International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Integrated Load Balancing Process in Cloud Environment using Bja Technique

[1]T Kannadasan, [2]Dr. R. Pragaladan, [3]N. Sureshbabu

Submitted: 04/02/2024 **Revised**: 12/03/2024 **Accepted**: 18/03/2024

Abstract: Cloud computing is a parallel and distributed computing system which comprises a network of interconnected and virtual computers. Various computing tasks are performed in the cloud environment due to the growing benefits and demands of cloud computing platform. However, task scheduling (TS) is the most critical issue in this system that directly affects the utilization of resources from the cloud. Due to the enormous impact on both front and back end, load balancing (LB) scheduling is undoubtedly a key component that must be studied in the cloud research sector. When a proper load balance is accomplished in the cloud, good resource utilization will be achieved as well. Effective load balancing entails evenly spreading the supplied workload across cloud virtual machines (VMs), resulting in great resource utilization and user satisfaction. Thus, this paper proposes a Balanced Job Allocation (BJA) task scheduling algorithm for task scheduling and load balancing. Taking consideration of parameters like response time, makespan, resource utilization, and service reliability, the proposed approach aims to optimize resources and improve the load balance.

Key words: Cloud computing, scheduling algorithm, Load balancing, Cloud networks and Virtual machine.

I. INTRODUCTION

Cloud computing, sometimes known as clouds, has exploded in popularity in the corporate and scientific worlds in recent years.It has gained popularity as a result of recent advancements in virtualization technology.Users who do not have sufficient computing power can use cloud services.Multiplexing allows them to obtain economies of scale [1, 2].Cloud is utilized by various consumers because of its excellent reliability performance, ultra-large-scale virtualization, and unique on-demand service mode. In addition, the cloud service has been used in a variety of businesses.The details of the results are shown in Figure 1 from Flexera's Right Scale 2019 State of the Cloud Report.

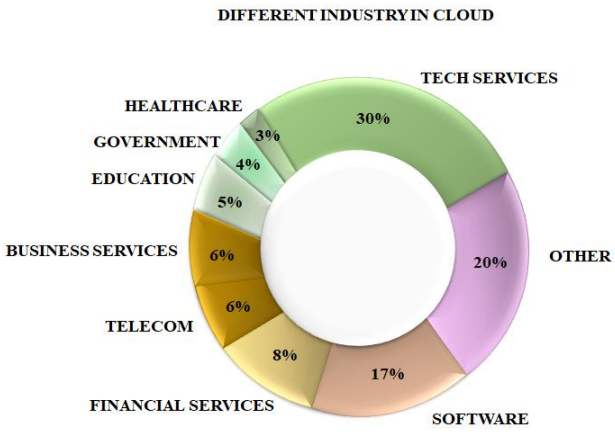


Figure 1 Different industries utilizing cloud (in %)

Cloud computing is used for a wide range of applications.Their resource requirements vary appropriately; some require a lot of storage, while others require CPUs with a lot of processing power; some are data-intensive and require a lot of I/O's.All of these things lead to a load imbalance [3, 4].The computers, network facilities, storage devices and other linked components that make up cloud infrastructure are used to provide cloud computing resources and services to other clients.The majority of these hardware components are found in enterprise data centers.Hard disc driveswith robust storage, solid-state drives, and Multicore servers, as well as network devices as routers, firewall and switches.Other than these hardware components, cloud computing infrastructure also includes software components that gives hand to the cloud service

[1]*Associate Professor of Computer Science, Thanthai Periyar Government Arts and Science College (Autonomous)*
profkannadasant@gmail.com
[2]*Assistant Professor & Head, Department of Computer Science, Sri Vasavi College, Erode,*
pragaladanr@gmail.com
[3]*Assistant Professor, PG and Research Department of Computer Science, Rajah Serfoji Government College, Thanjavur*
sureshbabunagarajan@yahoo.co.in

paradigm, such as virtualization software. Figure 2 represents the simple architecture of cloud computing environment.

