A Meta-Heuristic based work Scheduling and Resource Optimization Framework for Cloud Virtual Machines

M.Rupasri[1], G.P.S. Varma[2], Hemalatha Indukuri[3]

[1]Research Scholar, Department of Computer Science and Engineering, , Koneru Lakshmaiah Education Foundation,Vaddeswaram, Guntur, Andhra Pradesh.

[2]Professor, Department of computer science and engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh

[3]Professor, Department of IT, S.R.K.R. Engineering college,Bhimavaram Andhra pradesh

HOW TO CITE:
M.Rupasri, G.P.S. Varma, Hemalatha Indukuri3(2026). A Meta-heuristic based work scheduling and resource optimization framework for cloud virtual machines. International Journal of Special Education, 41(8s), 515-527

COPYRIGHT STATEMENT:
Copyright: © 2026 Authors.
Open access publication underthe terms and conditions of the Creative CommonsAttribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).

ABSTRACT:
Resource allocation and cloudlet scheduling in cloud computing are challenging issues, particularly when a medium to large number of tasks are involved. The concurrent execution of multiple cloudlets on available resources is important to satisfy user expectations and maximize system performance. Dealing with load balancing patterns and minimizing makespan holds significant importance in this context. To a large extent, task scheduling remains a difficult problem despite advances in cloud computing.Task scheduling is the assignment of user tasks to Virtual Machines (VMs) in such a way as to reduce turnaround time and increase resource utilization. The NP-hard problem complexity of (O(mn)) makes scheduling (n) tasks on (m) devices a challenging problem. The wide range of solutions explored in the task scheduling process highlights the need for algorithms that can return optimal solutions within polynomial runtime.To tackle these issues, this paper presents the Cluster-based Energy Efficient Q-Fruitfly Algorithm (CEQFA) schema for cloud computing systems. Utilizing clustering, the proposed approach groups cloud Virtual Machines (VMs) according to their workloads. It then employs a Q-Fruitfly Algorithm to optimize the assignment of VMs to physical machines. The results show that CEQFA can reduce computation time and energy consumption by up to 6%, while satisfying the Quality of Service (QoS) requirements of cloud users..

Keywords: Federated Learning, Mobile Edge Computing, Energy Optimization, Task Load Balance, Nonlinear SVM, Swarm Optimization.

1. Introduction

Cloud computing is the sharing of resources based on consolidation and multitenancy for application deployment and user information sharing. Cloud providers give high priority to load balancing, which automatically distributes service loads in response to growing user demands and business requirements.

Effective load balancing is essential for improving server performance in dynamic cloud

environments. In a cloud computing environment with multiple workloads, load balancing is a challenging issue. The selection of appropriate algorithms is important to ensure that all computing nodes process a similar amount of workload simultaneously. Load balancing protocols contribute to the efficient distribution of tasks within a cloud infrastructure, allowing distributed resources to be utilized effectively. Large computational tasks can be processed efficiently by dividing them into smaller computations, which is a common load balancing strategy used in cloud platforms.

Static load balancing methods are based on system application behavior and statistical information, utilizing knowledge of machine resources and processor outputs available at the beginning of execution [1–2].

Virtualization, a fundamental technology in cloud computing, provides shared resources. When users initiate multiple requests, virtualization assigns a logical representation to each physical resource and provides a reference to the corresponding physical resource. Through the optimized mapping of Virtual Machines (VMs) to Physical Machines (PMs), virtualization establishes the foundation for an efficient computing environment. This dynamic mapping technique enables virtualization to accommodate variations in workload demand and computational requirements.

Various forms of virtualization are employed in cloud computing environments. Network virtualization abstracts end-user network components, allowing network hardware to be virtualized and managed through a centralized network management software console [3]. Data virtualization makes data available in the appropriate format to any authenticated machine. The database schema should be sufficiently flexible to avoid data compatibility issues or data loss when different clients access data in various ways.

At the foundation of virtualization lies the Virtual Machine Manager (VMM), commonly known as the hypervisor. The hypervisor virtualizes the underlying hardware, creating an environment for operating systems and enabling application virtualization. It manages host processors and hardware resources to support the execution of multiple guest operating systems on a single physical machine.

In cloud environments, pending jobs are assigned to dedicated resources through finite-size multiple-queue scheduling within an Intercloud framework to optimize resource utilization. Auto-scaling mechanisms consider dynamic factors such as waiting time and queue length while continuously monitoring workload variations [4].

