Understanding IT System Failures: Primary Fault Types, Severity Patterns, and Evolution in Modern Operations

An Analysis of Public Incident Reports Using Large Language Models

Jakub Rutkowski[1]

Supervisor(s): Diomidis Spinellis[1]**, Eileen Kapel**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Jakub Rutkowski
Final project course: CSE3000 Research Project
Thesis committee: Diomidis Spinellis, Eileen Kapel, Benedikt Ahrens

Abstract

Modern businesses increasingly rely on software-driven operations, making system reliability a critical concern. Despite advances in automated operations, gaps remain in understanding how the primary causes of system failures manifest, impact operational severity, and evolve in cloud-native environments. This study analyzes $7{,}804$ publicly available incident reports spanning 2014–2022 to examine trends in operational fault types across modern IT systems. A state-of-the-art large language model was employed to classify incidents into a consolidated fault taxonomy with an overall accuracy of $92\,\%$ and a macro-averaged $F1$-score of 0.89. The results reveal that *Misconfigurations and Deployment Failures* ($32.3\,\%$), *External Dependency Failures* ($30.0\,\%$), and *Capacity Issues* ($16.1\,\%$) are the most frequent fault types. Significant correlations were found between fault types and incident duration, with *Security Incidents* exhibiting particularly long resolution times. Temporal analysis shows a rising prevalence of *Misconfigurations/Deployment Failures* and *Software Bugs*, alongside a decline in *Infrastructure Failures*, reflecting the growing complexity and automation of modern IT environments. These findings contribute to a deeper understanding of evolving digital fragility, reveal how different fault types impact operational resilience, and offer actionable insights for improving incident management and system reliability.

1 Introduction

Modern businesses have become "software-defined businesses" [1], adopting agile methodologies that prioritize rapid development and deployment cycles [2]. These methodologies introduce risks leading to operational incidents — unplanned interruptions to services or reductions in service quality [2]. Approximately $70\,\%$ of system outages are caused by changes to live systems [3]. Artificial Intelligence for IT Operations (AIOps) has emerged as a potential solution, leveraging big data and machine learning to improve IT operations [4]. However, AIOps effectiveness depends on data quality, often compromised by manual reporting and inconsistent analysis practices [5].

While studies have examined incident management within specific companies [6] and incident-inducing changes [7], broader cross-organizational analysis remains necessary. Oppenheimer et al. [8] proposed a taxonomy of internet service failures, but cloud-native architectures and distributed systems have since transformed the landscape.

Three knowledge gaps exist: limited visibility into prevalent fault types in contemporary IT systems, a poor understanding of the operational impact of different fault types on incident duration, and a lack of systematic study of fault pattern shifts over the past decade.

This study addresses these gaps through large-scale analysis of operational incidents in modern IT environments, answering: **"What are the most common primary fault types causing operational incidents in modern IT systems, how do they relate to incident severity, and how have these patterns evolved?"** The research question breaks down into four subquestions:

1. **RQ1:** What taxonomy of primary fault types can be established for modern IT incidents, and how reliable is automated classification of these categories?
2. **RQ2:** What is the relative frequency of different primary fault types across incident reports?
3. **RQ3:** Are there correlations between specific primary fault types and the duration of incidents?
4. **RQ4:** Has the frequency of specific fault types changed over time, particularly with the adoption of cloud-native architectures?

We analyze $7{,}804$ publicly available incident reports using a large language model (LLM) to classify fault types. Results show that *Misconfigurations/Deployment Failures*, *External Dependency Failures*, and *Capacity Issues* are the most common fault categories, with significant differences in incident duration and temporal frequency shifts reflecting the growing complexity of the IT ecosystem.

This study contributes empirical evidence on digital system failures and demonstrates the value of LLM-based classification in understanding system fragility and supporting operational resilience strategies.

2 Related Literature and Background

Oppenheimer et al. [8] introduced one of the first empirical taxonomies of large-scale service failures, identifying configuration errors, software bugs, and operator mistakes as primary causes. Their work highlighted the complexity of diagnosing faults in distributed systems and offered a practical perspective on real-world outages. However, it reflects the technological landscape of its time, before the rise of cloud-native architectures, infrastructure-as-code, and continuous deployment. Huang [9] extends this perspective with a comprehensive study of real-world cloud outages, emphasizing misconfiguration, hidden dependencies, and fault propagation across microservice architectures. Our study builds upon this foundational work by establishing a contemporary taxonomy that captures fault types specific to modern distributed systems, including automated operations, third-party service dependencies, security vulnerabilities, and environmental factors, enabling a more accurate representation of failure causes in today's complex infrastructure landscape.

While the above studies focus on causes, Menges and Pernul [10] address how incidents are captured and reported. They demonstrate that inconsistent formats and limited semantic expressiveness hinder comparative analysis and shared understanding across organizations. Kapel et al. [7] highlight the challenges of accurately linking incidents to the changes that induced them, citing data quality issues and contextual variability.

To address these limitations in incident reporting and analysis, Natural Language Processing (NLP) techniques have emerged as a promising solution for extracting structured insights from unstructured incident data. Young et al. [11]

systematically reviewed the use of NLP in healthcare for classifying incident reports, emphasizing that NLP enables the extraction of structured insights from unstructured data like free text descriptions. Saha and Hoi [12] take a different approach by mining root cause knowledge from incident reports using neural NLP methods. Their Incident Causation Analysis framework constructs a Causal Knowledge Graph from thousands of historical investigations, enabling retrieval-based root cause analysis (RCA) by mapping new incident symptoms to previously resolved cases. Further extending NLP's reach, Chen et al. [13] propose RCACopilot, an end-to-end system that leverages LLMs for automated RCA of cloud incidents. Their work emphasizes the integration of multi-source data (logs, metrics, traces) and combines it with an LLM-based reasoning component, demonstrating both accuracy and real-world utility in Microsoft's production environment. Ahmed et al. [14] investigated the use of large LLMs to automate root cause identification for cloud incidents at Microsoft. Their large-scale study, spanning over 40,000 incidents, demonstrated that LLMs significantly outperformed traditional models. The study revealed not only the feasibility of using LLMs for these complex tasks but also the potential for reducing manual toil and improving response times in incident management.

