# THRESHOLD BASED VM PLACEMENT TECHNIQUE FOR LOAD BALANCED RESOURCE PROVISIONING USING PRIORITY SCHEME IN CLOUD COMPUTING

Mayank Sohani and Dr. S. C. Jain

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

***ABSTRACT***

*The unbalancing load issue is a multi-variation, multi-imperative issue that corrupts the execution and productivity of processing assets. Workload adjusting methods give solutions of load unbalancing circumstances for two bothersome aspects over-burdening and under-stacking. Cloud computing utilizes planning and workload balancing for a virtualized environment, resource partaking in cloud foundation. These two factors must be handled in an improved way in cloud computing to accomplish ideal resource sharing. Henceforth, there requires productive resource, asset reservation for guaranteeing load advancement in the cloud. This work aims to present an incorporated resource, asset reservation, and workload adjusting calculation for effective cloud provisioning. The strategy develops a Priority-based Resource Scheduling Model to acquire the resource, asset reservation with threshold-based load balancing for improving the proficiency in cloud framework. Extending utilization of Virtual Machines through the suitable and sensible outstanding task at hand modifying is then practiced by intensely picking a job from submitting jobs using Priority-based Resource Scheduling Model to acquire resource asset reservation. Experimental evaluations represent, the proposed scheme gives better results by reducing execution time, with minimum resource cost and improved resource utilization in dynamic resource provisioning conditions.*

***KEYWORDS***

*Cloud Computing, Load Balancing, Quality of Service, Scheduling, Virtual Machine*

## 1. INTRODUCTION

Cloud Computing is a model which provides convenient access to on-demand network access, computing power, and storage from a shared pool of configurable resources [1]. Cloud models create the illusion of "infinite" resources available on demand for computing purposes [2]. Cloud computing is getting progressively mainstream due to its capacity to give boundless registering administrations with the pay-as-you-go model. Cloud frameworks utilize virtualization innovation to provide physical machines (PMs) assets in the form of virtual machines (VMs). Cloud providers create VMs and deploy them to the cloud on end-user requests. Each VM runs its working framework and devours assets (e.g., CPU, memory, and transmission capacity) from its host PM [3]. The cloud computing provider and consumers signed and agreed in the pre-established contract known as Service Level Agreement (SLAs) to meet Quality of Service (QoS) requirements. The resource distribution process takes place based on predefine SLAs, which define Service Level Objectives (SLOs), for every SLA term, like response time, availability of services, storage, bandwidth, etc. [4].

In any case, if the defined SLAs are violated, cloud providers need to pay SLAs defined penalties to customers in a stipulated period mentioned in the contract. On the other perspective, the cloud provider needs to manage the issues of allotting resources satisfactory to every software adequately by ensuring effective utilization of used resources. Otherwise, we have to face performance degradation due to incompetent and deficient usage and allotment of resources. To handle such resource uses and allotment issues, we need an appropriate resource scheduling policy. The resource scheduling procedure in the cloud can be built through three phases:

- Asset disclosure and refinement: The assets present in the system framework are found, and data identified with them is gathered.
- Asset choice: The target asset is chosen depending on the specific boundaries of the task and asset.
- Task admission: The task to be executed is submitted to the asset selected.

Resource provisioning in cloud computing is distributing the task load on different virtual machines and getting maximum utilization of the same while minimizing the total execution time.

Scheduling can be categorized into two main categories:

- Static Scheduling Algorithm
- Dynamic Scheduling Algorithm

In static scheduling, all the resource allotment and task execution considerations are done in advance. In contrast, in dynamic scheduling, all the resource allotment and task execution considerations are taken care of at the run time. Many overheads need to be handled at runtime. These overheads arise due to a lack of resources and improper allocation of resources to specific task execution.

Cloud users, cloud brokers, physical machines, virtual machines, and cloud service providers are the main entities in the cloud. Cloud users can submit their service requests from anywhere and execute their requests by available resources. These resources are virtual machines created on top of physical machines through virtualization technology and kept in a cloud datacentre. Cloud broker is the entity that works in between the cloud datacentre and cloud users to allocate cloud resources to the client’s workflow applications. Many resource scheduling algorithms exist in cloud computing and provide benefits to cloud users and cloud service providers to effectively utilize cloud resources. The algorithms designed in the following manner for resource scheduling in cloud computing-

- The algorithm is structured in such a way that it fulfills the Quality of Service (QoS) requirements forced by cloud clients.
- The algorithm is structured in such a way that it is intended to perform load adjusting among virtual machines, which results in the progress of asset usage at the service provider’s end [5].

When cloud consideration takes place, usually load balancing technique highly requires to distribute the task load among different VMs. Without it, the unequal burdens in server formation may cause asset wastage, execution corruption, an SLA violation. Accordingly, using the correct load balancing strategy can also improve servers' usage and improve Quality of Service (QoS). A portion of the specialists in this field centers around the virtual machine distribution or virtual machine movement to accomplish load adjusting [6]. When the server consolidation process takes place, it must be very effective for energy consumption and operating cost estimation. It

may also cause performance degradation of servers if it is not correctly defined. From another perspective, server consolidation to solidify virtual machines on specific servers, making it more probable for machine over-burden to happen. Then again, the shutdown time and correspondence cost brought about by server consolidation is unavoidable and should be taken care of appropriately to fulfil the conveyed QoS [7]. Other than VM placement, there is other research focusing on job allocation for load balancing. The author uses the matrix for handling the problem of assigning tasks to virtual machines. Such consideration for task allocation can invite an unbalanced task in a cloud environment [8].

For handling the load balancing, the task allocation can be applied by well-recognized heuristics, such as First Come First Serve, Random Selection, and Round Robin techniques. This paper proposes a load balancing algorithm of priority-based resource allocation using a threshold-based VM load balancing technique. The tasks are first prioritized based on required resources, i.e., processors, number of users, job completion time, job category, user category, and used software cost [9]. Then they are assigned to various available hosts based on priority. This proposed algorithm incorporates the benefits of Max-Min and Min-Min algorithms [10]. Here we proposed a heuristic algorithm for resource allocation based on task load balancing through different available resources to get a better make-span. The algorithm is split into two phases: resource allocation from existing resources based on task and virtual machine mapping and resource allocation from newly created resources when existing resources cannot handle the current task load. The proposed algorithm heuristic works based on VMs threshold for utilization to maintain SLA response time violations. A proper VM utilization threshold is chosen for realistic workload situations in the cloud environment.

The main contribution to this study includes:

- Performance evaluation of existing algorithms for static and dynamic scheduling criteria in cloud computing.
- Propose and implement a Threshold-based VM Placement Technique for Load Balanced Resource Provisioning using the Priority Scheme in cloud computing.
- Proposing and implementing a new load balancing scheme for cloud computing.

We propose a model, Threshold Based VM Placement Technique for Load Balanced Resource Provisioning Using Priority Scheme in Cloud Computing, in this paper, which would be based on the priority queue based execution for all the submitted requests. In load balancing network systems (cloud computing), this Threshold-based VM placement technique using a Priority scheme for load balancing in the cloud computing approach will be easier and more effective. Our proposed method is working for both dynamic and static load balancing. 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 the architecture of our proposed algorithm heuristics along with the proposed methodology. The simulation setup, results, and detailed analysis are given in section 4. Section 5 concludes the paper and specifies future work directions.

