International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2024/51 Volume 11, Issue 6, November – December (2024)

RESEARCH ARTICLE

Implementation and Analysis of Fog Node-Assisted Scheduling and Optimization of Resource Allocation and Utilization

Neha Sharma
Department of Computer Science and Engineering, Chandigarh University, Mohali, Punjab, India.
sharma876neha@gmail.com

Deepti Sharma
Department of Computer Science and Engineering, Chandigarh University, Mohali, Punjab, India.
deepti.e14308@cumail.in

Received: 02 September 2024 / Revised: 29 November 2024 / Accepted: 08 December 2024 / Published: 30 December 2024

Abstract – The Internet of Things (IoT) has recently become popular for collecting and storing data in third-party datasets. When combined with IoT devices, fog computing (FC) efficiently manages large data volumes and processing demands. However, concerns persist regarding privacy, edge node latency, data security, and energy consumption. With the increasing automation in smart cities, the workload for fog nodes (FNs) is developing, and additional FNs are needed. The optimal allocation of resources is essential in addressing the resource allocation (RA) issues in executing IoT applications within FC. To tackle this, the mixed integer linear Ant Lion optimizer (MILALO) model has been deployed to optimize resource allocation, reduce execution time, and conserve energy in fog computing. The proposed model overcomes challenges by optimizing resource allocation, reducing execution time, and conserving energy in fog computing. It targets efficient resource utilization and enhances scheduling, optimization, and cloud resource management to improve overall time and energy consumption. This model mediates between the network and users to process and present results by constructing an allocation matrix for the allocator. Simulations confirm the effectiveness of the MILALO model, with demonstrated 20-25% cloud optimization improvement and 50-60% reduction in time and energy consumption. It conducts a thorough assessment of the proposed model's effectiveness through key performance indicators such as execution time (ET), energy consumption (EC), and resource utilization (RU). Finally, a detailed comparative analysis against established techniques enriches the discussion, providing valuable insights into the superiority of the proposed technique.

Index Terms – Internet of Things (IoT), Fog Computing (FC), Resource Allocation (RA), Execution Time (ET), Energy Consumption (EC), Mixed integer linear Ant Lion optimizer (MILALO).

1. INTRODUCTION

The development of smart cities has become an important feature of the new society. Billions of new IoT devices are used daily for everything from industrial production to personal use [1], [2].

As per statistics, Figure 1 shows the quantity of IoT-associated devices in the past and next year's [3]. Nowadays, cloud computing offers processing and storage capabilities for IoT environments. However, cloud computing demands maximum latency because it is far from the end user. Furthermore, it takes time for the cloud to evaluate the data formed by IoT devices [4]. The quantity of data generated and the number of IoT devices will rise. The large quantity of data collected from the distinct devices needs to be transferred with minimal latency. Fog computing (FC) arose as a solution to this problem. It identifies issues of excessive data transfer latency by acting as an intermediate layer (IL) between cloud and IoT devices. In IoT environments, many sensors transmit data through fog nodes (FNs) instead of straight to the cloud to fulfill the high processing demand [5]. It is present in several systems, including smart grids and smart cities between the cloud and smart devices [6], [7]. This additional layer, or FC, increases the attack surface that can be compromised by threats like breaches, and data loss, which can result in the introduction of new vulnerabilities. Furthermore, FC environments gave rise to several risks, including malicious FNs, insider threats, and denial-of-service (DoS) attacks. For example, in FC situations, attackers may seek infinite processing and store data in fog devices (FDs) that prevent customers from accessing the FDs [8].

The number of active connections increases consistently each year, reflecting significant technological advancements and

International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2024/51 Volume 11, Issue 6, November – December (2024)

adoptions [9]. This rapid growth of IoT devices generates a lot of information. A significant segment of this information is big data (BD), which needs to be processed by an extremely strong computer system [10].

Moreover, several devices require real-time facilities and maximum precision in decision-making, producing a direct effect on the data center (DC) and Internet, involving the cloud. Data, computers, software, and infrastructure are all made available via the cloud, which also offers security, flexibility, and dependability [11]. Cloud computing does, however, have several challenges, such as request-reply latency in real-time services, network congestion (NC), etc.

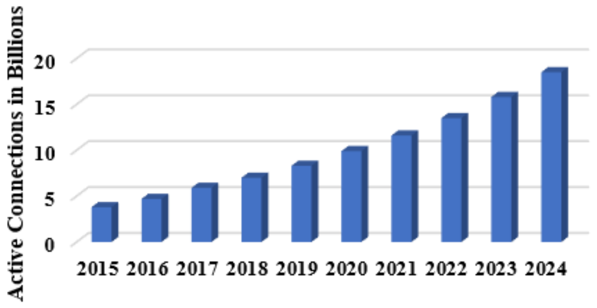

Figure 1 Growing Number of IoT Devices [9]

1.1. Fog Computing

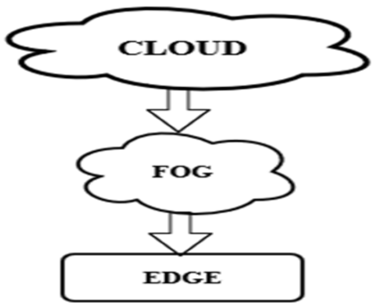

Figure 2 Fog Computing Environment [9]

An intermediate layer between the cloud and the IoT device is called FC. Figure 2 shows the FC environment. To reduce latency and NC, FC brings cloud services closer to IoT devices at the network edge. These days, minimum latency is a desirable feature in apps such as backup replies in the medical field, and FC guarantees minimum latency by providing real-time processing abilities for the transferred information [11]. FC, according to CISCO [12], [13], is the field where IoT data analysis takes place closer to the IoT devices that produce and procedure data. The nodes in a typical FC system are devices associated with the IoTs. These nodes are denoted as FNs. Fog devices can be the only device which has storage, connectivity, computing, etc. The FC layer stands between the cloud and IoT devices and has useful features, stored network bandwidth, storage near to the IoT, and Secure IoT Data.

1.2. Fog Computing Layers

FC is a dependent paradigm but an addition of cloud facilities to the edge. The Fog atmosphere [14] consists of three different layers: (i) Cloud (CL) (ii) Fog (FL) and (iii) Terminal Layer (TL) shown in Figure 3, while the architecture comprises groupings of FNs. Furthermore, data processing locally with a desired latency is possible with the fog.

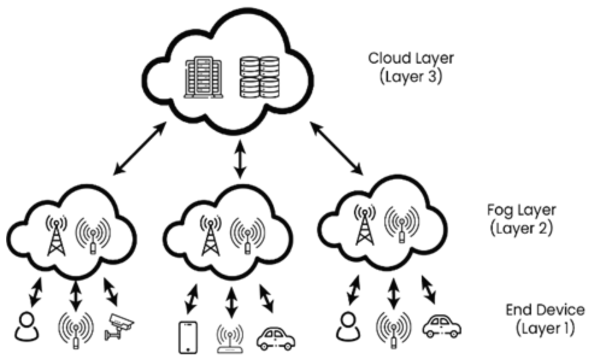

Figure 3 Design of Fog Computing [14]

The terminal layer consists of distinct end devices, which are physically separated. These devices take charge of becoming data and transmitting it to a high-level layer for data storage and processing. The devices could involve sensors, mobile phones, etc. The FL is situated at the edge of the network, suited at the network's edge [15]. This layer device is called FN and it is reliable for data transmission, storage, and computation. An FN is located in a fixed strategic area and might be movable or stationary. The cloud layer consists of storage strategies and servers with maximum presentation and computation power [16]. It is set for performing non-latency complex jobs transmitted by the lower, or Fog layer [17].

