HPC Digital Twins for Evaluating Scheduling Policies, Incentive Structures and their Impact on Power and Cooling

Matthias Maiterth*
Wesley H. Brewer
Terry Jones
Feiyi Wang
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

Arunavo Dey
Jaya S. Kuruvella
Tanzima Z. Islam
Texas State University
San Marcos, Texas, USA

Kevin Menear
Dmitry Duplyakin
National Renewable Energy Laboratory
Golden, Colorado, USA

Rashadul Kabir
Colorado State University
Fort Collins, Colorado, USA

Tapasya Patki
Lawrence Livermore National Laboratory
Livermore, California, USA

Abstract

Schedulers are critical for optimal resource utilization in high-performance computing. Traditional methods to evaluate schedulers are limited to post-deployment analysis, or simulators, which do not model associated infrastructure. In this work, we present the first-of-its-kind integration of scheduling and digital twins in HPC. This enables what-if studies to understand the impact of parameter configurations and scheduling decisions on the physical assets, even before deployment, or regarching changes not easily realizable in production. We (1) provide the first digital twin framework extended with scheduling capabilities, (2) integrate various top-tier HPC systems given their publicly available datasets, (3) implement extensions to integrate external scheduling simulators. Finally, we show how to (4) implement and evaluate incentive structures, as-well-as (5) evaluate machine learning based scheduling, in such novel digital-twin based meta-framework to prototype scheduling. Our work enables what-if scenarios of HPC systems to evaluate sustainability, and the impact on the simulated system.

CCS Concepts

• **Computer systems organization**; • **General and reference** → *Cross-computing tools and techniques*; • **Computing methodologies** → **Discrete-event simulation**; **Distributed simulation**; *Simulation evaluation*;

Keywords

Scheduling Simulators, Digital Twin, Data Center Digital Twin, System Simulator, Distributed Systems Simulation, Batch Scheduling

*Corresponding author: *maiterthm@ornl.gov*

SC25-W, November 16–21, 2025, St. Louis, MO

ACM ISBN 978-1-4503-XXXX-X/2025/11…$15.00
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Matthias Maiterth, Wesley H. Brewer, Terry Jones, Feiyi Wang, Arunavo Dey, Jaya S. Kuruvella, Tanzima Z. Islam, Kevin Menear, Dmitry Duplyakin, Rashadul Kabir, and Tapasya Patki. 2025. HPC Digital Twins for Evaluating Scheduling Policies, Incentive Structures and their Impact on Power and Cooling. In *Proceedings of The International Conference for High Performance Computing, Networking, Storage, and Analysis – Workshops (SC25-W).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The increasing complexity of highly-efficient supercomputing centers fuels an ever increasing demand for more powerful models. Digital twins (DTs) have emerged as a means of integrating system telemetry, modeling and simulation, artificial intelligence (AI), and system control mechanisms to create a virtual representation of the physical system, modelling cooling, power, and workloads [6].

Aiming for optimal usage of high-performance computing (HPC) systems, different stakeholders face various challenges. For example: *users* seek feedback regarding job usage, estimated runtime, and application efficiency; *operators* monitor and tune operational parameters based on load and conditions; *center managers and vendors* seek insight into which machine aspects are the biggest barriers to performance, and seek trends for future procurements. Data center digital twins (DCDTs) can provide estimates and simulations and even serve for design considerations and virtual prototyping of future systems [6], without consuming the system's own resources.

Scheduling is critical for efficient use of HPC [3, 29]. Therefore, integrating scheduling into DTs of HPCs is necessary to form a representative twin of the overall system. The sound integration of scheduling capabilities into a DCDT extends its capability from a reactive role to a predictive one, enabling *what-if* studies. A representative yet modifiable scheduling simulator integrated into a DCDT allows to study how a system responds to alteration of its parameters. Production systems are not suitable for such changes, unless service interruptions are acceptable. Similarly, scheduling simulators in isolation or DCDT's without these capability can not answer such questions. Integrating a scheduler into a DCDT is therefore a valuable contribution, revealing holistic insights into the operation of our systems and optimization opportunities.

In this paper, we introduce Scheduled-RAPS (S-RAPS) [10], which extends the resource allocator and power simulator (RAPS), originally developed for the open-source digital twin framework ExaDigiT by Brewer et al. [9]. We present what-if studies evaluating the integration of schedulers with digital twins. Specifically, we evaluate:

(1) **Scheduling impact on system response** – We explore the impact of scheduling to understand a system's power, cooling, and workload response with our integration, a factor not observable for scheduling simulators in isolation.
(2) **Evaluation of incentive structures** – We explore incentive structures and impact of imposed reward metrics on workloads and system — a case-study impossible with the current state-of-the-art, without deploying to production.
(3) **Machine learning (ML) for scheduling** – We study ML guided scheduling based on metrics usually not accessible to scheduling simulators in isolation, or hardly trainable due to limited data and context without an integrated DCDT.
(4) **Integration with external schedulers** – We demonstrate the feasibility of our approach by extending scheduling-integrated DCDT to interface with external schedulers.

This work extends to the original work of [9] and use open datasets for validation and use-cases. Our contributions are:

- **Integration of scheduling into DCDTs**: Previous work does not integrate scheduling simulators with digital twins. Our research is the first to enable such integration, allowing the exploration of "what-if" scenarios to study power and cooling, given real workload and system data.
- **Use of open datasets**: Previous work has utilized open datasets for post-mortem analysis; however, this work is the first to utilize such datasets to simulate the anticipated system behavior given altered scheduling parameters.
- **Extension to other schedulers**: As each system and scheduler setup is unique, we provide the extensions necessary such that users can model and simulate their system and their use-cases, providing wide applicability beyond what is presented in this work.

The remainder of the paper is structured as follows: Section 2 presents background on digital twins and scheduling, and discusses the open data sets used in this research. In Section 3, we introduce S-RAPS. In Section 4, we present our evaluation and use-cases. Section 5 discusses future work, and we conclude in Section 6.

2 Background

2.1 Related Work

2.1.1 Digital Twins for Operational Optimization: DT research has surged in recent years, and useage of traditional modeling and simulation techniques have found wider adoption for operation [39], using data-driven approaches [32] and by coupling AI with online decision making [40].

In the context of HPC, the work on ExaDigiT [9] is seminal as an open-source framework, consisting of the RAPS module, a transient thermo-fluid cooling module, and an visual analytics model of the supercomputer and central energy plant. The work, however, only presented an initial framework with large emphasis on the thermo-fluid cooling simulator, as well as power loss modeling for job-trace replay, without venturing into the explorative aspects. The initial work does not include a batch scheduler and only *replays* the given workloads, therefore unable to *re-schedule* for what-if analysis.

In turn, our work builds upon the ExaDigiT framework, with explicit focus on scheduling, introducing S-RAPS. We extend ExaDigiT to support open data sets for additional systems, and explore use-case driven analysis for HPC, and evaluate the impact of various scheduling policies on power and cooling.

2.1.2 Scheduling Simulators: Simulating scheduler behavior has been an active area of research, consistently supporting advances in HPC [7]. Popular examples of batch scheduling simulators are Slurm Simulator [33–35], and scheduler specific alternatives such as the work by Wilkinson et al. [**?**]. CQSim [30], which originated in QSim is a prominent example outside of the Slurm ecosystem. This is a non-exhaustive list, as simulators such as GridSim [11], SimGrid [12], Bricks [38], Simbatch [20], Alea [24], AccaSim [19], BBSched [17], ScSF [31], Batsim [15], as well as schedulers that have built in simulators, such as Torque/Maui [23] and Moab Scheduler, played an important role in the development of scheduling simulators and should not be left unmentioned.

