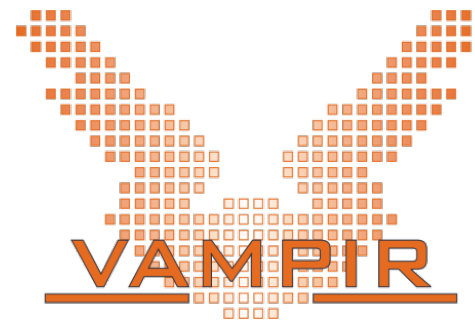


Performance Analysis Exercises with Vampir

Matthias Weber, Holger Brunst, Hartmut Mix
Technische Universität Dresden

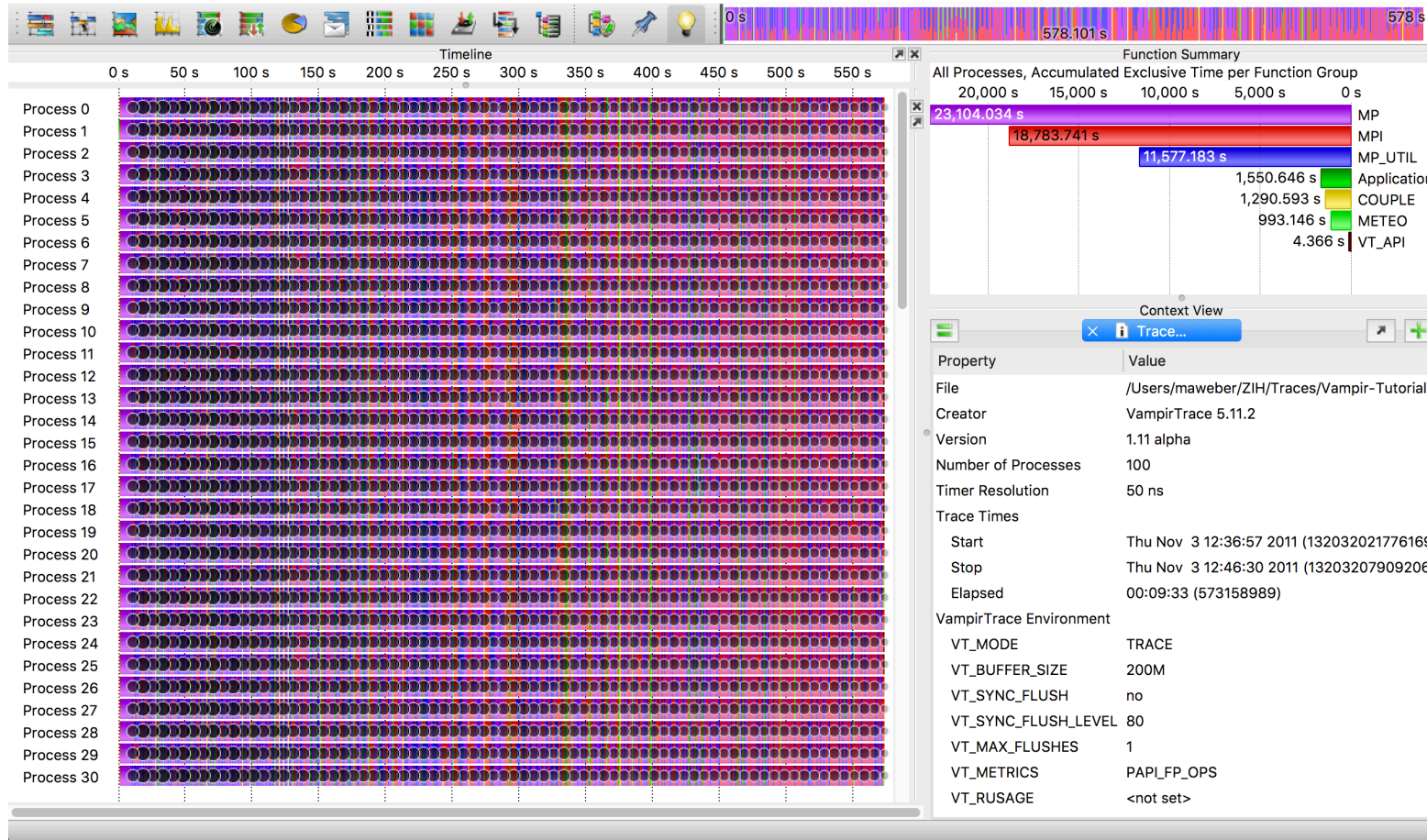


Exercise Trace Files

```
% ls $TW36/tools/vampir/examples/  
  
01-p100-cosmo-specs-orig  
02-p100-cosmo-specs-fd4  
03_wrf_deimos  
04_sbmfd4_jugene  
scorep_tea_leaf_8x3_trace.vampir
```

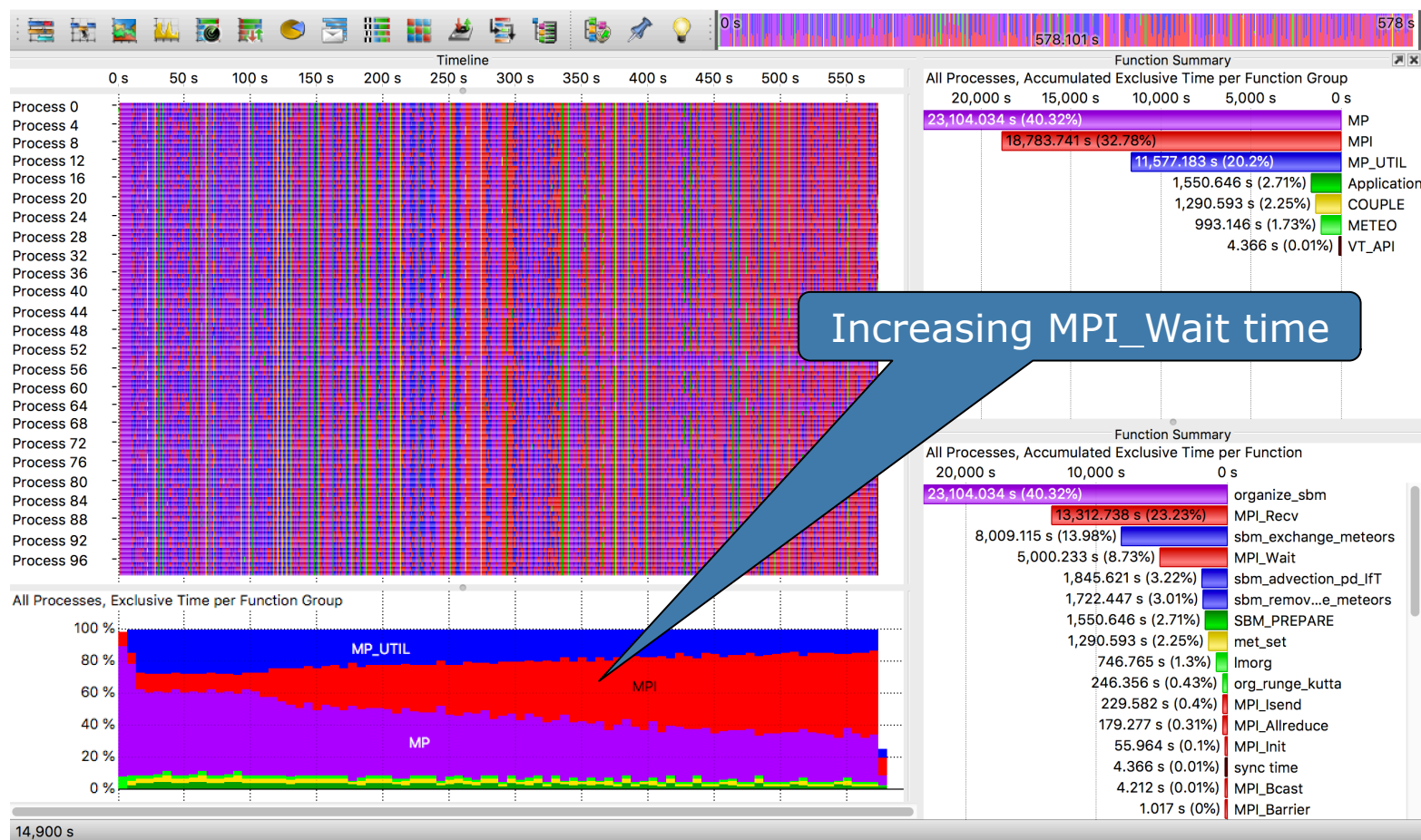
- Four trace files for exercising performance analysis with Vampir
- Traces show real application runs
- Do the traces contain performance problems?
- If yes, try to find their causes