Quality of Service (QoS) parameters, including latency, packet loss, throughput, availability, and reliability, are important metrics for evaluating the performance of networks, systems, and services. Effective load balancing through load balancers or traffic managers improves resource utilization, reduces response time, and increases system availability.

Improved clustering techniques group servers or nodes according to resource utilization characteristics while considering factors such as geographical location, server capacity, and application requirements. Throughput is a key factor influencing data transmission efficiency and is particularly important for high-performance applications. Availability and reliability are critical for uninterrupted service delivery. Availability refers to the operational uptime of a system or service, whereas reliability represents the probability that the system performs its intended function without failure when required. Both are fundamental aspects of service quality and continuous service delivery.

Figure 1: Federated machine learning local and global models

Mobile cloud computing provides powerful functionalities and improved user experiences for mobile terminals. An in-depth evaluation of trade-offs involving heterogeneity, mobility, and security considerations has been extensively studied. Prolonged or permanent network disruptions caused by user mobility have continuously challenged application execution performance. To overcome the limitations of static partitioning, adaptive partitioning methods have been developed to accommodate dynamic network conditions.A heuristic-based computational solution has been proposed to improve energy efficiency in mobile devices. The method identifies an optimal path with minimum energy consumption and delay for offloading smartphone application tasks to cloud infrastructure, thereby reducing processing costs. This work exploits the concept of code offloading [15] to minimize mobile execution time and reduce energy consumption. The primary objective is to improve the energy efficiency of mobile devices and enhance application performance through reduced execution time. Furthermore, a queueing-theoretic formulation is introduced for the weighted minimization of a composite metric that includes energy consumption and several additional performance factors [3].

In Mobile Cloud Internet of Things (MCIoT) environments, analytical methods have been developed to determine optimal task offloading strategies within IoT infrastructures, thereby improving application performance. The results reveal variations in processing time and energy consumption between smart neighboring devices and cloud nodes. The studies further investigate network-related factors that influence application efficiency.

Challenges associated with live video processing, particularly severe latency and bandwidth constraints, have been reported in [4]. Research on computational federated data offloading methodologies indicates that application partitioning can be performed either through pre-defined application segmentation for one or more offloading locations or dynamically during run-time execution. In pre-determined partitioning (application slicing), executable segments are identified before the application enters the run-time phase. In contrast, dynamic application partitioning determines executable segments during execution [4]. These frameworks support multiple levels of segmentation granularity, including class-level, method-level, component-level, and module-level partitioning [5].Mobile applications can generally be classified into two categories: computation-intensive applications and communication-intensive applications. Evaluating computational requirements is important for effective task offloading, particularly in mobile cloud computing environments. Task offloading efficiency

depends on several factors, including network conditions, application characteristics, and the computational capabilities of mobile devices and cloud infrastructures.

As mobile devices become increasingly integrated into everyday life, they have evolved into powerful communication platforms unrestricted by time and location. Users access a wide range of services through mobile applications that execute either locally on devices or remotely on cloud servers through wireless communication technologies [6].Portable computing devices rely on battery power while providing user interaction, data management, and wireless communication capabilities. These devices operate under several resource constraints, including limited energy, memory, processing capability, transmission speed, mobility, and security requirements. Such limitations significantly affect Quality of Service (QoS). Consequently, software applications for mobile devices must support rapid deployment and rollback with minimal intervention from service providers.

The emergence of mobile applications and cloud-based service platforms has accelerated the integration of cloud technologies into mobile computing environments, giving rise to Mobile Cloud Computing (MCC) [7].

The key contributions of this work are summarized as follows:

This paper proposes EEQPMHO, an energy-efficient queueing-based metaheuristic framework to improve task distribution across mobile edge-cloud devices in mobile computing environments.

Queueing theory is integrated with swarm intelligence dynamics, enabling the optimization process to consider system congestion and task waiting delays, thereby improving real-time processing performance.

A localized non-linear Support Vector Machine (SVM) training scheme is proposed for edge devices, enabling privacy-preserving learning through kernel-based classification and personalized adaptation to device-specific data distributions.A federated learning aggregation mechanism is introduced to synchronize global models without transmitting sensitive user data from edge devices, providing improved scalability and robustness.

A hierarchical edge-cloud aggregation framework is proposed, incorporating edge-server-level model aggregation before cloud-level aggregation. This significantly reduces communication overhead and supports scalable deployment in large-scale mobile edge-cloud environments.

