An LLM-enabled Workflow for Understanding and Evolving HPC Scheduling Practices

Anderson Borch
Colorado State University
Oak Ridge National Lab.
Fort Collins, CO, USA

Ketan Maheshwari
Oak Ridge National Lab.
Oak Ridge, TN, USA

Justin M. Wozniak
Argonne National Lab.
Lemont, IL, USA

Rafael Ferreira da Silva
Oak Ridge National Lab.
Oak Ridge, TN, USA

Abstract

Emerging AI-enhanced and near real-time scientific workloads are challenging the traditional assumptions of HPC job scheduling systems. In this work, we present an LLM-enabled, portable workflow for analyzing a subset of Slurm job trace data from leadership-class supercomputers, exemplified through over 1.5 million jobs and 18 million job-steps on OLCF's Frontier system. Our hybrid workflow integrates a static data analysis pipeline with dynamic, AI-powered components to generate interactive dashboards and automated, interpretable insights into scheduling behavior, efficiency, and system usage patterns. We demonstrate the workflow's portability across HPC systems and show how AI-driven interpretations augment traditional visualization to uncover inefficiencies, guide policy evolution, and support more responsive scheduling strategies, enabling consistent analytics across HPC system architectures. This approach enables computer science researchers and HPC sysadmins to systematically evaluate workload characteristics and adapt HPC resource management to meet the evolving demands of modern scientific discovery.

CCS Concepts

• **Computing methodologies** → **Distributed computing methodologies**; **Multi-agent systems**; **Distributed computing methodologies**; **Parallel computing methodologies**.

Keywords

HPC scheduling, Slurm job analytics, Workload characterization, AI-enabled workflows

ACM Reference Format:
Anderson Borch, Ketan Maheshwari, Justin M. Wozniak, and Rafael Ferreira da Silva. 2025. An LLM-enabled Workflow for Understanding and Evolving HPC Scheduling Practices. In *Proceedings of 1st Workshop on Workflows, Intelligent Scientific Data, and Optimization for Automated Management (WISDOM 2025).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

WISDOM 2025, San Diego, CA

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

High-performance computing (HPC) centers play a central role in enabling cutting-edge scientific research across a wide range of domains. These systems have been optimized for large-scale, compute-intensive simulations that run in a batch mode. However, the nature of scientific discovery is changing as emerging workloads driven by the Department of Energy's Integrated Research Infrastructure (IRI) program [11], artificial intelligence (AI) models [8], and autonomous scientific workflows [4] are increasingly dynamic, interactive, and data-driven. These workloads often require rapid job turnaround, seamless coupling with experimental facilities, and intelligent orchestration across multiple resources and sites. The shift toward more agile and responsive science is placing new demands on HPC systems, which must now accommodate workloads that cannot be effectively served by existing scheduling policies.

Many of these new scientific applications involve near real-time experimental data collection, iterative decision-making, and AI-driven steering [5]. For example, applications that analyze incoming data from synchrotrons or beamlines must produce results quickly enough to influence ongoing experiments. Similarly, AI training and inference tasks often require launching a large number of small, short-lived jobs, or maintaining persistent interactive sessions [8]. These use-cases are ill-suited to conventional HPC scheduling policies, which are primarily designed for maximizing throughput and fairness among long-running jobs. As a result, users are increasingly encountering limitations in responsiveness, flexibility, and resource adaptability when deploying these workloads on current HPC systems.

Several HPC centers have made incremental changes to better support diverse workloads, such as introducing debug partitions for short interactive jobs, enabling job arrays for task parallelism, or adopting preemptible queues for lower-priority jobs. For instance, NERSC has taken steps toward supporting near real-time and experiment-driven workflows through its Superfacility model [1], which integrates HPC with experimental data streams and provides dedicated near real-time Quality of Service (QoS) settings. However, such capabilities remain the exception rather than the norm. Most centers continue to rely on scheduling policies that assume static, resource-hungry jobs with generous lead times, which limits their ability to support urgent, event-triggered, or low-latency computing demands. Furthermore, while job trace archives exist and have enabled various one-time analysis studies, a widely adopted and systematic framework for data-driven analysis of job history to uncover insights into system behavior and scheduling efficiency is still lacking.

Existing research on HPC scheduling has generally focused on static optimization strategies or simulated workloads, without thoroughly addressing the performance implications of emerging near real-time and AI-enhanced use cases. There is a lack of systematic studies that examine how current scheduling policies accommodate (or hinder) the requirements of modern workflows, particularly those involving adaptive feedback loops, autonomous agents, and cross-facility coordination. Moreover, there are few practical tools available for HPC sysadmins to explore historical scheduler data in a way that directly informs policy evolution. This gap makes it difficult to anticipate how changes in workload patterns might affect resource allocation, queue latency, and overall system performance.

To address these challenges, we propose a reusable and extensible workflow for analyzing the historical job scheduling data, with an initial focus on Slurm-based HPC systems due to their widespread adoption. Although the workflow is designed around Slurm, its structure allows adaptation to other job schedulers across HPC centers. This workflow provides a practical mechanism for systematically exploring how HPC resources are utilized, where bottlenecks arise, and how workloads of different types behave under current scheduling policies. It supports filtering, formatting, and transforming job records, generating visualizations that reveal trends in queue wait times, resource usage, job states, and backfilling efficiency. Additionally, the workflow includes an AI-based component that performs interpretive analyses of the data and augments human-driven exploration with automated insights.