01-p100-cosmo-specs-orig



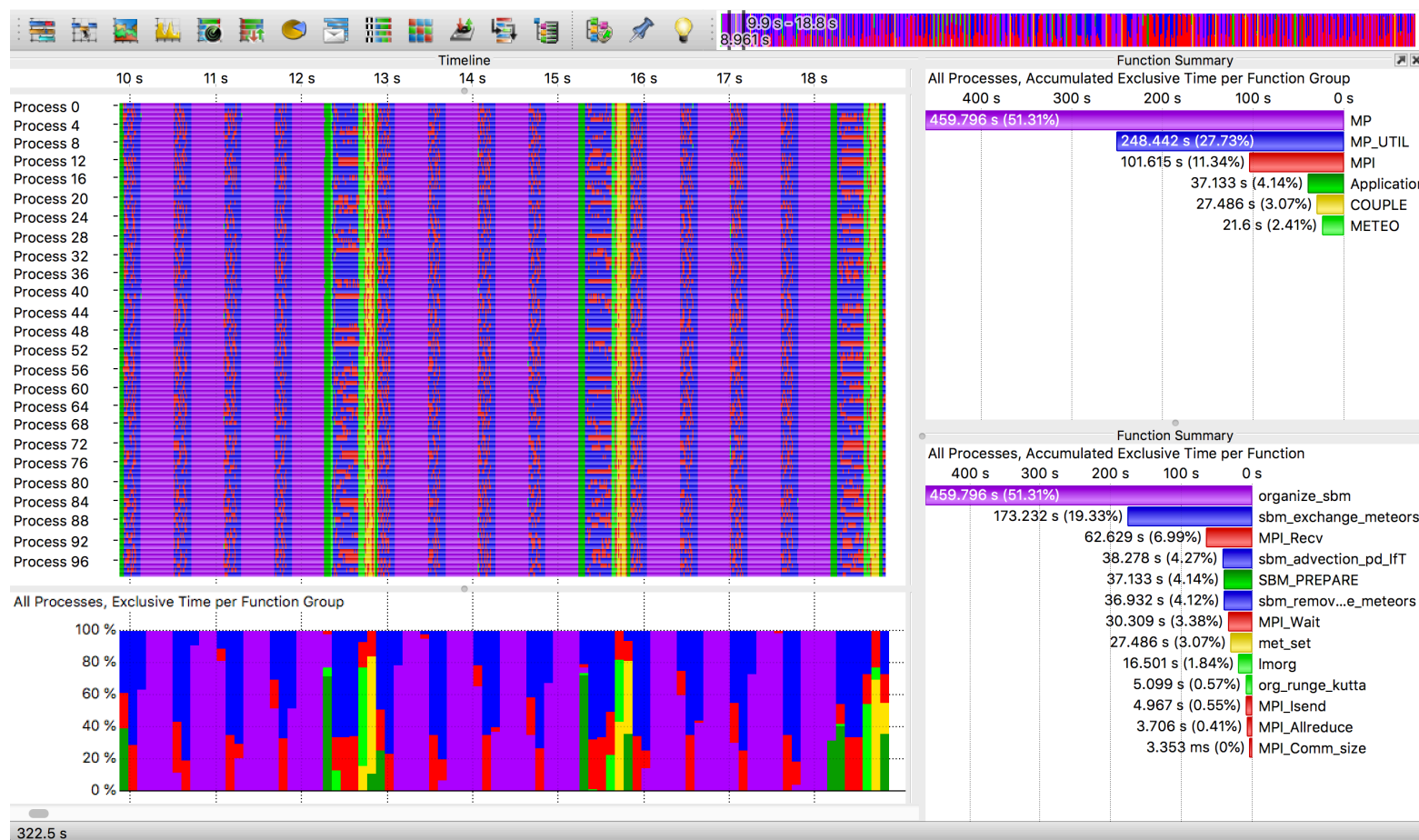
- Weather forecast code COSMO-SPECS
- Run with 100 processes
- COSMO: weather model (METEO group)
- SPECS: microphysics for accurate cloud calculation (MP and MP_UTIL group)
- Coupling of both models done in COUPLE group

01-p100-cosmo-specs-orig



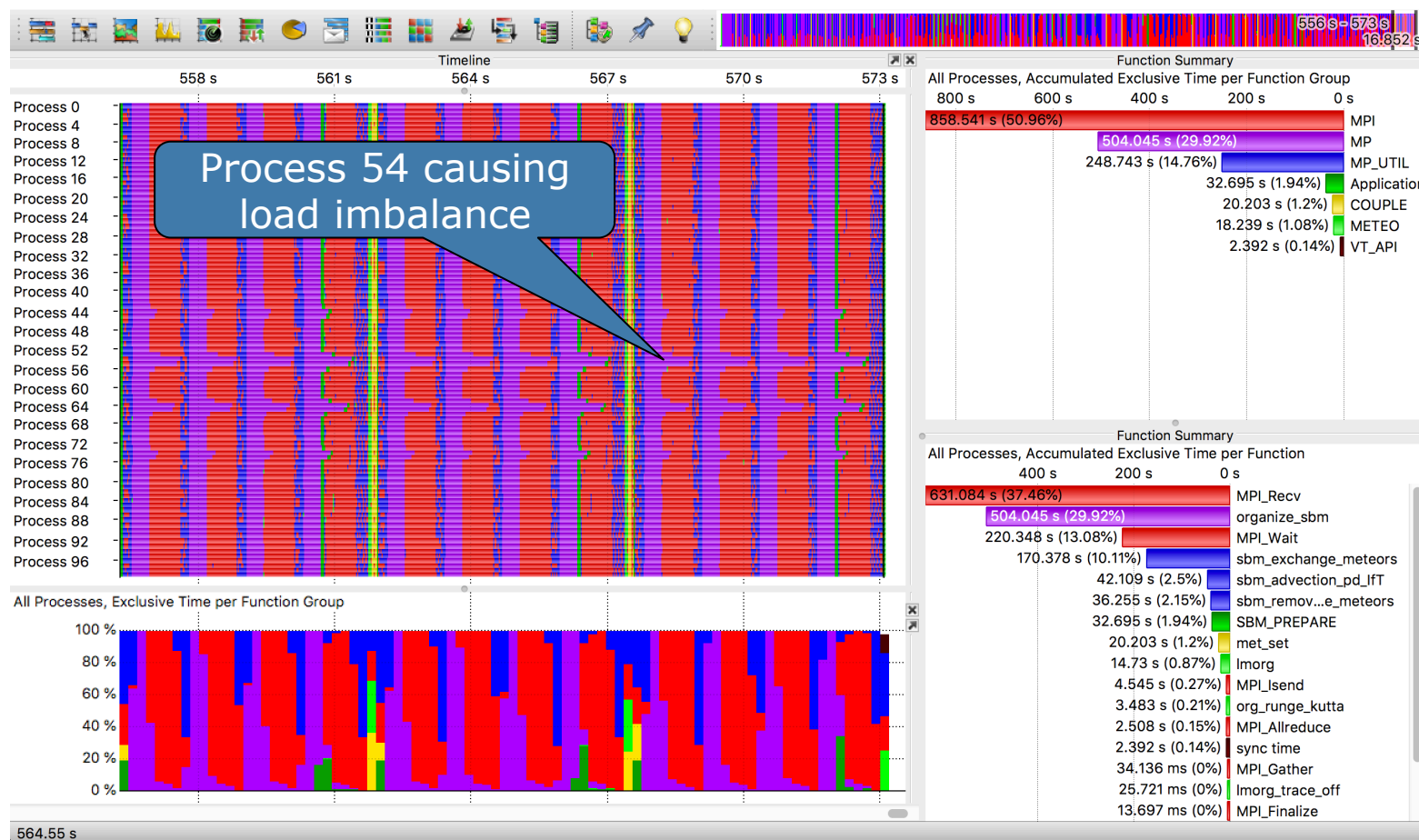
- Compared to METEO, MP and MP_UTIL are very compute intensive, however this is due to more complex calculations and no performance issue
- Problem: >32% of time spent in MPI
- MPI runtime share increases throughout the application run

