Google Cluster Trace Analysis and Failure Prediction

CSIT5970 Course Project Report

May 2025

LI, Zhicheng 21133094 zlijw@connect.ust.hk
WANG, Ziling 21086760 zwanglm@connect.ust.hk
YANG, Zhuorui 21095046 zyangdm@connect.ust.hk
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

Contents

Abstract

As large-scale cluster management systems become the backbone of cloud computing platforms, understanding workload behavior and scheduling dynamics has become increasingly critical. In this project, we conduct a comprehensive analysis of a real-world dataset derived from Google Borg traces. We preprocess semi-structured logs, extract key features such as resource requests and task runtime, and derive metrics like resource efficiency and queuing delay.

Through exploratory data analysis (EDA), we reveal patterns linking task failures to resource request sizes and scheduling classes. We further apply machine learning models—including Logistic Regression, Random Forest, and XGBoost—to predict task failures, achieving a ROC-AUC of 0.998 with XGBoost. Finally, we use clustering techniques to uncover structural differences among failed tasks.

Our findings suggest opportunities for improving scheduling fairness and resource utilization. The analysis pipeline is reproducible and offers valuable insights for optimizing cluster management strategies. The source code and data analysis notebooks are available at `https://github.com/HKUST-CSIT5970-25S/course-project-google-cluster-analysis.git`.

1 Introduction

Large-scale cluster computing systems such as Google's Borg, Kubernetes, and Apache Mesos form the backbone of modern cloud infrastructure. These systems orchestrate thousands of heterogeneous tasks across massive clusters while maintaining high availability and efficiency. Analyzing job traces from such systems provides critical insights into workload characteristics, scheduling behavior, and resource utilization patterns.

However, studying production-scale traces presents several challenges. First, the data is high-volume and semi-structured, with key features such as resource requests embedded in string-encoded dictionaries. Second, task behaviors are highly diverse, with failure events being rare but impactful. Third, many features—such as task priorities, scheduling class, and runtime—interact in non-trivial ways that affect both performance and robustness.

To address these issues, we conduct a comprehensive data analysis on a real-world cluster trace dataset derived from Google's Borg system. Our analysis aims to answer the following research questions:

- How are resource requests related to task failure and efficiency?
- What are the characteristics of tasks with long queuing delays?
- Can we predict task failure using basic scheduling features?

- What patterns exist among failed tasks in terms of clustering?

We preprocess the raw trace data into a structured format and perform exploratory data analysis (EDA) to identify core behavioral trends. We then apply machine learning models—including logistic regression, random forest, and XGBoost—to predict task failures based on their attributes. Finally, we cluster failed tasks using KMeans to uncover behavioral subgroups.

Our study highlights inefficiencies in resource usage and scheduling fairness, and provides actionable insights for system operators and schedulers. The methods and results are reproducible via open-source notebooks and visualized with annotated charts.

2 Related Work

Understanding cluster workloads and scheduling behavior has been a long-standing area of interest in systems research. The release of anonymized traces from Google's Borg system [4] has enabled a wide range of empirical studies on data center dynamics, including resource usage modeling [6], job co-location analysis [5], and failure characterization [3].

Wilkes [6] discusses over-provisioning and underutilization issues in large-scale clusters. Other works focus on job placement fairness and scheduling delay, which also motivate our exploration of queuing patterns and their relationship with resource utilization [1].

Failure prediction in large-scale systems has also been explored using statistical and machine learning models. Gupta et al. [2] propose logistic regression and tree-based methods to anticipate job failures, showing the potential for predictive monitoring systems.

Our work builds on these foundations and provides an end-to-end empirical pipeline for workload characterization, failure prediction, and clustering analysis in data center workloads.

3 Methodology and Experimental Setup

This section outlines the overall methodology adopted in our analysis and describes the technical steps required to prepare, process, and analyze the large-scale Google cluster trace dataset.

3.1 Data Preprocessing

We begin by reading the raw CSV trace data, consisting of over 2 million task instances. To reduce memory pressure and improve processing efficiency, the data was first cleaned using a PySpark pipeline. Key fields such as `resource_request`, `run_duration`, `priority`, and `failed`

were extracted and parsed using regular expressions. Tasks with missing or invalid fields were removed, and extreme values in `queueing_delay`, `resource_efficiency`, etc., were capped or filtered based on quantile thresholds. The cleaned data was stored in Parquet format to facilitate fast reloading.