Adding another dimension, Anandayuvaraj et al. [15] introduced the *FAIL system*, leveraging LLMs to analyze software failures reported in the news. This innovative approach bridges the gap where private postmortem reports are inaccessible, automating the collection, grouping, and analysis of publicly available data. Their findings revealed high recurrence rates of similar failures across organizations and an increasing severity of failure consequences over the past decade. This highlights the growing need for cross-sector learning and proactive fault analysis, using accessible and scalable data sources.

Our study builds upon prior work by leveraging a large, cross-organizational dataset of postmortems, providing more reliable insights than news-based sources, and by utilizing LLMs to extract fault information from unstructured incident descriptions at scale. This enables a systematic analysis of fault types, their severity, and their evolution, addressing core knowledge gaps in contemporary IT operations research.

3 Methodology

This section outlines our research approach to identifying the primary fault types underlying operational incidents in modern IT systems, which combines data from public incident repositories, LLM-based fault classification, and statistical analysis with visualizations.

3.1 Data Collection

Incident data were collected from the publicly available Verica Open Incident Database (VOID) API [16], which compiles postmortem reports from organizations across various sectors, enabling cross-industry analysis.

The collection was performed using a custom Python scraper [17] developed in collaboration with other researchers to interface with the API. The scraper retrieved structured metadata (e.g., date, organization, duration) and full-text descriptions when available. At the time of access, **10,328 reports** were available.

To ensure data quality and analytical utility, the following filters were applied:

1. Reports without a `description` were excluded.
2. Descriptions shorter than 100 characters were removed.
3. Reports missing duration metadata were discarded.
4. Years with fewer than 50 reports were excluded.

After filtering, **7,804 reports** remained, spanning an **8-year period from 2014 to 2022**. The reduction process is visualized in Figure 1. Organizations publicly disclosed all data as part of transparency or learning-focused initiatives. Use of the official API ensured reliable and compliant access.

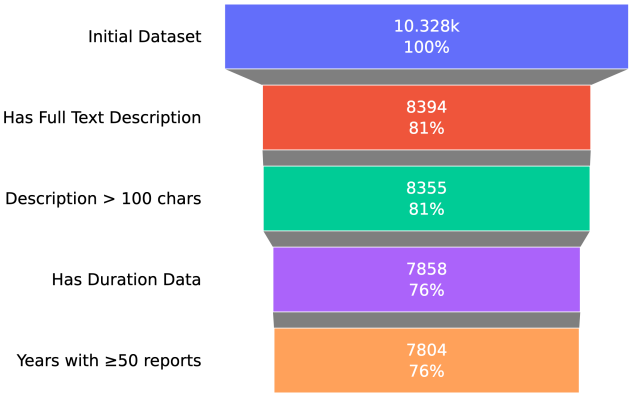

Figure 1: Filtering pipeline for the VOID incident report dataset.

3.2 Incident Classification Using LLMs

Incident report classification used the `Athene V2` model, a fine-tuned variant of `Qwen 2.5 72B` developed by Nexusflow and optimized for long-form log extraction [18]. Inference was performed on the Delft Blue supercomputing infrastructure [19].

Among 70 B-class language models, `Qwen 2.5` and `Llama 3.3` represent state-of-the-art systems, with `Qwen 2.5` consistently outperforming peers across benchmarks [20]. `Athene V2` extends `Qwen 2.5`'s capabilities with fine-tuning for structured classification.

Each report was classified using a single few-shot prompt, which included the full incident text and annotated examples illustrating the fault taxonomy to improve accuracy [21]. The model assigned a single primary fault type based on incident description, timeline, and root cause summary.

Two additional labels—*Unknown* and *Scheduled Maintenance*—handled reports lacking diagnostic detail or describing routine maintenance. Prompt instructions explicitly encouraged selection of *Unknown* when appropriate.

3.3 Model Evaluation Approach

To assess the reliability of the automated classification pipeline, the model's predictions were evaluated against a

manually annotated ground truth sample selected using a stratified sampling approach.

The evaluation assessed classification performance across all fault categories using standard metrics, including accuracy and $F1$ scores. Detailed results of this validation are presented in Section 5.1.

Ground Truth Sampling

The ground truth sample was selected using a statistically rigorous approach designed to provide reliable validation while maintaining cost efficiency. The sample size was determined using the binomial margin-of-error formula:

$$n = \frac{z^2 p(1-p)}{E^2}$$

Using the worst-case variance assumption $p = 0.5$, a $95\,\%$ confidence level ($z = 1.96$), and a desired half-width $E = 0.08$ ($\pm 8\,\%$), we calculated a required sample size of $n = 151$ incidents.

The 151 cases were drawn using a population-proportional stratified sampling design that incorporates two safeguards against skewed class coverage:

- **Per-class minimum:** Every fault category contributes at least $k = 5$ incidents to the validation set (or its entire population if the category contains fewer than 5 incidents).
- **Head-class cap:** The majority category is limited to $\leq 30\,\%$ of the sample (45 incidents maximum), preventing it from crowding out rarer but important classes.

The sampling procedure operates as follows: First, compulsory minimum allocations and the head-class cap are applied. The remaining sample budget is then distributed proportionally to the true class frequencies in the full dataset. Any rounding residue is corrected to ensure the final sample contains exactly 151 incidents. Sampling is performed without replacement within each stratum using a fixed random seed (42) to guarantee reproducibility.

The first author manually labeled the resulting stratified sample of 151 reports according to the predefined fault taxonomy described in Section 5.1.

3.4 Statistical Analysis and Visualization

To analyze incident duration distributions across fault types, the Kruskal-Wallis H test was employed as a non-parametric alternative to ANOVA for data that is not normally distributed.

To evaluate changes in fault type distribution over time, a chi-squared test for trend was used to assess overall shifts from 2014 to 2022. For individual fault types, two methods were employed:

- Generalized Linear Models (GLMs) to detect and quantify linear trends in fault type frequencies, suitable for proportional data with year-to-year sample imbalance.
- Mann–Kendall test as a robust non-parametric method to detect monotonic trends without assuming specific distributions or functional forms.

