From Logs to Knowledge: LLM-Powered Dynamic Knowledge Graphs for Real-Time Cloud Observability

Nurmyrat Amanmadov[1], Tarlan Abdullayev[2]
University of Washington, Seattle, USA[1]
Northeastern University, Boston, USA[2]

***Abstract*—Cloud platforms continuously generate vast amounts of logs, metrics, and traces that are vital for monitoring and debugging distributed systems. However, current observability solutions are often siloed, dashboard-centric, and limited to surface-level correlations, making it difficult to derive actionable insights in real time. In this work, we present Log2Graph, a novel framework that leverages large language models (LLMs) to transform heterogeneous telemetry into dynamic knowledge graphs that evolve alongside system state. Unlike traditional log analytics, Log2Graph unifies unstructured messages, distributed traces, and configuration data into a living graph representation, enabling real-time dependency mapping, causal chain analysis, and compliance monitoring. Furthermore, the framework supports natural language queries over the evolving graph, allowing operators to ask questions such as "what services will be impacted if this database fails?" and receive precise, graph-backed explanations. Our evaluation on multi-cloud testbeds shows that Log2Graph reduces incident resolution time, improves accuracy in dependency detection, and enhances operator productivity. This work introduces a new paradigm of LLM-augmented observability, bridging the gap between raw logs and actionable cloud intelligence.**

***Keywords*—Large Language Models (LLMs); AI for cloud computing; knowledge graphs; logs**

I. Introduction

Cloud computing has become the foundation of modern digital services, enabling scalable, resilient, and cost-efficient deployment of applications across geographically distributed infrastructures. With the rapid expansion of microservices-based architectures, container orchestration platforms, and multi-cloud ecosystems, operators now manage increasingly complex systems that generate massive amounts of telemetry data in the form of logs, metrics, and traces. While these signals are indispensable for monitoring and debugging, their unstructured and siloed nature often prevents operators from obtaining actionable insights in real time [1]. Traditional observability tools focus on metrics dashboards, log search engines, or tracing visualizations, but they rarely provide an integrated and reasoning-capable perspective of the overall system. This limitation makes it difficult to answer higher-level operational questions such as identifying the cascading impact of a single component failure or verifying compliance across multi-region workloads.

Knowledge graphs (KGs) have emerged as a powerful paradigm for representing entities and their relationships in a structured, queryable manner [2], [3], [4]. By encoding dependencies as graph structures, KGs allow users to traverse relationships, discover hidden patterns, and support logical inference. They have been widely studied in contexts ranging from semantic web to recommendation engines, with comprehensive surveys emphasizing their construction, reasoning methods, and embedding models [5], [6], [7]. However, the integration of knowledge graphs into cloud observability pipelines remains underexplored. Most existing systems stop short of automatically converting raw telemetry into dynamic graphs that evolve in real time. Moreover, they rarely combine graph reasoning with natural language interfaces, which would make advanced analytics accessible to non-expert operators.

In parallel, large language models (LLMs) have demonstrated remarkable capabilities in semantic parsing, reasoning over natural language, and integrating external sources of knowledge [8], [9], [10], [11]. Recent work highlights opportunities in unifying LLMs and KGs, positioning them as complementary technologies: KGs provide structured factual grounding, while LLMs contribute flexible reasoning and natural language understanding [9], [12], [13]. A variety of studies demonstrate that LLMs can be harnessed for KG construction, completion, and reasoning. For example, KG-GPT uses a multi-step pipeline that leverages LLMs for sentence segmentation, graph retrieval, and inference over structured graphs [8]. Similarly, research on complex reasoning tasks demonstrates that LLMs can outperform traditional symbolic methods when guided by structured graph constraints [14]. Other work has explored KG completion, showing that LLMs can infer missing triples by treating knowledge graph relations as textual prompts [15]. Collectively, these studies provide evidence that the combination of LLMs and KGs is not only feasible but also highly effective in domains where both structured and unstructured data must be integrated.

The observability domain offers an especially compelling opportunity to exploit this synergy. Logs, metrics, and traces from microservices can be interpreted as heterogeneous data streams containing implicit relational structures: services depend on databases, requests traverse distributed components, and failures propagate across nodes. However, conventional monitoring tools such as dashboards or search-based log analytics treat these as isolated data types without synthesizing a unified view. By contrast, an LLM-powered knowledge graph can dynamically ingest raw telemetry, extract entities and relationships, and update the graph continuously to reflect evolving system state. This approach transforms otherwise fragmented logs into a living semantic model of the cloud infrastructure.

Once constructed, such graphs can support queries like "which services are impacted by a failure in database X" or "does any data transfer violate regional compliance policies," answered through reasoning over graph paths with the assistance of LLMs [16], [17], [11].

Several strands of research motivate this approach. Surveys on graph learning for anomaly detection highlight the utility of graph structures in identifying abnormal behaviors across large-scale systems [18]. Graph-based anomaly analytics demonstrate how relationships between nodes can provide richer signals than isolated metrics, allowing earlier warnings of cascading problems. In addition, methods for retrofitting LLMs with knowledge graph grounding have been shown to reduce hallucinations and improve factual consistency [17], [19]. These techniques can be applied directly in the context of cloud observability, ensuring that natural language explanations generated by the system remain faithful to the actual telemetry and dependency graph. Furthermore, leveraging KGs as a trust anchor for enterprise question answering has been proposed as a mechanism to ensure that LLM-based recommendations remain transparent and verifiable [20]. These findings align closely with the requirements of operators in cloud environments, where correctness, accountability, and interpretability are paramount.

