Journal of Neonatal Surgery
ISSN(Online): 2226-0439
Vol. 14, Issue 4s (2025)
https://www.jneonatalsurg.com

OPEN ACCESS

Defending the Overloading System in Cloud Computing using Preamble Fuzzy Estimator based RNN

Ananya Saha[1], Dr. S. K. Manju Bargavi[2]

[1]Research Scholar (Jain (Deemed-to-be) University), Bengaluru, India.

[2]Professor, (Jain (Deemed-to-be) University), Bengaluru, India.

Cite this paper as: Ananya Saha, Dr. S. K. Manju Bargavi, (2025) Defending the Overloading System in Cloud Computing using Preamble Fuzzy Estimator based RNN. *Journal of Neonatal Surgery*, 14 (4s), 945-961.

ABSTRACT

Cloud server overloading poses a significant challenge in modern computing environments where scalability and reliability are paramount. Cloud server overloading occurs when the demand for resources exceeds the server's capacity, resulting in performance degradation or even system failures. Preventive measures such as Load balancing, Auto-scaling, Resource Monitoring, Predictive Analytics, Content delivery networks, Caching mechanisms, and Distributed Databases are used to manage cloud server overloading. This work is indented to introduce a novel preventive methodology using Fuzzy logic and Recurrent Neural Network (RNN) to diminish server overloading in cloud environments. Preamble Fuzzy Load Estimator and RNN based Dynamic Resource Allocator are the two models integrated in this work – titledas "Defending the Overloading System in Cloud Computing using Preamble Fuzzy Estimator based RNN" (PFEROD). The proposed work is developed and tested in a real-time cloud environment to measure the benchmark parameters such as Resource utilization, Balance Degree, Average Response Time, Migration Cost, Throughput based on number tasks and Throughput based on number of Virtual Machines. A notable about of stability in the performance score is achieved by PFEROD putting in effort.

Keywords: *Fuzzy Logic, Load Balancing, RNN, Resource Allocation, Server overload, Server Optimization*

1. INTRODUCTION

Cloud computing finds its origins in the 1950s, when central mainframe systems were accessible through rudimentary, non-intelligent terminals. The concept of utility computing emerged in the 1960s, where computing resources were provided as a service.Time-sharing systems in the 1970s allowed multiple users to access a single computer simultaneously.In the 1990s, Application Service Providers (ASPs) began offering applications over the internet [1]. The dot-com boom saw the emergence of internet-based companies in late 90s, laying the foundation for web-based applications and services.Cloud computing gained mainstream acceptance, with businesses and individuals migrating data, applications, and services to the cloud.Microsoft Azure, IBM Cloud, and other major cloud providers entered the market [2].Hybrid [3] and multi-cloud [4] strategies became popular, allowing organizations to combine on-premises and cloud resources.

Cloud providers expanded their offerings to include artificial intelligence (AI) [5], machine learning (ML) [6], Internet of Things (IoT) [7], and container orchestration services.Kubernetes became a standard for container management and orchestration.Cloud computing has transformed how businesses operate, providing scalability, cost-efficiency, and flexibility. Its evolution continues to shape the IT landscape and drive innovation across various industries.Infrastructure as a Service represents a core paradigm within the foundational service models of cloud computing [8]. IaaS provides virtualized computing resources over the internet, allowing organizations to rent and manage essential IT infrastructure components, such as servers, storage, networking, and sometimes even virtualization software.Prominent IaaS providers include "Amazon Web Services" (AWS) "Elastic Compute Cloud" (EC2), "Microsoft Azure Virtual Machines", "Google Cloud Compute Engine", and "IBM Cloud Infrastructure". Users can use these platforms to create and manage virtual machines, storage, and networking components, building their own IT infrastructure in the cloud without the need for whatever kind corporealdevices ownership and maintenance [9].

Server overloading refers to a situation in which a computer server, whether physical or virtual, is subjected to a demand for resources that exceeds its capacity to handle effectively. This overload can result in a range of issues, including reduced performance, slower response times, service disruptions, or even complete server failure. Server overloading can occur for

various reasons, and it is a common concern in the context of web hosting, cloud computing, and data centers. Sudden Traffic spikes, Inadequate Resources, Inefficient applications, "Distributed Denial of Service" attacks, and hardware failures are the determinants behind the overloading [10].

Cloud server overload can cause complications for instance, Performance reduction, Server downtimes, Data loss, and Data corruption. Server overloading is a critical concern for organizations, as it can lead to significant disruptions and impact user satisfaction. Proactive management and the adoption of scalable, resilient infrastructure are essential for addressing this issue effectively. PFEROD work is one initiative to diminish the server overloading probabilities.

2. EXISTING METHODS

A substantial amount of efforts put into the concept of preventing server overloads from be beginning itself. Most recent related works in concern are selected here to give an elaborate insight about the methodologies used, their advantages and their limitations. "Proactive and dynamic load balancing model for workload spike detection in cloud" [11], "Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment" [12], "An Efficient Virtual Machine Consolidation Algorithm for Cloud Computing" [13], "An efficient multi-objective scheduling algorithm based on spider monkey and ant colony optimization in cloud computing" [14], and "Design of cultural emperor penguin optimizer for energy-efficient resource scheduling in green cloud computing environment" [15] works are studied in this section to analyze the performance and scope for new research works.

2.1. "Proactive and dynamic load balancing model for workload spike detection in cloud" (DLBM)

In 2023, Archana Patil et.al, posited a proactive everchanging load balancing Method (DLBM) to overcome the sudden spike in workloads. Instant overloading, Workload spikes, Repetitive migrations, and Service Level Agreement violations are common limitations in existing cloud scheduling algorithms. To mitigate these challenges in managing spike overloading, this paper proposes a proactive approach tailored to handle spike overloading through optimal solutions. The introduced "Dynamic Load Balancing Model" (DLBM), coupled with a spike detection algorithm, anticipates workload surges to prevent host overloading and Service Level Agreement (SLA) breaches. Experiments conducted using the Cloudsim tool and the PlanetLab workload dataset show that the DLBM effectively identified workload spikes and successfully managed frequent migrations. A few additional modules are introduced in DLBM work for the phases such as Dynamic load balancing architectural design, dynamic load balancing model creation, Resource utilization monitoring, Resource selection, and for Dynamic Threshold selection. Dedicated algorithms are provided in DLBM work to detect the workload spikes and to balance the dynamic loads.

The core focus of DLBM is the development of an efficient, dynamic load balancing model that proactively detects workload spikes and prevents overloading instances on hosts. DLBM anticipates these spikes, helping to mitigate sudden overloading issues. Trial findings show that the proposed DLBM reduces energy expenditure, recurrent migrations, overloading incidents, and the rate of SLA violations. While the CPU is the main resource for processing, other resources also play a crucial role in influencing processing performance.

Achievement of Power efficiency and reduced number of migrations is learned as the advantage of DLBM work whereas, Single resource constrain model causes subpar overall performance is noted as the limitation.

2.2. "Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment" (HDWO)

HDWO work is proposed by K.Ramya et.al., in 2023 to solve cloud overloading using Dingo and Whale optimization algorithms. This article introduces an innovative load balancing structure called HDWOA-LBM, which combines the attributes of "dingo" and "whale optimization algorithms" to enhance load balancing in cloud environments. The goal is to achieve maximum throughput, reliability, and resource utilization. HDWO is inspired by the hunting nature of dingoes, where tasks are akin to prey, and virtual machines (VMs) are the hunters.

HDWO work adopts the exploration and exploitation processes seen in dingo optimization algorithms (DOA), involving "chasing", "approaching", "encircling", and "harassing", to optimize the allocation of arriving workload to suitable VMs. Additionally, it leverages the strengths of whale optimization to improve the exploitation stage of DOA, striking a balance between local search and global search.

Simulation assessments conducted with CloudSim validate the effectiveness of HDWO, showing a 21.28% increase in throughput, a 25.42% boost in reliability, a 22.98% reduction in makespan, and a 20.86% improvement in resource allocation. These results are comparable to other competitive intelligent load balancing schemes explored during the investigation.

Improvements in throughput and reduction of makespan are the advantage of HDWO work. Ensemble of two different optimization algorithms occupies more computational resources while running in a dynamic cloud environment is understood as the limitation of HDWO work.

