Technische Universität Dresden

ZIH

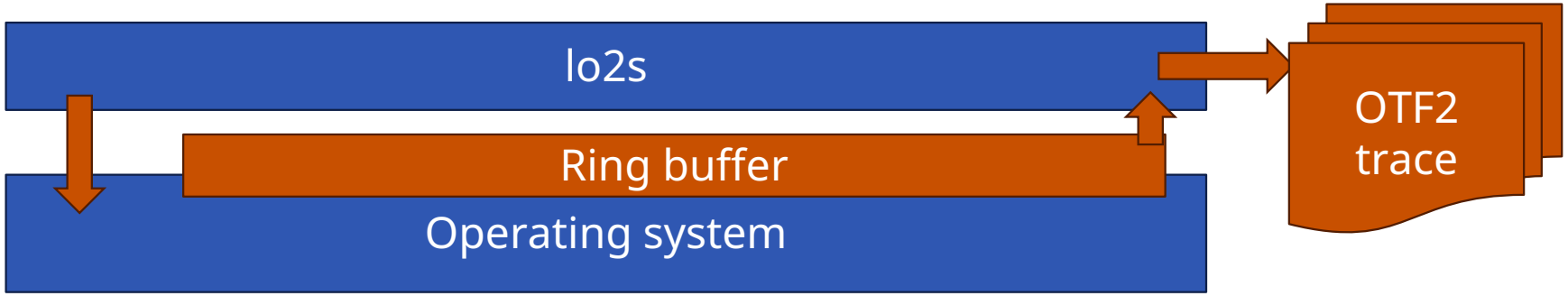
Towards Transparent Continuous Performance Monitoring in HPC Systems through Batch System Integration

International Parallel Tools Workshop 2026, 2026-10-02, Dresden

Background lo2s

- Performance Monitoring Tool
- Usage close to perf record, writes OTF2 traces
- Sampling events to interrupt workloads and store runtime information
- Other events (kprobes, uprobes, Score-P metric plugins)

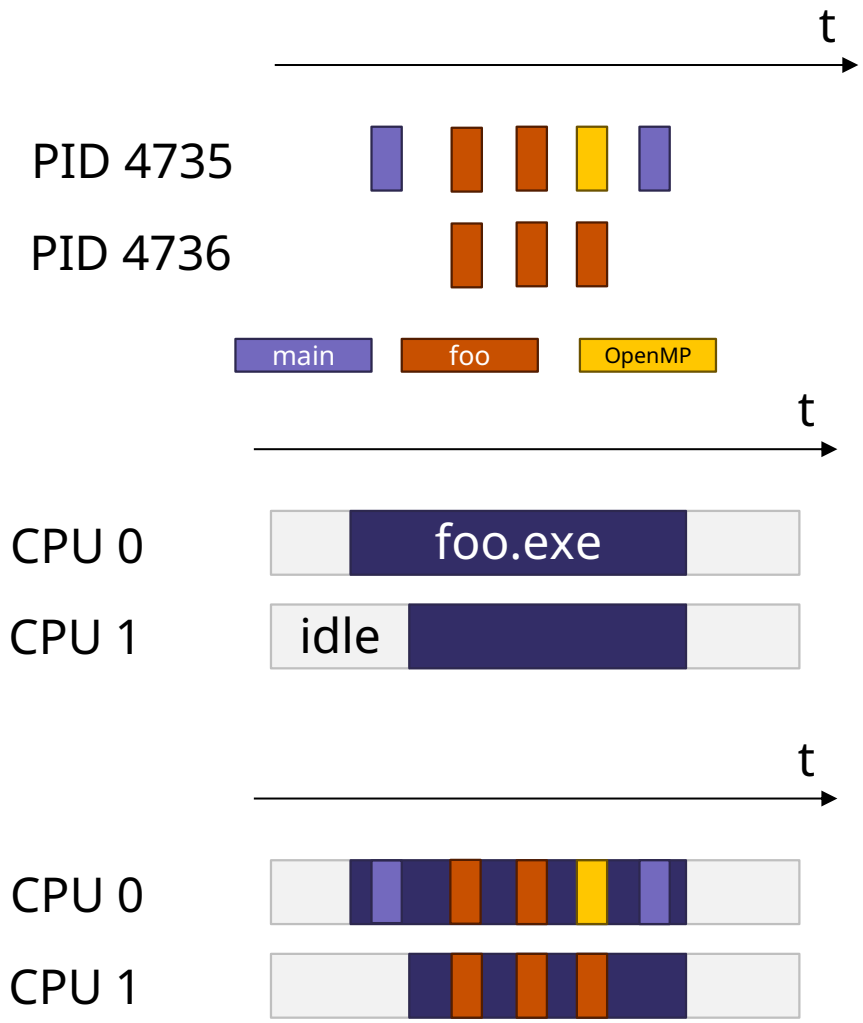
Ilsche, T., Schöne, R., Bielert, M., Gocht, A., & Hackenberg, D. (2017, September). ***lo2s—multi-core system and application performance analysis for linux***. In *2017 IEEE international conference on cluster computing (CLUSTER)* (pp. 801-804). IEEE.



Background lo2s

- Mode 1: Per process (`-p`)
  - Samples process (and it sub-processes/threads) events
  - One location per PID
- Mode 2: Overall (`-a`)
  - Checks scheduling of processes and threads → instrumented events
  - One location per CPU
- Mode 3: Overall with sampling (`-A`)
  - Combines both information
  - One location per CPU

