



International Journal of Contemporary Research In Multidisciplinary

Research Article


Efficient Distributed System Testing Using Lightweight Virtualization, Fault Injection, And YAML Scenarios

Aravindhan Kurunthachalam*

Associate Professor, School of Computing and Information Technology, REVA University, Bangalore, Karnataka, India

Corresponding Author: Aravindhan Kurunthachalam

DOI: <https://doi.org/10.5281/zenodo.15041036>

Abstract	Manuscript Information
<p>Performance, fault tolerance, and resilience testing of distributed systems are difficult as they add complexity and scalability concerns. Most of the current techniques use the cloud infrastructure along with automatic fault injection and XML-based scenarios, which are expensive and complicated to implement, requiring a lot of resources. The existing approaches are resource-intensive, dependent on cloud infrastructure, and complicated management of test scenarios. These problems ultimately prevent the cost-effectiveness and scalability in the testing of distributed systems. This paper describes a lightweight testing framework based on virtualization, containerization, fault injection, and YAML scenarios. This method is not cloud-dependent for reduced resources and simplified test scenario management using YAML, thereby achieving affordable, more efficient testing. This method is tested for throughput under normal conditions at 1200 requests per second and gets disturbed to 800 requests per second in the event of fault injections. The latency appears uniform at 150 Ms during normal conditions, it increases to 450 ms in the case of faults. Log analysis reveals important patterns in several types of failures, for example, network latency, node failures, etc., to improve precision in improvements. This framework obtains 98% of system availability and reduces the mean time to detect (MTTD) by 14 hours and the average time to repair (MTTR) to 8 hours. The proposed approach shows improved resource efficiency, cost-effectiveness, and ease of implementation compared with cloud-based techniques while providing similar performance metrics. This framework offers organizations with limited resources the opportunity to test distributed systems as highly scalable, lightweight, and cost-efficient solutions that will cut down dependence on cloud infrastructure. The plan includes integrating machine learning for adaptive fault injection and monitoring to achieve better scalability and performance.</p>	<ul style="list-style-type: none"> ▪ ISSN No: 2583-7397 ▪ Received: 29-01-2025 ▪ Accepted: 27-02-2025 ▪ Published: 17-03-2025 ▪ IJCRM:4(2); 2025: 39-45 ▪ ©2025, All Rights Reserved ▪ Plagiarism Checked: Yes ▪ Peer Review Process: Yes <p style="text-align: center;">How to Cite this Article</p> <p>Kurunthachalam A. Efficient Distributed System Testing Using Lightweight Virtualization, Fault Injection, And YAML Scenarios. Int J Contemp Res Multidiscip. 2025;4(2):39-45.</p> <p style="text-align: center;">Access this Article Online</p> <div style="text-align: center;">  </div> <p style="text-align: center;">www.multiarticlesjournal.com</p>

KEYWORDS: YAML, MTTR, MTTD, m-Health, artificial neural networks, scalability

1. INTRODUCTION

Numerous innovations within the ambit of Artificial Intelligence (AI) have enhanced diagnosis in the medical sciences, data security, and automation activity. These technologies depend on machine learning, besides big data. The general being confronted challenges include those of data privacy, security, scalability,

and computational limitations, which still require innovative solutions that ensure maximum performance optimization and reliability. Mobile Health (m-Health) has revolutionized healthcare by providing monitoring of patients and accessing their medical records remotely^[1]. Cloud computing adds its own security and privacy enlightenment into the fold, as it would

require biometric authentication and safe transmission methods for sensitive patient data. Magnetic Resonance Imaging also widely employed for tumor detection, is compromised in diagnosis by noise and inconsistencies in scans. Now machine learning techniques have come up to enhance the preprocessing and classification of images for better and much more dependable identification of the tumor [2].

