A Predictive Priority-Based Dynamic Resource Provisioning Scheme With Load Balancing in Heterogeneous Cloud Computing

MAYANK SOHANI [ID], (Member, IEEE), AND S. C. JAIN

Department of Computer Science and Engineering, Rajasthan Technical University, Kota 324010, India

Corresponding author: Mayank Sohani (sohanimayank@gmail.com)

ABSTRACT In cloud computing, resource provisioning is a key challenging task due to dynamic resource provisioning for the applications. As per the workload requirements of the application's resources should be dynamically allocated for the application. Disparities in resource provisioning produce energy, cost wastages, and additionally, it affects Quality of Service (QoS) and increases Service Level Agreement (SLA) violations. So, applications allocated resources quantity should match with the applications required resources quantity. Load balancing in cloud computing can be addressed through optimal scheduling techniques, whereas this solution belongs to the NP-Complete optimization problem category. However, the cloud providers always face resource management issues for variable cloud workloads in the heterogeneous system environment. This issue has been solved by the proposed Predictive Priority-based Modified Heterogeneous Earliest Finish Time (PMHEFT) algorithm, which can estimate the application's upcoming resource demands. This research contributes towards developing the prediction-based model for efficient and dynamic resource provisioning in a heterogamous system environment to fulfill the end user's requirements. Existing algorithms fail to meet the user's Quality of Service (QoS) requirements such as makespan minimization and budget constraints satisfaction, or to incorporate cloud computing principles, i.e., elasticity and heterogeneity of computing resources. In this paper, we proposed a PMHEFT algorithm to minimize the makespan of a given workflow application by improving the load balancing across all the virtual machines. Experimental results show that our proposed algorithm's makespan, efficiency, and power consumption are better than other algorithms.

INDEX TERMS Cloud computing, load balancing, Predictive Priority-based Modified Heterogeneous Earliest Finish Time, Quality of Service, Service Level Agreement, scheduling, virtual machine.

I. OBJECTIVE

We propose a PMHEFT model in this paper, based on the priority queue for the prediction base. In load balancing network systems (cloud computing), this prediction-based PMHEFT approach will be easy and more effective. Our proposed scheme is working for both dynamic and static load balancing.

II. INTRODUCTION

Load balancing is a crucial factor in optimizing cloud resources, i.e., compute, storage, and networking [1]. When cloud consideration takes place, usually load balancing technique highly requires to distribute the task load among

The associate editor coordinating the review of this manuscript and approving it for publication was Anton Kos [ID].

different VMs. Without it, the unequal burdens in server formation may cause asset wastage, execution corruption, and SLA violation [2]. Accordingly, using the right load balancing strategy can improve servers' usage and give better assurance of Quality of Service (QoS) [3]. A portion of the specialists in this field centers around the virtual machine distribution or virtual machine movement to accomplish load adjusting [4].

A. MOTIVATION

In IT industries, different cloud providers fulfill QoS to end clients as indicated by their requirements. As a result, it brings about a varying number of clients over a period. It embodies that static resource provisioning may tend towards inefficiency to resource handling. It shows that clients may get resources plentifully at some time, and at any other time,
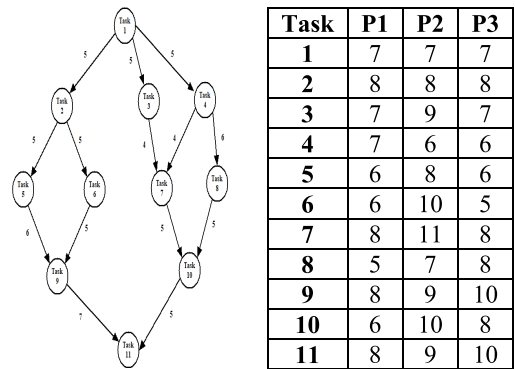
resources might be inadequate, which represents the load imbalance and poor QoS with higher cost. Many researchers explicate plenty of techniques to manage the addressed issue with reduced cost and higher resource utilization. The proposed work addressed a diverse facet of resource provisioning in the cloud computing domain, and the work satisfies the client's workload need based on predicted resource for workload in the cloud. To fulfill the client's needs, efficient resource provisioning done using prediction with enhanced performance. The work is proposed for provisioning the services based on the QoS approach to accomplish user satisfaction. As user's need predicted by our proposed algorithm, sufficient resources can be provisioned to serve the user request in due time.

B. CONTRIBUTION
When the server consolidation process occurs, it must be very effective for energy consumption and operating cost estimation. It may also cause servers' performance degradation if it is not properly defined [5], [6]. From another perspective, server consolidation to solidify virtual machines on certain servers, making it more probable for machine over-burden to happen. Then again, the shutdown time and correspondence cost brought by server consolidation is unavoidable and should be taken care of appropriately to fulfill the conveyed QoS [7].

Researchers were prompted to potentially contribute heuristic algorithms because of the complete NP problem [8]. In this research paper, the algorithm for heterogeneous computer models has proposed an efficient critical task prediction algorithm for distributing the load over the cloud server. To evaluate this proposed simulation algorithm's performance, we use a cloud simulator to demonstrate the algorithm's efficiency. The ease and speed of application of the prediction-based algorithm have been proved efficient. This scheme is useful for using a load balancing system via HEFT Prediction Priority Queue Loading. In load balancing, these tiny HEFT requests need to load for completing the data processing, acquisition, and transmission. This is the most influential and significant task of load balancing. Earlier, when such HEFT requests were used, it was an extrusive task to retain the algorithm's finite size and raise loads due to many types of load requests [9]. The directed acyclic graph (DAG) schedules problems effectively solved by the HEFT algorithm on heterogeneous systems due to its low running time and stout performance. It gives a steady performance for a wide structure range of graphs. The HEFT algorithm's limitation is that it works on the techniques that all are static approaches of mapping the problem, which deals only for static conditions. It can easily fail in complex situations to find the optimal scheduling [10].

An application can be embodied by a DAG, $G = (V, E)$, as shown in Figure 1, where V and E represent a set of v nodes and e-communication edges between tasks. The node $vi \in V$ shows an application task instruction executed on the same machine. The task n_i should execute completely

Task	P1	P2	P3
1	7	7	7
2	8	8	8
3	7	9	7
4	7	6	6
5	6	8	6
6	6	10	5
7	8	11	8
8	5	7	8
9	8	9	10
10	6	10	8
11	8	9	10

FIGURE 1. An example of DAG and a computation time matrix of the tasks for each processor.

before starting the task n_j for each $e(i, j) \in E$ shows the task's dependency constraints. The DAG is completed by matrix W that is a computation cost matrix $v \times p$, where v, p represents the number of tasks and number of processors in the system [11], [12].

The proposed Predictive Priority-based Modified Heterogeneous Early Finish Time (PMHEFT) algorithm resolves the prominent obstacle of load balancing and ensures deadlock problems never arise using a prediction priority queue.

The main contribution to this study includes:

- Performance evaluation of existing algorithms for static and dynamic scheduling criteria in cloud computing.
- Performance evaluation of existing algorithms for energy consumption in cloud computing.
- Proposing and implementing a predictive priority-based algorithm on cloud computing.
- Proposing and implementing a new load balancing scheme for cloud computing.

