OpsMem: Dual-Memory Reasoning with Cross-Memory Resonance for Failure Diagnosis

Yongqian Sun[†], Rongchen Gao[†], Yu Luo[†], Wenwei Gu[†], Shenglin Zhang[†*]
Qingyi Guo[§], Qiuai Fu[‡], Yaoliang Wu[‡], Dan Pei[§]
[†]*Nankai University* [§]*Tsinghua University* [‡]*Huawei Technologies Co., Ltd.*

***Abstract*—Failure diagnosis in modern software systems requires iterative evidence acquisition and hypothesis reasoning guided by operational experience. Existing LLM-based methods improve diagnosis through agentic reasoning or knowledge augmentation, but they often lack a mechanism to coordinate the evolving diagnostic state with operational experience during iterative diagnosis. We propose *OpsMem*, a dual-memory framework that maintains a short-term memory for the current diagnostic state and a long-term memory for reusable operational experience. *OpsMem* uses cross-memory resonance to activate state-relevant long-term memory, conditions multi-agent diagnosis on the short-term and activated long-term memories, and consolidates reusable experience from solved incidents back into long-term memory. Experiments on a real-world Huawei microservice failure diagnosis dataset show that *OpsMem* outperforms representative agentic-reasoning and knowledge-augmented baselines, improving Match and Relevant by up to 46.88% and 18.39% over the strongest baseline, respectively.**

***Index Terms*—failure diagnosis, large language models, multi-agent systems, agent memory**

I. Introduction

Failure diagnosis is critical for maintaining the reliability of modern software systems, as engineers must quickly identify root causes and restore affected services when failures occur [1]. Traditionally, this task has relied heavily on manual inspection and expert experience, which is costly, time-consuming, and difficult to scale in today's large-scale distributed environments [2]. To automate this process, prior studies have explored data-driven methods based on machine learning and deep learning [3], [4]. Although these methods have shown promise in specific scenarios, they often depend on stable data distributions, predefined failure patterns, and sufficient labeled data [5]. As a result, they suffer from limited generalizability and interpretability, making them difficult to apply reliably in real-world operations [6].

Recent advances in large language models (LLMs) have demonstrated remarkable capabilities in language understanding, knowledge utilization, and complex reasoning [7], [8]. With these capabilities, LLMs can process diverse operational information and perform diagnostic reasoning, providing new opportunities for improving failure diagnosis [9]–[12]. In real-world operations, engineers diagnose failures through iterative evidence collection, observation analysis, and hypothesis refinement guided by operational experience [13].

*Shenglin Zhang is the corresponding author.

To support such iterative diagnosis, ReAct [14] enables LLMs to interleave reasoning and actions, forming a feedback loop between evidence acquisition and hypothesis refinement. However, in traditional ReAct-style diagnosis, the evolving diagnostic trajectory is often accumulated as a linear sequence of thoughts, actions, and observations in the context window. As the trajectory grows longer, such linear context becomes less reliable, making long-horizon diagnosis unstable [15], [16]. GoS [13] highlights typical long-horizon failures, including evidence fabrication, context drift, and failed backtracking, and mitigates them by organizing evidence and hypotheses into a structured belief state. Nevertheless, reliable diagnosis still requires operational experience for guidance.

Although LLMs acquire broad parametric knowledge during training, such knowledge is often insufficient for real-world failure diagnosis, which relies heavily on system-specific operational experience [17]. Existing methods usually incorporate such experience through retrieval, where a diagnosis query is used to retrieve relevant knowledge as auxiliary context. Typical implementations include VectorRAG [18], which retrieves semantically similar document chunks, and graph-based RAG [19]–[21], which exploits structured relations for retrieval. Regardless of the retrieval form, deciding what knowledge to expose to the context is critical: missing relevant knowledge provides limited guidance, while irrelevant knowledge causes context pollution [22]. Moreover, as observations and hypotheses evolve during diagnosis, statically retrieved knowledge can be weakly aligned with the current diagnostic state.

