International Journal of Advanced Engineering and Technology
www.allengineeringjournal.com
ISSN: 2456-7655
Received: 26-02-2022, Accepted: 11-03-2022, Published: 28-03-2022
Volume 6, Issue 1, 2022, Page No. 38-46

A honey bee foraging algorithm based load balancing strategy in cloud computing

Brototi Mondal
Assistant Professor, Department of Computer Science, Sammilani Mahavidyalaya, University of Calcutta, Kolkata, West Bengal, India

Abstract
Since real-time resource provisioning is a component of cloud computing, it deals with enormous amounts of data transit to and from the server. Due to flexibility, the majority of firms are shifting their operations to the cloud. In order to meet the rising demand for their services from clients, service providers are constructing additional data centers. Virtual machines (VMs) are one of the essential elements of virtualization, and the bulk of the resources are virtualized. The task allocation system in the VM may cause the VM to be underloaded or overloaded as a result of user tasks transmitted to the cloud, which would result in system failure or cause user operations to be delayed. There are a great number of services available and selecting or combining them is an NP-hard optimization problem. The complexity of system composition is NP-hard, hence numerous metaheuristic strategies have been employed up to this point. Based on the foraging habit of honey bees, this paper proposes a novel method for effective load balancing and foraging activity to manage load through VMs. A load balancing algorithm's effectiveness may be evaluated using response time, data centre processing time, and other variables. A method for balancing load that draws inspiration from honey bee foraging activity is proposed, and it has been shown to enhance reaction time. Experimental research utilizing the Cloud Analyst simulator shows that the suggested method significantly surpasses the current load balancing strategies in terms of reaction time when applied to the standard data sets.

Keywords: Cloud computing, load balancing, honey bee foraging, virtual machine, response time

Introduction

Today's fast developing technology, cloud computing, has become a potent paradigm for resolving challenging problems. A vast number of cloud users share computer and virtual resources such services, storage, applications, servers, and networks in this well-known Internet-based computing architecture. This is done by cloud service providers (CSP), including companies like Amazon, Google, Microsoft, and others. Customers can choose from a variety of computing services, depending on the abstraction level of the service [1]. The most fundamental level of service, infrastructure as a service (IaaS), gives consumers access to hardware for deploying virtual machines, software platforms for running their applications, and the application itself. IaaS services include those from Amazon EC2 Cloud, NetmagicIaa S, Tata Communications Insta Compute, etc. Users of the cloud do not require advanced virtual machine management. Customers have access to a software platform that has already been built into an infrastructure and is used to host applications, often web apps. Users use the platform to build individualized apps. The strategy used here is known as "Platform as a Service." Examples of this circumstance include Google App Engine, Force.com, Joyent, Azure, and others. PaaS companies offer databases, application frameworks, programming languages, and other resources. Next, the Software as a Service (SaaS) approach provides consumers with an application without needing them to handle the management of the virtual machines and software hosting the programme.

Virtual computers are viewed as middlemen and processing units in a cloud environment. Users' requests for data access are met by virtual machines (VMs), and efficient load balancing enables the best utilization of these VMs [2].

A key issue and cutting-edge technique to deliver maximum throughput while slashing reaction time is load balancing. Load balancing is a technique used in cloud computing to manage resources based on maximum throughput with the quickest reaction time and for equally and lag-free distributing traffic between the data on the server and various users. Resource efficiency is improved through load balancing, which equally distributes the load among servers to preserve system stability without too or underloaded servers. Demands on the CPU, memory, or network might make up the load. Users frequently asked the load balancing method to distribute them around virtual machines according to their accessibility [3]. A node that is overloaded transfers its burden to a node that is underloaded if its load is greater than the threshold value. Finding the best load balancing solution is a big challenge for cloud computing. As a result, effective load balancing techniques must be employed to reduce reaction times as a whole without raising expenses [4]. In order to achieve the best resource utilisation, increase throughput, decrease response times, and avoid resource overload, load balancing techniques are used.

The study's main objective is to develop a time-efficient load balancing system based on the foraging activity of honey bees in order to enhance job scheduling and decrease makespan, concurrent resource usage, and cost. The order of the paper's remaining sections is as follows. In Section 2, the related research of cloud computing load balancing is presented. Section 3 discusses load balancing in cloud computing, while Section 4 defines the issue. In Section 5, the proposed load balancing method for cloud computing is displayed. The simulation findings are introduced in Section 6 along with an explanation of how the approach was applied using the Cloud Analyst simulator. The conclusion is then presented in Section 7.

