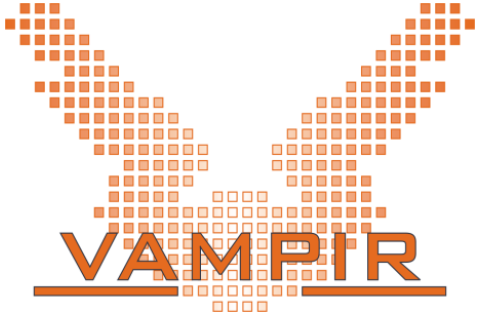
Interactive visualization and time-interval statistics with Vampir

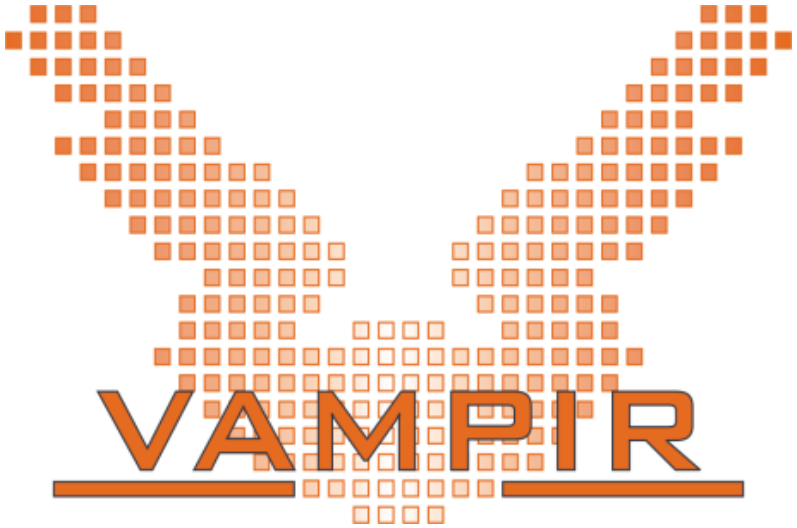
Bert Wesarg
Technische Universität Dresden / GWT-TUD GmbH



GWT Gesellschaft für Wissens- und Technologietransfer

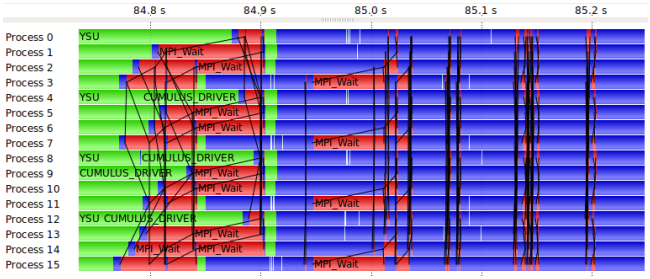
Outline

- **Part I: Welcome to the Vampir Tool Suite**
  - Mission
  - Event Trace Visualization
  - Vampir & VampirServer
- **Part II: Vampir Hands-On**
  - Visualizing and analyzing BT-MZ
- **Part III: Vampir Demos**



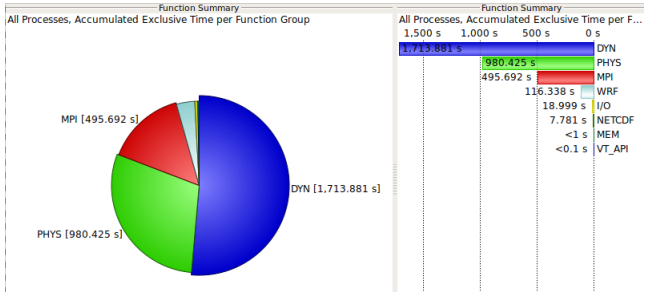
Event Trace Visualization with Vampir

- Alternative and supplement to automatic analysis
- Show dynamic run-time behavior graphically at any level of detail
- Provide statistics and performance metrics



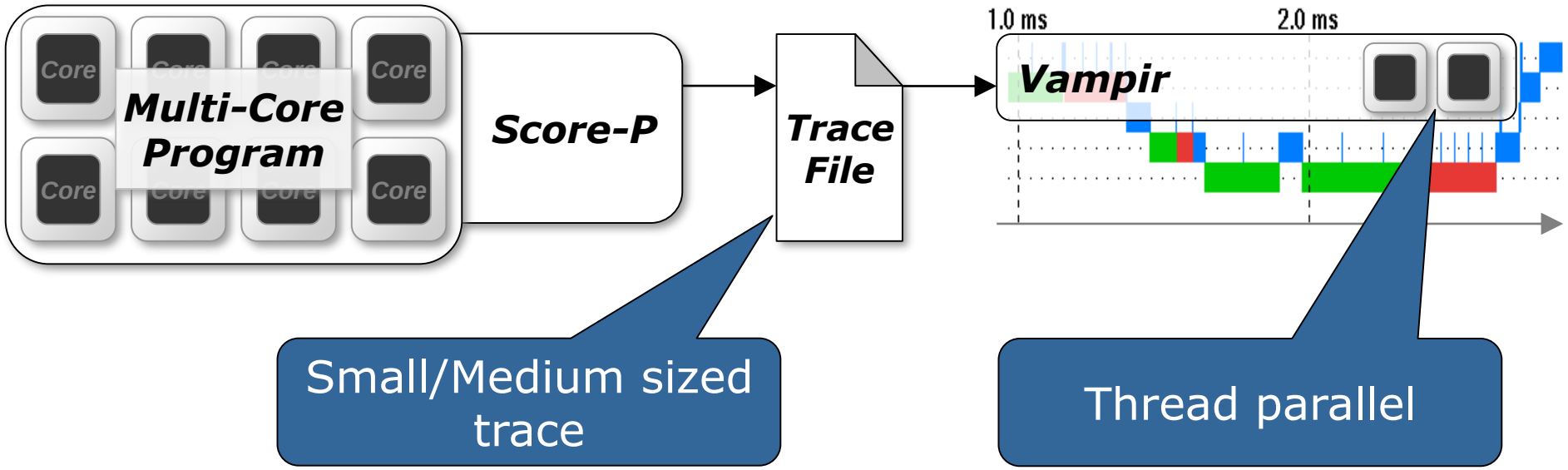
- **Timeline charts**
  - Show application activities and communication along a time axis

- **Summary charts**
  - Provide quantitative results for the currently selected time interval



Visualization Modes (1)
Directly on front end or local machine

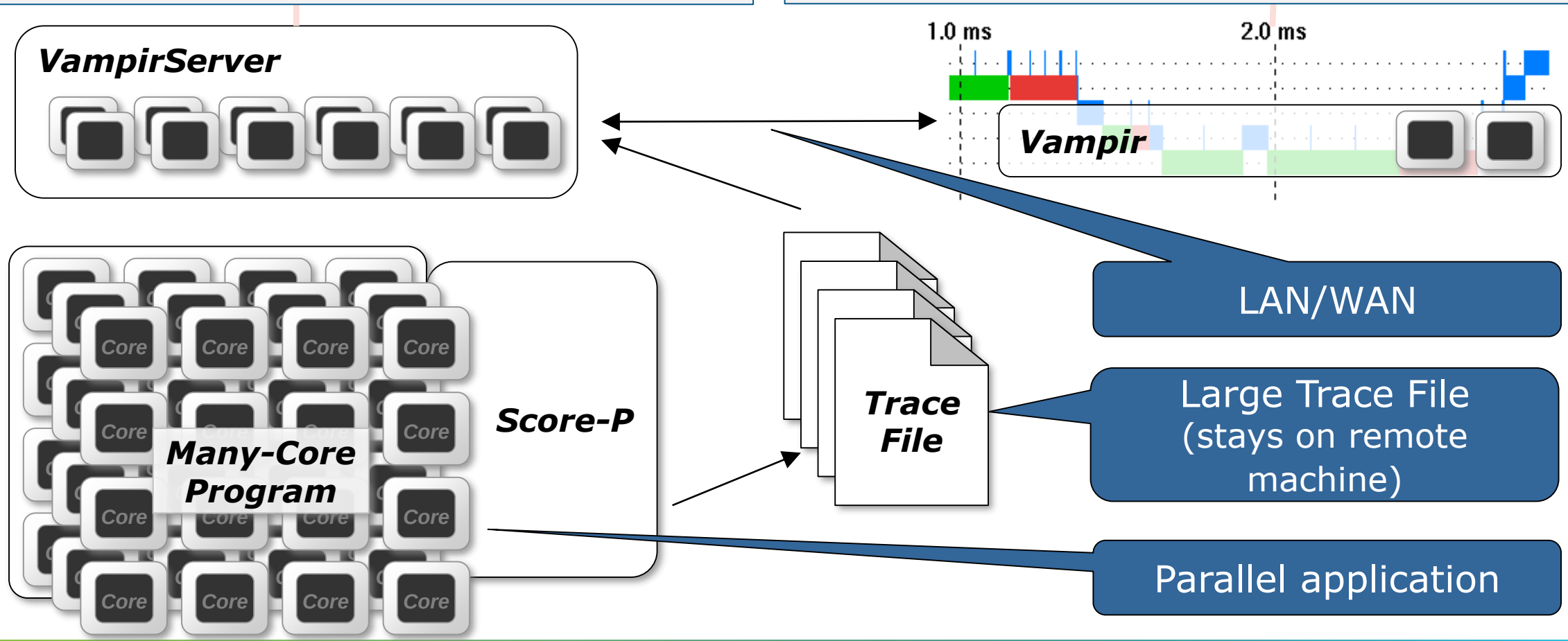
```
% ml vampir
% vampir
```



Visualization Modes (2)
On local machine with remote VampirServer

```
% ml vampir
% vampirserver start
```

```
% ml vampir
% vampir
```



Starting VampirServer

```
USAGE
    vampirserver [SUBCOMMAND] [ARGUMENTS ...] [-- [CUSTOM-ARGUMENTS ...]]

SUBCOMMANDS
    list, ls  [servers | launchers]
        List server related information. Currently, this command lists all
        active servers or the available launch scripts (launchers). If no
        argument is provided, all active servers are listed.
…
    start, up  [-n] [-p] [-t] [LAUNCHER] [-- LAUNCHER-ARGUMENTS...]
        Start a new server instance. LAUNCHER identifies the launch script to
        be used. LAUNCHER defaults to "slurm".

        -n, --ntasks=TASKS      set the number of analysis tasks
        -t, --timeout=SECONDS    set the startup timeout to SECONDS seconds
        Try 'LAUNCHER -- --help' for launcher specific arguments.
…
    stop, ex  [SERVER_ID]
        Stop the given server or the most recent server if no SERVER_ID is
        provided. The server ID is printed during startup. Alternatively, use
        the list command to print a list of available servers.
```

- Account for one extra task: launcher script starts TASKS+1 MPI processes

Starting VampirServer: SLURM launcher

```
% vampirserver start slurm -- --help
…
Launcher usage: slurm -- [--time=TIME] [-- SALLOC-ARGUMENTS...]
    -h, --help          show this little help
    -t, --time=TIME     total run time of the allocation

    -- SALLOC-ARGUMENTS...
                        remaining arguments are passed directly to salloc
```

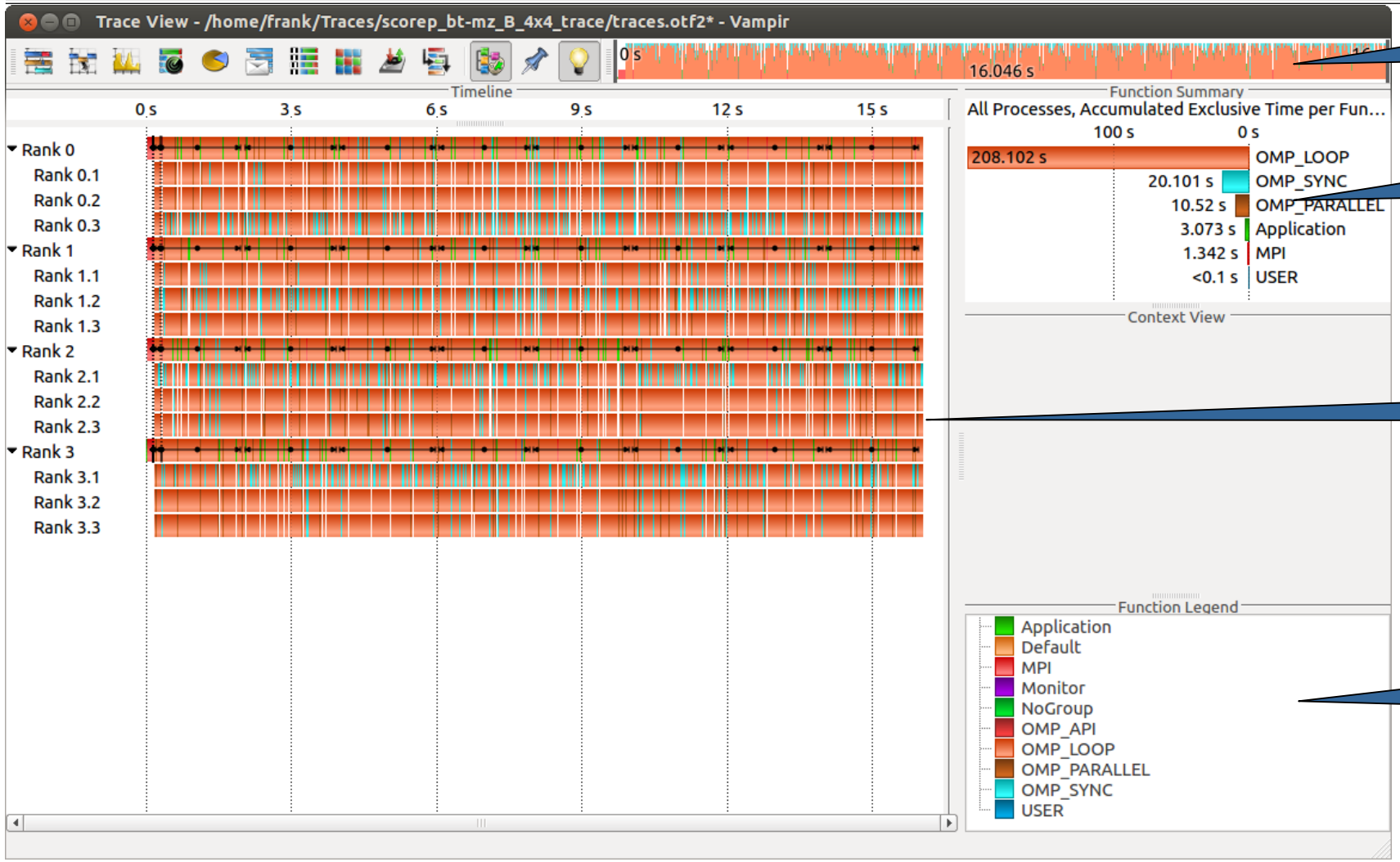
Starting VampirServer

```
% vampirserver start -n 31 \
    -- --time=3:00:00 \
       -- -A $SBATCH_ACCOUNT --reservation=$SBATCH_RESERVATION \
          -N 1 -c 2 --mem=0
Launching VampirServer...
Submitting slurm 3:00:00 minutes job (this might take a while)...
salloc: Pending job allocation 3208476
…
salloc: Nodes n1589 are ready for job
VampirServer 10.4.1 Professional (271537cd)
Licensed to ZIH, TU Dresden
Running 51 analysis processes... (abort with vampirserver stop 29603)
VampirServer <29603> listens on: <host>:30059

% vampirserver list
29603 <host>:30059 [31x, slurm]
% vampirserver stop 29603
Shutting down VampirServer <29603>...
Disconnecting client: <host>:30059
VampirServer <29603> is down.
```

