Received: September 2, 2021. Revised: October 28, 2021.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Entropy Based Monotonic Task Scheduling and Dynamic Resource Mapping in Federated Cloud Environment

Jeny Varghese[1,2]* **Jagannatha Sreenivasaiah[1]**

[1]*Department of Computer Applications, MS Ramaiah Institute of Technology, Affiliated to VTU, Belagavi, India*
[2]*Department of Science and Humanities, PES University, India*
* Corresponding author's Email: jens4u@gmail.com

Abstract: Multiple cloud computing services are collectively managed as federated cloud. The growth of users into this federated cloud for accessing a variety of services has introduced challenges on resource utilization and load imbalance that consumes larger waiting time of users. To address these two issues, this paper proposes data center (DC) clustering, virtual machine (VM) clustering, resource mapping and task scheduling. Initially the DCs are clustered using region based fuzzy possibilistic C-means clustering (R-FPCM) algorithm. DC clustering is performed by gathering the information of data dependency, million instructions per second (MIPS), latency, storage, bandwidth and counts of VM. Depending on DC clustering, the VMs are clustered by multi-objective density-based spatial clustering based on the estimated capacity and bandwidth. In order to balance load, Markov chain is applied to predict future load and balance accordingly. An intermediate broker is employed to monitor the VM resources and map based on the user's task requirement. Service level agreement (SLA) is satisfied for every user by the broker and then fast 1 to N resource mapping algorithm is involved for mapping resources. This is followed by entropy-based monotonic scheduling algorithm that arranges user tasks in an order that is determined from task type, task size, task arrival time and deadline. The dynamic computation of entropy enables to improve scheduling with the current status of the system. The extended simulations of this proposed system are simulated in Cloudsim and the results are evaluated in terms of execution time, latency, resource utilization and response time and compared with the performance of Capacity based VM clustering algorithm. These simulation results show that the performance of the proposed system is much better than the existing approach.

Keywords: Clustering, Federated clouds, Load management, Resource utilization, Task scheduling.

1. Introduction

Cloud platforms provisions remote data access via internet using modern technologies to cope up with tremendous users' participation [1]. The incoming tasks are scheduled to support scalability in the system. Scheduling of user tasks is handled based on the resource that is utilized in the system. On knowing the availability of resources, the tasks are assigned into cloud for processing. Service-level agreement (SLA) defines the requirements of user which need to be satisfied by the cloud environment [2, 3]. The proper handling of SLA constraints leads to improve quality-of-service (QoS). Generally, the attainment of SLA is essential to improve system performance. Energy efficiency is also associated with SLA constraints which are more peculiar in enabling user requirement. The reduction in SLA violations reflects its impact on the increase in energy efficiency [4]. On satisfying the SLA requirements, the user tasks are processed faster.

The processes of clustering and task scheduling are commonly concentrated in federated clouds for task assignment based on the resources [5]. Task scheduling is supported by optimization algorithm in which the key constraints that are taken in account are deadline and budget [6]. Tasks are scheduled for the purpose if reducing makespan and increasing the resource utilization [7]. Broker acts as intimidator who manages information of datacenter (DC) and

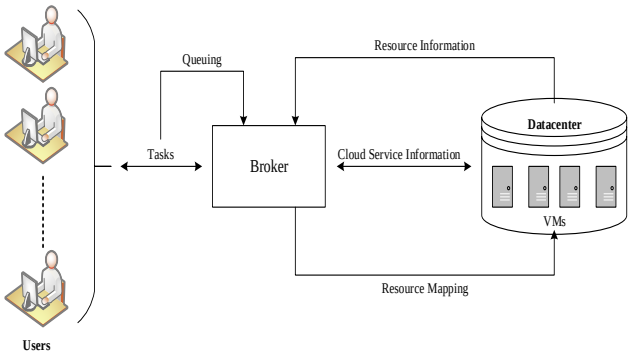

Figure. 1 Broker assisted cloud design

user tasks as shown in Fig. 1.

The common advantages of using federated cloud are: economical, availability at diverse location, satisfied SLA and others [8]. The peculiar challenges that are involved in federated cloud are enlisted as follows,

- Firstly, resource provisioning and resource management based on the incoming user tasks. A scalability supported federated cloud design is required.
- Secondly, satisfaction of SLA for individual user task is critical due to the participation of multiple users.
- Thirdly, load balancing is critical due to the processing speed of users request into the DC.

The above challenges in federated cloud are solved by proposed clustering, resource monitoring and task scheduling. The process of task scheduling and allocation of resources for tasks processing is presented using meta-heuristic algorithm [9, 10]. Clustering was performed using k-means algorithm for improving resource allocation process.

Cloud federation also focuses on allocation of cloud resources with the provisioning of QoS [11]. The major metrics that are included into QoS are execution price, execution time, reliability, availability and security. On behalf of such constraints, the resource allocation process is performed. An efficient allocation of resources results with better utilization of energy in DCs [12], [13]. Hybrid optimization algorithm is enabled to support efficient resource utilization as well as manage SLA violations.

In this paper the objective of designed federated cloud environment is to ensure minimized time. This objective in federated cloud is achieved by constructing DC clustering, VM clusters, load balancing, resource mapping and task scheduling. Based on the clustered VMs the resources are mapped by broker and then allows scheduled tasks for processing. On the other hand, the SLA requirements are also satisfied before being processed into federated cloud.

1.1 Motivation

Federated cloud is developed from smaller clouds and it looks as one. The user task processed into the cloud is more time sensitive [14]. Hereby, extensively the growth of incoming users into federated cloud requests for faster response time. Time constraint is a key constraint in federated cloud that is resolved by proper design. In order to enable faster processing scheduling is incorporated using static threshold or dynamic threshold. Task scheduling is associated with the provisioning of SLA constraints by the use of heuristic algorithms. Also, controlling load in cloud is an essential process that is supported by clustering [15, 16]. Ubiquitous growth of users has increased the demand to access cloud with the requirement of faster processing. The main objective of this proposed federated cloud is to develop a faster processing system.

The faster processing system achieves minimized execution time, latency and response time. These constraints are achieved by incorporating effective clustering, resource mapping, load balancing and task scheduling. Clusters are formed to enable resource mapping and task allocation to satisfied VM. On behalf of the above discussed motive, this system proposes peculiar solution to minimize time of processing and deliver user with lesser response time.

1.2 Research contribution

The major research contributions of this paper are enlisted below,

- User tasks are processed in federated cloud with the aim of reducing execution time, latency and response time. These constraints are achieved by incorporating DC clustering, VM clustering, resource mapping and task scheduling.
- DC clustering and VM clustering are performed in federated cloud using region based fuzzy possibilistic C-means clustering and multi-objective density based spatial clustering respectively. These clustering enable to manage resources by which the appropriate tasks are allotted. Markov chain is involved for predicting the load in the clustered VMs.
- Fast 1 to N resource mapping process is employed in broker for mapping the resource utilization and then allots the scheduled tasks into federated cloud for processing. The process of mapping is ensured to meet SLA requirements of the task.

- The user tasks are scheduled using entropy based monotonic scheduling algorithm by taking in account of task's type, size, arrival time and deadline.

1.3 Paper organization

This paper is organized into following sections as: section II deals with the study of previous research works that have been involved for clustering and task scheduling, section III keenly illustrates the defined problems, section IV details the proposed solutions that resolve the identified problems, section V elaborates the results that are obtained from the implementation and section VI concludes the proposed work along with its future research directions.

