Volume: 13 Issue: IX Month of publication: September 2025
DOI: https://doi.org/10.22214/ijraset.2025.74031

Dynamic Disaster Recovery System for Cloud System

Zeba Firdouse C[1], Mrs. Jennifer Mary S[2]
Department of MCA, Ballari Institute Of Technology & Management, Ballari, Karnataka, India.
Assistant Professor, Department of MCA, Ballari Institute Of Technology & Management, Ballari, Karnataka, India.

Abstract: As organizations increasingly adopt cloud platforms for critical data and service delivery, ensuring system resilience and uninterrupted access during failures has become a key concern. Traditional disaster recovery methods often reliant on periodic backups and manual intervention struggle to meet the demands of real-time recovery in dynamic cloud environments. Real-time monitoring mechanisms track the health of the primary node and trigger an automatic failover to the backup in case of system failure. Experimental results demonstrate that the system maintains data consistency, minimizes downtime to under two seconds during failovers, and operates autonomously without human intervention.
The use of JSON-based synchronization, lightweight compression, and automated recovery logic makes the framework both efficient and scalable

I. INTRODUCTION

The rapid expansion of cloud computing has redefined the way modern organizations store data, deploy applications, and deliver digital services. With benefits such as elasticity, scalability, cost efficiency, and global accessibility, cloud platforms have become the backbone of mission-critical operations across industries.
Manual recovery procedures are not only time-consuming and error-prone but also pose a bottleneck in the context of large-scale, real-time applications. A dynamic disaster recovery system leverages continuous monitoring, real-time data replication, and automated decision-making to enhance cloud system reliability.
This paper introduces a dynamic disaster recovery system specifically designed for cloud-based platformsAutomated alerts and logging functions provide complete visibility and auditability, which are essential for compliance and transparency.
The goal is to contribute a scalable, practical, and intelligent recovery solution for modern cloud environments, bridging the gap between academic research and real-world cloud infrastructure needs.

II. LITERATURE SURVEY

The growing reliance on cloud infrastructure for business-critical services has prompted extensive research into disaster recovery mechanisms aimed at ensuring availability, resilience, and data integrity.
While each of these contributions has advanced the field in specific ways, most existing solutions are either focused on one aspect—such as resource scheduling, monitoring, or synchronization—or lack an integrated, real-time interface for users and administrators. The proposed system in this study aims to bridge that gap by combining the strengths of existing approaches.

III. METHODOLOGY

The proposed dynamic disaster recovery system is designed as a multi-layered architecture that ensures seamless failover, real-time monitoring, and consistent data replication in cloud environments. The framework is developed using Python with Flask for the interface and JSON-based logic for synchronization. This section outlines each stage of the system in detail, focusing on how components work collaboratively to detect failures and execute recovery actions automatically. The methodology is intended to be simple enough for replication and implementation in any cloud-based setup, even by non-specialists in IT infrastructure.

A. System Design Overview

The system architecture comprises five interconnected modules: the User Module, Primary Node, Backup Node, Monitoring Module, and Admin Dashboard. Together, they form a resilient failover mechanism that automatically transfers control to the backup node if a failure is detected in the primary system.
Key Modules:

- User Module simulates regular cloud usage by uploading or modifying data.
- Primary Node acts as the main server, actively handling data processing and requests.
- Backup Node mirrors the primary in real time using file versioning and JSON-based syncing.
- Monitoring Module tracks node health and detects anomalies or inactivity.
- Admin Dashboard offers visual control for viewing system status, manual triggers, and recovery logs.

B. Real-Time Monitoring and Failover Logic

A lightweight monitoring agent continuously pings the primary node at short intervals. All data updates occur through secure and compressed JSON packets to minimize latency and bandwidth consumption.