3.2 Exploratory Analysis Pipeline

Exploratory Data Analysis (EDA) was conducted via Jupyter notebooks in multiple stages. We began by examining the relationship between resource request sizes and failure rates, followed by an investigation into queuing behavior across different scheduling classes and priorities. Additional analyses were conducted to explore runtime distributions and task efficiency extremes. All plots were generated using Seaborn and Matplotlib, and where necessary, sampled data (e.g., top 5% or 10,000 records) was used to improve visualization clarity.

3.3 Failure Prediction and Clustering Strategy

For predictive modeling, we trained three classification models: Logistic Regression, Random Forest, and XGBoost. Each model was evaluated using accuracy, precision, recall, F1-score, and ROC AUC. The input features included task `priority`, `scheduling_class`, `cpu_request`, `mem_request`, and normalized `run_duration`. To mitigate class imbalance, class weights were adjusted, and stratified sampling was applied.

Additionally, we performed unsupervised clustering of failed tasks using KMeans to reveal distinct behavioral patterns. Clustering was based on the same features used in prediction, and results were visualized in 2D plots to show group separation based on runtime and priority.

4 Results and Insights

This section presents detailed findings from our analysis, based on the full Google Cluster workload dataset (over 240k tasks across 4k collections). Visualizations are organized by thematic focus, highlighting the interplay between scheduling behavior, failure risk, and resource efficiency.

4.1 Resource Request Patterns and Failure Risk

We explored how requested resources—specifically CPU and memory—impact task failure rates. Figure 1 displays the distributions of requested CPUs and memory, revealing a highly skewed

pattern where most tasks request fewer than 0.1 cores and small memory allocations. This suggests that the workload is dominated by lightweight tasks.

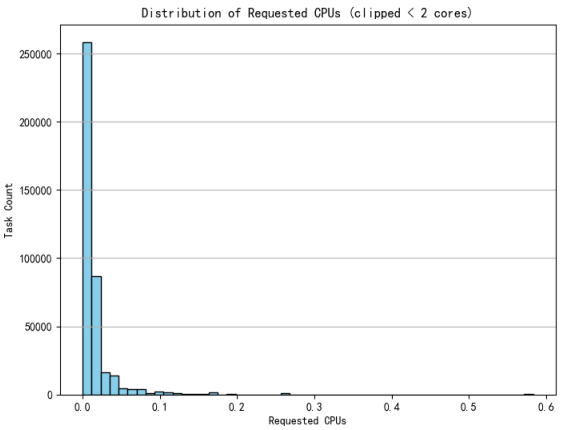


Figure 1: Distribution of Requested CPUs

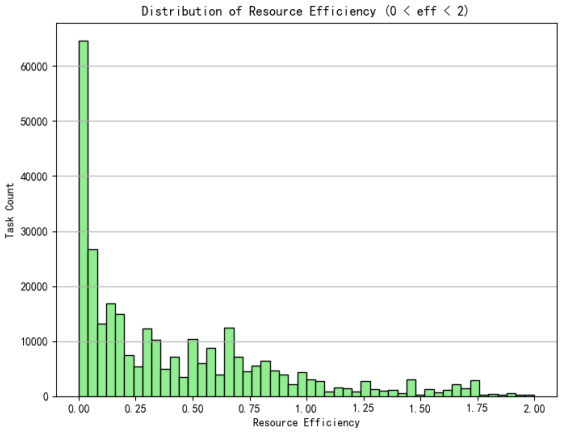


Figure 2: Distribution of Resource Efficiency

To further assess the impact of resource requests on failure risk, we grouped tasks by request size bins and computed failure rates for each. As seen in Figure 3 and Figure 4, tasks with smaller resource requests exhibited slightly higher failure rates. This pattern may reflect stricter constraints or higher contention for lightweight jobs, which are more susceptible to being preempted or misconfigured.

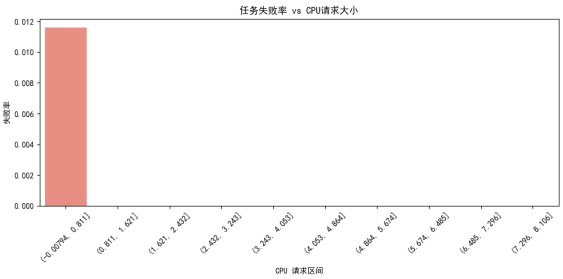


Figure 3: Failure Rate vs. CPU Request Size

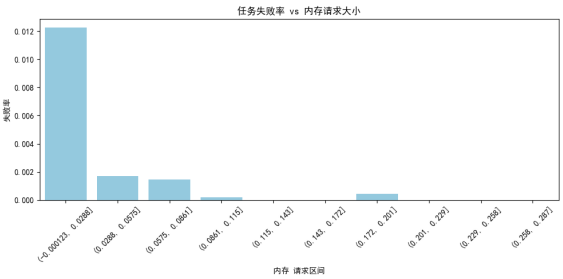


Figure 4: Failure Rate vs. Memory Request Size

Overall, while the absolute failure rate remains low (Figure 5), the slight elevation in failure frequency among smaller-request tasks highlights the importance of robust scheduling for resource-efficient workloads.