2. Literature review

The authors have developed a two-stage strategy for scheduling the incoming tasks [17]. Initially, in the first stage Bayes classifier algorithm was used for classifying the job and then followed by matching with VMs in second stage. In this work, the VMs are previously created by using the collected historical scheduling data. The designed task scheduling framework is composed of task classifier, matcher, ready queue and waiting queue. The precreation of VMs based on historical data was not able to serve current user tasks, since the incoming number of tasks will not be similar as in the previous history. In [18] the authors have performed scheduling in federated clouds using two broker-level schedulers. The algorithms used are ant colony optimization (ACO) and particle swarm optimization (PSO). These optimization algorithms were enabled to pick a datacenter with respect to latency, monetary cost and availability. The weighted values are computed for monetary cost and communication latency based on which the datacenter was preferred. Once, the VMs were assigned, then the tasks enter into first-in-first-out (FIFO) policy for processing. FIFO based task processing in VMs increases waiting time for short tasks. In [32] the authors have given importance to weighted parameters in possibilistic fuzzy c-means which worked to improve the objective function which resolved the problem of overcoming the defect of sensitivity to noisy data.

A flexible model was implemented by new cloud operators who can join or leave the federation [19]. This model has integrated the interactions between broker agent optimization. In this method, the user request was migrated between broker agents to satisfy the user request requirements. Hereby each broker maintains a list of providers. An un-satisfied user request was migrated to another broker agent. In this way the user request migration was performed until the user request requirements were satisfied. This process consumes larger time due to the migration of request until it finds a satisfying VM. A temporal load-based resource allocation in the system was proposed [20]. Column generation (CG) method was used to solve optimization problem. In this work, power consumption of VM plays a major role based on which the resources were allotted for processing.

An effective approach for VM allocation was proposed to maximize the energy efficiency of the DCs [21]. An evolutionary algorithm was used to for the purpose of VM allocation. The algorithms used are first fit heuristic (FF) and modified best fit decreasing heuristic (MBFD) approach. A simulation engine was used to accelerate the exploration of optimal VM allocation. VM-to-PM mapping was involved for allocating VMs. This work was resulted with poor reliability due to the un-optimized energy utilization. The provisioning of resource was developed using residue-based approach [22]. Here, each user with valid request should have minimum quantities of essential utilities. The residue placer element was used which acts as a combination of two placers i.e., basic and equi placer. The basic placer was used to perform one-time query processing and equi placer was used for horizontal scaling across the clouds. This was enabled to achieve low transaction time for resource satisfaction and high transaction success rate. The participation of multiple entities into resource allocation makes the system more complex. The authors in [33] emphasize on a meta-heuristic, multi-objective approach for dynamic VM allocation. The approach has been worked with Google cluster traces and resulted in optimal VM placements with better accuracy and diversity.

Resource allocation was also concentrated with the satisfaction of SLA constraints of the system. A reinforcement learning method was applied to the dynamic environment for interactions [23]. The designed framework was modeled to achieve energy-efficient resource allocation. In this, the cloud services are distributed and the satisfying cloud users are allotted with resources in the basis of energy-saving manner. The major SLA metrics that are taken in account for process are CPU resources, amount of money, permanent storage, system availability and system performance. The appointed energy-efficient resource allocator was employed to select the optimal host by which it allocates VMs for the user request. Another entity of SLA manager was present to track the requirements of user. The power usage effectiveness and data center infrastructure efficiency were calculated. The resource allocation was handled

by reinforcement learning and fuzzy logic method. According to the reward function in reinforcement learning mechanism the fuzzy rules are mapped for VM allocation with satisfied resources. In this work, the metrics that are taken in account are not sufficient which reflects on degradation in the performance.

Resource allocation was also handled by clustering and load balancing [24]. Then, task scheduling in cloud was performed for appropriate allocation of the resources and to deliver user faster responses. An improved PSO (IPSO) algorithm was proposed to increase the performance when a large number of tasks in process [25]. The incoming tasks were collected in a queue and their attributes are estimated. Adaptive splitting procedure was applied to collect the tasks in the batches. Load was balanced by moving the task from heaviest loaded VM with maximum completion time to lightest loaded VM with minimum completion time. Execution time-based classification may lead to cause frequent balancing of load. Backpropagation (BP) neural network was used in a hybrid cloud to ensure processing of all the tasks can within the specific deadline [26]. The jobs were submitted into a hybrid cloud and then allocated to the private cloud. If the private cloud was not able to meet the demands of the user, then the jobs were given to the public cloud. Jobs were scheduled based on I/O intensive and CPU intensive by logistic regression method. Optimal choices of job queues were obtained by genetic algorithm (GA). BP neural network and GA consumes larger processing time.

The task execution within the deadline was the major aim of task scheduling. Scheduling with parallelism awareness (SPA) method was proposed [27]. Tasks were assigned to be processed with the earliest deadline first on the server. Here the tasks are scheduled based on first come first serve method. This SPA method rejects a task, in case of absence of server and hence the task completion within the deadline is tedious. A federated cloud processing of tasks requires being appropriate in which all the tasks are requested to be completed within the deadline. For achieving this requirement, scheduling and resource allocation are performed. As discussed above, the major demerits are overwhelmed in the proposed work.

3. Defined problem

In this section, the proposed work algorithms are the major problem that is identified from previous research work is addressed. Clustering is defined as a solution in federated cloud for load balancing, resource allocation and reduces processing time. VM clustering and optimal sequencing algorithm was in federated cloud environment that estimates MIPS and bandwidth [28]. Then federation was applied on DCs in concerned to cost and MIPS of the DC. Later the VM clustering was handled based on VM capacity. From the utilized capacity of the VM, load was determined. In this work, clustering was fixed, but the loads at VMs were dynamic and hence, this clustering was not suitable. In [29] mean shift clustering algorithm for resource allocation and dominant sequence clustering for task scheduling. Tasks were scheduled based on the priority and clusters were constructed with the determined CPU speed, memory and I/O capacity. The use of mean shift clustering has the conventional demerit of higher time consumption for clustering.

DC clustering was performed to mitigate latency, in which k-means algorithm was used [30]. End-to-end latency was determined and then clusters were formed. Generally, in k-means clustering the prediction of k-value is complex and if the k-value is chosen random, then the clustering results performance degradation. Both task scheduling and resource allocation was developed in [31]. A contract-based resource sharing model was developed. The job requests were queued before processing into cloud. The jobs were scheduled in first-come-first out method in which the waiting time for shorter jobs will be longer. The centralized control for resource allocation causes single-point failure in the system. On taking in account of these problems that exist in scheduling, resource allocation and load balancing, the proposed work incorporates appropriate solution to mitigate consumption of excess amount of processing time.

4. Proposed system

This section is categorized into four sub-sections that details with the proposed solutions and their work procedure. Hereby, this process is entirely concentrated to complete the submitted task within the specified deadline. This improves execution time and response time. The proposed system consists of users, task schedulers, brokers and federated cloud groups. Users submit tasks into the scheduler and then broker map to resources, later the allotted tasks are processed in the federated cloud environment. Each entity plays a significant role that reflects on minimizing execution time and response time while processing larger number of tasks. Separate entities appointed for their processing also reflect to minimize complexity in the system. The distributed federated cloud environment is supported for larger number of incoming users.

4.1 Paper organization

The proposed system in federated cloud is designed to map resources dynamically and schedule the incoming tasks. The increase in number of user participation reflects in the growth of number of arrival tasks. This system is designed with N number of tasks as $T_{tasks} = T_1, T_2, \dots\dots, T_N$ to be processed in the federated cloud environment. Let the federated cloud system be composed of n number of DCs that is denoted as $DC_{fc} = \{DC_1, DC_2, \dots\dots, DC_n\}$, the DCs in this federated cloud is represented as DC_{fc} which is clustered into groups. Each DC in this cloud environment consists of $VM_{DC_n} = \{VM_l, VM_m, \dots\dots, VM_k\}$, here $\{l, m, k\}$ denotes the number of VMs that are present in individual DC in the federated cloud. In this federated cloud design multiple brokers are employed to solve single point failure.