C. ORGANIZATION
The paper is organized as follows: Section 2 gives the literature review of the state-of-the-art algorithms proposed for resource allocation and load balancing. Section 3 briefly describes our proposed model. The simulation setup and detailed result analysis have been discussed in section 4. Section 5 concludes the paper and specifies future work directions.

III. RELATED WORK
This section's key objective is to identify the diverse load-balancing studies/articles released over time and their significance by increasing cloud usage. Several load balancing strategies have been proposed to date, which need to be measured based on the different load balancing parameters. This helps in understanding needs and conflicts with current load balancing approaches to ensure Quality of Service based facilities [13]. The resource planning problem has been extensively explored in a heterogeneous system where the processors differ in computational abilities and communicate over a network underlying them. A group of investigators suggested papers in this area. Typically, the scheduling

problem is NP-complete, so that the desire to be optimal can lead to higher overhead planning of resources [14]. The negative result encourages heuristic approaches to resolve the problem of resource scheduling [15].

The author [16] evaluated numerous current load balancing systems and listed complex and mixed subdomains. The author did not discuss task-based load balancing. They defined these strategies' behavior based on the above-mentioned criteria and their advantages, challenges, and drawbacks. The concerns developed by these algorithms were to create more aware algorithms to power consumption, reduce resource usage, and make load balancing strategies more energy aware. Many task scheduling algorithms have been proposed to minimize the makespan of meta tasks in heterogeneous surroundings, such as Minimum Completion Time (MCT), Minimum Execution Time (MET), and Opportunistic Load Balancing (OLB) [17]. The opportunistic load balancing randomly assigns each task to the VM that is supposed to be available regardless of the task's expected time to run on the machine, making all VMs as busy as possible. One of the advantages of OLB is its versatility. The MET heuristic randomly allocates user tasks to that VMs, which takes less execution time. However, this heuristic faces load unbalancing because it does not consider VM availability and VM current load. The MCT works based on assigning tasks to the VM, which least task completion time of allocated tasks. This provides such tasks to be assigned VMs that don't have less execution time for them [18].

The author [19] evaluated the grid network's load balancing and static task distribution methods, concentrating on delay in communication, task migration, and mean response time for user-submitted tasks. VM placement performs on physical machines at data centers. VM placement methods aim to reduce wastage of resources along with minimization of power consumption. The VM placement problem solution needs to entertain based on VM management in heterogeneous environments, the number of VM on a server, and VM configuration complexities in a large-scale data center [20]. The facility of migration permits tasks to move towards underloaded VM from overloaded VM [21]. Although, this technique does not consider task makespan and completion time. A user task is first stored in the queue manager. Then the priority of a task is calculated, and for the execution of the task, suitable resource gets provisioned only if it belongs to a repeated task category. Every new task gets analyzed and submitted in the on-demand queue, and the algorithm evaluates best match resources for on-demand queue listed user tasks [22].

The heterogeneous multi-cloud environment is currently a prevalent topic and challenging due to the functionality and varying capacities of cloud resources under heterogeneity. Task scheduling algorithms aimed to minimize the makespan and improve the average cloud utilization [23].

The Heterogeneous Earliest Finish Time First (HEFT) and Critical Path on a Processor (CPOP) are the two most popular ranking algorithms. Each task rank is calculated based on computation and average communication cost between current task and successor in HEFT ranking algorithms. In the CPOP algorithm, task ranking is done through upward and downwards task rank summation. The HEFT ranking algorithm performs better than the CPOP ranking algorithm. Immediate predecessor communication time is not considered in the HEFT ranking algorithm. The same two tasks that are dependent on each other might be scheduled on different resources and increase the schedule length [24], [25].

The author states that MHEFT "Modified Heterogeneous Early Finish Time for task scheduling in cloud environments." In this algorithm, the load balancing task scheduling finds the order in which the activity is completed. The next step is mapping the resources and tasks after that task is submitted for their completion to the cloud [26]. This proposed scheduling solved the load balancing problem. As this author proposed, the resources are to be allocated to appropriate tasks to decrease the execution time and increase resource utilization. Scheduling of tasks is the best approach to achieve maximum utilization of resources and economic efficiency [27].

The author presents a Generalized Critical Task Anticipation (GCA) algorithm in a heterogeneous computing environment for weighted directed acyclic graph (DAG) scheduling. This scheduling CA algorithm employs processor selection based on heterogeneous communication cost with task prioritization method [28]. The GCA algorithm is effective in terms of reduced scheduling cost and better performance [29].

Since load balancing refers to dealing with resources currently utilized below their defined minimum capacity, the computing resources get wasted due to not being utilized up to their optimal limit. When VMs are overloaded, the time is taken to complete allotted tasks increases, i.e., makespan increases. If VMs are less utilized, makespan decreases, and resource utilization cost increases because VMs are not extensively utilized [30]. Table 1 presents a concrete summary of the related work. Synthesis of the state-of-the-art has revealed that most of the resource provisioning and load balancing heuristic results in low resource utilization and load imbalance. Thus load balancing across VMs required to control makespan, usage, and energy consumption parameters.

IV. PROPOSED METHODOLOGY

Cloud users and cloud providers are two main cloud computing entities with their objectives about task execution through cloud resources. The cloud provider focuses on the utilization factor of resources and tries to achieve maximum. In contrast, the cloud user focuses on resources' performance and tries to achieve less makespan time. A task in a cloud environment has various characteristics, i.e., task length, expected execution time, priority, and emergency execution constraints over other tasks in the queue. For the understanding of the proposed model, all the variables and constants are listed in Table 2. The scheduling mechanism's load balancing is

TABLE 1. Summary of the related work.

Heuristics	Strengths	Weaknesses
EPRD [31]	Effectively improve VM utilization and scheduling performance	No fairness in energy consumption and security issues not handled
ECOS [32]	The cost minimization without comprising the deadline constraint and light-weight complexity algorithm	Poor resource utilization and load imbalance
F-MRSQN [33]	Improves the resource scheduling efficiency and reduces the response time	Privacy-aware resource scheduling is not considered
DPLS [34]	Minimize the makespan of the job and effectively reduce the scheduling length	Load balancing is not considered
DRP [35]	Improve power efficiency, throughput and system fairness over memory partitioning	Limited to single ISA systems and poor resource utilization
SJF-MMB [36]	Optimal in a heavy loaded and highly dynamic environment	System efficiency is not considered
ACOPS [37]	Fair treatment of load balancing in a dynamic environment	Poor memory management
MHEFT [38]	Minimize makespan, communication cost and reduces energy consumption	Load imbalance and implementation complexity not improved
AMS [39]	Obtain better task completion time and achieve load balancing	More scheduling overhead
HLBZID [40]	Higher resource utilization, Minimum makespan and low costs	Virtual Machine live migration and energy factors are not considered.
DHSJF [41]	For heterogeneous cloudlets it gives better results in terms of CPU utilization, energy efficiency and total execution time.	Service Level Agreement violations are not considered
MRLB[4]	Provide better results for Service Level Agreement (SLA), response time, energy consumption and load balancing	More scheduling overhead due to sorted, reordered task queue in each scheduling decision and workload prediction is not considered.
RALBA [42]	Reduced makespan, increased resource utilization, and higher throughput produces for compute intensive, non-preemptive and independent tasks using a load balanced distribution of workload.	SLA aware scheduling is not considered.
SLA-RALBA [43]	Reduced execution time and cost along with resource utilization enhancement.	Communication cost is not considered
IWRR [44]	Under varying load patterns, it gives high performance along with VMs better utilization.	SLA aware scheduling and load balancing are not considered
FLASDIWA [45]	Minimum makespan and maximization in average cloud utilization	Dynamic load balancing is not considered
CBLBCC [46]	Improved waiting time, execution time, turnaround time and throughput	SLA and energy being consumed is not considered
GRP-HEFT [47]	Minimizing the makespan and budget constraint cost model	Energy consumption, Load balancing and SLA are not considered
ACM [48]	Efficient proactive load balancing and provide stability	System efficiency is not considered
IHPCAPC [49]	Reduce execution time and support dynamic load balancing	SLA violation is not considered
HALO [50]	Across various load and cluster configurations, it reduces the response time without additional resource or energy overhead	Limiting scalability and SLA violation are not considered

