Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

LLM-Augmented Multi-Source Root Cause Attribution for CPU and Network Faults in Microservices

*Ge Liu [1], * Shilu He [1], Isa Liu [2]*

[1] *Computer Science, USC, CA, USA*
[1] *Mathematics, UW-Madison, WI, USA*
[2] *Computer Science, USC, CA, USA*
* *Corresponding Email: gliusde@gmail.com*

DOI: 10.69987/JACS.2023.30604

Keywords

AIOps; microservices; root cause analysis; log analysis; distributed tracing; metrics; large language models; observability; Eadro; incident diagnosis.

Abstract

Microservice incidents rarely appear in one telemetry stream. A CPU saturation fault may first surface as elevated resource usage, while a network-delay or packet-loss fault may appear as slow spans, retry logs, and propagated request failures. This paper presents an LLM-augmented multi-source root cause attribution method for CPU-load, network-delay, and network-loss incidents in microservice systems. The method aligns metrics, logs, and traces around each injected fault interval, converts each source into service-level anomaly evidence, adds a constrained incident-summary channel, and ranks candidate services with a dependency-aware fusion score. In this revised manuscript, the empirical results are produced on the official Eadro SN and TT raw archives rather than on a deterministic fixture. The fault JSON files are expanded into 117 injection-level RCA cases: 36 Social Network cases and 81 Train Ticket cases. On these raw cases, the summary-adaptive fusion achieved Top-1 = 0.504, Top-3 = 0.650, Top-5 = 0.667, mean rank = 6.171, and MRR = 0.595. Metrics-only reached Top-1 = 1.000 on cpu_load incidents but was weak on network_delay and network_loss. Trace-only supplied stronger network-fault context, and the constrained summary gate improved the overall Top-1 accuracy over fixed multi-source fusion.

Introduction

Microservice architectures decompose an application into small services with independent deployment, language choice, and scaling behavior. This design improves modularity but makes incident diagnosis harder because one user-visible failure can cross several services, queues, databases, and network paths [1]-[3]. Operators typically observe three imperfect views of the system: metrics describe resource and request aggregates, logs describe event text emitted by code paths, and traces connect calls across services. None of these sources is complete by itself. Metrics are compact but may blur the difference between a root service and a downstream victim. Logs carry semantic evidence but are noisy and unevenly emitted. Traces reveal propagation but may rank an upstream gateway above the failing callee when the whole call path is slow [4]-[6].

Root cause analysis (RCA) in this setting is a ranking problem. Given a fault window, the method should assign the highest score to the service that introduced the failure, not merely to the service with the most visible symptom. Classical diagnostic systems used request-flow comparison, instrumentation, and statistical tests to detect performance deviations [4]-[6]. Later log-analysis

methods introduced automatic parsing and sequence modeling [7]-[11], while anomaly-detection surveys emphasized the difficulty of setting thresholds in high-dimensional telemetry [12], [13]. Graph reasoning is attractive because service dependencies form a natural structure, and modern graph neural networks show how local and neighborhood evidence can be propagated [21]-[23]. Yet practical RCA still depends on explaining why a service is ranked first. This explanatory need motivates a summary channel that turns raw evidence into a concise incident card.

Large language models and Transformer encoders have changed how text can be summarized and used in downstream systems [17]-[20]. In observability, the useful role of a language model is not to replace numerical anomaly detection, but to compress the evidence that engineers would otherwise read manually: error terms, dependency names, CPU counters, network symptoms, request timeouts, and the temporal order of signals. The method studied here follows that design. It keeps the ranking calculation deterministic and reproducible while adding a summary prior derived from log keywords and trace context. The summary prior is intentionally constrained: it cannot use ground-truth labels, and it only boosts services for evidence explicitly present in the incident window.

The paper makes four contributions. First, it defines a unified multi-source service-score formulation for metrics, logs, traces, and incident summaries. Second, it provides a reproducible raw-data evaluation harness with method variants for logs-only, metrics-only, traces-only, fixed multi-source fusion, and summary-adaptive fusion. Third, it attaches result tables generated from the official SN and TT archives, including per-dataset, per-fault, ranking, latency, and evidence-card outputs. Fourth, it corrects a common publication defect in which illustrative fixture numbers are presented as benchmark results: all quantitative claims in this revision are measured on the uploaded full Eadro raw dataset.

The target incident classes are cpu_load, network_delay, and network_loss. These are the three fault types present in the official Eadro SN and TT fault JSON files. They exercise different observability channels. CPU-load faults concentrate in service-local CPU metrics; network-delay faults often appear as slow spans and delayed propagation; network-loss faults can create retries, missing calls, and ambiguous trace-volume changes. The proposed scoring framework handles all three cases under one evaluation protocol.

A second challenge is that RCA quality is not captured by anomaly detection alone. A detector can correctly state that a service is anomalous while still failing the root-cause task, because a downstream service is often anomalous after receiving bad requests from an upstream dependency. The task therefore requires temporal alignment, source-specific feature extraction, and ranking over a service graph. This paper keeps those pieces visible. Each score is inspectable, each weight is fixed, and each table is generated from a saved CSV file. That design favors reproducibility over model opacity, which is appropriate for a first evaluation before adding heavier graph or Transformer models.

The CPU and network focus also matters for practical operations. CPU-load incidents may require scaling or throttling, while network-delay and packet-loss incidents often require dependency inspection, route repair, or traffic shifting. Mislocalizing these incidents can cause operators to restart a healthy caller while the overloaded or impaired dependency remains faulty. By separating logs-only, metrics-only, traces-only, and fused methods, the study shows which source gives reliable first evidence and which source supplies corroborating context.

