AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

AI-Augmented Dynamic GPU Scheduling: A Trace-Based Workload Analysis and Conceptual Framework for Large-Scale Machine Learning Inference

[1]*Sasank Varma Sarpella
[1]University of Alabama at Birmingham, USA.

Abstract

Large-scale machine learning inference environments nowadays are evolving into clusters of non-uniform GPUs with different workload durations, computational needs and submission rates. The study does not test a GPU scheduling algorithm, but rather it performs workload analysis and runtime prediction based on workload traces to present the concept of workload-aware GPU scheduling and to support an AI-informed dynamic GPU scheduling framework. The public dataset with 466,867 machine learning jobs from 84 organizations was processed using descriptive statistics, Spearman correlation, Kruskal–Wallis and Mann–Whitney tests, temporal workload analysis, requested capacity-time measures, and chronological run-time prediction with a HistGradientBoostingRegressor. The result indicated that job durations were highly skewed, GPU-model usage was also unbalanced, and there were significant temporal variations in arrivals, completions, and concurrent demand for GPU models. The run-time was positively associated with CPU request, GPU request, and worker count, with the relationships being generally weak to moderate. The prediction model achieved a positive but low (R2) result, but failed to see a reduction in MAE compared to the median baseline, suggesting low run-time predictability from the available features. These results led to the development of a conceptual AI augmented approach, which brings together workload monitoring, feature extraction, run-time prediction, resource planning and decision support. The study concludes that workload-trace analytics can provide insights for the future capacity planning of GPUs and workload-aware scheduling, but reliable validation requires richer queue, memory, utilization, energy and hardware telemetry data.

Keywords: AI-Augmented GPU Scheduling, Machine Learning Inference, GPU Workload Analytics, Runtime Prediction, Resource Planning, Large-Scale GPU Clusters.

1. Introduction

The Graphics Processing Unit (GPU) has emerged as an indispensable tool for training and deployment of modern machine learning applications (Dally et al., 2021). Their parallel processing allows them to quickly process computationally intensive tasks such as deep learning, computer vision, recommendation systems and large-scale inference services. As more organizations roll out machine learning models in production, the expectations for the GPU cluster to operate a massive volume of diverse jobs that have different run-time, resource needs, worker configurations, and operational priorities increase (Jeon et al., 2018). Effective handling of such workloads is therefore important to maintain predictable throughput, control completion timelines, and prevent over-provisioning of resources.

However, in large-scale inference systems, the problem of workload management is very hard because the computational requirements and workload distribution are not uniform (Liang et al., 2024). Production tasks can consist of numerous small jobs and only a few long-running and resource-intensive jobs (Mahajan et al., 2020). The rate of submission is also very diverse depending on the case, resulting in increased "concurrent demand". In addition, different GPU models can be used for different workloads based on specific application needs, available resources, or organizational policies (Rosenfeld et al., 2022; Sun et al., 2019). These traits make it difficult to use resource-management techniques that rely solely on predetermined rules or on an average workload.

Dynamic GPU scheduling aims to address this variation by taking into account the current workload demand and available computational resources during scheduling decisions (X. Wang et al., 2023). In this evolution, newer

Sasank Varma Sarpella
University of Alabama at Birmingham, USA.
Email: varma.shashank0303@gmail.com

Received: 23-July-2026 Revised: 20-Aug-2026 Accepted: 25-Aug-2026

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

approaches have started using machine learning and predictive analytics to estimate the duration of workloads, predict demand and assist with resource allocation (Kamble et al., 2023). AI-driven scheduling can thus offer a way to combine insights from past workload performance with current resource data (Khakifirooz et al., 2026). However, the usefulness of predictive scheduling relies on the accuracy and comprehensiveness of the available workload attributes (Pasham, 2018). Production traces may not capture run-time, queue state, memory requirements, hardware utilization, or application-level information, potentially limiting the accuracy of predictions and resulting conclusions of scheduling performance (Lu et al., 2026).

Previous works have often focused on designing and evaluating full algorithms based on simulations, simulated experiments or cluster testbeds (Amaran et al., 2016). While such assessments are crucial, there has been relatively little focus on building an empirical, trace-based basis for AI-based resource planning. In order to make reliable decisions in a scheduling framework, it is crucial to understand workload distributions, patterns of resource requests, temporal demand fluctuation, the differences between various workload categories, and the limits to which the actual job run-time can be predicted from the available metadata (Khoshkbarforoushha et al., 2016). If these characteristics are not investigated, then scheduling models may be developed under assumed conditions and not be adapted to the real production situation.

In this work, we meet this demand by analyzing a large workload trace of 466,867 machine learning jobs received by 84 organizations over a publicly accessible GPU workload trace. The trace contains CPU and GPU requests, the number of workers, submission time, job time, GPU-model categories, and job types (Liu et al., 2025). The study includes data quality, workload description characteristics, workload distribution at run-time, resource–run-time association, difference between job classes, GPU models, temporal workload dynamics, concurrent requested workload, and chronological run-time prediction.

This study presents an AI-enhanced, dynamic GPU scheduling framework consisting of four main components: workload monitoring, feature extraction, run-time prediction, resource planning, and decision support. The framework is based on concepts and analysis, and is not an experimentally supported scheduling algorithm itself. Rather, it shows how trace-derived workload information and predictive modelling can be combined in a way that can aid future decisions for dynamic resource allocation. It is crucial to note that this distinction is important as the available dataset does not contain any information about the scheduling assignment, queue waiting times, GPU physical utilization, memory consumption, or energy measurements.

The goal is to: (1) validate and characterize the large scale GPU workload trace; (2) investigate the distribution of job run-time and requested computational resources; (3) explore associations between requested resource and run-time; (4) examine run-time differences between the various GPU models and job types; (5) analyze temporal arrivals, inferred completions, workload concurrency and requested GPU demand; and (6) evaluate the feasibility of AI-based run-time prediction using a chronological evaluation design.

The study makes three contributions toward the study of resource management in GPUs. First, based on a large production trace, it gives an empirical characterization of the workload heterogeneity and temporal demand that is repeatable and reproducible. Second, it integrates non-parametric statistical analysis, requested capacity-time estimation and chronologically predicting run-times in a single analytical procedure. Third, it maps the empirical results to a conceptual AI-augmented framework which can guide future workload-aware scheduling, admission control, capacity planning, and GPU allocation. The results confirm the potential and feasibility of workload metadata to aid intelligent resource management on GPUs for dynamic machine learning scenarios.

