



Topology-Aware Generative AI for Fault Diagnosis and Recovery Planning in Cloud Data Centers

Manoj Kumar Reddy Kalakoti

Texas A&M University, Kingsville, USA

ABSTRACT: Cloud data center incidents increasingly span compute, storage, GPU, network, and virtualization layers. A single physical fault can generate hundreds of alerts across layers, and engineers must correlate them, identify the root cause, and plan safe recovery under time pressure. Large language models (LLMs) can reason over heterogeneous operational text, but without grounding in the actual infrastructure they may hallucinate causes and propose recovery actions that violate redundancy, capacity, or dependency constraints.

This study presents a topology-aware generative AI framework for fault diagnosis and recovery planning. Telemetry from five infrastructure domains is normalized and correlated in time, and attached to a knowledge graph that represents the physical and logical topology. A retrieval-augmented LLM receives the blast-radius subgraph of an incident together with retrieved runbooks and prior incidents, and produces ranked root cause hypotheses with cited evidence and a structured recovery plan. Every plan passes through a constraint validator that checks topology consistency, blast radius, capacity and redundancy preservation, dependency order, and policy rules, followed by a dry-run simulation. Rejected plans are returned to the model with the violated constraint for repair. The framework was evaluated on historical and fault-injected incidents against rule-based correlation, an LLM with retrieval only, and an unvalidated LLM planner.

The framework achieved top-1 root cause accuracy of 82-88% and top-3 accuracy of 92-96%, compared with 55-65% top-1 for rule-based correlation and 68-75% for retrieval-only LLM without topology. Median time to recovery decreased by 35-50% relative to manual response. The unsafe-action rate of generated plans was 0-1% after validation, compared with 9-14% for the unvalidated LLM planner.

Conclusions: Grounding generative AI in infrastructure topology and enforcing deterministic validation of its recovery plans allows the benefits of LLM reasoning to be used in data center operations without accepting the risk of unsafe automated actions. The framework provides a replicable pattern for human-supervised, constraint-bounded AI operations.

KEYWORDS: Fault diagnosis; Root cause analysis; Large language models; Retrieval-augmented generation; Knowledge graph; Recovery planning; AIOps; Data center reliability

I. OPENING CONTEXT AND FOUNDATION

1.1 The Diagnosis Problem in Heterogeneous Data Centers

Modern cloud data centers are tightly coupled stacks of heterogeneous components. A single customer workload may depend on virtual machines scheduled on hypervisor hosts, GPUs attached through PCIe and interconnected by high-speed fabrics, storage reached over the network, and a switching topology that spans racks and pods. Faults in one layer routinely appear as symptoms in others. A degraded optical link may appear as storage latency in one application, as collective communication timeouts in a distributed training job, and as virtual machine restarts elsewhere, all at the same moment.

Failures of this kind are not rare. Studies of large-scale production systems report that hardware faults, network failures, and cascading software effects are routine, and that diagnosis time is a major component of outage duration [1], [2]. Operators face alert storms in which hundreds of alerts fire from independent monitoring systems, each of which reflects only a fragment of the underlying fault. The first minutes of an incident are spent determining which alerts matter, which are consequences rather than causes, and what the actual scope of impact is.



Large language models offer a plausible aid. They can read unstructured logs, tickets, and runbooks, summarize evidence, and propose hypotheses and remediation steps in natural language. However, using an LLM directly for data center operations raises two serious problems. First, an LLM has no inherent knowledge of the specific infrastructure it is asked about and may produce confident but incorrect diagnoses that cite components that do not exist or are not related. Second, the recovery plans it generates may be unsafe: draining the wrong host, rebooting a node holding the only healthy replica, or removing a switch that eliminates redundancy for an entire rack.

This paper addresses the problem of how to use retrieval-augmented generative AI for fault diagnosis and recovery planning in cloud data centers while guaranteeing that recommended actions respect infrastructure constraints. We present a topology-aware framework that correlates compute, storage, GPU, network, and virtualization telemetry, grounds the LLM in a knowledge graph of the infrastructure, and validates every plan before execution. We evaluate it on diagnosis accuracy, recovery time, and unsafe-action rate.

1.2 Characteristics of Cross-Layer Faults

Four properties of cross-layer faults motivate the design.

Symptom propagation. The component showing the strongest symptoms is rarely the one that failed. Failures propagate along dependency edges, such as network paths and storage mounts, and a root cause upstream can create symptoms across many downstream components.

Alert multiplicity. Each domain has its own monitoring stack, so one physical fault produces many alerts with different naming, timestamps, and severities.

Domain specificity. GPU failures involve error codes and thermal and interconnect behaviors that differ from CPU or disk failures, and distributed training jobs are sensitive to a single degraded link or device, which can stall an entire job [3].