Visualizations included boxplots for incident duration distributions, bar charts for fault type frequencies, line plots for temporal changes, and confusion matrices for classification evaluation. All analyses used Python with `pandas` [22], `NumPy` [23], `matplotlib` [24], and `seaborn` [25]. The analysis pipeline was implemented in reproducible Jupyter notebooks under Git version control.

3.5 Experimental Setup

All classification tasks were performed on the Delft Blue high-performance computing cluster (HPC) at TU Delft. The `Athene V2` model was executed on two NVIDIA A100 $80\,\mathrm{GB}$ GPUs using HuggingFace's `Transformers` library [26]. To optimize GPU memory usage and performance, model weights were quantized to $8\,\mathrm{bit}$ precision using the `bitsandbytes` library [27].

Before presenting the results, this section addresses the ethical considerations and reproducibility aspects of the study.

4 Ethics of the Research

This section outlines the ethical considerations underlying the study, focusing on responsible research practices, data sourcing, and the use of artificial intelligence. Given the use of publicly shared incident data and advanced AI models, care was taken to ensure transparency, reproducibility, and compliance with ethical standards. The subsections below detail how these principles were applied throughout the research process.

4.1 Responsible Research

Reproducibility: This study is designed to support reproducibility by providing a transparent and detailed description of the data collection, processing, classification, and analysis procedures. All steps from querying the VOID API to filtering incident reports, running LLM-based classification, and generating visualizations are described and utilize open-source tools. The configuration of the HPC environment (Delft Blue cluster) is also documented to enable comparable computational setups.

Replicability: To support replicability, a complete replication package is made available on Zenodo [17], [28]. This package includes:

- All code used for data collection and filtering via the VOID API,
- The cleaned and filtered dataset used in this study ($7{,}804$ reports),
- The ground truth sampling script implementing the stratified sampling procedure
- The manually annotated ground truth sample (151 reports) used for model evaluation,
- Prompt templates and inference scripts for running the LLM-based classification,
- Jupyter notebooks for statistical analysis and visualizations,
- Detailed specifications of the Delft Blue HPC configuration used,

- Environment and dependency specifications (`requirements.txt`)

These resources enable any competent researcher to reproduce the study's findings, from raw data acquisition to final results, using publicly available tools and hardware of similar capacity. While minor variations may occur due to model nondeterminism or updates to the VOID dataset, the core methodology remains stable and is fully documented.

Ethical Web Scraping: No unauthorized web scraping was conducted in this research. All data were obtained via the official VOID API, which is explicitly provided for public and academic use. The study adheres to the terms of use of the VOID platform, and all analyzed data consist of incident reports and postmortems voluntarily shared by organizations as part of their transparency and reliability initiatives. This approach ensures compliance with ethical standards for web data collection.

4.2 Use of Artificial Intelligence (AI):

An LLM was used to categorize incidents into predefined fault types, leveraging its natural language understanding capabilities. However, the interpretation of the classification outputs, including the identification of trends and drawing conclusions, was carried out by the researcher with careful consideration of context, potential biases, and limitations. The AI-generated classifications were validated against a manually annotated subset to ensure reliability. Prompt engineering was used to explicitly encourage appropriate use of the *Unknown* label, counteracting the model's natural tendency to prefer confident predictions even when incident details were insufficient. Thus, while AI contributed significantly to data processing and pattern identification, human oversight and domain knowledge were integral to the interpretation of results and the formulation of conclusions.

With these ethical safeguards and methodological controls in place, the following section presents the results of the analysis, structured around the study's four main research subquestions.

5 Results

Results are presented in four parts addressing the research questions from Section 1: (1) taxonomy development and classification model performance, (2) fault type distribution and frequency, (3) correlation with incident duration, and (4) temporal evolution of fault types.

5.1 RQ1 - Fault Taxonomy Development and Classification Model Performance

This subsection develops a fault taxonomy based on prior research and evaluates the reliability of automated classification for understanding factors that impact system reliability.

Taxonomy of Primary Fault Types

Studies classify information system failures into three categories: technical failures, human and organizational factors, and external disruptions.

Technical causes include infrastructure failures, software bugs, configuration and deployment errors, and capacity issues. Oppenheimer et al. [8] show software bugs as primary contributors to Internet service failures. Misconfigurations and faulty deployments frequently cause incidents in complex cloud environments, where automation can propagate errors at scale.

Human and organizational factors encompass operator mistakes, inadequate procedures, monitoring failures, insufficient training, and organizational culture issues. Dogga et al. [29] found inadequate monitoring most common in Microsoft Azure postmortems, while Saarelainen and Jäntti [30] determined many IT incidents stem from change planning failures rather than technical faults.

External and environmental disruptions include third-party service failures, cyberattacks, and physical infrastructure problems. Lawrence and Simon [31] identify power failures as the most significant cause of major data center outages. Security incidents, including ransomware and DDoS attacks, increasingly disrupt operations [32].

A consolidated taxonomy informed by these studies was developed to facilitate automated LLM analysis. This taxonomy reduces redundancy by merging semantically similar causes, resulting in eight categories that reflect the technical, human, and external factors affecting service reliability. Table 1 presents the complete taxonomy organized by main categories.

Table 1: Condensed Root Causes of IT Incidents

Category	Causes
Technical	(A) Infrastructure Failure (B) Software Bug (C) Misconfiguration/Deployment Failure (D) Capacity Issue
Human/Organizational	(E) Human/Process Error (operator mistakes, inadequate procedures, monitoring/alerting failures, etc.)
External/Environmental	(F) External Dependency Failure (G) Security Incident (H) Environmental Hazard (power, fire, fiber cut, etc.)
Other	(I) Scheduled Maintenance (J) Unknown

Model Evaluation Results

The classification model achieved $92\,\%$ accuracy with a macro-averaged F1-score of 0.89, demonstrating strong performance across fault types.

