Efficient Task Scheduling and Fair Load Distribution Among Federated Clouds

Rajeshwari B S[1*], M. Dakshayini[2] & H.S. Guruprasad[3]

[1]Department of CSE, B.M.S College of Engineering, Bangalore 560019, India
[2]Department of ISE, B.M.S College of Engineering, Bangalore 560019, India
*E-mail id: madurahr@gmail.com

Abstract. The federated cloud is the future generation of cloud computing, allowing sharing of computing and storage resources, and servicing of user tasks among cloud providers through a centralized control mechanism. However, a great challenge lies in the efficient management of such federated clouds and fair distribution of the load among heterogeneous cloud providers. In our proposed approach, called QPFS_MASG, at the federated cloud level, the incoming tasks queue are partitioned in order to achieve a fair distribution of the load among all cloud providers of the federated cloud. Then, at the cloud level, task scheduling using the Modified Activity Selection by Greedy (MASG) technique assigns the tasks to different virtual machines (VMs), considering the task deadline as the key factor in achieving good quality of service (QoS). The proposed approach takes care of servicing tasks within their deadline, reducing service level agreement (SLA) violations, improving the response time of user tasks as well as achieving fair distribution of the load among all participating cloud providers. The QPFS_MASG was implemented using CloudSim and the evaluation result revealed a guaranteed degree of fairness in service distribution among the cloud providers with reduced response time and SLA violations compared to existing approaches. Also, the evaluation results showed that the proposed approach serviced the user tasks with minimum number of VMs.

Keywords: *cloud computing; fair load distribution; federated cloud; service level agreement; task scheduling.*

1 Introduction

Federated cloud computing is a recent trend in cloud computing, where a large cloud is formed by different cloud service providers who collaborate to provide better cloud services [1,2]. This federated cloud is coordinated by a Federated Cloud Broker (FCB), which interacts with the different cloud service providers [3]. The cloud service providers make an agreement with the FCB for sharing their resources with specific details of economical and technical aspects [4,5]. The economic as well as operational benefits of cloud computing encourage users to send complex applications and data to the cloud [6].

Received July 8th, 2021, Revised September 8th, 2021, Accepted for publication October 4th, 2021.
Copyright © 2021 Published by IRCS-ITB, ISSN: 2337-5787, DOI: 10.5614/itbj.ict.res.appl.2021.15.3.2

There is a tremendous growth in service requests so that the availability of resources at the right moment and optimal distribution of requests among the federated cloud providers as well as within the cloud while meeting the stringent service requirements become nontrivial [7].

Load distribution promotes availability of cloud resources and enhances performance. Hence, optimal distribution of the workload among the federated cloud providers as well as within the cloud with changeable capacities and functionality are important and challenging research topics. Creating a fair load distribution in the federated cloud real-time is a huge challenge. To tackle this, researchers have focused on federated cloud architecture design and dynamic scheduling of tasks in the federated cloud, considering performance parameters such as quality of service (QoS), price, SLA, CPU utilization, elasticity, etc. [8,9].

This paper proposes a two-fold hierarchical scheduling approach, called QPFS_MASG. At the federated cloud level it uses the Queue Partitioned based Fair Load Distribution System (QPFS) for a fair load distribution among the cloud service providers, while at the cloud level it uses the Modified Activity Selection-based Task Scheduling by Greedy (MASG) technique to enhance the cloud performance by distributing the tasks to the most appropriate virtual machines (VMs), considering the task deadline as the key QoS factor while maintaining SLA as well as response time.

The contributions of this paper are as follows:

1. Introduction of the concept of queue partitioning at the federated cloud level, which treats all cloud providers equally and takes care of a fair distribution of the load.
2. Optimal task scheduling among the VMs within the cloud inspired by the Activity Selection algorithm using the Greedy technique, considering the task deadline as the key QoS parameter, reducing SLA violations while maintaining quality of service.
3. Providing an example that demonstrates the servicing of user tasks with the least number of VMs at any time t in the cloud compared to existing approaches.
4. Systematic study with mathematical evidence to show task distribution based on a modified version of the Activity Selection algorithm using the Greedy technique in a cloud environment.
5. Performance analysis of the QPFS_MASG approach with respect to existing algorithms.

The rest of this paper is organized as follows. Section 2 discusses related works on existing task distribution techniques and federated cloud architecture. Section 3 describes the mathematical model of the proposed QPFS_MASG approach. Section 4 presents the proposed architecture and algorithms. Section 5 describes the experimental setup for the evaluation of the proposed method and Section 6 discusses the performance evaluation that was carried out by comparing the performance of the proposed approach with that of existing approaches. Finally, the conclusion section highlights the main contributions of this paper.

2 Literature Review

Wei, *et al.* [10] have proposed an approach that chooses data centers based on a defined latency. This approach uses two techniques: K-means and Binary Quadratic Programming. Datacenters are classified based on their latency using the K-means technique. The proposed approach calculates the latency among several data centers and then finds the best low-latency data center.

Xu, *et al.* [11] have proposed contract-based resource sharing in federated clouds. This proposed model maximizes the revenue and also allocates the resources fairly among the cloud service providers.

Zhao, *et al.* [12] presented a novel resource allocation mechanism. Under this model, the resources are divided equally among all users and no users claim the allocation of other users, improving their own allocation. The experimental result showed better resource utilization.

Habibi, *et al.* [13] implemented an efficient approach for dispatching customer requests among multiple clouds in a federated cloud that decides the distribution of the requests by examining the use of the coefficient of variation and other associated statistical metrics. The behavior of individual's request is checked at a single time frame, dispatching them among multiple cloud providers in one go.

Taha, *et al.* [14] have proposed SLA based service selection, choosing a combination of services that satisfies the customer requirements optimally by assessing the service levels provided by various providers in a multi-cloud environment. The presented approach automatically detects conflicts resulting from dependencies among the selected services and provides an explanation of identified conflicts to the providers as well as the customers in order to resolve conflicts.

Levin, *et al.* [15] implemented load balancing as service architecture for federated clouds. Applications are divided into smaller components and distributed across the clouds for appropriate load balancing.

Motwani, *et al.* [16] discussed three different implementations of service broker algorithms: 1) Service Proximity Service Broker Policy: the broker chooses the quickest path from the user's site to the cloud; 2) Best Response Time Service Broker Policy: the broker monitors the responses of all clouds and directs requests to the cloud that gives the best response time; 3) Dynamic Service Broker Policy: not fully implemented.

Kumar, *et al.* [17] discussed reliability issues of cloud providers. Hence, they designed a scheduling algorithm that balances the load dynamically among all VMs by scaling up and scaling down resource capacities dynamically on the basis of the last optimal k-interval. The developed deadline constrained algorithm maximizes the number of tasks, meeting the deadline and reducing the make span.

Sharma, *et al.* [18] presented the novel Efficient VM Load Balancing algorithm. Upon receiving a request from the user, the algorithm estimates the expected response time at the different VMs and then selects the most appropriate VM.

