JEECS (Journal of Electrical Engineering and Computer Sciences)
Vol. 11, No. 2, December 2026, pp. 135-145
e-ISSN: 2579-5392 p-ISSN: 2528-0260

Correlation-Driven Root Cause Analysis from Multi-Service Logs via Template-Aware Graph Inference: An Empirical Study on LogHub Hadoop and RCAEval Sock Shop

Qi Xin

Management Information Systems, University of Pittsburgh, 4200 Fifth Ave, Pittsburgh, PA 15260, USA

Article Info

Article history:
Received: 8 February 2026
Revised: 30 June 2026
Accepted: 14 July 2026

Keyword:
AIOps
Correlation graph
Microservices root cause
Analysis
System logs

Abstract

Root cause analysis (RCA) has become urgent in AIOps because modern incidents rarely stay inside one component: failures propagate across services and operators must identify the initiating service, not only the loudest symptom. Prior log analytics studies have mainly emphasized parsing and anomaly detection, while recent microservice RCA benchmarks have focused on service localization with multimodal telemetry. This paper studies a narrower and practical question: how far can correlation-driven RCA go when only multi-service logs and their event templates are used? We present a template-aware correlation-graph framework that converts raw logs into service-template events, learns directed lagged edges, and ranks root-cause candidates using direct anomaly evidence, one-step edge attribution, and reverse personalized PageRank. Experiments are conducted on the complete LogHub Hadoop logs and the RCAEval RE2-SS Sock Shop data. The Hadoop corpus contains 394,310 raw log lines, 55 applications, and 44 labeled abnormal applications. RE2-SS contains 90 fault cases, five root services, six fault types, and 7,566,220 log rows. On RE2-SS, the proposed HybridRCA reaches 0.267 Top-1, 0.456 Top-3, and 0.435 MRR for service-level root localization, improving Top-1 and MRR over frequency-only ranking. A metric-only reference reaches 0.578 Top-1, showing that log-only RCA remains limited for resource faults whose strongest symptoms appear in metrics rather than logs. The results support correlation graphs as useful diagnostic evidence while clarifying the unresolved weakness of log-only RCA in multimodal microservice incidents.

Corresponding author:
Qi Xin, qix29@pitt.edu

DOI: https://doi.org/10.54732/jeecs.v11i2.3

1. Introduction

Automated IT operations has moved from detecting anomalous windows toward explaining incidents. In a distributed system, a single failure can trigger retries, timeouts, exception logs, and scheduler reactions across several services. The operational question is therefore not only whether the system is abnormal, but which service-template event most plausibly initiated the incident and which downstream components were affected.

System logs are attractive for this task because they are already collected in many production systems and provide semantic information about executed code paths. LogHub and related toolkits made log parsing and anomaly detection reproducible across public datasets [1], [2], [3]. However, a parsed template sequence or an anomaly flag does not by itself identify an initiating component. Sequence models such as DeepLog, LogAnomaly, and LogBERT improve detection accuracy [4]–[6], while empirical studies show that log-based anomaly detection remains sensitive to parser and feature choices [7], [8].

Recent RCA research has increasingly emphasized graph and correlation reasoning. MicroRCA introduced graph-based localization for microservice systems [9], and Chain-of-Event learns event-causal graphs for interpretable microservice RCA [10]. RCAEval and a related causal-inference evaluation show that synthetic datasets can misrepresent real microservice behavior and that service localization remains difficult [11], [12]. MRCA uses traces and logs to shortlist abnormal services before metric-level causal graph analysis [13]. HeMiRCA, BARO, and CHASE further show the value of heterogeneous telemetry, robust change detection, and causal heterogeneous graphs [14]–[16]. Recent survey work also indicates that root-cause localization is shifting from single-signal anomaly scoring toward graph-centric, multimodal, and auditable diagnosis [17].

These studies motivate the present work, but they also reveal a specific gap: there is still limited empirical evidence on what can be achieved by a log-only, template-aware correlation graph under real service-level RCA labels. Correlation is not causality, yet operations teams often begin diagnosis by asking which event usually precedes the observed symptoms. This paper adopts that pragmatic view and evaluates whether lagged log-template relationships can improve root-cause ranking over frequency-only and service-only baselines.

The contributions are as follows. First, we provide a deterministic pipeline for converting multi-service logs into service-template event streams and directed lagged correlation graphs. Second, we formulate HybridRCA, a scoring method that combines direct anomaly evidence, one-step edge attribution, and reverse personalized PageRank. Third, we evaluate the method on the complete LogHub Hadoop dataset and the real-label RCAEval RE2-SS Sock Shop dataset rather than relying on generated incident labels. Fourth, we analyze sensitivity, ablation behavior, strengths, and remaining weaknesses of log-only RCA.

2. Research Methodology

This section describes the research flow, datasets, preprocessing, graph construction, scoring formulas, and evaluation protocol, as presented in Figure 1. Figure 2 summarizes the steps from data preparation to evaluation, and each step is implemented with fixed parameters so that tables and figures are produced from the same runs.

