Vrije Universiteit Amsterdam

Bachelor Thesis

Enhancing Operational Data Synthesis and Predictive Analysis in HPC Clusters Using Large Language Models

Author: Yizhen Zang (2721673)

1st supervisor:	prof. dr. ir. Alexandru Iosup
daily supervisor:	ir. Xiaoyu Chu
2nd reader:	ir. Dante Niewenhuis

A thesis submitted in fulfillment of the requirements for the VU Bachelor of Science degree in Computer Science

August 14, 2024

Contents

List of Figures

List of Tables

Abstract

High-Performance Computing (HPC) clusters are integral to advancing scientific research, industrial optimization, and various computational tasks. Researchers, industrial users, and institutions relying on HPC systems are particularly interested in enhancing the efficiency and reliability of these systems to support critical tasks and innovations.

The increasing complexity and scale of HPC clusters present significant challenges in resource management and optimization, exacerbating issues such as job failures and the scarcity of diverse operational data. These challenges not only impact the performance and reliability of HPC systems, but also increase operational costs and reduce the overall effectiveness of computational resources. Traditional models often struggle with the dynamic nature and complexity of HPC workloads, highlighting the urgent need for innovative approaches that can adapt and learn from evolving patterns.

In this setting, Large Language Models (LLMs) have emerged as powerful tools for identifying complex patterns and dependencies within HPC workloads. This work aims to explore the applications of LLMs in HPC environments. We focus on leveraging LLMs to address two specific problems in HPC: job data synthesis and job end-state prediction. The research aims to evaluate the efficacy of LLMs in generating high-fidelity synthetic datasets and predicting job termination states, which are critical for optimizing the performance and reliability of HPC systems.

Our approach involves a multi-stage design and evaluation process. For job data synthesis, we utilize eight pre-trained LLMs and two specialized LLM-based models. We explore various textualization techniques, preprocessing strategies, and evaluation metrics such as Distance to Closest Record (DCR) and Machine Learning Efficiency (MLE) to assess the quality and utility of the synthetic data. For job end-state prediction, we integrate additional layers into eight pre-trained LLMS to handle categorical and numerical features, comparing their performance with traditional machine learning models using the AUC-ROC metric.

Our experiments show that RoBERTa-Large, GPT-2-based models excel in generating synthetic data with low DCR values, closely mimicking real data. However, MLE evaluations indicate that LLM-synthesized data do not yet match the utility of real datasets for training predictive models. In job end-state prediction, GPT-2 performs competitively with traditional models such as Random Forest and XGBoost, achieving an AUC-ROC of 0.94, highlighting its potential for broader applicability with further refinement.

These results suggest that while LLMs show significant promise in synthesizing and predicting workload data in HPC, ongoing advancements in model architecture, training strategies, and data-handling techniques are necessary to fully realize their potential.

The code and datasets are available at `https://github.com/yyzangg/vu-bsc-thesis`.

Keywords: workload synthesis, workload prediction, data generation, job failure, large language models (LLMs), high-performance computing (HPC)

1 Introduction

In this chapter, we present the context, problem statement, research questions, and contributions of our study, focusing on exploring the application of Large Language Models (LLMs) in High-Performance Computing (HPC) domains.

1.1 Context

HPC clusters are known for their computational power, enabling the execution of complex tasks across various fields, from scientific simulations to industrial optimizations [1]. Despite their capabilities, the growing complexity and scale of HPC clusters present challenges in resource management and optimization [2, 3, 4]. Issues such as scarce diverse operational data and frequent job failures hinder efficient system functioning [5]. These challenges not only affect the performance and reliability of HPC systems, but also increase operational costs and reduce the overall effectiveness of computational resources [6].

Although traditional models have advanced in addressing problems such as job failure prediction within HPC clusters [7, 8], they often struggle with the complexity and variability of operational workloads. Moreover, traditional approaches tend to struggle to adapt to the dynamic nature of HPC environments and follow direct instructions [9], which is crucial for real-time decision-making and optimization [10]. This limitation underscores the urgent need for more innovative approaches to dynamically learn and adapt to the evolving patterns of HPC workloads [11].

In this context, LLMs have become prominent tools for capturing intricate patterns and dependencies within HPC workloads [12]. LLMs offer promising approaches to generating synthetic operational data [13], which can significantly enhance decision-making and operational efficiency by providing high-fidelity data for testing and development without the constraints of data scarcity or privacy concerns [14]. Synthetic data allows for extensive testing and optimization of HPC systems in a controlled environment, ensuring that new algorithms or models can be evaluated thoroughly before deployment. In addition, synthetic data can fill gaps where real-world data is scarce, ensuring that models trained on these data can handle a wide range of scenarios. Furthermore, due to their advanced learning capabilities, LLMs are highly suitable for modeling complex relationships in data, making them ideal for predictive tasks in HPC environments [15].

While LLMs have been utilized for tasks such as parallel programming [16] and data management [17] within the HPC community, their full potential to tackle the unique challenges of HPC clusters remains unexplored. Integrating LLMs effectively into HPC workflows could improve operational efficiency, reduce job failure rates, and improve computational resource management [18]. These improvements can lead to more reliable and efficient HPC systems, which are critical to advance scientific research, industrial applications, and various computational tasks that support modern digital society [19].

1.2 Problem Statement

Inefficiencies in HPC clusters impact a wide range of stakeholders, including researchers, industrial users, and institutions that rely on HPC for critical tasks [20, 21]. Given the challenges and opportunities, our study aims to explore the capabilities of LLMs in HPC environments, with a focus on workload synthesis and predictive analysis. Workload generation and prediction are crucial, interdependent facets of HPC operations, which are pivotal for improving resource allocation [22], reducing job failures [23], and enhancing the overall efficiency of HPC systems [24]. Predictive models leverage historical data to anticipate future outcomes, whereas synthesis techniques can augment sparse datasets by creating new samples that replicate the patterns observed in existing data. Previous works have highlighted the limitations of traditional models in and data synthesis capabilities predictive accuracy [25, 26], underscoring the need for more advanced methodologies.

To bridge the gaps, our research seeks to provide insights into how effectively LLMs can perform these tasks compared to traditional modeling approaches. This study aims to evaluate their efficacy in generating synthetic operational data and predicting job failures, thereby contributing to the development of advanced methodologies for workload synthesis and improved predictive modeling in HPC clusters.

The overarching research question of this project is **How to leverage LLMs to improve operational decision-making for HPC datacenters?** By addressing this question, our objective is to uncover the transformative potential of LLMs in enhancing HPC operations, leading to more reliable and efficient systems.

1.3 Research Questions

The overarching research question is too large for a BSc-sized research project; therefore, we scope our project through a sequence of three narrower research questions that address together a significant part of the overarching research question. In the following, we outline the key research questions (RQs) guiding the investigation into the potential of LLMs to address challenges within HPC clusters.

RQ1 (LLM applications in HPC): **What are the existing applications of LLMs for HPC?**

This question is fundamental to establish the current state of research and applications of LLMs in HPC domains. It is essential to understand existing efforts and successes before proposing new solutions. However, gathering an exhaustive list of applications of LLMs in such a broad and dynamically evolving field can be challenging. The rapid pace at which both LLMs and HPC technologies evolve might also render the findings obsolete rapidly, requiring continuous updating. Moreover, the limited literature on this specific intersection necessitates a systematic approach to identify emerging applications [27]. To guide the systematic review to explore the breadth and depth of LLM applications within the HPC domain, we formulate RQ1 into the following sub-research questions:

- **RQ1.1** How are LLMs applied in HPC for workload synthesis? This sub-research question studies how LLMs are applied in HPC for workload synthesis. Creating high-fidelity data for effective testing and development in HPC environments is challenging. The current literature lacks comprehensive reviews on this topic, highlighting the need for a structured synthesis of knowledge to guide future research and practical applications.

- **RQ1.2** How are LLMs applied in HPC for workload prediction? With this sub-research question, we aim to explore how LLMs are applied in predictive tasks in HPC domains, such as job failure prediction. These tasks require models to handle the high variability and complexity of HPC workloads, making this an essential area of investigation. The potential for LLMs to improve predictive accuracy is critical for resource management and operational efficiency of HPC systems.

- **RQ1.3** How are LLMs applied in other tasks in HPC? This sub-research question investigates how LLMs are applied in other tasks in HPC environments. This is crucial because LLMs have the potential to enhance various HPC-related tasks beyond synthesis and prediction, such as system optimization and resource management. Identifying these applications can open new avenues for research and practical implementation. The challenge here is mapping out the diverse potential applications and understanding how LLMs can be effectively integrated into different aspects of HPC workflows, which has not been comprehensively explored in the current literature.

RQ2 (HPC Job Data Synthesis Using LLMs): **How to design and evaluate LLM-based models for synthesizing job data in HPC clusters?**
Workload synthesis is critical for testing and optimizing HPC systems [24]. RQ2 explores new methodologies for generating synthetic job data and benchmarks these methods against established ones to justify the adoption of LLM technology. Developing methods to effectively generate realistic and useful synthetic data using LLMs involves a deep understanding of both LLM capabilities and HPC operational characteristics, presenting a complex challenge in implementation. Establishing fair and comprehensive benchmarks for comparison also involves statistical and domain-specific challenges. Synthetic data generation is an active research area [28], and while LLMs offer new avenues, they also require adjustments and innovations in model training and data handling to be effective.

- **RQ2.1** (Design): How to generate synthetic job data in HPC clusters using LLMs? RQ2.1 focuses on designing how to generate synthetic job data in HPC clusters using LLMs. This is a significant area of research because generating high-quality synthetic data can vastly improve the ability to test and optimize HPC systems without the constraints of limited or sensitive real-world data. The complexities lie in ensuring the synthetic data accurately represents the statistical properties and variability of real job data, which is crucial for meaningful testing and development. Furthermore, since LLMs were originally designed for natural language processing (NLP) tasks, adapting them to generate tabular or structured data presents an additional layer of difficulty. Techniques need to be developed to convert HPC job data into formats that LLMs can process and vice versa, which is not straightforward.

- **RQ2.2** (Evaluation): How are the techniques developed in RQ2.1 performing, relative to the non-LLM state-of-the-art? This sub-research question seeks to establish the practical utility of LLM-based synthetic data generation methods. The main challenges include using appropriate benchmarks and metrics to assess the quality and utility of the synthetic data. Comparing LLM-based methods to traditional ones helps to validate their effectiveness and identify areas where they may offer superior performance or need further improvement.

RQ3 (HPC Job End-State Prediction Using LLMs): **How to design and evaluate LLM-based models for predicting job end-state in HPC clusters?**

In the context of workload prediction within HPC clusters, our focus shifts towards a specific predictive task: job end-state classification. Prediction of job failures is critical to improve the efficiency and reliability of HPC clusters [29]. Employing LLMs for this task involves understanding not just the technical aspects of model training but also the specific characteristics of HPC workloads and failure modes. Predictive modeling of job failures has been explored using various machine learning techniques [30, 31, 32], but LLMs may offer new insights or capabilities that traditional models do not. Yet the novelty of LLM applications in this area adds complexity due to their black-box nature and the need for large-scale data.

- **RQ3.1** (Design): How to predict job failures in HPC clusters using LLMs? RQ3.1 is crucial as accurate job failure prediction can significantly enhance the operational efficiency and reliability of HPC systems. The complexity of this task lies in the need to handle structured data before feeding input to LLMs.

- **RQ3.2** (Evaluation): How are the techniques developed in RQ3.1 performing, relative to the non-LLM state-of-the-art? With this sub-research question, we aim to quantify the effectiveness of LLM-based predictive models. This evaluation is essential to validate the practical applicability and advantages of LLMs in predicting job failures in HPC clusters. The challenges include establishing robust evaluation frameworks and metrics that can fairly compare LLM-based methods with traditional approaches.

1.4 Research Methodology

In this section, we briefly introduce the approaches used to address each research question.

RM1 - Systematic literature review: To address RQ1, which explores the existing applications of LLMs for HPC, our research employs a systematic literature review methodology [33, 34], as detailed in section 2.2. The approach begins with the formulation of sub-research questions to narrow down the focus: examining how LLMs are applied in HPC for workload synthesis, workload prediction, and other HPC tasks. We utilize a comprehensive set of keywords to conduct searches across major academic databases, such as Google Scholar and IEEE Xplore, ensuring a broad collection of relevant literature. This approach is essential to identify both the current uses and the potential areas for innovation of LLMs within the HPC domain. This methodology enables us to map out the landscape of LLM applications in HPC and identify areas needing further research in Chapter 2, providing a foundational overview for the subsequent sections of our study.

RM2 - Design and evaluation of LLMs for job data synthesis: To explore the generation and performance evaluation of synthetic job data in HPC clusters using LLMs, our methodology incorporates a series of practical steps aimed at synthesizing, validating, and comparing this data against state-of-the-art models. This methodology is detailed in Chapter 3. We use a pre-processed dataset from SURFLisa [35] to fine-tune and evaluate 8 general-purpose LLMs and 2 table-generation LLMs. For the set of general-purpose LLMs, we implement a multi-stage design process that includes detailed preprocessing, fine-tuning, and generation pipeline to produce synthetic operational traces that closely mimics real HPC job characteristics (details in 3.2). The effectiveness of the synthetic data is assessed against traditional models (TabGAN and CTGAN) using two key metrics: Distance to Closest Record (DCR) [36] and Machine Learning Efficiency (MLE) [37] (details in 3.4). These metrics evaluate the similarity of the synthetic data to actual data and its utility in predictive modeling tasks, respectively.

RM3 - Design and evaluation of LLMs for job end-state prediction: To examine the effectiveness of LLMs for predicting job end-states in HPC clusters, we delineate a structured methodology integrating various stages of model preparation, training, and evaluation. This approach is elaborated upon in Sections 4.2 and 4.3. Our strategy begins with the preparation of a specialized dataset derived from SURFLisa [35], focusing only on features available at job submission. We utilize eight pre-trained LLMs to implement a structured design. The predictive performance of these LLMs is evaluated against established baseline models including Logistic Regression, Random Forest, and XGBoost [38], which are known for their effectiveness in classification tasks.

1.5 Research Contributions

Our research contributions are categorized into conceptual contributions (CC) and technical contributions (TC), reflecting the theoretical and practical advancements achieved through this work.

CC1 This work presents a comprehensive understanding of the current state of LLM applications in HPC, highlighting key areas such as workload synthesis and prediction, and identifying significant gaps and challenges in existing literature.

