Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 14
TPCEE 2026

EC-MMRCA: Evidence-Constrained LLM-Distilled Multimodal Root Cause Analysis for Cloud-Native Microservices

Xuanping Zou [1], Zhou Lu [2], Kailang Wei [3], and Shan Huang [4, *]

[1] Jiangxi University of Science and Technology, Nanchang, China

[2] Northeaster University, Boston, United States

[3] Emory University, Atlanta, United States

[4] University of California San Diego, La Jolla, United States

* Corresponding Author: shanhuang@ucsd.edu

Abstract. Root-cause analysis (RCA) in cloud-native microservices is difficult because resource metrics, logs, and distributed traces reveal different and sometimes contradictory symptoms. Large language models (LLMs) offer useful fault semantics, but unconstrained generation can introduce unsupported candidates and makes online diagnosis expensive and difficult to reproduce. This paper presents EC-MMRCA, an evidence-constrained multimodal RCA framework that uses an LLM only to distill a frozen fault-to-feature compatibility prior. Runtime inference is deterministic and combines robust anomaly evidence, adaptive modality reliability, service-call-graph propagation fit, anomaly-onset precedence, cross-modal agreement, and an explicit evidence floor. To avoid unverified claims, we evaluate the method on a fully disclosed synthetic benchmark whose topologies, telemetry modalities, and six fault categories are aligned with the public RCAEval taxonomy. The generator creates 2,430 unique incidents across three topology families, three random seeds, and standard, high-noise, and one-modality-missing conditions. Every reported value is calculated by the accompanying code. Under the standard condition, EC-MMRCA reaches 86.3% Top-1 service localization, 99.8% Top-3, 0.929 mean reciprocal rank, 79.6% fault-type accuracy, and 69.0% joint service-and-fault accuracy. It improves Top-1 by 16.7 percentage points over the strongest baseline (exact paired McNemar $p=1.46\times10^{-17}$) and retains 83.8% and 84.2% Top-1 under high noise and missing telemetry. The results support evidence-gated semantic reasoning, while the synthetic design and fixed prior remain explicit limitations requiring external validation.

Keywords: Root cause analysis; Microservices; Multimodal telemetry; Large language models; Observability; Evidence-constrained reasoning.

1. Introduction

Cloud-native applications decompose user-facing functions into independently deployable services connected through remote calls. This architecture improves deployment flexibility, but it also turns a local resource or network fault into a distributed symptom pattern. A delayed payment service may inflate checkout latency, trigger retries in upstream callers, and generate secondary errors in notification or order services. Consequently, the service with the largest anomaly is not necessarily the service where the fault started. Distributed tracing systems such as Dapper established the value of request-path evidence [1], while benchmark suites such as DeathStarBench demonstrated the operational complexity of realistic service graphs [2]. OpenTelemetry now provides common conventions for correlating traces, metrics, logs, and resources [3]. The remaining challenge is to transform these heterogeneous signals into a reliable and auditable causal ranking.

Existing RCA methods typically specialize in one evidence source or assume that multimodal fusion will be automatically beneficial. Metric-only approaches are inexpensive but confuse propagated symptoms with origins. Trace-based methods exploit request structure, yet incomplete instrumentation or black-box services can break the graph. Log analysis captures semantic details, but templates may be absent, duplicated downstream, or unstable across releases. Recent multimodal systems improve coverage by combining these sources [14-21], although their fusion mechanisms

Content from this work may be used under the terms of CC BY-NC 4.0 licence (https://creativecommons.org/licenses/by-nc/4.0/).
Published by Warwick Evans Publishing.

may still be dominated by a high-amplitude modality and their learned representations can be difficult to audit. Public evaluations also show that benchmark choice, fault injection, and metric definitions substantially affect conclusions [21].

LLMs create a new opportunity because they can map textual fault descriptions to expected observable indicators and produce human-readable explanations. OpenRCA investigates whether LLMs can locate software failures from telemetry [22], and agent-based work explores staged or tool-augmented diagnosis [23]. However, a free-form LLM placed directly in the online loop introduces three risks. First, the model may name a service or mechanism unsupported by observed telemetry. Second, large raw contexts can dilute decisive evidence and make output sensitive to prompt order. Third, proprietary inference endpoints impede exact reproducibility and add latency and cost. General LLM reasoning techniques—chain-of-thought, ReAct, retrieval augmentation, tree search, and self-training [25-29]—do not by themselves guarantee that a diagnosis is grounded in the deployed topology.

This work therefore treats language-model knowledge as a bounded semantic prior rather than an autonomous diagnostician. We propose EC-MMRCA (Evidence-Constrained Multimodal Microservice Root Cause Analysis), which distills fault-to-feature compatibility once and performs deterministic online ranking from telemetry. Semantic compatibility is gated by two causal cues: whether a candidate's anomaly begins early and whether the observed service-wide anomaly vector matches the propagation footprint expected if that candidate were the origin. A candidate with weak measured evidence is penalized regardless of semantic plausibility. This separation makes the online stage fast, auditable, and exactly repeatable from a small JSON prior.

The paper makes three contributions. First, it introduces an evidence-constrained design for incorporating LLM-derived fault semantics without allowing generative output to override telemetry or expand the candidate set. Second, it develops a graph-temporal multimodal score with adaptive reliability and cross-modal support, jointly ranking root services and fault types. Third, it supplies a reproducible experimental package: a disclosed synthetic generator aligned with public microservice RCA taxonomies, incident-level predictions, statistical tests, figures, and a verification log for every bibliography entry. The package deliberately does not claim production-world validity; instead, it provides a controlled, falsifiable basis for subsequent validation on public and industrial telemetry.

