MRCA: Metric-level Root Cause Analysis for Microservices via Multi-Modal Data

Yidan Wang
phoebeyidanwang@gmail.com
The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen)
Shenzhen, China

Zhouruixing Zhu
zhouruixingzhu@link.cuhk.edu.cn
The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen)
Shenzhen, China

Qiuai Fu
fuqiuai@huawei.com
Huawei Cloud Computing Technologies CO., LTD, China
Shenzhen, China

Yuchi Ma
mayuchi1@huawei.com
Huawei Cloud Computing Technologies CO., LTD, China
Shenzhen, China

Pinjia He*
hepinjia@cuhk.edu.cn
The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen); Shenzhen Research Institute of Big Data
Shenzhen, China

ABSTRACT

Due to the complexity and dynamic nature of large-scale microservice systems, manual troubleshooting is time-consuming and impractical. Therefore, automated Root Cause Analysis (RCA) is essential. However, existing RCA approaches face significant challenges. (1) Multi-modal data (*e.g.* traces, logs, and metrics) record the status of microservice systems, but most existing RCA approaches rely on single-source data, failing to understand the system fully. (2) Existing RCA approaches ignore the services' anomaly state and their anomaly intensity. (3) The service-level RCAs lack detailed information for quick issue resolution. To tackle these challenges, we propose MRCA, a metric-level RCA approach using multi-modal data. Our key insight is that using multi-modal data allows for a comprehensive understanding of the system, enabling the localization of root causes across more anomaly scenarios. MRCA first utilizes traces and logs to obtain the ranking list of abnormal services based on reconstruction probability. It further builds causal graphs from services with high anomaly probability to discover the order in which abnormal metrics of different services occur. By incorporating a reward mechanism, MRCA terminates the excessive expansion of the causal graph and significantly reduces the time taken for causal analysis. Finally, MRCA can prune the ranking list based on the causal graph and identify metric-level root causes. Experiments on two widely-used microservice benchmarks demonstrate that MRCA outperforms state-of-the-art approaches in terms of both accuracy and efficiency.

KEYWORDS

Root Cause Analysis, Microservices, Reinforcement Learning, Multi-modal

ACM Reference Format:
Yidan Wang, Zhouruixing Zhu, Qiuai Fu, Yuchi Ma, and Pinjia He. 2024. MRCA: Metric-level Root Cause Analysis for Microservices via Multi-Modal Data. In *39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24), October 27-November 1, 2024, Sacramento, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3691620.3695485

1 INTRODUCTION

Nowadays, microservice systems have become the first choice for large-scale system design [8, 29]. Microservice architecture decomposes complex application systems into multiple small-scale, autonomous service units [40]. This architecture enhances system flexibility and scalability, enabling quick adaptation to business needs and automatic adjustment of service scale for optimal performance and availability [4, 34]. As systems and user demands grow in complexity and diversity, manual root cause analysis (RCA) becomes a challenging and time-consuming task [54]. Delays in addressing these failures can significantly impact user experience and lead to substantial economic losses [12]. Hence, there is a critical need to explore automated algorithms capable of quickly identifying the root causes to mitigate such risks effectively [7].

The demand for RCA automation generated tremendous academic research [38]. These methods are based on runtime data, including logs, traces, and performance metrics such as CPU usage, network received bytes, and response latency. Metric-based RCA usually involves directly analyzing correlations (*e.g.* Pearson correlation coefficient) between services, or constructing causal graphs to pinpoint the root cause [6, 19, 23]. For instance, the ϵ-Diagnosis [36] uses ϵ-statistics to identify key metrics causing small-window long-tail latency in web services. Wang *et al.* proposed Cloudranger [42] that utilizes metrics to perform dynamic causality analysis without presetting topology and identify critical services that cause failures. For trace-based RCA, the process starts by using the fluctuations of latency in traces to detect anomalies. Once anomalies are identified, calling information in traces is then used to construct dependency

*Pinjia He is the corresponding author of this work. E-mail: hepinjia@cuhk.edu.cn

ASE '24, October 27-November 1, 2024, Sacramento, CA, USA

ACM ISBN 979-8-4007-1248-7/24/10…$15.00
https://doi.org/10.1145/3691620.3695485

graphs between services, which help in accurately locating the root cause [11, 45, 47]. Ding *et al.* proposed TraceDiag [9] which uses traces for constructing service dependency graph and optimizes pruning strategies through reinforcement learning. Additionally, Anunay Amar *et al.* proposes LogFaultFlagger [2], which excludes lines from the log that pass the test, more precisely locating the faulty. However, log-based methods are relatively rare because the lack of structure and context of logs make it difficult to locate the root cause alone.

Although there have been significant recent advances in RCA, current methods still face limitations as follows:

(1) *Insufficient exploitation on multi-modal data:* These methods [35] rely on data from single modality, thus failing to capture the complex relationships between different exception patterns, ultimately leading to suboptimal solutions. Table 1 demonstrates anomaly information discovered in multi-modal data. We observe that none of the data modalities can fully cover the five types of faults that commonly occur in microservices systems. For example, the log cannot reflect network congestion and CPU contention. Besides, some methods that utilize multi-modal data also have many limitations. For example, Eadro [22] cannot handle program logic bugs. Nezha [46] relies on pattern mining, which is time-consuming. These methods show significant limitations in detection capabilities and efficiency.

(2) *Current RCA methods overlook the importance of anomaly detection:* The accuracy of anomaly detection is crucial for the overall RCA algorithm, as subsequent localization steps are initiated only after anomalies are detected. Current methods divide anomaly detection and localization into two independent stages. And the root cause localization phase relies on the results of anomaly detection. Inaccurate anomaly detection will cause some false alarms and reduce the accuracy of the whole process. However, the anomaly detection modules of current RCA methods commonly use too simple techniques such as N-sigma [3], where the data utilized by RCA are not even normally distributed. These methods mostly rely on single-modal data and ignore the relationship between data. In addition, customized anomaly detection methods such as MicroHECL [26] may restrict the model's generalization ability and may not be effective in identifying new anomalies.

(3) *Coarse-grained root cause localization at the service level:* Most existing RCA methods provide service-level results, such as MicroScope [25] and MicroRCA [44], which lack sufficient detailed information. This means that Site Reliability Engineers (SREs) cannot deeply understand the pattern of fault to prevent it from happening again. Therefore, automatically locating and determining the types of anomalies (*e.g.* resource depletion or network disconnections *etc.*) is helpful for assisting SREs in rapidly resolving faults. However, the existing metric-level RCA approaches are usually implemented by complex causal analysis which needs to establish causal graphs for each metric, making the response speed of RCA not fast enough. Especially in large systems, delays in resolving faults can result in severe financial losses.

To address the above limitations, we propose MRCA, a highly efficient root cause analysis approach that leverages reinforcement learning to dynamically terminate causal analysis for microservices. The core ideas are as follows: 1) By effectively harnessing the valuable information within multi-modal data, we can achieve more accurate root cause localization. 2) Reinforcement learning techniques are employed to terminate the causal analysis process dynamically, significantly reducing time consumption. As system data becomes increasingly diverse, the difficulty of accurately implementing RCA has also increased. Therefore, it drives us to utilize multi-modal data, which provides more comprehensive and in-depth insights from various dimensions.

In detail, MRCA consists of three components: (1) *Feature learning* incorporates extracting log templates and the trace latency. For traces, we focus on the latency generated by each service call, considering it as a feature of the called services. For logs, we utilize a widely used log parser to model log event occurrences. (2) In *anomaly detection* phase, we use the time-series data processed above and employ the variational autoencoder (VAE) [20] to calculate the difference between normal and abnormal modes, thus minimizing the impact of incorrect detection on the root cause localization process. We obtain an initial rank list, which refers to reconstruction probability. (3) *Root causes localization* applies causal analysis to troubleshoot. In addition, Q-learning is employed to appropriately stop the expansion of the causal graph, thus avoiding time waste and balancing accuracy with speed.

