© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718118

ResiliLens: An AI-Driven Framework for Simulation and Failure Analysis in Distributed Systems

VEDANT MESHRAM[1], ASHWINI GARKHEDKAR[2]
[1,2]*MCA Department, P E S Modern College of Engineering Pune, India*

Abstract- ***Distributed systems are increasingly complex due to the integration of heterogeneous components such as cloud services, IoT devices, and network infrastructures. Ensuring reliability and performance under varying conditions remains a significant challenge. This paper proposes ResiliLens, a conceptual framework that leverages simulation techniques and artificial intelligence to analyze distributed system behavior under failure scenarios. The framework enables virtual system modeling, failure injection, performance evaluation, and AI-driven recommendations for system improvement. By providing an intuitive and unified approach, ResiliLens aims to simplify distributed system validation and enhance reliability through intelligent insights. Modern distributed systems form the backbone of applications such as smart cities, cloud computing, and real-time analytics. These systems involve multiple interconnected components operating across diverse environments. However, their complexity introduces challenges in ensuring performance, reliability, and fault tolerance.***

I. INTRODUCTION

Modern distributed systems form the backbone of applications such as smart cities, cloud computing, and real-time analytics. These systems involve multiple interconnected components operating across diverse environments. However, their complexity introduces challenges in ensuring performance, reliability, and fault tolerance.

Failures such as server crashes, network delays, and traffic spikes can significantly impact system behavior. Traditional testing approaches rely on fragmented tools and complex configurations, making it difficult to evaluate system performance comprehensively.

To address these challenges, this paper introduces ResiliLens, a theoretical framework designed to simulate distributed systems, inject failures, and provide intelligent recommendations using AI techniques.

II. LITERATURE SURVEY:

Das, et.al. (2025). The authors proposed a model-driven middleware framework supported by Agentic AI for distributed system design and validation. Their work focuses on federating multiple testbeds and automating experiment orchestration to evaluate system performance and reliability under varying conditions. However, the approach is complex and requires advanced infrastructure for practical implementation.

Gokhale, et.al. (2008). The authors introduced the concept of model-driven middleware for developing distributed real-time and embedded systems. The study emphasizes the use of abstract models to simplify system design and ensure correctness. While effective, it requires deep technical expertise and lacks support for interactive simulation.

Barve, et.al. (2017). The authors developed a cloud-based environment for learning distributed system algorithms. The platform enables users to experiment with system behavior, but it is primarily focused on educational purposes and does not support real-time failure analysis or intelligent recommendations.

Lan, et.al. (2025). The authors presented a federated experimentation framework that automates resource configuration across multiple testbeds. Their approach simplifies deployment but does not fully address dynamic failure handling or provide AI-driven insights for system optimization.

Hu, et.al. (2025). The authors explored automated environment setup using intelligent agents. Their work highlights the potential of Agentic AI in

automating complex system configurations, but it does not focus on failure simulation or system behavior analysis.

Kovrigin, et.al. (2025). The study proposed reinforcement learning-based techniques for automated system environment setup. It demonstrates how intelligent agents can optimize system configuration, but lacks integration with distributed system failure analysis.

Yu (2025). The author conducted a systematic review on fault injection testing in distributed systems. The study highlights the importance of controlled failure simulation for evaluating system resilience. However, it does not provide a unified framework for integrating simulation with intelligent recommendations.

Miranda (2024). The author proposed a catalog of fault-tolerant design patterns for microservices systems. The study organizes resilience techniques such as replication, retry mechanisms, and circuit breakers, but does not address simulation-based validation.

Sabuhi, et.al. (2024). The authors presented a microservice-based architecture for distributed systems with fault tolerance and scalability. Their work focuses on system reliability but lacks mechanisms for interactive simulation and failure analysis.

Enjam and Tekale (2023). The authors proposed a self-healing architecture for distributed applications using cloud-based infrastructure. Their approach emphasizes automated recovery and monitoring, but does not include failure simulation or AI-based recommendation systems.

Ramsingh, et.al. (2022). The authors analyzed reliability patterns in distributed systems and studied failure propagation across interconnected services. Their work provides insights into system dependencies but does not offer simulation tools for practical evaluation.

Hannousse and Yahiouche (2020). The authors conducted a systematic mapping study on distributed systems, focusing on security and reliability challenges. The study identifies architectural vulnerabilities but lacks a unified simulation framework.

Addeen (2019). The author proposed a dynamic fault tolerance model using state-based system evaluation. The approach focuses on predicting system behavior under failures but does not integrate AI-based analysis or interactive system modeling.

III. RESEARCH METHODOLOGY:

This study adopts a conceptual and design-oriented research approach to develop a framework for analyzing distributed system behavior under failure conditions. The methodology focuses on combining system modeling, simulation techniques, and AI-driven analysis to evaluate system performance and reliability.

The research is primarily qualitative and analytical, supported by conceptual experimentation through simulated scenarios rather than real-world deployment.

