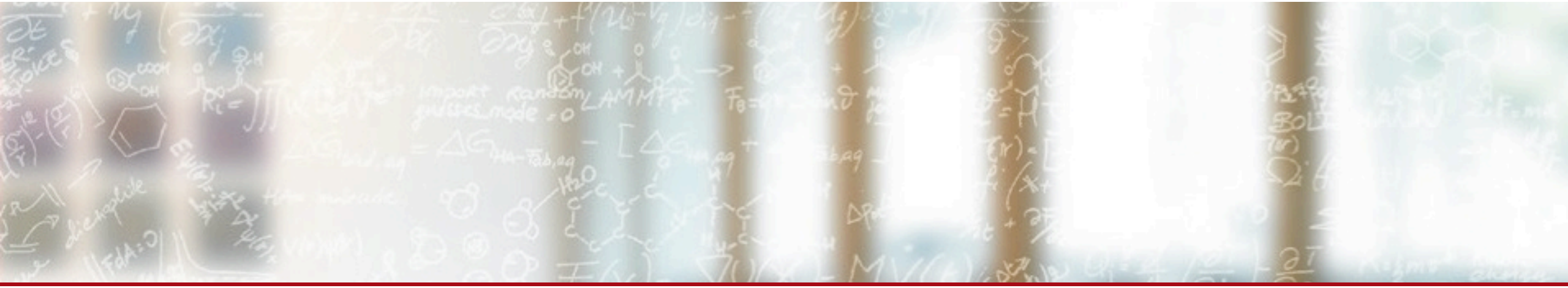
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



# RM-Replay: A High-Fidelity Tuning, Optimization and Exploration Tool for Resource Management

Maxime Martinasso
Swiss National Supercomputing Centre (CSCS)
ADAC 7 2019

# Exploring resource manager parametrization of a production system

- HPC center view
  - Goal is to improve the usage of the resource: higher utilization, higher throughput
  - What-if scenario:
    - Changing RM parameters
    - Introducing new policies
    - Updating RM and using new features

- User point of view
  - How long will my job wait in the queue?
    - If I improve my runtime estimate
    - If I use this specific queue
    - If I use a special constraint

- Current approaches
  - Identify a better configuration? How do you test it?
    - Apply a change between two maintenances, monitor
  - Using a simulator?
    - Complexity of a production system: partitions, priorities, node states, reservation, …
    - Accuracy of the simulation
    - Translate simulation output into decision making

# New approach: RM-Replay

- Use the exact same software stack for the RM
  - No modification of the code (or very minimal)
- Use fewer resources than the production system
  - Less physical resources: nodes
  - Less time: faster-than-real-time clock
- “Replay” the exact set of actions for a workload:
  - Job submissions
  - Node availability
  - Reservations
- Allow historical studies, gather statistics on events (end of allocation period, GB)
- Use a well-known interface (the RM itself) by the system engineers

