Augmenting Automatic Root-Cause Identification with Incident Alerts Using LLM

Komal Sarda[§]
York University
Toronto, Canada
komal253@yorku.ca

Zakeya Namrud[§]
York University
Toronto, Canada
Zakeya10@yorku.ca

Ian Watts
IBM Research
Toronto, Canada
ifwatts@ca.ibm.com

Larisa Shwartz
IBM Research
Yorktown Heights, USA
lshwart@us.ibm.com

Seema Nagar
IBM Research
Bangalore, India
senagar3@in.ibm.com

Prateeti Mohapatra
IBM Research
Bangalore, India
prateeti@in.ibm.com

Marin Litoiu
York University
Toronto, Canada
mlitoiu@yorku.ca

2024 34th International Conference on Collaborative Advances in Software and COmputiNg (CASCON) | 979-8-3315-0483-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/CASCON62161.2024.10838171

***Abstract*—Ensuring the reliability and availability of cloud services relies heavily on efficient root cause analysis (RCA) for cloud incidents. Traditionally, RCA involved labor-intensive manual investigations across various data sources, such as logs, metrics, and traces. Despite the increasing adoption of AI-driven assistants for RCA, their effectiveness for Site Reliability Engineers (SREs) is often hindered by low accuracy due to the inherent complexity of the task. This study introduces an on-call system powered by a large language model (LLM) designed to automate RCA processes for cloud incidents in practical, privacy-aware industrial settings. Our approach integrates incoming multimodal datasets (Logs, Metrics, Traces, Alerts-LMTA), aggregates critical runtime diagnostic information, including developer set alerts with precise offset values and thresholds, and utilizes LMTA data to predict the root cause category of incidents. We evaluated the effectiveness of our approach using two open-source, real-world-like datasets, demonstrating that the proposed LLM based approach can achieve an RCA accuracy of up to 97%.**

***Index Terms*—Root cause analysis, Cloud native applications, Large language models, Incident management**

I. Introduction

Large-scale cloud systems like AWS, Azure, and Google Cloud are crucial platforms for modern IT, supporting numerous applications and services globally, and generating substantial revenue [1]. Ensuring the reliability of these cloud services is essential as it directly impacts revenue and customer satisfaction [2]. Despite efforts to build robust systems, incidents within cloud infrastructures remain common, leading to service disruptions, significant economic losses, and negative impacts on user experience and trust [3], [4]. An IDC [54] survey reports that infrastructure failures can cost organizations between $1.25 and $2.5 billion annually. On average, an infrastructure failure incurs a cost of $100,000 per hour, while application failures can result in losses ranging from $500,000 to $1 million per hour. To address these challenges, the field of Artificial Intelligence for IT Operations (AIOps) has introduced various techniques to streamline incident management [35], [55]. However, many aspects of the incident management lifecycle still rely heavily on human intervention and manual investigation. One of the most important and challenging tasks is root cause analysis (RCA). Before resolving an incident, Site Reliability Engineers (SREs) must identify its root cause to ensure that corrective actions address the underlying issue comprehensively. RCA remains one of the most labor-intensive and skill-demanding components of the incident management lifecycle [6].

Various techniques have been proposed to assist SREs with RCA, such as incident prioritization and retrieval of similar historical incidents [36]. While earlier methods aimed at automating parts of the RCA process, the exceptional capabilities demonstrated by large language models (LLMs) in recent years have shifted the focus towards end-to-end RCA systems [57]. Recently, Ahmed et al. [7] proposed the use of fine-tuned LLMs for incident RCA and mitigation, finding that LLMs can identify root causes with minimal information. Their findings indicate that LLMs can identify the root causes of incidents even with minimal information about the incident. Chen et al. [8] introduced RCACopilot, an enhanced LLM-based RCA framework incorporating retrieval augmentation and diagnostic collection mechanisms. This involves developing tailored workflows specific to various incident types for systematic data collection. These datasets are used to predict the underlying root cause category, aiding SREs in thorough RCA. However, these methodologies do not enable LLMs to dynamically retrieve real-time diagnostic data related to the impacted service(s). RCACopilot requires manual engineering of predefined handlers and predicts root cause categories rather than individual root causes. Similarly, Ahmed et al. [7] rely solely on incident titles and descriptions for RCA.

Despite the evident efficacy displayed by LLM-based agents across various domains and tasks, their adoption for RCA presents a notable challenge. The proprietary nature of incident production data renders it largely unusable for LLMs without tailored refinement, a process that can incur substantial costs and logistical hurdles, particularly for expensive models [8].

[§] Both authors contributed equally to this work.

Authorized licensed use limited to: Tsinghua University. Downloaded on March 13,2025 at 07:59:58 UTC from IEEE Xplore. Restrictions apply.

While in-context examples offer a potential avenue for domain adaptation without explicit fine-tuning, the application of such examples for agent-driven RCA necessitates the crafting of comprehensive reasoning trajectories, a task fraught with complexity. Moreover, this challenge is compounded by the sophisticated prompting requirements of agents, which often demand additional fine-tuning [9] or reliance on in-context examples [10]. Additionally, most research focuses on metrics, logs, and trace datasets for RCA [39], [40]. In online service systems, observability tools and alert management systems are crucial for rapid fault detection and RCA. These systems continuously monitor for rule violations and trigger alerts upon breaches. However, minor faults in complex systems can cause numerous alerts, leading to an alert storm due to fault propagation across services [41]. Simultaneous independent faults can exacerbate this, generating mixed alerts from different sources. To mitigate this, existing approaches employ alert aggregation [42] and alert linking [43]. Alert aggregation consolidates alerts from the same root cause, while alert linking identifies relationships between alerts. However, using aggregated alerts for RCA remains unexplored.

This paper introduces RCA methodology designed to tackle some of the challenges of system diagnostics. We propose a framework capable of integrating diverse data streams, including system and application logs, metrics, traces, and alerts (LMTA) with the goal of identyfing the RCA fault type, e.g memory overload, CPU bottleneck, etc. Our approach uses alerts created by rules derived from metrics. We posit that the integration of alerts leads to superior diagnostic effectiveness compared to traditional methods relying solely on Key Performance Indicators (KPIs) or log data. To account for privacy-aware diagnosis, we evaluate both public and on-premises LLMs and answer the following research questions:

- **RQ1:** How effective is the proposed LMTA-based framework in identifying the root causes of incidents when leveraging integrated data streams?
- **RQ2:** What impact does excluding Alerts in prompts have on the LLMs' performance in identifying root causes through in-context learning?

The main contributions of this paper are: (1) We propose the novel use of LMTA data for incident RCA, leveraging data collection from multiple sources in a cohesive framework. (2) We propose an automated root cause type identification method that enables SREs to create incident-specific workflows for efficient data collection from multiple LMTA sources. (3) We demonstrate the integration of LLMs for autonomous analysis of diagnostic data, showcasing their capability in identifying incident root cause categories within the LMTA framework. (4) We evaluate LLM performance in RCA using synthetic alerts generated from two open-source datasets, assessing their effectiveness and feasibility in real-world scenarios.

