ROOT CAUSE ANALYSIS WITH KNOWLEDGE GRAPHS AND LLMS

• Venkata Mahesh Kundurthi • Bharath Cherukuru • Ushabindu Velpula

MENTOR: DR. DALAL ALHARTHI

College of Information Science

Problem & Motivation

The Silo Effect: Modern observability suffers from disconnected data. Logs and metrics live in separate repositories, forcing manual correlation.

> ***Goal:*** *Automate the transition from correlation to causation using Knowledge Graphs to map how low-level faults (e.g., driver errors) propagate to high-level symptoms (e.g., CPU spikes).*

Methodology: Phase 1 & 2

Phase 1: Feature Engineering

Applied **Z-Score Normalization** (standardizing each metric relative to its mean and standard deviation) and **Event Burst Detection** to normalize heterogeneous data sources. A one-minute fuzzy temporal window was used to align events and metrics.

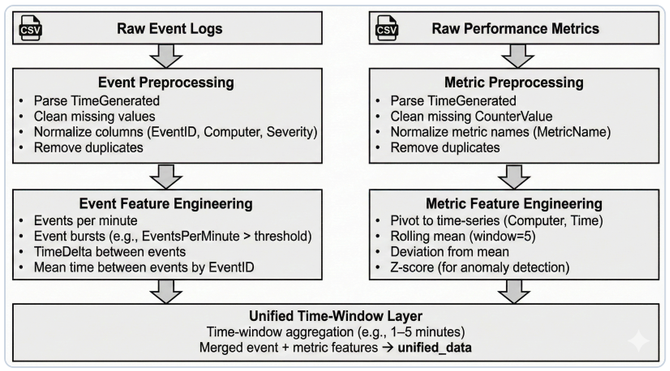


Fig 1: Pre-processing & Normalization Pipeline

Phase 2: Graph Construction

A directed graph was built using **NetworkX**. We generated PRECEDES edges when one event consistently occurred within 60 seconds before another, and CORRELATES edges for metric pairs whose Pearson correlation was greater than 0.5.

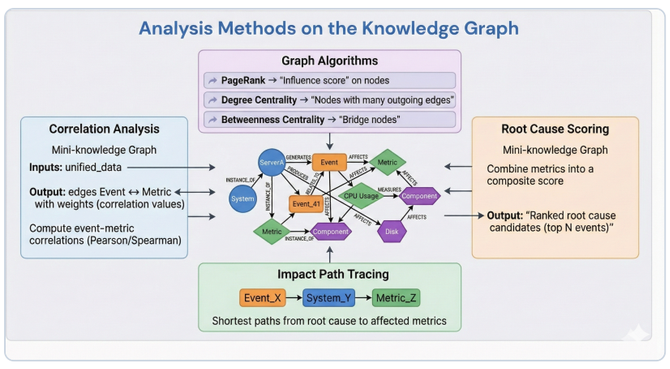


Fig 2: Knowledge Graph & Analysis Layer

Phase 3: Causal Inference

We identify the "Smoking Gun" using a composite centrality algorithm:

```
Score = 0.4(OutDegree) + 0.3(PageRank) +
0.3(Betweenness)
```

Logic: High *Out-Degree* suggests a node that triggers many downstream effects (root cause), while high *Betweenness* highlights key propagators of impact.

System Architecture

End-to-End Pipeline: From Raw Telemetry to GenAI Reporting

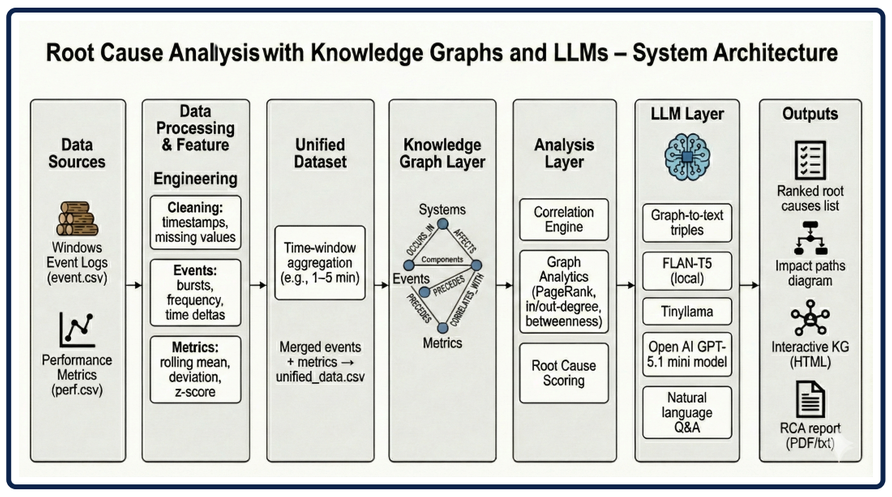


Figure 3: Complete System Architecture Flow

Graph Ontology (Schema)

Nodes	Edges
System: Host/VM	**PRECEDES:** Temporal
Component: Disk/CPU	**CORRELATES:** Statistical
Event: Log Signature	**AFFECTS:** Logical
Metric: Perf Counter	**HOSTS:** Structural

Phase 4: GenAI & RAG

Graph paths are converted into text "triples" for Retrieval Augmented Generation (RAG), allowing LLMs to reason over ranked impact paths and produce human-readable root-cause narratives.

