The American Journal of Interdisciplinary Innovations and Research
ISSN 2642-7478 Volume 08 - 2026

LLM-Based Intelligent Fault Detection and Self-Healing Framework for Microservices

Vivek Arora

Senior Software Engineer Wawa Inc

Received: 11 May 2026 | Received Revised Version: 24 May 2026 | Accepted: 16 June 2026 | Published: 26 June 2026

Volume 08 Issue 06 2026 | DOI: 10.37547/tajiir/Volume08Issue06-06

Abstract

Microservice-based cloud applications have become the preferred architectural style for modern digital services due to their scalability, flexibility, and ease of deployment. However, the distributed nature of microservices introduces significant operational challenges, including service crashes, network failures, resource exhaustion, configuration anomalies, and dependency-related faults. These failures can propagate rapidly across interconnected services, resulting in performance degradation, service interruptions, and increased operational costs. Traditional fault management strategies are mainly dependent on predefined rules and manual action, which are not suitable for dynamic cloud environments. In view of these challenges, this study puts forward an intelligent cloud fault management framework utilizing Llama 3.1 for automatic fault detection, fault root cause analysis, and self-healing. The framework combines real-time monitoring with anomaly detection, contextual reasoning and autonomous remediation to enhance the operational resilience. Experimental evaluation demonstrates substantial improvements over conventional and semi-automated approaches, achieving 97.5% fault detection accuracy, 96.4% root cause identification accuracy, and a 95.7% self-healing success rate while reducing Mean Time to Recovery to 5.4 minutes. The findings demonstrate the capabilities of LLM to support scalable, reliable, and autonomous cloud fault managements.

Keywords: Microservices, Cloud Computing, Fault Management, Fault Detection, Root Cause Analysis, Self-Healing Systems, Large Language Models (LLMs).

Cite This Article: Arora, V. (2026). LLM-Based Intelligent Fault Detection and Self-Healing Framework for Microservices. The American Journal of Interdisciplinary Innovations and Research, 8(06), 88–99. https://doi.org/10.37547/tajiir/Volume08Issue06-06

I. INTRODUCTION

Microservices have transformed the design and deployment of modern cloud-native applications by enabling large systems to be decomposed into smaller, independent, and loosely coupled services [1][2]. This architectural approach offers several advantages, including scalability, flexibility, fault isolation, and continuous deployment capabilities [3]. Organizations increasingly adopt microservice architectures to support rapidly evolving business requirements and deliver highly responsive digital services [4]. Despite these benefits, the distributed nature of microservices significantly increases operational complexity because numerous services must

communicate, coordinate, and function seamlessly across diverse cloud environments.

As microservice ecosystems grow, they become vulnerable to a wide range of faults and failures. Service crashes, resource exhaustion, network disruptions, configuration inconsistencies, and dependency failures frequently occur due to the dynamic characteristics of cloud infrastructures [5][6]. These faults can propagate across interconnected services, leading to cascading failures that impact application availability and user experience [7]. The increasing complexity of service interactions creates additional security vulnerabilities and operational risks that are difficult to identify and manage using conventional monitoring tools [8]. Traditional fault management solutions primarily depend on predefined rules, static thresholds, and manual troubleshooting processes, which often result in delayed fault detection and prolonged recovery times. Consequently, ensuring reliability and resilience has become a critical concern in microservice-based cloud environments.

To overcome these challenges, self-healing systems have emerged as a promising approach for automating fault detection, diagnosis, and recovery. Self-healing mechanisms continuously monitor system behavior, identify abnormal conditions, and execute corrective actions without requiring extensive human intervention [9][10]. Recent advances in Artificial Intelligence have further enhanced self-healing capabilities by enabling intelligent analysis of operational data and adaptive decision-making [11][12]. In recent times, Large Language Models or LLMs have shown outstanding abilities in reasoning, understanding context, identifying patterns, and integrating knowledge in various fields [13][14]. Conventional machine learning models are usually trained for specific tasks, whereas LLMs can learn from the logs, analyze system events, synthesize data from various sources, and provide valuable explanations for intricate operational challenges [15][16]. By comprehending context and enabling intelligent decision making, they have enabled autonomous cloud operations, fault diagnosis, incident management and reliability improvements in next generation cloud computing environments.

A. Motivation and Contributions of the Study

The growing complexity of cloud infrastructures has made traditional fault management approaches increasingly ineffective in ensuring high service reliability and availability. Manual diagnosis and rule-based remediation often result in delayed responses, increased downtime, and operational inefficiencies. As cloud environments continue to scale, there is a strong need for intelligent solutions capable of understanding system behavior, identifying root causes, and executing recovery actions autonomously. The emergence of advanced Large Language Models provides an opportunity to enhance cloud fault management through contextual reasoning, automated decision-making, and self-healing capabilities, motivating the development of the proposed framework. The key contributions of the study are given:

- Proposes an intelligent cloud fault management framework based on Llama 3.1.
- Integrates real-time monitoring, fault detection, root cause analysis, and self-healing mechanisms.
- Reduces Mean Time to Recovery (MTTR) through autonomous remediation strategies.
- Improves fault detection accuracy and service availability compared to traditional approaches.
- Evaluates framework performance across multiple fault categories, including service crashes, network failures, and security-related faults.