## 2. RELATED WORK

In this section, the background of the research work is analyzed. This strategical overview at the beginning of the load balancing begins with the job distribution and further job schedule planning done under the cloud job processing. In the initial phases of the research, the job distribution schedule and job concurrent execution control identification are the major challenges [11]. The different researchers searched for a framework where the submitted job is executed on suitable resource combinations, and this combination is decided by characterizing the job load. Due to

this characterization, the cloud has received attention in recent years. Resource capacity up-gradation and scheduler design require a deep understanding of the job load properties, i.e., arrival rate, job duration, and required resources. Hence, further research continues to build such scheduling heuristics and framework to fulfill the required goal of cloud computing environments, resource provisioning in dynamic scenarios [12]. In the cloud, researchers propose predictive resource management frameworks in which user workload patterns are monitored, analyzed, and based on that, resource provisioning and de-provisioning take place [13]. The task scheduling is performed based on task patterns in a workload pool. This task pattern then splits the submitted task into different batches in a dynamic way [14].

VM migration is another option for load balancing in cloud environments, and while doing so, the VM is still accessible by the end-user. This live VM migration takes place within due time by which it will not affect the SLAs. To maintain SLA violation, task prioritization and deadline-based migration policies are implemented [15]. Evictions of tasks will do workload management during bursty workloads with different resource demands of tasks. To achieve this, eviction policies are designed so that the most recently started task is evicted first and workload awareness slot-based priority scheduling implemented so that system load and resource time can be kept minimum [16]. Task scheduling is performed through deadline guaranteed services. Each user needs to submit a percentage of their requirements, and they wish to serve within a specified deadline [17]. Some researchers proposed fault tolerance-based proactive and reactive scheduling in the cloud through which the resource scheduler selects the resources based on their location, availability, and reliability. It mainly focuses on the mean type between availability and allocating such resources for the task execution purpose and improving the QoS [18,19]. The workflow application's execution in the cloud becomes more and more attractive because on-demand computing resources can be workflow applications enabled. The workflow applications on cloud services work for user-defined loose and tight deadline constraints and minimize the execution cost [20].

Cloud computing provides an application provisioning environment offered through Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) to cloud users. These cloud users can run workload-based workload characteristics through flexible resource provisioning strategies for achieving QoS and cost effective resource configuration. The cost-saving and QoS requirements are fulfilled through a dynamic instance provisioning strategy by altering the active instances during runtime [21]. In cloud workflow, scheduling requires desired QoS achievement under minimum execution cost. This technique recursively plans the scheduling through the partial critical paths ending on previously scheduled tasks in SaaS clouds [22]. Task scheduling in cloud computing is crucial, and cloud providers have to manage resource allocation for user tasks from the available pool of configurable heterogeneous resources. If the best appropriate resources are allocated to a given task, then we can achieve better performance on some parameters. Some heuristic cloud resource allocation approaches entertain VMs requirements, cloudlet tasks, and real workloads by improved load balancing and task scheduling. Optimal completion time has been introduced to achieve load balancing and get max resource utilization in the cloud. The allocation method works for two task category requests of optimal competition time and earliest finish time. This approach performance achieved ideal load balancing, maximum resource utilization, and minimum makespan time factors [23].

The shared infrastructures handle tens of thousands of applications under a broad application of cloud computing environment. Thus, in such a diverse application environment, shared resource allocation to meet QoS requirements has become a real challenge due to the properties and application workload differ widely and may change over time. These issues are handled through Round Robin, static allocation schemes, and even load distribution, probabilistic allocation policy [24]. In dynamic tasks, batches-based workflow applications use DAG-based algorithms,

implementation for renting virtual machines, resources in cloud computing. These tasks under different task-batches merged into a unit of the task, and this unit aware deadline allocation policy executes workload and task deadlines to minimize rented interval utilization. This kind of rented VMs allocation saves rented costs in DAG-based platforms [25]. The hypervisor improves the efficiency of a single physical server by converting it into multiple virtual servers with the help of virtualization technology [26]. Service Level Agreement (SLAs) violations are very important issues and may happen while doing virtual machine instances, resource allocation from the overloaded host. To protect from SLA violations, we have to do the allocation of VMs from underloaded host machines, which turn in idle host conditions during under load conditions. We can save energy consumption to start new resources for the workload request allocation purpose [27, 28]. Some comparing factors of existing and proposed algorithm heuristics are considered for comparison and represented by Table 1, including the following parameters: Response Time, Execution Time, Resource Utilization, Load Balancing, QoS, Scalability, and Energy Consumption.

Table1. Comparing factors of the previous works vs. the proposed model

| Heuristics Name | Response Time | Execution Time | Resource Utilization | Load Balancing | Quality of Service | Scalability |
|---|---|---|---|---|---|---|
| EPRD [29] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ECOS [30] | ✓ | ✘ | ✘ | ✘ | ✘ | ✓ |
| F-MRSQN [31] | ✓ | ✘ | ✓ | ✘ | ✓ | ✓ |
| DPLS [32] | ✓ | ✓ | ✓ | ✘ | ✓ | ✘ |
| DRP [33] | ✓ | ✓ | ✓ | ✓ | ✘ | ✘ |
| ACOPS [34] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| MHEFT [35] | ✓ | ✓ | ✓ | ✘ | ✘ | ✘ |
| AMS [36] | ✓ | ✘ | ✘ | ✓ | ✗ | ✗ |
| HLBZID [37] | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| DHSJF [38] | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| MRLB[39] | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ |
| RALBA [40] | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ |
| IWRR [41] | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ |
| FLASDIWA [42] | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ |
| GRP-HEFT [43] | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ |
| ACM [44] | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ |
| PROPOSED ALGORITHM HEURISTICS | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

## 3. PROPOSED METHODOLOGY

This section gives details of the proposed VM allocation policy based on Priority and Preemptable request execution through online adaptive resource allocation. These two methodologies will minimize the response time through the effective utilization of cloud resources.

### 3.1. Proposed Virtual Machine Distribution Policy

In Cloud computing, dynamic resource allocation for web applications aims at maintaining the response time to achieve a predefined SLA. When peak workload is handled through virtual

machines by a virtualization administrator spread over cloud systems, virtual machine performance degrades due to fierce resource contention. The proposed policy will address how virtual machines will get a guaranteed configurable resource pool during the contention period.

The feature through which we can make resources available when it is required is the reservation of resources. This proposed policy ensures the proper virtual machine allocation based on the workload request to fulfill the required SLAs during resource contention. We categorized three virtual machine types for achieving the above solution and improved performance: Immediate, Best Efforts, and Advanced Reservation. The virtual machine grading is allocated based on the importance of VM performance. The policy represents the sharing of processing power for different virtual machines running on host machines, and it supports how the virtual machines categorization into three different categories. The priority field is given to each virtual machine sets and based on this, priority CPU share VMs. Immediate virtual machine priority works for smaller value is accepted. Here the priority is interpreted as one VM runs faster than others for how many times, i.e., Virtual machine Immediate and Best Effort category have priority 1 and 2 respectively. The two-priority virtual machine runs twice as fast as a one-priority virtual machine. The Virtual Machine distribution policy pseudo-code shows in Algorithm no. 2.

