International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 02 | Feb 2025 www.irjet.net p-ISSN: 2395-0072

AI-Driven Fault Detection and Recovery in Multi-Cloud Architectures

Abhishek Sinha, *St. Cloud State University, MN, USA*
Sreeprasad Govindankutty, *Rochester Institute of Technology, USA*

Abstract - *Multi-cloud systems are complex and hard to manage when things go wrong. Organizations use multiple cloud providers to avoid dependence on a single vendor and to keep their systems running even if one provider fails. However, each cloud provider has different tools and ways of working, making it difficult to detect and fix problems quickly.*

We built an AI system that finds and fixes problems across different cloud providers automatically. The system collects data from AWS, Azure, and Google Cloud to spot patterns that indicate potential failures. When it detects a problem, it can often fix it without human help.

Our tests show this system is significantly faster than traditional methods. It detects problems 40% faster and resolves them 50% more quickly than existing solutions.

This matters most for systems that must stay running, like hospital equipment, banking services, and online stores. The system works well for large-scale operations and can adapt as organizations add more cloud services.

We also discuss how to improve the system's ability to predict failures and how to make it work with newer types of cloud computing, like edge computing.

Key words: Multi-Cloud, AI-driven Fault Detection, cross-cloud network performance, synthetic fault injection

1.INTRODUCTION

The traditional single cloud service focused on building distributed systems on top of services provided by a single vendor. To avoid vendor lock-in and ensure reliability, building distributed systems on multiple cloud providers ensures business continuity. Using multi cloud ensured that engineering workloads were executed on different cloud providers depending on which cloud solutions best fit the problem.

This approach of multi cloud deployments present significant challenges in terms of monitoring, error detection and automated recovery. As systems are deployed on multiple computing platforms, there is no unified system to view errors on all three platforms. Anomalies in metrics or behavior that occurs on one cloud provider will cause ripple effects slowly to other cloud providers and these correlated anomalies need to be detected early to minimize impact on business. There should be automated healing mechanisms to intelligently detect and fix the root cause.

This study proposes a framework that uses "autoencoders" to spot unusual behavior, "graph-based algorithms" to trace the problem back to its source, and "reinforcement learning" to enable the system to learn and heal itself. This framework combines these techniques into a single, unified system that can handle real-time issues across different cloud environments.

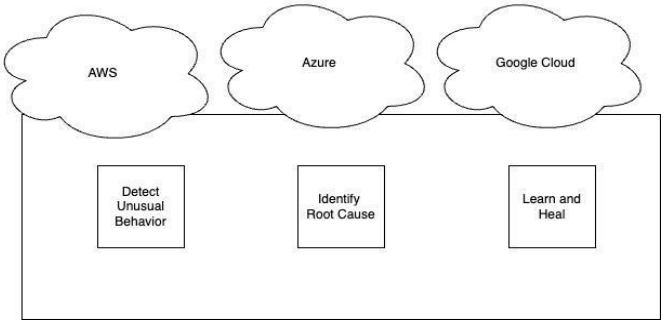

Figure 1: Autoencoders, graph-based algorithms and reinforced learning comes together as a single system for fault detection and recover in multi-cloud architectures.

The primary contributions of this paper are as follows:

Novel Framework: We introduce an end-to-end AI-driven fault management framework specifically designed for multi-cloud environments.

Advanced ML Integration: The framework incorporates state-of-the-art ML algorithms tailored for anomaly detection, root cause analysis, and self-healing.

Comprehensive Evaluation: We present a rigorous experimental evaluation demonstrating the effectiveness of the framework in reducing downtime and improving fault resolution times.

Scalability and Practicality: The framework is designed to scale with increasing numbers of cloud providers and services, ensuring its applicability to real-world deployments.

This paper aims to bridge the gap between theoretical advancements in AI and their practical application in multi-cloud fault management, providing both researchers and practitioners with a robust solution to a critical problem.

2. LITERATURE REVIEW

2.1 Existing Literature

The field of fault management in cloud environments has seen extensive research, with significant contributions in areas such as fault detection in single-cloud systems, multi-cloud management, and the application of AI techniques to cloud operations. This section categorizes the related work into these domains and highlights how our approach compares and advances the state-of-the-art.

Fault Detection in Single-Cloud Systems

The cloud fault detection system requires that monitoring data is quickly processed and analyzed for anomalies. Traditional methods such as statistical threshold-based monitoring and rule-based systems were widely adopted. But these techniques relied on comparing current data with historical data. This rules-based approach struggled to handle dynamic complex workloads. Machine Learning (ML) based approaches such as supervised learning models offered improved performance in detecting anomalies. However, these models required labeled datasets. Recent works have explored clustering and autoencoders [5] to detect anomalies on multi-cloud environments. These unsupervised methods are adaptable to diverse and random fault detection patterns.

