A Novel Resource Scheduler for Resource Allocation and Scheduling in Big Data Using Hybrid Optimization Algorithm at Cloud Environment

Aarthee Selvaraj
Department of Electrical and Electronics Engineering, University College of Engineering, India
aarthee@aubit.edu.in

Prabakaran Rajendran
Department of Electrical and Electronics Engineering, University College of Engineering, India
hiprabakaran@aubit.edu.in

Kanimozhi Rajangam
Department of Electrical and Electronics Engineering, University College of Engineering, India
kanimozhi_17@aubit.edu.in

Abstract: *Big Medical Data (BMD) is generated by cellular telephones, clinics, academics, suppliers, and organizations. Collecting, finding, analyzing, and managing the big data to make people's lives better, comprehending novel illnesses, and treatments, predicting results at initial phases, and making real-time choices are the actual issues in healthcare systems. Dealing with big medical data in resource scheduling is a major issue that aims to offer higher quality healthcare services. Hadoop MapReduce has been widely used for parallel processing of large data tasks and efficient job scheduling. The number of big data tasks is constantly growing; it is becoming more essential to minimize their energy usage to reduce the environmental effect and operating expenses. Hence to overcome these disadvantages, we propose a novel resource scheduler for big data using a Hybrid 2-GW Optimization Algorithm (H2-GWOA). We employ the Improved GlowWorm Swarm Optimization Algorithm (IGSOA) and Mean GreyWolf Optimization Algorithm (MGWOA) for optimizing the MapReduce framework in heterogeneous big data. The CloudSim platform was used for the simulations. The performance of the proposed scheduler is proved to be better than the conventional methods in terms of metrics like latency, makespan, resource utilization, skewness, and Central Processing Unit (CPU) consumption.*

Keywords: *Resource management, particle swarm optimization, computer performance, data analysis, virtualization.*

Received August 4, 2022; accepted May 24, 2023
https://doi.org/10.34028/iajit/20/6/3

1. Introduction

On-demand delivery of needed resources and services via high-speed computer systems is among the most significant advantages of cloud computing. It has shown improved results in servicing different types of large-scale and complicated client activities due to continuous development, network technologies, and infrastructure improvements. Complex development systems and apps, cutting-edge virtualized and physical technology met heterogeneous customers' needs. Cloud technology and the Internet of Things (IoT) have become essential ideas in today's technological advances, setting new objectives for technology innovation. Resource scheduling is one of the most significant problems in the era of cloud technology. A Satisfactory Quality of Service (QoS) must be achieved using appropriate hardware architecture and techniques while conducting resource scheduling. In contemporary literature, a cloud infrastructure component known as the broker is responsible for mapping required end-user tasks to accessible virtualization equipment, often implemented in Virtual Machines (VMs) [14]. By performing the task scheduling, the broker conducts mapping.

With the rise in the number of tasks submitted and the number of possible resources, mapping tasks to the suitable VMs for execution becomes more challenging. If an inefficient scheduling method is employed, certain VMs may cause over-utilized or under-utilized, resulting in cloud system performance deterioration. The resource scheduling issue falls under the NP (pseudo-random polynomial duration) categories of hard design problems. It is worth noting that the words task and cloudlet planning have been used in recent computer science research to describe the process of matching requested end-user tasks to available VMs.

Classic algorithms for resource scheduling are employed in specific low-level applications and optimization methods are inefficient because deterministic methods cannot produce satisfactory, optimum, or near-optimal solutions for NP-hard problems in a reasonable amount of time. Classical methods cannot assess every viable solution from the dimensional search space in the polynomial period due to the random search complexity and increasing range of available solutions. Implementing meta-heuristics and heuristics-based methods [12] is among the most effective approaches to resource provisioning rather than

NP-hard methodology. While these approaches do not ensure discovering an optimum solution, they have been shown to produce satisfactory answers in polynomial time practice. The primary benefit of employing a resource allocation method in cloud computing is optimizing resource usage while lowering operational costs. In a cloud computing environment, the fundamental purpose of resource allocation is to efficiently optimize Physical Machines (PMs) or manage workloads in distributed PMs to decrease resource consumption, bottlenecks, and overloaded resource utilization [20]. The practice of distributing cloud resources to cloud apps over the network in a structured way is referred to as resource allocation. The solutions may not even be long-lasting if cloud solutions fail to provide a resource to consumers on request. This issue was addressed by allowing cloud providers to allocate resources individually at every module. As a result, the allocation of resources is emphasized as a component of the management framework, demonstrating that it is a cost-effective element in allocating resources. In the cloud architecture, allocation of resources is used to enhance customer satisfaction while decreasing processing time. Minimizing resource consumption guarantees cloud service quality, satisfaction for the service provider, and enhances the makespan and latency. The fundamental concept behind dynamic scheduling is allocating requests according to the time of program execution.

2. Related Works

Cloud computing has evolved into a new computing paradigm with enormous corporate and commercial possibilities in this real-world [11]. In a world with finite energy supplies and an ever-increasing need for greater processing capacity, green cloud computing is becoming more essential, must handle resources correctly, and virtual machines must assign suitable host nodes to conduct computations to optimize utilization and reduce the total cost of cloud computing infrastructure and application maintenance. We proposed a novel resource allocation scheduler based on the QoS-aware VMs consolidation technique for cloud settings and efficient resource allocation that uses the mechanism of virtual machine resource usage for performance analysis to allocate virtual servers on the public cloud [8]. We have used the CloudSim simulator to develop and test the proposed methods. Chen *et al*. [4] presents a system for self-adaptive resource allocation that is made up of feedback loops that are performed independently using our constructed:

a) Iterative QoS prediction model.
b) Particle Swarm Optimization (PSO)-based runtime decision method.

Unlike earlier QoS prediction algorithms that predicted a QoS value once and for all, our method improves the anticipated QoS value over time until it reaches the greatest possible value. [7] In the first step, we conduct resource provisioning while taking into consideration resource prices, quality of service violations, and cloud-based root server redirection, as well as QoS restrictions and limited cloud site resources. In the second step, cloud site assignment is performed, in which expenses associated with QoS violations and cloud-based root server redirection are avoided or decreased by utilizing three distinct proposed methods for fixed assigned resources. The challenge of cloud-based computation requires MapReduce (MR) computations that are abstracted as a dynamic optimization problem [23]. To solve this issue, an event-driven resource provisioning system is proposed. Experiments compared this new event-driven framework to the commonly used static resource provisioning framework and periodic resource provisioning techniques.

