ISSN 1816-6075 (Print), 1818-0523 (Online)
Journal of System and Management Sciences
Vol. 11 (2021) No. 2, pp. 29-47
DOI:10.33168/JSMS.2021.0203

Minimizing Makespan Time in Cloud Computing using Heuristic Elasticity based Dynamic Task Scheduling Algorithms

Geeta Singh[1], Shiva Prakash[2], Santosh Kumar[3]

[1] Department of Computer Science and Engineering, Dr. APJ Abdul Kalam Technical University, Lucknow, India

[2] Department of Computer Science and Engineering, Madan Mohan Malaviya University of Technology, Gorakhpur, India

[3] Department of Computer Science and Engineering, ABES Engineering College Ghaziabad, Uttar Pradesh, India

geetasingh02@gmail.com, shiva_pkec@yahoo.com, santoshg25@gmail.com

Abstract. Cloud computing technique is rapidly spreading in all over the world due to its pay per usage, anywhere any time property. Most of the IT [23] firms and business are using this technique with the help of web based application and tools over the internet As a result of growing large number of users, cloud service providers and applications in the cloud is growing quite rapidly due to the problem of preserving QoS parameters such as throughput, reliability, availability, elasticity and makespan time, etc. Here, with the aid of ERPD (elastic resource provisioning / de-provisioning) and scalability mechanism, we have proposed, built and developed an algorithm to balance the load dynamically between virtual machines for optimizing resource usage and minimizing the make-pan time in the cloud environment. The experimental results compared to the SJF, FCFS algorithm, and the Min-Min algorithm and this new proposed algorithm showed better performance.

Keywords: Makespan time, virtual machine, task scheduling, elasticity.

1. Introduction

The cloud, Big data, artificial intelligence (AI), Internet of Things (IoT) (Song and Le, 2020) and mobile devices technologies are the new technologies which changes the IT structure and provides newly extra values. In computer Science, Cloud computing (Mell and Grance, 2011) is the world's and ICT's (Information and Communication Technology) (Ciurea et al., 2020) most important technology. Today, due to its features such as pay per use and anywhere any time, cloud technology is used in large or small companies or businesses .Cloud computing user can access data or application without download its software and with the help of internet connection. Cloud computing uses stimulating ICT inventions like internet and distribution and virtualized computing to provide effectively integrated system. Microsoft, AMAZON and IBM are some examples of cloud computing service providers in ICT business. Cloud provides several services to the users like IaaS, PaaS and SaaS in the cloud environment with Cloud Service Provider (Yoon and Kim, 2020; Park and Seo, 2020). Users can request to the cloud service provider for the resource at anywhere, any time and the cloud service broker selects the best resource to the users with their best budge and deadline. Day by day, the number of users in cloud environments is growing because of its heterogeneous resources with the characteristics like higher availability, scalability, on demand self-services, access broad network and rapid elasticity so the work load and traffic is also increasing in cloud and the cloud service need an efficient technique to balance the user requests or load properly in cloud system. The role of Load balancing is very important issue in cloud environment because of its increasing users and it is required to manage all the user requests. Load balancing means to balance the user requests in the manner that improves the response time, scalability and maximum resource utilization. Proper load balancing is very important because it increase the resource utilization, decrease makespan time (Kurniawan et al., 2014), and improves the overall performance of cloud system. Load balancing is a method of spreading the load (a collection of tasks) over a set of cloud resources (software / hardware) in such a way that the full (maximum) use of their resources (Kumar and Sharma, 2017) should be made and the cloud system performance should be improved. In cloud computing, there are two steps to balance the cloud load-Monitoring VM (Virtual Machine) (Zhang et al., 2018) and Task (job) scheduling. Task scheduling is a full N-P complete problem because cloud computing is the collection of heterogeneous resources (or different types of configuration of host and virtual machines (VM)) and very easily changes user requests on demand. It is very difficult to find the all possible mapping between task and resources in the cloud. An effective task (job) scheduling technique is therefore needed to allocate the task set in such a way that no virtual machine should be under loaded or over loaded and all VMs should be balanced. Monitoring the virtual machine continuously with the help of task (job) migration or VM migration in the cloud

system so that balanced the load properly. If any VM is overloaded in the CRB (Cloud Resource Broker), the cloud broker monitors the VM, executes the load balancing feature on the VM and migrates the job from the overloaded VM to the under loaded virtual machine. Task or Job scheduling (Sarathambekai and Umamaheswar, 2017) is an important issue in cloud computing. The primary objective of the work scheduling approach is to map user requests to the required cloud resources available to provide the best output in the cloud environment. In the cloud system, there are three important phases needed for job scheduling techniques as User phase, Task/Job scheduler phase and Cloud phase.

User phase- In user phase, cloud users send their request T1, request T2, request T3….. request Tn with the help of web interface or GUI in service requiring in terms of QoS parameter, software or hardware.

Task/Job scheduler phase- in this second phase all functions (load balancing and scheduling) are performed with the help of task scheduler. Here user request handler forward all legitimate requests to task/job scheduler for process the user requests where matching list match or find all the user requests to corresponding VM and here scheduler assigned all the requests to available VM. The task scheduler has all the details about the VMs, either idle or active (busy).