2. Background And Related Work

2.1. Telemetry-Specific RCA

Log parsing and anomaly detection form one major line of evidence. Drain incrementally extracts templates with a fixed-depth tree [4]. DeepLog models event sequences for anomaly detection and diagnosis [5], while LogAnomaly and LogRobust address quantitative patterns and unstable log data [6], [7]. These methods are valuable for extracting structured symptoms, but logs alone have weak guarantees about inter-service propagation. A downstream service may repeat the same error signature as the origin, and a missing template can eliminate otherwise decisive semantic evidence.

Metric- and trace-oriented RCA exploit complementary structure. MicroRCA combines service dependencies and anomalous response times [8]. Graph-based trace analysis uses request paths for architecture understanding and diagnosis [9]. MicroHECL follows anomaly propagation chains on dynamically constructed call graphs [10], whereas TraceRCA contrasts abnormal and normal traces [11]. MicroRank combines trace coverage, PageRank, and spectrum analysis for latency issues [12]. Causal outlier analysis provides a formal basis for reasoning about interventions and anomaly propagation [13]. These techniques motivate our graph-fit and temporal components, but EC-MMRCA does not estimate a full structural causal model from limited incident data; instead, it tests each candidate against a simple, interpretable propagation kernel.

2.2. Multimodal and Fine-Grained RCA

Multimodal RCA has progressed from concatenating observations to explicitly modeling interactions. Eadro jointly learns anomaly detection and localization from traces, logs, and key performance indicators [14]. Nezha builds interpretable event patterns from multimodal observability data [15]. TrinityRCL supports multiple localization granularities, including code-level diagnosis [16]. HeMiRCA correlates trace-derived anomaly scores with heterogeneous metrics for fine-grained root causes [17]. BARO couples robust online change-point detection with statistical ranking [18]. MRCA targets metric-level causes using multimodal data [19], and Medicine uses modality-independent encoding and adaptive optimization [20]. RCAEval consolidates datasets, fault cases, baselines, and evaluation procedures, reporting 735 failures across three microservice systems [21]. TORAI later highlights blind spots created by services without traces [24].

Our method is complementary to learned multimodal encoders. It does not require labeled training incidents or end-to-end neural optimization. Instead, it builds a bounded score from robust statistics, service topology, onset order, and a frozen semantic matrix. This makes its failure modes visible: a wrong graph affects graph fit; delayed sampling affects temporal precedence; missing modalities reduce adaptive evidence; and a mismatched semantic prior affects fault typing. The design favors reproducibility and operational auditability over representational capacity. Beyond microservice diagnosis, Ping et al.'s TriAudit provides a strong example of interpretable multimodal fusion through hierarchical cross-modal attention, reliability-aware gating, and contrastive evidence-chain construction [32], reinforcing our choice to expose modality-specific contributions and cross-modal corroboration.

2.3. LLM-Based Diagnosis

OpenRCA provides a benchmark for LLM-based root-cause localization and demonstrates the emerging interest in post-deployment software diagnosis [22]. MicroRCA-Agent filters logs and traces before staged LLM analysis [23]. More generally, chain-of-thought encourages intermediate reasoning [25], ReAct interleaves reasoning with tool use [26], retrieval-augmented generation injects external evidence [27], Tree of Thoughts searches alternative reasoning paths [28], and STaR bootstraps rationales [29]. These methods improve reasoning flexibility but do not automatically enforce a service inventory, causal order, or telemetry threshold. EC-MMRCA introduces those constraints by design: the LLM contributes only a fixed compatibility matrix; all incident-specific conclusions are computed from observed data. Zhao et al.'s AutoML-Pipeline offers a particularly practical RAG-plus-pre-validation design that checks dependency and resource constraints before deployment [33], closely supporting our principle that model-derived knowledge should be accepted only after domain evidence is satisfied.

3. Problem Formulation

Let a microservice deployment be a directed call graph $G=(V, E)$, where an edge $u \rightarrow v$ means that caller u invokes callee v. For an incident window, telemetry X contains metric, log-feature, and trace-feature time series for every observable service. A single injected root event is represented by $y=(s^*, f^*)$, where $s^* \in V$ is the origin service and f^* belongs to the fault set $F=\{\text{CPU, memory, disk I/O, delay, packet loss, socket exhaustion}\}$. The online task is to produce a ranking π over V and a fault-type prediction f-hat. Service localization is evaluated with Top-k accuracy and mean reciprocal rank (MRR); fault typing uses accuracy; joint accuracy requires both Top-1 service correctness and fault correctness.

The design follows four operational constraints. C1: a candidate must be an observed service in V. C2: semantic plausibility cannot compensate for absent measured evidence. C3: missing telemetry must reduce that modality's contribution rather than cause failure. C4: the same input, topology, prior, and code must produce the same ranking. These constraints rule out unconstrained natural-language generation as the final decision procedure and motivate the deterministic architecture in Fig. 1.

