RESEARCH PAPER

A Dynamic Task Scheduling Model for Efficient Resource Management in Large-Scale Cloud Environments

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

Abstract—Cloud computing systems usually have to deal with a vast number of tasks to be executed on distributed resources. One key challenge in such systems is how to allocate resources and schedule cloudlet effectively among many tasks that come simultaneously. The quality of a cloud platform's performance is dependent on how well it is able to allocate these jobs to its virtual machines. Good schedule results in balanced workload, better responsiveness of the system, and multiple cloudlets could run concurrently without waiting for each other.Scheduling tasks is the process of associating user requests or application tasks to virtual machines in a manner that enhances execution and resource utilization. To minimize total completion time, enhance system throughput and avoid resource congestion, a good scheduling strategy should be developed. But this task is hard to compute. Combinatorial scheduling is usually thought to be NP-hard because the space of candidate solutions can increase exponentially with the number of tasks and computing resources. As tasks (n), for execution, are to be assigned to (m) resources the number of different possible schedules rapidly increases rendering the identification of the best schedule hardly feasible in limited time. To deal with above challenge, an efficient scalable scheduling framework is proposed in cloud. The proposed framework clusters the virtual machines based on their workload and behavior. This categorization step contributes to further reshape the resource pool, and also confines the searching space for scheduling. Once VMs are clustered, a Q-ant colony optimization method is used to find suitable allocations among tasks, VMs, and physical servers.The process of optimization simulates cooperative search behavior such as ant colonies, which the system can discover efficient task allocation routes gradually. With iterating over evaluation and adjustment, the algorithm refines its scheduling choices to achieve better resource balance in the cloud environment.Simulation results show that our proposed model can enhance the efficiency of the cloud environment. The complex results show that about 6% decrease in computational overhead and energy consumption of the cloud compared with the state-of-the-scheduling algorithm. But at the same time, the framework preserves necessary quality of service properties that the cloud users are accustomed to, since the completed tasks will have acceptable response times and the resource usage over infrastructure is relatively balanced.

Keywords: Quality of service, cluster based task scheduling , cloud servers, cloud scheduling.

How to cite this article: Rupasri M, Varma GPS, Indukuri H. A Dynamic Task Scheduling Model for Efficient Resource Management in Large-Scale Cloud Environments. Int J Drug Deliv Technol. 2026;16(6s): 615-627; DOI: 10.25258/ijddt.16.6s.87

I. INTRODUCTION

Cloud computing is a type of computing that allows for on-demand delivery of computing resources, such as processing and data storage, over the Internet. Companies turn to cloud infrastructure to handle compute-intensive work without investing in costly on-premises hardware. The service providers take care of the infrastructure and they load balance between various servers to ensure that the user application has sufficient resources to run. Stable performance is, in fact, one of the most

important issues when running an environment like this. Load balancing works to ensure that any given node will not be overwhelmed by distributing the incoming request load across multiple servers or computing nodes. With a more balanced distribution of tasks, the total execution time decreases, the system maintaining better responsiveness, and the hardware resources being utilized more efficiently.Processing load balancing considerations on cloud platforms are becoming complicated as numerous users are submitting tasks at the same time. Different tasks may have different processing time, memory, storage, or network bandwidth requirements. An appropriate scheduling or load balancing policy in this sense needs to determine how such tasks are to be allocated to the computing resources[1]. Efficient algorithms tries to balance the load among all the nodes such that all the processors are busy and none is idle or overloaded. In cloud computing, however, the tasks are typically broken down into a number of small parts and each part is executed concurrently on separate machines. This parallel processing technique enhances the computational efficiency and also the system throughput. Static load balancing techniques rely on information of the system behavior, application patterns and resource capabilities, which are collected offline. These techniques can determine the processor capacity and available resources prior to the execution, which helps the scheduler to schedule tasks by considering the anticipated workload patterns.Virtualization technology underpins the cloud infrastructure. It meets someone's physical computing needs such as processors, memory, storage, and network interfaces, by making them feel like virtual resources that are consumed in a self-service manner by various users. Each physical resource is given a logical identity with which the cloud platform can associate user requests to physical hardware elements. Virtual machines provide a sandboxed environment, so they behave as separate machines even though they run on the same physical hardware. This level of abstraction enables a lot more flexibility and scalability for cloud providers to dynamically pool resources based on the fluctuations in demand. While handling multiple requests from different users, the virtualization software can also easily manage the allocation of VMs to appropriate physical servers, achieving optimal load balancing.Various types of virtualization are employed in cloud environments to virtualise different infrastructure components. Network virtualization enables a number of virtual network interfaces on a single physical network, and supports centralized monitoring and simplified management. Data virtualization allows applications to access data from a number of different storage locations without users needing to know the underlying structure of the data. Authorized systems can retrieve such data, if required, in the same format in which it is physically stored, no matter where it is residing physically. Central to virtualization is the hypervisor, sometimes known as the Virtual Machine Manager. It governs the hardware resources and it runs multiple operating systems on the same machine concurrently. Typical modern cloud platforms have automatic scaling solutions based on metrics such as queue length, waiting time, and workload size. When the system detects that the load is rising, it will spin up the additional virtual machines to process the tasks as fast as possible. When the load ebb, waste resources are released to reduce operation cost and energy consumption[2-6].

