Received Date: 31st May, 2024
Revision Date: 12th June, 2024
Accepted Date: 8th July, 2024

Task Scheduling Optimization in the Cloud Using Improved Heuristic Algorithm

Ritu Bajracharya [1*], Subarna Shakya [2]

[1] Lecturer, Dept of Computer Engineering, Kathmandu Engineering College, Email: ritu.bajracharya@kecktm.edu.np
[2] Professor, Dept of Computer & Electronics Engineering, Pulchowk Campus, Institute of Engineering, Tribhuvan University, Email: drss@ioe.edu.np

***Abstract*— Cloud Computing has become the most efficient and reliable technology in today's era. Almost every organization and individual depend upon this technology to perform their task and even for storage purpose. As the number of users is growing, the complexity of this technology has also increased massively. Thus, for reliable and efficient use of cloud technology, the tasks, infrastructures, and load must be balanced in the system. Among different methods, one of the ways to efficiently manage the complexity of the system is task scheduling. Task scheduling helps to optimize CPU utilization and make the tasks done with minimum loss. Also, there are many task-scheduling algorithms which have been proposed and implemented to date. Every algorithm has its pros and cons too. Thus, this project aims to implement the proposed improved heuristic (B-Sufferage) Algorithm to schedule tasks in a cloud environment and compare the result with the existing PSO and Min-Min Task Scheduling Algorithm. The B-Sufferage Algorithm depends upon the sufferage value to schedule a particular task on a particular VM. The required infrastructure has been set up using CloudSim 3.0.3 and the implementation has been carried out by configuring respective algorithms. As a result, it has been found that the B-Sufferage algorithm in task scheduling works better than the existing one. Thus, the result has been compared based on metrics like makespan, resource utilization, turn-around time, and waiting time where there is a significant difference for scheduling tasks using this B-Sufferage; an improved heuristic algorithm.**

Keywords*— *Cloud environment, Task Scheduling, B-Sufferage Algorithm, improved Heuristic, Virtual Machines, cloudlets

Introduction

Cloud Computing is a growing technology that provides services to users on demand via the internet. It provides the required software, hardware, and other infrastructure to user without letting the user buy any of them. Users only need to pay for the used services. The computing resources include applications, servers (physical or virtual), data storage, development tools, networking capabilities, and many more which are managed by the third party known as cloud service providers (CSPs). The CSPs make these resources available to users and bill them according to usage. Cloud is one of the indulging technologies in today's era. From individual users to sophisticated enterprises organizations use the cloud for data storage and other utilities as the cloud provides services anytime anywhere and any place. Hence, there are three types of cloud deployment models:

* Corresponding Author

a) Public clouds: These clouds are run by the Cloud Services Providers and they offer computing, storage, and network resources over the internet to any user on demand.

b) Private clouds: They are built and managed by a single organization and privately hosted in their data centers used by the organization only. Private cloud provides better control, security, and management of data through a shared pool of resources.

c) Hybrid clouds: Hybrid clouds are considered to be the combination of public and private clouds which leverages the advantages and disadvantages of both clouds. Similarly, cloud services can be divided into three broad categories:

i. Software as a service (SaaS): Cloud Service Providers provide different software applications to be used by the user which can considered as the software as a service provided by the cloud. These applications are accessed via different interfaces such as web browsers and mobile applications. One need not download and install the applications on their device, they can be used online. Web-based email is one of the typical examples of SaaS as it allows to send and receive email without having to manage feature upgrades to the email and maintain servers.

ii. Platform as a service (PaaS): In platform as a service (PaaS) provided by the cloud, the consumer can deploy cloud infrastructure on its applications. The user has control over deployed applications and can configure the environment of application hosting. "Aneka" is one of the services provided by the cloud as PaaS.

iii. Infrastructure as a service (IaaS): With Infrastructure as a Service (IaaS) clients can rent the storage, processing power, network, and other computing resources. The user has access and control over databases, OS, and applications deployed. IaaS provides required hardware and other infrastructure without having the business firm purchase them on their own. IaaS has services like automation, dynamic scalability, desktop virtualization, and policy-based services. Amazon Web Services (AWS) is a widely used IaaS which provides a virtual environment to work on.