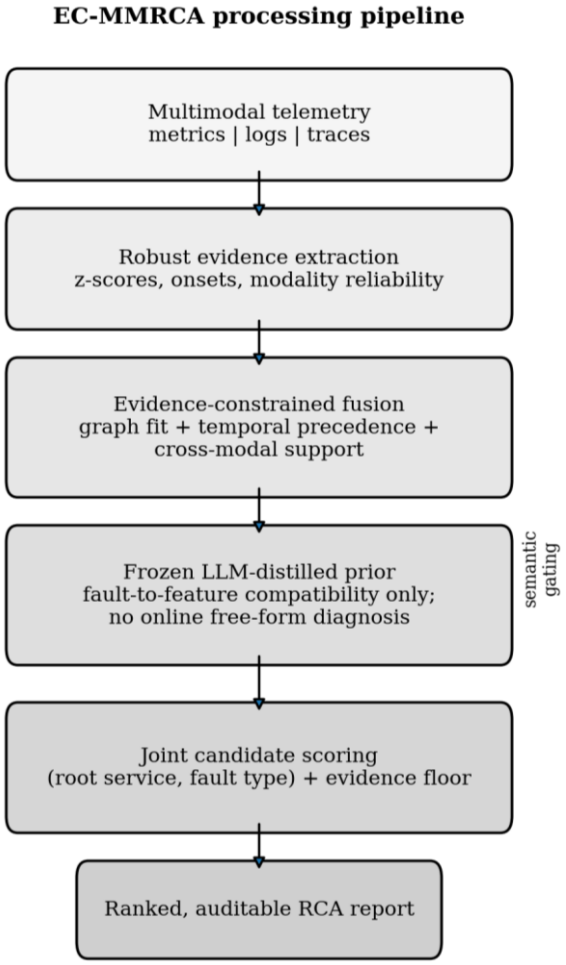

Figure 1. EC-MMRCA separates offline semantic distillation from deterministic, evidence-constrained online ranking.

4. Ec-Mmrca Framework

4.1. Robust Feature Evidence

For feature j and service s, the pre-incident segment defines a median μ_js and a robust scale 1.4826·MAD_js+0.08. The post-incident observations are transformed into positive robust z-scores; the 80th percentile provides a persistent anomaly magnitude a_j(s), avoiding sensitivity to one-sample spikes. The first post-injection index whose z-score exceeds 3 is recorded as onset o_j(s). Missing modalities are represented by unavailable arrays and contribute zero evidence rather than imputed anomalies.

$$z_{jst} = \max\{(x_{jst} - \mu_{js})/\sigma_{js}, 0\}, \qquad \sigma_{js} = 1.4826\ \mathrm{MAD}(x_{js}^{pre})+0.08 \tag{1}$$

$$a_j(s) = P_{80}(\{z_{jst}: t \text{ in post window}\}) \tag{2}$$

4.2. Adaptive Modality Reliability

Within each modality m, feature magnitudes are averaged into A_m(s). Reliability is estimated from the contrast between the strongest service and the modality's background median, normalized by its median absolute spread. Clipping prevents degenerate values, square-root compression prevents one modality from dominating, and normalized weights w_m sum to one. If a modality is completely missing, its reliability is zero. The resulting evidence vector E captures the incident's service-wise anomaly distribution.

$$r_m = \mathrm{clip}((\max_s A_m(s) - \mathrm{med}_s A_m(s))/(\mathrm{MAD}_s(A_m)+\varepsilon), 0.2, 8) \quad (3)$$

$$w_m = \sqrt{r_m}/\Sigma_q \sqrt{r_q}, \quad E(s)=\Sigma_m w_m A_m(s) \quad (4)$$

4.3. Frozen LLM-Distilled Semantic Prior

During study preparation, an instruction-tuned LLM was asked to assign nonnegative relevance weights between concise cloud-native fault definitions and observable feature names. Direct physical indicators were preferred over propagated symptoms. The resulting matrix is stored in llm_prior.json, including its prompt summary and constraints. It is not regenerated during experiments, and the online method sends no telemetry to an external model. For fault f, semantic evidence $L_f(s)$ is a weighted average of the measured feature magnitudes. Thus, even a high prior weight produces no incident evidence when the corresponding feature is normal or unavailable.

$$L_f(s) = (\Sigma_j \alpha_{fj} a_j(s))/(\Sigma_j |\alpha_{fj}|+\varepsilon) \quad (5)$$

The prior gives, for example, high weight to CPU utilization for CPU saturation, memory utilization for memory pressure, disk I/O for disk contention, span latency for delay, and retry/error indicators for packet loss or socket exhaustion. Fault-specific log signatures receive substantial but not exclusive weight because they may be missing or propagated. This design uses the LLM for semantic compression, not for incident-specific causal claims. Mo et al.'s LS-CM presents a well-designed dual-memory architecture separating short-term contextual retrieval from long-term strategy abstraction under execution feedback [34], inspiring our separation of incident-local telemetry from a compact, reusable semantic prior.

4.4. Propagation Fit and Temporal Precedence

A fault in callee s primarily propagates to its callers. For candidate s, a graph kernel k_s assigns value 1 to s and δ^d to a service d reverse-call hops away, with $\delta=0.62$; unreachable services receive zero. The graph-fit term G(s) is the cosine similarity between the normalized observed evidence vector and k_s. It acts as a lightweight counterfactual test: if s were the origin, would the spatial pattern of symptoms resemble the expected caller-side footprint?

$$k_s(v)=\delta^{\mathrm{dist}(v, s)} \text{ if } v \text{ reaches } s, \text{ else } 0, \quad G(s)=\cos(E, k_s) \quad (6)$$

For temporal precedence, the earliest detected feature onset for each service is compared with the earliest onset in the incident. $T(s)=\exp(-\Delta o/7)$ rewards early candidates while decaying smoothly rather than imposing a brittle total order. Cross-modal support C(s) is the fraction of available modalities whose service score exceeds that modality's median plus median absolute deviation. Together, G, T, and C help distinguish a causal origin from a delayed but high-amplitude distractor.

$$T(s)=\exp(-(o(s)-\min_v o(v))/7), \quad C(s)=\mathrm{mean}_m \mathbb{1}[A_m(s)>\mathrm{med}(A_m)+\mathrm{MAD}(A_m)] \quad (7)$$

4.5. Evidence-Gated Joint Score

Raw semantic evidence is normalized per fault and gated by $0.55T(s)+0.45G(s)$. The final score for pair (s,f) combines gated semantics, adaptive evidence, graph fit, onset precedence, and cross-modal support. Coefficients are fixed in code and are not fitted to the reported labels. Candidates at or below the 20th percentile of aggregate evidence receive a 0.35 penalty. The service ranking uses the maximum pair score over faults; fault typing uses the largest semantic evidence over observed services.

$$Q(s, f)=0.30 L_f(s)H(s)+0.17E'(s)+0.23G(s) +0.22T(s)+0.08C(s), \quad H(s)=0.55T(s)+0.45G(s) \quad (8)$$

Here E' is max-normalized aggregate evidence. The bounded additive form allows each recommendation to be decomposed into observable contributions. Unlike an online LLM agent, EC-MMRCA cannot create a nonexistent service, silently alter its reasoning strategy, or cite a feature absent from the input schema. Algorithm 1 summarizes the online procedure.

