LEMMA-RCA: A LARGE MULTI-MODAL MULTI-DOMAIN DATASET FOR ROOT CAUSE ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Root cause analysis (RCA) is crucial for enhancing the reliability and performance of complex systems. However, progress in this field has been hindered by the lack of large-scale, open-source datasets tailored for RCA. To bridge this gap, we introduce LEMMA-RCA, a large dataset designed for diverse RCA tasks across multiple domains and modalities. LEMMA-RCA features various real-world fault scenarios from Information Technology (IT) and Operational Technology (OT) systems, encompassing microservices, water distribution, and water treatment systems, with hundreds of system entities involved. We evaluate the performance of six baseline methods on LEMMA-RCA across various settings, including offline and online modes, as well as single and multi-modal configurations. Our study demonstrates the utility of LEMMA-RCA in facilitating fair evaluation and promoting the development of more robust RCA techniques. The dataset and code are publicly available at `https://www.lemmarca.info`.

1 INTRODUCTION

Root cause analysis (RCA) is essential for identifying the underlying causes of system failures, ensuring the reliability and robustness of real-world systems. Recent advancements in artificial intelligence and software development have led to increased complexity and interdependence in modern systems. This complexity heightens their vulnerability to faults arising from interactions among modular services, which can disrupt user experiences and incur significant financial losses. Traditional manual RCA, however, is labor-intensive, costly, and prone to errors due to the complexity of systems and the extensive data involved. Therefore, efficient and effective data-driven RCA methods are crucial for pinpointing failures and mitigating financial losses when system faults occur.

Root cause analysis has been extensively studied across various domains and settings (Capozzoli et al., 2015; Deng & Hooi, 2021; Brandón et al., 2020; Fourlas & Karras, 2021; Gao et al., 2015). Based on the application scenarios, RCA can be carried out in *offline/online* fashion with *single/multi-modal* system data. Existing studies on RCA in these settings involve numerous learning techniques such as Bayesian methods (Alaeddini & Dogan, 2011), decision trees (Chen et al., 2004), *etc*. Particularly, causal structure learning based technique (Burr, 2003; Pamfil et al., 2020; Ng et al., 2020; Tank et al., 2022; Yu et al., 2023; Wang et al., 2023a;b; Zheng et al., 2024) has proven effective in constructing causal or dependency graphs between different system entities and key performance indicators (KPIs), thereby enabling the tracing of underlying causes through these structures.

Data is the oxygen of data-driven methods. Despite significant progress in RCA techniques, the availability of large-scale public datasets remains limited, often due to confidentiality concerns (Harsh et al., 2023). This scarcity hinders fair comparisons between RCA methods. Additionally, publicly accessible datasets often contain manually injected faults rather than real faults, and each dataset typically covers only a single domain. These limitations can prevent existing RCA methods from effectively identifying various types of system faults in real-world scenarios, potentially leading to regulatory and ethical consequences in critical sectors.

To address these limitations, we introduce **LEMMA-RCA**, a collection of Large-scalE Multi-ModAl datasets with various real system faults to facilitate future research in Root Cause Analysis. LEMMA-RCA encompasses real-world applications such as IT operations and water treatment systems, with **hundreds of system entities** involved. LEMMA-RCA accommodates **multi-modal** data including textual system logs with millions of event records and time series metric data with more than $100,000$

timestamps. We annotate LEMMA-RCA with ground truth labels indicating the precise time stamps when **real system faults** occur and their corresponding root-cause system entities.

Table 1: **Existing datasets for root cause analysis.** The top row corresponds to our dataset. The symbols ✓ and ✗ indicate that the dataset has or does not have the corresponding feature, respectively.

Dataset	Public	Real Faults	Large-scale	Multi-domain	Modality	
					Single	Multiple
LEMMA-RCA	✓	✓	✓	✓	✓	✓
NeZha	✓	✗	✗	✗	✓	✓
PetShop	✓	✗	✗	✗	✓	✗
Sock-Shop	✗	✗	✗	✗	✓	✗
ITOps	✗	✓	✓	✗	✓	✗
Murphy	✗	✓	✗	✗	✓	✗

A comparison between LEMMA-RCA and existing datasets for RCA is presented in Table 1. We briefly discuss the status of existing datasets: 1) *NeZha* (Yu et al., 2023) has limited size and contains many missing parts in the monitoring data, and it is confined to one domain: microservice architectures. 2) *PetShop* (Saurabh Garg, Imaya Kumar Jagannathan, 2024) has a small size. Additionally, the system comprises only 41 components, limiting its complexity and reducing the practicality for real-world scenarios. 3) *Sock-Shop* (Ikram et al., 2022) is small-scale with only 13 microservices, and the injected faults (CPU hog and memory leak) are synthetic. Additionally, the data is not publicly available and consists solely of single-modality metrics, lacking diversity in data sources such as logs or traces. 4) *ITOps* (Li et al., 2022c) dataset is not public and contains structured logs that do not contribute to comprehending the underlying causal mechanism of system failures, making it difficult to conduct fine-grained RCA. 5) *Murphy* (Harsh et al., 2023) is collected from a simple system and also not public. In comparison to prior work, LEMMA-RCA demonstrates a comprehensive maturity on the accessibility, authenticity, and diversity.

LEMMA-RCA enables fair comparison across RCA methods. We evaluate six baselines and assess data modality quality in offline settings, then adapt these methods for online evaluation using a standardized LEMMA-RCA formulation. These efforts enable, for the first time, rigorous comparison of online RCA methods on a common ground. Experimental results demonstrate LEMMA-RCA's effectiveness in evaluating related methods and its broad utility for advancing research in root cause analysis.

2 Preliminaries and Related Work

Key Performance Indicator (KPI) is a monitoring time series that indicates the system status. For instance, latency and service response time are two common KPIs used in microservice systems. A large value of latency or response time usually indicates a low-quality system performance or even a system failure.

Entity Metrics are multivariate time series collected by monitoring numerous system entities or components. For example, in a microservice system, a system entity can be a physical machine, container, pod, *etc*. Some common entity metrics in a microservice system include CPU utilization, Memory utilization, disk IO utilization, *etc*. An abnormal system entity is usually a potential root cause of a system failure.

Data-driven Root Cause Analysis Problem. Given the monitoring data (including metrics and logs) of system entities and system KPIs, the root cause analysis problem is to identify the top K system entities that are most relevant to KPIs when a system fault occurs. RCA techniques can be implemented in various settings, where offline/online and single-modal/multi-modal are mostly commonly concerned. Offline RCA is conducted retrospectively with historical data to determine past failures, whereas online RCA operates in real-time using current data streams to promptly address issues. On the other hand, Single-modal RCA relies solely on one type of data for a focused analysis, while multi-modal RCA uses multiple data sources for a comprehensive assessment. We illustrate the procedure of RCA in single-modal offline and multi-modal online settings in Figure 1.

Single-modal Offline Root Cause Analysis (RCA) retrospectively identifies the primary cause of system failures using a single data type after an event has occurred (Wang et al., 2023b; Tang et al., 2019; Meng et al., 2020b; Li et al., 2021; Soldani & Brogi, 2022). For example, Meng *et al.* (Meng et al., 2020b) analyze monitoring metric data to discern sequential relationships and integrate causal and temporal information for root cause localization in microservice systems. Similarly, Wang *et al.* (Wang et al., 2023b) construct an interdependent causal network from time series data, using a random walk strategy to pinpoint the most probable root causes. Li *et al.* (Li et al., 2021) evaluate microservice traces, determining that a service with a higher ratio of abnormal to normal traces is likely the root cause. Recently, large language model (LLM) based methods become a new research direction to learn causal relation for root cause identification due to the success of LLMs in performing complex tasks (Chen et al., 2024; Shan et al., 2024; Goel et al., 2024; Zhou et al., 2024; Roy et al., 2024; Wang et al., 2024). Although these studies demonstrate notable efficacy, they rely exclusively on single-modal data, which may lead to suboptimal and biased outcomes in root cause localization.

