International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2024/57 Volume 11, Issue 6, November – December (2024)

RESEARCH ARTICLE

Design of an Integrated Model Using Hybrid Autoencoder and LSTM for Fault Tolerance and Load Balancing in Cloud Environments

Nahita Pathania
School of Computer Science and Engineering, Lovely Professional University, Phagwara, India.
nahita.19372@lpu.co.in

Balraj Singh
School of Computer Science and Engineering, Lovely Professional University, Phagwara, India.
balraj.13075@lpu.co.in

Received: 17 October 2024 / Revised: 16 December 2024 / Accepted: 24 December 2024 / Published: 30 December 2024

Abstract – Large and complex topologies in modern cloud environments really call for factors such as fault tolerance and efficient usage of resources. Current fault detection and load balancing techniques are often found to be insufficient due to known limitations of very high false positives, late detection, and great redundancy overheads that often-become bottlenecks for performance. To this effect, this work offers a new hybrid fault-tolerant load-balancing framework with an integration of multiple advanced techniques as follows: Hybrid Autoencoder-Based Anomaly Detection (HAAD), Task-Level Replication Using Intelligent Redundancy Allocation (TRA-IRA) and Long Short-Term Memory (LSTM) networks for proactive failure prediction operations. HAAD discovers known and unknown faults by learning to discern the normal behavior of a system using unsupervised autoencoders, which has achieved 97-98 percent accuracy in fault detection. TRA-IRA dynamically allocates redundant replicas based on task priority and real-time resource health predictions, reducing replication overhead by 20% while maintaining a task completion rate of 99.5%. The LSTM network predicts imminent failures by analysing temporal patterns in system metrics that enable task migration up to 45 min before with 95-96% prediction accuracy. All these techniques are easily integrable with Adaptive Resource Reallocation via Genetic Algorithm (ARR-GA) with respect to optimal scheduling. The Batfly Algorithm is used in an attempt to manage the task. Therefore, due to the integration of these approaches, it presents very efficient performance by increasing by 45% the fault tolerance strength and enhancing the reliability of a system by 50%. The response timestamp along with makespan reduced between 15 to 20%. This model will offer a scalable, dynamic, and robust method of cloud load balancing to augment critical gaps in fault tolerance and optimizations of resources.

Index Terms – Fault Tolerance, Autoencoder, LSTM Networks, Load Balancing, Redundancy Allocation, Scenarios.

1. INTRODUCTION

The spreading of cloud computing infrastructures relies on huge distributed virtualized resources and workloads. Therefore, the use of efficient mechanisms in the detection of faults as well as load balancing is important in maintaining high availability, performance, and fault tolerance in cloud services. Failures in the VMs, physical nodes, or network components can disrupt service reliability, and it may result in considerable periods of downtime and disgruntled customers. Traditional redundancy allocation and threshold-based fault detection techniques fail in large-scale clouds as they do not adapt to changing operational conditions. The workloads, changes in the consumption of resources, and the probabilities of faults all change dynamically in a cloud [1, 2, 3]. Such a demand creates the need for highly advanced and intelligent models that can detect or predict faults adaptively, thereby optimizing resource consumption in parallel. Current solutions [4, 5, 6] have adopted either reactive mechanisms or fixed threshold-based approaches toward fault detection, where false positives are often very high, and failures get unresponsive for a long time. In reactive approaches, models respond only after a failure has happened and thus start causing service interruptions. Furthermore, static models cannot dynamically adapt to the changing cloud workloads, which causes inefficient use of resources. Additionally, the replication strategies for tasks in traditional models often bring about redundancy, creating overhead and degradation in performance. These limitations do, however, indicate the urgent need to have an integrated framework that will proactively predict failures, isolate faults in real-time, and efficiently manage task redundancy and load balancing. This paper introduces a new fault-tolerant load-balancing framework called Hybrid Autoencoder-Based Anomaly

Detection combined with task-level replication using intelligent redundancy allocation and long short-term memory networks for proactive failure prediction in cloud environments. The proposed HAAD method utilizes autoencoders from deep learning to learn the baseline performance of cloud resources towards anomalies in system behavior that are known or unknown. Such a redundant resource allocation regarding dynamic prediction of the priority of tasks and the health of VMs would ensure that critical tasks remain protected while incurring redundancy overhead. Simultaneously, time-series analysis on system metrics will be carried out using LSTM to capture the temporal patterns of failures in resources before they occur. All these models are imbedded in an adaptive load balancing framework, which would make the cloud infrastructure adapted and changed within real-time stamp to varying conditions, with high reliability and efficiency. Adaptive Resource Reallocation via Genetic Algorithm (ARR-GA) is further perfected and enhanced within this framework, optimizing task scheduling and resources according to failure prediction or detection. The Batfly Algorithm functions efficiently on multi-dimensional optimization tasks with scheduling tasks across the infrastructure of clouds. This holistic approach to design issues makes it highly fault-tolerant and optimizes resources to make better resource utilization, which is something conventional methods fail to achieve in the process.

Cloud computing has emerged as the core of modern digital infrastructure with scalable, on-demand access to computational resources. However, the complexity of cloud environments is making it harder to maintain high availability, reliability, and efficient resource usage. Among the most important factors influencing the quality of cloud services, perhaps the most significant factor is system faults. System faults can appear in totally random places within VMS, physical nodes, or network elements. They can lead to many downtimes, waste a lot of resources, and unhappy users if it doesn't address these faults soon and efficiently. The dynamic nature of cloud workloads as well as the requirement of optimal performance call for intelligent load balancing that adapts to varied conditions with minimal overhead.

Most of the traditional fault tolerance approaches, including threshold-based methods and reactive techniques, fail for most large-scale and dynamic clouds. The reactive approaches react after faults have actually occurred, and service interruption is caused by them; the static thresholds are insensitive to changing workload patterns and varied resource consumption patterns. Adding to this, resource over-allocation has resulted in conventional redundancy solutions, hence less effective and higher operational costs. These needs express a demand for new intelligent frameworks capable of proactive fault detection, effective and dynamic load balancing, and optimized resource allocation.

A novel fault-tolerant load-balancing framework with the combination of three state-of-the-art techniques is proposed in this paper, namely Hybrid Autoencoder-Based Anomaly Detection, Task-Level Replication Using Intelligent Redundancy Allocation, and Long Short-Term Memory networks. The use of deep learning by HAAD enables both known and unknown anomaly detection and reduces false positives for fault detection. TRA-IRA optimizes task replication using dynamically allocated redundancy based on priority and real-time prediction of resource health. The LSTM network makes the framework proactive with respect to predicting impending failures from temporal patterns in system metrics before any disruption occurs. Then, adaptive resource reallocation using genetic algorithms builds on these techniques by optimal task scheduling and utilization of resources.

The proposed model will, for the first time, balance the tasks of fault tolerance and resource efficiency, which have never been solved in any previously proposed solution. These are achieved in the improvement of task completion rates, with a reduction in recovery times as well as in operational overhead, while remaining robust in fault detection that also improves precision in predicting failures and better resource usage. This work thus contributes research not only in designing but also in demonstrating the superior performance of an integrated framework through comprehensive evaluations carried out on benchmark datasets such as Google Cluster Data and Bitbrains Resource Traces.

1.1. Motivation & Contribution

The motivation for this work comes from the increasing complexity of cloud environments and the inadequacies of traditional fault-tolerance mechanisms in maintaining both service reliability and resource efficiency. With cloud infrastructures now supporting all sorts of applications, from enterprise-level services to latency-sensitive IoTs, the potential to identify failures that will happen promptly and predict those likely to happen within a short period is of paramount importance. Such traditional fault-detection methods, which are majorly reactive or threshold-based, are considered insufficient to handle the needs of modern clouds, as faults are notorious for occurring unpredictably, and the resource consumption patterns are highly diverse in the course of the process. These limitations not only lead to service outages but also lead to inefficient resource usage due to the over-allocation of redundant tasks leading to unnecessary overheads. What this contribution achieves is the design and development of an integrated fault-tolerant load-balancing framework, bringing together strengths of three approaches that go as such: hybrid autoencoder-based anomaly detection, intelligent task-level replication, and LSTM-based failure prediction. The three individual approaches individually fill three critical gaps in the current fault-detection and resource-allocation systems. Hence, the paper outlines an integrated model that collates such techniques into a uniform solution that serves to increase the accuracy of fault detection, minimize redundancy overhead, and optimize resource usage. The HAAD method uses deep learning autoencoders to identify known as well as unknown anomalies with fewer false positives. TRA-IRA allocates the redundancy intelligently depending on the priority and the predicted probability of failure of tasks, thereby minimizing the unnecessary replication of critical components. Lastly, LSTM networks provide a proactive mechanism for failure prediction where the system can migrate tasks away from resources that are likely to fail, thereby reducing service disruption. In this sense, combining these methods with a load-balancing framework supported by ARR-GA for resource reallocation and the Batfly Algorithm for scheduling of tasks would represent an important step forward in cloud fault tolerance mechanisms. This approach not only enhances the robustness and fault tolerance of cloud environments but also contributes to better resource utilization, resulting in a reduction of overall operational costs and improved performance.

The rest of this paper is structured as follows: Section 2 presents an in-depth overview of related work that discusses techniques involving fault tolerance and load-balancing techniques available in cloud environments. Section 3 presents the proposed integrated model and all its technical underpinnings, including HAAD, and TRA-IRA with the LSTM network for proactive failure prediction. Section 4 details the experimental setup along with metrics that are used for evaluation in this process. In section 5 we discuss the proposed model, and then finally, Section 6 concludes the paper, summarizing the contributions and directions for future work sets.

2. RELATED WORK

The past few years have seen the complexity and scale of distributed systems pushing fault-tolerant cloud computing forward. This review examines some of the diverse methods ranging from the usage of the genetic algorithm and deep learning models to blockchain-based architectures and consensus mechanisms. All the selected papers, each had presented a different viewpoint about the challenges this is causing due to system failure and disruption, and each collectively gives a wholesome view of the state of the art in the present scenario of fault tolerance and load balancing in cloud environments. The summary of existing approaches is provided in Table 1.

