SECURE AND ENERGY-EFFICIENT TASK SCHEDULING IN CLOUD CONTAINER USING VMD-AOA AND ECC-KDF

Muthakshi S[1], Mahesh K[2]
[1]Doctoral Research Scholar, Department of Computer Applications, Alagappa University
Karaikudi, Tamilnadu, India
[e-mail: muthakshi.researchscholar@gmail.com]
[2] Professor and Head, Department of Computational Logistics, Alagappa University, Karaikudi, Tamilnadu, India
[e-mail : mahesh.alagappa@gmail.com]

ABSTRACT

The surge of lightweight containers in cloud storage, propelled by technological advancements, offers dependable services. However, the inherent security concerns arising from shared access to the host Operating System (OS) necessitate an effective solution. This paper introduces an energy-efficient and secure scheduling approach to tackle this issue. Users register task requests with a resource manager, which collects and preprocesses data to eliminate redundancies. The Levenberg-Marquardt Multi-Layer Perceptron Neural Network (LM-MLPNN) optimizes container resource utilization by analyzing user requests. The Homography Transform-based K-Mode Algorithm (HT-KMA) facilitates efficient clustering through attribute extraction. To address imbalances, the Weighted Round Robin (WRR) technique is employed. An optimal container, selected by the Variational Mode Decomposition-based Archimedes Optimization Algorithm (VMD-AOA), undergoes Elliptic Curve-based Key Derivation Function (EC-KDF) for enhanced security before being transmitted to the resource manager. Experimental results demonstrate the superiority of our methodology over existing algorithms.

Keywords: Cloud container; Levenberg-Marquardt Multi-Layer Perceptron Neural Network (LM-MLPNN); Homography Transform-based K-Mode Algorithm (HT-KMA); Variational Mode Decomposition based Archimedes Optimization Algorithm (VMD-AOA); Weighted Round Robin (WRR).

1.0 INTRODUCTION

In various fields, the cloud has emerged as a recent technology due to its flexibility and ease of management. Many service providers, often referred to as tenants, have adopted cloud technology for a majority of their applications, transitioning from traditional local service environments (Kong, Wang, Ma, Chen, et al., 2020). One of the cutting-edge technologies widely employed for running a diverse range of applications is cloud containers. These containers are essentially self-contained software packages that include all the essential elements needed to execute a specific application in various environments, from private data centers to the public cloud, or even on a developer's personal laptop (Kang et al., 2021). These portable and lightweight abstractions encompass all the necessary components, such as files, system libraries, and environment variables, required for running the desired software; these are commonly referred to as containers (Barati et al., 2022). Containers have gained popularity for specific applications because they enable the use of the same OS kernel for different applications, reducing startup times and resource requirements. They find applications in various domains, including Internet of Things (IoT) services, fog computing, smart cars, and service meshes, among others. Some of the well-known containerization tools include Docker, Kubernetes, and OpenVZ (Gholipour et al., 2020).

Portability stands out as a crucial feature of containers, allowing for effortless deployment on various servers once created (Kong, Wang, Ma, Xu, et al., 2020). This attribute offers a significant advantage, as containers can be readily employed to establish environments for development, testing, integration, and production across the software development lifecycle. Nevertheless, container security is a paramount concern and represents a primary hurdle to their widespread adoption (Walkowski et al., 2019). Given that containers share a substantial portion of their computing resources, including the operating system, any intrusion that compromises their integrity may result in the malfunctioning of running applications. Consequently, ensuring container security against malicious users is imperative (Sun et al., 2020). In the networking domain, various approaches have been introduced to enhance the efficient utilization of cloud containers. Deep learning techniques are employed to detect attacks occurring within the container environment (Bhowmik et al., 2020). Furthermore, resource allocation within the container environment, a critical issue for processing a large number of user requests, is addressed to enable effective task execution (Hussein

et al., 2019). Various techniques have been introduced to streamline resource allocation. However, certain limitations persist, as elaborated in section 1.1. In light of these challenges, this paper puts forth a proposal for energy-efficient and secure task scheduling in the cloud container environment. The utilization of cryptographic techniques for container security serves to prevent forgery and breaches that might otherwise compromise the integrity of cloud containers (Banse et al., 2021).

1.1 Problem statement

The prevailing cloud storage landscape has witnessed significant benefits from existing methodologies. However, it is not devoid of certain inherent challenges, including:

- Container Compromises Due to Traffic: The increased network traffic over containers leads to vulnerabilities, such as compromised containers and Distributed Denial of Service (DoS) attacks, resulting in system flooding and performance degradation.
- Malicious User Requests: Some users may submit requests with malicious intent, seeking to maximize their resource utilization on cloud containers, ultimately resulting in resource loss for legitimate users.
- Confidential Data Storage: Ensuring the secure scheduling of user data in highly confidential containers is paramount. However, contemporary challenges arise as the volume of incoming data continues to grow, necessitating innovative solutions for efficient data storage.

To address these issues, our paper focuses on an energy-efficient and secure scheduling approach that leverages advanced techniques to enhance the utilization and security of cloud containers.

1.2 Contributions

To address the identified limitations, this paper introduces a secured cloud container-based framework, which makes the following significant contributions:

- Enhanced Attack Detection and Resource Balancing: Accidental data processing failures or disruptions are mitigated through the utilization of the LM-MLPNN technique, which efficiently detects potential attacks. This, in turn, aids in the effective balancing of resources within the cloud container environment.
- Optimal Container Selection for Task Allocation: The paper introduces a method based on Variational Mode Decomposition with the Archimedes Optimization Algorithm (VMD-AOA) to determine the most suitable container for each task, optimizing throughput and performance.
- Load Balancing and Container Lifecycle Management: The framework incorporates the Homography Transform-based K-Mode Algorithm (HT-KMA) and a load balancing component to distribute workloads across available cluster nodes and oversee the lifecycle of containers. These elements play a crucial role in ensuring efficient resource utilization and container management.

1.3 Organization of the paper

The remaining portion is structured as follows: Section 2 provides an in-depth exploration of the relevant literature, Section 3 presents a comprehensive elucidation of the proposed methodology, Section 4 conducts a comparative analysis of the outcomes and discussions derived from the proposed work in contrast to the pre-existing research, and the paper concludes with a discussion of future avenues for exploration in Section 5.

2.0 LITERATURE REVIEW

(Jin et al., 2021) presented an effective Dynamic Security Evaluation and Optimization of the Moving target defense (DSEOM) technique. In their study, optimization strategies were determined by periodically updating the cloud containers. Additionally, an established specific Multi-Dimensional Attack Graphs (MDAG) approach was used for the effective attack detection process. As a result, the presented approach effectively avoided the problem of effectiveness drift that had occurred in the proactive defense environment. However, in the container environment, the presented approach failed to concentrate on the weak violation problem.

(Shameem Ahamed et al., 2021) demonstrated a methodology grounded in the secure auditing of the Docker container in the cloud environment. Initially, an effective identification and assessment of vulnerabilities in the container images were achieved through the use of a Vulnerability Centric Approach (VCA). To secure the container, a checklist was employed, and it was aligned based on the Container Security Verification Standards (CSVS). Consequently, sensitive information was effectively preserved. However, as critical vulnerabilities could never be entirely avoided by this approach, utilizing the checklist for preserving the security of the container proved to be ineffective.

(Vaishali et al., 2021) introduced the Guided Container Selection (GCS) technique for the effective scheduling of data using neural networks. In their study, ranking was carried out based on user suggestions, and the container was selected using the Deep Neural Network (DNN). This approach effectively mitigated transmission and computational overhead. However, container selection based on user suggestions proved to be ineffective and led to a degradation in performance.

(Karn et al., 2021) explored an Automatic Anomaly Detection in the container using Explainable Machine Learning (AAD-EML) technique. Features were extracted from the continuous matching of the largest syscalls subsequence, resulting in the generation of signatures for the anomalous pods. An effective anomaly detection criterion was achieved by creating boundaries using ML techniques. This approach improved performance by avoiding the problem of inference. However, the major downsides of this approach were convergence instability and longer training time.

(Wang et al., 2022) deployed a Container Guard-based attack detection technique in the cloud environment. In this study, multivariate time-series data was collected from the container in a non-intrusive manner, and the attack patterns were represented normally using an ensemble of Variational Autoencoders (VA). This approach harnessed the generative nature of the VA, resulting in improved performance with efficient detection of meltdown and spectre attacks. Nevertheless, one drawback was the performance's dependence on the window size.