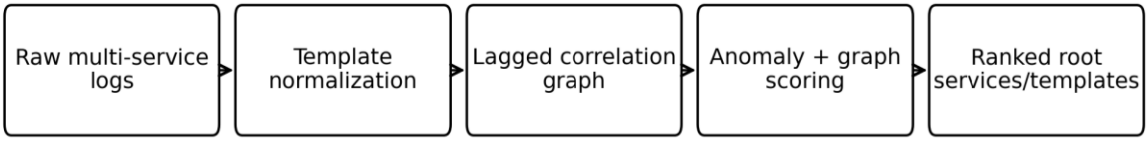

Figure 1. End-to-end architecture of the template-aware correlation-graph RCA pipeline

Table 1. Dataset overview used in the experiments.

Dataset	Role in study	Raw log records	Cases/ applications	Labels	Services	Templates/ nodes
LogHub Hadoop	Log parsing and graph-scale characterization	394310	55	normal, machine down, network disconnection, disk full	7	688
RCAEval RE2-SS	Root-service RCA evaluation	7566220	90	five root services, six fault types	5	861

Table 2. Service distribution and template counts in the complete LogHub Hadoop logs

Service	Lines	Templates
mapreduce	151632	599
ipc	11237	17
yarn	6928	18
hdfs	5483	22
metrics	3885	11
org	972	11
http	759	10

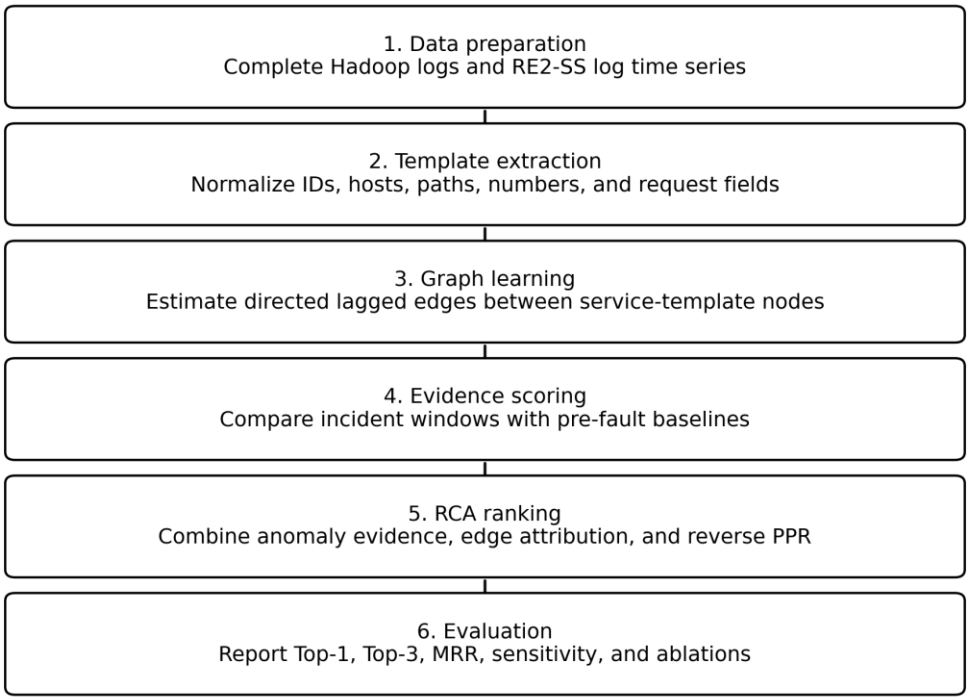

Figure 2. Research flowchart used in the empirical study.

Table 3. Examples of raw log messages and extracted event templates.

Raw log message	Extracted event template	Service
Created MRAppMaster for application appattempt_1445062781478_0011_000001	Created MRAppMaster for application <ID>	mapreduce
Address change detected. Old: msra-sa-41/10.190.173.170:9000 New: msra-sa-41:9000	Address change detected. Old: msra-sa-<NUM>/<IP>:<NUM> New: msra-sa-<NUM>:<NUM>	ipc
POST /orders 503 54.540 ms - -	front-end POST /orders <:HTTP5XX:> <:NUM>.<:NUM> ms - -	front-end

LogHub Hadoop is used because it contains complete raw logs and labeled abnormal Hadoop applications. The Hadoop labels identify normal jobs and three abnormal failure families: machine down, network disconnection, and disk full. Therefore, this dataset is used for parsing, service-template statistics, and graph-scale characterization. RE2-SS is used for root-cause evaluation because each fault case has a root service label and log time series suitable for service-level RCA [12]. The resume and raw of data are presented in Table 1 and Table 2.