1.3. Characteristics of Fog Computing

Using the data produced by IoT devices and moving it to the cloud may get more difficult as the amount of IoT devices rises. To overcome these limitations, FC was developed. Table 1 defines how FC and Cloud computing (CC) differ in terms of the subsequent shared features.

Table 1 Characteristics that Contrast Between FC and Cloud Computing [9, 18]

Characteristics	Cloud Computing	Fog Computing
Deployment	Network core	Edge of the network
Scalability	Limited	High
Delay	High	Low

Geographical position	No	Yes
Mobility	Limited	Supported
Position awareness	No	Yes
Latency	High	Low
Architecture	Centralized	Decentralized

1.4. Fog Computing: Problem and Challenges

Because FC is heterogeneous, it has structural problems. Since the edge network can be utilized as an FN, some nodes may not be calculated to provide computations for general-purpose use. As a result, it could be not easy to integrate the general-purpose function into its usual duties. Furthermore, because some FNs have limited resources, expanding large-scale applications may encounter difficulties in the face of service orientation.

Thus, a programming environment that facilitates the creation of distributed applications is required. The fog's quantity of service can be greatly impacted by how the security system works. Data-centric integrity is implemented. Furthermore, it could be challenging to maintain the privacy of a large distribution network and authenticate access to services [19, 20, 21].

The problem formulation is related to the utilization of resource allocation in a fog environment, as directed below:

- The resource allocation (RA) issue is one of the key issues in executing IoT applications in FC. The optimal allocation of resource strategy will be the most effective in resolving this problem because it will reduce energy usage and communication costs.
- Another problem faced during resource allocation in a fog environment is that workload allocation should be done fairly in the fog and cloud layer to achieve minimum power usage with the restricted service delay.
- To solve existing research works, an optimized resource allocation algorithm will be used to optimize the execution time of priority tasks has to be minimized. Cloud dependency should be delayed to achieve better system performance.

1.5. Motivation

The motivation for deploying a MILALO model in fog computing is to optimize resource allocation, reduce execution time, and conserve energy. This model enhances scheduling, overall time and energy consumption, and cloud resource optimization.

1.6. Objectives

The various objectives of this research work are listed below:

- To design a novel fog node-assisted scheduling framework using an optimized mixed integer linear Ant Lion optimizer for resource allocation and utilization.
- Implement the suggested algorithm in a simulator and assess the performance of the algorithm.
- Compare the proposed algorithm with current resource allocation techniques and evaluate based on execution time, energy consumption, and resource utilization.

1.7. Focus of the Paper

- Designed a new fog node-assisted scheduling architecture using optimized mixed integer linear programming (MILP) scheduling for resource allocation (RA) and utilization with Ant Lion Optimizer (ALO). MILP is frequently utilized for system analysis (SA) and optimization as it defines a reliable and energetic approach for resolving huge, difficult issues.
- MILP method that rapid conversion and worldwide optimum with well-known solving techniques. The method builds upon existing efforts in procedure integration, considers constraints, and resolves them. After that, the ALO method is a current meta-heuristic approach that simulates antlions (ALs) foraging performance in exploring and attacking ants. Moreover, the ALO method suffers from local optimal and slow convergence speeds for some optimization issues.
- It deploys the researched model MILALO (Mixed Integer Linear Ant Lion Optimizer), which reduces cloud dependency and gives an energy-efficient (EE) and cost-effective solution. It also enhances the overall time and energy consumption as well as the cloud dependency of resources.
- It analyses the presentation of the researched method by comparing it with the current RA method, considering the metrics, execution time (ET), energy consumption (EC), and resource utilization.

1.8. Organization of the Paper

The rest of the paper is organized as follows: Section 2 explains related work. Section 3 presents the proposed mixed integer linear Ant Lion optimizer. Section 4 discusses the results. The paper ends in Section 5 with the conclusion.

2. RELATED WORK

This section elaborates on the various existing task scheduling, and RA-based methods in FC with the proposed method, problem, findings, and so on in Table 2. Next-generation communication applications are expected to require FC and IoT. However, the communication capability was explored to be controlled upon assessment of the growth in IoT devices.

Md Razon Hossain [10] presented a Scheduling-based Dynamic Fog Computing (SDFC) to overcome the gap of resource scattering. Moreover, three energy-related parameters are incorporated into the method to improve the resource allocation and the comparison is done based on two previous existing algorithms. The author in [22] proposed a task scheduling approach to reduce the complexity in real word and improve better efficiency.

Chao Yin and Juan Wang [23, 24], introduced a new hybrid approach for complex resource allocation to enhance energy efficiency and cost by considering the parameter load balancing, delay, energy consumption, and reliability. Weimin Liu et al. [25] explored the complex process of task scheduling using various optimization algorithms, including PSO, ABC, PGABC–PSO, SJF–PSO, and HSF.ABC & PSO. These methods were implemented on a system with 10 GB RAM and Intel(R) processors to evaluate performance in terms of time delay and energy consumption. Among the methods, the PGABC–PSO demonstrated notable improvements. The researchers found that this method significantly reduced both delay and energy consumption, making it a highly efficient approach for task scheduling. The results highlight the potential of advanced optimization techniques in enhancing system performance and energy efficiency.

Amit Kishor et al. [26] employed Smart ACO, Round Robin throttled, MPSO, and BLA algorithms to tackle complications in load balancing and latency due to NP-hard issues using MATLAB on a system with GB RAM and Intel(R) processors. Their approach improved QoS in the IoT-Fog tool. Jaspal Singh et al. [27] proposed a self-adaptive hybrid optimization algorithm (SHOA) to address complications in multiple task allocation using CloudSim. This method reduced energy consumption, migration rate, and SLA violation, providing better performance.

Author [28] utilized the Lotka-Volterra load balancer and Elman Hebbian-recurrent neural network cache (LV-EHRCC) for load balancing in FoT, addressing complications in resource allocation with IFogSim. This method improved makespan, bandwidth, and load balancing. Jalasri Mahendran et al. [29] introduced a Data Security Management model combining ACOMKSVM and block chain technology with WBANM for secure, private data sharing. This method enhanced accuracy, precision, response time, data security, and recall, effectively eliminating intermediate attacks. In [30] Raspberry Pi in fog and cloud computing, addressing transmission and computational delays, and improving response time and data transfer for better end-user services.

Table 2 Comparison of Discussed Approaches

Author's Name	Method	Comparative Methods	Gaps	Simulator\ Implementation Tool	Parameters	Findings
Md Razon Hossain et al. (2021) [10]	Scheduling-Based Dynamic Fog Computing (SDFC)	Real-Time Efficient Scheduling (RTES) First Come First Serve (FCFS)	Issues Of Scattered Resources.	iFogSim Java	Execution time Bandwidth Energy Consumption	It provides better augmenting resource utilization.
Hardik Mahendrabhai Patel et al. (2023) [22]	Task Scheduling Approach	Simple Priority-Based Allocation Priority-Based Allocation Using Load Comfort Index	Complications in the real-world method.	Fog Java	Latency Bandwidth Scalability Energy Efficiency Cost Location of Processing	This method provides better efficiency.
Chao Yin et al. (2023) [23]	GA and ACO	ACO IACO PSO	More implementation cost	Ifogsim Windows 7	Economic Cost Load balancing	It reduces the total cost, considering the user's QoS.