The remainder of this paper is organized as follows. Section II presents a comprehensive literature review of traditional federated learning and static task scheduling approaches, along with their limitations. Section III describes the proposed model. Section IV presents the experimental setup, performance metrics, and comparative analysis. Finally, Section V concludes the study, summarizes the major contributions, and discusses potential future research directions.

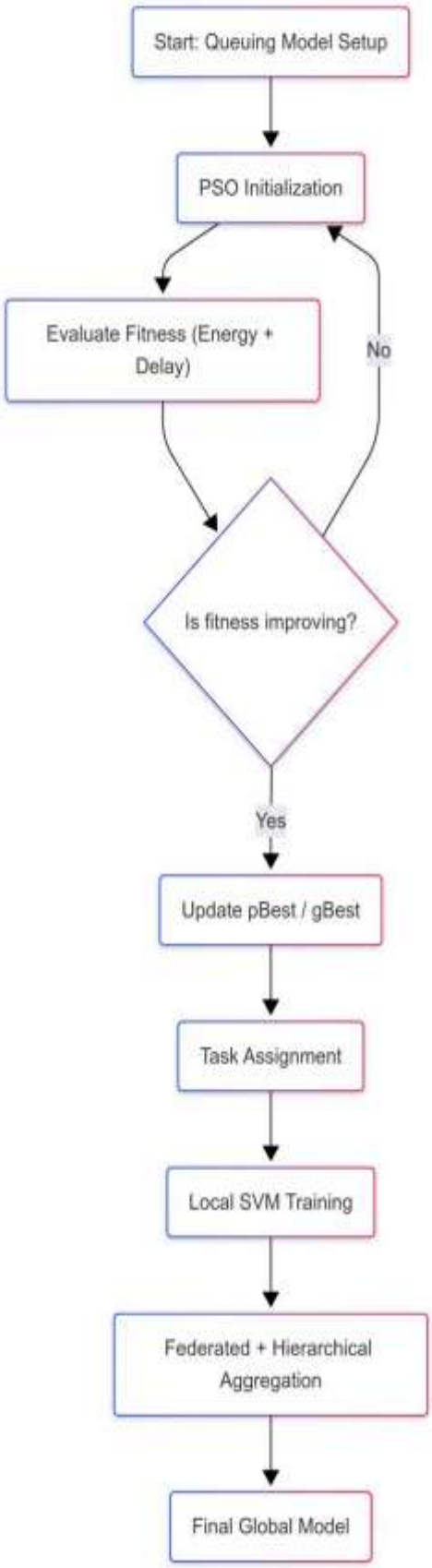

Figure 2: Overall EEQPMHO Model

A queuing-based PSO model, such as the Energy Eficient Queuing Probability Based Meta-Heuristic Optimization (EEQPMHO), is motivated by the need to address dynamic and resource-constrained environments especially in systems like edge computing, cloud offloading, or IoT task scheduling. Here is a precise explanation of queuing metric on dynamic task scheduling process:

Realistic System Modeling via Queuing Theory

In real-world edge or cloud systems, task arrivals and processing times are naturally modeled using queuing theory (e.g., M/M/1 or M/G/1 queues).

Queuing metrics (e.g., waiting time, service time, utilization) directly affect energy consumption and response latency.

Integrating Queuing Metrics into Fitness Evaluation Unlike standard PSO, which typically targets resource consumption or balancing loads, the queuing-based variant integrates system latency and task wait times into the fitness evaluation.It assesses queue lengths, service times, and arrival rates, allowing particles to explore solutions that reduce overall response time.This approach adapts to real-time task flow, improving scheduling accuracy by factoring in congestion and expected service delay.

$F(\chi_i(t)) = f_{energy} + \lambda 1\ f_{queue_delay} + \lambda 2\ f_{drop_probability}$

This leads to a multi-objective optimization aligned with Quality of Service (QoS) metrics.

Improved Convergence Behavior

By encoding queuing delay probabilities into particles' fitness values, the search space reflects temporal congestion, reducing convergence toward impractical allocations.

Dynamic Adaptation to Workload Fluctuations

Queuing models adapt to fluctuating workloads, allowing PSO particles to dynamically reposition based on updated task arrival rates and service rates.

Energy Efficiency Enhancement

Queuing theory allows estimation of system energy as a function of task wait times and server idleness:

$E = \eta 1 \cdot \text{Service Energy} + \eta 2 \cdot \text{Idle Energy} + \eta 3 \cdot \text{Rescheduling Cost}$ The model thus promotes minimization of energy-per-task while ensuring timely execution.