Related work

Numerous researches on load balancing methods in the cloud environment have been compiled in this part over the past several years.

Regardless of the current load on the node, Min-Min and Min-Max are used to distribute each work in any order to the nodes where it is anticipated that it would be completed the quickest by cutting the makespan and energy consumption. Also utilised are Minimum Execution Time (MET) and Minimum Completion Time (MCT) [5]. Task distribution across all processing units or data centers is equitable thanks to a round-robin mechanism.

It has been recommended to use a genetic algorithm for load balancing in cloud computing environments [6]. Three phases make up the genetic algorithm: crossover, mutation, and selection. The initial step of the algorithm entails selecting a virtual machine (VM) based on the price and turnaround time of the scheduled jobs. The appropriate costs and timings are established in the second step in order to complete the transition between the scheduled tasks and the virtual machine. Before being given the go-ahead to execute, the algorithm adjusts the scheduled tasks and the available VM.

Ant colony optimisation, a metaheuristic algorithm, was presented in [7] as a load balancing method for cloud computing. Ant colony optimisation is a key foraging behaviour of ants that drives them to choose the best, fastest path from their nest to food. If virtual machines are not available to distribute the next job, a random number of ants with the same pheromone value are formed and put randomly to traverse. In this scenario, a new request is distributed to virtual machines in line with the FCFS scheduling rules. An ant chooses a VM and then determines whether the tour is complete. If the tour is over, the pheromone value is updated. This continues until the perfect VM is found.

The Bacterial Swarm Optimisation (BSO) Algorithm was put out in [8] as a method for load balancing and resource distribution in data centres. Using a different set of tasks, the suggested BSO algorithm computed a set of resources for each work. This investigation revealed the potential for speedy merging optimal spots and local and worldwide search. This method improves resource utilisation, boosts efficiency, and decreases operational costs. But the BSO approach made extensive use of cloud and live migration data that was highly sensitive.

The multi-objective genetic algorithm (MO-GA), which prioritised encoding rules, crossover operators, selection operators, and the method of sorting Pareto solutions, was developed to conserve energy [9]. When there are deadline restrictions, it also increases service profitability. It starts by offering a task scheduling architecture for cloud computing that comprises of a variety of elements to evaluate the application and provide the proper resources to the applications in order to improve computing effectiveness and efficiency.

A dynamic consolidation technique for VMs using an online deterministic approach and an adaptive heuristic was proposed in [10] on the basis of an analysis of historical data from the resource usage by the VMs. Finally, rather of utilising a predetermined threshold, they have taken into account the previous degree of resource utilisation. They claim that this approach can reduce the amount of energy used by data centres. Additionally, authors employed techniques like Median Absolute Deviation (MAD), Inter Quartile Range (IQR), Local Regression (LR), and Robust Local Regression (LRR) in order to identify hosts that are overloaded.

They decide which VMs to transfer from the overused host to the other hosts when it has been detected. The fact that they recommended three different VM selection rules, including the random choosing method, the maximum correlation policy, and the shortest migration time policy, to choose VMs from over or underutilised hosts, is noteworthy. After identifying all overused hosts and moving some of the VMs, their methodology then looks at underutilized hosts. To achieve this, it considers all hosts aside from those that are overwhelmed as underutilized hosts. As a consequence, it makes an effort to transfer virtual machines from idle servers to other hosts while keeping in mind that the other hosts shouldn't be overly taxed during the migration process. Once all migrations of idle hosts are complete, the hosts are put into sleep mode.

Data is stored in the cloud, a centralised virtual computer, and cloud service providers are in charge of providing their services to end users [11]. The end users must pay for the services they receive and are granted access to the offers depending on their needs. As the volume of requests rises, load balancing becomes more and more important in order to maximize the effective use of resources and energy utilization.

In [12], a balancing solution known as Cloud Light Weight (CLW) was unveiled. It distributes the load among virtual computers while ensuring the users' quality of service. After using CLW, all nodes in their approach have almost the same weight, and they evenly distribute workloads across all hosts.