Virtualized technology [16] manages resources in the cloud environment to enhance the energy efficiency, and virtual machines often host applications. QVMS, a QoS-aware VM scheduling technique for energy saving in cloud-based CPS with QoS standards, has been developed in response to this issue. Stanik *et al*. [18] proposed an orchestrating and federating heterogeneous networks in API-based system software architecture that allows for QoS-aware network resource settings in a single cloud datacenter's infrastructure, as well as federated networking across various SDN-based cloud networks beyond the information edge networks. Additionally, this architecture employs a Service Level Agreement (SLA) protocol, language to disclose Key Performance Indicators (KPI) and to negotiate suitable QoS constraints that are applied to the virtualized network substrate [9]. The study [17] also seeks to investigate the function of virtualization in efficiently delivering resources depending on customers' needs. It also looked at how virtualization may improve network speed, save money by decreasing the number of physical machines in the datacenter, balance load, reduce server energy, and distribute resources dynamically while meeting customers' needs. This provides [13] a Reinforcement Learning (RL)-based resource allocation system, which enables the cloud to choose whether to approve a request and exactly how many resources to provide in response. The dynamic resource allocation technique [24] and the energy conservation method are the two approaches, we considered for our proposed work. To test the proposed method with live virtual machine migration, we first built an infrastructure architecture based on OpenStack VMs. Second, we developed an energy-saving proposed method for assigning the dynamic resources. Finally, we discovered in the tests that the proposed methods could efficiently use VM resources and save energy on datacenters. Tong *et al*. [21] used the Fog Radio Access Network (F-RAN) model in which content transmission and cooperative caching methods are optimized together. To obtain an ultra-low latency for the model F-RAN, they constructed a mixed-integer nonlinear programming problem. In response to this issue [22], a dynamic resource allocation strategy for balancing load in fog environments was provided. The load-balance assessment for various computing hubs is presented first, followed by a fog computing system design. Then, to establish a load balancing for fog computing systems, a matching allocation of resource technique in the environment of fog is developed using the allocation of resources for the static method and service migration for the dynamic method. The categorization of big data types in healthcare-associated analytical methods [6], generated value for stakeholders, platforms, tools for managing large health data, and future elements of the client are discussed.

Improved Fruit Fly Optimization (IFFO) algorithm is used for optimizing complex scenarios with multiple workflows scheduling over the cloud that helps to reduce the cost and makespan in the minimal amount of time [1]. The Sunflower Whale Optimization Algorithm (SWOA) achieves effective resource allocation based on the execution of the selected task where every application handles different tasks that improves the efficiency of the cloud model [19]. PSO algorithm dealt with optimizing the cloud resources. A scheduler model for efficient allocation of resources for data processing, as well as a [5] PSO-based scheduling algorithm named PSO-COGENT not only optimizes execution cost and time but also reduces the energy consumption of cloud datacenters while keeping the deadline in mind. It monitors the virtual machine placement to identify the optimal resource mapping. The sequence of mapping is carried out between PMs and VMs concerning particles for optimization.

3. Proposed Methodology

Big data task, MapReduce, and its open-source implementation have received significant popularity. There have been several efforts to enhance existing MR schedulers and build more efficient techniques or algorithms. Because the number of such big data jobs is always expanding, minimizing their energy use is crucial to save the environmental and operational expenditures. We proposed a new resource scheduler for big medical data utilizing the hybrid 2-GW optimization algorithm to tackle these challenges. Here, the term 2-GW refers to the combination of techniques GreyWolf and GlowWorm. The flow of the proposed approach is described in this section which is shown schematically in Figure 1. Resource management's job is to distribute resources to service providers based on their needs and limitations. A service provider's goal is to follow its SLA with clients. We propose a hybrid 2-GW optimization algorithm to optimize the MapReduce framework and heterogeneous workloads in handling large datasets. The main elements of the proposed approach are described in detail.

a) Cloud Portal: it gives you a single point of entry to a cloud-based distributed application that helps to find and manage cloud resources. This module is in charge of converting user requests into Resource Specification Language (RSL), a cloud-friendly language. The final product is initially sent to the cloud portal, which then sends it to the users.
b) Enforcement of the SLA: the various cloud computing models and levels should be considered while deploying SLA in the cloud since the measurements and needs of QoS specified in SLAs vary with each model and every level. Throughout the contractual process, service terms and levels, as well as QoS requirements, are created, including SLA metrics that will be expressed and monitored to ensure that the SLA agreement is fulfilled.
c) Migration Monitoring: it monitors VM migration, which refers to the transfer of VMs from one resource to another, such as from one physical host to another physical host. One of automated migration tools, the VMware vSphere Distributed Resource Scheduler, makes use of vMotion to enhance virtual machine performance across virtual clusters.
d) Resource Scheduling: the MR class schedule is the heart of cloud computational resources scheduled, and it implements the logical step. This approach will handle all scheduling schemes. Concurrent analysis, fault-tolerant handling, and load-balancing issues are handled while managing the resources.
e) Resource Pool: it also helps with centralized resource delivery and management by recording resource information such as resource type, quantity, and resource location.
f) Workload Analyzer: the workload analyzer is the component in charge of producing request arrival rate estimations. This data is utilized to calculate the precise number of application instances. It notifies the load predictor and efficiency modeler when the service request rate is anticipated to vary, in addition to the specific technique for estimating future load. This warning must be sent before the expected time for the arrival rate to change, giving enough time for the load predictor and performance modeler to compute system modifications and the applications provisionary for enough time to release or deploy the necessary VMs.
g) MapReduce AppMaster: it coordinates the set of MR tasks running in the cluster. Each application has a dedicated AppMaster who is responsible for managing the containers and collaborating with the node managers to execute and monitor the tasks.

