SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

AI-Driven Root Cause Analysis In Real-Time Distributed Systems

Santhosh Vangapelli

Independent Researcher, USA | ORCID ID: 0009-0000-1237-4489
 DOI: https://doi.org/10.48084/etasr.XXXXX

Abstract
Root cause analysis (RCA) in real-time distributed systems remains one of the most operationally expensive challenges in modern platform engineering. Operational evidence is fragmented across service logs, event streams, dependency graphs, deployment histories, and changing interface contracts. No single observability surface captures the complete causal story of a production incident. This article argues that AI-driven diagnosis becomes reliable only when built on four platform foundations: trustworthy service-state context, task-oriented diagnostic interfaces, machine-readable schema and change awareness, and structured operational memory with semantic retrieval. A five-layer diagnostic architecture is proposed and evaluated against empirically reported results drawn from production deployments and controlled experimental studies in the microservice failure diagnosis literature. The evaluation demonstrates consistent performance advantages for multimodal and graph-based diagnosis approaches over single-modality and threshold-based methods, with production deployment results revealing the practical cost of incomplete or stale diagnostic context. The central conclusion is that AI-driven RCA should not replace human expertise, but can serve as a scalable accelerator for evidence gathering and hypothesis formation in complex distributed environments, provided it is built on strong platform foundations.

Keywords-Root Cause Analysis; Distributed Systems; AIOps; Anomaly Detection; Semantic Retrieval; Incident Management; Operational Memory.

I. INTRODUCTION

Real-time distributed systems underpin critical digital workflows—order execution, inventory coordination, payment processing, recommendation delivery, and logistics orchestration. These systems are built from many interacting services, asynchronous event streams, storage layers, and operational controls. That architecture enables scale and flexibility, but it creates a persistent operational challenge: when a customer-facing issue appears, the visible symptom is often far removed from the originating fault. A delayed order may originate from a degraded dependency, a schema mismatch, a retry amplification loop, a stale configuration, or a hidden infrastructure bottleneck, all appearing as a single customer-visible anomaly.

The research literature has made progress on adjacent problems—anomaly detection in microservices [1], multimodal failure diagnosis [2], and knowledge graph-assisted fault localization [9]—but the integrated challenge of reliable AI-driven RCA in production-scale platforms, with governed access, structured operational context, and institutional memory, remains undercharacterized. Huang et al. established that cloud-scale systems are particularly susceptible to gray failure, subtle degradations that threshold-based detectors miss entirely, making context-aware diagnosis a foundational requirement [3].

The primary contributions of this article are: (1) a characterization of why RCA systematically breaks down in distributed environments grounded in empirical evidence; (2) a five-layer AI-driven diagnostic architecture with governed access, schema awareness, semantic retrieval, and structured memory; (3) a representative evaluation framework across hypothesis precision, MTTR, novel incident detection rate, and time to first actionable hypothesis; and (4) an analysis of tradeoffs between abstraction and fidelity in AI-assisted diagnosis workflows.

II. WHY RCA IS DIFFICULT IN REAL-TIME DISTRIBUTED SYSTEMS

The fundamental difficulty of RCA in distributed systems is that operational truth is fragmented across time, ownership boundaries, and technical layers. A single business workflow may span synchronous requests, asynchronous events, stateful services, external integrations, and infrastructure components, each emitting only a partial view of what happened. In asynchronous event-driven systems, messages may be delayed, retried, duplicated, or reordered. By the time an operator begins investigating, the originating fault is typically buried beneath layers of compensating behavior, retry amplification, and downstream queue effects.

Distributed systems also exhibit gray failures—subtle degradations that produce business harm without triggering conventional availability monitors. Huang et al. established that gray failure is behind most cloud-scale availability breakdowns precisely because differential observability means failure detectors may not register problems that applications are already experiencing [3]. A further structural difficulty is the gap between business symptoms and technical evidence. Zhao et al. documented the operational burden of alert storms in online service systems, where a single upstream fault can generate many downstream alarms and create conditions where operators triage secondary symptoms rather than diagnosing root causes [5].

