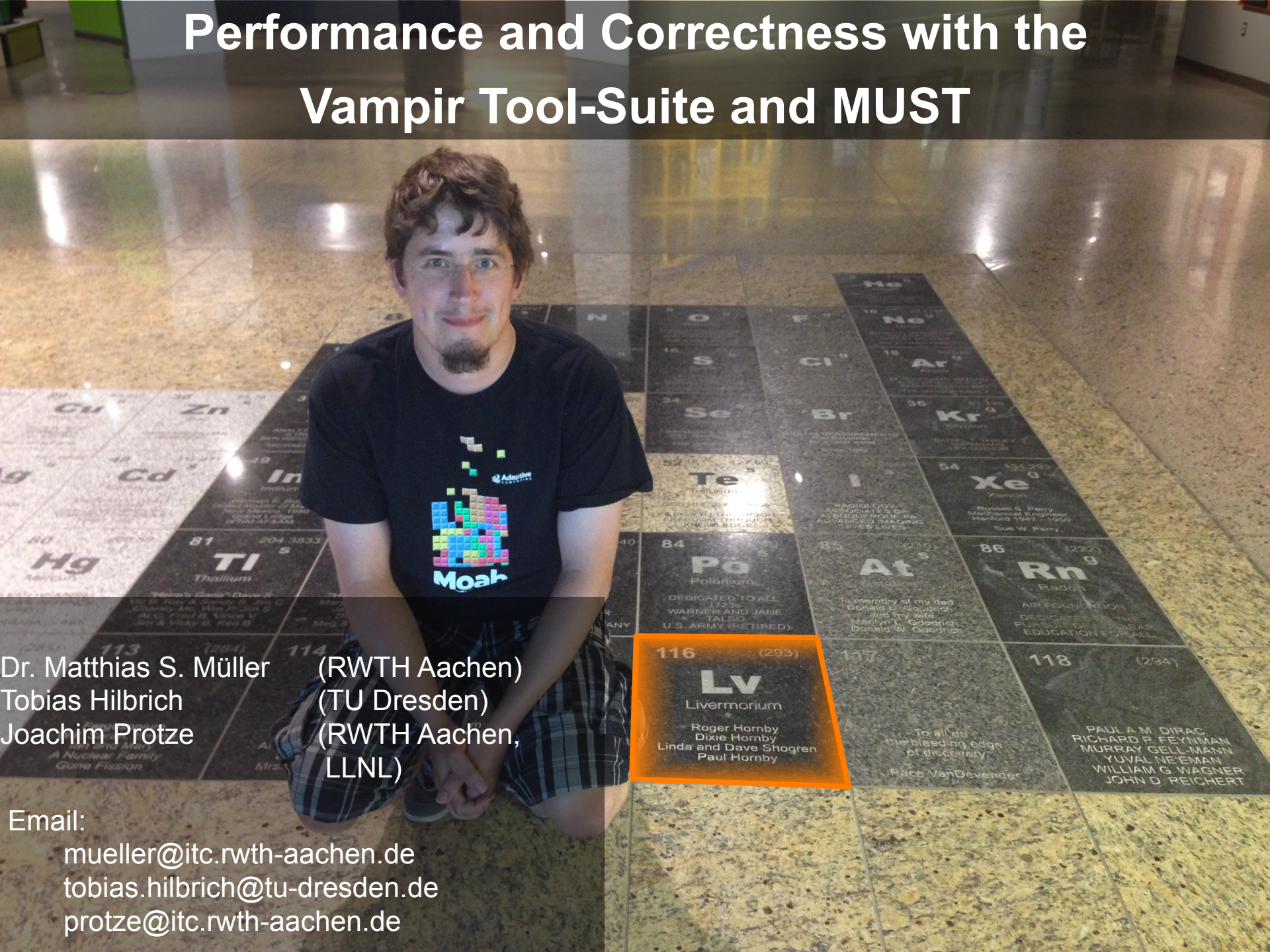
Performance and Correctness with the Vampir Tool-Suite and MUST



Dr. Matthias S. Müller (RWTH Aachen)
Tobias Hilbrich (TU Dresden)
Joachim Protze (RWTH Aachen, LLNL)

Email:
mueller@itc.rwth-aachen.de
tobias.hilbrich@tu-dresden.de
protze@itc.rwth-aachen.de

TECHNISCHE UNIVERSITÄT DRESDEN

RWTH AACHEN UNIVERSITY

Center for Information Services and High Performance Computing (ZIH)

Performance and Correctness with the Vampir Tool-Suite and MUST

Dr. Matthias S. Müller (RWTH Aachen University)
Tobias Hilbrich (Technische Universität Dresden)
Joachim Protze (RWTH Aachen University, LLNL)

Email:
mueller@itc.rwth-aachen.de
tobias.hilbrich@tu-dresden.de
protze@itc.rwth-aachen.de

ZIH
Center for Information Services & High Performance Computing

Agenda

9:00am-10:30am	Vampir – Performance Analysis
Short break	
10:45am-noon	MUST – Correctness Checking
Lunch break	
1:15pm-2:45pm	Hands-On
Short break	
3:00pm-4:30pm	Advanced Use Cases and Discussion

TECHNISCHE UNIVERSITÄT DRESDEN

RWTH AACHEN UNIVERSITY

Center for Information Services and High Performance Computing (ZIH)

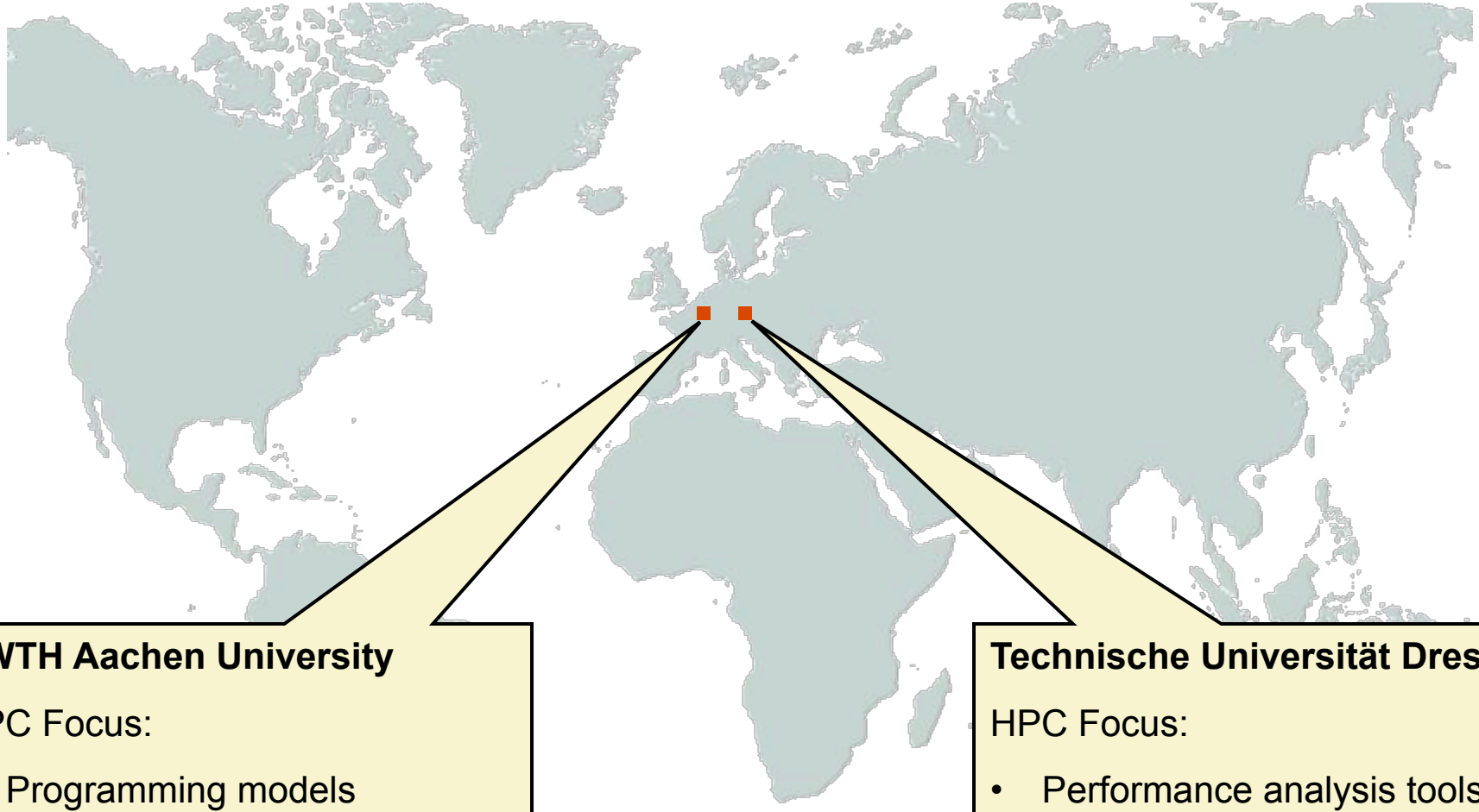
Welcome

Dr. Matthias S. Müller (RWTH Aachen University)
Tobias Hilbrich (Technische Universität Dresden)
Joachim Protze (RWTH Aachen University, LLNL)

Email:
mueller@itc.rwth-aachen.de
tobias.hilbrich@tu-dresden.de
protze@itc.rwth-aachen.de

ZIH
Center for Information Services & High Performance Computing

About us



RWTH Aachen University

HPC Focus:

- Programming models
- Correctness tools
- Immersive visualization

RWTH AACHEN UNIVERSITY

Technische Universität Dresden

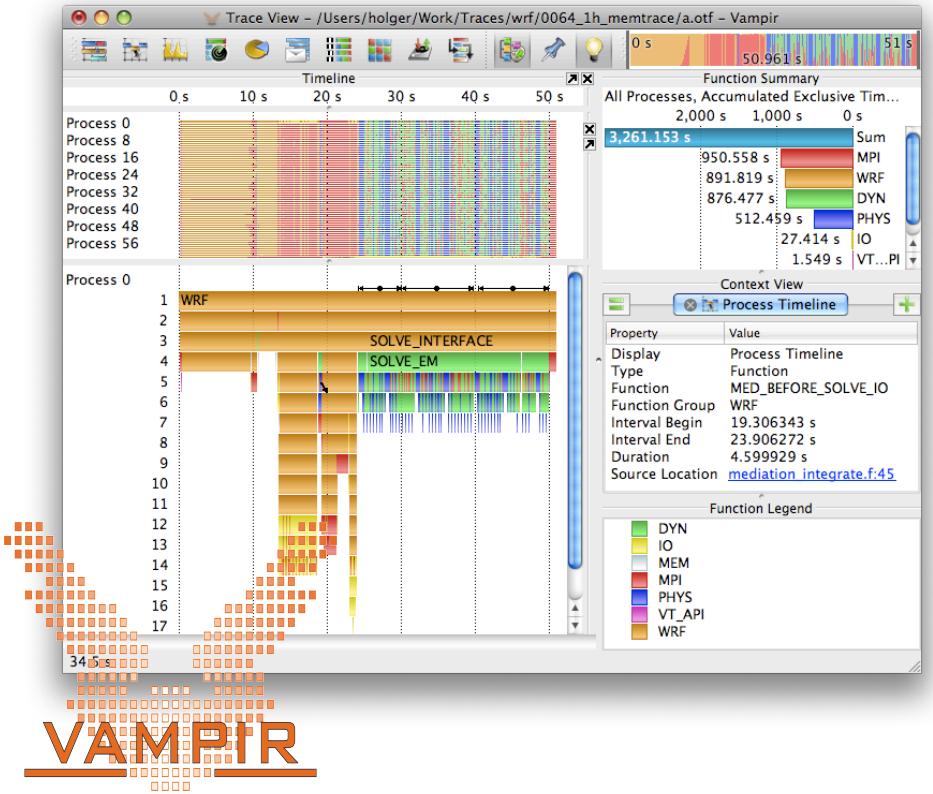
HPC Focus:

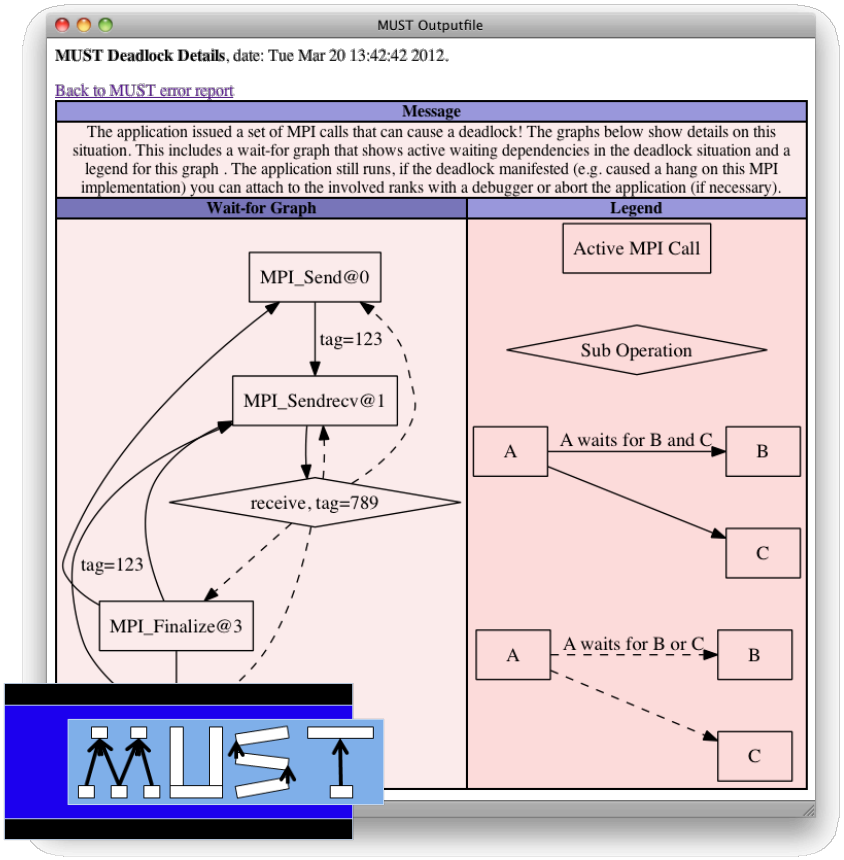
- Performance analysis tools
- Data-intensive computing
- Energy efficiency

TECHNISCHE UNIVERSITÄT DRESDEN

Tools

- Tools assist you in your HPC development:





Vampir: Performance optimization

MUST: Correct MPI usage

Tool Consortia – Performance

- Community efforts to make tools more versatile

Measurement: **Score-P**

Partners:

Technische Universität München

gns

JÜLICH FORSCHUNGSZENTRUM

TECHNISCHE UNIVERSITÄT DRESDEN

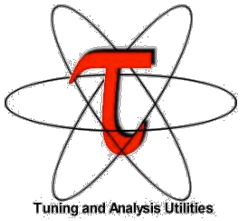
RWTH AACHEN UNIVERSITY

German Research School for Simulation Sciences

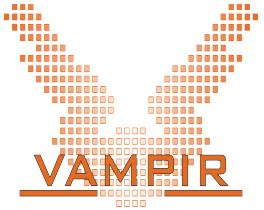
UNIVERSITY OF OREGON

Analysis:

- TAU



- **Vampir**



- Scalasca

Tool Consortia – Correctness

- Merging runtime MPI correctness approaches

Marmot
Developers:

TECHNISCHE UNIVERSITÄT DRESDEN

H L R S

Issue:
- Lack of Non-local checks

Umpire
Developers:

Lawrence Livermore National Laboratory

Issue:
- Scalability, local checks

MUST
Developers:

RWTH AACHEN UNIVERSITY

TECHNISCHE UNIVERSITÄT DRESDEN

Lawrence Livermore National Laboratory

Goal:
- Scalable, Push-button, no false positives

Experts on Site

- PnMPI, GTI, MUST, Vampir basics:
  - Joachim Protze (protze1@llnl.gov till Sept.)
- ScoreP:
  - David Boehme (boehme1@llnl.gov till 2017)

Follow-Up

- Do not hesitate to contact us
- LLNL on-site experts:
  - Joachim Protze (protze1@llnl.gov till Sept.)
  - David Boehme (boehme1@llnl.gov till 2017)
- Upcoming Score-P/Vampir/MUST tutorials:
  - EuroMPI 2014 tutorial:

„Practical Parallel Application Performance Engineering“

  - SC’14 tutorial:

„Hands-On Practical Hybrid Parallel Application Performance Engineering”

  - SC’14 tutorial:

„Efficient Parallel Debugging for MPI, Threads, and Beyond”

TECHNISCHE UNIVERSITÄT DRESDEN

RWTH AACHEN UNIVERSITY

Center for Information Services and High Performance Computing (ZIH)

Session 1: Vampir

Performance Analysis

Dr. Matthias S. Müller (RWTH Aachen University)
Tobias Hilbrich (Technische Universität Dresden)
Joachim Protze (RWTH Aachen University, LLNL)

Email:
mueller@itc.rwth-aachen.de
tobias.hilbrich@tu-dresden.de
protze@itc.rwth-aachen.de

ZIH
Center for Information Services & High Performance Computing

Content

- Motivation
- Workflow
- Use Cases
- Tracing
- Visualization
- Advanced

Content

- **Motivation**
- Workflow
- Use Cases
- Tracing
- Visualization
- Advanced

Motivation

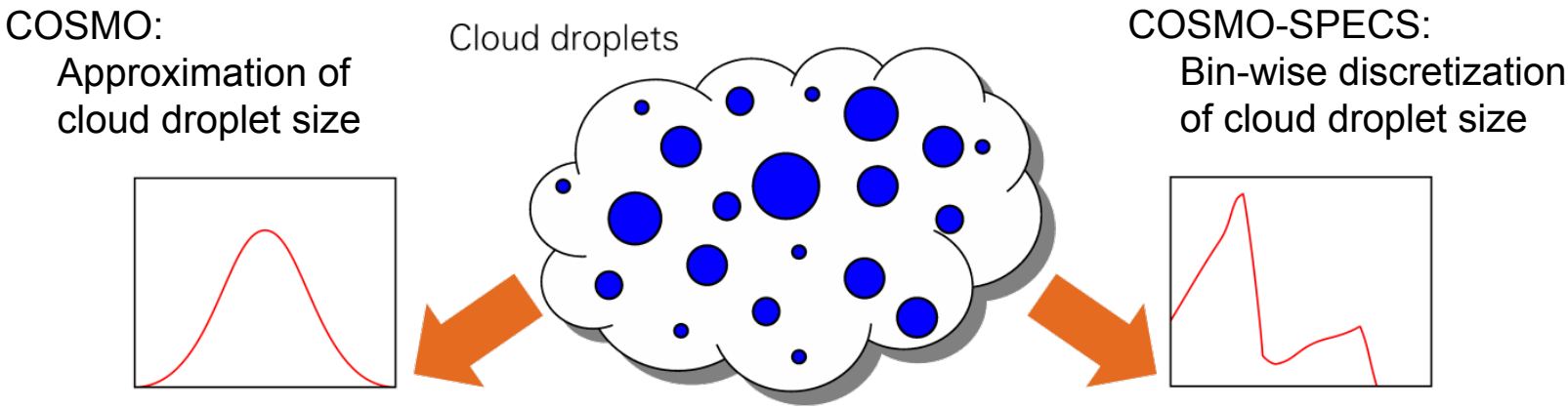
What happens in my application execution during a given time in a given process or thread?

How do the communication patterns of my application execute on a real system?

Are there any imbalances in computation, I/O or memory usage and how do they affect the parallel execution of my application?

Motivation – Case Study

- COSMO-SPECS a coupling of:
  - Weather forecast model
  - Detailed cloud microphysics scheme



Developer observation:

Runtime per iteration increases over time, why?

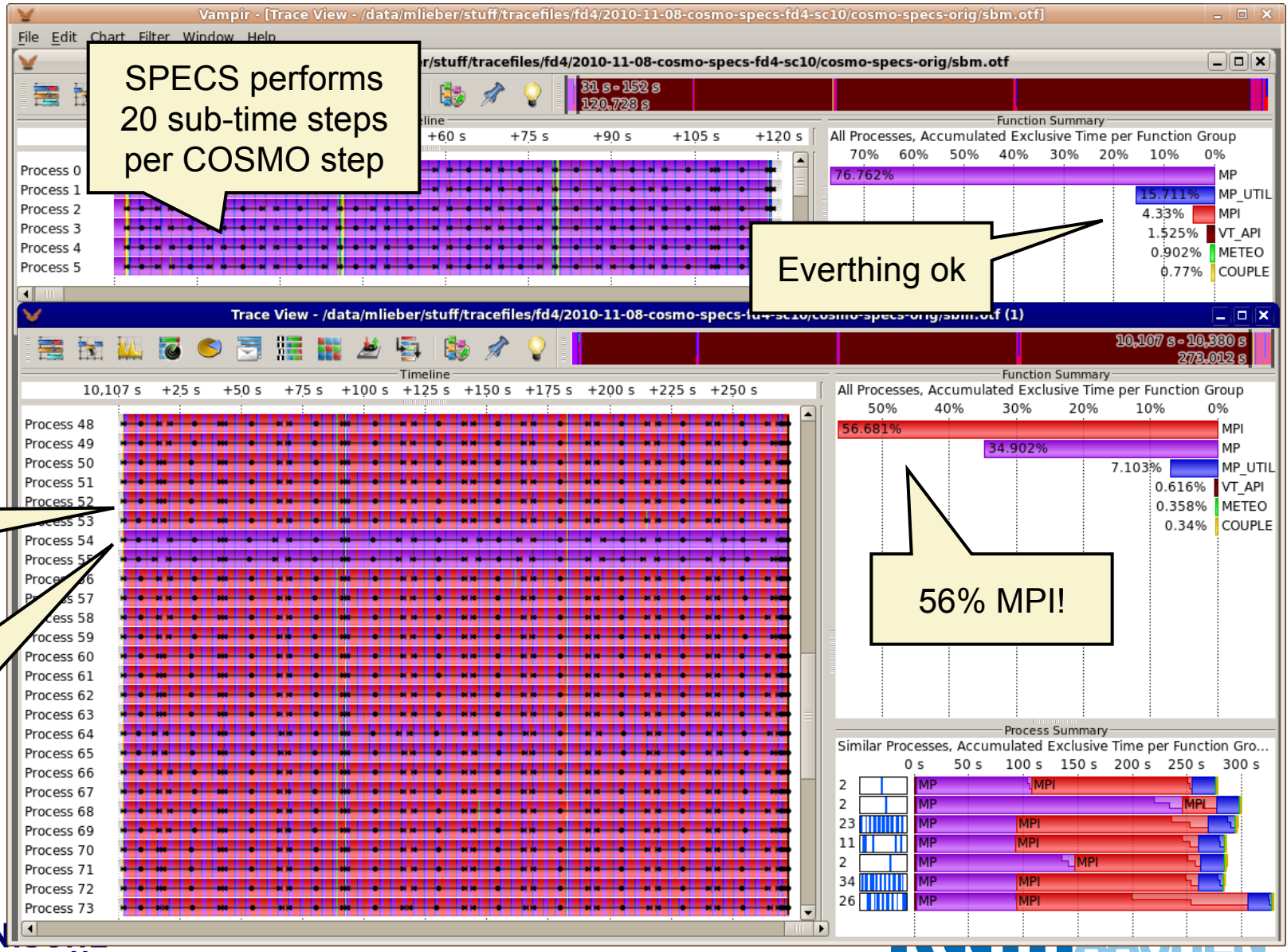
Motivation – Case Study (2)

First 3 time steps of COSMO-SPECS run

Last 3 time steps of COSMO-SPECS run

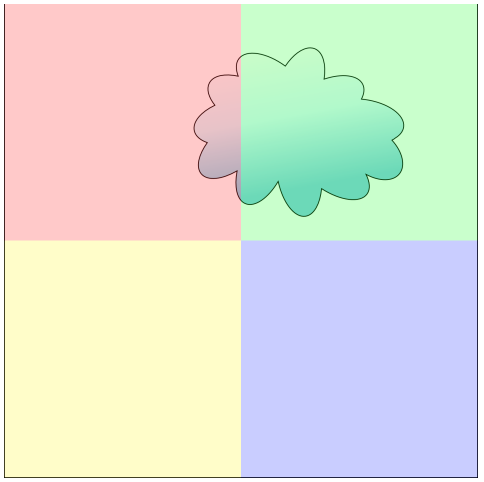
Heavy load imbalance

Cloud grows in grid cells of these MPI ranks



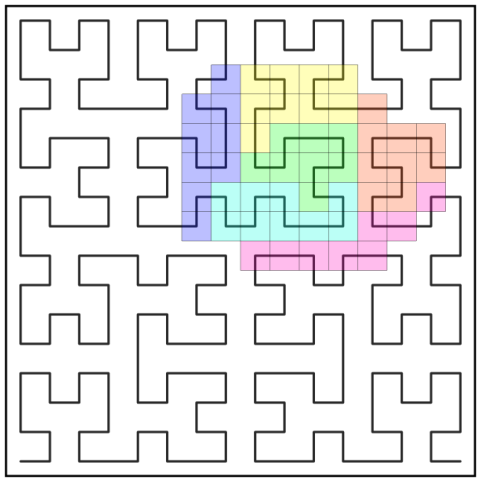
Motivation – Case Study (3)

– Domain decomposition:
  – 2D (horizontal) decompostion into MxN processor domains
  – No dynamic load balancing



Lesson learned:

Domain decomposition for detailed cloud microphysics must adapt to presence of clouds.