B. Justification and Novelty

The novelty of this work lies in leveraging the reasoning and contextual understanding capabilities of Llama 3.1 for intelligent cloud fault management. Unlike traditional rule-based and semi-automated approaches, the proposed framework combines fault detection, root cause identification, and self-healing within a unified AI-driven architecture. The framework not only identifies failures with high accuracy but also recommends and executes corrective actions autonomously. The integration of LLM intelligence into cloud operations is significant progress towards self-managed cloud environments, improves system reliability, reduces downtime, and reduces the reliance on human interaction in complex operational environments.

C. Organization of the paper

Here is how the rest of the paper is structured. Section II provides a review of the literature on the topic of

microservice fault management. Section III describes the proposed Llama 3.1-driven fault management framework and its architecture. The performance outcomes and comparison analysis are shown in Section IV. The paper's conclusion and future study directions are highlighted in Section V.

II. Literature Review

The papers reviewed indicate improvement in integrated use of intelligent, scalable microservice-specific frameworks, but limited attention to the utilization of AI tools such as ML, RL, or LLMs in fault recovery, anomaly detection and self-healing. Several recent studies are examined:

Chen et al. (2026) introduce GRACE, an integrated system for fault recovery that maximizes the utilization of LLMs, dynamic graph modelling, and reinforcement learning (RL). GRACE describes the system in real-time using graph neural networks, yielding a dynamic graph that represents the relationship and status of resources and microservices. In this context, RL agents can readily learn strategies for common failures, and the LLM module can tune parameters and apply contextual reasoning to improve strategies when resource-related failures become more complicated. Long-term experiments have demonstrated that GRACE performs best and scales most efficiently in real-life microservices environments in the context of complex scenarios [17].

Wang et al. (2026) introduces a new paradigm to manage cloud-native observability with the help of LLMs, enabling automated root cause analysis and self healing. We integrate the Open Telemetry-based telemetry collection with the domain adapted LLM that can perform multimodal analysis of metrics, logs and traces. The LLM's fine-tuning on operational data and chain-of-thought reasoning results in explainable root cause hypotheses and actionable remediation plans. Experimental testing on public microservice datasets shows that our method can decrease mean time to resolution (MTTR) by 84.2% than rule-based methods, and achieve an anomaly detection accuracy of 95% with low computational overhead. 91% of common incidents were automatically addressed without human involvement, resulting in improved service reliability and reduced workload [18].

Sawalkar et al. (2025) discusses automated remediation in the cloud, anomaly detection with ML and predictive failure analytics. Some of the AI methods include DL, graph-based analysis and RL, which enable intelligent fault recovery and real-time anomaly identification. Moreover, cloud-native orchestration technologies like Kubernetes and Terraform play a crucial role in automating infrastructure management and problem resolution. They test the effectiveness of AI-based self-healing methods to maximize system uptime, minimize MTTD and MTTR with empirical studies and case studies [19].

Avgerinos et al. (2025) propose a closed loop framework for integrating LLM agents to perform automated RCA and mitigation of faults in cloud–edge and IoT systems. The agent, when it sees service-level agreement (SLA) violations, determines what are the likely root causes and then applies corrective actions like pod rescheduling, pod scaling or configuration updates through a model context protocol (MCP) server that exposes management tool functionalities through an application programming interface (API). The use of this RCA-plus-mitigation loop allows for explainable and adaptive fault handling. The evaluate our system on a cluster running synthetic IoT workloads under emulated stressors using a reproducible benchmarking setup. Results show that the agent identifies SLA violations with 52.9% accuracy and mitigates 70.7% of them successfully [20].

Arulappan et al. (2024) uses a DRL strategy to examine the issue of a Virtual Machine (VM) that the VNF has assigned to its adaptive self-healing. The big data analytics and collecting engine that relies on DRL aggregates data in order to investigate and analyze it for the purposes of performance management and troubleshooting. Scaling or relocating VNFs are examples of self-healing remedial actions that this engine may assist identify. Therefore, the virtualized infrastructure manager's recommended method for self-healing VNFs is based on DQNs and DQL. Using the stochastic gradient descent approach for VNF service assurance and network stability, virtual network probes of closed-loop orchestration automate the VNF and provide analytics for real-time, policy-driven orchestration on an open networking automation platform. The suggested DQN/DDQN technique improves pricing and reduces resource use costs by 18% without compromising the VNF's QoS. When compared to other cutting-edge methods, the results of adaptive self-healing of VNFs improve computing performance by 27% [21].

Diego (2023) adopts a conceptual-analytical approach, supported by a literature review of 10 foundational papers

published. It presents frameworks for self-healing through intelligent monitoring and autonomous decision-making using supervised and reinforcement learning. Self-healing microservices can autonomously detect, predict, and resolve failures using AI models trained on telemetry, logs, and behavioural patterns. These models outperform rule-based systems in adaptability, especially in cloud-native environments. It helps to lower maintenance expenses, minimize human intervention and improves availability. It is particularly beneficial for large-scale distributed systems with dynamic workloads [22].

Table I presents a summary of methodologies, findings, benefits, limitations, and future directions of the existing studies, where gaps in research motivate an LLM-based intelligent fault detection and self-healing framework for microservices.

