Fail-Slow Algorithm Analysis: Language Model Based Detection of the RAID Slowdown Conditions

Project Mentors:

Ruidan Li
Kexin Pei

Introduction:

- In RAID systems, early detection of potential slowdowns or failures is crucial for minimizing their impact on system performance. Fail-slow conditions, where a system continues operating but with significantly reduced performance, are challenging to detect in storage systems due to their gradual onset (“silent failure”) and the variability of normal system performance.

- To tackle the fail-slow issue, various strategies have been developed, but their effectiveness and wide applicability are still under concern. Specifically, the precision and ability to pinpoint the problematic disk may not often perform well.

- Our methodology enhances the detection of fail-slow conditions by integrating Large Language Models (LLMs) such as ChatGPT-4 or the locally implemented LLaMA-2. We assess the effectiveness of these innovative approaches by benchmarking them against existing methods, aiming to improve both accuracy and issue localization.

Project Goals:

- Objectives:
    - To develop a reliable method for detecting fail-slow conditions in RAID systems using LLMs.
    - Enhance the efficiency and accuracy of identifying fail-slow faults to reduce the time and resources required for problem resolution.
    - Implementing LLMs in fail-slow detection processes could revolutionize how storage systems manage and mitigate performance degradation, offering a novel, effective approach to a persistent and complex issue.

- Expected Deliverables:
    - An enhanced fail-slow detection algorithm incorporating LLM analysis.
        - A Google Colab notebook for quick replay
    - A comprehensive evaluation of the algorithm's performance in real-world RAID system environments.
        - A GitHub repository hosting the complete evaluation results.

- Future Work:
    - Building on this project's findings, future research should consider integrating more sophisticated LLMs or customizing the LLM specifically for the target system to enhance detection accuracy and address limitations.

Implementation Plan:

- Project Methodology:
    - We will adopt a structured approach to the project, segmenting it into manageable phases. Regular progress reports will be provided during each meeting, with a biweekly presentation to showcase completed tasks and ensure alignment with the project's objectives. Communication with mentors will be Zoom meetings. For interactions with members of the open-source community, we will utilize email for collaborating and engaging in discussions, also slack channels for chatting.

- Technical Elements:
    - Deployment of LLM:
        - Integrate a Large Language Model (LLM) into our pipeline, transforming it into an expert storage agent capable of analyzing and diagnosing RAID system behaviors.
    - Pipeline Optimization:
        - Refine the agent's pipeline to enhance its relevance and accuracy in identifying fail-slow conditions within the storage system.
    - Data Preprocessing and Automation:
        - Utilize Python for effective data preprocessing and Bash scripts to automate routine tasks.
    - Prompt Engineering
        - Refine the prompt template to better accommodate extensive time series data, and improve the accuracy of the analysis by employing methods like few-shot learning.

- Challenges and Solutions:
    - Challenge: The proficiency of Large Language Models (LLMs) in handling time-series data is inadequate.
    - Proposed Solution: Enhance the pipeline at every stage using optimization methods, such as Retrieval-Augmented Generation (RAG), or by modify the model to improve its effectiveness in processing time-series data, for example, using LSTM to deal with the Time Series before passing into LLMs .

Project Timeline:

- Duration:
    - The project is planned for the summer quarter, lasting approximately 10 weeks (working 350 hours) .

- Weekly Commitment:
    - An estimated 30-40 hours per week will be dedicated to the project.

- Project Plan:

    - First Half:
        - Weeks 1-2:
            - Complete the implementation of the LLM pipeline to obtain viable results.
        - Weeks 3-4:
            - Integrate advanced optimizations to enhance system robustness and conduct tests using the test data set to finalize the implementation.
    - Mid-Project:
        - Week 5:
            - Post-initial setup and testing, focus on evaluating and refining the system to ensure its effectiveness.
    - Second Half:
        - Weeks 6-8:
            - Evaluate and summarize the accuracy and benefits of the previously implemented methodologies.
            - Begin by implementing these methodologies to identify and understand any challenges.
            - Conduct a comparative analysis between these methodologies and our LLM approach.
        - Weeks 9-10:

- Optimize our LLM agent based on the gathered data and insights.
- Conclude the project by preparing a Google Colab notebook and a report on RAID system tests in the cloud.

Personal Information:

- Name:
  - Xikang Song
- Affiliation:
  - Master of Computer Science (MPCS) student at the University of Chicago, specializing in Computer Science
- Education:
  - Bachelor of Science (BS) degree in Electrical and Computer Engineering (ECE) from Rutgers University, with a minor in Computer Science
- Relevant Experience and Background:
  - Solid knowledge in computer systems
  - Proficient in using the PyTorch framework and Python programming
  - Interested in implementing Large Language Models (LLMs) across various applications
- Contact Information:
  - GitHub:
    - https://github.com/songxikang
  - Email:
    - xikang@uchicago.edu