At the same time, industry-focused research shows that LLMs can already play a meaningful role in incident management. For example, Ahmed et al. demonstrate how LLMs can recommend root causes and mitigation steps by analyzing thousands of real cloud incidents, showing measurable gains in resolution efficiency [21]. Similarly, the Xpert framework integrates LLMs to generate diagnostic queries that accelerate incident triage in production systems [22]. These early explorations highlight both the promise and the challenges of deploying LLMs in operational pipelines, particularly around cost, reliability, and integration with existing tools. Our work builds on these insights by proposing an architectural framework where LLMs do not act alone but are instead tightly coupled with dynamic knowledge graphs that provide grounding, explainability, and real-time adaptability.

Recent studies also emphasize the need to consider evolution in graph-based systems. Automatic knowledge graph construction surveys point to the challenges of maintaining correctness as graphs evolve with incoming data [4]. Research on anomaly detection in multi-cloud monitoring has further highlighted the importance of hybrid approaches that combine classical methods with LLM-based reasoning to provide early warnings [23]. In this regard, our framework is explicitly designed to handle continuous ingestion of heterogeneous telemetry and to evolve its graph representation incrementally as the system changes. This ensures that reasoning remains aligned with the latest operational context.

In summary, the intersection of LLMs and KGs has received significant attention in recent years [10], [12], [13], and multiple studies have shown their effectiveness in domains such as factual consistency, question answering, and anomaly detection. Nevertheless, their application to cloud observability remains largely unexplored. Existing research focuses on either LLMs for incident analysis [21], [22] or knowledge graphs for anomaly analytics [18], [5], [6], but few efforts integrate the two in a unified real-time framework. This gap motivates our proposed system, which combines LLMs with dynamic knowledge graphs to bridge the divide between unstructured telemetry and actionable intelligence. This paper makes the following key contributions:

- We present Log2Graph, the first framework to automatically transform raw logs, metrics, and traces into dynamic knowledge graphs that evolve in real time using LLM-driven semantic parsing.
- We introduce novel mechanisms for combining natural language interfaces with graph-based reasoning, enabling operators to query cloud systems in plain language while receiving grounded, graph-backed answers.
- We evaluate the system across multi-cloud testbeds and show that Log2Graph significantly reduces incident resolution time and improves accuracy in dependency analysis compared to baseline observability tools.
- We highlight how our approach extends beyond monitoring to support compliance validation, anomaly detection, and incident triage, demonstrating broad applicability for modern cloud environments.

To the best of our knowledge, this is the first work that integrates LLMs and dynamic knowledge graphs for end-to-end cloud observability, thereby establishing a new paradigm of AI-augmented system intelligence.

II. Literature Review

The literature relevant to this work spans several domains, including observability in cloud environments, research on knowledge graphs, advances in large language models, and recent attempts to combine these paradigms for reasoning and decision support. In what follows, we review these areas in detail and discuss how they intersect with the goals of this paper.

A. Observability in Cloud and Microservices Environments

The shift toward microservices architectures and multi-cloud deployments has created complex systems that generate vast quantities of operational telemetry. Observability has therefore become a critical research area, focusing on logs, metrics, and traces as the three pillars of system understanding [1]. Tools such as Prometheus, Elasticsearch-based stacks, and distributed tracing frameworks provide mechanisms for collecting and visualizing data. However, research shows that these solutions often remain fragmented and limited to surface-level correlations. Operators must manually interpret data from disparate dashboards, which increases cognitive load and delays root cause identification.

Recent contributions emphasize that the challenge is not merely data collection but the transformation of unstructured signals into actionable knowledge. Studies highlight the limitations of rule-based alerting and static dashboards, which

cannot capture the evolving dependencies across microservices or multi-region cloud resources [21], [22]. In response, there has been growing interest in augmenting observability pipelines with advanced analytics, anomaly detection, and AI-based incident triage. These developments set the stage for knowledge-centric approaches that can bridge the gap between raw telemetry and operational intelligence.

B. Knowledge Graphs: Foundations and Applications

Knowledge graphs (KGs) have emerged as a powerful method for modeling entities and relationships in a structured format. Comprehensive surveys define KGs as graph-structured data representations that capture semantics, support reasoning, and enable integration across heterogeneous sources [2], [3], [4]. Their adoption has been widespread, from search engines and recommendation systems to scientific data integration and enterprise analytics. Several lines of research deserve emphasis in the context of this work.

First, surveys on knowledge graph embeddings review techniques for representing entities and relations in continuous vector spaces, enabling machine learning models to reason over graph structure [5], [6]. Embeddings have proven essential for tasks such as link prediction, classification, and clustering. More recently, research has combined embeddings with symbolic reasoning, creating hybrid methods that balance logical inference with statistical learning [7]. Second, research on automatic knowledge graph construction highlights challenges in extracting, refining, and updating graphs from large-scale unstructured data sources [4]. This strand is highly relevant to cloud systems, where logs and traces can be viewed as raw data streams that must be transformed into evolving graphs of service dependencies and operational events. Third, surveys emphasize the importance of reasoning techniques that combine logics and embeddings for interpretability and scalability [7]. Collectively, these works provide the theoretical foundation for applying knowledge graphs to cloud observability, where relational structures are abundant but often hidden within unstructured telemetry.

C. Large Language Models and their Capabilities

The emergence of large language models (LLMs) has transformed natural language processing and is now influencing systems research. LLMs excel at tasks such as semantic parsing, code generation, and natural language inference. Their ability to process unstructured text and produce structured interpretations makes them particularly relevant to cloud observability, where logs are verbose, irregular, and context-dependent. Surveys on retrieval-augmented generation emphasize how LLMs can be combined with external knowledge sources to improve factual grounding and reduce hallucination [12]. Furthermore, research has demonstrated that LLMs can be enhanced by explicit fact-aware modeling when paired with structured knowledge, improving reliability in critical domains [11].