By applying this workflow to historical job data from the Oak Ridge Leadership Computing Facility (OLCF), we demonstrate how sysadmins and researchers can extract actionable information that informs the design of more adaptive, flexible, and workload-aware scheduling strategies. Our goal is not only to characterize past usage but to enable proactive planning and policy development for the future. This includes guiding decisions around queue configurations, preemptive scheduling, node sharing, near real-time prioritization, and workflow-aware resource allocation. The workflow's design ensures portability across HPC systems, enabling consistent data analysis regardless of system architecture or workload profile. Last, this work supports the broader transformation of HPC systems from static batch processors into responsive, intelligent platforms capable of supporting the next generation of scientific discovery.

The remainder of this paper is organized as follows. Section 2 describes the Slurm job trace dataset used in our study. Section 3 details the structure of the hybrid workflow, including both static analysis and AI-based components. Section 4 presents the results of applying the workflow to data from OLCF systems, highlighting insights and demonstrating its portability. Section 5 reviews related work in HPC scheduling and job trace analytics. Finally, Section 6 concludes the paper and outlines future directions for extending the workflow's capabilities.

2 Slurm Job Trace Dataset

The analyses presented in this paper leverage historical job records from the Slurm scheduler at OLCF, specifically from the Frontier supercomputer. The dataset spans from April 2023 (when Frontier went into production) through 2024 and includes over 0.5 million jobs submitted by more than 1,000 users (Figure 1). These jobs generated over 7 million job steps, where each step generally represents a distinct execution unit within a job, often corresponding to a specific task or parallel program launched during the job's runtime. The job records were collected using the `sacct` command, which provides access to a wide range of Slurm accounting fields. This comprehensive dataset captures the operational behavior of the system across multiple years and offers a detailed view into usage patterns, job structures, and resource allocation behaviors.

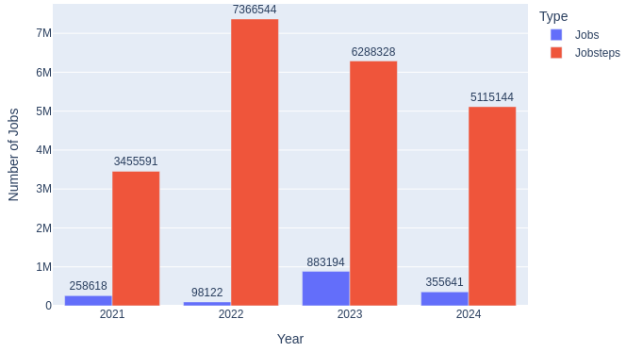

Figure 1: Total number of jobs and job-steps executed on the Frontier supercomputer from 2021 to 2024. The figure highlights the high volume of activity and the prevalence of multi-step jobs, reflecting diverse workload patterns and extensive use of task parallelism through `srun`.

From the 118 fields available in the Slurm accounting database, a subset of 50+ fields was selected based on their relevance and utility for characterizing scheduling behavior and enabling downstream analytics. Redundant, sensitive, or less informative fields, such as those offering duplicative time representations (e.g., `Elapsed` vs. `ElapsedRaw`), were excluded. This curated dataset includes key information on job timing, resource requests and usage, job state, account associations, and scheduling flags such as backfill indicators, allowing for deep exploration of queue dynamics and system load patterns.

A light preprocessing step was performed to normalize and clean the extracted data. For instance, certain fields required unit conversions (e.g., node counts expressed as 'K' for thousands) or formatting adjustments (e.g., converting raw seconds to minutes for readability). Additionally, malformed records, mostly associated with hardware errors and accounting for less than 0.002% of the total, were discarded. The resulting dataset enabled robust visualization and AI-based interpretation of scheduling phenomena. Table 1 summarizes the key fields selected for this study, grouped by category.

Figure 1 provides a high-level overview of job and job-step volume on the Frontier system across the analyzed period from 2021 to 2024. Note that the jobs run in 2021 until April 2023 pertains to acceptance tests and early hero runs. We exclude those jobs from the study presented here for consistency. The visualization illustrates the scale and consistency of workload submitted to the system, showing sustained high-throughput usage patterns typical of a leadership-class supercomputer. The plot shows that, while job submissions remained relatively stable each year, the number

Category	Selected Fields
Job Identification	JobID, Partition, Reservation, ReservationID
Timing Information	SubmitTime, StartTime, EndTime, Elapsed, Timelimit
Resource Requests	NNodes, NCPUs, NTasks, ReqMem, ReqGRES, Layout
Resource Usage	VMSize, AveCPU, MaxRSS, TotalCPU, NodeList, ConsumedEnergy
IO Related	WorkDir, AveDiskRead, AveDiskWrite, MaxDiskRead, MaxDiskWrite
Job State	State, ExitCode, Reason, Suspended, Restarts, Constraints
Scheduling Metadata	Priority, Eligible, QOS & QOSReq, Flags, TRESUsageInAve, TRESReq
Special Indicators	Backfill (from Flags), Dependency, ArrayJobID
Misc	Comment, SystemComment, AdminComment

Table 1: Selected fields from Slurm accounting data used in this study.

of job-steps was significantly higher than the job count. This reflects the frequent use of `srun` to launch multiple tasks within a single job. The distinction between jobs and job-steps is important because many scientific workflows depend on fine-grained task execution that occurs at the job-step level rather than through single, monolithic jobs.