Cloud phase- In this phase, there are several host in a datacenter and each host has a heterogeneous VM, where the number of VMs can be increased or decreased at runtime based on the host capacity and the number of user requests. Here Cloud monitoring and discovering service (CMDS) plays an important role to find the status (idle or active) of VM and cloud resource information. Task/Job scheduling methods (Dubey et al., 2018; Zhou et al., 2018) categorized into two types in cloud system shown in Fig.1. Distributed and centralized task or job scheduling in cloud. The tasks are not allocated to the same resources or the same location in distributed scheduling and all the resources are at the same location in the centralized task scheduling. Centralized task scheduling having low complexity level in comparison to distributed task scheduling. Distributed task scheduling methods are further categorized in three types as metaheuristic (Talbi, 2019), heuristic and hybrids. Metaheuristic task scheduling are swarm intelligence and nature inspired. Heuristic task scheduling again categorized into two types static task scheduling and dynamic task scheduling. Static task scheduling requires the advance information about the task/job (length of jobs, deadline of jobs and number of jobs) and resources (node processing power, memory, processing capacity etc.). The static scheduling approach is not best choice for the cloud since static task scheduling works properly when there is very little change in workload and system behavior that also does not change. Static task scheduling doesn't optimize the QoS in cloud and in real environments; it does not have good results. Some examples of Static task scheduling are FCFS, SJF, Round Robin, Min-Min algorithms etc. dynamic task scheduling is very efficient and accurate scheduling method to the cloud

environment. In dynamic task scheduling advance information is not required but it require to handle the nodes continuously. Examples of dynamic task scheduling are Heterogeneous Early Finish Time, Dynamic Round Robin (DRR), Ant Colony Optimization, Particle Swarm Optimization etc. dynamic task scheduling is suggested for the researchers because of its dynamic workload balancing and dynamic system behavior in the cloud computing. Hybrid task scheduling are energy based cost based and efficiency based.

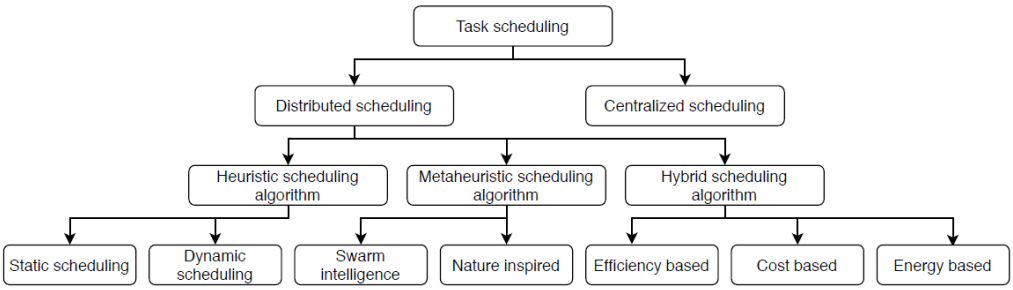

Fig.1: Task Scheduling in Cloud

We have introduced or designed a heuristic task/task scheduling dynamic load balancing approach in the cloud. Balancing the load is a major challenge of cloud because of increasing the huge number of users on demand. We have used elastic resource provisioning or deprovisioning (Somasundaram et al., 2013; Jung and Seo, 2020) and scalability approach to minimize the makespan time. Scalability (Li, 2009) enables to manage huge workloads in cloud environment without disruption of its existing infrastructure. Here three kinds of scalability are available in cloud system: first is vertical, second one is horizontal and third is diagonal scalability. Vertical scalability is also known as scale up, which means vertical scaling can add more power like number of central Processing Unit, hard disk, memory etc. in cloud environment without change its infrastructure. Horizontal scaling is also known as scale out, which means horizontal scaling can add number of VM's in resource pool of cloud environment and diagonal scaling combines the vertical and horizontal scaling. Here, we are using horizontal (scale out) scaling.

We have proposed, developed and implemented a dynamically task scheduling algorithm which is based on scheduling interval that balance the tasks among the VM with cloud elasticity which reduce the makespan time and improve the utilization of cloud resource in cloud system. In this algorithm tasks are distributing among the overloaded (OL) machine to under loaded (UL) virtual machine with the help of task migration. Cloud resources are adding with the help of horizontal Scaling method whenever the task rejection is greater than the Service Level Agreement (SLA) threshold value.

The rest of this paper is summarized here: We describe the related work of current QoS-based load balancing techniques related to our research work in the

cloud setting in Section 2, in Section 3 we explain the problem formulation, in Section 4 we describe our proposed and built load balancing technique or algorithm, in Section 5 we explain the different simulation tools and also experimental results. In section 6 we describe conclusion of our developed approach and its future work.

2. Related Work

The researchers have suggested and developed many task scheduling algorithms in the literature in cloud environment. Here we have discussed some QoS based task scheduling techniques with their Strength, weakness and tools in Table-1. In (Kumar and Sharma, 2017), Kumar M. has proposed and implemented an algorithm to improve the makespan time in cloud system using task meet to deadline approach. The Author did not consider othe QoS parameters. In (Ren et al., 2013), Author has proposed and implemented an algorithm to minimize the total execution time and get good schedule length in cloud environment but author has suffered from the load balancing in cloud system. In (Wang and Yu, 2017), author has proposed and implemented task scheduling algorithms to improve Min-Min algorithm in cloud with minimizing the total completion time. The Author has considered only one parameter to enhance the performance. In (Javanmardi et al., 2014), Author has shown that execution cost in Hybrid Job scheduling algorithm in cloud is decreased. He is also considered only one parameter to enhance the performance in cloud. In [6], Author has proposed and implemented credit based scheduling algorithm in cloud to increase resource utilization and decrease makespan time. The Author did not compare performance with other algorithms. In (Babu and Samuel, 2016), Author has proposed and implemented an algorithm to enhance the performance of bee colony algorithm with the help of decreased the makespan time. The Author compared only one algorithm to enhance the performance. In (Azad and Navimipour, 2017), Author has proposed and implemented an algorithm to reduce the execution cost and improve the throughput. No other QoS parameters have considered. In (Patra, 2018), author has proposed and implemented an efficient energy consumption algorithm to maximize the resource utilization. Other QoS parameters like execution time, throughput etc. has not considered. Adhikari M. has proposed and implemented meta heuristic-based algorithm and multi objective accelerated PSO algorithms in et el. (Adhikari, 2019; Adhikari, 2019). The Researcher improve the makespan time, availability, cost, computational time, resource utilization, energy consumption and throughput but there is novel compromise solution between cost conflicting objective has not discussed.

