ReconRCA: Root Cause Analysis in Microservices with Incomplete Metrics

Zekun Zhang[1], Jian Wang[1*], Bing Li[1], Liuxiaoxiao Zhang[2]
[1]School of Computer Science, Wuhan University, Wuhan, China
[2]R&D Center, TaiKang Insurance Group, Wuhan, China
Email: {zekunzhang, jianwang, bingli}@whu.edu.cn, zhanglxx@taikanglife.com

***Abstract*—With the widespread adoption of microservice architectures, system complexity has increased significantly, making fault root cause localization a critical issue in system operations. Under resource constraints or load pressure, the monitoring metrics that reflect system status often exhibit incompleteness, negatively impacting operational accuracy and system stability. To address this challenge, we propose ReconRCA, which consists of an offline reconstruction stage and an online localization stage. During reconstruction, ReconRCA leverages historical data, along with spatio-temporal models and attention mechanisms, to reconstruct missing metrics. In the online stage, it captures both intra and inter-metric correlations of microservices to achieve precise root cause localization. Experiments show that ReconRCA outperforms existing methods with both incomplete and complete data, achieving average top-1 hit rates at 33.28% and 39.74%, respectively.**

***Index Terms*—Microservice, Root cause analysis, Monitoring metrics, Incompleteness**

I. Introduction

Microservices architecture has become the mainstream choice for modern distributed systems and is widely used in application fields, including finance, education, and entertainment [1]. Its primary advantage lies in dividing applications into a series of small, independent services that can be deployed, scaled, and maintained independently, offering high flexibility and scalability [2]. Compared to traditional monolithic architectures, microservices reduce code coupling through modularization, enhancing both development efficiency and operational agility. As a result, microservice architectures excel in responding to complex and dynamic business requirements and have become the first choice for many organizations.

However, as the scale and complexity of microservice systems grow, operation and troubleshooting become increasingly challenging, especially in root cause localization tasks. Traditional monitoring and log analysis tools often struggle to cope with the large-scale and dynamically changing microservice systems [3][4]. The highly distributed nature of microservices means that an anomaly in any component can trigger a cascade of issues, impacting the overall health of the system [5]. In microservice systems, the health state is commonly assessed using key performance indicators (KPIs), such as CPU utilization, memory usage, and service latency. The sidecar module in Istio [6] is commonly adopted to support the observability of service mesh and maintenance. However, in resource-constrained or high-load scenarios, interruptions or performance degradation in sidecar processes can lead to incomplete monitoring data, particularly for the metric data. This incompleteness exacerbates root cause analysis (RCA) and hampers accurate fault localization.

*Corresponding author

By analyzing four commonly used datasets for anomaly diagnosis (GAIA[1], Social Network[2], TopoMAD[3], and ISP[4]), we observed two prevalent forms of incompleteness: chunk-gap and star-missing points. As shown in Figure 1, the metric data collected from two different microservice instances in the Social Network dataset exhibit missing segments (highlighted in red), demonstrating the severity of this issue. Such data incompleteness can significantly hinder subsequent analysis and root cause localization. To further investigate the impact of incompleteness on RCA, we randomly masked portions of the metric data in the ISP dataset and applied a classic PC-based algorithm [42] for root cause localization. The causal graphs and RCA results are shown in Figure 2, where the black nodes in the graphs represent identified root causes (the darker the color, the higher the anomaly degree), and the $HR@k$ metric in the table measures the hit rate of root causes within the candidate list (higher $HR@k$ values indicate better localization performance). Due to the incompleteness of metrics, the fault propagation path generated by the PC algorithm became ambiguous, leading to a degraded RCA outcome. As the missing rate increases, the HR@k metric values gradually decrease.

Our study aims to enhance operational efficiency and diagnostic accuracy in the presence of incomplete metric data, ultimately improving system reliability and observability in microservice environments. It is important to note that while missing metrics can be considered anomalies, our primary task is diagnosing failures within the microservice system rather than determining the causes of data incompleteness. To address this challenge, we present ReconRCA (Incompleteness-**Recon**struction based **R**oot **C**ause **A**nalysis for Microservice Systems), a novel approach designed to improve the accuracy and robustness of root cause localization in microservice systems.

ReconRCA restores data integrity by reconstructing missing metrics based on historical patterns. It fills in current missing values to enable effective fault analysis. After data reconstruction, ReconRCA employs a dual-attention mechanism: self-attention within individual metrics and mutual attention across different metrics—to enhance feature extraction. This dual-attention structure enables the model to uncover internal correlations and cross-metric dependencies, providing stronger support for precise root cause localization. The main contributions of this paper are as follows:

1) We propose a data reconstruction framework for handling incomplete metric data in microservice systems, offering a foundational step before root cause localization. By

[1] https://github.com/CloudWise-OpenSource/GAIA-DataSet
[2] https://github.com/delimitrou/DeathStarBench/tree/master/socialNetwork
[3] https://github.com/QAZASDEDC/TopoMAD
[4] https://github.com/deepflowio/deepflow

2836-3868/25/$31.00 ©2025 IEEE
DOI 10.1109/ICWS67624.2025.00024

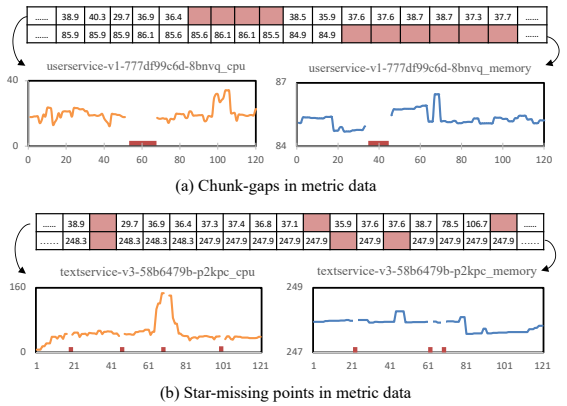
Authorized licensed use limited to: Wuhan University. Downloaded on August 30,2026 at 03:57:52 UTC from IEEE Xplore. Restrictions apply.

2025 IEEE International Conference on Web Services (ICWS) | 979-8-3315-5563-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICWS67624.2025.00024

effectively reconstructing missing data, our framework improves the accuracy and reliability of downstream analysis.

2) We propose an online RCA model that integrates intra-node self-attention and inter-node mutual attention during metric feature learning. This dual-attention mechanism effectively captures internal sequence patterns and inter-feature relationships, enabling precise root cause localization.

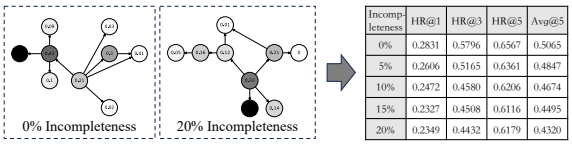
3) We evaluate our method on three real-world datasets. Our approach not only outperforms existing methods when data is complete but also demonstrates superior performance under incomplete metric data conditions, confirming the robustness and effectiveness of our proposed solution.



(a) Chunk-gaps in metric data

(b) Star-missing points in metric data

Fig. 1. An example of incomplete metric data in microservice systems



Incompleteness	HR@1	HR@3	HR@5	Avg@5
0%	0.2831	0.5796	0.6567	0.5065
5%	0.2606	0.5165	0.6361	0.4847
10%	0.2472	0.4580	0.6206	0.4674
15%	0.2327	0.4508	0.6116	0.4495
20%	0.2349	0.4432	0.6179	0.4320

Fig. 2. Causal graphs and localization results with incomplete metrics. The incompleteness indicates the proportion of metric data that is missing.

The rest of the paper is organized as follows: Section II gives the related work of root cause localization in microservice systems. Section III elaborates on the technical details of reconRCA. Section IV validates the performance of reconRCA through experiments, and finally, Section V provides the summary and future work.

