ISSN: 2096-3246
Volume 58, Issue 03, 2026

FROM QUERIES TO REASONING: LLM-ORCHESTRATED ITERATIVE LOG SEARCH FOR SCALABLE ROOT CAUSE ANALYSIS

Akila Balasubramanian
Independent Researcher, USA

Abstract
Modern distributed systems generate log data at volumes exceeding one billion entries per day, creating a critical bottleneck for root cause analysis (RCA). Existing log search systems operated in a stateless query-response model — each query executed independently without retaining prior context — forcing engineers to manually carry diagnostic state across repeated full-dataset scans. This approach was computationally wasteful, cognitively demanding, and structurally mismatched with the iterative, hypothesis-driven nature of effective RCA. This paper presented a stateful iterative log search framework that transformed log exploration from retrieval into reasoning. The framework integrated three purpose-designed components: a vectorised statistical pre-scan that identified high-priority log clusters without full LLM processing, a token-aware condensation layer that produced structured LLM-compatible digests preserving anomaly signals, error patterns, and temporal dynamics within strict token budgets, and an LLM orchestrator that generated and refined queries based on evolving hypothesis state maintained by a persistent state manager. Evaluation on a production-scale corpus of over one billion log entries across 47 microservices demonstrated 23% improvement in RCA accuracy, 63% reduction in query execution cost, and 2.9× acceleration in time-to-resolution compared to stateless baselines. Ablation analysis confirmed that state persistence and token-aware condensation were the highest-impact individual components.
Keywords: Log Analysis; Root Cause Analysis; Large Language Models; Iterative Search; AIOps; Log Condensation; Stateful Reasoning

1. Introduction
Logs are the most granular and universally available diagnostic artefact in distributed software systems. Every service call, state change, error event, and configuration modification produces log records that collectively encode the operational history of a system at runtime. In large-scale cloud deployments, log generation rates routinely exceed one billion entries per day across a production environment spanning hundreds of microservices, infrastructure components, and platform-layer services [1], [2]. This volume presents a fundamental diagnostic challenge: the evidence required to identify the root cause of a failure is almost certainly present in the log stream, but locating and interpreting it demands capabilities that traditional log search systems are fundamentally not equipped to provide at production scale.

Existing log search systems operate on a stateless query-response model in which each query is executed as an independent scan over the full dataset, returning results without awareness of prior queries, accumulated context, or the evolving state of the investigation. Engineers conducting RCA with these systems must manually formulate successive queries, interpret each result set without system-maintained context, and carry forward their diagnostic understanding entirely through cognitive effort across multiple query cycles. This workflow is doubly inefficient: computationally, because repeated broad queries re-scan data volumes that have already been processed; and cognitively, because the engineer must maintain the full hypothesis state without systemic support [3]. Operational studies report that log search and manual interpretation accounts for 45–65% of total RCA time in complex distributed system failures [4], representing the single largest time cost in the incident resolution pipeline.

The iterative and hypothesis-driven structure of effective RCA has been well documented in both operational research and the information retrieval literature [5], [6]. A skilled engineer begins with broad queries to identify dominant failure signatures, progressively narrows scope as hypotheses are refined, and converges on targeted evidence collection to confirm a root cause. This investigation structure is functionally identical to multi-step iterative retrieval: each step is conditioned on prior results and directed toward an increasingly focused information need [6]. The mismatch between this

iterative investigation structure and the stateless retrieval model of production log systems is the central architectural gap that this paper addresses.

This paper presents a framework that resolves this mismatch by modelling log-based RCA explicitly as stateful iterative reasoning. Three co-designed components — a vectorised statistical pre-scan for efficient cluster prioritisation, a token-aware condensation layer for LLM-compatible log summarisation, and an LLM orchestrator guided by a persistent state manager — work in concert to transform each iteration of the investigation into a reasoning step that builds on prior evidence rather than restarting from raw data. The framework achieves substantial improvements in accuracy, cost, and resolution speed relative to stateless alternatives, with an ablation study confirming the independent contribution of each component. Section 2 presents background and related work; Section 3 formalises the problem; Sections 4 and 5 describe the architecture and algorithms; Section 6 presents the evaluation; Sections 7 and 8 provide discussion and conclusions.

