Optimization of Datacenter Selection Policy in Cloud Computing using Differential Evolution Algorithm

Shusmoy Chowdhury
Department of Computer Science
Missouri State University
Springfield, USA
sc26s@missouristate.edu

Ajay Katangur
Department of Computer Science
Missouri State University
Springfield, USA
ajaykatangur@missouristate.edu

2023 IEEE Cloud Summit | 979-8-3503-2217-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/CloudSummit57601.2023.00012

***Abstract*—Cloud computing has become one of the most influential inventions in modern-day computer science. It has become one of the integral parts of our day-to-day life. The success of our regular activities relies on the efficiency and effectiveness of the cloud services. The datacenter selection policy is one of the vital elements in cloud computing. The user requests are processed in the datacenter. Therefore, an optimal datacenter ensures user satisfaction with cloud services. In this research, we propose a differential evolution-based datacenter selection policy to choose the optimal datacenter for the userbase. Differential evolution mainly depends on the evolutionary process of mutation, which helps the algorithm explore the whole search space and find the optimal datacenter. We have used the CloudAnalyst simulator to experiment with the correctness and performance of our algorithm. The simulation results demonstrate that the proposed differential evolution algorithm performs better than the existing datacenter selection algorithm, such as the closest datacenter selection policy and optimized response time policy.**

***Index Terms*—Cloud computing, Differential Evolution Algorithm, Service Broker Policy**

I. Introduction

Researchers are working relentlessly to make our life easier and more comfortable. In recent years cloud computing has become one of the leading solutions for resources, processing, and distribution [1]. Cloud computing provides services for IT and non-IT companies. Nowadays, large companies, as well as small start-ups, are shifting their application from local physical servers to cloud services.

Cloud computing facilitates on-network services for shared and configurable computing as servers (IaaS), operating systems (PaaS), applications, and services (PaaS). These resources can be made available and released with less human involvement and administrative efforts. These advantages of cloud services make it easier for users to deploy and maintain their applications in it. Cloud computing has become essential to our next-generation computing infrastructure due to its on-demand, low-cost services.

In recent years the use of cloud services has been increasing rapidly. Due to this increased usage, many companies are providing cloud services. Some popular cloud services are Amazon Web Services (AWS), Microsoft Azure, Google, IBM Cloud, Rackspace, Red Hat, Verizon cloud, and VMware [2]. These cloud providers need to keep the users hooked on their cloud services. For this reason, user satisfaction plays a vital role in engaging users in cloud services.

To maintain user satisfaction, cloud providers need to maintain the effectiveness and efficiency of the cloud services. The efficiency of the cloud services depends on how fast the user request can be processed and the response sent back.

The success of cloud services relies on two crucial elements of cloud computing. The first one is the service broker policy. Service broker policy means selecting the optimal datacenter for a userbase based on distinct factors such as the number of requests, latency, bandwidth, datacenter capacity, etc. Optimal datacenter selection plays a significant role in cloud computing by processing user requests efficiently and effectively to maximize user satisfaction.

Another critical factor in cloud computing is load balancing. Load balancing helps the datacenters (DCs) to use the resources appropriately and process the user tasks efficiently. DCs are a collection of physical servers with abstraction over the physical servers called virtual machines (VMs). Load balancing distributes the tasks among the VMs so that no VMs are overloaded or underloaded with tasks. Load balancing ensures the efficient processing of user requests in the DCs. Therefore the success of load balancing also relies on the appropriate DC selection based on the user's requirements.

Numerous factors, such as the distance between the userbase and the DC, the data transfer cost of the DC, the response time of the DC, the processing time of the DC, bandwidth, fault tolerance of the DC, energy consumption, play a crucial role in the DC selection policy. But from the user perspective, the cost of the DC, response time, and processing time are vital factors for the DC selection. Although the cost of the DC is a crucial factor, the response time and processing time ensure user satisfaction with cloud services. The cost of the DCs includes the cost of the physical servers and their maintenance charges. Response time defines the time needed

This material is based upon work supported by the National Science Foundation under Grant OAC-1828380

979-8-3503-2217-0/23/$31.00 ©2023 IEEE
DOI 10.1109/CloudSummit57601.2023.00012

by DCs to send feedback to the user for a particular request, and Processing time means the amount of time required to process the user requests in the DCs. Most cloud services offered by several companies charge almost similarly based on the user's requirements. If the DC needs a considerable time to process the requests and send the response to the users, it annoys the users and decreases user satisfaction with that cloud service. As a result, the response time and processing time must be prioritized over the cost of the DC while designing an efficient DC selection policy.

Our model aims to minimize response time and processing time for user requests. We select the optimal DC using the proposed differential evolution-based DC selection policy in cloud computing. The optimal DC will utilize the storage and resources appropriately to increase the efficiency of the cloud.

Section II summarizes previous accomplishments in the DC selection policy. Section III provides the problem of applying differential evolution (DE) in the cloud research domain. Section IV outlines our proposed DE-based service broker policy and the motivation behind using DE. Section V details the cloud workflow and the integration of the service broker policy into the cloud environment. Section VI describes the experimental setup in CloudAnalyst, which is used to test the performance of the proposed DE-based service broker policy. Section VII compares the DE-based algorithm's results to the Closest Datacenter (CDC) and Optimized Response Time (ORT) policies. Finally, Section VIII concludes this paper with a summary and future extension of this research.

