REVA UNIVERSITY
Bengaluru, India

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A Project Report

on

LOAD BALANCING FOR TRAFFIC CONTROL IN CLOUD COMPUTING USING HONEYBEE OPTIMIZATION

Submitted in fulfilment of the requirements for the award of the Degree of

Bachelor of Technology

Submitted by

V LOHITH KUMAR (R19CS367)
Y CHARISH CHOWDARY (R19CS377)
VISHRUTH SAI G (R19CS375)
Y KUMARNATH (R19CS383)

Under the guidance of

Prof. A AJIL

2023

Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru-560064
www.reva.edu.in

DECLARATION

We, **Mr. V Lohith Kumar**, **Mr. Y Charish Chowdary**, **Mr. Vishruth Sai G**, **Mr. Y Kumarnath** students of Bachelor of Technology, belong to School of Computer Science and Engineering, REVA University, declare that this Project Report / Dissertation entitled "LOAD BALANCING FOR TRAFFIC CONTROL IN CLOUD COMPUTING USING HONEYBEE OPTIMIZATION" is the result the of project / dissertation work done by us under the supervision of **Prof. A Ajil** at School of Computer Science and Engineering, REVA University.

We submit this Project Report / Dissertation in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering by the REVA University, Bangalore during the academic year 2023.

We declare that this project report has been tested for plagiarism and has passed the plagiarism test with the similarity score less than 20% and it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

We further declare that this project / dissertation report or any part of it has not been submitted for award of any other Degree / Diploma of this University or any other University/ Institution.

Signature of the candidates with dates

1.

2.

3.

4.

Certified that this project work submitted by

We, **Mr. V Lohith Kumar**, **Mr. Y Charish Chowdary**, **Mr. Vishruth Sai G**, **Mr. Y Kumarnath** *has been carried out under my / our guidance and the declaration made by the candidate is true to the best of my knowledge.*

Signature of Guide *Signature of Director of School*

Date: ……………. *Date: …………….*

Official Seal of the School

REVA UNIVERSITY
Bengaluru, India

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING.

CERTIFICATE

Certified that the project work entitled **LOAD BALANCING FOR TRAFFIC CONTROL IN CLOUD COMPUTING USING HONEYBEE OPTIMIZATION** carried out under our guidance by V Lohith Kumar(R19CS367),Y Charish Chowdary (R19CS377), Vishruth Sai G(R19CS375), Y Kumarnath(R19CS383), are bonafide students of REVA University during the academic year 2019-2023, are submitting the project report in partial fulfillment for the award of **Bachelor of Technology** in Computer Science and Engineering during the academic year **2023.** The project report has been tested for plagiarism and has passed the plagiarism test with the similarity score less than 20%. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Signature with date		**Signature with date**
.	.	.
<Guide name>		**<Name of the Director>**
Guide		**Director**

External Examiners

Name of the Examiner with affiliation	**Signature with Date**
1.	
2.	

ACKNOWLEDGEMENT

Any given task achieved is never the result of efforts of a single individual. There are always a bunch of people who play an instrumental role leading a task to its completion. Our joy at having successfully finished our mini project work would be incomplete without thanking everyone who helped us out along the way. We would like to express our sense of gratitude to our REVA University for providing us the means of attaining our most cherished goal.

We would like to thank our Hon'ble Chancellor, **Dr. P. Shyama Raju** and Hon'ble ViceChancellor, **Dr. M Dhanamjaya** for their immense support towards students to showcase innovative ideas.

We cannot express enough thanks to our respected Director, **Dr Ashwinkumar U M** for providing us with a highly conducive environment and encouraging the growth and creativity of each and every student. We would also like to offer our sincere gratitude to our Project Coordinators for the numerous learning opportunities that have been provided.

We would like to take this opportunity to express our gratitude to our Project Guide, **Prof.A Ajil** for continuously supporting and guiding us in our every endeavor as well for taking a keen and active interest in the progress of every phase of our Project. Thank you for providing us with the necessary inputs and suggestions for advancing with our Project work. We deeply appreciate the wise guidance that sir has provided.

Finally, we would like to extend our sincere thanks to all the faculty members, staff from School of Computer Science and Engineering.

V Lohith Kumar (R19CS367)

Y Charish Chowdary (R19CS377)

Vishruth Sai G (R19CS375)

Y Kumarnath (R19CS383)

Contents

List of Tables with titles and page reference.

List of illustrations / Screen Shots if any, with titles and page references.

List of Symbols, Abbreviation of Nomenclature

Label	**Title**	**Page Reference**
Eq 1	PT_{host}	6
Eq 2	Load of each VM	6

Abstract:

A Common issue that found in cloud is load balancing which makes performance unstable and slow. Load balancing involves distributing of application tasks to different processors to decrease the execution time and improve cloud computing efficiency and user satisfaction. It has emerged as a popular growing technology now a days. This is achieved by optimizing resource use, maximizing performance, and minimizing latency. An algorithm is proposed that improves load balancing by ensuring that no virtual machines remain idle and aims for a well-balanced load across all virtual machines for optimized solutions and reduced latency. This approach ensures that underloaded virtual machines are assigned tasks to maintain balance, rather than overloading fully loaded virtual machines. A comparison was made between the newly proposed algorithm and existing dynamic load balancing. Based on experimental results, it was found that the proposed methodology is more effective than the already existing algorithm with considerable decrease in the average processing time.

1.Introduction:

Cloud computing is the on-demand availability of computing resources including tools and applications through the internet. it can provide both private and public cloud service for a fee.

The cloud offers number of benefits to all sectors of companies which encompass the ability to make use of software from any device [1]. End users in thousands will access the resources at a time which may cause uneven balance and break down in entire system, slows down entire processing in cloud

Cloud computing offers shared processing resources and data without the need for users to buy servers or pay for power. It's also convenient for remote workers who can access their applications from anywhere [2]. However, as the users across cloud computing environment increase, so that demand for shared resources also grows, making load balancing a key challenge.

Load balancing involves distributing workloads and computing infrastructure among multiple users, and service providers to manage the workload demands. It helps avoid bottlenecks and achieves several benefits, such as equal task distribution, improved service quality, better system performance, reduce time to time, and optimal resource infrastructure [3]. The first aspect of load balancing is to equally balance the workload in hosts based on their processing speed, memory space, and also bandwidth.

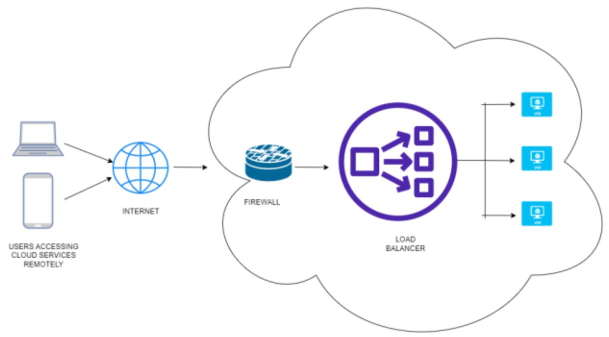


Fig 1: Architectural Design of Load balancing

Here, load balancer acts as a device that distributes the network traffic among the servers equally without Causing any traffic blocking.

