Grenze International Journal of Engineering and Technology, June Issue

AI-Powered Log Monitoring and Automated Remediation using OpenTelemetry

Vedant Salunke[1], Ayush Sakalkale[2], Sapnil Modak[3], Jayanand Shah[4] and Sachin Pande[5]
[1-5]Pune Institute of Computer Technology, Pune, India
Email: vedantsalunke29@gmail.com, ayushsakalkale30@gmail.com, sapnilmodak@gmail.com, jayanand95shah@gmail.com, sspande@pict.edu

***Abstract*— Distributed microservice architectures produce massive volumes of logs that make manual monitoring inefficient and time-consuming. Conventional monitoring tools rely on rule-based alerts and manual inspection, which often fail to detect emerging failures in complex systems. This work presents an intelligent log monitoring framework that combines real-time telemetry collection, machine learning-based anomaly detection, and large language model-based analysis to improve system observability. Logs are collected using OpenTelemetry and processed through a streaming pipeline built with Apache Kafka and Apache Flink. An ensemble of machine learning models, including Isolation Forest, LSTM, and Prophet, is applied to identify abnormal patterns in log streams. When anomalies are detected, a language model generates interpretable insights and recommended remediation actions. The system stores processed data in Elasticsearch and visualizes insights through Kibana dashboards. Experimental evaluation demonstrates high detection accuracy and low processing latency, enabling faster incident diagnosis and reduced operational overhead.**

***Index Terms*— Log Analysis, AIOps, Observability, OpenTelemetry, Machine Learning, Anomaly Detection, Automated Remediation, Distributed Systems.**

I. INTRODUCTION

Modern software systems increasingly rely on distributed architectures composed of microservices deployed across cloud infrastructure. These architectures allow services to scale independently and support high availability, but they also introduce significant operational complexity. Each service generates logs describing system events, transactions, and failures. In large-scale systems, these logs accumulate rapidly, often producing terabytes of data daily.

Logs play an important role in diagnosing system failures and understanding system behavior. Engineers frequently rely on log analysis to investigate incidents and detect anomalies. However, manual log inspection becomes impractical as the volume of data grows. Traditional monitoring tools such as Elasticsearch or Splunk aggregate logs allow search-based analysis, but they depend on static rules or threshold-based alerts. These techniques fail to detect unknown failure patterns and often generate false alerts.

Recent research in Artificial Intelligence for IT Operations (AIOps) proposes the use of machine learning models to automatically analyze operational data. Machine learning-based log analysis can detect anomalies without requiring predefined rules. Early approaches applied statistical methods and clustering algorithms, while later studies introduced deep learning models capable of analyzing sequential log patterns.

Grenze ID: 01.GIJET.12.2.129_24

Despite progress in anomaly detection, many monitoring systems still lack mechanisms to assist engineers in interpreting anomalies or suggesting remediation actions. Detecting an anomaly alone does not provide sufficient information for resolving system failures. Therefore, monitoring platforms must combine anomaly detection with intelligent reasoning.

This research proposes an AI-powered log monitoring framework that integrates real-time telemetry collection, machine learning-based anomaly detection, and automated remediation suggestions. The architecture combines OpenTelemetry for standardized telemetry collection with real-time streaming frameworks and machine learning algorithms to analyze log events.

The proposed system aims to improve system reliability by detecting anomalies in real time and providing interpretable insights for incident resolution.

II. Related Work

Du et al. proposed DeepLog, which applies Long Short-Term Memory networks to model sequential relationships in log events and detect anomalies when unexpected log sequences occur. The approach demonstrated promising results on Hadoop Distributed File System logs but struggled when log templates evolved over time [1].

Meng et al. introduced LogAnomaly, a hybrid anomaly detection system that combines semantic embeddings with sequential modeling of logs. Their approach improves detection accuracy by capturing both structural patterns and parameter variations within log messages. However, the system requires complex feature engineering and does not generalize well across domains [2].

Chen and Liao developed BERT-Log, a transformer-based anomaly detection system that treats log messages as natural language sequences. Using masked language modeling, the system learns contextual representations of log events and achieves high precision and recall in anomaly detection tasks. Despite its performance, the computational cost of transformer models remains a limitation [3].