CC2 We provide a detailed analysis of the limitations and challenges in integrating LLMs into HPC workflows, focusing on the high computational intensity, substantial memory requirements, and the specialized expertise needed to interpret model outputs.

CC3 We systematically explore the use of LLMs for HPC job data synthesis, highlighting their potential to generate high-fidelity synthetic datasets that closely resemble real operational data. This capability is crucial for enhancing the testing and optimization of HPC systems.

CC4 Our study extends the scope of LLMs to predictive analysis within HPC environments, showcasing their ability to predict job termination states. This advancement aids in improving the management and operation of HPC systems.

TC1 We develop and evaluate a robust methodology for synthesizing job data in HPC clusters using eight pre-trained LLMs, including BART-Base, BART-Large, BERT-Base, BERT-Large, DistilBERT-Base, RoBERTa-Base, RoBERTa-Large, and GPT-2, alongside two specialized LLM-based models, GReaT and REaLTabFormer.

TC2 Our research introduces and evaluates various textualization techniques for preprocessing input data, such as natural language serialization and flattened data representation, providing insights into their impact on synthetic data quality across different LLMs.

TC3 We demonstrate that models such as RoBERTa-Large and GPT-2, when fine-tuned appropriately, can generate high-quality synthetic data with low Distance to Closest Record (DCR) values, indicating their capability to closely mimic real data.

TC4 Our evaluation using Machine Learning Efficiency (MLE) metrics reveals that while LLM-generated synthetic data shows promise, it does not yet match the utility of real datasets for training predictive models, indicating a need for further refinement.

TC5 In the context of job end-state prediction, we fine-tune eight pre-trained LLMs, including two variants of ALBERT, and integrate additional layers to handle both categorical and numerical features. Our research demonstrates a novel approach to leveraging LLMs for structured data classification tasks, offering a foundation for future advancements in LLM-based predictive modeling in HPC environments.

TC6 Our comparative analysis of traditional models (Logistic Regression, Random Forest, XGBoost) with LLMs (GPT-2, and variants of BERT, DistilBERT, ALBERT, and RoBERTa) using the AUC-ROC metric highlights that GPT-2 performs competitively with traditional models, while other LLMs show varying levels of effectiveness.

TC7 We reveal that simple prompt engineering strategies for transforming structured numerical data into textual formats can impact predictive accuracy, emphasizing the need for more sophisticated techniques to preserve critical nuances in the data.

These contributions collectively advance the field of HPC by leveraging the capabilities of LLMs, providing a foundation for future research and practical applications aimed at optimizing and enhancing the performance of HPC systems.

1.6 Statement of Non-Plagiarism

I hereby declare that this study is a result of my independent research and original work. Except where references are made to the work of others, all content is my own work and has not been submitted for any other degree or professional qualification. I have adhered to all guidelines for the use of copyrighted material, including full and accurate citations of all sources consulted, both printed and electronic.

1.7 Thesis Structure

The rest of this work is organized as follows. Chapter 2 addresses RQ1 by reviewing existing applications of LLMs in HPC, focusing on workload synthesis, workload prediction, and other relevant tasks. Chapter 3 answers RQ2 by detailing the methodology for generating synthetic job data using LLMs and evaluating the quality of the synthesized data. RQ3 is addressed in Chapter 4, where we explain how LLMs can be used to predict job end-states in HPC clusters and compare their performance with traditional models. Finally, we summarize the key findings, discuss limitations, and suggest directions for future research in the application of LLMs in HPC in Chapter 5.

2 Literature Review of LLM Applications in HPC

In this chapter, we address RQ1 *What are the existing applications of LLMs for HPC?*, with a specific focus on workload synthesis and prediction.

2.1 Context and Scope of LLM Applications in HPC

The intersection of LLMs and HPC has gained significant attention, with prior research focusing primarily on the design and implementation of HPC systems to facilitate LLM training [39, 40, 41, 42]. Although recent efforts have leveraged LLMs for various HPC tasks [43, 44, 19], a notable gap remains in the literature concerning the application of LLMs to optimize the performance of HPC systems themselves. Moreover, there is an absence of a thorough review that summarizes and compares applications of LLMs in the HPC domain.

Existing studies concentrate on programming-related tasks within HPC [18], such as parallel programming [45, 46] and code optimization [47]. However, this leaves a critical need to explore broader applications of LLMs in optimizing HPC systems, which would lead to improved resource management, improved job scheduling, and overall better performance of HPC clusters [48].

In response to these gaps, our literature review adopts a systematic approach to scrutinize the breadth of LLM applications within HPC settings, particularly focusing on workload synthesis and prediction. This method, influenced by established systematic literature review techniques by Booth et al. [33] and Xiao et al. [34], involves a comprehensive search in multiple academic databases using well-defined keywords related to our research questions. The process follows a meticulous screening methodology, ensuring that only the most relevant studies are included in our analysis, as described in Section 2.2.

Our review is structured to provide a clear understanding of how LLMs are currently applied in the HPC domain and to identify unexplored opportunities that could benefit from LLM capabilities. The primary areas of focus include the generation of synthetic workload data for simulation and testing, as well as the development of predictive models to enhance decision-making processes within HPC systems. Table 1 presents a comparative overview of current LLM applications, highlighting various models and their applications across different HPC tasks, thus setting a foundation for the detailed discussions in the subsequent sections of this chapter.

This overview serves as a primer for the subsequent sections, where each segment of the literature is dissected according to the key areas identified: workload synthesis (Section 2.3), workload prediction (Section 2.4), and other relevant applications (Section 2.5). By mapping out these areas, we aim not only to present what is currently known but also to illuminate the pathways for future research endeavors (Section 2.6), particularly those that harness the untapped potential of LLMs in enhancing the efficiency and capabilities of HPC systems.

2.2 Methodology for Systematic Literature Review

To systematically explore the existing applications of LLMs within the HPC domain, we adopt a comprehensive approach to identify, screen, and analyze relevant literature.

Table 1: Overview of applications of LLMs in HPC environments.

Reference	Applications	Models	Datasets
Shi et al. [49]	memory workload synthesis	REaLTabFormer [50]	SPEC2017 [51]
NetLLM [52]	viewport prediction adaptive bitrate streaming cluster job scheduling	NetLLM	Jin2022 [53] FCC [54] TPCH [55]
LLMTune [56]	database knob tuning	GPT-4 Mistral 7B [57]	TPC-H JOB BIRD [58]
LLMDB [17]	query rewrite database diagnosis data analytics	LLMDB	self-collected
HPC-GPT [59]	AI model management dataset management data race detection	LLaMa LLaMa 2	self-collected
HPC-Coder [16]	code completion OpenMP labeling performance prediction	GPT-2 GPT-Neo PolyCoder	self-collected
LM4HPC [19]	code similarity analysis parallelism detection OpenMP Q&A	CodeBERT GraphCodeBERT gpt-3.5-turbo Dolly 2.0 Cerebras-GPT augAST DeepSCC StarChat-Alpha	POJ-104 DRB-ML OMP4Par OMPQA
MPIrigen [60]	MPI-based parallel program	MonoCoder PolyCoder GPT-3.5	HPCorpusMPI

We begin with defining the keywords and phrases to capture the scope of each sub-research question. For workload synthesis, the search includes "workload synthesis", "synthetic data", and "data generation". For workload prediction, terms such as "LLM" and "job failure prediction" are used. For the broader applications of LLMs in HPC, the keyword list comprises "Large Language Models", "LLMs," and "HPC". The literature search is then conducted across multiple academic databases, including Google Scholar and IEEE Xplore. We carefully document each paper from the search results in the reference management software Zotero. The screening process consists of two stages. First, we check the title and abstract to exclude articles that are not aligned with the research questions. This step ensures that only potentially relevant studies proceed to the next stage. In the second stage, the full text of the selected articles is reviewed to confirm their relevance, ensuring the quality and pertinence of the analyzed literature.

Our methodology ensures a systematic and rigorous review process, offering a comprehensive understanding of the current applications of LLMs in HPC, especially in workload synthesis and prediction. This structured approach lays the groundwork for identifying gaps and opportunities for future research to enhance HPC system performance using LLMs.

2.3 Applications of LLMs in HPC Workload Synthesis

Workload synthesis is essential in HPC domains, as synthetic datasets are vital for simulating diverse scenarios without being constrained by limited historical data. This capability supports more robust experimentation and optimization of HPC systems [49].

Traditionally, workload synthesis has relied on statistical methods to generate representative datasets [61]. Recent advancements have introduced machine learning techniques [37, 62]. For example, Lin et al. [63] propose a GAN-based method specifically tailored for time-dependent cloud workload generation, demonstrating its ability to simulate realistic and diverse workload scenarios.

Although not yet widely adopted, LLMs offer significant advantages in workload synthesis for HPC systems. LLMs can generate synthetic tabular data with higher fidelity and diversity compared to traditional methods [64, 65, 66], thereby enabling more accurate performance evaluation and resource allocation strategies. Despite these advancements, gaps remain in the application of LLMs for workload synthesis in HPC settings. While LLMs have shown promising performance in generating synthetic tabular and relational data [14, 50, 13], most prior works focus on text-to-text generation [12, 67, 68]. This highlights the need for further research into adapting LLMs for HPC-specific tabular data generation.

2.4 Applications of LLMs in HPC Workload Prediction

Predictive analysis of HPC workload is crucial for enhancing system performance, reliability, and efficiency [8, 30]. By forecasting job termination states, resource utilization, and potential failures, predictive models can significantly improve the management and operation of HPC systems.

However, few studies have explored predictive analysis within the HPC domain. Nichols et al. [16] investigate the modeling of parallel programs to predict performance bottlenecks and system failures. While their approach offers valuable insights, it primarily focuses on specific aspects of parallel programming rather than providing a comprehensive predictive model for workload outcomes. This highlights a gap in the literature, particularly in using LLMs for workload prediction in HPC.

To bridge this gap, we look at the advancements in general LLMs and their applications in predictive tasks. Generative table pre-training models have been adapted to improve predictive tasks associated with tabular data [9, 69, 70, 71, 72]. Zhang et al. [66] introduce TAPTAP, which leverages table pre-training to enhance models for tasks such as classification and regression on tabular data. TAPTAP shows significant improvements over traditional methods, particularly in scenarios with imbalanced or missing data. Hegselmann et al. [71] design TabLLM, a method for zero-shot and few-shot classification using

LLMs, showcasing competitive performance against deep learning and traditional methods on benchmark datasets. TABLET by Slack and Singh [73] introduces a benchmark of tabular datasets annotated with natural language instructions, demonstrating significant improvements in model performance. Similarly, Yang et al. [74] highlight the potential of LLMs trained on enriched tabular datasets to perform zero-shot and few-shot predictions effectively, setting new benchmarks in predictive accuracy for tabular data.

General-purpose LLMs have also been effectively repurposed for time-series forecasting. Jin et al. [11] present TIME-LLM, which reprograms LLMs for time-series forecasting through innovative alignment of time-series data with textual data, achieving superior performance in both few-shot and zero-shot scenarios. Gruver et al. [75] demonstrate that models such as GPT-3 and LLaMA-2 can perform zero-shot time-series extrapolation by treating time-series data as sequential text, achieving performance comparable to specialized models. Garza and Mergenthaler-Canseco [76] introduce TimeGPT, a foundation model for time series, capable of generating accurate predictions across diverse datasets, highlighting the broad applicability of LLMs in time series analysis.

Despite the progress, the potential of LLMs to handle the complexity and variability of HPC workloads remains largely unexplored. Further research is needed to utilize LLMs to predict job termination states, resource usage, and failure rates in HPC environments.

2.5 Other Applications of LLMs in HPC

The applications of LLMs have gained increasing interest in various aspects of HPC domains, including performance tuning, data management, and programming language-related tasks.

System Optimization

Lu et al. [48] explore the intersection of LLMs and cloud-native computing architectures and propose an AI-native computing paradigm that optimizes resource usage and costs. Wu et al. [52] develop NetLLM, an LLM-adapted framework for networking tasks. NetLLM efficiently processes task-specific information and improves the efficiency and accuracy of predictions across various networking scenarios such as viewport prediction, adaptive bitrate streaming, and cluster job scheduling. Zhou et al. [17] introduce LLMDB, a data management paradigm augmented with LLMs to improve query rewriting, database diagnosis, and data analytics.

Data Management

Chen et al. [19] present LM4HPC, a framework designed to support the application of LLMs in HPC software analysis and optimization. The framework is tailored to handle HPC-specific datasets, AI models, and pipelines. Ding et al. [59] introduce HPC-GPT, a LLaMA-based model fine-tuned for HPC domain tasks. This model addresses the gap between general-domain NLP performance and tHPC-specific needs, demonstrating effectiveness in detecting data races and managing AI models within HPC environments. Fernandez et al. [77] discuss the transformative potential of LLMs in data management. They argue that LLMs can significantly enhance automation in areas such as entity resolution, schema matching, data discovery, and query synthesis.

Code-Based Tasks

LLMs have shown significant potential in assisting with various code-based tasks in HPC settings, including parallel code generation, code cleaning, and optimization. Nichols et al. [46] evaluate the capabilities of state-of-the-art language models in generating parallel code using the ParEval benchmark. Their study also introduces novel metrics for evaluating generated code and highlights areas where LLMs excel and where they need improvement. Godoy et al. [45] assess the OpenAI Codex's ability to generate HPC parallel programming model kernels across different languages and programming models. They propose a proficiency metric to quantify the results, finding that the OpenAI Codex performs well for mature programming models but less so for emerging ones such as HIP. This evaluation provides valuable insights into the current capabilities and limitations of LLMs in HPC programming. Schneider et al. [60] introduce MPIrigen, a domain-specific language model fine-tuned to generate MPI-based parallel programs, significantly improving the accuracy of generating MPI function calls. This study underscores the importance of domain-specific fine-tuning in optimizing LLMs for parallel computing tasks. Chen et al. [78] propose an LLM-based approach for data race detection, combining prompt engineering and fine-tuning techniques, demonstrating the effectiveness of LLMs in detecting data races but also highlighting current limitations compared to traditional tools.

2.6 Challenges in LLMs Integration and Future Directions

The integration of LLMs into HPC tasks has shown promise, but there are still significant gaps and challenges to address.

A major challenge is the computational intensity and memory demands of LLMs, which can strain HPC resources and hinder scalability [19, 39]. Besides, often requires tailored optimization techniques and efficient use of distributed computing resources [40, 43]. The efficacy of LLMs in HPC tasks has also been hindered by the specialized knowledge necessary for interpreting the model's outputs [59]. Another challenge identified is the lack of comprehensive studies addressing the integration of LLMs with existing HPC workflows and infrastructures [18, 41].

