TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Towards Parallel Performance Analysis Tools for the OpenSHMEM Standard

Sebastian Oeste, Andreas Knüpfer, Thomas Ilsche, Ronny Tschüter, Felix Schmitt

OpenSHMEM Workshop 2014-03-06, Annapolis

ZIH
Center for Information Services & High Performance Computing

Overview

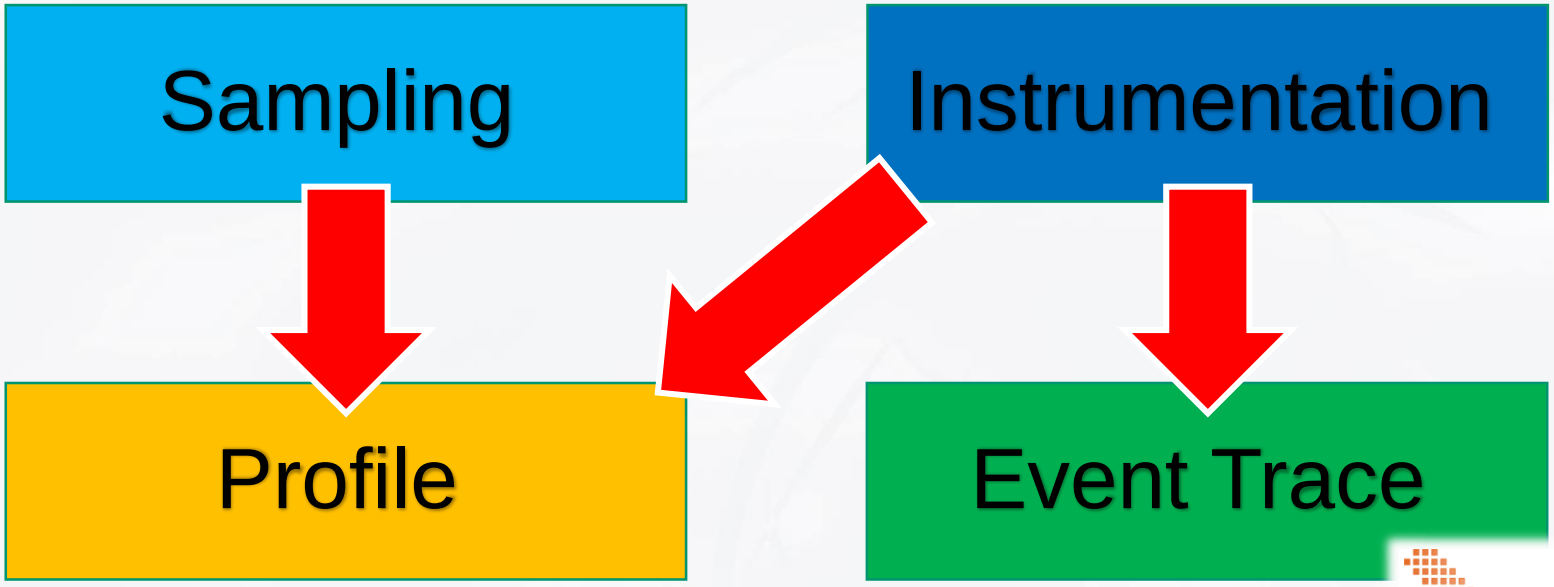
- Introduction
- Vampir Event Trace Visualization
- Generic PGAS Event Type Definitions
- OpenSHMEM support in Score-P
- Hybrid OpenSHMEM+CUDA programs
- First experiments and insights

- Conclusions, Outlook, Feedback

Parallel Performance Analysis and Tools

- Performance is important in HPC, isn‘t it?
- Have a performance tuning phase
  - just like a testing and debugging phases
- Use dedicated tools
  - Do not DIY, really!