These simulators generally are not focused on the systems infrastructure but the core of scheduling. With continuous progress in scheduling simulators in general, the aim is to enable the integration of such advanced scheduler developments into DTs. We designed S-RAPS to leverage existing work such that users can interface with other scheduling simulators, such as the Slurm Simulator or FastSim. This enables easy extensions to S-RAPS for studying power and cooling. We demonstrate an integration of FastSim [41] into S-RAPS in Sect. 4.2.2.

2.2 Open Datasets

In this work, we selected four open datasets for accessibility and reproducibility, as shown in Table 1: PM100 [5] from Marconi100, F-Data [4] from Fugaku, LAST [26] from Lassen, and Cirou's dataset [14] from Adastra. Additionally, we use a proprietary dataset from Frontier due to its extensive verification and validation in the context of DCDTs from previous work [9] for reproducibility.

The telemetry traces used for simulation vary according to the data source, which we discuss as follows: • **Marconi100:** The Marconi100 system at CINECA has two public datasets: the M100 [8] and the PM100 dataset [5]. We use the PM100 data as it is precurated. We filter jobs containing shared nodes as this is not yet supported in our model. The data includes CPU, memory and node power in a 20-second interval. As the data has been filtered, it does not reflect the system's full operational utilization. This means that replay and reschedule will differ [5].
• **Fugaku F-Data**: F-Data [4] is a dataset containing job and performance information with derived metrics for job classification from Fugaku. It includes monthly data from March 2021 to April 2024, with the following job metrics: energy consumed, node power (minimum, maximum and average), performance characteristics on operations, memory activity and the resulting performance class identified as either *compute-* or *memory-bound*.
• **Lassen LAST**: This is a 1.4-million-job dataset from the Lassen supercomputer [26]. It includes information on job allocation, node allocation, and job-step disposition. These are combined to get

Table 1: Systems and datasets used in study.

System	Architecture	Nodes	Dataset	Scheduler	Job Count	Characteristics
Frontier	HPE/Cray EX	9600	Proprietary	Slurm	1,238	job traces (15s), CPU/GPU power & temp.
Marconi100	IBM POWER9	980	PM100 [5]	Slurm	231,238	job traces (20s), CPU/node power
Fugaku	Fujitsu AF64FX	158,976	F-Data [4]	Fujitsu TCS	116,977	job summary, node-level power only
Lassen	IBM POWER9	792	LAST [26, 27]	LSF	1,467,746	job summary, includes network tx/rx
Adastra	HPE/Cray EX	356	Cirou [14]	Slurm	30,570	job summary, job avg component power

usable information for each job allocated with accumulated energy data. Lassen uses the IBM's LSF Job scheduler [28].

• **Adastra:** CINES has published 15 days of the Adastra system [14], including node power, memory power, and CPU power. The heterogeneous system has a CPU and GPU partitions. GPU power is not provided, but can be derived from node power and the other components. It uses Slurm [2], but no scheduling policy is stated.

• **Frontier:** The Frontier dataset was initially used to develop the ExaDigiT DCDT. The dataset is an excerpt from the center's continuous collection, obtained from both Slurm data, as well as Cray EX Telemetry API, collected in the STREAM system [1]. The scheduling policy is a priority-based mechanism that uses a modified first-in, first-out (FIFO) queue, boosted based on node count and penalized on allocation overuse [16]. This dataset is the only one not publicly available. Since it was used for initial verification of RAPS[9], it was important for cross validation of S-RAPS.

3 Method & Design

In the following, we show the current state-of-the-art, the ExaDigiT framework [9], with its resource allocator and power simulator (RAPS), for context, and present our Scheduled-RAPS (S-RAPS) extension. We begin by discussing the mechanisms of the existing simulation loop of the forward-time DCDT simulator by Brewer et al. [9]. We then present S-RAPS, with its built-in scheduler and how this ties into the DCDT simulators. We then show how S-RAPS extends to external forward-time or event-based simulators. Finally, we show how extensions for dataloaders allow us to load and simulate diverse datasets and systems, showing the true value of an open-source digital twin framework.

3.1 Prior state of ExaDigiT

The original design of ExaDigiT consists of three main modules [9]:

(1) Modelica-based cooling model
(2) Resource allocator and power simulator (RAPS)
(3) Visual analytics model

ExaDigiT's simulation is depicted in Fig. 1. The digital twin reads a sequence of job traces or telemetry and placed on the system simulator as recorded. For each timestamp, the observed node utilization is replayed. The simulated utilization is converted to a power profile, with power rectification and conversion losses applied [42]. The power and therefore generated heat is fed into a cooling model [22, 25]. The cooling model simulates from cooling distribution unit (CDU) to cooling towers, giving an accurate representation of the system at each timestep.

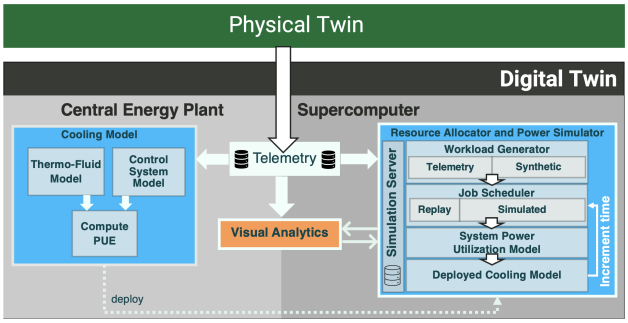


Figure 1: Simplified OriginalExaDigiT overview in accordance with Brewer et al. [9], with RAPS module on the right.

In the simulation, the RAPS module provides the inputs to the cooling model and is also the main driver of the simulation loop. This is outlined in Algorithm 1 of [9]:

(1) Initialization of system and data, and start of simulation.
(2) Simulation loop:
  (a) Addition of newly arriving jobs to the job queue
  (b) ScheduleJobs: selection and placement of available jobs to available resources.
  (c) Tick: management of resources, and calculation of compute resource utilization, power and cooling

The original work processes jobs in a scheduler class, which contains the DT's replay mechanism. As shown in Fig. 1, this only considers replay of recorded *telemetry* or *synthetic* data, not scheduling. For a generic HPC digital twin, the ability to alter the scheduling *policy* and resulting job placement is key. For this, we describe the refactoring necessary to enable generic built-in scheduling and allow for the integration of external schedulers with S-RAPS, enabling the scheduling scenarios we present in this paper.

3.2 S-RAPS: Scheduled - Resource Allocator and Power Simulator

We now present the improved simulation loop for comprehensive integration of schedulers within ExaDigiT's RAPS, named S-RAPS. Figure 2 shows the overhauled design of S-RAPS, enabling the integration of forward-time or event-based schedulers. Key changes include refactoring and generalizing of the following five components: (1) system initialization, (2) dataloaders, (3) simulation engine, (4) scheduler abstraction, (5) systems accounting. This is

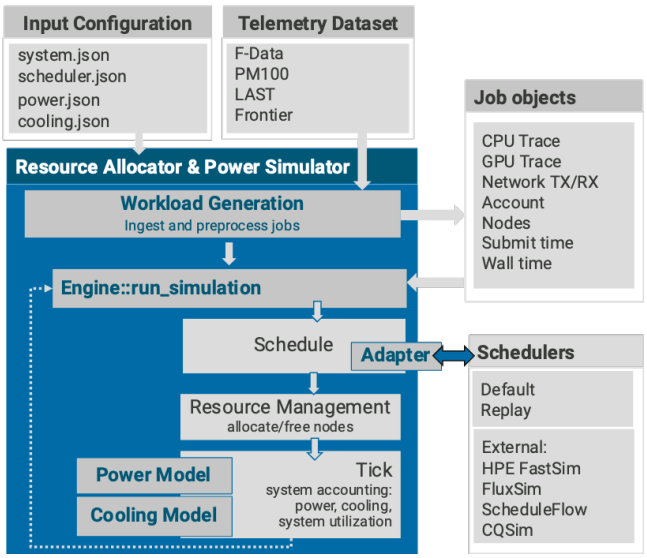


Figure 2: Scheduled-RAPS (S-RAPS): Integration of scheduling into the design of ExaDigiT's resource allocator and power simulator (RAPS). With improved configuration mechanisms, pluggable dataloaders, interface to build-in and externals schedulers, and overhauled simulation loop.

accomplished while keeping the original RAPS simulation concept with intended scheduling, resource management, and `tick` intact.

