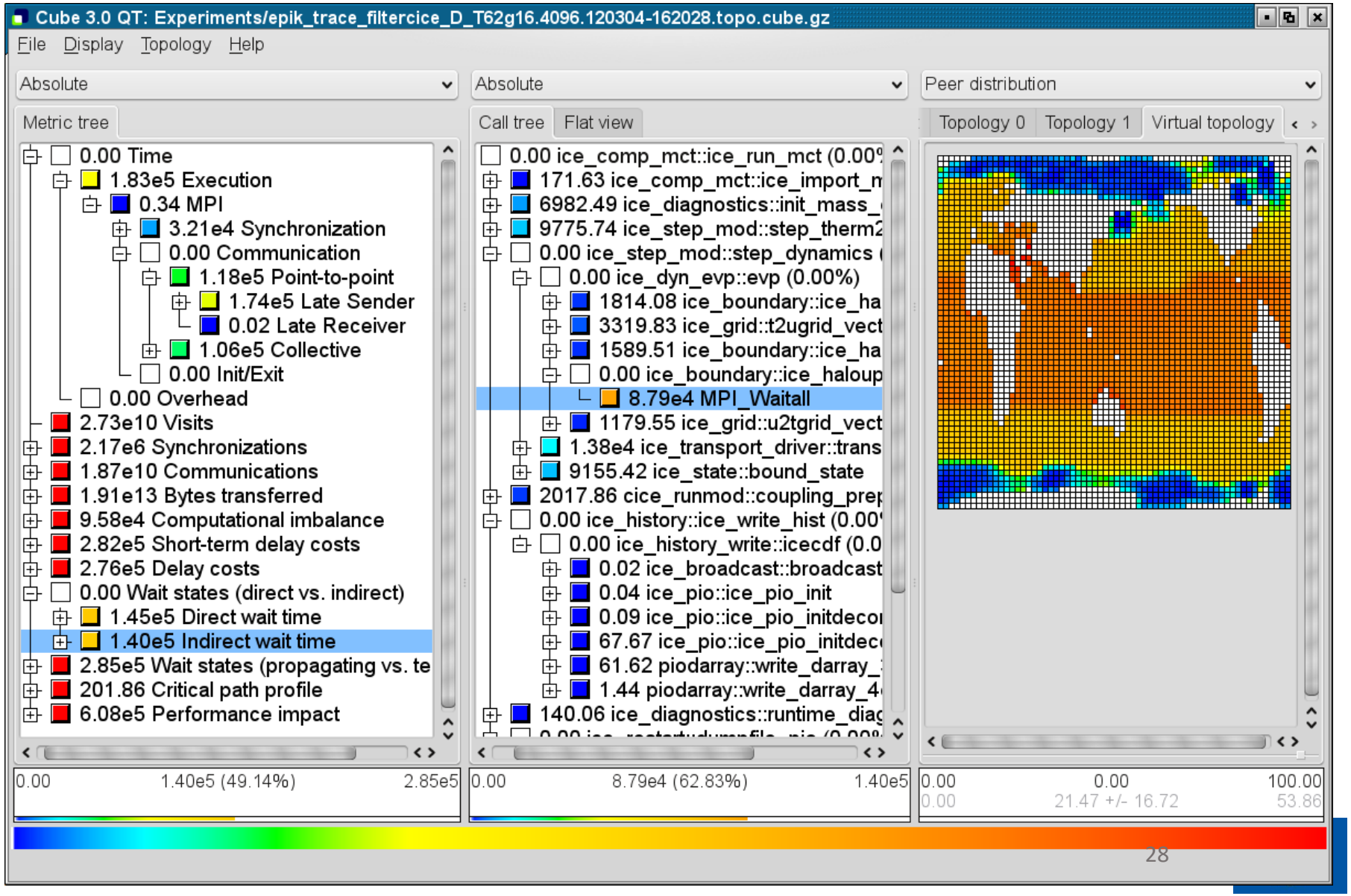
- Direct wait caused by ranks processing areas near the north and south ice borders