Table 1: Existing task scheduling algorithms

S.No.	Author/ year	Paper title	QoS Matrices	Tools	strength	weakness
1.	Kumar M.	Based on deadline constrained and	Makespan time	cloud sim	Decreasing makespantime	not improve other QoS

	2017	dynamic load balancing approach in cloud with elasticity			& increase ARUR	
2.	Ren H. 2012	Performance effective and low complicity task scheduling for heterogeneous computing ”	speed up, execution time execution time Scheduling length ratio (SLR),	Random graph generator	better scheduling length, Minimizing execution time	Suffer from the load balancing
3.	Wang G. 2014	“Task scheduling algorithm improved Min—Min algorithm in cloud	Load balancing and makespan time	GridSim , Cloud Sim	Enhance Performance and compare with Min algorithm	Compared only completion time compared
4.	Javanmardi S. 2014	Hybrid Job Scheduling Algorithm in Cloud Computing	makespan time and degree of Imbalance	cloud sim	decreased Execution cost also increased overall profit	Consider only one parameter
5.	Antony T. 2015	Credit Based task Scheduling in Cloud Computing	Priority task and makespan time	cloud sim	decreased makspan time and increase Resource utilization	No comparison of the algorithm
6.	Babu 2016	Enhanced Bee Colony Algorithm for Efficient	Imbalance degree	Physical cloud environment	Reduce makespan time	Compared with one algorithm
7.	Azad P. 2017	Cultural & ACO algorithm to optimize QoS parameters	Makespan time, energy consumption	Cloud azure	improve throughput, decrease execution cost, task rejection ratio and	Not considered Other QoS parameters
8.	Patra 2018	Efficient energy consumption based algorithm	resource utilization and Energy consumption	Cloud sim	Maximize resource utilization	not considered execution time, execution cost, throughput etc.
9.	Adhikari M. 2019	Meta heuristic task deployment approach for load balancing	Load-balancing resource clustering	-	Improve the throughput,makespan time, cost	Novel compromise solution between time

			(LBRC) with BAT technique		And availability	and cost conflicting objectives has not discussed
10.	Adhikari M. 2019	Multi-objective accelerated PSO with a container-based scheduling technique for IoT in cloud	PSO algorithm to process IoT based and non-IoT-based tasks	-	Improve energy consumption, computational time, and resource utilization etc.	conflicting objectives has not discussed

We have seen in above mentioned table that these all scheduling algorithms have different concept, QoS Matrices, tools with their advantage and weakness.

3. 3. Problem Formulation

In cloud Environment, we scheduled all the user tasks to the VM in a manner that the user tasks can be executed in a minimum amount of time and maximum resource utilization as cloud user is anticipated to minimum cost and makespan time while CSP anticipation is to maximize the cloud resources. In cloud system, cloud request scheduler receives n number of user request T1, request T2, request T3, request T4…..request Tn are independent. Each tasks length is TLTi in MI. Each task requires number of CPU q, processing speed p, main memory r and the bandwidth B in Megabits per second. Cloud user task scheduler has all information of M heterogeneous (different processor speed, memory, bandwidth, Number of Central Processing Unit etc.) Virtual Machine $VM1$, $VM2$, $VM3$, VM4……. VM. Cloud user task broker allocate all tasks to the VM, each VM has a task queue to collect the tasks and overall queue length on VM represents total load on that virtual machine.

Capacity of VM: We have calculated the individual VM capacity and the capacity of all VM's with

$$C_{VM} = p_c * q_c \quad (1)$$

Where p_c is the processing speed of CPU and q_c is the no. of busy CPU to execute job or task

$$\text{Datacenter Capacity } C = \sum_{j=1}^{M} C_{VM} \quad (2)$$

Load of a virtual machine

The Task or job scheduler assigned the task or job to VMs after finding the matchlist from matching node.

Load of a virtual machine can be find as

$$LVM_{i,t} = K * TL_{i(t)} / S(VM_{i,t}) \quad (3)$$

Where the value of K is 1,2,3,4…….……N are tasks and S(VMI,t) is the virtual machine's service rate at time t, it is expressed with computing power p and the number of CPU q

Virtual machine Service rate at a specific time, t

$$S(VM_{I,t}) = p_c * x(t) \text{ where the value of } x=1,2,3\ldots\ldots..q \quad (4)$$

