RCAFlow: A Workflow-Informed Hierarchical Planning Multi-Agent System for Root Cause Analysis

Yufei Gao, Zhengong Cai*, Bowei Yang

Zhejiang University
{yufeigao,cstcaizg,boweiy}@zju.edu.cn

Abstract

As microservice architectures become increasingly complex and system events become more frequent, Root Cause Analysis (RCA) has emerged as a critical task to ensure system reliability. However, existing deep learning-based methods often struggle with limited flexibility and a lack of interpretability when addressing complex system failures. Recent efforts to integrate large language models (LLMs) have shown promise in enhancing diagnostic transparency and reasoning capability. However, expansive search spaces, intricate workflows, and entangled constraints constrain practical adoption. We propose RCAFlow, a multi-agent framework that integrates structured workflow knowledge with hierarchical planning to address these challenges. RCAFlow transforms semi-structured documents into behavior tree-style workflows to support interpretable plan generation, employs a Git-inspired branching mechanism for modular and hierarchical task execution with path isolation, and leverages state-aware task execution with semantic analysis to improve result understanding and feedback. We evaluate RCAFlow on three benchmark datasets provided by OpenRCA. Experimental results demonstrate that RCAFlow consistently outperforms existing methods across all datasets. Further ablation studies confirm the effectiveness of each core module, highlighting the reliability, extensibility, and interpretability of RCAFlow to support complex RCA tasks within intelligent IT operations.

Introduction

During the past decade, with the rapid development of cloud computing, more enterprises have migrated from monolithic architectures to large-scale cloud service platforms to achieve greater scalability, agility, and reliability. Despite the growing maturity of cloud ecosystems, operations still frequently encounter system events such as outages, performance degradation, and repeated alarms. These incidents impact service availability and user experience, resulting in financial losses. Cloud platforms have widely adopted monitoring and alerting systems to ensure service stability, enabling real-time health monitoring and rapid response. However, the surge in event volume imposes growing diagnostic pressure on engineers. In microservice architectures, the complex inter-service dependencies make Root Cause Analysis (RCA) the most critical and challenging part of the incident response process. (Nguyen et al. 2013; Aggarwal et al. 2020; Sun et al. 2024; Xie et al. 2024a).

To effectively address system events, researchers have proposed various learning-based methods to mitigate the complexity of manual diagnosis. These approaches typically employ deep learning to extract diagnostic patterns from historical failure data and utilize techniques such as causal discovery (Lin, Chen, and Zheng 2018; Pham, Ha, and Zhang 2024), graph neural networks (Yen et al. 2022; Wang et al. 2023) and fault localization (Xu et al. 2022; Han et al. 2025) to improve modeling of causal chains in complex systems. However, they often rely on predefined workflows, causal graphs, static fault-symptom models, or fixed associations learned from data, resulting in one-off analyses with limited adaptability to evolving system or novel failure scenarios.

In recent years, with the development of LLMs, researchers are incorporating them into RCA as intelligent agents (Ahmed et al. 2023a; Wang et al. 2024; Xu et al. 2025). Leveraging their strong language understanding capabilities, LLMs can generate traceable paths, improving transparency and reliability of analysis process. To further enhance their capacity to handle large numbers of APIs and complex data, such as service dependencies and performance metrics, some studies have introduced multi-agent systems (Zhang et al. 2024a). In contrast, others leverage semantic retrieval to incorporate historical incidents, troubleshooting guides (TSGs) (Jiang et al. 2020; Ghosh et al. 2022), or standard operating procedures (SOPs) as domain knowledge for LLMs (Pei et al. 2025; Goel et al. 2025). However, these methods rely on static flows, where agents follow predefined action sequences without adapting to intermediate feedback. Integrating structured knowledge guidance with autonomous planning capabilities may significantly improve the system's flexibility and scalability.

Moreover, in RCA tasks, existing methods (Ahmed et al. 2023a; Wang et al. 2024) often adopt ReAct-style (Yao et al. 2023) paradigms, where tool invocations and their outputs are directly embedded into the reasoning chain. While effective in simple contexts, this strategy tends to introduce low-level noise when applied to complex systems and multi-stage diagnostic workflows, thereby interfering with identifying and modeling critical causal paths (Kim et al.

*Corresponding author

2025). Since RCA typically involves synthesizing multiple anomaly signals, the reasoning process is susceptible to the accuracy of each anomaly detection step. This undermines the model's ability to capture the global causal structure and makes it difficult to track the completion of intermediate tasks, often leading to logical inconsistencies during multi-turn interactions. Introducing finer-grained task management mechanisms can significantly improve the quality and consistency of RCA decision-making.

To address these limitations, we propose RCAFlow, a novel Multi-Agent framework that, to our knowledge, is the first to integrate workflow structures and hierarchical planning into LLM-based RCA systems. RCAFlow combines the general reasoning capabilities of LLMs with the domain knowledge encoded in structured workflows, enabling hierarchical task planning and task management. This design aims to mitigate hallucination and instability in complex reasoning, while enhancing the robustness and interpretability of the RCA decision process.

Specifically, we transform structured knowledge such as Troubleshooting Guides (TSGs) and Standard Operating Procedures (SOPs) into three types of workflow to construct a knowledge base that supports task planning. The hierarchical planning process is jointly driven by *Planner* and the Workflow Knowledge Base (WKB), and is organized and controlled through a Git-inspired branch management mechanism. *Brancher* creates and merges branches, while *Coordinator* manages global task scheduling. Based on the output of *Planner*, tasks are either delegated to the next-level branch or directly executed by *Executor* following the planned steps. Upon execution, the system can perceive the current status of the branch task, enabling dynamic adjustment of reasoning paths and strategy refinement. In addition, the system records the complete reasoning trajectory and execution results in the current branch's workspace, enabling data access and process traceability for other branches.