(Coppolino et al., 2021) designed the Virtual Secure Enclave-based Homomorphic Encryption (VSE-HE) methodology for securing data transmission. This approach effectively protected data-in-use from attackers with high privilege. Through the use of the HE approach, data was initially encrypted, transformed into ciphertext, and stored in the cloud for further processing. Consequently, the presented approach ensured the privacy of all users in the environment. However, performing ciphertext conversion after homomorphic encryption rendered the use of, HE unusable.

(Xie et al., 2021) developed an HBRSS-based method for securing the processing and transmission of data in the cloud environment. In this approach, data was initially divided into blocks and organized in a ring format, followed by encryption using the Homomorphic Encryption (HE) technique. To enhance function privacy and prevent flaws and injection attacks, fuzzy processing was employed. As a result, this method effectively minimized the risks of data destruction and tampering. However, the approach's effectiveness was compromised by performing encryption without authentication.

(Liu et al., 2020) employed a protocol-independent container network observability system. Initially, data was collected from a cluster of users, and features were extracted and updated in the controller. The approach involved many containers where protocol matching and filtering occurred. Experimental results demonstrated the efficacy of the approach, primarily through throughput. However, the transparency in data collection impacted the security of data in the cloud container.

(Niu et al., 2020) recommended a Geo-aware Multi-agent Task Allocation (GMTA) framework for securing data in the container environment. Initially, data was divided into numerous subtasks, and an auction was conducted by the agents to allocate tasks to the respective bidders. The winner of the auction was then provided with the respective container for performing the prescribed tasks, resulting in minimized traffic overhead and makespan. However, task allocation based on the auction process impacted the security of the data.

(Singh et al., 2021) presented an efficient ensemble-based resource allocation technique. In this approach, workloads were allocated to multiple containers situated at the edge servers to perform various tasks. The resource allocation process utilized a game-based approach, with Software Defined Network (SDN) serving as a centralized controller in this methodology. The utilization of numerous containers at the edge reduced energy consumption, consequently improving network latency. However, the presented method did not adequately address data security.

In the presented study, a comprehensive overview of contemporary approaches in cloud container security and efficiency was provided. The Table 1 below succinctly captured key methodologies, strengths, and weaknesses from recent literature.

Reference	Technique/Methodology	Main Focus	Pros	Cons
Jin et al., 2021	DSEOM	Dynamic Security Evaluation and Optimization of Moving Target Defense (DSEOM)	Effectively addressed effectiveness drift in proactive defense	Failed to concentrate on the weak violation problem
Shameem Ahamed et al., 2021	VCA, CSVS	Secure auditing of Docker container in the cloud environment	Identified vulnerabilities, aligned with CSVS, preserved sensitive information	Checklist proved ineffective against critical vulnerabilities
Vaishali et al., 2021	GCS, DNN	Guided Container Selection (GCS) for effective scheduling using neural networks	Mitigated transmission and computational overhead	Container selection based on user suggestions led to performance degradation
Karn et al., 2021	AAD-EML	Automatic Anomaly Detection in the container using Explainable Machine Learning (AAD-EML)	Effective anomaly detection, avoided inference problems	Convergence instability, longer training time
Wang et al., 2022	Container Guard, VA	Container Guard-based attack detection technique in the cloud environment	Efficient detection of attacks, leveraging Variational Autoencoders (VA)	Performance dependent on the window size
Coppolino et al., 2021	VSE-HE	Virtual Secure Enclave-based Homomorphic Encryption (VSE-HE)	Secured data-in-use, ensured user privacy	Ciphertext conversion after homomorphic encryption rendered, HE unusable
Xie et al., 2021	HBRSS, HE, Fuzzy Processing	HBRSS-based method for securing data processing and transmission in the cloud environment	Minimized risks of data destruction and tampering	Effectiveness compromised by performing encryption without authentication
Singh et al., 2021	Ensemble-based Resource Allocation	Ensemble-based resource allocation technique	Reduced energy consumption, improved network latency	Did not adequately address data security

Table 1. Cloud Container Security & Efficiency Techniques Overview

2.1 Research Gap

The existing literature review highlights several research gaps that motivate the development of the proposed secured cloud container-based framework:

- The reviewed studies primarily focus on either security or performance optimization in cloud containers. A critical research gap exists in the integration of both aspects to provide a comprehensive solution that balances security and performance effectively.
- The identified challenge of addressing the "weak violation problem" has not been adequately explored in previous research. This presents a significant research gap that this paper aims to bridge.
- The use of user suggestions for container selection, as discussed in the literature review, has shown limitations. There is a need for research that explores alternative, more effective methods for user-centric container selection that enhance performance without compromising security.
- While some papers introduce effective anomaly detection techniques, challenges related to convergence instability and longer training times must be addressed to make these methods practical for real-world applications.
- Papers focusing on resource allocation and network observability should consider the transparency of data collection to avoid compromising data security in cloud containers.

2.2 Motivation of Research

Given the identified research gaps, the motivation for this research is to develop a comprehensive secured cloud container-based framework that addresses the shortcomings identified in the existing literature. This framework aims to make the following contributions:

- Our proposed framework offers an integrated solution that effectively balances security and performance. It leverages LM-MLPNN for efficient attack detection to prevent data processing failures while ensuring optimal resource allocation.
- This research is motivated to provide a solution for the "weak violation problem," a challenge that has been underexplored in prior studies. By doing so, we contribute to a more robust security posture for cloud containers.
- We seek to overcome the limitations of user-centric container selection by introducing alternative methods that improve performance without compromising security.
- Our framework addresses the stability issues associated with anomaly detection, making it more suitable for practical application by mitigating convergence instability and reducing training times.
- We emphasize the importance of data security and transparency, ensuring that the collection and processing of data in cloud containers are conducted in a secure and transparent manner.

By addressing these research gaps and making these contributions, our research aims to provide a holistic solution that enhances the security and performance of cloud containers, offering a practical and balanced approach to address the evolving challenges in this environment.

3.0 PROPOSED ENERGY-EFFICIENT AND SECURED CLOUD CONTAINER METHODOLOGY

Virtualization is a central component in the development of cloud infrastructure and the delivery of cloud services. Within cloud computing systems, a service model that has gained prominence is that of containers, which provides horizontally scalable systems. However, a critical concern in the realm of container security emerges due to the fact that containers on the same physical host share the same operating system. This sharing can lead to security violations and potentially impact other applications running on the same physical host. In response to these challenges, this paper presents a proposal for an energy-efficient and secure task scheduling system tailored for cloud containers. Fig. 1. illustrates the architecture of the proposed system.

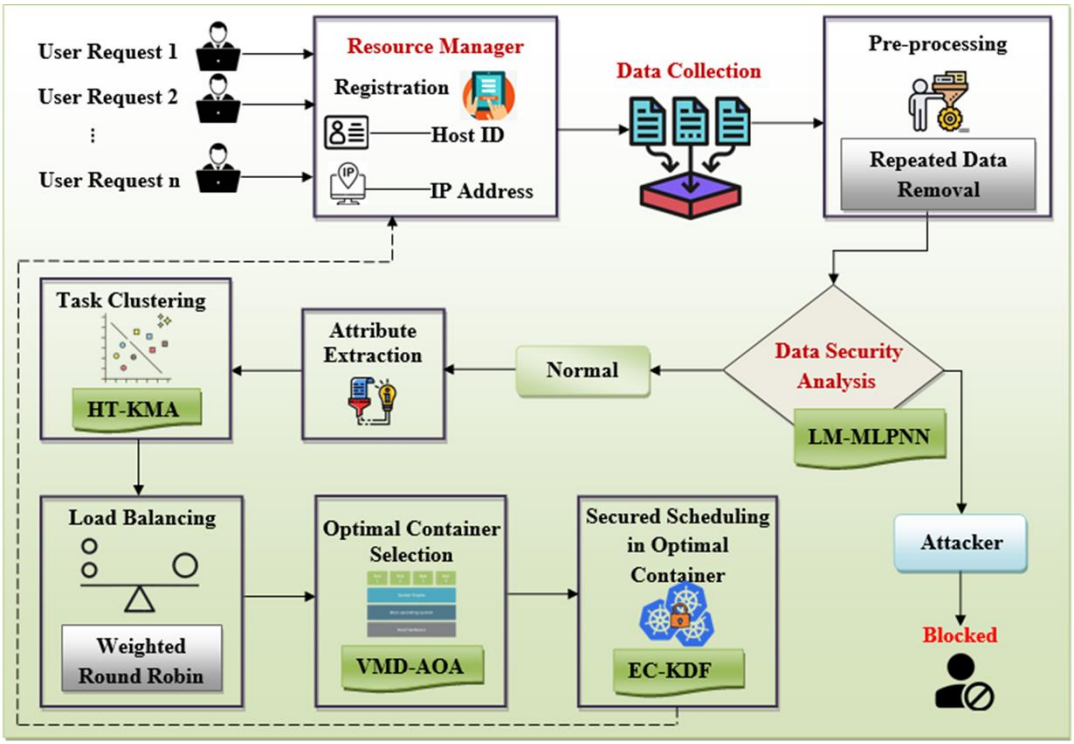

Fig. 1. General structure of the proposed methodology

3.1 Data Collection

Primarily, multiple users send the request to the resource manager $\left(R^m\right)$ requesting the service in the cloud environment and register their details like the host ID $\left(H_{(ID)}\right)$ and IP address $\left(I_{ID}\right)$. Hence, the $n-$number of data collected from $m-$number of users $\left\langle U^M = U^1, U^2, U^3, \ldots, U^m \right\rangle$ is depicted as,

$$D^N = \left\{D^1, D^2, D^3, \ldots, D^n\right\} \tag{1}$$

Here, D^N implies the data collected.