- Use host and port for SSH port forwarding

Visualization of the NPB-MZ-MPI / BT trace



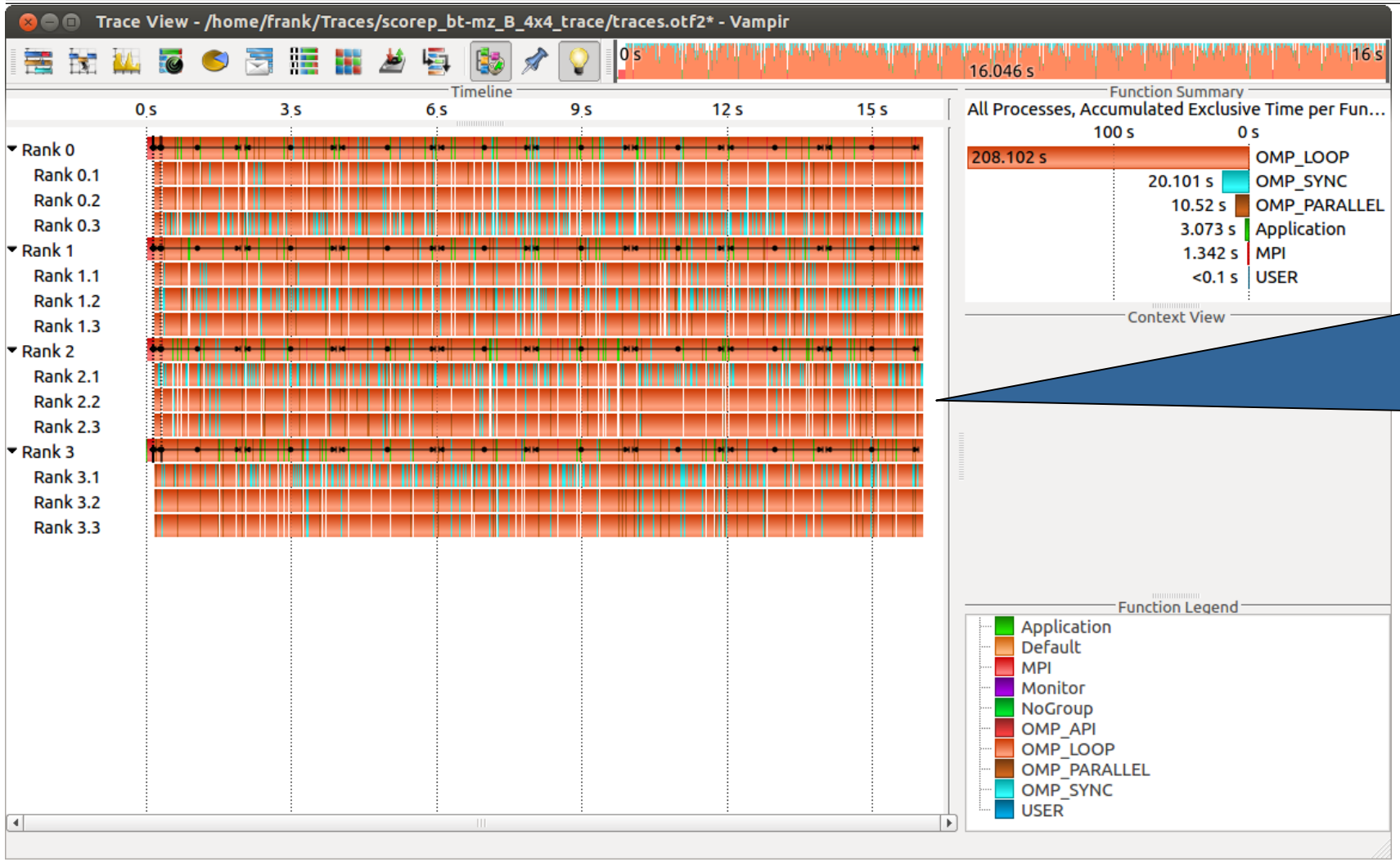
Navigation Toolbar

Function Summary

Master Timeline

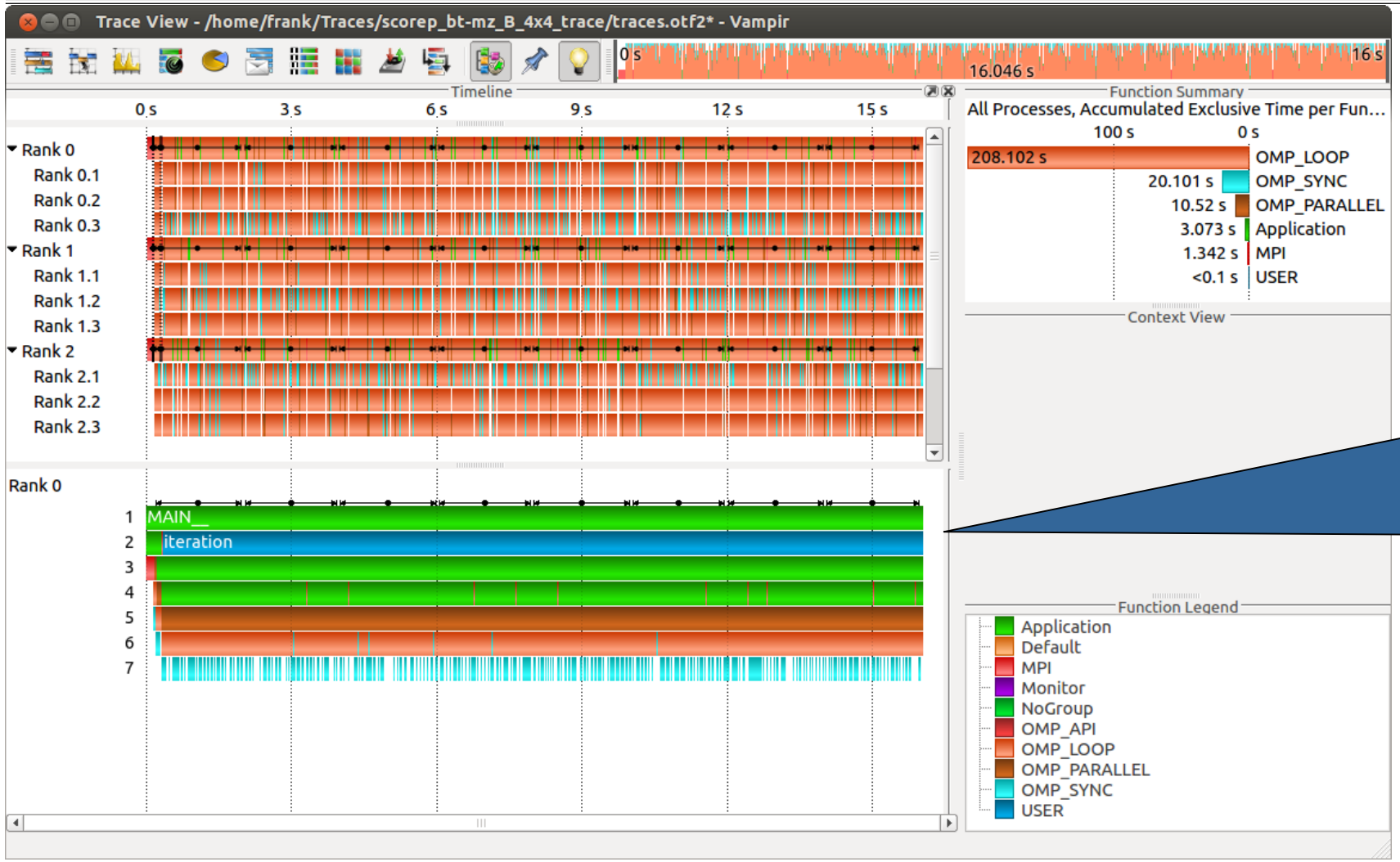
Function Legend

Visualization of the NPB-MZ-MPI / BT trace
Master Timeline



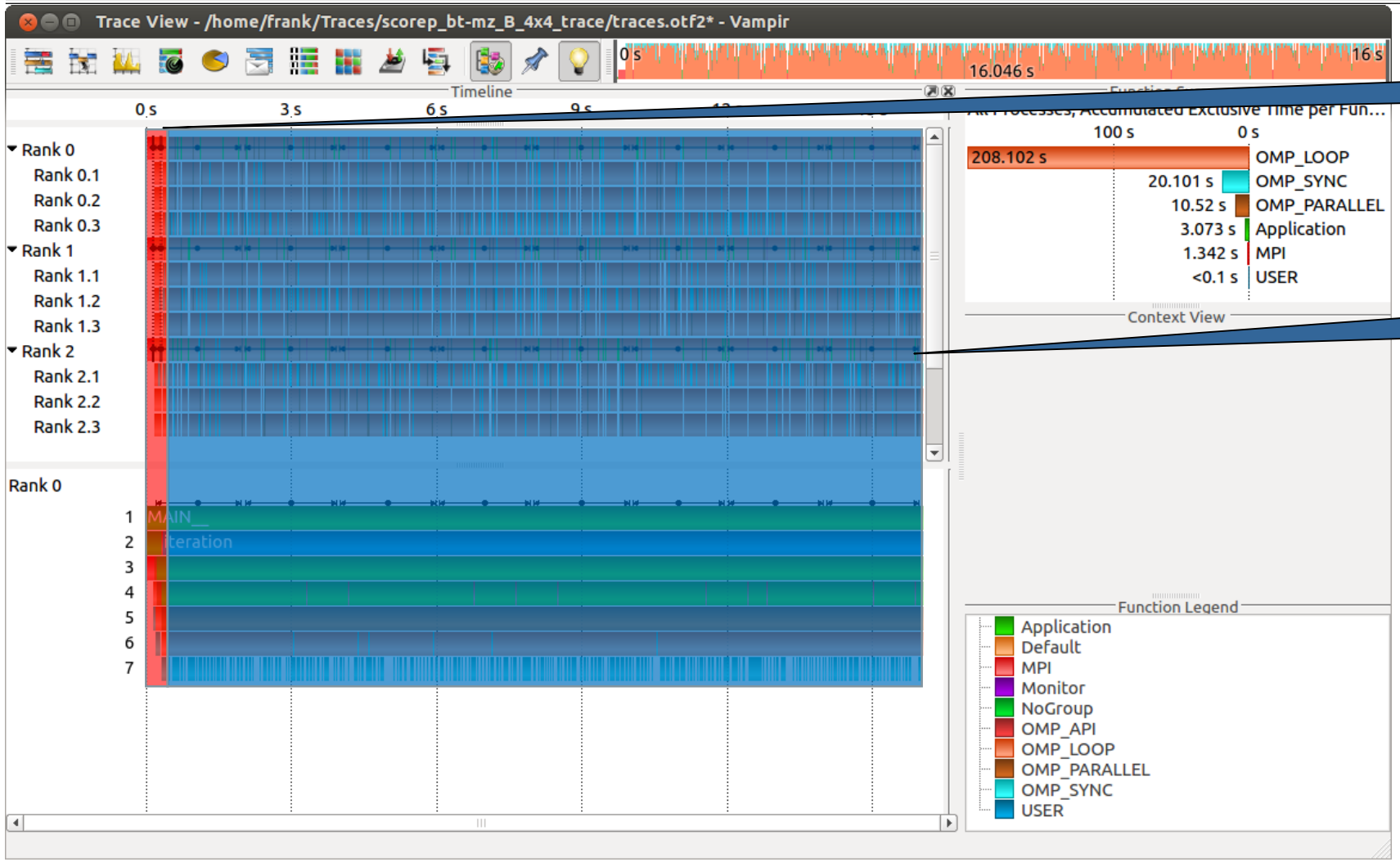
Detailed information about functions, communication and synchronization events for collection of processes.

Visualization of the NPB-MZ-MPI / BT trace
Process Timeline



Detailed information about different levels of function calls in a stacked bar chart for an individual process.