We conducted extensive experiments using data collected from two widely-used microservice benchmarks (*i.e.* OnlineBoutique [13] and TrainTicket [10]). For anomaly detection, MRCA significantly surpasses other baseline approaches, increasing the $F1$ score by 56.4% ~ 115.7%. For root causes analysis, MRCA achieves high top-3 accuracy (81.3%) on average and outperforms all compared approaches by a large margin (25.0% ~ 117.5%). A series of comprehensive ablation experiments further confirm the contributions of incorporating multi-modal data and reinforcement learning.

In summary, our work has the following contributions.

- We identified three limitations of existing RCA methods, which motivates us to explore the application of multi-modal data and an efficient RCA method by employing reinforcement learning.
- We propose MRCA, a fine-grained method that provides root causes at the metric level, which offers an efficient solution for SREs to rapidly resolve anomalies in large-scale microservice systems.
- MRCA significantly enhances the model's speed by employing filtering at the anomaly detection stage and incorporating reinforcement learning algorithms, thereby enabling timely feedback to facilitate system recovery.
- We conduct experiments on two widely used benchmarks to evaluate MRCA and show that MRCA identifies the root cause at the metric level with the best accuracy and efficiency compared to other baseline methods.

2 MOTIVATION

Motivation 1: Existing RCA methods insufficiently exploit multi-modal data, resulting in fewer anomaly types they can cover. Existing RCA methods predominantly use single-source data due to the high complexity of integrating and processing multi-modal data. However, they cannot comprehensively understand complex microservice systems because different anomalies manifest

Table 1: Abnormal patterns in multi-modal data for different failures. '-' represents no detected unusual patterns.

Fault Type	Metric	Log	Trace
CPU Contention	High CPU usage	-	-
Code logic error	-	Error/ Warning	Changed calling relationship
Network Interruption	Error rate raises	Error/ Warning	Latency increases
Network Jam	Throughput decreases	-	-
Configuration Error	-	Error/ Warning	Call interrupt

in different data dimensions, each capturing distinct information shown in Table 1. Traces record the propagation path and time span of requests within the system, logs capture events and state information during system operation, and metrics quantify system performance and resource usage. Therefore, we prepare to integrate diverse data (traces, logs, and metrics), we can capture the full spectrum of system behavior, leading to more accurate and efficient detection and resolution of issues.

Motivation 2: Coarse-grained root cause localization at the service level that increases the time consumption of SREs resolving anomalies. Some existing methods utilize multi-modal data, but they provide root causes at a coarse granularity, limited to the service level [9, 22]. However, in the service-level, engineers need to spend more time and resources to troubleshoot. In addition, this coarse-grained positioning results in poor warning effects, which affects system reliability and availability, thereby affecting user experience.

Therefore, more fine-grained results (such as metric-level results) are required to assist SREs in resolving anomalies rapidly. For example, if we accurately locate the memory usage metric of one service, SREs can determine whether the problem is caused by a memory leak and take more precise interventions, such as adjusting memory management strategies or optimizing code. Therefore, metric-level positioning can improve the accuracy of problem diagnosis and repair efficiency, restore normal operation of the system faster, and reduce downtime and resource waste. Additionally, incorporating causal analysis of inter-service metrics increases the complexity and duration of the process. Therefore, we integrated the Q-learning reinforcement learning algorithm to dynamically terminate the causal analysis, achieving fine-grained and relatively efficient model.

Motivation 3: Both normal and abnormal services are fed into the RCA, resulting in the inaccuracy and high complexity of the localization phase. To the best of our knowledge, previous RCA approaches for microservice systems follow the workflow: 1) Anomaly detection is applied. 2) If an alarm is triggered, then all services are inspected in root cause location phase. However, these RCA algorithms ignore the services' anomaly state and their anomaly intensity. Moreover, normal service will misleading cause location and waste computation resource. Additionally, without a preliminary ranking of anomalies, all detected anomalies are treated equally, which results in low efficiency of RCA. Therefore, we inspect the abnormal services only during the anomaly detection phase to enhance the accuracy and efficiency of subsequent root cause localization.

3 PRELIMINARIES & PROBLEM STATEMENT

3.1 PRELIMINARIES

Multi-modal data. In microservice systems, multi-modal monitoring data plays an essential role in maintaining the system's stability and reliability [51]. Metric data monitors system health and performance by collecting and analyzing system-level and service-level metrics, helping to identify and address anomalies. Log data records detailed event information including timestamps, severity levels, messages, and contextual details, aiding in the rapid identification and resolution of issues [48]. Trace data uses unique identifiers for each service request and its spans, capturing the duration and calling relationships of requests across microservices, helping to detect and analyze abnormal service behavior [21]. These three types of data are considered multi-modal data because metric, log and trace represent numerical data, text data and distributed tracing data respectively [5]. The combination of metrics, logs, and traces provides a comprehensive view of system behavior, enhancing the accuracy and efficiency of problem detection and resolution in microservice systems.

Reinforcement learning (RL). RL is a machine learning method where an agent interacts continuously with its environment, learning to take optimal actions to maximize cumulative rewards [24]. The agent adjusts its strategy based on reward feedback in each state, thereby gradually optimizing its decision-making process [55]. We can incorporate different levels of requirements (such as time consumption, accuracy) into the reward components, enabling the model to balance time and accuracy. Moreover, the reinforcement learning approach possesses several advantages, making it suitable for RCA, which are as follows:

- **Adaptability**: RL learns universal policies, making it well-suited for the complex and variable structures of microservice systems [41, 50]. This adaptability enhances RCA algorithms' effectiveness in diverse environments.
- **Few prior knowledge required**: RL improves decision-making through trial-and-error interactions, requiring little prior knowledge [18, 37]. There is a lack of extensive public datasets that contain multi-modal data. Therefore, incorporating RL can improve RCA by using existing limited data more effectively.
- **Optimize complex strategies**: RL dynamically integrates multiple variables to optimize decision-making. This helps RCA analyze numerous metrics more effectively, leading to better termination strategies.

3.2 Problem Statement

Consider a large system with $\mathcal{K}$ microservices, where each service independently generates system logs, trace data, and metric data. Within an observation window of length T, for the k-th microservice, we define the multi-source data as $\mathbf{X} = \{(\mathbf{X}_k^{\mathcal{L}}, \mathbf{X}_k^{\mathcal{T}}, \mathbf{X}_k^{\mathcal{M}})\}_{k=1}^{\mathcal{K}}$, where $\mathbf{X}_k^{\mathcal{L}}$ represents log events ordered by time, $\mathbf{X}_k^{\mathcal{T}}$ includes trace records, and $\mathbf{X}_k^{\mathcal{M}}$ represents time series of multiple metrics. Our algorithm mainly consists of two stages: when $\mathbf{X}_{[1:\mathcal{K}]}$ is available, the system uses log and trace data for anomaly detection. Upon detecting an anomaly, the system obtains an initial ranking list $\mathbf{List}_{initial} = \{\mathbf{X}_m, \mathbf{X}_n, ...\}$ based on ascending reconstruction probability denoted by $\mathbf{P} = [p_m, p_n, ...] \in [0, 1]^{\mathcal{K}}$ and initiates subsequent steps. In root cause localization, we apply the causal analysis to get the final ranking list $\mathbf{List}_{final} = \{(\mathbf{X}_i, \mathbf{Y}_a), (\mathbf{X}_j, \mathbf{Y}_b), ...\}$ where $\mathbf{Y}_a, \mathbf{Y}_b$ mean the type of anomalies. The entire process is built on a parameterized model $\mathcal{F} : \mathbf{X} \rightarrow \mathbf{List}_{final}$.

4 METHODOLOGY

In this section, we first present the overall workflow of MRCA. Then we explain each step in detail, including feature extraction, using VAE for anomaly detection, causal graphs construction via Granger [14] algorithm, learning termination strategy via Q-learning, and pruning rule.

4.1 Overview

The overall workflow of MRCA is illustrated in Figure 1. MRCA consists of three phases: *multi-modal data feature learning*, *anomaly detection* and *root causes localization*.