The remainder of the paper is structured as follows: Section II presents the preliminary information, theoretical background and motivation. Section III outlines our approach and methodology. Section IV provides an experimental evaluation of our approach and discusses the results. Section V addresses practical considerations and identifies potential threats to validity. Section VI provides a review of relevant literature and discusses works related to our topic. Finally, Section VII concludes the paper and outlines potential directions for future research.

II. Background and Motivation

A. Dataset Description

This section offers a comprehensive overview of the fundamental concepts employed in our study: Logs, Metrics, and Traces. Each of these components plays a pivotal role in comprehending and analyzing root causes.

1) Metrics

In cloud-based microservice systems, metrics datasets are vital collections of quantitative data points essential for incident analysis and ongoing monitoring of system performance and user experience [16], [17]. These datasets include metric names, timestamps, and KPIs, offering detailed insights into system behavior. Metrics monitor service status, forming time series data from system metrics (e.g., CPU, memory utilization) and user-perceived metrics (e.g., response time, error rate) [15]. SREs utilize metrics to identify performance issues, detect anomalies, and optimize resource allocation, ensuring continuous system health assessment. Integrated with alerting mechanisms, metrics datasets trigger alerts when predefined thresholds are exceeded, promptly alerting administrators to potential issues and facilitating quick resolution to maintain system reliability [30], [36].

2) Logs

A log dataset comprises semi-structured records essential for incident analysis and system monitoring in cloud-based environments. It includes service logs (requests, responses, errors), system logs (OS events, resource usage), application logs (custom messages, events), and infrastructure logs (containers, cloud providers), all capturing operational statuses with timestamps, verbosity levels, and raw messages. Advanced anomaly detection methods like LogAnomaly [49] and Deeplog [44] use deep learning to detect deviations from normal patterns. Aggregating logs into a central repository and employing parsing and visualization techniques help extract insights, detect patterns, and identify anomalies. These insights, combined with system-level metrics (CPU, memory usage) and user-perceived metrics (response times), enable efficient incident management and support continuous system improvement efforts.

3) Traces

A trace in microservices architecture consists of spans representing service invocations, detailing start times, callers, callees, response times, and status codes. These spans track the execution paths of user requests across multiple service instances, facilitating the calculation of overall Response Time (RT). Anomaly detection methods like TraceAnomaly [18]

Authorized licensed use limited to: Tsinghua University. Downloaded on March 13,2025 at 07:59:58 UTC from IEEE Xplore. Restrictions apply.

detect deviations in response times or unusual invocation patterns. However, trace anomalies may not always indicate service failures and not all instance failures are captured, potentially leading to missed alerts. Integrating trace data with additional context enhances RCA. Traces are visualized as trees with microservices as nodes and invocations as edges, aiding operators in identifying problematic services and understanding request flows [19]. Alert generation from trace data involves setting RT thresholds or using anomaly detection algorithms to promptly alert operations teams to abnormal patterns, ensuring quick issue investigation and maintaining microservice architecture reliability.

4) Alerts

Alerts are critical notifications based on predefined thresholds or anomaly detection mechanisms, derived from LMT. Alerts capture anomalies and notify SREs of potential issues in online services. Service failures often cause an alert storm, overwhelming engineers and complicating manual investigation [61], [62]. By analyzing alerts, SREs can identify issues, create summaries, and develop mitigation plans to prevent future incidents. Previous studies used LMT for RCA. For instance, Zhao et al. [63] focused on instance failure prediction, while Zhang et al. [64] predicted anomaly types. Our approach extends LMT by incorporating alerts, forming LMTA, for RCA type prediction.

B. Incident Root Cause Analysis

Incident RCA in cloud-based microservices is essential for ensuring system reliability and swiftly resolving issues. Initially, incidents are detected through automated monitoring or customer reports [16], [56]. SREs then triage and assign incidents to engineering teams for detailed diagnosis, which involves correlating logs, metrics, and traces to identify abnormal patterns and pinpoint the specific service or component responsible [50], [51]. Automated tools utilizing machine learning and advanced analytics enhance the accuracy and efficiency of this process by analyzing datasets like LMT to detect subtle deviations indicative of potential issues. Once the root cause is identified, engineers devise mitigation strategies such as code rollbacks, hotfixes, or infrastructure adjustments, with careful consideration to prevent further issues [52]. Effective RCA not only resolves immediate problems but also enhances overall service resilience and reliability. Continuous improvement practices and post-incident reviews play a crucial role in learning from past incidents and preventing future occurrences.

C. The Opportunities and Challenges of Multi-Source Data in Incident Management

Most failure diagnosis approaches traditionally rely on single-modal data—traces, logs, or metrics—to identify root causes, using methods like machine learning to extract features from service invocations, log item vectors, or dependency graphs [37]. However, single-modal data is often insufficient because failures can impact multiple service aspects, and some failures may not be reflected in certain modalities. Therefore, combining traces, logs, and metrics (multimodal data) is essential for accurate diagnosis [39]. Moreover, the incorporation of alerts for RCA using LLMs remains an underexplored area. Incorporating alerts from metrics, logs, traces, and alerts enhances incident management by enabling early anomaly detection, comprehensive incident correlation across components [44], [46], and richer contextual insights for RCA. This integration boosts operational efficiency through the prioritization of critical alerts and the application of machine learning for predictive incident management. Continuous improvement is facilitated by analyzing alerts over time, refining monitoring strategies, and implementing preventive measures to enhance system reliability and resilience.

D. The Potential of LLMs

In recent years, LLMs like GPT-4 have become a significant trend in natural language processing. These models, characterized by billions of parameters, are trained on extensive and diverse textual corpora, including books, web texts, and Wikipedia articles. This comprehensive training allows LLMs to understand a wide range of prompts and queries, achieving high levels of accuracy and precision in addressing complex tasks [53]. LLMs have surpassed state-of-the-art models in various NLP tasks such as machine translation, question-answering, and other related activities. Notably, they have demonstrated that unsupervised language models, when trained with sufficient data, can match the performance of fine-tuned models even with minimal examples of new tasks, leading to significant performance improvements. Additionally, LLMs now feature token limits exceeding 32,000, enabling them to tackle complex questions by synthesizing information from a wide array of sources. They have shown impressive abilities to understand the context of downstream tasks and generate relevant information based on demonstrations, making them a potential choice for incident RCA [13], [14], [17] . However, determining the root cause of an incident is inherently complex, and LLMs may struggle to achieve optimal results on long-tail or domain-specific tasks without proper guidance.