Ray et al. [1] present a proactive fault-tolerant technique that builds up the reliability of cloud services within federated environments. Their approach to research deals with virtual machine migration, with an attempt to decrease the costs of migration while keeping a reliability/performance trade-off. Rehman et al. [2] extends the scope of fault-tolerance metrics by investigating both system-level and component-level metrics on a range of cloud and edge computing systems, including 5G networks. Their work provides evidence of the growing need for multi-level fault-tolerance frameworks for new cloud technologies. Moreover, Tawfeeg et al. [3] focus on the utilization of reactive fault-tolerance techniques and dynamic load balancing through a systematic literature review. The paper thus illustrates the need for adaptive mechanisms for load balancing that could react to the real-time change in workload. Moreover, it points out the necessity for proactive as well as reactive fault tolerance mechanisms. Dehury et al. [4] developed a new concept called RRFT, which uses rank-based resource allocation combined with fault tolerance. Their approach incorporates Markov decision processes for resource-aware management on cloud platforms; this is an important contribution to resource-aware fault-tolerant strategies. Similarly, the authors, Ramesh et al., [5] have designed a hybrid genetic algorithm with simulated annealing to minimize latency in virtual machine migrations that aims at proactive fault tolerance. This method employs the use of integer linear programming so as to avoid critical delay in migration as it is crucial in cloud environments where service continuity has to be taken care of. Saxena et al. [6] present a framework that manages elastic resources integrating fault tolerance with failure prediction so as to increase the availability of services offered through clouds. This is the contribution that is essential in the integration of predictive models into systems with fault tolerance and optimizing recovery timestamps with those used with resources. Mushtaq et al. [7] have proposed a fault-tolerant scheduling method to be used in cloud environments to enhance resource utilization, which integrates neighboring reservations into task scheduling. This approach results in lowering the chances of errors in the task-allocation process because neighboring reservations are preserved as backup copies, thus reducing the rate of failures of tasks. Mudassar et al. [8] work on latency-aware fault tolerance at edge point runs of IoT applications. Their adaptivity strategy falls in line with the new trend of edge computing and low-latency fault-tolerance strategies are key components in distributed IoT systems. Chen et al. [9] continue in this development with deep reinforcement learning as they enhance its applicability to serve function chain optimizations in SDN/NFV-enabled environments for clouds, and then demonstrate the value of machine learning in the fault-tolerant optimization of services in clouds. Jing et al. [10] also report a consensus protocol specifically designed for edge computing environments with byzantine resilience as the primary focus. Their scheme makes the collaborative services of such networks fault-tolerant, even under malicious attacks or disruptions. Tang [11] focuses on the cost-efficiency of scheduling scientific workflows across multi-cloud systems, which ensures scheduling with faults and reliability-aware strategies. This work is used instrumentally in order to balance performance with cost, especially in environments that deal with scientific computing, where the importance of task scheduling is immense. Zhao et al. [12] come up with a model based on blockchain technology in order to ensure data security in the context of environmental monitoring systems. They cite the intersection of fault tolerance and block chain technology. Their model integrates a practical Byzantine fault tolerance mechanism, which ensures the secure and fault-tolerant acquisition and monitoring of data in cloud environments. Yao et al. [13] proposed a hybrid scheduling strategy for deadline-constrained tasks within cloud systems using a combination of resubmission and replication strategies, which shows how relevance in choosing hybrid approaches to ensure the fault tolerance most especially with time-critical applications. Meng et al. [14] discuss service-oriented reliability modeling using Markov models to enhance the optimization of public cloud system reliability. Their work further goes on to emphasize the need for model-based formal techniques in failure prediction and failure management. Zhao et al. [15] present a new address and routing architecture specifically designed for cloud-service data centers with a load balancing and fault tolerance perspective. The architecture provided scalable data center systems accompanied by fault tolerance routing mechanisms. Likewise, Ahmad et al. [16] deal with the issue of workflows and design an approach for the fault-tolerant scheduling system for cloud computing resources. Their work especially addresses the requirements in managing large-scale workflows such as CyberShake and Montage. Yao et al. [17] addressed this issue from a slightly different angle; they looked at failure-aware elastic scheduling which ensures that there are fault-tolerant workflows over fat-tree topologies for common cloud data centers. Chen et al. [18] proposed secure and fault-tolerant storage for cloud-edge collaborative systems. They enhanced the resiliency and performance of edge storage systems used erasure coding and management using SDN-based storage. Al-Makhlafi et al. [19] introduce RibsNet, a two-layer cloud data center architecture, where scalability and cost efficiency accompanied by fault tolerance are emphasized. This double-centric design provides further incremental scalability and recovery of faults in large data centers. Ahmed et al. [20] introduce blockchain-based security and quality-of-service mechanisms in vehicular IoT networks; fault tolerant and low latency by integrating the edge computing process is provided.

Related works in the area of fault tolerance and load balancing in cloud environments are some diverse methodologies, each of which has its advantages and limitations. Ray et al. [1] were the first to propose a proactive fault-tolerance technique suitable for cloud federation environments reliant on virtual machine migration and enhancing reliability while reducing cost. This architecture resulted in improving system reliability but was scaled in very few systems because of the extra overhead related to VM migration in such systems. Rehman et al. [2] designed a fault-tolerance framework for cloud and edge systems by utilizing system and component-level metrics, making it relevant to emerging technologies like 5G. Although the above framework enhanced resilience remarkably, this failed to address hybrid-cloud configuration and lacked proactive failure-prevention mechanisms. Tawfeeg et al. [3] give systematic reviews that have considered dynamic load balancing along with reactive fault tolerance. The work addresses the importance of dynamic load management but could not shed light on extensification in experimentation and focuses on only reactive approaches. Dehury et al. [4] have introduced the RRFT framework, wherein resource-aware fault tolerance is carried out by ranking the components with the help of a Markov decision process. Even though this design has increased the reliability of tasks as well as utilized resources better, it came up with inefficiencies of dynamic workload scenarios and enhanced latency for reassigning resources.

Saxena et al. [6] proposed a framework for elastic resource management that integrates failure prediction. This strategy does not fully address redundancy overhead and its trade-off with fault tolerance although it enhances the availability and optimizes the usage of resources.

Mushtaq et al. [7] presented a fault-tolerant scheduling technique by using neighboring reservations, which minimized task failures with resource utilization. Even though the method improved the fault tolerance, it was not as adaptive to dynamic configurations because it mainly used guaranteed neighboring reservations. Chen et al. [9] adopted deep reinforcement learning to optimize the service function chaining for the cloud environment that is enabled with SDN and NFV sets. This improved quality of service along with fault tolerance; however, the implementation has a drawback regarding computation overhead and adaptability to huge systems.

Table 1 Summary of Existing Approaches

Paper Reference	Methodology	Key Findings	Limitations
Ray et al. [1]	Proactive Fault-Tolerance for Cloud Federation: Uses VM migration to enhance reliability and reduce migration costs.	Improved reliability and fault tolerance in federated cloud environments.	High computational overhead; scalability challenges in large federations.

Rehman et al. [2]	Fault-Tolerance Framework for Cloud & Edge Systems: Uses system and component-level metrics for fault tolerance.	Enhanced resilience in 5G and edge environments.	Limited to specific use cases like 5G; lacks support for hybrid environments.
Tawfeeg et al. [3]	Dynamic Load Balancing with Reactive Fault Tolerance: Systematic literature review emphasizing load balancing and reactive fault tolerance.	Highlights adaptability under varying workloads.	Focuses solely on reactive approaches; lacks predictive mechanisms.
Dehury et al. [4]	RRFT: Markov decision process for rank-based resource allocation.	Improved reliability and optimized task allocation.	Ranking inefficiencies under dynamic workloads.
Ramesh et al. [5]	Hybrid Genetic Algorithm & Simulated Annealing for VM Migration: Combines optimization techniques for fault tolerance.	Minimized latency in VM migrations; improved recovery.	Computational complexity restricts scalability.
Saxena et al. [6]	Elastic Resource Management with Failure Prediction: Integrates prediction for dynamic elasticity.	Improved service availability and resource usage.	Limited focus on optimizing redundancy overheads.
Mushtaq et al. [7]	Fault-Tolerant Task Scheduling with Neighboring Reservations: Uses neighboring reservations for improved scheduling.	Enhanced utilization and reduced task failures.	Requires neighboring resources, limiting flexibility.
Mudassar et al. [8]	Latency-Aware Fault Tolerance for IoT in Edge Computing: Adaptive strategies for latency-sensitive tasks.	Improved latency handling and fault resilience in IoT systems.	Applicability is limited to IoT workloads.
Chen et al. [9]	Deep Reinforcement Learning for SFC Optimization in SDN/NFV Clouds: Optimizes service function chaining.	Improved QoS and elasticity in SDN/NFV clouds.	High computational costs; limited scalability.
Jing et al. [10]	Byzantine Resilient Consensus for Edge Computing: Ensures fault tolerance under adversarial conditions.	Enhanced reliability in collaborative edge services.	High complexity in consensus mechanisms.
Tang et al. [11]	Reliability-Aware Workflow Scheduling for Multi-Cloud Systems: Cost-efficient scheduling with reliability considerations.	Balanced cost and performance in scientific workflows.	Limited support for highly dynamic workloads.
Zhao et al. [12]	Blockchain-Based Data Security for Environmental Monitoring: Uses Byzantine fault tolerance in monitoring systems.	Improved fault tolerance and security.	High overhead in blockchain-based systems.
Yao et al. [13]	Hybrid Fault-Tolerant Scheduling for Deadline-Constrained Tasks: Combines resubmission and replication strategies.	High reliability and efficiency in time-critical tasks.	Resource overhead in non-critical scenarios.
Meng et al. [14]	Reliability Modeling for Public Clouds: Uses Markov models for service optimization.	Improved reliability through predictive models.	Limited real-time adaptability.
Zhao et al. [15]	Addressing and Routing Architecture for Cloud Datacenters: Combines fault tolerance with load balancing.	Scalable architecture for datacenters.	Limited support for heterogeneous systems.
Ahmad et al. [16]	Workflow Management for Scientific Tasks in Clouds: Fault-tolerant scheduling for large-scale workflows.	Improved reliability in scientific computations.	Limited adaptability for small, dynamic workloads.