We can calculate the Load on a particular virtual machine at a time t as

Lvm= (number of request on the virtual machine*length of the request) /service rate of virtual machine. Total load at all virtual machine (datacenter)

$$TLvm=\sum_{j=1}^{m} Lvm \quad (5)$$

1. Execution Time: when the task allocated to the required resources then we can calculate the expected processing time of the task on resources. When the number of cloud user requests is greater than the capacity of all virtual machines, with the aid of the principle of elasticity, the number of virtual machines increases. If cloud user request numbers are less than the capacity of all virtual machines, then all overloaded and under loaded virtual machines are verified in the cloud environment and user requests are moved from overloaded virtual machines to under loaded virtual machines, and all user requests can be executed in minimum time. We can find the user request migration time on one virtual machine to another virtual machine as

Task migration time (TT) =length of user request/bandwidth = (TL)/(B) (6)

and we can find execution time (ET) of user request T_i on virtual machine (VM_j)

$$\text{Execution time (ET)} = \sum_{i}^{N} Eij * TLi/(p_c * q_c) \quad (7)$$

Here the value of Eij=1 if user request TLi is assigned to virtual machine VMj otherwise Eij=0

now we can find the user task/job completion time as summation of task/job execution time (ET) and task/job migration time (TT) at any virtual machine VM

$$\text{Completion Time of the task (CT)} = ET + TT \quad (8)$$

2. Makespan time: The objective of our research is to minimize the makespan time which contains user request execution time and user request migration time (if any user request is migrate from overloaded O_L VM to under loaded UL virtual machine).

$$\text{(Makespan time of VM) } MT=\max \{ \sum_{j=1}^{M} CT \} \quad (9)$$

Subject to- $CT_{Ti} > A_{Ti} + ET_{TiRj}$

Where CT_{Ti} is the user task completion time at VM_j and A_{Ti} is the arrival time of user task.

ET_{TiRj} is the execution time of task T_i at VM_j. if arrival time of task A_{Ti} is previously known it means the problem will be static otherwise problem will be dynamic.

Equation (9) shows that a user task start to execute at a particular VM when previous user task/job has been executed completely at a particular VM.

$$CT_{i,Rj} \geq CT_{i-1,Rj} + ET_{TiRj} \quad (10)$$

Above equation (10) shows that new user task/job start its execution on the virtual machine if its previous task has completed the execution.

Resource utilization: one more objective of our work is to maximize the utilization of resources in cloud environment and the main objective of balancing the load in cloud environment is to maximum uses of cloud resources. So we can calculate the Average of maximum resource utilization ratio as

$$AMRU = (meantime/makespan\ time)*100 \quad (11)$$

Here meantime is the total completion time of the cloud resources (VM_j) to complete the job. The maximum average resource utilization ratio is 1, it means 100% resource utilization and the worst value of AMRU is 0, it means resources are in ideal state.

4. Proposed Dynamic Task Scheduling Architecture

We have designed and developed a cloud resource broker architecture for a dynamic task scheduling load balancing algorithm with elasticity of cloud resources (virtual machine) in figure (2) which is the modification of the proposed architecture in (Somasundaram, 2013). Objective of this architecture is to reducing the makespan time and increasing the resource utilization in cloud. Brief description of this proposed framework is as follows:

Application or User/Consumer Request handler: It handles all the consumer/ user requirements in cloud environment. The user sends request for cloud service with the help of web Interface or GUI with required service. The required service is mainly in the form of QoS (response time, throughput, efficiency, deadline etc.), Hardware (processor speed, hard disk speed, RAM speed etc.) and software (required library, operating system etc.). When request handler receives user request first it verified at the verification node weather user request is legitimate or not if the user request is legitimate then send it for further processing otherwise discard it and then require user request match with the cloud resources which are available and capable to run it in cloud environment. It is an important and challenging issue to allocate the cloud resources to the user request because user change request at run time with its requirement

Controller: Controller is an important component of the proposed architecture and handles or controls all the incoming user from the user request handler and forward the user request to other components such as Cloud Resource Provisioner (CRP), dynamic Load Balancer (ALB), Cloud Load and Resource Information (CLRI) Aggregator and Cloud Scheduler (CS).

Matching Node: This node contains all information about Virtual Machine and

cloud user request. When the matching node receives the cloud user request then first it check the user request status as it is based on priority or non-priority and also check the receiving user request having any deadline or not. Matching node matched the cloud user request to particular available require virtual machine and forward the matching list to load balancer and dynamic job scheduler for further processing.

Dynamic job/task Scheduler and cloud load balancer: here the best resource is selected from the matching list which is capable to satisfying cloud user request and allocating the job(task) to the resource as per the task scheduler method. There are many task / job scheduling techniques and cloud load balancing techniques or algorithms that operate on several parameters such as throughput, response time, cost, resource usage, makespn time etc. Dynamic user job/task scheduler schedules the jobs in such a way that all jobs are completed in minimum time.

The Load Balancer collects all the work load information of running virtual machine in cloud and monitors and computes the entire vritual machine and if the virtual instances are overloaded then user jobs of that VM is migrated to other virtual Instances (under loaded). If under loaded VM is not available then jobs is migrated (or transferred) to the balanced virtual instance which can run the job in minimum time. It also maintains threshold value for all virtual instances for each user request. If the load of the running virtual instance is above the threshold (Service Level Agreement) value then load balancer invokes the CRP for creating new virtual machine. If the load of running virtual instances is below the threshold then load balancer invokes the CRP for removing the virtual instances.

4.1. Cloud Load, Resource Information Aggregator (CLRI)

This CLRI is an important component for aggregating basic resource information as load, network, processor, memory etc. from the previously registered CSP's. CLRI periodically monitors all the resources and collects all information about these resources. It also communicates with Cloud Monitoring and Discovery Service (CMDS) to find the all information of cloud resource. CMDS is a technique to monitor the state (busy or idle) of virtual machine and collect all the available resource information and reply to Cloud Load resource information Aggregator in cloud.