III. Proposed Approach

Our proposed RCA framework, utilizing multimodal data termed LMTA, aims to significantly enhance the effectiveness and precision of LLMs through contextual training. LMTA integrates logs, metrics, traces, and alerts, facilitating comprehensive data capture, analysis, and utilization in real-time. This approach enables the development of more robust and context-aware AI systems. Figure 1 illustrates the framework, which is structured around three critical stages: (1) Data Integration and Mapping, where disparate data sources are consolidated and mapped for analysis; (2) Alert Synthesis and Prompt Engineering, which involves the creation of alerts and prompts based on the integrated data; and (3) Root Cause Type Identification, where the synthesized information is analyzed to identify the underlying causes.

Authorized licensed use limited to: Tsinghua University. Downloaded on March 13,2025 at 07:59:58 UTC from IEEE Xplore. Restrictions apply.

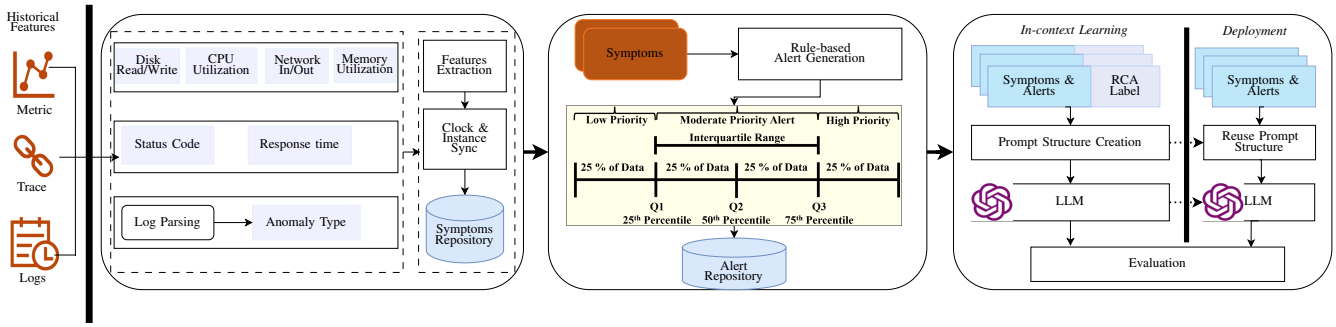

Fig. 1: Overall architecture for identifying root cause

A. Data Integration and Mapping

TABLE I: Sample of mapped LMT Dataset

Clock Synchronization of LMT Dataset	
Time	2021-07-14 00:01:00
Status code	500
Service name	Root
Target	webservice1
Metric	docker cpu system pct: 4.0, docker cpu total pct: 1.968019508515815, docker cpu user pct: 1.9075425790754257, docker memory usage pct: 0.0, docker memory stats active anon: 0.0, docker memory stats dirty: 0.0, docker memory stats total dirty: 0.0, response time: 328.03, docker cpu kernel pct: 0.0583941605839416
Response Time	861.55
Log	2021-07-14 00:01:00 | INFO | 0.0.0.1 | 172.17.0.3 | webservice1 | a1ca15f7a0bae3ad | call service:logservice1, inst:http://0.0.0.3:9384 as a downstream service
Alert details	This log entry indicates a service interaction where webservice1 made a call to logservice1, treated as a downstream service. Alert triggered due to critical thresholds being surpassed: - Docker CPU kernel percentage (0.058) exceeded the alert threshold. - Response time (328.03 ms) is above the alert threshold. These values indicate potential resource usage issues or performance bottlenecks.

In the realm of cloud-based microservice application incident management, the complexity of system failures often surpasses the capability of single-modal data to capture them adequately. This complexity is further compounded by the simultaneous manifestation of anomalous patterns across different modalities when failures occur. To address this challenge effectively, microservice systems and operators typically collect three types of monitoring data: traces, logs, and metrics. These data types are crucial within our incident management framework, particularly during the integration and mapping stage where they are serialized for unified analysis. Metrics, gathered as time-series data from microservice instances (instances from now on), are inherently serialized. Our approach focuses on selecting KPIs relevant to root cause detection, streamlining analysis to essential metrics. Parsing logs accurately and extracting log templates are critical for log serialization. Our emphasis lies in aligning logs with clock (timestamps) and associating them with specific instances to detect anomaly types and error messages effectively. Similarly, traces data undergoes alignment with clocks and instances, where each clock corresponds to tracing data, primarily RT, relevant to instance invocations. Synchronization of clocks across the three modal data types ensures post-serialization. Our primary objective is to discern the root cause type for a singular instance, with all monitoring data pertaining to that instance. Consequently, the clocks of the three modalities are synchronized relatively, facilitating cohesive analysis. Notably, metric data is collected at minute intervals, while log entries are generated upon instance events, and traces are recorded during request processing. Harmonizing these disparate datasets poses a notable challenge, yet it is a prerequisite for effective incident management. Once dataset mapping is achieved, we proceed with alert generation, utilizing the synthesized information to identify and address potential incidents promptly. Table I provides an example of mapped data, including a detailed description.

B. Alert Generation and Prompt Engineering

Central to generating alerts are Service Level Indicators (SLIs) and Service Level Objectives (SLOs). SLIs, also known as KPIs, quantify metrics such as latency, error rates, and availability. SLOs set target thresholds for these metrics; for example, ensuring the 95th percentile response time remains below 200 milliseconds 99.9% of the time. Defining SLIs and SLOs is pivotal as they underpin monitoring and observability, enabling continuous assessment of microservice performance against objectives. Deviations trigger alerts or automated actions, allowing proactive issue resolution before impacting system performance or user experience. Our research identified that available public datasets lack alerts, motivating us to create alerts from existing multimodal datasets (metrics, logs, traces). To achieve this, similarly to production environments, we adopted a rule-based approach to detect KPIs aligned with SLOs. This involves defining rules or conditions triggering alerts based on KPI thresholds. For example, an alert might trigger if error rates exceed a set threshold or latency surpasses an acceptable limit.

Due to the absence of SLOs in open-source datasets, we adapted percentile-based thresholds for each KPI metric, including response time (RT). This method categorizes metrics based on specific percentiles: below the 25th percentile,

Authorized licensed use limited to: Tsinghua University. Downloaded on March 13,2025 at 07:59:58 UTC from IEEE Xplore. Restrictions apply.

between the 25th and 75th percentiles, and above the 75th percentile. This approach effectively highlights metric severity concerning anomalies, filtering out anomalous values and focusing on significant contributors to anomalies. Initially, the independent generation of metric, log, and trace data resulted in numerous empty metrics within alerts and inclusion of irrelevant data points, unnecessarily consuming prompt length (as illustrated in Table I). To address this, we implemented filtering mechanisms to exclude missing values and enhance the relevance of alerts. The benefits of our percentile-based approach include eliminating empty metrics, pinpointing specific anomaly-contributing metrics, and enhancing incident management efficiency in cloud-based microservice applications. The use of percentiles involves computing the values that demarcate the boundaries for different severity levels. For instance, the 25th percentile denotes the value below which 25% of the data points lie, while the 75th percentile signifies the value below which 75% of the data points fall. To classify metrics based on these percentiles, we employ the following formula:

$$P_k = \frac{k}{100}(n+1)$$

Where P_k represents the k-th percentile, n denotes the total number of data points. Once we compute the percentiles, we can categorize metrics as follows:

- Metrics falling below the 25th percentile are considered to indicate low severity.
- Metrics lying between the 25th and 75th percentiles signify moderate severity.
- Metrics exceeding the 75th percentile denote high severity.

