An Empirical Study of Production Incidents in Generative AI Cloud Services

Haoran Yan*
Huazhong University of Science and Technology
China

Yinfang Chen*
University of Illinois Urbana-Champaign
USA

Minghua Ma†
Microsoft
USA

Ming Wen
Huazhong University of Science and Technology
China

Shan Lu
Microsoft Research
USA

Shenglin Zhang
Nankai University
China

Tianyin Xu
University of Illinois Urbana-Champaign
USA

Rujia Wang
Microsoft
USA

Chetan Bansal
Microsoft
USA

Saravan Rajmohan
Microsoft
USA

Chaoyun Zhang
Microsoft
China

Dongmei Zhang
Microsoft
China

Abstract

The ever-increasing demand for generative artificial intelligence (GenAI) has motivated cloud-based GenAI services such as Azure OpenAI Service and Amazon Bedrock. Like any large-scale cloud service, failures are inevitable in cloud-based GenAI services, resulting in user dissatisfaction and significant monetary losses. However, GenAI cloud services, featured by their massive parameter scales, hardware demands, and usage patterns, present unique challenges, including generated content quality issues and privacy concerns, compared to traditional cloud services. To understand the production reliability of GenAI cloud services, we analyzed production incidents from a leading GenAI cloud service provider spanning in the past four years. Our study (1) presents the general characteristics of GenAI cloud service incidents at different stages of the incident life cycle; (2) identifies the symptoms and impacts of these incidents on GenAI cloud service quality and availability; (3) uncovers why these incidents occurred and how they were resolved; (4) discusses open research challenges in terms of incident detection, triage, and mitigation, and sheds light on potential solutions.

1 Introduction

In recent years, there have been significant advancements in generative artificial intelligence (GenAI), particularly in Large Language Models (LLMs) and their applications across various fields. Beyond natural language processing, these models have also shown new capabilities in image recognition [22, 49], data analysis[39, 57], software engineering [10, 27, 28], and more [35, 41, 53]. The emergence of models like the GPT-4 family marks a new era, with capabilities extending to complex reasoning [42, 52], creative thinking [33, 64], and even surpassing human expertise in certain tasks [6]. This innovation has resulted in impactful research findings and practical applications with substantial implications for scientific research and socio-economic development.

The demands of GenAI come with the requirements of unprecedented computational resources, including the hardware for operating the models as well as infrastructure systems for efficiently allocating and utilizing such resources [37]. However, both the acquisition of the required resources and their efficient management pose significant challenges to individuals and even enterprises. Therefore, it motivates the development of *GenAI cloud services*, which offer a platform where developers and users can create, deploy, and utilize large models without substantial hardware and software investments, e.g., Cloud for AI (Cloud4AI), and also incorporate model APIs within cloud systems, e.g., AI for Cloud (AI4Cloud) [10, 27, 28]. Popular GenAI cloud services include *Azure OpenAI*, Amazon Bedrock, IBM Watson, and Anthropic Claude. GenAI cloud services afford enterprises the infrastructure necessary for the deployment and maintenance of GenAI models, based on which users can further interact, analyze, and fine-tune such models. Moreover, they are crucial in promoting collaboration among researchers by providing shared access to advanced models and computational resources.

As with any large-scale cloud services, GenAI cloud services are not immune to occasional incidents. These events, while often unavoidable due to the complexity and scale of the systems involved, have the potential to impact user experience and, in some cases, result in challenges such as user dissatisfaction or economic implications. For example, OpenAI recently experienced an incident where request failures and high latency severely impacted ChatGPT's API and functionalities [1]. However, despite the critical importance of reliability in GenAI cloud services, there is a notable lack of research focusing on their reliability and incident

*Haoran Yan and Yinfang Chen contributed equally. Work was performed during their internship at Microsoft.

†Corresponding author. Email: minghuama@microsoft.com

management. Therefore, understanding the characteristics of these incidents—including detection, triage, diagnosis, and mitigation—is crucial for enhancing the quality of GenAI cloud services.

Before the era of GenAI cloud services, traditional ML platforms like AzureML, AWS SageMaker, and Google Cloud ML were primarily used for tasks such as training, inference, and model fine-tuning [48]. These services have been well-studied for issues like deployment challenges, fault taxonomy, and bug characteristics [12, 24, 25], while extensive research has similarly examined incident management practices, root causes, and triage procedures in conventional cloud services [7, 10, 14, 18, 20, 32, 44]. However, GenAI cloud services fundamentally differ from these. Specifically, GenAI services such as large language models (LLMs) rely on massive parameter scales, high hardware demands, and provide natural language-driven applications like text generation, summarization, and translation, which traditional cloud services do not [31, 38]. These services also allow users to fine-tune models using user-uploaded datasets [2], exposing risks from model-level behavior changes. Moreover, they provide intuitive conversational user interfaces, making them accessible to a broader audience while adding complexity and risks in managing user interactions. Such characteristics create new reliability issues related to model quality, privacy, and performance, layered atop conventional reliability concerns. Therefore, due to the distinctive challenges of GenAI cloud services, it is necessary to investigate GenAI incident patterns, impacts, and mitigation strategies to ensure future dependable and reliable GenAI services.

In this study, we examine incidents in the GenAI cloud service of *Microsoft*, a leader in the GenAI field, known for hosting GPT series models. *Microsoft*'s Incident Management system (IcM) documents a wide range of incident data, including root causes, mitigation steps, and detailed engineer discussions, enabling a comprehensive and comparative analysis of GenAI cloud service incidents alongside conventional cloud services. Our investigation reveals that while some traditional reliability challenges, such as system downtime or latency issues, remain relevant in GenAI services, new and unique challenges have emerged. For example, incidents like response quality degradation show that models can unexpectedly produce low-quality or even inappropriate output from simple prompts. We term these incidents *GenAI incidents* in this study.

Our study leads to crucial findings. For instance, we find that (1) GenAI incidents manifest as performance degradation (49.8%), deployment failure (35.7%), and invalid inference (14.5%), significantly impacting both service reliability and user satisfaction; (2) GenAI cloud services experience a higher rate of incidents detected by humans (38.3%) compared to other services (13.7%) rather than automated monitors. Also, there is a higher false alarm rate for GenAI (11.0%) versus other services (3.8%); (3) Due to human-reported nature, many GenAI incidents need to be re-assigned to different teams, and GenAI incidents need more time (1.12 time units on average) to mitigate compared to those in other services (0.65 time units on average); (4) During mitigation, a specific root cause is not tied to a single type of fix. For example, while code bugs account for 21.5% of the GenAI incidents, only 7.6% of fixes are code changes, with other strategies being employed. Given the tight deadlines for on-call engineers, quick approaches like rollback are prioritized to reduce downtime.

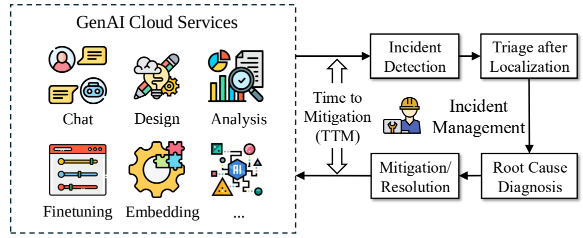


Figure 1: Incident management of GenAI cloud services.

In summary, this paper makes the following main contributions:

- We make the first attempt to unravel the general behavior of incidents occurring in GenAI cloud services by collecting and analyzing a large number of GenAI-related incidents from *Microsoft*[1].
- We identify not only the symptoms and impacts of high severity GenAI incidents but also uncover the root causes behind them and how they were mitigated with many real-world incident cases.
- We reveal the challenges of handling GenAI incidents at different incident life cycles and provide insights into improving the reliability of large-scale GenAI cloud services.

2 Background and Motivation

In this section, we begin by introducing Large Language Model (LLM) cloud services and incident management, as illustrated in Figure 1. Subsequently, we outline the motivation behind our study.

