ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 8, Issue 12, December 2021

Improved Credit Based Cost Aware Scheduling Algorithm in Cloud Environment

Dr. S. Vaaheedha Kfatheen[1], B. Rekha[2]

[1]Assistant Professor, [2]Research Scholar in Computer Science

Department of Computer Science, Jamal Mohamed College (Autonomous)

(Affiliated to Bharathidasan University)

Tiruchirappalli 620 020, Tamil Nadu, India

[1]svaaheedha73@gmail.com, [2]rekaguru7@gmail.com

Abstract: Cloud computing is developing as an incipient paradigm of immense-scale distributed computing. Cloud computing is a model for allowing appropriate on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned as well as unconfined with minimal management power or service provider interaction. In order to maximize the effectiveness of cloud computing, we require an efficient scheduling algorithm. Task scheduling is one of the main issues in cloud computing environment. Efficient task scheduling is significant to achieve minimum makespan, cost-effective execution and increase resource utilization. Here, an Improved Credit Based Hybrid Algorithm (Max-Min and Min-Min) is proposed for resolving task scheduling problem in the cloud. The objective of the proposed hybrid algorithm is to attain maximum resource utilization, minimum makespan, and minimum total computation cost. The modified scheduling technique assigns tasks depending upon the grouping of credits given to the parameters: Task Length, Task Priority, Deadline of the Task and Cost. These four parameters are joined to obtain the final credit for the task; this will decrease the chances of same priority occurrence among the two tasks. Based on the highest credits and length of the task, the tasks will be allotted to the VM with minimum cost using Max-Min or Min-Min. The proposed algorithm produces an improved schedule that maximizes resource utilization, minimizes makespan and total computation cost than the existing HAMM, Max-Min and Min-Min algorithms.

Keywords: Cloud computing, Task Scheduling, Max-Min algorithm, Min-Min algorithm, Hybrid algorithm, Cost-aware.

I. INTRODUCTION

Cloud computing is one of the forms of utility computing in which the consumer need not own any organisation or software or platform needed for them. Instead the consumer can make use of the service provided by the cloud and they can pay as per usage. Cloud offers dynamic services using the conception of virtualization and scalability.Cloud can be defined as an execution environment in flexible fashion which includes a metered service for multiple diverse consumers. Some of the services provided by cloud environment are Software as a service (SaaS), Platform as a service (PaaS) as well as Infrastructure as a service(IaaS). Cloud technology is the pioneer in applying commercial form of using the computer science by public users. The concept of virtualization is used to share the obtainable resources among the users who demanded the service.The cloud offers a high performance by distributing the workloads among all the obtainable resources efficiently which decreases the execution time, waiting time and throughput [1].The distributed computing environments provide numerous services out of which job scheduling is a main activity. The performance can be improved by efficient task scheduling which can also balance the load between the resources. Scheduling is the process of mapping the jobs given by users to the best accessible resource so as to reduce the execution time, waiting time and cost. Scheduling maps the user jobs based on user requirements and it can be done at job level or task level.

The cloud environment comprises huge number of tasks and computing resources. Identification of appropriate virtual machine for allocation of resources to complete any given task is done through the task scheduling algorithms which plays a significant role in the cloud computing process. Task scheduling techniques are mainly used to improvise the makespan & resource utilization and decrease

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 8, Issue 12, December 2021

the total computation cost which makes many users to prefer cloud computing widely at affordable cost [2].

The concept of cloud scheduling states that mapping jobs to a range of virtual machines or allocating virtual machines to use the resources accessible to fulfil user requirements. The purpose of using scheduling methods in cloud computing is to improve system throughput and load balancing, reduce costs, save energy, increase resource utilization, and reduce processing time. Scheduling manages CPU and memory accessibility; a good scheduling plan rises resource utilization**.** While the task scheduling techniques are used to discover the order in which tasks or activities should be completed. It focuses on mapping the user tasks to the accessible resources.

II. TASK SCHEDULING

Task scheduling is the key challenge for the service provider in cloud Computing. One of the main key concepts in the task scheduling is to allot tasks to the available virtual machines so that certain machines do not overload or else under load [3].To do this, load balancing plays a vital role in the scheduling problem. Using a suitable load balancing method can decrease response time and increase resource utilization.