This classification scheme enables the system to effectively discern the severity levels of anomalies and prioritize the analysis of metrics contributing significantly to these anomalies, optimizing incident management processes. The alerts generated serve as a component in the prompts for the LLM (LMTA in context), triggering its analysis to identify potential root causes. We have created a comprehensive alert dataset that closely resembles real-world scenarios. Our strategy is to use an in-context tuning approach where examples of incidents and their LMTA are labeled with their root cause types. These examples are incorporated into the prompt to provide contextual information, helping the model better understand the relationship between incidents and their root causes. By including these examples, we aim to significantly enhance its ability to accurately predict and analyze root causes based on the given incident data. Table II presents illustrative examples featuring incident information, in-context examples, and LMTA details. Our alerts datasets, including alerts and the ground truth data used in this paper, are available at the following link [1]. This resource provides comprehensive access to the data of our study.

[1]https://github.com/LMTA-LLMs/LMTA

TABLE II: Prompt with Incident Information, In-context Examples, and LMTA Details

Task Description:
As a Site Reliability Engineers, you are responsible for identifying the root cause type by analyzing the provided data.
You will be provided with a prompt that includes timestamp, source and target, metrics, and logs. The provided metrics fall into three categories:
- alert_25: Metric is less than the 25th percentile and is relatively low.
- alert_25_75: Metric is between the 25th and 75th percentiles and has increased moderately.
- alert_75: Metric is greater than the 75th percentile and indicates the highest severity of the metric.
Just provide the root cause type without additional details, following the historical examples stated below. Complete the last one based on the provided data.
Historical Examples:
Example 1
Timestamp: 2021-07-14 00:01:00, Source: root, Target: webservice1, Log: 2021-07-14 00:01:00 | INFO | 0.0.0.1 | 172.17.0.3 | webservice1 | a1ca15f7a0bae3ad | call service:logservice1, inst:http://0.0.0.3:9384 as a downstream service, alert_0_25: docker cpu system pct: 4.0, docker cpu total pct: 1.968, docker cpu user pct: 1.908, alert_25_75: response time: 328.03, alert_75: docker cpu kernel pct: 0.058
Root Cause Type: Misconfiguration

C. Root Cause Type Identification

To identify the root cause, our first step involves creating a prompt for the LLM using the retrieved in-context examples. As shown in Figure 1, the prompt includes the task definition, in-context examples, and the LMT of the new incident. To commence, we outline the task of root cause analysis, directing the LLM to assume the role of a software engineer. Subsequently, we present a collection of pertinent in-context examples, meticulously arranged with their respective timestamps, source names, and alerts derived from LMT, each paired with its corresponding root cause. It is pertinent to highlight our method of leveraging alerts based on LMTA incidents and their root causes within the examples. This approach optimizes prompt space utilization while ensuring the retention of essential incident information, serving as a valuable reference for addressing future incidents. The prompt encompasses the task definition, in-context examples, and the LMTA associated with the new incident.

D. Baseline Models

In this section, we outline the baselines utilized for our evaluation in RQ1 and RQ2. We focus exclusively on BAM models sourced from IBM resources and GPT-4o. All the baselines employ in-context tuning and utilize an identical number of samples for both the few-shot learning and testing stages. Except GPT-4o, all the other model were tested on IBM BAM platform [60].

1) **Granite.13b.instruct-v1** is an instruction-tuned Supervised Fine-Tuning (SFT) model that has undergone additional refinement using a diverse range of datasets. These include the Flan Collection, 15,000 samples from Dolly, Anthropic's human preference data on helpfulness and harmlessness, Instructv3, and approximately

Authorized licensed use limited to: Tsinghua University. Downloaded on March 13,2025 at 07:59:58 UTC from IEEE Xplore. Restrictions apply.

700,000 internally generated synthetic datasets tailored for summarization and dialog tasks.

2) **Codellama-34b-instruct** is part of the Code Llama collection, featuring generative text models that have been both pretrained and fine-tuned. These models vary in size, with parameters ranging from 7 billion to 34 billion [23].
3) **Mistral-7B-Instruct-v0.1** is a LLM that has been fine-tuned for instructional tasks, based on the Mistral-7B-v0.1 generative text model. It utilizes various publicly available conversational datasets for its training. This model is capable of supporting function calls and operating in JSON mode.
4) **Mixtral-8x7B Instruct v0.1** is an advanced AI model specifically engineered for instruction-following tasks. Boasting an impressive 56 billion parameters, it excels at comprehending and executing intricate instructions, delivering precise and relevant responses across diverse contexts [21].
5) **GPT-4o** is our latest generation model, boasting enhanced capabilities and an updated knowledge cutoff as of April 2023. It features an impressive 128k context window, which equates to 300 pages of text in a single prompt. Additionally, the model can generate up to 4096 output tokens [22].

E. Datasets

Here, we provide a comprehensive overview of the datasets utilized in our study, detailing their characteristics and sources.

1) MicroSS Dataset

TABLE III: Anomaly and Ground truth information of MicroSS dataset

Anomaly type	**RCA(Ground truth_Categories)**
System stuck	High memory Usage
Process crash	Incorrect deallocation
Login failure	Network interuption
File missing	Code bug
Access denied	Misconfiguration

The MicroSS [59] dataset, sourced from the Generic AIOps Atlas (GAIA), comprises multimodal data collected from a simulated environment featuring ten microservices, alongside two database services (MySQL and Redis), distributed across five host machines, and catering to both mobile and PC users. Intentionally subjected to various failure scenarios encompassing system failures like System Stuck and Process Crash, as well as service failures such as Login Failure, File Missing, and Access Denied, each failure type exhibits distinct symptoms challenging for single-modal diagnostic methods to discern effectively. With over 0.7 million metrics, 87 million logs, and 28 million traces collected over a two-week period, the dataset highlights both the diversity of failure manifestations and the discrepancies in their occurrence frequencies, emphasizing the necessity for a multimodal approach to failure diagnosis and system maintenance in complex distributed environments [48]. Table III provides a comprehensive breakdown of anomalies in the MicroSS dataset along with their corresponding ground truth information.

