VIETNAM NATIONAL UNIVERSITY OF HO CHI MINH CITY

INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Anomaly Detection in HDFS Logs using Machine Learning Integrated with LLM-Based Mitigation

By

Nguyen Hoang Quan

The thesis submitted to School of Computer Science and Engineering in partial fulfillment of the requirements of the degree of Bachelor of Engineering of Information Technology

Ho Chi Minh, Viet Nam

2025

Anomaly Detection in HDFS Logs using Machine Learning Integrated with LLM-Based Mitigation

APPROVED BY: _____

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to the School of Computer Science and Engineering for providing a supportive academic environment and the essential resources that made this thesis possible.

I am deeply thankful to my advisor, Dr. Le Hai Duong, for his invaluable guidance, encouragement, and constructive feedback throughout the development of this thesis. His expertise and commitment has greatly enriched both my technical understanding and research direction.

I would also like to extend my appreciation to all lecturers and staff members who have contributed to my academic journey with their instruction, support, and mentorship. Special thanks go to my family and friends for their constant support, motivation, and understanding during the preparation of this work. Lastly, I would like to acknowledge the developers and the researchers behind the resources for providing the foundation upon which this project was built.

This thesis would not have been possible without the guidance, resources, and support from all of the above.

Contents

List of Tables

List of Figures

ABSTRACT

Anomaly detection is essential for managing today's large-scale distributed systems, where system logs are a key resource for identifying unusual behavior. Traditionally, system operators relied on manual inspection methods such as keyword searches and rule-based matching. However, due to the massive volume and complexity of modern system logs, manual approaches are no longer practical. To tackle this, many automated log-based anomaly detection methods have been proposed. Still, developers often struggle to choose a suitable method, as there hasn't been a clear comparison of these approaches.

In this research, I will propose a solution to addresses the fundamental challenge of automated log analysis. Specially, the research tackles three interconnected problems: (1) automated parsing of diverse log formats into structured templates, (2) anomaly detection in high-volume log streams, and (3) generation of contextual, actionable recommendations for identified issues.

Previous research has established some foundational approaches including the algorithms for log parsing and machine learning techniques for anomaly detection. However, existing solutions typically focus on individual components rather than providing end-to-end integration. Most academic implementations lack production-ready deployment architectures, user-friendly interfaces, and the integration of modern Large Language Models (LLMs) for intelligent recommendations—creating a significant gap between theoretical algorithms and practical deployment. Therefore, this research is important since it close the gap between academic log analysis algorithms and production-ready systems.

Furthermore, the research establishes a framework for integrating emerging LLM capabilities into traditional system administration workflows, suggesting broader implications for AI-assisted DevOps practices. The findings indicate that intelligent automation of log analysis is not only technically feasible but can significantly enhance

organizational capabilities in system reliability, security monitoring, and operational efficiency.

CHAPTER 1: INTRODUCTION

1.1 Background of the study

In modern software systems, log files serve as critical sources of information for system monitoring, debugging, troubleshooting, and security analysis. As applications or systems scale and increase its complexity, the volume of generated log data gets bigger exponentially. Therefore, traditional manual approaches for log analysis like manually examine through log files using basic tools such as grep or text editors, has become less efficient and more defective [1]. As the result, it is becoming more difficult to detect anomalies within large scale system.

Over the year, a lot of automated log-based methods have been introduced to help detecting system anomalies. These approaches usually require raw log preprocessing techniques, feature extraction and machine-learning-based algorithms for processing vast amounts of unstructured log data efficiently, identifying potential issues before they escalate into critical failures. Recent advancements in natural language processing and large language models (LLMs) have also increased the potential for intelligent log analysis systems. However, despite these development, traditional machine-learning techniques and LLM-based approaches have yet to be efficiently integrated into a single, cohesive log analysis framework.

This study focuses on leveraging existing machine learning approaches where log parsing and anomaly detected are used, and augmenting them with large language

model to provide actionable insights and helpful recommendations.

1.2 Problem Statement

Despite the important role of log analysis that play in making the system more secure and reliable, several significant challenges still persist in current practices:

- *Limited interpretability.* In log anomaly detection, the ability to interpret model's outputs is important for people who work as system administrators or analysts to effectively action to the alerts. They need to understand which log entries may be responsible for the detected abnormality. Yet, many traditional approaches only provide basic classified prediction without any explanation. As a result, engineers still have to perform further manual root cause analysis which in large-scale and complex systems becomes an very time-consuming and heavy task.

- *Poor adaptability.* Many existing methods rely on a predefined set of log event templates during feature extraction phase (This phase will use the set of log event templates generated by log parsing to create numerical features for machine learning models [4]). However, as applications scale up and expand in term of feature, new and unseen log events will definitely appear. Therefore, adapting to these changes require retraining models from scratch which make the systems become less practical in dynamic environments.

