PARALLEL PERFORMANCE ANALYSIS AT SCALE: FROM SINGLE NODE TO ONE MILLION HPC CORES

SEP 11, 2019 I BERND MOHR

Mitglied der Helmholtz-Gemeinschaft

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

SETTING THE CONTEXT

- Parallel Performance Analysis can be
  - Analytical ⇨ Using analytical models
  - **Empirical ⇨ Using experiments (“monitoring”)**

  to assess performance

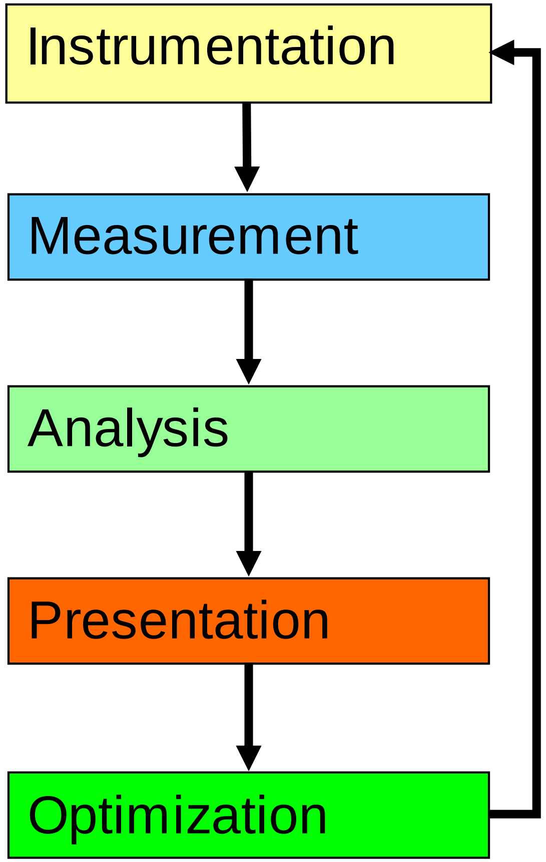
- Parallel could mean
  - Loosely-coupled ⇨ “Grid” / distributed computing
  - **Tightly-coupled ⇨ HPC**

- Performance Monitoring can target
  - Computer systems
  - **Applications**

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 file**
  - Sequence of events over time

NO SINGLE SOLUTION IS SUFFICIENT!

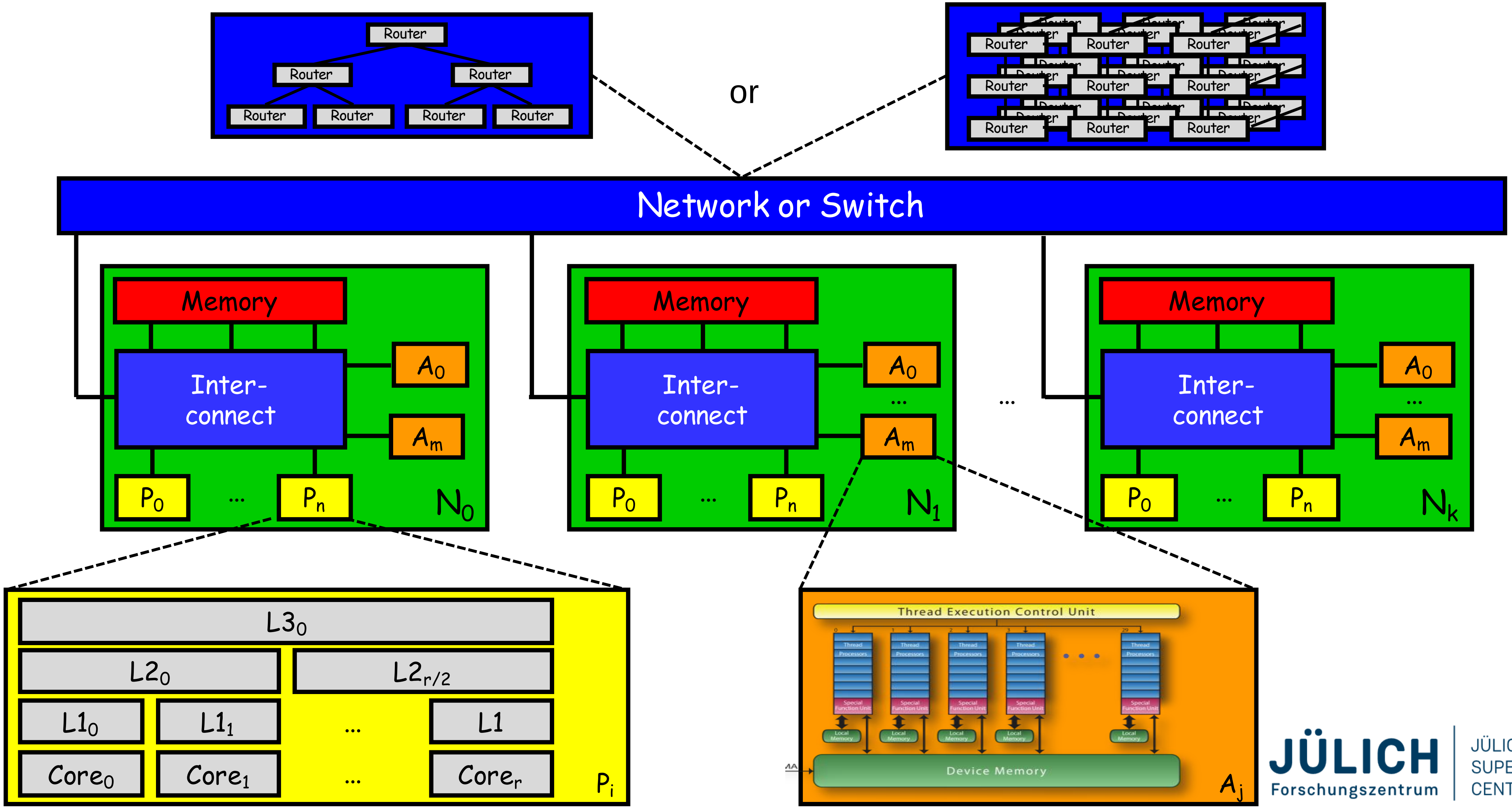
⇨ **Combination of methods, techniques and tools needed**

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

How and Why

MULTI- AND MANY-CORE PERFORMANCE ANALYSIS

PARALLEL ARCHITECTURES: STATE OF THE ART



JÜLICH
Forschungszentrum

JÜLICH
SUPERCOMPUTING
CENTRE

PERFORMANCE CHALLENGES FOR HPC SYSTEMS

- HPC systems consist of
  - **Complex** configurations
  - With a **huge** number of components
    - Very likely **heterogeneous**
  - With never enough memory
  - **Dynamically changing** configuration due to fault recovery + power saving

⇨ Deep software **hierarchies of large, complex software** components are needed to make use of such systems

⇨ **Sophisticated integrated performance measurement, analysis, and optimization capabilities are required to efficiently operate an HPC system**

DESIRED TOOL FEATURES

This requires tools to be

- Portable
- Insightful
- Scalable
- Integrated
- [Versatile]
- [Maintained]
- ~~Easy to use~~

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

TYPICAL PERFORMANCE TUNING

NOT MANY HPC TOOLS MATCH THESE REQUIREMENTS

- **TAU**
  - University of Oregon, US
  - http://tau.uoregon.edu
- **HPCToolkit**
  - Rice University, US
  - http://hpctoolkit.org
- **Extrae / Paraver**
  - BSC, Spain
  - http://www.bsc.es/paraver
- **Vampir / VampirServer**
  - TU Dresden, Germany
  - http://www.vampir.eu
- **Scalasca**
  - JSC/TU Darmstadt, Germany
  - http://www.scalasca.org
- **[Score-P]**
  - JSC, TUD, TUDA, TUM, RWTH, Germany
  - http://www.score-p.org

Run everywhere

PORTABILITY

SCALASCA: SUPPORTED ARCHITECTURES

- **Instrumentation and measurement only (visual analysis on front-end or workstation)**
  - Cray XT, XE, XK, XC
  - IBM BlueGene/L, BlueGene/P, BlueGene/Q
  - K Machine, Fujitsu FX10 and FX100
  - Tianhe 1A and 2
  - Intel MIC (KNC, KNL)
- **Full support (instrumentation, measurement, and automatic analysis)**
  - Linux IA32, IA64, x86_64, PPC, ARM, and ARM64 based clusters
  - IBM AIX Power3/4/5/6/7/8/9 based clusters

TYPICAL HPC PLATFORMS

- **OS**
  - Now: Mostly Linux (and HPC microkernels)

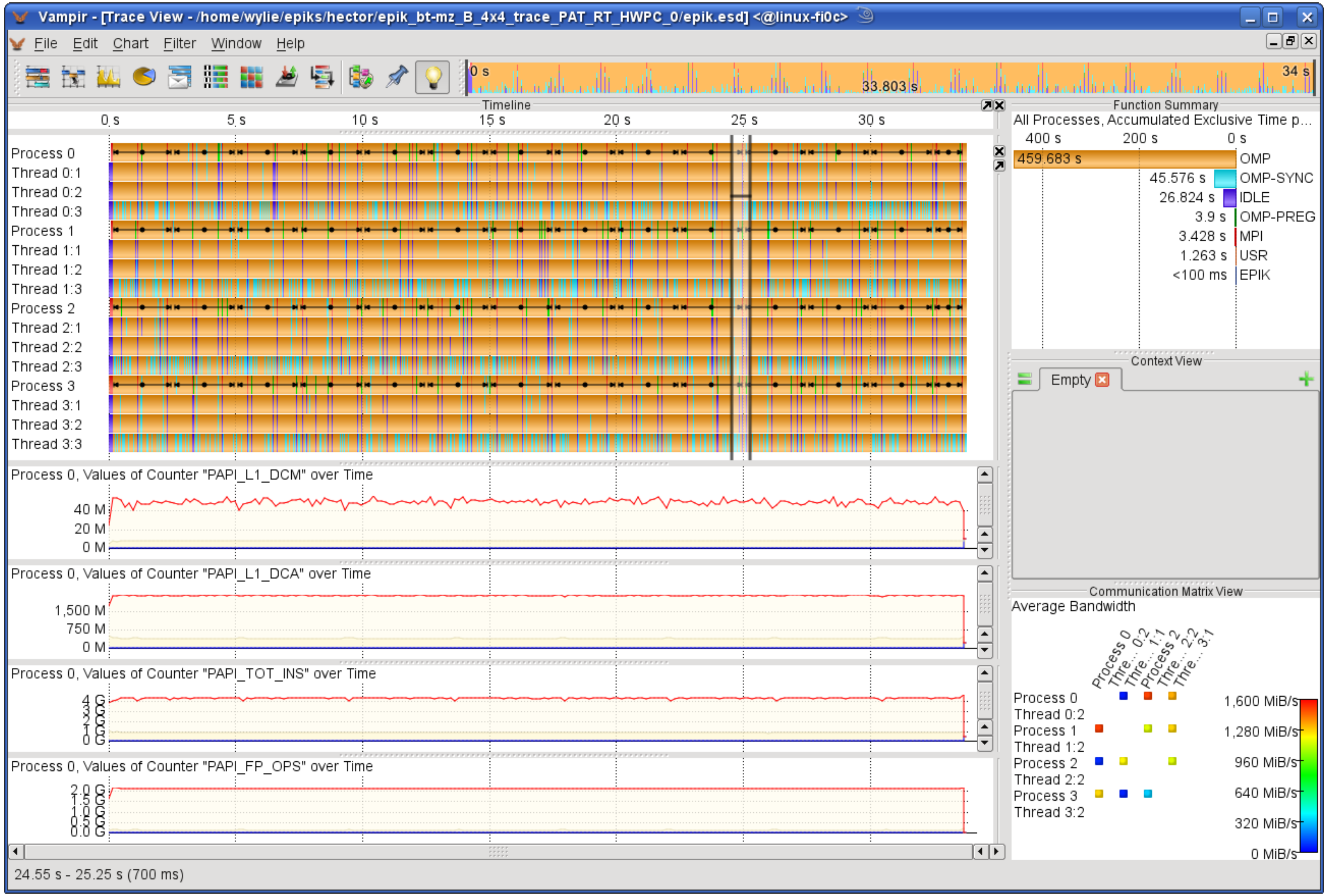
- **C/C++ and Fortran Compilers (⇨ OpenMP, OpenACC)**
  - GNU, Intel, PGI, Clang, IBM XL, Cray, Fuijtsu, ARM, …
  - Different versions supporting different versions of OpenMP and OpenACC

- **MPI**
  - MPICH, OpenMPI, Intel, Cray, IBM PE, SGI, Fujitsu, …
  - Different versions supporting different versions of MPI

More than numbers and diagrams

INSIGHTFULNESS

INTERACTIVE EVENT TRACE ANALYSIS: VAMPIR

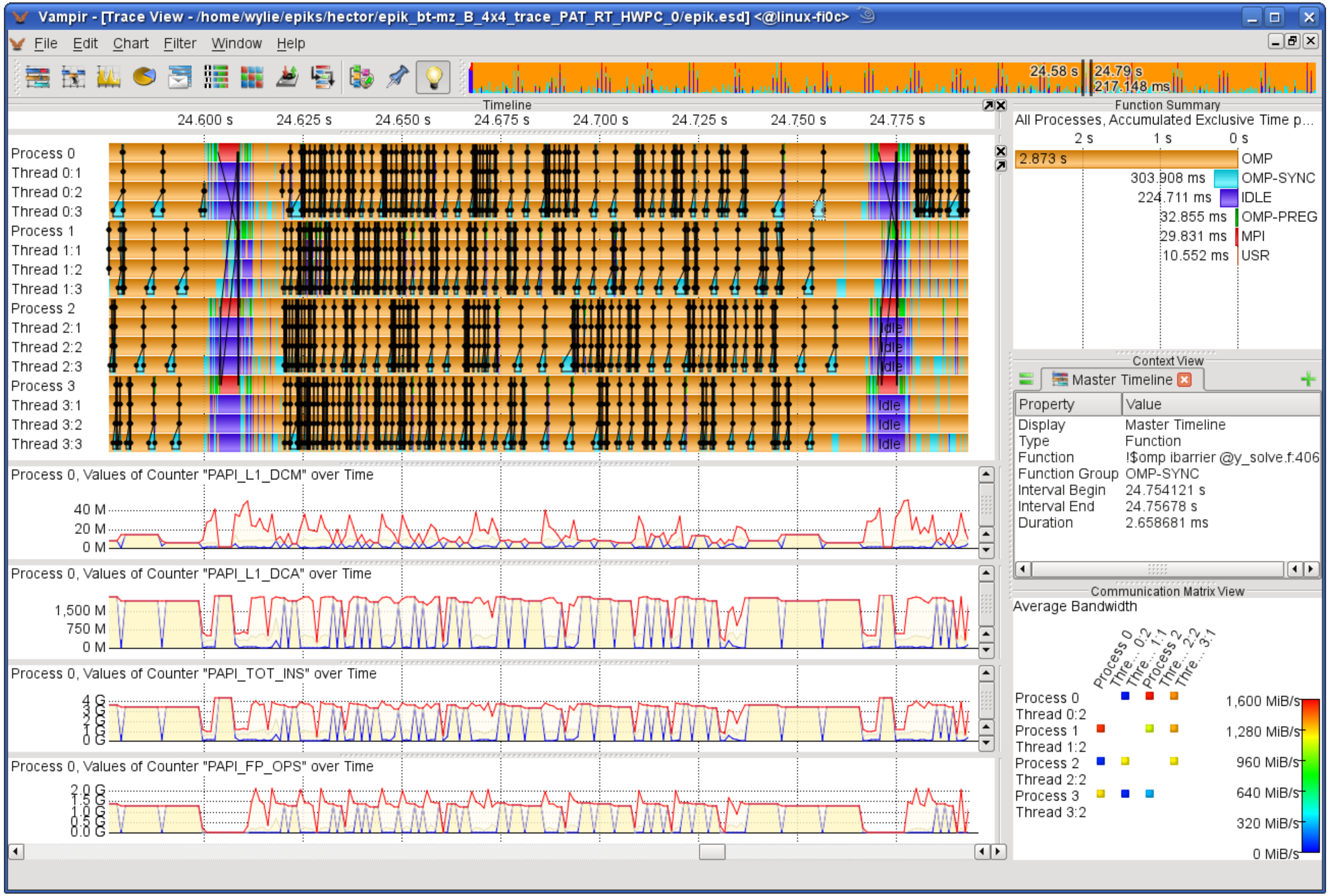


Visual presentation of dynamic runtime behaviour

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

http://www.vampir.eu/

VAMPIR GUI (ZOOM)