Juan Wang et al. (2019) [24]	A Hybrid Heuristic Method	IPSO IACO RR	Complications in multiple task allocation.	MATLAB Windows 7	Delay Energy Consumption Reliability	It improves the better performance of energy consumption.
Weimin Liu et al. (2023) [25]	PSO, ABC	PGABC–PSO PGABC SJF–PSO HSF.ABC & PSO MFO	Complex process of task scheduling	Window - 10 GB RAM Intel(R)	Time delay Energy Consumption	This method reduced delay and energy consumption.
Amit Kishor et al. (2021) [26]	Smart ACO	Round Robin Throttled MPSO BLA	Complications in Load balancing and Latency due to NP-hard issues	MATLAB GB RAM Intel(R)	Latency Time	It improves the QoS in the IoT-Fog tool.
Jaspal Singh et al. (2022) [27]	Self-Adaptive Hybrid Optimization Algorithm (SHOA)	LR-MMT RE-VMC MOABC-VMC.	Complications in multiple task allocation.	CloudSim	Energy Consumption Migration Rate SLA Violation	This method reduces energy consumption and provides better performance.
S.V. Nethaji et al. (2023) [28]	Lotka-Volterra Load Balancer And Elman Hebbian-Recurrent Neural Network Cache (LV-EHRCC)	LVEHRCC EPRAM Load Balancing For Fot	Complications in resource allocation.	IFogsim	Makespan Bandwidth Load Balancing	It improves the better performance of energy consumption.
Jalasri Mahendran et al. (2021) [29]	Data Security Management Model	ACOMKSVM Blockchain Technology With Wireless Body Area Networks (BC-WBANM) Model O Compressed And Private Data Sharing Framework (Cpds)	Security and privacy issues.	Window - 10	Accuracy Precision Response Time Data Security Recall	This method eliminates intermediate attacks.

Pratik Kanani et al. (2020) [30]	Raspberry Pi	Cloud Computing Fog Computing	Not suitable for time-scale healthcare services.	Ifogsim Java	Transmission Delay Computational Delay Data Transferred Response Time	This method provides better services to end-users.

3. PROPOSED MODELLING

This research aims to reduce the execution time required to perform resource allocation in fog computing by making enhancements to the optimized Model for Resource Allocation. The best cost solutions for the task execution are changed here with the aim of resource-allocating jobs. This results in increasing the time of execution and saving energy in a fog environment in comparison to the previous technique. There are three phases in which the Scheduling-based Dynamic Fog Computing algorithm is executed. Between the cloud and the citizen fogs, or general-purpose fog nodes, there is an MF layer. The MF is committed to bearing all computing costs associated with job scheduling and instantly the highest priority is being assigned to the highest qualified CF. The MF utilizes one technique to sort the citizen fogs depending on the available computational capacity, and another method is job scheduling jobs depending on their priority. Additionally, the proposed method ensures that the CF layers assets are used effectively, lowering dependency on the cloud.

3.1. Flowchart of Proposed Optimized Resource Allocation Technique

The proposed algorithm is divided into various steps in Figure 4. These entire steps help to build a cloud network with fog layers and ensure the successful execution of resource allocation and utilization. The proposed flow shows building the cloud network at the initial stage and building micro data centers at the fog layer. All the collected tasks from the physical layer are submitted to the initial fog layer in the queue. The listed tasks are processed with micro data centers at the fog layer with optimized load allocations and utilization process modules. The proposed optimization builds various execution patterns and provides low-cost execution solutions in terms of performance like energy consumption, time consumption, etc.

3.1.1. Research Design

The proposed framework for optimized resource allocation in fog computing. Figure 4 shows the research framework with the proposed methods. Initially, it built a cloud network and assigned the initial metrics for resources and task requirements. When a cloud network is built, a fog layer is built with micro data centers (DCs).

The proposed framework for RA in Fog Computing in figure 4, it collected the task from the physical layer (PL). If execution is prepared, then load the micro-DC layers with virtual machines (VMs). If execution is not prepared, then collect the task from the PL. Its initial metrics for resources and task requirements align sets with occupied sub-sets and available sub-sets. It initializes ants and locations of Ant Lions (ALs). It constructs an iterative procedure to evaluate the fitness of prepared solutions. When a procedure all the elements with the given objective function (OF) are done. It constructs the fitness set of procedure solutions. If the process is not completed, then the iterative procedure is to evaluate the fitness of the prepared solutions. It prioritizes solutions in the MILP method.

When it constructs an allocation matrix with optimal solutions. It gives feedback to the ALO phase and fine-tunes both metrics. Load balancing and allocations at VMs with optimization. It simulates a cloud and a physical layer for data transmission. It evaluates resource utilization, energy consumption, and execution time and compares them with the existing methods.

The process that is followed in the proposed architecture has various steps and sub-processes to achieve resource allocation and utilization in FC.

- Build cloud network: The first process is to create a virtual environment of cloud architecture where a network is formed with data centers and virtual machines. This helps to make the process smooth for getting tasks and managing resources on the cloud side along with optimization processes that get data to process and finalize the outcome. There are various submodules of this process attached to provide the list of resources virtually along with the specifications like what type of resources are in the network that participate within the process of resource allocation and utilization process.
- Build Fog layer with micro data centers and VMs: The micro data centers and VMs are part of the Fog layer which works processing the small tasks without interrupting or providing heavy load on the cloud network. This process needs some calculation that is used to find the right way to how the network processes task queues and how the VMs get managed.

RESEARCH ARTICLE

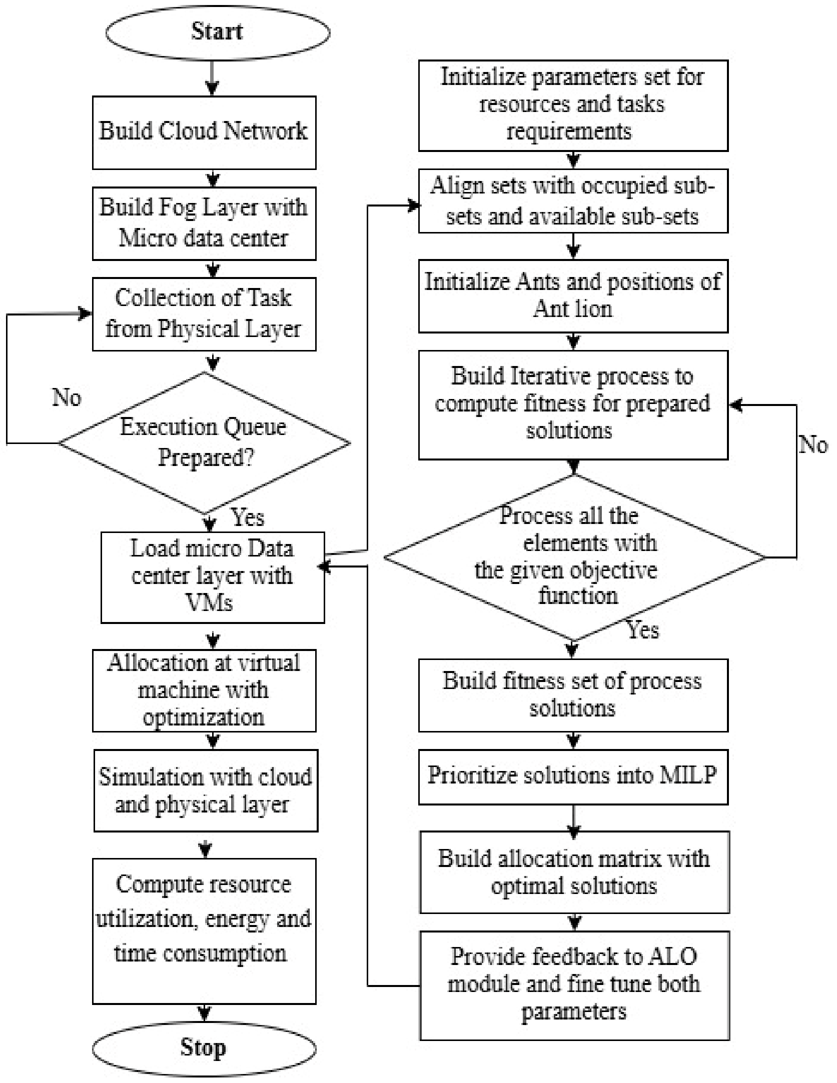
Start
Build Cloud Network
Build Fog Layer with Micro data center
Collection of Task from Physical Layer
Execution Queue Prepared?
No
Yes
Load micro Data center layer with VMs
Allocation at virtual machine with optimization
Simulation with cloud and physical layer
Compute resource utilization, energy and time consumption
Stop
Initialize parameters set for resources and tasks requirements
Align sets with occupied sub-sets and available sub-sets
Initialize Ants and positions of Ant lion
Build Iterative process to compute fitness for prepared solutions
Process all the elements with the given objective function
No
Yes
Build fitness set of process solutions
Prioritize solutions into MILP
Build allocation matrix with optimal solutions
Provide feedback to ALO module and fine tune both parameters