Parallel Performance Analysis Approaches



Concise data sets

Good overview, limited detail, no outliers

Extensive data sets

Most detailed

Vampir

Trace View - /home/frank/Traces/scorep_bt-mz_B_4x4_trace/traces.otf2* - Vampir

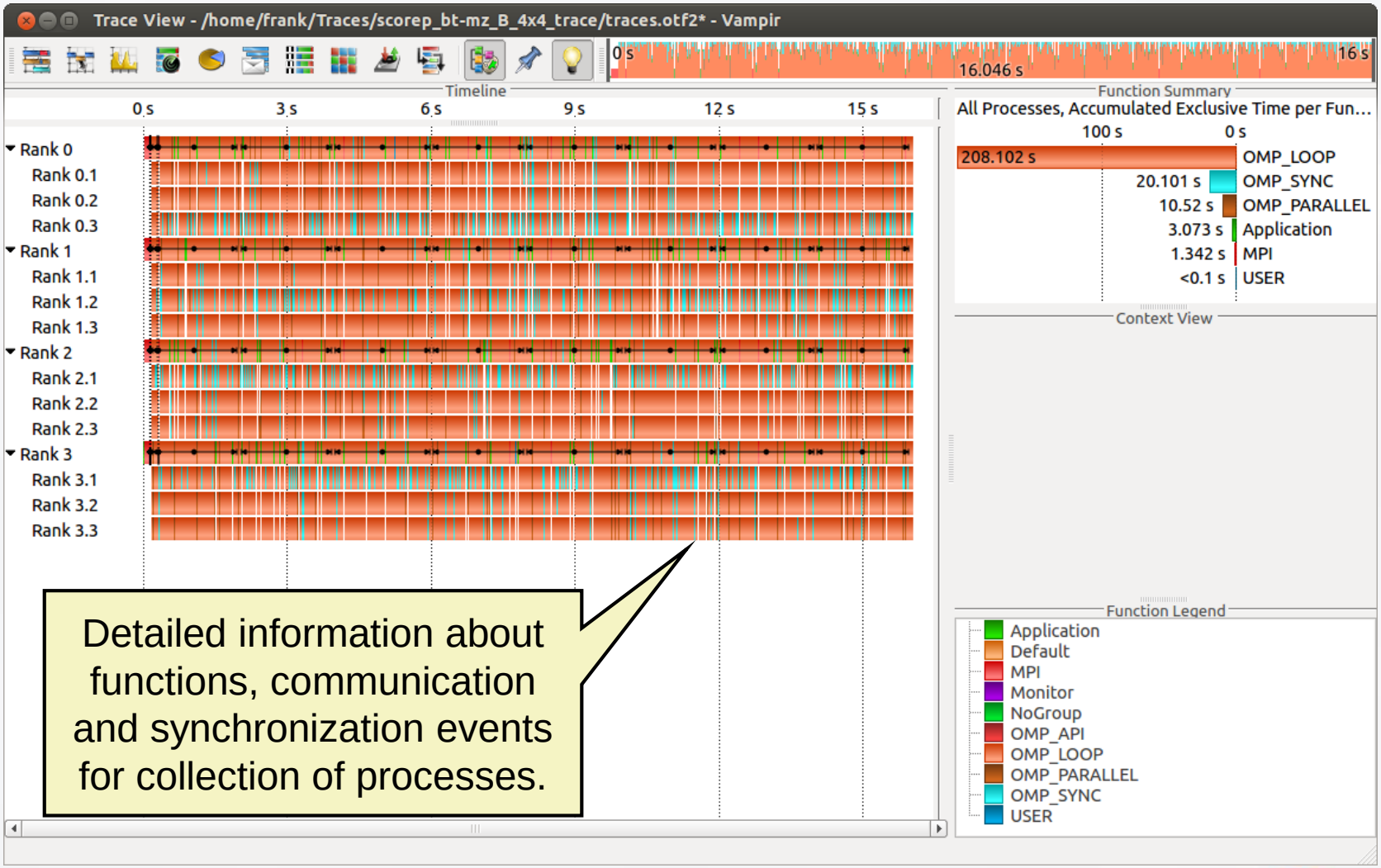
Navigation Toolbar

Function Summary

Master Timeline

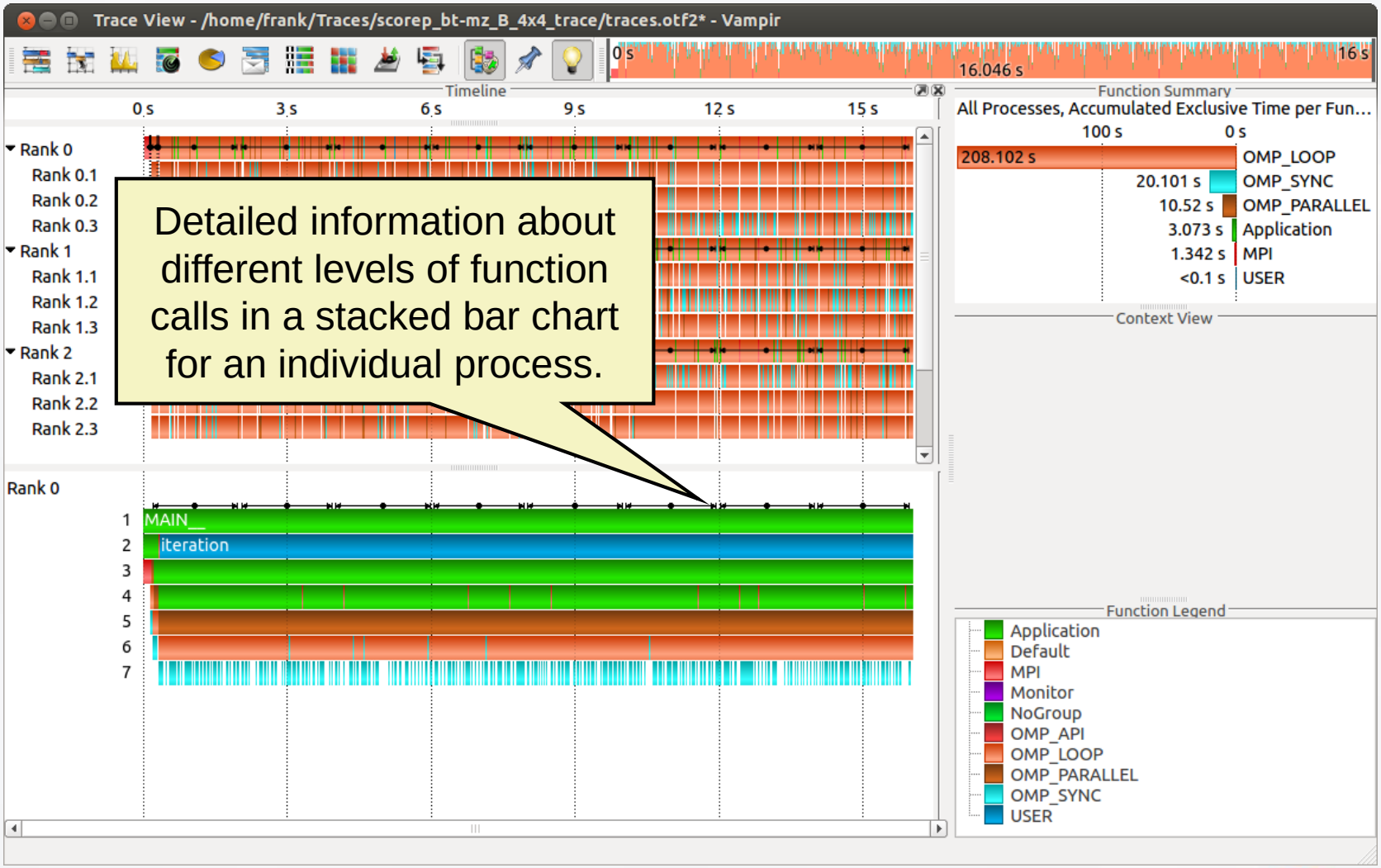