3.2 Pre-processing

Here, to ensure data integrity and reliability, the collected data are pre-processed. One of the crucial pre-processing techniques employed in the proposed methodology is discussed further down.

3.2.1 Repeated data removal

There is a chance of repeated data from the same user requesting the same service since multiple requests reach the resource manager at a time, which might result in the wastage of container resources and time. Thus, to remove the

repeated request from the same user, repeated data removal takes place. Hence, the output obtained after pre-processing $\left(\hat{D}^{N}\right)$ is modeled as,

$$\hat{D}^{N} = \left\{\hat{D}^{1}, \hat{D}^{2}, \hat{D}^{3}, ..., \hat{D}^{R}\right\} \tag{2}$$

Where, $N = \{1,2,3,...,R\}$ implies the pre-processed output.

3.3 Data security analysis using LM-MLPNN

After pre-processing, to secure the container and its resources, the nature of the incoming data is evaluated via the LM-MLPNN technique. A type of feed-forward neural network made of many input layers, hidden layers, and output layers is termed Multi-Layer Perceptron Neural Network (MLPNN). MLPNN is responsible for both forward and backward propagation with weights and bias values in each neuron. Nevertheless, the training time of the network is augmented by the supervised learning nature of MLPNN. Thus, to produce a reduced error rate, Levenberg-Marquardt (LM) is used for effective training. This introduction of LM in MLPNN is called as LM-MLPNN. In Fig. 2, the structure of the proposed LM-MLPNN is cited.

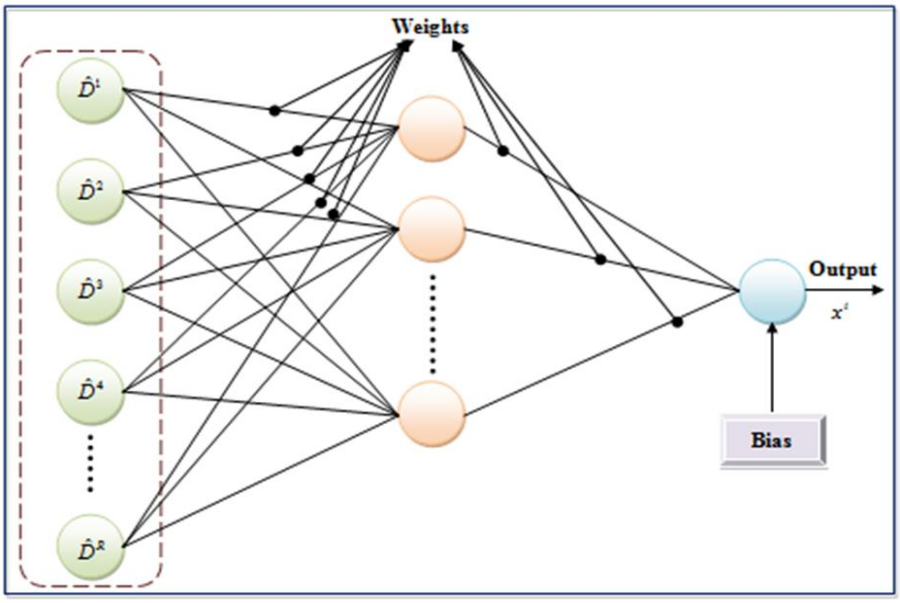

Fig. 2. Baseline structure of proposed LM-MLPNN.

The process involved in the analysis phase is detailed as follows.

LM-based training: Initially, to train the IP address of the users to send the request in reduced time, the proposed LM-MLPNN framework is trained using LM approach. The steps involved in the LM-based training process are explained below,

- For training the feed-forward neural network, LM is the most widely used method, and hence its objective function $\left(Q(z)\right)$ is explained as,

$$Q(z) = \frac{1}{2}\sum_{z=1}^{R} \varepsilon_{z}^{2}(T) \tag{3}$$

Here, $\varepsilon^2(T)$ implies the non-linear error function and is defined in (4), and R refers to the number of collected data requests.

$$\varepsilon_z^2(T) = A(T) - D(T) \tag{4}$$

Where, $A(T)$ and $D(T)$ denotes the actual and the expected error values.

- Then, the weight values in the feed-forward neural network $(\nabla(Q(z)))$ are effectively updated using the following equations.

$$\nabla(Q(z)) = -\left[\nabla^2 Q(z)\right]^{-1} \nabla^2 Q(z) \tag{5}$$

Where, ∇ signifies the gradient vector and is given in (6).

$$\nabla(Q(z)) = \zeta^T(Q(z))\varepsilon_z(T) \tag{6}$$

Here, ζ^T implies the jacobian matrix and is expressed below,

$$\zeta^T = \begin{bmatrix} \frac{\partial \varepsilon_z(1)}{\partial Q_{10}} & \cdot & \frac{\partial \varepsilon_z(1)}{\partial Q_{Ne}} & \cdot & \frac{\partial \varepsilon_z(1)}{\partial Q_{Rn^l}} \\ \frac{\partial \dot{\varepsilon}_z(1)}{\partial Q_{10}} & \cdot & \frac{\partial \dot{\varepsilon}_z(1)}{\partial Q_{Ne}} & \cdot & \frac{\partial \dot{\varepsilon}_z(1)}{\partial Q_{Rn^l}} \\ \frac{\partial \dot{\varepsilon}_z(T)}{\partial Q_{10}} & \cdot & \frac{\partial \dot{\varepsilon}_z(T)}{\partial Q_{Ne}} & \cdot & \frac{\partial \dot{\varepsilon}_z(T)}{\partial Q_{Rn^l}} \end{bmatrix} \tag{7}$$

- Similarly, the hidden layers perform error calculations $\left(\varepsilon_z^{\bar{h}}(T)\right)$ according to (8).

$$\varepsilon_z^{\bar{h}}(T) \cong \sum_{T=1}^{n^l} \chi_e^{n^l}(T) Q(z) \tag{8}$$

Here, $\chi_e^{n^l}$ implies the LM function. Also, all the weights of the neural network are placed inside a single vector $(\Omega(Q(z)))$ and are detailed below.

$$\Omega(Q(z)) = \sum_{z=1}^{R} \sum_{T=1}^{n^l} \varepsilon_z(T) \nabla^2 \varepsilon_z(T) \tag{9}$$

Weight updation is performed only once, particularly at the end of each epoch in the proposed LM-MLPNN. Until the target value is achieved, the process is repeated and the error value is determined with the corresponding weight updation phenomena. Then, the analysis phase is detailed below.

Step 1: Initially, the pre-processed output is applied as input to the input layer of the LM-MLPNN network. Each node performs both summation and activation functions in LM-MLPNN and is connected by means of weight $\left(w^x\right)$ and bias $\left(B^y\right)$ values. Let the input layer output $\left(\Im^g\right)$ is defined as,

$$\Im^g = \hat{D}^1, \hat{D}^2, \ldots\ldots, \hat{D}^R \tag{10}$$

Step 2: Then, the output from the input layer is just forwarded to the hidden layer, where the processing of the user request takes place. Hence, the output of $l-th$ a neuron in the hidden layer $\left(\bar{h}\right)$ is calculated as,

$$\bar{h}_{(l)} = \Gamma\left(\sum_{g=1}^{R} w_{(\bar{h}l)}\Im^{g} + B^{\bar{h}}\right) \tag{11}$$

Here, $w_{(\bar{h}l)}$ implies the weight values of the $l-th$ neuron in the hidden layer $\left(\bar{h}\right)$, $B^{\bar{h}}$ denotes the bias value of the hidden layer, and $\Gamma(\cdot)$ exhibits the ReLU activation function.