Shahidinejad, *et al.* [19] presented a model that uses the Imperialist Competition algorithm and the K-means algorithm for clustering the workload and a decision-tree algorithm to decide on scaling for effective resource provisioning. The proposed model minimizes cost, response time and also increases CPU utilization and scalability.

Aslanpour *et al.* [20] have proposed a 3D mechanism for resource provisioning. The resources are allocated based on SLA, resource and user behavior features. The presented mechanism uses a radial basis neural function network to provide flexibility and providence features. The proposed 3D mechanism minimizes the cost with guaranteed quality of service.

Shahidinejad, *et al.* [21] have proposed a Colored Petri Nets (CPN) model and a queueing system to manage cloud infrastructures automatically. The Colored Petri Net model comprises three transitions and four places. The queueing system adjusts the number of VMs on the basis of current load automatically. The proposed system reduces the response time and also enhances resource utilization and scalability.

Ghobaei-Arani, *et al.* [22] presented a framework for controlling the resource elasticity through buffer management and elasticity management. The input queue of user requests is controlled by a buffer manager and the elasticity of the cloud platform is controlled by an elastic manager using a learning automata technique. The proposed framework reduces the response time and also enhances the resource utilization and elasticity.

All these works have majorly concentrated on developing various scheduling algorithms to reduce the response time to user requests, as shown in Table 1. In this work, we propose fair load distribution among federated cloud providers and an efficient task scheduling algorithm to achieve a better response time to user requests in federated cloud environments, increasing the availability of resources.

Table 1 Evaluation tools, workload type and limitations of existing work.

Authors and Year	Model/ Mechanism	QoS Parameters Analyzed	Evaluation Tools Used	Workload Type	Limitation
Wei, *et al.* [10]	K-Means and Binary Quadratic Programming Based Resource Allocation Model	Latency, monetary overhead	Simulation using Python with PyCharm	Real-world datacenters trace	Constraints such as task deadline and power consumption are not considered.
Xu, *et al.* [11]	Contract-Based Resource Sharing Model among Multiple Cloud Providers	Resource utilization, response time, revenue maximization for cloud providers	Trace-driven simulation with realistic workload traces	Google cluster trace	Scheduling technique does not guarantee a balanced load. Reliability and fault-tolerance requirements are not considered.
Zhao, *et al.* [12]	Dominant Resource with Bottlenecked Fairness Resource Allocation Model	Resource utilization, fair resource allocation among users	CloudSim simulation tool	Request generated in a simulation scenario	The technique is not able to provide an intuitively fair allocation, since some users with dominant resources may not able to increase their allocations.
Habibi, *et al.* [13]	Coefficient of Variation and Statistical Metrics Based Request Scheduling Model	Resource utilization, efficient request distribution among multiple cloud providers	Mathematical model and simulation	Google Cloud tracelog data	Constraints such as task deadline are not considered.
Taha, *et al.* [14]	Multi-Cloud Service Selection and Scheduling Model	Efficient service composition from multiple cloud providers, SLA	Evaluated by case study	Real-world data with SLA structure as per the ISO/IEC 19086 standards.	Scheduling technique does not give the guarantee of load balancing.
Levin, *et al.* [15]	Hierarchical Load Balancing Model	Fair load distribution across multiple cloud providers	Open stack	ApacheBench-based plan	Constraints such as cost and heterogeneous virtual machines are not considered.

Authors and Year	Model/ Mechanism	QoS Parameters Analyzed	Evaluation Tools Used	Workload Type	Limitation
Motwani, *et al.* [16]	Profit Based Data Center Service Broker Policy	Minimizes the cost of running the resources over the cloud, providing QoS	CloudSim Simulation tool	Request generated in a simulation scenario	Constraints such as task deadline are not considered. Scheduling technique does not guarantee a balanced load.
Kumar, *et al.* [17]	Deadline Constrained Based Scheduling Model	Task deadline, make-span, load balancing, horizontal scalability	CloudSim Simulation tool	Request generated in the simulation scenario	If threshold value and last k optimal interval value is changed, the simulation results of the model may vary. Does not consider cost for ensuring high priority requests.
Sharma, *et al.* [18]	Efficient VM Load Balancing Model	Response time	CloudSim Simulation tool	Request generated in a simulation scenario	Constraints such as task deadline and fault tolerance are not considered.
Shahidinejad, *et al.* [19]	Hybrid Resource Provisioning Model	Cost, Response time, CPU utilization, scalability	CloudSim Simulation tool	Two real workload traces are used: FIFA traces and NASA traces	Fairness and power consumption parameters in resource provisioning are not considered
Aslanpour *et al.* [20]	Autonomic Resource, SLA and User Behavior Aware Resource Provisioning Mechanism	Cost, resource utilization, QoS, horizontal scaling	CloudSim Simulation tool	NASA's real dataset	Vertical scaling of the resources is not considered.
Shahidinejad, *et al.* [21]	An Elastic Controller using Colored Petri Nets System	Response time, resource utilization, scalability	CloudSim Simulation tool	Three real workload traces: Google Cluster traces, Yahoo Cluster traces, Wikipedia traces	Power consumption parameters in cloud infrastructure management are not considered.
Ghobaei-Arani, *et al.* [22]	Controlling Resources Elasticity Approach	Response time, resource utilization, scalability	CloudSim Simulation tool	Three types of real workloads: FIFA World Cup, Clark Net and NASA	Power consumption parameters in cloud resource management are not considered.

3 *QPFS_MASG* Mathematical Model

The federated cloud architecture considered for the proposed work is as shown in Figure 1. It consists of *M* multiple clouds {$\boldsymbol{cloud_1}$, $\boldsymbol{cloud_2}$..., $\boldsymbol{cloud_M}$} forming a federated cloud environment (FCE) that is managed centrally by a federated cloud manager (FCM). Each cloud provider dedicates *m* hosts {$\boldsymbol{h_1}$, $\boldsymbol{h_2}$... $\boldsymbol{h_m}$} of different hardware configurations to the FCE. Thus, the total infrastructure capacity (TIC) of the federated cloud is given by Eq. (1).

$$FC_{TIC} = \sum_{C=1}^{M}\sum_{i=1}^{m} h^{i}{}_{cpu,mem,bw} \tag{1}$$

When the tasks of varied size arrive at the FCM, it needs to schedule all these tasks optimally among *M* clouds. Scheduling of all these tasks must be done in such a way that it accomplishes a fair load distribution among the federated cloud providers based on their resource contribution to the FCE, reduced waiting time and response time, with good service quality while maintaining SLA.

The objectives of the proposed *QPFS_MASG* approach are:

1. Fair distribution of the load among the multiple cloud providers of the FCE.
   $\boldsymbol{C_1^{LoadLevel\%}} \simeq \boldsymbol{C_2^{LoadLevel\%}} \simeq \ldots \boldsymbol{C_M^{LoadLevel\%}}$
2. Minimizing the response time $\boldsymbol{R_T^t}$ of user task *t* and reducing the SLA violation rate.
   Minimize $\boldsymbol{R_T^t} = \boldsymbol{\omega_T^t} + \mu_{\boldsymbol{T}}^{\boldsymbol{t}}$ in such a way that $\boldsymbol{R_T^t} <= \boldsymbol{DL^t}$
3. Providing service to all user requests with the minimum number of VMs in the cloud.