Multi-modal Offline RCA. Recent studies have explored utilizing multi-modal data for offline RCA, which can be divided into two approaches (Yu et al., 2023; Hou et al., 2021; Zheng et al., 2024; Lan et al., 2023). The first approach, exemplified by Nezha (Yu et al., 2023) and PDiagnose (Hou et al., 2021), involves extracting information from each modality separately and then integrating it for analysis. Conversely, the second approach focuses on the interactions between different modalities. For instance, MULAN (Zheng et al., 2024) develops a comprehensive causal graph by learning correlations between modalities, while MM-DAG (Lan et al., 2023) aims to jointly learn multiple Direct Acyclic Graphs, improving both consistency and depth of analysis.

Online RCA. Despite significant advances, most RCA methods are designed for offline use, requiring extensive data collection and full retraining for new faults, which delays response times. To address this, Wang *et al.* (Wang et al., 2023a) introduced an online RCA method that decouples state-invariant and state-dependent information and incrementally updates the causal graph. Li *et al.* (Li et al., 2022a) developed a causal Bayesian network that leverages system architecture knowledge to mitigate potential biases toward new data. However, these methods are limited to single-modal data, and there is a critical need for online RCA methods that can effectively handle multi-modal data.

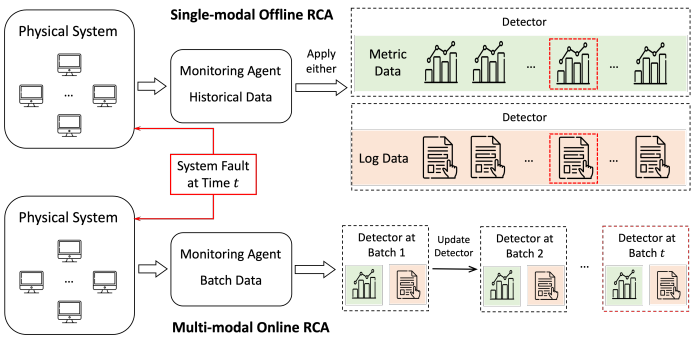


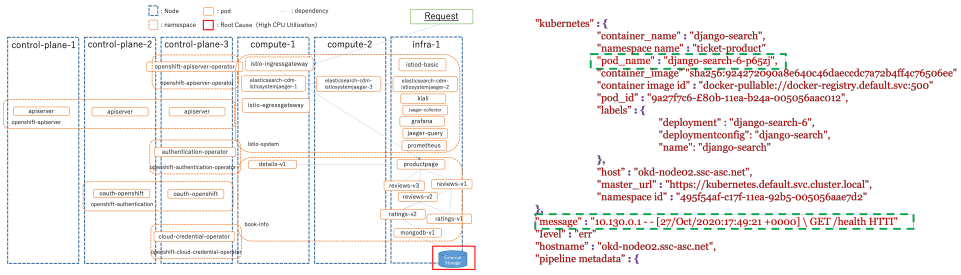
Figure 1: Illustration of RCA workflow in the single-modal offline setting (top) and the multi-modal online setting (bottom). The other two settings can be viewed as an ensemble of corresponding components (data collection, detector, modality) and follow the same systematic procedure.

3 LEMMA-RCA DATA

This section outlines the data resources, details the preprocessing steps, and presents visualizations to illustrate the characteristics of the data released. The data licence can be found in appendix F.

3.1 DATA COLLECTION

We collect real-world data from two domains: IT operations and OT operations. The IT domain includes sub-datasets from Product Review and Cloud Computing microservice systems, while the OT domain includes Secure Water Treatment (SWaT) and Water Distribution (WADI) sub-datasets from water treatment and distribution systems. Data specifics are provided in Table 2.



(a) The architecture of Product Review Platform (b) Log data captured by the ElasticSearch

Figure 2: Visualization of the microservice system platform, which contains 6 nodes and multiple pods that may vary across different stages; and the ElasticSearch log data. **We provide the figure with high resolution in the Appendix.**

In the IT domain, we developed two microservice platforms: the **Product Review** Platform and the **Cloud Computing** Platform. The Product Review Platform is composed of six OpenShift nodes (such as ocp4-control-plane-1 through ocp4-control-plane-3, ocp4-compute-1 and ocp4-compute-2, and ocp4-infra-1) and 216 system pods (including ProductPage, MongoDB, review, rating, payment, Catalogue, shipping, *etc*.). In this setup, four distinct system faults are collected, including out-of-memory, high-CPU-usage, external-storage-full, and DDoS attack, on four different dates. Each system fault ran the microservice system for at least 49 hours with different pods involved. The pods running in different stages may vary, and the pods associated with different types of system faults also differ. The structure of this microservice system with some key pods during one fault is depicted in Figure 2 (a). Both log and metric data were generated and stored systematically to ensure comprehensive monitoring. Specifically, eleven types of node-level metrics (*e.g.*, net disk IO usage, net disk space usage, *etc*.) and six types of pod-level metrics (*e.g.*, CPU usage, memory usage, *etc*.) were recorded by Prometheus (Turnbull, 2018), and the time granularity of these system metrics is 1 second. Log data, on the other hand, were collected by ElasticSearch (Zamfir et al., 2019) and stored in JSON files with detailed timestamps and retrieval periods. The contents of system logs include timestamp, pod name, log message, *etc*., as shown in Figure 2 (b). The JMeter (Nevedrov, 2006) was employed to collect the system status information, such as elapsed time, latency, connect time, thread name, throughput, *etc*. The latency is considered as system KPI as the system failure would result in the latency significantly increasing.

For the Cloud Computing Platform, we monitored six different types of faults (such as cryptojacking, mistakes made by GitOps, configuration change failure, *etc*.), and collected system metrics and logs from various sources. In contrast to the Product Review platform, system metrics were directly extracted from CloudWatch[1] Metrics on EC2 instances, and the time granularity of these system metrics is 1 second. Log events were acquired from CloudWatch Logs, consisting of three data types (*i.e.*, log messages, api debug log, and mysql log). Log message describes general log message about all system entities; api debug log contains debug information of the AP layer when the API was executed; mysql logs contain log information from database layer, including connection logs to mysql, which user connected from which host, and what queries were executed. Latency, error rate, and utilization rate were tracked using JMeter tool, serving as Key performance indicators (KPIs). This comprehensive logging and data storage setup facilitates detailed monitoring and analysis of the system's performance and behavior.

In the OT domain, we constructed two sub-datasets, SWaT and WADI, using monitoring data collected by the iTrust lab at the Singapore University of Technology and Design (iTrust, 2022). These two sub-datasets consist of time-series/metrics data, capturing the monitoring status of each sensor/actuator as well as the overall system at each second. Specifically, SWaT (Mathur & Tippenhauer, 2016) was collected over an 11-day period from a water treatment testbed equipped with 51 sensors. The system operated normally during the first 7 days, followed by attacks over the last 4 days, resulting in 16 system faults. Similarly, WADI (Ahmed et al., 2017) was gathered from a water distribution testbed over 16 days, featuring 123 sensors and actuators. The system maintained normal operations for the first 14 days before experiencing attacks in the final 2 days, with 15 system faults recorded.

[1]https://aws.amazon.com/cloudwatch/

Table 2: Data statistics of IT and OT operation sub-datasets.

Microservice System (IT)	Product Review	Cloud Computing
Original Dataset Size	765 GB	540 GB
Number of (#) fault types	4	6
Average # entities per fault	216.0	167.71
Average # metrics per fault	11 (node-level) + 6 (pod-level)	6 (node-level) + 7 (pod-level)
Average # timestamps per fault	131,329.25	109,350.57
Average max log events per fault across pods	153,081,219.0	63,768,587.25
Water Treatment/Distribution (OT)	**SWaT**	**WADI**
Original Dataset Size	4.47G	5.67G
Number of (#)fault types	16	9
Average # entities per fault	51.0	123.0
Average # metrics per fault	7 (node-level) + 7 (pod-level)	7 (node-level) + 7 (pod-level)
Average # timestamps per fault	56239.88	85248.47

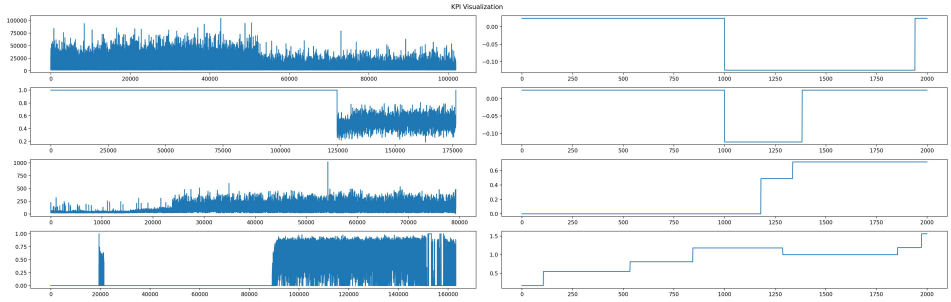


Figure 3: Visualization of KPI for system failure cases. **Left**: the first two sub-figures are from the Product Review sub-dataset; the third and fourth sub-figures are from the Cloud Computing sub-dataset; **Right**: the first two sub-figures are from the SWaT sub-dataset; the last two sub-figures are from the WADI sub-dataset.