01-p100-cosmo-specs-orig



- Zoom into the first three iterations
- MP/MP_UTIL perform four sub-steps in one iteration
- Low MPI time share
- Everything is balanced and looks okay

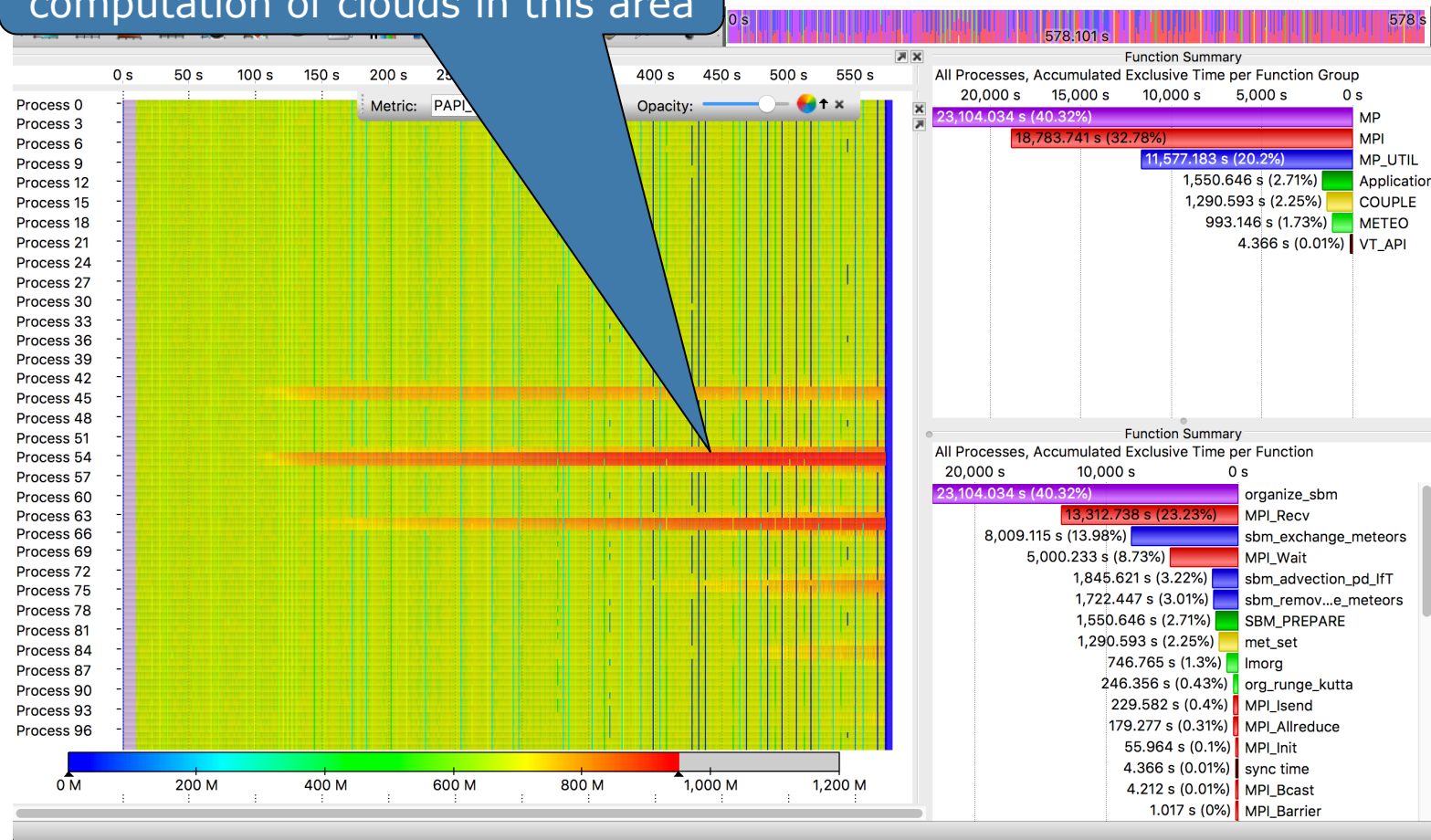
01-p100-cosmo-specs-orig



- Zoom into the last three iterations
- Very high MPI time share (>50%)
- Large load imbalance caused by MP functions around **Process 54** and **Process 64**

