An Empirical Study of Automation Gaps in LLM Serving Systems

Bhala Ranganathan
Microsoft
Redmond, WA, USA

Minghua Ma
Microsoft
Redmond, WA, USA

Mickey Zhang
Microsoft
Redmond, WA, USA

Klein Hu
Microsoft
Redmond, WA, USA

Rakesh Kelkar
Microsoft
Redmond, WA, USA

Chetan Bansal
Microsoft
Redmond, WA, USA

ABSTRACT

Hyperscale large language model (LLM) inference systems require high reliability, yet incident mitigation in practice remains largely manual. We present an empirical study of 156 high-severity production incidents from a large-scale LLM serving platform, analyzed using a validated four-way taxonomy that captures failure causes, detection, and mitigation actions. Our analysis identifies a persistent automation gap: while 74% of incidents are automatically detected, most are mitigated manually, leading to time-to-mitigate (TTM) values of up to 49 hours. Using a taxonomy-driven, multi-dimensional analysis, we systematically localize automation bottlenecks and derive prescriptive self-healing opportunities aligned with dominant root causes. We empirically evaluate selected taxonomy-prescribed mitigations deployed in production. One representative strategy, automated connection liveness recovery, reduced timeout-related errors by 99.6% and improved service-level agreement (SLA) compliance. These results demonstrate that structured incident taxonomies can support evidence-based identification and validation of automation opportunities, enabling measurable reliability improvements in LLM serving systems with reduced human intervention.

1 INTRODUCTION

The rapid deployment of large language models (LLMs) in production systems [7, 12, 18] has fundamentally changed the operational profile of cloud services. Unlike traditional stateless microservices, LLM serving pipelines couple latency-critical inference, dynamic batching, accelerator scheduling, and complex dependency chains. As a result, a single partial failure, such as a stalled connection pool or degraded accelerator node, can cascade into widespread request timeouts and user-visible outages within minutes. In practice, even short-lived incidents in LLM inference services have been observed to affect millions of requests and require coordinated intervention across multiple engineering teams.

Despite this increased operational complexity, the mitigation workflows used to recover LLM serving systems remain largely inherited from conventional cloud services. Existing AIOps and SRE tooling has made detection increasingly automated, but recovery actions frequently depend on manual diagnosis and ad hoc decision-making, especially under peak load. This mismatch raises a fundamental question: *why does mitigation automation lag behind detection in modern LLM serving systems, and where exactly do the bottlenecks occur?* Answering this question requires empirical evidence grounded in production incident data, rather than intuition or isolated failure anecdotes.

Prior empirical studies of cloud incidents have provided valuable insights into failure causes, cascading behaviors, and postmortem practices. However, these studies largely focus on general-purpose cloud infrastructure, storage systems, or microservice architectures. LLM serving differs in several important respects. First, inference workloads exhibit bursty, user-driven traffic with tight tail-latency constraints. Second, mitigation actions, such as model-specific hotfixes, connection liveness resets, or capacity rebalancing across accelerators, are often high-impact and risky to automate without structured guidance. Third, growth dynamics are unusually rapid: production systems may experience order-of-magnitude traffic changes within weeks, invalidating mitigation heuristics derived from stable workloads. These differences limit the applicability of prior cloud incident analyses to LLM serving and motivate a dedicated empirical study.

In this paper, we argue that improving resilience in LLM serving systems requires shifting the focus of incident analysis from failure detection to mitigation execution. Specifically, understanding why automation stalls at the mitigation stage demands a structured view of the incident lifecycle that links root causes, mitigation actions, and human effort. Without such structure, organizations risk over-investing in monitoring while leaving the most time-consuming and error-prone recovery steps manual.

To this end, we present an empirical study of 156 high-severity production incidents from a large-scale LLM serving platform. Our study spans two complementary phases. A development and validation phase (February–May 2025) is used to construct and validate a taxonomy-driven analysis framework. A subsequent high-growth phase (April–June 2025), during which the system experienced a 1.27× increase in token generation load [19], serves as a natural stress test for operational workflows. Studying incidents during this growth period allows us to observe mitigation behavior under realistic hyperscale pressure, where automation gaps have the highest impact.