We visualized the key performance indicator (KPI) for eight failure cases in Figure 3, where sudden spikes or drops in latency indicate system failures. The first two sub-figures on the left show the KPIs for two faults in the Product Review sub-dataset, while the third and fourth sub-figures depict faults in the Cloud Computing sub-dataset. The first two sub-figures on the right display faults in the SWaT dataset, and the last two show faults in the WADI dataset. The x-axis represents the timestamp, and the y-axis shows the system latency.

3.2 Data Preprocessing

After collecting system metrics and logs, we assess whether each pod exhibits stationarity, as non-stationary data are unpredictable and cannot be effectively modeled. Consequently, we exclude non-stationary pods, retaining only stationary ones for subsequent data preprocessing steps.

Log Feature Extraction for Product Review and Cloud Computing. The logs of some system entities we collected are limited and insufficient for meaningful root cause analysis. Thus, we exclude them from further analysis. Additionally, the log data is unstructured and frequently uses a special token, complicating its direct application for analysis. How to extract useful information from unstructured log data remains a great challenge. Following (Zheng et al., 2024), we preprocess the log data into time-series format. We first utilize a log parsing tool, such as Drain, to transform unstructured logs into structured log messages represented as templates. We then segment the data using fixed 10-minute windows with 30-second intervals, calculating the occurrence frequency of each log template. This frequency forms our first feature type, denoted as $X_1^L \in \mathbb{R}^T$, where T is the number of timestamps. We prioritize this feature because frequent log templates often indicate critical insights, particularly useful in identifying anomalies such as Distributed Denial of Service (DDoS) attacks, where a surge in template frequency can indicate unusual activity.

Moreover, we introduce a second feature type based on 'golden signals' derived from domain knowledge, emphasizing the frequency of abnormal logs associated with system failures like DDoS attacks, storage failures, and resource over-utilization. Identifying specific keywords like 'error,' 'exception,' and 'critical' within log templates helps pinpoint anomalies. This feature, denoted as

$X_2^L \in \mathbb{R}^T$, assesses the presence of abnormal log templates to provide essential labeling information for anomaly detection.

Lastly, we implement a TF-IDF based method, segmenting logs using the same time windows and applying Principal Component Analysis (PCA) to reduce feature dimensionality, selecting the most significant component as $X_3^L \in \mathbb{R}^T$. We concatenate these three feature types to form the final feature matrix $X^L = [X_1^L; X_2^L; X_3^L] \in \mathbb{R}^{3\times T}$, enhancing our capacity for a comprehensive analysis of system logs and improving anomaly detection capabilities.

KPI Construction for SWaT and WADI. The SWaT and WADI sub-datasets include the label column that reflects the system status; however, the values within this column are discrete. To facilitate the root cause analysis, it is beneficial to transform these values into a continuous format. Specifically, we propose to convert the label into a continuous time series. To achieve this, we employ anomaly detection algorithms, such as Support Vector Data Description and Isolation Forest, to model the data. Subsequently, the anomaly score, as determined by the model, will be utilized as the system KPI. More data preprocessing details on SWaT and WADI can be found in Appendix C.

3.3 System Fault Scenarios

There are 10 different types of real system faults in Product Review and Cloud Computing sub-datasets. Due to the space limitation, we select two representative cases (one from each) and provide the details below. Other fault scenarios are presented in Appendix D. We also visualize the system fault of these two cases in Figure 4.

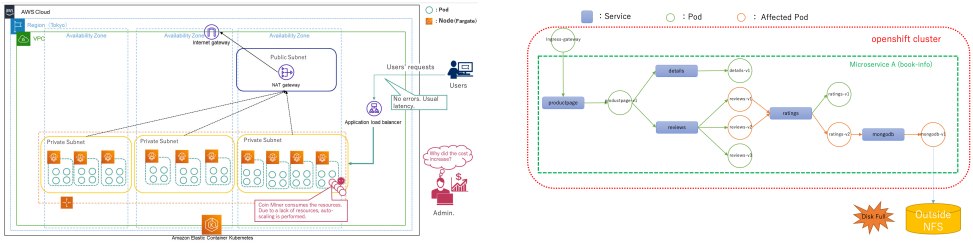


Figure 4: Visualization of two system fault scenarios. **Left:** Cryptojacking. **Right:** External storage failure.

- **Cryptojacking**. In this scenario, cloud usage fees increase due to cryptojacking, where a Coin Miner is covertly downloaded and installed on a microservice (details-v1 pod) in an EKS cluster. This miner gradually consumes IT resources, escalating the cloud computing costs. Identifying the root cause is challenging because the cost (SLI) encompasses the entire system, and no individual service errors are detected. Periodic external requests are sent to microservices, and after a day, the miner's activity triggers auto-scaling in details-v1, increasing resource usage. Fargate's impact on EKS costs is significant due to its resource dependency. KPI (SLI) is calculated from resource usage, with all pod and node metrics collected from CloudWatch. However, there are no node logs for Fargate, complicating diagnosis.
- **External Storage Failure**. In this system failure, we fill up the external storage disk connected to the Database (DB) pod (*i.e.*, mongodb-v1) within Microservice A's OpenShift[2] cluster. When the storage becomes full, the DB pod cannot add new data, resulting in system errors. These errors propagate to pods that depend on the DB pod, causing some services (ratings) within Microservice A to encounter errors. We monitor changes in response and error information for Microservice A using Jaeger logs. Metrics for all containers and nodes, including CPU and memory usage, are obtained from Prometheus within OpenShift. Logs for all containers and nodes are retrieved from Elasticsearch within OpenShift. Additionally, we collect message logs from the external storage. We illustrate the metrics and log data of the root cause pod in Figure 5.

[2]https://www.redhat.com/en/technologies/cloud-computing/openshift

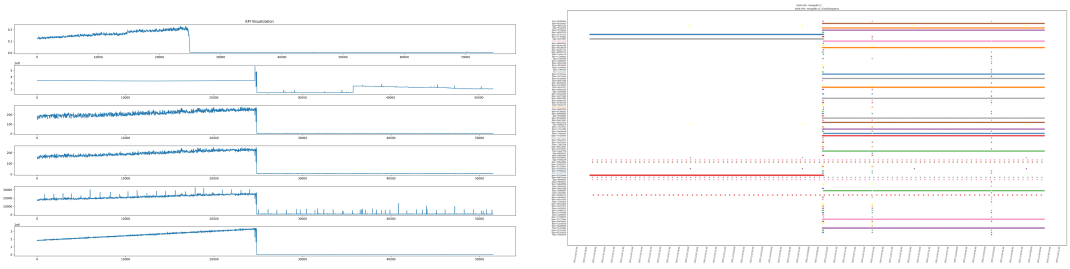


Figure 5: Visualization of root cause for one system failure case (*i.e.*, **External Storage Failure**) on the Product Review Platform. **Left:** six system metrics of root cause. **Right:** the system log of the root cause pod (*i.e.*, Mongodb-v1) with the x-axis representing the timestamp, the y-axis indicating the log event ID, and the colored dots denoting event occurrences. Sudden drops in the metrics data, as well as new log event patterns observed at the midpoint, indicate a system failure.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Evaluation Metrics. To assess baseline RCA method on LEMMA-RCA, we choose three widely-used metrics (Wang et al., 2023b; Meng et al., 2020a; Zheng et al., 2024) and introduce them below.