We briefly introduce these three phases. **Feature learning** incorporates extracting log templates and the latency information contained in traces. For traces, we primarily focus on the latency generated by various service calls, treating latency as one of the characteristics of the called services. For logs, we adopt a state-of-the-art log parsing approach Drain [16] to extract static log templates for each service and then count the frequency of these templates. These features are aggregated according to timestamps to obtain unified time-series data. VAE model is trained offline using aggregated data collected during fault-free periods to learn the normal pattern of the system. In **anomaly detection** phase, we use the time-series data processed in the feature learning phase and employ the trained VAE to distinguish between normal and abnormal modes. An anomaly alert is triggered when the time-series data significantly deviates from the normal pattern. This deviation is indicated by a sharp drop in reconstruction probability. Abnormal services are also ranked based on the reconstruction probability. Our anomaly detection model not only provides more accurate alarms but also selects services with higher anomaly score as input for the next stage, reducing the complexity and time of subsequent analysis. **Root cause localization** conducts causal analysis based on metric data, which constructs and prunes a causal graph to identify the root causes of anomalies at the metric level. Additionally, Q-learning is employed to appropriately halt the expansion of the causal graph, preventing time waste and balancing accuracy with speed.

4.2 Multi-modal Data Feature Extraction

This phase aims to extract key features from the multi-modal monitoring data (*i.e.* logs and traces). In this part, the input is time series data $\{(\mathbf{X}_k^{\mathcal{L}}, \mathbf{X}_k^{\mathcal{T}})\}_{k=1}^{\mathcal{K}}$, the output is For trace learning, the latency of traces effectively reflects the performance characteristic of the operation, specifically indicating the time taken to complete a request. So we choose latency as the feature of the called service. Specifically, we process all traces over a given time period to extract the latency of all calls and construct time-series data. We use a 5s time window, averaging the duration of all calls within time window to obtain the latency value for each sample. According to previous studies[1, 52], the types of logs generated by systems differ between normal and abnormal states. Therefore, our insight is to extract templates and determine the anomaly degree of service based on log frequency fluctuation. For log learning, we use a widely-used log parser Drain [16] to extract the static log templates of each service, and then count the frequency of occurrence of these templates. Then these features will be aggregated into a whole according to the timestamp. The extraction of these two types of features not only helps us understand the operating mode of different services. And it provides the model with the necessary data basis to analyze the deviation of features.

The process of feature fusion can be outlined as follows: Each service within the system is characterized by a set of m Template features along with a single Latency feature. These features are combined or fused over a period of time based on time series. After the above process, we can get the characteristic time series data $\mathcal{F} = \{\mathcal{F}_k\}_{k=1}^{\mathcal{K}}$. A critical assumption in this process is the selection of a time window size denoted as n, which implies that each individual feature is represented by n dimensions of data. This approach facilitates the integration and analysis of multiple features across the specified time intervals, contributing to a comprehensive understanding of system behavior and performance.

4.3 Anomaly Detection

During the anomaly detection process, by analyzing log and trace data, the model can generate anomaly alerts and calculate anomaly scores. The scores can help us evaluate the abnormality of the service and sort the services according to the abnormality level. This process is based on the reconstruction probability of Variational Autoencoder (VAE) to quantitatively analyze the difference in log frequency and tracking delay between normal and abnormal states.

VAE consists of two parts: encoder and decoder. The encoder maps input data to a latent space distribution (usually Gaussian), while the decoder maps latent space points back to the original data. We use the logs and traces of the historical data when the system is running normally to train the VAE model to obtain the variance and mean. Then input the data in the online stage into the VAE to calculate the reconstruction error. The algorithm in this paper obtains the anomaly score by calculating the reconstruction probability. Because it is more interpretable and takes into account the uncertainty between differences. The formula for calculating the reconstruction probability is as follows:

$$p(x|z) = \mathcal{N}(x; \mu(x), \sigma^2(x)I) \tag{1}$$

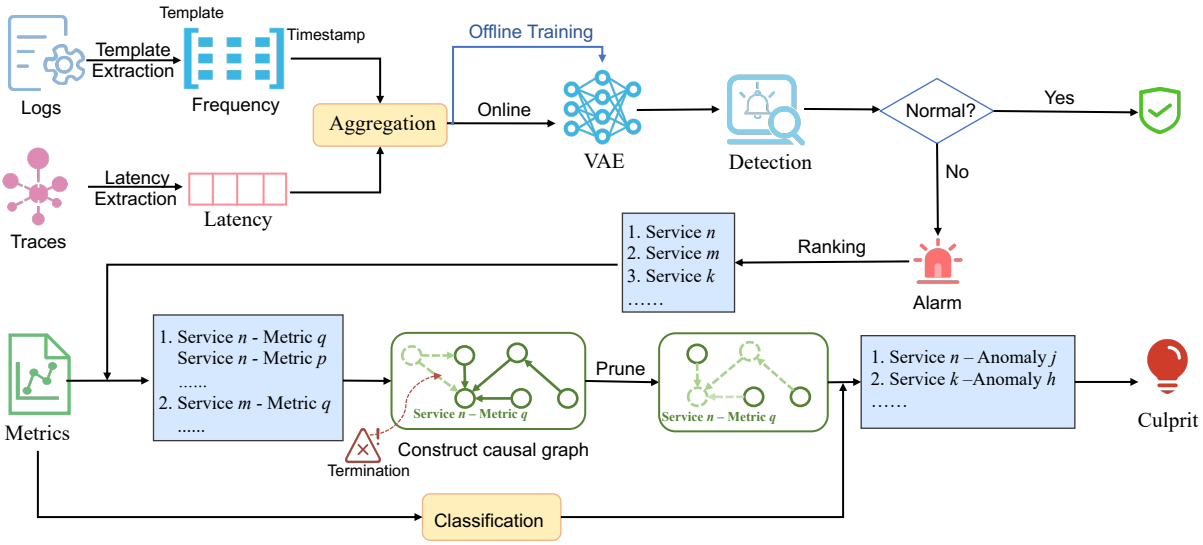

Figure 1: The overflow of MRCA.

In this formulation, x represents the input and z stands for latent representations. $\mu(x)$ and $\sigma^2(x)$ are the mean and variance of the decoder output respectively.

First, we use the $\mathcal{F} = \{\mathcal{F}_k\}_{k=1}^{\mathcal{K}}$ when the system is running normally according to the time window size T. These data are input into the VAE model to train it. Thus VAE is able to reconstruct the normal data with high accuracy. When an anomaly occurs, there are significant changes in the data distribution. Once an abnormality occurs, due to the difference between abnormal data and normal data, it is difficult for VAE to reconstruct abnormal data, which means the reconstruction probability decreases significantly. We need to set a threshold ϵ for the reconstruction probability. When the reconstruction probability of any service is lower than this threshold, the system will generate an anomaly alert and initiate subsequent processing steps.

After receiving the anomaly alert, the next step is to obtain the reconstruction probabilities of different services, the abnormal services can be ranked. The lower the reconstruction probability, the higher the anomaly score. Meanwhile, we can incorporate an attention mechanism to enhance the specific pattern recognition ability of the VAE model. For example, when an anomaly occurs, the model can pay special attention to those log templates that never appear in normal mode, or those log templates that usually appear frequently in normal mode but suddenly reduce significantly in frequency. Such improvements enable VAE to more effectively identify and respond to abnormal situations. Besides, we will remove the normal service from the list which has a reconstruction probability higher than ξ. After the above processing, we get a ranking list $\mathbf{List}_{initial} = \{\mathbf{X}_m, \mathbf{X}_n, ...\}$.