2.1 GenAI Cloud Service

With the substantial parameter scale of foundation models such as GPT-4, they are typically deployed in cloud systems like *Azure OpenAI*. This Cloud4AI service offers users a convenient means to access advanced language models without the complexities of managing infrastructure or undertaking extensive local computations. GenAI also has APIs for cloud services, as seen with Copilot [3], referred to as AI4Cloud. In our study, both Cloud4AI and AI4Cloud are the subjects of our investigation, which we collectively refer to as GenAI cloud services. In our study of GenAI cloud services' incidents (termed as GenAI incidents), we collect incident data from the *Microsoft* Incident Management system (IcM) [13] (Section 3).

2.2 Incident Management

In cloud services, incidents are common and can lead to service disruptions, economic losses, and other unexpected severe consequences. To address such issues, major cloud providers like *Microsoft* typically involve four main procedures: detection, triage, diagnosis, and mitigation (Figure 1).

- **Detection**. This step detects service violations or performance issues and creates a ticket to record relevant information [23, 30, 36, 40, 54, 55, 58, 62]. Such incidents can be detected manually

[1]Due to company policy, we hide the actual numbers and present normalized numbers in this paper.

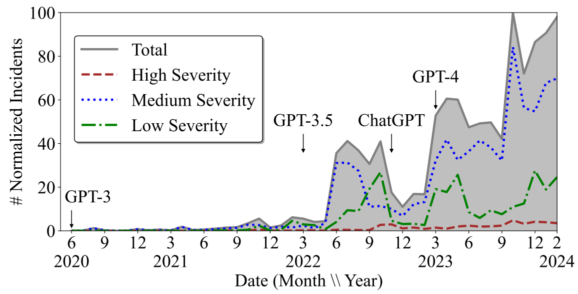


Figure 2: Number of GenAI incidents at different time.

(e.g., by customers or engineers) or automatically (e.g., by the service monitor) [23, 30, 36, 40, 54, 55, 58, 62].

- **Triage**. This process assigns the detected incident to a responsible team [5, 8, 9]. Due to the complexity of cloud service systems, determining the appropriate team may require multiple rounds of discussions, and reassignment is also necessary.
- **Diagnosis**. The assigned team analyzes the incident to determine its root cause by examining system logs and configuration settings to isolate the problem and identify corresponding factors [28, 29, 45, 51, 59–61].
- **Mitigation**. The mitigation step often accompanies the diagnosis, as engineers strive to promptly resolve the incident to minimize the Time to Mitigate (TTM) [4, 26].

2.3 Motivation

GenAI, especially LLMs such as OpenAI's ChatGPT, has witnessed a surge in their popularity, with ChatGPT having over one million users in its debut week. However, such increased adoption has also unveiled potential risks, including outages, and errors. Figure 2 showcases the variation in terms of the number of GenAI-related incidents within *Microsoft* over the recent four years, also highlighting changes in the number of incidents across different severity levels. The lower the severity level, the higher the impact to customers.

The total number of incidents in gray color shows an upward trend. Specifically, before the release of the GPT-3.5 model in March 2022, GenAI-related incidents account for a mere 3% of the total incidents within the GenAI cloud service. After 2023, there is a significant increase in incidents, with a pronounced spike following the introduction of GPT-4 in March 2023. At this point, the volume of incidents had increased nearly tenfold relative to the figures reported during the GPT-3.5 era. This dramatic rise can be attributed to the global fame attained by the GPT model, which attracted millions of users worldwide. This trend also holds across all severity levels, with lower-impact incidents constituting most cases.

The proliferation of GenAI-related incidents affects both the associated cloud services and their end users. Unfortunately, the characteristics of the incidents of GenAI cloud services have not yet been comprehensively unveiled. This study aims to bridge this gap, thus providing insights for future research and practical guidance for the software engineering community maintaining GenAI cloud services.

3 Methodology

Microsoft, a leader in cloud computing, hosts the training and APIs for OpenAI and offers various GenAI cloud services, including *Azure OpenAI*, which utilizes *Microsoft* platform to provide access to the GPT series models. Incidents in these services are documented in a dedicated database. Prior researches [20, 32, 44, 63] have utilized this database to collect incidents and derive analytical insights. Consistent with this approach, this study leverages *Microsoft*'s database to collect GenAI-related incidents.

The database contains key details for each incident, including its description, root cause, mitigation steps, discussions by the on-call engineers (OCEs), and severity-level tags (high, medium, and low). To conduct our empirical study, we collect both GenAI-related and non-GenAI incidents as a comparative dataset. Following the methodology of previous research [20], we focus on significant incidents characterized by their high severity and detailed root cause descriptions, thereby facilitating an insightful qualitative analysis. The following shows the details of our methodology.

3.1 Data Collection

We first introduce the detailed procedures to collect the dataset. In particular, we collect two datasets that serve for the two incident study respectively. The general incident study is designed to explore general characteristics of GenAI incidents within the incident management process, such as the distribution of incidents' detection methods. It also aims to compare these incidents with those from other cloud services, analyzing the differences between the two. It requires a dataset with broad coverage. Therefore, we endeavor to collect all incidents that meet the criteria as comprehensively as possible. The in-depth incident study focuses on understanding the categories of an incident's symptoms, root cause, and mitigation strategies based on detailed information, such as discussions by OCEs. Given the large volume of data, we opt to select only high-severity incidents as in-depth analysis cases, as these incidents have a more significant impact on the system and tend to attract greater attention. Through both the study, we can comprehensively understand the characteristics of GenAI incidents.

• **GenAI incidents collection for general analysis**. In this phase, our primary goal is to gather data on incidents, encompassing the period from June 2020, following the release date of GPT-3 model by OpenAI, to February 2024. Here are the criteria we use to collect GenAI-related incidents:

(1) We choose incidents that have been mitigated or resolved. The incident status is categorized within the "Status" field as "Active", "Mitigated", and "Resolved". Our collection excludes incidents marked as "Active" due to the lack of comprehensive data, such as discussions by OCEs, root cause analysis, and mitigation steps;
(2) The "Service" field indicates whether the incident is associated with a GenAI cloud service and its team or not. Incidents are considered GenAI-related if they are linked to a specific GenAI cloud service, such as *Azure OpenAI*;
(3) Given the complex architecture and dependencies of GenAI cloud services, certain GenAI incidents may be managed by dependent (sub-)services and cannot be directly found by "Service". Thus, we define a vocabulary of words related to GenAI

(*e.g.*, "OpenAI", "GPT", "LLM", *et.al*). Then we perform a case-insensitive search of these terms within the "Title" of an incident.

Following these criteria, we obtain hundreds of thousands of GenAI-related incidents.

• **GenAI incidents collection for in-depth analysis**. We meticulously select a subset of GenAI incidents based on three criteria:

(1) The incident must be of high severity. Incidents of this nature typically result in significant service disruptions, affecting numerous tenants and customers;
(2) The incident should include a detailed root cause analysis;
(3) The incident must be valid. We deem an incident as invalid if its mitigation steps are described as a "False Alarm".

Following these criteria, we identified and selected many incidents for our detailed analysis. Given that high-severity incidents inherently constitute a smaller proportion of total incidents, the data collected at this step is significantly less than what is gathered for qualitative analysis.

• **Other incidents collection**. For discussion, especially a comparative mitigation analysis between GenAI incidents and those unrelated to GenAI, we collect the same number of other incidents using the same time frame and criteria in general analysis (omitting the (2) and (3)) and in-depth analysis.

3.2 Research Questions

In this study, we aim to reveal the behaviors of GenAI incidents in the incident management life cycle. Such insights are critical for the development, maintenance, and management of LLMs, aiming to improve the robustness and reliability of the LLM cloud systems. This exploration is pivotal for providing a scientific basis to prevent future incidents, thereby contributing valuable knowledge and experience to both the research and practical applications in the field. In particular, we design the following research questions (RQs).

RQ1. What is the general behavior of GenAI incidents in terms of different incident life cycles?

RQ2. What are the symptoms of GenAI incidents?

RQ3. What are the root causes of GenAI incidents?