Our contributions can be summarized as follows:

1. We propose RCAFlow, a multi-agent framework that integrates structured knowledge guidance, hierarchical task planning, modular execution, and self-reflection and is tailored for RCA in complex, heterogeneous cloud systems.

2. We construct a structured workflow knowledge base by converting troubleshooting documents into behavior-tree-style processes enriched with control logic and tool metadata, significantly improving the stability and interpretability of LLM reasoning.

3. We design a Git-inspired branching mechanism and hierarchical coordination to decouple reasoning paths, support modular execution, and enable state-aware, dynamically adjustable reasoning.

4. RCAFlow achieves state-of-the-art performance on the challenging OpenRCA benchmark, demonstrating its effectiveness in complex root cause analysis.

Related Works

LLM-Based Root Cause Analysis Agents

The development of LLMs has enabled agents to invoke tools and perform RCA tasks in complex environments autonomously (Ahmed et al. 2023b; Chen et al. 2024; Roy et al. 2024; Wang et al. 2024). To address the complexity of multi-modal data, some studies have introduced multi-agent systems that collaborate to extract information and perform reasoning, advancing the automation of RCA. For example, mABC (Zhang et al. 2024a) adopts multi-agent blockchain-inspired collaboration, while AgentFM (Zhang et al. 2025) models systems and task roles, with a meta-agent coordinating multiple sub-agents. Although multi-agent approaches enhance collaborative processing, they still face challenges such as large search spaces, low tool selection efficiency, and limited adaptability when dealing with heterogeneous APIs and complex contexts. To mitigate these challenges, another line of work introduces knowledge-guided methods: Flow-of-Action (Pei et al. 2025) constrains agent behavior using SOPs; ICL RCA (Zhang et al. 2024b) and eARCO (Goel et al. 2025) build knowledge bases from unstructured TSGs and historical incidents to provide contextual cues and action suggestions; and Atlas (Xie et al. 2024b) constructs causal graphs from system documentation. RCA tasks are highly procedural and enriched with expert knowledge, allowing agents to adapt efficiently under knowledge guidance without requiring full autonomous exploration. However, existing methods rely on static workflows and cannot dynamically adjust or generate feedback-based strategies. To address this, RCAFlow proposes a hierarchical and decoupled planning framework that integrates a behavior-tree-style workflow knowledge base and a coordination mechanism, enabling strategy adjustment and thus demonstrating stronger adaptability in complex systems.

Hierarchical Planning with LLMs

In recent years, increasing attention has been paid to applying LLMs to hierarchical task planning for complex long-term tasks. These methods typically adopt a two-level architecture—high-level planning and low-level execution—recursively decomposing tasks into executable atomic actions for efficient completion (Sun et al. 2023; Prasad et al. 2024). This paradigm has been widely applied across various domains, such as web automation with Rada (Kim et al. 2024) and long-form text generation with the HRP system (Xiong et al. 2025).

RCAFlow introduces a hierarchical planning mechanism to enhance the system's structured reasoning and decision-making capability in complex tasks. The system adopts a Git-inspired branch-based task management mechanism to manage multi-level reasoning processes efficiently, supporting task isolation, modular scheduling, and cross-branch information sharing. Given that multi-modal anomaly detection results in RCA tasks can all impact the final root cause localization, and most subtasks lack clear completion boundaries—making it difficult to determine whether they are sufficiently completed—each subtask must be analyzed as precisely as possible to avoid incomplete reasoning affecting overall diagnostic accuracy. To address this, the system enables subtasks to dynamically perceive task status and adjust reasoning paths based on real-time feedback, improving adaptability in complex scenarios.