4.4 Root Causes Localization

4.4.1 Causal Graphs Construction via Granger Algorithm. Causal learning aims to uncover causal relationships through data analysis, enhancing our understanding of phenomena. It extracts key patterns to identify decisive factors. Despite challenges like high costs and biases, recent advancements demonstrate its efficacy across various fields, utilizing causal graphs and structural equations. In this paper, we use the Granger algorithm for causal analysis since it can accurately determine the causal relationship between time series through statistical tests. Granger's core principle is that X_t causes Y_t if past values of time series X significantly improve the prediction of the current value of time series Y_t. Granger causality test is designed to detect such causality ($X_t \rightarrow Y_t$). We only consider causal relationships between service-metric pairs within the time window T_{causal} after the anomaly is detected. And we use the $\mathbf{List}_{initial}$'s highest-ranked nodes to first expend.

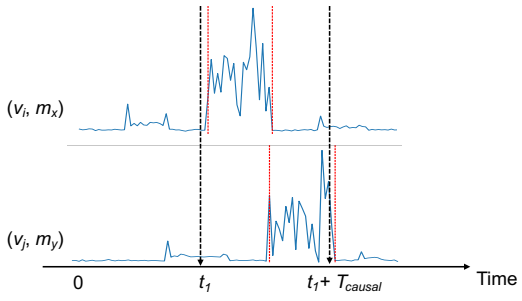

Figure 2: Example of granger causality test on time series.

In the process of causal analysis, we use $\{\mathbf{X}_k^{\mathcal{M}})\}_{k=1}^{\mathcal{K}}$ with time series relationships. To analyze the causality between service-metric pair (v_i, m_x) and (v_j, m_y), we use their metric data, denoted as time

series $x_t, y_t \in [0, T_{causal}]$. Then, we construct two linear regression models to predict y_t using past observations and apply ordinary least squares to estimate the model parameters with the time series data x_t and y_t. The difference between these two models is that the full model carries additional information from service-metric pair (v_i, m_x). The functions are as follows:

$$\mathcal{H}_{part}: \quad \hat{y}(t) = \sum_{i=1}^{p} \alpha_i^{part} y_{t-i} + b^{part} \tag{2}$$

$$\mathcal{H}_{full}: \quad \hat{y}(t) = \sum_{i=1}^{p} \left(\alpha_i^{full} y_{t-i} + \beta_i^{full} x_{t-i}\right) + b^{full} \tag{3}$$

In the formula, $\hat{y}(t)$ is the predicted value and α_i^{part}, b^{part}, α_i^{full}, β_i^{full}, b^{full} are the parameters. We first assume that the additional information contained in x_t has no effect on the prediction of y_t. As a result, statistical calculation proves that the value:

$$F = \frac{(SSE_{part} - SSE_{full})/(d_{full} - d_{part})}{SSE_{full}/(T_{causal} - d_{full} - 1)}, \tag{4}$$

SSE_{part} and SSE_{full} represent the error sum of squares from the above models. d_{full}, d_{part} are the degrees of freedom of the models, which are τ, 2τ. We can estimate the probability of the null hypothesis from the F distribution, and if the probability is below the significance level γ, it indicates that there is Granger causality from x_t to y_t. In other words, the service-metric pair $(v_i, m_x) \rightarrow (v_j, m_y)$.

4.4.2 Learning Termination Strategy via Q-learning. After the first round of filtering in the anomaly detection phase, it is still complicated to establish the causal graph. For example, there are still a services, and each service has b metrics. Then there will be $a \times b$ nodes in the causal graph. Too complex causal graphs will also affect subsequent processing. Manually constructing pruning models is time-consuming and ineffective. Moreover, artificially constructed

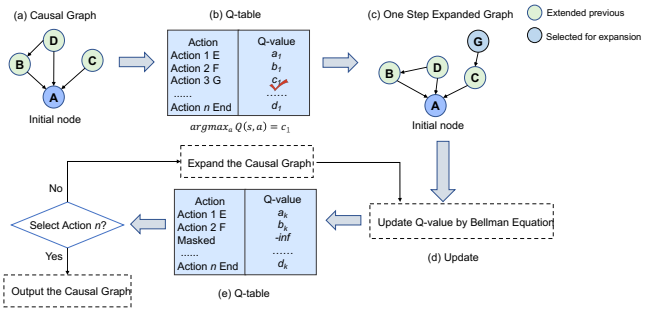

Figure 3: Example of termination strategy.

pruning models may have poor generalization performance. It is not suitable for systems with high complexity and continuous changes that require frequent modifications. To overcome the above problems, we propose an automatic termination strategy with Q-learning. This strategy can balance the time-consuming efficiency and accuracy. Once trained, this policy is capable of being deployed on fresh incidents autonomously, without the need for human intervention. Figure 5 shows the overflow of extensions to the causal graph. Before each expansion, we will select the action with the maximum Q-value for expansion, and after the expansion, we will update the Q-table globally. The Bellman function is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{5}$$

Markov Decision Process (MDP) [17]: When expanding causal graphs, at any given moment, the impact of actions on the environment is only related to the current state and is independent of previous states. In this paper, we use MDP which consists of three main elements: states, actions, and rewards. At each time step t, given a current state st, the best action is selected based on the value of the Q-table. The action is to expand two nodes to the current node. Update the Q-table according to the reward r received by the action until the entire list is traversed. The different components of the MDP are defined as follows:

- **State** $s_t \in \mathbf{S}$ contains the number of nodes, edges, layers, and services of the current causal graph.
- **Action** Action space $A = \{a_0, a_1 ...., a_i, ..., a_m, a_{m+1}\}$, where m represents the number of services in the initial ranking list. The first m actions ai represent the expansion of the $i-th$ node when operating the current node. In particular, when a node has been expanded, all actions containing it will be unavailable. The last action is to end expansion. When the model determines that the current causal graph can meet the accuracy requirements of RCA, it should stop expansion to improve efficiency.
- **Reward Function** The reward function contains two components, one is the complexity of the graph, and the other is the accuracy of RCA. This can take into account the accuracy and efficiency of RCA. The reward function is shown in Equation 6.

$$R(s_t, a_t) = \alpha \cdot r^{\text{com}} + \beta \cdot r^{\text{rca}} \tag{6}$$

$$r^{\text{com}} = -(|N| + |E| + \frac{|E|}{|N|}^2) \tag{7}$$

$$r^{\text{rca}} = \sum_{i=1}^{N} \sigma_k PR@k \tag{8}$$

In the above equations, α and β denote the weights assigned to the rewards. $|N|$ and $|E|$ represent the number of the services and edges. r^{rca} measures the accuracy of the RCA based on the pruned trace structure. The value of σ decreases as k assumes the values of 1, 3, and 5, in that order.

Incorporating reinforcement learning into the construction of causal graphs allows for a dynamic analysis termination mechanism. This approach improves the speed of RCA while ensuring sufficient accuracy. It permits the expansion of the causal graph to stop promptly when it reaches the relatively optimal state. Instead of stopping after traversing the ranking list, thereby avoiding time wastage due to over-extension.

4.4.3 Prune Strategy. Each point in this causal graph respectively stands for one service-metric pair. After obtaining the causal graph, we can perform pruning by analyzing the relationships between nodes. In a causal graph involving time relationships, the cause event must occur before the effect event. Therefore, based on the

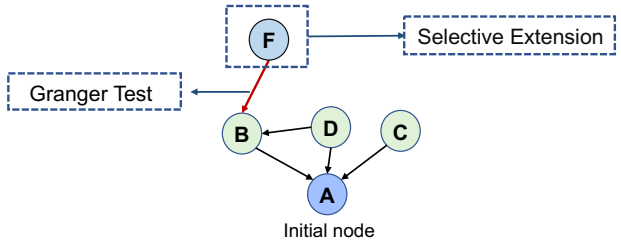

Figure 4: Use the granger test to determine edge existence and select the node to expand based on Q-value.

cause-and-effect diagram, we can determine the chronological order in which abnormal events occurred. The root service causing the exception usually has no parent node, meaning it does not depend on other nodes. Thus, we can identify the root service that may cause the anomaly by removing the nodes with parents in the cause-and-effect graph, leaving only the root node and orphan nodes. Ultimately, we can get the $\mathbf{List}_{final} = \{(\mathbf{X}_i, \mathbf{Y}_a), (\mathbf{X}_j, \mathbf{Y}_b), ...\}$.

