TAMO:Fine-Grained Root Cause Analysis via Tool-Assisted LLM Agent with Multi-Modality Observation Data in Cloud-Native Systems

Xiao Zhang, Qi Wang, Mingyi Li, Yuan Yuan, Mengbai Xiao, Fuzhen Zhuang, and Dongxiao Yu

***Abstract*—Implementing large language models (LLMs)-driven root cause analysis (RCA) in cloud-native systems has become a key topic of modern software operations and maintenance. However, existing LLM-based approaches face three key challenges: multi-modality input constraint, context window limitation, and dynamic dependence graph. To address these issues, we propose a tool-assisted LLM agent with multi-modality observation data for fine-grained RCA, namely TAMO, including multi-modality alignment tool, root cause localization tool, and fault types classification tool. In detail, TAMO unifies multi-modal observation data into time-aligned representations for cross-modal feature consistency. Based on the unified representations, TAMO then invokes its specialized root cause localization tool and fault types classification tool for further identifying root cause and fault type underlying system context. This approach overcomes the limitations of LLMs in processing real-time raw observational data and dynamic service dependencies, guiding the model to generate repair strategies that align with system context through structured prompt design. Experiments on two benchmark datasets demonstrate that TAMO outperforms state-of-the-art (SOTA) approaches with comparable performance.**

***Index Terms*—Root cause analysis, Tool-Assisted LLM agent, Cloud-native systems, Multimodal data, Diffusion.**

I. Introduction

In recent years, microservice architecture in cloud-native systems has become foundational elements in modern enterprise software development, driving the rapid improvement of distributed systems [25] [30] [37] [45]. Microservices decompose traditional monolithic applications into independent distributed services, while cloud native technologies leverage containerization and container orchestration to enhance the capabilities for flexible resource allocation and autoscaling [21] [33]. In detail, cloud-native systems are composed of numerous independent microservices and containerized entities (e.g., pod, node) [52], which inevitably brings inter-service dependencies and complex communication patterns. Therefore, the failure of any entity (e.g., network corruption) can trigger cascading effects that damage overall system stability, potentially leading to significant economic losses [2] [38].

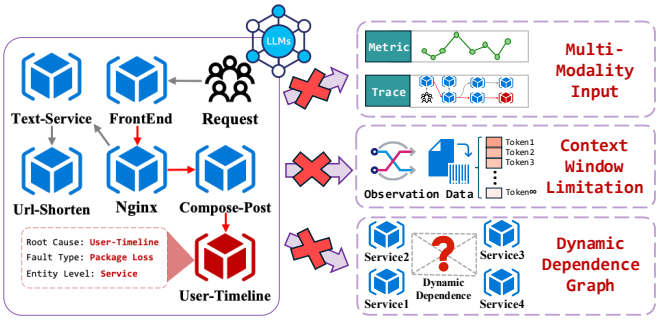


Fig. 1. A failure case in a cloud-native system demonstrates three major challenges when using LLM directly for root cause analysis.

Root cause analysis (RCA) aims to quickly correlate features of multiple related entities, localize root cause, identify corresponding fault type, and provide reasonable solutions, thus realizing an automatic fault response mechanism and minimizing human intervention [5] [48]. Therefore, how to perform accurate and efficient root cause analysis in cloud-native systems has become an important research topic recently [14] [51]. On one hand, traditional RCA methods [29] [31] typically rely on rule-based anomaly detection or statistical correlation analysis (e.g. causal graphs) [24] to localize faults. These methods struggle to jointly analyze heterogeneous data modalities with unstructured logs and metric data, often leading to undetected faluts. On the other hand, deep learning (DL) based RCA methods attempt to incorporate multimodal feature fusion and dependency-sensitive graph neural networks (GNNs) [23] [50] to achieve accurate fault localization. For instance, MULAN [51] infers causal structures from heterogeneous observation data for service-level diagnosis, while PDiagnose [18] treats multimodal data independently and aggregates decisions via voting to pinpoint root causes.

This work was supported in part by the National Natural Science Foundation of China under Grant 62202273, 62176014, in part by the Joint Key Funds of National Natural Science Foundation of China under Grant U24B20149, in part by Major Basic Research Program of Shandong Provincial Natural Science Foundation under Grant ZR2025ZD18, in part by China Postdoctoral Science Foundation under Grant 2024M761806, and in part by the Fundamental Research Funds for the Central Universities. (*Corresponding author: Dongxiao Yu.*)

Xiao Zhang, Qi Wang, Mingyi Li, Mengbai Xiao, and Dongxiao Yu are with the School of Computer Science and Technology, Shandong University, Qingdao 266237, China (Email: xiaozhang@sdu.edu.cn; wsq5457@mail.sdu.edu.cn; limee@mail.sdu.edu.cn; xiaomb@sdu.edu.cn; dxyu@sdu.edu.cn).

Yuan Yuan is with the School of Software & Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR), Shandong University, Jinan, 250000, China (Email: yyuan@sdu.edu.cn).

Fuzhen Zhuang is with the Institute of Artificial Intelligence, Beihang University, Beijing, 100191, China, and State Key Laboratory of Complex & Critical Software Environment (SKLCCSE), Beihang University, Beijing 100191, China. (Email: zhuangfuzhen@buaa.edu.cn).

The abovementioned methods mainly focus on specific tasks such as fault localization in complex systems, ignoring fine-grained RCA with fault type identification, contextual anomaly interpretation, and cross-level dependency reasoning. Recently, large language models (LLMs) have demonstrated revolutionary capabilities in automated RCA, opening new avenues for AIOps [9] [12]. LLMs can combine RCA with current system context to generate reasonable fault diagnosis reports, emerging with RCACopilot [6]. Recent works such as RCAgent [41] and mABC [49] develop LLM-driven automated RCA frameworks that enable accurate fault localization and provide reasonable repair suggestions, thus further advancing the field of automated operations and maintenance. RCAgent proposes a tool-augmented autonomous agent framework that leverages log and code data with an internally deployed LLM to perform root cause analysis, while mABC mitigates circular dependency issue in microservice RCA through blockchain-inspired multi-agent voting within a structured workflow.

However, existing LLM-based methods still face some tricky challenges when applied to fine-grained RCA in cloud-native systems involving multi-level entities and heterogeneous multi-modality data:

- **Multi-Modality Input**: Existing LLM-driven methods such as COCA [26], RCAgent [41], only support log data or code data as input, ignoring abundant context information such as metric data and trace data. However, directly feeding multimodal data into LLMs often hinders accurate RCA due to inherent modality misalignment, for instance, the semantic gap between time-series metrics and textual logs. Therefore, *how to map different modalities into unified semantic space to assist the LLM in accurate RCA* is challenging.
- **Context Window Limitation**: Limited by fixed-length context windows, the LLM is unable to effectively process the massive real-time metric data generated by continuous monitoring across kinds of microservices, nodes or pods in cloud-native systems. The real-time metric data typically reflect system states but does not explicitly indicate anomalies or fault types, yet lacking aligned semantic meaning with text logs. *So how to transform raw measurements into meaningful diagnostic prompts and input them into the LLM with limited context length?*
- **Dynamic Dependence Graph**: The dynamic service call chains from the trace data in the system inevitably bring evolving dependencies. However, exsiting LLM-based methods [40] struggle with fine-grained RCA over dynamic dependence graphs, making it difficult to capture implicit fault propagation path. *So how to assist the LLM agent in modeling fault propagation paths based on dynamic dependence graph to improve root cause localization and fault types classification?*

To address these chanllenges, we propose a tool-assisted LLM agent framework based on kinds of domain-specific tools. This framework decouples the LLM from the raw multi-modality observation data by utilizing specialized tools: multi-modality alignment tool $\mathcal{T}_1$, root cause localization tool $\mathcal{T}_2$, and fault types classification tool $\mathcal{T}_3$. These specialized model tools are used to perceive the contextual environment, and their results form structured prompts that are fed back to the RCA expert agent for further analysis, generating accurate root cause analysis results and reasonable maintenance recommendations. In detail, multi-modality alignment tool $\mathcal{T}_1$ is based on a dual-branch diffusion architecture, which unifies multi-modality observation data, such as log data, trace data and metric data, into temporally consistent representations. To address dynamic dependencies challenge, the root cause localization tool ($\mathcal{T}_2$) jointly utilizes the extracted representations from $\mathcal{T}_1$ and dependence graphs extracted from trace data to model fault propagation and identify the root cause via frequency-domain attention-driven temporal causal analysis. Subsequently, the fault types classification tool ($\mathcal{T}_3$) further utilizes the casual graph from $\mathcal{T}_2$ to identify the anomalous patterns.

The main contributions are summarized as follows:

- We propose a domain-specific tools assisted LLM agent framework for fine-grained RCA, including multi-modality alignment tool, root cause localization tool, and fault types classification tool to empower the LLM for real-time system diagnosis.
- In detail, we present a dual-branch collaborative diffusion tool for multi-modality alignment that captures potentially consistent patterns across modalities and provides unified representations for downstream tasks through condition-oriented collaborative reconstruction. Otherwise, the root cause localization tool is based on frequency-domain attention mechanisms, which identifies causal dependency patterns in frequency-domain features while filtering out low-frequency noise, achieving precise root cause localization.
- Experiments on two benchmark datasets demonstrate that our proposed TAMO outperforms state-of-the-art (SOTA) approaches, achieving average *Acc@1* improvement of 4.8% for root cause localization and *MiPr* improvement of 10.8% for fault types classification respectively.