Mode 2 and 3 need elevated privileges



Background Slurm

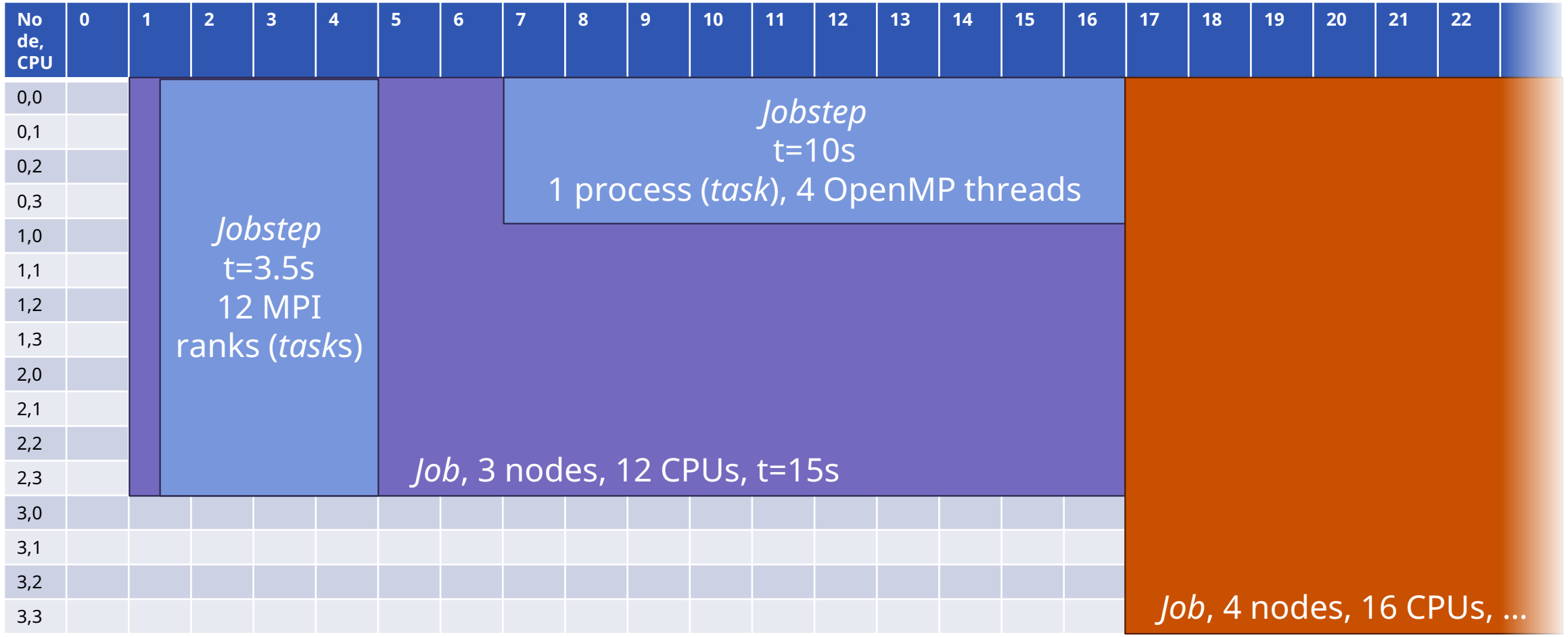


- Workload Management System
- Ressource Manager
- Job Scheduler
- Used by “over 60% of the world's top clusters and supercomputers” [1] or “over half of the top 100 systems in the TOP500”[2]
- Open Source

Jette, M., & Grondona, M. (2002). ***SLURM: Simple Linux Utility for Resource Management*** (No. UCRL-MA-147996-REV-2). Lawrence Livermore National Lab., CA (US).

[1]https://www.chpc.utah.edu/documentation/software/slurm.php
[2]https://www.nvidia.com/en-us/software/slurm

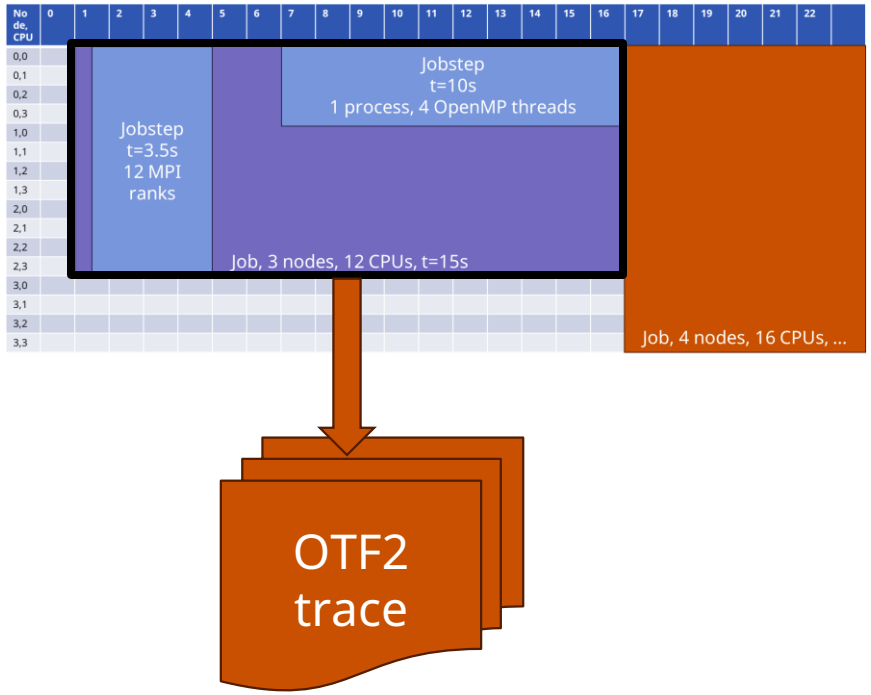
Overview (*italic* → Slurm term)