Load Balancing In Cloud Computing

This study intends to deliver effective performance of external services and long-term load balancing of cloud data centers. The data center is often located distant from the end consumers. The components of a cloud environment that are accessible through the various internet hosting apps are distributed servers. Effective load balancing and scheduling among the nodes in the cloud environment are necessary to deliver higher Quality of Service (QoS) and effective execution of external services. An effective load balancing method works to reduce system imbalance, shorten job execution times, and give users equitably quick response times. A better load balancing and scheduling technique prevents data centers from being overloaded or underloaded. To increase QoS parameters like effective response time, resource usage, scalability, and task migration time, tasks that are overloaded with many tasks on certain VMs are transferred to the underloaded VMs in the same data centers. A technique for distributing workload over a number of servers, network ports, hard drives, or other computer resources is called load balancing. Typical datacenter implementations rely on sizable, potent (and expensive) computing hardware and network infrastructure, which are vulnerable to the standard risks associated with any physical device, including hardware failure, power and/or network outages, and resource limitations during periods of high demand. By leveraging commodity servers to carry out the load balancing, load balancing in the cloud varies from traditional thinking on load-balancing design and execution. This presents both fresh opportunities and economies of scale in addition to its own special set of difficulties.

1. Load balancing algorithms

A load balancing system is primarily made up of three components: the web customer service terminal of the SaaS layer makes requests, the cloud platform of the IaaS layer implements the collection of resources, and the PaaS layer implements the load balancing strategy. The OpenStack cloud platform is mostly utilized to implement the underlying virtualization at the IaaS layer, while Ganglia is primarily used for the monitoring function to track and gather data on each node's resources. The primary component of load balancing is the PaaS layer, which is accomplished by load balancing based on the dynamic load balancing algorithm discussed in Section 3. WSO2 software is used at this layer to implement the dynamic scheduling of virtual machines on the nodes. The PaaS layer is mostly used to create scheduling and load balancing techniques. The queue manager and the strategy controller make up the core processing module. The selection of a strategy is mostly based on an analysis of the server's load statistics. The database may be used to analyze whether real machines and virtual machine resources are overloaded, underloaded, etc. and store the resulting data, which then has to be accessed during scheduling. The queue manager mostly queues data, though.

The primary component of load balancing is the PaaS layer, which is accomplished by load balancing based on the dynamic load balancing algorithm discussed in Section 3. WSO2 software is used at this layer to implement the dynamic scheduling of virtual machines on the nodes. The PaaS layer is mostly used to create scheduling and load balancing techniques. The queue manager and the strategy controller make up the core processing module. The selection of a strategy is mostly based on an analysis of the server's load statistics. The database may be used to analyze whether real machines and virtual machine resources are overloaded, underloaded, etc. and store the resulting data, which then has to be accessed during scheduling. The queue manager mostly queues data, though.

As a result, the dynamic burden balancing approach is a great choice. The shortcomings of static payload trading are solved by the animated workload trading algorithm. The program strikes a balance by observing each server's source condition and scrutinizing each node's load capacity adjustment.

This article selects a dynamic load balancing method to achieve resource load balancing based on the study and comparison of various load balancing techniques mentioned above. The basis for load balancing is the gathering of PM (physical machine) and VM (virtual machine) data from each node in the cluster. This is because algorithm balancing may be finished through each node's loading function, and load balancing is then decided. Real-time collection is needed when gathering node resource data since load balancing is handled dynamically. The causes of an unequal demand on virtual machine resources may be effectively investigated and processed through analysis of the data gathered. Real-time monitoring of each node's resource condition is required while gathering nodes, and this monitoring is what will ultimately allow the scheduling task to be completed.

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.

Proposed algorithm

In this paper, a time efficient load balancing strategy is used which is inspired by honey bee foraging behaviour. 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 ith 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,

$$x_{i,k} = p_{i,k} + \alpha_{i,k} \times (p_{i,k} - p_{j,k}) \qquad (1)$$

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 α⁡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,

$$Pro_i = \frac{fitness_i}{\sum_j fitness_j} \qquad (2)$$

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,

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

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 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. φstand 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.

$$VMV = f(mips, \varphi, dc) \qquad (4)$$

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.

$$RUV = f(t, NIC, RAT, tc) \qquad (5)$$

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.

$$fitness = w1 \times \varphi\,(NIC \div mips) + w2 \times dc \qquad (6)$$

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$P_i = \{p_{i,1}, p_{i,2}, p_{i,3}, \dots, p_{i,n}\}$.

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.