2.3. "An Efficient Virtual Machine Consolidation Algorithm for Cloud Computing" (EWMCA)

Ling Yuan et.al., proposed EWMCA work in 2023 for the purpose of efficient virtual machine amalgamation in cloud computing milieu. In EWMCAwork, a VMC protocol based on the load projection was developed to enhance efficiency. In the first phase, animmigrated VM selection strategy, known as LIP (Load Increment Prediction), was introduced. LIP combines present load and load expansion to enhance VM selection accuracy, especially for overloaded bodily machines (PMs). Subsequently, a VM migration criteria of selection called "SIR (Load Sequence Prediction)" was presented. SIR groups VMs with Supplementary load profiles onto the same PM, improving PM load stability and consequently reducing SLAV and VM migrations due to resource competition among PMs. Lastly, a more effective "Virtual Machine Consolidation" (VMC) algorithm based on load estimation from LIP and SIR was proposed. Results of the conducted experiments demonstrate the significant enhancement in energy efficiency achieved by this VMC algorithm

EWMCA work is filled with plenty of algorithms for different types of tasks such as Short-Term load increment prediction, Load growth trend detection, Load volatility, Load weight calculation, Weighted curve fitting, Curve evaluation, Load sequence prediction, Load sequence standardization, Load sequence function conversion, Load sequence similarity calculation, VM Migration selection Strategy, Load fitness function modeling, Computation of saturation, Saturation increase rate determination, and identification of underloaded host process. The EWMCA approach leverages predicted future load data for virtual machines (VMs) and physical machines (PMs) to improve virtual machine consolidation, ultimately enhancing mobile cloud computing performance. Experimental results indicate that both the "LIP" migration virtual machine picking mehtod and the SIR virtual machine exodus policy selection strategy surpass other methods across multiple evaluation criteria.

These findings indicate that one VMC algorithm presented here adequately enhances the overall cloud service operational performance in blockchain, encompassing aConsistency, stability, and efficient energy handling -which is the stated advantages of EWMCA work. Load sequence of different resources causes more dynamic calculations that may cause performance issues which is noted as the limitation of EWMCA work.

2.4. "An efficient multi-objective scheduling algorithm based on spider monkey and ant colony optimization in cloud computing" (SMOACO)

In 2023, Amer D.A. et.al., introduced SMOACO work in pursuit of applying multi-objective coordinationas per Spider Monkey and Ant Colony optimizations. In addition, a fitness function is defined to handle four scheduling problem objectives, by taking into account in relation totime frame of the schedule, execution expense, energy consumption, and provision utilization. SMOACO work is establishedby using the "Cloud Sim toolkit" and assessed under various workloads. Its effectiveness is evaluated using several metrics and compared against the most uo-to-date existing computational methods. These results demonstrate that the SMOACO approach efficiently allocates resources during enhancing performance of the cloud to increase profitability. Several models related to cloud computing environment such as Cloud model, Virtual Machine model, Task model and Multi-Objective models are defined distinctively in SMOACO work. Similarly, different phases as "Initialization phase", "Local leader phase", "Global leader phase", "Global leader learning phase", "Local leader learning phase", Local leader selection phase, and Global leader selection phase are discussed clearly in SMOACO work.

A novel multi-objective scheduling technique. SMOACO combines the "SMO" and "ACO" algorithms to enhance the Speed of convergence for the optimal answer while addressing multiple scheduling objectives. The algorithm aims to optimize four key objectives, which include minimizing energy consumption, makespan, and deployment expenses, while maximizing provision utilization. SMOACO is evaluated using various workloads within the CloudSim toolkit and its performance is weighted to well-liked scheduling algorithms like "ACO", "GA", "WWO", and "HHO", utilizing diverse performance metrics. Experimental results demonstrate the efficiency of the SMOACO algorithm in resource allocation, leading to more stable and satisfactory itineraries for end-user application tasks.

Achievement of reduced energy consumption and makespan is the noted advantage of SMOACO work.The evaluation process is carried out using CloudSim simulation for public cloud environment. Performance of SMOACO is not evaluated for private and hybrid cloud environments that causes performance uncertainty – which is observed as the limitation of SMOACO work.

2.5. "Design of cultural emperor penguin optimizer for energy-efficient resource scheduling in green cloud computing environment" (EERSCEPO)

EERSCEPO is proposed by R.F.Mansoor et.al., in 2022 for energy efficient resource scheduling in green cloud computing atmosphere. Green cloud computing (GCC) has emerged as a fresh computing platform with the primary goal of addressing energy expense in the "cloud data centers" (CDC). Load equilibrationis commonly applied to improve resource utilization, "Throughput", and "Latency". In pursuit of reducing energy consumption in GCC data centers, EERSCEPO work presents the design of an energy-efficient provision assignment schedule system using the "Cultural Emperor Penguin Optimizer (CEPO)" algorithm, referred to as "EERS-CEPO" within the GCC environment.

EERSCEPO model aims to share workload across multiple data centers and resources, thereby preventing individual resource overloads. The CEPO algorithm is developed by combining the "Cultural Algorithm" (CA) with the "Emperor Penguin Optimizer (EPO)", enhancing EPO's exploitation capabilities with CA. This hybrid approach introduces a novel aspect to the work. The EERSCEPO algorithm integrates a fitness-function to optimize resource scheduling within data centers, aiming to minimize "operational & maintenance" costs in green cloud computing while subsequently reducing energy consumption and heat generation.

To validate the improved effectiveness of the EERSCEPO algorithm, and a comprehensive set of experiments is conducted. Better performance than the compared methods is the advantage of EERSCEPO method. VM migration policies and fault tolerance methodologies are not included in EERSCEPOwork that prevents achievement of better results during the scalability process. Most of the state-of-the-art cloud environments provide elastic cloud scalability, thus missing VM migration policies in EERSCEPO work is identified as the limitation.

A summary of existing works, methodologies used and the limitations are listen in Table 1.

Table 1: Existing methods Summary

Author	Work	Methodology	Advantages	Limitations
Archana Patil et.al,	"Proactive and dynamic load balancing model for workload spike detection in cloud"	Dynamic Load Balancing	Power Efficiency, Minimized migrations	Single resource dependency
K.Ramya et.al.,	"Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment"	Dingo and Whale Optimization	Throughput, Makespan	Deficient performance in Dynamic environments
Ling Yuan et.al.,	"An Efficient Virtual Machine Consolidation Algorithm for Cloud Computing"	Load sequence prediction	Accuracy, Stability, Energy efficiency	Single resource dependency
Amer D.A. et.al.,	"An efficient multi-objective scheduling algorithm based on spider monkey and ant colony optimization in cloud computing"	"Spider Monkey and Ant Colony Optimizations"	Energy efficiency, Makespan	Limited to Public Cloud environment
R.F.Mansoor et.al.,	"Design of cultural emperor penguin optimizer for energy-efficient resource scheduling in green cloud computing environment"	Cultural Emperor Penguin Optimizer	Energy efficiency, Response time	Scalability issues

3. BACKGROUND WORKS

Proposed PFEROD work uses Fuzzy logic and Recurrent Neural Network (RNN) as the base to contrive the new methodology. Expounding the basic principles of Fuzzy and RNN will help to give a clear interpretation about the proposed modules thus the essential explanations are provided in this section.

3.1. Fuzzy Logic

Analog and Digital are the two primary processing procedures. While measuring something, analog method used to perform the linear operations to sample the input data to a high precision. Digital processing is operates based on binary principle, thus it can produce either 0 or 1 as the measurement quantity. But most of the real-world environments neither require analog nor digital for processing. Therefore, fuzzy logic is introduced to play the main character in the processing field. Fuzzy logic uses partial quantitate values such as low, medium or high, which is a more convenient to handle in the real-time scenario. Fuzzy logic is a mathematical architecture that improves classical binary (true/false) logic to deal with uncertainty plus vagueness in decision-making. It was developed by Lotfi Zadeh in the 1960s as a means of capturing the imprecise nature of human reasoning and natural language. In fuzzy logic, variables can have degrees of truth between 0 and 1, allowing for a more nuanced representation of reality [16]. Fuzzy logic has found applications in various fields, that includes"control systems", "artificial intelligence", and "decision support systems", where it excels at handling complex, real-world problems that involve ambiguity and incomplete information. Its ability to model and manage uncertainty has made it a valuable tool for designing systems that can make intelligent, human-like decisions [17] in situations where traditional binary logic falls short. A simple diagram to demonstrate the fuzzy logic is illustrated in Figure 1.