II. Related Work

DC selection policy tries to find the optimal DC that can handle the request of the users effectively and efficiently. The DC selection policy is crucial in achieving user satisfaction and improving cloud performance. Many researchers have done several works on the DC selection policy in cloud computing.

Three well-known service broker policies are available in the CloudAnalyst platform [3]. The first one is the closest DC policy which chooses the nearest DC to process the request. It is also known as a proximity-based service broker policy. The second is the Optimized response time service broker policy which uses the DC with a better response time to process the request. Third, dynamically reconfigurable routing with load balancing extends the proximity-based service broker policy. This policy considers current processing times, the best processing time ever achieved, and the distance between the user base and DC.

Many researchers have worked to design efficient and effective DC selection policies. Some of them prioritized cost minimization [4]–[7]. Other researchers focused on the response time [8]–[12] and execution time [13], [14] to increase the efficiency and effectiveness of the DC selection policy. If we focus on one of the factors, other factors may be affected negatively. When the DC selection policy tries to minimize response time, it might increase the cost and execution time [8]–[12]. Sometimes, this will cause improper utilization of resources and overcrowding the selected DC with many tasks [15]–[17]. Again, besides the time and cost, several other factors, like bandwidth, latency, storage, etc., must be considered while designing the DC selection policies.

Researchers tried to extend the existing DC selection policies to increase their performance. Sarfaraz et al. worked with the proximity-based DC selection policy in the CloudAnalyst and enhanced its performance by not overcrowding the single DC in a region with tasks [10]. The enhanced proximity-based DC selection policy shifts the task to another closest DC in the region if the current DC is overloaded with tasks. Dhaval et al. [18] considered the cost-effective DC among the closest DCs in proximity-based DC selection policy. Devyaniba et al. also worked on a similar cost-effective DC selection policy with a proximity-based service broker policy[16]. Deepak Kapgate et al. used the round-robin algorithm to design the DC selection policy. The algorithm tries to reduce the response time on the client side by incorporating the round-robin mechanism in the DC selection policy. Radi et al. [8] modified the round-robin DC selection list policy with a weighted DC list. Zakaria et al. [19] used the efficiency/cost ratio to select the appropriate DC. The closest DC will be chosen if the ratio is not minimum. D. Arya et al. [15] designed a DC selection policy that selects the optimal DC based on the user's request priority and the current load of the DC selections. Manasrah et al. [16] designed a heuristic-based DC selection policy where the algorithm considers the communication channel bandwidth, latency, and the size of the job to minimize the response time and reduce the loads in the DC by shifting extra load to the next DC with better response time and execution time. To improve resource management, Minhaj et al. [20] proposed a normalization-based hybrid service brokering approach combined with throttled round-robin load balancing. The algorithm uses the cost- and performance-aware provision of cloud services.

Cloud services will be changing their configurations based on user requirements. As a result, the DC selection policy should change dynamically when the conditions change. Evolutionary algorithms progress with time and look for optimal solutions from the solution search space. Researchers worked to incorporate the strategies of evolutionary algorithms with cloud computing concepts to design effective and efficient cloud services. Kessaci et al. [21] used the multi-objective genetic algorithm for the DC selection policy. The algorithm will choose the optimal solution from the Pareto of solutions based on minimized response time and overall cost.

After surveying different cloud services, we have seen that cloud services provided by Amazon, Microsoft, IBM, Google, etc., have almost similar costs for various cloud services and applications [22]. Considering similar prices, it is essential to focus more on the response and processing time rather than the overall cost of cloud services. The user always expects timely responses from the cloud services, and if the services fail to accomplish that, it will decrease the user satisfaction level for that particular cloud service. For this reason, we have focused on minimizing the cloud services' response and processing times. We have proposed a differential evolution-based DC selection policy that will choose the optimal DC based on two

criteria: minimum response time and data processing time.

III. Problem Encoding for Genetic Algorithm

A. Chromosome Structure

Chromosomes or individuals are fundamental concepts of any evolutionary algorithm. The chromosome represents the solution structure of the algorithm. The chromosome pattern in this research defines a userbase allocated to a particular DC. Table I illustrates an individual chromosome used in this research. The size of the chromosome will be the size of the userbase. Therefore, the chromosome index represents the userbase number, and the gene in the chromosome represents the DC allocated for the userbase.

TABLE I
CHROMOSOME REPRESENTATION OF THE GENETIC ALGORITHM

UB1	UB2	UB3	UB4	UB5	UB6	UB7	UB8	UB9
DC2	DC0	DC1	DC5	DC4	DC3	DC2	DC1	DC3

B. Population

Population defines the individuals or chromosomes from which the solutions can be found. Generating the initial population plays a vital role in the progress of the DE. We need to ensure proper diversity in the population. For this reason, the initial population is generated randomly. The randomness in the population confirms the exploration of different search spaces in cloud computing. We will also inherit the result of the closest DC policy in the initial solution. We assume that if the closest DCs are the fittest in the evolution process, they will survive toward the optimal solution search; otherwise, they will die during the evolution process.

C. Fitness Function

User satisfaction depends on the cloud environment's response and data processing times. So, the response time and data processing time of the DCs define the efficiency of the service broker policy. In a cloud environment, different DCs have various numbers of VMs. Also, the transfer cost of the DC varies with distinct factors like region, number of physical machines, etc. We will consider these factors in our evolutionary algorithms and find an optimal DC for userbases. We aim to optimize the DCs response time and data processing time. The optimization or fitness function is shown in the equation 1.

$$F(R_t, D_t) = (\sum R_t + \sum D_t)/N \tag{1}$$

Here,
R_t = Response time of each DC
D_t = Data Processing time of each DC
N = Number of DCs

Our fitness function is dynamic as the response time and data processing time will change with respect to time. So, our study is divided into multiple episodes, and we will find the optimal DCs for each episode.