RQ4. How are GenAI incidents mitigated?

3.3 Categorization Strategy

While each incident is documented with detailed information, these records are typically composed by humans, e.g., OCEs' discussion, and may contain images, URLs, and other elements that complicate automatic categorization. Therefore, we need to analyze all the incidents manually to further understand their symptoms, root causes, and mitigation strategies.

We divide our dataset of incidents into three subsets randomly: (1) *taxonomy set*: 40% incidents, (2) *validation set*: 20% incidents, and (3) *test set*: 40% incidents. Firstly, two authors independently following the open coding strategy [43] to label both symptoms, root causes and mitigation strategies for the *taxonomy set.* Next, for categories with inconsistent classifications, a meeting involving other authors will be convened to determine the final categorization. The two authors then label the *validation set* to check for the emergence of new categories to perform further discussions to refine their understanding of each category. Finally, they label the *test set* and employ Cohen's kappa [15] coefficient to measure the consistency between annotators.

After multiple rounds of the labeling process described above, we ultimately adopt the best result, achieving near-perfect agreement across the three taxonomies: Symptom: 0.921, Root Cause: 0.930, Mitigation: 0.893. For incidents that can fit into multiple categories, e.g., multiple symptoms, disagreements are resolved by focusing on the category most prominently reflected in the incident and OCE's discussions.

3.4 Threats to Validity

Internal threat. Subjectivity may occur during manual labeling as an internal threat. To mitigate this threat, our study go through multiple rounds involving independent labeling, meetings to discuss categorization, and the calculation of Cohen's kappa [15]. We ultimately select the round of labeling that is near-perfect as our final result, which demonstrates the highest consistency.

External threat. All incidents we collect come from *Microsoft*'s cloud systems. Given that *Microsoft* employs various effective tools and techniques to eliminate bugs and deploys multiple automated tools to mitigate some incidents before they impact customers, the incidents we collect may not fully represent the behavior of other GenAI cloud services. We plan to perform a larger scale evaluation of GenAI cloud services from different companies in the future.

4 RQ1: General Statisitcs

We explore the characteristics of GenAI incidents from three aspects, *detection*, *triage*, and *mitigation*, each corresponding to a phase of the incident life-cycle.

4.1 Incident Detection

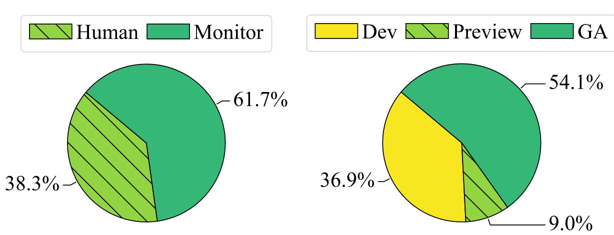
Detection is the initial step in incident management for a cloud service. Engineers can identify incidents by noticing unusual system behaviors [7, 11, 46, 56], while customers can also report issue tickets when encountering failure messages or experiencing delay [21]. To improve the efficiency of incident detection, automated monitoring tools are deployed [62]. These tools either passively collect real-time system telemetry data (e.g., CPU usage) and performance measures (e.g., response time and throughput), or proactively check the health of the system by periodically performing heartbeats or sanity checks. Figure 3 shows a monitor detecting the calling failure rate of a service.

Missing Alarms (False Negative): As shown in Figure 4, we find that 38.3% of the incidents related to GenAI are reported by humans, such as engineers and customers, instead of automated incident monitors. To explain this high human-reported percentage (*i.e.*, such a ratio is only 13.7% for other cloud services, as will be further discussed in Section 8.1) , we find that 45.9% of GenAI cloud services are still under development or in the preview stage, while 54.1% of GenAI cloud services are in the General Availability status. Moreover, many GenAI cloud service monitors currently build on adaptations of existing frameworks designed for other types of cloud services, which may not yet fully align with the unique requirements of GenAI-specific scenarios. For instance, invalid inference incidents are often identified and reported by users,

Title: monitor evaluated high fail rate for scope [ServiceA], zone [WestRegion2]
Monitor Name: [Service]_FailRate
Metric: DependencyCallCounter
Description: Marks the target as 'Unhealthy' and raises a high-severity incident if the failure rate exceeds 4% over the past 60 minutes.
Trouble-shooting Guide: [Link to the TSG]
Diagnostic Information:

Failure Description	Count
ServiceClient failure for [ServiceB]: Failed to call [ServiceB], ReasonPhrase=Failed Dependency	52
RequestTimeout for [EncoderService]	13
ServiceClient failure for ChatGPT: No service for 'BotClientLibrary' has been registered.	8
ServiceClient failure for ChatGPT: Failed to call 'ChatGPT' at LoadBalancer, ErrorStatusCode=400	3
ClientSecretCredential authentication failed: A configuration issue is preventing authentication. Details: The provided client secret keys for app [ApplicationA-UUID] are expired.	6
... ...	

Figure 3: Incident detected by a monitor and the collected diagnostic information attached to the monitor.



(a) Detection type. **(b) Service stage.**

Figure 4: GenAI incident detection type and different stages of GenAI services. GA: General Availability, Dev: Development.

reflecting the collaborative effort to refine these systems further. Our study observes that there are around 25.9 unique monitors per 100 monitor-reported GenAI incidents, compared to 74.4 for other cloud services, offering an opportunity to enhance monitoring diversity. These insights highlight the ongoing evolution of GenAI monitoring approaches, as the industry continues to refine automated detection capabilities and improve response efficiency.

Wrong Alarms (False Positives): The false alarm rate for incidents detected by monitors in GenAI services is notably high at 11.0% (Table 1 in Section 8.1), compared to the 6.6% detected by humans. This higher false positive rate is primarily from the sensitivity of the monitoring systems. For example, the monitor in Figure 3 issues an incident report if the failure rate exceeds 4% within one hour. If the failure rate threshold is set lower or the monitoring period is shortened, the monitor becomes more sensitive, possibly leading to more false positives. These false alarms burden engineers with unnecessary investigations, thus delaying the resolution of true incidents.

Finding 1. GenAI cloud services and their monitoring are still in an early stage. A high percentage (38.3%) of GenAI incidents are reported by humans. Besides, among the incidents detected by the automated monitors, there is an 11.0% false alarm rate, which points to opportunities for further enhancement in monitoring precision.

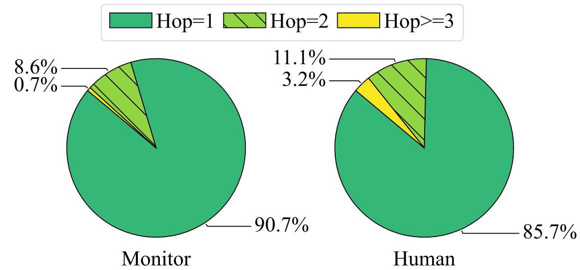


Figure 5: Transfer hops for incidents.

4.2 Incident Triage

Triage is a crucial component of the incident management life cycle, significantly affecting the Time-to-Mitigate (TTM) [7]. Incidents can be sent to incorrect teams or need collaborative efforts, leading to cases where they are re-assigned between different teams. The process of reassigning an incident from one team to another is called a *transfer hop*.

As shown in Figure 5, incidents that are initially detected by monitoring systems are usually accurately triaged to the correct team on their first attempt (90.7%). However, the proportion of incidents needing triage increases when detected by humans. GenAI incidents detected by humans that undergo reassignment is 14.3%. This shows the effectiveness of using automatic monitors for triage. For example, the monitor-generated ticket title embeds the name of the service that leads to the incident, as shown in Figure 3, so the incident can be accurately triaged to the service team. Another factor for the incident re-assignment is the interdependency on other services. Resolving an incident might exceed the capabilities of a single team, and collaborative efforts across different service domains are needed. Further details on the root causes of GenAI incidents will be elaborated in Section 6.