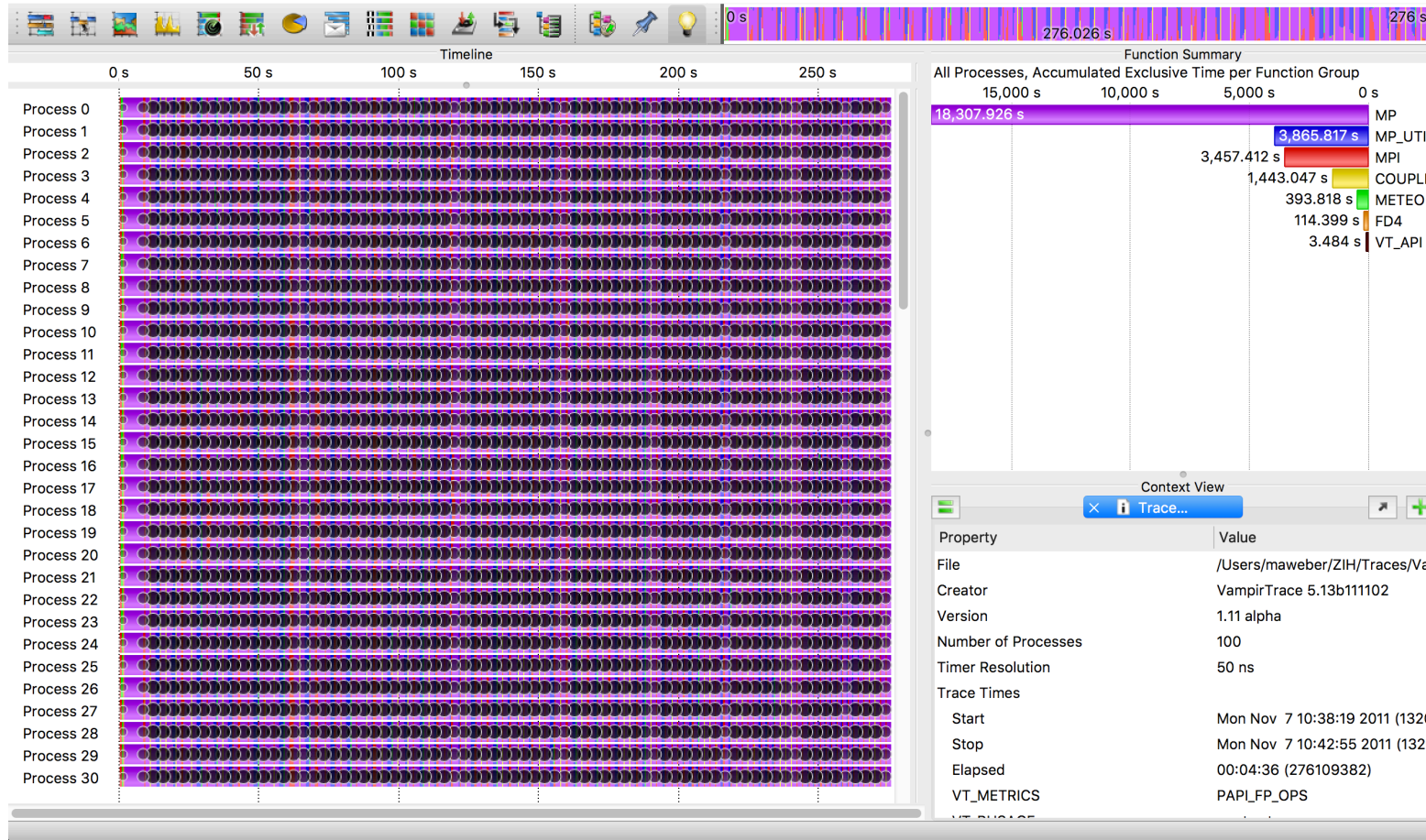
01-p100-cosmo-specs-orig

High FLOPs rates due to computation of clouds in this area



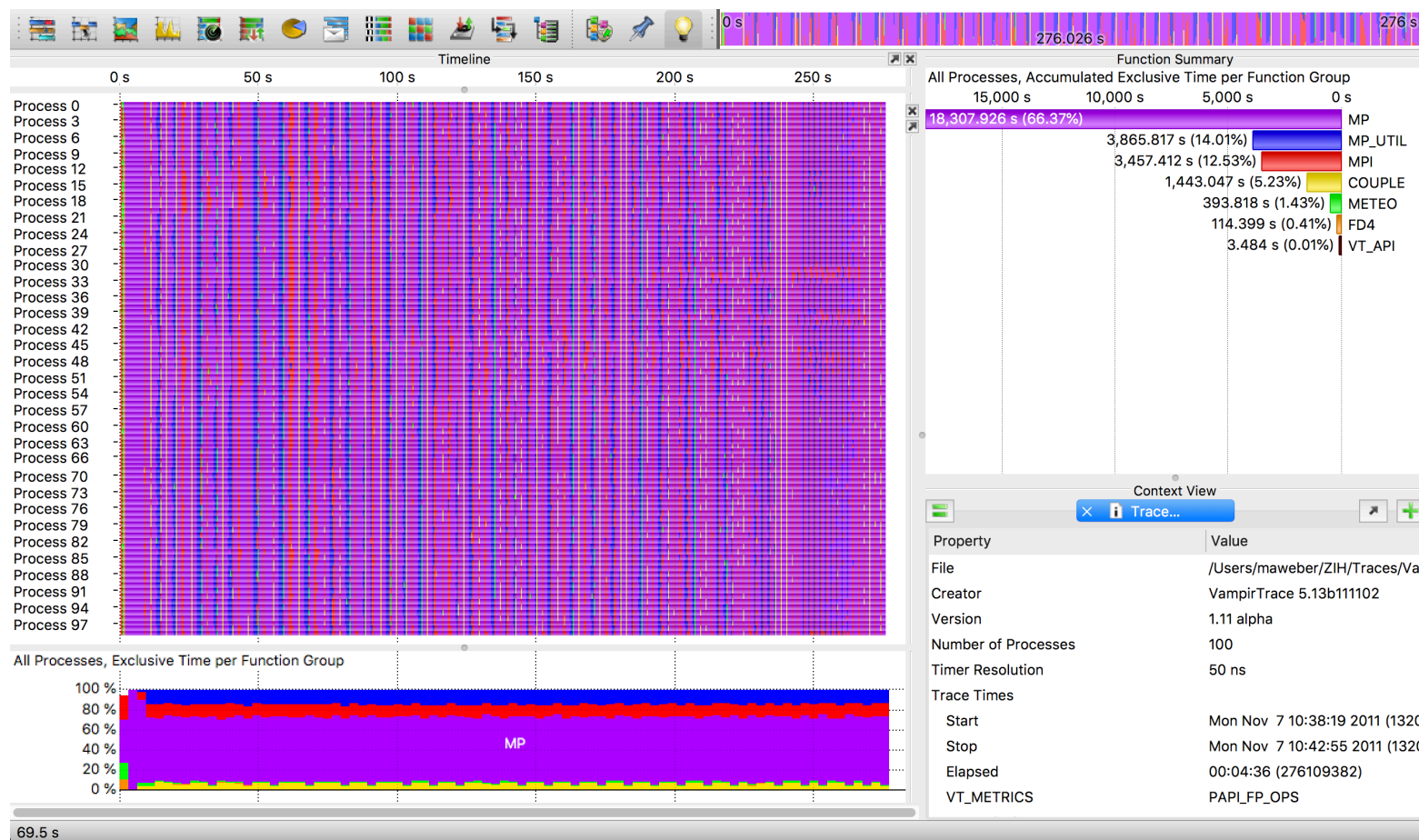
- **PAPI_FP_OPS** counter showing higher FLOPs rates on processes causing the imbalance
- Reason for imbalance: Static grid used for distribution of processes. Depending on the weather, expensive cloud computations (MP group) may be only necessary on some processes

02-p100-cosmo-specs-fd4



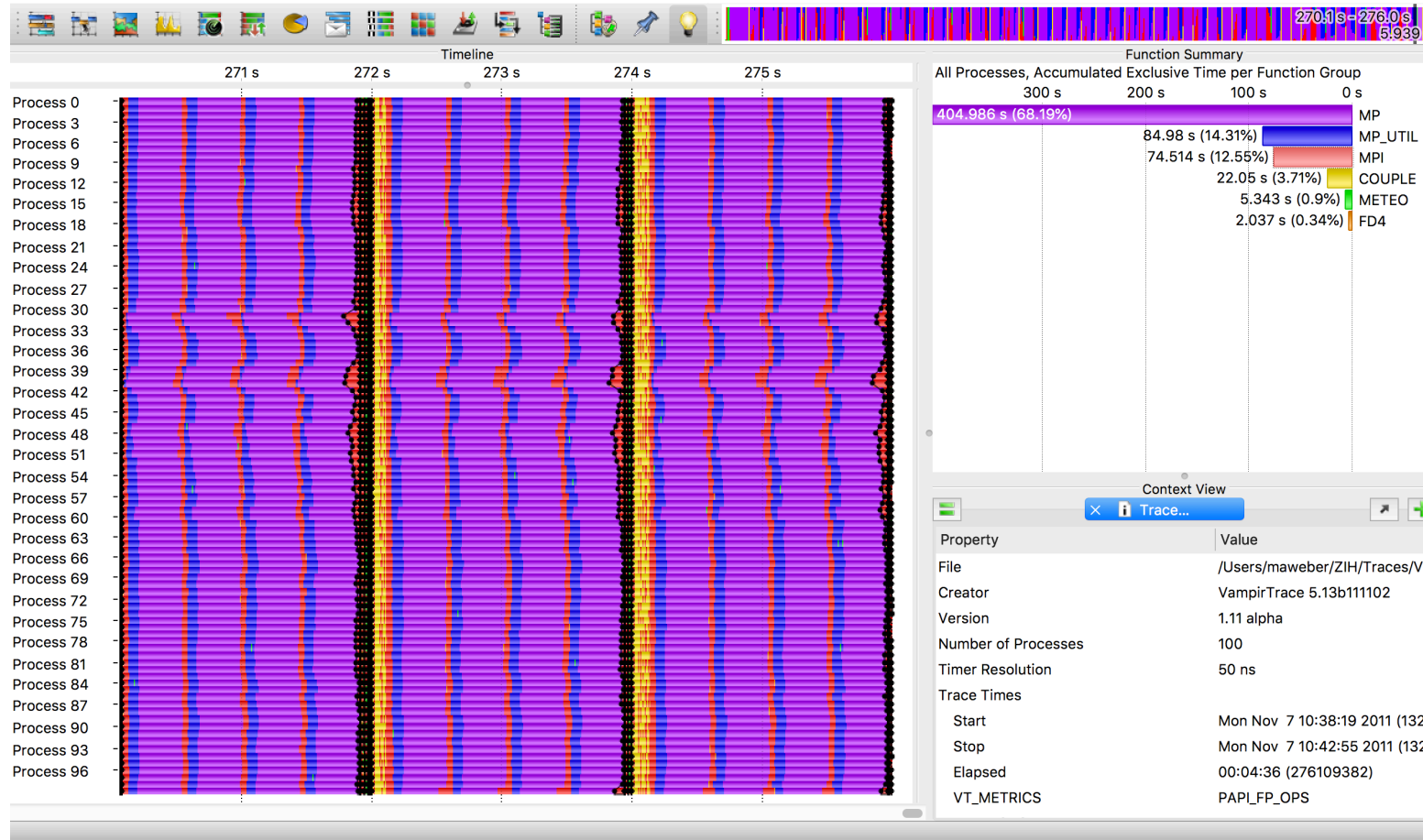
- Weather forecast code COSMO-SPECS
- Run with 100 processes
- COSMO: weather model (METEO group)
- SPECS: microphysics for accurate cloud calculation (MP and MP_UTIL group)
- Coupling of both models done in COUPLE group
- Dynamic load balancing (FD4 group)

