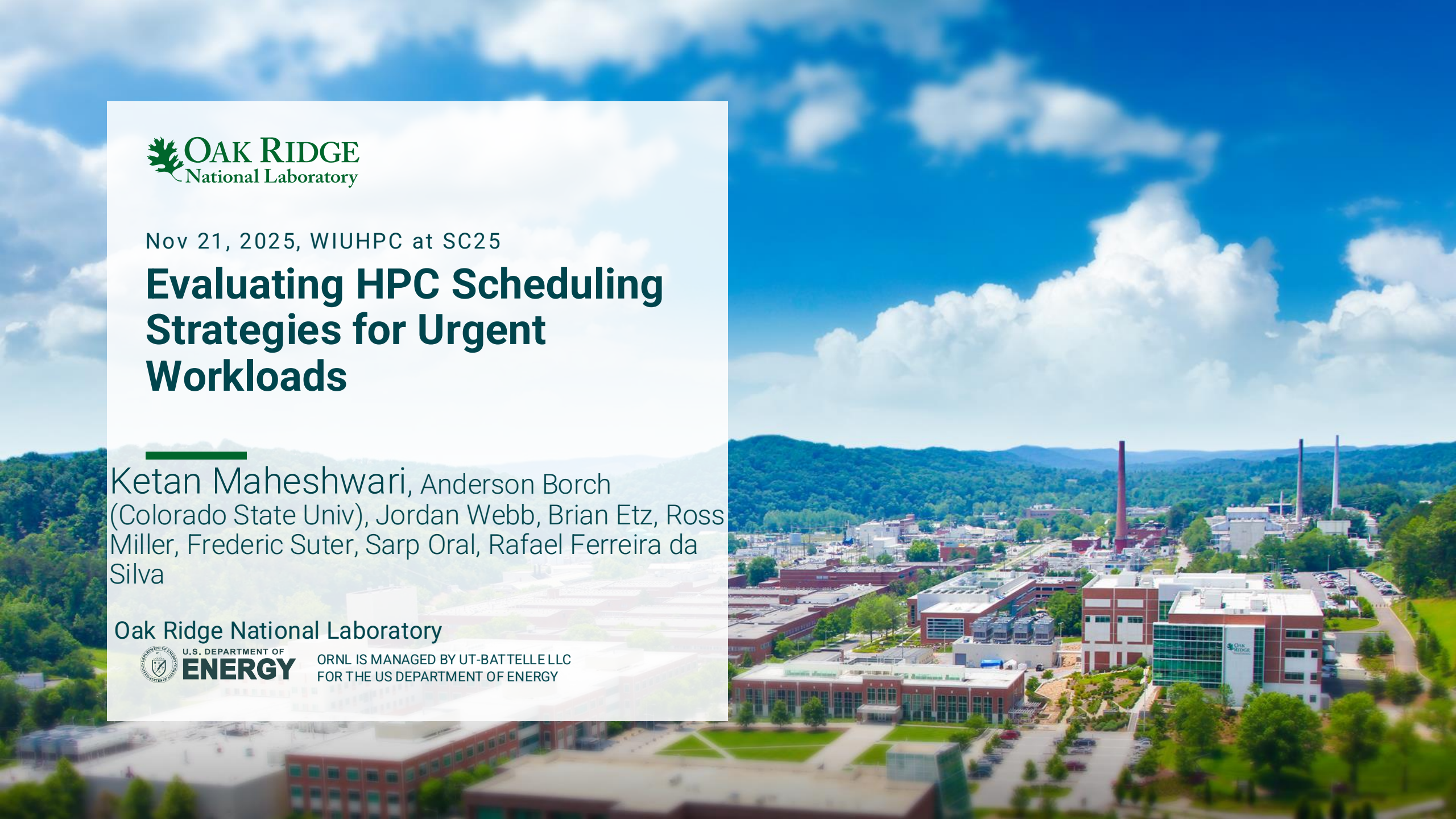


OAK RIDGE National Laboratory

Nov 21, 2025, WIUHPC at SC25

Evaluating HPC Scheduling Strategies for Urgent Workloads

Ketan Maheshwari, Anderson Borch (Colorado State Univ), Jordan Webb, Brian Etz, Ross Miller, Frederic Suter, Sarp Oral, Rafael Ferreira da Silva

Oak Ridge National Laboratory

U.S. DEPARTMENT OF ENERGY

ORNL IS MANAGED BY UT-BATTELLE LLC
FOR THE US DEPARTMENT OF ENERGY

Overview

A study and analysis of **responsiveness**, **(un)fairness**, **user-experience** and **system utilization** in an HPC job scheduling environment where a fraction of jobs have requirements for **urgent computing**.