II. Related Work

In modern cloud-native systems, with the widespread adoption of microservice architectures and dynamic infrastructures, it has become particularly important to ensure high system availability and rapid fault recovery, making root cause analysis (RCA) a critical part of operations and maintenance. As a result, various RCA methods have been developed to guarantee system reliability, including traditional RCA methods, multi-modal RCA methods, and LLM-based RCA methods.

Traditional RCA methods: Traditional RCA methods are mainly based on statistical correlation analysis techniques, such as anomaly detection based on predefined rules and root cause localization strategies based on causal discovery [7] [43]. While these approaches perform well in static environments, they have difficulties in adapting to the dynamic service dependencies that characterize cloud native environments. With the rapid advancement of deep learning, neural network-based methods for real-time system data analysis continue to emerge. DeepLog [11] models log features via long short-term

memory networks. GDN [10] and GTA [8] construct graph neural networks using metrics data to capture fault propagation paths. Sage [13] leverages causal Bayesian networks with trace data for root cause localization. However, these approaches predominantly focus on unimodal data, failing to exploit complementary inter-modal relationships.

Multi-modal RCA methods: The multimodal-based approach significantly reduces the undetected fault rate by comprehensively considering system observation data from different modalities. MULAN [51] learns causal structures from logs and metrics for service-level localization. PDiagnose [18] analyzes different modality data as independent entities and uses a voting mechanism to identify the root cause entity. Although this method successfully utilizes data from multiple modalities, it overlooks the consistency relationships between features from different modalities. Eadro [23] performs gated fusion of features extracted from different modalities and uses a graph neural network to fuse embedded features at the service level for root cause localization. HolisticRCA [14] compared to Eadro, fully considers the heterogeneous characteristics of cloud native systems and standardizes the embedding of different modality data using an assembling building blocks strategy, combining masked embeddings for overall root cause analysis. In contrast to directly using raw data for root cause localization, Nezha [46] and DiagFusion [48] convert heterogeneous multimodal data into homogeneous event representations, performing joint analysis through event graphs to achieve root cause localization. CoE [44] further supports the operational experience of site reliability engineers as input on the event graph, enhancing the understanding and interpretability of the event.

LLM-based RCA methods: Large language models are increasingly leveraged for root cause analysis due to their strong reasoning capabilities. Recent works explore various LLM applications: Adarma [36] combines LLMs with traditional ML for microservice anomaly detection and remediation using logs and metrics. RCAcopilot [6] maps multimodal observation data to predefined events and predicts root cause categories in production through an LLM-driven workflow, and agent-based approaches [35] utilize tool-assisted LLMs to actively retrieve diagnostic data like logs and metrics for localization. Despite these advancements, significant challenges persist. The limited context window of LLMs restricts their ability to process large scale of observation data, while their inherent text-based input modality makes handling time series metrics difficult, risking loss of critical temporal patterns. Furthermore, existing methods often face limitations in generalizability (e.g., reliance on predefined rules [6]),lack integration of crucial trace data for precise localization [35] [36]. These challenges highlight the need for frameworks that can effectively unify multimodal data (logs, metrics, traces) within an LLM architecture to achieve both fine-grained root cause localization and accurate classification. We compare these existing methods with our approach in Table I.

TABLE I
COMPARISON OF DIFFERENT LLM RCA MODELS BY MODALITY AND TASK. **FTC** DENOTES FAULT TYPES CLASSIFICATION; **RCL** DENOTES ROOT CAUSE LOCALIZATION

Methods	Modality			Task	
	Logs	Metrics	Traces	FTC	RCL
RCAcopilot [6]	✓	✓	✓	✓	×
Agent Work [35]	✓	✓	×	×	✓
Adarma [36]	✓	✓	×	×	×
Ours	✓	✓	✓	✓	✓

III. METHODOLOGY

To fully exploit the multi-modality observation data of cloud native systems while integrating prior domain knowledge, we propose **TAMO**, **T**ool-Assisted LLM **A**gent that Integrates **M**ulti-Modality **O**bservation Data to address root cause entity localization and fault types classification in cloud-native systems. This framework utilizes observation tools composed of a dual-branch collaborative diffusion model to effectively extract and integrate multimodal data from different entities. Additionally, task-specialized tools explicitly model the complex dependencies among entities using a causal graph, with feature propagation through the causal graph enabling accurate root cause analysis. Furthermore, to fully incorporate domain knowledge, we employ a LLM with operational expertise as an expert agent. The expert agent $\mathcal{A}$ integrates the outputs from the preceding tools $\mathcal{T}_2$-$\mathcal{T}_3$ with context knowledge of the system to offer human engineers a comprehensive fault analysis and recommended solutions. As shown in Fig.2, the proposed framework consists of three tools and one agent, $\mathcal{T}_1$–$\mathcal{T}_3$ and $\mathcal{A}$, each responsible for a specific task in the root cause analysis process:

1) Multi-modality Alignment Tool ($\mathcal{T}_1$): This tool employs a dual-branch collaborative diffusion model to integrate multimodal features. It reconstructs and fuses textual data features, such as logs, into time-series data based on conditional guidance;
2) Root Cause Localization Tool ($\mathcal{T}_2$): Utilizing the multimodal-enhanced time-series data as input, this tool models causal relationships between entities using a frequency-domain self-attention module, enabling accurate fault root cause localization;
3) Fault Types Classification Tool ($\mathcal{T}_3$): Based on the causal relationships between entities and the calling chain topology, this tool classifies the types of faults;
4) RCA Expert Agent ($\mathcal{A}$): Leveraging the expertise of a LLM, this agent synthesizes the outputs from previous tools along with system environment context to provide a comprehensive fault analysis and recommended solutions.

A. Multi-modality Alignment Tool ($\mathcal{T}_1$)

Diffusion models are a class of generative models based on Markov processes that have demonstrated exceptional performance in the generation of data in various domains, including images [19], voice [34], and time series [47]. The core idea is to gradually add noise to the data through a forward denoising (forward diffusion) process, which converts the original data distribution to an approximate Gaussian distribution. In the reverse denoising process, the model learns how to gradually

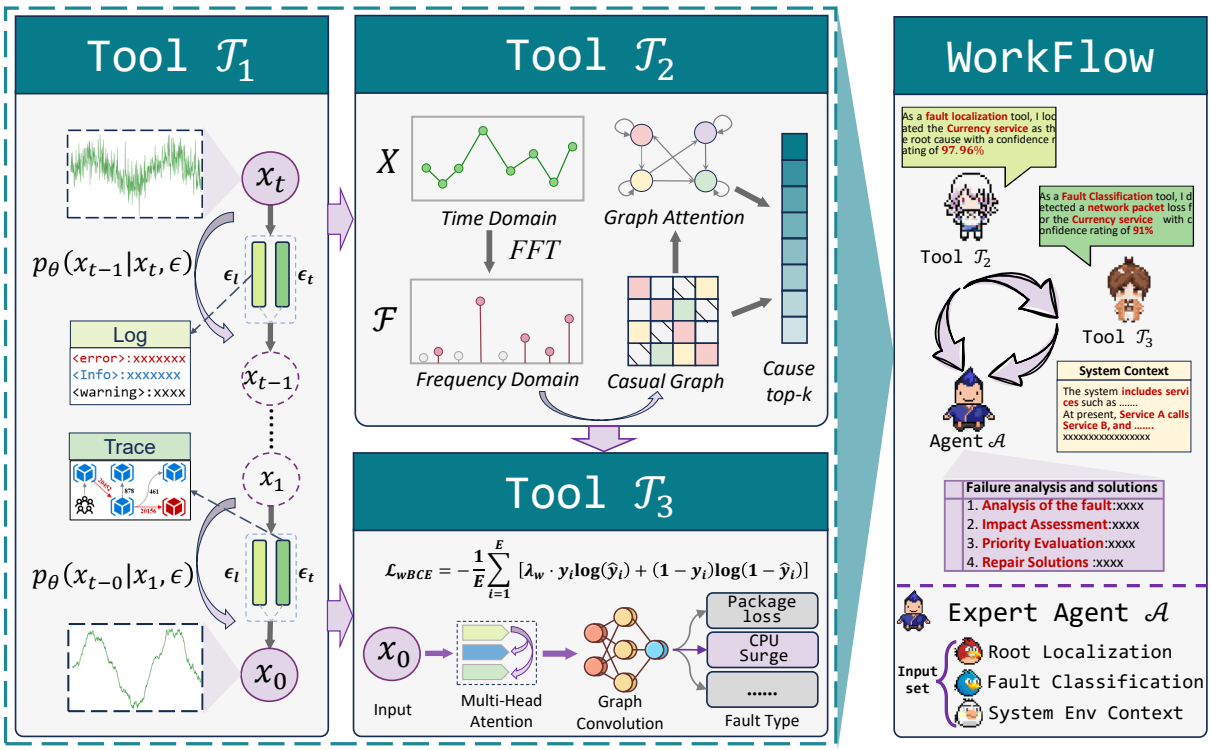


Fig. 2. The proposed TAMO framework consists of a Multi-modality Alignment Tool ($\mathcal{T}_1$), a Root Cause Localization Tool ($\mathcal{T}_2$), a Fault Types Classification Tool ($\mathcal{T}_3$), and an RCA Expert Agent ($\mathcal{A}$). In the framework, the agent $\mathcal{A}$ calls tools $\mathcal{T}_1$-$\mathcal{T}_3$ as perception tools to analyze the system contextual observation data in real time. The results of these perceptions are structured into text inputs that are fed back into the agent. By analyzing the perception results, the expert agent will provide the corresponding root cause analysis and repair suggestions.

remove the noise to generate the desired data. In this tool, we use log patterns and temporal features as control conditions for two diffusion branches. During the denoising process, the two branches collaborate to guide the generation of time-series data with multimodal features from the noise-perturbed original data. Specifically, during the forward diffusion process, the original data $x_0 \sim q(x_0) \in \mathbb{R}^{E\times m\times l}$ consists of E entities, m channels, and a time series of length l. This process can be expressed as:

$$q\left(x_1, \cdots, x_T \mid x_0\right) := \prod_{t=1}^{T} q\left(x_t \mid x_{t-1}\right). \tag{1}$$

If $\alpha_t \in (0,1)$ is a predefined noise scaling factors that determines how much information from the previous step is retained, then each step of the Eq.1 is:

$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, \left(1-\alpha_t\right)\mathbf{I}\right). \tag{2}$$

After T steps of noise addition, x_T converges to random noise $x_T \sim \mathcal{N}(0, I)$, which approximates an Gaussian distribution.

