FoundRoot: Towards Foundation Model for Root Cause Analysis via Structured Deep Thinking

Zhe Xie
Tsinghua University
China
xiez22@mails.tsinghua.edu.cn

Zeyan Li
ByteDance
China
lizeyan.42@bytedance.com

Xiao He
ByteDance
China
xiao.hx@bytedance.com

Shenglin Zhang
Nankai University
China
zhangsl@nankai.edu.cn

Longlong Xu
Tsinghua University
China
xll23@mails.tsinghua.edu.cn

Yuzhuo Yang
Tsinghua University
China
yangyz24@mails.tsinghua.edu.cn

Tieying Zhang*
ByteDance
USA
tieying.zhang@bytedance.com

Jianjun Chen
ByteDance
USA
jianjun.chen@bytedance.com

Rui Shi
ByteDance
China
shirui@bytedance.com

Dan Pei
Tsinghua University
China
peidan@tsinghua.edu.cn

Abstract

Root Cause Analysis (RCA) for service systems is critical for ensuring their reliability, while its application remains challenging because of the large number of metrics and the complex causal relationships. Classical RCA methods typically rely on statistical or rule-based approaches, making them difficult to generalize to unseen systems. The introduction of Large Language Models (LLMs) has partly addressed these challenges with their understanding of the domain-specific semantics of metrics and reasoning capabilities. However, they still struggle with incomplete or shallow reasoning when facing a large amount of metrics. To address these limitations, we present FoundRoot, a reinforcement learning (RL)-enhanced LLM foundation model for zero-shot RCA. FoundRoot features a novel structured deep thinking paradigm, which breaks down the RCA reasoning into several goal-oriented substeps, enhancing the completeness and reasoning capability of the causal relationship. We collect and curate diverse open-sourced RCA datasets across different systems and introduce a data augmentation technique to ensure data scalability. We design a two-stage training pipeline that includes supervised fine-tuning (SFT) and RL to align the LLM with the structured deep thinking paradigm, which significantly improves its reasoning quality. Extensive experiments on four datasets show that FoundRoot outperforms both classical and LLM-based methods in RCA accuracy on unseen systems, achieving 4.5%-48.6% mean reciprocal rank (MRR) improvements. The source code and data of this paper is available at: https://github.com/NetManAIOps/FoundRoot.

CCS Concepts

• **Software and its engineering** → **Maintaining software**; • **Computing methodologies** → *Natural language generation.*

Keywords

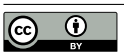
Root Cause Analysis, LLM Reasoning

ACM Reference Format:
Zhe Xie, Zeyan Li, Xiao He, Shenglin Zhang, Longlong Xu, Yuzhuo Yang, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. 2026. FoundRoot: Towards Foundation Model for Root Cause Analysis via Structured Deep Thinking. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3744916.3787814

1 Introduction

Nowadays, service-oriented and component-based systems are widely adopted. As their scale grows and failures become inevitable, accurately identifying the root cause of failures is essential for rapid recovery. Therefore, root cause analysis (RCA) is critical to ensuring the reliability of such systems [53]. In these systems, metrics are collected to monitor the status of components. RCA algorithms analyze these metrics to locate the root cause of failures, assisting operators in achieving efficient recovery. However, in real-world systems, failures often propagate across components due to dependencies, and their underlying causal relationships are highly

*Corresponding author.



ICSE '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2025-3/2026/04
https://doi.org/10.1145/3744916.3787814

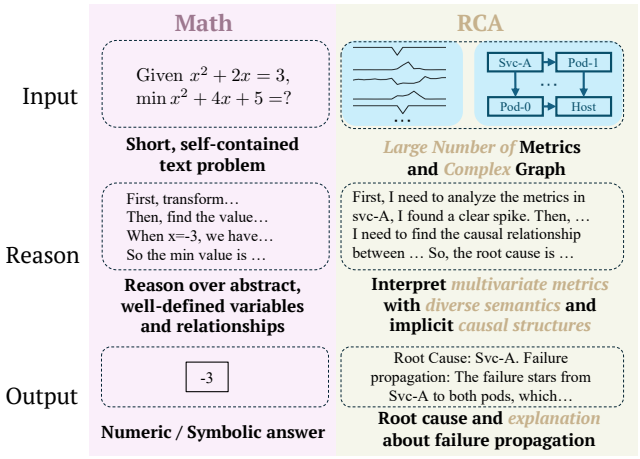


Figure 1: Comparison of math and RCA reasoning.

complex, making RCA particularly challenging. Additionally, historical failure data is often scarce or unavailable because collecting such data requires extensive manual effort. This highlights the need for *zero-shot* RCA methods that can identify faults directly from current monitoring data without relying on historical labels.

Classical metric-based RCA approaches usually rely on the analysis of numerical values of the metrics to infer causal relationships [20, 23, 25]. However, real-world metric data is often noisy and incomplete. To address this, many methods incorporate expert-defined rules tailored to specific systems [16, 26]. These rules are difficult to scale and cannot be generalized across different environments. Fundamentally, these methods cannot interpret the semantics carried by the meta information of metrics (*e.g.*, the domain-specific semantics inferred from metric names), limiting their effectiveness in complex and dynamic systems.

Since the emergence of LLMs, many studies have explored their application to RCA [34, 54], leveraging their strengths in understanding the domain-specific semantics of monitoring metrics. Mainstream approaches employ agent-based methods [32, 38, 49], where time series analysis tools are invoked to assist in analyzing metrics. However, these approaches still struggle to reason about the complex causal relationships among multivariate time series due to the large number of metrics and their intricate causal dependencies.

With the introduction of the "deep thinking" paradigm in OpenAI's o1 [31] model, the reasoning capabilities of LLMs are significantly improved, which we find to be beneficial for RCA tasks (see Section 2.3). Nevertheless, the current application of LLM deep thinking to RCA remains limited in its effectiveness. Existing deep-thinking benchmarks for LLMs have primarily focused on structured domains such as mathematics and code [45]. As illustrated in Figure 1, RCA tasks differ significantly, requiring comprehensive analysis of a large number of metrics and complex component graphs. We find that existing LLMs often encounter serious incompleteness and shallow reasoning during deep thinking (see Section 2.3) when applied to RCA tasks, limiting their accuracy. However, building an LLM for RCA faces several challenges. First, RCA tasks involve a large amount of metrics data and component dependencies, which differ fundamentally from the well-structured inputs in math or code reasoning. Second, to perform zero-shot generalization across different environments, the model should be easily adapted to systems with varying architectures. Third, collecting high-quality datasets for RCA remains challenging due to the diversity of system architectures and data formats, resulting in limited training resources for LLMs.

In this work, we propose **structured deep thinking**, a novel LLM-based reasoning strategy for RCA tasks. Structured deep thinking breaks down the overall reasoning process of RCA into several *substeps*. Each substep focuses on reasoning for a specific purpose, and all substeps are completed within a single LLM inference, ensuring logical coherence in the output. This design significantly mitigates the incompleteness and shallow reasoning while preserving the logical coherence inherent in numerous metric inputs. To address the challenge of dataset scarcity, we conducted an extensive search for existing open-source RCA datasets that include metrics. Finally, we collected 10 high-quality open-sourced datasets, which are across different systems. To the best of our knowledge, this is the largest collection of datasets in existing RCA research. Building upon structured deep thinking, we design a two-stage training pipeline: a warm-up supervised fine-tuning (SFT) followed by reinforcement learning (RL), to significantly improve the LLM's reasoning capabilities for RCA. Based on these efforts, we introduce **FoundRoot**, an LLM equipped with structured deep thinking capabilities for zero-shot RCA. Our extensive experiments across multiple real-world RCA datasets demonstrate that FoundRoot generalizes well to unseen systems and significantly outperforms both traditional statistical methods and strong LLM baselines. The datasets, source code, and the model checkpoints are available at https://github.com/NetManAIOps/FoundRoot.

Our contributions are as follows:

- We present FoundRoot, the first reinforcement learning-enhanced LLM foundation model for zero-shot root cause analysis.
- We propose a novel *structured deep thinking* paradigm that decomposes RCA reasoning into goal-oriented substeps, improving the completeness and causal reasoning capability of LLMs.
- We show that the two-stage training pipeline, which integrates SFT and RL with structured deep thinking, can significantly enhance the LLM's zero-shot generalization to unseen systems.
- We collect and curate a diverse set of high-quality RCA datasets, and show that the proposed FoundRoot model consistently outperforms both classical and LLM-based baselines.