Vision

Enable users to

- Record job related data
- Including all job steps
- Also from other processes

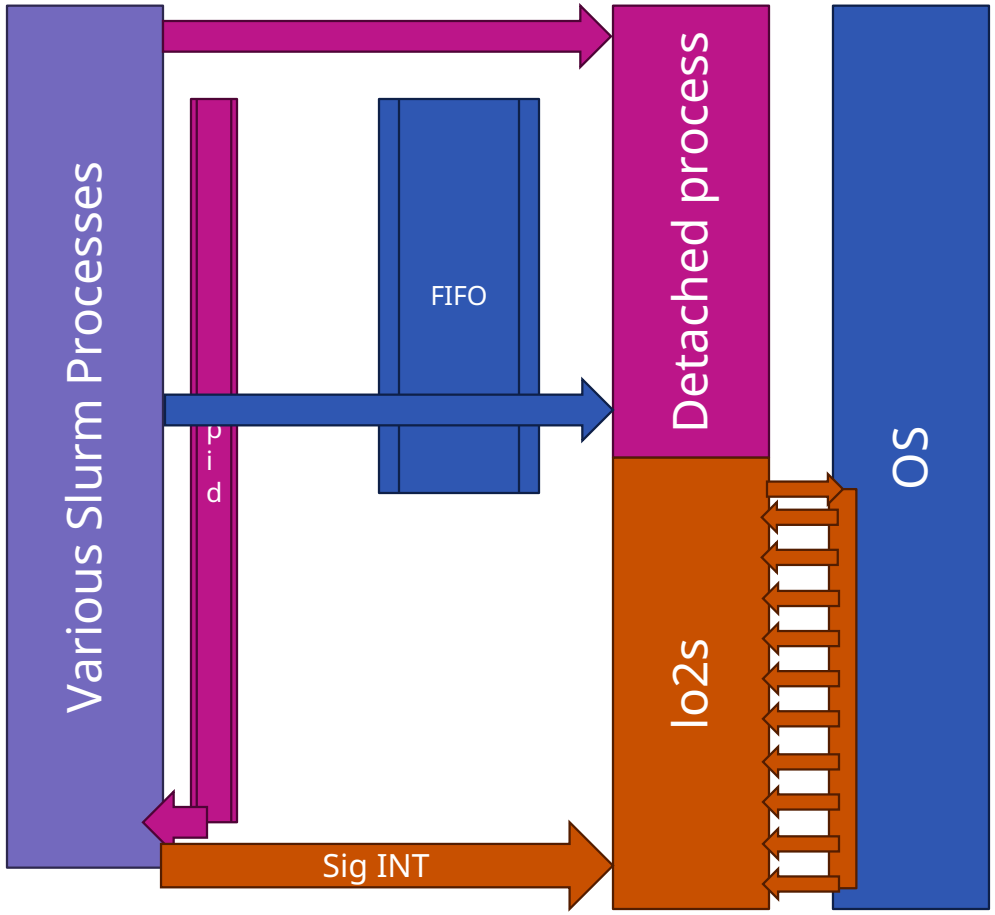


Slurm Plugins

- So many interfaces
- Relevant for us: Spank plugin
  - Called on various times during initialization and finalization of jobs / job steps
  - With elevated privileges and user privileges
- and somewhat: cgroups
  - Attributes hardware to jobs
  - We would not want to attribute the monitoring to other(s peoples) CPUs
  - We would want to use the same resources as the Job (If the user wants monitoring, they should pay the resources.)

Implementation (runs per host)

- Job prolog
  - Elevated privileges
  - Fork process, store pid in file
  - Read lo2s option from FIFO file
  - Pin to job cgroup, execve lo2s, monitor jobs cgroup
- Job step start (job step 0)
  - **User privileges**
  - Only if first job step
  - Write lo2s option to FIFO file (or CANCEL)
- Job exit
  - Elevated privileges
  - Stop lo2s process using pid
  - Clean up and chown trace



Registered Slurm Options

- Result folder (`--lo2s=…`)
  Resulting traces will be written to this folder. The plugin will check whether the job's user is allowed to write to this folder and cancel if it is not writable. Currently, it is written to `$FOLDER/$SLURM_JOB_ID/$HOSTNAME`
- Sampling event (`--lo2s-event=…`)
  Sets the metric that is used to generate sampling signal, default: instructions
- Sampling event rate (`--lo2s-event-rate=…`)
  Sets the event rate at which the event triggers
- Metrics (`--lo2s-metrics=…`) and their frequency (`--lo2s-metrics-frequency=`)
  Sets additional hardware metrics to be measured at a certain frequency

Example

```
#!/bin/bash
#SBATCH …

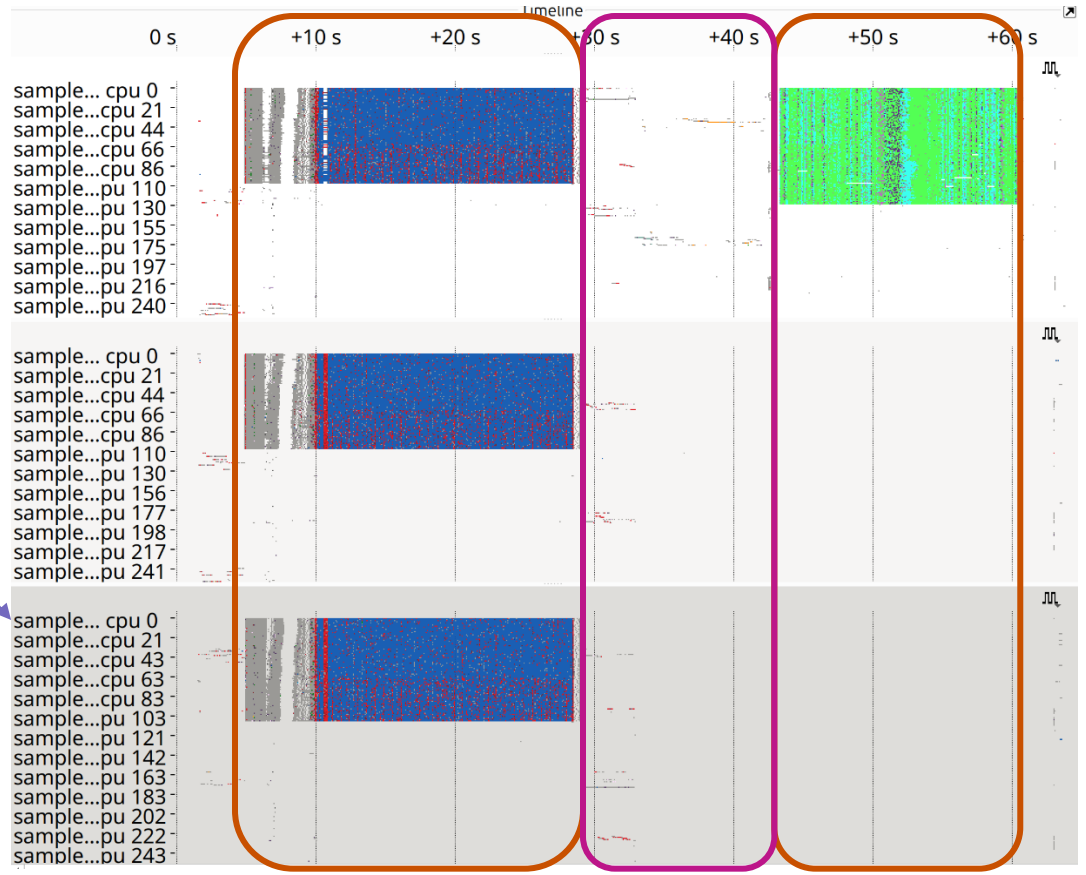
cd NPB3.4-MPI
module load release/2026 GCC/14.3.0 OpenMPI
make clean
make bt CLASS=C
srun --lo2s=/home/rschoene --lo2s-metrics=cache-misses -N 3 -n 324 ./bin/bt.C.x

module unload release/2026 GCC/14.3.0 OpenMPI
module load intel/2025a
cd ../NPB3.4-OMP
make clean
make bt CLASS=C
srun -N 1 -n 1 -c 128 --lo2s-event=cpu-cycles ./bin/bt.C.x
```

BAILEY, D. (1995). ***The NAS Parallel Benchmarks 2.0***. *Technical Report NAS-95-020, NASA Ames Research Center*.