Through this process, we can further update and refine the initially obtained abnormal service rankings, transitioning from preliminary service-level results to specific metric-level results. This method enhances the accuracy of system anomaly detection and the efficiency of troubleshooting, providing better assistance to SREs in resolving subsequent anomalies.

5 EXPERIMENTAL EVALUATION

In this secion, we aim to evaluate our method to answer the following research questions (RQs):

- **RQ1:** How much does anomaly detection benefit from multi-modal data?
- **RQ2:** How does MRCA perform at the metric-level and service-level?
- **RQ3:** How much do the different data sources and pruning module contribute to RCA?
- **RQ4:** Can the pruning module of MRCA reduce the time cost of RCA?

5.1 Experimental Setup

Experimental data. The public data set provided by Nezha is used. This data set is collected on OnlineBoutique (OB) and TrainTicket (TT). TrainTicket is a platform that provides railway ticketing services through which users can easily inquire, book, and pay for train tickets. Similarly, OnlineBoutique is a microservice system focused on e-commerce. These two systems have been widely used in many previous studies and have become important experimental platforms for researchers to study microservice architecture and performance. There are three categories of anomalies injected into the system: CPU contention anomalies [45], network congestion anomalies [25, 27], and code defect anomalies [28, 48]. The dataset composition is shown in Table 2, and the ratio of training to test set is 60% and 40%.

Evaluation Metrics. To evaluate the performance of the model, we selected two widely used metrics to measure its performance in various aspects [32]:

(1) Since anomaly detection is a binary classification task, we use Precision, Recall, and F1-Score (denoted as F1) to evaluate the performance of models:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{9}$$

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

In the above equations, TN is the number of normal samples that are correctly classified. FN represents the number of missed anomalies. FP stands for the number of normal samples triggering alarms.

(2) Precision@K (PR@K): PR@K evaluates the top K RCA results for any matches to the injected anomaly. The higher the PR@K, the more accurate the RCA results are, which means the better the algorithm performance, defined as:

$$PR@K = \frac{1}{|A|} \sum_{a \in A} \frac{\sum_{i<K} R_a(i) \in V_a}{\min(K, |v_a|)} \tag{12}$$

$PR@K$ is the precision of top K results. In this formulation, $|A|$ represents the collection of all anomaly cases. $R_a(i)$ represents the rank of the case a. $R_a(i) \in V_a$:If the result ranked at i is a true root cause, then the value is 1, otherwise, it is 0. v_{rc} is a collection of root causes.

(3) Mean Reciprocal Rank (MRR): By averaging the reciprocal of the rank of the correct answer for each query, a metric is obtained that measures the model's ability to correctly locate the cause of an anomaly. This indicator pays special attention to the ranking of the first correct answer and is an effective indicator to evaluate the system's ability to give the correct root cause at top1:

$$MRR = \frac{1}{|A|} \sum_{i=1}^{|A|} \frac{1}{Index_i} \tag{13}$$

$Index_i$ represents the ranking position of the correct answer returned by the system in the i-th query.

Baseline. To evaluate the accuracy and efficiency, we compare our method with the following baselines.

- CloudRanger[42]: It proposes a dynamic causal analysis method that does not rely on preset topology to quickly identify problem services in cloud services. This system can be quickly deployed on a variety of cloud-native systems without requiring prior knowledge.
- MicroCause [32]: It combines the path conditional time series algorithm and the time causality-oriented random walk method, effectively integrating the sequential relationship of the time series and the causal relationship of the monitoring data.
- TraceDiag[9]: It uses reinforcement learning to formulate pruning strategies for service dependency graphs to improve the efficiency of RCA. This strategy has strong interpretability and generalization performance.

Table 2: Data set composition.

Benchmark	Sum anomalies	Resource faults	Code faults	Number of logs	Number of traces
OnlineBoutique	56	42	24	3,907,981	80,934
TrainTicket	45	20	25	1,352,335	51,604

- Nezha[46]: It applies multi-modal data to pinpoint root causes by analyzing event patterns and contrasting normal with faulty operational phases.
- ϵ-Diagnosis [36]: It introduces a new web service latency issue, small-window long-tail latency, and proposes a cost-effective, unsupervised algorithm that uses statistical tests to detect root causes by comparing time series data across numerous metrics.

Table 3: Performance comparison for anomaly detection.

Approaches	*F1*	*Precision*	*Recall*
CloudRanger (2018)	0.414	0.379	0.458
MicroCause (2020)	0.519	0.467	0.583
TraceDiag (2023)	0.571	0.538	0.609
Nezha (2023)	0.400	0.457	0.356
MRCA (Ours)	**0.893**	**0.875**	**0.913**

Implementation and Settings. We implement the prototype of our method built on Python 3.10. All experiments are conducted on a Mac server with an Intel Xeon Gold 5318Y 2.10GHz CPU, 256 GB RAM, and 1TB SSD Disk. The hyperparameters of the experiment include the sliding window length, sliding step size, and manually adjusted thresholds. Specifically, we set the sliding window length to 20 to adequately capture the relationships between timestamps in the time series. The sliding step size is set to 5, and thresholds are selected based on reconstruction errors from the training set. The VAE model is trained with a batch size of 64 for 1000 epochs, using a learning rate of 10^{-3}, and a regularization weight of 10^{-5} to prevent overfitting.

5.2 Experimental Results

5.2.1 ***RQ1****.* How much does anomaly detection benefit from multi-modal data?

Ground truths are determined based on the operations known to involve fault injections. Once the fault is injected, data within the observation time window are regarded as abnormal data; the remaining data is treated as normal data. Table 3 presents the performance comparison of the anomaly detection processes in different RCA approaches. We can draw the following conclusions:

(1) MRCA significantly outperforms other methods in terms of F1 score, Precision, and Recall, indicating that it has fewer missed anomalies and false alerts. A more accurate anomaly detection process aids subsequent cause location. If the anomaly is not detected correctly, subsequent steps will not be performed, greatly affecting the accuracy of RCA. Likewise, reducing false alarms avoids wasting system resources and ensures the efficiency of the analysis process.

(2) Generally, multi-modal data methods perform better than single-modal data methods in the anomaly detection process. As shown in Table 1, when only using a single data source such as traces, some types of anomalies may be missed. MRCA combines log and trace, which can complement each other and cover abnormal situations more comprehensively.

(3) In addition, MRCA also uses VAE as an anomaly detection model, which is a deep learning model. By learning and generating data through a probabilistic model, its regularized latent space helps capture the core features of the data. This benefits the generalization ability and recognition capability of a new kind of fault. We can also choose advanced deep-learning models in further work to improve model performance.

In summary, MRCA is greatly improvement at detecting anomalies and improving $F1$ score by 56.4% $\sim$ 115.7%. The high accuracy of anomaly detection is helpful in reducing the noise in the inputs for subsequent cause location steps.

5.2.2 ***RQ2****.* How does MRCA perform at the metric-level and service-level?

To demonstrate our method's effectiveness in fault localization, we compare its accuracy with baselines, visually highlighting its advantages. Besides, our research compares the accuracy of data sets collected from two different platforms, providing a comprehensive perspective and data support for our analysis. Table 4 and Table 5 present the comparison between SOTA methods in service-level and metric-level, respectively, underpinning the following conclusions:

First, we evaluate the accuracy of MRCA at the service-level. As the Table 4 shows, MRCA achieves *PR*@1 of 60.1%, *PR*@3 of 81.3%, *PR*@5 of 88.2%, *MRR* of 71.3%, and improve *PR*@1 by 74.72% $\sim$ 367.67% on average at service-level. Since code errors generally don't reflect in metrics, we only measured the accuracy of identifying the resource anomalies at metric-level. And MRCA achieves *PR*@1 of 71.9%, *PR*@3 of 85.4%, *PR*@5 of 90.3% and improves *PR*@1 by 23.8% $\sim$ 73.8% compared to baselines on average at metric-level. The above results demonstrate the superior accuracy and effectiveness of MRCA in localizing anomalies compared to other methods. Nezha's accuracy is lower than reported in its original paper because this paper calculates the accuracy across all samples, while its original paper reports the accuracy based on conditional probability.

