International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569| A Monthly Peer Reviewed & Referred Journal |

|| Volume 10, Issue 10, April 2021 ||

| DOI:10.15680/IJIRSET.2021.1010117 |

An Improved Fuzzy-based Meta-Heuristic for Optimal Load Balancing in Heterogeneous Cloud

Brototi Mondal

Assistant Professor, Department of Computer Science, Sammilani Mahavidyalaya, Kolkata, West Bengal, India

ABSTRACT: A new paradigm for the delivery and consumption of internet-based services and protocols is being fostered by cloud computing. It offers large-scale computer infrastructure that is defined by consumption, as well as highly flexible infrastructure services that can be scaled up or down in response to customer demand. One of the primary objectives for cloud service providers is to achieve QoS and promptly satisfy end customers' needs for resources. This makes it extremely difficult to choose a suitable node that can fulfill end users' tasks with quality of service. Load balancing, then, refers to the equitable distribution of dynamic workload among several nodes in a distributed context in cloud computing. This study presents a unique hybrid approach integrating the principles of Ant Colony Optimization (ACO) with Fuzzy Logic, to enhance load balancing in cloud environments. Unfortunately, the traditional load balancing techniques are rendered useless by the sheer volume of requests handled and servers that are accessible at any given time. The suggested algorithm takes into account the cloud's response time and load balancing goals. The CloudAnalyst has been used to simulate the suggested load balancing technique. In a typical example application, the experimental result outperformed several existing Ant Colony Based strategies, conventional techniques such as Round Robin (RR), Simulated Annealing (SA), and Genetic Algorithm (GA).

KEYWORDS: Cloud Computing, Load Balancing, Fuzzy logic, Meta-heuristic, Ant Colony Optimization.

I. INTRODUCTION

Cloud computing is the process of distributing storage and computational resources among several data centers, with internet connectivity being the means of accessing cloud-based services. The primary goal of cloud computing is to bring about economies of scale while also enabling low-cost access to large amounts of processing power and storage space (Rajkumar et al., 2013) Different data centers situated in disparate geographic locations are managed and operated by cloud service providers. Although cloud computing has a bright future, several important issues are still not resolved. Load balancing makes sure that every processor or resource in the cloud is performing roughly the same amount of work at all times by distributing the total local load evenly over the cloud. This keeps resources from being underloaded—i.e., operating at a low capacity—while others are fully used. A suitable load balancing algorithm should be environment-adaptive and dynamic in order to achieve the requirements (Alazzam et al., 2019). Cloud load balancing is the procedure used to distribute workloads, storage, processing power, and computing resources. Cloud load balancing is implemented using a variety of methods, such as load-balancing algorithms and scheduling (Ghomi et al., 2017). Essentially, Ant Colony Optimization (ACO) is a problem-solving approach derived from the way ants navigate to find the best routes from their nest to food (Ergu et al., 2013). The purpose of this study is to build a fuzzy module for calculating pheromones and to optimize the ACO algorithm's parameters. The factor that the suggested method seeks to optimize is response time. The efficacy of the suggested method in comparison to earlier algorithms was validated by the various simulations carried out in the CloudAnalyst simulator.

The remainder of this paper is organized as follows: the review of relevant literature on load balancing is condensed in the Section 2. The objective of the study is covered in Section 3. The methodology and the proposed Fuzzy-ACO load balancing method is described in Section 4. Section 5 contains the simulation findings. The last part summarizes the analytical part of the current study, addresses its shortcomings, and suggests future research on this field.