As illustrated in Fig. 1, agentic-reasoning and knowledge-augmented methods address two complementary aspects of failure diagnosis. Agentic-reasoning methods emphasize diagnostic state, while knowledge-augmented methods emphasize operational experience. However, effective diagnosis requires the two to be tightly coupled: the current diagnostic state should activate relevant experience to guide reasoning. For example, when current state contains evidence of sleeping database connections, it should activate experience about idle-slot occupation rather than generic overload. This motivates a dual-memory perspective. Building such a framework faces challenges. **(1) Memory Representation.** Modeling incident-specific states and cross-incident experience with different granularities. **(2) Memory Coordination.** Bridging dynamic diagnostic states with relevant experience during reasoning. **(3) Memory Consolidation.** Incorporating new experience while avoiding unreliable memory accumulation.

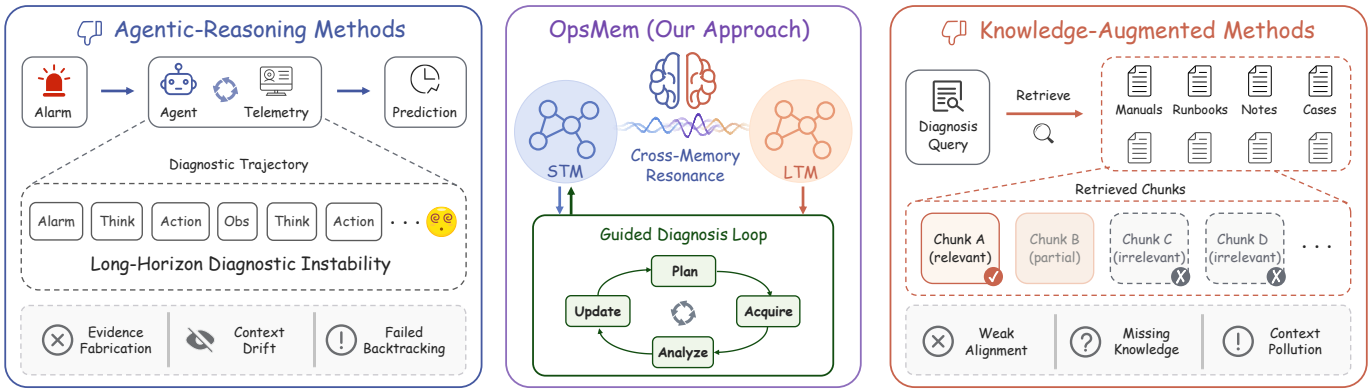

Fig. 1. Motivation of OpsMem from a state–experience coupling perspective. Existing agentic-reasoning methods maintain diagnostic trajectories but suffer from long-horizon instability, whereas knowledge-augmented methods retrieve operational experience without continuously aligning it with the evolving diagnostic state. OpsMem couples short-term state memory and long-term experience memory via cross-memory resonance to guide diagnosis.

To address these challenges, we propose *OpsMem*, a dual-memory framework for failure diagnosis. *OpsMem* maintains two graph-structured memories: a short-term memory (STM) to capture the current diagnostic state and a long-term memory (LTM) to organize reusable operational experience. During diagnosis, *OpsMem* uses cross-memory resonance to activate LTM subgraphs relevant to the current STM, and then performs memory-conditioned diagnosis based on both the STM and the activated LTM. After diagnosis, *OpsMem* consolidates solved incidents into reusable experience, enabling the LTM to evolve while avoiding noise. Our main contributions are summarized as follows:

- To the best of our knowledge, *OpsMem* is the first dual-memory framework that jointly models diagnostic state and operational experience for failure diagnosis.
- We introduce cross-memory resonance to activate state-relevant experience, enabling the diagnostic process conditioned on both STM and LTM.
- We design a long-term memory consolidation mechanism that distills solved incidents into reusable operational experience, allowing the LTM to continuously evolve.
- We evaluate *OpsMem* on a real-world Huawei dataset, where it improves Match and Relevant by up to 46.88% and 18.39% over the strongest baseline, respectively.
- To support reproducibility, we make all code and prompts publicly available. [1]

II. Related Work

Agent-Based Failure Diagnosis. Recent studies have explored LLM-based agents for failure diagnosis. ReAct [23] explore ReAct-style agents for root cause analysis. TrioXpert [5] employs multi-agent collaboration for anomaly detection, failure classification, and root cause localization, while FoundRoot [12] and R-Log [24] enhance LLM reasoning through structured reasoning and reinforcement learning, respectively. GoS [13] further models diagnosis as an abductive reasoning task and maintains an explicit belief state to support failure diagnosis. However, how to incorporate operational experience into diagnosis remains underexplored in these methods.