II. Related Work

Based on the observable data (logs, metrics, and traces) in microservice systems, existing works on root cause localization can be categorized into unimodal-based and multimodal-based methods.

A. Unimodal-based root cause localization

Log-based methods: Microservice systems generate extensive log data during operation, providing key insights for system maintenance. [9]- [11]. Li et al. [9] proposed SwissLog, a robust tool for detecting and localizing anomalies in interleaved, unstructured logs of microservices. SwissLog identifies log sequence anomalies by constructing an ID relationship graph to organize log messages across distributed components and detects sequence anomalies using an attention-based bidirectional long short-term memory (Bi-LSTM) network. It further localizes the root cause through a heuristic search algorithm. Cinque et al. [10] developed a performance monitoring tool that analyzes HTTP request-response logs and categorizes faults based on log patterns. Jia et al. [11] achieved anomaly detection and fault identification by building graph models from historical event logs. Logs from different services may lack consistent correlation identifiers, making it challenging to reconstruct execution flows. As a result, real-time root cause analysis (RCA) requires more advanced techniques that integrate multiple observability signals beyond logs alone.

Metric-based methods: In addition to logs, researchers use metric data to build RCA models [12]-[21]. Song et al. [12] proposed Corellative-GNN, which integrates multi-head self-attention and an autoregressive mechanism. This method utilizes two parallel graph neural networks to capture temporal and spatial interdependencies, leveraging GRU and autoregressive models to enhance localization accuracy and robustness. Chen et al. [14] introduced MFRL-CA, a root cause localization method based on correlation analysis, aimed at reducing the time required for fault detection and root cause localization. This approach constructs a Microservice Fault Propagation Graph (MFPG) by analyzing correlations between observable data and historical faults. It then infers the root cause using a novel anomaly scoring mechanism and a random walk algorithm. Later, Chen et al. [16] extended this work with MFPG-FC, an improved method that models fault propagation and impact scopes based on fault correlations. Zhang et al. [17] proposed PUAD, a method leveraging PU learning (Positive-Unlabeled learning) for precise KPI anomaly detection with minimal labeled data. This active learning method prioritizes samples that are more likely to belong to the positive class in each iteration, thereby reducing the likelihood of misclassification and improving model accuracy. Xin et al. [21] proposed CausalRCA, a real-time root cause localization framework using gradient-based causal structure learning. CausalRCA generates weighted causal graphs across metrics, pinpointing specific metrics and services where anomalies originate. While metric-based methods provide valuable insights, their effectiveness is often constrained by data quality and availability. In real-world environments, metric data may meet incompleteness due to factors such as system failures, monitoring overhead, and inconsistent sampling rates. This incompleteness can lead to unreliable RCA results, as missing values may disrupt dependency modeling and anomaly propagation analysis.

Trace-based methods: In distributed applications, end-to-end tracing tools can accurately record the call chains between services. Many trace-based methods [22]-[28] have been proposed. For instance, Yu et al. [22] proposed MicroRank, an unsupervised learning algorithm for root cause analysis. MicroRank first identifies anomalous traces and then applies a PageRank-based scoring mechanism to evaluate the contribution of each trace segment to system behavior. By analyzing both anomalous and normal traces at a spectral level, MicroRank assigns a weighted score to each trace and ranks potential root causes based on this scoring. Li et al. [26] introduced TraceRCA, which is based on the insight that microservice instances exhibiting more anomalous traces and fewer normal traces are more likely to be the root cause. The

Authorized licensed use limited to: Wuhan University. Downloaded on August 30,2026 at 03:57:52 UTC from IEEE Xplore. Restrictions apply.

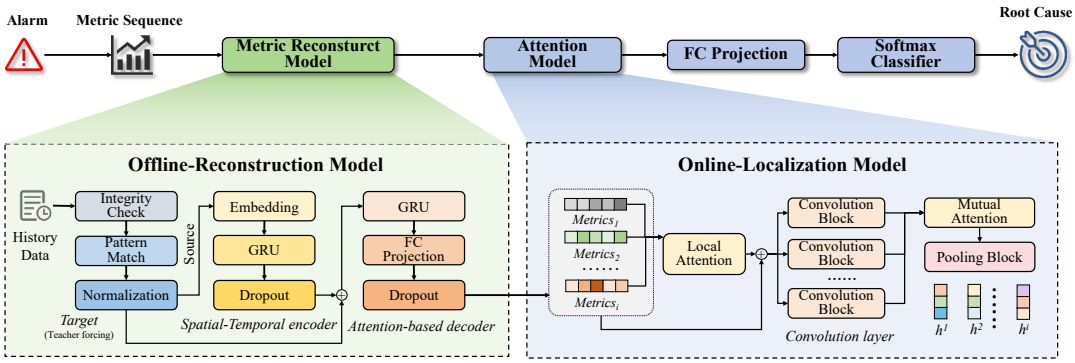


Fig. 3. Framework of ReconRCA

framework of TraceRCA includes three main steps: anomaly detection, candidate set mining, and root cause ranking. Lun et al. [28] proposed a trace disparity-based method that dynamically collects traces between microservices. These traces are used to construct a call tree that represents the application's execution paths. By computing the edit distance between normal and abnormal call trees, the method identifies structural differences indicative of potential failures. While trace data provides fine-grained root cause analysis by capturing the flow of requests, it often overlooks internal state changes within microservices, such as thread pool saturation and memory leaks.

B. Multimodal-based root cause localization

In recent years, several studies have explored leveraging multimodal observable data for root cause localization in microservice systems [29]-[37]. Wang et al. [29] proposed Groot, which models logs, traces, and metrics as events. It constructs a real-time causal graph, where each node represents an attributed event. Lee et al. [30] developed Eadro, an end-to-end model that leverages direct correlations across data modalities and inter-service dependencies. Eadro employs the Hawkes process and CNNs (Convolutional Neural Networks) to extract features from different data types. It then constructs system dependency graphs from traces and uses multi-task learning to share intermediate variables across tasks. It then applies a graph network to perform root cause localization. Li et al. [32] proposed PSqueeze, which leverages the generalized ripple effect (GRE) attributes of multimodal data to improve localization efficiency. By clustering attribute combinations and decomposing the problem into simpler sub-problems (i.e., a single root cause), PSqueeze reduces the search space and speeds up localization. Yu et al. [36] presented an interpretable fine-grained root cause localization method, Nezha, which identifies root causes at the code region and resource type levels. Nezha unifies heterogeneous multimodal data into event representations and extracts event patterns by constructing and mining event graphs. The core idea of Nezha is to compare the event patterns of the failure-free phase with those of the failure phase, thereby providing an interpretable approach for fault localization. Zhang et al. [37] proposed DiagFusion, a graph network-based root cause localization method. It uses abnormal event templates to standardize text extraction across data types and applies word embeddings for feature extraction. It combines system metadata and trace information to create a system dependency graph and employs a graph neural network to localize the root cause of microservice failures.

In practical systems, data collection tools often fail to capture all metric data due to resource constraints or environmental changes. The incomplete data can hinder the model's ability to interpret system behavior, leading to inaccurate assessments and reduced precision in root cause localization. To address this challenge, our approach enhances RCA reliability in microservice systems by reconstructing missing values using an offline model. Our research currently focuses on metric data, and the technical details of ReconRCA are presented in the following section.

III. METHOD

ReconRCA comprises two core components: an offline reconstruction model and an online localization model. The offline reconstruction model, based on an encoder-decoder architecture, is trained using historical metric data to recover missing values accurately. Upon receiving real-time data, ReconRCA first checks for completeness. If any values are missing, the reconstruction model fills in the gaps before passing it to the online localization model. The localization model then leverages self-attention to capture intra-node features and mutual attention to model inter-node dependencies, ultimately generating feature representations to support root cause localization. Figure 3 illustrates the overall architecture, with further details provided below.