Performance varied by category: *Security Incidents*, *Scheduled Maintenance*, and *Human/Process Errors* achieved perfect classification ($F1 = 1.00$). *External Dependency Failures* ($F1 = 0.96$) and *Unknown* cases ($F1 = 0.93$) were also handled reliably. *Misconfiguration/Deployment Failures* and *Capacity Issues* showed good performance ($F1 = 0.86$), while *Software Bugs* ($F1 = 0.78$) demonstrated reasonable accuracy despite some misclassifications.

Infrastructure Failures proved most challenging ($F1 = 0.73$), likely due to semantic overlap with other technical categories. The model correctly abstained on *Unknown* cases in 87 % of instances, though some concrete faults were misclassified as *Unknown* due to vague reporting.

Figure 2 shows the confusion matrix, highlighting strong diagonal performance with primary misclassifications occurring between technical fault types and the *Unknown* category. The labels A-J correspond to categories as shown in Table 1.

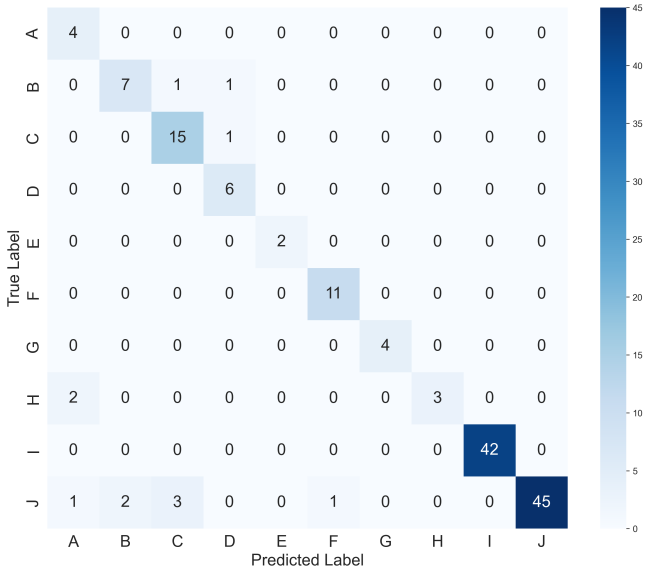

Figure 2: Confusion matrix of automated fault classification vs. manual annotations ($n = 151$).

5.2 RQ2 - Distribution and Frequency of Primary Fault Types

Using the taxonomy, each incident report in the dataset was classified into one primary fault category (Section 3.2). For subsequent statistical analyses, reports classified as *Unknown* or *Scheduled Maintenance* were excluded, as they do not represent actionable fault types. The resulting analysis set consisted of **2,323 reports**. The distribution of these categories across the whole dataset is shown in Figure 3.

5.3 RQ3 - Relationship Between Fault Types and Incident Severity

To assess whether different primary fault types are associated with varying levels of incident severity, this subsection analyzes incident duration as a proxy for severity. Incident duration distributions were examined across fault types, and statistical tests were performed to determine whether observed differences were significant.

Distribution of Incident Duration per Fault Type

Figure 4 shows the distribution of incident durations across primary fault types. Incident durations exhibit substantial variation between fault categories. *Environmental Hazards* display the longest median durations, while *Misconfiguration/Deployment Failures* are generally resolved more

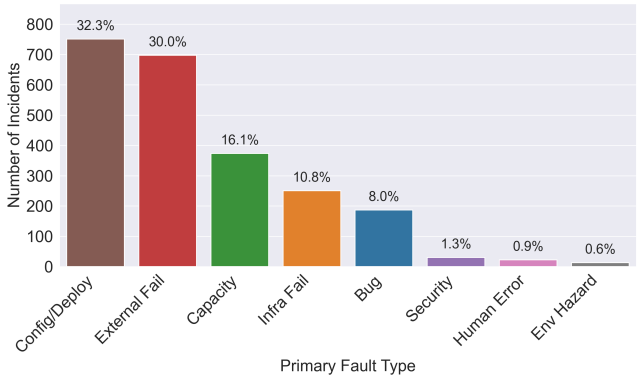

Figure 3: Primary Fault Types – Absolute Frequency

quickly. Table 2 summarizes key statistics for incident duration by fault type.

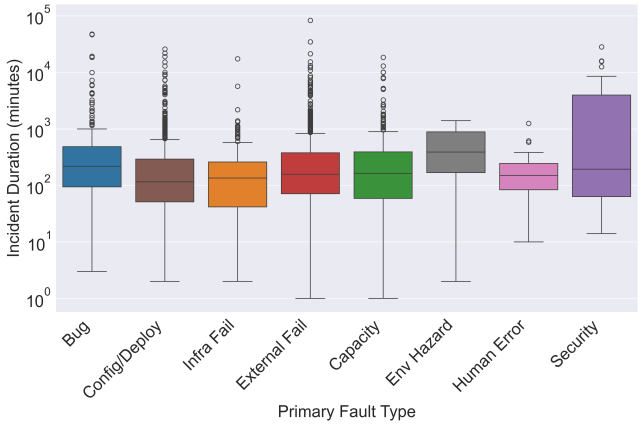

Figure 4: Incident duration distribution per primary fault type (log scale).

Statistical Analysis of Duration Differences

To assess whether differences in incident duration across primary fault types were statistically significant, a Kruskal-Wallis H test was performed. The test revealed significant differences in duration distributions between fault types ($H = 45.16$, $p < 0.00001$). These findings suggest that fault type is an important factor influencing the severity of incidents in terms of time to resolution.

5.4 RQ4 - Temporal Evolution of Fault Type Patterns

The distribution of primary fault types evolves as system architectures and operational practices change. The adoption of cloud-native architectures, container orchestration platforms, and infrastructure-as-code introduced new sources of risk, while automation reduced others. Analyzing temporal trends in fault types offers insight into how these shifts impact system reliability. This subsection examines changes in fault type patterns over time within the incident dataset.

Table 2: Summary of incident duration statistics (in minutes) by primary fault type, including median, mean, and 95th percentile durations.