II. LITERATURE REVIEW

Recently, several scheduling techniques have been put out to balance load and improve QoS measures. Ergu et al. (2013) devised a comparison matrix approach that allows for pair wise task comparisons. To expedite job completion, the chosen task was scheduled first. Load balancing was accomplished by Florence et al. (2014) using proactive task scheduling. They improved scheduling by using the firefly method, which maximized resource utilization. The notion of the game theory has been used by the authors in the research study (Bui et al., 2017) to define a virtual machine provisioning policy that ensures an ideal condition of load balancing. An enhanced ACO method was presented in the publication (Yang & Zhuang, 2010) in order to effectively handle the mobile agent routing problem. Some characteristics of the suggested technique include a positive feedback process and resilience. Two methods based on ant colony optimization were suggested by the authors of the paper (Gonzalez et al., 2017), enabling community search functions depending on the information sources that are accessible. The conducted trials have shown that the algorithm with an ACO topology orientation is better suitable for the community research scenario. Dynamic min-min scheduling and dynamic cloud list scheduling are two dynamic scheduling methods that are included in the optimal resource allocation mechanism proposed by Li et al. (2017). The resources' information is updated dynamically, and scheduling is done in advance. The expanded bee algorithm in (Verma et al., 2014) mimics the feeding habits of honey bees by applying a divisible load scheduling theory.

III. LOAD BALANCING IN CLOUD COMPUTING

The idea of load balancing in computing refers to the distribution of workloads among several computing units. By maximizing throughput, minimizing response times, optimizing resource utilization, and preventing overloads, load balancing seeks to improve performance. In cloud, load balancing is mostly accomplished at the scheduling level and uses scheduling and load balancing techniques. In this article a novel cloud load balancing technique using the fundamental foraging behaviour of an ant, known as "ant colony optimization," is prompted by Dorigo and Gambardella (1997) to choose the optimal and shortest route from their nest to their food. Ants leave a chemical trail known as pheromones behind as they travel from their nest to food or the other way around. Ants first select pathways at random. The probability that a single ant will select a certain route from a variety of possibilities, always based on past trails. The likelihood that an ant will be able to distinguish the best possible path from a variety of options depends on the pheromone concentration along each path. Denser pheromone therefore draws in more ants. The process that ultimately aids ants in discovering the best course is essentially a positive feedback system. The adopted Ant Class explains the steps involved in the suggested Fuzzy ACO algorithm, such as the pheromone update and the fuzzy logic based module that assesses the quality of the destination node. Every time there is an iteration, ants begin searching for the ideal state that ensures the optimum load balancing level and the fastest reaction time. Every ant must visit every node that is accessible. The capacity of every node that is available, the desirable qualities of every path, and the network load balancing are all used to determine the best option. A fuzzy logic module does this evaluation by calculating each node's pheromone level based on its available memory, CPU, and bandwidth.

IV. PROPOSED METHODOLOGY

The proposed the Fuzzy-ACO algorithm has two key stages. Initially, it identifies the local solution, optimizing virtual machine allocation. Subsequently, it updates the pheromone matrix, prioritizing available servers and avoiding unavailable host machines. In each iteration, ants update both the pheromone $\rho(a,z)$ and $P_x(a,z)$ probability values. The calculation of the pheromone value $\rho(a,z)$ utilizes fuzzy module, eq.(1) and eq.(2),

$$\varphi(a,z) = \frac{1}{\text{TotalCost}} \quad \ldots\ldots\ldots\ldots\ldots\text{eq.(1)}$$

$$\delta(a,z) = \frac{1}{\text{TotalCloudLets}} \quad \ldots\ldots\ldots\ldots\ldots\text{eq.(2)}$$

The ants then use a probabilistic function to choose the next step to move after each step, one by one.
The computation of the probability P for ant x, positioned at node a, to transition to node , follows eq.(3),

$$P_x(a,z) = \frac{[\rho(a,z)]^{\gamma}\,[\varphi(a,z)]^{\beta}\,[\delta(a,z)]^{\alpha}}{\sum_x[\rho(a,z)]^{\gamma}\,[\varphi(a,z)]^{\beta}[\delta(a,z)]^{\alpha}} \quad \ldots\ldots\ldots\ldots\ldots\text{eq.(3)}$$

where, α, β and γ are weights with values between 0 to 1.

The proposed Algorithm:

Step 1: Randomly initialize a population of ants, each with a unique pheromone value, then arrange them in a random traversal pattern.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569| A Monthly Peer Reviewed & Referred Journal |

|| Volume 10, Issue 10, April 2021 ||

| DOI:10.15680/IJIRSET.2021.1010117 |

Step 2: Execute the following steps either when the highest number of iterations is exceeded or an optimal solution is obtained:
Step 3: On the initial nodes, the ants are positioned.
Step 4: Each individual ant attaches a process for state transition, which aids in the gradual construction of a solution and the concurrent updating of local pheromone levels.
Step 5: Continue to formulate the solutions of all ants.
Step 6: Assess the pheromone levels using the fuzzy logic based module.
Step 7: Continue updating the overall pheromone levels based on the preset rules until the condition for closure is met.
Step 8: Test for the end condition.