- *Poor adaptability.*

1.3 Objectives of the Study

This study will perform the investigation on existing anomaly-detection methodologies, compares the performance of different machine-learning models, and develops a practical framework that integrates large language models (LLMs) to automate and improve anomaly detection in real-world scenario.

1.4 Limitations of the Study

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction to system log

System logs, also known as event logs or audit trails, are automically generated by a computer system (its operating system, services, applications) that record events, changes and error that occur inside that system. These log files serve an important role in monitoring system health, diagnose failures, detecting anomolies, and ensure system security and complience [6]. System logs usually follow a semi-structured format which consist of several fields that capture both the context and content of the event.

```
1  2016-10-02 12:24:52,337 INFO org.apache.hadoop.hdfs.server.datanode.DataNode.clienttrace: src: /10.10.34.11:58237, dest: /10.10.34.11:50010,
           bytes: 144, op: HDFS_WRITE, cliID: DFSClient_NONMAPREDUCE_1903091109_103, offset: 0, srvID: d9ef1b17-4314-4cd8-91eb-095413c3427f,
           blockid: BP-108841162-10.10.34.11-1440074360971:blk_1074555986_815162, duration: 16128279
2  2016-10-02 12:24:52,337 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: PacketResponder:
           BP-108841162-10.10.34.11-1440074360971:blk_1074555986_815162, type=HAS_DOWNSTREAM_IN_PIPELINE terminating
3  2016-10-02 12:24:52,393 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: Receiving
           BP-108841162-10.10.34.11-1440074360971:blk_1074555988_815164 src: /10.10.34.11:58242 dest: /10.10.34.11:50010
4  2016-10-02 12:24:52,394 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: Receiving
           BP-108841162-10.10.34.11-1440074360971:blk_1074555989_815165 src: /10.10.34.11:58243 dest: /10.10.34.11:50010
5  2016-10-02 12:24:52,399 INFO org.apache.hadoop.hdfs.server.datanode.DataNode.clienttrace: src: /10.10.34.11:58242, dest: /10.10.34.11:50010,
           bytes: 66, op: HDFS_WRITE, cliID: DFSClient_NONMAPREDUCE_1530486806_104, offset: 0, srvID: d9ef1b17-4314-4cd8-91eb-095413c3427f,
           blockid: BP-108841162-10.10.34.11-1440074360971:blk_1074555988_815164, duration: 2155456
```

Figure 2.1: This is the descriptive text that explains the figure.

Figure 2.1 show a log snippet that was generated by HDFS DataNode, one of the core storage components in the Hadoop ecosystem (a more detail explanation of this system will be provided below). These log entries are created from the storage operations of the system such as data writes, packet handling, and communication between DataNodes. A log entry typically follow a structure as follow:

- *Timestamp: 2016-10-02 12:24:52,337.* Indicates the exact moment when the system processed the event.
- *Log level: INFO.* Shows that the entry reports normal operational activity
- *Log message: org.apache.hadoop.hdfs.server.datanode.DataNode.clienttrace:...* Log messages provide a structured description of system behavior, capturing events, performance indicators. This is the most important part of a log entries and therefor become the foundation for monitoring, auditing and anomaly detection.

Given their important role in providing such essential information about the system, logs must be reliably collected, centrally stored and appropriately maintained to enable accurate monitoring, troubleshooting, and subsequent analytical processes.

2.2 Traditional log analysis method

Traditional log analysis have relied on two primary approaches (cite something): manual log inspection and rule-based pattern matching systems. While these techniques are the foundation of the system administrators fields and troubleshooting, it were used during the time when the system complexity and log volumes were much smaller than nowadays modern systems. Understanding the strengths and limitations of these approaches will provide you a better context and insights of logs which will enhance your log analysis systems.

2.2.1 Manual Log Inspection

Manual log inspection is one of the earliest and most straightforward methods used by system administrators to identify issues. In this approach, developers or administrators will directly operate searching on raw log files using command-line tools such as `grep` , `tail`, `less`, to locate error logs or correlate timestamp of the incidents. While

feasible for small log volumes, this method is very time-consuming, highly error-prone and relies heavily on an operator's familiarity with the system. Therefore, this approach is no longer a feasible solution for detecting anomalies in large-scale systems which may produce millions of log entries per hour. For instance, in 2013 the Alibaba Cloud system was reported to generate approximately 100 to 200 million log lines in one hour [3].