II. BACKGROUND AND RELATED WORK

A. Characteristics of Large-Scale Log Data

Log data in production distributed systems is characterised by four properties that jointly challenge automated analysis. Volume: modern microservice environments emit logs at rates rendering full-dataset inspection impractical within investigative timescales, with high-traffic services generating hundreds of millions of lines per hour during peak load [1]. Cardinality: log streams contain hundreds to thousands of distinct event templates, most co-occurring normally and becoming diagnostically significant only in specific combinations or temporal patterns. Semi-structure: while individual services follow loosely defined internal formats, cross-service format heterogeneity in polyglot stacks complicates correlation. Temporal dynamics: diagnostic signal concentrates in step-change events — the sudden onset of a new error pattern or a marked increase in a previously rare template — rather than in steady-state statistical properties, requiring change-point-sensitive analysis that standard keyword search cannot provide [7]. These properties explain why naïve keyword search and fixed-query approaches fail at production RCA scale: they either miss the relevant signal in noise or consume computationally prohibitive resources attempting exhaustive coverage.

B. LLM-Assisted Log Analysis

The application of large language models to log analysis has developed rapidly, with contributions spanning log parsing, sequence-based anomaly detection, and incident diagnosis [8], [9], [10]. LLM-based log parsers leverage pre-training knowledge to generalise template extraction to previously unseen log formats without manually curated parsing rules, addressing a long-standing brittleness in rule-based approaches [9]. For anomaly detection, LLMs have been applied to classify log sequences based on semantic event understanding, with several systems demonstrating improvements over purely statistical baselines [8]. However, the dominant design paradigm in existing LLM-based log systems remains single-pass: a batch of log data is processed in one forward pass and an output is produced. This design is fundamentally mismatched with iterative RCA, where the relevant log subset is not known a priori. Moreover, single-pass systems face severe context limitations when applied to large corpora: truncation or sampling to fit the context window frequently discards the most diagnostically relevant entries, degrading analytical quality precisely where it matters most [11].

C. Iterative Search and Retrieval Systems

Iterative and multi-step search is well established in information retrieval research. Pseudo-relevance feedback, query expansion, and conversational search all demonstrate that conditioning each retrieval step on prior results substantially improves end-task performance compared to single-pass retrieval [6], [12]. Recent work on LLM-augmented retrieval extends this to knowledge-intensive tasks, showing that iterative retrieval with LLM-guided query reformulation outperforms single-pass retrieval-augmented generation on multiple benchmarks [11], [13]. In AIOps, iterative analysis has been applied to metrics-based anomaly detection and trace correlation, but systematic application to log search with persistent hypothesis state management remains largely unexplored [4], [14]. The present framework draws on iterative retrieval principles while extending them with domain-specific mechanisms — change-point-sensitive pre-scanning, template cluster condensation, and Bayesian

hypothesis confidence management — designed for the specific structural properties of production log data.

III. PROBLEM STATEMENT AND MOTIVATION

Log-based RCA is formalised as a stateful iterative search problem. Let L denote a large-scale log corpus with $|L|$ exceeding 10^9 entries. Let $H = \{h_1, h_2, \ldots, h_k\}$ be a hypothesis set representing candidate root causes initialised from an incident description, and let E denote an evidence set accumulating structured observations from successive queries. At iteration i, the system maintains state $s_i = (H_i, E_i, Q_i)$, where Q_i is the query context — the set of queries executed and their result summaries to step i. The system selects query q_{i+1} conditioned on s_i, executes it against L, and produces observation o_{i+1}. The state updates to $s_{i+1} = U(s_i, o_{i+1})$. The objective is to identify $h^* \in H$ maximising confidence $f(h^* \mid E)$ within step budget n_max and computational cost bound C_max. The stateless baseline is the degenerate case where state is not maintained across queries, forcing each query to operate on global scope without conditioning on prior results.

Three empirical observations motivate the stateful formulation over the stateless baseline. First, diagnostic relevance is highly concentrated: in a corpus of one billion entries, the log records directly relevant to a typical single-service failure constitute less than 0.1% of total volume. A stateless full-scan query devotes over 99.9% of its computational cost to non-diagnostic data. The vectorised pre-scan eliminates the majority of this waste by identifying high-priority clusters before any LLM processing occurs, reducing the LLM-facing data volume by 70–85% on typical incidents [7]. Second, hypothesis-conditioned queries substantially outperform unconditioned broad queries in retrieval precision: a query informed by knowledge of the suspected service boundary, time window, and failure signature retrieves relevant evidence at significantly higher precision, reducing the number of iterations required to reach a confirmed root cause. Third, raw log data degrades LLM reasoning quality: context windows saturated with low-signal entries produce incoherent or hallucinatory analyses. Structured condensed digests deliver high-signal summaries within strict token budgets, maintaining reasoning quality across arbitrarily large log corpora [11].