Step 3: The vanishing gradient problem that arises in the LM-MLPNN structure is avoided by the ReLU activation function and is mathematically expressed below,

$$\Gamma\left(\tilde{x}\right) = \max\left(0, \tilde{x}\right) \tag{12}$$

Here, $\tilde{x} \in \sum_{g=1}^{R} w_{(\bar{h}l)}\Im^{g} + B^{\bar{h}}$.

Step 4: The output of $t-th$ neuron in the output layer $\left(O_{(t)}\right)$ is distinguished as,

$$O_{(t)} = \Gamma\left(\sum_{j=1}^{M} w_{(to)}\bar{h}_{(t)} + B^{o}\right) \tag{13}$$

Where, t signifies the output layer neurons, and $w_{(to)}, B^{o}$ implies the weight and bias values of the t^{th} neural network.

Step 5: Next, to determine the efficiency of the proposed approach in the attack detection phase, the error values are evaluated. The error value $\left(\hat{\varepsilon}\left(T\right)\right)$ determination is defined as,

$$\hat{\varepsilon}\left(T\right) = \alpha^{a} - O_{(t)} \tag{14}$$

Here, $\alpha^{a}, O_{(t)}$ signifies the actual and desired value. In case, the error value is zero, then the network is effectively trained or else, backpropagation takes place. In the end, the classifier delivers the output as an attacker $\left(\hat{n}_{i}\right)$ or normal user $\left(\hat{A}_{i}\right)$. Then, the attackers are blocked from further processing in the system whereas the normal users continue the following criterion. The pseudocode of the proposed LM-MLPNN is elucidated below.

Pseudocode for the proposed LM-MLPNN

Input: Pre-processed output $\left(\hat{D}^{N}\right)$

Output: Classified output

Begin

Initialize theinput layer, hidden layer, and output layer

For $1 \le N \le R$

// LM based training

Evaluate objective function $\left(Q(z)\right)$

Update $\nabla\left(Q(z)\right) = -\left[\nabla^2 Q(z)\right]^{-1} \nabla^2 Q(z)$

For every N

Perform error calculations $\left(\varepsilon_z^{\bar{h}}(T)\right)$

End for

// LM-MLPNN network

Obtain input layer output $\left(\Im^{g}\right)$

Determine $\bar{h}_{(l)} = \Gamma\left(\sum_{g=1}^{R} w_{(\bar{h}l)} \Im^{g} + B^{\bar{h}}\right)$

Compute $\Gamma\left(\tilde{x}\right) = \max\left(0, \tilde{x}\right)$

Generate output of $t-th$ neuron $\left(O_{(t)}\right)$

End for

Calculate the error value

Return attacker $\left(\hat{n}_i\right)$ or normal user $\left(\hat{A}_i\right)$

End

3.4 Attribute extraction

Then, to aid in the effective task (User request) clustering process, the most important and required attributes are extracted from the normal user. Some of the handcrafted features like task speed, task weight, data size, resource requirement, and bandwidth utilization are extracted and are detailed below.

- *Task speed* $\left(\tau^{s}\right)$*:* It defines the time required to allocate the resources in the container $\left(\tau^{r}\right)$ to the time taken to respond to the request $\left(\tau^{s}\right)$ and is formulated in (15).

$$\tau^{s} = \frac{\tau^{r}}{\tau^{t}} \tag{15}$$

- *Task weight* $\left(\tau^{w}\right)$*:* Task weight refers to the speed and cost of the user request $\left(\tau^{c}\right)$ and is expressed as,

$$\tau^{\omega} = \gamma \times \left(\tau^{s} \times \tau^{c}\right) \tag{16}$$

Where, γ is a constant. Thus, the $k-$ number of attributes extracted $\left(E^{a}\right)$ is modeled as,

$$E^{a} = \left\{E^{1}, E^{2}, E^{3}, \ldots, E^{k}\right\} \tag{17}$$

3.5 Task clustering using HT-KMA

Here, based on the extracted attributes, the tasks are clustered. By scheduling the tasks to a container, this clustering process drastically reduces the system's overhead. The clustering of similar requests is done by the means of HT-KMA in the proposed methodology. A clustering approach that is well suitable for categorical and numerical data is termed the KMA. Cluster centroids are selected randomly in KMA. This random selection results in the local optimum problem. Hence, to select the cluster centers, Homography Transform (HT) is used. This initialization increases the clustering accuracy. Hence, the introduction of HT with KMA is the so-called HT-KMA. The process involved in HT-KMA is detailed further down.

- Let $E^1, E^2, E^3, \ldots, E^k$ be the attributes extracted and N be the number of available cloud containers (cluster centroids) $\left(B^k = B^1, B^2, \ldots\ldots, B^N\right)$. Initially, HT-KMA determines the number of clusters.

- Then, the cluster centroids are evaluated by determining the difference between them to calculate the dissimilarities between the tasks and cluster centroids. The resultant thus obtained is then transformed using the HT equation $\hbar^f$, which is given as follows,

$$\hbar^f = \begin{bmatrix} \hbar_{11} & \hbar_{12} & \hbar_{13} \\ \hbar_{21} & \hbar_{22} & \hbar_{23} \\ \hbar_{31} & \hbar_{32} & 1 \end{bmatrix} \tag{18}$$

  $\left(B^{\hat{k}}\right)$ is the optimized centroid calculated from equation (18).

- Next, to compute the distance between the tasks and optimized cluster centroids, the difference (d) is used. The dissimilarities are estimated below,

$$d\left(E^a, B^{\hat{k}}\right) = \sum_{a=1}^{k} \left(B^{\hat{k}} - E^a\right) \tag{19}$$

- The task is allocated based on the difference to the nearest container (cluster centroids) for efficient transmission of data without any loss and the process continues by recalculating the cluster centroids and calculating the distance between tasks and Cluster centroids. Until the cluster centers are not changed, the process is repeated. In this way, the tasks are grouped together. Hence, the $P-$ number of clusters formed $\left(C^p\right)$is grouped as,

$$C^p = \left\langle C^1, C^2, C^3, \ldots, C^{P1} \right\rangle \tag{20}$$

3.6 Load Balancing by WRR

After the task clustering, to prevent failure caused by overloading a particular container, the imbalance in the task request is balanced. Here, the Weighted Round Robin (WRR) technique is utilized for request balancing, which divides incoming traffic requests evenly across all the workload instances by assigning weight values to each container based on its processing power or the total required bandwidth. Here, the maximum weight value $\left(\xi^m\right)$ is assigned by the container with the maximum processing power $\left(\hat{C}^j\right)$ while the other with half the capacity of $\hat{C}^j$ assigned half

the weight value of ξ^m. But, the container with a capacity lesser than that of the previous one was assigned lower weight values. Hence, the output obtained after load balancing $\left(l^b\right)$ is given as,

$$l^b = \left\{l^1, l^2, l^3, \ldots, l^B\right\} \tag{21}$$

Here, $b = 1,2,3,\ldots,B$ implies the number of balanced outputs.

3.7 Optimal container selection through VMD-AOA

Then, by employing the VMD-AOA algorithm, the most appropriate container to perform the clustered task is selected. Archimedes' optimization algorithm (AOA) is a population-based algorithm. Here, population refers to the immersed objects. The volume, density, and acceleration of various objects are calculated randomly. Nevertheless, the overall optimization performance is affected by the random updating of density and volume factors. Thus, for the updating process, Variational Mode Decomposition (VMD) is used based on the center frequencies and thus increasing the convergence speed. This amalgamation of VMD in conventional AOA is named VMD-AOA and the process is discussed below.

Initialization phase: Let the initial AOA population in D dimensional search space be $\left\{E^1, E^2, E^3, \ldots, E^k\right\}$ (extracted attributes). Primarily, the objects are placed on a fluid surface. Thus, the position of the objects (container) $\left(\varsigma_b\right)$is initialized as below,

$$\varsigma_b = L_b + RND \times \left(U_b - L_b\right); b = 1,2,3,\ldots,B \tag{22}$$

Where, ς_b signifies the b^{th} object, and L_b, U_b implies the lower and upper bound limits of the search space. Also, the volume $\left(Volume\right)$ and density $\left(Density\right)$ of the b^{th} object are also initialized randomly and are given as,

$$\begin{aligned} Density^b &= RND \\ Volume^b &= RND \end{aligned} \tag{23}$$

Where, RND signifies the random values.