Overview

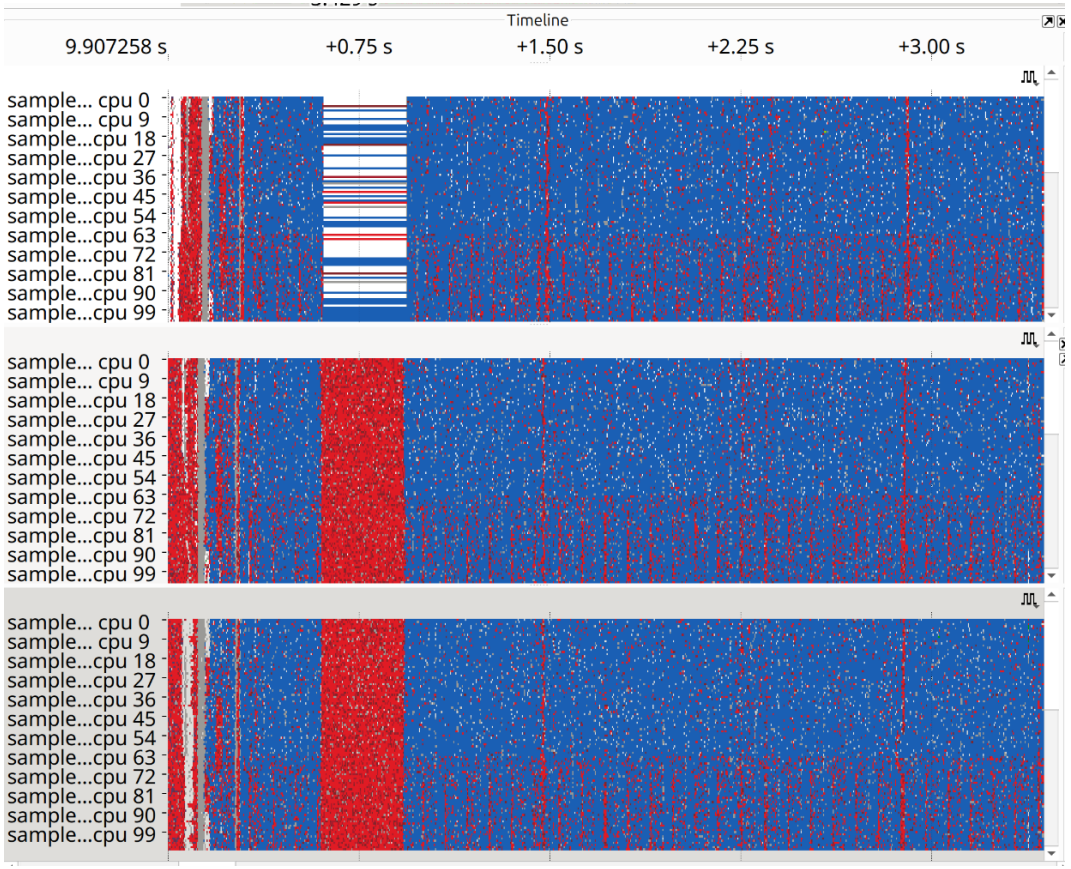
- Three traces (-N 3), 256 CPUs each (2 sockets, each 64 cores + SMT)
- Two main phases (2x srun)
  - MPI parallel
  - OpenMP parallel
- In between:
  - make process
  - slurm processes



Details MPI

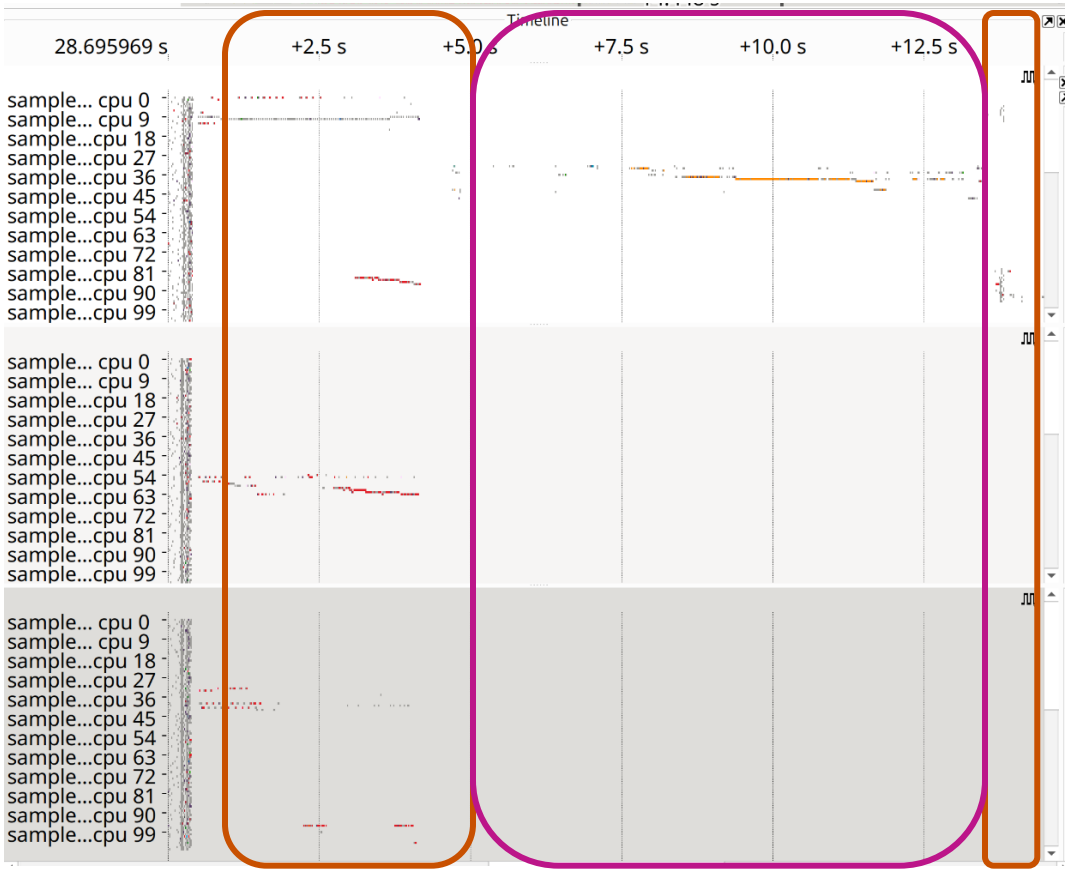
- Issue on first node leads to more samples hitting MPI functions [1] (red) at t=10.7s on other nodes
- CPUs on 2nd socket spend more time in MPI (higher CPU numbers in each trace have higher share of red)
- Vertical lines in red indicate global synchronization

[1] Libraries and functions from ucp/uct/orte/opal/mpi/...