3 Workflow Structure and Analysis Pipeline

This workflow is designed to analyze historical job scheduling data from HPC scheduling systems, with the objective of uncovering patterns, diagnosing inefficiencies, and guiding policy refinement. The workflow is organized into two main components: a **data analysis subworkflow**, which is static and applies to any dataset of interest, and **user-defined subworkflow**, which allows dynamic integration of domain-specific AI/LLM analyses. Figure 2 visually represents the structure and flow between these components.

3.1 Data Analysis Subworkflow

The following steps, highlighted in blue in Figure 2, compose the core of the workflow. These are executed in a fixed sequence and structure, driven by the scope and granularity of the dataset:

Obtain data queries the Slurm database for a curated set of 60 accounting fields and writes the output in text format to a designated directory. This stage is highly flexible and fully parameterized: users can define the desired date range (e.g., spanning multiple years), choose the data granularity (yearly or monthly), and indicate whether previously cached data should be used. If cached data is unavailable, the system automatically fetches fresh records and stores them in the specified output location. For large-scale retrievals across many months or years, GNU Parallel is employed to execute multiple database queries concurrently, improving performance and scalability.

Curate Data cleans the raw output by removing malformed entries and reformats the dataset from pipe-separated text to CSV for compatibility with Python-based analysis libraries.

Field-Specific Plotting encompasses several analytical stages that each generate visualizations for a selected set of Slurm fields, providing insights into system behavior and job characteristics. For instance, these may include stacked bar plots of job end states per user (*Job States*), scatter plots of queue wait times color-coded by final job status (*Wait Times*), and comparative scatter plots that highlight the difference between requested and actual runtimes to analyze scheduler backfilling behavior (*Backfill*). Each stage uses Plotly to produce interactive HTML charts that support zooming and filtering for exploratory analysis.

Dashboard consolidates all generated plots into an interactive dashboard using Plotly Dash, enabling users to explore and filter results from a single unified interface.

3.2 User-Defined Subworkflows

The stages shown in orange in Figure 2 represent user-defined, dynamic subworkflows that can be customized based on analysis goals. These subworkflows are modular and can evolve to incorporate additional tools or insights tailored to the user's needs. For example, scientists running workloads on Frontier might define workflows to analyze job efficiency, identify bottlenecks, or correlate performance with input parameters. In contrast, system administrators may use the framework to detect scheduling anomalies, monitor system utilization trends, or evaluate the effectiveness of recent configuration changes.

In this paper, their primary purpose is to enable richer interpretation of the static analysis outputs via AI-assisted techniques:

HTML2PNG Converts HTML-based plots into PNG images using browser automation with the Firefox screenshot utility in headless CLI mode. This step enables compatibility with LLM tools, which are not well-suited to process large raw datasets directly. Instead, the plots serve as compact visual summaries of the data, allowing the LLMs to extract insights more efficiently from these visual representations.

LLM Compare Sends paired PNG images to an LLM API for comparative analysis between visualizations. Here, the model is provided with two related images and the following prompt:

> *Act as a data scientist to compare and contrast the two provided charts. Provide a quantitative and qualitative analysis of the key trends, relationships, and statistics, highlighting similarities and differences. Be specific and mention any notable patterns or outliers. Calculate meaningful statistics from the plots.*

LLM Insight Sends individual PNG plots to the LLM for summarization and insight generation. Here, the prompt is tailored to summarize a single chart:

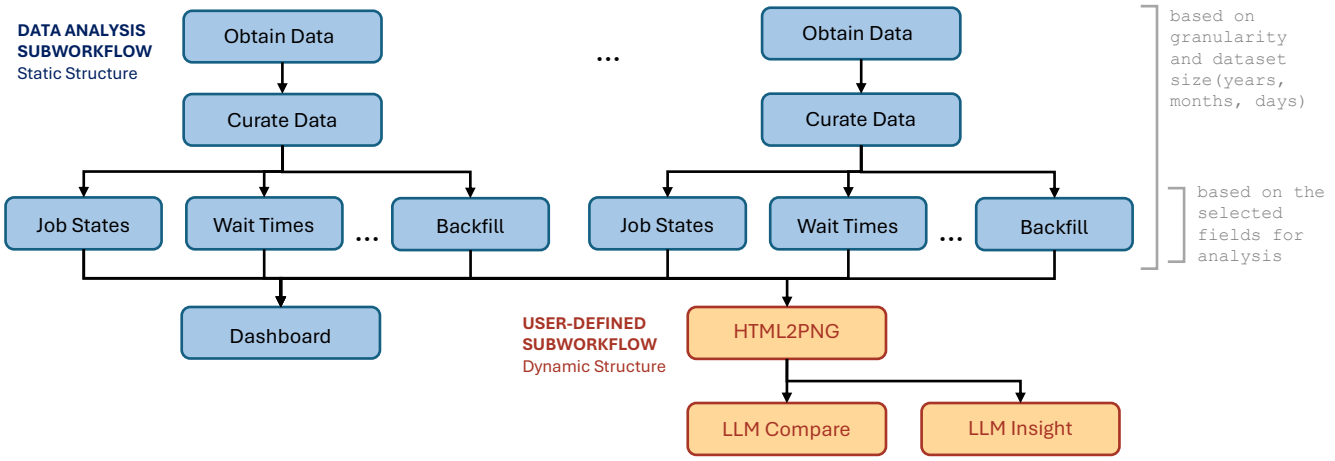

Figure 2: Overview of the hybrid workflow combining a static data analysis pipeline with dynamic, user-defined AI components. Blue elements represent fixed analytical stages applied to Slurm job data, while orange elements indicate customizable extensions for automated chart interpretation and insight generation. Tasks in the same horizontal row may be executed concurrently by the workflow.