Looking forward, future research must focus on several key areas. First, there is a need for comprehensive evaluations of LLMs across diverse HPC tasks to better understand their performance and limitations in these contexts [18]. Godoy et al. [45] believe that generative AI can have an extraordinary and beneficial impact on HPC software development, maintenance, and education in the future, emphasizing the importance of specialized benchmarks and performance metrics tailored to HPC applications of LLMs. Furthermore, Nichols et al. [46] highlight the need for improved capabilities in generating parallel code, suggesting that current state-of-the-art LLMs still have significant room for improvement in this area. Lastly, Fernandez et al. [77] point out that LLMs have the potential to disrupt existing systems, necessitating developing new methodologies to integrate LLMs effectively into current data management and computational frameworks.

2.7 Summary of LLM Applications in HPC

This review of the literature has systematically explored the application of LLMs within HPC settings, highlighting significant advancements and persistent gaps in current research. Our comprehensive examination covered several key areas: workload synthesis, workload prediction, and broader applications of LLMs in HPC. We summarize the key findings as listed below.

F1 The intersection of LLMs and HPC has gained significant attention, but there remains a notable gap in the literature concerning the application of LLMs to optimize the performance of HPC systems.

F2 For workload synthesis, LLMs have shown the potential to generate synthetic datasets with higher fidelity and diversity compared to traditional methods, which is vital for robust experimentation and optimization of HPC systems.

F3 Despite the promise of LLMs in workload synthesis, their application in HPC settings is still emerging, highlighting the need for further research to adapt these models for HPC-specific tabular data generation.

F4 Advances in general LLMs and their adaptation for predictive tasks, including generative table pre-training models and time-series forecasting, have shown significant improvements in predictive accuracy for tabular data and time-series data, respectively.

F5 The application of LLMs to handle the complexity and variability of HPC workloads remains largely unexplored, necessitating further research to leverage their full potential in predicting job termination states, resource usage, and failure rates in HPC environments.

F6 Beyond workload synthesis and prediction, existing studies predominantly focus on programming-related tasks within HPC, leaving a critical need to explore broader applications of LLMs in enhancing resource management, job scheduling, and overall system performance.

F7 Challenges in integrating LLMs into HPC tasks include the computational intensity and memory requirements of these models, the need for tailored optimization techniques, and the specialized knowledge required to interpret model outputs.

Overall, while the integration of LLMs into HPC tasks shows considerable promise, the complexity and novelty of such applications necessitate ongoing research.

3 Design and Evaluation of LLMs for Job Data Synthesis

3.1 Overview of Job Data Synthesis Using LLMs

In this chapter, we explore the application of LLMs for synthesizing operational job data in HPC clusters. Our goal is to determine how effectively LLMs can generate synthetic datasets that closely replicate real operational data. This synthesis is crucial for improving the efficiency and effectiveness of HPC systems, as it provides high-fidelity data for testing and development without the limitations of data scarcity or privacy concerns.

The methodology employed spans several critical stages as detailed in subsequent sections. Initially, we address the selection and preparation of input data, which involves standardizing the features and adopting various textualization techniques to prepare the data for LLM processing (details in Section 3.2). Here, we explore different textualization approaches, such as flattening and serialization, and assess their impact on synthetic data quality using the Distance to Closest Record (DCR) metric (details in Section 3.4.2). This metric, alongside Machine Learning Efficiency (MLE), provides a quantitative basis to evaluate both the fidelity and utility of the synthesized datasets(details in Section 3.4.3). Preprocessing is followed by the training phase where models are fine-tuned on HPC-specific tasks. The choice of model architecture and the tuning parameters are critical to the success of this phase, as they directly influence the quality of the synthetic data generated. Post-processing of the synthetic data ensures that it adheres to the required standards and formats for realistic application scenarios.

Throughout the chapter, we dissect the performance of various models and discuss their efficacy in replicating the complexities of real HPC workloads. In Section 3.4, we provide a detailed evaluation of these models against benchmark datasets, aiming to highlight effective strategies and pinpoint areas that need improvement.

3.2 Methodology for LLM-Based Job Data Synthesis

This section details the methodology to utilize pre-trained LLMs for generating synthetic job data in HPC clusters. Our design process is guided by specific requirements to generate realistic and useful job traces.

3.2.1 LLMs Explored for Job Data Synthesis

The models used include BART-Base, BART-Large, BERT-Base, BERT-Large, DistilBERT-Base, RoBERTa-Base, RoBERTa-Large, and GPT-2. Except for the 8 general-purpose LLMs, we also examine two GPT-2-based models designed specifically for generating synthetic tabular data. Below is a brief introduction to each model.

GPT-2 [79] is a robust, autoregressive model that generates coherent and contextually relevant text based on the input it receives. Its utility in generating large volumes of coherent text makes it particularly suitable for synthesizing complex job data in HPC clusters.

BERT [80] introduces deep bidirectional training of transformers to effectively capture context from both directions, making it robust across a variety of NLP tasks. The model's architecture allows for minimal changes for task adaptation, leading to state-of-the-art performance on multiple benchmarks.

BART [81] is designed with an encoder-decoder architecture, where the encoder functions similarly to BERT. It is trained to reconstruct the original text from corrupted versions, combining the strengths of both bidirectional and autoregressive transformers. This makes BART exceptionally versatile for both generation and comprehension tasks.

RoBERTa [82] iterates on BERT by optimizing hyperparameters and training data size, significantly improving performance and robustness in language tasks.

DistilBERT [83] focuses on retaining most of the BERT's performance while being smaller, faster, and lighter, making it suitable for environments with constraints on computational resources.

GReaT [65] leverages an autoregressive generative LLM framework to produce synthetic tabular data that closely resembles real datasets. It supports conditional generation, allowing it to adapt to various feature distributions and data sizes, making it ideal for synthesizing realistic HPC cluster job data.

REaLTabFormer [50] extends the capabilities of LLMs to relational tabular data generation, efficiently modeling complex inter-table relationships as well as intra-table data characteristics. REaLTabFormer uses a combination of autoregressive and sequence-to-sequence models to ensure high fidelity and structural integrity in generated datasets.

3.2.2 Design Requirements for Job Data Synthesis Using LLMs

Our methodology for synthesizing job data with Transformer-based LLMs is guided by the key requirements below.

Data Integrity and Realism The synthetic data must closely resemble the original dataset in terms of statistical properties and patterns to be useful for analysis and application.

Scalability The design must efficiently handle large datasets, typical in HPC environments, ensuring computational and memory efficiency.

Generality and Flexibility The approach should be adaptable to various types of data and different HPC workloads.

Simplicity and Reproducibility The process should be straightforward enough to be easily replicated and validated by other researchers or practitioners.

Model Performance The chosen architecture and training regimen should optimize the model's ability to generate high-quality synthetic data.

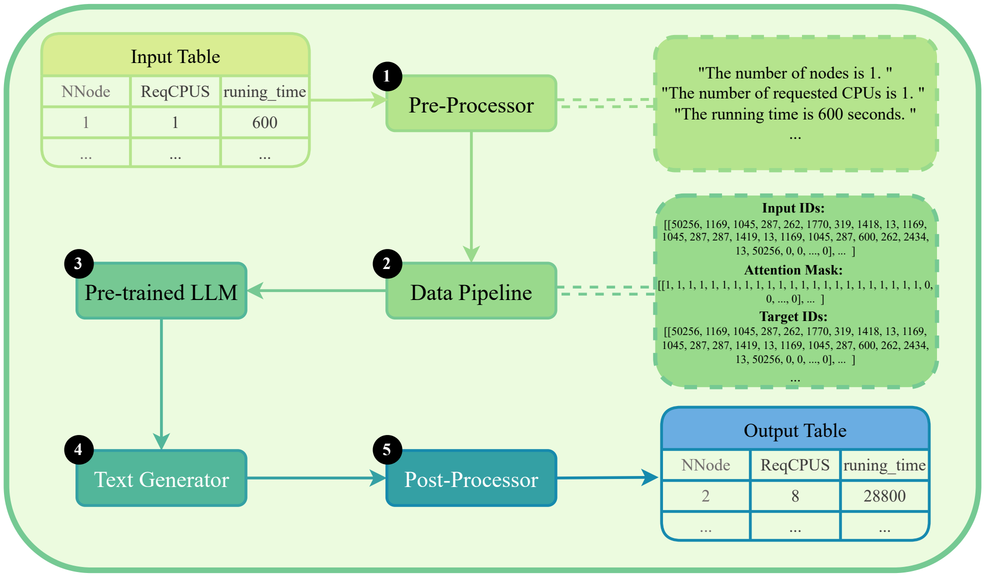
3.2.3 Design Choices for Job Data Synthesis Using LLMs

Our approach incorporates several stages, from preprocessing the input data to generating and post-processing the synthetic outputs, as illustrated in Figure 1.

Preprocessing

For job data synthesis, the input is typically a dataset containing numerical and categorical metrics derived from historical job execution records within HPC clusters. Therefore, before feeding the dataset into the models, we need to prepare the dataset through the necessary preprocessing steps (❶). These include standardization of numerical features, handling missing values, and textualization of the table.

Various approaches exist to textualize the input table [14, 65, 13]. One method is to flatten the table and concatenate the numerical values into a single string. This method



Input Table

NNode	ReqCPUS	runing_time
1	1	600
...	...	...

"The number of nodes is 1. "
"The number of requested CPUs is 1. "
"The running time is 600 seconds. "
...

Input IDs:
[[50256, 1169, 1045, 287, 262, 1770, 319, 1418, 13, 1169, 1045, 287, 287, 1419, 13, 1169, 1045, 287, 600, 262, 2434, 13, 50256, 0, 0, ..., 0], ...]
Attention Mask:
[[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, ..., 0], ...]
Target IDs:
[[50256, 1169, 1045, 287, 262, 1770, 319, 1418, 13, 1169, 1045, 287, 287, 1419, 13, 1169, 1045, 287, 600, 262, 2434, 13, 50256, 0, 0, ..., 0], ...]
...

Output Table

NNode	ReqCPUS	runing_time
2	8	28800
...	...	...

Figure 1: A high-level illustration of utilizing pre-trained LLMs for job data synthesis.

focuses purely on numerical data without adding a natural language context. Alternatively, features can be directly converted into embeddings, which are then used as inputs to the model. Another way is to treat each cell of the table as a token and process the entire table as a sequence of tokens.

In our design, we adopt the textual encoding method proposed by Borisov et al. [65]. This approach involves serializing rows into detailed natural language descriptions for RoBERTa-Base, RoBERTa-Large, and GPT-2. As LLMs preprocess tabular data as full text, they circumvent the curse of dimensionality associated with one-hot encoding high-dimensional data [14]. However, for BART-Base, BART-Large, BERT-Base, BERT-Large, and DistilBERT-Base, we use flattened data representation. This decision is driven by empirical findings that serialization degrades the performance of these LLMs in terms of distance to closest record (DCR) mean by three orders of magnitudes, as detailed in Section 3.4.1.

Although the natural language serialization method is more computationally expensive due to the length and complexity of the input sequences, it leverages the LLMs' capabilities to understand and generate human-like text and provide more context to the model. In contrast, flattened data representation is simpler and faster to implement and process, but with the cost of losing some contextual information.

Data Pipeline

With preprocessing completed, the dataset is processed through a structured data pipeline (❷) to facilitate model training and generation. This phase begins by segmenting the input data into smaller chunks to facilitate batch processing, ensuring efficient memory utilization and computational performance. Each chunk undergoes independent processing to maintain data integrity, with overlapping between consecutive chunks aiding in capturing interdependencies and patterns across job execution records.

The processed dataset is tokenized using an appropriate tokenizer associated with the chosen LLM, which ensures that each data point is represented in a format suitable for the model input.

Fine-Tuning

Fine-tuning adapts the pre-trained model to a new task by training it on a task-specific dataset. There are several alternative approaches for fine-tuning pre-trained LLMs. One prominent method is self-supervised fine-tuning, where the model is trained on unlabeled data [84]. This method allows the model to learn representations that capture meaningful features from the data without requiring labeled examples for the specific downstream task. Another approach is supervised fine-tuning, where the model is fine-tuned on a task-specific dataset with labeled examples. This method is straightforward and effective when sufficient labeled data is available. Reinforcement learning (RL) can also be used, where the model learns through interaction with an environment, receiving rewards based on its predictions [85].

In our implementation, we use self-supervised fine-tuning (❸). This approach is in line with current practices in the field where fine-tuning pre-trained language models without architectural modifications is standard for adapting these models to diverse tasks [69]. The pre-trained LLM is fine-tuned to adapt its parameters to the distinct characteristics of the job data. The prompts in this context serve as the initial input that the model processes to produce outputs that match the desired characteristics or patterns of the job data. Training parameters such as batch size, learning rate, and number of epochs are carefully selected and optimized to achieve convergence and maximize model performance. During training, the model iteratively adjusts its weights based on the loss computed between generated sequences and target sequences, updating all layers of the model to improve its ability to generate synthetic job trace samples that resemble real-world data.

Text Generation

Once trained, the fine-tuned model is ready to generate new synthetic job data (❹). This process begins with providing a prompt to the model, which then autonomously predicts and generates sequences of data points that resemble real job characteristics. In our design, we use a prompt that mirrors the structure and content of the input sequences used during fine-tuning. Various generation parameters such as sampling strategies (e.g., top-k sampling, temperature) are configured to ensure diversity and realism in the generated sequences.

In terms of the prompt variations, we also experiment with three types of additional prompts to enhance the quality of generated samples. The first type is an instructional prompt that explicitly instructs the model to generate synthetic data, but this method led to outputs with nearly identical rows, indicating limited variation and complexity in the generated data. Additionally, attempts to provide statistical information about the input dataset or specific characteristics of jobs (such as arrival and failure patterns) did not significantly improve data quality. Therefore, we maintain the use of only the prompt structured similarly to the input sequences used during fine-tuning. This approach capitalizes on the model's training to accurately reflect the statistical distributions and patterns inherent in real-world job executions, ensuring that the generated job data are both representative and useful for downstream analysis and simulation purposes.

Post-Process

Following the generation step, we perform post-processing (❺) on the generated sequences to produce the final output table. This crucial step involves converting generated sequences into valid data format, denormalizing numerical values, and ensuring that all generated samples adhere to constraints such as non-negativity or integer values where appropriate. This meticulous process guarantees that all the generated samples comply with real-world data standards, enhancing their utility for comprehensive analysis and application scenarios.