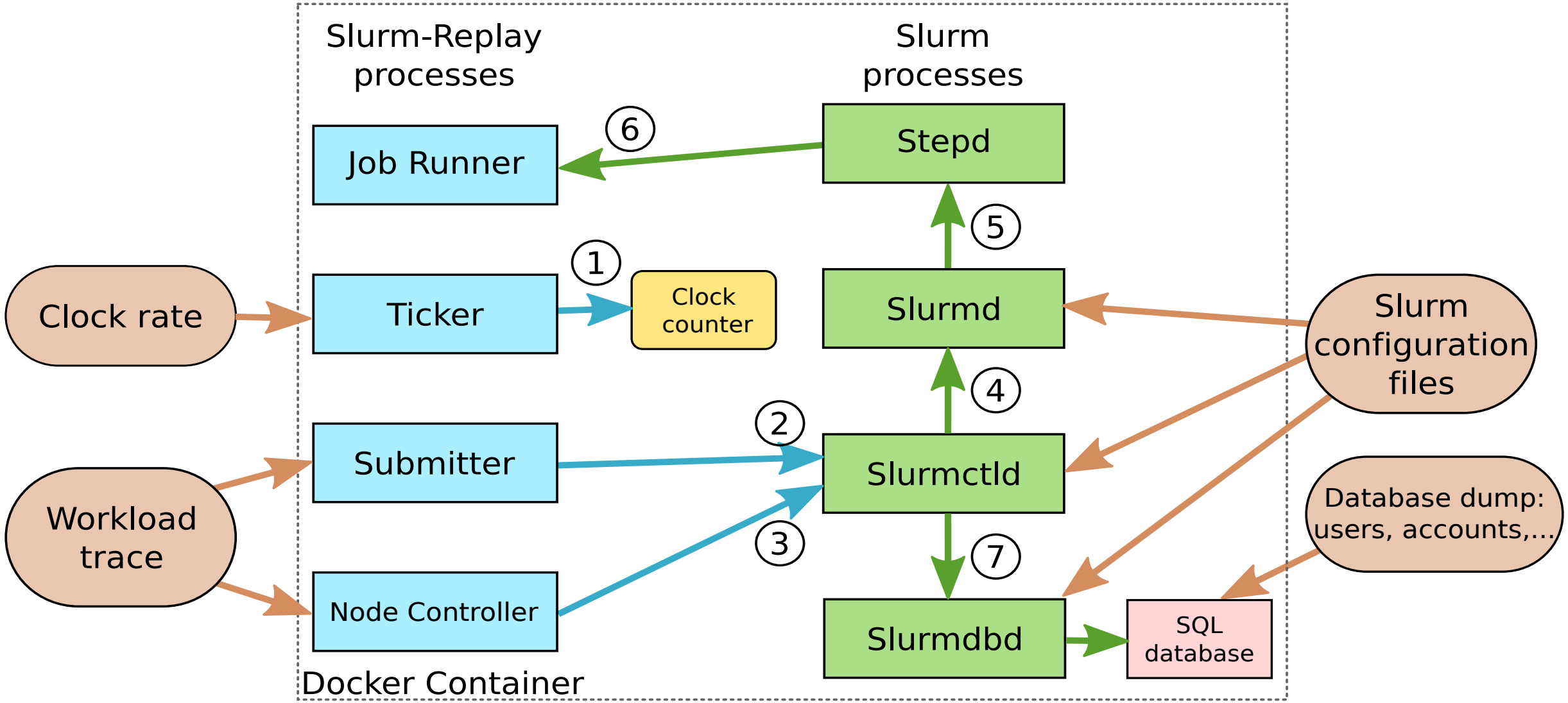
# How does RM-Replay work?

- Use a container to recreate the entire resource manager software stack in an isolated environment
- Inside the same container recreate the set of users (who submit jobs)
- Create an adjustable clock which will replace the system clock and will be used by the software stack
- Develop a set of programs to recreate the interaction originating from a workload
- Provide configuration data outside the container to enable portability to other HPC systems

# Instance of RM-Replay: Slurm-Replay

- Slurm is a resource manager used on many large HPC systems
  - 6 of top 10 in the top500 (November 2017)
  - Many features: High scalability and performance, fair-share, reservation, plugins, …
- Slurm is a complex software:
  - Accurate simulation is very difficult
  - Integrated simulator modifies code base and the scheduling behavior
    - Event based simulation, capturing all events? Impact on the scheduling?
    - Lack of portability
- Characteristics of Slurm-Replay:
  - Use the original and not modified Slurm software stack
  - Use configuration parameters from a large HPC production system (Piz Daint)
  - Replay production workloads
  - Evaluate scheduling metrics: throughput, utilization, waiting time, …