IV. Proposed Differential Evolution Algorithm

Differential Evolution (DE) is one of the most powerful stochastic optimization algorithms for real-parameter optimization. The algorithm uses the same computational steps as evolutionary algorithms (EAs) [23]. By randomly selecting and comparing small numbers of current-generation population members, DE-variants perturb the present-generation population members. In this case, no separate probability distribution is required for generating offspring. Globally, DE has drawn the attention of several researchers. Consequently, many variants of the basic algorithm have been developed that are more efficient. "Fig 1" shows the flowchart of the DE algorithm.

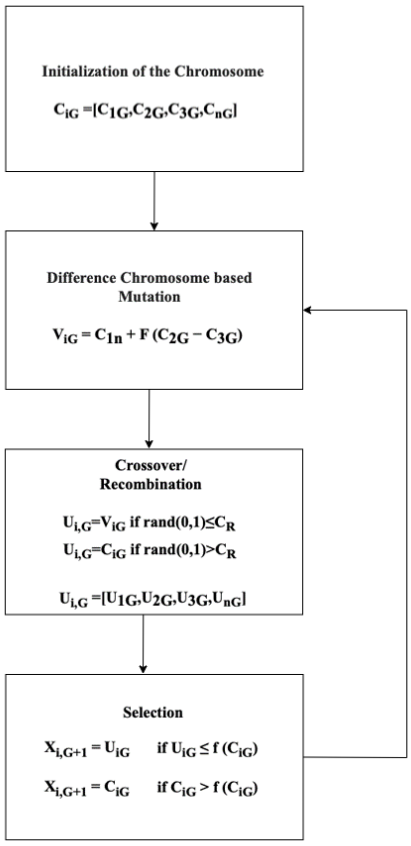

Fig. 1. Flowchart of Differential Evolution Algorithm

A. Initialization

In the first step, we randomly generate a population of real-valued NPD parameter vectors. Vectors, also known as chromosomes (C), can be used to solve the multidimensional optimization problem. G = 0,1..., Gmax denotes successive generations in DE [24]. In equation 2 we can see the representation of the ith parameter vector in the current generation since the parameter vectors may change over the generations.

$$C_{iG} = [C_{1G}, C_{2G}, C_{3G}, C_{nG}] \tag{2}$$

B. Mutation

A mutation is a sudden change in the gene characteristics of a chromosome. In evolutionary computing, however, the mutation can also be viewed as an unexpected change, or perturbation [25]. A parent vector from the current generation is called a target vector in DE literature. In genetic engineering, a donor vector as shown in equation 3 obtains mutants using differential mutations. The offspring formed by recombining the donor and target vector is a trial vector. When DE-mutation is performed by creating a donor vector for each mutation, it is the simplest form. We need to choose three different random base vectors from the population. The difference between any two of these three vectors is scaled by a scalar number F.

$$\text{Donor Vector } V_{iG} = C_{1n} + F\left(C_{2G} - C_{3G}\right) \tag{3}$$

Several mutation strategies can be applied in the DE algorithm to enhance the algorithm's performance [26]. The selection criteria of the base vector in mutation accelerates the performance of the DE algorithm. In our proposed DE algorithm, we inherit the best individual in the population during the mutation step. In equation 3, C_{1n} will be the best individual in the population, and C_{2G} and C_{3G} will be any other random vectors in the solution. "Fig 2" shows how the mutation is performed in our proposed DE algorithm.

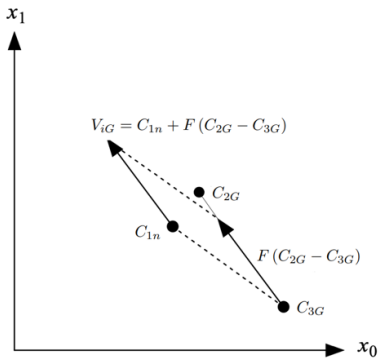

Fig. 2. Mutation in Differential Evolution Algorithm

C. Crossover

The crossover operation enhances the diversity of the population after the donor vector has been generated through mutation. [25] With the target vector, components are exchanged between the donor vector and the target vector C_i.

$$U_{i,G} = V_{iG} \quad \text{if } rand(0,1) \leq (C_R) \tag{4}$$

$$U_{i,G} = C_{iG} \quad \text{if } rand(0,1) > (C_R) \tag{5}$$

$$U_{iG} = [U_{1G}, U_{2G}, U_{3G}, U_{nG}] \tag{6}$$

The trial vectors are denoted as shown in equation 6. The crossover rate(c_R) shown in equation 4 and equation 5 decides the crossover procedure of our proposed algorithm. We generate a random number from 0 to 1. If the number is greater than the crossover rate original vector will be chosen for the crossover; otherwise, the donor vector will move forward to perform the crossover.

We have two types of crossover methods in DE, exponential (or two-point modulo) and binomial (or uniform).