F. Prioritized Resource Allocation

The queuing discipline (FIFO, priority queue, etc.) enables differentiated task handling, which is crucial for heterogeneous task types in mobile and edge environments.

Table 2 Notations

$\chi i(0)$	Position vector of particle i at time 0
$\chi ia(0)$	Assignment of task α in particle i's position vector at time 0
$\nu i(0)$	Velocity vector of particle i at time 0
$\nu ia(0)$	Velocity component for task α in particle i at time 0
$p\Pi i$	Personal best position of particle i
$g\Pi$	Global best position among all particles
ω	Inertia weight controlling exploration and exploitation
$\varkappa 1$	Acceleration coefficient guiding towards personal best

ϰ2	Acceleration coefficient guiding towards global best
ϱ1	Random value between 0 and 1 influencing personal experience
ϱ2	Random value between 0 and 1 influencing global experience
Λ	Parameter controlling the decay rate of inertia weight
Σ	Parameter controlling the spread of inertia decay
s	Total number of available servers
K	Total capacity of the servers
Prob_ 1	State probability for the first task in the system
P_0	Probability that the system is empty
Prob_ n	State probability when n tasks are in the system
n	Number of tasks in the system
pBest _i	Personal best position of particle i
gBest	Global best position across all particles
c_ 1	Acceleration coefficient towards personal best
c_2	Acceleration coefficient towards global best
r_ 1	Random variable guiding cognitive component
r_2	Random variable guiding social component
ηi	Energy efficiency coefficient of resource i

QPMHO initializes particles as task assignment candidates, each influenced by past performance and global trends. These particles evolve via velocity and position updates to minimize energy use, factoring in delay probabilities. Each device uses kernel-based SVM to map local data into higher-dimensional space for effective separation. A convex optimization balances regularization with hinge loss, ensuring accurate local models without sharing raw data. The kernel handles complexity without explicit transformation, supporting efficient training. Each device produces a model reflecting its unique data, preserving privacy and reducing transmission.After local training, model parameters are sent to a central aggregator for weighted combination based on dataset sizes. This ensures that models trained on more data influence the global output more heavily. The aggregation aligns with a global convex objective, leading to convergence under typical assumptions. To lower communication load, intermediate edge servers first collect and average updates from local devices. These regional models are then sent to the cloud for final aggregation, balancing scale and precision. Each level applies weighted averaging to maintain fairness based on data volume.

I. Queuing Probability Based Meta-Heuristic Optimization (QPMHO)

This phase introduces a metaheuristic approach based on swarm intelligence to optimize task-to-resource assignment in

edge- cloud environments. The design employs a particle swarm mechanism where each particle represents a candidate scheduling policy. The key focus is to minimize energy consumption while accounting for queuing delay probabilities associated with offloading tasks to computational nodes.

Phase Description:

B. Particle Position Vector Initialization

Each particle ϑ_ι at time $\tau=0$ is initialized with a position vector:

$\xi_\iota(0)= [\xi_{\iota\alpha}(0), \xi_\iota\beta(0), \ldots, \xi_\iota\zeta(0)]\in RT$

where $\xi_{\iota\gamma}(0)$ signifies the task assignment state of task τ_γ in particle ϑ_ι.

C. Initialization of Velocity

The corresponding velocity vector is given by:

$\nu_\iota(0)= [\nu_{\iota\alpha}(0), \nu_\iota\beta(0), \ldots, \nu_\iota\zeta(0)]\in RT$

This vector determines the directional movement in the solution space across iterations.

Step 2: Fitness Evaluation Based on Energy Efficiency

The energy-aware fitness function $F(\xi_i(\tau))$ is used to assess each particle's performance:

N * *

$\xi i(\tau)$ i=1

π_i =arg min$F(\xi_i(\tau))$, π_g =arg min$F(\xi_i(\tau))$ *

π_i is the best solution identified by particle i.

π is the best global solution found among all N particles at iteration τ.

Let me know if you'd like this in LaTeX code or embedded in a document.

Step 3: Adaptive Best Update Mechanism

Update rules at each iteration are formulated as:

$$\pi_i^*(\tau)= \begin{cases} \xi_i(\tau), & \text{if } F(\xi_i(\tau))<F(\pi_i^*(\tau\text{-}1)) \\ \pi_i^*(\tau\text{-}1), & \text{otherwise} \end{cases}$$

$\pi(\tau)$=arg min$F(\pi(\tau))_\iota$

Step 4: Evolution of Velocity and Position

Velocity Update Equation

Each particle velocity is updated according to:

$\nu_\iota(\tau+1)= \omega\nu_\iota(\tau)+\varkappa 1\ \varrho 1\ (\pi\text{-}\xi_\iota(\tau))+\varkappa 2\ \varrho 2\ (\pi\text{-}\xi_\iota(\tau))$

Where:

$\varkappa 1, \varkappa 2>0$ are cognitive and social influence coefficients,

$\varrho 1, \varrho 2$ $U(0,1)$ are uniform random variables.