# How does Slurm-Replay work?

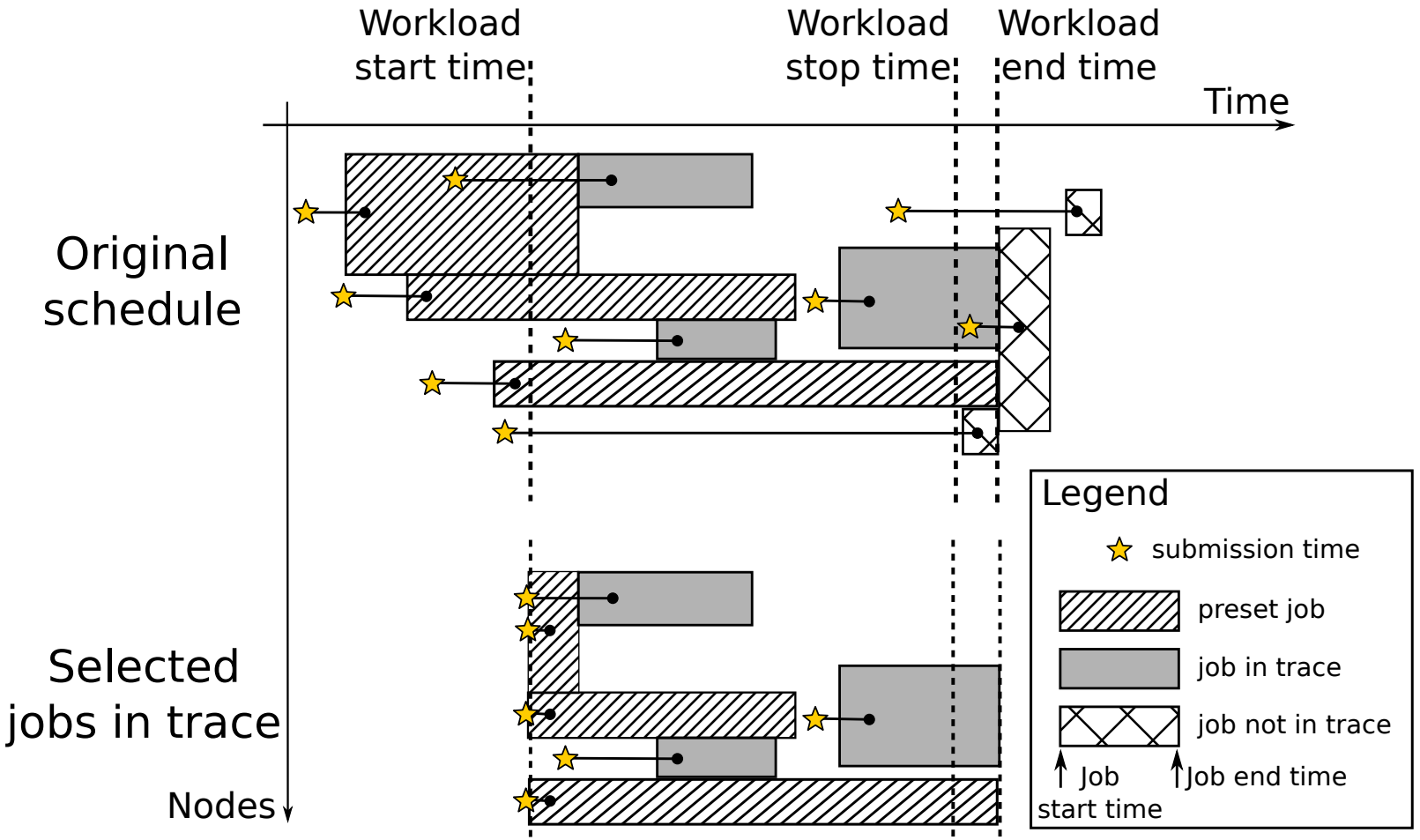


# Workload:

- Jobs
- Job dependencies
- Node states
- Reservations

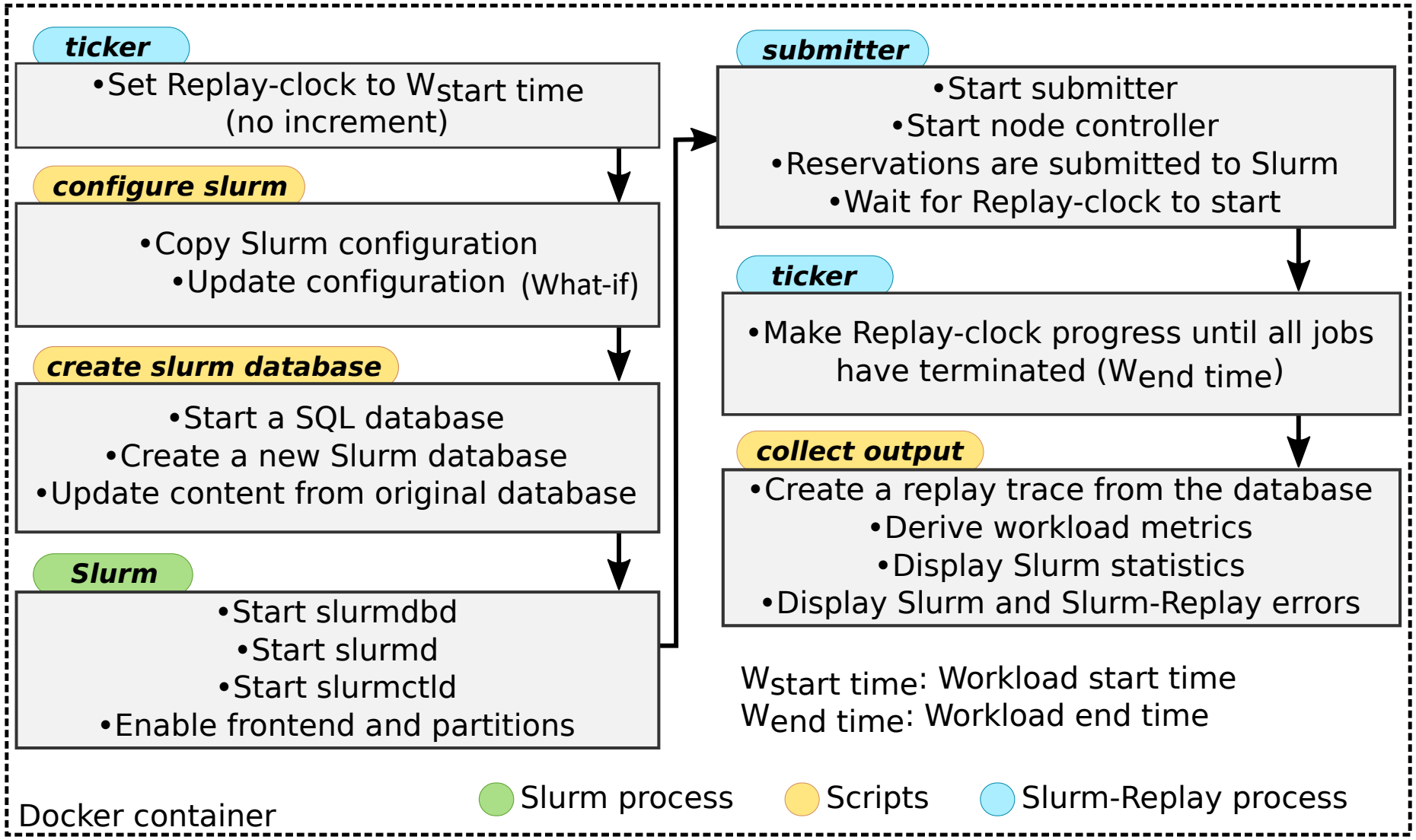
# Multitenant:

- Extract users/groups




*Elements in a trace:*
**Job**={id, submission time, start time, end time, number of nodes, time limit, partition, dependency, priority, qos, reservation name, user, account, state, exit code, eligible time, list of nodes}
**Reservation**={id, name, account, start time, end time, list of nodes}
**Node state**={start time, end time, node name, state, reason}

