ChronoLog & AI: A Scalable, Collaborative Solution for LLM Conversation Logging.

Soham Sonar, Jaime Cernuda, Dr. Anthony Kougkas and Dr. Xian-He Sun

Gnosis Research Center

Department of Computer Science

ILLINOIS TECH
College of Computing

Introduction

There is an exponential growth in chat interactions with large language models, which necessitates an efficient and scalable solution to capture, manage, and analyze conversation data effectively. This project presents an integrated solution that combines ChronoLog a scalable, high-performance distributed shared log store with an AI inference interface to create a fully traceable, end-to-end system for logging and retrieving conversations with large language models. By capturing both internal and external prompts and responses in real-time, our architecture not only supports comprehensive monitoring and analysis of AI interactions but also ensures reproducibility and accountability. Benchmarking tests comparing performance with and without ChronoLog logging demonstrate the system's efficiency and the overhead introduced.

Problem Statement

- Rapidly increasing chat interactions with large language models create a significant challenge in efficiently managing and analyzing vast amounts of conversational data.
- Existing systems often lack integrated, real-time logging mechanisms, making it difficult to trace and debug AI interactions.
- The solution's potential for cross-disciplinary applications makes it relevant not only to AI specialists but also to a broader audience interested in data management, system performance, and ethical AI practices.

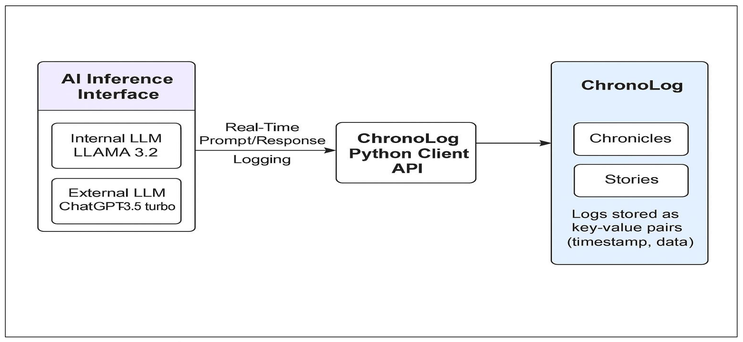

Fig 1. AI–ChronoLog Integration and Data Flow

Background and Methodology

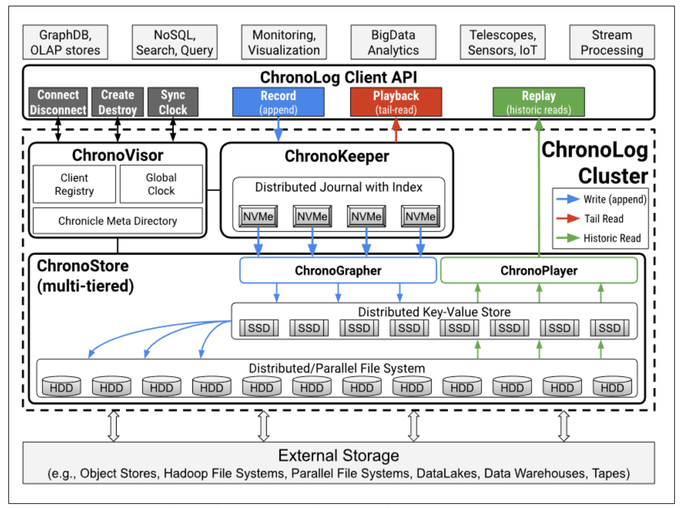

Fig 2: Chronolog Design overview

ChronoLog Architecture:

- ChronoLog's architecture (as shown in fig. 2) features a client interface that connects to a centralized server managing multiple chronicles, which are essentially separate data streams. Within each chronicle, individual sessions (stories) are recorded as events which is a key-value pair, where key is the timestamp and value is an uninterpreted byte array.
- The system utilizes physical time to record each log entry, ensuring a consistent and accurate chronological order of events across distributed systems.

Approach & Techniques:

- We connected the AI inference interface (Both Local LLM as well as External LLM) directly with ChronoLog, using ollama for local LLM interface and OpenAI API to connect with external LLM. (fig. 1)
- Leveraged ChronoLog's Python client APIs to seamlessly interact with the logging system in real time. This direct API integration minimizes latency, allowing for efficient data capture without interrupting the AI inference process.
- Harnessed ChronoLog's capability to manage high-velocity data by organizing logs into chronicles (data streams) and stories (session logs), providing scalability and fault tolerance.

Results

Benchmark Specifications:

•We tested logging for 100 prompts and responses on a single node ChronoLog deployment.
•For the internal LLM benchmark, we used the LLAMA 3.2 model, and for the external LLM, we used ChatGPT-3.5 turbo.

•Interpretation:

•ChronoLog introduces minimal performance overhead despite real-time logging of each interaction.
•The architecture scales well for both external API calls and local model inference.
•This validates the feasibility of ChronoLog as a reliable logging backend for LLM-based systems without significantly affecting performance.

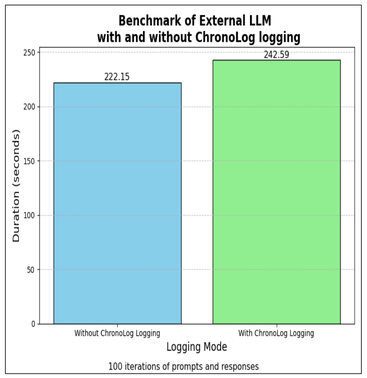

Fig 3: External LLM Benchmark

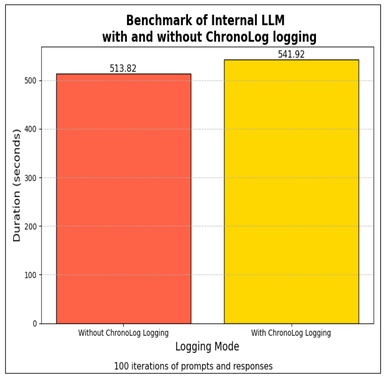

Fig 4: Internal LLM benchmark

Conclusion

- The integration of ChronoLog with AI inference interfaces successfully logs closed chat conversations with LLM's in real time with high speed and minimal performance overhead.

•The project is open-source, and you can explore the underlying code and architecture at:
•https://github.com/grc-iit/ChronoLog
•https://github.com/sohamvsonar/chronoai