2. Literature Review

2.1 GPU Workload Management in Large-Scale ML Systems

The widespread adoption of machine learning services has driven the need for heterogeneous GPU clusters that can handle a variety of resource-heavy workloads (Zhang et al., 2025). The significant variation in job durations, resource requirements, user behavior, and GPU utilization makes capacity planning and workload management difficult (Hu et al., 2021). Previous workload characterization research has focused on the importance of understanding such patterns for better cluster design and resource management (Cortez et al., 2017).

However, one of the difficulties is that aggregate averages may mask the bursty and heavy-tailed production workloads (Sajal et al., 2021). A significant amount of short jobs can run concurrently with a handful of long-running

jobs that consume most of the requested capacity-time and concurrency. It follows that, to manage the workloads on GPUs effectively, one must consider the distributions of workloads in run-time, their temporal arrival, completion pattern, and demand for resources, as well as mean workload parameters (Dev & Reda, 2016; Li et al., 2022).

2.2 Dynamic and Heterogeneity-Aware GPU Scheduling

Prior GPU scheduling studies have been mainly oriented towards optimizing job completion time, fairness, throughput, and resource utilization (Chab et al., 2025). To enhance scheduling efficiency, the workload-specific information is used in systems like Gandiva, and the performance heterogeneity of GPUs is taken into consideration in allocation decisions in Gavel (Mahajan et al., 2020; Sultana et al., 2026). Recent methods take adaptive resource sharing, prediction-based scheduling, and performance variability into account for allocating workloads to GPU resources.

Although these advances have been made, the usefulness of dynamic scheduling is reliant on the information that can be gathered at the time of the decision (Renke et al., 2021; Yang et al., 2023). Heterogeneous workloads can vary in length, resource needs for GPUs, number of workers, and criticality of the workloads, and heterogeneous GPUs can be different in the applications that can run on them (Kaur et al., 2025). This results in the demand for some scheduling models that include workload monitoring, predictive workload estimates, and temporal demand information in addition to fixed allocation rules.

2.3 AI-Based Run-time Prediction and Workload Analytics

Predicting run-times can be helpful for admission control, capacity provisioning and workload prioritization (Lu et al., 2026). The accuracy of predictions, however, is often limited by workload heterogeneity, incomplete metadata, and limited application- or hardware-level information (Dinçer & Kilimci, 2026). Tree-based ensemble models are often used due to their ability to model the nonlinear relationships between resource requests, workload types and the behaviour of execution (Moolchandani et al., 2022). However, highly skewed run-time, extreme outliers, or time distribution shifts can decrease the predictive accuracy. Comparisons of the chronological validation and baseline information are therefore important to establish the practicability of using a predictive model beyond merely summarizing historical data (Kyriazi & Thomakos, 2026).

2.4 Research Gap

The majority of the current literature considers complete scheduling systems by using simulations or controlled GPU testbeds. Less focus is placed on developing a validated trace-based analytical foundation that comprehensively considers the distribution of run-time, requested capacity-time, temporal demand, workload-category differences, and chronological run-time predictability. To fill this gap, the present study will rely on large-scale workload evidence to guide a conceptual AI-assisted GPU scheduling framework, without making unsupported statements regarding the experimentally demonstrated performance of the scheduler.

Based on this study, the development of a scheduler is considered to be a prerequisite for workload analytics (Kashyap & Singh, 2023). It uses data validation, non-parametric statistical analysis, temporal workload modelling and run-time prediction to isolate the usable signals and the constraints of the available trace attributes. This gives a more realistic foundation for designing AI-driven resource-planning and scheduling systems, which can be tested later in controlled experiments with more detailed infrastructure telemetry.

3. Research Methodology

3.1 Research Design

The research design used in this study was quantitative, retrospective and trace-based to analyze workload run-time, requested resource demand, temporal dynamics and run-time predictability in large-scale machine learning inference environments. The study was observational in nature since the workload characteristics were only analyzed, not altered, replayed or experimented with different scheduling policies. Hence, the research wasn't about the performance of causal schedulers, but rather aimed to provide empirical evidence to support the design of an AI-augmented GPU resource-planning framework.

3.2 Overall Research Workflow

The research process consisted of six steps as depicted in Figure 1. The first step was to validate the workload trace for missing values, duplication and invalid resource and timing values. Second, descriptive and distributional analysis were performed to describe the composition of workload, run-time skewness, and the

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

patterns of resources which are requested. Third, derived measures such as completion time and requested GPU- and CPU-time were calculated. Fourth, non-parametric tests were used to analyze resource–run-time associations and differences in run-time between GPUs and job types. Fifth, a temporal workload analysis was conducted to estimate job arrivals, assumed completions, workload during the same time period, and resource demand. Lastly, a machine learning model was proposed and created to assess the feasibility of run-time prediction with a chronological train-test split.

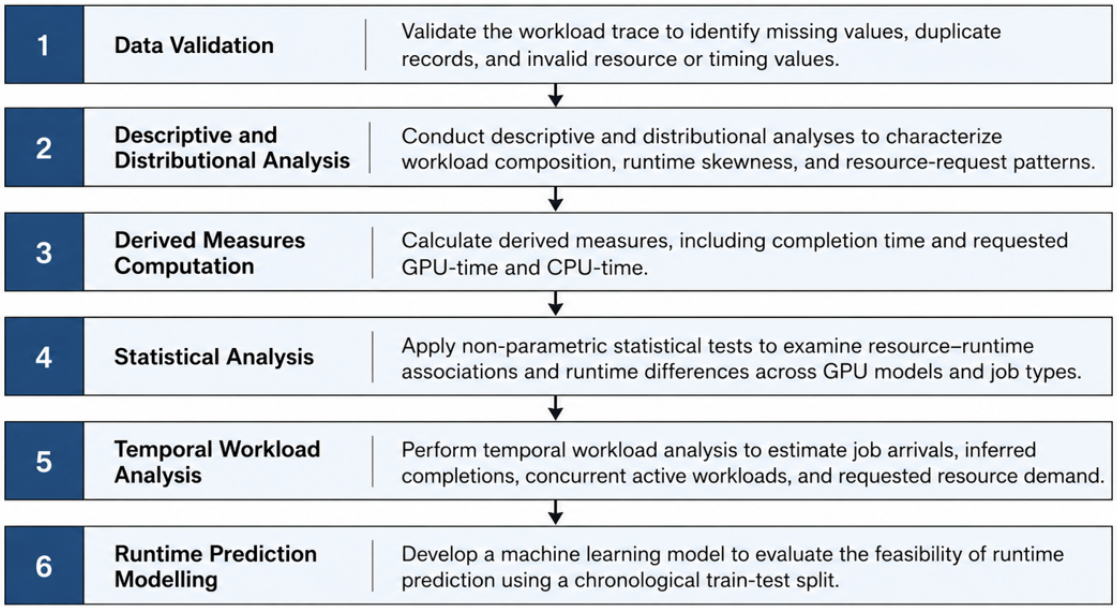

Figure 1. Overall research workflow of the study.