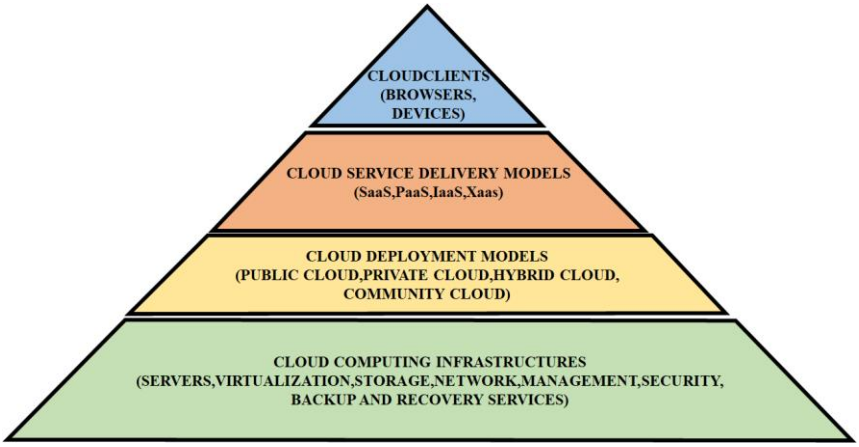


Figure 2 Simple architecture of cloud

There are various service models in a cloud setting, including SaaS, IaaS, PaaS, and XaaS.All of these service models must meet the clients' incoming tasks while adhering to the service level agreements (SLA).We have to build Virtual Machines (VMs) on the cloud to complete these duties, and these VMs are critical in providing the incoming client task [5].Because of limited resources/virtual machines available in cloud computing, efficient resource allocation is essential.In a typical cloud environment, there seem to be two components: the frontend and the backend. The frontend seems to be the user side, and it is accessed via the Internet.The cloud service models where the Data Center stores many physical computers are handled on the backend side which is called servers [6][7][8].

The main goal of this work is to achieve proper task scheduling to avoid load imbalance in the cloud. Section II demonstrates the workflow of the suggested framework. Section III presents the detailed description of the TS and LB algorithm that have been utilized in this work. The performance and accuracy of the algorithm are investigated and resulted in section IV.

II. PROPOSED CONTROL SCHEME

The major purpose of this proposed approach is to enable optimal resource allocation in a cloud system and thereby avoiding imbalanced load in cloud computing applications. This paradigm addresses workload transfer and task rejection difficulties in the cloud [9]. In the proposed study, task scheduling and load balancing techniques are applied to enhance the response time.Figure 3 depicts the architecture of the developed work.

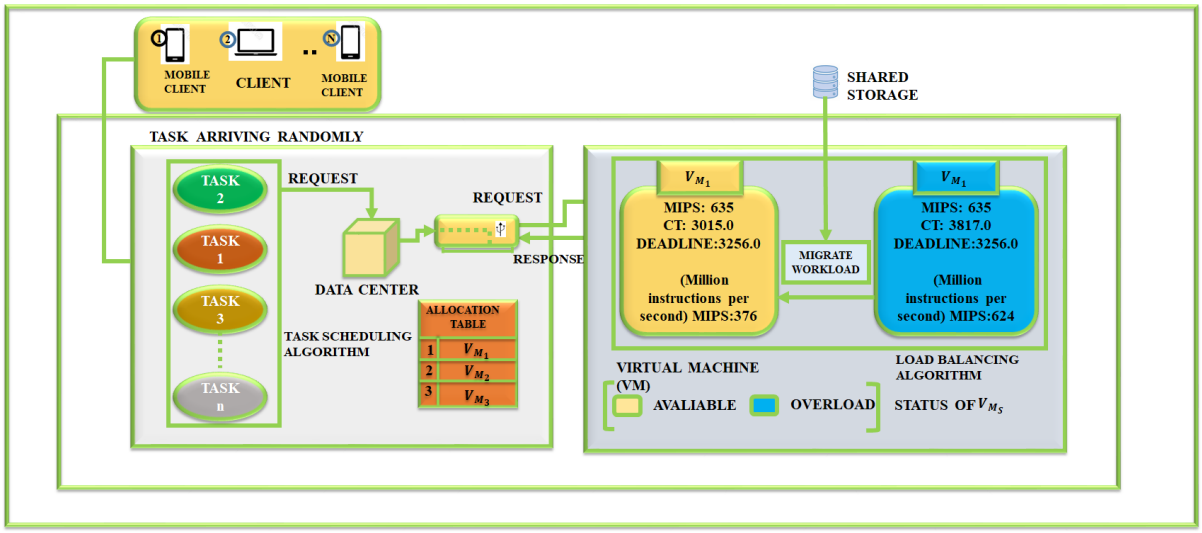


Figure 3 Proposed block diagram

The BJA algorithm is utilized to schedule the incoming task on the basis of their priority. Tasks are prioritized based on the deadline and turnaround time and also they are rescheduled and sorted in order to identify the task

with the highest priority, which is assigned first for the forthcoming processes.The MSC technique is used to categories virtual machines based on the kernel function.To achieve load balance, the WLC algorithm is employed, which takes into account client connectivity and server capacity and then the tasks are subsequently assigned to VMs.

III. MODELING OF THE PROPOSED CONTROL SCHEME

A. Overview of Task Scheduling (TS) and Load Balancing (LB)

Load balancing is considered as the most important approach for ensuring an even distribution of workload as well as effective utilization of resources. It aids in distributing the dynamic workload among the nodes.Load Balancing greatly minimizes the data transmitting and receiving delay in cloud systems.

Task Scheduling is an important goal of load balancing.As the number of user accessing the cloud increases, it leads to the messing up of task scheduling.As a result, Task Scheduling difficulties should be resolved using an efficient algorithm.Task planning refers to the process of efficiently completing tasks in order to fully utilize the system's resources.

Cloud computing service is a crucial part of giant organizations like Google, Amazon, and others.Such services guarantee constant data transfer or streaming flexibility.However, when the number of clients grows, the algorithms driving these necessary activities may become slower, posing obstacles or issues. Therefore, load balancing is an important feature of Cloud Computing technology; without it, clients' tasks may be delayed and responses may take longer.