> *Act as a data scientist to summarize the chart and provide a quantitative analysis of the key trends, relationships, and statistics of the provided chart. Be specific and mention any notable patterns or outliers. Calculate meaningful statistics from the plot.*

These dynamic steps are optional and extensible, allowing the workflow to adapt to future needs or incorporate alternative AI models and APIs.

To choose the most suitable LLM for this task, we conducted a brief survey of available options (Table 2). Key factors included accessibility (API availability), support for image input, cost, and performance. While models like OpenAI GPT-4, Anthropic Claude, and DeepSeek offer advanced capabilities, they are either paywalled, geo-restricted, or come with usage limitations. Others like Meta Llama and Apple's on-device LLMs either lack public API support or are designed for local-only inference.

We chose Google's **Gemma 3** as the LLM backend for this workflow for its advantages: (1) Free API access with no usage restrictions; (2) Strong support for multimodal input, including image-based prompts; (3) Low latency and lightweight footprint, making it suitable for integration in real-time or automated workflows.

Although this LLM integration currently serves as a proof of concept, and we do not claim scientific rigor for all generated insights, it demonstrates the feasibility of using AI-driven analysis to complement traditional HPC data exploration and visualization.

3.3 Workflow Composition and Concurrency

The workflow was implemented using the Swift/T parallel scripting language [17], enabling scalable execution of both the static and AI-driven components. Its source code is publicly available at [10], encouraging community reuse and extension. The modular architecture and use of Swift/T scripting facilitate deployment on various HPC platforms, ensuring portability and scalability without requiring system-specific customizations.

The workflow script generalizes across diverse workloads and HPC systems, but also supports extension through user-defined AI functional subcomponents. These include shell scripts and Python scripts that invoke the LLM operations described in Section 3.2. The main workflow, defined in `workflow.swift`, implements a *parallel pipelines* model, in which each dataset retrieved from the Slurm database is processed concurrently in a "subworkflow." The various tasks making up the subworkflow are coded as an apparently linear list of the functional subcomponents with input and output file references; however, Swift/T automatically determines the data dependencies and produces/executes the dataflow diagram illustrated in Figure 2, producing additional available concurrency.

The workflow is invoked with:

```
$ swift-t -m local|slurm -n N workflow.swift \
          date_spec date1 date2 cache data
```

for local or Slurm execution, respectively. `N` is the number of Swift/T processes to start (the physical concurrency). The workflow arguments `date_spec` and dates specify the query to the Slurm database, `cache` is a location on a fast filesystem to use as a cache, and `data` is a permanent filesystem location for data reuse.

4 Advancing HPC Scheduling Analysis with Visualization and AI

This section presents an analysis of job scheduling data from the Frontier supercomputer using the developed workflow. Through a combination of interactive visualizations and AI-generated insights, we demonstrate how this workflow uncovers inefficiencies and patterns that can inform more responsive scheduling strategies.

4.1 Visual and Analytical Insights into HPC Scheduling Behavior

The developed workflow provides an in-depth, interactive lens into the operational patterns of HPC systems, transforming static Slurm logs into dynamic visual narratives. Figure 3, which plots

LLM / AI	Version	API	Access	Remarks
OpenAI	All Models	Yes	Paid	03, 04, best for vision
Google	Gemini 2.5 Flash	Yes	Free	No limit on usage
Google	Gemma 3	Yes	Free	AI for "developers"
Anthropic	All Models	Yes	Paid	Interoperable with other models
Apple	All Models	No	Free	All LLMs must run locally on iOS devices
DeepSeek	All Models	Yes	Paid	Geo-restricted
Mistral	All Models	Yes	Paid	Restricted and limited free trial
Meta	Llama	Yes	Unclear	Waitlist for API, cost unclear
Microsoft	Copilot	Yes	Paid	Integrated into MS tools eg. Office suite
Github	Copilot	No	Free	Built into IDE, limited req/month

Table 2: Brief comparison of selected LLM offerings based on API availability, cost, and support for image input.

allocated nodes versus job durations from 2021 through 2024, reveals the diverse scale and intensity of compute jobs. It illustrates the system's ability to accommodate both small, short-lived jobs and massively parallel, long-duration tasks. This diversity reflects the growing complexity of modern workloads and highlights the necessity for scheduling strategies that can flexibly accommodate a broad spectrum of resource needs.

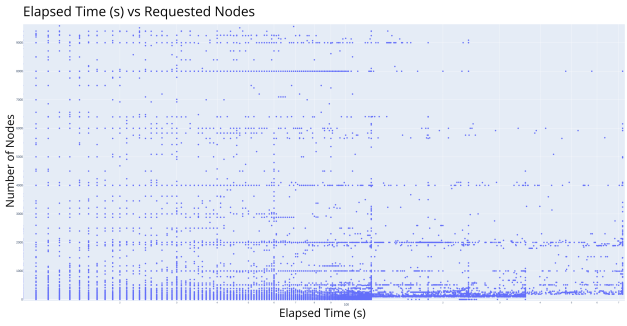

Figure 3: Allocated compute nodes versus job elapsed time for all jobs executed on Frontier between April 2023 and December 2024.