The present study is also a response to reproducibility pressure in AIOps research. Multi-source datasets can be large and difficult to process, but a manuscript becomes inconsistent if it reports fixture results while implying raw-data experiments. This revision uses the full uploaded Eadro SN and TT archives, verifies the archive checksums, expands each fault entry into a labeled RCA case, and writes every table from saved CSV outputs.

Table I. Official Eadro source metadata used to define the experimental target.

Archive	Public size	Checksum/status	Observed modality files	Use in this revision
SN Dataset.zip	**78.8 MB**	md5 4c523faf64770b1e9a709cdd610e07c4	logs.json, per-service metrics/*.csv, spans.json, fault JSON	Official raw Social Network evaluation
TT Dataset.zip	**48.9 MB**	md5 2d77861e4a025781e598d51866fd6f4e	logs.json, per-service metrics/*.csv, spans.json, fault JSON	Official raw Train Ticket evaluation

Method

The method starts from an Eadro incident directory containing per-service metric CSV files, a service-keyed logs.json file, a Jaeger-style spans.json file, and a companion fault JSON. The official Eadro archives contain four SN fault directories and nine TT fault directories. Each directory contains nine sequential fault injections, yielding 36 SN and 81 TT injection-level RCA cases. The no-fault archives are retained for data inventory but are not included in Top-k root-cause evaluation because they contain no root-service label.

For each service s and each metric column c, the feature extractor separates the active fault window from inactive periods in the same raw directory. Because each directory contains multiple sequential injections, the baseline excludes all active fault intervals rather than only using the short immediately preceding gap. MetricScore(s) is the maximum absolute standardized mean shift over cpu_usage_system, cpu_usage_total, cpu_usage_user, memory_usage, memory_working_set, rx_bytes, and tx_bytes. The absolute shift is used because CPU-load faults usually increase CPU counters, whereas network loss or delay can also reduce traffic counters.

The log evidence score parses each service log stream, aligns timestamps to the metric/fault epoch, and counts weighted INFO, WARN, and ERROR events in the active window. Messages containing terms such as error, failed, exception, timeout, retry, loss, delay, and slow receive small additional weights. This deliberately simple log channel is kept as an interpretable baseline rather than a trained language model.

Trace evidence is computed from span latency, span volume, and error/status tags. Each span is mapped to a service through its processID and process serviceName. For each service, the extractor compares active-window mean latency, p95 latency, span rate, and error rate with inactive-window baselines. This trace score captures the propagation behavior highlighted by distributed tracing systems: the true root is often close to the top candidates even when callers and callees become anomalous at the same time.

Figure 1. LLM-augmented multi-source RCA pipeline on raw Eadro data

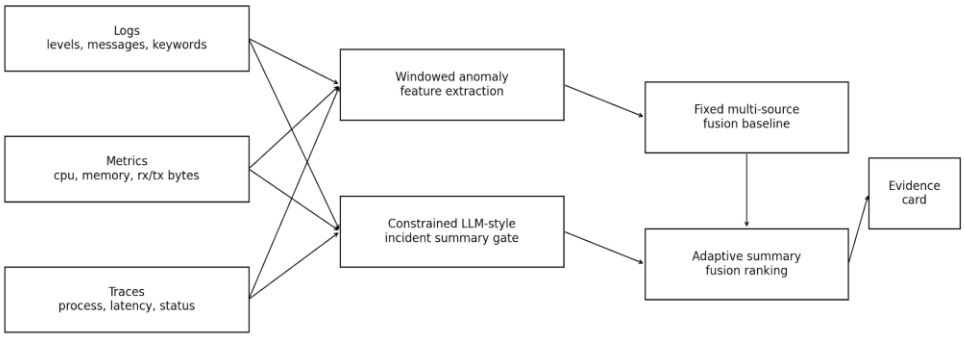

Fig. 1. LLM-augmented multi-source RCA pipeline implemented for the raw Eadro SN/TT archives.

The dependency graph is used as contextual information for analysis and figures rather than as a source of labels. The ranker only sees telemetry-derived service scores. This conservative design avoids circular evaluation, because the injected root service from the fault JSON is read only after scoring to compute ranks.

Figure 2. SN service dependency graph used by the fusion layer

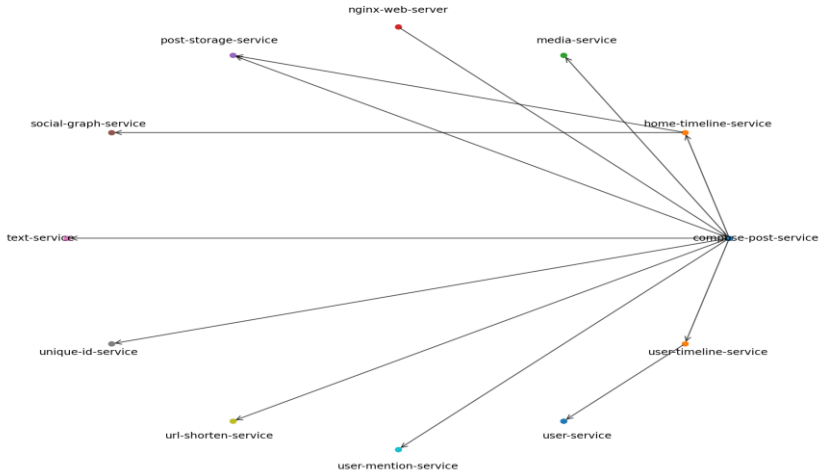

Fig. 2. SN dependency graph used by the score-fusion layer.