A. Offline reconstruction model

The offline reconstruction model adopts a Seq2Seq architecture to recover missing metric data and achieve full reconstruction. The Seq2Seq model consists of an encoder and a decoder. The encoder employs a GRU layer to capture temporal dependencies from input sequences and produces a hidden state that serves as the initial input to the decoder. The decoder then reconstructs metric values sequentially. To improve training efficiency and reduce error propagation, we apply the *teacher forcing* technique [7], which replaces predicted values with actual values probabilistically during training, thereby mitigating gradient vanishing issues and speeding up convergence.

Spatial-Temporal encoder. Firstly, we define the dimension of input data $\mathbf{X}$ as $[B, T, F, N]$, where B is the data batch size, T is the number of time steps, F denotes the number of nodes in the microservice system, and N represents

Authorized licensed use limited to: Wuhan University. Downloaded on August 30,2026 at 03:57:52 UTC from IEEE Xplore. Restrictions apply.

the number of metrics monitored per node. Initially, dynamic temporal sequences and spatial relationships are used to extract both temporal and spatial features from the incomplete metric data. Specifically, temporal features capture the time series dynamics, while spatial dependencies reflect the interactions between nodes within the service network.

To capture temporal dependencies in the metric data, we apply convolution along the time dimension. We define the convolution kernel size as (k_t, F), where k_t is the size of the time dimension and F represents the number of features. This configuration enables each time step in the data to leverage information from $(k-1)$ neighboring time steps, including adjacent ones. To ensure a consistent length of time series, zero padding is applied along the time dimension. After configuration, the input data is reshaped as $\mathbf{X}_{\text{time}}$ ($[B, F, T, N]$), and the time-dimension convolution can be expressed as:

$$\mathbf{X}'_{\text{time}}(b,h,t,n) = \text{ReLU}\left(\sum_{f=0}^{F-1}\sum_{m=0}^{k_i-1} X_{time}\ (b,f,t+m,n)\cdot W_t(h,f,m) + b_t(h)\right), \quad (1)$$

where the batch index b indicates the current batch of samples being processed, the output index h corresponds to a specific 'depth' or 'channel' of the convolution layer, the time step t refers to the position in the time dimension after convolution, n denotes the index of the node currently processed, m is the local index of the convolution kernel within the time dimension, W_t represents the weight parameter of the convolution kernel and b_t is the bias term for the convolution operation along the time dimension.

To capture temporal dependencies in the metrics, we set the convolution kernel size to (k_n, N), where k_n is the size of the convolution kernel and N represents the number of features. This operation enables the model to capture interactions between nodes by incorporating information from each node along with its $(n-1)$ neighboring nodes. Zero padding is also applied in the spatial dimension to maintain the original node structure. In the same way, the convolution across the spatial dimension can be expressed as:

$$\mathbf{X}'_{\text{node}}(b,h,f,t) = \text{ReLU}\left(\sum_{n=0}^{N-1}\sum_{m-0}^{k_n-1} X_{node}\ (b,n,f+m,t)\cdot W_n(h,n,m) + b_n(h)\right), \quad (2)$$

where W_t denotes the weight parameter of the convolution kernel in the spatial dimension, and b_n denotes the bias term for the spatial convolution.

In this case, convolutional operations are applied to both the temporal and spatial dimensions of the input data $\mathbf{X}$, resulting in two sets of learned features: $\mathbf{X}'_{\text{time}}$ and $\mathbf{X}'_{\text{node}}$. These results are then combined as follows:

$$\mathbf{X}_{\text{combined}} = \text{concat}(\mathbf{X}'_{\text{time}}, \mathbf{X}'_{\text{node}}, dim = 1), \quad (3)$$

where the $concat$ operation denotes the concatenation of two tensors, resulting in $\mathbf{X}_{\text{combined}}$ with shape $[B, (H_t + H_n), T, N]$.

To better capture the spatial-temporal features, we apply 1D convolution [41] to $\mathbf{X}_{\text{combined}}$. First, a reshaping operation is performed, which merges the time step T and the number of nodes N into a new dimension. As a result, $\mathbf{X}_{\text{combined}}$ is reshaped to $[B, H_t + H_n, T \times N]$. Assuming the 1D convolution has a kernel size of k and a stride s, each slide covers k consecutive elements. The 1D convolution process can then be represented as:

$$\mathbf{X}_{\text{seq}}[b, h', t'] = \text{ReLU}\left(\sum_{h=0}^{H_f+H_n-1}\sum_{m=0}^{k-1} \mathbf{X}_{\text{combined}}[b, h, s\cdot t' + m]\cdot W[h', h, m] + b[h']\right). \quad (4)$$

Following the 1D convolution, which extracts local temporal and spatial features, a sequential model is necessary to capture long-range dependencies and temporal dynamics across the entire sequence. The sequential model can improve the performance of tasks that require long-term sequence prediction. Here, we choose GRU (Gated Recurrent Unit) as the backbone, for each time step t, the hidden state updating formula of the GRU is:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t, \quad (5)$$

$$z_t = \sigma\left(W_z\left[h_{t-1}, \mathbf{X}_{\text{seq},t}\right] + b_z\right), \quad (6)$$

$$r_t = \sigma\left(W_r\left[h_{t-1}, \mathbf{X}_{seq,t}\right] + b_r\right), \quad (7)$$

$$\tilde{h}_t = \tanh\left(W_h\left[r_t \odot h_{t-1}, \mathbf{X}_{seq,t}\right] + b_h\right), \quad (8)$$

where $\mathbf{X}_{seq,t}$ denotes the input feature at time step t, h_{t-1} is the hidden state from the previous time step, z_t represents the update gate, r_t is the reset gate, and $\tilde{h}_t$ denotes the candidate hidden state. The operation $\odot$ denotes elementwise multiplication, while W_z, W_r, W_h are the weight matrices for the update gate, reset gate, and candidate hidden state, respectively, and b_z, b_r, b_h are the corresponding bias terms. The function σ denotes the sigmoid function, which applies activation to the gates.

As a result, the model processes and identifies distinct metric patterns for each service node, while capturing the dynamic temporal features and interdependencies across nodes. These analyses form the basis for subsequent data reconstruction, where the model uses the spatial-temporal features to restore the metric data. Specifically, the missing metric data is imputed through the decoder.

Attention-based Decoder. During the decoding phase of the offline reconstruction model, we integrate an attention mechanism, which effectively focuses on key features to enhance the accuracy and efficiency of spatio-temporal data reconstruction. The attention mechanism aligns the convolutional features across time and node dimensions with the target sequence, optimizing the relevant features of the input sequence to improve reconstruction performance. The attention module performs the following steps:

1). Calculating the energy score. The energy score is computed based on the previous hidden state h_{t-1} of the decoder and the output of the encoder O_{enc}. The calculation of the energy score is performed as follows:

$$\mathbf{E} = \tan h(W_{\text{attn}} \cdot [h_{t-1}; O_{\text{enc}}] + b_{\text{attn}}), \quad (9)$$

where W_{attn} is the weight matrix of shape $[H_{\text{dec}} + H_{\text{enc}}, H_{\text{dec}}]$ and b_{attn} is the bias term. The energy score E represents the contribution of each time step to the decoding process.

2). Computing attention weights. The attention weights are obtained by applying the softmax function to the energy scores E as follows:

$$\mathbf{A} = \text{softmax}(v \cdot \mathbf{E}), \quad (10)$$

Authorized licensed use limited to: Wuhan University. Downloaded on August 30,2026 at 03:57:52 UTC from IEEE Xplore. Restrictions apply.