3.2.1 System initialization: The system initialization has been improved to capture not only the system's configuration, but also to include information necessary for scheduler simulation. The rework introduces cleaner abstractions and separation of concerns, with future extensions in mind. The core loop of the simulator was refactored introducing a simulation engine. During system initialization, telemetry is used to initialize the job objects augmented with information for scheduling. The telemetry is now also used to initialize user-account information (clear or anonymized). The refactored simulation engine separates resource manager and scheduler interface, which loads either the build-in or external scheduler. Finally, objects for tracking statistics of the simulation are initialized.

The HPC System configurations, their dataloaders, and the schedulers are implemented as plugins. The specific configurations are selectable on simulation start via the command line interface (CLI) and are designed to be easily extensible. This helps with rapid testing of configuration and experimental design: administrators can easily represent their systems, and developers of scheduling simulators can easily load their policies, once extended.

3.2.2 Dataloaders: A dataloader's task is to load and parse the telemetry data and generate the list of to-be-scheduled jobs. Each job requires information on: *submit time*, *start time*, *end time*, *time limit*, and the *number of requested nodes* (alternatively, the exact set of nodes to which the job was assigned). This is a standard for scheduling simulators as for example used in the standard workload format (SWF) [13]. The dataloaders also load the job traces for replay in the DCDT simulation. If traces are not available for a dataset,

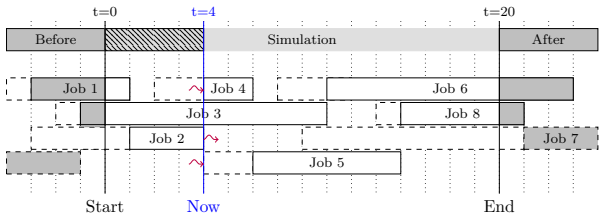


Figure 3: Example job trace, with job-submit time, -start time and -end time. The time-stepped simulator triggers on each time step, while the event based scheduling simulator only has to react to triggered events (magenta arrows) such as start of a job (job 4), end of a job (job 2), and submission of a new job (job 5).

scalar values can for example represent a job's average power, energy, or other characteristics – depending on data availability.

When rescheduling, the job traces recorded may not coincide with the time slice needed for the simulation. We treat such occurrence as missing data, using the last known value. Therefore, for correct simulation the dataloader must identify the following key times for each job:

- Job submit, start and end time
- Telemetry start and end time

Given the individual job times and the overall timespan of the dataset, we can run the simulation within the range of the overall telemetry. Additionally, this change enables us to either replay the recorded data as is or simulate a new schedule.

An example of such timeline is presented in Fig. 3, showing a simulation from $t = 0$ to $t = 20$, with submission times of jobs indicated by dashed outline extending in front of the jobs, and actual execution indicated by the solid outline of the jobs. During rescheduling, the jobs can be placed as early as they have been submitted, which is ultimately decided by the selected scheduling policy. Jobs that ended before start of the simulation time or were submitted after end of the simulation time are dismissed. The simulation of the power and cooling behavior can then be simulated as implemented in `tick` of S-RAPS, with the modified timeline.[1]

3.2.3 Simulation engine: The simulation engine contains the main simulation loop. At start of the simulation, the user selects explicit simulation start and end time. Given this information, the system state is prepared, and in case the dataset contains jobs before the selected simulation start, these jobs are placed to prepopulate the system. This allows us to represent the actual system condition as observed in the telemetry at start of the simulation[2,3]. We can

[1]There are two edge cases to consider: jobs which originally started before the capture time (see Fig. 3, Job 1) and jobs which ended after the capture time (see Fig. 3, Jobs 6, 7, 8). When simulating a new schedule, these jobs may therefore have no corresponding telemetry at the associated simulation times. Therefore, when rescheduling these jobs within the simulation time, these cases need to be flagged as they can cause potential discrepancies in other simulations as no known ground truth is available to S-RAPS.
[2]This is often neglected by scheduling simulators, which ignore jobs before simulation start, and therefore need time to fill up the queue and system, distorting the results.
[3]This also allows S-RAPS to simulate anticipated schedules and profiles from live-data.

then enter the main simulation loop. It is refactored into four well-defined steps:

(1) Preparation of the time step: Before each iteration, the system state is updated. For this, completed jobs are cleared from the system, freeing resources, and updating the state of nodes.
(2) Addition of eligible jobs to the job queue: Jobs that have been submitted — according to simulation time — are added to the job queue.
(3) Call of `schedule`: The jobs are scheduled according to the job queue and selected scheduling policy, and placed on the system in coordination with the resource manager.
(4) Call of `tick`: The engine's `tick` function calls the sequence of DCDT simulators and models, and increments the time.

These descriptions seem simple, but required major changes from the simple replay mechanism of the original design. For example, in the original replay mechanism, node placement was not enforced. In the overhauled design, the exact node placement as specified in the telemetry is used in replay mode. However, when rescheduling the scheduler select the appropriate nodes. The resource manager then completes the job placement, allocating nodes. Additionally, the refactor resolved timing and allocation issues for nodes with both ending and starting jobs coinciding in the same time step.

Regarding job eligibility, in the original replay mode, all jobs were part of the job queue. Jobs were then placed on the system as soon as indicated by their start time. With the updated design, jobs can only be scheduled and placed once they have been submitted. This also means that pre-computing a schedule is not possible as the digital twin observes the jobs as they are submitted, just like a real system. The scheduler is not aware of jobs not yet in the queue.

When calling the `schedule` function the loaded implementation is triggered. This recomputes the order of the job queue according to selected policy and coordinates with the resource manager to place eligible jobs. The split of resource manager and scheduler via this interface was a major improvement, separating the built-in scheduling capabilities and enabling the use of external schedulers[4].

The update to `tick` now also represents a clear separation of concerns enabling easier replacement of simulation sub-modules, where tick is only responsible for the simulation of physical subsystems. Our redesign of S-RAPS with the simulation engine puts strong focus on extensibility and the use of plug-ins. This also enables support for future site-specific customizations.

3.2.4 Scheduler abstraction: As outlined, the simulation engine triggers the scheduler in each iteration of the simulation loop. Any external scheduler and scheduling simulator has its own set of logics regarding which events to track and react to. The abstraction we use enables users who interface S-RAPS with their scheduling simulator to implement the logic for triggering and sending these events. Figure 3 illustrates such case, with the triggered evens for time step $t = 4$ shown as magenta arrows.

S-RAPS interfaces with the scheduler in case the simulator provides new information in a given iteration: (1) by triggering the scheduler to recompute the schedule, or deciding to skip if no change has occurred; (2) it interprets the information returned from the scheduler; and (3) S-RAPS then triggers the resource manager, placing identified jobs on the system, and maintains the job queue. This ensures that the remainder of the simulation can progress.