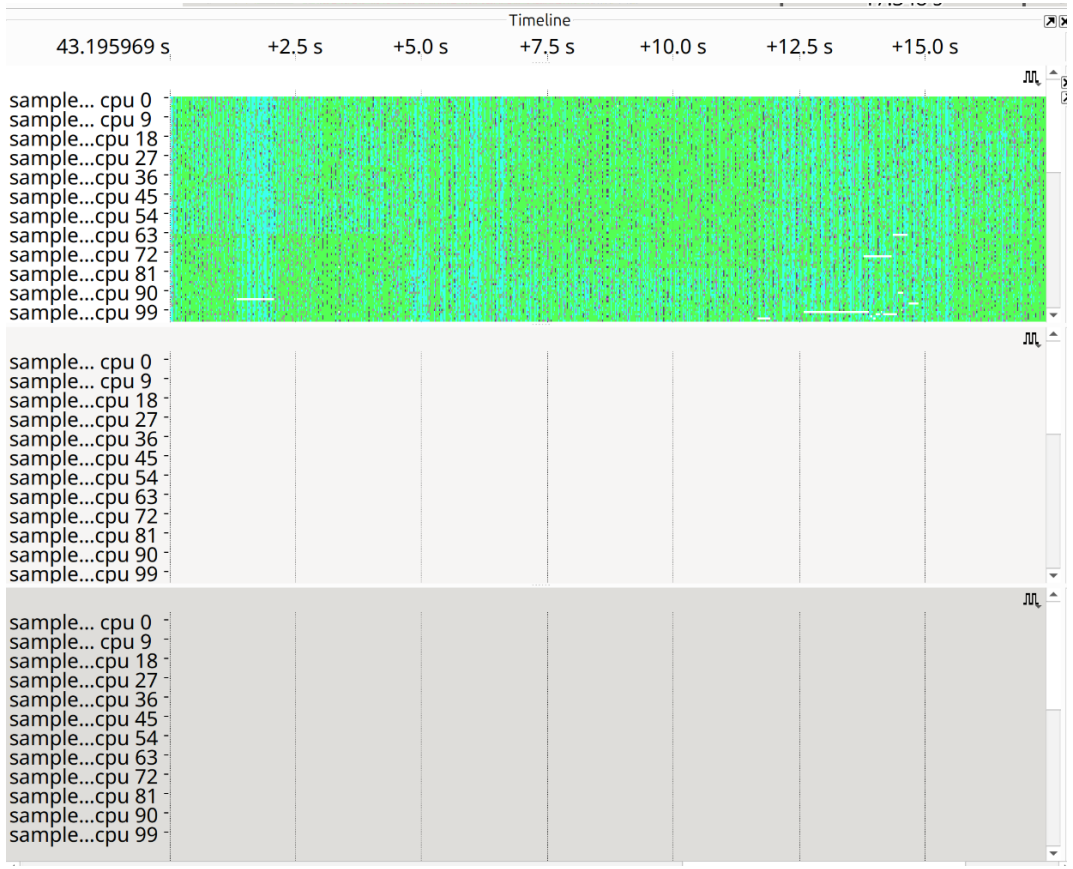
Details in Between Job Steps

- First phase:
  - Fade out old MPI job step,
  - Includes Lua and Python calls
- Second phase:
  - Build new binary
  - Includes calls to linker and compiler
- Third phase:
  - Preparation of new job step
  - Again lua/Python