Primary Fault Type	Median	Mean	95th Percentile
Capacity Issue	164.0	539.9	1510.6
Environmental Hazard	390.0	518.8	1270.6
External Dep. Failure	157.0	722.8	1969.8
Human/Process Error	150.0	229.7	613.4
Infrastructure Failure	135.0	303.0	860.8
Misconfig/Deploy. Failure	116.0	514.1	1504.0
Security Incident	193.0	3761.0	15808.3
Software Bug	217.0	1421.2	4361.4

Temporal Evolution of Primary Fault Types

Figure 5 presents the temporal evolution of primary fault types as a percentage of total incidents, based on raw relative frequencies.

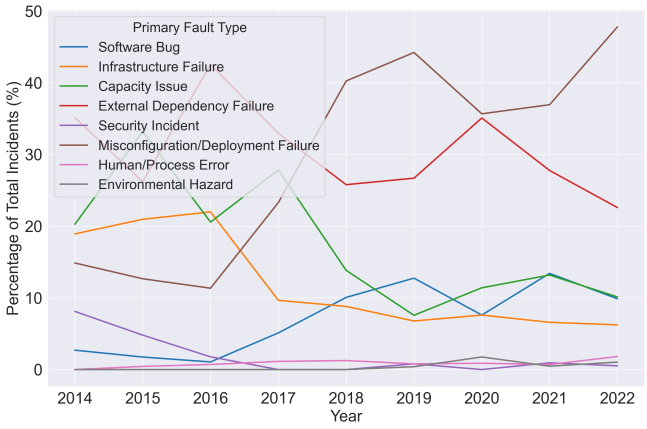

Figure 5: Temporal evolution of primary fault types as a percentage of total incidents (2014–2022).

Trend Analysis

A chi-squared test for trend was used to determine whether the overall distribution of fault types changed significantly across the study period. In addition, for each primary fault type, both GLMs and the Mann–Kendall test were applied to assess trends in their relative frequency.

The chi-squared test confirmed a significant change in the distribution of fault types over the 2014–2022 period ($\chi^2 = 462.02$, $p < 0.00001$, $df = 56$).

Table 3 summarizes the outcomes of the GLM and Mann-Kendall analyses. *Software Bugs*, *Infrastructure Failures*, *Capacity Issues*, *Misconfiguration/Deployment Failures*, and *Environmental Hazards* exhibited significant and consistent trends in both GLM and MK analyses. Of these, the first, fourth, and fifth showed increasing trends, while the second and third showed decreasing ones. *Security Incidents* and *Human/Process Errors* were significant only in GLM, while *External Dependency Failures* showed no significant trend in either method.

These results highlight key patterns in the types, severity, and evolution of primary faults in modern IT systems. The following discussion examines their implications for system reliability and operational practices.

6 Discussion

This study analyzed 7,804 publicly available incident reports spanning 2014–2022 to examine fault types, their operational impact, and temporal evolution in modern IT systems. The findings provide empirical evidence that addresses three critical knowledge gaps in understanding digital system fragility, offering actionable insights for improving operational resilience.

6.1 Fault Type Analysis: Prevalence, Impact, and Evolution

Misconfigurations and Deployment Failures (32.3 %)

Misconfigurations and Deployment Failures emerge as the dominant cause of modern IT incidents, representing nearly one-third of all reported failures. This finding highlights the significant challenges associated with managing complex, automated deployment pipelines and system configurations. The high prevalence likely reflects the trade-offs between deployment speed and reliability that characterize modern DevOps practices, where pressure for rapid deployment cycles can compromise thorough configuration validation.

Despite their high frequency, these incidents show relatively manageable resolution times (median: 116 min, mean: 514 min), likely due to the availability of established rollback procedures and the straightforward nature of reverting configuration changes. However, the temporal analysis reveals a concerning upward trend (GLM slope $= 0.0457$, $p < 0.0001$; MK $\tau = 0.6111$, $p = 0.0286$), presenting a paradox where automation intended to reduce human error may actually be creating new categories of systematic failures. As organizations adopt infrastructure-as-code and automated deployment pipelines, individual configuration errors can be propagated at scale, potentially affecting multiple environments simultaneously [9].

External Dependency Failures (30 %)

External Dependency Failures represent the second most common incident type, aligning with the increasing interconnectedness of modern systems and widespread adoption of microservices architectures and third-party integrations. This finding echoes observations of Beyer et al. [3] regarding the complexity of distributed systems, where cascading failures from external dependencies can rapidly propagate across service boundaries.

These incidents demonstrate considerable variability in resolution complexity, with a mean duration of 723 min but a median of only 157 min, indicating a highly skewed distribution. The 95th percentile of 1,970 min suggests that while many external dependency issues can be resolved relatively quickly, some require extensive coordination with third-party providers or complex workaround implementations. Notably, the temporal analysis reveals no significant trend, suggesting that organizations may have reached a steady state in terms of dependency complexity or that improved dependency management practices are offsetting the increasing system interconnectedness.

Table 3: Summary of trend analysis for all primary fault types.

Fault Type	GLM Slope	GLM p-value	GLM Trend	MK Trend	MK Tau	MK p-value
Software Bug	0.0141	0.0002	increasing	increasing	0.5556	0.0476
Infrastructure Failure	-0.0209	0.0000	decreasing	decreasing	-0.7778	0.0049
Capacity Issue	-0.0232	0.0038	decreasing	decreasing	-0.5556	0.0476
External Dependency Failure	-0.0111	0.1457	no trend	no trend	-0.3333	0.2515
Security Incident	-0.0074	0.0043	decreasing	no trend	-0.4167	0.1363
Misconfiguration/Deployment Failure	0.0457	0.0000	increasing	increasing	0.6111	0.0286
Human/Process Error	0.0013	0.0067	increasing	no trend	0.5000	0.0763
Environmental Hazard	0.0016	0.0094	increasing	increasing	0.6111	0.0155

Capacity Issues (16.1 %)
Capacity Issues account for a substantial portion of incidents and they exhibit considerable resolution complexity (median: 164 min, mean: 540 min), likely reflecting the time required to provision additional resources, optimize system performance, or implement architectural changes to address capacity constraints.

Encouragingly, the temporal analysis reveals a significant decreasing trend (GLM slope $= -0.0232$, $p = 0.0038$; MK $\tau = -0.5556$, $p = 0.0476$), likely reflecting improvements in cloud elasticity and automated scaling capabilities. Modern cloud platforms provide sophisticated auto-scaling mechanisms that can respond to demand fluctuations more effectively than traditional capacity planning approaches.