Fitness evaluation: After initialization, fitness evaluation occurs. Based on the efficient bandwidth utilization (traffic handling capacity), the container (object) with the best fitness value is selected and hence, the fitness function $\left(f\left(\varsigma_b\right)\right)$ is elucidated as,

$$f\left(\varsigma_b\right) = \sum_{\min}\left(E^a\right) \tag{24}$$

VMD-based updation: Next, when the object is placed in a fluid, it begins to immerse in the fluid. Now, an upward force pulls it toward the outside of the surface, which is detailed by means of VMD as follows.

$$\left\{MIN_{\left\{f^a\right\}\left\{BW^a\right\}}\left[\sum_k \left\| \partial_t \left(\left\langle \lambda(t) + \frac{b}{\pi t} \right\rangle * f^a(t) \right) EXP\left(-bBW^{at}\right) \right\|_2^2\right] = E^a(t)\right. \tag{25}$$

Here, f^a signifies the center frequencies, BW^a implies the bandwidth utilized, $\lambda(t)$ depicts the unit impulse function, and t signifies the time. The updated density $\left(density^*\right)$is given as,

$$density^* = e^{\left(\frac{E^a(t)-t}{E^a(t)}\right)-\left(\frac{t}{E^a(t)}\right)} \tag{26}$$

Exploration phase: Then, because of the upward force in the fluid, the objects begin to collide with each other. Hence, the updated acceleration $\left(acc^{*}\right)$ is modeled as,

$$acc^{*} = \frac{Density^{b} + Volume^{b} \times acc^{b}}{density^{*} \times volume^{*}} \tag{27}$$

Where, acc^{b} implies the acceleration of the b^{th} object.

Exploitation phase: If there is no collision arises between the objects, the acceleration is updated as,

$$acc^{*} = \frac{Density^{best} + Volume^{best} \times acc^{best}}{density^{*} \times volume^{*}} \tag{28}$$

Here, $Density^{best}, Volume^{best}, acc^{best}$ signifies the density, volume, and acceleration of the best object obtained through the fitness determination phase.

Acceleration normalization: The acceleration is normalized $\left(acc^{*}_{b-NORM}\right)$to determine the change rate, and is explained below.

$$acc^{*}_{b-NORM} = s \times \frac{acc^{*} - \min\left(acc\right)}{\max\left(acc\right) - \min\left(acc\right)} + \wp \tag{29}$$

Here, $s, \wp$ implies the normalization range, and acc signifies the acceleration of the initial objects, and is given in (30).

$$acc = g + RND \times \left(\aleph - g\right) \tag{30}$$

Where, $g, \aleph$ signifies the upper and lower bound limits. In the end, the global optimal solution (cloud container) is obtained, which is denoted as ϑ^{c} and is formulated below.

$$\vartheta^{c} = \left\{\vartheta^{1}, \vartheta^{2}, \vartheta^{3}, \ldots, \vartheta^{C}\right\} \tag{31}$$

Where, $c = 1,2,3,\ldots,C$ implies the $C-$number of optimal containers selected. The pseudocode of the VMD-AOA is given below.

Pseudocode for the proposed VMD-AOA

Input: Extracted attributes $\left(E^{a}\right)$

Output: Optimal container $\left(\vartheta^{c}\right)$

Begin

Initialize AOA population, maximum iteration $\left(Iter_{\max}\right)$

Initializethe volume $\left(Volume\right)$, density $\left(Density\right)$, acceleration $\left(acc\right)$

For $1 \le a \le k$

Evaluate fitness $\left(f\left(\varsigma_{b}\right)\right)$

Determine

$$\left\{MIN_{\left\{f^{a}\right\}\left\{BW^{a}\right\}}\left[\sum_{k}\left\|\partial_{t}\left(\left\langle\lambda(t)+\frac{b}{\pi t}\right\rangle * f^{a}(t)\right)EXP\left(-bBW^{at}\right)\right\|_{2}^{2}\right]=E^{a}(t)\right.$$

Set $iter = 1$

While $iter \le iter_{\max}$ **do**

Update density $density^{*} = e^{\left(\frac{E^{a}(t)-t}{E^{a}(t)}\right)-\left(\frac{t}{E^{a}(t)}\right)}$

If acollision occurs**then**

Update acceleration $\left(acc^{*}\right)$

Elseno collision

Update $acc^{*} = \dfrac{Density^{best} + Volume^{best} \times acc^{best}}{density^{*} \times volume^{*}}$

End if

Compute normalized acceleration

End while

Return optimal container $\left(\vartheta^{c}\right)$

End for

End

3.8 Secured Scheduling in the optimal container using EC-KDF

It is necessary to protect the task from being attacked by establishing a suitable security service since there is a lot of security issue in the container. Multi-application synchronization attacks, Dos attacks, and hyper jacking are frequent attacks in the container. A security model named EC-KDF is deployed to prevent the container from these attacks. The most efficiently utilized cryptographic technique for security is Elliptic Curve Integrated Encryption Scheme (ECIES). Here, the keys are generated using Elliptic Curve Cryptography (ECC) followed by the generation of the Message Authentication Code (MAC) using the random number. The security level of ECIES affected this utilization of the random number. For generating a secret key, an efficient Key Derivation Function (KDF) is adapted, so that it can be utilized in the MAC generation procedure. This induction of KDF in the traditional ECIES is named EC-KDF and the process is described further.

- Let ρ^{pub} and $\widehat{\rho}^{priv}$ be the public and private key generated by the ECC approach. The public key $\left(\rho^{pub}\right)$ is computed using (32).

$$\rho^{pub} = \widehat{\rho}^{priv} \times \overline{u} \tag{32}$$

  Where, $\overline{u}$ implies the point on the elliptic curve.

- Then, by employing KDF, a secret key is generated. Here, the obtained secret key $\left(F^{\text{sec}}\right)$ is the combination of ρ^{pub} and $\widehat{\rho}^{priv}$. It is modeled as,

$$F^{\text{sec}} = \rho^{pub} + \rho^{priv} \tag{33}$$

  Then, a MAC $\left(\overline{\overline{m}}\right)$is generated by employing the obtained secret key $\left(F^{\text{sec}}\right)$using the following expression.

$$\overline{\overline{m}} = F^{\text{sec}} \times \left(\vartheta^{c}, I_{ID}\right) \tag{34}$$

- Hence, the MAC is transmitted to the resource manager where decryption takes place for allocating the optimal container to the respective user. The secret key is shared since the user is classified as normal and hence it is utilized for successful decryption and utilization of the optimal container. The mathematical formulation for the decryption operation is given as.

$$\left(\overline{\overline{m}}\right)^{*} = F^{\text{sec}} / \left(\vartheta^{c}, I_{ID}\right) \tag{35}$$

  Hence, the tasks are securely scheduled in the optimal container.

4 RESULTS AND DISCUSSION

Here, the proposed methodology's performance is analogized to the prevailing techniques grounded on a few performance metrics. The work is implemented in the working platform of PYTHON.

4.1 Database Description

In the proposed methodology, the NSL-KDD dataset is used for security analysis. Frequent biasing is avoided since it has no redundant records. The number of details present in the dataset is affordable to have experimented with in various critical environments. It contains details regarding both attacked and normal characteristics.

4.2 Performance Measurement of proposed EC-KDF

Here, the proposed EC-KDF's performance is weighed against the prevailing Elliptical Curve Cryptography (ECC), Rivest-Shamir-Adleman (RSA), ElGamal, and Diffie-Hellman (DH).

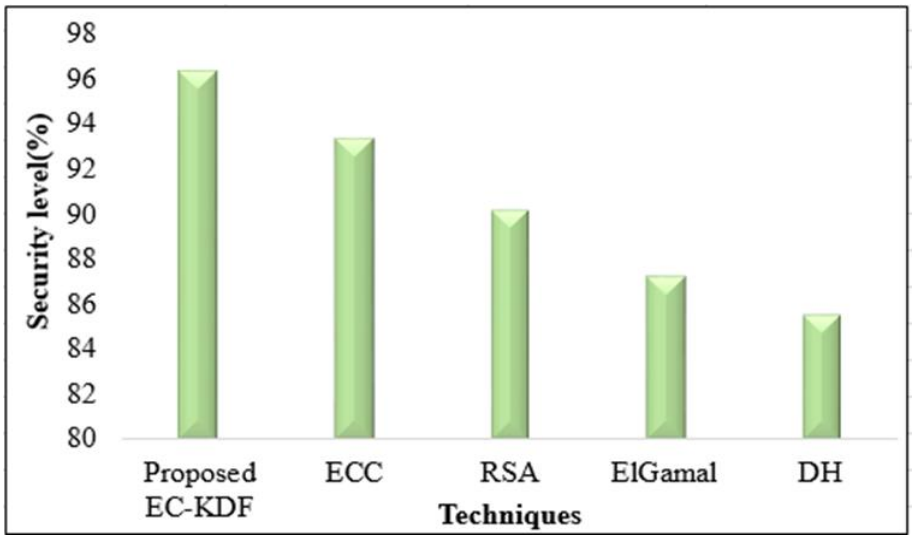

Fig. 3. Performance measurement based on the security level.