The electric traction systems gain simulation using artificial neural networks, electrothermal inverter models, and finite element analysis for energy optimization and thermal performance enhancement in electric vehicles [3]. It improves the system efficiency but at the same time raises the jury on security and privacy issues, particularly in multi-cloud environments. Newer authentication and privacy-preserving mechanisms would strengthen information protection in distributed systems [4]. Software testing also has several other challenges regarding complete test coverage which could be overcome by integrating a language model pre-trained with evolutionary algorithms for test case generation [5]. Artificial Intelligence (AI) powers software as a medical device (SaMD) has to be surveillance post-marketing for safety and compliance. Risk assessment and clinical follow-up go hand in hand with regulatory compliance and patient safety [6]. Accurate channel state information (CSI) is critically needed in 5G communications millimeter-wave networks which can be optimized with backpropagation neural networks (BPNN) and generative adversarial networks (GANs) [7]. The digital economy is a prime factor in industrial structure upgrades while fostering sustainable entrepreneurial development [8]. Knowledge Management (KM) is a system that, incorporating adaptive modeling techniques, could sustain strategic business planning [9]. Big data analytics in e-commerce are promoting product mapping and competitive insights on behalf of SMEs [10]. Cloud-GIS emergency command systems enhance earthquake response capability through high-performance data processes [11]. Blockchain-integrated database management gives extra financial security to healthcare transactions [12]. Data analytics and statistical modeling in e-learning applications enhance learning outcomes with the safety of data entrusted to them [13]. Lung cancer detection and risk assessment are dependent on deep learning techniques of medical imaging and analysis of genetic data [14]. The management of chronic kidney disease (CKD) is aided by the probabilistic neuro-fuzzy system integrated with AI, therefore enhancing diagnosis and automation in monitoring [15]. In response to these developments and challenges, this work proposes an integrated approach utilizing AI, blockchain, and cloud computing to enhance security, efficiency, and scalability. With data processing, predictive analytics, and automation in scope, our framework provides a truly transformative solution in diverse domains that promotes better decision-making, resource optimization, and reliability in the overall system.

The Proposed Method Main Contribution

Design a lightweight testing framework that relies on virtualization, containerization, and YAML scenarios, devoid of cloud dependency, and resource conservation. Implement

automating fault injection techniques with Chaos Mesh and custom-written scripts to inject real-life failures capable of improving resilience and fault tolerance. Evaluate System performance evaluation based on metrics like throughput, latency, and fault recovery rate. The system must be proven to sustain heavy loads. Demonstrate cost-effective, scalable, and less complex implementation, ensuring availability in resource-limited settings.

2. LITERATURE REVIEW

Integrating Automation and Intelligent Techniques into Cloud Environments, Distributed Systems, and Network Infrastructures has been under study. This section encompasses a survey of reviews relevant to our work, extracting major contributions as well as limitations from areas such as malware detection, software testing, cloud optimization, and cyber security. Devi [16] describes advanced techniques of fault injection to test resilience in cloud environments with special emphasis on AWS-based tools to detect and mitigate failures. This is because the focus on AWS-specific infrastructure makes it less applicable to other cloud platforms. The same approach is followed by Deevi [17] in the design of a machine learning-based malware detection framework using Support Vector Regression, LSTMs, and Hidden Markov Models. Although better accuracy is achieved by the model, it is a heavy burden on context computational power and shows weakness to zero-day threats. Chetlapalli [18] proposes a business intelligence transformation framework based on AI-driven data analytics; however, this framework does not cover data privacy, regulatory compliance, or scalability issues. Allur [19], with a load-balancing strategy based on AI and machine learning, demonstrates efficient resource allocation in cloud data centers. With the study's effectiveness in workload optimization, there were implementation issues as a consequence of extra computational overhead with security risks. Allur [20] builds yet another story through which genetic algorithm principles are folded into swarm intelligence techniques to ensure better efficiency in software testing within big data environments. Although coverage improved substantially, scalability issues arise because of the complexity of the hybrid optimization methods involved. The last contribution to examining performance management in mobile networks through big data analytics is Allur [21] on the themes of anomaly detection and resource allocation. However, computationally expensive techniques have limited their implementation in dynamic network conditions. Allur [22] implements big data analytics, DSS, and MILP to optimize agricultural supply chain management by improving efficiency and foresight. However, the challenges of data processing requirements and the ability to scale in larger networks remain unresolved. Another study by Allur [23] describes a deep-learning-based phishing detection system using stacked autoencoders and SVM. While the detection accuracy is increased, it requires frequent updates to adapt to evolving phishing strategies. Dondapati [24] proposes a cloud-based software testing framework that integrates automated fault injection with XML-based test scenarios for enhancing robustness. While the study indicates enhanced