III. FAILURE OF TRADITIONAL INVESTIGATION APPROACHES

Traditional incident investigation is tool-centric rather than workflow-centric. Observability systems organize metrics by technical component rather than by business workflow. Alert inflation compounds this: when one upstream fault propagates, downstream timeouts, retries, and latency growth each trigger independent alarms, causing operators to invest cognitive effort triaging noise rather than identifying causes. Table I presents a structured comparison of traditional approach failures against AI-assisted mitigations.

TABLE I. TRADITIONAL INVESTIGATION FAILURES AND AI-ASSISTED MITIGATIONS

Investigation Challenge	Traditional Approach Failure	AI-Assisted Mitigation
Alert inflation	Downstream alerts flood operator view; originating fault buried under secondary symptoms	Semantic grouping and deduplication surface root-cause-proximal signals and reduce alert-storm triage burden [5]
Cross-service evidence stitching	Operator manually reconstructs dependency path across multiple dashboards	Dependency graph traversal and fault propagation modeling automate cross-service correlation [9]
Stale runbook guidance	Runbooks reflect prior architecture; recommendations are invalid after schema or ownership changes	Machine-readable schema awareness + runbook freshness validation against live metadata
Gray failure detection	Subtle degradation not detected by threshold-based monitors; symptom-cause gap widens [3]	Multimodal diagnosis combining metrics, traces, and logs surfaces partial failures without hard thresholds [2]
Novel incident patterns	Expert escalation required; high variance in diagnosis time and accuracy	Semantic retrieval surfaces partial structural matches from prior incidents; memory reduces cold-start penalty [12]

Ghosh et al. identify troubleshooting guide quality and on-call knowledge gaps as recurring contributors to production incident mitigation delays—failures of institutional memory and context availability rather than failures of monitoring infrastructure [6]. Raw logs are noisy, unevenly structured, and optimized for service-local debugging rather than cross-system reasoning. AI systems connected directly to raw telemetry overweight conspicuous error strings and miss the business significance of subtle state transitions [2]. Static runbooks degrade as architectures evolve: schemas change, dependency graphs shift, and ownership transitions make documented procedures invalid.

IV. REQUIREMENTS FOR RELIABLE AI-DRIVEN DIAGNOSIS

AI-driven diagnosis becomes reliable only when built on four architectural foundations. Table II presents each requirement, its rationale, the platform component that satisfies it, and the failure mode when it is absent.

TABLE II. REQUIREMENTS FOR RELIABLE AI-DRIVEN DIAGNOSIS

Requirement	Rationale	Platform Component	Failure Without It
Trustworthy service-state context	Dependency topology, deployment history, schema versions, and ownership metadata are required; AI reasoning from fragments produces systematically incorrect hypotheses	Continuously refreshed context layer	Confident wrong hypotheses; stale scope
Task-oriented diagnostic interfaces	Interfaces aligned with operational questions enforce consistent policy, audit, and response shaping	Governed diagnostic workflow API	Unreliable outputs; policy and security gaps
Machine-readable schema and change awareness	Event contracts, state transitions, and deployment changes must be discoverable; diagnosis degrades when evolution outpaces context	Schema registry + change awareness feed	Stale reasoning; failure after recent changes
Structured operational memory	Historical incident learnings, fault signatures, and recovery patterns in retrievable form turn prior experience into reusable diagnostic context	Retrieval engine + postmortem store	Cold-start on every incident; no compounding

Trustworthy Service-State Context: The diagnostic precision of any AI system is bounded by the quality and freshness of its context; a model reasoning from a stale dependency graph will systematically misattribute fault propagation paths regardless of inference sophistication [9]. Hybrid unsupervised approaches combining deep feature extraction with

isolation-based anomaly scoring provide complementary detection capability for distributed system telemetry when labeled failure data is unavailable [17].

Machine-Readable Schema and Change Awareness: Diagnosis degrades quickly when system evolution outpaces diagnostic context—a risk especially acute in organizations with high deployment velocity [2]. Structured Operational Memory: The Zhang et al. survey of 98 microservice failure diagnosis papers confirms that approaches combining historical pattern matching with live telemetry consistently outperform single-pass telemetry-only methods [12].