Function Legend

Vampir



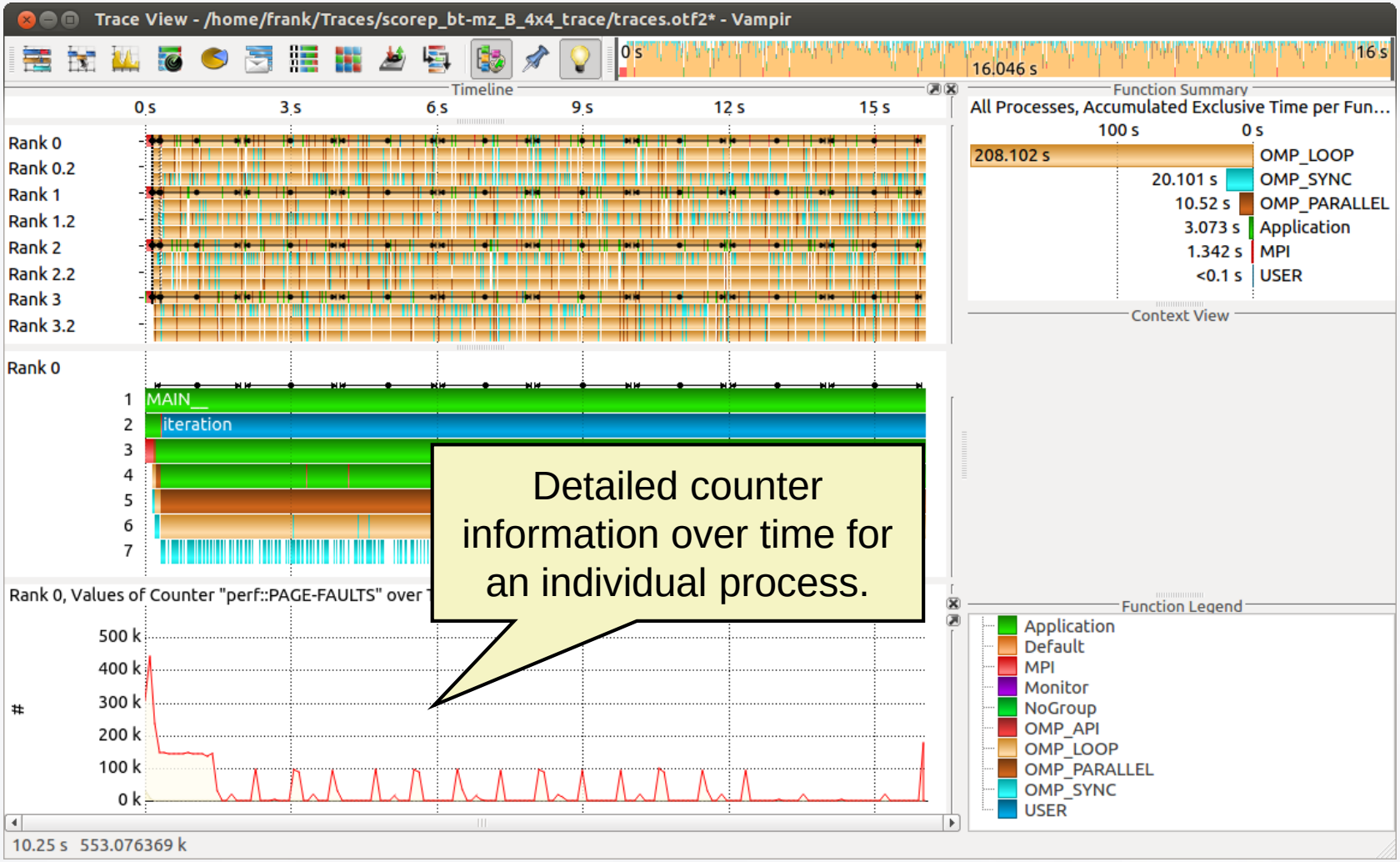
Detailed information about functions, communication and synchronization events for collection of processes.

Vampir



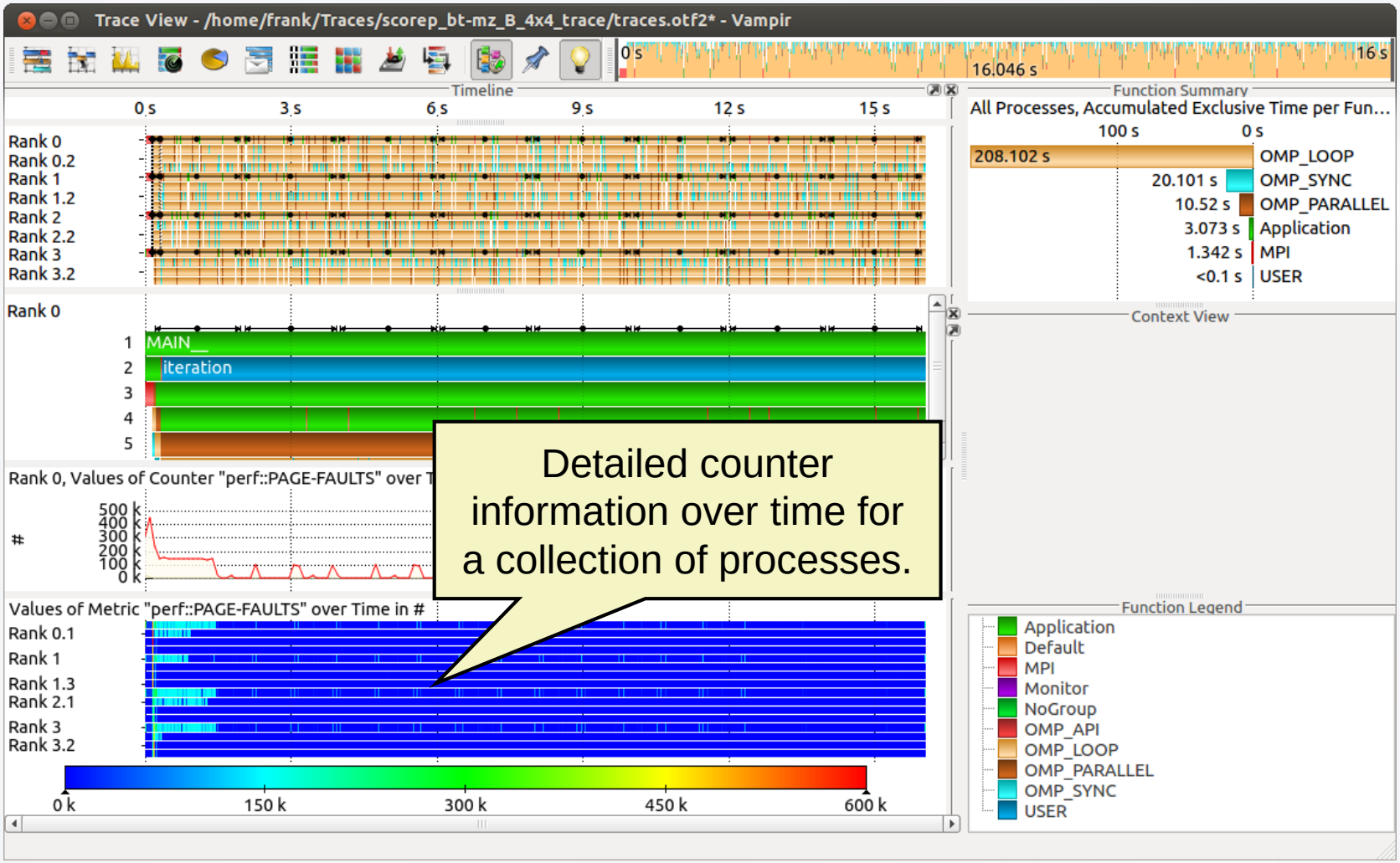
Detailed information about different levels of function calls in a stacked bar chart for an individual process.