In post-processing, there are various approaches beyond basic rounding and ensuring non-negativity. Advanced techniques include outlier detection and removal, data validation against external constraints (e.g., business rules), normalization to a specific range, and incorporating domain-specific heuristics to enhance the realism of generated data.

The post-processing method in our design is a straightforward approach commonly used to finalize generated data for practical applications. Choosing more sophisticated architectures would be justified when additional validation or transformation steps are necessary to meet specific application requirements or when data quality needs to be further enhanced through advanced statistical methods or domain-specific rules.

The structured methodology described above provides a systematic approach to synthesizing job data using LLMs within HPC clusters. Our exploration of textualization strategies contributes to understanding the trade-offs between complexity and performance in preprocessing for LLM-based data synthesis. Our experimentation on various prompts elucidates how prompts influence data diversity and quality in synthetic outputs, offering insights into the effective prompt design to optimize LLM performance in generating realistic data.

3.3 Experiment Setup for Job Data Synthesis

3.3.1 Dataset for Job Data Synthesis

The dataset used for synthesis consists of job data gathered from the Slurm scheduler logs of SURFLisa [86], covering the period from March 1, 2022, to February 28, 2023. Each row in the dataset corresponds to a single job or task execution and includes various metrics related to job performance and resource allocation. These features include the number of CPUs requested, the hour of the day the job was submitted, the running time of jobs, etc. For this work, a subset comprising 10,000 rows and 12 distinctive features is selected. The dataset is fully cleaned with no missing values. The data is divided into training and validation sets using an 80-20 split.

3.3.2 Baseline Models for Synthetic Data Evaluation

To effectively evaluate the effectiveness of LLMs in synthesizing job data, we select 2 models based on Generative Adversarial Networks (GAN) [87], Tabular GAN (TabGAN) [62] and Conditional Tabular GAN (CTGAN) [37], to serve as benchmarks.

TabGAN uses GANs to generate synthetic datasets that maintain the statistical properties of the original data. TabGAN consists of a generator and a discriminator. The generator focuses on producing realistic tabular data, while the discriminator learns to differentiate between real and synthesized data. In our experiment, the TabGAN is trained with 100 epochs, 500 batches, and a 0.0002 learning rate for both the generator and discriminator.

CTGAN is also designed to model tabular data, using conditional generation to capture the distribution of each feature conditioned on other features. Similar to TabGAN, CTGAN includes a generator and a discriminator network and is trained with the same setup.

3.3.3 Evaluation Metrics for Synthetic Data

Synthetic data must undergo rigorous validation to assess their fidelity and utility. This process includes statistical comparisons with the input dataset, ensuring that the distribution of features is realistic, and evaluating the performance of downstream tasks (e.g., predictive modeling) using synthetic data. These measures are crucial to ensure that the synthesized data retains the essential characteristics and variability observed in real-world HPC workloads.

In this study, the evaluation of LLMs built for the generation of synthetic job data focuses on two dimensions. First, we assess the similarity between the synthetic and the original input datasets using distance to closest record (DCR) [36]. Furthermore, we evaluate the utility of synthetic data by comparing the performance of machine learning models trained on synthesized data to those trained on real data, using Machine Learning Efficiency (MLE) metrics [37].

Distance to Closest Record

DCR measures how similar the generated synthetic records are to the nearest real records in the original dataset. It serves as an indicator of both the diversity and realism of synthetic data. A lower DCR indicates that the synthetic data closely mimic the real data.

The DCR of each synthetic record is calculated by determining its Euclidean distance to the closest real record in the original dataset. While both Euclidean and Manhattan distances are common metrics, Euclidean distance is preferred in this study due to the continuous nature of most features in our dataset. Euclidean distance computes the straight-line distance between two points in space, taking into account both the magnitude and direction of differences. In contrast, the Manhattan distance sums the absolute differences in all dimensions.

To ensure fair comparison and meaningful distance calculations, we standardize the data to handle features with different units and scales. Furthermore, we assign different weights to categorical and numerical features. As the disparity in categorical features is more significant than in numerical features, we set the weight of categorical features as 1.5 and that of numerical features as 1. These weights are applied during the distance calculation to ensure that categorical features contribute more to the overall distance, reflecting their significance in the job dataset. Besides, to establish a good DCR threshold, we consider a 5% error of numerical features and a 1% error of categorical features to be the benchmark. The threshold of the dataset used here is 0.39.

Machine Learning Efficiency

MLE assesses the comparative performance of machine learning models trained on synthetic data versus those trained on real data. MLE helps determine the utility of synthetic data for predictive modeling tasks. A high MLE indicates that synthetic data can effectively replace real data for training models without significant loss of performance.

In our study, we conduct binary classification on the termination state of each job using the Decision Tree (DT) and Random Forest (RF) models. We evaluate the performance of these 2 models on the holdout test set derived from the original dataset. To capture variability in model performance, each synthetic dataset undergoes 50 iterations of evaluation, with the Area Under the ROC Curve (AUC-ROC) serving as the primary metric for assessing model accuracy and reliability.

3.4 Evaluation of Synthetic Datasets

3.4.1 Evaluation of Distance to Closest Record

In Figure 2, we use box plots to visualize the DCR metric with the dashed line indicating the DCR threshold that denotes good synthetic data quality. The box plots underscore the varied performance of different models in generating synthetic data. In Table 2, we report the mean, 10% trimmed mean, and median of DCR for each dataset, with serialized input for RoBERTa-Base, RoBERTa-Large, and GPT-2, and flattened input for BART-Base, BART-Large, BERT-Base, BERT-Large, and DistilBERT-Base.

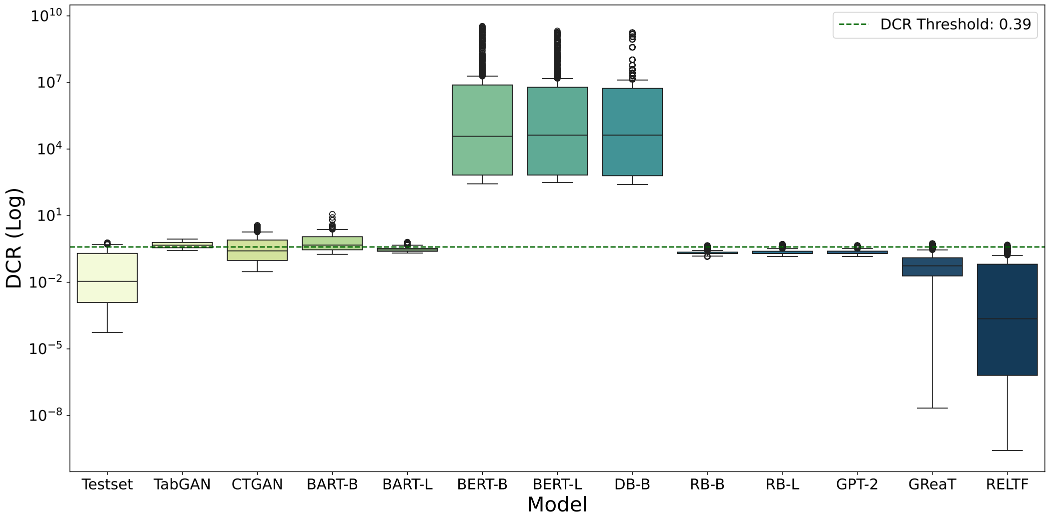


Figure 2: Box plot of distance to closest record (DCR) for original and synthetic datasets (10% trimmed).

The test set derived from the original data serves as a benchmark for comparison, showing a mean DCR of 0.2249, a trimmed mean of 0.1163, and a median of 0.0110, suggesting high fidelity of the synthetic data. The performance of synthetic data generation models varies significantly, with RoBERTa-Large and REaLTabFormer showing the most promise in producing high-quality synthetic data that closely mimics real data. In the following, we discuss the key findings, categorizing the models into GAN-based (baselines), BART-based, BERT-based, and GPT-2-based approaches.

Baselines TabGAN and CTGAN-generated job data have moderate DCR, with trimmed mean DCR of 0.4975 and 0.5323, respectively. However, the mean DCR of CTGAN is sig-

Table 2: DCR mean, trimmed mean, and median for different synthetic datasets. The best two results are in bold.

Model	Mean	Trimmed Mean (10 %)	Median
Test Subset	0.2249	0.1163	0.0110
TabGAN	0.5346	0.4975	0.4660
CTGAN	3.6963	0.5323	0.2600
BART-Base	6.03×10^{34}	0.7847	0.4727
BART-Large	1.28×10^{23}	0.3021	0.2868
BERT-Base	2.00×10^{22}	9.96×10^{7}	3.77×10^{4}
BERT-Large	2.10×10^{22}	6.70×10^{7}	4.24×10^{4}
DistilBERT-Base	1.00×10^{22}	7.20×10^{7}	4.24×10^{4}
RoBERTa-Base	0.3762	0.2225	0.1968
RoBERTa-Large	0.3635	0.2329	0.1968
GPT-2	**0.3427**	0.2327	0.1968
GReaT	0.3868	**0.1008**	**0.0551**
REaLTabFormer	**0.2237**	**0.0553**	**0.0002**

nificantly high at 3.6963, indicating variability in synthetic data quality. In the box plot, the test set shows relatively low DCR values, with most data points below the threshold, confirming its close resemblance to the original data. The box plots for TabGAN and CTGAN display significant variability. CTGAN, in particular, shows a wide range of DCR values with many outliers far above the threshold, indicating inconsistent quality in the synthetic data.

BART-Based-Models Similar to CTGAN, BART-Base and BART-Large exhibit extremely high mean DCR (6.03×10^{34} and 1.28×10^{23}) due to outliers. The presence of extreme values poses a potential challenge to the utility of synthetic data, which is proved by the poor AUC-ROC scores as shown in Section 3.4.3 later. Both BART-Base and BART-Large have box plots with extremely high upper bounds, reflecting the presence of substantial outliers. This supports the observation of high mean DCR values due to extreme synthetic records.

BERT Based Models BERT-based models exhibit diverse performance. Datasets synthesized by BERT-Base, BERT-Large, and DistilBERT-Base have extremely high mean DCR, indicating that these LLMs fail to capture the characteristics of real data. However, RoBERTa-Base and RoBERTa-Large show much lower DCR mean and trimmed mean that outperform the baselines, suggesting high-quality synthetic data closely resembling real data. The box plots for BERT-Base, BERT-Large, and DistilBERT-Base exhibit large ranges and high median values, showing poor performance. In contrast, RoBERTa-Base and RoBERTa-Large have lower and more consistent DCR values, with the majority of data points below the threshold.

GPT-2-Based Models GPT-2-based models show promising results with low DCR values. The dataset generated by GPT-2 shows moderate similarity to real data with a trimmed mean of 0.2327 and a median of 0.1968. GReaT synthesized data has a mean DCR of 0.3868 with a much lower trimmed mean (0.1008) and median (0.0551), showing a mix of close and far synthetic records, indicating variability in quality. REaLTabFormer data achieves close similarity to real data, with a mean of 0.2237 and a trimmed mean of 0.0553. The GPT-2 and GReaT box plots display relatively low DCR values, with medians below the threshold. REaLTabFormer, in particular, shows very low DCR values, indicating excellent synthetic data quality. One possible concern for the extremely low DCR value of the data generated by REaLTabFormer is that some traces of the data, although similar to the real data, have low availability. In the next section, we will employ MLE to look into this issue.

For a comprehensive comparison, we visualize the DCR distribution for each synthesized dataset in Figure 3. Most models show a sharp peak at lower DCR values, indicating a close similarity to the real data, except for BART-Base and BART-Large, which exhibit extremely large DCR values on a logarithmic scale. The significant variability and outliers are not as pronounced as in previous work [50, 65, 66], suggesting potential issues with the stability of the model. Furthermore, we observe that DCR distributions appear to differ from dataset to dataset [50], suggesting that the nature of the dataset has a significant impact on DCR values and, by extension, on the quality of the synthetic data generated by the models.

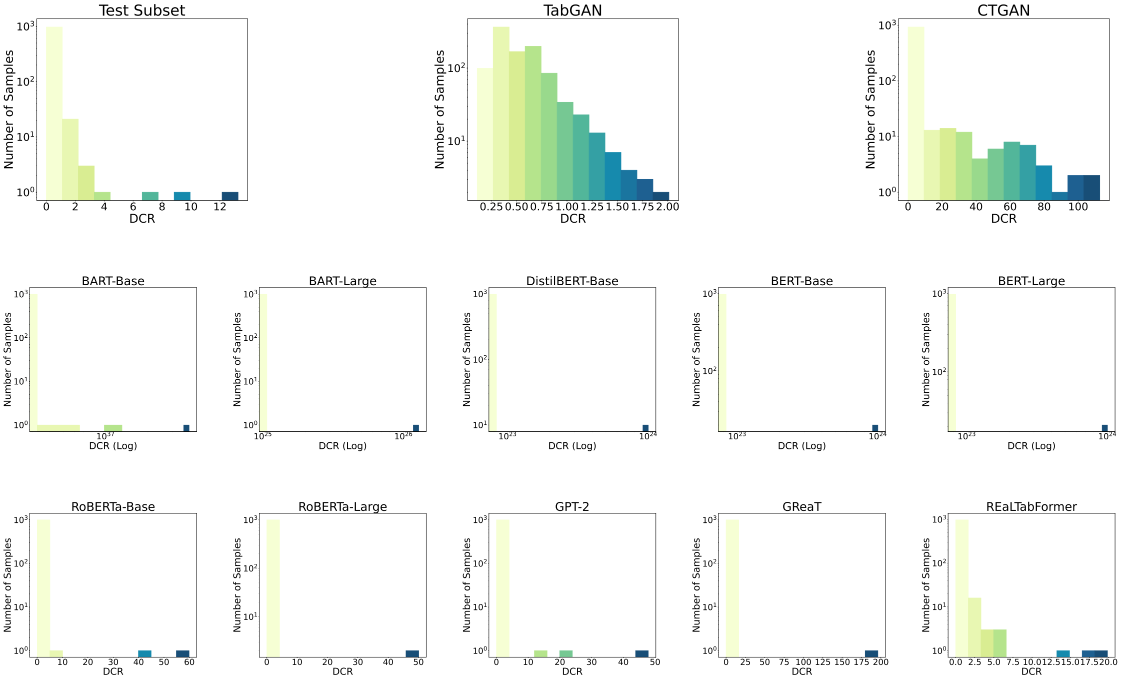


Figure 3: Distance to closest record (DCR) distributions for original and synthetic datasets.