TABLE I. RESEARCH GAP ANALYSIS OF EXISTING STUDIES ON INTELLIGENT FAULT DETECTION AND SELF-HEALING IN MICROSERVICES

Author	Methodology	Key Findings	Advantages	Limitation	Future Work
Chen et al. (2026)	Proposed GRACE integrating GNN, RL, and LLMs for fault recovery in microservices.	Improved recovery efficiency and success rate in complex failures.	Hybrid intelligent recovery with dynamic system modeling.	Focuses more on fault recovery than fault detection; limited explainability.	Integrate proactive detection and explainable self-healing.
Wang et al. (2026)	Used LLMs with observability tools for anomaly detection and root cause analysis.	Reduced MTTR by 84.2% and automated 91% of incidents.	Strong multimodal analysis of logs, metrics, and traces.	Limited focus on microservice-specific self-healing and scalability.	Develop adaptive LLM-based self-healing for microservices.
Sawalkar et al. (2025)	Applied ML, RL, and orchestration tools for anomaly detection and remediation in cloud systems.	Reduced MTTD and MTTR, improving uptime.	AI-driven automation with cloud-native orchestration.	No LLM-based reasoning for intelligent fault diagnosis.	Add LLM intelligence for automated remediation.
Avgerinos et al. (2025)	Proposed LLM agents for RCA and fault mitigation in cloud-edge/IoT systems.	Achieved 70.7% mitigation success.	Explainable and adaptive fault management.	Limited accuracy; not focused on microservice environments.	Improve accuracy and adapt to microservices.
Arulappan et al. (2024)	Used DRL (DQN/DDQN) for self-healing in virtualized networks (VNF).	Reduced costs by 18% and improved performance by 27%.	Efficient resource optimization and QoS support.	Lacks LLM-based intelligence and microservice focus.	Extend to microservices with LLM integration.
Diego (2023)	Literature-based study on AI-enabled self-healing microservices.	AI improves adaptability and reduces manual effort.	Highlights benefits of autonomous recovery.	Conceptual work with limited practical validation.	Build experimentally validated LLM-based frameworks.

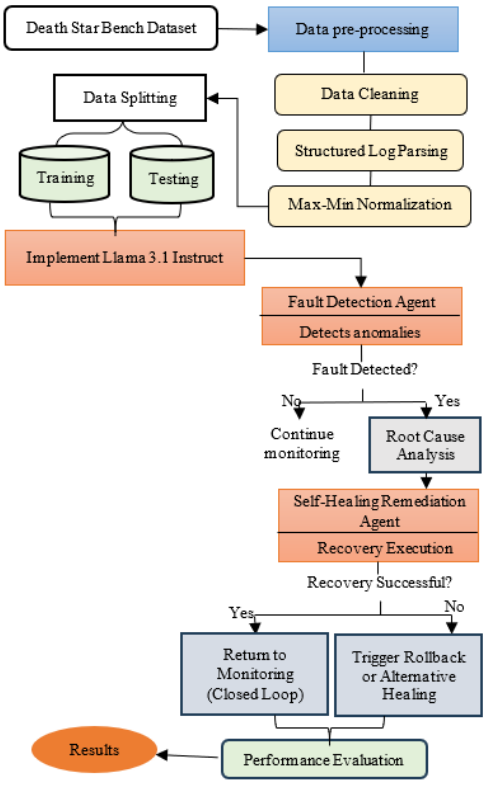

Fig. 1. Methodology Flowchart of the Proposed LLM-Based Intelligent Fault Detection and Self-Healing Framework for Microservices

III. METHODOLOGY

The proposed LLM-Based Intelligent Fault Detection and Self-Healing Framework for Microservices workflow is illustrated in Figure. 1. The telemetry data is collected from the Death Star Bench dataset, which is subsequently pre-processed to enhance data quality and consistency through data cleaning, structured log parsing, and Max-Min normalization. The result data is then split into training and testing data for evaluation. Then, the Llama 3.1 Instruct model is used as the Core Reasoning Engine for the framework. The Fault Detection Agent is constantly monitoring the telemetry data and detects anomalies and abnormal system behavior. In absence of fault, the system keeps monitoring. Otherwise the detected fault is passed to the Root Cause Analysis module for the fault diagnosis. The Self-Healing Remediation Agent performs the correct recovery procedures based on the cause that was identified. Recovery is then validated, with a successful recovery moving the system back to monitoring mode and unsuccessful recovery moving the system back to monitoring mode or alternative healing mechanisms. Lastly, the framework is tested with several performance metrics, and the experimental results are analyzed to investigate the performance of the framework in terms of the accuracy of fault detection, the efficiency of fault recovery, and the reliability of service.

A. Data Collection

The study utilizes the Death Star Bench microservices benchmark dataset, a widely recognized benchmark for evaluating cloud-native distributed systems. The dataset contains approximately 4.8 million runtime telemetry records from a set of microservices containing service logs, CPU usage, memory usage, request counts, response latency, API call interactions, service dependency interactions and more. It also enables for controlled fault-injection scenarios like Service Crashes, network delays, Resource Exhaustion and cascading failures, that make it suitable for testing intelligent fault detection and autonomous self-healing frameworks.