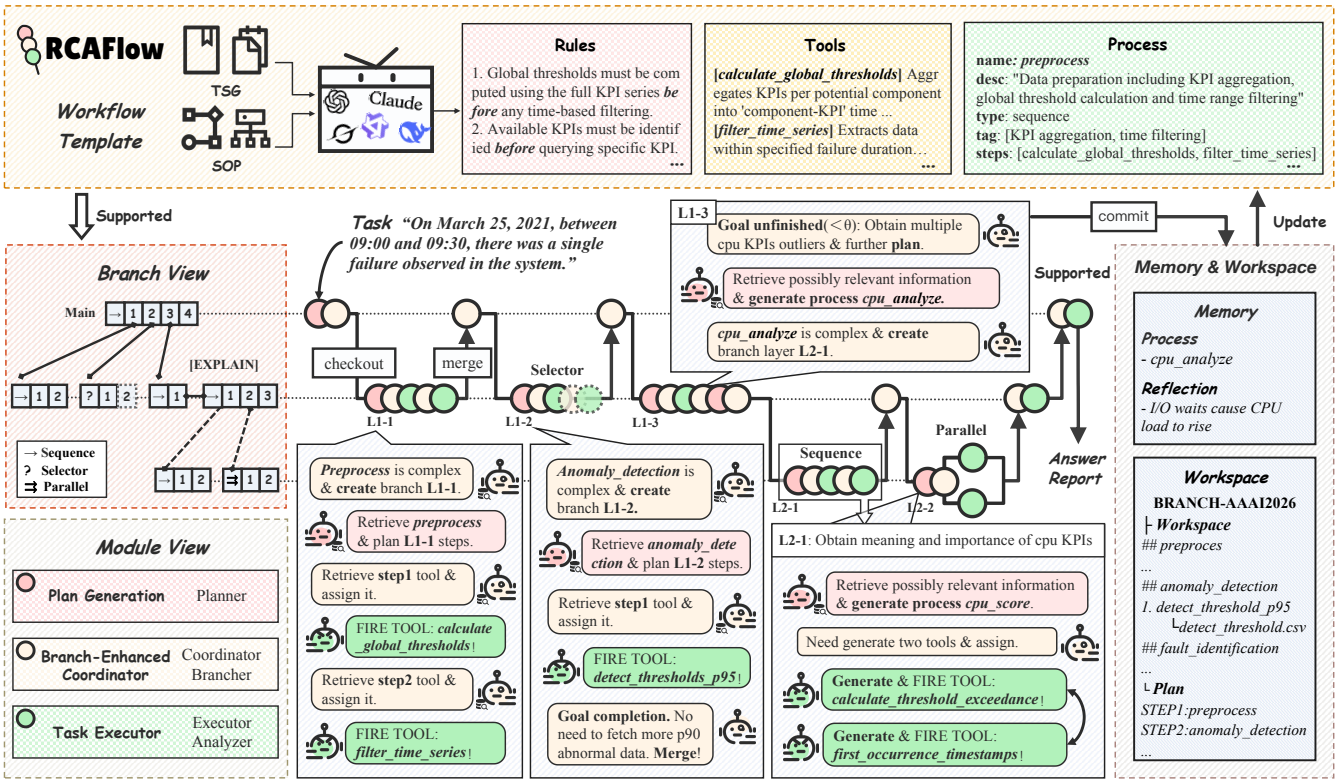

Figure 1: Overview of the RCAFlow Framework.

Method

Problem Formulation

Formally, given a complex RCA task $\mathcal{T}$ and an external environment $\mathcal{E}$, our goal is to build a system $\mathcal{A}$ that generates the final root cause explanation along with its corresponding reasoning trajectory. $\mathcal{A}$ multi-agent system is a tuple

$$\Sigma_{\mathcal{A}} = (\mathcal{A}, \mathcal{K}, \mathcal{O}, \mathcal{M}, \mathcal{W}), \tag{1}$$

where $\mathcal{A}$ is the multi-agent system, $\mathcal{K}$ is the structured knowledge base, $\mathcal{O}$ is the input-output interface that provides access to multimodal information from the environment $\mathcal{E}$ (e.g., logs, metrics, and trace data), as well as the input task $\mathcal{T}$, $\mathcal{M}$ is the internal memory module for storing intermediate reasoning states and knowledge, and $\mathcal{W}$ is the workspace that maintains the execution trajectories of branches. Upon receiving the task $\mathcal{T}$, the system may decompose it into a set of subtasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$, and incrementally construct a reasoning trajectory $\mathcal{R} = \{r_1, r_2, \ldots, r_k\}$ with the support of $\mathcal{K}$ and $\mathcal{O}$, ultimately producing a complete and interpretable RCA result.

Overview of the RCAFlow Framework

In this section, we introduce the RCAFlow framework, as depicted in Figure 1, which consists mainly of three modules: (1) **Workflow-Informed Task Planning**: This module leverages structured workflows to guide the generation of interpretable hierarchical reasoning plans; (2) **Branch-Enhanced Hierarchical Coordinator**: Designed for procedural control, this module incorporates Git-inspired branching mechanisms to enable hierarchical management and flexible scheduling of tasks and (3) **State-Aware Task Execution**: This module can generate or invoke existing tools for data processing and analysis, with the additional ability to assess the execution state and task completion status.

Workflow-Informed Task Planning

In RCA tasks, agents must rapidly identify issues within highly dynamic and heterogeneous failure signals. Relying on scattered tools or heuristic rules often leads to a search space explosion and unstable decision paths. To address this, we construct a structured Workflow Knowledge Base $\mathcal{K}$ that encodes operation processes, rules, and tools in an explicit format to guide LLM agents during reasoning.

Structured Workflow Definition The $\mathcal{K}$ in this work is structured into three components (Xiao et al. 2024):

Operation process related: This component describes the necessary steps and their execution order for task completion. Given the RCA tasks' inherent complexity and uncertainty, we adopt the Behavior Tree (Colledanchise and Ögren 2018) paradigm as the organizational framework for process control logic. BTs are well-suited for diagnostic scenarios involving incomplete information and multiple coexisting paths. They overcome the limitations of traditional linear if-else structures in terms of expressiveness and scalability, while also supporting real-time monitoring of runtime states. Specifically, we utilize the following three types of

control nodes to orchestrate task execution hierarchically:

Sequence: Executes all child nodes sequentially; succeeds only if all succeed. Suitable for strongly dependent task chains.

Selector: Attempts child nodes in order; succeeds if any one succeeds. Suitable for multi-strategy exploration.

Parallel: Executes child nodes concurrently; completion is determined by an aggregation condition. Suitable for multi-source data fusion.

Condition/rule related: Define explicit preconditions and exception handling mechanisms for each workflow node to standardize valid state transitions.

Tool/data related: Encapsulate atomic operational capabilities as callable nodes with standardized APIs and data dependencies, enabling a decoupled design between reasoning logic and low-level execution.

Through the above definitions, the structured workflow can be represented as a tuple $\mathcal{K} = (\mathcal{K}_p, \mathcal{K}_r, \mathcal{K}_t)$, systematically organizing previously fragmented knowledge into a knowledge base that supports the modeling of task execution logic, the specification of rules mechanisms, and the encapsulation and orchestration of execution capabilities.