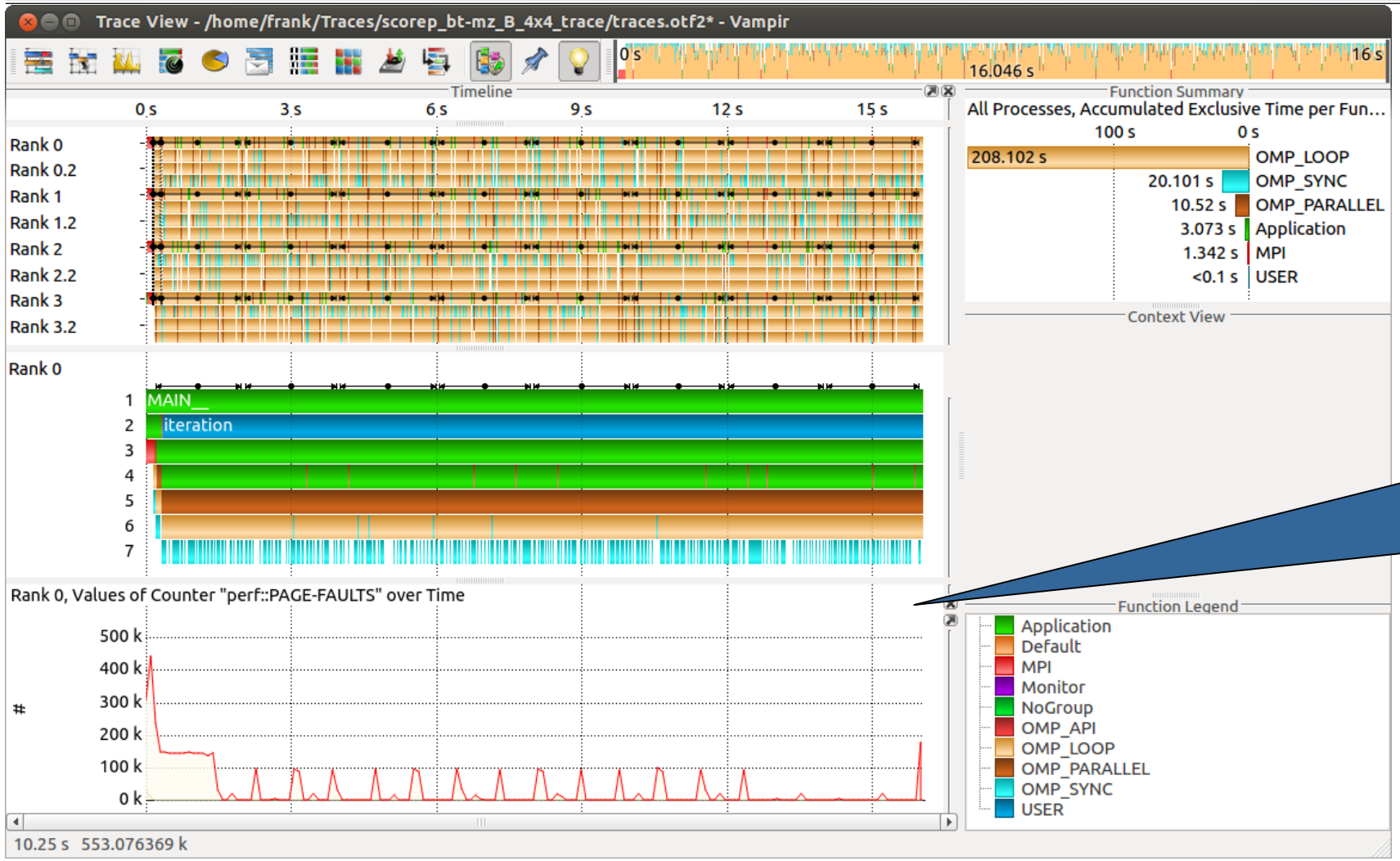
Visualization of the NPB-MZ-MPI / BT trace
Typical program phases



Initialisation Phase

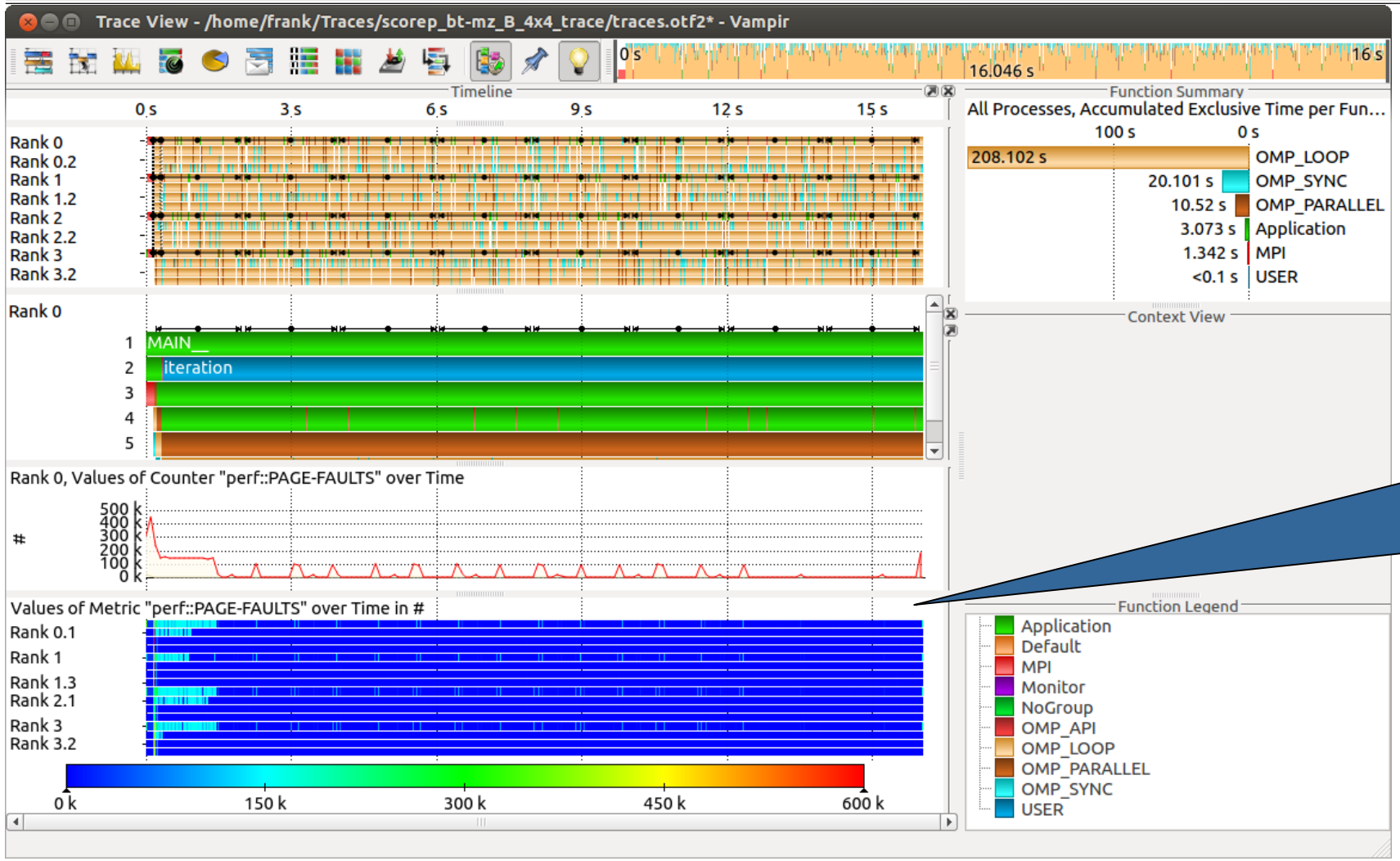
Computation Phase

Visualization of the NPB-MZ-MPI / BT trace
Counter Data Timeline



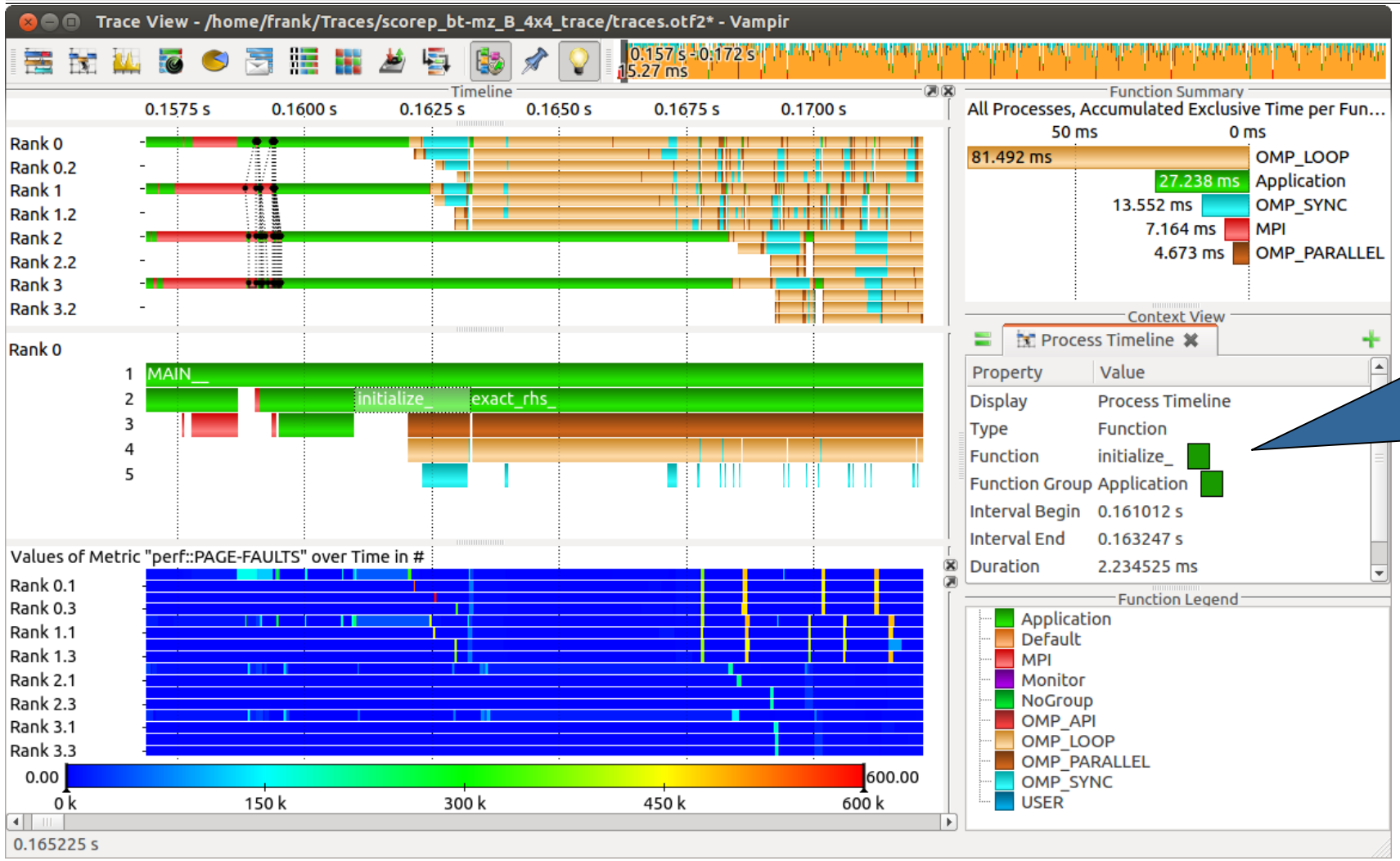
Detailed counter information over time for an individual process.

Visualization of the NPB-MZ-MPI / BT trace
Performance Radar



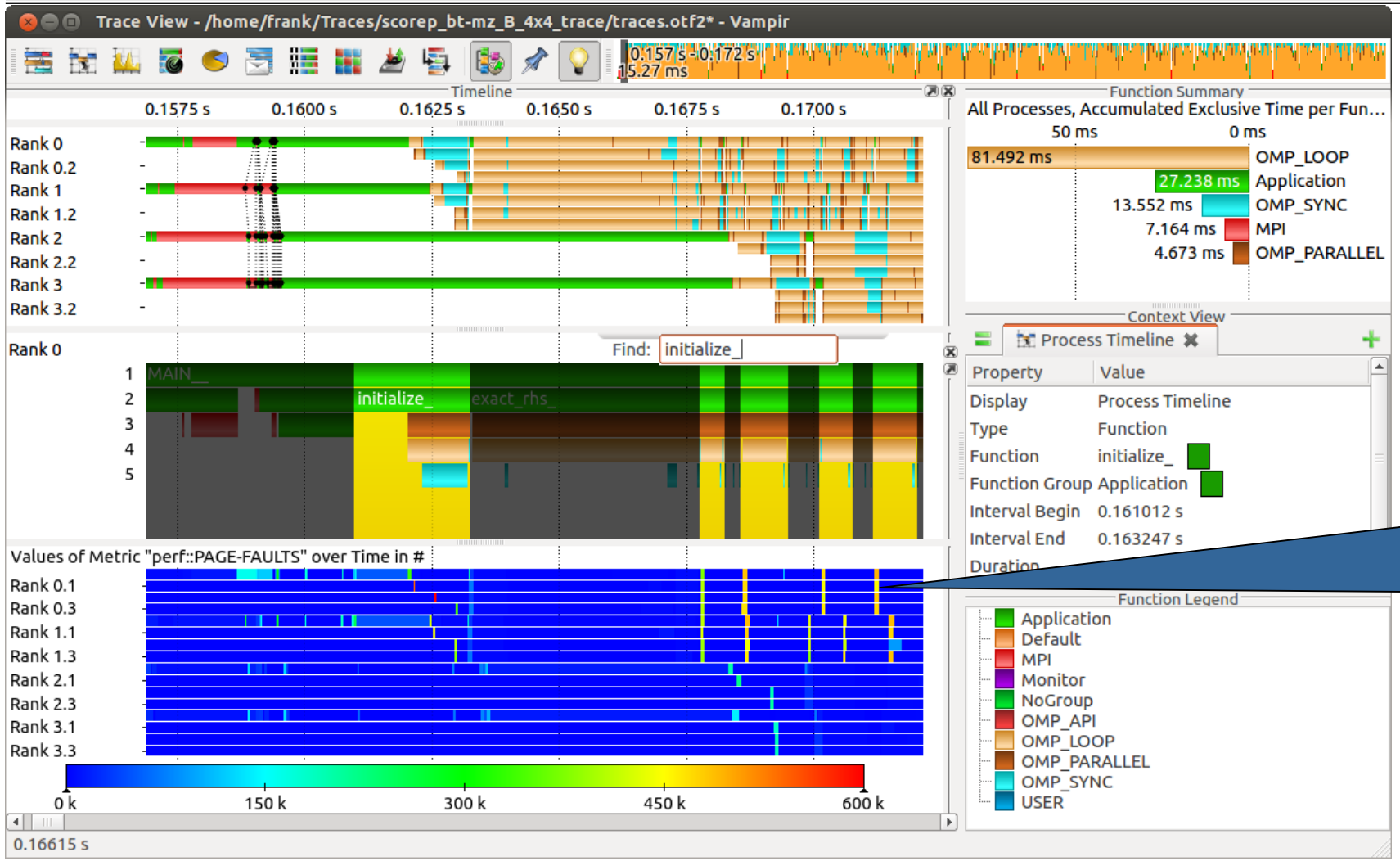
Detailed counter information over time for a collection of processes.