The graphical representation of the proposed EC-KDF is depicted in Fig. 3. It is appropriate that the security level of the proposed model increases at about 96.34% but the existing techniques such as ECC (93.29%), RSA (90.78%), ElGamal (87.19%), and DH (85.49%) are low. The generation of MAC using the secret key generated via the KDF function improved the performance of the proposed system, which is superior to the conventional techniques.

4.3 Performance analysis of proposed VMD-AOA

Here, the proposed VMD-AOA's superiority is evaluated and is compared with the existing Archimedes Optimization Algorithm(AOA), Rain optimization algorithm (ROA), Chimp Optimization Algorithm (ChOA), and Coyote Optimization Algorithm (COA).

Table 2. Comparative analysis of proposed VMD-AOA on the basis of energy consumption.

Number of tasks	Energy Consumption rate(mJ)				
	Proposed VMD-AOA	**AOA**	**ROA**	**ChOA**	**COA**
100	2315	3957	5821	7984	10470
200	4532	5680	6998	9754	12471
300	8521	10742	12472	14875	16987
400	10247	13589	15823	17843	19658
500	12354	15784	17894	19875	21578

The energy consumption rates of the proposed VMD-AOA regarding various tasks are delineated in Table 2. The optimal container selection process is accomplished by the VMD-AOA with less consumption of energy. The proposed VMD-AOA method consumes 2315 mJ, 4532 mJ, 8521 mJ, 10247 mJ, and 12354 mJ for 100, 200, 300,

400, and 500 tasks, respectively; while the existing works like ROA consume a large amount of energy varying between 5821 mJ-17894mJ. Likewise, the energy consumption rate varies for other methods also. Thus, in the proposed methodology, the usage of the variational mode decomposition approach reduced the unwanted artifacts with the corresponding reduction in the energy consumption rate. So, the proposed VMD-AOA executes the tasks with limited energy.

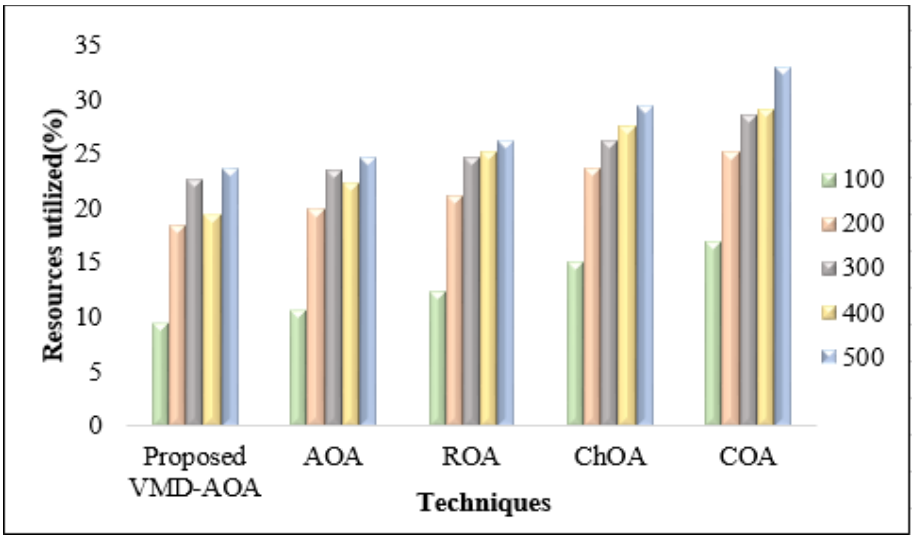

Fig. 4. Superiority analysis of the proposed VMD-AOA in terms of resource utilization

The number of resources utilized by the proposed VMD-AOA with the existing techniques is elucidated in Fig. 4. for performing 100, 200, 300, 400, and 500 tasks, the VMD-AOA utilizes resources up to 9.4545%, 18.3587%, 22.6448%, 19.4578%, and 23.5478%; while, the prevailing AOA, ROA, ChOA, and COA utilize the resources of the order of 10.6780%, 12.3242%, 14.9876%, and 16.8321% for 100 tasks. Similarly, the amount of resources utilized varies for various tasks but is not lower than the proposed approach. Thus, the VMD-AOA properly uses the resources, thereby the over-utilization and under-utilization problems are avoided in the proposed work.

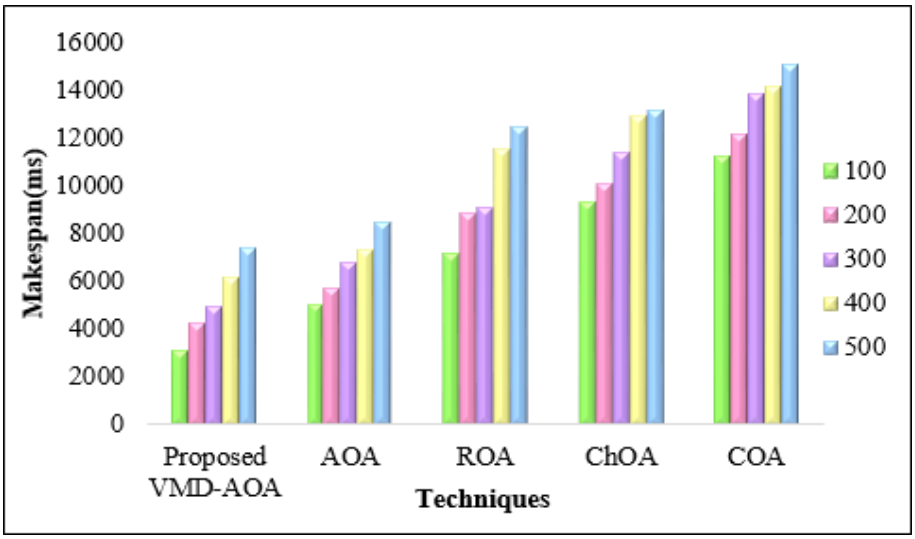

Fig. 5. Performance analysis of the proposed VMD-AOA by means of makespan.

The makespan of the proposed VMD-AOA with the other existing works is depicted in Fig. 5. To complete 100 tasks, the proposed technique requires an average of 3100ms, whereas the existing techniques need 5050 ms (AOA),

7230(ROA), 9380(ChOA), and 11250(COA). Similarly, the makespan varies as the number of tasks varies. Hence, the proposed VMD-AOA executes the tasks more faster.

Table 3. Degree of imbalance measure of proposed VMD-AOA

Number of tasks	Degree of imbalance				
	Proposed VMD-AOA	**AOA**	**ROA**	**ChOA**	**COA**
100	0.1489	0.1786	0.1973	0.2047	0.2287
200	0.1477	0.1756	0.1954	0.2057	0.2278
300	0.1454	0.1732	0.1945	0.2038	0.2264
400	0.1428	0.1725	0.1932	0.2021	0.2251
500	0.1405	0.1711	0.1920	0.1918	0.2201

The proposed approach's degree of imbalance is given in Table 3. For the 100, 200, and 300 tasks, the VMD-AOA depicted 0.1489, 0.1477, and 0.1454 for the degree of imbalance. For 100,200,300,400, and 500 tasks, the degree of imbalance decreased at the rate of 0.1786-0.1711 for AOA, 0.1973-0.1920 for ROA, 0.2047-0.1918 for ChOA, and 0.2287-0.2201 for COA, respectively. Thus, when analogized to the prevailing methodologies, the VMD-AOA performance is better.

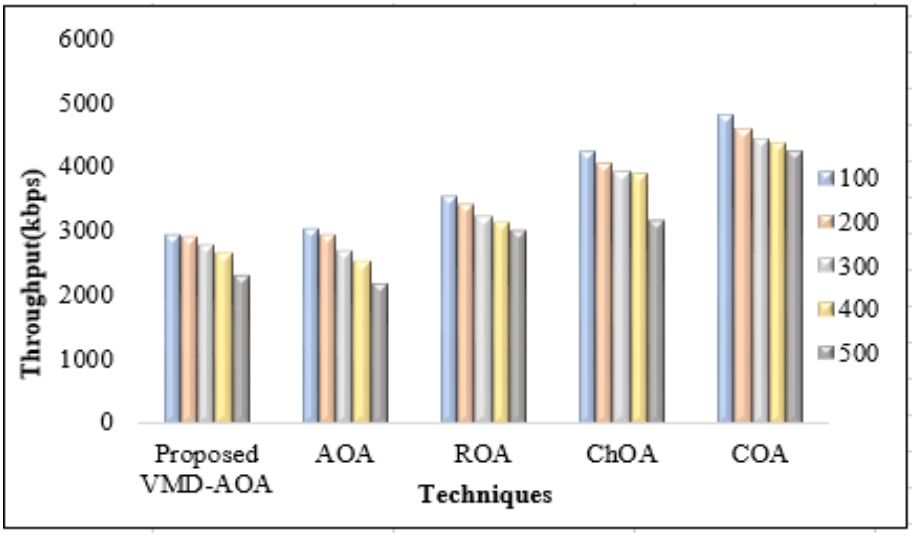

Fig. 6. Performance evaluation of proposed VMD-AOA in terms of throughput

The throughput of the proposed VMD-AOA approach is explained in figure 6. For throughput, the VMD-AOA attained 2945kbps (100); while, the existing methods such as AOA (2892kbps), ROA (2784kbps), ChOA (2656kbps), and COA (2290kbps) which are lower for 100 number of tasks. Likewise, when analogized to the prevailing

methodologies, the VMD-AOA's throughput is higher for the remaining number of tasks also. Hence, the VMD-AOA performs well and achieves a better result.