The cloud users are categorized into administrative staff, patients, doctors and clinical staff who request the set of services to execute the jobs through cloud portal. The workload analyzer performs the healthcare analysis such as electronic health records, patient health data and clinical data are integrated for the personalized exact treatment methodology. The resource manager allocates the desired resources for the exact VM depend on the type and size of the healthcare data through the node manager. It maintains the SLA with cloud service providers for the live VM migration if it is over-utilized by the set of resources. The node manager maintains the container for task allocation and MR Application Master (AppMaster) allows the user to get the status of the requested healthcare service through resource manager

3.1. Hybrid 2-GW Optimization Algorithm

The GreyWolf and GlowWorm techniques are combined with the queuing theory to boost cloud performance by decreasing overall application execution time via optimized resource scheduling. It uses preemptive scheduling, which reschedules jobs that have been waiting in line for a long time. This utilizes the join procedure to stop a thread from running. The proposed 2GW hybrid optimization Algorithm (1) provides a better performance analysis of the five scheduling metrics with enhanced QoS scheduling techniques which is derived below, along with a glossary of terminology

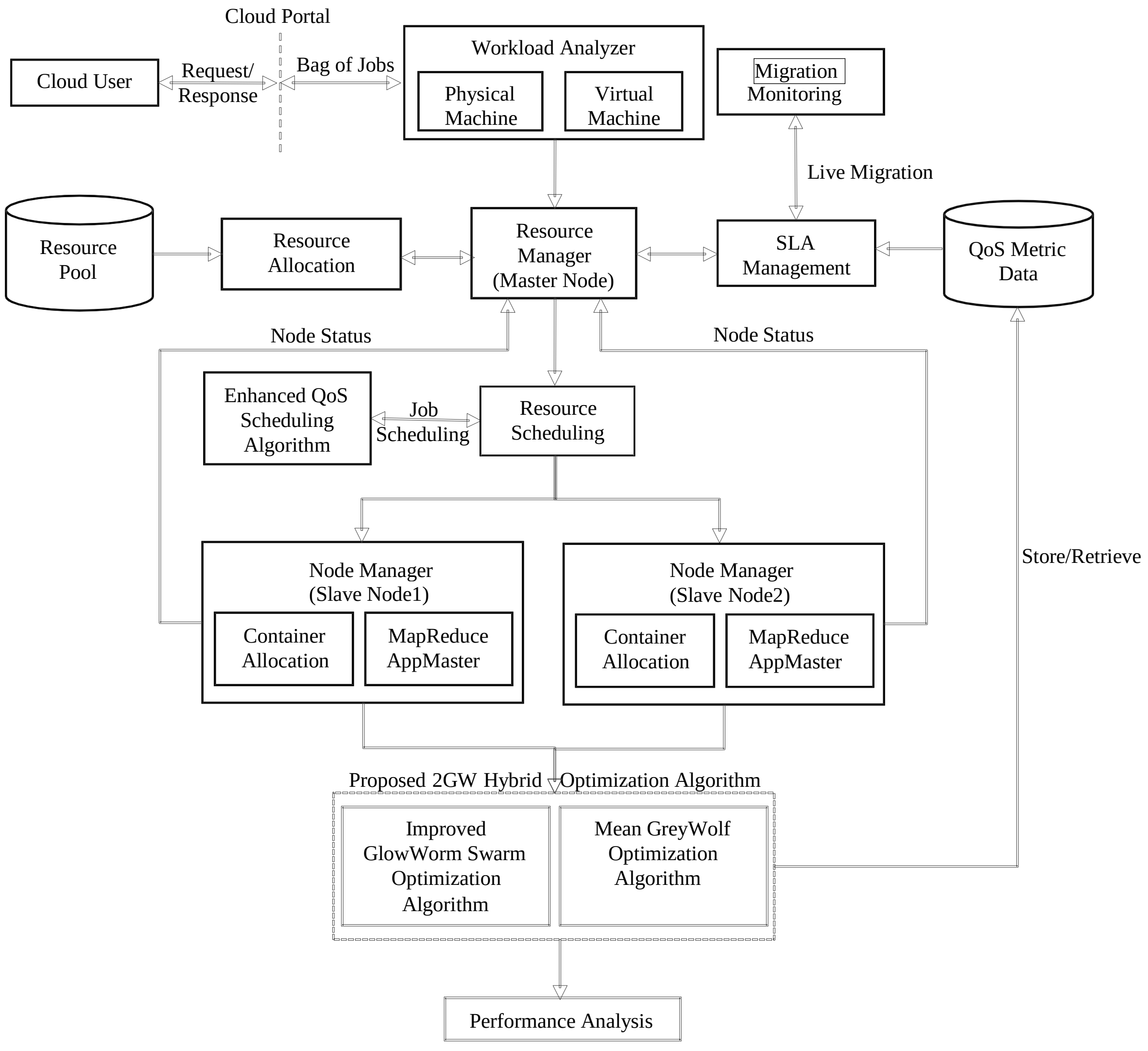

Figure 1. Schematic representation of the proposed method.

Algorithm 1: Hybrid 2-GW Optimization Algorithm

```
SET task Tk to ready queue q
INIT accessible resources to nodes N
COMPUTE each completed job at different metrics on N
ASSIGN current status of the job as CS
  FOR each task Tk has a deadline TD
  CALCULATE arrival time for Tk
        DETERMINE probability mass function to arrival time
  END FOR
CALCULATE resource capacity as rc along with waiting time on TD
 FOR Tk on different metrics
        GET minimal arrival time as D from TD
        IF D is minimum than rc then
          DETERMINE resource to Tk
ELSE
    DETERMINE resource rescheduling Tk with
    D on CS that maximize TD
END
    IF arrival time is greater than executing tasks in q include
    waiting time then
    SCHEDULE required job from q with     minimal TD
END
END FOR
```

The tasks Tk to be completed and will be queued in the ready queue 'q'. The variable 'D' denotes the virtual machines. A similar set of jobs is categorized into different categories that represent 'C' from 1 to m resources. The variable rc is the average number of tasks performed by 'D' in a given amount of time and CS represents the task's current status. The combination of available resources and processing time is used to calculate the Task Deadline (TD). To calculate the waiting period, the arrival time of each job is taken into consideration. The set of task arrival times is calculated using the probability mass function. Because the arrival time of jobs is unknown, it is considered to be stochastic. The continuous-time sequences for the tasks are handled by the Markov process that is used to determine the arrival time. The exact solution space in VM is identified to schedule the resources to execute the set of MR jobs.