an important task execution parameter to achieve optimal utilization of cloud resources [51].

The proposed approach is based on the Prediction of load demand rate and emergency of cloud request. This novel approach will apply for cloud computing task requests for load balancing, and this load request will always be in working condition due to the PMHEFT algorithm. Figure 2 represents the block diagram of the proposed system. Cloud

TABLE 2. Notations and description.

Notation	Description
$LRj(t)$	Load demand rate
LE	Allow more request to get load according to Prediction Priority Queue load request
LNe	Total loading emergency request
LP(j)	Loading load request at index j in Priority Queue
L_α	Variable factor ($0 < L\_\alpha < 1$)
LNr	The Serial number of Prediction Priority Queue load
LT	Loading of the workload request
LTi	Prediction Priority Queue arrival time for workload request j
LTremain	Prediction Priority Queue remaining length
LN	Total load request
Tn	Time when queue arrives at load request LN
τLN	Loading time for Prediction Priority Queue load
Vm	Time to load requests in a Prediction based Queue
LEth	Load request threshold
L_β	Control factor ($0 < L\_\beta < 1$)
c	Loading rate of load request
Lη	PMHEFT capacity
Ew	Prediction Priority Queue full load
qm	Load consumed
Lead	Indicates the Prediction Priority Queue load has sufficient load i.e. Adaptive load threshold
Les	Prediction-based Priority Queue load full efficiency
ELN	Represent adoptive load loading
Emth	Load request threshold
LE(t)	Residual load of Predictive Priority Queue at time t

computing, communication will help to render convenience to the load balancing system and distribute loads from within the cloud platform. Our proposed approach will predict load demand rate, emergency of cloud request, and the prediction that our Prediction-based Priority Queue framework will work. The real-time dynamic change of load demand on the network would adopt this framework.

Proposed Model Scheme Steps

Step 1: Calculate load demand rate for cloud request in Prediction Scheme.

Step 2: Add load request into the Prediction Priority Queue based on load demand rate and emergency request.

Step 3: Calculate loading demand queue load and loading threshold for Prediction-based Priority Queue.

Step 4: Calculate loading demand queue predicted based priority loading sequence based on emergency or highest demand request.

We will discuss the prediction-based priority algorithm in this segment. First, we assumed a threshold for the efficient load to avoid the deadlock situation, and then we discussed the algorithm based on prediction-based priority in detail.

Priority Based Loading Scheme for Scheduling

In this section, we will present PMHEFT to solve the above-mentioned issues of the prediction priority-based loading scheme. The scheme mainly comprises three parts, the next location of the loading DAG Prediction Priority Queue Load for the inclusion in the list, potential loading criterion conditions, and the efficient loading load threshold.

We'll discuss this in detail below.

DAG Prediction Priority Queue Add Load Request Selection

When we receive multiple requests for loading, the workload request selects a predictive queue, and this loading scheme will work according to the predictive priority scheme. In this scheme, we consider the demand rate of workload and emergency as the two important factors to select the loading DAG Prediction Priority Queue Load for added into the queue. Firstly, we are proposing the prediction model of load demand rate for cloud requests. Then we would present the method for loading DAG Prediction Priority Queue Load selection add into the queue.

In this section we are describing the scheme given below-

A. LOAD DEMAND RATE FOR CLOUD REQUEST IN PREDICTION SCHEME

If the request workload is below the predetermined LEth level, our scheme sends a request for loading to attach a request thread. The request loading order includes the load request-id and load usage of the respective prediction priority queue. The workload priority list load demand rate of the prediction base request and current load demand rates are determined depending on the workload request. The workload request load demand rate is assumed to be LRj. As seen in equation (1), we present a prediction of workload usage in two states: load and no load. L_α is a variable factor in the equation ($0 < L\_\alpha < 1$). When the workload request is not loaded, it has to contend with other queue requests. Therefore, to measure the loading sequence, we can use the cloud request load usage average of any single time. When the Load request is in loading mode, the workload series is calculated, and instead of measured very fairly, can use the current actual load demand rate. When the loading period is finished, repeat the equation (1). The equation estimates the current loading Prediction Priority Queues Load Demand Rate until the next loading Load request is chosen.

$$LRj(t + \Delta) = (L\_\alpha) LRj(t) + (L\_\alpha)LEj(t - \Delta) - LEj(t) \Delta LEj(t - \Delta) - LEj(t) \Delta \quad (1)$$

B. ADD INTO PREDICTION PRIORITY QUEUE FOR LOAD REQUEST ALGORITHM

It is necessary to remember that our proposed workload usage prediction model relies on the time taken to adjust
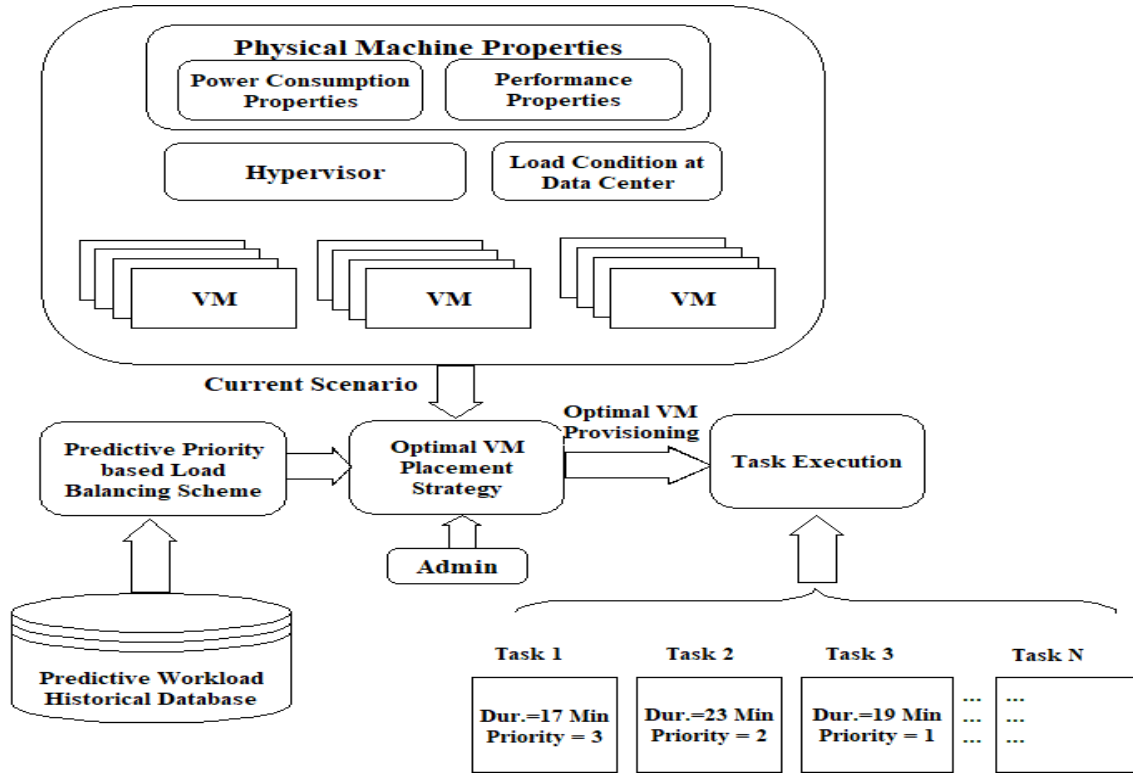
FIGURE 2. Proposed system block diagram.