Scalasca Example: CESM Sea Ice Module

Indirect Wait Time Analysis

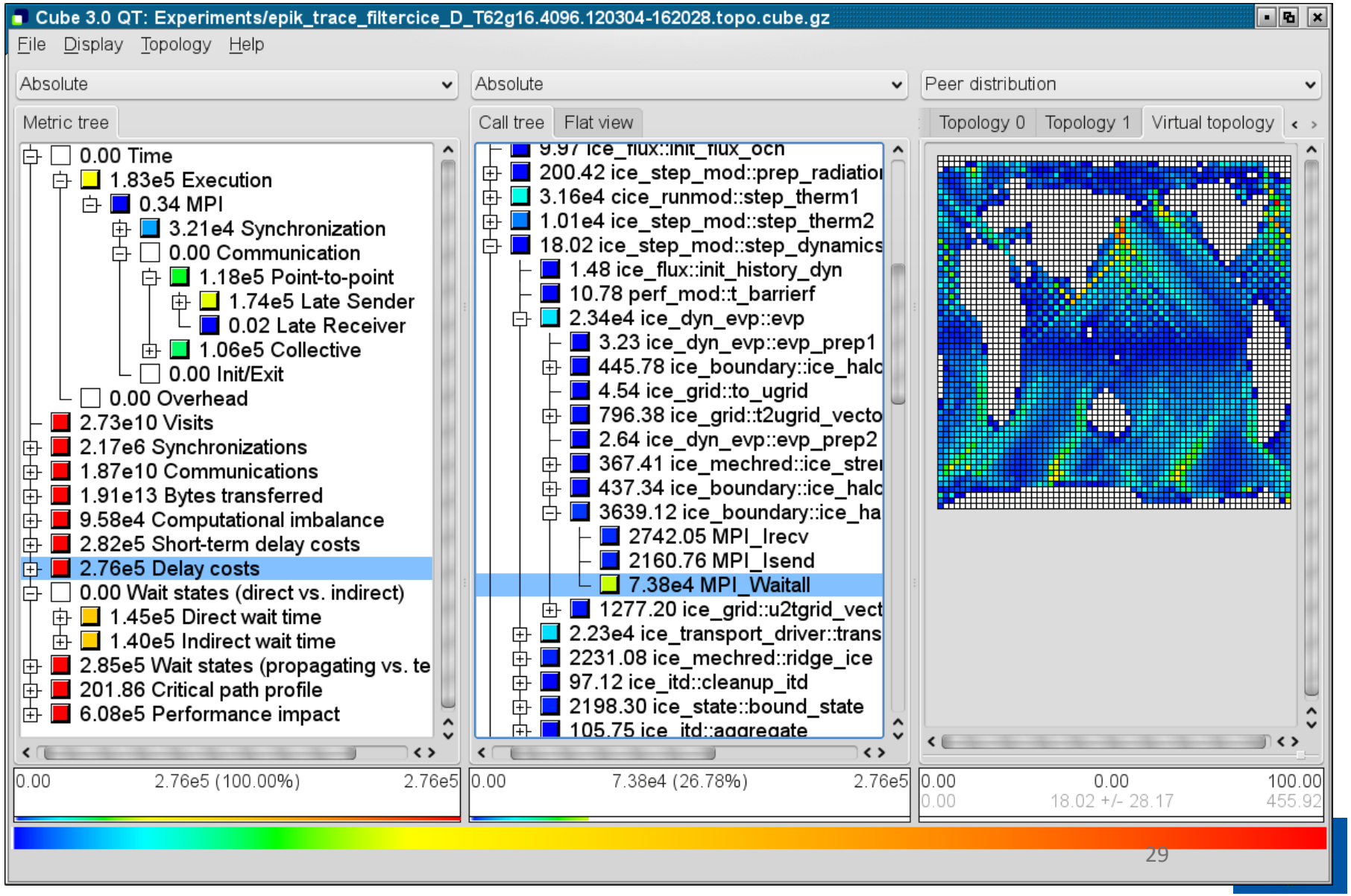
- Indirect waits occurs for ranks processing warmer areas