An excellent metric to assess how well cloud systems provide computing services is how much of the riverside water they can draw to fill their pool. QoS is a system level performance measure, which include latency, packet loss, throughput, reliability and service availability. The latency is the delay that a request experiences to travel to a server and back. Packet loss is defined as the percentage of packets sent that never reach the destination. Throughput is the number of bytes that are successfully processed in a given time. A good throughput means a good data processing ability. Availability is the fraction of time a service is up and running and available to users. Reliability is a measure of the ability of a system to function properly without failure. Strong QoS parameters can be achieved only under the ideal combination of resource scheduling and allocation[7].

Clustering methods are commonly applied to group computing resources in the cloud environment. Servers or VMs with similar properties can be organized into clusters. This way, the scheduling algorithm does not need to look at all the infrastructure, but only the clusters to find the best fit cluster for the task. Clustering minimizes scheduling complexity and enhance response time. The performance is better and the energy consumption is lower when tasks are assigned to clusters that are compatible with their workload. To analyse task scheduling methods and test novel scheduling strategies, simulation platforms are often adopted. Among them, CloudSim is a very popular simulation environment. CloudSim is a simulation-based tool designed for modeling and simulation of cloud computing environments. This enables the researchers to investigate data center, virtual machines, task scheduling policies, and inter-networking without the need of real cloud computing platform such as Amazon EC2. Via simulation, different scheduling algorithms may be evaluated on even playing fields. CloudSim is a cloud infrastructure simulation tool Schema including multiple layers with high use case in the following Layer1 with Data center, Hosts, Virtual Machines (VM), Cloudlets, Network and Layer2 with user resources, billing and application. A data center is a physical building filled with computing servers, storage devices, and networking hardware. Each server within the data center is

called a host. Hosts have CPU's, RAM, storage, and network bandwidth that can be divided and assigned to virtual machines. Virtual machines consume user applications and perform compute jobs. A cloudlet is a single task that has been submitted from a user application. It includes details like the length of the task, the resources it needs, and its priority for execution.CloudSim also includes models of network communication between cloud entities. Such models capture the network bandwidth and the communication latency, as well as the traffic matrices within the infrastructure. By studying network idle states, communication performance bottlenecks that could deteriorate scheduling efficacy could be also identified by researchers. Resource provisioning policies in CloudSim resemble the policies of distributing workload to virtual machine (VM) in the actual scenario. These models enable analisar que scheduling decisions impact system performance, energy consumption and response time.With CloudSim simulatiions, you may apply different scheduling policies in the same infrastructure and test them. Researchers can evaluate system parameters such as makespan, execution time, resources utilization, energy consumption and cost of operation. These analysis have been used to assess the performance scheduling method under different types of workloads. Such simulation studies play a vital role in prediction of efficient work flow scheduling strategies prior to realization in actual cloud environment, by which developers can design energy-aware resource management mechanism with higher efficiency. The resource allocation policies in CloudSim depend on how the resource owner distributes the available resources among the submitted application according to its computational requirements and the current status of the system. These models take into account multiple parameters such as the priority of the task, capacity of resources, execution length and preferences of the users prior to the mapping of the task to the virtual machine. The goal is to pair each job's computational requirements with the correct resource in that way the system performance as a whole is maximized. Consequently, by intelligently performing such allocations, the system can prevent resources from becoming congested, keep the processors from sitting idle, and accomplish tasks more effectively. The idea of this allocation mechanism could be attributed, which user tasks submitted to the system are first examined by the scheduling phase and then bind to suitable virtual machines in different data centers.

CloudSim provides an experimental toolkit for simulating cloud computing infrastructures and to investigate the behavior of scheduling algorithms in a controlled environment. Based on this platform, it is quite simple to build models of data center, host, virtual machines, network communication, and task execution flow. Such simulations help businesses analyze various resource management schemes without investing in expensive real-world cloud infrastructure. By a series of experiments, the researchers in [38] could find the policies of scheduling for consum- ing resources that can minimize the costs of operations and at the same time provide a stable service performance.In real Cloud computing systems, it is common that multiple tasks are submitted at the same time and each task has different computational and execution priority requirements. Multi-task scheduling deals with the problem of efficiently and fairly allocating a set of heterogeneous tasks on a given set of resources. The CloudSim simulator supports a comprehensive and customizable simulation of cloud computing, enabling the simulation and comparison of different scheduling algorithms upon these complex task sets. Scheduling and optimization techniques are generally employed to find the good schedules, which minimize the total execution time and maximize the resource utilization.PSO and GA are among the most famous optimization method utilized for multi-task scheduling among the most prevalent solvers. These methods are based on the observation of naturally occurring processes and collective behaviors of biological systems. They consider a large number of potential schedules and move iteratively towards an improved solution. This initia- group of candidate task schedules. Genetic Algorithm produces an initial population of candidate task schedules. Each candidate corresponds to a possible mapping of tasks to resources. In subsequent iterations, the algorithm calculates the quality of each schedule according to an evaluation function, representing the time needed for the execution of all the tasks and the efficiency of the system. TaBest schedules are then merged and are reproduced for enhanced solutions in next generation. Eventually, the population of schedules evolves and converges to scheduling solutions with better fitness, resulting in minimizing the total execution time[8].

In addition to evolutionary optimization techniques, a number of other methods are applied to the problem of multi-task scheduling in cloud environments. Heuristic algorithms are based on predefined rules during the process of task assignment. These methods do not explore the whole solution space in search of the optimal ordering, but attempt to get a good ordering as quickly as possible. Linear programming exploits formal mathematical expressions relations among tasks and resources to model the scheduling problem. Optimization rules are then applied to find the most appropriate allocation pattern that fulfills the system constraints. We call this Branch-and-bound techniques solve the problem in a different way by enumerating smaller subsets of solution space. Each portion is pursued, and those that cannot yield an improved solution pruned away, enabling the algorithm to concentrate on fruitful scheduling possibilities[9].