the VM load for each workload request at various times and add it into the prediction priority queue for the loading request algorithm. The algorithm input and output information are given below, along with adding a load request into the Prediction Priority Queue based on load demand rate and emergency request. In this algorithm, LN is the total workload request $LRj(t)$ based on a load demand rate. We calculate the minimum weight corresponding load request. According to the sn, the second condition for emergency-based cloud requests into the prediction priority queue in ascending order is a sequence number. Total Load request LN, validate the $LRj(t)$, $LNe(j)$ condition based on-demand rate $LRj(t)$ are marked.

$$LP(j) = L\_\beta * LNe(j) + LNr(j) \qquad (2)$$

After that we calculate the condition values and according to that we calculate the minimum weight for loading load request. We evaluate this requirement prediction base

if (min LP(j))

{

Load Request Prediction Priority Queue Load corresponding t, the least LP(j) is added into the queue.

}

Prediction Priority Queue Load for adding into a queue. After checking the condition LN == 0. In our proposed model we calculate the user request load according to the emergencies that will be calculated by LP(j) using equation (2).

According to the emergency request value that represents LP(j), the actual load sequence queue of the load request is calculated. The sampling effects of the above two variables, decide the importance of LP(j). We will therefore obtain LP(j). A control factor L_β ($0 < L\_\beta$ to 1) was introduced to neutralise the impact of the emergency load, given the assumption that the emergency is relatively higher in the current situation than the load consumption rate. This will be calculated by formula. Here j is the index of the priority queue.

C. ALGORITHM FOR SELECTION OF LOAD REQUEST FOR ADDING INTO THE PREDICTION PRIORITY QUEUE DESCRIPTION

As defined above, when the workload request load demand rate is forecast, and a prediction priority queue load is loaded after deciding the Prediction Priority Queue scheme, the emergency between all requests from other loading cloud requests to the current queue request variant is considered.

i   Loading demand Queue Calculation

In the load algorithm, all workload requests are in a priority queue based on prediction. We denote LT as the loading of the workload request, and LTi is the prediction priority queue arrival time for workload request j. LT*remain* is the remaining prediction priority queue length. Assume there are LN requests that need to be loaded into the loading queue, then the loading cycle LT can be calculated by equation (3),

in which LTn is the time when queue arrives at workload request LN, τLN is the loading time for the prediction priority queue load and n is the emergence in the queue. LNe (LN, 0) is the time for the queue, returning from the last load request n to load. The period that queue arrives at prediction-based priority queue load calculation is done by equation (4).

$$LT = LTn + \tau LN + LNe\,(LN, 0)\,arrive + LTremain \quad (3)$$
$$LTi = LT - 1 + \tau i - 1 + LNe(i - 1, i) \quad (4)$$

As with the LNe indicating the load request sequence, equation (2) can be represented by the period that Prediction Priority Queue consumes on loading cloud request, in equation (4). The LNe loading emergency sequences are calculated according to equation (5) and *vm* is the time to load requests in a prediction-based queue.

$$LT = LNe * vm + \sum j = 1 * Ln * \tau j + LTremain \quad (5)$$
$$LNe = \sum j = 1 nLne\,(j - 1, k) + Lne\,(sn, 0) \quad (6)$$

All requests are in a prediction-based queue, and our scheme, ensuring the request's regular activity in each loading request queue. The residual load of request is calculated by equation (7), in which LRj is the load demand rate of the next workload request to add into the queue. Using the prediction, we also add the workload request and the load expended by the forecast priority queue load during the waiting time (LTj−LRj) is not more than the workload request threshold LEth. This scheme will prevent the workload cloud request would never die due to workload loss.

$$LEth - LRj \times (LTj - LRj) > 0 \quad (7)$$

As the Prediction based priority queue is full. Therefore, the relation between the load LEs of the request satisfies the equation (8). The loading rate of workload request is c, and lη is the PMHEFT capacity.

$$LEs = LEth - LRi \times (LTi\text{-}LRi) + L\tau i \times l\eta \times c \quad (8)$$

Next, we can identify the specifications that Predictive Priority Queue full load algorithm Ew needs to satisfy. To ensure that the Prediction Priority Queue scheme effectively completes the loading activities and ultimately returns to the queue in the last order, the total algorithm load of the Prediction-based Priority Queue is not less than the total load assigned to the request and total load consumed. The condition that Ew should satisfy is shown in equation (9), in which qm is the load consumed.

$$Ew \geq c \times \sum i = 1 * n\tau i + e \times qm \quad (9)$$

ii **Loading Threshold Calculation**

The LEad states that the adaptive load threshold means that the prediction-based priority queue load has ample loads. The submission shall be made to the queue in compliance with the threshold values. The importance of LEad falls between the prediction-based priority queue load full efficiency LEs and the load request threshold LEth. Using a

forecasting-based priority queue submission, we agree on an adaptive load threshold. Equation (10) is the calculation of the load threshold. The prediction-based priority queue load begins changing the LEth stage, which causes an emergency forecast priority queue load to be demanded.

$$LEad = (LEs - LEth) \times LN - LN\,\sqrt{} LN\,\sqrt{} + LEth \quad (10)$$

D. ALGORITHM FOR PRIORITY-BASED LOADING

The Algorithm is based on the Prediction Model and the Sensitive Cloud's request. The next prediction priority queue load selection for the priority information queue is provided in the load request algorithm. In this algorithm, the prediction is for the next workload request of the algorithm. Here the system will detect whether the load level is below or less than the pre-set threshold value or not for the workload cloud request. It is a center-based priority system, where request queue loading information is stored in the cloud-id of the corresponding workload request and its load demand rate. This algorithm gives the highest importance to the emergency or high demand cloud request, which prevents the deadlock condition when the request is overloaded, such as high demand rate priority. In this algorithm, a priority queue will be maintained using the mathematical model for prediction-based contemplation. Our algorithm worked on all types of cloud request load based on a prediction priority. In the proposed model, the loading interval will commence when the request value is below the threshold level, and the model load is at the predicted level. Here we demonstrate how to measure the threshold value.

