International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-06, Issue-04, July 2020

Optimized Load Balancing and Task Scheduling using Dominant Sequence Clustering and Honey Bee Algorithm in Cloud

[1]Arunima V R, [2]Jasmine A, [3]Sherene Antony, [4]Neethu M S

[1,2,3]B. Tech Student, [4]Assistant Professor LBS Institute of Technology for Women, Trivandrum, India. [1]arunimavr05@gmail.com, [2]jasminmaheen003@gmail.com, [3]sherene.antony@gmail.com, [4]neethums.ms@gmail.com

Abstract- The ready availability of the computer systems and its resources, especially data storage and computing power with the least or no direct active supervision by the users contribute to cloud computing. Cloud computing offers services that enable users to access SaaS, PaaS, and IaaS over the internet. Issues in Cloud include security, energy efficiency, load balancing, etc. Load balancing and task scheduling are two challenging issues. Load balancing is the process of distributing the workload to nodes to maximize throughput, reduce response time, etc. On the other hand, task scheduling is the balancing of jobs over the nodes. Load balancing and task scheduling are mandatory in Cloud for efficient resource utilization. To surpass the drawbacks of existing approaches, we have proposed an optimized method that incorporates Dominant Sequence Clustering (DSC) for task scheduling and Honey Bee Optimization (HBO) algorithm for load balancing. The first step of the process is task clustering. The second step is task ranking using the Modified Heterogeneous Earliest Finish Time (MHEFT) algorithm. We have verified the proposed algorithm with existing algorithms and results show improvement in average execution time and significant decrease in waiting time of tasks.

Keywords --Cloud computing; DSC algorithm; HBO algorithm; Load balancing; MHEFT algorithm; task scheduling; VMs.

I. INTRODUCTION

Cloud computing is an internet-based distributed computing that operates as a pay-as-you-go model to support the increased user demand. In recent years, an enormous variety of commercial corporations and organizations are deploying their applications in cloud knowledge centres because of the economy of scale provided by cloud computing. There are several challenges related to the cloud setting. Task programming and load balancing are major challenges that are being featured by cloud adopters.

Scheduling is a balancing process where tasks are scheduled based on the algorithm used and specific requirements. A task is executed like an action that uses the input resources to produce efficient output in the computation nodes. The primary goal is to maximize resource utilization by reducing task execution. Several internal and external requirements vary for each task; bandwidth, storage, response time, etc.

In the cloud, multiple jobs running at the same time demand for different resources. With the increase in consumers, the number of tasks that need to be scheduled increases proportionately. The scheduling algorithm used needs to be effective for interruption handling, fault tolerance, and response time reduction. Tasks are assigned to the processor for execution, resulting in minimum execution time and maximum profit of the cloud owner. Task scheduling thereby assigns tasks to the apt processor. By scheduling tasks efficiently, we can utilize resources well and have economic efficiency.

Load balancing is a strategy that distributes the dynamic local workload uniformly across all the nodes in the entire cloud to avoid a situation where certain nodes are heavily loaded while others are idle or least active. It helps organizations to manage the application or workload demands by allocating resources among multiple computers, networks, or servers. Some of the jobs may be rejected due to overcrowding. Hence various algorithms have been proposed.

In a cloud computing environment, whenever a virtual machine is heavily loaded or overloaded with multiple tasks, these tasks have to be removed and submitted to the underloaded virtual machine to balance the load among machines of the same data centre. Two classifications of load balancing algorithms are static and dynamic. In a static algorithm, the traffic is divided equally between the servers which require an earlier knowledge of system resources, so that the decision of shifting of the load does not depend on the present state of the system. In dynamic algorithms, the lightest server in the whole network or system is looked for and chosen for balancing a load. The present state of the system is making use of to build decisions to handle the load.

II. PROPOSED SYSTEM

In this project, we mainly applied two algorithms: Honey Bee Optimization (HBO) algorithm for load balancing and Dominant Sequence Clustering (DSC) algorithm for task scheduling that mainly aims to do the following: -

- Balancing load of virtual nodes
- Minimizing Response Time (RT)

Task scheduling and load balancing algorithms are employed in our work to enhance response time. The design of the projected work, within which we used the DSC algorithm to schedule tasks based on their priorities, is conferred in Figure. Task priority is based on the deadline and makespan. To determine the highest priority task, each of the tasks is rescheduled and ranked for the upcoming processes. Virtual Machines (VMs) are then clustered.