Particle Swarm Optimization is also a bio-inspired approach widely applied to tackle scheduling problems in cloud computing system. This technique simulates the behavioral patterns of cooperation among the flocks of birds or schools of fishes. In the scheduling scenario, a particle corresponds to an feasible schedule of tasks. Such particles^ are capable of moving in the space of solutions by changing their position according to their historical performance as well as the performance of the neighbouring particles. An evaluation function in the CloudSim system evaluates the effectiveness of every schedule which is based on the makespan mx the total time to complete all tasks or resource utilization. Iteratively, particles share information on good scheduling solutions to the algorithm, Shimizu and Hoffmann [36] enabled better new task assignment solutions to discover.

Genetic Algorithm applies a similar improvement procedure iteratively, but instead of deterministic rules it uses evolutionary manner which means mechanisms of reproduction, variation and survival of the fittest solutions. Initially, the scheduling procedure is started with an initial random population of schedules where each individual represents a feasible mapping of tasks to resources. Its execution efficiency is a measure of quality of each schedule. A pool-based genetic algorithm utilizing a selection process is implemented to select and promote good schedules to generate the other schedules of the succeeding generation. During recombination, subsequences of two chosen schedules are exchanged to generate new schedules patterns. Mutation For maintaining diversity among solution candidates and to avoid premature convergence, mutation procedure randomly makes small changes in the task assignments. This iterative process of evaluation, selection, recombination, and variation continues until the predetermined stop criterion is met or a satisfactory scheduling solution, that minimizes execution time and maximizes system throughput, is identified.

The paper is structured as follows: In Section 2, the related works of cloud task scheduling models and their limitations are presented. Section 3 outlines the proposed solution for cluster-based cloud task scheduling. Section 4 provides details on the experimental results and analysis. Finally, in Section 5, the paper is concluded.

II. RELATED WORKS

Task scheduling algorithms are essential for the performance of cloud computing systems. These algorithms seek to allocate the tasks to the resources so that the makespan of the execution is minimized and resources are effectively utilized. In most scheduling methods, the algorithm starts with an initialization step, during which the system parameters, tasks, and resources are ready. After initialization, scheduling operations are performed by means of stages that consider task allocation and enhance scheduling decisions. They can include analyzing task dependencies, mapping tasks to appropriate virtual machines, or even at the level of schedules through iterative refinement. The process is iterated until a termination criteria is met, e.g., achieving a stable scheduling behavior or all task assignments are made. In this process, a variety of performance measures including average execution time and system utilization are utilized to evaluate the performance of the scheduling algorithm as a whole[10].

The growing popularity of cloud computing has brought scheduling research, among other things, into the field of cloud manufacturing. In such scenarios, manufacturing is aligned with digital platforms and production tasks are managed via centralized cloud systems. Traditional distributed planning approaches are also increasingly being replaced by centralized control paradigms with the production resource, machine details, and operation data kept in shared cloud databases. This facilitates the better coordination of intricate manufacturing activities by manufacturing systems. Cloud scheduling systems have the ability to evaluate production workloads, assign tasks among the resources, and respond to evolving operational needs more flexibly.

Recently, the attention of study shifted towards hybrid and nature-inspired optimization techniques to tackle scheduling problems in cloud. These methods try to conduct efficient searches over large solution spaces by simulating processes from biology or nature. One such scheme is to combine Differential Evolution (DE) with a novel enhanced Moth Search (MS) algorithm. The algorithm is based on the natural behavior of the moths that are attracted toward light sources. In terms of computation, this behavior also helps to keep the algorithm focused on potentially good solutions in the scheduling stage. By applying the two optimization algorithms, the scheduling process can be completed in a timely manner, especially when dealing with multitask execution over a pool of VMs.

Energy efficiency is another critical issue resulting from the growth of cloud computing infrastructures. Virtual machines are also important in data center energy efficiency. It was virtualization that enabled us to run

multiple virtual machines on the same physical server instead of dedicating physical hardware to single applications. This consolidation also improves hardware utilization, enabling an organization to reduce how many physical machines are active in doing the work. Energy consumption reduces, operation cost reduces and the workload is more evenly distributed in these infrastructures.

Another aspect of energy management is the management of the operational states of virtual machines. In this framework, a VM can hold one of multiple states, such as active, idle and sleep. On active machines computations are done at full speed, idle machines are not executing tasks but still running, and sleeping machines are momentarily shut down until new workloads come in. Efficiently controlling these states enables cloud systems to drastically decrease unwarranted energy usage. As workloads diminish, some machines can move into idle or sleep modes, conserving energy until they are needed again.

Nevertheless, the resource provisioning in the cloud datacenters should be managed by the book. Too many virtual machines--that is, guest operating system instances--running on a single physical server leads to contention for processing power, memory, and network bandwidth. This rivalry can impact the performance of the system and slow down the execution of tasks. Conventional vM placement heuristics consider that each VM is assigned to a fixed set of resources, in the form of stable CPUs in the data center. The aim of such techniques is to get the best equalization of resources in a physical machine that will not cause poor performance. Resource allocation policies strive to allocate tasks to computing resources that are neither over- nor under-utilized. A recently proposed scheduling paradigm employs pre-emptible virtual machine instances to enhance resource utilization. In this scheme, some of the virtual machines may be pre-empted if there are urgent-high priority jobs that must be processed immediately. This enables the cloud service providers to better utilize resource, especially for some of the workloads that can be interrupted in the process. Commercial entities have more fully utilized resources and increased profitability, and research institutions and the non-profit sector have access to flexible computing resources[11].