Interactive browsing, zooming, selecting

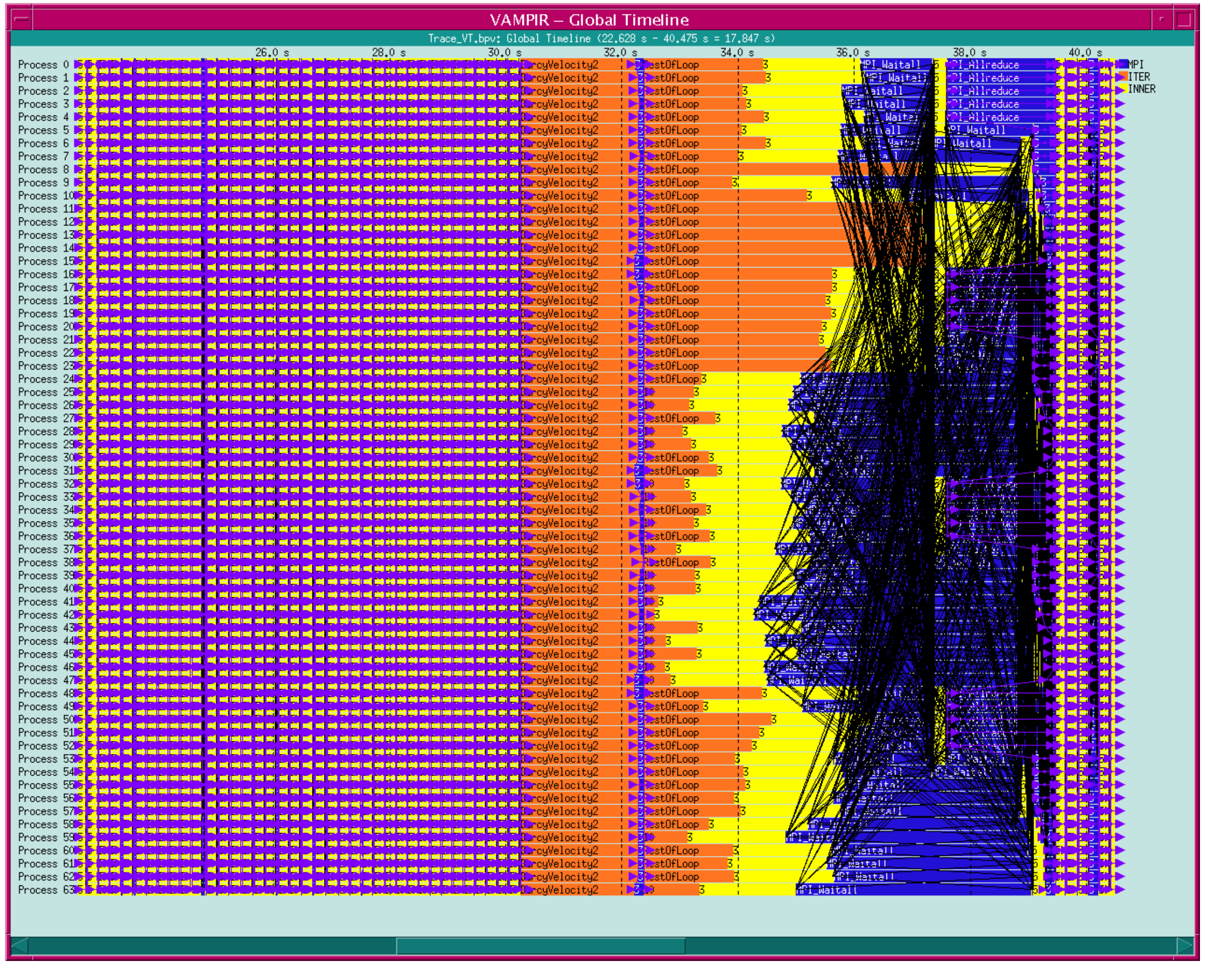
- Linked displays & statistics adapt to selected time interval

Trace formats

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

“A PICTURE IS WORTH 1000 WORDS…”

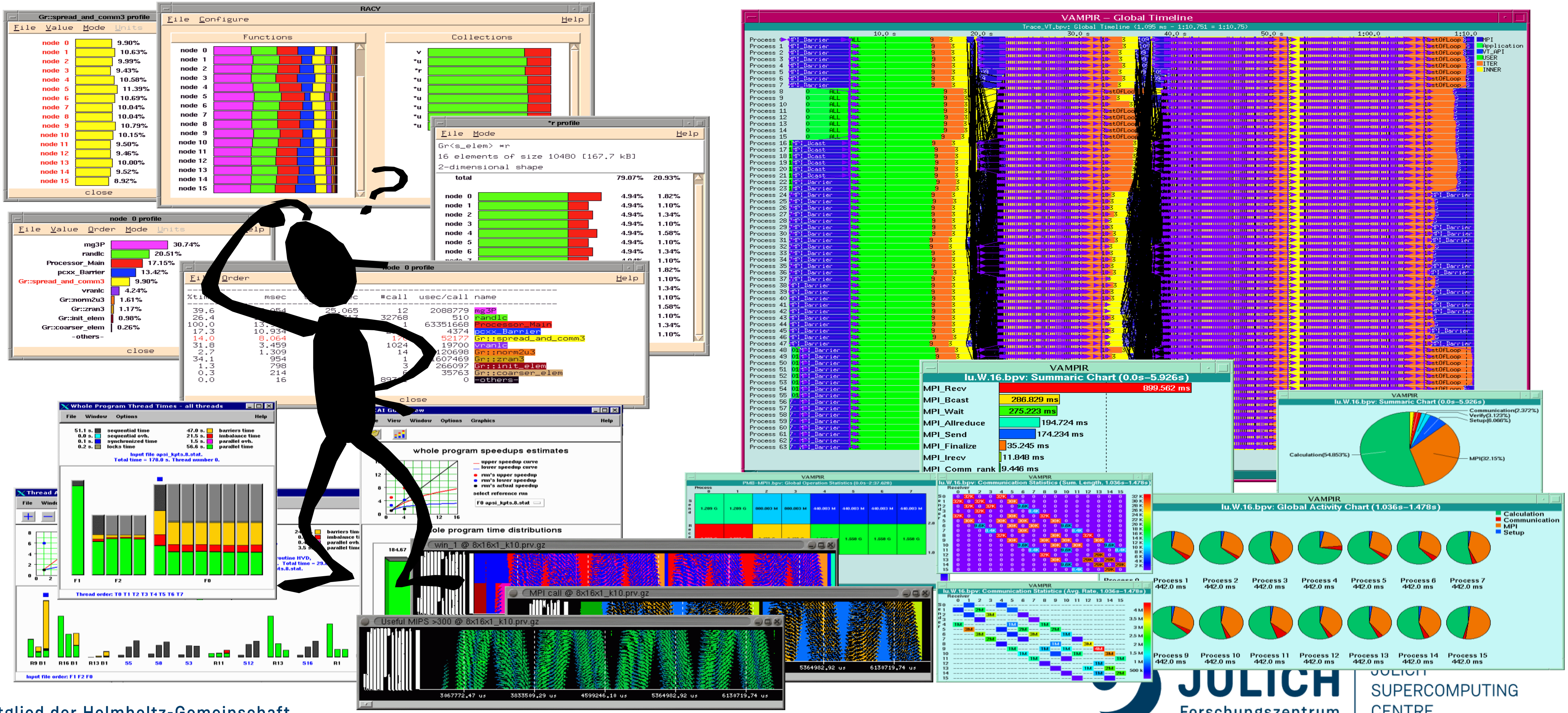
- MPI ring program



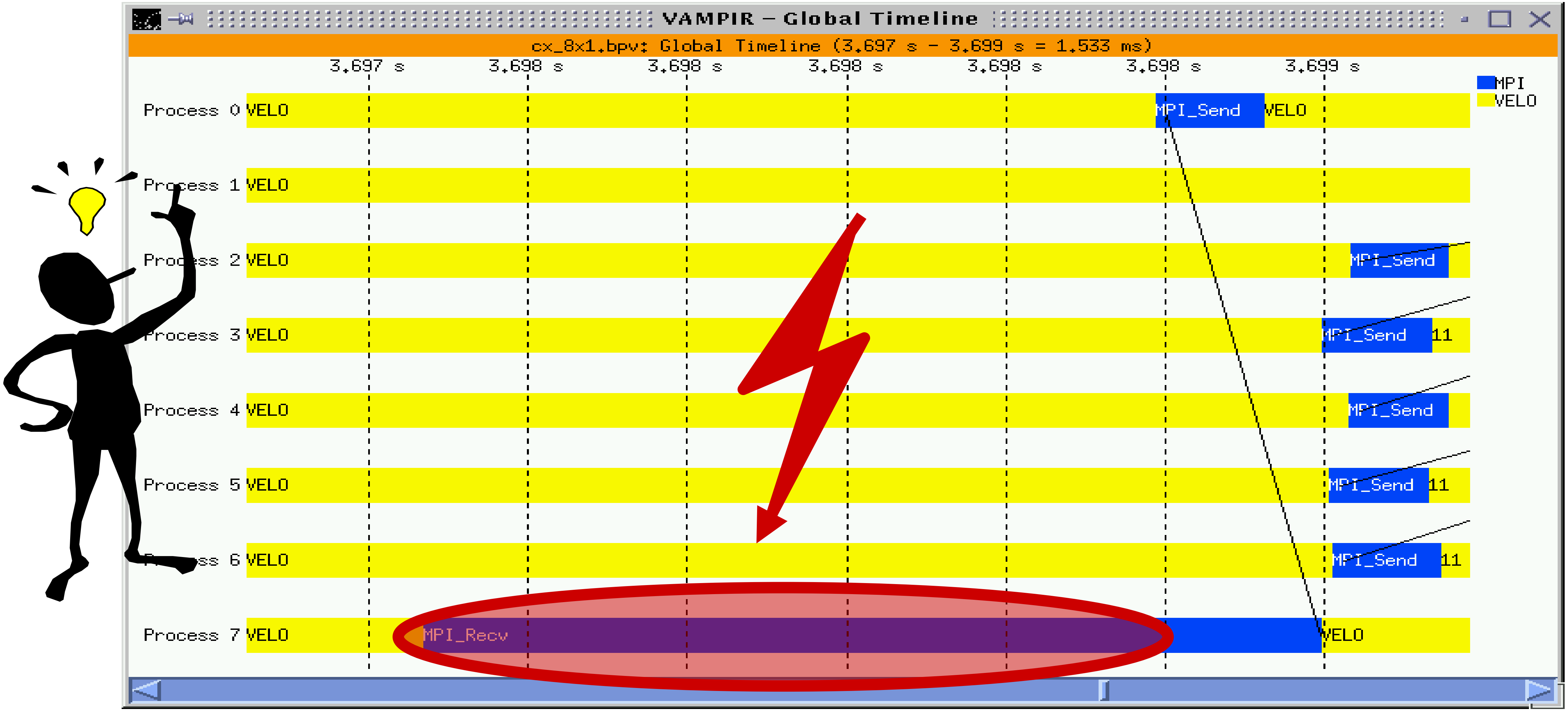
- “Real world” example

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

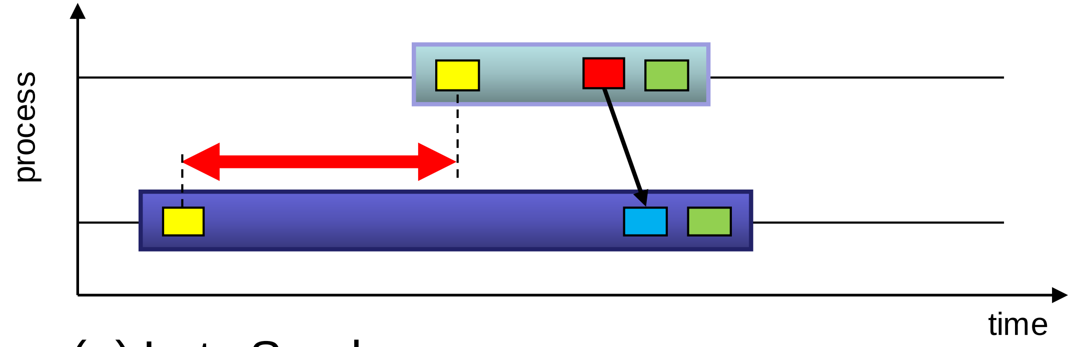
"WHAT ABOUT 1000'S OF PICTURES?" (WITH 100'S OF MENU OPTIONS)