ELN = this reflects an adaptive load

The Prediction-based Priority Queue Load has ample load to operate.

Emth = Load request threshold.

The ELN capacity value is between maximum algorithm capacity and threshold.

If an algorithm hits ELN instead, it will interrupt the loading phase and contribute to the loading period for certain requests with incredibly high load crises. The adaptive load threshold measurement is provided, and the prediction-based priority queue load starts loading when the load exceeds Emth (Maximum value) threshold value.

$$ELN = (Emth - Ereq) \times N - n - \sqrt{N} - \sqrt{} + Ereq \quad (11)$$

The next phase is the predictions-based time estimate with PMHEFT. For results, we adopt the Min-Min principle and predict VM status when it is available and usable for other tasks. Figure 3 represents the flow diagram of the proposed system.

V. SIMULATION RESULTS AND DISCUSSION

We have used NetBeans, Java, and WorkflowSim software's for simulation. The WorkflowSim extends the existing CloudSim simulator by providing a higher layer of workflow management. A general-purpose framework is needed to support workflow-based research that can execute
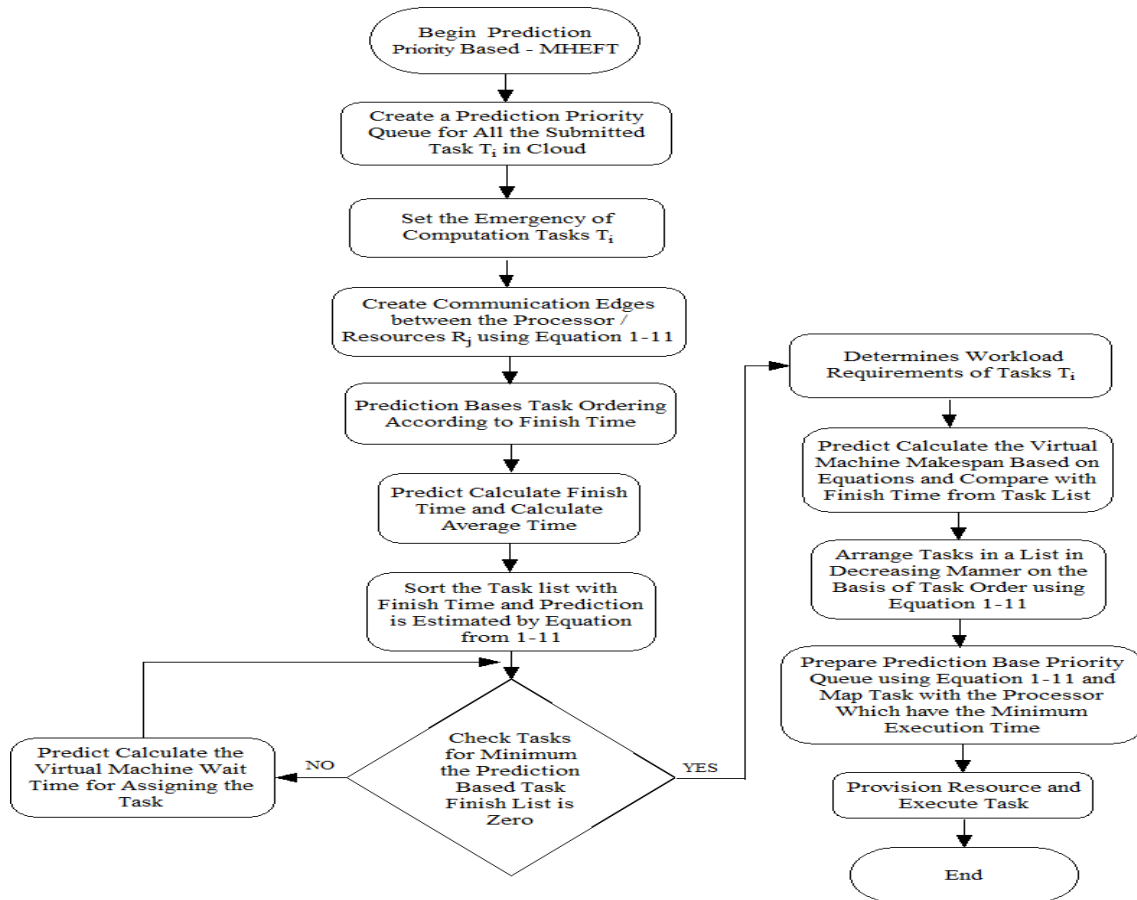
FIGURE 3. Proposed system flow diagram.

all workflows, features, and optimization techniques widely accepted. The CloudSim/GridSim simulators fail to support workflow simulation up to fine granularity, and these simulators only support static scheduling algorithms. WorkflowSim supports dynamic workflow algorithms. The consistent and perfect job-level execution model provided by WrokflowSim relies on CloudSim and improves simulation accuracy [52].

This simulation's system configuration is the 64-bit operating system- Windows 10, CPU- Intel(R) Core(TM) i3, 2.20 GHz speed, and RAM- 8 GB. The proposed algorithm developed using the Java language. The WorkflowSim simulator is used to evaluate the proposed algorithm's performance because workflowsim simulator supports large-scale scheduling, clustering, and provisioning studies. The outcomes of the HEFT, Modified HEFT, Cluster HEFT, Dynamic HEFT, and proposed Prediction based Priority MHEFT algorithm are discussed under three parameters, namely: length of time, speed, and effectiveness of the algorithm.

Comparison of the Algorithms Based on Below Performance Parameter:

1. Schedule Duration: The duration of the schedule (makespan) is the length of the schedule.

2. Total time of operation of an application.

Makespan = EFT- EST
   Where   EFT: Execution Finish Time
                   EST: Execution Start Time

3. Efficiency: Speedup is divided by the number of processors on each run.
4. Power Consumption: Total power used to execute a group of tasks.

For simulation, first, we set the standard for all algorithms in simulation. The Cloud Environment, VM, and Host specifications are given below in respective Tables 3,4,5 used for simulation of algorithms.

The Table 6 is showing the description about the performance factors that are improved from existing to proposed algorithms.

A. MAKESPAN COMPARISON CHART FOR FIVE DIFFERENT ALGORITHMS

The makespan results show the comparison of Predictive Priority-based Modified HEFT, HEFT, Modified HEFT, Cluster-based HEFT, and Dynamic HEFT algorithms are shown in Figure 4. It can even see that the total makespan time of workflows lengthens with increases as per submitted

Algorithm 1 Prediction Priority Based MHEFT

1. Build a Prediction Priority Queue for all the enrolled tasks Ti in Cloud.
2. Set the emergency of computing the tasks.
3. Prepare communication edges between Rj processor/resources using equations 1 to 11.
4. Predict task load ordering bases according to the time of completion.
5. Calculate finish time and the total time of task request.
6. Sort the task load list with a completion time of the task and determine prediction by equations 1 to 11 until the priority queue does not end.
7. Calculate virtual machine wait time for assigning the task load.
8. Calculate virtual machine makespan and compare with a finish time from the predictions-based priority queue.
9. Arrange prediction-based priority queue in decreasing manner based on task order using prediction equations 1 to 11.
10. Prepare a prediction-based priority queue based on equations 1 to 11 and then map workload with the minimum execution time processor.