B. Data Preprocessing

Data preparation is a crucial step for the development and application of models. Therefore, the model will have a better chance of finding patterns in the data if the data is clean, consistent, and well-structured. The pre-processing phase involved the following steps:

- **Missing Value Handling:** The gaps in the measurements and missing telemetry data are identified and eliminated to minimize inconsistencies in the measurements during runtime.
- **Duplicate Record Removal:** Any duplicate telemetry records and repeated monitoring logs are removed to enhance the quality of the dataset and prevent duplicated fault patterns.
- **Noise Filtering:** The unnecessary monitoring signals are filtered, and the logs and unwanted runtime records are filtered to enhance the clarity of the fault signals.
- **Structured Log Parsing:** The runtime logs are tokenized to extract useful information such as time-stamps, service identifiers, exception messages, warning logs, and latency irregularities.

C. Max-Min Normalization

To ensure that the features with higher values are not more influential than those with lower values, normalization is crucial [23]. For this research the values of each characteristic are used in the range of 0 to 1 with the help of min-max normalisation technique. This method can be expressed as Equation (1):

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

X represents the present value of the data feature, whereas X_{min} and X_{max} stand for its minimum and maximum values, respectively.

D. Data Splitting

The data set was partitioned into two groups (80% training set and 20% testing set) in order to investigate the effectiveness of the model for training and estimating the parameters and testing and evaluating the performance of the model.

E. Proposed Llama 3.1 Instruct model

The proposed framework involves the usage of the intelligent LLM, Llama 3.1 Instruct (70B), as the core LLM for intelligent fault detection, contextual reasoning, root cause diagnosis, and autonomous remediation planning in microservices environments. The model is embedded in a multi-agent architecture to allow real-time monitoring and self-healing capabilities in cloud-native systems based on Kubernetes. Llama 3.1 Instruct (70B) is chosen because of its ability to handle a wide range of operational data, process long contexts, and exhibit exceptional reasoning capabilities for structured and unstructured data. In contrast to rule-based monitoring systems, the proposed monitoring system infers run-time relationships of services, dependency, anomalies in latency, and service logs to generate intelligent recovery actions dynamically.

F. Implementation Procedure of the Self-Healing Framework

The proposed LLM Based Intelligent Fault Detection and Self-Healing Framework is a closed loop process that automatically observes microservices, detects faults, analyses their root causes, generates remediation plans, and verifies their recovery.

- **Fault Detection:** The Fault Detection Agent continuously analyzes pre-processed telemetry data to identify anomalies such as CPU spikes, memory saturation, increased latency, service failures, and timeout events. When the anomaly score exceeds a predefined threshold, the event is classified as a fault and forwarded for diagnosis.
- **Root Cause Analysis:** The Root Cause Analysis Agent investigates the detected fault using service dependencies, runtime logs, and operational metrics. The agent identifies the underlying cause of failure, including resource exhaustion, network disruptions, service crashes, configuration errors, or dependency-related faults.
- **LLM-Based Contextual Reasoning:** Llama 3.1 Instruct (70B) is given the detected fault context to perform intelligent reasoning. The model uses telemetry data, fault history patterns and

dependency relationships to determine severity of the fault and provide relevant remediation suggestions.

- **Autonomous Self-Healing Execution:** The Self-Healing Remediation Agent takes action based on the LLM's recommendations, including actions like restarting containers, rescheduling pods, scaling resources, rolling back deployments, and reconfiguring services, to restore stability to the system.

- **Recovery Validation and Feedback Loop:** The Validation and Rollback Agent measures recovery effectiveness based on metrics for service availability, latency and resource usage. When a successful recovery takes place, the system is returned to monitoring mode, and if a successful recovery fails, the system automatically rolls back or takes other recovery measures, while maintaining continuous self-healing feedback.

G. Performance Metrics

Evaluation metrics play an important role in ML to assess the performance of the models. Gives a numerical evaluation of the performance of the model. The metrics are formulated in Equation (2) to (6):

1) Fault Detection Accuracy (FDA)

Fault Detection Accuracy defines the correct identification of occurrence of fault in the microservices environment by the framework. The higher the FDA value, the more reliable the fault detection performance.

$$FDA = \frac{TP+TN}{TP+TN+FP+FN} \times 100$$

2) Root Cause Identification Accuracy (RCIA)

The Root Cause Identification Accuracy test assesses the ability of the framework to accurately identify the true cause of the identified faults. The higher the value the more precisely the fault is diagnosed.

$$RCIA = \frac{N_{correct}}{N_{total}} \times 100$$

3) Self-Healing Success Rate (SHSR)

Self-Healing Success Rate is a percentage of faults that are successfully healed because of self-healing actions. The greater the SHSR, the more effective the self-healing ability.

$$SHSR = \frac{N_{resolved}}{N_{detected}} \times 100$$

4) Mean Time to Recovery (MTTR)

MTTR is considered the mean time to restore a normal service operation following the occurrence of a fault. The lower MTTR values the quicker the recovery performance.

$$MTTR = \frac{\sum_{i=1}^{n}(T_{recovery,i} - T_{failure,i})}{n}$$

5) False Positive Rate (FPR)

The False Positive Rate is the percentage of normal events that are incorrectly identified as faults. The lower the FPR results the more accurate the anomaly detection.

$$FPR = \frac{FP}{FP+TN} \times 100$$

These are evaluation measures which provide a detailed insight into the capability of the model for making predictive accuracy.