2.1 Event template extraction

Each timestamped log record is mapped to a node v = (s, t), where s is the inferred service and t is the normalized event template. Classical parsers such as Drain and earlier message-type extraction methods motivate this template abstraction [18], [19]. Hadoop services are inferred from Java logger namespaces, for example mapreduce, yarn, hdfs, ipc, metrics, and http. In RE2-SS, the container and template identifiers are provided with the log time series. Identifiers, IP addresses, host names, paths, timestamps, hexadecimal tokens, and numeric fields are replaced by symbolic tokens. Continuation lines from multi-line stack traces are not treated as independent events because they do not contain timestamp and logger fields. The samples of data are presented in Table 3.

Table 4. Correlation graph statistics

Dataset	Nodes	Directed edges	Density	Avg. out-degree	Largest SCC	Lag	min_count
LogHub Hadoop	688	14030	0.030	20.390	420	1.0 s	5
RCAEval RE2-SS	861	6169	0.008	7.160	35	15 s	1

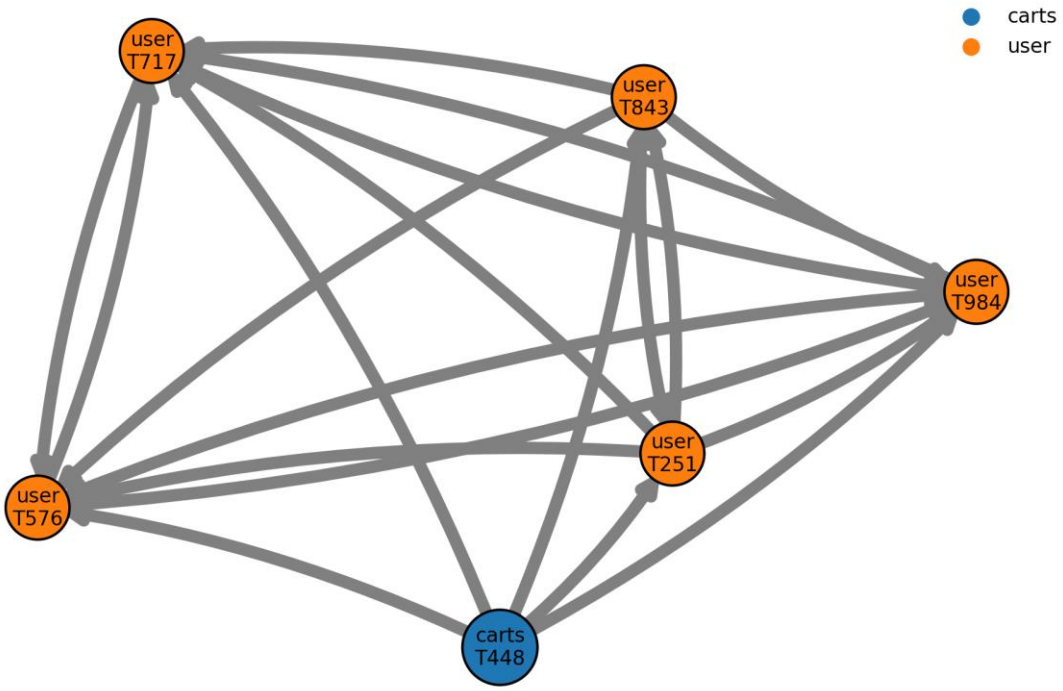

Figure 3. Local neighborhood of a service-template node in the RE2-SS correlation graph

2.2 Correlation graph construction

A directed graph G = (V, E) is constructed over service-template nodes. An edge u -> v means that v repeatedly appears after u within a bounded lag. For Hadoop, the lag is measured in milliseconds from timestamped records; for RE2-SS, the log time series is provided in 15-second bins, so the main lag is one bin. The conditional edge weight is:

$$w(u \to v) = c(u \to v) / \text{count}(u) \quad (1)$$

where c(u -> v) is the support count for the lagged ordered pair, and count(u) is the number of source occurrences of u. The main RE2-SS graph uses tau = 15 s, min_count = 1, and at most 30 outgoing neighbors per source node. Table 4 reports graph statistics for Hadoop and RE2-SS.

In Figure 3, each node represents one service-template pair. The first line in a node is the service name, and the second line is an anonymized template identifier. A directed arrow indicates that the target template appeared in the next 15-second bin after the source template in the training portion of the log time series. Edge thickness is proportional to the conditional weight in (1). The local graph is intentionally small so that predecessor and successor relationships can be inspected by an operator.