Visualization of the NPB-MZ-MPI / BT trace
Zoom in: Inititialisation Phase



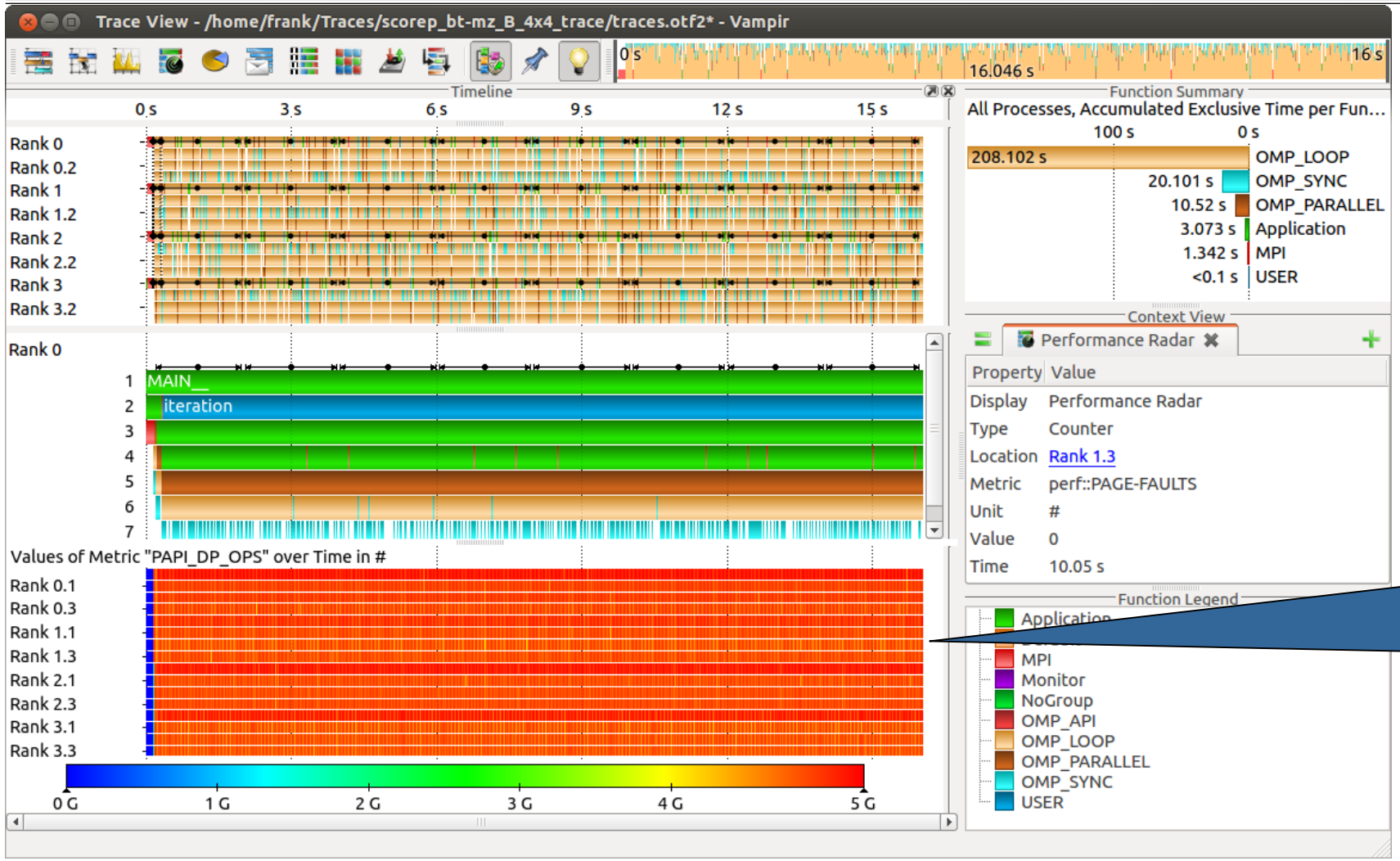
Context View: Detailed information about function “initialize_”.

Visualization of the NPB-MZ-MPI / BT trace
Find Function



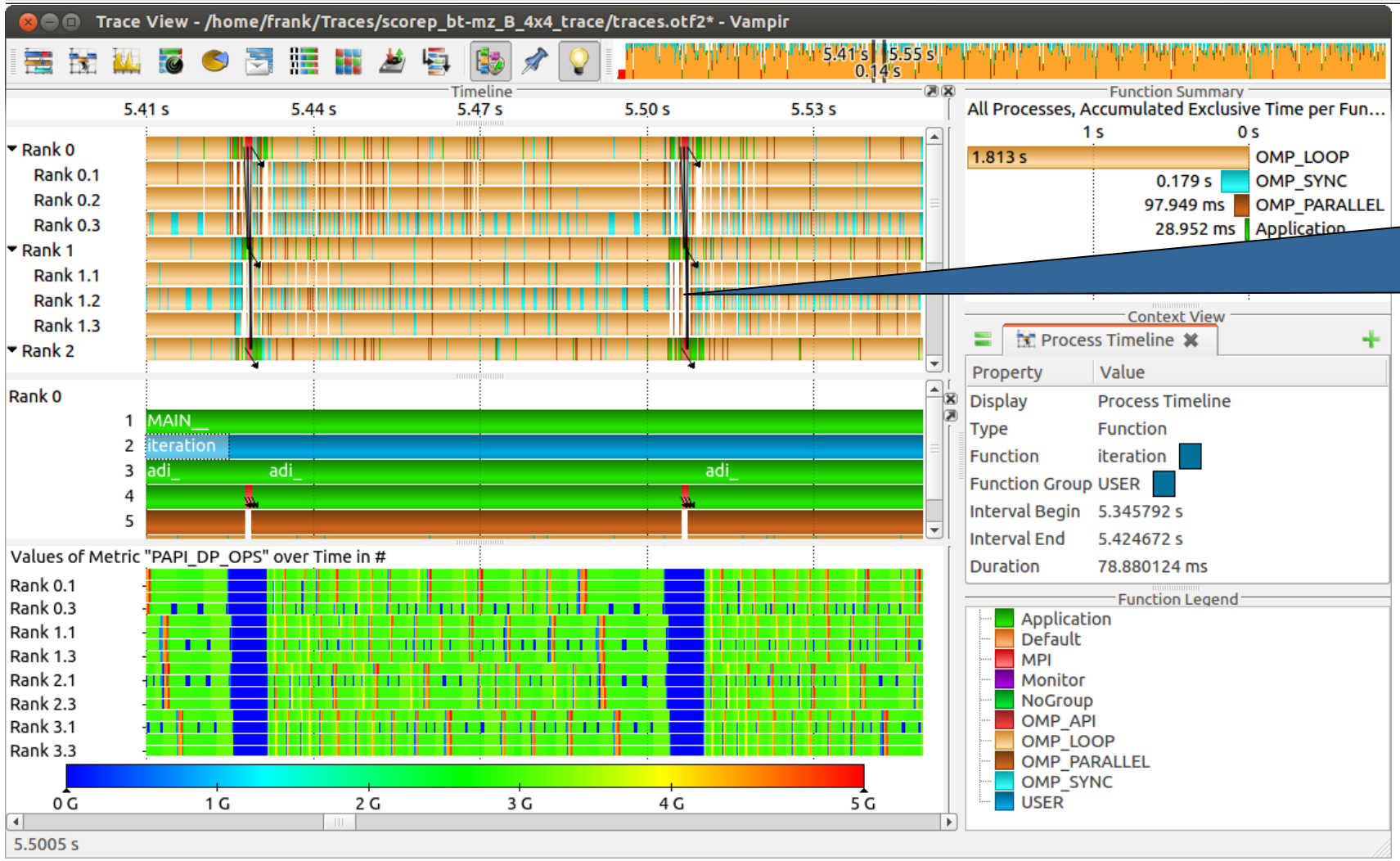
Execution of function "initialize_" results in higher page fault rates.

Visualization of the NPB-MZ-MPI / BT trace
Computation Phase



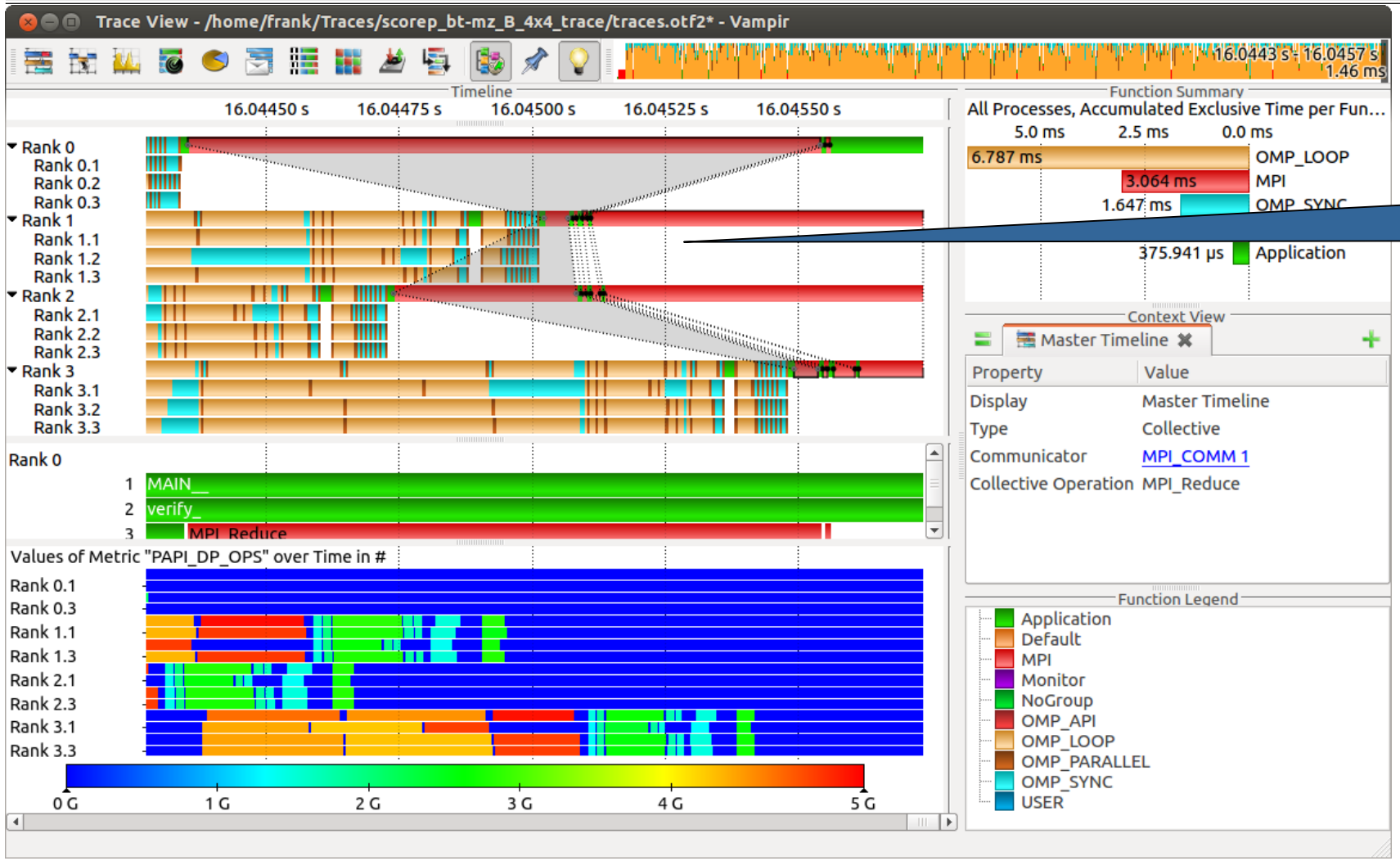
Computation phase results in higher floating point operations.