IV. Result Analysis and Discussion

The experiments were run on an AMD Ryzen 9 7950X processor with 128 GB of DDR5 memory and an NVIDIA A100 GPU having 80 GB of VRAM. It consisted of Ubuntu 22.04 LTS, Python 3.11, Docker 25.0, Kubernetes v1.30, Prometheus, Grafana, Jaeger, PyTorch 2.3 and Transformers library. This setup was adequate to deploy and test the recommended framework. Table II shows that the proposed framework based on Llama 3.1 model outperforms other conventional rule-based and semi-automated approaches. The results demonstrate very notable improvements with fault detection, root cause analysis and self-healing effectiveness. Moreover, the suggested framework achieves minimal manual effort, highest service availability, and the lowest MTTR with the lowest false-positive rate, which is proof of the worth of autonomous microservices management.

TABLE II. COMPARATIVE PERFORMANCE EVALUATION OF FAULT DETECTION AND SELF-HEALING FRAMEWORKS

Metric	Rule-Based System	Semi-Automated System	Proposed Llama 3.1 Framework
Fault Detection Accuracy (%)	82.4	89.6	97.5
Root Cause Identification Accuracy (%)	76.8	86.9	96.4
Self-Healing Success Rate (%)	74.6	87.3	95.7
Mean Time to Recovery (MTTR) (Minutes)	44.2	25.1	5.4
False Positive Rate (%)	14.5	8.3	2.8
Service Availability (%)	90.4	95.3	99.2
Manual Intervention Required (%)	68.7	34.8	6.5

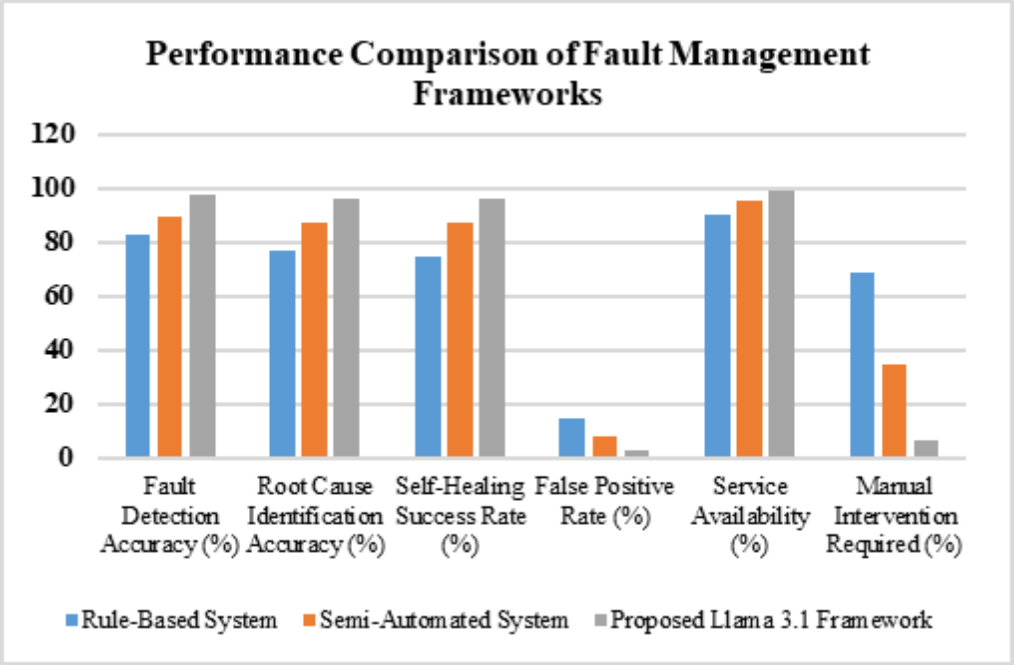

Fig. 2. Performance Comparison of Fault Management Frameworks

Figure 2 presents a comparative study of the Rule Based System, Semi-Automated System and proposed Llama 3.1 Framework according to various fault management parameters. The proposed framework has maximum accuracy in fault detection, root cause identification, success rate in self-healing and service availability throughout. Besides, it drastically lowers false positive and manual handling necessity which proves better automation, reliability and operational efficiency for cloud fault management environment.