#### 3.1.1. Advance Reservation Virtual Machine:

This type of virtual machine performs the reservation of required resources. These reserved resources are available at a specified time. These VM's are responsible for performing the required task based on the highest priority value compared to other categories of virtual machines.

#### 3.1.2. Best Effort Virtual Machine:

This type of virtual machine performs resource allocation to the task request submitted in a waiting queue, and such kind of virtual machine priority values is less than AR VM's and more excellent than Immediate VM's.

#### 3.1.3. Immediate Virtual Machine:

This type of virtual machine allocates resources to requested tasks immediately based on the availability of resources. If resources are not available, then the task request is rejected without any advanced reservation of resources. This type of virtual machine priority value is smaller than all other VM's categories and due to the same VM's can run slowly.

### 3.2. Cloud Resource Load Balancing

Virtual machines are created through a hypervisor on physical resources, and once the virtual machines are configured, then based on workload requests submitted to the cloud interface, resource allocation performs. The load balancing policies are applied while virtual machine resources are allocated to request. The load balancer pseudo-code shows in Algorithm no 3.

### 3.3. Task Execution Based on Preemptable Policy

The cloud scheduler first accepts the user requests and partitions all user requests into the tasks for Direct Acyclic Graph (DAG) input purpose. Now for the cloud request list scheduling, static resource allocation is performed.

### 3.3.1. Min-Min Cloud Scheduling (MMCS):

The task dependency is not entertained in the greedy Min-Min Cloud Scheduling algorithm. So, the Min-Min algorithm improved form is used in every scheduling step for maintaining the mappable task set through dynamic allocation of task dependencies. MMCS pseudo-codes are shown in Algorithm no. 4.

### 3.3.2. Online Adaptive Scheduling:

Due to contention in cloud system resources, task actual execution finish time may not be the same as the estimated execution finish time. Hence, based on information on tasks available, the proposed online adaptive scheduling algorithm performs the dynamic resource allocation. This proposed technique continuously checks for remaining configurable static resources and re-evaluates their finish time for submitting tasks into the ready queue [23].

## 3.4. Proposed Mechanism:

The proposed scheduling heuristic allocates tasks on virtual machines as per agreed SLA objectives. This scheduling heuristic instantly creates required virtual machines on available physical resources if required to fulfill the current task request. This proposed scheduling heuristic builds with the help of three sub algorithms of Virtual Machine Load Balancing, Preemption Technique, and Monitoring. Through this proposed heuristic, application performance is optimized, QoS parameters performance improves, and SLA violation possibilities are reduced. Figure 1 represents the proposed system block diagram, and Figure 2 illustrates the proposed system flow diagram.

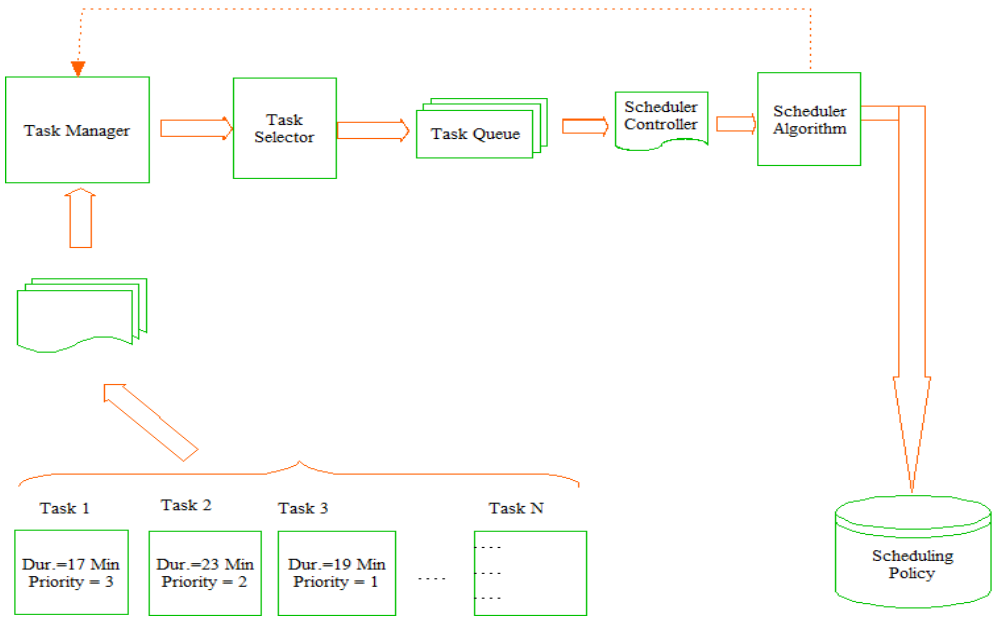


Figure 1. Proposed System Block Diagram

The cloud service provider gives different types of VMs configured based on pricing and QoS models to achieve the maximum resource utilization for various workloads. We have calculated two thresholds, namely upper threshold level (UTL) and lower threshold level (LTL), to scale up and scale down situations in a cloud environment. These scale-up and scale-down movements are generated continuously due to the dynamic change, resource demands during peak workload

hours. This undesirable situation in an application considers as hysteresis. The UTL and LTL values determine through resource utilization, and values can be considered by static threshold and proportional threshold. In static threshold UTL value is set as high as possible, then there are chances of high resource utilization, but it may overload the currently running application. So UTL values should be set high as much as possible by considering the current application workload under-provisioning. If LTL value is set high in the static threshold, it may increase resource utilization by a hysteresis chance. The value of LTL should be given as low as by equation 1 for controlling this hysteresis situation.

$$LTL < \frac{UTL}{2} \qquad (1)$$

This equation 1 ensures that any application never enters into a hysteresis situation. However, this solution may give minimum utilization of resources and cut down the no of VM migrations. A fixed value of UTL is considered in the proportional threshold. In contrast, the LTL value is set dynamically for each scale down and scales up the situation in the current application workload. The LTL value updates as per equation 2 by monitoring system where n consider as total resource quantity.

$$LTL < \left(\frac{n-1}{n}\right) UTL \qquad (2)$$

The proportional threshold gives better utilization of resources since the termination of resource is done earlier than the static threshold by maintaining the application's current workload within UTL without SLA violations. The VM utilization threshold function AvgUTL () compares previously calculated system utilization with current average system utilization. Algorithm no. 1 presents AvgUTL () function. It works based on the agent monitoring system that traces entire logs for average system utilization calculation of last and current usages to finalize any need to create new VMs to the existing VM pools. With this, we can ensure VMs load balancing along with avoiding response time SLA violation.

The following QoS metrics parameters are used:

- Execution Time: In cloud computing, users submit their workload from the user bases to data centers, and these data centers execute these workloads based on the available resource in the resource pool. The time duration from the beginning of the first task starts processing and ends with the last task finished the processing for the user's workload request is considered execution time.

$$ET = TE - TS \qquad (3)$$

$$Where\ ET\ is\ Exectuion\ Time, TE\ is\ Time\ of\ ending\ last\ task,\ and\ TS\ is\ Time\ of\ starting\ the\ first\ task$$

- Response Time: Average time elapsed from when a process is submitted until get the first response is obtained for execution.
- Turnaround Time: Average time elapsed from when a process is submitted to when it has been completed.
- Waiting Time: Average time a process spends in the run queue.
- Makespan: It is the total amount of time required to complete a group of tasks.