Batch systems sometimes employ backfilling policies (OPT also implements backfilling) in an attempt to increase system utilization. Backfilling permits execution of smaller jobs in slots that would otherwise be wasted waiting for larger jobs to begin. This technique enhances the system throughput and the processor-busy time. Pre-emptible instances and backfilling based scheduling frameworks enable flexible workloads and lead to new pricing models for cloud service providers.

Load balancing and remain one of the essential guarantee stable performance in distributed cloud environments. It divides up the incoming work among a collection of nodes, so that no single node gets overwhelmed. A uniform distribution of the load among processors leads to a better system response time and high resource utilization. An equilibrium load distribution between these mechanisms processing on different nodes of a distributed system ensures that the system can serve more user requests with no bottleneck in performance. Furthermore, at load balancing also contributes to fault tolerance by reassigning tasks to other resources should a computing node go down.

There are multiple load balancing policies which are applied to the cloud environment. Static load balancing algorithms assign the load based on some predetermined conditions which are related to resources availability rather than any other computation. These techniques do not take into account the dynamic changes in the system, but are easy to apply and have low computational burden. One popular static method is the round-robin scheduling algorithm in which the tasks are distributed to the VMs in rounds. In this way, each machine has equal chance to get workloads and the amount of tasks allocated per each machine are consistent.

Different workflow allocation schemes are proposed to enhance the task placement in cloud platform. A few of these techniques concentrate on minimize the time required to process the task and the cost of the service without increasing the energy spent. Good work flow allocation facilitates flexible execution of a task, and it also maximizes the resource availability to improve infrastructure utilization. The system can maintainif higher productivity levels when the scheduling systems of the system can detect free resources and assignwork accordingly.

Ant Colony Optimization is the new one promising technique to solve workflow scheduling problem in cloudcomputing. This behaviour is simulated by this technique when antslooking for food. In computational models ofscheduling, an artificial ant explores a candidate task allocation and communicates good candidates not only through pheromone but also through…

This is done by reinforcing good scheduling paths after each iteration, the algorithm following this scheme as well then converges to good, and evenoptimal task allocation policies [1]. This leads to reduction in theprocessing time, balancing of workload and improvement of throughput.

Some of the existing works are based onoptimization concepts, which are generalized into the different scheduling procedures these arereported. Some of the works proposed static scheduling algorithm that take into account of task execution time and provision delay during the execution in cloud infrastructures. Other presentedSLA based scheduling schemes where decisions for the execution of tasks take into account the service requirements and the profit of the provider. Artificial Bee Colony (ABC) algorithms had also been utilizedto balance workload among virtual machines as well as reduce the overall runtime[12].

Scheduling schemes based on genetic algorithms have also been introduced for coping with dynamic cloud environment by monitoring resource utility and Runtime Performance. Cost of operation and time of processing, together with processor idle times, were minimizated by use of cat swarm optimization class based multi-objective optimization. Some other scheduling schemes also take into account multiple system parameters while making scheduling decisions, e.g. task queue length, communication delay, task execution cost, energy consumption.

In addition, further works investigate foraging optimization techniques such as firefly algorithms, to choose best machines according to their load status, genetic algorithm-based improvements to reduce system damage and cost of operation, particle optimization (PO) based dynamic dispatching systems that assigns work efficiently among nodes in cloud infrastructure. These approaches are to enhance system performance by a better use of resources, minimum computational delay and fast overall task execution in the context of large-scale cloud computing environment[13-15].

III. PROPOSED MODEL

In this paper, to enhance resource utilization and energy consumption in cloud, we propose a novel theorem called Adaptive Cluster-Guided Swarm Scheduling (ACGSS). This model applies a workload grouping strategy, an adaptive resource assessment strategy as well as a swarm- based scheduling strategy to raise task execution effectiveness in the cloud environments. The methodology comprises clustering, adaptive monitoring, and swarm-guided optimization for assigning tasks to virtual machines (VMs) with the objective of minimizing processing delay and balancing workload among the VMs.

The entire scheduling procedure in the cloud environment is envisaged to be covered by the following submodels of the proposed scheme.

Phase 1: Cloud Infrastructure Configuration

The initial stage sets up the simulation environment in CloudSim by defining core components of the cloud platform. Data centers, hosts, virtual machines, are created with custom properties such as processor speed, memory, storage, and network bandwidth. These parameters dictate how many operations the system can execute and how efficiently it processes them.

A correct setting in this step contributes to reproducing an operational running cloud environment which imitates real data centers. With the proper infrastructure in place, the hardware and software can scale to accommodate hundreds or thousands of users queries at stable performance and resource utilization.

Phase 2: Workload Profiling and Task Characterization In this stage the incoming user requests are mapped as cloudlets, which are computational job simulated in the system Each cloudlet includes details such as the length of the task, the processing capacity needed, the time completing expected, and the level of priority.

The system does workload profiling to estimate the computational need of each task. Similar task types can be extracted by analyzing the workload. This data is further applied to enable more intelligent scheduling options and to eliminate potential unnecessary computational overheads when selecting resources.

--- ## Phase 3: Resource Clustering and Capability Grouping Instead of considering all virtual machines separately, the proposed scheme clusters the available resources in terms of their computational and operational features. Virtual machines are classified as groups by homogeneous computational power, available memory and workload potential.