Automated Workflow Generation RCA knowledge typically originates from semi-structured documents such as TSGs and SOPs. These documents encode domain knowledge but are difficult to retrieve due to their loose structure. RCAFlow incorporates LLMs to transform such documents into structured workflows to address this automatically.

Our system leverages LLMs to automatically extract key task elements, control logic, and tool metadata from fault diagnosis rules, organizing them into JSON-based workflow graphs that conform to a predefined schema. The RCA process is modeled as a hierarchical and composable structure, where each node represents either a tool or a nested sub-workflow, enabling modularity, reusability, and efficient updates when rules evolve. The system further extracts node-level preconditions, execution rules, and dependencies, and performs semantic validation to ensure logical consistency and executability. It also supports automatic generation of names and descriptions for undocumented tools based on context, enabling dynamic code generation within RCAFlow and contributing to a high-quality, searchable knowledge base.

Reasoning Plan Generation Given the current subtask $\mathcal{T}_i$, *Planner* in RCAFlow generates an executable and interpretable plan trajectory $\pi_i = \{s_1^i, s_2^i, \ldots, s_n^i\}$ by leveraging the current context and the structured workflow knowledge base $\mathcal{W}_p$. This process relies on a semantic relevance retrieval mechanism: the system queries the structured template library based on module P_{name} and P_{tag}, computes embedding similarities, and ranks candidate templates to retrieve the most relevant process template P_k in $\mathcal{K}_p$ for $\mathcal{T}_i$, and constructs the behavior tree-style plan π_i. Each step $s_j^i \in \pi_i$ is derived from a corresponding step in the retrieved process, which the *Planner* further completes and adapts based on $\mathcal{T}_i$, forming the atomic unit for downstream sub-task modeling and execution. Formally,

$$\hat{s}_j^i = \textit{Planner}(\mathcal{K}_p, s_j^i, \mathcal{W}_i) \tag{2}$$

where $\mathcal{W}_i$ is the current workspace. Suppose the relevance scores of all candidate modules fall below a predefined threshold τ. In that case, the system falls back to a curated set of high-quality templates P_{ex} and augments it with the top k matches P_{ctx} from $\mathcal{K}_p$ as contextual prompts to guide the LLM in the generation of a plan. The final plan generated by *Planner* is handed over to *Coordinator* for scheduling and execution.

Branch-Enhanced Hierarchical Coordinator

This section introduces the hierarchical coordination mechanism in RCAFlow for multi-branch reasoning. The system adopts a Git-inspired branch management strategy to achieve path decoupling and execution isolation. *Coordinator* is responsible for dynamically scheduling tasks and aggregating results, ensuring efficient inter-agent communication while preserving the hierarchical architecture's scalability and modularity advantages.

Git-Inspired Branch Manager To support state management and trajectory tracking in multi-branch reasoning, RCAFlow draws inspiration from Git by introducing Git-like abstractions of *Workspace* and *Branch*. For each subtask $\mathcal{T}_i$, the system creates an independent branch $\mathcal{B}_i$, representing an isolated reasoning path. Inspired by Git's file system–oriented storage model, each branch maintains its execution record $\mathcal{W}_i$, the reasoning steps π_i generated by the Planner, and relevant metadata (e.g., timestamps, parent branch identifiers). A fully decoupled file-level mechanism allows branches to proceed independently, avoiding state pollution and context interference.

The lifecycle of each branch is centrally managed by a global stack structure maintained throughout the RCAFlow reasoning process. The currently active branch always resides at the top of the stack. When creating a new branch, *Brancher* module automatically pushes it onto the stack and switches the execution context. During the merging phase, the system pops the current branch from the stack. The *Coordinator* aggregates the execution results of $\mathcal{B}_i$ into the workspace of its parent branch $\mathcal{W}_{p(i)}$, ensuring that the memory footprint of the main reasoning path remains bounded. *Brancher* then merges the completed branch into the main reasoning path. This mechanism ensures the consistency of the main branch state and supports atomic-level multi-branch rollback and data access, enhancing the reproducibility and isolation of reasoning trajectories.

Adaptive Hierarchical Coordinator To decouple reasoning planning from execution and efficiently coordinate hierarchical workflows, RCAFlow introduces the core scheduling module, Adaptive Hierarchical Coordinator. This module not only accurately dispatches reasoning steps generated by *Planner* to *Executor* for tool invocation, but also supports key capabilities such as branch switching, state alignment, and dynamic trajectory adjustment.

Coordinator incrementally parses the plan trajectory π generated by *Planner*, evaluates the execution complexity of each step s_i, and coordinates its assignment to the appropriate next agent $\mathcal{A}_i$ for execution. If a step matches an existing process template, it is handed over to *Brancher* to create a

new child branch $\mathcal{B}_{c(i)}$ for a new task $\mathcal{T}_{c(i)}$. Otherwise, the system evaluates the implementation difficulty of the step to decide whether a new branch should still be created. *Coordinator* then invokes *Brancher* to construct a new subtask and suspends the current branch while awaiting its response. For steps that do not require further decomposition, the system directly assigns the appropriate tool for execution. Once all tasks within a branch are completed, or when a selector step finishes execution, the system assigns a confidence score c and compares it against a threshold θ. If $c \leq \theta$, the current information cannot complete the task. In such cases, *Coordinator* re-invokes *Planner* to insert supplementary steps or replan the current execution trajectory:

$$\mathcal{A}_i = \textit{Coordinator}(\mathcal{K}_p, s_i, \mathcal{W}_i, \mathcal{T}_i) \tag{3}$$

As reasoning progresses, once $\mathcal{B}_i$ completes its intermediate goals, *Coordinator* summarizes and aggregates the execution results and invokes *Brancher* to merge them back into the parent branch. This ensures state consistency and global memory alignment, prevents low-level noise from disrupting high-level reasoning, and facilitates cross-level knowledge transfer. Furthermore, upon completing task $\mathcal{T}$, the system performs reflective analysis based on the execution outcome. In the case of failure, it conducts a post-hoc examination of both the result and the reasoning trajectory. It archives the extracted insights into the memory $\mathcal{M}_r$, as a knowledge substrate for subsequent RCA processes and strategy refinement.