From the view of customers, a proper scheduling algorithm should run their demanded tasks in the shortest volume of time on the virtual machine. On the other hand, the service provider needs a type of scheduler, which can exploit resource usage while growing customer satisfaction. This increases the service provider`s need to select a appropriate method to schedule tasks. Regarding the task scheduling problem, numerous solutions have been proposed, and each targets to satisfy one or numerous constraints considered by the users and service providers.

Task scheduling plays an important role of affecting quality of services on the Cloud Computing. It targets to select appropriate and available resources to execute jobs or to allot computer machines to jobs in such a method that the completion time is abridged as possible. In a task scheduling algorithm, a list of jobs is made by assigning priority to each job on the basis of dissimilar parameters. Jobs are then nominated based on their priorities and allocated to accessible resources that gratify a pre-determined target function.

One of the majorgoals of the task scheduling process is a decrease in makespan of applications [4]. Makespan represents the difference time among starting and ending an arrangement of tasks. Thus, algorithms that allot the tasks to the obtainable resources and decrease makespan are needed. Load balancing targets to balance the load of the whole system by transferring additional tasks from an overloaded virtual machine (VM) to a suitable under loaded VM. In Cloud Computing, the task scheduling process is known as an NP-complete problem. In this type of problem, the essential time to manage the solution differs depending on the size of the problem.

III. EXISTING HEURISTIC ALGORITHM

Heuristic algorithms are types and applications of the batch mode which process the task at a given time as batches. The heuristic algorithms depends on Makespan (also called overall completion time) of the schedule. The proposed algorithm is compared with the existing heuristic algorithms: Min-Min, Max-Min and Hybrid Algorithm of Min-Min & Max-Min (HAMM) which are discussed below:

A) Min-Min Algorithm

The Min-Min algorithm initiates with a set of tasks that are yet to be scheduled on the resources. It calculates the completion time of each task present in the set. It then picks the task with minimum completion time and allots it with a resource that is expected to complete the allotted task faster. The process described above is repeatedly executed till every task is assigned with resource [5]. Min-Min algorithm gave priority to tasks with minimum completion time. when task with minimum completion time are much more in number then the total response time of the system will be increased which is the major drawback in Min-Min algorithm [6].

B) Max-Min Algorithm

The Max-Min algorithm first calculates the completion time of each task present in the set and then selects the task with the maximum expected completion time and assigns that task to the resource with a minimum overall execution time [7]. Max-Min algorithm gave priority to task with maximum completion time. However, its disadvantage is that, it sometimes leaves the short tasks unattended or waited for so long time when we have much more numbers of tasks with maximum completion time [8].

C) Hybrid Algorithm of Min-Min and Max-Min (HAMM)

HAMM is a hybrid of two heuristic algorithms: Min-Min and Max-Min which is proposed to overcome the drawbacks and to utilize the advantages of both the Min-Min and Max-Min algorithms. The HAMM scheduling process starts with calculating the average of Task's Length (TL) for

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 8, Issue 12, December 2021

all tasks in the Meta-Task (MT) queue. After calculating the AvgTL, two empty counters are formed, (a) Minimum Task Length Counter (TLCmin) for counting all tasks which have Task length lesser than or else equivalent to the AvgTL, and (b) Maximum Task Length Counter (TLCmax) for counting all tasks which have Task length better than the AvgTL. Followed by, choosing task Ti from MT queue and relate its length with average of all task length; if TLi < AvgTL, the counter TLCmin will be improved with one degree, or else the counter TLCmax will be improved with one. This process reprises until all the tasks in MT have been compared. After comparing the counters, the task scheduling takes place. If TLCmin >= TLCmax, the HAMM scheduler chooses Max-Min which allots longest length task. If TLCmin <= TLCmax, the HAMM scheduler selects Min-Min which allots smallest length task. The HAMM algorithm offers better in most of optimization parameters; minimum makespan, best utilization of resources, proficiently balancing of workloads among resources, improve average waiting time, and best concurrent execution among small & long length tasks[9].The major drawback in HAMM algorithm is that the scheduling process is done by considering the maximum and minimum task length counter. If the minimum task length counter >= maximum task length counter, then the scheduler selects Max-Min algorithm which allots longer length tasks; otherwise the scheduler selects Min-Min algorithm which allots smallest length tasks. This may lead to improved wait time due to task length counter is being checked for every task before assigning the suitable algorithm.