Vampir



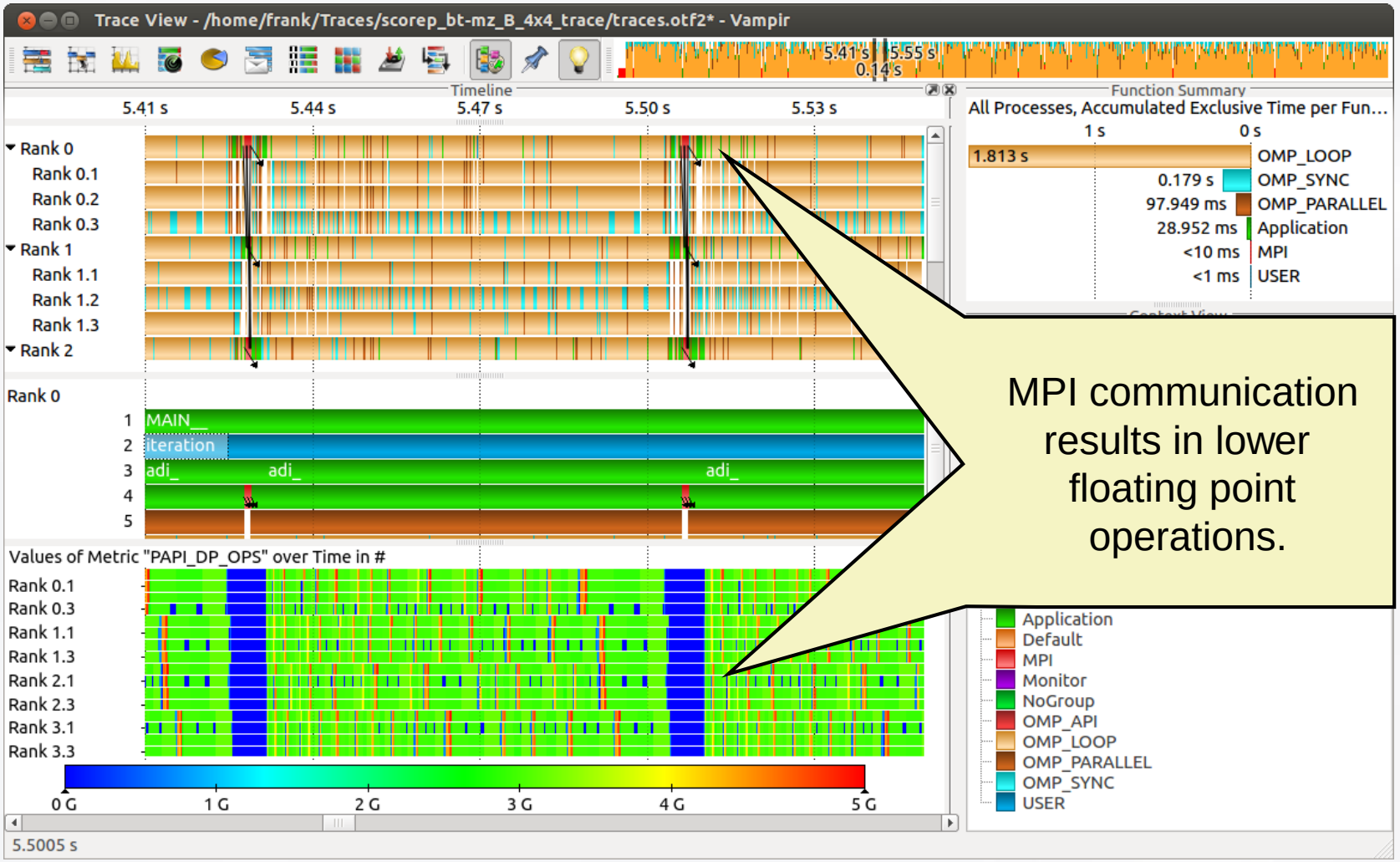
Detailed counter information over time for an individual process.

Vampir



Detailed counter information over time for a collection of processes.

Vampir

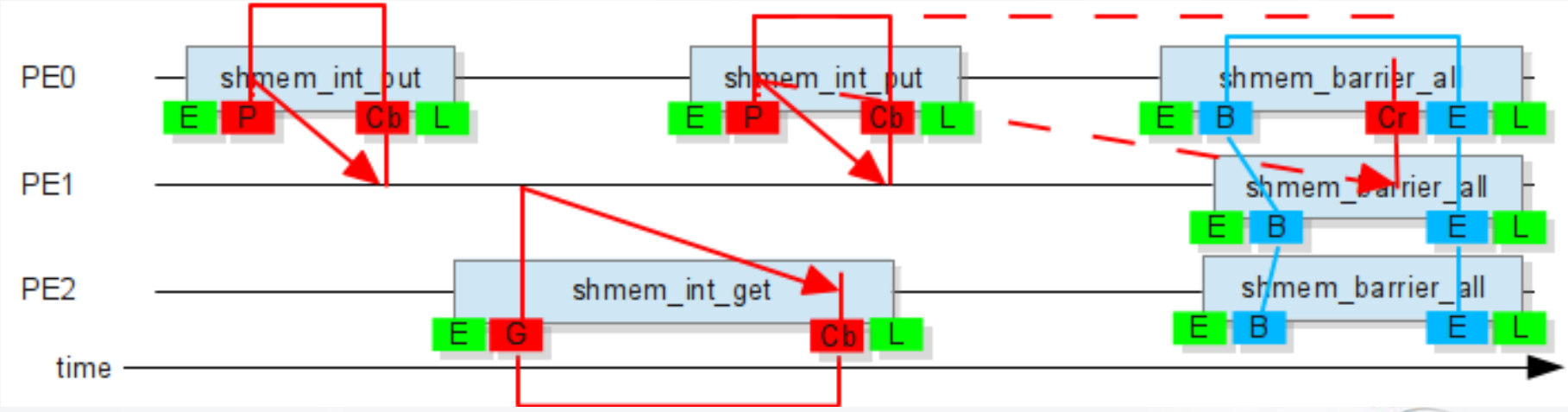


MPI communication results in lower floating point operations.