[1]https://github.com/gaorch85/OpsMem

Knowledge-Augmented Failure Diagnosis. Another line of work augments diagnosis with operational knowledge. RCA-Copilot [17] uses historical failure cases to support root cause analysis. Flow-of-Action [10] and OScope [2] further introduce SOP documents and historical cases to constrain diagnostic reasoning and improve interpretability. DBAIOps [11] represents database operational experience as a heterogeneous knowledge graph, while FlowXpert [9] constructs hybrid knowledge bases for troubleshooting workflow generation. However, the coordination between operational knowledge and the evolving diagnostic state remains insufficiently studied.

III. Methodology

We present *OpsMem*, a novel dual-memory framework for failure diagnosis. As shown in Fig. 2, given the alarms from the monitoring platform, *OpsMem* first initializes a short-term memory (STM) to represent the evolving diagnostic state of the current incident. At each diagnostic round, the current STM triggers cross-memory resonance (CMR) over the long-term memory (LTM), which activates relevant operational experience for the current diagnostic state. The current STM and the activated LTM then jointly condition a multi-agent diagnosis loop, where agents plan actions, acquire evidence, analyze observations, and update the STM. The updated STM further drives the next round of CMR, keeping the activated LTM aligned with the STM until convergence. After the diagnosis is completed, a multi-agent consolidation module distills the reusable experience into LTM for future diagnosis.

The remainder of this section introduces the four core components of *OpsMem*. Section III-A describes the graph-structured representations of STM and LTM. Section III-B presents how CMR aligns the current diagnostic state with reusable operational experience. Section III-C explains how dual memory conditions multi-agent diagnosis. Finally, Section III-D introduces the LTM consolidation mechanism.

A. Dual-Memory Architecture

To support failure diagnosis, *OpsMem* maintains two graph-structured memories: a short-term memory (STM) that captures the current diagnostic state, and a long-term memory (LTM) that organizes reusable operational experience.