Subsequently, we perform stepwise denoising on the noisy data using a dual-branch conditional mechanism. The parameterized conditional transition distribution can be expressed as:

$$\begin{aligned} p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathcal{C}\right) &= \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t, \mathcal{C}\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t, \mathcal{C}\right)\right), \\ p_\theta\left(\mathbf{x}_{0:T}, \mathcal{C}\right) &= p\left(\mathbf{x}_T\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathcal{C}\right), \end{aligned} \tag{3}$$

where $\mathcal{C} = \{c_{log}, c_{time}\}$ represents the set of control conditions, with c_{log} and c_{time} corresponding to the log branch and the time-series branch, respectively. These conditions guide the denoising process, ensuring that the generated data have multi-modal features by utilizing both log-based patterns and temporal features.

For the generation of the log-conditioned variable c_{log}, since unstructured log representations are difficult to extract high-quality features, inspired by previous work [14] [23], we extract event patterns $\mathcal{E}$ and keywords $\mathcal{W}$ from the raw logs using Drain [16], and by calculating the TF-IDF scores, we select the highest scoring pattern-keyword pairs as the conditional variable c_{log}.

$$c_{log} = \left\{\arg\max_{e\in\mathcal{E}} \text{TF-IDF}(e)\right\} \cup \left\{\arg\max_{w\in\mathcal{W}} \text{TF-IDF}(w)\right\}. \tag{4}$$

For the generation of the time-conditioned variable c_{time}, we extract key performance indicators (KPIs) such as latency and duration from trace files. These KPIs x_{kpi} are then embedded into a latent space using a Multi-Layer Perceptron (MLP) network, enabling the model to effectively incorporate temporal characteristics into the denoising process.

$$c_{time} = \text{Embedding}\left(x_{kpi}\right) \in \mathbb{R}^{h_{time}}. \tag{5}$$

During training, we use a dual-branch network to learn the reverse diffusion process, gradually denoising the noisy data. In this process, the noise term estimated at each step is

collaboratively generated by the two branches.

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta\left(\mathbf{x}_t, t, \mathcal{C}\right)\right) + \sigma_t \mathbf{z} \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}), \tag{6}$$

in which, $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ represents the cumulative product of the noise scaling factors α_s and ϵ_θ represents the noise term predicted by the model, which is jointly generated based on the computations from the log branch ϵ_{log} and time-series branch ϵ_{time}. A hyperparameter μ is used as a mixing coefficient to balance the contributions of the two branches.

$$\epsilon_\theta\left(\mathbf{x}_t, t, \mathcal{C}\right) = \mu\epsilon_{log}\left(\mathbf{x}_t, t, \mathbf{c}_{log}\right) + (1-\mu)\,\epsilon_{time}\left(\mathbf{x}_t, t, c_{time}\right). \tag{7}$$

The original denoising diffusion model employs U-Net as the denoising backbone [17]. However, considering the characteristics of time-series data, we adopt a patch-based Transformer network [32] as the backbone for each branch to better capture the representations of time-series subsequences. The training objective of the denoising process is to minimize the KL-divergence between the distributions p and q. According to [17], the objective derivation is formulated as follows:

$$\mathcal{L}_{diff} = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(x_t, t, \mathcal{C})\|_2^2\right]. \tag{8}$$

Through this denoising process, we obtain time series data $x'_{ent} \in \mathbb{R}^{L \times d_{ent}}$ that integrates multimodal features for subsequent analysis, where ent denotes the corresponding resource entity, L denotes the length of the time window, and d_{ent} denotes the dimension of the feature associated with each entity. This generative alignment avoids the semantic misalignment inherent in feature concatenation, yielding a more coherent and temporally consistent multimodal representation for downstream analysis ($\mathcal{T}_2$ and $\mathcal{T}_3$).

B. Root Cause Localization Tool ($\mathcal{T}_2$)

Through the dual-branch diffusion model of Tool $\mathcal{T}_1$, we obtain time-series data that integrates multimodal features. However, since this process operates at the level of system resource entities, the generated time-series x'_{ent} maintains the same dimensionality as the original resource entities. There exists heterogeneity in the observable data from different resource entities in cloud-native systems, leading to inconsistencies between them, which hinders the unified learning of the model. To address this issue, we first perform data embedding on the features of different entities, mapping them into a unified dimensional space for consistency:

$$H^{ent} = \mathrm{DataEmbedding}\left(x'_{ent}\right) \in \mathbb{R}^{L \times d_{model}}, \tag{9}$$

where d_{model} represents the unified embedding dimension. After mapping all entity features into the latent space, we concatenate the output results to obtain the embedded vector $H^{emb} \in \mathbb{R}^{E \times L \times d_{model}}$.

After obtaining the unified embedding vector H^{emb}, we consider that root cause localization tasks typically require the detection and identification of short-term anomalies or disturbance signals within the system entity. These transient features may be obscured by smoothed trends in the time domain. To address this issue, we apply the Fast Fourier Transform (FFT) to convert the data from the time domain to the frequency domain. Formally, the transformation can be expressed as:

$$\mathcal{F}\left(H^{emb}\right) = \mathrm{FFT}\left(H^{emb}\right) = \sum_{i=0}^{L-1} H_i^{emb} e^{-j2\pi ki/L}, \tag{10}$$

where $e^{-j2\pi ki/N}$ is the complex exponential fourier basis function, j is the imaginary unit, H_i^{emb} represents the time-series value corresponding to the time step i, and $k = 0, 1, \ldots, N-1$ is the frequency index.

Considering the real-time demands of root cause localization, the model needs to focus on short-term fluctuations in the observed data. To achieve this, we apply a high-pass filter to the frequency-domain data, retaining the first k high-frequency components which helps to amplify the anomalous signals within the data. Formally, this can be expressed as:

$$\mathrm{Indices} = topk\left(\left|\mathcal{F}\left(H^{emb}\right)\right|, k = freq\_topk\right), \tag{11}$$

$$Mask = \begin{cases} 1, & \text{if } k \in \text{ Indices} \\ 0, & \text{otherwise} \end{cases}, k = 0, 1, \ldots, K-1, \tag{12}$$

$$\mathcal{F}_{\text{filtered}} = \mathcal{F}\left(H^{emb}\right) \odot Mask, \mathcal{F}_{\text{filtered}} \in \mathbb{R}^{E \times K \times d_{model}}, \tag{13}$$

Since the interactions between microservices in the cloud-native systems can be described by a dependency graph, in order to fully consider the relationships between entities in the system and capture potential fault propagation pathways, we use the self-attention mechanism to construct a causal dependency graph in the frequency domain. The process can be defined as:

$$A = Softmax\left(Attention(\mathcal{F}_{\text{filtered}} W^Q, \mathcal{F}_{\text{filtered}} W^K)\right) \tag{14}$$

where W^Q,W^K are the query and key matrices respectively. Subsequently, to reduce the influence of irrelevant entities and remove redundant relationships, we introduce topological constraints into the graph generation process. These constraints can be defined as:

$$\mathbf{A}_{\text{maske}} = \mathbf{A} \odot (\mathbf{I} - \mathbf{I}_E) \odot \mathrm{TopkMask}(\cdot, k = topk) \tag{15}$$

in which $\mathbf{I}_E$ is the entity diagonal matrix for avoiding self-loops and $\mathrm{TopkMask}$ preserves the top k most significant dependencies for each entity. Then, we apply graph neural network to perform root cause inference and localization. Specifically, we select the Graph Attention Network (GAT) [3] as the backbone for aggregating neighborhood feature information among system entities, in order to simulate the fault propagation process as follows:

$$\mathcal{H}_i^{s+1} = Relu\left(\alpha_{ii} W \mathcal{H}_j^s + \sum_{j \in N(i)} \alpha_{ij} W \mathcal{H}_j^s\right), \tag{16}$$