Visualization of the NPB-MZ-MPI / BT trace
Zoom in: Computation Phase



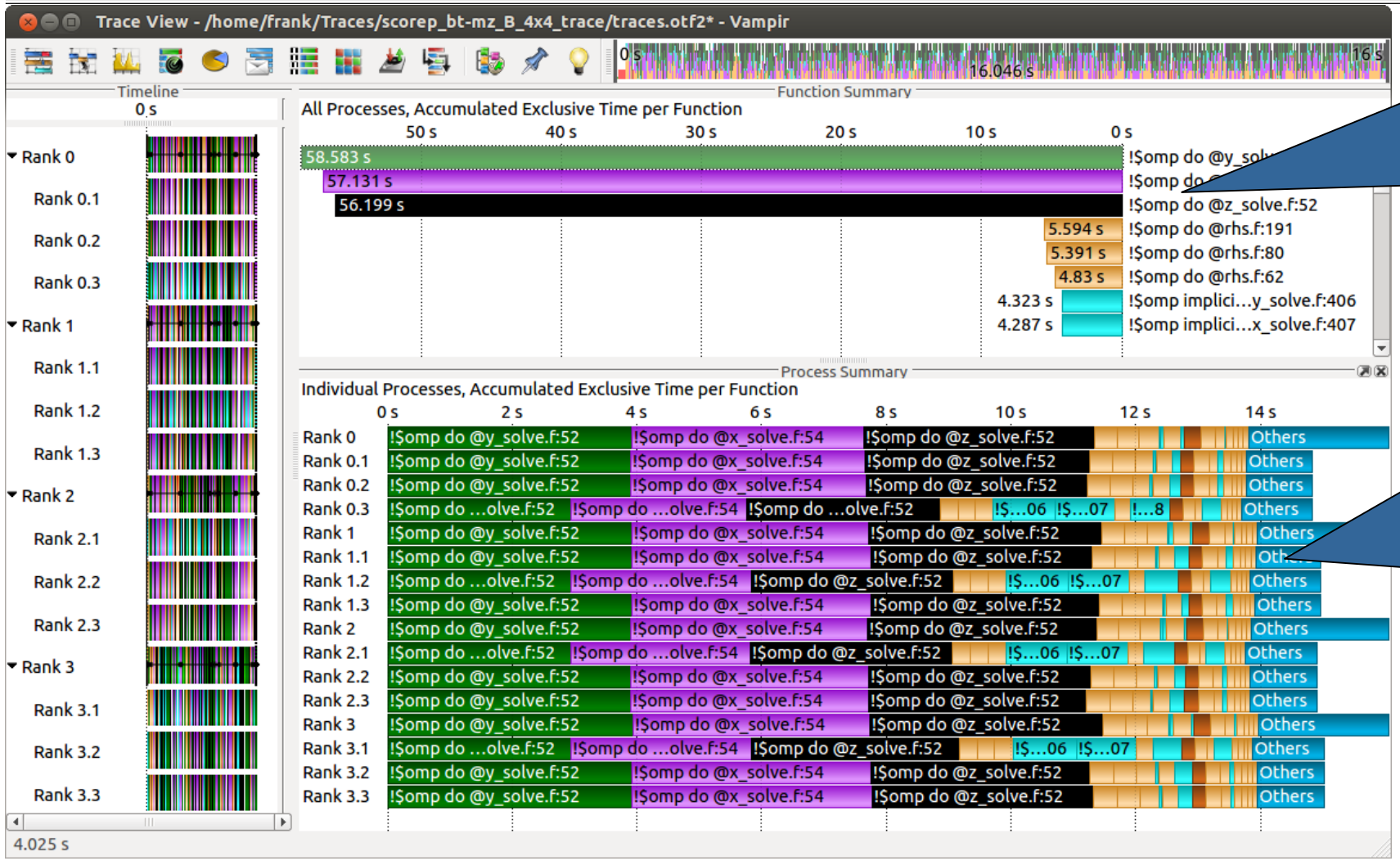
MPI communication results in lower floating point operations.

Visualization of the NPB-MZ-MPI / BT trace
Zoom in: Finalisation Phase



"Early reduce" bottleneck.

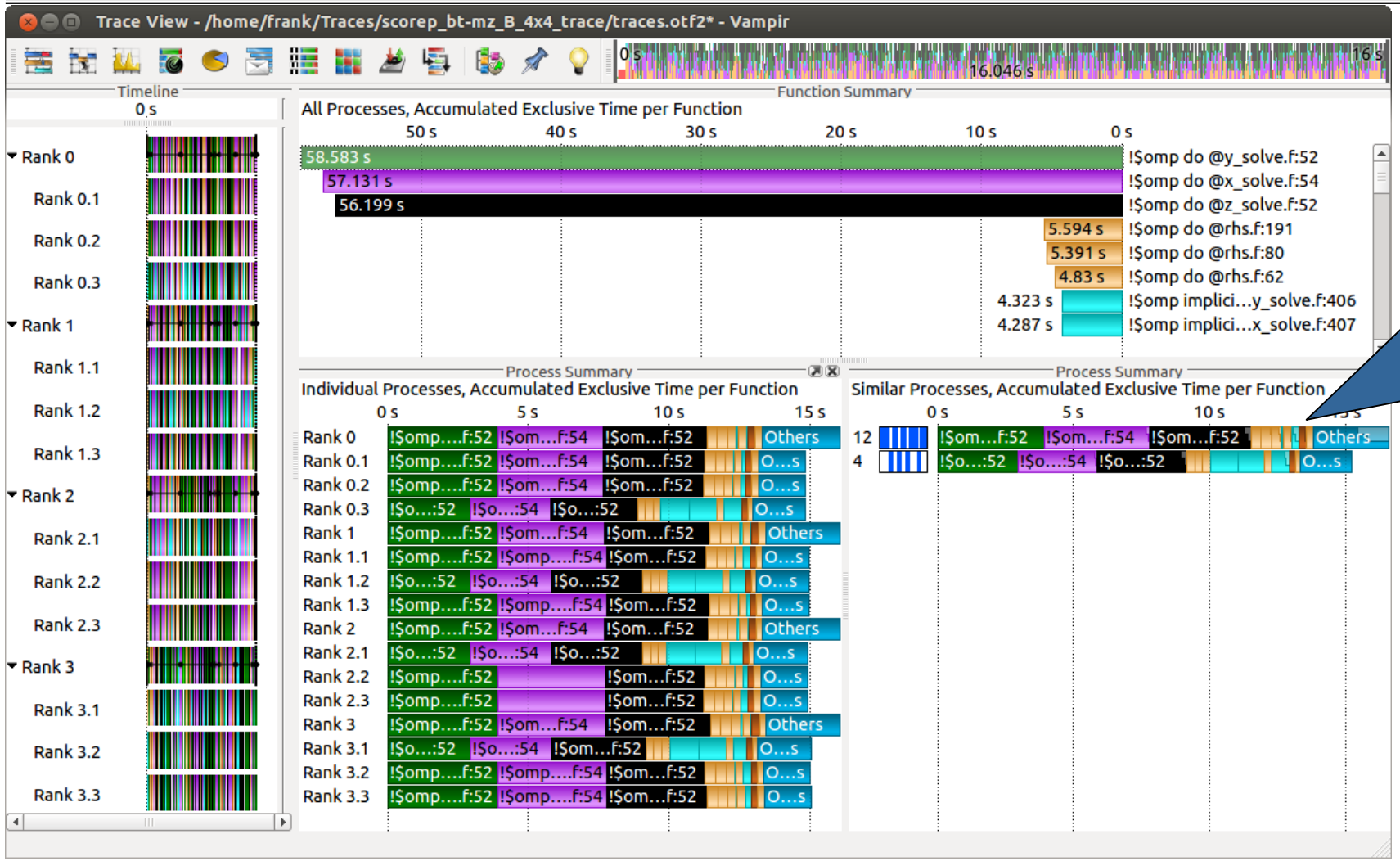
Visualization of the NPB-MZ-MPI / BT trace
Process Summary



Function Summary: Overview of the accumulated information across all functions and for a collection of processes.

Process Summary: Overview of the accumulated information across all functions and for every process independently.

Visualization of the NPB-MZ-MPI / BT trace
Process Summary



Find groups of similar processes and threads by using summarized function information.

Evolution of Vampir

- Started with MPI/OpenMP to analyze load imbalances
  - Floating Point Load Balance
  - Message Passing Memory Issue
  - Instructions per Cycle with Custom Metrics
- TU Dresden helped designing the CUPTI interface for NVIDIA
  - GROMACS MPI+OpenMP+CUDA
- I/O stack visualization
  - Multi-layer I/O
- Beyond HPC-Applications
  - Chrome Traces
  - SLURM job scheduling
  - Workflow execution traces

Demo: Floating Point Load Balance

- Weather code with coupled multi-physics and cloud components
- Load imbalance due to computation of emerging clouds only in parts of the simulation grid

Demo: Floating Point Load Balance

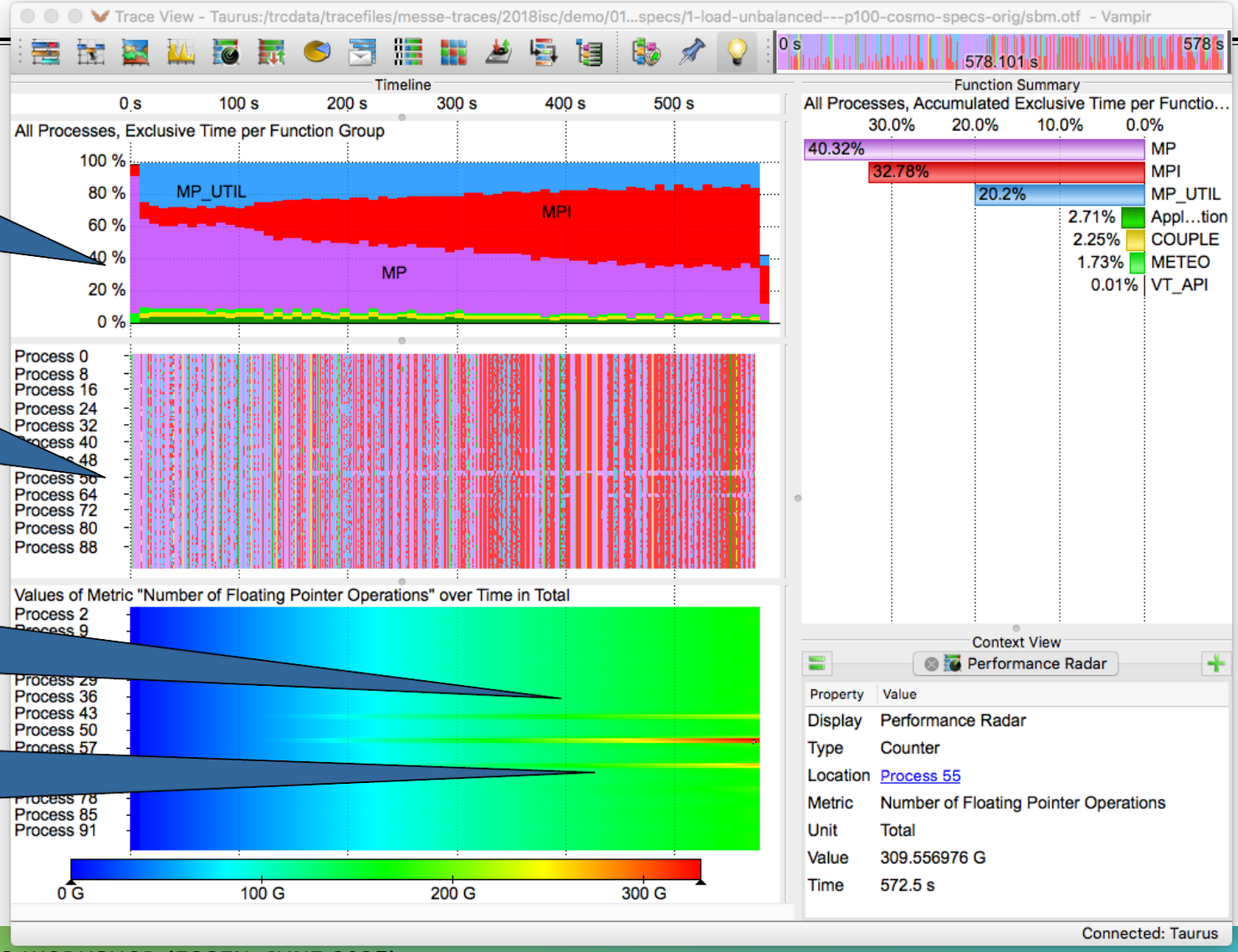
`traces/01-study-floating-point-load-balance`

Message Passing share increases over time. Happens uniformly?

No, MPI Share does not increases on Process 56. Why not?