4.4 Superiority measure of proposed HT-KMA

The proposed HT-KMA's performance is analogized to the prevailing K-Means Algorithm (KMA), Fuzzy-C means clustering (FCM), K-Nearest Neighbor (KNN), and Mean Shift (MS).

Table 4. The superiority measure of the proposed HT-KMA on the basis of clustering accuracy.

Number of clusters	Clustering accuracy (%)				
	Proposed HT-KMA	**KMA**	**FCM**	**KNN**	**MS**
2	92.354	86.123	83.897	82.896	81.024
3	88.378	85.258	82.925	81.925	80.147
4	86.923	83.369	81.036	79.036	78.258
5	85.893	82.470	80.147	76.147	76.369
6	83.485	80.236	78.244	75.924	75.876

The clustering accuracy of the proposed HT-KMA is depicted in Table 4. For clustering accuracy, the HT-KMA obtains a maximum order of 92.354% (2), 88.378%(3), 86.923%(4), 85.893%(5), and 83.485%(6), respectively; while the KMA, FCM, KNN, and MS for the 2 clusters are 86.123%, 83.897%, 82.896%, and 81.024%, which are comparatively low. The clustering accuracy varies as the cluster size varies but is not higher than the HT-KMA. Thus, the usage of the homography transforms in the KMA technique avoided the local optimal problem and resulted in higher clustering accuracy.

4.5 Performance analysis of proposed LM-MLPNN

Here, the proposed LM-MLPNN is compared with existing works like MLPNN, Deep Belief Networks (DBN), Deep Neural Network (DNN), and Convolution Neural Networks (CNN).

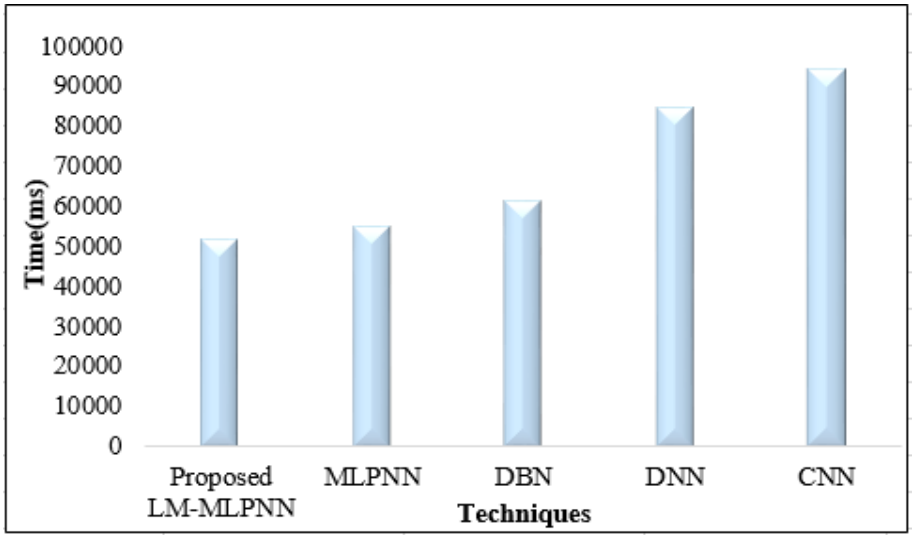

Fig. 7. Performance analysis of the proposed LM-MLPNN with respect to training time

The training time taken by the proposed LM-MLPNN technique is depicted in Fig. 7. to complete the training process, the proposed LM-MLPNN classifier takes 51774 ms; while the existing technique like MLPNN, DBN, DNN, and CNN requires 54987 ms, 61774 ms, 84774 ms, and 94774 ms, respectively. Thus, when weighed against the prevailing methodologies, the LM-MLPNN completes the entire training process with less consumption time.

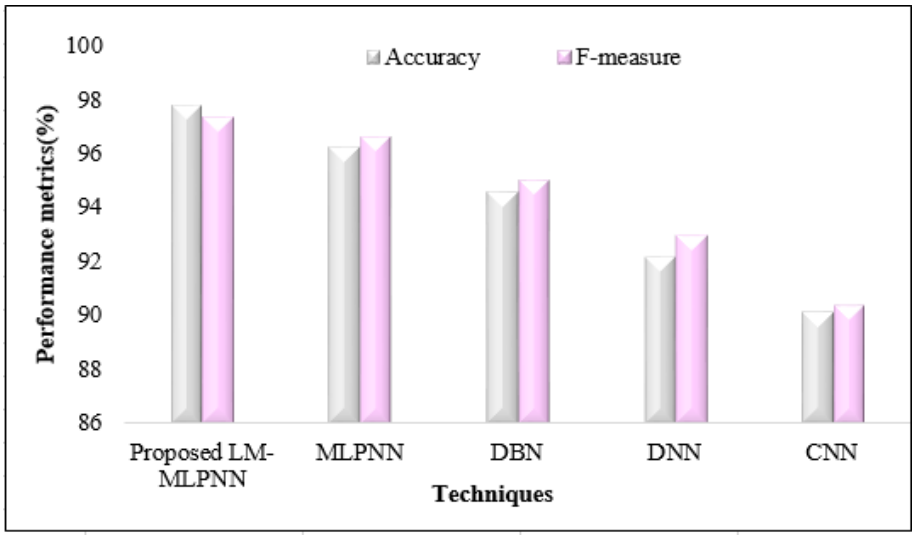

Fig. 8. Performance analysis of the proposed LM-MLPNN.

The performance analysis of the proposed technique regarding the accuracy and F-measure is exhibited in Fig. 8. The proposed technique achieves higher metrics rates, such as 97.74% of accuracy, and 97.34% of F-measure. But the existing works obtain an accuracy rate those overall ranges between 90.14%-96.24%, and F-measure rates that overall range between 90.35%-96.57%. Thus, the proposed LM-MLPNN efficiently analyses the nature of the user more efficiently.

4.6 Comparative analysis with literature papers

Here, regarding efficiency, the proposed approach's performance is weighed against the prevailing DSEOM (Jin et al., 2021), VCA (Shameem Ahamed et al., 2021), AAD-EML (Karn et al., 2021), and VSE-HE (Coppolino et al., 2021).

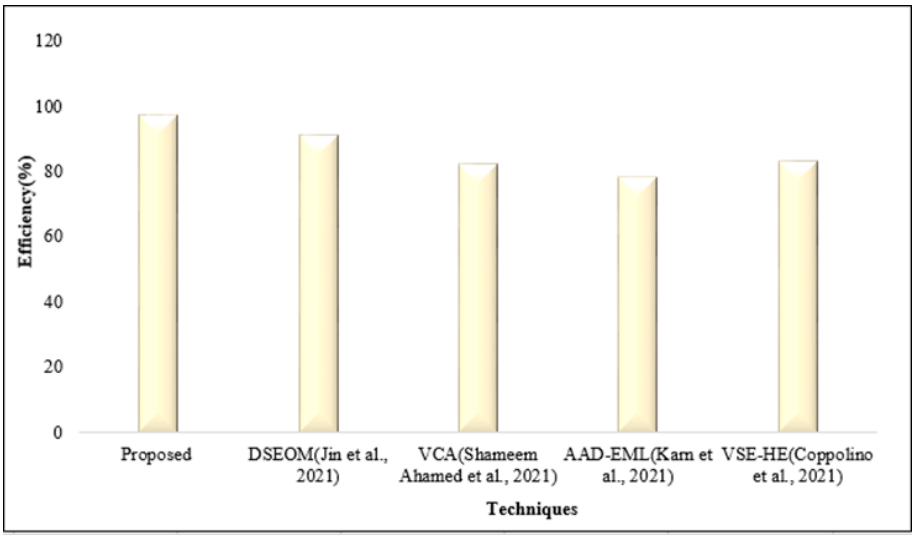

Fig. 9. Comparative measure of the proposed approach.