Results quantify user and system performance outcomes, reusable code artefacts and generalized visualization methods from standard Slurmdb data.

Code and other artefacts at: github.com/ketancmaheshwari/wiuhpc25

Context

DOE IRI vision to accommodate applications with diverse urgency requirements within a unified environment.

An attempt to analyze scenarios where a fraction of jobs must be allocated urgently.

Contributions

- A comparative evaluation of four representative job scheduling simulators
- Design and implementation of an emulation-based methodology that operates on a real cluster
- Visual analytics, including per-user timelines, scheduling and cluster utilization trends
- A recommended mapping between workload profiles and recommended scheduler configs

Evaluation of Job Scheduling Simulators

- We begin with looking into the existing job scheduling simulators
- Selected 4 representative simulators: Batsim, Accasim, Alea and The Slurm Scheduler

P-F Dutot, M Mercier et al 2016. Batsim: a Realistic Language-Independent Resources and Jobs Management Systems Simulator. doi:10.1007/978-3-319-61756-5_10

C Galleguillos, Z Kiziltan et al 2020. AccaSim: A Customizable Workload Management Simulator for Job Dispatching Research in HPC Systems. doi:10.1007/s10586-019-02905-5

D Klusacek, M Soysal, and F Suter. 2019. Alea - Complex Job Scheduling Simulator. doi:10.1007/978-3-030-432225_19

N A Simakov, R L DeLeon et al 2018. Slurm Simulator: Improving Slurm Scheduler Performance on Large HPC systems by Utilization of Multiple Controllers and Node Sharing. doi:10.1145/3219104.3219111

Job Scheduling Simulators: Batsim and Accasim

- **Batsim** is a C++ based job simulator
- Supports the Standard Workload Format (SWF) and JSON data
- Includes built-in tools to generate workloads and define simulated platforms
- We ran Batsim with a custom workload that parameterizes number of jobs, submission intervals, wall-time ranges and failure rates.
- However, we could not induce urgency QoS levels, priority boosts and preemption mechanisms.

- **Accasim** is Python based
- Supports only SWF data
- Offers limited built-in scheduling algorithms compared to Batsim
- We were able to run Accasim using Batsim generared loads.
- Faced some difficulties in interpreting results due to limited labeling of outputs.
- No native support for QoS levels or preemption mechanisms.

[See our paper for feature comparison in more details]

Job Scheduling Simulators: Alea and the Slurm scheduler

- **Alea**: Java-based, provides a few scheduling algorithms and user-defined scheduling policy; a form of QoS is supported via multiple queues however preemption is not supported.
- Alea turned out to be close to our requirements -- we engaged with its main developer and are in the process of adding features resulting in a next gen version ”AleaNG”.

- **The Slurm scheduler**: Aims to reproduce actual Slurm scheduling behavior in a controlled environment mimicking real environment as close as possible.
- Designed to run inside a Docker container. Posed challenges in deployment in lab managed environments. Current release runs an older Slurm (v20.11) compared to OLCF systems (v25.05)