The flowchart of our projected work indicates that the incoming tasks of users are directly given to the DSC algorithm, and also the changed Heterogeneous Earliest Finish Time (MHEFT) algorithm is employed to rank scheduled tasks. The task with the highest priority is transferred to the VM allocation method. The Optimized_HBO algorithm considers the capacity and client connectivity of servers, is employed to balance loads, and so tasks are allotted to VMs that successively decreases the response time.

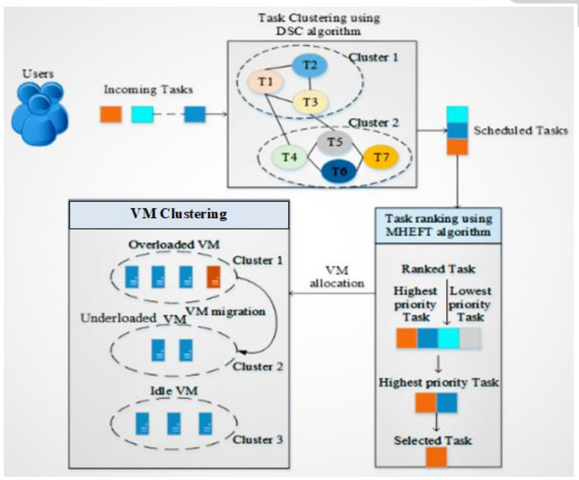


Figure 1. The Architecture of the proposed work

A. TASK SCHEDULING

The DSC algorithm is employed to schedule tasks. The priority of each forthcoming task is taken under consideration for scheduling and clustering.

B. TASK GROUPING

The DSC algorithm clusters forthcoming tasks by computing priority (P(i)) values using the highest (top(i)) and bottom (bot(i)) levels of the tasks. The highest level (top(i)) of a task defines the length of the longest path from entry task to i. The lowest level (bot(i)) is the length of the longest path from i to exit task.

Task priority is computed as:-

$$P(i) = top(i) + bot(i) \qquad (1)$$

where P(i) denotes the priority of task i, top(i) denotes the highest level of task i, and bot(i) denotes the lowest level of task i. The highest and lowest levels of a task are added up to figure out its priority P(i). Task clustering and task scheduling using the DSC algorithm are illustrated in Figure 2. Priority values are computed for each of the tasks and utilised in task clustering, followed by task scheduling. Tasks are clustered on the thought of their priorities via the DSC algorithm. Scheduled tasks are ranked to choose the perfect priority task to be transferred to the forthcoming processes.

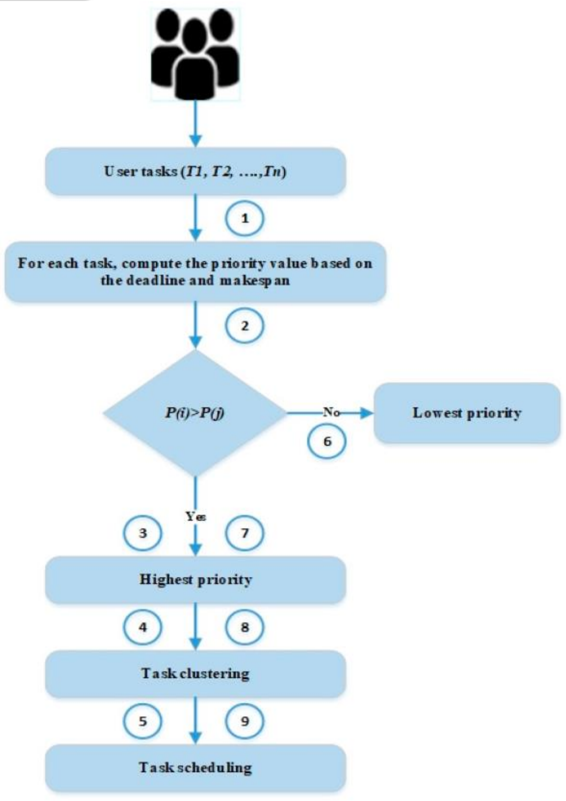

User tasks (T1, T2, ...,Tn)
1
For each task, compute the priority value based on the deadline and makespan
2
P(i)>P(j)
No
6
Lowest priority
Yes
3
7
Highest priority
4
8
Task clustering
5
9
Task scheduling