TABLE 3. Cloud environment specification.

Sr. No.	Entities / Component	Ranges / Specification
1	Host Machines	100
2	Virtual Machines (VM)	40
3	Number of User	01
4	Virtual Machine Manager (VMM)	Xen
5	Bandwidth	30 Mbps

several tasks, which represents in the experimental results of Figure 4. It is clearly evident from the makespan graph that the PMHEFT algorithm is more efficient than the other four algorithms. According to workflow, our proposed algorithm (PMHEFT) outperforms HEFT, DHEFT, CHEFT, and MHEFT algorithms in terms of the average makespan of the submitted tasks by 95.97%, 73.65%, 50.49%, and 50.02%, respectively, as shown in Figure 4.

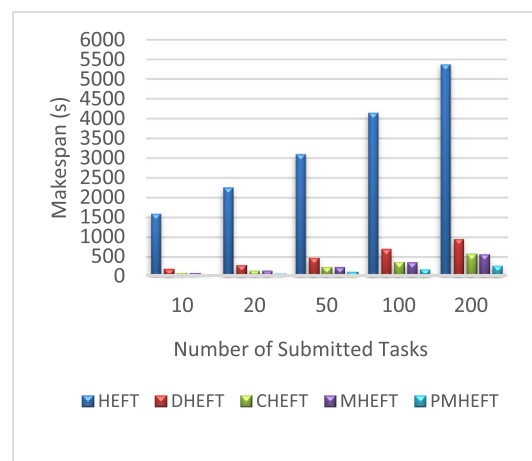## B. USAGE COMPARISON CHART FOR FIVE DIFFERENT ALGORITHMS

The efficiency results show the comparison of Predictive Priority-based Modified HEFT, HEFT, Modified HEFT, Cluster-based HEFT, and Dynamic HEFT algorithms are shown in Figure 5. It is clearly evident from the usage graph

TABLE 4. Virtual machine specification.

Sr. No.	Entities / Component	Ranges / Specification
1	RAM	1024 / 2048 MB
2	Bandwidth	12 / 30 Mbps
3	Number of CPU	01 / 02
4	Operating System	Linux / Windows / Mac
5	Storage	10 / 20 GB

TABLE 5. Host specification.

Sr. No.	Entities / Component	Ranges / Specification
1	RAM	4096 / 8192 MB
2	Bandwidth	12 / 30 Mbps
3	Number of CPU	02 / 04
4	Processor Speed	1.80 GHz & 2.00 GHz
5	Operating System	Linux / Windows / Mac
6	Storage	20 / 40 GB

FIGURE 4. Makespan v/s performance of different algorithms for submitted task.

that the PMHEFT algorithm is more efficient than the other four algorithms. According to workflow, our proposed algorithm (PMHEFT) outperforms HEFT, DHEFT, CHEFT, and

TABLE 6. Comparison between existing and proposed algorithm.

Algorithm Details	Prediction Nature	Support Static Load Balancing	Support Dynamic Load Balancing	Use Emergency Factor	Performance Speed (Based on Results)	Total Load Consumption	Mortality
HEFT	NO	YES	NO	NO	Slower	4.022 Mw	May be Die
DHEFT	NO	NO	YES	NO	Slower	0.567 Mw	May be Die
CHEFT	NO	YES	NO	NO	Slower	0.326 Mw	May be Die
MHEFT	NO	NO	YES	NO	Slower	0.323 Mw	May be Die
Proposed Algorithm (PMHEFT)	YES	YES	YES	YES	Faster	0.320 Mw	Never Die

MHEFT algorithms in terms of the average usage for the submitted tasks by 21.30%, 32.70%, 49.73%, and 51.41%, respectively, as shown in Figure 5.
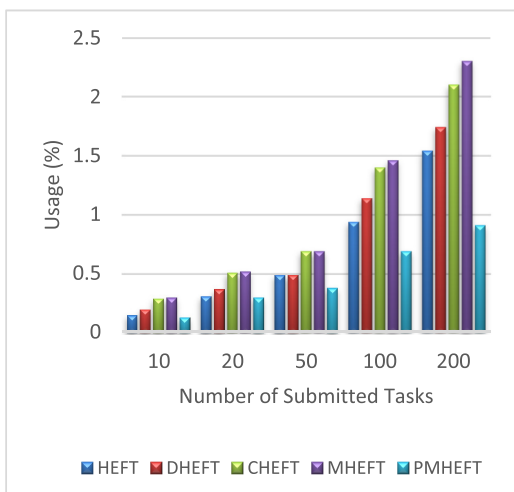

FIGURE 5. Usage v/s performance of different algorithms for submitted task.

C. LOAD CONSUMPTION COMPARISON CHART FOR FIVE DIFFERENT ALGORITHMS

The power consumption results show the comparison of Predictive Priority-based Modified HEFT, HEFT, Modified HEFT, Cluster-based HEFT, and Dynamic HEFT algorithms are shown in Figure 6. It can be seen that the total power consumption of workflow execution increases as per submitted several tasks, which represents in the experimental results of Figure 6. It is clearly evident from the power consumption graph that the PMHEFT algorithm is more efficient than the
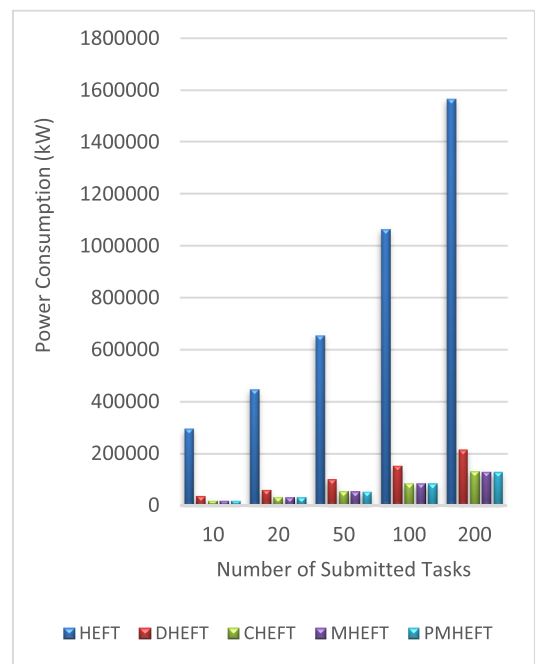

FIGURE 6. Power consumption v/s performance of different algorithms for submitted task.

other four algorithms. According to workflow, our proposed algorithm (PMHEFT) outperforms HEFT, DHEFT, CHEFT, and MHEFT algorithms in terms of the average power consumption of the submitted tasks by 92.34%, 45.19%, 01.56%, and 0.78%, respectively, as shown in Figure 6.

This prediction scheme result establishes that the scheme has a better edge over the non-predictive algorithms. The high CPU utilization demands applications can be implemented

using a prediction-based dynamic scheme and where the application load's stability is needed. Predicted demands can be easily fulfilled by VMs allocation/deallocation. In this paper, the results show that the studied scheduling algorithms, PMHEFT unveils the best performance with the lowest quadratic time complexity for the dynamic scheduling of DAGs in heterogeneous platforms. Simulation results outperform the benchmark methods while meeting the QoS goal using workload traces on PlanetLab servers demonstrate that the introduced method. However, this method focused on improving cloud application's performance by reducing the number of overloaded hosts.