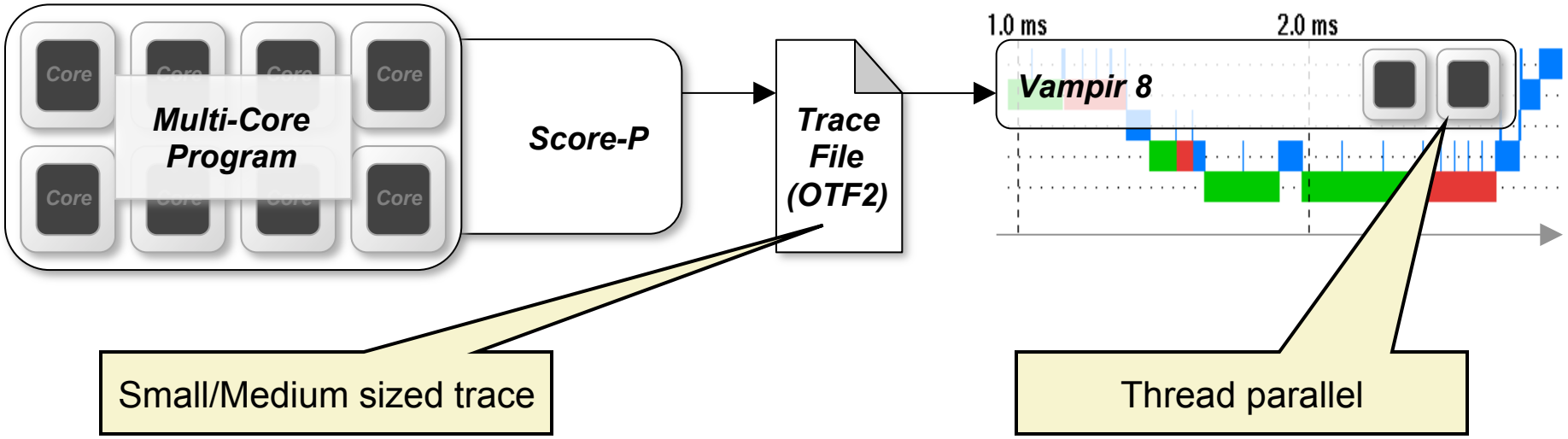
Content

- Motivation
- **Workflow**
- Use Cases
- Tracing
- Visualization
- Advanced

Workflow – Small Scale

- Directly on front end or local machine

```
% module load vampir
% vampir
```

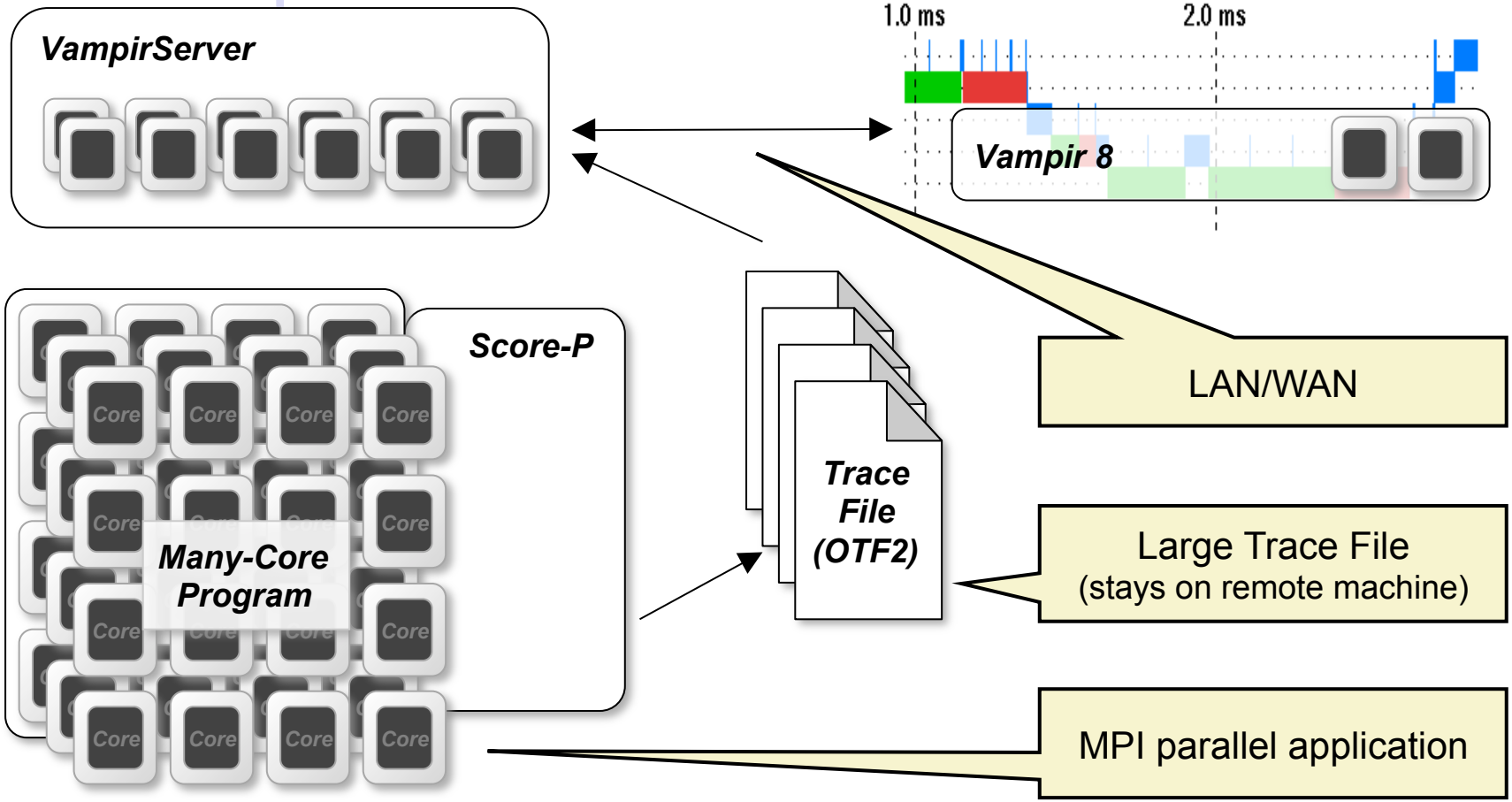


Workflow – Large Scale

- On local machine with remote VampirServer

```
% module load UNITE vampirserver
% vampirserver start -n 12
```

```
% module load UNITE vampir
% vampir
```

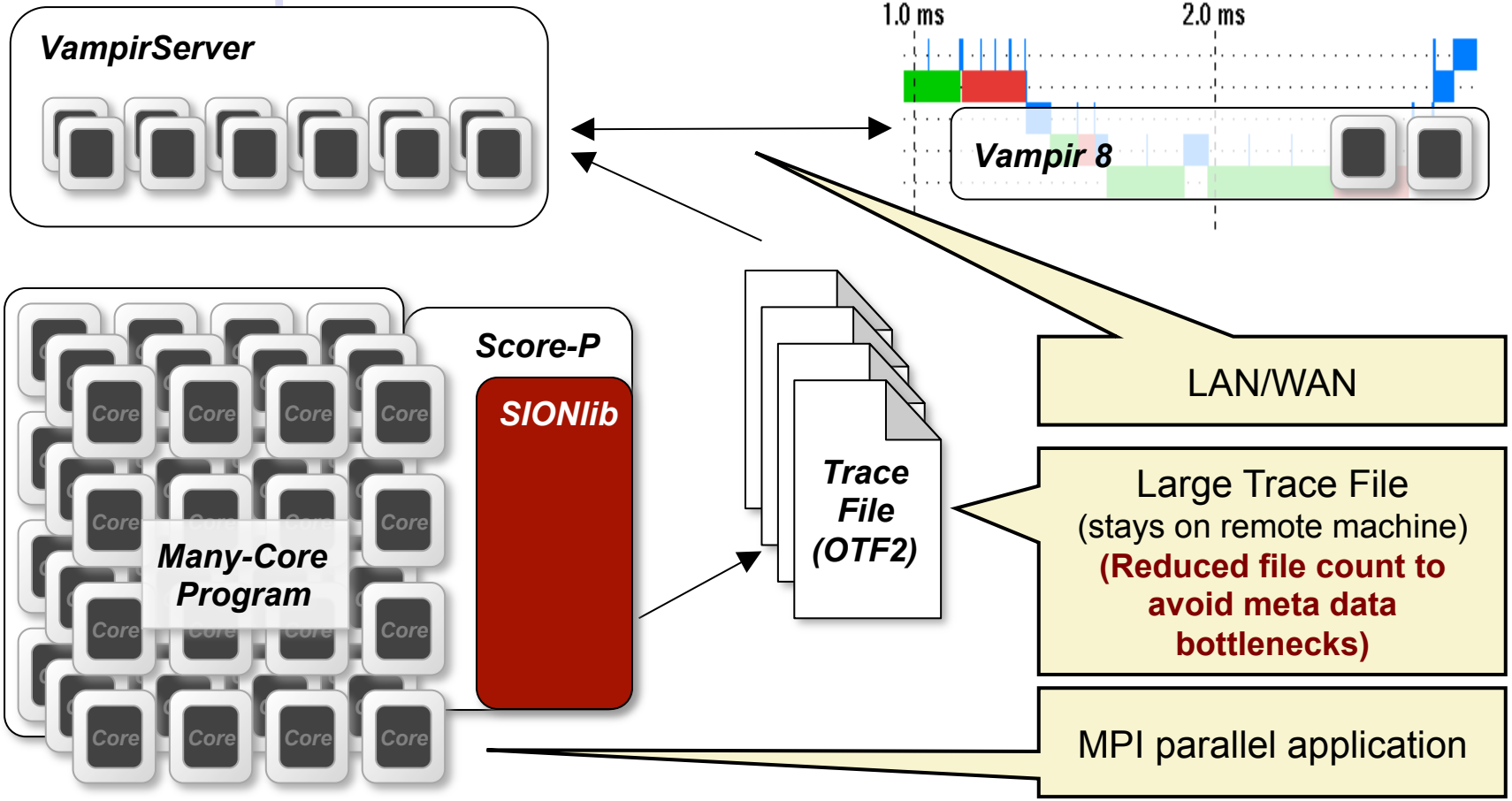


Workflow – Very Large Scale (Score-P 1.3)

- On local machine with remote VampirServer

```
% module load UNITE vampirserver
% vampirserver start -n 12
```

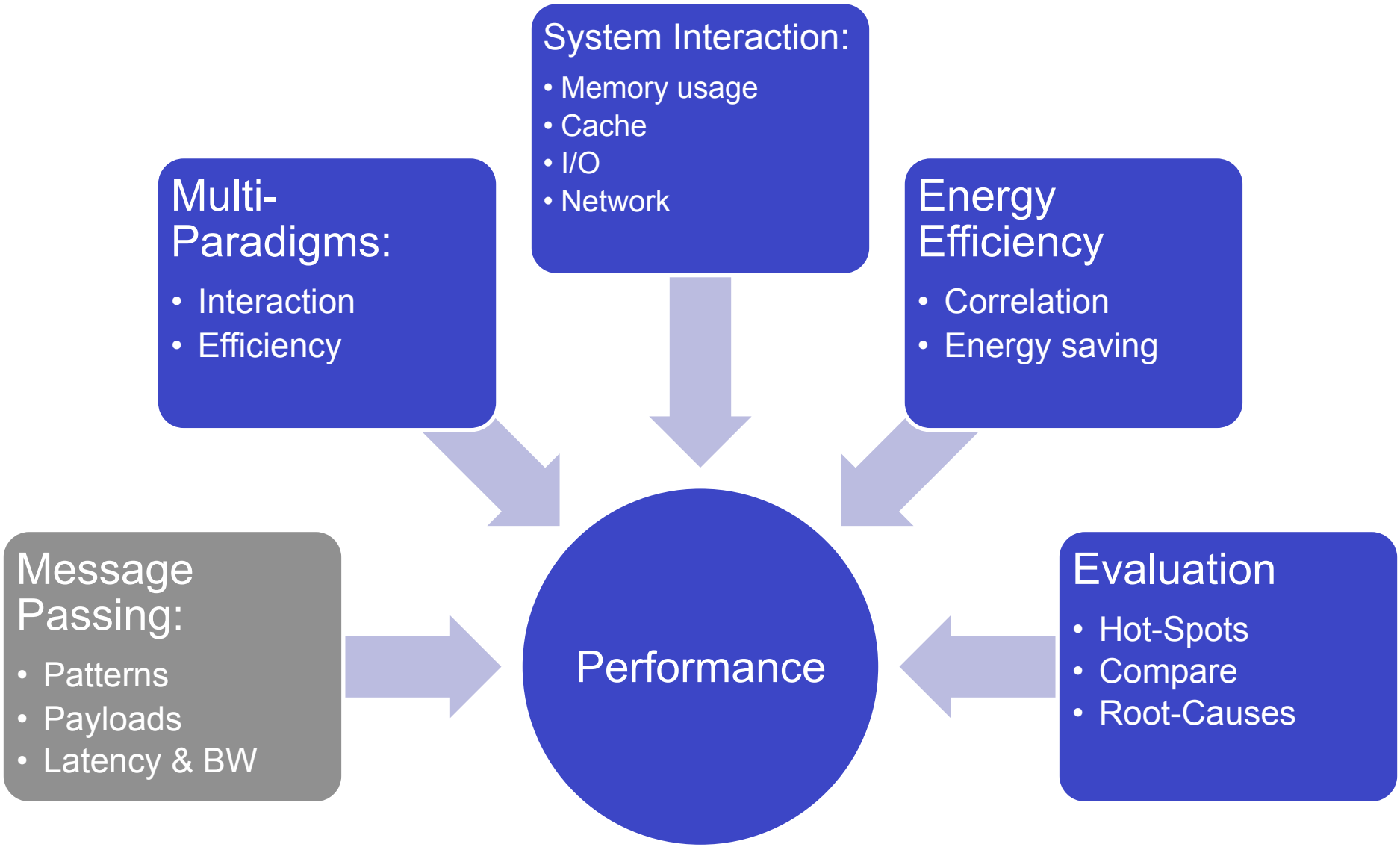
```
% module load UNITE vampir
% vampir
```



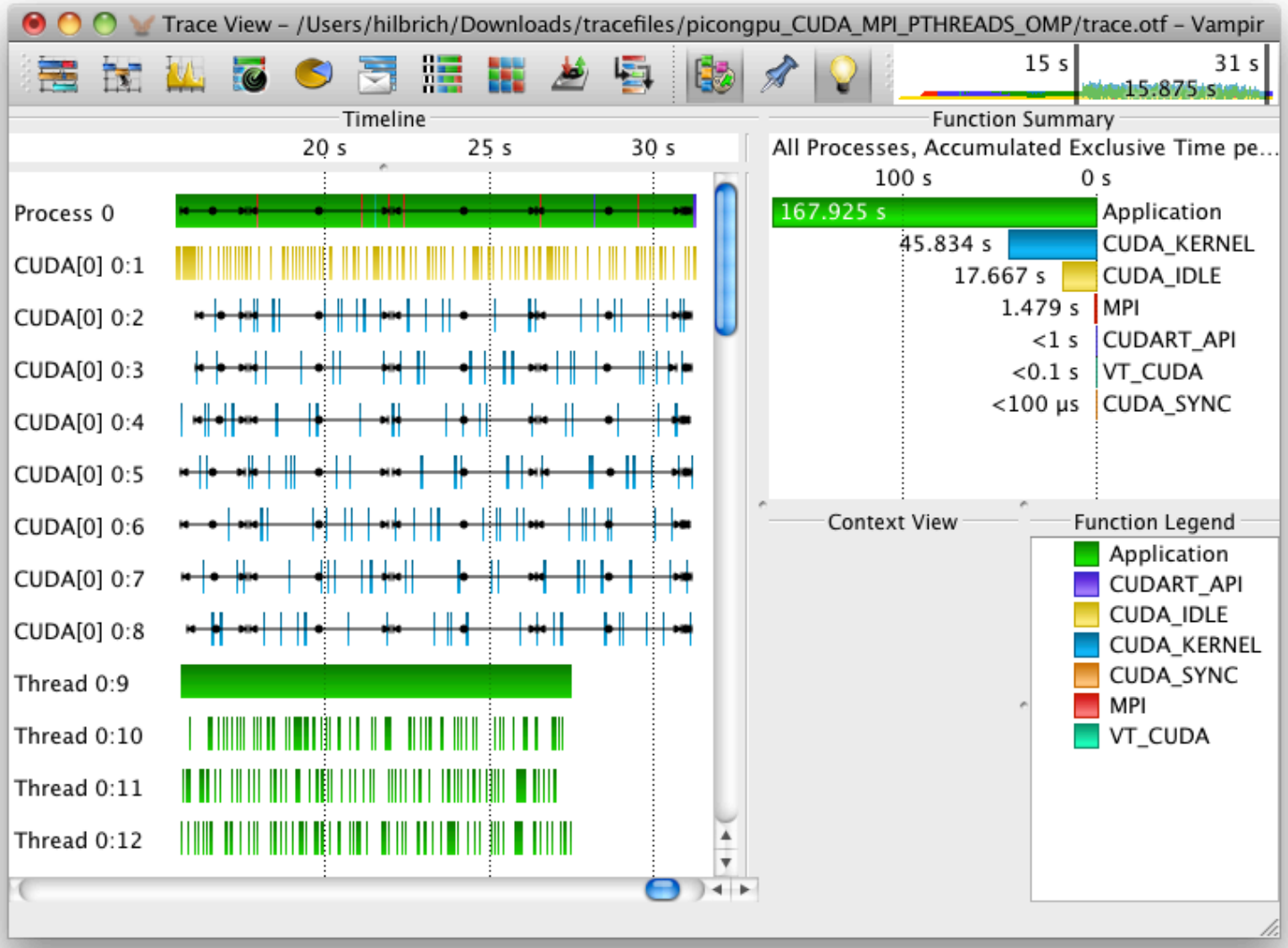
Content

- Motivation
- Workflow
- **Use Cases**
- Tracing
- Visualization
- Advanced

Use Cases


System Interaction:
• Memory usage
• Cache
• I/O
• Network
Multi-Paradigms:
• Interaction
• Efficiency
Energy Efficiency
• Correlation
• Energy saving
Message Passing:
• Patterns
• Payloads
• Latency & BW
Performance
Evaluation
• Hot-Spots
• Compare
• Root-Causes