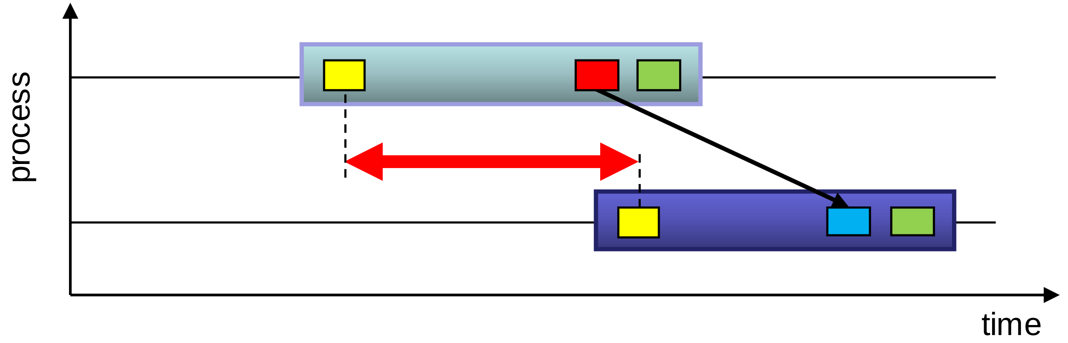
EXAMPLE AUTOMATIC ANALYSIS: LATE SENDER



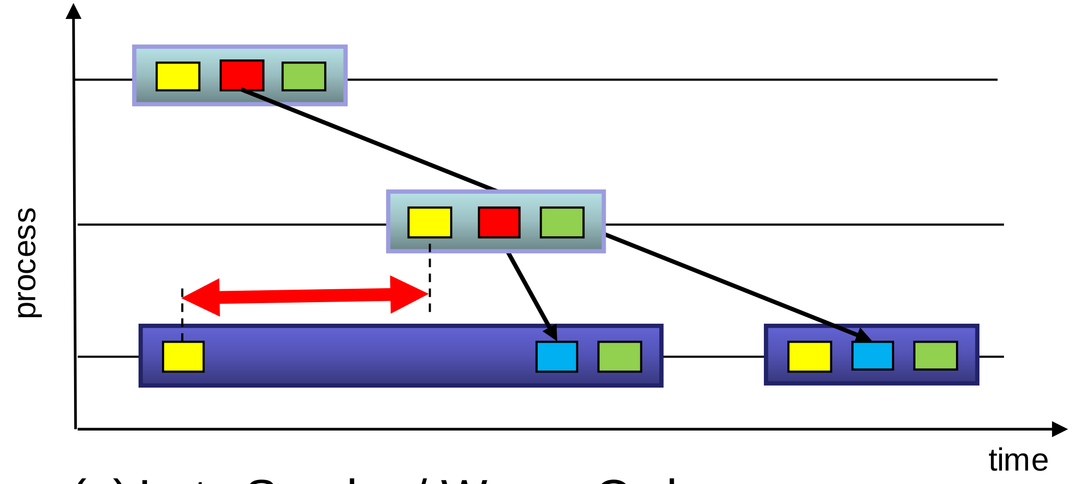
EXAMPLE MPI WAIT STATES



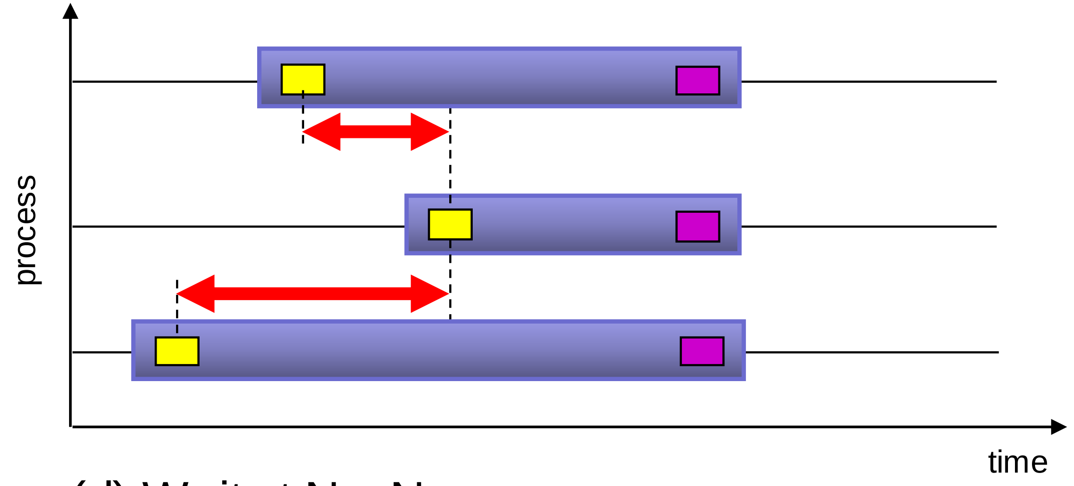
(a) Late Sender



(b) Late Receiver



(c) Late Sender / Wrong Order



(d) Wait at N x N

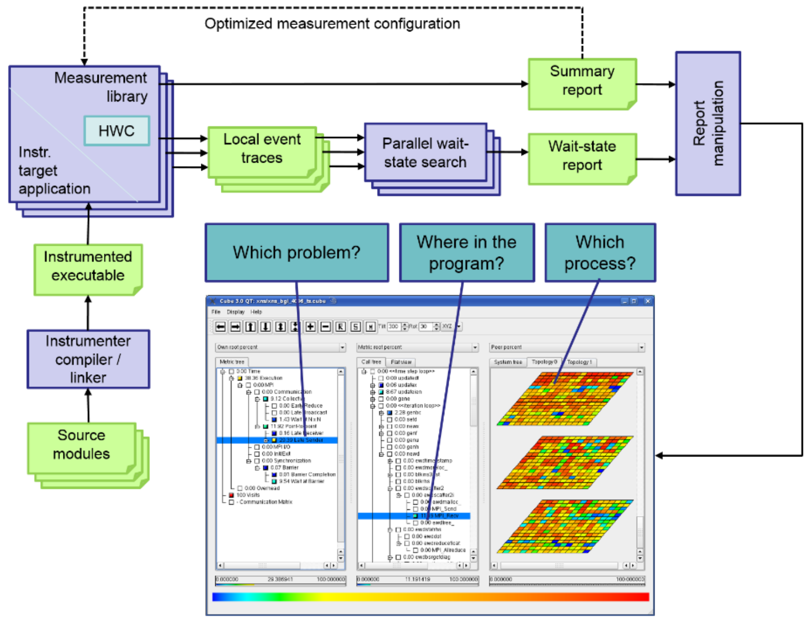
ENTER EXIT SEND RECV COLLEXIT

SCALASCA

http://www.scalasca.org/

- Scalable Analysis of Large Scale Applications

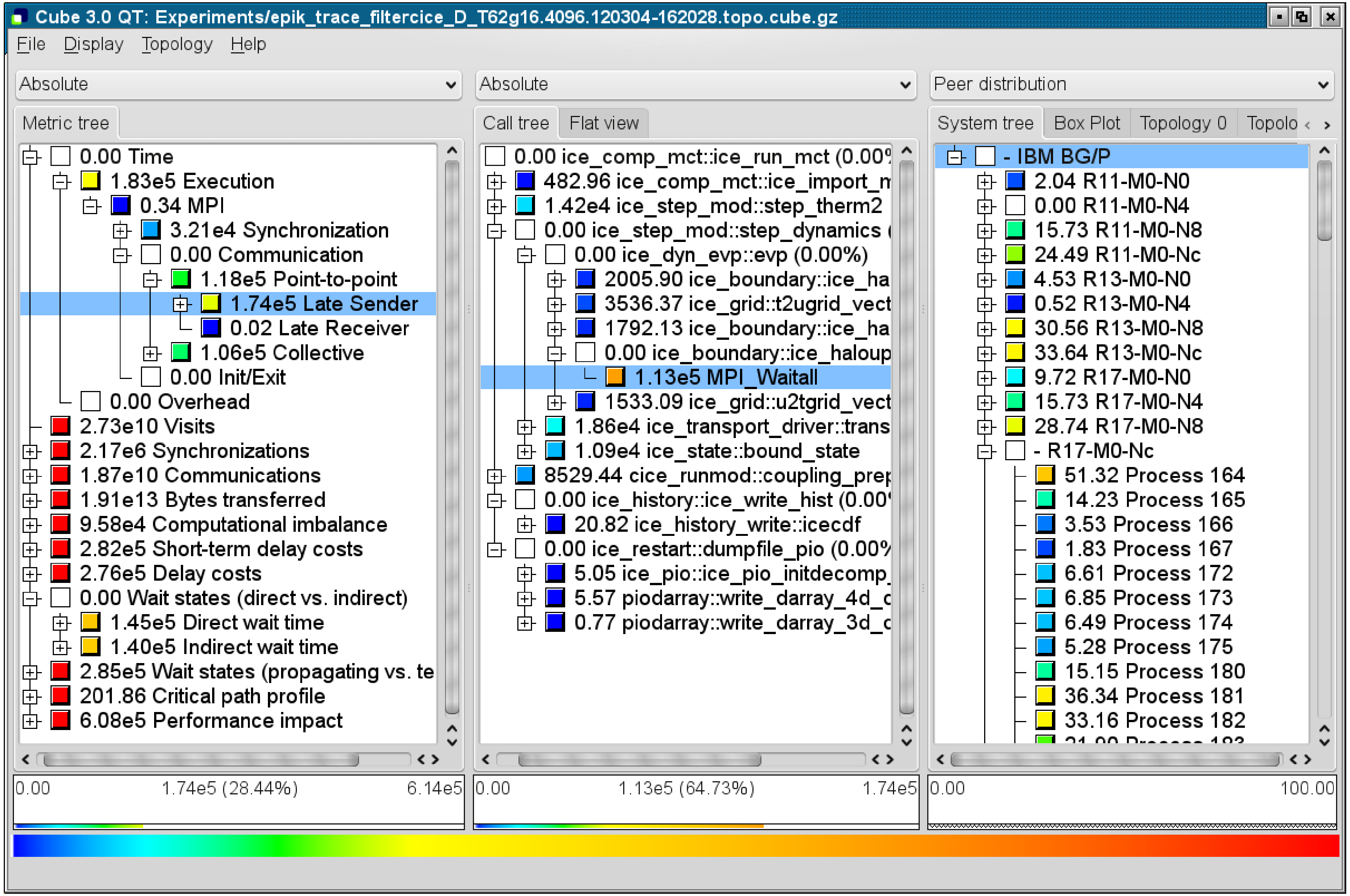
- Approach
  - **Instrument** C, C++, and Fortran parallel applications (with Score-P)
  - Option 1: **scalable call-path profiling**
  - Option 2: **scalable event trace analysis**
    - **Collect** event traces
    - **Process trace in parallel**
      - Wait-state analysis
      - Delay and root-cause analysis
      - Critical path analysis
    - **Categorize and rank** results



SCALASCA EXAMPLE: CESM SEA ICE MODULE

Late Sender Analysis

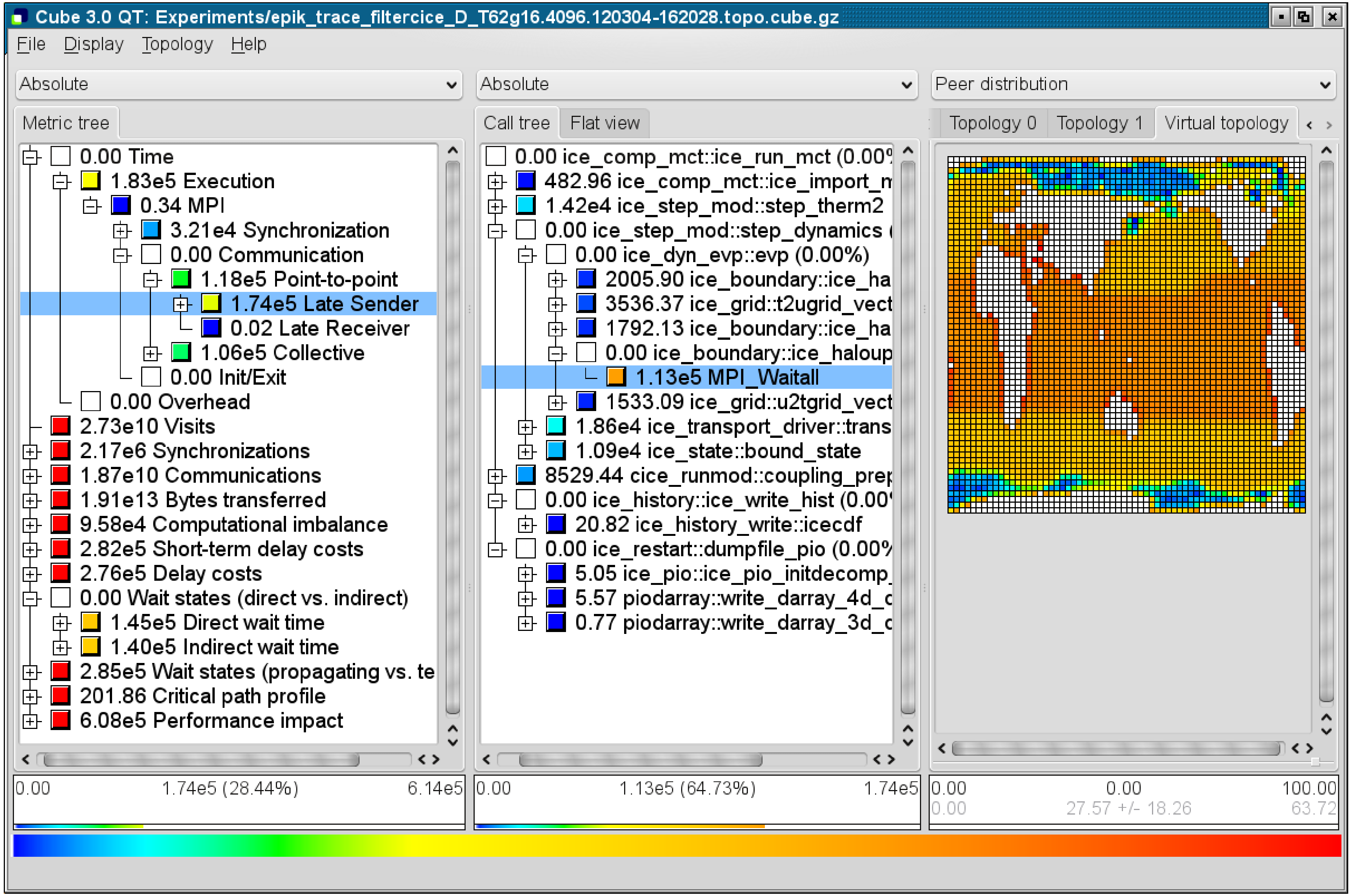
- Finds waiting at MPI_Waitall() inside ice boundary halo update
- Shows distribution of imbalance across system and ranks