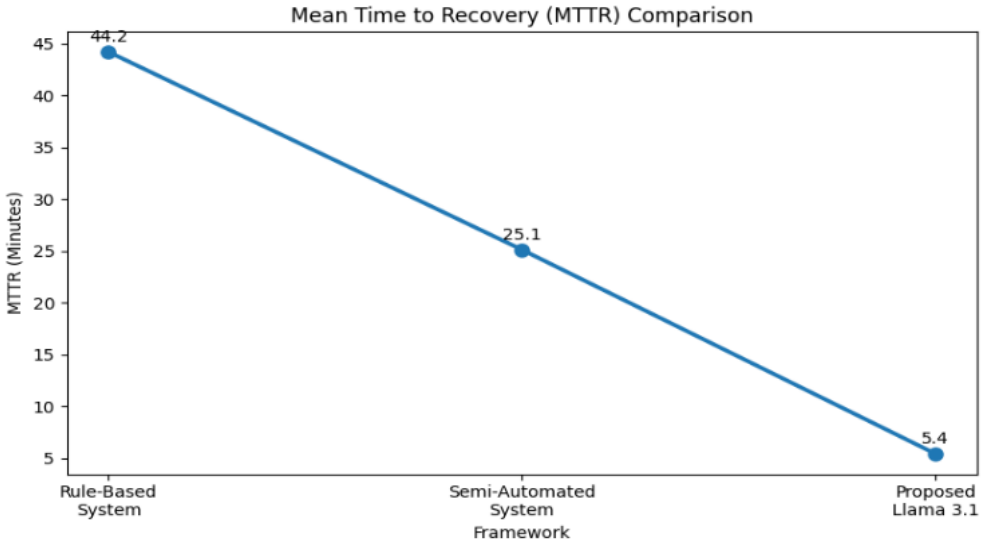

Fig. 3. Mean Time to Recovery (MTTR) Comparison of Fault Management Frameworks

Figure 3 shows the MTTR achieved with the three approaches for fault management. The highest recovery time for the Rule Based System is recorded to be 44.2 minutes while that of the Semi-Automated System is 25.1 minutes. The proposed Llama 3.1 Framework boasts the fastest MTTR of 5.4 minutes, signifying swift fault diagnosis and recovery, which can help to reduce service disruptions and enhance overall system resilience.

The accuracy for detecting the faults obtained by the proposed framework for various categories of faults are reported in Table III. The results indicate good detection performance for various kinds of service crashes, resource exhaustion, network failures, configuration anomalies, dependency failures and security related faults. The framework shows detection accuracy of more than 94% in various failure scenarios, proving its robustness and ability to detect a variety of failure scenarios in a distributed microservices environment.

TABLE III. FAULT CATEGORY-WISE DETECTION ACCURACY OF THE PROPOSED FRAMEWORK

Fault Category	Detection Accuracy (%)
Service Crash	98.2
Resource Exhaustion	97.6
Network Failure	96.8
Configuration Anomaly	95.9
Dependency Failure	96.4
Security-Related Fault	94.7

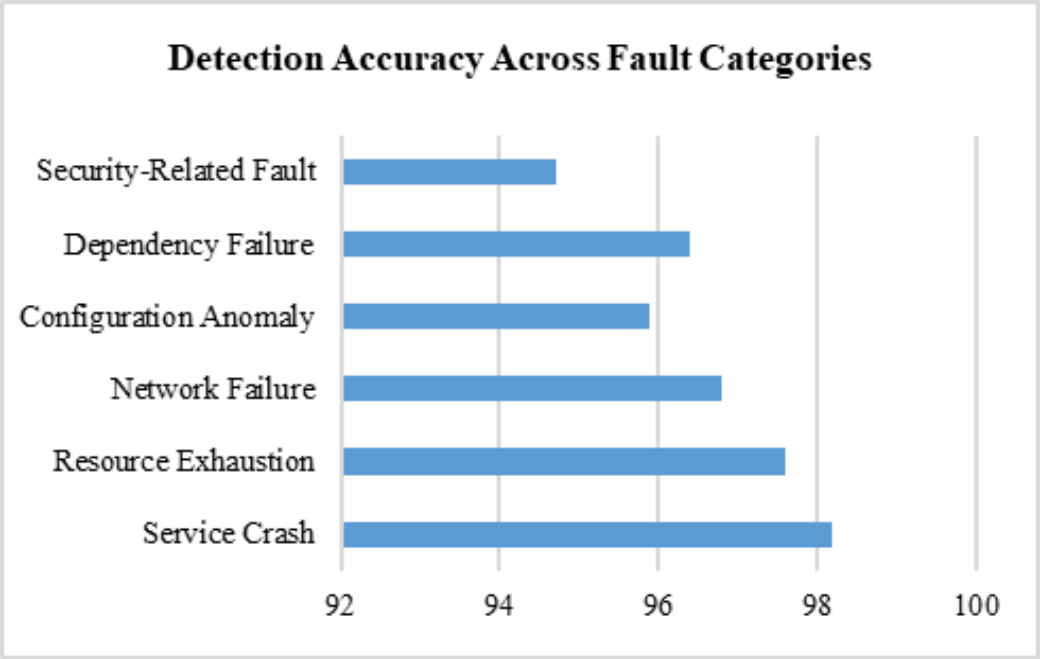

Fig. 4.Detection Accuracy Across Fault Categories

Figure 4 shows the accuracy of fault detection for the proposed Llama 3.1 Framework in detecting faults in different categories. The framework can attain very high accuracy on all the fault types with a Service Crash fault accuracy of 98.2%, Resource Exhaustion, and Network Failure coming in at a respectable number. The accuracy of Security-Related Faults is slightly lower, but still remains above 94%, highlighting the framework's strength and ability to detect a wide range of cloud infrastructure failures.