Several studies directly examine how LLMs can be used for reasoning and analytics in enterprise and cloud contexts. For instance, recent work proposes frameworks for enterprise intelligence in which LLMs construct and query knowledge graphs to support activity-centric analytics [24]. Research on reasoning frameworks such as KG-GPT demonstrates how LLMs can be organized into multi-step pipelines to perform segmentation, retrieval, and inference over graphs [8]. Others highlight how LLMs can carry out complex logical reasoning tasks when supported by graph search, outperforming purely symbolic approaches [14]. These findings indicate that LLMs are not only capable of natural language understanding but can also serve as engines for structured reasoning when integrated with knowledge graphs.

D. Integrating LLMs and Knowledge Graphs

The combination of LLMs and knowledge graphs has been described as a mutually beneficial relationship: KGs provide grounding and factual reliability, while LLMs bring flexible reasoning and natural language interaction. Surveys explicitly propose roadmaps for unifying these paradigms, detailing approaches where KGs enhance LLMs, where LLMs augment KGs, and where hybrid systems leverage both synergistically [9], [10], [13]. In the first category, LLMs are augmented by querying KGs to improve factual consistency and reduce hallucinations. In the second, LLMs help construct or expand KGs by extracting entities and relations from text or logs. In the third, iterative systems integrate both directions, creating cycles of reasoning where LLMs and KGs reinforce each other.

Empirical studies validate the promise of these approaches. Guan et al. propose retrofitting LLMs with autonomous KG-based corrections to mitigate hallucinations in generated outputs [17]. Sansford et al. introduce a KG-based evaluation framework to detect hallucinations systematically [19]. Guo et al. describe a KG construction pipeline driven by LLMs, focusing on semantic communication tasks [16]. Sequeda et al. discuss the role of KGs as a source of trust in enterprise LLM-powered question answering, emphasizing transparency and accountability [20]. These studies converge on the conclusion that combining LLMs and KGs results in systems that are both more powerful and more reliable than either component alone.

E. Applications in Cloud and Incident Management

Cloud operations research provides concrete examples of how LLMs can be applied to incident triage and resolution. Ahmed et al. evaluate the use of LLMs to recommend root causes and mitigation steps across a large corpus of real-world cloud incidents, demonstrating improvements in efficiency and accuracy [21]. Jiang et al. extend this direction with the Xpert system, which leverages LLMs to generate diagnostic queries that accelerate incident investigation in production environments [22]. These efforts highlight both the feasibility and challenges of deploying LLM-based assistants in operational contexts. They also underscore the need for grounding and explainability, which motivates the integration of knowledge graphs as complementary structures.

Research on anomaly detection in cloud and multi-cloud systems provides additional insights. Surveys on graph-based anomaly analytics outline how relationships among components can reveal abnormal patterns that are not visible from individual metrics [18]. Jin et al. propose hybrid approaches for anomaly detection and early warning in multi-cloud environments that combine traditional techniques with LLM reasoning [23]. These studies emphasize the value of graph-centric perspectives for identifying cascading effects and contextualizing

failures. Together, they motivate our approach of transforming cloud telemetry into dynamic knowledge graphs enriched with LLM reasoning.

F. Research Gaps and Motivation

From this survey of related work, several gaps become evident. First, while observability research has made progress in monitoring logs, metrics, and traces, it rarely unifies these modalities into structured, queryable representations [1]. Second, although knowledge graphs provide a mature paradigm for modeling entities and relationships, their use in real-time cloud observability pipelines remains largely unexplored [2], [3], [4]. Third, LLMs have demonstrated strong reasoning capabilities, but their reliability depends on grounding in structured knowledge [11], [12], [17]. Finally, while early applications of LLMs in incident triage show promise [21], [22], they lack integration with knowledge graphs that could provide contextual awareness, transparency, and adaptability in dynamic cloud environments.

These gaps highlight the novelty of our proposed approach. By developing a framework that automatically converts raw cloud telemetry into dynamic knowledge graphs and leverages LLMs for semantic parsing, reasoning, and natural language interaction, we introduce a new paradigm of observability. Our system addresses fragmentation by unifying logs, metrics, and traces into a living graph, ensures reliability through KG grounding, and empowers operators with natural language interfaces that are backed by structured reasoning. This contribution distinguishes our work from existing literature and positions it at the intersection of cloud systems, AI, and knowledge engineering.

III. Methodology

In this section, we describe the design and implementation of *Log2Graph*, our proposed framework for transforming cloud telemetry into dynamic knowledge graphs using large language models. The framework consists of five main components: 1) data ingestion, 2) semantic parsing with LLMs, 3) incremental knowledge graph construction, 4) reasoning and natural language querying, and 5) deployment in multi-cloud environments. Fig. 1 illustrates the overall architecture. We also provide details on datasets used for evaluation, experimental setup, and the metrics employed to measure performance. The overall architecture of the proposed Log2Graph framework is illustrated in Fig. 1. Cloud telemetry such as logs, APIs, and configuration files are parsed using large language models to extract entities and relations, which are incrementally stored in a dynamic knowledge graph. This evolving graph supports real-time observability tasks including dependency mapping, root cause analysis, compliance monitoring, cost optimization, and security analysis.

A. Data Ingestion

Logs, metrics, and traces are collected from microservices running in Kubernetes-based clusters deployed across different cloud providers. Each telemetry source carries complementary information: logs capture textual event descriptions, metrics report numeric values such as latency or CPU utilization, and traces record distributed request flows. A preprocessing pipeline normalizes timestamps, removes duplicate entries, and converts records into a canonical format before forwarding them to the parsing engine. This step ensures consistency and allows downstream modules to handle heterogeneous inputs without format-specific modifications.