Constraint-laden remediation. Recovery in a data center is governed by physical and logical constraints: redundancy requirements, quorum rules for distributed storage, spare GPU capacity, tenant isolation, maintenance windows, and ordering requirements such as draining before rebooting.

Fault Propagation Across Data Center Layers and Topology-Based Correlation

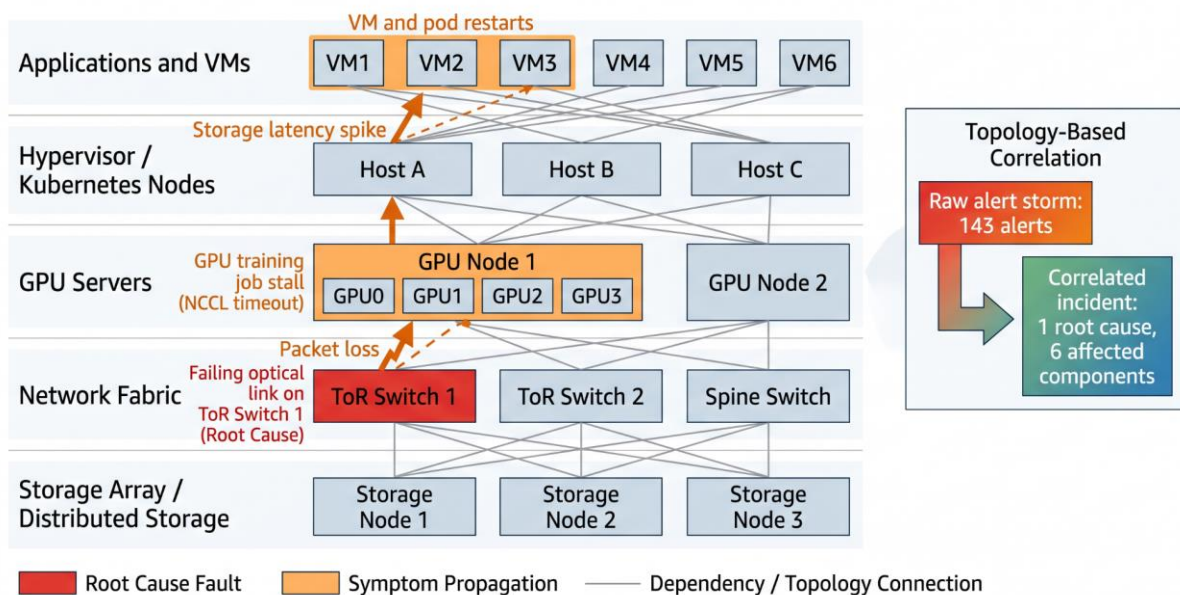


Figure 1. A single physical fault propagating across layers. A failing optical link on a top-of-rack switch (red) produces packet loss, GPU training stalls, storage latency spikes, and VM restarts (orange). Topology-based correlation reduces an alert storm of 143 alerts to one incident with a single root cause and six affected components.



1.3 Limitations of Existing Approaches

Rule-based correlation engines depend on hand-maintained rules and break when infrastructure or failure modes change. Statistical and graph-based root cause methods can identify anomalous components but typically do not produce actionable, explained recovery plans. Plain LLM assistants, even with retrieval over runbooks, lack the structural understanding of which component connects to which, so they cannot reliably reason about propagation or blast radius, and nothing prevents them from producing a harmful recommendation.

1.4 Contributions

The contributions of this work are:

- A cross-domain telemetry normalization and correlation layer linked to a topology knowledge graph spanning compute, storage, GPU, network, and virtualization.
- A retrieval design that provides the LLM with an incident-specific blast-radius subgraph together with relevant runbooks and historical incidents.
- A deterministic constraint validator, including dry-run simulation, that gates all recovery plans and supports an LLM repair loop.
- An evaluation measuring diagnosis accuracy, time to recovery, and unsafe-action rate against rule-based, retrieval-only, and unvalidated baselines.

II. BACKGROUND AND RELATED WORK

2.1 Failures in Large-Scale Infrastructure

Empirical studies of failures in large-scale computing systems document the frequency and diversity of hardware, network, and software faults. Analysis of network failures in data centers shows that link and device failures are common and that redundancy does not always mask their effect [1]. Studies of disk failures and memory errors in production fleets show that component degradation is gradual and detectable in telemetry before outright failure [4], [5]. In accelerator clusters, large-scale analysis of GPU training workloads has found that hardware faults, interconnect issues, and straggling devices are significant causes of job failure and slowdown [3].