IV. PROPOSED METHODOLOGY

In the proposed scheduling technique,task's priority is allotted depending upon combination of credits given to parameters like Task Length, Task Priority, Deadline of the Task and Cost. This technique decreases the chances of same priority existence among the two tasks. To enhance the problems faced during scheduling, proposed algorithm uses the concept of load balancing with credit based scheduling algorithm. System load is accomplished by Load balancing method which calculates the load on each virtual machine and transfers the loads based on demand and supply values of overloaded and available machines. Also Scheduling process in the proposed algorithm focuses on VM cost and task execution cost, as it focuses on cost aware scheduling.

Proposed methodology of Improved Credit Based Cost Aware Scheduling Algorithm (ICCASA) has three steps:

A. Calculating credit values for the four parameters
B. Calculating Total Credit of the Task(i)
C. Scheduling Tasks using Hybrid Algorithm

A) Calculating credit values for the four parameters

(i) Task Length:

The credit system based on task length will operate as follows: (a) to find the length of every task $T_{len(i)}$ (b) to calculate the average of tasks length avg_{len}(c)to calculate variance in length with respect to avg_{len}. Credits are allotted to each task after determining the variance in task lengths of each task. Here equation (1) is used for finding the variance in task length with respect to average task length. This data is useful when tasks are organized in an array of growing order of task length (as lowest to highest). The proposed ICCASA algorithm takes every task from the middle instead of taking task with larger length and smaller length.

$$| T_{LD(i)} = avg_{len} - T_{len(i)} | \quad ...(1)$$

where $T_{LD(i)}$ is the task length difference of task i, computed by taking the absolute variance of the average of task length and task length of the i^{th} task.

(ii) Task Priority:

Every task has dissimilar priorities which are allotted to each task and in some cases the priority can be the same for one or more tasks. In general scheduling algorithms based on task priority has the difficult of treating tasks with same priority. In the proposed approach, this problem will not rise since the task priority is allotted based on the total credits obtained from the credits of four parameters where dissimilar priority are allotted to dissimilar tasks.

$$C\_Priority(T_i) = prior\_(T_i)/division\_factor \quad ...(2)$$

Credit value of the priority of task (T_i) is calculated as priority of the task T_i divided by the division factor as shown in the above equation (2). The value of the division_factor is determined based on number of digits in highest value priority. If the highest value priority is two digits then the division_factor is 100, if the highest value priority is three digits then the division_factor is 1000 and rises in multiples of 10 as the number of highest value priority is incremented.

(iii) Deadline of the Task:

Target time frame indicating the time constraint of the submitted task is the deadline. In real time applications tasks come with a deadline intended for completion of the task.

$$C\_Deadline(T_i) = C\_Length(T_i) * C\_Priority(T_i) / MAX(VM_{MIPS}) \quad ...(3)$$

Deadline Credits of the task is calculated in two steps. (1) Maximum Capability of the VM is recognized depending

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 8, Issue 12, December 2021

on the MIPS of the VM List which is accessible for the task execution, (2) then, the Deadline Credits is calculated as the product of Credit of Task Length and Credit of Task Priority is divided by Maximum Capability of the VM as shown in the above equation (3).

(iv) Cost of the Task:

Cost of Task is the amount that has to be paid for consuming the resources &utilities. Cost of a task can be calculated as the sum of the two products (i) Cost per Memory Size & VM_{RAM} and (ii) Cost per Storage &VM_{size} as shown in equation (4)

$$C\_Cost(T_i) = (DataCenter\_Cost_{perMemory} * VM_{RAM}) + DataCenter\_Cost_{perStorage} * VM_{size}) \quad ...(4)$$

B) Calculating Total Credit of the Task(i)

Four credits are calculated distinctly. Final step in the algorithm is to find out the entire credit based on task length and task priority.

$$Tot\_Credit(T_i) = C\_Length(T_i) + C\_Priority(T_i) + C\_Deadline(T_i) + C\_Cost(T_i) \quad ...(5)$$

$Tot\_Credit(T_i)$ is the total credit of Task i which is calculated by using the above equation(5). In the above equation $C\_Length(T_i)$ is the credit based on task length, $C\_Priority(T_i)$ is the credit based on priority, $C\_Deadline(T_i)$ is the credit based on deadline and $C\_Cost(T_i)$ is the credit based on cost. Then the tasks are arranged on descending order (from highest to lowest) based on $Tot\_Credit(T_i)$. Lastly, task having highest credit will be scheduled first.