The task of scheduling is simplified, by narrowing the search region during the process of task assignment, through resource

clustering. When a task arrives in the system the scheduler first chooses the best cluster and then the best virtual machine. This kind of grouping can accelerate the scheduling process, as well as improve the utilization of resource for workload in the data center.

Phase 4: Dynamic Resource Assessment

Following the generation of clusters, the system assesses the state of individual virtual machines in the clusters. The parameters like CPU utilization, Memory usage, Network traffic, and Power consumption are continuously monitored.

This evaluation process enables the framework to decide if a resource can perform additional tasks without resulting in overloading. The scheduler does not assign new tasks to a virtual machine when it becomes heavily loaded. Considering resource status in real time, the proposed framework keeps the workload balanced and minimizes the performance degradation probability.

Phase 5: Swarm-Guided Task Allocation

1.2) We utilize a swarm-oriented strategy to search for feasible task-resource mappings in this stage. Potential scheduling patterns are explored in parallel by autonomous agents that work on various task and virtual machine pairings.

These agents exchange knowledge pertaining to good scheduling paths along the optimization. As the algorithm iterates, the trustworthiness of good task decisions is boosted and the algorithm progressively converges to better scheduling. This joint investigation allows the system to find good resource assignments efficiently without searching exhaustively over all possible schedules.

Phase 6: Adaptive Execution Scheduling

After finding the best resource for each task, the scheduler calculates the tasks' execution order on the virtual machines. Consideration is given to task priority, resources availability, and the status of the system load in the scheduling.

This stage aims for execution delay minimization and system throughput maximization By applying a suitable tradeoff between the two performance metrics. By dynamically scheduling the tasks, system does not have any conflicts between tasks or at any time, none of the CPU or I/O are kept idle (even when working load is on the rise).

Phase 7: Performance Monitoring and Adaptive Adjustment

Monitoring system performance in the task execution is the final step. These indicators are task finishing time, resource utilization, energy consumption and workload balance.

When it detects certain nonuniform scheduling behaviors or resource imbalances, it modifies the future task distributions accordingly. Such adaptive feedback strategy enables the framework to deliver stable performance even when the workload changes unpredictabl

Algorithm 1: Infrastructure Initialization and Workload Profiling

1: Initialize simulation environment S
2: Create data centers DC = {dc_1, dc_2, ... , dc_m}
3: For each dc_i ∈ DC do
4: Configure host set H_i = {h_1,h_2,...,h_k}
5: Assign CPU capacity C(h_j), memory M(h_j), bandwidth B(h_j)
6: End For
7: Generate virtual machines VM = {vm_1,vm_2,...,vm_p}
8: Map VM resources to host capacity
9: Receive task requests T = {t_1,t_2,...,t_n}
10: For each task t_i ∈ T do
11: Extract task length L(t_i)
12: Extract priority P(t_i)
13: Extract processing demand D(t_i)
14: Store workload profile
15: End For
16: Output workload profile W

Algorithm 2: Resource Clustering and Dynamic Resource Evaluation

1: Input VM set VM = {vm_1,vm_2,...,vm_p}
2: For each vm_i ∈ VM do
3: Measure CPU utilization U_cpu(vm_i)
4: Measure memory utilization U_mem(vm_i)
5: Measure network load U_net(vm_i)
6: End For
7: Compute resource capability score

```
R(vm_i) = α·U_cpu(vm_i) + β·U_mem(vm_i) + γ·U_net(vm_i)
```

8: Apply clustering function

```
C = cluster(R(vm_i))
```

9: For each cluster c_j ∈ C do
10: Calculate available capacity

```
Cap(c_j) = Σ R(vm_i) , vm_i ∈ c_j
```

11: End For
12: Monitor cluster utilization dynamically
13: Update cluster status table
14: Output cluster set C

Algorithm 3: Swarm-Based Task Allocation and Adaptive Scheduling

1: Initialize agent population A = {a_1,a_2,...,a_s}
2: For each agent a_k do
3: Generate task-resource mapping M_k
4: End For
5: Evaluate scheduling fitness

```
  F(M_k)   =   1   /   (Makespan(M_k)   +
Energy(M_k))
```

6: Select best agents based on fitness value
7: Update mapping using swarm search

```
  M_k ← M_k + λ · Δ(M_k)
```

8: For each task t_i ∈ T do
9: Select VM from cluster with highest capacity
10: Assign task t_i → vm_j
11: End For
12: Determine execution order

```
   ExecOrder     =     sort(Priority(t_i),
Load(vm_j))
```

13: Execute scheduled tasks
14: Monitor system performance
15: If imbalance detected then
16: Adjust mapping M
17: End If
18: Return optimal schedule M