3.3 Research Process

The proposed AI-augmented dynamic GPU scheduling framework was conceptually developed grounded on the analytical results. Based on the observed workload characteristics, workload monitoring, feature extraction, run-time prediction, resource planning, and decision-support components were defined. The detailed description of dataset preparation, statistical procedures, predictive modelling and implementation settings can be found in Section 5, and the corresponding empirical findings in Section 6.

4. Proposed AI-Augmented Dynamic GPU Scheduling Framework

4.1 Framework Overview

The workload characteristics analyzed in this study suggest that an intelligent resource planning framework in the large-scale machine learning inference environment is proposed by combining AI and dynamic scheduling of GPUs. The framework is not an experimentally-proven scheduling algorithm, but is intended as a conceptual decision-support architecture. It includes workload monitoring, feature extraction, AI-based workload prediction, workload resource consumption analysis, and scheduling decision support to help manage GPU resources under dynamic workloads, as illustrated in Figure 2.

The framework is driven by the empirical results of this study, such as highly skewed run-times, highly variable workload arrival patterns over time, varying resource-demand patterns and the inability of resource-request information alone to predict run-times. Results display the need to incorporate workload analysis and predictive intelligence into future scheduling systems for optimal resource allocation.

4.2 Workload Monitoring

The first layer continuously observes the submission of workloads and the collection of operational data from the submitted inference jobs. Factors of the workload include: submission time, GPU model, CPU request, GPU request, worker count, organization, and job type. These variables are used as a preliminary indication of the amount of computation required prior to scheduling. In addition to individual workload characteristics, the monitoring layer

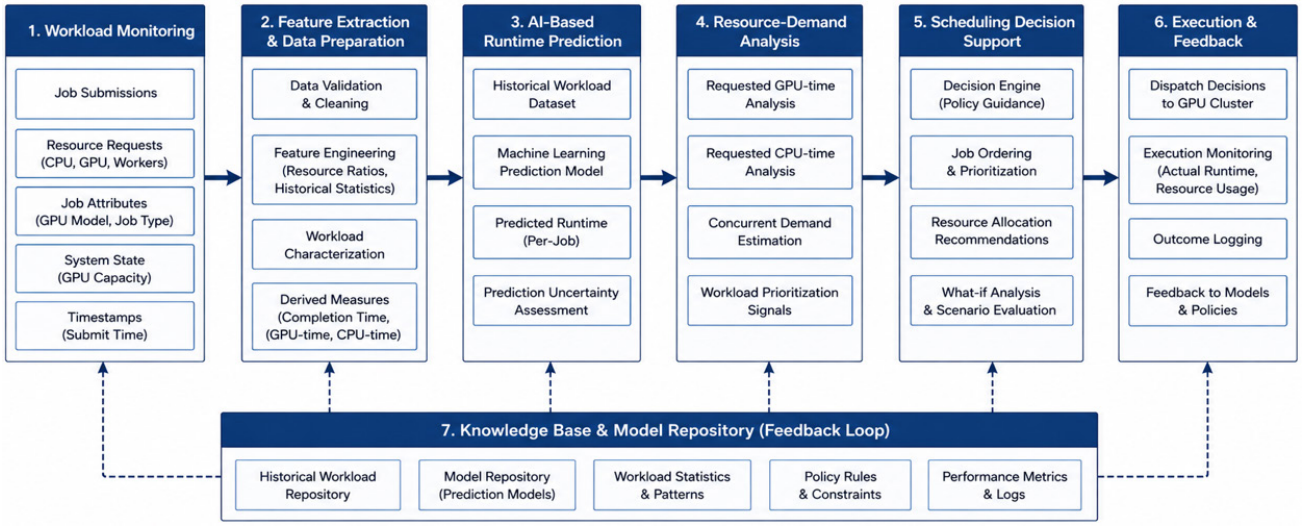

Figure 2. Proposed AI-augmented dynamic GPU scheduling framework.

also keeps aggregated workload statistics like the workload submission rate, the workload completion, the number of concurrent workloads that are running, and the number of GPUs that are requested by the workloads. The framework can utilize continuous monitoring to identify variations in workload and determine when the computation is heavy, hence requiring adaptive resource-allocation policies.

4.3 Feature Extraction

The feature-extraction layer converts unstructured workload data into structured data for the prediction and resource planning logic. It combines some original parameters such as GPU model, CPU request, GPU request, worker count, job type, organization and submission time with some derived parameters such as job completion, requested capacity-time, temporal demand and workload concurrency. These features provide a compact representation of a workload property and the entire GPU ecosystem. The aggregated demand measures are then provided to the AI prediction module, and runtime-related and temporal indicators to resource-planning and decision-support functions. These mathematical definitions and computational methods will be detailed in Section 5.3.

4.4 AI Prediction Module

The AI prediction module predicts workload run-time on the basis of historical workload data and the current resource requests (Duc et al., 2025). A HistGradientBoostingRegressor was used in this study because of its capability to handle large-scale datasets and capture nonlinear relationships (Bridge-Nduwimana et al., 2025). The predictor variables are: organization, GPU model, job type, CPU request, GPU request, worker count, and temporal variables based on submission time.

The prediction target is the log-transformed run-time,

$$Y_i = \log(1+Di) \quad (1)$$

where

Y_i denotes the transformed target variable and

D_i represents the run-time (duration) of job i

This decreases the impact of super-long-running jobs on the model and enhances its stability. The estimated run-time offers insights into the behavior of workloads that can be considered in future scheduling and resource planning decisions.

4.5 Resource Planning

The resource-planning layer integrates workload data from monitoring with predictions of how long tasks will take to run to estimate future demands (Duc et al., 2025). The framework leverages workload characteristics, requested GPU demand, requested CPU demand, and temporal workload trends to produce workload-aware estimates which can be used for capacity planning. In contrast, the proposed framework accounts for the fact that workloads may be heterogeneous and temporal demand patterns may occur to make better planning decisions during periods of variable computational load. This enables infrastructure managers to forecast future demand and allocate the resources of the available GPUs to satisfy this demand.

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

4.6 Decision Support

The final layer provides decision support for scheduling, using workload monitoring, derived workload properties, AI-based run-time estimation, and analysis of resource requirements. The framework is meant to be used alongside the current scheduling policies, and it offers predictive information that can support workload scheduling, admission control and GPU resource allocation. The analytical components of this framework, namely workload characterization, statistical analysis, temporal demand modeling, and run-time prediction, are assessed in this paper. However, experimental implementation and comparison with the existing GPU scheduling algorithms can still be considered as future directions.