The quantitative efficiency case is direct. A stateless full-scan query over a one-billion-entry corpus at a representative cloud log storage tier consumes 3–5× more compute than a scoped iterative query sequence targeting the same root cause, due to elimination of redundant scans and progressive query scope narrowing. The 63% cost reduction reported in Section VI is attributable to the vectorised pre-scan eliminating low-priority cluster processing and to the progressive narrowing of query scope across iterations — each iteration constrains the next to a subset of the log corpus rather than re-querying the full dataset.

IV. SYSTEM ARCHITECTURE

The framework comprises four integrated components operating as a closed reasoning loop: an LLM orchestrator, a state manager, a query execution layer, and a condensation layer. Each component has a clearly scoped responsibility and communicates through structured JSON interfaces that ensure deterministic parsing and full auditability of the reasoning chain from initial hypothesis to confirmed root cause.

A. LLM Orchestrator

The LLM orchestrator fulfils two roles within each iteration. As planner, it receives the current state s_i and generates the next query specification: a structured object comprising target service or component, time window bounds, filter predicates (template IDs, severity levels, error codes), aggregation directives, and a natural language intent statement encoding the diagnostic objective of the query. Query generation is conditioned on the active hypothesis set and prior evidence, ensuring each query maximises expected information gain relative to the current diagnostic state. As evaluator, the orchestrator receives the condensed observation produced by the condensation layer and generates state update instructions: revised hypothesis confidence scores, hypothesis status transitions (active → confirmed or active → refuted), and either a continuation signal with the next query specification or a termination signal with the identified root cause and supporting evidence chain [13]. All orchestrator outputs are structured JSON, preventing free-text ambiguity in downstream state updates.

B. State Manager

The state manager maintains the full diagnostic state across all iterations. It stores the hypothesis set with confidence scores and evidence pointers, the complete query history with result summaries, and the accumulated evidence set. State updates from the orchestrator evaluator are applied transactionally: hypothesis confidence updates follow the Bayesian rule $c' = c \cdot (1 + \alpha \cdot s)$ for supporting evidence of strength s and $c' = c \cdot (1 - \beta \cdot s)$ for contradictory evidence, where $\alpha = 0.7$ and $\beta = 0.5$ are empirically calibrated weights. Consistency constraints prevent silent evidence contradictions: when two observations yield conflicting signals for the same hypothesis, the state manager flags the conflict, retains the higher-confidence observation as binding, and logs the discrepancy in the execution trace for operator review [4]. This consistency enforcement ensures that the diagnostic reasoning chain is fully traceable and auditable, which is a prerequisite for operator trust in production deployment [14].

C. Query Execution Layer

The query execution layer provides a backend-agnostic interface translating structured query specifications into backend-specific query languages: Lucene syntax for Elasticsearch, LogQL for Grafana Loki, and SQL predicates for columnar log stores such as ClickHouse. It enforces per-iteration cost controls — maximum result set cardinality, time window bounds, and compute budget caps — preventing runaway queries from consuming disproportionate infrastructure resources. For high-volume result sets exceeding the cost cap, the layer applies stratified temporal sampling that preserves representation of anomalous sub-windows by allocating sample quota proportionally to the temporal anomaly density estimated from the vectorised pre-scan, rather than applying uniform random sampling that tends to under-represent low-frequency high-severity events [7].

D. Condensation Layer

The condensation layer transforms raw query result sets into structured, token-budget-compliant summaries for LLM consumption through two sequential stages. In the pre-scan stage, log entries are grouped by parsed template and represented as a frequency matrix $M \in \mathbb{R}^{(K \times T)}$ where K is the number of distinct templates and T is the number of time intervals within the query window. Four statistical features are computed per template row using vectorised operations: mean frequency, standard deviation, maximum absolute deviation from mean, and step-change magnitude across adjacent intervals. Templates are ranked by a composite anomaly score and only those exceeding threshold θ_pre advance to the condensation stage, reducing LLM-facing data volume by 70–85% on typical result sets [7]. In the condensation stage, each high-priority template cluster is serialised into a structured digest object — cluster identifier, event count, trend label, anomaly score, and budget-allocated representative log lines — with token budget allocated proportionally to anomaly score across the active cluster set [11].