2 Preliminaries and Motivation

2.1 Problem Formulation

In this paper, we formulate the task for zero-shot RCA foundation models as follows. In a collection of datasets $\mathcal{D}$, each dataset $D_m \in \mathcal{D}$ contains several failure cases. Each failure case is composed of:

- A set of multivariate time series windows $\mathcal{X} = \{(x_i, m_i, c_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^T$ is the time series of the i-th metric, m_i is its name, and $c_i \in C$ is the associated component.
- A component dependency graph $\mathcal{G} = (C, \mathcal{E})$, where C denotes the set of components and $\mathcal{E}$ denotes the set of edges.
- A failure root cause component c_{rc}.

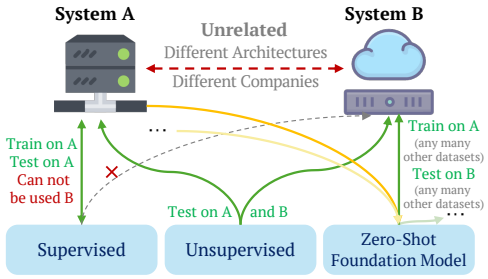
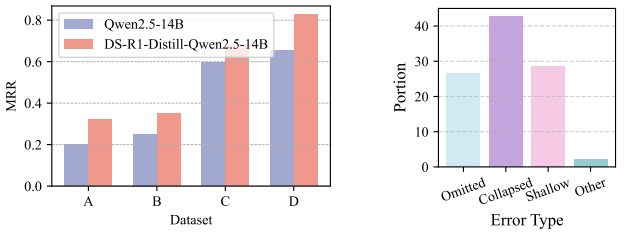


Figure 2: Types of Models for RCA



(a) w/ or w/o deep thinking **(b) Error w/ deep thinking**

Figure 3: LLMs achieve consistent improvements on RCA tasks with deep thinking. However, they still frequently fail when handling inputs with numerous metrics.

Omitted: Root cause metric is completely omitted during reasoning

Example (Root Cause: os_018 Network)
<think>Alright, I'm trying to figure out...
First, looking at db_008, I've ... Looking further, docker_002 ...
...(metrics related to os_018 Network are completely omitted)...
So, the root cause seems to be **db_007 ...** </think>

Collapse: Metrics are analyzed initially but contradictions arose later

Example (Root Cause: docker_006)
<think>...docker_006##proc spikes **indicate increased processing**... db_003's drop **might be the downstream** of docker_006... Therefore, **db_007 is the root cause**, other components like **docker_006** is either **noise or normal...**

Shallow: Reasoning is shallow and contains some clear factual errors

Example (Root Cause: db_003)
<think>...looking at the component graph, I see that **docker_005 calls db_003**... Considering the component graph, if docker_005 fails, it would affect db_003...

Figure 4: Examples of different categories of errors w/ deep thinking in LLMs

The goal is to predict a ranked list $\mathcal{Y}$ of the components:

$$\mathcal{Y} = f_\theta(\mathcal{X}, \mathcal{G}) \quad \text{where} \quad \mathcal{Y} \subseteq C \text{ and } y_0 = \hat{c}_{rc}$$

In the **zero-shot** foundation model setting (Figure 2), the model f_θ is trained on datasets $\mathcal{D}_{\text{train}}$ and evaluated on disjoint datasets $\mathcal{D}_{\text{test}}$ with different system structures and metric spaces (*i.e.*, $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \emptyset$). This setting requires the model to generalize across heterogeneous systems *without retraining*.

2.2 Classical Methods

Classical unsupervised RCA methods [19, 24, 26, 29, 39, 44] typically follow a three-stage pipeline:

(1) First, anomaly detection algorithms are applied to identify the anomalies in time series data.
(2) Next, causal graphs are constructed to capture relationships among metrics. Given the component graph $\mathcal{G}$, the causal graphs are usually built using statistical methods (*e.g.*, causal discovery) and predefined expert rules. However, such methods often fail to incorporate the domain-specific semantics of the metrics, leading to inaccurate or incomplete causal relationships.
(3) Finally, components are ranked based on the graph, where PageRank, random walk, and intervention-based techniques are employed. This step also considers only the statistical and topological relationships among metrics and ignores their domain-specific semantics. This limits its effectiveness in real-world scenarios where understanding system semantics is critical.

Overall, while classical methods are intuitive and easy to implement, they struggle to handle noise, analyze complex failure propagation behaviors, and provide meaningful, interpretable explanations.

2.3 LLM-based Methods

2.3.1 ***Deep thinking can bring consistent improvement in RCA tasks to LLMs.*** To explore the effectiveness of deep thinking in LLMs in RCA tasks, we compared models of the same size with and without deep thinking. Figure 3a compares the performance of the base model (*Qwen2.5-14B-Instruct*) and its deep-thinking version (*DeepSeek-R1-Distill-Qwen2.5-14B*) across four evaluation datasets (see Section 3.3). The model with deep thinking capability consistently outperforms the base model. This demonstrates that deep thinking indeed contributes to enhancing the reasoning capabilities required for RCA. LLMs with deep thinking not only retain the advantage of understanding domain-specific semantics but also leverage their reasoning capabilities to achieve further improvements.

2.3.2 ***LLMs with deep thinking still fail when facing complex MTS inputs. Common failure modes include omitting key metrics, inconsistent reasoning, and shallow analysis.*** Although deep thinking can improve RCA reasoning capabilities, we find that these models still generate a large number of incorrect answers. We classify these error cases into four major categories (omitted, collapsed, shallow, and other). The different categories and corresponding examples are illustrated in Figure 4. Figure 3b shows the proportion of different categories within the error cases produced by LLMs with deep thinking. Among 49 incorrect cases, a significant portion falls into these three major categories. Notably, the "Collapsed" error category accounts for the highest proportion. This suggests that LLMs with deep thinking are prone to forgetting or generating inconsistencies when faced with a large volume of metrics, which significantly reduces RCA accuracy. Furthermore, error cases of "metric omitted" and "shallow reasoning" are also highly relevant. This is also caused by the large number of metrics

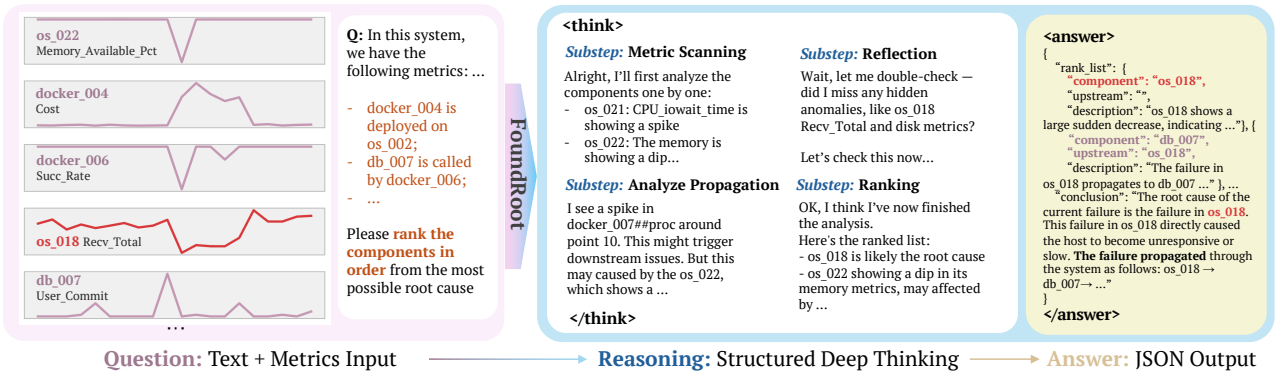


Figure 5: The input/output format and structured deep thinking in FoundRoot. Structured deep thinking employs substeps with explicit goals to ensure comprehensive reasoning while maintaining depth through continuous inference.

and the complex causal relationships in RCA tasks. These findings suggest that while deep thinking improves overall reasoning quality, it remains struggling in reasoning about the causal relationship across multiple metrics.

To address these challenges, we propose *structured deep thinking*, an approach that explicitly organizes reasoning into substeps, guiding the model to systematically analyze and reflect. By breaking down the thinking process into several substeps with preset goals, structured deep thinking forces the LLMs to focus on every metric and to do reflection on the causal relationships.

3 Method

In this section, we first present the general idea of structured deep thinking, followed by its details in Section 3.1. Next, we present an overview of the pipelines to build FoundRoot in Section 3.2. Finally, we describe the dataset and the LLM pipelines in detail in Section 3.3 and Section 3.4, respectively.

3.1 Structured Deep Thinking

As motivated in Section 2.3, while general-purpose deep thinking capabilities can enhance LLM performance on RCA tasks (Finding 1), they still frequently produce incomplete and shallow reasoning (Finding 2). To address these issues, we introduce **structured deep thinking**, inspired by the standard RCA workflow of detection, analysis, causal attribution, and ranking [29].