Figure 4 provides a granular view of job queue wait times in 2024, color-coded by job completion state. This visualization captures important temporal and operational characteristics, such as spikes in wait times that could be linked to specific usage patterns or policy inefficiencies. The end-state differentiation offers valuable insights into how resource contention, prioritization, or backfilling mechanisms might impact overall system reliability and throughput. While outliers are omitted for clarity, the underlying distribution shows distinct stratifications that warrant further analysis and tuning of scheduling parameters.

Figure 5 builds on this by presenting job end states per user, enabling a multifaceted view of user behavior, job completion success, and potential system friction points. The inclusion of state color-coding within user-level breakdowns makes it easier to identify users with disproportionately high failure or cancellation rates. These trends can guide training, user support, or system configuration changes. This figure demonstrates how the workflow can

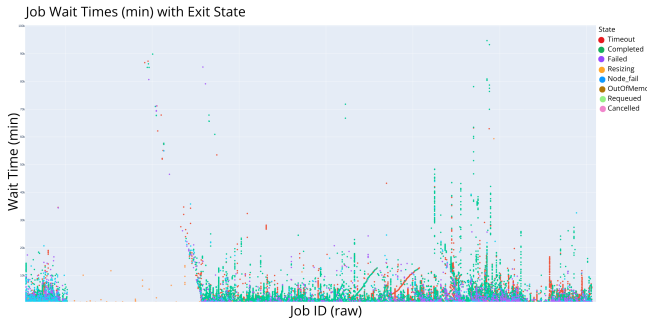

Figure 4: Visualization of job wait times in seconds for April 2023 till December 2024, color-coded by final job state (e.g., completed, failed, cancelled).

surface high-level usage metrics while maintaining the granularity necessary for targeted interventions.

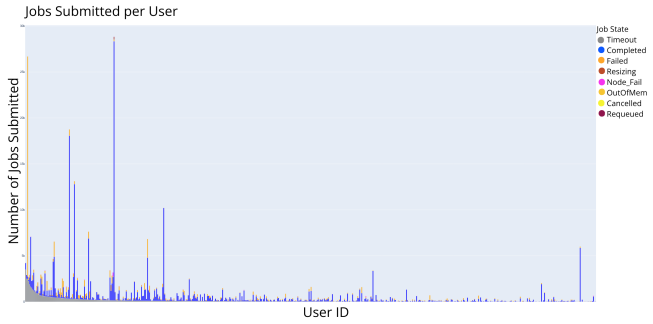

Figure 5: Distribution of job completion states per user for a selected April 2023 till December 2024 data subset.

Figure 6 dives deeper into scheduling efficiency by visualizing the discrepancy between requested walltimes and actual job durations. Here, regular jobs and backfilled jobs are distinguished by dot and plus symbols, respectively. The chart shows that many jobs, particularly backfilled ones, complete in less time than requested, revealing underutilization and missed opportunities for finer-grained resource scheduling. This insight is particularly relevant for designing adaptive scheduling policies that reclaim unused time to reduce queue delays and improve system throughput.

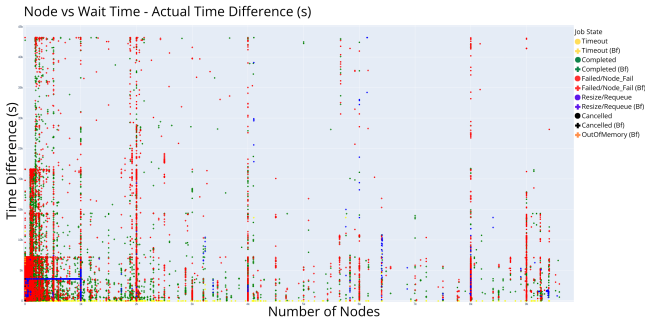

Figure 6: Scatter plot comparing requested walltimes and actual job durations for April 2023 till December 2024, with backfilled jobs marked by plus symbols and standard jobs by dots.

Together, these visualizations showcase the flexibility and potential impact of our workflow. They not only present historical trends in an interpretable format but also identify areas for policy refinement. The interactivity of the dashboard—zooming, filtering, and overlaying dimensions—makes this tool suitable for both exploratory analysis and strategic planning. The workflow's ability to adapt to different query types and visualizations means it can serve administrators, performance engineers, and researchers alike.

4.2 Augmenting Analysis with LLM-Derived Interpretations

To extend the analytical capabilities of the workflow, we integrated an AI-based subworkflow that applies LLMs to interpret individual and comparative visualizations. These enhancements augment human reasoning by transforming complex visual information into accessible summaries. The LLM component enhances situational awareness by surfacing patterns, anomalies, and inefficiencies that may not be immediately visible in complex charts. It effectively acts as a digital analyst, distilling key messages and pointing users toward impactful observations. This integration serves as a bridge between data science and decision-making, helping both novice and expert users to understand the implications of scheduler behavior.

For example, when comparing two scatter plots showing queue wait times across months, the LLM offered the following observation[1]:

> *The majority of jobs that completed successfully have shorter wait times in June compared to March, suggesting either a decrease in queue load or more efficient scheduling policies being implemented. Notably, March has a higher density of jobs with extended wait times exceeding 100,000 seconds, which could indicate batch congestion or policy thresholds being hit more frequently.*

This kind of temporal comparison provides actionable insights for system administrators and policy designers. It can trigger further investigation into scheduling rule changes, maintenance windows, or emerging workload profiles.