02-p100-cosmo-specs-fd4



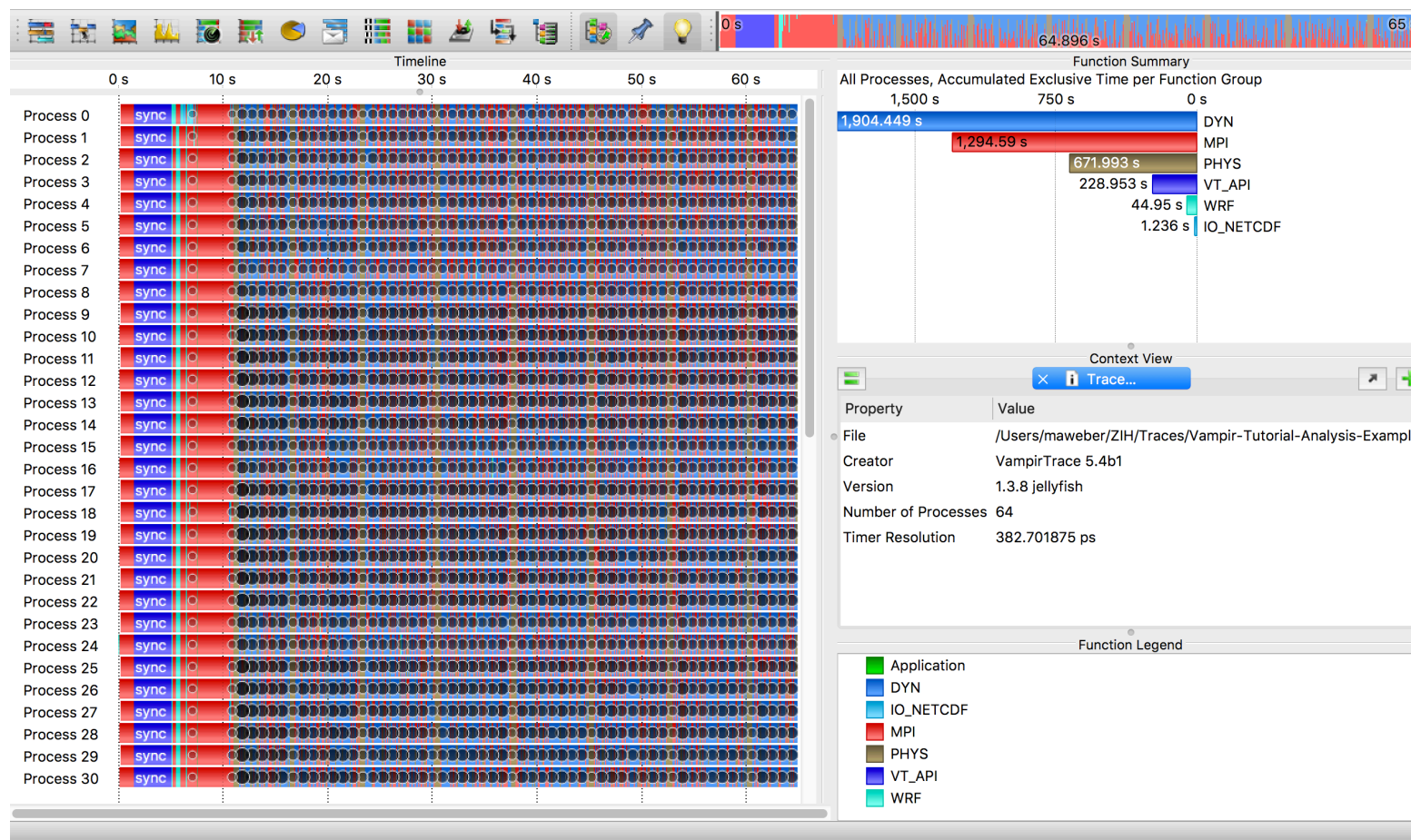
- Dynamic load balancing mitigates the balance problems of the original COSMO-SPECS version
- MPI time share is reduced to <13%
- MPI time share stays constant throughout the application runtime
- Runtime reduced by factor of 2.1, from initially 578s to 276s

02-p100-cosmo-specs-fd4



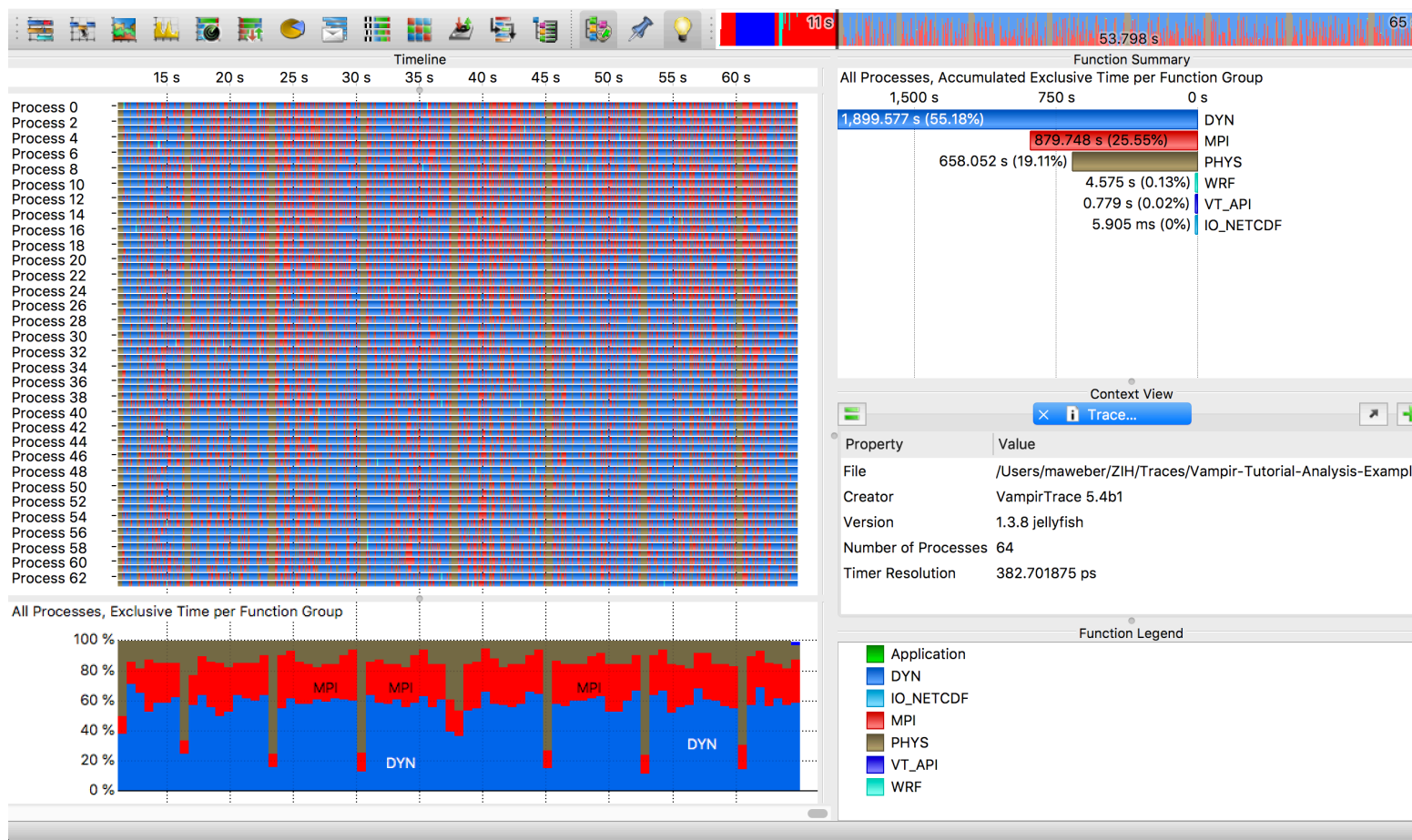
- Zoom into last three iterations
- FD4 balances MP load (precipitation processes in clouds) across all available processes