$\omega\in(0\ ,1)$ is the inertia weight,

~

Inertia Weight Scheduling

The inertia weight decays exponentially:

$$\omega= \omega_{\max}\cdot\exp\left(-\frac{\Lambda^2}{2\Sigma^2}\right)$$

with Λ being the iteration index, and Σ a decay rate controller.

Position Update Rule

Updated position vector:

$\xi_\iota(\tau+1)= \xi_\iota(\tau)+\nu_\iota(\tau+1)$

To maintain feasible solutions, projection or rounding functions may be applied post-update.

Step 5: Termination Criterion

Algorithm halts if one of the following is satisfied:

$F(\pi(\tau))\leq\epsilon$ or $\tau\geq\tau_{max}$

Otherwise, the procedure continues iteratively.

Step 6: Final Output and Solution Representation

Output is expressed as:

* *

Final optimal task assignment vector $\xi = \pi_g$

Corresponding minimum energy cost $F(\xi^*)$

Mathematical Convergence Properties

The fitness function F is bounded below.

. The velocities $\nu_\iota(\tau)$ diminish over time due to decreasing ω.*

. The local optima π_ι stabilize asymptotically.

. The global optimum π_g approaches ξ as $\tau \to \infty$ under smooth fitness landscapes.

. Convergence is probabilistically guaranteed under bounded randomness and contraction mapping.

0.1. Fruitfly Algorithm for Task Scheduling

The Fruitfly Algorithm (FA) is an optimization algorithm inspired by the foraging behavior of fruitflies in nature, so an application of FA to complex problems is scheduling problems, mainly in task scheduling in cloud computing environment. The basic procedures for using the Fruitfly Algorithm to perform task scheduling in CloudSim can be described as follows:

Specify the Fitness Function:

The main purpose is to express a fitness function to minimize the makespan and maximize other relevant parameters.

Create an Initial Population:

Initialize the population of candidate schedules. The schedules may be randomly generated or obtained by some heuristics. Each schedule encapsulates a set of tasks, with each task having a resource and an execution time.

Select:

From the population, select a collection of schedules to be parent schedules for the next generation.

Crossover:

Use the crossover operator to combine parent schedules to create new child schedules for the next generation. In task scheduling, crossover is the process to swap some tasks between two schedules to generate new and interesting schedules.

Mutation:

Apply random changes to the schedules using a mutation operator to visit new places in the search space. In task scheduling, mutation could also include alterations of start times or resource allocations of each task.

Selection and Termination:

Repeat the choosing operation so that the process of crossover, mutation, and fitness evaluation can continue until a specified end condition is fulfilled.

As cloud computing is becoming the trend, the best strategies to effectively manage cloud resource are worth investigating. What has to be done is the initialization, task distribution, scheduling of CloudSim along with resource limitations.

The initialization procedure is the first phase for the creation of a cloud environment, addressing a variety of settings related to the virtualization infrastructure. This step is crucial because the quality of the cloud environment and the performance of the running applications are both determined by it. CloudSim enables such an easy initial setup of the cloud environment by defining the properties of the datacenter, host, and VM with basic parameters.

The cloud property of initialization parameters can be used to specify features, e.g., the number of hosts, CPU power, or type of VMs, in the cloud. The normal processing of the initialization parameters will ensure good performance, which in turn minimizes resource wastage.

An important issue in cloud computing is task scheduling, which is related to how to assign the computing tasks to the virtual machines (VMs) in a datacenter. It is about making use of resources as fully as possible, with least amount of delay and wastage of resources. Task allocation strategy for VMs should take into account the processing capability of each VM, the workload of datacenter and the resource constraints of the datacenter.

The resource limitations can be such as power consumption, network bandwidth and storage space and so on. By performing better task allocation, the cloud infrastructure is expected to support a higher service quality with less energy

consumption and shorter application execution time.

The cloud environment with its performance is determined by the controlling operation of CloudSim parameter scheduling. Scheduling consists in deciding when each task will be executed, and which resources will be allocated to them. Tasks need to be scheduled in a maximum resource utilization, minimum latency and minimum conflicts among them.