Another LLM interpretation, focused on differences between requested and actual walltimes, highlighted inefficiencies in user-specified estimates[2]:

> *There is a consistent trend of users significantly overestimating their walltime requests. This creates a systemic gap that reduces scheduling efficiency. The presence of tightly clustered short-actual, long-requested jobs suggests a potential for implementing automated time prediction or adaptive rescheduling mechanisms.*

Such commentary encourages proactive improvements in user guidance, automated tuning, or system-level enforcement of predictive runtime estimates. The AI-generated insights also help justify the development of smarter scheduling models that can learn from past trends.

In summary, the LLM module adds a new interpretive layer to traditional visual analytics. Its ability to quantify, compare, and narrate differences between visualizations makes the workflow more accessible and decision-supportive. By summarizing complex visual data into human-readable guidance, the workflow becomes a powerful tool for improving scheduler transparency, user behavior, and system adaptability.

4.3 Workflow Portability

To demonstrate the workflow's portability and generalizability, we deployed it on Andes, a general-purpose system at OLCF that also uses the Slurm scheduler. Unlike Frontier, which supports large-scale GPU-intensive workloads, Andes has a CPU-centric architecture and serves a wider range of job sizes and scientific domains. In addition to hardware differences, the systems also differ in configuration details such as partition structures and job scheduling policies. We collected data from Andes for the full year of 2024 and applied the same workflow without modification, highlighting its adaptability across both system configurations and workload profiles. These results also suggest opportunities for future workload-aware tuning of the workflow components.

Figure 7 presents the allocated compute nodes versus job durations for Andes, enabling a direct comparison to Frontier's data in Figure 3. While both systems support a wide range of job scales and runtimes, Andes exhibits a denser concentration of short-duration jobs with fewer nodes, reflecting its suitability for smaller, throughput-oriented workloads. In contrast, Frontier's distribution includes a larger fraction of high-node, long-duration jobs, consistent with its exascale mission. This difference underscores the value of flexible scheduling strategies that are tuned to system profiles: Frontier benefits from policies that manage parallelism at scale, while Andes requires optimizations for high job turnover and interactive usage.

Figure 8 shows the distribution of job completion states per user on Andes in 2024, comparable to the Frontier view in Figure 5. Interestingly, Andes users tend to have fewer failed or canceled jobs overall, possibly due to more interactive or exploratory work that allows faster feedback cycles and user adaptation. Additionally, the lower variance in failure rates across users suggests a more uniform

[1]Double-file LLM comparison results available at: https://github.com/Andy-Borch/ORNL-Work/blob/main/llm_analysis/llm_double_file_analysis.md

[2]Single-file LLM analysis results available at: https://github.com/Andy-Borch/ORNL-Work/blob/main/llm_analysis/llm_single_file_analysis.md

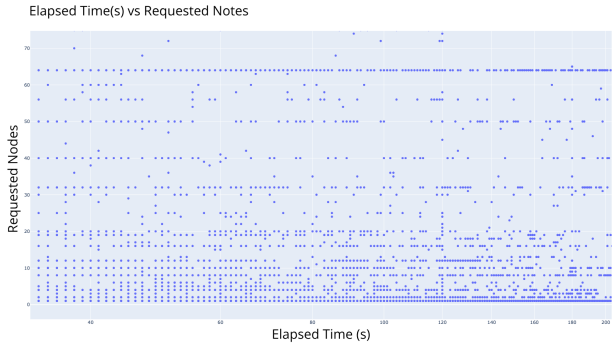

Figure 7: Allocated compute nodes versus job durations on Andes in 2024, showing a concentration of smaller, shorter jobs compared to Frontier.

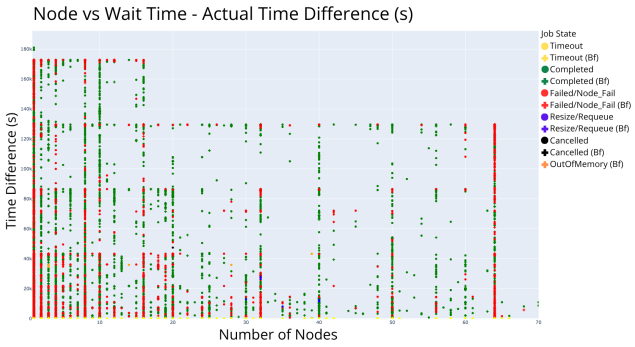

Figure 9: Comparison of requested versus actual walltimes for jobs on Andes in 2024, highlighting overestimation patterns and backfilling opportunities.

usage pattern on Andes compared to the more heterogeneous workload seen on Frontier, where some users dominate failure counts. These trends highlight the importance of tailoring user support, training, and system defaults to the dominant workload types and user behaviors on each system.

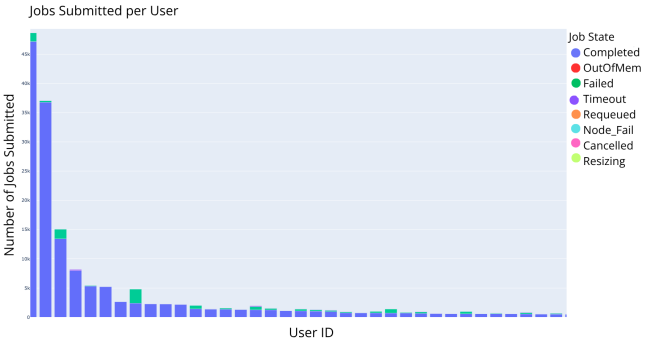

Figure 8: Distribution of job completion states per user on Andes in 2024, revealing lower failure rates and more consistent user behavior.