Use Cases – Multi-Paradigm Support



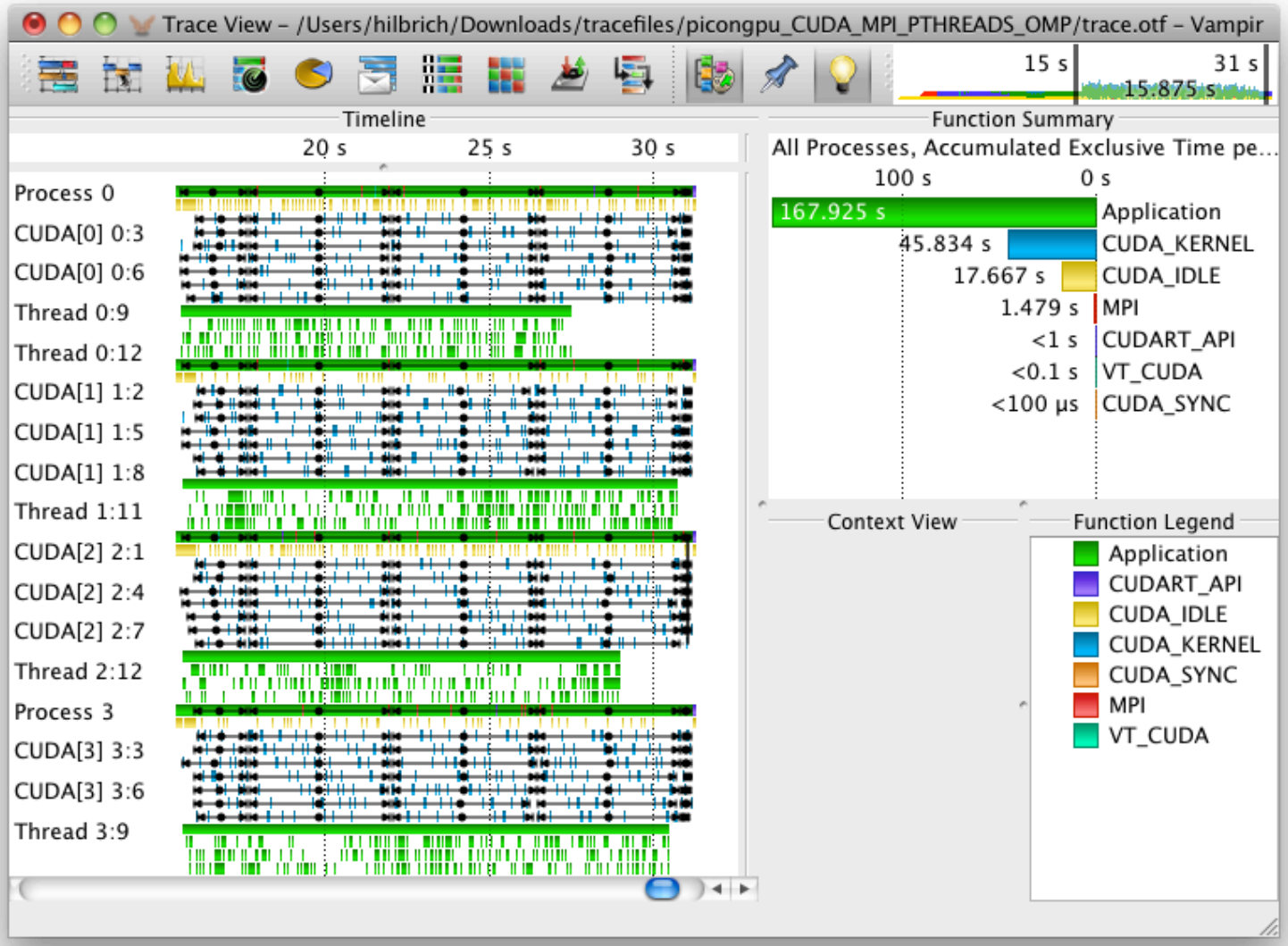
Paradigms:

- MPI
- Pthreads
- CUDA

View:

- Initial

Use Cases – Multi-Paradigm Support



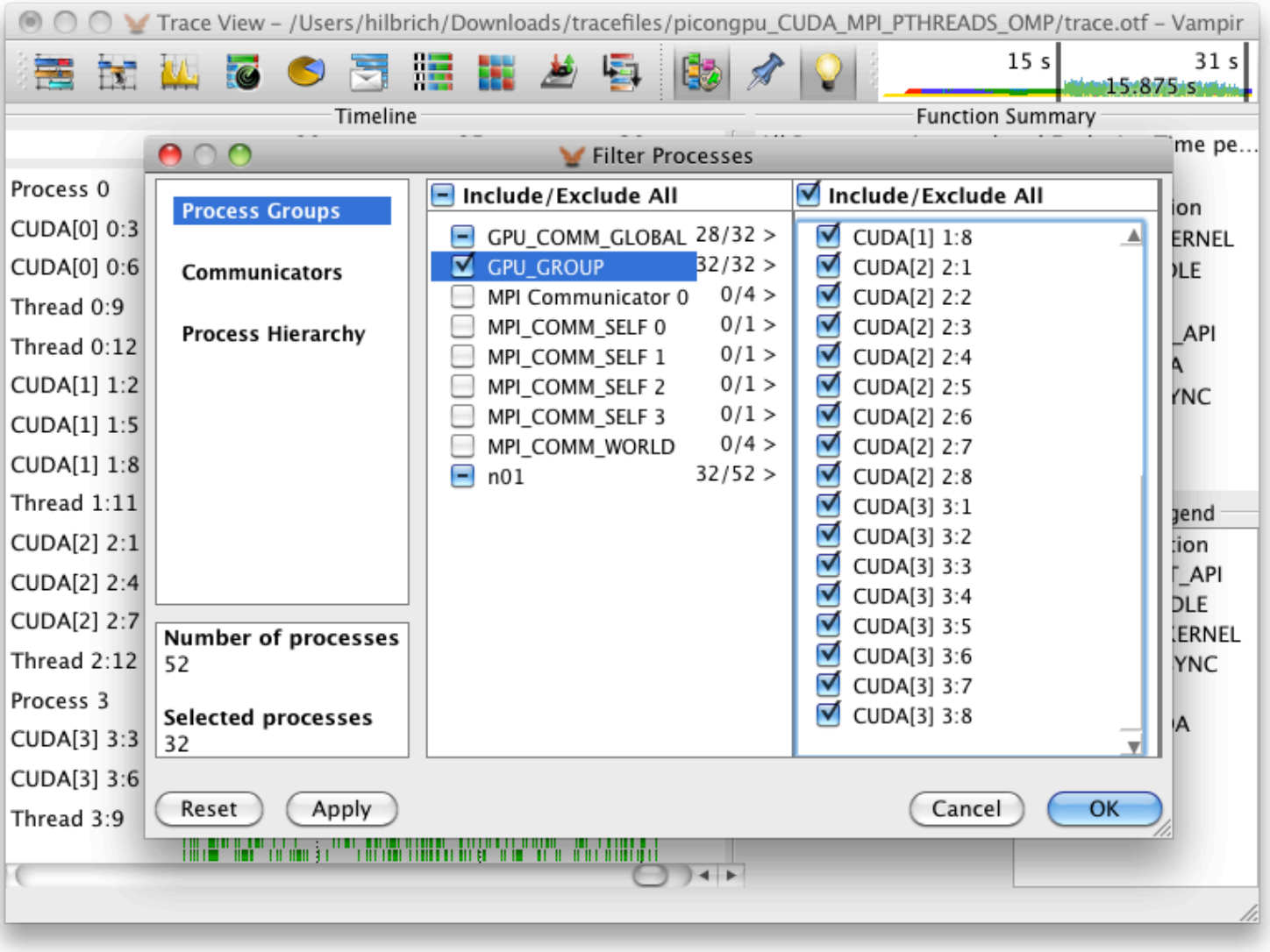
Paradigms:

- MPI
- Pthreads
- CUDA

View:

- All

Use Cases – Multi-Paradigm Support



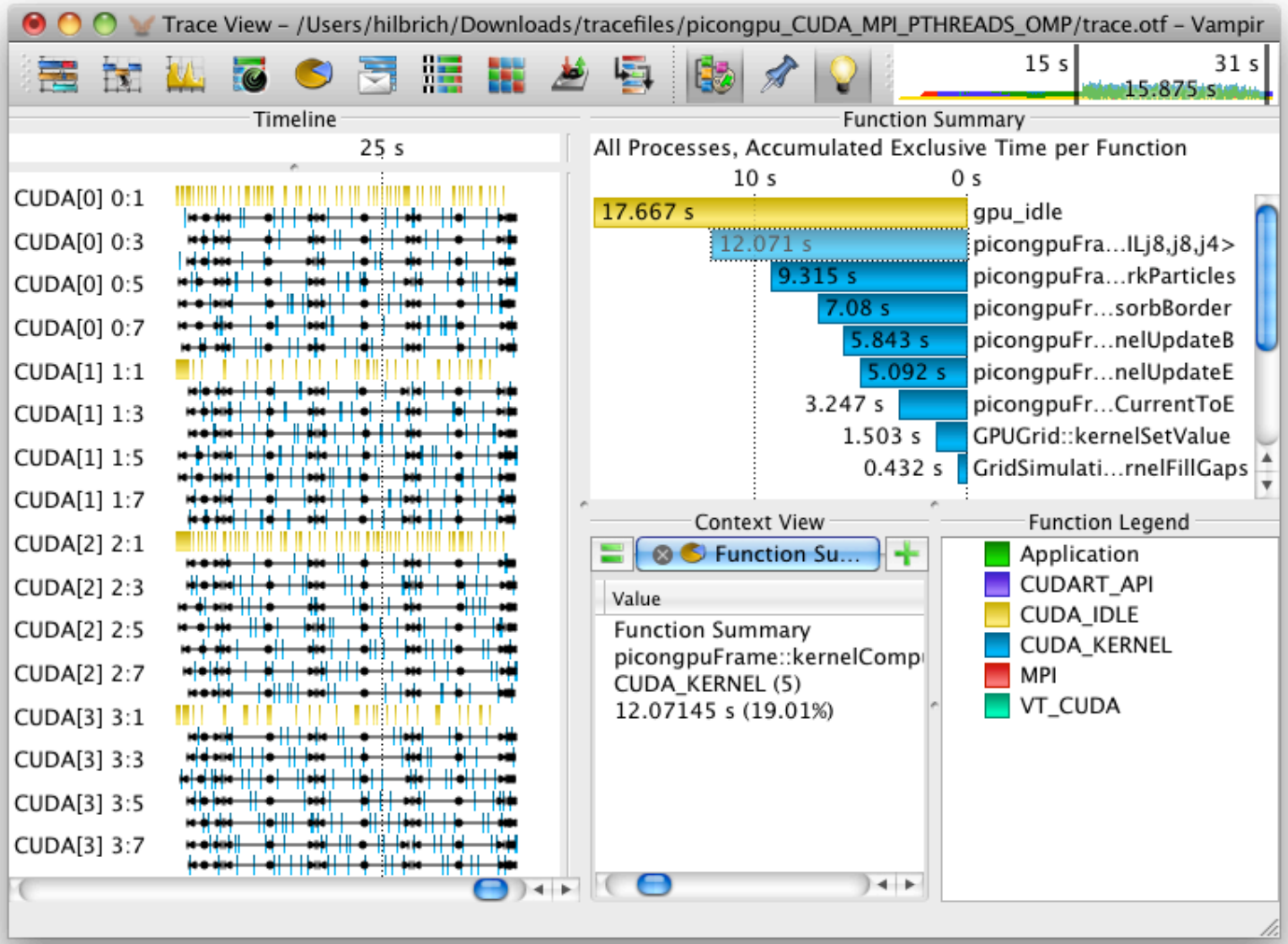
Paradigms:

- MPI
- Pthreads
- CUDA

View:

- Filter

Use Cases – Multi-Paradigm Support



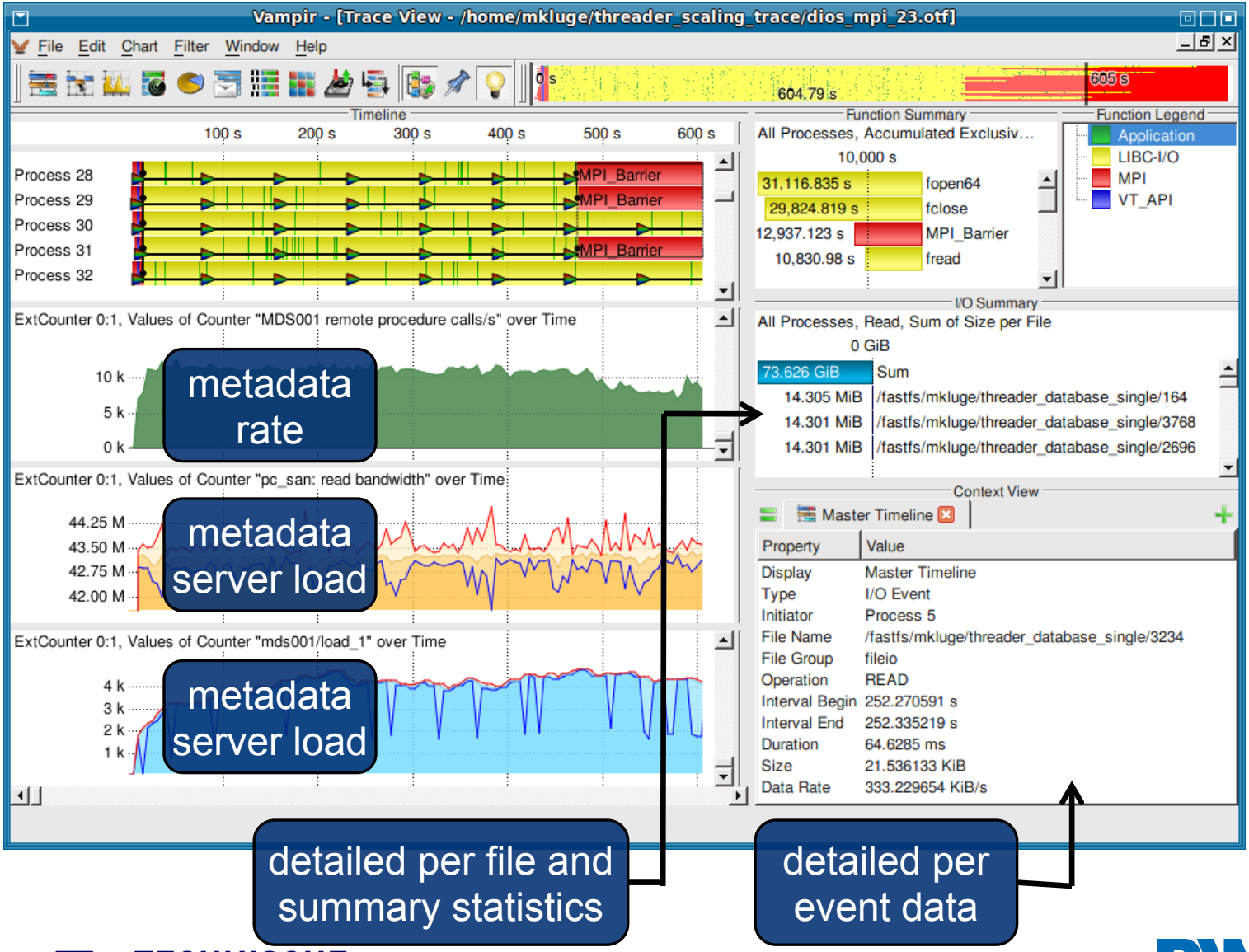
Paradigms:

- MPI
- Pthreads
- CUDA

View:

- Just CUDA

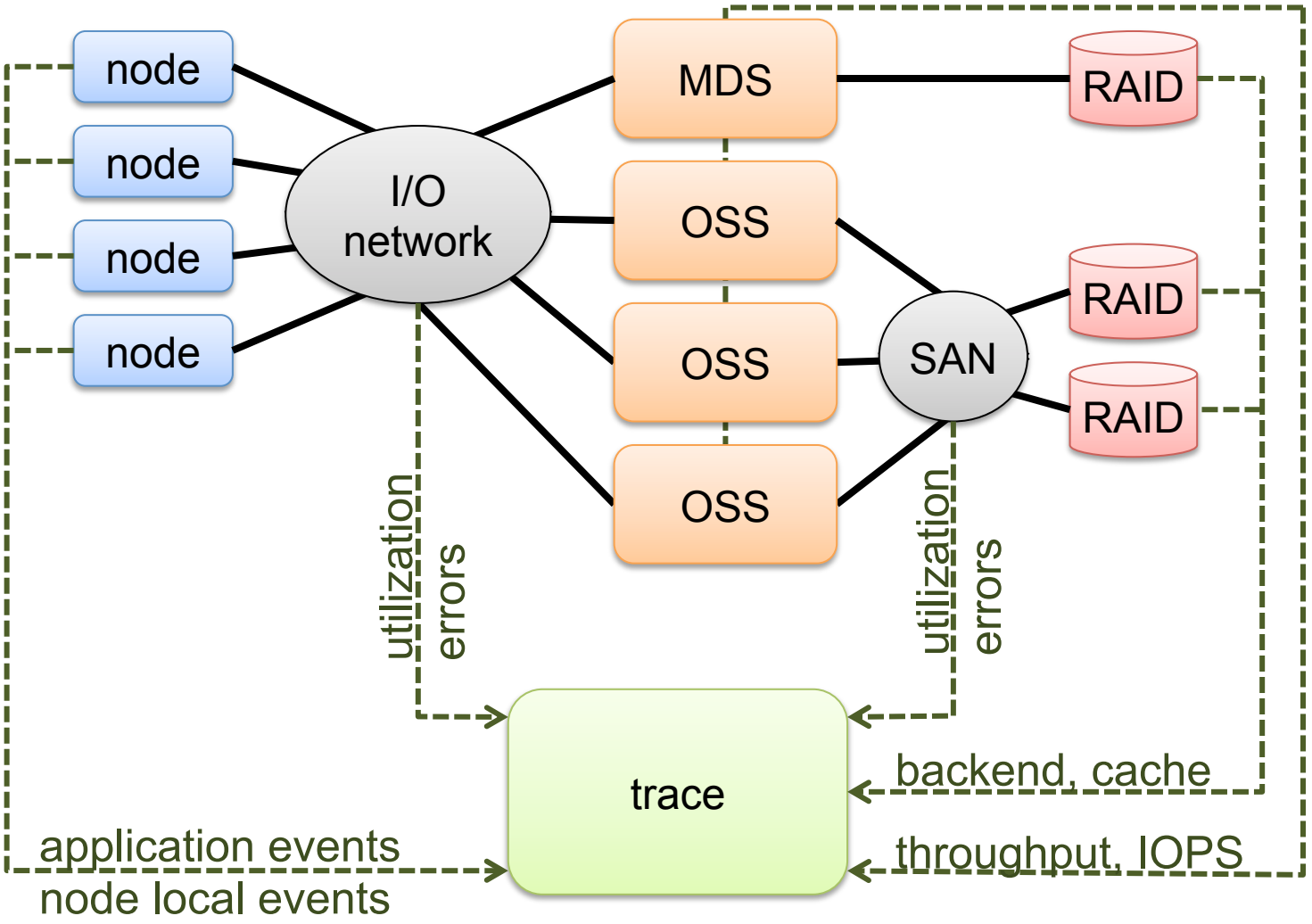
Use Cases – System Interaction



Interaction:

- MPI
- I/O calls
- I/O system

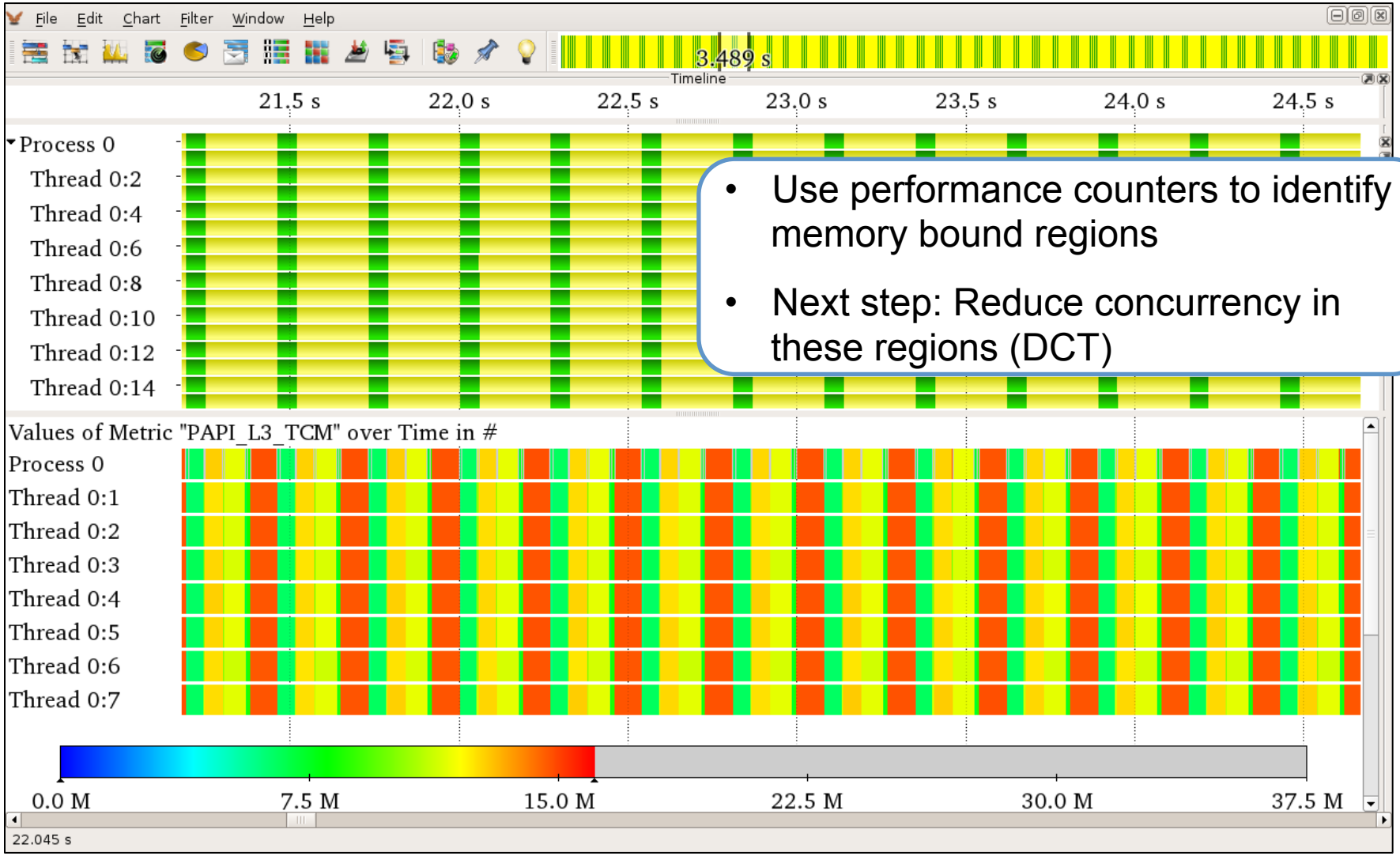
Use Cases – System Interaction (2)



Interaction:

- MPI
- I/O calls
- I/O system

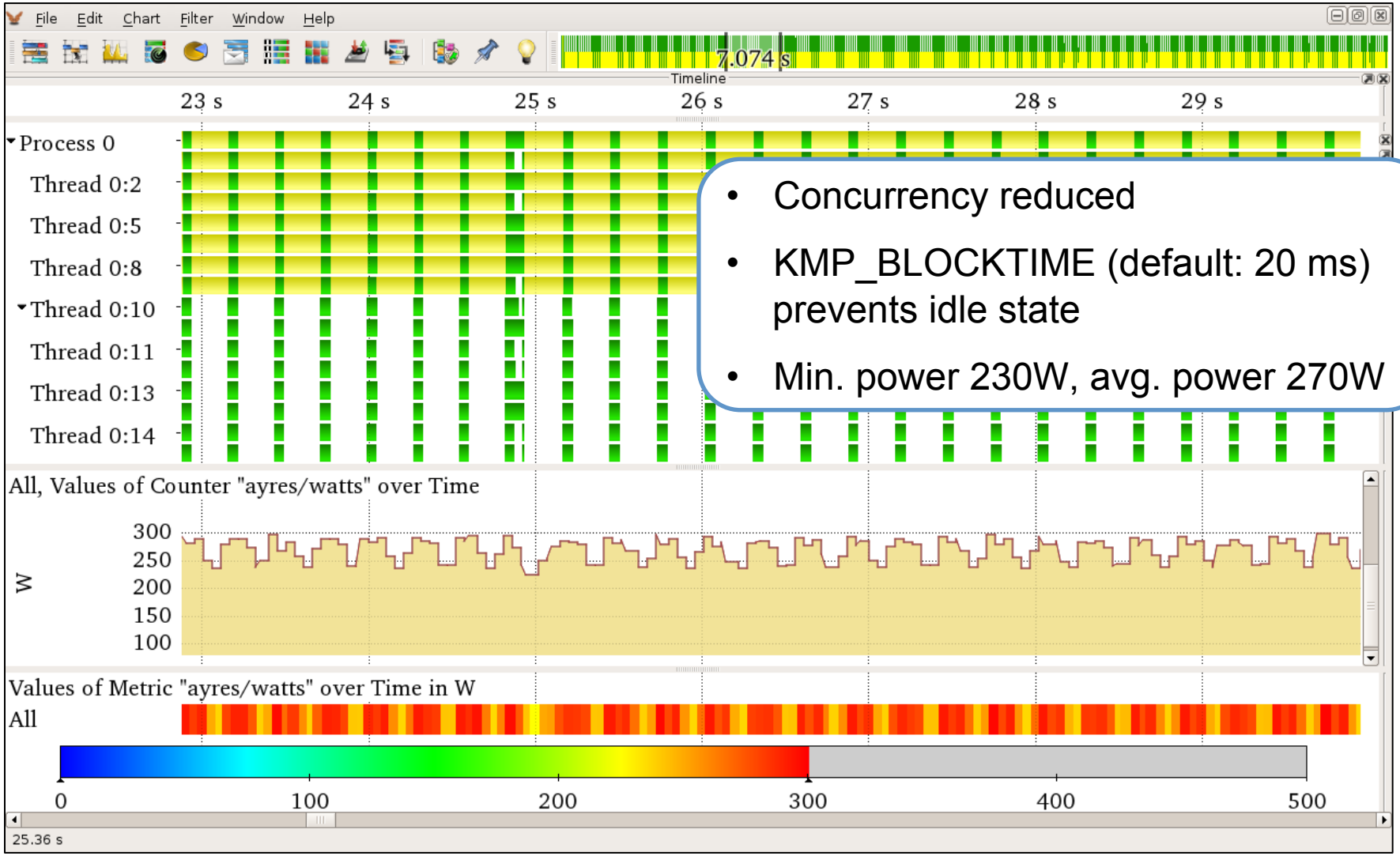
Use Cases – Energy Efficiency



- Use performance counters to identify memory bound regions
- Next step: Reduce concurrency in these regions (DCT)

NAS Parallel Benchmark sp.C on Intel Xeon E5-2670

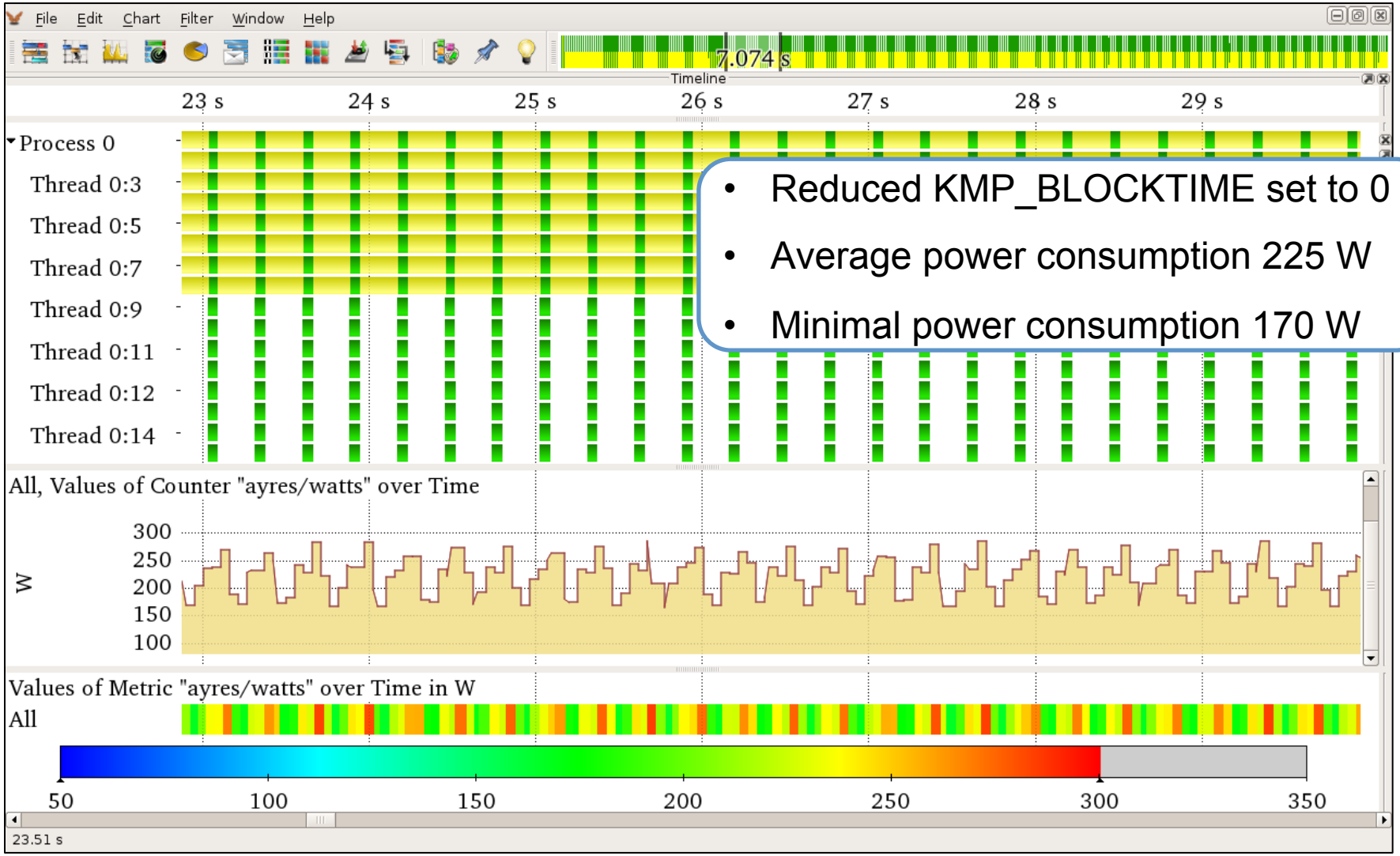
Use Cases – Energy Efficiency (2)



- Concurrency reduced
- KMP_BLOCKTIME (default: 20 ms) prevents idle state
- Min. power 230W, avg. power 270W

NAS Parallel Benchmark sp.C on Intel Xeon E5-2670

Use Cases – Energy Efficiency (3)



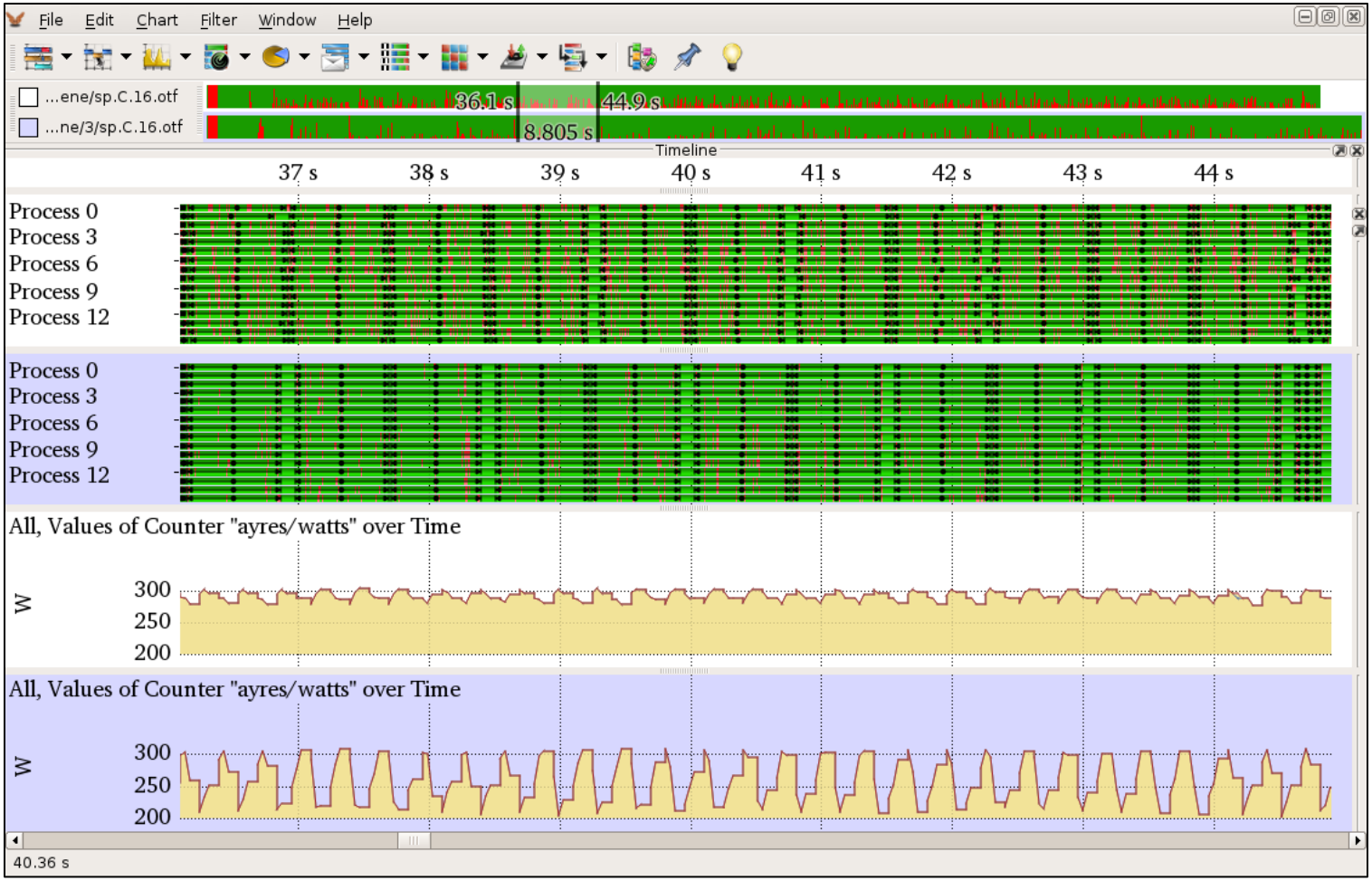
- Reduced KMP_BLOCKTIME set to 0
- Average power consumption 225 W
- Minimal power consumption 170 W

NAS Parallel Benchmark sp.C on Intel Xeon E5-2670

Use Cases – Evaluation: libadapt

- Library that can handle region enters and exits and change system properties
- Easy to use
- Energy efficiency settings
- DVFS: Dynamic Voltage and Frequency Scaling
- DCT: Dynamic Concurrency Throttling
- Small interface

Use Cases – Evaluation: libadapt, Example



Trace Comparison:

- NPB SP
- MPI Version
- Reference vs. DVFS with libadapt

Content

- Motivation
- Workflow
- Use Cases
- **Tracing**
- Visualization
- Advanced

Tracing – Monitors

- Score-P supports multiple performance analysis tools

Measurement: **Score-P**

Partners:

Technische Universität München

gns

JÜLICH FORSCHUNGSZENTRUM

TECHNISCHE UNIVERSITÄT DRESDEN

RWTH AACHEN UNIVERSITY

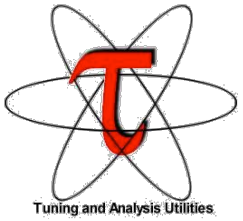
German Research School for Simulation Sciences

UNIVERSITY OF OREGON

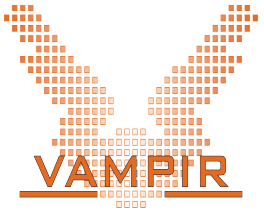
Measurement:
VampirTrace
(Legacy)
Just for Vampir

Analysis:

- TAU



- **Vampir**



- Scalasca

Tracing – Monitors

- Score-P supports multiple performance analysis tools

Measurement: **Score-P**

Partners:

Technische Universität München

gns

JÜLICH FORSCHUNGSZENTRUM

TECHNISCHE UNIVERSITÄT DRESDEN

RWTH AACHEN UNIVERSITY

German Research School for Simulation Sciences

UNIVERSITY OF OREGON

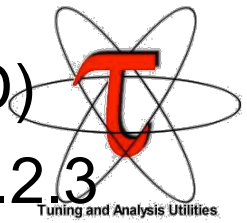
– Focus today

– OpenSource (BSD)

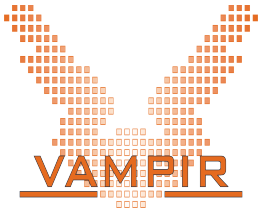
– Current version: 1.2.3

Analysis:

- TAU



- **Vampir**



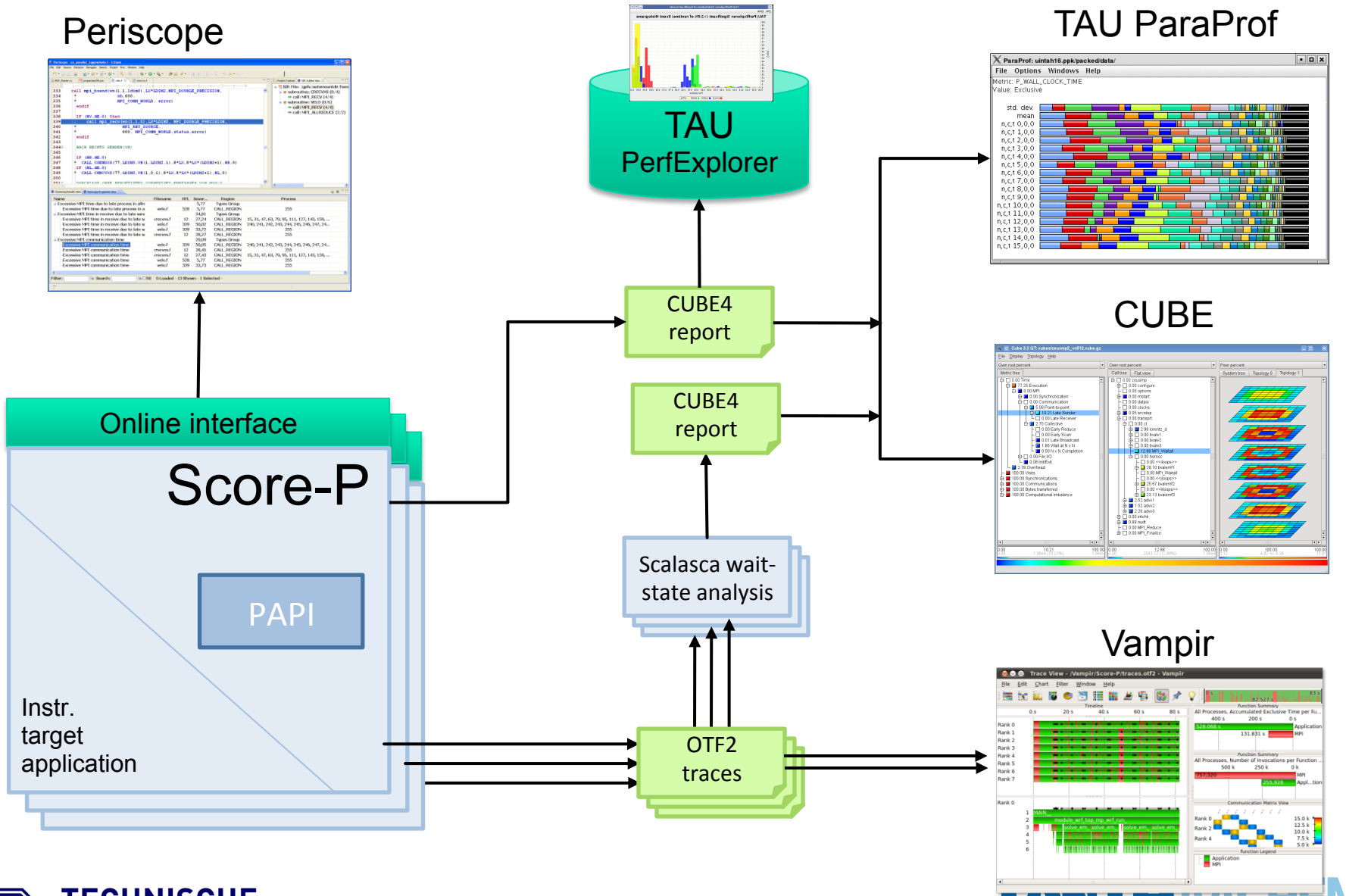
- Scalasca

Measurement:
VampirTrace
(Legacy)
Just for Vampir

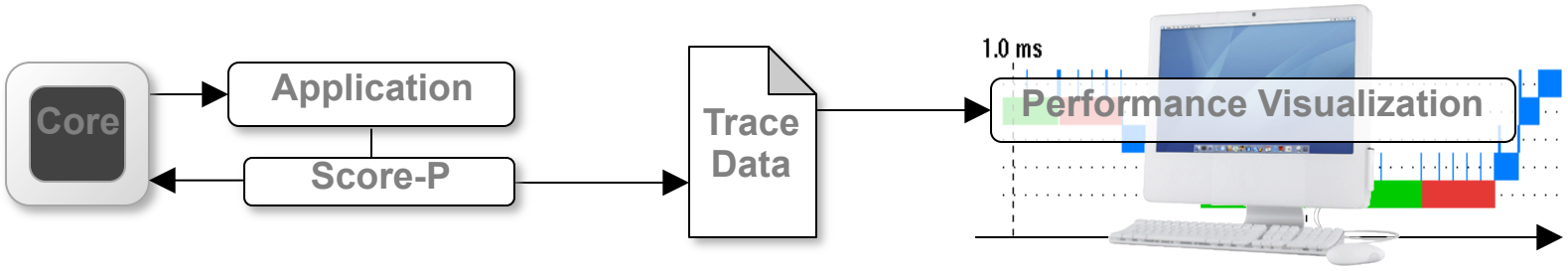
– Backup for outstanding features

– Only bug fix releases

Tracing – Monitors, Score-P Ecosystem

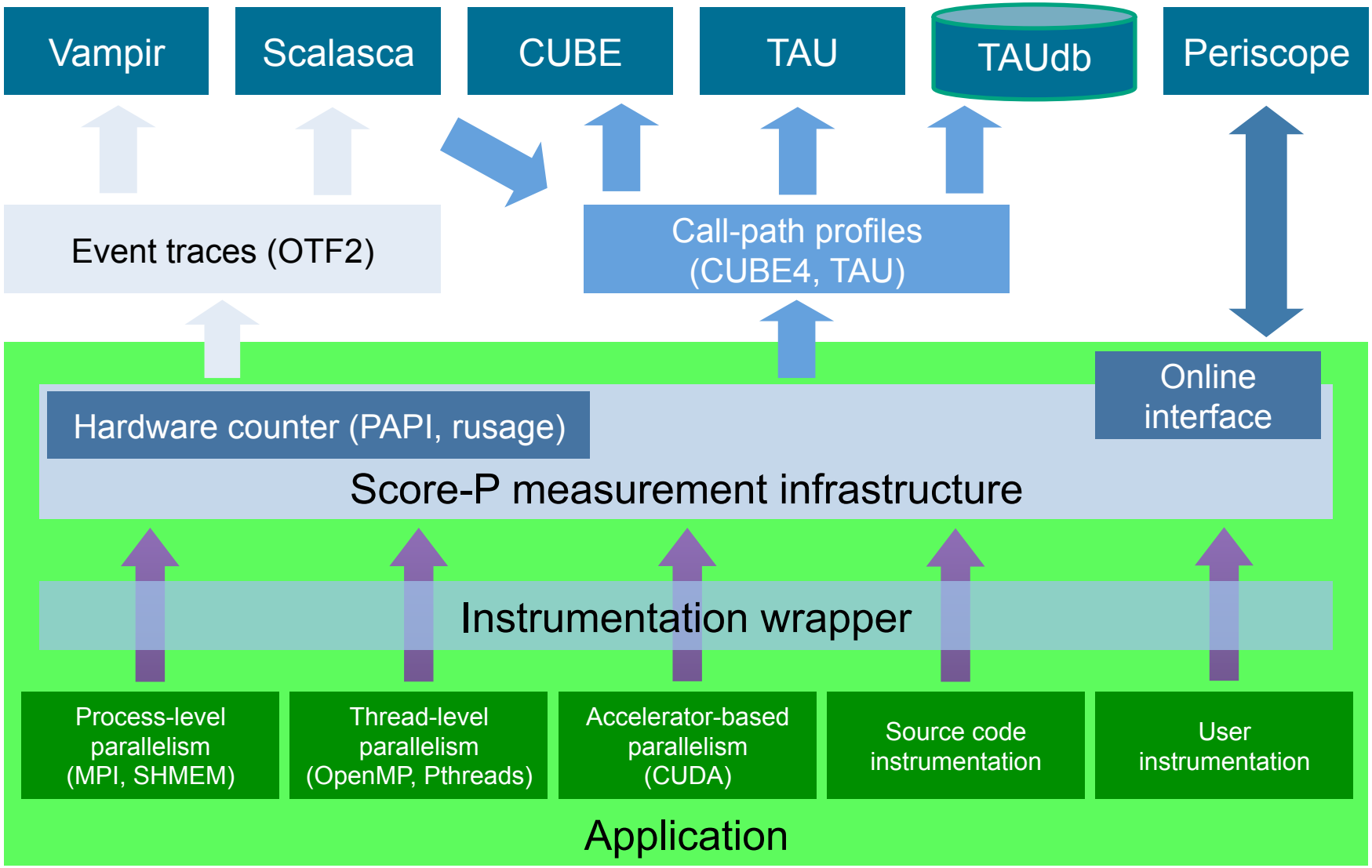


Tracing – Overview

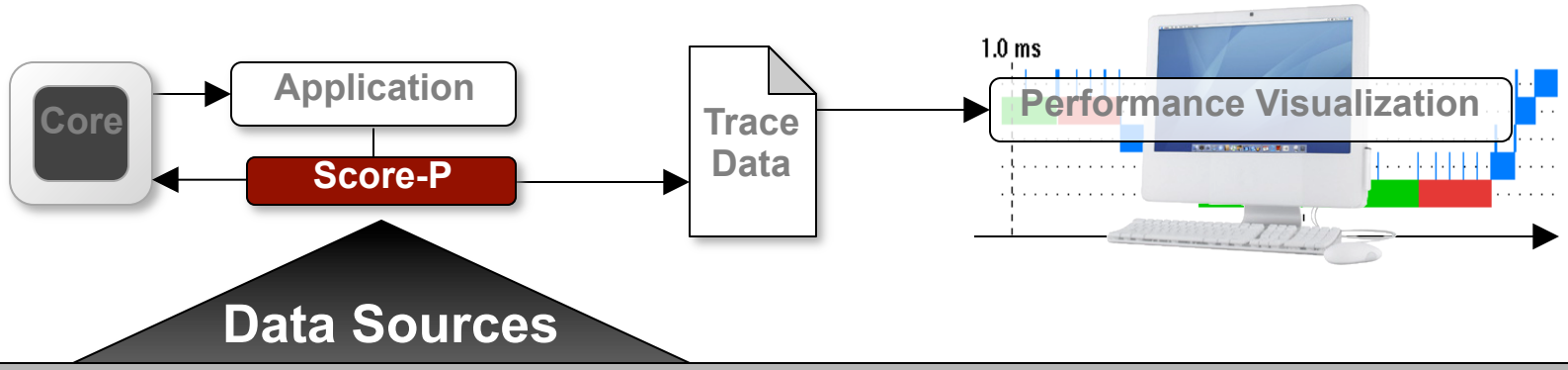


- Workflow:
  - Attach Score-P to application
  - Run with the attached monitor
    - ⇒ Result: **trace**/profile data
  - Analyze the trace with Vampir
- Repeat to:
  - Adapt instrumentation (“what you measure”)
  - Evaluate result of a change