State-Aware Task Execution

This section introduces a state-aware task execution mechanism that improves RCAFlow's capability to handle large-scale multi-modal fault data via a collaborative framework with external tools. A State-Aware Analyzer conducts a multi-dimensional evaluation of tool outputs, ensuring precise and controllable context for reasoning, enabling task completion assessment, path refinement, and dynamic decision updates.

Dynamic Task Execution The effectiveness of LLMs in RCA tasks depends not only on their reasoning and planning capabilities but also on their ability to handle large-scale, multi-modal data. In real-world scenarios such as the anomaly detection tasks in OpenRCA, the combined volume of logs, metrics, and traces can exceed 68GB, far beyond the context limits of current mainstream LLMs. Therefore, relying on LLMs to parse and reason over raw anomaly data is impractical and unstable.

To this end, RCAFlow is designed to orchestrate external analytical tools for dynamic data processing. The system first leverages the Coordinator to retrieve appropriate analytical tools $\mathcal{K}_t$ based on the current reasoning plan S and task objective T. The system invokes LLM to automatically generate executable analytical code if the required tool is not in the predefined toolset. The Executor then executes this code in an isolated Python environment to perform data analysis and anomaly detection.

$$a_j^i = \textit{Executor}(\mathcal{K}_t\, s_j^i, \mathcal{W}_j^i), \tag{4}$$

where a_j^idenotes the action taken at step j of branch i, enhancing the system's flexibility and adaptability.

State-Aware Analyzer RCA is a multi-round, progressive reasoning process that relies on repeated tool invocations based on intermediate results, making it challenging to complete in a single step. Since anomaly data rarely provides explicit signals of whether the root cause has been identified, the system must iteratively integrate information and construct causal chains to approach the actual cause. Poorly designed reasoning chains may lead to premature termination (missing critical steps) or excessive expansion (introducing redundant information), which can disrupt LLM context retention and degrade diagnostic accuracy and stability.

To address this issue, RCAFlow introduces a State-Aware Analyzer that evaluates the results of each tool invocation across multiple dimensions. Specifically, for standard data processing tasks, *Analyzer* writes the results to the file directory of the corresponding workspace $\mathcal{W}_i$, making them accessible for subsequent execution by *Executor*; For analytical tasks, it first contextualizes low-level raw outputs by associating them with system metadata such as service names, container identifiers, and failure types, thereby extracting structured, semantically meaningful high-level information. The results are then returned as structured summaries with highlighted key fields, effectively filtering out noisy elements and emphasizing critical insights. Furthermore, the Analyzer computes metrics such as the number of abnormal entries and the percentage of anomalies to assess task completeness. These aggregated evaluation results are passed back to *Coordinator*, serving as the basis for trajectory updates and subsequent strategy decisions, enabling completeness assessment, dynamic plan adjustment, and intelligent decision-making.

Experiments

Experimental Setup

Datasets To comprehensively evaluate the effectiveness of our method on RCA tasks, we selected three challenging datasets from OpenRCA(Xu et al. 2025): Bank, Market, and Telecom, corresponding to a banking system, an online retail system, and a telecommunications database system, respectively. These datasets originate from real-world enterprise systems with diverse architectures and failure patterns, and integrate large-scale multi-modal observability data, fully capturing the complexity and diversity of RCA tasks.

Evaluation Metrics We adopt the evaluation framework proposed in the OpenRCA benchmark. The primary evaluation metric is Correct accuracy, which requires all predicted root cause elements—including the component, timestamp, and reason—to match their corresponding ground-truth labels exactly. We also report Partial Accuracy, which measures whether the model successfully identifies at least one correct root cause element. As Correct accuracy can be extremely low in some challenging tasks, Partial Accuracy provides a finer-grained performance analysis, helping to reveal the model's capability boundaries and potential weaknesses.

Backbone Model	Method	Bank		Market		Telecom	
		Correct	Partial	Correct	Partial	Correct	Partial
GPT-4o	CoT	5.15	13.24	2.70	10.14	3.92	13.73
	ReAct	4.41	16.18	3.39	14.19	5.88	19.61
	RCA-agent	6.62	19.85	7.43	15.54	9.80	21.57
	mABC	26.47	36.03	26.35	32.43	31.37	45.10
	Flow-of-Action	27.94	37.50	23.65	31.76	33.33	43.14
	RCAFlow (ours)	**41.91**	**57.35**	**32.43**	**44.59**	**45.10**	**56.86**
DeepSeek-V3	CoT	4.41	12.50	4.05	10.81	4.92	15.69
	ReAct	6.62	15.44	4.73	12.84	7.84	19.61
	RCA-agent	7.35	22.06	5.41	21.62	5.88	25.49
	mABC	22.06	34.56	20.95	33.11	31.37	47.06
	Flow-of-Action	27.21	36.76	24.32	34.46	29.41	41.18
	RCAFlow (ours)	**40.44**	**55.88**	**34.46**	**49.32**	**47.06**	**60.78**

Table 1: Main Results On Three OpenRCA Datasets.