where $N(i) = \{j | A_{mask}[i, j] = 1\}$, $\mathcal{H}_i^s$ is the representation of entity i in the s-th layer and $\mathcal{H}_i^0 = \mathcal{F}_{\text{filtered}}$. $\alpha_{i,j}$ is the attention coefficient between system entity i and its related entity j, which can be calculated by the following formula:

$$\xi(i, j) = LeakyRelu\left(a^\top\left(W^s \cdot h_i^s \oplus W^s \cdot h_j^s\right)\right), \tag{17}$$

$$\alpha_{ij} = \frac{\exp(\xi(i,j))}{\sum_{p \in N_i \cup \{i\}} \exp(\xi(i,p))}, \tag{18}$$

where $\oplus$ denotes the concatenation operation, a^T represents the vector of attention parameters and W^s is the weight matrix for the s-th layer. Finally, we use a max-pooling layer to aggregate the importance of instances, thereby obtaining the root cause probability for the system entities:

$$\hat{y} = \text{MaxPool}\left(\mathbf{H}^S\right) \in \mathbb{R}^E, \tag{19}$$

Considering that multiple types of entities may experience faults simultaneously within the same time period, meaning that the root cause entity may not be unique, we use binary cross-entropy (BCE) as the loss function to optimize the model. Specifically, for each label y_i and the model predicted value $\hat{y_i}$ (where $i \in \{0, 1, ..., E-1\}$, and E represent the total number of system entities) , the loss function is defined as:

$$\mathcal{L}_{BCE} = -\frac{1}{E}\sum_{i=1}^{E}\left[y_i \log(\hat{y_i}) + (1-y_i)\log(1-\hat{y_i})\right], \tag{20}$$

C. Fault Types Classification Tool ($\mathcal{T}_3$)

After the root cause entity localization is performed by Tool $\mathcal{T}_2$, the fault diagnosis process typically requires further identification of the specific fault type of the root cause entity (e.g., hardware overload, resource leakage, or configuration errors) to guide precise remediation strategies. Although root cause localization narrows down the scope of fault analysis (i.e., determining "where" the anomaly occurs), in practical operational scenarios, different fault types often correspond to differentiated handling strategies (e.g., memory leaks require service restarts, while CPU contention necessitates resource scaling). Therefore, this tool uses the time series data x'_{ent} generated by tool $\mathcal{T}_1$ (for simplicity, denoted as z) to train a fault classification model.

To model the latent temporal dependencies in the input time-series data z, we use a Transformer model [39] as the encoder network. The output of its single-head self-attention mechanism is given by:

$$\text{Attention} = \text{softmax}\left(\frac{zW^{\hat{Q}} \cdot z(W^{\hat{K}})^\top}{\sqrt{d_k}}\right) zW^{\hat{V}}, \tag{21}$$

where $W^{\hat{Q}}$,$W^{\hat{K}}$,$W^{\hat{V}}$ are linear projection matrices. Subsequently, we concatenate the results of the multi-head attention to obtain the latent vector representation:

$$V = \text{MultiHead}\left(zW^{\hat{Q}}, zW^{\hat{K}}, zW^{\hat{V}}\right), \tag{22}$$

After capturing the internal temporal dependencies of the time-series data, we employ the GAT model to capture the spatial dependencies between the time steps. According to formulas Eq.16-Eq.18, we perform feature propagation on the latent vector representation. The representation of each entity is obtained by aggregating the weighted features of its neighboring entities:

$$Z' = \text{ELU}\left(\text{GATConv}(X, A)\right), \tag{23}$$

RCA Expert Prompt

You are an RCA expert responsible for analyzing and diagnosing failures of complex systems based on cloud-native, microservices architectures. You have access to detailed failure reports, system performance data, and historical failure patterns. You provide detailed analysis based on the following structured inputs, which include root cause information, failure classification, system architecture context, historical failure data, and other system state details. You are expected to apply advanced reasoning methods, drawing on knowledge of system architecture, historical failure modes, and industry best practices, to provide comprehensive, practical solutions to improve system reliability and minimize downtime.

[Failure root cause localization]
- Root Entities: <localization results from $\mathcal{T}_2$>
- Related Entities: <the related calling entities>

[Fault Classification]
- Fault Type: <classification results from $\mathcal{T}_3$>
- Symptoms: <detailed Symptom Description>

[System architecture background]
- Architecture: <microservice functionality>
- Context: <related logs and trace information>

[Task Objective]

Based on the information above, please analyze the root cause, impact scope, and proposed solutions for the fault, and provide the following:

1) Analysis of the root cause of the fault.
2) Analysis of the type and level of failure.
3) Repair solutions and preventive measures.

Fig. 3. Prompt templates for this RCA expert agent.

where ELU is the activation function and $A \in \mathbb{R}^{E \times E}$ is the dependency between entities. Subsequently, we use an MLP network to output the fault type for the corresponding entity. Considering the issues of multi-class overlap and class imbalance in the fault classification task, we adopt a weighted binary cross-entropy (wBCE) loss function with the introduction of class weight coefficients λ as the model's training objective:

$$\mathcal{L}_{wBCE} = -\frac{1}{E}\sum_{i=1}^{E}\left[\lambda_w \cdot y_i \log(\hat{y_i}) + (1-y_i)\log(1-\hat{y_i})\right], \tag{24}$$

in which λ_w represents the ratio of negative to positive samples within each fault type class, which is used to enhance the loss contribution of the samples. Additionally, to prevent overfitting, we include an L2 weight decay term in the loss function. The final loss function can thus be formulated as:

$$\mathcal{L} = \mathcal{L}_{wBCE} + \beta\|\mathbf{W}\|_2^2. \tag{25}$$

where β is a predefined hyperparameter and $\mathbf{W}$ is the model weights.

D. RCA Expert Agent ($\mathcal{A}$)

After the three tools $\mathcal{T}_1$-$\mathcal{T}_3$ have completed tasks such as multi-modality alignment, root cause localization, and fault types classification, the framework has achieved a relatively deep understanding of the fault. However, relying solely on these outputs makes it difficult to provide comprehensive and accurate failure analysis and remediation plans. Therefore, to further improve fault response and system recovery efficiency, we introduce the RCA expert agent $\mathcal{A}$. Using large language models, this agent synthesizes the predictions from the previous tools and incorporates the current context of cloud-native microservice architecture to offer a more comprehensive and accurate fault diagnosis and resolution strategy for site reliability engineers.

Specifically, we use the GPT-4 [1] as the expert model, which gathers the outputs from the previous tools as background knowledge, including:

- **Root Cause Localization Results** (from $\mathcal{T}_2$): Identifying the root cause location of the fault and the associated system components.
- **Fault Types Classification Results** (from $\mathcal{T}_3$): Providing the specific type of fault, such as network failure, hardware failure, configuration errors, etc.
- **System Context Information**: Includes the microservice architecture (e.g., service roles, dependencies, and call chains), deployment topology, and resource usage patterns. This context, often derived from system documentation, guides the agent in understanding operational relationships and diagnosing faults.

In practice, we prepare the prompt in the format shown in Fig.3 as input to the model, and the expected output is a detailed fault analysis report along with corresponding solution recommendations. Such outputs assist the engineers in quickly understanding the root cause, impact scope, and taking reasonable remediation actions, thus enhancing system reliability and response efficiency.

IV. Evaluation

A. Research Questions

This section answers the following research questions:

- **RQ1:** How effective is TAMO in Root Cause Localization?
- **RQ2:** How effective is TAMO in Fault Types Classification?
- **RQ3:** How well does TAMO work when ablation studies are performed on its various components?
- **RQ4:** What are the training and inference efficiency of TAMO, and how do hyperparameters affect its performance?
- **RQ5:** How well does TAMO perform in real-world case studies and fault scenarios?

B. Experiment Setup

1) Datasets: We conducted extensive experiments on two public datasets:

- **Dataset** A. A is a large-scale public dataset from the AIOps Challenge[1], which is collected through fault injection into the real-world deployed microservice system HipsterShop[2]. The dataset primarily consists of three types of data: metrics, logs, and traces. The system uses a dynamic deployment architecture, consisting of 10 services, each with 4 pods, totaling 40 pods that are dynamically deployed across 6 nodes. The dataset includes 15 types of faults in total. Among these, 9 fault types are related to services and pods in the context of Kubernetes (K8s) containers [4], while node faults include 6 types: sudden memory pressure, disk space exhaustion, disk read/write issues, CPU pressure, and slow CPU growth.
- **Dataset** B. B is a public dataset[3] collected from the broadcast-style SocialNetwork system in [23], which only contains service-level entities. It includes 21 microservices and also consists of three types of data: metrics, logs, and traces. The dataset is generated by injecting faults into the system using Chaosblade, producing data that includes three types of faults: CPU resource exhaustion, network latency, and packet loss.

2) Baselines: We compare TAMO with the most popular integrated multimodal methods and derivative methods from the time series domain. Specifically, we select two state-of-the-art integrated multimodal root cause analysis methods, Eadro [23] and HolisticRCA [14], as well as four unimodal time series analysis methods (i.e., TimesNet [42], SegRNN [28], DejaVu [27], Transformer [39]). These methods are implemented using their publicly available and reproducible open-source code, and we adjust the data format to match the required input. Furthermore, since our research task is relatively novel and includes both fault localization and classification, we incorporate the same fault classifier used in TAMO for methods that do not natively implement fault classification (such as methods [23], [39], [42], etc.) for comparison. Furthermore, we introduce LightGBM [20], a method specifically designed for fault classification tasks, as a benchmark to evaluate TAMO performance.