Our empirical findings expose a critical automation paradox. Although 74% of high-severity incidents are automatically detected by existing monitoring pipelines, the majority of impactful mitigations remain manual. High-latency actions such as hotfix deployments (28.21%) and capacity increases (9.62%) frequently require human judgment, causing time-to-mitigate (TTM) to increase to as much as 49 hours during peak demand. These results show that detection-centric automation alone is insufficient for LLM serving resilience and motivate the need for decision-logic frameworks that can safely guide mitigation automation.

This paper makes the following contributions to the study of reliability in production LLM serving systems:

- **Empirical incident study of LLM serving at hyperscale.** We present an empirical analysis of 156 high-severity production incidents from a large-scale, customer-facing LLM serving platform, spanning periods of rapid workload growth. To our knowledge, this is one of the first studies to quantitatively characterize mitigation behavior and automation gaps specifically in LLM inference systems.
- **A validated taxonomy for mitigation-centric incident analysis.** We introduce and validate a four-way incident taxonomy that captures failure modes, detection mechanisms, and mitigation actions, enabling systematic localization of automation bottlenecks across the incident lifecycle. The taxonomy is platform-agnostic and designed to support reproducible analysis across LLM and non-LLM cloud systems.
- **Quantification of the mitigation automation gap.** Applying the taxonomy, we empirically demonstrate an automation paradox in LLM serving: although 74% of high-severity incidents are automatically detected, the majority require high-latency manual mitigation, with time-to-mitigate reaching up to 49 hours during peak load.
- **Evidence-based evaluation of self-healing strategies.** We show that taxonomy-prescribed mitigation strategies can be safely automated and yield measurable reliability improvements. In production, automated connection liveness recovery reduced timeout-related errors by 99.6% and improved service-level agreement (SLA) compliance.

2 BACKGROUND AND RELATED WORK

2.1 System architecture overview

At a high level, our LLM serving system similar to [20] is composed of control plane and data plane components. The control plane manages configurations across different layers, including model-specific settings, endpoint traffic routing, load balancing policies, and engine deployment parameters. The data plane provides components for scheduling and serving inference requests, enabling flexible and high performance model execution. AI models are deployed as containerized services on GPU clusters managed by Kubernetes, with both off-the-shelf and custom variants in production. Inference engine loads the model weights and executes requests. This cloud-scale inference platform analyzed in this study serves a diverse set of production AI models, primarily LLMs for text generation and classification (e.g., Llama-2 [18]), vision and speech models (e.g., Llama-3 [7]) and reasoning models (e.g., models targeting multi-step reasoning benchmarks such as BIG-bench [17]). Key system characteristics include high concurrency, variable prompt lengths, and support for streaming inference, which collectively drive unique reliability challenges compared to traditional cloud workloads.

2.2 Cloud services reliability

Prior work on cloud reliability has traditionally emphasized redundancy, failover mechanisms, and reactive monitoring, with more recent studies moving toward automation and AI-driven operations. Ghosh et al. [6] present a multi-dimensional analysis of troubleshooting stages, correlating detection, diagnosis, and mitigation to guide automation at different granularities. Subsequent research has explored predictive and proactive reliability techniques, including SLO-driven control loops and intelligent monitoring. Srinivas et al. [16] propose a deep learning framework for recommending monitors based on service properties, while Kokolis et al. [9] analyze large-scale ML workloads over 11 months, introducing failure taxonomies and reliability metrics that highlight the need for workload-aware infrastructure. In parallel, intelligent incident management systems have incorporated AI for diagnosis and triage, such as DeCaf [1], IcM BRAIN [3], ICA [15], and recent multi-agent LLM-based systems like Triangle [21]. Public outage analyses further study cross-provider incident trends [4]. However, existing work primarily focuses on detection, diagnosis, or general ML workloads, and does not empirically quantify mitigation automation gaps in customer-facing LLM serving systems or evaluate taxonomy-driven self-healing strategies in production. Our study addresses this gap through a mitigation-centric empirical analysis grounded in LLM serving incidents.