In this research, we used an exponential crossover, where an integer n is randomly chosen among the numbers [1, D] [27]. This integer n acts as a starting point in the target vector, where the crossover or exchange of components with the donor vector starts. We also choose another integer, L, from the interval [1, D]. L denotes the number of components the donor vector contributes to the target vector. After choosing n and L, we perform the crossover from starting point n to ending point $n+L$ of the target vector and donor vector. We can see the results after crossover in equation 7. "Fig 3" exhibits the crossover mechanism in the proposed DE algorithm.

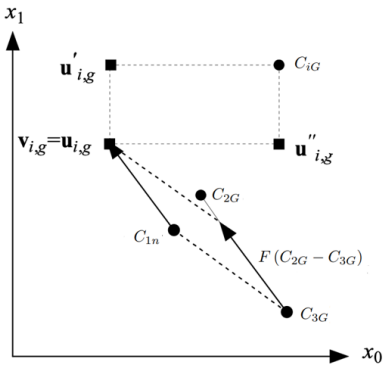

Fig. 3. Crossover in Differential Evolution Algorithm

$$\text{Trial Vector } U_{iG+1} = [U_{1G+1}, U_{2G+2}, U_{3G+3}, U_{nG+n}] \tag{7}$$

D. Selection

A "greedy" selection scheme is used in our proposed algorithm, if the trial vector yields a better cost function value than the parameter vector. In contrast, the selection determines whether the target or the trial vector survives to the next generation, i.e., at $G = G + 1$.

$$X_{i,G+1} = U_{iG} \quad \text{if } U_{iG} \leq f\left(C_{iG}\right) \tag{8}$$

$$X_{i,G+1} = C_{iG} \quad \text{if } C_{iG} > f\left(C_{iG}\right) \tag{9}$$

The selection procedure functions for our proposed algorithm are given by equation 8 and equation 9. Identical or lower values of the new trial vector replace the corresponding target vector in the next generation; otherwise, it remains in the population [25]. Minimizing the objective function either improves (in terms of fitness) or maintains the population's current status. Even when both target and trial vectors produce the same objective function value, DE-vectors replace the target vector with the trial vector so that generations can travel across flat fitness landscapes.

E. Motivation for using Differential Evolution Algorithm

Cloud services work continuously with different request sizes and a different number of requests. So, DC selection needs to be efficient and effective as time progresses. Differential evolution executes with the natural selection policies to advance with time and find the optimum solution. Many stochastic algorithms can be stuck in the local optimum solution while finding the optimal solution. We can see in "Fig 4" that many algorithms will find the optimal solutions in $L1$ and $L2$, but the actual optimal solution will be in G. The crucial step in the differential evolution algorithm is mutation. Mutation helps the algorithm to shift from one search space to another. As a result, differential evolution has more chances to explore the whole search space than any other algorithm. On the other hand, the fitness function helps the differential evolution algorithms to compare the fitness among different solutions from different search spaces. So, the differential evolution algorithm is more likely to find the optimal DCs in the cloud.

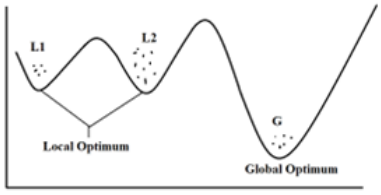

Fig. 4. Local optimum and Global optimum

V. PROPOSED CLOUD ENVIRONMENT

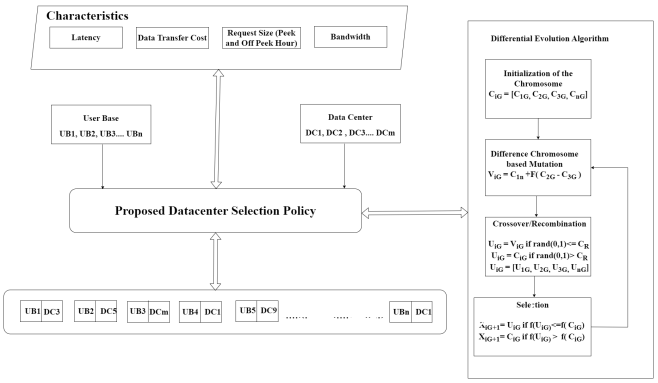

Fig. 5. Workflow of Cloud Environment

A cloud environment is a complex system with attributes such as userbase, DC, load balancing policy, and service broker policy. "Fig 5" shows the workflow of the proposed cloud environment. The cloud components have several characteristics: data transfer cost, request size in peak and off-peak hours, latency, bandwidth, request number, etc. The service broker policy is responsible for choosing the DC for the userbase based on these attributes of the cloud. In the beginning, the user will configure the environment according to their requirements, such as:

- the number of DCs needed for their requests
- the request size
- timing of the peak and off-peak hours
- the number of requests at these hours
- the DC configuration
- the load balancing policy

The user request and DC are configured in the service broker policy. Based on these inputs, the service broker policy will decide the selection of DCs for a particular userbase for a specific time. We have used evolutionary-based algorithms, specifically DE, to choose the optimal DCs

VI. ENVIRONMENT SETUP

Many cloud computing providers exist, such as Amazon Web Services (AWS), Microsoft Azure, Google, IBM Cloud, Rackspace, Red Hat, Verizon cloud, and VMware [28]. They all provide users access to various configurable computing resources (servers, storage, networks, applications). These providers will only let us integrate our proposed solution once we prove the correctness of our algorithms in the cloud environment. Therefore, we need a simulation environment to simulate real-world cloud scenarios and demonstrate the algorithm's efficiency.