Load unbalancing difficulties in IaaS clouds environment could be caused by a number of parameters, which are described below.

- There is no accurate or efficient task mapping to appropriate VMs/resources.
- Improper task scheduling
- For heterogeneous (diverse) client's tasks, several task scheduling are required.
- Uneven task distribution to the VMs.

This study intends to address the aforementioned challenges in the IaaS cloud platform by presenting a dynamic TS algorithm that takes into account essential task requirements like deadline and completion time.As QoS considerations, these parameters are critical.The technique produces a balanced workload on the cloud with optimal scheduling and no VM violations.

B. Task Scheduling

The flowchart for the presented design shows that users' incoming tasks are passed straight to the BJA (Dominant Sequence Clustering) algorithm, and the MHEFTalgorithm is assigned to prioritize scheduled jobs. The task with high priority is passed to VM allocation process, which clusters VMs using the MSC algorithm. For load balancing, the WLC algorithm is utilized, which takes into account server capacity and client connectivity. Following that, each VM is assigned a weight and then the tasks are assigned to the VMs with the highest weight value.

Task Clustering

By considering the computing priority the BJA algorithm groups incoming tasks($P(i)$)values based on the tasks' top ($t(i)$) and bottom ($b(i)$) levels.The longest path length from an entry task to i is the top level ($t(i)$) of a task.The longest path length from i to an exit task is defined as the bottom level ($b(i)$).The priority of a task is determined as follows:

$$P(i) = t(i) + b(i) \quad (1)$$

Where,

$P(i)$ - Task priority

$t(i)$ - Top level of the task

$b(i)$ - Bottom level of the task

Figure 4 shows task clustering and scheduling using the BJA technique.For every task, prioritization values are calculated and employed in task clustering, which is followed by task scheduling.

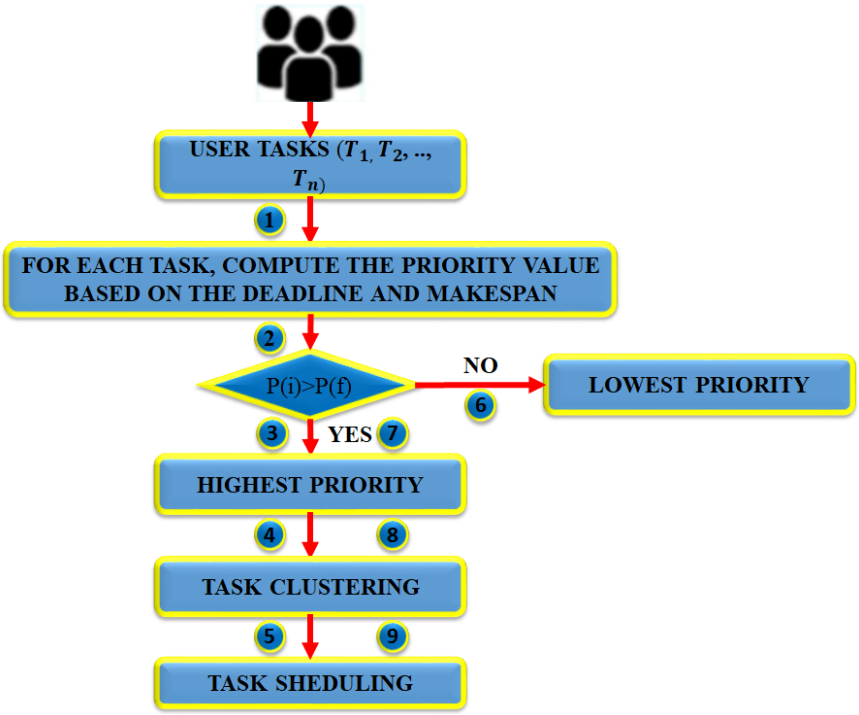


Figure 4.Shows the working Architecture of Task Scheduling Techniques

Figure 5 shows task scheduling on the basis of priority, with cluster 1 denoting top high priority activities and cluster 2 denoting task in next level priority.To prioritize tasks, utilize Equation 1.The BJA algorithm is used to group tasks based on their priorities.Tasks 4 to 7 are prioritized as priority 1 while tasks 1 to 3 are prioritized as priority 2 on the basis of their deadlines and durations.Scheduled tasks have been ranked to determine which work should be passed to the next stages.

