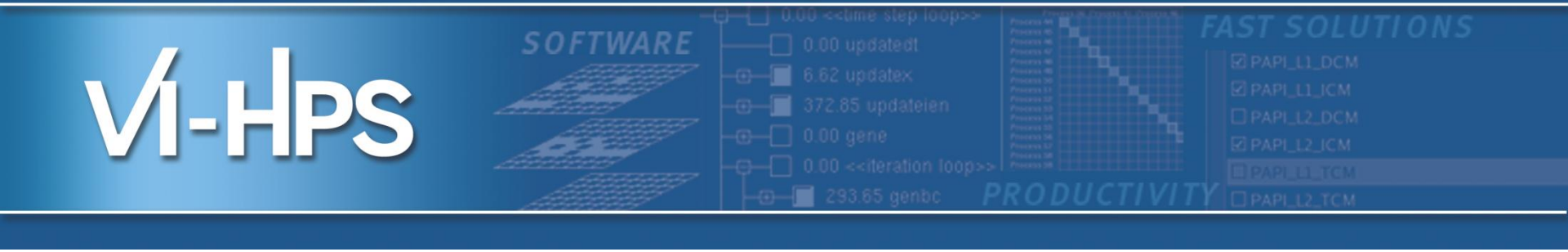


Interactive Performance Analysis with Vampir

UCAR Software Engineering Assembly
in Boulder/CO, 2013-04-03

Andreas Knüpfer, Thomas William

TU Dresden, Germany

Overview

- Introduction
- Vampir displays
- GPGPU support
- Scalability
- Performance Problems
- Conclusions & Preview on Tutorial

Why should I bother with performance analysis?

- Efficient usage of expensive and limited resources
- Scalability to achieve next bigger simulation
- Well, why are you here after all?

How should I do it?

- Optimization phase, just like the testing phase
- Use tools, avoid do-it-yourself-with-printf, really!

What methods are available?

- Profiling
- Event Tracing

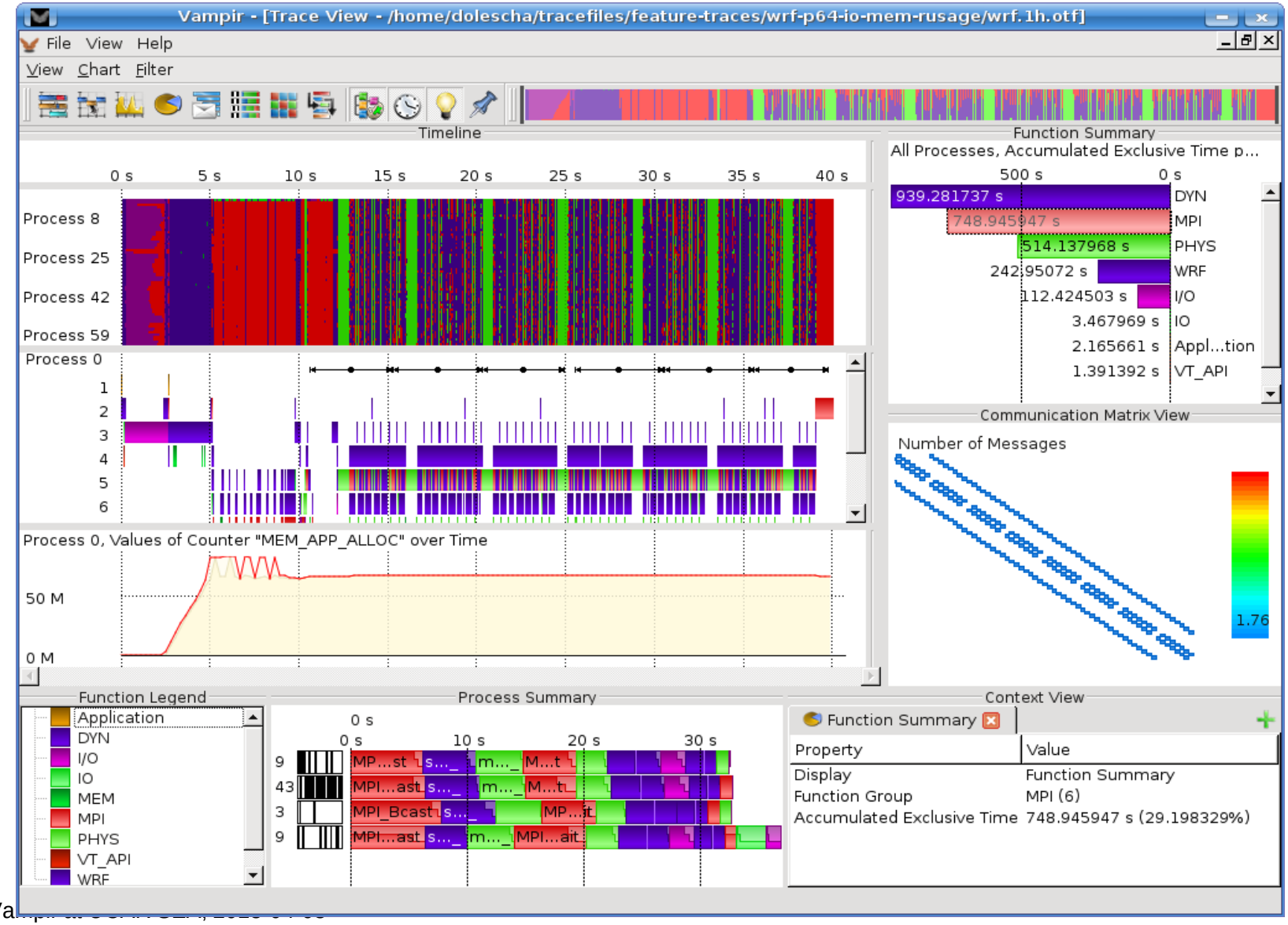
Vampir Event Trace Visualization

- Based on recorded event traces
  - Sorted collection of run-time events per process
- Event types with time stamp and process:
  - Enter given function/subroutine
  - Send or receive message of N bytes to/from P
  - Hardware performance counter values
  - More …
- Visualize dynamic run-time behavior
  - Various timeline displays
  - Various statistics displays
- Interactive browsing/zooming/selection
  - Zoom in time and processes
  - Feasible for very large and highly parallel traces

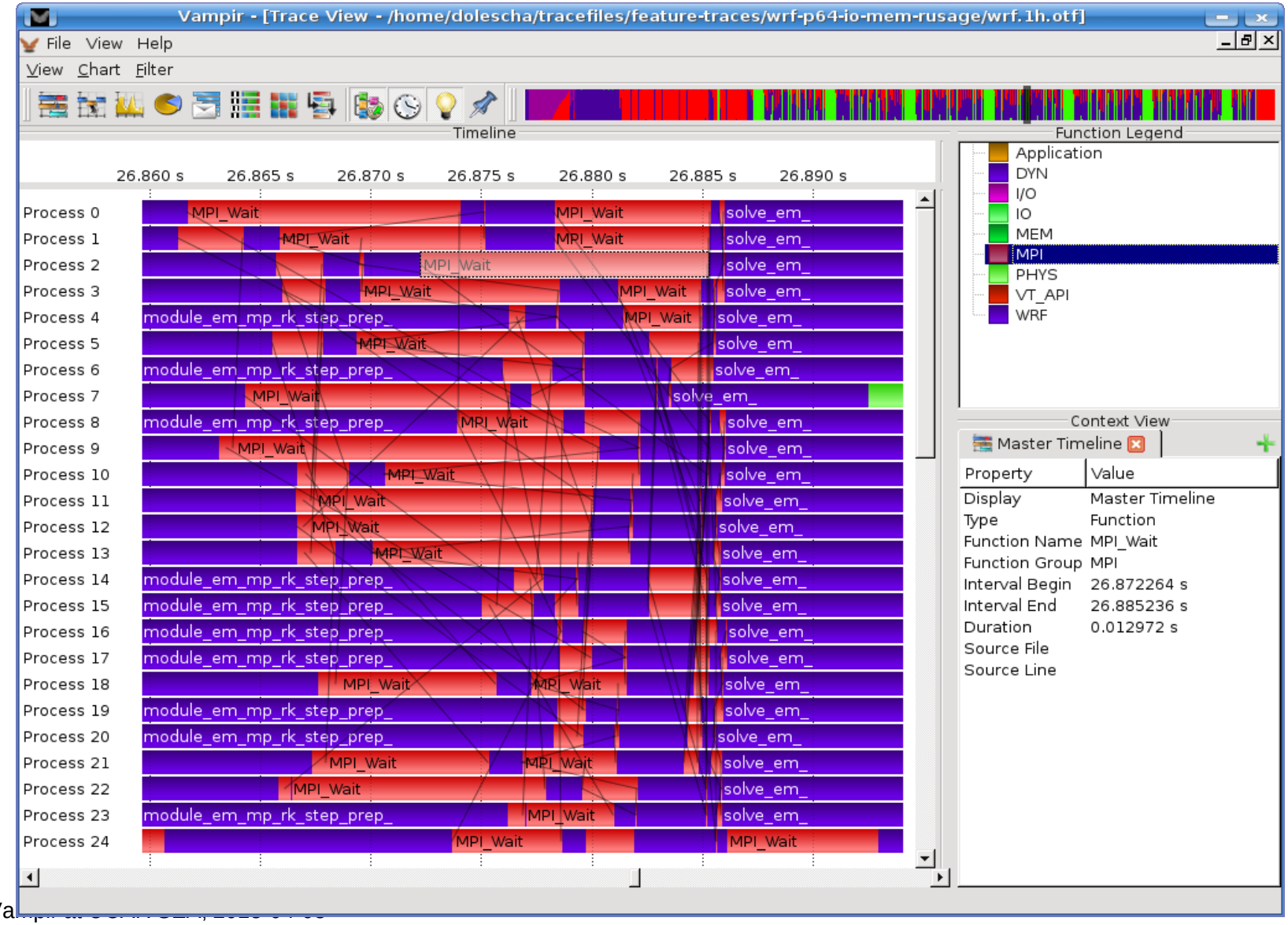
Overview

- Introduction
- Vampir displays
- GPGPU support
- Scalability
- Performance Problems
- Conclusions & Preview on Tutorial