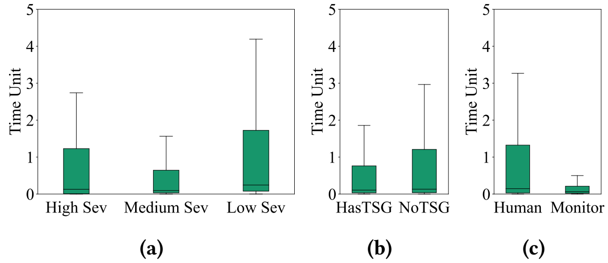


Figure 6: TTM distribution across different factors: Y-axis is the normalized TTM of all incidents; the top whisker of each box plot represents the maximum value; the top and bottom edge of the box represent the upper quartile and the lower quartile, respectively, and the line inside the box represents the median value. (a) Different severity levels; (b) The presence of a TSG; (c) Detection types.

4.3 Incident Mitigation

Intuitively, we would expect incidents with higher severity to have longer TTM, as these incidents usually require more extensive investigation and resolution efforts. For example, as shown in Figure 6a, high-severity incidents generally take longer to resolve than medium-severity. However, low-severity incidents exhibit a significantly longer TTM compared to other severity levels because these lower-priority GenAI incidents often remain unresolved for extended periods due to their low impact.

Incidents accompanied by a TSG are resolved more swiftly than those without one, as TSGs provide clear guidelines and solutions that facilitate faster mitigation (Figure 6b). Furthermore, our analysis reveals that incidents generated by monitors are mitigated more quickly than those reported by humans (Figure 6c). This is partly because monitors, as illustrated in Figure 3, often include links to corresponding TSGs. By following the TSG instructions, diagnostic information is more readily collected. For example, it becomes immediately clear that the root cause of the incident in Figure 3 is expired secret keys of the application, thereby enabling quicker resolution.

Finding 2. Automatic monitors and trouble-shooting guides (TSGs) can significantly boost the mitigation process, and reduce the Time to Mitigation for GenAI incidents.

5 RQ2: Symptom of GenAI Incidents

We analyze the symptoms of GenAI incidents from available incident-related telemetry data (metrics, logs, traces, etc.) and discussion threads from on-call engineers. We categorize the symptoms into invalid inference, deployment failure, and degraded performance. Note that one incident may have multiple symptoms, and we choose the major symptom as its category as mentioned in Section 3.3. The following subsections are ordered based on their perceived impact on service operation and user experience.

5.1 Invalid Inference (14.5%)

While the model inference executes successfully and the service returns results to clients without errors, the model output can be invalid. Inaccuracies in the output directly affect the core functionality of GenAI services. *(1) Response Quality Degradation (10.7%):* Models can generate low quality content with even simple user prompt. Another scenario involves the generation of invalid content, where the model could not understand the user's prompt, leading to invalid content creation [47]; *(2) Prompt/Response Content Filter Malfunction (3.8%):* GenAI cloud services deploy policy filters for both user prompts and model responses to prevent the generation of harmful content. However, these content filters can sometimes malfunction, resulting in inappropriate or harmful content from the model, as well as false alarms that incorrectly filter out valid prompts or responses.

5.2 Deployment Failure (35.7%)

Deployment failures reflect the impact on GenAI service continuity. We find: (1) *Model deployment failure (12.0%)*: When users are training or fine-tuning large language models, the deployment failure may happen. For instance, all of the user fine-tuned models were not successfully deployed in time for a specific deployment region; (2) *Resource deployment failure (14.4%)*: GenAI cloud services heavily depend on different types of resource deployment, like computing, networking and storage resources for consuming, transmitting and storing vast volumes of data. Failed deployment of these can propagate exceptions to other parts of the GenAI services; (3) *Fine-tune API failure (9.3%)*: GenAI cloud services offer interfaces for uploading/downloading data, model selection, and parameter setting, which users can customize to fine-tune their own models. However, a failure may happen when calling such fine-tune REST APIs. For instance, a conflict version requirement caused the failure of the fine-tune API calls.

5.3 Degraded Performance (49.8%)

There are two typical performance degradation: (1) *Service-level Degradation (27.2%)*: Multiple APIs within a GenAI service can fail simultaneously, impacting the overall availability and performance of that service. Also, if multiple service nodes become unhealthy, e.g., out-of-memory or disk pressure, the performance of the whole service can be influenced; (2) *API-level Degradation (22.6%)*: A particular GenAI API can be delayed. Degraded performance is primarily due to infrastructure and configuration issue as discussed in Section 6.

Finding 3. GenAI incidents can occasionally include challanges, including invalid inference (14.5%), deployment failure (35.7%), and performance degradation (49.8%).

6 RQ3: Root Cause

We categorize the root causes of GenAI incidents into five distinct types. The relationships between symptoms and root causes are shown in Figure 7. Each cell represents the percentage of a specific symptom associated with a particular root cause. We can observe that a single symptom can come from multiple root causes rather

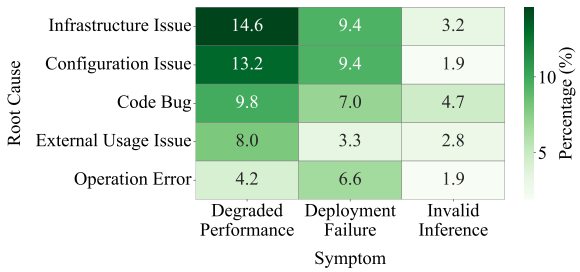


Figure 7: Relationships between symptom and root cause.

than a simple one-to-one relationship. This indicates that diagnosing the root cause from symptoms is not straightforward.

6.1 Infrastructure Issue (27.2%)

GenAI cloud services are built upon a complex hierarchical infrastructure comprising VMs, nodes, clusters, and data centers that host tightly coupled resources, including CPU, memory, storage, and networks. We find that infrastructure issues are a major cause of degraded performance and deployment failure (Figure 7). The infrastructure is categorized into the following types: (1) *Infrastructure Maintenance Issues (17.8%)*: Failures of hardware components, such as worn-out GPUs, can impact the fine-tuning and inference of GenAI services. For instance, faulty GPUs can process requests incorrectly, resulting in errors such as gibberish outputs. (2) *Network Issues (4.7%)*: Besides the network bandwidth, incidents can happen between the communication of VMs and nodes within clusters, including connectivity issues and DNS resolution failures. Such network problems can severely disrupt the performance and reliability of the service. (3) *Storage Issues (4.7%)*: The management of vast amounts of data needs robust storage solutions. Failures in data storage or IO operations, such as data corruption or delays, can lead to service disruptions.

Finding 4. Infrastructure issues are a key area of focus for understanding and addressing incidents in GenAI cloud services, especially for degraded performance and deloyment failure. To meet the growing user demands, GenAI cloud services should not only scale up the size of GPU cluster but also prioritize robust infrastructure management.

6.2 Configuration Issue (24.5%)

GenAI cloud services rely on a multitude of configuration settings to ensure the seamless operation of their interconnected components. However, mismanagement of these configurations is occasionally observed. Incorrect or unsynchronized settings can ruin service functionality. We categorize these configuration issues into the following types: *(1) Misconfiguration (13.1%)*: Operators may employ incorrect configurations or commit errors, typically due to human mistakes. For example, engineers might configure much fewer model instances than required during system maintenance, leading to an outage of degraded performance. (2) *Configuration Update (6.4%)*: Changes in one cloud component's configurations can lead to incompatibilities with other components due to the configuration dependencies among them. Additionally, version conflicts for the same configuration may result in one configuration overriding another, e.g., using a removed parameter in its latest version or using an added parameter in its previous version, leading to malfunctions. (3) *Configuration Missing and Gaps (5.0%)*: Missing or disabled configurations can disrupt normal operations. Additionally, certain configurations impose range restrictions on values, such as timeout thresholds or maximum sizes for prompt tokens. Under unexpected circumstances, such as a sudden surge in user traffic, these static configurations can constrain system performance.