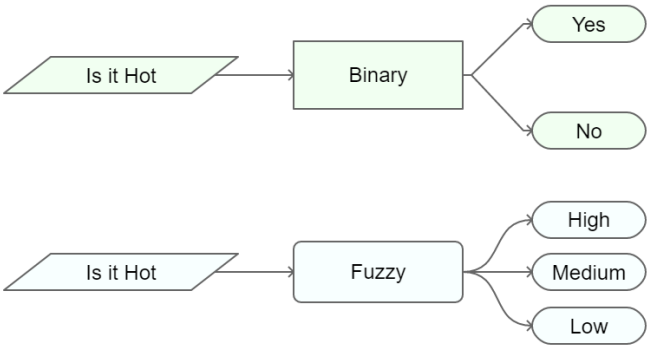

Figure 1: Binary Vs Fuzzy

3.2. Recurrent Neural Network (RNN)

"Recurrent Neural Networks (RNNs)" are the class of "Artificial Neural Networks" specially devised for processing sequences of datapoints. In contrast to traditional feedforward neural networks, RNNs have links that connect back on themselves, permitting them to maintain a memory of past inputs. This recurrent nature makes them highly suitable for tasks related to sequential data, such as NLP, speech recognition, time series analysis, and more. RNNs excel at capturing temporal dependencies and patterns within sequential data, making them a fundamental building block in many advanced machine learning applications. However, they do have challenges, such as vanishing gradients contributing to the advancement of more sophisticated RNN architectures like "Long Short-Term Memory (LSTM)" [18] and "Gated Recurrent Unit (GRU)"[19] networks to address these issues and improve their performance.

GRU and LSTM networks are both advanced recurrent neural network (RNN) architectures used for handling sequential data. While they share similarities, such as their ability to capture long-term dependencies, there are some advantages to using GRUs over LSTMs in certain situations such as Simplicity, Reduced overfitting, Training Speed and Memory efficiency. Therefore, GRU based RNN is selected to use with one of the proposed modules of PFEROD work. A standard GRU architecture is visualized in Figure 2.

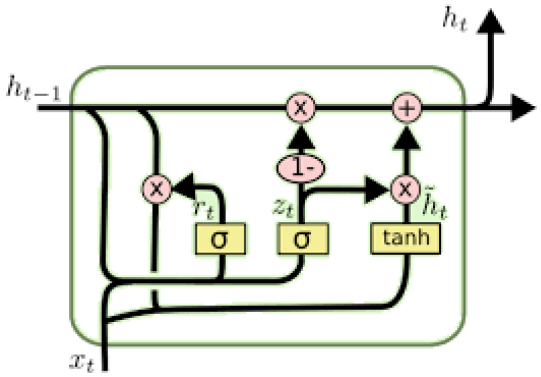

Figure 2: Standard GRU model

4. PROPOSED METHOD

Proposed PFEROD method consists of two novel functional modules namely Preamble Fuzzy Load Estimator, and RNN based Dynamic Resource Allocator. Functionality procedures of these methods and their integration are comprehensively explained in this section.

4.1. Preamble Fuzzy Load Estimator (PFLE)

PFLE module is used to estimate the computational resource consumption based on the work queue. Tasks are represented and managed as part of the overall cloud computing environment. The representation of a task in cloud computing typically involves several components and metadata about the tasks such as description identifier, parameters, process state, dependencies and execution information. Tasks may have various parameters or settings that control their behavior, such as input data, execution time limits, priority levels, and resource requirements such as CPU, Memory, and Storage. While considering memory, it can be further classified into high speed Cache memory and regular Random-Access Memory (RAM). Tasks can have dependencies on other tasks, meaning they rely on the output or results of other tasks before they can proceed. Managing these dependencies is crucial to ensuring the correct execution sequence.The hypervisor, which

manages the VMs, often provides built-in monitoring tools to measure CPU usage. These tools can report metrics like CPU utilization, ready time, Cache usage, Memory Usage and Storage Usage. CPU usage is typically measured as a percentage, representing the portion of CPU capacity used. Monitoring tools and methods provide insights into factors like CPU load, idle time, usage peaks and long average usage duration.

Important computational resources like CPU, Cache, RAM, and Storage are represented as a Set ρ in PFLE module with members $\rho_P, \rho_C, \rho_M, \rho_S$ in which ρ_P refers the Processor usage, ρ_C refers the Cache usage, ρ_M refers the Memory usage, and ρ_S refers the storage requirements of a task. The virtual machines are used to assign a sequence of tasks to complete a process. Therefore, a task queue is maintained for every VM in cloud environment. PFLE treats the Task Queue as a task set $\varphi = \{\varphi_1, \varphi_2 \dots \varphi_{n_\varphi}\}$ where n_φ is the number of tasks in the queue. For every taskφ_x, the resource consumption Set ρ_x is mapped for along the corresponding member values. Therefore, for the task queue, the computational resource set is assigned as $\rho = \{\rho_1, \rho_2 \dots \rho_n\}$ and the expansion is visualized as $\rho = \{\rho_1 = \{\rho_P, \rho_C, \rho_M, \rho_S\}, \rho_2 = \{\rho_P, \rho_C, \rho_M, \rho_S\} \dots \rho_{n_\varphi} = \{\rho_P, \rho_C, \rho_M, \rho_S\}\}$

A dedicated equation is used in PFLE module to combine different resource types such as CPU, Cache, RAM, and Storage into a single resource usage representation quantity γ_x

$$\gamma_x = \left(2^3 \times \left(\frac{100}{\rho_{P_{max}}} \times \rho_P\right) + 2^2 \times \left(\frac{100}{\rho_{C_{max}}} \times \rho_C\right) + 2 \times \left(\frac{100}{\rho_{M_{max}}} \times \rho_M\right) + \left(\frac{100}{\rho_{S_{max}}} \times \rho_S\right)\right) \quad \text{Equation (1)}$$

where $\rho_{P_{max}}, \rho_{C_{max}}, \rho_{M_{max}}$, and $\rho_{S_{max}}$ are the maximum allocated capacity of the resources Processor Cache Memory, and RAM respectively with the predicament $\{\rho_{P_{max}}, \rho_{C_{max}}, \rho_{M_{max}}, \rho_{S_{max}}\} \in \rho_x$.

The weights 2^x are used to prioritize the resources in a hierarchical sequence of the resource that is Processor, Cache, Memory and Storage listed from the primary.The highest possible value γ_{max} of γ by equation 1 is 1500.

The state of the virtual machine is divided into 4 different states such as Low, Medium, High, and Critical depending on the resource consumption of the particular VM. A fuzzy equation is used in PFLE to determine the state $\delta_x$of a VM for the task φ_x

$$\delta_x = \begin{cases} Low\ if\ \gamma_x < \frac{1}{4}\gamma_{max} \\ Medium\ if\ \frac{1}{4}\gamma_{max} \leq \gamma_x < \frac{1}{2}\gamma_{max} \\ High\ if\ \frac{1}{2}\gamma_{max} \leq \gamma_x < \frac{3}{4}\gamma_{max} \\ Critical\ otherwise \end{cases} \quad \text{Equation (2)}$$

VM state of the tasks are accumulated in a set Δ such as $\Delta = \{\delta_1, \delta_2 \dots \delta_{n_\varphi}\}$for the proceedings with the successive module

The interactions between PFLE module with Cloud Services, VMs, Hypervisor and Task queue are portrayed in Figure 3.