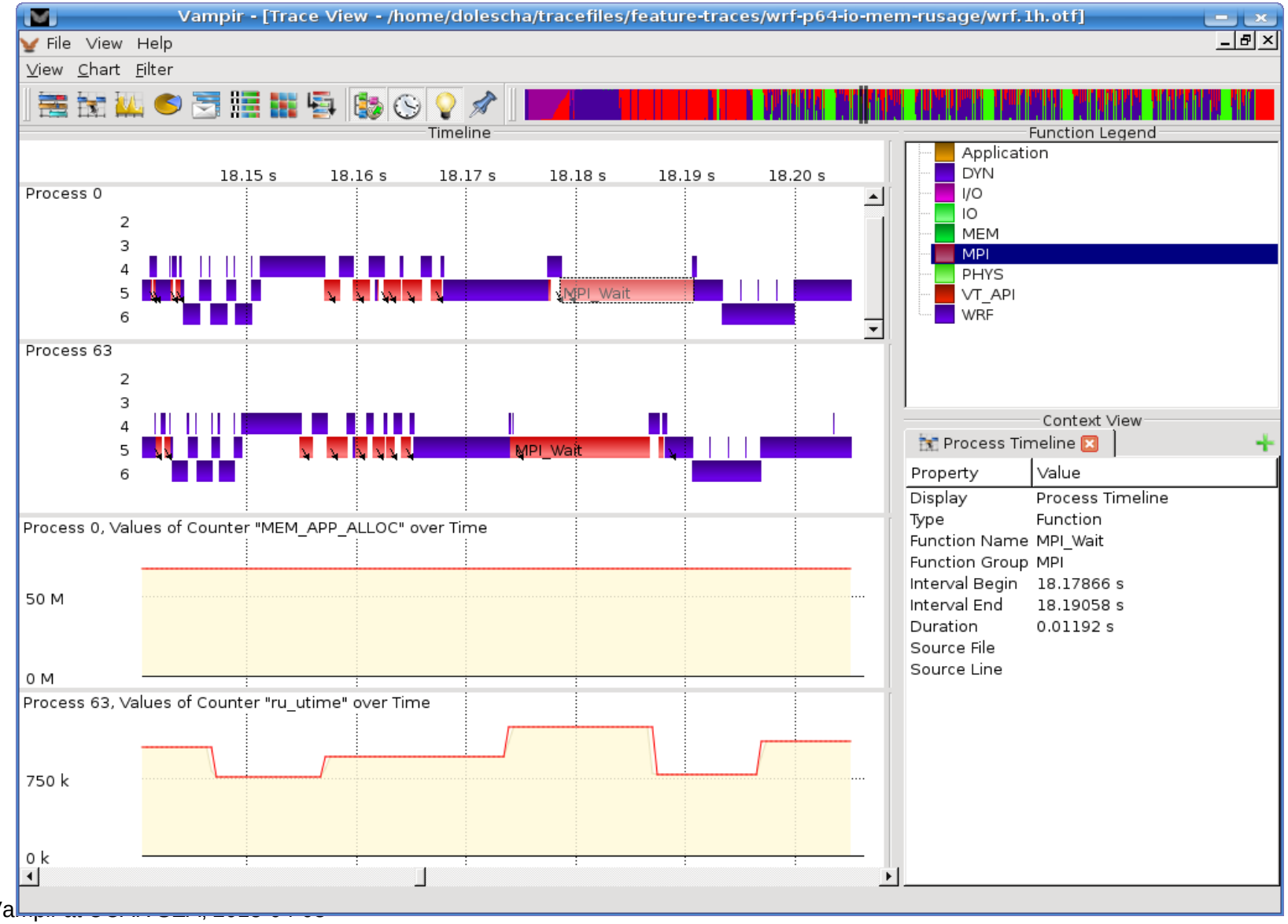
Vampir Display Overview



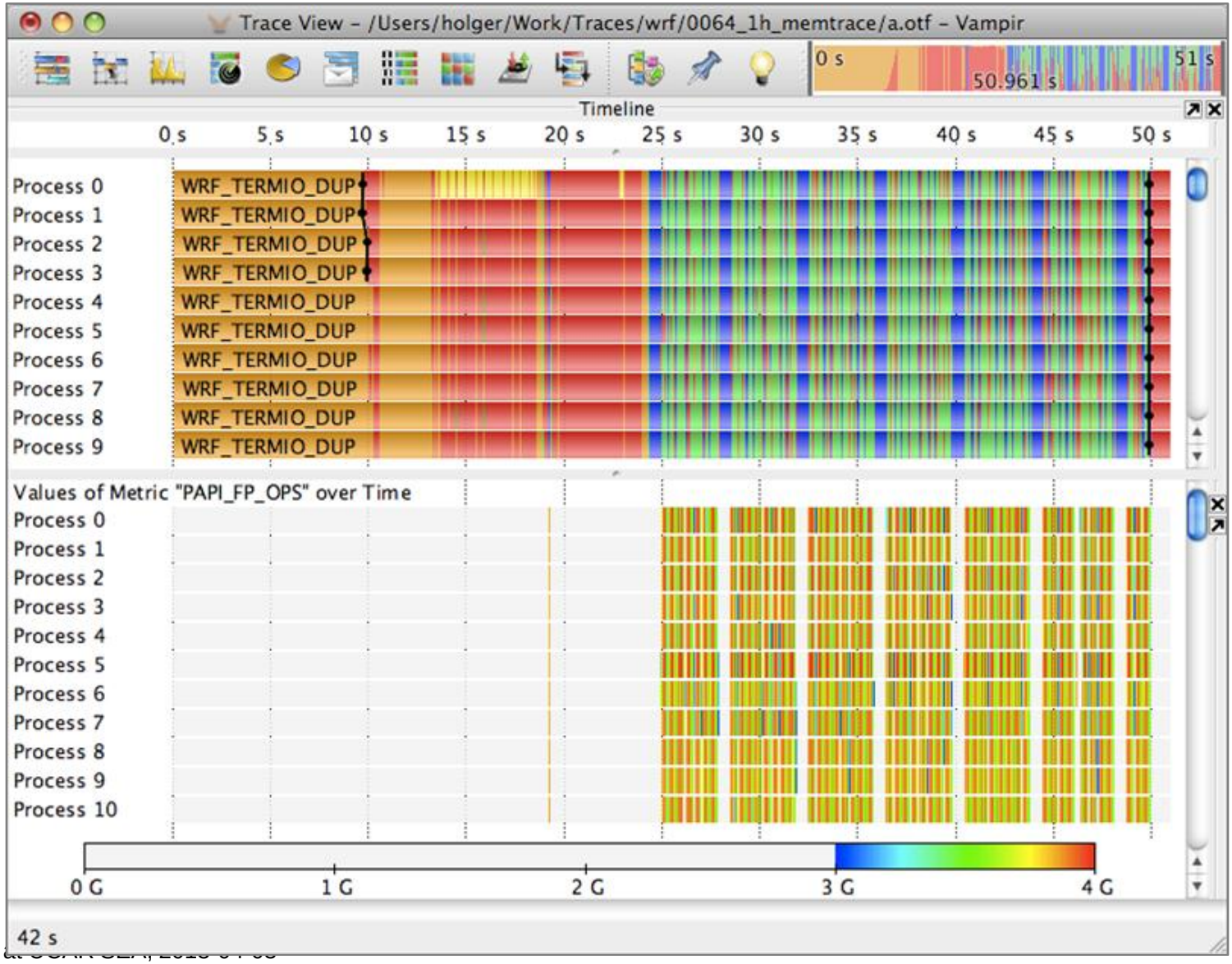
Master Timeline



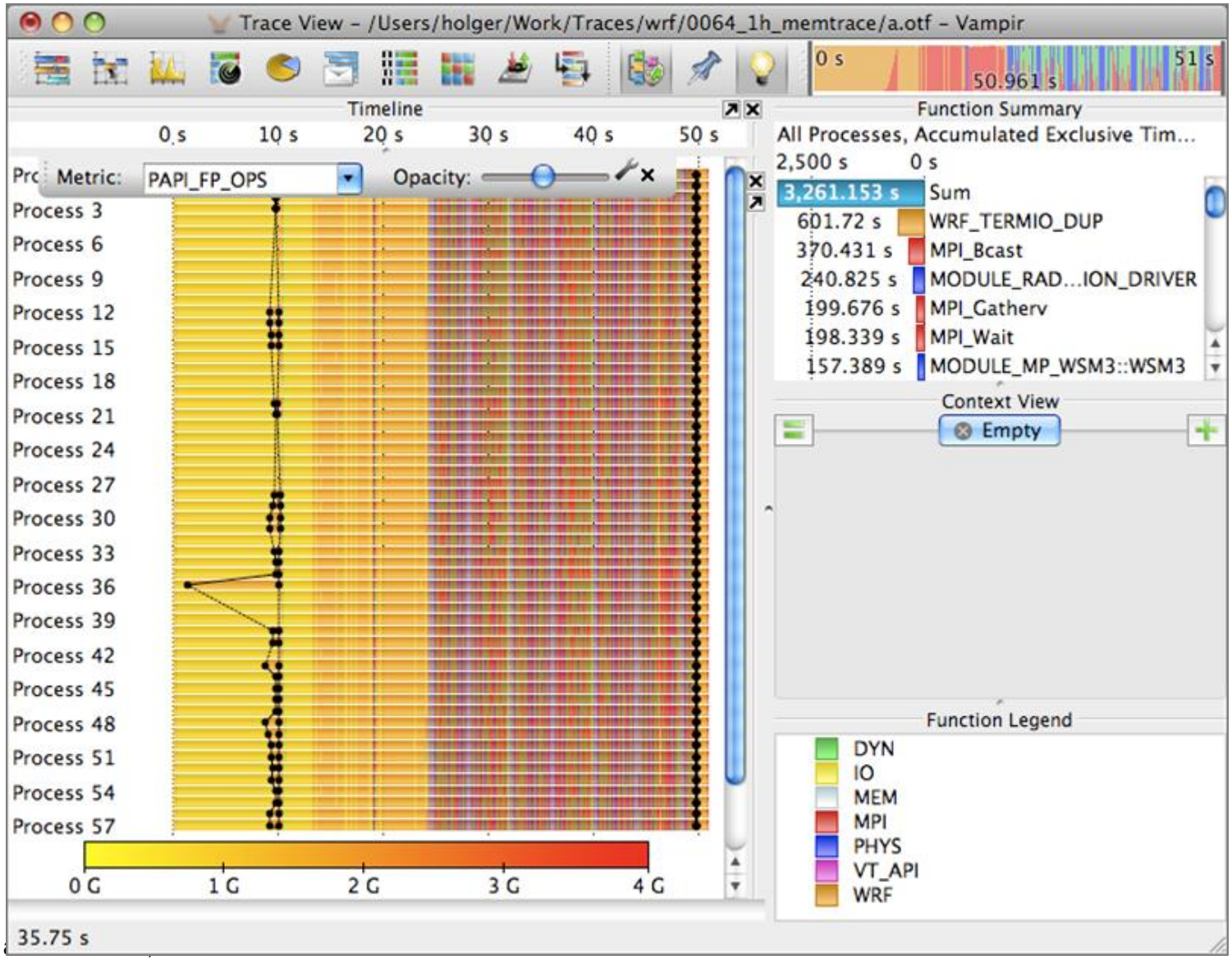
Process and Counter Timelines



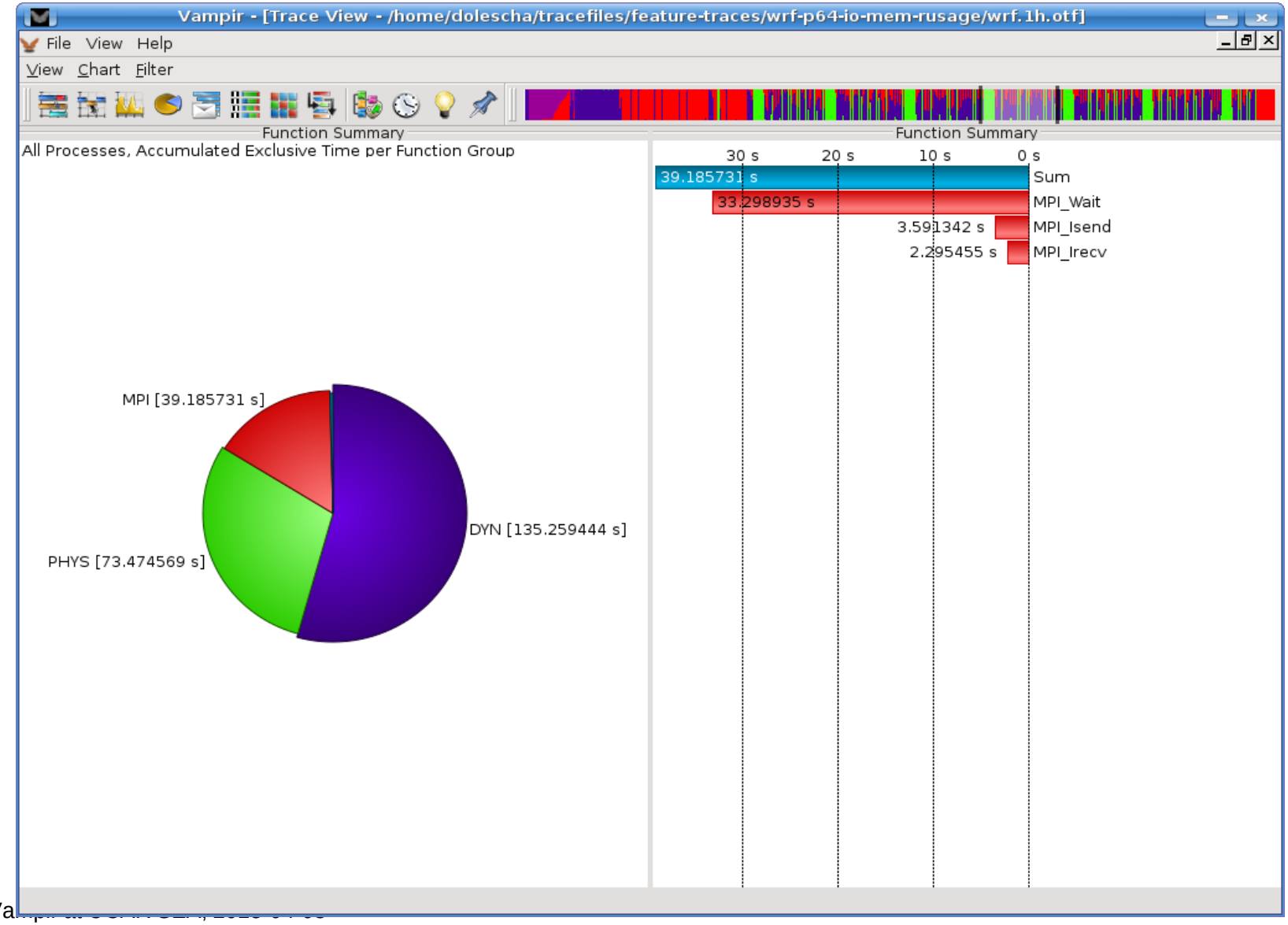
Performance Radar



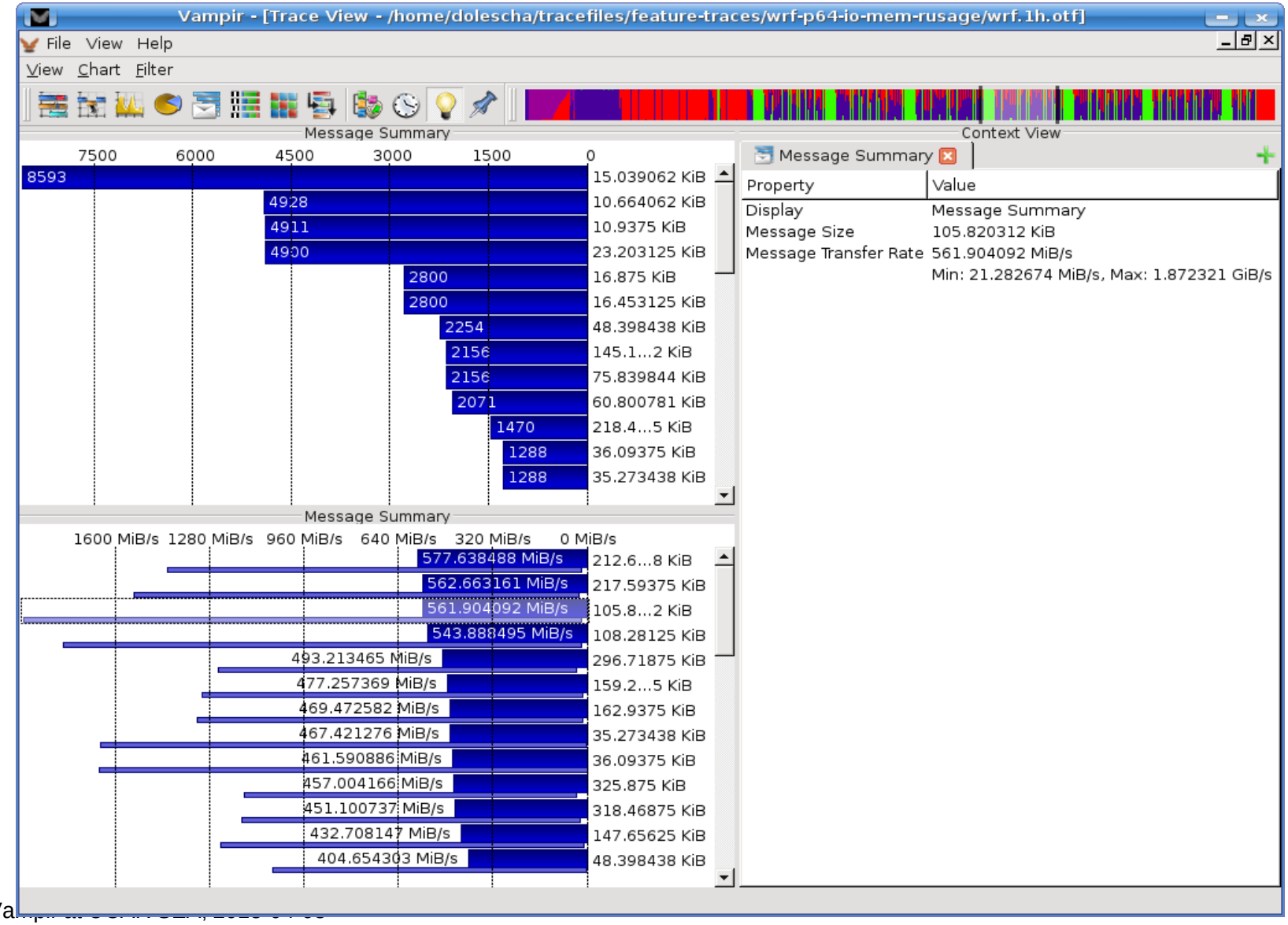
Performance Radar in Overlay Mode



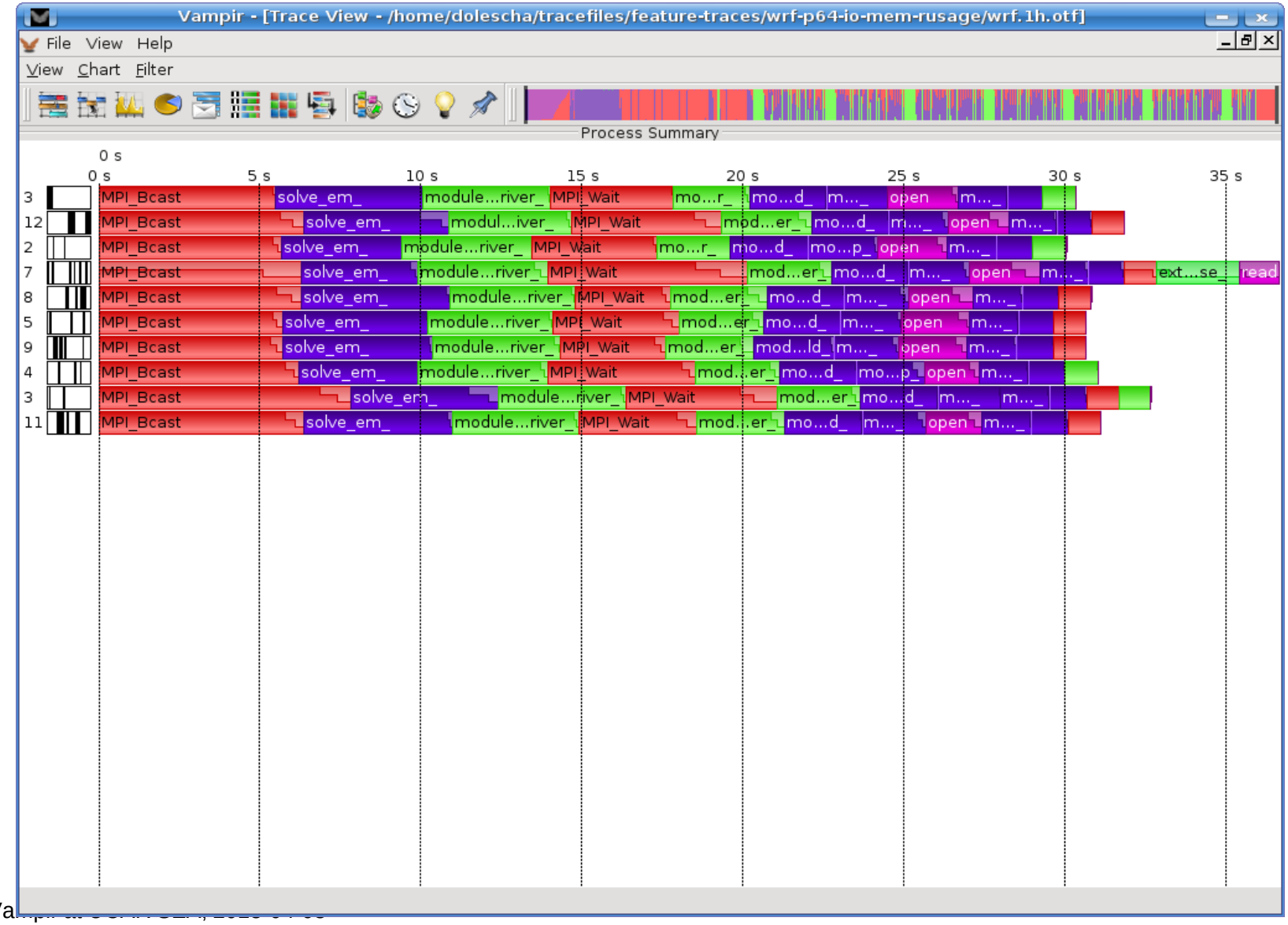
Function Summary



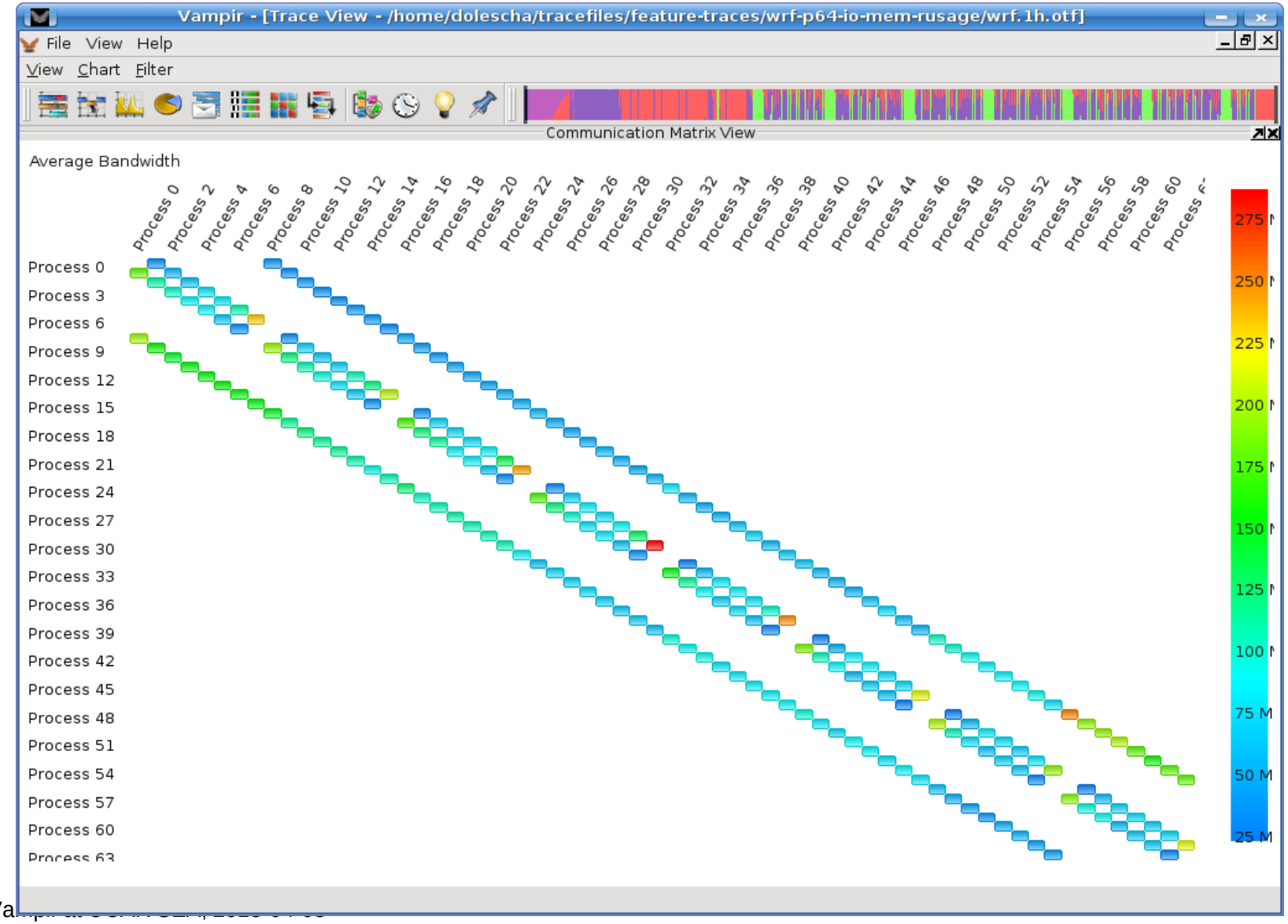
Message Summary



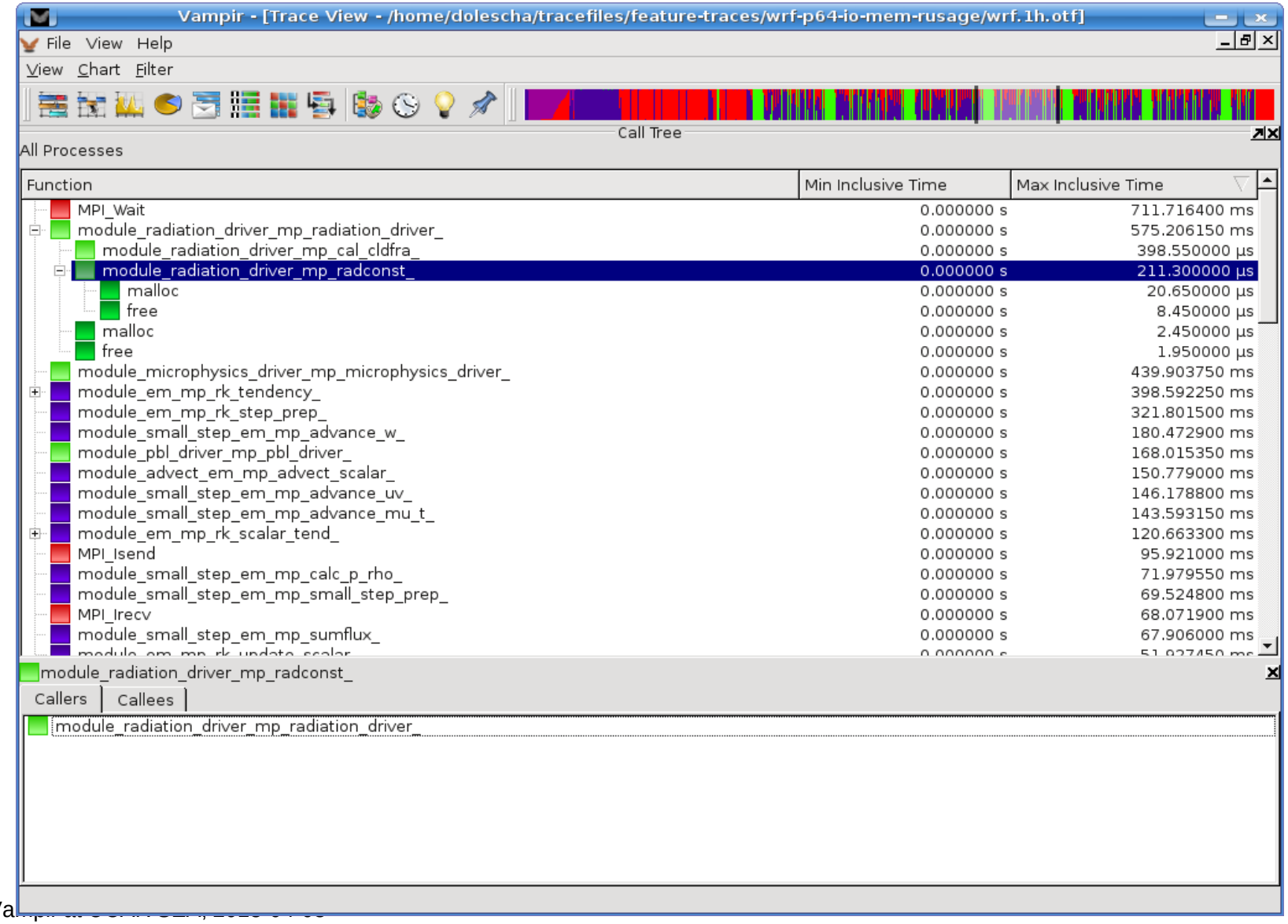
Process Summary



Communication Matrix



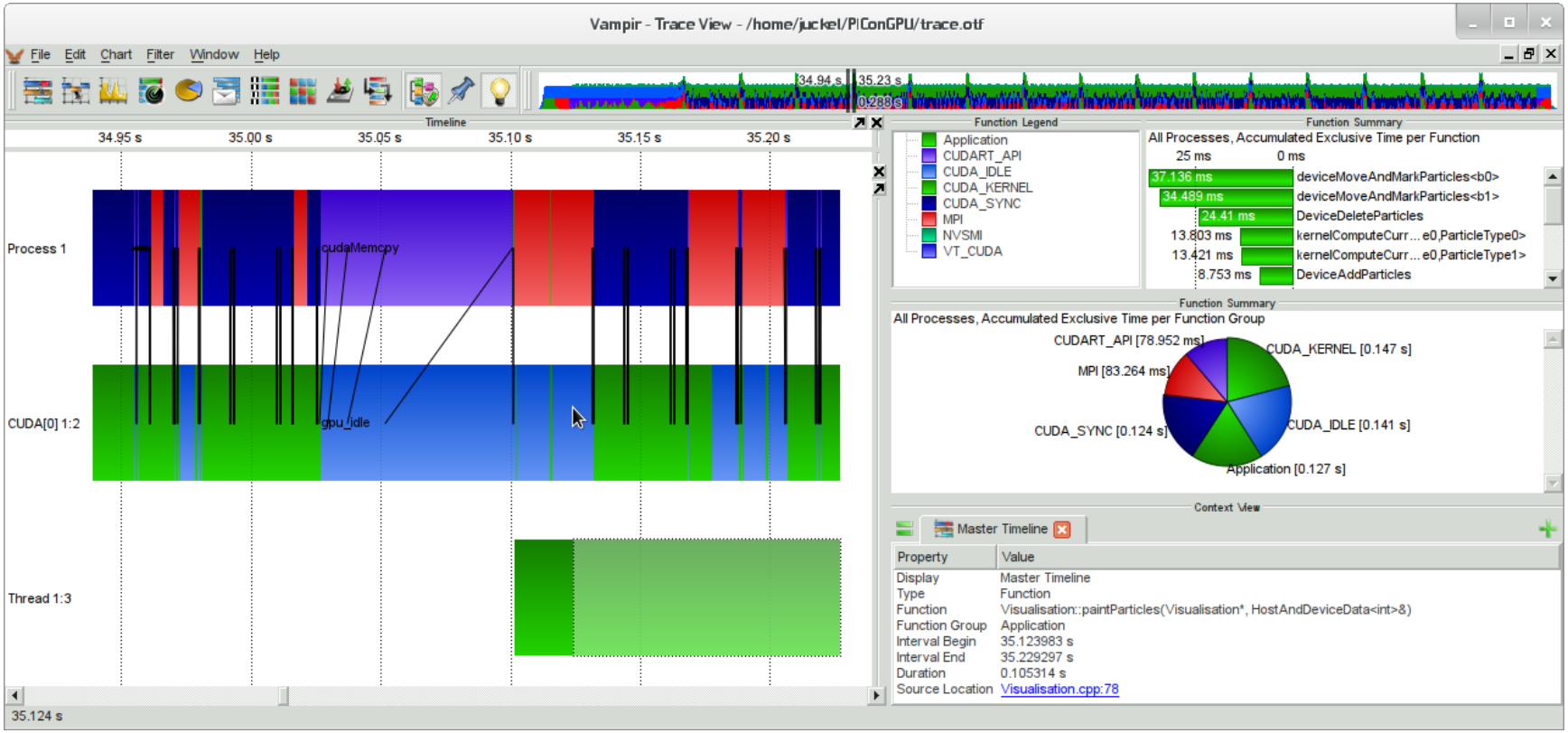
Call Tree


Vampir - [Trace View - /home/dolescha/tracefiles/feature-traces/wrf-p64-io-mem-rusage/wrf.1h.otf]