Scheduling also contributes to the cloud infrastructure working seamlessly in high loads of work. It is also required to apply an effective scheduling algorithm that takes into consideration the resource limitations and workload balancing across the candidate VMs.

Resource limitation is a significant issue in cloud computing, which need to be taken into account in the initialization, task assignment, and scheduling of CloudSim parameters. CloudSim offers methods to have the infrastructure layer of the cloud being constrained to the available resource while reducing energy and the execution time.

It is important to define realistic resource constraints which are compatible with the capacity of the cloud environment and the needs of the applications.

The datacenter is used to offer processing power to the users that need it. Used, Data Transferred, and Run Length of the simulation.

0.2. Ant Colony Optimization for Task Allocation and Scheduling

The Proposed Ant Colony Optimization (ACO) algorithm is a metaheuristic technique based on the foraging behavior of ant colonies. This is how the ACO is applied in the task allocation and scheduling in cloud computing.

- Initialize VMs, Tasks, Ants.
- For each task in task list.
- Randomly select k-tasks as initial clusters for m datacenters.
- For each neighbor task in kth datacenter.
- For each task t in datacenter D.
- Initialize ants, VMs, cluster tasks in datacenter.
- Repeat Steps 2--6 until termination criteria is satisfied.
- Perform heuristic probability to each task pheromone.

Pheromone is a chemical substance released by ants to communicate with each other. Under the ACO algorithm, pheromone is utilized to encode information shared among ants about the best task allocation and scheduling solutions they have found.

The ACO algorithm calculates the heuristic probability for each task pheromone. The heuristic probability reflects the quality of task allocation and scheduling solutions found so far. The higher the quality of the solutions, the higher the heuristic probability.

The heuristic probability is employed to update the pheromone amounts on each job. The level of pheromone on the tasks assigned to the VMs with the highest fitness values is raised by the ACO technique. Conversely, the pheromone levels of tasks assigned to VMs with lower fitness values are reduced.

Using heuristic probability to update task pheromone levels allows the algorithm to gradually converge toward optimal task allocation and scheduling solutions. The ants follow pheromone trails and become more likely to select superior task allocation and scheduling solutions.

The final step of the ACO algorithm for task allocation and scheduling in cloud computing is to calculate the mutual task importance score for all the tasks in the task list.

Experimental Results

Experimental result is simulated in Java based CloudSim. There are 5 datacenters and 2 hosts per datacenter for the given configuration, which means that the total number of hosts is 10.

A cloudlet is a job in cloud computing and represents a task or unit of execution. The

number of cloudlets or jobs to be run for given configuration lies between 200 and 3600 and such a range can be used to model different kinds of cloud computing scenarios from small-scale applications to large-scale applications.

A broker is an entity that hosts the cloud environment and deals with the users to the datacenter. In this case, there is only one broker managing the entire cloud system. It is the broker that owns tasks.

The broker manages the process of receiving, handling, processing requests of users, resource allocation, and schedules task on hosts.

The datacenters, hosts, cloudlets, and brokers configuration is essential in enabling cloud applications to be properly managed and executed efficiently. By tuning these parameters the cloud service provider can improve the efficiency of resource usage, the latency of response and the overall performance, thus providing an efficient and dependable cloud computing environment for end-user.

In cloud computing, a host is a physical computer on which a virtual machine (VM) is run. A host can host multiple VMs.

The minimal host configurations have the following parameters: maximum number of VMs, RAM size, MIPS number of the processor, storage size, and maximum bandwidth.

Each host can host up to 80 VMs for the configuration provided. The amount of RAM for each host is by default 20480 MB. The CPU speed of each host is 5000 MIPS. The storage size of each host is 1,048,576 MB. The initial bandwidth of the host is set to 500,000 MB/s.

- **Initialization:** Creation of datacenters, VMs and cloudlets.
- **Task Assignment:** Allocation of cloudlets to VMs.
- **Scheduling:** Determination of cloudlet execution order.
- **Execution:** Running the simulation and collecting performance statistics.

To execute these components in Java, the CloudSim API provides classes and methods for modeling and managing cloud infrastructure as well as executing cloudlets on VMs. The CloudSim API provides predefined classes such as Cloudlet, Datacenter, DatacenterBroker, and VM through which the components of a cloud infrastructure and application can be represented.

