International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-08, Issue-11, Feb 2023

Response Time Optimization in Cloud Using Artificial Honey Bee Colony Inspired Load Balancing Strategy

Brototi Mondal, Assistant Professor, Sammilani Mahavidyalaya, University of Calcutta, Kolkata, India, brototi.snp@gmail.com

Abstract: Cloud computing deals with massive amount of data transfer to and from the server, because it involves real-time resource provisioning. Most businesses are moving their operations to the cloud due to flexibility. Service providers are building new data centers to accommodate the growing demand for their services from customers. The majority of the resources are virtualized, and virtual machines (VMs) are one of the key components of virtualization. However, owing to the task allocation system in the VM, user tasks sent to the cloud might result in the VM being underloaded or overloaded, which would cause the system to fail or cause user actions to be delayed. A huge number of services are offered, and choosing or combining those services is NP-hard optimization issue. Numerous metaheuristic methods have been employed thus far due to the NP-hard complexity of service composition. This research proposes a novel approach for efficient load balancing and foraging activity to regulate load through VMs, based on an artificial honey bee colony. Response time, data center processing time, and other factors can be used to measure how successful a load balancing algorithm is. A load balancing technique inspired by the artificial honey bee colony algorithm is suggested and has demonstrated improved response time. On the standard data sets, experimental investigations using CloudAnalyst simulator reveals that the proposed strategy outperforms the existing load balancing techniques in terms of response time by a significant margin.

Keywords — Cloud Computing, Load Balancing, Artificial Honey Bee Colony, Virtual Machine, Response Time.

I. INTRODUCTION

Cloud computing is now a rapidly expanding technology that has evolved into a powerful paradigm for handling complicated issues. It is well-known as an Internet-based computing architecture in which a large number of cloud users share computer and virtual resources such services, storage, applications, servers, and networks. Cloud service providers (CSP) like Amazon, Google, Microsoft, and others carry out this operation. Depending on the service's level of abstraction, several levels of computing services are supplied to customers [1]. Infrastructure as a Service (IaaS), the most basic level of service, provides users with hardware on which they can deploy virtual machines, software platforms on which they can run their applications, and the application itself. Amazon EC2 Cloud, Netmagic IaaS, Tata Communications InstaCompute etc. are illustrations of IaaS service. Cloud users do not need to manage virtual machines at the next level. A software platform is already established in an infrastructure and made available to customers for hosting applications (usually, web apps). The platform is then utilized by users to create customized applications. This tactic is referred to as "Platform as a Service." Google App Engine, Force.com, Joyent, Azure etc. are examples of this situation. Programming languages, application frameworks, databases, and other tools are provided by PaaS providers. Next, Software as a Service (SaaS) model offers users an application without requiring them to manage the virtual machines and software platforms that host the program.

In a cloud context, virtual machines are thought of as mediators and processing units. Virtual machines (VMs) meet user demands for data access, and effective load balancing allows for effective usage of these VMs [2].

Load balancing has emerged as a crucial topic and a cutting-edge method to provide maximum throughput while reducing response time. In cloud computing, load balancing is a strategy for managing resources based on a maximum throughput with the shortest response time and for evenly distributing traffic between the server's data and various users without any lag. Load balancing evenly shares the load among servers to maintain system stability without too or inadequately loaded servers, enhancing resource efficiency. The load may consist of CPU, memory,

 DOI : 10.35291/2454-9150.2023.0023

or network demands. The load balancing approach got several requests from users to distribute them around virtual machines based on their accessibility [3]. If a node is overloaded means its load exceeds the threshold value, its burden is transferred to a node that is underloaded. A significant difficulty for cloud computing is locating the optimal solution for load balancing. Therefore, appropriate load balancing strategies must be used to cut down on response times overall without adding costs [4]. Therefore, load balancing approaches aid in attaining the optimum resource use, improving throughput, reducing response time, and preventing resource overload.