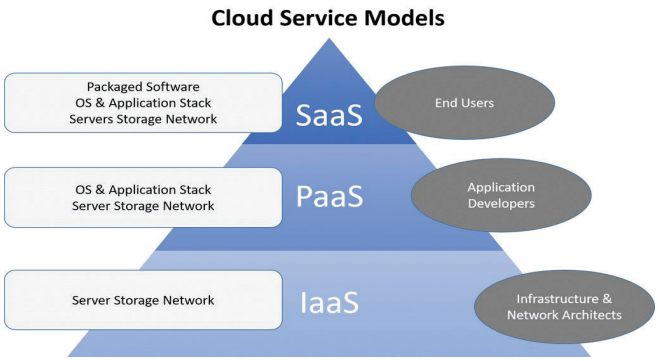

Fig.1. Cloud Service Models

Cloud Computing is emerging as a vital backbone for the varieties of internet businesses using the principle of virtualization [4]. Cloud provides virtualization by separating the service from underlying hardware which was developed during the mainframe era. With Virtualization, the cloud provides the facility of multiple Operating Systems and applications to run on the same machine at the same time. It is accomplished by merging computation, memory, and storage resources.

As the use of cloud computing has been increasing day by day, load balancing has become one of the major challenges in it. Throughout the cloud, always distributed solution is always required as it is not practically feasible or cost-efficient to maintain one or more idle services just to fulfill the required demands. Hence, to overcome the problems related to load balancing, task scheduling procedures to virtual machines could be helpful.

The cloud computing environment includes various virtual machines (VMs) that handle different types of tasks assigned to it. Load Balancing, which is one of the challenging agendas for the cloud can be resolved using the task scheduling algorithms assigned to different VMs. Task scheduling is the process of assigning various jobs/ tasks to virtual machines and resources available. A task may have different constraints such as deadline, computational time and power, priority, and so on which we need to consider while scheduling a task to VMs. Thus, the proper task scheduling algorithm has optimal usage of resources and high performance.

Motivation

Different task-scheduling algorithms in the cloud have been developed which works on different scheduling criteria. However, these scheduling algorithms have limitations rendering maximum scheduling time, computation complexity, and delay whereas many of the algorithms have the constraints task resource requirements, CPU memory, execution time, and execution cost as a single constraint. Also, these traditional algorithms suffer in increased turnaround and waiting time of a task facing difficulties like Starvation and uneven distribution of hindrance on nodes. The load balancing would be much easier if we consider multi-objective task scheduling and more constraints.

A lot of factors have already been covered in the area of task scheduling taking execution time, cost, response time, flow time, and throughput into account but improvement is still required in some areas like makespan, execution cost, and time and space complexity.

I. LITERATURE REVIEW

Many task-scheduling algorithms have been implemented in cloud computing environments. At initial times, the task scheduling algorithms included a First Come First Served (FCFS) basis, Shortest Execution Time First, Round Robin algorithms, and many more. Each of the algorithms has its advantages and disadvantages. Various optimized and hybrid algorithms have been developed till the present as task scheduling is the critical criterion to be considered in cloud computing and it is very difficult to balance load between each virtual machine.

Shree Laxmi and S. Sindhu used a multi-objective PSO-based task scheduling algorithm with the motive of reducing makespan time, cost of communication, and completing the task within the deadline. As a result, the proposed algorithm helped in the distribution of incoming traffic in an efficient way among backend servers [4].

Wang and Yu introduced a task scheduling algorithm based on an improved min-min algorithm which as a result maximized the efficiency of the cloud environment. The improved min-min algorithm had a load balance and high reliability towards the cloud [6].

Later different task scheduling algorithms like genetic algorithm, heuristic sufferage algorithm, Ant Colony Optimization, Grass Hopper Optimization Algorithm, and many more have been developed. Various comparative analyses and studies have also been carried out in later papers.

Keivani and Tapamo reviewed different task-scheduling algorithms in their paper. The comparison between the algorithms resulted in various pros and cons of each algorithm based on the execution time, cost, QoS, and CPU efficiency where the comparison was done between fifteen different algorithms [8].

Similarly, Xin and Zhang also reviewed different algorithms in their paper where task scheduling based on a single problem and task scheduling based on multi-objective optimization were discussed which concluded on further improvement on existing task scheduling algorithms [7].

Many of these comparisons led to development and improvement in the existing algorithm and even new optimized algorithms have also been introduced.

Yong Shi in his paper proposed an improved sufferage algorithm known as B-Sufferage Algorithm which

considered load balancing as a major constraint. The paper compared the performance analysis concerning turnaround time and throughput between the sufferage algorithm and the proposed algorithm which led to improved results [1].