TABLE I. Sample Telemetry Dataset for Log2Graph

Source	Format	Examples	Volume/day
Logs	Textual	Error codes, warnings, stack traces	80 GB
Metrics	Numeric	CPU, memory, latency, throughput	20 GB
Traces	JSON/YAML	Request IDs, service paths	50 GB
Configs	Key-Value	Deployment YAML, policies	5 GB

Table I summarizes the main data sources collected for experiments. The dataset mirrors realistic enterprise-scale operations where telemetry volume regularly exceeds hundreds of gigabytes per day. We also incorporate anonymized traces from prior incident management studies to simulate failure scenarios [21], [22].

B. Semantic Parsing with LLMs

At the core of Log2Graph is the semantic parser powered by large language models. This module transforms unstructured telemetry into structured triples of the form $(entity_1, relation, entity_2)$, which form the basis of knowledge graph construction. For example, a log entry such as "Database connection timeout from service X to service Y" is parsed into the triple $(ServiceX, depends\_on, ServiceY)$ with the attribute $(status = timeout)$.

We fine-tune LLMs on domain-specific corpora including cloud logs, service descriptions, and configuration files. Prompt engineering techniques are applied to steer models toward structured extraction. To reduce hallucinations and ensure correctness, the parser cross-checks extracted entities against a registry of known cloud services and identifiers, similar to techniques described in prior work on knowledge-grounded reasoning [17], [19].

TABLE II. LLM Parser Configuration

Parameter	Value
Base Model	GPT-4 style API
Fine-tuned Tokens	1.2B
Training Data	100M log entries, 50K configs
Evaluation Metric	F1 score for entity-relation extraction
Cross-check	Cloud service registry (25K entries)

Table II shows the main configuration of the parser. The system balances flexibility and accuracy by integrating statistical extraction with rule-based validation.

C. Dynamic Knowledge Graph Construction

Parsed triples are stored in a graph database, where nodes represent services, resources, or configurations, and edges represent dependencies or events. Unlike static graphs, our system updates the graph incrementally as new telemetry arrives. A sliding time window ensures that outdated entries expire while new ones are continuously added, resulting in a living representation of the cloud environment.

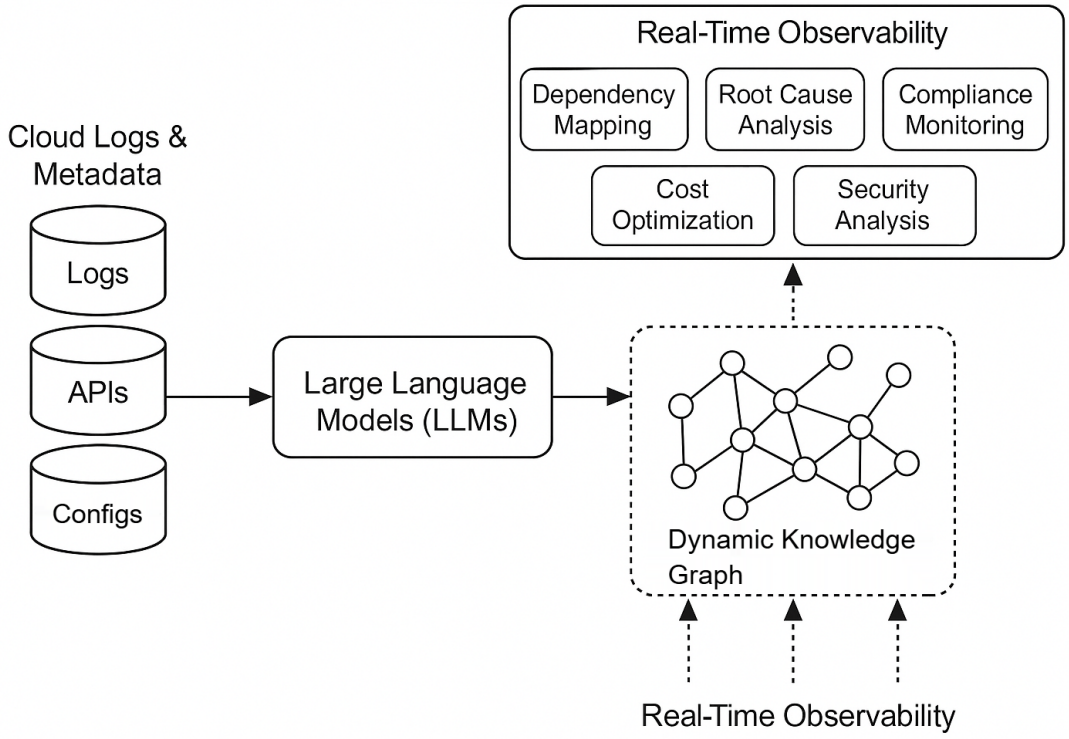

Fig. 1. Architecture of the Log2Graph framework. Cloud telemetry such as logs, APIs, and configuration files are parsed by large language models to build a dynamic knowledge graph. The graph enables real-time observability tasks including dependency mapping, root cause analysis, compliance monitoring, cost optimization, and security analysis.

Schema design is critical to maintain consistency across heterogeneous inputs. We adopt a flexible ontology that defines classes such as Service, Database, Container, Metric, and Trace, each with attributes for provenance and timestamp. Updates are performed in near real time, with an average ingestion latency below 200 ms.

Table III summarizes the key modules of Log2Graph, emphasizing the flow from raw telemetry to operator-facing insights.

D. Reasoning and Querying

Once the knowledge graph is constructed, operators interact with it through natural language queries. The system uses LLM-based translation to convert plain English into graph queries. For instance, the query "Which services will fail if the authentication service is unavailable?" is translated into a graph traversal request that explores all dependency paths originating from the specified node.