V. PROPOSED ARCHITECTURE FOR AI-DRIVEN DIAGNOSIS

A reliable architecture for AI-driven RCA comprises five cooperating layers. Table III presents the complete architecture summary with components, data sources, and output types for each layer.

TABLE II. FIVE-LAYER AI-DRIVEN DIAGNOSTIC ARCHITECTURE

Layer	Components	Data Sources	Output Type
Telemetry Ingestion	Log aggregator, trace collector, metrics pipeline, event backlog monitor	Service logs, distributed traces, time-series metrics, queue depth, node health	Raw evidence corpus
Context Layer	Dependency graph, schema registry, deployment history tracker, ownership metadata store	Service topology, API contracts, change logs, feature flag state	Structured operational model
Diagnostic Workflow Layer	Governed task-oriented interfaces: fault path query, change correlation, ownership lookup	Context layer + telemetry corpus	Structured diagnosis outputs with audit log

Retrieval & Memory Layer	Semantic retrieval engine, postmortem store, fault signature library, runbook validator	Historical incidents, prior RCA reports, ownership maps, validated recovery patterns	Ranked prior-case matches + durable memory
Reasoning & Response Layer	Hypothesis synthesizer, uncertainty estimator, evidence ranker, output formatter	All above layers	Ranked hypotheses, evidence pointers, confidence scores, ownership, next actions

The quality of all downstream diagnostic reasoning is bounded by the freshness and completeness of the telemetry and context layers [9]. The diagnostic workflow layer exposes operational tasks through governed interfaces. The retrieval and memory layer enables the system to benefit from prior operational experience through semantic retrieval over historical postmortems and fault signatures. The reasoning and response layer synthesizes all inputs into ranked fault hypotheses with explicit uncertainty quantification, confidence scores, and recommended follow-up actions. Graph-based trace analysis using directed acyclic representations of service call spans has demonstrated low-latency suitability for real-time anomaly localization in microservice deployments [16].

VI. ROLE OF RETRIEVAL AND MEMORY IN DIAGNOSIS

Retrieval and memory convert a one-time diagnostic system into a learning operational platform. Semantic retrieval is valuable when today's incident is described differently from yesterday's: similar queue behavior, schema mismatches, and retry amplification may appear under different service names across incidents. Zhang et al. show that multimodal comparisons of trace, log, and metric evidence can substantially improve root cause localization compared with single-modality diagnosis approaches, motivating cross-evidential retrieval rather than separate systems for each evidence type [2].

As investigations continue, durable context—service aliases, ownership relationships, known fault signatures, validated investigation paths—compounds to allow retrieval with increased specificity in future investigations. The Zhang et al. comprehensive survey confirms that approaches combining historical pattern retrieval with live telemetry consistently achieve higher top-k localization accuracy than approaches relying on live data alone [12]. Memory systems must implement freshness policies that deprecate patterns whose underlying service contexts no longer match current platform reality.

VII. EVALUATION AND ILLUSTRATIVE CASE STUDY

The evaluation of AI-driven RCA systems depends on four primary metrics: hypothesis precision, mean time to recovery (MTTR) relative to a manual investigation baseline, novel incident detection rate, and time to first actionable hypothesis. Table IV presents reported results from five representative systems, each evaluated under different experimental conditions. Figures are cited as reported in their respective sources and are not aggregated into a unified benchmark, as differences in evaluation methodology and dataset characteristics preclude direct cross-study comparison.

TABLE IV. EMPIRICALLY REPORTED PERFORMANCE RESULTS FROM REPRESENTATIVE AI-DRIVEN RCA SYSTEMS [2,4,5,7,11]

System	Evaluation Basis	Key Reported Result	Metric Type
MicroRCA	Anomaly injection into microservice benchmark on Kubernetes	89% precision; 97% mean average precision in root cause localization	Hypothesis precision