6.3 Code Bug (21.5%)

Code bugs are a primary cause of incidents, and a prior work [32] has specifically investigated the code bugs leading to cloud incidents. The following shows four types of code bugs for GenAI incidents: data constraints bugs, content filter bugs, exception handling bugs, and cross-system bugs. (1) *Bugs violating Data Constraints of the Model (6.7%)*: Bugs can arise due to inadequate validation for data format or missing data that the model needs to consume. Take a fine-tune failure as an example, it can be caused by the lack of validation on dataset format in `FileUpload` API. The malformed dataset was not rejected during the file upload stage, and was delivered to the backend services; (2) *Prompt/Response Content Filter Bugs (2.2%)*: Code defects can exist in the prompt or response filter. (3) *Exception Handling Bugs (6.3%)*: Exceptions are a normal occurrence during code execution. However, the code can be unable to effectively handle certain exceptions or failures. For example, errors may occur during model deployment, such as an invalid model being deployed to an endpoint. Due to a code defect in processing such an error, e.g., simply swallowing the exception, the invalid model remains there and serve requests; (4) *Cross-system Bugs (6.3%)*: These bugs are mostly caused by issues in the code across multiple components. To fix this type of bugs, changes are needed for multiple services.

6.4 External Usage Issue (14.1%)

Incidents can arise from incorrect usage of GenAI service by the customer. For example, a customer missed indexes when performing queries to LLM, which caused the high CPU usage in the service.

6.5 Operation Error (12.7%)

Operation errors in GenAI cloud services are typically caused by human errors during the management and operational processes. This error occurs when operators mistakenly introduce erroneous or outdated dependencies, or use expired credentials.

7 RQ4: Mitigation

To answer RQ4, we delve into the common categories of mitigation strategies utilized to address GenAI incidents. Specifically, we inspect the title and the detailed description of the mitigation steps in each incident ticket and its corresponding postmortem report. these descriptions, engineers' discussion thread, and completed work bullets, we classify the mitigation methods into the following distinct types: ad-hoc fix, self-recover, rollback, configuration fix, infrastructure fix, external fix, code fix, and others.

7.1 Code Fix (7.6%)

This category is to address incidents by updating and fixing buggy code or by incorporating new code [32], such as adding exception handling mechanisms to improve resilience or implementing new features for specific purpose. For example, certain Unicode characters cannot be rendered in a font and thus do not appear in the user interface, resulting in what is called hidden text. However, the hidden text can still be understood and processed by the LLM. This could be potentially exploited as an attack surface to change the response from the user's intent. The following code update adds a new feature to remove the Unicode characters (within the range of U+E0000 to U+E007F) that can be used as hidden text from the user's request.

```
const getCommandText = () =>
    featureFlags.
        enableRemoveUnicodeFromRequest
    ? removeUnicodeFromRequest(text) :
        text;
...

const removeUnicodeFromRequest = (msg:
    string) => {
    const unescapedMsg = unescapeUnicode(
        msg);
    const regex = /[\u{E0000}-\u{E007F}]/
        gu;
    return unescapedMsg.replace(regex, "")
        ;
};
```

Finding 5. Given the tight deadlines for mitigating GenAI incidents, only a small proportion (7.6%) of incidents are resolved through code fixes. This approach is time-consuming, requiring more efforts to design and implement the solution and navigate through an end-to-end CI/CD pipeline. Consequently, other mitigation strategies are preferred by engineers for their faster resolution times in the initial stages of mitigation.

7.2 Rollback (15.2%)

For incidents triggered by changes, such as configuration adjustments or code updates, rollback is a widely used and efficient mitigation strategy. Engineers revert these changes to a previous, stable version. Our study identifies: (1) *Deployment Rollback (8.9%)*: Updates to code or third-party libraries can introduce bugs. These incidents can be addressed by reverting to a previous commit or an older stable build version of the third-party library. For example, an inference API error which caused by the compatibility issue between fine-tuning code and inference code can be fixed by rolling back to a previous inference engine for users in specific regions; (2) *Configuration Rollback (6.3%)*: This involves undoing bad configuration changes to alleviate the issue.

7.3 Configuration Fix (13.0%)

To address the majority of configuration errors, engineers often fix bugs in configuration files to reinstate the service. We identify two primary approaches to configuration fixes: (1) *Add or Disable Features (7.6%)*: Incidents can be mitigated either by adding new features that enhance service stability or by disabling features that are causing failures, thus aiding in the swift resolution of the issue; (2) *Increase the Configuration Limit (5.4%)*: Besides the configuration issues, a number of incidents from resource capacity as mentioned in Section 6.1 can also be mitigated by configuration changes as a short-term strategy, such as increasing timeout thresholds.

7.4 Infrastructure Fix (12.1%)

For incidents caused by infrastructure issues, an infrastructure fix is a frequently utilized mitigation method. Common infrastructure fixes include scaling operations, component restarts or rebuilds, and traffic failovers. One of the following actions can be performed: (1) *Scaling (6.3%)*: Due to infrastructure limitations, a service may not be able to handle a large volume of traffic, and simple configuration of increasing the capacity does not work. Therefore, scaling out more instances or nodes to increase capacity is needed. For example, increasing the compute capacity allows the service to process more requests, thus avoiding an excessive number of request failures; (2) *Restart or Rebuild (3.1%)*: This category involves mitigating incidents by restarting or rebuilding faulty components; (3) *Traffic Failover (2.7%)*: This involves failing over traffic to another healthy service component, including nodes, clusters, or another cloud region.

Finding 6. In practice, increasing the limit of resource configuration (5.4%) is a straightforward mitigation strategy. However, when these configuration changes are insufficient due to the allocated resources or infrastructure reaching their capacity, re-scaling (6.3%) becomes necessary to resolve the GenAI incidents, even though it may take a long time to deploy the additional infrastructure.

7.5 Ad-hoc Fix (22.4%)

LLM incidents can be complex, and engineers may not always be familiar with the root cause of GenAI incidents. To address the impact quickly, standardized procedures can be costly, so a series of improvised, situation-specific steps are applied to mitigate the symptom first. For instance, in response to a malicious user bypassing the batch size limitation, engineers mitigated it by identifying and blocking the malicious user, enabling the validation logic to check the `ImageModelA-batch-size` parameter in the request headers, and enforcing a maximum limit for the batch size. Also, in other cases where a single user's request consumed too many background resources and resulted in service overload, the issue was mitigated by temporarily limiting the user's request rate, adjusting the throttling from 10 seconds throttling to one second for the customer with a high workload. Note that over half of the incidents from other cloud services are mitigated by ad-hoc fix (Figure 9), while GenAI cloud services often require more development and deployment efforts (other mitigation approaches to be discussed in the following) to fully resolve the incidents. Consequently, the TTM of GenAI incidents are longer.

7.6 External Fix (10.0%)

GenAI cloud services support external company partners and customers, so some incidents are mitigated externally, including by

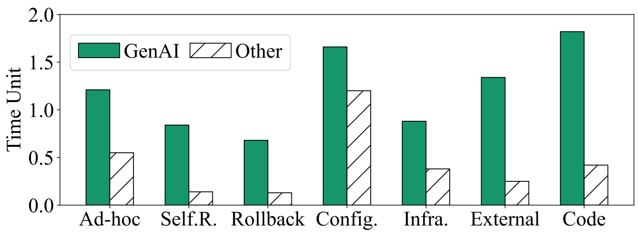


Figure 8: Average TTM for different mitigation approaches.

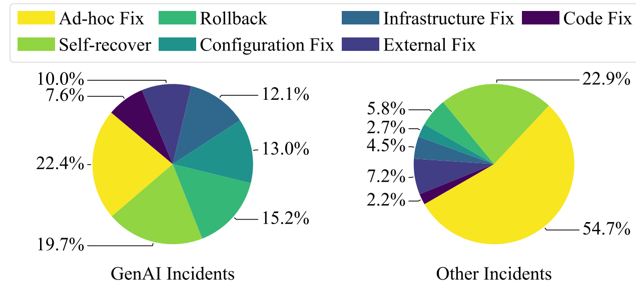


Figure 9: The distribution of mitigation approaches.