3) Evaluation metrics: TAMO is dedicated to locating root cause instances and identifying fault types. Based on previous research works [14] [15] [23] [48], we select different evaluation metrics to evaluate the performance of model for these two tasks. Specifically, for the root cause localization task, we use top-k accuracy (Acc@K) as the evaluation metric, which represents the probability that the true root cause entity appears within the top-k results produced by the method. Here, we set $k = \{1, 3, 5\}$. Given S as the set of faults, RC is the ground truth root cause, and $rc[k]$ are the top-k root causes generated by the model. Therefore, Acc@K can be defined as follows:

$$Acc@k = \frac{1}{|S|} \sum_{i \in S}^{S} \begin{cases} 1, & \text{if } RC_i \in rc_i[k] \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

[1]https://competition.AIOps-challenge.com/home/competition/1496398526429724760

[2]https://github.com/GoogleCloudPlatform/microservices-demo

[3]https://doi.org/10.5281/zenodo.7615393

TABLE II
EXPERIMENTAL RESULTS OF DIFFERENT APPROACHES ON ROOT CAUSE LOCALIZATION AND FAULT CLASSIFICATION. BEST RESULTS ARE BOLDED, AND SECOND-BEST RESULTS ARE UNDERLINED

Data	Approach	Root Cause Localization			Fault Types Classification					
		Acc@1	Acc@3	Acc@5	MiPr	MaPr	MiRe	MaRe	MiF1	MaF1
A_s	HolisticRCA	65.62%	76.33%	81.69%	0.6355	**0.7130**	**0.7728**	**0.8037**	**0.6974**	**0.7507**
	TimesNet	14.81%	32.10%	61.72%	-	-	-	-	-	-
	Eadro	13.58%	37.03%	45.68%	0.1887	0.0770	0.1235	0.1046	0.1493	0.0758
	SegRNN	14.81%	40.75%	60.49%	-	-	-	-	-	-
	DejaVu	62.50%	65.62%	78.13%	-	-	-	-	-	-
	LightGBM	-	-	-	0.1967	0.1536	0.2000	0.2282	0.1983	0.1707
	Transformer	16.22%	37.84%	56.76%	0.0114	0.0123	0.1892	0.1932	0.0214	0.0230
	PatchTST	14.81%	34.57%	53.09%	0.2000	0.0741	0.0247	0.0185	0.0440	0.0296
	AutoFormer	11.11%	29.63%	37.04%	-	-	-	-	-	-
	TAMO	**71.87%**	**82.14%**	**88.83%**	**0.7164**	0.6671	0.6486	0.6149	0.6809	0.6299
A_p	HolisticRCA	**65.62%**	**80.80%**	86.60%	0.5913	**0.6682**	**0.7534**	**0.7598**	0.6626	**0.7008**
	TimesNet	10.71%	22.14%	32.14%	-	-	-	-	-	-
	Eadro	9.38%	13.13%	15.63%	0.3901	0.3344	0.4665	0.4792	0.4249	0.3761
	SegRNN	18.75%	56.25%	84.38%	-	-	-	-	-	-
	DejaVu	18.75%	21.88%	28.13%	-	-	-	-	-	-
	LightGBM	-	-	-	0.6849	0.6450	0.6792	0.6808	0.6820	0.6583
	Transformer	10.71%	19.29%	24.29%	0.0106	0.0106	0.4254	0.4439	0.0207	0.0208
	PatchTST	10.00%	18.13%	23.75%	0.6357	0.5359	0.4069	0.4107	0.4962	0.4543
	AutoFormer	13.13%	25.00%	31.25%	0.2600	0.1602	0.1613	0.1660	0.1991	0.1466
	TAMO	64.28%	80.36%	**87.50%**	**0.7182**	0.6597	0.6840	0.6642	**0.7007**	0.6445
A_n	HolisticRCA	73.66%	75.89%	79.01%	0.6219	0.6437	0.7612	0.7322	0.6846	0.6717
	TimesNet	21.15%	48.07%	78.85%	-	-	-	-	-	-
	Eadro	17.18%	28.13%	42.19%	0.5426	0.6003	0.7969	0.8145	0.6456	0.6575
	SegRNN	7.50%	20.63%	23.75%	-	-	-	-	-	-
	DejaVu	81.25%	90.63%	**100.00%**	-	-	-	-	-	-
	LightGBM	-	-	-	0.5208	0.5491	0.8333	0.8178	0.6410	0.6406
	Transformer	13.46%	48.08%	78.85%	0.1415	0.1423	0.5577	0.5207	0.2257	0.2212
	PatchTST	17.19%	42.19%	79.69%	0.3521	0.3597	0.3906	0.3609	0.3704	0.3476
	AutoFormer	21.88%	59.38%	92.19%	0.5075	0.5286	0.5313	0.5195	0.5191	0.5181
	TAMO	**84.37%**	**91.96%**	97.77%	**0.8718**	**0.8869**	**0.8947**	**0.8681**	**0.8831**	**0.8706**
B	HolisticRCA	61.11%	75.00%	77.78%	0.5000	0.5936	**0.8056**	**0.8056**	0.6170	**0.6636**
	TimesNet	15.62%	25.00%	40.63%	0.4893	0.5789	0.7187	0.7398	0.5823	0.6394
	Eadro	40.62%	56.25%	71.87%	0.4167	0.5211	0.6250	0.6516	0.5000	0.5694
	SegRNN	18.75%	43.75%	62.50%	0.1496	0.2359	0.5937	0.6287	0.2389	0.3296
	DejaVu	15.62%	34.38%	50.00%	-	-	-	-	-	-
	LightGBM	-	-	-	0.7832	0.6434	0.4186	0.3860	0.4687	0.4122
	Transformer	46.88%	68.75%	84.38%	0.4464	0.5664	0.7813	0.7955	0.5682	0.6406
	PatchTST	12.50%	34.38%	53.13%	0.7200	0.7222	0.5625	0.5934	0.6316	0.6417
	AutoFormer	46.88%	62.50%	75.00%	0.5185	0.6177	0.4375	0.4705	0.4746	0.5013
	TAMO	**72.22%**	**80.55%**	**86.11%**	**0.8500**	**0.8750**	0.5313	0.5682	**0.6538**	0.6421

TABLE III
THE EXPERIMENTAL RESULTS OF THE ABLATION STUDY CONDUCTED ON OUR PROPOSED TAMO FRAMEWORK.

Models	Root Cause Localization			Fault Types Classification					
	Acc@1	Acc@3	Acc@5	MiPr	MaPr	MiRe	MaRe	MiF1	MaF1
$\text{TAMO}_{w/o\ \mathcal{T}_1}$	43.75%	56.25%	68.75%	0.5909	0.5556	0.4063	0.4453	0.4815	0.4911
$\text{TAMO}_{w/o\ branch}$	59.38%	68.75%	71.87%	0.7200	0.7350	0.5625	0.6010	0.6316	0.6167
$\text{TAMO}_{w/o\ FFT}$	53.13%	68.75%	78.13%	0.8000	0.7639	0.5000	0.5404	0.6154	0.5977
TAMO	**72.22%**	**80.55%**	**86.11%**	**0.8500**	**0.8750**	**0.5313**	**0.5682**	**0.6538**	**0.6421**

For the fault classification task, we use $Precision(Pre)$, $Recall(Rec)$, and $F1-score(F1)$ to evaluate performance. Given true positives (TP), false positives (FP), and false negatives (FN), the formulas for these metrics are as follows: $Pre = \frac{TP}{TP+FP}$,$Rec = \frac{TP}{TP+FN}$, $F1 = \frac{2 \cdot Pre \cdot Rec}{Pre+Rec}$. Considering that this task is a multi-label classification problem, we adopt both micro and macro metrics for a comprehensive evaluation. Micro metrics aggregate the performance across all labels, treating all instances equally, which is useful when the class distribution is imbalanced. In contrast, macro metrics compute the metric for each label independently and then average the results, giving equal weight to each label regardless of its frequency. Thus, the final performance is measured using micro-precision ($MiPr$), macro-precision ($MaPr$), micro-recall ($MiRe$), macro-recall ($MaRe$), micro-F1 score ($MiF1$), and macro-F1 score ($MaF1$).

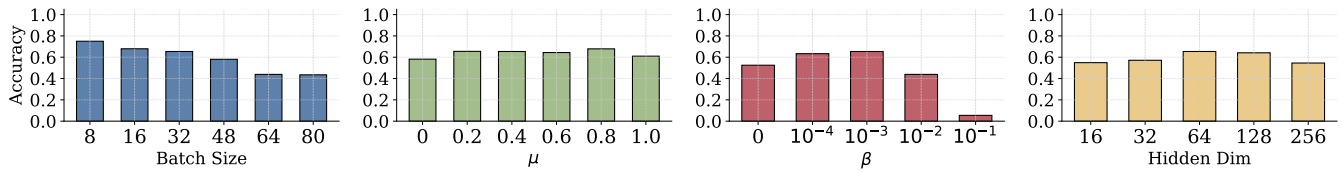


Fig. 4. Hyperparameter sensitivity analysis across batch size, μ, regularization strength (β), and hidden dimension.