3.2.5 Built-in and external schedulers: The scheduler can be selected via the `--scheduler` CLI option. Its policies are selected via the `--policy` and `--backfill` options. The default is the build-in scheduler which implements the policies: first-come, first-served (FCFS), shortest-job-first (SJF), largest-job-first (LJF), and priority-based scheduling. Additionally, the default scheduler also provides the replay mechanism of the original RAPS implementation. Regarding backill options the supported defaults are *no-backfill*, *first-fit*, and *easy* (i.e. Earliest Available Start-time Yielding (EASY)[36]). All options are extensible for use with external schedulers.

While the default scheduler provides basic scheduling policies, it does not provide implementations for best-fit, greedy, conservative, or other more sophisticated implementations. For this, we provide the interface for external schedulers and scheduling simulators. As shown in Sect. 2.1.2, the numerous schedulers and scheduling simulators all have their validity, and we do not compete but try to enable them, and provide example integration in the source-code.

3.2.6 Systems accounting: The final major rework to discuss is the addition of system accounting and statistics. The original RAPS design kept track of general simulation and HPC system statistics, with focus on the power and cooling simulation of the DCDT. S-RAPS extends those and adds collection of statistics for jobs, users, accounts, as well as scheduler-focused statistics. This allows users to easily extend metrics of interest for their facility or experiments.

Previously tracked information includes: completed jobs, job throughput, average system power, power loss, system power efficiency, total energy consumed, and the cost estimates for carbon emissions. S-RAPS adds more scheduler-specific information and also aggregates according to user accounts, such as (non-exhaustive): queued and running jobs average job size, histogram of job size scheduled (small, medium, large, by node count), aggregate node hours, average power and energy per job, their energy-delay-product (EDP), energy-delay^2-product (ED^{2}P), average CPU and GPU utilization, wait time, turnaround time, as well as area weighted response time (the average turnaround time per unit of node-hour across all scheduled jobs), and priority-weighted specific-response time (average sensitivity-adjusted turnaround time per unit of node-hour), which helps to capture packing efficiency and fairness [21].

By tracking this information for both the system and user accounts, we can assess if a setting of the scheduler favors specific jobs or users. The generated statistics can be compared and correlated within a single simulation and across multiple simulations. This allows us to investigate, e.g. how changes of the job-mix are related to job-turnaround time and observed power swings. For the user account metrics, we added the option to store and reload collected user account statistics at the start of a run, supporting aggregation of this information across simulations.

In summary, these changes establish capabilities for extracting broader and deeper insight about jobs, users, and the system, which standalone scheduling simulators cannot provide without integration into a DCDT framework. It is worth emphasizing that the holistic modeling of power, cooling, and job behavior relies on the

[4]The original design included a reschedule functionality, however it simply redistributed the job start times according to a Weibull distribution and was not representative of batch scheduling.

integrated design of the full DCDT (Fig. 1), where interactions between subsystems are critical. Such cross-disciplinary dynamics cannot be replicated by aggregating telemetry data in isolation, even with comprehensive scheduling records.

4 Evaluation and Use-Cases

In the following, we evaluate the S-RAPS scheduler interfaces and extensions, utilizing datasets from a diverse set of HPC systems and external schedulers. In Sec. 4.1 we evaluate the rescheduling mechanism with different policies and datasets. This is followed by use-cases: interfacing and integration with external schedulers, such as ScheduleFlow and FastSim in Sec. 4.2; incentive structures in Section 4.3; and evaluating of ML-guided scheduling in Sec. 4.4.

4.1 Evaluation of the built-in scheduler

We implemented dataloaders for Frontier, Marconi100, Lassen, Fugaku, Adastra, given the datasets of Table 1. Each system provides slighly different information with telemetry for Adastra, Fugaku, and Lassen providing average values for utilization, while Marconi100 and Frontier provide traces for their jobs' resource utilization. A system and its associated dataloader is selected with the `--system` CLI option. We present Figs. 4, 5, and 6, where each plot shows full-system power as calculated by the power model and system utilization according to node occupancy[5]. Figure 6 additionally shows power usage effectiveness (PUE) and the water temperature arriving at the cooling towers, as simulated by the cooling model. We show replay according to the telemetry (*replay*, blue), as well as FCFS scheduling (*fcfs-nobf*, teal), FCFS scheduling with EASY backfill (*fcfs-easy*, orange), and priority scheduling with first-fit backfill (ffbf) (*priority-ffbf*, brown). Priorities are used as provided by the datasets and respective documentation.

Figure 4 shows day 50 of the PM100 dataset, from 17:00 to the next day at 10:00. The *replay* utilization curve (blue) is near 80%, with a filling job queue. The rescheduled runs achieve very high utilization at 100% continued utilization using backfill. In the plot, the system shows higher aggregate power in the non-backfilled approach (teal). The statistics show that average power consumption (−2%) per job and job size (−5%) decreased using either backfilled policy. In combination, the adjusted job placement and start times result in smoothing of the aggregate load, mitigating the power jump at 21:00, observed in the non-backfilled schedule.

Figure 5 shows 15 days of replay and reschedule of the Adastra dataset. As the system utilization is lower and queues not filling up, the choice of scheduling algorithm makes little difference. Noteworthy is that with information on the jobs' power profiles and correct estimates of runtimes, the S-RAPS simulator can match the observed changes in both utilization and simulated power.

Figure 6 shows the same snapshot, as presented in the original paper by Brewer et al. [9], with the cooling model of Kumar et al. [25]. We apply the same scheduling policies as used in the previous cases. The utilization plot shows that the system is making space for three full-system runs, emptying the nodes. Then, the three full-system runs are executed, and afterwards a normal job mix of varied size and lower total power is observed. The different power, PUE[6] and return temperature behavior for the different scheduling policies is clearly visible with regard to how each policy clears the system for the large scale runs. Regarding differences of replay to reschedule, S-RAPS is able to place the large 9216 node jobs earlier (*fcfs-nobf*, *fcfs-easy* and *priority-ffbf*, all overlap and start them at the same time). While freeing nodes for the large

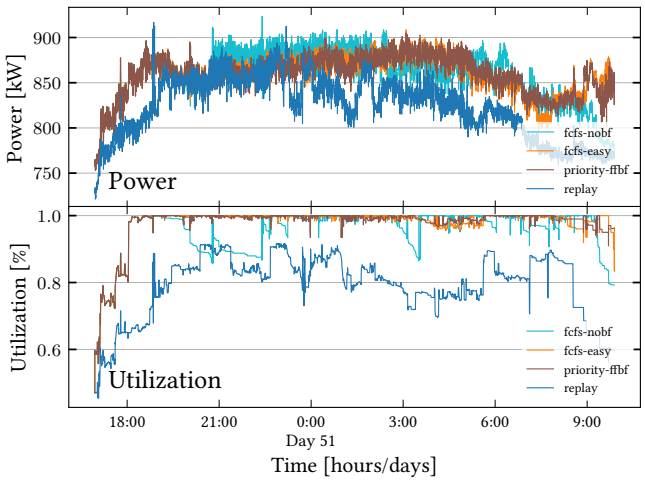


Figure 4: Replay and reschedule of the data from the PM100 Dataset (offset 50 days +17h). Showing FCFS with no backfill (fcfs-nobf), FCFS with EASY backfill (fcfs-easy), priority scheduling with first-fit backfill (priority-ffbf) and replay as jobs were executed, for system power and utilization.