5. Experimental Setup

5.1 Dataset Description

The assessment was performed through an experiment on a public GPU workload trace comprising 466,867 machine learning tasks submitted by 84 organizations. Every record corresponds to an individual workload event and contains nine attributes that influence the computational resource requests and execution properties of the events: job identifier, organization, GPU model, CPU request, GPU request, worker count, submission time, job duration, and job type. There are six categories of GPU models (A10, A100-SXM4-80GB, A800-SXM4-80GB, GPU-series-1, GPU-series-2 and H800) and two job classes (HP and Spot). The analysis was based on characteristics of the workloads, requested resource demand, run-time behaviour and temporal workload dynamics because the data provided did not offer a measurement of hardware utilization and no information on queue waiting times, scheduling decisions, memory usage, energy usage, or physical GPU usage was available. Data source and usage statement. The workload trace evaluated in this paper was provided by the public Alibaba GPU Cluster Spot Resource dataset on Kaggle. The data set includes metadata for machine learning tasks such as workload level, resource requests, type of GPU models, time when the task was submitted, duration of the task, and type of task. The data is available on Kaggle and the corresponding GitHub repository with the MIT License. The data set was used solely for academic research and analysis, and an effort was not made to identify any individual users or organizations beyond the information contained in the publicly available trace (Ovi).

5.2 Data Validation and Preprocessing

The data was standardized and validated before analysis to make it consistent for analysis. The column names were standardized by stripping unnecessary spaces, converting text to lowercase, and replacing spaces with underscores. Numerical data were converted to appropriate numerical formats, and categorical data were cleansed by stripping leading and trailing spaces.

Data validation covered for missing values, duplicate records, duplicate job identifiers, negative submission times, invalid CPU requests, non-positive GPU requests, non-positive worker counts and non-positive job durations. All 466,867 records were validated and included for analysis. The job identifier was used only as a unique identifier and not included in any statistical modelling; there were no categorical variables coded other than for use in machine learning.

5.3 Derived Workload Measures

Multiple workload measures were developed for temporal and resource demand analysis. Job completion time was defined as

$$CT_i = ST_i + D_i \quad (2)$$

where:

CT_i = completion time of job i
ST_i = submission time of job i
D_i = run-time (duration) of job i

Requested GPU-time was estimated as:

$$RGT_i = G_i \times D_i \quad (3)$$

where:

RGT_i = requested GPU-time of job i
G_i = requested number of GPUs for job i
D_i = run-time (duration) of job i

Similarly, the requested CPU time was computed as

$$RCT_i = C_i \times D_i \quad (4)$$

where:

RCT_i = requested CPU-time of job i
C_i = requested CPU resources for job i
D_i = run-time (duration) of job i

These derived variables are not meant to be interpreted as GPU utilization or computational efficiency, but rather as a request for computational resources during the execution period. For exploratory analysis, worker-scaled demand measures were also computed since the resource requests are not reported per worker or per workload.

5.4 Statistical Analysis

First, descriptive statistics were computed for the

characteristics of workloads (mean, standard deviation, median, quartiles, minimum, maximum and selected percentiles) of CPU request, GPU request, worker count, job duration, GPU-time requested, and CPU-time requested. Frequency distributions were also created for job types and the categories of GPUs.

Skewness, excess kurtosis and D'Agostino's (K2) normality test were used to investigate the distribution of job duration. The test was also conducted on a random sample of 5,000 observations, which was selected in a reproducible manner; normality tests are too sensitive for a very large data set.

Spearman's rank correlation was used to explore the relationships between workload characteristics; percentile bootstrap confidence intervals were constructed from 300 bootstrap replications using reproducible subsets of 20,000 observations. The Kruskal–Wallis test was used as an analysis for run-time difference in the category of GPU models, using epsilon-squared (E^2) as an effect size measure. Pairwise Mann–Whitney U tests with Holm correction were performed when significant differences were found. The Mann–Whitney U test was also used to compare the run-time of the HP and the Spot jobs and rank-biserial correlation as an effect size.

Consecutive windows of 3,600 units of trace-time were used to analyze the temporal workload behaviour. If a workload was considered active, then it was:

$$ST_i \leq t < CT_i \quad (5)$$

where:

ST_i = submission time of job i
CT_i = completion time of job i
t = observation time

The event-sweep algorithm was used to estimate the number of concurrent active jobs. Hence, the demand for GPUs, CPUs, and the throughput of jobs completed during the trace.

5.5 Run-time Prediction Model

In order to explore the potential of AI-assisted workload planning, an AI model based on machine learning was created to forecast the run-time of jobs. The target variable was transformed using a natural logarithm as the distribution was highly right-skewed at run time.

To maintain the temporal order of workloads arriving at the train and prevent information leakage, an 80/20 split by chronology was used. The model used is a HistGradientBoostingRegressor, which was chosen for its computational efficiency and its ability to handle large-scale datasets and model nonlinear relationships. The Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Median Absolute Error (MedAE) and the coefficient of determination (R2) were used to compare the model's performance with a training-median baseline. Values predicted were recalculated on the original run-time scale.

5.6 Implementation Details

All analyses were done in the Python programming language with the use of Pandas, NumPy, SciPy, Scikit-learn, Matplotlib and Statsmodels. To ensure the reproducibility of the statistical analyses and machine learning experiments, fixed random seeds were used throughout. Data validation, feature generation, statistical testing and prediction modelling were all performed programmatically from the workload trace. All results reported were taken directly from available variables in the dataset, to ensure transparency and reproducibility of the results without jumping to conclusions about the performance of the scheduler or the physical utilization of the GPU.

6. Results and Performance Evaluation

6.1 Dataset Quality and Workload Characteristics

A total of 466,867 valid machine learning jobs were submitted by 84 organizations representing six GPU-model categories and two types of jobs. Data validation showed that there were no missing data values, duplicated data records, duplicated job identifiers, negative submission durations, invalid CPU requests, non-positive GPU requests, non-positive number of workers, or non-positive durations. As a result, all records were kept for analysis. The trace ran from 0 to 15,902,470 trace-time units, and the latest inferred completion time was 15,903,908. A summary of these features is given in Table 1.

6.2 Descriptive Workload Statistics

The descriptive results show that there is a great amount of heterogeneity in the run-time of the workloads and in the resources requested, as shown in Table 2. CPU requests averaged 24.54 ± 41.00, whereas GPU requests averaged 2.02 ± 2.42. The median GPU request was 1, with the 1st and 3rd quartiles also being 1, showing that the distribution was very peaked, with a high proportion of jobs having only one GPU. The number of workers was similar: the median was 1, though the top was 332, showing that there were a few large distributed workloads.