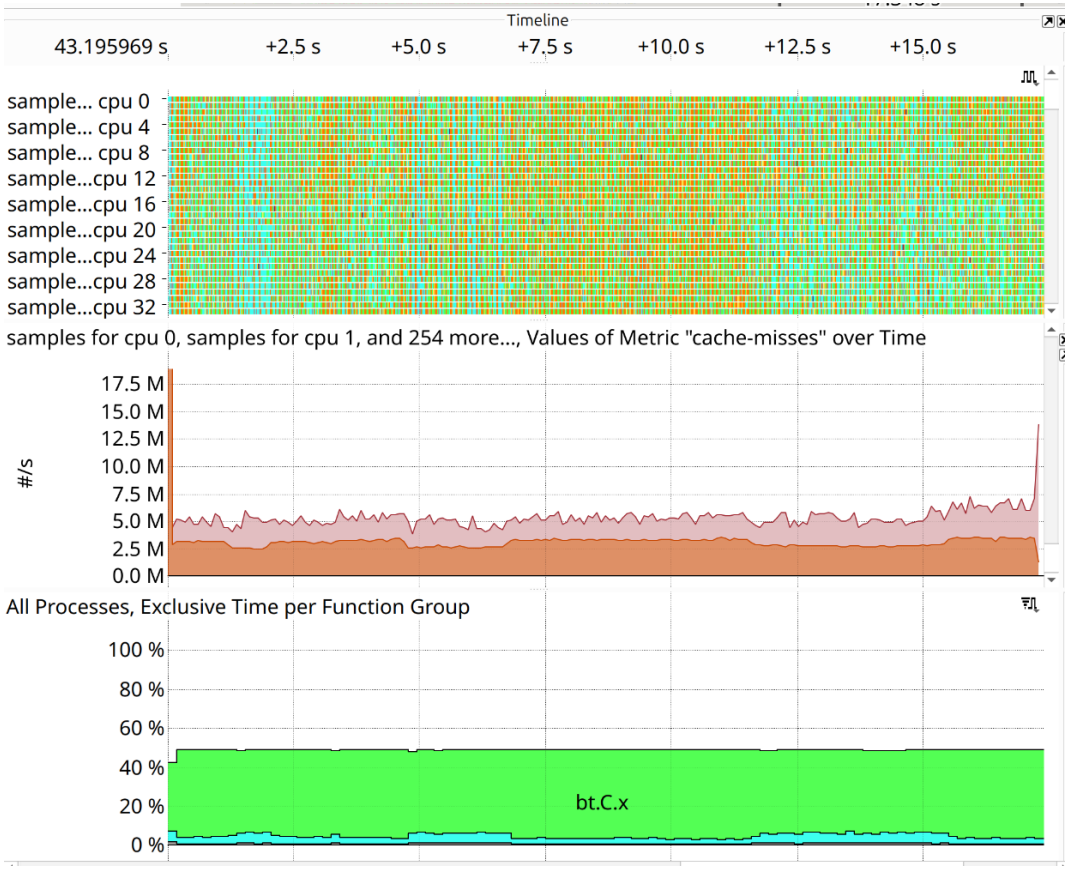
Details OpenMP

Only at one node, so lets use the vertical space for something else …



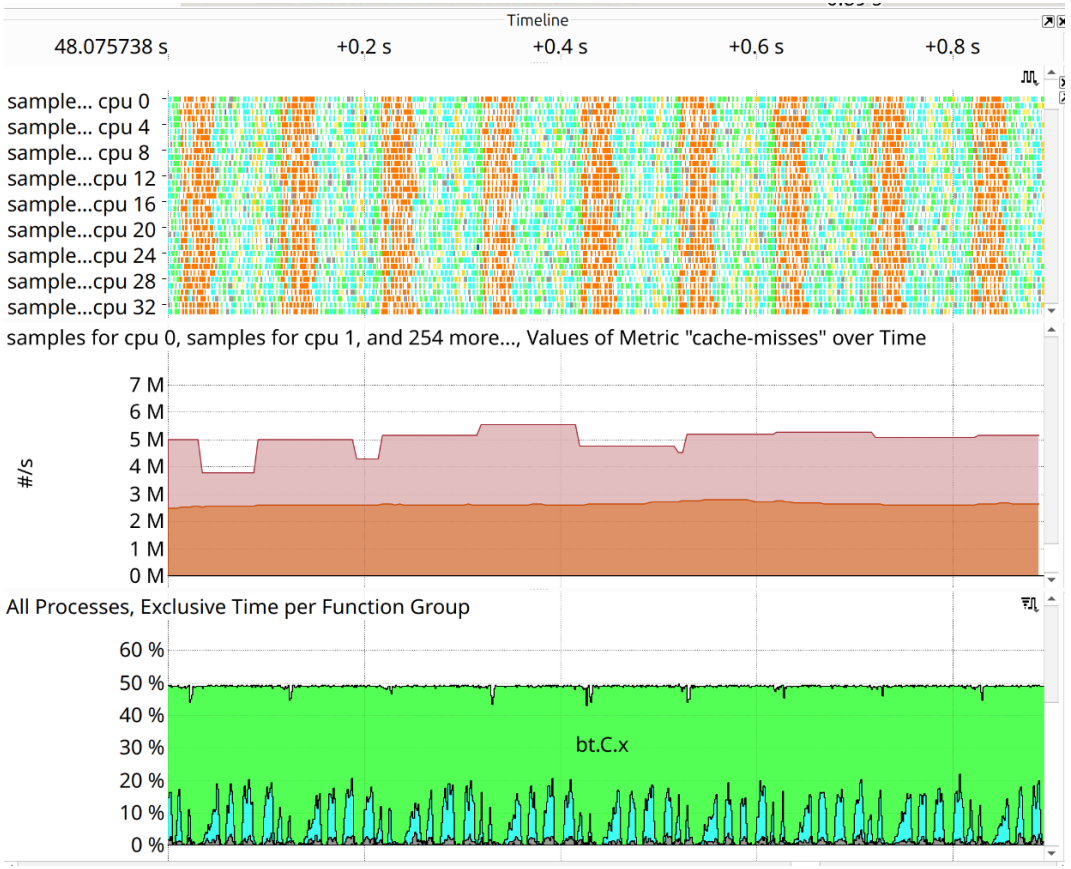
Details OpenMP

- Lot of different functions
- About 2-5 M cache-misses per second
- 50 % of CPUs used (since HT not used)
- Less cache misses, when in OpenMP library (cyan)



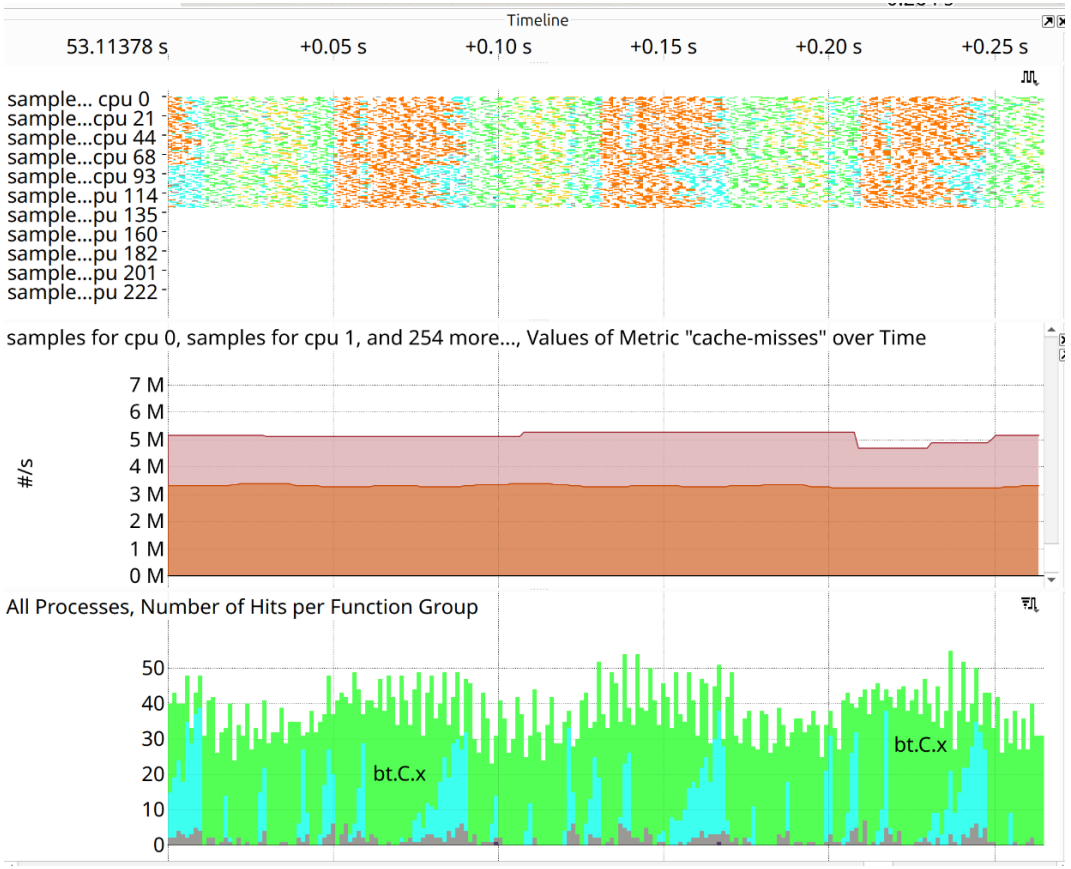
Details OpenMP

- Repetitive pattern
- Regular phases with OpenMP synchronization (bottom panel)