2.3 Anomaly evidence and RCA scoring

For each RE2-SS case, bins before the injection time form the baseline and bins after the injection time form the incident window. Let x_v be the post-injection count of node v and mu_v be the expected count estimated from the pre-injection bins. We use a Poisson-style positive deviation score:

$$z_v = (x_v - \mu_v) / \sqrt{\mu_v + 1} \quad (2)$$

The non-negative evidence is a_v = max(z_v, 0). Five log-only methods are compared. FreqDelta ranks nodes by direct anomaly evidence. ServiceOnly sums direct evidence by service. EdgeAttr sends evidence backward from anomalous nodes to their graph predecessors:

score_EA(p) = sum over u in A of a_u * w(p -> u) (3)

CorrGraphRCA runs personalized PageRank on the reverse graph, using normalized anomaly evidence as the teleport vector. With transition matrix P, restart parameter alpha, and teleport vector p0, the PageRank vector r satisfies:

r = alpha * P^T * r + (1 - alpha) * p0 (4)

HybridRCA combines direct evidence with graph evidence:

score_H(v) = lambda_z * norm(a_v) + lambda_EA * norm(score_EA(v)) + lambda_PPR * norm(score_PPR(v)) (5)

The main setting uses lambda_z = 0.8, lambda_EA = 0.1, and lambda_PPR = 0.1. This weighting reflects a practical log-only setting: direct log anomalies remain important, while graph evidence adjusts the rank when temporal predecessors support a candidate. Service-level rankings are obtained by taking the maximum node score within each service. Table 5 lists the experimental parameters.

2.4 Evaluation protocol

The primary task is service-level root-cause localization on RE2-SS. For every case, each method produces a complete ranking of candidate services. Top-1 accuracy measures whether the true root service is ranked first. Top-3 accuracy measures whether it appears in the first three services. MRR is the mean reciprocal rank. Runtime is measured per case for the online scoring step after template extraction and graph construction. A metric-only reference is also reported to show how much root-service information is available in RE2-SS metrics; it is not a log-only RCA method.

3. Results and Discussions

3.1 Main root-cause localization results

Table 6 reports service-level RCA results on 90 RE2-SS cases. HybridRCA obtains the best Top-1 and MRR among log-only methods, with Top-1 accuracy of 0.267 and MRR of 0.435. The improvement over FreqDelta is modest but consistent in Top-1 and MRR, indicating that lagged graph evidence can correct some loud-symptom rankings. ServiceOnly improves Top-3 but has lower Top-1, because aggregating all log evidence can favor busy services such as front-end, user, or orders even when they are not the injected root service.

Table 5. RE2-SS root-cause evaluation configuration

Parameter	Value
RE2-SS cases	90
Root services	5
Fault types	6
Time-bin size	15 s
Main graph lag	15 s
Top outgoing neighbors	30
Anomaly threshold for EdgeAttr	z >= 2.0
PageRank alpha	0.85
PageRank iterations	20
Hybrid weights	lambda_z=0.8, lambda_EA=0.1, lambda_PPR=0.1

Table 6. Root-service localization results on RCAEval RE2-SS

Method	Top1	Top3	MRR	Runtime_ms
FreqDelta	0.244	0.467	0.416	0.012
ServiceOnly	0.156	0.511	0.399	0.018
EdgeAttr	0.211	0.433	0.385	0.055
CorrGraphRCA	0.256	0.456	0.430	27.192
HybridRCA	0.267	0.456	0.435	27.232
MetricOnly reference	0.578	0.889	0.741	0.021