The superiority measure of the proposed methodology regarding its efficiency is depicted in Fig. 9. When analogized to the proposed technique, the efficiency of the existing techniques is lower. The inclusion of the security analysis using the LM-MLPNN and container security using the EC-KDF approach phase improved the efficiency of the proposed methodology to 97%; while the conventional methodologies have only 82% (VCA), 78% (AAD-EML), and 83% (VSE-HE) efficiency level. Hence, the proposed approach is superior to traditional methods.

5.0 CONCLUSION

Here, an energy-efficient and secure task scheduling method for cloud containers is proposed, employing VMD-AOA, LM-MLPNN, and the EC-KDF technique. The proposed methodology involves several steps, including registration, pre-processing, data security analysis, attribute extraction, task clustering, optimal container selection, and secure request transmission in the chosen container. Subsequently, experimental analysis is conducted using publicly available datasets. Performance metrics are employed to assess and compare the effectiveness of the proposed technique. The proposed method achieves a clustering accuracy of 92.354% for 2 clusters with reduced training time (51774ms), utilizing minimal resources (9.4545), and achieving a throughput of 2945 for 100 tasks. The superiority measure varies for varying numbers of tasks. Overall, the proposed energy-efficient and secure scheduling framework outperforms existing state-of-the-art methods, demonstrating increased reliability and robustness. However, it's important to note that while the proposed methodology secures the OS of the cloud container, the open nature of the kernel may still impact the secure scheduling process. Therefore, future work will aim to enhance security using advanced techniques.

ACKNOWLEDGEMENT

This research work has been supported by UGC NET Senior Research Fellowship, University Grants Commission, New Delhi, India.

REFERENCES

[1] Banse, C., Kunz, I., Schneider, A., & Weiss, K. (2021). Cloud Property Graph: Connecting Cloud Security Assessments with Static Code Analysis. *IEEE International Conference on Cloud Computing, CLOUD*, *2021-September*, 13–19. https://doi.org/10.1109/CLOUD53861.2021.00014

[2] Barati, M., Aujla, G. S., Llanos, J. T., Duodu, K. A., Rana, O. F., Carr, M., & Ranjan, R. (2022). Privacy-Aware Cloud Auditing for GDPR Compliance Verification in Online Healthcare. *IEEE Transactions on Industrial Informatics*, *18*(7), 4808–4819. https://doi.org/10.1109/TII.2021.3100152

[3] Bhowmik, S., Saira Bhanu, S. M., & Rajendran, B. (2020). Container Based On-Premises Cloud Security Framework. *Proceedings of the 5th International Conference on Inventive Computation Technologies, ICICT 2020*, 773–778. https://doi.org/10.1109/ICICT48043.2020.9112561

[4] Coppolino, L., D'Antonio, S., Formicola, V., Mazzeo, G., & Romano, L. (2021). VISE: Combining Intel SGX and Homomorphic Encryption for Cloud Industrial Control Systems. *IEEE Transactions on Computers*, *70*(5), 711–724. https://doi.org/10.1109/TC.2020.2995638

[5] Gholipour, N., Arianyan, E., & Buyya, R. (2020). A novel energy-aware resource management technique using joint VM and container consolidation approach for green computing in cloud data centers. *Simulation Modelling Practice and Theory*, *104*, 102127. https://doi.org/10.1016/j.simpat.2020.102127

[6] Hussein, M. K., Mousa, M. H., & Alqarni, M. A. (2019). A placement architecture for a container as a service (CaaS) in a cloud environment. *Journal of Cloud Computing*, *8*(1), 1–15. https://doi.org/10.1186/s13677-019-0131-1

[7] Jin, H., Li, Z., Zou, D., & Yuan, B. (2021). DSEOM: A Framework for Dynamic Security Evaluation and Optimization of MTD in Container-Based Cloud. *IEEE Transactions on Dependable and Secure Computing*, *18*(3), 1125–1136. https://doi.org/10.1109/TDSC.2019.2916666

[8] Kang, H., Kim, J., & Shin, S. (2021). MiniCon: Automatic enforcement of a minimal capability set for security-enhanced containers. *2021 IEEE International IOT, Electronics and Mechatronics Conference, IEMTRONICS 2021 - Proceedings*, *Vm*. https://doi.org/10.1109/IEMTRONICS52119.2021.9422529

[9] Karn, R. R., Kudva, P., Huang, H., Suneja, S., & Elfadel, I. M. (2021). Cryptomining Detection in Container Clouds Using System Calls and Explainable Machine Learning. *IEEE Transactions on Parallel and Distributed Systems*, *32*(3), 674–691. https://doi.org/10.1109/TPDS.2020.3029088

[10] Kong, T., Wang, L., Ma, D., Chen, K., Xu, Z., & Lu, Y. (2020). ConfigRand: A Moving Target Defense Framework against the Shared Kernel Information Leakages for Container-based Cloud. *Proceedings - 2020 IEEE 22nd International Conference on High Performance Computing and Communications, IEEE 18th International Conference on Smart City and IEEE 6th International Conference on Data Science and Systems, HPCC-SmartCity-DSS 2020*, 794–801. https://doi.org/10.1109/HPCC-SmartCity-DSS50907.2020.00104

[11] Kong, T., Wang, L., Ma, D., Xu, Z., Yang, Q., Lu, Z., & Lu, Y. (2020). Automated Honeynet Deployment Strategy for Active Defense in Container-based Cloud. *Proceedings - 2020 IEEE 22nd International Conference on High Performance Computing and Communications, IEEE 18th International Conference on Smart City and IEEE 6th International Conference on Data Science and Systems, HPCC-SmartCity-DSS 2020*, 483–490. https://doi.org/10.1109/HPCC-SmartCity-DSS50907.2020.00059

[12] Liu, C., Cai, Z., Wang, B., Tang, Z., & Liu, J. (2020). A protocol-independent container network observability analysis system based on eBPF. *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, *2020-December*, 697–702. https://doi.org/10.1109/ICPADS51040.2020.00099

[13] Niu, M., Cheng, B., Feng, Y., & Chen, J. (2020). GMTA: A Geo-Aware Multi-Agent Task Allocation Approach

for Scientific Workflows in Container-Based Cloud. *IEEE Transactions on Network and Service Management*, *17*(3), 1568–1581. https://doi.org/10.1109/TNSM.2020.2996304

[14] Shameem Ahamed, W. S., Zavarsky, P., & Swar, B. (2021). Security Audit of Docker Container Images in Cloud Architecture. *ICSCCC 2021 - International Conference on Secure Cyber Computing and Communications*, *April*, 202–207. https://doi.org/10.1109/ICSCCC51823.2021.9478100

[15] Singh, A., Aujla, G. S., & Bali, R. S. (2021). Container-based load balancing for energy efficiency in software-defined edge computing environment. *Sustainable Computing: Informatics and Systems*, *30*, 100463. https://doi.org/10.1016/j.suscom.2020.100463

[16] Sun, J., Wu, C., & Ye, J. (2020). Blockchain-based Automated Container Cloud Security Enhancement System. *Proceedings - 2020 IEEE International Conference on Smart Cloud, SmartCloud 2020*, 1–6. https://doi.org/10.1109/SmartCloud49737.2020.00010

[17] Vaishali, K. R., Rammohan, S. R., Natrayan, L., Usha, D., & Niveditha, V. R. (2021). Guided container selection for data streaming through neural learning in cloud. *International Journal of Systems Assurance Engineering and Management*. https://doi.org/10.1007/s13198-021-01124-9

[18] Walkowski, M., Biskup, M., Szewczyk, A., Oko, J., & Sujecki, S. (2019). Container based analysis tool for vulnerability prioritization in cyber security systems. *International Conference on Transparent Optical Networks*, *2019-July*, 1–4. https://doi.org/10.1109/ICTON.2019.8840441

[19] Wang, Y., Wang, Q., Chen, X., Chen, D., Fang, X., Yin, M., & Zhang, N. (2022). ContainerGuard: A Real-Time Attack Detection System in Container-Based Big Data Platform. *IEEE Transactions on Industrial Informatics*, *18*(5), 3327–3336. https://doi.org/10.1109/TII.2020.3047416

[20] Xie, H., Zhang, Z., Zhang, Q., Wei, S., & Hu, C. (2021). HBRSS: Providing high-secure data communication and manipulation in insecure cloud environments. *Computer Communications*, *174*(November 2020), 1–12. https://doi.org/10.1016/j.comcom.2021.03.018