3.2. Mean GreyWolf Optimization Algorithm

The mean greywolf optimization method improves the GWO method's performance and accuracy. The surrounding and hunting equations have been changed in this proposed approach. The processes for the following equations are identical to the GWO algorithm as a whole. The primary goal of this method is to increase the efficiency of each wolf's mobility and find the best route for each wolf in the search region. The following are the divisions of the MGWOA:

a. *Surrounding the prey*: the following Equations (1), (2), (3), and (4) are used to create a hunt where grey wolves surround the prey.

$$\vec{M} = \left|\vec{F}.\vec{Y}_q(ci) - \theta.\left(\vec{Y}(ci)\right)\right| \tag{1}$$

$$\vec{Y}(ci+1) = \vec{Y}_q(ci) - \vec{B}.\vec{M} \tag{2}$$

Where, $\vec{F}.\vec{Y}_q(ci)$ represents the vector of prey's location, ci represents the current iteration, and $\vec{Y}(ci)$ represents the vector of grey wolf location. The vectors are denoted as,

$$\vec{B} = 2\vec{b}_1.\vec{s}_1 - \vec{b}_2 \tag{3}$$

$$\vec{M} = 2.\vec{s}_2 \tag{4}$$

It has reduced the element $\vec{b}$ from 2 to 0 and the vectors $\vec{b}_1, \vec{b}_2$ each take a random value among [0,1].

b. *Hunting*: the three best and optimum design parameters are designated by ρ, μ, δ, and the grey wolf vector solution is derived by $\vec{Y}(ci+1)$. By computing the mean of the locations, each wolf's location in the search space area has been improved by using the below Equations (5), (6), (7), (8), (9), (10), and (11).

$$\vec{M}_\rho = \left|\vec{F}_1.\vec{Y}_\rho - \theta.\vec{Y}(ci)\right| \tag{5}$$

$$\vec{M}_\mu = \left|\vec{F}_2.\vec{Y}_\mu - \theta.\vec{Y}(ci)\right| \tag{6}$$

$$\vec{M}_\delta = \left|\vec{F}_3.\vec{Y}_\delta - \theta.\vec{Y}(ci)\right| \tag{7}$$

$$\vec{Y}_1 = \vec{Y}_\rho - \vec{B}_1.\left(\vec{M}_\rho\right) \tag{8}$$

$$\vec{Y}_2 = \vec{Y}_\mu - \vec{B}_2.\left(\vec{M}_\mu\right) \tag{9}$$

$$\vec{Y}_3 = \vec{Y}_\delta - \vec{B}_3.\left(\vec{M}_\delta\right) \tag{10}$$

$$\vec{Y}(ci+1) = \frac{\vec{Y}_1 + \vec{Y}_2 + \vec{Y}_3}{3} \tag{11}$$

3.3. Improved GlowWorm Swarm Optimization Algorithm

The Glowworm Swarm Optimization Algorithm (GSOA) is an excellent candidate for parallel processing due to its clustering nature. We used the MapReduce framework in conjunction with the sequential glowworm optimization technique.

The initialization phase and the MapReduce phase are the two major stages of improved GSOA. An initial glowworm swarm is generated during the startup process. Within the specified search space, regular randomized produces a random location vector (Y_j) for every glowworm '*j*' which explores its own neighborhood region; '*t*' is the current iteration; 's' refers to the luciferin decay constant (s∈(0,1)) and β is the luciferin enhancement fraction. The Y_j vector is then used to assess the objective function I(Y_j). After that, using the starting luciferin level LL_0, I(Y_j), and other specified variables, (12) is used to determine the Luciferin Level (LL_j).

$$LL_j(t) = (1-s)L_j(t-1) + \beta I\left(Y_j(t)\right) \tag{12}$$

The MR task in the MapReduce phase uses the first saved file as its input. Step two of MR-Glowworm Swarm Optimization (MR-GSO) involves an iterative sequence of MapReduce tasks, with each MapReduce task representing glowworm swarm optimization iteration. Each MR task produces an upgraded glowworm swarm with updated data, subsequently fed into the next MR operation. The method uses the MapReduce model's power to speed up the time-consuming phases of luciferin level updating and glowworm movements at every MapReduce operation. For each glowworm, an objective function I(Y_j) uses (13) to identify the neighbor group $NG_j(t)$ during the movement step, which necessitates distance computations and luciferin levels (LL_i and LL_j) comparison between every glowworm and other swarming members.

$$i \in NG_j(t) \text{ iff } d_{ji} < qd_j(t) \text{ and } LL_i(t) > LL_j(t) \tag{13}$$

Where *'i'* is one of the closer glowworms to glowworm *'j'*; d_{ji} is the Euclidean distance between glowworm *'j'* and glowworm *'i'*; *qd* is the radial sensor range that is initialized with the same value q0. This procedure is repeated N2 times, where N is the size of the swarm. The map function, which is part of a MapReduce task, is responsible for identifying neighbor groups. A duplicate of the saved glowworm swarm is obtained from the distributed environment. The functionality offered by the MR framework for file storage before the neighbor group discovery procedure is done in the map phase. To identify the best neighbor (14) using the roulette wheel selection technique, the neighbor probabilities are computed.

$$prb_{ji} = \frac{LL_i(t) - LL_j(t)}{\sum_{m \in NG_j(t)} LL_m(t) - LL_j(t)} \tag{14}$$

The reduce function receives glowworm ID *'j'*, including its value and glowworm ID *'j'* with the chosen neighbor position vector *(Yj)* from the map phase after the map operations. The mapped function's produced intermediary outputs are divided using the standard partitioner by allocating the glowworms to the removers depending on the IDs using the modulus hashing algorithm as an intermediate stage in the MapReduce process. The MR Job's reducer is in charge of upgrading the luciferin level LL_m, which is the most costly step in the glowworm optimization since the optimization problem is assessed for the current glowworm location at this point. When the controller gets the key, a list of values from the map function, where the key is the glowworm ID, and the list of values includes the glowworm values and its best neighbor location, the luciferin level upgrading process starts (Y_j). The reduce function retrieves the glowworm data Y_j, I(Y_j), LL_j, and qd_j, as well as the neighbor position vector. After that, (15) is used to update the glowworm position vector.

$$Y_j(t) = Y_j(t-1) + q\frac{Y_i(t) - Y_j(t)}{d_{ji}} \tag{15}$$

Following that, the goal function is calculated using the new glowworm location vector $Y_j(t)$, and the luciferin level is adjusted using Equation (12) and at the end of the IGSO iteration, the local decision range Equation (16) is used to adjust qd_j. Finally, the reduce function outputs the glowworm ID *'j'*, which contains the freshly modified glowworm data. The glowworm swarming replaced the previous swarm in the distributed system, which is utilized after the next MapReduce task.

$$qd_j(t) = \min\left\{qs, \max\left[0, qd_j(t-1) + \gamma\left(mt - \left|M_j(t-1)\right|\right)\right]\right\} \tag{16}$$

Where, $qd_j(t\text{-}1)$ is the previous iteration of qd_j. *qs* is the radial sensor range constant, γ is a model constant, *mt* is a constant parameter used to restrict the neighborhood set size, and $\left|M_j(t)\right|$ is the actual neighborhood set size.