SCALASCA EXAMPLE: CESM SEA ICE MODULE

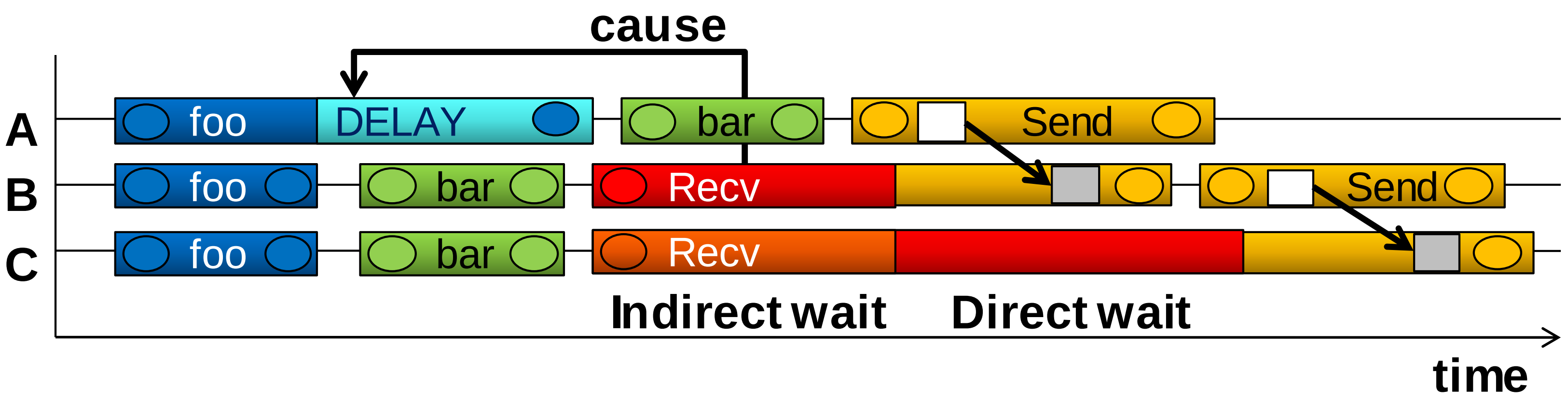
Late Sender Analysis + Application Topology

- Shows distribution of imbalance over topology
- MPI topologies are automatically captured



SCALASCA ROOT CAUSE ANALYSIS

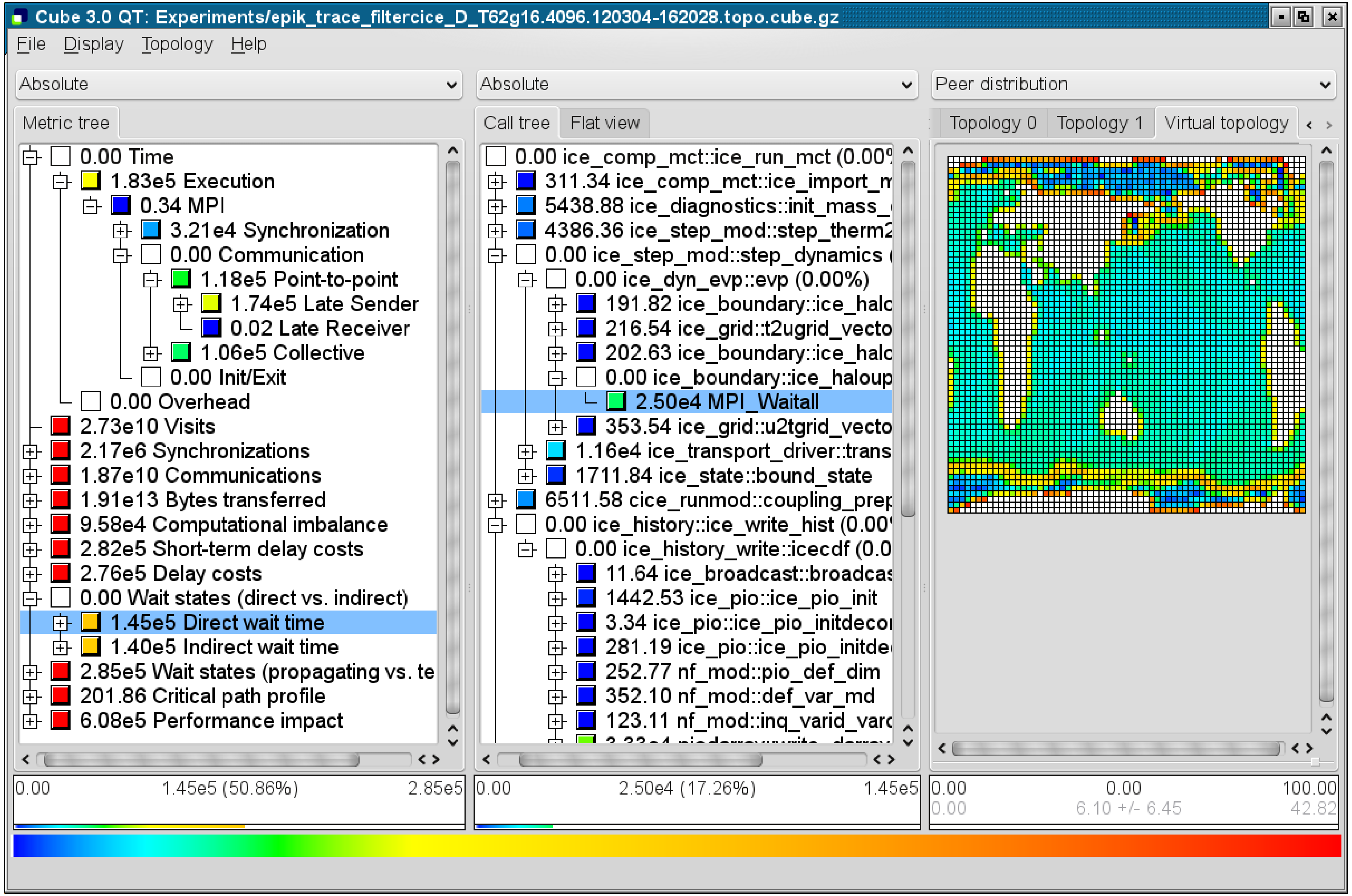
- **Root-cause analysis**
  - Wait states typically caused by load or communication imbalances earlier in the program
  - Waiting time can also propagate (e.g., indirect waiting time)
  - Enhanced performance analysis to find the root cause of wait states
- **Approach**
  - Distinguish between direct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

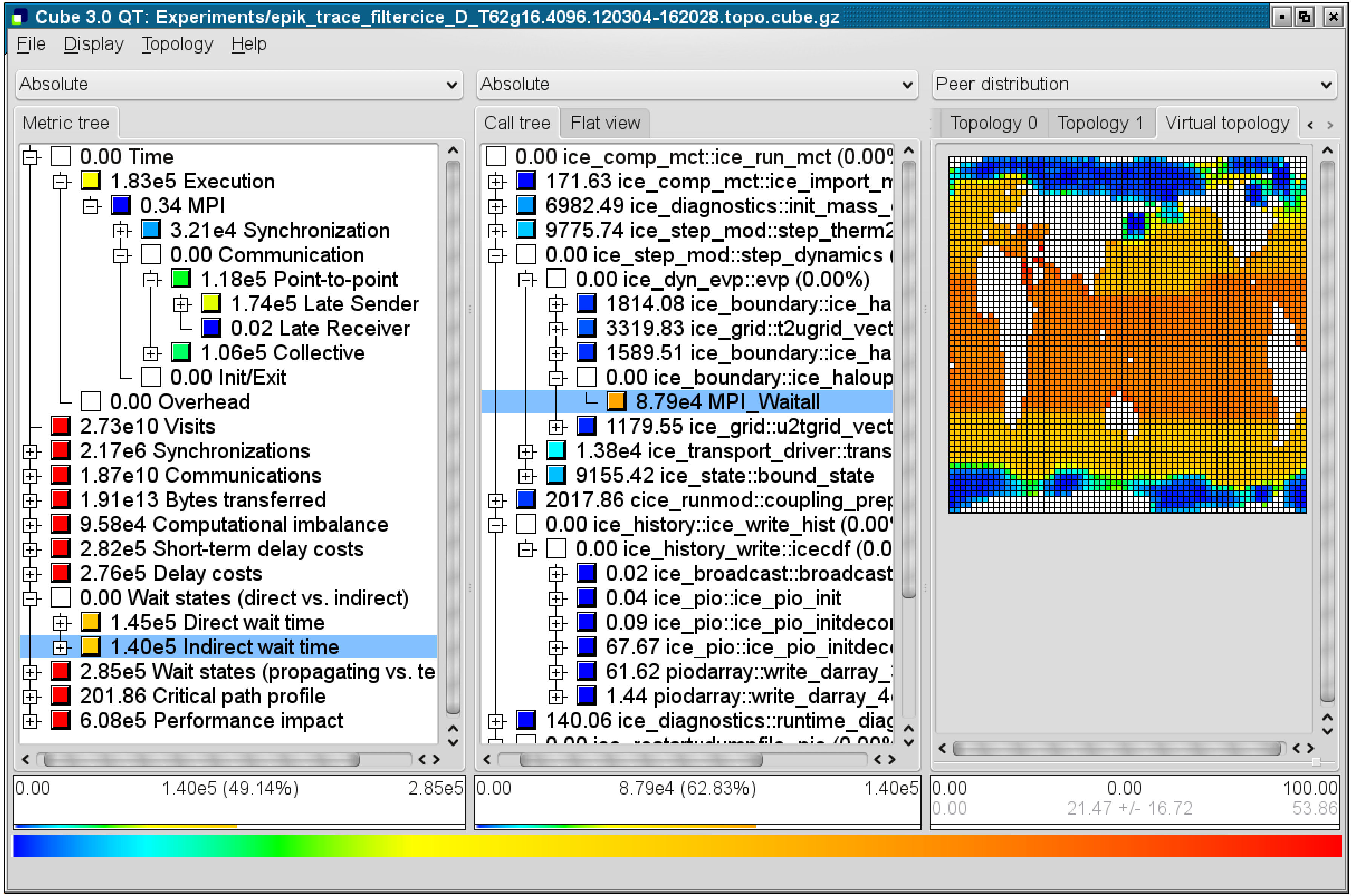
- Direct wait caused by ranks processing areas near the north and south ice borders



SCALASCA EXAMPLE: CESM SEA ICE MODULE

Indirect Wait Time Analysis

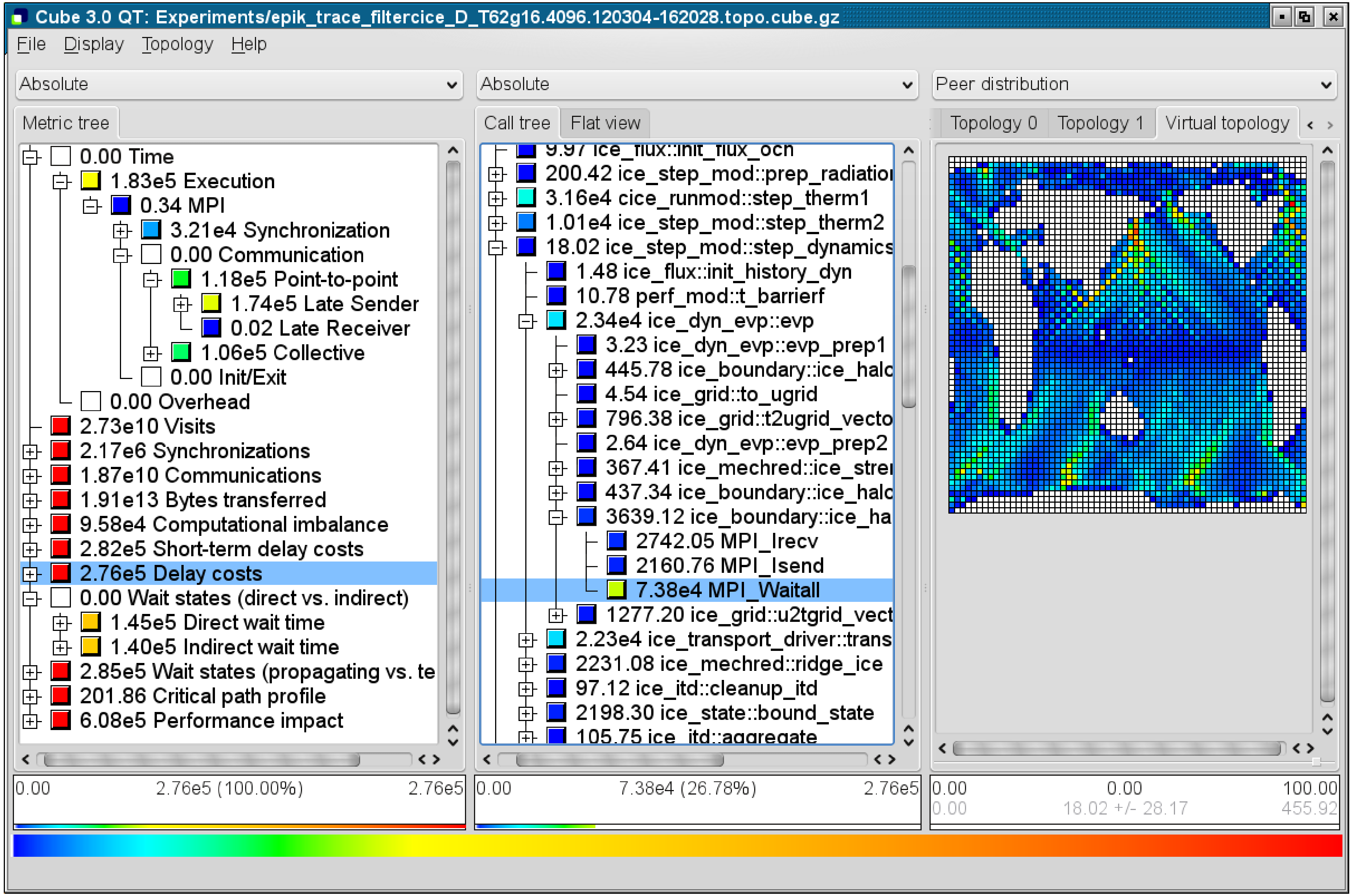
- Indirect waits occurs for ranks processing warmer areas



SCALASCA EXAMPLE: CESM SEA ICE MODULE

Delay Costs Analysis

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



Together we are strong

INTEGRATION

Mitglied der Helmholtz-Gemeinschaft

JÜLICH Forschungszentrum

JÜLICH SUPERCOMPUTING CENTRE

INTEGRATION

- Need integrated tool (environment) **for all levels of parallelization**
  - Inter-node (MPI, PGAS, SHMEM)
  - Intra-node (OpenMP, multi-threading, multi-tasking)
  - Accelerators (OpenACC, CUDA, OpenCL, and many more)