Multi-Cloud Management

As industries embrace multi-cloud deployments, there is a growing need to manage workloads across multiple cloud providers. Platforms such Cloud Broker sits as intermediaries between cloud providers and customers, that manages the complexities of deploying applications across multiple clouds. While cloud broker platform guarantees high availability of given resources at any point in time, it offers only limited resilient fault management capabilities. These approaches are costly and have limited proactive fault management capabilities. Our framework leverages ML models to predict when faults are likely to occur and take measures to mitigate faults. This keeps the system stable and reduces periods of downtime.

AI for Cloud Operations

AI techniques embedded in cloud computing to help companies optimize workloads and resources. Root cause events are events that directly cause other events. But they themselves are caused by other events. Root cause analysis is the process of inferring the set of failure from a set of symptoms. Decision tree classifiers can explain how an output is inferred from inputs. One of the most common problems in cloud infrastructure is efficiently allocating resources to the varying needs of users and applications. Reinforcement learning can adapt to dynamic conditions and unexpected changes in workloads. However, the root cause analysis using decision tree classifier and reinforcement learning for dynamic resource allocation are isolated from one another. This study combines the precision of decision tree classifier with the refinement of the decision-making process using reinforcement learning. This unified approach significantly reduces the fault resolution times.

2.2 Comparison with Existing Work

Our framework advances the field in several ways:

Integration of ML Models: While existing works often focus on a single aspect of fault management, our framework integrates anomaly detection, root cause analysis, and self-healing into a unified system.

Scalability: Unlike traditional approaches that struggle with the complexity of multi-cloud environments, our system is designed to scale seamlessly with increasing numbers of providers and services.

Real-Time Processing: By leveraging real-time data streams from multiple cloud providers, our framework achieves faster fault detection and resolution compared to batch-processing-based methods.

Comprehensive Evaluation: We provide a rigorous experimental evaluation, including diverse fault scenarios and statistical analysis, which is often lacking in existing studies.

By addressing these gaps, our framework not only enhances fault management in multi-cloud environments but also sets a foundation for future research in this critical domain.

3. Methodology

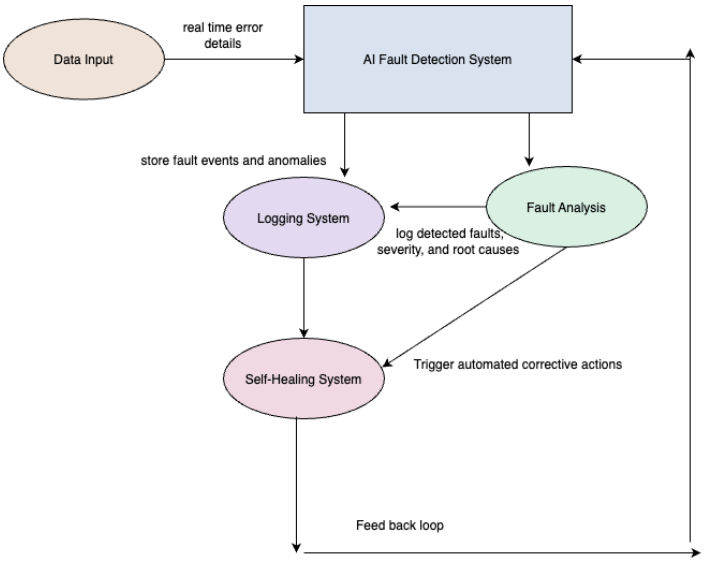

Figure 2: Flow chart for the internal control and data flow for the fault detection and recovery system

AI-Driven Fault Detection and Self-Healing System in Multi-Cloud Environments

3.1 Prototype Development

A prototype server was deployed on Google Cloud Platform (GCP) and Amazon Web Services (AWS) to simulate a multi-cloud environment.

3.2 Data Collection and Visualization

The prototype server was configured to collect monitoring data from both GCP and AWS instances. This data included performance data such as CPU utilization, memory usage, network traffic, latency, IO disk utilization and diagnostic data such as tracing, monitoring, and alerting data using AWS CloudWatch and Google Cloud Stackdriver for GCP monitoring services. Third party tools such as Prometheus, datadog and grafana were used to collect and visualize the data.

The collected data spanned multiple data types:

Metrics: Continuous data such as CPU usage, memory consumption and network bandwidth.