Scalasca Example: CESM Sea Ice Module

Delay Costs Analysis

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

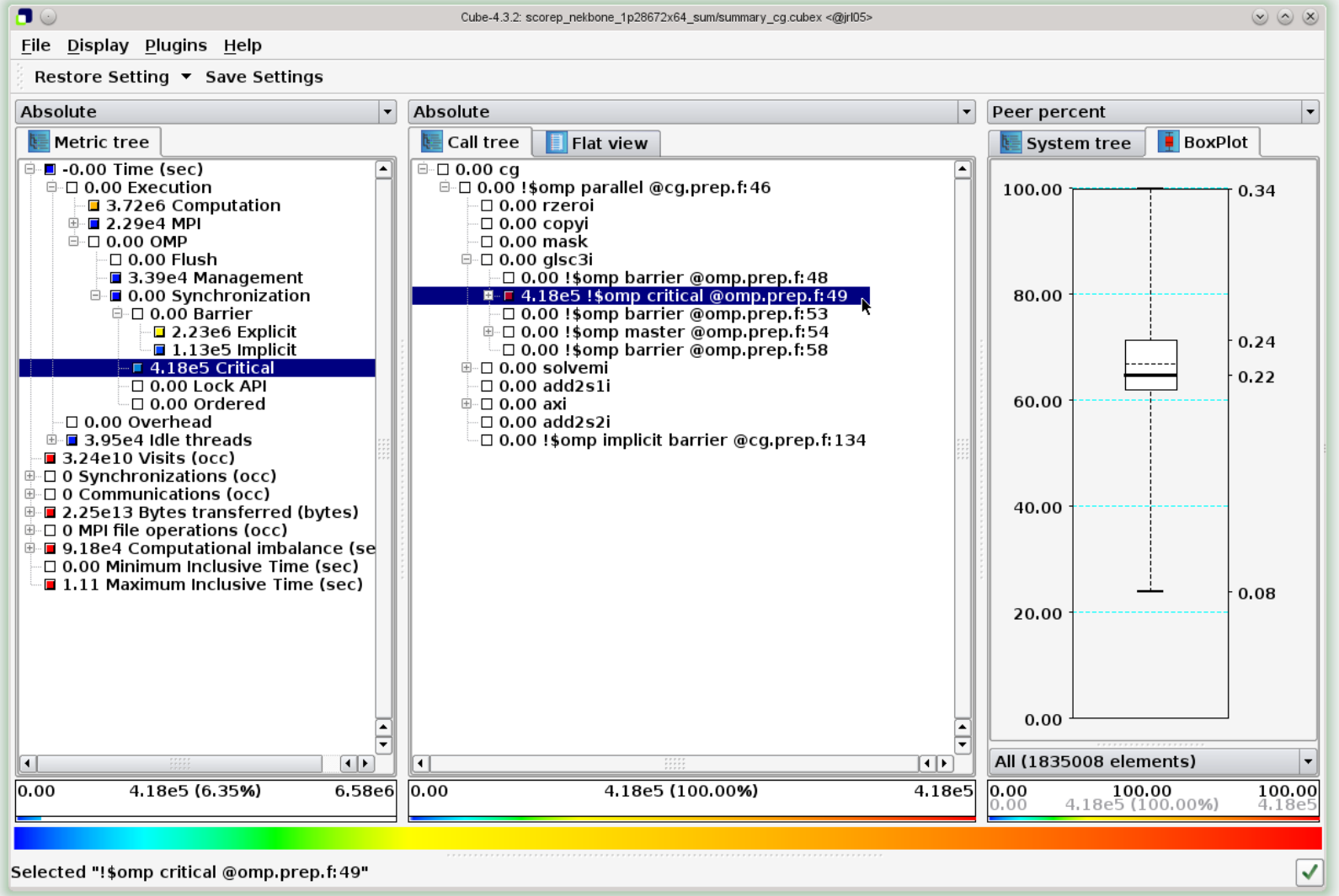


Scalability

Towards Exascale

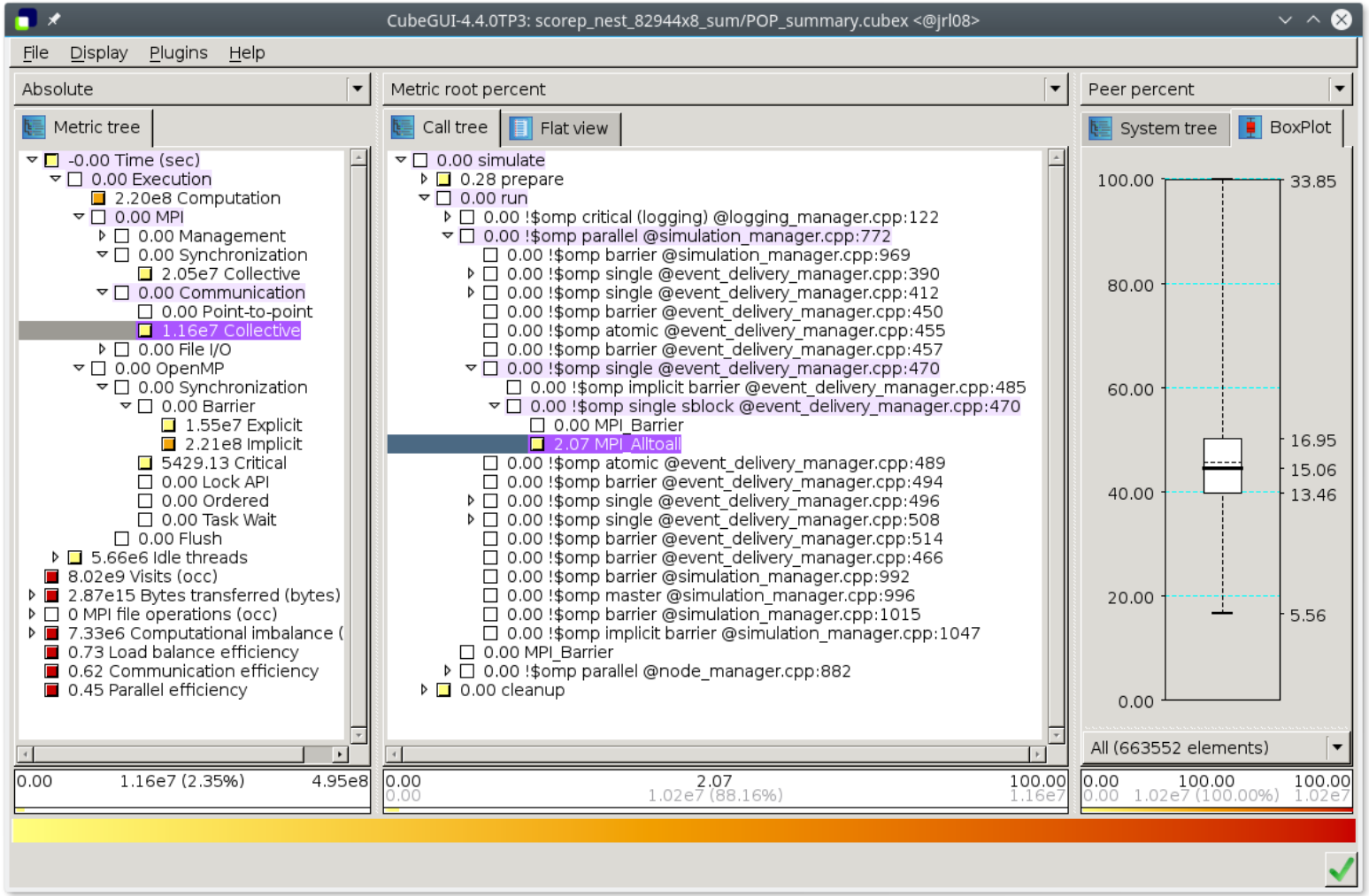
Scalasca: 1,835,008 Threads Test Case

- Nekbone (CORAL benchmark)
- Measurement of **full system BGQ JuQueen computer run**
  - 28,672 x 64 = 1,835,008 threads
  - A few TB trace data!
  - A few Million events!
- **Possible because:**
  - **Scalable parallel measurement and trace analysis**



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



Wrapup

What I hope you remember

Executive Summary

- **Scalasca** facilitates
  - **Simple POP assessments** based on POP metrics!
  - **Advanced performance analysis**!

- Even on **extreme scales**!

Further Info

https://pop-coe.eu/further-information/online-training

- POP Online Training modules
  - Installing POP Tools: Score-P, Scalasca, Cube
  - Using POP Tools: Score-P and Scalasca
  - Using POP Tools: Cube

https://pop-coe.eu/blog/19th-pop-webinar-identifying-performance-bottlenecks-in-hybrid-mpi-openmp-software

- POP Hybrid MetricsWebinar

Upcoming Training

https://www.vi-hps.org/training/tws/tw40.html

- 40th VI-HPS Tuning Workshop
  - 14th to 18th June 2021
  - Online (ZOOM)

https://app.swapcard.com/widget/event/isc-high-performance-2021-digital/planning/UGxhbm5pbmdfNDUzNTIw

- ISC 2021 Tutorial “Determining Parallel Application Execution Efficiency and Scaling using the POP Methodology”
  - 24th June 2021, 14:00 – 18:00 CET
  - Judit Gimenz (BSC), Brian Wylie (JSC)
  - Online (SwapCard)

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.