Baseline Models We selected three baselines based on LLM: mABC (Zhang et al. 2024a), Flow-of-Action (Pei et al. 2025), and RCA-agent. We try to reproduce the original methods faithfully; however, the results may exhibit slight deviations from the original implementations due to differences in experimental settings. We provide each method with a fault rule knowledge base based on the RCA-agent to ensure fairness. Since RCA agents are usually highly dependent on specific scenarios, and most are not open source and difficult to migrate, we also introduce several general open source reasoning frameworks (such as CoT (Wei et al. 2022) and ReAct (Yao et al. 2023)) as a baseline to ensure the generality and reproducibility of the evaluation.

Variants	Bank	Market	Telecom
w/o $\mathcal{K}$	12.50	10.14	15.69
w/o *Planner*	29.41	25.68	39.22
w/o *Coordinator*	25.00	27.45	27.45
w/o *Analyzer*	34.56	30.41	43.14
w/o Reflection	33.82	31.08	41.18
RCAFlow	**40.44%**	**34.36%**	**47.06%**

Table 2: Ablation studies for key components of RCAFlow based on DeepSeek-V3, evaluating the correct across Bank, Market and Telecom.

Main Results

As shown in Table 1, we evaluate RCAFlow's effectiveness across benchmarks. We can get the following key results:

1) RCAFlow consistently achieves the highest success rate across all datasets regardless of the backbone. For example, on the Bank dataset, RCAFlow achieves 41.91% Correct with GPT-4o and 40.44% with DeepSeek-V3, significantly outperforming the best baselines (e.g., Flow-of-Action at 27.94% and 27.21%, respectively). These results demonstrate RCAFlow's robustness, generalization capability, and adaptability across backbone models and domains.

2) RCAFlow achieves leading performance across both evaluation metrics. On the most challenging Market dataset, RCAFlow reaches 34.46% Correct and 49.32% Partial with DeepSeek-V3, outperforming Flow-of-Action (24.32%, 34.46%) and RCA-agent (5.41%, 21.62%). These results highlight RCAFlow's strong capability in task decomposition and trajectory recovery. Overall, RCAFlow demonstrates superior performance in solving complex tasks and robust partial reasoning under uncertain or incomplete conditions, exhibiting consistent and reliable performance across multiple evaluation metrics.

Ablation Study

To evaluate the impact of RCAFlow components, we design five ablated variants: RCAFlow *w/o* $\mathcal{K}$ removes $\mathcal{K}$, treating all domain knowledge as unknown and relying solely on the LLM for task planning; RCAFlow *w/o Planner* omits semantic retrieval and structured planning, directly feeding the original task context into the language model for decision-making; RCAFlow *w/o Coordinator* eliminates the branch manager and adaptive hierarchical coordinator, replacing it with a linear chain-of-thought reasoning structure; RCAFlow *w/o Analyzer* removes the execution analysis module, preventing the model from summarizing execution outcomes; RCAFlow *w/o* Reflection discards the reflection module, disabling reasoning trace introspection and experience retention. As shown in Table 2, all variants exhibit varying degrees of performance degradation, confirming the necessity and effectiveness of each module. Removing $\mathcal{W}_i$ and the *Coordinator* leads to the most significant drops in performance, highlighting the importance of task decomposition grounded in prior knowledge and reliable subtask execution with branch-aware adaptability. Furthermore, the performance decline of RCAFlow *w/o Planner* is greater than that of RCAFlow *w/o* Reflection and RCAFlow *w/o Analyzer*, indicating that structured planning plays a vital role in enhancing task compliance and reasoning efficiency. Although the absence of reflection and *Analyzer* causes a minor degradation, overall system performance still suffers, suggesting the necessity of execution summarization

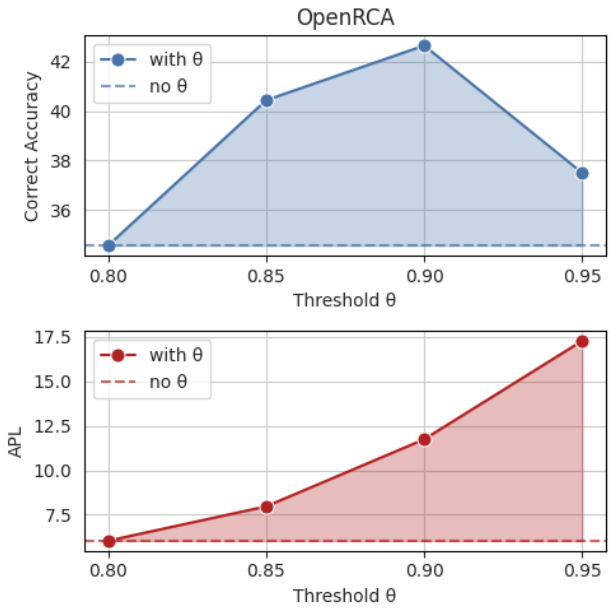

Figure 2: Impact of θ.

and reasoning reflection. These findings further validate the complementary and critical roles of structured knowledge guidance, hierarchical planning and execution, reflection, and adaptive memory in achieving robust and interpretable RCA.