Algorithms in load balancing are categorized into two types, namely static and dynamic. While static is less complex, they work best when the hosts have minimal fluctuations in their load. This is because they do not take current behaviour while distributing the workload. On the other hand, dynamic algorithms are preferred for distributed systems like cloud computing [3]. An example of a static scheduling algorithm is Round Robin (RR), each turn it assign task to node without regard to the VM's resource capacity or the task's execution time. In contrast, a dynamic load balancing algorithm such as the Modified Throttled Algorithm equally distributes the task among the available VMs, but it does not consider resource utilization while allocating the task. Traditional load balancing algorithms face several limitations in the dynamic workload dynamics of cloud computing. To overcome the challenges, in recent years artificial bee colony and ant colony optimization have emerged. These algorithms have made significant strides in addressing the dynamic nature of cloud computing.

Our proposed method suggests that load balancing is mimicking the foraging behavior in honeybees. Here this algorithm is based on a thorough analysis how honeybees search and collect food. In a beehive, scout bees search for food and communicate the location and quality of the food by performing a waggle dance. Forager bees follow scout bees to the food source and collect it. Then they return to the beehive and perform a waggle dance to inform other bees about the quantity of food remaining, resulting in more exploitation of the food source.

2.Literature Survey:

P Kumar and R. Kumar published a article “Issues and challenges of load balancing techniques in cloud computing” [1] discusses the challenges and issues associated with them. As it ensures that resources are distributed efficiently across multiple servers, thereby maximizing resource utilization, and minimizing response time.

The paper “A Comprehensive Study of LB Technique in Cloud Infrastructure” which is written by A. Ajil [2] and Kumar shows the technique used in load balancing for minimizing the load efficiently and also provides the information of different parameters that are there in load balancing.

The authors Ali, Mrs Anooja, and D. P. Sumalatha for an article titled "A survey on Balancing the Load of Big Data for Preserving Privacy Access in Cloud" published in 2018, as it ensures that the resources are distributed evenly across multiple nodes or servers. The authors focus on the challenge of balancing the load of big data in a way that maintains privacy access, meaning that sensitive data is not accessible to unauthorized users [3].

A paper titled "Load Balancing of Nodes in Cloud Using Ant Colony Optimization" presented in 2012 written by Kumar Nishant, Pratik Sharma [4], and Vishal Krishna. This shows algorithms for cloud computing that uses ant colony optimization. ACO is a swarm intelligence-based algorithm that simulates the behavior of ants in nature to find the shortest path between a source and destination. The proposed algorithm aims to distribute workloads across multiple nodes or servers in cloud environment using the ACO algorithm.

"A review on dynamic load balancing algorithms" which is written by Shalu Rani, Sakshi Dhingra and Dharminder Kumar [5]. The paper provides a review of various dynamic load balancing algorithms used in distributed systems such as cloud computing, grid computing, and parallel computing. Dynamic load balancing refers to the process of equally distributing workloads across multiple nodes or servers in real-time based on current state of system.

This is a citation titled "Survey on Existing Load Balancing Algorithms in Cloud Environment" in 2021 which was written by Suman Sansanwal and Nitin Jain [6]. The paper provides a survey of existing algorithms that are used in cloud computing environments to distribute workloads across multiple servers or nodes in load balancing. The authors review different types of load balancing algorithms, including static and dynamic algorithms, and discuss their advantages and limitations and also shows a comparative analysis of different algorithms based on criteria such as performance, scalability, and fault tolerance.

This is a citation for an article titled "Load Balancing Algorithms for Cloud Computing Environment" published in May 2020. [7] it provides an overview among algorithms that are used to optimize the utilization of resources and minimize response time and compares their performance parameters.

A research paper titled "A Systematic Study of Load Balancing Approaches in the Fog Computing Environment" published in 2021 written by M. Kaur and R. Aron. [8] provides a systematic study to approach the fog computing environment. which extends cloud computing processing and storage to be located closer to the user or device. as it ensures that resources are distributed efficiently across multiple fog nodes or servers. Architectural Design

3.Positioning:

3.1 Problem statement:

The optimization problem needs to be converted into the problem of finding the shortest path on a weighted graph.

4.Project overview:

4.1 Objectives:

The main objective is to distribute the workload or task among all available VM's to minimize the Response Time and Traffic using the Honeybee Optimization.

4.2 Goals:

1. To distribute workload in efficient way that each node is assigned with equal amount of work.
2. To have a backup plan in case the system fails even partially.
3. To avoid over loading and maintain the system stability.

5. Project Scope:

General Project Information:
Optimization of workflow in Load Balancing using Honeybee Algorithm.

Business Benefits:

Improves the Performance of E-Commers Websites like Flipkart, amazon.
And it helps the User requests quickly and accurately.
Network load efficiently across multiple servers.

Project Deliverables:

The complexity can be reduced and reduces the time taken to compute the order of requests on virtual servers and cloud computing environment.

6. Methodology:

Until recently, most research that are done on load balancing assumed stable nodes, which is often unrealistic in dynamic and heterogeneous cloud computing systems. As demand for resources or services fluctuates, new instances of the platform are launched, causing to increase the size and complexity of system [7]. Rule sets are imposed to manage the workload, but as the systems grow, these rules become unwieldy, and it becomes difficult for observation and response cycle. In summary, the meta systems is to manage efficiently organized load balancing strategy and agile sufficiently.

Therefore, a self-regulating load system is needed that can balance the load within the entities without any information about the system. This self-organized regulation can be distributed by distributed algorithms. To address this, we propose an implementation algorithm to manage the dynamic environment of virtual machine programming and requests in cloud computing. To balance the load in a cloud computing environment, tasks should be sent to the under loaded virtual machine, similar to how foraging bees send consecutive tasks to a flower patch until it is full. By taking inspiration from honeybee behaviour, we can improve the overall efficiency of the system in load balancing. Priority-based balancing focuses mainly on decreasing the time spends of task in a virtual machine's queue, similar to how honey bees prioritize certain flower patches to maximize efficiency. This approach reduces the response time of virtual machines.

Honeybee foraging behaviour in Load Balancing

To achieve load balancing, it is recommended to allocate tasks to the less loaded virtual machine. This strategy is analogous to how foraging bees direct sequential tasks to a flower patch until it reaches capacity. By emulating the behaviour of honeybees, the efficiency of the load balancing process can be improved. The priority-based balancing method concentrates on minimizing the duration that a task spends in the queue of a virtual machine, which is similar to how honeybees select particular flower patches to maximize productivity. By adopting this technique, the time of response in virtual machines can be reduced.

The algorithm of Artificial Bee Colony begins by pairing each bee with randomly selected food sources, without any bias. The bees then gather nectar from these sources and return to the hive, sharing their findings with other bees through dance. Next, a population of scout bees is established and randomly sent out to the food sources to calculate their fitness values.

Bees that are currently employed collect and store information about food sources in their vicinity. They then share this information through dance to other bees. Afterward, the employed bees revisit previously visited food sources and use the shared information to select new sources in the area. The scout bees based on nectar information they choose food which information is provided by the employed bees. The onlooker bees in the hive receive the information from the working bees about food source, and then choose one of the sources to visit. These bees wait for a dance performed by the employed bees to determine which food source to choose. The employed bees perform a specific type of dance, such as the waggle or tremble dance, to communicate about the quality, quantity, and distance of the food source from the hive. To find a food source scout bee conducts a random search, and when the bees no longer visit a particular source, the scout bee randomly selects a new source to investigate.