Figure 2. Flowchart illustrating task clustering and scheduling [1]

C. TASK RANKING

The MHEFT algorithm is used to rank scheduled tasks generated by utilising the DSC algorithm. The top priority task among the scheduled tasks is chosen from the approaching processes, like VM clustering and load balancing. To rank tasks, the average execution and communication times for transferring information are calculated. The arrival time of the tasks are checked in the cases when two tasks with identical priority are received.

The final priority is then assigned to the task that arrived first.

The ranking of task R(ti) is calculated as: -

$$R(t_i) = E(t_i) + \max_{t_j = s(t_i)} (c_{i,j} + R(t_j)) \quad (2)$$

where $E(t_i)$ is that the average execution time of the task across all VMs, $s(t_i)$ is the set of immediate successors of the task, $R(t_j)$ is the computation of all of its children, $c_{i,j}$, the communication time that corresponds to transfer of $data_{i,j}$, via edges i and j;

$c_{i,j}$ is computed as: -

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

where y is the average latency, b denotes the average bandwidth of communication links among VMs within the system.

D. OPTIMIZED_HB FLOW CHART

In this section, a flow diagram describing the control structure for the Optimized_HB is depicted in Figure 3. It describes the mechanism of load balancing among overloaded virtual machines and underloaded virtual machines. The basic idea is to submit the tasks to the virtual machine until the machine gets overloaded i.e. load on that virtual machine exceeds the threshold value. We do not submit tasks to the overloaded machine and we send the remaining tasks to underloaded virtual machines that can be found by the RANDOMLY SEARCH method. Here in the flowchart Cloudlets corresponds to tasks/jobs that user requests to the VM.

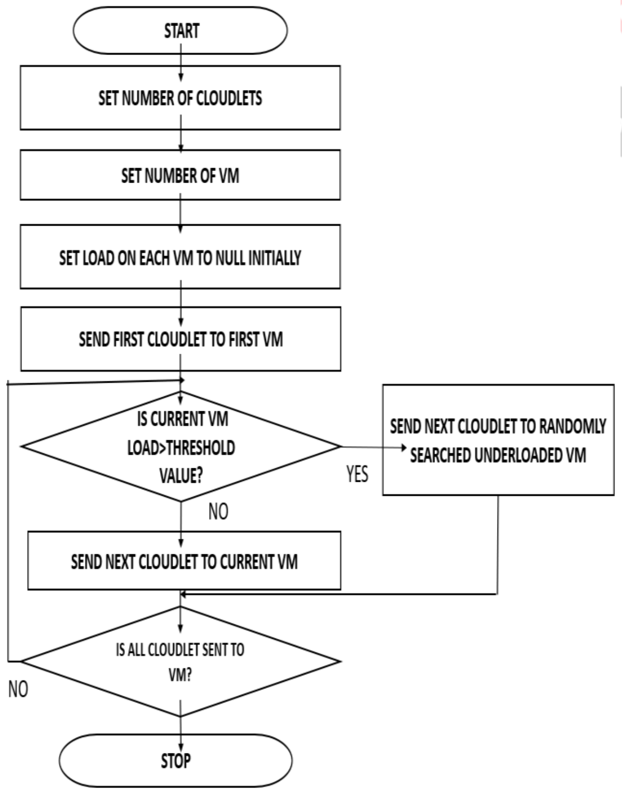

START
SET NUMBER OF CLOUDLETS
SET NUMBER OF VM
SET LOAD ON EACH VM TO NULL INITIALLY
SEND FIRST CLOUDLET TO FIRST VM
IS CURRENT VM LOAD>THRESHOLD VALUE?
YES
SEND NEXT CLOUDLET TO RANDOMLY SEARCHED UNDERLOADED VM
NO
SEND NEXT CLOUDLET TO CURRENT VM
IS ALL CLOUDLET SENT TO VM?
NO
STOP

Figure 3. HBO Algorithm Flowchart [3]

E. ALGORITHM

1. Start
2. For each task do
3. Calculate the load on VM and decide whether to do load balancing or not
4. Group the VMs based on load as overloaded or underloaded.
5. Find the supply of underloaded VMs and the demand of overloaded VMs.
6. Sort the overloaded and underloaded VM sets
7. Sort the tasks in overloaded VMs based on priority.
8. Find the capacity of VMs in the underloaded set.
9. For each task in each overloaded VM find a suitable underloaded VM based on capacity.
10. Update the overloaded and underloaded VM sets.
11. End of step 2.
12. Stop.