4) Implementation: We implement TAMO using Python torch 2.4.0 and CUDA 12.1. The experiments are conducted on a Linux server equipped with an NVIDIA GeForce RTX 3090 GPU, using Python 3.8. The model is trained with the Adam optimizer [22], and all experiments use a fixed random seed. For hyperparameters, we set the batch size to 32, μ to 0.5, β to 0.001, $epoch$ to 200, and the initial learning rate to 0.001. Considering the requirements of TAMO for system contextual awareness and temporal consistency, data cleaning is performed on datasets A and B before training. This process remove noisy and misaligned multimodal records and realigned log and trace data to ensure temporal consistency and enhance the quality of input observation data for $\mathcal{T}_1$. Additionally, considering the pre-training needs of $\mathcal{T}_1$, an extra dataset of observation data from normal runtime operations is constructed. This dataset is used for $\mathcal{T}_1$ pre-training to enable learning of normal system patterns.

For training, we adopt a sequential and modular strategy to ensure stable learning across the tool ensemble. We first pre-train tool $\mathcal{T}_1$ using normal operational data from Datasets A and B, leveraging the diffusion framework to learn robust representations of multi-modal system observations. Once $\mathcal{T}_1$ is pre-trained, its parameters are frozen and it serves as a fixed feature extractor for subsequent stages. Tool $\mathcal{T}_2$ is then trained using the embeddings generated by $\mathcal{T}_1$ to perform root cause localization. After $\mathcal{T}_2$ is trained, both $\mathcal{T}_1$ and $\mathcal{T}_2$ are kept frozen while training $\mathcal{T}_1$-generated embeddings and the causal graphs from $\mathcal{T}_2$ as input for fault classification. Finally, all components are jointly fine-tuned for 5 epochs with a small learning rate of 0.0001 to refine the overall pipeline. This staged training ensures effective knowledge transfer and prevents instability during end-to-end optimization.

Additionally, due to the heterogeneity of system entities in Dataset A, which includes three levels: Pod, Service, and Node. The collected data features vary across these levels, making it challenging to train certain baselines uniformly. For such baselines, we partition Dataset A into subsets composed of different entity types (e.g., A_p, A_s, and A_n) and perform separate training and testing for each subset to evaluate their performance. Finally, the symbol '-' is used in the results table II to indicate cases where a method failed to achieve meaningful classification performance (e.g., accuracy close to 0) on different datasets.

C. ***RQ1:*** *Effectiveness in Root Cause Localization*

Table II presents the results for root cause localization. The experimental data shows that our proposed TAMO model achieves the best or second-best performance across all datasets. Specifically, on the A_s dataset, the score of TAMO exceeds the second-best score by at least 6%, and even in the second-best performance case, the gap with the best result is kept under 3%, clearly outperforming other baselines. Furthermore, we observe that the Eadro and Transformer models, which perform well on the B dataset, experience a significant drop in performance on heterogeneous datasets such as A_s, A_p, and A_n. This is because these two methods can only handle homogeneous entity features (e.g., in the dataset B, there is only one type of service entity). When confronted with heterogeneous entity features that need to be processed separately, their performance significantly deteriorates due to the lack of inter-entity correlation information. In contrast, both HolisticRCA and our approach model heterogeneous entities in a unified manner, maintaining strong performance even on the dataset A. Compared to HolisticRCA, we map multimodal information into time-series features and further optimize it through spatiotemporal modeling, achieving better performance in the root cause localization task.

D. ***RQ2:*** *Effectiveness in Fault Types Classification*

In the fault classification task, as shown in Table II, TAMO demonstrates significant improvments in two key metrics: Micro Precision (MiPr) and Micro F1 (MiF1). Particularly on the A_n dataset, TAMO achieves a MiPr of 0.8718 and MiF1 of 0.8831, outperforming the second-best model (HolisticRCA) by 24.99% and 19.85%, respectively, and ranks first across all evaluation metrics. With MiPr reaching 0.7164 and 0.7182 on A_s and A_p, respectively, TAMO outperforms the best baseline by 8.09% and 12.69%, significantly reducing the misclassification. Furthermore, we observe that transformer-based models achieve relatively strong performance on relatively simpler dataset B. However, their performance degrades significantly on the large-scale, complex system dataset A, indicating that metric-only transformer architectures struggle to capture the intricate, multi-modal dependencies inherent in large-scale distributed systems.

E. ***RQ3:*** *Ablation Study*

To explore the effectiveness of each component in our TAMO framework, we designed three variant models and conducted ablation experiments. The variant models are as follows:

- $\text{TAMO}_{w/o\ \mathcal{T}_1}$: To investigate the effectiveness of the diffusion method used by tool $\mathcal{T}_1$ in fusing multimodal feature, in this variant we remove the tool $\mathcal{T}_1$ and directly use the raw time series data as input to evaluate the variant performance.

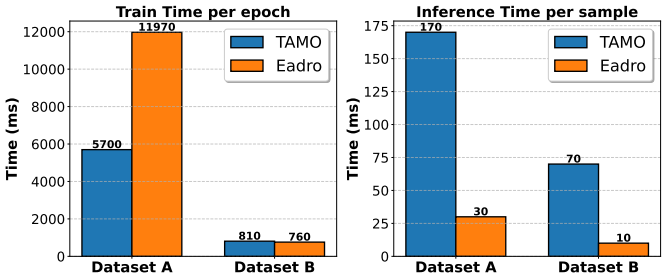


Fig. 5. Comparision of training time and inference time of TAMO with Eadro.

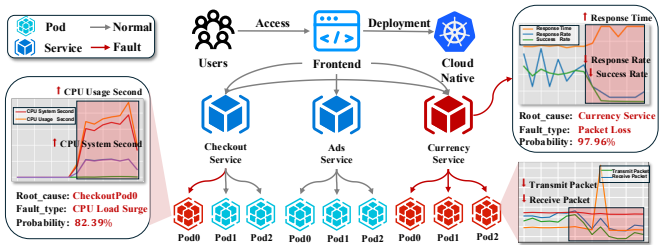


Fig. 6. In case study experiments, two types of faults were induced in microservices, with the root cause entities identified as the Currency Service and CheckoutPod0.

- $\text{TAMO}_{w/o\ branch}$: To investigate the effectiveness of the dual-branch diffusion model in tool $\mathcal{T}_1$, we remove the branch ϵ_{time}, which is responsible for maintaining temporal features, in this variant. Only the log branch ϵ_{log} is used in the diffusion reconstruction process. The reconstructed data are then used for fault classification to evaluate variant performance.
- $\text{TAMO}_{w/o\ FFT}$: To investigate the effectiveness of the frequency domain-based root cause localization method proposed for tool $\mathcal{T}_2$, we remove the Fourier process in this variant. Instead, the original time-domain data are directly used for subsequent causal graph generation and graph convolution processes. The root cause localization results are then used to evaluate variant performance.

Table III presents the results of the ablation experiments, from which it is evident that the removal of any component in TAMO leads to performance degradation. This demonstrates that each of the component within TAMO has an important effect. In particular, the variant of TAMO without tool $\mathcal{T}_1$ shows a significant decrease in performance. This is because the fact that the variant $\text{TAMO}_{w/o\ \mathcal{T}_1}$ lacks the ability of the diffusion model to fuse multi-modality information, thereby being reduced to learn with only a single modal feature. The absence of key feature information makes accurate root cause analysis challenging. Similarly, removing the time branch ϵ_{time} lead to noticeable decreases in performance for both tasks, indicating that time series features are crucial for accurate fault localization and classification. Eliminating the FFT process result in a substantial degradation in RE localization metrics, demonstrating that frequency domain analysis is essential for precise root cause localization.

F. *RQ4: Performance Study*

To evaluate computational efficiency, we measure the training and inference time of TAMO on both datasets, as shown in Figure 5, with Eadro as a representative baseline. For training, TAMO requires 0.81 seconds per epoch on Dataset B and 5.7 seconds on Dataset A, indicating favorable scalability with minimal computational overhead relative to Eadro. For inference, TAMO processes each sample in 0.17 seconds, which is slightly slower than Eadro but still meets the real-time requirement for root cause analysis.

In addition, we provide the parameter counts for each component of the TAMO framework in Table IV. The multi-modality alignment tool $\mathcal{T}_1$ accounts for the majority of parameters (9.32M), as aligning heterogeneous multimodal data into a unified temporal representation requires a large number of parameters to learn effective alignment. In contrast, the root cause localization tool $\mathcal{T}_2$ and fault types classification tool $\mathcal{T}_3$ are significantly more lightweight (1.38M and 258.32K parameters, respectively).

TABLE IV
PARAMETER COUNTS OF THE TOOL MODELS IN TAMO.

Model	Parameters
Multi-modality Alignment Tool ($\mathcal{T}_1$)	9.32M
Root Cause Localization Tool ($\mathcal{T}_2$)	1.38M
Fault types Classification Tool ($\mathcal{T}_3$)	258.32K
Total parameters	**10.96M**

Finally, we analyze sensitivity to key hyperparameters (batch size, β, μ, hidden dimension). As shown in Fig. 4, performance is highly sensitive to β, with minor changes causing sharp accuracy drops. In contrast, smaller batch sizes consistently yield better results, while μ causes significant performance degradation at its boundary values (μ = 0 or 1), indicating that it is designed to achieve critical branch balancing.