This work integrated cloud providers with different capacities, classified as large-sized, medium-sized and small-sized providers, in a federated cloud. At the federated cloud level, the Queue Partitioned based Fair Service Distribution System (QPFS) is proposed to consider all these providers impartially and to assign the tasks fairly among them based on the resources they dedicate to the FCE. All the incoming tasks are directed through the main queue Q_{Fc} at the FCM.

QPFS partitions this main queue and distributes the tasks among *M* clouds of the FCE based on the computed load level factor (LLF) at each cloud, such that the load is distributed fairly among all clouds. The Modified Activity Selection based Task Scheduling by Greedy (MASG) technique is adopted within the cloud for scheduling the tasks among VMs, considering the deadline as the main QoS parameter and minimizing the number of VMs running at any time t in the cloud to satisfy the user requests.

4 *QPFS_MASG* Architecture and Algorithms

The proposed federated cloud-based QPFS_MASG architecture with M clouds $\{\boldsymbol{cloud_1}, \boldsymbol{cloud_2}\ldots, \boldsymbol{cloud_M}\}$, each cloud with m hosts $\{\boldsymbol{h_1}, \boldsymbol{h_2}\ldots\ \boldsymbol{h_m}\}$ and varying hardware configurations works in collaboration with the FCM, the cloud broker and the users. The FCM controls and manages all the resources of the different clouds among multiple user requests in the FCE and represents the contact point for users to connect with the federated cloud. Each cloud is managed by a cloud broker, providing the required computing resources and services for the user tasks.

At the federated cloud level, QPFS distributes the load by partitioning the task queue based on the calculated LLF, and at the cloud level, MASG schedules the tasks on the VMs based on the Activity Selection by Greedy technique, considering the deadline as the main QoS parameter.

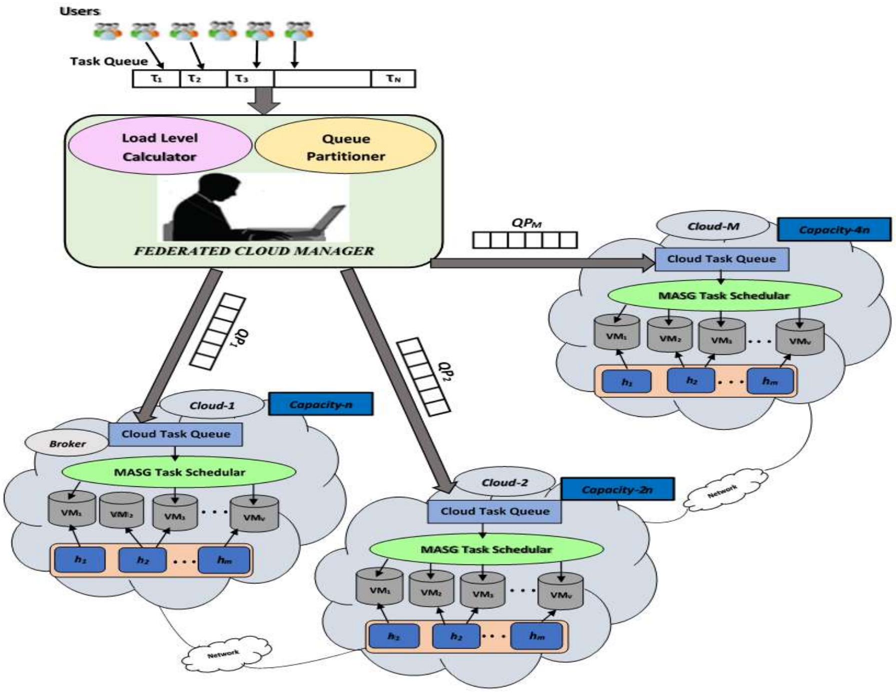


Figure 1 QPFS_MASG architecture.

Figure 2 shows a sequence diagram of the interaction between the components of the QPFS_MASG architecture.

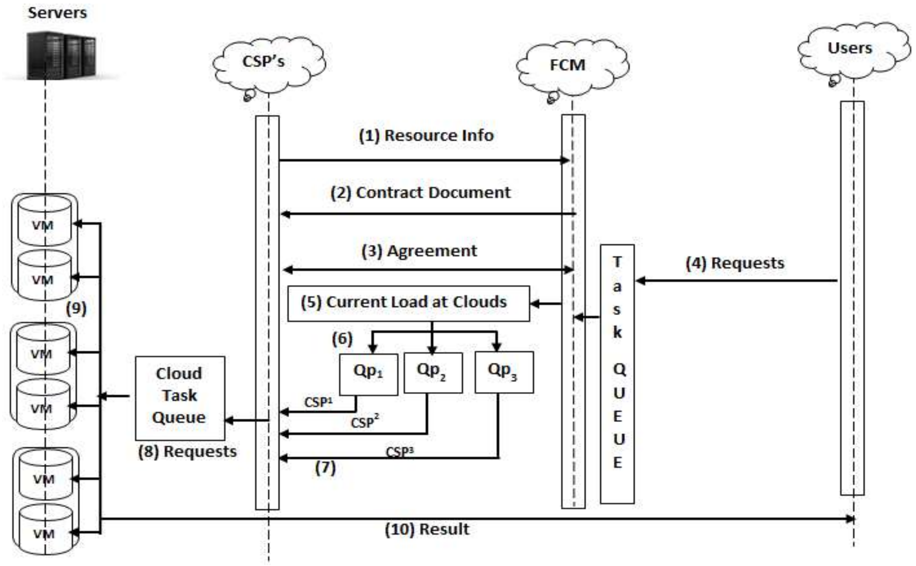


Figure 2 Sequence diagram of the QPFS_MASG architecture.

4.1 Queue Partitioned based Fair Service Distribution System (QPFS)

At the FCE, all cloud providers register with the FCM, offering their resources to the federated cloud. Each cloud provider provides the details of their total resources available and dedicated to the FCE, the current usage of these resources, the list of available services as well as other economic and technical aspects. Based on all these parameters, a weightage is assigned to each cloud $[\boldsymbol{W_{Ci}}]$ as per Table 2. When the tasks arrive at the FCE, the FCM puts all the incoming tasks $\boldsymbol{\tau_1, \tau_2, \tau_3 \ldots}$, which are independent in nature, into the main queue, $\boldsymbol{Q_{FC}}$. Each task τ_i, has a deadline $\boldsymbol{DL^t}$ and has task length $\boldsymbol{\tau_i^L}$, defined in terms of MIPS (million instructions per second).

QPFS utilizes the following two modules: i) a load level calculator module, ii) a queue partitioning module.