M. Hussain proposed an Energy-efficient task scheduling algorithm in his paper the result resulted in a reduction of energy consumption [2]. He also proposed a Task reassignment algorithm which ultimately ensured that the tasks were completed more quickly. Hence, this was applicable in a heterogeneous environment.

D.I George Amalarethinam and S. Kavitha used a rescheduling enhanced min-min (REMM) Algorithm for Meta-task Scheduling in the cloud which is based on makespan and resource utilization. The result concluded that the LBMM algorithm somehow eliminated the limitations of the Min-Min meta task algorithm but REMM has better performance than LBMM as it is concerned about resources and run time of the task [3].

Every result and analysis has been done using CloudSim Simulator. It can handle large-scale platforms and combined simulation of both private and public domains [9]. It provides various packages for setting up the required infrastructure, initializing the parameters, and calculating of required performance of each task.

Research Methodology

A. *SetUp*

The basic method for scheduling tasks involves the division of tasks among the virtual machines that has been considered for operation in a cloud environment. For the experiment, a set of data would be distributed among the VMs. The data size required has been formulated manually and fits into the given set of cloudlets. The sizes vary from 100 MB to 10000 MB.

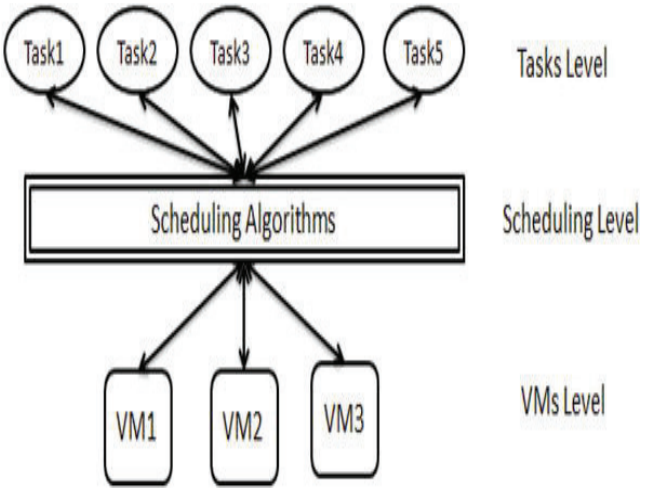

Fig. 2. Scenario of Task Scheduling in Cloud System

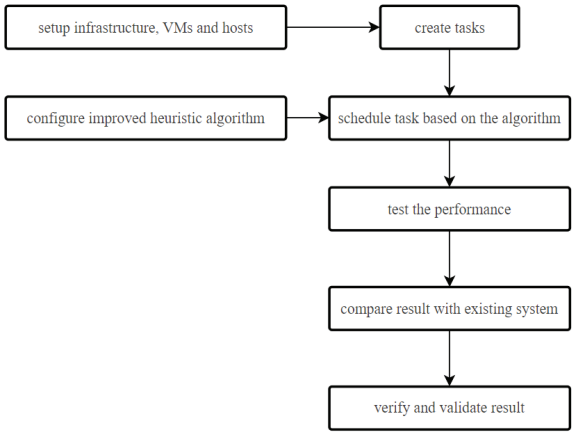

Fig.3. System Block Diagram

Initially, required infrastructures, virtual machines and hosts have been setup for the process. Then, the proposed algorithm has been configured using Java programming language. Required tasks basically known as cloudlets has been created and scheduled based on PSO, Min-Min and B-Sufferage algorithms.

B. *Algorithms*

i. Particle Swarm Optimization:

Particle swarm optimization (PSO) is an important intelligent algorithm for solving the task scheduling problem in cloud computing. It is the meta heuristic algorithm inspired by swarm behavior observed in nature such as fish and bird. In the PSO algorithm, the individual are particles that move in the population space. Each particle has two parameters: the current position x_i and the current velocity v_i, each having fitness value. Each particle keeps track of the particle_best_Fitness_value, particle_best_Fitness_position.