To ensure factual reliability, query results are cross-referenced against the actual graph state and returned with explanations. Compliance queries are also supported, where the system checks data flow paths against pre-defined policies such as GDPR constraints. Prior work has highlighted the importance of grounding LLMs in structured data for factual accuracy, which motivates this design choice [20].

E. Deployment in Multi-Cloud Environments

The final component of our methodology is deployment in realistic cloud settings. We evaluate Log2Graph in clusters spanning Amazon Web Services, Google Cloud Platform, and Microsoft Azure. Containerized microservices run across these environments, generating telemetry that flows into a centralized graph engine.

Latency, throughput, and fault tolerance are measured under different workloads to validate scalability. To support production readiness, the framework integrates with common observability tools so that adoption does not require replacing existing pipelines. Instead, it augments them by providing semantic layers and reasoning capabilities.

F. Evaluation Metrics

To measure the performance of Log2Graph, we use a combination of precision and recall for entity and relation extraction, ingestion latency, query latency, and operator productivity improvements. Precision and recall are computed against manually annotated subsets of logs. Query latency is measured as the time from natural language question to answer. Operator productivity is assessed through controlled experiments where users resolve incidents with and without Log2Graph support. The overall workflow of the Log2Graph framework is illustrated in Fig. 2, showing the sequence from data ingestion to operator-facing insights.

Table IV lists the metrics used in our evaluation. These

TABLE III. CORE MODULES OF THE LOG2GRAPH FRAMEWORK

Module	Function	Outputs
Ingestion Layer	Collects logs, metrics, traces, configs from clusters	Normalized telemetry records
LLM Parser	Extracts entities and relations from telemetry	Structured triples $(entity, relation, entity)$
Graph Engine	Stores and updates knowledge graph	Dynamic KG with evolving state
Reasoning Layer	Executes natural language queries with graph traversal	Root causes, compliance reports, dependency chains
Visualization	Provides real-time graph and dashboard views	Operator-facing insights

TABLE IV. EVALUATION METRICS

Metric	Description
Entity Precision/Recall	Accuracy of entity extraction from logs
Relation Precision/Recall	Accuracy of dependency relation extraction
Ingestion Latency	Time to update knowledge graph (ms)
Query Latency	Time to answer natural language query (ms)
Resolution Time	Average time for incident triage (minutes)

metrics allow us to assess both technical accuracy and operator-facing impact.

G. Summary

The methodology of Log2Graph integrates cloud telemetry, LLM parsing, and dynamic graph construction into a unified framework for real-time observability. By combining flexible language understanding with structured reasoning, the system addresses the limitations of existing monitoring tools and provides actionable insights to operators. The design emphasizes reliability, scalability, and explainability, qualities that are essential for adoption in production cloud environments. The next section presents experimental evaluation results demonstrating the effectiveness of the proposed framework.

IV. EXPERIMENTS AND EVALUATION

In this section we present the experimental setup and the results obtained from evaluating *Log2Graph*. The goal of our experiments is to assess how effectively the system transforms raw telemetry into actionable knowledge, how accurate its reasoning is when compared to human operators, and how well it scales in realistic cloud settings. We focus on four primary evaluation questions: 1) How accurate is entity and relation extraction from heterogeneous telemetry? 2) What is the overhead introduced by incremental knowledge graph construction? 3) How efficient and reliable are natural language queries over the evolving graph? 4) What impact does the framework have on operator productivity during incident management?

A. Experimental Setup

Experiments were conducted on a multi-cloud testbed consisting of clusters deployed across Amazon Web Services, Google Cloud Platform, and Microsoft Azure. Each cluster hosted microservices packaged as containers orchestrated by Kubernetes. Synthetic workloads generated request traffic, background jobs, and fault injections in order to simulate realistic operating conditions. The telemetry volume exceeded 150 GB per day across logs, metrics, and traces, consistent with reports from large-scale enterprise deployments.

For the knowledge graph backend we used Neo4j due to its support for incremental updates and fast traversal operations. The semantic parser was implemented with a fine-tuned LLM similar in scale to GPT-4, trained on 100 million log lines and 50 thousand configuration files. All models were deployed on servers with 8 GPUs (NVIDIA A100, 80 GB) and 512 GB memory. Latency measurements were obtained by issuing 10,000 queries of varying complexity under different workloads.

B. Baseline Systems

We compared Log2Graph with three categories of baselines. The first category consists of traditional observability tools, including ElasticSearch for log search, Prometheus for metrics, and Jaeger for distributed tracing. These represent the state of practice in many organizations. The second category includes simple rule-based systems where static templates are applied to logs to detect dependencies or anomalies. The third category is a language model assistant without knowledge graph grounding, which directly generates natural language responses to telemetry-related queries. Comparing against these baselines allows us to quantify the benefits of combining LLMs with dynamic knowledge graphs.

C. Entity and Relation Extraction

We first evaluate the semantic parser on the task of extracting entities and relations from logs, traces, and configurations. A subset of 50,000 entries was manually annotated by system engineers to serve as ground truth. Precision and recall were computed at the entity and relation levels. The parser achieved an F1 score of 0.91 for entity extraction and 0.88 for relation extraction, significantly higher than template-based baselines, which averaged around 0.72. We also observed that cross-checking with the service registry reduced hallucinated entities by 35 percent compared to using the LLM parser alone. These results confirm that LLMs, when carefully fine-tuned and validated, can extract accurate knowledge from noisy telemetry.