F. CALCULATIONS OF OPTIMIZED_HB

The proposed Optimized_HB algorithm allocates the incoming task to a VM that currently has fewer tasks than the remaining VMs. Also, the deviation of the processing time of this VM from the average processing time of all VMs is a smaller amount than the predefined threshold value. So it avoids underutilization and overutilization of resources. The calculations employed in the Optimized_HB are shown below.

1. VM Current Load Calculation:-

It measures the magnitude relation between total lengths of the submitted tasks to a VM to the process rate of that VM at a selected instance. Suppose the total range of tasks appointed to a VM is N, Len corresponds to the length of single tasks and MIPS denotes Million Instruction Per Second rate of that VM, then the current load is calculated using the equation (4).

$$Load_{VM} = \frac{N*Len}{MIPS} \quad (4)$$

In order to find the total load on the entire data center, the sum of load on each VMs is calculated as per the equation (5).

$$Load_{DC} = \sum_{vm=1}^{n} Load_{VM} \quad (5)$$

The processing capacity of VM can be calculated using the equation (6) as given below.

$$Capacity_{VM} = PE_{num} * PE_{mips} + VM_{bw} \quad (6)$$

Where PE_{num} corresponds to the number of processing elements in a particular VM, PE_{mips} corresponds to the processing power of PE in MIPS rate and VM_{bw} corresponds to the bandwidth associated with a VM. A data centre may consist of several VMs. Thus, the total capacity of the entire data centre can be calculated from using the equation (7).

$$Capacity_{DC} = \sum_{vm=1}^{n} Capacity_{VM} \quad (7)$$

Then the proposed algorithm computes the processing time of each task using equation (8).

$$PT = \frac{Current\ Load}{Capacity} \quad (8)$$

The processing time required for the datacentre to complete all the tasks in it can be calculated by the equation (9).

$$PT_{DC} = \frac{Load_{\ DC}}{Capacity_{\ DC}} \quad (9)$$

The proposed method uses the Standard Deviation (SD) for measuring the deviations in the workload on each VM. Equation (10) gives the SD of loads.

$$SD = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(PT_i - PT)^2} \quad (10)$$

2. Load Balancing & Scheduling Decision:-

In this phase, load balancing and rescheduling of tasks are decided depending on the SD value calculated using equation (10). In order to maintain system stability by minimizing the number of migrations, the decision will be taken only when the capacity of the datacentre is greater than the current load. For finding the load, a threshold value is set (value lies in 0-1) based on the SD calculated which is compared with the calculated SD measure. The load balancing and scheduling are done only if the calculated SD exceeds the threshold value.

3. VM Grouping:-

VMs are classified into two groups: overloaded VMs and underloaded VMs. This may cut back the time needed to seek out the best VM for task migration. Within the proposed technique the overloaded VMs are thought of as honeybees and the underloaded VMs are their food sources. The VMs are classified in line with the SD and threshold value already calculated on the basis of the load.

4. Task Scheduling:-

The system should realize the demand for every overloaded VM and provide it to the overloaded VMs. The VMs are sorted based on the capacity in ascending order. The proposed methodology selects the task with the lowest priority from the overloaded VM and it is rescheduled to an underloaded VM (target VM) having the best capacity. The supply to a specific VM is the difference between its capacity and current load and may be calculated utilising equation (11).

$$Supply_{VM} = Capacity - Load \quad (11)$$

Then the demand of a VM is calculated using the equation (12).