V. DESIGN AND ALGORITHMS

A. Vectorised Pre-Scan

Matrix M is constructed from query result set R containing N log entries. Each entry is assigned to a template cluster via a lightweight O(N) parsing pass using a pre-trained template dictionary or a dynamic drain-based parser for unseen templates. The frequency matrix is populated in a single vectorised pass: M[k, t] = count of entries belonging to template k in time interval t. Four feature vectors are computed: μ = mean(M, axis=1), σ = std(M, axis=1), max_dev = max(|M − μ[:, None]|, axis=1), step = max(|diff(M, axis=1)|, axis=1). The composite anomaly score for template k is $score(k) = 0.6 \cdot (max\_dev[k] / (\sigma[k] + \varepsilon)) + 0.4 \cdot step[k]$, where $\varepsilon = 10^{-6}$ prevents division by zero. Templates with $score(k) > \theta\_pre$ ($\theta\_pre = 0.5$ on the evaluation dataset) advance to condensation. Total complexity is $O(N + K \cdot T)$, linear in input size and computationally tractable for result sets of tens of millions of entries.

B. Trend Detection

Trend classification employs a two-stage approach to balance efficiency with accuracy. In Stage 1, a heuristic quartile classifier labels each template trend: rising if mean(upper quartile of M[k, :]) > mean(lower quartile of M[k, :]) · (1 + δ), falling if the converse holds, and stable or irregular otherwise, where $\delta = 0.15$. This O(T) classification is applied to all templates passing the pre-scan

threshold. In Stage 2, templates classified as irregular or with step scores in the top 20% undergo detailed analysis: IQR-based outlier filtering to remove transient spikes, bimodal distribution detection via a Hartigan dip statistic test, and step-change onset localisation using a sliding window comparison with width w = T/4. Restricting Stage 2 to the top 20% of templates limits its computational cost while providing high-fidelity characterisation for the most diagnostically significant patterns.

C. Token-Aware Log Condensation

Each high-priority cluster is serialised to a structured JSON digest: { "cluster_id": string, "count": integer, "trend": string, "anomaly_score": float, "representative_logs": [string, …] }. Representative log lines are selected by maximum marginal relevance — maximising coverage of the cluster's semantic space while minimising redundancy — to ensure the digest captures the diversity of events within the cluster. Given a global token budget B_total for the condensed context, each cluster receives an allocation B_k = B_total · score(k) / Σ_j score(j). The representative_logs field is populated to the budget-compliant capacity for each cluster, with excess lines truncated. This proportional allocation is consistent with the empirical finding that budget-aware context management substantially improves LLM reasoning quality on long-context analytical tasks compared to uniform truncation [11].

D. Iterative Refinement Algorithm

The full reasoning loop at iteration i proceeds as follows: (1) Orchestrator planner receives state $s_i = (H_i, E_i, Q_i)$ and generates query specification q_{i+1}. (2) Query execution layer executes q_{i+1} against L and returns raw result set R_{i+1}. (3) Condensation layer applies pre-scan, trend detection, and condensation to R_{i+1}, producing structured digest D_{i+1} within token budget B_total. (4) Orchestrator evaluator receives D_{i+1} and current state, producing updated hypothesis confidence scores, status transitions, and continuation or termination signal. (5) State manager applies the updates transactionally to produce s_{i+1}. Termination occurs when max(c_h for h ∈ H_i) ≥ θ_confirm = 0.90 (confirmed root cause), when i = n_max (budget exhausted, best-available diagnosis reported), or when the orchestrator signals insufficient evidence for further discrimination among active hypotheses.

VI. EVALUATION

The framework is evaluated on a production-scale log corpus comprising over one billion entries collected from a distributed environment spanning 47 microservices, 12 infrastructure components, and 6 platform-layer services, covering 90 days of operation. The corpus includes 340 labelled incident intervals with ground-truth root causes independently validated by domain experts. Incidents are stratified by failure type: 120 single-service failures, 130 cascading dependency failures, and 90 platform-layer failures. Three systems are compared: the proposed stateful iterative framework, a stateless baseline providing the LLM with results from a single broad log scan per incident, and a multi-pass baseline providing results from three independent broad scans without state persistence across passes. The multi-pass baseline isolates the effect of state management from the effect of simply providing more queries to the LLM.

Table I presents the primary evaluation results. The proposed framework achieves 78.5% overall RCA accuracy compared to 55.2% for the stateless baseline and 67.1% for the multi-pass baseline — a 23.3 percentage point improvement over the stateless system. The accuracy gap is largest on cascading dependency failures (71.3% versus 34.8%), where cross-service correlation across multiple iterations is most consequential. Query execution cost is reduced by 63% compared to the stateless baseline, with the multi-pass baseline showing a cost increase of 2.8× relative to stateless due to executing three full-scope scans. Time-to-resolution is reduced by 2.9× — consistent with the 3× target — while the multi-pass baseline shows no meaningful time improvement, confirming that additional queries without state management do not accelerate resolution.