The primary focus of the study is on creating a time-efficient load balancing method that is inspired by an artificial honey bee colony algorithm to improve task scheduling with the aim of reducing makespan, concurrent resource utilization and cost. The remaining sections of the paper are arranged as follows. The related study of cloud computing load balancing is introduced in Section 2. The suggested load balancing algorithm for cloud computing is shown in Section 3. Section 4 introduces the simulation results and explains how the technique was implemented using CloudAnalyst simulator. Finally, Section 5 presents the conclusion.

II. RELATED WORK

This section summarizes numerous studies of load balancing techniques in the cloud environment conducted over recent years.

Min-Min and Min-Max are used to distribute each job in any order to the nodes where it is predicted to be completed the quickest by reducing the makespan and energy consumption, regardless of the present load on the node. Minimum Execution Time (MET), Minimum Completion Time (MCT) are also used [5]. A round-robin algorithm equitably allocates tasks among all data centers or processing units.

In a cloud computing setting, load balancing using a genetic algorithm was suggested [6]. The genetic algorithm comprises three stages: crossover, mutation, and selection. The algorithm's initial phase involves choosing a virtual machine (VM) depending on the cost and turnaround time of the scheduled jobs. In the second stage, the ideal costs and timings are determined in order to accomplish the crossover between the scheduled tasks and the virtual machine. The algorithm then modifies the available VM and the scheduled tasks before being approved for execution.

The metaheuristic algorithm ant colony optimization, which is a fundamental foraging habit of ants that pushed them to locate the best, quickest path from their nest to food, was proposed in [7] as a load balancing approach in cloud computing. In this case, a new request is assigned to virtual machines in accordance with the FCFS scheduling rules, however if virtual machines are not available to distribute the next work, a random number of ants with the same pheromone value is created and placed randomly to traverse. An ant selects a VM and then checks to see if it has finished its tour. The pheromone value is updated if the tour is finished. This keeps happening until the ideal VM is discovered.

In [8], the Bacterial Swarm Optimization (BSO) Algorithm was proposed for resource deployment and load balancing in data centers. The proposed BSO algorithm calculated a set of resources for each work using a separate set of jobs. This study identified the capability of local and global search and quick merging optimum points. This algorithm lowers operating expenses, increases efficiency, and better utilizes resources. However, the BSO method used a lot of vulnerable live migration and cloud data.

In order to save energy [9], the multi-objective genetic algorithm (MO-GA) was created, which placed an emphasis on encoding rules, crossover operators, selection operators, and the technique of sorting Pareto solutions. It also boosts service profitability when there are deadline constraints. It begins by suggesting a task scheduling architecture for cloud computing that consists of a range of components to assess the application and assign the appropriate resources to the applications in order to increase the effectiveness and efficiency of computing.

Based on examination of historical data from the resource utilization by the VMs, an online deterministic method and adaptive heuristic for dynamic consolidation of VMs was suggested in [10]. In conclusion, they have taken into account the prior level of resource use rather than using a fixed threshold. They have asserted that this strategy can lower data centers' energy usage. Additionally, for the purpose of identifying hosts that are overloaded, writers used methods like Median Absolute Deviation (MAD), Inter Quartile Range (IQR), Local Regression (LR), and Robust Local Regression (LRR). Once the overused host has been identified, they choose one or more VMs to move from it to the other hosts. It is notable that they suggested three various VM selection policies, including the random choice strategy, the maximum correlation policy, and the minimal migration time policy, to choose VMs from over or underutilized hosts. Finally, their approach looks into underutilized hosts after all over utilized hosts have been found and some of the VMs have been relocated. To do this, it views all hosts other than the ones that are overloaded as underutilized hosts. As a result, it tries to move virtual machines from underutilized hosts to other

 DOI : 10.35291/2454-9150.2023.0023

hosts while keeping in mind that the migration process shouldn't put the other hosts under excessive strain. The hosts are switched to sleep mode once all migrations of idle hosts have been finished.

The cloud is a centralized virtual computer where data is kept, and cloud service businesses are in charge of distributing the offerings to end customers [11]. The end customers get access to the offerings based on their requirements and must pay for the services received. To maximize the efficient use of resources and energy usage, load balancing becomes increasingly necessary as the amount of requests increases.