Feature gaps in simulators wrt our requirements

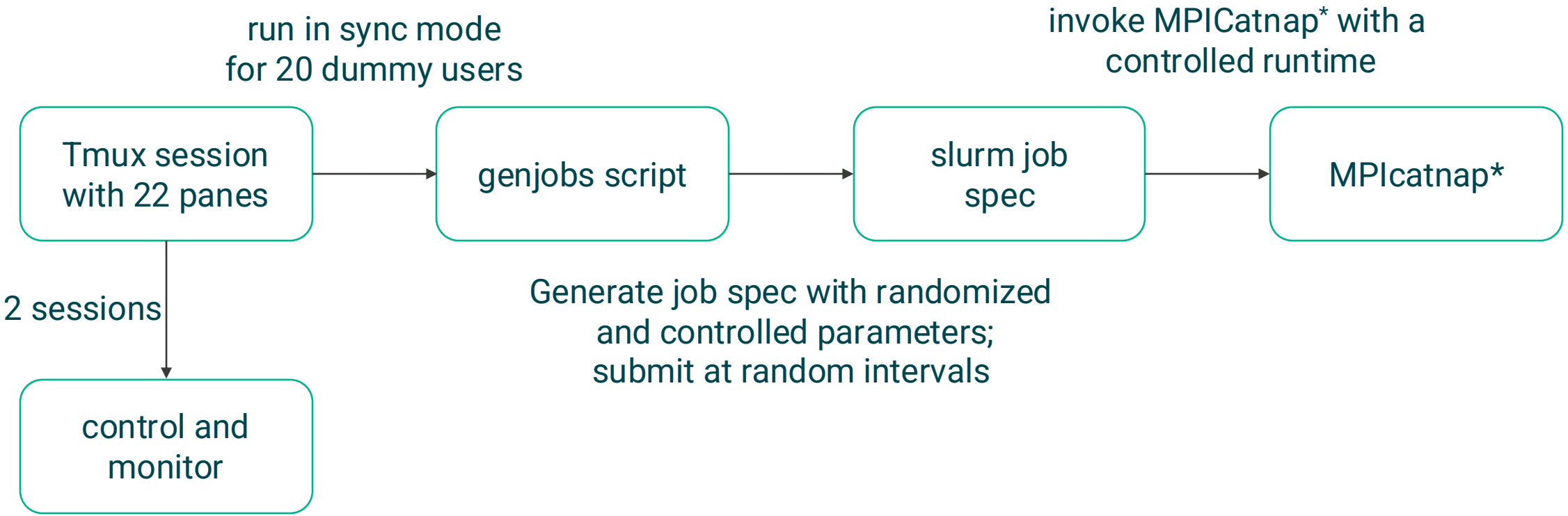
- No or limited native support for urgent QoS or urgency aware scheduling
- Limited or no implementation of job preemption with realistic handling of cancellation or requeuing of preempted jobs
- Limited modeling of dynamic user behavior (cancellations, timeouts) and event-driven job arrivals
- Practical deployment barriers in our testbed environment

- Nevertheless, job scheduling simulators are very valuable tools

A Rare Opportunity:
The OLCF Advanced Compute Ecosystem (ACE) Testbed

- A promise of “Emulating” Job Scheduling over a real Cluster
- 20 Intel nodes with 64 CPUs (2 logical cores/CPU) and 8 Nvidia H200 GPU nodes
- Slurm version 25.05
- 20 dummy user accounts were created: common group memberships and privileges