Algorithm 1 Evidence-constrained online ranking

Input: graph G, telemetry X, frozen prior A

1: extract robust magnitudes and anomaly onsets;

2: estimate reliability and aggregate evidence E(s);

3: compute graph fit G(s), precedence T(s), and support C(s);

4: compute and gate semantic compatibility L_f(s);

5: combine terms with (8) and apply the evidence floor;

6: return the service ranking and fault prediction.

4.6. Complexity and Deployment

For |V| services, |J| features, and |F| fault types, feature extraction is O(|J||V|T) for a telemetry window of length T. With precomputed graph distances, online ranking is O(|F||J||V|+|V|²). The prior is small, human-inspectable, and versionable alongside deployment configuration. A production implementation can populate features from OpenTelemetry collectors, update the call graph from traces or service-mesh metadata, and attach the score decomposition to an incident ticket. The proposed framework intentionally stops at diagnosis; it does not execute remediation actions.

5. Reproducible Experimental Design

5.1. Benchmark Construction

The numerical study uses a synthetic benchmark generated entirely by the accompanying Python code. We chose this route because a complete, verified execution on large public telemetry archives was not feasible within the available environment, and copying published scores would violate reproducibility. The generator is constrained by the public RCAEval design space [21]: metrics, structured log features, and trace features; three representative microservice topology families; and six commonly injected fault types. No RCAEval measurement values, incident traces, or baseline results are used in our calculations.

The topology families contain 10 services (boutique), 12 services (sock-shop), and 14 services (train-ticket style). Five non-edge target services are selected in each topology. Every combination of topology, target service, and fault type is repeated three times per random seed. With six faults and three seeds, each experimental condition contains 810 incidents (3 topologies×5 services×6 faults×3 repetitions×3 seeds). The complete study has 2,430 unique incidents across three conditions. Each time series has 120 steps, with the fault introduced at step 60 after a clean baseline segment.

Root anomalies use smooth step responses with small amplitude and onset jitter. Caller-side symptoms are delayed with graph distance and attenuated by 0.62^d. Propagation primarily affects latency, error, retry, and log-count signals. To prevent trivial localization, the generator adds unrelated bursts, incorrect fault signatures, propagated copies of the correct log signature, and a delayed fault-like distractor with strong local telemetry but no matching propagation footprint. The high-noise condition multiplies baseline noise by 1.8. In the missing condition, one of the three modalities is removed per incident (33.3% modality loss; retained filename missing30 for experiment compatibility).

5.2. Baselines and Ablations

Metric-Z ranks services by aggregated metric anomalies and infers fault type from direct metric mappings. MM-Average uses the proposed adaptive aggregate but ignores graph, temporal, and semantic terms. Graph-Fit combines aggregate evidence with the propagation-kernel similarity. LLM-Semantic uses only the frozen fault-feature matrix. These baselines isolate the contributions of raw metrics, generic multimodal fusion, topology, and semantic knowledge. Four ablations remove adaptive weighting, graph fit, semantics, or temporal precedence from EC-MMRCA. They are implementations within the released package rather than claims of exact reproduction of external systems.

5.3. Metrics and Statistical Protocol

Top-1, Top-3, and Top-5 measure whether the true root service appears in the first k recommendations. MRR averages the reciprocal rank. Fault accuracy measures the six-way fault prediction, and joint accuracy requires a correct Top-1 service and fault type. We report means over all 810 incidents per condition and mean±sample standard deviation across three seeds. Ranking-stage runtime is measured with perf_counter after shared feature extraction; it excludes telemetry generation, parsing, and I/O and should be interpreted as relative Python overhead rather than a hardware-normalized latency benchmark.

For each condition, the proposed method is compared with the strongest baseline Top-1 result on the same incident identifiers. We use the exact paired McNemar test on discordant outcomes and a 10,000-sample paired bootstrap confidence interval for the absolute Top-1 difference. The random seed for statistical resampling is fixed. All incident-level predictions, scripts, package versions, and the experiment manifest are included.

5.4. Research Integrity Boundaries

The study makes three explicit boundaries. First, all experimental values are synthetic; no production telemetry is represented. Second, the LLM prior is a frozen artifact included verbatim, not a claim of reproducible access to a particular commercial endpoint. Third, fusion coefficients and the generator were developed in the same study, so design bias is possible even though no supervised training or test-label optimization is performed. These boundaries are carried into the interpretation and threats-to-validity section.

6. Results

6.1. Standard Condition

Table I reports the standard condition. EC-MMRCA achieves 86.3% Top-1, 99.8% Top-3, and 0.929 MRR. The strongest baseline for Top-1 is Metric-Z at 69.6%, producing an absolute gain of 16.7 percentage points and a relative gain of 23.9%. The proposed method's fault accuracy is 79.6%, equal to the semantic-only baseline, while its joint service-and-fault accuracy rises to 69.0%. This distinction is important: semantics identifies fault class reasonably well, but graph and onset evidence are needed to assign that class to the originating service.