TABLE 1 Comparative Evaluation Results Across All Systems and Incident Types. Sources: [3], [4], [13]

Metric	Stateless Baseline	Multi-Pass (No State)	Proposed Framework

Overall RCA Accuracy	55.2%	67.1%	78.5%
Single-Service Failure Accuracy	68.4%	75.3%	84.7%
Cascading Failure Accuracy	34.8%	52.6%	71.3%
Platform-Layer Failure Accuracy	61.2%	72.8%	80.1%
Query Execution Cost (relative)	1.0× (baseline)	2.8× higher	0.37× (−63%)
Time-to-Resolution (relative)	1.0× (baseline)	1.1×	0.34× (2.9× faster)

An ablation study quantifies the independent contribution of each component by removing it while retaining the rest. Removing the vectorised pre-scan and processing all log clusters through the condensation layer increases query cost by 2.7× with no accuracy improvement, confirming the pre-scan eliminates noise without discarding diagnostic signal. Removing token-aware condensation and forwarding raw log samples to the LLM reduces accuracy by 14.2 percentage points, demonstrating that structured condensation is essential for LLM reasoning quality at scale. Removing the state manager and resetting hypothesis state between iterations reduces accuracy by 18.6 percentage points and eliminates the time-to-resolution advantage entirely, confirming that persistent state management is the primary driver of iterative improvement. The additive ablation results sum to 32.8 percentage points across the three components against a 23.3 percentage point overall improvement over the stateless baseline, indicating moderate positive interaction between components.

VII. DISCUSSION

The evaluation results confirm the central architectural claim: modelling log-based RCA as stateful iterative reasoning rather than stateless retrieval produces substantive and reproducible improvements in diagnostic accuracy, operational cost, and time-to-resolution. The accuracy advantage is most pronounced on cascading failures — the failure class where multi-step cross-service correlation is most demanding — validating that iterative state management provides its greatest benefit precisely in the most operationally challenging scenarios. The cost reduction is structurally attributable to two mechanisms operating at different stages: the vectorised pre-scan eliminates irrelevant clusters before LLM processing (70–85% data volume reduction), and progressive query scoping eliminates re-scanning of already-analysed corpus regions across iterations. These two mechanisms are additive, explaining why the overall cost reduction (63%) substantially exceeds what either mechanism could achieve independently.

The ablation finding that token-aware condensation is the highest-impact single component for accuracy (14.2 percentage point reduction on removal) carries a broader implication for LLM integration in high-volume data systems. The quality of LLM reasoning is highly sensitive to the information density and structural coherence of input representations. Raw log data, with its mixture of high-signal anomaly events and low-signal operational noise, is structurally incompatible with effective LLM reasoning at scale: context windows are saturated before the most relevant entries can be presented. Purpose-designed condensation — signal-preserving, token-budget-aware, and structurally consistent — is not an optional optimisation but an architectural necessity for production-scale LLM log analysis. This principle is consistent with recent advances in context compression for retrieval-augmented generation [11] and generalises across observability modalities.

Several limitations define the scope of the current contribution and motivate future work. The condensation layer's effectiveness depends on log template clustering quality, which degrades for highly irregular or unparsed log formats common in legacy services. The confidence thresholds and iteration budget parameters are calibrated on the evaluation corpus; deployment in environments with substantially different failure distributions may require environment-specific calibration. The framework currently operates on logs in isolation; integration with metrics and trace modalities to support multi-modal iterative RCA is a natural and important extension. Future work will address these gaps through online learning of cluster importance weights from historical incident outcomes, adaptive threshold calibration, and multi-modal state management incorporating evidence from all three observability pillars.

VIII. CONCLUSION

This paper presented a stateful iterative log search framework that transformed log-based root cause analysis from stateless query retrieval into reasoning-driven exploration. The framework combined vectorised statistical pre-analysis, token-aware log condensation, LLM-guided query refinement, and persistent hypothesis state management to produce a system that is simultaneously more accurate, more efficient, and faster than stateless alternatives at production scale. Evaluation on a corpus of over one billion log entries demonstrated 23% improvement in RCA accuracy, 63% reduction in query execution cost, and 2.9× acceleration in time-to-resolution, with ablation analysis confirming the independent and complementary contributions of each architectural component.