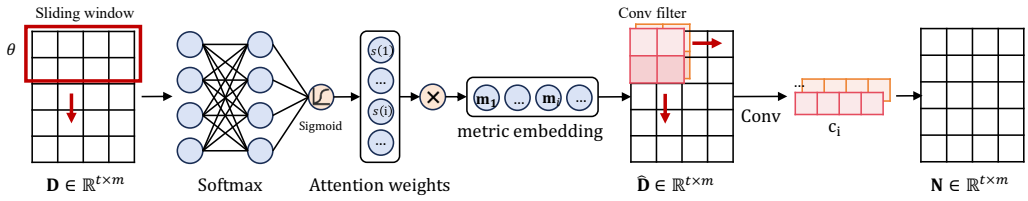


Fig. 4. Self-attention module of ReconRCA

where v are learnable parameters. This step converts E into a probability distribution, which indicates the relative importance of each element in the input sequence for the current decoding step.

3). Applying attention weights. The outputs of the encoder are weighted by the attention weights **A** and then summed to form the context matrix **C**:

$$\mathbf{C}_t = \sum_{i=1}^{T} A_{ti} \cdot O_{enc,i}\,, \tag{11}$$

where A_{ti} is the i-th element in the attention weights, and $O_{\text{enc},i}$ denotes the output of the encoder at the i-th time step. The context vector C_t is derived by calculating a weighted average of all encoder outputs, with weights provided by **A**. Combining the above steps, the resulting context vector can be represented as:

$$\boldsymbol{C_t} = \sum_{t=1}^{T} softmax(tan\,h(W_{attn} \cdot [h_{dec}, O_{enc}] + b_{attn}))_t \odot O_{enc,t}\,. \tag{12}$$

It dynamically adjusts the attention weights to different parts of the input sequence, providing key information for generating the output of the current decoding step.

4). Decoding and output. At each time step, the decoder combines two inputs: the context vector from the attention module and the current embedded input. This combination updates the decoder's hidden states via the recurrent neural network (e.g., GRU). Subsequently, the decoder generates the output for the current step, typically represented as the probability distribution of the next symbol in the sequence:

$$\text{output}_t, h_t = \text{RNN}\big(\text{Concat}\big(\text{input}_t, C_t\big), h_{t-1}\big), \tag{13}$$

where $input_t$ denotes the input of the current time step, C_t is the context vector, and h_{t-1} represents the state of the previous time step.

After processing all time steps in the sequence, a series of outputs are generated, each corresponding to the prediction for a specific time step. Then the output, with shape $[B, N \times H_{\text{dec}}]$, is reshaped to restore the original multidimensional structure. This operation aggregates the predictions from each time step to form the complete sequence prediction tensor, ensuring that temporal and spatial dependencies are preserved.

B. Online attention model

When anomalies are detected, the online monitoring system sends metric data to an offline reconstruction module. This module assesses data integrity, performs reconstruction, and supports subsequent online RCA. In this subsection, we provide technical details of the online model, which applies the dual attention mechanism. First, an intra-node attention layer processes critical data segments before applying convolution. Then, an inter-node mutual attention layer captures interactions between nodes. This combined approach enables thorough data analysis, which consists of the following modules.

Self attention. As shown in Figure 4, to better capture the importance of metric series in a microservice node, we employ an attention-based sliding window to learn the data weights at each timestamp. Specifically, we generate a metric vector $\mathbf{D} \in \mathbb{R}^{t\times m} = [\ldots, d_{i-1}, d_i, d_{i+1}, \ldots]$, where t represents the length of the metric sequence and m denotes the embedding dimension. The sliding window, with a size of θ, is applied to each sequence, enabling the calculation of attention weights for the sequence as follows:

$$w_{h,i} = \left[d_{i+\frac{-\theta+1}{2}}, d_{i+\frac{-\theta+3}{2}}, \cdots, d_i, \cdots, d_{i+\frac{\theta-3}{2}}, d_{i+\frac{\theta-1}{2}}\right], \tag{14}$$

$$s(i) = \delta\big(w_{h,i} * W_h + b_h\big), \tag{15}$$

where δ denotes the sigmoid activation function, and $s(i)$ represents the attention weight of metric data at the i-th timestamp, quantifying the data importance within the entire sequence. Based on these attention weights, the embedding at the i-th timestamp can be further computed as follows:

$$\hat{\mathbf{d}}_i = s(i) * \mathbf{d}_i, \tag{16}$$

where $s(i)$ and $\mathbf{m}_i$ denote the attention weights and the metric embedding vectors at the i-th timestamp, respectively. Using the above embeddings, a new metric vector matrix $\widehat{\mathbf{D}}$ is constructed, incorporating the importance of each timestamp and providing a richer and more refined data representation for subsequent analyses:

$$\widehat{\mathbf{D}} = \left[\hat{d}_0, \hat{d}_1 \cdots, \hat{d}_i, \cdots, \hat{d}_{t-2}, \hat{d}_{t-1}\right]. \tag{17}$$

The model then extracts the semantic information of $\widehat{\mathbf{D}}$ by convolution. Specifically, each convolutional filter is configured with a sliding window of size θ, designed to capture and refine local contextual features:

$$\mathbf{c}_i^j = W_c^j * \widehat{\mathbf{D}}[:, i:(i+\theta-1)], \tag{18}$$

where $*$ denotes the convolution, and W_c^j represents the weight vector of the j-th filter. The expression $\widehat{\mathbf{D}}[:, i:(i+\theta-1)]$ refers to a slice of $\widehat{\mathbf{D}}$, starting at position i and spanning θ consecutive elements. To ensure each timestamp is centered within the convolutional window, we append $\theta-1$ zero vectors to the end of $\widehat{\mathbf{D}}$. Since a single convolutional filter with shared weights captures only one type of contextual feature, we employ multiple filters with distinct weights to extract diverse contextual representations for each timestamp's metric data. The contextual feature c_i for the i-th timestamp is computed as:

$$c_i = \left[c_i^1, c_i^2, \cdots, c_i^i, \cdots, c_i^f\right], \tag{19}$$

Authorized licensed use limited to: Wuhan University. Downloaded on August 30,2026 at 03:57:52 UTC from IEEE Xplore. Restrictions apply.

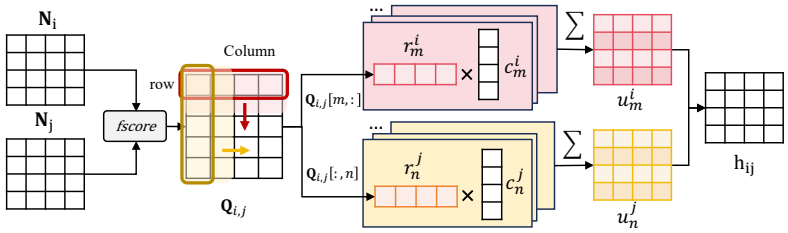


Fig. 5. Mutual attention module of ReconRCA

where c_i^f is the contextual feature generated by the f-th convolutional filter. Thus, the contextual feature matrix of each node is formulated as follows:

$$\mathbf{N}_k = [c_1^k, c_2^k, c_3^k, \cdots, c_t^k], \tag{20}$$

where c_i^k denotes the feature vector at the i-th timestamp of node k, and t represents the temporal length of the labeled sample data.