The load level calculator module calculates load level factor $[C_i^{LLF}]$ at each cloud, C_i. This is the ratio between the present utilization of resources ${C_i}^{PUR}$ allocated to the FCE and the total resources contributed ${C_i}^{TRC}$ to the FCE, as specified in Eq. (2).

$$\boldsymbol{C_i^{LLF}} = \frac{\boldsymbol{{C_i}^{PUR}}}{\boldsymbol{{C_i}^{TRC}}} \tag{2}$$

The queue partitioning module used in the proposed approach, which makes decisions on partitioning the complete set of user requests, balances and takes care of a fair distribution of the incoming load among the cloud providers based on their resource contribution to the FCE. The queue partitioner module analyzes the LLF at each cloud and partitions the incoming task queue Q_{FC} into M sub queues according to the value of C_i^{LLF}.

Q_{p1}, Q_{p2}, Q_{p3}, . . . Q_{pM}, where each sub queue Q_{pi} has a different size

$$\boldsymbol{Q_{FC} = \sum_{i=1}^{M} Q_{pi}}$$

Algorithm 1: Queue Partitioning (Q)

Input: Task queue $\boldsymbol{Q_{FC}}$ at federated cloud level

Output: *M* sub queues $\boldsymbol{Q_{p1}, Q_{p2}, Q_{p3}, \ldots Q_{pM}}$

```
1. Find a cloud with the largest load level factor at time t among M clouds
   $C^{LrLLF}$ = $C_1$

   for each cloud $C_i$, where i = 2 to M
       if ($C_i^{LLF}$ >= $C^{LrLLF}$)
           $C^{LrLLF} \leftarrow C_i^{LLF}$
       end if
   end for

2. Balance load level factor of the remaining clouds to equalize the load with the
   largest load level factor
   for each cloud $C_i$ where i = 1 to M
       $C_i^{BLLF}$ = $C^{LrLLF}$ - $C_i^{LLF}$
   end for
3. Partitioning the main queue $Q_{FC}$ with N number of tasks into M sub queues, one
   for each $C_i$. $Q_{pi}$ is assigned based on $C_i^{BLLF}$ of $C_i$ and the weight of $C_i$, except for
   the cloud with the largest load level such that the factors of all clouds are
   balanced.

   for each cloud $C_i$ where i = 1 to M
       if ( $Ci$ != $C^{LrLLF}$ )
           $Q_{pi}^{TNT} \leftarrow$ Task allotment based on $C_i^{BLLF}$ and $W_{Ci}$
       end if
   end for
4. If $Q_{FC}$ still has tasks, partition $Q_{FC}$ and add to $Q_{pi}$ of all $C_i$ based on $W_{Ci}$ such
   that the load level factors of all clouds are the same
   if [ $\sum_{i=1}^{M} Q_{pi}^{TN}$ < N]
       for each cloud $C_i$ where i = 1 to M
           $Q_{pi}^{TNT} \leftarrow$ Task allotment based on $W_{Ci}$
```

```
        end for
    end if
5.  End
```

Thus, the FCM dispatches the set of *N* tasks among *M* clouds of FCE based on its defined weight and estimated LLF.

4.2 Optimum Task Scheduling Using Modified Activity Selection Algorithm by Greedy Approach (MASG)

The Activity Selection algorithm using Greedy technique is a combinatorial optimization technique for the selection of non-conflicting activities that always gives an optimal solution.

At the cloud level, within each cloud an MASG is invoked for optimal scheduling of tasks among the VMs based on the deadline constraint of the tasks. MASG uses the concept of the Activity Selection algorithm using the Greedy technique. Here, FCM sends *k* tasks τ_1, τ_2, τ_3. . . τ_k to the cloud queue for the execution of the tasks in that cloud.

MASG schedules a task to the i^{th} virtual machine VM_i considering the finishing time of running task j at VM_i [$ft(runtask_j(VM_i))$]. However, the real difficulty is in choosing the maximum number of tasks that can be allotted to the VMs with the goal of getting all these tasks completed with the minimum number of VMs by the constraint that a VM can execute only one single task at a time [23].

This MASG approach further calculates the processing time of task i [$task_i$] on VM_i [$pt\ (task_i\ (VM_i))$] and analyzes whether the task can be accomplished within the deadline on VM_i. The processing time is calculated based on the CPU and memory configuration on VM_i as well as the task requirements.

Thus, the proposed *MASG* maintains two queues: $Queue_{Start\_Time}$ and $Queue_{Deadline}$. $Queue_{Start\_Time}$ contains the start times of tasks $\tau^{\mathbf{st}}$ and $Queue_{Deadline}$ contains the deadlines of tasks DL^t. On the basis of the defined deadlines, $Queue_{Deadline}$ is organized in increasing order in conjunction with $Queue_{Start\_Time}$. Hence, MASG starts mapping $task_i$ onto VM_i such that

$$\boldsymbol{\tau_i^{st}}(\mathrm{VM_i}) >= \mathrm{ft}\ (\mathrm{runtask_j}\ (\mathrm{VM_i})) \tag{3}$$

and

$$\mathrm{ft}\ (\mathrm{runtask_j}) + \mathrm{pt}\ (\mathrm{task_i}\ (\mathrm{VM_i})) < = \boldsymbol{DL^t} \tag{4}$$

Once task $task_i$ is mapped onto $VM_{i,}$ the response time of $task_i$ becomes the finish time, which is used for mapping the next task in the queue, i.e.

$$\mathrm{ft}(\mathrm{runtask}_j(\mathrm{VM}_i)) \leftarrow \mathrm{response\_time}(\mathrm{task}_i) \quad (5)$$

Eq. (3) ensures that the starting time of the next task *i* is greater than or equal to the finishing time of running task *j* on VM_i.

The processing time of $task_i$ on VM_i is calculated considering the task size and the service rate of VM_i as shown in Eq. (6):

$$\mathrm{pt}\ (\mathrm{task}_i\ (\mathrm{VM}_i)) = \boldsymbol{\tau_i^L} /\ \mathrm{Service\_rate}\ (\mathrm{VM}_i) \quad (6)$$

The main goal of MASG is that the users get good quality of service by servicing their tasks within the deadline, satisfying SLA, improving the response time and servicing the user tasks with the least number of VMs in the cloud at any time t.

Algorithm 2: ***MASG*** $(\boldsymbol{Q_{pi}}, \boldsymbol{V})$

Input: Q_{pi} with *k* tasks $\tau = \{, \tau_1, \tau_2, \ldots \tau_{k\}}$
Set of *VMs* $V = \{VM_1, VM_2, \ldots VM_V\}$

Output: Subset of tasks mapped onto each *VM*

```
1.  for each task $\tau_i$ in $Q_{pi}$ has a deadline
        $Queue_{Deadline}[i]$ ← Estimate deadline of task $\tau_i$ based on the priority of the task
    end for

2.  for each $VM_i$ where $i$ = 1 to V
        for each task $\tau_i$ in $Q_{pi}$
            $Queue_{Start\_Time}$ [i] ← Assess starting time of the task such that it can be serviced
              within its defined deadline on $VM_i$.
        end for
      Sort $Queue_{Deadline}$ on the basis of the task deadlines
        for each task $\tau_i$
            if ( $\tau_i^{st}(VM_i)$ >= $ft(runtask_j(VM_i))$) then
                $pt\ (task_i\ (VM_i)) = \tau_i^L / Service\_rate\ (VM_i)$
                if ($ft\ (runtask_j) + pt(task_i(VM_i)) <= DL^t$) then
                    Map $\tau_i$ to $VM_i$
                    Remove $\tau_i$ from $Q_{pi}$
                  $ft(runtask_i(VM_i))$ ← $response\_time(task_i)$
                end if
            end if
        end for
    end for
3.  End
```

The Activity Selection based Scheduling using Greedy always yields an optimal solution. Thus, MASG optimally maps all the tasks to the least number of VMs. Hence, the proposed technique not only completes the tasks within their deadlines, but also reduces the response time using the minimum number of VMs in the cloud, leading to optimized resource utilization.