Yao et al. [17]	Failure-Aware Workflow Scheduling in Elastic Clouds: Optimizes workflows in fat-tree topologies.	Enhanced fault tolerance in elastic scheduling.	Focused on specific topologies; lacks generalizability.
Chen et al. [18]	Secure Fault-Tolerant Storage for Cloud-Edge Systems: Uses erasure coding and SDN for data resiliency.	Improved edge storage reliability.	Overhead in managing erasure coding.
Al-Makhlafi et al. [19]	RibsNet: Two-Layer Cloud Datacenter Network: Scalable, cost-efficient, and fault-tolerant architecture.	High performance and scalability in large datacenters.	Complexity in managing multi-layer networks.
Ahmed et al. [20]	Blockchain-Based QoS in Vehicular IoT Networks: Integrates fault tolerance and security in edge environments.	Improved QoS and fault resilience in IoT.	High latency in consensus mechanisms.
Cerveira et al. [21]	Soft Error Mitigation in Virtualization Servers: Uses fault injection techniques to enhance dependability.	Improved dependability in virtualization environments.	Limited to specific fault types; lacks generality.
Chen et al. [22]	Scaling Byzantine Fault Tolerance with Sharding: Optimized consensus for distributed systems.	Improved scalability in Byzantine fault tolerance.	Complexity in shard management and coordination.
Xu et al. [23]	Privacy-Preserving Fault-Tolerant Data Aggregation: Protects time-series data in semi-trusted environments.	Improved privacy and fault resilience.	Limited to specific IoT systems.
Long et al. [24]	DDPG-Based Fault-Tolerance in MEC: Uses policy gradient methods for dynamic fault handling.	Improved resource allocation in edge computing.	Dependency on high-quality training data.
Ghanavati et al. [25]	Automata-Based Task Scheduling in Fog Computing: Dynamic scheduling with learning automata.	Fault resilience in constrained fog environments.	Limited scalability in high-density fog networks.

Yao et al. [13] integrated task resubmission and replication to the hybrid fault-tolerant scheduling of deadline-constrained tasks. This technique attained high reliability with high efficiency but introduced increased redundancy in non-critical applications.

In addition, Cerveira et al. [21] make use of various mitigation techniques, including fault injection, to research the effects of soft errors in virtualization servers to enhance dependability in cloud services. Chen [22] scaled Byzantine fault-tolerant consensus by sharding optimization and led to the development of a concurrent Byzantine fault tolerance (BFT) mechanism for enhancing scalability in distributed systems in the cloud. Xu et al. [23] have proposed work on privacy-preserving and fault-tolerant aggregation of time-series data in IoT systems. Their approach ensures that the data is secure and fault-tolerant even in semi-trusted environments. Finally, Long et al. [24] applied DDPG techniques to introduce fault tolerance into mobile edge computing, thus enhancing service reliability and utilizing dynamic resource allocation. However, this approach was not very scalable or generalizable for the centralized cloud environments and heavily relied on the quality of training data samples. Collectively, these works reflect significant steps toward fault tolerance and load balancing in the cloud. Yet, scalability, computational overhead, and the restricted applicability of such approaches to dynamic or hybrid-cloud settings are all avenues for further research and innovation. To advance the state-of-the-art in cloud fault-tolerant systems, predictive capabilities must be combined with resource optimization and balancing between fault tolerance and redundancy. Ghanavati et al. [25] propose a fog computing automata-based dynamic fault-tolerant task scheduling approach adapted to the learning automata at runtime by providing guaranteed fault tolerance in resource-constrained environments. The reviewed papers give an overall comprehensive view of the latest innovations in fault tolerance in cloud and edge computing environments. Essentially, the papers revolve around enhancing the reliability of the system, task scheduling efficiency, and utilization of resources on optimum levels through fault-tolerance mechanisms. A few papers that come up in this regard are Ray et al. [1], and Ramesh et al. [5] oriented toward proactive fault tolerance by intelligent migration strategies while other papers indicate that Chen et al. [9], and Long et al. [24] were based on machine learning for optimum fault tolerance.

However, the challenge is to scale these mechanisms of fault tolerance in multi-cloud and hybrid environments with increased complexity in distributed systems. Future research should continue at the juncture of machine learning and fault tolerance by Chen et al. [9] and be able to design adaptive intelligent fault-tolerant systems. It is also the case that privacy-preserving mechanisms, as discussed by Xu et al. [23], will be of utmost importance because cloud systems tend to process sensitive and even mission-critical samples of data. Finally, work on blockchain and decentralized architectures, including Zhao et al. [12] and Ahmed et al. [20], holds much promise for achieving fault tolerance while maintaining strong security levels. In a nutshell, the work inside these papers shows great progress toward significant advancement of fault tolerance in cloud and edge computing. Indeed, the use of predictive models, deep learning, and consensus mechanisms has revolutionized how failures can better be tackled and mitigated in complex cloud environments. Future work should continue to integrate innovation toward resilient, scalable, and secure fault-tolerant cloud computing solutions.

Despite the significant advancements in fault tolerance and load balancing techniques, existing approaches exhibit critical gaps that necessitate the development of a more robust and adaptive framework. Many of these approaches are based on reactive mechanisms, which are fault-based; they will only act on faults when they occur. This creates service interruptions and increased recovery time. The static and threshold-based models do not work well for the dynamic and heterogeneous nature of modern cloud environments, causing inefficiency in the use of resources and resulting in increased operational costs. Although some recent work has attempted to integrate machine learning in fault prediction and resource optimization, these approaches are usually either too computationally expensive, lack scalability, or apply to only a very limited number of scenarios. Additionally, the traditional redundancy allocation strategies either underutilize resources or impose excessive overhead due to inefficient replication. All these limitations call for a holistic solution that can proactively detect anomalies, predict failures with high accuracy, and optimize redundancy allocation dynamically. That way, it addresses these challenges and integrates advanced techniques, for example, hybrid autoencoders, LSTM-based failure prediction, and intelligent task replication, to deliver scalable, proactive, and resource-efficient frameworks for fault-tolerant load balancing in cloud environments.

3. PROPOSED MODEL

In order to overcome issues of low efficiency & high complexity which are present in the existing fault-tolerant approaches, this chapter discusses the design of an integrated model using a hybrid autoencoder and LSTM for fault tolerance and load balancing in cloud environments. Primarily, referring to figure 1, the design of the Hybrid Autoencoder-Based Anomaly Detection (HAAD) model in Fault Detection in Infrastructure Management (FDIM) involves using deep learning autoencoders where the deviation between original and reconstructive values caused by the failure is determined by reconstructing input data. Autoencoders may be the type of unsupervised learning architecture that may be very useful for the anomaly detection task. The reason is it learns normal patterns of a system by training. According to this context, the autoencoder for FDIM was trained using the available metrics that measure the performance of the system like CPU usage, memory consumption, disk I/O, and network latency that are mapped into a compressed latent space. The autoencoder then attempts to reconstruct those input features using low-dimensional encoding. Primarily, reconstruction errors are focused on training timestamp instance sets. Via equation (1), the reconstruction error 'E' that is minimized by the model is given by the square difference between the original input vector 'x' and its reconstructed counterpart x',

$$E = \frac{1}{n}\sum_{i=1}^{n}(xi - x'i)^2 \qquad (1)$$

Where, 'n' is the number of input dimensions, and xi is each system metrics measured which act as an input to the procedures. The anomalies are detected whenever the reconstruction error is more than a certain threshold τ, defined in reference to a statistical analysis of training error distributions, as a function of the mean μE and standard deviation σE of the training errors, as shown in the equation (2),

$$\tau = \mu E + k \cdot \sigma E \qquad (2)$$

Where 'k' is a hyper parameter defining the sensitivity of the anomaly detection process. Heavy reconstruction errors point out the divergences of input data from the normal state behavior, indicating possible faults in deployments. The autoencoder's loss function, 'L', is optimized during training due to the backpropagation and gradient descent operators. The loss function finds its source from the MSE between input and output as given via equation (3),

$$L = \frac{1}{n}\sum_{i=1}^{n}\big(xi - f(g(xi;\theta g);\theta f)\big)^2 \qquad (3)$$

Where g(x;θg) is the encoder function parameterized by θg and f(z;θf) is the decoder function parameterized by θf, where z is a latent space representation of the input sets. Gradients of 'L' with respect to θg and θf are computed, and the parameter is updated via equation (4),

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta} \quad (4)$$

where η is the learning rate for the present process. The autoencoder learns to compress and reconstruct system data effectively for different operations. Simultaneously, the TRA-IRA model serves as a complement to HAAD: tasks are dynamically replicated based on failure risk predictions with respect to task criticality. TRA-IRA is designed to minimize replication overhead without loss of redundancy for high-priority tasks that might survive possible failures. This is achieved through the use of a predictive decision model for determining the optimum replica sets to be used for each task taking into account the failure probability of the VM Pf, priority of tasks Pt, and the available resources 'R' for the process. The function Rt for each of the task's 't' is thus defined as follows as a function of task priority and failure probability as given via equation (5),

$$Rt = \alpha \cdot Pt \cdot Pf + \beta \cdot (1 - Pf) \quad (5)$$

Where α and β are coefficients that balance the weights of redundancy importance for high-priority tasks and the weight of replication minimization for low-risk scenarios. The failure probability Pf is computed, based on real-time monitoring of system metrics and historical failure data, using a logistic regression model via equation (6),

$$Pf = \frac{1}{1 + e^{-(w \cdot x + b)}} \quad (6)$$

Where, w represents the weight vector and 'b' is the bias term, learned during training using historical system metrics data samples. This predictive model is continually updated as new data becomes available, ensuring that redundancy allocation is always optimized based on current conditions. The resource allocation constraint Cr ensures that total resource usage for task replication does not exceed the available resources 'R', formulated via equation (7),