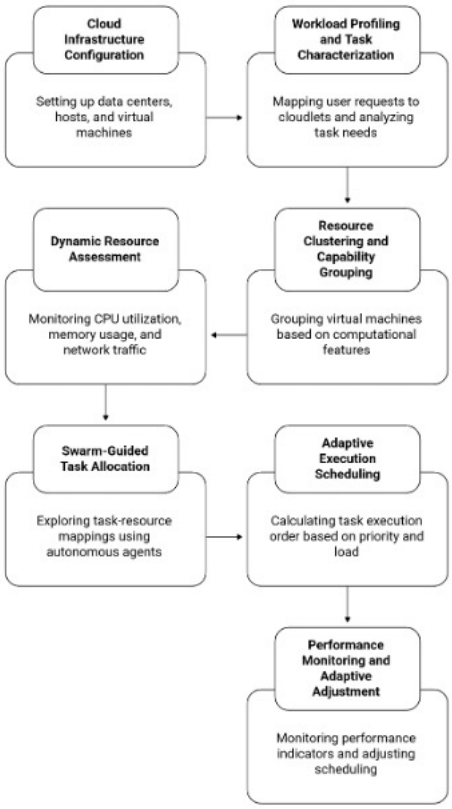


Figure 1: Proposed model

The first algorithm sets up the cloud and profiles the workloads on incoming jobs to be scheduled. The simulation platform is first initialized to simulate the cloud computing environment. A host group (physical machines) is conceptually defined for multiple datacenters, then a datacenter can have multiple hosts/nodes. Each host has particular CPU, memory, network and storage capacity descripting computing power, memory size, network bandwidth and storage capacity. After the hosts have been specified, vms are created and mapped to these physical hosts based on the capacity. These virtual machines are execution environments for executing user tasks.

The infrastructure in place, user requests start flowing in and are mapped as cloudlets, which can be viewed as computational jobs. The execution time, priority and processing requirements of each task are known. These parameters are crucial for each task to describe workload characteristics. The system uses this information to build a workload profile based on computational needs of every task the users submit. This profiling step help scheduling algorithm understand how the tasks vary in complexity and urgency). By setting up the infrastructure and profiling tasks prior to scheduling, the algorithm guarantees that the cloud system has precise knowledge to make good resource allocation decisions at the later stage.

The second algorithms deals with the grouping of these resources into clusters and working conditions evaluation. Each virtual machine in the cloud is observed for critical system parameters such as processor utilization, memory utilization, and network utilization. These numbers also tell us whether a resource is under- or over-utilized at any particular moment. A resource capacity score is then computed for each vm using these utilization measurements. This score is referred to as the general capacity of a vm to handle more jobs.

Once the capability scores are calculated, the algorithm clusters the similar vms. Create two clusters of machines, each with equal processing capacity and equal workload capacity. This grouping into clusters reduces the scheduling complexity in terms of the amount of resource to be taken into account in the task assignment. Instead of looking for a machine on the entire pool, the scheduler will first look for a suitable cluster matching the task. The algorithm now sums up the resource capacity scores of the vms to get the total capacity within each cluster. Cluster usage is being monitored continuously while the system is running to find the over- or under-utilized hosts. The cluster information is dynamically updated so that the scheduling procedure can adapt to workload variations. This monitoring and clustering technique enhances scheduling

efficiency and contributes in keeping uniform resource utilization ratios across the cloud infrastructure.

In fact, the central algorithm that carries out the task scheduling is the same third algorithm. The algorithm starts with a population of artificial agents, each representing a potential solution to the task assignment among the available VMs. Every agent produces an initial task-resource mapping that shows a way task can be distributed across compute resources. These potential schedules are evaluated applying a fitness function which assesses their quality. The fitness value is usually derived based on the entire makespan of tasks and the energy used by system in executing them. Schedules that reduce makespan and energy consumption are strongly favoured by the fitness function. After evaluating the solutions candidates, the algorithm chooses the top-performing candidate solutions (i.e., the agents) and guides the next activities of agents in the s space accordingly. Agents exchange information about good assignments of tasks to resources and consequently modify their task maps by altering their search directions. Through this collaborative search mechanism, the algorithm incrementally searches for better scheduling solutions. The algorithm allocates an appropriate virtual machine for each task from the most powerful cluster according to the capacity.

The sequence of execution for the tasks are then established taking into account the task priority and the present load of the machine that it is assigned to. Those tasks that have higher priority or tight deadline are executed earlier in the schedule.

The system monitors continuously the performance parameters, including CPU utilization, task finish time, and system workload during execution. If the scheduler feels that the schedule is unbalanced or the resources are under-utilized, it modifies the task-resource mapping to obtains a better schedule. This process of adaptation enable the system to adapt to varying workload patterns with a balanced resource utilization and a minimized execution idle time over the cloud environment.

functions, including resource brokering, cloudlet to VM mapping, and task execution across multiple hosts. This centralized control mechanism also aids in ensuring uniform scheduling policy over the simulated cloud.

IV. EXPERIMETNAL RESULTS

The performance evaluation of the proposed scheduling method is done based on Java CloudSim simulation toolkit. CloudSim enables modelling and simulation of cloud infrastructure and application services, and it supports evaluation of task scheduling mechanisms in the presence of diverse workload patterns. The simulation environment is a distributed cloud infrastructure model composed of data centers, hosts, virtual machines, and cloud tasks. We create 5 data centers with 2 hosts in each for the simulation setup. Thus, there are ten physical hosts involved in the cloud infrastructure. This allows the simulation to capture the essence of cdistributed computing while keeping the experimental complexity at a reasonable level.

User applications in this environment produces tasks which are modeled as cloudlets. A cloudlet is a unit of workload of computational submitted by users to the cloud system. The cloudlet has some information about the length of a task, the processing requirement and the execution characteristics. The number of cloudlets is varied from 200 to 3600 in the to represent different workloads in the simulations. Small task sets model light computing environment, while larger task sets model high-demand cloud environments with large-scale application workload. The system can be evaluated both for moderate and heavy loads through changing the tasks number.