3 ROOT CAUSE TAXONOMY FOR LLM SERVING SYSTEM

The taxonomy and rulebook Table 1 was derived from a year of operational learnings using the methodology described in this section. Our methodology combines quantitative and qualitative approaches to capture the full incident lifecycle. While the taxonomy was applied to drive reliability gains starting in February 2025, we focused our quantitative gap analysis on the April→June window to observe system performance under a 1.27x growth in token generation load. This high growth period serves as a stress test for current operational workflows ensuring the data is representative of LLM serving hyperscale challenges.

3.1 Production incident data collection

While automation can streamline data processing, we manually curated data from multiple sources to ensure rigor and accuracy in all correlation and association analysis. For each day in the study period, we recorded incident details and other relevant variables based on operational logs and deployment schedule. Data entry was cross checked with engineers to ensure accuracy. We utilized the Incident Management System (ICM) to extract metadata for all high severity incidents, including timestamps, impacted endpoints, and resolution notes,. Detailed narratives of root causes, mitigation steps, and follow up actions were sourced from postmortem reports, while runtime logs and dashboards provided essential telemetry on engine health, latency, and error rates. Finally, Designated Responsible Individual (DRI) playbooks and tools were used to contextualize the operational workflows and automation strategies employed during incident response.

3.2 Root cause categorization

Each incident was manually reviewed and assigned to one of the four root cause categories (Table 1). In cases where an incident involves multiple contributing factors, we resolve classification ambiguity by selecting the one that appears earliest in the incident summary. For example, if engine resource exhaustion is caused by faulty compute nodes, we classify it as an infrastructure failure; however, if the compute nodes are healthy but the engine is overloaded, we classify it as an inference engine failure. This taxonomy was iteratively refined through cross validation with incident reviewers and inference serving team engineers. To capture the nuances of the incident lifecycle, we followed the six operational dimensions that influence the effectiveness of incident management [6]. These dimensions span detection latency, escalation clarity, tooling support, mitigation complexity, documentation quality, and cross-team coordination. Together, they provide a structured lens through which to evaluate incident response workflows and identify opportunities for automation, process refinement, and knowledge sharing. This taxonomy serves as the decision-logic layer for LLM serving resilience. By mapping incident signatures to specific categories, we aim to eliminate manual toil.

3.3 Labeling procedure and inter-rater reliability

Two raters independently labeled a pilot subset (≈20%) using the rulebook covering the four root cause categories. Disagreements were reconciled and decision rules were refined. Then, five engineers independently labeled ≈31 incidents each. Subsequently, to assess labeling reliability, we selected 31 incidents (random ≈6 per rater) and independently assigned labels using our rulebook. Agreement between the initial and reassigned labels was ≈90%. Cohen's kappa [5], which adjusts for chance agreement, was 0.89 (95% CI [0.75, 1.00]), indicating almost perfect consistency in the application of labeling rules. Remaining disagreements were resolved by consensus, and the adjudicated labels were used for all analysis.

Table 1: Taxonomy of incidents

Category	Rulebook scenarios
Infrastructure failures	Compute capacity constraints, delayed compute allocation, misconfigured deployment templates, bad nodes, redis [14], MPI ring [11], sidecar, istio [8], node scheduling and storage.
Model configuration failures	Invalid weights, missing or misconfigured headers and invalid output.
Inference engine failures	Resource exhaustion, inference timeouts, latency spikes, KV cache [13] errors, crashes, input fetch, constraint sampling, memory leaks.
Operational failures	Delayed detection due to low-traffic, misconfigured alerts, or missing telemetry, error code translation, imbalanced traffic routing and rollout issues.