The DCR values across different models and datasets reveal significant variability. Models such as RoBERTa-Large and REaLTabFormer consistently yield lower DCR values, which signifies their capacity to generate high-quality synthetic data that closely resembles the actual dataset. This fidelity is crucial for reliable simulations and analyses in environments where accurate data representation is critical. Conversely, models such as BART-Base and BART-Large exhibit extremely high DCR mean due to the presence of outliers, which compromises the utility of the generated synthetic data. Besides, the results also underscore the sensitivity of DCR to the underlying characteristics of the dataset. This observation suggests that the choice of a generative model should be tailored to the specific traits of the dataset to optimize data synthesis quality.

3.4.2 Impact of Textualization Techniques on DCR

Table 3 shows the DCR values of GPT-2 and variants of RoBERTa using flattened input while Table 4 shows the DCR values of variants of BART and BERT models using serialized inputs.

Table 3 illustrates a stark degradation in DCR metrics compared to their serialized counterparts. DCR values skyrocketed to the magnitudes of 10^6 to 10^{10}, clearly demonstrating the ineffectiveness of the flattened approach for GPT-2 and variants of RoBERTa. In Table 4, the DCR values reach as high as 10^{36} for BART-Base, underscoring the severe unsuitability of serialized inputs for these models under tested conditions.

Table 3: DCR mean, trimmed mean, and median for datasets synthesized by RoBERTa-Base, RoBERTa-Large, and GPT-2 with flattened input.

Model	Mean	Trimmed Mean (10 %)	Median
RoBERTa-Base	6.54×10^{11}	6.49×10^{8}	1.06×10^{7}
RoBERTa-Large	1.12×10^{10}	6.29×10^{8}	1.13×10^{7}
GPT-2	1.07×10^{11}	5.23×10^{8}	9.79×10^{6}

Table 4: DCR mean, trimmed mean, and median for datasets synthesized by BART-BASE, BART-Large, BERT-Base, BERT-Large, and DistilBERT-Base with serialized input.

Model	Mean	Trimmed Mean (10 %)	Median
BART-Base	5.92×10^{36}	1.06×10^{28}	8.59×10^{17}
BART-Large	1.60×10^{26}	19.8920	5.6566
BERT-Base	4.67×10^{22}	4.92×10^{21}	3.06×10^{21}
BERT-Large	2.19×10^{23}	5.74×10^{21}	4.71×10^{21}
DistilBERT-Base	8.77×10^{11}	3.00×10^{8}	2.07×10^{6}

The evaluation suggests that the effectiveness of textualization techniques is highly model-specific. While serialized inputs benefit models such as RoBERTa and GPT-2 by aligning well with their architecture and enhancing data fidelity, they prove detrimental for BART and BERT models, likely due to their different processing capabilities or architectural constraints. Conversely, flattened inputs, though simpler and faster to implement, resulted in poor performance for RoBERTa and GPT-2, indicating a loss of critical contextual information.

These findings highlight the importance of selecting appropriate preprocessing strategies tailored to the specific characteristics and strengths of the chosen LLMs. Such alignment is crucial not only for optimizing synthetic data quality but also for enhancing the overall effectiveness of the models in practical high-performance computing applications. This strategic approach ensures that textualization methods enhance, rather than hinder, the capability of LLMs to process and synthesize high-quality data.

3.4.3 Evaluation of Machine Learning Efficiency

Table 5 presents the MLE results from our experiments, and Figure 4 offers visualizations of each metric. RoBERTa-Base is omitted from MLE analysis as it generates synthetic datasets with uniform target values.

Models trained on the original dataset exhibit high performance, setting a benchmark for comparing synthetic datasets. The Decision Tree and Random Forest models achieve nearly perfect performance, with both models showing MLE above 0.94. These results indicate that the real data provides a reliable basis for predictive modeling.

The performance of models trained on synthetic datasets varies significantly. As in DCR evaluation, we discuss the models by category.

Baselines and BART-Based Models Both CTGAN and TabGAN, along with BART-based models, exhibit markedly lower MLE performance metrics compared to the original dataset, underscoring challenges in capturing the complexity of operational data within HPC environments.

BERT-Based Models BERT-based models exhibit a wide range of performance. BERT-Base and BERT-Large synthetic datasets resulted in low performance, with both Decision Tree and Random Forest models showing AUC-ROC values below 0.50 and accuracies of 0.34 and 0.38. DistilBERT-Base produces the lowest performance. These results are consistent with the extremely high DCR values discussed in Section 3.4.1. In contrast, the dataset synthesized by RoBERTa-Large shows the highest performance among the synthetic datasets, largely outperforming the baselines. The Decision Tree achieves an AUC-ROC of 0.74, and the Random Forest achieves an AUC-ROC of 0.82. This indicates that RoBERTa-Large is effective in generating synthetic data that closely matches the real data distribution.

GPT-2-Based Models Synthetic datasets generated by GPT-2 and 2 off-the-shelf GPT-2-based models also provide moderate performance. The Random Forest trained on GPT-2 data achieves an AUC-ROC of 0.76. GReaT and REaLTabFormer synthetic data show similar results. Though falling short compared to real data, GPT-2-based models demonstrate the potential of LLMs, highlighting the need for further improvements.

Contrary to previous studies [65, 64, 66, 13] in which LLM-generated datasets often outperformed baselines and even real data, our findings do not mirror these outcomes. The discrepancy may stem from the specific characteristics of the HPC datasets and the complexity of the job states modeled in our experiments. These factors could influence the synthetic data's ability to accurately mimic real data complexities, as reflected in our

Table 5: MLE evaluated on the original and synthetic datasets by Decision Tree (DT) and Random Forest (RF) models. The best results of each model are in bold.

Datasets	Models	AUC	Accuracy	Precision	Recall	F1-Score
Input	DT	0.99	0.94	0.94	0.94	0.94
	RF	0.99	0.95	0.96	0.95	0.96
CTGAN	DT	0.53	0.69	0.64	0.69	0.65
	RF	0.55	0.68	0.56	0.68	0.60
TabGAN	DT	0.48	0.70	0.57	0.70	0.61
	RF	0.48	0.70	0.57	0.70	0.61
BART-Base	DT	0.50	0.68	0.62	0.68	0.63
	RF	0.54	0.74	0.70	0.74	0.67
BART-Large	DT	0.49	0.70	0.59	0.70	0.62
	RF	0.46	0.64	0.60	0.64	0.62
BERT-Base	DT	0.48	0.34	0.57	0.34	0.31
	RF	0.47	0.34	0.57	0.34	0.31
BERT-Large	DT	0.48	0.38	0.58	0.38	0.38
	RF	0.47	0.38	0.58	0.38	0.38
DistilBERT-Base	DT	0.50	0.34	0.61	0.34	0.28
	RF	0.55	0.27	0.80	0.27	0.11
RoBERTa-Large	DT	**0.74**	**0.78**	**0.78**	**0.78**	**0.78**
	RF	**0.82**	**0.79**	**0.80**	**0.79**	**0.79**
GPT-2	DT	0.63	0.71	0.72	0.71	0.72
	RF	0.76	0.72	0.72	0.72	0.72
GReaT	DT	0.65	0.70	0.70	0.70	0.70
	RF	0.72	0.70	0.70	0.70	0.70
REaLTabFormer	DT	0.65	0.72	0.73	0.72	0.72
	RF	0.69	0.68	0.69	0.68	0.68

MLE assessments. Besides, performance variability from dataset to dataset, as observed in previous works, likely contributes to the less satisfactory MLE scores in our experiments. The diverse nature of data in different studies and the unique challenges presented by HPC environments might explain the inconsistency in model effectiveness across different implementations.

Although RoBERTa variants and GPT-2-based models outperform baselines, none of the synthetic datasets reached the performance levels of models trained on real data. This gap underscores the ongoing challenge and necessity for advancing synthetic data generation techniques to achieve better fidelity and utility in HPC and other demanding fields.

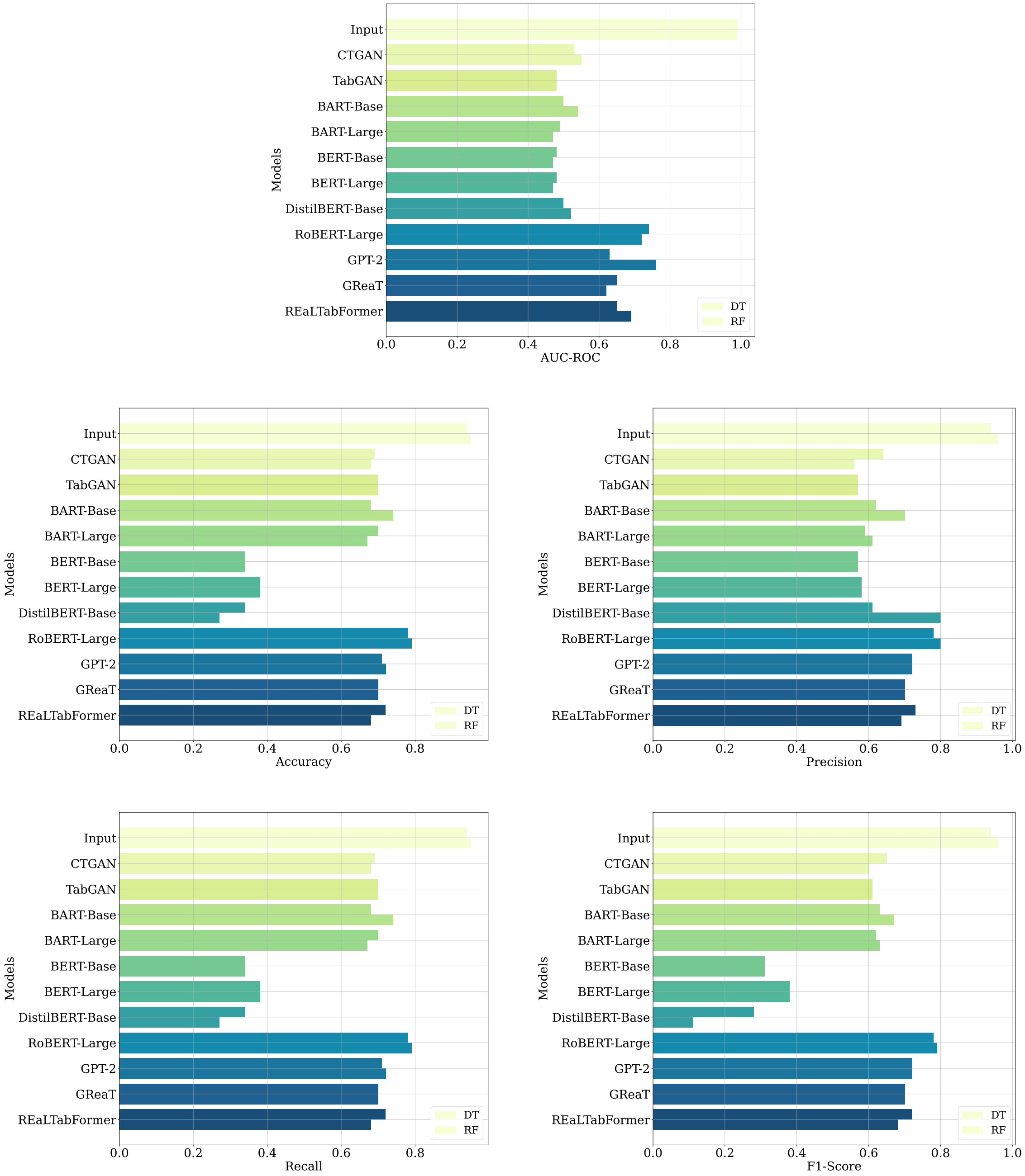


Figure 4: MLE comparison between traditional models and LLMs.

3.5 Summary of Job Data Synthesis Using LLMs

This chapter has systematically explored the capabilities of LLMs in generating high-fidelity synthetic job data, aiming to create high-fidelity datasets that mimic real operational data to enhance the efficiency and effectiveness of HPC systems. Below we outline the main findings of our design and evaluation.

F1 The choice of textualization technique is crucial for optimizing synthetic data quality. Textual encoding methods, such as the natural language serialization technique proposed by Borisov et al. [65], enhance the performance of models such as RoBERTa and GPT-2, while resulting in poor performance for BART and BERT models.

F2 RoBERTa-Large and GPT-2-based models demonstrate the most promise with low DCR values, indicating high-quality synthetic data.

F3 BART-based and some BERT-based models show extremely high DCR due to outliers, compromising data utility.

F4 When used to train models for predictive tasks, synthetic datasets show varied performance. RoBERTa-Large and GPT-2-based models outperform baselines but do not reach the performance levels of models trained on real data. This highlights the need for further improvements in synthetic data generation techniques.

F5 Unlike previous studies where LLM-generated datasets often outperformed baselines and real data, this study finds that the specific characteristics of HPC datasets and job states may influence the synthetic data's ability to mimic real data complexities, underscoring the ongoing challenges in achieving better fidelity and utility.

The insights gained point to several areas for further development. First, the nuanced approach to model tuning and data handling must be refined. The process involves not only selecting the appropriate model and textualization technique but also continuously optimizing the training and generation processes to enhance the realism and applicability of the synthetic data. Moreover, the exploration highlights the need for advancements in how synthetic data is post-processed and validated against external benchmarks. This step is critical to ensure that the generated data not only mimics the real data in terms of numbers and patterns but also behaves similarly under various operational scenarios, thereby truly supporting the development and testing of HPC systems.

In conclusion, while models such as RoBERTa-Large and REaLTabFormer have set promising benchmarks in synthetic data generation, the full potential of LLMs in this domain remains untapped. Future research should focus on closing the gap between the fidelity of synthetic data and its utility, pushing the boundaries of how synthetic datasets can support complex, data-driven decision-making in HPC and beyond. This ongoing endeavor will require a concerted effort to refine data synthesis techniques, improve model training methodologies, and develop robust frameworks for evaluating synthetic data in diverse application areas.

4 Design and Evaluation of LLMs for Job End-State Prediction

4.1 Overview of Job End-State Prediction Using LLMs

In this chapter, we focus on addressing RQ3 of using LLMs to predict job termination states in HPC clusters. We detail our experimental setup and evaluate the performance of LLMs in classifying job end-states, comparing their efficacy with traditional machine learning models.