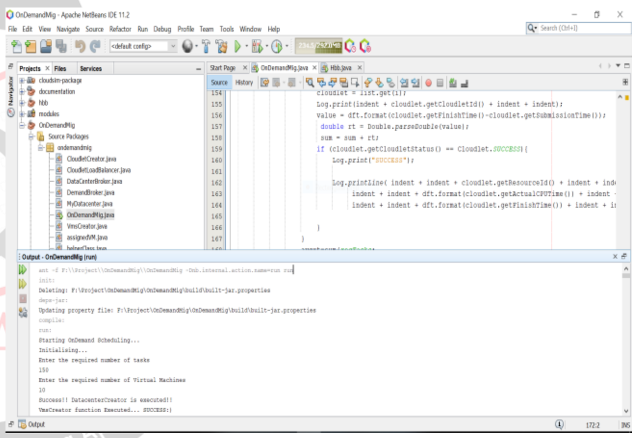
$$Demand_{\ VM} = Load - Capacity \quad (12)$$

On the submission of each task into the cloud, the system calculates the standard deviation. If the standard deviation of the VM load σ is underneath or adequate to the threshold condition, whose value is between [0–1] then the system is balanced. Otherwise, the system is in an unbalanced state, overloaded, or underloaded. If the standard deviation of loads is larger than the threshold, then the load balancing method is initiated. Throughout this load balancing method, VMs are classified into underloaded and full VM sets. Then the submitted tasks are rescheduled to the VM having the best capacity. Once the present workload of a VM cluster exceeds the utmost capacity of the group that is predefined by the threshold value, then the cluster is overloaded. Load balancing isn't attainable in this VM cluster.

III. RESULTS

For the analysis section, we carried out a comparison between the normal honey bee algorithm and the newly proposed Optimized_HB.

A. INPUT

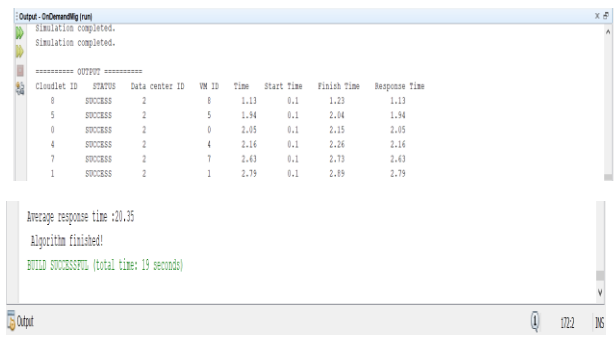


```
cloudlet = list.get(i);
Log.print(indent + cloudlet.getCloudletId() + indent + indent);
value = dft.format(cloudlet.getFinishTime()-cloudlet.getSubmissionTime());
 double rt = Double.parseDouble(value);
 sum = sum + rt;
if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){
    Log.print("SUCCESS");

    Log.printLine( indent + indent + cloudlet.getResourceId() + indent + ind
          indent + indent + dft.format(cloudlet.getActualCPUTime()) + indent
          indent + indent + dft.format(cloudlet.getFinishTime()) + indent + i

    }
}
```

```
ant -f F:\\Project\\OnDemandMig\\OnDemandMig -Dnb.internal.action.name=run run
init:
Deleting: F:\Project\OnDemandMig\OnDemandMig\build\built-jar.properties
deps-jar:
Updating property file: F:\Project\OnDemandMig\OnDemandMig\build\built-jar.properties
compile:
run:
Starting OnDemand Scheduling...
Initialising...
Enter the required number of tasks
150
Enter the required number of Virtual Machines
10
Success!! DatacenterCreator is executed!!
VmsCreator function Executed... SUCCESS:)
```

Figure 4. User input on the screen

B. OUTPUT



```
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID   STATUS   Data center ID   VM ID   Time   Start Time   Finish Time   Response Time
    8         SUCCESS       2             8     1.13      0.1          1.23          1.13
    5         SUCCESS       2             5     1.94      0.1          2.04          1.94
    0         SUCCESS       2             0     2.05      0.1          2.15          2.05
    4         SUCCESS       2             4     2.16      0.1          2.26          2.16
    7         SUCCESS       2             7     2.63      0.1          2.73          2.63
    1         SUCCESS       2             1     2.79      0.1          2.89          2.79
```

```
Average response time :20.35
 Algorithm finished!
BUILD SUCCESSFUL (total time: 19 seconds)
```

Figure 5. Average response time outputted

The performance analysis of the proposed Optimized_HB is carried out in a simulated dynamic environment. In this heterogeneous environment, VMs having different specifications are deployed. Cloudlets with varying specifications are submitted into this cloud environment. The response time for each of the submitted tasks is measured and compared with the Normal Honey Bee technique. The response time of the proposed Optimized_HB method with Normal Honey Bee while keeping the number of VMs constant and the number of tasks varying is shown in Table 1. These values are used as a reference and the corresponding graph is plotted in Figure 6.