Algorithm: -

1.The user provides a list of available virtual resources in the data center (e.g., VM1, VM2… VMn.) and a list of tasks to be completed (T1, T2 … Tn.).

2.When a new request is received, the scheduler computes the computing capacity that is expected to be needed for the given tasks.

3.The average computing capacity for each task is determined using an equation - $PT_{host} = \frac{Total\ length\ of\ tasks\ submitted\ to\ host}{number\ of\ processors\ in\ host * processor\ speed\ of\ host}$

4.The load on each virtual machine (VM) is found - Load of each VM = $\frac{Number\ of\ tasks\ at\ time\ t\ on\ service\ queue\ of\ VM}{service\ rate\ of\ VMi\ at\ time\ t}$

5.The average system load is computed.

6.The load is checked against a probability value ranging from 0 to 1, and if the value is between 0 and 1, the VM is added to the underloaded list; otherwise, it is added to the overloaded list.

```
if (0< P()<1):
        Underloaded_list[]= VMi
    else:
    Overloaded_list[]= VMi
```

7.The underloaded VMs are selected and compared to the expected computing power of the tasks. If the average computing capacity of the VM is greater than or equal to the expected computing power of the tasks, the VM is marked as the fittest and the tasks are allocated to it.

8.After the tasks are allocated, some VMs may remain underutilized, resulting in wasted processor time. Randomization is performed in the system on certain threshold value or less than to it. VMs with a system load greater than another threshold value is selected, and jobs are randomly allocated to VMs with a system load less than or equal to the first threshold value. The algorithm will make N-1/N attempts on average to reallocate jobs among the VMs.

The flowchart shows that path how a task is being processed step by step uniformly. The flowchart consist of 4 sub parts i.e. workflow submission, task-in, task-out, delay

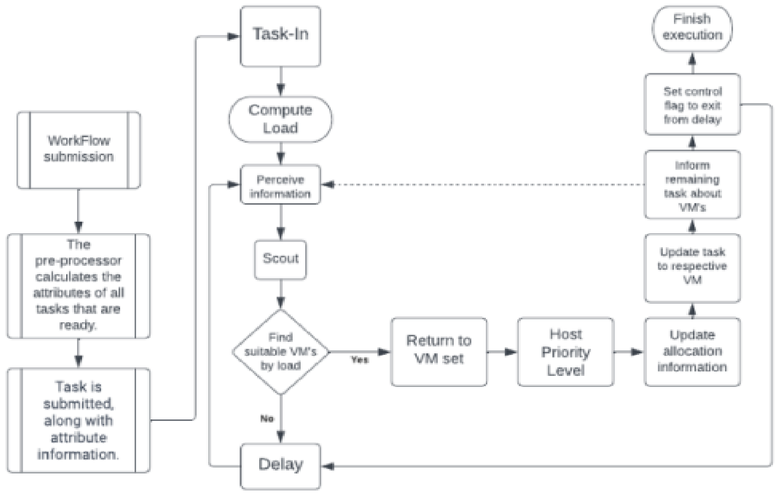

WorkFlow submission
The pre-processor calculates the attributes of all tasks that are ready.
Task is submitted, along with attribute information.
Task-In
Compute Load
Perceive information
Scout
Find suitable VM's by load
Yes
No
Delay
Return to VM set
Host Priority Level
Update allocation information
Update task to respective VM
Inform remaining task about VM's
Set control flag to exit from delay
Finish execution

Fig 2: Flow diagram for the behaviour of the proposed algorithm

7. Modules identified:

7.1 Workflow submission:

It deals with workflow submission. The task is arrived and the preprocessor computes the no of tasks and length of the tasks. The task attribute information is stamped along with the task.

7.2 Task in:

The task request the VM load balancer for execution. The task are inserted into compute load for allocation of VM.

The task are assigned to respective VM's based on there priority level. the allocation of VM to task is done. the process is same as honey bee visit the flower based on food is available in a flower or not. If the VM's are not free then the task will be in delay state. The allocation will be done based on host priority level[FCFS].

7.3 Task out:

The status of VM's is updated   whether the VM is being processed or not ,processing time, status of host or the VM is overloaded. The task allocation of respective VM's is done. The deallocation of VM's is done after the completion of the task assigned.

The information is sent back about the status of VM’s for remaining task which are in delay state and the process is done again . It will send a control flag to delay when the allocation tasks are finished.

7.4 Delay:

If the task doesn’t found a VM for execution then the task is allocated on delay state. It will go for delay until it receive control flag from task which has finished execution. If there is not task in delay state then the process is completed.

8.Project Implementation:

Load balancing involves distributing application tasks to different processors to decrease program execution time and improve cloud computing efficiency and user satisfaction. This is achieved by optimizing resource use, maximizing performance, and minimizing latency. Load balancing algorithms aim to reduce the latency of user applications by making the best use of available resources.

To achieve optimal machine utilization, tasks from overloaded virtual machines are transferred to less burdened virtual machines. An algorithm is proposed that improves load balancing by ensuring that no virtual machines remain idle. This algorithm, called the "Honey Bee Behaviors" based load balancing, aims to achieve a well-balanced load across virtual machines for optimized solutions and reduced latency. The proposed approach ensures that underloaded virtual machines are assigned tasks to maintain balance, rather than overloading fully loaded virtual machines. A comparison was made between the newly proposed algorithm and existing dynamic load balancing and scheduling algorithm. Based on experimental results, it was found that the proposed algorithm is more effective than the existing algorithm with considerable decrease in the average processing time.

8.1 Architectural Design:

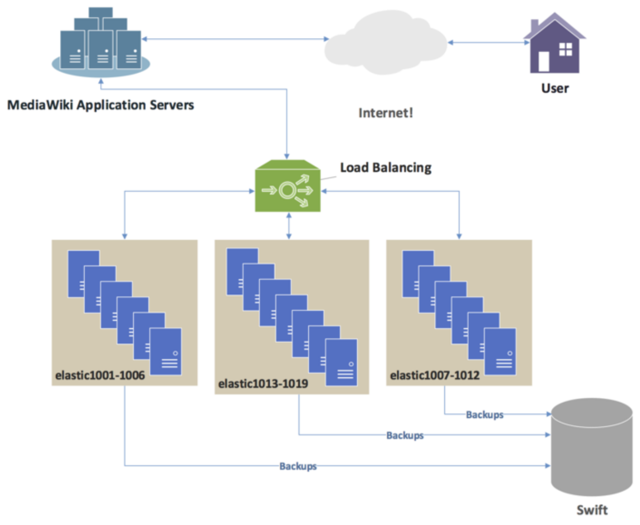


Fig 3 Architectural design of load balancing

Load balancing is a process of distributing an equal amount of work as it refers to efficiently distributing incoming network traffic across a group to improve performance and scalability . It is an invisible facilitator that sits between the Load user and the server [1].