4.3 Time Analysis of QPFS_MASG Approach

The Queue Partitioning algorithm analyzes the load level factor at *m* individual clouds and partitions the incoming task queue into *m* sub queues, each with *n* number of tasks. Hence, the time complexity of the Queue Partitioning algorithm is *O(mn).*

Time Complexity (Queue Partitioning) = O (*mn*)

The proposed MASG approach uses the Activity Selection based technique for selecting *n* tasks to a VM based on the start and the finish time of the tasks, which can be solved in *O (n log n)* time.

Time Complexity (MASG) = O (n log n)

5 Experimental Setup

QPFS_MASG is operated in a federated cloud environment, which is hard to implement in practice due to the large size of the network [24]. Hence, the performance of the proposed approach was simulated using the CloudSim report toolkit, including CloudSim version 3.0.3 and NetbeansIDE8.0.

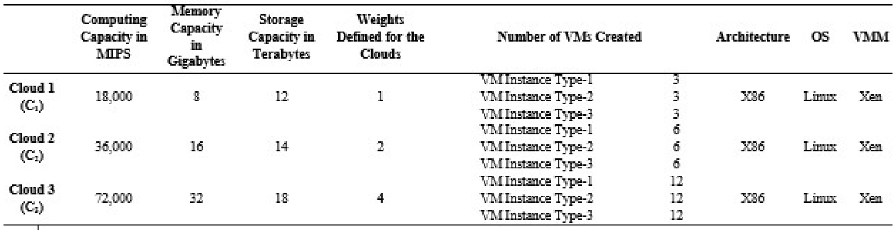
The classes of CloudSim simulator were extended to simulate the proposed algorithms, implemented in Java [25]. The experiment was performed on an Intel Core i7-3770 with a 64-bit Windows 10 platform machine and 4 GB of DDR3 SD RAM.

The FCE setup was made by deploying a set of computational resources across multiple clouds. For the evaluation, three clouds configured with varying capacities were created to evaluate fairness in load distribution among them based on their resource capacity dedicated to the FCE. The cloud configuration settings made for the evaluation are shown in Table 2.

The weight for the clouds was defined based on their resource contribution to the FCE. Three different VM instance types were considered at each cloud in the evaluation to process the incoming tasks, as shown in Table 3.

As shown in Table 2, Cloud 3 had more capacity than Cloud 1 and Cloud 2 with respect to processing speed, memory capacity and storage capacity and hence 12 VMs were created in this cloud. The VMs of type 1 had more capacity then the VMs of types 2 and 3 with respect to processing speed, memory capacity and storage capacity.

Table 2 FCE configuration.



	Computing Capacity in MIPS	Memory Capacity in Gigabytes	Storage Capacity in Terabytes	Weights Defined for the Clouds	Number of VMs Created		Architecture	OS	VMM
Cloud 1 (C_1)	18,000	8	12	1	VM Instance Type-1	3	X86	Linux	Xen
					VM Instance Type-2	3			
					VM Instance Type-3	3			
Cloud 2 (C_2)	36,000	16	14	2	VM Instance Type-1	6	X86	Linux	Xen
					VM Instance Type-2	6			
					VM Instance Type-3	6			
Cloud 3 (C_3)	72,000	32	18	4	VM Instance Type-1	12	X86	Linux	Xen
					VM Instance Type-2	12			
					VM Instance Type-3	12			

Table 3 VM instance configuration.

VM Instance Type	Computing Capacity in MIPS	Memory Capacity in Megabytes	Storage Capacity in Megabytes
VM Instance Type-1	3000	1024	100000
VM Instance Type-2	2000	800	10000
VM Instance Type3	1000	512	10000

6 QPFS_MASG Results and Discussion

To show the effectiveness of the proposed approach, the results were compared with the Service Proximity Service Broker Policy (SPSBP) [10] at the federated cloud level and with Efficient VM Load Balancing (EVMLB) [12] at the cloud level.

As shown in Figure 3, on average, QPFS_MASG dispatched 64.6%, 70% and 66.6% of service tasks to Cloud 1, Cloud 2 and Cloud 3, while SPSBS provided 73%, 34.3%, and 19.86% of service tasks to Cloud 1, Cloud 2 and Cloud 3 respectively.

To achieve a fair and balanced distribution of the load among multiple cloud providers, the QPFS_MASG approach dynamically calculates the LLFs of the individual clouds and partitions the incoming task queue on the basis of the calculated LLFs at time t. Thus, the proposed approach at the federated cloud level achieves fairness in the load distribution among multiple clouds in the FCE based on their resource contribution and avoids overloading of any cloud.

In Figure 4, Task Deadline (denoted by the red line) indicates the deadlines of the tasks. *QPFS_MASG Finishtime* (denoted by the green line) indicates the finish time of the tasks from the QPFS_MASG approach, while *SPSBP Finishtime* (denoted by the blue line) depicts the finish time from the SPSBP approach.

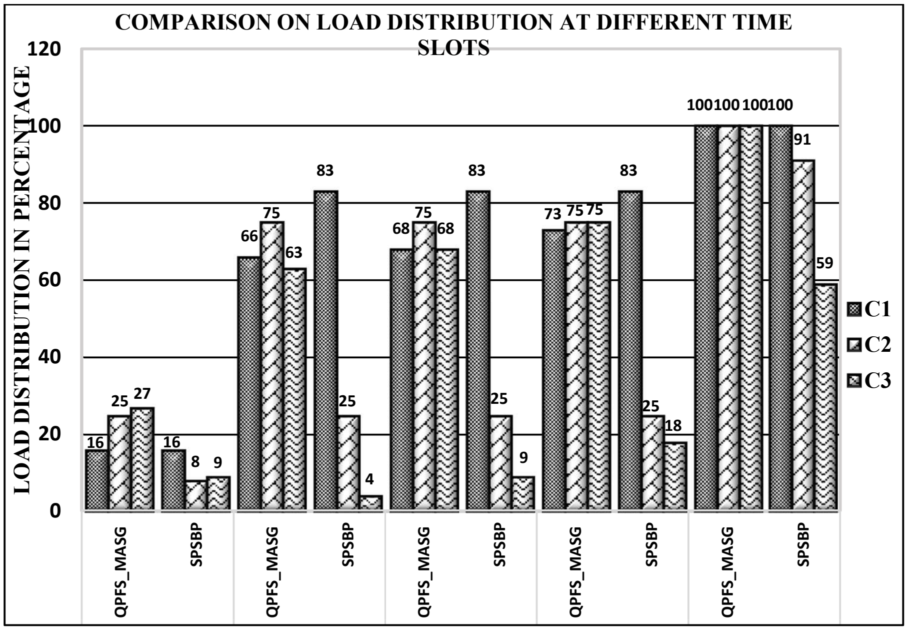


Figure 3 Load distributions at different time slots.