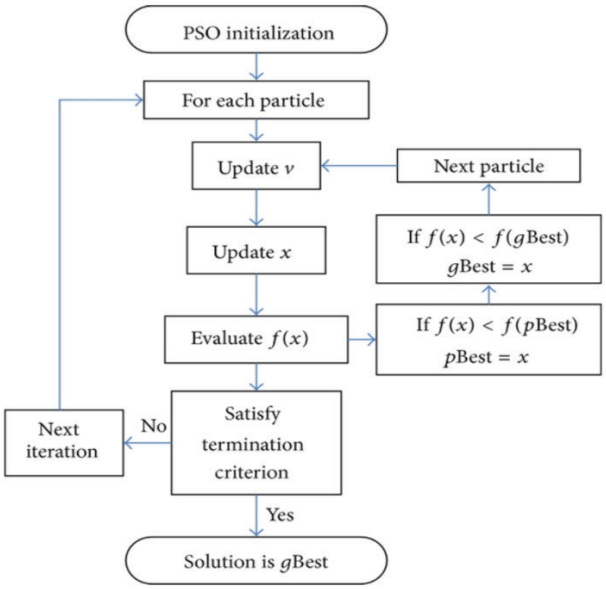

Fig.4. Flowchart of PSO Algorithm

ii. Min-Min Heuristic Algorithm:

Min-Min algorithm in task scheduling is one of the heuristic algorithms which assigns small tasks to the resources and perform task with minimum execution time.

Pseudo-Code for Min-Min Algorithm:

for i=1 to M //Mdenotes number of tasks to be scheduled

for j= 1 to N //N denotes the number of VMs

Cij=Eij+Rj (1)

Cij denotes completion time

Eij denotes execution time

Rij denotes ready time of task i on VM j

end for

end for

do until all unscheduled tasks are exhausted

for each unscheduled task

find minimum completion time of the task and VM that obtains it.

end for

find task tp with earliest completion time and assign task tp to the VM that gives minimum completion time

delete task tp from pull of unscheduled tasks

update the ready time of the machine that gives the minimum completion time

end do

iii. B-Sufferage Algorithm:

It is an extension of the traditional Sufferage Algorithm. Initially, we calculate the execution time for each virtual machine and record availability time and hence compute the complete computation time of the task to be completed on a virtual machine. The algorithm is performed iteratively. In each iteration, for each task t_i, the completion time on all VMs is sorted and three minimum values are calculated. Also, the standard deviation is calculated.

$$s_i = (2nd\_min - min) * (3rd\_min - min) * \sigma (Complete_{ij} - min) \quad (2)$$

In each iteration, task T_q with largest value of s_i is chosen which is assigned to VM with earliest completion time. At final stage of iteration, the available time of VM is updated as new task T_p is accepted by that particular VM and hence the completion time of all remaining tasks are updated accordingly.

Pseudo-Code:

for i=1 to M //M is number of tasks to be scheduled

for j=1 to N // N is total number of VMs

calculate E_{ij} //Execution time for task t_i in VM_j

record $Avail_{ij}$ //Availability time of VMs

$C_{ij}=E_{ij}+R_{ij}$ // C_{ij} is completion time,

// R_{ij} is ready time

end for

end for

do until all unscheduled tasks are exhausted

for each unscheduled task t*

find min, 2^{nd}_min and 3^{rd}_min completion time

find the standard deviation of (C_{ij}-min)

calculate:

$s_i=(2^{nd}\_min-min) *(3^{rd}\_min-min) *s(Cij-min)$ (3)

find task t_q with the largest value of s_i and assign it to VM_q with the earliest completion time.

Delete task t_q from a pool of unscheduled task.

end for

update C

update Avail

end do

This improved heuristic algorithm here considers three minimum completion times of task T and also calculates the standard deviation. The standard deviation thus calculated acts as the quantifier that helps to determine which tasks are brokered to which virtual machines.

C. *Performance Metrics*

The performance metrics that has been considered are makespan, total resource utilization and total processing cost.

- Makespan:

Makespan is defined as the maximum time elapsed for completing all tasks.

$$makespan = \max (FT(Ti; VMj)) \quad (4)$$

where, FT (Ti; VMj) is the Completion time of task Ti on Virtual Machine VMj.

- Resource Utilization:

It is considered to be the utilization of Processing Elements in VMs.

$$RUavg = SRUi/N \quad (5)$$

where, RUi is Resource Utilization of VMi and N is total number of VMs used.

Also,

RUi = (CPU Utilization+Bandwidth Utilization+RAM Utilization)/3 (6)

- Total Processing Cost:

It is defined as the total cost required to complete all tasks starting from initialization of infrastructures to the completion of all tasks in the infrastructure.