Microsoft Partners and customers. For example, engineers will recommend that customers modify their prompts when their wrong usage causes the model to return unexpected content or switch to a stable model.

7.7 Self-recover (19.7%)

These transient incidents are automatically mitigated as the service recovers on its own due to its resilience mechanisms, for example, back-off retry, or when the monitoring system no longer detects abnormal indicators, e.g., heartbeat detection rate returns to normal. Note that self-recovered incidents are not false alarms in our dataset.

8 Discussion

8.1 Lessons Learnedd

Since the mitigation strategy categories for both GenAI and non-GenAI share high similarities, we further perform a comparative study to identify their distinctions. We find that GenAI incidents generally require more time to mitigate compared to other types. Specifically, on average, GenAI incidents take 1.12 time units to resolve, compared to 0.65 time units for non-GenAI incidents.

To reveal the underlying reason: (1) We calculate the TTM for each type of mitigation category, and find that the longer TTM for GenAI incident holds across all mitigation categories, as shown in Figure 8, reflecting the complexity of solving various GenAI incidents. Additionally, across all factors we consider (severity levels, detection types, troubleshooting guides) in the general analysis in Section 4.3, the Time to Mitigation (TTM) for LLM incidents is consistently longer than for incidents in other services. (2) We compare the distribution of mitigation approaches, as depicted in Figure 9. The ad-hoc fix (54.7%) is the majority of the mitigation for other cloud services, which have shorter TTM compared to any GenAI incident mitigation in Figure 8. The mitigation distribution of GenAI incidents is more balanced, with ad-hoc fixes comprising only 22 4%. This indicates that, for GenAI cloud services in their early development stage, more diverse, sophisticated, and time-consuming methods are required as opposed to applying the ad-hoc fixes. (3) The current monitoring tools for GenAI cloud services are being continuously improved to better align with their unique requirements. Enhancements in accuracy and adequacy are expected to help reduce TTM and improve overall efficiency. Unlike conventional cloud services monitored by automated watchdogs, a high percentage of GenAI incidents are detected by humans. According to Table 1, only 13.7% of the incidents were detected by humans for non-GenAI cloud services in our dataset, compared to 38.3% for GenAI incidents. Furthermore, monitor-detected GenAI incidents have an 11.0% false positive alarm rate, significantly higher than the 3.8% observed in other services. This suggests that the current monitor is not mature compared to conventional incidents, and requires additional effort to improve.

Longer TTM is also attributed to the difficulty in performing root cause analysis for GenAI incidents. As discussed in Section 7, a single symptom can stem from multiple root causes, thus complicating the debugging of GenAI services. For example, diagnosing unexpected model outputs can be complex; potential causes include faulty hardware, misconfigurations, code defects, or misuse.

Table 1: Detection type distribution and false alarms rate for GenAI and non-GenAI incidents.

	Detection Type		False Alarm Rate	
	Human	Monitor	Human	Monitor
GenAI	38.3%	61.7%	6.6%	11.0%
Other	13.7%	86.3%	4.8%	3.8%

8.2 Implications

Our findings offer actionable insights for a wide range of stakeholders, including researchers, model providers, service maintainers, developers, and etc.

Researchers. Our study highlights several avenues for future research, particularly in automated methods to detect invalid inference results. Currently, invalid outputs (14.5%), such as hallucinations or irrelevant responses, are challenging to detect. The current state-of-the-art detection methods generally include 1) self-judgment by the LLM, 2) fine-tuning another model with human-labeled data, or 3) calculating consistency scores after multiple attempts. However, neither of them is cost-efficient nor fully effective. More robust research is needed to address these limitations and develop scalable validation algorithms that can operate across various GenAI applications.

Model Providers. Besides the high ratio of invalid inference results (14.5%) and challenges in detecting hallucinations or invalid content, another notable finding is that 38% of GenAI incidents are reported by humans, reflecting that monitoring tools are underdeveloped. Moreover, many GenAI cloud services (45.9%) are still under development or in the preview stage, coupled with the scarcity of incident monitor types. Providers should enhance service observability to detect and diagnose issues more effectively, and provide better support and documentation to help users navigate the complexities of GenAI service integration and management.

Service Maintainers. Our study reveals that the Time-to-Mitigate (TTM) for GenAI incidents is 1.83 times longer than for non-GenAI incidents, highlighting the need for automation in incident mitigation. The complexity of GenAI systems, which involve vast and interconnected layers of infrastructure, dependencies, and configurations, is a significant factor. For example, GenAI cloud systems require 2.5x more infrastructure fixes, 3.0x more code changes, and 3.0x as many configuration updates compared to non-GenAI services. Despite these, more straightforward ad-hoc fixes are applied in only 22.4% of GenAI incidents, compared to 54.7% in non-GenAI services, indicating a reliance on more complex, time-consuming fixes for GenAI systems. Furthermore, diagnosing root causes of GenAI incidents is often complex. A single symptom, such as poor performance (49.8%) or deployment failure (35.7%), can have multiple root causes, including infrastructure problems (27.2%), configuration problems (24.5%), or code bugs (22.5%). Services should provide observability from different dimensions to obtain granular insight into these symptoms and their underlying causes. Maintainers should consider 1) implementing more automation tools or agents for distinct mitigation approaches, 2) adopting more infrastructure-as-code practices to manage complex GenAI cloud infra more effectively, and 3) integrating more automated rollback mechanisms to address compatibility issues swiftly.

Application Developers and Users. For developers, input validation and dynamic rate limiting are critical areas needing improvement. Incidents reveal that special characters, fragmented prompts, and excessive token usage, even within token limits, can disrupt model processing. Developers should implement strict input validation processes to prevent these issues and adopt dynamic rate-limiting strategies that adapt to real-time conditions.

9 Related Work

Empirical studies on cloud incidents. A significant amount of prior work has been devoted to studying the characteristics of incidents occurring in production systems. Ganatra et al. [19] examined incident detection at Microsoft to identify monitoring gaps in cloud platforms. Chen et al. [7] studied incident triage in Microsoft's online service systems to understand industry practices. Zhao et al. [63] explored change-induced incident lifecycles in large-scale online services, offering management insights. Wang et al. [44] analyzed the time-to-mitigation (TTM) of incidents across 20 Microsoft online services. Building on this, our study delves into incident characteristics, comparing incidents related to GenAI with those of other services. In related work, Liu et al. [32] investigated software bugs causing cloud incidents in Microsoft Azure and their resolutions. Ghosh et al. [20] analyzed incidents in Microsoft Teams, classifying root causes and mitigation steps. Martino et al. [17] characterized failures in a business data processing platform using event log data. Our study closely aligns with this body of research.

LLMs empirical study. In recent years, with the rise of large language models (LLMs), numerous related studies have emerged. Cui et al. [16] organize existing studies related to LLMs and propose a comprehensive taxonomy, which systematically analyzes potential risks in LLM systems and discusses corresponding mitigation strategies. Liu et al. [34] investigate the use of jailbreak prompts to bypass restrictions imposed on ChatGPT. They conduct an empirical study to evaluate the effectiveness and robustness of prompts collected from the real world. Zhuo et al. [65] present an empirical study on the adversarial robustness of a prompt-based semantic parser based on Codex. Yang et al. [50] conduct a study on GPT-3 in knowledge-based visual question answering (VQA), treating GPT-3 as a knowledge base (KB) and adapting GPT-3 to solve the VQA task in a few-shot manner. Our study focuses on the incidents in GenAI cloud services.

10 Conclusion

In this paper, we present a comprehensive study of incidents from GenAI cloud services within *Microsoft*. We explore the symptoms, root causes, and mitigation strategies of GenAI incidents. Our findings reveal unique characteristics in GenAI cloud services. For example, we identify notable differences between incidents from LLM cloud services and other cloud services, such as significant disparities in the time to mitigation of incidents. Additionally, we find that the primary cause of incidents in LLM cloud services is related to infrastructure. These findings provide guidance for future academic and industrial research in the field of LLM cloud services. We hope to inspire the development of advanced, specialized tooling and raise discussions on GenAI incidents, so that our community can monitor the GenAI cloud system with early warnings, triage incidents to the correct teams with fewer hops, pinpoint root causes accurately, and mitigate the incidents with optimal plans.