PGAS Event Types: Put and Get

New event types for PGAS parallelization libraries:

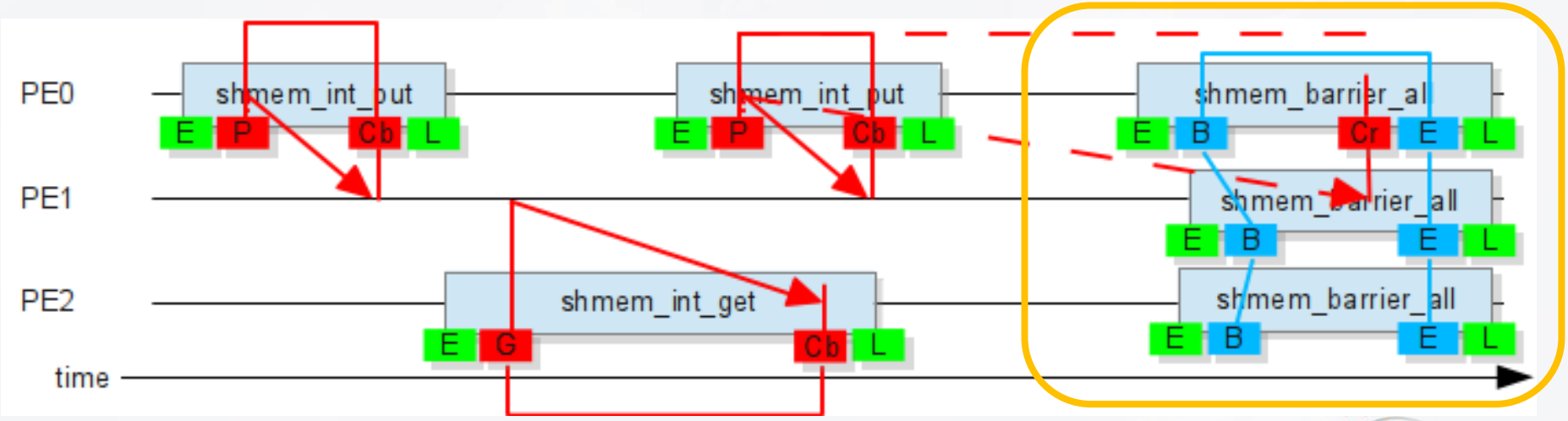
- Generic event model for OpenSHMEM, MPI one-sided, GASPI, …
- Put and Get recorded on the active side only
  - Local and remote completion
  - Identical for get but separate for put
  - Visualization for put events based on local completion only!



PGAS Event Types: Collective Operations

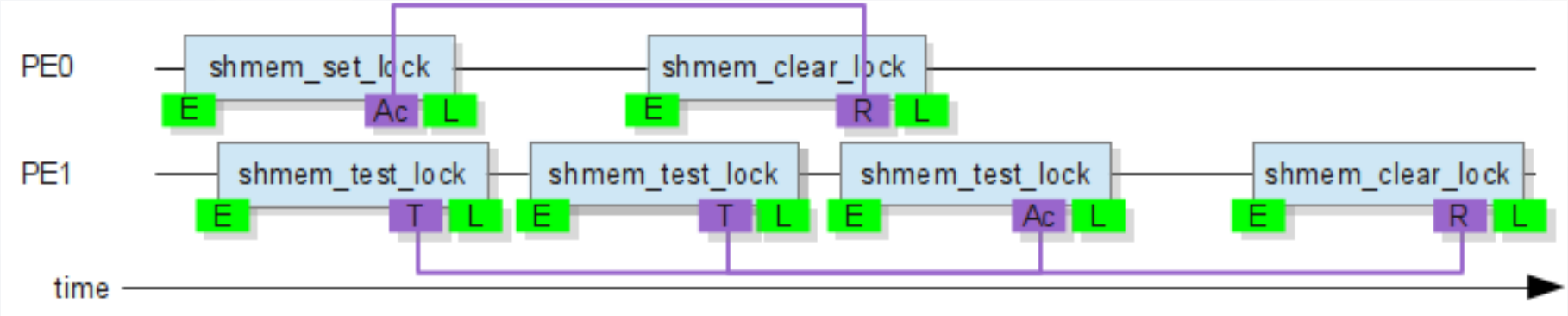
Refined event types for collective operations:

- Similar to MPI collectives
  - Extra event types, also track “memory windows”
  - Visualization identical



PGAS Event Types: Locks etc.

- New event types for mutual exclusive locks
  - Diverse definitions in different PGAS standards
  - Track attempts, lock operations, release operations



- More new event types e.g., atomic operations
- Many other event types reused
  - Enter and Leave for API calls and user routine calls
  - Performance counter samples

Demonstrator based on Cray SHMEM and VampirTrace

From the original OpenSHMEM 2013/2014 workshop paper:

Sebastian Oeste , Andreas Knüpfer, Thomas Ilsche:
Towards Parallel Performance Analysis Tools
for the OpenSHMEM Standard

Demonstrator for Cray SHMEM based on VampirTrace:

- Mapping to MPI event types
- Internal communication via Cray MPI
  - Cray SHMEM can co-exist with MPI
  - PE numbers == MPI rank in **`MPI_COMM_WORLD`**
- First experimental results

OpenSHMEM in Score-P

Score-P

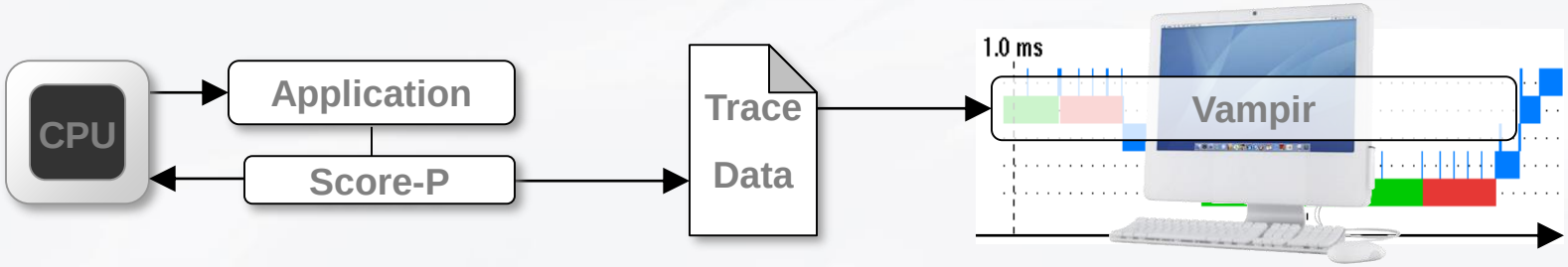
- Joint instrumentation and run-time measurement system for Vampir, Scalasca, TAU, and Periscope, in collaboration between TU Dresden, FZ Jülich, TU Munich, University of Oregon, RWTH Aachen University et.al. Open for new partners.