- Integration with **performance modeling and prediction**

- No tool fits all requirements
  - **Interoperability of tools**
  - Integration via open interfaces

STATUS: GPU SUPPORT* (BEYOND MPI+OPENMP)

Tool	GPU programming systems supported
TAU	AMD ROCm+HIP, Kokkos, OpenCL, OpenACC, CUDA • Plans to support OpenMP target
HPCToolkit	OpenMP target, CUDA, RAJA, Kokkos
Extrae/Paraver	CUDA, OpenCL, OmpSs • Plans to support OpenACC, OpenMP target
Score-P/Scalasca/Vampir	CUDA, OpenACC, OpenCL • Experimental support for Kokkos, OmpSs • Plans to support OpenMP target

* No publicly accepted definition what “XXX support” actually means

Score-P

Scalable performance measurement infrastructure for parallel codes

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

TECHNISCHE UNIVERSITÄT DRESDEN

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

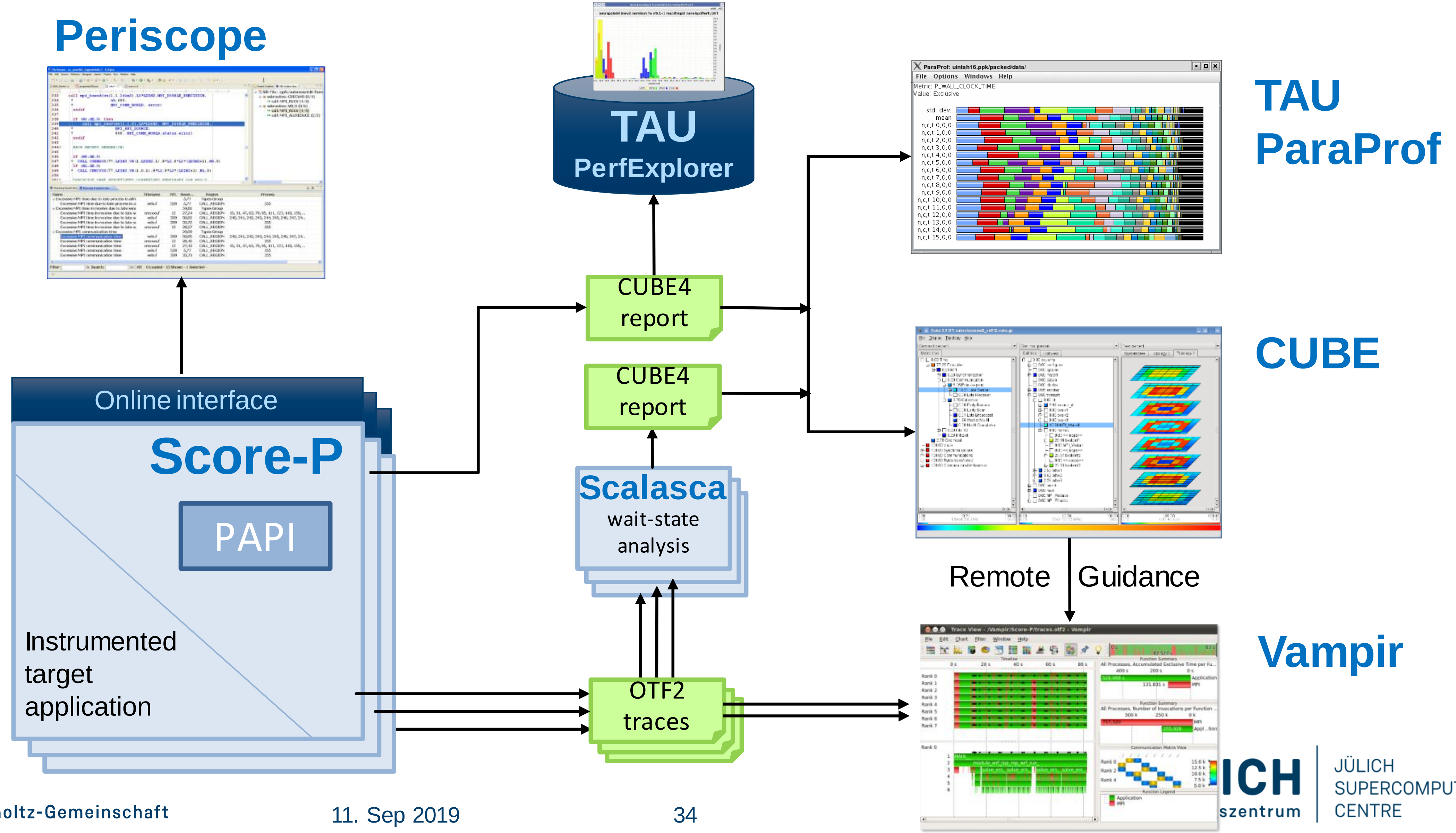
German Research School for Simulation Sciences

RWTH AACHEN UNIVERSITY

TUM Technische Universität München

UNIVERSITY OF OREGON

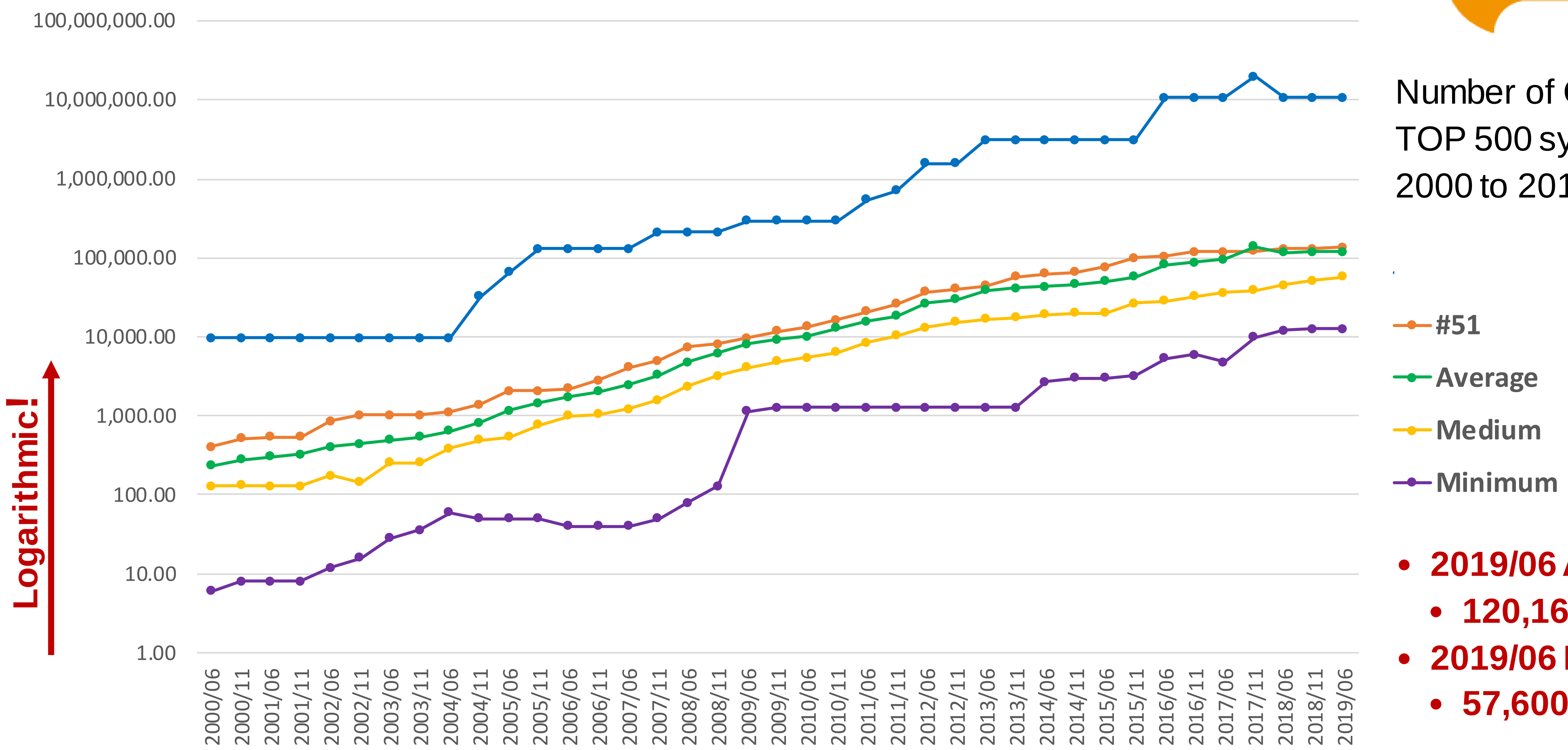
Score-P TOOL ECOSYSTEM



To infinity and beyond

EXTREME CONCURRENCY

TYPICAL HPC SYSTEM SIZE (NO. OF CORES)



Number of Cores
TOP 500 systems
2000 to 2019

- **2019/06 Avg:**
  - **120,160**
- **2019/06 Median:**
  - **57,600**

ROADS TO PERFORMANCE TOOL SCALABILITY

- **Scalable data collection and reduction**
  - Parallel collection + reduction based on MPI + parallel I/O (All tools)
  - Automatic detection of most important execution phases (Paraver)

- **Scalable parallel data analysis**
  - Parallel client/server processing and visualization (Vampir)
  - Parallel wait-state, delay and critical-path analysis (Scalasca)
  - Parallel analyzer and visualizer (Paraver)

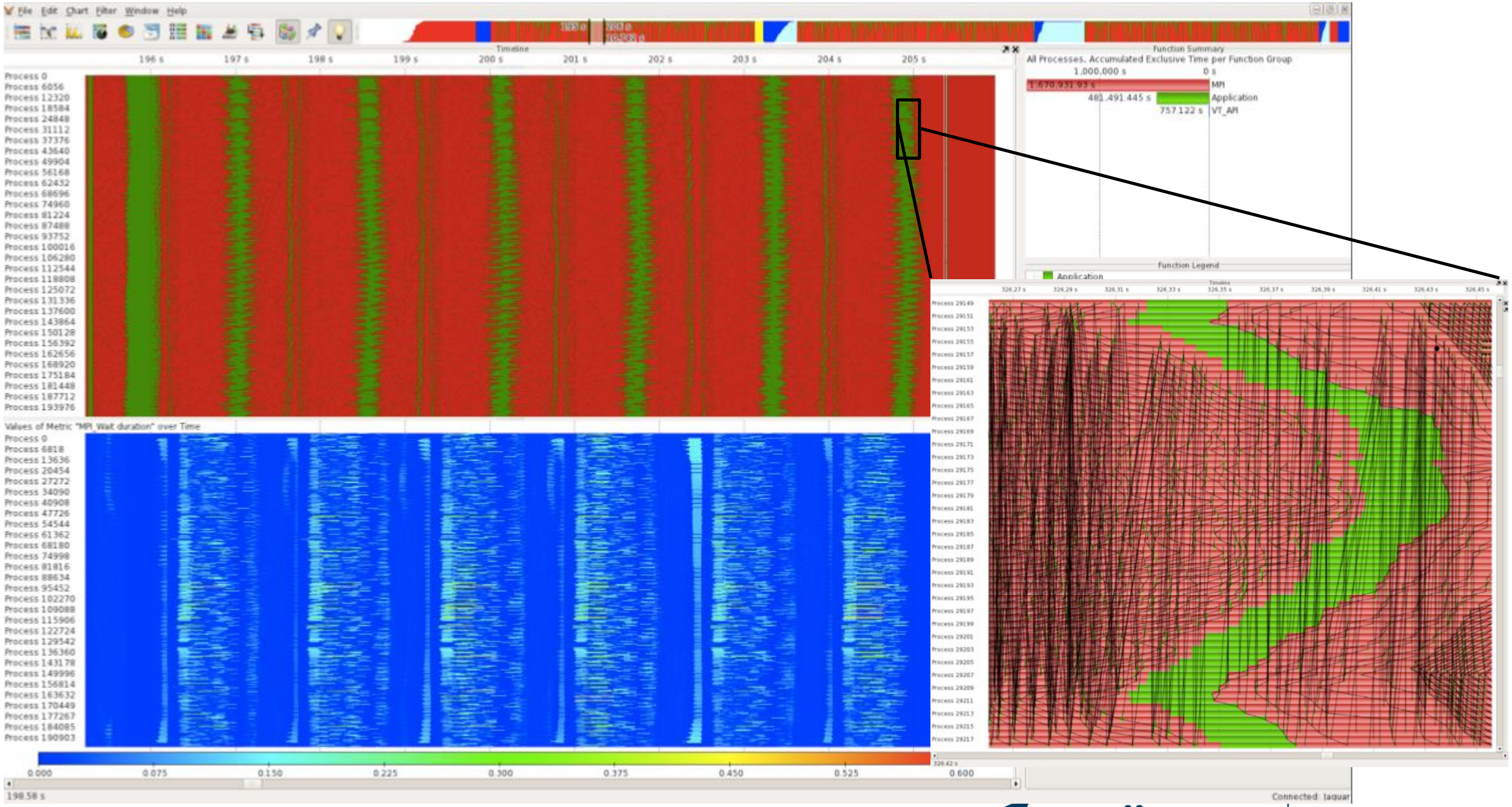
- **Scalable visualizations**
  - 3D charts and topology displays (TAU, Scalasca)
  - Hierarchical browsers (Scalasca)

STATUS: TOOLS SCALABILITY

Tool	Largest (stunt) run by developer	Max size expert user
TAU	786,432 processes • 48 racks Mira, BG/Q, ALCF • KG (Klein Gordon) code. MPI only User	O(100K)
HPCToolkit	64K processes • Cielo, SNL/LANL • Shock physics code User	O(10K) • ECP funded scalability enhancements by Q4/2019
Extrae/Paraver	64K processes • Cray XT5 • PFLOTRAN	O(1K)
Score-P/Scalasca	28,672 x 64 1,835,008 threads (28,672 x 64) • 28 racks JuQueen, BG/Q, JSC • Nekbone (CORAL benchmark)	O(100K)
Score-P/Vampirserver	200,448 processes • JaguarPF, OLCF • S3D (SNL) • Required 21,516 analysis processes	O(10K)