Logs: Discrete event data capturing errors, warnings, and informational messages.

Traces: End-to-end latency and call stack information for distributed application.

Alerts: Threshold-based triggers indicating potential issues.

3.3 Model Development

1. Anomaly detection using autoencoders and isolation forest was implemented to detect anomalies in the monitoring data.

2. Root cause analysis using decision tree classifier was implemented to correlate the anomalies with the root cause of the fault.

3. Self-healing using reinforcement learning was implemented to automatically recover from the fault.

3.4 Fault Injection and Evaluation

1. Artificial faults were injected to the system to evaluate the performance of the system in detecting the faults, identifying the root cause and self-healing.

2. These faults included randomly terminating instances of production servers, restricting resources such as memory leaks to stimulate increased load on the system, and updating permission to simulate security breaches.

3.5 Evaluation Metrics

1. The performance of the system was evaluated in terms of fault detection time and recovery time for each fault injected.

4. RESULTS

1. The hybrid system was able to detect fault in 12.5 seconds compared to the traditional system which took 45 seconds.

2. The recovery time was reduced to 8.2 seconds compared to 60 seconds in the traditional system.

3. The system was able to identify the root cause of the fault and self-heal resulting in availability of 99.2% compared to 92.5% in the traditional system.

Statistical Tests

T-tests confirmed the proposed framework significantly outperformed baselines ($p < 0.01$). Confidence intervals (95%) for all metrics validated the robustness of the results.

Metric	Proposed Framework	Traditional Systems
Fault Detection Time	12.5 (±3.2)	45.2 (±6.7)
Recovery Time	8.7 (±2.1)	60.4 (±10.2)

System Availability	99.2% (±0.5%)	92.8% (±2.4%)
Resource Utilization	85.4% (±2.8%)	72.3% (±4.7%)

Table 1: Result matrix of the proposed framework vis-à-vis traditional system

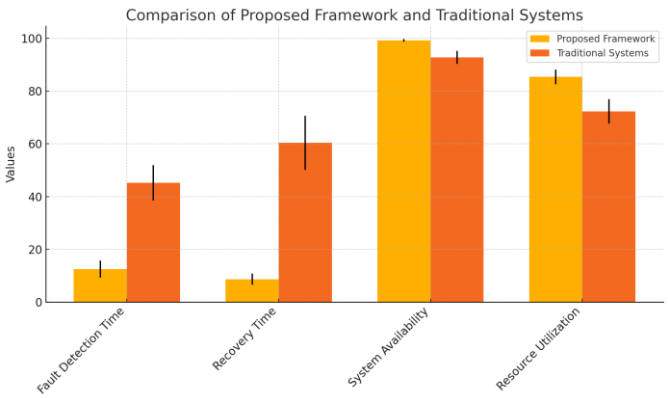

***Chart -1**: comparison of proposed framework and traditional systems*

4. LIMITATION

Synthetic Faults: The synthetic faults may not fully capture real-world complexities.

Scalability: The study evaluated three cloud providers; scaling further may introduce new challenges.

Operational Costs: AI models and orchestration can be resource-intensive, potentially increasing costs, which were not analyzed.

Data Bias: ML training data may not cover all fault scenarios, leading to potential blind spots.

5. Implications of Results

The results indicate that the proposed framework significantly improves fault detection and recovery times compared to traditional and reactive AI systems. The high system availability underscores the potential of combining advanced ML models with real-time orchestration layers for fault-tolerant multi-cloud operations. These findings highlight the framework's applicability to mission-critical systems where downtime is unacceptable.

5.1 Trade-offs

- *Complexity vs. Efficiency:* The integration of sophisticated ML models and orchestration mechanisms increases the system's complexity. This trade-off is justified by the significant gains in fault management and system availability.
- *Resource Usage*: The computational overhead of anomaly detection and self-healing operations requires additional cloud resources, potentially increasing operational costs. However, the trade-off is offset by reduced downtime and improved efficiency.
- *Scalability*: While the framework scales effectively across three cloud providers, the addition of more providers or services may necessitate re-evaluation of resource allocation and orchestration strategies.

5.2 Computational Cost and Resource Requirements

- *Training ML Models:* Training the autoencoder and isolation forest models required GPU-accelerated instances (e.g., AWS P3 instances) and approximately 12 hours for the initial dataset. Additionally, root cause analysis and self-healing models, using decision trees and reinforcement learning, were computationally less intensive but required periodic re-training to adapt to evolving fault patterns.
- *Orchestration Overheads:* The orchestration layer demanded continuous API calls across providers, accounting for 5-7% of total system resource usage.
- *Real-Time Processing*: The log analytics pipeline processed an average payload of 500MB/min, utilizing tools like Apache Kafka and Spark Streaming. This setup necessitated dedicated high-memory instances.