The federated cloud environment manages clustered DCs and VMs. The DCs are clustered using Region-Based Fuzzy Possibilistic C-Means Clustering Algorithm and the VMs are clustered by multi-objective density-based spatial clustering. Based on the clustered VMs, the incoming tasks are allocated for processing with respect to the mapped resources. Meanwhile the loads at VMs are balanced by Markov chain method. On the other hand, the incoming tasks are scheduled by entropy based monotonic scheduling algorithm which is operated based on the obtained resource information. Then, as per the mapped resources, the tasks are allotted into VMs for processing. Fig. 2 illustrates the complete process performed in the proposed system. In this architecture the terms $C1, C2, C3, \dots$ denotes the clusters that are formed from DCs and VMs.

4.2 DC clustering

The DCs in the federated cloud environment is clustered using region-based fuzzy possibilistic C-means clustering algorithm. The federated clouds are nothing but a composition of smaller clouds and so they are spread distributed. Due to its widespread nature, the initial step in this clustering is to identify the regions of DC. After predicting their regions, the clustering parameters that are taken into consideration are data dependency, MIPS, latency, storage, bandwidth and number of VMs.

The proposed clustering is a combination of possibilistic c-means and fuzzy c-means algorithm. In this proposed clustering algorithm, the membership grades and possibilities play an important role. Hereby the steps followed in the proposed possibilistic fuzzy c-means algorithm are descripted in the following,

Step 1: Begin with the initialization of the number of DC clusters that is represented as c which need to satisfy $1 \leq c \leq n$. Then initialize partition matrix as $U \in M_{fc}$, the term M_{fc}denotes the set obtained from fuzzy partitioning space Z. Determine region in which the particular DC is located in the designed federated cloud.

Step 2: Compute cluster prototype i.e., mean based on the individual constraints of DC as data dependency, MIPS, latency, storage, bandwidth and number of VMs. The mathematical formulation of cluster prototype is given as,

$$v_i = \frac{\sum_1^n (a\mu_{ik}^m + bt_{ik}^\eta)^m Z_k}{\sum_1^n (a\mu_{ik}^m + bt_{ik}^\eta)^m}, 1 \leq i \leq c \qquad (1)$$

The weighted mean value v_i is given based on the constant values as $0 \leq \mu_{ik}, t_{ik} \leq 1$, μ_{ik}, t_{ik} are the typicality constraint row and column sum and the values $a > 0$, $b > 0$, $m > 1$, $\eta > 1$ are pre-defined. Z_k is the row vector that is given from the matrix.

Step 3: The distance is measured between a new DC and the identified center. Distance is determined using Euclidean formula.

Step 4: Update partition matrix from the estimated distance value and also update typicality matrix.

Step 5: Repeat steps until clusters are constructed, iterations are performed up to the tolerance value ε is reached.

The proposed DC clustering algorithm takes in account of the following measures using which clusters are constructed. The parameters that are involved are,

- Data dependency – The data dependency is denoted as D_D
- MIPS – This parameter is significant from which the processing speed is identified. The faster processing of tasks is attained by the key of MIPS.
- Latency – Latency L_c is defined as the time delay that the resources consume to complete a particular task.
- Storage – Storage S_g denotes the space to maintain data for processing. An insufficient storage in DC also fails to operate tasks.
- Bandwidth – Bandwidth B_w defines the amount of bandwidth that are supported at particular DC using which the incoming tasks are performed by VMs.
- Number of VMs – Each DC is composed of VMs that are responsible for processing the tasks. The

number of VMs that are present in an individual DC is represented as U_{VMs}.

The objective function of this proposed DC clustering is given based on all the above-mentioned constraints. The representation of objective function is given as follows,

$$Max\{D_D, MIPS, S_g, B_w, U_{VMs}\}, Min\{L_c\} \quad (2)$$

Region-based fuzzy possibilistic C-means clustering algorithm is proposed and the clusters are created based on the developed objective function. The DCs participating in the federated cloud are clustered only once and here.

4.3 VM clustering

VM clustering is performed by multi-objective density based spatial clustering algorithm. The clustering of VMs is presented by the estimation of capacity and bandwidth of individual VM. The capacity of each VM is determined as follows,

$$VM_{capa} = N_{PE} + MIPS + B_w \quad (3)$$

The VM capacity VM_{capa} is computed from number of processing elements that is represented as N_{PE}, B_w that denotes the bandwidth utilized by VM. On behalf of the VM_{capa} and B_w, the VMs are clustered in federated cloud environment. These two measurements are applied into density based spatial clustering algorithm for obtaining the objective using the two constraints.

The potentialities of this proposed clustering algorithm are,

The clusters shapes are not essential to be in geometry.

The construction of only specified number of clusters is not required.

This clustering algorithm requires only $O(n)$ memory for processing and $O(n \log n)$ for run time.

In this proposed system, multi-objective density based spatial clustering is handled on the basis of neighborhood point denoted as ε i.e., eps and minimum points as $minPts$. The ε defines data point that are present around the particular point. The points that are closer to particular point in distance is considered to be neighbor. As per the selected ε value the clusters will be split are merged into one. This value is determined using distance measure which is given based on the similarity that is determined from VM capacity and bandwidth of the VMs. Distance equation used in this work for identifying the neighborhood is given as,

$$D = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2} \quad (4)$$

Let (p_1, p_2) and (q_1, q_2) be the two individual points of VMs respectively. Then the value of $minPts$ has to be selected based on the number of VMs that are participating in the federated cloud. If the total number of VMs are in large number, then certainly the $minPts$ is selected larger. As per this work, the numbers of VMs are larger.

The following steps are performed to construct VMs into clusters using multi-objective density based spatial clustering,

Step 1: Divide the VMs data points into K dimension

Step 2: Estimate neighbor points ε and predict core points

Step 3: If the core point is not present in a cluster, then create a new cluster with that core point

Step 4: After identifying a new cluster point, the other connected points are determined and then added into newly formed cluster

Step 5: Perform iteration until all the points in the environment are visited.

Hereby, based on the objective of VM capacity and bandwidth the VMs are clustered. The core point in this work is defined as a point which has least number of $minPts$ exists within the ε. Based on the clustered VMs the resources utilization is monitored and updated by brokers. The clustered VMs are required to be balanced in load, so that the frequent clustering could be reduced. For this, Markov chain is incorporate to balance load among clusters based on their execution. The proposed Markov chain model is associated with three states as under load, normal load and overload. Let S be the possible states that whose transition probability is represented as (i, j) i.e., i and j denotes two different states respectively. The initial state vector is given as $S \times 1$ matrix.

According to the predicted VM load, the load balancing is handled. From the transition matrix the changes of states from one to another is determined in iterations. Hereby, this k-step transition matric is estimated by performing matrix multiplication. The clustered VMs are also balanced with the load which is periodically updated in broker.

4.4 Task scheduling

Task scheduling is a key process performed to enable faster response of the submitted task. Task scheduling is handled using entropy-based