Infrastructure Failures (10.8 %)
Infrastructure Failures show moderate frequency compared to configuration and external dependency issues, suggesting that cloud providers have achieved reasonable success in abstracting infrastructure complexity away from application developers.

These incidents typically require moderate resolution times (median: 135 min, mean: 303 min). The temporal analysis shows a significant decreasing trend (GLM slope $= -0.0209$, $p < 0.0001$; MK $\tau = -0.7778$, $p = 0.0049$), aligning with the maturation of cloud computing platforms and improved infrastructure automation. Cloud providers have invested heavily in redundant systems, automated failover mechanisms, and predictive maintenance, reducing the frequency of infrastructure-related incidents.

Software Bugs (8 %)
Software Bugs represent a smaller but operationally significant category, characterized by notably long resolution times (median: 217 min, mean: 1,421 min). The extended tail of the distribution (95th percentile: 4,361 min) reflects the complexity of diagnosing and fixing code-level issues in production environments. While some software bugs can be resolved quickly through rollbacks or hotfixes, others require extensive investigation and development efforts.

The temporal analysis reveals a significant increasing trend (GLM slope $= 0.0141$, $p = 0.0002$; MK $\tau = 0.5556$, $p = 0.0476$), likely reflecting several interconnected factors. The acceleration of development cycles associated with agile methodologies and continuous deployment practices may reduce time available for comprehensive testing, leading to more bugs reaching production environments. Additionally, the increasing complexity of distributed systems and the proliferation of programming languages and frameworks may increase the likelihood of code-level failures.

Security Incidents (1.3 %)
Security Incidents, while relatively infrequent, represent the most operationally disruptive fault type, exhibiting exceptionally long resolution times (median: 193 min, mean: 3,761 min, 95th percentile: 15,808 min). This finding aligns with established understanding of security incident complexity, where forensic analysis, containment, and remediation often require extensive coordination across multiple teams and external stakeholders.

The temporal analysis reveals a decreasing trend in frequency (GLM slope = -0.0074, $p = 0.0043$), although this may reflect reporting biases rather than actual security improvements, as organizations may be reluctant to disclose security breaches due to reputational concerns.

Human/Process Errors and Environmental Hazards
Human/Process Errors (0.9 %) and *Environmental Hazards* (0.6 %) represent the least frequent categories. The low *Human/Process Error* contradicts literature emphasizing human factors in system failures [29]. This may reflect reporting biases against attributing public incidents to human error, or indicate that while direct human errors have decreased, they have been transformed into systematic configuration and deployment errors.

Environmental hazards show an increasing trend (GLM slope $= 0.0016$, $p = 0.0094$; MK $\tau = 0.6111$, $p = 0.0155$), though this should be interpreted cautiously given the small absolute numbers involved.

6.2 Implications for Practitioners and Researchers

The dominance of *Misconfiguration/Deployment Failures* suggests that organizations should prioritize investment in configuration management, automated testing of deployment processes, and robust rollback capabilities. The substantial presence of *External Dependency Failures* reinforces that traditional approaches focusing primarily on internal system reliability may be insufficient in today's interconnected ecosystem. Organizations should prioritize dependency management through service-level monitoring, circuit breakers, and graceful degradation strategies.

The correlation between fault types and resolution times suggests that resource allocation should consider both the frequency and complexity of faults. *Security Incidents* require specialized capabilities and extended response times despite being relatively rare, while rising *Software Bugs* highlight the need for strengthened testing, particularly in distributed scenarios.

The findings challenge the assumption that automation necessarily improves system reliability, suggesting the need for theoretical frameworks that account for the complexity trade-offs of DevOps. The methodology demonstrates replicable LLM-based incident classification, while the significant within-category variability in resolution times suggests that current taxonomies may be insufficient for capturing the complexity of modern system failures.

While these findings offer valuable insights, it is essential to consider the study's limitations, which are discussed in the following section.

7 Limitations

This research relies significantly on publicly available incident reports, introducing potential selection bias. Organizations may selectively disclose incidents, particularly avoiding those that are highly sensitive or damaging to their reputation. Consequently, the data may disproportionately represent less severe or better-managed incidents.

The incident classification used the Athene V2 LLM with manual validation on 151 examples. However, classification accuracy is limited by LLM interpretability constraints and the small ground truth dataset, which should be cross-validated with additional annotators. Very small sample sizes in specific categories prevented achieving statistical significance. Ambiguous incident descriptions may cause misclassifications, and despite prompt engineering, the model may under-select the *Unknown* label when details are incomplete.

The Athene V2 model was quantized to 8-bit precision for efficient inference on available HPC resources. At the same time, this is unlikely to affect high-level classification trends; it may introduce minor differences in individual prediction confidence compared to full-precision inference [33].

Temporal analysis assumes continuous and consistent reporting practices and technological contexts over the study period. However, industry practices rapidly evolve, and variations in reporting standards or organizational transparency over time could undermine the longitudinal reliability of the observed trends.

Addressing these limitations offers opportunities for future research, as outlined in the next section.

8 Future Work

To strengthen the quality and applicability of future research in this area, three key improvements could be made:

Broaden the data sources used for analysis: While this study utilized publicly available reports, collaborating with industry partners to access anonymized internal postmortems could capture more severe or complex incidents that organizations don't publicly disclose, thereby making the dataset more comprehensive and reducing potential biases.

Expand the manually annotated dataset used for training and validating the classification model: By increasing the number and diversity of labeled incidents, ensuring they cover a wider range of incident types and descriptions, and cross-validating annotations with multiple expert annotators to improve labeling reliability, the model's performance could be more accurately assessed.

Fine-tune the LLM used for incident classification: Fine-tuning the model on expert-labeled incident reports could improve classification accuracy by adapting it to the specific terminology and patterns in incident descriptions [14]. This supervised learning approach would adjust the model's parameters to handle nuanced or ambiguous cases better, thereby enhancing performance beyond what general pre-trained models can achieve.