Instrumentation: Experimental Setup



* code courtesy: MJ Wilde, ANL/UChicago

Features of the Experiment setup

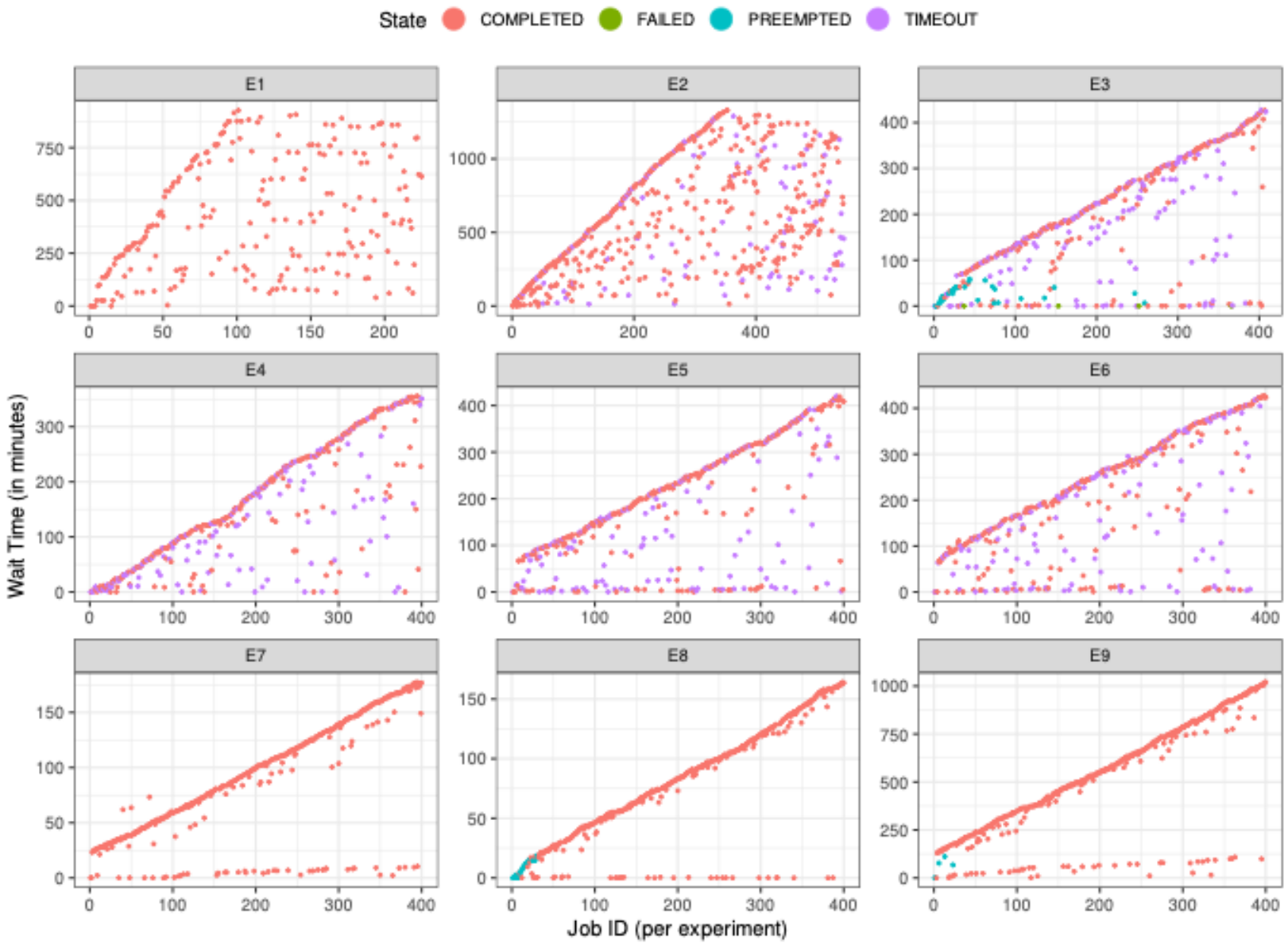
- Generate and submit jobs at randomized intervals
- Randomized job parameters within defined limits
    - walltimes, node size,
    - Urgent jobs marked via “urgent” QoS in Slurm
- Customizable application runtimes
    - wrt to the job walltimes
- The mock mpicatnap application:
    - cats an input file to an output file (minimal I/O)
    - runs sleep process for defined time on all cpu cores
- Monitoring enabled but may run unsupervised in detached tmux sessions