2. Simulation and results

Numerous tests are carried out using the Cloud Analyst simulation toolbox that has been altered to include the suggested approach. Additionally, the Cloud Analyst makes it quick and easy to conduct a number of simulation experiments with modest parameter adjustments repeatedly. The simulation setup is listed in Table 1. The world's six continents are divided into six regions, each of which is modeled after a particular group of users. at peak hours, it is anticipated that 5% of all registered users will be online at once, and that only 10% will be online at off-peak times. 100MB-sized virtual machines are used in the experiment to host programmes. VMs have a 1GB RAM capacity and 10MB of available bandwidth. In simulated hosts, X86 architecture is utilized. Each virtual data centre hosts a certain number of VMs that are devoted to the application. The computers have 4 GB of RAM and 100GB of storage. There are four CPUs on each system, each with a 10000 MIPS capability. 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

The determined overall average Response Time (RT) in ms for the proposed algorithm, 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 HBF	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 Honey Bee Foraging 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 Honey Bee Foraging 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.

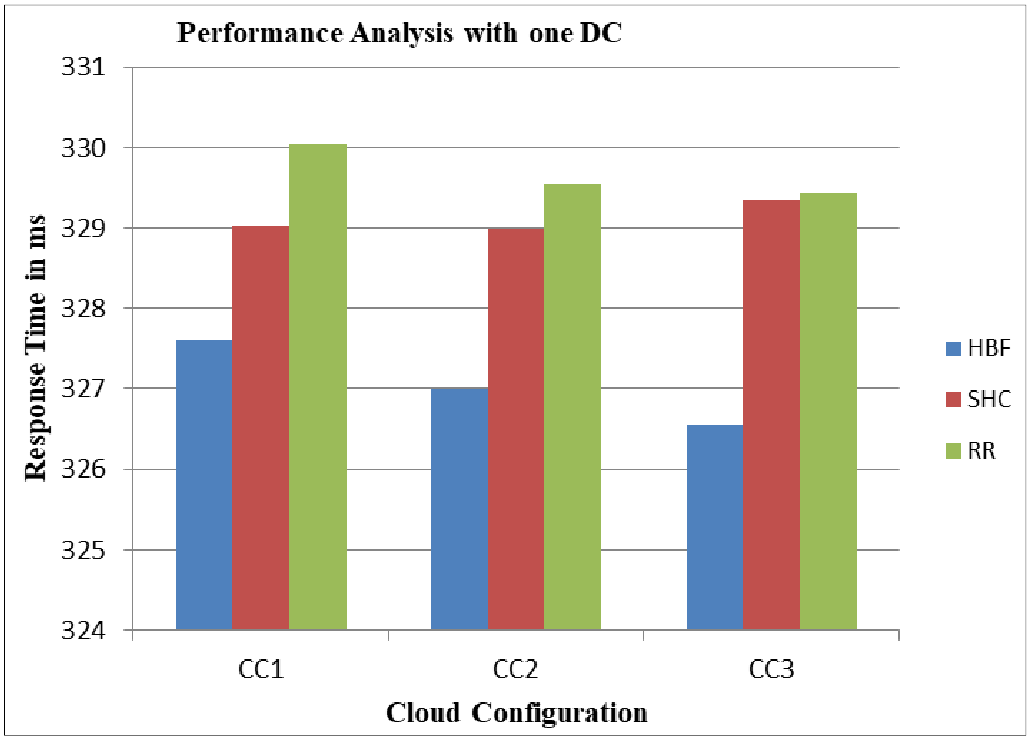

Fig 1: Performance analysis of proposed HBF with SHC and RR Result using One DC

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

Sl. No	CC	DC specification	RT in ms For HBF	RT in ms For 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