4.2 Queuing Behavior and Scheduling Fairness

To investigate scheduling fairness, we divided tasks into “high-queued” and “low-queued” groups based on queuing delay percentiles. Specifically, the top 25% queuing delay tasks were labeled as “high-queued”, and the rest as “low-queued”.

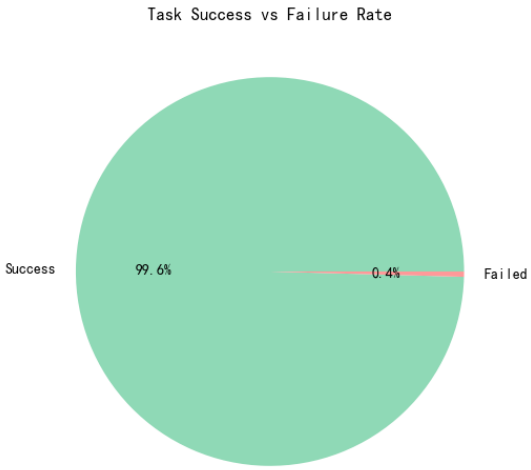


Figure 5: Success vs. Failure Rate across all tasks

Resource Requests. Figure 6 and Figure 7 shows the distribution of CPU and memory requests. High-queued tasks tend to request marginally larger resources, suggesting they may be deprioritized in scheduling due to their heavier footprint.

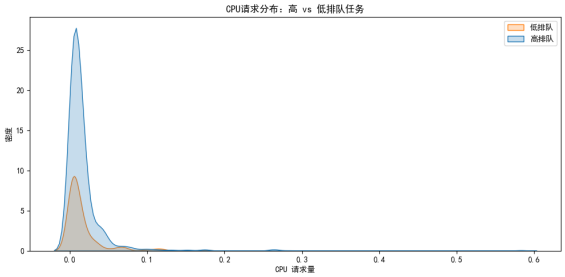


Figure 6: CPU Request Distribution

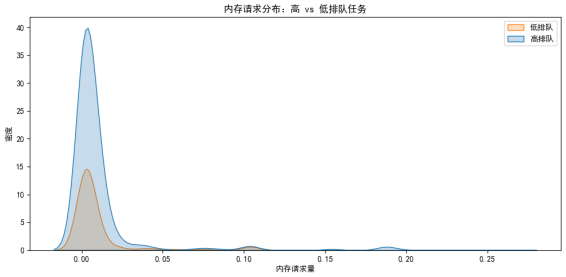


Figure 7: Memory Request Distribution

Priority and Scheduling Class. From Figure 8 and Figure 9, high-queued tasks are more concentrated in lower scheduling classes (0 and 1), while low-queued tasks are more prevalent in class 2 and 3. This suggests the scheduler favors latency-sensitive or interactive tasks. Priority distributions also indicate that low-queued tasks tend to have slightly higher priorities.

Failure Risk by Queue Level. Interestingly, as shown in Figure 10, low-queued tasks exhibit a significantly higher failure rate (1.84%) compared to high-queued ones (0.46%). This counterintuitive result suggests that tasks scheduled more quickly may be speculative, short-lived, or misconfigured, increasing their likelihood of failure.