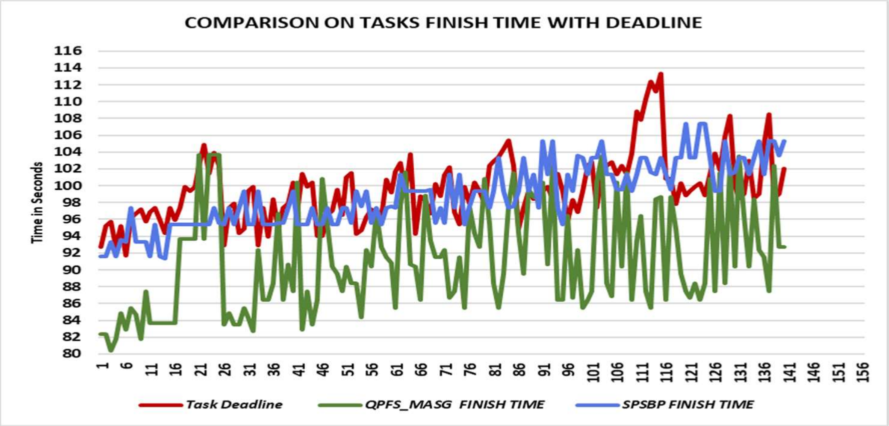


Figure 4 Comparison of task finish time with deadline.

The curve shows that QPFS_MASG serviced the maximum number of tasks before the deadline comparatively. Here, the task deadline was taken as the key parameter. Thus, the task queue was organized in increasing order of deadline and mapped the tasks to the VMs considering the starting time of the next task and the response time of the running task. Further, it analyzed whether the processing time of the task on that VM can be accomplished within its deadline. Thus, almost 90% of the tasks were finished within their deadline.

In Figure 5, a comparison of SLA violations is shown. Figure 5 shows that the QPFS_MASG approach had 10% improvement in adhering to SLA compared to the SPSBP approach. The proposed QPFS_MASG framework improved the average response time of the tasks in the range of 31% to 40% on the different clouds, as shown in Figure 6. The QPFS_MASG framework distributed the tasks based on their resource contribution, so overloading of any cloud was avoided, which reduces the waiting time of the tasks at any cloud, as shown in the Figure 7, and in turn reduced the response time.

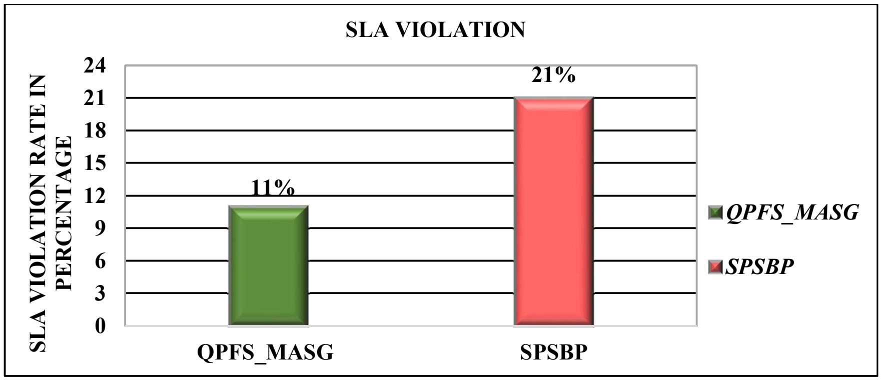


Figure 5 Comparison on SLA violation rate.

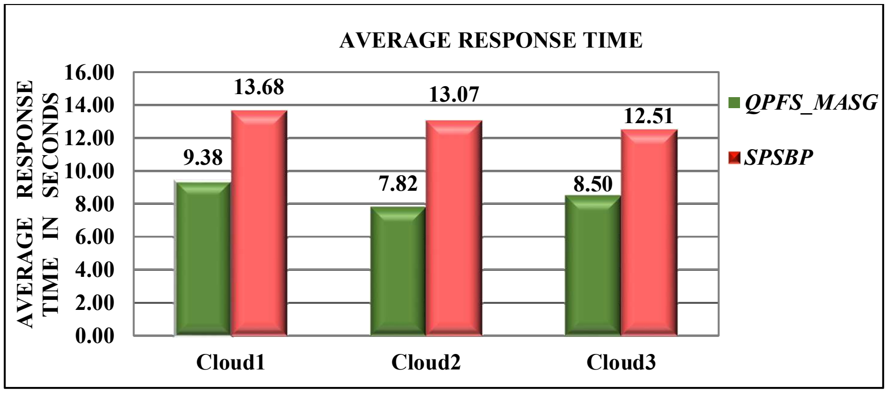


Figure 6 Comparison of average response time.

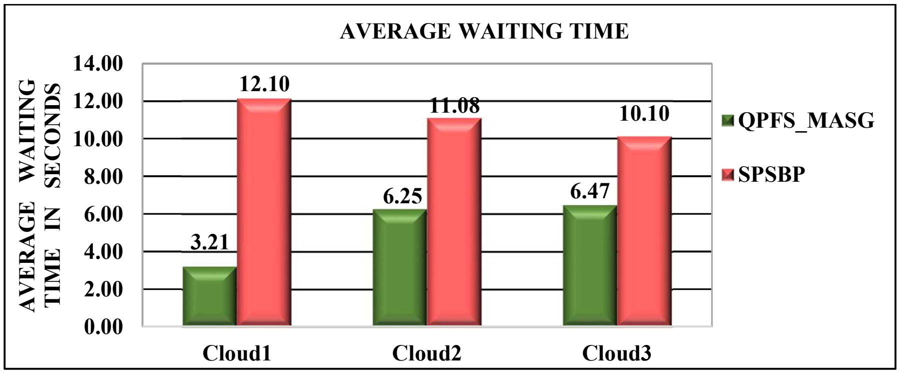


Figure 7 Comparison of average waiting time.

6.1 Example Demonstrating User Task Servicing with Least Number of VMs by MASG Approach Compared to EVMLB Approach

Consider 3 virtual machines VM_1, VM_2, VM_3 with a capacity of 1000 MIPS and 8 tasks arriving at time t. The arrival time, deadline and assessed starting time of the task as well as the number of instructions in each task are shown in Table 4.

Table 4 Arrival time, deadline and number of instructions of the tasks.