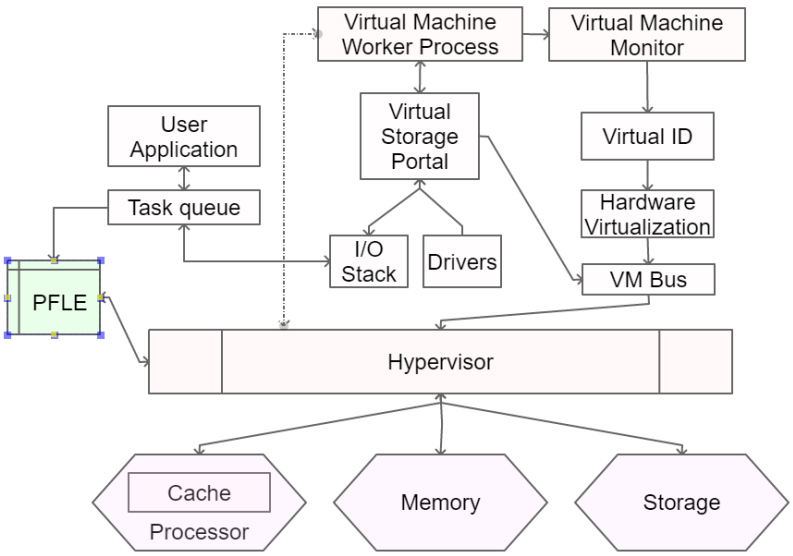

Figure 3: PFLE interaction with Tasks Queue and Hypervisor

4.2. RNN based Dynamic Resource Allocator (RNN-DRA)

RNN-DRA module operates based on two influential factors. The first one is the output of PFLE that gives the estimation of the required resources for the upcoming tasks. A timeseries based load forecasting which is achieved using RNN model. The load estimation determined my the PFLE may diverge due to several causes such as varying workloads, Inaccurate projections, Application changes, Failure scenarios, Run time errors, User Behavior, Security attacks and seasonal patterns.

Workloads in a cloud environment are often dynamic and subject to change. Seasonal factors, day-to-day fluctuations, and unexpected spikes in usage can lead to load estimation differences. For example, an e-commerce website may experience higher traffic during holiday sales, leading to increased workloads.The initial load estimation may not accurately reflect actual usage. Overestimating or underestimating load can result in resource underutilization or over-provisioning, leading to increased costs or degraded performance.Updates or modifications to applications can impact load estimation. Changes in code, features, or data handling can affect the amount of resources required, making previous estimates obsolete.Load estimation may differ in the event of hardware failures, network outages, or other disruptions that can lead to a shift in the distribution of workloads. Failover mechanisms and redundancy strategies may need to accommodate these scenarios.Cloud environments often employ auto-scaling mechanisms to adapt to varying workloads. Load estimation may differ when scaling policies and triggers are configured to respond to load thresholds, leading to resource allocation changes that cause runtime errors. Changes in user behavior, such as new usage patterns or increased interactions with applications, can lead to load estimation differences. It's essential to monitor and adapt to evolving user needs.Load estimation can change in response to security incidents, such as DDoS attacks or attempts to breach the cloud infrastructure. Increased traffic and resource utilization may occur during these events.Some applications experience significant load variations due to seasonal factors. For example, tax-related services may see increased load during tax season.

RNN-DRA module is indented to manage these divergences in workload estimation by the inclusion of tracking and predicting time-based resource load pattern. That is RNN-DRA module encompasses the task-based load estimation and time-based load prediction to allocate resources in a more conscientious way. Data collection and data preprocessing phases are not included because the proposed RNN-DRA model collects data from the bootstrap, and the reliability of hypervisor's ability to provide high precision noiseless data. GRU model is used in the proposed RNN-DRA module that receives the resource usage pattern as the input and initiates the training process during the first run. For every successive executions RNN-DRA uses the knowledgebase created during the previous sessions, and the new weights are updated during the ongoing execution.

The equations used for Update gate "Reset gate", "Hidden state", and "Candidate hidden state" of RNN-DRA modules are as follows

Update Gate:

$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right) \quad \text{Equation (3)}$$

Where x_t is the input at timestamp t, W is the weight of "input data", h_{t-1} is the antecedent timestamp hidden state and U is the corresponding weight of applied to "Hidden State".

Reset Gate:

$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right) \quad \text{Equation (4)}$$

Hidden State:

$$h_t' = \tanh\left(w_{x_t} + r_t \otimes Uh_{t-1}\right) \quad \text{Equation (5)}$$

Candidate Hidden State:

$$h_t = z_t \otimes h_{t-1} + (1 - z_t) \otimes h_t') \quad \text{Equation (6)}$$

Where $\otimes$ is the Hadamard product, that is elementwise multiplication

A dedicated component Transition Gate (TG) is introduced to standard GRU cell to extract the intermediate data ω_t from the GRU cell.Since there is no specific output gate defined for standard GRU model, ω_t is used for the purpose of proposed dynamic resource module. The intermediate result from the TG gate determines the RNN predicted load category that can be of any of the three states such as Below normal, Normal and Above Normal as helmed by the following equation

$$\omega_x = \begin{cases} Below\ normal\ if\ h_t < \frac{1}{3} \\ Normal\ if\ \frac{1}{3} \le h_t < \frac{2}{3} \\ Above\ Normal\ otherwise \end{cases} \quad \text{Equation (7)}$$

The GRU-RNN architecture of the proposed module is given in Figure 4.

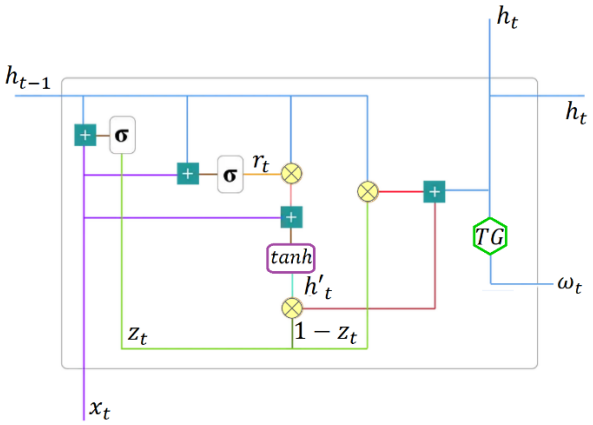

Figure 4: Proposed GRU Cell Architecture

The task-based resource estimation is provided by the proposed PFLE module. The time- based resource usage is predicted by the proposed GRU module. The dynamic resource allocation is performed based on the outputs of PFLE module and proposed GRU. A steadfast algorithm is introduced in this module to allocate computational resources dynamically based on the aggregation of PFLE and GRU results given below.

Algorithm: Dynamic Resource Allocation (DRA)

Input: δ_x, ω_t

Output: Resource allocation

Step 1: Get δ_x from PFLE

Step 2: Get ω_t from GRU

Step 3: If ω_t = Below Normal

Step 4: If $\delta_x = Low$, Release resource by 40%

Step 5: else if $\delta_x = Medium$ Release resource by 20%

Step 6: else if $\delta_x = High$, Maintain resource as it is

Step 7: else if $\delta_x = Critical$, Expand resource by 20%

end if (Step 4)

end if (Step 3)

Step 8: If $\omega_t = Normal$

Step 9: If $\delta_x = Low$, Release resource by 20%

Step 10: else if $\delta_x = Medium$, Maintain resource as it is

Step 11: else if $\delta_x = High$, Expand resource by 20%

Step 12: else if $\delta_x = Critical$, Expand resource by 40%

end if (Step 9)

end if (Step 8)

Step 13: If $\omega_t = Above\ Normal$

Step 14: If $\delta_x = Low$, Maintain resource as it is

Step 15: else if $\delta_x = Medium$, Expand resource by 20%

Step 16: else if $\delta_x = High$, Expand resource by 40%

Step 17: else if $\delta_x = Critical$, Expand resource by 60%

end if (Step 14)

end if (Step 113)

Step 18: Continue from Step 1 for next timestamp

The resource free up or allocation threshold is set to 20% in DRA algorithm since most of the service providers allow this level to avoid SLA violations. Through the proposed modules all together the cloud server overloads are diminished by efficient and dynamic resource allocation.

5. EXPERIMENTAL SETUP

A computer with i7 4.7 GHz processor, 16GB RAM and 1TB SSD is used to develop the entire model. A committed User Interface (UI) is designed using Visual Studio IDE [20], and the methodology is coded using C++ 20.0 [21] programming language. The UI is designed in a way to upload the resource management scripts which can interact with the hypervisor of the cloud server. A dedicated server is leased from i2k2 cloud service providers [22], to measure the accomplishment of the offered method. The communication between the UI and the Server is accomplished through Common Could Gateway Interface with registered credentials by the service provider.