For instance, to create a Datacenter, a new instance of the Datacenter class is created, its attributes such as cpuCapacity and storageCapacity are specified, and then it is added to the simulation using the CloudSim class. Similarly, to create a Cloudlet, a new Cloudlet object is instantiated, its parameters such as length and priority are specified, and then it is added to a DatacenterBroker, which handles the distribution of cloudlets among VMs.

1) Average Load Resources per Algorithm

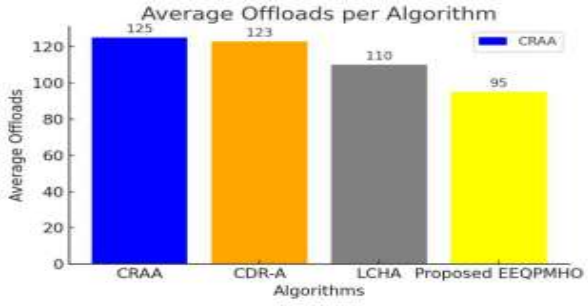

Fig.3 Average task offloads for mobile -edge- cloud processing

2) Average Ofloads per Algorithm

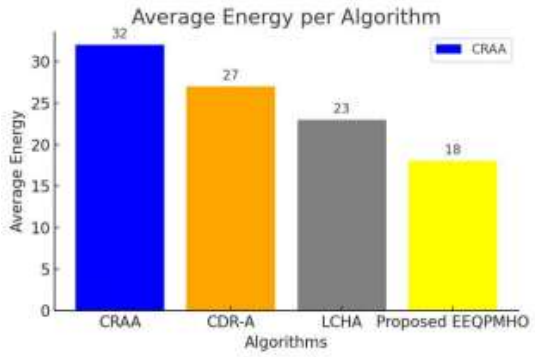

Fig. 4 Average energy for mobile -edge processing

3) Average Energy per Algorithm

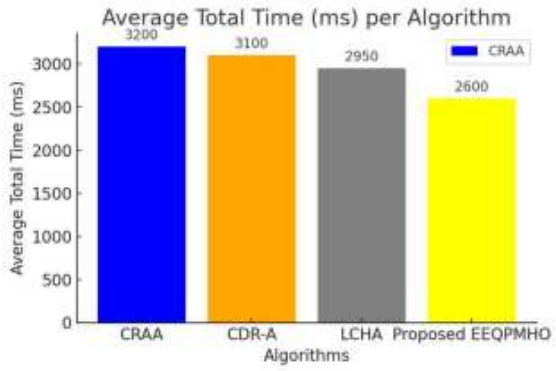

Fig. 5 Average total runtime for mobile - edge processing

Conclusion

Cloud based workload scheduling is of importance among the real-time cloud based applications. As most of the traditional methods cannot schedule the tasks among various cloud instance types, the critical resource must be identified and assigned to each virtual machine by means of the enhanced meta-heuristic approach.

Traditional methods employ static metrics to characterize the task allocation problem under available resources. In this paper, a Cluster-based Energy Efficient Q-Ant Colony Optimization (CEQACO) approach is proposed for cloud computing environment. The proposed scheme applies clustering algorithm to cluster cloud virtual machines (VMs) according to their workload activities, and then utilizes a Q-Ant Colony Optimization algorithm to find the optimal solution for assigning VMs to physical servers.

The results highlight that the CEQACO approach can reduce time computation and energy up to 6\% while satisfying the quality service requirements of cloud users.

In light of sustainability and energy efficiency considerations, a possible direction of research would be to design cloud task scheduling models which enhance resource allocation with consideration of energy consumption. This may involve researching methods to modify resource allocation dynamically in response to energy consumption levels, or by taking into account energy consumption as a parameter in task scheduling decisions.

References

M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," Journal of Network and Computer Applications, vol. 143, pp. 1–33, Oct. 2019, doi: 10. 1016/j.jnca.2019.06.006.

M. Hussain, L.-F. Wei, F. Abbas, A. Rehman, M. Ali, and A. Lakhan, "A multi-objective quantum-inspired genetic algorithm for workflow healthcare application scheduling with hard and soft deadline constraints in hybrid clouds," Applied Soft Computing, vol. 128, p. 109440, Oct. 2022, doi: 10. 1016/j.asoc.2022. 109440.

K. Dubey and S. C. Sharma, "A novel multi-objective CR- PSO task scheduling algorithm with deadline constraint in cloud computing," Sustainable Computing: Informatics and Systems, vol. 32, p. 100605, Dec. 2021, doi: 10. 1016/j.suscom.2021. 100605.