2) Multi-Source Distributed System (MSDS) Dataset

This recent high-quality, multi-source dataset is composed of distributed traces, application logs, and metrics from a complex distributed system [47]. Specifically built for AI operations, this dataset supports automated anomaly detection, root cause analysis, and remediation. Named MSDS [58], this dataset is meticulously designed for artificial intelligence (AI) operations, including automated anomaly detection. It encompasses distributed traces, application logs, and metrics generated from running a complex distributed system (OpenStack) . The dataset features sequential data produced by executing the workload of sequential user requests, and concurrent data generated by executing the workload of concurrent user requests. Additionally, the provided workload and fault scripts, along with the Rally report, serve as ground truth for various analyses. The MSDS dataset lacks predefined anomaly types or root causes as ground truth. Therefore, we thoroughly analyze the dataset and create detailed root cause categories as presented in Table IV within the MSDS dataset. These categories serve as the basis for labeling the dataset.

TABLE IV: Ground truth information of MSDS dataset

RCA(Ground truth_Categories)
Resource overload
Network/Configuration issues
Network latency/Server overload
I/O Bottlenecks
Image actively utilized

IV. Evaluation Results

In this section, we provide a comprehensive explanation of the results obtained from our study. Additionally, we detail the performance metrics employed to validate and assess the effectiveness of our research findings.

A. Evaluation Metrics

For RCA, several measures have been proposed to assess the quality of RCA algorithms. In our approach, we manually establish a ground truth for each dataset to assess the performance of LLM predictions in identifying the root cause. In general, precision and recall are both considered as evaluation criteria to ensure a comprehensive assessment of the detection methods. Precision measures the proportion of relevant instances among the retrieved instances, and recall measures the proportion of relevant instances that were successfully retrieved. Let P and N represent the number of actual positive and negative points, while TP, FP, TN, and FN denote true positive, false positive, true negative, and false negative

Authorized licensed use limited to: Tsinghua University. Downloaded on March 13,2025 at 07:59:58 UTC from IEEE Xplore. Restrictions apply.

classifications, respectively. The following metrics are then defined:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{P} \tag{2}$$

$$F_1 = \frac{TP}{TP + FN} \tag{3}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

B. RQ1 AND RQ2 RESULTS

RQ1: To what extent do LLM-based agents demonstrate efficacy in identifying the root causes of incidents when prompted by LMTA?

Table V showcases the effectiveness of our in-context learning approach using the GPT-4o and Bam models. We compared five models: Granite.13b.instruct, Codellama-34b-instruct, Mistral-7B-Instruct-v0.1, Mixtral-8x7B Instruct v0.1, and GPT-4o. Our methodology involved an in-context learning model with a few-shot samples setting using the MicroSS dataset, allowing a maximum of 15 samples for training and 30 samples for testing. The results indicate the following conclusions: The GPT-4 and Mixtral-8x7b-instruct-v01 models demonstrate exceptional performance, surpassing other Bam models by an average of 97% across four different metrics. While some LLMs like Codellama-34b-instruct perform well, others such as Granite-13b-instruct-v2 face challenges in accurately capturing and generating content in the given context. The high performance of GPT-4 and Mixtral-8x7b-instruct-v01 across all metrics indicates their robustness and effectiveness in understanding and generating content in the specified domain. These results highlight the influence of model architecture and training data on performance. Models like GPT-4 and Mixtral-8x7b-instruct-v01, which likely benefit from larger architectures and extensive training, tend to outperform others across various metrics.

TABLE V: Evaluation generalizability and adaptability of LLMs With alerts for RCA categories

LLMs	Precision	Recall	F1	Accuracy
MicroSS Dataset				
Granite-13b-instruct-v2	0.66	0.71	0.65	0.83
Mixtral-8x7b-instruct-v01	0.96	0.97	0.96	0.93
Mistral-7b-v0-1	0.65	0.78	0.58	0.62
Codellama-34b-instruct	0.72	0.87	0.71	0.72
GPT-4o	**0.97**	**0.98**	**0.98**	**0.97**
MSDS Dataset				
Granite-13b-instruct-v2	0.79	0.80	0.77	0.77
Mixtral-8x7b-instruct-v01	0.82	0.86	0.84	0.93
Mistral-7b-v0-1	0.80	0.73	0.72	0.73
Codellama-34b-instruct	0.72	0.78	0.73	0.73
GPT-4o	**0.99**	**0.97**	**0.98**	**0.97**

Finding RQ1

- Multi-modal data that includes alerts, together with LLMs, can achieve very good performance for RCA tasks. Public models such as GPT-4o have better Precision, Recall, F1 score, and Accuracy (0.97), however, on-premises and smaller models, such as Mixtral-8x7b-instruct-v01, achieve very good performance, too (0.93).

RQ2: How does excluding Alerts in prompts impact the identification of root cause performance through in-context learning in LLMs?

To showcase the efficacy of Alerts in identifying root causes and improving adaptability for LLMs, we conducted an ablation study using the two datasets, MicroSS and MSDS and excluding the Alerts from prompts. We evaluated the same five distinct LLMs as in the first experiment. Table VI illustrate that excluding Alerts from the prompt reduces the models' performance for both datasets. The Mixstral Model exhibits the highest performance among all models without Alerts; however, its performance decreases by 10% compared to when Alerts are included. Figure 2 illustrates the impact of alerts on various LLMs, including Granite-13b, Mixtral-8x7b, Mistral-7b, Codellama-34b, and GPT-4o. The figure displays SD and Mean values for key performance metrics: precision, recall, F1 score, and accuracy, across these models.

TABLE VI: Evaluation generalizability and adaptability of LLMs without Alerts

LLMs	Precision	Recall	F1	Accuracy
MicroSS Dataset				
Granite-13b-instruct-v2	0.51	0.49	0.37	0.43
Mixtral-8x7b-instruct-v01	**0.92**	**0.92**	**0.90**	**0.83**
Mistral-7b-v0-1	0.46	0.63	0.38	0.37
Codellama-34b-instruct	0.71	0.79	0.67	0.60
GPT-4o	0.85	0.87	0.85	0.77
MSDS Dataset				
Granite-13b-instruct-v2	0.61	0.39	0.40	0.60
Mixtral-8x7b-instruct-v01	0.63	0.61	0.58	0.57
Mistral-7b-v0-1	0.31	0.34	0.28	0.40
Codellama-34b-instruct	**0.70**	**0.77**	**0.71**	**0.70**
GPT-4o	0.40	0.46	0.42	0.60

Finding RQ2

- Prompts without Alerts decrease significantly the models' performance in identifying the fault types. Among all models, the Mixstral model performs the best without Alerts, but its performance still drops by 10% compared to its results with Alerts included. This suggests that Alerts are essential in achieving good performance in identifying the fault category.

Authorized licensed use limited to: Tsinghua University. Downloaded on March 13,2025 at 07:59:58 UTC from IEEE Xplore. Restrictions apply.