C) Scheduling Task using Hybrid Algorithm (Min-Min and Max-Min)

Improved Credit Based Cost Aware Scheduling Algorithm (ICCASA) calculates cost of CPU, RAM, storage, bandwidth to certify cost effective VM is allotted for tasks accordingly. In this approach tasks are scheduled based on the priority of the task; where the task priority is allotted by using the total credit value attained from the four parameters: task length, task priority, deadline of the task and cost of the task. The task are sorted in descending order on the basis ofhighest to lowest total credit value and then prioritized. The task having highest total credit value gets highest priority. After prioritizing the task, the scheduling process takes place.

In ICCASA, a hybrid algorithm is used in allotting the task on accessible cost effective VM. Here, Min-Min and Max-Min are the algorithms used as a hybrid algorithm. In general, Min-Min strategy is used to implement small tasks before large tasks and Max-Min strategy is applied to evade the delays in large tasks execution.In proposed algorithm's scheduling process, the task with highest credit will be allotted to a VM with minimum execution time & minimum execution cost using Min-Min algorithm. Next in line, task with second highest credit will be allotted to a VM with minimum execution time & minimum execution cost using Max-Min algorithm till the task set is empty. By using Min-Min and Max-Min algorithm alternatively; the proposed algorithm utilizes the benefits and overwhelm the limitations in both the Min-Min and Max-Min algorithms. The proposed scheduling algorithm focuses on resource utilization, makespan, load balancing and total execution cost, which produce better outcome than the existing scheduling algorithms.

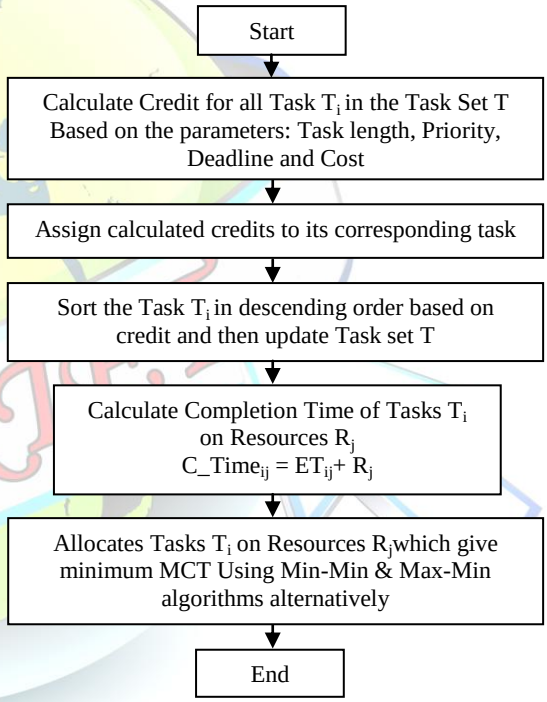

Figure 1. Work Flow of the Proposed ICCASAAlgorithm

Algorithm: Improved Credit Based Cost Aware Scheduling Algorithm (ICCASA)

Stage 1: Calculate credit value & Assign Priority to the task based on Credits

For all submitted tasks in the set Ti in task set T

// Calculate Credit for Task Length

$T_{LD(i)}$= Absolute_difference($avg_{len} - T_{len(i)}$)

val_1= $highest_{len}$ / 5;

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 8, Issue 12, December 2021

val_2= $highest_{len}$ / 4;
val_3=val_2+val_1;
val_4=val_3+val_2;
If $T_{LD(i)} \leq val_1$ then set *C_Length*=5 to T_i
else if $val_1 < T_{LD(i)} \leq val_2$ then set *C_Length*=4 to T_i
else if $val_2 < T_{LD(i)} \leq val_3$ then set *C_Length*=3 to T_i
else if $val_3 < T_{LD(i)} \leq val_4$then set *C_Length*=2 to T_i
else $val_4 > T_{LD(i)}$then set *C_Length*=1 to T_i

