CS546 Term Project Proposal

LLM-Assisted Workflow I/O Bottleneck Diagnosis

Name: Jiyi Chen, Alistair DREUX-EGGER

PhD Lead: Meng Tang

1 Introduction

Modern scientific and data-intensive applications are increasingly executed as complex workflows composed of multiple computational stages and data movement operations.[1, 2] In such environments, I/O performance often plays a dominant role in determining end-to-end execution time.[3] Although existing monitoring and profiling tools expose detailed metrics such as throughput, latency, and access patterns, diagnosing I/O bottlenecks from these low-level signals remains challenging. Correct interpretation typically requires substantial domain expertise and manual reasoning, making analysis of causes difficult.

Recent advances in large language models (LLMs) suggest a potential opportunity to address this issue.[4, 5] LLMs have demonstrated strong capabilities in reasoning over structured and semi-structured inputs, synthesizing information across multiple signals, and producing human-interpretable explanations.[6] These properties make them a promising candidate for assisting with diagnostic tasks that go beyond raw measurement, particularly those requiring both classification and explanation.

This project explores the feasibility of using LLMs to diagnose I/O bottlenecks in workflow executions. Given structured snapshots of workflow execution data, the goal is to evaluate whether LLMs can accurately identify bottleneck categories and provide meaningful explanations aligned with expert reasoning. The study will focus on dataset curation, prompt design, diagnostic accuracy and other evaluation criteria such as confidence calibration, failure mode taxonomy. Through this investigation, the project aims to assess the potential and limitations of LLM-assisted workflow I/O bottleneck diagnosis.

2 Background

Workflow performance analysis has traditionally relied on monitoring, profiling, and tracing tools[7], as well as automated or machine-learning-based methods[8], to expose low-level system metrics such as I/O throughput, latency, access patterns, and execution timelines. While these approaches are effective at capturing and analyzing runtime behavior, translating such signals into accurate diagnoses of I/O bottlenecks typically requires substantial domain expertise. Most existing methods emphasize measurement, detection, or prediction, and provide limited support for explicit, human-interpretable diagnostic reasoning, particularly in complex workflows where multiple factors interact.

More recently, large language models (LLMs) have been investigated for systems-related tasks such as log analysis[9], debugging assistance[10], and performance interpretation. Their ability to reason over structured inputs and generate natural-language explanations suggests potential for supporting diagnostic workflows. However, the applicability of LLMs to structured workflow I/O bottleneck diagnosis, as well as their ability to produce explanations aligned with expert reasoning, has not been well examined or evaluated.

3 Problem Statement

The problem in this project is to diagnose I/O bottlenecks in workflow executions based on structured execution data. Given a snapshot of workflow runtime information, including I/O-related metrics, resource utilization, and execution characteristics, the task is to infer the dominant I/O bottleneck that affects performance. In addition to identifying the bottleneck category, the diagnosis is expected to include a natural-language explanation.

In this work, I/O bottleneck diagnosis is formulated as a combined classification and explanation task, instead of a purely prediction or optimization problem. I/O Bottlenecks are defined using a predefined, expert-informed set of classes that capture common I/O-related performance issues observed in practical workflow executions. The quality of a diagnosis is determined not only by the correctness of the identified bottleneck category, but also by the coherence of the output explanation. Diagnostic results are evaluated through comparison with expert-annotated ground truth, with particular attention paid to whether the explanations align with expert reasoning and appropriately reference relevant execution signals.

This project is framed as a feasibility study aimed at assessing the applicability of LLMs to workflow I/O bottleneck diagnosis under a controlled setting. The scope is intentionally constrained to a predefined bottleneck taxonomy, a limited collection of workflow execution snapshots, and offline evaluation against expert-annotated ground truth. Within these boundaries, the study seeks to produce empirical insights into when LLM-based diagnoses succeed or fail, as well as how diagnostic performance and explanation quality depend on task formulation and prompting strategies. These considerations directly inform the methodological choices described in the following section.

4 Methodology

This project develops a methodology to evaluate the feasibility of using LLMs for workflow I/O bottleneck diagnosis. The approach is organized into three main components: constructing structured representation workflow execution behaviors, designing prompts that guide LLMs to produce diagnostic outputs, and evaluating the resulting diagnoses against the ground truth. Rather than solely aiming to build an automated diagnostic system, the methodology is intended to assess how accurately LLMs can identify bottleneck categories, how well their explanations align with expert reasoning, and which factors influence their diagnostic behavior in this context.