CloudSim [29] is one of the first cloud simulation modeling platforms. CloudSim allows VMs to be managed by hosts managed by DCs. The platform does not have any user interface for the users, so it is hard to configure the system according to real-world scenarios. Moreover, the output results are presented in the console in text-based form, making it hard to analyze compared to graphical results.

Another simulation environment CloudAnalyst [30] overcomes these challenges by providing a graphical user interface for the users to configure the cloud environment according to their requirements. "Fig 6" shows the world view of the CloudAnalyst simulator. The world in the CloudAnalyst platform is divided into six regions defined as R0, R1, R2, R3, R4, and R5. The blue dots in the map represents the userbase, and the red dots represent the DCs of the cloud services around the globe. The four main factors of the CloudAnalyst are userbase, DC, Load balancing policy, and Service broker policy. We integrated the proposed DE-based algorithm into the service broker policy.

A. Userbase Configuration

Userbase defines a group of users from a particular region. The user base has several properties. The user request per hour can be identical or different for each user base. Peak hours are a vital factor for the user base. Peak hours define the time when most users are active and when the cloud is processing an enormous number of requests. The average number of users during peak hours is exceptionally high compared to the

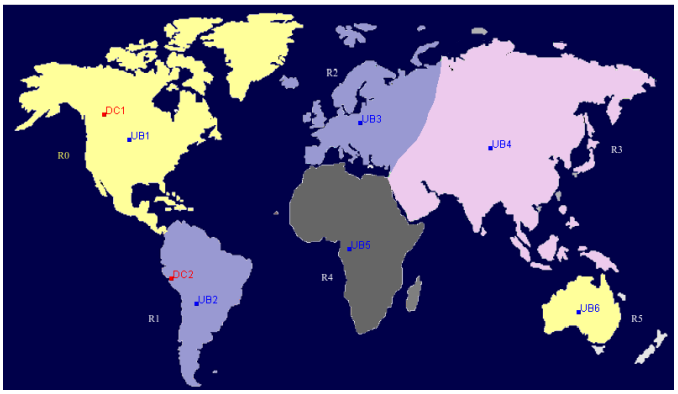

Fig. 6. CloudAnalyst Simulator

average number during off-peak hours. Tables II, III, IV shows different user base configurations used in our experiment.

TABLE II
USERBASE CONFIGURATION 1 (UBC1)

Name	Region	Request Per User per Hra	Data Size per Request (Bytes)	Peak Hours Start (GMT)	Peak Hours End(GMT)	Avg Peak Users	Avg Off-Peak Users
UB1,UB11,UB14, UB16,UB18,UB40	0	12,60,60, 60,60,60	100	13,3,3, 3,3,3	15,9,9, 9,9,9	400000,10000,10000, 10000,10000,1000	40000,1000,1000, 1000,100,100
UB2,UB6,UB20, UB35,UB39	1	12,12,60, 60,60	100	15,3,3, 3,3	17,9,9, 9,9	100000,80000,1000, 1000,1000	10000,1000,1000, 100,100
UB3,UB7,UB8, UB9,UB10,UB13, UB15,UB17,UB19, UB23,UB24	2	12,60,60, 60,60,60, 60,60,60, 60,60	100	20,3,3, 3,3,3, 3,1,3, 3,3	22,9,9, 9,9,9, 9,2,10, 9,9	300000,10000,10000, 10000,10000,10000, 10000,1000,10000, 1000,10000	30000,1000,1000, 1000,1000,1000, 1000,1000,1000, 1000,1000
UB4,UB12,UB26, UB30,UB32,UB38	3	12,60,60, 60,60,60	100	1,3,3, 3,3,3	3,9,9 9,9,9	150000,10000,10000, 10000,10000,1000	15000,1000,100, 1000,1000,100
UB5,UB21,UB25, UB31,UB33,UB34	4	12,60,60, 60,60,60	100	21,3,1, 15,7,1	23,9,7, 18,8,4	50000,10000,10000, 10000,10000,10000	5000,1000,1000, 1000,1000,1000
UB22,UB27,UB28, UB29,UB36,UB37,	5	60	100	3,15,3, 9,3,3	9,19,9, 12,9,9	10000,10000,10000, 10000,1000,1000	1000,100,100, 100,100,100

TABLE III
USERBASE CONFIGURATION 2 (UBC2)

Name	Region	Request Per User per Hra	Data Size per Request (Bytes)	Peak Hours Start (GMT)	Peak Hours End(GMT)	Avg Peak Users	Avg Off-Peak Users
UB1,UB11,UB14, UB40,UB16,UB18, UB46	0	12,60,60, 60,60,60, 60	100	13,3,3, 13,3,4, 6	15,9,9, 21,9,9, 24	400000, 10000 ,10000 , 10000 ,10000 ,10000 , 10000	40000 , 1000 ,1000 , 1000 ,1000 ,1010 , 1000
UB2,UB6,UB35, UB39,UB20,UB41	1	12,12,60, 60,60,60	100	15,9,3, 7,3,10	17,11,23, 15,9,15	100000, 80000, 10000, 10000, 10000, 10000	10000, 8000, 1000, 1000, 1000, 1000
UB23,UB24,UB3, UB7,UB8,UB9, UB10,UB13,UB15, UB17,UB19,UB42	2	60,60,12, 60,60,60, 60,60,60, 60,60,60	100	3,3,20, 3,7,3, 1,3,3, 1,3,14	9,9,22, 9,9,6, 9,9,9, 2,10,17	10000, 10000, 300000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000	1000, 1000, 30000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000
UB4,UB26,UB30, UB32,UB12,UB38, UB43	3	12,60,60, 60,60,60, 60	100	1,3,3, 3,5,1, 5	3,9,9, 9,9,5, 12	150000, 10000, 10000, 10000, 10000, 10000, 10000, 10000	15000, 100, 1000, 1000, 1000, 1000, 1000
UB21,UB25,UB5, UB31,UB33,UB34, UB44	4	60,60,12, 60,60,60, 60	100	5,1,21, 15,7,1, 7	9,7,23, 18,8,4, 19	10000, 10000, 50000, 10000, 10000, 10000, 10000	1000, 1000, 5000, 1000, 1000, 1000, 1000
UB22,UB27,UB28, UB29,UB36,UB37, UB45	5	60	100	23,15,3, 9,5,8, 3	24,19,9, 12,9,19, 9	10000	1000, 100, 100, 100, 1000, 1000, 1000