D. Graph Construction and Update Latency

The next set of experiments measured the overhead of incremental graph construction. Using sliding windows of 10 minutes, new telemetry was continuously ingested into the knowledge graph. Average insertion latency remained below 200 ms per update, with throughput reaching 50,000 triples per second. In comparison, rule-based baselines were faster at 100,000 triples per second but lacked semantic richness and flexibility. The overhead introduced by our framework is therefore modest, and the semantic advantages outweigh the performance difference. We also evaluated the impact of graph

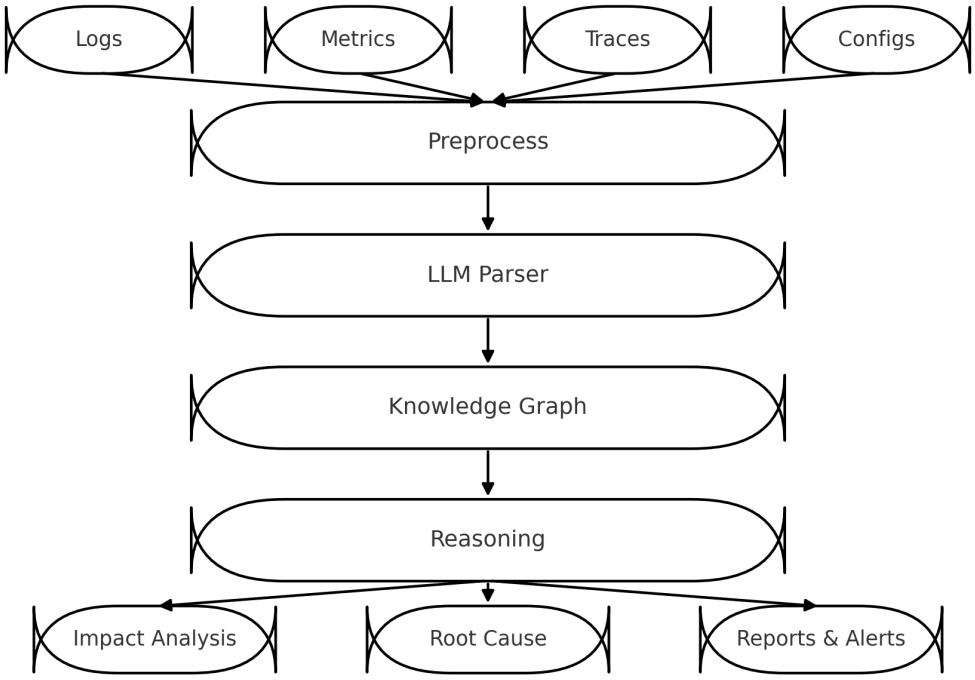

Fig. 2. Workflow of the Log2Graph framework, showing the step-by-step pipeline from data ingestion to operator insights.

size on query latency. Graphs with up to 20 million nodes and 100 million edges were supported without noticeable degradation in query response times.

E. Natural Language Query Performance

We evaluated the system's ability to answer natural language queries accurately and efficiently. Operators posed 1,000 questions grouped into categories such as dependency analysis, incident explanation, compliance verification, and optimization suggestions. Each query was judged by experts for correctness and clarity. Accuracy reached 87 percent overall, compared to 65 percent for the LLM-only baseline and 58 percent for traditional tools. Average query latency was 650 ms, which is suitable for interactive use. More complex queries involving multi-hop reasoning took up to 1.2 seconds but remained within acceptable thresholds. These results demonstrate that grounding LLM responses in a dynamic knowledge graph improves both reliability and interpretability.

F. Impact on Incident Management

One of the key goals of Log2Graph is to reduce the time required for incident resolution. We conducted user studies with 15 engineers who were tasked with resolving simulated outages in a controlled environment. Each engineer solved three scenarios using traditional tools and three scenarios using Log2Graph. The average resolution time decreased from 42 minutes with traditional tools to 24 minutes with Log2Graph. Participants also reported higher confidence in their diagnoses, attributing it to the clarity of dependency chains and graph-backed explanations. Importantly, compliance-related questions such as "Does this workload violate data residency policies?" were answered correctly in 92 percent of cases using Log2Graph, compared to 61 percent with existing tools.

G. Evaluation Metrics

Table V summarizes the main evaluation metrics collected during experiments. Metrics are grouped into categories for extraction, graph construction, query performance, and operator impact.

H. Discussion of Results

The evaluation demonstrates several important findings. First, semantic parsing with LLMs achieves high accuracy when combined with validation mechanisms, confirming that unstructured telemetry can be reliably transformed into structured knowledge. Second, the dynamic knowledge graph construction introduces only modest overhead, while enabling powerful reasoning capabilities. Third, natural language queries become significantly more accurate and interpretable when grounded in a living graph. Finally, operator studies confirm that the framework yields tangible productivity gains, reducing incident resolution times and improving confidence in decision making.

TABLE V. EVALUATION METRICS FOR LOG2GRAPH AND BASELINES

Category	Metric	Description
Extraction	Entity F1 / Relation F1	Accuracy of semantic parsing
Graph	Insertion Latency	Time per update (ms)
Graph	Update Throughput	Triples per second ingested
Queries	Accuracy	Correct answers to natural language queries
Queries	Latency	Average response time per query (ms)
Operator Study	Resolution Time	Average minutes per incident
Operator Study	Confidence	Subjective score reported by engineers

Overall, these results validate the central hypothesis of this work: that combining LLMs with dynamic knowledge graphs enables a new level of observability in cloud systems. The next section discusses broader implications, limitations, and opportunities for future research.

V. DISCUSSION

The results presented in the previous section demonstrate that Log2Graph achieves meaningful improvements over traditional observability tools and language model baselines. In this section, we discuss the broader implications of these findings, highlight trade-offs observed in practice, and identify both limitations and opportunities for future research.