Load Balancer can be placed at different stages of workload

1. Between Client and Frontend web server
2. Between Fronted web server and backend application server
3. Between backend application server and cache servers
4. Between cache servers and database server

8.2 Sequence Diagram:

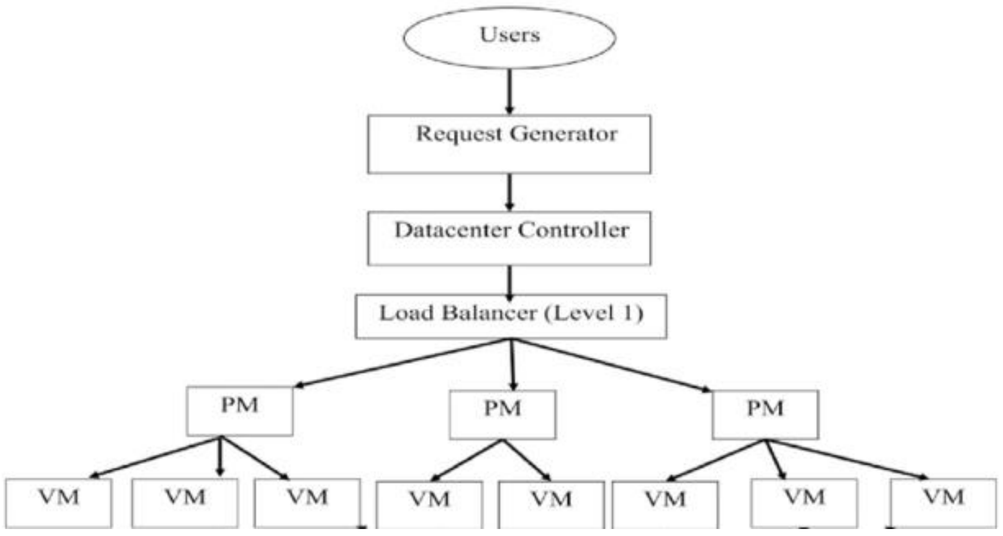


Fig 4 Sequence of Request in Load Balancing

8.4 Description of Technology Used:

Software: Cloud Analyst

Cloud Analyst is a tool which is used to simulate large scale cloud applications to understand performance under multiple configurations. The principle of the tool is to support evaluation of social networks based on geographic distribution of users and data centers.

Programming language: Java

9. Findings / Results of Analysis:

Cloud Analyst is used to stimulate task scheduling and model. The experimental evaluations are conducted using a simulation platform consisting of 50 VMs, Data centers and 100-1000 tasks. The tasks' length varies between 1000 to 2000million Instructions (MI) , while the speed of processer, memory size in VM, and bandwidth dictate the allowable load to each VM.

Table 1:Simulation Parameters

PARAMETERS	VALUES
Total no of tasks	100-1500
Length of task	2000-20000 bytes
No of VM's	50
Processor speed	500 - 10000
VM Memory size	256 - 2048 MB

To customize the simulation outcomes, an examination was conducted on the impact of various parameters. These parameters include the average response time, the length of executable instructions, and the average frequency of user requests per hour.

Fig 5 presents the average response time of algorithm, varying the length of executable instructions for each task between 2000 and 20000 bytes, and testing at different numbers of tasks. The cumulative response time of task measures the duration between request submission and the initial response, in milli-seconds. Results show that, under all conditions, the average response time begins at approximately 51.67 seconds. However, this value slightly increases to 52.05 seconds when the executable instruction length for a task is 2000 bytes and increases to 52.89 seconds when it is 20000 bytes. This is attributed to the fact that an increase in the tasks and as well as the length of instruction will impact the load of system, resulting in increased response times.

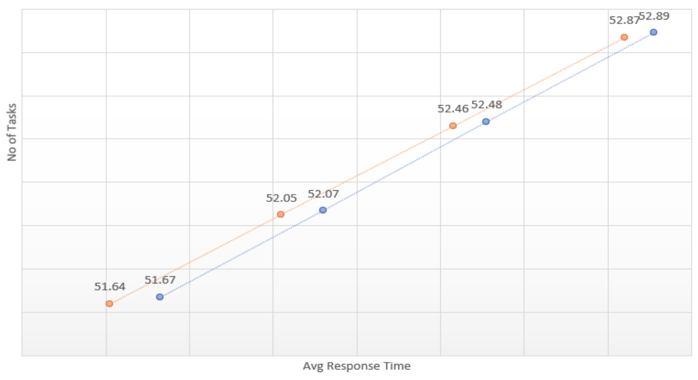
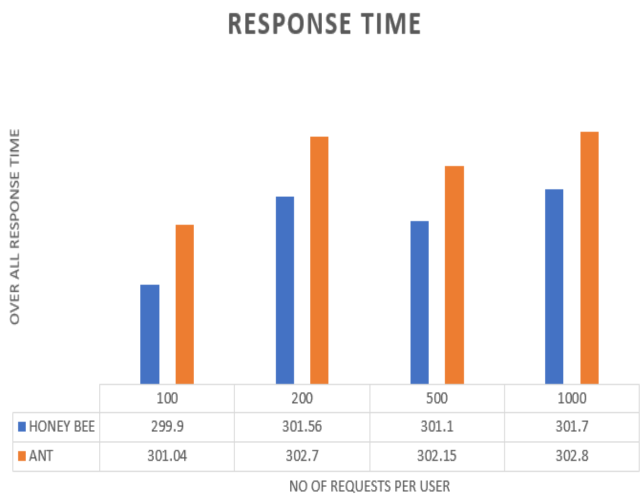


Fig 5: The study measures the average response time of Honey-Bee algorithm for different values of instruction length, while varying the number of tasks.

Comparison with swarm-based ant colony algorithm:

This compares simulation results of proposed algorithm with swarm-based ant colony algorithm. As noted in the previous section, the performance of the system is significantly impacted by the executable instruction size and no of tasks. Therefore, to demonstrate the impact of proposed algorithm a high default value for the tasks is chosen.

Fig 6 represents the average response time in proposed honeybee algorithm with ant colony load balancing algorithm. The tasks are varied in between 100 to 1000. The instruction size for these tasks vary randomly between 2000 and 20000. It is shown that the proposed honeybee algorithm saves up overall response time compared to ant colony algorithm. This is because, when assigning tasks to VMs, the proposed algorithm considers factors such as the availability of VMs, least load including load variation on each VM.



	100	200	500	1000
■ HONEY BEE	299.9	301.56	301.1	301.7
■ ANT	301.04	302.7	302.15	302.8

Fig 6: Average response time versus number of tasks

Fig 7 represents the average data processing time of proposed honeybee algorithm with ant colony algorithm. The tasks change from 100 to 1000. As it is shown that proposed honeybee algorithm performs computation faster thereby reducing the vm cost the data transfer cost as well.