(1). **Precision@K (PR@K)**: It measures the probability that the top K predicted root causes are real, defined as:

$$\text{PR@K} = \frac{1}{|\mathbb{A}|} \sum_{a \in \mathbb{A}} \frac{\sum_{i<k} R_a(i) \in V_a}{\min(K, |v_a|)} \tag{1}$$

where $\mathbb{A}$ is the set of system faults, a is one fault in $\mathbb{A}$, V_a is the real root causes of a, R_a is the predicted root causes of a, and i is the i-th predicted cause of R_a.

(2). **Mean Average Precision@K (MAP@K)**: It assesses the top K predicted causes from the overall perspective, defined as:

$$\text{MAP@K} = \frac{1}{K|\mathbb{A}|} \sum_{a \in \mathbb{A}} \sum_{i \le j \le K} \text{PR@j} \tag{2}$$

where a higher value indicates better performance.

(3). **Mean Reciprocal Rank (MRR)**: It evaluates the ranking capability of models, defined as:

$$\text{MRR@K} = \frac{1}{|\mathbb{A}|} \sum_{a \in \mathbb{A}} \frac{1}{\text{rank}_{R_a}} \tag{3}$$

where rank_{R_a} is the rank number of the first correctly predicted root cause for system fault a.

Baselines. We evaluate the performance of the following RCA baselines on the benchmark sub-datasets, selecting only those with publicly available code to ensure fair and reproducible comparisons: (1). **PC-based** (Ma et al., 2020): This approach first employs a Peter-Clark (PC) algorithm Burr (2003) to construct the anomaly behavior graph, and then applies a random walk algorithm to rank the root causes based on the estimated graph structure. (2). **CIRCA** (Li et al., 2022b): This model utilizes structural graph construction, regression-based hypothesis testing, and descendant adjustment to identify root cause metrics. (3). **ϵ-Diagnosis** (Shan et al., 2019): This model diagnoses small-window, long-tail latency in large-scale microservice platforms using a two-sample test and ϵ-statistics. (4). **RCD** (Ikram et al., 2022): This technique hierarchically localizes the root cause of failures by focusing on relevant sections of the dependency graph. (5). **BARO** (Pham et al., 2024): It is an end-to-end approach integrating Bayesian change point detection and nonparametric hypothesis testing to accurately detect anomalies and identify root causes in microservice systems. (6). **Nezha** (Yu et al., 2023): A multi-modal method designed to identify root causes by detecting abnormal patterns.

When using only metric data (*e.g.*, CPU usage, memory usage) for root cause analysis, we first identify the potential root cause associated with each metric. We then compute the final score for each candidate by averaging its ranking scores across all metrics. This procedure is similarly applied in log-only and multi-modal scenarios to rank and identify the root causes. For the hyperparameters, we use the default parameter values for all baselines to ensure a fair comparison.

Table 3: Results for RCA baselines with multiple modalities on the Product Review dataset.

Modality	Model	PR@1	PR@5	PR@10	MRR	MAP@3	MAP@5	MAP@10
Metric Only	PC	0	0	0.250	0.053	0	0	0.050
	RCD	0	0.250	0.750	0.185	0.167	0.200	0.350
	ϵ-Diagnosis	0	0	0.250	0.038	0	0	0.050
	CIRCA	0	0.750	0.750	0.283	0.250	0.450	0.600
	BARO	0.250	0.250	0.250	0.286	0.250	0.250	0.250
Log Only	PC	0	0	0.250	0.069	0	0	0.125
	RCD	0	0	0.250	0.056	0	0	0.025
	ϵ-Diagnosis	0	0	0.250	0.038	0	0	0.050
	CIRCA	0	0	0.500	0.080	0	0	0.125
	BARO	0	0.250	0.250	0.139	0.167	0.200	0.225
Multi-Modality	PC	0	0	0.250	0.064	0	0	0.125
	RCD	0	0.500	0.750	0.231	0.167	0.300	0.525
	ϵ-Diagnosis	0	0	0.250	0.041	0	0	0.075
	CIRCA	0	**0.750**	**1.000**	0.299	0.250	0.450	0.650
	BARO	**0.750**	**0.750**	**1.000**	**0.775**	**0.750**	**0.750**	**0.775**
	Nezha	0	0.500	0.750	0.193	0.083	0.250	0.475

4.2 Root Cause Analysis Results

Product Review and Cloud Computing. We evaluate six RCA methods including both single-modal and multi-modal methods on Product Review and Cloud Computing sub-datasets. The experimental results are presented in Table 3 with respect to Precision at K (PR@K), Mean Reciprocal Rank (MRR), and Mean Average Precision at K (MAP@K). Due to space limitations, the experimental results on Cloud Computing (Table 7) are reported in the Appendix E. Our observations reveal the following insights: (1) The PC algorithm and ϵ-Diagnosis perform worst on both the Product Review and Cloud Computing sub-datasets. We conjecture that PC, RCD, and ϵ-Diagnosis struggle to capture long-term dependencies in large-scale datasets, making it difficult to detect abnormal temporal patterns. (2) CIRCA outperforms RCD and ϵ-Diagnosis, consistent with findings from the Petshop study (Saurabh Garg, Imaya Kumar Jagannathan, 2024), where CIRCA's regression-based hypothesis testing and adjustment mechanisms led to higher diagnostic accuracy. (3) Multi-modal input—combining both metric and log data—significantly enhances the performance of RCA methods compared to using either modality alone. For example, BARO achieves only 25% PR@1 with metric data and 0% with log data on the Product Review sub-dataset, but reaches 75% PR@1 when both are used, correctly identifying the root cause in 75% of fault scenarios. This highlights the complementary nature of log and metric data and the importance of integrating both to improve diagnostic accuracy and overall performance, particularly in terms of MRR.

Table 4: Results for RCA baselines on the SWaT sub-dataset.

Dataset	Model	PR@1	PR@5	PR@10	MRR	MAP@3	MAP@5	MAP@10
SWaT	PC	0.125	**0.344**	0.583	0.262	0.129	**0.204**	0.350
	RCD	0.125	0.125	0.625	0.228	0.125	0.125	0.344
	ε-Diagnosis	0.125	0.125	0.563	0.217	0.125	0.125	0.294
	CIRCA	**0.188**	0.250	**0.688**	**0.287**	**0.188**	0.200	**0.394**
	BARO	0	0.208	0.208	0.124	0.083	0.133	0.171

Water Treatment/Distribution. We evaluate five single-modal RCA methods on the SWaT and WADI sub-datasets using the same set of evaluation metrics. Table 4 presents the results for SWaT, while Table 5 shows the results for WADI. We observe that CIRCA consistently achieves the best overall performance on both datasets, although the PC algorithm occasionally outperforms it on

Table 5: Results for RCA baselines on the WADI sub-dataset.

Dataset	Model	PR@1	PR@5	PR@10	MRR	MAP@3	MAP@5	MAP@10
WADI	PC	0.071	0.350	0.500	0.277	0.163	0.239	0.346
	RCD	0	0 .071	0.119	0.054	0.048	0.057	0.076
	ε-Diagnosis	0	0	0.022	0.020	0	0	0.009
	CIRCA	**0.143**	**0.550**	**0.714**	**0.350**	**0.301**	**0.400**	**0.529**
	BARO	0	0.143	0.143	0.085	0.071	0.100	0.121

SWaT by a small margin. This trend aligns with the observations from the Product Review and Cloud Computing sub-datasets. Notably, the results also indicate considerable room for improvement, likely due to the characteristics of the SWaT and WADI datasets—faults are short-lived, and the intervals between them are brief. These fleeting events are easily overlooked by most RCA methods, posing a significant challenge for accurate root cause identification.

4.3 Online Root Cause Analysis Results

We evaluate three RCA methods on the Product Review sub-dataset to demonstrate the utility of the LEMMA-RCA in an online setting in Table 6. Due to space limitations, the experimental results on Cloud Computing (Table 8) are reported in the Appendix E. Because the runtime of CIRCA, PC, and Nezha exceed 24 hours per case, they are excluded from the online evaluation. Notably, LEMMA-RCA is a large-scale real-world dataset, consisting of more than 100,000 timestamps across several days with various system fault scenarios, which can be naturally transformed to the online setting, compared with small datasets with limited timestamps for online RCA. Although RCD, ϵ-Diagnosis, and BARO are originally designed for offline analysis, we adapt them to the online setting by formatting the data into a sequence of streaming snapshots. Each baseline method is then evaluated on consecutive snapshots until similar results appear three times in succession or the data stream reaches its final snapshot. The detailed implementation can be found in Appendix B. Empirically, we observe that all three methods suffer a significant drop in performance under the online setting compared to their offline results. We conjecture that this degradation arises from their inability to consistently capture temporal dependencies across snapshots. This observation highlights the need for developing dedicated online root cause analysis methods.