The core idea of structured deep thinking is to break down the complex RCA deep thinking process into a sequence of distinct, goal-oriented substeps, while keeping them in a single output without additional prompts. By designing the thinking goal at each substep, LLMs are guided to analyze all the metrics in the failure case, which prevents them from missing key points. Different from agent-based or workflow-based methods [32, 49], structured deep thinking is an ability *embedded within LLMs through training* and performed within a *single, continuous inference*, rather than being guided with prompts. This preserves the logical coherence and context of a unified thought process, mitigating the error accumulation and context loss with decoupled tool calls or prompts. Each substep allows unconstrained internal reasoning and is then optimized through RL. As illustrated in Figure 5, the structured deep thinking guides FoundRoot through four substeps following the RCA practice of human operators:

(1) **Metric Scanning:** Iterate through all components and their associated metrics to identify any potential anomalies or noteworthy patterns. This step is explicitly designed to ensure no critical information is overlooked, directly addressing the *omitted* metric problem.
(2) **Propagation Analysis:** Analyze the potential failure propagation paths according to the anomalous metrics in the previous step. This is the core substep of the RCA process. LLMs can analyze the failure propagation using their internal reasoning capabilities without any additional constraints.
(3) **Reflection:** A reflection step to prevent the LLM from reasoning collapse when faced with a large number of metrics. The LLM double-checks its reasoning and tries to identify the inconsistency of previous steps. This self-correction step is crucial for preventing *collapsed* and *shallow* reasoning.
(4) **Ranking:** Finally, based on the comprehensive analysis from the previous stages, the model concludes its findings to generate a *ranked list* of potential root causes. Each ranked component is accompanied by a detailed explanation, including the evidence from the metrics and its upstream component.

By integrating these structured substeps into a single, continuous output, FoundRoot internalizes a more robust and generalizable reasoning pattern to mitigate the incompleteness and shallow reasoning for RCA. Motivated by OpenAI o1 [31] and DeepSeek-R1 [9], we choose to employ RL to enable the model's structured deep thinking ability. Further implementation details are provided in the following sections.

3.2 Data and Model Training Pipelines

In this section, we illustrate the overall training workflow of building FoundRoot, which is depicted in Figure 6. The data pipeline initiates with input data in the question-answering (QA) format, followed by data augmentation to improve its diversity. Subsequently, we utilize an LLM to generate *warmup SFT data* in a structured deep thinking format by means of *structurally-constrained continuation generation*. Based on this, we use Supervised Fine-Tuning (SFT) to train the *WarmUp-SFT Only* version of FoundRoot model, which is

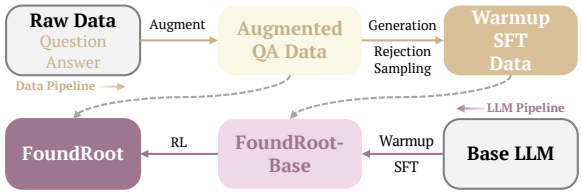


Figure 6: The data and model pipelines of FoundRoot.

Table 1: Datasets overview. (Aug.) denotes augmented cases.

Dataset	# Cases (Aug.)	System	Split
A [22]	63	A service system of a major ISP	Test
B [22]	90		
C [15]	10	SocialNetwork	
D [5]	255	MicroSS (GAIA)	
E [22]	136 (1,158)	A service system of a bank	Train
F [21]	81 (769)	Oracle DB	
G [15]	43 (345)	TrainTicket	
H [50]	34 (259)		
I [21]	94 (771)		
J [50]	27 (200)	HipsterShop	

capable of structured deep thinking. To obtain the final *FoundRoot* model, we further apply RL with the augmented QA data, which enhances the model's reasoning capability on the RCA task.

3.3 Curation of Datasets

As shown in Figure 6, our data pipeline includes several steps. First, we have collected and preprocessed a large number of datasets (Section 3.3.1). Next, to support model training, we augmented the dataset (Section 3.3.2) and constructed a warm-up SFT dataset (Section 3.3.3) with structured deep thinking traces.

3.3.1 Dataset Construction. Although several works have introduced synthetic datasets for model training [47, 52], it is very difficult to synthesize data for RCA. This is because fault propagation is closely related to the domain-specific semantics of the components. Therefore, we focus on using real-world datasets for training and evaluation. To train a generalizable RCA foundation model, our dataset construction process aims to ensure diversity in system architectures, metric semantics, and failure propagation patterns, which are essential for promoting cross-system generalization. We collect and curate a collection of large-scale datasets from a large number of open-sourced datasets. The details of the collected datasets are shown in Table 1. Our dataset collection includes ten datasets in total, with six datasets for training and four datasets for evaluation. All datasets are formatted into a uniform Q&A format (refer to Figure 5) for LLM training and evaluation.

The training datasets cover a variety of domains and system types, including bank service system [22], Oracle database [21], and microservice-based applications from [15, 21, 50]. In contrast, the evaluation datasets are all collected from totally different systems from the systems in the training datasets. To comprehensively evaluate the models in different system architectures, we include datasets from both service systems (A and B) [21] and microservices systems (C [15] and D [5]) in the collection of evaluation datasets.

Data quality is crucial for training LLMs. Therefore, we perform extensive preprocessing and cleaning across all datasets in order to ensure the data quality. First, most open-source RCA datasets are generated using failure injection methods on testbeds. These datasets may contain many cases of failed injections. Therefore, we used both manual and automated methods to check all fault cases. This process ensures that the training and evaluation are correct. Furthermore, the data contains many noisy or normal metrics that are unrelated to the failure. We used a k-sigma anomaly detection [3] method to filter out normal metrics, which significantly reduced the number of metrics. For each failure case, we collect a time window of data before and after the failure and convert it into text as input. All metric values were converted into text format for input. We do not focus on the encoding of time series, which is beyond the scope of this work. We also unified the input and output formats of the datasets for training the foundation model.

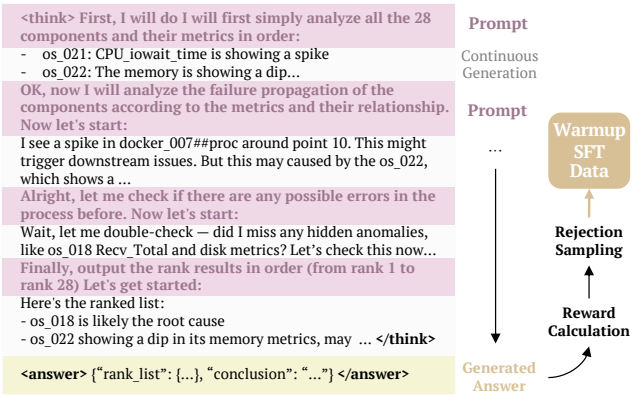


Figure 7: An example of generating warm-up SFT data with structured deep thinking. Prompts are interleaved with LLM's continuous generation to ensure coherence, and rejection sampling is used to retain high-quality data.

3.3.2 Data Augmentation. Given the limited size of RCA datasets and the risk of overfitting, we further apply augmentation techniques to the training datasets. To increase data diversity and simulate different observability conditions, we apply a sampling-based data augmentation strategy. Given a multivariate time series $\mathbf{X}$ with N metrics and T time points, we generate multiple augmented variants through two sampling operations:

Temporal sampling. For each *training* case, we randomly select sub-windows of variable length $L \sim \mathcal{U}(L_{\min}, L_{\max})$ to simulate diverse monitoring durations. A sampled window is denoted as

$$\mathbf{x}_i^{(t)} = \mathbf{x}_i[t : t + L - 1]. \tag{1}$$

This increases temporal diversity by varying both the start position and the window length.

Metric sampling. To avoid removing root-cause signals, we always keep all root-cause metrics $\mathcal{R}$ in the sampled subset. Then, we randomly sample additional non-root-cause metrics $\mathcal{S}_{\text{non-rc}}$ with size drawn from a uniform range, and form the final metric set as $\mathcal{S} = \mathcal{R} \cup \mathcal{S}_{\text{non-rc}}$. The resulting sampled multivariate series is $\mathbf{X}' = \{\mathbf{x}_i^{(t)} \mid i \in \mathcal{S}\}$.

The augmented training dataset is obtained by repeatedly sampling time windows and metric subsets for each original training case:

$$\mathcal{D}_{\text{train}}^{\text{aug}} = \bigcup_{j=1}^{M_{\text{train}}} \{(\mathbf{X}'_{j,k}, y_j)\}_{k=1}^{K_j}. \tag{2}$$

In practice, this procedure expands the 415 original *training* cases into 3,502 augmented samples. All *test* cases remain unchanged to ensure a fair evaluation.