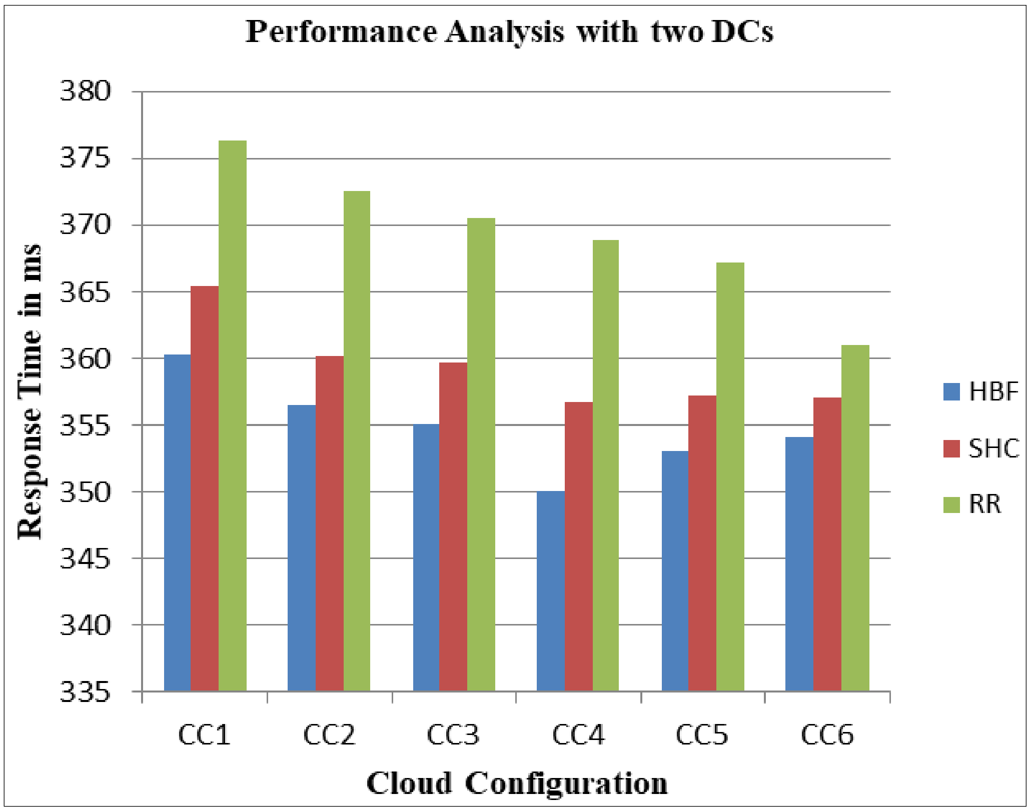

Fig 2: Performance analysis of proposed HBF with SHC and RR Result using Two DCs

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 HBF	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

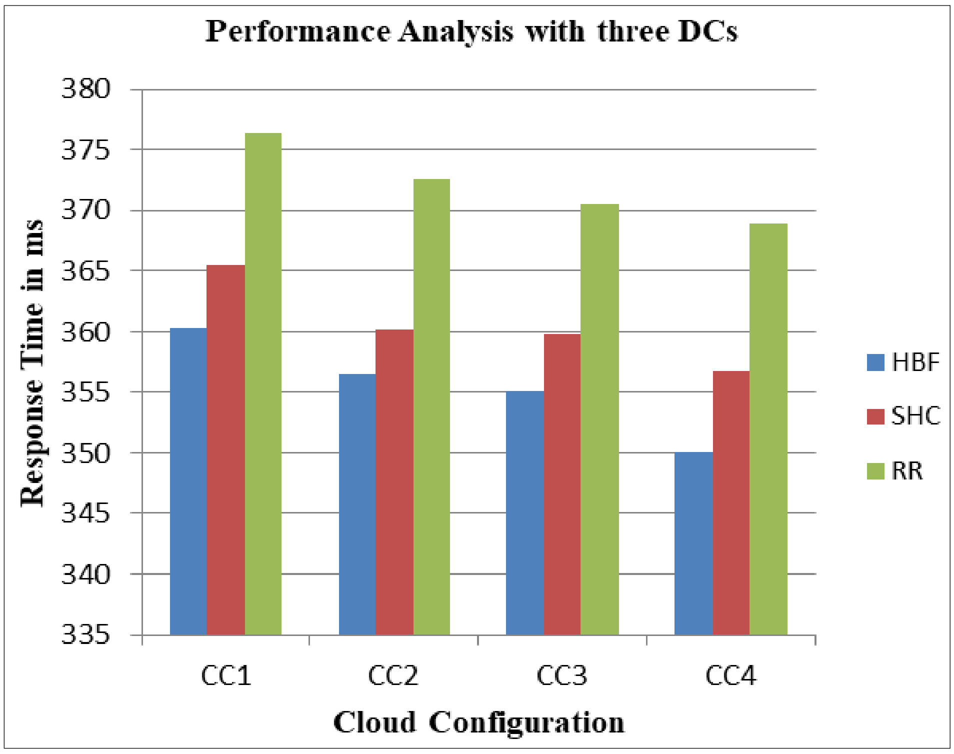

Fig 3: Performance analysis of proposed HBF with SHC and RR Result using Three DCs