Run Duration Correlation. To further examine execution patterns, we analyzed run duration against failure rates (Figure 11). Tasks that fail tend to cluster within very short or

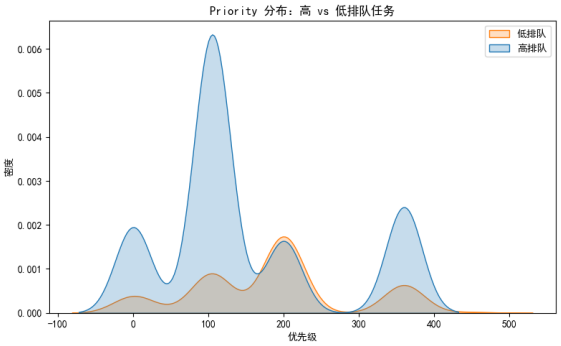


Figure 8: Priority Distribution

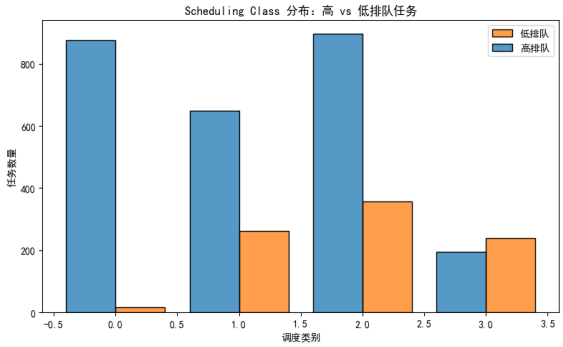


Figure 9: Scheduling Class Distribution

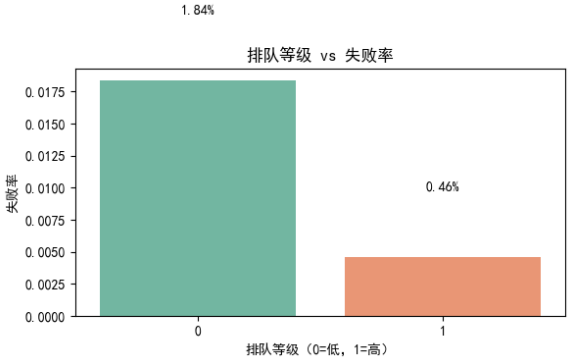


Figure 10: Failure Rate vs. Queuing Level

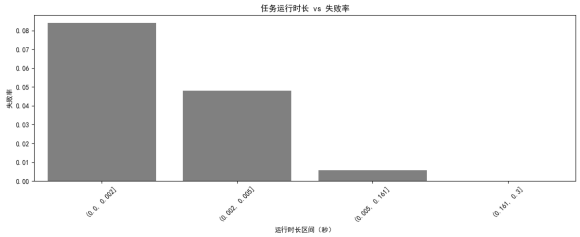


Figure 11: Task Runtime vs. Failure Rate

marginal run durations, reinforcing the hypothesis that many failed jobs are prematurely terminated or misconfigured.

4.3 Efficiency Distribution Analysis

To explore inefficiencies in cluster resource usage, we analyze tasks with extremely low resource efficiency (defined as the ratio of average used CPUs to requested CPUs, below 0.1). This analysis reveals how scheduler decisions or task configurations may lead to underutilization.

Overall Distribution. Figure 12 shows the kernel density of efficiency for both successful and failed tasks. Failed tasks are notably concentrated at very low efficiency values, suggesting a link between poor utilization and failure.

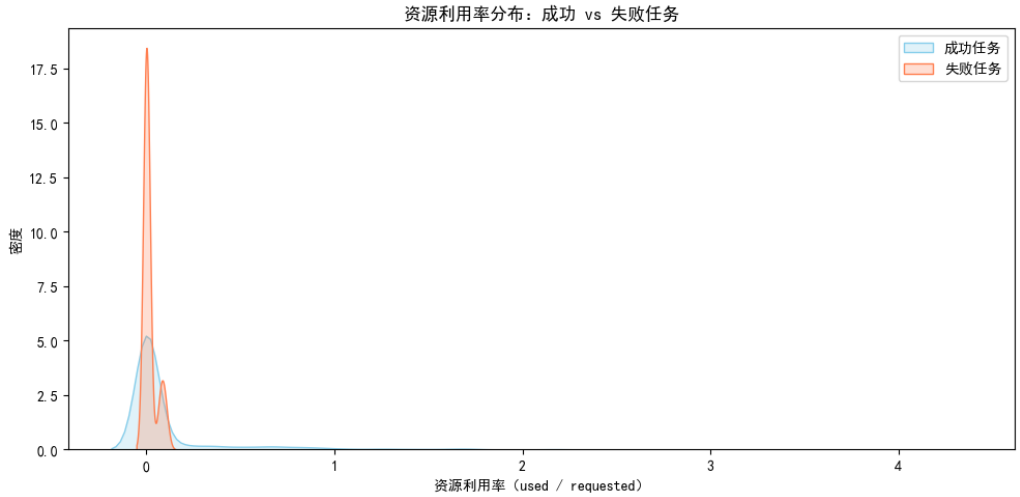
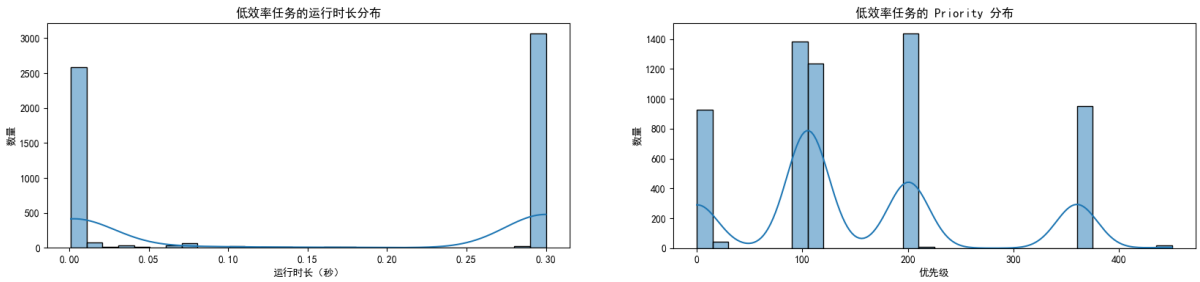