V. EXPERIMENTAL RESULTS AND ANALYSIS

Using the simulation tools CloudAnalyst, a scenario of "Internet Banking" for a global bank is used to model the proposed Fuzzy-ACO algorithm. CloudAnalyst does a GUI-based simulation using CloudSim's features. A simulated setup has been created using CloudAnalyst, where the globe is segmented into six "Regions," each aligning with a major continent. Six "User bases" represent user groups corresponding to these continents. A uniform time zone is applied to all user bases, with fluctuating numbers of registered users online during peak hours. During off-peak hours, only a fraction of these users, specifically one twentieth, are assumed to be online. Table 1 provides specifics regarding the user bases utilized in the experiment. Each virtual data center in the simulation contains a specific number of dedicated virtual machines (VMs) for the application. Each machine is standardized with 100 GB of storage, 4 GB of RAM, and 4 CPUs. Additionally, each CPU is equipped with a power capacity of 10,000 MIPS.

Table 1: Simulation environment setup

Serial No.	User Base	Region	Users online during peak hours	Users online during off-peak hours
1	UB#1	North America	5,00,000	85000
2	UB#2	South America	520000	100000
3	UB#3	Europe	400000	64000
4	UB#4	Asia	900000	100000
5	UB#5	Africa	880000	15000
6	UB#6	Oceania	220000	35000

Various experimental scenarios are explored, beginning with one cloud Data Center (DC). In this setup, all user requests worldwide are handled by this lone DC, with allocations of 25, 50, and 75 VMs of Cloud Configuration (CCs) for the application. Details of this simulation environment, along with the measured average Response Time (RT) in milliseconds for Fuzzy-ACO Algorithm (Fuzzy-ACO), traditional Ant Colony Optimization (ACO), soft computing approach Genetic Algorithm (GA), local search technique Simulated Annealing (SA) and Round Robin (RR) scheduling algorithms, are outlined in Table 2. Analysis of the performance is visually represented in Fig.1. Table 3 describes two DCs, having each with varying combinations of 25, 50, and 75 VMs.. The comparison of performance illustrated in Figure 2. Following the initial setups, the experimentation extends to include three and four Data Centers (DCs). Each DC configuration includes combinations of 25, 50, and 75 VMs for each CC, detailed in Tables 4 and 5, respectively. The performance comparisons for these configurations are visually represented in Figures 3 and 4, respectively.

Table.2. Setup of simulation and RT calculated in milliseconds (ms) utilizing a single DC

Serial. Number	**CC**	**DC specification**	**RT using Fuzzy-ACO(ms)**	**RT using ACO(ms)**	**RT using GA(ms)**	**RT using SA(ms)**	**RT using RR(ms)**
1.	CC1	1 DC with 25 VMs	327	327.4	328.75	329.02	330.05
2.	CC2	1 DC with 50 VMs	327.01	327.2	328.42	329	329.55