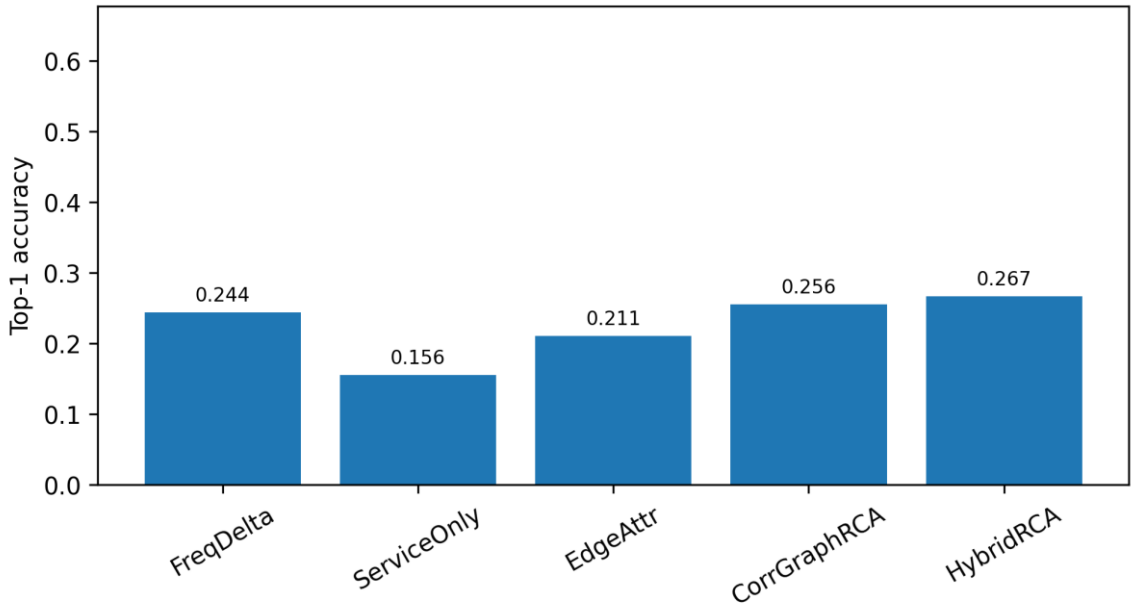

Figure 4. Top-1 root-service accuracy across log-only methods on RE2-SS

Table 7. Top-1 accuracy by fault type on RE2-SS

Fault type	Cases	FreqDelta Top1	ServiceOnly Top1	EdgeAttr Top1	CorrGraph RCA Top1	HybridRCA Top1
cpu	15	0.333	0.067	0.200	0.333	0.333
delay	15	0.200	0.200	0.200	0.200	0.200
disk	15	0.267	0.133	0.200	0.267	0.333
loss	15	0.333	0.267	0.333	0.333	0.333
mem	15	0.333	0.267	0.333	0.333	0.333
socket	15	0.000	0.000	0.000	0.067	0.067

Table 8. Top-1 accuracy by root service on RE2-SS.

Root service	Cases	FreqDelta Top1	ServiceOnly Top1	EdgeAttr Top1	CorrGraph RCA Top1	HybridRCA Top1
carts	18	0.556	0.389	0.556	0.667	0.667
catalogue	18	0.056	0.000	0.056	0.056	0.056
orders	18	0.333	0.333	0.389	0.333	0.333
payment	18	0.056	0.000	0.000	0.000	0.056
user	18	0.222	0.056	0.056	0.222	0.222

The metric-only reference reaches 0.578 Top-1 and 0.889 Top-3. This gap is important: many RE2-SS faults are CPU, memory, disk, delay, loss, or socket faults, and their strongest root evidence often appears in resource metrics rather than in application log counts, as presented in Figure 4. Therefore, the log-only graph should be interpreted as useful diagnostic evidence, not as a complete replacement for metrics and traces.

3.2 Breakdown by fault type and service

Table 7 shows that socket faults are the hardest for log-only methods, while CPU, disk, loss, and memory faults provide more usable log evidence. Table 8 shows that carts is the easiest root service because its injected faults produce distinctive service-level log deviations. Payment and catalogue are more difficult because their symptoms are often observed through front-end or user-facing request logs rather than as dominant root-service templates. This service-level imbalance explains why Top-3 is more stable than Top-1 across methods.

3.3 Sensitivity and ablation analysis

Table 9 indicates that the least aggressive support threshold gives the best Top-1 result for this dataset. Increasing min_count reduces the number of edges and slightly lowers MRR. The reason is that each RE2-SS case has only 49 pre-injection bins, so overly strong pruning removes rare but informative service-template transitions.

Table 10 and Figure 5 clarify which parts of the proposal help on real log-only RCA labels. Removing graph propagation gives the same result as direct anomaly evidence. Removing reverse PPR reduces the method to anomaly evidence with edge attribution and does not improve Top-1. Removing edge attribution but keeping reverse PPR preserves Top-1 but slightly lowers Top-3. Same-service-only edges do not hurt Top-1 on RE2-SS, which suggests that many root-service labels are best recovered from local log changes rather than cross-service log cascades. This does not invalidate cross-service reasoning; rather, it shows that cross-service log edges must be weighted carefully because user-facing services generate many secondary symptoms.

Table 9. Sensitivity to edge support threshold in the RE2-SS graph

Tau_s	MinCount	Edges	Nodes	Top1	Top3	MRR
15.000	1.000	6169.0	861.0	0.267	0.456	0.435
15.000	2.000	2757.0	861.0	0.256	0.467	0.424
15.000	4.000	1863.0	861.0	0.256	0.444	0.418
15.000	8.000	1710.0	861.0	0.256	0.444	0.419

Table 10. Ablation study across graph and score-design choices

Setting	Edges	Hybrid Top1	Hybrid Top3	Hybrid MRR
Full graph (tau=15s, mc=1)	6169	0.267	0.456	0.435
No graph propagation (anomaly only)	0	0.244	0.467	0.416
No edge attribution	6169	0.267	0.444	0.434
No reverse PPR	6169	0.244	0.467	0.416
No cross-service edges	2030	0.267	0.478	0.438
Stronger pruning (mc=4)	1863	0.256	0.444	0.418