The user interface uploads the resource allocation schema to the hypervisor of the dedicated server, and the resource utilization data are fetched from the server to prepare the log report file and to generate the graphs. The performance parameters are logged while the dedicated server is handling the real-time data which is shared by different sites such as Dropbox and Foxrenderfarm. Websites such as dropbox.com produce higher storage and data transferring demands, whereas websites specifically foxrenderfarm.com demands intensive processor related tasks. The combinational data from these websites produce a distributed range of computational resources which is used to carryout the experiments more legitimately.

6. RESULTS AND DISCUSSIONS

Benchmark cloud resource management metrics such as “Resource Utilization”, “Degree of Balance”, “Average Response Time”, “Migration Cost”, “Throughput” for different number of tasks, and Throughput for different number of Virtual Machines are logged during the experiments carried out. A heap of 100 to 1000 number of tasks are executed in 2 to 20 number of virtual machines and the readings are deliberated in this section.

6.1. Resource Utilization

Resource utilization is a benchmark parameter in cloud computing because it contributes a fundamental role in optimizing the efficiency, cost-effectiveness, and overall performance of cloud services and infrastructure.Resource utilization in cloud computing is measured by monitoring and collecting data on various system resources and components to understand how they are being used. These resources include CPU, memory, storage, and network bandwidth. Cloud providers often offer built-in monitoring and management tools through which, resource utilization scores for existing and proposed methods are measured and enumerated in the following table.

Table 2: Resource Utilization

esource Utilization (%)						
Number of Tasks	**SMOACO**	**EVMCA**	**EERSCEPO**	**HDWO**	**DLBM**	**PFEROD**
100	34.076923	28.102564	27	37	28.10256	36.076923
200	51.562305	47.919262	46.368484	52.6792	47.36848	54.08744
300	61.698673	59.515644	57.535759	61.81031	58.5614	64.729843
400	68.970764	67.735962	65.583122	68.3584	66.53184	72.174881
500	74.591545	74.132019	71.836647	73.44901	72.81101	78.015121
600	79.158417	79.357979	76.852959	77.56643	77.85296	82.765999
700	83.092606	83.827751	81.086273	80.9451	82.13756	86.705879
800	86.456139	87.629578	84.900322	84.06324	85.84905	90.211037
900	89.371704	91.031288	88.071518	86.62061	89.17409	93.305832
1000	92.025642	94.102562	91.051285	89.05129	92.05129	96

As per the observed results, it is attested that the proposed PFEROD method achieved96% as the maximum resource utilization score, which is higher than the methods in comparison. The minimum resource utilization of PFEROD is sustained with 36% with the average of 75.41%. EVMCA follows with the subsequent resource utilization Maximum, Minimum, and

Average values 94%, 28%, and 71% respectively A comparison graph of resource utilization reading is provided in Figure 5 for pictorial representation.

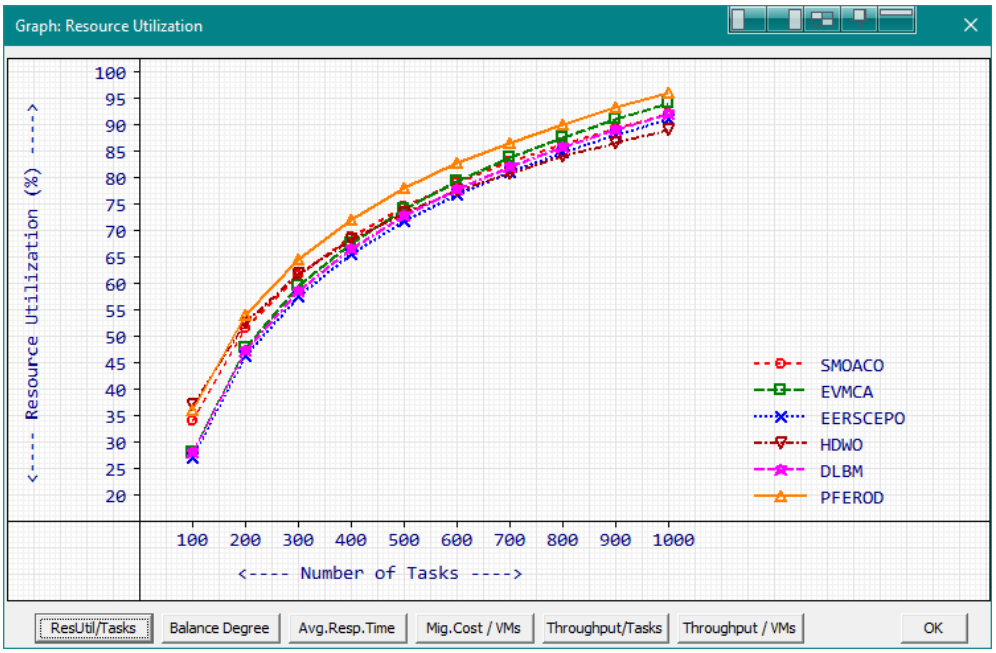

Figure 5: Resource Utilization

The performance rank sequence of the compared methods in terms of resource utilization is PFEROD, EVMCA, DLBM, SMOACO, EERSCEPO, and HDWO listed from the best.

6.2. Degree of Balance

The Degree of Balance parameter is the consolidation of several cloud computing parameters such as Load balancing, Resource balancing, Auto-Scaling, Resource optimization, and Availability index. Degrees of balance is calculated using the formula

$$Degree\ of\ Balance = \frac{\frac{1}{n}\sum_{i=1}^{n} d_{l_i} + \frac{1}{n}\sum_{i=1}^{n} d_{r_i} + \frac{1}{n}\sum_{i=1}^{n} d_{a_i} + \frac{1}{n}\sum_{i=1}^{n} d_{o_i} + \frac{1}{n}\sum_{i=1}^{n} d_{v_i}}{5}$$

Where d_{l_i} refers the Load balancing index, d_{r_i} is the Resource balancing index, d_{a_i} is Auto-scaling index, d_{o_i} is the optimization index, d_{v_i} is the Availability index, and n is the number of tasks performed by the virtual machine.

The degree of balance is directly proportional to the performance of a cloud computing environment. A higher degree of balance indicates the higher performance of the computational environment. Measured Degree of Balance values for discussed methods are given in Table 3.

Table 3: Degree of Balance

Degree of Balance (%)						
Number of Tasks	**SMOACO**	**EVMCA**	**EERSCEPO**	**HDWO**	**DLBM**	**PFEROD**
100	80.051285	78.025642	76	82.02564	76.5	81.051285
200	84.240059	83.14315	80.86776	84.45952	81.3934	85.51545
300	86.679695	86.111061	83.685219	85.91953	84.13394	88.156822
400	88.50576	88.286301	85.684242	86.84212	86.23553	90.133461
500	89.811218	89.985054	87.286087	87.6174	87.70916	91.48455

600	90.971039	91.254211	88.552986	88.22521	88.97606	92.672272
700	91.908295	92.417946	89.521568	88.86335	90.12413	93.753395
800	92.643257	93.378174	90.44944	89.32729	91.02636	94.597633
900	93.436317	94.299049	91.370445	89.7365	91.79352	95.390564
1000	94	95.07692	92	90.10256	92.5	96.102562

The observational outcomes show that proposed PFEROD method gained the maximum degrees of balance score 96%. The guaranteed degree of balance value of 81% is assured by PFEROD method. The degree of balance average score of PFEROD is 90.88%. The performance rank sequence of the discussed method in terms of Degree of Balance is PFEROD, SMOACO, EVMCA, HDWO, DLBM, EERSCEPO, and with the degrees of balance score averages 90.88%, 89.22%, 89.19%, 78.31%, 87.31, 87.03%, and 86.54% in order listed from the best. When comparing the methods SMOACO and EVMCA, Higher balance degree index 95.07 is achieved by EVMCA, at the same time the higher balance degree average of 89.22 is achieved by SMOACO method. Both EVMCA and SMOACO are very competitive with each other, but not matching with the performance of proposed PFEROD method.