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

Table 1. Dataset characteristics and validation

Measure	Value
Total jobs	466,867
Valid unique jobs	466,867
Organizations	84
GPU models	6
Job types	2
Missing values	0
Duplicated records	0
Duplicate job identifiers	0
Invalid CPU requests	0
Non-positive GPU requests	0
Non-positive worker counts	0
Negative submission times	0
Non-positive durations	0
Trace start	0
Latest submission time	15,902,470
Latest inferred completion time	15,903,908

The length of time in employment varied considerably. The average duration was 185,391.64, but the median was just 1,216, which is a very right-skewed distribution. The duration varied from 1 to 15,897,599 trace-time units. The two other metrics, requested GPU-time and requested CPU-time, also displayed significant spread, indicating the presence of short, less demanding jobs and a small number of long-running, resource-heavy jobs. The distribution of run-time is strongly right-skewed, as shown in Figure 3.

6.3 Workload Distribution and Resource Associations

Table 3(a) presents the workload composition, normality results, and correlation estimates. The A10 category represented 48.97% of all jobs, followed by A100-SXM4-80GB with 36.20%. These two types of GPU combined made up over 85% of the workload trace. 89.04% of all jobs were HP workloads, while 10.96% were Spot workloads, as shown in Table 3(b).

The distribution of run-time was highly skewed. The skewness of raw duration was 9.33, and the excess kurtosis was 108.31. Logarithmic transformation was used to minimize skewness to 0.81 for both raw and transformed duration, but D'Agostino's K2 test was still significant ($p<0.001$) as shown in Table 3(c).

Spearman correlation analysis revealed there was a weak to moderate correlation between CPU request and duration ($\rho=0.306$). GPU request ($\rho=0.195$) and worker count ($\rho=0.140$) showed weak positive correlations with duration. Among all the correlations, the highest was found between CPU and GPU requests ($\rho=0.633$), meaning that the

Table 2. Descriptive workload statistics

Variable	Mean	SD	Median	Q1	Q3	Maximum
CPU request	24.54	41.00	8.00	4.00	15.00	192.00
GPU request	2.02	2.42	1.00	1.00	1.00	8.00
Worker count	1.69	5.57	1.00	1.00	1.00	332.00
Duration	185,391.64	1,061,666.55	1,216.00	240.00	4,584.50	15,897,599.00
Requested GPU-time	231,038.36	1,438,460.24	1,613.00	226.00	8,141.50	127,180,792.00
Requested CPU-time	5,812,120.30	54,052,915.09	11,792.00	1,252.00	94,080.00	3,052,339,008.00

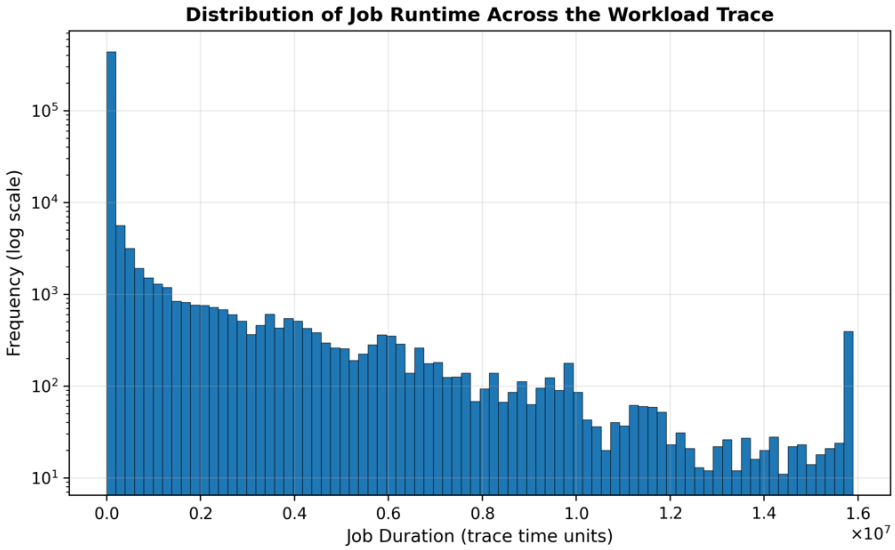

Figure 3. Distribution of job run-time across the workload trace

requests for both resources were more likely to be high for more resource-intensive workloads, as illustrated in Table 3(d). The duration was also positively related to worker-scaled GPU request (ρ=0.228). While all relationships were statistically significant, their magnitudes suggest that resource requests have weak associations with run-time and would not be expected to offer enough information to predict it.

Table 3. Workload distribution and statistical analysis

3(a). GPU-model distribution

GPU model	Jobs	Percentage
A10	228,643	48.97%
A100-SXM4-80GB	169,020	36.20%
GPU-series-2	30,397	6.51%
GPU-series-1	18,911	4.05%
H800	11,097	2.38%
A800-SXM4-80GB	8,799	1.88%

3(b). Job-type distribution

Job type	Jobs	Percentage
HP	415,713	89.04%
Spot	51,154	10.96%

3(c). Run-time normality

Variable	Skewness	Excess kurtosis	K^2	p-value
Duration	9.328	108.307	7484.252	<0.001
log1p(Duration)	0.810	0.976	521.935	<0.001

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

3(d). Spearman correlations

Variables	ρ	95% CI	p-value
CPU request vs Duration	0.306	0.295–0.318	<0.001
GPU request vs Duration	0.195	0.188–0.212	<0.001
Worker count vs Duration	0.140	0.133–0.155	<0.001
CPU request vs GPU request	0.633	0.623–0.641	<0.001
Worker-scaled GPU request vs Duration	0.228	0.215–0.238	<0.001

6.4 Run-time Differences Across GPU Models and Job Types

The Kruskal–Wallis test showed significant differences in run-times between the six GPU models:
H (5) = 67,541.15,p<0.001,ε^2= 0.145
A10 workloads had the lowest median run-time (420), and GPU-series-1 workloads had the highest median run-time (4,553). Intermediate medians were observed for H800 (1,039), A800-SXM4-80GB (1,403), GPU-series-2 (1,582), and A100-SXM4-80GB (1,854) as summarized in Table 4(a).
These differences are shown in a logarithmic boxplot in Figure 4, which illustrates the spread and median run-time of the distributions for different GPU models. Some of the outlier markers were removed for clarity. As it is an observation trace, the results do not reflect a difference in the inherent speed of the GPU models, but rather in the workload composition and workload allocation.
The run-time was also statistically correlated with job type. The median duration for HP workloads was 1,101, while for spot workloads it was 2,095.5. In Table 4(b), a Mann–Whitney U test showed that this difference was significant:
U=8,746,743,822,p<0.001

The rank-biserial correlation was −0.177. This suggests that the run-times for Spot workloads were typically longer, but it was a relatively small effect.