Cloud Light Weight (CLW), a balancing solution was introduced in[12], balances the burden of virtual machines while guaranteeing the users' quality of service. All nodes in their method have almost the same weight after applying CLW, and they balance workloads among all the hosts.

III. PROBLEM DEFINITION

Transferring workload from Virtual Machines (VMs) that are overloaded relative to other VMs which have received a less of work is the main goal of load balancing. Using load balancing, the system's overall performance can be attained. There are certain computer resources available at each data center to carry out user tasks. Numerous tasks are included in the various cloud users, and each task is given to a distinct VM. It is possible to determine the load on a VM based on how long each job takes to complete. If the VM is overloaded, the load is distributed to the VM that is underloaded in order to maximize resource use. The Scheduler has the ability to select the most appropriate VM and distribute the jobs among VMs in accordance with the chosen methodology. To balance the load, based on the least-used VM at the time the job is due, the scheduler assigns the jobs in the most appropriate VMs. At runtime, whenever the load balancer detects an idle or least loaded VM by using the resources' current status information, it selects how to migrate the workload from the heavily loaded VM to the idle or least loaded VM.

IV. PROPOSED ALGORITHM

In this paper, a time efficient load balancing strategy is used which is inspired by artificial honey bee colony algorithm. It is an optimization method that mimics honey bee foraging. The life style of honey bees can be demonstrated with two phases. First phase is discovering food source and second one is collecting food.

A. *Discovering the food source:*

Honey bees come in three different varieties like employed bees, onlooker bees, and scout bees. The employed bees use their memories to find food near the food source while also informing the onlooker bees about these food sources. From the food sources discovered by the employed bees, the onlooker bees frequently choose the best ones.

B. *Collecting food from the food source:*

The likelihood that the onlooker bees will choose the food source with greater quality (fitness) is much higher than the chance that they would choose the one with lower quality. The scout bees are derived from a small number of employed bees that leave their food sources and look for new ones.

The number of solutions is equal to the number of employed bees or onlooker bees. The algorithm creates an initial population of N solutions (food sources) that are spread at random. N stands for population size.

Let $P_i = \{p_{i,1}, p_{i,2}, p_{i,3}, \dots, p_{i,n}\}$ represent the i^{th} solution in the population, where n is the dimension size.

Every employed bee P_i generates a new candidate solution X_i in the neighborhood of its current position as equation 1,

$$x_{i,k} = p_{i,k} + \alpha_{i,k} \times \left(p_{i,k} - p_{j,k}\right) \quad \text{.....(1)}$$

where P_j is a candidate solution chosen at random and $(i \neq j)$ and k is a random dimension index selected from the set $\{1,2,3,\dots n\}$, and $\alpha_{i,k}$ is a random number within $[-1,1]$. A greedy selection is used after the generation of the new candidate solution X_i. If the fitness value of new candidate solution X_i is better than that of its parent P_i, then P_i is updated with X_i; otherwise keep P_i unchanged.

After all employed bees have finished their search, they waggle dance to tell any onlooker bees of their food sources. An onlooker bee assesses the nectar data collected from all employed bees and selects a food source with a probability based on the amount of nectar it contains. The probability Pro_i is equation 2,

$$Pro_i = \frac{fitness_i}{\sum_j fitness_j} \quad \text{.....(2)}$$

where $fitness_i$ is the fitness value of the i^{th} solution in the population. The scout bee finds a new food source to replace P_i, which was the abandoned source with the equation 3,

$$P_{i,k} = low_k + \delta_{i,k} \times (upp_k - low_k) \quad \text{.....(3)}$$

Let's consider m is the number of tasks provided by cloud customers that need to be distributed and q is the amount of VMs that are available in the cloud at any one moment. A Virtual Machine Vector (VMV) identifying the current

 DOI : 10.35291/2454-9150.2023.0023

level of VM usage will be present on each VM. $mips$ is a measurement of a machine's ability to process one million instructions in a second. φ stand for the relative execution cost and dc delay cost of an instruction. The delay cost is an estimated fine that the cloud service provider must give to the client if the project takes longer to complete than the service provider's announced deadline but is actually done sooner.