VAMPIRSERVER: TRACE VISUALIZATION S3D@200,448

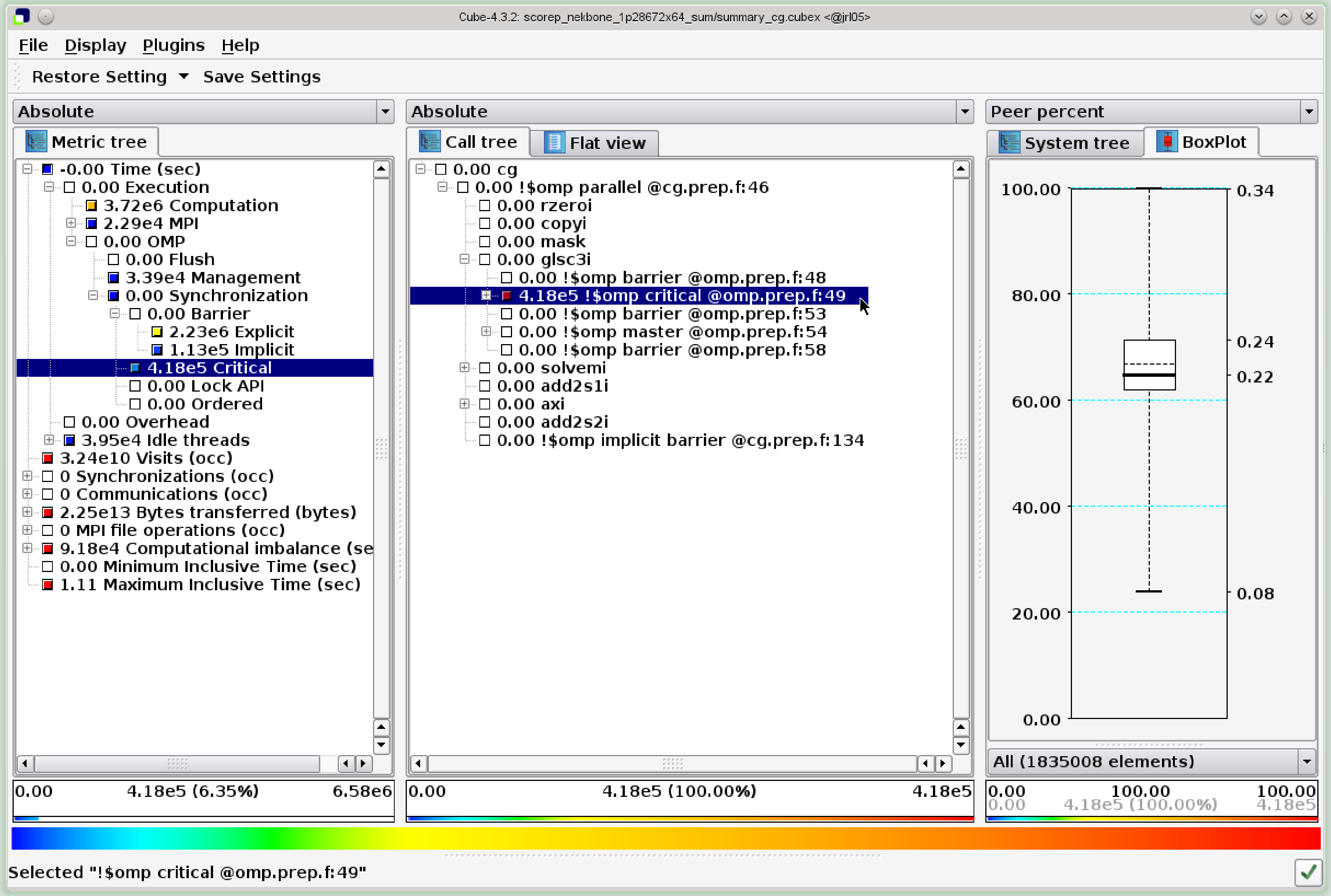
- OTF2 trace 4.5 TB
- Vampir Server running with 20,000 analysis processes



JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

SCALASCA: 1,835,008 THREADS TEST CASE

- Nekbone
- CORAL benchmark
- JuQueen experiment
- 28,672 x 64 = 1,835,008 threads
- Load imbalance at OpenMP critical section

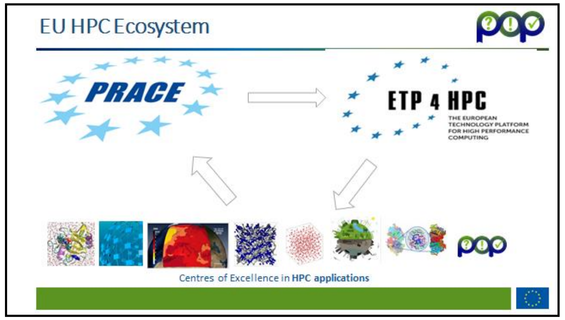


Do I really need that?

PERFORMANCE ASSESSMENT AS A SERVICE

POP CoE (https://pop-coe.eu)

- A **Centre of Excellence**
  - On **Performance Optimisation and Productivity**
  - Promoting **best practices in parallel programming**



- Providing **FREE Services**
  - Precise understanding of application and system behaviour
  - Suggestion/support on how to refactor code in the most productive way
- **Horizontal**
  - Transversal across application areas, platforms, scales
- **For (EU) academic AND industrial codes and users !**

Partners

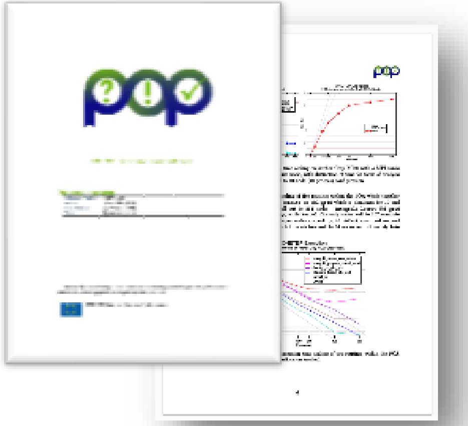
- **Who?**
  - BSC, ES (coordinator)
  - HLRS, DE
  - IT4I, CZ
  - JSC, DE
  - NAG, UK
  - RWTH Aachen, IT Center, DE
  - TERATEC, FR
  - UVSQ, FR

A team with

- Excellence in performance tools and tuning
- Excellence in programming models and practices
- Research and development background AND
  proven commitment in application to real academic and industrial use cases

FREE Services provided by the CoE

- **Parallel Application Performance Assessment**
  - Primary service
  - Identifies performance issues of customer code (at customer site)
  - If needed, identifies the root causes of the issues found and qualifies and quantifies approaches to address them (recommendations)
  - **Combines former Performance Audit (?) and Plan (!)**
  - Medium effort (1-3 months)



- **Proof-of-Concept (✓)**
  - Follow-up service
  - Experiments and mock-up tests for customer codes
  - Kernel extraction, parallelisation, mini-apps experiments to show effect of proposed optimisations
  - Larger effort (3-6 months)

Note: Effort shared between our experts and customer!

Status after 2½ Years (End of Phase1)

Performance Assessments

- 139 completed or reporting to customer
  - 13 more in progress

Proof-of-Concept

- 19 completed Proofs of Concept
  - 3 more in progress

Some PoC Success Stories

- See ⇨ https://pop-coe.eu/blog/tags/success-stories

  - Performance Improvements for SCM's ADF Modeling Suite
  - **3x Speed Improvement** for zCFD Computational Fluid Dynamics Solver
  - **25% Faster time-to-solution** for Urban Microclimate Simulations
  - **2x performance improvement** for SCM ADF code
  - Proof of Concept for BPMF leads to around **40% runtime reduction**
  - POP audit helps developers **double their code performance**
  - **10-fold scalability improvement** from POP services
  - POP performance study improves performance **up to a factor 6**
  - POP Proof-of-Concept study leads to **nearly 50% higher performance**
  - POP Proof-of-Concept study leads to **10X performance improvement** for customer

Improvements

Reductions

ROI Examples

Application Savings after POP Proof-of-Concept

- POP PoC resulted in 72% faster-time-to-solution
- Production runs on ARCHER (UK national academic supercomputer)
- Improved code saves €15.58 per run
- Yearly savings of around €56,000 (from monthly usage data)

Application Savings after POP Performance Plan

- Cost for customer implementing POP recommendations: €2,000
- Achieved improvement of 62%
- €20,000 yearly operating cost
- Resulted in yearly saving of €12,400 in compute costs ⇨ ROI of 620%

What does not work right now very well

OUTSTANDING ISSUES

FUTURE WORK

- **Memory and vectorization** performance analysis
  - Hard to capture performance data
    - Only possible if suitable hardware counters are provided
    - VERY processor specific ⇨ hard for open-source portable tools
- Trend towards **task-based / asynchronous programming models**
  - Very dynamic execution might be non reproducible ⇨ off-line tools fail
  - Hard to get the “big picture” ⇨ good high-level metrics still missing here
- Trend towards more **modern programming languages (Python, C++)**
  - How to automatically instrument template-based frameworks and programming styles?
  - How to present the data on Python level (and not on the interpreter lowlevel)?
- Performance assessment of **data analytics codes**

USEFUL RESOURCES

Overview Parallel Performance and Debugging Tools

- http://pramodkumbhar.com/2017/04/summary-of-profiling-tools/
- http://pramodkumbhar.com/2018/06/summary-of-debugging-tools/
- http://pramodkumbhar.com/2019/05/summary-of-python-profiling-tools-part-i/

MY REQUEST

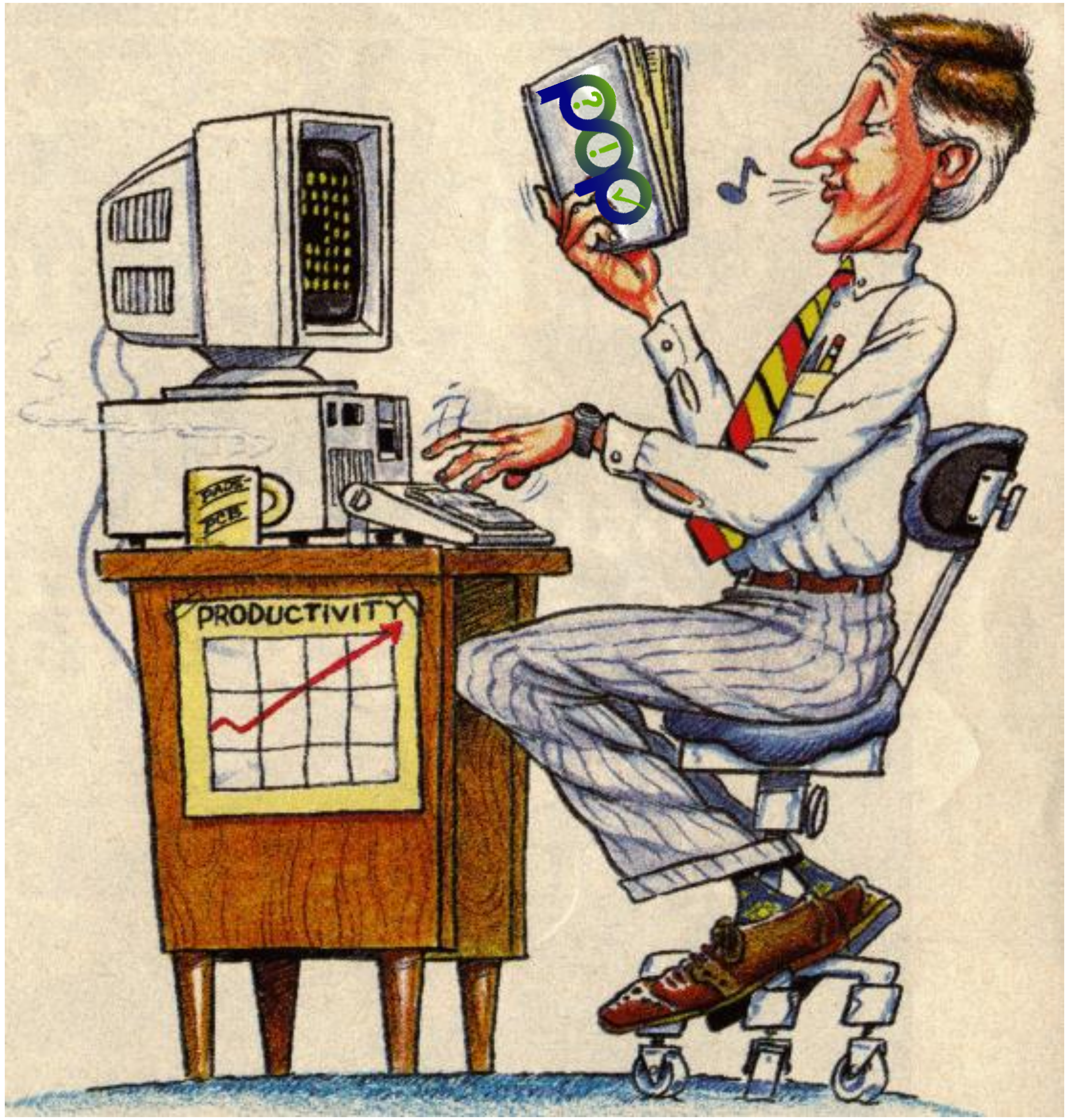
- **Give performance tools a chance!**

- **It will require effort**
  - Need to read and understand tool documentation
  - Attend tool tutorial at conference or tool training at HPC centres
  - Attend tuning workshops or performance hackathons

- **Do not give up at the first thing that does not work**
  - Ask for help from tool developers
  - Report tool (and documentation) bugs

JÜLICH Forschungszentrum JÜLICH SUPERCOMPUTING CENTRE

PERFORMANCE TUNING: STILL A PROBLEM?



Performance Optimisation and Productivity

A Centre of Excellence in HPC

Contact:

https://www.pop-coe.eu

mailto:pop@bsc.es

@POP_HPC

youtube.com/c/POPHPC

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

QUESTIONS?