GreyWolf optimizer helps to schedule workflow tasks in a cloud environment, which helps to minimize task completion time and VM usage cost, and to maximize resource throughput. It identifies the exact block pool in the available solution space (i.e., appropriate virtual machine selection) to execute the set of jobs based on their priority, size and resource type. The GlowWorm optimizer uses evolutionary computation and quantum strategies to identify the neighbourhood tasks that achieve more efficient task scheduling with minimal costs. Each glowworm is obligated to move towards the outstanding individuals within its selection solution range until the ideal value is discovered by the proposed algorithm to be optimally solved. A pre-scheduling strategy helps to improve the resource scheduling in the execution of large dataset.

The two main parameters of the medical data are the set of tasks or jobs and the resources needed to accomplish the task. First initialize the parameters and resource population to compute the fitness of every single job in the VM. Secondly, our proposed algorithm generates the random population to calculate the target value of the corresponding scheduling strategy for each individual block in the VM. Then provide judgement by performing iterations and mutations that guarantees the exact solution space in that required population. Local and boundary searches are performed for the evaluation of the target value. Finally, the solution space is updated to find the optimal solution in the targeted VM for resource scheduling. The GreyWolf technique lacks poor exploration, a prematurely low convergence rate, and imbalanced exploitation to find the exact optimal solution. The GlowWorm technique has a few drawbacks, such as poor stability, low solution precision, and slow convergence. The main purpose of the proposed hybrid technique is to increase the efficiency of the motion to access the block of resources from the resource pool and the suitable path for each task present in the desired virtual machine. It reduces the replicated computation, identifies dependencies among the virtual machines, and accelerates the convergence rate depending on the resource type that provides the optimal solution.

4. Result Analysis

The CloudSim framework [2], a simulation tool for modeling cloud systems, was employed for the experimental investigation. The hardware configuration for the proposed implementation is windows 10 OS on 64-bit with the Intel (R) Core (TM) i5, 6 GB ram, 4 CPU cores as 40 number of virtual machines with 925 tasks, number of datacenter is 1 with the network bandwidth of 2800 Mbps, SLA policy by the cloud provider on VM is Time_Shared and the number of resources is assumed to be 75. Although the cloud environment is thought to contain an infinite amount of resources, we must restrict the number of resources to arrive at an ideal solution. The enhancement in our proposed algorithm is analyzed with the diabetes prediction dataset from Kaggle as the single resource request for the simulation entity. The size of the parent population is 500, number of iterations is 50 with the fitness extraction value of 0.4 and Luciferin volatilization factor is 0.2. Based on five scheduling goals such as latency, makespan, resource utilization, memory consumption, and skewness, the proposed hybrid 2-GW optimization algorithm is compared and analyzed against IFFO, SWOA, and PSO, which provide better performances in resource scheduling on given metrics. The experimental findings are shown in Figures 2 to 6 as graphs and the performance of the proposed scheduler is improved by 20% to 30%, which is higher than the existing scheduling methodologies.

4.1. Latency

The latency *u(t)* is the measurement of the delay that occurred between the cloud user request and the response from the service provider. The average delay for the computation tasks plays an important role in resources allocation as well as improving resource utilization (17),

$$u(t) = f\left(t, u(t), u\left(t - \tau(t)\right)\right), u(t) \in Rd \qquad (17)$$

Where, delay $\tau>0$ is a constant, *t* denotes time, *u(t)* denotes error rate, delay $\tau(t)\geq 0$ is a given function and d is the matrix calculated for each delay R.

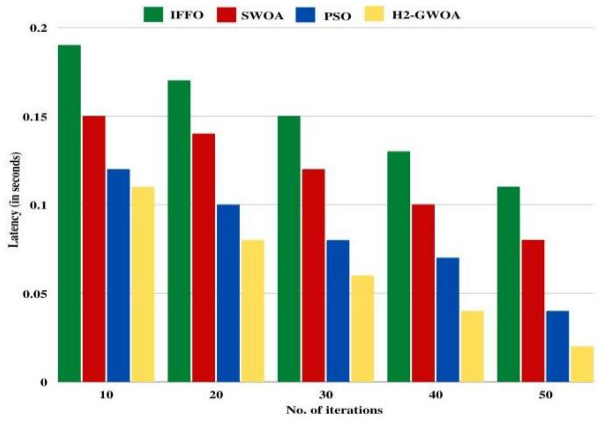

Figure 2. Latency vs. no. of iterations.

Figure 2 demonstrates that our proposed approach results in achieved low latency vs. the number of iterations compared to existing methods. If iteration number=10, when compared to existing techniques such as IFFO, SWOA, and PSO, the latency is 0.19, 0.15, and 0.12 correspondingly, but the proposed H2-GWOA latency is 0.11. If iteration number=20, when compared to existing techniques such as IFFO, SWOA, and PSO, the latency is 0.17, 0.14, and 0.10 correspondingly, but the proposed H2-GWOA latency is 0.08. If iteration number=30, when compared to existing techniques such as IFFO, SWOA, and PSO, the latency is 0.15, 0.12, and 0.08 correspondingly, but the proposed H2-GWOA latency is 0.06. If iteration number=40, when compared to existing techniques such as IFFO, SWOA, and PSO, the latency is 0.13, 0.10, and 0.07 correspondingly, but the proposed H2-GWOA latency is 0.04. If iteration number=50, when compared to existing techniques such as IFFO, SWOA, and PSO, the latency is 0.11, 0.08, and 0.04 correspondingly, but the proposed H2-GWOA latency is 0.02.