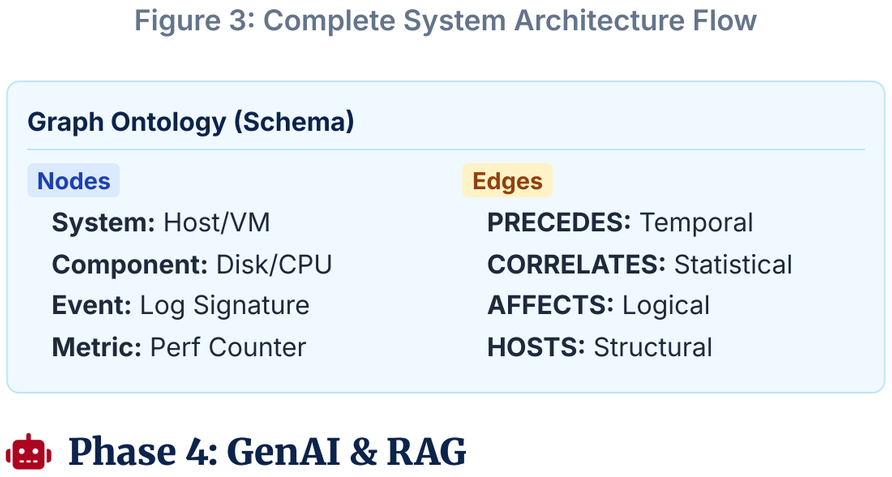


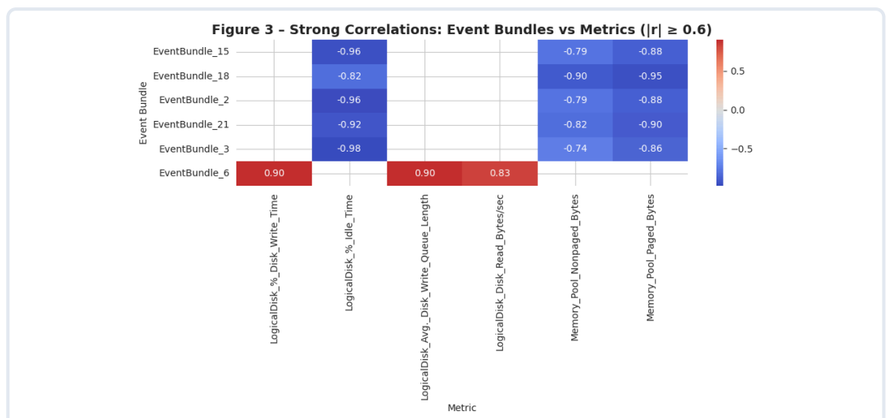
Fig 4: LLM Integration Layer

Model Stack:

FLAN-T5 (Local): Secure execution for sensitive logs.

TinyLLM (Edge): Efficient, low-latency inference.

GPT-5 Mini (Cloud): Narrative synthesis for executive reports.



Experimental Results

Tested on **forensicsacl2** dataset (June 26, 2024).

Rank	Candidate	Score
#1	**Event 16 (IOMMU Fault)** Hardware Driver Failure	0.71
#2	**Event 4672 (Privilege)** Escalation Loop	0.57

> ***Success:*** *Correctly ranked the hardware fault as the primary cause (#1) and the privilege escalation as a secondary effect (#2).*

Future Scope

Semantic Retrieval: Move from sparse keyword matching to dense retrieval using models such as LogBERT or Sentence-BERT. This helps resolve vocabulary mismatches (for example, "disk full" vs. "no space remaining").

Ontology-Based Reasoning: Evolve the current schema into a formal ontology with SWRL rules to enable automated logical inference of hidden system dependencies.

Dynamic Simulation: Implement Temporal Knowledge Graphs and Graph Neural Networks (GNNs) to run "What-If" simulations and predict how faults would propagate under different scenarios.

Ethics & Data Privacy

Local Execution: Sensitive logs are processed entirely on-premise using a local FLAN-T5 model, ensuring that raw operational data never leaves the secure environment.

Sanitization: Automated removal of personally identifiable information (PII)—including security identifiers (SIDs), user IDs, and file paths—before any external API interaction.

Hallucination Control: The RAG architecture constrains model outputs to verified Knowledge Graph edges only, reducing the risk of fabricated or unsupported explanations.

PROJECT REPOSITORY

SCAN ME

Capstone QR Code