IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2019, Doi Number

Improving the Response Time of M-Learning and Cloud Computing Environments Using a Dominant Firefly Approach

Kaushik Sekaran[1], Mohammed S. Khan[2], Rizwan Patan[3*], Amir H. Gandomi[4], Venkata Krishna Parimala[5], Suresh Kallam[6]

[1] Department of Computer Science and Engineering, Vignan Institute of Technology & Science, Hyderabad, India
[2]Department of Computing, East Tennessee State University, Johnson City, TN, USA
[3,6] School of Computing Science & Engineering, Galgotias University, U.P., India
[4]School of Business, Stevens Institute of Technology, Hoboken, NJ 07030, USA
[5] Department of Computer Science and Engineering, Sri Padmavati Mahila Visvavidyalayam, Govt. State Level University, Tirupati, India
Corresponding author: Patan Rizwan (e-mail: prizwan5@gmail.com).

ABSTRACT Mobile learning (m-learning) is a relatively new technology that helps students learn and gain knowledge using internet and Cloud computing technologies. Cloud computing is one of the recent advancements in the computing field that makes internet access easy to end users. Many Cloud services rely on Cloud users for mapping Cloud software using virtualization techniques. Usually, Cloud users' requests from various terminals will cause heavy traffic or unbalanced loads at the Cloud data centers and associated Cloud servers. Thus, a cloud load balancer that uses an efficient load balancing technique is needed in all the cloud servers. We propose a new meta-heuristic algorithm named the dominant firefly algorithm that optimizes load balancing of tasks among multiple virtual machines (VMs) in the Cloud server, thereby improving the response efficiency of Cloud servers that concomitantly enhances the accuracy of m-learning systems. Our methods and findings used to solve load imbalance issues in Cloud servers, which will enhance the experiences of m-learning users. Specifically, our findings such as Cloud SQL Structured Query Language, querying mechanism in mobile devices will ensure users receive their m-learning content without delay; additionally, our method will demonstrate that by applying an effective load balancing technique would improve the throughput and the response time in mobile and cloud environments.

INDEX TERMS Cloud computing, dominant firefly algorithm, load balancing, Mobile learning (m-learning), virtual machines

I. INTRODUCTION

Many computing methodologies are available in the computing field for maximizing automation. Among those, m-learning and Cloud computing are considered to be the best service oriented computing technologies to automate tasks in virtual machines as well as to enable users to access information very efficiently. Also, m-learning offers cost-effective solutions for a wide range of services.

Mobile learning and Cloud computing are two essential domains to explain distributed data sharing [23]. In m-learning, mobile devices used by end users are called the m-learning clients. Through internet connectivity, m-learning clients store and retrieve data from Cloud data centers. Hence, m-learning systems integrated with Cloud data centers are quite advantageous for transferring all types of data and applications to mobile device easily and accurately. However, load balancing issues in Cloud data centers should be addressed to improve performance and efficiency. In this paper, we propose a meta-heuristic algorithm to overcome this load balancing issue.

M-learning technologies have been deployed in many m-learning systems and applications to improve the learning styles of current students. On average, m-learning technologies enhance the learning capacity of individuals by 70% [22]. Some of the Cloud computing services that could be used for m-learning approaches are Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), Infrastructure as-a-Service (IaaS), and Hardware-as-a-Service (HaaS).

Load balancing techniques are used to distribute incoming traffic across multiple servers to minimize the delay of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2896253, IEEE Access

Cloud server response to the Cloud users. Cloud load balancing is considered adequate only if the throughput in the Cloud server is high, delays are minimal, and jitter is minimal while addressing Cloud user requests. Sometimes, failure of load balancing in the Cloud leads to poor image resolution and poor video streaming for users [24]. Thus, load balancing in Cloud servers is essential to maximize throughput and to achieve superior performance in both public and private Clouds.

II. LITERATURE REVIEW

In literature review, we have discussed in detailed about various load balancing methods in Cloud computing and findings used to provide various methods and procedures for assigning the client's requests to available Cloud nodes. Cloud load balancing scenarios advantages are scalability and agility to meet rerouted workload demands and to improve overall availability more details provided as follows:

2.1 Load balancing in Cloud computing

The concept of load balancing was first recognized as an important issue for computing in the year 2001. In [1], they proposed a few genetic algorithms that are essential to load balancing techniques. Additionally, the authors investigated dynamicity in load balancing approaches. Also, they described an algorithm that can optimally balance the loads in parallel computing systems during process mapping. Also, they discussed other load balancing issues. In [2], described the deficit round-robin load balancing technique for scheduling of tasks that was based on an efficient fair queuing load balancing technique. In publications by [3], it was proposed that the Cloud scheduler could be based on an ant colony optimization method. Several studies and reviews are useful in understanding scheduling of tasks in the Cloud load balancer [4].