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

BACKUP

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**

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

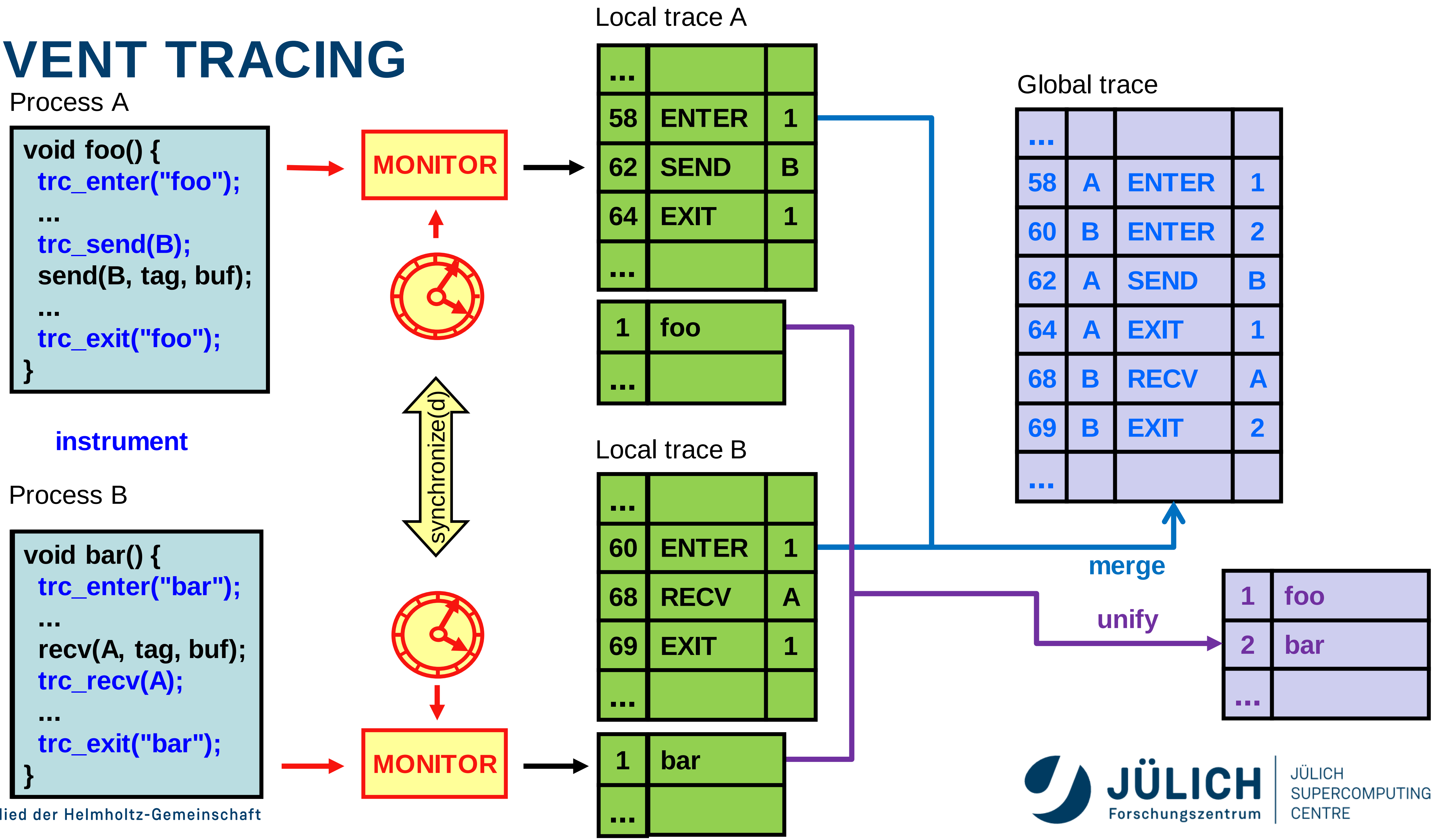
Disadvantages

– Can only be used for short durations or small configurations

– HUGE trace files

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

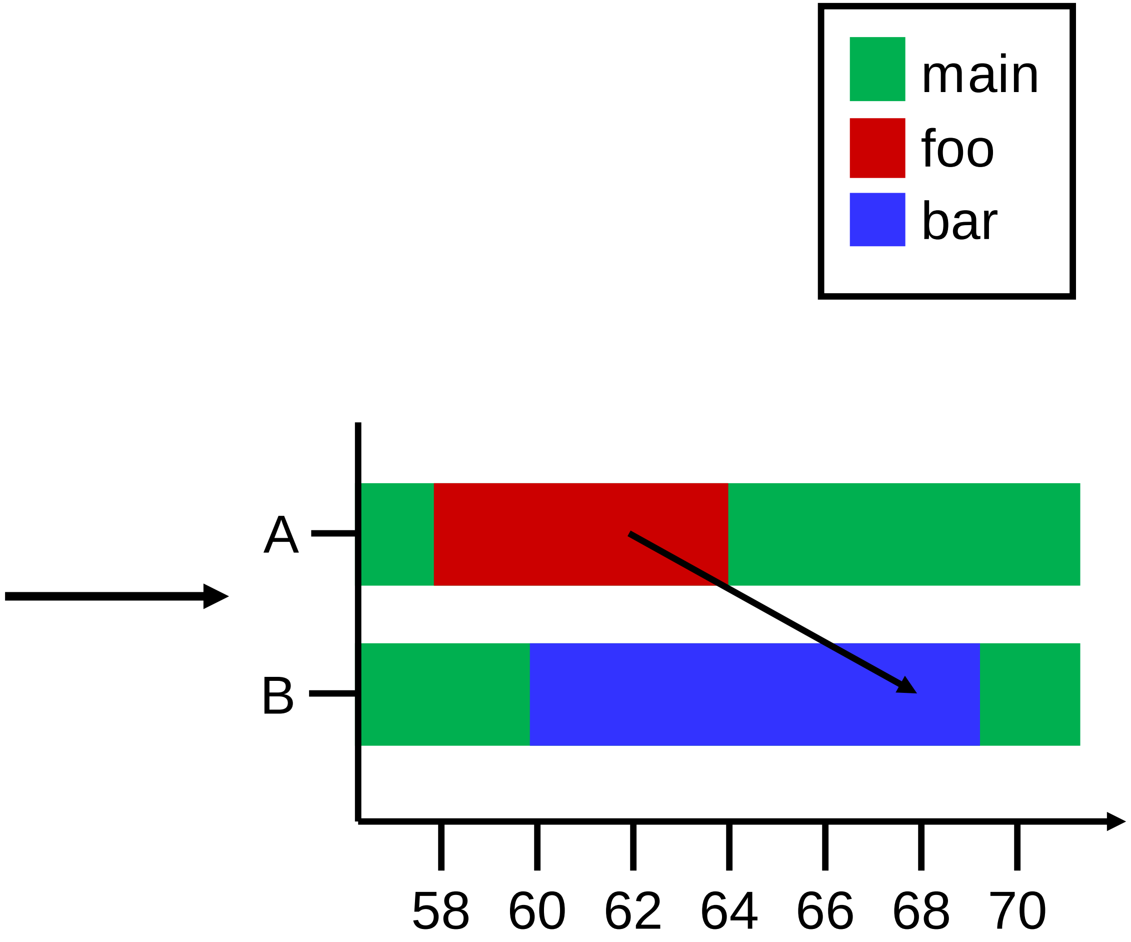
EVENT TRACING



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
...			

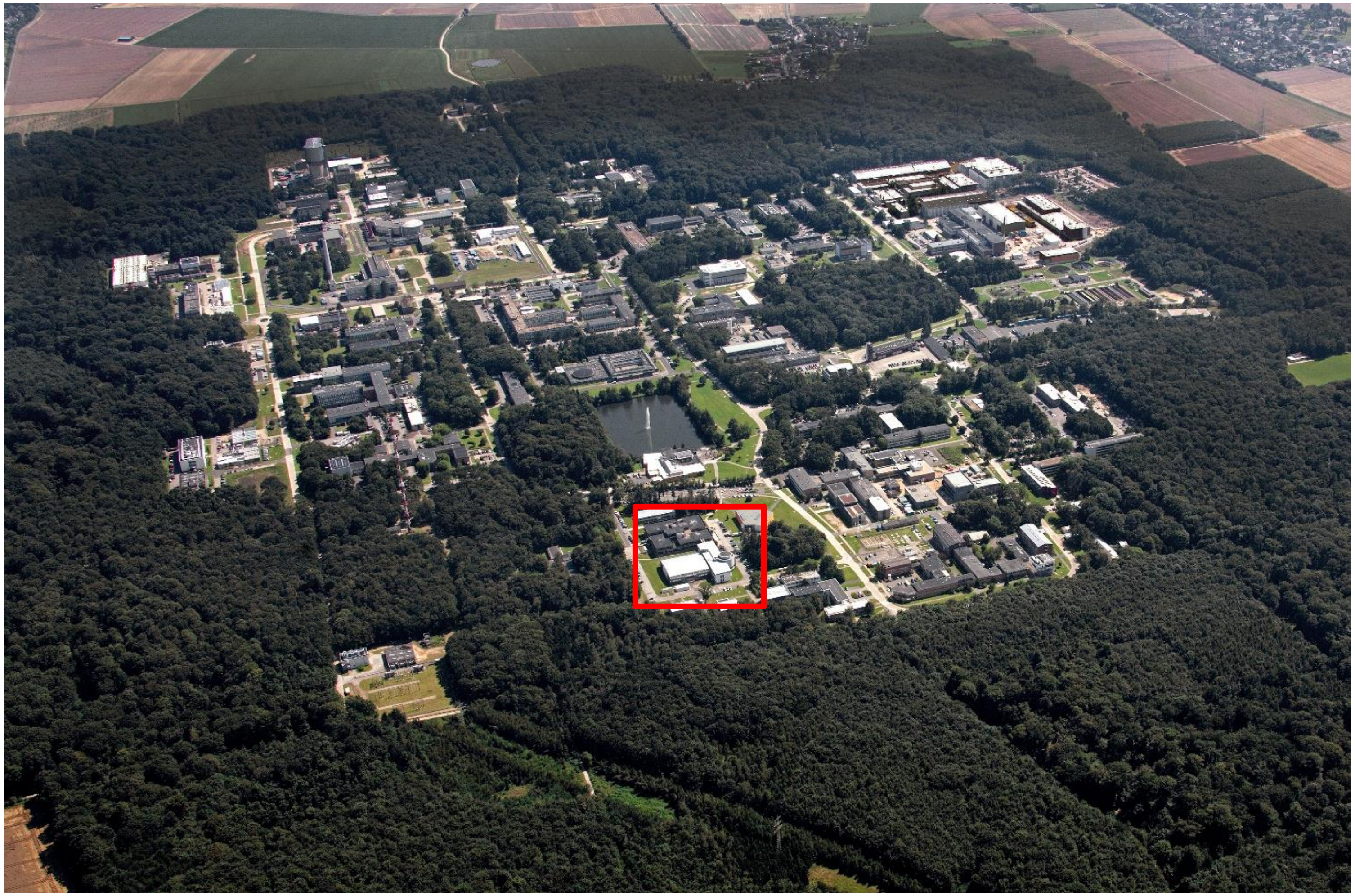


Forschungszentrum Jülich GmbH

JÜLICH SUPERCOMPUTING CENTRE

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

FORSCHUNGSZENTRUM JÜLICH GMBH



- Germany's largest national laboratory
- About 5800 employees
- Research areas
  - Information technology
  - Health (Neuroscience / brain research)
  - Energy
  - Atmosphere + Climate

JÜLICH SUPERCOMPUTING CENTRE (JSC)

HPC Centre for

- Forschungszentrum Jülich
- Jülich Aachen Research Alliance (JARA)
- Germany as GCS (1 of 3 German National Centres)
- Europe (1st European Centre inside PRACE)

JSC MACHINE HALL (JULY 2018)

You KNOW YOU made it …

… WHEN LARGE COMPANIES "COPY" YOUR STUFF

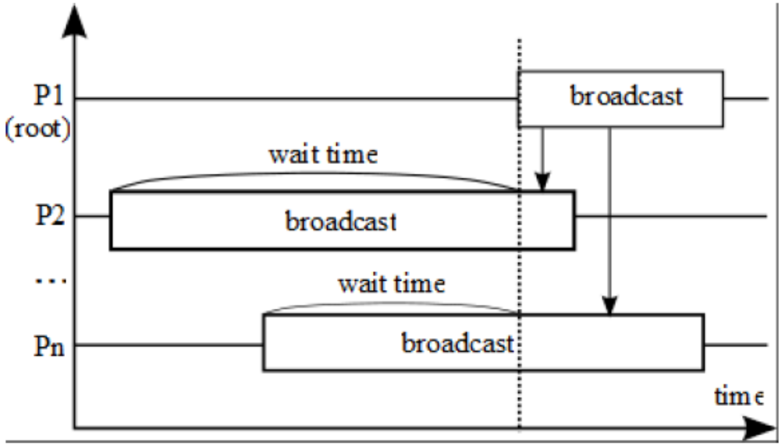
Introducing the Intel® Trace Analyzer and Collector Performance Assistant

Motivation: Improve method of performance analysis via the GUI

Solution:

- Define common/known performance problems
- Automate detection via the Intel® Trace Analyzer

Example: A "Late Broadcast" is not easy to identify with existing views



Source:
https://software.intel.com/en-us/videos/quickly-discover-performance-issues-with-the-intel-trace-analyzer-and-collector-90-beta

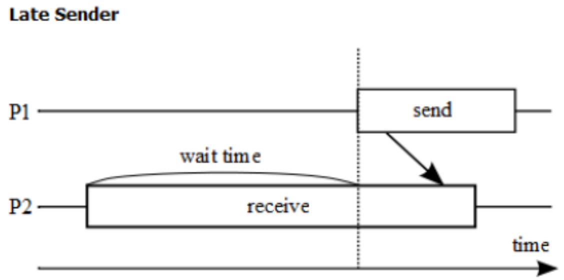
Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Optimization Notice

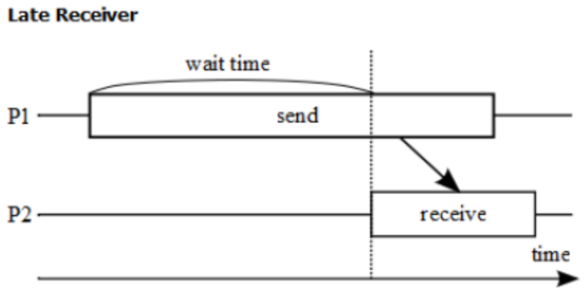
Which Performance Issues are automatically identified?