In our proposed scheduling heuristic threshold based VM allocation manage load balancing and proper utilization of associated resources. If current workload is greater than the threshold as per algorithm no. 1 value of respective VM then algorithm no. 5 gets activated and prepare a new virtual machine for execution purpose.

**Algorithm 1: Average System Utilization**

**Needs:** isScale = FALSE
**Guarantee:** VM management to maintain SLA

1. While TRUE do.
2. Readlogs.Agent().
3. CalculatePreviousAvgUTL.Agent(System)
4. CalculateCurrentAvgUTL.Agent(System)
5. If Previous AvgUTL of System ≥ 80 or Current AvgUTL of System ≥ 80 and isScale = FALSE, then isScale = TRUE
6. PrepareAddVM.Agent()
7. isScale=FALSE
8. end
9. end

### 3.4.1. Proposed Scheduling Heuristic:

**Step 1:** Customer/ User: The scheduler gets the input in terms of provisions application data and users service deployment request composed of service level agreement terms.

**Step 2:** Cloud Scheduler: The scheduler gets the requests from the customer/user then the scheduler checks the dependencies of requests then prepares priority based task lists.

**Step 3:** Virtual Machine's Manager: VMM shares processing power to the virtual machine's which are active on a host as per the first step of proposed virtual machine allocation policy. Each virtual machine shares CPU power based on the priority field [Algorithm no. 2].

**Algorithm 2: Virtual Machine Distribution Policy**

**Needs:** PTs- Processing Tasks Quantity, VMs- Virtual Machines Quantity.
**Guarantee:** Allocation of PTs

1. Prepare PTs list.
2. Collect every PTs share of MIPS on Host available machines.
3. Fetch priority of every virtual machine.
4. Do PTs distribution to the virtual machines on the basis of priority.
5. Release PTs allocated to a virtual machine

Next step, get the information of total running VMs on each cloudlet along with a number of configurable available resources in a resource pool. The VMs list is prepared as per SLA terms and these listed VMs are skilled to entertain the requested service.

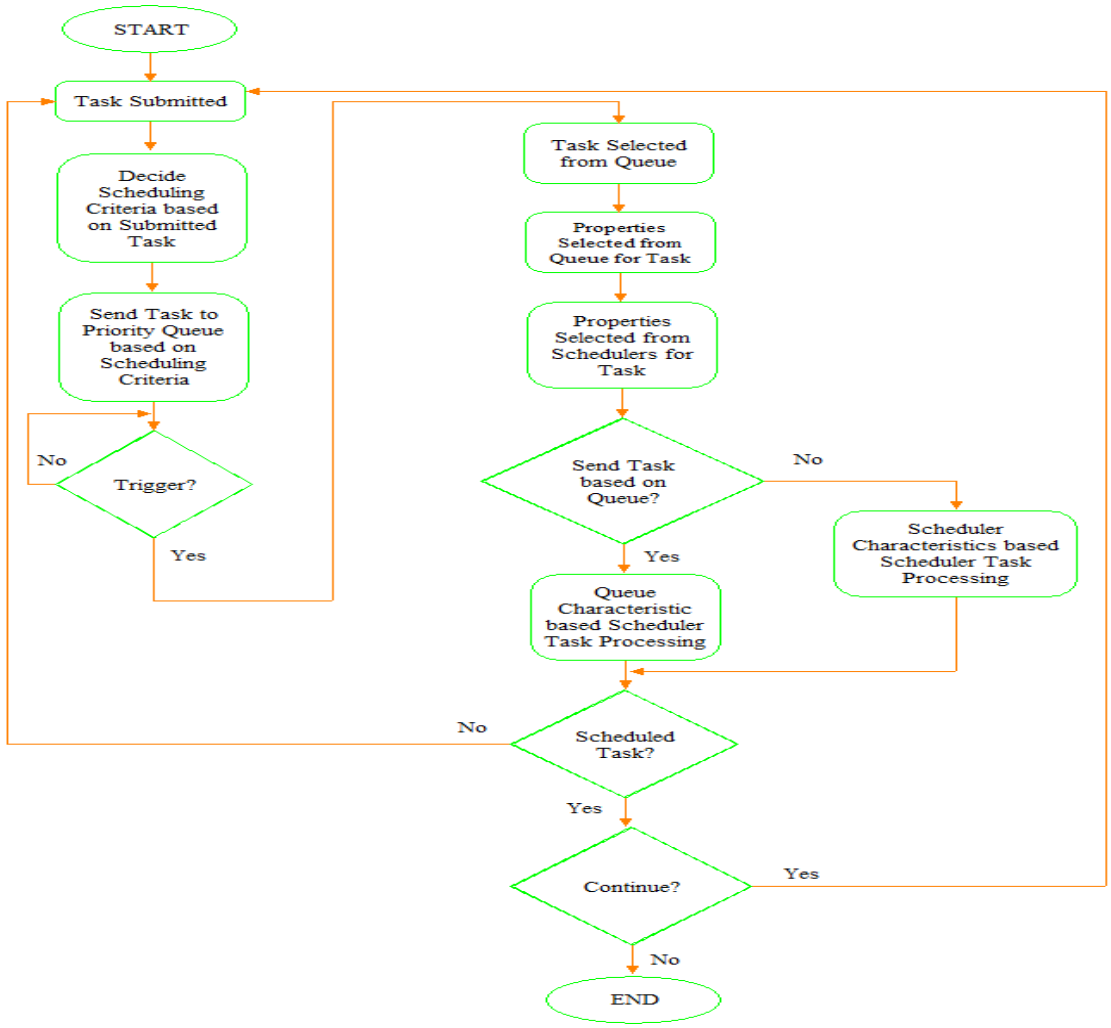


Figure 2. Proposed System Flow Diagram

**Step 4:** After preparing virtual machine bifurcation details as per SLA standards and QoS parameters, then it has been decided by the load balancer that from the prepared list, which virtual machine is allotted to service request to maintain the data centre load in each cloud [Algorithm no. 3].

**Algorithm 3: Virtual Machine's Load Balancing**

**Needs:** SU (UR, ARL) // SU- Suitable VM's, UR- User Request, ARL- Available Resources List

**Guarantee:** Virtual Machine's Load Balancing

1. Prepare each cloud available virtual machine list
2. For each VM available in the list prepare and maintain its state and index table

// State - BUSY/ AVAILABLE

3. Prepare all VM's list, which are busy in serving to a certain REQUEST
4. **IF** VM's AVAILABILITY State in List! = NULL
5. **FOR** Virtual Machine in (SU, UR, ARL) **do**
6. **IF** BUSY Virtual Machine List, VM hasn't founded **then**
7. Do allocation of VM to request
8. Change such VM state to BUSY VM and add it to the BUSY VM List
9. Break
10. **End if**

11. **End for**
12. **Else** add such requests from users into waiting queue
13. **End if**
14. Return allocated VM's Id

**Step 5:** In a data centre of any cloud if no virtual machine running with suitable resources found, the scheduler check for service request belongs to reservation in advance category, then search for any best effort other category request executing on different cloud environment or not. If such scenario is identifying the task with best matching effort execution has been preempted and advance reservation request gets executed [Algorithm no. 4].