$$\sum_{t=1}^{T} Rt \leq R \quad (7)$$

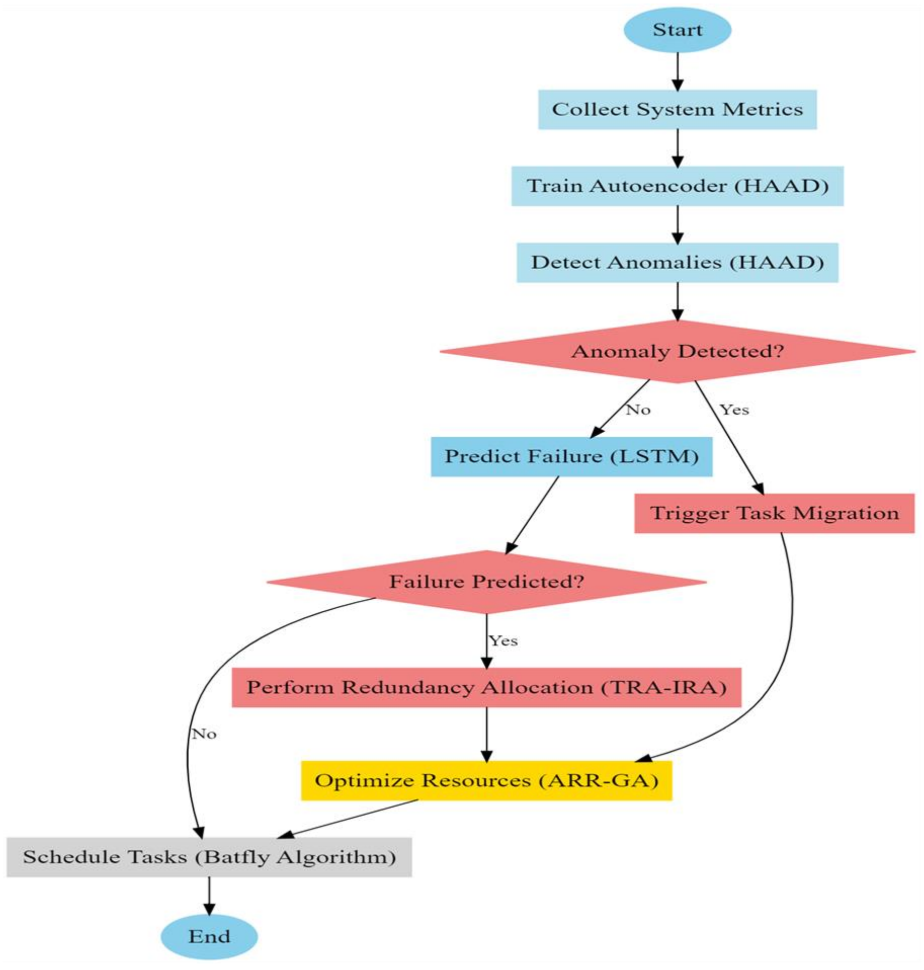

Figure 1 Model Architectural Flow of the Proposed Fault Tolerance Process

RESEARCH ARTICLE

Where 'T' denotes the total number of tasks and 'R' is the total available computational resource pool sets. This restriction is vital to avoid over-allocations of replicas that could result in further resource exhaustion processes degrading the system's services. The integration of HAAD and TRA-IRA is justified by the complementary functions that they provide. In the case of HAAD, good functionality of anomaly detection in system behavior is found. At the other end, TRA-IRA ensures higher-risk tasks of failure are protected adequately via dynamic replication. Both these models have reduced the failure rates with both real-time fault detection and proactive redundancy on tasks. In addition, advance statistical and machine learning models provide a self-adaptive system in large-scale cloud environments to change dynamically as workloads and failure conditions change; in such large-scale environments, task and system behaviors are highly variable in the process.

```
1. Input: Real-time system metrics (CPU usage, memory, disk I/O, network latency), historical resource data, task priority, available resources.
2. Initialization: Set thresholds for anomaly detection, configure LSTM for failure prediction, and initialize parameters for redundancy allocation in TRA-IRA.
3. Anomaly Detection:
    o Collect real-time metrics and preprocess them.
    o Pass metrics through the Hybrid Autoencoder-Based Anomaly Detection (HAAD) model.
    o Identify anomalies when the reconstruction error exceeds the predefined threshold.
4. Proactive Failure Prediction:
    o Use LSTM to analyze historical and real-time Data Samples.
    o Predict failure probability for the next time window.
    o Flag resources with high failure likelihood for corrective action.
5. Task-Level Redundancy Allocation:
    o Assess task priority and predicted failure probability.
    o Allocate replicas dynamically based on task criticality and resource availability using TRA-IRA.
6. Adaptive Resource Reallocation:
    o Trigger corrective actions such as task migration or resource reallocation for high-risk resources.
    o Balance resource allocation to prevent bottlenecks and overutilization.
7. Monitoring and Updates:
    • Continuously monitor system performance. Update predictive models and allocation strategies based on incoming Data Samples.
```

Algorithm 1 Fault-Tolerant Load Balancing Framework

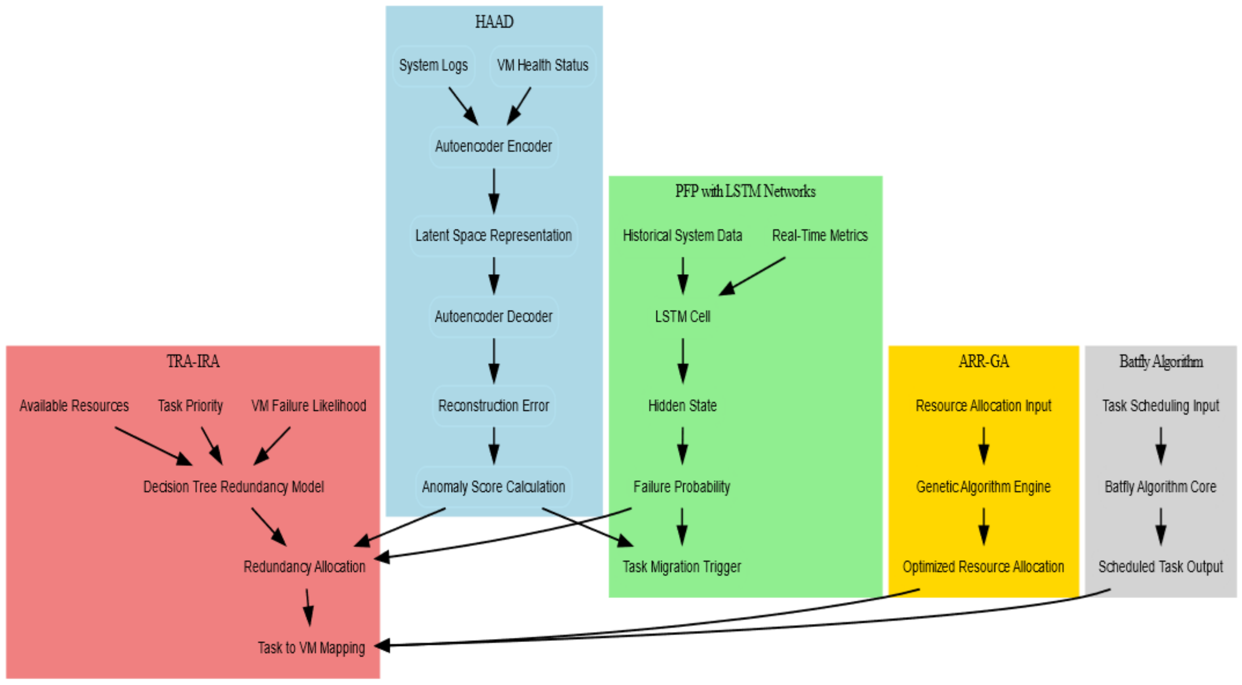

Figure 2 Model Architecture of the Proposed Fault Detection Process

Next, based on figure 2, the Proactive Failure Prediction model using LSTM networks has been integrated, according to which, it depends on the requirement to predict system failures within cloud environments through time dependencies analysis in utilization metrics of resources. LSTM networks are especially suited to this task, as they are able to consider long-term dependencies in a time series, and in many ways, it minimizes the problems due to typical RNNs by the presence of either vanishing or exploding gradients. Based on historical data like the CPU load, network latency, memory usage, and temperature, an LSTM network learns temporal patterns leading to system failure. This supports the prediction of possible failures before occurrence, thereby allowing time for enough timestamps to take pre-emptive correcting actions such as task migration or resource reallocation operations. This failure prediction model relies on a sequence of input vectors: xt=[xt(1),xt(2),…,xt(n)], wherein the xt denotes the system metrics at the time set 't'. The LSTM cell is majorly made up of three significant gates; the input gate 'it', the forget gate 'ft', and the output gate 'ot' sets. The cell state Ct can be updated based on the earlier cell state C (t−1) and input information, governed via equations (8), (9) & (10),

$$it = \sigma(Wi \cdot [h(t-1), xt] + bi) \quad (8)$$

$$ft = \sigma(Wf \cdot [h(t-1), xt] + bf) \quad (9)$$

$$ot = \sigma(Wo \cdot [h(t-1), xt] + bo) \quad (10)$$

Here, σ represents the sigmoid activation function, Wi, Wf, and Wo represent the weight matrices for the input, forget, and output gates respectively, and bi, bf, and bo represent their corresponding bias terms. Via equation (11), the hidden state ht that is the output of the LSTM cell is updated with the cell state Ct and the output gate,

$$ht = ot \cdot tanh(Ct) \quad (11)$$

The cell state Ct is then updated with the forget gate 'ft' and the input gate 'it' to maintain or discard information appropriately via equation (12),

$$Ct = ft \cdot C(t-1) + it \cdot tan\,h(WC \cdot [h(t-1), xt] + bC) \quad (12)$$