4 THE MITIGATION GAP

4.1 Failure mode distribution

We apply the taxonomy (Table 1) to the 156 high-severity incidents and quantify outcomes in Table 2. Categorizing incidents via the taxonomy revealed inference engine failures as the most prevalent root cause (≈60% of incidents; Table 2 A3), with timeouts and resource exhaustion accounting for the majority, thereby demonstrating the taxonomy's operational efficacy. These failures are driven by the high concurrency and variable prompt lengths unique to hyperscale LLM serving. To highlight repeat effectiveness, in section 5 we also show how the taxonomy enabled systematic identification and evaluation of targeted mitigations during the development phase. These mitigations correlated with measurable reductions in HTTP 408 rates and improved Service Level Aggrement (SLA). The method's granularity thus directly supported engine improvements and provides a template for ongoing engine level resilience work.

Table 2: Incident analysis by root cause category (Apr→Jun 2025)

ID	Category	Total n/N	% of all	Breakdown (within category)
A1	**Infrastructure failures**	31/156	≈20%	Misconfigured deployment templates: ≈26%; Bad nodes: ≈19%; MPI [11] ring: ≈18%; Sidecar: ≈11%; Istio [8]: ≈7%; Delayed compute: ≈7%; Redis [14]: ≈4%; Storage: ≈4%; Scheduling: ≈4%.
A2	**Model configuration failures**	25/156	≈16%	Header misconfig/omission: ≈81%; Invalid outputs due to misconfig: ≈19%.
A3	**Inference engine failures**	94/156	≈60%	Timeouts: ≈40%; Resource exhaustion: ≈29%; Crashes: ≈12%; Latency spikes: ≈6%; Input fetch: ≈6%; KV cache [13]: ≈5%; Constraint sampling: ≈1%; Memory leaks: ≈1%.
A4	**Operational failures**	6/156	≈4%	During rollout windows: ≈33%; low-traffic: ≈17%; Misconfigured alerts: ≈17%; Error-code translation: ≈17%; Imbalanced traffic routing: ≈16%.

4.2 The automation paradox: detection vs mitigation

Standard LLM assisted DRI tooling such as incident analyzers, copilots and incident enrichment pipelines improved triage and reduced manual toil, but did not eliminate the need for human decisions during mitigations in most cases. This is most evident in the Table 3 M5 category, which despite involving no technical change needed human judgment to confirm that no action was needed. We define automation-detected as alerts fired by health probes/DRI pipelines before customer reporting and as observed in Figure 1, across the 156 incidents, 115 were automation-detected (≈74%), and the rest were still not automation-detected indicating automation gaps. These set of incidents were mitigated using one of the actions listed in Table 3 which summarizes mitigation actions (M1–M6), their shares, usage and automation. It is noted that majority of the incidents were manually mitigated.

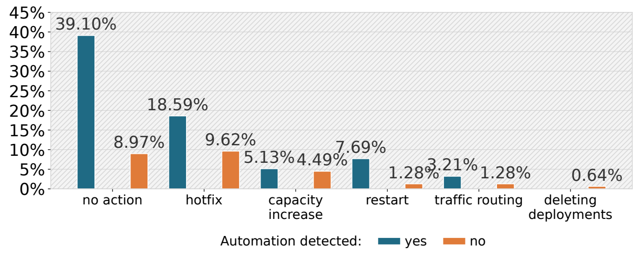

Figure 1: Automatic detection status by mitigation action

Table 3: Mitigation actions, automation level, and observed operational effects (Apr→Jun 2025)