The proposed framework, ResiliLens, is designed as a multi-layered system consisting of the following components:

- System Modeling Layer: Defines distributed system architecture including nodes, services, and communication links
- Simulation Engine: Executes system behavior under defined workloads
- Failure Injection Module: Introduces controlled faults into the system
- Monitoring and Metrics Module: Tracks performance indicators such as latency and error rate
- AI-Based Analysis Module: Interprets system behavior and provides recommendations

This layered design ensures modularity, scalability, and ease of analysis.

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718118

IV. ARCHITECTURE:

The architecture consists of multiple interconnected layers that work together to simulate distributed system behavior and provide intelligent insights. These layers include:

- System Modeling Layer
- Simulation Layer
- Failure Injection Layer
- Monitoring and Metrics Layer
- AI Recommendation Layer
- Output Layer

Each layer communicates with adjacent layers to ensure seamless data flow and accurate analysis of system behavior.

2. System Modeling Layer
The System Modeling Layer is responsible for defining the structure of the distributed system. It allows users to represent system components such as clients, servers, load balancers, databases, and network links in an abstract form.

Each component is characterized by parameters such as processing capacity, load, response time, and operational status. This layer acts as the foundation of the architecture by providing a blueprint of the system to be simulated.

3. Simulation Layer
The Simulation Layer executes the behavior of the modeled system under predefined conditions. It simulates request flow, load distribution, and interactions between system components.

This layer applies logical rules to mimic real-world system operations, such as distributing incoming requests across servers and processing data through multiple nodes. It enables the evaluation of system performance under different workloads.

4. Failure Injection Layer
The Failure Injection Layer introduces controlled faults into the system to replicate real-world failure conditions. Common failure scenarios include server crashes, network delays, database overload, and sudden traffic spikes.

This layer is critical for analyzing system resilience, as it allows observation of how failures propagate through the system and affect overall performance.

5. Monitoring and Metrics Layer
The Monitoring and Metrics Layer collects and analyzes performance data generated during simulation. Key performance indicators include latency, throughput, error rate, and system health score.

This layer provides quantitative insights into system behavior and helps in identifying bottlenecks and performance degradation caused by failures.

6. AI Recommendation Layer
The AI Recommendation Layer enhances the analytical capabilities of the architecture by interpreting system logs and performance metrics. Using intelligent analysis, it identifies root causes of failures and suggests improvements.

These recommendations may include strategies such as load balancing, caching, replication, and auto-scaling, enabling better system design and optimization.

7. Output Layer
The Output Layer presents the final results of the simulation and analysis process. It includes performance reports, system logs, visualizations, and AI-generated recommendations.

This layer ensures that the results are presented in an understandable format, supporting effective decision-making.

8. Architectural Characteristics
The proposed architecture exhibits the following characteristics:

- Modularity: Each layer operates independently, allowing easy modification and extension
- Scalability: The framework can be extended to simulate large-scale distributed systems
- Flexibility: Supports multiple system configurations and failure scenarios
- Intelligence: Integrates AI for automated analysis and recommendations
- Simplicity: Provides an intuitive structure for understanding complex systems

The Role and Impact of the Proposed AI-Driven Simulation Based Architecture

1.1 Simplified System Modeling
The architecture enables users to represent complex distributed systems using abstract models. By defining components such as clients, servers, load balancers, and databases, the framework simplifies system design and allows users to visualize interactions between different components without requiring deep infrastructure-level knowledge.

1.2 Failure Simulation and Analysis
A key role of the architecture is the ability to simulate real-world failure scenarios. Through controlled failure injection, such as server crashes, traffic spikes, database overloads, and network delays, the system allows users to observe how distributed systems behave under stress. This helps in identifying vulnerabilities and understanding failure propagation across components.

1.3 Performance Monitoring
The architecture provides continuous monitoring of system performance using metrics such as latency, throughput, error rate, and system health score. These metrics offer quantitative insights into system behavior and help evaluate the impact of failures on overall system performance.

1.4 AI-Based Insight Generation
The integration of artificial intelligence enhances the analytical capabilities of the framework. The AI module processes system logs and performance data to identify root causes of failures and generate actionable recommendations. This reduces the need for manual analysis and supports informed decision-making.

1.5 Support for Learning and Design Optimization
The proposed architecture serves as an educational and analytical tool for understanding distributed system concepts. It enables users to experiment with different system configurations and observe the effects of design choices, thereby improving system architecture and resilience.

V. APPLICATIONS OF THE SYSTEM

1. Distributed System Design and Testing
One of the primary applications of the system is in the design and testing of distributed systems. Developers can model different system architectures and simulate real-world scenarios such as server failures, network delays, and traffic spikes. This helps in identifying potential bottlenecks and improving system reliability before deployment.

2. Cloud Computing and Web Applications
The framework can be applied in cloud-based systems and large-scale web applications, where maintaining high availability and performance is critical. By simulating workload variations and failure conditions, developers can evaluate how cloud architectures respond under stress and implement strategies such as load balancing, replication, and auto-scaling.