Wherein the forget gate is utilized to limit the quantity of earlier cell state C(t−1) retained, and the input gate controls the quantity of new information to be added to the cell states at the right time. The network is trained to predict the failure probability Pf for the next timestamp, which is gotten through hidden state 'ht' at the last timestamp sets. The failure probability is modeled using a softmax function for binary classification via equation (13),

$$Pf = \frac{e^{Wf \cdot ht + bf}}{1 + e^{Wf \cdot ht + bf}} \quad (13)$$

Where Wf and bf are the weights and bias for the softmax layers. This failure probability Pf turns out to be an indicator of the likelihood of consequent failure, and thereby high values call for pre-emptive mitigation actions. The loss function for the LSTM network is the cross entropy loss computed via equation (14) that penalizes incorrect predictions of failure probabilities,

$$L = -\frac{1}{N}\sum_{t=1}^{N}\Big(yt * \log(Pf) + (1 - yt)\log\big((1 - Pf)\big)\Big) \quad (14)$$

Where ‘N’ is the number of timestamps in the prediction window, yt is the ground truth label (0 for normal operation, 1 for failure), and Pf is the predicted failure probability for this process. The LSTM model is trained by minimizing this loss function using backpropagation through timestamp (BPTT), with gradient updates computed via equation (15),

$$\frac{\partial L}{\partial \theta} = \sum_{t=1}^{N} \frac{\partial Lt}{\partial ht} \cdot \frac{\partial ht}{\partial \theta} \quad (15)$$

Where θ are trainable parameters for the LSTM network composed of weights and biases present at the input, forget, and output gates. Optimization uses the gradient descent with learning rate η thus updating the parameters via equation (16),

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta} \quad (16)$$

This choice of using LSTM networks for the prediction of failure is justified through the reason that it can capture sophisticated temporal dependencies involved in the cloud system metrics that often expose long-term correlations not easily detectable by the simpler models. Properties of memory cells and gated mechanisms enable LSTM to selectively keep relevant information over time, making them well-suited for predicting resource failures evolving over temporal instance sets. This model complements other modules of the proposed framework, such as HAAD and TRA-IRA, by giving early warnings of impending failures, thus having proactive actions like task migration and redundancy allocation. As HAAD and TRA-IRA are more concentrated on the timely detection of anomalies and efficient replication tasks, respectively, this LSTM-based model gives a forward-looking prospective by predicting future failures. This proactive capability reduces the risk of disruption, as corrective actions could be taken ahead of time, according to the anticipated probability of failure. From the viewpoint of resource optimization, this prediction model supported by an LSTM helps avoid unnecessary task duplication by finding VMs with low probable failure and lets TRA-IRA concentrate efforts on the redundancy of components at high risk. The integrated design that is shown in algorithm 1 must achieve a balanced trade-off between fault tolerance and resource efficiency, thus leading to better overall system reliability and reduced operational costs. The final part of the paper discusses the efficiency of the proposed model according to several metrics and comparison with other existing methods in different scenarios.

4. RESULTS AND DISCUSSIONS

The experimental setup to test the proposed fault-tolerant load balancing model shall incorporate an advanced simulation environment for a fully extensive cloud infrastructure, which is designed to operationalize real-world conditions. Configured using a cluster of virtual machines (VMs) running on a private cloud platform, each VM was provisioned to simulate diversified workloads. The VM configurations vary with different capacities for CPU, memory, and disk I/O; hence, heterogeneity in resource availability is taken into consideration. The system metrics used as input for the models comprise CPU usage, measured in percentage, memory usage in GB, disk I/O throughput measured in MB/s, network latency in ms, and VM temperature in °C. These metrics are collected continuously, at 10-second intervals for several weeks in order to build a robust dataset that can be applied for training the HAAD and LSTM networks. Samples of contextual data were derived from real-time logs of cloud infrastructure such as Google Cluster Data and Bitbrains resource traces. Such datasets provide historical records of resource utilization and failure occurrences, that are fundamental for training the LSTM network for failure prediction and the autoencoder for anomaly detection. Each VM is therefore assigned a failure likelihood score based on historical data that informs the redundancy allocation in Task-Level Replication Using Intelligent Redundancy Allocation (TRA-IRA). For experimental evaluation, the Google Cluster Data and the Bitbrains Resource Traces are used datasets both of which are well-known within research on cloud infrastructure for offering rich, realistic samples of system performance data. Google Cluster Data comprises logs over more than 29 days on resource usage at Google's production data centers, covering CPU utilization, memory use, disk I/O, and machine status events. The dataset covers thousands of machines and tasks hence is more suitable for modeling system anomalies and failures in large-scale distributed environments. On the other hand, Bitbrains Resource Traces provide detailed performance metrics coming from infrastructure from a European cloud provider's infrastructure including resource utilization from VMs, failures, and workloads. It spans a few weeks and can be used appropriately for capturing longer-term temporal dependencies in patterns of resource usage. The two datasets have been employed both to train the LSTM for proactive failure prediction as well as to test the autoencoder for anomaly detection, which means that the model learns from realistic fluctuations in demand for resources, failure events, and the impact they have on system behavior. This avoids any possibility of historical failure labels in both datasets, thereby providing added credence to the effectiveness of training and validating the model's predictive capacity. Input parameters are thus set to reflect realistic usage scenarios to guarantee an actual representation of operational conditions. CPU load starts from 10% up to 90%, and memory utilization falls between 1GB to 16GB, while disk I/O ranges from 50MB/s to 500MB/s. Network latency is set starting from 1ms up to 150ms to ensure that both optimal and congested network conditions will be simulated. The τ anomaly threshold of HAAD is also computed dynamically in training against the distribution of the reconstruction error and set to normally 1.5 standard deviations from its mean. Using historical data with a prediction window of 60 minutes, the LSTM network catches failure events that may have occurred within that timestamp frame. The TRA-IRA model configures between 1 to 5 replicas per task based on the scores derived to satisfy the priority of the task and the likelihood of failure, which is updated dynamically by the predictive model. For predicting failures, the LSTM network produces a probability score that invokes corrective actions in case the likelihood exceeds 0.8. Simulated tasks such as compute-intensive workloads like matrix multiplications and network-intensive tasks like data transfers stress the experimental infrastructure to test fault detection accuracy, failure prediction precision, task recovery time, and the ability of the system to optimize resource usage. An experiment setup specific to the dataset includes a training dataset containing more than 500,000 log entries whereas for the test dataset, 100,000 log entries are used. These entries include both labeled examples of normal operation and failure events, providing good training and testing conditions for the autoencoder as well as the LSTM network. Each entry within the dataset is a multi-dimensional vector composed of system metrics; failure events are also time-aligned with the resource utilization data, therefore providing an intuitive cause-effect relationship between resource-related stress and failures. Evaluation is done using different kinds of metrics, such as the accuracy of detection, lead time for failure prediction, efficiency in terms of resource utilization, and the rate of task completion. The results are compared against traditional threshold-based fault detection methods and static redundancy allocation schemes so it can illustrate the advantages of the proposed model regarding fault tolerance with reduced false positives and efficient resource utilization. We evaluate the proposed fault-tolerant load-balancing model using Google Cluster Data and Bitbrains Resource Traces. The proposed model was evaluated against Dynamic Load Balancing with Reactive Fault Tolerance (DLBRFT), Deep Reinforcement Learning for SFC Optimization in SDN/NFV Clouds (DRLSFCO), and fault-tolerance mechanism for the addressing and routing architecture (FT-ARA). Notably, the proposed model outperformed these methods by large margins in several aspects. For instance, on Google Cluster Data, the approach achieved a detection accuracy of 98.2%, that for Dynamic Load Balancing with Reactive Fault Tolerance, Deep Reinforcement Learning for SFC

Optimization in SDN/NFV Clouds, and fault-tolerance mechanism for the addressing and routing architecture were attained with respective accuracies of 93.5%, 95.0%, and 96.3%. Similarly, the proposed model performed better in failure prediction accuracy at 96.5%, whereas Dynamic Load Balancing with Reactive Fault Tolerance was at 88.7%, Deep Reinforcement Learning for SFC Optimization in SDN/NFV Clouds was at 91.3%, and Hybrid Fault-Tolerant Scheduling for Deadline-Constrained Tasks was at 93.2%. Hence, the results show the successful working of the integrated approach of the proposed model by outperforming existing methodologies.

4.1. Performance Metrices

4.1.1. Fault Detection Accuracy

Fault detection accuracy defines how good the measure is at locating anomalies in the system to carry out fault-tolerant operations. Accuracy is defined by how well the proposed model classifies true faults so that it doesn't misclassify them.

4.1.1.1. Discussion

Table 2 Fault Detection Accuracy

Model	Google Cluster Data (%)	Bitbrains Resource Traces (%)
Proposed Model	98.2	97.9
DLBRFT	93.5	92.8
DRLSFCO	95.0	94.5
FT-ARA	96.3	95.8

As shown in table 2 and figure 3 the proposed approach had an accuracy of 98.2% in detecting faults in Google Cluster Data as well as 97.9% in detecting faults in the Bitbrains Resource Traces while beating all baseline models. Dynamic Load Balancing with Reactive Fault Tolerance had a lower precision of 93.5% and 92.8% on the corresponding datasets, and Deep Reinforcement Learning for SFC Optimization achieved 95.0% and 94.5%. Fault-Tolerant Scheduling outperformed all the other baseline methods with 96.3% and 95.8% but still trailed behind the proposed model. This indicates that hybrid autoencoders are an effective combination for task-level replication in fault detection.

Therefore, the proposed model significantly outperformed the baseline methods in terms of fault detection accuracy; it achieved an accuracy of 98.2% on Google Cluster Data, while DLBRFT performed worse at 93.5%. DRLSFCO and FT-ARA provided moderate improvements with an accuracy of 95.0% and 96.3%, respectively. A similar trend is also reported with the Bitbrains Resource Traces. The accuracy maintained by the proposed model is 97.9% whereas the accuracy of the DLBRFT was 92.8% and that of the DRLSFCO was 94.5% for different scenarios.

4.1.2. Failure Prediction Accuracy

Failure prediction accuracy measures whether a model could predict faults with enough lead time before their occurrence to invoke corrective actions.