Table 1. Standard-Condition Results (% Except Mrr).

Method	T1	T3	MRR	Fault	Joint
Metric-Z	69.6	98.8	0.838	63.6	44.8
MM-Average	31.2	89.9	0.593	68.3	18.1
Graph-Fit	50.9	97.7	0.731	68.3	31.7
LLM-Semantic	53.3	96.7	0.739	79.6	49.6
EC-MMRCA	86.3	99.8	0.929	79.6	69.0

Top-3 is high for several methods because each topology has only 10–14 services and propagated symptoms usually keep the origin near the front. Top-1 is therefore the more discriminative endpoint. Graph-Fit improves over generic multimodal averaging but remains vulnerable when a delayed distractor also fits part of the call graph. Metric-Z performs strongly on service localization because several root faults have direct resource indicators, yet it loses fault and joint accuracy when error-related faults share similar metric manifestations. EC-MMRCA succeeds by requiring semantic, spatial, and temporal cues to agree.

6.2. Noise and Missing Telemetry

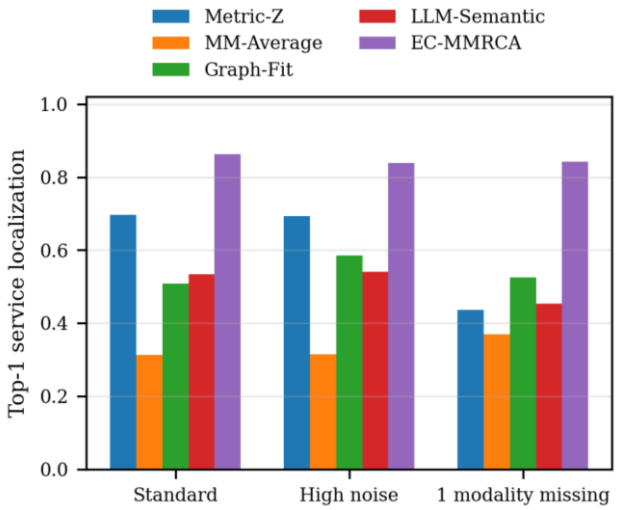

Figure 2. Top-1 service localization under standard, high-noise, and one-modality-missing conditions.

Under high noise, EC-MMRCA reaches 83.8% Top-1, only 2.5 percentage points below its standard result. It exceeds Metric-Z by 14.4 points. When one entire modality is absent, EC-MMRCA retains 84.2% Top-1; the strongest baseline becomes Graph-Fit at 52.6%, yielding a 31.6-point gain. Adaptive reliability sets the missing modality weight to zero and renormalizes the remaining sources, while graph and temporal cues compensate for reduced feature coverage.

Table 2. Paired Top-1 Comparisons (%).

Condition	EC	Best base	Base	Gain	p
Standard	86.3	Metric-Z	69.6	+16.7	1.5e-17
High noise	83.8	Metric-Z	69.4	+14.4	9.9e-13
1 modality missing	84.2	Graph-Fit	52.6	+31.6	5.5e-56

The paired tests reject equality in all three conditions. In the standard condition, EC-MMRCA wins 197 discordant incidents while Metric-Z wins 62; the exact McNemar p-value is 1.46×10^{-17} and the paired bootstrap 95% interval for the absolute gain is [12.8, 20.4] points. The corresponding intervals are [10.6, 18.3] under high noise and [28.0, 35.3] with missing telemetry. Because the incidents are synthetic and not independently sampled production failures, these p-values quantify consistency within the generator rather than population-level deployment certainty.

6.3. Component Ablation

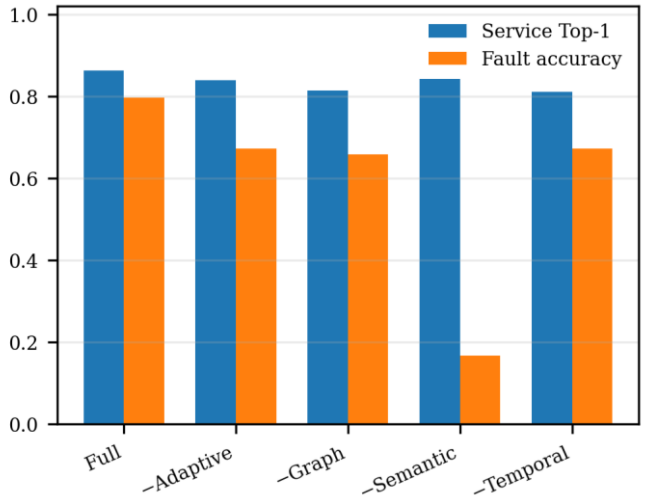

Figure 3. Ablation results under the standard condition.

Removing temporal precedence causes the largest service-localization decline, from 86.3% to 81.1% Top-1. Removing graph fit reduces Top-1 to 81.4%, confirming that spatial propagation and onset order are complementary. Replacing adaptive reliability with an unweighted modality average yields 84.0%. Removing semantics leaves service Top-1 at 84.3% but collapses fault accuracy to chance level (16.7%), demonstrating that semantic knowledge is primarily responsible for fine-grained fault typing while causal telemetry structure determines the origin service.

Table 3. Ablation Results (% Except Mrr).