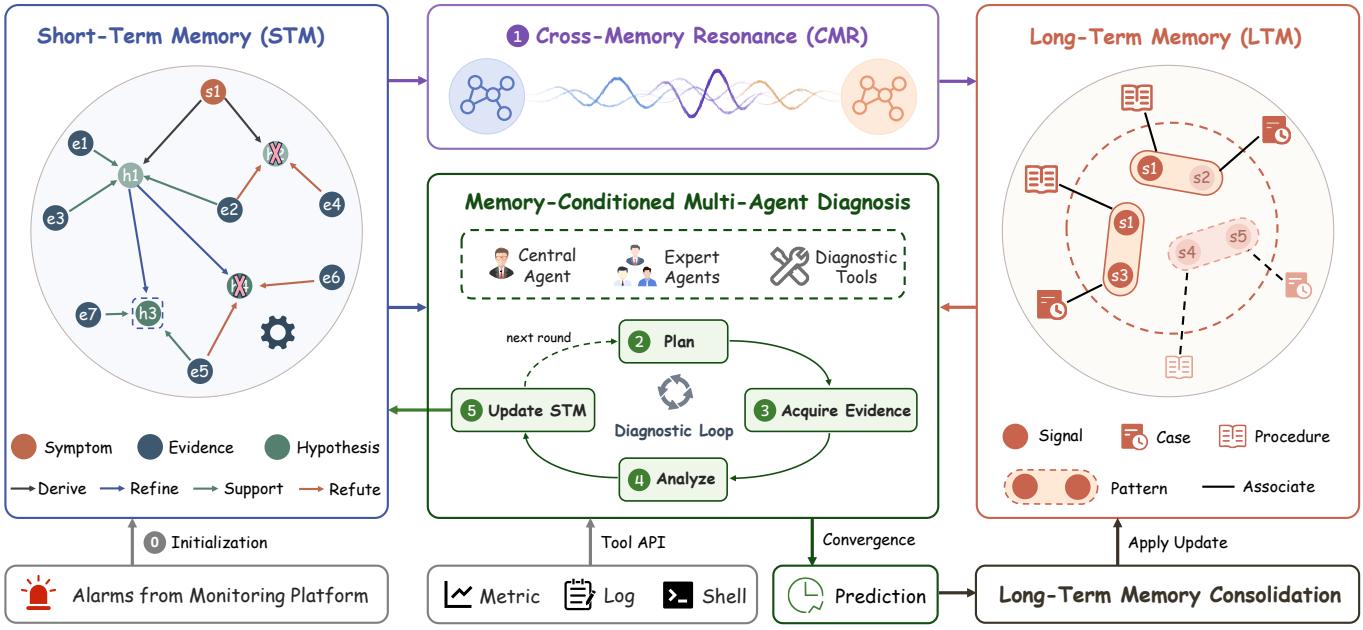

Fig. 2. Overview of *OpsMem*

Short-Term Memory. The STM is represented as a graph $\mathcal{M}_s = (\mathcal{V}_s, \mathcal{E}_s)$ that maintains the evolving diagnostic state of the current incident, following the belief-state abstraction in GoS [13]. The nodes in $\mathcal{V}_s$ include observed symptoms, acquired evidence, and candidate hypotheses. The edges in $\mathcal{E}_s$ encode diagnostic relations among them, such as derive, refine, support, and refute. During diagnosis, newly acquired evidence and intermediate analyses are continuously written into the STM, making it an explicit memory of what has been observed, inferred, and ruled out so far.

Long-Term Memory. The LTM is represented as another graph $\mathcal{M}_l = (\mathcal{V}_l, \mathcal{E}_l)$ that organizes cross-incident operational experience. The node set $\mathcal{V}_l$ contains three types of nodes: patterns, cases, and procedures. A pattern is the core node in the LTM, representing a typical failure mode as a structured association between diagnostic signals, where each signal is a normalized cue that describes a system condition, such as high memory usage. A case records a historical incident related to a pattern, while a procedure records diagnostic steps for that pattern. The weighted edge set $\mathcal{E}_l$ associates patterns with cases and procedures, enabling matched patterns to activate both historical examples and actionable diagnostic guidance. Unlike the STM, the LTM accumulates cross-incident experience and can be repeatedly activated during diagnosis.

B. Cross-Memory Resonance

As shown in Fig. 3, cross-memory resonance (CMR) aligns the diagnostic state with operational experience. It consists of three steps: signal coupling, pattern activation, and memory propagation, which together produce an activated LTM subgraph for the subsequent diagnosis loop.

Signal Coupling. CMR first extracts symptom and evidence nodes from the STM and normalizes them into query signals through an LLM. These query signals are then matched with the signals in LTM patterns. Each LTM signal receives a coupling score based on its maximum semantic similarity to the query signals, and scores below a predefined threshold are set to zero, while the remaining signals are treated as active. This step grounds LTM activation in the concrete observations already recorded in the current STM.

Pattern Activation. After signal coupling, CMR lifts signal-level scores to pattern-level activation. For each pattern, CMR computes its activation score from two factors: the average score of active signals and the ratio of active signals within the pattern. The two factors respectively capture the alignment strength of activated signals and the coverage of the pattern by current observations. Only patterns whose activation scores exceed a predefined threshold are retained for subsequent propagation. In this way, LTM activation focuses on recurring diagnostic associations rather than isolated observations.

Memory Propagation. Starting from the activated patterns, CMR propagates activation to associated nodes according to the edge weights. The propagation scores are used to select the most relevant cases and procedures. Together, activated patterns suggest diagnostic directions, cases provide historical references, and procedures guide subsequent diagnosis.

Whenever the STM is updated during diagnosis, CMR is applied again to refresh the activated LTM subgraph according to the latest diagnostic state.

C. Memory-Conditioned Multi-Agent Diagnosis

Given the current STM and the activated LTM subgraph, *OpsMem* performs a memory-conditioned diagnosis loop. Both memories are serialized into each agent's prompt to jointly condition agent actions.

Each diagnostic round consists of four steps. First, in *Plan*, the CentralAgent plans one or more diagnostic tasks and assigns them to selected ExpertAgents. Second, in *Acquire*