M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," Egyptian Informatics Journal, vol. 16, no. 3, pp. 275–295, Nov. 2015, doi: 10. 1016/j.eij.2015.07.001.

S. A. Murad, A. J. M. Muzahid, Z. R. M. Azmi, M. I. Hoque, and M. Kowsher, "A review on job scheduling technique in cloud computing and priority rule based intelligent framework,"

Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 6, Part A, pp. 2309–2331, Jun. 2022, doi: 10.1016/j.jksuci.2022.03.027.

M. Gill and D. Singh, "ACO Based Container Placement for CaaS in Fog Computing," Procedia Computer Science, vol. 167, pp. 760–768, Jan. 2020, doi: 10.1016/j.procs.2020.03.406.

H. Xing, J. Zhu, R. Qu, P. Dai, S. Luo, and M. A. Iqbal, "An ACO for energy-efficient and traffic-aware virtual machine placement in cloud computing," Swarm and Evolutionary Computation, vol. 68, p. 101012, Feb. 2022, doi: 10.1016/j.swevo.2021.101012.

M. Alaei, R. Khorsand, and M. Ramezanpour, "An adaptive fault detector strategy for scientific workflow scheduling based on improved differential evolution algorithm in cloud," Applied Soft Computing, vol. 99, p. 106895, Feb. 2021, doi: 10.1016/j.asoc.2020.106895.

S. G. Domanal and G. R. M. Reddy, "An efficient cost optimized scheduling for spot instances in heterogeneous cloud environment," Future Generation Computer Systems, vol. 84, pp. 11–21, Jul. 2018, doi: 10.1016/j.future.2018.02.003.

M. Abdullahi, M. A. Ngadi, S. I. Dishing, S. M. Abdulhamid, and B. I. Ahmad, "An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment," Journal of Network and Computer Practice and Theory, vol. 110, p. 102323, Jul. 2021, doi: 10.1016/j.simpat.2021.102323.

J. Wen, J. Yang, T. Wang, Y. Li, and Z. Lv, "Energy- efficient task allocation for reliable parallel computation of cluster-based wireless ensor network in edge computing," Digital Communications and Networks, Jun. 2022, doi: 10.1016/j.dcan.2022.06.014. Applications, vol. 133, pp. 60–74, May 2019, doi: 10.1016/j.jnca.2019.02.005.

A. Talha, A. Bouayad, and M. O. C. Malki, "An improved pathfinder algorithm using opposition-based learning for tasks scheduling in cloud environment," Journal of Computational Science, vol. 64, p. 101873, Oct. 2022, doi: 10.1016/j.jocs.2022.101873.

N. Sharma, Sonal, and P. Garg, "Ant colony based optimization model for QoS- based task scheduling in cloud computing environment," Measurement: Sensors, vol. 24, p. 100531, Dec. 2022, doi: 10.1016/j.measen.2022.100531.

G. J. Samuel Babu and M. Baskar, "Application of blockchain methodology in secure task scheduling in cloud environment," Advances in Engineering Software, vol. 172, p. 103175, Oct. 2022, doi: 10.1016/j.advengsoft.2022.103175.

G. Rjoub, J. Bentahar, and O. A. Wahab, "BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments," Future Generation Computer Systems, vol. 110, pp. 1079–1097, Sep. 2020, doi: 10.1016/j.future.2019.11.019.

S. Vishwakarma et al., "Cloud data storage with improved resource scheduling in healthcare application based on security system," Optik, vol. 272, p. 170225, Feb. 2023, doi: 10.1016/j.ijleo.2022.170225.

A. M. Yadav and S. C. Sharma, "Cooperative task scheduling secured with blockchain in sustainable mobile edge computing," Sustainable Computing: Informatics and Systems, vol. 37, p. 100843, Jan. 2023, doi: 10.1016/j.suscom.2022.100843.

X. Wang et al., "Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning," Journal of Manufacturing Systems, vol. 65, pp. 130–145, Oct. 2022, doi: 10.1016/j.jmsy.2022.08.004.

S. Mangalampalli, G. R. Karri, and G. N. Satish, "Efficient Workflow Scheduling algorithm in cloud computing using Whale Optimization," Procedia Computer Science, vol. 218, pp. 1936–1945, Jan. 2023, doi: 10.1016/j.procs.2023.01.170.

R. Medara, R. S. Singh, and Amit, "Energy-aware workflow task scheduling in clouds with virtual machine consolidation using discrete water wave optimization," Simulation Modelling