Data Curation The study uses a set of structured workflow execution snapshots as the primary input for I/O bottleneck diagnosis. Each snapshot provides an independent summary of a workflow execution including throughput, latency, access patterns, and resource utilization. Snapshots are curated from representative workflow runs and paired with expert-annotated bottleneck labels as well as corresponding diagnostic reasoning. This dataset design follows a predefined bottleneck taxonomy and makes it possible to fairly compare different methods on diagnosing I/O bottlenecks.

Prompt Design LLMs are prompted to perform I/O bottleneck diagnosis based on the constructed workflow execution snapshots. Diagnostic prompts guide the model to select a bottleneck category from a predefined set of classes and, in some cases, to generate an explanation grounded in the provided execution data. To enable meaningful comparison, the methodology includes several simpler baseline approaches alongside LLM-based methods, including simple metric-based rules without using LLM reasoning and minimally structured LLM prompting that performs only bottleneck classification without explanation. Those baseline methods together serve as reference points for examining how prompt structure and explanation requirements affect diagnostic performance.

Evaluation Evaluation focuses on assessing the correctness and validity of diagnostic outputs produced by LLMs. Bottleneck identification is evaluated by comparing predicted categories against expert-annotated ground truth. Explanation quality is assessed by examining whether explanations are consistent with expert reasoning and reference relevant execution features present in the snapshot. Results from different prompt formulations and baseline approaches are compared to understand their relative effectiveness under the same evaluation setting.

The analysis focuses on understanding how and when LLM-based diagnoses are effective, as well as identifying common success cases and failure patterns. Particular attention is paid to how diagnostic outcomes and explanations vary with different prompt formulations.

5 Timeline

Week	Project Tasks
1–2(3)	Set up working environment, code repository and required LLM-related tools. Review relevant materials on workflow I/O bottlenecks and LLMs; Study LLM-based reasoning and prompting techniques.
3(4)–6	Define the workflow execution snapshot structure and collect data. Implement mentioned baseline LLM-based diagnostic methods w/ the data.
7–10	Perform evaluation of LLM-based diagnosis results. Continue to refine LLM prompting strategies.
11–12	Summarize evaluation results, and finalize conclusions. Prepare the presentation and report.

References

[1] Peter Harrington et al. "Diagnosing parallel i/o bottlenecks in hpc applications". In: *International Conference for High Performance Computing Networking Storage and Analysis (SCI7) ACM Student Research Competition (SRC)*. 2017, p. 4.

[2] Jakob Lüttgau et al. "Toward Understanding I/O Behavior in HPC Workflows". In: *2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS)*. 2018, pp. 64–75. DOI: 10.1109/PDSW-DISCS.2018.00012.

[3] Luanzheng Guo et al. "Improving I/O-aware Workflow Scheduling via Data Flow Characterization and trade-off Analysis". In: *2024 IEEE International Conference on Big Data (BigData)*. 2024, pp. 3674–3681. DOI: 10.1109/BigData62323.2024.10825855.

[4] Yongxin Zhao et al. "When LLMs Listen to Experts: Accurate Failure Diagnosis in Operating Systems". In: ().

[5] Yusuke Miyashita, Patrick Kin Man Tung, and Johan Barthélemy. "LLM as HPC Expert: Extending RAG Architecture for HPC Data". In: *arXiv preprint arXiv:2501.14733* (2024).

[6] Dibyanayan Bandyopadhyay, Soham Bhattacharjee, and Asif Ekbal. "Thinking machines: A survey of llm based reasoning strategies". In: *arXiv preprint arXiv:2503.10814* (2025).

[7] Shane Snyder et al. "Modular HPC I/O Characterization with Darshan". In: *2016 5th Workshop on Extreme-Scale Programming Tools (ESPT)*. 2016, pp. 9–17. DOI: 10.1109/ESPT.2016.006.

[8] Alok Singh et al. "A machine learning approach for modular workflow performance prediction". In: *Proceedings of the 12th workshop on workflows in support of large-scale science*. 2017, pp. 1–11.

[9] Xin Huang, Ting Zhang, and Wen Zhao. "LogRules: Enhancing Log Analysis Capability of Large Language Models through Rules". In: *Findings of the Association for Computational Linguistics: NAACL 2025*. 2025, pp. 452–470.

[10] Runchu Tian et al. "Debugbench: Evaluating debugging capability of large language models". In: *Findings of the Association for Computational Linguistics: ACL 2024*. 2024, pp. 4173–4198.