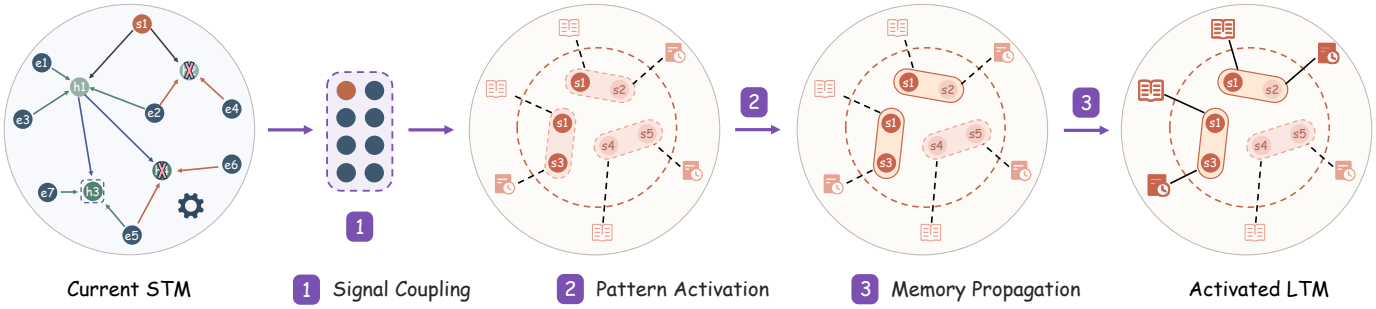

Fig. 3. Cross-Memory Resonance

Evidence, the ExpertAgents iteratively invoke diagnostic tools, such as a log retrieval, to collect observations. Third, in *Analyze*, the ExpertAgents interpret the collected observations and return concise diagnostic reports to the CentralAgent. Finally, in *Update STM*, the CentralAgent aggregates these reports and updates the STM with new evidence, hypotheses, and diagnostic relations.

After each STM update, *OpsMem* checks for convergence following GoS [13]. If a sufficiently supported hypothesis is identified, *OpsMem* outputs the prediction. Otherwise, the updated STM triggers another round of CMR to refresh the activated LTM subgraph for the next diagnostic round.

D. Long-Term Memory Consolidation

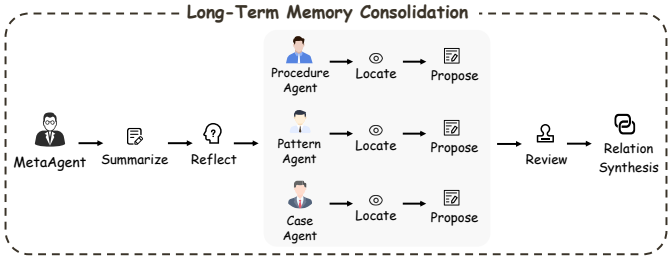

Fig. 4. Long-Term Memory Consolidation

After a failure is resolved, *OpsMem* invokes another multi-agent consolidation module to update the LTM. As shown in Fig. 4, consolidation is coordinated by MetaAgent, which first summarizes the diagnostic trace and reflects on it with the final STM and the activated LTM subgraph. Based on this reflection, the MetaAgent identifies which memories need updates and invokes the corresponding memory agents.

For each invoked memory agent, consolidation first locates related memory nodes in the current LTM, using the activated LTM subgraph as the context. The agent then proposes node-level operations based on the diagnosis summary, the final STM, and the located memory nodes. Each proposal consists of CREATE and DELETE operations: CREATE adds new nodes, while DELETE removes obsolete nodes. When an existing memory node needs revision, the agent deletes the old version and creates a revised one.

The MetaAgent reviews all proposals to filter out unqualified updates. Finally, relation synthesis integrates the validated memory nodes into the LTM through new edges, making them reusable in future incidents.

IV. EXPERIMENTS

In this section, we aim to answer the following research questions (RQs):

RQ1: How effective is *OpsMem* for failure diagnosis?
RQ2: Does each component contribute to *OpsMem*?
RQ3: Can *OpsMem* improve over time?

A. Experimental Setup

1) Datasets: To comprehensively evaluate *OpsMem* under realistic failure diagnosis scenarios, we construct a dataset from Huawei's production microservice systems. The dataset contains 120 real-world failure incidents, covering a broad spectrum of fault types across application services, resource management, databases, and infrastructures. For each incident, we collect the initial failure alarms and multi-source diagnostic observations obtained during incident handling, such as time-series metrics, system logs, and shell-level snapshots. The label is confirmed by on-call engineers through post-incident analysis and used as the ground truth for evaluation.

2) Evaluation: We adopt an LLM-as-a-Judge protocol with Qwen3.5-27B to evaluate diagnostic effectiveness. Following GoS [13], the judge compares each predicted root cause with the ground truth on a three-point scale: 2 for exact match, 1 for relevant, and 0 otherwise. We report *Match* as the proportion of predictions scored 2, and *Relevant* as the proportion of predictions scored at least 1. To improve reliability, we apply self-consistency [25] by obtaining at least five independent judgments and aggregating them via majority voting, and further validate the results by expert sampling.