4.2. Makespan

Makespan can be defined as the maximum completion time of all tasks consumed by any virtual machine. The minimization of makespan improves the performance by allocating the exact resources on the specific node. If the desired node is determined, then the energy consumption per node can also be measured. The makespan is an important factor to be considered for cloud users and resource manager [15]. By selecting the suitable collection of jobs to allocate to VMs, the goal is to minimize the makespan to the bare essentials. Let (1, 2, 3, ... X_{ab}, y, n, y=1, 2, 3,..., n) be the efficient implementation for running the *bth* task on the *ath* VM; Here, *'a'* indicates the quantity of VMs, and 'b' represents the variety of tasks. The execution time (19) of *jth* VM is based on the decision variable X_{ab} that is denoted (18),

$$X_{ab} = \{1 if T_i \in VM_j \mid 2 if T_i \notin VM_j\} \qquad (18)$$

Where the set of all tasks, $T_i = \{T_1, T_2, T_3, T_4, \ldots, T_n\}$.

$$ET_y = \sum_{y=1}^{n} \left(X_{ab} * \left(lh_b / ps_a\right)\right) \qquad (19)$$

Where, lh_b=Length of the bth task, ps_a=Processing speed of the ath virtual machine. The makespan's (*MS*) fitness value is calculated (20),

$$MS = max\left\{ET_y, \text{ for all tasks } T_i \text{ is mapped to } VM_j\right\} \qquad (20)$$

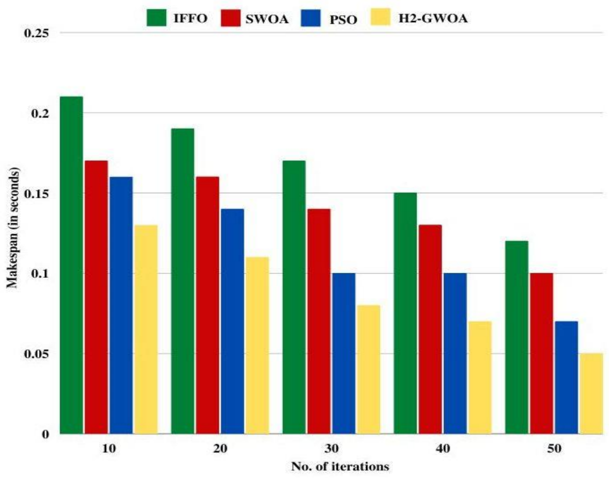

Figure 3. Makespan vs. no. of iterations.

Figure 3 illustrates a comparison of makespan based on the number of iterations. When compared to existing methods, our proposed method H2-GWOA is better. If iteration number=10, when compared to existing techniques such as IFFO, SWOA, and PSO, the makespan is 0.21, 0.17, and 0.16 correspondingly, but the proposed H2-GWOA makespan is 0.13. If iteration number=20, when compared to existing techniques such as IFFO, SWOA, and PSO, the makespan is 0.19, 0.16, and 0.14 correspondingly, but the proposed H2-GWOA makespan is 0.11. If iteration number=30, when compared to existing techniques such as IFFO, SWOA, and PSO, the makespan is 0.17, 0.14, and 0.10 correspondingly, but the proposed H2-GWOA makespan is 0.08. If iteration number=40, when compared to existing techniques such as IFFO, SWOA, and PSO, the makespan is 0.15, 0.13, and 0.10 correspondingly, but the proposed H2-GWOA makespan is 0.07. If iteration number=50, when compared to existing techniques such as IFFO, SWOA, and PSO, the makespan is 0.12, 0.10, and 0.07 correspondingly, but the proposed H2-GWOA makespan is 0.05.

4.3. Resource Consumption

The resource consumption is unstable with the use of smart devices, so the dynamic pricing quoted in the SLA is mandatory for both cloud users and service providers. The concurrent usage of resource consumption by cloud users deals with the efficiency in handling the cloud resources. The increased pricing on resource consumption has occurred for the service providers due to the maximum utilization of the same cloud resources [3]. The consumption of resources (β) is expressed in the Equation (21) which represents the percentage of the total number of resources available divided by the total amount of resources allocated.

$$\beta = \frac{C}{RA} \tag{21}$$

Where, C refers to the total number of resources allocated and RA refers to the total number of resources available.

Figure 4 illustrates a comparison of resource usage based on the number of iterations. Compared to conventional methods, our proposed H2-GWOA method used the most system resources. If iteration number=10, when compared to existing techniques such as IFFO, SWOA, and PSO, the resource usage is 0.085, 0.12, and 0.175 correspondingly, but the proposed H2-GWOA resource usage is 0.185. If iteration number=20, when compared to existing techniques such as IFFO, SWOA, and PSO, the resource usage is 0.09, 0.14, and 0.1 correspondingly, but the proposed H2-GWOA resource usage is 0.16. If iteration number=30, when compared to existing techniques such as IFFO, SWOA, and PSO, the resource usage is 0.01, 0.125, and 0.085 correspondingly, but the proposed H2-GWOA resource usage is 0.15. If iteration number=40, when compared to existing techniques such as IFFO, SWOA, and PSO, the resource usage is 0.08, 0.1, and 0.075 correspondingly, but the proposed H2-GWOA resource usage is 0.125. If iteration number=50, when compared to existing techniques such as IFFO, SWOA, and PSO, the resource usage is 0.07, 0.125, and 0.15, correspondingly, but the proposed H2-GWOA resource usage is 0.175.

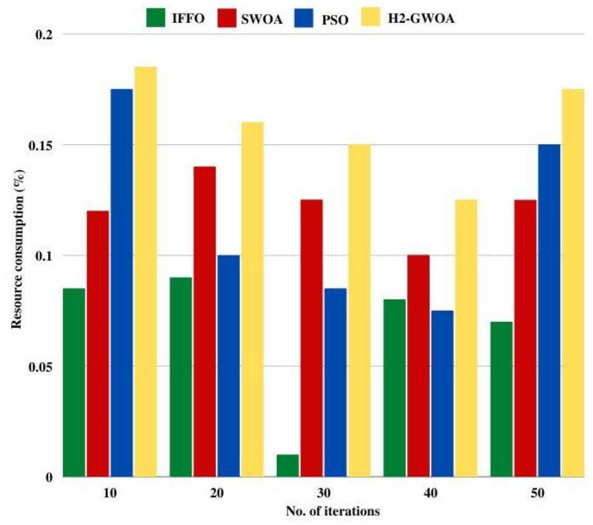

Figure 4. Resources consumption vs. no. of iterations.

4.4. Memory Consumption

Memory consumption (Mcon) deals with allocating memory to the data nodes where the data is transferred from the cloud user to the resource manager for computation in virtual machines. This proposed scheduler reduces the memory utilization based on priority task execution, dynamic workloads computation, and even with a huge amount of data transfer on the cloud environment [10]. Memory consumption is defined as the ratio of the amount of memory needed to perform an assignment to the total amount of memory available in the cloud, as calculated by the Equation (22),

$$Mcon = \sum_{p=1}^{y} \left(\frac{s_p}{q_p} \right) \tag{22}$$

Where, s_p denotes the task required the use of memory and q_p shows the total amount of memory that is available.