Table 6: Results for root cause analysis baselines on Product Review sub-dataset in the online setting.

Modality	Model	PR@1	PR@5	PR@10	MRR	MAP@3	MAP@5	MAP@10
Metric Only	RCD	0	**0.250**	**0.250**	0.054	0	0.050	0.150
	ϵ-Diagnosis	0	0	0	0.019	0	0	0
	BARO	**0.250**	**0.250**	**0.250**	0.269	**0.250**	**0.250**	**0.250**
Log Only	RCD	0	0	0	0.012	0	0	0
	ϵ-Diagnosis	0	0	0	0.025	0	0.100	0.175
	BARO	0	0	0	0.023	0	0.100	0.175
Multi-Modality	RCD	0	**0.250**	**0.250**	0.054	0	0.050	0.150
	ϵ-Diagnosis	0	0	0	0.027	0	0	0
	BARO	**0.250**	**0.250**	**0.250**	**0.270**	**0.250**	**0.250**	**0.250**

5 Conclusion

In this work, we introduce LEMMA-RCA, the first large-scale, open-source dataset featuring real system faults across multiple application domains and data modalities. We conduct a comprehensive empirical study using six baseline RCA methods, evaluating their performance on both single-modal and multi-modal data under offline and online settings. The experimental results highlight the utility of LEMMA-RCA as a benchmarking resource. By releasing this dataset publicly, we aim to advance research in root cause analysis for complex systems and support the development of more robust and reliable methodologies, particularly for mission-critical applications.

REFERENCES

Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. Wadi: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, pp. 25–28, 2017.

Adel Alaeddini and Ibrahim Dogan. Using bayesian networks for root cause analysis in statistical process control. *Expert Systems with Applications*, 38(9):11230–11243, 2011.

Álvaro Brandón, Marc Solé, Alberto Huélamo, David Solans, María S Pérez, and Victor Muntés-Mulero. Graph-based root cause analysis for service-oriented and microservice architectures. *Journal of Systems and Software*, 159:110432, 2020.

Tom Burr. Causation, prediction, and search. *Technometrics*, 45(3):272–273, 2003.

Alfonso Capozzoli, Fiorella Lauro, and Imran Khan. Fault detection analysis using data mining techniques for a cluster of smart office buildings. *Expert Systems with Applications*, 42(9):4324–4338, 2015.

Mike Chen, Alice X Zheng, Jim Lloyd, Michael I Jordan, and Eric Brewer. Failure diagnosis using decision trees. In *International Conference on Autonomic Computing, 2004. Proceedings.*, pp. 36–43. IEEE, 2004.

Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, et al. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pp. 674–688, 2024.

Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4027–4035, 2021.

George K Fourlas and George C Karras. A survey on fault diagnosis methods for uavs. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 394–403. IEEE, 2021.

Zhiwei Gao, Carlo Cecati, and Steven X. Ding. A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, 62(6):3757–3767, 2015. doi: 10.1109/TIE.2015.2417501.

Drishti Goel, Fiza Husain, Aditya Singh, Supriyo Ghosh, Anjaly Parayil, Chetan Bansal, Xuchao Zhang, and Saravan Rajmohan. X-lifecycle learning for cloud incident management using llms. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, pp. 417–428, 2024.

Vipul Harsh, Wenxuan Zhou, Sachin Ashok, Radhika Niranjan Mysore, Brighten Godfrey, and Sujata Banerjee. Murphy: Performance diagnosis of distributed cloud applications. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pp. 438–451, 2023.

Chuanjia Hou, Tong Jia, Yifan Wu, Ying Li, and Jing Han. Diagnosing performance issues in microservices with heterogeneous data source. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), New York City, NY, USA, September 30 - Oct. 3, 2021*, pp. 493–500. IEEE, 2021.

Azam Ikram, Sarthak Chakraborty, Subrata Mitra, Shiv Saini, Saurabh Bagchi, and Murat Kocaoglu. Root cause analysis of failures in microservices through causal discovery. *Advances in Neural Information Processing Systems*, 35:31158–31170, 2022.

iTrust. The website of itrust lab. [EB/OL], 2022. `https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/`.

Tian Lan, Ziyue Li, Zhishuai Li, Lei Bai, Man Li, Fugee Tsung, Wolfgang Ketter, Rui Zhao, and Chen Zhang. Mm-dag: Multi-task dag learning for multi-modal data-with application for traffic congestion analysis. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1188–1199, 2023.

Mingjie Li, Zeyan Li, Kanglin Yin, Xiaohui Nie, Wenchi Zhang, Kaixin Sui, and Dan Pei. Causal inference-based root cause analysis for online service systems with intervention recognition. In Aidong Zhang and Huzefa Rangwala (eds.), *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pp. 3230–3240. ACM, 2022a.

Mingjie Li, Zeyan Li, Kanglin Yin, Xiaohui Nie, Wenchi Zhang, Kaixin Sui, and Dan Pei. Causal inference-based root cause analysis for online service systems with intervention recognition. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3230–3240, 2022b.

Zeyan Li, Junjie Chen, Rui Jiao, Nengwen Zhao, Zhijun Wang, Shuwei Zhang, Yanjun Wu, Long Jiang, Leiqin Yan, Zikai Wang, et al. Practical root cause localization for microservice systems via trace analysis. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pp. 1–10. IEEE, 2021.

Zeyan Li, Nengwen Zhao, Shenglin Zhang, Yongqian Sun, Pengfei Chen, Xidao Wen, Minghua Ma, and Dan Pei. Constructing large-scale real-world benchmark datasets for aiops. *arXiv preprint arXiv:2208.03938*, 2022c.

Meng Ma, Jingmin Xu, Yuan Wang, Pengfei Chen, Zonghua Zhang, and Ping Wang. Automap: Diagnose your microservice-based web applications automatically. In *Proceedings of The Web Conference 2020*, pp. 246–258, 2020.

Aditya P Mathur and Nils Ole Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pp. 31–36. IEEE, 2016.

Yuan Meng, Shenglin Zhang, Yongqian Sun, Ruru Zhang, Zhilong Hu, Yiyin Zhang, Chenyang Jia, Zhaogang Wang, and Dan Pei. Localizing failure root causes in a microservice through causality inference. In *28th IEEE/ACM International Symposium on Quality of Service, IWQoS 2020, Hangzhou, China, June 15-17, 2020*, pp. 1–10. IEEE, 2020a.

Yuan Meng, Shenglin Zhang, Yongqian Sun, Ruru Zhang, Zhilong Hu, Yiyin Zhang, Chenyang Jia, Zhaogang Wang, and Dan Pei. Localizing failure root causes in a microservice through causality inference. In *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, pp. 1–10. IEEE, 2020b.

Dmitri Nevedrov. Using jmeter to performance test web services. *Published on dev2dev*, pp. 1–11, 2006.

Ignavier Ng, AmirEmad Ghassami, and Kun Zhang. On the role of sparsity and DAG constraints for learning linear dags. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. DYNOTEARS: structure learning from time-series data. In Silvia Chiappa and Roberto Calandra (eds.), *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1595–1605. PMLR, 2020.

Luan Pham, Huong Ha, and Hongyu Zhang. Baro: Robust root cause analysis for microservices via multivariate bayesian online change point detection. *Proceedings of the ACM on Software Engineering*, 1(FSE):2214–2237, 2024.

Devjeet Roy, Xuchao Zhang, Rashi Bhave, Chetan Bansal, Pedro Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. Exploring llm-based agents for root cause analysis. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, pp. 208–219, 2024.

Saurabh Garg, Imaya Kumar Jagannathan. Root cause analyses on petshop application, 2024. https://github.com/amazon-science/petshop-root-cause-analysis/tree/main?tab=readme-ov-file.