Figure 12: Resource Efficiency Distribution: Successful vs. Failed Tasks

Runtime Behavior of Inefficient Tasks. Figure 13 (left) reveals a U-shaped distribution: many inefficient tasks terminate quickly or occupy full time slots (300s), possibly due to misconfiguration or job starvation. The right panel shows their `priority` distribution—widely spread and not tightly concentrated, indicating priority alone may not explain inefficiency.



(a) Runtime of Inefficient Tasks (b) Priority Distribution

Figure 13: Behavioral Patterns of Inefficient Tasks

Scheduling Class Disparity. Figure 14 highlights that inefficient tasks are most frequently found in Scheduling Classes 0–2, with Class 3 underrepresented. This pattern indicates latency-sensitive jobs (class 3) are likely better optimized by the cluster scheduler, while best-effort or batch jobs suffer inefficiencies.

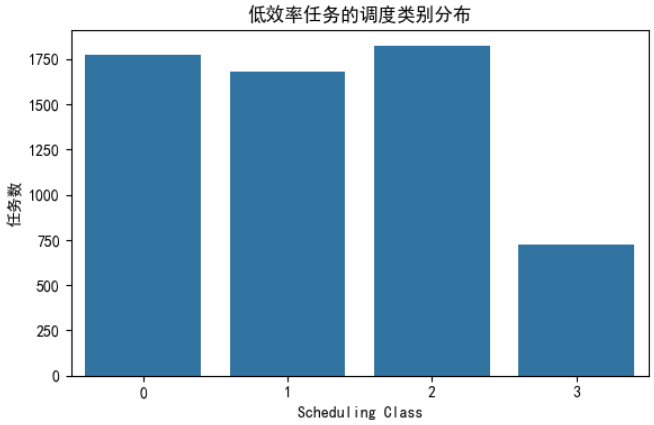


Figure 14: Scheduling Class Distribution for Inefficient Tasks

4.4 Predicting Task Failures via ML

We trained Logistic Regression, Random Forest, and XGBoost models using 9000 samples with class-weight balancing. XGBoost achieved the highest AUC (0.998), followed by Random Forest (0.987), and Logistic Regression (0.918).