3) Baselines: We compare *OpsMem* with two groups of representative baselines. For agentic-reasoning methods, we include ReAct [14], which interleaves reasoning and actions, and GoS [13], which maintains a belief state for long horizon diagnosis. For knowledge-augmented methods, we equip GoS with three representative RAG strategies. VectorRAG [18] retrieves semantically similar chunks from the corpus. GraphRAG [19] builds an entity graph and uses community summaries for retrieval. LinearRAG [21] constructs a relation-free graph and retrieves passages through entity activation.

4) Long-Term Memory Construction: Before evaluation, *OpsMem* is initialized with an LTM built from interviews, questionnaires, and operational documents. We use GPT-5.4 to assist in extracting failure patterns, cases, and procedures from these sources. The LTM sources were finalized before

evaluation, with all 120 evaluation incidents excluded. For fair comparison, all knowledge-augmented baselines use the same knowledge sources but organize them in their own retrieval formats.

5) Implementation Details: For signal coupling, we encode signals with BGE-M3 and compute cosine similarity between embeddings. Both the signal and pattern activation thresholds are set to 0.6. The two factors in pattern activation are equally weighted. At each CMR step, we retain the top-3 patterns, cases, and procedures. The diagnosis loop runs up to 3 rounds and returns the best-supported hypothesis if not converged.

B. RQ1: Overall Performance

TABLE I
OVERALL PERFORMANCE (%)

Seed LLM	Method	Match	Relevant
Qwen3.5-27B	*Agentic-Reasoning Methods*		
	ReAct	20.83	49.17
	GoS	30.83	55.83
	Knowledge-Augmented Methods		
	GoS + VectorRAG	48.33	61.67
	GoS + GraphRAG	53.33	71.67
	GoS + LinearRAG	53.33	72.50
	OpsMem	**78.33**	**85.83**
Gemma-4-31B	*Agentic-Reasoning Methods*		
	ReAct	22.50	58.33
	GoS	29.17	51.67
	Knowledge-Augmented Methods		
	GoS + VectorRAG	50.00	73.33
	GoS + GraphRAG	56.67	79.17
	GoS + LinearRAG	48.33	72.50
	OpsMem	**63.33**	**82.50**
GLM-4-32B	*Agentic-Reasoning Methods*		
	ReAct	17.50	42.50
	GoS	21.67	59.17
	Knowledge-Augmented Methods		
	GoS + VectorRAG	34.17	66.67
	GoS + GraphRAG	33.33	65.83
	GoS + LinearRAG	40.00	69.17
	OpsMem	**53.33**	**77.50**

Table I reports the overall diagnosis performance. Across all three seed LLMs, *OpsMem* consistently achieves the best results on both Match and Relevant, showing that its benefit is not tied to a specific backbone. Compared with the strongest baseline under each seed LLM, *OpsMem* improves Match by 6.66–25.00 points and Relevant by 3.33–13.33 points.

Agentic-Reasoning methods, such as ReAct [14] and GoS [13], perform worse because they mainly organize the current reasoning process but lack operational experience. Knowledge-Augmented variants improve over GoS, confirming the value of external experience. However, they still lag behind *OpsMem*, since retrieved knowledge is not explicitly aligned with the evolving diagnostic state. By maintaining the current diagnostic state in STM and activating state-relevant experience from LTM through CMR, *OpsMem* better guides evidence acquisition and hypothesis evaluation, leading to more accurate root-cause identification.

C. RQ2: Ablation Study

TABLE II
ABLATION STUDY (%)

Seed LLM	Method	Match	Relevant
Qwen3.5-27B	***OpsMem***	**78.33**	**85.83**
	w/o STM	45.00	63.33
	w/o LTM	30.83	55.83
	w/o CMR	56.67	68.33
	w/o LTM Consolidation	70.83	80.83

Table II validates the contribution of each component. Removing LTM causes the largest degradation, confirming the importance of operational experience. Removing STM also substantially hurts performance, indicating that explicit state tracking is necessary to maintain accumulated evidence and hypotheses during long-horizon diagnosis. The drop without CMR further shows the importance of state-aware LTM activation, which aligns operational experience with the evolving diagnostic state. Removing LTM consolidation yields a smaller but consistent decline, suggesting that LTM evolution brings additional gains. Overall, both memories, their coordination, and consolidation are necessary for *OpsMem*.