By logically leveraging multi-modal data, MRCA enables a more comprehensive understanding of the system's status, thereby allowing for more accurate root cause localization. Relying solely on trace or metric data for RCA has significant limitations. Trace-based approaches lack the granularity needed to capture subtle performance anomalies, while metric-based methods miss the detailed contextual information provided by traces, leading to incomplete or inaccurate diagnoses. Besides, MRCA obtains the initial ranking of abnormal services during the anomaly detection stage and

Table 4: Comparison of approaches at service-level.

Approaches	OnlineBoutique				TrainTicket			
	PR@1	*PR@3*	*PR@5*	*MRR*	*PR@1*	*PR@3*	*PR@5*	*MRR*
CloudRanger (2018)	0.167	0.411	0.661	0.328	0.133	0.378	0.625	0.294
MicroCause (2020)	0.232	0.589	0.714	0.412	0.222	0.489	0.578	0.356
TraceDiag (2023)	0.304	0.643	0.768	0.473	0.356	0.533	0.689	0.465
Nezha (2023)	0.321	0.357	0.357	0.337	0.378	0.444	0.444	0.401
MRCA (Ours)	**0.589**	**0.804**	**0.875**	**0.691**	**0.622**	**0.822**	**0.889**	**0.735**

Table 5: Comparison of PR at metric-level.

Approach	OnlineBoutique			TrainTicket		
	PR@1	*PR@3*	*PR@5*	*PR@1*	*PR@3*	*PR@5*
MicroCause (2020)	0.238	0.381	0.548	0.250	0.350	0.550
ϵ-Diagnosis (2019)	0.262	0.429	0.548	0.300	0.550	0.600
Nezha (2023)	0.333	0.381	0.357	0.400	0.450	0.450
MRCA (Ours)	**0.738**	**0.857**	**0.905**	**0.700**	**0.850**	**0.900**

then performs causal analysis to further adjust the ranking, thereby efficiently obtaining more accurate results.

5.2.3 ***RQ3***. How much do the different data sources and pruning module contribute to RCA?

Data sources Ablation In this part, we conducted a series of ablation experiments to explore the contribution of different data sources and pruning module to the results:

- MRCA w/o $\mathcal{L}$: Drops logs while inputs traces and KPIs by removing the log modeling module.
- MRCA w/o $\mathcal{T}$: Drops latency extracted from traces by removing the trace modeling module
- MRCA w/o $\mathcal{A}$: Drop attention mechanism from the model.
- MRCA w/ $\mathcal{A}$ in $\mathcal{T}$: Shift attention from the log to the trace.

The ablation study results are shown in Table 6, and we focus on root cause location which is our major task. All data modalities work together to achieve the highest efficiency of MRCA, but their contributions are not exactly the same.

Table 6: Experimental results of the ablation study.

Variants	OnlineBoutique		TrainTicket	
	PR@1	*PR@5*	*PR@1*	*PR@5*
MRCA	**0.589**	**0.875**	**0.622**	**0.889**
MRCA w/o $\mathcal{L}$	0.482	0.768	0.533	0.733
MRCA w/o $\mathcal{T}$	0.554	0.857	0.600	0.822
MRCA w/o $\mathcal{A}$	0.534	0.839	0.578	0.800
MRCA w/ $\mathcal{A}$ in $\mathcal{T}$	0.518	0.803	0.556	0.756

In particular, traces contribute the least, since MRCA w/o $\mathcal{T}$ is the second best. Meanwhile, logs contribute the most, because dropping them from the data has a major impact on accuracy. A significant portion of the anomalies we inject into the two benchmark systems are related to the code area. These code faults are correlated with changes in log frequency rather than trace latency.

In addition, we explored the contribution of the attention mechanism. Experimental results show that the attention mechanism improves the accuracy of the results. However, it is less effective on traces compared to logs. Therefore, MRCA focuses the attention mechanism on logs during the anomaly detection phase. MRCA w/ $\mathcal{A}$ in $\mathcal{T}$ actually leads to worse performance than MRCA w/o $\mathcal{T}$, further demonstrating the importance of log in the data collected by two benchmarks. When deploying MRCA on other datasets or platforms, the attention mechanism can be reasonably adjusted based on the frequency of different types of anomalies.

In conclusion, the involved data sources and model all contribute to the effectiveness of MRCA. Different data contribute to varying degrees, which emphasizes the importance of properly modeling multi-modal data.

Parameter Sensitivity The termination strategy is a key to the accuracy and efficiency of root cause localization. We evaluate the effect of the termination strategy by analyzing how the accuracy and time of root cause localization change with the hyperparameters. In Section 4.4.2, we configured two parameters, α and β, in the reward function of the Q-learning algorithm. The α and β respectively represent the complexity of the causal graph and the accuracy of the RCA.

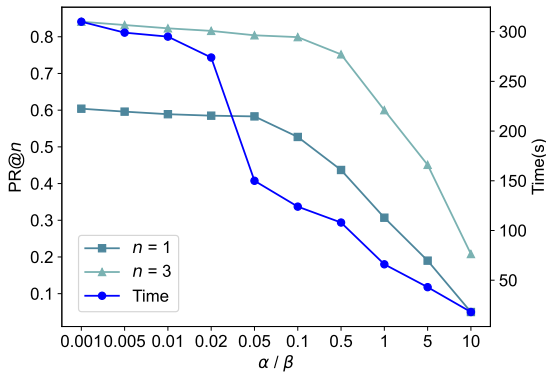

Figure 5: The sensitivity of the reward function parameters.

To the end, we set different α/β, and measure the accuracy and time. We choose the *PR*@1 and *PR*@5 as the indicator of accuracy. The results of the evaluation are shown in Figure 5. It can be seen that both the accuracy and time decrease with the increase of the α/β, as the analysis process is terminated earlier and fewer services are reached and considered. However, within a specific range of parameter settings, accuracy remains relatively constant while there is a sharp decline in time, thereby demonstrating the efficacy of the strategy. In order to balance accuracy and time, the best α/β is 0.05 for availability issues.

5.2.4 ***RQ4***. Can the pruning module of MRCA reduce the time cost of RCA?

To answer this question, we compare the average time to locate a single anomaly between the baselines and MRCA on two benchmarks. The result is shown in Figure 6. We can find that the time cost of RCA on platform TrainTicket is generally higher than that on the platform OnlineBoutique.Since TrainTicket offers a wider range of services and features, it leads to more complex data and consequently higher processing times. MRCA can terminate the causal analysis process in time. In RCA, the relatively time-consuming part is the causal analysis process. Although we have introduced a pruning module that adds some time overhead, the overall time savings significantly outweigh this additional cost. Therefore, the MRCA required will be greatly reduced, as can be observed from the result. For these two benchmarks, the difference in our time consumption is not very large, which proves the stability of MRCA and its great potential to be expanded to large systems.

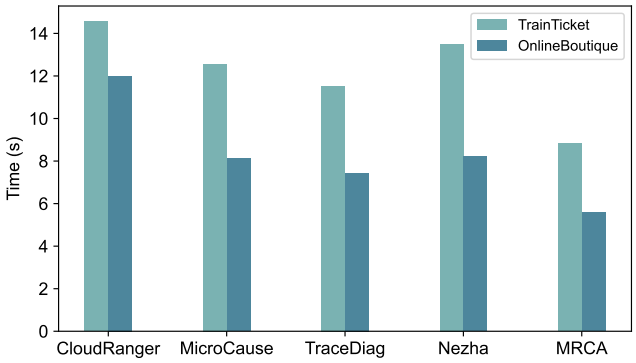

Figure 6: Average time consuming to localize a single anomaly by different methods.