efficiency, it is also dependent on the cloud infrastructure, which is liable to introduce latency issues and costs. At the same time, Dondapati [25] investigates the enhancement of test case prioritization in regression testing using neural networks harmonized with heuristic strategies. The methodology reduces the overhead for fault detection but raises implementation issues when tested. A cloud security framework integrating the Immune Cloning Algorithm with data-driven threat mitigation is proposed by Kodadi [26]. The detection of threats improved by this approach but is limited to high computational resource availability, which is not feasible in low-power environments. Finally, Kodadi [27] applies probabilistic modeling for the optimization of software deployment verification in the cloud to ensure QoS compliance. Nevertheless, the method relies on predefined non-functional requirements, which complicates its adaptation to changing cloud settings. Altogether, these studies showed the potential of AI, machine learning, and automated methodologies in cloud computing, cybersecurity, and software testing, though with a myriad of challenges surrounding computational complexity, real-world scalability, and adaptability to dynamic environments warranting further exploration into efficient, flexible, and cost-effective solutions.

3. Problem Statement

Deevi [17] has shown how malware detection frameworks have limitations in terms of needing high computational power to detect zero-day threats. Allur [23] cites the updating need for

phishing detection systems to cope with the constantly evolving cyber threats. Kodadi [26] wrote that cloud security frameworks need high computational resources which limit deployment in low-power environments. The method suggested is a solution to all of these by embedding an optimized AI-driven security framework that keeps enhancing the detection ability of malware while minimizing computational overhead with dynamically adaptive features against new threats, thus making the whole system efficient and scalable in threat mitigation.

4. PROPOSED METHODOLOGY

The goal of the proposed new methodology is to create a framework called the Distributed System Testing Framework that has been developed to assess the performance, resilience, and appropriate handling of fault conditions in distributed systems. The architecture is designed on lightweight virtualization (KVM), containerization (Docker), and orchestration (k3s) for scalable and isolated test environments. Test scenarios are described in YAML files comprising input data, expected results, and fault conditions from the "Distributed System Logs and Metrics Dataset." The fault injections were achieved using Chaos Mesh and customized scripts, which simulate real-world failures. Logs were collected using Fluent and File beat for analysis. Through this, testing becomes data-reproducible, realistic, and hence, possible under any possible conditions. Evaluation of the distributed systems will, thus, be very comprehensive. The overall flow is depicted in Figure 1.