B. DC configuration

The DC tries to process the userbase request effectively and efficiently. DC is a collection of physical servers, where each

TABLE IV
USERBASE CONFIGURATION 3 (UBC3)

Name	Region	Request Per User per Hra	Data Size per Request (Bytes)	Peak Hours Start (GMT)	Peak Hours End(GMT)	Avg Peak Users	Avg Off-Peak Users
UB1,UB11,UB14, UB16,UB18,UB40	0	12,60,60, 60,60,60	100	13,3,3, 3,4,3	15,9,9, 9,9,9	400000, 10000 ,10000 , 10000 ,10000 ,10000	40000 , 1000 ,1000 , 1000 ,1010 , 1000
UB2,UB6,UB35, UB39,UB20	1	12,12,60, 60,60	100	15,9,3, 3,3	17,11,9, 9,9	100000, 80000, 10000, 10000, 10000	10000, 8000, 1000, 1000, 1000
UB3,UB23,UB24, UB7,UB8,UB9, UB10,UB13,UB15, UB17,UB19	2	12,60,60, 60,60,60, 60,60,60, 60,60	100	20,3,3, 3,7,3, 1,3,3, 1,3	22,9,9, 9,9,6, 9,9,9, 2,10	300000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000	30000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000
UB4,UB26,UB30, UB32,UB12,UB38	3	12,60,60, 60,60,60	100	1,3,3, 5,3,3	3,9,9, 9,9,9	150000, 10000, 10000, 10000, 10000, 10000	15000, 100, 1000, 1000, 1000, 1000
UB5,UB21,UB25, UB31,UB33,UB34	4	12,60,60, 60,60,60	100	21,5,1, 15,7,1	23,9,7, 18,8,4	50000, 10000, 10000, 10000, 10000, 10000	5000, 1000, 1000, 1000, 1000, 1000
UB22,UB27,UB28, UB29,UB36,UB37	5	60	100	23,15,3, 9,3,3	24,19,9, 12,9,9	10000	1000, 100, 100, 100, 1000, 1000

physical server works like a computer's CPU in the cloud. Cloud computing uses an abstraction over the physical servers named VM. The VMs can have the same configuration as physical servers or be different from the physical servers. The combined configuration of the VMs must be equal to that of the available physical servers. Creating more VMs than the available resources or physical servers is impossible. The tasks are divided among the VMs using load balancing algorithms. The more significant number of VMs enhances the performance of the cloud environment.

The DC is configured as per the parameters defined in Table V. In this research, we have used twenty physical servers

TABLE V
DC CONFIGURATION

Region	Architecture	Operating System	Virtual Machine Monitor	Cost per VM Hour	Cost per Mb Memory Hour	Storage cost per Gb	Data Transfer cost per Gb	No of server
0-5	eX86	Linux	XEN	0.1$	0.05$	0.1$	0.1$	20

in every DC. Each physical server is configured, as shown in Table VI. Each physical server is further abstracted with VMs. We have used 25 VMs for our experiments. They are all identical, with 1024 MB memory, 10000 MB image size, and 1000 Mbps bandwidth.

TABLE VI
PHYSICAL SERVER CONFIGURATION

Memory	Storage	Available Bandwidth	Number of Processor	Processor Speed	Virtual Machine Policy
2048 MB (2GB)	100000 MB	10000	4	100	Time Shared/ Space Shared

C. Load balancing Policy

In the DC, the load balancing policy is vital in managing the user requests among the VMs. DCs are a collection of physical servers with identical or different configurations. Cloud computing uses abstraction over the DCs, which are called VMs. The tasks are processed on these VMs. Load balancing dynamically distributes the workload at the DC among different VMs so that they are not overwhelmed with

the tasks or remain idle because of no tasks. A load balancing policy ensures the proper resource utilization of the DC. Many load balancing policies are available in the cloud services, such as First Come-First Serve (FCFS), Round Robin, Equally Spread Current Execution Load, and Throttled load balancing policy. The round robin load balancing policy distributes the tasks among the VM in a rational order of the VMs. FCFS chooses the first VM until it is loaded with tasks and no longer responds. The throttled load balancing algorithm marks the VM as busy or available to assign tasks. Equally Spread Current Execution load balancing algorithm equally distributes the tasks among different VMs. In this work, we have used the round-robin load balancing policy for all the experiments.

D. Experimental Scenarios

We have used five cloud configurations to examine the correctness of our proposed algorithm. Table VII shows the cloud configurations used for the experiments.