3.3.3 Warm-Up SFT Dataset. To enable the model to support structured deep thinking, we construct a warm-up SFT dataset, which includes complete thinking traces and Q&A pairs. Structured deep thinking requires each substep to follow a specific format and objective. As a result, it is difficult for existing LLMs to directly generate thinking process that meet these requirements. To address this, we designed a strategy called *structurally-constrained continuation generation*. As shown in Figure 7, each substep begins with a set of carefully designed prompts. These prompts guide the LLM to continuously generate the thinking process for the current step in the structured thinking format. By alternating between prompt and LLM continuous generation, it forms a complete and fluent structured reasoning trace. During generation, we insert explicit control phrases such as "After this, I will perform Step 2" in the prompts to mark the end of each substep, which are later removed during data extraction. When the model produces outputs that cannot be parsed within the predefined `max_tokens` limit, we truncate or discard those samples to ensure structural consistency. We further apply rejection sampling [30] with the reward functions defined in Section 3.4.4 to filter out samples that violate the structural constraints or contain substantial reasoning errors. Through this process, we obtain the WarmUp SFT dataset that conforms to the required structure and maintains reliable quality without manual post-editing.

It is important to note that this interleaved prompting is *only used in the data generation stage.* During inference, the trained model performs deep thinking in a continuous and unified manner with its inherent structured deep thinking ability, without the need for intermediate prompts.

3.4 Model and Training Details

The training pipeline of FoundRoot follows a two-stage process, as illustrated in Figure 6. The first stage is a warm-up supervised fine-tuning (SFT), where the model is trained to follow the format of structured deep thinking. The second stage adopts Dynamic Sampling Policy Optimization (DAPO) [51] to further enhance the reasoning capabilities of the LLM for RCA.

3.4.1 Input and Output. The input and output format of FoundRoot is illustrated in Figure 5. As FoundRoot is a text-based model, we convert values in the time series windows to text in the input. To reduce the number of input metrics, we use k-sigma anomaly detection to filter out normal metrics. The model's output consists of two parts. The reasoning part contains the complete structured deep thinking content and is enclosed within `<think>` tags. The answer part is enclosed by a `<answer>` tag and contains a dictionary in `json` format. In the answer part, the `rank_list` field is a list. Each entry in the list represents a component, ranked from the highest to the lowest probability of being the root cause. An entry also includes the *upstream component* of the fault propagation and a `corresponding description`. The `conclusion` field contains a complete explanation of the fault propagation. This allows the model's output to be parsed accurately and makes it convenient for operators to check the failure.

3.4.2 Warm-Up SFT. In the first stage, we finetune a base LLM (DeepSeek-R1-Distill-Qwen2.5-14B) on the warm-up SFT dataset (see Section 3.3). The warm-up SFT dataset includes the question as input and the complete reasoning traces and answer as output. The outputs are supervised under a fixed format with explicit '<think>' and '<answer>' tags, helping the model to learn coherent and complete causal reasoning. The warm-up SFT stages equip the LLM with the ability of structured deep thinking, thereby reducing the difficulty of enforcing format constraints in the following RL step.

3.4.3 RL with DAPO. After the warm-up SFT stage, the model acquires a basic ability for structured deep thinking, but its reasoning on RCA tasks still needs further improvement. Therefore, we adopt reinforcement learning (RL) with DAPO [51] in the second stage to enhance reasoning quality and robustness.

DAPO strengthens the model's reasoning process by samplling multiple possible reasoning paths for each RCA case, evaluating them with the rewards in Section 3.4.4, and learning from the best reasoning paths. Through repeated sampling and optimization, the model gradually learns to produce more accurate reasoning and perform RCA correctly, without requiring any manually labeled reasoning traces. Compared with previous RL methods such as GRPO [9], DAPO introduces a more flexible update strategy that allows the model to explore diverse reasoning paths, which helps the model better handle complex reasoning in RCA tasks. The training in this stage is performed on the augmented RCA datasets described earlier.

3.4.4 Reward Design. We define the total reward $r(x, y)$ as a weighted sum of multiple components, each targeting a specific aspect of RCA accuracy and output structure:

$$r(x, y) = \lambda_{\text{format}}\, r_{\text{format}} + \lambda_{\text{json}}\, r_{\text{json}} + \lambda_{\text{acc}}\, r_{\text{acc}} + \lambda_{\text{think}}\, r_{\text{think}} \tag{3}$$

The components are defined as follows:

- **Format (r_{format})**: A binary reward where $r = 1$ if and only if the output contains both `<think>` and `<answer>` tags.
- **JSON format (r_{json})**: A binary reward where $r = 1$ if and only if the answer is valid JSON and its fields meet the requirements.
- **RCA accuracy (r_{acc})**: Model's output is a JSON list that contains a ranked list ordered from the component most likely to be the root cause to the one least likely. The ground-truth root cause rank rank_{gt} is computed by finding the position of the root cause component in this list. Given the ground-truth root cause rank

rank_{gt}, this reward is defined as the MRR (mean reciprocal rank):

$$r_{\text{mrr}} = \frac{1}{\text{rank}_{\text{gt}} + 1} \tag{4}$$

- **Structured thinking (r_{think})**: A binary reward where $r = 1$ if and only if the entire reasoning trace strictly follows the four-substep structured deep thinking format (metric scanning, propagation analysis, reflection, and ranking).

4 Evaluation

To comprehensively assess the effectiveness and generalizability of FoundRoot, we design a series of experiments to address the following research questions:

- **RQ1.** Does FoundRoot outperform classical and LLM-based RCA baselines?
- **RQ2.** How effective is the proposed structured deep thinking in improving RCA performance?
- **RQ3.** Does RL really enhance the model's reasoning performance?
- **RQ4.** How well does FoundRoot generalize to unseen systems with different architectures and metric distributions?
- **RQ5.** How do smaller/quantized variants of FoundRoot perform?

The following sections detail our experimental setup and provide a systematic analysis for each of the above questions.

4.1 Experimental Setup

4.1.1 Evaluation Metrics. We adopt three widely used metrics for evaluating root cause localization: *Top-1 Accuracy*, *Top-3 Accuracy*, and *MRR* (Equation 4), following common practice in existing approaches [21, 48]. These metrics quantify how accurately the model can identify the root cause component(s) from a ranked list. All metrics are computed at the component level. For all metrics, higher values indicate better performance.

4.1.2 Evaluation Datasets. As shown in Table 1, we collected a total of 10 datasets, among which 4 are used for evaluation. The datasets used for evaluation have system architectures that are completely distinct from those used for training, ensuring the reliability of the zero-shot evaluation.

- **Dataset A, B**: Two server datasets from a large financial institution, originally used in the AIOps Challenge [22].
- **Dataset C**: Collected from the SocialNetwork microservice [15].
- **Dataset D**: Collected from the MicroSS microservice by GAIA [5].

These datasets are completely held out during training to ensure a strict zero-shot generalization setting. Each incident in the dataset includes a set of metric values, their corresponding components, a component dependency graph, and a ground truth root cause label.

4.1.3 Baselines. We compare FoundRoot against a diverse set of baseline approaches, including both classical RCA algorithms and LLM-based methods.

Classical Methods: To compare the performance of different types of algorithms, we choose classic causal-graph-based algorithms, as well as the recently studied intervention-based algorithms, the change-point-based algorithm, and the E2E algorithm.

- **MicroScope**: MicroScope [23] uses PC algorithm [39] for graph construction and ranks the root cause with Pearson correlation.
- **MonitorRank**: MonitorRank [13] ranks the metrics with personalized PageRank algorithm.
- **MicroCause**: MicroCause [29] uses PCMCI [35] for causal discovery and ranks the metrics with a second-order random walk.
- **CIRCA**: CIRCA [16] proposes an intervention recognition-based approach for causal inference.
- **RCD**: RCD [12] propose localized a hierarchical learning algorithm for intervention-based RCA.
- **BARO**: BARO [33] employs BOCPD [1] to detect failures and a nonparametric statistical hypothesis testing technique for robustly identifying root causes.
- **ART**: ART [40] proposes to represent failures as system and instance-level deviations with an E2E framework. It ranks the instances according to their deviation similarity to the system.

LLM-based Methods: We comprehensively compare the abilities of non-thinking models, thinking models, and agent-based models.

- **LLMs**: We select several mainstream open-source and closed-source LLMs as baselines. These include models without deep reasoning capabilities, such as Qwen2.5-14B and GPT-4o, as well as models with deep reasoning capabilities, including OpenAI o4-mini, DeepSeek-R1-Distill-Qwen2.5-14B (R1-14B), DeepSeek R1 (R1-Full), and Doubao-Seed-1.6-Thinking (Doubao-Thinking). The input question format for these models is kept consistent with that of FoundRoot, without any additional processing.
- **RCA-Agent**: RCA-Agent, proposed by OpenRCA [49], leverages LLMs to generate and execute Python code for reading metric data, using R1-Full as the LLM.