6 RELATED WORK

In today's data-intensive system environment, the data that the system can collect is becoming increasingly rich and diverse. Therefore, how to reasonably utilize the multidimensional data recorded by these systems to perform RCA has become a key area of research. In view of this, it is particularly important to accurately process multi-dimensional data and screen out indicators that are closely related to the root cause of the problem. Rajeev Gupta *et al.* [15] proposed a problem-determination platform with multi-dimensional knowledge integration. There are also some studies [22, 39, 49] using machine learning or deep learning methods to process multi-modal data. The above methods are explorations of utilizing multi-modal data, but they do not fully utilize the effective information in multi-modal data.

The collection of metrics contains the time dimension, thus revealing the sequence. This feature makes adding time series relationships the key to determining the direction of causality and identifying potential confounding variables. In addition, time series analysis can be used to explore nonlinear causal relationships and further improve the accuracy of results. Zheng *et al.* [53] used contrastive learning to extract features in time series to implement the RCA algorithm. Yuan *et al.* proposed an improved PC algorithm to capture the sequential relationship of time series data [32]. In the field of causal analysis, researchers have implemented much more efficient algorithms [33, 56]. This allows us to use more advanced causal analysis algorithms to optimize the speed and accuracy of our algorithms.

By locating the root cause at the metric level, SREs can more accurately pinpoint the problem. This avoids a misunderstanding of the problem and leads to more targeted solutions. Additionally, it helps the team objectively assess the extent of the problem. The existing RCA algorithms [30, 31, 43, 57] positioned at the metric-level are too complex in constructing casual graphs. Making them unsuitable for quickly locating root causes in large systems. Our algorithm will use methods such as pruning to obtain simpler results.

7 DISCUSSION

7.1 Limitations

MRCA is a data-driven model specifically designed for the scope of reliability management. Therefore, it is inherently limited to detecting and locating anomalies that manifest within observable data. Consequently, MRCA is unable to identify and pinpoint issues that do not leave an imprint in the data it analyzes, such as external attacks or internal disruptions. Additionally, for faults like code errors which generally do not cause significant changes in metric, we can only localize them at the service-level instead of metric-level.

Moreover, MRCA is applicable to anomalies that can be reflected in the multi-modal data we utilize. However, some systems cannot fully collect these multi-modal data. Although our approach allows for the lack of some data modalities, it is best to provide all data types to maximize the model effect. With the development of microservice systems, there are now many integrated tools that can be loaded into the system to easily monitor and obtain these multi-modal data.

In addition, MRCA employs representative metrics to determine the type of anomalies. However, more complex anomalies might manifest across multiple metrics, or anomalies in a specific metric could be caused by various types of anomalies. Therefore, future work could involve exploring the pattern matching between anomaly types and metrics to achieve more accurate results and enhance the generalizability of our model. In about 50% of fail cases, services with low activity which provide less normal data, making it challenging to detect anomalies using MRCA. Specifically, in 40% of fail cases, code errors that cause network timeouts, evident only

in the call chain, are not addressed by MRCA due to the lack of exploitation of service dependencies for code errors.

7.2 Threats to Validity

(1) Internal threat: The primary internal threat is the validity of the baseline implementation, which is based on our understanding due to the lack of publicly available code for except Nezha. Our reproduction may not fully capture specific implementations detailed in the original works. To mitigate this potential threat, we have carefully reviewed and followed the original papers associated with the baselines. (2) External threat: The external threat is the generalizability of our experimental results. We only evaluate our method on the Nezha dataset, which is a simulated dataset with a limited number of injected anomalies. This limitation can be alleviated by Nezha's injection of typical resource and code anomalies that are common in industrial systems. Besides, a single service request in Nezha includes over 600 events, comparable to industrial microservice systems where a request involving dozens of service calls typically generates hundreds of log events. Thus, the amount of events is comparable.

8 CONCLUSION

In this work, we identified three significant limitations of existing RCA methods: (1) reliance on single-modality data, which misses complex interrelationships, resulting in suboptimal solutions; (2) service-level RCA lacks sufficient detail, and metric-level RCA is slow due to the complexity of causal analysis; and (3) separation of detection and localization stages, along with reliance on inaccurate techniques, hampers accurate RCA. To address these challenges, we introduced MRCA, a fine-grained and highly efficient RCA approach designed for microservices utilizing multi-modal data. MRCA integrates features from traces and logs into VAE to generate an initial anomaly ranking. This ranking guides the expansion of a causal graph, with reinforcement learning dynamically terminating the expansion process, ultimately achieving detailed metric-level root cause identification. Our extensive experiments on the OnlineBoutique and TrainTicket platforms demonstrated MRCA's effectiveness, showing high accuracy in both anomaly detection and root cause analysis, significantly outperforming state-of-the-art methods. Ablation experiments further confirmed the contributions of different data sources, highlighting MRCA's high generalization and stability in large-scale microservice systems. Overall, MRCA provides a robust solution for rapid and accurate RCA, addressing critical limitations of current methodologies and enhancing the reliability and efficiency of microservice systems.

ACKNOWLEDGMENTS

We thank all reviewers for their insightful comments. This paper was supported by the National Natural Science Foundation of China (No. 62102340), Guangdong Basic and Applied Basic Research Foundation (No. 2024A1515010145), and Shenzhen Research Institute of Big Data Innovation Fund (No. SIF20240009).

REFERENCES

[1] Crispin Almodovar, Fariza Sabrina, Sarvnaz Karimi, and Salahuddin Azad. 2024. LogFiT: Log anomaly detection using fine-tuned language models. *IEEE Transactions on Network and Service Management* (2024).

[2] Anunay Amar and Peter C Rigby. 2019. Mining historical test logs to predict bugs and localize faults in the test logs. In *IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 140–151.

[3] Stefan Axelsson. 2000. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)* 3, 3 (2000), 186–205.

[4] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. 2016. Microservices architecture enables devops: Migration to a cloud-native architecture. *IEEE Software* 33, 3 (2016), 42–52.

[5] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2018. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 2 (2018), 423–443.

[6] Pengfei Chen, Yong Qi, Pengfei Zheng, and Di Hou. 2014. Causeinfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. 1887–1895.

[7] Zhuangbin Chen, Yu Kang, Liqun Li, Xu Zhang, Hongyu Zhang, Hui Xu, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, et al. 2020. Towards intelligent incident management: why we need it and how we make it. In *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1487–1497.

[8] Paolo Di Francesco, Ivano Malavolta, and Patricia Lago. 2017. Research on architecting microservices: Trends, focus, and potential for industrial adoption. In *IEEE International Conference on Software Architecture (ICSA)*. 21–30.

[9] Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Xiaomin Wu, Meng Zhang, Qingjun Chen, Xin Gao, Xuedong Gao, et al. 2023. TraceDiag: Adaptive, Interpretable, and Efficient Root Cause Analysis on Large-Scale Microservice Systems. In *The 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1762–1773.

[10] FudanSELab. 2023. TrainTicket. https://github.com/FudanSELab/train-ticket Accessed: April 25, 2024.

[11] Yu Gan, Mingyu Liang, Sundar Dev, David Lo, and Christina Delimitrou. 2021. Sage: practical and scalable ML-driven performance debugging in microservices. In *The 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 135–151.

[12] Sukhpal Singh Gill and Rajkumar Buyya. 2018. Failure management for reliable cloud computing: a taxonomy, model, and future directions. *Computing in Science & Engineering* 22, 3 (2018), 52–63.

[13] GoogleCloudPlatform. 2023. OnlineBoutique. https://github.com/GoogleCloudPlatform/microservices-demo. Accessed: April 25, 2024.

[14] Clive WJ Granger. 1969. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society* (1969), 424–438.

[15] Rajeev Gupta, K Hima Prasad, Laura Luan, Daniela Rosu, and Chris Ward. 2009. Multi-dimensional knowledge integration for efficient incident management in a services cloud. In *IEEE International Conference on Services Computing*. 57–64.

[16] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. 2017. Drain: An online log parsing approach with fixed depth tree. In *IEEE International Conference on Web Services (ICWS)*. 33–40.

[17] Ronald A Howard. 1960. Dynamic programming and markov processes. (1960).