$$VMV = f(mips, \varphi, dc) \quad \text{.....(4)}$$

A request unit vector (RUV) may similarly be used to represent each request made by a cloud user. Here, *t* stands for the kind of service that a request requires, which can be Software as a Service (SAAS), Infrastructure as a Service (IAAS), or Platform as a Service (PAAS). NIC is the number of instructions in the request that have been counted by the processor. The worst case completion time tc is the shortest amount of time needed for a processing unit to finish a request, and the request arrival time (RAT) is the clock time at which the request enters the system.

$$RUV = f(t, NIC, RAT, tc) \quad \text{.....(5)}$$

The various properties of requests are therefore provided by equation 4

The cloud service provider must divide these *N* tasks across *M* processors in a way that minimizes the *C* fitness function as shown in equation 5.

$$fitness = w1 \times \varphi\,(NIC \div mips) + w2 \times dc \quad \text{.....(6)}$$

Where $w1$ and $w2$ are weights with values 0.8 and 0.2 respectively.

Proposed Algorithm:

Step 1: Create the initial population of processing unit $P_i = \{p_{i,1}, p_{i,2}, p_{i,3}, \cdots, p_{i,n}\}$.

Step 2: While the maximum number of iterations is reached or the optimum solution is discovered, do the following:

Step 3: Deploy Employed Bees and create a new food source at location $x_{i,k}$ using equation 1 in the neighborhood of $p_{i,k}$.

Step 4: Apply greedy selection method between $x_{i,k}$ and $p_{i,k}$ for neighborhood search of VM_i .

Step 5: Evaluate the fitness value using equation 5 and probability value for the solution $p_{i,k}$ using equation 2.

Step 6: Generate the new solution (new position) $x_{i,k}$ based on probability Pro_i for the onlooker bees from $p_{i,k}$.

Step 7: Apply greedy selection technique for onlooker bees between $x_{i,k}$ and $p_{i,k}$.

Step 8: Determine the exhausted sources using equation 3 , replace it with

new randomly produced solutions $x_{i,k}$ by sending scout bees.

Step 9: Output the optimal food source solution. Assign task to underloaded VM_i.

Step 10: End of while loop.

V. SIMULATION AND RESULTS

Extensive experiments are conducted with CloudAnalyst simulation toolkit which is modified with the proposed algorithm. The CloudAnalyst also makes it quick and simple to repeatedly run a series of simulation trials with minor parameter modifications. Table 1 contains the simulation setup. The six areas that make up the world's six major continents are modeled after a group of users. It is expected that 5% of all registered users are online at once during peak hours, and that only 10% are online during off-peak hours. VMs utilized in the experiment to host apps are 100MB in size. VMs have 10MB of accessible bandwidth and 1GB of RAM capacity. X86 architecture is used in simulated hosts. A certain number of VMs specifically dedicated to the application are hosted by each virtual data center. The machines contain 100GB of storage and 4 GB of RAM. Each machine has four CPUs, each with a 10000 MIPS capacity.

Several simulation scenarios are considered for experimentation. The experiment begins by using a single data center (DC) with 25, 50, and 75 virtual machines (VMs) to handle all requests from across the world. In Table 2, three different Cloud Configurations (CC) were taken into account to determine different response times. Six potential Cloud Configurations (CC) were considered in Table 3 to ascertain the various response times. Also, in Table 4, Table 5, Table 6 and Table 7, four distinct Cloud Configurations (CC) were considered in order to calculate response times for various load balancing algorithms.