OpenSHMEM support in Score-P

- Generic PGAS event types
- Instrumentation via P-symbols or library wrapping
- Records profiles or event traces
- Native internal communication via OpenSHMEM
- Requires insertion of explicit finalize call

Acknowledgements to ORNL and UT Battelle for project funding

OpenSHMEM in Score-P




```
CC=scorep oshcc

CXX=scorep oshcxx

CFLAGS=-DNDEBUG -O3
```

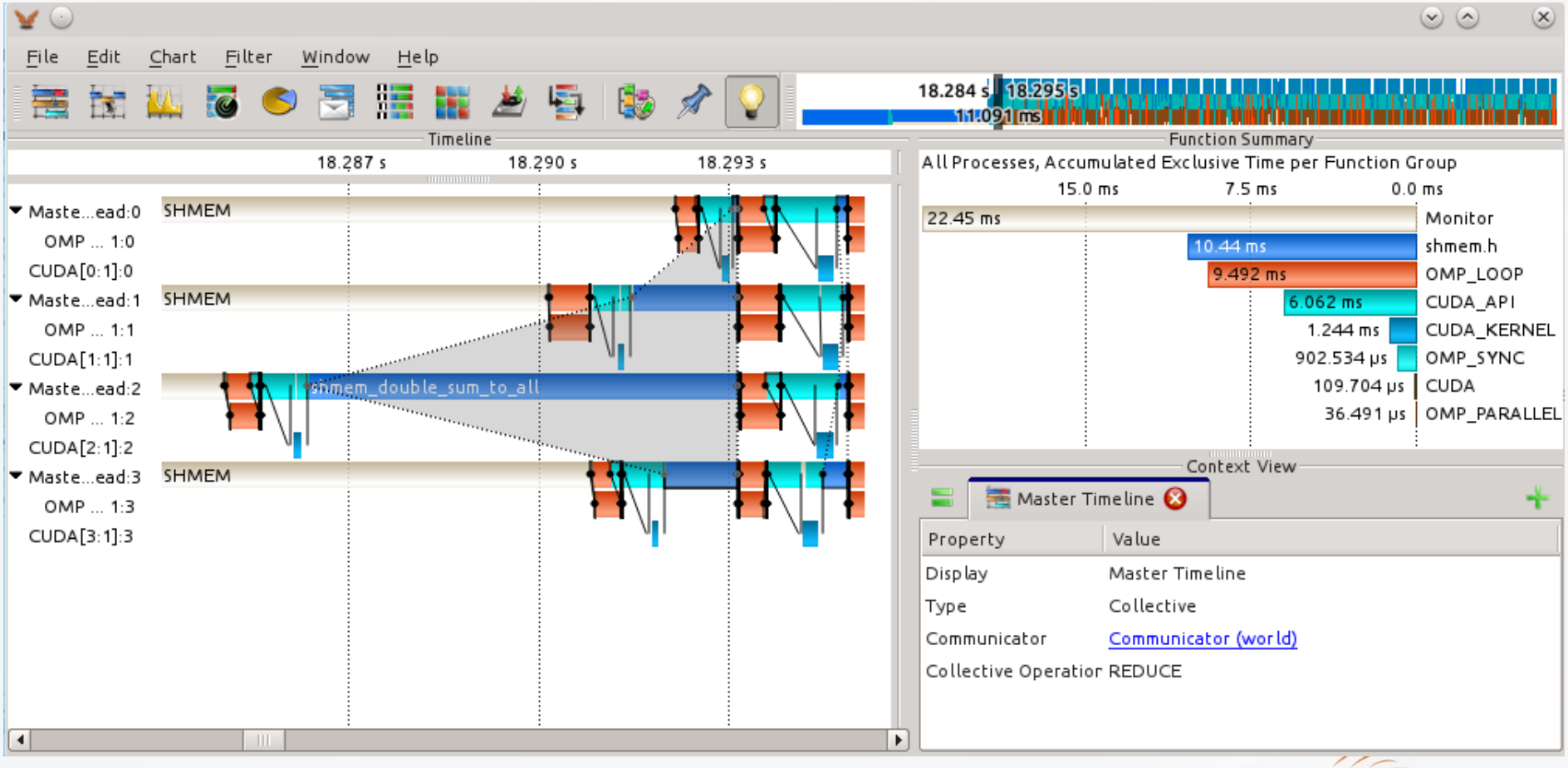
For instrumentation simply add the Score-P compiler prefix command

Status and Features

- Score-P 1.3 pre-release version supports OpenSHMEM instrumentation and monitoring
  - Using library wrapping or PSHMEM interface
  - Automatically detects the SHMEM compiler wrappers (e.g. oshcc)
  - Explicit finalize call required
  - Also provides performance counters, energy monitoring, etc.
- Score-P 1.3 latest beta-version supports hybrid combinations of OpenSHMEM with CUDA and OpenMP
  - No oshnvcc wrapper available, CUDA code must be compiled/linked separately
- First results …

OpenSHMEM+CUDA+OpenMP Application in Vampir

- Example with OpenSHMEM, CUDA, and OpenMP



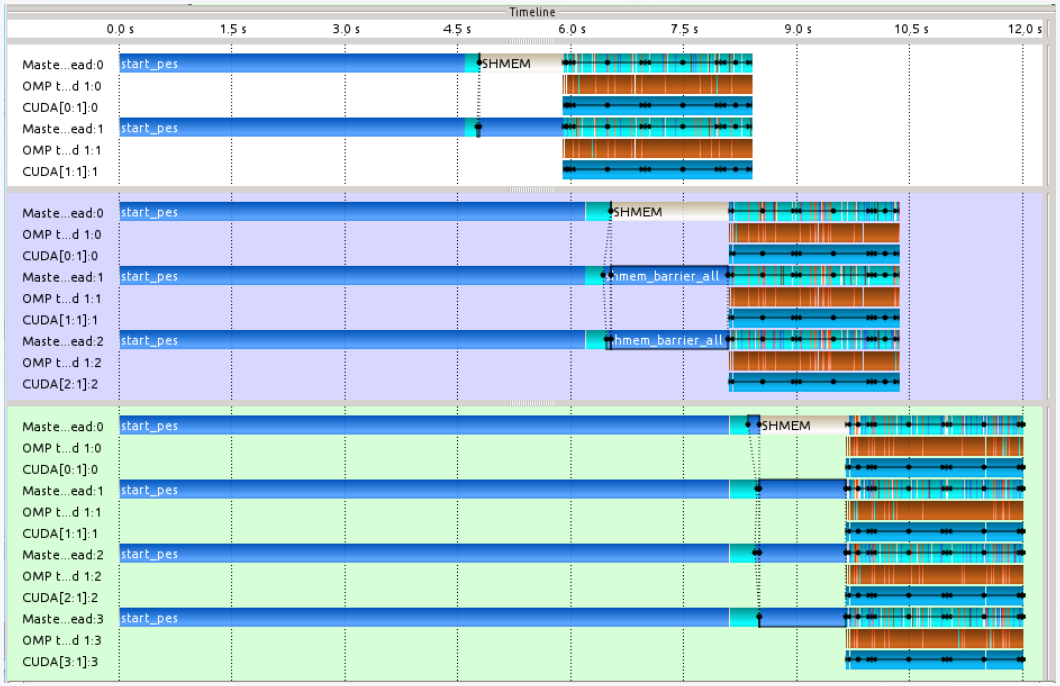
Experiments

Ported MPI codes to OpenSHMEM as test cases

- MPI/CUDA/OpenMP applications ported to use OpenSHMEM
  - Jacobi CUDA: iterative discrete Poisson equation solver using Dirichlet boundary conditions, max. two PEs/processes
  - φGPU: direct astrophysics N-Body simulation using 4th order Hermite integration scheme
- Measurement configuration for the following examples
  - BULL HPC system “Taurus” at ZIH, Infiniband interconnect
  - OpenSHMEM v1.0e, GASNET 1.22.0, conduit: ibv
  - GCC 4.4.6, bullxmpi/1.2.4.3, Nvidia K20X, CUDA 5.0
- First experiments and insights …