ID	Mitigation action	Applied how	Total n/N	% of all[6]
Impactful but high latency				
M1	**Hotfix**[1]	Manual (code/config change)	44/156	≈28.21%
M2	**Capacity increase**[2]	Manual (scale out)	15/156	≈9.62%
Rapid recovery actions				
M3	**Traffic routing (intelligent failover)**[3]	Partial automation (policy + health probes)	7/156	≈4.49%
M4	**Restart (node rebalancing)**[4]	Partial automation (infra playbooks)	14/156	≈8.97%
Manual intervention bottlenecks				
M5	**Manual triage/monitor (No action).**[5]	Manual decision	75/156	≈48.08%
M6	**Deleting deployments**	Manual decision	1/156	≈0.64%

[1] Contributes to high TTM (Figure 2).
[2] Reduced overload/timeout symptoms.
[3] Shortens impact window; reroutes to healthy endpoints/regions.
[4] Auto node restart [2, 10]. Faster recovery for bad nodes/MPI ring events.
[5] Used when transient or self-healing; still need manual verification.
[6] All percentages sum to 100%; Mitigation actions are coded as mutually exclusive per incident using the first effective action rule. See Figure 1.

4.3 Multi-dimensional analysis

Figure 3 provides an empirical cross-tabulation of root cause categories (A1–A4) by mitigation actions (M1–M6), using the same labeling as Table 2 and Table 3. This visualization shows that capacity increase (M2), traffic routing (M3) and restart (M4) are predominantly applied to mitigate inference engine failures (A3), while deleting deployments (M6) is exclusive to model configuration failures (A2). The figure also clarifies the distribution of hotfixes (M1) and monitor-only actions (M5) across all root cause categories. Such analysis helps us derive targeted self-healing strategies that are most impactful to address.

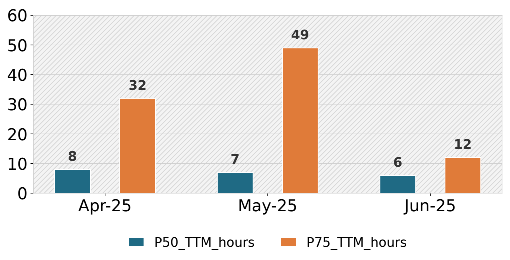

Figure 2: Time to Mitigate by month during high growth window

4.4 Identifying bottlenecks in the incident lifecycle

To evaluate the effectiveness of incident response and identify systemic inefficiencies, we analyze P50 and P75 for the incident lifecycle, Detection (TTD), Diagnosis (TTE), and Mitigation (TTM); we report Postmortem insights qualitatively. Focusing on the 50th and 75th percentiles surfaces central tendency and early degradation signals that are more actionable in day-to-day operations than extreme outliers. In April 2025, P75 TTM was ≈ 32 hours; in May it spiked to ≈ 49 hours (Figure 2). Over the same window, token generation increased by ≈ 1.27× (April⟶June). Manual mitigations especially hotfixes (M1) and transient errors (M5) contribute to high TTM. While capacity is also driven by compute availability, in contrast, throughput oriented mitigations (Table 3: M3, M4) have high potential to reduce TTM. Also, M5 incidents constitute a gap where human verification of transient errors is required to confirm that no action is needed and M6 corresponds to bad deployments that were deleted. We therefore classify the mitigation actions into three groups and interpret M1, M2 in Table 3 as "Impactful but high latency", M3, M4 as "Rapid recovery actions" and the remaining M5, M6 as "Manual intervention bottlenecks".

5 IMPACT OF TAXONOMY GUIDED MITIGATIONS

5.1 Quantitative reliability gains

The application of our four-way taxonomy directly transforms incident response from reactive patching to prescriptive self-healing, addressing the critical paradox where detection outpaces mitigation. Firstly, we look back to the validation phase (February⟶May 2025). This allows us to demonstrate that the connection liveness strategy once codified by the taxonomy successfully reduced the normalized HTTP 408 rate (calculated as the percentage of 408 errors divided by total requests) from 2.72% to 0.01% (Figure 4) for the selected impacted customer prior to the high-load period analyzed in section 4. This mechanism specifically addresses engine-level timeouts during streaming inference by keeping the long-running sessions alive. This suggests a directional association between the connection liveness mitigation and the reduction in timeout errors. Also to clearly illustrate the trend, we present normalized data from one customer, although similar improvements were observed across others where the mitigation was rolled out.