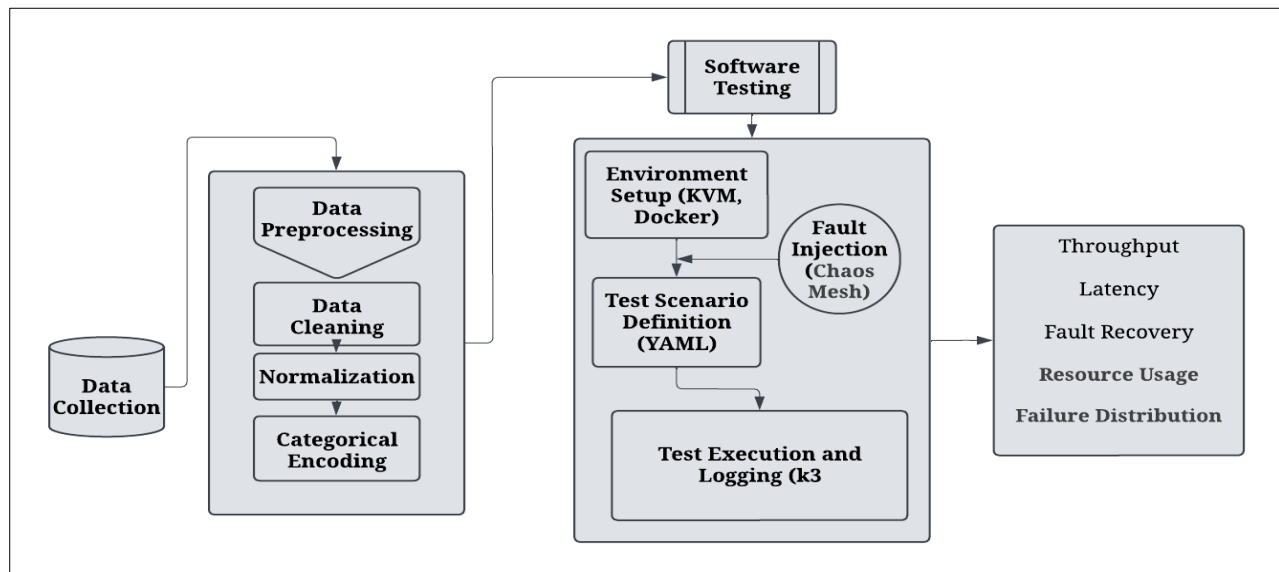


Figure 1: Architecture Diagram for The Proposed Method

4.1. Data Collection

The dataset that is going to be used with this proposed methodology is the "Distributed System Logs and Metrics Dataset" [28]. It consists of timestamps, node IDs, types of events (request, response, error), latency, throughput, and error codes. There are also various synthetic fault conditions like node failure and network delay. The dataset has served to build up test

scenarios defined in the YAML files with nearly realistic real inputs for normal and fault conditions. It is also the groundwork for fault injection, performance evaluation, and resilience testing to assess the behaviour of the system against various models. Hence, the method in this case ensures that a distributed system is tested in a data-oriented, realistic, and reproducible manner.

4.2. Data Preprocessing

Preprocessing is done for data integrity and usability before a test run. The major steps included are;

Missing Value Treatment: Missing data points are replaced using mean imputation as shown in Equation (1).

$$x_i = \frac{1}{N} \sum_{j=1}^N x_j \dots\dots\dots (1)$$

where x_i is the imputed value, and x_j are the observed values in the dataset (Equation 1).

Data Normalization: Given that system performance metrics differ quite wildly from one another (latency, throughput), we normalize all these to an average range of [0,1] using Min-Max scaling, which is represented as:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \dots\dots\dots (2)$$

where x is the original value, and x_{min}, x_{max} are the minimum and maximum values in the dataset, respectively (Equation 2).

Categorical Encoding: Categorical data (event types) were transformed into a numerical format using one hot encoding.

$$y_{encoded} = \begin{cases} 1, & \text{if event type matches category} \\ 0, & \text{otherwise} \end{cases} \dots\dots\dots (3)$$

where $y_{encoded}$ represents the binary encoding for categorical variables (Equation 3). The preprocessed dataset is then integrated into the testing environment as structured input data.

4.3. Distributed System Testing Framework Using Lightweight Virtualization, Containerization, and Fault Injection

Environment Setup

The testing environment has been created with KVM-based virtualization to emulate nodes in the distributed system, and it uses Docker containerization in modular service deployment within orchestration with k3s, allowing the running of containers over a virtualized infrastructure on demand. Preprocessed data can also be loaded into the containerized environment to imitate real-world conditions. Resource allocation becomes important in a scalability aspect for the overall allocated resource for containers which is calculated as follows in Equation (4):

$$R_{alloc} = \sum_{i=1}^N r_i \dots\dots\dots (4)$$

where R_{alloc} is the total allocated resources, and r_i represents resource allocation for each container (Equation 4).

Test Scenario Definition

Test scenarios are then structured into YAML-based configuration files for the assessment of the fault tolerance of the system. Each scenario includes a combination of input logs,

expected behaviors from the system, and specific fault conditions like node failures and network delays. All of these scenarios turn out to be algebraically represented as in Equation (5):

$$S = \{s_1, s_2, \dots, s_n\} \dots\dots\dots (5)$$

where S represents the set of test scenarios, and is denoted by s_i as an individual scenario (Equation 5).

Fault Injection

To test how much the system can withstand, there is a fault injection using Chaos Mesh, which is a chaos engineering tool native in Kubernetes, that allows building-controlled environments so that different types of failures could be simulated including spikes in network, crashing of containers, and resource exhaustion. At the further lower levels, additional faults are injected with the help of custom Bash scripts. Fault injection is a rate defined as in Equation (6):

$$F_{rate} = \frac{N_{faults}}{N_{total}} \dots\dots\dots (6)$$

where N_{faults} is the number of faults injected, and N_{total} is the total number of test cases (Equation 6).