TABLE VII
CLOUD CONFIGURATIONS

Configuration Name	User base Configuration	DC Configuration
Cloud Configuration 1 (CC1)	UBC1	4 DCs, 2 in region 0, 1 in region 2 and 1 in region 3
Cloud Configuration 2 (CC2)	UBC3	4 DCs, 2 in region 0, 1 in region 2 and 1 in region 3
Cloud Configuration 3 (CC3)	UBC2	4 DCs, 2 in region 0, 1 in region 2 and 1 in region 3
Cloud Configuration 4 (CC4)	UBC1	4 DCs, 1 in region 0, 1 in region 2, 1 in region 4 and 1 in region 3
Cloud Configuration 5 (CC5)	UBC2	4 DCs, 1 in region 0, 1 in region 2, 1 in region 4 and 1 in region 3

VII. ANALYSIS

TABLE VIII
AVERAGE RESPONSE TIME FOR DIFFERENT ALGORITHMS (IN MILLISECONDS)

Configuration Name	DE	ORT	CDC
CC1	937.44	2719.48	2752.09
CC2	1758.34	2500.77	4942.96
CC3	768.55	882.01	2429.67
CC4	904.08	812.23	2419.39
CC5	1407.91	1667.57	3901.81

Table VIII and Table IX show the average response time and data processing time between the proposed differential evolution (DE) algorithm and traditional algorithms for different scenarios in milliseconds (ms). The results show that the CDC policy performs the worst among all the service broker policies. The CDC policy chooses the nearest DC of the userbase. As a result, the closest DC gets overcrowded with user bases in the same region. ORT policy picks the DC with the best response time and yields better results than CDC, as evidenced by the results in Table VIII and Table IX, but it also overcrowds the DC with the best response time.

From Table VIII and Table IX, we can see the results of the proposed differential evolution algorithm. Our proposed

TABLE IX
AVERAGE DATA PROCESSING TIME FOR DIFFERENT ALGORITHMS (IN MILLISECONDS)

Configuration Name	DE	ORT	CDC
CC1	568.82	2151.20	2343.70
CC2	1288.32	1977.13	4205.59
CC3	460.72	662.20	2108.89
CC4	575.65	601.34	2100.54
CC5	943.16	1356.07	3514.47

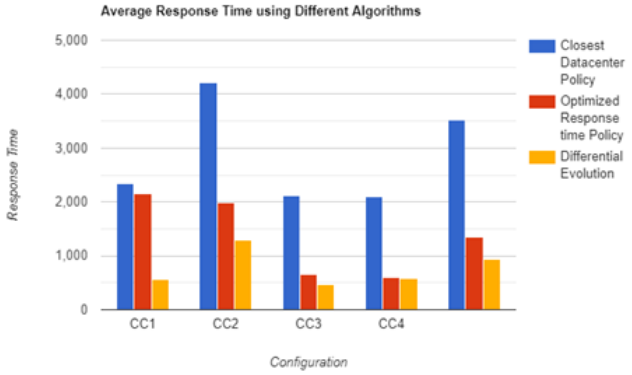

Fig. 7. Comparison of Response time with DE and traditional algorithms

DE outperforms the traditional algorithms in all test case scenarios. DE algorithms use the mutation step to explore different search spaces. As a result, it has a high possibility of finding the optimal DC for the userbase compared to traditional algorithms such as ORT and CDC. Again, the response and data processing times are changing dynamically as the user requests come dynamically. So we need to execute the algorithm several times based on the task arrival rate. In our research, we run the algorithm once every hour. In our proposed algorithm, we store the previous optimal solutions. If the current optimal solution is better than the old one, then we use that solution to replace the old solution. “Fig 7” and “Fig 8” show that our proposed DE algorithm provides better response and data processing times than traditional algorithms like ORT and CDC.

VIII. CONCLUSION

User satisfaction is essential for the success of cloud services. DC selection policy plays a vital role in cloud computing. DC processes the user requests and sends the response back to the user. So optimal DC selection is critical to process the requests efficiently and maintain user satisfaction.

This paper proposes a differential evolution-based DC selection policy to choose the optimal DC for the userbases. The differential evolution algorithm is one of the critical evolutionary algorithms which progresses with time and finds the optimal solutions from the search space. The algorithm has three steps which are mutation, crossover, and selection. Mutation helps the algorithm to change the search space and

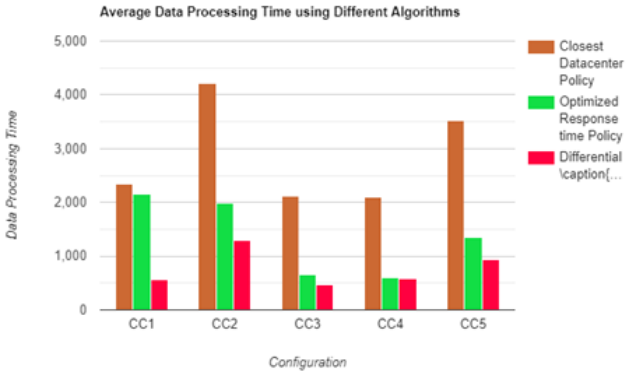

Fig. 8. Comparison of Data processing time with DE and traditional algorithm

look for more optimal solutions than the previous ones. The results indicate that the proposed algorithm performs satisfactorily compared to the other existing algorithms like ORT and CDC. Therefore the proposed DE Datacenter selection policy can be used by the cloud providers for better performance in the cloud environment.