03_wrf_deimos



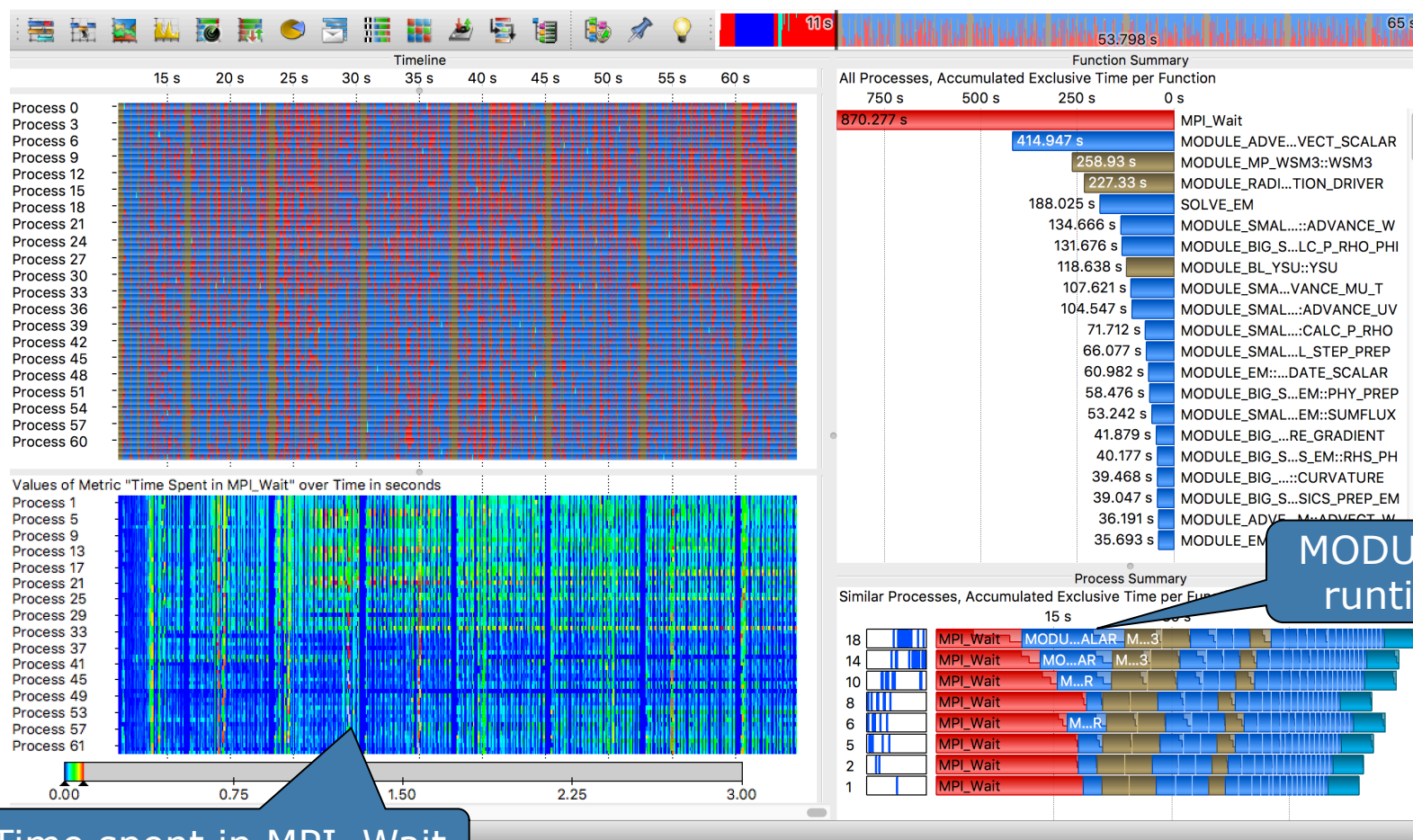
- Weather forecast code WRF
- Run with 64 processes
- *Dynamical core*: e.g., density, temperature, pressure, and winds in the atmosphere (DYN group)
- *Physical parameterization*: e.g., clouds, rain, and radiation (PHYS group)

03_wrf_deimos



- Problem: 25% MPI run time fraction during the iterations of the model
- Behaviour is constant throughout all iterations
- Question: Which user function causes the problem? And why?

03_wrf_deimos

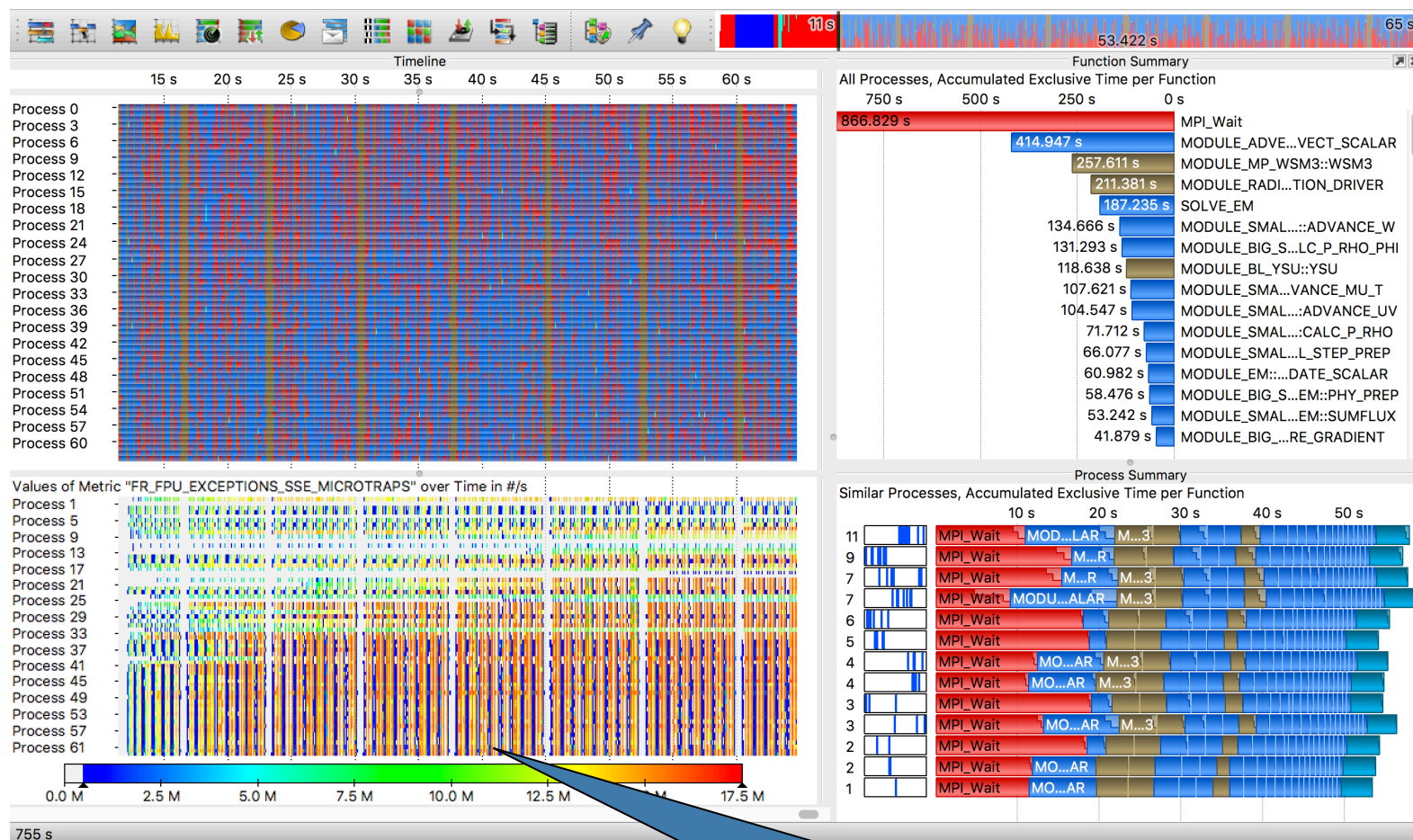


- Most time is spent in **MPI_Wait**
- Top processes spent more time in **MPI_Wait** than bottom processes
- Load imbalance in DYN group