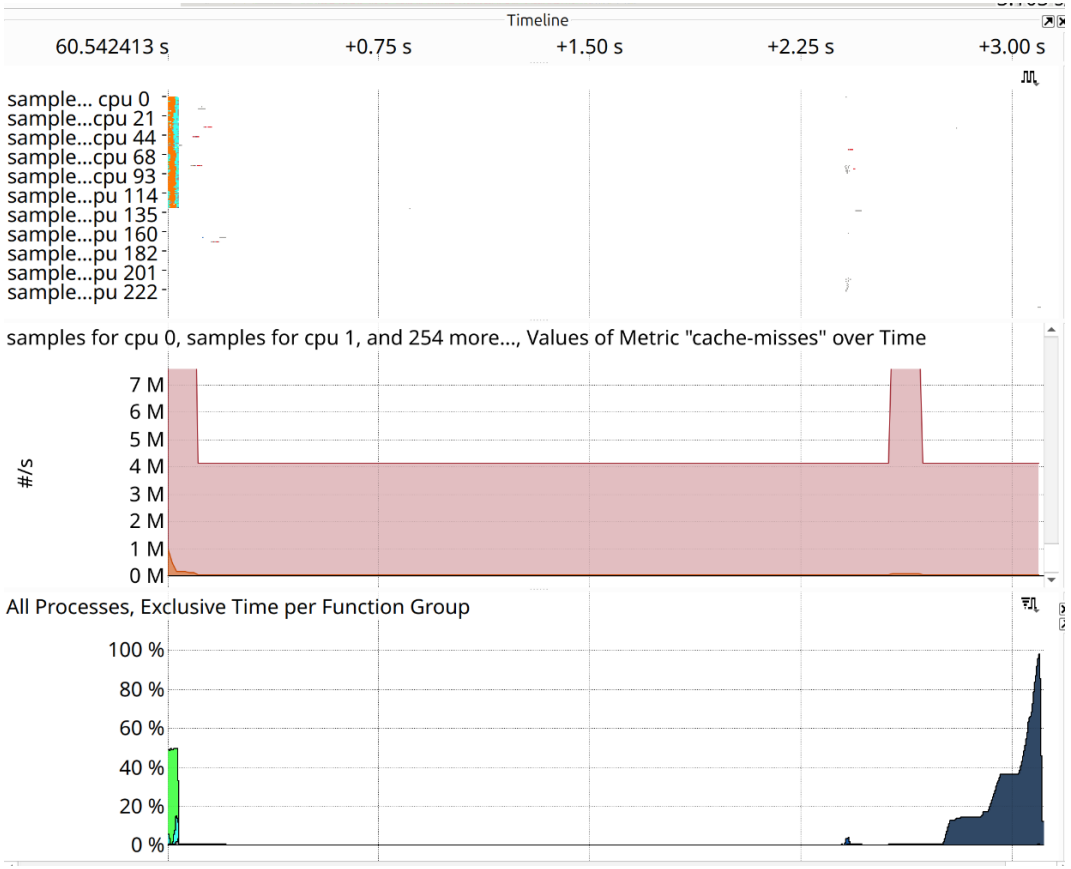
Details OpenMP

- Repetitive pattern
- Regular phases with OpenMP synchronization (bottom panel)
- Most OpenMP synchronization at end of `compute_rhs` (brown functions)
- In between OpenMP synchronization after `x_solve` , `y_solve` , `z_solve`, …(light green, yellow, light orange, …)