In the proposed DE algorithm, we used one of the mutation strategies to achieve better performance. In the future extension of this work, we will test other mutation strategies, along with selection and crossover strategies, to analyze the changes in cloud environment performance. We plan to implement different evolutionary algorithms such as swarm optimization, genetic algorithm, ant colony optimization, etc.

REFERENCES

[1] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, "On the use of cloud computing for scientific workflows," in *2008 IEEE fourth international conference on eScience*. IEEE, 2008, pp. 640–645.

[2] "Top 10 cloud computing service providers in 2017," 2017. [Online]. Available: https://www.technavio. com/blog/top-10-cloud-computing-service-providers-2017

[3] H. Patel and R. Patel, "Cloud analyst: an insight of service broker policy," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 1, pp. 122–127, 2015.

[4] V. Sharma, R. Rathi, and S. K. Bola, "Round-robin data center selection in single region for service proximity service broker in cloudanalyst," *International Journal of Computers & Technology*, vol. 4, no. 2a1, pp. 254–260, 2013.

[5] A. Jaikar, G.-R. Kim, and S.-Y. Noh, "Effective data center selection algorithm for a federated cloud," *Advanced Science and Technology Letters*, vol. 35, pp. 66–69, 2013.

[6] D. Chudasama, N. Trivedi, and R. Sinha, "Cost effective selection of data center by proximity-based routing policy for service brokering in cloud environment," *International Journal of Computer Technology and Applications*, vol. 3, no. 6, pp. 2057–2059, 2012.

[7] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[8] M. Radi, "Weighted round robin policy for service brokers in a cloud environment," in *The International Arab Conference on Information Technology (ACIT2014), Nizwa, Oman*, 2014, pp. 45–49.

[9] D. Kapgate, "Improved round robin algorithm for data center selection in cloud computing," *International Journal of Engineering Sciences and Research Technology*, vol. 3, no. 2, pp. 686–691, 2014.

[10] A. S. Ahmed, "Enhanced proximity-based routing policy for service brokering in cloud computing," *Int. J. Eng. Res. Appl.(India)*, vol. 2, no. 2, pp. 1453–1455, 2012.

[11] A. Semwal and P. Rawat, "Performance evaluation of cloud application with constant data center configuration and variable service broker policy using cloudsim," *International Journal of Enhanced Research In Science Technology & Engineering*, vol. 3, no. 1, pp. 1–5, 2014.

[12] D. Kapgate, "Weighted moving average forecast model based prediction service broker algorithm for cloud computing," *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 2, pp. 71–79, 2014.

[13] P. Rekha and M. Dakshayini, "Service broker routing polices in cloud environment: a survey," *International Journal of Advances in Engineering & Technology*, vol. 6, no. 6, p. 2717, 2014.

[14] O. M. Naha Ranesh Kumar, "Cost-aware service brokering and performance sentient load balancing algorithms in the cloud," *Journal of Network and Computer Applications*, vol. 75, pp. 47–57, 2016.

[15] D. Arya and M. Dave, "Priority based service broker policy for fog computing environment," in *International Conference on Advanced Informatics for Computing Research*. Springer, 2017, pp. 84–93.

[16] A. M. Manasrah, T. Smadi, and A. ALmomani, "A variable service broker routing policy for data center selection in cloud analyst," *Journal of King Saud University-Computer and Information Sciences*, vol. 29, no. 3, pp. 365–377, 2017.

[17] J. Zhang, H. Huang, and X. Wang, "Resource provision algorithms in cloud computing: A survey," *Journal of network and computer applications*, vol. 64, pp. 23–42, 2016.

[18] D. Limbani and B. Oza, "A proposed service broker strategy in cloud-analyst for cost-effective data center selection," *International Journal of Engineering Research and Applications, India*, vol. 2, no. 1, pp. 793–797, 2012.

[19] Z. Benlalia, A. Beni-hssane, K. Abouelmehdi, and A. Ezati, "A new service broker algorithm optimizing the cost and response time for cloud computing," *Procedia Computer Science*, vol. 151, pp. 992–997, 2019.

[20] M. A. Khan, "Optimized hybrid service brokering for multi-cloud architectures," *The Journal of Supercomputing*, vol. 76, no. 1, pp. 666–687, 2020.

[21] Y. Kessaci, N. Melab, and E.-G. Talbi, "A pareto-based genetic algorithm for optimized assignment of vm requests on a cloud brokering environment," in *2013 IEEE congress on evolutionary computation*. IEEE, 2013, pp. 2496–2503.

[22] A. Patrizio, "Cloud computing costs and comparisons," Jan 2023. [Online]. Available: https://www.datamation.com/cloud/cloud-costs/

[23] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," *Advances in intelligent systems, fuzzy systems, evolutionary computation*, vol. 10, no. 10, pp. 293–298, 2002.

[24] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 482–500, 2011.

[25] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE transactions on evolutionary computation*, vol. 15, no. 1, pp. 4–31, 2010.

[26] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[27] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," *Advances in intelligent systems, fuzzy systems, evolutionary computation*, vol. 10, no. 10, pp. 293–298, 2002.

[28] "Top 10 cloud computing service providers in 2017," 2017. [Online]. Available: https://www.technavio. com/blog/top-10-cloud-computing-service-providers-2017

[29] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[30] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *2010 24th IEEE international conference on advanced information networking and applications*. IEEE, 2010, pp. 446–452.