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

Gnosis Research Center @ ILLINOIS TECH

Soham Sonar Jaime Cernuda Dr. Anthony Kougkas Dr. Xian-He Sun
ssonar2@hawk.iit.edu, jcernudagarcia@hawk.iit.edu,
akougkas@iit.edu, sun@iit.edu

ILLINOIS TECH

Introduction

The widespread use of Large Language Models (LLMs) has led to a surge in conversational data across research and industry. Capturing and managing these interactions is essential for reproducibility, debugging, and auditing, yet most current systems rely on proprietary or cloud-based infrastructure not designed for high throughput, AI specific logging. To address this, we explore the use of ChronoLog, a high-performance, HPC-oriented distributed log store with precise physical time ordering, multi-user concurrent access, and seamless read integration, making it ideal for capturing fast, structured data such as AI conversations.

On top of the ChronoLog, we built -

- A Python Inference which logs prompt-response pairs from local and remote LLM models directly into ChronoLog. It supports real time logging and time range retrieval for seamless integration into workflows.
- A Model Context Protocol server which is a service that implements a standardized interface for managing and relaying contextual data and tool interactions between language models and external systems. By integrating it with ChronoLog and exposing a uniform protocol, it enables context-aware retrieval and cross-platform communication.

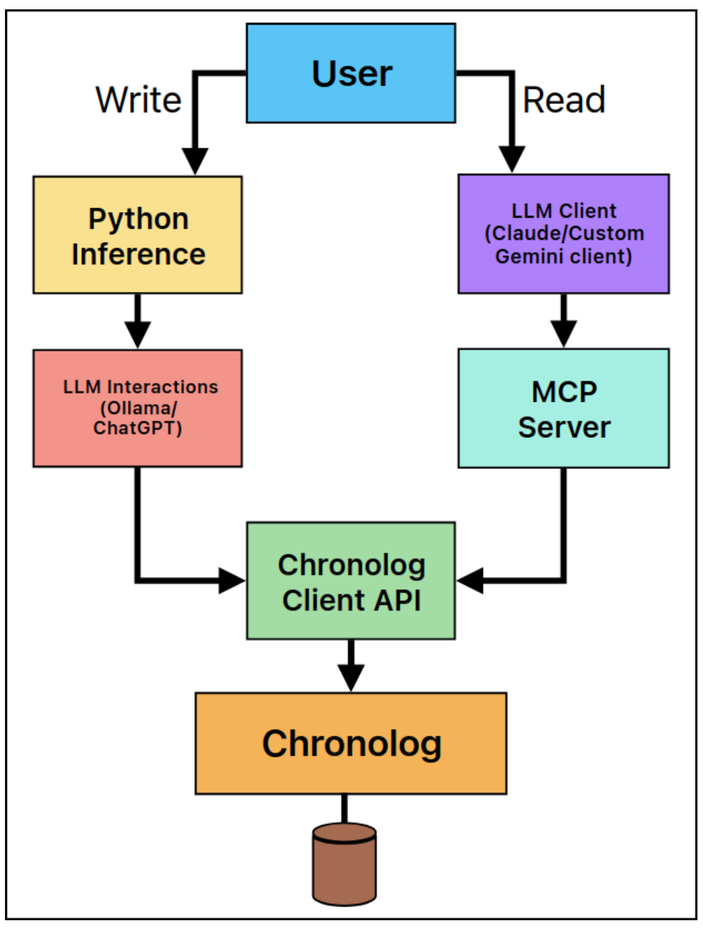

Figure 1: Collaborative Write/Read using Python Inference and Mcp Server.

Methodology

1. **ChronoLog:** Forms the core of our logging infrastructure, built for high-throughput, distributed environments. It organizes data into chronicles (streams) and stories (sessions), recording events as key-value pairs with physical timestamps for globally ordered logging. In-memory writes ensure speed and long term durability.
2. **Python inference pipeline:** Prompts are issued to either a local model (LLAMA 3.2) or a remote model through API (ChatGPT 4.0), and the prompt-response pair is logged in real time into ChronoLog. Logs can be retrieved using a time range reader API for offline analysis.
3. **The MCP Server:** Extends this into a robust API service using FastMCP, exposing tools to start sessions, log interactions, retrieve archived logs, and end sessions. It supports real time and contextual querying across conversations, enabling reproducible and collaborative AI workflows. This server can be easily integrated with any open source LLM Client Applications (Claude AI, Microsoft Copilot) or a Custom LLM Client using Gemini.