Most classical methods, including CIRCA, RCA, BARO, and ART, are implemented using the official code released by their original authors. The remaining classical baselines are re-implemented due to the unavailability of publicly released source code. For LLM-based baselines, all methods are implemented using their official repositories and APIs.

4.1.4 Implementation Details. All models are trained using the Open-R1 framework [11] and the TRL library [42]. FoundRoot is initialized from DeepSeek-R1-Distill-Qwen2.5-14B[1] and trained for 400 steps with a batch size of 32 on 8×A100 GPUs, with temperature=1.0 and lr=1e-6. Inference is conducted using vLLM [14] with top-5 sampling. In our experiments, we empirically set the reward weights to: $\lambda_{\text{format}} = 1.0$, $\lambda_{\text{struct}} = 1.0$, $\lambda_{\text{mrr}} = 3.0$, and $\lambda_{\text{stf}} = 1.0$. Implementation details can be found in our source code.

4.2 RQ1: Does FoundRoot outperform classical and LLM-based RCA methods?

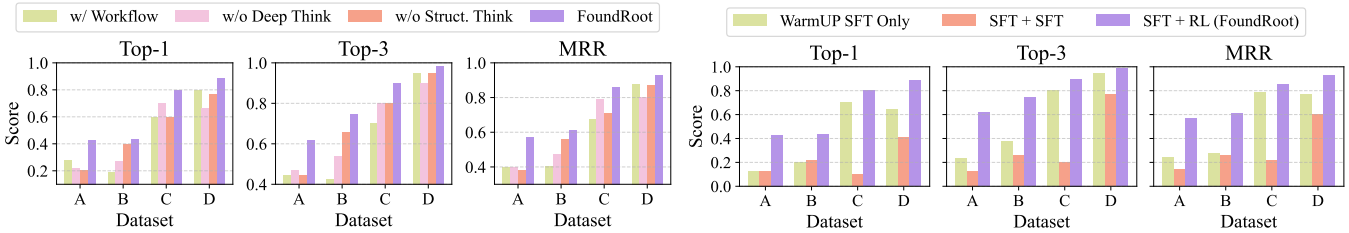
To answer RQ1, we compare FoundRoot with a range of classical RCA methods and modern LLM-based approaches across four benchmark datasets (A–D). The evaluation results are shown in Table 2. The results demonstrate that FoundRoot consistently outperforms all baselines across all datasets and metrics, indicating both superior RCA accuracy and strong generalization capability.

Looking at the results in Table 2, FoundRoot presents a large improvement compared to the second-best option. However, in certain datasets (*e.g.*, datasets A and B), its accuracy in some cases

[1] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B

Table 2: Comparison of baseline models. Higher is better.

	Dataset	A			B			C			D		
Type	Method	Top-1	Top-3	MRR	Top-1	Top-3	MRR	Top-1	Top-3	MRR	Top-1	Top-3	MRR
Classical	MicroScope	0.143	0.159	0.221	0.089	0.233	0.211	0.300	0.400	0.462	0.436	0.795	0.645
	MonitorRank	0.063	0.095	0.154	0.078	0.167	0.228	0.200	0.200	0.296	0.461	0.897	0.687
	MicroCause	0.159	0.413	0.292	0.089	0.489	0.361	0.100	0.700	0.383	0.410	0.923	0.636
	CIRCA	0.079	0.302	0.272	0.144	0.400	0.329	0.400	0.700	0.571	0.282	0.871	0.553
	RCD	0.143	0.270	0.188	0.111	0.144	0.121	0.100	0.100	0.125	0.539	0.744	0.637
	BARO	0.048	0.222	0.212	0.133	0.333	0.290	0.400	**0.900**	0.583	0.615	0.949	0.776
	ART	0.095	0.127	0.177	0.133	0.289	0.242	0.600	0.700	0.708	0.256	0.564	0.486
LLM	Qwen2.5-14B	0.048	0.222	0.221	0.067	0.267	0.250	0.500	0.800	0.597	0.462	0.846	0.652
	GPT-4o	0.143	0.460	0.338	0.122	0.256	0.256	0.700	0.800	0.763	0.590	0.949	0.762
	o4-mini	0.079	0.175	0.168	0.056	0.133	0.162	0.700	0.800	0.754	0.820	0.949	0.891
	R1-14B	0.175	0.317	0.330	0.178	0.344	0.329	0.500	0.700	0.677	0.794	0.923	0.827
	R1-Full	0.254	0.413	0.383	0.200	0.467	0.426	0.700	0.800	0.811	0.785	0.929	0.859
	Doubao-Thinking	0.111	0.429	0.364	0.178	0.500	0.404	**0.800**	0.800	0.834	0.785	0.929	0.859
	RCA-Agent+R1-Full	0.032	-	-	0.178	-	-	0.700	-	-	0.462	-	-
	FoundRoot-14B	**0.429**	**0.619**	**0.569**	**0.433**	**0.744**	**0.610**	**0.800**	**0.900**	**0.858**	**0.886**	**0.984**	**0.931**
	Improvement	+68.9%	+34.6%	+48.6%	+116.5%	+48.8%	+43.2%	0%	0%	+2.9%	+8.0%	+3.7%	+4.5%



(a) Ablation Studies of Structured Deep Thinking **(b) Ablation Studies of RL**

Figure 8: Results of ablation studies. FoundRoot shows consistent improvements over models without structured deep thinking or deep thinking, and models trained without RL (WarmUp SFT Only, SFT + SFT). These results show the effectiveness of both structured deep thinking and the RL method.

remains relatively low. The main reason lies in system observability and metric quality. Systems with more comprehensive metric coverage, clearer component boundaries, and well-defined dependency graphs enable better RCA performance. In contrast, systems with sparse metrics and noisy data pose greater challenges. For instance, Dataset A comes from a complex service system with many components, making RCA inherently more difficult.

First, we observe that the performance of classical methods is generally poor. Especially in datasets A and B, classical methods can only achieve a Top-1 accuracy of 15.9%, which is much lower than the LLMs like R1-Full. One key limitation of classical methods is their reliance on numeric values for causal discovery and ranking. Thus, they often overlook the domain-specific semantics underlying the metrics. Additionally, real-world monitoring metrics frequently contain significant noise, which compromises the robustness of these methods. Among classical approaches, the recently proposed intervention-based algorithms, CIRCA and BARO, perform relatively better due to their more accurate modeling of fault propagation relationships. However, their modeling of causal relationships still exhibits many inaccuracies.

Compared to classical methods, LLM-based approaches without deep reasoning capabilities also fail to demonstrate significant advantages. Although these methods can infer the underlying domain-specific semantics based on metric names, their limited reasoning abilities make it difficult to analyze the complex failure propagation, achieving much lower MRRs from 0.036 to 0.151 on datasets A, B, C, and D compared with the LLMs with deep thinking, respectively. In contrast, models equipped with deep reasoning capabilities generally achieve better performance, indicating that the deep reasoning ability of LLMs is crucial for RCA tasks. However, as discussed above, even models with deep reasoning capabilities are primarily trained on math and code tasks. This limits their effectiveness in handling complex RCA scenarios and prevents them from fully leveraging their deep thinking potential.

Another noteworthy approach is the agent-based method. Agents can repeatedly execute code to analyze metrics step by step, which intuitively allows the LLM to focus more on critical metrics and

reduce reasoning hallucinations. However, based on the evaluation results, we find that this method does not significantly enhance the reasoning capabilities of LLMs in RCA tasks. On the contrary, its performance falls short compared to a series of models that directly apply deep thinking to RCA problems. This indicates that breaking down RCA reasoning into action-reflection processes does not significantly enhance the ability to reason about complex causal relationships. On the contrary, it often results in some severe hallucinations when analyzing fault propagation because of the lack of intrinsic reasoning capabilities. A detailed comparison of the workflow-based approach is provided in RQ2.

In summary, compared to these various types of baseline models, our proposed FoundRoot achieves outstanding performance, with 68.9%-116.5% Top-1 improvement on datasets A and B, and 4.5%-48.6% MRR improvement on all datasets. This demonstrates that the training approach introduced by FoundRoot effectively enhances the reasoning capabilities of LLMs for RCA tasks. However, accuracy in some of the datasets (A and B) is relatively low. The main reason lies in system observability and metric quality. For instance, datasets A and B come from complex service systems with many interdependent components, making RCA more difficult compared with datasets C and D.

4.3 RQ2: How effective is structured deep thinking for RCA?

To evaluate the contribution of structured deep thinking, we conduct ablation studies as shown in Figure 8. First, we compare FoundRoot with the *w/o Struct. Think* variant, which adopts the same two-stage SFT+RL pipeline as FoundRoot, but the warm-up SFT data only contains ordinary deep thinking traces rather than structured ones. The evaluation results in Figure 8a show that FoundRoot achieves consistent improvement compared with this variant. This is because the absence of structured substeps leads to incomplete reasoning or inconsistent failure propagation analysis. These results demonstrate that structured deep thinking plays a crucial role in enhancing the reasoning depth and robustness of zero-shot RCA. Moreover, we provide illustrative comparisons of reasoning traces in Section 4.7 to further support this conclusion.