// Calculate Credit for Task Prioirity
Find out task T_i with h_priority
Choose division_factor based on number of digit in h_prioity
$C\_Priority(T_i) = prior\_(T_i)/division\_factor$

// Calculate Credit for Deadline
Find out the $MAX(VM_{MIPS})$
$C\_Deadline(T_i) = C\_Length(T_i) * C\_Priority(T_i) / MAX(VM_{MIPS})$

// Calculate Credit for Cost
$C\_Cost(T_i) = (DataCenter\_Cost_{perMemory} * VM_{RAM}) + DataCenter\_Cost_{perStorage} * VM_{size})$

// Calculate Total Credit for Cost
$Tot\_Credit(T_i) = C\_Length(T_i) + C\_Priority(T_i) + C\_Deadline(T_i) + C\_Cost(T_i)$
Assign $Tot\_Credit(T_i)$ for all task

Stage 2: Sort the task based on Credit value
Sort T_i in descending order based on $Tot\_Credit(T_i)$
Update task set T with sorted T_i
End For

Stage 3: Calculate Completion Time
For all tasks (T_i) in the Task set (T)
For all resources (R_j)
Compute $C\_Time_{ij} = ET_{ij} + R_j$
End For
End For

Stage 4: Task scheduling using Min-Min & Max-Min alternatively
Loop until all tasks T_i in the task set (T) are mapped.
For every task T_i in MT queue
a) Choose T_i with $Min(C\_Time_{ij})\&R_j$ with minimum cost
Allocate T_i to the R_j that gives the MCT using Min-Min
Delete the task Ti from the T.
b) Choose T_i with $Max(C_Time_{ij})\&R_j$with minimum cost
Allocate T_i to the R_j that gives the MCT using Max-Min
Delete the task Ti from the T.
c) Choose T_i with $Min(C\_Time_{ij})\&R_j$ with minimum cost
Allocate T_i to the R_j that gives the MCT using Min-Min
Delete the task Ti from the T.
// Allocate task using Min-Min and Max-Min alternatively till the task set T is empty
End for

PERFORMANCE PARAMETERS IN ICCASA

The performance of result is assessed according to the below parameters:

a) Makespan (MS): Makespan is defined as the time necessary for completing all the tasks. If the makespan is less, the load balancing also better in result. Makespan is calculated by using equation (6).

$$MS = Max\ (C\_Time_{ij}) \quad \ldots(6)$$

Where C_Time_{ij} is the completion time of tasks on resource

b) Load Balancing (LB): Load Balancing is the distribution of workloads among all cloud resources in effective and balanced manner. Load Balancing should be high, for effectual and best task scheduling algorithm. Load in virtual machine is calculated by using equation (7).

$$Load(VM) = \frac{\sum_{i=0}^{n} TLi}{C(VM)} \quad \ldots(7)$$

c) Response time (RT): Response time is the time, a system needs for reacting to start a particular task. Lesser the response time produces greater performance of the system. The response time is calculated by using the equation (8)

$$RT = actual\_CPUTime - Exec\_Start\_time \quad \ldots(8)$$

d) Cost:Total processing cost of the task comprises CPU, RAM, Bandwidth, storage & total cost of all resources are calculated by using equations (9), (10), (11) & (12).

$$Cost(CPU) = \sum_{i=1}^{n} \frac{TL}{C(VM)} * Cost\_CPU(j) \quad \ldots(9)$$

$$Cost(RAM) = \sum_{i=1}^{n} ((in_{task} + out_{task}) * Cost\_RAM(j) \ldots(10)$$

$$Cost(BW) = \sum_{i=1}^{n} ((in_{task} + out_{task}) * Cost_{Bw(j)} \quad \ldots(11)$$

$$Cost(ST) = \sum_{i=1}^{n} ((in_{task} + out_{task}) * Cost\_Storage(j) \ldots(12)$$

The total cost of all resources can be attained by using equation (13).

$$Total\ cost = Cost(CPU) + Cost(RAM) + Cost(BW) + Cost(ST) \quad \ldots(13)$$

e) Memory Cost: Cost related with the memory utilization of a task can be calculated by using equation (14).