Tasks	*T1*	*T2*	*T3*	*T4*	*T5*	*T6*	*T7*	*T8*
Arrival Time	1	1	1	1	1	1	1	1
Starting Time	1	1	8	2	3	5	4	5
Deadline	3	2	9	4	5	7	5	8
Number of Instructions	2000	1000	1000	2000	2000	2000	1000	3000

The tasks in the queue are sorted as per the deadline, is shown in Table 5.

Table 5 Tasks sorted based on deadline.

Tasks	*T2*	*T1*	*T4*	*T5*	*T7*	*T6*	*T8*	*T3*
Arrival Time	1	1	1	1	1	1	1	1
Starting Time	1	1	2	3	4	5	5	8
Deadline	2	3	4	5	5	7	8	9
Number of Instructions	1000	2000	2000	2000	1000	2000	3000	1000

Schedule first task T2 to VM_1

ft(T2) = 1 + 1 = 2

Next, schedule tasks to VM_1 iff,

st(task)>=ft(runtask) and *ft(runtask) + pt(task) < dl(task)*

as shown in Table 6.

Table 6 Tasks scheduled on VM1.

Tasks	T1	T4	T5	T7	T6	T8	T3
Arrival Time	1	1	1	1	1	1	1
Starting Time	1	2	3	4	5	5	8
Deadline	3	4	5	5	7	8	9
Number of Instructions	2000	2000	2000	1000	2000	3000	1000
τ_i^{st} >=ft(runtask$_j$) ?	1>2? No	2>=2? Yes	3>=4? No	4>=4? Yes	5>=5? Yes	5>=7? No	8>=7? Yes
ft(runtask$_j$) + pt(task$_i$) < = DL^t?	2 + 2=4<= 3? No	2+2=4< =4? Yes ft(T4) =4	4+2=6 <=5? No	4+1=5 <=5? Yes ft(T7) =5	5+2=7< =7? Yes ft(T6) =7	7+3=10<=8? No	7+1=8<= 9? Yes ft(T3) =8

Thus, tasks scheduled to VM_1 are *T2*, *T4*, *T7*, *T6*, *and T3.*

Step repeats for VM_2. Schedule first task *T1* to VM_2

ft(T1) = 1 + 2 = 3

Next schedule tasks to VM_2 iff,

st(task)>= ft(runtask) and *ft(runtask) + pt(task) < dl(task)*

as shown in Table 7.

Table 7 Tasks scheduled to VM2.

Tasks	T5	T8
Arrival Time	1	1
Starting Time	3	5
Deadline	5	8
Number of Instructions	2000	3000
τ_i^{st} >=ft(runtask$_j$) ?	3>=3? Yes	5>=5? Yes
ft(runtask$_j$) + pt(task$_i$) < = DL^t?	3+2=5<=5? Yes ft(T5) =5	5+3=8<=8? Yes ft(T8)=8

Thus, tasks scheduled to VM_2 are *T1*, *T5*, *and T8*

Using the *EVMLB* [12] approach, if the same tasks scheduled to VMs that offer the minimum expected completion time, then the tasks scheduled to the VMs takes place as follows:

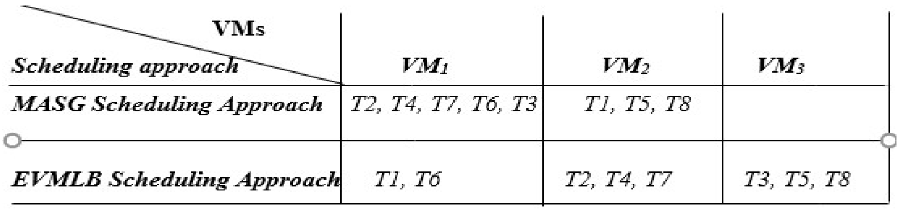
VM_1 → T1, T6

VM_2 → T2, T4, T7

VM_3 → T3, T5, T8

Figure 8 precisely illustrates the effectiveness of task allocation by the MASG approach compared to the EVMLB approach. The MASG approach optimally scheduled the tasks to VMs and services 8 tasks with 2 VMs within the deadline, whereas the EVMLB approach used 3 VMs to service the same number of tasks.

MASG tries to assign tasks to a VM as long as the tasks can be finished within the deadline and also avoids creating and assigning tasks to a new VM. Thus, MASG consolidates the tasks onto the least number of VMs.



VMs *Scheduling approach*	*VM_1*	*VM_2*	*VM_3*
MASG Scheduling Approach	*T2, T4, T7, T6, T3*	*T1, T5, T8*	
EVMLB Scheduling Approach	*T1, T6*	*T2, T4, T7*	*T3, T5, T8*

Figure 8 Comparison on VM allocation.

7 Conclusions

In this paper, a hierarchical approach called QPFS_MASG was proposed for efficient distribution of user tasks among multiple cloud providers in an FCE as well as among VMs within the clouds. The proposed QPFS_MASG presents two scheduling approaches at two levels: Queue Partitioned based Fair Service Distribution for Federated Cloud (QPFS) at the federated cloud level and the Modified Activity Selection based Task Scheduling by Greedy (MASG) technique at the cloud level.

Queue partitioned scheduling at the federated cloud level partitions the incoming queue aiming towards a fair distribution of the load among all participating cloud service providers of the federated cloud. MASG at the cloud level schedules tasks among VMs considering the task deadline as the key parameter.

The results obtained showed fairness in load distribution among multiple cloud service providers with an average of 64.6%, 70% and 66.6% of load to Cloud 1, Cloud 2 and Cloud 3, while SPSBP scheduled 73%, 34.3%, and 19.86% of load to Cloud 1, Cloud 2 and Cloud 3 respectively.

The proposed framework also showed that 90% of the tasks were finished before their deadline ended, with an improvement in average response time between 31% and 40%. The presented example demonstrated that the proposed MASG algorithm always consolidated tasks onto the least number of VMs. Thus, the proposed approach not only performed fair distribution of load among multiple clouds, but also enhanced the response time, servicing tasks within their deadline and consolidating tasks onto the minimum number of VMs.

Nomenclature:

C_i	: i^{th} cloud
R_T^t	: response time of task t
μ_T^t	: execution time of task t
ω_T^t	: waiting time of task t
DL^t	: deadline of the task t
C_i^{TRC}	: total resource capacity dedicated by C_i to FCE
C_i^{PUR}	: present usage of resources at C_i
C_i^{LLF}	: load level factor at C_i
C^{LrLLF}	: cloud with largest load level factor
$C_{\mathrm{i}}^{\mathrm{BLLF}}$	: balancing load level factor of C_i
$\mathrm{Q}_{\mathrm{pi}}^{\mathrm{TNT}}$	: queue with total number of tasks for C_i
W_{Ci}	: weight defined for C_i based on contribution of resources by the cloud provider to the FCE
Q_{Fc}	: main queue at federated cloud level

References

[1] Assis, M.R.M, Bittencourt, L.F. & Tolosana-Calasanz, R., *Cloud Federation: Characterization and Conceptual Model*, IEEE/ACM 7th International Conference on Utility and Cloud Computing, pp. 585-590, Dec. 2014. DOI: 10.1109/UCC.2014.90.