Tracing – Score-P Architecture



Tracing – Instrumentation



Data Sources

Source level
- Compiler
- Manual
- OpenMP with Opari2
- Tau PDT

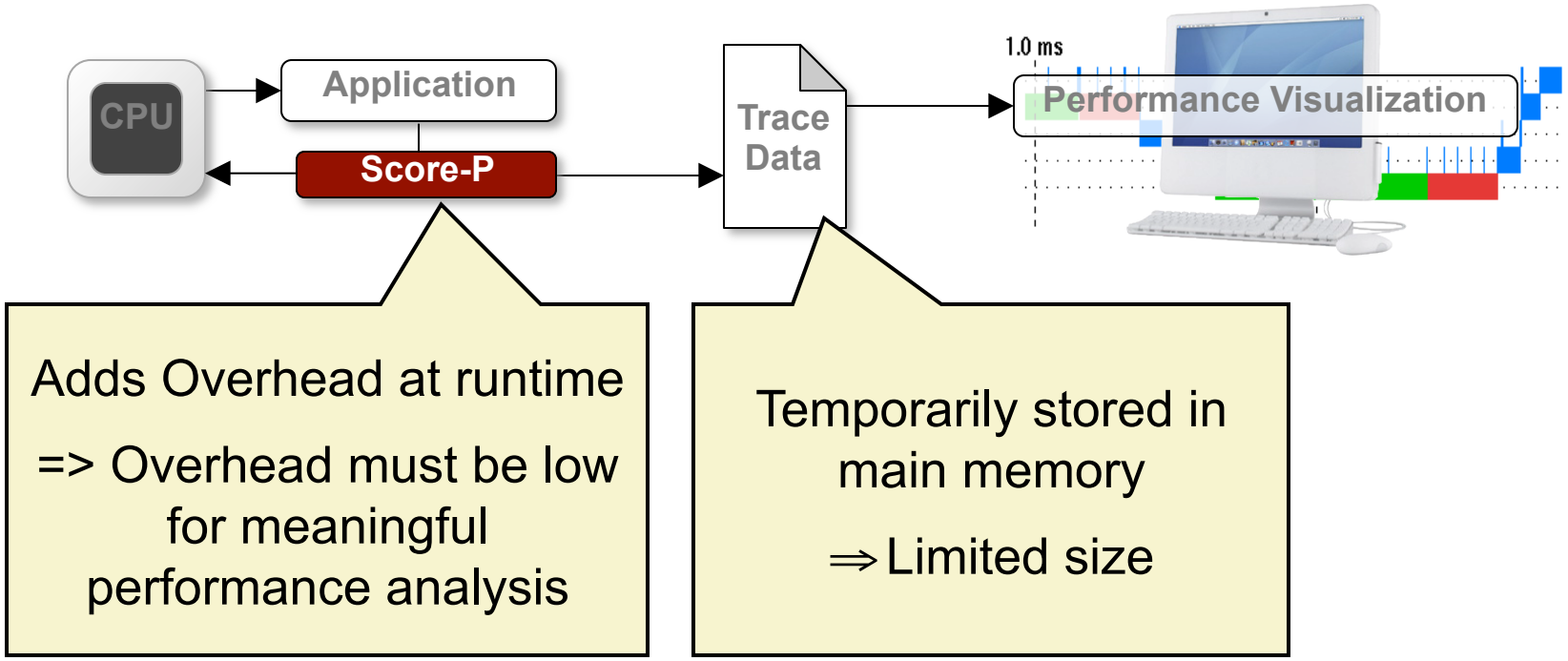
MPI Profiling Interface
- MPI

Runtime/Library
- Pthreads
- NVIDIA CUDA
- OpenSHMEM (Cray-SHMEM)
- External counters
- Plugin counters

Operating System
- Resource usage

Hardware
- Performance counters

Tracing – The Hard Part



- Event tracing requires trade-off's:
  - ⇒ Only add the data sources you need
  - ⇒ Limit granularity (i.e., filtering)
- Thus: Score-P default is a profiling experiment

Tracing – Usage

- Compiler pre-command provides instrumentation:

```
CC=gcc
CXX=g++
F90=gfortran
MPICC=mpicc
…
```

```
CC=scorep gcc
CXX=scorep g++
F90=scorep gfortran
MPICC=scorep mpicc
…
```

- Basic Score-P usage:
  - Put “scorep” in front of compiler commands
  - Re-compile & re-link
  - Run as usual (details follow)
- User function instrumentation: compiler-based

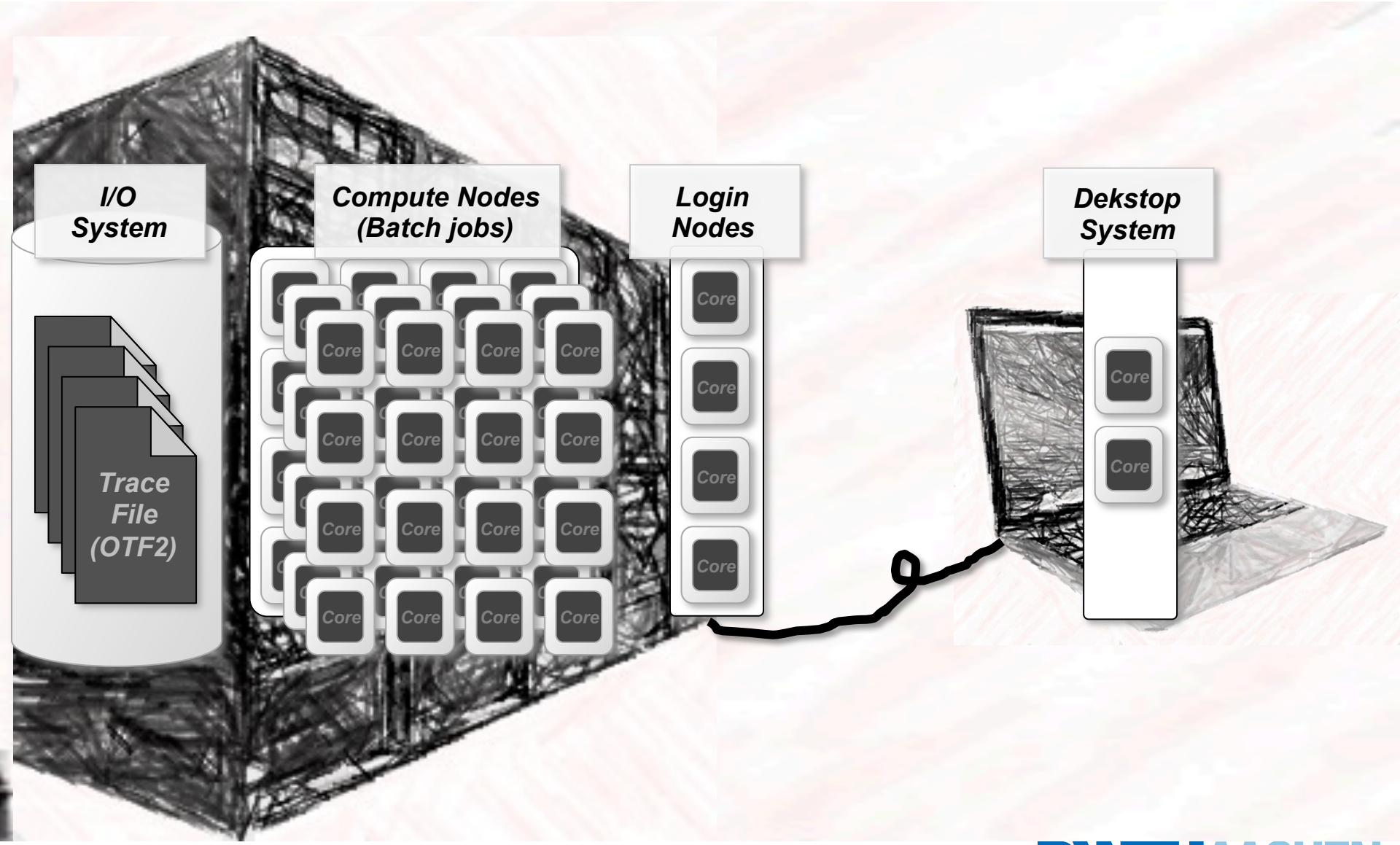
Tracing: Good Practices

- Traces can become large; Their handling challenging
  - Trace size proportional to: number of processes/threads (width), duration (length), and measurement detail (depth)
- Intermediate flushes => High degree of perturbation
  - Either use less detail or larger trace buffers
- Traces should be written to a parallel file systems
  - E.g. “/work”, “/scratch”, or “/p/lscratchc”
- Moving large traces can become challenging
  - However, systems with more memory/core can analyze larger traces
  - Alternatively, undersubscribe for increased memory/core ratios

Content

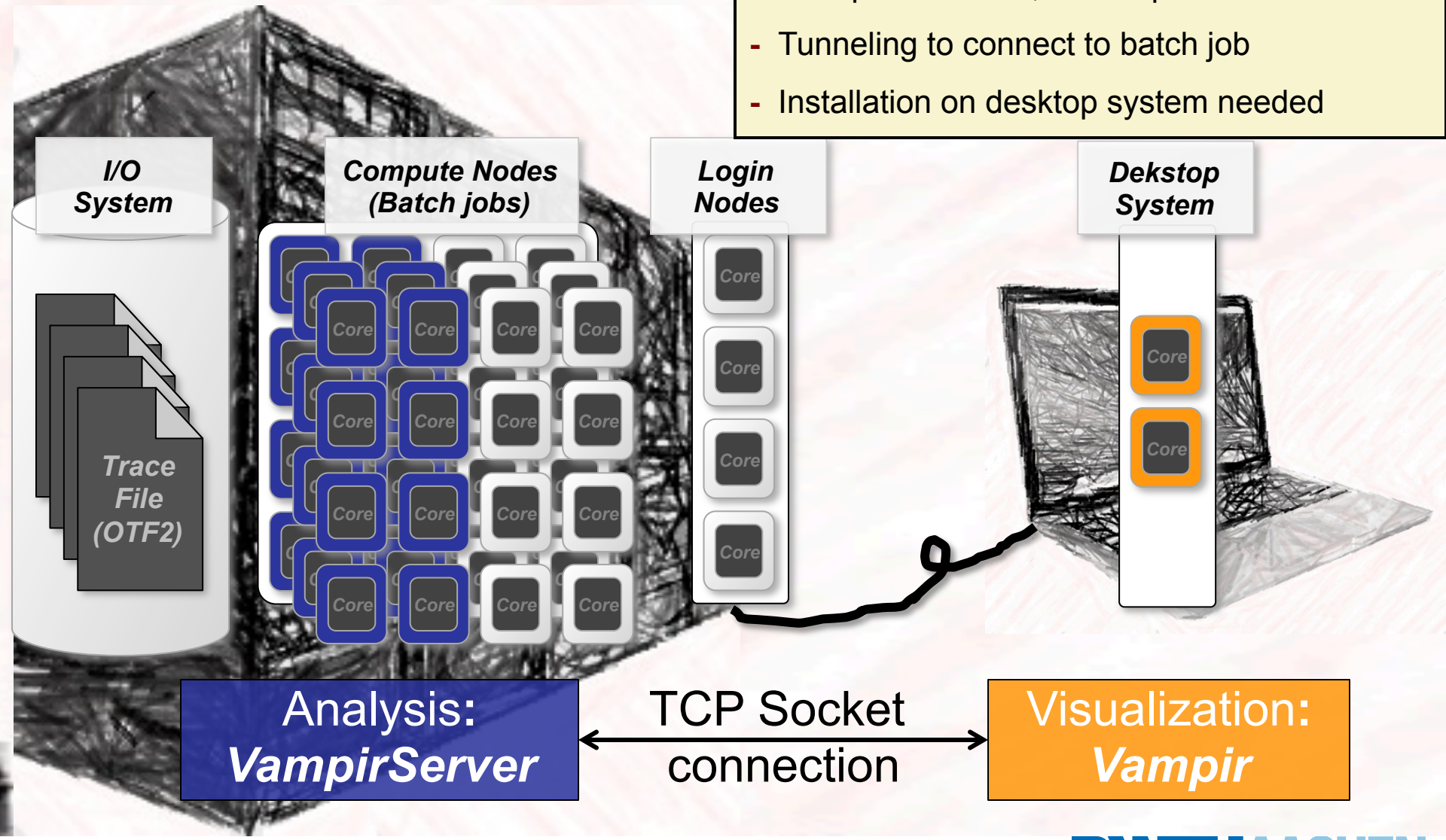
- Motivation
- Workflow
- Use Cases
- Tracing
- **Visualization**
- Advanced

Visualization: After Tracing



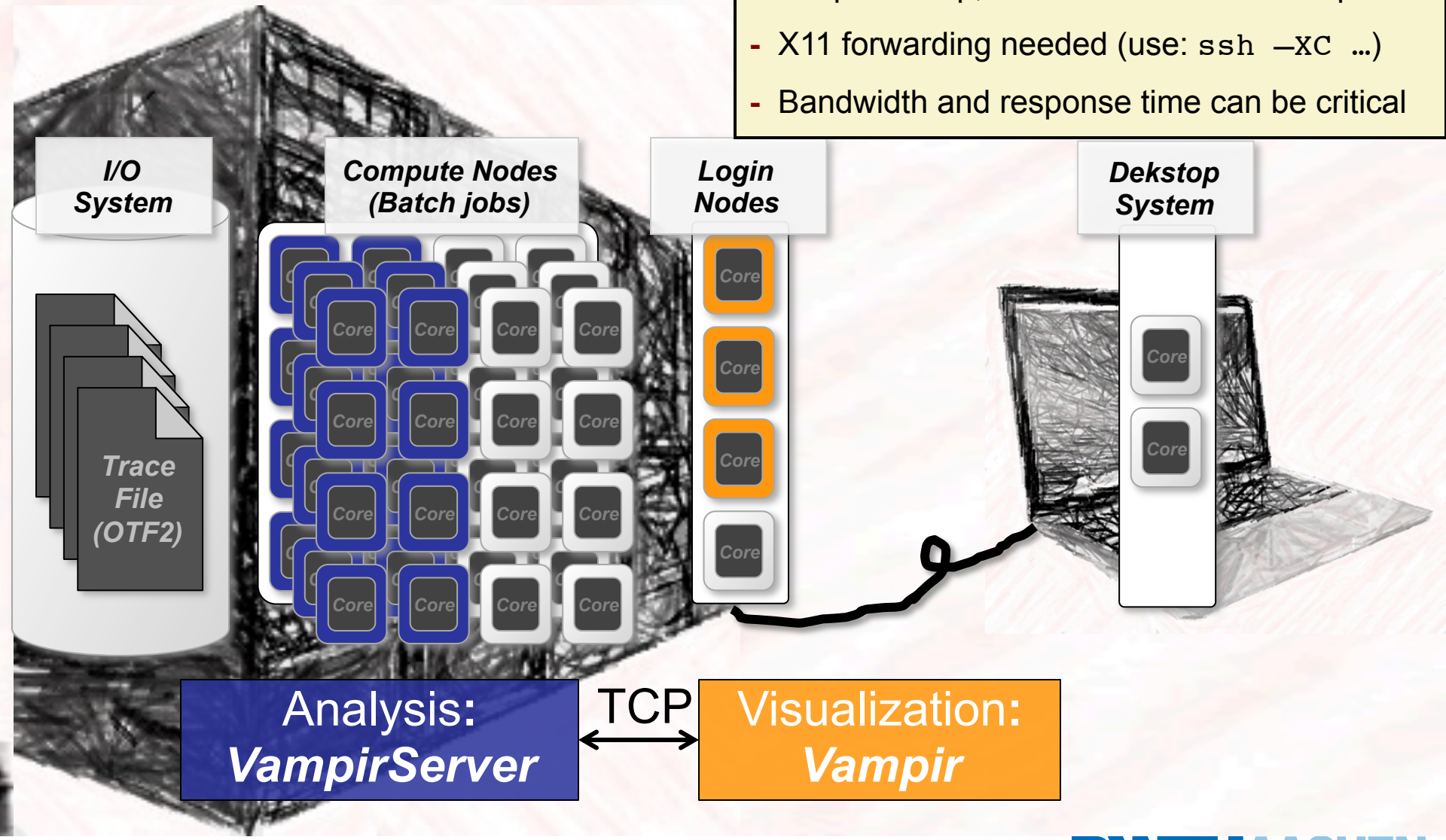
Visualization: Best Option (Analysis on HPC system)

+ Best performance, low response time
- Tunneling to connect to batch job
- Installation on desktop system needed



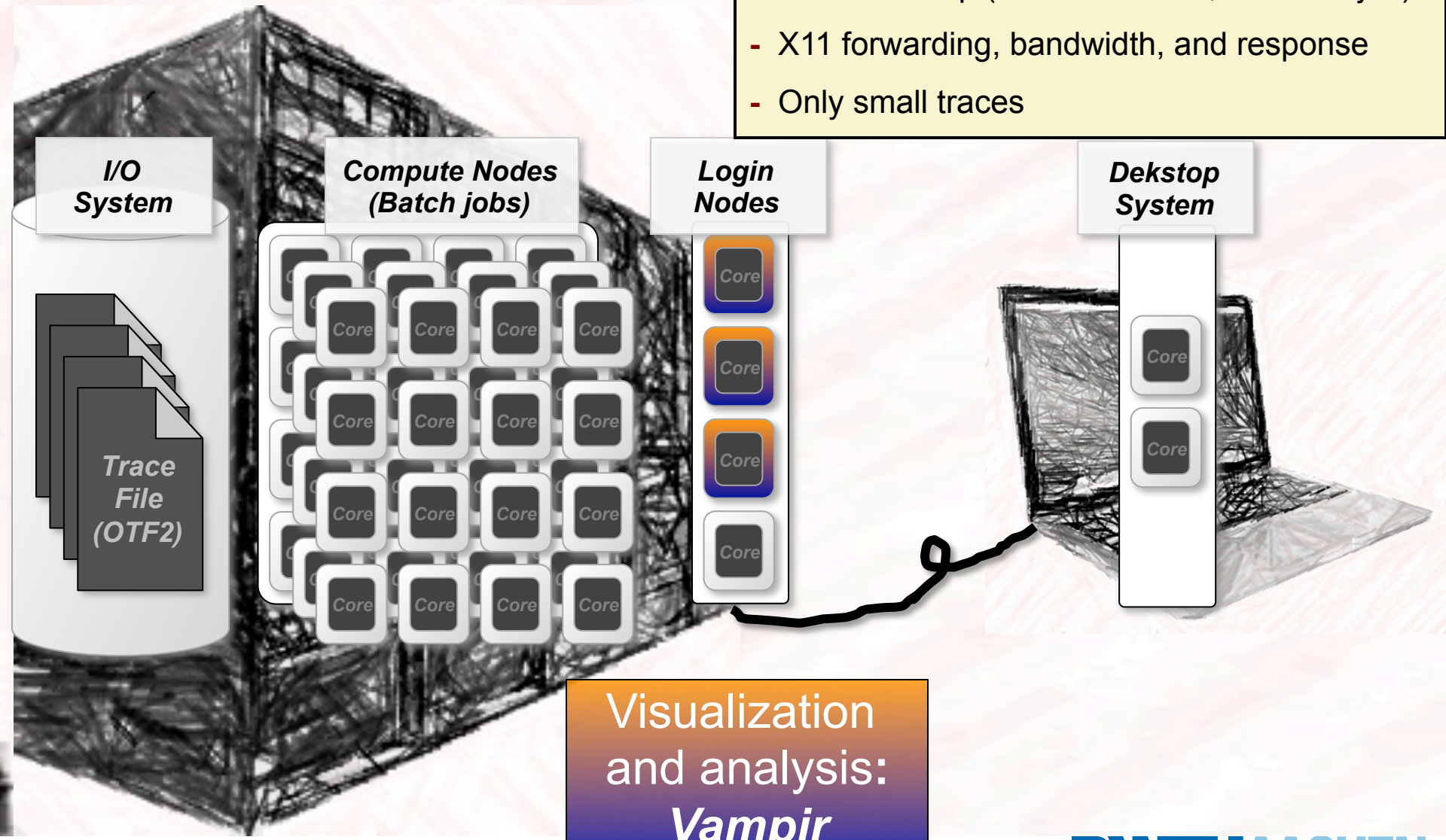
Visualization: Alternative (Analysis on HPC system)

+ Simpler setup, no installation on desktop
- X11 forwarding needed (use: `ssh –XC …`)
- Bandwidth and response time can be critical