Experiments Design: Nine experiments with varying conditions

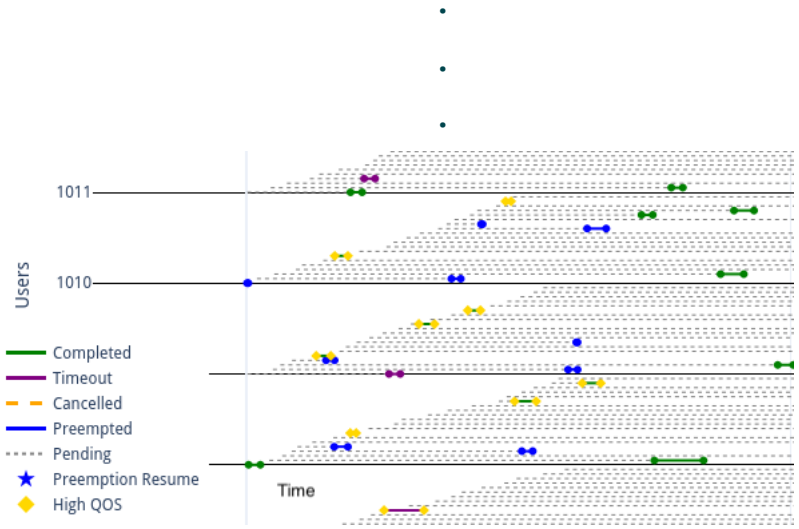
- E1, E2 and E4: Calibrations on short and long, fixed and varying number of jobs, timeouts, backfill verifications
- E3: 20 jobs/user, 5-10% urgent jobs, preempt and cancel
- E5: Same as E3, except no preemption and priority boost for urgent QoS jobs
- E6: Same as E3 except preempt and requeue
- E7: Short 10-min app runtime that is 10-20% of job walltimes, preempt and requeue
- E8: Same as E7 except preempt and cancel
- E9: Same as E7 except app runtime is 80-90% of walltimes, preempt and cancel

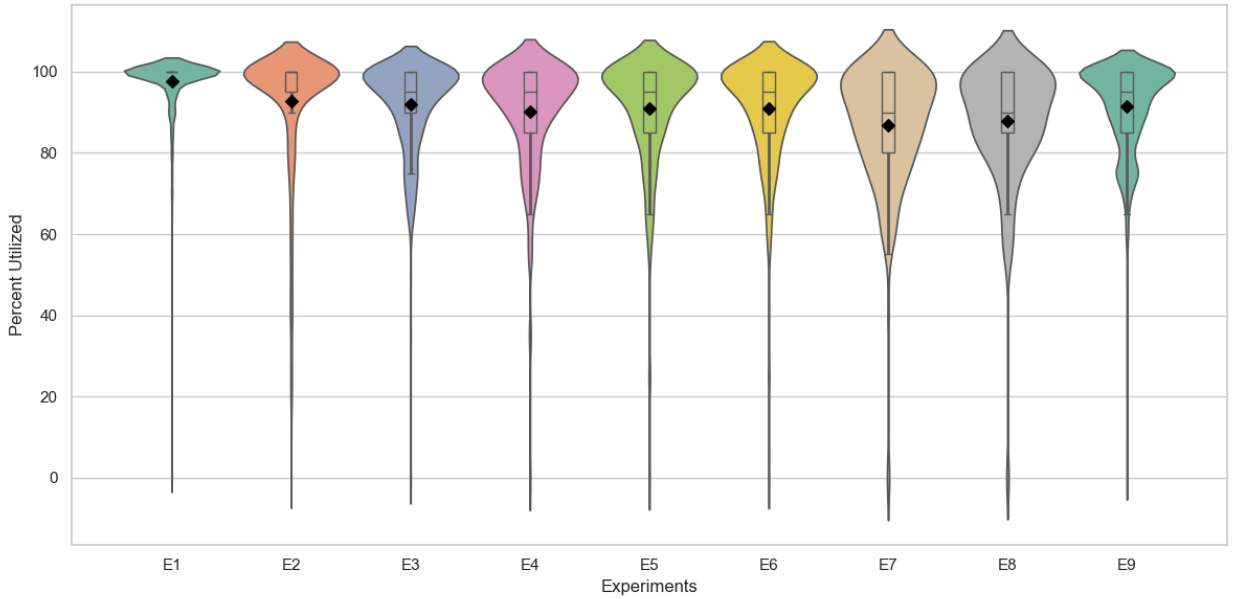
Visual overview of all experiment runs

- Plots show jobs' end state vs wait time in queue for each job across all experiments
- Note the differences in the Y-axes values that indicate the disparities in the wait times resulting from each experiment setup



Results Visualization: User view of jobs and cluster utilization





"Swimlanes" for user job states and events
Gives a vizualization of simultaneous events for users