[2] Lokesh, V., Jayaraman, S., N, A., Soni, A., & Guruprasad, H.S., *A Survey on the Capabilities of Cloud Simulators*, International Journal of Engineering Research and General Science, **3**(4), pp. 958-969, Aug. 2015.

[3] Rajeshwari, B.S. & Dakshayini, M., *Optimized Bit Matrix-based Power Aware Load Distribution Policy among Federated Cloud*, Elsevier

Procedia Computer Science, **167**, pp. 1771-1790, April 2020. DOI: 10.1016/j.procs.2020.03.387.

[4] Culhane, W., Eugster, P., Jayalath, C., Kogan, K. & Stephen, J., *Cloud Federation and Geo-Distribution*, Encyclopedia of Cloud Computing, John Wiley and Sons Ltd., pp. 178-190, May 2016.

[5] Sudhakara, S., Nithya, N.S. & Radhakrishnana, B.L., *Fair Service Matching Agent for Federated Cloud*, Computers and Electrical Engineering, Elsevier, **76**, pp. 13-23, June 2019. DOI: 10.1016/ j.compeleceng.2019.03.002.

[6] Sindhu, K. & Guruprasad, H.S., *A Performance Analysis on Cloud Based Mobile Augmentation in Mobile Cloud Computing*, McGraw-Hill International Conference on Signal, Image Processing Communication and Automation, JSSATE, Bangalore, pp. 398-403, 6th and 7th June 2017.

[7] Bhavani B.H. & Guruprasad, H.S., *A Comparative Study on Resource Allocation Policies in Cloud Computing Environment*, Compusoft, An International Journal of Advanced Computer Technology, **3**(6), pp. 893-899, June 2014.

[8] Rajeshwari B.S., Dakshayini, M. & Guruprasad, H.S., *Service Level Agreement-based Scheduling Techniques in Cloud: A Survey*, International Journal of Computer Applications, **132**(5), pp. 20-26, Dec. 2015. DOI: 10.5120/ijca 2015907358.

[9] Ghobaei-Arani, M. & Shahidinejad, A., *An Efficient Resource Provisioning Approach for Analyzing Cloud Workloads: A Metaheuristic based Clustering Approach*, The Journal of Supercomputing, **77**(1), pp. 711-750, Jan. 2021. DOI: 10.1007/s11227-020-03296-w.

[10] Wei, J., Zhou, A., Yuan, J. & Yang, F., *AIMING: Resource Allocation with Latency Awareness for Federated-Cloud Applications*, Hindawi: Wireless Communications and Mobile Computing, pp.1-11, April 2018. DOI: 10.1155/2018/4593208.

[11] Xu, J., & Palanisamy, B., *Cost-Aware Resource Management for Federated Clouds using Resource Sharing Contracts*, IEEE 10th International Conference on Cloud Computing, pp. 238-245, June 2017. DOI: 10.1109/CLOUD.2017.38.

[12] Zhao, L., Du, M. & Chen, L., *A New Multi-Resource Allocation Mechanism: A Tradeoff between Fairness and Efficiency in Cloud Computing*, China Communications, IEEE, **15**(3), pp. 57-77, April 2018. DOI: 10.1109 /CC.2018.8331991.

[13] Habibi, M., Fazli, M. A. & Movaghar, A., *Efficient Distribution of Requests in Federated Cloud Computing Environments Utilizing Statistical Multiplexing*, Future Generation Computer Systems, Elsevier, **90**, pp. 451-460, Jan. 2019. DOI: 10.1016/j.future.2018.08.032.

[14] Taha, A., Manzoor, S. & Suri, N., *SLA-Based Service Selection for Multi-Cloud Environments*, IEEE International Conference on Edge Computing, pp. 65-72, June 2017. DOI: 10.1109/ IEEE.EDGE.2017.17.

[15] Levin, A., Lorenz, D., Merlino, G., Panarello, A., Puliafito, A. & Tricomi, G., *Hierarchical Load Balancing as a Service for Federated Cloud Networks*, Computer Communication, Elsevier, **129**, pp. 125-137, Sept. 2018. DOI: 10.1016/j.comcom.2018.07.031.

[16] Motwani, A., Chaturvedi, R. & Shrivastava, A., *Profit Based Data Center Service Broker Policy for Cloud Resource Provisioning*, International Journal of Electrical, Electronics and Computer Engineering, **5**(1), pp. 54-60, 2016.

[17] Kumar, M. & Sharma, S.C., *Deadline Constrained Based Dynamic Load Balancing Agorithm with Elasticity in Cloud Environment*, Computers and Electrical Engineering, Elsevier, **69**, pp. 395-411, July 2018. DOI: 10.1016/j.compeleceng.2017.11.018.

[18] Sharma, M. & Sharma, P., *Performance Evaluation of Adaptive Virtual Machine Load Balancing Algorithm*, International Journal of Advanced Computer Science and Applications, **3**(2), pp. 86-88, 2012. DOI: 10.14569/ IJACSA.2012.030215

[19] Shahidinejad, A., Ghobaei-Arani, M., & Masdari, M., *Resource Provisioning using Workload Clustering in Cloud Computing Environment: A Hybrid Approach*, Cluster Computing, **24**(1), pp. 319-342, March 2021. DOI: 10.1007/s10586-020-03107-0.

[20] Aslanpour, M.S., Dashti, S.E., Ghobaei-Arani, M. & Rahmanian, A.A., *Resource Provisioning for Cloud Applications: A 3-D Provident and Flexible Approach*, The Journal of Supercomputing, **74**(12), pp. 6470-6501, Dec. 2018. DOI 10.1007/s11227-017-2156-x.

[21] Shahidinejad, A., Ghobaei-Arani, M. & Esmaeili, L., *An Elastic Controller using Colored Petri Nets in Cloud Computing Environment*, Cluster Computing, **23**(2), pp. 1045-1071, June 2020. DOI: 10.1007/s105 86-019-02972-8.

[22] Ghobaei-Arani, M., Souri, A., Baker, T. & Hussien, A., *ControCity: An Autonomous Approach for Controlling Elasticity using Buffer Management in Cloud Computing Environment*, IEEE Access 7, Special Section on Mobile Edge Computing and Mobile Cloud Computing: Addressing Heterogeneity and Energy Issues of Compute and Network Resources, pp. 106912-106924, Aug. 2019, DOI: 10.1109/ACCESS.2019.2932462.

[23] Horowitz, E., Sahni, S. & Rajasekhara, *Fundamentals of Computer Algorithms*, 2nd Edition, University Press Pvt. Ltd, 2009.

[24] Singh, D.A.A.G., Priyadharshini, R. & Leavline, E.J., Analysis *of Cloud Environment using CloudSim*, Artificial Intelligence and Evolutionary Computations in Engineering Systems, Springer, pp. 325-333, March 2018.

[25] Goyal, T., Singh, A. & Agrawal, A., *CloudSim: Simulator for Cloud Computing Infrastructure and Modeling*, Procedia Engineering, Elsevier, **38**, pp. 3566-3572, Sep. 2012. DOI: 10.1016/j.proeng.2012.06.412.