We begin with discussing the design requirements and choices in Section 4.2, where we digest the four key stages of the design: preprocessing, data pipeline, model fine-tuning, and performance evaluation. Notably, we integrate extra layers with pre-trained LLMs specifically tailored for handling both categorical and numerical features, crucial for the classification of job end-states.

Section 4.3 details the dataset and baseline models used for comparison. The dataset focuses on job submission metrics and is balanced using SMOTETomek. Three baseline models—Logistic Regression, Random Forest, and XGBoost—are employed to provide benchmarks for performance comparison.

Finally, we evaluate the performance of the models using the AUC-ROC metric in Section 4.4. The evaluation sheds light on how well LLMs, particularly GPT-2 and variants of RoBERTa, perform against traditional predictive models in classifying end-states of jobs within HPC environments. This assessment not only highlights the potential of LLMs in this field but also identifies areas where further tuning and model adjustments are necessary to enhance their predictive accuracy and reliability in real-world applications.

4.2 Methodology for LLM-Based Job End-State Prediction

To explore LLMs' ability in classification, we fine-tune eight pre-trained LLMs. In addition to the six pre-trained LLMs outlined in Section 3.2.1 (GPT-2, DistilBERT-Base, BERT-Base, BERT-Large, RoBERTa-Base, and RoBERTa-Large), we also use two variants of ALBERT [88], which modify BERT's architecture to increase training speed and decrease model size, proving beneficial for scaling to larger datasets and models. The following are the requirements and choices for our design.

4.2.1 Design Requirements for Job End-State Prediction Using LLMs

The following requirements are essential to utilize LLMs for the classification task.

Data Integrity and Quality Ensuring that the dataset is properly prepared and cleaned for accurate model training and evaluation.

Consistency and Efficiency Implementing processes that standardize and streamline data handling to maintain consistent input dimensions and efficient model training.

Flexibility Designing the model architecture to handle both categorical and numerical features effectively.

Model Performance Ensuring that the models are fine-tuned and optimized to achieve high predictive accuracy.

Evaluation and Validation Rigorously evaluating the models to ensure their predictive capabilities are robust and reliable.

4.2.2 Design Choices for Job End-State Prediction Using LLMs

Similar to synthetic models, the process can be generalized into 4 stages as shown in Figure 5. Below we present details of each step.

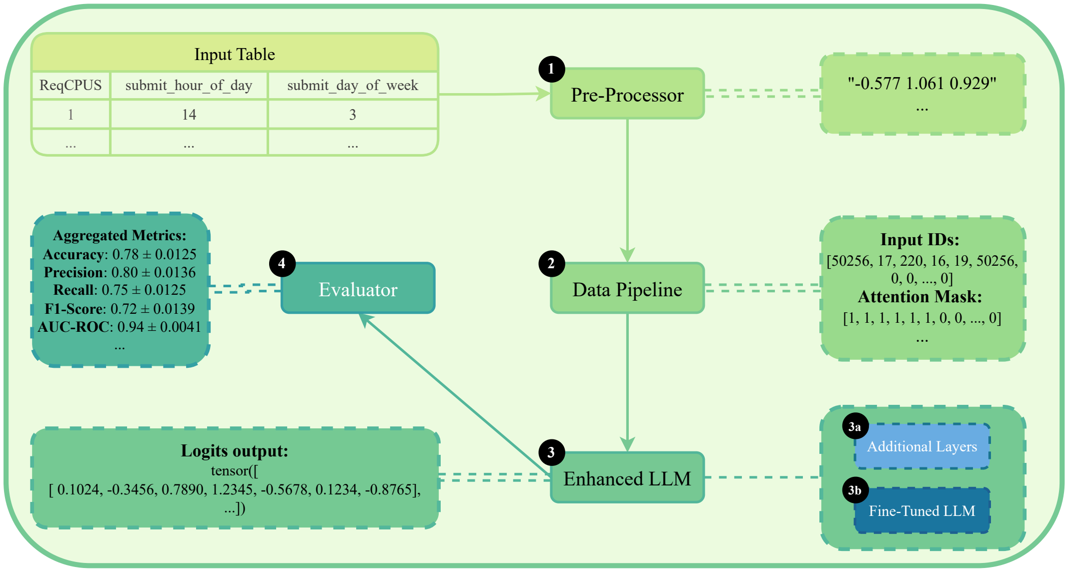


Figure 5: A high-level illustration of utilizing pre-trained LLMs for job end-state prediction.

Preprocessing

The preprocessing phase (❶) focuses on preparing the dataset and ensuring its suitability for subsequent model training. As the target feature is highly imbalanced, we start with resampling the original dataset using SMOTETomek, which balances the dataset by combining synthetic minority oversampling with majority class undersampling (details in Section 4.3.1). For numerical features, we perform standard scaling to ensure equal contribution during model training.

Following scaling, numerical features are converted into tensors suitable for input into LLMs. Each row of the dataset is then transformed into a string representation, where each feature value is concatenated with a separator. This transformation is essential as LLMs typically operate on textual data, requiring tabular data to be converted into a format they can interpret.

There are various ways to convert tabular data into text formats for LLM input [27]. Common approaches include using formats such as line-separated JSON [89] or transforming tables into human-readable sentences by creating templates based on column headers and cell values, as discussed in Section 3.2.3.

Our chosen method of converting rows into concatenated strings strikes a practical balance between information richness, simplicity, and efficiency. For prediction tasks, particularly in classification, the primary goal is to classify instances based on discriminative features. Naive preprocessing ensures that relevant features are preserved and presented to the model in a format it can effectively interpret. This straightforward approach mitigates the risk of introducing noise or irrelevant data that could potentially confuse the model or compromise classification performance.

Data Pipeline

The data pipeline (❷) manages the preparation and flow of data for training and evaluation. Similar to job data synthesis, the processed input is tokenized by corresponding tokenizers, which pad and truncate the sequences to a fixed maximum length, ensuring consistency in input dimensions during training. In addition, other tokenization strategies exist, such as dynamic padding based on the longest sequence in a batch. Our fixed-length approach guarantees consistency in input dimensions across all samples during training.

We create custom datasets and data loaders. These datasets combine the scaled numerical features, tokenized text data, and corresponding labels. This integration facilitates seamless data handling during the training and evaluation phases. The data loaders are configured to handle batching, shuffling, and sampling of the data efficiently. For the GPT-2-based model, we define a custom dataset class that concatenates numerical features into a single text string. For other models, we define a similar dataset class, but retain the numerical features as separate inputs alongside the tokenized text data. This adaptability is crucial in research and application scenarios where different models may require varying data representations for optimal performance.

Additional Layers

For classification tasks, additional layers (3a) are integrated with pre-trained LLMs (3b). The GPT-2-based model is modified to include a classification layer that outputs logits corresponding to different termination states of jobs. The model processes the tokenized input sequences and generates hidden states from which the final token's output is used for classification. Other models are extended with additional layers to handle numerical features in conjunction with text-based features. This involves incorporating fully connected layers to process numerical data, followed by concatenating this output with pooled representations from the LLMs. Subsequently, the combined representation is fed into a final classification layer.

This approach streamlines model architecture and training protocols, effectively merging the strengths of LLMs with conventional neural network components. However, alternative architectures may explore different types or depths of neural network layers instead of directly adding fully connected layers post-LLMs. For example, utilizing convolutional layers or recurrent neural networks (RNNs) alongside LLMs could capture different hierarchical or temporal aspects of data, especially useful for sequential or time-series classification tasks.

Fine-Tuning

The combined models (❸) comprising LLMs and additional layers with numerical features are fine-tuned using a cross-entropy loss function and the AdamW optimizer. Drawing from the fine-tuning methodology in job data synthesis, we use supervised fine-tuning on all layers. The training procedure spans multiple epochs, where the model iteratively adjusts its weights to minimize classification errors and enhance predictive accuracy.

Evaluation

After training, the models are evaluated on the validation set to measure their performance (❹). This evaluation process involves multiple iterations of training and assessment in various random splits of the dataset to ensure the robustness and reliability of the results. In alignment with established benchmarks [71, 9], we prioritize AUC-ROC as the primary metric for evaluating the classification capabilities of the models. AUC-ROC is particularly suitable for assessing the model's discriminative power across different classes, which is advantageous in scenarios with class imbalance.

While AUC-ROC is a common metric for evaluating classification models, other metrics could also be considered depending on the specific requirements of the task. Metrics including accuracy, recall, and F1-score are informative for understanding the model's performance across different classes, especially in scenarios where specific class performance matters.

The framework presented above not only streamlines model training, but also underscores adaptability across diverse research and application domains, especially with the incorporation of additional layers and pre-trained LLMs.

4.3 Experiment Setup for Job End-State Prediction

4.3.1 Dataset for Job End-State Prediction

For building predictive LLMs, we use a subset of the dataset described in Section 3.3.1. In job scheduling, certain job features, such as allocated numbers of CPUs and running time, are unknown before job completion. Therefore, we focus only on metrics available at job submission. The selected features include the number of requested CPUs, the hour of the day, the day of the week, and the day of the month when the job was submitted. The target column is the encoded termination state of each job. There are 6 types of end-states for jobs in the dataset under examination, including completed, canceled, failed, timeout, out of memory, and node fail.

A significant challenge in this dataset is the class imbalance in the target column, where some job termination states are much less frequent than others. To address this, we employ the SMOTE (Synthetic Minority Over-sampling Technique) combined with Tomek links (SMOTETomek) to balance the classes. SMOTE generates synthetic samples for the minority class, whereas Tomek links help clean the data by removing overlapping samples from different classes. The dataset is then split into training and validation sets using an 80-20 split to ensure robust evaluation of the models.

4.3.2 Baseline Models for Predictive LLMs Evaluation

To predict the termination states of HPC jobs, we employed three baseline models, Logistic Regression, Random Forest, and XGBoost [38]. These models are chosen for their ability to handle classification tasks effectively and their prevalence as a benchmark in predictive modeling scenarios. We deploy the random search algorithm to fine-tune the hyperparameters of the three benchmarks.

Logistic Regression is a fundamental model for classification tasks and serves as a baseline due to its simplicity and interpretability. Parameters optimized include regularization strength, penalty, solver, and maximum iterations.

Random Forest is an ensemble learning method known for its robustness, ability to handle complex interactions in data, and resistance to overfitting. Hyperparameters such as the number of trees and maximum depth of trees are optimized during training.

XGBoost is an optimized gradient boosting algorithm renowned for its speed and performance in structured data, often outperforming other models in various competitions and benchmarks. Key hyperparameters tuned include the number of boosting rounds, maximum tree depth, learning rate, and subsampling ratio.

4.4 Evaluation of Predictive LLMs

Our objective is to investigate whether LLMs, known for their prowess in natural language processing, can compete with traditional machine learning models in handling structured and numerical data. We primarily use the AUC-ROC as the performance metric for this evaluation, consistent with previous studies [72].

Table 6 presents a detailed comparison of the AUC-ROC scores between traditional models (Logistic Regression, Random Forest, and XGBoost) and LLMs (GPT-2 and variants of BERT, DistilBERT, ALBERT, and RoBERTa) for the task of classification of the end-state of the job in HPC clusters. The visualization of the performance of the models can be found in Figure 6.

Table 6: AUC-ROC performance comparison between traditional models and LLMs. The best three results are in bold.

Models	AUC-ROC
Logistic Regression	0.62 ± 0.0038
Random Forest	**0.96** ± 0.0010
XGBoost	**0.96** ± 0.0011
BERT-Base	0.78 ± 0.0042
BERT-Large	0.75 ± 0.0014
DistilBERT-Base	0.82 ± 0.0020
ALBERT-Base	0.80 ± 0.0019
ALBERT-Large	0.77 ± 0.0099
RoBERTa-Base	0.81 ± 0.0051
RoBERTa-Large	0.83 ± 0.0002
GPT-2	**0.94** ± 0.0041

Bselines Both Random Forest and XGBoost models demonstrate superior performance with an AUC-ROC of 0.96. This high performance can be attributed to their ensemble learning methods, which effectively capture complex data patterns and interactions, making them particularly well-suited for structured, numerical datasets such as the dataset used in this work. Logistic Regression, while simpler and more interpretable, lags significantly behind with an AUC-ROC of 0.62. This indicates that linear models may struggle with the non-linear relationships and complexities present in the HPC job dataset. Although its simplicity and interpretability are advantageous, these benefits come at the cost of lower predictive power in more complex scenarios.

GPT-2 Among the LLMs, GPT-2 achieves the highest AUC-ROC of 0.94, making it competitive with Random Forest and XGBoost. This suggests that despite being originally designed for NLP tasks, GPT-2 can effectively adapt to structured data with proper

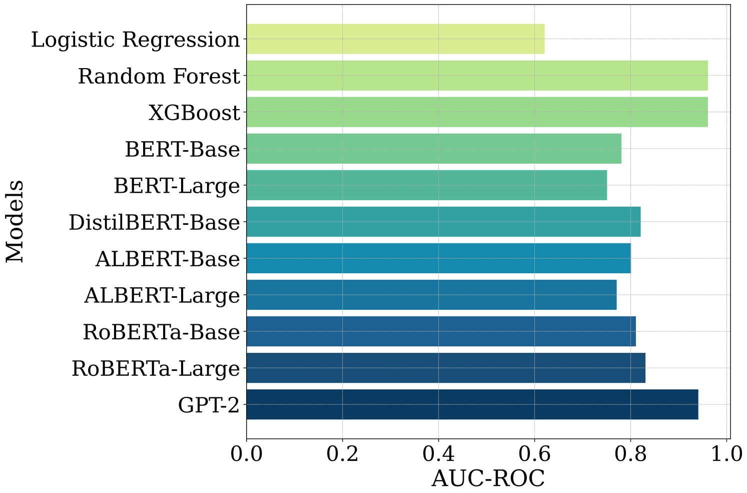


Figure 6: AUC-ROC performance comparison between traditional models and LLMs.

training and tuning.

BERT-Based Models DistilBERT-Base and RoBERTa-Large also show promising results with AUC-ROC scores of 0.82 and 0.83, respectively. These models benefit from their transformer-based architectures, which allow them to capture complex patterns in the data. Other variants of BERT and ALBERT, while outperforming Logistic Regression, do not match the performance of the top traditional models or GPT-2.

The variance in performance across different LLMs indicates that model architecture and pre-training significantly influence their ability to handle structured data. The variability in the AUC-ROC scores among LLMs (e.g., ALBERT-Large's higher standard deviation of 0.0099) suggests that some LLMs may require more fine-tuning and careful hyperparameter optimization to achieve consistent performance. In contrast, traditional models, especially Random Forest and XGBoost, exhibited very low variability, reflecting their stability and reliability in handling the given dataset.