Figure 5 depicts a comparison of memory usage based on the number of iterations performed. This graph shows that the proposed H2-GWOA technique used the most RAM compared to conventional methods. If iteration number=10, when compared to existing techniques such as IFFO, SWOA, and PSO, the memory usage is 0.03, 0.07, and 0.14 correspondingly, but the proposed H2-GWOA memory usage is 0.18. If iteration number=20, when compared to existing techniques such as IFFO, SWOA, and PSO, the memory usage is 0.04, 0.08, and 0.09 correspondingly, but the proposed H2-GWOA memory usage is 0.12. If iteration number=30, when compared to existing techniques such as IFFO, SWOA, and PSO, the memory usage is 0.01, 0.02, and 0.03 correspondingly, but the proposed H2-GWOA memory usage is 0.15.

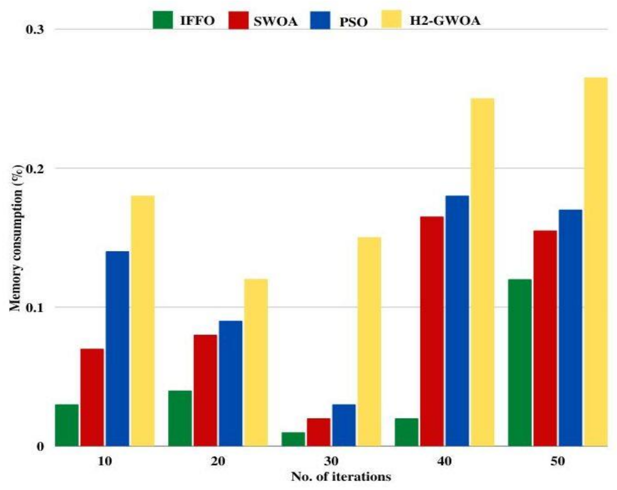

Figure 5. Memory consumption vs. no. of iterations.

If iteration number=40, when compared to existing techniques such as IFFO, SWOA, and PSO, the memory usage is 0.02, 0.165, and 0.18 correspondingly, but the proposed H2-GWOA memory usage is 0.25. If iteration number=50, when compared to existing techniques such as IFFO, SWOA, and PSO, the memory usage is 0.12, 0.155, and 0.17 correspondingly, but the proposed H2-GWOA memory usage is 0.265.

4.5. Skewness

The different number of iterations for various workload combinations helps to improve the resource utilization that minimizes the skewness (p_n). This proposed algorithm restricts the over-utilized resources gradually and saves the server energy. Equation (23) states that it is the measure used to detect the cloud server's uneven resource usage,

$$P_n = \left(\frac{Q}{Q_n} - 1\right)^2 \sqrt{2} \tag{23}$$

Where, Q is denoted random variable and Q_n is denoted as a number of variables in distribution.

Figure 6 shows a performance comparison of skewness with the number of repetitions. This graph shows that the proposed H2-GWOA technique has the least skewness among the conventional methods. If iteration number=10, when compared to existing techniques such as IFFO, SWOA, and PSO, the skewness is 0.11, 0.05, and 0.02 correspondingly, but the proposed H2-GWOA skewness is 0.01. If iteration number=20, when compared to existing techniques such as IFFO, SWOA, and PSO, the skewness is 0.05, 0.01, and 0.009 correspondingly, but the proposed H2-GWOA skewness is 0.005. If iteration number=30, when compared to existing techniques such as IFFO, SWOA, and PSO, the skewness is 0.015, 0.013, and 0.01 correspondingly, but the proposed H2-GWOA skewness is 0.009. If iteration number=40, when compared to existing techniques such as IFFO, SWOA, and PSO, the skewness is 0.17, 0.02, and 0.01 correspondingly, but the proposed H2-GWOA skewness is 0.008. If iteration number=50, when compared to existing techniques such as IFFO, SWOA, and PSO, the skewness is 0.15, 0.03, and 0.01 correspondingly, but the proposed H2-GWOA skewness is 0.005.

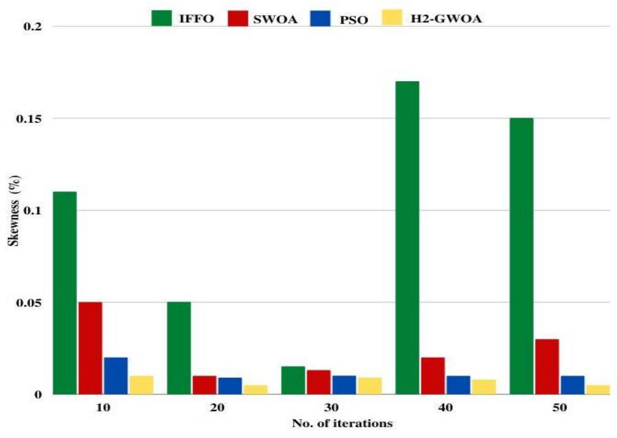

Figure 6. Skewness vs. no. of iterations.

References

[1] Aggarwal A., Dimri P., Agarwal A., Verma M., and Alhumyani H., "IFFO: An Improved Fruit Fly Optimization Algorithm for Multiple Workflow Scheduling Minimizing Cost and Makespan in Cloud Computing Environments," *Mathematical Problems in Engineering*, vol. 2021, pp. 1-9, 2021. https://doi.org/10.1155/2021/5205530

[2] Calheiros R., Ranjan R., Beloglazov A., Rose C., and Buyya R., "Cloudsim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Software Practice and Experience*, vol. 41, no. 1, pp. 23-50, 2011. https://doi.org/10.1002/spe.995

[3] Cao B., Wang K., Xu J., Hou C., and Fan J., "Dynamic Pricing for Resource Consumption in Cloud Service," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1-11, 2018. Https://Doi.Org/10.1155/2018/4263831

[4] Chen X., Wang H., Ma Y., Zheng X., and Guo L., "Self-Adaptive Resource Allocation for Cloud-Based Software Services Based on Iterative Qos Prediction Model," *Future Generation Computer Systems*, vol. 105, pp. 287-296, 2020. https://doi.org/10.1016/j.future.2019.12.005

[5] Chou L., Chen H., Tseng F., Chao H., and Chang

Y., "DPRA: Dynamic Power-Saving Resource Allocation for Cloud Data Center Using Particle Swarm Optimization," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1554-1565, 2018. doi: 10.1109/JSYST.2016.2596299.