Table 1: Comparison of Response time with varying no. of tasks and constant no. of VMs

No. of Tasks	Normal Honey Bee	Proposed Technique
100	80.82	13.08
200	150.58	23.73
300	213.47	36.51
400	300.9	47.02
500	389.07	59.67

Figure 6 shows the average response time of honey bee based algorithm and proposed Optimized_HB algorithm versus the number of tasks. The X-axis represents the number of tasks and Y-axis represents the response time in milliseconds. The number of tasks takes the values from 100, 200, 300, 400, and 500. It is clearly evident that the response time is greatly reduced for Optimized_HB compared to that of the Normal Honey Bee algorithm.

The above experimental results show how the proposed Optimized_HB algorithm reduces the response to time to a great extent. Our Optimized_HB saves up to 84% of the average response time over the Normal Honey Bee algorithm. This is because the Optimized_HB considers the least load, availability of VMs, and load variation of each VM when assigning tasks to VMs.

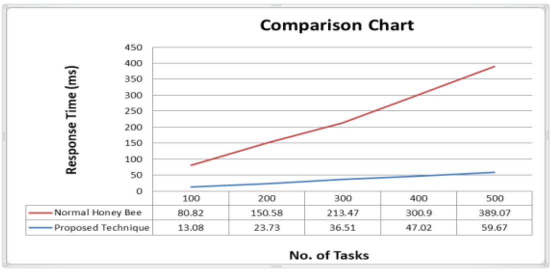


Figure 6. Comparison chart for response time versus no. of tasks

The graph considers a constant number of virtual machines i.e. 10 virtual machines. When the number of tasks is 100, the normal honey bee algorithm responds in 80.82 ms time whereas the proposed technique responds in 13.08 ms. At 200 tasks, the normal algorithm responds in 150.58 ms and the latter in 23.73 ms. At 300 tasks, the former responds in 213.47 ms and the latter in 36.51 ms. At 400 tasks, the former responds in 300.9 ms and the latter in 47.02 ms. Finally, at 500 tasks, the former responds in 389.07 ms and the latter in 59.67 ms.

The response time of the proposed Optimized_HB method with Normal Honey Bee while keeping the number of tasks constant and the number of VMs varying is shown in Table 2. These values are used as a reference and the corresponding graph is plotted in Figure 7.

Table 2: Comparison of response time with varying no. of VMs and constant no. of tasks

No. of VMs	Normal Honey Bee	Proposed Technique
10	751.01	128.28
30	258.66	43.75
50	156.91	27.03
70	112.25	19.49
90	86.26	15.45

Figure 7 shows the average response time of honey bee based algorithm and proposed Optimized_HB algorithm versus the number of VMs. The X-axis represents the number of VMs and Y-axis represents the response time in milliseconds. The number of VMs takes the values from 10, 30, 50, 70, and 90. It is evident that the response time is greatly reduced for Optimized_HB compared to that of the Normal Honey Bee algorithm.

The above experimental results show how the proposed Optimized_HB algorithm reduces the response to time to a great extent. Our Optimized_HB saves up to 83% of the average response time over the Normal Honey Bee technique.

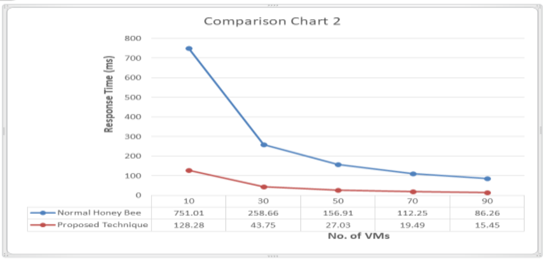


Figure 7. Comparison chart for response time versus no. of VMs

The graph considers a constant number of tasks i.e. 1000 tasks. When the number of VMs is 10, the normal honey bee algorithm responds in 751.01 ms time whereas the proposed technique responds in 128.28 ms. At 30 VMs, the normal algorithm responds in 258.66 ms and the latter in 43.75 ms. At 50 VMs, the former responds in 156.91 ms and the latter in 27.03 ms. At 70 VMs, the former responds in 112.25 ms and the latter in 19.49 ms. Finally, at 90 VMs, the former responds in 86.26 ms and the latter in 15.45 ms.