Impact of threshold θ

We evaluate the confidence-based subtask execution mechanism on the Bank dataset. Specifically, the system employs a threshold θ, based on which *Coordinator* determines whether the current subtask has been sufficiently completed. To systematically assess the impact of θ on model performance, we introduce an additional evaluation metric: Average Path Length (APL). APL is denoted by $\frac{1}{N_p}\sum_{k=1}^{N_p} L_k$, where L_k denotes the diagnostic path length of the k-th sample, and N_p represents the number of samples that complete diagnosis within the specified maximum path length. As shown in Figure 2, Correct remains relatively stable across different threshold values, benefiting from the decision constraints imposed by the structured workflow, which help maintain reasoning accuracy across diverse task paths. In contrast, APL increases with the threshold, as the system tends to over-extend task execution—continuing analysis even after certain subtasks are completed—resulting in redundant reasoning steps. Notably, Correct exhibits a rise-then-fall pattern as the threshold increases: moderate thresholds help complete reasoning paths more accurately, improving performance; however, excessively high thresholds introduce noisy or unnecessary steps, leading to path bloat, information redundancy, and increased cognitive load, thereby degrading performance. Conversely, overly low thresholds may cause the system to oversimplify complex subtasks, skipping critical steps and resulting in incomplete root cause analysis. Based on this trade-off, we set the default threshold to $\theta = 0.90$, which balances reasoning efficiency and task accuracy well.

Sensitivity Analysis of Workflow

Initial Workflow In our experiments, we adopt the original RCA-agent's fault-handling rules without introducing complex task templates or prior knowledge. Results show that even a simplified workflow can achieve high diagnostic accuracy, demonstrating the effectiveness of basic structured knowledge in constraining reasoning and guiding decisions. Furthermore, as shown in Table 3, enhancements to the original workflow—through additional SOPs and rules—further improve Correct Accuracy, underscoring the value of high-quality workflow design.

Method	Bank	Market	Telecom
RCAFlow	40.44	34.36	47.06
RCAflow with few SOPs	43.38	36.49	49.02

Table 3: Sensitivity analysis of initial workflow, evaluating Correct based on DeepSeek-V3.

Workflow Generation Since workflow generation relies on LLMs and involves relatively simple operations, the system may struggle to produce accurate and effective workflows when handling complex TSGs—a limitation for future work. However, our experiments are conducted on fault rules with relatively simple structures. Results show that the system can still significantly improve fault localization efficiency, demonstrating its practicality in typical scenarios. In addition, we perform multiple workflow generation trials for each rule to account for the stochastic nature of LLMs. As shown in Table 4, the accuracy metrics exhibit low variance, indicating that the system is robust.

Backbone	Test-1	Test-2	Test-3
GPT-4o	41.91	37.50	43.38
DeepSeek-V3	40.44	36.76	41.18

Table 4: Sensitivity analysis of workflow generation, evaluating Correct across Bank.

Conclusion

This paper presents RCAFlow, a hierarchical multi-agent system informed by structured workflows, designed to address RCA in complex microservice environments. By integrating interpretable plan generation, Git-inspired branch management, hierarchical task decomposition, and state-aware execution mechanisms, RCAFlow demonstrates superior performance on the OpenRCA benchmark, validating its effectiveness in accuracy, interpretability, and generalization in complex diagnostic scenarios.

Acknowledgments

This work was supported by grants from the National Key Research and Development Program of China(No.2024YFB4505900) and Meituan.

References

Aggarwal, P.; Gupta, A.; Mohapatra, P.; Nagar, S.; Mandal, A.; Wang, Q.; and Paradkar, A. 2020. Localization of Operational Faults in Cloud Applications by Mining Causal Dependencies in Logs Using Golden Signals. In *ICSOC Workshops*.

Ahmed, T.; Ghosh, S.; Bansal, C.; Zimmermann, T.; Zhang, X.; and Rajmohan, S. 2023a. Recommending Root-Cause and Mitigation Steps for Cloud Incidents Using Large Language Models. In *Proceedings of the 45th International Conference on Software Engineering*, ICSE '23, 1737–1749. IEEE Press. ISBN 9781665457019.

Ahmed, T.; Ghosh, S.; Bansal, C.; Zimmermann, T.; Zhang, X.; and Rajmohan, S. 2023b. Recommending root-cause and mitigation steps for cloud incidents using large language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 1737–1749. IEEE.

Chen, Y.; Xie, H.; Ma, M.; Kang, Y.; Gao, X.; Shi, L.; Cao, Y.; Gao, X.; Fan, H.; Wen, M.; et al. 2024. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*, 674–688.

Colledanchise, M.; and Ögren, P. 2018. Behavior Trees in Robotics and AI.

Ghosh, S.; Shetty, M.; Bansal, C.; and Nath, S. 2022. How to fight production incidents?: an empirical study on a large-scale cloud service. In *ACM Symposium on Cloud Computing*.

Goel, D.; Magazine, R.; Ghosh, S.; Nambi, A.; Deshpande, P.; Zhang, X.; Bansal, C.; and Rajmohan, S. 2025. eARCO: Efficient Automated Root Cause Analysis with Prompt Optimization. arXiv:2504.11505.

Han, X.; Absar, S.; Zhang, L.; and Yuan, S. 2025. Root Cause Analysis of Anomalies in Multivariate Time Series through Granger Causal Discovery. In *The Thirteenth International Conference on Learning Representations*.

Jiang, J.; Lu, W.; Chen, J.; Lin, Q.; Zhao, P.; Kang, Y.; Zhang, H.; Xiong, Y.; Gao, F.; Xu, Z.; Dang, Y.; and Zhang, D. 2020. How to mitigate the incident? an effective troubleshooting guide recommendation technique for online service systems. In *ESEC/SIGSOFT FSE*.

Kim, J.; Rhee, S.; Kim, M.; Kim, D.; Lee, S.; Sung, Y.; and Jung, K. 2025. ReflAct: World-Grounded Decision Making in LLM Agents via Goal-State Reflection. *arXiv preprint arXiv:2505.15182*.