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 HBF	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

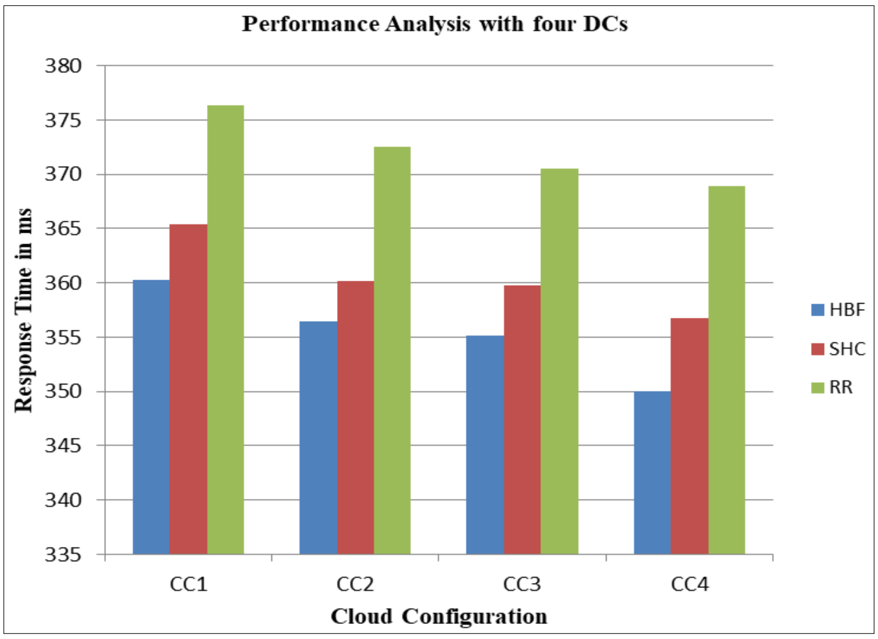

Fig 4: Performance analysis of proposed HBF with SHC and RR Result using Four DCs

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 HBF	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

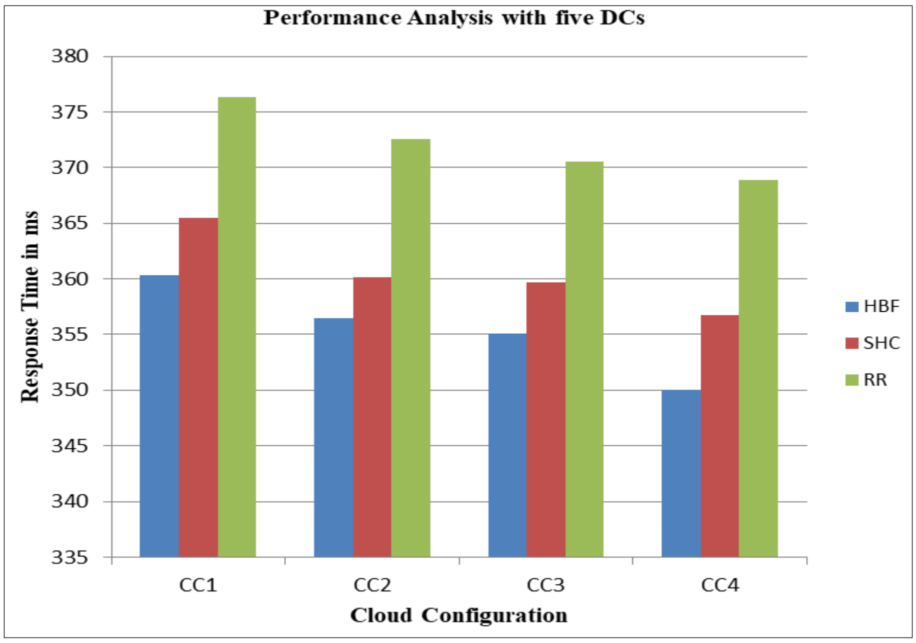

Fig 5: Performance analysis of proposed HBF with SHC and RR Result using Five DCs

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 HBF	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