4.2. Resource and Load Monitor (RLM)

It is basically important for monitoring the private cloud (as eucalyptus, Nebula etc.) resource. It uses external data providers such as Ganglia to get the speed of the processor, ram memory, hard disc space, etc., and NWS to get bandwidth, latency, etc.

Cloud Resource Provisioner: it is mainly used to create and delete the virtual machine in the cloud according to user request requirement. For example cloud middleware interacts with Eucalyptus with the help of provision/deprovision for the

virtual instances if users require it.

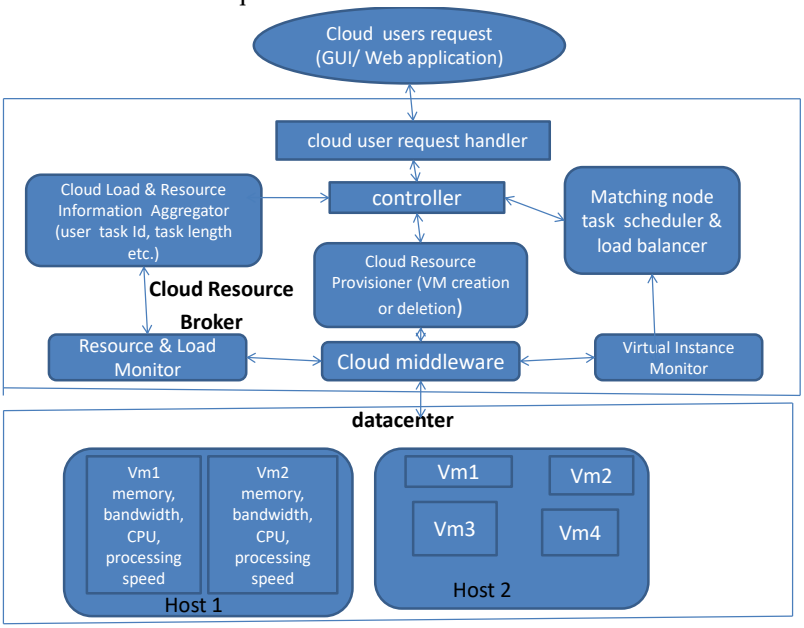

Fig. 2: Proposed cloud task scheduling Architecture

Virtual Machine Monitor: Virtual Machine Monitor: a main component of the cloud world, too. It resides at the cloud middleware level and finds the ID of the virtual machine at the cloud middleware level. It monitors load and traffic of virtual machines which is deployed with the web application. Here two methods are available of monitoring the cloud resources. The first is event-based: in case-based monitoring, if a job is assigned to a VM or removed from a VM, the event-based approach monitors the virtual machine status. And second is Time based: in this monitoring method resources are monitoring continuously with a specific time interval. Event based monitoring is used in this paper.

5. Proposed dynamic Task/ job scheduling Algorithm

We have designed and developed a heuristic task scheduling load balancing (dynamic) approach. An objective of this approach is to minimizing the makespan time and maximizing the utilization of resources in cloud computing environment with scalability mechanism. Model of task scheduling is given in fig. 2, which is the modification of the proposed architecture in (Somasundaram, 2013). Here we have taken N number of user task with random task length lies from 20K (MI) to 40K (MI) and M number of VM with dissimilar processing speed (MIPS, RAM etc). We sorts the N task in non-increasing order according to their length and sorts M VM in non-increasing order according to their computing speed with the help of bubble sort. After sorting task and virtual machine, we start to assign all tasks to VM in

FCFS. Now we generate an array which contains all assigned task registered ID of all virtual machine and start the operation of our proposed heuristic task scheduling load balancing algorithm on it. Now we check the status of all running VM's and finding load of each virtual machine to calculate the total load of the datacenter at a specific time t with the help of equation 1&2 and also calculate the capacity of virtual instance and datacenter with the help of equation 3 &4. At this stage we check all load conditions at the virtual machine. If load of the running VM's is less than the capacity of the datacenter then start to compute all virtual machine or not then cloud broker have to create virtual machine (scalability) to balance the load and we find the number of under loaded, balanced and overloaded virtual machine and we check it with threshold value. We have assigned the threshold value based on latest literature to check the virtual machine condition. If the use of the virtual machine is greater than 80% of its capacity, then it is defined as overloaded VM as variable U_{LM} and if the use of the virtual machine is <25% to 30% of its capacity, the virtual machine is defined as under loaded VM as variable O_{LM}. Now we find the list of under loaded virtual machine, balanced and overloaded virtual machine. We define total Overloaded virtual machine as $TO_{VM,}$ total under loaded virtual machine as TU_{VM} and sort the total under loaded in increasing order and total overloaded virtual machine in decreasing order. Now, we start to migrate task from overloaded VM to under loaded VM and if there is no overloaded VM then transfer task to balance all virtual machine and calculate the task migration time of overloaded VM to under loaded VM with the help of Equation (6) and add this task migration time with task execution time. Now finally we find the results of the proposed heuristic algorithm with the help of cloudsim simulation in cloud environment.