In [5], they mentioned configuring Cloud storage services as an Infrastructure as a Service model, which was the key point for the most of the researchers who works on Cloud load balancing issues. In [6], the authors have surveyed many patterns of Cloud computing and also proposed the knowledge level framework for the development and inclusion in Cloud computing technologies. Authors in [7] have proposed partition-based techniques for load balancing in Cloud computing based on the switch mode mechanism. Also, they discussed a game theory model of Cloud computing. Authors in [8] have suggested a dynamic load balancing technique through mobile agents for Peer to Peer (P2P) networks. In their paper, a resource grouping strategy and a dynamic load balancing methodology among the groups were proposed by identifying the congestion in network.

Krishna et al., [9] proposed an algorithm for load balancing of tasks that was inspired by honey bee behavior. In this paper, load balancing of tasks is created in virtual machines to achieve maximum throughput. In [10], they proposed a stochastic hill approach algorithm to balance loads in Cloud computing. For this, jobs are mapped to the Cloud server and to virtual machines instances in a shorter span of time.

In [11], they analyzed fast downloading of files in the Cloud with dynamic load balancing using a dual direction technique. The authors also introduced an efficient and effective technique to download large files from different Cloud data centers in a more efficient way than previous load balancing methods like [12] suggested in evolutionary algorithms. An article by [13], addressed challenges in the m-learning environments such as the security of mobile devices in the corresponding mobile network. In [14], the authors proposed a firefly load balancing algorithm for balancing Cloud computing loads. The results showed better performance in terms of computational time, task migration, and load arrival ratio. The authors [15], discussed the importance of load balancing in Cloud computing, various types of loads in the Cloud, and the importance of proper utilization of resources.

In [16], the authors discussed genetic algorithms used for dynamic load balancing in the distributed environment, problems of load balancing, and compared genetic algorithms given by [17] with other algorithms. In [18] discussed load balancing techniques with improvements in many Qualities of Service (QoS) metrics. Several security issues and scheduling problems were discussed and compared with existing load balancing algorithms used in Cloud computing. In Xin-She Yang firefly algorithm [7], the author talked only about modelling and Multimodal Optimization. But our dominant firefly algorithm model is for optimizing the response time for M-Learning environments especially for cloud data centers issues. Also, the algorithms we have taken for calculating the Makespan and designed for efficient load balancing. However, our focus is on to the load balancing and response time comparison with the previous genetic algorithms.

In [19], authors proposed an algorithm for load balancing using a genetic algorithmic approach designed for task minimization in Cloud computing platforms. By comparing with traditional algorithms, the authors have proved that their algorithm requires minimal time to finish load balancing in the Cloud computing environment. This load balancing algorithm focused on infrastructure as a service (IaaS) model. The load balancing scenario of scheduling work in scientific Clouds was evaluated by creating a number of virtual machines instances and hosting them in the respective Cloud [20].

2.2 Cloud load balancing algorithms limitations

Some of the load balancing algorithms and their performance have been studied and analyzed. Qualities of Service (QoS) metrics with the proposed load balancing algorithm were compared with other load balancing algorithms (Table I). Out of all compared algorithms, the proposed algorithm showed better QoS metric results in terms of service delay, throughput, service availability, response time, network overhead, and authentication.

TABLE I

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2896253, IEEE Access

QoS METRICS COMPARISON OF THE PROPOSED LOAD BALANCING ALGORITHM WITH OTHER RELATED ALGORITHMS

Load balancing algorithms	Quality of Service Metrics						
	QoSM 1	QoSM 2	QoSM 3	QoSM 4	QoSM 5	QoS M6	QoS M7
LBA1	Y	N	Y	N	N	N	N
LBA2	Y	Y	Y	Y	N	N	N
LBA3	Y	Y	Y	Y	Y	Y	N
LBA4	Y	N	N	N	Y	N	N
LBA5	Y	Y	Y	Y	N	Y	N
LBA6	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**

Y: achieved; N: not achieved.

Notes: The compared QoS metrics are represented as QoSM. They are:

- QoSM1: Throughput
- QoSM2: Service Delay
- QoSM3: Response time
- QoSM4: Network Overhead
- QoSM5: Service availability
- QoSM6: Authentication mechanisms
- QoSM7: Servers performance

Notes: The load balancing algorithms are represented as LBA. They are:

- LBA1: "Round-Robin Load Balancing" by Shreedhar et al. (1996)
- LBA2: "Honey bee behavior Load Balancing (HBB-LB)" by Krishna (2013)
- LBA3: Ant Colony Optimization Load Balancing by Rao, Lei, et al. (2010) and Mishra et al. (2012)
- LBA4: Particle Swarm Optimization Load Balancing by Jin, Xiaoling, et al. (2004)
- LBA5: Delay-tolerant dynamic load balancing by Mohamed, Nader et al. (2011)
- LBA6: Proposed Dominant Firefly Load balancing algorithm.