The inferior performance of certain LLMs is consistent with findings in previous works [9, 71], where LLM-based models (LLaMA-GTL and TabLLM) fail to outperform baselines in most of the benchmark datasets in fine-tuned settings. Transformation of tabular data into a textual format suitable for LLMs might not preserve all relevant information as effectively as traditional models designed for numerical data. The fine-tuning process for LLMs might require more sophisticated techniques and additional hyperparameter optimization to match the performance of traditional models.

4.5 Summary of Job End-State Prediction Using LLMs

In this chapter, we utilize LLMs to predict the termination states of jobs and benchmark their performance against traditional machine learning models. We summarise the main findings obtained as follows.

F1 GPT-2 stands out among the LLMs, showing competitive performance against the best-performing traditional models such as Random Forest and XGBoost. This underscores the adaptability of GPT-2 to structured data, despite its original design for text-based tasks.

F2 RoBERTa-Large and DistilBERT-Base also show promising results. However, other variants of BERT and ALBERT outperform Logistic Regression but do not achieve the same level of performance as the top traditional models or GPT-2. This indicates that while LLMs have potential, their effectiveness varies significantly based on the specific model and the task at hand.

F3 The integration of additional layers with pre-trained LLMs enhances their capability to handle both categorical and numerical features effectively.

F4 LLMs demonstrate significant potential in predicting job termination states within HPC clusters. Despite being originally designed for natural language processing tasks, LLMs can be effectively adapted for structured and numerical data with appropriate preprocessing and fine-tuning.

While traditional models remain preferred for structured numerical tasks, the promising results from GPT-2 suggest that LLMs, with further refinement and appropriate training, could broaden their applicability beyond natural language tasks. Future research might explore more sophisticated fine-tuning approaches or hybrid models that integrate the strengths of traditional machine learning and modern LLM techniques.

5 Conclusion and Future Work

Our study embarks on exploring the application of LLMs in HPC domains, particularly focusing on their capabilities in job data synthesis and job end-state prediction.

5.1 Conclusion for Each Research Question

Our study addresses the three key research questions by providing insights into the current state and potential future directions of the use of LLMs in HPC environments.

RQ1: **What are the existing applications of LLMs for HPC?** We provide a comprehensive overview of the existing applications of LLMs and identify the remaining gaps in the field. While there is a growing body of work exploring how LLMs can optimize and enhance HPC operations, existing work focus on programming-related tasks within HPC. Our comprehensive literature review has highlighted the promising capabilities of LLMs in areas such as workload synthesis and predictive analysis but also points out significant gaps and challenges that remain, including scalability and data availability.

RQ2: **How to design and evaluate LLM-based models to synthesize job data in HPC clusters?** The investigation into generating synthetic job data in HPC clusters using LLMs has revealed both advantages and challenges. Our study uses eight general-purpose LLMs and two table-generation LLMs to create synthetic datasets that mimic real operational data. RoBERTa-Large, GPT-2, and REaLTabFormer particularly excel in capturing complex patterns and variability in job execution data, showing low DCR values. However, MLE evaluations indicate that while the LLM-synthesized data shows promise, it does not yet match the utility of real datasets for training models in predictive tasks.

RQ3: **How to design and evaluate LLM-based models to predict the end-state of the job in HPC clusters?** Our research on the application of LLMs to predict end-states of jobs in HPC clusters reveals significant potential and notable challenges. Among the evaluated models, GPT-2 demonstrates competitive performance, closely approaching the results of traditional machine learning models such as Random Forest and XGBoost. This indicates that LLMs, particularly GPT-2, can adapt effectively to structured data when appropriately fine-tuned. However, other LLMs, including variants of BERT and ALBERT, did not perform as well as the traditional models, highlighting the need for more specialized adaptation techniques. While traditional models, known for their robustness and efficiency in handling structured data, still lead in performance, the promising results from certain LLMs suggest the potential for broader applicability with further refinement and specialized tuning.

5.2 Discussion of Limitations for Each Research Question

In the following, we provide a critical reflection on the constraints and potential areas for improvement in addressing each research question.

Limitatioins on RQ1 Although we adopt a systematic approach to address RQ1, our literature review has several limitations. The fast pace of advancements in both LLMs and HPC technologies means that our findings may quickly become outdated, with new applications and innovations emerging that were not captured within the timeframe of our review. Besides, the interdisciplinary nature of the intersection of LLMs and HPC spans multiple disciplines, including machine learning and domain-specific applications. This makes it challenging to fully capture all relevant studies and contextualize them within a unified framework.

Limitatioins on RQ2 The methodology employed in this study to address the generation of synthetic job data using LLMs has several limitations. Firstly, the choice of LLMs and their configurations might not have been exhaustive, and there could be other models or architectures that perform better in this context. Additionally, the evaluation metrics, DCR and MLE, while effective, may not capture all aspects of data fidelity and utility. Other metrics or evaluation techniques might provide additional insights into the quality of synthetic data. Furthermore, the reliance on a specific dataset from SURFLisa may limit the generalizability of the findings. The performance of LLMs in generating synthetic data could vary with different datasets or under different conditions [50].

Limitatioins on RQ3 For the predictive task, this study employs a basic prompt engineering strategy of transforming structured numerical data into textual formats, which may not preserve all relevant information as effectively as traditional models designed for such data. This transformation process could lead to the loss of critical nuance, impacting the predictive accuracy. Furthermore, the inherent complexity of LLMs makes them computationally intensive and resource-demanding, which could limit their practicality in real-world HPC environments.

5.3 Directions for Future Research

To address limitations and build on current findings, future research could focus on several key areas. *First*, expanding the range of LLMs and exploring newer architectures could provide better performance and more robust synthetic data generation. *Second*, improving evaluation methodologies by incorporating additional metrics that assess different aspects of data quality and utility will provide a more comprehensive understanding of synthetic data performance. *Third*, developing more advanced techniques to transform tabular data into formats that preserve its inherent structure and relationships will be crucial [72]. *Fourth*, exploring more sophisticated fine-tuning methods and hyperparameter optimization can help improve the performance consistency of LLMs. *Last*, future studies may also explore hybrid models that combine LLMs with traditional machine-learning techniques to leverage the strengths of both approaches.

In conclusion, our study has highlighted both the potential and the challenges of employing LLMs within the HPC domain. Although LLMs demonstrate significant promise in synthesizing and predicting workload data, bridging the gap between their performance and that of traditional models requires ongoing advancements in model architecture, training strategies, and data handling techniques.

References

[1] S. Elmisaoui, I. Kissami, and J. Ghidaglia, "High-performance computing to accelerate large-scale computational fluid dynamics simulations: A comprehensive study," in *International Conference on Advanced Intelligent Systems for Sustainable Development (AI2SD'2023)* (M. Ezziyyani, J. Kacprzyk, and V. E. Balas, eds.), (Cham), pp. 352–360, Springer Nature Switzerland, 2024.

[2] A. Iosup, G. Andreadis, V. van Beek, M. Bijman, E. V. Eyk, M. Neacsu, L. Overweel, S. Talluri, L. Versluis, and M. Visser, "The opendc vision: Towards collaborative datacenter simulation and exploration for everybody," in *16th International Symposium on Parallel and Distributed Computing, ISPDC 2017, Innsbruck, Austria, July 3-6, 2017* (R. Prodan, F. Pop, and R. Mundani, eds.), pp. 85–94, IEEE, 2017.

[3] A. Iosup, A. Uta, L. Versluis, G. Andreadis, E. V. Eyk, T. Hegeman, S. Talluri, V. van Beek, and L. Toader, "Massivizing computer systems: A vision to understand, design, and engineer computer ecosystems through and beyond modern distributed systems," in *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, pp. 1224–1237, IEEE Computer Society, 2018.

[4] A. Iosup, F. Kuipers, A. L. Varbanescu, P. Grosso, A. Trivedi, J. S. Rellermeyer, L. Wang, A. Uta, and F. Regazzoni, "Future computer systems and networking research in the netherlands: A manifesto," *CoRR*, vol. abs/2206.03259, 2022.

[5] P. O. A. Navaux, A. F. Lorenzon, and M. da Silva Serpa, "Challenges in high-performance computing," *J. Braz. Comput. Soc.*, vol. 29, no. 1, pp. 51–62, 2023.

[6] J. Park, X. Huang, and C. Lee, "Analyzing and predicting job failures from HPC system log," *J. Supercomput.*, vol. 80, no. 1, pp. 435–462, 2024.

[7] C. Liu, J. Han, Y. Shang, C. Liu, B. Cheng, and J. Chen, "Predicting of job failure in compute cloud based on online extreme learning machine: A comparative study," *IEEE Access*, vol. 5, pp. 9359–9368, 2017.

[8] J. Gao, H. Wang, and H. Shen, "Task failure prediction in cloud data centers using deep learning," *IEEE Trans. Serv. Comput.*, vol. 15, no. 3, pp. 1411–1422, 2022.

[9] H. Zhang, X. Wen, S. Zheng, W. Xu, and J. Bian, "Towards foundation models for learning on tabular data," *CoRR*, vol. abs/2310.07338, 2023.

[10] M. Gerndt, M. Kondo, B. P. Miller, and T. Patki, "Adaptive resource management for HPC systems (dagstuhl seminar 21441)," *Dagstuhl Reports*, vol. 11, no. 10, pp. 1–19, 2021.

[11] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P. Chen, Y. Liang, Y. Li, S. Pan, and Q. Wen, "Time-llm: Time series forecasting by reprogramming large language models," *CoRR*, vol. abs/2310.01728, 2023.

[12] B. Radharapu, K. Robinson, L. Aroyo, and P. Lahoti, "AART: ai-assisted red-teaming with diverse data generation for new llm-powered applications," in *Proceedings of the*

2023 Conference on Empirical Methods in Natural Language Processing: EMNLP 2023 - Industry Track, Singapore, December 6-10, 2023 (M. Wang and I. Zitouni, eds.), pp. 380–395, Association for Computational Linguistics, 2023.

[13] M. Gulati and P. F. Roysdon, "Tabmt: Generating tabular data with masked transformers," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023* (A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds.), 2023.

[14] Z. Zhao, R. Birke, and L. Y. Chen, "Tabula: Harnessing language models for tabular data synthesis," *CoRR*, vol. abs/2310.12746, 2023.

[15] N. A. W. A. Hamid and B. Singh, *High-Performance Computing Based Operating Systems, Software Dependencies and IoT Integration*, pp. 175–204. Singapore: Springer Nature Singapore, 2024.

[16] D. Nichols, A. Marathe, H. Menon, T. Gamblin, and A. Bhatele, "Modeling parallel programs using large language models," *CoRR*, vol. abs/2306.17281, 2023.

[17] X. Zhou, X. Zhao, and G. Li, "Llm-enhanced data management," *CoRR*, vol. abs/2402.02643, 2024.

[18] L. Chen, N. K. Ahmed, A. Dutta, A. Bhattacharjee, S. Yu, Q. I. Mahmud, W. Abebe, H. Phan, A. Sarkar, B. Butler, N. Hasabnis, G. Oren, V. A. Vo, J. P. Muñoz, T. L. Willke, T. Mattson, and A. Jannesari, "The landscape and challenges of HPC research and llms," *CoRR*, vol. abs/2402.02018, 2024.

[19] L. Chen, P. Lin, T. Vanderbruggen, C. Liao, M. Emani, and B. R. de Supinski, "LM4HPC: towards effective language model application in high-performance computing," in *OpenMP: Advanced Task-Based, Device and Compiler Programming - 19th International Workshop on OpenMP, IWOMP 2023, Bristol, UK, September 13-15, 2023, Proceedings* (S. McIntosh-Smith, M. Klemm, B. R. de Supinski, T. Deakin, and J. Klinkenberg, eds.), vol. 14114 of *Lecture Notes in Computer Science*, pp. 18–33, Springer, 2023.

[20] X. Li, N. Qi, Y. He, and B. McMillan, "Practical resource usage prediction method for large memory jobs in HPC clusters," in *Supercomputing Frontiers - 5th Asian Conference, SCFA 2019, Singapore, March 11-14, 2019, Proceedings* (D. Abramson and B. R. de Supinski, eds.), vol. 11416 of *Lecture Notes in Computer Science*, pp. 1–18, Springer, 2019.

[21] T. Muhammed, R. Mehmood, A. Albeshri, and F. Alsolami, *HPC-Smart Infrastructures: A Review and Outlook on Performance Analysis Methods and Tools*, pp. 427–451. Cham: Springer International Publishing, 2020.

[22] Z. Hou, H. Shen, X. Zhou, J. Gu, Y. Wang, and T. Zhao, "Prediction of job characteristics for intelligent resource allocation in HPC systems: a survey and future directions," *Frontiers Comput. Sci.*, vol. 16, no. 5, p. 165107, 2022.

[23] N. Kumbhare, A. Marathe, A. Akoglu, H. J. Siegel, G. Abdulla, and S. Hariri, “A value-oriented job scheduling approach for power-constrained and oversubscribed HPC systems,” *IEEE Trans. Parallel Distributed Syst.*, vol. 31, no. 6, pp. 1419–1433, 2020.

[24] Z. Zhou, J. Sun, and G. Sun, “Automated HPC workload generation combining statistical modeling and autoregressive analysis,” in *Benchmarking, Measuring, and Optimizing - 15th BenchCouncil International Symposium, Bench 2023, Sanya, China, December 3-5, 2023, Revised Selected Papers* (S. Hunold, B. Xie, and K. Shu, eds.), vol. 14521 of *Lecture Notes in Computer Science*, pp. 153–170, Springer, 2023.

[25] S. Kumari and P. Muthulakshmi, “High-performance computation in big data analytics,” in *Intelligent Systems Design and Applications - 22nd International Conference on Intelligent Systems Design and Applications (ISDA 2022) Held December 12-14, 2022 - Volume 1* (A. Abraham, S. Pllana, G. Casalino, K. Ma, and A. Bajaj, eds.), vol. 646 of *Lecture Notes in Networks and Systems*, pp. 543–553, Springer, 2022.

[26] N. More, M. Galphade, V. B. Nikam, and B. Banerjee, *High-Performance Computing: A Deep Learning Perspective*, pp. 247–268. Cham: Springer International Publishing, 2021.