A. Comparative Analysis & Discussion

Table IV presents the fault detection accuracy of the proposed framework by comparing the results with the ones reported in previous studies. The proposed framework with Llama 3.1 generated the highest accuracy of fault detection at 97.5%, which is higher than the methods presented in [24] and [25], with 94.2% and 89.3%, respectively. The outcomes show that the proposed method is effective in detecting cloud infrastructure faults with high accuracy and increasing system reliability with AI-powered fault management.

TABLE IV. COMPARISON OF FAULT DETECTION ACCURACY WITH EXISTING STUDIES

Framework	Fault detection accuracy
Proposed	97.5
[24]	94.2
[25]	89.3

The results from the experiment demonstrate its ability to address significant cloud failure management challenges with the proposed framework, which is built on top of Llama-3.1. The framework outperformed rule-based and semi-automated in terms of fault detection, root cause identification and self-healing success rates. The marked decrease in Mean Time to Recovery demonstrates the framework's ability to quickly identify and correct faults with minimal human interaction. The high detection accuracy for different categories of faults further

demonstrates the robustness of the proposed approach for dealing with different types of cloud failures. Furthermore, lower false positive rates and higher service availability translate to better service reliability. The findings show the potential of LLMs in cloud management to improve system resilience, automate tasks, and ultimately increase the quality of cloud services in modern cloud environments.

V. Conclusion and Future Scope

The increasing complexity of reliability and availability in microservice-based cloud environments was addressed with a clever fault handling system that was designed with Llama 3.1. The framework includes real-time monitoring, fault detection, root cause analysis, and automated self-healing features, allowing for quick identification and fixing the failure of cloud infrastructure. Experimental results showed that the accuracy of fault detection, root cause identification, service availability and fault recovery performance were significantly improved compared to the conventional rule based and semi-automated approaches. These significant savings in MTTR and the lower need for manual intervention confirm the impact of using Large Language Models for autonomous cloud operations. The results show that with LLM-based fault management, the operational resilience in complex distributed systems can be improved, and the administrative effort can be reduced. The framework can be expanded to address large-scale multi-cloud and hybrid-cloud deployments, which are increasingly being attempted, thus making fault diagnosis and remediation more difficult, because of the heterogeneity of infrastructures. The incorporation of reinforcement learning methods may help to develop adaptive self-healing approaches that continually learn from past recovery decisions to optimize the strategy. Other investigations are also possible, such as predictive fault prevention, remediation mechanisms that take cybersecurity into account, and optimization of scalability in real-time. Simulations of the framework in production environments in the cloud will further validate its applicability and effectiveness for next-generation autonomous cloud management systems.

References

1. N. D. Bhandarwar, "A Mathematical Framework for Explainable and Adversarially Robust IDS Using ML for Large-Scale Enterprise and Cloud Systems," *Int. J. Appl. Math.*, vol. 39, no. 1, 2025.

2. V. Sharma, "Cloud-Native 5G Deployments: Kubernetes and Microservices in Telco Networks," *Int. J. Innov. Res. Eng. Multidiscip. Phys. Sci.*, vol. 10, no. 3, pp. 1–8, May 2022, doi: 10.37082/IJIRMPS.v10.i3.232706.

3. S. Jain and D. Jain, "Artifact Comparison Analyzer: Evaluating Microservice Build Metrics for Performance and Efficiency Improvements," in *2026 IEEE International Conference on AI Engineering and Innovations (AIEI)*, 2026, pp. 1–6. doi: 10.1109/AIEI69164.2026.11497468.

4. S. Gupta, G. Asirvatharaj, and R. T. Talluri, "AI-Powered Intelligent System through context aware log scrutiny for Anomaly detection," in *2026 18th International Conference on Communication Systems and Networks (COMSNETS)*, 2026, pp. 1303–1307.

5. M. Parikh, A. A. Soni, S. M. Shah, and A. R. Jha, "Big Data Workload Profiling for Energy-Aware Cloud Resource Management," Jan. 2026, doi: 10.48550/arXiv.2601.11935.

6. V. K. Bollu, "Threat Landscape in Artificial Intelligence Systems: Taxonomy, Attack Vectors and Security Implications," *World J. Adv. Res. Rev.*, vol. 29, no. 1, pp. 285–294, 2026, doi: 10.30574/wjarr.2026.29.1.0007.

7. R. K. Gadiraju, "A Novel Machine Learning Method for Fault Prediction and Reliability in Software Systems," *Int. J. Sci. Res. Sci. Eng. Technol.*, vol. 12, no. 3, pp. 1226–1238, Jun. 2025, doi: 10.32628/IJSRSET2512163.

8. V. Methuku, S. Kamatala, P. Naayini, and P. R. Vontela, "From Ethical Principles to Technical Safeguards: A Unified Framework for Safe and Human-Centered Artificial Intelligence," *Am. Int. J. Comput. Sci. Technol.*, vol. 4, no. 5, pp. 26–34, Sep. 2022, doi: 10.63282/3117-5481/AIJCST-V4I5P103.

9. B. P. Singh, "Securing the Boundary: Trust Context Separation in Privileged AI Agent Systems," *Comput. Fraud Secur.*, vol. 2026, no. 1, pp. 998–1009, 2026, doi: 10.5281/zenodo.19487302.