3	CC3	1 DC with 75 VMs	325.4	326	327.04	329.34	329.44

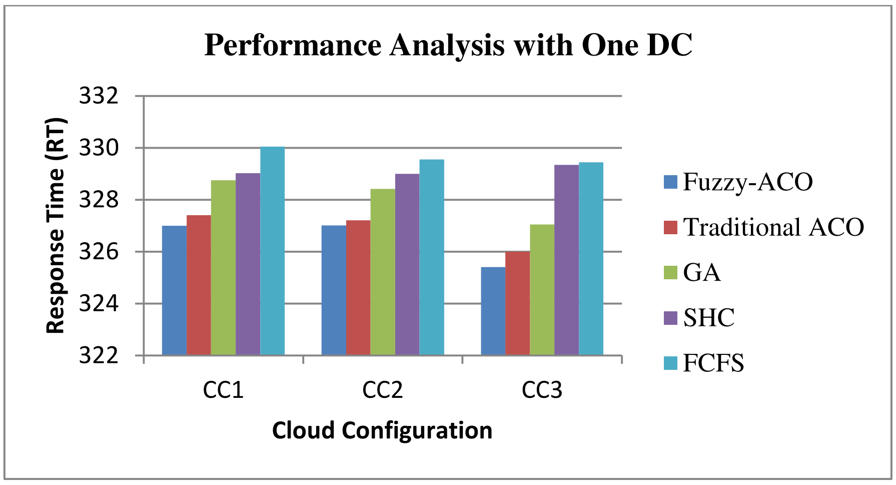

Fig.1. Performance evaluation of the proposed Fuzzy-ACO alongside ACO, GA, SA, and RR results employing one DC

Table.3.Setup of simulation and RT calculated in milliseconds (ms) utilizing two DCs

Serial. Number	CC	DC specification	RT using Fuzzy-ACO(ms)	RT using ACO(ms)	RT using GA(ms)	RT using SA(ms)	RT using RR(ms)
1.	CC1	2 DCs with 25 VMs each	358.7.3	359.5	360.77	365.44	376.34
2.	CC2	2 DCs with 50 VMs each	353.78	354.7	355.72	360.15	372.52
3	CC3	2 DCs with 75 VMs each	353.87	354.76	355.32	359.73	370.56
4	CC4	2 DC swith 25,50 VMs	349.05	348.55	350.58	356.72	368.87
5	CC5	2 DCs with 25,75 VMs	347.85	348.32	351.56	357.23	367.23
6	CC6	2 DCs with 50,75 VMs	349.65	351.65	352.01	357.04	361.01

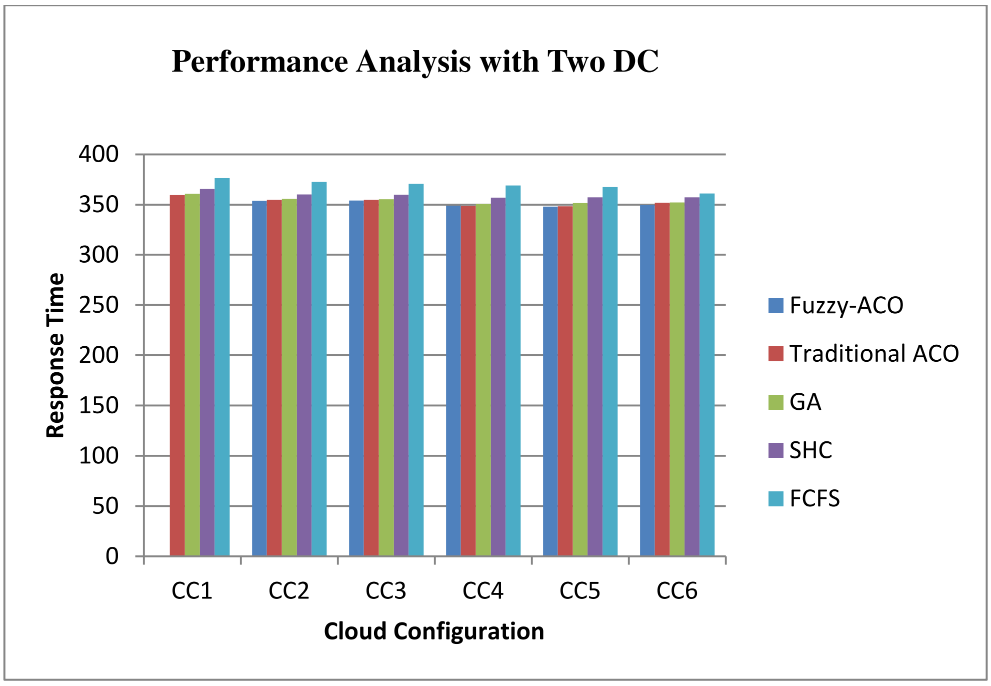

Fig.2. Performance evaluation of the proposed Fuzzy-ACO alongside ACO, GA, SA, and RR results employing two DCs

Table.4. Setup of simulation and RT calculated in milliseconds (ms) utilizing three DCs