G. *RQ5: Case Study*

To validate the root cause analysis capabilities of TAMO in cloud-native ecosystems, this study conducts a case investigation using real-world failure data from HipsterShop. As shown in Fig. 6, the microservices system is deployed using a Kubernetes (K8s) architecture, with service components developed in multiple programming languages. The system entry is managed through the Frontend service, and each service node is configured with three Pod replicas. Monitoring data collection points are set at both the service and Pod levels. Experimental data shows that TAMO demonstrates precise root cause identification for the current failure scenario: the root cause localization confidence for the Currency Service reaches 97.96% (service level), while the confidence for CheckoutPod0 is 82.39% (Pod level). Notably, while the overall Checkout service is functioning correctly, only Pod0 experiences issues. In contrast, the Currency service suffers from a global failure at the service level, causing anomalies in Pods 0-2. This comparative case indicates that TAMO effectively integrates multi-entity features and topological causal relationships to achieve precise root cause diagnosis across multiple layers. In

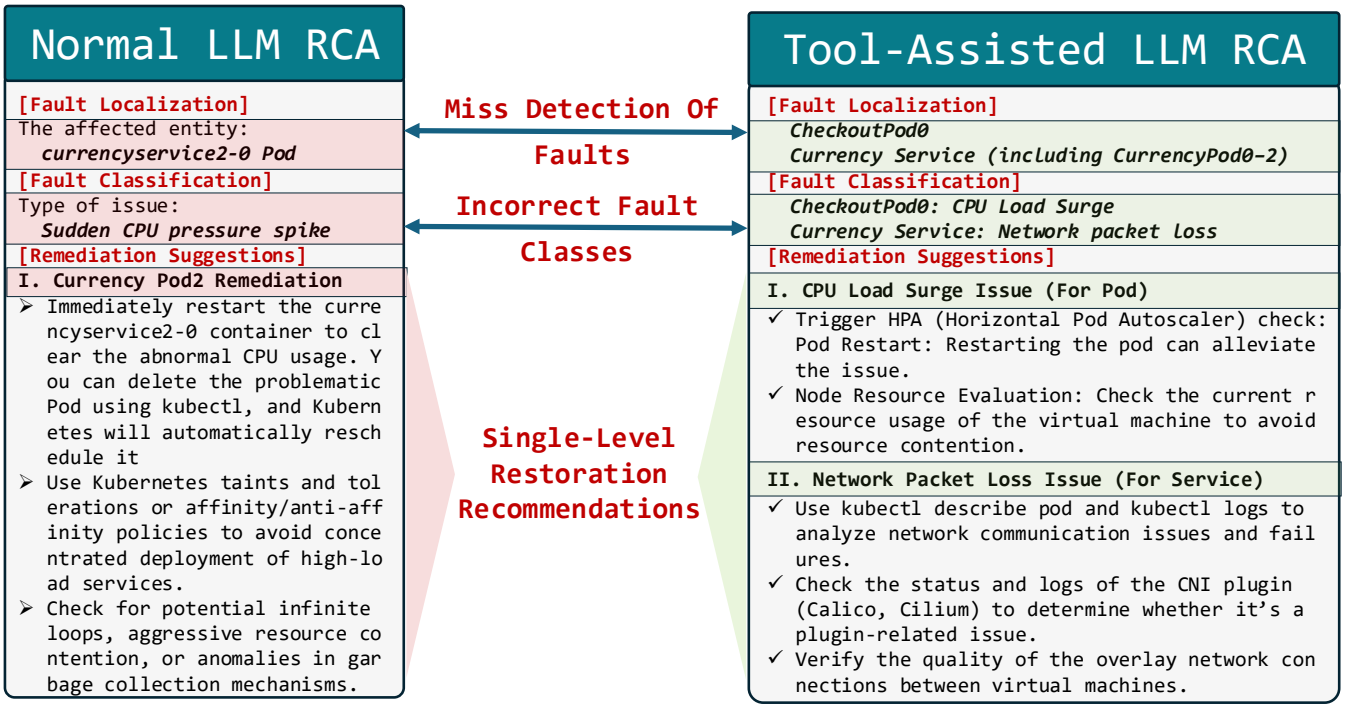


Fig. 7. Comparison of LLM results between RCA analysis of Agent $\mathcal{A}$ enhanced with TAMO tools and root cause analysis performed directly on raw data

the fault classification task, we perform a visual analysis of observation data for the anomalous services/Pods: CheckoutPod0 is accurately classified as a CPU overload fault due to a significant spike in core metrics such as "CPU Usage Second." The Currency service exhibits a correlation between increased service-level response time and decreased request success rate, while its Pod-level Receive Packet count continues to decline. TAMO successfully identifies the anomaly as a network packet loss fault through multi-dimensional time-series pattern analysis.

Subsequently, we input the model inference results along with system context information into the RCA expert LLM and compare its output with that of an LLM fed directly with raw data, as shown in Fig.7. This framework leverages tool information and domain expertise to accurately analyze current system faults and provide reasonable remediation suggestions. In contrast, the normal LLM using only raw data struggles to handle the vast amount of observation data due to severe context limitations, resulting in significant omissions and misjudgments in root cause identification and fault classification, as well as corresponding errors in the proposed remediation suggestions.

V. Conclusion

In this paper, we introduced a tool-assisted LLM agent framework named TAMO to address the complex challenges of root cause analysis in cloud-native systems. By integrating domain-specific tools with large language models, TAMO efficiently handled high-dimensional, multi-modal cloud-native observation data and accurately located root causes of failures in dynamically changing service dependencies. The approach not only overcame the information loss issues incurred when converting raw observability data into a format suitable for LLM processing, but also addressed the challenge that static pre-trained knowledge struggled to adapt to real-time changes in service dependencies. The experimental results show that, compared to existing advanced methods and benchmarks, TAMO performs notably better on two standard datasets, highlighting its potential to improve fault diagnosis efficiency and accuracy.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Ayman Amin, Lars Grunske, and Alan Colman. An approach to software reliability prediction based on time series modeling. *Journal of Systems and Software*, 86(7):1923–1932, 2013.

[3] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[4] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, omega, and kubernetes. *Communications of the ACM*, 59(5):50–57, 2016.

[5] Jiyu Chen, Heqing Huang, and Hao Chen. Informer: Irregular traffic detection for containerized microservices rpc in the real world. *High-Confidence Computing*, 2(2):100050, 2022.

[6] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, Jun Zeng, Supriyo Ghosh, Xuchao Zhang, Chaoyun Zhang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Tianyin Xu. Automatic root cause analysis via large language models for cloud incidents, 2023.

[7] Yujun Chen, Xian Yang, Qingwei Lin, Hongyu Zhang, Feng Gao, Zhangwei Xu, Yingnong Dang, Dongmei Zhang, Hang Dong, Yong Xu, et al. Outage prediction and diagnosis for cloud service systems. In *The world wide web conference*, pages 2659–2665, 2019.

[8] Zekai Chen, Dingshuo Chen, Xiao Zhang, Zixuan Yuan, and Xiuzhen Cheng. Learning graph structures with transformer for multivariate time-series anomaly detection in iot. *IEEE Internet of Things Journal*, 9(12):9179–9189, 2021.

[9] Qian Cheng, Doyen Sahoo, Amrita Saha, Wenzhuo Yang, Chenghao Liu, Gerald Woo, Manpreet Singh, Silvio Saverese, and Steven CH Hoi. Ai for it operations (aiops) on cloud platforms: Reviews, opportunities and challenges. *arXiv preprint arXiv:2304.04661*, 2023.

[10] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4027–4035, 2021.

[11] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1285–1298, 2017.

[12] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M Zhang. Large language models for software engineering: Survey and open problems. In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, pages 31–53. IEEE, 2023.

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

[14] Yongqi Han, Qingfeng Du, Ying Huang, Pengsheng Li, Xiaonan Shi, Jiaqi Wu, Pei Fang, Fulong Tian, and Cheng He. Holistic root cause analysis for failures in cloud-native systems through observability data. *IEEE Transactions on Services Computing*, 2024.

[15] Yongqi Han, Qingfeng Du, Ying Huang, Jiaqi Wu, Fulong Tian, and Cheng He. The potential of one-shot failure root cause analysis: Collaboration of the large language model and small classifier. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pages 931–943, 2024.

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

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[18] Chuanjia Hou, Tong Jia, Yifan Wu, Ying Li, and Jing Han. Diagnosing performance issues in microservices with heterogeneous data source. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 493–500. IEEE, 2021.

[19] Ziqi Huang, Kelvin C. K. Chan, Yuming Jiang, and Ziwei Liu. Collaborative diffusion for multi-modal face generation and editing, 2023.

[20] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[21] Asif Khan. Key characteristics of a container orchestration platform to enable a modern application. *IEEE cloud Computing*, 4(5):42–48, 2017.

[22] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[24] Peiwen Li, Xin Wang, Zeyang Zhang, Yuan Meng, Fang Shen, Yue Li, Jialong Wang, Yang Li, and Wenwu Zhu. Realtcd: temporal causal discovery from interventional data with large language model. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4669–4677, 2024.

[25] Yan Li, Mingyi Li, Xiao Zhang, Guangwei Xu, Feng Chen, Yuan Yuan, Yifei Zou, Mengying Zhao, Jianbo Lu, and Dongxiao Yu. Unity is power: Semi-asynchronous collaborative training of large-scale models with structured pruning in resource-limited clients. *arXiv preprint arXiv:2410.08457*, 2024.