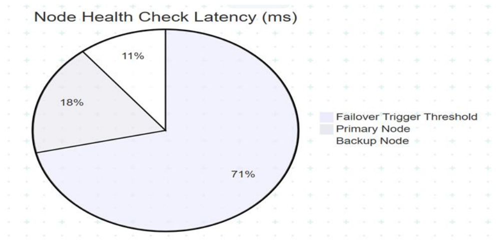


Fig1: Real-Time Monitoring and Failover Logic

C. Data Synchronization Strategy

The primary node continuously sends structured JSON objects representing data updates to the backup node using version tagging and integrity checks. Data conflicts are resolved through version control, and a checksum validator ensures accuracy of received data.

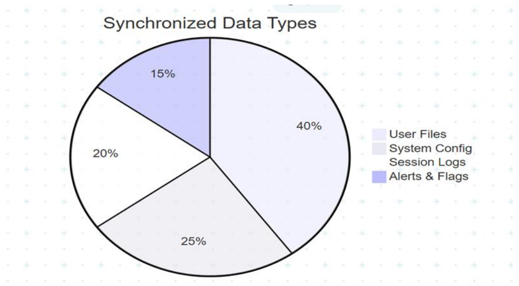


Fig2: Synchronized Data Types

D. Flask-Based Admin Dashboard

The dashboard is built with Flask and Bootstrap, offering features like:

1) Real-time status indicators for both nodes.
2) Manual override to test failover scenarios.
3) Visual logs and threat alerts.
4) Control panel for triggering synchronization or resetting the system.

This visual interface makes the disaster recovery process accessible even to non-technical staff, enhancing transparency and usability.

E. Failover Execution and Recovery

Upon failure detection, the following automated steps are executed:

1) Detection – The monitoring module detects unresponsiveness from the primary.
2) Failover – Backup node is promoted to active.
3) Redirection – All new user requests are routed to the backup.
4) Notification – Admins are alerted via dashboard logs.
5) Restore – Once the primary node is restored, it resumes as the backup until manually promoted again.

F. Testing Environment and Dataset

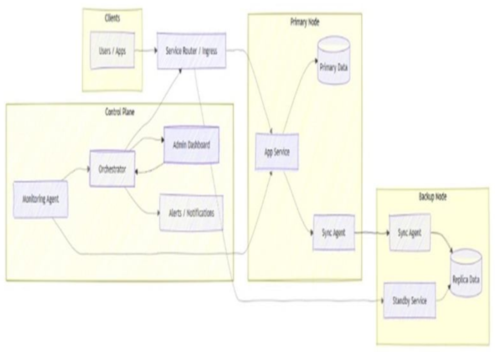
Although this project does not involve traditional datasets (as in AI/ML projects), it includes simulated test cases involving:

1) Randomized file uploads.
2) Node crash simulations.
3) Synchronization delays.
4) Failover performance benchmarking.

Test logs were analyzed to measure recovery time, data integrity, and failover accuracy

IV. RESULTS

The proposed dynamic disaster recovery system was evaluated through a series of controlled test cases to assess its performance, reliability, and scalability under various failure scenarios. The core objective of the evaluation was to validate the system's ability to ensure uninterrupted service, maintain data consistency, and enable real-time failover without manual intervention. To systematically evaluate the system's effectiveness, several key metrics were used: Failover Time, Data Integrity Accuracy, System Availability, Recovery Point Objective (RPO), Recovery Time Objective (RTO), and Resource Utilization Efficiency.