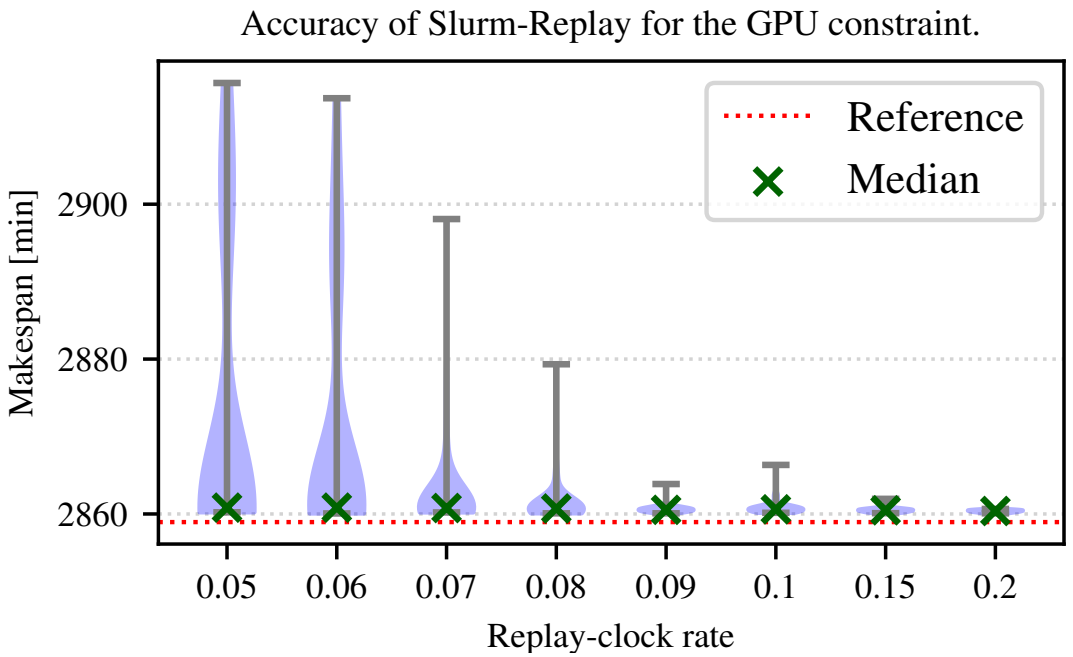
# Workflow

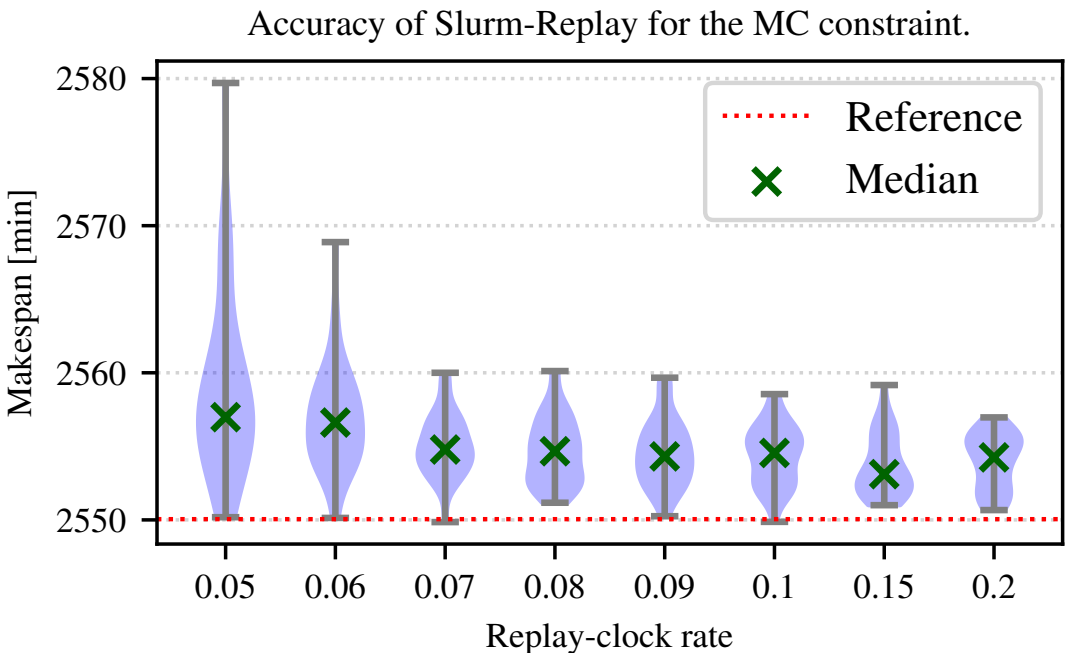

ticker
•Set Replay-clock to Wstart time (no increment)
configure slurm
•Copy Slurm configuration
•Update configuration (What-if)
create slurm database
•Start a SQL database
•Create a new Slurm database
•Update content from original database
Slurm
•Start slurmdbd
•Start slurmd
•Start slurmctld
•Enable frontend and partitions
submitter
•Start submitter
•Start node controller
•Reservations are submitted to Slurm
•Wait for Replay-clock to start
ticker
•Make Replay-clock progress until all jobs have terminated (Wend time)
collect output
•Create a replay trace from the database
•Derive workload metrics
•Display Slurm statistics
•Display Slurm and Slurm-Replay errors
Wstart time: Workload start time
Wend time: Workload end time
Slurm process
Scripts
Slurm-Replay process
Docker container