Table 1: Simulation setup

S.No	User Base	Region	Online users during peak hrs.	Online users During off-peak hrs.
1.	UB1	N.America	4,70,000	80,000
2.	UB2	S.America	6,00,000	1,10,000
3.	UB3	Europe	3,50,000	65,000
4.	UB4	Asia	8,00,000	1,25,000
5.	UB5	Africa	1,25,000	12,000
6.	UB6	Oceania	1,50,000	30,500

 DOI : 10.35291/2454-9150.2023.0023

The determined overall average Response Time (RT) in ms for the Artificial Honey Bee Colony, Stochastic Hill Climbing, Round Robin algorithms are provided in Table 2 with one data center containing 25, 50, 75 VMs respectively. Figure 1 shows the performance analysis graph for it, with cloud configuration along the x-axis and response time in milliseconds along the y-axis. Then, as shown in Tables 3, 4, 5, 6 and 7, two, three, four, five and six DCs are taken into consideration with combinations of 25, 50, and 75 VMs for each Cloud Configuration. Figures 2, 3, 4, 5 and 6 show the related performance analysis graphs next to them. Table 2 depicts the average response time for single Data Center with 25, 50 and 75 VMs using the proposed algorithm and existing algorithms. The response time for two Data Centers with all possible combination of 25, 50, and 75 VMs is shown in Table 3. The average response time for three, four, five, and six data centers with all possible combinations of 25, 50, and 75 VMs are shown in Tables 4, 5, 6, and 7.

Table.2.Simulation setup and computed overall average response time (RT) in (ms) using One DC

Sl. No	CC	DC specification	RT in ms For AHBC	RT in ms For SHC	RT in ms for RR
1.	CC1	One DC with 25 VMs	327.6	329.02	330.05
2.	CC2	One DC with 50 VMs	327	329	329.55
3	CC3	One DC with 75 VMs	326.55	329.34	329.44

The graph in Figure 1 demonstrates that the suggested Artificial Honey Bee Colony algorithm has faster response time than the Round Robin and Stochastic Hill Climbing algorithms that are already in use. Simulation results illustrate that the proposed Artificial Honey Bee Colony algorithm have a faster response time than the already existing Round Robin and Stochastic Hill Climbing algorithms, as shown in Figure 2, Figure 3, Figure 4, Figure 5 and Figure 6.

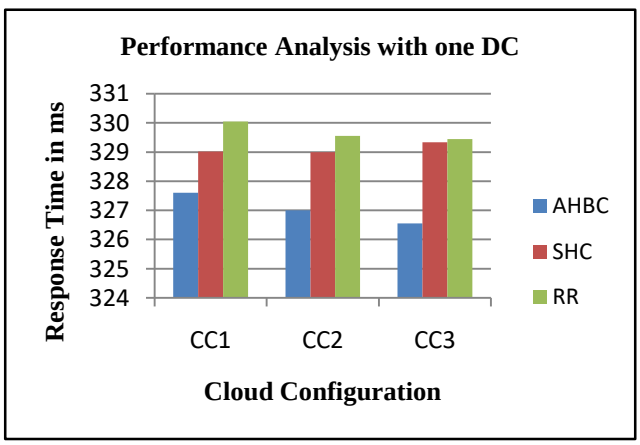

Fig.1. Performance analysis of proposed AHBC with SHC and RR Result using One DC

Table.3.Simulation scenario and calculated overall average response time (RT) in (ms) using Two DCs

Sl. No	CC	DC specification	RT in msFor AHBC	RT in msFor SHC	RT in ms for RR
1.	CC1	Two DC with 25 VMs each	360.30	365.44	376.34
2.	CC2	Two DC with 50 VMs each	356.44	360.15	372.52
3	CC3	Two DC with 75 VMs each	355.10	359.73	370.56
4	CC4	Two DC with 25,50 VMs	350.05	356.72	368.87
5	CC5	Two DC with 25,75 VMs	353.04	357.23	367.23
6	CC6	Two DC with 50,75 VMs	354.06	357.04	361.01