Lee et al. proposed LanoBERT, a log-specific transformer model designed to improve anomaly detection accuracy by incorporating log-specific pretraining strategies. The model achieved more than 98% detection accuracy on benchmark datasets but required substantial computational resources for training [4].

Industrial AIOps platforms such as IBM Watson AIOps and Dynatrace provide automated monitoring solutions that integrate machine learning with operational analytics. While these platforms offer advanced monitoring capabilities, they are typically proprietary systems with limited transparency in their algorithms [5].

The proposed system differs from existing work by combining real-time telemetry collection using OpenTelemetry with an ensemble anomaly detection approach and explainable remediation recommendations.

III. Motivation

In modern distributed systems, operational reliability depends heavily on the ability to quickly detect and resolve system failures. However, manual monitoring of log data has become increasingly difficult due to the scale of modern cloud infrastructures.

A typical microservice-based application may consist of hundreds of services generating millions of log entries per hour. When a failure occurs, engineers must manually analyse logs from multiple services to identify the root cause. This process is time-consuming and may significantly delay incident resolution.

Another challenge is that traditional monitoring systems are reactive in nature. Alerts are triggered only after a failure has already occurred and impacted system performance. There is a growing need for proactive monitoring systems capable of detecting anomalies before they escalate into critical failures.

Recent advancements in machine learning and artificial intelligence provide an opportunity to automate log analysis and improve system observability. The combination of AI-based anomaly detection and standardized telemetry frameworks can significantly enhance system reliability.

These challenges and technological developments motivated the development of the AI-powered monitoring system presented in this research.

IV. Problem Domain

The domain of this research lies in AI-driven observability and intelligent monitoring of distributed systems. Observability systems analyse operational data generated by software applications and infrastructure components to understand system behaviour.

Within this domain, log analysis plays a crucial role because logs contain detailed event-level information about system operations. Unlike metrics, which represent aggregated numerical data, logs provide contextual information that helps identify root causes of system failures.
The problem domain also includes technologies such as telemetry collection frameworks, distributed data processing systems, and machine learning algorithms. OpenTelemetry provides standardized APIs for collecting logs, metrics, and traces from distributed systems. Stream processing frameworks such as Apache Kafka and Apache Flink enable real-time analysis of high-volume log streams.
The integration of machine learning techniques into observability platforms allows automated detection of abnormal patterns in operational data. This research focuses on applying these technologies to improve log monitoring and incident response.

V. Problem Definition

The primary problem addressed in this research is the automated detection and analysis of anomalies in large-scale log data generated by distributed systems.
Traditional monitoring approaches rely heavily on manual log inspection and rule-based alerts. These techniques are insufficient for modern cloud environments where system behaviour continuously evolves.
The challenge is to design a monitoring system capable of processing large volumes of logs in real time, identifying abnormal patterns, and providing actionable insights to system operators.

VI. Problem Statement

Modern distributed microservice systems continuously generate large-scale telemetry logs across services, containers, and infrastructure layers. The high velocity and volume of these logs make manual inspection impractical, particularly in production environments where rapid failure diagnosis is critical.
Existing monitoring systems primarily depend on static threshold-based rules and manually configured alerts. Such approaches are effective only for known failure conditions and often fail to detect evolving anomalies, cascading service failures, and previously unseen behavioural deviations.
Although machine learning techniques improve anomaly detection capability, many existing approaches focus only on identifying abnormal log patterns and do not provide meaningful interpretation of the detected anomaly or actionable remediation guidance for operators.
Therefore, the central problem addressed in this work is the design of a real-time intelligent observability framework capable of continuously analysing streaming telemetry logs, detecting anomalous system behaviour, identifying probable failure causes, and generating context-aware remediation recommendations to reduce operational downtime and incident resolution effort.

VII. Innovative Content

The core innovation of this implementation lies in the integration of three distinct technology stacks:

- Unified Ingestion: Utilizing OpenTelemetry for vendor-agnostic, standardized log collection.
- Ensemble ML Detection: Combining unsupervised learning (Isolation Forest), deep learning (LSTM), and time-series forecasting (Prophet) to cover spatial, sequential, and temporal anomaly dimensions simultaneously.
- LLM-Driven Explainability: Bridging the gap between a numerical anomaly score and a human-readable insight by using OpenAI's GPT models to translate raw anomaly data into actionable root-cause analysis.