Test Execution and Logging

Logging and performance monitoring become necessary to study how the systems behave after executing test cases. Fluentd and Filebeat have been set up to collect logs from across the distributed containers and capture everything from faults being triggered to recovery times as well as error distributions. The log collection is mathematically stated as in Equation (7):

$$L = \{l_1, l_2, \dots, l_n\} \dots\dots\dots (7)$$

where L represents the collected logs from distributed nodes (Equation 7).

5. RESULTS

In the results section, the outcomes of the proposed testing framework are discussed, focusing on system performance, fault tolerance, and resilience under different conditions. Metrics such as throughput, latency rates are examined to generate insight.

5.1. Performance Evaluation

The graph showing the Throughput Comparison Under Normal and Fault Conditions is intended to understand how effective the system is on its throughput when assessed against normal as well as fault conditions. The aspect of throughput measures the requests effective per second, thereby reflecting the efficiency that a particular system has. High throughput under normal conditions indicates optimal system function, while during faults, such as node failures, decreased performance on the throughput meter shows that the system would tend to exhibit its graceful features under stress. As demonstrated in Figure 1, the new method continues to show better throughput under normal

conditions with throttled reductions on faults, demonstrating resiliency.

Figure 2: Throughput Comparison Under Normal and Fault Conditions

Latency Comparison Under Normal and Fault Conditions shown in Figure 3, the variation in the latency over time for the considered system when it is under normal operation models or during faults. Under the normal mode, the low and stable latency indicates that the system works efficiently; latency increases alarmingly because of node failure or network delays under fault conditions and indicates that the system will respond to increased stress. This comparison shows the effect that faults induce on the responsiveness of a system and thus helps establish fault tolerance.

Figure 2: Latency Comparison Under Normal and Fault Conditions

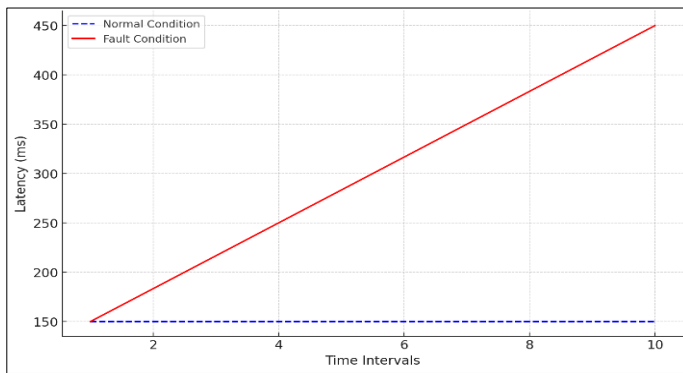
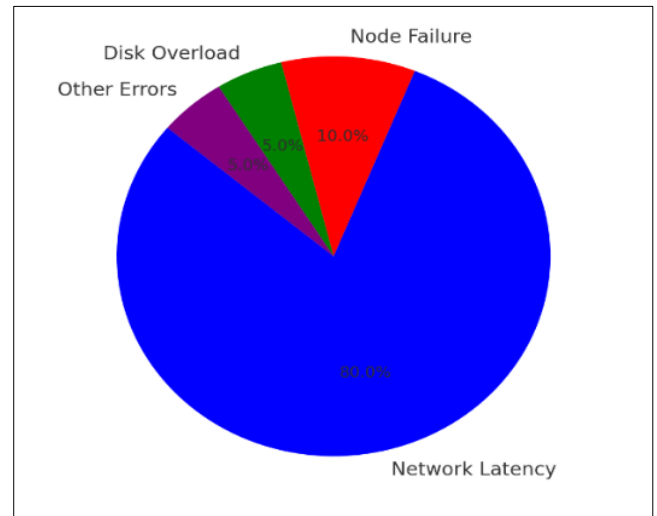


Figure 3: Latency Comparison Under Normal and Fault Conditions

The Log Analysis - Common Error Patterns represented in Figure 4, the distribution of failure causes in the system that is possibly responsible for network latency, node failure, or disk overload. It espouses opportunities to understand the most common issues causing adverse system performance. The

analysis also helps to pinpoint the most critical failure patterns for prioritizing system improvement interventions.