File View Help
View Chart Filter
Call Tree
All Processes
Function
Min Inclusive Time
Max Inclusive Time
MPI_Wait 0.000000 s 711.716400 ms
module_radiation_driver_mp_radiation_driver_ 0.000000 s 575.206150 ms
module_radiation_driver_mp_cal_cldfra_ 0.000000 s 398.550000 µs
module_radiation_driver_mp_radconst_ 0.000000 s 211.300000 µs
malloc 0.000000 s 20.650000 µs
free 0.000000 s 8.450000 µs
malloc 0.000000 s 2.450000 µs
free 0.000000 s 1.950000 µs
module_microphysics_driver_mp_microphysics_driver_ 0.000000 s 439.903750 ms
module_em_mp_rk_tendency_ 0.000000 s 398.592250 ms
module_em_mp_rk_step_prep_ 0.000000 s 321.801500 ms
module_small_step_em_mp_advance_w_ 0.000000 s 180.472900 ms
module_pbl_driver_mp_pbl_driver_ 0.000000 s 168.015350 ms
module_advect_em_mp_advect_scalar_ 0.000000 s 150.779000 ms
module_small_step_em_mp_advance_uv_ 0.000000 s 146.178800 ms
module_small_step_em_mp_advance_mu_t_ 0.000000 s 143.593150 ms
module_em_mp_rk_scalar_tend_ 0.000000 s 120.663300 ms
MPI_Isend 0.000000 s 95.921000 ms
module_small_step_em_mp_calc_p_rho_ 0.000000 s 71.979550 ms
module_small_step_em_mp_small_step_prep_ 0.000000 s 69.524800 ms
MPI_Irecv 0.000000 s 68.071900 ms
module_small_step_em_mp_sumflux_ 0.000000 s 67.906000 ms
module_radiation_driver_mp_radconst_
Callers
Callees
module_radiation_driver_mp_radiation_driver_