[26] Yichen Li, Yulun Wu, Jinyang Liu, Zhihan Jiang, Zhuangbin Chen, Guangba Yu, and Michael R Lyu. Coca: Generative root cause analysis for distributed systems with code knowledge. *arXiv preprint arXiv:2503.23051*, 2025.

[27] Zeyan Li, Nengwen Zhao, Mingjie Li, Xianglin Lu, Lixin Wang, Dongdong Chang, Xiaohui Nie, Li Cao, Wenchi Zhang, Kaixin Sui, et al. Actionable and interpretable fault localization for recurring failures in online service systems. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 996–1008, 2022.

[28] Shengsheng Lin, Weiwei Lin, Wentai Wu, Feiyu Zhao, Ruichao Mo, and Haotong Zhang. Segrnn: Segment recurrent neural network for long-term time series forecasting. *arXiv preprint arXiv:2308.11200*, 2023.

[29] Dewei Liu, Chuan He, Xin Peng, Fan Lin, Chenxi Zhang, Shengfang Gong, Ziang Li, Jiayu Ou, and Zheshun Wu. Microhecl: High-efficient root cause localization in large-scale microservice systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 338–347. IEEE, 2021.

[30] Shutian Luo, Huanle Xu, Chengzhi Lu, Kejiang Ye, Guoyao Xu, Liping Zhang, Yu Ding, Jian He, and Chengzhong Xu. Characterizing microservice dependency and performance: Alibaba trace analysis. In *Proceedings of the ACM symposium on cloud computing*, pages 412–426, 2021.

[31] Nina Marwede, Matthias Rohr, André van Hoorn, and Wilhelm Hasselbring. Automatic failure diagnosis support in distributed large-scale software systems based on timing behavior anomaly correlation. In *2009 13th European Conference on Software Maintenance and Reengineering*, pages 47–58. IEEE, 2009.

[32] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers, 2023.

[33] Jannatun Noor, MD Badsha Faysal, MD Sheikh Amin, Bushra Tabassum, Tamim Raiyan Khan, and Tanvir Rahman. Kubernetes application performance benchmarking on heterogeneous cpu architecture: An experimental review. *High-Confidence Computing*, 5(1):100276, 2025.

[34] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, Mikhail Kudinov, and Jiansheng Wei. Diffusion-based voice conversion with fast maximum likelihood sampling scheme. *arXiv preprint arXiv:2109.13821*, 2021.

[35] Devjeet Roy, Xuchao Zhang, Rashi Bhave, Chetan Bansal, Pedro Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. Exploring llm-based agents for root cause analysis. In *Companion proceedings of the 32nd ACM international conference on the foundations of software engineering*, pages 208–219, 2024.

[36] Komal Sarda, Zakeya Namrud, Raphael Rouf, Harit Ahuja, Mohammadreza Rasolroveicy, Marin Litoiu, Larisa Shwartz, and Ian Watts. Adarma auto-detection and auto-remediation of microservice anomalies by leveraging large language models. In *Proceedings of the 33rd Annual International Conference on Computer Science and Software Engineering*, pages 200–205, 2023.

[37] Jacopo Soldani, Damian Andrew Tamburri, and Willem-Jan Van Den Heuvel. The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software*, 146:215–232, 2018.

[38] Lihua Song, Ying Han, Yufei Guo, and Chenying Cai. Idl-ltsoj: Research and implementation of an intelligent online judge system utilizing dnn for defect localization. *High-Confidence Computing*, 5(2):100268, 2025.

[39] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[40] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36:30840–30861, 2023.

[41] Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Jihong Wang, Fengbin Yin, Lunting Fan, Lingfei Wu, and Qingsong Wen. Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4966–4974, 2024.

[42] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

[43] Meixia Yang and Ming Huang. An microservices-based openstack monitoring tool. In *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, pages 706–709. IEEE, 2019.

[44] Zhenhe Yao, Changhua Pei, Wenxiao Chen, Hanzhang Wang, Liangfei Su, Huai Jiang, Zhe Xie, Xiaohui Nie, and Dan Pei. Chain-of-event: Interpretable root cause analysis for microservices through automatically learning weighted event causal graph. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, pages 50–61, 2024.

[45] Dongxiao Yu, Zhenzhen Xie, Yuan Yuan, Shuzhen Chen, Jing Qiao, Yangyang Wang, Yong Yu, Yifei Zou, and Xiao Zhang. Trustworthy decentralized collaborative learning for edge intelligence: A survey. *High-Confidence Computing*, 3(3):100150, 2023.

[46] Guangba Yu, Pengfei Chen, Yufeng Li, Hongyang Chen, Xiaoyun Li, and Zibin Zheng. Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 553–565, 2023.
[47] Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation, 2024.
[48] Shenglin Zhang, Pengxiang Jin, Zihan Lin, Yongqian Sun, Bicheng Zhang, Sibo Xia, Zhengdan Li, Zhenyu Zhong, Minghua Ma, Wa Jin, et al. Robust failure diagnosis of microservice system through multimodal data. *IEEE Transactions on Services Computing*, 16(6):3851–3864, 2023.
[49] Wei Zhang, Hongcheng Guo, Jian Yang, Zhoujin Tian, Yi Zhang, Chaoran Yan, Zhoujun Li, Tongliang Li, Xu Shi, Liangfan Zheng, et al. mabc: multi-agent blockchain-inspired collaboration for root cause analysis in micro-services architecture. *arXiv preprint arXiv:2404.12135*, 2024.
[50] Xiao Zhang, Shuqing Xu, Huashan Chen, Zekai Chen, Fuzhen Zhuang, Hui Xiong, and Dongxiao Yu. Rethinking robust multivariate time series anomaly detection: A hierarchical spatio-temporal variational perspective. *IEEE Transactions on Knowledge and Data Engineering*, 2024.
[51] Lecheng Zheng, Zhengzhang Chen, Jingrui He, and Haifeng Chen. Mulan: multi-modal causal structure learning and root cause analysis for microservice systems. In *Proceedings of the ACM Web Conference 2024*, pages 4107–4116, 2024.
[52] Xiang Zhou, Xin Peng, Tao Xie, Jun Sun, Chao Ji, Wenhai Li, and Dan Ding. Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study. *IEEE Transactions on Software Engineering*, 47(2):243–260, 2018.

Xiao Zhang is now an associate professor in the School of Computer Science and Technology, Shandong University. His research interests include data mining, distributed learning and edge intelligence. He has published more than 60 papers in the prestigious refereed journals and conference proceedings, such as IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Mobile Computing, ICML, NeurIPS, SIGKDD, UBICOMP, and INFOCOM.

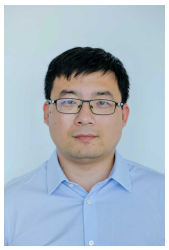
Qi Wang is a M.D. student in the School of Computer Science and Technology at Shandong University. He received his B.S. degree from Shandong University. His current research interests include time series and predictive maintenance.

Mingyi Li is currently a Ph.D. student in the School of Computer Science and Technology, Shandong University. She received her B.S. degree in Shandong University. Her research interests include distributed collaborative Learning and the theoretical optimization of distributed algorithms.

Yuan Yuan received the BSc degrees from the School of Mathematical Sciences, Shanxi University in 2016, and the Ph.D. degree from the School of Computer Science and Technology, Shandong University, Qingdao, China, in 2021. She is currently a postdoctoral fellow at the Shandong University-Nanyang Technological University International Joint Research Institute on Artificial Intelligence, Shandong University. Her research interests include distributed computing and distributed machine learning.

Mengbai Xiao , Ph.D., is a Professor in the School of Computer Science and Technology at Shandong University, China. He received the Ph.D. degree in Computer Science from George Mason University in 2018, and the M.S. degree in Software Engineering from University of Science and Technology of China in 2011. He was a postdoctoral researcher at the HPCS Lab, the Ohio State University. His research interests include multimedia systems, parallel and distributed systems. He has published papers in prestigious conferences such as USENIX ATC, ACM Multimedia, IEEE ICDE, IEEE ICDCS, IEEE INFOCOM.

Fuzhen Zhuang received the Ph.D. degrees in the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2011. He is currently a full Professor with the Institute of Artificial Intelligence, Beihang University. He has published more than 150 papers in some prestigious refereed journals and conference proceed-ings such as Nature Comunications, the IEEE Transactions on Knowledge and Data Engineering, the IEEE Transactions on Cybernetics, the IEEE Transactions on Neural Networks and Learning Systems, the ACM Transactions on Knowledge Discovery from Data, the ACM Transactions on Intelligent Systems and Technology, Information Sciences, Neural Networks, SIGKDD, IJCAI, AAAI, TheWebConf, ACL, SIGIR, ICDE, ACM CIKM, ACM WSDM, SIAM SDM, and IEEE ICDM. His research interests include transfer learning, machine learning, data mining, multitask learning, knowledge graph and recommendation systems. He is a Senior Member of CCF. He was the recipient of the Distinguished Dissertation Award of CAAI in 2013.



Dongxiao Yu received the B.S. degree in 2006 from the School of Mathematics, Shandong University and the Ph.D degree in 2014 from the Department of Computer Science, The University of Hong Kong. He became an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, in 2016. He is currently a professor in the School of Computer Science and Technology, Shandong University. His research interests include edge intelligence, distributed computing and data mining.