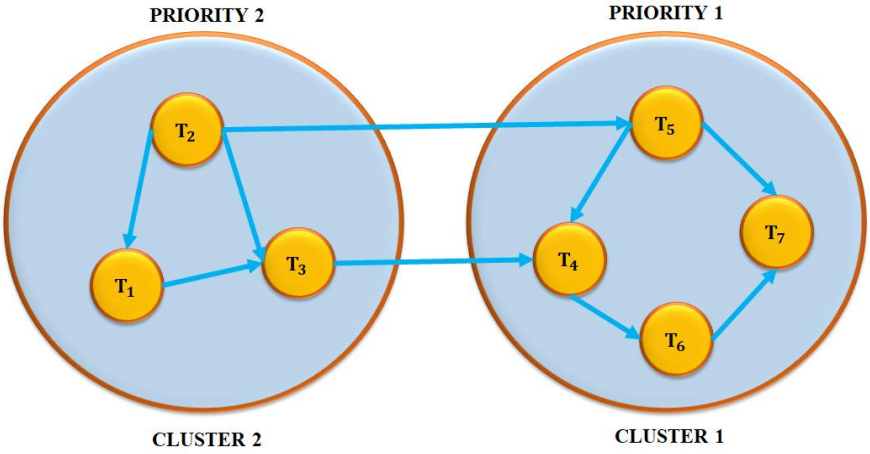


Figure 5 Task clustering based on priority

Task Ranking

To rank scheduled tasks created by the BJA algorithm, the Modified Heterogeneous Earliest Finish Time (MHEFT) algorithmis employed.For forthcoming processes like load balancing and VM clustering, the highly prioritized task among the planned tasks is chosen.To rank tasks, the average execution and transmission time taken for data transfer are computed.The most effective ranking algorithm is MHEFT, a variation of the static list algorithm that prioritizes activities with significant effects on total workflow execution time.

The time arrival has been checked for the tasks with same priority and the task that comes first is assigned as first priority task.$R(t_i)$represents the task's rank and it has been written as,

$$R(t_i) = E(t_i) + \sum_{t_j=s(t_i)}^{max}(C_{i,j} + R(t_j)) \quad (2)$$

In which,

$E(t_i)$ - Average time taken for task execution

$s(t_i)$ - Set of next successive task

$C_{i,j}$ - Communication time that completely dependent to the transfer of data through the nodes i, j.

$R(t_j)$ - Estimation of all its children

Therefore, $C_{i,j}$ can be expressed as,

$$C_{i,j} = y + \frac{data_{i,j}}{b} \quad (3)$$

Where,

y - Average latency

b - Average bandwidth of communication links between the VM

Thus the tasks have been ranked on the basis of equation (2) and (3).The job with high priority is chosen and handed to the VMallocationprocess for further processing.The MHEFT method is used to accomplish expressionranking and task selection for the highest priority task.

C. Load Balancing

For LB, the WLC (Weighted Least Connection) algorithm is utilized.The MSC (Mean Shift Clustering) approach clusters VMs at first, which reduces VM migration.Then the load balance is achieved, and tasks are sent to the best virtual machine to improve response time.

Virtual Machine Clustering

To cluster the VMs, MSC algorithm has been utilized.This non-parametric clustering requires no special knowledge of cluster shape or size, finds the highest density function, and is thus known as a mode-seeking algorithm.It is based on the sliding window technique, which seeks out the most densely packed data points.Itis a centroid-based algorithm whose main goal is to find the group's center points.It works by changing the sliding window's candidate center points.By filtering out nearby copies of candidate windows, the final set of center points and their parallel groups can be constructed.In the MSC method, input is treated as data points.The Kernel $K(x)$ is used to calculate a multivariate kernel density function which is given as,

$$F(x) = \frac{1}{nr^d}\sum_{i=1}^{n} K\left(\frac{x-x_i}{r}\right) \quad (4)$$

In which, n denotes number of data points and$r$ represents the radius of sliding window. The K(x) for radially symmetric function is give as,

$$K(x) = e_{k,d}k(\|x\|^2) \quad (5)$$

Where,$e_{k,d}$ denotes the normalization constant that ensures $K(x)$ is integrated into zero.The modes of density functions are found at the gradient function's zero$\nabla F(x) = 0$.

$$\nabla F(x) = \frac{2e_{k,d}}{nr^{d+2}}\sum_{i=1}^{n}(x - x_i)\, g\left(\left\|\frac{x-x_i}{r}\right\|^2\right) =$$

$$\frac{2e_{k,d}}{nr^{d+2}}\left[g\left(\left\|\frac{x-x_i}{r}\right\|^2\right)\right]\left[\frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x-x_i}{r}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{r}\right\|^2\right)} - x\right] \quad (6)$$

Where,

$$g(s) = -k'(s)$$

The first term is proportional to the density estimation at x computed with kernel $G(x) = e_{k,d}g(\|x\|^2)$.The mean shift vector is a second term which constantly approaches the maximum density increase direction.The mean shift process is realized by doing the following vector computations:Mean shift vector estimation and window translation $x^{t+1} = x^t + M_r(x^t)$.

The mean shift vector,

$$M_r(x^t) = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x-x_i}{r}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{r}\right\|^2\right)} - x \quad (7)$$

Mean shift is a hill-climbing technique in which the kernel shifts to a higher-density zone for each iteration until convergence.Every shift in this method is defined by the mean shift.Each iteration, the kernel reaches the centroid or mean points in it.The total points in the sliding window equals the density in it.The mean computation is determined by the kernel functions chosen.Figure 6 demonstrates clustering using the MSC algorithm, with the kernel function approaching the highest density area in each iteration.To improve response time, VMs are clustered and tasks are optimized.