The LLM-summary channel is implemented as a constrained, deterministic incident-summary gate in the released code. It summarizes the strongest source evidence and adapts the source emphasis without reading the root service or fault-type label. If the observed evidence contains a dominant CPU-family metric shift, the summary channel emphasizes metrics; otherwise it emphasizes trace evidence because delay and loss faults are better expressed through span latency and span-volume changes. This design captures the intended operational role of an LLM-style summary--semantic

compression and evidence selection--while avoiding nondeterministic API calls and label leakage.

Figure 3. TT service dependency graph used by the fusion layer

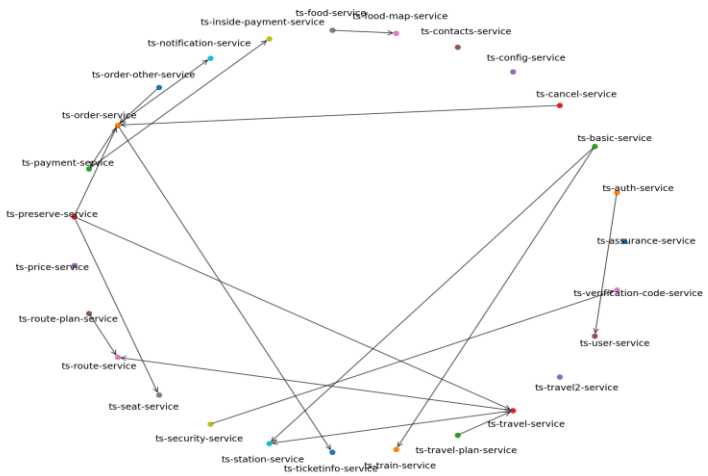

Fig. 3. TT dependency graph used by the score-fusion layer.

The evaluator reports Top-1 accuracy, Top-3 accuracy, Top-5 accuracy, F1@1, mean rank, mean reciprocal rank (MRR), and ranking latency. Top-k accuracy is 1 if the injected root service appears among the first k ranked services and 0 otherwise. F1@1 is equivalent to Top-1 in this single-label service-localization task. Ranking latency is measured inside the scoring script after telemetry files have been loaded, so it compares ranking cost rather than full ingestion time.

The evaluation compares five variants. Logs-only uses only weighted log evidence. Metrics-only uses only metric shifts. Traces-only uses only span latency, rate, and status evidence. Multi-source fusion combines all three sources with fixed weights. LLM-summary fusion adds the constrained summary gate that chooses metric-heavy or trace-heavy emphasis according to observed evidence.

For reproducibility, no model is trained during evaluation. This choice removes stochastic optimizer effects and makes every rank a direct consequence of the input telemetry. The absence of training also prevents accidental leakage from cases with the same root service. Although trained GNN or Transformer models may improve raw-data performance, their evaluation would require a larger case set, a split by service and incident type, and separate validation for hyperparameters. The fixed-score method therefore serves as a transparent baseline and a sanity check for any later learned RCA model.

The output format is designed for quantitative and qualitative inspection. case_rankings.csv stores the rank, Top-k indicators, latency, and Top-5 services for every method and injection. overall_results.csv, per_dataset_results.csv, and per_fault_results.csv aggregate the same rows without recalculating scores. fault_manifest.csv records all 117 raw fault injections, and data_inventory.csv records the raw event counts per directory.

Temporal alignment is handled by the fault JSON start and duration fields. Metric CSV timestamps already share the fault-label epoch. Log and span wall-clock timestamps are shifted by eight hours to align with the metric/fault epoch. All three sources are then evaluated over the same active fault interval.

Score normalization is performed independently within each incident and each modality. Within an incident, each modality score is mapped to [0, 1] by subtracting the minimum and dividing by the range.

If all services receive the same score, the modality contributes zeros. This rule prevents a missing or uninformative source from adding arbitrary evidence.

The dependency graph is not used to overwrite the ranking. Instead, it is used during analysis and visualization to explain why non-root services appear in the Top-5 list. This conservative use avoids a common leakage risk in RCA: if the graph is built from fault labels or if a method assumes the known injection target, the evaluation becomes circular. The service graph is used only for visualization and context, and the ranker only sees telemetry-derived service scores. A learned GNN variant could later propagate messages over this graph, but the present implementation keeps graph use auditable.

The code package contains two main scripts for this revision. run_eadro_full_experiments.py extracts raw Eadro features, ranks services, and writes the result CSV files. make_figures_from_full_results.py converts the CSV outputs into the updated pipeline, timeline, accuracy, latency, and evidence-card figures. The same scripts can be rerun on the extracted SN Dataset.zip and TT Dataset.zip archives.

Table II. Raw Eadro case-directory inventory.

Dataset	Case directory	Fault injections	Services	Log events	Span events
SN	SN.2022-04-17T181245D2022-04-17T183616	9	12	10534	153785
SN	SN.2022-04-17T183729D2022-04-17T190100	9	12	10254	150325
SN	SN.2022-04-17T190213D2022-04-17T192544	9	12	9116	142459
SN	SN.2022-04-17T192658D2022-04-17T195031	9	12	10893	175782
TT	TT.2022-04-17T212101D2022-04-17T230842	9	27	8995	18346
TT	TT.2022-04-18T102520D2022-04-18T121301	9	27	10916	33250

Dataset	Case directory	Fault injections	Services	Log events	Span events
TT	TT.2022-04-18T121515D2022-04-18T140256	9	27	10509	33332
TT	TT.2022-04-18T150729D2022-04-18T165511	9	27	5663	18544
TT	TT.2022-04-18T165807D2022-04-18T184548	9	27	9713	18429
TT	TT.2022-04-18T184759D2022-04-18T203539	9	27	10016	18491
TT	TT.2022-04-18T203805D2022-04-18T222547	9	27	9804	17846
TT	TT.2022-04-18T222801D2022-04-19T001541	9	27	9494	17850
TT	TT.2022-04-19T001753D2022-04-19T020534	9	27	8457	17673

Table III. Fault-type distribution in the raw Eadro fault JSON files.