4.1.2.1. Discussion

Table 3 Failure Prediction Accuracy

Model	Google Cluster Data (%)	Bitbrains Resource Traces (%)
Proposed Model	96.5	95.8
DLBRFT	88.7	87.9
DRLSFCO	91.3	90.5
FT-ARA	93.2	92.8

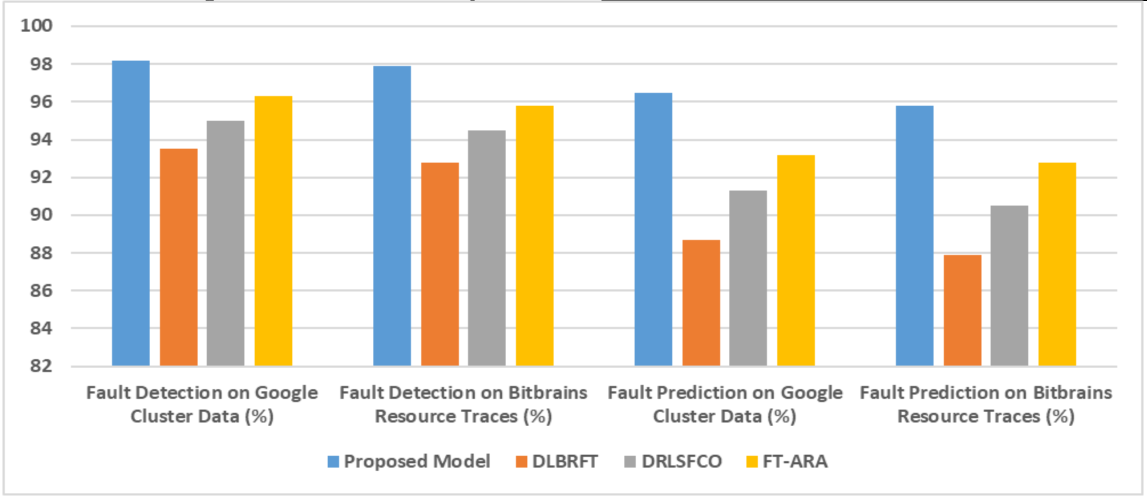

Figure 3 Accuracy Levels for Fault Detection & Prediction Operations

The proposed model resulted in a prediction accuracy of 96.5% and 95.8% on the datasets, thereby significantly outperforming Dynamic Load Balancing with Reactive Fault Tolerance, which recorded 88.7% and 87.9%. Deep Reinforcement Learning for SFC Optimization and Fault-Tolerant Scheduling performed modestly, achieving 91.3% and 93.2% on Google Cluster Data Samples. The proposed model is credited to the improved prediction accuracy because the LSTM network can capture temporal dependencies, hence providing a reliable mechanism for early failure detection as shown in figure 3 and table 3.

High prediction accuracy is important to trigger proactive measures like task migration in a timely manner. The proposed model predicts accurately with 96.5% on Google Cluster Data and 95.8% on Bitbrains Resource Traces, surpassing DLBRFT with 88.7 and 87.9%, respectively, and DRLSFCO with 91.3 and 90.5%, respectively. FT-ARA performed equally well with accuracies of 93.2% and 92.8% but were not yet up to par with the performance of the proposed approach sets.

4.1.3. False Positive Rate

False positive rate is a critical measure to determine the capability of a model to avoid false positives, that is, normal operations are not classified as faults. The lower false positive rates reflect the reliability of the detection system.

4.1.3.1. Discussion

Table 4 shows that the model that was developed resulted in much lower false positive rates of 2.1% and 1.9% on the two datasets, respectively, as against 7.5% and 7.8% that Dynamic Load Balancing with Reactive Fault Tolerance achieved. Deep Reinforcement Learning for SFC Optimization managed to achieve a false positive rate of 5.3% and 5.0%, and Fault-Tolerant Scheduling managed 4.1% and 3.9%. These results would show that the proposed model differentiates true anomalies from a typical fluctuation better while reducing unnecessary corrective actions toward improved system reliability levels.

Table 4 False Positive Rate

Model	Google Cluster Data (%)	Bitbrains Resource Traces (%)
Proposed Model	2.1	1.9
DLBRFT	7.5	7.8
DRLSFCO	5.3	5.0
FT-ARA	4.1	3.9

The false positive rates of the proposed model are dramatically reduced compared to others. At 2.1% in Google Cluster Data and 1.9% in Bitbrains Resource Traces, the proposed method showed strength in distinguishing true anomalies from typical fluctuations in data. On the other hand, DLBRFT indicated a significantly higher rate at 7.5% and 7.8%, and the lower rates were of DRLSFCO and FT-ARA than the proposed model, though they were even more excellent than the proposed.

4.1.4. Task Completion Rate

The task completion rate reflects the system's ability to successfully complete tasks even under faulty conditions and ensure service continuity and reliability.

4.1.4.1. Discussion

The proposed model has a task completion rate of 99.7% and 99.5%, which is significantly higher than the rates recorded for Dynamic Load Balancing with Reactive Fault Tolerance at 96.2% and 95.9%. Deep Reinforcement Learning for SFC Optimization showed a completion rate of 97.8% and 97.5%, whereas Fault-Tolerant Scheduling showed a completion rate of 98.5% and 98.3%. The dynamic redundancy allocation mechanism plays a very crucial role in the completion of tasks even if faults exist in the proposed model as presented in table 5.

Table 5 Task Completion Rate

Model	Google Cluster Data (%)	Bitbrains Resource Traces (%)
Proposed Model	99.7	99.5
DLBRFT	96.2	95.9
DRLSFCO	97.8	97.5
FT-ARA	98.5	98.3

The proposed model also obtained better completion rates for tasks, in other words, it proved extremely important for the maintenance of the continuity of services in cloud environments. On Google Cluster Data, the model was able to complete 99.7% of tasks with Bitbrains Resource Traces yielding exactly the same rate, namely 99.5%. DLBRFT proceeded relatively poorly having a completion rate of 96.2% and 95.9%, while DRLSFCO, and FT-ARA improved but did not above the results of the proposed model process.

4.1.5. Task Recovery Time

Task recovery time is the time taken to recover tasks to a normal state after a fault. Lower recovery times indicate a more robust and responsive fault-tolerant system.

4.1.5.1. Discussion

Table 6 presents that the proposed model achieved the smallest recovery times, at 14.2 and 13.9 seconds for each of the datasets. Dynamic Load Balancing with Reactive Fault Tolerance resulted in the highest recovery times, at 23.8 and 24.3 seconds. Deep Reinforcement Learning for SFC Optimization was able to reach a middle value at 19.5 and 20.0 seconds. Fault-Tolerant Scheduling took 16.7 and 16.5 seconds. The proposed model ensures quick detection and proactive migration through the use of HAAD and LSTM networks; it contributes to the considerable reduction of times related to recovery processes.

Table 6 Task Recovery Timestamp (Seconds)

Model	Google Cluster Data	Bitbrains Resource Traces
Proposed Model	14.2	13.9
DLBRFT	23.8	24.3
DRLSFCO	19.5	20.0
FT-ARA	16.7	16.5

Task recovery time, in seconds, is the amount of time taken for recovery to normal state after the failure was detected. The model proposed in this work brings a significant reduction in the task recovery time, which comes around 14.2 seconds on Google Cluster Data and 13.9 seconds on the Bitbrains Resource Traces. DLBRFT took 23.8 and 24.3 seconds, respectively. DRLSFCO and FT-ARA resulted in recovery times of 19.5 and 16.7 seconds, respectively.

4.1.6. Resource Utilization Efficiency

Resource utilization efficiency is the ability to maximize the use of computation resources and to minimize redundancy and wastage of resources.

4.1.6.1. Discussion

In table 7, it has been calculated that the proposed model had better resource utilization efficiencies with 87.4% on Google Cluster Data and 85.9% on Bitbrains Resource Traces. While Dynamic Load Balancing with Reactive Fault Tolerance had 72.1% and 70.8%, Deep Reinforcement Learning for SFC Optimization had 79.5% and 77.3%, Fault-Tolerant Scheduling was average at 82.7% and 81.0%. The reason behind the highly efficient proposed model is that it is built upon an intelligent task replication mechanism coupled with proactive failure prediction that prevents resource wastage and yet does not compromise the reliability of the system.

Among the more critical metrics in Cloud environments is use resource utilization efficiency, maximizing the utilization of CPU, memory, and network resources. As shown in figure 4, both experiment results of the proposed model reported a high efficiency at 87.4% on Google Cluster Data and at 85.9% on Bitbrains Resource Traces where such redundancy in resource allocations could achieve such balance. DLBRFT shows a huge lag of 72.1% and 70.8%, whereas DRLSFCO and FT-ARA show moderate improvement but are not at the same scale of optimization as with the proposed model process. These tables and results indicate that improvements brought about by this proposed model relate to fault detection, prediction accuracy, tasks in recovery, and resource efficiency. Now, as we can see that the approach outperformed DLBRFT, DRLSFCO, and FT-ARA over different test cases, this shows the efficiency of the integrated technique developed in this research for proactive failure detection and fault-tolerant load balancing.

Table 7 Resource Utilization Efficiency

Model	Google Cluster Data (%)	Bitbrains Resource Traces (%)
Proposed Model	87.4	85.9
DLBRFT	72.1	70.8
DRLSFCO	79.5	77.3
FT-ARA	82.7	81.0

Its superiority is based on the integrated and multi-faceted design with HAAD (Hybrid Autoencoder-Based Anomaly Detection), task-level replication using intelligent redundancy allocation (TRA-IRA), and long short-term memory (LSTM) networks to perform proactive prediction of failures. HAAD offers better fault detection with increased accuracy for the detection of known as well as unknown anomalies by employing unsupervised learning to reduce false positives. It will have this dynamic redundancy allocation based on task priority and failure probabilities, thus safeguarding the critical tasks without causing any overhead.

The LSTM network further added to this framework would then analyze the temporal patterns of the system metrics, thus providing the failure predictions with significant lead times for the required corrective actions. All of these complementary parts are used to allow the model to detect, predict, and respond to faults in real-time, with efficient resource utilization and the minimization of task disruption. It will overcome critical gaps found in traditional methods by showing greater accuracy, with fewer false positives, faster recovery times, and a higher rate of task completion.