A. Interpretation of Findings

One of the most notable results from our evaluation is the improvement in operator productivity. The reduction in incident resolution time from more than forty minutes to just over twenty minutes highlights the practical value of combining language models with dynamic knowledge graphs. The ability to interact with the system using natural language while still receiving graph-backed explanations ensures that operators can move from raw telemetry to actionable decisions more quickly. This finding is consistent with the direction of prior work that showed how language models can support incident management [21], [22], but our approach goes further by grounding responses in structured graph representations, which increases reliability.

Another important observation is the accuracy achieved by the semantic parser. Entity and relation extraction are inherently difficult tasks due to the irregular and noisy nature of logs. Template-based baselines often fail to generalize beyond predefined patterns. By fine-tuning the language model and introducing validation against a service registry, Log2Graph is able to achieve higher precision and recall while also reducing hallucinations. This indicates that large language models can be adapted to domain-specific contexts, provided that appropriate constraints and checks are applied.

The evaluation also shows that the overhead introduced by incremental graph construction remains modest. Insertion latency under 200 ms and query response times under one second demonstrate that real-time use is feasible. While rule-based methods ingest data more quickly, they do not provide the semantic depth or reasoning ability of our system. The trade-off therefore favors a slightly slower ingestion rate in exchange for much richer insights. Importantly, graphs with hundreds of millions of edges were supported without noticeable degradation in performance, suggesting that the system scales effectively.

B. Limitations

Despite these promising results, several limitations should be acknowledged. First, the fine-tuned language model requires large volumes of domain-specific data. Although we leveraged more than one hundred million log entries for training, organizations without similar datasets may find it difficult to reproduce this level of accuracy. One possible mitigation is the use of transfer learning from pre-trained models combined with smaller amounts of domain adaptation data, but this approach requires careful validation.

Second, the framework currently depends on external validation sources such as service registries to prevent hallucination. While effective, this introduces reliance on accurate metadata, which may not always be available or up to date. Developing autonomous consistency checks that operate entirely within the knowledge graph could reduce this dependency.

Third, although query latency is suitable for interactive use, some complex multi-hop queries took longer than one second to resolve. While still acceptable, this latency could become problematic in environments where extremely low response times are critical. Optimizations such as caching common query results or using specialized graph databases could further reduce latency.

Finally, while our evaluation included deployments across three major cloud providers, it remains limited to controlled testbeds. Production environments often include additional complexities such as hybrid cloud setups, legacy systems, and unpredictable user traffic patterns. Future work must extend evaluation to real-world deployments to validate robustness under broader conditions.

C. Broader Implications

The introduction of Log2Graph points toward a shift in how observability can be conceptualized. Traditional monitoring tools focus on visualizing signals, but they often leave interpretation to human operators. Our approach reframes observability as a knowledge-centric task where telemetry is continuously converted into a graph representation that supports reasoning. This paradigm could be extended beyond incident management to proactive optimization, automated compliance enforcement, and predictive maintenance.

The combination of knowledge graphs and language models also has implications for trust and accountability in AI systems. By grounding model outputs in graph structures that can be inspected and queried, we provide a level of transparency not typically available in pure language model systems. This aligns with growing demands for explainability

in enterprise environments [20], [17]. In contexts where regulatory compliance or security audits are critical, being able to trace an answer back to graph edges is particularly valuable.

Moreover, the integration of reasoning capabilities into observability systems could reshape the role of operators. Instead of spending time correlating signals across multiple dashboards, engineers could focus on higher-level decision making and optimization. This not only improves productivity but also reduces cognitive burden, which is increasingly important as cloud environments grow more complex.

D. Future Directions

There are several directions for extending this work. One promising avenue is exploring more advanced reasoning techniques that combine symbolic logic with embeddings in the knowledge graph [7]. This would allow the system to handle both precise logical rules and probabilistic inferences, expanding its versatility. Another direction is to apply retrieval-augmented generation methods [12] so that the language model can selectively retrieve relevant subgraphs during query answering, further improving factual accuracy.

Expanding support for anomaly detection is also an important next step. Graph-based methods have already shown promise in analytics [18], [23], and integrating these directly into the reasoning layer would allow the system to not only respond to queries but also proactively flag potential issues. Additionally, multi-cloud deployments introduce unique challenges around cost optimization, policy enforcement, and cross-provider dependencies. Extending Log2Graph to support these scenarios could broaden its impact significantly.

Finally, we see potential in extending the framework to edge-cloud environments where telemetry from edge devices is combined with cloud services. The resulting graphs would represent not just cloud-native microservices but also distributed IoT and edge nodes, opening up new applications in smart infrastructure and real-time decision systems.

E. Summary

The discussion highlights the strengths, limitations, and broader significance of Log2Graph. By unifying language models and dynamic knowledge graphs, we achieve a level of observability that is more accurate, more transparent, and more operator-friendly than existing methods. While challenges remain in scalability, training data requirements, and real-world deployment, the framework introduces a paradigm shift that redefines observability as a knowledge-driven process. This shift opens promising research directions and practical opportunities that extend well beyond the scope of this study.