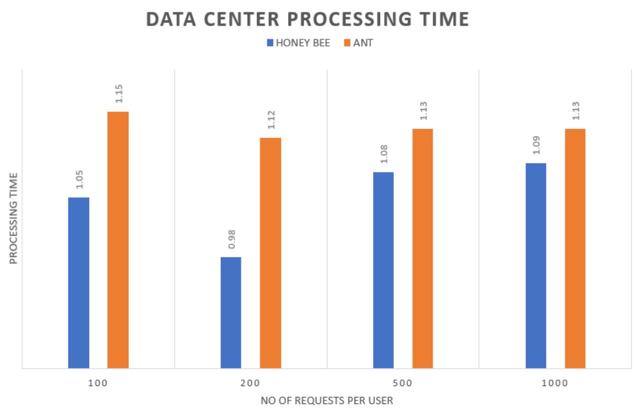


Fig 7: Data processing time verses no of tasks

10.Cost Analysis:

As this is a software application, funds are not required as of now.

11.Conclusion:

The paper proposes load balancing approach inspired by foraging strategy through behavior of honeybee in cloud system. The algorithm stabilizes the load by removing heavily loaded virtual machines while also considering the priorities of tasks. The honeybees are treated to be tasks to globally update the information. The algorithm also prioritizes tasks to improve the overall processing throughput and reduce the response time of VMs. When this proposed algorithm is set side by side to other existing techniques, and results show its efficiency without additional overheads. This technique is suitable for balancing independent tasks that are non-primitive and heterogeneous cloud computing systems.

12. Project Limitations and Future Enhancements:

Currently, our project is focused on generating simulations using a honey-bee-based swarm approach. However, as we look toward the future, we plan to take our project to the next level by integrating other swarm-based techniques into our algorithm. Our goal is to enhance the performance of our simulation models, and we believe that exploring a variety of swarm-based techniques will help us achieve this objective. With this approach, we hope to create even more robust and efficient simulations that can be applied in a wide range of fields.

13. References:

[1] P. Kumar and R. Kumar, ``Issues and challenges of load balancing techniquesin cloud computing: A survey," *ACM Comput. Surveys*, vol. 51,no. 6, pp. 1_35, Feb. 2019, doi: 10.1145/3281010.

[2] Ajil, A., Kumar, A Comprehensive Study of LB Technique in Cloud Infrastructure. SN COMPUT. SCI. 4, 181, 2023.

[3] Ali, Mrs Anooja, and D. P. Sumalatha. "A survey on Balancing the Load of Big Data for Preserving Privacy Access in Cloud." Asian Journal of Engineering and Technology Innovation (AJETI) (2018): 176.

[4] Kumar Nishant, Pratik Sharma, Vishal Krishna "Load Balancing of Nodes in Cloud Using Ant Colony Optimization" 14th International Conference on Modelling and Simulation, 2012.

[5] Shalu Rani, Dharminder Kumar, Sakshi Dhingra, "A review on dynamic load balancing algorithms", 2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), pp.515-520, 2022.

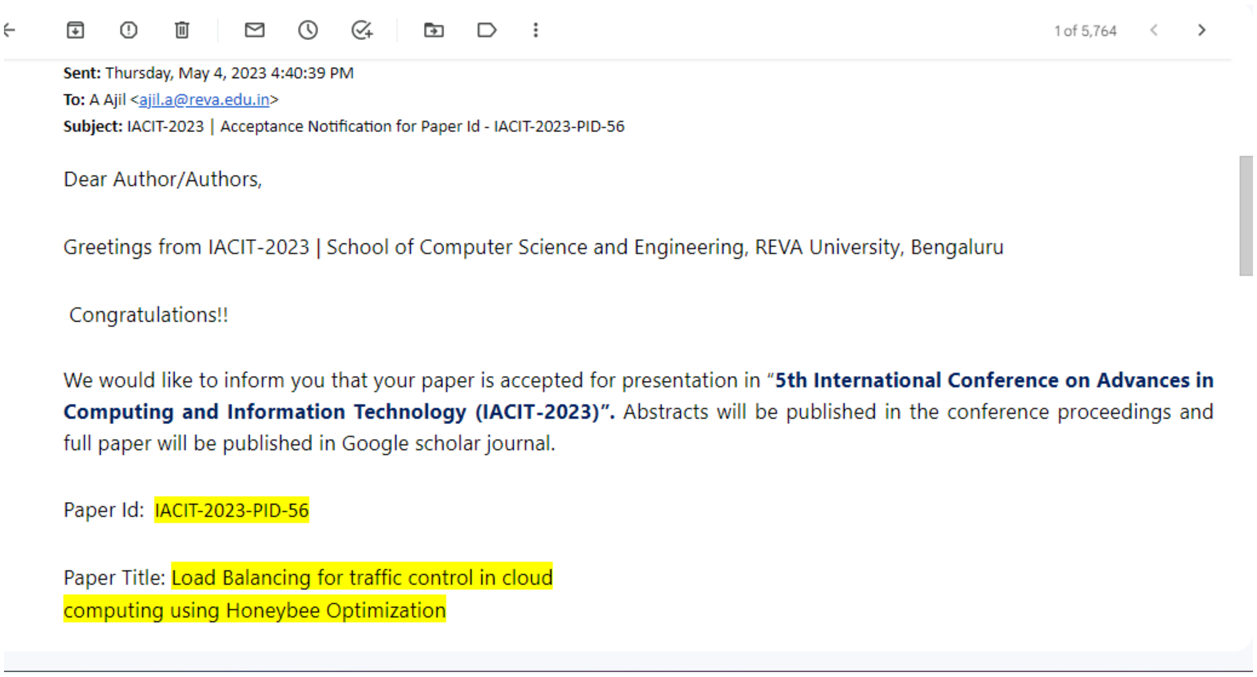
[6] Sansanwal, Suman and Jain, Nitin, Survey on Existing Load Balancing Algorithms in Cloud Environment (July 12, 2021). Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2021

[7] "Load Balancing Algorithms For Cloud Computing Environment ", International Journal of Emerging Technologies and Innovative Research (www.jetir.org | UGC and issn Approved), ISSN:2349-5162, Vol.7, Issue 5, page no. pp59-67, May-2020

[8] Kaur, M., Aron, R. A systematic study of load balancing approaches in the fog computing environment. J Supercomput 77, 9202–9247 (2021).

15.Copies of Articles:

15.1 Conference Paper Published:



1 of 5,764

Sent: Thursday, May 4, 2023 4:40:39 PM
To: A Ajil <ajil.a@reva.edu.in>
Subject: IACIT-2023 | Acceptance Notification for Paper Id - IACIT-2023-PID-56

Dear Author/Authors,

Greetings from IACIT-2023 | School of Computer Science and Engineering, REVA University, Bengaluru

Congratulations!!

We would like to inform you that your paper is accepted for presentation in "**5th International Conference on Advances in Computing and Information Technology (IACIT-2023)".** Abstracts will be published in the conference proceedings and full paper will be published in Google scholar journal.

Paper Id: IACIT-2023-PID-56

Paper Title: Load Balancing for traffic control in cloud computing using Honeybee Optimization

Int. Jrl. of Human Computations & Intelligence
Vol. 02, Issue No. 01, 2023
ISSN: 2583-5696

ORIGINAL RESEARCH

Load Balancing for Traffic Control in Cloud Computing Using Honeybee Optimization.