Algorithm1: Procedure of ReconRCA

Input: Incomplete data M_U and compelete data M_F
Output: Candidate set of root causes F^*

```
begin
    # Offline model for data reconstruction;
    E_output, H_hidden ← Encoder(M_U, M_F);
    # Attention-based decoder to get reconstructed data;
    D ← Decoder(E_output, H_hidden);
    # Self-attention layer for mocroservice nodes;
    for i in 0 to len(Nodes) − 1 do
        D_i ← D[i]
        D̂_i ← AttentionWeighted(D_i, θ)
        N_i ← Convolution(D̂_i, θ)
    # Mutual attention on all nodes;
    for i in 0 to len(Nodes) − 1 do
        N'_i ← N_i;
        # Mutual attention on other nodes except itself;
        for j in 0 to len(Nodes) − 1 do
            . if j ≠ i then
                N'_i ← MutualAttention(N'_i, N_j);
            end if
        end for
    end for
    # Concatenate representations of all nodes;
    H_w ← []
    for i in len(Nodes) do
        H_w ← Concate(N'_i);
    end
    F* ← Softmax(H_w)[:−5]
    return F*
end
```

Mutual attention. The above process captures the temporal dependencies among metrics within each node. Next, we introduce a mutual attention layer to analyze the inter-node correlation in depth, as illustrated in Figure 5. First, the function f_{score} is defined to compute the correlation between two different nodes:

$$f_{\text{score}} = 1/\left(1 + \left|c_i^m - c_j^n\right|\right), \tag{21}$$

where $\left|c_i^m - c_j^n\right|$ represents the absolute difference (L1 distance) between the contextual features of node i at timestamp m and node j at timestamp n. The correlation matrix is then computed as:

$$\mathbf{Q}_{i,j} = f_{\text{score}}\left(\mathbf{N}_i, \mathbf{N}_j\right), \tag{22}$$

where $\mathbf{Q}_{i,j}$ represents the correlation between the contextual feature of node i at timestamp m and node j at timestamp n.

To summarize the inter-node correlations, we define:

$$r_m^i = \sum \mathbf{Q}_{i,j}[m,:], \tag{23}$$

$$r_n^j = \sum \mathbf{Q}_{i,j}[:,n], \tag{24}$$

where r_m^i aggregates the correlation scores between node i at timestamp m and all other nodes, while r_n^j aggregates the correlation scores between node j at timestamp n.

To enhance the metric feature representation, we apply a sliding window over the time dimension, smoothing the data and generating more stable and high-level feature representations. Next, we use mutual attention to capture dependencies between nodes, enriching the understanding of inter-node relationships:

$$u_m^i = \sum_{m}^{m+\omega} r_m^i * c_m^i, \tag{25}$$

$$u_n^j = \sum_{n}^{n+\omega} r_n^j * c_n^j, \tag{26}$$

where u_m^i represents the m-th context vector of the i-th node and u_n^j represents the n-th context vector of the j-th node. This process is applied to all nodes in parallel, computing inter-attentive feature representations that capture both temporal dependencies within each node and mutual dependencies between nodes. The resulting features are aggregated and concatenated, forming the final representation denoted as $\mathbf{H}_\text{w}$.

C. Root cause analysis

The final representation of the input data is derived from the online attention-based model, which directly facilitates root cause localization by ranking the most fault-prone nodes in the system. Specifically, $\mathbf{H}_\text{w}$ is downscaled into a one-dimensional vector $\mathbf{H}_G$, and then passed through a fully connected layer. Next, the top five fault-prone nodes are identified based on their rankings:

Authorized licensed use limited to: Wuhan University. Downloaded on August 30,2026 at 03:57:52 UTC from IEEE Xplore. Restrictions apply.

$$\text{Faults} = \text{argsort}\left(\frac{\exp[(\mathbf{W}\mathbf{H}_G + b)_i]}{\sum_{j=1}^{N} \exp[(\mathbf{W}\mathbf{H}_G + b)_i]}\right)[-5:], \quad (27)$$

where N represents the number of nodes in the microservice system, $(\cdot)_i$ denotes the i-th output, $\mathbf{W}$ is the weight matrix of the fully connected layer, and b represents the bias vector. A higher ranking indicates a higher likelihood that the corresponding node is the root cause.

Algorithm 1 outlines the pseudocode of ReconRCA. In Lines 2-5, ReconRCA performs data reconstruction using an autoencoder. Then, in Lines 6-19, it applies attention mechanisms to extract node features. The final step, executed in Line 20, generates root cause predictions.

IV. Experiments

To evaluate the effectiveness of our proposed method, we conducted experiments on three publicly available datasets, aiming to answer the following research questions:

- RQ1: How does ReconRCA perform in localizing root causes under both incomplete and complete metric conditions?
- RQ2: How do key hyperparameters affect the performance of ReconRCA?
- RQ3: What is the contribution of each attention module in ReconRCA?

A. Preparation

All experiments were conducted on a workstation featuring an NVIDIA GeForce RTX 3080 GPU and an Intel i7-13700K CPU. We implemented the code using Python 3.7 and PyTorch 1.13.0.

(1) Datasets

We evaluate the performance on three datasets (A, B, and C). The first two datasets [23] were jointly released as open-source datasets through a collaboration between Tsinghua University and China Construction Bank. They contain monitoring data collected from microservice systems of major Internet Service Providers (ISPs). The third dataset [21] is derived from the SockShop benchmark. To ensure a consistent experimental setting, we applied a controlled masking process to each dataset to introduce controlled data incompleteness, ensuring fair comparisons across different methods. In the missing metrics, the number of star-missing points and chunk gaps remains consistent.

(2) Baseline methods

To fully assess the efficacy of ReconRCA, we conducted a comparative analysis with seven baseline methods. These methods include five causal inference-based methods and two state-of-the-art methods. We integrate the causal graphs generated by these causal learning methods with personalized parameters in PPR, serving as causal weights for root cause identification. The forward-filling method is used to fill the missing metrics.

- GC-based [38]: Granger Causality is a time series analysis method commonly used for constructing causal graphs. We use the χ^2 test for causality assessments.
- CAM-based [39]: The Causal Additive Model considers linear and nonlinear interactions among multiple factors. It identifies causal relationships through the SCM that fit with Gaussian errors.
- PC-Based [42]: The PC algorithm is first employed to construct an undirected graph based on the correlations among KPI metrics. In this graph, each node represents a variable, while edges indicate potential associations between variables. To identify root causes, the PageRank algorithm is applied to navigate the graph, predicting the most probable root nodes.
- GES-Based [43]: The Greedy Equivalent Search (GES) algorithm constructs a directed acyclic graph from metrics to maximize causal relationships between variables. It iteratively optimizes the graph structure and traverses from nodes to identify root causes.
- LiNGAM-Based [44]: The Linear Non-Gaussian Acyclic Model is a causal discovery method that identifies causal relationships based on statistical independence among variables. It constructs a DAG (Directed Acyclic Graph) where nodes represent variables and edges indicate causality.
- MicroDiag [40]: It employs the Direct LiNGAM to infer causality from non-Gaussian linear models. The graph is first weighted using the Pearson correlation coefficient between two metrics, followed by root cause ranking via the PageRank algorithm.
- CausalRCA [21]: CausalRCA applies unsupervised learning with DAG-GNN to learn causal structures. A GNN-based variational autoencoder (VAE) is parameterized to construct an anomaly propagation graph. In this graph, the nodes represent metrics, while the edges capture causal traces. The model then performs root cause localization based on this causal graph.

(3) Evaluation metrics

Root cause localization tasks commonly use the Hit Rate to evaluate the accuracy of predicted root causes [8][13][21]. Here, the $HR@1$, $HR@3$, $HR@5$ accuracy, and the $Avg@k$ accuracy are used as the main evaluation metrics. $HR@k$ quantifies the probability of containing real root cause instances in the first k predicted instances. Specifically, $HR@k$ is calculated as:

$$HR@k = \frac{1}{|F|}\sum_{f\in F}^{\Delta} \begin{cases} 1, & \text{if } RC_i \in RC_{pk} \\ 0, & \text{otherwise} \end{cases}, \quad (28)$$

where F is the set of samples with errors in the test set, RC_i denotes the real anomaly root cause instance for the i-th sample, RC_{pk} represents the result of the set of top-k anomaly instances predicted by the model, and k represents the hit rate value of the specific corresponding result.