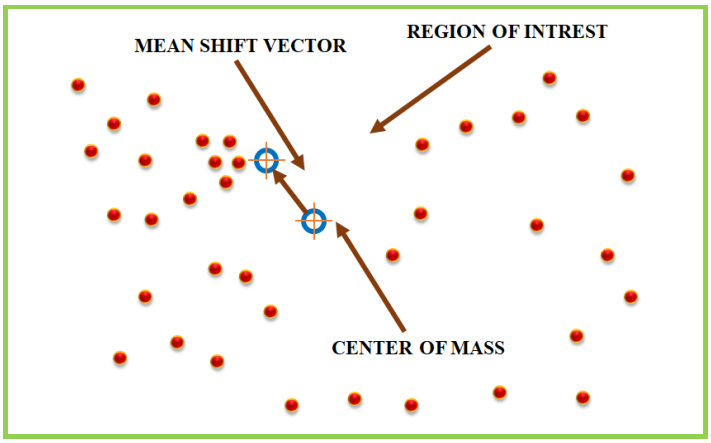


Figure 6 Plot of MSC algorithm

VM Allocation

The WLC assigns a value to each server based on its capacity and client connectivity.The server with the highest weight distributes tasks. The WLC is used to calculate the weight of each virtual machine. The capacity of the server is defined by factors like CPU speed, I/O capacity, memory size and idle rate. Depending upon these values, the weight factor computation for each VM has been performed.

$$G_i = \sqrt{(\rho_m \times H_m \times D_m) + (\rho_c \times S_c \times D_c)} \quad (8)$$

Where,

D_m - Idle rate of node memory

H_m - Memory size

S_c - Speed of CPU in MHz

ρ_m&ρ_c - Proportionality constant (whose summation is always equal to 1)

D_c - Idle rate of node CPU

Therefore, by using Equation 8, the weight of one and all tasks has been estimated. On the hand, the load weight of every VM is derived using the expression given below,

$$L = \frac{N(t)}{E(t)} \quad (9)$$

In which, $N(t)$ denotes the number of task and $E(t)$ represents the task execution time. Each VM's maximum weight and lowest load are taken into account when assigning tasks.The MHEFT method generates the highest priority task, which is then assigned to VMs.We use the abovementioned approach for load balancing and task allocation to enhance the response time of every activity.

IV. RESULTS & DISCUSSIONS

The main goal of this work is to ensure effective resource allocation in cloud system, thereby avoiding uneven load distribution in the cloud environment. Thus the response time for the completion of each task has been improved. Also, the efficiency of the proposed work has been resulted in this section by comparing it with the existing method. The entire work is simulated in a CloudSim 3.0 simulation platform.

Performance Indices

There are four performance indices that are considered to analogize the efficiency of the developed work with the existing approach named as TTSA (Temporal task scheduling algorithm).

Response Time: The time gap between the execution of each task in the system until the completion of entire task is referred as response time.

$$R_t = T_c - T_a + T_t \quad (10)$$

Where,

T_c - Time for completing a task

T_a - Task's arrival time

T_t - Task transfer time

Makespan: Makespan is the overall time it takes to process a collection of tasks from start to finish. Therefore, it has been expressed as,

$$M_a = \max(CT) \quad (11)$$

Where, $\max(CT)$ represents the maximum time taken for task completion.

Resource Utilization: The number of resources required to complete a task is referred to as resource utilization. It is expressed as,

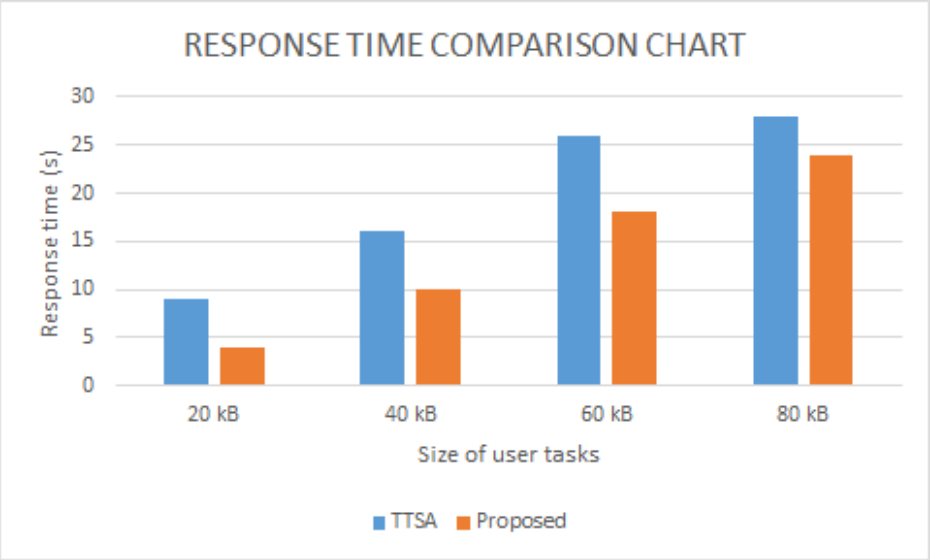
$$R_u = \frac{T_c}{M_a \times N} \quad (12)$$

In which, N refers to the number of resources.