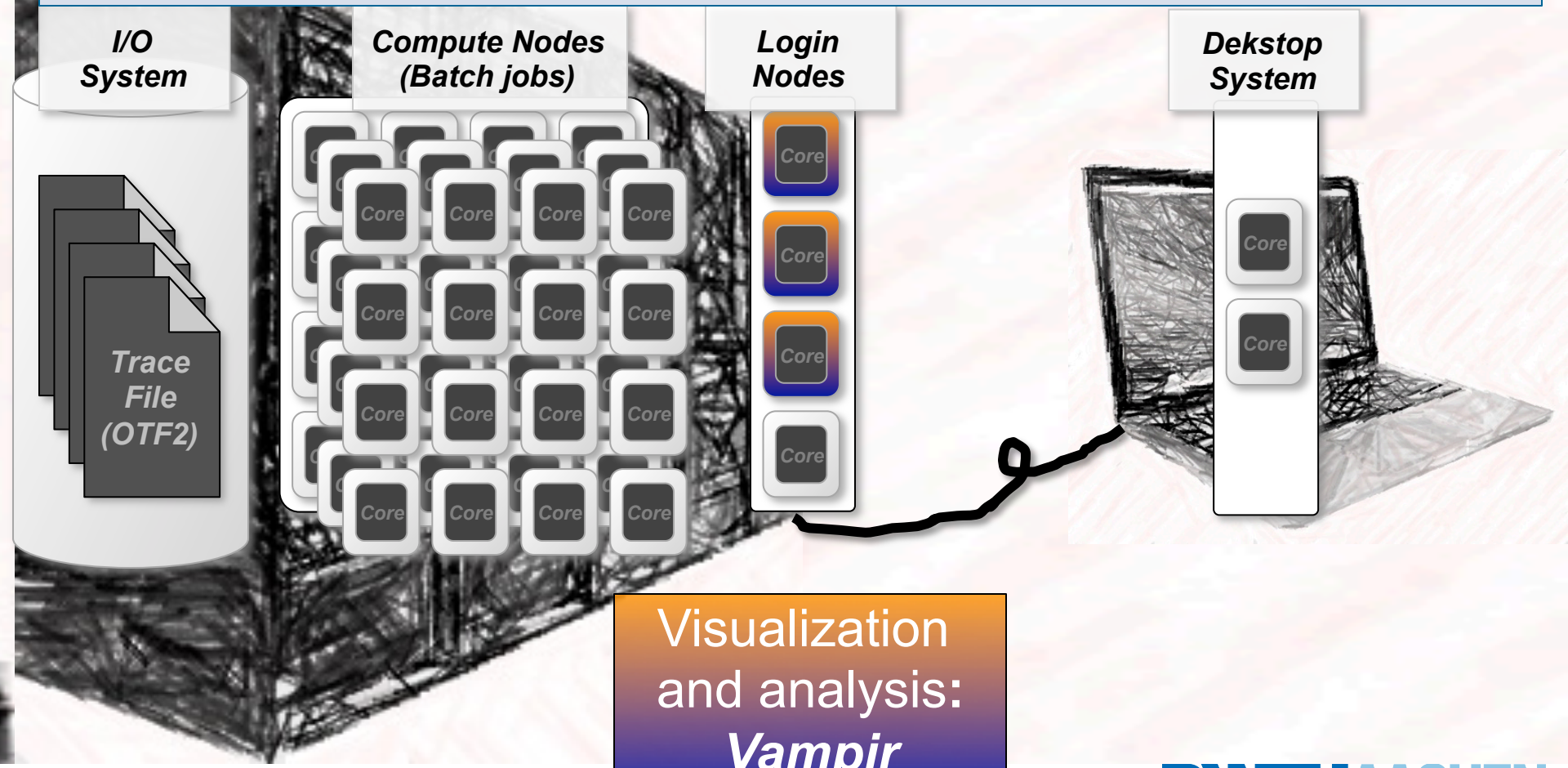
Visualization: Most simple (Analysis on Frontend)

+ Minimal setup (no installations, no batch job)
- X11 forwarding, bandwidth, and response
- Only small traces



Visualization: Most simple (Analysis on Frontend)

```
% ssh -XC <user>@<machine>
% module load vampir
% vampir <scorep-trace-directory>/traces.otf2
```