Cluster utilization via **sinfo (probed once per min)**
violin width = distribution at that percent utilization
Diamond = mean, box = median and quartile

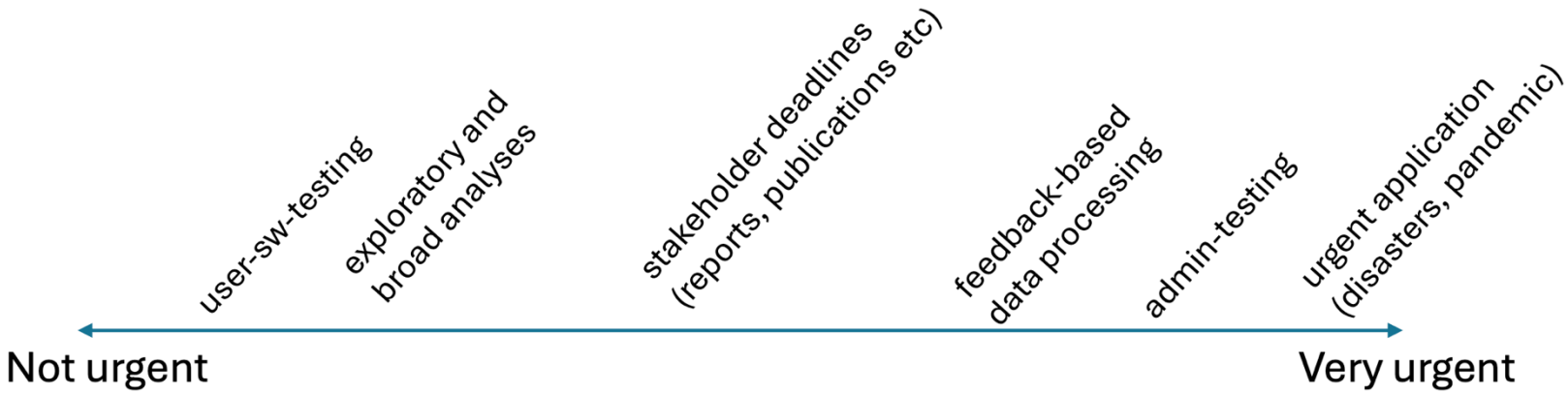
Statistical analysis for various outcomes on each of the experiments

- Tradeoffs between system utilization and user outcomes (both normal and urgent) under various Experimental conditions!
- Wasted node minutes ”undetermined” for cases where jobs are preempted and requeued since Slurm do not keep a record of preemption time
- No single ”ideal” setup

Exp Name	Percent System Utilization (σ)	Normal QoS Wait Time	High QoS Wait Time	Wasted Node Minutes
E1	97.7 (6.9)	430.7	–	–
E2	92.8 (15.8)	597.0	–	–
E3	92.0 (11.0)	198.2	1.5	257.7
E4	90.3 (13.5)	152.6	–	–
E5	91.0 (13.5)	206.8	4.5	–
E6	90.9 (13.0)	220.9	5.6	undetermined
E7	86.8 (15.0)	97.2	5.0	undetermined
E8	87.8 (14.6)	83.6	0.2	66.1
E9	91.4 (10.8)	559.4	48.9	78.1

Experiments Conclusion: Recommendations for HPC Scheduling environments based on Outcomes

Job Profile	Recommended Configuration	Rationale
Urgent, short runtime (< 15 min)	High QoS with preemption	Maximizes responsiveness with minimal wasted compute time.
Urgent, long runtime (> 1 hr)	High QoS priority boost without preemption	Reduces wait time while avoiding costly interruptions.
Normal priority, exploratory	Normal QoS with possible preemption and requeue	Balances availability for urgent jobs with continued progress for non-critical work.
Normal priority, production-scale	Normal QoS without preemption	Protects long-running, resource-intensive jobs from disruption.
Feedback-driven workloads	High QoS priority boost without preemption	Enables timely updates while preserving stability.
Testing and validation runs	Normal QoS with preemption allowed	Facilitates rapid turnover to make room for critical work.