2.2 Root Cause Analysis and AIOps

Automated root cause analysis for services has been approached through causal graphs, anomaly propagation over dependency graphs, and trace-based localization [6]. AIOps research applies machine learning to incident triage, alert aggregation, and failure prediction, though industrial reports emphasize that practical adoption is limited by trust, explainability, and the difficulty of integrating heterogeneous data [7]. Recent work evaluates LLMs for root cause analysis of cloud incidents, finding that models can propose plausible causes from incident descriptions and logs, but that accuracy is modest without additional context and that hallucination remains a concern [8], [9].

2.3 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) conditions an LLM's output on documents retrieved from an external corpus, which improves factual grounding and allows knowledge to be updated without retraining [10]. For operations, the corpus includes runbooks, postmortems, vendor documentation, and prior incident tickets. Standard RAG retrieves unstructured text by semantic similarity, which is insufficient for infrastructure reasoning, since relevance depends on structural relationships such as which hosts share a switch rather than lexical similarity.

2.4 Knowledge Graphs and Graph-Grounded Reasoning

Knowledge graphs represent entities and relations explicitly and have been used to ground language models in structured facts. Combining graph retrieval with text retrieval allows models to reason over multi-hop relations, such as a path from a failing link to affected virtual machines. For data centers, the topology is naturally a typed graph in which nodes include racks, hosts, GPUs, switches, storage nodes, and virtual workloads, and edges encode physical connectivity, containment, and logical dependency.

2.5 Reasoning, Tool Use, and Safety of LLM Agents

Prompting methods that interleave reasoning with actions allow LLMs to query tools and observe results while diagnosing a problem [11]. Studies of LLM-based planning warn that language models do not reliably verify the feasibility of their own plans, and that external verification against constraints improves correctness [12]. This is consistent with the established engineering principle that automation affecting production systems should be bounded by deterministic safety checks rather than trusting model output [13].



2.6 Limitations of Existing Approaches

Prior work typically addresses either diagnosis or remediation, uses a single telemetry domain, or evaluates LLM suggestions without enforcing infrastructure constraints. The framework in this paper integrates cross-domain correlation, topology grounding, retrieval, plan generation, and deterministic validation in one pipeline and evaluates safety explicitly.

III. TECHNICAL FRAMEWORK AND PLATFORM DESIGN

3.1 Platform Context

The framework was developed for a cloud data center environment comprising virtualized and containerized compute clusters, GPU clusters for training and inference, a distributed storage system, and a multi-tier switching fabric. Each domain has its own monitoring and event sources. Operators respond to incidents with runbooks and have change management policies governing maintenance actions.

Topology-Aware Generative AI Architecture for Fault Diagnosis and Recovery Planning