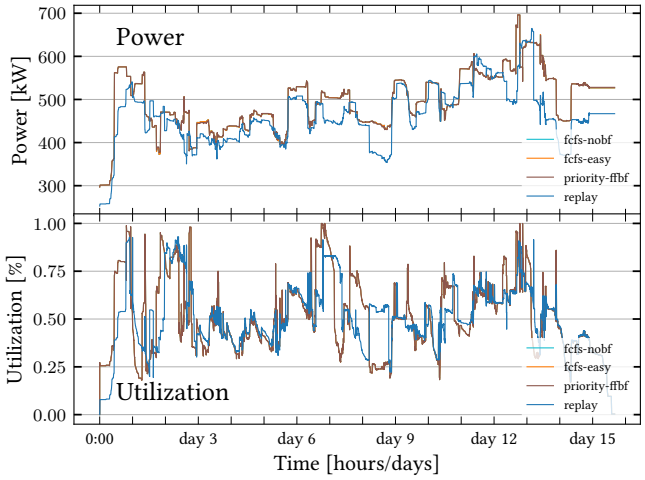


Figure 5: Replay and Reschedule of 15 days of Adastra (full dataset [14]). Replay is shown in blue, while all rescheduled runs (FCFS & priority) overlap almost exactly (brown line). Given known job-power profiles and schedule information, the simulator can predict and match the observed power profile, seen as matching timed up/down-swings.

[5]The datasets do not contain reservations and job dependencies, nor was information about down or drained nodes available. This information could greatly increase the accuracy of schedules, especially when linking the DCDT to a live system.

[6]Power is modeled using [25]. PUE for the actual system is at an average of 1.06%.

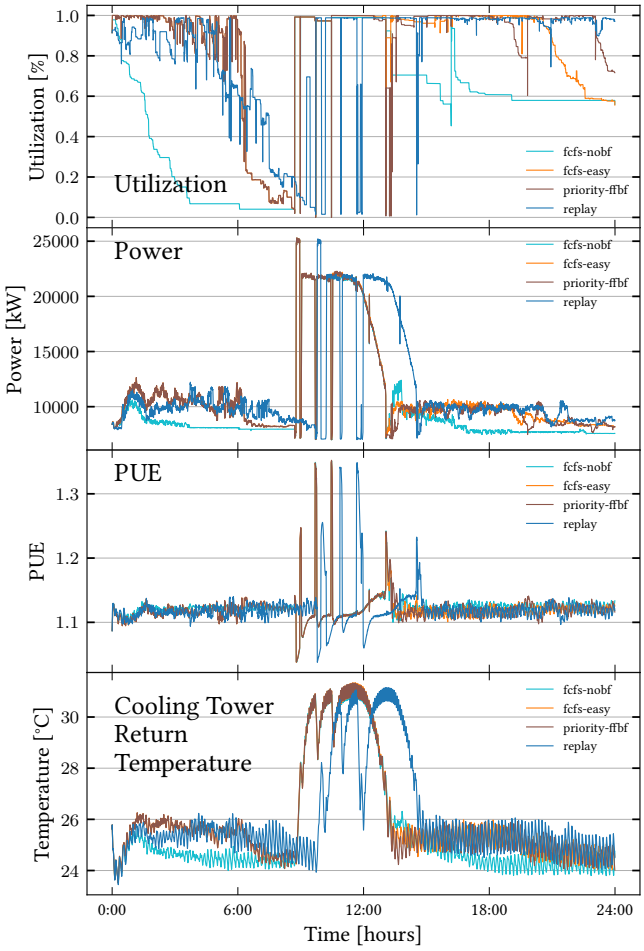


Figure 6: Replay and reschedule of the data used in [9]. Showing FCFS with no backfill (fcfs-nobf), FCFS with EASY backfill (fcfs-easy), priority scheduling with first-fit backfill (priority-ffbf) and replay. The plots show system utilization, system power, as well as PUE[8] and cooling tower return temperature as simulated given the cooling model provided by [25].

runs, the backfilled policies are able to achieve higher utilization compared to *replay*. This, however, is due to the fact that we do not have access to node status, such as information on down or drained nodes. The backfilled policies smooth out the power (and cooling temperature) jump observed in the *fcfs-nobf* case, after the large runs, in similar fashion as described for Fig. 4.

4.2 Evaluation with external schedulers

The integration of external scheduling simulators — ScheduleFlow and FastSim — serve to highlight opportunities for community extensions and future exploration.

4.2.1 ScheduleFlow Scheduler: To prototype the integration of external schedulers, we implemented an interface to ScheduleFlow by Gainaru et al. [18]. The scheduler is event-based and maintains its own internal system state. We implement the interface as described

Figure 7: The results of simulating a synthetic job trace running on Frontier with the FastSim scheduler. The simulated job schedule is passed to ExaDigiT, which can then compute the resource usage over time.

in Sec. 3.2.4 and trigger the necessary internal ScheduleFlow functionality. Hereby, we couple the event-based scheduler of ScheduleFlow with the forward-time simulation of S-RAPS. As ScheduleFlow is not designed for this use-case, nor optimized for performance, this initiates frequent recalculation of the schedule incurring large overheads. The proof of concept was evaluated using synthetic runs, but shows poor performance for any of the real datasets. The main purpose is however achieved: we are able to trigger external schedulers, which allows them to interact with the DCDT simulations made available via S-RAPS. This serves as template for other schedulers, as successfully demonstrated by FastSim.

4.2.2 FastSim Scheduler: FastSim [41] is a lightweight emulation of the Slurm scheduler software. This external tool can simulate cluster behavior up to thousands of times faster than real-time. This integration moves the ExaDigiT DCDT towards the capacity to forecast future events. In Fig. 7, we show a dip followed by a spike in Frontier's power usage on Tuesday morning, as simulated by FastSim using a synthetic job trace developed for Frontier based on the workload statistics in [9]. Accurate forecasting of such events can inform energy-aware scheduling to mitigate the effects of such significant fluctuation in the power draw.

To integrate this simulator with S-RAPS, FastSim was modified by developing a plugin mode option. When operating in this plugin mode, FastSim responds to a request for the system state at a time step specified by the driving simulator. FastSim then processes any events which have occurred up until the requested time step and responds with a list of running jobs indexed by job ID. When triggered by S-RAPS, the returned list is used to allocate resources and subsequently continue in the simulation procedure of S-RAPS. This process requires both S-RAPS and FastSim to maintain separate copies of the system state, which reduces communication between the two simulators at the cost of additional computational overhead. While this process is effective for the real-time simulation necessary for a DT, for the purpose of historical job trace rescheduling, we found it was faster to run FastSim and RAPS sequentially, with FastSim handling the job scheduling and RAPS managing the resources. To generate the results seen in Figure 7, a synthetic job trace of 5,324 jobs run over a period of 15 days was simulated using this sequential approach. The entire simulation time was completed in

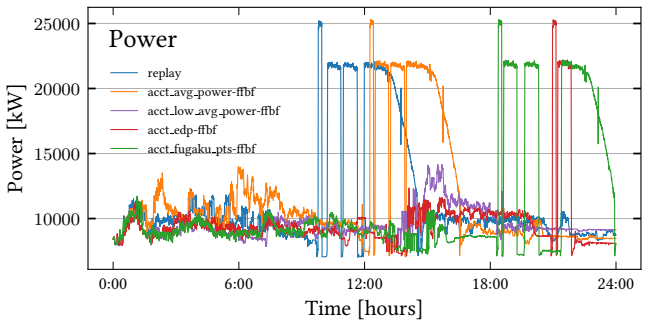


Figure 8: Studying the effects of incentive structures by using account information for prioritization. An Account's priority is based on the accumulated job behavior in the replay case (blue). Reprioritization based on this behavior is shown based on descending average power (orange), ascending average power (purple), EDP (red), and Fugaku points (green).