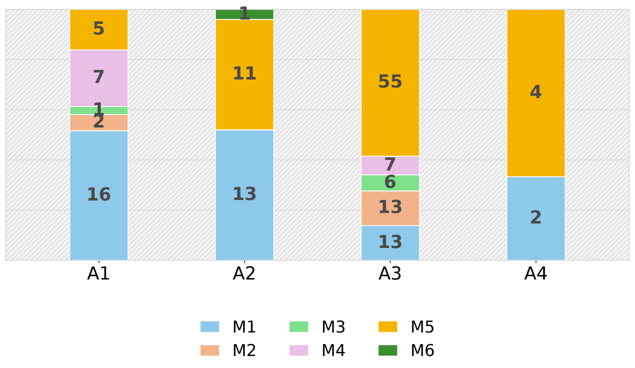

Figure 3: Count of mitigation actions (M1-M6) by root cause categories (A1-A4); Column totals equal Table 2 (A1=31, A2=25, A3=94, A4=6); row totals equal Table 3 (M1=44, M2=15, M3=7, M4=14, M5=75, M6=1)

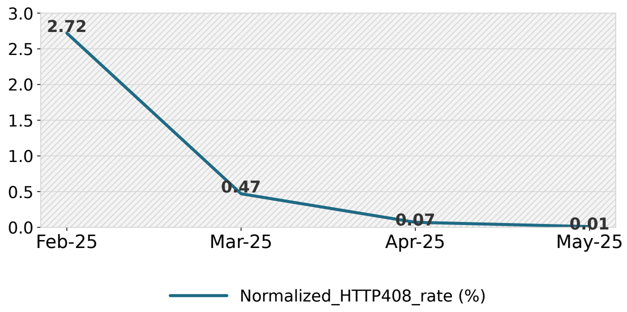

Figure 4: Normalized HTTP 408 rate by month during validation phase (February⟶May 2025)

Secondly, while initial rollouts began in February 2025 to address early timeout signals, our subsequent 3-month study (April⟶June) confirms that systematizing and automating these mitigation actions (subsection 5.2) provides a repeatable path to maintaining a 0.01% error rate even as load increased 1.27x. Traffic routing (M3), another mitigation, reduced impact windows by shifting traffic off degraded endpoints. We observed that failover to a different region or endpoint resulted in better SLA (Figure 5). SLA is calculated as the percentage of successful requests over total valid requests, excluding client-side issues. These actions are especially effective when combined with accurate health probes and routing policies.

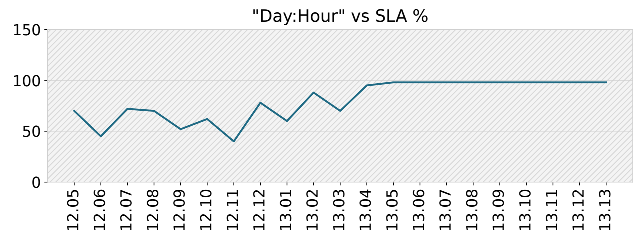

Figure 5: SLA improvement after failover to different endpoint during validation phase (February⟶May 2025)

To ensure these gains were strictly derived from taxonomy prescribed actions rather than concurrent engineering progress, we implemented rigorous attribution bias mitigation. We enforced a "first effective action rule," coding mitigations as mutually exclusive and attributing success only to the intervention that initially restored the SLA. To isolate the signal of specific mitigations like connection liveness, we analyzed customer-specific normalized telemetry Figure 4 and excluded all incidents that overlapped with maintenance windows, major deployment freezes, or concurrent capacity expansions (M2) within a 24hr window. This methodological rigor ensures that the measured error reduction was solely attributable to the identified engine-level interventions and shows that our taxonomy prescribed mitigations were the primary drivers of reliability gains.