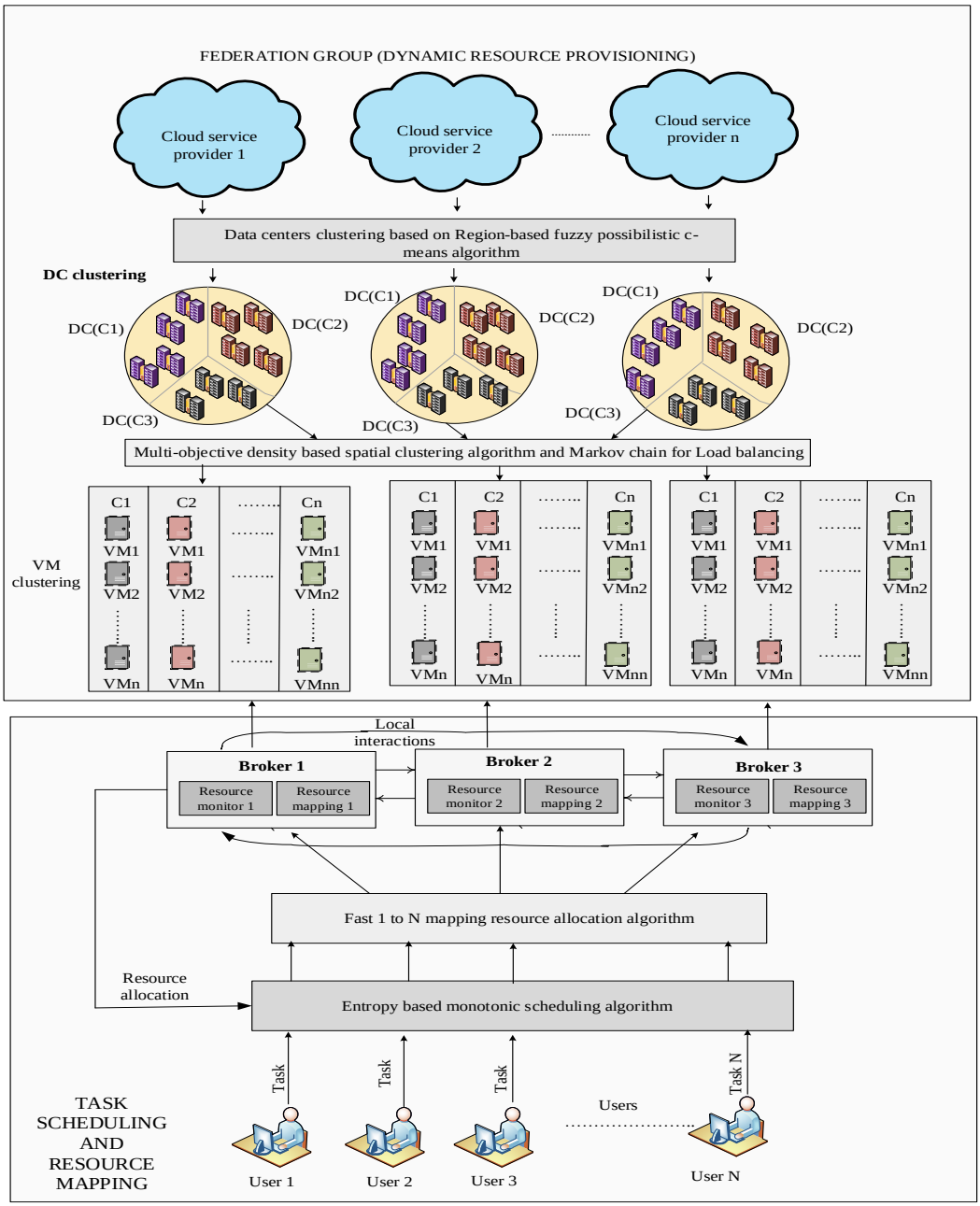

Figure. 2 Proposed federated cloud architecture

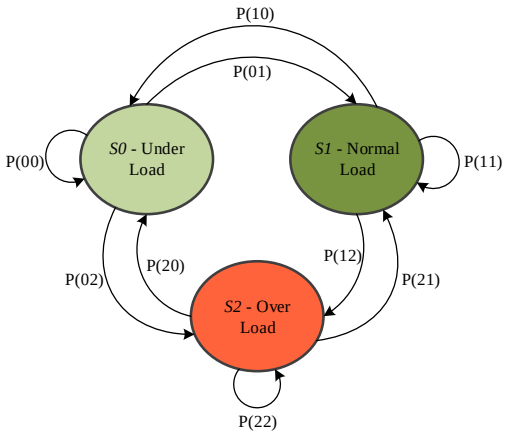

Figure. 3 Load balancing by Markov chain

monotonic scheduling algorithm. Scheduling of the tasks is based on the task type, task size, task arrival time and deadline. The threshold for scheduling is predicted dynamically using entropy function that is based on the task arrival rate. Hereby, the entropy function $E(V)$ is mathematically formulated as,

$$E(V) = -\sum_{x \in X} p(x) \log_2 p(x) \quad (5)$$

From the above equation, V is the random variable, $p(x) = \Pr\{X = x\}, x \in X$ defines the probability mass function, the minimum and maximum entropy values are 0 and $\log_2 o$ respectively, where o denotes the number of outcomes. This monotonic scheduling algorithm is operated with the provisioning of priority for tasks based on task type t_p, task size t_s, task arrival time t_a and task deadline t_d. The utilization bound (UB) test is given as,

$$UB_M = N[2^{(1/N)} - 1] \quad (6)$$

N denotes the number of tasks that are prioritized based on the task constraints. The tasks are schedulable only if the estimated UB is not greater than UB_M. If the utilization value is lesser then the deadline of individual task is definitely met and so the time taken for scheduling is shortened. Due to dynamic threshold based on entropy, the scheduling is performed appropriately and also it is capable to assist processing within deadline.

Pseudo Code for task scheduling

1. begin
2. Submit i tasks $\{t_1, t_2, \ldots\ldots, t_i\}$
3. $t_1 \rightarrow (t_p,\ t_s, t_a,\ t_d)$
4. Assign i priority to i tasks as $\{p_1, p_2, \ldots p_i\}$
5. Compute UB_M for i tasks
6. if $(UB_m < UB)$
   {
   i tasks are schedulable
   else
   i tasks are non-schedulable
   }
7. Scheduled i tasks
8. end

A task T is submitted with the following parameters as $(t_p,\ t_s, t_a,\ t_d)$ each constraint is significant in scheduling since the processing time for each task is not similar and so these four parameters are taken in account for scheduling the tasks. The definition for each parameter is illustrated in the following,

- Task type – This denotes the type of tasks that the user requires to perform. In this work, the task type is categorized as real-time and non-real time which is based on the application preferred by the user. The real-time tasks usually have lesser deadline which is given higher priority than the non-real time tasks.
- Task size – The task size defines the size of the task.
- Task arrival time – Arrival time denotes the time during which the task is arrived at scheduler.
- Task deadline – This is one of the significant constraints in scheduling based on which the task has to be processed and the response needs to be delivered.

Once the set of tasks received at time Tm_1 is scheduled, then the next set of tasks will be scheduled. All the scheduled tasks are processed in broker for assigning a VM for processing. All the scheduled tasks are assigned to VMs at one instance.

4.5 Resource mapping

The process of resource mapping is performed with fast 1 to N resource mapping algorithm. The proposed algorithm is designed to map with the resources with respect to the SLA requirements of the tasks. The three significant constraints that are considered for processing are CPU, memory and bandwidth. The mapping of resources is applicable only when the SLA constraints of the particular task are satisfied. The participation of many numbers of VMs will certainly satisfy the SLA requirements. This fast 1 to N resource mapping is operated on multiple brokers who are already equipped with the information of VM utilization and the availability of resources in individual VM. On knowing this updated

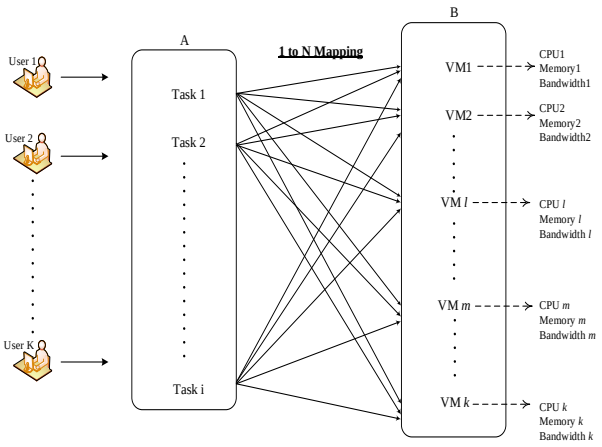

Figure. 4 Fast 1 to N resource mapping

information, the mapping of all the resource constraints is performed. Mapping is a simpler process that is functioned by matching the SLA requirements of the tasks with respect to the SLA constraints that currently exists in the VMs. According to this mapping the scheduled tasks are allotted to VM by the brokers for processing.