Conclusion and Future Work

- An emulator for job scheduling emulating tunable, real-world job scenarios
- A reproducible, reusable set of configurations, code artefacts, and results viz routines
- Identifying kinds of job on an urgency-continuum is a challenge
- Featureful simulators are still valuable
- Future: Alea -> AleaNG addressing the limitations of simulators and enabling many more experimental scenarios rapidly

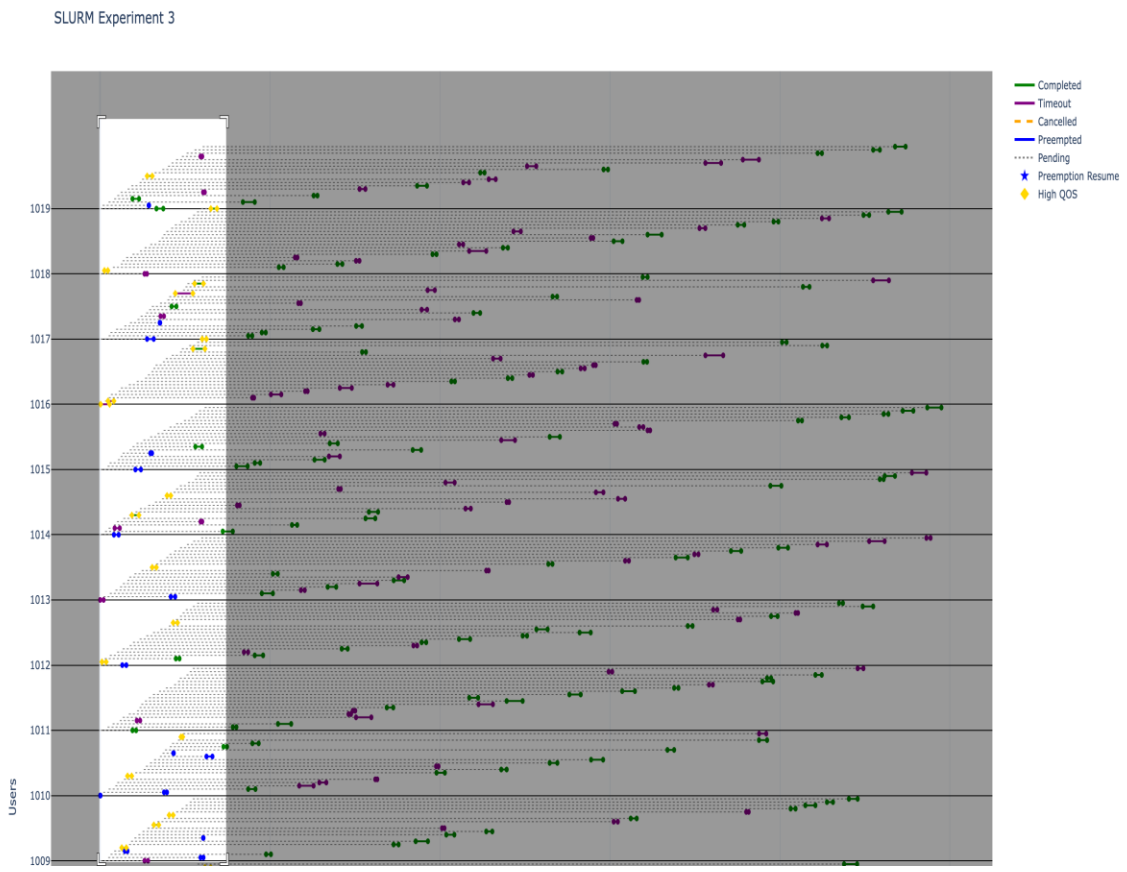
Acknowledgements

This research used resources of the OLCF at ORNL, which is supported by DOE's Office of Science under Contract No. DE-AC05-00OR22725.

Thank You for your time! Questions?

km0@ornl.gov | https://github.com/ketancmaheshwari/wiuhpc25

Extra Slide: Dynamic and interactive plots that may be selectively visualized by timeslice or by user(s)



Overall View, regions may be selected for enlargement

Enlarged view

Extra Slide: Violin plot for system utilization without the extrapolation