Service Reliability: In cloud technology, service reliability is determined by three factors: accessibility,

continuity, and performance.When a consumer demands service, accessibility relates to the availability of that service.The non-disruption of services over a set period of time is referred to as continuity.The complete fulfilment of a customer's expectation is referred to as performance.The three parameters are considered in our suggested work's service reliability measures.

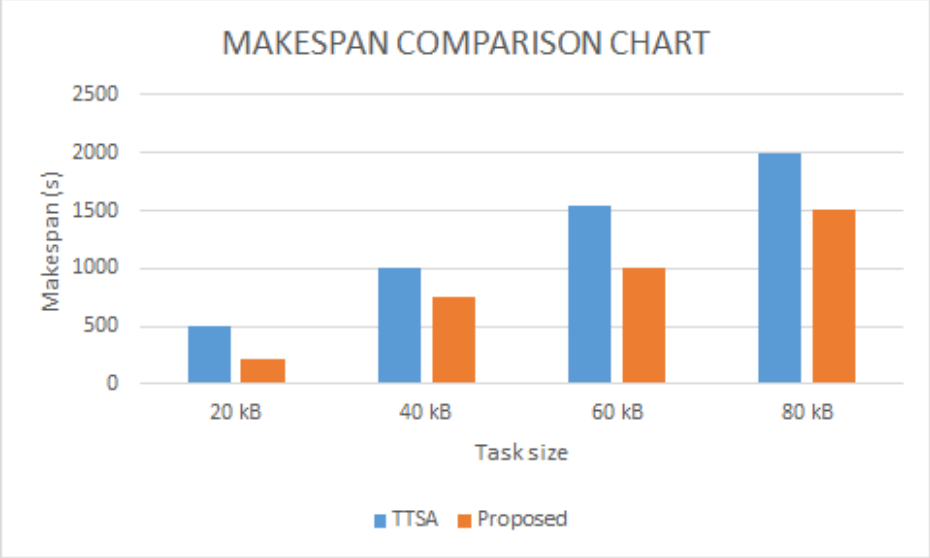
Our proposed framework's major goal is to reduce response time.The response time of our planned work and those of TTSA have been compared.The time that passed between task arrivals and task completion is referred to as response time.The simulation results show that our suggested work has a faster response time than the TTSA approach and the graphical plot representing the aforementioned criteria is portrayed in Figure 7.



Graph.1.shows the Response time comparison vs User Tasks

Therefore, from the Figure 7, it is noticed that our proposed work provides a task response time of 4 seconds for a user with a 20 kB file size, whereas the TTSA method takes 9 seconds.Also, the proposed approach takes 10 seconds to respond to a 40-kb user, whereas TTSA takes 15 seconds.As a result, the proposed approach has a faster reaction time than the existing method, which is based on virtual machine clustering.Thus, the amount of VM migration is decreased, and tasks are assigned and processed quickly, reducing response time.

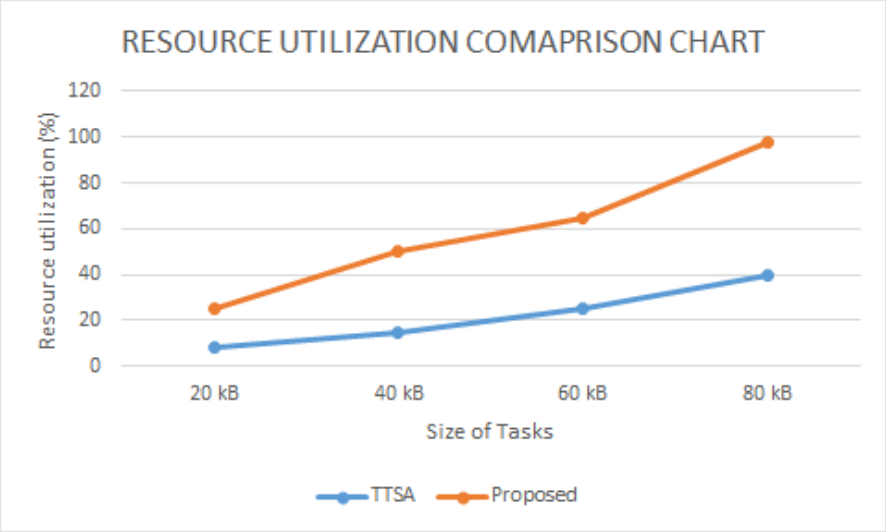
The makespan has also been compared, which is the entire time it takes to complete a set of tasks.Our proposed method has a lower makespan time than TTSA, according to the simulation results.The makespan comparison chart is illustrated in Figure 8.