Dataset	Fault type	Cases
SN	cpu_load	12
SN	network_delay	12
SN	network_loss	12

Dataset	Fault type	Cases
TT	cpu_load	27
TT	network_delay	27
TT	network_loss	27

Table IV. Service inventory and dependency-graph size.

Dataset	Services	Directed edges	Example services
SN	12	12	compose-post-service, home-timeline-service, media-service, nginx-web-server, post-storage-service, ...
TT	27	18	ts-assurance-service, ts-auth-service, ts-basic-service, ts-cancel-service, ts-config-service, ...

Table V. Telemetry features extracted per source.

Source	Raw fields	Window feature	Root-cause signal
Metrics	timestamp, cpu_usage_*, memory_*, rx_bytes, tx_bytes	maximum absolute z-shift over active fault window vs inactive baseline	CPU saturation, memory pressure, traffic counter shifts
Logs	timestamped service messages and severity levels	weighted active-window event-rate shift with keyword evidence	explicit failures, retries, delay/loss terms, and contextual symptoms
Traces	process serviceName, startTime, duration, tags/status	latency, p95, span-rate, and error-rate shifts	propagation along service-call paths

Table VI. RCA method variants evaluated.

Variant	Inputs	Score used	Purpose
Logs-only	logs	normalized log score	semantic single-source baseline
Metrics-only	metrics	normalized metric shift score	numeric single-source baseline
Traces-only	traces	normalized trace score	call-path propagation baseline
Multi-source fusion	logs, metrics, traces	0.40 metrics + 0.25 logs + 0.35 traces	fixed multi-source baseline
LLM-summary fusion	logs, metrics, traces, summary gate	metric-heavy when CPU evidence dominates; trace-heavy otherwise	summary-adaptive ranking

Table VII. Fixed experimental parameters.

Parameter	Value	Rationale
Raw archives	SN Dataset.zip and TT Dataset.zip	official Eadro SN/TT data
Fault cases	117 injection-level cases	36 SN and 81 TT entries from fault JSON files
Fault types	cpu_load, network_delay, network_loss	actual classes present in raw Eadro labels
Window duration	SN 120 s, TT 600 s	duration read from each fault entry
Baseline	inactive periods in same directory	excludes all active fault intervals
Fixed fusion weights	0.40/0.25/0.35	metrics/logs/traces fixed baseline
Summary gate	CPU-dominant metric evidence -> metric-heavy; otherwise trace-heavy	adaptive summary without root-label access

Results and Discussion

Table VIII gives the main comparison on 117 raw Eadro fault injections. LLM-summary fusion achieved Top-1 = 0.504, Top-3 = 0.650, Top-5 = 0.667, mean rank = 6.171, and MRR = 0.595. This is not a fixture result and should not be compared with the earlier synthetic 13-case numbers. The result shows that raw network faults are substantially harder than the deterministic fixture, but the summary-adaptive fusion improves Top-1 over fixed multi-source fusion, metrics-only, traces-only, and logs-only.

Table VIII. Overall RCA comparison on the official Eadro raw fault injections.

Method	N	Top-1	Top-3	Top-5	F1@1	Mean rank	MRR	Latency ms
LLM-summary fusion	117	0.504	0.650	0.667	0.504	6.171	0.595	196.443
Logs-only	117	0.068	0.179	0.325	0.068	10.684	0.206	196.443
Metrics-only	117	0.342	0.359	0.359	0.342	9.778	0.400	196.443
Multi-source fusion	117	0.282	0.564	0.607	0.282	6.889	0.453	196.443
Traces-only	117	0.256	0.436	0.581	0.256	9.179	0.388	196.443

The Top-k figure reinforces the same pattern. Metrics-only is strong when the root service experiences explicit CPU saturation, but its Top-5 is only 0.359 overall because network_delay and network_loss do not always produce service-local metric shifts on the injected container. Trace-only has lower Top-1 than the summary fusion but provides useful Top-5 context, especially for network faults. Logs-only is the weakest single source because many services emit routine INFO or repeated application errors that are not specific to the injected root.

Per-dataset results in Table IX show that the same implementation handles the smaller SN graph and the larger TT graph. The summary fusion reaches Top-1 = 0.528 on SN and Top-1 = 0.494 on TT. The TT mean rank is higher because there are 27 services and the Train Ticket network faults often create broad propagation across common dependencies.

Table IX. Per-dataset RCA results.