In this proposed fast 1 to N resource mapping algorithm, the individual tasks are mapped to the corresponding VMs that matches with SLA requirements. As shown in the figure, assume A as total tasks and B as total number of VM that are clustering in the federated cloud. Let the 1 to N resource mapping function be,

$$f: A \rightarrow B; \forall i \in A, \forall k \epsilon B, f(a) \rightarrow b_x \qquad (7)$$

The $f(a)$ maps with the suitable b_x i.e., the VM which satisfies the SLA constraints of that particular task. The mapped VM will be selected to operate the task. Similarly, each task is mapped with the appropriate VM and then the requested process is performed. The parallel mapping of resources for each task simultaneously leads to faster allocation of tasks into VMs.

5. Result evaluation

In this section, the proposed system is evaluated by implementing in a tool with the proposed methods. This section is composed of implementation setup, comparative results and discussion of achievements.

5.1 Implementation environment

Federated cloud environment is incorporated with task scheduling, resource mapping and clustering for faster resource allocation that improves execution time and response time. The proposed task scheduling and resource mapping in federated cloud environment is implemented in Cloudsim 3.0 framework that offers flexibility on building multiple concepts in cloud environment.

Cloudsim is an efficient simulation tool which is presented to evaluate the performances of proposed operating mechanisms. This is similar to the real-world deployment and so Cloudsim is preferred for designing federated cloud environment. Cloudsim is a toolkit that is programmed using JDK 1.8 installed in NetBeans 8.2. Cloudsim is supported in Windows 7 operating system.

The significant requirements that are essential for implementing this work include hardware elements as well as simulation parameters.

Table 1. Hardware specifications

Element	Range
RAM	2.00 GB
Processor	Pentium
Speed	3.00 GHz
System type	32-bit
Operating system	Windows 7 (X86 ultimate)

Table 2. Cloudsim specifications

Parameter		Value
Number of Users		5 – 10
Number of DCs		10 – 30
Number of VMs in each DC		12 – 15
Number of Tasks		25 and above
Number of Brokers		3 – 4
VM	Bandwidth	500 – 1000 kbps
	Memory	128 – 2048 GB
	MIPS	9600
Task length		1500 – 3000
Storage capacity		11 TB
Scheduling interval		30 ms
Monitoring interval		180 ms

Table 1 and 2 depicts the major hardware specification and simulation specification respectively. Cloudsim is enabled to develop simulation of federated clouds with the participation of large scale DCs, VMs and brokers. The flexibility of Cloudsim has enabled to develop new problem-solving solutions in the proposed system. Hereby, the Cloudsim toolkit is presented with the development of DC clustering using region based fuzzy possibilistic c-means clustering, VM clustering by multi-objective density based spatial clustering, resource mapping by fast 1 to N mapping and entropy based monotonic scheduling algorithm. Also, the loads at VMs are balanced by the involvement of Markov chain method.

Fig. 6 depicts the cloud reports that are obtained for the developed system. As per the number of tasks the process is begun for each task and then the successful completion time of each task is resulted on cloud log reports. These reports are generated based on the efficiency of the proposed methods as mentioned above. Each method is appointed for performing its own process and finally the finish time shows the time at which the particular task is completely processed.

The major entities that are involved in the proposed system design are,

- Users

Users are the initial participants those who submit tasks into the cloud for processing. Each user's task differs in task type, size, deadline and others. The

users in this system are supposed to submit the task and wait for the response.

- Task Scheduler

Scheduler is appointed in this system to schedule the incoming tasks based on the task information. The task information includes task type, task size, task deadline. The proposed algorithm is applied into task scheduler for scheduling the arrived set of tasks in the cloud.

- Broker

All the scheduled tasks enter into broker entity for processing in a VM. In this proposed work more than one broker is involved to avoid single point failure problem. Here the brokers exchange with the resource information via local interactions. Also, the broker communicates with the cloud environment and updates the current resource information.

- Federated Cloud

This is composed of DCs and VMs in the clustered form. Using algorithms, the DCs are clustered static and the VMs are dynamically clustered based on their varying capacity.

As per the specified specifications the federated cloud environment is developed with the proposed algorithms for scheduling and allocating resources in VMs. Individual entity in the designed federated cloud environment has their own working functionalities in the system.

5.2 Comparative results

In this section the results of proposed system are compared with previous works to determine the efficiencies of the proposed federated cloud. The key objective of processing the task within deadline is enabled with the processes of DC clustering, VM clustering, resource mapping, load balancing and task scheduling. For comparison, the parameters that are computed as, execution time, latency, resource utilization, and response time. Each parameter in this system is taken in account of evaluating the processing efficiency of the proposed system.

Hereby, the disadvantages that existed in previous research works are illustrated in Table 3. On behalf of the methods, algorithms and constraints that are used in previous work the demerits were existed. So, these demerits are limited in the proposed work by designing appropriate solution of clustering, load balancing, resource allocation and task scheduling. The efficiency in the result is attained only due to the perfect development of proposed algorithms. Each method in the proposed work reflects on better achievements in the proposed work. To be noted, in this work, DC clustering is performed only once, whereas the VMs are clustered dynamic in accordance to their time varying property. The performances of the proposed federated cloud environment are studied.

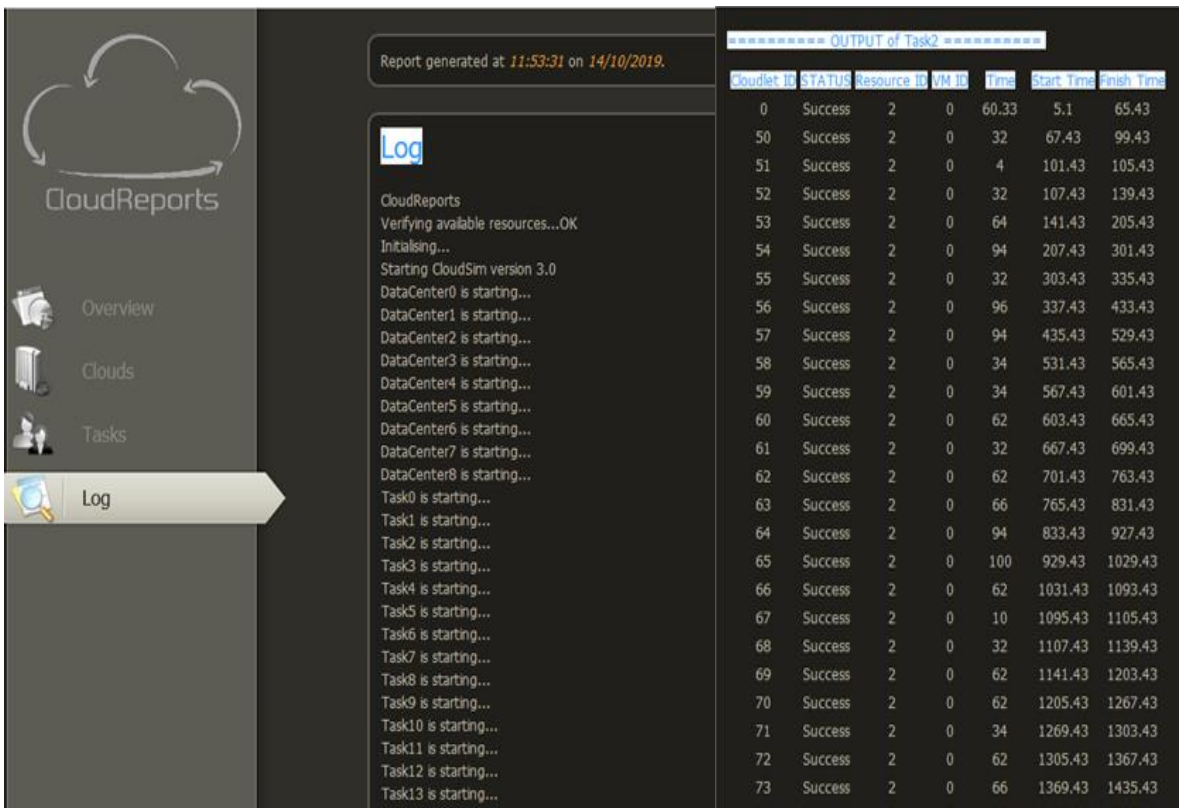

Figure. 6 Cloudsim reports

Table 3. Existing work and their demerits

Prior method	Demerit
Broker agent-based task processing [19]	Larger processing time, since the broker forwards request until the requirements are satisfied.
VM capacity based VM clustering [28]	Clusters are constructed initially, however the capacity of VM is dynamic based on the arrival of tasks. Hence this clustering increases load and processing time.
Mean shift clustering for VM clustering [29]	Conventionally the mean shift clustering algorithm consumes larger time.
DC clustering using k-means algorithm [30]	1. Tedious in selection of *k* value 2. Single constraint of latency-based clustering is inefficient, since the low latency DC always stays idle.
First-come first-out method-based task scheduling [31]	1. The task with larger size at the last has to wait until all the other tasks are processed. 2. Centralized entity for resource allocation cause single point failure 3. Unsatisfied SLA requirements.