Figure 4 The Proposed Framework for RA in Fog Computing

- Collect tasks: The collection of tasks is the process where users submit all the tasks that need to be processed on the cloud side. These tasks can be anything like getting data from a database, doing some calculations, building images from data, etc. The first interaction layer of cloud architecture gets all the tasks and builds some task queues to send them within the optimization process. Here the tasks can be anything with some specification or requirement like what type of VM it needs to get processed on the cloud side. The heavy tasks need more configuration and bigger VM and small tasks need a small configuration VM to get processed.
- Loading VMs and load balancing: Before data comes under the load balancing and scheduling process, it needs to get loaded with the configuration within the optimization process. This layer provides extraction of data from the given network with the configuration like what type of machines are present within the network and what are various parameters of those machines.
- Align sets and subsets process: The process of sets and subsets is a way to get the data and their dependencies over multiple processes. Here it is the first layer of the optimizer where the optimizer gets aligned with the data that is available and that needs to be processed on the cloud server.
- Initialize Ants and positions of ANT-Lion and fitness value: The process of optimization starts from the initialization of the population where the optimizer initializes the population as a network and calculates the fitness value over all the tasks. It is used to find the best route of execution for all the tasks over the cloud network on behalf of some parameters like utilization, and energy

consumption. Time efforts etc. The calculated fitness value is used to find the optimal route and repeat till the end of the defined iteration. The iteration process helps to provide various cases so that the optimizer can able to find the best fitness score with a high accuracy value. Once the fitness value is evaluated by the optimizer, it is prioritized by MILP and it builds an allocation matrix for the allocator that works in between the network and users to process and show results. Later, according to the matrix, the task gets processed on the cloud network and updates results accordingly.

- Transmission and performance computation: The transmission belongs to results that the resource allocator evaluates from the network after execution and then transfers to the user. After completion of all the execution then results such as energy, time, etc., get calculated to see the performance of the researched design.

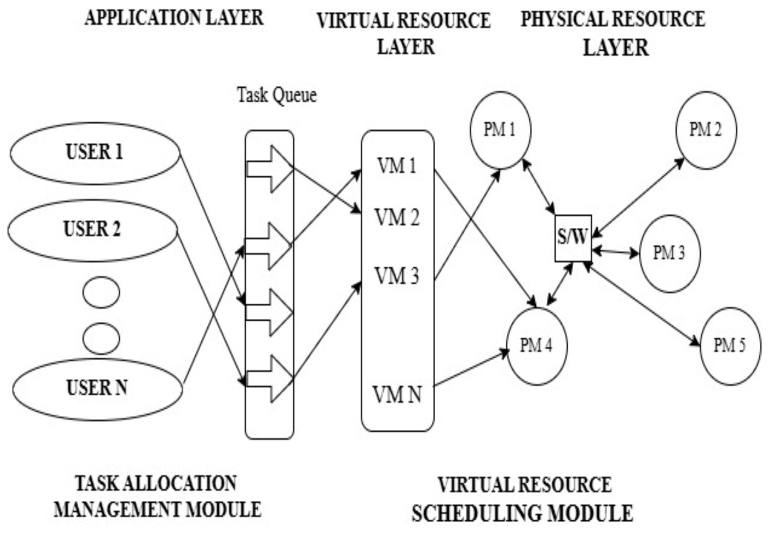

Figure 5 The Framework of the FC Resource Management System (RMS)

Figure 5 shows various modules of the resource management system. The main two modules that are responsible for the collection of tasks and the executions are:

- Task Allocation and Management Module: This module is attached to the application layer and is responsible for collecting tasks and preparing a queue at the server end. All submissions from the user end are arranged into a task queue and processed using the proposed architecture to provide efficient results. The first phase of the proposed approach begins at this layer, where the task queue is taken as input.

- Virtual resource scheduling module: The scheduling is implemented using a multi-threading approach, enabling the system to perform multiple tasks simultaneously. Virtual machines play a significant role in this processing as they are a subset of physical machines. The proposed approach utilizes these virtual machines to execute the tasks captured at the task allocation and management layer. Allocation and load-balancing tasks are performed in this layer by the proposed architecture. It processes the tasks and generates response data for users, which is then transferred back to the previous layer for interaction with the user.

Figure 5 shows the allocation of tasks in the cloud environment.

- The number of users who submit their tasks to the cloud network. All the tasks build queues at the cloud platform for executions.
- These queues are used to schedule tasks on virtual machines that are a subset of the actual physical machines.
- The data centers are high-speed computers and to give high availability to users and better utilization the network uses virtual machine mechanism.
- The execution at this stage is handled by schedulers and tasks management algorithms.

4. RESULTS AND DISCUSSIONS

In this section, a comprehensive description of the proposed MILALO (mixed integer linear Ant Lion optimizer) model using the MATLAB project desktop application is presented. An evaluation of the MILALO model's effectiveness is detailed in section 4.1, comparing current research with other methods such as SDFC (Scheduling-Based Dynamic Fog Computing) [10] and PGABC-PSO [19] optimization methods.

The scheduling-based dynamic fog computing framework (SDFC) aims to enhance resource utilization by optimizing task allocation in cloud-fog environments. This framework leverages various algorithms to address the challenges posed by dynamic workloads and resource availability, ultimately improving performance metrics such as response time and operational costs.

Section 4.2 provides an in-depth discussion of performance metrics, including energy consumption (EC), utilization consumption (UC), and time. The simulation setup is thoroughly explored in section 4.3, with the result analysis showcasing the MILALO model's capacity to reduce energy and time parameters when compared to existing methods. The proposed model aims to diminish cloud dependency and provide an energy-efficient (EE) and cost-effective solution

4.1. Experimental Settings

The experimental tool is elaborated and the proposed MILALO (mixed integer linear Ant Lion optimizer) model uses the MATLAB project desktop application and software with system requirements, OS window 8 up to, RAM 64GB, Intel Processor i3, and hard disk -4TB.

4.2. Performance Metrics

This section described resource allocation (RA) in fog computing (FC) performance parameters as are set of metrics used to calculate the performance of the MILALO model, such as SDFC, and PGABC-PSO. These introduced performance metrics are different such as EC, RU, and TC.