$$Cost_{PerMem} = VM_{getRam} + Cost_{PerMem} * VM_{getSize} \quad \ldots(14)$$

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 8, Issue 12, December 2021

f) Total Computational Cost:The overall computation cost is calculated by using equation (15).

$$Total\_Cost_{Comp} = (Cloudlet_{Length}/VM_{Mips} * VM_{NumberOfPes}) * (Cost_{PerMem} * VM_{getRam} + Cost_{PerMem} * VM:_{getSize}) \quad ...(15)$$

g) Resource Utilization (RU):Resource Utilization is the usage of resources in the system. If load balancing is maximum means resource utilization is maximized. The rate of resource utilization is calculated by using equation (16).

$$RU_j = \Sigma T_i(t_{end(i)} - t_{start(i)}) \quad ...(16)$$

where, $t_{end(i)}$is the finishing time and $t_{start(i)}$ is the start time of task T_i on resource r_j

h) Average of Resource Utilization (ARU): Average resource utilization is the sum of Resource utilization rate of all resource by total number of resources. The average resource utilization is calculated by using equation (17).

$$ARU = \Sigma RU_j / N \quad ...(17)$$

where, $RU_j$is the Resource utilization rate of all resource, N is the Total number of resources

V. Experimental Result

Cloud computing environment consists of several heterogeneous resources called data centers, which include a number of hosts having numerous characteristics. Each host has a number of VMs with many configurations (CPU, memory, bandwidth and storage). User sends the requests to resources through service provider. Service provider serves these requests with effective algorithms and executes tasks in virtual machines using scheduling algorithms that are accessible on resources. The proposed algorithm has been executed in Java using CloudSim 3.0.3 as a Cloud Simulator. It helps to arrange numerous cloud applications and let's creating datacenters, virtual machines and other facilities which can be quickly generated as per need with utmost ease. The simulation is done under the subsequent conditions. Following three tables, Table 1,2&3 defines the simulation conditions.Basic configuration is stated below Table 1.

Table 1. Basic configuration

Number of Datacenters	2
Number of cloudlets	10
Number of brokers	1
Number of hosts under each datacenter	2

Each data center consists of numerous hosts. Every host has its own configuration. Here same configuration is applied for all hosts. Host configuration is stated in below Table 2.

Table 2. Host configuration

RAM (in MB)	16384
Bandwidth (in mbps)	10000
Storage (MB)	1000000
MIPS (Lines of Codes)	1000
Number of VM	3

Host in the datacenter contains numerous virtual machines. Every virtual machine has its own configuration. Here similar configuration is applied for each VMs. Virtual machine configuration is stated below Table 3.

Table 3. VM Configuration

Number of Cores	3
Size (MB)	10000
MIPS (Lines of Codes)	1000
RAM (in MB)	1024
Bandwidth (in mbps)	1000

The performance of the proposed algorithm is evaluated according to the following parameters:

a) Makespan: From the Table 4 and Figure 2, the simulation result of the proposed ICCASA algorithm is better than the existing HAMM, Max-min and Min-Min algorithms in term of makespan parameter.

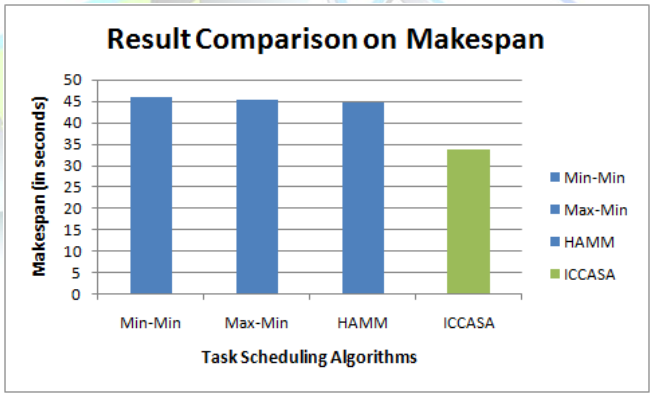

Figure 2. Simulation Result in terms of Makespan

b) Average Resource Utilization: The average of resource utilization (ARU) of the proposed ICCASA algorithm is shown in the Table4 and Figure 3; where the proposed (ICCASA) and the existing Max-Min, HAMM achieved

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 8, Issue 12, December 2021

same degree in terms average of resource utilization for the illustrated example.