- Turn Around Time:

Turnaround time (TAT) is defined as the difference between the time of submission of a process to the time of the completion of the process. It is calculated by:

TAT_i=Finish Time $(FT)_i$-Submission time of task T_i (7)

- Waiting Time (WT):

It can be defined as the time spent by a process waiting in the ready queue. It can be calculated by:

$WT_i = TAT_i$ – Execution Time of task T_i (8)

D. *Infrastructure Setup:*

For implementation and analysis of the mentioned algorithm; the infrastructure setup includes the configuration of data centers, VMs, processing elements, and a total number of cloudlets (tasks). The "x86" architecture of the data center has been used with the Linux operating system. The host with respective ID has been created with 2048 MB RAM, 1000000 storage, and 10000 bandwidths.

Also, five virtual machines with memory of 512 MB have been created with mips ranging from 100 to 800.

Once the infrastructure was set up, cloudlets ranging from 30 to 60 and fed to VMs for execution. The data for the respective cloudlet has been created manually and fits into the list.

CloudSim 3.0.3 was used as a simulator for the configuration of machines and cloudlets and was deployed in NetBeans 8.0 using Java Programming Language.

II. RESULT, ANALYSIS AND COMPARISON

TABLE I.

OUTPUT OF TOTAL MAKESPAN

Algorithm	MakeSpan		
	cloudlet=30	cloudlet=40	cloudlet=50
PSO	3356.95	2201.218	3002.543
Min-Min	381	486	500
B-Sufferage	171	499	504

TABLE II.

OUTPUT OF TOTAL RESOURCE UTILIZATION

Algorithm	Resource Utilization		
	cloudlet=30	cloudlet=40	cloudlet=50
PSO	30	40	50
Min-Min	29	39	49
B-Sufferage	30	40	50

TABLE III.

OUTPUT OF TOTAL TURN AROUND TIME

Algorithm	Turn Around Time		
	cloudlet=30	cloudlet=40	cloudlet=50
PSO	428216.6	556011.59	983713.08
Min-Min	3293.399	5953.148	6545.001
B-Sufferage	2096.4374	3896.5936	4269.937

TABLE IV.

OUTPUT OF TOTAL WAITING TIME

Algorithm	Average Waiting Time		
	cloudlet=30	cloudlet=40	cloudlet=50
PSO	3579.865	2611.1267	2411.7365
Min-Min	109.7833	148.731	130.802
B-Sufferage	69.7812	97.31	85.298