**Algorithm 4: Cloud Scheduling using Min-Min Technique [1].**

**Needs:** A Request set, n clouds with different configuration
**Guarantee:** A execution plan defines by MMCS

1. Prepare T set of mappable task
2. **While** the assignment of task isn't done **do**
3. Modify T set of mappable task
4. **For** i: task Si ∈ T **do**
5. Find all other schedulers for task request of Si
6. Get the schedulers status about available time responses of earliest resources
7. Fetch the Cmin Cloud (Si) prepare the earliest completion time of Si considering it as non-preemptable
8. **End for**
9. Get the earliest finish time for the pair of Task & Cloud (Si, Cmin (Si)) prepare in for loop
10. Allocate tasks Si to Cloud Cmin (Si)
11. Take out Si from T
12. Regenerate the mappable T task set
13. **End While**

**Step 6:** If any cloudlet is not having Best-Effort category task for preempting then the scheduler find that a new virtual machine can start based on globally available resources which consisting of physical resources, if current VM threshold value meets based on algorithm no. 1 then, it automatically prepares and start's a new virtual machine using predefined configuration to execute service request [Algorithm no. 5].

**Algorithm 5: Preparing a New Virtual Machine**

1. **If** globally available resources can configure & host extra virtual machine.
2. **Then** prepare and start the new virtual machine instance.
3. Get include newly created virtual machine in VMs pool.
4. Deploy task request on newly created VM.

**Step 7:** If globally available resources are not competent to prepare and host another virtual machine, then such service request gets added into a queue by the resource scheduler until a required configurable resource virtual machine is available in the resource pool.

## 4. SIMULATION SETUP AND RESULTS

Cloud Analyst simulation, the experimental setup, performs the proposed scheduling heuristic. The proposed scheduling algorithm and existing scheduling algorithm get compared, which does not prioritize virtual machine allocation and preemption mechanism. Still, it uses round-robin-based load balancer—analysis and modeling of large-scale cloud computing environments

performed on CloudSim-based tool CloudAnalyst. The CloudAnalyst is built on top of the CloudSim tool kit by extending CloudSim functionality with the introduction of concepts that model Internet and Internet Application behaviours. Following are the domain entities and main components of the CloudAnalyst.

- **Region:** In CloudAnalyst entire world is divided in to 'Region'. Other entities User Bases and Data Centers belongs to any one of these regions.
- **User Base:** In Cloud Analyst User Base is responsible to generate traffic for the simulation. A User Base is considered a group of users that is modelled in the simulation as a single unit. The User Base defines detail such as number of users, user's geographic distribution, usage frequency, usage patterns, etc.
- **Data Centers:** In CloudAnalyst Data Centers is the most important entity. Data Center manages VM creation and destruction. It will perform routing of user request's received from User Base through internet to the VMs.

## 4.1. Experimental Setup

The proposed provisioning algorithms in cloud computing are simulated using Cloud Analyst tool for resource management. We prepare a first testbed set of 06 client's / user's requests for simulation purposes, and every set of client's / user's requests is made up of 600 sub-tasks approx. The parameters we fixed for simulation performance purposes are as below in Table 2. We prepare a second testbed set of 10 client's / user's requests for simulation purposes, and every set of client's / user's requests is made up of 600 sub-tasks approx. The parameters we fixed for simulation performance purposes are as below in Table 3. During every simulation run, cloud data centers increased from DC1 to DC2 for the first testbed environment and from DC1 to DC5 for the second testbed environment. The proposed algorithms are simulated for two testbed configuration environments under different workload categories of users' requests.

Table 2. Cloud Environment Specification for First Test Bed

| **Sr. No.** | **Entities / Component** | **Ranges / Specification** |
|---|---|---|
| 1 | Virtual Machine's | 10 |
| 2 | Number of User | 06 |
| 3 | Per User Request | 600 |
| 4 | Virtual Machine Manager (VMM) | Xen |
| 5 | Operating System | Linux |
| 6 | Number of Processors per Data Centre | 08 |
| 7 | Executable Instructions Length per Request (bytes) | 1000 |

Table 3. Cloud Environment Specification for Second Test Bed

| Sr. No. | Entities / Component | Ranges / Specification |
|---|---|---|
| 1 | Virtual Machine's | 25 |
| 2 | Number of User | 10 |
| 3 | Per User Request | 600 |
| 4 | Virtual Machine Manager (VMM) | Xen |
| 5 | Operating System | Linux |
| 6 | Number of Processors per Data Centre | 08 |
| 7 | Executable Instructions Length per Request (bytes) | 1000 |

## 4.2. Result Analysis and Discussion

The following figures represent the result computed through existing and proposed scheduling heuristic algorithms simulation. One by one, both the policies are executed using the above-shown configuration. Based on that all the result generated from First and Second Test Bed executing the user request with an average response time has been represented in Figure 3, Figure 4, Figure 5 and Figure 6 respectively. We have tested our results with three existing CloudAnalyst simulator algorithms, i.e., Round Robin, Equally Spread Current Execution Load, and Throttled. Figure 3 represents request execution average time for peak workload situations, and it is clearly shown through the result that our proposed scheduling heuristic algorithm gives improved request execution average time.

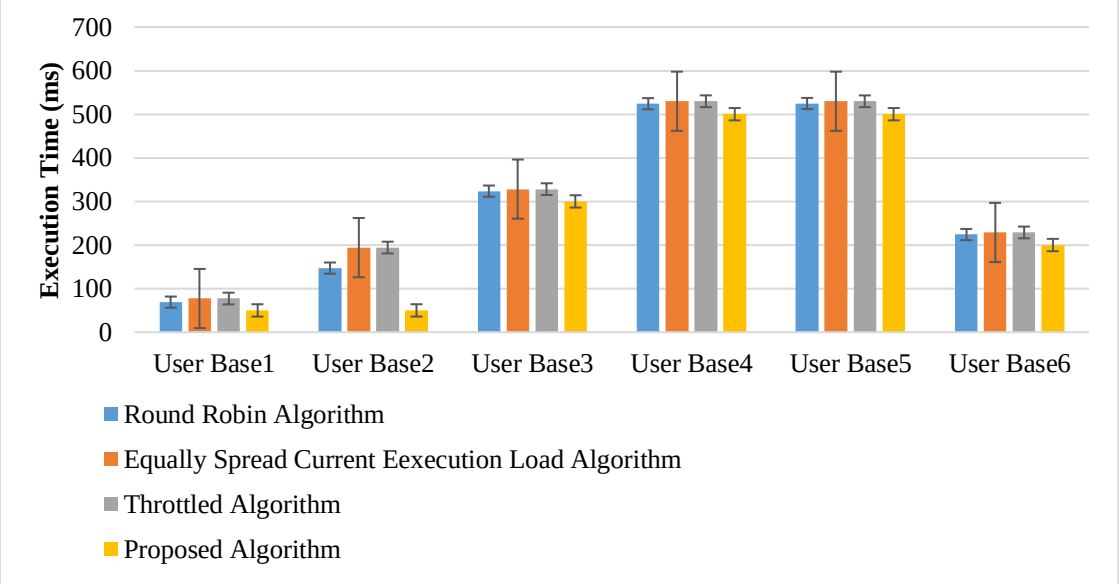


Figure 3. Request Average Execution Time for Different Algorithms

The Figure 4 represents request average response time at data centre for different workload situations and it is clearly shown through the result that our proposed scheduling heuristic algorithms gives improved request average execution time with minimum energy consumption.