31 minutes and 24 seconds, amounting to a simulation speedup of 688x compared to real time.

4.3 Use-Case: Incentive Structures

By incorporating information on user accounts, S-RAPS is able to mimic implementations of scheduling incentives and their impact on jobs and system. In the presented use-case, we study the ability to calculate a Fugaku point score according to Solórzano et al. [37]. For any completed job, its statistics are accumulated to the issuing account. This information can be accumulated across multiple simulations or used for prioritization, as described in the redeeming phase of [37]. The implementation is added in *schedulers/experimental.py* of [10], which has policies to derive priorities from account based on: *Fugaku Points*, *power* usage, accumulated EDP, ED^{2}P, and others.

In Fig. 8, we show the resulting power plots when applying different redeeming mechanisms to the same day as in Fig. 6. For the collection phase the replay policy (blue) was used, to illustrate the example. The prioritization for the policies is based on an account's previous behavior on: *average power* (orange, higher is better), *low average power* (purple, lower is better), *EDP* (red), *Fugaku points* (green). Fugaku points reward low average energy consumption in the collection phase. The high power demand of the three large jobs in the collection was not rewarded in the redeeming phase (green), while the generally low power profile was rewarded as intended in [37]. This example illustrates how S-RAPS can be used to run what-if studies that are difficult to realize on production systems.

4.4 Use-Case: Using ML for scheduling decisions

To demonstrate S-RAPS utility, we evaluate a new machine learning (ML) guided scheduling policy prototyped using S-RAPS.

4.4.1 Training Phase: **(1) Clustering.** We partition historical jobs into behavioral clusters using both static (e.g., job size) and dynamic (e.g., power traces) features using K-means clustering. **(2) Classification.** Since dynamic features are unavailable at submission, we train a Random Forest model to learn the relationships between job characteristics (using pre-submission features) and the target metric. This enables real-time mapping during inference time, without requiring telemetry for new jobs. **(3) Prediction.** For each cluster, we train a model to predict target metrics such as runtime, power, memory – based on static inputs.

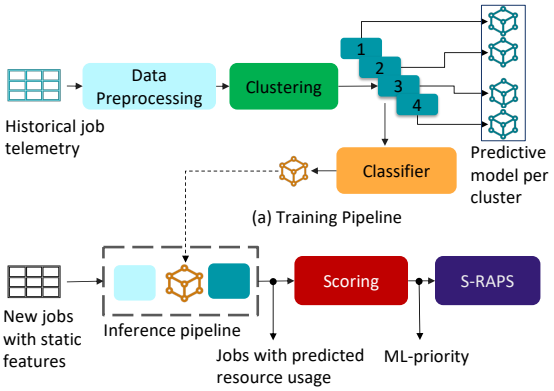


Figure 9: Overview of the ML-guided scheduling pipeline.

4.4.2 Inference Phase: Upon job submission, we normalizes static features, predict the cluster label, invoke the corresponding model, and estimate performance. This design avoids global approximations and ensures predictions are tied to the job's class.

Jobs are ranked via a score computed from predicted metrics and selected static features with the equation:

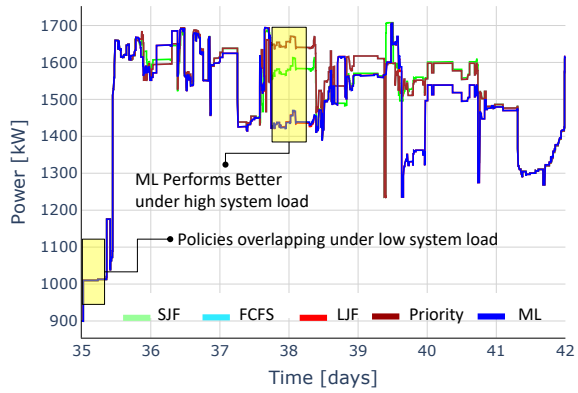
$$S(X_i) = \sum_{j=1}^{K} \alpha_j \cdot \exp\left(\sqrt{X_i^j + 1}^{-1}\right).$$

Where $S(X_i)$ denotes the score of job X_i. X_i^j denotes the j-th feature of the i-th job. α_j is the coefficient of feature j. The exponential function captures fine-grained differences, allowing prioritization based on predicted system-level impact. Unlike single-objective schedulers, this supports trade-offs across throughput, wait time, turnaround, and energy.

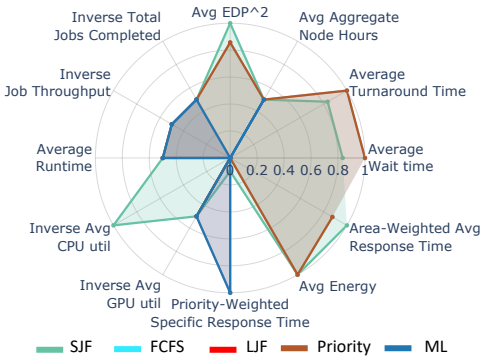
4.4.3 Evaluation: Some datasets (e.g., PM100) report time-series metrics, while others provide scalar summaries. Since timeseries data is inherently noisy and high-dimensional, this causes inaccuracies in the clustering. Hence, we extract summary statistics from timeseries metrics such as maximum, minimum, and standard deviation to retain key behavioral patterns.

In the Fugaku dataset, under low system load (16% requested node utilization), as observed in the left yellow-marked region in Fig. 10(a), all scheduling policies exhibit similar behavior. This happens because with abundance in resources, most jobs are scheduled immediately, resulting in minimal queuing delay and limited influence of scheduling policy. In contrast, under high load, right yellow-marked region in Fig. 10(a), when aggregate job demand exceeds available nodes, the ML-guided policy reduces power spikes per timestep by prioritizing smaller jobs over larger ones.

To evaluate the broader impact of scheduling policies on the overall efficiency of the system, we analyze the policies for a time window with higher resource constraints and variable job sizes.



(a) Power Consumption vs timestep for Fugaku



(b) L2-Normalized multi-objective comparison among policies for Fugaku

Figure 10: Comparison of various scheduling policies. (a) Under high system load, ML-guided policy yields lower power per timestep. and (b) It also improves overall system efficiency across multiple metrics (lower is better).

Figure 10(b) shows consistent trends across datasets: (1) the ML-guided policy achieves the best trade-off across multiple objectives–including lower average wait time, turnaround, and energy consumption, compared to the baseline. (2) Under high load, ML-guided policy consistently yields the lowest job turnaround time and energy-delay product, increasing science per energy spent.

5 Discussion and Future Work

The presented capabilities show a novel way of utilizing operational data — now generally collect during operation — to better understand and predict the behavior of our systems. This work allows studying scenarios and extensions for system- and simulation-needs expanding potential use-cases. We enable integration of arbitrary scheduler and job-trace datasets and allow replay and rescheduling to study their characteristics and exercise what-if scenarios. We are able to seamlessly interact with system specific cooling models which can be easily generated [22], while the power simulation is not a mere aggregation of synchronized trace information, but an accurate computation of component behavior [42]. The work presented allows any user to model their system with the existing tools, and study use-cases based user behavior, job-mix, and the scheduler, extending beyond system, cooling, and location.

The examples show this clearly: the use-cases presented in Sec. 4.1 and Sec 4.2 demonstrate what impact scheduling makes on the power response of a system. Sec. 4.4 shows how S-RAPS allowed to prototype new algorithms, potentially avoiding averse effects.