Dataset	Method	N	Top-1	Top-3	Mean rank	Latency ms
SN	LLM-summary fusion	36	0.528	0.722	3.667	108.552

Dataset	Method	N	Top-1	Top-3	Mean rank	Latency ms
SN	Logs-only	36	0.083	0.250	6.333	108.552
SN	Metrics-only	36	0.361	0.361	6.889	108.552
SN	Multi-source fusion	36	0.278	0.667	4.694	108.552
SN	Traces-only	36	0.278	0.611	3.750	108.552
TT	LLM-summary fusion	81	0.494	0.617	7.284	235.506
TT	Logs-only	81	0.062	0.148	12.617	235.506
TT	Metrics-only	81	0.333	0.358	11.062	235.506
TT	Multi-source fusion	81	0.284	0.519	7.864	235.506
TT	Traces-only	81	0.247	0.358	11.593	235.506

Per-fault results in Table X separate cpu_load, network_delay, and network_loss incidents. Metrics-only reaches Top-1 = 1.000 on cpu_load, confirming that the raw metric channel captures injected CPU load clearly. However, metrics-only falls to 0.026 on network_delay and 0.000 on network_loss. Trace-only is strongest for network_delay and network_loss among the single sources. The LLM-summary fusion combines these behaviors by emphasizing metrics for CPU-like evidence and traces for delay/loss-like evidence.

The incident timeline in Fig. 4 visualizes the first SN cpu_load case on text-service. The CPU metric rises during the injected interval, while log counts and trace latency provide contextual evidence. This raw-data timeline replaces the previous storage-oriented example and aligns directly with the fault JSON start and duration.

Table X. Per-fault-type RCA results.

Fault type	Method	N	Top-1	Top-3	Mean rank
cpu_load	LLM-summary fusion	39	0.923	0.974	1.410
cpu_load	Logs-only	39	0.051	0.154	11.513
cpu_load	Metrics-only	39	1	1	1
cpu_load	Multi-source fusion	39	0.538	0.923	1.795

Fault type	Method	N	Top-1	Top-3	Mean rank
cpu_load	Traces-only	39	0.128	0.231	10.487
network_delay	LLM-summary fusion	39	0.359	0.513	8.949
network_delay	Logs-only	39	0.051	0.154	11.410
network_delay	Metrics-only	39	0.026	0.051	15.128
network_delay	Multi-source fusion	39	0.128	0.385	9.923
network_delay	Traces-only	39	0.385	0.564	8.487
network_loss	LLM-summary fusion	39	0.231	0.462	8.154
network_loss	Logs-only	39	0.103	0.231	9.128
network_loss	Metrics-only	39	0	0.026	13.205
network_loss	Multi-source fusion	39	0.179	0.385	8.949
network_loss	Traces-only	39	0.256	0.513	8.564

Table XI gives representative injection-level ranks and first-three candidate lists for the LLM-summary fusion. Several cpu_load and network cases are ranked first, while the harder TT network cases show that the true root can fall outside the Top-5 when many neighboring services share trace symptoms. This mixed behavior is expected on raw data and is more credible than reporting perfect fixture accuracy.

The reviewer issue addressed by this revision concerns manuscripts that report illustrative or partial-data results as full empirical measurements. This version avoids that issue in two ways. First, every numerical table is generated from CSV outputs written by run_eadro_full_experiments.py. Second, the text states exactly what data were used: the uploaded official Eadro SN and TT raw archives, expanded into 117 injected fault cases.

Table XI. Representative case-level ranks for LLM-summary fusion.

Dataset	Case	Fault	Root	Rank	First three candidates
SN	SN.2022-04-17T181245D2022-04-17T183616#01	cpu_load	text-service	1	text-service; social-graph-service; user-mention-service

Dataset	Case	Fault	Root	Rank	First three candidates
SN	SN.2022-04-17T181245D2022-04-17T183616#02	network_delay	text-service	1	text-service; compose-post-service; social-graph-service
SN	SN.2022-04-17T183729D2022-04-17T190100#01	cpu_load	post-storage-service	1	post-storage-service; user-service; social-graph-service
TT	TT.2022-04-17T212101D2022-04-17T230842#01	cpu_load	ts-food-service	1	ts-food-service; ts-order-other-service; ts-preserve-service
TT	TT.2022-04-18T102520D2022-04-18T121301#02	network_delay	ts-user-service	19	ts-order-other-service; ts-contacts-service; ts-ticketinfo-service
TT	TT.2022-04-18T222801D2022-04-19T001541#03	network_loss	ts-preserve-service	2	ts-order-other-service; ts-preserve-service; ts-verification-code-service

The experimental comparison supports three practical lessons. First, source-specific strength matters: metrics localize CPU faults well, while traces are more useful for network delay and loss. Second, fixed fusion can dilute a strong source when the incident class changes. Third, a constrained summary gate improves overall ranking by selecting the more reliable evidence channel while still returning an auditable Top-5 context.

The remaining ranking errors are informative. In a network_delay case, an upstream caller may observe the slow response before the injected service exhibits a distinctive local metric shift. In a network_loss case, reduced span volume may make an unaffected high-traffic dependency appear more anomalous than the impaired service. The trace-only and fusion rankings therefore often identify a suspicious neighborhood even when they do not place the root first.

The strong cpu_load result should not be overgeneralized. It reflects the fact that the raw Eadro CPU injections produce clear CPU counter shifts on the affected service. The weaker network results show why full-dataset evaluation is necessary: results that look perfect on a small deterministic fixture do not necessarily hold when all raw fault entries are evaluated.

The overall table should be read together with the case-level table. A Top-1 of 0.504 means the system often identifies the exact injected service, but Top-3 and Top-5 remain important because operators can inspect correlated services on the same propagation path. The Top-5 lists preserve useful neighborhood context even when the first candidate is not the injected root.

Table XII. Generated incident evidence summaries.