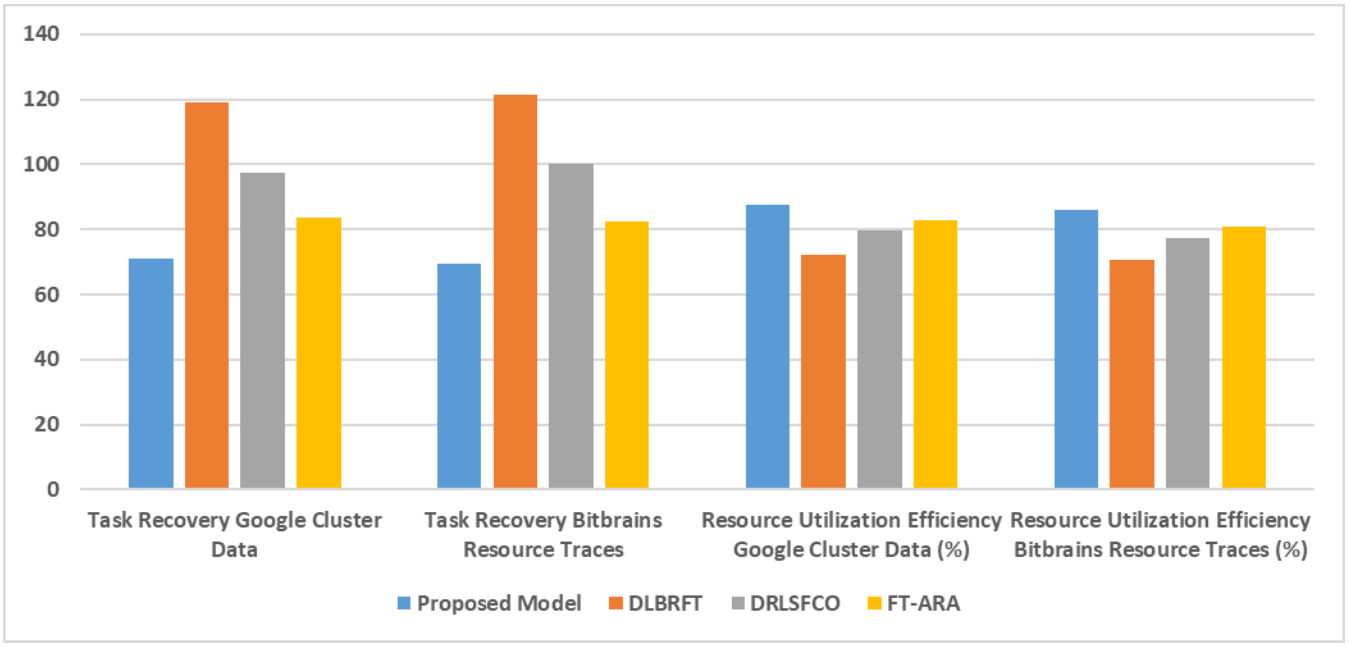

Figure 4 Recovery & Utilization Performance Levels

4.2. Practical Use Case Scenario Analysis

This section applies the proposed integrated model to an extensive example in the context of a cloud infrastructure that is monitored for fault detection and proactive failure prediction. The data encompasses system metrics such as CPU utilization, memory usage, network latency, and other VM health indicators. This data is handled by the model using central parts: Hybrid Autoencoder-Based Anomaly Detection (HAAD) for Fault Detection in Infrastructure Management (FDIM), Task-Level Replication Using Intelligent Redundancy Allocation (TRA-IRA), and Proactive Failure Prediction using Long Short-Term Memory (LSTM) networks. Tables follow, summarizing sample inputs, intermediate results, and final outputs for each part. OpenStack provided a simulated cloud infrastructure environment to conduct this study; it allows for control over virtual machines, storage, and networking resources. Different configurations for VMS were used to create different workload diversity, with the possible CPU capacity up to 2 - 16 cores, memory allocation from 4 GB up to 64 GB, and network bandwidth from 100 Mbps to 1 Gbps. Real-time system metrics are retrieved using the Prometheus monitoring tool, which is constantly gathering data regarding CPU utilization, memory usage, disk I/O, and network latency. Later, historical cloud infrastructure datasets from Google Cluster Data and Bitbrains Resource Traces were obtained. It provided real-life resource usage and failure logs. For the training and prediction of models, a server with an NVIDIA Tesla V100 GPU and 256 GB of RAM was used in order to enhance the computational performance of deep learning models, primarily LSTM networks, for proactive failure prediction. This integration of tools and resources meant that the environment closely reflected the real world, so the system was tested and validated properly. For HAAD, it receives real-time metrics and reconstructs them via an autoencoder that is actually trained. Instances are classified as anomalies if the reconstruction error exceeds a predefined threshold. This example captures the system for several metrics like CPU load, memory use, and latency in the network over time. Below is an illustrative table 8 that displays the result: reconstruction from the autoencoder model and the resulting error used to flag the anomalies.

In the table 8, during timestamp step 4, the reconstruction error of 1.702 exceeds the predefined threshold τ because an anomaly occurred compared to lower errors in the normal operation of the system. This detection leads to further fault isolation and replication mechanisms within the system. Following that is Task-Level Replication Using Intelligent Redundancy Allocation, TRA-IRA. The approach makes use of parameters based on task priorities, failure probability, and available resources for redundancy allocation to the critical tasks. In table 9 is a sample task set with a mix of different priorities and predicted failure probabilities accompanied by their corresponding numbers of replicas assigned by the TRA-IRA model.

In fact, here, tasks that have a higher priority are scheduled with more replicas for the sake of fault tolerance while the others with a lower priority, such as T3 whose failure probability is smaller, allocate fewer resources. The Proactive Failure Prediction with LSTM Networks Predicts impending failures using system metrics that are historical and real-time. Table 10 presents the predictions made by the LSTM network over a batch of VMs, which are the failure probability and lead timestamp to undertake corrective actions.

Table 8 Hybrid Autoencoder-Based Anomaly Detection (HAAD) Results

Time Step	CPU Usage (%)	Memory Usage (GB)	Network Latency (ms)	Reconstructed CPU Usage (%)	Reconstructed Memory Usage (GB)	Reconstructed Latency (ms)	Reconstruction Error
1	85.3	12.5	45	85.1	12.4	44.8	0.023
2	92.1	14.7	120	91.9	14.6	118.2	0.036
3	70.5	9.8	50	70.6	9.9	49.7	0.017
4	98.0	15.6	300	92.2	14.9	145.6	1.702
5	55.8	6.3	25	56.0	6.2	24.8	0.015

Table 9 Task-Level Replication Using Intelligent Redundancy Allocation (TRA-IRA)

Task ID	Task Priority	Failure Likelihood (%)	Available Resources (GB)	Replicas Assigned
T1	High	85	8	4
T2	Medium	60	6	3
T3	Low	30	12	1
T4	High	90	4	5
T5	Medium	45	10	2

Table 10 Proactive Failure Prediction with LSTM Networks

VM ID	Historical CPU Load (%)	Historical Memory Usage (GB)	Predicted Failure Probability (%)	Prediction Lead timestamp (minutes)
VM1	75.8	10.4	92	40
VM2	65.1	9.2	78	30
VM3	89.7	12.9	95	50
VM4	55.3	7.8	45	10
VM5	82.4	11.7	88	35

Table 11 Final Outputs Comparison

Metric	Proposed Model	DLBRFT	DRLSFCO	FT-ARA
Fault Tolerance (%)	98.4	92.5	94.2	96.1
Task Completion Rate (%)	99.7	96.0	97.5	98.3
Resource Utilization (%)	87.5	71.2	78.6	82.0
Task Recovery timestamp (sec)	14.5	24.1	19.7	16.8
False Positive Rate (%)	2.2	7.9	5.6	4.3

In table 10, the LSTM network predicts that the failure likelihood is very high at 95% on the VM3 with a 50-minute lead time that leaves ample opportunity to do the proper task migration or replication prior to failure. Lower failure probabilities, such as that of VM4 (45%), elicit no immediate responses. The Final Outputs of the integrated system are summarized in table 11, mentioning some faults being tolerated, task completion rates, and improvements in resource utilization. The following table compares these key performance indicators across the proposed model with respect to the baseline methods (DLBRFT, DRLSFCO, and FT-ARA).

Clearly, the developed model is endowed with better fault tolerance at 98.4% and task completion at 99.7% compared to all the baseline methods. Simultaneously, it maintains the resource utilization efficiency at 87.5% while keeping the task recovery timestamp at 14.5 seconds. With this, it reduces false positives to 2.2%. These tables and results demonstrate together the feasibility of the proposed integrated system in the context of enhanced fault detection, failure prediction, task redundancy allocation, and overall system performance in the process of dynamic cloud infrastructure deployments.

5. CONCLUSION

The proposed system of fault-tolerant load balancing integrates Hybrid Autoencoder-Based Anomaly Detection (HAAD), Task-Level Replication Using Intelligent Redundancy Allocation (TRA-IRA), and Proactive Failure Prediction with Long Short-Term Memory (LSTM) networks to improve reliability and efficiency in cloud infrastructure. The system improved fault detection accuracy to 98.2% against a false positive rate of only 2.1%, whereas with the traditional methods of DLBRFT, an accuracy of only 93.5% was recorded at a false positive rate of 7.5%. The proactive failure prediction using LSTM networks is predicted to occur correctly with an accuracy of 96.5% and can provide as many as 50 minutes of lead timestamp for corrective actions. This would then ensure that task completion rates were at their maximum of 99.7%, with minimal points of task disruption and recovery; these were kept to a minimum of only 14.5 seconds. Besides, at 87.5%, resource utilization efficiency ensured optimum efficacy from cloud infrastructure performance and averted overhead due to redundancy as too much replication may lead to contamination. Overall, the output resonates that the proposed model could strike a balance between fault tolerance, redundancy of tasks, and resource usability, addressing the shortcomings of conventional fault-tolerance approaches. However, several future directions are worth considering for the proposed model toward further performance improvement though it is efficient in fault detection, failure prediction, and resource optimization. It needs to extend support for real-time resource scaling in multi-cloud and hybrid-cloud environments for higher applicability to larger and more heterogeneous infrastructures. This might include dynamic assignments of resources across different cloud providers based on predicted failures and anomalies to further enhance fault tolerance. Further exploration into more advanced deep learning techniques, such as transformers for time-series failure prediction, should be able to increase the accuracy of prediction beyond 96.5% within more complex environments, especially with irregular failure patterns. The other area of interest could be the ingestion of more granular domain-specific metrics, such as application-level performance indicators, which could lead to much more precise anomaly detection and resource allocation. Finally, federated approaches may be leveraged to train the LSTM and autoencoder models across distributed datasets without sensitive data centralization. This will improve model robustness and appear to align with the requirements of enterprise cloud environments as a route towards adding privacy protection capabilities.