10. J. B. Mehta, "Designing Self-Healing Automation Frameworks for Flaky CI Environments," in *2025 International Conference on Computer and Applications (ICCA)*, IEEE, Dec. 2025, pp. 1–7. doi: 10.1109/ICCA66035.2025.11430985.

11. N. Kolli, J. W. Sajja, and A. Nerella, "Building Secure AI Agents for Autonomous Data Access in Compliance/Regulatory-Critical Environments," *Comput. Fraud Secur.*, vol. 2024, no. 9, pp. 363–

373, Sep. 2024, doi: 10.52710/cfs.746.

12. T. P. Patel, A. K. Elengovan, V. Ranganathan, M. Parikh, and D. Kole, "Self-Healing AI Systems Using Multi-Agent Learning," in *2026 International Seminar on Intelligent Business and Edge-Computing Research (ISIBER)*, 2026, pp. 7–12. doi: 10.1109/ISIBER68248.2026.11470173.

13. Y. Jin, Z. Yang, J. Liu, and X. Xu, "Anomaly detection and early warning mechanism for intelligent monitoring systems in multi-cloud environments based on LLM," in *2025 5th International Symposium on Computer Technology and Information Science (ISCTIS)*, 2025, pp. 167–170.

14. K. Gandhi, P. Verma, V. Govindarajan, and R. Sonani, "Advancing Software Maintenance with LLMs and Cloud-Based Deep Learning," in *International Conference on AI and Robotics*, 2025, pp. 388–406. doi: 10.1007/978-3-032-05548-4_31.

15. B. Krishnan, A. Thaneeru, R. Lingam, and S. K. Kaata, "The Future of Cloud Data Engineering: Multi-Tenant, Multi-Region Pipelines Leveraging LLM-Powered Data Governance," in *2025 1st International Conference on Advancement in Futuristic Technologies (ICAFT)*, IEEE, Dec. 2025, pp. 1–8. doi: 10.1109/ICAFT66710.2025.11453308.

16. M. R. C. Mukkolakkal, "InfraLLM: A Generic Large Language Model Framework for Production-Grade Microservice Auto-Scaling in Cloud Infrastructure," *Int. J. Sci. Res. Mod. Technol.*, vol. 4, no. 11, pp. 113–123, 2025, doi: 10.38124/ijsrmt.v4i11.1023.

17. R. Chen *et al.*, "GRACE: A Strategic LLM-Enhanced Graph Reinforcement Learning Framework for Adaptive Fault Recovery in Microservice Systems," in *Service-Oriented Computing*, Springer Nature Singapore, 2026, pp. 155–170. doi: 10.1007/978-981-95-5012-8_12.

18. C. Wang, T. Yuan, C. Hua, L. Chang, X. Yang, and Z. Qiu, "Integrating Large Language Models with Cloud-Native Observability for Automated Root Cause Analysis and Remediation," in *Proceedings of the 2025 3rd International Conference on Artificial Intelligence, Systems and Network Security*, Nov. 2025, pp. 327–334. doi: 10.1145/3797161.3797213.

19. V. Sawalkar, N. More, S. Jagadale, B. Shendkar, P. Chandre, and C. Mhaske, "Self-Healing Cloud Infrastructure: Leveraging AI for Fault Detection and Recovery," in *Data Science and Big Data Analytics*, Springer Nature Switzerland, 2025, pp. 22–33.

20. V. Avgerinos, K. Ramantas, L. Alonso, and C. Verikoukis, "ARM: Autonomous Remediation and Management With LLM Agents for Intent-Driven Control," *IEEE Internet Things J.*, vol. 13, no. 9, pp. 18305–18315, 2025, doi: 10.1109/JIOT.2025.3648858.

21. A. Arulappan, A. Mahanti, K. Passi, T. Srinivasan, R. Naha, and G. Raja, "DQN Approach for Adaptive Self-Healing of VNFs in Cloud-Native Network," *IEEE Access*, vol. 12, pp. 34489–34504, 2024, doi: 10.1109/ACCESS.2024.3365635.

22. A. J. Diego, "AI-Powered Autonomous Microservices : A Self-Healing Approach using Machine Learning Techniques," *Int. J. Artif. Intell. Appl.*, vol. 2, no. 1, pp. 96–101, 2023.

23. K. M. Alsaif, A. A. Albeshri, M. A. Khemakhem, and F. E. Eassa, "Multimodal Large Language Model-Based Fault Detection and Diagnosis in Context of Industry 4.0," *Electronics*, vol. 13, no. 24, p. 4912, Dec. 2024, doi: 10.3390/electronics13244912.

24. T. Song, W. Zhang, S.-N. Lang, and H. Yan, "LLM-Enhanced Intelligent Fault Diagnosis and Self-Healing Framework for Cloud Computing Systems," Jan. 09, 2026. doi: 10.20944/preprints202601.0630.v2.

25. T. P. Patel, S. R. K. V. Bayyavarapu, V. Soni, R. Purushothaman, G. B. Thokala, and V. Ranganathan, "LLMDebug: Prompt-Engineered Large Language Models for Automated Root Cause Analysis in Microservices Architectures," in *2026 International Conference on Advances in Artificial Intelligence and Machine Learning (AAIML)*, IEEE, Mar. 2026, pp. 373–380. doi: 10.1109/AAIML67890.2026.11498111.