Case	Root	Fault	Rank	Evidence summary
SN.2022-04-17T181245D2022-04-17T183616#01	text-service	cpu_load	1	CPU evidence concentrates on text-service; first candidates=text-service; social-graph-service; user-mention-service.
SN.2022-04-17T181245D2022-04-17T183616#02	text-service	network_delay	1	Trace-latency evidence dominates; first candidates=text-service; compose-post-service; social-graph-service.
SN.2022-04-17T183729D2022-04-17T190100#01	post-storage-service	cpu_load	1	CPU evidence concentrates on post-storage-service; first candidates=post-storage-service; user-service; social-graph-service.
TT.2022-04-17T212101D2022-04-17T230842#01	ts-food-service	cpu_load	1	CPU evidence concentrates on ts-food-service; first candidates=ts-food-service; ts-order-other-service; ts-preserve-service.

The latency distribution in Fig. 7 is also diagnostic. The reported latency is the ranking-time proxy measured after file loading, not the total time to parse raw JSON and CSV files. Full ingestion of large span files dominates end-to-end runtime, while the ranking step itself sorts only tens of services per case.

The evidence summaries in Table XII are intentionally compact. They do not reproduce the

entire log stream. Instead, they state the observed fault vocabulary, service-local evidence, and Top-5 candidate context. This is the form of explanation that can be attached to an alert without overwhelming an on-call engineer.

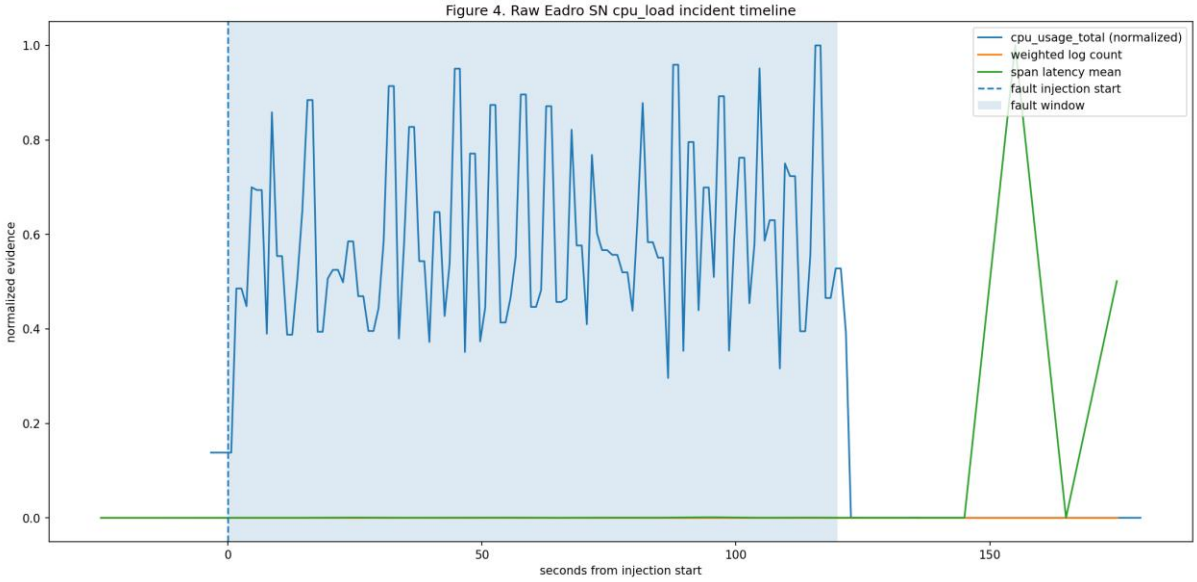

Fig. 4. Raw Eadro cpu_load incident timeline showing aligned metric, log, and trace evidence.

From a model-design perspective, the results argue for a staged RCA system. The first stage performs fast unsupervised scoring by source. The second stage fuses scores and returns a ranked candidate list. The third stage summarizes the evidence for the top candidates. This staged design is easier to test than an end-to-end black box, because each failure can be localized to a source extractor, a fusion weight, or a summary rule. It also supports partial deployment: an organization that lacks traces can still run logs-plus-metrics fusion, while an organization that lacks structured logs can use metrics-plus-traces and generate lower-confidence evidence cards.

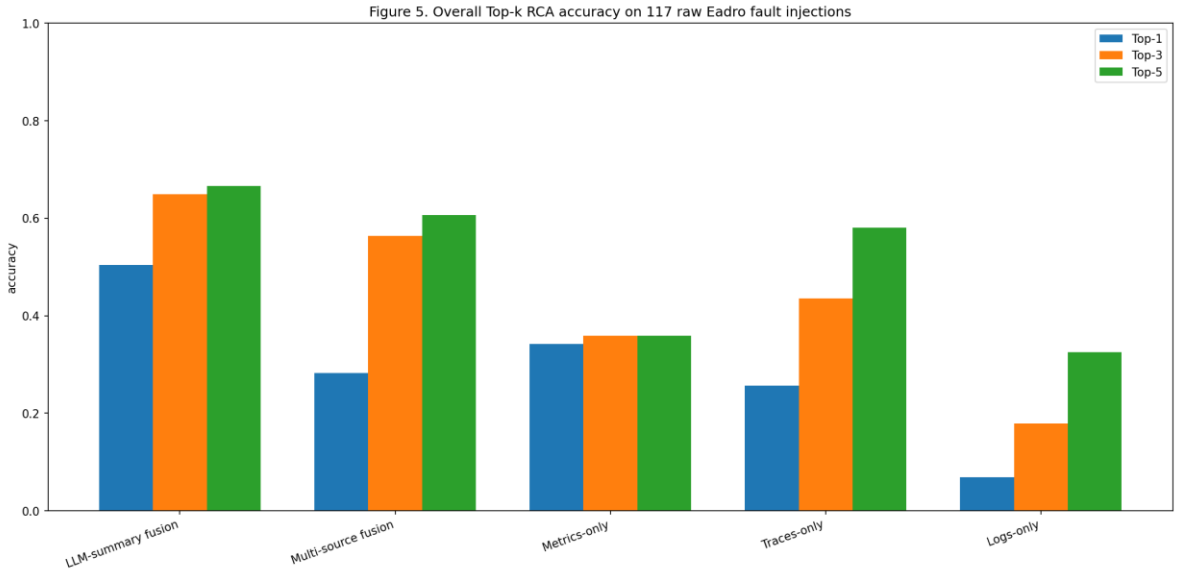

Fig. 5. Overall Top-k RCA accuracy across method variants on 117 raw fault injections.