In addition to the basic evaluation metrics, we introduce $Avg@5$ as an additional evaluation metric in this scenario. It explicitly compares the performance of each method on an average, which is calculated as follows:

$$Avg@5 = \frac{\sum_{i=1}^{5} HR@\text{i}}{5}. \quad (29)$$

B. Overall performance (RQ1)

Tables I and II show the results of performing the root cause localization task with complete and incomplete data,

Authorized licensed use limited to: Wuhan University. Downloaded on August 30,2026 at 03:57:52 UTC from IEEE Xplore. Restrictions apply.

TABLE I. PERFORMANCES WITH COMPLETE-METRICS

Method	Dataset A				Dataset B				Dataset C			
	HR@1	*HR@3*	*HR@5*	*Avg@5*	*HR@1*	*HR@3*	*HR@5*	*Avg@5*	*HR@1*	*HR@3*	*HR@5*	*Avg@5*
GC-Based	0.2344	0.5816	0.7373	0.6230	0.0240	0.0725	0.2240	0.0808	0.1230	0.2825	0.5230	0.3265
CAM-Based	0.2149	0.4484	0.6705	0.6276	0.0127	0.0612	0.2127	0.0921	0.0776	0.2805	0.5276	0.2994
PC-Based	0.2831	0.5796	0.6567	0.4065	0.0449	0.1914	0.2765	0.1709	0.1388	0.2732	0.3625	0.2582
GES-Based	0.2918	0.5482	0.8189	0.5530	0.0637	0.1276	0.2641	0.1518	0.1098	**0.4137**	0.4529	0.3255
LiNGAM-Based	0.2476	0.7619	**0.9524**	0.6540	0.0372	0.1234	0.1975	0.1194	0.1439	0.3888	0.5277	0.3530
MicroDiag	0.2994	0.6748	0.8112	0.6528	0.1215	0.1703	0.3215	0.1780	0.1528	0.4062	0.6228	0.3538
CausalRCA	0.1808	0.5238	0.7619	0.4888	0.0812	0.1763	0.2914	0.1830	0.1327	0.2573	0.5291	0.3064
ReconRCA	**0.3012**	**0.8513**	0.9091	**0.6872**	**0.1267**	**0.2473**	**0.3978**	**0.2573**	**0.2118**	0.3974	**0.6518**	**0.4203**
Improvements	+0.18%	+8.94%	-4.33%	+3.32%	+0.52%	+5.59%	+7.63%	+7.43%	+5.90%	-0.75%	+2.90%	+6.65%

Note: *Bold text indicates the best performance among all methods, while underlined text marks the second-best. Improvements show the performance gap between the best and second-best results.*

TABLE II. PERFORMANCES WITH 20% INCOMPLETENESS

Method	Dataset A				Dataset B				Dataset C			
	HR@1	*HR@3*	*HR@5*	*Avg@5*	*HR@1*	*HR@3*	*HR@5*	*Avg@5*	*HR@1*	*HR@3*	*HR@5*	*Avg@5*
GC-Based	0.1665	0.4045	0.6102	0.4156	0.0097	0.0921	0.1741	0.1065	0.0112	0.2373	0.4240	0.2102
CAM-Based	0.1344	0.5062	0.7087	0.5069	0.0097	0.1260	0.2149	0.1149	0.0069	0.2260	0.4127	0.2132
PC-Based	0.2349	0.4432	0.6179	0.4320	0.0436	0.1276	0.2553	0.1422	0.1093	0.2341	0.3055	0.2613
GES-Based	0.2491	0.4789	0.7729	0.5003	0.0566	0.0975	0.2264	0.1268	0.1023	0.3058	0.3789	0.2633
LiNGAM-Based	0.1984	0.6763	0.8632	0.5793	0.0188	0.0749	0.1509	0.0815	0.0549	0.3047	0.4719	0.2772
MicroDiag	0.2149	0.5780	0.7348	0.4981	0.1011	0.1425	0.2069	0.1480	0.1087	0.2348	0.5215	0.2320
CausalRCA	0.1632	0.5323	0.7695	0.4883	0.0697	0.1643	0.2264	0.1528	0.0773	0.2244	0.4287	0.2435
ReconRCA	**0.2842**	**0.7942**	**0.8823**	**0.6536**	**0.1189**	**0.2331**	**0.3423**	**0.2314**	**0.1772**	**0.3328**	**0.5923**	**0.3874**
Improvements	+3.51%	+11.79%	+1.91%	+7.43%	+1.78%	+6.88%	+8.70%	+7.86%	+6.79%	+2.70%	+7.08%	+11.02%

Note: *Bold text indicates the best performance among all methods, while underlined text marks the second-best. Improvements show the performance gap between the best and second-best results.*

respectively. We choose 20% incompleteness as the main research scenario. It strikes a balance between being too optimistic (e.g., <5%) and too pessimistic (e.g., >50%), making it suitable for general-purpose evaluation and sensitivity analysis. The following observations can be obtained from the experimental results:

Superior Performance of ReconRCA with complete metrics: ReconRCA outperforms existing state-of-the-art methods across most evaluation metrics. Thanks to the dual-attention mechanism, ReconRCA effectively explores the latent features within the metrics, demonstrating its comprehensive feature extraction capability. However, due to the lack of causal inference support, ReconRCA performs slightly worse than the LiNGAM-based and GES-based methods on some aspects of datasets A and C. In microservice architectures, a single root fault often causes anomalies in multiple downstream services. Causal models can distinguish between effect services (e.g., timeout propagation) and actual cause services by modeling directed dependencies.

Robust Localization with incomplete metrics: Across all datasets, ReconRCA demonstrates remarkable resilience in root cause localization, with minimal performance degradation in the presence of incomplete data. This robustness is attributed to the Seq2Seq-based offline model, which captures spatio-temporal dependencies in the encoder and reconstructs sequences through the attention mechanisms. This architecture enables ReconRCA to restore underlying data patterns, enhancing localization accuracy even in scenarios with incomplete metrics. It is worth noting that causal analysis suffers from causal ambiguity in scenarios with partial data loss, making it more challenging to identify the root cause accurately.

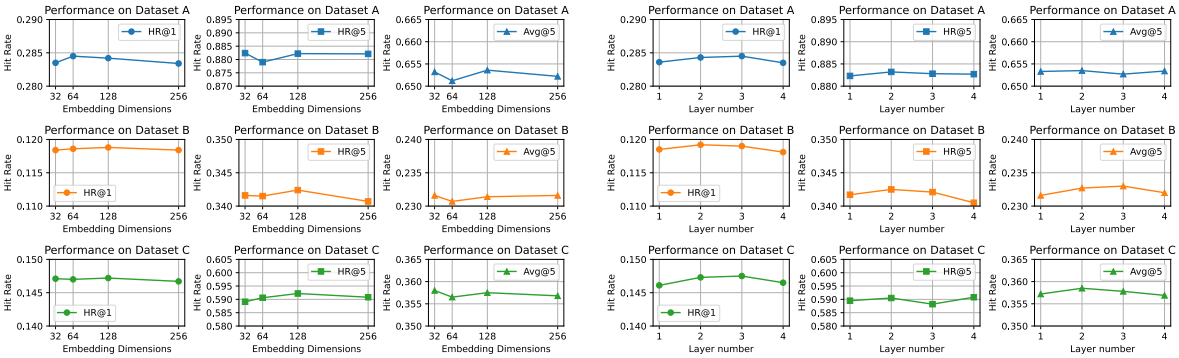
C. Parameter analysis *(RQ2)*

In this section, we delve into the effects of embedding dimension and the number of fully connected layers on localization performance.