Table 4. Run-time comparison across workload categories

4(a). GPU-model median run-time

GPU model	Jobs	Median duration
A10	228,643	420.0
H800	11,097	1,039.0
A800-SXM4-80GB	8,799	1,403.0
GPU-series-2	30,397	1,582.0
A100-SXM4-80GB	169,020	1,854.0
GPU-series-1	18,911	4,553.0

4(b). Statistical tests

Test	Statistic	p-value	Effect size
Kruskal–Wallis	H=67,541.15 (df=5)	<0.001	ε^2=0.145
Mann–Whitney (HP vs Spot)	U=8,746,743,822	<0.001	Rank-biserial=-0.177

6.5 Temporal Workload Dynamics and Prediction Performance

The summary of the temporal workload characteristics and runtime-prediction results is shown in Table 5(a). The trace was broken down into 4,418 consecutive windows of 3,600 trace-time units. The average number of jobs submitted and completed per window was 105.67. The workload intensity, however, was very different, with a raw of 3,740 submitted jobs and 4,668 completed jobs in individual windows.
The mean number of jobs active at the same time was 5,438.45, and the maximum number was 6,676. The requested GPU demand for concurrency had an average of 6,773.28 and a max of 8,247.47, indicating significant short-term fluctuations in the number of requested GPU demands.

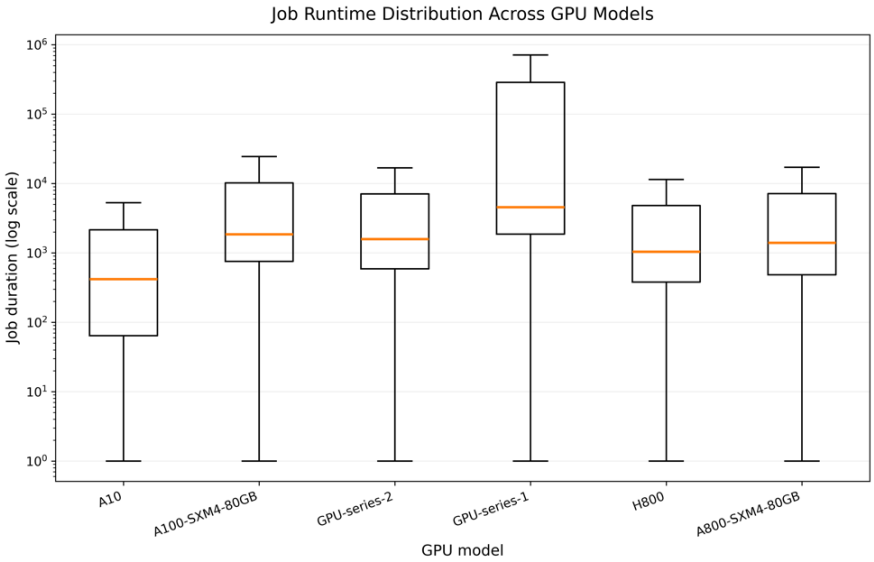

Figure 4. Job run-time distribution across GPU models

Figure 5 displays temporal trends of the submitted and inferred completed jobs, smoothed with 3,600 consecutive units. The series exhibits segments of fairly consistent activity, broken up by periods of distinct workload 'bursts', and completion activity appears to fluctuate with submission activity.

The requested capacity-time analysis revealed that A10 workloads consumed 48.74% of the total requested GPU-time, followed by A100-SXM4-80GB with 29.03% of the total. The total requested GPU time was dominated by HP jobs, with 98.74% of the GPU time.

The model was trained on 373,493 chronologically earlier jobs and evaluated on 93,374 subsequent jobs. HistGradientBoosting decreased RMSE from 471,979.81 to 446,090.77 with R2=0.067 when compared to the baseline of the median of the training set. However, MAE increased slightly from 97,324.50 to 97,376.37 as illustrated in Table 5(b). The model was thus able to capture a small amount of the general variation in run time, but it did not always perform better than the baseline on the reported metrics.

Table 5. Temporal workload analysis and run-time prediction

5(a). Temporal workload summary

Measure	**Mean**	**Peak**
Submitted jobs/window	105.67	3,740
Completed jobs/window	105.67	4,668
Active jobs	5,438.45	6,676
Concurrent GPU request	6,773.28	8,247.47

5(b). Run-time prediction

Model	**MAE**	**RMSE**	**R^2**
Training-median baseline	97,324.50	471,979.81	-0.044
HistGradientBoosting	97,376.37	446,090.77	0.067

7. Discussion

The study analyzes the run-time behavior, resource demand, temporal workload dynamics and run-time predictability of a large-scale GPU workload trace. Four main findings were obtained. First, the run-time was highly right-skewed, with a significant number of short jobs and a relatively small number of long-running jobs. Second, the relationships between requested CPU and GPU resources and duration were generally weak to moderate, but were positive. Third, the distribution of run-times

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

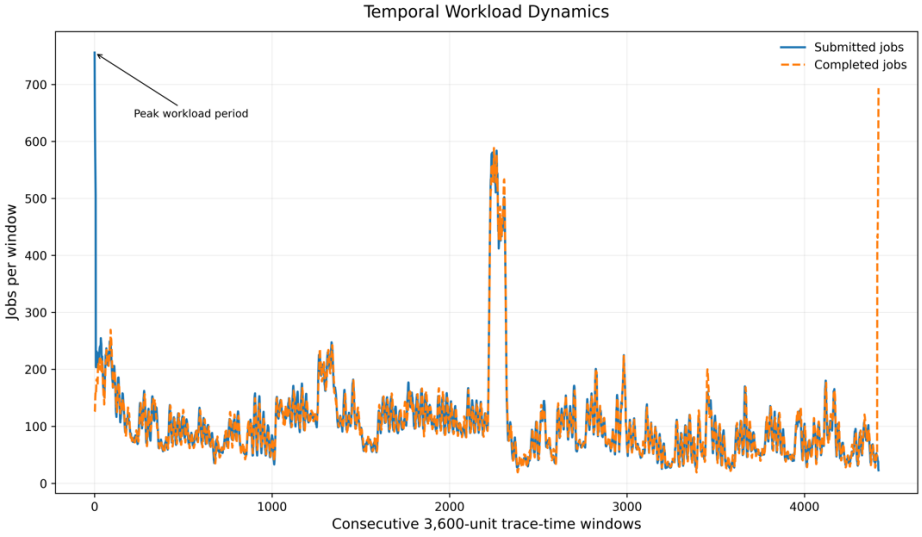

Figure 5. Temporal workload dynamics

across GPU models and types varied widely, indicating heterogeneous distributions of workload composition. Finally, the temporal analysis showed significant workload variation in arrivals, inferred completions and concurrent requested demands, and the prediction model showed slight predictability of run-time. These results collectively offer empirical support for future approaches to resource planning on GPUs that rely on AI.