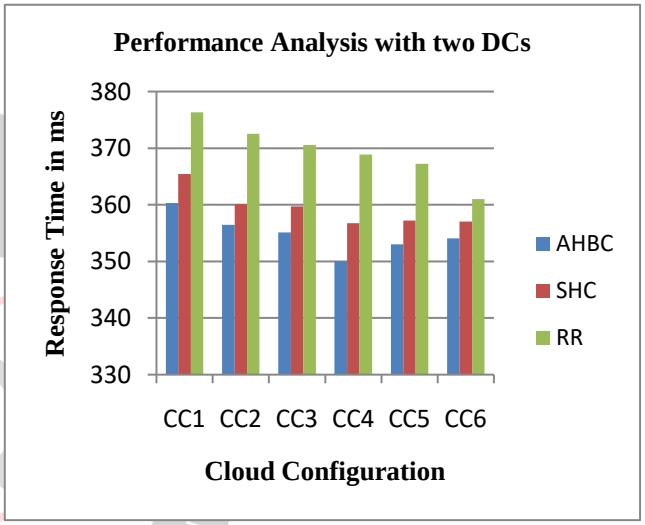

Fig.2. Performance analysis of proposed AHBC with SHC and RR Result using Two DCs

Table.4.Simulation scenario and calculated overall average response time (RT) in (ms) using Three DCs

Sl. No	CC	DC specification	RT in ms For AHBC	RT in ms For SHC	RT in ms for RR
1.	CC1	DC with 25 VMs each	350.10	356.82	363.34
2.	CC2	DC with 50 VMs each	351.15	355.25	363.52
3	CC3	DC with 75 VMs each	346.05	350.73	361.56
4	CC4	DC with 25,50,75 VMs	344.95	350.01	360.87

 DOI : 10.35291/2454-9150.2023.0023

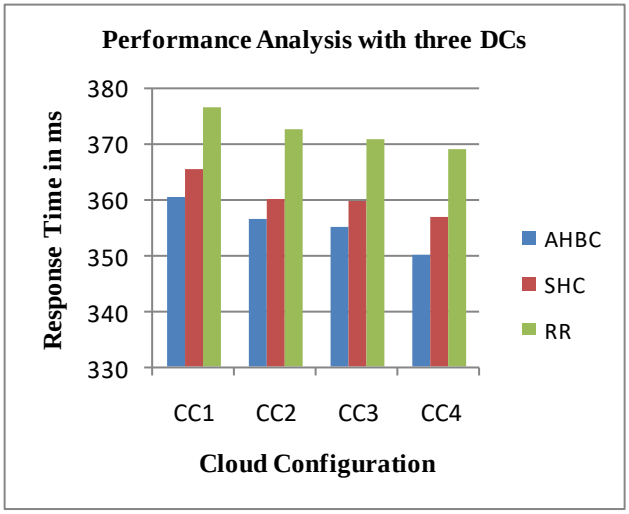

Fig.3. Performance analysis of proposed AHBC with SHC and RR Result using Three DCs

Table.5.Simulation scenario and calculated overall average response time (RT) in (ms) using Four DCs

Sl. No	CC	DC specification	RT in ms For CS	RT in ms For SHC	RT in ms for RR
1.	CC1	DC with 25 VMs each	348.23	354.35	360.95
2.	CC2	DC with 50 VMs each	345.08	350.71	359.97
3	CC3	DC with 75 VMs each	340.11	346.46	358.44
4	CC4	DC with 25,50,75VMs	336.76	344.31	355.94

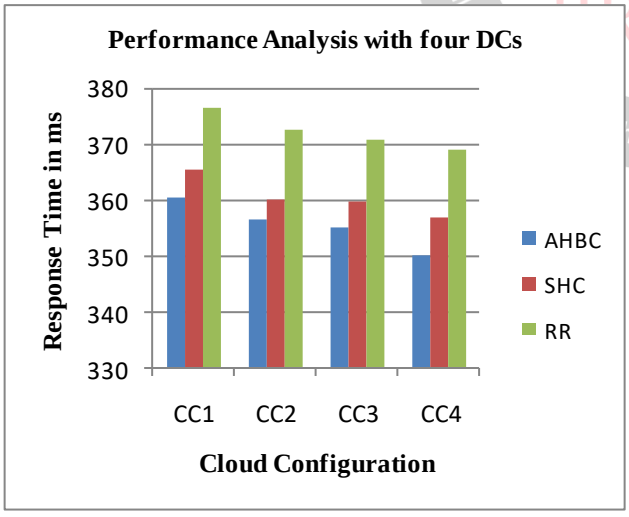

Fig.4. Performance analysis of proposed AHBC with SHC and RR Result using Four DCs