Huasong Shan, Yuan Chen, Haifeng Liu, Yunpeng Zhang, Xiao Xiao, Xiaofeng He, Min Li, and Wei Ding. ?-diagnosis: Unsupervised and real-time diagnosis of small-window long-tail latency in large-scale microservice platforms. In *The World Wide Web Conference*, pp. 3215–3222, 2019.

Shiwen Shan, Yintong Huo, Yuxin Su, Yichen Li, Dan Li, and Zibin Zheng. Face it yourselves: An llm-based two-stage strategy to localize configuration errors via logs. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 13–25, 2024.

Jacopo Soldani and Antonio Brogi. Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey. *ACM Computing Surveys (CSUR)*, 55(3):1–39, 2022.

LuAn Tang, Hengtong Zhang, Zhengzhang Chen, Bo Zong, LI Zhichun, Guofei Jiang, and Kenji Yoshihira. Graph-based attack chain discovery in enterprise security systems, May 14 2019. US Patent 10,289,841.

Alex Tank, Ian Covert, Nicholas J. Foti, Ali Shojaie, and Emily B. Fox. Neural granger causality. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(8):4267–4279, 2022.

James Turnbull. *Monitoring with Prometheus*. Turnbull Press, 2018.

Dongjie Wang, Zhengzhang Chen, Yanjie Fu, Yanchi Liu, and Haifeng Chen. Incremental causal graph learning for online root cause analysis. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2269–2278, 2023a.

Dongjie Wang, Zhengzhang Chen, Jingchao Ni, Liang Tong, Zheng Wang, Yanjie Fu, and Haifeng Chen. Interdependent causal networks for root cause localization. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pp. 5051–5060. ACM, 2023b.

Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Lunting Fan, Lingfei Wu, and Qingsong Wen. Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2024.

Guangba Yu, Pengfei Chen, Yufeng Li, Hongyang Chen, Xiaoyun Li, and Zibin Zheng. Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data. 2023.

Vlad-Andrei Zamfir, Mihai Carabas, Costin Carabas, and Nicolae Tapus. Systems monitoring and big data analysis using the elasticsearch system. In *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pp. 188–193. IEEE, 2019.

Lecheng Zheng, Zhengzhang Chen, Jingrui He, and Haifeng Chen. Multi-modal causal structure learning and root cause analysis. *arXiv preprint arXiv:2402.02357*, 2024.

Bin Zhou, Xinyu Li, Tianyuan Liu, Kaizhou Xu, Wei Liu, and Jinsong Bao. Causalkgpt: industrial structure causal knowledge-enhanced large language model for cause analysis of quality problems in aerospace product manufacturing. *Advanced Engineering Informatics*, 59:102333, 2024.

A Broader Impact and Limitation

Broader impact: To facilitate accurate, efficient, and multi-modal root cause analysis research across diverse domains, we introduce LEMMA-RCA as a new benchmark dataset. Our dataset also offers significant potential for advancing research in areas like **multi-modal anomaly detection**, **change point detection**, and **system diagnosis**. Based on the thorough data analysis and extensive experimental results, we highlight the following areas for future research:

- **Expanding Domain Applications**: To enhance the LEMMA-RCA dataset's versatility and impact, we plan to incorporate data from additional domains such as cybersecurity and healthcare. This integration of diverse data sources will facilitate the development of more comprehensive root cause analysis technologies, significantly extending the dataset's applicability across various industries.
- **Online and/or Multi-Modal Root Cause Analysis**: Most RCA methods are offline and single-modal, leaving a gap for real-time and/or multi-modal approaches. Developing these methods can enable instant analysis of diverse data streams, essential for dynamic environments like industrial automation and real-time monitoring.
- **LLM-Based Root Cause Analysis**: The emergence of LLMs presents new opportunities for RCA by enabling systems to reason over complex, unstructured, and heterogeneous data sources. Future research can explore how LLMs can be adapted or fine-tuned for root cause inference, how they can incorporate domain knowledge, and how their interpretability and reliability can be improved in safety-critical or high-stakes environments.

Limitations: Despite its broad capabilities, the LEMMA-RCA dataset may have limited generalizability, as its fault scenarios may not fully capture the diversity of real-world conditions due to factors like system interruptions and unforeseen circumstances.

B Implementation Details of Online Setting

In this section, we describe how we format the data for the online setting and adapt offline RCA methods accordingly. For each case, the data is split chronologically into two parts: historical data and streaming data. The split point is determined by the "online start," a timestamp prior to the system fault (typically 1–2 hours before the failure), indicating that a fault is imminent. The streaming data is further divided into K snapshots, each covering a fixed time window of 400 seconds. To construct the first batch, we combine the historical data with the first snapshot, which serves as input for each baseline method to identify the root cause. We then update the batch iteratively by sliding the window: at each step, we remove the earliest 400 seconds from the historical data and replace it with the next 400 seconds from the streaming snapshots, ensuring the total batch length remains fixed. Offline methods adapted to this setting are evaluated sequentially on the snapshots until either (i) consistent results are obtained for three consecutive steps, or (ii) the data stream reaches its final snapshot.

C Monitoring Time Series Segmentation for SWaT and WADI

In the original SWaT and WADI datasets, the attack model demonstrates irregular attack patterns, occasionally targeting multiple sensors simultaneously, or executing attacks at closely spaced intervals. To follow the principles of RCA, we have established two specific preprocessing rules for these datasets: 1) Each recorded attack event must only involve a single sensor or actuator. 2) The duration of the dataset corresponding to each attack event must be standardized to two hours. Consequently, we selectively keep attack events that impact only one sensor or actuator. If the interval between successive attack events is insufficiently short, we assume the stability in the monitoring data immediately before and after each attack event. To ensure the necessary two-hour duration for each event, we concatenate normal-state data from both before and after the attack period. This adjustment positions the attack event centrally within a continuous two-hour segment, facilitating consistent and accurate analysis.

D ADDITIONAL SYSTEM FAULT SCENARIOS

This section describes the processes used to generate and monitor system fault scenarios, with emphasis on mimicking real-world fault patterns. Each scenario involved the induction of specific failure conditions, while allowing the microservice system to exhibit its natural behavior under stress. Metrics and logs were collected using established monitoring tools, such as Prometheus, Elasticsearch, CloudWatch, Jaeger, and JMeter.

- **Silent Pod Degradation Fault.**
  - **Description:** A pod in a load balancer contains a latent bug causing its CPU usage to rise, which gradually increases latency for a subset of users without triggering autoscaling or error alerts.
  - **Method:** We periodically sent requests to Microservice A over a 24-hour period. After this initial observation, we manually increased the CPU load on one specific `productpage-v1` pod to simulate the bug.
  - **Data Collection:** Metrics and logs were collected from CloudWatch, while KPIs such as latency were measured using JMeter. The goal was to trace latency increases back to the specific pod with elevated CPU utilization.
- **Noisy Neighbor Issue.**
  - **Description:** A neighboring pod in a shared node generates high CPU load, impacting the performance of the `productpage-v1` pod and causing elevated error rates.
  - **Method:** Requests were sent to Microservice A, while the pod ratings of Microservice B (`robot-shop`) were moved to the same node as `productpage-v1`, generating contention.
  - **Data Collection:** Metrics (CPU usage, memory usage) were gathered using Prometheus, while logs were obtained from CloudWatch Logs. Configuration changes, such as node assignments, were also recorded.
- **Node Resource Contention Stress Test.**
  - **Description:** A stress test on CPU resources was conducted by inducing high load on Microservice B, co-located with Microservice A on the same node.
  - **Method:** Periodic requests were sent to Microservice A using JMeter, while a high CPU load was generated on Microservice B using the `OpenSSL speed` command.
  - **Data Collection:** HTTP response logs from JMeter were analyzed for performance impacts. System metrics (CPU and memory usage) were retrieved from Prometheus, while container logs were collected from Elasticsearch.
- **DDoS Attack.**
  - **Description:** A Distributed Denial of Service (DDoS) attack was simulated to overload the system, causing Out-of-Memory (OOM) errors in targeted pods.
  - **Method:** Over a monitoring period of approximately 48 hours, we gradually increased the request rate to Microservice A, eventually overwhelming the `reviews-v2` and `reviews-v3` pods.
  - **Data Collection:** Metrics such as CPU and memory utilization were collected via Prometheus. Logs from Jaeger and Elasticsearch provided insights into the system's response to the attack.
- **Malware Attack.**
  - **Description:** A malware pod executed a password list attack to compromise other pods, propagating DDoS scripts to degrade overall system performance.
  - **Method:** The attack started from a designated pod (`scenario10-malware-deployment`) and targeted others via SSH password brute-forcing, ultimately generating high load on `productpage-v1`.
  - **Data Collection:** JMeter was used to monitor KPIs (latency, error rate), while Prometheus and CloudWatch Logs provided system metrics and logs for root-cause analysis.
- **Bug Infection.**

  - **Description:** A latent bug in the API caused asymmetric CPU load increases, degrading response times without fully utilizing the CPU capacity.
  - **Method:** Requests were sent periodically to the web service, and after a day, a script induced increased CPU utilization on one core.
  - **Data Collection:** KPIs were measured using JMeter, while system metrics and logs were collected via CloudWatch for detailed analysis.