From these results, it is clear that response time is reduced into a significant amount while using our Optimized_HB algorithm. This is because Optimized_HB considers an optimized technique for distributing loads in VMs. Now, the users will get a faster response than older methods. Response time is a good measure of QoS provided by the service provider. So here the provider can assure good QoS to their customers.

IV. CONCLUSION

This project focuses on an optimized approach for load balancing and task scheduling in the cloud computing environment by incorporating the foraging behaviour of honey bee (HBO) and DSC algorithm that eventually led to a decrease in the total response time of the user tasks and also effectively handling and servicing requests from the users. Incoming tasks are clustered using the DSC algorithm on the thought of priority. The priority value is computed consistent with the highest and lowest levels of the tasks and is employed to make clusters for task scheduling. The MHEFT algorithm is adopted to rank scheduled tasks via rank value computation, during which the average execution and communication times for every task are reconsidered. The highest priority task is chosen and appointed first for approaching processes. The tasks are to be sent to the underloaded machine and like foraging bees, succeeding tasks also are sent to that virtual machine until the machine gets overloaded as flower patches exploitation is completed by scout bees. Load balancing that's inspired by honey bee behaviour improves the final throughput of the process. Priority-based balancing focuses on reducing the quantity of time a task possesses to serve a queue of the VM. Thus, it reduces the response time of VMs. As a part of the evaluation strategy, we compared our proposed Optimized_HB with the Normal Honey Bee algorithm. Results demonstrated that our algorithm stands good in terms of a minimal response time compared to the normal honey bee algorithm without increasing additional overheads. In the subsequent years, we plan to improve this proposed algorithm by considering other QoS factors of tasks such as makespan, accepted rate, etc.

V. REFERENCES

[1] A. Al-Rahayfeh, S. Atiewi, A. Abuhussein and M. Almiani, "Novel Approach to Task Scheduling and Load Balancing Using the Dominant Sequence Clustering and Mean Shift Clustering Algorithms", *Future Internet*, vol. 11, no. 5, p. 109, 2019. Available: 10.3390/*fi*11050109.

[2] C. K. P. Gohel, "A Novel Honey Bee Inspired Algorithm for Dynamic Load Balancing In Cloud Environment," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 4, no. 8, pp. 6995–7000, 2015.

[3] H. Gupta and K. Sahu, "Honey Bee Behaviour Based Load Balancing of Tasks in Cloud Computing", *International Journal of Science and Research (IJSR)*, vol. 3, no. 6, pp. 842-846, 2014.

[4] K. R. Remesh Babu and P. Samuel, "Enhanced Bee Colony Algorithm for Efficient Load Balancing and Scheduling in Cloud," *Journal of Network and Innovative Computing*, vol. Volume 4.

[5] M. Hamza, S. Pawar and Y. Jain, "A New Modified HBB Optimized Load Balancing in Cloud Computing", *International Journal of Computer Science and Network*, vol. 4, no. 5, pp. 799-809, 2020.

[6] S. Mittal and A. Katal, "An Optimized Task Scheduling Algorithm in Cloud Computing," *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, Bhimavaram, 2016, pp. 197-202.

[7] T. Mathew, K. C. Sekaran, and J. Jose, "Study and analysis of various task scheduling algorithms in the cloud computing environment," *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2014.

[8] W. Hashem, H. Nashaat and R. Rizk, "Honey Bee Based Load Balancing in Cloud Computing," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 12, pp. 5694-5711, 2017. DOI: 10.3837/tiis.2017.12.001.

[9] Z. Y. Zhang and M. Moh, "Randomized Load Balancing for Cloud Computing using Bacterial Foraging Optimization," *Proceedings of the 2019 ACM Southeast Conference on ZZZ - ACM SE '19*, 2019.