# Technical solutions and limitations

- No root privileges inside the container (to use HPC container runtimes)
- Wrap functions used to impersonate users to always return true:
  - *setuid*, *setgid*, *chown*
- Wrap common C time functions to use a faster clock:
  - *sleep*, *gettimeofday*, …, clock counter is in */dev/shm*
- Create and bind mount */etc/passwd* and */etc/group* from users in the workload
- Use Slurm *Frontend* feature to execute only one *slurmd* daemon for an arbitrary number of nodes
- Missing data from the Slurm database taken from system logs
  - job dependency, topology, reservation submission time

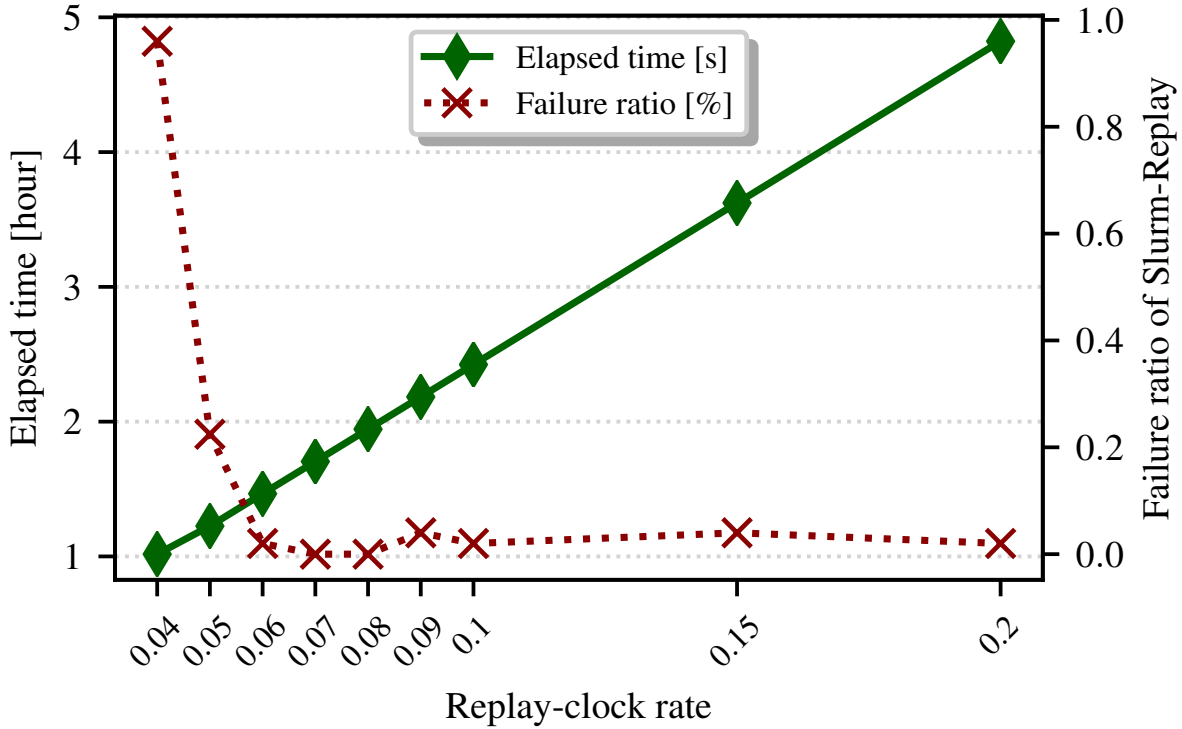
# Accuracy: makespan

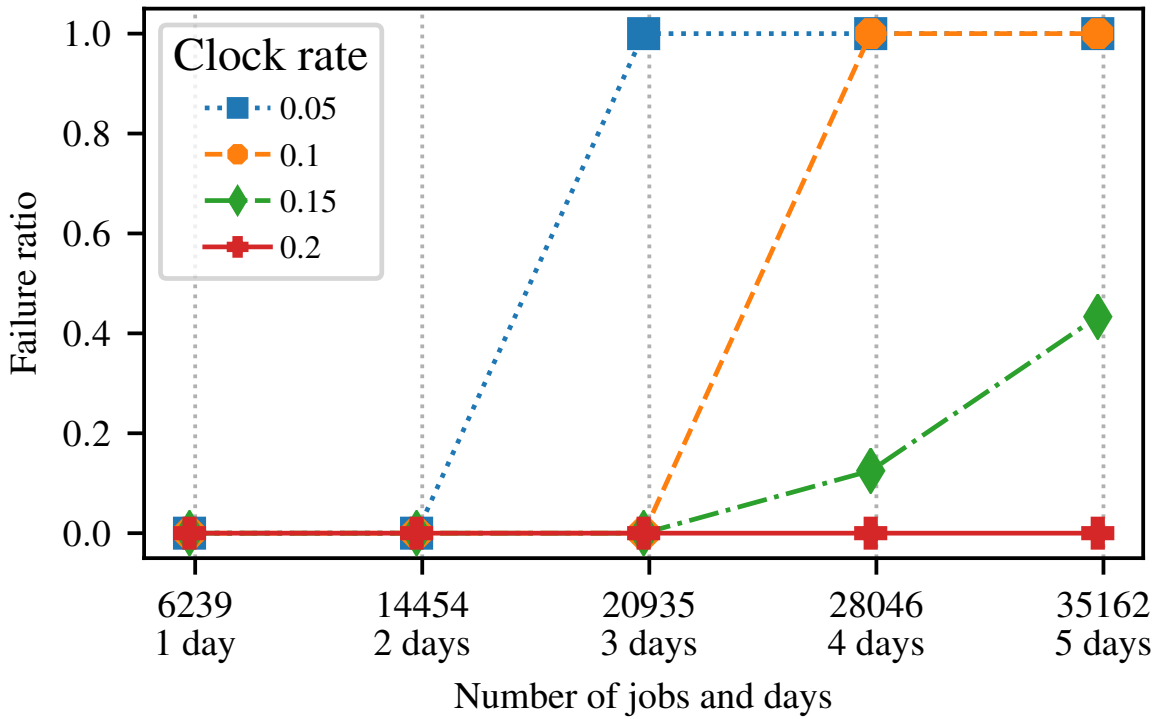



- Data distribution over 50 tests
- 24 hours, 6025 jobs: 2664 jobs GPU constraint, 2409 jobs MC constraint, 169 jobs in other partitions
- 10 reservations and 51 state changes of nodes
- Peak utilization of 97% of GPU nodes and 54% of MC nodes
- Schedule completed in 47.65 hours for GPU and 42.7 hours for MC