For future work, identified gaps are the current need of job traces, and power profiles. Figure 5 showed that with perfect information of the job profile, we can accurately predict the systems power swings. However, if this information is not available, we have to rely on user estimates, or fingerprinting and prediction, which are prime candidates for future work.

6 Conclusion

This work provides the first data center digital twin extended for scheduling. We show how we build on a community-driven approach, with the ExaDigiT effort, and integrate scheduling capability into this interacting systems of simulators. We present Scheduled-RAPS (S-RAPS) the scheduler extension of ExaDigiT's RAPS and show how we can study scheduler-induced what-if scenarios and observe the connected digital twin simulations.

This is a first-of-its-kind study, which demonstrates significant improvements over the state of the art, by enabling other schedulers to tie into DCDTs via S-RAPS, as shown for the two external scheduling simulators, ScheduleFlow and FastSim. Finally, we provided case-studies on how the work is used to simulate a scheduling power dip using FastSim, we showed how S-RAPS can be used to study incentive structures for schedulers and showed how ML-guided schedulers can be studied using S-RAPS as interface to digital twins.

Acknowledgments

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. Part of this work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 24-SI-005 (LLNL-CONF-2004842). This material is based upon work supported by the U.S. Department of Energy, Office of Science under Award Number DE-SC0022843 (ECRP). Part of this work was authored in part by the National Renewable Energy Laboratory for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308.

References

[1] Ryan Adamson, Tim Osborne, Corwin Lester, and Rachel Palumbo. 2023. STREAM: A Scalable Federated HPC Telemetry Platform. In *Proceedings of the Cray User Group 2023*. Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States). https://www.osti.gov/biblio/1995656

[2] Naima Alaoui Ismaili, Philippe Wautelet, Juan Escobar Munoz, and Gabriel Hautreux. 2023. Porting and optimizing Meso-NH to AMD MI250X GPUs. In *Proceedings of the SC'23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*. 1900–1905.

[3] William Allcock, Paul Rich, Yuping Fan, and Zhiling Lan. 2017. Experience and Practice of Batch Scheduling on Leadership Supercomputers at Argonne. In *21st Workshop on Job Scheduling Strategies for Parallel Processing held in conjunction with IPDPS 2017 (Orlando, FL, US, 06/02/2017 - 06/02/2017)*. Springer, Orlando, Florida, 1 – 24,. https://doi.org/10.1007/978-3-319-77398-8_1

[4] Francesco Antici, Andrea Bartolini, Jens Domke, Zeynep Kiziltan, and Keiji Yamamoto. 2024. F-DATA: A Fugaku Workload Dataset for Job-centric Predictive Modelling in HPC Systems. https://doi.org/10.5281/zenodo.11467483

[5] Francesco Antici, Mohsen Seyedkazemi Ardebili, Andrea Bartolini, and Zeynep Kiziltan. 2023. PM100: A job power consumption dataset of a large-scale production HPC system. In *Proceedings of the SC'23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*. Zenodo, Denver, CO, 1812–1819.

[6] Jyotika Athavale, Cullen Bash, Wesley Brewer, Matthias Maiterth, Dejan Milojicic, Harry Petty, and Soumyendu Sarkar. 2024. Digital Twins for Data Centers. *Computer* 57, 10 (2024), 151–158. https://doi.org/10.1109/MC.2024.3436945

[7] Robin Boëzennec, Fanny Dufossé, and Guillaume Pallez. 2024. Qualitatively Analyzing Optimization Objectives in the Design of HPC Resource Manager. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 9, 4, Article 15 (Dec. 2024), 28 pages. https://doi.org/10.1145/3701986

[8] Andrea Borghesi, Carmine Di Santi, Martin Molan, Mohsen Seyedkazemi Ardebili, Alessio Mauri, Massimiliano Guarrasi, Daniela Galetti, Mirko Cestari, Francesco Barchi, Luca Benini, et al. 2023. M100 exadata: a data collection campaign on the cineca's marconi100 tier-0 supercomputer. *Scientific Data* 10, 1 (2023), 288.

[9] Wesley Brewer, Matthias Maiterth, Vineet Kumar, Rafal Wojda, Sedrick Bouknight, Jesse Hines, Woong Shin, Scott Greenwood, David Grant, Wesley Williams, and Feiyi Wang. 2024. A Digital Twin Framework for Liquid-cooled Supercomputers as Demonstrated at Exascale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '24)*. IEEE Press, Atlanta, GA, USA, Article 23, 18 pages. https://doi.org/10.1109/SC41406.2024.00029

[10] Wesley Brewer, Rafal Wojda, Matthias Maiterth, Sedrick Bouknight, Jesse Hines, Jake Webb, Rashadul Kabir, Bertrand Cirou, and Kevin Menear. 2025. exadigit/RAPS – S-RAPS branch. https://code.ornl.gov/exadigit/raps/-/tree/S-RAPS?ref_type=tags.

[11] Rajkumar Buyya and Manzur Murshed. 2002. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and computation: practice and experience* 14, 13-15 (2002), 1175–1220.

[12] Henri Casanova, Arnaud Giersch, Arnaud Legrand, Martin Quinson, and Frédéric Suter. 2014. Versatile, scalable, and accurate simulation of distributed applications and platforms. *J. Parallel and Distrib. Comput.* 74, 10 (2014), 2899–2917. https://doi.org/10.1016/j.jpdc.2014.06.008

[13] Steve J. Chapin, Walfredo Cirne, Dror G. Feitelson, James Patton Jones, Scott T. Leutenegger, Uwe Schwiegelshohn, Warren Smith, and David Talby. 1999. Benchmarks and Standards for the Evaluation of Parallel Job Schedulers. In *Job Scheduling Strategies for Parallel Processing*, Dror G. Feitelson and Larry Rudolph (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 67–90.

[14] Cirou. 2024. *Adastra jobs MI250 15days.* https://doi.org/10.5281/zenodo.14007065

[15] Pierre-François Dutot, Michael Mercier, Millian Poquet, and Olivier Richard. 2016. Batsim: a Realistic Language-Independent Resources and Jobs Management Systems Simulator. In *20th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*. Chicago, United States. https://hal.science/hal-01333471

[16] Oak Ridge Leadership Computing Facility. [n. d.]. Frontier User Guide. https://docs.olcf.ornl.gov/systems/frontier_user_guide.html

[17] Yuping Fan, Zhiling Lan, Paul Rich, William E. Allcock, Michael E. Papka, Brian Austin, and David Paul. 2019. Scheduling Beyond CPUs for HPC. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing* (Phoenix, AZ, USA) *(HPDC '19)*. Association for Computing Machinery, New York, NY, USA, 97–108. https://doi.org/10.1145/3307681.3325401

[18] Ana Gainaru, Hongyang Sun, Guillaume Aupy, Yuankai Huo, Bennett A Landman, and Padma Raghavan. 2019. On-the-fly scheduling versus reservation-based scheduling for unpredictable workflows. *The International Journal of High Performance Computing Applications* 33, 6 (2019), 1140–1158. https://doi.org/10.1177/1094342019841681 arXiv:https://doi.org/10.1177/1094342019841681

[19] Cristian Galleguillos, Alina Sîrbu, Zeynep Kiziltan, Ozalp Babaoglu, Andrea Borghesi, and Thomas Bridi. 2018. Data-Driven Job Dispatching in HPC Systems. In *Machine Learning, Optimization, and Big Data*, Giuseppe Nicosia, Panos Pardalos, Giovanni Giuffrida, and Renato Umeton (Eds.). Springer International Publishing, Cham, 449–461.