[27] X. Fang, W. Xu, F. A. Tan, J. Zhang, Z. Hu, Y. Qi, S. Nickleach, D. Socolinsky, S. H. Sengamedu, and C. Faloutsos, “Large language models(llms) on tabular data: Prediction, generation, and understanding - A survey,” *CoRR*, vol. abs/2402.17944, 2024.

[28] K. Kim, C. Lee, J. H. Jung, and W. W. Ro, “Workload synthesis: Generating benchmark workloads from statistical execution profile,” in *2014 IEEE International Symposium on Workload Characterization, IISWC 2014, Raleigh, NC, USA, October 26-28, 2014*, pp. 120–129, IEEE Computer Society, 2014.

[29] A. Rosà, L. Y. Chen, and W. Binder, “Predicting and mitigating jobs failures in big data clusters,” in *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2015, Shenzhen, China, May 4-7, 2015*, pp. 221–230, IEEE Computer Society, 2015.

[30] H. Liu, Z. Li, C. Tan, R. Yang, G. Cao, Z. Liu, and C. Guo, “Predicting GPU failures with high precision under deep learning workloads,” in *Proceedings of the 16th ACM International Conference on Systems and Storage, SYSTOR 2023, Haifa, Israel, June 5-7, 2023* (Y. Moatti, O. Biran, Y. Gilad, and D. Kostic, eds.), pp. 124–135, ACM, 2023.

[31] T. N. T. Asmawi, A. B. Ismail, and J. Shen, “Cloud failure prediction based on traditional machine learning and deep learning,” *J. Cloud Comput.*, vol. 11, p. 47, 2022.

[32] A. Banjongkan, W. Pongsena, N. Kerdprasop, and K. Kerdprasop, “A study of job failure prediction at job submit-state and job start-state in high-performance computing system: Using decision tree algorithms,” *Journal of Advances in Information Technology (JAIT)*, vol. 12, no. 2, pp. 84–92, 2021.

[33] A. Booth, A. Sutton, and D. Papaioannou, *Systematic Approaches to a Successful Literature Review.* Los Angeles: Sage, second edition ed., 2016.

[34] Y. Xiao and M. Watson, "Guidance on conducting a systematic literature review," *Journal of Planning Education and Research*, vol. 39, pp. 93–112, Mar 2019.

[35] SURF, "lisa - surf user knowledge base - surf user knowledge base." `https://servicedesk.surf.nl/wiki/display/WIKI/Lisa`. Accessed: 2024-04-09.

[36] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," *Proc. VLDB Endow.*, vol. 11, no. 10, pp. 1071–1083, 2018.

[37] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 7333–7343, 2019.

[38] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016* (B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, eds.), pp. 785–794, ACM, 2016.

[39] Q. Hu, Z. Ye, Z. Wang, G. Wang, M. Zhang, Q. Chen, P. Sun, D. Lin, X. Wang, Y. Luo, Y. Wen, and T. Zhang, "Characterization of large language model development in the datacenter," in *21st USENIX Symposium on Networked Systems Design and Implementation, NSDI 2024, Santa Clara, CA, April 15-17, 2024* (L. Vanbever and I. Zhang, eds.), pp. 709–729, USENIX Association, 2024.

[40] H. Qi, L. Dai, W. Chen, and X. Lu, "Early experience in characterizing training large language models on modern hpc clusters," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '23)*, (Denver, Colorado), Association for Computing Machinery, Dec 2023.

[41] J. Yin, A. Bose, G. Cong, I. Lyngaas, and Q. Anthony, "Comparative study of large language model architectures on frontier," in *IEEE International Parallel and Distributed Processing Symposium, IPDPS 2024, San Francisco, CA, USA, May 27-31, 2024*, pp. 556–569, IEEE, 2024.

[42] X. Liu, J. Zhang, G. Wang, W. Tong, and A. Walid, "Fingpt-hpc: Efficient pretraining and finetuning large language models for financial applications with high-performance computing," *CoRR*, vol. abs/2402.13533, 2024.

[43] Z. Li, S. He, P. Chaturvedi, V. V. Kindratenko, E. A. Huerta, K. Kim, and R. K. Madduri, "Secure federated learning across heterogeneous cloud and high-performance computing resources - A case study on federated fine-tuning of llama 2," *CoRR*, vol. abs/2402.12271, 2024.

[44] N. Jain, T. Zhang, W. Chiang, J. E. Gonzalez, K. Sen, and I. Stoica, "Llm-assisted code cleaning for training accurate code generators," *CoRR*, vol. abs/2311.14904, 2023.

[45] W. F. Godoy, P. Valero-Lara, K. Teranishi, P. Balaprakash, and J. S. Vetter, "Evaluation of openai codex for HPC parallel programming models kernel generation," in *Proceedings of the 52nd International Conference on Parallel Processing Workshops, ICPP Workshops 2023, Salt Lake City, UT, USA, August 7-10, 2023*, pp. 136–144, ACM, 2023.

[46] D. Nichols, J. H. Davis, Z. Xie, A. Rajaram, and A. Bhatele, "Can large language models write parallel code?," *CoRR*, vol. abs/2401.12554, 2024.

[47] C. Cummins, V. Seeker, D. Grubisic, M. Elhoushi, Y. Liang, B. Rozière, J. Gehring, F. Gloeckle, K. M. Hazelwood, G. Synnaeve, and H. Leather, "Large language models for compiler optimization," *CoRR*, vol. abs/2309.07062, 2023.

[48] Y. Lu, S. Bian, L. Chen, Y. He, Y. Hui, M. Lentz, B. Li, F. Liu, J. Li, Q. Liu, R. Liu, X. Liu, L. Ma, K. Rong, J. Wang, Y. Wu, Y. Wu, H. Zhang, M. Zhang, Q. Zhang, T. Zhou, and D. Zhuo, "Computing in the era of large generative models: From cloud-native to ai-native," *CoRR*, vol. abs/2401.12230, 2024.

[49] C. Shi, F. Jiang, Z. Liu, C. Ding, and J. Xu, "Memory workload synthesis using generative AI," in *Proceedings of the International Symposium on Memory Systems, MEMSYS 2023, Alexandria, VA, USA, October 2-5, 2023*, pp. 7:1–7:7, ACM, 2023.

[50] A. V. Solatorio and O. Dupriez, "Realtabformer: Generating realistic relational and tabular data using transformers," *CoRR*, vol. abs/2302.02041, 2023.

[51] J. Bucek, K. Lange, and J. von Kistowski, "SPEC CPU2017: next-generation compute benchmark," in *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering, ICPE 2018, Berlin, Germany, April 09-13, 2018* (K. Wolter, W. J. Knottenbelt, A. van Hoorn, and M. Nambiar, eds.), pp. 41–42, ACM, 2018.

[52] D. Wu, X. Wang, Y. Qiao, Z. Wang, J. Jiang, S. Cui, and F. Wang, "Large language model adaptation for networking," *CoRR*, vol. abs/2402.02338, 2024.

[53] Y. Jin, J. Liu, F. Wang, and S. Cui, "Where are you looking?: A large-scale dataset of head and gaze behavior for 360-degree videos and a pilot study," in *MM '22: The 30th ACM International Conference on Multimedia, Lisboa, Portugal, October 10 - 14, 2022* (J. Magalhães, A. D. Bimbo, S. Satoh, N. Sebe, X. Alameda-Pineda, Q. Jin, V. Oria, and L. Toni, eds.), pp. 1025–1034, ACM, 2022.

[54] F. C. Commission, "Raw data - measuring broadband america 2016." `https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016`, Dec 2016. Accessed: 2023-07-21.

[55] T. Chiba and T. Onodera, "Workload characterization and optimization of TPC-H queries on apache spark," in *2016 IEEE International Symposium on Performance*

Analysis of Systems and Software, ISPASS 2016, Uppsala, Sweden, April 17-19, 2016, pp. 112–121, IEEE Computer Society, 2016.

[56] X. Huang, H. Li, J. Zhang, X. Zhao, Z. Yao, Y. Li, Z. Yu, T. Zhang, H. Chen, and C. Li, "Llmtune: Accelerate database knob tuning with large language models," *CoRR*, vol. abs/2404.11581, 2024.

[57] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de Las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," *CoRR*, vol. abs/2310.06825, 2023.

[58] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. C. Chang, F. Huang, R. Cheng, and Y. Li, "Can LLM already serve as A database interface? A big bench for large-scale database grounded text-to-sqls," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023* (A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds.), 2023.

[59] X. Ding, L. Chen, M. Emani, C. Liao, P. Lin, T. Vanderbruggen, Z. Xie, A. Cerpa, and W. Du, "HPC-GPT: integrating large language model for high-performance computing," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis, SC-W 2023, Denver, CO, USA, November 12-17, 2023*, pp. 951–960, ACM, 2023.

[60] N. Schneider, N. Hasabnis, V. A. Vo, T. Kadosh, N. Krien, M. Capota, A. Wasay, G. Tamir, T. Willke, N. K. Ahmed, Y. Pinter, T. Mattson, and G. Oren, "Mpirigen: MPI code generation through domain-specific language models," *CoRR*, vol. abs/2402.09126, 2024.

[61] A. Bahga and V. K. Madisetti, "Synthetic workload generation for cloud computing applications," *J. Softw. Eng. Appl.*, vol. 4, no. 7, pp. 396–410, 2011.

[62] I. Ashrapov, "Tabular gans for uneven distribution," *CoRR*, vol. abs/2010.00638, 2020.

[63] W. Lin, K. Yao, L. Zeng, F. Liu, C. Shan, and X. Hong, "A gan-based method for time-dependent cloud workload generation," *J. Parallel Distributed Comput.*, vol. 168, pp. 33–44, 2022.

[64] N. Seedat, N. Huynh, B. van Breugel, and M. van der Schaar, "Curated LLM: synergy of llms and data curation for tabular augmentation in ultra low-data regimes," *CoRR*, vol. abs/2312.12112, 2023.

[65] V. Borisov, K. Seßler, T. Leemann, M. Pawelczyk, and G. Kasneci, "Language models are realistic tabular data generators," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net, 2023.

[66] T. Zhang, S. Wang, S. Yan, L. Jian, and Q. Liu, "Generative table pre-training empowers models for tabular prediction," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023* (H. Bouamor, J. Pino, and K. Bali, eds.), pp. 14836–14854, Association for Computational Linguistics, 2023.

[67] Y. Yu, Y. Zhuang, J. Zhang, Y. Meng, A. J. Ratner, R. Krishna, J. Shen, and C. Zhang, "Large language model as attributed training data generator: A tale of diversity and bias," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023* (A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds.), 2023.

[68] M. Josifoski, M. Sakota, M. Peyrard, and R. West, "Exploiting asymmetry for synthetic training data generation: Synthie and the case of information extraction," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023* (H. Bouamor, J. Pino, and K. Bali, eds.), pp. 1555–1574, Association for Computational Linguistics, 2023.

[69] T. Dinh, Y. Zeng, R. Zhang, Z. Lin, M. Gira, S. Rajput, J. Sohn, D. S. Papailiopoulos, and K. Lee, "LIFT: language-interfaced fine-tuning for non-language machine learning tasks," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), 2022.

[70] H. Manikandan, Y. Jiang, and J. Z. Kolter, "Language models are weak learners," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023* (A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds.), 2023.

[71] S. Hegselmann, A. Buendia, H. Lang, M. Agrawal, X. Jiang, and D. A. Sontag, "Tabllm: Few-shot classification of tabular data with large language models," in *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain* (F. J. R. Ruiz, J. G. Dy, and J. van de Meent, eds.), vol. 206 of *Proceedings of Machine Learning Research*, pp. 5549–5581, PMLR, 2023.

[72] S. Jaitly, T. Shah, A. Shugani, and R. S. Grewal, "Towards better serialization of tabular data for few-shot classification with large language models," *CoRR*, vol. abs/2312.12464, 2023.

[73] D. Slack and S. Singh, "TABLET: learning from instructions for tabular data," *CoRR*, vol. abs/2304.13188, 2023.

[74] Y. Yang, Y. Wang, S. Sen, L. Li, and Q. Liu, "Unleashing the potential of large language models for predictive tabular tasks in data science," *CoRR*, vol. abs/2403.20208, 2024.

[75] N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson, "Large language models are zero-shot time series forecasters," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023* (A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds.), 2023.

[76] A. Garza and M. M. Canseco, "Timegpt-1," *CoRR*, vol. abs/2310.03589, 2023.

[77] R. C. Fernandez, A. J. Elmore, M. J. Franklin, S. Krishnan, and C. Tan, "How large language models will disrupt data management," *Proc. VLDB Endow.*, vol. 16, no. 11, pp. 3302–3309, 2023.

[78] L. Chen, X. Ding, M. Emani, T. Vanderbruggen, P. Lin, and C. Liao, "Data race detection using large language models," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis, SC-W 2023, Denver, CO, USA, November 12-17, 2023*, pp. 215–223, ACM, 2023.

[79] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[80] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)* (J. Burstein, C. Doran, and T. Solorio, eds.), pp. 4171–4186, Association for Computational Linguistics, 2019.

[81] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020* (D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, eds.), pp. 7871–7880, Association for Computational Linguistics, 2020.

[82] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019.

[83] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019.

[84] Y. Liu, H. He, T. Han, X. Zhang, M. Liu, J. Tian, Y. Zhang, J. Wang, X. Gao, T. Zhong, Y. Pan, S. Xu, Z. Wu, Z. Liu, X. Zhang, S. Zhang, X. Hu, T. Zhang, N. Qiang, T. Liu, and B. Ge, "Understanding llms: A comprehensive overview from training to inference," *CoRR*, vol. abs/2401.02038, 2024.

[85] D. Cruz, E. Pona, A. Holness-Tofts, E. Schmied, V. A. Alonso, C. Griffin, and B. Cirstea, "Reinforcement learning fine-tuning of language models is biased towards more extractable features," *CoRR*, vol. abs/2311.04046, 2023.

[86] X. Chu, S. Talluri, L. Versluis, and A. Iosup, "How do ML jobs fail in datacenters? analysis of a long-term dataset from an HPC cluster," in *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering, ICPE 2023, Coimbra, Portugal, April 15-19, 2023* (M. Vieira, V. Cardellini, A. D. Marco, and P. Tuma, eds.), pp. 263–268, ACM, 2023.

[87] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," *CoRR*, vol. abs/1406.2661, 2014.

[88] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[89] A. Singha, J. Cambronero, S. Gulwani, V. Le, and C. Parnin, "Tabular representation, noisy operators, and impacts on table structure understanding tasks in llms," *CoRR*, vol. abs/2310.10358, 2023.