The final section summarizes the key findings of this study and their implications for both research and practice.

9 Conclusions

In conclusion, this study analyzed $7{,}804$ publicly available incident reports from 2014–2022 to investigate the dominant fault types in modern IT systems, their relationship to incident severity, and their evolution over time.

Our analysis reveals that *Misconfiguration/Deployment Failures* ($32.3\,\%$) and *External Dependency Failures* ($30\,\%$) have emerged as the primary sources of operational incidents, fundamentally shifting the reliability landscape from traditional infrastructure concerns. This finding challenges conventional approaches to system reliability and highlights the unintended consequences of automation, as well as the complexity of distributed systems.

Statistical analysis revealed significant differences in incident duration across fault types. Notably, *Security Incidents*, though infrequent, exhibited the longest resolution times, indicating their disproportionate operational impact. On the other hand, *Misconfiguration/Deployment Failures*, despite their high frequency, were resolved the quickest, suggesting effective rollback procedures.

The temporal analysis reveals a significant shift in failure patterns, coinciding with the adoption of cloud-native technologies. While *Misconfiguration/Deployment Failures* and *Software Bugs* increased substantially over the study period, *Infrastructure Failures* and *Capacity Issues* declined, reflecting the success of cloud abstraction and auto-scaling technologies.

These findings have important implications for practitioners. Organizations should prioritize investment in configuration management and deployment validation. The prevalence of external dependency failures necessitates new approaches to resilience that extend beyond organizational boundaries, emphasizing circuit breakers, graceful degradation, and comprehensive dependency monitoring.

The study contributes the first large-scale, cross-organizational analysis of IT incident patterns in the cloud-native era, updating foundational reliability taxonomies with empirical evidence. The LLM-based classification approach achieved $91\,\%$ accuracy, demonstrating the viability of automated analysis for large-scale incident data.

Future work should address the limitations of publicly available data through industry partnerships and investigate domain-specific models for incident classification. The reliability challenges of modern IT systems require continued empirical investigation as automation and system complexity continue to evolve.

References

[1] R. Alt, J. M. Leimeister, T. Priemuth, S. Sachse, N. Urbach, and N. Wunderlich, "Software-defined business," *Business & Information Systems Engineering*, vol. 62, no. 6, pp. 609–621, Dec. 1, 2020, ISSN: 1867-0202. DOI: 10.1007/s12599-020-00669-6. [Online]. Available: https://doi.org/10.1007/s12599-020-00669-6 (visited on 04/23/2025).

[2] "ISO/IEC/IEEE international standard - systems and software engineering–vocabulary," *ISO/IEC/IEEE 24765:2017(E)*, pp. 1–541, Aug. 2017. DOI: 10.1109/IEEESTD.2017.8016712. [Online]. Available: https://ieeexplore.ieee.org/document/8016712 (visited on 04/23/2025).

[3] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, *Site reliability engineering: How Google runs production systems*, 1 online resource vols. Sebastopol, CA: O'Reilly Media, 2016, ISBN: 978-1-4919-5118-7 978-1-4919-5117-0. [Online]. Available: http://www.dawsonera.com/depp/reader/protected/external/AbstractView/S9781491951187 (visited on 04/23/2025).

[4] A. A. Hinai and M. A. Mazroui, "Optimizing and enhancing IT operation operating models through artificial intelligence," in *2024 2nd International Conference on Computing and Data Analytics (ICCDA)*, Nov. 2024, pp. 1–5. DOI: 10.1109/ICCDA64887.2024.10867366. [Online]. Available: https://ieeexplore.ieee.org/document/10867366 (visited on 04/23/2025).

[5] L. Rijal, R. Colomo-Palacios, and M. Sánchez-Gordón, "AIOps: A multivocal literature review," in *Artificial Intelligence for Cloud and Edge Computing*, ISSN: 2199-1081, Springer, Cham, 2022, pp. 31–50, ISBN: 978-3-030-80821-1. DOI: 10.1007/978-3-030-80821-1_2. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-80821-1_2 (visited on 04/23/2025).

[6] E. Kapel, L. Cruz, D. Spinellis, and A. van Deursen, "Enhancing incident management: Insights from a case study at ING," in *Proceedings of the 1st IEEE/ACM Workshop on Software Engineering Challenges in Financial Firms*, ser. FinanSE '24, New York, NY, USA: Association for Computing Machinery, Aug. 9, 2024, pp. 1–8, ISBN: 979-8-4007-0568-7. DOI: 10.1145/3643665.3648048. [Online]. Available: https://dl.acm.org/doi/10.1145/3643665.3648048 (visited on 04/23/2025).

[7] E. Kapel, L. Cruz, D. Spinellis, and A. Van Deursen, "On the difficulty of identifying incident-inducing changes," in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, ser. ICSE-SEIP '24, New York, NY, USA: Association for Computing Machinery, May 31, 2024, pp. 36–46, ISBN: 979-8-4007-0501-4. DOI: 10.1145/3639477.3639755. [Online]. Available: https://dl.acm.org/doi/10.1145/3639477.3639755 (visited on 04/23/2025).

[8] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do internet services fail, and what can be done about it?," presented at the 4th USENIX Symposium on Internet Technologies and Systems (USITS 03), 2003. [Online]. Available: https://www.usenix.org/conference/usits-03/why-do-internet-services-fail-and-what-can-be-done-about-it (visited on 04/23/2025).

[9] P. Huang, "Toward understanding and dealing with failures in cloud-scale systems," Ph.D. dissertation, UC San Diego, 2016. [Online]. Available: https://escholarship.org/uc/item/6nn0x49p (visited on 06/21/2025).

[10] F. Menges and G. Pernul, "A comparative analysis of incident reporting formats," *Computers & Security*, vol. 73, pp. 87–101, Mar. 1, 2018, ISSN: 0167-4048. DOI: 10.1016/j.cose.2017.10.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404817302250 (visited on 06/21/2025).

[11] I. J. B. Young, S. Luz, and N. Lone, "A systematic review of natural language processing for classification tasks in the field of incident reporting and adverse event analysis," *International Journal of Medical Informatics*, vol. 132, p. 103 971, Dec. 1, 2019, ISSN: 1386-5056. DOI: 10.1016/j.ijmedinf.2019.103971. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1386505619302370 (visited on 06/02/2025).