Point-to-point exchange problems:

- Late Sender

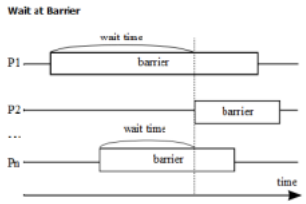


- Late Receiver

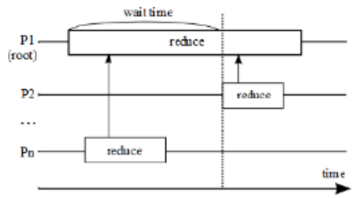


Problems with global collective operation performance:

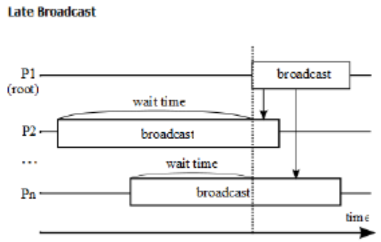
- Wait at Barrier



- Early Reduce



- Late Broadcast



Source:
https://software.intel.com/en-us/videos/quickly-discover-performance-issues-with-the-intel-trace-analyzer-and-collector-90-beta

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Optimization Notice

Together we are strong

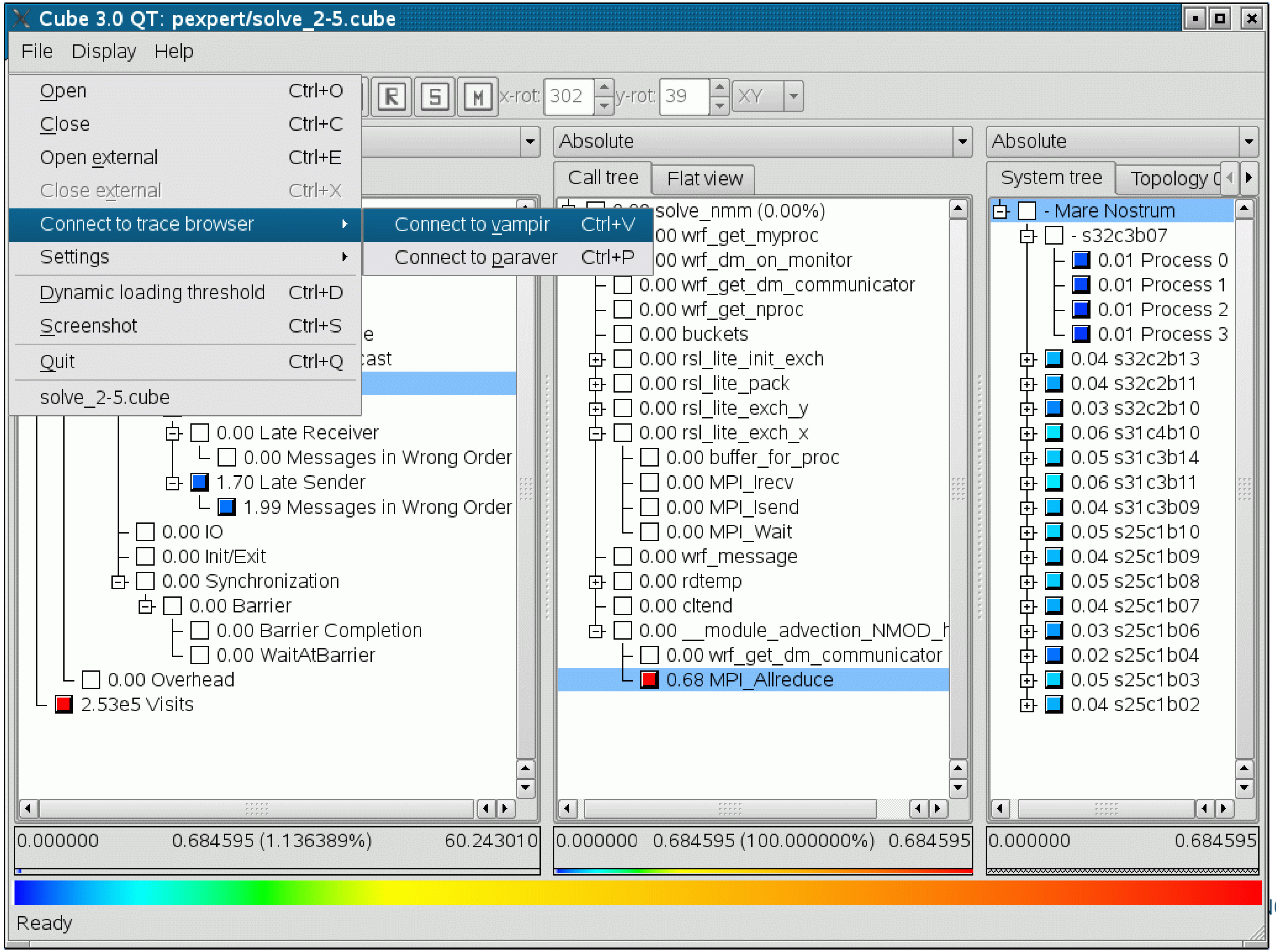
INTEGRATION

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)
  - **In any combination!**
- Flexible **measurement** without re-compilation:
  - Basic and advanced **profile** generation (⇨ CUBE4 format)
  - Event **trace** recording (⇨ OTF2 format)
  - Online access to profiling data
- Highly scalable I/O functionality
- Support all fundamental concepts of partner's tools

SCALASCA ⇨ VAMPIR INTEGRATION

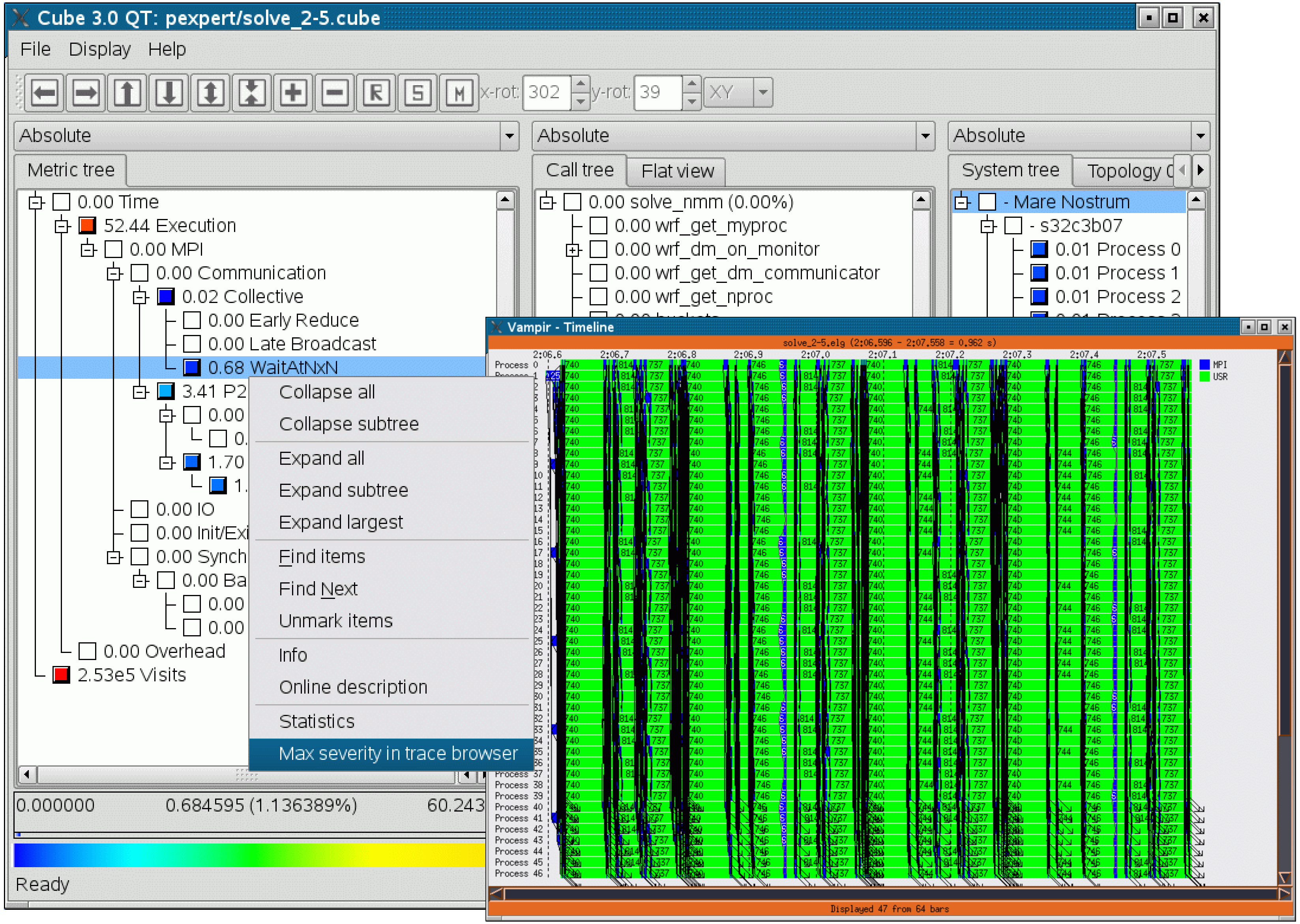
1. Connect to Vampir
   - Loads underlying trace



SCALASCA ⇨ VAMPIR INTEGRATION

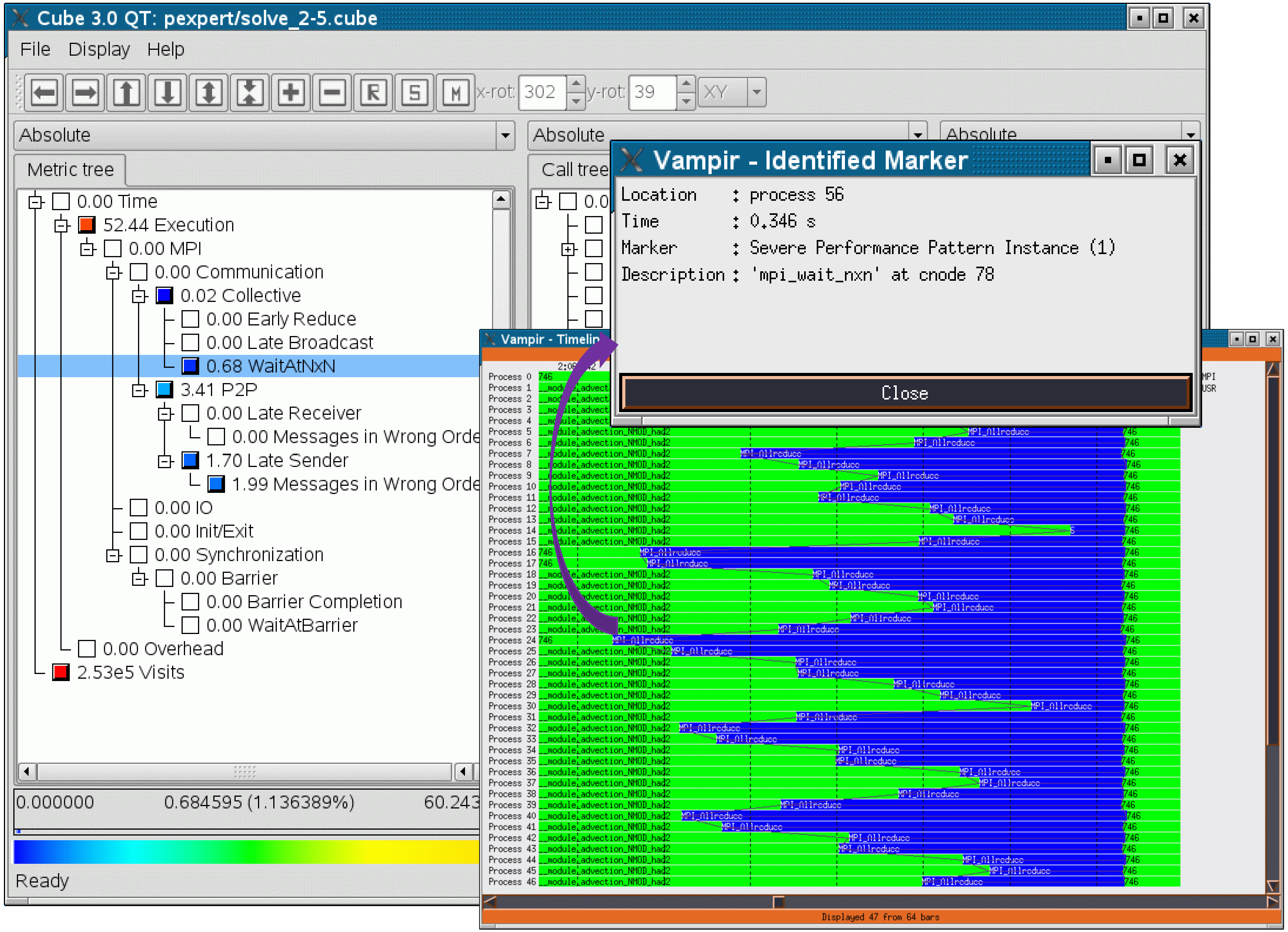
1. Connect to Vampir
   - Loads underlying trace

2. Use context menu
   - Max severity
   - Zooms to corresponding view



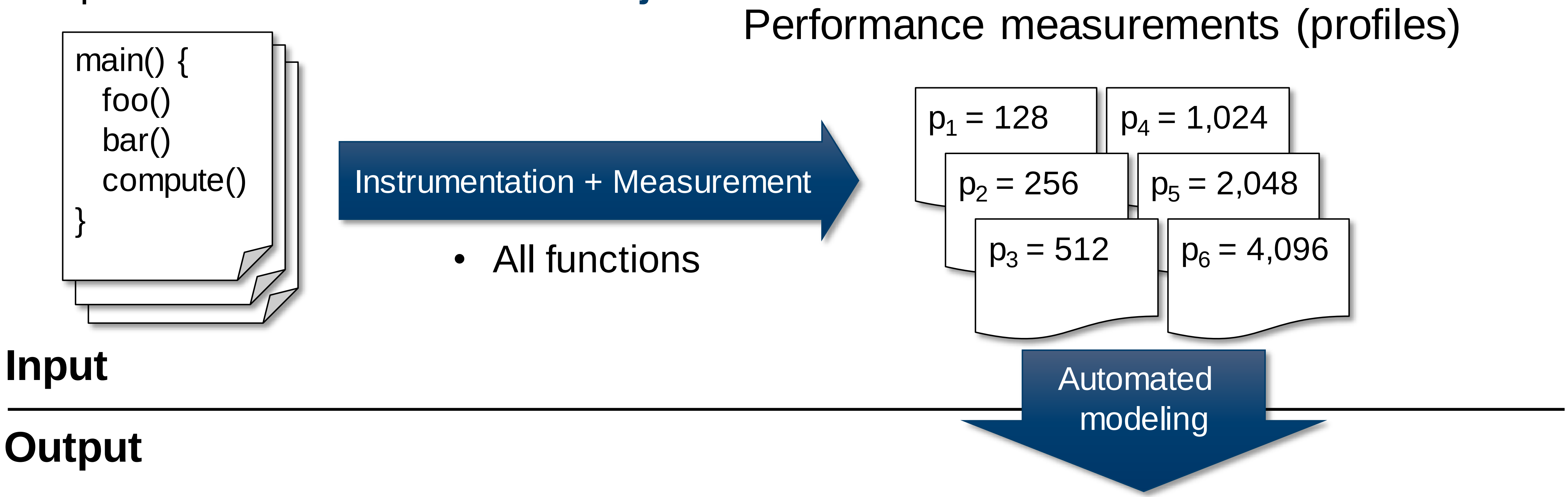
SCALASCA ⇨ VAMPIR INTEGRATION

1. Connect to Vampir
   - Loads underlying trace
2. Use context menu
   - Max severity
   - Zooms to corresponding view
3. Use extensive Vampir features to investigate further



INTEGRATION OF MEASUREMENT AND MODELLING

- Example: **DFG SPPEXA Catwalk Project**



Rank	Function	Model [s]
1	bar()	4.0 * p + 0.1*log(p)
2	compute()	0.5 * log(p)
3	foo()	65.7

CATWALK: RESULT VISUALIZATION

- Reusing Cube result browser
- However: browsing functions instead of values