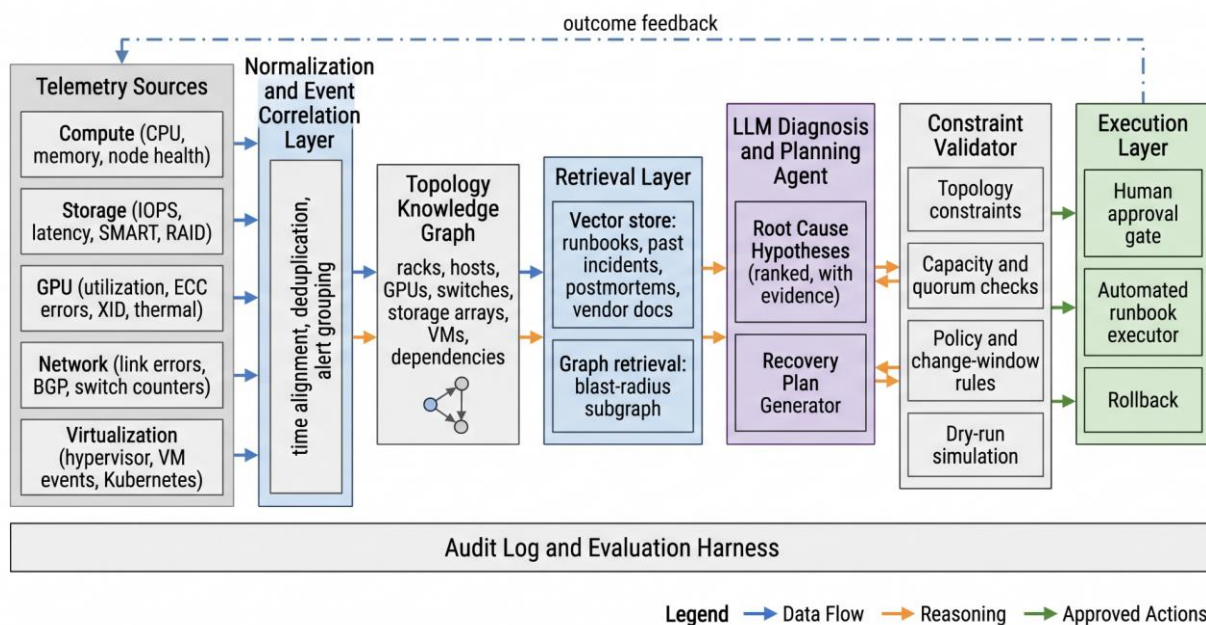


Figure 2. System architecture. Blue arrows indicate data flow, orange arrows indicate reasoning, and green arrows indicate approved actions. Outcome feedback from the execution layer returns to the telemetry layer.

The overall architecture of the framework is shown in Figure 2. Telemetry from the five infrastructure domains is normalized and correlated, then linked to the topology knowledge graph. A retrieval layer supplies the LLM with the incident-specific subgraph and relevant documents. The LLM produces ranked diagnoses and a structured recovery plan, and the constraint validator gates the plan before it reaches the execution layer.

3.2 Telemetry Normalization and Correlation

Telemetry from the five domains is ingested through a streaming pipeline and converted to a common event schema. Each event carries a timestamp, entity identifier, domain, severity, signal type, and value or message. Signals include:

- Compute: node health, CPU and memory pressure, kernel errors, machine check events.
- Storage: IOPS and latency percentiles, device error counts, SMART indicators, rebuild and resync state.
- GPU: utilization, memory ECC errors, driver error codes, thermal and power throttling, interconnect link state.
- Network: interface errors and discards, link flaps, optical power levels, routing protocol state, switch resource counters.
- Virtualization: hypervisor events, virtual machine state transitions, scheduler events, and container orchestration events.



Entities are resolved against the topology graph using identifiers such as serial numbers, hostnames, and port identifiers. Events are grouped into candidate incidents using time windows and graph proximity. Two events are linked if they occur within a time window Δt and the shortest path in the topology graph between their entities is at most k hops. Connected groups of linked events form an incident, which reduces alert volume from hundreds of raw alerts to a small number of correlated incidents.

3.3 Topology Knowledge Graph

The topology is represented as a typed directed graph $G = (V, E)$. Vertex types include datacenter, pod, rack, host, GPU, NIC, switch, port, storage node, volume, virtual machine, and workload. Edge types include contains, connects-to, mounts, depends-on, and replicates-to. Each vertex carries attributes such as health state, capacity, firmware version, tenant, and criticality. The graph is populated from configuration management and inventory systems and from discovery protocols, and is refreshed continuously so that it reflects the current infrastructure. Redundancy groups, quorum sets, and spare pools are represented explicitly, since these are the structures that recovery constraints refer to.

For each incident, the blast-radius subgraph $G_I \subseteq G$ is extracted. It includes the entities with anomalous signals, their h -hop neighborhoods, and all entities that depend on them:

$$G_I = \bigcup_{v \in A} \text{Neighborhood}_h(v) \cup \text{Dependents}(A)$$

where A is the set of anomalous entities in the incident. This subgraph, serialized in a compact text format, provides the model with the structural information needed to reason about causality and impact.

3.4 Retrieval-Augmented Diagnosis

Two retrieval sources feed the model. A vector store indexes runbooks, postmortem documents, vendor troubleshooting guides, and historical incident records, chunked and embedded for semantic search. Queries combine the textual description of anomalous signals with entity types and error codes, and results are re-ranked by a cross-encoder. Graph retrieval returns the blast-radius subgraph and, when available, historical incidents that involved structurally similar subgraphs, identified by matching entity types and relationship patterns rather than text alone.

The model is prompted with the correlated event timeline, the subgraph, and the retrieved documents, and is instructed to produce:

- A ranked list of root cause hypotheses, each specifying the suspected entity, the failure mechanism, and the evidence supporting it, with references to specific events and graph paths.
- For each hypothesis, a statement of what additional evidence would confirm or refute it.
- A confidence estimate.

Entity identifiers in the output are verified against the topology graph. Any hypothesis that cites a nonexistent entity or an entity outside the incident scope without justification is discarded, which suppresses a common class of hallucination. The model may call diagnostic tools, such as querying a switch counter or a storage health report, to test hypotheses before finalizing the ranking.

3.5 Recovery Plan Generation and Constraint Validation

For the top hypothesis, the model generates a recovery plan as a structured sequence of actions, each with a type (for example drain, cordon, restart, replace, reroute, failover, migrate, isolate), a target entity, preconditions, expected effects, and a verification check. The plan is expressed in a typed schema rather than free text, so it can be parsed and checked mechanically.