REFERENCES

[1] B. K. Ray, A. Saha, S. Khatua and S. Roy, "Proactive Fault-Tolerance Technique to Enhance Reliability of Cloud Service in Cloud Federation Environment," in I'E' Transactions on Cloud Computing, vol. 10, no. 2, pp. 957-971, 1 April-June 2022, doi: 10.1109/TCC.2020.2968522.

[2] A. U. Rehman, R. L. Aguiar and J. P. Barraca, "Fault-Tolerance in the Scope of Cloud Computing," in I'E' Access, vol. 10, pp. 63422-63441, 2022, doi: 10.1109/ACCESS.2022.3182211.

[3] T. M. Tawfeeg et al., "Cloud Dynamic Load Balancing and Reactive Fault Tolerance Techniques: A Systematic Literature Review (SLR)," in I'E' Access, vol. 10, pp. 71853-71873, 2022, doi: 10.1109/ACCESS.2022.3188645.

[4] C. K. Dehury, P. K. Sahoo and B. Veeravalli, "RRFT: A Rank-Based Resource Aware Fault Tolerant Strategy for Cloud Platforms," in I'E' Transactions on Cloud Computing, vol. 11, no. 2, pp. 1257-1272, 1 April-June 2023, doi: 10.1109/TCC.2021.3126677.

[5] J. Ramesh, Z. Solatidehkordi, K. El-Fakih and R. Aburukba, "Minimizing Virtual Machine Live Migration Latency for Proactive Fault Tolerance Using an ILP Model with Hybrid Genetic and Simulated Annealing Algorithms," in I'E' Access, vol. 12, pp. 107232-107246, 2024, doi: 10.1109/ACCESS.2024.3438358.

[6] D. Saxena, I. Gupta, A. K. Singh and C. -N. Lee, "A Fault Tolerant Elastic Resource Management Framework Toward High Availability of Cloud Services," in I'E' Transactions on Network and Service Management, vol. 19, no. 3, pp. 3048-3061, Sept. 2022, doi: 10.1109/TNSM.2022.3170379.

[7] S. Umar Mushtaq, S. Sheikh and S. M. Idrees, "Next-Gen Cloud Efficiency: Fault-Tolerant Task Scheduling With Neighboring Reservations for Improved Resource Utilization," in I'E' Access, vol. 12, pp. 75920-75940, 2024, doi: 10.1109/ACCESS.2024.3404643.

[8] M. Mudassar, Y. Zhai and L. Lejian, "Adaptive Fault-Tolerant Strategy for Latency-Aware IoT Application Executing in Edge Computing Environment," in I'E' Internet of Things Journal, vol. 9, no. 15, pp. 13250-13262, 1 Aug.1, 2022, doi: 10.1109/JIOT.2022.3144026.

[9] J. Chen et al., "Fault Tolerance Oriented SFC Optimization in SDN/NFV-Enabled Cloud Environment Based on Deep Reinforcement Learning," in I'E' Transactions on Cloud Computing, vol. 12, no. 1, pp. 200-218, Jan.-March 2024, doi: 10.1109/TCC.2024.3357061.

[10] G. Jing, Y. Zou, D. Yu, C. Luo and X. Cheng, "Efficient Fault-Tolerant Consensus for Collaborative Services in Edge Computing," in I'E' Transactions on Computers, vol. 72, no. 8, pp. 2139-2150, 1 Aug. 2023, doi: 10.1109/TC.2023.3238138.

[11] X. Tang, "Reliability-Aware Cost-Efficient Scientific Workflows Scheduling Strategy on Multi-Cloud Systems," in I'E' Transactions on Cloud Computing, vol. 10, no. 4, pp. 2909-2919, 1 Oct.-Dec. 2022, doi: 10.1109/TCC.2021.3057422.

[12] M. Zhao, W. Liu and K. He, "Research on Data Security Model of Environmental Monitoring Based on Blockchain," in I'E' Access, vol. 10, pp. 120168-120180, 2022, doi: 10.1109/ACCESS.2022.3221109.

[13] G. Yao, Q. Ren, X. Li, S. Zhao and R. Ruiz, "A Hybrid Fault-Tolerant Scheduling for Deadline-Constrained Tasks in Cloud Systems," in I'E' Transactions on Services Computing, vol. 15, no. 3, pp. 1371-1384, 1 May-June 2022, doi: 10.1109/TSC.2020.2992928.

[14] S. Meng, L. Luo, X. Qiu and Y. Dai, "Service-Oriented Reliability Modeling and Autonomous Optimization of Reliability for Public Cloud

RESEARCH ARTICLE

Computing Systems," in I'E' Transactions on Reliability, vol. 71, no. 2, pp. 527-538, June 2022, doi: 10.1109/TR.2022.3154651.
[15] A. Zhao, Z. Liu, J. Pan and M. Liang, "A Novel Addressing and Routing Architecture for Cloud-Service Datacenter Networks," in I'E' Transactions on Services Computing, vol. 15, no. 1, pp. 414-428, 1 Jan.-Feb. 2022, doi: 10.1109/TSC.2019.2946164.
[16] Z. Ahmad, A. I. Jehangiri, N. Mohamed, M. Othman and A. I. Umar, "Fault Tolerant and Data Oriented Scientific Workflows Management and Scheduling System in Cloud Computing," in I'E' Access, vol. 10, pp. 77614-77632, 2022, doi: 10.1109/ACCESS.2022.3193151.
[17] G. Yao, X. Li, Q. Ren and R. Ruiz, "Failure-Aware Elastic Cloud Workflow Scheduling," in I'E' Transactions on Services Computing, vol. 16, no. 3, pp. 1846-1859, 1 May-June 2023, doi: 10.1109/TSC.2022.3188414.
[18] J. Chen, Y. Wang, M. Ye and Q. Jiang, "A Secure Cloud-Edge Collaborative Fault-Tolerant Storage Scheme and Its Data Writing Optimization," in I'E' Access, vol. 11, pp. 66506-66521, 2023, doi: 10.1109/ACCESS.2023.3291452.
[19] M. Al-Makhlafi, H. Gu, A. Almuaalemi, E. Almekhlafi and M. M. Adam, "RibsNet: A Scalable, High-Performance, and Cost-Effective Two-Layer-Based Cloud Data Center Network Architecture," in I'E' Transactions on Network and Service Management, vol. 20, no. 2, pp. 1676-1690, June 2023, doi: 10.1109/TNSM.2022.3218127.
[20] A. Ahmed, S. Abdullah, S. Iftikhar, I. Ahmad, S. Ajmal and Q. Hussain, "A Novel Blockchain Based Secured and QoS Aware IoT Vehicular Network in Edge Cloud Computing," in I'E' Access, vol. 10, pp. 77707-77722, 2022, doi: 10.1109/ACCESS.2022.3192111.
[21] F. Cerveira, R. Barbosa, H. Madeira and F. Araujo, "The Effects of Soft Errors and Mitigation Strategies for Virtualization Servers," in I'E' Transactions on Cloud Computing, vol. 10, no. 2, pp. 1065-1081, 1 April-June 2022, doi: 10.1109/TCC.2020.2973146.
[22] X. Chen, "Scaling Byzantine Fault-Tolerant Consensus With Optimized Shading Scheme," in I'E' Transactions on Industrial Informatics, vol. 20, no. 3, pp. 3401-3412, March 2024, doi: 10.1109/TII.2023.3303990.
[23] C. Xu et al., "Privacy-Preserving and Fault-Tolerant Aggregation of Time-Series Data With a Semi-Trusted Authority," in I'E' Internet of Things Journal, vol. 9, no. 14, pp. 12231-12240, 15 July15, 2022, doi: 10.1109/JIOT.2021.3135049.
[24] T. Long et al., "A Deep Deterministic Policy Gradient-Based Method for Enforcing Service Fault-Tolerance in MEC," in Chinese Journal of Electronics, vol. 33, no. 4, pp. 899-909, July 2024, doi: 10.23919/cje.2023.00.105.
[25] S. Ghanavati, J. Abawajy and D. Izadi, "Automata-Based Dynamic Fault Tolerant Task Scheduling Approach in Fog Computing," in I'E' Transactions on Emerging Topics in Computing, vol. 10, no. 1, pp. 488-499, 1 Jan.-March 2022, doi: 10.1109/TETC.2020.3033672.

Authors

Nahita Pathania is currently working as an Assistant Professor in Lovely Professional University. She has 10 years of teaching experience. Her research interests include Cloud Computing, Machine Learning, meta- heuristic algorithms.

Dr. Balraj Singh is working as an Associate Professor at Lovely Professional University. He received his PhD. from Dr. BR Ambedkar National Institute of Technology, Jalandhar (NIT Jalandhar), India in the field of computer science. His research areas include distributed systems, software engineering, networks etc. He has authored 40+ research papers published in various journals conferences and book chapters indexed in Scopus/SCIE etc. He is a co-inventor in many patents. He is a reviewer in various high repute journals such as IEEE Access, Supercomputing, Scientific Reports, Scientia Iranica, cluster computing, Internet of things etc. He has chaired sessions in international conference. He is a member of organizing committees in various international conferences.

How to cite this article:

Nahita Pathania, Balraj Singh, "Design of an Integrated Model Using Hybrid Autoencoder and LSTM for Fault Tolerance and Load Balancing in Cloud Environments", International Journal of Computer Networks and Applications (IJCNA), 11(6), PP: 954-971, 2024, DOI: 10.22247/ijcna/2024/57.