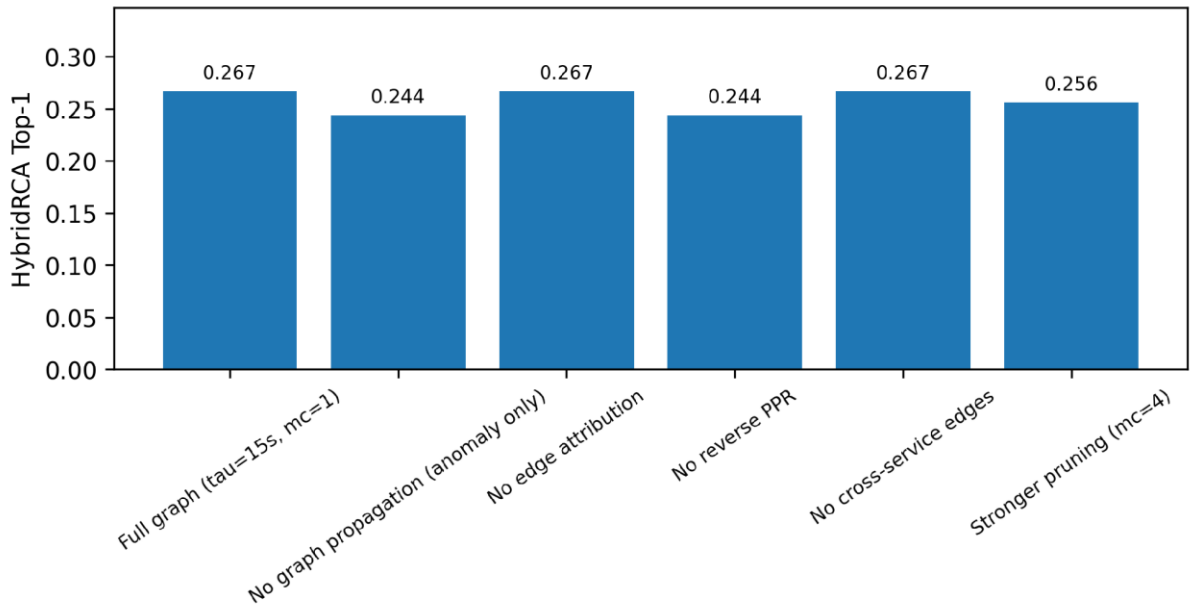

Figure 5. Ablation effect on HybridRCA Top-1 accuracy

Table 11. Sensitivity of HybridRCA to direct anomaly-evidence weight

lambda_z	lambda_EA	lambda_PPR	Top1	Top3	MRR
0.000	0.500	0.500	0.256	0.456	0.432
0.200	0.400	0.400	0.256	0.456	0.432
0.400	0.300	0.300	0.256	0.456	0.432
0.600	0.200	0.200	0.256	0.456	0.432
0.800	0.100	0.100	0.267	0.456	0.435
1.000	0.000	0.000	0.244	0.467	0.416