Each plan is validated by a deterministic constraint validator before any execution, in five gates:

- Topology consistency. All targets exist in the graph and fall within the diagnosed scope. Actions on entities outside scope are rejected.
- Blast radius. The set of entities affected by each action, computed by graph traversal, does not exceed configured limits on affected virtual machines, tenants, racks, and criticality levels.
- Capacity and redundancy preservation. After each action, redundancy groups retain required members, quorum sets retain a majority, spare capacity such as GPU pools meets thresholds, and network paths retain required diversity. Formally, for every redundancy group g with required minimum m_g , the number of healthy members after the action h is $g' \geq m_g$.



- Dependency order and policy. Actions respect ordering rules such as draining workloads before restarting a host, and conform to policy including change windows, tenant agreements, and required approvals for high-risk actions.
- Dry-run simulation. The plan is executed against a model of the topology to verify that the resulting state remains valid and that the incident's affected entities become healthy under the plan's assumptions.

If a gate fails, the validator returns a structured explanation of the violated constraint to the model, which revises the plan. The repair loop is limited to a small number of iterations, after which the incident is escalated to an on-call engineer with the diagnosis, the failed plan, and the violation reasons. A risk classification determines execution mode, with low-risk actions executed automatically and high-risk actions requiring explicit human approval. Executions proceed step by step with checkpoints, verification after each step, and automatic rollback when verification fails.

Recovery Plan Validation Pipeline Against Infrastructure Constraints

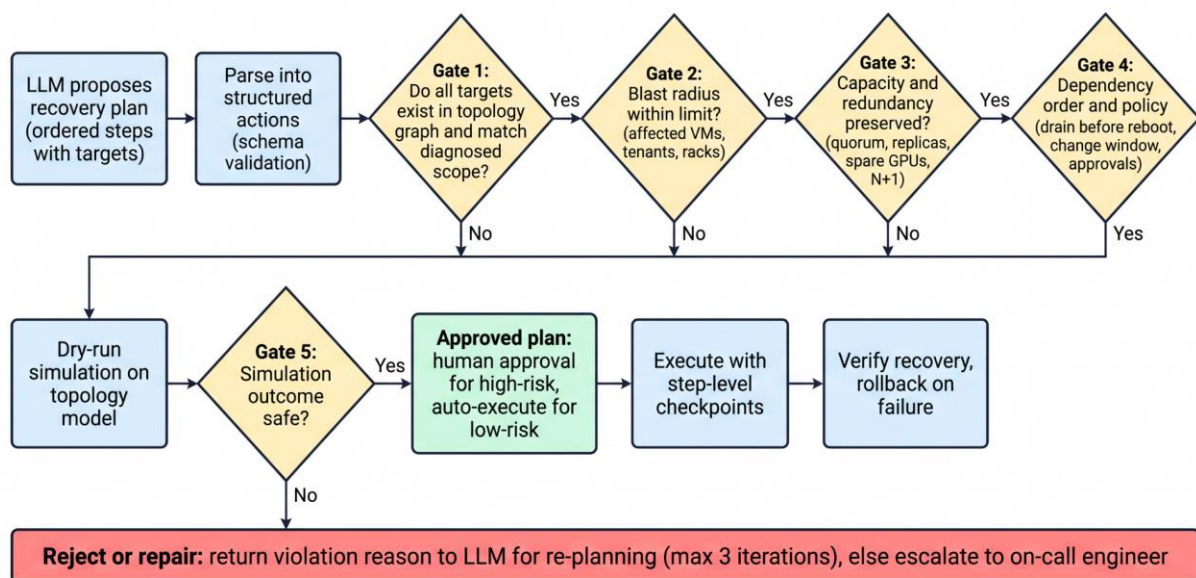


Figure 2. Recovery plan validation pipeline. Each gate has a rejection path that returns the violation to the model for repair or escalates to an engineer.

3.6 Validation and Testing Methodology

The framework was evaluated on an incident set drawn from three sources: historical incidents with postmortem-confirmed root causes, faults injected in a test environment that mirrored production topology (including link degradation, GPU errors, storage node failures, and hypervisor faults), and synthetic multi-fault scenarios generated in a topology simulator. Ground truth labels for root cause and for a set of acceptable recovery plans were established by senior engineers. Four configurations were compared:

- Rule-based correlation: An existing correlation engine with hand-written rules.
- LLM with text retrieval only: The same LLM with runbook and incident retrieval but without topology graph.
- Topology-aware LLM without validation: Full diagnosis pipeline, but plans executed without the constraint validator.
- Proposed framework: Topology-aware diagnosis with constraint validation and repair loop.