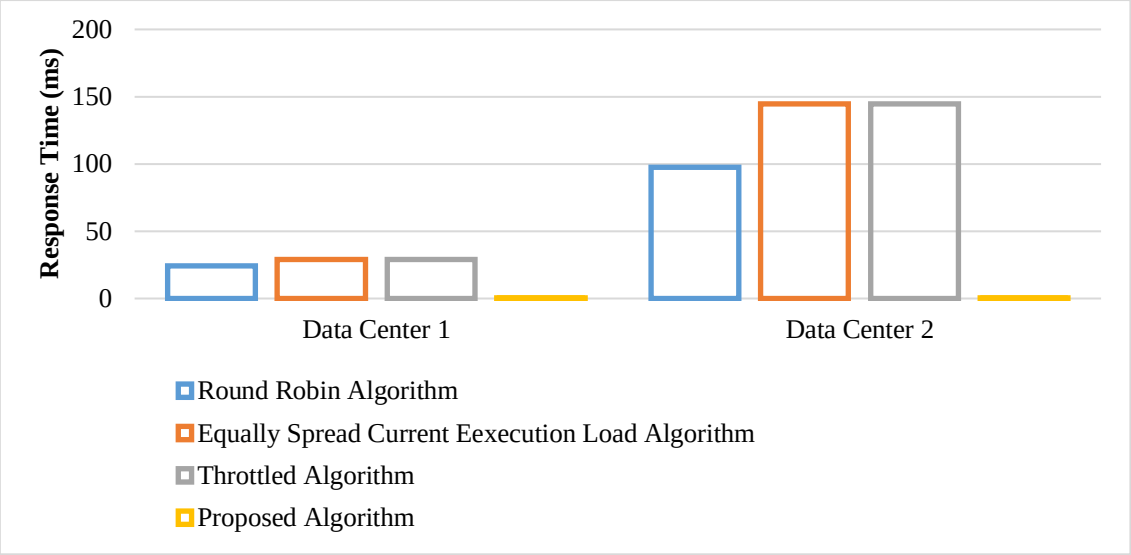


Figure 4. Request Average Response Time at Data Center for Different Algorithms

The Figures 3 & 4 gives improved results for request execution and response average time in heavy workload situations submitted by a number of users at the data center.

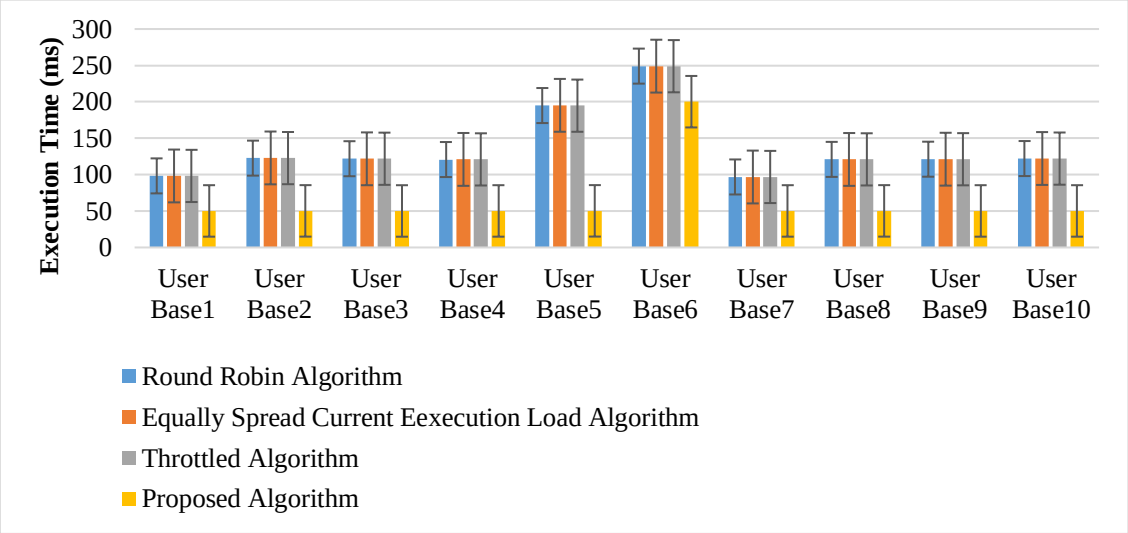


Figure 5. Request Average Execution Time for Different Algorithms

The Figure 6 represents request average response time at data centre for different workload situations and it is clearly shown through the result that our proposed scheduling heuristic algorithms gives improved request average execution time with minimum energy consumption.