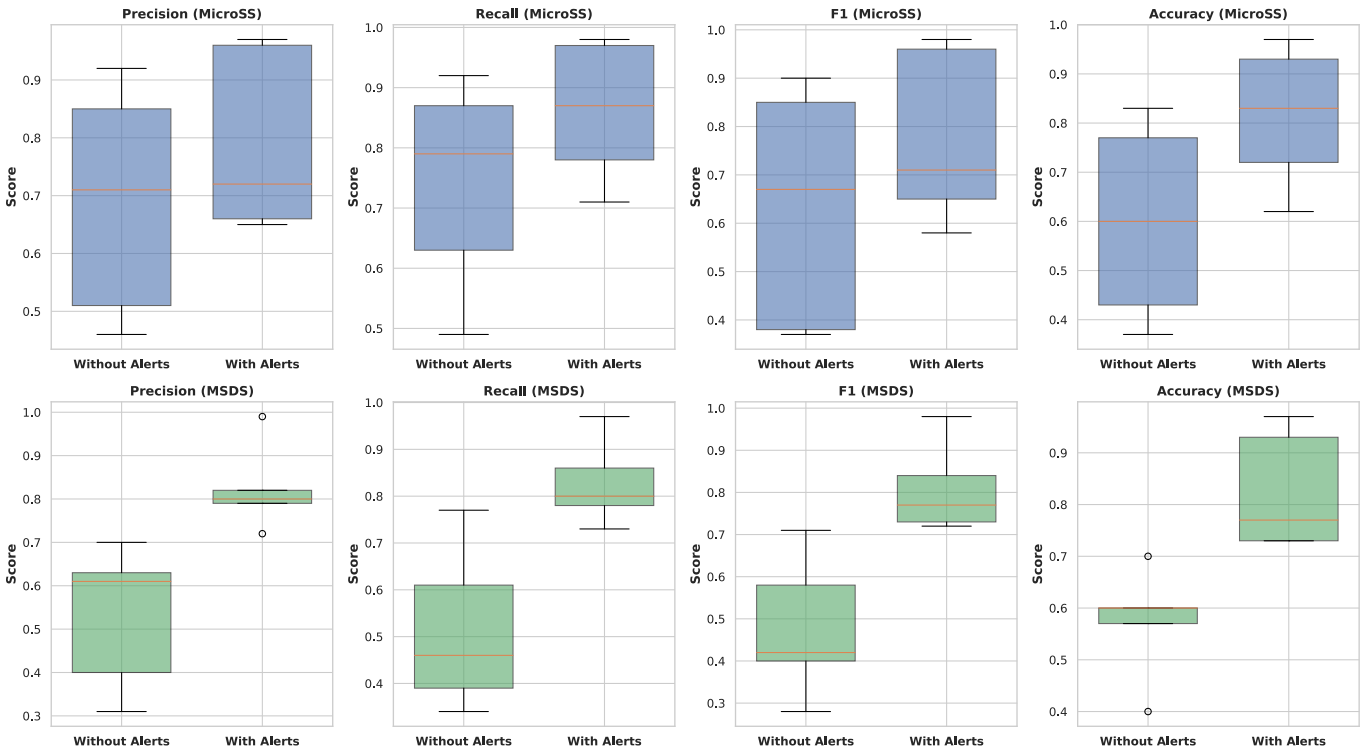

Fig. 2: Impact of Alerts on LLMs (Granite-13b, Mixtral-8x7b, Mistral-7b, Codellama-34b, GPT-4o): Standard Deviation and Mean of Metrics (Precision, Recall, F1, Accuracy) across the models.

V. Discussion/Threats to Validity

Our proposed approach used GPT-4 and models from IBM's WatsonX to explore their effectiveness in identifying the category of fault. Other models might yield a different performance. To evaluate our methodology, we utilized two open incident datasets, MicroSS and MSDS. The quality of the root cause categories, crucial for LLM effectiveness, relies on manual labeling. We manually labeled the MicroSS dataset, including its anomaly types, to establish the ground truth. For the MSDS dataset, we meticulously analyzed each sample to determine accurate labeling. Other datasets and root cause categories might produce different results. Our experimental setup involved testing LLMs across three different alert thresholds to gauge their effectiveness. Furthermore, the accuracy and effectiveness of alert generation are system, application, and requirement-specific. In our study, we concentrated on generating alerts for performance, network, and misconfiguration issues. The applicability of our approach to other types of applications may vary.

VI. Related Work

Root cause analysis has emerged as a significant area of research within the system and software engineering communities, particularly in the context of large cloud services [23]–[26]. The goal is to identify the root causes of failures and performance issues by analyzing diverse data sources, including metrics, logs, and traces. Previous studies have introduced various methods for root cause analysis, each utilizing one of these data sources. For instance, some approaches utilize logs to analyze a subset of log messages [30] or to examine the specifics within each log message [31]. Other methods depend on metrics to identify failure patterns [32] or to create service dependency graphs [33]. Additionally, some methods employ tracing to identify faulty services [34]. AnoFusion [16] proposed an unsupervised approach to analyzing metric data along with other modalities, effectively reducing false alarms and improving incident management in cloud-based microservice systems. Unlike previous approaches, we have developed a system that automatically integrates metrics, logs, and traces to create alerts for root cause analysis, leveraging LLMs.

The ascent of LLM in recent years has ushered in a wave of fresh opportunities within the realm of software systems.Many researchers have focused on using LLMs for automating software engineering tasks like identifying code bug locations, as demonstrated by AutoFL [11]. However, limited research exists on utilizing LLMs for RCA in cloud incidents. RCA-CoPilot [8] enhances RCA for on-call engineers by collecting runtime diagnostic information, summarizing incidents, and predicting root causes using LLMs, improving flexibility and scalability in incident management. PACE-LM [13] addresses LLM accuracy issues by retrieving relevant historical incidents and using a confidence-of-evaluation (COE) pre-examination and root cause evaluation (RCE) process, utilizing models like

Authorized licensed use limited to: Tsinghua University. Downloaded on March 13,2025 at 07:59:58 UTC from IEEE Xplore. Restrictions apply.

TABLE VII: Comparison of Works for RCA in Cloud-Based Microservice Incident Management

Study	Techniques Used	Dataset	Data modality	Performance
Paper [12]	In-context learning (GPT-4)	CompanyX	Incident Summary	84.96%
Paper [13]	In-context learning (GPT-4)	CompanyX	Incident Summary	43.40%
Paper [14]	In-context learning (GPT-4-0613)	Alibaba Cloud	Logs	74.53%
Paper [17]	In-context learning (GPT-3.5-turbo & GPT-4)	Microsoft	Upstream Dependency failure	86.65%
Our work	In-context learning (GPT-4o & Granite.13b.instruct-v1, Codellama-34b-instruct, Mistral-7b-v0-1,mixtral-8x7b-instruct-v01) from Bam (IBM)	MicroSS & MSDS	Logs, Metrics, Traces, Alerts (LMTA)	97%

GPT-4o for confidence estimation. RCAgent [14] leverages AIOps for RCA to reduce MTTR in cloud systems, using enhanced prompting cycles and interactive environments, demonstrated on Alibaba Cloud. X-lifecycle [17] focuses on dependency failure incidents, enhancing LLM prompts with contextual information to improve root cause recommendations, and comparing different strategies to enhance prediction performance, particularly for identifying resource and service level objective classes. Table VII provides a comprehensive comparison between state-of-the-art methods and our approach, detailing the techniques and LLMs utilized, the datasets employed, data modalities considered, and the highest achieved performance metrics.