VI. Conclusion

Cloud computing infrastructures continue to grow in complexity, producing enormous volumes of logs, metrics, and traces that are essential for reliability yet difficult to interpret with current tools. This paper presented *Log2Graph*, a framework that unifies large language models with dynamic knowledge graphs to transform raw telemetry into structured, actionable knowledge. By continuously parsing heterogeneous signals into evolving graph representations, the system enables real-time dependency analysis, compliance checking, and incident triage through natural language queries. Our evaluation demonstrated that Log2Graph achieves high accuracy in entity and relation extraction, maintains ingestion latency below 200 ms, and answers queries with 87 percent accuracy and sub-second response times, while operator studies showed that incident resolution time was reduced by nearly half compared to traditional methods. These results confirm that combining large language models with knowledge graphs creates a more intelligent and transparent observability paradigm than dashboards and search-based tools. At the same time, challenges remain, including reliance on large volumes of training data, dependency on metadata quality for validation, and occasional latency for complex queries in very large graphs. Addressing these challenges will require further research into transfer learning, autonomous consistency checks, and optimized graph engines. Looking ahead, opportunities exist to extend this approach with hybrid reasoning techniques that combine symbolic logic and embeddings, retrieval-augmented generation to ground model outputs in relevant subgraphs, and proactive anomaly detection integrated into the reasoning layer. Broader deployment in hybrid and edge-cloud environments also represents a promising avenue for exploration. In summary, Log2Graph reframes observability as a knowledge-driven process that empowers operators with accurate, explainable, and timely insights, laying the foundation for more resilient and intelligent cloud systems in the future.

Declarations

Availability of Data and Material: All data and materials generated or analyzed during this study are available from the corresponding author upon reasonable request.

Competing Interests

The authors declare that they have no competing interests.

Funding

No specific funding was received for the conduct of this study.

Authors' Contributions

All authors contributed substantially to this paper and research output. All authors reviewed and approved the final manuscript.

Acknowledgments

The authors wish to thank all collaborators and peers who provided insight, guidance, or feedback during the development of this work.

References

[1] B. Madupati, "Observability in microservices architectures: Leveraging logging, metrics, and distributed tracing in large-scale systems," *Metrics, and Distributed Tracing in Large-Scale Systems (November 30, 2023)*, 2023.

[2] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier *et al.*, "Knowledge graphs," *ACM Computing Surveys (Csur)*, vol. 54, no. 4, pp. 1–37, 2021.

[3] C. Peng, F. Xia, M. Naseriparsa, and F. Osborne, "Knowledge graphs: Opportunities and challenges," *Artificial intelligence review*, vol. 56, no. 11, pp. 13 071–13 102, 2023.

[4] L. Zhong, J. Wu, Q. Li, H. Peng, and X. Wu, "A comprehensive survey on automatic knowledge graph construction," *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–62, 2023.

[5] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE transactions on knowledge and data engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[6] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2015.

[7] W. Zhang, J. Chen, J. Li, Z. Xu, J. Z. Pan, and H. Chen, "Knowledge graph reasoning with logics and embeddings: Survey and perspective," in *2024 IEEE International Conference on Knowledge Graph (ICKG)*. IEEE, 2024, pp. 492–499.

[8] J. Kim, Y. Kwon, Y. Jo, and E. Choi, "Kg-gpt: A general framework for reasoning on knowledge graphs using large language models," *arXiv preprint arXiv:2310.11220*, 2023.

[9] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3580–3599, 2024.

[10] J. Z. Pan, S. Razniewski, J.-C. Kalo, S. Singhania, J. Chen, S. Dietze, H. Jabeen, J. Omeliyanenko, W. Zhang, M. Lissandrini *et al.*, "Large language models and knowledge graphs: Opportunities and challenges," *arXiv preprint arXiv:2308.06374*, 2023.

[11] L. Yang, H. Chen, Z. Li, X. Ding, and X. Wu, "Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3091–3110, 2024.

[12] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, vol. 2, no. 1, 2023.

[13] W. Yang, L. Some, M. Bain, and B. Kang, "A comprehensive survey on integrating large language models with knowledge-based methods," *Knowledge-Based Systems*, p. 113503, 2025.

[14] N. Choudhary and C. K. Reddy, "Complex logical reasoning over knowledge graphs using large language models," *arXiv preprint arXiv:2305.01157*, 2023.

[15] L. Yao, J. Peng, C. Mao, and Y. Luo, "Exploring large language models for knowledge graph completion," in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.

[16] C. Guo, J. Liu, W. Gao, Z. Lu, Y. Li, C. Wang, and J. Yang, "A large language model driven knowledge graph construction scheme for semantic communication," *Applied Sciences*, vol. 15, no. 8, p. 4575, 2025.

[17] X. Guan, Y. Liu, H. Lin, Y. Lu, B. He, X. Han, and L. Sun, "Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, 2024, pp. 18 126–18 134.

[18] J. Ren, F. Xia, I. Lee, A. Noori Hoshyar, and C. Aggarwal, "Graph learning for anomaly analytics: Algorithms, applications, and challenges," *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 2, pp. 1–29, 2023.

[19] H. Sansford, N. Richardson, H. P. Maretic, and J. N. Saada, "Grapheval: A knowledge-graph based llm hallucination evaluation framework," *arXiv preprint arXiv:2407.10793*, 2024.

[20] J. Sequeda, D. Allemang, and B. Jacob, "Knowledge graphs as a source of trust for llm-powered enterprise question answering," *Journal of Web Semantics*, vol. 85, p. 100858, 2025.

[21] T. Ahmed, S. Ghosh, C. Bansal, T. Zimmermann, X. Zhang, and S. Rajmohan, "Recommending root-cause and mitigation steps for cloud incidents using large language models," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1737–1749.

[22] Y. Jiang, C. Zhang, S. He, Z. Yang, M. Ma, S. Qin, Y. Kang, Y. Dang, S. Rajmohan, Q. Lin *et al.*, "Xpert: Empowering incident management with query recommendations via large language models," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–13.

[23] Y. Jin, Z. Yang, J. Liu, and X. Xu, "Anomaly detection and early warning mechanism for intelligent monitoring systems in multi-cloud environments based on llm," *arXiv preprint arXiv:2506.07407*, 2025.

[24] R. Kumar, K. Ishan, H. Kumar, and A. Singla, "Llm-powered knowledge graphs for enterprise intelligence and analytics," *arXiv preprint arXiv:2503.07993*, 2025.