Overview

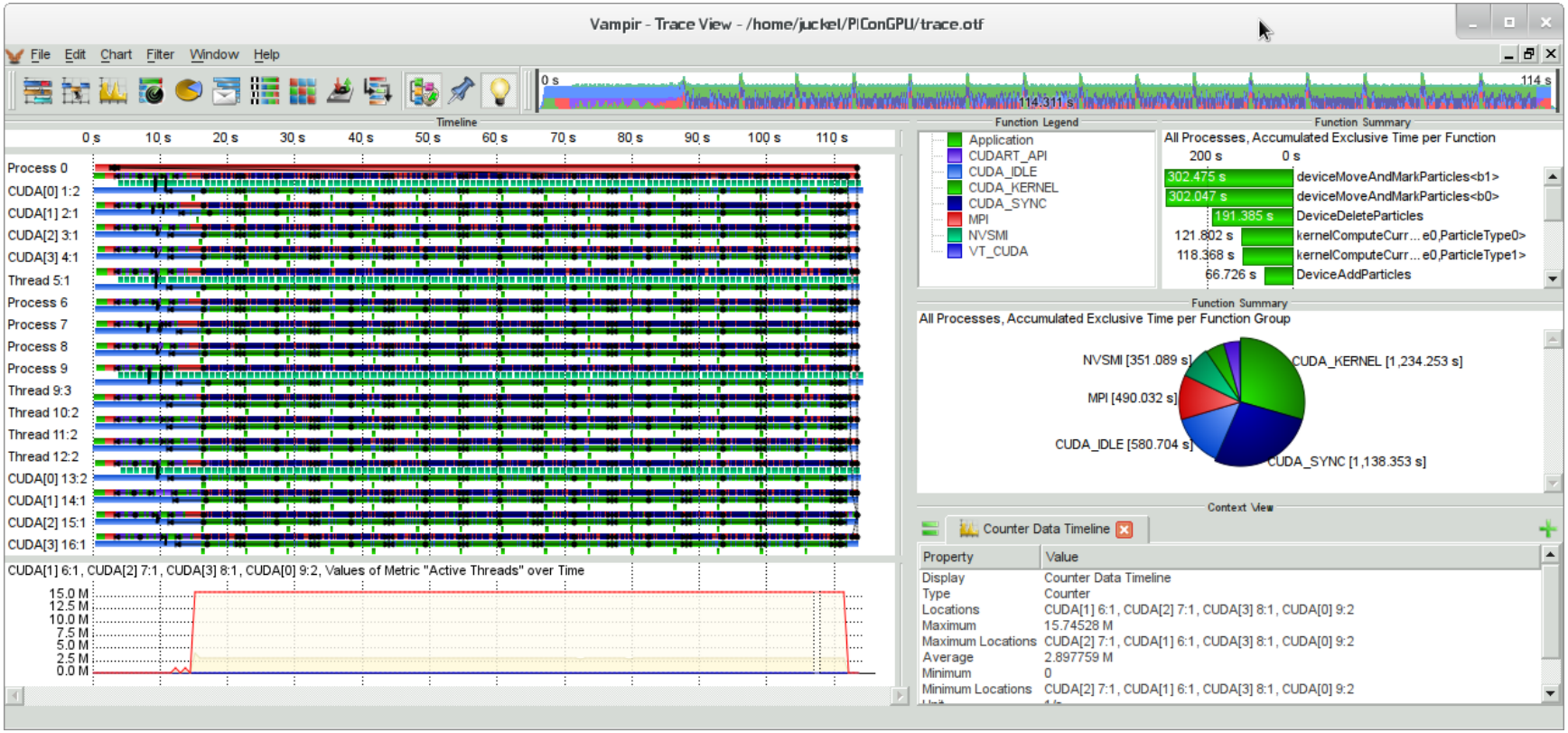
- Introduction
- Vampir displays
- GPGPU support
- Scalability
- Performance Problems
- Conclusions & Preview on Tutorial

GPU Tracing for CUDA



- Support for GPGPU computing with CUDA (OpenCL support in Score-P soon)
- Shows Kernel invocations and idle time (individual DMA transfers shown with Score-P soon)

GPU Tracing and MPI

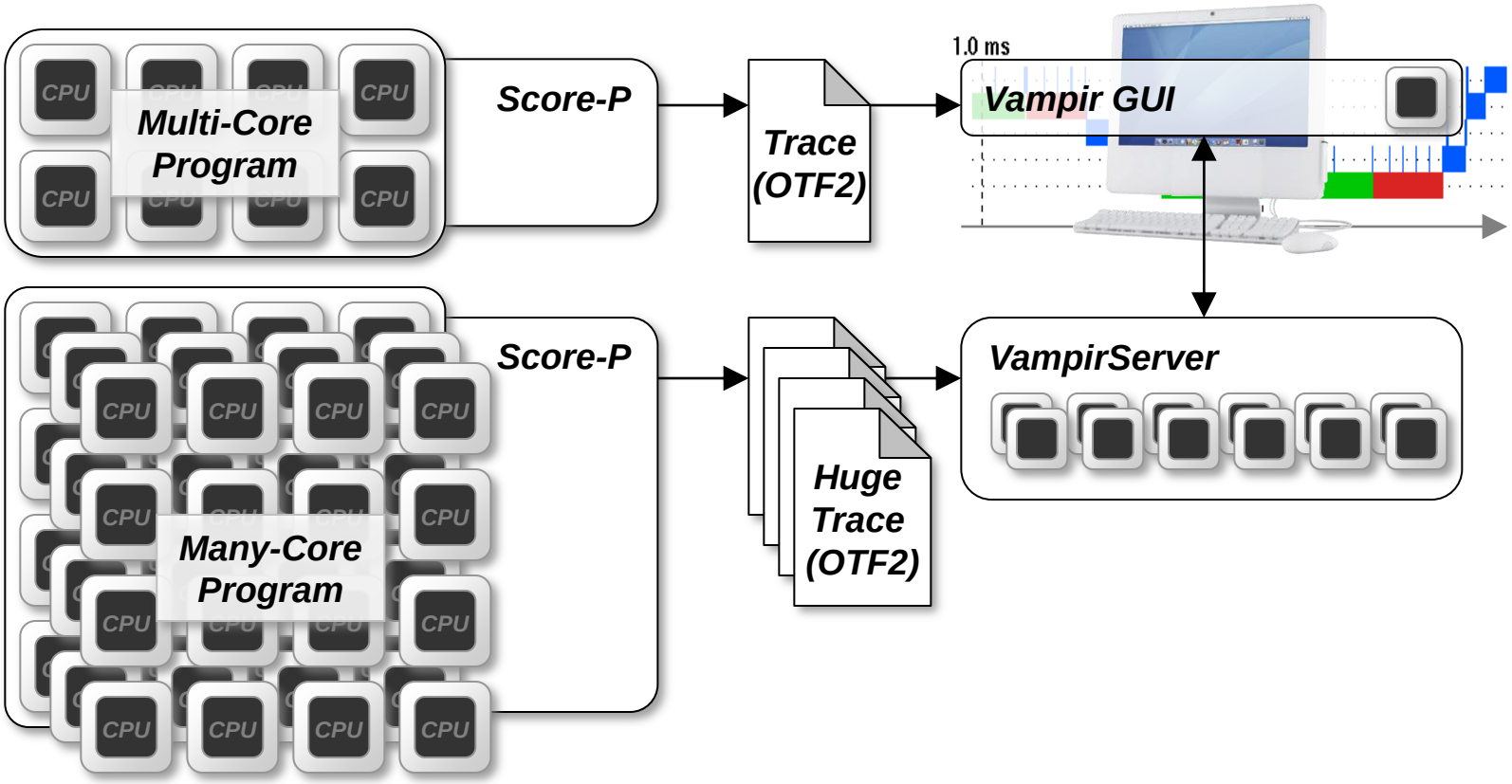


- Also for hybrid MPI+CUDA applications for large scale GPU computing beyond a single compute node

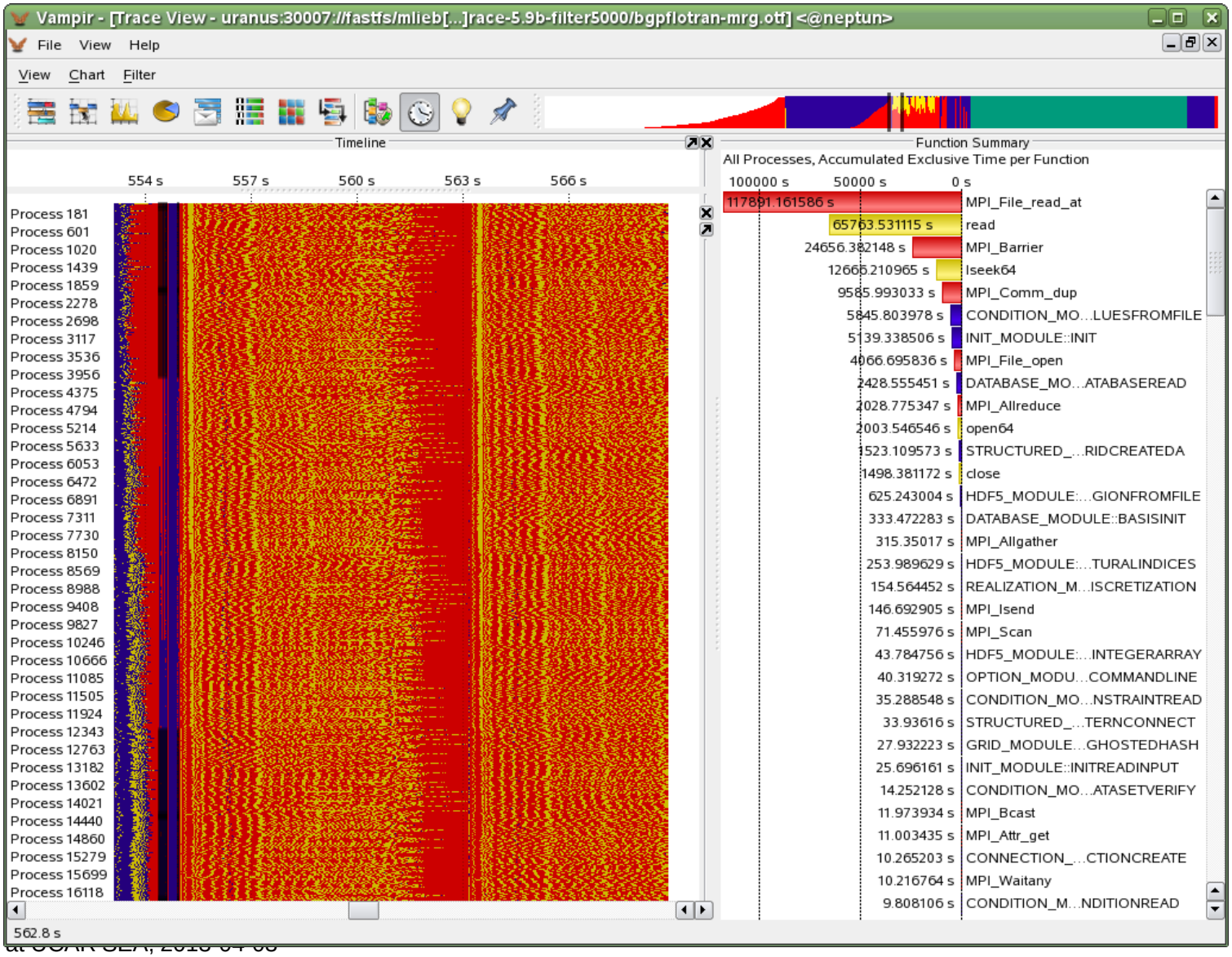
Overview

- Introduction
- Vampir displays
- GPGPU support
- Scalability
- Performance Problems
- Conclusions & Preview on Tutorial