Variant	T1	T3	MRR	Fault
Full	86.3	99.8	0.929	79.6
w/o adaptive	84.0	99.9	0.918	67.2
w/o graph	81.4	99.3	0.902	65.8
w/o semantic	84.3	99.9	0.920	16.7
w/o temporal	81.1	100.0	0.902	67.2

6.4. Fault and Topology Analysis

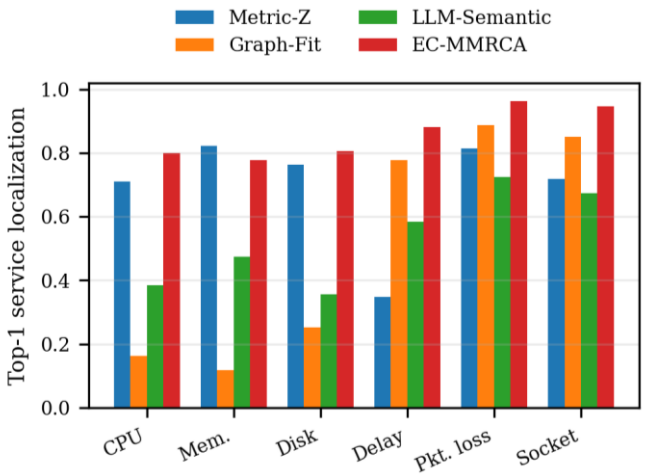

Figure 4. Standard-condition Top-1 localization by fault type.

EC-MMRCA is strongest for packet loss (96.3% Top-1) and socket exhaustion (94.8%), where correlated error, retry, and trace signals create a distinctive propagation footprint. CPU, memory, and disk faults are more difficult because delayed workload bursts can mimic direct resource indicators; Top-1 is 80.0%, 77.8%, and 80.7%, respectively. Delay reaches 88.1%. Fault-type accuracy is lower

for socket exhaustion (67.4%) because socket and packet-loss incidents share error and retry manifestations, an ambiguity deliberately retained in the generator.

Across topologies, proposed Top-1 is 89.3% for boutique, 82.2% for sock-shop, and 87.4% for train-ticket. The 7.0-point range indicates that the score is not tied to a single graph, although all graphs are generated from fixed templates. Sock-shop is hardest because several selected services share upstream callers, making their propagation kernels less separable.

6.5. Runtime and Interpretation

The mean ranking-stage runtime of EC-MMRCA is 1.156 ms per incident in the measured Python environment, compared with 0.500 ms for Graph-Fit, 0.068 ms for LLM-Semantic, and 0.035 ms for Metric-Z. EC-MMRCA is roughly 2.3× slower than Graph-Fit and 33× slower than Metric-Z in the ranking stage, but remains below 2 ms. No online LLM request is made, so latency and monetary cost are independent of model endpoints. Shared feature extraction and telemetry collection are excluded and would dominate an end-to-end production pipeline.

The experiment supports a restrained conclusion: an LLM-derived semantic matrix is useful when it is attached to measurable feature evidence and gated by causal cues. Semantic-only ranking is insufficient for origin localization, and graph-only ranking cannot distinguish all distractors. The combined method performs well in the controlled generator and degrades gracefully when noise increases or a modality is removed. It does not establish superiority over published multimodal systems on their datasets, because those systems are not reimplemented here and no external benchmark scores are imported.

6.6. Auditable Incident Case Study

To illustrate the score rather than estimate aggregate performance, we inspect one preselected discordant incident from the 197 standard-condition cases in which EC-MMRCA is correct and Metric-Z is not. The incident is standard-0-0007: CPU saturation originates in the boutique catalog service. A delayed, high-amplitude distractor makes recommendation the largest aggregate anomaly. Metric-Z, MM-Average, and Graph-Fit therefore rank recommendation first, whereas LLM-Semantic and EC-MMRCA rank catalog first and predict CPU correctly.

The adaptive weights are balanced (metrics 0.337, logs 0.326, traces 0.337), so the result is not caused by suppressing a modality. Recommendation has greater normalized aggregate evidence than catalog (1.000 versus 0.663) and substantial CPU semantic compatibility (0.930 versus 1.000). However, catalog has the strongest graph fit (0.922 versus 0.768) and the earliest anomaly (temporal score 1.000 versus 0.867). After semantic gating and fusion, catalog obtains a CPU-pair score of 0.914, compared with 0.847 for recommendation. Cart and frontend remain plausible propagated symptoms but score 0.536 and 0.473.

Table 4. Score Decomposition For Incident Standard-0-0007.

Service	E'	Lcpu	G	T	C	Q
catalog	0.663	1.000	0.922	1.000	1.00	0.914
recommendation	1.000	0.930	0.768	0.867	1.00	0.847
cart	0.724	0.243	0.624	0.651	1.00	0.536
frontend	0.764	0.466	0.469	0.424	1.00	0.473

This case exposes the intended safeguard: anomaly magnitude alone can select a convincing secondary event, and semantics alone can localize correctly only when the relevant signature is available. EC-MMRCA makes the decision auditable by showing exactly how temporal precedence and propagation consistency overturn the larger distractor. The table is computed by src/case_study.py; no explanatory text from an online LLM participates in the ranking.

7. Threats To Validity And Limitations

Construct validity is the primary limitation. The generator encodes assumptions about how six faults affect metrics, logs, traces, and callers. Although distractors, missing signatures, noise, propagation delays, and modality loss make the task nontrivial, real incidents may exhibit gradual degradation, multiple simultaneous causes, asynchronous queues, autoscaling, retries across circuit breakers, schema drift, or operator interventions. The semantic prior and generator share the same fault vocabulary, which may favor fault-type classification. The reported figures must therefore be read as controlled evidence about the scoring design, not as deployment accuracy.

Internal validity is affected by method-generator co-development. Fusion coefficients are fixed and no label-trained optimization is performed, but design choices were made with awareness of the synthetic process. Three seeds quantify only a limited portion of generator variability. Runtime measurements exclude feature extraction and use a shared, non-dedicated environment. Exact reruns may show small timing changes even though rankings and accuracy are deterministic.