To further illustrate the importance of deep thinking, we also show the results of *w/o Deep Think* variant, which is trained using the same two-stage pipeline and datasets as FoundRoot but based on a non-thinking model without thinking capability. As shown in Figure 8a, this variant performs significantly worse than FoundRoot, showing that the deep thinking paradigm itself substantially improves RCA reasoning quality.

We also implement a workflow-based method, which uses prompt engineering to achieve structured thinking format rather than training with RL. The RCA workflow [32] executes a predefined, human-designed sequence of prompt-based steps to transform the input into a final output. To achieve this, we use several prompts to guide the LLMs to generate the same substeps as in structured deep thinking. We employ DeepSeek-R1 as the base LLM for this workflow model (w/ Workflow). As shown in Figure 8a, the model with workflow exhibits certain advantages over the LLMs without structured deep thinking across all datasets (in Table 2), which demonstrates that breaking down reasoning steps mitigates shallow

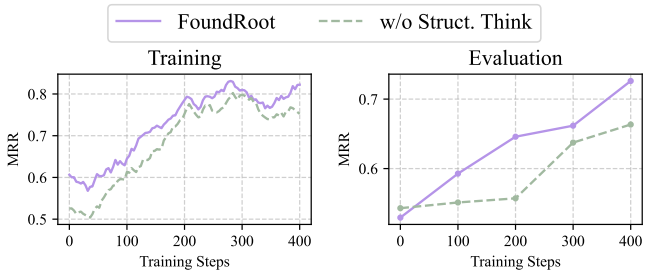


Figure 9: Overall MRR on training and evaluation datasets during training. FoundRoot is showing more generalization capability compared with one w/o structured thinking.

reasoning to some extent. However, the workflow-based model still fall short of FoundRoot on all of the datasets in terms of RCA accuracy. This is because simply decomposing the reasoning process through prompts cannot fully eliminate hallucinations or enforce coherent substep reflection. At each substep, the LLM must still rely on its *intrinsic reasoning* capability to analyze failure propagation accurately. This highlights that the SFT+RL approach, which embeds structured deep thinking as an inherent capability, is essential for enhancing the model's intrinsic reasoning abilities. Overall, our results indicate that integrating structured deep thinking *directly* into the model through RL is effective and generalizable.

4.4 RQ3: Does RL really improve reasoning performance?

To investigate this question, we replace the two-stage SFT + RL pipeline with a one-stage *WarmUp SFT Only* version and a two-stage *SFT + SFT* version which replaced the RL training with SFT training using the same datasets. As shown in Figure 8b, the two SFT-based variants exhibit very different behaviors. The *WarmUp SFT Only* model acquires basic structured deep thinking patterns from the WarmUp SFT datasets, which enables it to generate partially valid reasoning traces. However, without the RL stage, the model tends to focus on reproducing template-like reasoning structures and often fails to capture the causal dependencies among metrics, resulting in limited RCA accuracy on all datasets.

Similarly, the *SFT + SFT* variant performs even worse. Since the second-stage SFT dataset contains only fixed-format structured JSON with highly deterministic phrasing, the model is exposed to little variation or exploratory reasoning. This causes the second-stage SFT to learn format imitation only rather than to deepen the model's causal reasoning ability. As a result, the model gradually loses much of its RCA capability, leading to consistently low Top-1, Top-3, and MRR scores. In comparison, our *SFT + RL (FoundRoot)* model benefits from RL, which explicitly rewards reasoning completeness, structural consistency, and correct causal ranking, allowing the model to refine and strengthen the structured deep thinking behavior learned during warm-up SFT. This enables FoundRoot to move beyond format reproduction and develop more accurate, coherent, and generalizable RCA reasoning in unseen systems.

4.5 RQ4: Does structured deep thinking provide more generalization to unseen systems?

We compared the training and evaluation curves of FoundRoot and the variant without structured deep thinking (w/o Struct. Think), as shown in Figure 9. It can be observed that both models exhibit a significant upward trend in MRR on the training dataset as training progresses, with similar rates of increase. However, on the evaluation set, FoundRoot achieves a substantially greater improvement compared to the model without structured deep thinking.

The *training curve* of FoundRoot starts slightly higher than that of the model w/o structured deep thinking. However, the two models exhibit nearly the same initial performance on the *evaluation curve*. Therefore, the performance gains of FoundRoot primarily come from the RL phase, where structured deep thinking contributes to these improved results.

4.6 RQ5: How do smaller and quantized variants perform?

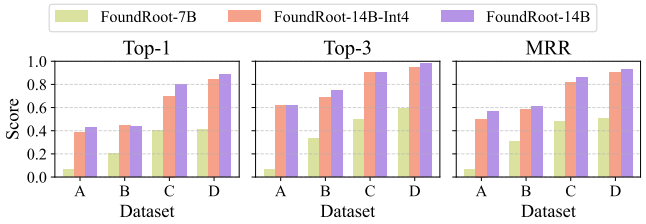


Figure 10: Study of smaller and quantized FoundRoot models.

To support easier and more cost-efficient deployment, we train a 7B variant (FoundRoot-7B) and a GPTQ-Int4 [6] quantized version of the 14B model (FoundRoot-14B-Int4). As shown in Figure 10, the 7B model shows noticeable performance degradation compared with 14B, especially on more complex datasets A and B. This shows that a 7B LLM may not be sufficient for complex reasoning in RCA. In contrast, the quantized GPTQ-Int4 version of the 14B model achieves much better performance than the 7B model and maintains performance comparable to the original 14B model.

4.7 Case Study

In Figure 11, we compare the reasoning processes among FoundRoot, R1-14B (base model of FoundRoot), and the R1-14B with Workflow. In this case, we show a failure case where the root cause is `os_021`. FoundRoot first analyzes all metrics in the Metric Scanning substep. In the reflection step, FoundRoot reflects on the correlation of `os_021##proc` and the drops in `docker_003`. It correctly reasons that `docker_003` depends on `os_021`, and thus the failure in `os_021` leads to downstream failures. Finally, FoundRoot correctly identified `os_021` as the root cause.

In contrast, the R1-14B fails to consider `os_021` at all in its entire deep thinking process. It finally selects `docker_001` as the possible root cause based on the incomplete analysis of the metrics. This is a typical case of an error caused by missing key metrics. Similarly, the model with workflow ignores the fact that `os_021` hosts `docker_003` and incorrectly attributes the root cause to `docker_003`.

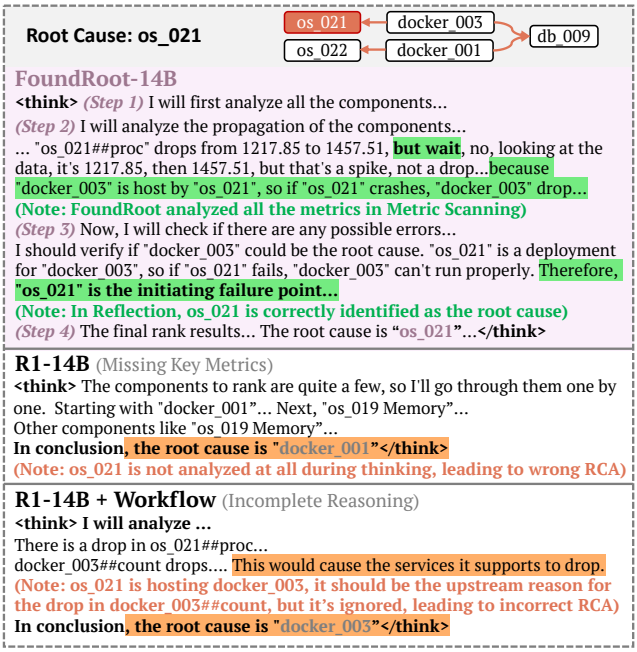
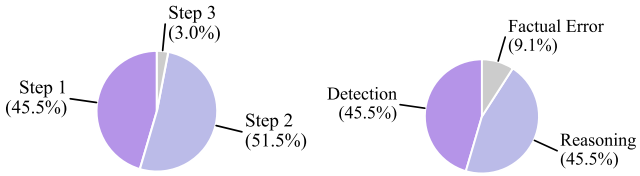


Figure 11: Case study on FoundRoot and baseline methods. Compared with the R1-14B and Workflow models, FoundRoot achieves better performance by comprehensively analyzing metrics and better causal reasoning.