Diagnosis accuracy was measured as the fraction of incidents for which the true root cause entity and mechanism appeared at rank 1 (top-1) and within the top 3. Recovery time was measured from incident declaration to verified restoration, comparing simulated automated execution with the historical median manual response time. An action was classified as unsafe if executing it in the topology model would violate any constraint in the validator's rule set or a separate, independently authored safety checklist, including actions affecting out-of-scope entities, breaking redundancy or quorum, or violating dependency ordering. The independent checklist reduced the risk that the validator was graded by its own rules. Hallucination rate was the fraction of hypotheses or plan steps citing entities nonexistent in the topology. Alert reduction was the ratio of raw alerts to correlated incidents.



IV. DEPLOYMENT MECHANICS

4.1 Data and Graph Infrastructure

Telemetry flows through a streaming platform into a time series store and an event store. The topology graph is maintained in a graph database, synchronized from inventory and configuration management systems with change events applied incrementally. Graph freshness was monitored, because stale topology data is a primary safety risk, and the validator refused to approve plans when the graph freshness for the incident scope exceeded a threshold.

4.2 Retrieval and Model Serving

Documents were chunked with metadata, including domain, component type, and vendor, to allow filtered retrieval. The LLM was deployed within the organization's security boundary, so telemetry and incident data did not leave the environment. Prompts were templated, outputs were constrained to schemas, and model outputs were validated syntactically before further processing. Prompt contents were bounded in size by pruning the subgraph to the most relevant entities.

4.3 Human-in-the-Loop Operation

The framework operates in graduated autonomy modes. In advisory mode, it presents the diagnosis and validated plan to engineers, who execute or modify it. In supervised mode, low-risk actions execute automatically after validation while higher-risk actions require approval. Autonomy was expanded per action type only after sustained measured safety in advisory mode. Every diagnosis, plan, validation result, approval, and execution step was recorded in an audit log.

4.4 Guardrails and Failure Modes

Additional guardrails included an allowlist of action types, rate limits on actions per time window, a global kill switch, and a rule that actions on entities with active incidents of higher severity require escalation. If retrieval returned no relevant documents or the model's confidence was low, the framework withheld a plan and presented diagnostic evidence only.

Table 1. Telemetry Domains and Associated Fault Examples

Domain	Example Signals	Example Faults	Typical Cross-Layer Symptom
Compute	MCE events, memory errors, node conditions	Failing DIMM, kernel panic	Pod evictions, VM restarts
Storage	Latency percentiles, device errors, rebuild state	Degraded disk, resync storm	Application I/O stalls
GPU	ECC errors, driver error codes, thermal throttle	Uncorrectable ECC, link downtrain	Training job stall or slowdown
Network	Link errors, optical power, BGP state	Degraded optic, switch fault	Storage latency, collective timeouts
Virtualization	Hypervisor events, scheduler failures	Host agent failure	VM unavailability

V. IMPLEMENTATION OUTCOMES AND MEASURED IMPACT

5.1 Alert Reduction and Correlation

Time and topology correlation reduced raw alert volume by 85-95% in multi-domain incidents, grouping hundreds of alerts into a small number of incidents. Correlation precision, defined as the fraction of grouped events that truly belonged to the same fault, was 88-94%. Errors were mainly incidents in which two independent faults occurred close together in the topology and were merged, which the model usually separated into distinct hypotheses at the diagnosis stage.

5.2 Diagnosis Accuracy

Topology awareness had the largest effect on diagnosis quality. The proposed framework achieved a top-1 root cause accuracy of 82-88% and top-3 accuracy of 92-96%. Rule-based correlation reached 55-65% top-1 accuracy and performed worst on novel failure modes and cross-domain propagation. The LLM with text retrieval only reached 68-75% top-1, and its errors were frequently confusion between symptom and cause, where the component with the loudest alerts was named as the root cause. The topology-aware model correctly attributed storage and GPU symptoms to upstream network faults in cases where the retrieval-only model did not. Entity hallucination, defined as citing components that do not exist, fell from 6-10% for the retrieval-only configuration to below 1% after entity verification against the graph.



Table 2. Diagnosis Performance

Configuration	Top-1 Accuracy	Top-3 Accuracy	Entity Hallucination Rate
Rule-based correlation	55-65%	70-78%	Not applicable
LLM with text retrieval only	68-75%	82-88%	6-10%
Topology-aware LLM	82-88%	92-96%	Below 1%
Proposed framework (with validation)	82-88%	92-96%	Below 1%

5.3 Performance by Fault Type

Accuracy varied across fault categories. Faults that propagate across layers benefited most from topology grounding, while faults localized in one component showed smaller gains. Multi-fault incidents, in which two independent causes overlapped, were the most difficult, and accuracy for them was 10-20 percentage points lower than for single-fault incidents.