The broker is at the midpoint between the users and the data centers in this simulation. The broker a. the queue b. the brokers queue and the virtual servers executes poll on the is server. In the experimental setup, the whole cloud is controlled by a single broker. The broker needs to execute certain critical

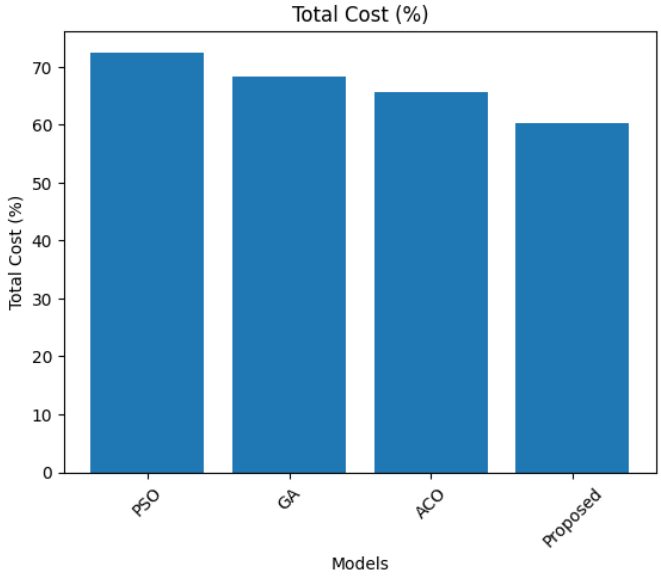
The total memory assigned to each host is set to 20,480 MB. This limit memory allocation is to say how many VM can work well in one host. Sufficiently large memory is also required for virtual machines to perform tasks without being slowed down. The host processing power is another significant parameter, and it is given in MIPS (Millions of Instructions Per Second). In our simulation, each host has a processing power of 5000 MIPS. This is the processing speed of the processor and determines how fast the job will run.

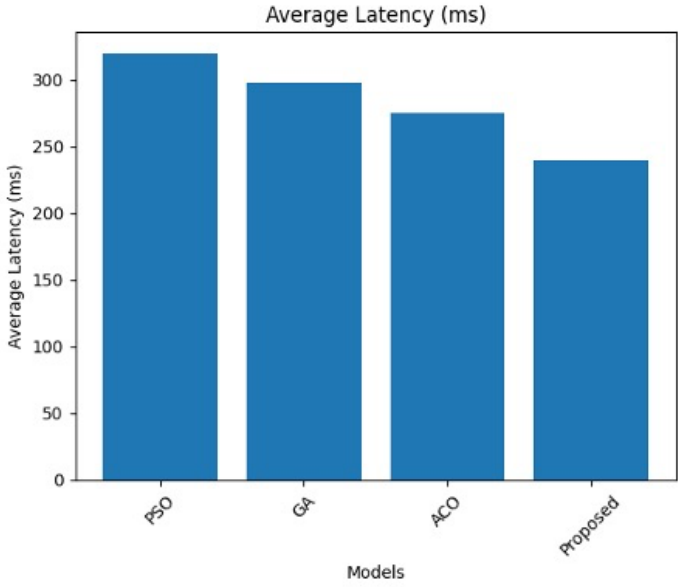
The size of the storage is also a critical factor in the simulated environment. Each host is provisioned with 1,048,576 MB of storage, which is more than enough for the application data, system files, and temporary processing needs of the disk. Network communication power, as well as processing power and storage, can be modeled using the bandwidth of the form of a resource. In the simulation we set the bandwidth of each host to 500,000 MB/s. This is the maximum throughput that can be achieved when transferring data between hosts and other elements in the cloud structure. Efficient communication of virtual machines with data center and task execution is achieved by high bandwidth.

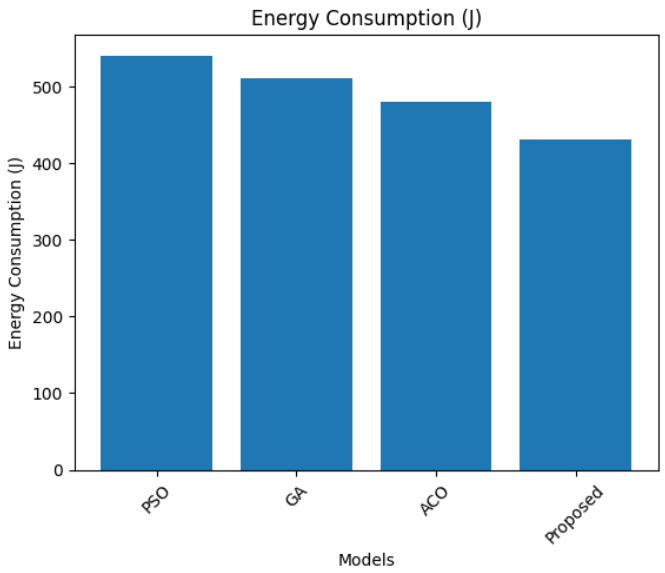
The experimental procedure during simulation has several consecutive steps. Stage One: Infrastructure Initialization: The simulation environment is set up by establishing data centers, hosts, virtual machines, and cloudlets. In this phase strictly the building blocks of the cloud environment are created and wired up inside the simulation. The next step, known as task allocation, is the process of assigning cloudlets to virtual machines based on their computation needs and the availability of resources. This distribution procedure tries to even the resource use among all the resources to maximize system performance. The third phase is scheduling, which determines priorities for the cloudlets execution order in accordance with priorities levels, time execution deadline, system resources availability. Scheduling algorithms try to minimize the overall task completion time and maximize the system (hosts and VMs) utilization. Consequently, the output of that final step is the simulation at which all the system's tasks have been processed and the performance measures obtained. In this phase, several system parameters including execution time, resource utilization, processing load, and so on are gathered and analyzed.