External validity is not established. The study does not execute on RCAEval, OpenRCA, industrial incidents, Kubernetes events, or black-box services. RCAEval provides a natural next evaluation target [21], while TORAI emphasizes the need to handle missing call-graph coverage [24]. A real deployment must also address telemetry sampling, clock skew, service aliases, version changes, confidential logs, and multiple root causes. The current graph kernel assumes caller-side propagation and may be inappropriate for event-driven or data-pipeline topologies.

The LLM component is intentionally narrow. The frozen prior is reproducible as an artifact, but its original stochastic generation is not recreated from a named model checkpoint. This avoids false claims about model-specific performance but limits conclusions about different LLMs or prompts. Future work should compare expert-authored, open-model-generated, and learned priors under blinded evaluation; calibrate confidence; incorporate evidence retrieval for explanations; test graph uncertainty; and validate recommendations with SRE review before any automated remediation. Rhee et al.'s TAPE provides an elegant trajectory-learning mechanism that contrastively aligns successful and failed executions in a curated dual-stream experience pool [35], suggesting a principled route for future RCA prior refinement from verified diagnoses and rejected hypotheses.

8. Artifact Availability And Reproduction

The accompanying artifact is designed to let a reviewer audit both the algorithm and every aggregate value without external services. The root directory contains the benchmark generator, ranking methods, experiment driver, post-processing scripts, figure generator, frozen semantic prior, environment requirements, and primary-source reference verification log. The results directory contains incident-level predictions rather than only summary tables, so alternative metrics or error analyses can be computed without rerunning the generator.

A full reproduction uses three commands: python src/run_experiments.py --seeds 3 --repetitions 3, followed by python src/analyze_results.py and python src/make_figures.py. The first command reconstructs 2,430 incidents and writes 15,390 method-incident rows: five main methods under all three conditions plus four ablations under the standard condition. The analysis script derives joint accuracy, topology results, exact McNemar tests, and fixed-seed bootstrap intervals. The figure script reads only saved CSV files. No network access, model API, or downloaded benchmark is required.

Table 5. Artifact Contents And Checks.

Artifact	Role	Integrity check
llm_prior.json	Frozen semantic weights	SHA-256 9e65301d28f
incident_level_results.csv	15,390 predictions	SHA-256 b361f4dc67d6
summary_results.csv	15 aggregate rows	SHA-256 24ffa8a421e1
statistical_tests.csv	3 paired tests	SHA-256 39b80e3ff1cb
references_verified.csv	35 source checks	Primary-source URLs/DOIs

The SHA-256 prefixes in Table V correspond to the delivered files; the complete hashes are stored in results/sha256sums.txt. The experiment manifest records Python 3.13.5, NumPy 2.3.5, and pandas 2.2.3, together with seeds, repetitions, fault types, and incident counts. Functional rankings are deterministic for the supplied versions and seeds. Runtime columns are intentionally not checksum-stable because operating-system scheduling and processor load affect perf_counter measurements; re-evaluation should compare their order of magnitude, not exact microseconds.

Two audit paths are supported. A lightweight audit can inspect the manifest, verify file hashes, recompute all paper tables from incident_level_results.csv, and compare the outputs with the delivered summaries. A full audit regenerates telemetry and predictions before aggregation. The case-study decomposition is separately produced by src/case_study.py. This artifact structure also makes negative results visible: all baseline errors and proposed-method failures remain in the incident-level file rather than being filtered from the release.

9. Conclusion

EC-MMRCA integrates LLM-derived fault semantics with multimodal telemetry without delegating the final diagnosis to free-form generation. Its deterministic score combines adaptive evidence, graph propagation fit, temporal precedence, and cross-modal support, then rejects low-evidence candidates. On 2,430 disclosed synthetic incidents, the method achieves 86.3% standard Top-1 service localization and remains above 83% under high noise and one-modality loss. Ablations show that semantics is essential for fault typing, while graph and temporal evidence are essential for locating the origin. The accompanying code, incident-level results, figures, manifest, and verified references make every numerical claim auditable. The next necessary step is independent evaluation on public and production telemetry; until then, the results should be treated as reproducible proof-of-concept evidence rather than real-world performance guarantees.

References

[1] Sigelman, B. H., et al. (2010). Dapper, a large-scale distributed systems tracing infrastructure. Google Research.

[2] Gan, Y., et al. (2019). An open-source benchmark suite for microservices and their hardware-software implications for cloud and edge systems. In Proceedings of the ASPLOS (pp. 3–18). https://doi.org/10.1145/3297858.3304013

[3] OpenTelemetry Authors. (2026). OpenTelemetry semantic conventions, Version 1.43.0. OpenTelemetry Specification.

[4] He, P., Zhu, J., Zheng, Z., & Lyu, M. R. (2017). Drain: An online log parsing approach with fixed depth tree. In Proceedings of the IEEE ICWS (pp. 33–40). https://doi.org/10.1109/ICWS.2017.13

[5] Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the ACM CCS (pp. 1285–1298). https://doi.org/10.1145/3133956.3134015

[6] Meng, W., et al. (2019). LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In Proceedings of the IJCAI (pp. 4739–4745).