4.2.1. Energy Consumption (EC)

Here, the overall power is shown as the sum of the power of the total assets used in the task performance. The avg. power is the avg. power of all the assets utilized for the model. Hence, EC is the quantity of energy spent for the execution of the number of tasks and the evaluation formulation is defined as eq (1).

$$EC = \frac{Total\ Power}{Average\ Power} * Total\ no.of\ \ resources \qquad (1)$$

4.2.2. Resource Utilization (RU)

It is a crucial aspect of evaluating the presentation of PHYSICAL machines in the form of managing assets. Thus, it shows the amount of assets used for the performance of a defined amount of tasks. The evaluation of this RU aspect is defined in eq (2).

$$Resource\ Utilization = \sum_{i=1}^{i=n} No.of\ task\ (Ki) executed\ per\ resource\ (Ri) \qquad (2)$$

4.2.3. Time Consumption (TC)

It shows the time taken by the physical machines for the completion of the tasks. Thus, for evaluating the time for the execution of the tasks by physical machines by utilizing the subsequent formulation shown in eq (3).

$$Time\ \ Consumption = \sum_{i=1}^{i=n} Total\ Time + \frac{number\ of\ tasks\ (i)}{million\ instruction\ per\ \sec(MIPS)(i)} \qquad (3)$$

4.3. Results

The result analysis section is assumed to examine the calculation of the planned MILALO model. The research model is normally compared with the existing models like SDFC [10], and PGABC-PSO [19], to validate its efficiency over the other algorithms used for comparison.

4.3.1. Energy Consumption (EC)

The physical machines consume to give the services to the consumers, and the energy consumption is based on the total power of the physical machines.

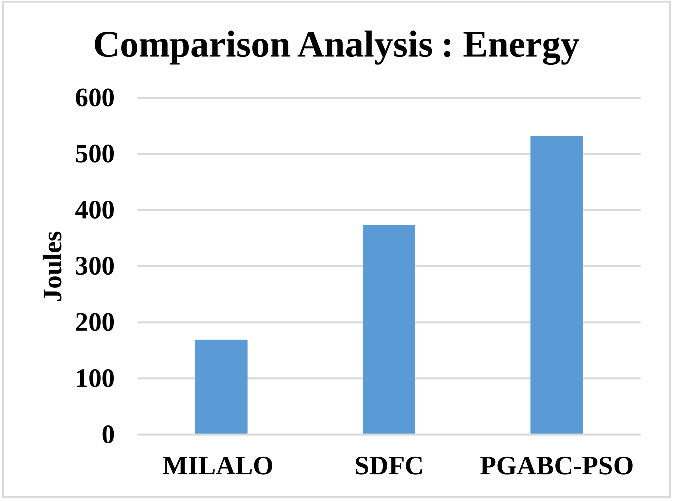

Figure 6 Comparison of EC (j)

The comparison of EC among various other algorithms is shown in Figure 6. This shows the effectiveness of the algorithms with the same number of input and processing architecture. The less EC shown by any algorithm is considered as high performance. Here MILALO, SDFC [10], and PGABC-PSO [19] are compared in Figure 6. The presentation of the proposed MILALO is high as it consumes less energy as compared to SDFC and PGABC-PSO.

4.3.2. Time Consumption

The time consumption is the total time taken by any algorithm to procedure the task queues. If the time consumption is high then the response time to the user will be high as well. The efficient algorithm takes less time and provides faster speed within the cloud environment for all the executions.

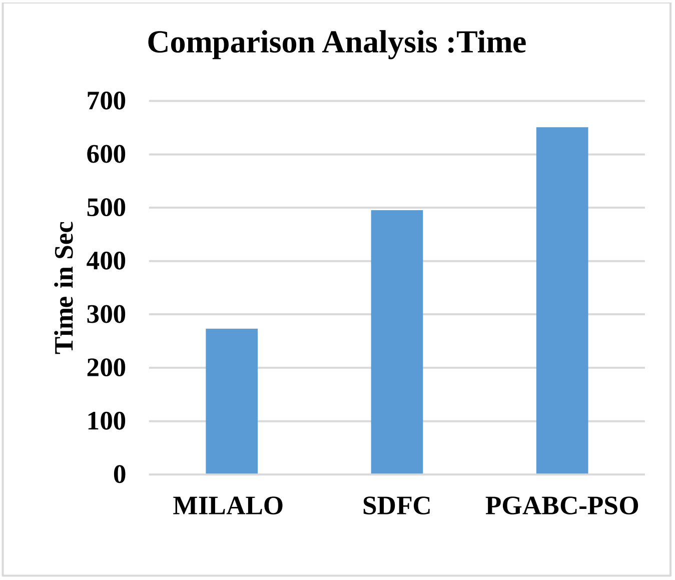

Figure 7 Comparison of Time in Seconds

Here the time taken by MILALO's proposed architecture takes less time as compared to the other SDFC [10] and PGABC-PSO [19] approaches as shown in Figure 7. So, it shows the high-speed execution of the proposed architecture as compared with the recent existing approaches.

4.3.3. Resource Utilization

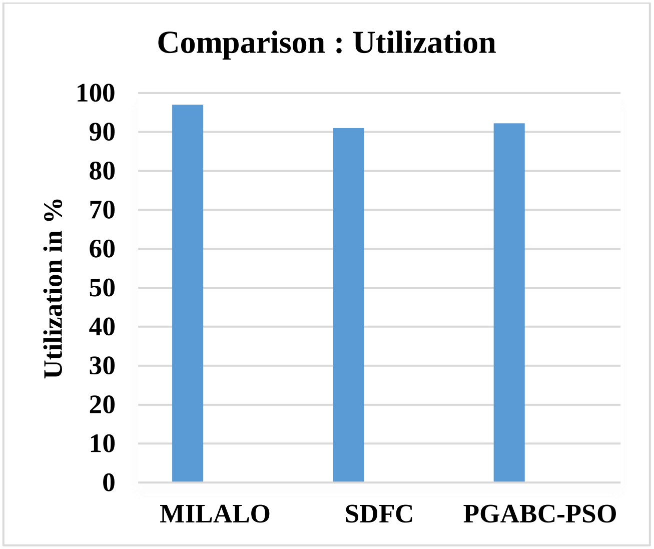

Figure 8 Comparison of Utilization Consumption

Resource utilization is a crucial aspect of evaluating the performance of physical machines in the form of managing resources. The main goal is to improve processing speed and reduce the demand on network bandwidth. Efficient resource utilization is crucial in fog computing to achieve high performance and low latency.

The resource utilization process shows how efficient the approach is to manage the network. Here in Figure 8, three different approaches are compared to get the results of the utilization process. The proposed architecture shows high performance as in Figure 8 shows high utilization as compared with the other existing approaches.

4.4. Discussions

The performance comparison needs various test cases where the approaches get executed for different inputs. Some of the inputs need to be bigger and some of them need smaller to check the approaches performance of different approaches.

Table 3 Performance with 10 Nodes with MILALO

Tasks	Energy consumption (J)	Time consumption (Sec.)	Resource Utilization (%)
10	21	17	95
100	178	287	97
500	612	491	97
1000	917	721	95
5000	1832	1312	97
10000	4619	2917	98

Table 3 shows various test cases with a static network size of 10 nodes. The testing of load is to show the effectiveness of the implemented model. While increasing the load from 10 to 10,000 tasks on a small network, the proposed architecture shows less energy and time consumption. It shows the stable performance of resource utilization within the execution of all the test cases for MILALO.