5.2 Towards self healing framework

Algorithm 1 codifies the successes observed in the Feb⟶May window to resolve the manual bottlenecks identified during the Apr⟶Jun window. Such codified output of the learnings is intended for integration into future AIOps agents that trigger the defined mitigation actions thereby offloading the manual toil that currently consumes significant engineering cycles. The quantified engine-level actions act as primary candidates for automated self-healing to bridge the mitigation gaps. Further, to achieve the vision of fully autonomous self-healing with minimal human intervention, future research must prioritize bridging the current automation gap, where 74% of incidents are auto-detected but mitigation remains largely manual. One such prioritization is the gap represented by M5 incidents i.e. autonomously determining when no action is required without human intervention.

Algorithm 1 Taxonomy triggered mitigation

```
Require: Incident alert $I$ (auto detected via health probes), Taxonomy
    $T = \{A_1, A_2, A_3, A_4\}$
Ensure: Executed mitigation action $M \in \{M_1, \dots, M_6\}$ and SLA verification
 1: Step 1: Classification via the decision-logic layer
 2: $Category \leftarrow$ Map Incident $I$ to $A_n \in T$ using rulebook Table 1
 3: Step 2: Illustrated effective mitigation towards self healing
 4: if $Category == A_3$ (Inference engine failure) then
 5:     if $I$ signature matches Timeouts then
 6:         $M \leftarrow$ Apply connection liveness template (See subsection 5.1)
 7:     else if $I$ signature matches Resource exhaustion then
 8:         $M \leftarrow$ Trigger $M_4$ (Restart/node rebalancing) or $M_2$ (Capacity)
 9:     else if $I$ signature matches Latency spikes then
10:         $M \leftarrow$ Trigger $M_3$ (Intelligent Traffic Routing)
11:     else if $I$ signature matches Transient then
12:         $M \leftarrow$ Trigger AIOps agent for manual triage verification $M_5$
13:     else
14:         $M \leftarrow$ Escalate for $M_1$ hotfix
15:     end if
16: end if
17: Step 3: Execution and verification
18: Execute action $M$
19: $SLA_{status} \leftarrow$ Verify success rate and TTM
20: if $SLA_{status}$ == False then
21:     $M \leftarrow$ Escalate for $M_1$ hotfix
22: end if
23: return $SLA_{status}, TTM$
```

5.3 Threats to validity

Although manual incident classification can be subjective, we ensured statistical rigor through inter-rater reliability. The three-month study window, while limited, captures representative high-load LLM serving behavior. To reduce attribution bias, we excluded maintenance windows and verified no overlapping mitigation actions. Finally, while our findings are based on a specific stack, the taxonomy is applicable to any LLM serving system.

6 CONCLUSION

In this paper, we present an empirical study of 156 high-severity incidents in a hyperscale LLM serving system, highlighting the gap between automatic detection and largely manual mitigation. Using a validated four-way incident taxonomy, we systematically correlate root causes with mitigation actions to identify automation bottlenecks. Our results show that taxonomy-driven recovery strategies can substantially reduce time-to-mitigate and improve system resilience. This study provides an evidence-based foundation for transitioning LLM serving systems toward autonomous, minimal-intervention operation.

REFERENCES

[1] Chetan Bansal, Sundararajan Renganathan, Ashima Asudani, Olivier Midy, and Mathru Janakiraman. 2020. DeCaf: diagnosing and triaging performance issues in large-scale cloud services. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice* (Seoul, South Korea) *(ICSE-SEIP '20)*. Association for Computing Machinery, New York, NY, USA, 201–210. doi:10.1145/3377813.3381353

[2] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. 2016. Borg, Omega, and Kubernetes. *Commun. ACM* 59, 5 (April 2016), 50–57. doi:10.1145/2890784