5.2.1. Execution time

The time taken for processing a set of tasks is defined as execution time. Execution time is improved by designing faster processing algorithms. The proposed task scheduling and resource mapping unique allocate the tasks into VMs with the satisfaction of SLA requirements. In contrast, the existing work failed to satisfy SLA requirements and hence it increases execution time.

The clustered VMs are easier in resource management; however, the faster mapping of resource based on the task requirements ensure with the improvement in the execution time. As shown in Fig. 7, the execution time of proposed work is lesser than the existing federated cloud due to the static clustering. The VMs were clustered initially based on their capacity and based on it the incoming tasks are allocated. Due to this, a greater number of tasks may be continuously allotted into the same VMs which increases load and hence execution time also increases. High execution time also reflects to degrade on other parameters. According to increase in the number of tasks the execution time will be increases, but if there exists larger number of idle VMs then certainly the execution time is minimized. The execution time difference of about 15 – 20 ms occurs between existing and proposed.

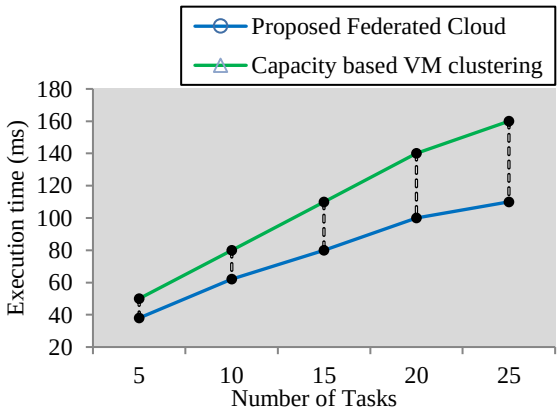

Figure. 7 Comparison on execution time

5.2.2. Latency

Latency is a delay constraint which increases in accordance to poor system design. The selection of optimal VM with respect to the resources enables to minimize latency. The reduction in latency will certainly reduce processing time of tasks. The latency in the federated cloud is estimated using the following formulations,

$$L_c = \sum_{j,k=1}^{n_{dc}} \sum_{i,l=1}^{n_{type}} \sum_{p,q=1}^{n(VM^{ij})} l(VM_p^{ij}, VM_q^{lk}) \quad (8)$$

The term VM_p^{ij} defines p^{th} VM with i^{th} instance type involved in j^{th} DC. Similarly, VM_q^{lk} denotes q^{th} VM with l^{th} instance type involved in k^{th} DC. Then, n_{dc} represents the number of DCs, n_{type} denotes number of types that the VM instances exists and $n(VM^{ij})$ defines the VM present with i^{th} instance type involved in j^{th} DC.

Fig. 8 illustrates the graphical plot of comparing latency of proposed and existing work. Latency in the proposed work is lesser than the existing due to the prompt assignment tasks for processing. Task

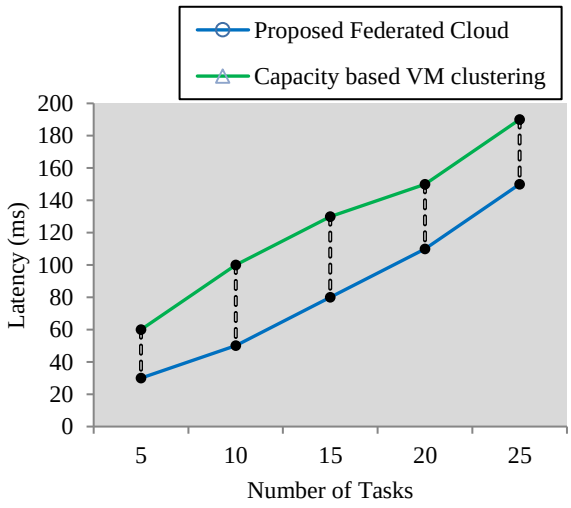

Figure. 8 Comparison on latency

Table 4. Measured mean latency

Work	Latency (ms)
Capacity based VM clustering	126
Proposed	84

scheduling is performed by considering task deadline, size and others which schedules the tasks accordingly. The proper scheduled tasks allotted with mapped VM have minimized latency. However, the latency increases with respect to the increase in number of tasks, it is tolerable in proposed system. On an average of 10 – 15% of latency is reduced in proposed than the previous research.

The average latency of proposed and existing is also varying nearly 35 – 40ms which is too large. The mean value for 25 tasks is too high in existing and so it grows higher with respect to increase in the task. Reduction in this parameter ensure with the improvement in the system performance.

5.2.3. Response time

Response time is one of the key parameters that is measured to evaluate the performance of the system in terms of time constraint. Shorter the response time ensures to improve performance and intimates that the processing of the tasks is attained within the deadline. Response time of the system is defined as the time taken for the cloud to process the request and response particular task. Response time in federated cloud environment is computed as per the following,

$$R_T = T_{Cp} - t_a + T_t \quad (9)$$

The response time R_T is determined from T_{Cp}, t_a and T_t that denotes time needed to complete the task, time at which the task is arrived and transfer time for the task respectively. In proposed system, the response time is reduced by the following,

- Tasks are scheduled using the constraints of task size, task type and deadline by which they are arranged appropriately in the schedule.
- Broker maps with the SLA requirement satisfying VM faster by mapping function.

Parallel mapping for each task in accordance to SLA requirements enables to satisfy SLA requirements as well as processes the task in most suitable VM.

The response time of the system is compared with previous research work and this plot is shown in terms of number of VMs and number of tasks as in Fig. 9. The response time curve in proposed system is increased and decreased where the curve in existing

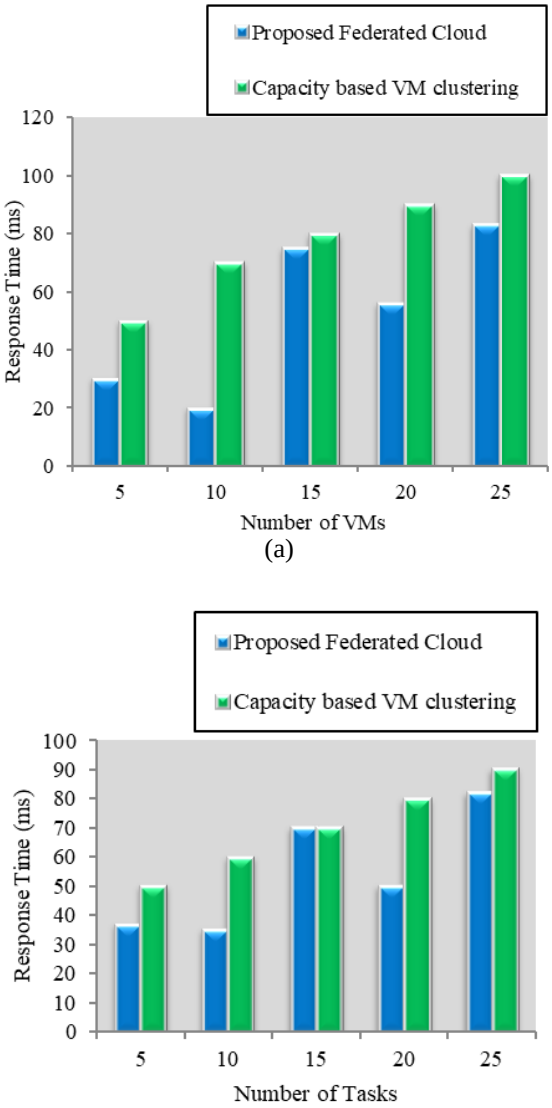

Figure. 9 Comparison on response time with respect to (a) Number of VMs and (b) Number of tasks

is gradually increasing. The immediate increase and decrease in the response time are due to the task type that is introduced for processing. However, the response time has ups and downs with respect to the number of VMs and number of tasks, it is higher in existing. The major response for increase in response time in previous work is,