The comparison graph for Degrees of balance is given in Figure 6.

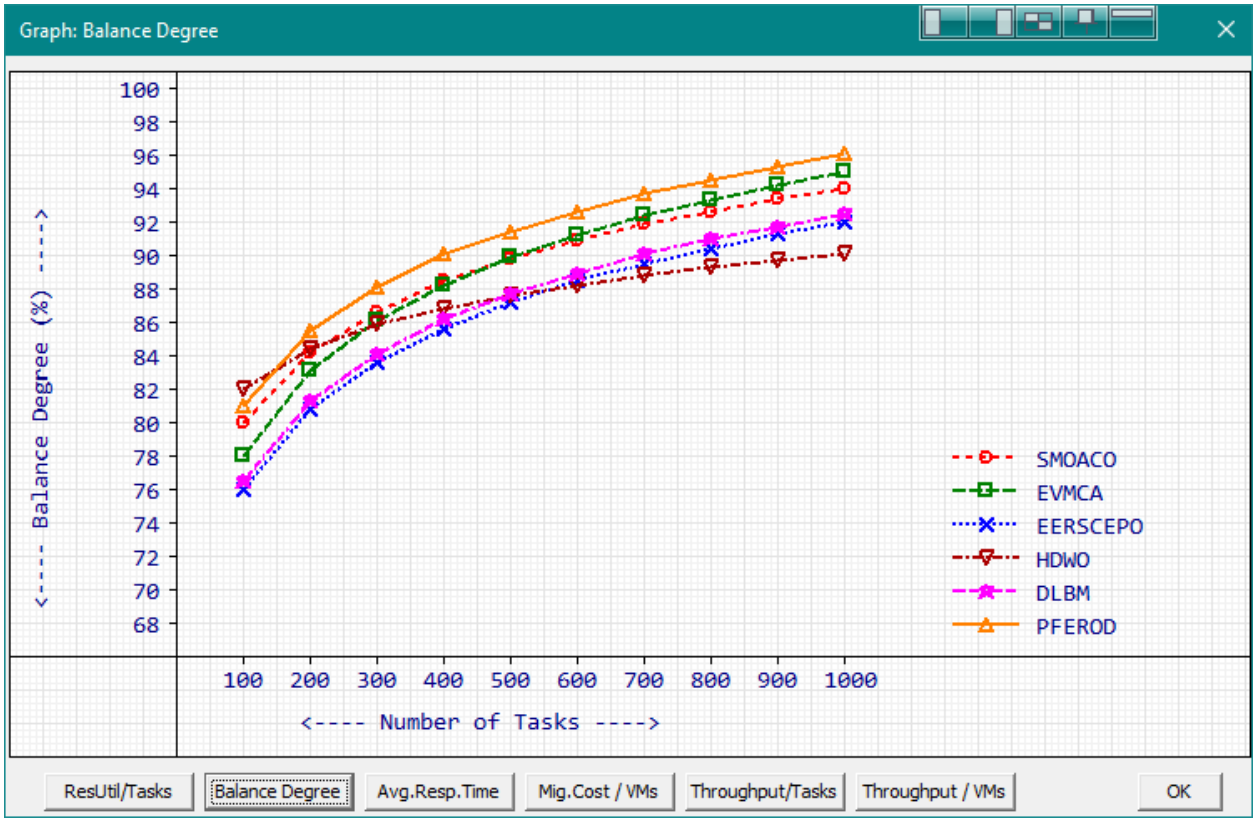

Figure 6: Degree if Balance

6.3. Average Response Time

Average response time is a vital metric in cloud computing because it affects user experience, cost efficiency, scalability, compliance with SLAs, competitive advantage, and the overall performance and success of cloud-based applications and services. Monitoring and optimizing response times are essential for businesses and organizations leveraging cloud computing to achieve their goals. Average response time is inversely proportional to the quality of the provided service, that is, the lesser average response time indicates the higher quality of the service provided.

Common tools and services for measuring response times in cloud computing include “Application Performance Monitoring (APM)” solutions, “Log Analysis Tools”, and cloud provider-specific monitoring services. A service provider specific cloud resource monitor desktop client is used to measure the average response time. The general measurement unit of Average

response time is milli-seconds (mS). verage response time for the cited methods are measured during the experiments carried out for every 100 number of tasks in the range from 100 to 1000 and the readings are recorded in Table 4.

Table 4: Average Response Time

Average Response Time (mS)						
Number of Tasks	**SMOACO**	**EVMCA**	**EERSCEPO**	**HDWO**	**DLBM**	**PFEROD**
100	80.051285	78.025642	76	82.02564	76.5	81.051285
200	84.240059	83.14315	80.86776	84.45952	81.3934	85.51545
300	86.679695	86.111061	83.685219	85.91953	84.13394	88.156822
400	88.50576	88.286301	85.684242	86.84212	86.23553	90.133461
500	89.811218	89.985054	87.286087	87.6174	87.70916	91.48455
600	90.971039	91.254211	88.552986	88.22521	88.97606	92.672272
700	91.908295	92.417946	89.521568	88.86335	90.12413	93.753395
800	92.643257	93.378174	90.44944	89.32729	91.02636	94.597633
900	93.436317	94.299049	91.370445	89.7365	91.79352	95.390564
1000	94	95.07692	92	90.10256	92.5	96.102562

The experiment results shows that the proposed method attains the best average response time 372mS. The performance rank in terms of average response time is PFEROD, EVMCA, SMOACO, DLBM, EERSCEPO, and HDWO with the values 1350 mS, 1406 mS, 1462 mS, 1504 mS, 1519 mS, and 1538 mS corresponding to the order listed from the best.

The comparison graph for Average Response Time is given in Figure 7.

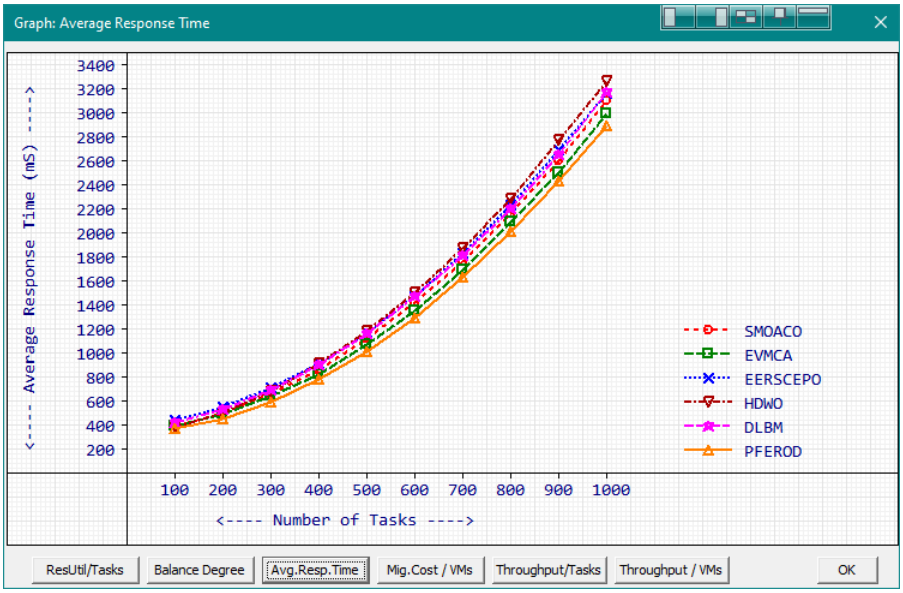

Figure 7: Average Response Time

6.4. Migration Cost

Migration cost is an important aspect of cloud computing that impacts budgeting, decision-making, risk management, resource allocation, and the capacity to fulfil the potential benefits of the cloud. Cost tracking and management tools provided by the cloud service provider is used to monitor and analyze expenses throughout the migration process and after the migration is completed. The migration cost is measured for 2 to 20 virtual machines Recorded migration costs are given

in Table 5, and the comparison graph is provided as Figure 8.