Serial Number	CC	DC specification	RT using Fuzzy-ACO(ms)	RT using ACO(ms)	RT using GA(ms)	RT using SA(ms)	RT using RR(ms)
1.	CC1	DC with 25 VMs each	346.23	348.65	350.32	356.82	363.34
2.	CC2	DC with 50 VMs each	347.46	349.42	350.19	355.25	363.52
3	CC3	DC with 75 VMs each	342	343.76	346.01	350.73	361.56
4	CC4	DC with 25,50,75 VMs	342.5	344.34	345.98	350.01	360.87

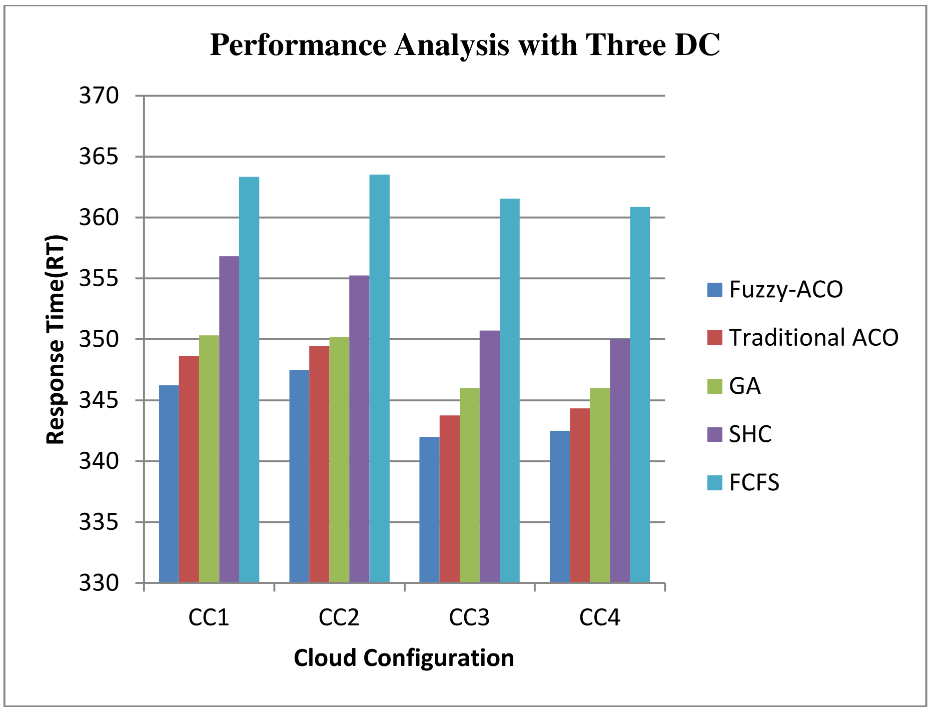

Fig.3. Performance evaluation of the proposed Fuzzy-ACO alongside ACO, GA, SA, and RR results employing three DCs

Table.5. Simulation scenario and the overall average response time (RT) calculated in milliseconds (ms) utilizing four DCs

Sl. No	CC	DC specification	RT using Fuzzy-ACO(ms)	RT using ACO(ms)	RT using GA(ms)	RT using SA(ms)	RT using RR(ms)
1.	CC1	DC with 25 VMs each	345.3	346.7	348.85	354.35	360.95
2.	CC2	DC with 50 VMs each	340.14	342.3	345.54	350.71	359.97
3	CC3	DC with 75 VMs each	334.7	337.4	340.65	346.46	358.44
4	CC4	DC with 25,50,75VMs	330.12	334.34	337.88	344.31	355.94