- Static clustering was employed in which cluster were created based on the VM capacity, whereas the proposed federated cloud constructs dynamic VM clustering.
- Due to static clustering with timely varying capacity, many numbers of tasks could be satisfied by a particular VM and hence it will be chosen for processing. Continuous selection of particular VM for processing will certainly increases load at VM and it increases response time.

Table 5. Measured mean response time

Work	Response time (ms)	
	# of VMs	# of tasks
Capacity based VM clustering	78	70
Proposed	53	55

Hereby, Table 5 illustrates the estimated mean response time with respect to the number of tasks and VMs. An approximate difference of 20 – 25 ms is higher in the previous work. In this time the proposed work algorithms are capable to process nearly 5 – 10 tasks. Hence the proposed federated cloud environment results with better response time than the previously existing capacity-based clustering method. The reduction in response time for multiple tasks ensures that all the tasks are satisfied in processing. The tasks are processed at faster rate, since they are assigned to an SLA satisfied VM.

5.2.4. Resource utilization

Resource utilization is a common parameter that is computed for evaluating the efficiency of VM resource. In federated cloud VM resources play a vital role in handling incoming tasks. The effective utilization of resource is achieved only by proper load balancing and task assignment to apt VM. The utilization of resource is significant in evaluating the performance of federated cloud. A federated cloud is deployed with sufficient number of resources; however, it is efficient only when they are utilized properly. The utilization of resources is computed using the following expression as,

$$RU = \frac{T_{Cp}}{Mk_{sp} \times R} \tag{10}$$

The utilized amount of resource RU is determined from Mk_{sp} that denotes makespan of the set of tasks that are operated in the VM and R denotes the number of resource present in VM.

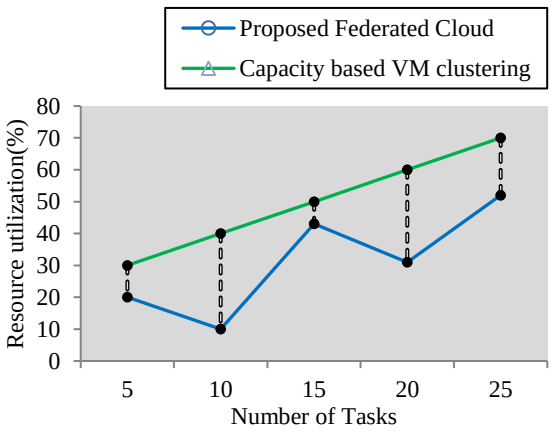

Figure. 10 Comparison on resource utilization

Table 6. Successful task processing

Number of Tasks	Number of tasks processed within deadline	
	Capacity based VM clustering	Proposed
10	8	10
20	15	19
30	23	28

Resource utilization for proposed and previous VM capacity-based clustering is compared as shown in Fig. 10. The utilization of required number of resources enables to assist improvement in system performance. On an average 30% and 50% of resources are utilized in proposed and existing respectively. This result is obtained while processing with same number of tasks with same type. Hereby nearly 70% of resources were save in proposed, in contrast equal half of the resource were utilized and save in VM capacity-based clustering.

Hereby, the increase in resource utilization will certainly degrades the performance and also it becomes tedious to support for large scale environment. Therefore, the proposed system with faster allocation of VM based on the satisfied SLA requirements ensures to improve resource utilization even for larger number of tasks. Also, the number of incoming tasks is successfully processed in the proposed work which is depicted in Table 6. According to the number of incoming tasks, they are processed in cloud and it is required to be completed within the deadline of the task. The tasks that fail to finish the particular task within deadline leads to poor system design.

From Table 6, the served number of tasks within the deadline is depicted. As per the increase in number of tasks into the cloud, it is essential to be allocated into efficient processing VM and so the task will be completed within deadline. In VM capacity-based clustering; multiple tasks may prefer a particular VM due to which it increases load and response time. This eventually fails to complete the task within deadline whereas the appropriate allocation of VM in proposed is capable to complete the task within deadline.

5.3 Result discussion

The main goal of this proposed system is to complete the task processing within the deadline. In order to achieve this objective, a system is developed on federated cloud. This federated cloud performs DC clustering, VM clustering and load balancing.

Table 7. Processes involved in proposed federated cloud

Process	Method used	Metrics estimated
DC clustering	Region-Based Fuzzy Possibilistic C-Means Clustering Algorithm	1.Data dependency 2.MIPS 3.Latency 4.Storage 5.Bandwidth 6.Number of VMs
VM clustering	Multi-objective density-based spatial clustering	1.Capacity 2.Bandwidth
Load balancing	Markov chain	VM capacity
Resource mapping	Fast 1 to N Resource Mapping Algorithm	SLA Constraints 1.CPU 2.Memory 3.Bandwidth
Task scheduling	Entropy-based monotonic scheduling algorithm	1.Task type 2.Task size 3.Task arrival time 4.Deadline

Clustering is involved for efficient resource allocation for the incoming tasks and based on which the time taken for processing is not extended. Similarly, the deployment of multiple brokers in this system avoids the problem of single point failure in the system. The tasks received from the users are scheduled and then each task is mapped with appropriate VM for processing. Hereby, the mapping of tasks with respect to the VM resources, result with appropriate VM for task processing.

On discussing the results of proposed work, the reason for the achievement is depicted in Table 7. The metrics that are taken in account for each processing reflects on the improvement in the proposed system. Due to these processing in federated cloud, the incoming tasks are allotted to appropriate VM and reduce response time.

6. Conclusion

The objective of this paper is to complete the task within deadline and improve response time, execution time and resource utilization. The proposed federated cloud environment is designed with the processes of DC clustering, VM clustering, resource mapping and task scheduling. The clustering processes are handled in federated cloud, resource mapping by brokers and scheduling by the task scheduler. Initially, the federated cloud maintains DC clusters and VM clusters that are created using region based fuzzy possibilistic c-means clustering and multi-objective density based spatial clustering respectively. The DC clusters are static and the VMs clusters are frequently upgraded due to the changes in their capacity. Consequently, load at VMs is balanced using Markov chain model. The incoming tasks are scheduled with entropy based monotonic scheduling algorithm which is performed by considering task information. After scheduling, the tasks are allotted into VM in accordance to the satisfied SLA requirements. Brokers are incorporate with fast 1 to N resource mapping algorithm for identifying a suitable VM to process the task. Herby, the appropriate assignment of VM for processing tends to reduce execution time and improve response time.

In future work we have planned to concentrate more on VM load balancing by performing VM migration and VM swapping for analyzing the utilization of resources.

Conflicts of Interest

The authors Jeny Varghese and Dr. Jagannatha Sreenivasaiah declare no conflicts of interest.

Author Contributions

Jeny Varghese and Dr. Jagannatha Sreenivasaiah contributed to the design and implementation of the research, to the analysis of the results and to the writing of the manuscript.

References

[1] J. Sun, Y. Zhang, Z. Wu, Y. Zhu, Z. Ding, Z. Wei, J. Plaza, and A. Plaza, “An Efficient and Scalable Framework for Processing Remotely Sensed Big data in Cloud Computing Environments”, *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 57, No. 7, pp. 4294-4308, 2019.