Table 3. Top-1 Accuracy by Fault Category

Fault Category	Rule-Based	Text Retrieval LLM	Proposed Framework
Single-component compute or storage fault	70-80%	78-85%	88-93%
Network fault with cross-layer symptoms	40-50%	55-65%	84-90%
GPU hardware or interconnect fault	45-55%	62-70%	80-87%
Virtualization or orchestration fault	55-65%	70-78%	82-88%
Multi-fault incidents	30-40%	45-55%	65-75%

5.4 Recovery Time

For incidents handled in supervised mode, median time to recovery decreased by 35-50% relative to the historical manual response. Most of the improvement came from faster diagnosis, since engineers began recovery with a ranked hypothesis and evidence rather than starting from raw alerts, and from reduced time spent searching runbooks. Automated execution of low-risk steps contributed additional savings. Time spent in approval for high-risk actions was a significant remaining component, which reflects a deliberate trade-off between speed and safety.

5.5 Safety of Recovery Plans

Plans produced by the unvalidated topology-aware model contained at least one unsafe action in 9-14% of incidents. Typical violations included draining a host that held the last healthy replica of a storage shard, removing a switch whose peer was already degraded, restarting a host before its workloads were drained, and choosing a target outside the incident's scope because of a naming similarity. With the constraint validator and repair loop, the unsafe-action rate in approved plans was 0-1%. In most rejected cases, the repair loop produced a valid plan within one or two iterations, and the remainder were escalated to engineers with the explanation. The few residual unsafe cases were traced to topology data that was incomplete in the model, which indicates that validator safety is bounded by the fidelity of the topology graph.



Table 4. Recovery Plan Safety and Efficiency

Metric	Unvalidated LLM Planner	Proposed Framework
Incidents with at least one unsafe action	9-14%	0-1%
Plans rejected at validation	Not applicable	11-18%
Rejected plans repaired within 3 iterations	Not applicable	70-85%
Plans escalated to engineers	Not applicable	3-6%
Median time to recovery (relative to manual)	45-60%	50-65%
Out-of-scope entity actions	3-6%	0%

5.6 Ablation of Framework Components

Removing the topology graph reduced top-1 accuracy by 10-15 percentage points and increased hallucination. Removing text retrieval reduced accuracy for known failure modes by a smaller margin, with the largest impact on vendor-specific faults whose symptoms are documented in specific troubleshooting guides. Removing the dry-run simulation did not substantially change the number of unsafe actions caught by the first four gates, but it detected a small number of ordering errors that the static checks missed. Removing the repair loop increased the number of incidents escalated to humans, since rejected plans had no chance to be corrected.

Table 5. Ablation Results

Component Removed	Effect on Top-1 Accuracy	Effect on Safety
Topology graph	10-15 points lower	Higher out-of-scope action risk
Text retrieval	4-8 points lower	No direct effect
Entity verification	Hallucination rises to 5-9%	Out-of-scope actions return
Constraint validator	No change	Unsafe rate rises to 9-14%
Dry-run simulation	No change	Small rise in ordering violations
Repair loop	No change	More escalations, slower recovery

5.7 Operational Observations

Engineers reported that cited evidence and graph paths in the diagnosis were the main factor that built trust, because they allowed quick verification. Graph freshness and completeness proved critical, and incidents involving recently changed hardware were the most common source of errors. Inventory quality, rather than model capability, was the dominant limit on reliability, which suggests that investment in topology data quality yields returns beyond this application.

VI. ADVANCED EXTENSIONS AND FUTURE DIRECTIONS

6.1 Learning from Incident Outcomes

Outcome feedback from executed plans, including whether verification succeeded and whether engineers modified the plan, can be used to refine retrieval ranking, calibrate confidence, and fine-tune models. Care is required to avoid reinforcing historical biases or rare unsafe edge cases.



6.2 Predictive Fault Detection

Degradation signals such as rising correctable error rates or increasing optical power loss precede many hardware failures [4], [5]. Extending the framework to produce proactive maintenance plans, with the same constraint validation, would shift operations from reaction to prevention.

6.3 Formal Verification of Constraints

The validator currently uses a rule set and graph traversal. Expressing redundancy and safety constraints in a formal specification language and verifying plans with a solver would provide stronger guarantees and make the constraint set easier to audit.

6.4 Limitations

The evaluation relied on a combination of historical, injected, and simulated incidents, and results may differ in other environments and failure distributions. The framework's safety depends on the accuracy and freshness of the topology graph and on the completeness of the constraint set, and constraints that are not encoded cannot be enforced. Diagnosis accuracy for multi-fault incidents and for novel failure modes remains lower than for common single-fault cases. The approach does not replace human judgment for high-risk actions, and its benefits depend on thoughtful graduated autonomy.