Figure 9 compares requested versus actual walltimes on Andes, paralleling the backfilling efficiency plot on Frontier shown in Figure 4. Similar inefficiencies are observed: a significant number of jobs request far more time than they use, especially for non-backfilled jobs. However, Andes demonstrates a tighter clustering of job durations and a more constrained range of walltime overestimation. This may reflect more conservative queue policies or a user base accustomed to fine-tuning resource estimates. Despite this, opportunities remain to improve scheduling efficiency by reclaiming unused time, perhaps through runtime prediction or adaptive rescheduling.

Overall, these comparative results affirm the workflow's portability across diverse HPC environments, and reveal how system-specific workload characteristics can drive different scheduling needs. The workflow not only generalizes across platforms with minimal configuration but also uncovers insights that can inform site-specific scheduler tuning. By supporting consistent analysis across systems like Frontier and Andes, it enables data-driven refinement of scheduling policies that reflect the operational realities of diverse HPC environments.

5 Related Work

The problem of HPC job scheduling has a long-standing history, with foundational strategies focused on maximizing throughput, fairness, and overall system utilization. Job schedulers such as Slurm, PBS, and LSF employ batch queuing models with priority-based algorithms and backfilling to optimize performance, especially for large-scale simulations. As scientific computing has evolved, however, these assumptions no longer hold for many emerging applications. Recent studies have highlighted growing workload heterogeneity in HPC systems, including increased use of short, data-intensive, and AI-driven jobs that require more agile and interactive scheduling mechanisms [3, 7, 15]. These modern workloads expose the limitations of current policies, such as poor support for responsiveness and inefficient handling of job granularity, and motivate the need for more flexible and intelligent schedulers [18].

In response, several efforts have explored enhancements to traditional job scheduling systems to better serve diverse and evolving workload requirements. For example, preemptive and opportunistic scheduling have been introduced to allow urgent or short jobs to interrupt low-priority or flexible workloads, as demonstrated by systems like TACC's "flex" queue and NERSC's "realtime" QoS. Node-sharing policies and job arrays have been employed to increase support for many-task workloads, such as hyperparameter tuning or ensemble simulations. For instance, Flux offers a modular, fully hierarchical job management architecture that enables fine-grained resource control, dynamic scheduling, and integration of heterogeneous workloads within a unified framework [6]. Flux's design allows it to co-schedule jobs at multiple levels (user, system, and workflow), making it well-suited for streaming and workflow-aware computing. Meanwhile, the NERSC Superfacility model integrates APIs and federated authentication to orchestrate real-time workflows between experimental facilities and HPC resources, showing early progress in bridging interactive science and scheduled computing [1]. Despite these developments, most HPC centers still operate under rigid policies that only partially support

emerging needs, and ad hoc configurations often lack generalizable evaluation frameworks.

Reconstructing processes from existing data and logs is a well-researched and practiced area in both computer systems [9, 14] and other domain sciences such as biomedicine [12]. Complementary to scheduling system enhancements, the analysis of historical job traces has become a valuable tool for understanding workload patterns and identifying inefficiencies in resource utilization. Several HPC centers, including NERSC, ALCF, OLCF, and BSC, have conducted workload characterizations using Slurm accounting databases and custom analytics pipelines to evaluate job submission behaviors, queue wait times, backfilling effectiveness, and CPU/GPU utilization [13, 15, 16]. Tools like Slurmmon [2] and job trace simulators provide administrators with insights into temporal trends and scheduling bottlenecks. However, few of these tools are extensible or designed to systematically support policy development tailored to modern workload types. There is a gap in frameworks that combine job trace analytics with AI-enhanced interpretation and integration into scheduling feedback loops. Moreover, very few studies have explicitly investigated how HPC systems must adapt to support IRI workflows, autonomous decision-making pipelines, or near real-time experiment analysis. This paper addresses that gap by proposing a flexible, AI-enabled workflow for analyzing historical Slurm job data and extracting actionable insights to guide future scheduling strategies.

6 Conclusions and Future Work

This work introduces a reusable, extensible workflow for analyzing Slurm job scheduling data that blends traditional visual analytics with AI-generated insights to better understand and evolve HPC scheduling practices. By applying our approach to data from OLCF's Frontier system, we revealed key trends in job structure, wait times, backfilling inefficiencies, and underutilized walltime estimates. The workflow's interactive dashboard facilitates user exploration, while the LLM component enhances decision-making by surfacing patterns that are not immediately visible. Together, these capabilities support data-driven policy adjustments and greater scheduler responsiveness, essential for accommodating dynamic, heterogeneous workloads. The workflow has been validated on two OLCF systems, demonstrating its portability, and is positioned as a valuable tool for both HPC administrators and researchers.

Future extensions of this workflow will focus on deeper integration with scheduling systems to support near real-time feedback and policy adaptation. This includes embedding AI-predicted walltime estimation into job submission workflows, enabling dynamic rescheduling and time reclamation. Additional work will explore multi-cluster and federated analytics, providing cross-facility visibility into scheduling behaviors and workload migration. We also plan to incorporate domain-specific insights by coupling the workflow with experiment-aware metadata (e.g., from beamline instruments or sensor networks), thereby enhancing situational context for autonomous scientific workflows. Finally, we aim to evaluate alternative LLM models and interactive agents that can guide users through visual narratives and recommend scheduling strategies in a more conversational and adaptive manner.