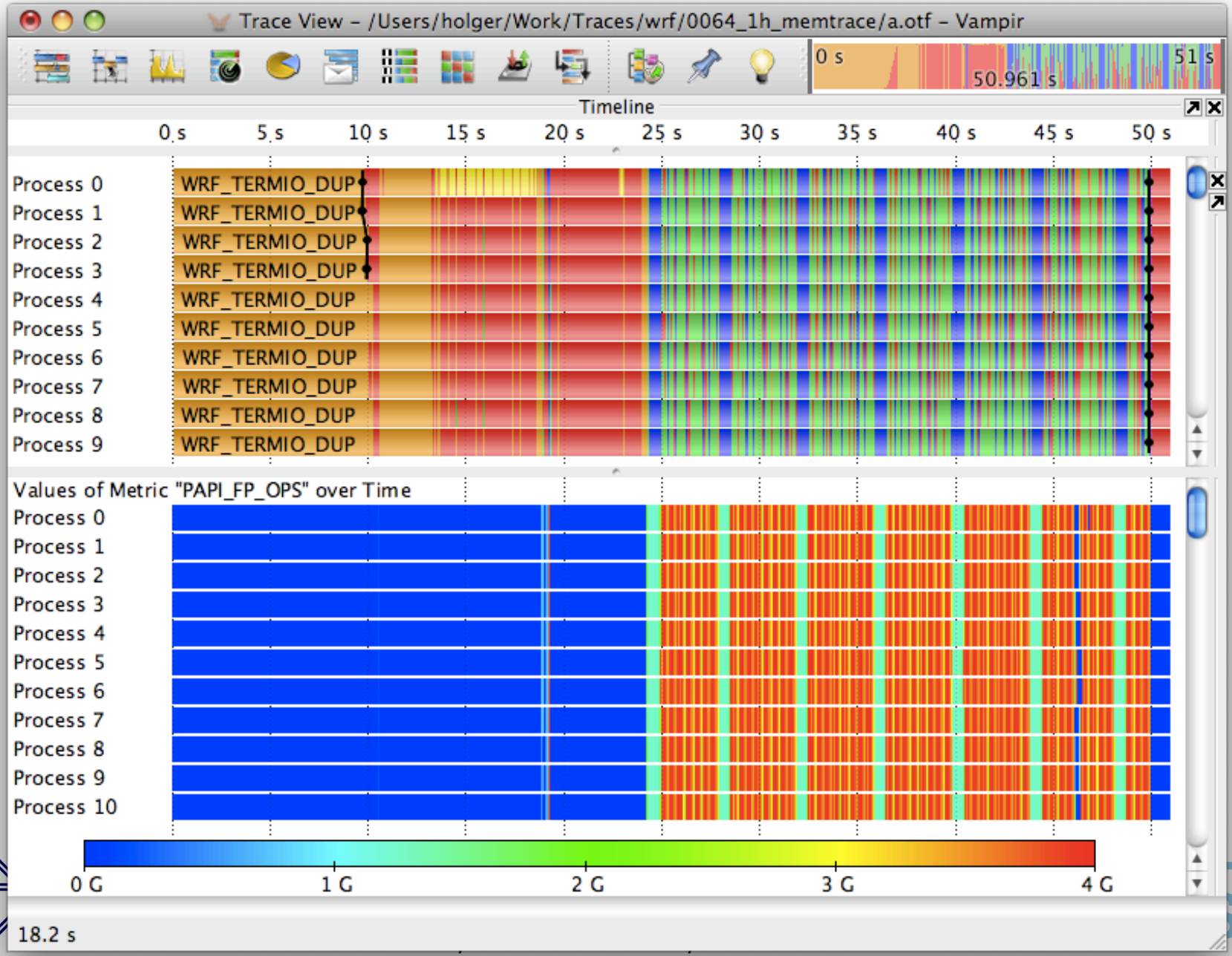


Visualization and analysis: ***Vampir***

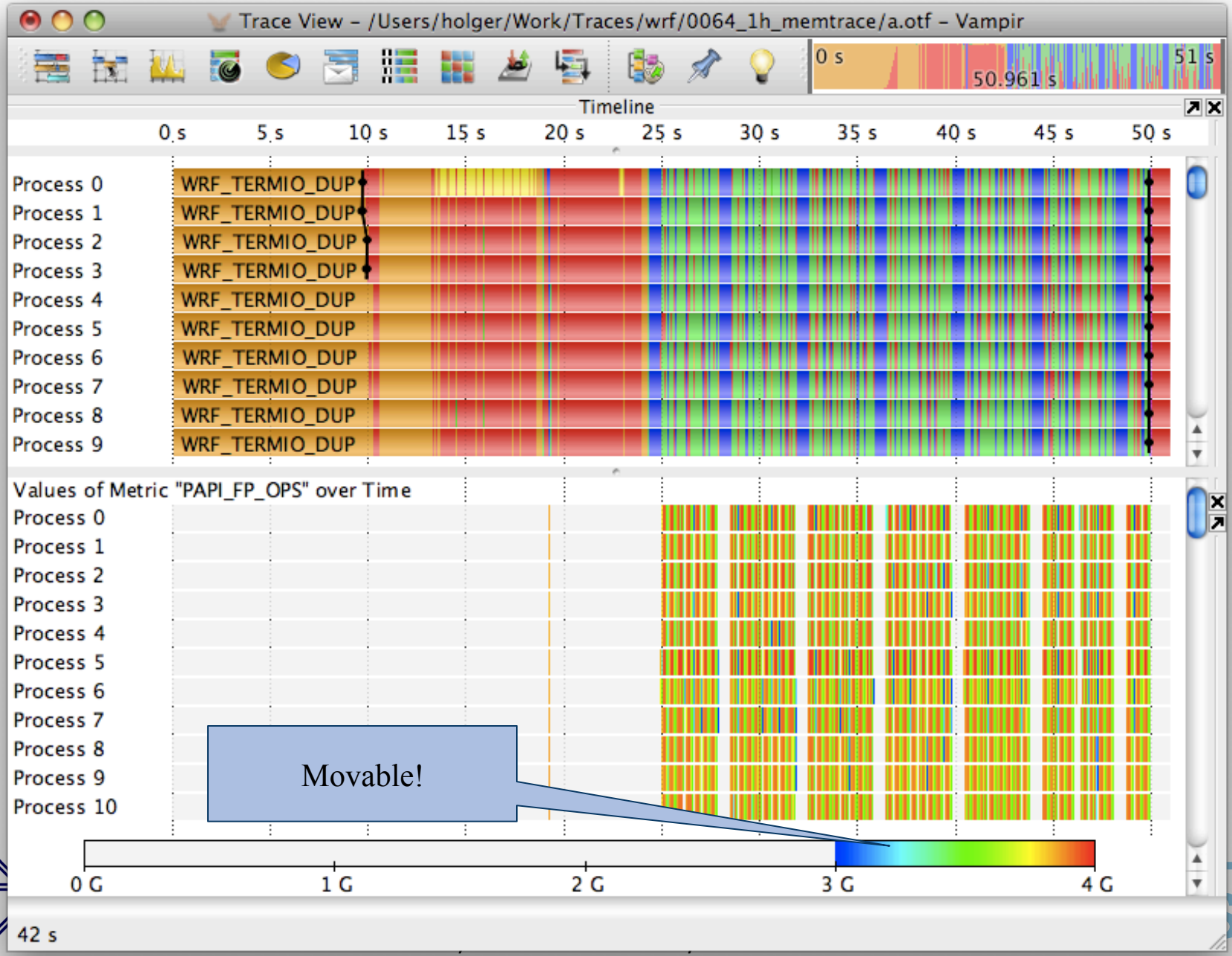
Content

- Motivation
- Workflow
- Use Cases
- Tracing
- Visualization
- **Advanced**

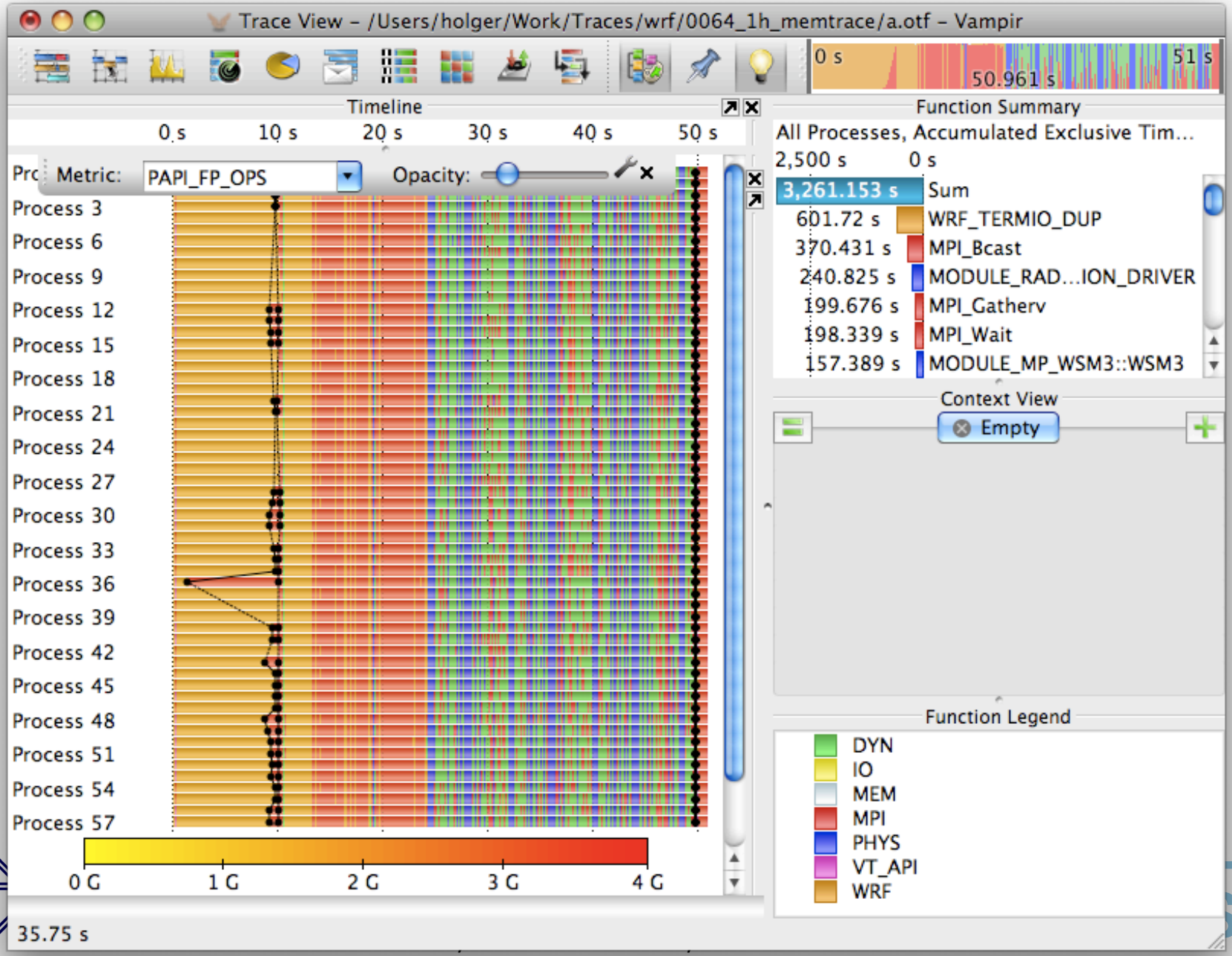
Advanced – Performance Radar



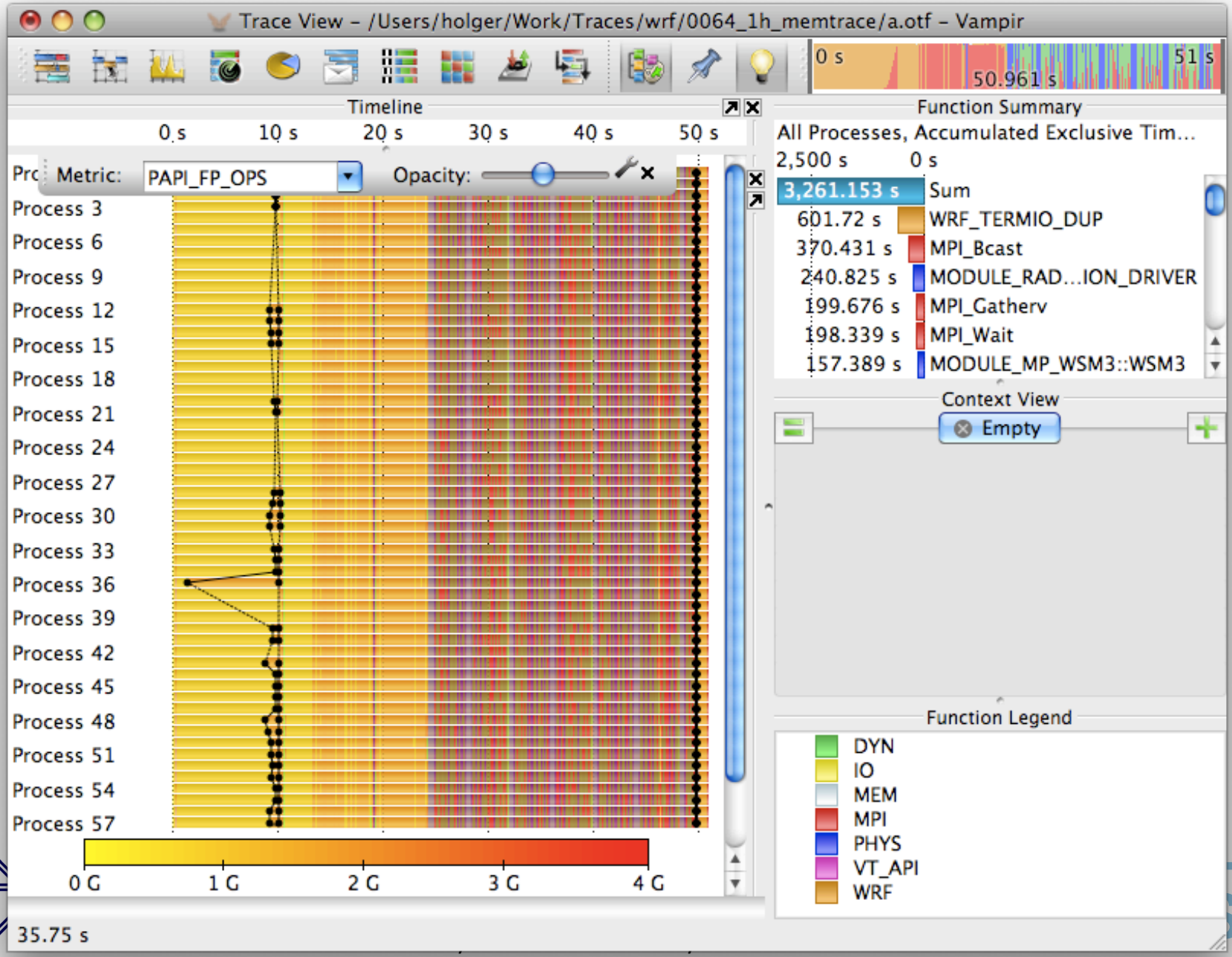
Advanced – Performance Radar



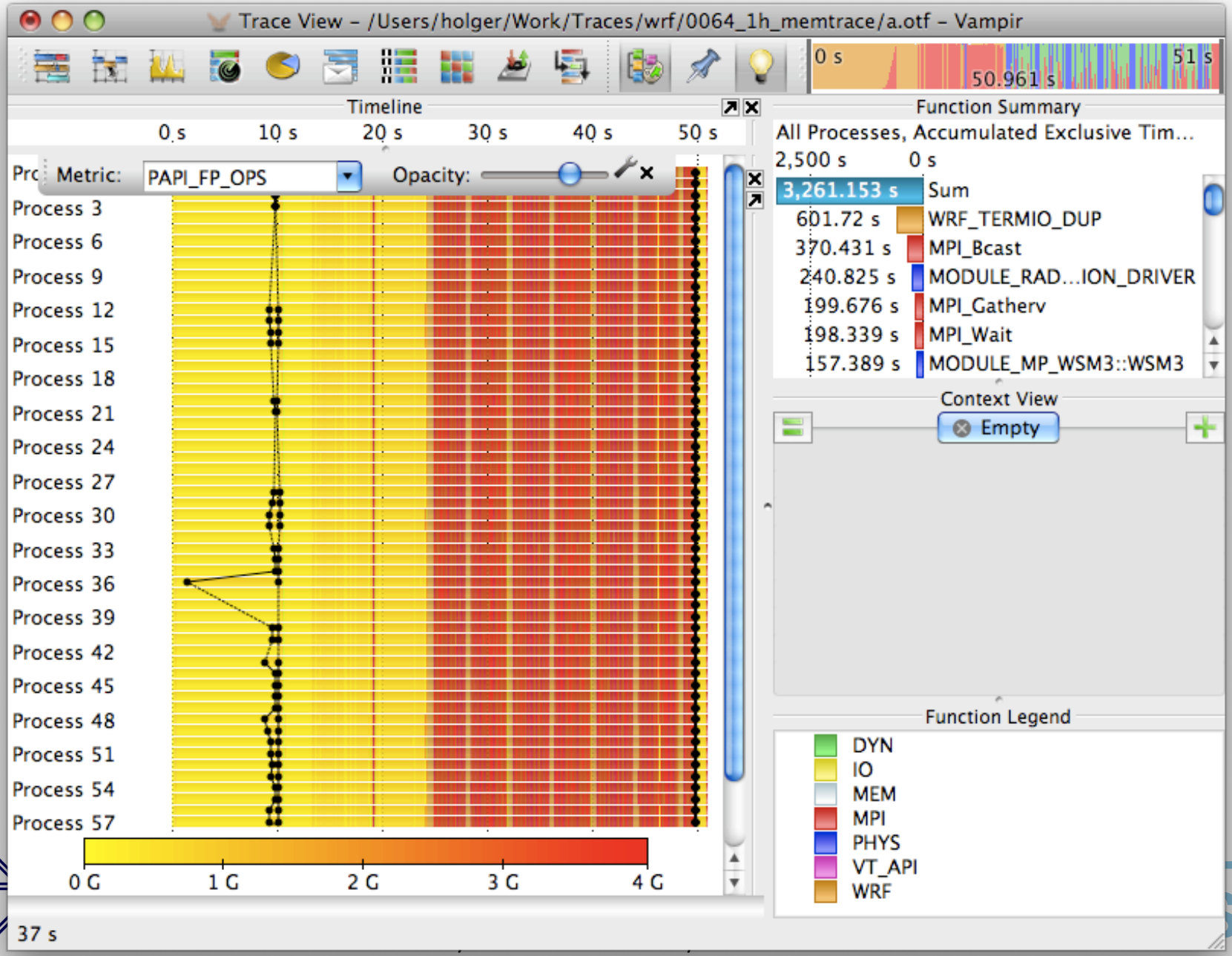
Advanced – Overlays



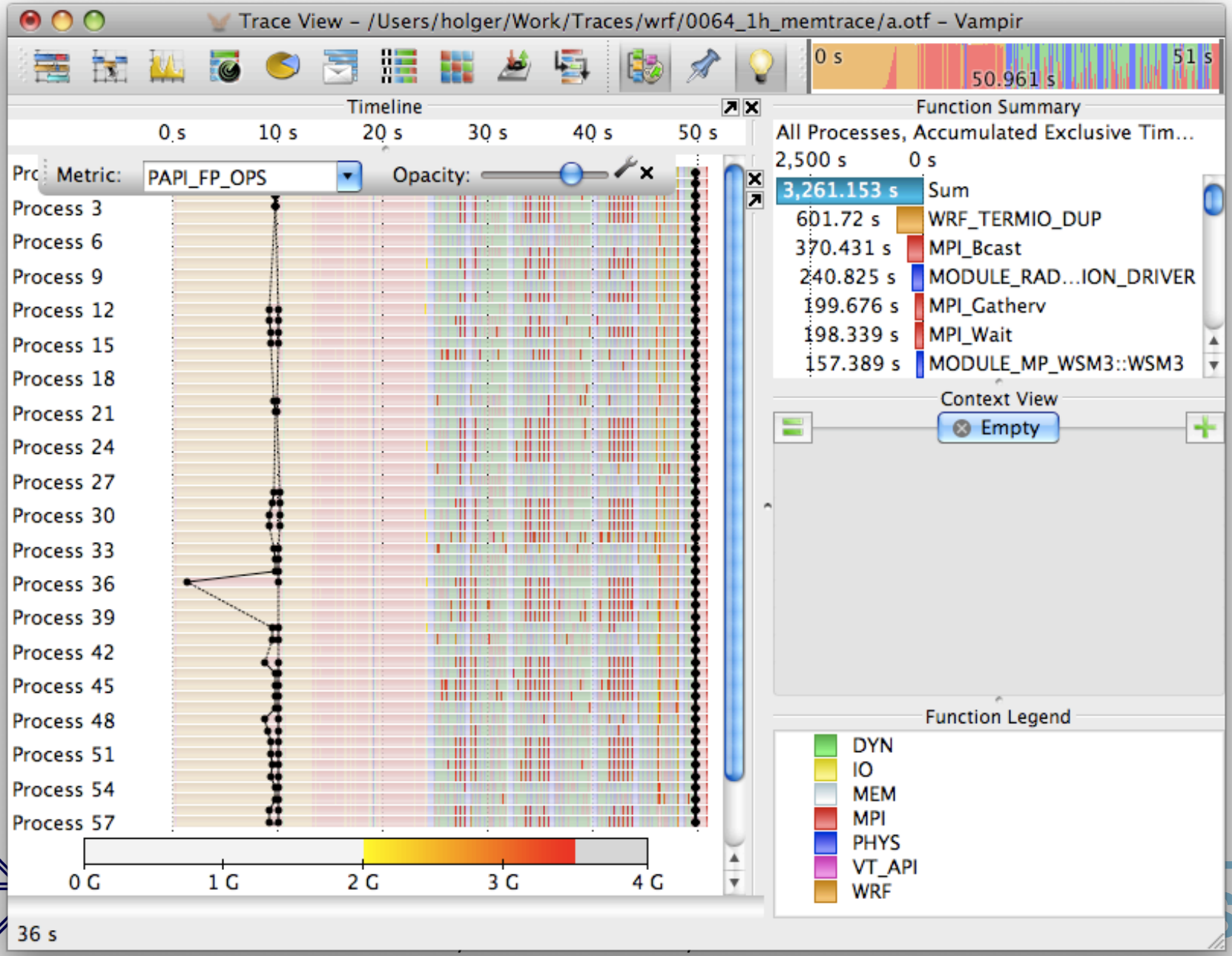
Advanced – Overlays



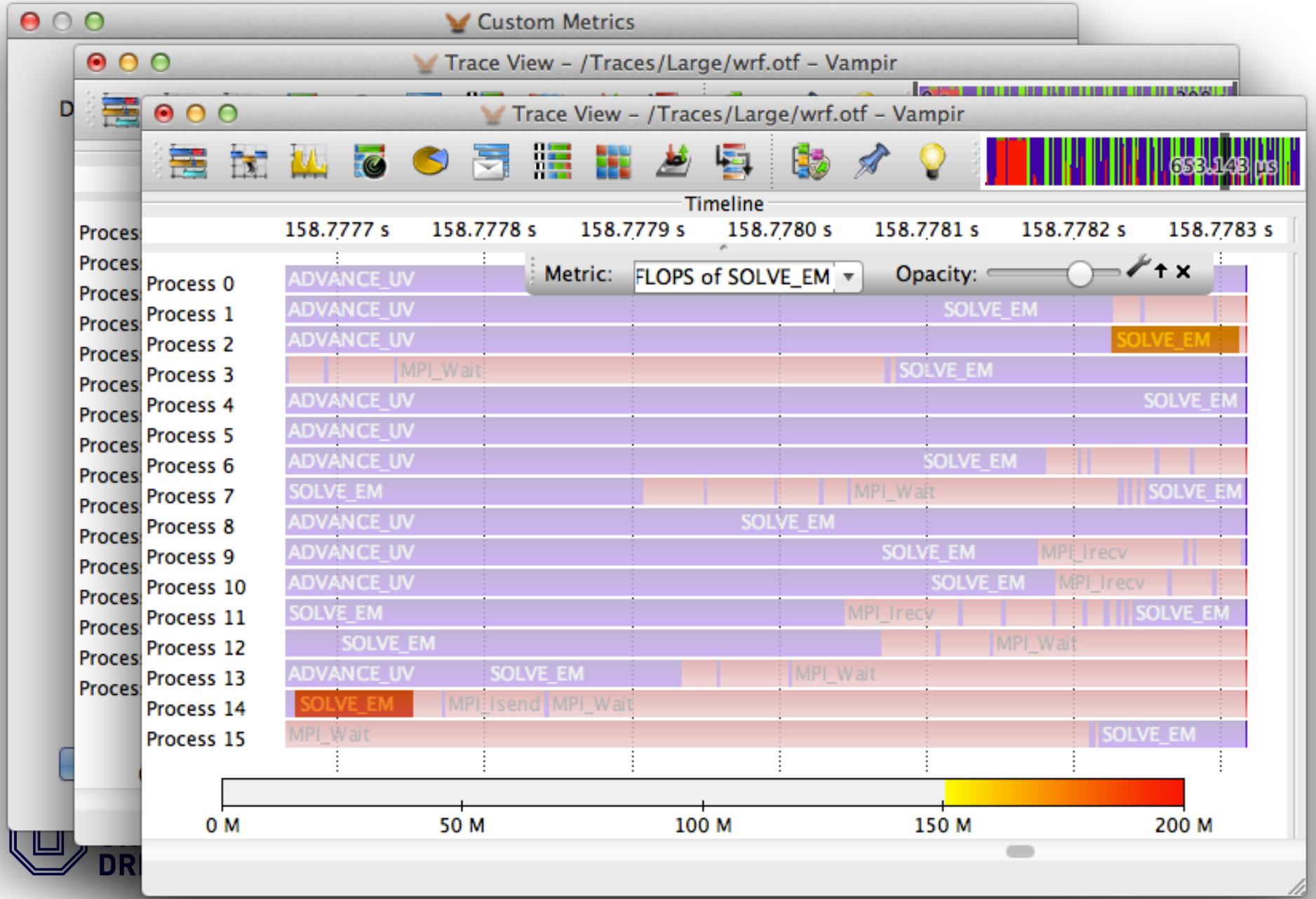
Advanced – Overlays



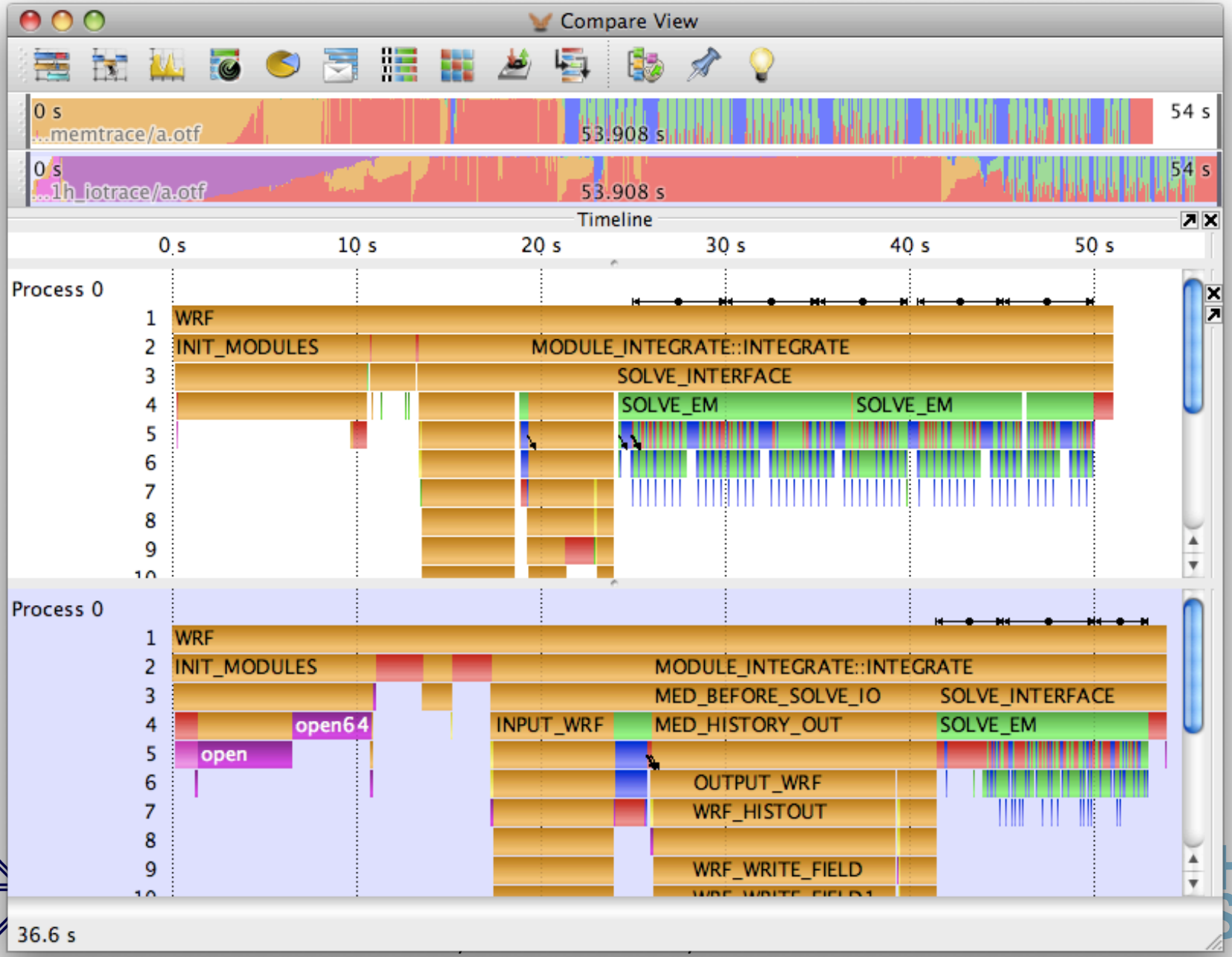
Advanced – Overlays



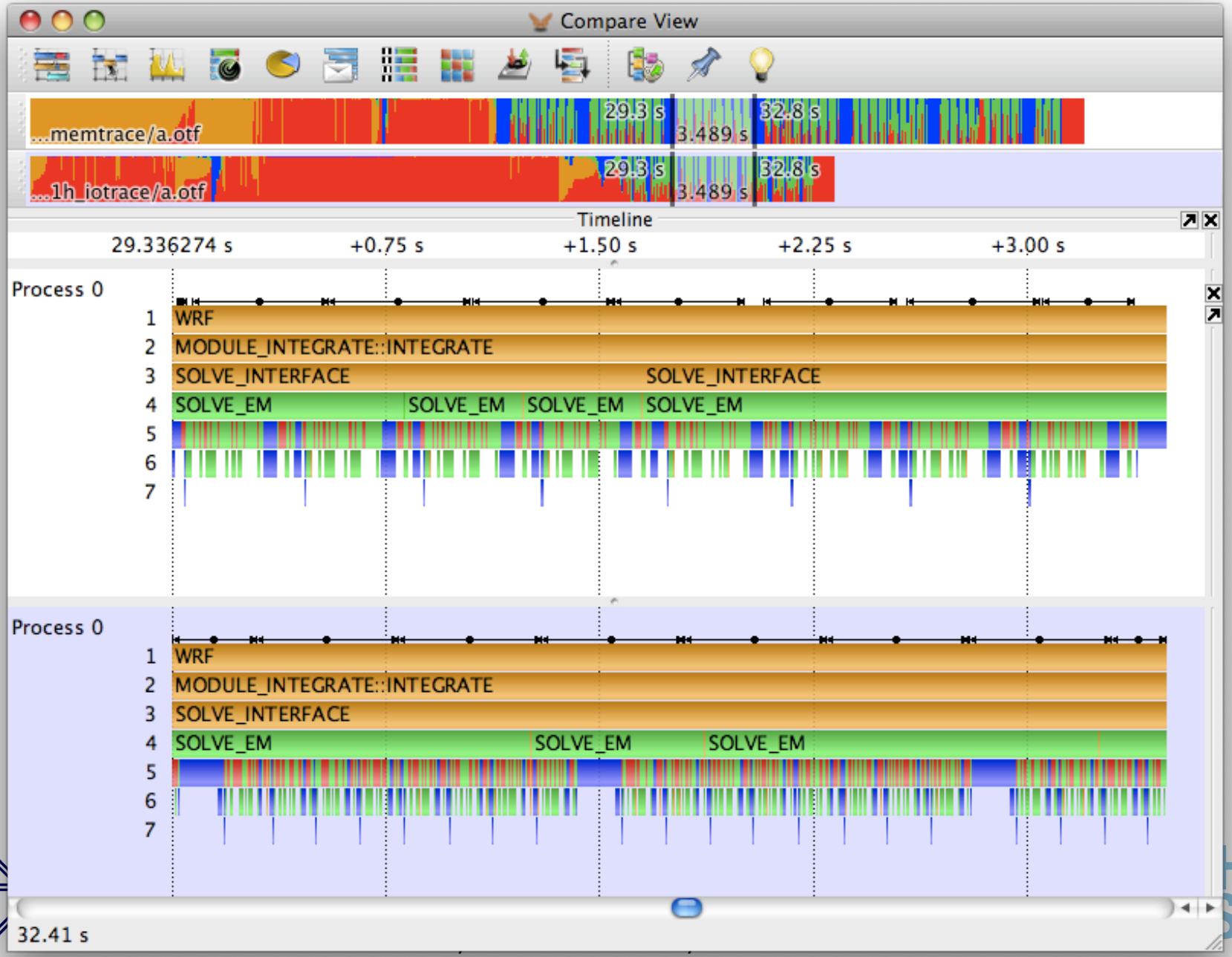
Advanced – Derived Counters



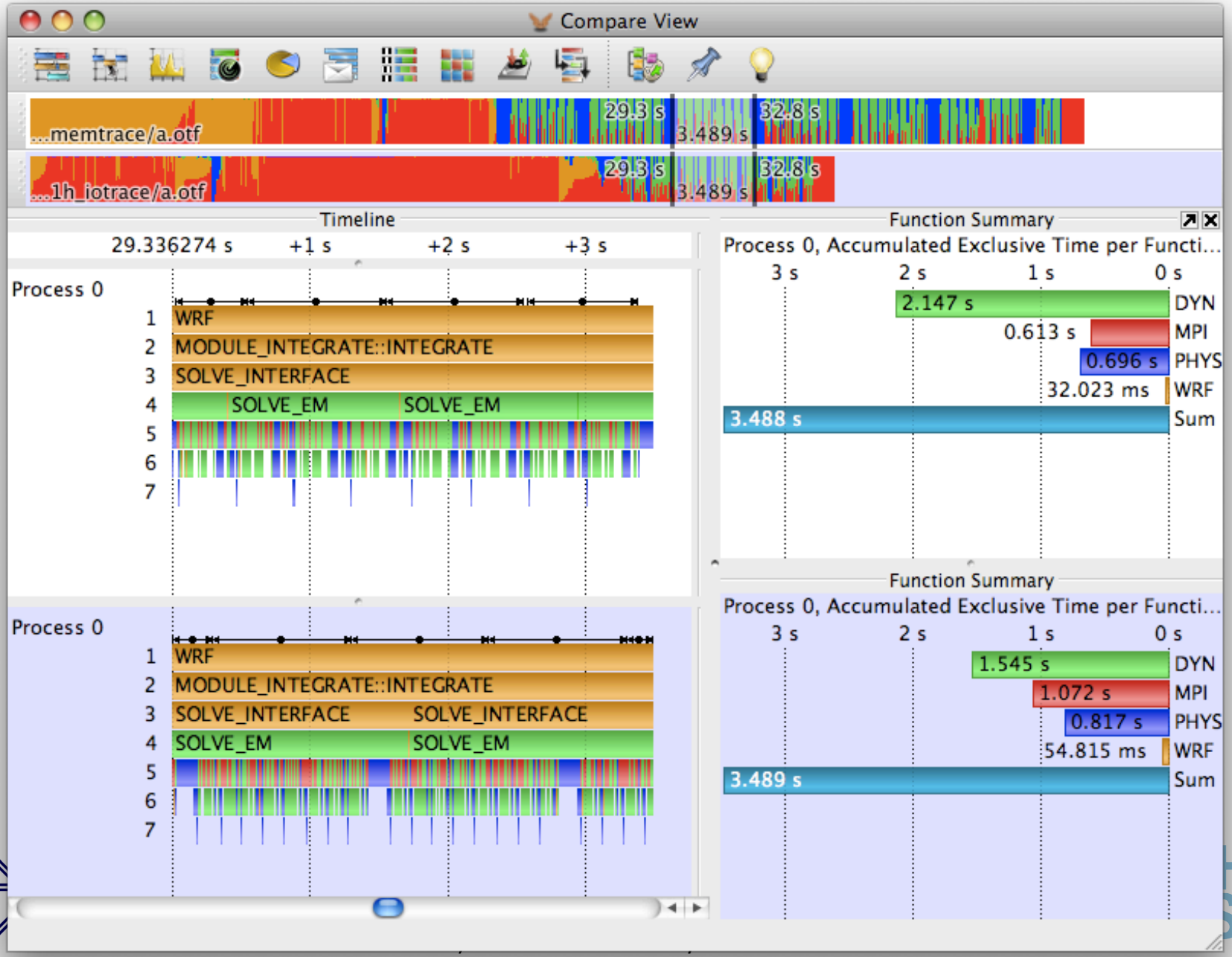
Advanced – Compare View



Advanced – Compare View

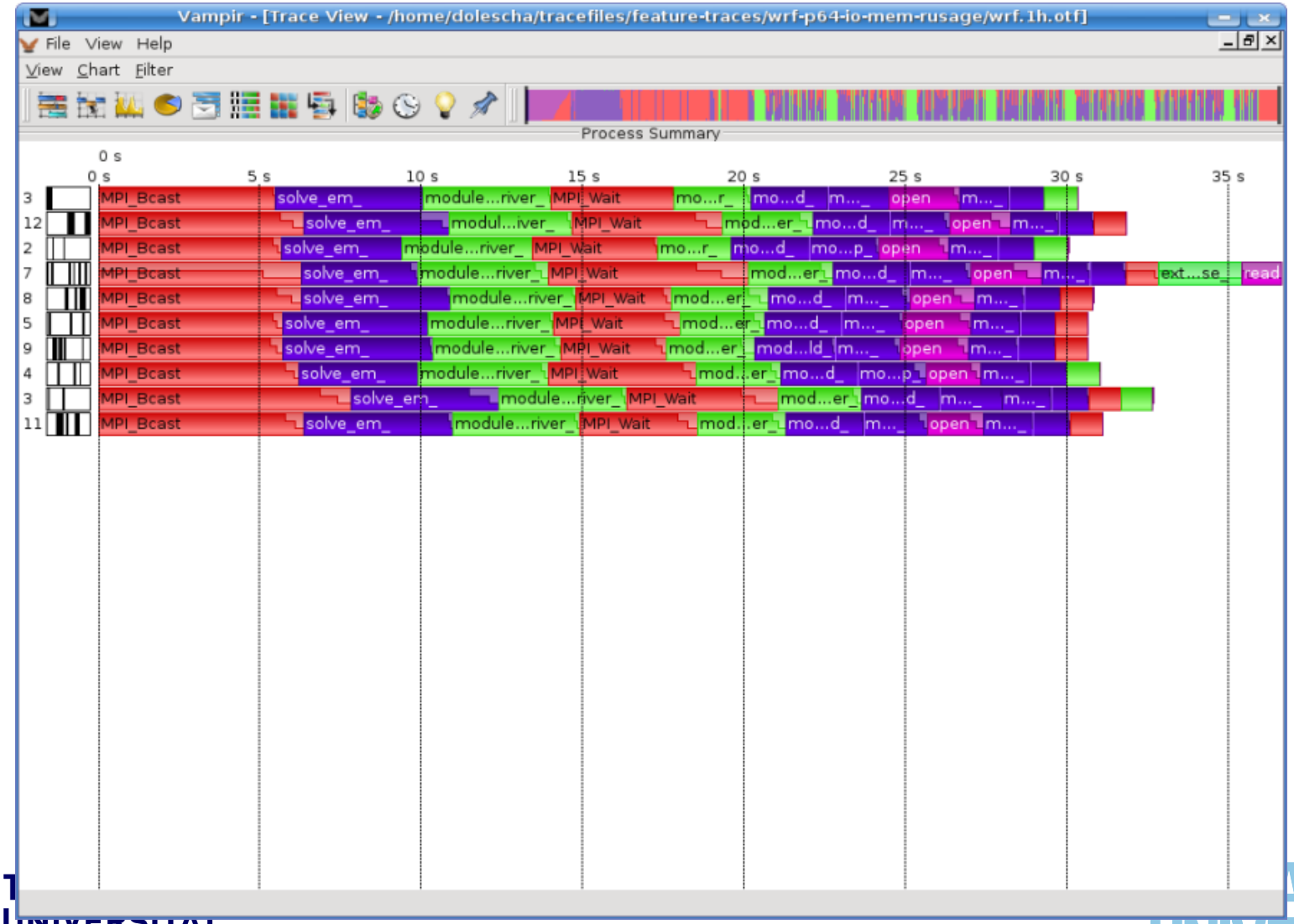


Advanced Visualization – Compare View



Advanced – Clustering View

- Group similar processes together:

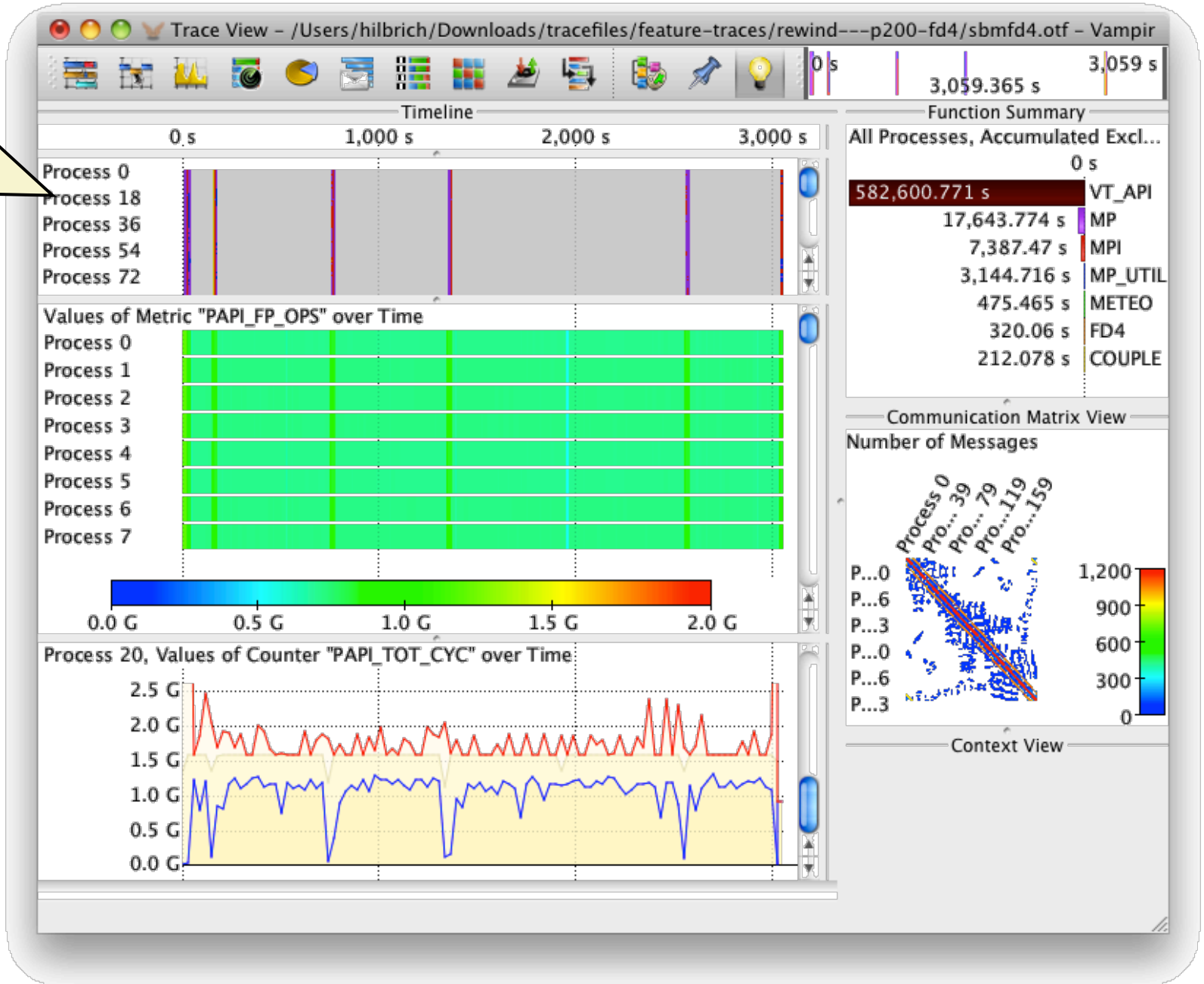


Advanced – Long Tracing Runs (1)

- Tracing long running applications is hard
- Case:
  - Long running application (~50min)
  - Non-reproducible bad performance in iterations
  - Coarse-grained traces gave no insight
  - Fine-grained traces would not fit into memory
- Solution:
  - Score-P (also in VampirTrace) feature “Rewind”
  - Allows to discard parts of the trace buffer
  - Controlled via instrumentation API

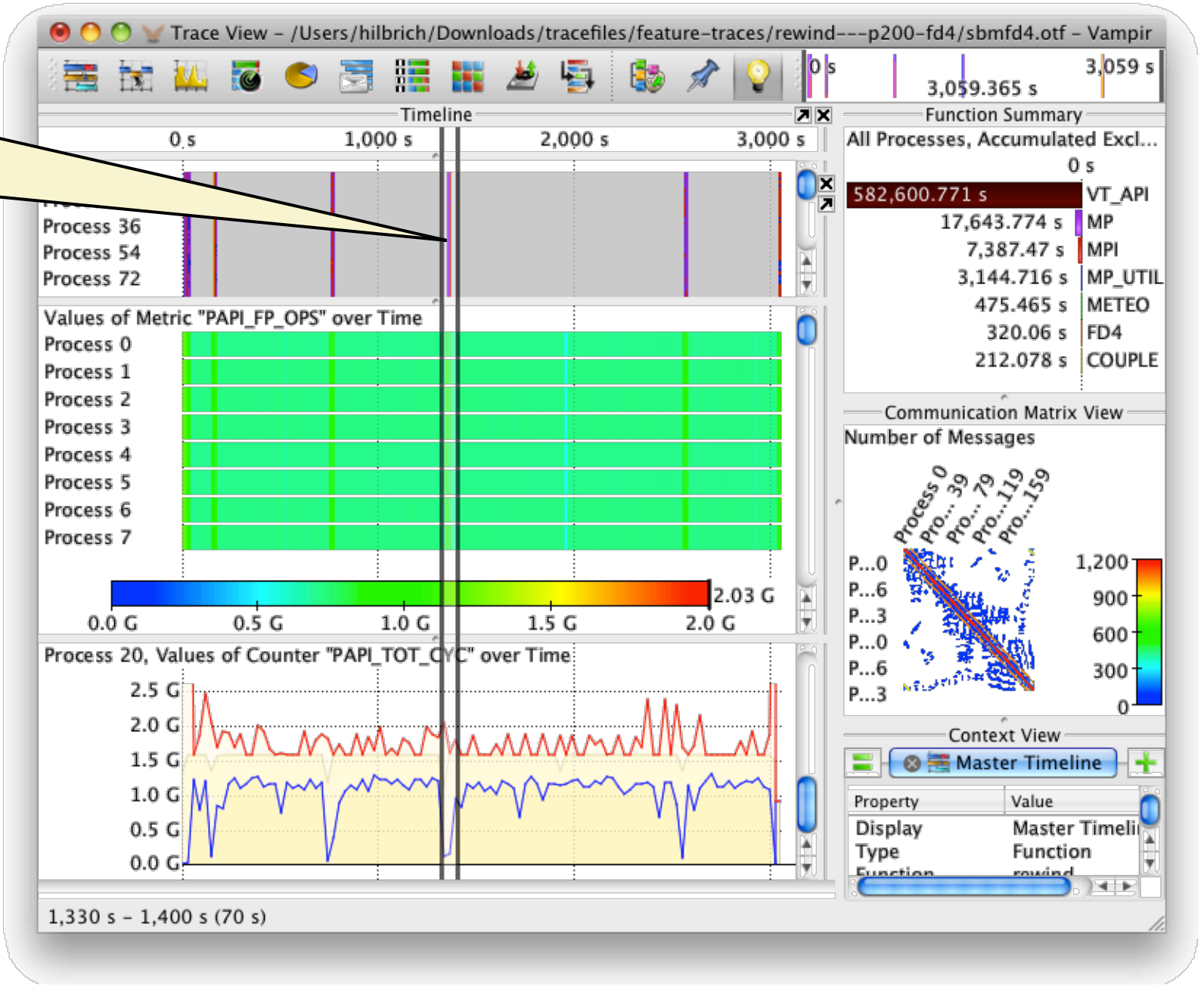
Advanced – Long Tracing Runs (2)

First and last iterations + 4 “slow” iterations



Advanced – Long Tracing Runs (3)

Zoom into a slow iteration

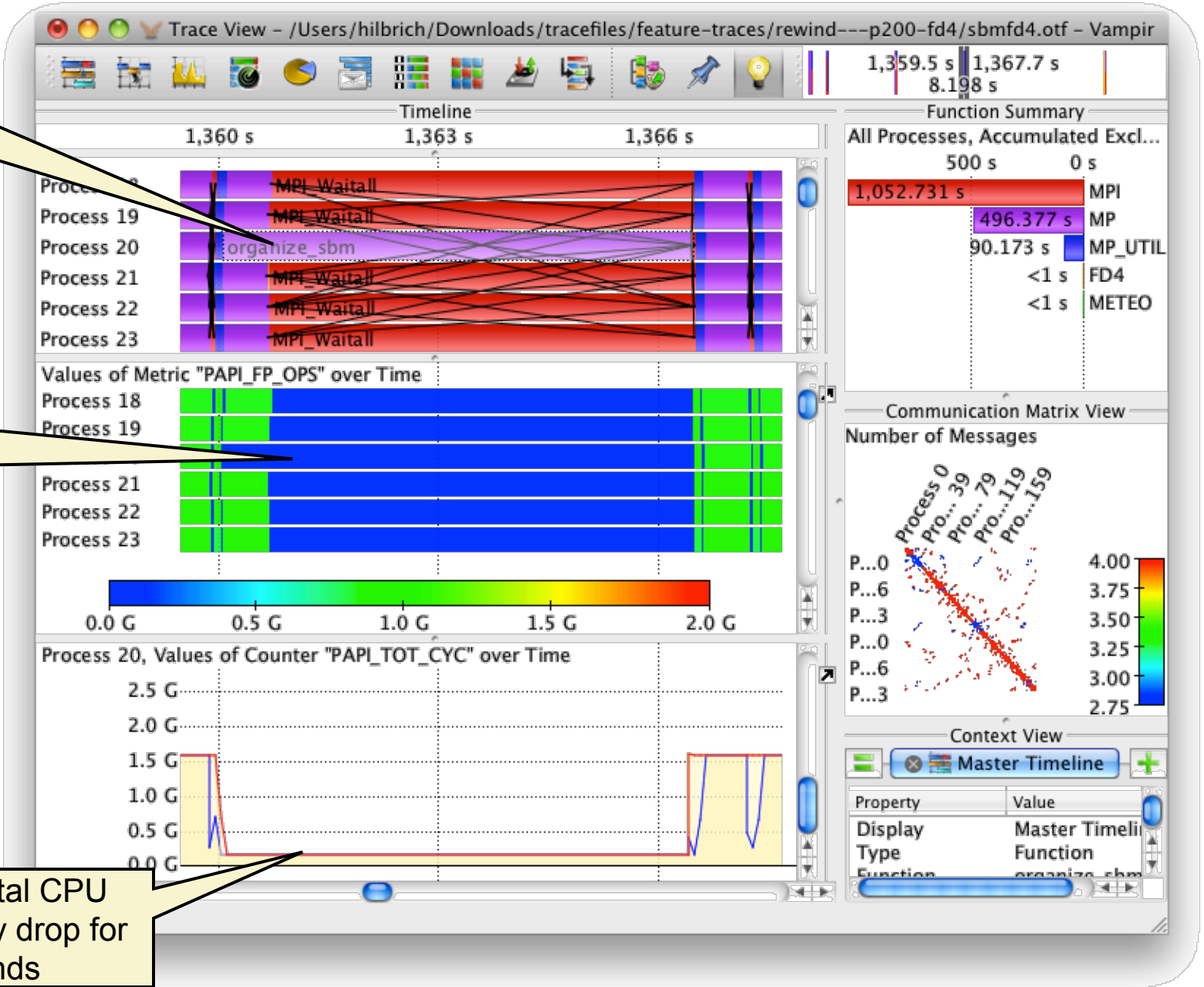


Advanced – Long Tracing Runs (4)

All except rank 20 need to wait

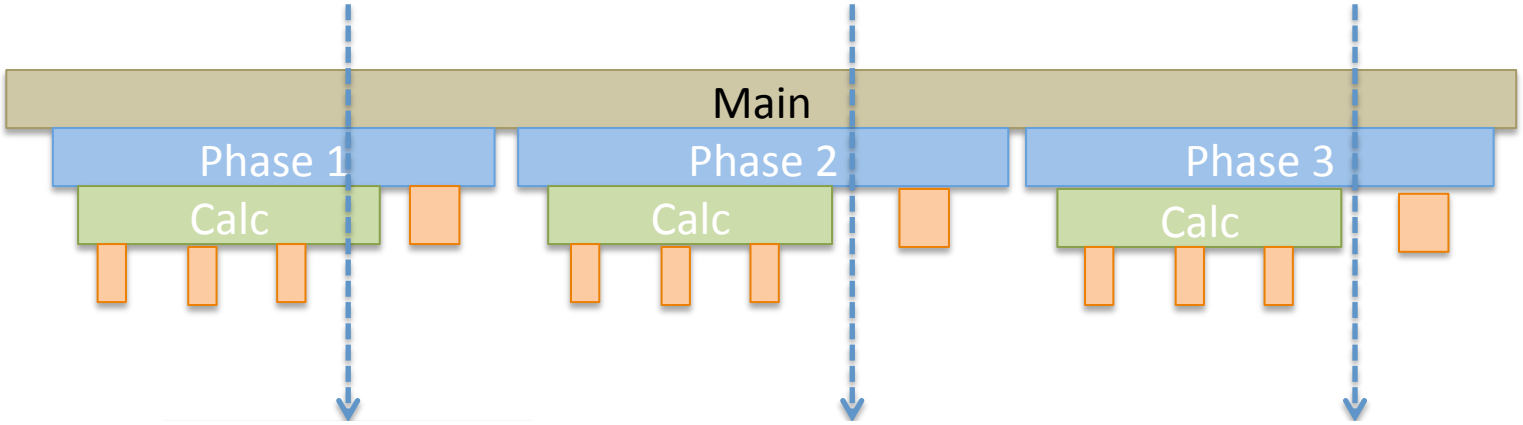
Can also be spotted in Radar

Process 20 total CPU cycles suddenly drop for > 6 seconds



Advanced – Sampling + Tracing

- Long running applications:
  - Requires large buffers or heavy filtering
  - Creating a filter requires runs in advance
- Codes with many small functions (e.g.: C++):
  - Function instrumentation a challenge
- In both cases, sampling simplifies workflows:



Advanced – Sampling + Tracing (2)

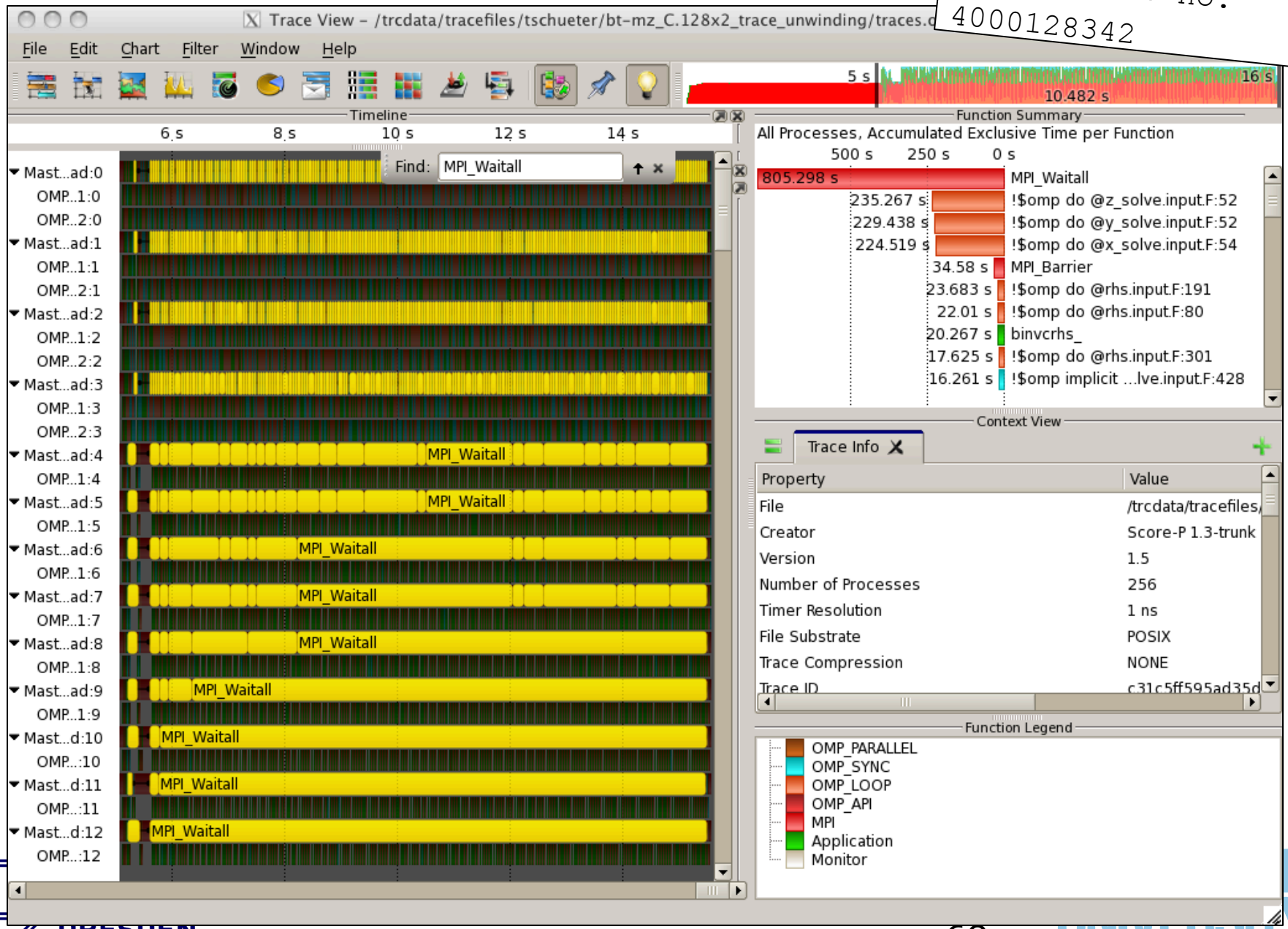
ORNL co-operation subcontract no. 4000128342

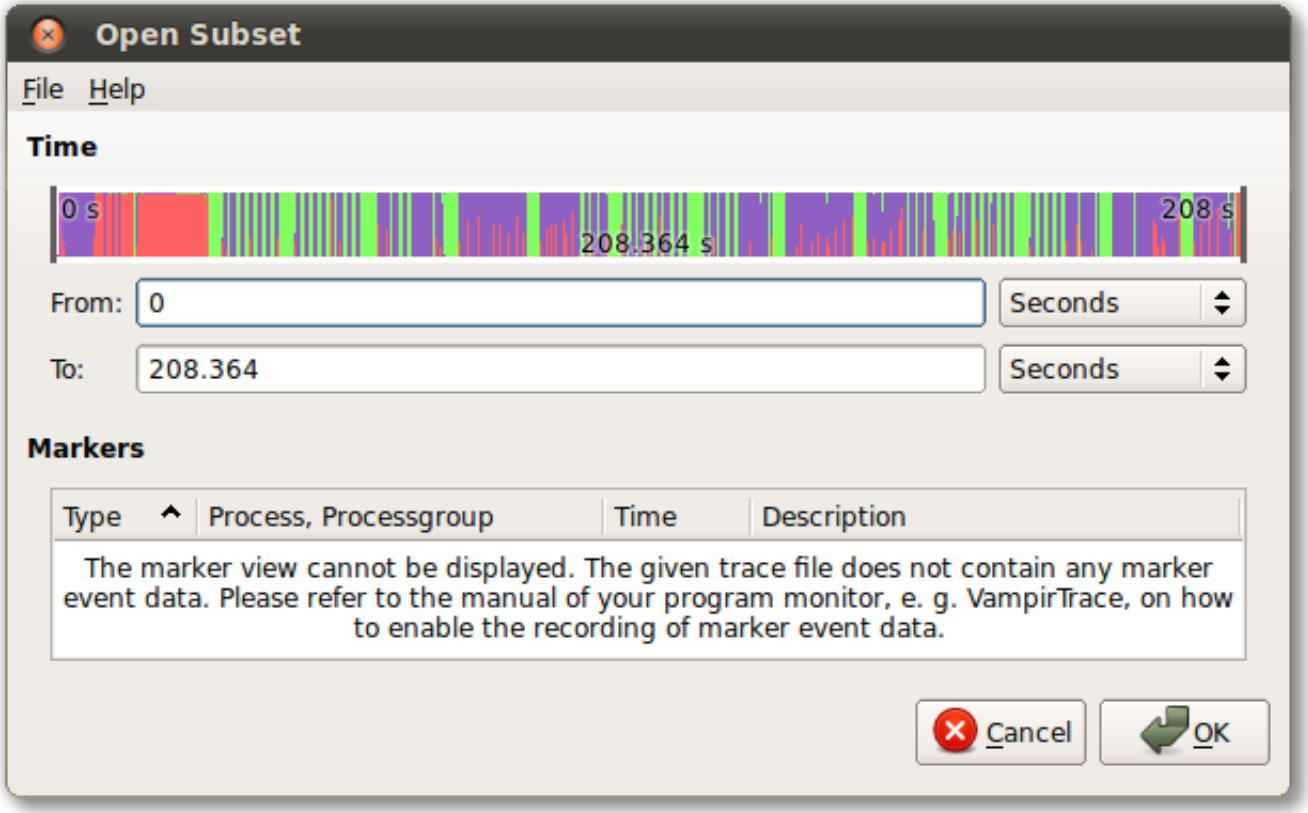
- Score-P 1.3 supports a combination of:
  - Tracing for MPI events
  - Sampling for everything else
- Simple configuration, e.g.:

```
% export SCOREP_ENABLE_TRACING=true
% export SCOREP_ENABLE_PROFILING=false
% export SCOREP_ENABLE_UNWINDING=true
% export SCOREP_TOTAL_MEMORY=200M
% export SCOREP_SAMPLING_EVENT=TIMER
% export SCOREP_SAMPLING_PERIOD=10000
% export SCOREP_EXPERIMENT_DIRECTORY=\
      'bt-mz_C.128x2_trace_unwinding'
```

Advanced – Sampling + Tracing (3)

ORNL co-operation subcontract no. 4000128342



Advanced – Subset Loading

- Quick look at large trace file with:
  - Just a specific time interval
  - A subset of the processes