Kim, M.; Bursztyn, V.; Koh, E.; Guo, S.; and Hwang, S.-w. 2024. RaDA: Retrieval-augmented Web Agent Planning with LLMs. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics: ACL 2024*, 13511–13525. Bangkok, Thailand: Association for Computational Linguistics.

Lin, J.; Chen, P.; and Zheng, Z. 2018. Microscope: Pinpoint Performance Issues with Causal Graphs in Micro-service Environments. *Lecture notes in computer science*.

Nguyen, H.; Shen, Z.; Tan, Y.; and Gu, X. 2013. FChain: Toward Black-Box Online Fault Localization for Cloud Systems. In *IEEE International Conference on Distributed Computing Systems*.

Pei, C.; Wang, Z.; Liu, F.; Li, Z.; Liu, Y.; He, X.; Kang, R.; Zhang, T.; Chen, J.; Li, J.; 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.

Pham, L.; Ha, H.; and Zhang, H. 2024. Root Cause Analysis for Microservice System based on Causal Inference: How Far Are We? In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, ASE '24, 706–715. New York, NY, USA: Association for Computing Machinery. ISBN 9798400712487.

Prasad, A.; Koller, A.; Hartmann, M.; Clark, P.; Sabharwal, A.; Bansal, M.; and Khot, T. 2024. ADaPT: As-Needed Decomposition and Planning with Language Models. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Findings of the Association for Computational Linguistics: NAACL 2024*, 4226–4252. Mexico City, Mexico: Association for Computational Linguistics.

Roy, D.; Zhang, X.; Bhave, R.; Bansal, C.; Las-Casas, P.; Fonseca, R.; and Rajmohan, S. 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.

Sun, H.; Zhuang, Y.; Kong, L.; Dai, B.; and Zhang, C. 2023. Adaplanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36: 58202–58245.

Sun, Y.; Lin, Z.; Shi, B.; Zhang, S.; Ma, S.; Jin, P.; Zhong, Z.; Pan, L.; Guo, Y.; and Pei, D. 2024. Interpretable Failure Localization for Microservice Systems Based on Graph Autoencoder. In *ACM Transactions on Software Engineering and Methodology*.

Wang, D.; Chen, Z.; Ni, J.; Tong, L.; Wang, Z.; Fu, Y.; and Chen, H. 2023. Hierarchical Graph Neural Networks for Causal Discovery and Root Cause Localization. arXiv:2302.01987.

Wang, Z.; Liu, Z.; Zhang, Y.; Zhong, A.; Wang, J.; Yin, F.; Fan, L.; Wu, L.; and Wen, Q. 2024. 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*, CIKM '24, 4966–4974. ACM.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xiao, R.; Ma, W.; Wang, K.; Wu, Y.; Zhao, J.; Wang, H.; Huang, F.; and Li, Y. 2024. FlowBench: Revisiting and Benchmarking Workflow-Guided Planning for LLM-based

Agents. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Findings of the Association for Computational Linguistics: EMNLP 2024*, 10883–10900. Miami, Florida, USA: Association for Computational Linguistics.

Xie, Z.; Zhang, S.; Geng, Y.; Zhang, Y.; Ma, M.; Nie, X.; Yao, Z.; Xu, L.; Sun, Y.; Li, W.; and Pei, D. 2024a. Microservice Root Cause Analysis With Limited Observability Through Intervention Recognition in the Latent Space. In *Knowledge Discovery and Data Mining*.

Xie, Z.; Zheng, Y.; Ottens, L.; Zhang, K.; Kozyrakis, C.; and Mace, J. 2024b. Cloud atlas: Efficient fault localization for cloud systems using language models and causal insight. *arXiv preprint arXiv:2407.08694*.

Xiong, R.; Chen, Y.; Khizbullin, D.; Zhuge, M.; and Schmidhuber, J. 2025. Beyond Outlining: Heterogeneous Recursive Planning for Adaptive Long-form Writing with Language Models. arXiv:2503.08275.

Xu, J.; Wu, H.; Wang, J.; and Long, M. 2022. Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy. In *International Conference on Learning Representations*.

Xu, J.; Zhang, Q.; Zhong, Z.; He, S.; Zhang, C.; Lin, Q.; Pei, D.; He, P.; Zhang, D.; and Zhang, Q. 2025. OpenRCA: Can Large Language Models Locate the Root Cause of Software Failures? In *The Thirteenth International Conference on Learning Representations*.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Yen, C.-C.; Sun, W.; Purmehdi, H.; Park, W.; Deshmukh, K. R.; Thakrar, N.; Nassef, O.; and Jacobs, A. 2022. Graph Neural Network based Root Cause Analysis Using Multivariate Time-series KPIs for Wireless Networks. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 1–7.

Zhang, L.; Zhai, Y.; Jia, T.; Huang, X.; Duan, C.; and Li, Y. 2025. AgentFM: Role-Aware Failure Management for Distributed Databases with LLM-Driven Multi-Agents. *arXiv preprint arXiv:2504.06614*.

Zhang, W.; Guo, H.; Yang, J.; Tian, Z.; Zhang, Y.; Chaoran, Y.; Li, Z.; Li, T.; Shi, X.; Zheng, L.; and Zhang, B. 2024a. mABC: Multi-Agent Blockchain-inspired Collaboration for Root Cause Analysis in Micro-Services Architecture. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Findings of the Association for Computational Linguistics: EMNLP 2024*, 4017–4033. Miami, Florida, USA: Association for Computational Linguistics.

Zhang, X.; Ghosh, S.; Bansal, C.; Wang, R.; Ma, M.; Kang, Y.; and Rajmohan, S. 2024b. Automated root causing of cloud incidents using in-context learning with GPT-4. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, 266–277.