VI. CONCLUSION

This paper proposed a novel approach for prediction priority scheduling-based schemes. This paper is explored as a new approach that is an efficient emergency priority aware algorithm. In this scheme, we consider the emergency cloud request, and priority is given to load that emergency cloud request for execution. This will ensure the load request availability and longevity of more sophisticated requests.

This research proposed a new method for task scheduling and loaded balancing to improve user response time and incoming tasks. To our knowledge, PMHEFT is the algorithm to outperform HEFT, DHEFT, CHEFT, and MHEFT with maintaining the time complexity of $O(v^2.p)$, where v is the number of tasks and p is the number of processors. Task scheduling clusters are formed using the calculation of top and bottom levels of tasks' predictive priority values. MHEFT algorithm is used for value calculation in which the average time for each task is taken into account. The prediction-based highest priority task for future procedures is selected and delegated first. Task clusters and ratings increase the use of resources in our work in comparison to existing methods. The highest weight VM is selected for assigning task load to reduce the user task's response time. In our proposed work, response time, makespan, use of resources, and service reliability are considered. The developed PMHEFT model works based on data harmonization across cloud services, resource management during cloud formation, user data integration with the cloud, provisioning of resources and services with improved performance. We proposed a predictive model to predict the advanced resource demands from the observed/historical database for efficient resource provisioning. This proposed PMHEFT model gives efficient resource provisioning for end users according to their needs based on an accurate workload prediction strategy. The calculations observation for the proposed prediction approach gives more accurate results as compared to the conversational approach.

Future work concerns particular mechanism deeper analysis and tries different models for new proposals by incorporating the Internet of Things with cloud computing emerging concepts. To develop a more efficient resource provisioning system for end-users, other proactive resource provisioning approaches might be addressed in future studies.

REFERENCES

[1] A. K. Kulkarni and B. Annappa, "Context aware VM placement optimization technique for heterogeneous IaaS cloud," *IEEE Access*, vol. 7, pp. 89702–89713, 2019.
[2] M. Liaqat, A. Naveed, R. L. Ali, J. Shuja, and K.-M. Ko, "Characterizing dynamic load balancing in cloud environments using virtual machine deployment models," *IEEE Access*, vol. 7, pp. 145767–145776, 2019.
[3] H. B. Alla, S. B. Alla, A. Touhafi, and A. Ezzati, "A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment," *Cluster Comput.*, vol. 21, no. 4, pp. 1797–1820, Dec. 2018.
[4] W. Lin, G. Peng, X. Bian, S. Xu, V. Chang, and Y. Li, "Scheduling algorithms for heterogeneous cloud environment: Main resource load balancing algorithm and time balancing algorithm," *J. Grid Comput.*, vol. 17, no. 4, pp. 699–726, Dec. 2019.
[5] J. Li, Y. Zhong, and X. Zhang, "A scheduling method of moldable parallel tasks considering speedup and system load on the cloud," *IEEE Access*, vol. 7, pp. 86145–86156, 2019.
[6] X. Ye, Y. Yin, and L. Lan, "Energy-efficient many-objective virtual machine placement optimization in a cloud computing environment," *IEEE Access*, vol. 5, pp. 16006–16020, 2017.
[7] S. D. Girase, M. Sohani, and S. Patil, "Dynamic resource provisioning in cloud computing environment using priority based virtual machine's," in *Proc. IEEE Int. Conf. Adv. Commun., Control Comput. Technol.*, Ramanathapuram, India, May 2014, pp. 1777–1782.
[8] S. Singhal and J. Patel, "Load balancing scheduling algorithm for concurrent workflow," *Comput. Informat.*, vol. 37, no. 2, pp. 311–326, 2018.
[9] Y. Samadi, M. Zbakh, and C. Tadonki, "E-HEFT: Enhancement heterogeneous earliest finish time algorithm for task scheduling based on load balancing in cloud computing," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Orleans, France, Jul. 2018, pp. 601–609.
[10] A. Kaur and B. Kaur, "Load balancing optimization based on hybrid heuristic-metaheuristic techniques in cloud environment," *J. King Saud Univ.-Comput. Inf. Sci.*, Mar. 2019, doi: 10.1016/j.jksuci.2019.02.010.
[11] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 682–694, Mar. 2014.
[12] L. Wu, R. Ding, Z. Jia, and X. Li, "Cost-effective resource provisioning for real-time workflow in cloud," *Complexity*, vol. 2020, Mar. 2020, Art. no. 1467274.
[13] P. Kumar and R. Kumar, "Issues and challenges of load balancing techniques in cloud computing: A survey," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–35, Feb. 2019.
[14] M. Sardaraz and M. Tahir, "A hybrid algorithm for scheduling scientific workflows in cloud computing," *IEEE Access*, vol. 7, pp. 186137–186146, 2019.
[15] K. Dubey, M. Kumar, and S. C. Sharma, "Modified HEFT algorithm for task scheduling in cloud environment," *Procedia Comput. Sci.*, vol. 125, pp. 725–732, Jan. 2018.
[16] A. S. Milani and N. J. Navimipour, "Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends," *J. Netw. Comput. Appl.*, vol. 71, pp. 86–98, Aug. 2016.
[17] A. S. Khalifa, T. A. Fergany, R. A. Ammar, and M. F. Tolba, "Dynamic on-line allocation of independent task onto heterogeneous computing systems to maximize load balancing," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol.*, Sarajevo, Bosnia and Herzegovina, Dec. 2008, pp. 418–425.
[18] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *J. Parallel Distrib. Comput.*, vol. 61, no. 6, pp. 810–837, Jun. 2001.
[19] A. N. Tantawi and D. Towsley, "Optimal static load balancing in distributed computer systems," *J. ACM*, vol. 32, no. 2, pp. 445–465, Apr. 1985.
[20] S. Azizi, M. Zandsalimi, and D. Li, "An energy-efficient algorithm for virtual machine placement optimization in cloud data centers," *Cluster Comput.*, vol. 23, no. 4, pp. 3421–3434, Dec. 2020.
[21] V. Kherbache, E. Madelaine, and F. Hermenier, "Scheduling live migration of virtual machines," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 282–296, Jan. 2020.
[22] A. M. S. Kumar and M. Venkatesan, "Task scheduling in a cloud computing environment using HGPSO algorithm," *Cluster Comput.*, vol. 22, no. S1, pp. 2179–2185, Jan. 2019.

[23] S. K. Panda and P. K. Jana, "Efficient task scheduling algorithms for heterogeneous multi-cloud environment," *J. Supercomput.*, vol. 71, no. 4, pp. 1505–1533, Jan. 2015.

[24] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.

[25] A. Mazrekaj, A. Sheholli, D. Minarolli, and B. Freisleben, "The experiential heterogeneous earliest finish time algorithm for task scheduling in clouds," in *Proc. 9th Int. Conf. Cloud Comput. Services Sci.*, vol. 1, 2019, pp. 371–379.