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569| A Monthly Peer Reviewed & Referred Journal |

|| Volume 10, Issue 10, April 2021 ||

| DOI:10.15680/IJIRSET.2021.1010117 |

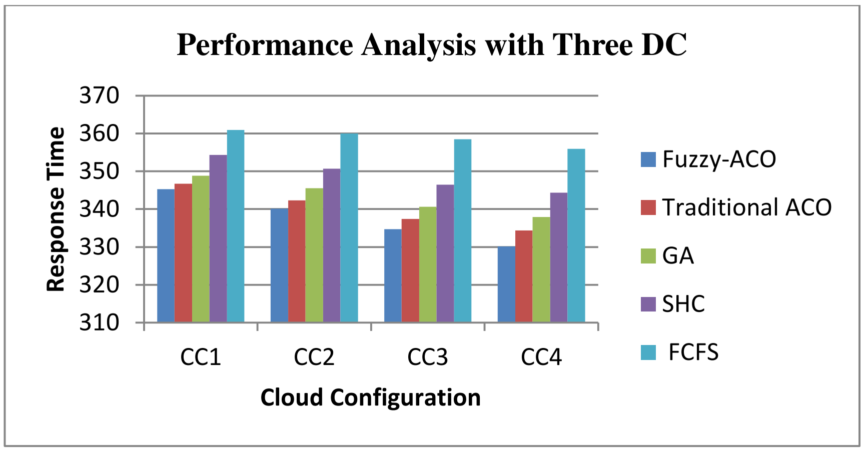

Fig.4. Performance evaluation of the proposed Fuzzy-ACO alongside ACO, GA, SA, and RR results employing four DCs

VI. CONCLUSION

Load balancing poses a significant challenge due to its NP-hard nature, rendering finding an optimal solution either infeasible or impractical. This paper introduces a soft computing approach leveraging Ant Colony Optimization and Fuzzy Logic to initiate load balancing within a cloud computing framework. Through a comprehensive analysis, the recommended load balancing strategy not only surpasses several other strategies but also ensures adherence to the Quality of Service (QoS) requirements of customer jobs. Experimentation findings indicate that the proposed load balancing technique outperforms existing optimization algorithms. Notably, the absence of fault tolerance considerations in this study and the assumption of uniform job priorities may not reflect real-world scenarios. However, it is feasible to incorporate fault tolerance considerations and account for diverse function variations while calculating pheromone values.

REFERENCES

1. Rajkumar, B., V. Christian, and S. S. Thamarai. "Mastering Cloud Computing." Editorial: Morgan Kaufmann (2013).
2. Alazzam, Hadeel, Esraa Alhenawi, and Rizik Al-Sayyed. "A hybrid job scheduling algorithm based on Tabu and Harmony search algorithms." The Journal of Supercomputing 75, no. 12 (2019): 7994-8011.
3. Ghomi, Einollah Jafarnejad, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. "Load-balancing algorithms in cloud computing: A survey." Journal of Network and Computer Applications 88 (2017): 50-71.
4. Ergu, Daji, Gang Kou, Yi Peng, Yong Shi, and Yu Shi. "The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment." The Journal of Supercomputing 64 (2013): 835-848.
5. Florence, A. Paulin, and V. Shanthi. "A load balancing model using firefly algorithm in cloud computing." Journal of Computer Science 10, no. 7 (2014): 1156.
6. Bui, Khiet Thanh, Tran Vu Pham, and Hung Cong Tran. "A load balancing game approach for VM provision cloud computing based on ant colony optimization." In Context-Aware Systems and Applications: 5th International Conference, ICCASA 2016, Thu Dau Mot, Vietnam, November 24-25, 2016, Proceedings 5, pp. 52-63. Springer International Publishing, 2017.
7. Yang, Jingan, and Yanbin Zhuang. "An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem." Applied soft computing 10, no. 2 (2010): 653-660.
8. Gonzalez-Pardo, Antonio, Jason J. Jung, and David Camacho. "ACO-based clustering for Ego Network analysis." Future Generation Computer Systems 66 (2017): 160-170.
9. Li, Jiayin, Meikang Qiu, Zhong Ming, Gang Quan, Xiao Qin, and Zonghua Gu. "Online optimization for scheduling preemptable tasks on IaaS cloud systems." Journal of parallel and Distributed Computing 72, no. 5 (2012): 666-677.
10. Verma, Pushpendra, Jayant Shekhar, and Amit Asthana. "A model for evaluating and maintaining load balancing in cloud computing." IJCSMC 3, no. 3 (2014): 501-509.
11. Dorigo, Marco, and Luca Maria Gambardella. "Ant colony system: a cooperative learning approach to the traveling salesman problem." IEEE Transactions on evolutionary computation 1, no. 1 (1997): 53-66.