5.1. Pseudo code for Proposed Heuristic based Task Scheduling Load balancing Algorithm

For Scheduling Task:

1. *Take Number of n task $T_1,T_2,T_3\ldots\ldots T_n$ (random length)*
2. *Sort all task in decreasing order with the help of tasks length*
3. *Take m heterogeneous virtualmachine VM_1, VM_2,M_3 ….M_m*
4. *sort all VM according to their processing speed in decreasing order*
5. *start For loop for tasks (for all $\forall\ T_i \in 0$ to n-1) &&*
6. *start for loop for VM (for all $\forall\ VM_j \in 0$ to m-1)*
7. *If ($T_i \neq \emptyset$ && $VM_j \neq \emptyset$) then allocate task $T_i \rightarrow VM_j$ in FCFS order*
8. *End virtual machine for loop*
9. *End task for loop*

For the Load Balancing:

1. *For loop (for $\forall\ VM_j \in 0$ to m-1 && $T_i \in 0$ to n-1)*
2. *Find load at virtual machine & datacenter with the help of equation (1 to 6)*

3. *If (TL_{VM}>C) go to step 19, else*
4. *find the list of overloaded, under loaded virtual machine*
    a. *U_{LM}=.25*VM_j & O_{LM}=.8VM_j*
5. *Sort the under loaded virtual machine (U_{LM}) in increasing order && overloaded virtual machine(O_{LM}) in decreasing order*
6. *If (O_{LM}!=Ø && U_{LM}!= Ø)*
7. *For loop (for k=0 to O_{LM})*
8. *Migrate all the task or user jobs from overloaded (OVM) to under loaded(ULV) VM (T_i →VM_j) until the load of VM (VM_j<O_{LM} && VM_j>U_{LM}) and calculate the task migration time by equation (6)*
9. *Now check the status of all VM, if any running virtual machine is still Overloaded then repeat load balancing again.*
10. *find the total number of overloaded virtual machine TO_{VM}( ), & check if (n/10<TO_{VM}<n/3) then boot 20% new VM or boot 30% new VM and distribute the load to all VM fairly.*
11. *verify the load condition (TL_{VM}>C) if yes, repeat the step 19 or go to next step.*
12. *find the total number of under loaded VM TU_{VM} if (n/5 <TU_{VM} <n/2) then remove 20% virtual machine or remove 30% VM and repeat it until the load of all VM is balanced.*
13. *Find the execution time and calculate the makespan time with the help of equation (8 to* 9)

6. Simulations and Analysis

Simulation is the best method to check the performance of proposed algorithm. It is not easy or not possible to implementation or experiment the new method/technique in real cloud system because it is very high cost and time consuming process. Cloudsim simulator is used to implement the proposed heuristic task scheduling algorithm. There are several cloud simulators to implement the research work such as cloudAnalyst, iCanCloud, GroudSim, Cloudsim, DCSim (Data Centre Simulation), GreenCloud etc. cloudsim is a new, extensible, highly generalised and Java based simulator kit. We have used cloudsim toolkit because it is easy to analyze the various parameters as response time, processing time, energy consumption , cost etc. and it is also repeatable, controllable dependable and scalable which make sense to do some valuable modification and correction in the existing resources before applying in the real cloud. We have run our algorithm in cloudsim simulator which contains datacenter and there are number of host in the data center and each host has number of VMs with various parameters as shown in Table 2.

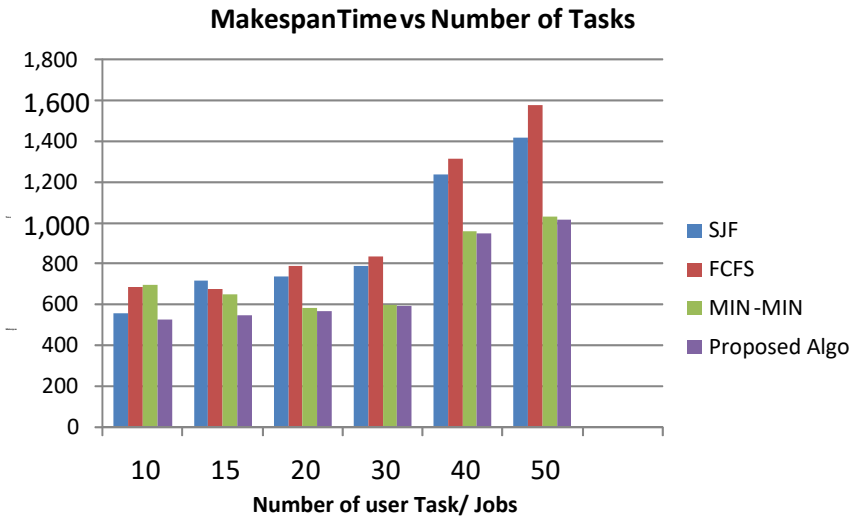

Fig 3. Comparison of makespan time for Proposed Algorithm with SJF, FCFS and Min-Min Algorithms

6.1. Minimum makespan time

Here we have taken n task with random task length from 20K to 40K in MI and m heterogeneous VM with different processing speed in terms of RAM, MIPS etc. Here we have 5 virtual machines and assigned all the tasks given in table- 3 on these VMs with the help of scheduler as per our proposed scheduling algorithms in fig. first we have optimized makespan time in cloud environment with the help of our proposed load balancing algorithm. We have run the simulation process in cloudsim approximately 200 times to find the makespan time with the help of our proposed task scheduling load balancing method in space shared manner. First we have taken 5 virtual machine (heterogeneous VM) and secondly we have allocated the tasks (10 independent task random length) to all virtual machine according to our proposed method using matching node & scheduler and monitor them continuously. Here we have not considered any task cost, priority, deadline etc. Now we got the list of under loaded virtual machine and overloaded virtual machine. Now we have increased number of task 10 to 50 and processed it the entire random length task allocated on the 5 virtual machine. After that, we found the list of VMs overloaded and underloaded and measured the VM 's capacity and load at a particular time t. If a VM load is > 80 percent, we have made the array list of overloaded, under loaded and balanced virtual machine, then add it to the overloaded array list and if a VM load is < 25 percent, then add it to the under loaded array list and the rest of the virtual machine added to the balanced array list. Now we have sort the overloaded array list of VM in decreasing order and under loaded array list of VM in increasing order and start to migrate load from