- **Configuration Fault.**
  - **Description:** An incorrect resource limit in a Kubernetes manifest file led to a pod being terminated by the OOM killer, impacting other services.
  - **Method:** Requests were sent to Microservice A, while a Git push introduced a faulty configuration for the `details-v1` pod. The misconfigured pod eventually failed under load.
  - **Data Collection:** Error rates were tracked using JMeter, and metrics/logs were retrieved from Prometheus and CloudWatch for root-cause identification.

E More Experimental Results

We present the results for RCA on Cloud Computing sub-dataset in the offline setting in Table 7 and in the online setting in Table 8.

Table 7: Results for RCA with multiple modalities on the Cloud Computing sub-dataset.

Modality	Model	PR@1	PR@5	PR@10	MRR	MAP@3	MAP@5	MAP@10
Metric Only	PC	0	0	0	0.029	0	0	0
	RCD	0	**0.250**	0.250	0.126	**0.083**	**0.150**	0.200
	ϵ-Diagnosis	0	0	0.250	0.067	0	0	0.025
	CIRCA	0	0	**0.500**	0.105	0	0	0.150
	BARO	0	**0.250**	0.250	0.105	0	0.100	0.175
Log Only	PC	0	0	0	0.032	0	0	0
	RCD	0	0	0	0.059	0	0	0
	ϵ-Diagnosis	0	0	0	0.059	0	0	0
	CIRCA	0	**0.250**	0.250	0.110	0	0.100	0.175
	BARO	0	**0.250**	0.250	0.105	0	0.100	0.175
Multi-Modality	PC	0	0	0.250	0.064	0	0	0.125
	RCD	0	**0.250**	0.250	0.092	0	0.050	0.150
	ϵ-Diagnosis	0	0	0.250	0.067	0	0	0.025
	CIRCA	0	**0.250**	**0.500**	**0.147**	**0.083**	**0.150**	**0.225**
	BARO	0	**0.250**	0.250	0.126	**0.083**	**0.150**	0.200
	Nezha	0	**0.250**	0.250	0.105	0	0.100	0.175

Table 8: Results for root cause analysis baselines on Cloud Computing sub-dataset in the online setting.

Modality	Model	PR@1	PR@5	PR@10	MRR	MAP@3	MAP@5	MAP@10
Metric Only	BARO	0	**0.250**	**0.500**	0.172	**0.167**	**0.200**	**0.275**
	ϵ-Diagnosis	0	**0.250**	**0.500**	0.173	**0.167**	**0.200**	0.250
	RCD	0	**0.250**	0.250	0.162	**0.167**	**0.200**	0.225
Log Only	BARO	0	0	0.250	0.067	0	0	0.075
	ϵ-Diagnosis	0	0	0.250	0.072	0	0	0.100
	RCD	0	0	0	0.044	0	0	0
Multi-Modality	BARO	0	**0.250**	**0.500**	0.173	**0.167**	**0.200**	**0.275**
	ϵ-Diagnosis	0	**0.250**	**0.500**	**0.181**	**0.167**	**0.200**	0.250
	RCD	0	**0.250**	0.250	0.162	**0.167**	**0.200**	0.225

F LEMMA-RCA LICENSE

The LEMMA-RCA benchmark dataset is released under a CC BY-ND 4.0 International License: https://creativecommons.org/licenses/by-nd/4.0. The license of any specific baseline methods used in our codebase should be verified on their official repositories.

G REPRODUCIBILITY

All experiments are conducted on a server running Ubuntu 18 with an Intel(R) Xeon(R) Silver 4110 CPU @2.10GHz and one 11GB GTX2080 GPU. In addition, all methods were implemented using Python 3.8.12 and PyTorch 1.7.1.

H DETAILED DESCRIPTION OF BASELINES

We evaluate the performance of the following RCA models on the benchmark sub-datasets:

- **PC-based** (Ma et al., 2020): It is an unsupervised root cause analysis method designed for complex systems using multiple types of performance metrics. It constructs an anomaly behavior graph using a PC-based causal discovery algorithm to capture service-level dependencies. Based on this graph, it applies heuristic random walk strategies—including forward, backward, and self-directed walks—to trace and rank potential root causes.
- **CIRCA** (Li et al., 2022b): CIRCA is an unsupervised root cause analysis method that formulates the problem as a causal inference task called intervention recognition. Its core idea is to identify root cause indicators by evaluating changes in the probability distribution of monitoring variables conditioned on their parents in a Causal Bayesian Network (CBN). CIRCA applies this approach to online service systems by constructing a graph among monitoring metrics, leveraging system architecture knowledge and causal assumptions to guide the analysis.
- **ϵ-Diagnosis** (Shan et al., 2019): ϵ-Diagnosis is an unsupervised, low-cost diagnosis algorithm designed to address small-window long-tail latency (SWLT) in web services, which arises in short statistical windows and typically affects a small subset of containers in microservice clusters. It uses a two-sample test algorithm and ϵ-statistics to measure the similarity of time series, enabling the identification of root-cause metrics from millions of metrics. The algorithm is implemented in a real-time diagnosis system for production microservice platforms.
- **RCD** (Ikram et al., 2022): RCD is a scalable algorithm for detecting root causes of failures in complex microservice architectures using a hierarchical and localized learning approach. It treats the failure as an intervention to quickly identify the root cause, focuses learning on the relevant portion of the causal graph to avoid costly conditional independence tests, and explores the graph hierarchically. The technique is highly scalable, providing actionable insights about root causes, while traditional methods become infeasible due to high computation time.
- **BARO** (Pham et al., 2024): BARO is a robust, end-to-end RCA framework designed for multivariate time-series data in microservice systems. It integrates anomaly detection and root cause localization using Multivariate Bayesian Online Change Point Detection (BOCPD) to detect failures and estimate their occurrence times. For RCA, BARO introduces RobustScorer, a nonparametric hypothesis testing approach that ranks candidate root causes based on their distributional shifts, using median and interquartile range rather than mean and standard deviation to improve robustness.
- **Nezha** (Yu et al., 2023): Nezha is an interpretable and fine-grained root cause analysis (RCA) method for microservices that unifies heterogeneous observability data (metrics, traces, logs) into a homogeneous event format. This representation enables the construction of event graphs for integrated analysis. Nezha statistically localizes actionable root causes at granular levels, such as specific code regions or resource types, offering high interpretability to support confident mitigation actions by SREs.