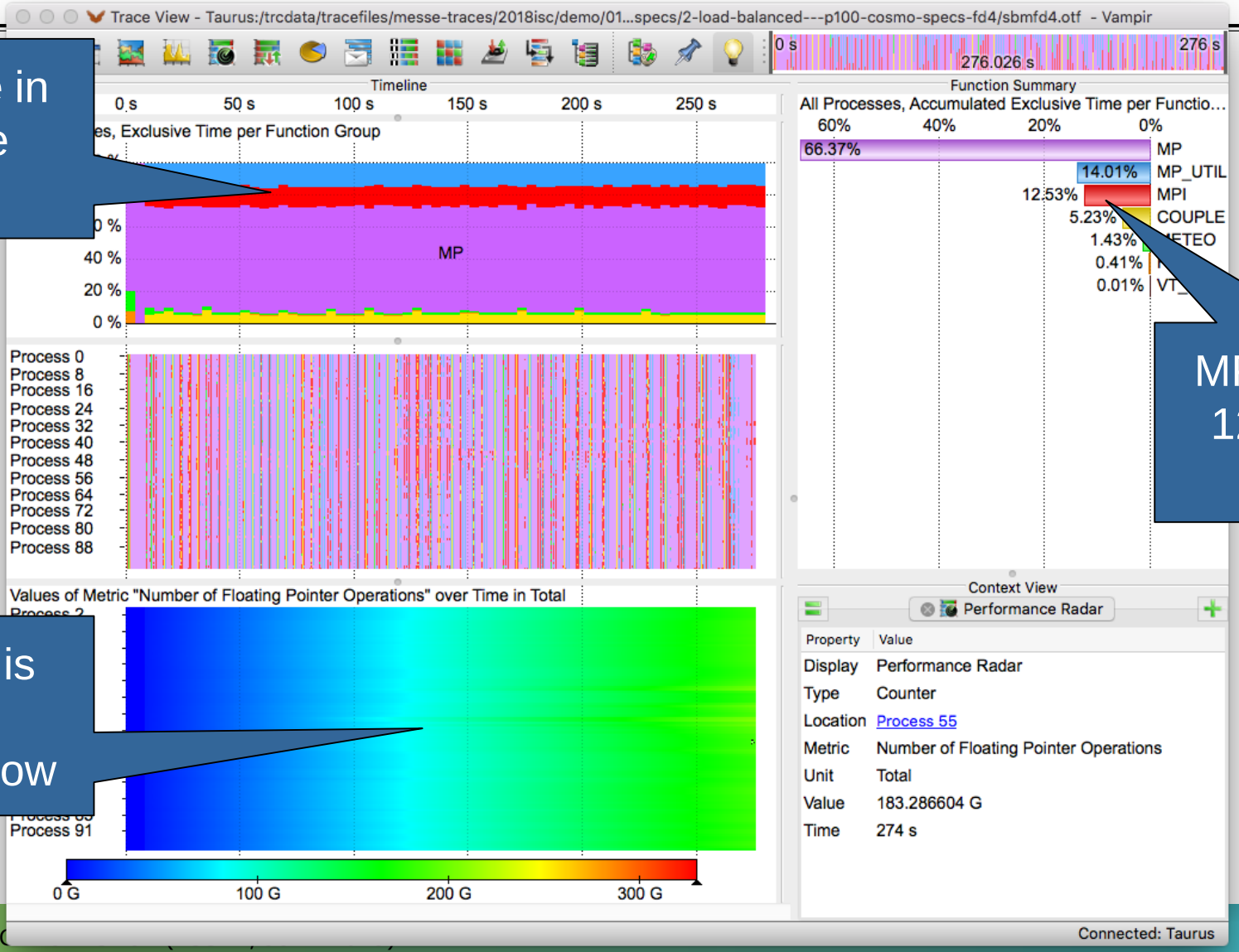
Process 56 is loaded heavily with FLOPs starting at t=130s

99 Processes wait for one process to finish



Demo: Floating Point Load Balance

`traces/01-study-floating-point-load-balance`



No increase in MPI share anymore

MPI consumes 12.5 % of the total time

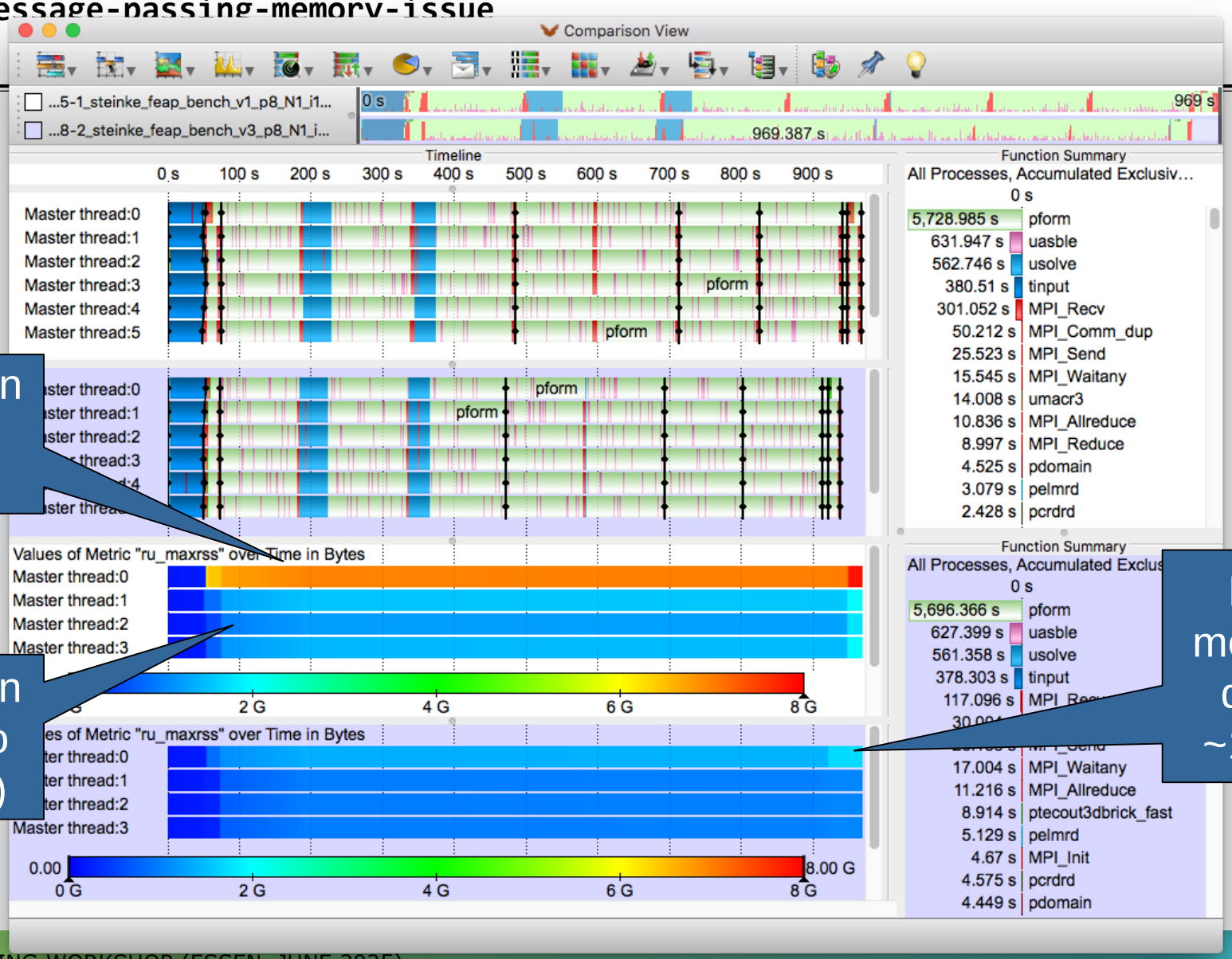
FLOP load is equally distributed now

Demo: Unexpected Memory Demand

- Unexpected memory demand from MPI implementation due to large number of small messages sent to one rank

Demo: Unexpected Memory Demand

`traces/02-study-message-passing-memory-issue`



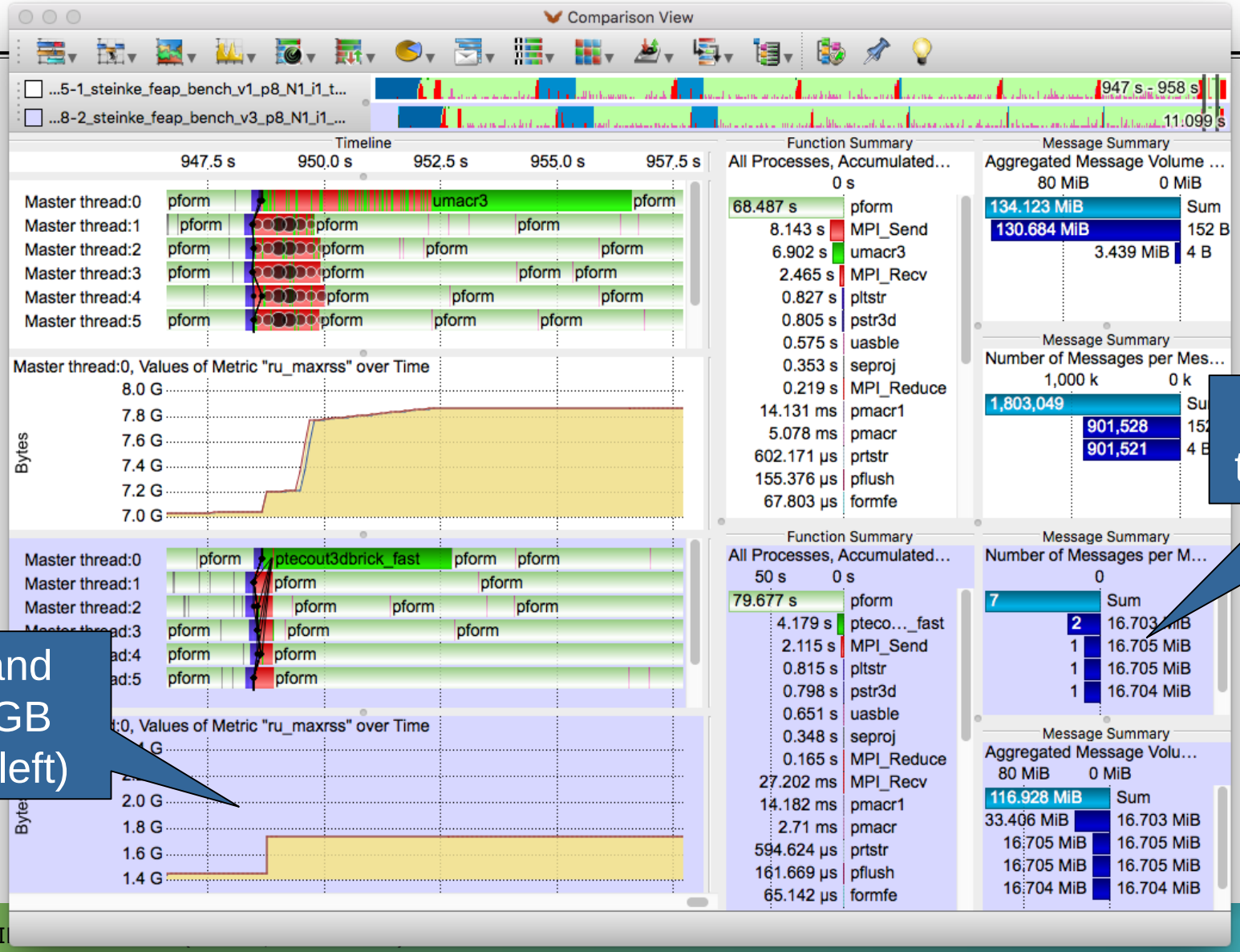
Memory demand on rank 0 explicably high (~ 8 GB)

Memory demand on rank 1 – (N-1) also too high (~1.8 GB)

Prior to new module (output) demand was ~1GB per rank

Demo: Unexpected Memory Demand

traces/02-study-message-passing-memory-issue



Memory demand down to ~1.8 GB (still one issue left)

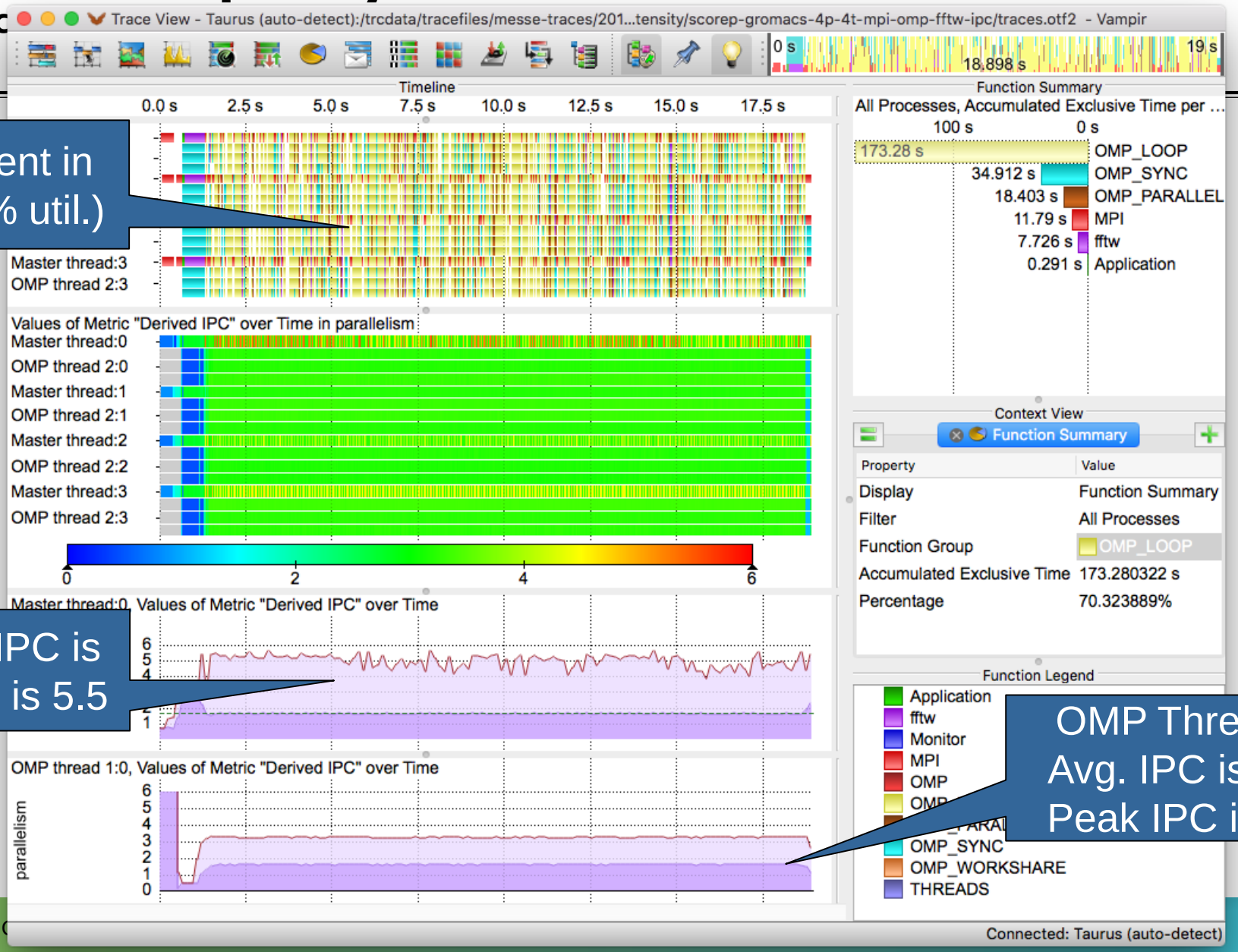
Messages merged to one big message

Demo: Instructions per Cycle with Custom Metrics

- Counters can be versatile used in calculations

Demo: Instructions per Cycle with Custom Metrics

`traces/03-study-c`



~12s time spent in OpenMP (62% util.)