[26] A. Al-Rahayfeh, S. Atiewi, A. Abuhussein, and M. Almiani, "Novel approach to task scheduling and load balancing using the dominant sequence clustering and mean shift clustering algorithms," *Future Internet*, vol. 11, no. 5, p. 109, May 2019.

[27] M. Sohani and S. C. Jain, "State-of-the-art survey on cloud computing resource scheduling approaches," in *Ambient Communications and Computer Systems* (Advances in Intelligent Systems and Computing), vol. 696. Singapore: Springer, 2018.

[28] G. Wang, Y. Wang, H. Liu, and H. Guo, "HSIP: A novel task scheduling algorithm for heterogeneous computing," *Sci. Program.*, vol. 2016, p. 11, Mar. 2016.

[29] C.-H. Hsu, C. W. Hsieh, and C.-T. Yang, "A generalized critical task anticipation technique for DAG scheduling," in *Algorithms and Architectures for Parallel Processing*, vol. 4494. Berlin, Germany: Springer, 2007, pp. 493–505.

[30] H. Chen, X. Zhu, G. Liu, and W. Pedrycz, "Uncertainty-aware online scheduling for real-time workflows in cloud service environment," *IEEE Trans. Services Comput.*, early access, Aug. 21, 2018, doi: 10.1109/TSC.2018.2866421.

[31] L. Zhang, L. Zhou, and A. Salah, "Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments," *Inf. Sci.*, vol. 531, pp. 31–46, Aug. 2020.

[32] M. Dong, L. Fan, and C. Jing, "ECOS: An efficient task-clustering based cost-effective aware scheduling algorithm for scientific workflows execution on heterogeneous cloud systems," *J. Syst. Softw.*, vol. 158, Dec. 2019, Art. no. 110405.

[33] V. Priya, C. S. Kumar, and R. Kannan, "Resource scheduling algorithm with load balancing for cloud service provisioning," *Appl. Soft Comput.*, vol. 76, pp. 416–424, Mar. 2019.

[34] Y. Fan, L. Tao, and J. Chen, "Associated task scheduling based on dynamic finish time prediction for cloud computing," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Dallas, TX, USA, Jul. 2019, pp. 2005–2014.

[35] G. Jia, G. Han, J. Jiang, N. Sun, and K. Wang, "Dynamic resource partitioning for heterogeneous multi-core-based cloud computing in smart cities," *IEEE Access*, vol. 4, pp. 108–118, 2016.

[36] M. Guo, Q. Guan, and W. Ke, "Optimal scheduling of VMs in queueing cloud computing systems with a heterogeneous workload," *IEEE Access*, vol. 6, pp. 15178–15191, 2018.

[37] K.-M. Cho, P.-W. Tsai, C.-W. Tsai, and C.-S. Yang, "A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing," *Neural Comput. Appl.*, vol. 26, no. 6, pp. 1297–1309, Aug. 2015.

[38] M. Divyaprabha, V. Priyadharshni, and V. Kalpana, "Modified heft algorithm for workflow scheduling in cloud computing environment," in *Proc. 2nd Int. Conf. Inventive Commun. Comput. Technol. (ICICCT)*, Coimbatore, India, Apr. 2018, pp. 812–815.

[39] M.-L. Chiang, H.-C. Hsieh, W.-C. Tsai, and M.-C. Ke, "An improved task scheduling and load balancing algorithm under the heterogeneous cloud computing network," in *Proc. IEEE 8th Int. Conf. Awareness Sci. Technol. (iCAST)*, Taichung, Taiwan, Nov. 2017, pp. 290–295.

[40] L. Kong, J. P. B. Mapetu, and Z. Chen, "Heuristic load balancing based zero imbalance mechanism in cloud computing," *J. Grid Comput.*, vol. 18, no. 1, pp. 123–148, Mar. 2020.

[41] S. Seth and N. Singh, "Dynamic heterogeneous shortest job first (DHSJF): A task scheduling approach for heterogeneous cloud computing systems," *Int. J. Inf. Technol.*, vol. 11, no. 4, pp. 653–657, Dec. 2019.

[42] A. Hussain, M. Aleem, A. Khan, M. A. Iqbal, and M. A. Islam, "RALBA: A computation-aware load balancing scheduler for cloud computing," *Cluster Comput.*, vol. 21, no. 3, pp. 1667–1680, Sep. 2018.

[43] A. Hussain, M. Aleem, M. A. Iqbal, and M. A. Islam, "SLA-RALBA: Cost-efficient and resource-aware load balancing algorithm for cloud computing," *J. Supercomput.*, vol. 75, no. 10, pp. 6777–6803, Oct. 2019.

[44] D. C. Devi and V. R. Uthariaraj, "Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks," *Sci. World J.*, vol. 2016, p. 14, Feb. 2016.

[45] M. S. Kumar, I. Gupta, and P. K. Jana, "Forward load aware scheduling for data-intensive workflow applications in cloud system," in *Proc. Int. Conf. Inf. Technol. (ICIT)*, Bhubaneswar, India, Dec. 2016, pp. 93–98.

[46] S. Kapoor and C. Dabas, "Cluster based load balancing in cloud computing," in *Proc. 8th Int. Conf. Contemp. Comput. (IC)*, Noida, India, Aug. 2015, pp. 76–81.

[47] H. R. Faragardi, M. R. Saleh Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: A budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1239–1254, Jun. 2020.

[48] A. Pellegrini, P. Di Sanzo, and D. R. Avresky, "Proactive cloud management for highly heterogeneous multi-cloud infrastructures," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, Chicago, IL, USA, May 2016, pp. 1311–1318.

[49] A. Gupta, O. Sarood, L. V. Kale, and D. Milojicic, "Improving HPC application performance in cloud through dynamic load balancing," in *Proc. 13th IEEE/ACM Int. Symp. Cluster, Cloud, Grid Comput.*, Delft, The Netherlands, May 2013, pp. 402–409.

[50] A. Gandhi, X. Zhang, and N. Mittal, "HALO: Heterogeneity-aware load balancing," in *Proc. IEEE 23rd Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst.*, Atlanta, GA, USA, Oct. 2015, pp. 242–251.

[51] M. A. Alworafi and S. Mallappa, "An enhanced task scheduling in cloud computing based on deadline-aware model," *Int. J. Grid High Perform. Comput.*, vol. 10, no. 1, pp. 31–53, Jan. 2018.

[52] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. IEEE 8th Int. Conf. E-Sci.*, Chicago, IL, USA, Oct. 2012, pp. 1–8.

MAYANK SOHANI (Member, IEEE) received the M.Tech. degree in computer science from the School of Computer Science, Devi Ahilya Vishwavidyalaya, Indore, in 2011. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Rajasthan Technical University, Kota, India.

S. C. JAIN received the master's degree in computer science and technology from IIT Roorkee and the Ph.D. degree in VLSI design from IIT Delhi. He served with Defence Research and Development (DRDO), Bengaluru, India. He is currently working as a Professor with the Department of Computer Science and Engineering, Rajasthan Technical University, Kota, India. His research work focuses particularly on high-performance computing systems, VLSI design, Real-Time Embedded Systems, and reversible computing.

• • •