[7] Zhang, X., et al. (2019). Robust log-based anomaly detection on unstable log data. In Proceedings of the ESEC/FSE (pp. 807–817). https://doi.org/10.1145/3338906.3338931

[8] Wu, L., Tordsson, J., Elmroth, E., & Kao, O. (2020). MicroRCA: Root cause localization of performance issues in microservices. In Proceedings of the IEEE/IFIP NOMS (pp. 1–9). https://doi.org/10.1109/NOMS47738.2020.9110353

[9] Guo, X., et al. (2020). Graph-based trace analysis for microservice architecture understanding and problem diagnosis. In Proceedings of the ESEC/FSE (pp. 1387–1397). https://doi.org/10.1145/3368089.3417066

[10] Liu, D., et al. (2021). MicroHECL: High-efficient root cause localization in large-scale microservice systems. In Proceedings of the ICSE-SEIP (pp. 338–347). https://doi.org/10.1109/ICSE-SEIP52600.2021.00043

[11] Li, Z., et al. (2021). Practical root cause localization for microservice systems via trace analysis. In Proceedings of the IEEE/ACM IWQoS (pp. 1–10). https://doi.org/10.1109/IWQOS52092.2021.9521340

[12] Yu, G., et al. (2021). MicroRank: End-to-end latency issue localization with extended spectrum analysis in microservice environments. In Proceedings of The Web Conf. (pp. 3087–3098). https://doi.org/10.1145/3442381.3449905

[13] Budhathoki, K., Minorics, L., Blöbaum, P., & Janzing, D. (2022). Causal structure-based root cause analysis of outliers. In Proceedings of the ICML (Vol. 162, pp. 2357–2369).

[14] Lee, C., Yang, T., Chen, Z., Su, Y., & Lyu, M. R. (2023). Eadro: An end-to-end troubleshooting framework for microservices on multi-source data. In Proceedings of the ICSE (pp. 1686–1698). https://doi.org/10.1109/ICSE48619.2023.00150

[15] Yu, G., et al. (2023). Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data. In Proceedings of the ESEC/FSE. https://doi.org/10.1145/3611643.3616249

[16] Gu, S., et al. (2023). TrinityRCL: Multi-granular and code-level root cause localization using multiple types of telemetry data in microservice systems. IEEE Transactions on Software Engineering, 49(5). https://doi.org/10.1109/TSE.2023.3241299

[17] Zhu, Z., Lee, C., Tang, X., & He, P. (2024). HeMiRCA: Fine-grained root cause analysis for microservices with heterogeneous data sources. ACM Transactions on Software Engineering and Methodology. https://doi.org/10.1145/3674726

[18] Pham, L., Ha, H., & Zhang, H. (2024). BARO: Robust root cause analysis for microservices via multivariate Bayesian online change point detection. Proceedings of the ACM on Software Engineering, 1(FSE). https://doi.org/10.1145/3660805

[19] Wang, Y., Zhu, Z., Fu, Q., Ma, Y., & He, P. (2024). MRCA: Metric-level root cause analysis for microservices via multi-modal data. In Proceedings of the ASE (pp. 1057–1068). https://doi.org/10.1145/3691620.3695485

[20] Tao, L., et al. (2024). Giving every modality a voice in microservice failure diagnosis via multimodal adaptive optimization. In Proceedings of the ASE. https://doi.org/10.1145/3691620.3695489

[21] Pham, L., Zhang, H., Ha, H., Salim, F., & Zhang, X. (2025). RCAEval: A benchmark for root cause analysis of microservice systems with telemetry data. In Proceedings of The Web Conf. https://doi.org/10.1145/3701716.3715290

[22] Xu, J., et al. (2025). OpenRCA: Can large language models locate the root cause of software failures? In Proceedings of the ICLR.

[23] Tang, P., Tang, S., Pu, H., Miao, Z., & Wang, Z. (2025). MicroRCA-Agent: Microservice root cause analysis method based on large language model agents. arXiv preprint arXiv:2509.15635.

[24] Pham, L., Ha, H., Zhang, X., & Zhang, H. (2026). TORAI: Multi-source root cause analysis for blind spots in the microservice service call graph. Proceedings of the ACM on Software Engineering, 3(FSE), 2928–2951. https://doi.org/10.1145/3808137

[25] Wei, J., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. In Advances in Neural Information Processing Systems (Vol. 35, pp. 24824–24837).

[26] Yao, S., et al. (2023). ReAct: Synergizing reasoning and acting in language models. In Proceedings of the ICLR.

[27] Lewis, P., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in Neural Information Processing Systems (Vol. 33, pp. 9459–9474).

[28] Yao, S., et al. (2023). Tree of thoughts: Deliberate problem solving with large language models. In Advances in Neural Information Processing Systems (Vol. 36).

[29] Zelikman, E., et al. (2022). STaR: Bootstrapping reasoning with reasoning. In Advances in Neural Information Processing Systems (Vol. 35).

[30] Soldani, J., & Brogi, A. (2023). Anomaly detection and failure root cause analysis in (micro)service-based cloud applications: A survey. ACM Computing Surveys, 55(3), 59. https://doi.org/10.1145/3501297

[31] He, S., et al. (2022). A survey on automated log analysis for reliability engineering. ACM Computing Surveys, 54(6). https://doi.org/10.1145/3460345

[32] Ping, W., Jiao, Y., Fan, H., & Zhang, X. (2026). Multimodal fraud detection in financial statements: A trimodal attention network with contrastive evidence chain construction. IEEE Access, 14, 80456–80468. https://doi.org/10.1109/ACCESS.2026.3695466

[33] Zhao, W., Chen, T., Yang, J. S., & Qiu, L. (2026). AutoML-Pipeline: A RAG-enhanced code generation framework with pre-validation for cloud-native machine learning workflows. IEEE Access, 14, 41932–41945. https://doi.org/10.1109/ACCESS.2026.3673923

[34] Mo, T., Zhang, C., Zou, J., Guo, Z., & Rhee, M. (2026). Self-evolving AI agents with dual memory for automated software testing and bug localization. IEEE Access, 14, 111086–111102. https://doi.org/10.1109/ACCESS.2026.3713401

[35] Rhee, M., Zou, J., Mo, T., Teng, D., & Yang, J.-S. (2026). Enhancing web search agents with self-play contrastive fine-tuning. IEEE Access. https://doi.org/10.1109/ACCESS.2026.3717594