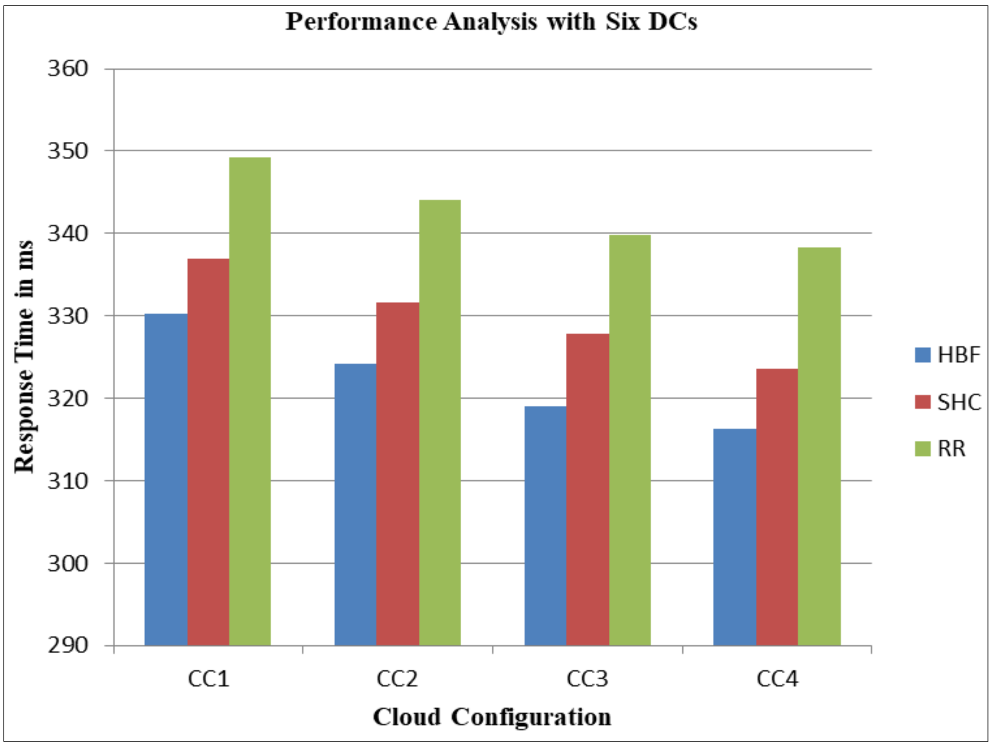

Fig 6: Performance analysis of proposed HBF with SHC and RR Result using Six DCs

Conclusion

A Honey Bee Foraging 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 honey bee foraging 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, Suma V, 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, 2012, 479-485.
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,2010:1-10:13-31.
3. Mohammadian, Vahid, Nima Jafari Navimipour, Mehdi Hosseinzadeh, and Aso Darwesh. "Fault-tolerant load balancing in cloud computing: A systematic literature review." *IEEE Access*,2021:10*:*12714-12731.
4. Uz Zaman SK, Maqsood T, Ali M, Bilal K, Madani SA, Khan AU. A load balanced task scheduling heuristic for large-scale computing systems. Comput. Syst. Sci. Eng. 2019;34:4.
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)*, IEEE, 1998, 79-87.
6. Dasgupta K, Mandal B, Dutta P, Mandal JK, Dam S. A genetic algorithm (ga) based load balancing strategy for cloud computing. Procedia Technology. 2013;10:340-7.
7. Dam, Santanu, GopaMandal, KousikDasgupta, 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*), 2014, 403-413.
8. Jeyakrishnan V, Sengottuvelan P. A hybrid strategy for resource allocation and load balancing in virtualized data centers using BSO algorithms. Wireless Personal Communications. 2017;94:2363-75.
9. Sun H, Chen SP, Jin C, Guo K. Research and simulation of task scheduling algorithm in cloud computing. TELKOMNIKA Indonesian Journal of Electrical Engineering. 2013;11(11):6664-72.
10. Beloglazov A, Buyya R. 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. 2012;24(13):1397-420.
11. Karki S, Goyal A. Performance evaluation of check pointing and threshold algorithm for load balancing in cloud computing. Int J ComputSci Eng. 2018;6(5):2347-693.
12. Mesbahi M, Rahmani AM, Chronopoulos AT. Cloud light weight: A new solution for load balancing in cloud computing. In2014 International Conference on Data Science & Engineering (ICDSE), 2014, 44-50. IEEE.