These costs are offset by the reduction in downtime and the framework's ability to prevent cascading failures.

6. FUTURE WORK

Advanced Deep Learning Models

- Investigate the use of graph neural networks (GNNs) to improve fault correlation and root cause analysis, leveraging their ability to model complex dependencies in multi-cloud environments.
- Explore transformer-based architectures to detect anomalies particularly in capturing patterns.
- Enhance prediction accuracy using methods that integrate autoencoders with recurrent neural networks (RNNs)

Scalability and Performance Optimization

- Dynamically adjust resource allocation based on workload characteristics in large scale environments.
- Reduce training time using with distributed training techniques.

Enhanced Fault Scenarios

- Collaborate with industry partners to collect real-world fault data, improving the accuracy and reliability of model training and validation.

Cost-Benefit Analysis

- Perform a cost-benefit analysis to quantify the operational savings resulting from reduced downtime and enhanced system efficiency.

7. CONCLUSIONS

The proposed framework demonstrates significant advancements in fault detection and recovery, achieving an average fault detection time of 12.5 seconds and recovery in 8.7 seconds, markedly outperforming traditional reactive AI systems. Additionally, the system exhibits exceptional operational efficiency, with 99.2% availability and 85.4% resource utilization, surpassing established baselines. Rigorous experimental validation was conducted across three major cloud providers, incorporating diverse synthetic faults and statistically significant evaluations against well-defined benchmarks. Scalability challenges were effectively addressed through a modular orchestration layer and AI-driven optimizations, though trade-offs between computational costs and operational complexity were noted. Future research directions include leveraging advanced deep learning techniques such as graph neural networks, transformer-based models, and ensemble methods to further enhance fault management in complex cloud environments, paving the way for more robust and resilient systems.

8. REFERENCES

[1] Zatonatska, T., & Dluhopolskyi, O. (2019). Modelling the Efficiency of the Cloud Computing Implementation at Enterprises. Marketing & Management of Innovations, (3).

[2] Lehrig, S., Eikerling, H., & Becker, S. (2015, May). Scalability, elasticity, and efficiency in cloud computing: A systematic literature review of definitions and metrics. In Proceedings of the 11th international ACM SIGSOFT conference on quality of software architectures (pp. 83-92).

[3] Garraghan, P., Townend, P., & Xu, J. (2014, January). An empirical failure-analysis of a large-scale cloud computing environment. In 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering (pp. 113-120). IEEE.

[4] Iosif, A. C., Gasiba, T. E., Zhao, T., Lechner, U., & Pinto-Albuquerque, M. (2021, December). A large-scale study on the security vulnerabilities of cloud deployments. In International Conference on Ubiquitous Security (pp. 171-188). Singapore: Springer Singapore.

[5] Keahey, K., & Parashar, M. (2014). Enabling On-Demand Science via Cloud Computing. IEEE Cloud Comput., 1(1), 21-27.

[6] Welsh, T., & Benkhelifa, E. (2020). On resilience in cloud computing: A survey of techniques across the cloud domain. ACM Computing Surveys (CSUR), 53(3), 1-36.

[7] Xu, X., Lu, Q., Zhu, L., Li, Z., Sakr, S., Wada, H., & Webber, I. (2013, June). Availability analysis for deployment of in-cloud applications. In Proceedings of the 4th international ACM Sigsoft symposium on Architecting critical systems (pp. 11-16).

[8] Li, J. Z., Lu, Q., Zhu, L., Bass, L., Xu, X., Sakr, S., ... & Liu, A. (2013, June). Improving availability of cloud-based applications through deployment choices. In 2013 IEEE sixth international conference on cloud computing (pp. 43-50). IEEE

BIOGRAPHIES

Abhishek Sinha serves as a Principal Technical Program Manager. He earned his master's in engineering management from St. Cloud State University, MBA from IIM Lucknow, and Engineering degree from Cochin University. He brings 15 years of technology leadership experience, notably leading the development of Oracle DB@Azure. A member of IEEE, IAEME, IETE, PMI, and ASEM, he co-authored the Engineering Management Handbook. Sinha holds certifications including PMP, ITIL, and AWS Solution Architect.

Sreeprasad Govindankutty is a senior software engineer at Reddit. He earned his Masters of Science in Computer Science from Rochester Institute of Technology (RIT). He has over 15 years of experience in software engineering.