As shown in Figure 6(a), an embedding dimension of 128 yields optimal performance. This dimensionality provides a sufficient representation to capture complex data relationships without excessive parameter growth. Higher dimensions, however, may lead to overfitting and increased computational costs, especially when data volume is limited. Thus, selecting an appropriate embedding dimension is crucial for balancing model accuracy and efficiency.

Figure 6(b) illustrates the impact of fully connected layers on performance. Fewer layers limit feature extraction, while additional layers enhance the model's expressiveness. However, excessive layers can lead to overfitting and increased computational costs. The results show that using

Authorized licensed use limited to: Wuhan University. Downloaded on August 30,2026 at 03:57:52 UTC from IEEE Xplore. Restrictions apply.



(a) *Parameter analysis on embedding diamension* (b) *Parameter analysis on layer number*

Fig. 6. Parameter analysis

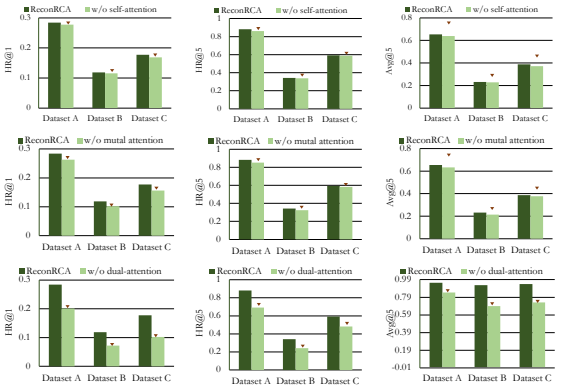


Fig. 7. Ablation study for online-attention modules

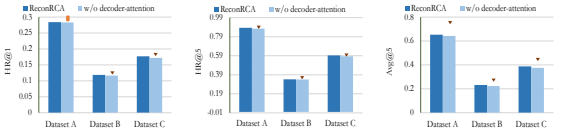


Fig. 8. Ablation study for offline-attention module

two fully connected layers effectively captures relevant features, mitigates overfitting, and strikes the best balance between accuracy and efficiency.

D. Ablation study (RQ3)

Next, we examine the contribution of each attention module, validating its impact on performance.

Figure 7 presents the ablation results for the attention mechanisms in the online localization module. Specifically we assess the performances when either intra-node self-attention, inter-node mutual attention, or both are omitted. The results demonstrate that removing either form of attention leads to a reduction in localization accuracy, with inter-node mutual attention having a particularly significant impact due to its role in capturing node interdependencies. Furthermore, intra-node self-attention is crucial for capturing fine-grained temporal dependencies within each service, ensuring that short-term variations and long-term trends are effectively integrated into the localization process.

Figure 8 shows the ablation results for the attention mechanism in the offline-reconstruction module. It reveals that removing attention from the decoder significantly degrades performance. The attention mechanism in offline reconstruction enables the model to focus on key elements of the input, facilitating the restoration of the original data and improving reconstruction through dynamic weighting.

E. Threats to validity

To assess the credibility and applicability of the research, we discuss the internal validity, external validity and construct validity for RCA tasks. (*i*) The internal validity threat pertains to the potential impact of confounding variables, which typically arises from implementation. To ensure reliability and consistency, we minimized this threat by conducting repeated experiments to confirm that results were primarily influenced by the manipulated variables. (*ii*) The threat to external validity pertains to the generalizability of the results beyond the experimental setup. We mitigated this threat by using diverse datasets, including different types of failures. In addition, the benchmark systems used in experiments are widely adopted in Devops. (*iii*) The threat to construct validity mainly lies in the selection of hyperparameters and evaluation metrics. In Section IV, we analyzed the hyperparameters in ReconRCA. Moreover, we carefully chose commonly used evaluation metrics in RCA problems to ensure that our results are comparable with previous studies.

Although ReconRCA performs relatively well, it has some limitations. It currently addresses root cause analysis in cases where the metric data is incomplete. However, in real-world environments, metrics, logs, and traces may be incomplete. It has not been optimized for these cases.

V. Conclusion and Future Work

To address the challenge of root cause analysis with incomplete metric data, we propose ReconRCA, a method that reconstructs missing metrics to improve fault localization accuracy. ReconRCA first employs an offline Seq2Seq model, which reconstructs incomplete data at a fine-grained level. Then, an online model utilizes a dual-attention mechanism—combining self-attention within nodes and mutual attention between nodes—to extract features from the reconstructed data, enabling more precise localization. Experiments on three real-world datasets show that ReconRCA outperforms existing methods on both incomplete and complete datasets,

Authorized licensed use limited to: Wuhan University. Downloaded on August 30,2026 at 03:57:52 UTC from IEEE Xplore. Restrictions apply.

demonstrating its robustness. Additionally, extensive ablation and parameter studies validate its effectiveness.

We have identified three primary directions for future improvements. First, we assume that the microservice system topology is static in this work. Implementing real-time updates is essential for more effective localization. Second, our current graph structures are primarily based on microservice dependencies. Future work may explore the use of hypergraphs, which can better represent complex topologies, such as instances co-located on the same server. Third, ReconRCA focuses solely on missing metric data, as it is the most commonly missing type during data collection. In the future, we will explore fault diagnosis in scenarios where multiple modalities are missing, potentially leveraging multi-modal learning to enhance adaptability and robustness.

VI. Acknowledgment

This work is supported by the National Key Research and Development Program of China (No. 2022YFF0902701), the National Natural Science Foundation of China (No. 62032016), and the Shenzhen Science and Technology Program under Grant CJGJZD20230724091659002.

References

[1] Nadareishvili I, Mitra R, McLarty M, et al. *Microservice Architecture: Aligning Principles, Practices, and Culture*. O'Reilly Media, Inc., 2016.

[2] Balalaie A, Heydarnoori A, Jamshidi P. *Microservices Architecture Enables Devops: Migration to a Cloud-Native Architecture*. IEEE Software, 2016, 33(3): 42-52.

[3] Xue X, Zhou D, Yu X, et al. *Computational Experiments for Complex Social Systems: Experiment Design and Generative Explanation*. IEEE/CAA Journal of Automatica Sinica. 2024, 11(4):1022-1038.

[4] Xue X, Yu X, Zhou D, et al. *Computational Experiments: Past, Present and Perspective*. Acta Automatica Sinica. 2023, 49(2):246-271.

[5] Xie S, Wang J, Li B, et al. *PBScaler: A Bottleneck-aware Autoscaling Framework for Microservice-based Applications*. IEEE Transactions on Services Computing, 2024: 604-616.

[6] *Istio 1.24 Release Announcement*, Sidecar Istio. Available: https://istio.io/latest/docs/reference/config/networking/sidecar/. Accessed: Nov. 13, 2024.

[7] LeCun Y, Bengio Y, Hinton G, *Deep Learning*, Nature, vol. 521, no. 7553, 2015, pp. 436–444.

[8] Zhang Z, Li B, Wang J, et al., *AAMR: Automated Anomalous Microservice Ranking in Cloud-Native Environment,* in *Proc. Int. Conf. Software Eng. and Knowledge Eng. (SEKE)*, 2021, pp. 86–91.

[9] Li X, Chen P, Jing L, et al. *SwissLog: Robust Anomaly Detection and Localization for Interleaved Unstructured Logs*. IEEE Transactions on Dependable and Secure Computing, 2023, 20(4): 2762-2780.

[10] Cinque M, Della Corte R, Pecchia A. *Microservices Monitoring With Event Logs and Black Box Execution Tracing*. IEEE Transactions on Services Computing, 2019, PP(99):1-1.