References

[1] 2024. Elevated errors affecting API and ChatGPT. https://status.openai.com/incidents/n38dwwksfkv9.

[2] 2024. Fine-tuning. https://platform.openai.com/docs/guides/fine-tuning.

[3] 2024. Microsoft Copilot: Your everyday AI companion. https://copilot.microsoft.com/.

[4] Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, Thomas Zimmermann, Xuchao Zhang, and Saravan Rajmohan. 2023. Recommending Root-Cause and Mitigation Steps for Cloud Incidents using Large Language Models. In *ICSE*.

[5] Chetan Bansal, Sundararajan Renganathan, Ashima Asudani, Olivier Midy, and Mathru Janakiraman. 2020. Decaf: Diagnosing and triaging performance issues in large-scale cloud services. In *ICSE (SEIP)*.

[6] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv:2303.12712* (2023).

[7] Junjie Chen, Xiaoting He, Qingwei Lin, Yong Xu, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2019. An empirical investigation of incident triage for online service systems. In *ICSE (SEIP)*.

[8] Junjie Chen, Xiaoting He, Qingwei Lin, Yong Xu, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2019. An empirical investigation of incident triage for online service systems. In *ICSE (SEIP)*.

[9] Junjie Chen, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2019. Continuous incident triage for large-scale online service systems. In *ASE*.

[10] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, et al. 2024. Automatic root cause analysis via large language models for cloud incidents. In *EuroSys*.

[11] Yuhang Chen, Chaoyun Zhang, Minghua Ma, Yudong Liu, Ruomeng Ding, Bowen Li, Shilin He, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. ImDiffusion: Imputed Diffusion Models for Multivariate Time Series Anomaly Detection. *Proc. VLDB Endow.* 17 (2023).

[12] Zhenpeng Chen, Yanbin Cao, Yuanqiang Liu, Haoyu Wang, Tao Xie, and Xuanzhe Liu. 2020. A comprehensive study on challenges in deploying deep learning