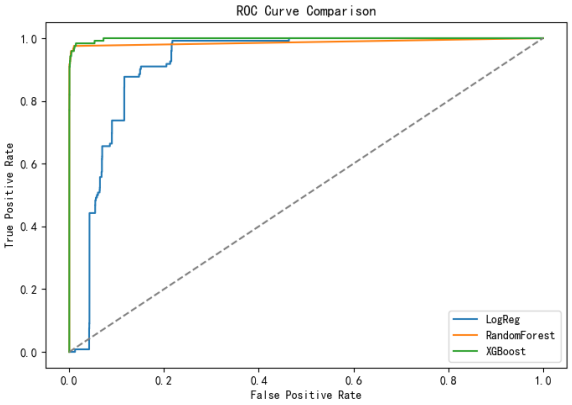


Figure 15: ROC Curve Comparison of Three Models

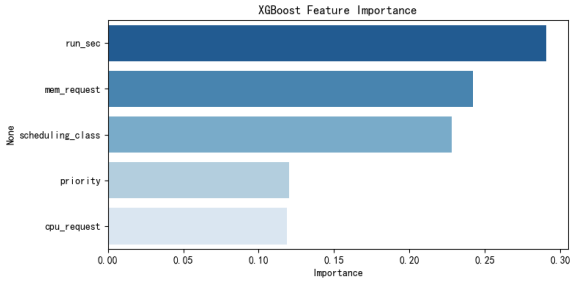


Figure 16: Feature Importance in XGBoost Model

4.5 Clustering Failed Tasks

To further investigate the structural patterns of failures, we applied KMeans (k=3) on failed samples, projecting onto priority and runtime dimensions. Results indicate three clusters: short-lived and low-priority, long-running with moderate priority, and scattered outliers.

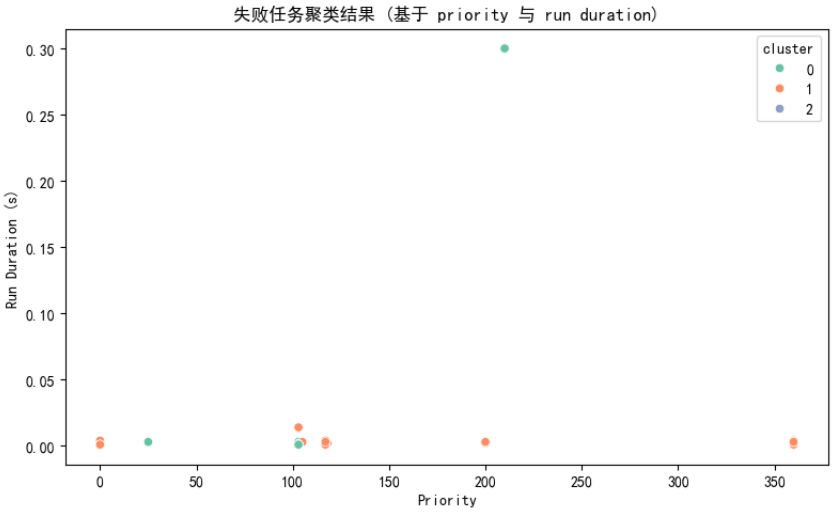


Figure 17: KMeans Clustering of Failed Tasks (projected)

5 Conclusion

In this project, we conducted an in-depth analysis of the Google Borg trace dataset to uncover patterns and risk factors related to task failures and inefficiencies in a large-scale cluster management system. By performing extensive exploratory data analysis (EDA), we observed that failure rates are notably higher among tasks with extremely low resource requests and shorter runtimes, especially within lower scheduling classes.

Our queuing analysis revealed a strong correlation between queuing delay and task priority, indicating fairness issues in resource scheduling. Moreover, we found that inefficient tasks are often either extremely short-lived or long-running but underutilized, and they tend to be distributed across different scheduling categories without clear separation.

We built and evaluated three binary classifiers—Logistic Regression, Random Forest, and XGBoost—for failure prediction. Among them, XGBoost demonstrated the best performance, achieving a ROC-AUC score of 0.998 and identifying `run_sec`, `mem_request`, and `scheduling_class` as the most informative features. Additionally, we applied KMeans clustering to failed tasks, discovering three interpretable failure clusters that may guide future failure handling strategies.

Overall, our study provides both empirical insights and practical tools for understanding workload behavior and improving resource management policies in distributed systems. Future work may involve incorporating temporal trends, performing anomaly detection over time, or designing interpretable predictive frameworks integrated into scheduling systems.

References

[1] Jorge Delgado and Vicente Pelechano. Job scheduling in google clusters: Pitfalls uncovered by simulation. *Computer Networks*, 93:519–538, 2016.

[2] Ishai Gupta, Anna Glikson, Eitan Hadad, and Lior Lapid. Failure prediction in large-scale systems: A case study. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 395–401. IEEE, 2014.

[3] Xue Qin, Minxian Liu, Keqiu Li, and Rajkumar Buyya. Analyzing and modeling historical job data in google clusters. In *Future Generation Computer Systems*, volume 68, pages 162–175. Elsevier, 2017.

[4] Charles Reiss, John Wilkes, and Joseph M Hellerstein. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, pages 1–13. ACM, 2012.

[5] Abhishek Verma, Luis Pedrosa, Mikhail Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems*, pages 1–17. ACM, 2015.

[6] John Wilkes. More google cluster data. Technical report, Google, 2011. `https://github.com/google/cluster-data/blob/master/ClusterData2011.md`.