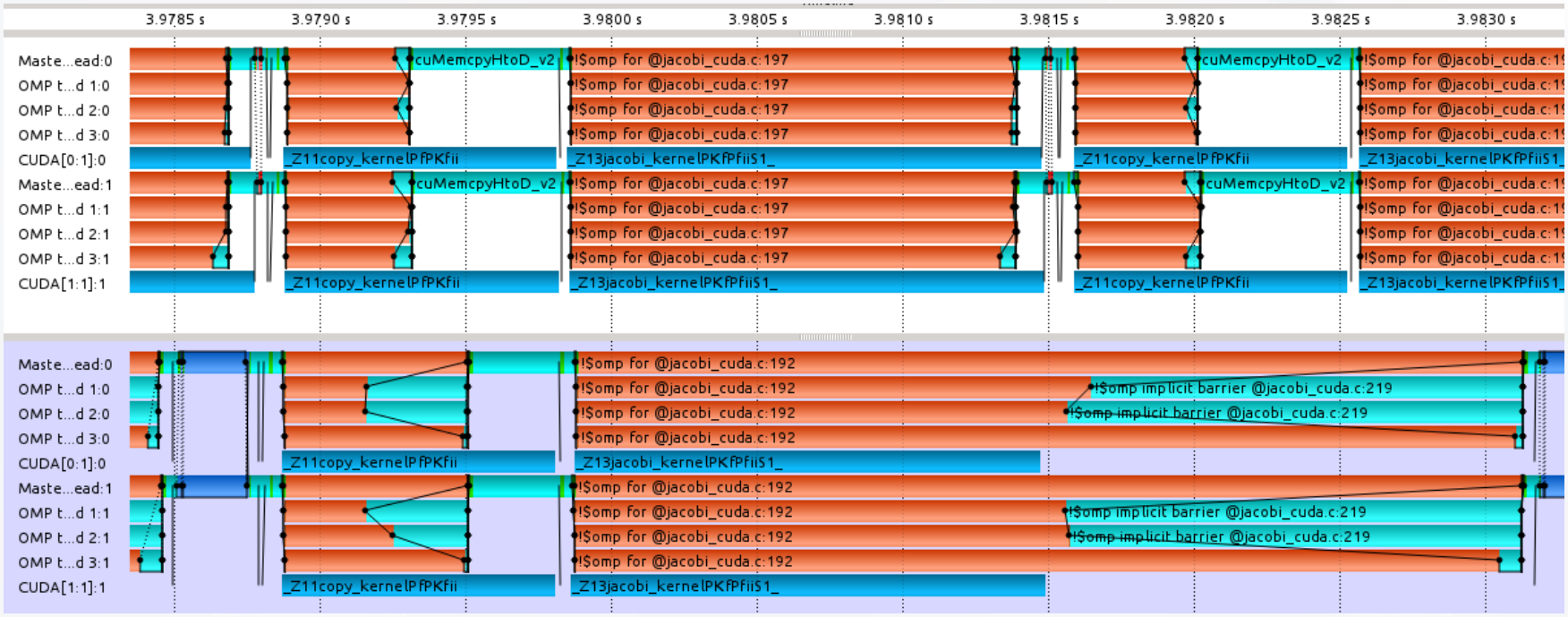
Issues with the OpenSHMEM Reference Implementation

- Compile errors (fixed together with maintainers)
- The **start_pes** initialization routine takes very long (overhead scales with number of PEs on a single node but not across multiple nodes)



Comparison of MPI and OpenSHMEM (1)

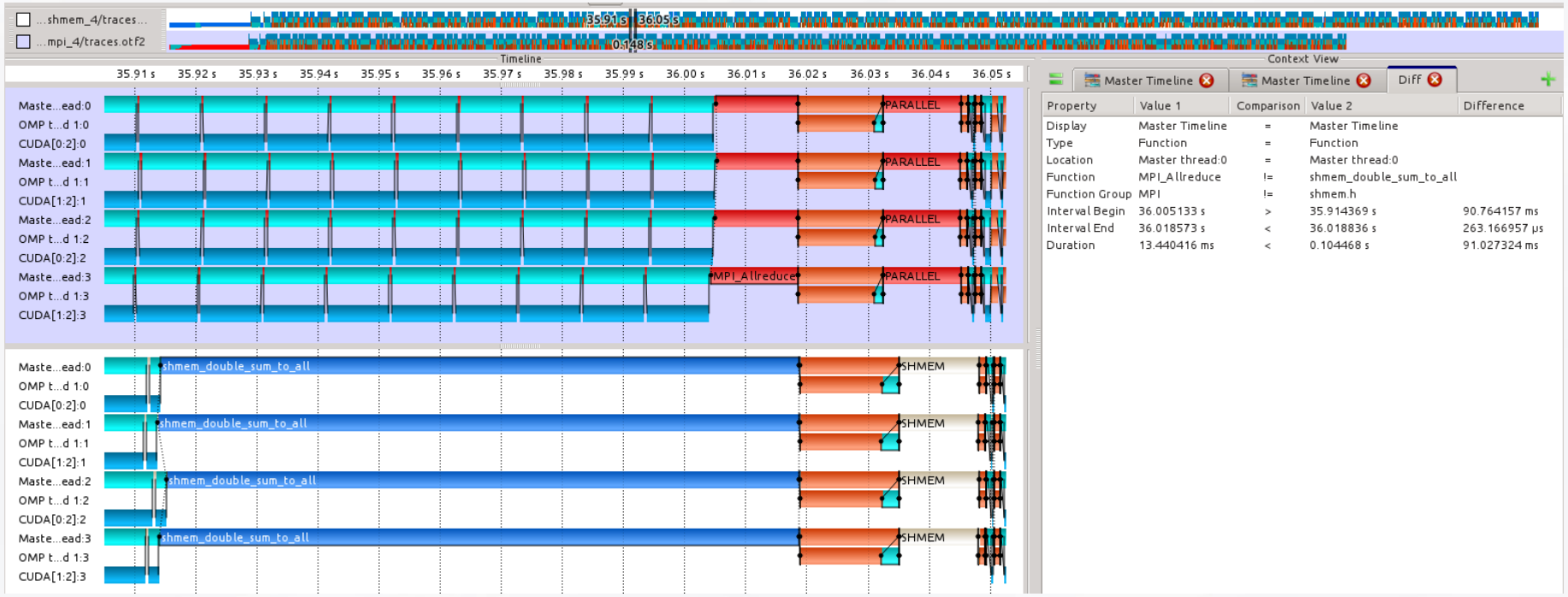
- OpenSHMEM introduces additional CPU overhead
- Results in longer and unbalanced OpenMP regions