D. RQ3: Self-Evolution via Long-Term Memory Consolidation

TABLE III
SELF-EVOLUTION ACROSS INCIDENT WINDOWS

Seed LLM	Incident Range	Gains vs. w/o LTM Con.	
		Match	Relevant
Qwen3.5-27B	1 – 30	+0	+1
	31 – 60	+3	+1
	61 – 90	+1	+3
	91 – 120	+5	+1

Table III further examines whether LTM consolidation enables *OpsMem* to improve over time. We process the 120 incidents sequentially and split them into four consecutive windows. After each diagnosed incident, *OpsMem* consolidates validated experience into the LTM, while the w/o LTM Consolidation variant keeps the initial LTM fixed. Compared with this variant, *OpsMem* correctly diagnoses additional incidents under both Match and Relevant metrics in every window, with the largest exact-match gain appearing in the last window. This suggests that consolidated experience can be reused by subsequent diagnoses, allowing the LTM to evolve over time.

E. Case Study

Fig. 5 shows one diagnosis round of *OpsMem* on an anonymized real-world Huawei incident. Starting from the current STM, *OpsMem* activates two relevant LTM patterns, workload saturation and idle-slot occupation, and dispatches two DBA agents for targeted checks. The workload-side evidence weakens the overload/slow-query hypotheses, while the connection-side evidence supports idle-slot occupation caused by connection-pool issues. *OpsMem* then updates the STM with the new evidence and analysis.

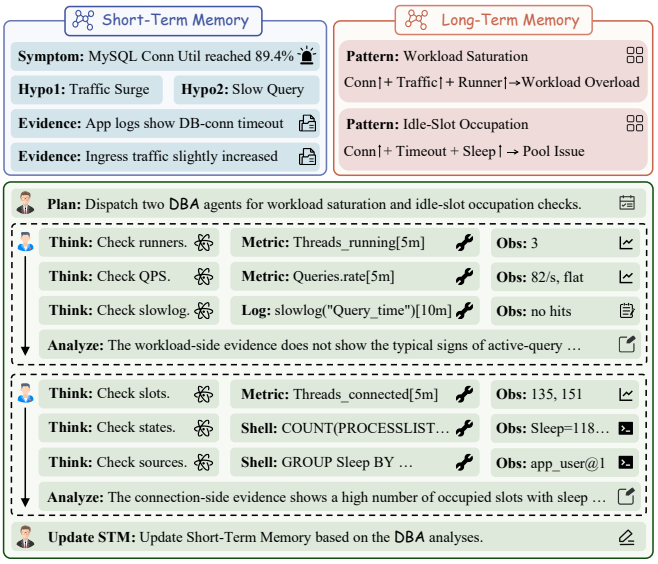

Fig. 5. A diagnosis round of *OpsMem* on a real-world incident

V. Conclusion

This paper presents *OpsMem*, a dual-memory framework for failure diagnosis that couples the short-term diagnostic memory with long-term operational memory. *OpsMem* uses Cross-Memory Resonance to align the current diagnostic state with relevant operational experience, and conditions the diagnosis on both memories. After a failure is resolved, *OpsMem* distills reusable experience into LTM, enabling *OpsMem* to evolve over time. Experiments on a real-world Huawei microservice dataset show that *OpsMem* consistently outperforms representative baselines, and further studies validate the effectiveness of dual-memory coordination and LTM consolidation.

References

[1] Y. Sun, B. Shi, M. Mao, M. Ma, S. Xia, S. Zhang, and D. Pei, "Art: A unified unsupervised framework for incident management in microservice systems," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024, pp. 1183–1194.

[2] Y. Zhao, S. Zhang, Y. Sun, W. Gu, Y. Sun, L. Wang, L. Shi, C. Huang, G. Yang, L. Zhang, and D. Pei, "When llms listen to experts: Accurate failure diagnosis in operating systems," in *Proceedings of the IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice*, 2026.

[3] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, D. Liu, Q. Xiang, and C. He, "Latent error prediction and fault localization for microservice applications by learning from system trace logs," in *Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 683–694.