Master: Avg. IPC is 1.8, Peak IPC is 5.5

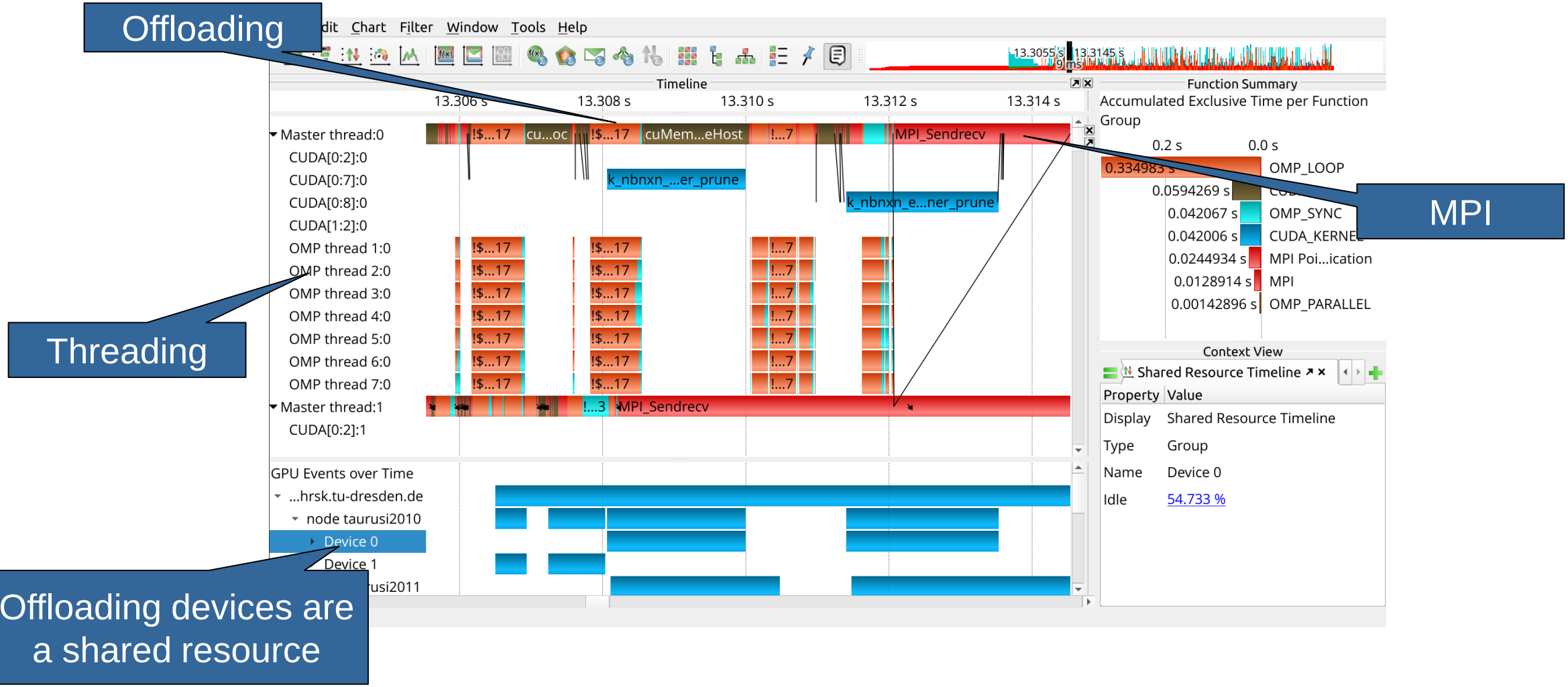
OMP Threads: Avg. IPC is 1.8, Peak IPC is 3.5

Demo: GROMACS MPI+OpenMP+CUDA

- Holistic view across multiple parallel paradigms

Demo: GROMACS MPI+OpenMP+CUDA

traces/05-study-offloading

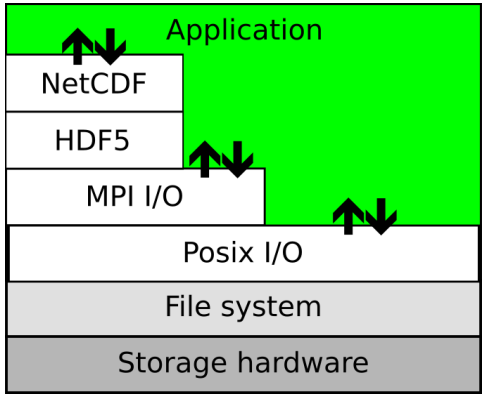


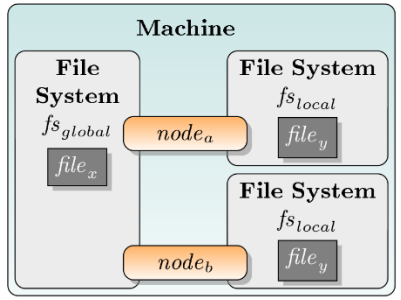
Demo: Multi-layer I/O

- Record interaction between multiple layers

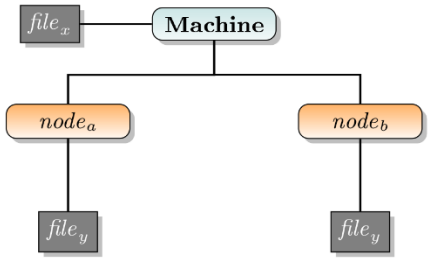
- MPI I/O (`MPI_File_open`)
- ISO C I/O (`fopen`)
- POSIX I/O (`open`)