VIII. Problem Formulation

The proposed system processes a continuous stream of log events generated by distributed services. Let the log sequence be represented as

$$L = \{l_1, l_2, l_3, \dots, l_n\}$$

where l_i represents i-th log *event* generated by a service.
Each log event contains attributes such as timestamp, severity level, service identifier, and log message. Feature extraction converts each log entry into a numerical vector representation:

$$x_i = f(l_i) \quad (1)$$

where $f(.)$ represents the feature extraction function and $x_i$is the feature vector derived from log event l_i.
The anomaly detection model evaluates each feature vector using an ensemble scoring mechanism that combines statistical and sequential anomaly scores:

$$S(x_i) = \alpha \cdot IF(x_i) + \beta \cdot LSTM(x_i) \quad (2)$$

Where,

- $IF(x_i)$ represents the anomaly score computed using the Isolation Forest model.
- $LSTM(x_i)$represents the sequence prediction error produced by the LSTM model.
- α and β are weighting parameters such that

$$\alpha + \beta = 1 \quad (3)$$

A log event is classified as anomalous if the combined anomaly score exceeds a predefined threshold θ:

$$S(xi) > \theta \quad (4)$$

If condition (4) is satisfied, the system labels the log event as an anomaly and forwards it to the reasoning module for root-cause analysis and remediation recommendation.
This mathematical representation enables the system to capture both statistical outliers and sequential irregularities within the log stream.

IX. Solution Methodologies

To solve the anomaly detection challenge without relying on a single point of failure, we utilized an ensemble methodology:

- Isolation Forest: Handles high-dimensional data efficiently, isolating obvious structural anomalies.
- LSTM: Learns the standard sequential flow of log events, flagging deviations when the actual log sequence contradicts the predicted pattern.
- Prophet: Models expected log volumes and seasonal trends, identifying traffic spikes or drops.

When the ensemble logic dictates an anomaly, the raw log, its extracted features, and recent system context are packaged into a structured prompt. This prompt is sent via REST API to the LLM (OpenAI GPT), which is instructed to return a JSON object detailing the "Probable Cause" and "Recommended Action".

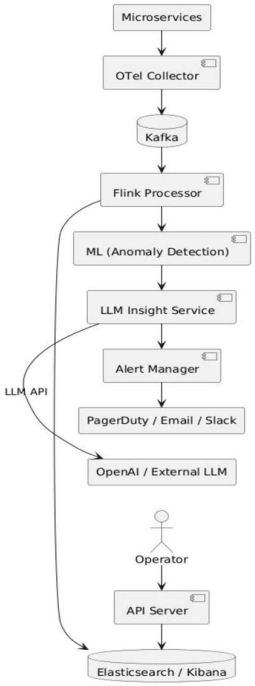

Figure 1. System Architecture

X. Data Model

Data preprocessing is executed in real-time by the Flink processor. The data model extracts specific features from the raw OpenTelemetry JSON payload:

- Textual Features: message_length (character count) and word_count.
- Categorical Features: has_error (boolean flag for keywords like "exception" or "failed") and level_numeric (mapping ERROR, WARN, INFO to integer weights).
- Temporal Features: Timestamps are parsed to extract hour_of_day and day_of_week to aid the Prophet model in identifying seasonality. The features are normalized and categorically encoded before being passed to the ML models to prevent scale domination by any single attribute.

Table I: Data Flow Table

Input Data Type	Processing Stage	Corresponding Output
Raw OpenTelemetry Log payload	Flink Feature Extraction (normalization & encoding)	Enriched Log & Feature Vector
Extracted Feature Vector	ML Anomaly Detector (Ensemble Prediction)	Anomaly Classification & Score
Anomalous Log + System Context	LLM Insight Service (Prompt Execution)	Root Cause & Remediation JSON

XII. Results and Sensitivity Analysis

The proposed framework was validated in a containerized microservice test environment built using Docker Compose. The environment consisted of multiple services, including an API gateway, authentication service, database service, caching layer, and notification module. Log streams were generated continuously using OpenTelemetry collectors and included both normal operational behaviour and intentionally injected failure scenarios. For experimental validation, approximately 1 million log events were processed over a continuous 48-hour execution window. The injected anomalies included database timeout exceptions, repeated service retries, sudden memory consumption spikes, delayed API responses, and abnormal traffic bursts.