Table 4 Performance with 10 Nodes with SDFC

Tasks	Energy consumption (J)	Time consumption (Sec.)	Resource Utilization (%)
10	47	59	91
100	383	507	89
500	857	690	86

1000	1491	1037	91
5000	2302	1783	91
10000	6082	4231	88

Table 4 shows various parameter calculations in 10 to 10000 task test cases. This is built on the network size of 10 nodes. The parameters are energy, time consumption, and resource utilization within the SDFC. The performance of SDFC shows more energy and time consumption than the proposed MILALO.

Table 5 Performance with 10 Nodes with PGABC-PSO

Tasks	Energy consumption (J)	Time consumption (Sec.)	Resource Utilization (%)
10	58	72	93
100	513	643	93
500	1096	1187	95
1000	1821	1452	93
5000	3152	2465	91
10000	7455	5241	90

Table 5 shows the performance of PGABC-PSO on behalf of three parameters same as table 4 and table 5. The network size is 10 nodes and performance is calculated on behalf of 10 to 10000 tasks. The performance of PGABC-PSO needs some improvements as compared to the MILALO and SDFC approach in Tables 3 and 4.

Table 6 Performance with 20 Nodes with MILALO

Tasks	Energy consumption (J)	Time consumption (Sec.)	Resource Utilization (%)
10	17	14	96
100	102	163	98
500	435	211	97
1000	714	513	97
5000	1488	918	98
10000	3425	2496	97

Similar to Table 6 the performance of MILALO is calculated based on 20 nodes network in Table 6. This various is used to see the time and energy consumption within the execution. If the network has a huge number of nodes, then the proposed architecture should be able to manage and utilize the network. The better the management of task load and network, the better an approach will perform. Here the performance is computed on behalf of energy consumption, time consumption, and resource utilization.

Table 7 Performance with 20 Nodes with SDFC

Tasks	Energy consumption (J)	Time consumption (Sec.)	Resource Utilization (%)
10	35	42	92
100	291	418	91
500	611	492	87
1000	1019	830	89
5000	1728	1423	91
10000	4521	3424	88

The performance over small to heavy loads is calculated in Table 7. It shows various executions where the number of tasks from 10 to 10000 are executed to see the performance of SDFC. The load taken in this ratio is to see whether the SDFC approach can handle the load and utilize the network in different scenarios or not. Here SDFC executed the tasks successfully and showed the parameters.

Table 8 Performance with 20 Nodes with PGABC-PSO

Tasks	Energy consumption (J)	Time consumption (Sec.)	Resource Utilization (%)
10	42	58	93
100	455	518	95
500	769	985	94
1000	1423	996	93
5000	2695	2014	94
10000	5547	4278	91

Table 8 shows the results of different executions on 20 node networks. The tasks started with 10 and at the final stage of test cases, it ended with 10000. The performance of every test case was noted in terms of energy consumption, time consumption, and resource utilization.

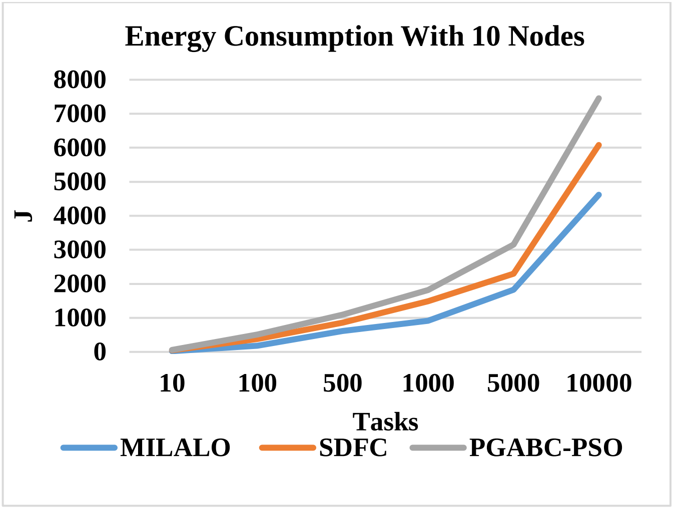

Figure 9 Test Cases Comparison of Energy Consumption with 10 Nodes

The test cases comparison in Figure 9 shows a plot to present the performance of MILALO, SDFC, and PGABC-PSO. Here the plot is used to see the one graph performance where all the approach shows their results for different executions of energy consumption. Here the performance of MILALO is better in all the cases with less energy consumption when it is tried with 10 nodes and 10 to 10000 tasks.

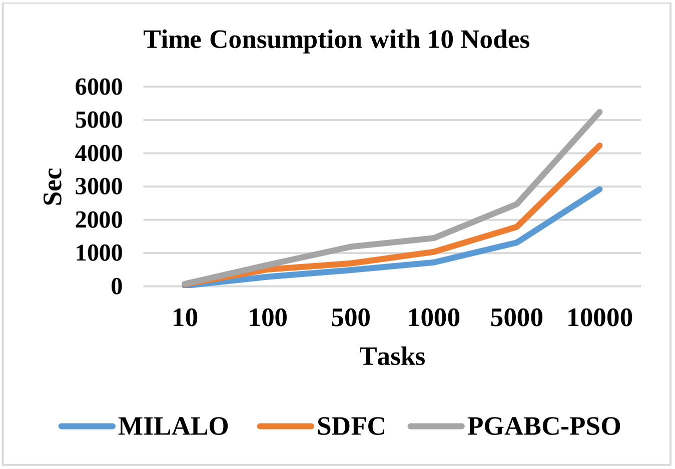

Figure 10 Test Cases Comparison of Time Consumption with 10 Nodes

Figure 10 shows test case results of time consumption for all three approaches. The tests are moving from 10 to 10000 and the performance of MILALO is showing less time consumption in all the cases. The rise in time is because of the number of tasks in the same network. But the better thing is here, all the cases seem good and show high-performance MILALO technique to process the task queues.

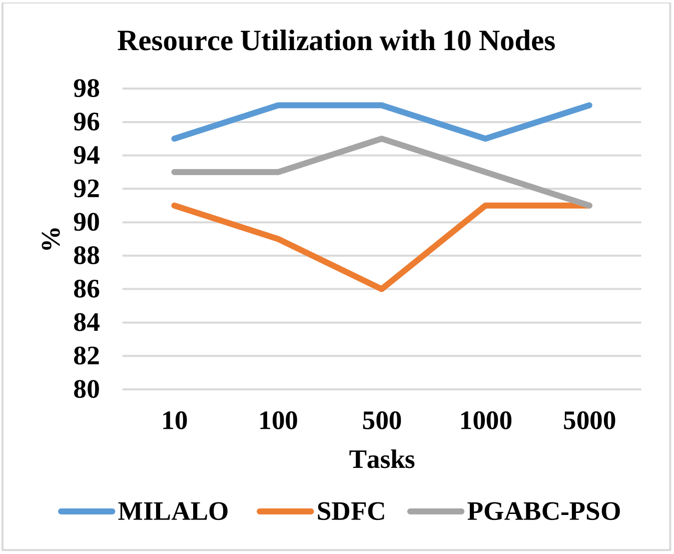

Figure 11 Test Cases Resource Utilization with 10 Nodes

The resource utilization tests using a load of 10 tasks to 10000 tasks are performed in Figure 11. This is a way where the resource allocator performance gets validated on the behalf of utilization metric. When a heavy load occurs then the allocation process needs to pay attention to send the tasks and get a response from the network end. It should be well planned and organized so that the issues can be rectified during allocation only. Here in all the cases, the performance of MILALO is better, and the shows high resource utilization as compared with the SDFC and PGABC-PSO approaches.