overloaded VM to under loaded VM and calculated the task migration time and divided it by the bandwidth. After completion task migration, the proposed algorithm checked that all the VMs are balanced or not. If all the VM are in balanced condition then terminate the algorithm or if all the VM are not in balanced then repeat the load balancing process again. We have calculated the makespan time of our proposed algorithm and compared the calculated result with other load balancing algorithms like FCFS, Min-Min and SJF. We have shown results in fig. 3 to validate our proposed algorithm. Results shows that proposed heuristic algorithm gave better performance in comparison to existing algorithms. Further we have increased virtual Machines with their speed shown in figure and allocated 12 tasks on theses virtual machines with random length as given in table. Now find the underload, over loaded virtual machine as per our proposed algorithm but we did not find any under loaded and over loaded virtual machine so we have calculated the makespan time and compared it with other SJF, Min-Min and FCFS algorithms as shown in figure 5. Further we increased number of tasks upto 45 to 50 and we have required to increase VMs. Now we calculate the makespan time and compare it again with other Min-Min, FCFS and SJF algorithm.

Table 2: Properties of virtual machine

VIM ID	VM image size	VM MIPS	No. of CPU	Memory	Hypervisor (VMM)
0	1000	1000	1	512	Xen
1	1000	680	1	256	Xen
2	1000	580	1	256	Xen
3	1000	600	1	512	Xen
4	1000	640	1	512	Xen
5	1000	400	1	256	Xen

Table 3: Properties of user tasks

Cloudlet ID	Task length	File size	Output size	CPU
0	110228	300	300	1
1	309350	300	300	1
2	643098	300	300	1
3	870987	300	300	11
4	250712	300	300	1
5	955696	300	300	1
6	182678	300	300	1
7	345678	300	300	1
8	445968	300	300	1

6.2. Maximum Resource Utilization

We have calculated the average resource utilization ratio using equation (11) and compare it with existing FCFS and SJF and Min-Min algorithm. We have taken the parameters from the Tables 1 & 2 to find the resource utilization in cloud environment. As per our proposed algorithm, all tasks is allocated to the cloud

resource (VM) but some VM will be in under loaded or over loaded state and find the list of under loadeVM and overloaded VM. Now sort the under loaded VM in increasing order and sort the overloaded VM in descending order and start to migrate load from overloaded VM to under loaded VM after that most of the VM's will be balanced and all the resources will be utilize efficiently. Calculated ARUR of the proposed dynamic concept shows that proposed concept utilizes the cloud resources better than 10% min-min and 30% better of SJF and FCFS algorithm.

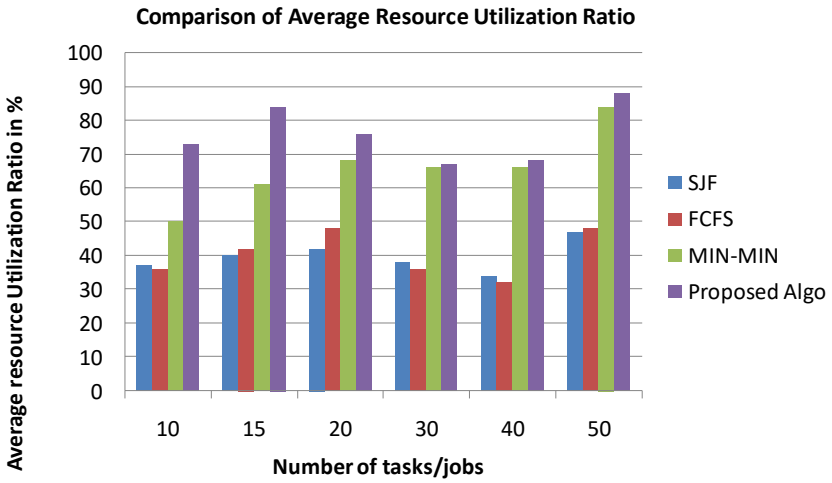

Fig. 4: Average Resource Utilization of proposed algorithm with SJF, FCFS and Min-Min Algorithms

7. Conclusion

We have design and implement a heuristic task scheduling load balancing concept to minimize the task's makespan time and maximize the resource utilization in cloud environment. We have provided the scalability service (horizontal scalability) if user task load is greater than the capacity of datacenter in cloud. The main objective of our designed algorithm is to maximum uses of the cloud resource so that increasing the execution speed of the cloud user applications. There are several existing algorithms available in research based on different parameters as execution time, resource utilization, makespan time, throughput, response time, etc., in cloud computing environment. We have proposed and implement a heuristic based task scheduling load balancing algorithm in cloud environment. In this algorithm we have used the task/job migration method and horizontal scalability approach to minimize the makespan time and maximize the resource utilization. We have implemented proposed algorithm in cloudsim simulator and the evaluated experimental comparison result shows that performance of the proposed concept is better than the other existing FCFS, Min-Min and SJF algorithm in all possible

circumstances. This proposed algorithm minimize the makespan time as well as maximize the average resource utilization ratio (ARUR) as comparison to FCFS, Min-Min and SJF in Fig. 4. In future we will work on other QoS parameters like deadline based task and priority based task in cloud environment.