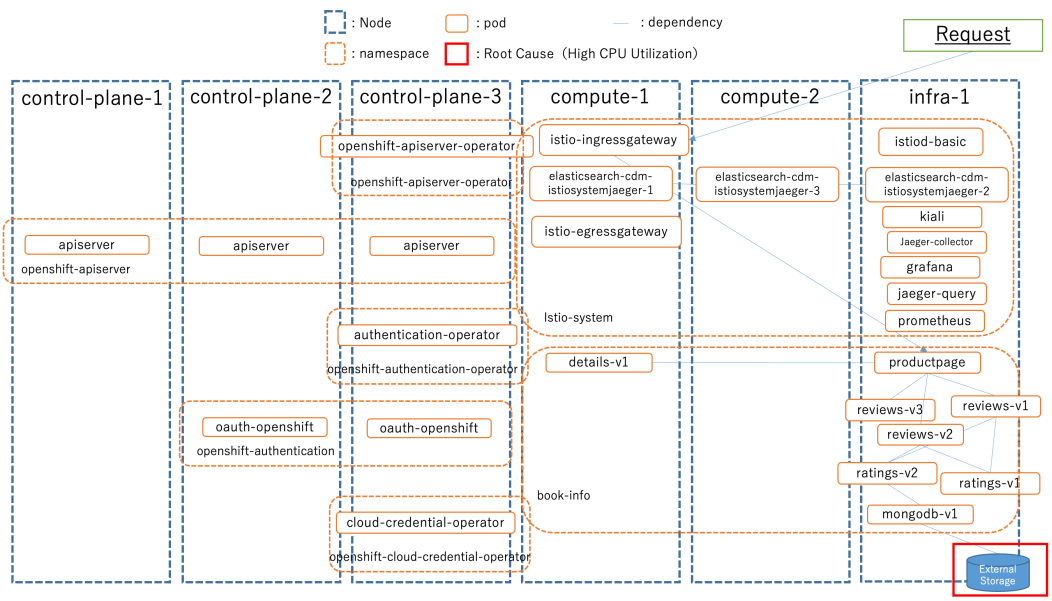


Figure 6: Corresponding to Figure 2 (a). The architecture of Product Review Platform

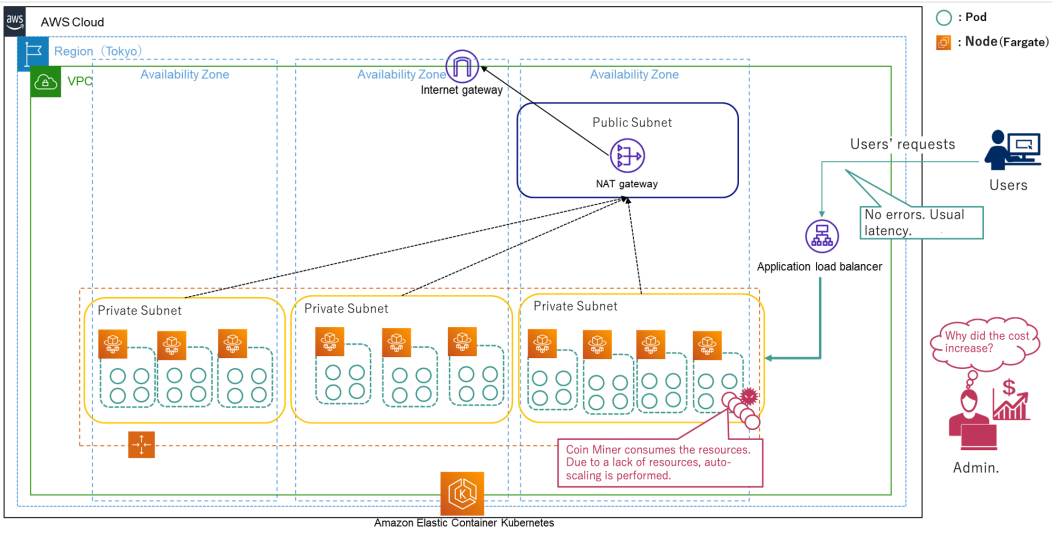


Figure 7: Corresponding to Figure 4 left. Visualization of Cryptojacking system fault scenario. **Right:** External storage failure.

I FIGURES FOR CLARITY

We provide figures related to the system architecture and fault scenarios in this section, for better readability. The architecture of Product Review Platform is shown in Figure 6, and the system fault scenarios are demonstrated in Figure 7 and Figure 8.

J DATASET LABELING METHODOLOGY

We provide more details on the system fault labeling strategy, which comes in two-fold: the root cause labeling process and label validation.

Root Cause Labeling Process.

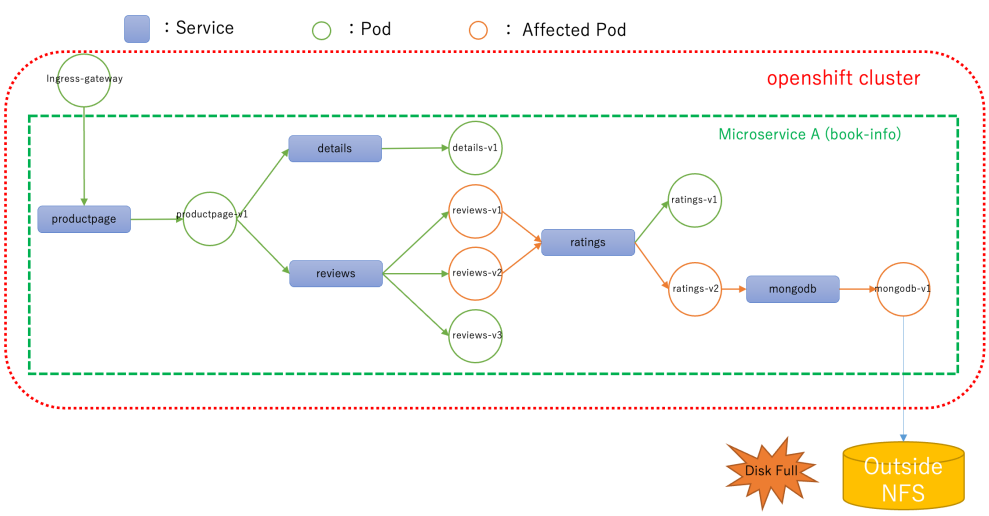


Figure 8: Corresponding to Figure 4 right. Visualization of External storage failure. system fault scenario.

- For each system fault, we designed controlled fault scenarios to mimic realistic fault patterns (e.g., external storage failure, database overload).
- During each controlled fault case, we monitored system behaviors, including metrics and logs, to identify the exact root cause of the fault.
- The ground truth root cause was then labeled based on the specific fault of the system. This ensures high accuracy in root cause labeling, as the faults were systematically induced and their impacts directly observed.

Label Validation.

- To ensure label correctness, we validated the root cause labels by analyzing the system's behavior during and after fault. This involved cross-checking the observed anomalies in system metrics and logs with the expected outcomes of the fault.
- Multiple experts reviewed the labeled faults to confirm the consistency and correctness of the root cause assignments.

K Parameter Settings for All Baseline Models

We provide the detailed parameter settings for all baseline models as follows:

- **PC:** `alpha=0.05` (*significance level for conditional independence tests*), `ci_test='fisherz'` (*type of conditional independence test*).
- **RCD:** `ci_test='chisq'` (*type of conditional independence test*). `k=10` (*top-k root causes*), `alpha_limit=0.5` (*the maximum alpha for search*),
- **ε-Diagnosis:** `test_size=0.6` (*train test split ratio*), `root_cause_top_k=10` (*top-k root causes*),
- **Circa:** `test_size=0.6` (*train test split ratio*), `root_cause_top_k=10` (*top-k root causes*),
- **Baro:** Parameters of this algorithm are related to the Bayesian online change point detection. `r=50` (*magnitude of the hazard function for Bayesian online learning*), `k=3` (*number of standard deviations from the mean to consider as an anomaly*),
- **Nezha:** level=service (detection at the service level)

L Dataset Representativeness

In this section, we aim to show the representativeness of the released dataset. While it is challenging to establish a universal metric for representativeness in benchmarks, we have made significant efforts to ensure the dataset covers diverse fault scenarios:

- **Real-World Fault Scenarios:** The IT domain datasets (Product Review and Cloud Computing) encompass realistic microservice faults such as out-of-memory errors, DDoS attacks, and cryptojacking, as outlined in Section 3.1 and Appendix B. Similarly, the OT domain datasets (SWaT and WADI) include real-world cyber-physical system faults recorded in controlled environments.
- **Diversity of Fault Types:** Across IT and OT domains, we include 10 distinct fault types, ensuring coverage of both transient and persistent system failures. This diversity reflects common issues faced by modern IT and OT systems.
- **Comparative Analysis:** As seen in Table 3 and related discussions, our dataset exhibits performance trends consistent with other benchmarks (e.g., Petshop), supporting its credibility as a representative evaluation platform.
- **Quality Assurance:** All data were collected using industry-standard monitoring tools like Prometheus, CloudWatch, and Elasticsearch. Each fault scenario was validated to ensure it mirrors real-world conditions.