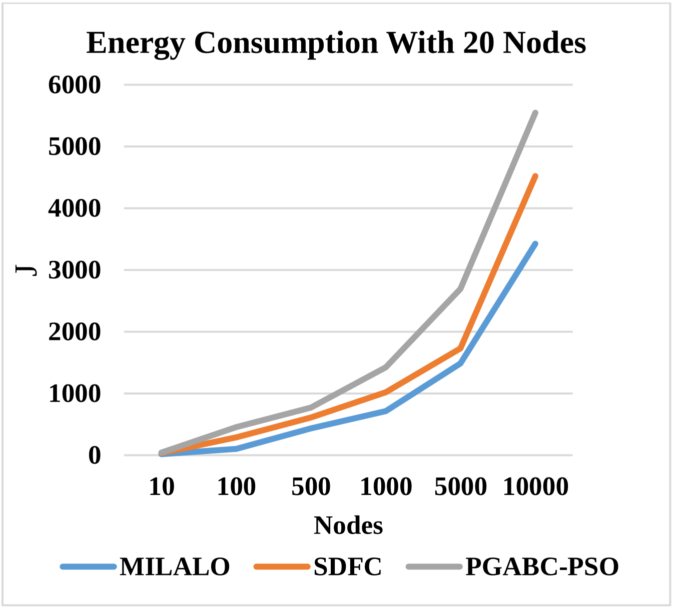

Figure 12 Test Cases Energy Consumption with 20 Nodes

In Figure 12, the test cases same as in Figure 9 were performed with the number of nodes 20 in size. The validation process here is to see if along with several tasks, the network size can be changed and increase in number of nodes. The execution process gets faster as the network has more nodes available and the ideal energy can be reduced with this process. So, in short here the MILALO architecture again shows better results for all the test cases and provides less energy consumption in all the cases.

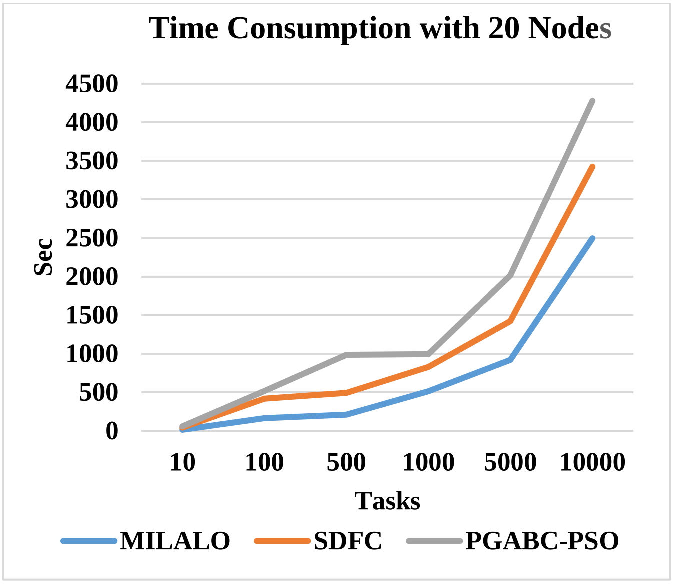

Figure 13 Test Cases Time Consumption with 20 Nodes

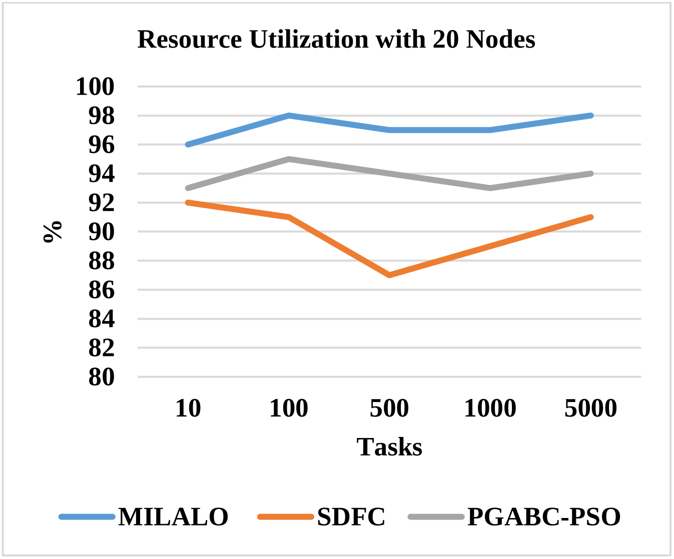

Figure 14 Test Cases Resource Utilization with 20 Nodes

The time parameter in the test cases is shown in Figure 13 with 20 nodes in the network. Here when nodes increase in size then the time also decreases. It happens due to the availability gets increased and the allocator has more resources to process the tasks. The time consumption should be less and here the test cases with 20 nodes in the network show the high performance of the proposed architecture with MILALO.

In figure 14 it shows the various test cases with 20 nodes and 10 to 10000 tasks. The graph shows resource utilization by the MILALO, SDFC [10], and PGABC-PSO [19] approaches. If the load increases, it's hard for the allocation process to manage and distribute the resources within the task queues. Here the performance of MILALO shows high performance in the utilization compared to other approaches.

Table 9 Comparison Analysis

Methods	Energy Consumption (J)	Time Consumption (Sec)	Resource Utilization (%)
MILALO	169	279	97
SDFC [10]	373	495	91
PGABC-PSO [19]	532	651	92.2

Table 9 depicts a comparison of the proposed method with existing algorithms in terms of energy consumption, time consumption, and resource utilization. The MILALO is designed to optimize resource allocation efficiently, reducing energy consumption. This method achieves the lowest energy usage among the compared algorithms. In contrast, SDFC and PGABC-PSO are less energy-efficient, consuming more energy than the proposed algorithm. In terms of time consumption, the proposed model demonstrates efficient time usage. The quick execution time highlights the method's capabilities to allocate resources and execute tasks rapidly. The existing algorithms have longer processing times. MILALO stands out as the best choice for scenarios requiring maximum resource utilization, while SDFC and PGABC-PSO are also efficient but slightly less effective in utilizing resources to their fullest extent. For resource allocation, the proposed model achieves the highest resource utilization at 97%, whereas SDFC and PGABC-PSO have resource utilization rates of 91% and 92.2%, respectively. They are slightly less effective in fully utilizing resources.

5. CONCLUSION

The proposed work concluded that the initialization of the cloud network and different parameters such as the number of resources and task requirements. The initial layer is a fog layer with microdata centers. All the tasks are collected from the physical layer. When the execution is prepared, they load the micro datacenter's layers with virtual machines. Before data comes under the load balancing and scheduling process, it needs to get loaded with the configuration within the optimization process. The process of optimization starts from the initialization of the population where the optimizer initializes the population as a network and calculates the fitness value over all the tasks. It is used to find the best route of execution for all the tasks over the cloud network on behalf of some parameters like utilization, energy consumption, etc. The calculated fitness value is used to find the optimal route and repeat till the end of the defined iteration. The iteration process helps to provide various cases so that the optimizer can able to find the best fitness score with a high accuracy value. Once the fitness value is evaluated by the optimizer. The transmission belongs to results that the resource allocator evaluates from the network after execution and then transfers to the user. After completion of all the execution then results like energy value at 169j, time value is 279 sec, and resource utilization value is 97% get calculated to see the performance of the research architecture. The proposed model has 6% improved resource utilization. The further enhancement of the proposed hybrid method offers limited scalability but can be improved by using a huge number of IoT devices and servers in the Fog background.

REFERENCES

[1] Katal, A. and Sethi, V. (2022) ‘Energy-efficient cloud and Fog computing in internet of things: Techniques and challenges’, Cloud and Fog Computing Platforms for Internet of Things, pp. 67–83.