VII. Conclusion and Future plan

Ensuring the dependability and accessibility of cloud services heavily relies on the efficacy of root cause analysis (RCA) for cloud incidents. Traditionally, RCA has been a manual and resource-intensive process, involving the examination of diverse data sources such as logs, metrics, and traces. While AI-driven assistants have gained traction for RCA, their effectiveness for on-call engineers often suffers due to inherent complexity, leading to decreased precision. This research introduces an on-call system powered by an advanced large language model (LLM), customized to automate RCA procedures for cloud incidents within pragmatic and privacy-conscious industrial settings. Our approach entails aligning varied incoming datasets, consolidating critical runtime diagnostic data, issuing alerts with precise offsets and thresholds, and predicting the root cause category of incidents. Through evaluation using open-source datasets that mimic real-world conditions, we illustrate that our LLM achieves RCA accuracy of up to 97%. Looking forward, our work aims to explore the potential of LLMs in providing comprehensive insights into compromised servers, including summaries and recommendations for mitigation strategies.

References

[1] Dillon, Tharam, Chen Wu, and Elizabeth Chang. "Cloud computing: issues and challenges." 2010 24th IEEE international conference on advanced information networking and applications. Ieee, 2010.

[2] YAN, Xiaohan, et al. Aegis: Attribution of Control Plane Change Impact across Layers and Components for Cloud Systems. In: 2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, 2023. p. 222-233.

[3] GROBAUER, Bernd; SCHRECK, Thomas. Towards incident handling in the cloud: challenges and approaches. In: Proceedings of the 2010 ACM workshop on Cloud computing security workshop. 2010. p. 77-86.

[4] JIN, Pengxiang, et al. Assess and summarize: Improve outage understanding with large language models. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2023. p. 1657-1668.

[5] WOLFE, Sean. Amazon's one hour of downtime on prime day may have cost it up to $100 million in lost sales. Business Insider (July 2018). URL: https://www. businessinsider. com/amazon-prime-day-website-issues-cost-itmillions-in-lost-sales-2018-7, 2018.

[6] MA, Minghua, et al. Diagnosing root causes of intermittent slow queries in cloud databases. Proceedings of the VLDB Endowment, 2020, 13.8: 1176-1189.

[7] AHMED, Toufique, et al. Recommending root-cause and mitigation steps for cloud incidents using large language models. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). IEEE, 2023. p. 1737-1749.

[8] Chen, Y., Xie, H., Ma, M., Kang, Y., Gao, X., Shi, L., Cao, Y., Gao, X., Fan, H.,Wen, M., Zeng, J., Ghosh, S., Zhang, X., Zhang, C., Lin, Q., Rajmohan, S., and Zhang, D. Empowering practical root cause analysis by large language models for cloud incidents.

[9] SONG, Chan Hee, et al. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023. p. 2998-3009.

[10] YAO, Shunyu, et al. React: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629, 2022.

[11] S. Kang, G. An, and S. Yoo, 'A preliminary evaluation of llm-based fault localization', arXiv preprint arXiv:2308. 05487, 2023.

[12] X. Zhang et al., 'Automated Root Causing of Cloud Incidents using In-Context Learning with GPT-4', arXiv preprint arXiv:2401. 13810, 2024.

[13] D. Zhang, X. Zhang, C. Bansal, P. Las-Casas, R. Fonseca, and S. Rajmohan, 'PACE: Prompting and Augmentation for Calibrated Confidence Estimation with GPT-4 in Cloud Incident Root Cause Analysis', arXiv preprint arXiv:2309. 05833, 2023.

[14] Z. Wang et al., 'RCAgent: Cloud Root Cause Analysis by Autonomous Agents with Tool-Augmented Large Language Models', arXiv preprint arXiv:2310. 16340, 2023.

[15] S. Zhang et al., 'Robust failure diagnosis of microservice system through multimodal data', IEEE Transactions on Services Computing, 2023.

[16] C. Zhao et al., 'Robust multimodal failure detection for microservice systems', in Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 5639–5649.

[17] D. Goel et al., 'X-lifecycle Learning for Cloud Incident Management using LLMs', arXiv preprint arXiv:2404. 03662, 2024.

[18] P. Liu et al., 'Unsupervised detection of microservice trace anomalies through service-level deep bayesian networks', in 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), 2020, pp. 48–58.

[19] B. H. Sigelman et al., 'Dapper, a large-scale distributed systems tracing infrastructure', 2010.

[20] ACHIAM, Josh, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.

[21] JIANG, Albert Q., et al. Mixtral of experts. arXiv preprint arXiv:2401.04088, 2024.

[22] Sanderson K. GPT-4 is here: what scientists think. Nature. 2023 Mar 30;615(7954):773.

[23] Qin Y, Hu S, Lin Y, Chen W, Ding N, Cui G, Zeng Z, Huang Y, Xiao C, Han C, Fung YR. Tool learning with foundation models. arXiv preprint arXiv:2304.08354. 2023 Apr 17.

Authorized licensed use limited to: Tsinghua University. Downloaded on March 13,2025 at 07:59:58 UTC from IEEE Xplore. Restrictions apply.

[24] Alquraan A, Takruri H, Alfatafta M, Al-Kiswany S. An analysis of Network-Partitioning failures in cloud systems. In13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18) 2018 (pp. 51-68).

[25] Chen H, Dou W, Jiang Y, Qin F. Understanding exception-related bugs in large-scale cloud systems. In2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) 2019 Nov 11 (pp. 339-351). IEEE.

[26] Gao Y, Dou W, Qin F, Gao C, Wang D, Wei J, Huang R, Zhou L, Wu Y. An empirical study on crash recovery bugs in large-scale distributed systems. InProceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering 2018 Oct 26 (pp. 539-550).

[27] Zhang Y, Yang J, Jin Z, Sethi U, Rodrigues K, Lu S, Yuan D. Understanding and detecting software upgrade failures in distributed systems. InProceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles 2021 Oct 26 (pp. 116-131).

[28] Lou C, Huang P, Smith S. Understanding, detecting and localizing partial failures in large system software. In17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20) 2020 (pp. 559-574).

[29] Ma M, Yin Z, Zhang S, Wang S, Zheng C, Jiang X, Hu H, Luo C, Li Y, Qiu N, Li F. Diagnosing root causes of intermittent slow queries in cloud databases. Proceedings of the VLDB Endowment. 2020 Apr 1;13(8):1176-89.

[30] Zhang X, Xu Y, Qin S, He S, Qiao B, Li Z, Zhang H, Li X, Dang Y, Lin Q, Chintalapati M. Onion: identifying incident-indicating logs for cloud systems. InProceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering 2021 Aug 20 (pp. 1253-1263).