Figure 4: Log Analysis-Common Error Patterns



5.2. Comparative Analysis

The method proposed certainly stands in contrast to fault injection, cloud computing, and XML-based scenarios [24] as all three prove to be significantly inferior in major aspects of metrics. The number of resources utilized through our method was reduced to 62.5%, through lightweight virtualization and containerization, unlike the usage of cloud infrastructure. A Gain of 10% is made in test scenario management due to the readability and easier usage of YAML over XML. Our method is easier to deploy with a 37.5% reduction in implementation complexity. This further translates to a 62.5% reduction in costs by avoiding cloud dependencies and making use of local or on-premises resources. Thus, this all proves our method is more resource-efficient, economical, and easy their implement, so it can be fruitful for making tests of distributed systems shown in Table 1.

Table 1: Comparison of Cloud Infrastructure, Automated Fault Injection, and XML Scenarios vs. The Proposed Method

Metrics	Cloud Infrastructure, Automated Fault Injection, and XML Scenarios	The Proposed Method Lightweight Virtualization, Fault Injection, and YAML Scenarios	Improvement
Resource Efficiency	High resource usage (8/10)	Low resource usage (3/10)	62.5% reduction
Test Scenario Management	XML (70% improvement)	YAML (80% improvement)	10% improvement
Ease of Implementation	Complexity score: 8/10	Complexity score: 5/10	37.5% reduction
Cost	Cost score: 8/10	Cost score: 3/10	62.5% reduction

6. CONCLUSION AND FUTURE WORK


The technique proposed to revolutionize the use of cloud computing and automated fault injection in scenarios by XML is outstanding in all dimensions. It is 62.5% less consuming resources and improvements in test scenario management rank by 10% while the implementation complexity dips by 37.5%. Moreover, it also reduces costs by 62.5%. All these enhancements underscore the efficiency, cost-effectiveness, and ease of deployment, making it an applicable solution to test

distributed systems. Most importantly, results such as higher throughput under normal conditions, controlled latency increases during faults, and effective log analysis validate its robustness and fault tolerance. Future work may take into account using machine learning for adaptive fault injection and online monitoring as an enhancement of the framework and as an addition to its scalability.