based software. In *ESEC/FSE*.
[13] Zhuangbin Chen, Yu Kang, Feng Gao, Li Yang, Jeffrey Sun, Zhangwei Xu, Pu Zhao, Bo Qiao, Liqun Li, Xu Zhang, et al. 2020. Aiops innovations of incident management for cloud services. (2020).
[14] Zhuangbin Chen, Yu Kang, Liqun Li, Xu Zhang, Hongyu Zhang, Hui Xu, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, et al. 2020. Towards intelligent incident management: why we need it and how we make it. In *ESEC/FSE*.
[15] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20 (1960).
[16] Tianyu Cui, Yanling Wang, Chuanpu Fu, Yong Xiao, Sijia Li, Xinhao Deng, Yunpeng Liu, Qinglin Zhang, Ziyi Qiu, Peiyang Li, et al. 2024. Risk taxonomy, mitigation, and assessment benchmarks of large language model systems. *arXiv:2401.05778* (2024).
[17] Catello Di Martino, Zbigniew Kalbarczyk, Ravishankar K. Iyer, Geetika Goel, Santonu Sarkar, and Rajeshwari Ganesan. 2014. Characterization of operational failures from a business data processing SaaS platform. In *ICSE*.
[18] Pradeep Dogga, Chetan Bansal, Richard Costleigh, Gopinath Jayagopal, Suman Nath, and Xuchao Zhang. 2023. {AutoARTS}: Taxonomy, Insights and Tools for Root Cause Labelling of Incidents in Microsoft Azure. In *ATC*.
[19] Vaibhav Ganatra, Anjaly Parayil, Supriyo Ghosh, Yu Kang, Minghua Ma, Chetan Bansal, Suman Nath, and Jonathan Mace. 2023. Detection Is Better Than Cure: A Cloud Incidents Perspective. In *ESEC/FSE*.
[20] 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 *SoCC*.
[21] Jiazhen Gu, Jiaqi Wen, Zijian Wang, Pu Zhao, Chuan Luo, Yu Kang, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, Bo Qiao, Liqun Li, Qingwei Lin, and Dongmei Zhang. 2020. Efficient customer incident triage via linking with system incidents. In *ESEC/FSE*.
[22] Jiaming Han, Renrui Zhang, Wenqi Shao, Peng Gao, Peng Xu, Han Xiao, Kaipeng Zhang, Chris Liu, Song Wen, Ziyu Guo, Xudong Lu, Shuai Ren, Yafei Wen, Xiaoxin Chen, Xiangyu Yue, Hongsheng Li, and Yu Qiao. 2023. ImageBind-LLM: Multimodality Instruction Tuning. *CoRR* abs/2309.03905 (2023).
[23] Jun Huang, Yang Yang, Hang Yu, Jianguo Li, and Xiao Zheng. 2023. Twin graph-based anomaly detection via attentive multi-modal learning for microservice system. In *ASE*.
[24] Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. 2020. Taxonomy of real faults in deep learning systems. In *ICSE*.
[25] Md Johirul Islam, Giang Nguyen, Rangeet Pan, and Hridesh Rajan. 2019. A comprehensive study on deep learning bug characteristics. In *ESEC/FSE*.
[26] Jiajun Jiang, Weihai Lu, Junjie Chen, Qingwei Lin, Pu Zhao, Yu Kang, Hongyu Zhang, Yingfei Xiong, Feng Gao, Zhangwei Xu, et al. 2020. How to mitigate the incident? an effective troubleshooting guide recommendation technique for online service systems. In *ESEC/FSE*.
[27] Yuxuan Jiang, Chaoyun Zhang, Shilin He, Zhihao Yang, Minghua Ma, Si Qin, Yu Kang, Yingnong Dang, Saravan Rajmohan, Qingwei Lin, et al. 2023. Xpert: Empowering incident management with query recommendations via large language models. *arXiv:2312.11988* (2023).
[28] Pengxiang Jin, Shenglin Zhang, Minghua Ma, Haozhe Li, Yu Kang, Liqun Li, Yudong Liu, Bo Qiao, Chaoyun Zhang, Pu Zhao, et al. 2023. Assess and summarize: Improve outage understanding with large language models. In *ESEC/FSE*.
[29] Cheryl Lee, Tianyi Yang, Zhuangbin Chen, Yuxin Su, and Michael R Lyu. 2023. Eadro: An end-to-end troubleshooting framework for microservices on multi-source data. In *ICSE*.
[30] Liqun Li, Xu Zhang, Xin Zhao, Hongyu Zhang, Yu Kang, Pu Zhao, Bo Qiao, Shilin He, Pochian Lee, Jeffrey Sun, et al. 2021. Fighting the fog of war: Automated incident detection for cloud systems. In *ATC*.
[31] Elizabeth D Liddy. 2001. Natural language processing. (2001).
[32] Haopeng Liu, Shan Lu, Madan Musuvathi, and Suman Nath. 2019. What bugs cause production cloud incidents?. In *HotOS*.
[33] Yiren Liu, Si Chen, Haocong Cheng, Mengxia Yu, Xiao Ran, Andrew Mo, Yiliu Tang, and Yun Huang. 2023. How AI Processing Delays Foster Creativity: Exploring Research Question Co-Creation with an LLM-based Agent. *CoRR* abs/2310.06155 (2023).
[34] Y Liu, G Deng, Z Xu, Y Li, Y Zheng, Y Zhang, L Zhao, T Zhang, and Y Liu. [n. d.]. Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study. arXiv 2023. *arXiv:2305.13860* ([n. d.]).
[35] Luca Luceri, Eric Boniardi, and Emilio Ferrara. 2023. Leveraging Large Language Models to Detect Influence Campaigns in Social Media. *CoRR* abs/2311.07816 (2023).
[36] Minghua Ma, Shenglin Zhang, Junjie Chen, Jim Xu, Haozhe Li, Yongliang Lin, Xiaohui Nie, Bo Zhou, Yong Wang, and Dan Pei. 2021. Jump-Starting Multivariate Time Series Anomaly Detection for Online Service Systems. In *ATC*.
[37] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. 2023. Towards efficient generative large language model serving: A survey from algorithms to systems. *arXiv:2312.15234* (2023).
[38] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv:2202.12837* (2022).
[39] Bo Qiao, Liqun Li, Xu Zhang, Shilin He, Yu Kang, Chaoyun Zhang, Fangkai Yang, Hang Dong, Jue Zhang, Lu Wang, et al. 2023. Taskweaver: A code-first agent framework. *arXiv:2311.17541* (2023).
[40] Haoran Qiu, Subho S Banerjee, Saurabh Jha, Zbigniew T Kalbarczyk, and Ravishankar K Iyer. 2020. {FIRM}: An intelligent fine-grained resource management framework for {SLO-Oriented} microservices. In *OSDI*.
[41] Marium M. Raza, Kaushik P. Venkatesh, and Joseph C. Kvedar. 2024. Generative AI and large language models in health care: pathways to implementation. *npj Digit. Medicine* 7 (2024).
[42] S. P. Sharan, Francesco Pittaluga, Vijay Kumar B. G, and Manmohan Chandraker. 2024. LLM-Assist: Enhancing Closed-Loop Planning with Language-Based Reasoning. *CoRR* abs/2401.00125 (2024).
[43] Anselm L Strauss and Juliet M Corbin. 1997. *Grounded theory in practice*. Sage.
[44] Weijing Wang, Junjie Chen, Lin Yang, Hongyu Zhang, Pu Zhao, Bo Qiao, Yu Kang, Qingwei Lin, Saravanakumar Rajmohan, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2021. How Long Will it Take to Mitigate this Incident for Online Service Systems?. In *ISSRE*.
[45] Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Lunting Fan, Lingfei Wu, and Qingsong Wen. 2023. Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. *arXiv:2310.16340* (2023).
[46] Zexin Wang, Changhua Pei, Minghua Ma, Xin Wang, Zhihan Li, Dan Pei, Saravan Rajmohan, Dongmei Zhang, Qingwei Lin, Haiming Zhang, Jianhui Li, and Gaogang Xie. 2024. Revisiting VAE for Unsupervised Time Series Anomaly Detection: A Frequency Perspective. In *WWW*.
[47] Joel Wester, Tim Schrills, Henning Pohl, and Niels van Berkel. 2024. "As an AI language model, I cannot": Investigating LLM Denials of User Requests. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–14.
[48] Doris Xin, Hui Miao, Aditya Parameswaran, and Neoklis Polyzotis. 2021. Production machine learning pipelines: Empirical analysis and optimization opportunities. In *SIGMOD*.
[49] Shuo Yang, Zirui Shang, Yongqi Wang, Derong Deng, Hongwei Chen, Qiyuan Cheng, and Xinxiao Wu. 2024. Data-free Multi-label Image Recognition via LLM-powered Prompt Tuning. *CoRR* abs/2403.01209 (2024).
[50] Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Yumao Lu, Zicheng Liu, and Lijuan Wang. 2022. An empirical study of gpt-3 for few-shot knowledge-based vqa. In *AAAI*.
[51] Guangba Yu, Pengfei Chen, Yufeng Li, Hongyang Chen, Xiaoyun Li, and Zibin Zheng. 2023. Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data. In *ESEC/FSE*.
[52] Junchi Yu, Ran He, and Rex Ying. 2023. Thought Propagation: An Analogical Approach to Complex Reasoning with Large Language Models. *CoRR* abs/2310.03965 (2023).
[53] Shengbin Yue, Wei Chen, Siyuan Wang, Bingxuan Li, Chenchen Shen, Shujun Liu, Yuxuan Zhou, Yao Xiao, Song Yun, Xuanjing Huang, and Zhongyu Wei. 2023. DISC-LawLLM: Fine-tuning Large Language Models for Intelligent Legal Services. *CoRR* abs/2309.11325 (2023).
[54] Jun Zeng, Zheng Leong Chua, Yinfang Chen, Kaihang Ji, Zhenkai Liang, and Jian Mao. 2021. WATSON: Abstracting Behaviors from Audit Logs via Aggregation of Contextual Semantics.. In *NDSS*.
[55] Jun Zeng, Xiang Wang, Jiahao Liu, Yinfang Chen, Zhenkai Liang, Tat-Seng Chua, and Zheng Leong Chua. 2022. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In *S&P*.
[56] Zhengran Zeng, Yuqun Zhang, Yong Xu, Minghua Ma, Bo Qiao, Wentao Zou, Qingjun Chen, Meng Zhang, Xu Zhang, Hongyu Zhang, Xuedong Gao, Hao Fan, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. TraceArk: Towards Actionable Performance Anomaly Alerting for Online Service Systems. In *ICSE (SEIP)*.
[57] Chaoyun Zhang, Zicheng Ma, Yuhao Wu, Shilin He, Si Qin, Minghua Ma, Xiaoting Qin, Yu Kang, Yuyi Liang, Xiaoyu Gou, et al. 2024. AllHands: Ask Me Anything on Large-scale Verbatim Feedback via Large Language Models. *arXiv:2403.15157* (2024).
[58] Chenxi Zhang, Xin Peng, Chaofeng Sha, Ke Zhang, Zhenqing Fu, Xiya Wu, Qingwei Lin, and Dongmei Zhang. 2022. Deeptralog: Trace-log combined microservice anomaly detection through graph-based deep learning. In *ICSE*.
[59] Dylan Zhang, Xuchao Zhang, Chetan Bansal, Pedro Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. 2023. Pace: Prompting and augmentation for calibrated confidence estimation with gpt-4 in cloud incident root cause analysis. *arXiv:2309.05833* (2023).
[60] Shenglin Zhang, Pengxiang Jin, Zihan Lin, Yongqian Sun, Bicheng Zhang, Sibo Xia, Zhengdan Li, Zhenyu Zhong, Minghua Ma, Wa Jin, et al. 2023. Robust failure diagnosis of microservice system through multimodal data. *IEEE Trans. Serv. Comput.* 16 (2023).
[61] Yingying Zhang, Zhengxiong Guan, Huajie Qian, Leili Xu, Hengbo Liu, Qingsong Wen, Liang Sun, Junwei Jiang, Lunting Fan, and Min Ke. 2021. CloudRCA: A root cause analysis framework for cloud computing platforms. In *CIKM*.

[62] Chenyu Zhao, Minghua Ma, Zhenyu Zhong, Shenglin Zhang, Zhiyuan Tan, Xiao Xiong, LuLu Yu, Jiayi Feng, Yongqian Sun, Yuzhi Zhang, Dan Pei, Qingwei Lin, and Dongmei Zhang. 2023. Robust Multimodal Failure Detection for Microservice Systems. In *KDD*.

[63] Y. Zhao, L. Jiang, Y. Tao, S. Zhang, C. Wu, Y. Wu, T. Jia, Y. Li, and Z. Wu. 2023. How to Manage Change-Induced Incidents? Lessons from the Study of Incident Life Cycle. In *ISSRE*.

[64] Yunpu Zhao, Rui Zhang, Wenyi Li, Di Huang, Jiaming Guo, Shaohui Peng, Yifan Hao, Yuanbo Wen, Xing Hu, Zidong Du, Qi Guo, Ling Li, and Yunji Chen. 2024. Assessing and Understanding Creativity in Large Language Models. *CoRR* abs/2401.12491 (2024).

[65] Terry Yue Zhuo, Zhuang Li, Yujin Huang, Fatemeh Shiri, Weiqing Wang, Gholamreza Haffari, and Yuan-Fang Li. 2023. On robustness of prompt-based semantic parsing with large pre-trained language model: An empirical study on codex. *arXiv:2301.12868* (2023).