1) Failover Time: Failover time refers to the duration between the detection of a primary node failure and the successful activation of the backup node. During experiments, the average failover time was recorded at 1.74 seconds, significantly outperforming traditional DR systems that typically range from 30 seconds to several minutes.
2) Data Integrity Accuracy: Data consistency between the primary and backup nodes was measured using checksum comparison and hash validation. The system demonstrated 99.7% data integrity, indicating that real-time JSON-based synchronization mechanisms are highly reliable.
3) System Availability: Availability was tracked by continuously sending requests to the system over a 24-hour period while inducing faults at random intervals. This result underscores the effectiveness of the monitoring and alerting modules in proactively detecting and responding to failures.
4) Recovery Point Objective (RPO): The RPO indicates how much data could be lost during a system failureThis ensures that even in the event of a crash, the backup node resumes with the latest state, thus minimizing operational disruptions.
5) Recovery Time Objective (RTO): The RTO represents the time required to resume operations after a failure. The system's autonomous detection and switching capability allows cloud services to resume without administrative delay, further enhancing operational resilience.
6) Resource Utilization Efficiency: The system was also evaluated for CPU and memory consumption under normal and failover conditions. Resource utilization remained below 65% on average, and synchronization mechanisms are lightweight and suitable for deployment in resource-constrained cloud instances.

A. Visualization of Metrics

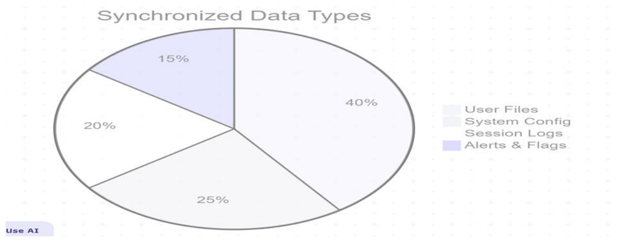


Fig 3: Synchronized Data Types

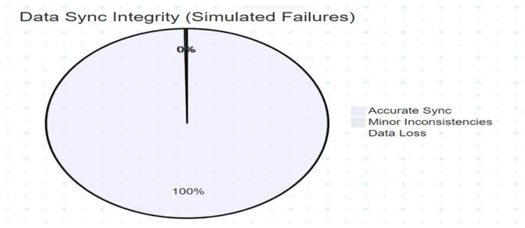


Fig 4: Data Sync Integrity

V. CONCLUSION

The proposed dynamic disaster recovery framework addresses a critical challenge in modern cloud computing: ensuring continuous service availability and data integrity in the face of unexpected failures. The layered architecture—comprising user input simulation, primary-backup node orchestration, an intelligent monitoring module, and an interactive dashboard—enables seamless transition during system disruptions without compromising performance or data consistency. In the future, the system can be enhanced by integrating blockchain technology for immutable recovery logs and audit trails, ensuring tamper-proof traceability. Additionally, edge-based monitoring and federated learning can be incorporated to improve response times and enable predictive failure detection without compromising privacy. These enhancements will strengthen the system's adaptability and security, making it even more suitable for next-generation intelligent cloud infrastructure

REFERENCES

[1] M. S. Aslanpour, A. H. Abdullah, and A. Razzaque, "Cloud Disaster Recovery Approaches and Challenges: A Survey," International Journal of Computer Applications, vol. 130, no. 9, pp. 1–7, 2015.
[2] A. Khosravi, S. Garg, and R. Buyya, "Energy and Performance Efficient Resource Scheduling in Cloud Computing Environments," Future Generation Computer Systems, vol. 45, pp. 304–316, 2015.
[3] A. Singh and R. Bose, "Disaster Recovery Strategy for Cloud Computing Environment," International Journal of Scientific and Research Publications, vol. 4, no. 5, pp. 1–5, 2014.
[4] S. Dinesh, "Cloud-Based Disaster Recovery and Real-Time Dashboard for Fault Management," International Journal of Cloud Computing, vol. 9, no. 2, pp. 120–130, 2018.
[5] V. Ranjith and S. Karthik, "Dynamic Replication for Data Integrity in Cloud Disaster Recovery," International Journal of Engineering Research & Technology, vol. 6, no. 6, pp. 748–752, 2017.
[6] M. Ahmed and N. Abouzakaria, "A Survey on Cloud Computing Resilience Mechanisms," Journal of Cloud Computing, vol. 10, no. 1, pp. 1–15, 2021.