References

Adhikari M., (2019). Meta heuristic-based task deployment mechanism for load balancing in IaaS, cloud. *Journal of Network Computer Application*, 128, 64–77.

Adhikari M, and Srirama S., (2019). Multi-objective accelerated particle swarm optimization with a container-based scheduling for Internet-of-Things in cloud environment, *Journal of Network Computer Application*, 137, 35–61.

Azad P. and Navimipour N. J., (2017). An Energy-Aware Task/job Scheduling in the Cloud Computing Using a Hybrid Cultural and Ant Colony Optimization Algorithm. *International Journal of Cloud Applications and Computing (IJCAC)*, 7(4), 20-40.

Babu, K.R., and Samuel, P., (2016). Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud. *Innovations in Bio-Inspired Computing and Applications*, 67-78.

Chen H., Wang F., Helian N., and Akanmu G., (2013). User-priority guided min-min scheduling algorithm for load balancing in cloud computing. *National Conference on Parallel Computing Technologies, (PARCOMPTECH) Bangalore*, 1-8.

Ciurea C., Pocatilu L., and Filip Fl. G., (2020). Using Modern Information and Communication Technologies to Support the Access to Cultural Values. *Journal of System and Management Sciences*, 10(2), 1-20.

Dubey, K., Kumar, M., and Sharma S.C.,(2018). Modified HEFT algorithm for task scheduling in cloud enviromnment. *procedia Computer science*, 125, 725-732.

Fidel M.B., and Carmen G. C. E.(2015). Human Resources and Information Technology: Business Strategies for Sustainable Human Development. *Journal of System and Management Sciences*, 5(3), 18-31.

Javanmardi, S., Shojafar, M., Amendola, D.,Cordeschi, N., Liu, H., and Abraham, A., (2014). Hybrid Job Scheduling Algorithm for Cloud Computing Environment.

Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA, 43-52.

Jung H.C., and Seo K.K., (2020). Jobs and Potential Threat Assessment for Continuous Provision of Cloud Computing Services. *Journal of System and Management Sciences*, 10(4), 45-61.

Kumar, M., and Sharma, S.C.(2017). Deadline Constrained based Dynamic load balancing algorithm with elasticity in cloud environment. *Journal of Computers & Electrical Engineering (CAEE)*, 69, 395-411.

Kurniawan H., Sofianti T.D., Pratama A.T., and Tanaya P.I.(2014). Optimizing Production Scheduling Using Genetic Algorithm in Textile Factory. *Journal of System and Management Sciences*, 4(4), 27-44.

Li J., Chinneck J., Woodside M., and Litoiu M. (2009). Fast Scalable Optimization to Configure Service Systems having Cost and QoS Constraints. *Proc. IEEE International Conference on Autonomic Systems*, Barcelona, 159-168.

Mell, P., and Grance, T., (2011). The NIST definition of cloud computing. *Technical report published in NIST Special Publication*, 800-145.

Park W. and Seo K., (2020). A Study on Cloud-Based Software Marketing Strategies Using Cloud Marketplace, *Journal of Logistics. Informatics and Service Science*, 7(2), 1-13.

Patra, S. S., (2018). Energy-Efficient Task Consolidation for Cloud Data Center. *International Journal of Cloud Applications and Computing (IJCAC)*, 8(1), 117-142.

Ren. H., Lan, Yihua, and Chao Yin, (2012). The load balancing algorithm in cloud computing environment. *International Conference on Computer Science and Network Technology*, Changchun., China, 925-928.

Sarathambekai, S. and Umamaheswari K. (2017). Task scheduling in distributed systems using heap intelligent discrete particle swarm optimization. *Comput. Intell.*, 33, 737-770.

Somasundaram, T.S., Govindarajan K, and Rajagopalan MR, (2013). A broker based architecture for adaptive load balancing and elastic resource provisioning and deprovisioning in multi-tenant based cloud environments. *Proceedings of International Conference on Advances in Computing*, 561-573.

Song Y. and Le J.(2020). A Blockchain-based Fog-enabled Energy Cloud in Internet of Things. *Journal of Logistics, Informatics and Service Science*, 7(2), 45-64.

Talbi, E.G., (2009). Metaheuristics: From Design to Implementation. *1st Ed. John Wiley and Sons Inc, Hoboken*, NJ, USA, ISBN-13: 9780470278581, pp: 500.

Thomas A., Krishnalal G., and Jagathy Raj V.P., (2015). Credit Based Scheduling Algorithm in Cloud Computing Environment. *Procedia Computer Science*, 46, 913-920.

Wang G., and Yu H. C., (2014). Task Scheduling Algorithm Based on Improved Min-Min Algorithm in Cloud Computing Environment. *Applied Mechanics and Materials*, 303-306, 2429-2432.

Yoon S., and Kim H., (2020). Design and Implementation of the IoT Cloud Web Server System for the Control of Insect Farming Facilities. *Journal of System and Management Sciences*, 10(3), 73-85.

Zhang, F., G. Liu, X. Fu and R. Yahyapour, (2018). A survey on virtual machine migration: Challenges, techniques and open issues. *IEEE Communication. Surveys Tutorials*, 20, 1206-1243.

Zhou, R., Li, Z., and Wu, C. (2018). Scheduling Frameworks for cloud container service. *IEEE/ACM Transportation Network*, 26(1), 436-40.