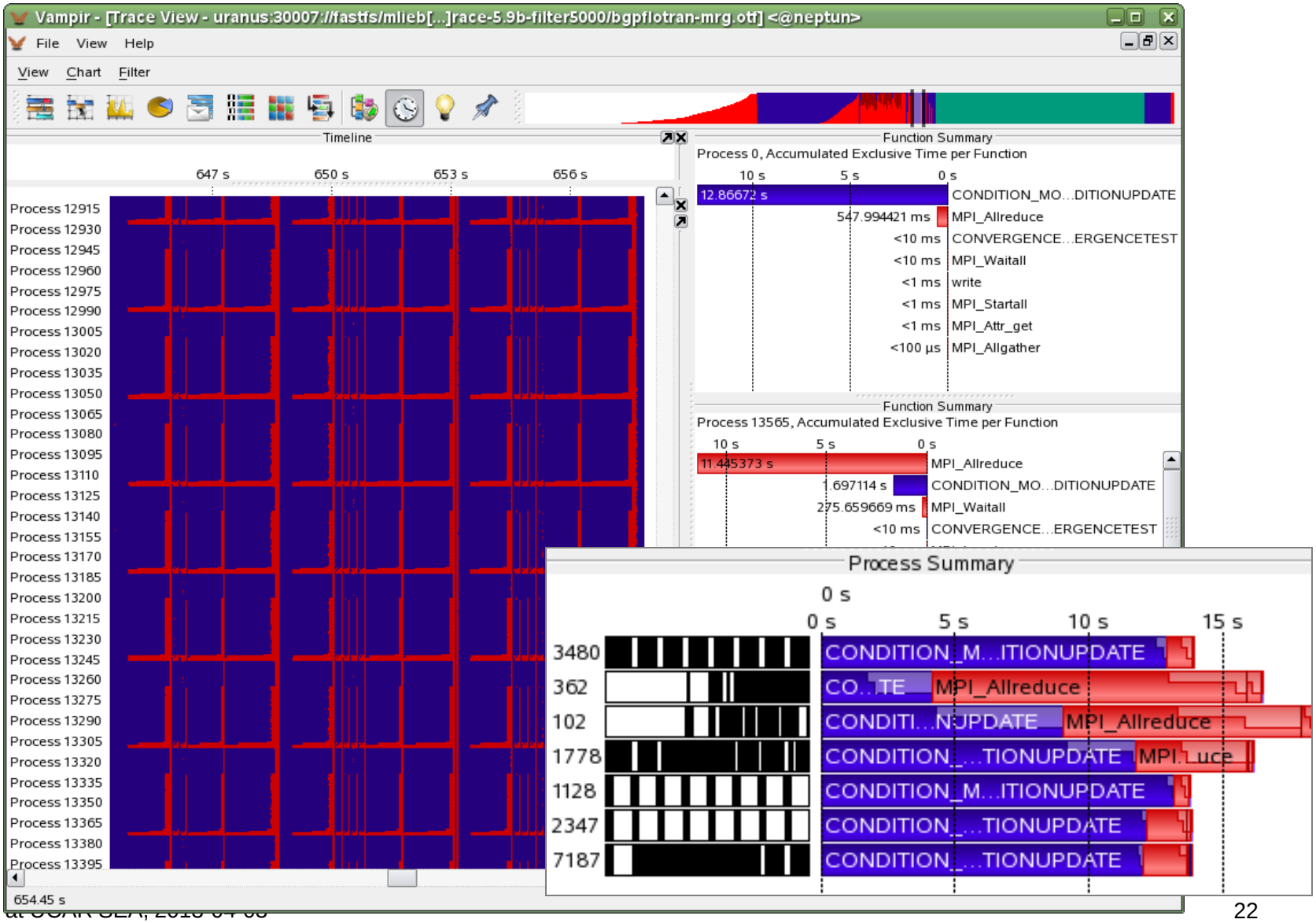
Scalable parallel Vampir visualization architecture



Vampir's Scalability: Full View



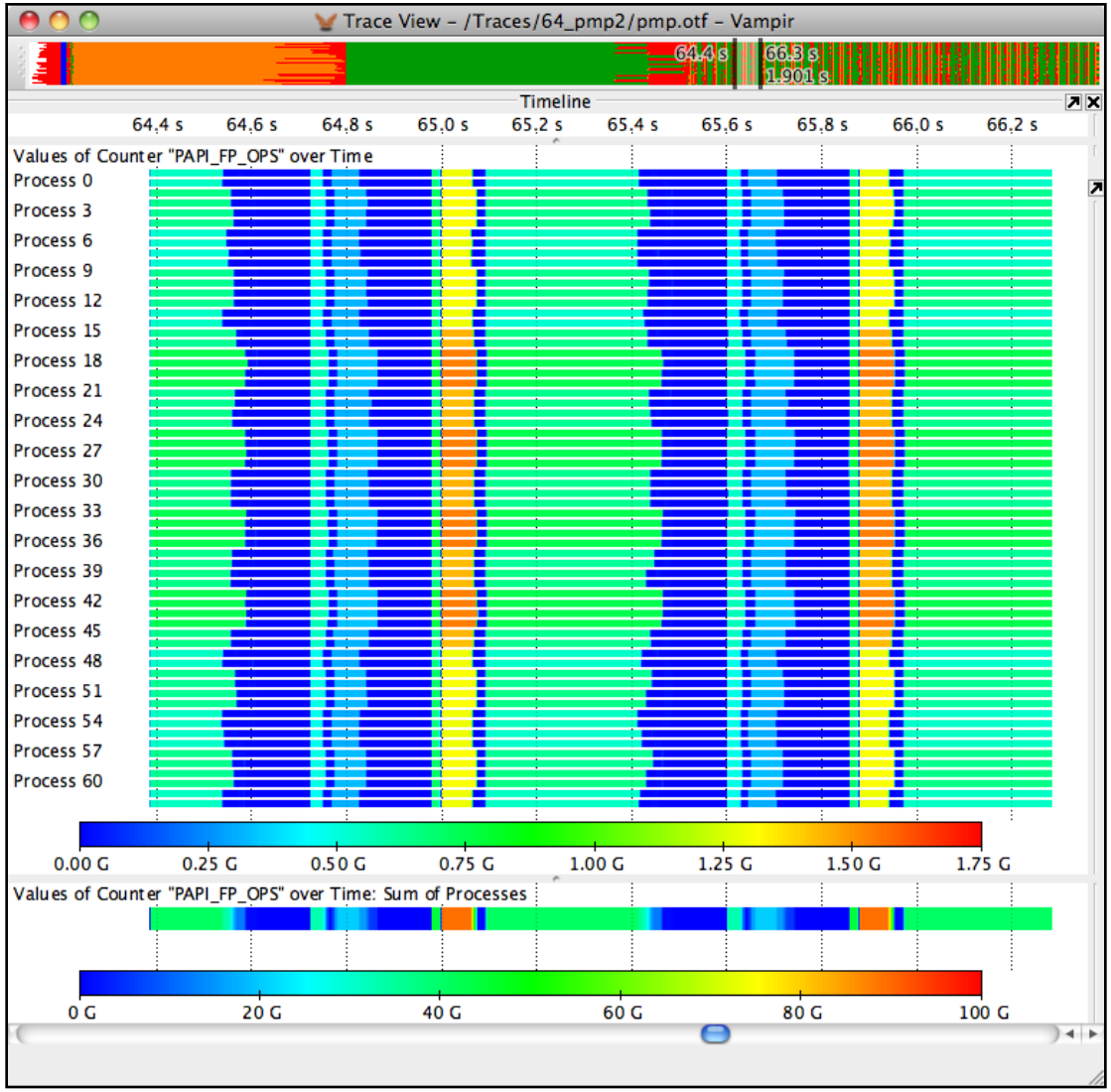
Vampir's Scalability: Process Clustering



Vampir's Scalability: Performance Radar

Identification of relevant spots with a heat map:

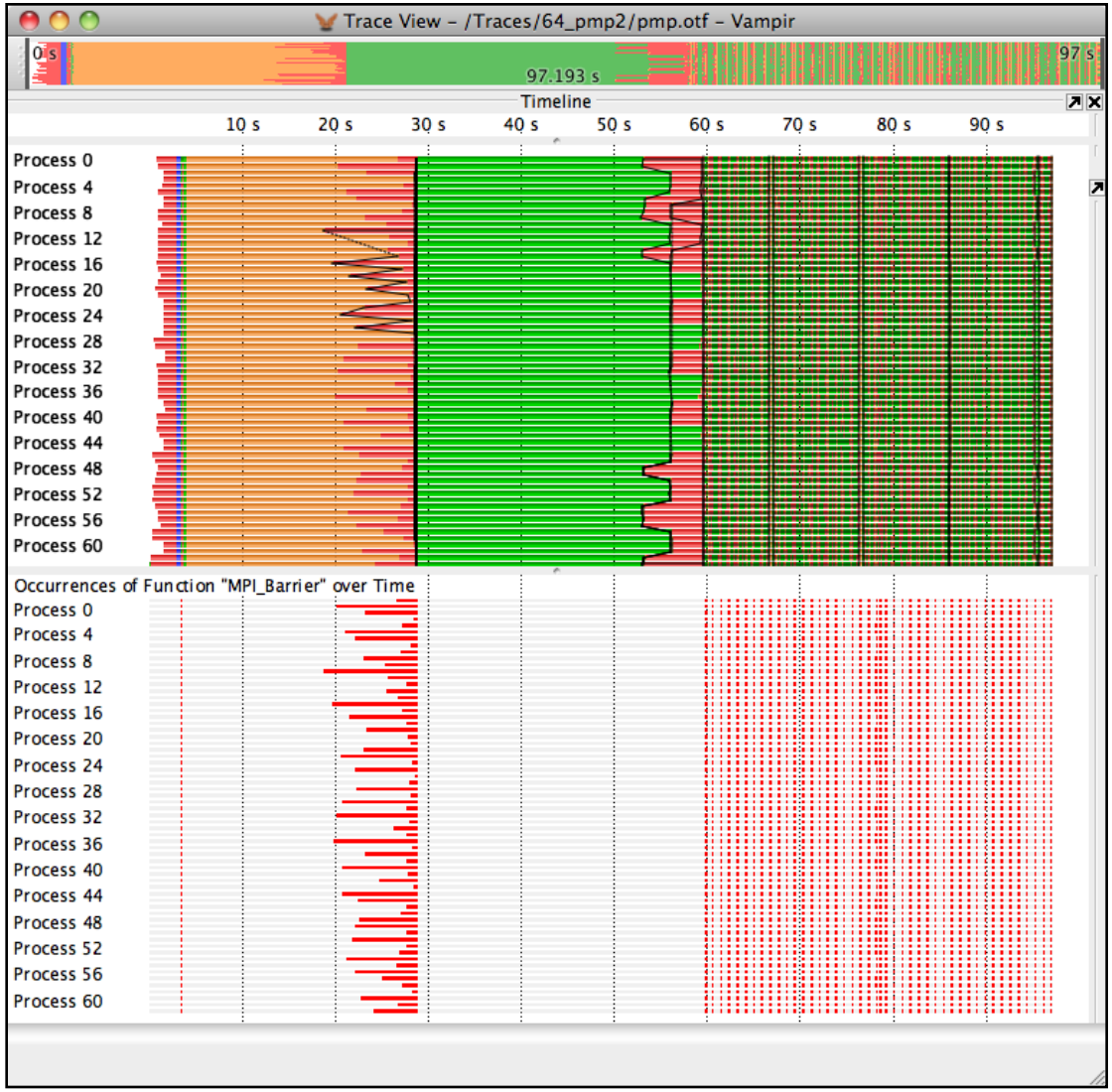
- Hardware performance counters
- Metrics derived from events, e.g. frequency of function calls, here MPI_Barrier