# Performance vs clock rate

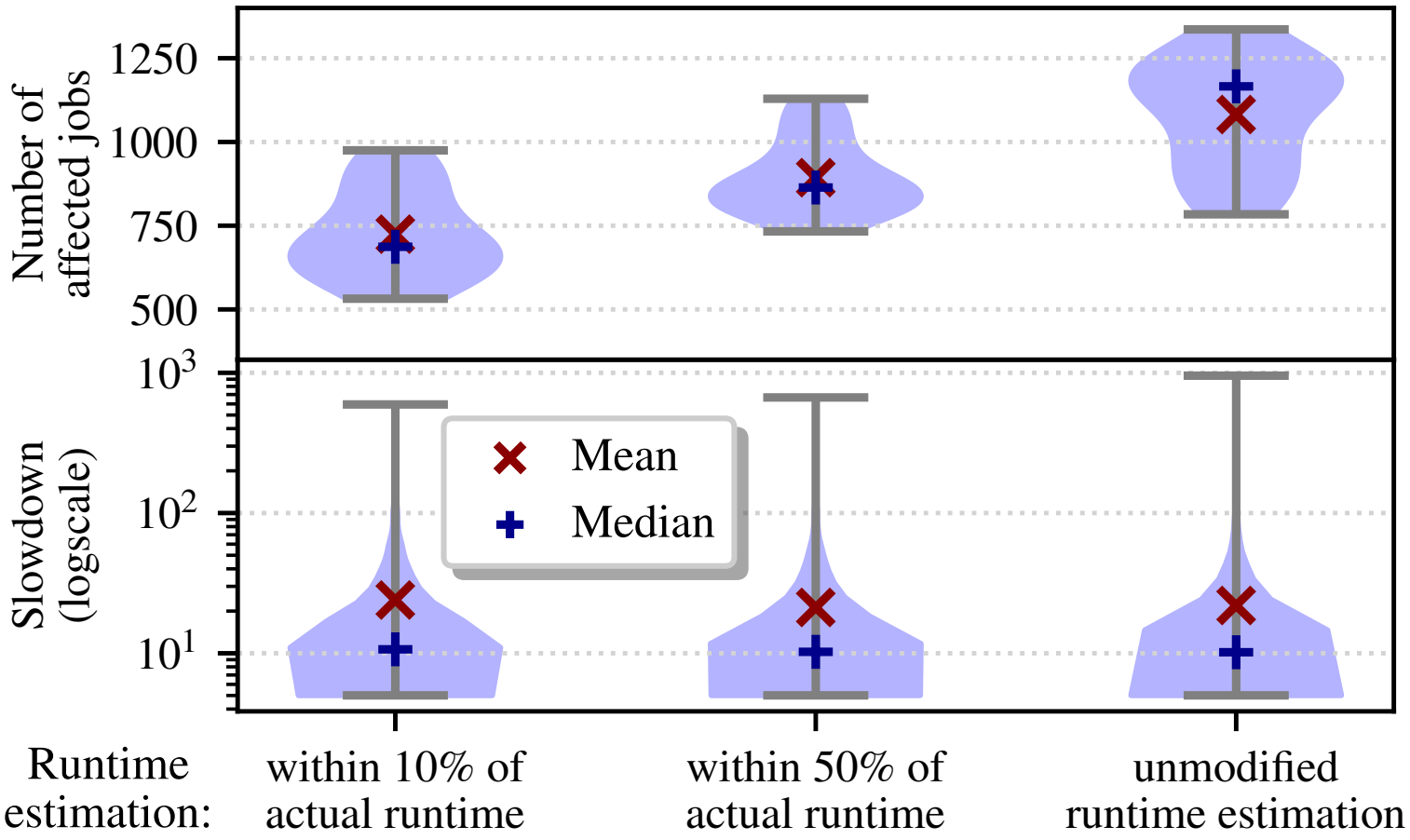



- Failure: Slurm-Replay takes too long to process backlog of jobs
- Performance dependent on the underlying CPU frequency
- A clock rate of 0.06 (16.7x faster) seems to be the best option on a 2.1 GHz CPU
- Use only one slurmd daemon

# Example of use case

User: “Will my job be scheduled earlier if I provide a better runtime estimation?”




Slowdown means: how much a user is willing to wait:

$$\frac{T_{wait} + T_{exec}}{T_{exec}}$$

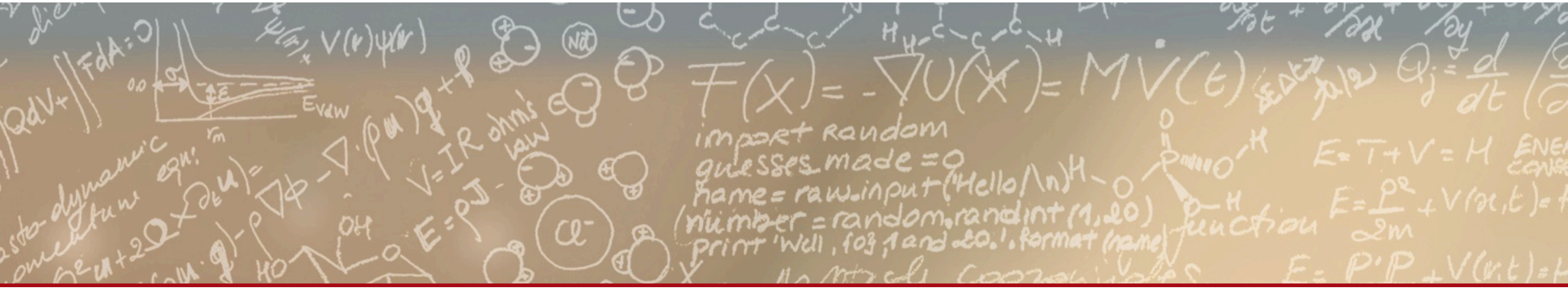
There are fewer jobs waiting but no improvement in the waiting time.

# Conclusion and future work

- RM-Replay: A High-Fidelity Tuning, Optimization and Exploration Tool for Resource Management
- Useful for testing what-if scenario: parameter, version, feature, policy, …
- Help to take better decision on a production system configuration
- Slurm-Replay an instance of RM-Replay for Slurm

https://github.com/eth-cscs/slurm-replay

- Use Slurm-Replay at CSCS
- Investigate slurmd performance




# Thank you for your attention.