Table.6.Simulation scenario and calculated overall average response time (RT) in (ms) using Five DCs

Sl. No	CC	DC specification	RT in ms For CS	RT in ms For SHC	RT in ms for RR
1.	CC1	DC with 25 VMs each	336.45	342.86	352.05
2.	CC2	DC with 50 VMs each	326.30	332.84	345.44
3	CC3	DC with 75 VMs each	320.54	329.46	342.79
4	CC4	DC with 25,50,75VMs	318.35	326.64	338.01

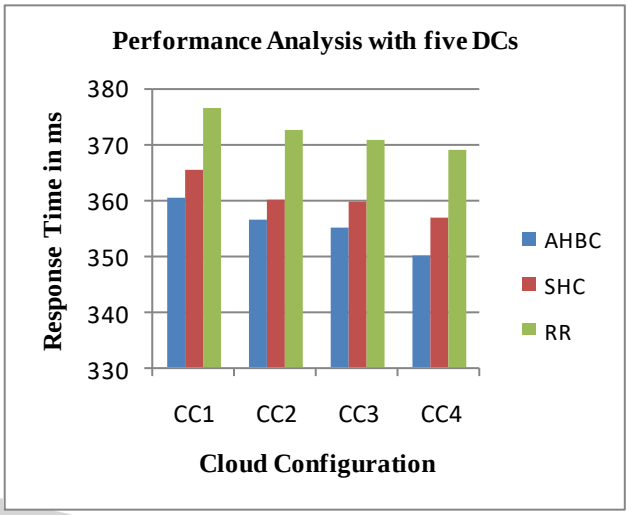

Fig.5. Performance analysis of proposed AHBC with SHC and RR Result using Five DCs

Table.7.Simulation scenario and calculated overall average response time (RT) in (ms) using Six DCs

Sl. No	CC	DC specification	RT in ms For AHBC	RT in ms For SHC	RT in ms for RR
1.	CC1	DC with 25 VMs each	330.32	336.96	349.26
2.	CC2	DC with 50 VMs each	324.24	331.56	344.04
3	CC3	DC with 75 VMs each	319.09	327.78	339.87
4	CC4	DC with 25,50,75VMs	316.35	323.56	338.29

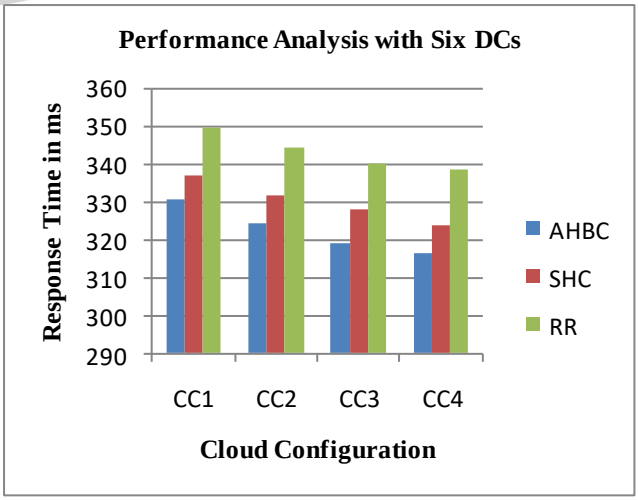

Fig.6. Performance analysis of proposed AHBC with SHC and RR Result using Six DCs

 DOI : 10.35291/2454-9150.2023.0023

VI. CONCLUSION

An Artificial Honey Bee Colony optimization technique, which addresses the load balancing issue in cloud computing, is presented in this study. This strategy adheres to the search process for locating the best VM for load shifting. The iteration process is taken into account by the suggested artificial honey bee colony technique for effective work allocation to the VMs. During the iteration process, it is determined whether or not the VM is overloaded. When compared to the existing techniques, the experimental assessment of the suggested model performs better in terms of minimizing average response time. The results of a detailed investigation show that the suggested load balancing strategy not only performs better than a few existing techniques, but also ensures that the QoS requirements of the customer job are met. Although here all jobs are anticipated to have the same priority but this may not be the actual situation. Fault tolerance issues are not taken into account here. Researchers can continue by include fault tolerance and various function variations while calculating the fitness function, which can then be used for additional research projects.