The study is empirical in nature and uses a large-scale production workload trace, and thus it focuses on the practical performance of the GPU scheduling algorithms, rather than simulation or cluster tests (Kaur et al., 2025). The adopted approach (data validation, descriptive workload profiling, non-parametric statistical analysis, temporal workload modelling and run-time predictions) gives a data-driven basis for designing and/or evaluating a scheduling policy in the presence of AI support. As the workloads are heterogeneous, the demands can be very different from one time to the next, and there is not much predictability in the run-time of the workloads (Lu et al., 2026). This indicates that scheduling using AI systems requires consideration of real workload properties, rather than only static workload heuristics or simplified workload benchmarks. Hence, the proposed work extends the ongoing work on scheduling as it provides an evidence-based foundation for possible workload-aware scheduling, capacity planning and resource allocation strategies.

7.1 Implications for AI-Augmented GPU Resource Planning

The identified heterogeneity is proof that a mean of resource demands cannot be represented for large-scale GPU environments. The workloads ranged significantly in terms of duration, resource demands, GPU-model allocation and intensity. Peak GPU usage and associated submissions indicate that static allocation techniques might not provide flexibility in rapidly changing workload situations (Tsenos & Kalogeraki, 2025). Positive resource run-time associations indicate that attributes for particular requests are useful cues of the complexity of the workload. But their small size and lack of predictive power indicate that these attributes, if considered individually, are not powerful enough. An AI-enhanced planning framework thus needs to be able to merge the desired resources with more information, such as workload history, application characteristics, queue state, model architecture, memory demand, and hardware telemetry. Based on the current analysis, the following indicators of workload and time may be useful in guiding future decisions about resource allocation.

7.2 Run-time Variability and Requested Resource Demand

The mean and median duration are significantly apart, indicating heavy-tailed workload behaviour. While most jobs were of relatively short duration, there were a few with noticeably longer durations and correspondingly high values of requested GPU-time and CPU-time. This variation makes it difficult to forecast capacity, since the mean run-time is sensitive to extreme values. The requested capacity-time was different between GPU types and workload classes. However, these specifications are

not a true measure of the resources consumed during computation. These should therefore not be used as a direct measure of GPU occupancy, scheduling efficiency, energy consumption or hardware productivity. Requests for resources may be very different from the actual resources used for execution.

7.3 GPU-Model and Job-Type Differences

The run-times varied significantly, depending on the type of GPU model, and are not meant to provide any comparative hardware performance. There is no indication that the same workloads were run on all GPU models under controlled conditions. The workloads assigned to different GPU categories might have application-specific requirements, complexity, or priorities for operations (Y. Wang et al., 2023). Likewise, Spot jobs had longer median run-times than HP jobs, which could be due to the composition of jobs or deployment factors, or simply due to the nature of the job and/or scheduling policy. The results are thus likely to reflect differences in workload distributions between their observations and not the effects of their hardware configurations.

7.4 Run-time Prediction Performance

The HistGradientBoosting model performed better than the median baseline in terms of the RMSE, and the R^2 was positive but slightly higher than the baseline in terms of MAE. This suggests that the model captured some variation in run-time, but it did not consistently outperform in terms of absolute prediction error. Thus, the prediction outputs are not compelling enough to support run-time prediction for scheduling decisions based on available workload metadata only. The model must therefore be viewed as a feasibility test and not as a proven part of the production scheduling prediction system.

7.5 Practical Implications

The results apply to machine learning platform users, GPU-cluster operators, and cloud providers. Workload profiling can assist in demand forecasting and capacity planning, and temporal analysis can help identify times when there is increased computational load. Furthermore, run-time information and requested-capacity information can help with admission control, workload prioritization, and better decision-making in resource-allocation. But the results should not be interpreted as evidence that the scheduler is performing better.

7.6 Limitations and Future Research

The workload used in this study was obtained from a single observation, which could be a drawback to generalizing to other GPU architectures. The information provided in the dataset did not contain data on queue waiting times, scheduling, GPU memory demands, physical utilization, energy usage, service-level goals, or hardware performance counters. The study, therefore, measured run-time and the resources used, but not actual latency, utilization, or scheduling efficiency.

Several of the aspects mentioned need to be combined in future works: scheduler logs, queue information, workload semantics, hardware telemetry, memory usage, and energy measurements. The proposed AI-guided framework must also be tested experimentally with known scheduling policies using metrics such as latency, throughput, fairness, utilization and energy. Further enhanced generalizability and practicality would be provided through multi-trace validation and controlled workload replay.

8. Conclusion

A large-scale GPU workload trace of 466,867 machine learning jobs was analyzed to investigate run-time behaviour, requested resource demand, temporal dynamics and run-time predictability. The results demonstrated that the workload was very heterogeneous, the run-times were very skewed, the GPU models were used in a very skewed manner, and the concurrent computational demands were quite variable. The CPU and GPU resources requested were mostly weak to moderate positive associations with run-time. The execution time also varied greatly between GPU models and job types, but this is more due to the composition of the workload. Finally, temporal analysis revealed that workload arrivals, inferred completions, and concurrent GPU demand were also quite dynamic. The HistGradientBoosting model achieved a lower RMSE and a positive (R2), which was not as effective as the median baseline on MAE. This means that the characteristics of the workload available have limited run-time predictability. Based on these findings, this study proposed a concept of an AI-based, adaptive GPU scheduling system that combines workload monitoring, feature extraction, run-time prediction, resource planning and decision support. The proposed framework should be treated as a conceptual basis for further research on workload-aware scheduling of GPUs instead of a tested scheduling algorithm. Experimental implementation and testing are needed to assess its impact on latency, throughput, resource

AI and Machine Learning Advances
Volume 2, Issue 2
ISSN: 3067-3216

utilization, fairness, and energy-efficiency when applied to existing scheduling policies.

Reference

Amaran, S., Sahinidis, N. V., Sharda, B., & Bury, S. J. (2016). Simulation optimization: a review of algorithms and applications. Annals of Operations Research, 240(1), 351-380.

Bridge-Nduwimana, C. B., El Ouaazizi, A., & Benyakhlef, M. (2025). An Integrated Intuitionistic Fuzzy-Clustering Approach for Missing Data Imputation. Computers, 14(8), 325.

Chab, R., Li, F., & Setia, S. (2025). Algorithmic techniques for GPU scheduling: a comprehensive survey. Algorithms, 18(7), 385.

Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., & Bianchini, R. (2017). Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. Proceedings of the 26th Symposium on Operating Systems Principles,

Dally, W. J., Keckler, S. W., & Kirk, D. B. (2021). Evolution of the graphics processing unit (GPU). IEEE Micro, 41(6), 42-51.

Dev, K., & Reda, S. (2016). Scheduling challenges and opportunities in integrated cpu+ gpu processors. Proceedings of the 14th ACM/IEEE Symposium on Embedded Systems for Real-Time Multimedia,

Dinçer, E., & Kilimci, Z. H. (2026). Real-time and offline large language models on edge devices: a systematic review. PeerJ Computer Science, 12, e3769.

Duc, T. L., Nguyen, C., & Östberg, P.-O. (2025). Workload prediction for proactive resource allocation in large-scale cloud-edge applications. Electronics, 14(16), 3333.

Hu, Q., Sun, P., Yan, S., Wen, Y., & Zhang, T. (2021). Characterization and prediction of deep learning workloads in large-scale gpu datacenters. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis,

Jeon, M., Venkataraman, S., Qian, J., Phanishayee, A., Xiao, W., & Yang, F. (2018). Multi-tenant GPU clusters for deep learning workloads: Analysis and implications. Technical report, Microsoft Research.

Kamble, T., Deokar, S., Wadne, V. S., Gadekar, D. P., Vanjari, H. B., & Mange, P. (2023). Predictive Resource Allocation Strategies for Cloud Computing Environments Using Machine Learning. Journal of Electrical Systems, 19(2).

Kashyap, S., & Singh, A. (2023). Prediction-based scheduling techniques for cloud data center's workload: a systematic review. Cluster Computing, 26(5), 3209-3235.

Kaur, R., Asad, A., Al Abdul Wahid, S., & Mohammadi, F. (2025). A survey of advancements in scheduling techniques for efficient deep learning computations on GPUs. Electronics, 14(5), 1048.

Khakifirooz, M., Fathi, M., & Dolgui, A. (2026). Theory of AI-driven scheduling (TAIS): a service-oriented scheduling framework by integrating theory of constraints and AI. International journal of production research, 64(2), 642-676.

Khoshkbarforoushha, A., Ranjan, R., Gaire, R., Abbasnejad, E., Wang, L., & Zomaya, A. Y. (2016). Distribution based workload modelling of continuous queries in clouds. IEEE transactions on Emerging Topics in Computing, 5(1), 120-133.

Kyriazi, F., & Thomakos, D. D. (2026). Benchmarks in univariate time-series forecasting: a historical and methodological review. IMA Journal of Management Mathematics, 37(2), 327-374.

Li, B., Arora, R., Samsi, S., Patel, T., Arcand, W., Bestor, D., Byun, C., Roy, R. B., Bergeron, B., & Holodnak, J. (2022). Ai-enabling workloads on large-scale gpu-accelerated system: Characterization, opportunities, and implications. 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA),

Liang, F., Zhang, Z., Lu, H., Li, C., Leung, V., Guo, Y., & Hu, X. (2024). Resource allocation and workload scheduling for large-scale distributed deep learning: A survey. arXiv preprint arXiv:2406.08115.

Liu, G., Lin, W., Zhang, H., Lin, J., Peng, S., & Li, K. (2025). Public datasets for cloud computing: A comprehensive survey. ACM computing surveys, 57(8), 1-38.

Lu, Y., He, D., Ma, T., Liu, Z., Ruan, L., Jiang, J., & Wu, Y. (2026). Bridging the GPU Utilization Gap: Predictive Multi-Dimensional Resource Scheduling for AI Workloads. Proceedings of the 21st European Conference on Computer Systems,

Mahajan, K., Balasubramanian, A., Singhvi, A., Venkataraman, S., Akella, A., Phanishayee, A., & Chawla, S. (2020). Themis: Fair and efficient {GPU} cluster scheduling. 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20),

Moolchandani, D., Kumar, A., & Sarangi, S. R. (2022). Performance and power prediction for concurrent execution on gpus. ACM Transactions on Architecture and Code Optimization (TACO), 19(3), 1-27.

Ovi, S. Alibaba GPU Cluster Spot Resource Dataset. https://www.kaggle.com/datasets/mdsultanulislamovi/alibaba-gpu-cluster-spot-resource-dataset

Pasham, S. D. (2018). Dynamic Resource Provisioning in Cloud Environments Using Predictive Analytics. The Computertech, 1-28.

Renke, L., Piplani, R., & Toro, C. (2021). A review of dynamic scheduling: context, techniques and prospects. Implementing Industry 4.0: the model factory as the key enabler for the future of manufacturing, 229-258.

Rosenfeld, V., Breß, S., & Markl, V. (2022). Query processing on heterogeneous CPU/GPU systems. ACM Computing Surveys (CSUR), 55(1), 1-38.

Sajal, S. M., Hasan, R., Zhu, T., Urgaonkar, B., & Sen, S. (2021). TraceSplitter: a new paradigm for downscaling traces. EuroSys,

Sultana, A., Pakka, N., Xu, F., Yuan, X., Chen, L., & Tzeng, N.-F. (2026). Resource heterogeneity-aware and utilization-enhanced scheduling for deep learning clusters. IEEE Transactions on Computers.

Sun, Y., Baruah, T., Mojumder, S. A., Dong, S., Gong, X., Treadway, S., Bao, Y., Hance, S., McCardwell, C., & Zhao, V. (2019). Mgpusim: Enabling multi-gpu performance modeling and optimization. Proceedings of the 46th International Symposium on Computer Architecture,

Tsenos, M., & Kalogeraki, V. (2025). Exploring GPU-Based Workload Scheduling Techniques for Edge Computing. 2025 IEEE International Conference on Cloud Engineering (IC2E),

Wang, X., Li, Y., Guo, F., Xu, Y., & Lui, J. C. (2023). Dynamic GPU scheduling with multi-resource awareness and live migration support. IEEE Transactions on Cloud Computing, 11(3), 3153-3167.

Wang, Y., Yu, J., & Yu, Z. (2023). Resource scheduling techniques in cloud from a view of coordination: a holistic survey. Frontiers of Information Technology & Electronic Engineering, 24(1), 1-40.

Yang, D., Zhang, W., Ye, Q., Zhang, C., Zhang, N., Huang, C., Zhang, H., & Shen, X. (2023). DetFed: Dynamic resource scheduling for deterministic federated learning over time-sensitive networks. IEEE Transactions on Mobile Computing, 23(5), 5162-5178.

Zhang, S., Diao, L., Meng, Z., Wang, S., Lin, W., & Wu, C. (2025). DyOrc: Efficient Serving of Dynamic Machine Learning Workflows. Proceedings of the 2025 ACM Symposium on Cloud Computing,