Acknowledgments

We thank Brian Etz for early discussions about backfill scheduling analyses and Katie Knight for helping explore python plotting libraries for this work. We thank Fred Suter for early review of the paper. This research used resources of the OLCF at ORNL, which is supported by DOE's Office of Science under Contract No. DE-AC05-00OR22725.

References

[1] Bjoern Enders, Debbie Bard, Cory Snavely, Lisa Gerhardt, Jason Lee, Becci Totzke, Katie Antypas, Suren Byna, Ravi Cheema, Shreyas Cholia, et al. 2020. Cross-facility science with the superfacility project at LBNL. In *2020 IEEE/ACM 2nd Annual Workshop on Extreme-scale Experiment-in-the-Loop Computing (XLOOP)*. IEEE, 1–7.

[2] Harvard FASRC. 2020. Slurmmon: Cluster Scheduling Analytics for Slurm. Available at https://github.com/fasrc/slurmmon.

[3] Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn. 2014. Job scheduling in multiprogrammed parallel systems. *J. Parallel and Distrib. Comput.* 74 (2014). Issue 1.

[4] Rafael Ferreira da Silva et al. 2024. *Shaping the Future of Self-Driving Autonomous Laboratories Workshop.* Technical Report ORNL/TM-2024/3714. Oak Ridge National Laboratory. doi:10.5281/zenodo.14430233

[5] Rafael Ferreira da Silva, Rosa M. Badia, Deborah Bard, Ian T. Foster, Shantenu Jha, and Frédéric Suter. 2024. Frontiers in Scientific Workflows: Pervasive Integration with HPC. *IEEE Computer* 57, 8 (2024). doi:10.1109/MC.2024.3401542

[6] Tom Grondo, Swen Boehm, Nate Grimmer, Ian Gyllinsky, and Todd Gamblin. 2019. Flux: Overcoming Scheduling Challenges for Exascale Workflows. In *Proceedings of the 2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS).*

[7] Nazmus Sakib Islam and Gokcen Kestor. 2023. Emerging Challenges in HPC Scheduling: A Workload Characterization Study. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC).*

[8] Shantenu Jha, Vincent R Pascuzzi, and Matteo Turilli. 2022. AI-coupled HPC workflows. *arXiv preprint arXiv:2208.11745* (2022).

[9] Jasper Paul Jürgensen. 2021. Trace reconstruction in system logs for processing with process mining. *Procedia Computer Science* 180 (2021), 352–357. doi:10.1016/j.procs.2021.01.173 Proceedings of the 2nd International Conference on Industry 4.0 and Smart Manufacturing (ISM 2020).

[10] Ketan Maheshwari. 2025. Scheduler Analysis via LLM Workflow GitHub Repository and Documentation. https://github.com/ketancmaheshwari/urgent-computing-sched

[11] William L Miller, Deborah Bard, Amber Boehnlein, Kjiersten Fagnan, Chin Guok, Eric Lançon, Sreeranjani Jini Ramprakash, Mallikarjun Shankar, Nicholas Schwarz, and Benjamin L Brown. 2023. *Integrated research infrastructure architecture blueprint activity (final report 2023).* Technical Report. US Department of Energy (USDOE), Washington, DC (United States). Office of ....

[12] M.L. Raghavan et al. 2014. Aneurysm shape reconstruction from biplane angiograms in the ISUIA collection. *Translational stroke research* 5 (2014), 252–259.

[13] Nika Sharma, Shane Canon, Shreyas Cholia, Valerie Hendrix, Lavanya Ramakrishnan, and Horst D Simon. 2021. Towards Superfacility: A system for orchestrating synchronous compute-data pipelines across distributed facilities. In *Proceedings of the Practice and Experience in Advanced Research Computing (PEARC).*

[14] FNU Shilpika, Bethany Lusch, Murali Emani, Venkatram Vishwanath, Michael E. Papka, and Kwan-Liu Ma. 2019. MELA: A Visual Analytics Tool for Studying Multifidelity HPC System Logs. In *2019 IEEE/ACM Industry/University Joint International Workshop on Data-center Automation, Analytics, and Control (DAAC).* 13–18. doi:10.1109/DAAC49578.2019.00008

[15] Tin-Yu Tseng, Shahar Shudler, and Mattan Erez. 2022. Slurm-Tracer: Characterizing Scheduling Delays and Queue Dynamics in HPC Systems. In *Proceedings of the International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC).*

[16] Pablo Vazquez, Davide Ricci, Emilio Luque, Rafael Mayo, and Jesús Labarta. 2022. Characterizing workload patterns and resource usage at a large-scale supercomputing center. *Concurrency and Computation: Practice and Experience* 34, 6 (2022).

[17] Justin M. Wozniak, Timothy G. Armstrong, Michael Wilde, Daniel S. Katz, Ewing Lusk, and Ian T. Foster. 2013. Swift/T: scalable data flow programming for many-task applications. *SIGPLAN Not.* 48, 8 (Feb. 2013), 309–310. doi:10.1145/2517327.2442559

[18] Andrew J. Younge, Shane Canon, and Maurício Tsugawa. 2021. Running AI and Simulation Workloads Together: The Case for Workflow-Driven Resource Management. *Future Generation Computer Systems* 122 (2021).