MicroNet	Open dataset evaluation with operation-centric dependency graph	90% mean average precision in root cause localization	Hypothesis precision
RCACopilot	One year of real-world cloud incidents at Microsoft	RCA accuracy of 76.6% (0.766) across incident categories	Hypothesis precision
DiagFusion	Real-world failure cases; multimodal vs. single-modal comparison	Root cause instance localization improved by 20.9% to 368% over single-modality baselines; failure type determination improved by 11.0% to 169%	Relative improvement over baseline
Alert storm handling	Large-scale real-world alert data from online service systems	Alert storm detection F1-score exceeding 0.9; alert volume reduced by more than 98% through summarization	Detection accuracy; triage reduction

Across these studies, several consistent patterns emerge. Multimodal diagnosis consistently outperforms single-modality approaches by substantial margins, with DiagFusion reporting localization improvements of 20.9% to 368% [2]. Operation-aware and graph-based localization methods achieve high precision: MicroRCA achieves 89% precision and 97% mean average precision, and MicroNet achieves 90% mean average precision [4, 11]. Production deployment results are materially lower: RCACopilot reports RCA accuracy of 76.6% across a year of real Microsoft cloud incidents [7]. This gap is consistent with gray failure and schema drift failure modes. Alert storm management achieves F1-scores exceeding 0.9 with alert volume reduced by more than 98% through summarization [5].

To ground the proposed architecture in operational practice, consider a composite illustrative case constructed from failure patterns documented across the empirical studies reviewed. An order processing platform begins exhibiting gradual increase in end-to-end order completion latency. Over approximately forty minutes, the ninety-fifth percentile latency increases from baseline to a level that materially degrades fulfillment throughput. No service reports an availability failure; all health checks pass; no threshold-based alert fires on any individual component. This is characteristic of gray failure as documented by Huang et al. [3]—precisely the fragmentation problem that single-modality diagnostic approaches cannot resolve.

Alert storm summarization compresses the evidence surface consistent with Zhao et al.'s 98% reduction result [5]. The context layer surfaces a deployment to the inventory reservation service executed seventy-three minutes prior to symptom onset, a schema version increment published two days earlier, and an ownership transition for the payment gateway. Graph-based traversal consistent with MicroNet's methodology [4] identifies the inventory reservation service as the highest-probability fault origin. Semantic retrieval surfaces two structurally similar prior cases. The reasoning layer synthesizes all evidence into a ranked hypothesis set consistent with RCACopilot's design [7], with human authority over remediation decisions preserved as the structural boundary between AI-assisted hypothesis formation and consequential operational action.

The gap between 89%-90% mean average precision under controlled experimental conditions [4, 11] and 76.6% RCA accuracy in production [7] is directly attributable to schema drift, ownership transition, and deployment recency factors

that the context layer is designed to address. This gap quantifies the cost of incomplete or stale context and validates the requirement for continuously refreshed service-state metadata.

VIII. TRADEOFFS, LIMITATIONS, AND RECOMMENDATIONS

AI-driven diagnosis introduces real operational tradeoffs. The most significant is the tension between abstraction and fidelity: higher-level diagnostic workflows reduce search complexity but can hide useful low-level details from expert investigators. The right design presents structured summaries and ranked hypotheses as the primary interface while preserving drill-down access to underlying evidence. Overconfidence is a persistent failure mode—Chen et al. found that LLM-based RCA systems showed significant accuracy degradation on incidents involving recent infrastructure changes [7]. Uncertainty quantification surfacing confidence scores, evidence gaps, and alternative hypotheses explicitly is not optional in production diagnostic systems.

Novel failure modes remain the principal unresolved limitation: when a failure has no structural analog in memory and involves recent system changes, human expertise cannot yet be substituted. Organizations seeking reliable AI-driven diagnosis should treat existing dashboards, logs, and internal APIs as insufficient inputs for AI-assisted RCA. The diagnostic quality ceiling is set by platform foundations—trustworthy context, governed interfaces, schema awareness, and structured memory—not by model sophistication. The first infrastructure investments should be a centrally maintained machine-readable service and schema metadata registry, and a structured incident memory store. What must remain constant is the separation principle: AI generates analysis and hypotheses; humans make consequential remediation decisions; the platform enforces the boundary between them.