[12] A. Saha and S. C. H. Hoi, "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*, ser. ICSE-SEIP '22, New York, NY, USA: Association for Computing Machinery, 2022, pp. 197–206, ISBN: 978-1-4503-9226-6. DOI: 10.1145/3510457.3513030. [Online]. Available: https://dl.acm.org/doi/10.1145/3510457.3513030 (visited on 06/21/2025).

[13] Y. Chen, H. Xie, M. Ma, *et al.*, "Automatic root cause analysis via large language models for cloud incidents," in *Proceedings of the Nineteenth European Conference on Computer Systems*, ser. EuroSys '24, New York, NY, USA: Association for Computing Machinery, 2024, pp. 674–688, ISBN: 979-8-4007-0437-6. DOI: 10.1145/3627703.3629553. [Online]. Available: https://dl.acm.org/doi/10.1145/3627703.3629553 (visited on 06/21/2025).

[14] T. Ahmed, S. Ghosh, C. Bansal, T. Zimmermann, X. Zhang, and S. Rajmohan, "Recommending root-cause and mitigation steps for cloud incidents using large language models," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, ISSN: 1558-1225, May 2023, pp. 1737–1749. DOI: 10.1109/ICSE48619.2023.00149. [Online]. Available: https://ieeexplore.ieee.org/document/10172904 (visited on 05/14/2025).

[15] D. Anandayuvaraj, M. Campbell, A. Tewari, and J. C. Davis, "FAIL: Analyzing software failures from

the news using LLMs," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '24, New York, NY, USA: Association for Computing Machinery, Oct. 27, 2024, pp. 506–518, ISBN: 979-8-4007-1248-7. DOI: 10.1145/3691620.3695022. [Online]. Available: https://dl.acm.org/doi/10.1145/3691620.3695022 (visited on 05/07/2025).

[16] "The verica open incident database," VOID. (), [Online]. Available: https://www.thevoid.community/database (visited on 04/27/2025).

[17] I. Aldea, M. Georgiev, J. Rutkowski, A. Mureşan, and D. Bunschoten, *What can we learn from incident reports? - web scraper*, Jun. 2025. DOI: 10.5281/zenodo.15711606. [Online]. Available: https://doi.org/10.5281/zenodo.15711606.

[18] "Nexusflow/athene-v2-chat · hugging face." (Nov. 14, 2024), [Online]. Available: https://huggingface.co/Nexusflow/Athene-V2-Chat (visited on 06/16/2025).

[19] D. H. P. C. Centre (DHPC), *DelftBlue supercomputer (phase 2)*, 2024. [Online]. Available: https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2.

[20] J. Wu, B. Gu, R. Zhou, *et al.*, *BRIDGE: Benchmarking large language models for understanding real-world clinical practice text*, May 1, 2025. DOI: 10.48550/arXiv.2504.19467. arXiv: 2504.19467[cs]. [Online]. Available: http://arxiv.org/abs/2504.19467 (visited on 06/10/2025).

[21] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (visited on 06/16/2025).

[22] T. p. d. team, *Pandas-dev/pandas: Pandas*, version latest, Feb. 2020. DOI: 10.5281/zenodo.3509134. [Online]. Available: https://doi.org/10.5281/zenodo.3509134.

[23] C. R. Harris, K. J. Millman, S. J. v. d. Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, Publisher: Springer Science and Business Media LLC. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[24] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007, Publisher: IEEE COMPUTER SOC. DOI: 10.1109/MCSE.2007.55.

[25] M. L. Waskom, "Seaborn: Statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021, Publisher: The Open Journal. DOI: 10.21105/joss.03021. [Online]. Available: https://doi.org/10.21105/joss.03021.

[26] T. Wolf, L. Debut, V. Sanh, *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[27] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "LLM.int8(): 8-bit matrix multiplication for transformers at scale," *arXiv preprint arXiv:2208.07339*, 2022.

[28] J. Rutkowski, *Understanding IT system failures: Fault types, severity patterns, and evolution in modern operations*, Jun. 2025. DOI: 10.5281/zenodo.15675301. [Online]. Available: https://doi.org/10.5281/zenodo.15675301.

[29] P. Dogga, C. Bansal, R. Costleigh, G. Jayagopal, S. Nath, and X. Zhang, "{AutoARTS}: Taxonomy, insights and tools for root cause labelling of incidents in microsoft azure," presented at the 2023 USENIX Annual Technical Conference (USENIX ATC 23), 2023, pp. 359–372, ISBN: 978-1-939133-35-9. [Online]. Available: https://www.usenix.org/conference/atc23/presentation/dogga (visited on 05/20/2025).

[30] K. Saarelainen and M. Jäntti, "Quality and human errors in IT service infrastructures — human error based root causes of incidents and their categorization," in *2015 11th International Conference on Innovations in Information Technology (IIT)*, Nov. 2015, pp. 207–212. DOI: 10.1109/INNOVATIONS.2015.7381541. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7381541 (visited on 05/20/2025).

[31] A. Lawrence and L. Simon, "Annual outages analysis 2023: The causes and impacts of IT and data center outages," Uptime Institute, Keynote Report, Mar. 2023.

[32] J. Buffington, J. Webb, and D. Russell, "Data protection trends report 2023," Veeam Software, Keynote Report, Jan. 2023.

[33] S. Li, X. Ning, L. Wang, *et al.*, *Evaluating quantized large language models*, Jun. 6, 2024. DOI: 10.48550/arXiv.2402.18158. arXiv: 2402.18158[cs]. [Online]. Available: http://arxiv.org/abs/2402.18158 (visited on 06/22/2025).

A AI Assistance Disclosure

AI tools were employed to assist in drafting portions of this paper, including generating initial outlines, refining text for clarity and readability, and suggesting alternative phrasings. These AI-generated drafts were always reviewed and edited by the researcher to ensure correctness and adherence to academic standards. The use of AI for writing was a means to enhance productivity and consistency rather than a substitute for critical thinking and domain expertise.