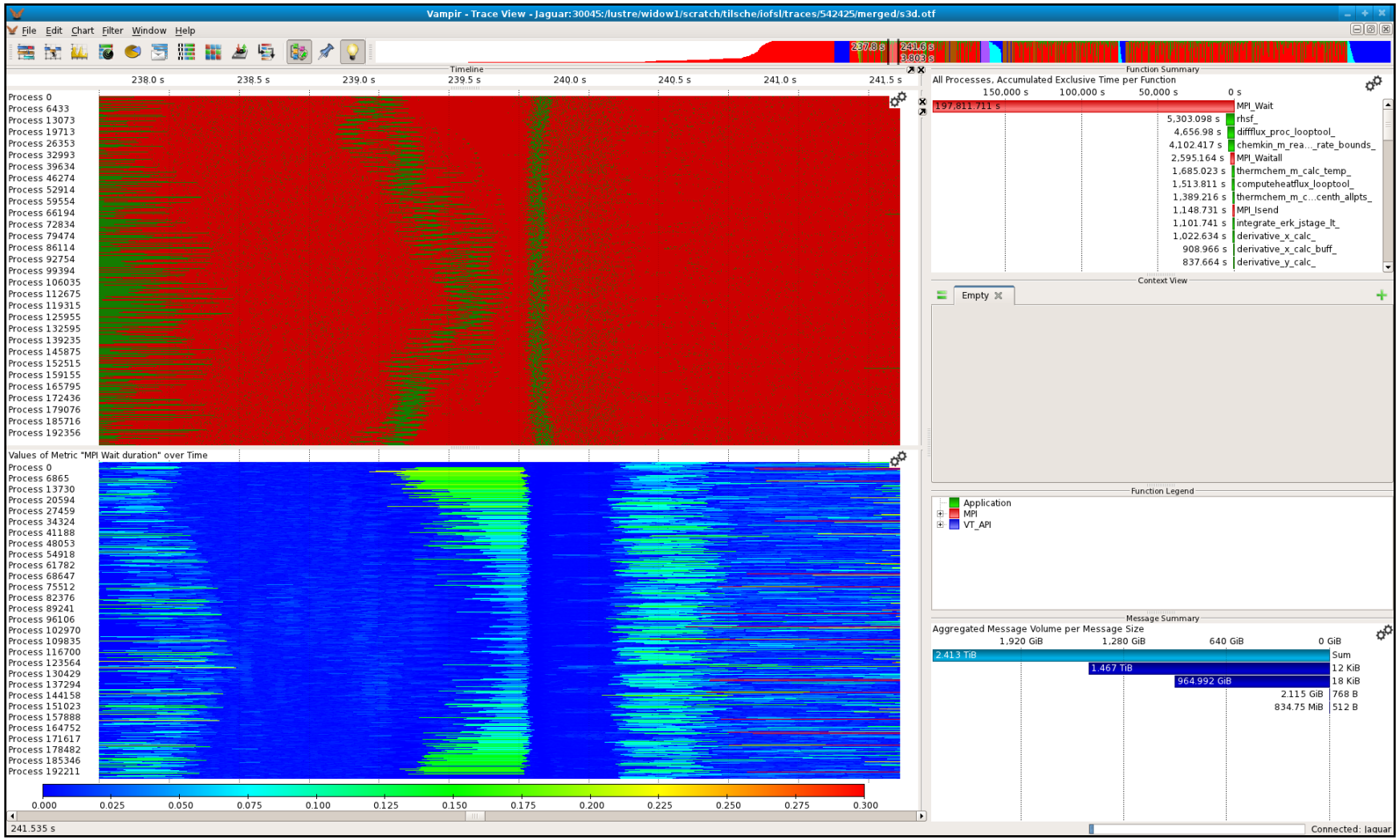
Vampir's Scalability: Performance Radar

Identification of relevant spots with a heat map:

- Hardware performance counters
- Metrics derived from events, e.g. frequency of function calls, here MPI_Barrier



Scalability: 200,000+ Processes in Vampir



Overview

- Introduction
- Vampir displays
- GPGPU support
- Scalability
- Performance Problems
- Conclusions & Preview on Tutorial

Types of performance problems:

- Serial vs. parallel performance
- Bottlenecks relatted to MPI, OpenMP, CUDA, …
- Computation vs. communication
- Program phases: initialization/finalization, iteration, I/O, checkpointing, …

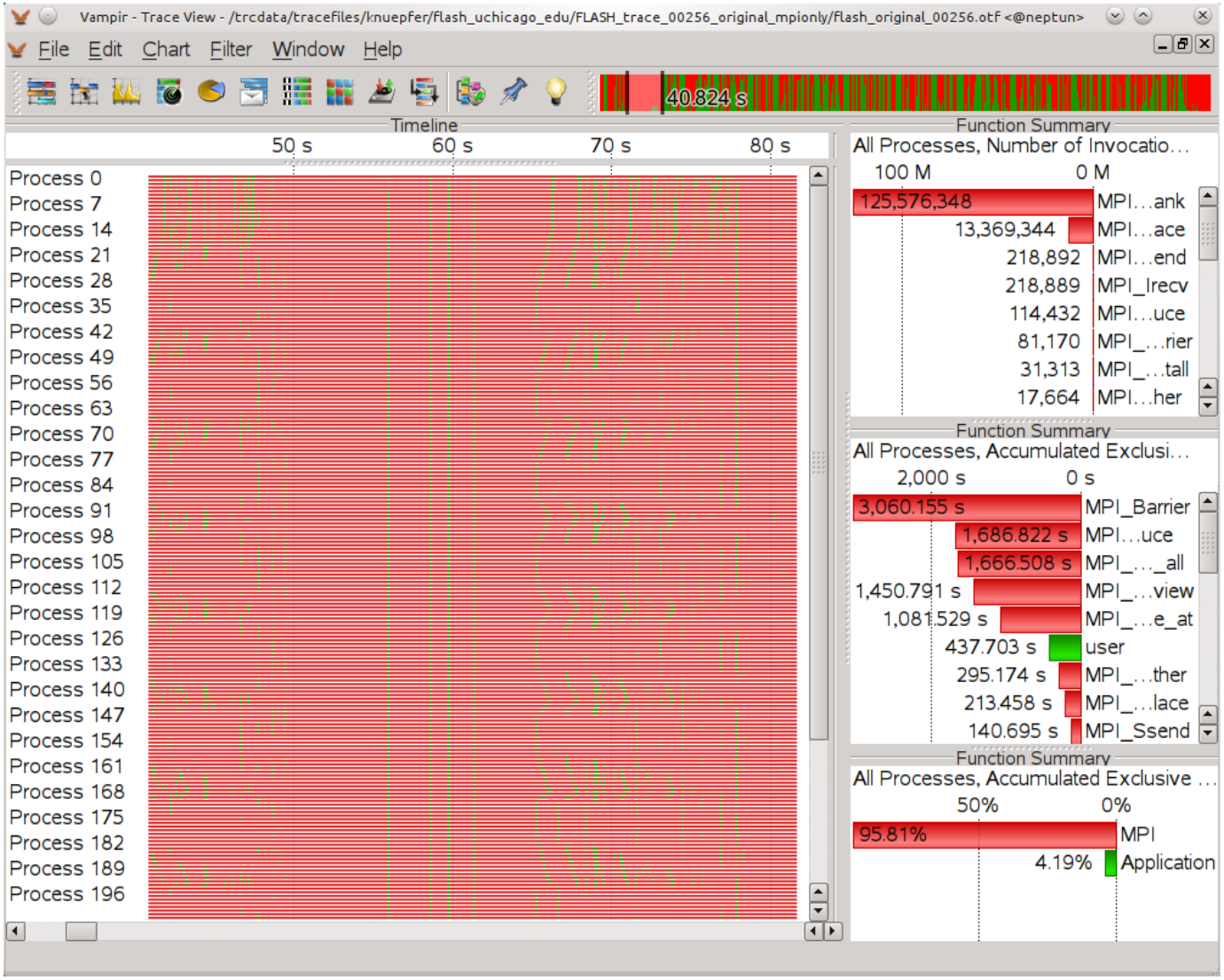
General strategy:

- Find highest consumer(s) of run-time
- Start with hypothesis about most severe performance problem, then confirm or disprove

Some examples (more in tutorial parts):