V Lohith Kumar. Y Charish. Vishruth Sai. Y Kumar Nath. A Ajil

School of Computing & Information Technology, REVA University, Bangalore, India

Abstract – Load balancing involves distributing application tasks to different processors to decrease the execution time and improve cloud computing efficiency and user satisfaction. This is achieved by optimizing resource use, maximizing performance, and minimizing latency. An algorithm is proposed that improves load balancing by ensuring that no virtual machines remain idle and aims for a well-balanced load across all virtual machines for optimized solutions and reduced latency. This approach ensures that underloaded virtual machines are assigned tasks to maintain balance, rather than overloading fully loaded virtual machines. A comparison was made between the newly proposed algorithm and existing dynamic load balancing. Based on experimental results, it was found that the proposed methodology is more effective than the already existing algorithm with considerable decrease in the average processing time.

Index Terms – Load balancing, optimizing, virtual machines, overloading, cloud computing.

I. INTRODUCTION

Cloud computing is the on-demand availability of computing resources including tools and applications through the internet. it can provide both private and public cloud service for a fee. The cloud offers number of benefits to all sectors of companies which encompass the ability to make use of software from any device [1] .

Cloud computing offers shared processing resources and data without the need for users to buy servers or pay for power. It's also convenient for remote workers who can access their applications from anywhere [2]. However, as the users across cloud computing environment increase, so that demand for shared resources also grows, making load balancing a key challenge.

Load balancing involves distributing workloads and computing infrastructure among multiple users, and service providers to manage the workload demands. It helps avoid bottlenecks and achieves several benefits, such as equal task distribution, improved service quality, better system performance, reduce time to time, and optimal resource infrastructure [3]. The first aspect of load balancing is to equally balance the workload in hosts based on their processing speed, memory space, and also bandwidth.

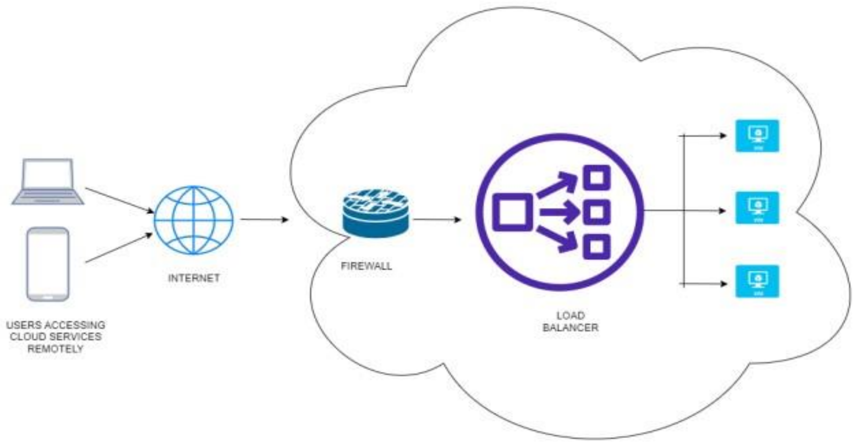


Fig 1: Architectural Design of Load balancing

Here, load balancer acts as a device that distributes the network traffic among the servers equally without Causing any traffic blocking.

Algorithms in load balancing are categorized into two types, namely static and dynamic. While static are less complex, they work best when the hosts have minimal fluctuations in their load. This is because they do not take current behavior while distributing the workload. On the other hand, dynamic algorithms are preferred for distributed systems like cloud computing [3]. An example of a static scheduling algorithm is Round Robin (RR), each turn it assign task to node without regard to the VM's resource capacity or the task's execution time. In contrast, a dynamic load balancing algorithm such as the Modified Throttled Algorithm equally distributes the task among the available VMs, but it does not consider resource utilization while allocating the task. Traditional load balancing algorithms face several limitations in the dynamic workload dynamics of cloud computing. To overcome the challenges, in recent years artificial bee colony and ant colony optimization have emerged. These algorithms have made significant strides in addressing the dynamic nature of cloud computing. Our proposed method suggests that load balancing is mimicking the foraging behavior in honeybees. Here this algorithm is based on a thorough analysis how honeybees search and collect food. In a beehive, scout bees search for food and communicate the location and quality of the food by performing a waggle dance. Forager bees follow scout bees to the food source and collect it. Then they return to the beehive and perform a waggle dance to inform other bees about the quantity of food remaining.

II. LITERATURE SURVEY

P Kumar and R. Kumar published a article “Issues and challenges of load balancing techniques in cloud computing” [1] discusses the challenges and issues associated with them. As it ensures that resources are distributed efficiently across multiple servers, thereby maximizing resource utilization, and minimizing response time.

The paper “A Comprehensive Study of LB Technique in Cloud Infrastructure” which is written by A. Ajil [2] and Kumar shows the technique used in load balancing for minimizing the load efficiently and provides the information of different parameters that are there in load balancing.

The authors Ali, Mrs Anooja, and D. P. Sumalatha for an article titled "A survey on Balancing the Load of Big Data for Preserving Privacy Access in Cloud" published in 2018, as it ensures that the resources are distributed evenly across multiple nodes or servers. The authors focus on the challenge of balancing the load of big data in a way that maintains privacy access, meaning that sensitive data is not accessible to unauthorized users [3].

A paper titled "Load Balancing of Nodes in Cloud Using Ant Colony Optimization" presented in 2012 written by Kumar Nishant, Pratik Sharma [4], and Vishal Krishna. This shows algorithms for cloud computing that uses ant colony optimization. ACO is a swarm intelligence-based algorithm that simulates the behavior of ants in nature to find the shortest path between a source and destination. The proposed algorithm aims to distribute workloads across multiple nodes or servers in cloud environment using the ACO algorithm.

"A review on dynamic load balancing algorithms" which is written by Shalu Rani, Sakshi Dhingra and Dharminder Kumar [5]. The paper provides a review of various dynamic load balancing algorithms used in distributed systems such as cloud computing, grid computing, and parallel computing. Dynamic load balancing refers to the process of equally distributing workloads across multiple nodes or servers in realtime based on current state of system.

This is a citation titled "Survey on Existing Load Balancing Algorithms in Cloud Environment" in 2021 which was written by Suman Sansanwal and Nitin Jain [6]. The paper provides a survey of existing algorithms that are used in cloud computing environments to distribute workloads across multiple servers or nodes in load balancing. The authors review different types of load balancing algorithms, including static and dynamic algorithms, and discuss their advantages and limitations and also shows a comparative analysis of different algorithms based on criteria such as performance, scalability, and fault tolerance.

This is a citation for an article titled "Load Balancing Algorithms for Cloud Computing Environment" published in May 2020. [7] it provides an overview among algorithms that are used to optimize the utilization of resources and minimize response time and compares their performance parameters.
A research paper titled "A Systematic Study of Load Balancing Approaches in the Fog Computing Environment" published in 2021 written by M. Kaur and R. Aron. [8] provides a systematic study to approach the fog computing environment. which extends cloud computing processing and storage to be located closer to the user or device. as it ensures that resources are distributed efficiently across multiple fog nodes or servers.