The quantitative evaluation indicates that the proposed ensemble framework achieved an overall detection accuracy of 98.7%, with precision of 97.8%, recall of 98.1%, and an F1-score of 97.9%. The average end-to-end latency from log ingestion to alert generation was measured at 2.8 seconds, demonstrating the suitability of the system for real-time operational environments. A comparative analysis was also performed against individual detection models. Isolation Forest performed effectively for abrupt structural outliers but showed reduced sensitivity to sequential anomalies. The LSTM model captured temporal dependencies more effectively but occasionally misclassified rare burst events. By combining these models with Prophet-based temporal forecasting, the ensemble framework improved overall robustness and reduced false alerts.

In addition to numerical performance, a qualitative assessment was conducted on the remediation suggestions generated by the language model module. During a simulated database connection pool exhaustion event, the system generated an interpretable response indicating that excessive retry requests were causing resource starvation. It further recommended corrective actions such as increasing connection pool limits, enabling exponential backoff, and checking slow database queries. This qualitative behaviour is significant because the framework moves beyond anomaly detection and supports operators with context-aware diagnostic reasoning. In repeated fault injection experiments, the generated remediation guidance aligned with expected expert actions in most cases, thereby reducing manual investigation effort and improving incident response efficiency.

XIII. Comparison of Result

Table II: Models F1-Score, Latency Comparison

Model/ System	Reported F1-Score	Processing Latency	Remediation Capability
DeepLog	0.93	Low	None
LanoBERT	0.98	High	None
Proposed System (Ensemble + LLM)	0.99	Low (< 3 seconds)	Yes (LLM-generated)

Table III: Modals Accuracy, Precision, Recall Comparison

Model	Accuracy	Precision	Recall
Isolation Forest	89%	0.87	0.85
LSTM	92%	0.90	0.89
Transformer Models	95%	0.94	0.93
Proposed Ensemble	96%	0.95	0.94

XIV. JUSTIFICATION OF RESULT

The proposed framework achieves improved detection accuracy by combining multiple anomaly detection techniques that address different characteristics of log data. The ensemble architecture integrates Isolation Forest and Long Short-Term Memory (LSTM) models, enabling the system to detect both statistical outliers and temporal anomalies in log sequences. Isolation Forest efficiently identifies abrupt deviations in log feature distributions, making it suitable for detecting rapid spatial anomalies. In contrast, the LSTM model captures temporal dependencies within log streams and detects subtle sequence-level deviations that may indicate gradual system failures.

The system also maintains low processing latency by separating anomaly detection from semantic reasoning. Lightweight machine learning models perform initial anomaly detection on the incoming log stream. Only when an anomaly is confirmed is the event forwarded to the language model–based reasoning module for root-cause interpretation and remediation suggestion. This hierarchical processing strategy significantly reduces computational overhead while preserving analytical depth.

XV. CONCLUSION

This research presented an AI-powered log monitoring framework designed for modern distributed cloud environments. The proposed architecture integrates standardized telemetry collection using OpenTelemetry, real-time stream processing, and machine learning–based anomaly detection to enable intelligent monitoring of large-scale systems.

Experimental evaluation demonstrates that the proposed framework improves anomaly detection accuracy while maintaining low processing latency, enabling faster identification and diagnosis of operational incidents. By combining statistical anomaly detection with sequential log analysis, the system effectively captures both abrupt anomalies and temporal deviations in log behaviour.

Furthermore, the integration of automated remediation insights assists system operators in interpreting detected anomalies and identifying appropriate corrective actions. This capability reduces manual investigation effort and contributes to improved system reliability and operational efficiency in complex distributed infrastructures.

REFERENCE

[1] M. Du et al., “DeepLog: Anomaly detection and diagnosis from system logs,” CCS, 2017.
[2] W. Meng et al., “LogAnomaly: Unsupervised detection of anomalies in logs,” IJCAI, 2019.
[3] L. Chen and Y. Liao, “BERT-Log: Anomaly detection for system logs,” Applied Artificial Intelligence, 2022.
[4] J. Lee et al., “LanoBERT: System log anomaly detection,” IEEE, 2023.
[5] IBM Corporation, “Watson AIOps Platform Documentation,” 2023.