3. Microservices Architecture Analysis
Modern applications are increasingly built using microservices architecture, where services operate independently but are interconnected. The proposed system helps in analyzing service dependencies, failure propagation, and system resilience, making it easier to design fault-tolerant microservices systems.

4. DevOps and System Monitoring
The framework can assist DevOps teams in understanding system behavior through simulation-based monitoring and analysis. It allows teams to test deployment strategies, observe system responses, and

evaluate recovery mechanisms without affecting live systems.

5. Educational and Learning Tool

ResiliLens can be effectively used as an educational platform for students learning distributed systems. It provides a visual and interactive way to understand concepts such as load balancing, fault tolerance, scalability, and system performance, which are otherwise difficult to grasp through theory alone.

6. Fault Tolerance and Resilience Planning

The system supports fault tolerance analysis by simulating different failure scenarios and evaluating system response. This helps organizations design resilient systems capable of handling unexpected failures and maintaining service continuity.

7. Research and Experimental Analysis

Researchers can use the framework to conduct experimental studies on distributed systems. It allows for controlled testing of different architectures, failure conditions, and recovery strategies, making it suitable for academic research and performance evaluation.

8. Performance Optimization

The framework can be used to analyze system performance and identify inefficiencies. By observing metrics such as latency, throughput, and error rate, developers can optimize system design and improve overall performance.

VI. CONCLUSION

In this paper, an AI-driven simulation-based framework, ResiliLens, has been proposed to address the challenges associated with the design, analysis, and validation of distributed systems. As distributed systems continue to grow in complexity, ensuring reliability, scalability, and performance under varying conditions becomes increasingly difficult. Traditional testing approaches often involve complex configurations and lack unified platforms for comprehensive analysis.

The proposed framework overcomes these limitations by integrating system modeling, failure simulation, performance monitoring, and AI-based analysis into a single unified architecture. Through simulation of real-world failure scenarios such as server crashes, network delays, and traffic spikes, the system enables a deeper understanding of system behavior under stress conditions.

A key contribution of this work is the incorporation of an AI-based recommendation module, which analyzes system performance and provides actionable insights for improving system design. This reduces the dependency on manual analysis and supports intelligent decision-making in distributed system development.

REFERENCES:

[1] S. Das, et al., “Model and Agentic AI-driven Middleware for Distributed Systems Design and Validation,” ACM Middleware Conference, 2025. 🔗 https://doi.org/10.1145/3721464.3777430

[2] A. Gokhale, et al., “Model-Driven Middleware: A New Paradigm for Developing Distributed Real-Time and Embedded Systems,” Science of Computer Programming, 2008. 🔗 https://doi.org/10.1016/j.scico.2008.02.001

[3] Y. D. Barve, et al., “PADS: Design and Implementation of a Cloud-Based Immersive Learning Environment for Distributed Systems Algorithms,” IEEE Transactions on Emerging Topics in Computing, 2017. 🔗 https://doi.org/10.1109/TETC.2015.2511750

[4] R. Hu, et al., “Repo2Run: Automated Building Executable Environment for Code Repository at Scale,” arXiv, 2025. 🔗 https://arxiv.org/abs/2502.13681

[5] A. Kovrigin, et al., “PIPer: On-Device Environment Setup via Online Reinforcement Learning,” arXiv, 2025. 🔗 https://arxiv.org/abs/2509.25455

[6] N. Dragoni, et al., “Microservices: Yesterday, Today, and Tomorrow,” Springer, 2017. 🔗 https://doi.org/10.1007/978-3-319-67425-4_12

[7] C. Richardson, “Microservices Patterns: With Examples in Java,” Manning Publications, 2018.

https://microservices.io/patterns/index.html

[8] M. Fowler and J. Lewis, "Microservices Architecture," 2014.
https://martinfowler.com/articles/microservices.html

[9] L. Yu, "Fault Injection Testing for Distributed Systems: A Survey," IEEE Access, 2021.
https://doi.org/10.1109/ACCESS.2021.3052102

[10] P. Alvaro, et al., "Lineage-Driven Fault Injection," ACM SIGMOD, 2015.
https://doi.org/10.1145/2723372.2723715

[11] J. Dean and L. A. Barroso, "The Tail at Scale," Communications of the ACM, 2013.
https://doi.org/10.1145/2408776.2408794

[12] B. Burns, et al., "Borg, Omega, and Kubernetes," Communications of the ACM, 2016.
https://doi.org/10.1145/2890784

[13] N. Kratzke, "Cloud-Native Microservices Architecture," IEEE Software, 2018.
https://doi.org/10.1109/MS.2018.2141024

[14] H. Sabuhi, et al., "Micro-FL: Microservice-Based Federated Learning Platform," 2024.
https://arxiv.org/abs/2109.07802

[15] A. Hannousse and A. Yahiouche, "Securing Microservices and Microservice Architectures: A Systematic Mapping Study," IEEE Access, 2020.
https://doi.org/10.1109/ACCESS.2020.2992418