III. METHODOLOGY

Until recently, most research that are done on load balancing assumed stable nodes, which is often unrealistic in dynamic and heterogeneous cloud computing systems. As demand for resources or services fluctuates, new instances of the platform are launched, causing to increase the size and complexity of system [7]. Rule sets are imposed to manage the workload, but as the systems grow, these rules become unwieldy, and it becomes difficult for observation and response cycle. In summary, the meta systems is to manage efficiently organized load balancing strategy and agile sufficiently.

Therefore, a self-regulating load system is needed that can balance the load within the entities without any information about the system. This selforganized regulation can be distributed by distributed algorithms. To address this, we propose an implementation algorithm to manage the dynamic environment of virtual machine programming and requests in cloud computing. To balance the load in a cloud computing environment, tasks should be sent to the under loaded virtual machine, similar to how foraging bees send consecutive tasks to a flower patch until it is full. By taking inspiration from honeybee behavior, we can improve the overall efficiency of the system in load balancing. Prioritybased balancing focuses mainly on decreasing the time spends of task in a virtual machine's queue, similar to how honey bees prioritize certain flower patches to maximize efficiency. This approach reduces the response time of virtual machines.

Honeybee foraging behavior in Load Balancing

To achieve load balancing, it is recommended to allocate tasks to the less loaded virtual machine. This strategy is analogous to how foraging bees direct sequential tasks to a flower patch until it reaches capacity. By emulating the behavior of honeybees, the efficiency of the load balancing process can be improved. The priority-based balancing method concentrates on minimizing the duration that a task spends in the queue of a virtual machine, which is similar to how honeybees select particular flower patches to maximize productivity. By adopting this technique, the time of response in virtual machines can be reduced.

The algorithm of Artificial Bee Colony begins by pairing each bee with randomly selected food sources, without any bias. The bees then gather nectar from these sources and return to the hive,
sharing their findings with other bees through dance. Next, a population of scout bees is established and randomly sent out to the food sources to calculate their fitness values.

Bees that are currently employed collect and store information about food sources in their vicinity. They then share this information through dance to other bees. Afterward, the employed bees revisit previously visited food sources and use the shared information to select new sources in the area. The scout bees based on nectar information they choose food which information is provided by the employed bees. The onlooker bees in the hive receive the information from the working bees about food source, and then choose one of the sources to visit. These bees wait for a dance performed by the employed bees to determine which food source to choose. The employed bees perform a specific type of dance, such as the waggle or tremble dance, to communicate about the quality, quantity, and distance of the food source from the hive. To find a food source scout bee conducts a random search, and when the bees no longer visit a particular source, the scout bee randomly selects a new source to investigate.

Algorithm: -

1.The user provides a list of available virtual resources in the data center (e.g., VM1, VM2… VMn.) and a list of tasks to be completed (T1, T2 … Tn.).

2.When a new request is received, the scheduler computes the computing capacity that is expected to be needed for the given tasks.

3.The average computing capacity for each task is determined using an equation - $PT_{host} = \frac{Total\ length\ of\ tasks\ submitted\ to\ host}{number\ of\ processors\ in\ host * processor\ speed\ of\ host}$

4.The load on each virtual machine (VM) is found -
Load of each VM = $\frac{Number\ of\ tasks\ at\ time\ t\ on\ service\ queue\ of\ VM}{service\ rate\ of\ VMi\ at\ time\ t}$

5.The average system load is computed.

6.The load is checked against a probability value ranging from 0 to 1, and if the value is between 0 and 1, the VM is added to the underloaded list; otherwise, it is added to the overloaded list. if (0< P()<1):

```
        Underloaded_list[]= VMi else:
      Overloaded_list[]= VMi
```

7.The underloaded VMs are selected and compared to the expected computing power of the tasks. If the average computing capacity of the VM is greater than or equal to the expected computing power of the tasks, the VM is marked as the fittest and the tasks are allocated to it.

8.After the tasks are allocated, some VMs may remain underutilized, resulting in wasted processor time. Randomization is performed in the system on certain threshold value or less than to it. VMs with a system load greater than another threshold value is selected, and jobs are randomly allocated to VMs with a system load less than or equal to the first threshold value. The algorithm will make N-1/N attempts on average to reallocate jobs among the VMs.

The flowchart shows that path how a task is being processed step by step uniformly. The flowchart consist of 4 sub parts i.e. workflow submission, taskin, task-out, delay

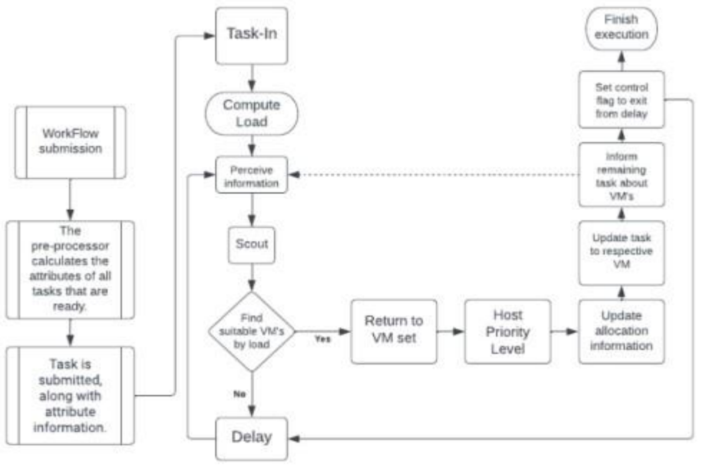

Task-In
Compute Load
Perceive information
Scout
Find suitable VM's by load
Yes
No
Delay
Return to VM set
Host Priority Level
Update allocation information
Update task to respective VM
Inform remaining task about VM's
Set control flag to exit from delay
Finish execution
WorkFlow submission
The pre-processor calculates the attributes of all tasks that are ready.
Task is submitted, along with attribute information.

Fig 2: Flow diagram for the behavior of the proposed algorithm

IV. RESULTS

Cloud Analyst is used to stimulate task scheduling and model. The experimental evaluations are conducted using a simulation platform consisting of 50 VMs, Data centers and 100-1000 tasks. The tasks' length varies between 1000 to 2000million Instructions (MI) , while the speed of processer, memory size in VM, and bandwidth dictate the allowable load to each VM.

Table 1: Simulation Parameters

PARAMETERS	VALUES
Total no of tasks	100-1500
Length of task	2000-20000 bytes
No of VM's	50
Processor speed	500 - 10000
VM Memory size	256 - 2048 MB

In order to customize the simulation outcomes, an examination was conducted on the impact of various parameters. These parameters include the average response time, the length of executable instructions, and the average frequency of user requests per hour.

Fig 3 presents the average response time of algorithm, varying the length of executable instructions for each task between 2000 and 20000 bytes, and testing at different numbers of tasks. The cumulative response time of task measures the duration between request submission and the initial response, in milli-seconds. Results show that, under all conditions, the average response time begins at approximately 51.67 seconds. However, this value slightly increases to 52.05 seconds when the executable instruction length for a task is 2000 bytes and increases to 52.89 seconds when it is 20000 bytes. This is attributed to the fact that an increase in the tasks and as well as the length of instruction will impact the load of system, resulting in increased response times.