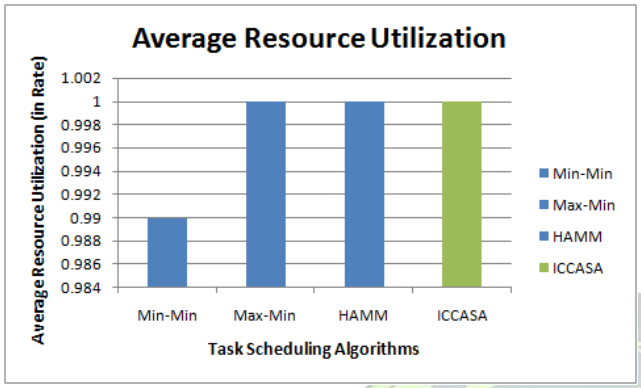

Figure 3. Simulation Result in terms of Average Resource Utilization

c) Load balancing: From the Table4 and Figure 4, the proposed (ICCASA) algorithm and the existing HAMM, Max-Min achieved same degree in terms load balancing for the illustrated example.

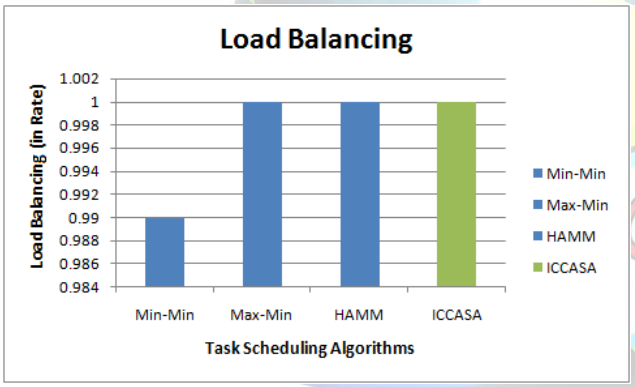

Figure 4. Simulation Result in terms of Load Balancing

d) Total computation cost: The total computation cost and total virtual machine cost is shown in Table 5 and Figure 5.

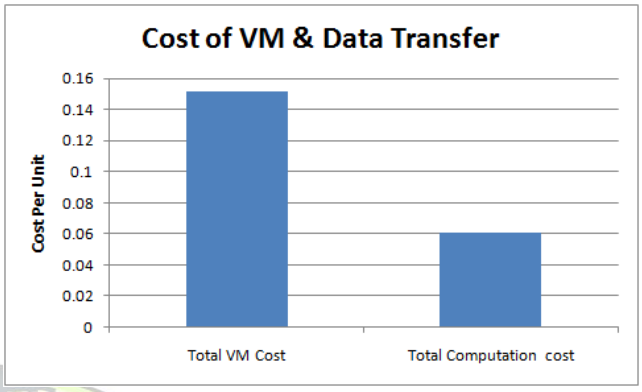

Figure 5. Simulation Result in terms of Cost

Table 4. Simulation results compared with existing algorithms

Parameters	Task Scheduling Algorithms			
	Min-Min	Max-Min	HAMM	ICCASA (Proposed)
Makespan	45.87	45.27	44.74	33.666
Average Resource Utilization	0.99	1.00	1.00	1.00
Load Balancing	0.99	1.00	1.00	1.00

Table 5. Simulation results in terms of cost in ICCASA

Total Virtual Machine Cost (per unit)	0.15145
Total Computation Cost (per unit)	0.06058

From the above simulation results, the proposed ICCASA Algorithm is better than the existing HAMM, Max-Min and Min-Min algorithms.

VI. CONCLUSION

One of the main concern in cloud computing is task scheduling of incoming user's tasks among the cloud resources to achieve better performance of cloud system. In this research work, Improved Credit Based Cost Aware Scheduling Algorithm (ICCASA) is proposed which uses the advantages of both the Min-Min and Max-Min algorithms by applying it consecutively. The proposed model fixes priority to the task based on credits, before task scheduling. Here, task priority is assigned by using a multiple credits based priority system that considers four parameters: Task length, Task priority, Deadline of the task and Cost of the task, which overcomes the same priority issue during task scheduling. The proposed algorithm is implemented on CloudSim 3.0.3, a Cloud Simulation Tool. Simulation result

achieves maximum resource utilization, efficient load balancing, minimum makespan and minimum total execution cost when compared to the existing algorithms. The study can be further extended by applying some other heuristic algorithm on actual cloud computing environment and considering additional factors & parameters.