This case shows the missing key metrics and incomplete causal reasoning in existing LLMs. Therefore, the structured deep thinking in FoundRoot demonstrates its superior reasoning capability for RCA by incorporating complete analysis on the large number of metrics and its reflection on the causal relationship.

4.8 Analysis of failure cases



(a) First Error Substep **(b) Error Reason**

Figure 12: Statistical analysis of error cases produced by FoundRoot. Most errors occur in substep 1 and 2. In particular, many root cause metrics are not recognized by the model as anomalous metrics, which leads to errors.

To further understand the limitations of FoundRoot, we conduct a detailed analysis of its failure cases, as illustrated in Figure 12. The results reveal that most errors arise during the first two reasoning substeps. Specifically, in substep 1 (Metric Scanning), the model frequently fails to **detect** the true root cause metrics as anomalous. In substep 2 (Propagation Analysis), the model tends to make logical mistakes when **reasoning** about the detected anomalies and

their causal relationships, which can be attributed to the limited reasoning capacity of the 14B LLM. This suggests that improving the model's capability to accurately detect anomalous metrics and enhancing its reasoning ability are promising directions.

5 Related Work

Classical RCA methods. Classical metric-based methods usually consist of two steps: graph construction and ranking. For graph construction, some adopt statistical causal discovery algorithms such as the PC algorithm [24, 39, 44], while others improve robustness by addressing propagation delay [29], grouping correlated metrics [26], or using a physical dependency graph [19]. In the ranking phase, a variety of techniques have been proposed, including DFS traversal [4], random walk [44], PageRank [46], hypothesis testing [33, 36], reconstruction-based [40], and intervention-based inference [12, 17, 48]. Beyond metrics, event-based methods [43, 50], log-based [2], and trace-based methods [25] leverage additional observability signals to improve precision but often assume complete instrumentation. Recent efforts also explore heterogeneous RCA [8, 21], though they typically require strong supervision or expert rules, limiting adaptability in dynamic environments.

LLM-based RCA methods. With the rise of LLMs, leveraging their advantages for RCA has been increasingly adopted by a growing number of researchers. Many studies focus on enabling LLMs to understand logs [10, 18, 27, 28, 37] for root cause localization, to fully leverage the rich semantic information contained in logs. Meanwhile, to enable LLMs to understand metric information, some agent-based approaches have been proposed for metric-based or multimodal data-based RCA [32, 34, 38, 47, 49, 54]. However, due to the complexity of RCA, root cause analysis requires extensive and sophisticated reasoning. Existing models have addressed some issues from an engineering perspective. However, limited research focuses on the enhancement of the reasoning capabilities for RCA.

Reinforcement Learning for LLM. Recent efforts have enhanced LLM reasoning by applying reinforcement learning [7, 31, 41]. To improve training stability and sampling efficiency, algorithms such as GRPO [9] have been proposed, which is further extended by DAPO [51], which refines reward shaping and policy optimization for long-form reasoning tasks. While these methods primarily target math and code domains, LLM reasoning for RCA based on time series has been scarcely studied. Our FoundRoot successfully adapts DAPO to the RCA setting through structured reasoning, offering a new direction for the application of LLMs in the RCA domain.

6 Threats to Validity

The main threat to the *internal validity* of this study lies in the implementation of the baseline methods. 3 out of 14 baselines (MicroScope, MonitorRank, and MicroCause) are reimplemented by us based on their original papers due to the lack of publicly available code. For all other methods, we use the official implementations and parameter settings. Although we followed the described algorithm as closely as possible, potential discrepancies may exist.

The threats to *external validity* primarily come from the selection of evaluation datasets. While our datasets are diverse and include both different types of systems, they may not fully represent the breadth of failure patterns seen in all real-world systems. To alleviate these threats, we collected as many evaluations as possible from different systems to evaluate the cross-system generalization.

7 Limitations and Future Work

Although FoundRoot demonstrates strong performance across multiple RCA datasets, several limitations remain. First, FoundRoot is trained and evaluated using metrics data only. While this modality has the advantages of universality, it may miss useful signals available in logs, traces, or configuration metadata [40, 49].

Second, our structured thinking format is designed based on domain knowledge. While it aligns with expert reasoning patterns, it may not fully reflect the diversity of RCA workflows in practice. Developing adaptive or learnable formats that evolve with new data or feedback is a promising direction.

Third, the DAPO is computationally expensive and difficult to implement. However, DAPO is only used during the training stage and does not introduce any additional cost during deployment. It also simplifies data preparation because it only requires metric time series and RCA labels.

Finally, although we evaluate our model on multiple real-world datasets, large-scale industrial deployment would require additional evaluations in production-grade environments. For practitioners, the rather low accuracy for datasets A and B suggests that FoundRoot should be used as a *decision-support* tool rather than a fully automated solution. In systems with lower absolute accuracy, FoundRoot can significantly narrow down the search space and provide interpretable reasoning traces to assist human operators. Furthermore, improving system observability by adding more informative metrics can enhance FoundRoot's effectiveness.

8 Conclusion

In this paper, we present FoundRoot, a foundation model designed for zero-shot RCA based on LLMs with metrics data. To address the incomplete and shallow reasoning and poor generalization in unseen systems, we introduce a novel structured deep thinking method for systematic analysis of the failure propagation. Structured deep thinking breaks down the RCA process into explicit reasoning steps, enabling the model to capture causal semantics and failure propagation patterns more effectively. To support the training of FoundRoot, we curate and clean ten diverse RCA datasets from real-world systems and develop a data augmentation strategy that ensures both data diversity and domain coverage. By leveraging a 2-stage training pipeline with warm-up SFT + DAPO built upon structured deep thinking, we demonstrate for the first time in research that RL can significantly enhance the reasoning and generalization capabilities of LLMs for zero-shot RCA tasks. Experimental results on four zero-shot benchmarks show that FoundRoot consistently outperforms both classical RCA algorithms and strong LLM-based baselines, achieving up to a 48.6% improvement in MRR. We believe that FoundRoot establishes a solid foundation for future research in LLM-based reasoning for RCA tasks.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No.2024YFB4505903).

References

[1] Ryan Prescott Adams and David JC MacKay. 2007. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742* (2007).

[2] Pooja Aggarwal, Ajay Gupta, Prateeti Mohapatra, Seema Nagar, Atri Mandal, Qing Wang, and Amit Paradkar. 2021. Localization of Operational Faults in Cloud Applications by Mining Causal Dependencies in Logs Using Golden Signals. In *Service-Oriented Computing – ICSOC 2020 Workshops*, Hakim Hacid, Fatma Outay, Hye-young Paik, Amira Alloum, Marinella Petrocchi, Mohamed Reda Bouadjenek, Amin Beheshti, Xumin Liu, and Abderrahmane Maaradji (Eds.). Vol. 12632. Springer International Publishing, Cham, 137–149. doi:10.1007/978-3-030-76352-7_17

[3] Vic Barnett, Toby Lewis, et al. 1994. *Outliers in statistical data*. Vol. 3. Wiley New York.

[4] Pengfei Chen, Yong Qi, Pengfei Zheng, and Di Hou. 2014. CauseInfer: Automatic and Distributed Performance Diagnosis with Hierarchical Causality Graph in Large Distributed Systems. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. IEEE, Toronto, ON, Canada, 1887–1895. doi:10.1109/INFOCOM.2014.6848128

[5] CloudWise. 2021. GAIA Dataset. https://github.com/CloudWise-OpenSource/GAIA-DataSet.

[6] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).

[7] Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. 2024. On designing effective rl reward at training time for llm reasoning. *arXiv preprint arXiv:2410.15115* (2024).

[8] Shenghui Gu, Guoping Rong, Tian Ren, He Zhang, Yongda Yu, Xian Li, Jian Ouyang, and Chunan Chen. 2023. TrinityRCL: Multi-Granular and Code-Level Root Cause Localization Using Multiple Types of Telemetry Data in Microservice Systems. *IEEE Transactions on Software Engineering* 49, 5 (May 2023), 3071–3088. doi:10.1109/TSE.2023.3241299

[9] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[10] Adha Hrusto, Per Runeson, and Magnus C Ohlsson. 2024. Autonomous monitors for detecting failures early and reporting interpretable alerts in cloud operations. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*. 47–57.

[11] Hugging Face. 2025. Open R1: A fully open reproduction of DeepSeek-R1. https://github.com/huggingface/open-r1

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

[13] Myunghwan Kim, Roshan Sumbaly, and Sam Shah. 2013. Root Cause Detection in a Service-Oriented Architecture. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*. ACM, Pittsburgh PA USA, 93–104. doi:10.1145/2465529.2465753

[14] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

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

[16] Mingjie Li, Zeyan Li, Kanglin Yin, Xiaohui Nie, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2022. Causal inference-based root cause analysis for online service systems with intervention recognition. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 3230–3240.