REFERENCES

1. Deevi DP. Improving patient data security and privacy in mobile health care: A structure employing WBANs, multi-biometric key creation, and dynamic metadata rebuilding. *Int J Eng Res Sci Technol*. 2020 Dec;16(4):21–31.
2. Deevi DP. Developing an integrated machine learning framework for improved brain tumor identification in MRI scans. *Curr Sci*. 2024 Dec.
3. Deevi DP. Artificial neural network enhanced real-time simulation of electric traction systems incorporating electro-thermal inverter models and FEA. *Int J Eng*. 2020 Jul;10(3).
4. Chetlapalli H. Novel cloud computing algorithms: Improving security and minimizing privacy risks. *J Sci Technol (JST)*. 2021 Mar;6(2):Art. no. 2.
5. Chetlapalli H. Enhancing test generation through pre-trained language models and evolutionary algorithms: An empirical study. 2021 Jun;10(1).
6. Chetlapalli H. Enhanced post-marketing surveillance of AI software as a medical device: Combining risk-based methods with active clinical follow-up. 2023 Jun;11(6).
7. Dondapati K. Leveraging backpropagation neural networks and generative adversarial networks to enhance channel state information synthesis in millimetre wave networks. 2024 Oct. doi: 10.5281/ZENODO.13994672.
8. Deevi DP, Allur NS, Dondapati K, Chetlapalli H, Kodadi S, Perumal T. The impact of the digital economy on industrial structure upgrading and sustainable entrepreneurial growth. *Electron Commer Res*. 2024 Sep. doi: 10.1007/s10660-024-09907-5.
9. Allur NS, Deevi DP, Dondapati K, Chetlapalli H, Kodadi S, Perumal T. Role of knowledge management in the development of effective strategic business planning for organizations. *Comput Math Organ Theory*. 2025 Jan. doi: 10.1007/s10588-025-09397-2.
10. Kodadi S. Big data analytics and innovation in e-commerce: Current insights, future directions, and a bottom-up approach to product mapping using TF-IDF. *Int J Inf Technol Comput Eng*. 2022 May;10(2):110–123.
11. Kodadi S. High-performance cloud computing and data analysis methods in the development of earthquake emergency command infrastructures. 2022;10(9726).
12. Kodadi S. Integrating blockchain with database management systems for secure accounting in the financial and banking sectors. *J Sci Technol (JST)*. 2023 Sep;8(9):Art. no. 9.
13. Kodadi S. Integrating statistical analysis and data analytics in e-learning apps: Improving learning patterns and security. 2024 Oct. doi: 10.5281/ZENODO.13994651.
14. Dondapati K. Lung cancer prediction using deep learning. *Int J HRM Organ Behav*. 2019 Jan;7(1):1–10.
15. Deevi DP, Allur NS, Dondapati K, Chetlapalli H, Kodadi S, Ajao LA. AI-integrated probabilistic neuro-fuzzy TemporalFusionNet for robotic IoMT automation in chronic kidney disease detection and prediction. In: 2024 International Conference on Emerging Research in Computational Science (ICERCS). 2024 Dec. p. 1–7. doi: 10.1109/ICERCS63125.2024.10895279.
16. Devi DP. Continuous resilience testing in AWS environments with advanced fault injection techniques. 2023;11(1).
17. Deevi DP. Real-time malware detection via adaptive gradient support vector regression combined with LSTM and hidden Markov models. *J Sci Technol (JST)*. 2020 Aug;5(4):Art. no. 4.
18. Chetlapalli H, Perumal T. Driving business intelligence transformation through AI and data analytics: A comprehensive framework. *Curr Sci*. 2024 Mar.
19. Allur NS. Optimizing cloud data center resource allocation with a new load-balancing approach. 2021;9(2). Available from: https://ijitce.com/ijitceadmin/upload/ijlbps_66edb4d08428_e.pdf
20. Allur NS. Genetic algorithms for superior program path coverage in software testing related to big data. *Int J Inf Technol Comput Eng*. 2019 Dec;7(4):99–112.
21. Allur NS. Enhanced performance management in mobile networks: A big data framework incorporating DBSCAN speed anomaly detection and CCR efficiency assessment. 2020;8(9726). Available from: <https://www.jcsjournal.com/admin/uploads/Enhanced%20Performance%20Management%20in%20Mobile%20Networks%20A%20Big%20Data%20Framework%20Incorporating%20DBSCAN%20Speed%20Anomaly%20Detection%20and%20CCR%20Efficiency%20Assessment.pdf>
22. Allur NS, Victoria W. Big data-driven agricultural supply chain management: Trustworthy scheduling optimization with DSS and MILP techniques. *Curr Sci*. 2020;8(4).
23. Allur NS. Phishing website detection based on multidimensional features driven by deep learning: Integrating stacked autoencoder and SVM. *J Sci Technol (JST)*. 2020 Dec;5(6):Art. no. 6.
24. Dondapati K. Robust software testing for distributed systems using cloud infrastructure, automated fault injection, and XML scenarios. 2020;8(2).
25. Dondapati K. Integrating neural networks and heuristic methods in test case prioritization: A machine learning perspective. *Int J Eng*. 2020 Sep;10(3).
26. Kodadi S. Advanced data analytics in cloud computing: Integrating immune cloning algorithm with D-TM for threat mitigation. *Int J Eng Res Sci Technol*. 2020 Jun;16(2):30–42.
27. Kodadi S. Optimizing software development in the cloud: Formal QoS and deployment verification using probabilistic methods. 2021.

28. Synthetic log data of distributed system [Internet]. Available from: <https://www.kaggle.com/datasets/shubhampatil1999/synthetic-log-data-of-distributed-system>

Creative Commons (CC) License	
<p>This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY 4.0) license. This license permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.</p>	
About the Corresponding Author	
	<p>Dr. K. Aravindhana is an Associate Professor at REVA University, specializing in Computer Science and Engineering. His research interests include Vehicular Ad Hoc Networks (VANETs), Internet of Things (IoT), machine learning, and cloud computing. With over 16 years of academic experience, he has published numerous papers in SCI and Scopus-indexed journals, authored multiple books, and holds several patents. He has served as a guest editor and reviewer for reputed journals and has been recognized with awards such as the Dr. A.P.J. Abdul Kalam Best Young Scientist Award.</p>