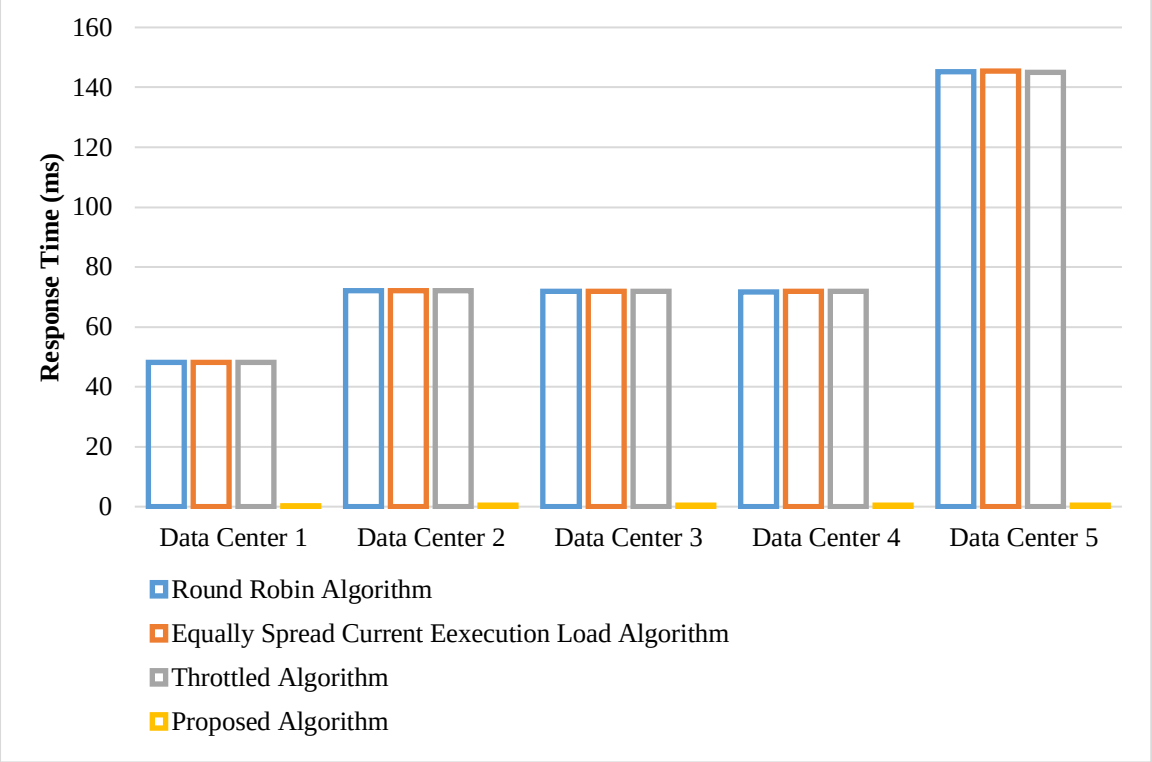


Figure 6. Request Average Response Time at Data Center for Different Algorithms

The Figures 5 & 6 gives improved results for request execution and response average time in heavy workload situations submitted by a number of users at the data center. Our proposed scheduling heuristic algorithms give better results as compared to existing available three scheduling algorithms, i.e. Round Robin, Equally Spread Current Execution Load and Throttled.

Figure 3 clearly shows that the proposed scheme performs better and needed less execution time by 11.70% from the Round Robin scheme, 15.22% from the Equally Spread Current Execution Load scheme, and 15.24% from the Throttled scheme for Testbed 1. Figure 4 clearly shows that compared to other existing schemes of CloudAnalyst simulator, proposed schemes are very responsive and took less response time to execute the users' submitted request's at two Data Center for Testbed 1. Figure 5 clearly shows that the proposed scheme performs better and needed less execution time by 52.37% from the Round Robin scheme, 52.39% from the Equally Spread Current Execution Load scheme, and 52.36% from the Throttled scheme for Testbed 2. Figure 6 clearly shows that compared to other existing schemes of CloudAnalyst simulator, proposed schemes are very responsive and took less response time to execute the users' submitted request's at five Data Center for Testbed 2. The result clarifies that our proposed scheme gives better results if workload size increases for execution in the cloud computing environment.

## 5. CONCLUSION

The paper presented the scheduling heuristic algorithms for the problem of resource contention in a cloud computing environment. The proposed scheduling heuristic algorithm system works for load balancing, allocation policy of virtual machines, and pre-emption techniques. The proposed scheduling heuristic of algorithms selects requests available in the waiting queue for execution pursuant to its capability. To improve the data centers utilization, virtualization technologies are commonly used. In this work, a priority-based execution scheme is proposed to schedule and

execute user's requests to improve utilization and responsiveness of computing capacity of data centres. This research proposed a new method for task scheduling and loaded balancing to improve execution time and response time for incoming user tasks. To our knowledge, Threshold based VM Placement Technique for Load Balanced Resource Provisioning using Priority Scheme in Cloud Computing is the algorithm to outperform Round Robin, Equally Spread Current Execution Load, and Throttled with maintaining the time complexity of O (v2 × p), where v is the number of tasks and p is the number of processors. Simulated scheduling heuristic algorithms for resource contention fierce in a cloud computing environment give better results for proposed scheduling heuristic algorithms than the existing scheduling heuristic algorithms.

Although the proposed scheme showed a considerable improvement in load balancing, response time, and execution time calculations, the critical factor that needs to address appropriately for efficiency is load balancing, i.e., the power consumption in data centres. The proposed technique gives enhanced results for QoS parameters such as execution time and response time. Other parameters, i.e., throughput, waiting time, turnaround time, and SLA violations, need to be addressed. Therefore, future work will extend the proposed scheduling heuristic algorithms scheme for VM live migration, waiting time, turnaround time, and energy efficiency factors in data centres for a real cloud computing environment through proposing a framework.

## CONFLICTS OF INTEREST

The author declares no conflict of interest.

## REFERENCES

[1] V. Priya, C. Sathiya Kumar, Ramani Kannan, "Resource scheduling algorithm with load balancing for cloud service provisioning", Applied Soft Computing, vol. 76, pp. 416-424, ,2019.
[2] Afzal, S., Kavitha, G. "Load balancing in cloud computing A hierarchical taxonomical classification", J Cloud Comp 8, vol. 22, 2019.
[3] H. Shen and L. Chen, "A Resource Usage Intensity Aware Load Balancing Method for Virtual Machine Migration in Cloud Datacenters," in IEEE Transactions on Cloud Computing, vol. 8, no. 1, pp. 17-31, 1 Jan.-March 2020.
[4] P. Haratian, F. Safi-Esfahani, L. Salimian and A. Nabiollahi, "An Adaptive and Fuzzy Resource Management Approach in Cloud Computing," in IEEE Transactions on Cloud Computing, vol. 7, no. 4, pp. 907-920, 1 Oct.-Dec. 2019.
[5] Renu Bala et al, "An Improved Heft Algorithm Using Multi-Criterian Resource Factors", in International Journal of Computer Science and Information Technologies, vol. 5, pp. 6958-6963, 2014,
[6] Lin, W., Peng, G., Bian, X. et al. "Scheduling Algorithms for Heterogeneous Cloud Environment: Main Resource Load Balancing Algorithm and Time Balancing Algorithm", JGridComputing, vol.17, pp.699–726,2019.
[7] Afzal and Kavitha, "Load balancing in cloud computing – A hierarchical taxonomical classification", Journal of Cloud Computing: Advances, Systems and Applications, issue 1,2019.
[8] Jun-jie Peng, Xiao-fei Zhi, Xiao-lan Xie, "Application type based resource allocation strategy in cloud environment", Microprocessors and Microsystems, vol. 47, part B, pp. 385-391, 2016.
[9] Wenqing Qi, "Optimization of cloud computing task execution time and user QoS utility by improved particle swarm optimization", Microprocessors and Microsystems, vol. 80, 2021.
[10] Zhao Tong, Xiaomei Deng, Hongjian Chen, Jing Mei, "DDMTS: A novel dynamic load balancing scheduling scheme under SLA constraints in cloud computing", Journal of Parallel and Distributed Computing, vol. 149, pp. 138-148, 2021.
[11] Kong, L., Mapetu, J.P.B. & Chen, Z. Heuristic Load Balancing Based Zero Imbalance Mechanism in Cloud Computing. J Grid Computing, vol.18, pp.123–148, 2020.