Table 5: Migration Cost

Migration Cost (%)						
Number of VMs	**SMOACO**	**EVMCA**	**EERSCEPO**	**HDWO**	**DLBM**	**PFEROD**
2	**10.076923**	**14.051282**	**15**	**9.025641**	**14.05128**	**10**
4	**14.265702**	**16.433882**	**19.618013**	**16.02626**	**18.56673**	**12.760552**
6	**16.73098**	**17.842611**	**22.182461**	**20.02507**	**21.18246**	**14.345373**
8	**18.454481**	**18.842121**	**24.133465**	**22.84738**	**23.0309**	**15.44418**
10	**19.78558**	**19.617401**	**25.561474**	**25.15323**	**24.51019**	**16.367653**
12	**20.894117**	**20.276491**	**26.723551**	**26.89748**	**25.69791**	**17.029003**
14	**21.908297**	**20.786425**	**27.67647**	**28.48854**	**26.75339**	**17.657164**
16	**22.745825**	**21.22472**	**28.571991**	**29.87363**	**27.59763**	**18.179092**
18	**23.385036**	**21.710863**	**29.313637**	**31.05014**	**28.33928**	**18.665106**
20	**24.051283**	**22**	**30**	**32.05128**	**29.10256**	**19.02564**

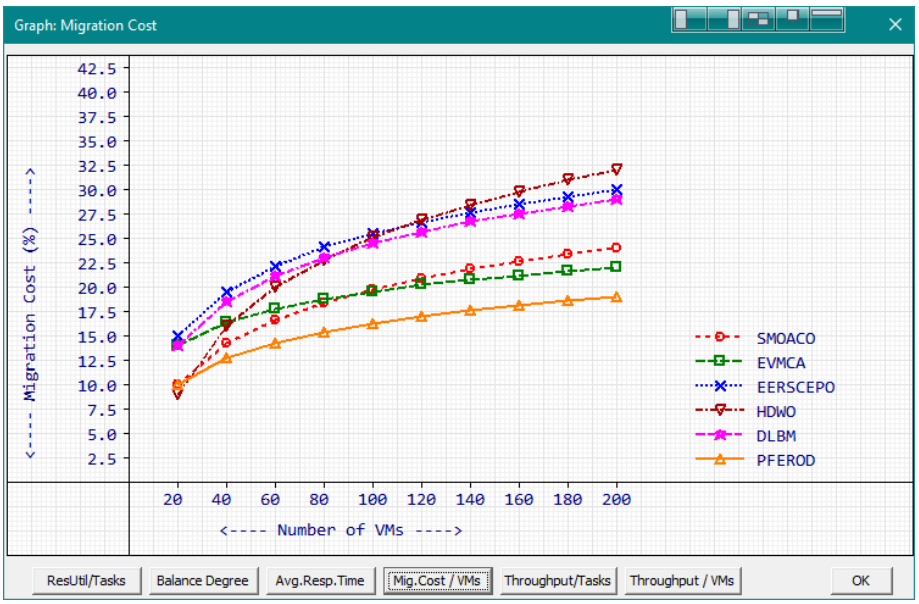

Figure 8: Migration Cost

Based on the observations, the lowest migration cost is achieved by the proposed method with a minimization of 3.3% when compared with the nearest performer SMOACO. The performance rank sequence of discussed methods based on migration cost average is PFEROD, SMOACO, EVMCA, DLBM, HDWO, and EERSCEPO with the values 15.95%, 19.23%, 19.28%, 23.88%, 24.14%, and 24.88% respectively.

6.5. Throughput

Throughput is a fundamental metric in cloud computing as it impacts user satisfaction, resource utilization, scalability, cost efficiency, and overall performance. Monitoring, optimizing, and maintaining high throughput are essential for the successful delivery of cloud services and applications. Throughput is measured with respect to "Number of Tasks" and "Number of Virtual Machines" during the conducted experiment. The results are presented in Table 6 and Table 7.

Table 6: Throughput / Tasks

Throughput (kB) / Number of Tasks						
Tasks	**SMOACO**	**EVMCA**	**EERSCEPO**	**HDWO**	**DLBM**	**PFEROD**
100	20	17	16	22	17	21
200	45	44	41	45	40	47
300	67	69	67	70	63	72
400	91	93	89	89	83	94
500	108	112	104	102	98	115
600	121	128	117	114	108	130
700	126	143	124	118	118	146
800	129	143	125	116	119	152
900	135	143	124	113	112	158
1000	126	140	112	99	112	149

Table 7: Throughput / Number of Virtual Machines

Throughput (kB) / Number of Virtual Machines						
Number of Tasks	**SMOACO**	**EVMCA**	**EERSCEPO**	**HDWO**	**DLBM**	**PFEROD**
2	20	18	17	21	19	21
4	50	56	45	46	47	61
6	68	80	62	61	63	84
8	81	96	73	72	75	101
10	90	108	84	81	86	113
12	99	117	90	86	92	125
14	104	126	96	94	100	135
16	110	136	104	98	103	144
18	116	142	106	105	112	151
20	120	146	115	107	116	156

Best Throughput average with respect to number of tasks 108.4 kB is achieved by proposed PFEROD method. The performance rank sequence for throughput average is PFEROD EVMCA, SMOACO, EERSCEPO, HDWO, and DLBM with the values 108.4 kB, 103.2 kB, 96.8 kB, 91.9kB, 88.8 kB, and 87 kB given in order listed from the best. The comparison graph for Throughput with respect to number of tasks is given in Figure 9.

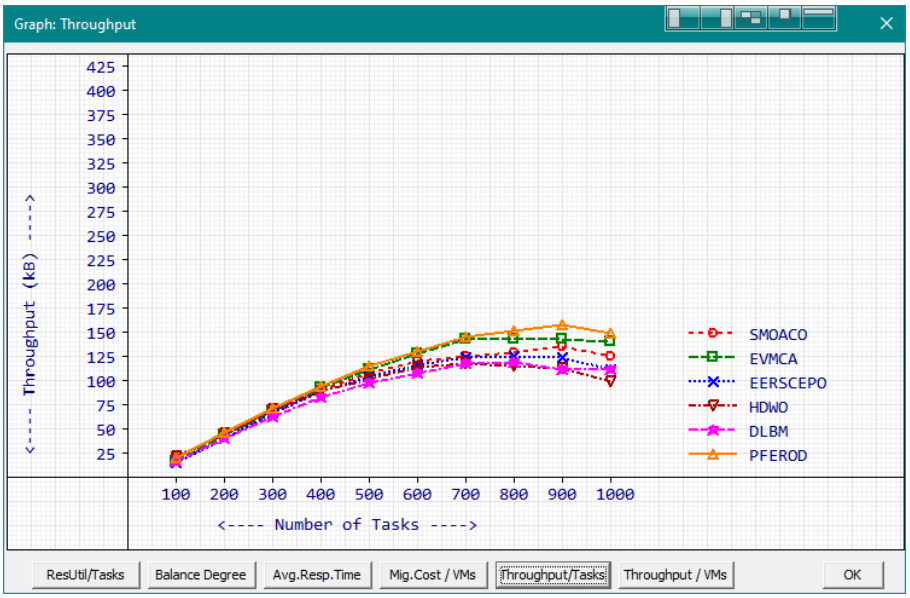

Figure 9. Throughput / Tasks

It is understood that the throughput begins to saturate around 700 number of tasks irrelevant of the resource scheduling procedure. During the throughput measurement with respect to "number of tasks", the performance improves along with the number of tasks till the count 700 to 800, then the performance is dropping down while increasing the number of tasks more than 750.

When considering the Throughput parameters with respect to Number of Virtual Machines, proposed PFEROD method secured the highest throughput value of 156 kb. The highest throughput achievement sequence is PFEROD, EVMCA, SMOACO, DLBM, EERSCEPO, and HDWO with the values 156 kb, 146 kb, 120 kb, 116 kb, 115 kb, and 107 kb respectively

The performance rank sequence based on Throughput average score with respect to number of virtual machines is PFEROD, EVMCA, SMOACO, DLBM, EERSCEPO, and HDWO with the values 109.1 kb, 102.5 kb, 85.8 kb, 81.3 kb, 79.2 kb, and 77.1 kb. The comparison graph for Throughput with respect to number of virtual machines is given in Figure 10.