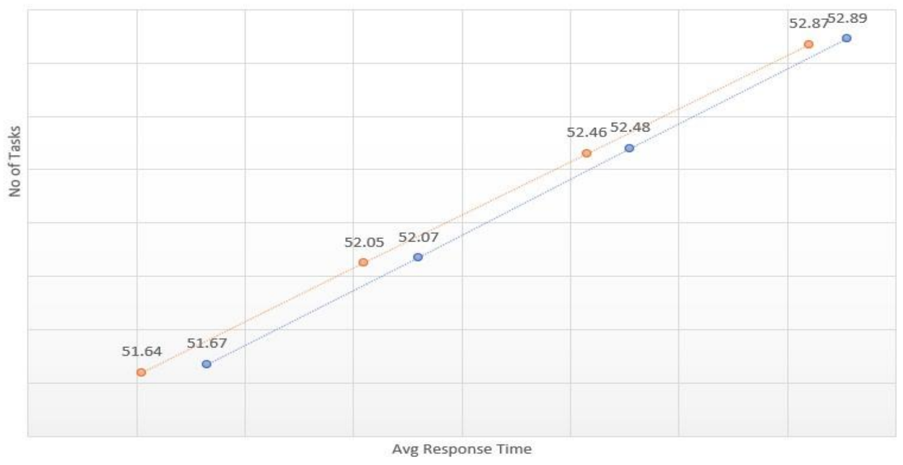
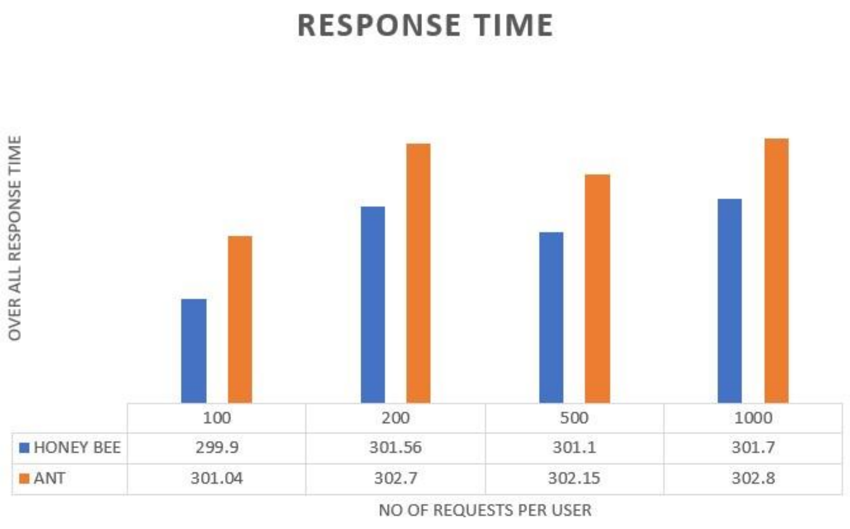


Fig 3: The study measures the Response time of Honey-Bee algorithm for different values of instruction length, while varying number of tasks

Comparison with swarm-based ant colony algorithm:

This compares simulation results of proposed algorithm with swarm-based ant colony algorithm. As noted in the previous section, the performance of the system is significantly impacted by the executable instruction size and no of tasks. Therefore, to demonstrate the impact of proposed algorithm a high default value for the tasks is chosen.

Fig 4 represents the average response time in proposed honeybee algorithm with ant colony load balancing algorithm. The tasks are varied in between 100 to 1000. The instruction size for these tasks vary randomly between 2000 and 20000. It is shown that the proposed honeybee algorithm saves up overall response time compared to ant colony algorithm. This is because, when assigning tasks to VMs, the proposed algorithm considers factors such as the availability of VMs, least load including load variation on each VM.



	100	200	500	1000
■ HONEY BEE	299.9	301.56	301.1	301.7
■ ANT	301.04	302.7	302.15	302.8

Fig 4: Average response time versus number of tasks

Fig 5 represents the average data processing time of proposed honeybee algorithm with ant colony algorithm. The tasks change from 100 to 1000. As it is shown that the proposed honeybee algorithm performs computation faster thereby reducing the vm cost the data transfer cost as well.

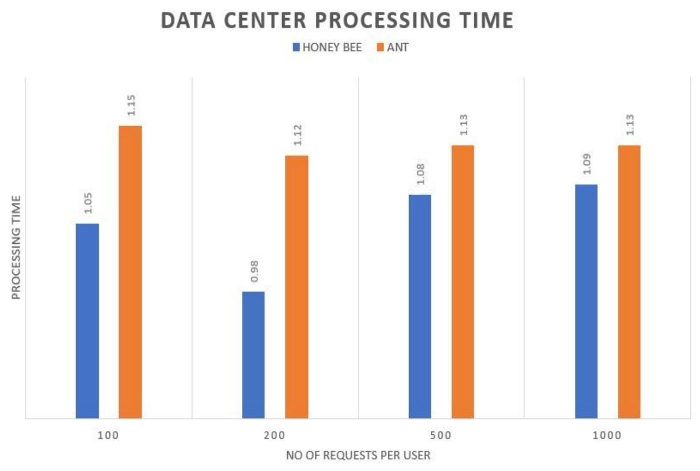


Fig 5: Data processing time verses no of tasks

V. CONCLUSION

The paper proposes a load balancing approach inspired by foraging strategy through behavior of honeybee in cloud system. The algorithm stabilizes the load by removing heavily loaded virtual machines while also considering the priorities of tasks. The honeybees are treated to be tasks to globally update the information. The algorithm also prioritizes tasks to improve the overall processing throughput and reduce the response time of VMs. When this proposed algorithm is set side by side to other existing techniques, and results show its efficiency without additional overheads. This technique is suitable for balancing independent tasks that are non-primitive and heterogeneous cloud computing systems.

REFERENCES

[1] P. Kumar and R. Kumar, ``Issues and challenges of load balancing techniques in cloud computing: A survey," *ACM Comput. Surveys*, vol. 51, no. 6, pp. 1_35, Feb. 2019.

[2] Ajil, A., Kumar, A Comprehensive Study of LB Technique in Cloud Infrastructure. SN COMPUT. SCI. 4, 181, 2023.

[3] Ali, Mrs Anooja, and D. P. Sumalatha. "A survey on Balancing the Load of Big Data for Preserving Privacy Access in Cloud." Asian Journal of Engineering and Technology Innovation (AJETI) (2018): 176.

[4] Kumar Nishant, Pratik Sharma, Vishal Krishna "Load Balancing of Nodes in Cloud Using Ant Colony Optimization" 14th International Conference on Modelling and Simulation, 2012.

[5] Shalu Rani, Dharminder Kumar, Sakshi Dhingra, "A review on dynamic load balancing algorithms", 2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), pp.515-520, 2022.

[6] Sansanwal, Suman and Jain, Nitin, Survey on Existing Load Balancing Algorithms in Cloud Environment (July 12, 2021). Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2021.

[7]"Load Balancing Algorithms For Cloud Computing Environment ", International Journal of Emerging Technologies and Innovative Research (www.jetir.org | UGC and issn Approved), ISSN:2349-5162, Vol.7, Issue 5, page no. pp59-67, May-2020.

[8] Kaur, M. Aron, A Systematic study of load balancing approach in the fog computing environment. J Supercomput 77,9202-9247 (2021).