Problems with MPI

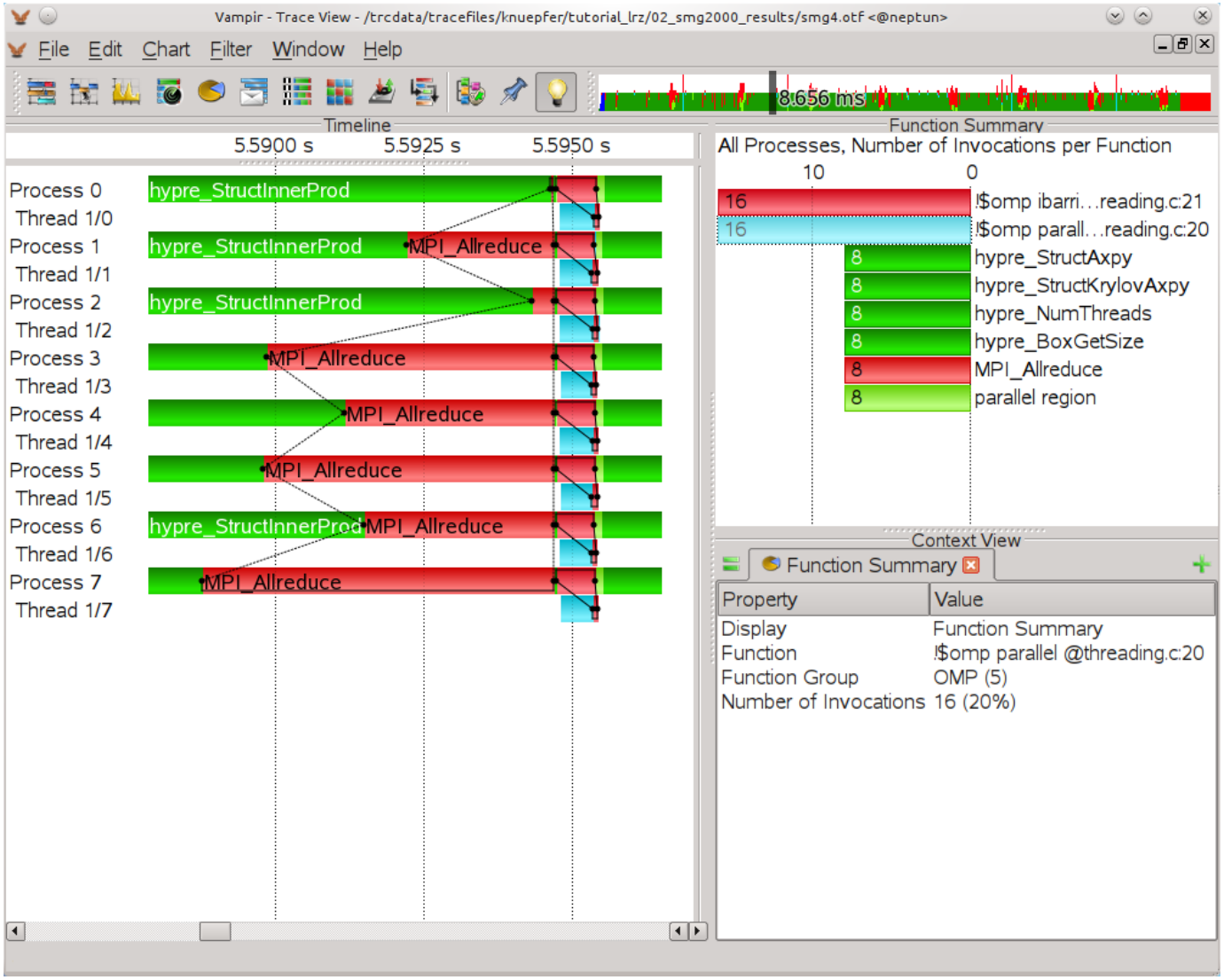
Too much runtime in MPI all-together



Problems with MPI + OpenMP

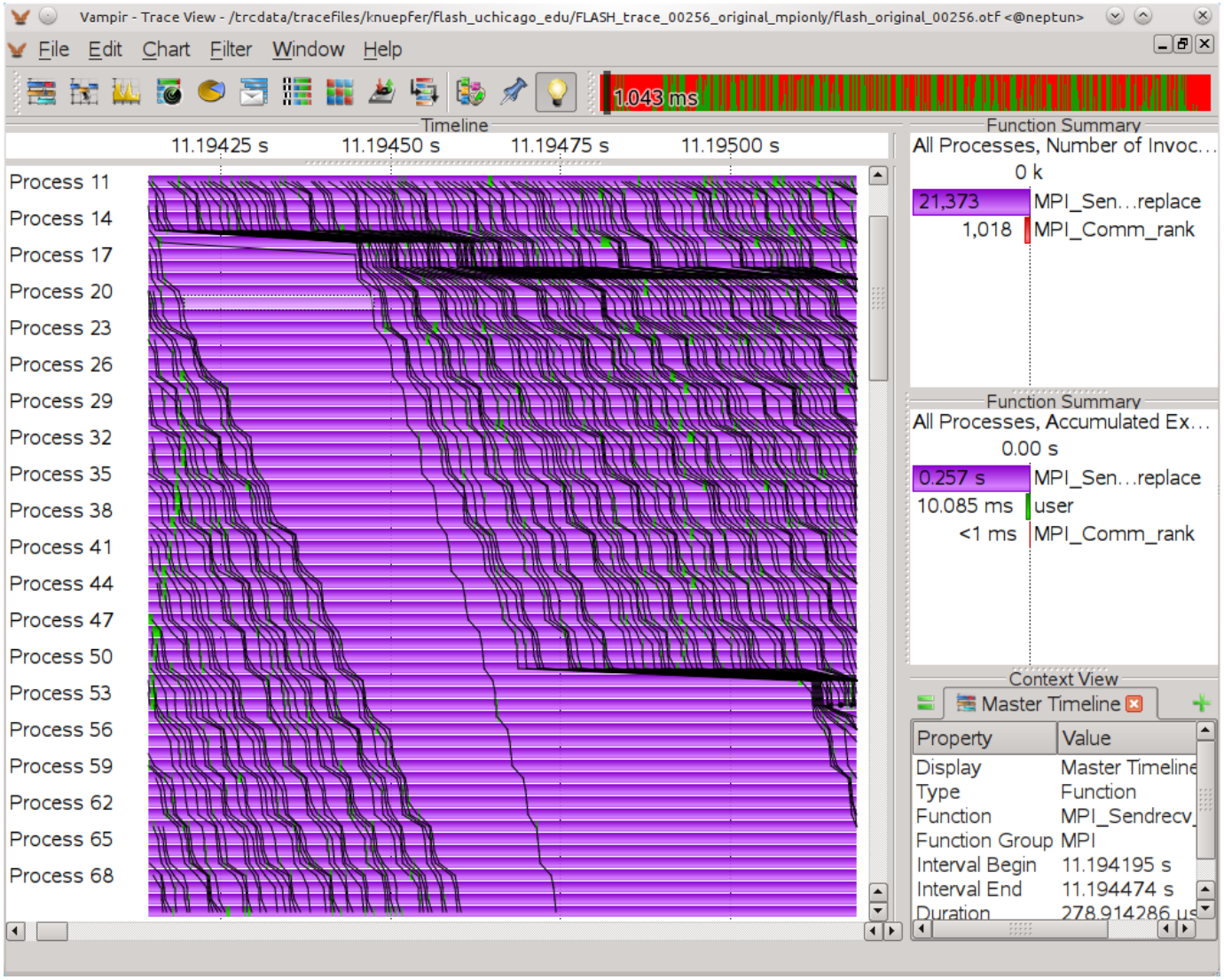
Imbalanced computation causes MPI waiting time

OpenMP threads mostly idle



Problems with strange MPI patterns

Strange MPI patterns propagate delays over successive MPI ranks

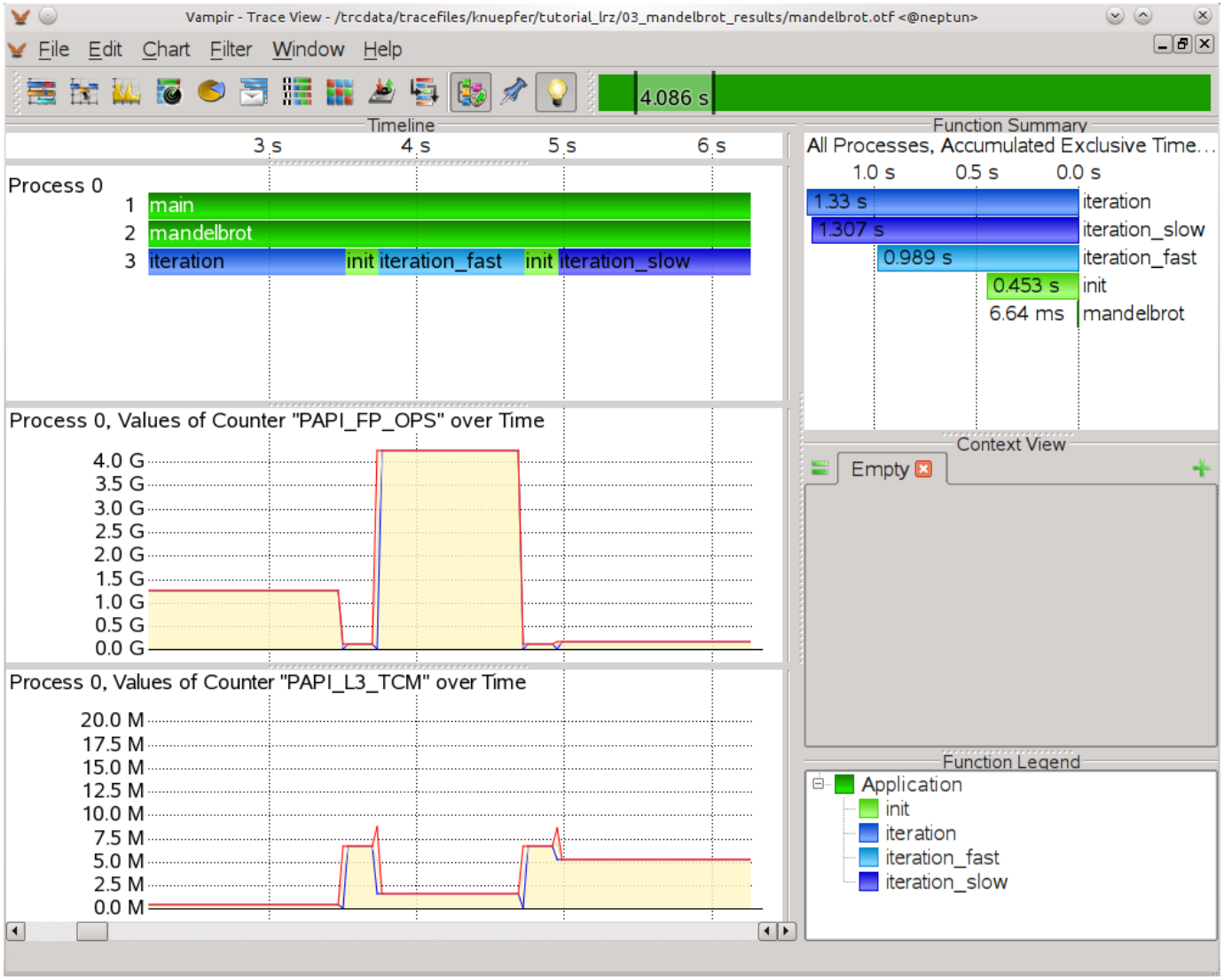


Problems for serial performance

High FP rate, low L3 cache miss rate

vs.

low FP rate due to a high L3 cache miss rate

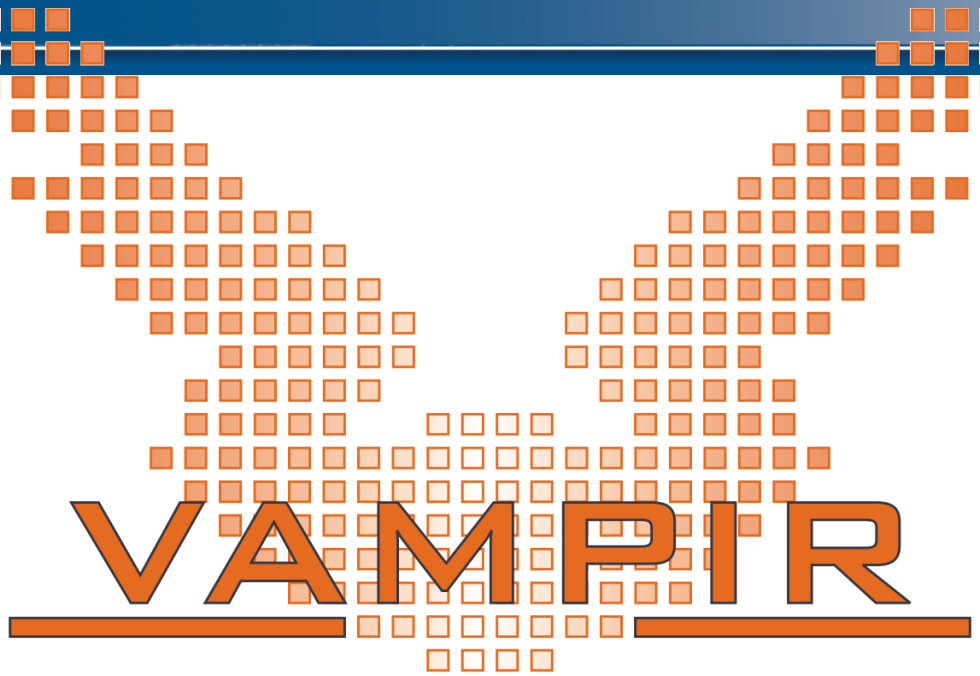


Conclusions:

- User-friendly and comprehensible visualization of complex and highly parallel run-time behavior
- Various aspects of parallel performance
- Highly scalable by itself

Tutorial on Thursday & Friday:

- Run-time measurement with Score-P + hands-on
- Analysis with Vampir, Scalasca, TAU + hands-on
- Combination of Vampir and Scalasca
- Work with your code and tool(s) of your choice
- GPGPU measurement and analysis (on request)



Matthias Jurenz, Andreas Knüpfer, Ronny Brendel, Matthias Lieber,
Jens Doleschal, Jens Domke, Holger Mickler, Daniel Hackenberg,
Michael Heyde, Thomas Ilsche, Guido Juckeland, Robert Dietrich,
Frank Winkler, Michael Kluge, Matthias Müller, Holger Brunst,
Ronald Geisler, Reinhard Neumann, Bert Wesarg, Rene Widera,
Thomas Ilsche, Matthias Weber, Thomas William,
Hartmut Mix, and Wolfgang E. Nagel

http://www.vampir.eu