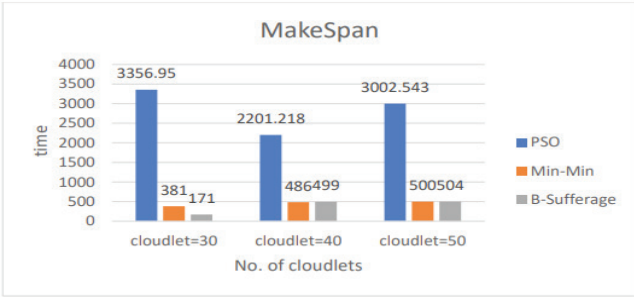

Fig.5.Comparison of makespan

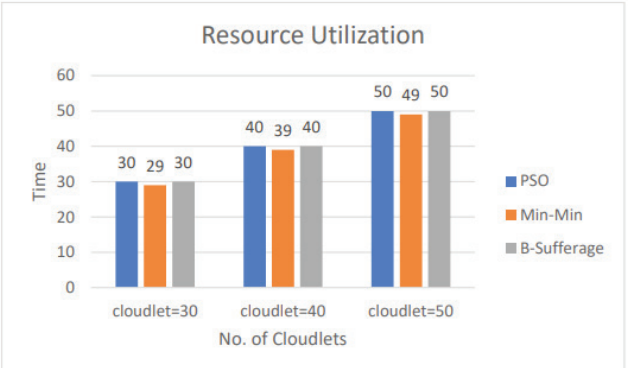

Fig. 6. Comparison of resource utilization

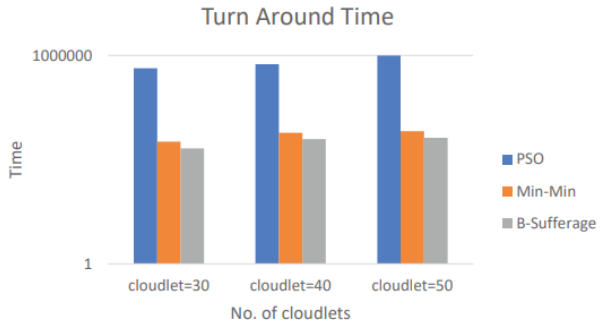

Fig.7. Comparison of turnaround time

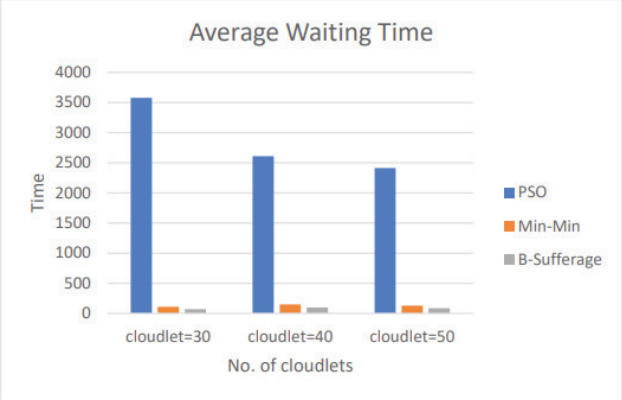

Fig.8. Comparison of average waiting time

From the obtained table and chart, we can find that for a total number of 30,40, and 50 tasks, makespan, resource utilization, total Turn Around Time, and average Waiting Time for three algorithms - PSO, Min-min and B-Sufferage has been observed and analyzed.

The makespan for PSO is greater than the greatest and for Min-min is the smallest. Similarly, the performance of the proposed B-Sufferage Algorithm is better than the other two in terms of total Turn Around Time and Average Waiting time. Similarly, the resource utilization for the min-min algorithm differs by one unit from other two whereas the case is the same for B-Sufferage and PSO Algorithms.

Conclusion

Hence, as the project aims to have a comparative analysis and study among the existing task scheduling algorithms and the proposed B-Suffrage algorithm, it can be finally concluded that the proposed algorithm has better performance in terms of total Turn Around Time and Waiting Time which are major metrics to be considered especially in task scheduling case. B-Suffrage has both pros and cons but the advantageous factor is higher as we can find minor differences in makespan and resource utilization and higher on TAT and WT.

Various task-scheduling algorithms have been proposed to date. Each algorithm has its key factor to be considered for scheduling purposes. In the case of the proposed B-Suffrage Algorithm by Yong Si [1] the major key value was the suffrage value that has been considered for scheduling tasks. As suffrage value, the first minimum, second minimum, and third minimum value for completing tasks on each VM was found out and related standard deviation was also calculated as given in the algorithm above. Hence, based on this sufferage value, the tasks were scheduled on five virtual machines. Finally, the completion time, execution time, and resource allocation time were calculated and evaluated based on the mentioned metrics above.

Acknowledgment

The authors would like to thank the Department of Computer and Electronics Engineering, Institute of Engineering (IOE) for their support and guidance

References

[1] K.S and Yong Shi, "Towards Optimizing Cloud Scheduling Process in Cloud Environment," IEEE, Marietta, USA, 2021.

[2] L.-F. W and Mehboob Hussain, "Energy and performance efficient task scheduling in heterogeneous virtualized cloud computing," Elsevier, China, 2021.

[3] D.I George Amalarethinam and S. Kavitha, "Rescheduling Enhanced Min-Min (REMM) Algorithm for Meta Task Scheduling in Cloud Computing," Springer Nature, Switzerland, 2019.

[4] Sindhu ShreeLaxmi, "Multi-Objective PSO Based Task Scheduling- A Load Balancing Approach In Cloud," in *International Conference on Innovations in Information and Communication Technology (ICIICT), 2019*, Chennai, India, 2019.

[5] V. Gajera, Shubham, R. Gupta and P. K. Jana,, "An effective Multi-Objective task scheduling algorithm using Min-Max normalization in cloud computing," in *2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCAT)*, Bangalore, India, 2016.

[6] Wang, G., & Yu, H. C., "Task Scheduling Algorithm Based on Improved Min-Min Algorithm in Cloud Computing Environment," Scientific.Net, Chicago, 2013.

[7] Zhang, Fengjun Xin, and Lina, "The Review of Task Scheduling," Springer, Singapore, 2019.

[8] Keivani, Arghavan, and Tapamo, "Task Scheduling in Cloud Computing: A Review," IEEE, South Africa, 2019.

[9] Shafiq, Dalia Abdulkareem Shafiq, "CloudSim 3.0.3",2021

[10] S.M. Muzammal, R.K Murugesan and N.Z. Jhanjhi, "A Comprehensive Review on Secure Routing in Internet of Things: Mitigation Methods and Trust-based Approaches,"in IEEE Journal, doi: 10.1109/JIOT.2020.3031162