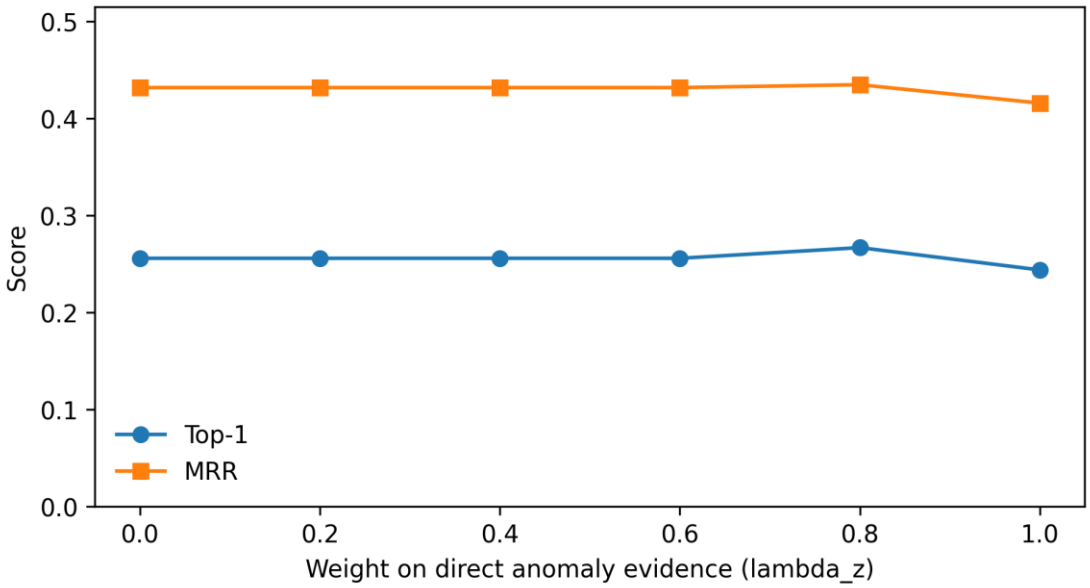

Figure 6. HybridRCA sensitivity to the direct anomaly-evidence weight

Table 12. Case-study incident: top-ranked templates under three methods

Method	Rank	Service	Template	Score	IsTrue Root
FreqDelta	1	user	user ts=<:NUM:>-<:NUM:>-20T03:<:NUM:>:<:NUM:>.32524Z caller=middlewares.go:<:NUM	0.257	
FreqDelta	2	carts	carts <:TIMESTAMP:> INFO [carts,<:ITEM:> <:NUM:> --- [p-nio-<:NUM:>-exec-<:NUM:>	0.257	YES
FreqDelta	3	user	user ts=<:TIMESTAMP:> caller=middlewares.go:<:NUM:> method=Register username=<:U	0.049	
FreqDelta	4	front-end	front-end { id: '<:ID:>' }	0.049	
FreqDelta	5	front-end	front-end Merging carts for customer id: <:ID:> and session id:<:SESSIONID:>	0.049	
EdgeAttr	1	user	user ts=<:NUM:>-<:NUM:>-20T03:<:NUM:>:<:NUM:>.32524Z caller=middlewares.go:<:NUM	0.013	
EdgeAttr	2	carts	carts <:TIMESTAMP:> INFO [carts,<:ITEM:> <:NUM:> --- [p-nio-<:NUM:>-exec-<:NUM:>	0.013	YES
EdgeAttr	3	user	user ts=<:TIMESTAMP:> caller=middlewares.go:<:NUM:> method=Register username=<:U	0.003	
EdgeAttr	4	front-end	front-end { id: '<:ID:>' }	0.003	
EdgeAttr	5	front-end	front-end Merging carts for customer id: <:ID:> and session id:<:SESSIONID:>	0.003	

HybridRCA	1	carts	carts <:TIMESTAMP:> INFO [carts,<:ITEM:> <:NUM:> --- [p-nio-<:NUM:>-exec-<:NUM:>	0.222	YES
HybridRCA	2	user	user ts=<:NUM:>-<:NUM:>-20T03:<:NUM:>:<:NUM:>.32524Z caller=middlewares.go:<:NUM	0.222	
HybridRCA	3	user	user ts=<:TIMESTAMP:> caller=middlewares.go:<:NUM:> method=Register username=<:U	0.043	
HybridRCA	4	front-end	front-end Posting Customer:<:CUSTOMER:>	0.043	
HybridRCA	5	front-end	front-end { id: '<:ID:>' }	0.043	

Table 11 and Figure 6 show that lambda_z = 0.8 yields the best Top-1 and MRR. When lambda_z = 1.0, the model becomes frequency-only. When lambda_z is too low, noisy graph propagation can over-rank secondary symptoms. The selected weighting therefore reflects the empirical balance between local evidence and graph-based explanation.

3.4 Case study

Table 12 illustrates a carts disk fault. FreqDelta ranks a user template slightly above a carts template because both become anomalous and the user-facing symptom is marginally louder. HybridRCA ranks the carts template first because the anomaly score remains strong while graph evidence adds support for the carts predecessor. This example shows the intended diagnostic behavior: graph evidence should not replace direct evidence, but it can break close ties in favor of a more plausible initiating service-template event.

3.5 Strengths, limitations, and implications

The main strength of the proposed approach is interpretability. Every candidate can be explained as a service-template node, with direct anomaly evidence and graph-based predecessor support. This is easier for operators to inspect than an opaque embedding score. The method is also lightweight: template extraction is linear in the number of log records, graph construction uses bounded lag windows, and online ranking runs in milliseconds to tens of milliseconds per case.

The empirical results also expose unresolved weaknesses. First, log-only evidence is insufficient for many resource faults. RE2-SS metrics provide a much stronger service-localization signal than log counts, especially for CPU, memory, disk, and network conditions. Second, cross-service edges can be noisy because front-end and user-facing services often log the visible symptom, not the root. Third, short RE2-SS pre-injection windows limit reliable edge estimation. These weaknesses explain why the observed improvement is modest and why practical RCA systems should combine logs with metrics and traces when those signals are available [14], [17], [18].

Compared with causal-inference approaches, the proposed graph should be interpreted as a correlation graph, not an intervention-level causal graph. The direction of an edge comes from temporal precedence, while causality in the formal sense would require assumptions or interventions beyond log co-occurrence [20], [21]. The practical value of the graph is that it supplies auditable hints for ranking and explanation. PageRank-style propagation provides a standard way to aggregate evidence over such directed graphs [22].

4. Conclusion

This paper presented a template-aware correlation-graph approach for root cause analysis from multi-service logs. The pipeline extracts service-template events, learns directed lagged correlations, computes anomaly evidence, and ranks root-cause candidates by combining direct evidence with one-step edge attribution and reverse personalized PageRank.

The empirical study used the complete LogHub Hadoop logs and RCAEval RE2-SS. Hadoop provided a larger raw-log setting for template extraction and graph characterization, while RE2-SS provided real root-service labels for RCA evaluation. On RE2-SS, HybridRCA achieved 0.267 Top-1, 0.456 Top-3, and 0.435

MRR, improving Top-1 and MRR over frequency-only log ranking. The metric-only reference achieved 0.578 Top-1, confirming that log-only RCA is useful but incomplete for resource-oriented microservice faults.

Future work will integrate the same template-aware graph with metrics and traces, learn delay-aware edge weights, and evaluate leave-one-incident-out graph learning to reduce transductive effects. The current results show that correlation graphs can improve the auditability and ranking of log-based RCA, while multimodal telemetry remains necessary for stronger root-cause localization in production microservices.