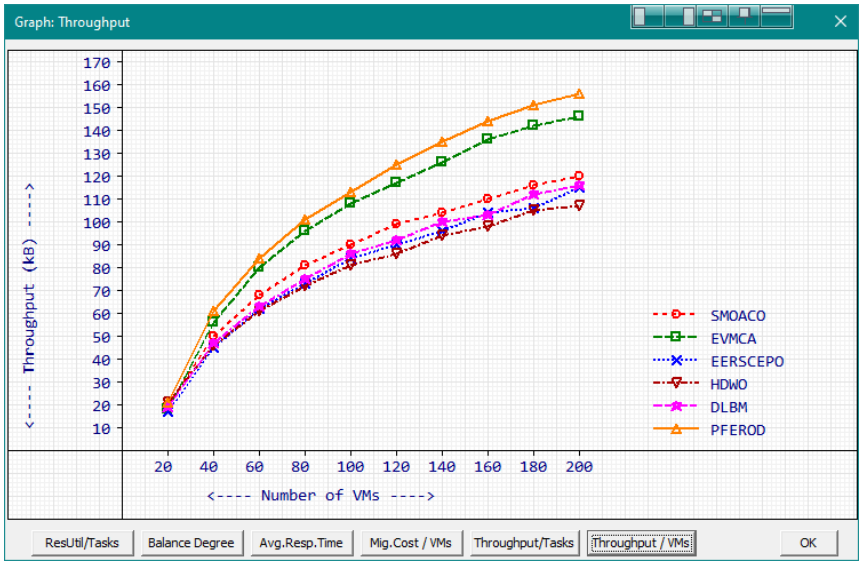

Figure 10: Throughput / Number of Virtual Machines

7. CONCLUSION

Cloud computing environments are susceptible to overload events, which can lead to performance degradation, service disruptions, and customer dissatisfaction. These overloads can arise from unexpected spikes in demand, resource contention, and various other factors. PFEROD is an attempt to apply the advantages of Fussy logic and machine learning methodology to improve the protection against cloud resource overloading. The performance of proposed PFEROD is evidently proved by the live experimental setup using a cloud server. The implementation of proposed PFEROD method shows a meaningful improvement in cloud computing overload protection makes the possibility of applying this method in real-time environments. Thus, the proposed work is humbly submitted here to serve the modern society to achieve an efficient cloud computing eco system.

Conflict of Interest: The authors declare no conflicts of interest related to this work

Dataset details: Real-time data is used to evaluate the proposed model, thus, no dataset is used in this work.

Code accessibility: The source code of the entire implementation is shared in GitHub. The link will be provided by the authors on E-Mail request.

REFERENCES

[1] J. Manner, "A Structured Literature Review Approach to Define Serverless Computing and Function as a Service," 2023 IEEE 16th International Conference on Cloud Computing (CLOUD), Chicago, IL, USA, 2023, pp. 516-522, https://doi.org/10.1109/CLOUD60044.2023.00068

[2] Abraham, A., Yang, J. (2023). A Comparative Analysis of Performance and Usability on Serverless and Server-Based Google Cloud Services. In: Daimi, K., Al Sadoon, A. (eds) Proceedings of the 2023 International Conference on Advances in Computing Research (ACR'23). ACR 2023. Lecture Notes in Networks and Systems, vol 700. Springer, Cham. https://doi.org/10.1007/978-3-031-33743-7_33

[3] Castro, P., Isahagian, V., Muthusamy, V., Slominski, A. (2023). Hybrid Serverless Computing: Opportunities and Challenges. In: Krishnamurthi, R., Kumar, A., Gill, S.S., Buyya, R. (eds) Serverless Computing: Principles and Paradigms. Lecture Notes on Data Engineering and Communications Technologies, vol 162. Springer, Cham. https://doi.org/10.1007/978-3-031-26633-1_3

[4] Saif, M.A.N., Niranjan, S.K., Murshed, B.A.H. et al. Multi-agent QoS-aware autonomic resource provisioning framework for elastic BPM in containerized multi-cloud environment. J Ambient Intell Human Comput 14, 12895–12920 (2023). https://doi.org/10.1007/s12652-022-04120-4

[5] Belen Bermejo, Carlos Juiz,"Improving cloud/edge sustainability through artificial intelligence: A systematic review," in Journal of Parallel and Distributed Computing, Volume 176, 2023, Pages 41-54, ISSN 0743-7315, https://doi.org/10.1016/j.jpdc.2023.02.006

[6] Keliang Du, Luhan Wang, Xiangming Wen, Yu Liu, Haiwen Niu, Shaoxin Huang, "ML-SLD: A message-level stateless design for cloud-native 5G core network," in Digital Communications and Networks, Volume 9, Issue 3, 2023, Pages 743-756, ISSN 2352-8648, https://doi.org/10.1016/j.dcan.2022.04.026

[7] Oztoprak K, Tuncel YK, Butun I. Technological Transformation of Telco Operators towards Seamless IoT Edge-Cloud Continuum. Sensors. 2023; 23(2):1004. https://doi.org/10.3390/s23021004

[8] Marinagi C, Reklitis P, Trivellas P, Sakas D. The Impact of Industry 4.0 Technologies on Key Performance Indicators for a Resilient Supply Chain 4.0. Sustainability. 2023; 15(6):5185. https://doi.org/10.3390/su15065185

[9] Matteo Repetto, "Adaptive monitoring, detection, and response for agile digital service chains," in Computers & Security, Volume 132, 2023, 103343, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2023.103343

[10] Maiyza, A.I., Korany, N.O., Banawan, K. et al. VTGAN: hybrid generative adversarial networks for cloud workload prediction. J Cloud Comp 12, 97 (2023). https://doi.org/10.1186/s13677-023-00473-z

[11] Archana Patil, Rekha Patil, "Proactive and dynamic load balancing model for workload spike detection in cloud," in Measurement: Sensors, Volume 27, 2023, 100799, ISSN 2665-9174, https://doi.org/10.1016/j.measen.2023.100799

[12] K. Ramya, SenthilselviAyothi, "Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment," in Emerging Telecommunications Technologies, Volume 34, Issue 55, March 2023, https://doi.org/10.1002/ett.4760

[13] Yuan L, Wang Z, Sun P, Wei Y. An Efficient Virtual Machine Consolidation Algorithm for Cloud Computing. Entropy. 2023; 25(2):351. https://doi.org/10.3390/e25020351

[14] Amer, D.A., Attiya, G. & Ziedan, I. An efficient multi-objective scheduling algorithm based on spider monkey

and ant colony optimization in cloud computing. Cluster Comput (2023). https://doi.org/10.1007/s10586-023-04018-6

[15] Mansour, R.F., Alhumyani, H., Khalek, S.A. et al. Design of cultural emperor penguin optimizer for energy-efficient resource scheduling in green cloud computing environment. Cluster Comput 26, 575–586 (2023). https://doi.org/10.1007/s10586-022-03608-0

[16] Dimitrios Novas, Dimitrios Papakyriakopoulos, Elizabeth PowleslandKartaloglou, Anastasia Griva, "A ranking model based on user generated content and fuzzy logic," in International Journal of Hospitality Management, Volume 114, 2023, 103561, ISSN 0278-4319, https://doi.org/10.1016/j.ijhm.2023.103561

[17] Xu, Sl., Yeyao, T. & Shabaz, M. Multi-criteria decision making for determining best teaching method using fuzzy analytical hierarchy process. Soft Comput 27, 2795–2807 (2023). https://doi.org/10.1007/s00500-022-07554-2

[18] Cahuantzi, R., Chen, X., Güttel, S. (2023). A Comparison of LSTM and GRU Networks for Learning Symbolic Sequences. In: Arai, K. (eds) Intelligent Computing. SAI 2023. Lecture Notes in Networks and Systems, vol 739. Springer, Cham. https://doi.org/10.1007/978-3-031-37963-5_53

[19] Ahmad O. Aseeri, "Effective RNN-Based Forecasting Methodology Design for Improving Short-Term Power Load Forecasts: Application to Large-Scale Power-Grid Time Series," in Journal of Computational Science, Volume 68, 2023, 101984, ISSN 1877-7503, https://doi.org/10.1016/j.jocs.2023.101984

[20] https://visualstudio.microsoft.com/

[21] https://en.wikipedia.org/wiki/C%2B%2B20

[22] https://www.i2k2.com/