Graph.2.shows the Makespan comparison vs Tasks size

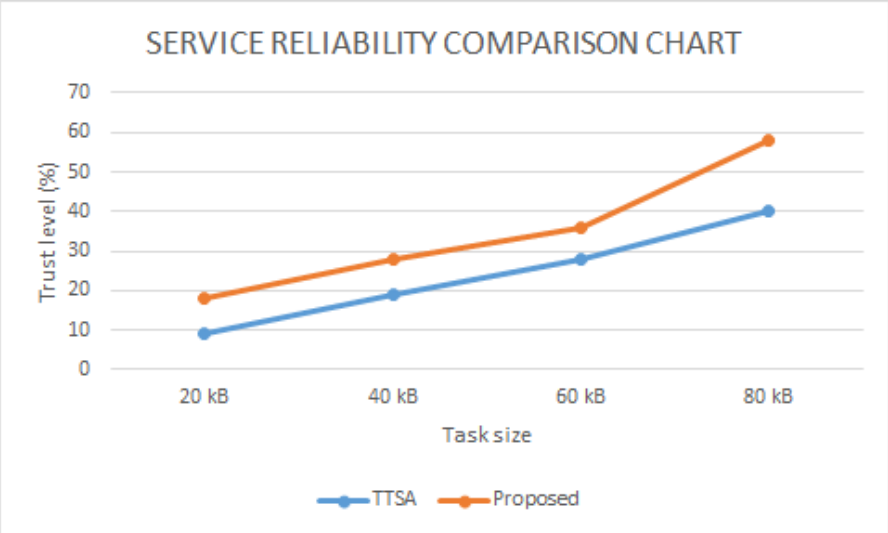
Graph.7.shows the The WLC algorithm aids in balancing the load based on the weight allocated to every VM, provides VMs with a low load and high computational capability for completing many tasks of various sizes.The highest-priority work is assigned first to the VM with the highest weight.This strategy improves the VM's efficiency in doing various activities, reducing the makespan time of our suggested work when compared to TTSA.

Task clustering by using BJA method improves resource utilization.Prioritization is done using deadlines and timelines to categorize each forthcoming work.Figure 9 depicts a resource utilization comparison.



Graph.3.shows Graphical plot representing the comparison of resource utilization

When compared to the TTSA approach, our suggested method improves resource utilization metrics by up to 98 percent by grouping and selecting the highest priority tasks.



Graph.4.shows the Comparison of service reliability

The proposed method's service reliability is improved by utilizing efficient TS and LB. The service reliability is increased by reducing the response time. Therefore, it is clearly figured out in the Figure 10, that the proposed approach has high service reliability than the TTSA approach.

V. CONCLUSION

This paper presented a novel task scheduling and load balancing strategy based on the BJA and MSC algorithms to improve user task response time. Based on the priority, input tasks are clustered by BJA algorithm. Priority values are calculated on the basis of top and bottom levels of tasks and are used to create task clusters for scheduling. The MHEFT algorithm is used to rank scheduled tasks using rank value computation, which takes into account each task's average execution and communication times. For upcoming procedures, the task with the highest priority is chosen and allocated first. The WLC algorithm is utilized to achieve sufficient load balance. The MSC algorithm clusters virtual machines and thereby reducing VM migration. The load balance is

then achieved, and tasks are assigned to the most appropriate virtual machine to enhance response time. In future the work will be improved with the different parameter set of attributes with objective of fault tolerant while the task scheduling in queue in the cluster of high and low bound service request from the users.

REFERENCES

[1] Shanchen Pang;Wenhao Li;Hua He;Zhiguang Shan;Xun Wang, Year: 2019, "An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing", IEEE Access, Vol: 7, pp: 146379 - 146389.

[2] PeiYun Zhang;MengChu Zhou, Year: 2018, "Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy", IEEE Transactions on Automation Science and Engineering, Vol: 15, Issue: 2, pp: 772 - 783.

[3] Liyun Zuo;Shoubin Dong;Lei Shu;Chunsheng Zhu;Guangjie Han, Year: 2018, "A Multiqueue Interlacing Peak Scheduling Method Based on Tasks' Classification in Cloud Computing", IEEE Systems Journal, Vol: 12, Issue: 2, pp: 1518 - 1530.

[4] Shaojin Geng;Di Wu;Penghong Wang;Xingjuan Cai, Year: 2020, "Many-Objective Cloud Task Scheduling", IEEE Access, Vol: 8, pp: 79079 - 79088.

[5] Shridhar G. Domanal;Ram Mohana Reddy Guddeti;Rajkumar Buyya, Year: 2020, "A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment", IEEE Transactions on Services Computing, Vol: 13, Issue: 1, pp: 3 - 15.

[6] Dalia Abdulkareem Shafiq;Noor Zaman Jhanjhi;Azween Abdullah;Mohammed A. Alzain, Year: 2021, "A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications", IEEE Access, Vol: 9, pp: 41731 - 41744.

[7] Niladri Sekhar Dey;T. Gunasekhar, Year: 2019, "A Comprehensive Survey of Load Balancing Strategies Using Hadoop Queue Scheduling and Virtual Machine Migration", IEEE Access, Vol: 7, pp: 92259 - 92284.

[8] Said Nabi;Muhammad Ibrahim;Jose M. Jimenez, Year: 2021, "DRALBA: Dynamic and Resource Aware Load Balanced Scheduling Approach for Cloud Computing", IEEE Access, Vol: 9, pp: 61283 - 61297.