MODULE_ADVECT_EM::ADVECT_SCALAR runtime increases in bottom processes

Time spent in MPI_Wait

03_wrf_deimos

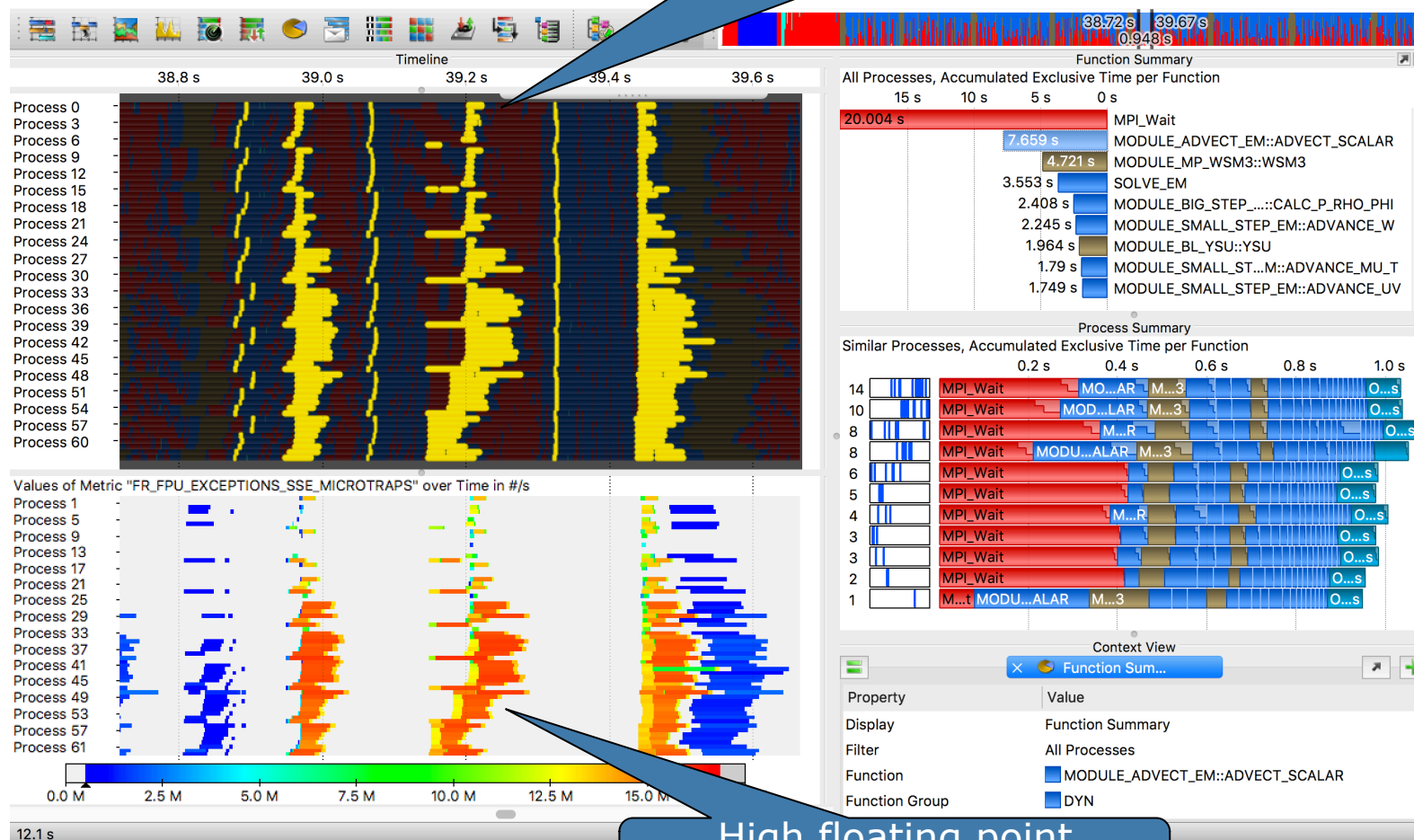


- Load imbalance is caused by floating point (FP) exceptions in WRF
- Counter `FR_FPU_EXCEPTIONS_SSE_MICROTRAPS` shows FP exceptions

Floating point exceptions

03_wrf_deimos

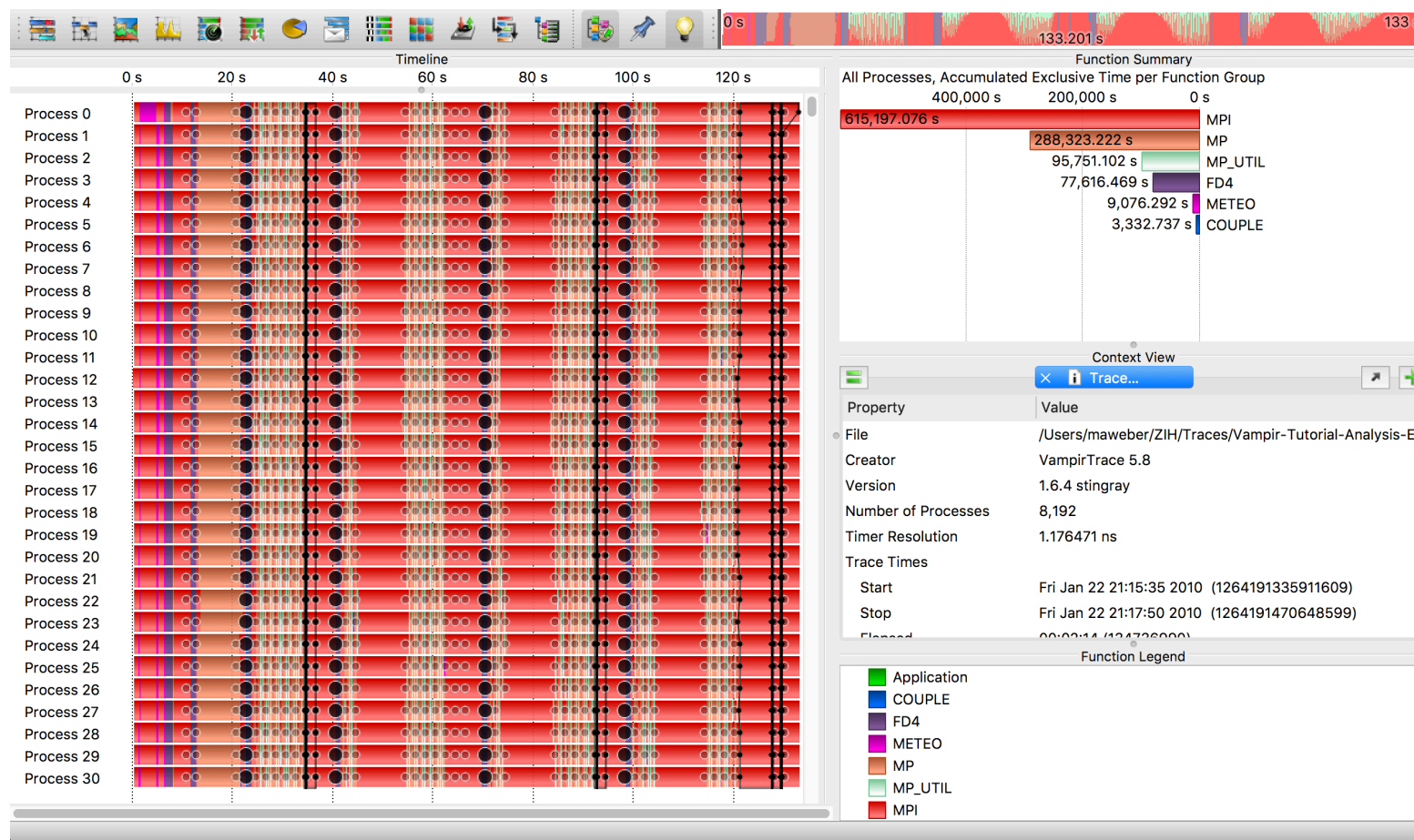
MODULE_ADVECT_EM::ADVECT_SCALAR
occurrences shown in yellow