The CloudSim API is leveraged to realize these simulation components in Java. CloudSim provides predefined classes which capture the entities of a cloud infrastructure. Key classes include the Datacenter class to model a computing resource or a computing cloud, the Host class for physical machines, the VM class for virtual machines, Cloudlet for the tasks, the DatacenterBroker class to submit andmanage the responsibilities of your VMs. With these classes, a developer can effectively build out theoretically any cloud environment within the simulation.

For instance, a data center is formed by creating an instance of the Datacenter class , and by specifying its characteristics like buying list of hosts, processing power, and storage facilities. So are tasks, which can be specified using the Cloudlet class (with length, priority, and number of required instructions per task). These cloudlets are then sent to the broker, which assigns them to virtual machines for running. Through this hierarchy simulation framework, one may study and compare the performances of scheduling algorithms in a controlled experimental environment.







Total Cost Metric

The total cost measure quantifies the cost of cloud operation associated with performing a computational workload in the mobile edge cloud. This cost is generally related to the use of resources, processing overhead and the communication between edge nodes and cloud resources. It can be seen from the comparison that classical optimization models like PSO, GA and ACO achieve even higher operational costs. These models usually allocate resource to the users by ignoring the workload dynamics and infrastructure efficiency, which may lead to high computational overhead. The proposed scheduling architecture achieves a significant reduction in overall cost due to its contribution to a more balanced workload distribution among the edge resources. Through the selection of efficient computing nodes and avoidance of unnecessary start-up of virtual machines, the system achieves better utilization of resource and minimizes the cost of task execution.

Explanation of Average Latency Metric

Average latency indicates how long it takes for a user request to pass through the system and receive a response after being processed by the system. Since a lot of applications need fast convergence rates, such as smart healthcare systems, autonomous vehicles, real-time monitoring, and so on, latency is considered as a key factor in mobile edge cloud computing environments. The Simulative results show that the latency values in PSO, GA and ACO based scheduling algorithms are high. This is because tasks can be assigned to far resources or congested nodes, resulting in high-processing delay and high network communication overhead. Due to task assignment to the neighboring edge computing nodes with adequate computational capacity, the proposed model significantly minimizes the average latency. Effective task placement reduces network delay and accelerates response delivery, leading to a better-responsive cloud.

Interpretation of Energy Consumption Metric

Power consumption is the amount of power required to run workloads on the DCI infrastructure. In large-scale mobile edge cloud environments, it is a challenging to achieve energy efficiency while sustaining sustainable operations for service providers with reduced cost. The experimental results indicate that the energy of traditional scheduling strategies is bigger for these reasons of computational resources wastage and longer task lengths.

V. Conclusion

In this paper, an adaptive workload scheduling for resource management under mobile edge cloud computing is introduced. The method proposed was aimed at enhancing distribution of workload, alleviating processing delay, and maximizing resources utilization in distributed edge nodes and cloud datacenters. The proposed framework took computational resource as well as cluster based scheduling into consideration and a novel adaptive scheduling approach was presented to dynamically select appropriate computing resources to perform submitted tasks. By means of workload profiling, resource grouping, and intelligent assignment of tasks, the framework effectively handled computing applications in a dynamic environment.

Simulation Results: Simulation on CloudSim platform showed that the proposed method was more effective with various workloads. Several performance metrics are evaluated such as total operational cost, average latency, energy consumption and the tasks completion time. Results show that the proposed scheduling model outperformed other existing optimization methods such as PSO, GA and ACO methods. The framework allowed for a more efficient distribution of workload across edge nodes, which led to decreased execution delays and an improved, unified resource utilization. Energy consumption was reduced by efficient task allocation and power-aware virtual machine activation. These improvements provide for faster task execution and a more robust operation of mobile edge cloud infrastructures. Experimental results analysis showed that the scheduling strategy could well adapt to moderate workloads as well as heavy loads, achieving load balancing between the computing resources. The integration of

workload-aware scheduling and dynamic resource monitoring made the system stable in performance and light in resource load. To sum up, the proposed model improves the system performance and allows trustworthy task execution for MEC applications with low delay and high computation reactivity.

Future Work

In the future, this work can be extended by integrating intelligent learning techniques to enhance the scheduling decisions in dynamic cloud environments. Techniques from machine learning or deep learning can also be used to predict workload and resource utilization trends so that the scheduling framework might be able to take proactive actions in addressing future system states. These predictive features could enhance latency reduction and system responsiveness even further, especially for applications as real-time games.

Another line of future work could be to build the proposed scheduling framework on real mobile edge computing systems rather than only simulating it. This would enable the evaluation of system behaviour in realistic network conditions, heterogeneous devices, and diverse user mobility patterns. Furthermore, a potential research direction is to investigate energ-aware scheduling models involving renewable energy availability and/or dynamic power management policies for edge data centers.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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, pp. 2309–2331, Jun. 2022.

[6] M. Gill and D. Singh, "ACO based container placement for CaaS in fog computing," *Procedia Computer Science*, vol. 167, pp. 760–768, Jan. 2020.

[7] 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.

[8] 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.

[9] 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.

[10] 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 Applications*, vol. 133, pp. 60–74, May 2019.

[11] 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.

[12] 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.

[13] G. J. S. 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.
[14] 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.
[15] 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.