[3] Zhuangbin Chen, Yu Kang, Liqun Li, Xu Zhang, Hongyu Zhang, Hui Xu, Yangfan Zhou, Li Yang, et al. 2020. Towards intelligent incident management: why we need it and how we make it. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 1487–1497. doi:10.1145/3368089.3417055

[4] Xiaoyu Chu, Sacheendra Talluri, Qingxian Lu, and Alexandru Iosup. 2025. An Empirical Characterization of Outages and Incidents in Public Services for Large Language Models. In *Proceedings of the 16th ACM/SPEC International Conference on Performance Engineering (ICPE '25)*. ACM, 69–80. doi:10.1145/3676151.3719372

[5] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46. doi:10.1177/001316446002000104

[6] Supriyo Ghosh, Manish Shetty, Chetan Bansal, and Suman Nath. 2022. How to fight production incidents? an empirical study on a large-scale cloud service. In *Proceedings of the 13th Symposium on Cloud Computing* (San Francisco, California) *(SoCC '22)*. Association for Computing Machinery, New York, NY, USA, 126–141. doi:10.1145/3542929.3563482

[7] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and Others. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[8] Istio. 2025. Istio. https://istio.io Accessed: 2025.

[9] Apostolos Kokolis, Michael Kuchnik, John Hoffman, Adithya Kumar, Parth Malani, Faye Ma, Zachary DeVito, Shubho Sengupta, Kalyan Saladi, and Carole-Jean Wu. 2025. Revisiting Reliability in Large-Scale Machine Learning Research Clusters . In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE Computer Society, Los Alamitos, CA, USA, 1259–1274. doi:10.1109/HPCA61900.2025.00096

[10] Kubernetes. 2025. kubectl commands. https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands Accessed: 2025.

[11] Message Passing Interface Forum. 2023. *MPI: A Message-Passing Interface Standard Version 4.1*. https://www.mpi-forum.org/docs/mpi-4.1/mpi41-report.pdf

[12] Meta. 2025. Open source LLM. https://www.llama.com/ Accessed: 2025.

[13] Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. 2025. Mooncake: Trading More Storage for Less Computation — A KVCache-centric Architecture for Serving LLM Chatbot. In *23rd USENIX Conference on File and Storage Technologies (FAST 25)*. USENIX Association, Santa Clara, CA, 155–170. https://www.usenix.org/conference/fast25/presentation/qin

[14] redis. 2025. Redis: An open-source, in-memory data structure store. https://github.com/redis/redis Accessed: 2025.

[15] Amrita Saha and Steven C. H. Hoi. 2022. Mining root cause knowledge from cloud service incident investigations for AIOps. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice* (Pittsburgh, Pennsylvania) *(ICSE-SEIP '22)*. Association for Computing Machinery, New York, NY, USA, 197–206. doi:10.1145/3510457.3513030

[16] Pooja Srinivas, Fiza Husain, Anjaly Parayil, Ayush Choure, Chetan Bansal, and Saravan Rajmohan. 2024. Intelligent Monitoring Framework for Cloud Services: A Data-Driven Approach. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice* (Lisbon, Portugal) *(ICSE-SEIP '24)*. Association for Computing Machinery, New York, NY, USA, 381–391. doi:10.1145/3639477.3639753

[17] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, et al. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research* (2023). https://openreview.net/forum?id=uyTL5Bvosj Featured Certification.

[18] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL] https://arxiv.org/abs/2307.09288

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[20] vllm. 2025. Easy, fast, and cheap LLM serving for everyone. https://github.com/vllm-project/vllm Accessed: 2025.

[21] Zhaoyang Yu, Aoyang Fang, Minghua Ma, Chaoyun Zhang, Ze Li, Murali Chintalapati, Xuchao Zhang, Rujia Wang, Chetan Bansal, Saravan Rajmohan, and Qingwei Lin. 2025. Triangle: Empowering Incident Triage with Multi-Agent. In *ASE'25*. https://www.microsoft.com/en-us/research/publication/triangle-empowering-incident-triage-with-multi-agents/