[17] Mingjie Li, Zeyan Li, Kanglin Yin, Xiaohui Nie, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2022. Causal Inference-Based Root Cause Analysis for Online Service Systems with Intervention Recognition. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, Washington DC USA, 3230–3240. doi:10.1145/3534678.3539041

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

[19] Ye Li, Jian Tan, Bin Wu, Xiao He, and Feifei Li. 2023. Shapleyiq: Influence quantification by shapley values for performance debugging of microservices. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 4*. 287–323.

[20] Zeyan Li, Junjie Chen, Rui Jiao, Nengwen Zhao, Zhijun Wang, Shuwei Zhang, Yanjun Wu, Long Jiang, Leiqin Yan, Zikai Wang, Zhekang Chen, Wenchi Zhang, Xiaohui Nie, Kaixin Sui, and Dan Pei. 2021. Practical Root Cause Localization for Microservice Systems via Trace Analysis. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, Tokyo, Japan, 1–10. doi:10.1109/IWQOS52092.2021.9521340

[21] Zeyan Li, Nengwen Zhao, Mingjie Li, Xianglin Lu, Lixin Wang, Dongdong Chang, Xiaohui Nie, Li Cao, Wenchi Zhang, Kaixin Sui, et al. 2022. 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*. 996–1008.

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

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

[24] Jinjin Lin, Pengfei Chen, and Zibin Zheng. 2018. Microscope: Pinpoint Performance Issues with Causal Graphs in Micro-service Environments. In *Service-Oriented Computing*, Claus Pahl, Maja Vukovic, Jianwei Yin, and Qi Yu (Eds.). Vol. 11236. Springer International Publishing, Cham, 3–20.

[25] Dewei Liu, Chuan He, Xin Peng, Fan Lin, Chenxi Zhang, Shengfang Gong, Ziang Li, Jiayu Ou, and Zheshun Wu. 2021. MicroHECL: High-Efficient Root Cause Localization in Large-Scale Microservice Systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, Madrid, ES, 338–347. doi:10.1109/ICSE-SEIP52600.2021.00043

[26] Xianglin Lu, Zhe Xie, Zeyan Li, Mingjie Li, Xiaohui Nie, Nengwen Zhao, Qingyang Yu, Shenglin Zhang, Kaixin Sui, Lin Zhu, and Dan Pei. 2022. Generic and Robust Performance Diagnosis via Causal Inference for OLTP Database Systems. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, Taormina, Italy, 655–664. doi:10.1109/CCGrid54584.2022.00075

[27] Lipeng Ma, Weidong Yang, Yixuan Li, Ben Fei, Mingjie Zhou, Shuhao Li, Sihang Jiang, Bo Xu, and Yanghua Xiao. 2025. AdaptiveLog: An Adaptive Log Analysis Framework with the Collaboration of Large and Small Language Model. *arXiv preprint arXiv:2501.11031* (2025).

[28] Lipeng Ma, Weidong Yang, Bo Xu, Sihang Jiang, Ben Fei, Jiaqing Liang, Mingjie Zhou, and Yanghua Xiao. 2024. Knowlog: Knowledge enhanced pre-trained language model for log understanding. In *Proceedings of the 46th ieee/acm international conference on software engineering*. 1–13.

[29] Yuan Meng, Shenglin Zhang, Yongqian Sun, Ruru Zhang, Zhilong Hu, Yiyin Zhang, Chenyang Jia, Zhaogang Wang, and Dan Pei. 2020. Localizing Failure Root Causes in a Microservice through Causality Inference. In *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. IEEE, Hang Zhou, China, 1–10. doi:10.1109/IWQoS49365.2020.9213058

[30] Radford M Neal. 2003. Slice sampling. *The annals of statistics* 31, 3 (2003), 705–767.

[31] OpenAI. [n. d.]. Learning to Reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/

[32] Changhua Pei, Zexin Wang, Fengrui Liu, Zeyan Li, Yang Liu, Xiao He, Rong Kang, Tieying Zhang, Jianjun Chen, Jianhui Li, et al. 2025. Flow-of-Action: SOP Enhanced LLM-Based Multi-Agent System for Root Cause Analysis. In *Companion Proceedings of the ACM on Web Conference 2025*. 422–431.

[33] Luan Pham, Huong Ha, and Hongyu Zhang. 2024. BARO: Robust Root Cause Analysis for Microservices via Multivariate Bayesian Online Change Point Detection. *Proc. ACM Softw. Eng.* 1, FSE, Article 98 (jul 2024), 24 pages. doi:10.1145/3660805

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

[35] Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. 2019. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science advances* 5, 11 (2019), eaau4996.

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

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

[38] Binpeng Shi, Yu Luo, Jingya Wang, Yongxin Zhao, Shenglin Zhang, Bowen Hao, Chenyu Zhao, Yongqian Sun, Zhi Zhang, Ronghua Sun, et al. 2025. FlowXpert: Expertizing Troubleshooting Workflow Orchestration with Knowledge Base and Multi-Agent Coevolution. (2025).

[39] Peter Spirtes, Clark N Glymour, and Richard Scheines. 2000. *Causation, prediction, and search*. MIT press.

[40] Yongqian Sun, Binpeng Shi, Mingyu Mao, Minghua Ma, Sibo Xia, Shenglin Zhang, and Dan Pei. 2024. Art: A unified unsupervised framework for incident management in microservice systems. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 1183–1194.

[41] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1. 5:

Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599* (2025).

[42] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. TRL: Transformer Reinforcement Learning. https://github.com/huggingface/trl.

[43] Hanzhang Wang, Zhengkai Wu, Huai Jiang, Yichao Huang, Jiamu Wang, Selcuk Kopru, and Tao Xie. 2021. Groot: An event-graph-based approach for root cause analysis in industrial settings. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 419–429.

[44] Ping Wang, Jingmin Xu, Meng Ma, Weilan Lin, Disheng Pan, Yuan Wang, and Pengfei Chen. 2018. CloudRanger: Root Cause Identification for Cloud Native Systems. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, Washington, DC, USA, 492–502. doi:10.1109/CCGRID.2018.00076

[45] Junde Wu, Jiayuan Zhu, and Yuyuan Liu. 2025. Agentic Reasoning: Reasoning LLMs with Tools for the Deep Research. *arXiv preprint arXiv:2502.04644* (2025).

[46] Li Wu, Johan Tordsson, Jasmin Bogatinovski, Erik Elmroth, and Odej Kao. 2021. MicroDiag: Fine-grained Performance Diagnosis for Microservice Systems. In *2021 IEEE/ACM International Workshop on Cloud Intelligence (CloudIntelligence)*. IEEE, Madrid, Spain, 31–36. doi:10.1109/CloudIntelligence52565.2021.00015

[47] Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. 2024. Chatts: Aligning time series with llms via synthetic data for enhanced understanding and reasoning. *arXiv preprint arXiv:2412.03104* (2024).

[48] Zhe Xie, Shenglin Zhang, Yitong Geng, Yao Zhang, Minghua Ma, Xiaohui Nie, Zhenhe Yao, Longlong Xu, Yongqian Sun, Wentao Li, et al. 2024. Microservice root cause analysis with limited observability through intervention recognition in the latent space. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6049–6060.

[49] Junjielong Xu, Qinan Zhang, Zhiqing Zhong, Shilin He, Chaoyun Zhang, Qingwei Lin, Dan Pei, Pinjia He, Dongmei Zhang, and Qi Zhang. [n. d.]. OpenRCA: Can Large Language Models Locate the Root Cause of Software Failures?. In *The Thirteenth International Conference on Learning Representations*.

[50] Guangba Yu, Pengfei Chen, Yufeng Li, Hongyang Chen, Xiaoyun Li, and Zibin Zheng. 2023. Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 553–565.

[51] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476* (2025).

[52] Haochuan Zhang, Chunhua Yang, Jie Han, Liyang Qin, and Xiaoli Wang. 2025. TempoGPT: Enhancing temporal reasoning via quantizing embedding. *arXiv e-prints* (2025), arXiv–2501.

[53] Shenglin Zhang, Sibo Xia, Wenzhao Fan, Binpeng Shi, Xiao Xiong, Zhenyu Zhong, Minghua Ma, Yongqian Sun, and Dan Pei. 2024. Failure diagnosis in microservice systems: A comprehensive survey and analysis. *ACM Transactions on Software Engineering and Methodology* (2024).

[54] Xuanhe Zhou, Guoliang Li, Zhaoyan Sun, Zhiyuan Liu, Weize Chen, Jianming Wu, Jiesi Liu, Ruohang Feng, and Guoyang Zeng. 2023. D-bot: Database diagnosis system using large language models. *arXiv preprint arXiv:2312.01454* (2023).