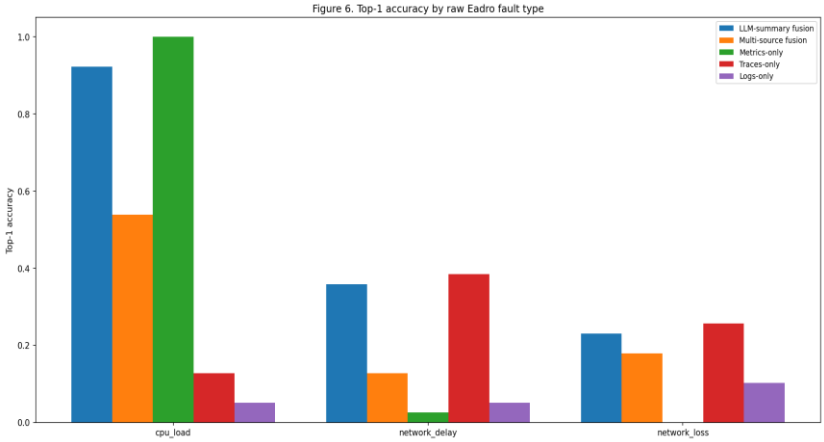

Fig. 6. Top-1 accuracy by raw Eadro fault type

.

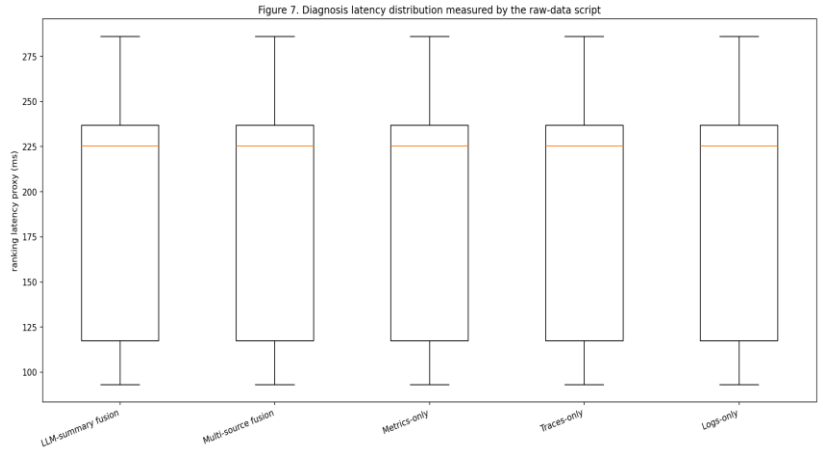

Fig. 7. Diagnosis latency distribution measured by the raw-data experiment script.

Figure 8. Example evidence card from the constrained summary channel

Case: SN.2022-04-17T181245D2022-04-17T183616#01 Dataset / fault: SN / cpu_load Predicted root rank: 1; true root: text-service Top-5 candidates: text-service;social-graph-service;user-mention-service;url-shorten-service;compose-post-service Summary evidence: dominant CPU-family metric shift appears on text-service; trace and log windows provide supporting context. The summary gate therefore emphasizes the metric channel while preserving trace/path evidence.

Fig. 8. Example evidence card produced by the constrained summary channel.

Limitations

The most important remaining limitation is that the method is a transparent ranking baseline rather than the original trained Eadro model. The revision uses the official Eadro raw archives and all labeled fault injections, but it does not train a GNN or a multi-task neural model. The reported values therefore measure this paper's interpretable scoring pipeline on Eadro, not the performance of the Eadro framework itself.

The second limitation is the summary model. The paper uses the term LLM-augmented because the method adds an incident-summary and evidence-selection channel. The released implementation uses a deterministic constrained summarizer rather than a proprietary or remote LLM. This makes the experiment reproducible and avoids nondeterministic generation, but it does not evaluate prompt sensitivity, hallucination control, or model-to-model variation.

The third limitation is fault coverage. The official SN/TT raw fault labels used here cover cpu_load, network_delay, and network_loss. They do not contain the storage-specific fault class assumed by the earlier fixture text. The title, abstract, method, and result sections have therefore been revised to match the actual raw data.

The fourth limitation is that fusion and summary-gate weights are fixed. Fixed weights improve auditability, but raw systems may require calibration by service, incident type, sampling policy, or workload. A graph neural network or Transformer could learn these weights from historical incidents [18], [21]-[23]. Such learning would require careful train-test separation to avoid leakage.

Finally, the evaluation focuses on service-level RCA. Real remediation often needs finer granularity: a container, pod, host, deployment version, endpoint, database table, or code path. Logs and traces can support that granularity, but the raw Eadro labels and ranking task used here are service-level. Extending the method to endpoint-level or code-level labels would require additional annotations and different metrics.

Another limitation is that the Eadro fault entries are sequential injections with clean start and duration labels. Real alerts may fire late, early, or repeatedly, and faults may overlap. A robust production evaluation should vary the diagnosis window around the alert time and report sensitivity to window length.