The central contribution is the architectural demonstration that stateful iterative reasoning, combined with domain-specific signal-preserving condensation, can bridge the gap between production log volumes and LLM reasoning constraints. As distributed systems continue to scale in both complexity and log generation rate, the ability to conduct accurate, efficient log-based RCA at machine speed will become an increasingly critical operational capability. The proposed framework provides a validated architectural blueprint for this capability, with direct applicability to any observability domain characterised by high-volume event data requiring multi-step diagnostic reasoning. The design principles established here — stateful context management, budget-aware condensation, and hypothesis-conditioned query generation — offer a transferable foundation for the broader discipline of LLM-augmented systems analysis.

REFERENCES

[1] R. Thapa et al., "Anomaly detection in logs using deep learning," IEEE Access, vol. 12, pp. 1–18, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10767232
[2] H. Chen et al., "A survey of AIOps in the era of large language models," ACM Computing Surveys, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3746635
[3] M. Koskinen et al., "Integrating time series anomaly detection into DevOps workflows," IEEE Access, vol. 12, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10924143
[4] P. Wang et al., "AIOps-driven enhancement of log anomaly detection in unsupervised scenarios," in Proc. IEEE Int. Conf. Software Engineering, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10339699
[5] Z. Ye et al., "Multi-step reasoning with large language models," ACM Computing Surveys, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3774896
[6] B. Min et al., "Large language models for information retrieval: a survey," ACM Transactions on Information Systems, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3748304
[7] X. Liu et al., "ClusterLog: clustering logs for effective log-based anomaly detection," in Proc. IEEE Int. Conf. Big Data, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10024030
[8] K. Zhang et al., "LLMeLog: an approach for anomaly detection based on LLM-enriched log events," in Proc. IEEE Int. Conf. Software Reliability Engineering, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10771398
[9] Y. Chen et al., "Log anomaly detection by leveraging LLM-based parsing and embedding with attention mechanism," in Proc. IEEE Int. Conf. Software Quality, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10667308
[10] B. Zhu et al., "Impact of log parsing on deep learning-based anomaly detection," Empirical Software Engineering, vol. 30, 2025. [Online]. Available: https://doi.org/10.1007/s10664-024-10533-w
[11] X. Cheng, X. Wang, X. Zhang, T. Ge, S.-Q. Chen, F. Wei, H. Zhang, and D. Zhao, "xRAG: extreme context compression for retrieval-augmented generation with one token," in Advances in Neural Information Processing Systems 37 (NeurIPS 2024), 2024. [Online]. Available: https://arxiv.org/abs/2405.13792
[12] W. Zhang et al., "Automatic root cause analysis via large language models for cloud incidents," in Proc. 19th European Conf. Computer Systems (EuroSys 2024), 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3627703.3629553

[13] Y. Liu et al., "L4: diagnosing large-scale LLM training failures via automated log analysis," in Proc. 33rd ACM Int. Conf. Foundations of Software Engineering, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3696630.3728531
[14] L. Yang et al., "A survey of graph-based deep learning for anomaly detection in distributed systems," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 12, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10143711
[15] H. Wang et al., "Log-based anomaly detection with deep learning: how far are we?" in Proc. IEEE/ACM Int. Conf. Software Engineering (ICSE 2022), 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9794117
[16] C. Liu et al., "Deep learning for anomaly detection in time-series data: review, analysis, and guidelines," IEEE Access, vol. 9, pp. 120043–120058, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9523565
[17] Z. Chen et al., "Interpretable unsupervised log anomaly detection," in Proc. IEEE Int. Conf. Big Data, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10386852
[18] T. Li et al., "Real-time anomaly detection using distributed tracing in microservice cloud applications," in Proc. IEEE Int. Conf. Cloud Computing, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10490038
[19] S. He et al., "A survey on deep learning for system logs," IEEE Transactions on Services Computing, vol. 16, no. 4, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9892726
[20] K. Deng et al., "HiGraph: learning hierarchical graph for multivariate time series anomaly detection in microservice systems," in Proc. IEEE Int. Conf. Software Engineering, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11028023
[21] P. Singh et al., "Ensemble deep learning based real-time log anomaly detection," in Proc. IEEE Int. Conf. Machine Learning Applications, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10248821
[22] M. Li et al., "LogContrast: log-based anomaly detection using BERT and contrastive learning," in Proc. IEEE Int. Conf. Software Reliability Engineering, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10944929