[31] Li Z, Luo C, Chen TH, Shang W, He S, Lin Q, Zhang D. Did we miss something important? studying and exploring variable-aware log abstraction. In2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) 2023 May 14 (pp. 830-842). IEEE.

[32] Zhang Y, Guan Z, Qian H, Xu L, Liu H, Wen Q, Sun L, Jiang J, Fan L, Ke M. CloudRCA: A root cause analysis framework for cloud computing platforms. InProceedings of the 30th ACM International Conference on Information & Knowledge Management 2021 Oct 26 (pp. 4373-4382).

[33] Li M, Ma M, Nie X, Yin K, Cao L, Wen X, Yuan Z, Wu D, Li G, Liu W, Yang X. Mining fluctuation propagation graph among time series with active learning. InInternational Conference on Database and Expert Systems Applications 2022 Jul 29 (pp. 220-233). Cham: Springer International Publishing.

[34] Xie Z, Xu H, Chen W, Li W, Jiang H, Su L, Wang H, Pei D. Unsupervised anomaly detection on microservice traces through graph vae. InProceedings of the ACM Web Conference 2023 2023 Apr 30 (pp. 2874-2884).

[35] CHEN, Junjie, et al. How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. 2020. p. 373-384.

[36] YU, Qingyang, et al. A survey on intelligent management of alerts and incidents in IT services. Journal of Network and Computer Applications, 2024, 103842.

[37] Liu P, Tao Q, Zhou JT. Evolving from Single-modal to Multi-modal Facial Deepfake Detection: A Survey. arXiv preprint arXiv:2406.06965. 2024 Jun 11.

[38] Ma M, Zhang S, Chen J, Xu J, Li H, Lin Y, Nie X, Zhou B, Wang Y, Pei D. Jump-Starting multivariate time series anomaly detection for online service systems. In2021 USENIX Annual Technical Conference (USENIX ATC 21) 2021 (pp. 413-426).

[39] C. Zhao et al., 'Robust multimodal failure detection for microservice systems', in Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 5639–5649.

[40] S. Zhang et al., 'Robust failure diagnosis of microservice system through multimodal data', IEEE Transactions on Services Computing, 2023.

[41] Z. Chen et al., 'Towards intelligent incident management: why we need it and how we make it', in Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event USA, 2020.

[42] Z. Chen et al., 'Graph-based incident aggregation for large-scale online service systems', in 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2021, pp. 430–442.

[43] Y. Chen et al., 'Dynamic Graph Neural Networks-Based Alert Link Prediction for Online Service Systems', in 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2023, pp. 79–90.

[44] Yuan Y, Adhatarao SS, Lin M, Yuan Y, Liu Z, Fu X. Ada: Adaptive deep log anomaly detector. InIEEE INFOCOM 2020-IEEE Conference on Computer Communications 2020 Jul 6 (pp. 2449-2458). IEEE.

[45] Vidal JM, Orozco AL, Villalba LJ. Alert correlation framework for malware detection by anomaly-based packet payload analysis. Journal of Network and Computer Applications. 2017 Nov 1;97:11-22.

[46] Remil Y, Bendimerad A, Mathonat R, Kaytoue M. Aiops solutions for incident management: Technical guidelines and a comprehensive literature review. arXiv preprint arXiv:2404.01363. 2024 Apr 1.

[47] , Sasho, et al. Multi-source distributed system data for ai-powered analytics. In: Service-Oriented and Cloud Computing: 8th IFIP WG 2.14 European Conference, ESOCC 2020, Heraklion, Crete, Greece, September 28–30, 2020, Proceedings 8. Springer International Publishing, 2020. p. 161-176.

[48] DELCHAMBRE, L., et al. Gaia Data Release 3-Apsis. III. Non-stellar content and source classification. Astronomy & Astrophysics, 2023, 674: A31.

[49] W. Meng et al., 'Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs', in IJCAI, 2019, vol. 19, pp. 4739–4745.

[50] YU, Guangba; HUANG, Zicheng; CHEN, Pengfei. TraceRank: Abnormal service localization with dis-aggregated end-to-end tracing data in cloud native systems. Journal of Software: Evolution and Process, 2023, 35.10: e2413.

[51] CAI, Zhengong, et al. A real-time trace-level root-cause diagnosis system in alibaba datacenters. IEEE Access, 2019, 7: 142692-142702.

[52] GHOSH, Supriyo, et al. How to fight production incidents? an empirical study on a large-scale cloud service. In: Proceedings of the 13th Symposium on Cloud Computing. 2022. p. 126-141.

[53] LIU, Yiheng, et al. Understanding llms: A comprehensive overview from training to inference. arXiv preprint arXiv:2401.02038, 2024.

[54] IDC, "IDC: The premier global market intelligence firm.," IDC: The premier global market intelligence company, 2019. https://www.idc.com/

[55] Z. Namrud, K. Sarda, M. Litoiu, L. Shwartz, and I. Watts, 'KubePlaybook: A Repository of Ansible Playbooks for Kubernetes Auto-Remediation with LLMs', in Companion of the 15th ACM/SPEC International Conference on Performance Engineering, 2024, pp. 57–61.

[56] K. Sarda et al., 'Adarma auto-detection and auto-remediation of microservice anomalies by leveraging large language models', in Proceedings of the 33rd Annual International Conference on Computer Science and Software Engineering, 2023, pp. 200–205.

[57] K. Sarda, 'Leveraging Large Language Models for Auto-remediation in Microservices Architecture', in 2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C), 2023, pp. 16–18.

[58] S. Nedelkoski, "snedelkoski/multi-source-observability-dataset," GitHub, Apr. 25, 2024. https://github.com/SashoNedelkoski/multi-source-observability-dataset/ (accessed Jun. 27, 2024).

[59] "CloudWise-OpenSource/GAIA-DataSet," GitHub, Jun. 27, 2024. https://github.com/CloudWise-OpenSource/GAIA-DataSet (accessed Jun. 27, 2024).

[60] "Foundation Models - IBM watsonx.ai," www.ibm.com. https://www.ibm.com/products/watsonx-ai/foundation-models

[61] N. Zhao et al., 'Understanding and handling alert storm for online service systems', in Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice, 2020, pp. 162–171.

[62] J. Kuang et al., 'Knowledge-aware Alert Aggregation in Large-scale Cloud Systems: a Hybrid Approach', in Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice, 2024, pp. 369–380.

[63] C. Zhao et al., 'Robust multimodal failure detection for microservice systems', in Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 5639–5649.

[64] S. Zhang et al., 'Robust failure diagnosis of microservice system through multimodal data', IEEE Transactions on Services Computing, vol. 16, no. 6, pp. 3851–3864, 2023.

Authorized licensed use limited to: Tsinghua University. Downloaded on March 13,2025 at 07:59:58 UTC from IEEE Xplore. Restrictions apply.