[9] Shudong Wang;Tianyu Zhao;Shanchen Pang, Year: 2020, "Task Scheduling Algorithm Based on Improved Firework Algorithm in Fog Computing", IEEE Access, Vol: 8, pp: 32385 - 32394.

[10] T.Thamaraiselvan "Nadi Aridhal: A Pulse Based Automated Diagnostic System" IEEE 3rd International Conference on Electronics Computer Technology (ICECT 2011), Vol.1.VI-305-208.

[11] Heba Saleh;Heba Nashaat;Walaa Saber;Hany M. Harb, Year: 2019, "IPSO Task Scheduling Algorithm for Large Scale Data in Cloud Computing Environment", IEEE Access, Vol: 7, pp: 5412 - 5420.

[12] Hui Zhao;Qinghua Zheng;Weizhan Zhang;Jing Wang, Year: 2018, "Prediction-Based and Locality-Aware Task Scheduling for Parallelizing Video Transcoding Over Heterogeneous MapReduce Cluster", IEEE Transactions on Circuits and Systems for Video Technology, Vol: 28, Issue: 4, pp: 1009 - 1020.

[13] Wenxiang Li;Chunsheng Zhu;Laurence T. Yang;Lei Shu;Edith C.-H. Ngai;Yajie Ma, Year: 2017, "Subtask Scheduling for Distributed Robots in Cloud Manufacturing", IEEE Systems Journal, Vol: 11, Issue: 2, pp: 941 - 950.

[14] Hashim Ali;Muhammad Shuaib Qureshi;Muhammad Bilal Qureshi;Ayaz Ali Khan;Muhammad Zakarya;Muhammad Fayaz, Year: 2020, "An Energy and Performance Aware Scheduler for Real-Time Tasks in Cloud Datacentres", IEEE Access, Vol: 8, pp: 161288 - 161303.

[15] T.Thamaraiselvan "A Survey on Hybrid Item Dependencies in Association Rule Mining", The International journal of analytical and experimental modal analysis Volume XI, Issue XII, December/2019 p.p: 1244-1249.

[16] Kefeng Deng;Kaijun Ren;Min Zhu;Junqiang Song, Year: 2020, "A Data and Task Co-Scheduling Algorithm for Scientific Cloud Workflows", IEEE Transactions on Cloud Computing, Vol: 8, Issue: 2, pp: 349 - 362.

[17] Muhammad Junaid;Adnan Sohail;Rao Naveed Bin Rais;Adeel Ahmed;Osman Khalid;Imran Ali Khan;Syed Sajid Hussain;Naveed Ejaz, Year: 2020, "Modeling an Optimized Approach for Load Balancing in Cloud", IEEE Access, Vol: 8, pp: 173208 - 173226.

[18] Lei Yu;Liuhua Chen;Zhipeng Cai;Haiying Shen;Yi Liang;Yi Pan, Year: 2020, "Stochastic Load Balancing for Virtual Resource Management in Datacenters", IEEE Transactions on Cloud Computing, Vol: 8, Issue: 2, pp: 459 - 472.

[19] Haiying Shen;Liuhua Chen, Year: 2020, "A Resource Usage Intensity Aware Load Balancing Method for Virtual Machine Migration in Cloud Datacenters", IEEE Transactions on Cloud Computing, Vol: 8, Issue: 1, pp: 17 - 31.

[20] T.Thamaraiselvan,"AN EFFICIENCT CLUSTERING ON HYBRID ITEM DEPENDENCY USING SCFCM AND SVM

TECHNIQUES",DesignEngineering Issue 7,July 2021 P.P:2275-2286.

[21] Songyun Wang;Zhuzhong Qian;Jiabin Yuan;Ilsun You, Year: 2017, "A DVFS Based Energy-Efficient Tasks Scheduling in a Data Center", IEEE Access, Vol: 5, Issue: 2, pp: 13090 - 13102.

[22] PeiYun Zhang;MengChu Zhou, Year: 2018, "Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy", IEEE Transactions on Automation Science and Engineering, Vol: 15, Issue: 2, pp: 772 - 783.

[23] Guisheng Fan;Liqiong Chen;Huiqun Yu;Dongmei Liu, Year: 2020, "Modeling and Analyzing Dynamic Fault-Tolerant Strategy for Deadline Constrained Task Scheduling in Cloud Computing", IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol: 50, Issue: 4, pp: 1260 - 1274.

[24] Xingquan Zuo;Guoxiang Zhang;Wei Tan, Year: 2014, "Self-Adaptive Learning PSO-Based Deadline Constrained Task Scheduling for Hybrid IaaS Cloud", IEEE Transactions on Automation Science and Engineering, Vol: 11, Issue: 2, pp: 564 - 573.

[25] Pengwei Wang;Yinghui Lei;Promise Ricardo Agbedanu;Zhaohui Zhang, Year: 2020, "Makespan-Driven Workflow Scheduling in Clouds Using Immune-Based PSO Algorithm", IEEE Access, Vol: 8, Issue: 2, pp: 29281 - 29290.