[6] Galetsi P., Katsaliaki K., and Kumar S., "Big Data Analytics in Health Sector: Theoretical Framework, Techniques and Prospects," *International Journal of Information Management*, vol. 50, pp. 206-216, 2020. https://doi.org/10.1016/j.ijinfomgt.2019.05.003

[7] Haghighi A., Heydari S., and Shahbazpanahi S., "Dynamic Qos-Aware Resource Assignment in Cloud-Based Content-Delivery Networks," *IEEE Access*, vol. 6, pp. 2298-2309, 2018. doi: 10.1109/ACCESS.2017.2782776.

[8] Horri A., Mozafari M., and Dastghaibyfard G., "Novel Resource Allocation Algorithms to Performance and Energy Efficiency in Cloud Computing," *The Journal of Supercomputing*, vol. 69, no. 3, pp. 1445-1461, 2014. https://doi.org/10.1007/s11227-014-1224-8

[9] Kumar M. and Sharma S., "PSO-COGENT: Cost and Energy Efficient Scheduling in Cloud Environment with Deadline Constraint," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 147-164, 2018. https://doi.org/10.1016/j.suscom.2018.06.002

[10] Lakkadwala P. and Kanungo P., "Memory Utilization Techniques for Cloud Resource Management in Cloud Computing Environment: A Survey," *International Conference on Computing Communication and Automation*, Greater Noida, pp. 1-5, 2018. doi: 10.1109/CCAA.2018.8777457

[11] Lee H., Jeong Y., and Jang H., "Performance Analysis Based Resource Allocation for Green Cloud Computing," *The Journal of Supercomputing*, vol. 69, no. 3, pp. 1013-1026, 2014. https://doi.org/10.1007/s11227-013-1020-x

[12] Lim A., Ma H., Rodrigues B., Tan S., and Xiao F., "New Meta-Heuristics for the Resource-Constrained Project Scheduling Problem," *Flexible Services and Manufacturing Journal*, vol. 25, no. 1-2, pp. 48-73, 2013. https://doi.org/10.1007/s10696-011-9133-0

[13] Liu H., Liu S., and Zheng K., "A Reinforcement Learning-Based Resource Allocation Scheme for Cloud Robotics," *IEEE Access*, vol. 6, pp. 17215-17222, 2018.

[14] Mishra M., Das A., Kulkarni P., and Sahoo A., "Dynamic Resource Management Using Virtual Machine Migrations," *IEEE Communication Magazine,* vol. 50, no. 9, pp. 34-40, 2012. 10.1109/MCOM.2012.6295709

[15] Peng Z., Barzegar B., Yarahmadi M., Motameni H., and Pirouzmand P., "Energy-Aware Scheduling of Workflow Using a Heuristic Method on Green Cloud," *Scientific Programming*, vol. 2020, pp. 1-14, 2020.

[16] Qi L., Chen Y., Yuan Y., Fu S., Zhang X., and Xu X., "A Qos-Aware Virtual Machine Scheduling Method for Energy Conservation in Cloud-Based Cyber-Physical Systems," *World Wide Web*, vol. 23, no. 2, pp. 1275-1297, 2020. https://doi.org/10.1007/s11280-019-00684-y

[17] Shukur H., Zeebaree S., Zebari R., Zeebaree D., Ahmed O., and Salih A., "Cloud Computing Virtualization of Resources Allocation for Distributed Systems," *JASTT*, vol. 1, no. 3, pp. 98-105, 2020.

[18] Stanik A., Koerner M., and Lymberopoulos L., "SLA-driven Federated Cloud Networking: Quality of Service for Cloud-Based Software Defined Networks," *Procedia Computer Science*, vol. 34, pp. 655-660, 2014. https://doi.org/10.1016/j.procs.2014.07.093

[19] Subhash L. and Udayakumar R., "Sunflower Whale Optimization Algorithm for Resource Allocation Strategy in Cloud Computing Platform," *Wireless Personal Communications*, vol. 116, no. 4, pp. 3061-3080, 2021. https://doi.org/10.1007/s11277-020-07835-9

[20] Tian W., He M., Guo W., Huang W., and Shi X., "On Minimizing Total Energy Consumption in the Scheduling of Virtual Machine Reservations," *Journal of Network and Computer Applications*, vol. 113, pp. 64-74, 2018. https://doi.org/10.1016/j.jnca.2018.03.033

[21] Tong S., Liu Y., Cho H., Chiang H., and Zhang Z., "Joint Radio Resource Allocation in Fog Radio Access Network for Healthcare," *Peer-to-Peer Networking and Applications*, vol. 12, no. 5, pp. 1277-1288, 2019. https://doi.org/10.1007/s12083-018-0707-4

[22] Xu X., Tang M., and Tian Y., "Qos-Guaranteed Resource Provisioning for Cloud-Based Mapreduce in Dynamical Environments," *Future Generation Computer Systems*, vol. 78, pp. 18-30, 2018. https://doi.org/10.1016/j.future.2017.08.005

[23] Xu X., Fu S., Cai Q., Tian W. and Liu W., "Dynamic Resource Allocation for Load Balancing in Fog Environment," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1-15, 2018. https://doi.org/10.1155/2018/6421607

[24] Yang C., Chen S., Liu J., Chan Y., Chen C., and Verma V., "An Energy-Efficient Cloud System with Novel Dynamic Resource Allocation Methods," *Journal of Supercomput*, vol. 75, no. 8, pp. 4408-4429, 2019. https://doi.org/10.1007/s11227-019-02794-w

Aarthee Selvaraj is a research scholar in the Dept. of EEE at UCE, BIT Campus, Anna University, Trichy, TN, India. She has completed M.C.A. and M.E. (CSE) at Anna University, Chennai. Her research interests include cloud computing, energy-aware computing, data mining, distributed systems, and big data analytics.

Prabakaran Rajendran received his B.E. (E&I) and M.Tech. (Embedded Systems) from Sastra University, Thanjavur, TN, India, between 2000 and 2005, respectively. He has also completed his Ph.D. from Anna University, Chennai. He has been working as an Assistant Professor in the Dept. of EEE, UCE, BIT Campus, Anna University, Trichy, TN, India, since 2007. His area of specialization includes embedded systems, microprocessor and microcontroller based systems, WSN, renewable energy, and fiber optics.

Kanimozhi Rajangam completed her M.E. and Ph.D. from Anna University, Chennai. She has 20 years of teaching experience and is currently working as the Assistant Professor in the Dept. of EEE, UCE, BIT Campus, Anna University, Trichy, TN, India. Her area of specialization includes microprocessor and microcontroller, power systems, optimization techniques in electrical engineering, and renewable energy.