2.2.2 Rule-Based Systems (Regex and Pattern Matching)

To counter the limitations of manual log review, developers created rule-based system that used fixed rules, regular expressions (regex), and predefined patterns to filter, parse, and identify specific events when a specification conditions are met. For example, system operators can create rules to detect keywords such as "ERROR", "FAILED" or extract key information from log entries. Rule-based methods offer some advantages: they are easy to implement and effective for detecting obvious or recurring errors. Many traditional monitoring tools such as Nagios, Splunk (early versions) [5], and Logstash [2], they are heavily depend on popular pattern-matching techniques to filter, categorize, and index log data.

However, the effectiveness of rule-based approaches highly depend on the completeness and accuracy of the hand-crafted predefined rules. It will become more defective in a constantly evolving environments where logs format change frequently and new types of anomalies will appear which do not match the existing patterns. Maintaining large sets of regex rules is also a labor-intensive action and require continuous manual updates on patterns which is unsuitable for adaptive anomaly detection.

2.3 Log parsing

2.4 Machine learning and deep learning for log anomaly detection

As system continuously scaled and logs volumes grew bigger, manual or rule-based approaches is no longer available. As a result, researchers began to adopt machine learning to automate anomoly detection. These methods are created to enable it to learn the structures, numerical representations, and patterns of logs without depending on predefined rules or templates. These approches are often categorized as supervised or unsupervised learning.

2.4.1 Supervised learning

a. Support Vector Machine (SVM)

   SVM is one of he

b. Random Forest (RF)

c. The final item summarizes the topic.

2.4.2 Unsupervised learning

a. K-means

b. DBSCAN

   - ml techniques: https://arxiv.org/pdf/2307.16714

2.5 Large Language Models (LLMs) in Log Analysis

Parsing with LLMs instead of templates:

Explainability + anomaly detection:

CHAPTER 3: METHODOLOGIES

This chapter presents the methodology used to develop and evaluate the proposed log anomaly detection framework, including data preparation, model design, integration of large language models, and evaluation metrics.

3.1 Overview

3.1.1 System Pipeline Overview

assdas

3.1.2 Research Approach

fasdfasd

3.2 System Architecture and Design

3.2.1 Overall Architecture

3.3 Dataset preparation

3.4 Log Parsing Methodology

3.5 Feature Extraction / Representation

3.6 Anomaly Detection Models

3.7 LLM Integration Method

3.8 Implementation Details

CHAPTER 4: IMPLEMETATIONS

4.1 Implementation Overview

4.2 Backend Implementation

4.2.1 FastAPI Application Architecture

- Code snippets from src/main.py - Router organization - Dependency injection - Database session management

4.2.2 Database Layer Implementation

- model diagram - detail

4.2.3 API Endpoints Implementation

4.3 Log Processing Pipeline

4.3.1 File Upload and Storage

- MinIO client integration - File validation logic - Unique filename generation

4.3.2 Asynchronous Processing with Celery

- Celery configuration - DatabaseTask base class - Task retry logic - Job status updates

4.3.3 Drain Parser Integration

- Parser service wrapper - Temporary file handling - Template extraction - Output structuring

4.3.4 Feature Extraction

- Event counting - Statistical features - CSV conversion

4.3.5 Decision Tree Anomaly Detection

- Model training code - Feature preparation - Prediction pipeline - Result storage

4.4 LLM Integration

4.4.1 LLM Service Design

- API client setup - Context preparation

4.4.2 LLM Service Design

- System prompts - User prompts with context - Few-shot examples (if any)

4.4.3 Recommendation Generation

- Anomaly result processing - LLM API calls - Response Parsing

4.5 Frontend Implementation

4.5.1 React Application Structure

- Component organization - Routing setup

4.5.2 Key Components

CHAPTER 5: CONCLUSION AND RECOMMENDATION

Bibliography

[1] John Doe and Jane Smith. Anomaly detection in distributed systems. *Journal of Computing*, 10:1–15, 2023.

[2] Elastic. Logstash grok filter documentation. `https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html`, 2024. Accessed: 2025-01-10.

[3] Haibo Mi, Huaimin Wang, Yangfan Zhou, Michael Rung-Tsong Lyu, and Hua Cai. Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1245–1255, 2013.

[4] Mohanad Sarhan, Siamak Layeghy, Nour Moustafa, Marcus Gallagher, and Marius Portmann. Feature extraction for machine learning-based intrusion detection in iot networks. *Digital Communications and Networks*, 10(1):205–216, 2024.

[5] Splunk Inc. Using the field extractor. `https://docs.splunk.com/Documentation/Splunk/latest/Knowledge/Extractfields`, 2024. Accessed: 2025-01-10.

[6] Hudan Studiawan, Ferdous Sohel, and Christian Payne. A survey on forensic investigation of operating system logs. *Digital Investigation*, 29:1–20, 2019.