[4] S. Zhang, P. Jin, Z. Lin, Y. Sun, B. Zhang, S. Xia, Z. Li, Z. Zhong, M. Ma, W. Jin *et al.*, "Robust failure diagnosis of microservice system through multimodal data," *IEEE Transactions on Services Computing*, vol. 16, no. 6, pp. 3851–3864, 2023.

[5] Y. Sun, Y. Luo, X. Wen, Y. Yuan, X. Nie, S. Zhang, T. Liu, and X. Luo, "Trioxpert: An automated incident management framework for microservice system," in *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2025, pp. 3239–3250.

[6] Y. Luo, J. Jiang, J. Feng, L. Tao, Q. Zhang, X. Wen, Y. Sun, S. Zhang, and D. Pei, "Opsagent: An evolving multi-agent system for incident management in microservices," *arXiv preprint arXiv:2510.24145*, 2025.

[7] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[8] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[9] B. Shi, Y. Luo, J. Wang, Y. Zhao, S. Zhang, B. Hao, C. Zhao, Y. Sun, Z. Zhang, R. Sun *et al.*, "Flowxpert: Expertizing troubleshooting workflow orchestration with knowledge base and multi-agent coevolution," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2025, pp. 4839–4850.

[10] C. Pei, Z. Wang, F. Liu, Z. Li, Y. Liu, X. He, R. Kang, T. Zhang, J. Chen, J. Li *et al.*, "Flow-of-action: Sop enhanced llm-based multi-agent system for root cause analysis," in *Companion Proceedings of the ACM on Web Conference 2025*, 2025, pp. 422–431.

[11] W. Zhou, P. Sun, X. Zhou, Q. Zang, J. Xu, T. Zhang, G. Li, and F. Wu, "Dbaiops: A reasoning llm-enhanced database operation and maintenance system using knowledge graphs," *Proceedings of the VLDB Endowment*, vol. 19, no. 6, pp. 1319–1331, 2026.

[12] Z. Xie, Z. Li, X. He, S. Zhang, L. Xu, Y. Yang, T. Zhang, J. Chen, R. Shi, and D. Pei, "Foundroot: Towards foundation model for root cause analysis via structured deep thinking," in *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering*, 2026.

[13] Y. Luo, R. Gao, L. Teng, X. Wen, J. Jiang, Q. Zhang, Y. Sun, S. Zhang, J. Feng, T. Liu *et al.*, "Graph of states: Solving abductive tasks with large language models," in *Forty-Third International Conference on Machine Learning*, 2026.

[14] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *The eleventh international conference on learning representations*, 2023.

[15] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the association for computational linguistics*, vol. 12, pp. 157–173, 2024.

[16] P. Laban, H. Hayashi, Y. Zhou, and J. Neville, "Llms get lost in multi-turn conversation," in *International Conference on Learning Representations*, 2026.

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

[18] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[19] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, "From local to global: A graph rag approach to query-focused summarization," *arXiv preprint arXiv:2404.16130*, 2024.

[20] B. J. Gutiérrez, Y. Shu, W. Qi, S. Zhou, and Y. Su, "From rag to memory: Non-parametric continual learning for large language models," 2025.

[21] L. Zhuang, S. Chen, Y. Xiao, H. Zhou, Y. Zhang, H. Chen, Q. Zhang, and X. Huang, "Linearrag: Linear graph retrieval augmented generation on large-scale corpora," *arXiv preprint arXiv:2510.10114*, 2025.

[22] F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. H. Chi, N. Schärli, and D. Zhou, "Large language models can be easily distracted by irrelevant context," in *Proceedings of the 40th International Conference on Machine Learning*, 2023, pp. 31 210–31 227.

[23] D. Roy, X. Zhang, R. Bhave, C. Bansal, P. Las-Casas, R. Fonseca, and S. Rajmohan, "Exploring llm-based agents for root cause analysis," in *Companion proceedings of the 32nd ACM international conference on the foundations of software engineering*, 2024, pp. 208–219.

[24] Y. Liu, Z. Chen, S. Xu, M. He, S. Tao, W. Meng, Y. Xie, T. Han, C. Zhao, J. Du, D. Wei, S. Zhang, and Y. Sun, "R-log: Incentivizing log analysis capability in llms via reasoning-based reinforcement learning," in *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering: Software Engineering in Practice*, 2026.

[25] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," in *The Eleventh International Conference on Learning Representations*, 2023.