High floating point
exceptions

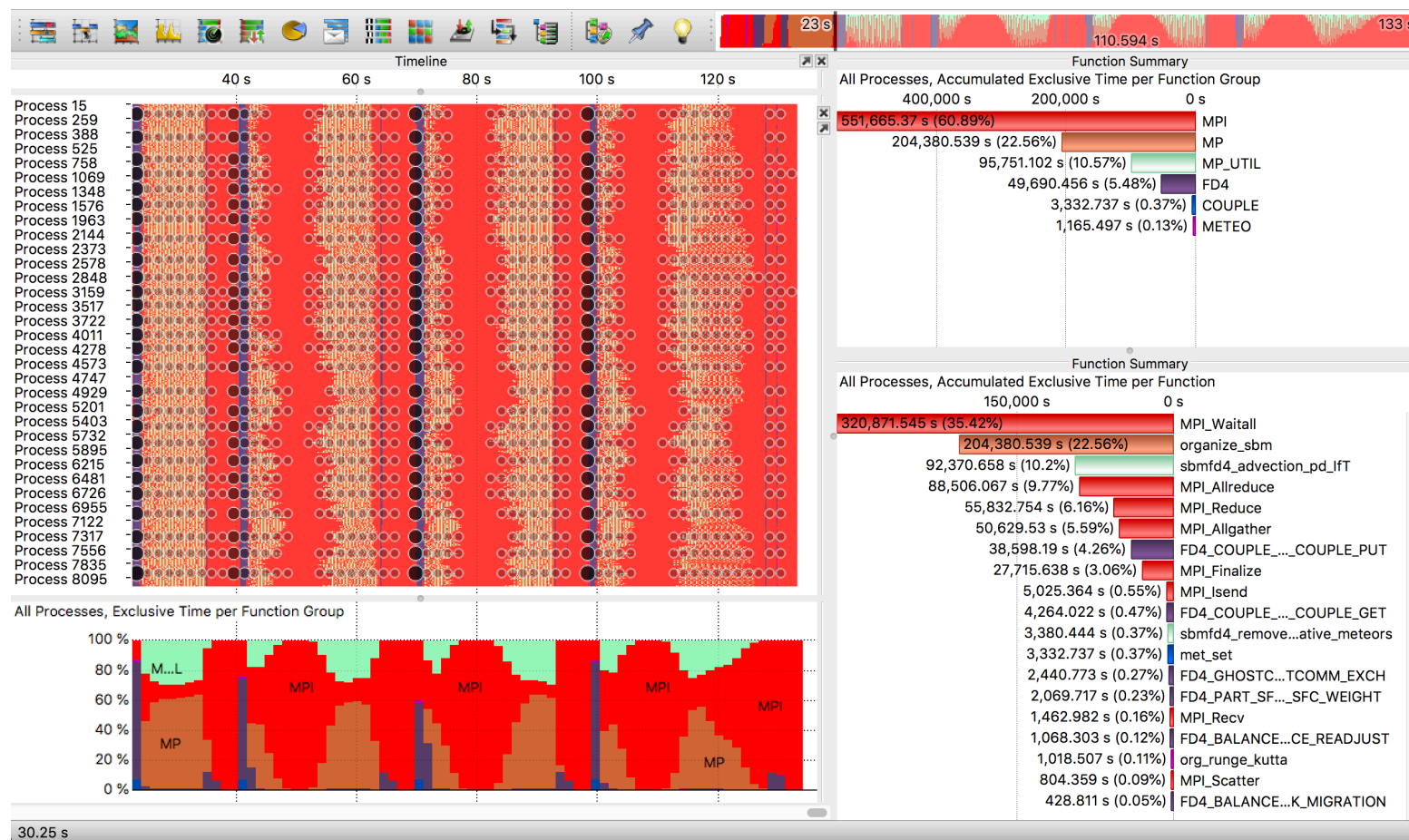
- Zoom into one iteration
- Function invocations of `MODULE_ADVECT_EM::ADVECT_SCALAR` (shown in yellow) match high floating point exception occurrences indicated by the counter at the bottom

04_sbmfd4_jugene



- Weather forecast code COSMO-SPECS
- Run with 8192 processes
- COSMO: weather model (METEO group)
- SPECS: microphysics for accurate cloud calculation (MP and MP_UTIL group)
- Coupling of both models done in COUPLE group
- Dynamic load balancing (FD4 group)

04_sbmfd4_jugene



- Problem: Large MPI runtime fraction (>60%) during iterations
- Especially in **MPI_Waitall** and **MPI_Allreduce**
- Behaviour is constant throughout all iterations

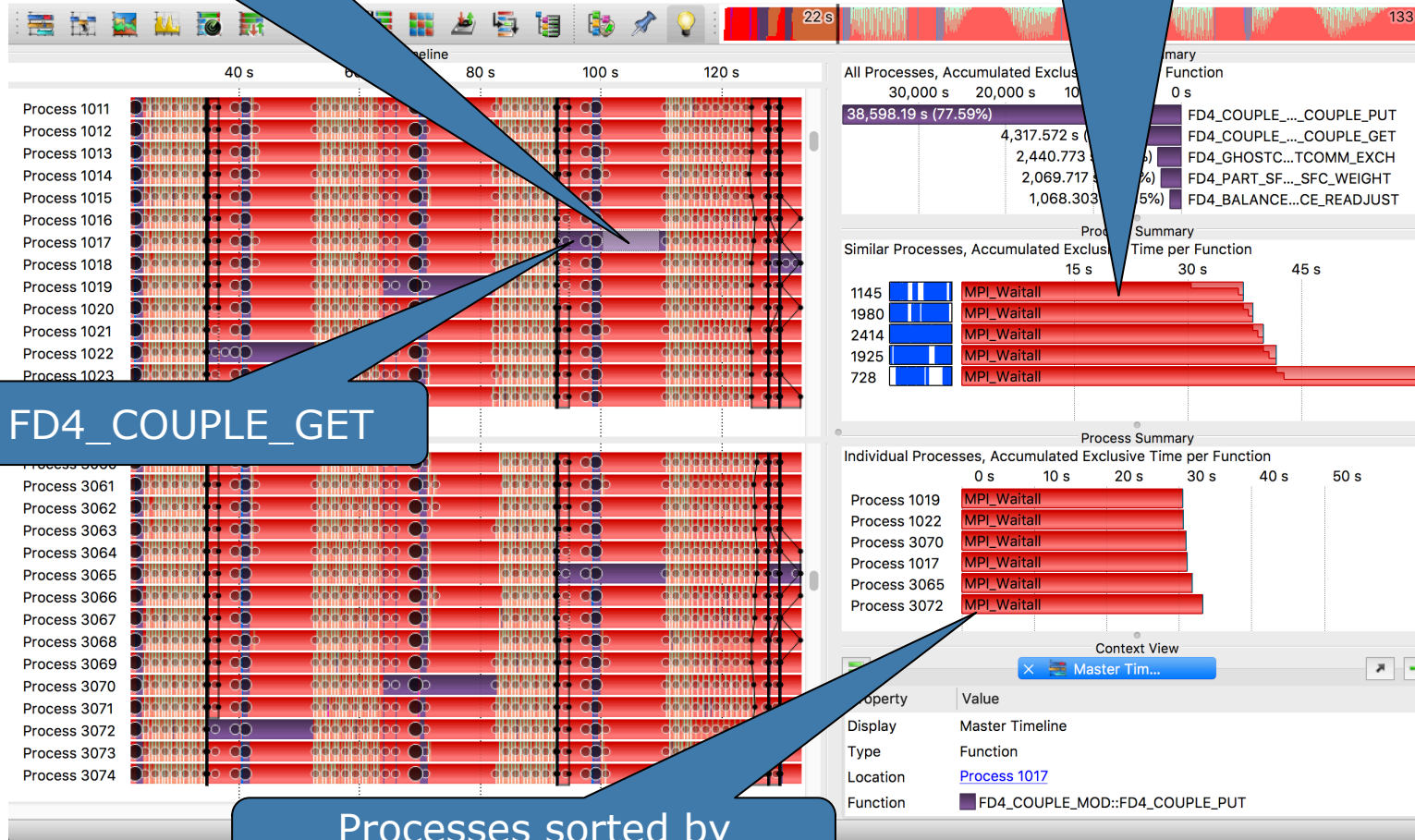
04_sbmfd4_jugene

High MPI_Allreduce
variance between processes

FD4_COUPLE_PUT

FD4_COUPLE_GET

Processes sorted by
MPI_Allreduce timeshare



- Large runtime variation in **MPI_Allreduce**
- Sorted profile reveals processes with small **MPI_Waitall** timeshare
- Reason: Load imbalance in **FD4_COUPLE_PUT** and **FD4_COUPLE_GET**
- Most processes need to wait at **MPI_Allreduce** and **MPI_Waitall** (asynchronously)

Summary

- Performance flaws can lead to significant runtime overheads
- Use resources efficiently
- Analyze your code
- Performance analysis tools are there to help you