[2] Basharat, A. and Mohamad, Mohd.M. (2022) ‘Security challenges and solutions for internet of things based smart agriculture: A Review’, 2022 4th International Conference on Smart Sensors and Application (ICSSA).

[3] Jamil, B. et al. (2022) ‘Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and Future Directions’, ACM Computing Surveys, 54(11s), pp. 1–38.

[4] Veith, A. da, Dias de Assunção, M. and Lefèvre, L. (2023) ‘Latency-aware strategies for deploying data stream processing applications on large cloud-edge infrastructure’, IEEE Transactions on Cloud Computing, 11(1), pp. 445–456.

[5] Chennam, K.K. et al. (2021) ‘Smart Cities Data Analysis Using Fog Computing’, 4th Smart Cities Symposium (SCS 2021).

[6] Aljićević, Z. et al. (2022) ‘Resource allocation model for cloud-fog based smart grid’, SSRN Electronic Journal.

[7] Alshouiliy, K. and Agrawal, D.P. (2021) ‘Confluence of 4G LTE, 5G, fog, and cloud computing and understanding security issues’, Fog/Edge Computing For Security, Privacy, and Applications, pp. 3–32.

[8] Jamil, B. et al. (2022) ‘Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and Future Directions’, ACM Computing Surveys, 54(11s), pp. 1–38.

[9] Daase, C. et al. (2023) ‘The Future of Commerce: Linking modern retailing characteristics with cloud computing capabilities’, Proceedings of the 25th International Conference on Enterprise Information Systems.

[10] Hossain, M.R. et al. (2021) ‘A scheduling-based dynamic fog computing framework for Augmenting Resource Utilization’, Simulation Modelling Practice and Theory, 111.

[11] Mahajan, K. (2023) Fog computing: A systematic review of Architecture, Iot Integration, algorithms and research challenges with insights into cloud computing integration.

[12] Sharma, N. and Prabha, C. (2021) ‘Computing paradigms: An overview’, 2021 Asian Conference on Innovation in Technology (ASIANCON).

[13] Awaisi, K.S. et al. (2021) ‘Simulating fog computing applications using ifogsim toolkit’, Mobile Edge Computing, pp. 565–590.

[14] Das, R. and Inuwa, M.M. (2023) ‘A review on Fog computing: Issues, characteristics, challenges, and potential applications’, Telematics and Informatics Reports, 10.

[15] Stavrinides, G.L. and Karatza, H.D. (2022) ‘Resource allocation and scheduling of real-time workflow applications in an IOT-fog-cloud environment’, 2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC).

[16] Katal, A. and Sethi, V. (2022) ‘Energy-efficient cloud and Fog computing in internet of things: Techniques and challenges’, Cloud and Fog Computing Platforms for Internet of Things, pp. 67–83.

[17] Solanki, M.S. (2021) ‘Fog computing: A conceptual and practical overview’, International Journal of Innovative Research in Computer Science &amp; Technology, pp. 158–162.

[18] Alsadie, D. (2024) ‘A Comprehensive Review of AI Techniques for Resource Management in Fog Computing: Trends, Challenges, and Future Directions’, IEEE Access, 12, pp. 1180–118059.

[19] Das, J., Ghosh, S.K. and Buyya, R. (2021) ‘Geospatial Edge-Fog Computing: A systematic review, taxonomy, and Future Directions’, Mobile Edge Computing, pp. 47–69.

[20] Premalatha, B. and Prakasam, P. (2024) ‘Optimal Energy-efficient resource allocation and fault tolerance scheme for task offloading in IOT-Fog Computing Networks’, Computer Networks, 238, p. 110080.

[21] Mebrek, A. and Yassine, A. (2024) ‘Intelligent Resource Allocation and task offloading model for IOT applications in fog networks: A game-theoretic approach’, IEEE Transactions on Emerging Topics in Computational Intelligence, pp. 1–15.

[22] Patel, M. and Modi, K. (2023) ‘Dynamic Resource Allocation for Real-Time Task Scheduling in Cloud-Fog Computing: A Cost-Effective and Low-Latency Approach’, Approach,” International Journal of Intelligent Systems and Applications in Engineering, 11(3), pp. 1222–1228.

[23] Yin, C. et al. (2023) ‘An optimized resource scheduling algorithm based on Ga and ACO algorithm’, The Journal of Supercomputing, 80, pp. 4248–4285.

[24] Wang, J. and Li, D. (2019) ‘Task scheduling based on a hybrid heuristic algorithm for smart production line with Fog Computing’, Sensors, 19(5).

[25] Movahedi, Z., &Defude, B. (2021). An efficient population-based multi-objective task scheduling approach in fog computing systems. Journal of Cloud Computing, 10(1), 1- 31.

[26] Kishor, A. and Chakarbarty, C. (2021) ‘Task offloading in fog computing for using smart ant colony optimization’, Wireless Personal Communications, 127(2), pp. 1683–1704.

[27] Singh, J. (2022) ‘An Optimal Resource Provisioning Scheme Using QoS in Cloud Computing Based Upon the Dynamic Clustering and Self-Adaptive Hybrid Optimization Algorithm’, International Journal of Intelligent Engineering and Systems, 15(3), pp. 148–160.

[28] Nethaji , S.V. and Chidambaram, M. (2023) ‘Lotkavoltera and Elman Hebbian Recurrent Neural Network Cache-based resource allocation in fog environment’, International Journal of Intelligent Engineering and Systems, 16(2), pp. 228–239.

[29] Mahendran, J. and Lakshmanan, L. (2022) ‘Fog computing with IOT device’s data security management using density control weighted election and extensible authentication protocol’, International Journal of Intelligent Engineering and Systems, 15(1).

[30] Kanani, P. and Padole, M. (2020) ‘Implementing and evaluating health as a service in fog and cloud computing using Raspberry Pi’, International Journal of Intelligent Engineering and Systems, 13(6), pp. 142–155.

Authors

Er. Neha Sharma is a research scholar (Pursuing Ph.D.) from chandigarh University Gharuan Mohali, Punjab, India. She received M.E. degree in Computer Science and Engineering from Chandigarh University, Gharuan, Mohali (Punjab) India. She also had published several research papers in national as well as international Journals and conferences of repute. She has research contributions in the area of cloud and Fog.

RESEARCH ARTICLE

Prof. (Dr.) Deepti Sharma is presently working as Program Leader, Cloud Computing, DevOps, ME and Full Stack Web Development. in Chandigarh University, Punjab, India. She has received M.Tech Degree(Computer Science Engineering)and Ph.D. Degree in Cloud Computing. She is having an experience of more than 24 years in the field of teaching, research and administration. Her area of interests are cloud computing, energy efficiency, mobile computing and IOT. She had published several reputed International/SCI papers and attended several National and International IEEE/Springer conferences. She has guided more than 40 PG theses and one Ph.D. scholar. She is currently guiding 5 PhD. Scholars. She has recently certified as facilitator for "AI for Future Workforce Program" by Intel corporation under Intel Readiness Program. She is also certified for AWS Academy graduate from AWS Academy foundations. She has written 5 book chapters in reputed publications (Springer, Scrivener). She is a reviewer of various reputed (IEEE, Springer) Conferences. She has published 6 Patents.

How to cite this article:

Neha Sharma, Deepti Sharma, "Implementation and Analysis of Fog Node-Assisted Scheduling and Optimization of Resource Allocation and Utilization", International Journal of Computer Networks and Applications (IJCNA), 11(6), PP: 855-869, 2024, DOI: 10.22247/ijcna/2024/51.