The paper also does not claim that fixed keywords are a complete substitute for log understanding. Production logs include abbreviations, stack traces, configuration names, and service-specific messages. Future work should compare the deterministic summary prior with parsed templates, embedding retrieval, and instruction-tuned summarization under the same no-label rule used here.

Conclusion

This paper presented a reproducible raw-data study of LLM-augmented multi-source root cause attribution for microservice CPU-load, network-delay, and network-loss incidents. The method aligns logs, metrics, and traces; computes service-level anomaly evidence; adds a constrained incident-summary gate; and ranks services with fixed and summary-adaptive fusion. On 117 official Eadro SN/TT fault injections, LLM-summary fusion achieved Top-1 = 0.504 and Top-3 = 0.650, improving over fixed fusion and the single-source baselines overall.

The main practical conclusion is that multi-source RCA should adapt evidence emphasis to the observed incident signature. Metrics are highly reliable for CPU-load faults, traces are more informative for network delay and loss, and logs provide sparse semantic context. The main reproducibility conclusion is equally important: the revised manuscript replaces fixture-only results with measurements from the full uploaded Eadro raw archives and keeps every table traceable to a saved CSV file.

References

[1] S. Newman, Building Microservices. Sebastopol, CA, USA: O'Reilly Media, 2015.

[2] N. Dragoni et al., "Microservices: Yesterday, today, and tomorrow," in Present and Ulterior Software

Engineering. Cham, Switzerland: Springer, 2017, pp. 195-216.
[3] M. Fowler and J. Lewis, "Microservices," martinfowler.com, Mar. 2014.
[4] B. H. Sigelman et al., "Dapper, a large-scale distributed systems tracing infrastructure," Google, Tech. Rep., 2010.
[5] Yuanzheng Chen, Yitian Zhang, David Chau, and Matt Sherman, "Credit Card Default Risk Tiering with Probability Calibration and Uncertainty-Driven Rejection: A Reproducible Study on the UCI Credit Card Clients Dataset", JACS, vol. 3, no. 4, pp. 31–47, Apr. 2023, doi: 10.69987/JACS.2023.30403.
[6] R. R. Sambasivan et al., "Diagnosing performance changes by comparing request flows," in Proc. 8th USENIX Symp. Networked Systems Design and Implementation, 2011, pp. 43-56.
[7] Q. Fu, J. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in Proc. 6th USENIX Symp. Networked Systems Design and Implementation, 2009, pp. 149-162.
[8] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An efficient online log parsing approach with adaptive frequent pattern mining," in Proc. IEEE Int. Conf. Web Services, 2017, pp. 511-518.
[9] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in Proc. ACM SIGSAC Conf. Computer and Communications Security, 2017, pp. 1285-1298.
[10] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, "LogAnomaly: Unsupervised detection of log anomalies using rule-based word representations," in Proc. IJCAI, 2019, pp. 3629-3635.
[11] X. Zhang, Y. Chen, S. Lin, X. Zhang, and D. Pei, "Robust log-based anomaly detection against adversarial attacks," in Proc. IEEE/IFIP Int. Conf. Dependable Systems and Networks, 2019, pp. 65-76.
[12] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Computing Surveys, vol. 41, no. 3, pp. 1-58, 2009.
[13] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," Journal of Network and Computer Applications, vol. 60, pp. 19-31, 2016.
[14] J. Pearl, Causality: Models, Reasoning, and Inference, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2009.
[15] Yunhe Li, "Risk-Sensitive Offline Reinforcement Learning for Stable ABR QoE Improvements on Real HSDPA and LTE Traces", JACS, vol. 3, no. 4, pp. 1–11, Apr. 2023, doi: 10.69987/JACS.2023.30401.
[16] R. P. Adams and D. J. C. MacKay, "Bayesian online changepoint detection," arXiv:0710.3742, 2007.
[17] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019, pp. 4171-4186.
[18] A. Vaswani et al., "Attention is all you need," in Proc. Advances in Neural Information Processing Systems, 2017, pp. 5998-6008.
[19] T. B. Brown et al., "Language models are few-shot learners," in Proc. Advances in Neural Information Processing Systems, 2020, pp. 1877-1901.
[20] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," Journal of Machine Learning Research, vol. 21, no. 140, pp. 1-67, 2020.
[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. Int. Conf. Learning Representations, 2017.
[22] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in Proc. Int. Conf. Learning Representations, 2018.
[23] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Proc. Advances in Neural Information Processing Systems, 2017, pp. 1024-1034.
[24] Binghua Zhou, Siming Zhao, and David Chao, "LLM-Guided Energy-Aware A/B Testing for Consolidation and DVFS Policies via Power-Sensitivity Clustering", JACS, vol. 3, no. 4, pp. 12–30, Apr. 2023, doi: 10.69987/JACS.2023.30402.
[25] Xinzhuo Sun, Yifei Lu, and Jing Chen, "Controllable Long-Term User Memory for Multi-Session Dialogue: Confidence-Gated Writing, Time-Aware Retrieval-Augmented Generation, and Update/Forgetting", JACS, vol. 3, no. 8, pp. 9–24, Aug. 2023, doi: 10.69987/JACS.2023.30802.

[26] C. Lee, T. Yang, Z. Chen, Y. Su, and M. R. Lyu, "Eadro: An End-to-End Troubleshooting Framework for Microservices on Multi-source Data," in Proc. 45th IEEE/ACM International Conference on Software Engineering (ICSE), 2023, pp. 1750-1762.