REFERENCES

[1] S. Vaaheedha Kfatheen, "Mean-Min task scheduling algorithm for cloud environment", Malaya Journal of Matematik (MJM), Vol. 9, No. 1, 2021.

[2] Pradeep Krishnadoss, Gobalakrishnan Natesan and Javid Ali, "CCSA: Hybrid Cuckoo Crow Search Algorithm for Task Scheduling in Cloud Computing", International Journal of Intelligent Engineering and Systems, Vol. 14, No. 4, May 2021.

[3] Fatemeh Ebadifard, Seyed Morteza Babamir and Sedighe Barani, "A Dynamic Task Scheduling Algorithm Improved by Load Balancing in Cloud Computing", 6th International Conference on Web Research, June 2020.

[4] Raja Masadeh, Ahmad Sharieh and Basel A. Mahafzah, "Humpback Whale Optimization Algorithm Based on Vocal Behavior for Task Scheduling in Cloud Computing", International Journal of Advanced Science and Technology, Vol. 13, Issue. 3, 2019.

[5] Pandaba Pradhan, Prafulla Ku. Behera and B. N. B. Ray, "Efficient Min Min Algorithm for Resource Allocation in Cloud Computing", International Journal for Research in Engineering Application & Management (IJREAM), Vol. 06, Issue. 08, November 2020.

[6] J. Y. Maipan-uku, A. Abdulganiyu, A. Abdulkadir and A. Mishra, "An Extended Min-Min Scheduling Algorithm in Cloud Computing", 2nd International Conference on Information and Communication Technology and Its Applications, pp. 472-476, September 2018.

[7] Tran Cong Hung, Phan Thanh Hy and Le Ngoc Hieu and Nguyen Xuan Phi, "MMSIA: Improved Max-Min Scheduling Algorithm for Load Balancing on Cloud Computing", Proceedings of the 3rd international conference on machine learning and soft computing, January 2019.

[8] Shilpa Kodli and Sujata Terdal, "Hybrid Max-Min Genetic Algorithm for Load Balancing and Task Scheduling in Cloud Environment", International Journal of Intelligent Engineering and Systems, Vol. 14, No. 1, 2021.

[9] Ibrahim A. Thiyeb and Sharaf A. Alhomdy, "HAMM: A Hybrid Algorithm of Min-Min and Max-Min Task Scheduling Algorithms in Cloud Computing", International Journal of Recent Technology and Engineering (IJRTE), Vol. 9, Issue. 4, November 2020.

[10] Khaldun Ibraheem Arif, "A Hybrid Min Min& Round Robin Approach for Task Scheduling in Cloud Computing", International Journal of Control and Automation Vol. 13, No. 1, 2020.

[11] P Praveen Kumar and V Kanchana Devi, "Efficient load balancing by optimized flexi max-min algorithm", National Science, Engineering and Technology Conference (NCSET), 2020.

[12] T. Manoranjitham and Dupati Srikar, "Efficient Task Scheduling for Quality of Service in Cloud Computing Network", International Journal of Recent Technology and Engineering (IJRTE), Vol. 8, Issue. 5, January 2020.

[13] D. I. George Amalarethinam and S. Kavitha, "Rescheduling Enhanced Min-Min (REMM) Algorithm for Meta-task Scheduling in Cloud Computing", Springer, pp. 895–902, January 2019.

[14] Syed Arshad Ali, Samiya Khan and Mansaf Alam, "Resource-Aware Min-Min (RAMM) Algorithm for Resource Allocation in Cloud Computing Environment", International Journal of Recent Technology and Engineering (IJRTE), Vol. 8, Issue. 3, September 2019.

[15] Fale Mantim Innocent, Sitlong Nengak Iliya and Ramson Emmanuel Nannim, "An Optimized Flexi Max-Min Scheduling Algorithm for Efficient Load Balancing on a Cloud", International Journal of Scientific & Engineering Research, Vol. 9, Issue. 7, July 2018.