VII. CONCLUSION

Cross-layer faults in cloud data centers produce alert storms, ambiguous symptoms, and recovery decisions that must respect strict physical and logical constraints. Generative AI can help with the reasoning load of diagnosis, but only if its reasoning is grounded in the real infrastructure and its recommendations are checked before they affect production. The framework in this paper combines telemetry correlation across compute, storage, GPU, network, and virtualization with a topology knowledge graph, retrieval of runbooks and prior incidents, and a deterministic constraint validator with a repair loop.

The measured outcomes of higher diagnosis accuracy, shorter recovery time, and a near-zero unsafe-action rate show that LLM-assisted operations can be both useful and safe when the model is bounded by verifiable constraints. The ablation results indicate that topology grounding drives accuracy while validation drives safety, and that neither substitutes for the other. The approach can be adopted incrementally, beginning with advisory diagnosis and extending autonomy as measured safety is established. Future work includes learning from incident outcomes, predictive maintenance planning, and formal verification of recovery constraints.

VIII. AUTHOR STATEMENTS

Ethical approval: The conducted research is not related to either human or animal use.

Conflict of interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement: The authors declare that they have nobody or no company to acknowledge.

Author contributions: Manoj Kumar Reddy Kalakoti: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization, Project administration.

Funding information: The authors declare that there is no funding to be acknowledged.

Data availability statement: The validation rules, evaluation harness, and benchmark results used in this study are available upon reasonable request. Incident data and topology information have been anonymized to protect confidential infrastructure details.

REFERENCES

- [1] P. Gill, N. Jain, N. Nagappan, (2011). Understanding network failures in data centers: Measurement, analysis, and implications, in: Proceedings of the ACM SIGCOMM 2011 Conference, ACM, Toronto, Canada, 350-361.
- [2] H.S. Gunawi, M. Hao, R.O. Suminto, A. Laksono, A.D. Satria, J. Adityatama, K.J. Eliazar, (2016). Why does the cloud stop computing? Lessons from hundreds of service outages, in: Proceedings of the Seventh ACM Symposium on Cloud Computing (SoCC), ACM, Santa Clara, CA, 1-16.



- [3] M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, F. Yang, (2019). Analysis of large-scale multi-tenant GPU clusters for DNN training workloads, in: Proceedings of the 2019 USENIX Annual Technical Conference (ATC), USENIX, Renton, WA, 947-960.
- [4] B. Schroeder, G.A. Gibson, (2007). Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?, in: Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST), USENIX, San Jose, CA, 1-16.
- [5] B. Schroeder, E. Pinheiro, W.-D. Weber, (2009). DRAM errors in the wild: A large-scale field study, in: Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), ACM, Seattle, WA, 193-204.
- [6] M. Ma, W. Lin, D. Pan, P. Wang, (2020). Self-adaptive root cause diagnosis for large-scale microservice architecture, IEEE Trans. Serv. Comput. 15 (3) 1399-1410.
- [7] Y. Dang, Q. Lin, P. Huang, (2019). AIOps: Real-world challenges and research innovations, in: Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), IEEE, Montreal, Canada, 4-5.
- [8] T. Ahmed, S. Ghosh, C. Bansal, T. Zimmermann, X. Zhang, S. Rajmohan, (2023). Recommending root-cause and mitigation steps for cloud incidents using large language models, in: Proceedings of the 45th International Conference on Software Engineering (ICSE), IEEE, Melbourne, Australia, 1737-1749.
- [9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, et al., (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks, in: Advances in Neural Information Processing Systems 33 (NeurIPS), Curran Associates, Red Hook, NY, 9459-9474.
- [10] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, (2023). ReAct: Synergizing reasoning and acting in language models, in: The Eleventh International Conference on Learning Representations (ICLR), Kigali, Rwanda.
- [11] K. Valmeekam, M. Marquez, A. Olmo, S. Sreedharan, S. Kambhampati, (2023). PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change, in: Advances in Neural Information Processing Systems 36 (NeurIPS), Curran Associates, Red Hook, NY.
- [12] B. Beyer, C. Jones, J. Petoff, N.R. Murphy, (2016). Site Reliability Engineering: How Google Runs Production Systems, O'Reilly Media, Sebastopol, CA.
- [13] B. Beyer, N.R. Murphy, D.K. Rensin, K. Kawahara, S. Thorne, (2018). The Site Reliability Workbook: Practical Ways to Implement SRE, O'Reilly Media, Sebastopol, CA.