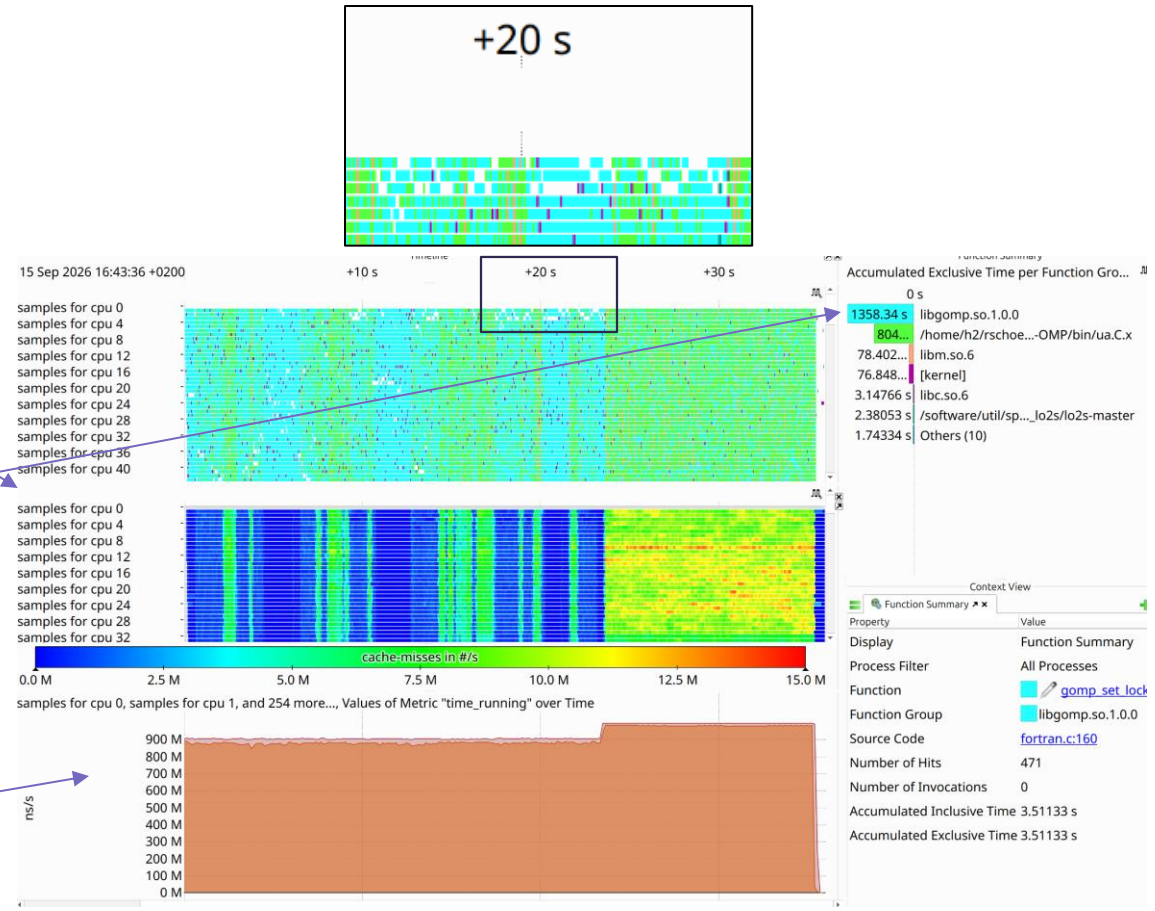
Finalization

- Lot of lo2s (dark blue, bottom panel)
- Flushing of data
- Not visible in top panel, since only samples are shown there (can be toggled). This information is derived from the scheduling info in the trace.



2nd Example. OpenMP-parallel Job Step (NPB ua)

- Low number of cache misses in first phase ☺
- … due to high usage of OpenMP ☹
- Why?
  One CPU is used for something else until second 22

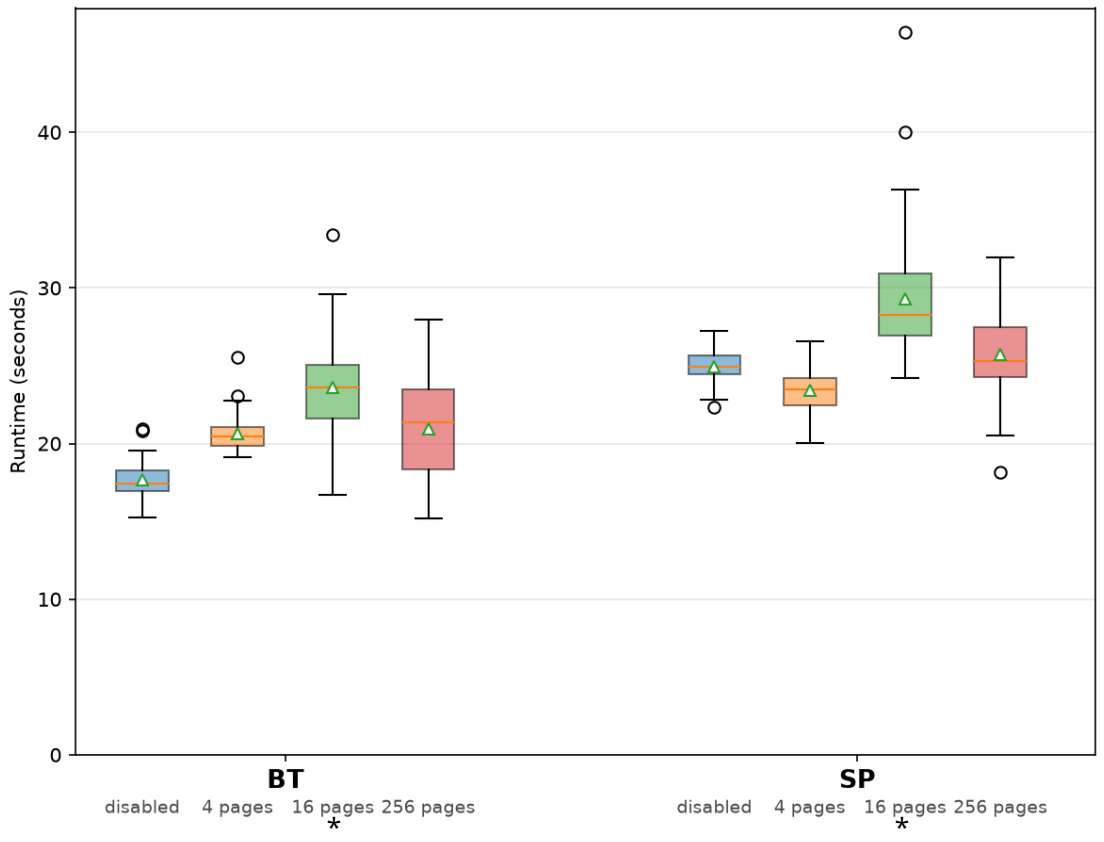


Overhead

- Different factors
- Slurm overhead
  - Job starts:
  - Job step starts/ends: instant check whether it is the correct job step
  - Job ends: High overhead due to flushing of data
- Job overhead
  - Processing performance lower, since CPU cycles spent with monitoring instead of calculation
  - Memory consumption higher to buffer events – can have indirect effects

Runtime Overhead Analysis

- 50 Executions of bt and sp of NAS parallel benchmarks (V3.4, OpenMP parallel)
- Default lo2s configuration from spank plugin (no metrics, ~182 samples/s),
- Lo2s current github master
- Romeo cluster of TU Dresden, ZIH
  - 2 x AMD EPYC 7702 (64 cores) @ 2.00 GHz
  - 512 GiB of RAM
  - Node i7009 (exclusively)
- Issues:
  - Global communication (here: barriers) sensitive to noise
  - Buffer size matters, 48B per sample, 96B per scheduling event, smaller buffer → missed events, more readouts; larger buffer → longer readouts, finalization
  - Finalizing OTF2 takes long and might be cancelled.



* debug version of Spank plugin

ZIH

Future Work and Thank You

Future Work

- Polishing
- Support Slurm without cgroup plugin
- Give further feedback to lo2s developers
- Pass more lo2s options as Slurm flags (e.g., disable scheduling events)
- Publish at github

- Collecting your ideas

Thanks

- Sponsors (for funding)
  - EU: 101177590 (Seanergys)
  - DFG: 545776403 (Mod4Comp)
  - NHR
- Tech support (for helping)
  - D. Rotscher and M. Kräußlein
- You (for listening)