[11] Jia T, Chen P, Yang L, et al. *An Approach For Anomaly Diagnosis Based On Hybrid Graph Model With Logs For Distributed Services*. IEEE International Conference on Web Services (ICWS). IEEE, 2017: 25-32.

[12] Song Y, Xin R, Chen P, et al. *Identifying Performance Anomalies In Fluctuating Cloud Environments: A Robust Correlative-GNN-Based Explainable Approach*. Future Generation Computer Systems, 2023, 145: 77-86.

[13] Ma M, Lin W, Pan D, et al. *ServiceRank: Root Cause Identification of Anomaly in Large-Scale Microservice Architectures*. IEEE Transactions on Dependable and Secure Computing, 2022, 19(5): 3087-3100.

[14] Chen Y, Chen N, Xu W, et al. *MFRL-CA: Microservice Fault Root Cause Location based on Correlation Analysis*. 2021 8th International Conference on Dependable Systems and Their Applications (DSA). 2021: 90-101.

[15] Liu D, He C, Peng X, et al. *MicroHECL: High-Efficient Root Cause Localization in Large-Scale Microservice Systems*. 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). 2021: 338-347.

[16] Chen Y, Xu D, Chen N, et al. *FRL-MFPG: Propagation-Aware Fault Root Cause Location for Microservice Intelligent Operation and Maintenance*. Information and Software Technology, 2023, 153 : 107083.

[17] Zhang S, Zhao C, Sui Y, et al. *Robust KPI Anomaly Detection for Large-Scale Software Services with Partial Labels*. 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE). 2021: 103-114.

[18] Pan Y, Ma M, Jiang X, et al. *Faster, Deeper, Easier: Crowdsourcing Diagnosis of Microservice Kernel Failure from User Space*. Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2021: 646-657.

[19] Su Y, Zhao Y, Sun M, et al. *Detecting Outlier Machine Instances Through Gaussian Mixture Variational Autoencoder with One Dimensional CNN*. IEEE Transactions on Computers, 2022, 71(4): 892-905.

[20] Li Z, Zhao N, Li M, et al. *Actionable And Interpretable Fault Localization for Recurring Failures in Online Service Systems*. Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2022: 996-1008.

[21] Xin R, Chen P, Zhao Z. *Causalrca: Causal Inference Based Precise Fine-Grained Root Cause Localization for Microservice Applications*. Journal of Systems and Software, 2023, 203.

[22] Yu G, Chen P, Chen H, et al. *MicroRank: End-to-End Latency Issue Localization with Extended Spectrum Analysis in Microservice Environments*. Proceedings of the Web Conference 2021. 2021: 3087-3098.

[23] Shen J, Zhang H, Xiang Y, et al., *Network-Centric Distributed Tracing with DeepFlow: Troubleshooting Your Microservices in Zero Code*. Proceedings of the ACM SIGCOMM 2023 Conference. 2023: 420-437.

[24] Brandón Á, Solé M, Huélamo A, et al. *Graph-Based Root Cause Analysis for Service-Oriented and Microservice Architectures*. Journal of Systems and Software, 2020, 159.

[25] Zhou X, Peng X, Xie T, et al. *Latent Error Prediction and Fault Localization for Microservice Applications by Learning from System Trace Logs*. Proceedings of the 2019 27th Proc. ACM Joint Eur. Softw. Eng. Conf. and Symp. Found. Softw. Eng. (ESEC/FSE). 2019: 683-694.

[26] Li Z, Chen J, Jiao R, et al. *Practical Root Cause Localization for Microservice Systems via Trace Analysis*; proceedings of the 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), F 25-28 June 2021, 1-10.

[27] Behera A, Panigrahi C R, Behera S, et al. *trACE - Anomaly Correlation Engine for Tracing the Root Cause on a Cloud based Microservice Architecture*. Computación y Sistemas (CyS), 2023, 27(3).

[28] Meng L, Ji F, Sun Y, et al. *Detecting anomalies in microservices with execution trace comparison*. Future Generation Computer Systems (FGCS), 2021, 116: 291-301.

[29] Wang H, Wu Z, Jiang H, et al. *Groot: An Event-graph-based Approach for Root Cause Analysis in Industrial Settings*. 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2021: 419-429.

[30] Lee C, Yang T, Chen Z, et al. *Eadro: An End-to-End Troubleshooting Framework for Microservices on Multi-source Data*. 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). 2023: 1750-1762.

[31] Chen Y, Yan M, Yang D, et al. *Deep Attentive Anomaly Detection for Microservice Systems with Multimodal Time-Series Data*. 2022 IEEE International Conference on Web Services (ICWS). 2022: 373-378.

Authorized licensed use limited to: Wuhan University. Downloaded on August 30,2026 at 03:57:52 UTC from IEEE Xplore. Restrictions apply.

[32] Li Z, Chen J, Chen Y, et al. *Generic And Robust Root Cause Localization for Multi-Dimensional Data in Online Service Systems*. Journal of Systems and Software (JSS), 2023, 203.

[33] Wang L, Zhang C, Ding R, et al. *Root Cause Analysis for Microservice Systems via Hierarchical Reinforcement Learning from Human Feedback*. 2023: 5116-5125.

[34] Zhao C, Ma M, Zhong Z, et al. *Robust Multimodal Failure Detection for Microservice Systems*. Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2023: 5639-5649.

[35] Gu S, Rong G, Ren T, et al. *TrinityRCL: Multi-Granular and Code-Level Root Cause Localization Using Multiple Types of Telemetry Data in Microservice Systems*. IEEE Transactions on Software Engineering, 2023, 49(5): 3071-3088.

[36] G. Yu, P. Chen, Y. Li, et al., *Nezha: Interpretable Fine-Grained Root Causes Analysis for Microservices on Multi-Modal Observability Data*, in Proc. ACM Joint Eur. Softw. Eng. Conf. and Symp. Found. Softw. Eng. (ESEC/FSE), USA. 2023:553 - 565

[37] Zhang S, Jin P, Lin Z, et al. *Robust Failure Diagnosis of Microservice System through Multimodal Data*. IEEE Transactions on Services Computing, 2023: 3851-3864.

[38] Clive WJ Granger. *Investigating Causal Relations By Econometric Models And Cross-Spectral Methods*, In: *Econometrica: journal of the Econometric Society* , vol. 37, no. 3, 1969: 424–438.

[39] P Bühlmann P, Peters J, Ernest J, *CAM: Causal Additive Models, High-Dimensional Order Search and Penalized Regression*. In: *The Annals of Statistics,* vol. 42, 2013.

[40] Wu L, Tordsson J, Bogatinovski J, et al., *Microdiag: Fine-Grained Performance Diagnosis for Microservice Systems*. In: *2021 IEEE/ACM International Workshop on Cloud Intelligence*. IEEE. 2021, pp. 31–36.

[41] Kiranyaz S, Avci O, Abdeljaber O, et al., *1D Convolutional Neural Networks and Applications: A Survey*. Mechanical systems and signal processing, 2021, 151: 107398.

[42] P Spirtes P, Glymour C, Scheines R, *Prediction, and Search, Second Edition*. MIT Press, 2000.

[43] Chickering DM, *Optimal Structure Identification with Greedy Search*. J Mach Learn Res, 2002, 3: 507-554.

[44] Shimizu S, Hoyer P O, Hyvärinen A, et al. *A Linear Non-Gaussian Acyclic Model for Causal Discovery*. J Mach Learn Res, 2006, 7: 2003-2030.

Authorized licensed use limited to: Wuhan University. Downloaded on August 30,2026 at 03:57:52 UTC from IEEE Xplore. Restrictions apply.