References

[1] J. Zhu, S. He, P. He, J. Liu, and M. R. Lyu, "Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics," *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, pp. 355–366, 2023, doi: 10.1109/ISSRE59848.2023.00071.

[2] J. Zhu, S. He, P. He, J. Liu, and M. R. Lyu, "Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics," *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, pp. 355–366, 2020, doi: 10.1109/ISSRE59848.2023.00071.

[3] J. Zhu *et al.*, "Tools and Benchmarks for Automated Log Parsing," *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019*, pp. 121–130, 2019, doi: 10.1109/ICSE-SEIP.2019.00021.

[4] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 1285–1298, 2017, doi: 10.1145/3133956.3134015.

[5] W. Meng *et al.*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2019-August, pp. 4739–4745, 2019, doi: 10.24963/IJCAI.2019/658.

[6] H. Guo, S. Yuan, and X. Wu, "LogBERT: Log Anomaly Detection via BERT," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2021-July, 2021, doi: 10.1109/IJCNN52387.2021.9534113.

[7] Z. Jiang *et al.*, "A Large-Scale Evaluation for Log Parsing Techniques: How Far Are We?," *ISSTA 2024 - Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 223–234, 2024, doi: 10.1145/3650212.3652123.

[8] N. Zhao *et al.*, "An empirical investigation of practical log anomaly detection for online service systems," *ESEC/FSE 2021 - Proceedings of the 29th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1404–1415, 2021, doi: 10.1145/3468264.3473933.

[9] L. Wu, J. Tordsson, E. Elmroth, and O. Kao, "MicroRCA: Root Cause Localization of Performance Issues in Microservices," *Proceedings of IEEE/IFIP Network Operations and Management Symposium 2020: Management in the Age of Softwarization and Artificial Intelligence, NOMS 2020*, 2020, doi: 10.1109/NOMS47738.2020.9110353.

[10] Z. Yao *et al.*, "Chain-of-Event: Interpretable Root Cause Analysis for Microservices through Automatically Learning Weighted Event Causal Graph," *FSE Companion - Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, pp. 50–61, 2024, doi: 10.1145/3663529.3663827.

[11] L. Pham, H. Ha, and H. Zhang, "Root Cause Analysis for Microservice System based on Causal Inference: How Far Are We?," *Proceedings - 2024 39th ACM/IEEE International Conference on Automated Software Engineering, ASE 2024*, pp. 706–718, 2024, doi: 10.1145/3691620.3695065.

[12] L. Pham, H. Zhang, H. Ha, F. Salim, and X. Zhang, "RCAEval: A Benchmark for Root Cause Analysis of Microservice Systems with Telemetry Data," *WWW Companion 2025 - Companion Proceedings of the ACM Web Conference 2025*, pp. 777–780, 2025, doi: 10.1145/3701716.3715290.

[13] Y. Wang, Z. Zhu, Q. Fu, Y. Ma, and P. He, "MRCA: Metric-level Root Cause Analysis for Microservices via Multi-Modal Data," in *2024 39th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2024, pp. 1057–1068, doi: 10.1145/3691620.369548.

[14] Z. Zhu, C. Lee, X. Tang, and P. He, "HeMiRCA: Fine-Grained Root Cause Analysis for Microservices with Heterogeneous Data Sources," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 8, pp. 1–25, 2024, doi: https://doi.org/10.1145/3674726.

[15] PhamLuan, HaHuong, and ZhangHongyu, "BARO: Robust Root Cause Analysis for Microservices via Multivariate Bayesian Online Change Point Detection," *Proceedings of the ACM on Software*

Engineering, vol. 1, no. FSE, pp. 2214–2237, 2024, doi: 10.1145/3660805.
[16] Z. Zhao *et al.*, "CHASE: A Causal Hypergraph based Framework for Root Cause Analysis in Multimodal Microservice Systems," *IEEE Transactions on Big Data*, pp. 1–14, 2025, doi: 10.1109/TBDATA.2026.3725931.
[17] N. Fu, G. Cheng, Y. Teng, G. Dai, S. Yu, and Z. Chen, "Intelligent Root Cause Localization in MicroService Systems: A Survey and New Perspectives," *ACM Computing Surveys*, vol. 57, no. 12, 2025, doi: 10.1145/3736755.
[18] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An Online Log Parsing Approach with Fixed Depth Tree," *Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017*, pp. 33–40, 2017, doi: 10.1109/ICWS.2017.13.
[19] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "A lightweight algorithm for message type extraction in system application logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 11, pp. 1921–1936, 2012, doi: 10.1109/TKDE.2011.138.
[20] J. Pearl, *Causality: Models, Reasoning and Inference*. New York: Cambridge University Press, 2009.
[21] Granger and C. W. J, "Investigating Causal Relations by Econometric Models and Cross-Spectral Methods," *Econometrica*, vol. 37, no. 3, pp. 424–438, 1969.
[22] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank Citation Ranking : Bringing Order to the Web*. 1999.