Results

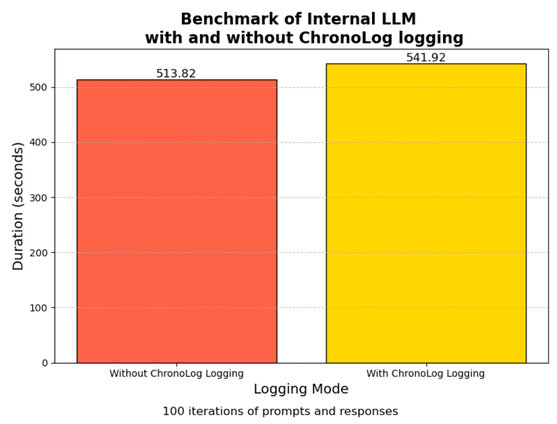

Figure 2(a): Internal LLM benchmark

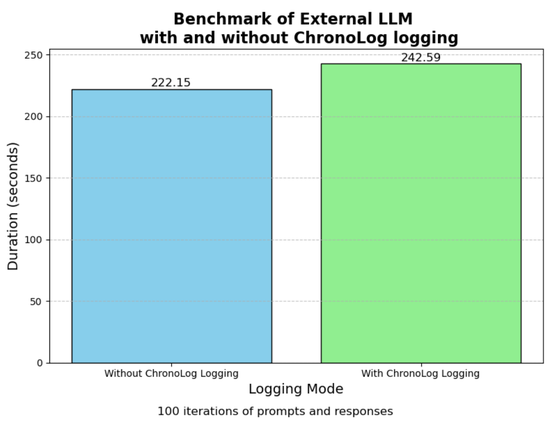

Figure 2(b): External LLM benchmark

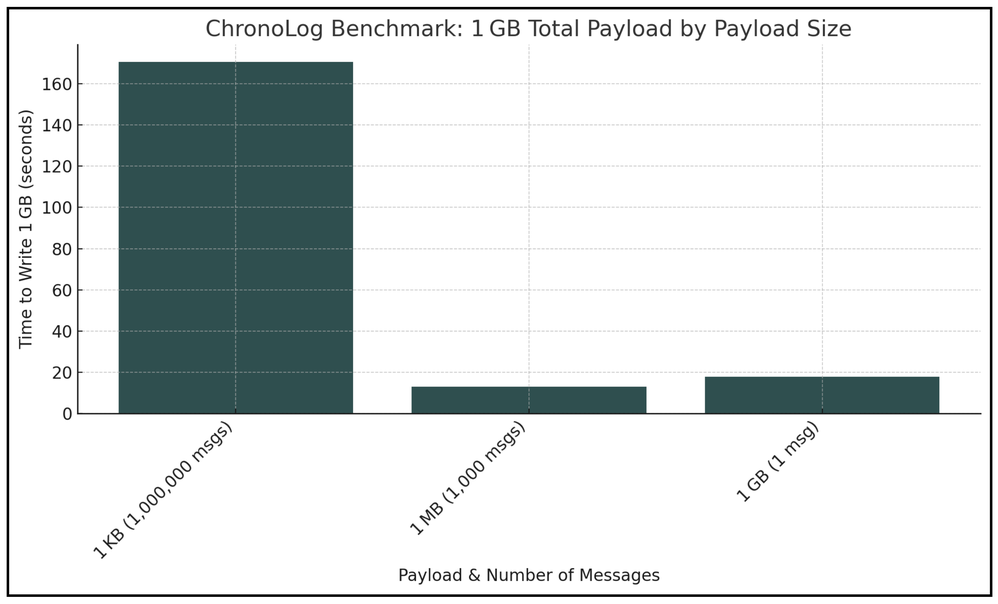

Figure 3: ChronoLog Write Performance

Observations: ChronoLog adds < 10% latency to internal and external LLM inference, and the Write tests show many 1 KB events are latency-bound, whereas 1 MB–1 GB batches saturate memory bandwidth and complete markedly faster.

Conclusion

We introduced ChronoLog as a unified backend for logging and replaying LLM interactions, combining a Python interface with a featured Model Context Protocol server. Experiments confirm low latency overhead while sustaining high-throughput ingest and fast archival queries, even at multi gigabyte scales. Overall, our work lays the groundwork for reproducible, transparent, and traceable AI workflows.