[12] N. S. Dey and T. Gunasekhar, "A Comprehensive Survey of Load Balancing Strategies Using Hadoop Queue Scheduling and Virtual Machine Migration," in IEEE Access, vol. 7, pp. 92259-92284, 2019.
[13] Mahesh Balaji, Ch. Aswani Kumar, G. Subrahmanya V.R.K. Rao, "Predictive Cloud resource management framework for enterprise workloads",Journal of King Saud University - Computer and Information Sciences, vol. 30, Issue 3, pp. 404-415, 2018.
[14] H. Saleh, H. Nashaat, W. Saber and H. M. Harb, "IPSO Task Scheduling Algorithm for Large Scale Data in Cloud Computing Environment," in IEEE Access, vol. 7, pp. 5412-5420, 2019.
[15] K. Tsakalozos, V. Verroios, M. Roussopoulos and A. Delis, "Live VM Migration Under Time-Constraints in Share-Nothing IaaS-Clouds," in IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 8, pp. 2285-2298, 1 Aug. 2017.
[16] D. Çavdar, L. Y. Chen and F. Alagoz, "Priority Scheduling for Heterogeneous Workloads: Tradeoff Between Evictions and Response Time," in IEEE Systems Journal, vol. 11, no. 2, pp. 684-695, June 2017.
[17] G. Liu, H. Shen and H. Wang, "Deadline Guaranteed Service for Multi-Tenant Cloud Storage," in IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 10, pp. 2851-2865, 1 Oct. 2016.
[18] S. Haider and B. Nazir, "Dynamic and Adaptive Fault Tolerant Scheduling With QoS Consideration in Computational Grid," in IEEE Access, vol. 5, pp. 7853-7873, 2017.
[19] A. Wolke, M. Bichler and T. Setzer, "Planning vs. Dynamic Control: Resource Allocation in Corporate Clouds," in IEEE Transactions on Cloud Computing, vol. 4, no. 3, pp. 322-335, 1 July-Sept. 2016.
[20] Q. Wu, F. Ishikawa, Q. Zhu, Y. Xia and J. Wen, "Deadline-Constrained Cost Optimization Approaches for Workflow Scheduling in Clouds," in IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 12, pp. 3401-3412, 1 Dec. 2017.
[21] Y. Ran, J. Yang, S. Zhang and H. Xi, "Dynamic IaaS Computing Resource Provisioning Strategy with QoS Constraint," in IEEE Transactions on Services Computing, vol. 10, no. 2, pp. 190-202, 1 March-April 2017.
[22] S. Abrishami, M. Naghibzadeh, Deadline-constrained workflow scheduling in software as a service Cloud, Scientia Iranica, vol. 19, Issue 3,pp. 680-689, 2012.
[23] J. P. B. Mapetu, Z. Chen and L. Kong, "Heuristic Cloudlet Allocation Approach Based on Optimal Completion Time and Earliest Finish Time," in IEEE Access, vol. 6, pp. 61714-61727, 2018.
[24] L. Wang and E. Gelenbe, "Adaptive Dispatching of Tasks in the Cloud," in IEEE Transactions on Cloud Computing, vol. 6, no. 1, pp. 33-45, 1 Jan.-March 2018.
[25] Z. Cai, X. Li and R. Ruiz, "Resource Provisioning for Task-Batch Based Workflows with Deadlines in Public Clouds," in IEEE Transactions on Cloud Computing, vol. 7, no. 3, pp. 814-826, 1 July-Sept. 2019.
[26] Z. Zhong, K. Chen, X. Zhai and S. Zhou, "Virtual machine-based task scheduling algorithm in a cloud computing environment," in Tsinghua Science and Technology, vol. 21, no. 6, pp. 660-667, Dec. 2016.
[27] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy and Y. Tian, "Adaptive Energy-Aware Algorithms for Minimizing Energy Consumption and SLA Violation in Cloud Computing," in IEEE Access, vol. 6, pp. 55923-55936, 2018.
[28] Sohani M., Jain S.C., "State-of-the-Art Survey on Cloud Computing Resource Scheduling Approaches", Ambient Communications and Computer Systems. Advances in Intelligent Systems and Computing, Springer, Singapore, vol. 696, 2018.
[29] Longxin Zhang, Liqian Zhou, Ahmad Salah, "Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments", Information Sciences, vol. 531, pp. 31-46, 2020.
[30] Minggang Dong, Lili Fan, Chao Jing, "ECOS: An efficient task-clustering based cost-effective aware scheduling algorithm for scientific workflows execution on heterogeneous cloud systems", Journal of Systems and Software, vol. 158, 2019.
[31] V. Priya, C. Sathiya Kumar, Ramani Kannan, "Resource scheduling algorithm with load balancing for cloud service provisioning", Applied Soft Computing, vol. 76, pp. 416-424, 2019.
[32] Y. Fan, L. Tao and J. Chen, "Associated Task Scheduling Based on Dynamic Finish Time Prediction for Cloud Computing", IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, pp. 2005-2014, 2019.

[33] G. Jia, G. Han, J. Jiang, N. Sun and K. Wang, "Dynamic Resource Partitioning for Heterogeneous Multi-Core-Based Cloud Computing in Smart Cities", in IEEE Access, vol. 4, pp. 108-118, 2016.
[34] Cho, KM., Tsai, PW., Tsai, CW. et al., "A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing", Neural Comput & Applic, vol. 26, pp. 1297–1309, 2015.
[35] M. Divyaprabha, V. Priyadharshni and V. Kalpana, "Modified Heft Algorithm for Workflow Scheduling in Cloud Computing Environment", Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, pp. 812-815, 2018.
[36] M. Chiang, H. Hsieh, W. Tsai and M. Ke, "An improved task scheduling and load balancing algorithm under the heterogeneous cloud computing network", IEEE 8th International Conference on Awareness Science and Technology (iCAST), Taichung, pp. 290-295, 2017.
[37] Kong, L., Mapetu, J.P.B. & Chen, Z., "Heuristic Load Balancing Based Zero Imbalance Mechanism in Cloud Computing", Journal of Grid Computing, vol. 18, pp. 123–148, 2020.
[38] Seth, S., Singh, N., "Dynamic heterogeneous shortest job first (DHSJF): a task scheduling approach for heterogeneous cloud computing systems", International Journal of Information Technology, vol. 11, pp. 653–657, 2019.
[39] Lin, W., Peng, G., Bian, X. et al., "Scheduling Algorithms for Heterogeneous Cloud Environment: Main Resource Load Balancing Algorithm and Time Balancing Algorithm", J Grid Computing, vol. 17, pp. 699–726 2019.
[40] Hussain, A., Aleem, M., Khan, A. et al., "RALBA: a computation-aware load balancing scheduler for cloud computing"' Cluster Computing, vol. 21, pp. 1667–1680 2018.
[41] D. Chitra Devi, V. Rhymend Uthariaraj, "Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Nonpreemptive Dependent Tasks", The Scientific World Journal, vol. 2016, pp. 14, 2016.
[42] M. S. Kumar, I. Gupta and P. K. Jana, "Forward Load Aware Scheduling for Data-Intensive Workflow Applications in Cloud System", International Conference on Information Technology (ICIT), Bhubaneswar, pp. 93-98, 2016.
[43] 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", in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 6, pp. 1239-1254, June 2020.
[44] A. Pellegrini, P. Di Sanzo and D. R. Avresky, "Proactive Cloud Management for Highly Heterogeneous Multi-Cloud Infrastructures", IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Chicago, IL, pp. 1311-1318, 2016.

## AUTHOR

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

**Dr. S. C. JAIN** has done his PG in Computer Science and Technology from IIT Roorkee and Ph.D. in VLSI design from IIT, Delhi. He has served Defence Research and Development (DRDO), Bangalore, India, and presently working as Professor, Computer Science and Engineering Department, Rajasthan Technical University, Kota, Rajasthan, India. His research work focuses particularly on High Performance Computing System, VLSI Design, Real Time Embedded System, and Reversible Computing.