[2] L. Chunlin, T. Jianhand, and L. Youlong, “Distributed QoS-aware scheduling optimization for resource-intensive mobile application in hybrid cloud”, *Cluster Computing*, *Springer*, Vol. 21, No. 2, pp. 1331-1348, 2018.

[3] S. Suprakash and S. P. Balakannan, “Service Level Agreement Based Catalogue Management and Resource Provisioning in cloud for optimal Resource Utilization”, *Mobile Networks and Applications*, pp. 1-9, 2019.

[4] S. Mustafa, K. Bilal, S. U. R. Malik, and S. A. Madani, “SLA-Aware Energy Efficient Resource Management for Cloud Environments”, *Emerging Trends, Issues, and Challenges in Energy-Efficient Cloud Computing*, *IEEE Access*, Vol. 6, pp. 15004-15020, 2018.

[5] N. Chitgar, H. Jazayeriy, and M. Rabiei, "Improving Cloud Computing Performance Using Task Scheduling Method Based on VMs Grouping", In: *Proc. of 27th Iranian Conference on Electrical Engineering*, 2019.

[6] Y. Cui and Z. Xiaoqing, "Workflow tasks scheduling optimization based on genetic algorithm in clouds", In: *Proc. of IEEE 3rd International Conference on Cloud Computing and Big Data Analysis*, 2018.

[7] B. Anushree and V. M. A. Xavier, "Comparative Analysis of Latest Task Scheduling Techniques in Cloud Computing environment", In: *Proc. of Second International Conference on Computing Methodologies and Communication*, 2018.

[8] C. S. Rajarajeswari, "Challenges in Federated Cloud", *International Journal of Current Engineering and Scientific Research*, Vol. 4, No. 10, pp. 55-59, 2017.

[9] G. M. Bhatu and K. S. Subhash, "Task scheduling and resource allocation in cloud computing using a heuristic approach", *Journal of Cloud Computing: Advances, Systems and Applications*, 2018.

[10] A. H. Maryam, M. Mehrdad, and H. Majid, "An Energy-Efficient Dynamic Resource Management Approach Based on Clustering and Meta-Heuristic Algorithms in Cloud Computing IaaS Platforms", *Wireless Personal Communications*, Vol. 104, No. 4, pp. 1367-1391, 2019.

[11] M. Kun, B. Antoine, M. Hope, and C. Antonio, "Modelling cloud Federation: A Fair Profit Distribution Strategy Using the Shapley Value", In: *Proc. of IEEE 6th International Conference on Future Internet of Things and Cloud*, 2018.

[12] S. Kim, "Dual-Level Cooperative Game Approach for Energy-Aware Resource Allocation in Data Centers", *IEEE Access*, Vol. 7, pp. 113642-113652, 2019.

[13] N. K. Sharma and G. R. M. Reddy, "Multi-Objective Energy Efficient Virtual Machines Allocation at the Cloud Data Center", *IEEE Transactions on Services Computing*, Vol. 12, No. 1, pp. 158-171, 2019.

[14] K. Spiros, M. Paul, Z. Huan, H. Yang, W. Junchao, C. Thierry, G. Baptiste, H. Jani, D. L. Cees, and Z. Zhiming, "Time-critical data management in clouds: challenges and a Dynamic Real-Time Infrastructure Planner (DRIP) Solution", *Concurrency and Computation Practice and Experience*, 2019.

[15] S. Sobhanayak, A. Turuk, and B. Sahoo, "Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm", *Future Computing and Informatics Journal*, Vol. 3, No. 2, pp. 210-230, 2018.

[16] N. Jargalsaikhan, H. Taejin, B. Jaewon, and L. Hyuk, "Dependency Analysis based Approach for Virtual Machine Placement in Software-Defined Data Center", *Applied Sciences*, 2019.

[17] Y. Z. Pei and Z. MengChu, "Dynamic Cloud Task Scheduling Based on Two-Stage Strategy", *IEEE Transactions on Automation Science and Engineering*, Vol. 15, No. 2, pp. 772 - 783, 2018.

[18] P. Elina, L. Lucas, M. Cristian, and G. G. Carlos, "A Bio-inspired Datacenter Selection Scheduler for Federated Clouds and Its Application to Frost Prediction", *Journal of Network and Systems Management*, Vol. 27, No. 3, pp. 688 - 729, 2019.

[19] K. Sofiane, Z. Abdelhafid, and D. Mahieddine "AMACE: agent-based multi-criterions adaptation in cloud environment", *Human-centric Computing and Information Sciences*, 2018.

[20] V. Shahin, "Energy efficient temporal load aware resource allocation in cloud computing datacenters", *Journal of Cloud Computing*, 2018.

[21] Z. Xinqian, W. Tingming, C. Mingsong, W. Tongquan, Z. Junlong, H. Shiyan, and B. Rajkumar, "Energy-aware virtual machine allocation for cloud with resource reservation", *Journal of Systems and Software*, Vol. 147, pp. 147-161, 2019.

[22] S. Kirthica and RajeswariSridhar, "A residue-based approach for resource provisioning by horizontal scaling across heterogeneous clouds", *International Journal of Approximate Reasoning*, Vol. 101, pp. 88-106, 2018.

[23] T. Thandar, M. M. Myint, P. Sazia, and G. Amjad, "Reinforcement Learning based Methodology for Energy-efficient Resource Allocation in Cloud Data Centers", *Journal of King Saud University - Computer and Information Sciences*, 2018.

[24] R. K. Devi and G. Murugaboopathi, "An efficient clustering and load balancing of distributed cloud data centers using graph theory", *International Journal of Communication System*s, Vol. 32, No. 5, 2019.

[25] S. Heba, N. Heba, S. Walaa, and H. Hany, "IPSO Task Scheduling Algorithm for Large Scale Data in Cloud Computing Environment", *IEEE Access*, 2018.

[26] C. Li, J. Tang, and Y. Luo, "Hybrid Cloud Adaptive Scheduling Strategy for Heterogeneous Workloads", *Journal of Grid Computing*, Vol. 17, pp. 419-446, 2019.

[27] W. Bo, S. Ying, C. Jie, C. Xiao, and Z. Ling, "Improving task scheduling with parallelism

awareness in heterogeneous computational environments", *Future Generation Computer Systems*, Vol. 94, pp. 419-429, 2019.

[28] S. K. Sonkar and M. U. Kharat, "Load prediction analysis based on virtual machine execution time using optimal sequencing algorithm in cloud federated environment", *International Journal of Information Technology*, Vol. 11, pp. 265-275, 2019.

[29] A. Amer, A. Saleh, A. Abdullah, and A. Muder, "Novel Approach to Task Scheduling and Load Balancing Using the Dominant Sequence Clustering and Mean Shift Clustering Algorithms", *Future Internet*, Vol. 11, p. 109, 2019.

[30] W. Jie, Z. Ao, Y. Jie, and Y. Fangchun, "AIMING: Resource Allocation with Latency Awareness for Federated-Cloud Applications", *Wireless Communications and Mobile Computing*, Vol. 2018, pp. 1-11, 2018.

[31] X. Jinlai and P.Balaji, "Optimized Contract-based Model for Resource Allocation in Federated Geo-distributed Clouds", *IEEE Transactions on Services Computing*, pp. 1-1, 2018.

[32] C. Jiashun, Z. Hao, P. Dechang, K. Mehmed, Q. Yin, and L. Xin, "A Weight Possibilistic Fuzzy C-Means Clustering Algorithm", *Hindawi Scientific Programming*, Vol. 2021, p. 10, 2021.

[33] T. Ennio, J. D. Juan, D. M. Vincenzo, A. Prateek, B. Shajulin, S. Nishant, and P. Radu, "A dynamic evolutionary multi-objective virtual machine placement heuristic for cloud data centers", *Information and Software Technology*, Vol. 128, 2020.