[20] Jean-Sébastien Gay and Yves Caniou. 2006. Simbatch: an API for simulating and predicting the performance of parallel resources and batch systems. Research Report.

[21] Alexander V. Goponenko, Kenneth Lamar, Christina Peterson, Benjamin A. Allan, Jim M. Brandt, and Damian Dechev. 2022. Metrics for Packing Efficiency and Fairness of HPC Cluster Batch Job Scheduling. In *2022 IEEE 34th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, Bordeaux, France, 241–252. https://doi.org/10.1109/SBAC-PAD55451.2022.00035

[22] S. Greenwood, V. Kumar, and W. Brewer. 2024. Thermo-fluid Modeling Framework for Supercomputer Digital Twins: Part 2, Automated Cooling Models. In *America Modelica Conference*. Modelica Association, 210–219.

[23] David Jackson, Quinn Snell, and Mark Clement. 2001. Core Algorithms of the Maui Scheduler. In *Job Scheduling Strategies for Parallel Processing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 87–102.

[24] Dalibor Klusáček, Mehmet Soysal, and Frédéric Suter. 2019. Alea–complex job scheduling simulator. In *International Conference on Parallel Processing and Applied Mathematics*. Springer, Bialystok, Poland, 217–229.

[25] Vineet Kumar, Scott Greenwood, Wes Brewer, Wesley Williams, David Grant, and Nathan Parkison. 2024. Thermo-Fluid Modeling Framework for Supercomputer Digital Twins: Part 1, Demonstration at Exascale. In *PROCEEDINGS OF THE AMERICAN MODELICA CONFERENCE 2024*. Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States). https://www.osti.gov/biblio/2480044

[26] Lawrence Livermore National Laboratory. 2024. Livermore Archive for System Telemetry (LAST). https://github.com/LLNL/LAST

[27] Tapasya Patki, Adam Bertsch, Ian Karlin, Dong H Ahn, Brian Van Essen, Barry Rountree, Bronis R de Supinski, and Nathan Besaw. 2021. Monitoring large scale supercomputers: A case study with the lassen supercomputer. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, Portland, OR, United States, 468–480.

[28] Tapasya Patki, Zachary Frye, Harsh Bhatia, Francesco Di Natale, James Glosli, Helgi Ingolfsson, and Barry Rountree. 2019. Comparing gpu power and frequency capping: A case study with the mummi workflow. In *2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*. IEEE, 31–39.

[29] Chris N. Potts and Mikhail Y. Kovalyov. 2000. Scheduling with batching: A review. *European Journal of Operational Research* 120, 2 (2000), 228–249. https://doi.org/10.1016/S0377-2217(99)00153-8

[30] Dongxu Ren, Wei Tang, Xu Yang, Yuping Fan, and Zhiling Lan. 2017. https://github.com/SPEAR-IIT/CQSim

[31] Gonzalo P Rodrigo, Erik Elmroth, Per-Olov Östberg, and Lavanya Ramakrishnan. 2017. ScSF: A scheduling simulation framework. In *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, Cham, DE, 152–173.

[32] Greyce N. Schroeder, Charles Steinmetz, Carlos E. Pereira, and Danubia B. Espindola. 2016. Digital Twin Data Modeling with AutomationML and a Communication Methodology for Data Exchange. *IFAC-PapersOnLine* 49, 30 (2016), 12–17. https://doi.org/10.1016/j.ifacol.2016.11.115 4th IFAC Symposium on Telematics Applications TA 2016.

[33] Nikolay A. Simakov, Robert L. DeLeon, Martins D. Innus, Matthew D. Jones, Joseph P. White, Steven M. Gallo, Abani K. Patra, and Thomas R. Furlani. 2018. Slurm Simulator: Improving Slurm Scheduler Performance on Large HPC systems by Utilization of Multiple Controllers and Node Sharing. In *Proceedings of the Practice and Experience on Advanced Research Computing: Seamless Creativity* (Pittsburgh, PA, USA) *(PEARC '18)*. Association for Computing Machinery, New York, NY, USA, Article 25, 8 pages. https://doi.org/10.1145/3219104.3219111

[34] Nikolay A. Simakov, Robert L. Deleon, Yuqing Lin, Phillip S. Hoffmann, and William R. Mathias. 2022. Developing Accurate Slurm Simulator. In *Practice and Experience in Advanced Research Computing 2022: Revolutionary: Computing, Connections, You* (Boston, MA, USA) *(PEARC '22)*. Association for Computing Machinery, New York, NY, USA, Article 59, 4 pages. https://doi.org/10.1145/3491418.3535178

[35] Nikolay A. Simakov, Martins D. Innus, Matthew D. Jones, Robert L. DeLeon, Joseph P. White, Steven M. Gallo, Abani K. Patra, and Thomas R. Furlani. 2018. A Slurm Simulator: Implementation and Parametric Analysis. In *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation*, Stephen Jarvis, Steven Wright, and Simon Hammond (Eds.). Springer International Publishing, Cham, Germany, 197–217.

[36] Joseph Skovira, Waiman Chan, Honbo Zhou, and David A. Lifka. 1996. The EASY - LoadLeveler API Project. In *Job Scheduling Strategies for Parallel Processing, IPPS'96 Workshop, Honolulu, Haiwai, USA, April 16, 1996, Proceedings (Lecture Notes in Computer Science, Vol. 1162)*, Dror G. Feitelson and Larry Rudolph (Eds.). Springer, 41–47. https://doi.org/10.1007/BFB0022286

[37] Ana Luisa Veroneze Solórzano, Kento Sato, Keiji Yamamoto, Fumiyoshi Shoji, Jim M. Brandt, Benjamin Schwaller, Sara Petra Walton, Jennifer Green, and Devesh Tiwari. 2024. Toward Sustainable HPC: In-Production Deployment of Incentive-Based Power Efficiency Mechanism on the Fugaku Supercomputer. , 16 pages. https://doi.org/10.1109/SC41406.2024.00030

[38] Atsuko Takefusa, Satoshi Matsuoka, Kento Aida, Hidemoto Nakada, and Umpei Nagashima. 1999. Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms. In *Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing (HPDC '99)*. IEEE Computer Society, USA, 11.

[39] Fei Tao, He Zhang, Ang Liu, and Andrew Y. C. Nee. 2019. Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics* 15 (2019), 2405–2415. https://api.semanticscholar.org/CorpusID:68170459

[40] Yi-Ping Chen Vispi Karkaria, Ying-Kuan Tsai and Wei Chen. 2025. An optimization-centric review on integrating artificial intelligence and digital twin technologies in manufacturing. *Engineering Optimization* 57, 1 (2025), 161–207. https://doi.org/10.1080/0305215X.2024.2434201

[41] Alex Wilkinson, Jess Jones, Harvey Richardson, Tim Dykes, and Utz-Uwe Haus. 2023. A Fast Simulator to Enable HPC Scheduling Strategy Comparisons. In *High Performance Computing: ISC High Performance 2023 International Workshops, Hamburg, Germany, May 21–25, 2023, Revised Selected Papers*. Springer-Verlag, Berlin, Heidelberg, 320–333. https://doi.org/10.1007/978-3-031-40843-4_24

[42] Rafal P. Wojda, Matthias Maiterth, Sedrick Bouknight, and Wesley Brewer. 2024. Dynamic Modeling of Power Conversion Stages for an Exascale Supercomputer. In *2024 IEEE Energy Conversion Congress and Exposition (ECCE)*. 1595–1601. https://doi.org/10.1109/ECCE55643.2024.10861715