- System tree information determine whether file resides in a shared file system





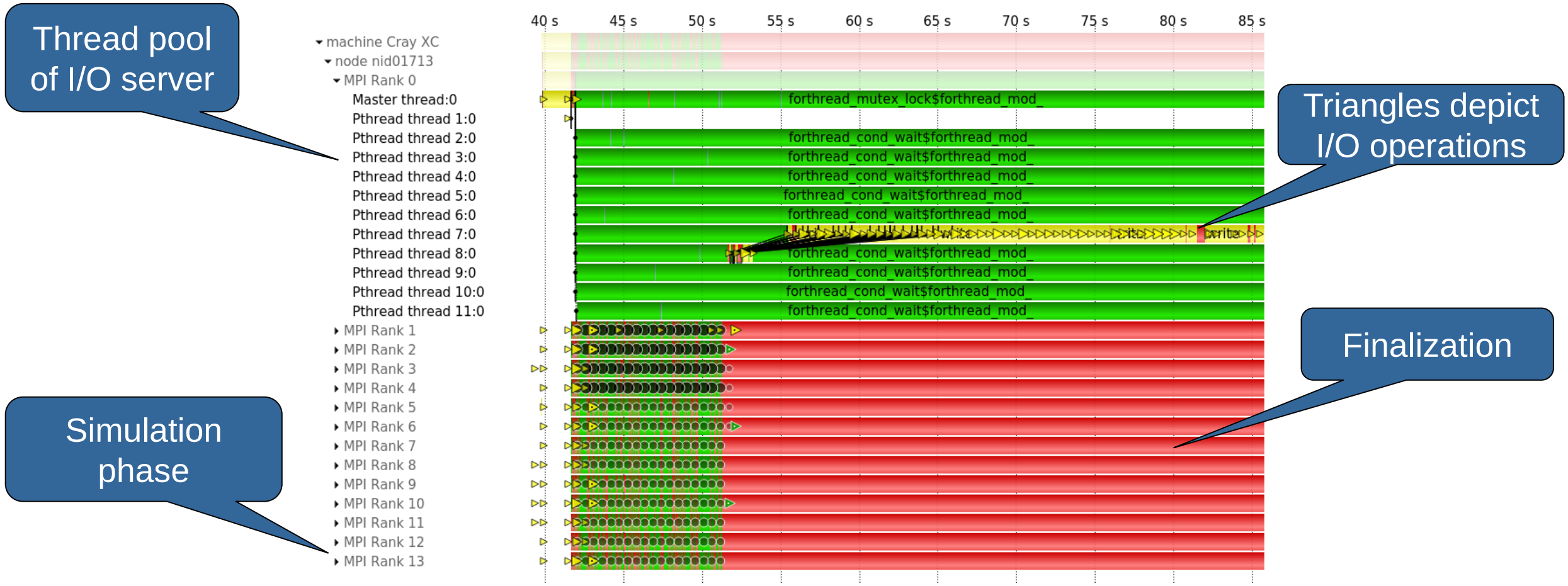
(a) Hardware topology



(b) System tree representation

Demo: Multi-layer I/O

traces/04-study-io-stack



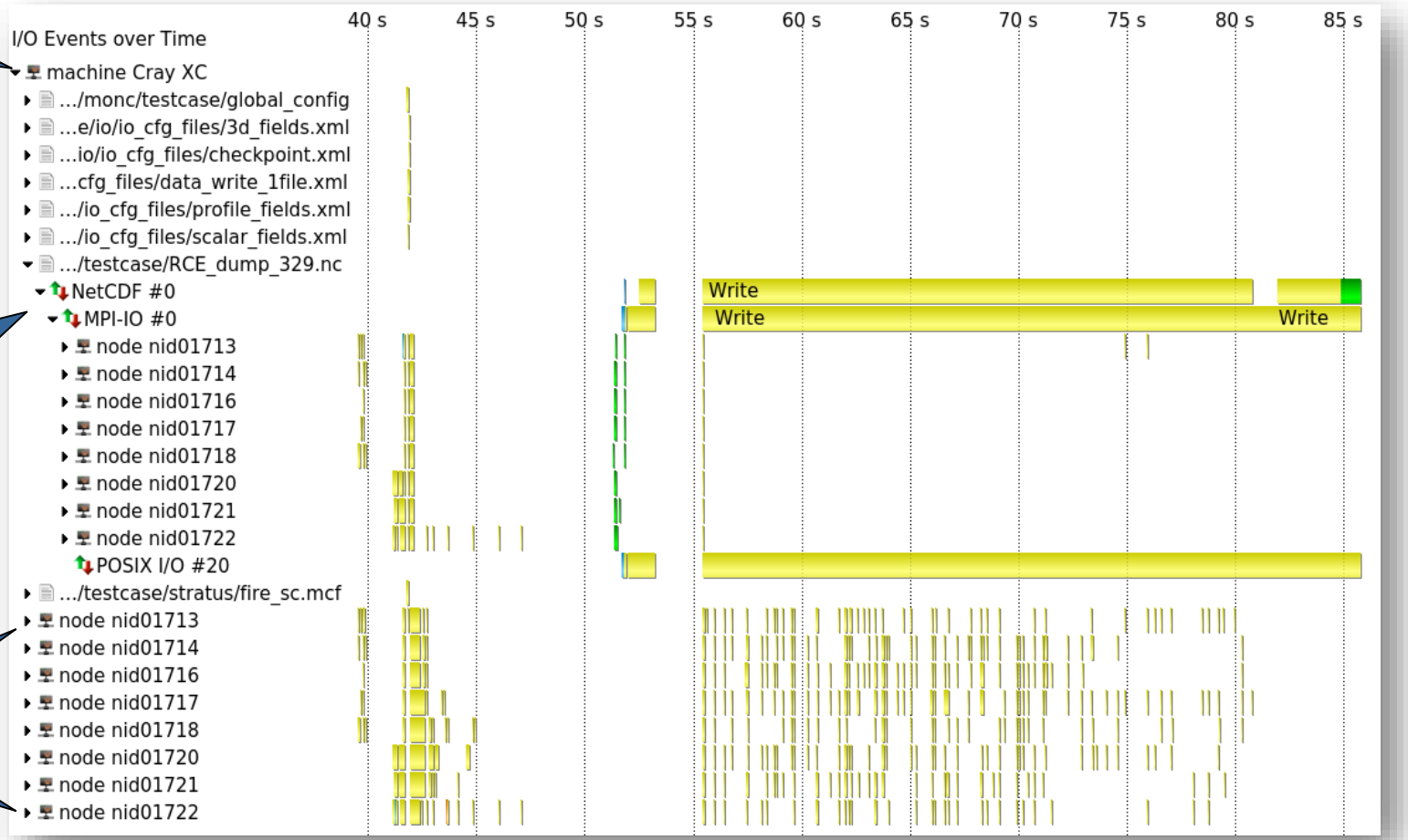
Demo: Multi-layer I/O

traces/04-study-io-stack

Shared files

I/O Stack represented by handles

Node local files

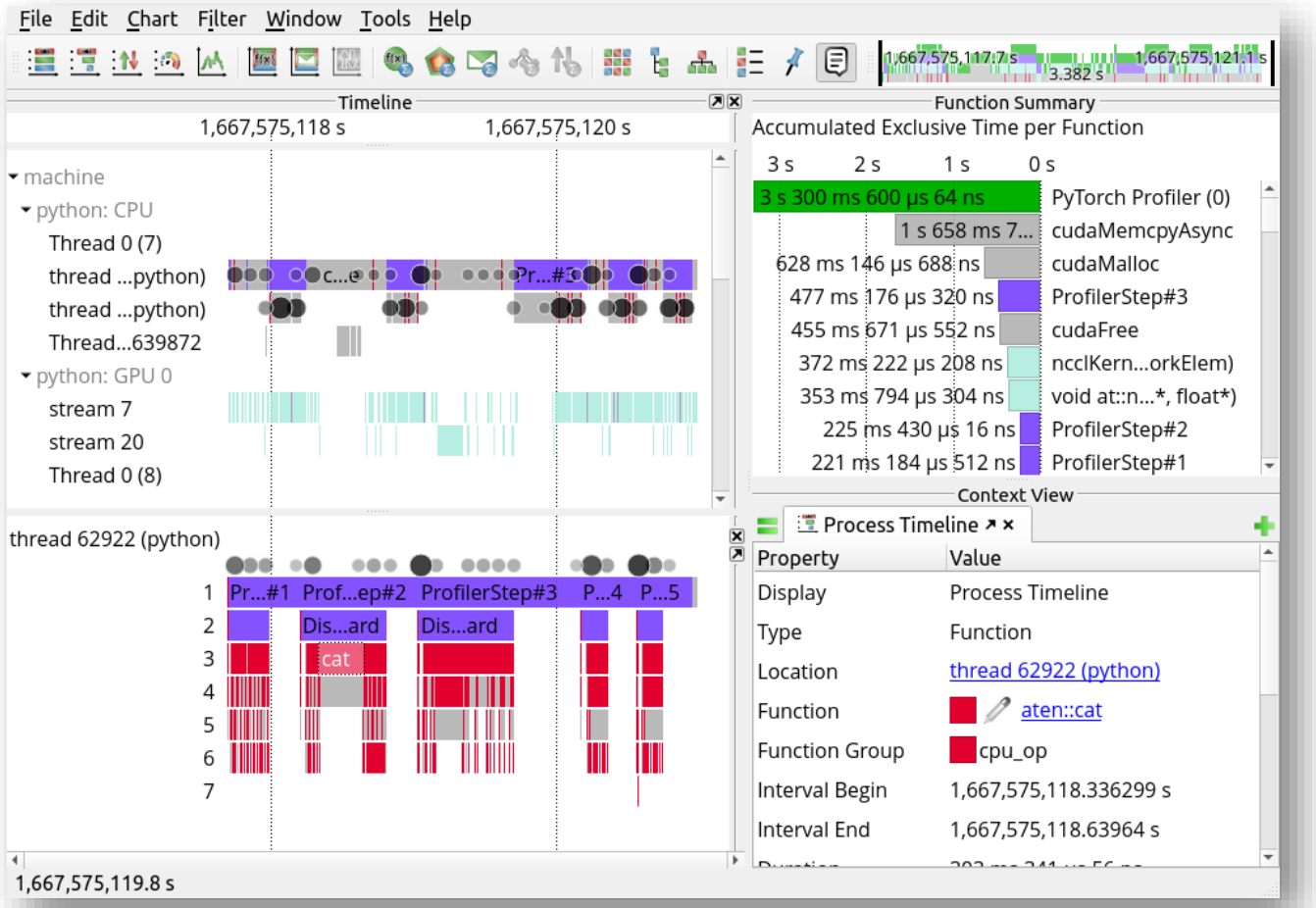


Demo: Chrome Traces

- Versatile trace format used by a multitude of applications and frameworks
  - PyTorch and TensoFlow
  - AMD rocprof
  - LLNL Caliper
  - …
- Browser based visualization limited by memory

Demo: Chrome Traces

Chrome Traces



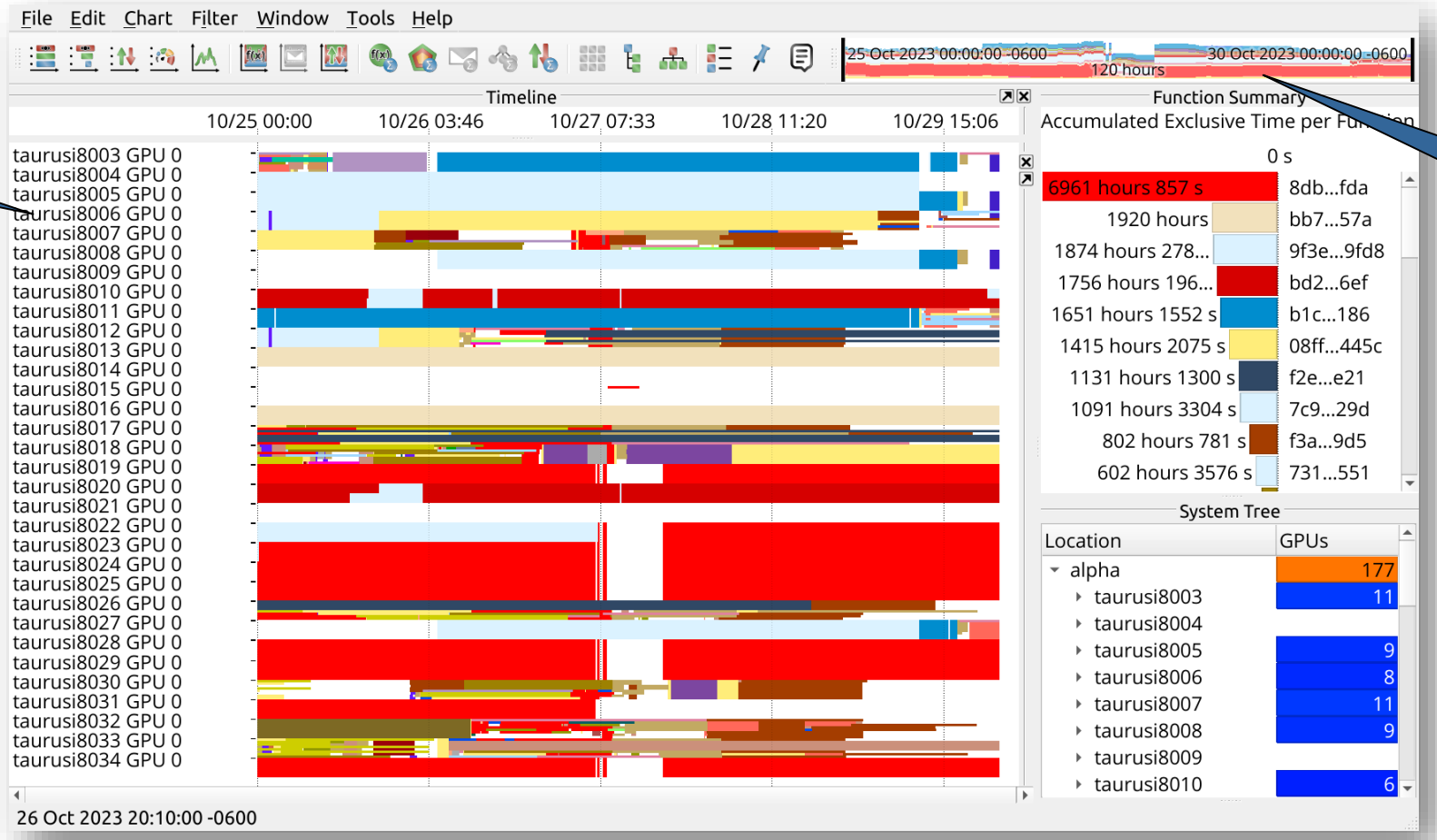
Demo: SLURM job scheduling

- Visualization of SLURM job scheduling

Demo: SLURM job scheduling

PIKA Slurm/alpha_20231025-30.json.gz

Each GPU of the system



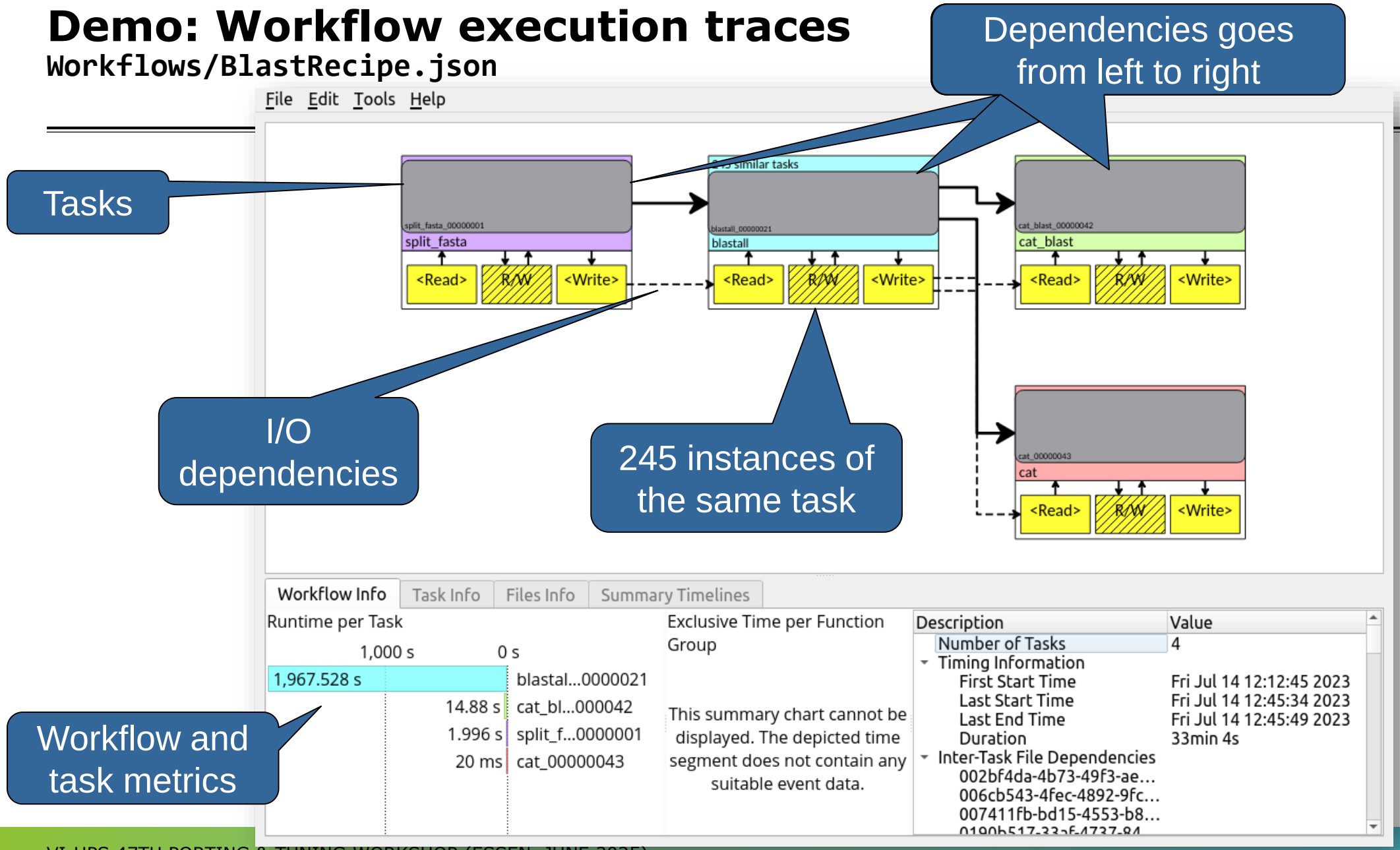
Full dates and durations in hours

Demo: Workflow execution traces

- Based on the wfcommons WfFormat JSON file

Demo: Workflow execution traces

`Workflows/BlastRecipe.json`



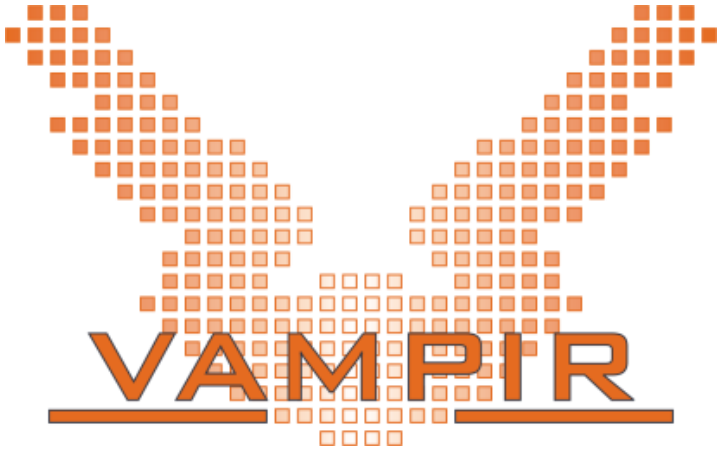
Summary and Conclusion

Summary

- Scalable visualization of event traces
- Color coding activities to easily identify program structure
- Client-Server (MPI) architecture to utilize HPC resources
- Supports multiple trace formats produced by different measurement tools
  - OTF2
    - Score-P
    - lo2s
    - TAU
    - Intel Trace Analyzer[1]
    - The Structural Simulation Toolkit[2]
  - Chrome Trace Format
    - TensorFlow
    - PyTorch
    - Cmake build
  - WfCommons
    - Fireworks
    - RADICAL
    - Nextflow
    - Snakemake

[1] *https://www.intel.com/content/www/us/en/docs/trace-analyzer-collector/user-guide-reference/2022-2/otf2-format-support.html*
[2] *http://sst-simulator.org*

GWT Gesellschaft für Wissens- und Technologietransfer



https://vampir.eu

service@vampir.eu