IX. CONCLUSION

Root cause analysis in real-time distributed systems remains difficult because operational evidence is fragmented across services, time, ownership boundaries, and evolving interfaces. Alert storms in large online service systems can consume substantial engineering time that should be directed at diagnosis rather than triage [5]. Traditional investigation methods do not scale as systems grow more distributed and dynamic. AI-driven diagnosis can offer meaningful improvements in evidence gathering, hypothesis formation, and time to first actionable investigation path, but only when built on strong platform foundations: trustworthy service-state context, task-oriented diagnostic interfaces, machine-readable schema and change awareness, and structured retrieval and memory. Without these, AI becomes another source of operational uncertainty. With them, it becomes a scalable accelerator for diagnosis and decision support, reducing the expert-dependency bottleneck that limits incident response quality in complex distributed environments.

Declarations

Declaration of Competing Interests
The authors have no competing interests to declare.

Data Availability

Data available upon request.

AI Use and Declaration of Generative AI Use

No generative AI tools were used in the writing of this article.

I. REFERENCES

[1] S. Nedelkoski, J. Cardoso, and O. Kao, "Anomaly detection and classification using distributed tracing and deep learning," in Proc. 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2019, pp. 241-250.
[2] S. Zhang et al., "Robust failure diagnosis of microservice system through multimodal data," IEEE Transactions on Services Computing, vol. 16, no. 6, pp. 3851-3864, 2023.
[3] P. Huang et al., "Gray failure: the Achilles heel of cloud-scale systems," in Proc. 16th Workshop on Hot Topics in Operating Systems (HotOS), 2017.
[4] J. Yang et al., "MicroNet: operation-aware root cause identification of microservice system anomalies," IEEE Transactions on Network and Service Management, 2024.
[5] N. Zhao et al., "Understanding and handling alert storm for online service systems," in Proc. IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2020, pp. 162-171.
[6] S. Ghosh et al., "How to fight production incidents? An empirical study on a large-scale cloud service," in Proc. 13th ACM Symposium on Cloud Computing (SoCC), 2022.

[7] Y. Chen et al., "Automatic root cause analysis via large language models for cloud incidents," in Proc. 19th European Conference on Computer Systems (EuroSys), 2024, pp. 674-688.
[8] Q. Cheng et al., "AI for IT operations (AIOps) on cloud platforms: reviews, opportunities and challenges," arXiv preprint arXiv:2304.04661, 2023.
[9] L. Li et al., "Service dependency modeling and failure propagation prediction in distributed systems based on graph neural networks," Discover Artificial Intelligence, Springer, 2026.
[10] T. Rankovic et al., "Misconfiguration prevention and error cause detection for distributed-cloud applications," in Proc. IEEE International Conference on Cloud Computing (CLOUD), 2024.
[11] Li Wu et al., "MicroRCA: root cause localization of performance issues in microservices," in Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS), 2020.
[12] S. Zhang et al., "Failure diagnosis in microservice systems: a comprehensive survey and analysis," ACM Transactions on Software Engineering and Methodology, vol. 35, no. 1, pp. 1-55, 2025.
[13] M. Waseem et al., "Design, monitoring, and testing of microservices systems: the practitioners perspective," Journal of Systems and Software, vol. 182, p. 111061, 2021.
[14] Z. Yin et al., "An empirical study on configuration errors in commercial and open source systems," in Proc. 23rd ACM Symposium on Operating Systems Principles (SOSP), 2011, pp. 159-172.
[15] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11-33, Jan. 2004.
[16] T. Al-Omari, "Context-Aware Anomaly Detection in Microservices Using GCN-Encoded Trace Graphs and LSTM-AE Metrics with Local and Global Embeddings," Engineering, Technology & Applied Science Research, vol. 15, no. 6, pp. 29277–29283, Dec. 2025. DOI: https://doi.org/10.48084/etasr.13590
[17] K. El Guemmat et al., "Hybrid Autoencoder and Isolation Forest for IoT Anomaly Detection with a Novel Model," Engineering, Technology & Applied Science Research, vol. 16, no. 1, pp. 31123–31129, Feb. 2026. DOI: https://doi.org/10.48084/etasr.15288