REFERENCES

[1] Srinivasan, Rashmi KrishnaIyengar, V. Suma, and Vaidehi Nedu. "An enhanced load balancing technique for efficient load distribution in cloud-based IT industries." *In Intelligent Informatics: Proceedings of the International Symposium on Intelligent Informatics ISI'12* Held at August 4-5 2012, Chennai, India, pp. 479-485. Springer Berlin Heidelberg, 2013.

[2] Buyya, Rajkumar, Rajiv Ranjan, and Rodrigo N. Calheiros. "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services." *In Algorithms and Architectures for Parallel Processing: 10th International Conference, ICA3PP 2010*, Busan, Korea, May 21-23, 2010. Proceedings. Part I 10, pp. 13-31. Springer Berlin Heidelberg, 2010.

[3] Mohammadian, Vahid, Nima Jafari Navimipour, Mehdi Hosseinzadeh, and Aso Darwesh. "Fault-tolerant load balancing in cloud computing: A systematic literature review." *IEEE Access 10*: 12714-12731, 2021.

[4] uz Zaman, Sardar Khaliq, Tahir Maqsood, Mazhar Ali, Kashif Bilal, Sajjad Ahmad Madani, and A. U. R. Khan. "A load balanced task scheduling heuristic for large-scale computing systems." *Comput. Syst. Sci. Eng* 34 : 4, 2019.

[5] Armstrong, Robert, Debra Hensgen, and Taylor Kidd. "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions." *In Proceedings Seventh Heterogeneous Computing Workshop (HCW'98)*, pp. 79-87. IEEE, 1998.

[6] Dasgupta, Kousik, Brototi Mandal, Paramartha Dutta, Jyotsna Kumar Mandal, and Santanu Dam. "A genetic algorithm (ga) based load balancing strategy for cloud computing." *Procedia Technology* 10 : 340-347, 2013.

[7] Dam, Santanu, Gopa Mandal, Kousik Dasgupta, and Paramartha Dutta. "An ant colony based load balancing strategy in cloud computing." *In Advanced Computing, Networking and Informatics-Volume 2: Wireless Networks and Security Proceedings of the Second International Conference on Advanced Computing, Networking and Informatics (ICACNI-2014)*, pp. 403-413. Springer International Publishing, 2014.

[8] Jeyakrishnan, V., and P. Sengottuvelan. "A hybrid strategy for resource allocation and load balancing in virtualized data centers using BSO algorithms." *Wireless Personal Communications*, 94: 2363-2375, 2017.

[9] Sun, Hong, Shi-ping Chen, Chen Jin, and Kai Guo. "Research and simulation of task scheduling algorithm in cloud computing." *TELKOMNIKA Indonesian Journal of Electrical Engineering* 11, no. 11: 6664-6672, 2013.

[10] Beloglazov, Anton, and Rajkumar Buyya. "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers." *Concurrency and Computation: Practice and Experience* 24, no. 13: 1397-1420, 2012.

[11] Karki, Sheetal, and Anshika Goyal. "Performance evaluation of check pointing and threshold algorithm for load balancing in cloud computing." *Int J Comput Sci Eng* 6, no. 5 : 2347-2693, 2018.

[12] Mesbahi, Mohammadreza, Amir Masoud Rahmani, and Anthony Theodore Chronopoulos. "Cloud light weight: A new solution for load balancing in cloud computing." *International Conference on Data Science & Engineering (ICDSE)*, pp. 44-50. IEEE, 2014.

 DOI : 10.35291/2454-9150.2023.0023