MPI (top) vs. OpenSHMEM (bottom)
for Jacobi CUDA example

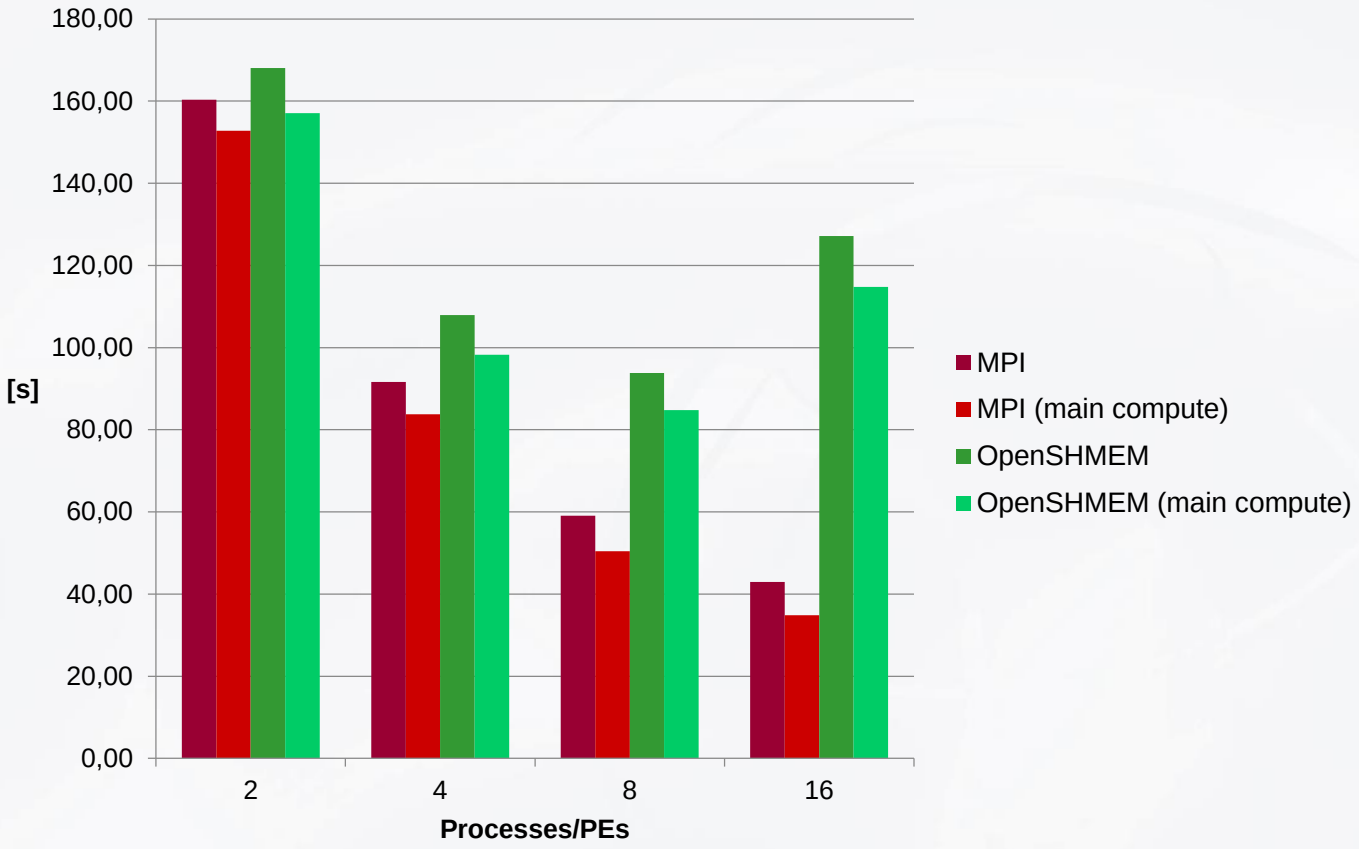
Comparison of MPI and OpenSHMEM (2)

- Collective operations (e.g. allreduce) less efficient than MPI counterparts



MPI_Allreduce (13.4ms, top) vs.
shmem_double_sum_to_all (103ms, bottom) in φGPU

Comparison of MPI and OpenSHMEM (3)



Comparison of execution times [s] for φGPU (256K particles) using MPI and OpenSHMEM

Conclusions

- Tool support for OpenSHMEM
  - Score-P for instrumentation and run-time measurement
  - Vampir for scalable interactive visualization
  - Hybrid cases with OpenSHMEM together with CUDA or OpenMP
- Using generalized event model for PGAS libraries
  - Even extends to PGAS languages as well as DMA operations as in CUDA
- Feedback from OpenSHMEM community welcome
- Frequent tools tutorials, user support, mailing lists, …

Outlook

- Reducing tracing overhead for one-sided communication
  - combine many small put/get operations in a useful way
  - Summarize number of ops and transfer volumes
  - … for sensible phases (not too long, not too short)
  - … per PE or per pairs of PEs
  - … always or only for bursts of small operations
  - Design appropriate visualization in Vampir
- Record information about local/remote communication partners (if available through OpenSHMEM API)
- Record memory accesses (addresses and variable names) for data layout optimization and/or race detection
- Record OpenSHMEM internal counters (if available)

Feedback to the OpenSHMEM Consortium

- Wish #1: Weak symbols and additional P-symbols
  - Currently optional in the OpenSHMEM reference implementation but not in the standard

```
shmem_char_put   →   pshmem_char_put
shmem_short_put  →   pshmem_short_put
shmem_int_put    →   pshmem_int_put
```

- Wish #2: Mandatory finalize call as counterpart to init

```
void start_pes( int npes );
void finalize_pes();
```

- Wish #3: New OpenSHMEM compiler wrapper for CUDA compiler