[18] Shusen Jing, Songyang Zhang, and Zhi Ding. 2024. Reinforcement Learning for Robust Header Compression (ROHC) under Model Uncertainty. *IEEE Transactions on Machine Learning in Communications and Networking* (2024).

[19] Srikanth Kandula, Ratul Mahajan, Patrick Verkaik, Sharad Agarwal, Jitendra Padhye, and Paramvir Bahl. 2009. Detailed diagnosis in enterprise networks. In *The ACM SIGCOMM Conference on Data Communication*. 243–254.

[20] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[21] Iman Kohyarnejadfard, Daniel Aloise, Seyed Vahid Azhari, and Michel R Dagenais. 2022. Anomaly detection in microservice environments using distributed tracing data analysis and NLP. *Journal of Cloud Computing* 11, 1 (2022), 25.

[22] Cheryl Lee, Tianyi Yang, Zhuangbin Chen, Yuxin Su, and Michael R Lyu. 2023. Eadro: An end-to-end troubleshooting framework for microservices on multi-source data. In *IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 1750–1762.

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

[24] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[25] JinJin Lin, Pengfei Chen, and Zibin Zheng. 2018. Microscope: Pinpoint performance issues with causal graphs in micro-service environments. In *International Conference on Service-Oriented Computing*. 3–20.

[26] Dewei Liu, Chuan He, Xin Peng, Fan Lin, Chenxi Zhang, Shengfang Gong, Ziang Li, Jiayu Ou, and Zheshun Wu. 2021. Microhecl: High-efficient root cause localization in large-scale microservice systems. In *IEEE/ACM 43rd International*

Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). 338–347.

[27] Ping Liu, Haowen Xu, Qianyu Ouyang, Rui Jiao, Zhekang Chen, Shenglin Zhang, Jiahai Yang, Linlin Mo, Jice Zeng, Wenman Xue, et al. 2020. Unsupervised detection of microservice trace anomalies through service-level deep bayesian networks. In *IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. 48–58.

[28] Chang Lou, Peng Huang, and Scott Smith. 2020. Understanding, detecting and localizing partial failures in large system software. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 559–574.

[29] Shutian Luo, Huanle Xu, Chengzhi Lu, Kejiang Ye, Guoyao Xu, Liping Zhang, Yu Ding, Jian He, and Chengzhong Xu. 2021. Characterizing microservice dependency and performance: Alibaba trace analysis. In *ACM Symposium on Cloud Computing*. 412–426.

[30] Meng Ma, Weilan Lin, Disheng Pan, and Ping Wang. 2019. Ms-rank: Multi-metric and self-adaptive root cause diagnosis for microservice applications. In *IEEE International Conference on Web Services (ICWS)*. 60–67.

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

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

[33] Meike Nauta, Doina Bucur, and Christin Seifert. 2019. Causal discovery with attention-based convolutional neural networks. *Machine Learning and Knowledge Extraction* 1, 1 (2019), 19.

[34] Claus Pahl and Pooyan Jamshidi. 2016. Microservices: A Systematic Mapping Study. *CLOSER (1)* (2016), 137–146.

[35] Luan Pham, Huong Ha, and Hongyu Zhang. 2024. BARO: Robust Root Cause Analysis for Microservices via Multivariate Bayesian Online Change Point Detection. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 2214–2237.

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

[37] Yuta Shimono, Masataka Hakamada, and Mamoru Mabuchi. 2024. NPEX: Never give up protein exploration with deep reinforcement learning. *Journal of Molecular Graphics and Modelling* (2024), 108802.

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

[39] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. 2022. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *arXiv preprint arXiv:2201.07284* (2022).

[40] Mario Villamizar, Oscar Garcés, Harold Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas, and Santiago Gil. 2015. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *10th Computing Colombian Conference (10ccc)*. 583–590.

[41] Sheetal Waghchaware and Radhika Joshi. 2024. Machine learning and deep learning models for human activity recognition in security and surveillance: a review. *Knowledge and Information Systems* (2024), 1–32.

[42] Ping Wang, Jingmin Xu, Meng Ma, Weilan Lin, Disheng Pan, Yuan Wang, and Pengfei Chen. 2018. Cloudranger: Root cause identification for cloud native systems. In *The 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 492–502.

[43] Li Wu, Jasmin Bogatinovski, Sasho Nedelkoski, Johan Tordsson, and Odej Kao. 2020. Performance diagnosis in cloud microservices using deep learning. In *International Conference on Service-Oriented Computing*. 85–96.

[44] Li Wu, Johan Tordsson, Erik Elmroth, and Odej Kao. 2020. Microrca: Root cause localization of performance issues in microservices. In *IEEE/IFIP Network Operations and Management Symposium*. 1–9.

[45] Guangba Yu, Pengfei Chen, Hongyang Chen, Zijie Guan, Zicheng Huang, Linxiao Jing, Tianjun Weng, Xinmeng Sun, and Xiaoyun Li. 2021. Microrank: End-to-end latency issue localization with extended spectrum analysis in microservice environments. In *Proceedings of the Web Conference 2021*. 3087–3098.

[46] Guangba Yu, Pengfei Chen, Yufeng Li, Hongyang Chen, Xiaoyun Li, and Zibin Zheng. 2023. Nezha: Interpretable Fine-Grained Root Causes Analysis for Microservices on Multi-modal Observability Data. In *The 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 553–565.

[47] Guangba Yu, Zicheng Huang, and Pengfei Chen. 2023. TraceRank: Abnormal service localization with dis-aggregated end-to-end tracing data in cloud native systems. *Journal of Software: Evolution and Process* 35, 10 (2023), e2413.

[48] Chenxi Zhang, Xin Peng, Chaofeng Sha, Ke Zhang, Zhenqing Fu, Xiya Wu, Qingwei Lin, and Dongmei Zhang. 2022. Deeptralog: Trace-log combined microservice anomaly detection through graph-based deep learning. In *The 44th International Conference on Software Engineering*. 623–634.

[49] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. 2019. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *AAAI Conference on Artificial Intelligence*, Vol. 33. 1409–1416.

[50] Ruici Zhang, Shoulong Xu, Rongjie Yu, and Jiqing Yu. 2024. Enhancing multi-scenario applicability of freeway variable speed limit control strategies using continual learning. *Accident Analysis & Prevention* 204 (2024), 107645.

[51] Chenyu Zhao, Minghua Ma, Zhenyu Zhong, Shenglin Zhang, Zhiyuan Tan, Xiao Xiong, LuLu Yu, Jiayi Feng, Yongqian Sun, Yuzhi Zhang, et al. 2023. Robust multimodal failure detection for microservice systems. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5639–5649.

[52] Zhijun Zhao, Chen Xu, and Bo Li. 2021. A LSTM-based anomaly detection model for log analysis. *Journal of Signal Processing Systems* 93, 7 (2021), 745–751.

[53] Lecheng Zheng, Zhengzhang Chen, Jingrui He, and Haifeng Chen. 2024. Multi-modal Causal Structure Learning and Root Cause Analysis. *arXiv preprint arXiv:2402.02357* (2024).

[54] Xiang Zhou, Xin Peng, Tao Xie, Jun Sun, Chao Ji, Wenhai Li, and Dan Ding. 2018. Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study. *IEEE Transactions on Software Engineering* 47, 2 (2018), 243–260.

[55] Ziyuan Zhou, Guanjun Liu, and Ying Tang. 2024. Multiagent Reinforcement Learning: Methods, Trustworthiness, Applications in Intelligent Vehicles, and Challenges. *IEEE Transactions on Intelligent Vehicles* (2024).

[56] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. 2019. Causal discovery with reinforcement learning. *arXiv preprint arXiv:1906.04477* (2019).

[57] Zhouruixing Zhu, Cheryl Lee, Xiaoying Tang, and Pinjia He. 2024. HeMiRCA: Fine-Grained Root Cause Analysis for Microservices with Heterogeneous Data Sources. *ACM Transactions on Software Engineering and Methodology* (2024).