III. SYSTEM MODEL

In propose system the dominant firefly load balancing algorithm to solve load imbalance issues in Cloud servers, to enhance the experiences of m-learning users. Specifically, Dominant firefly-based required Cloud server mapping algorithm for different VM methods will help ensure users receive their m-learning content without delay; additionally, in this technique, that demonstrates the load balancing improvement on throughput and the response time of mobile devices.

3.1 M-learning in Cloud computing

The importance of m-learning technologies has become quite apparent to many different institutions and individuals in recent years. The researchers [21] described the m-learning environment as one that changes the traditional learning system and gives freedom to learners. Also, m-learning has no boundaries for learning through mobile devices. Also [22], stated that "m-learning is a kind of e-learning which combines mobile technology and Cloud computing wireless technology for a better learning experience.

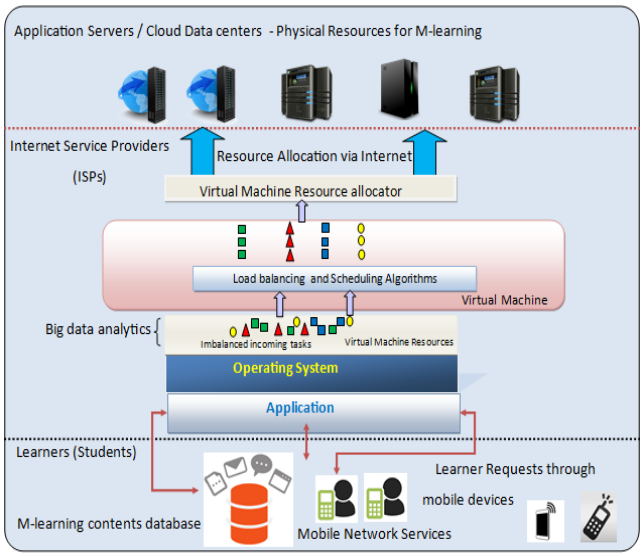

FIGURE 1. M-learning with Cloud computing architecture

3.2 Biological model of dominant firefly behavior

The firefly and its behavior for finding food sources and searching for partners are quite interesting. Fireflies that produce the most intense brightness are called dominant fireflies and others with less luminescence are called submissive fireflies. Also, the glow of the fireflies' brightness is akin to an on and off switch. In every four to six seconds, the firefly's tail will be on then off, usually visible during the late evening and night time.

The path selection of the firefly is an another interesting pattern in which fireflies find the optimal distance to reach its partner. Maximum brightness depends upon the distance of the location of the firefly with respect to its partner. A graphical representation of firefly behavior is shown in Figure 2. In firefly search behavior, the submissive fireflies are searching for the dominant firefly with its brightness value (BV).

Let F_1, F_2, F_3……F_n be the submissive fireflies in a group of fireflies F.

Let DF_1, DF_2….DF_n be the dominant fireflies.

By luminescence, dominant fireflies can attract other neighboring fireflies. In a given regions, all fireflies are attracted to a dominant firefly producing more brightness.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2896253, IEEE Access

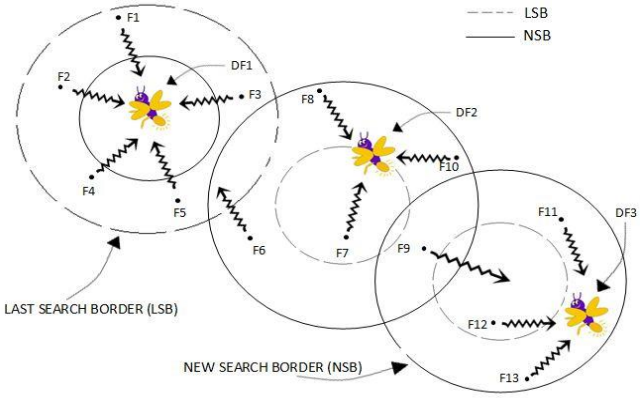

FIGURE 2. **Firefly behavior for finding the dominant firefly.**

It is assumed that submissive fireflies fly toward the dominant firefly in two flying patterns:

1. Flying away from less brightness; this value is denoted by last search border (LSB) given in equation 1.
2. By flying towards more brightness; this value is denoted by new search border (NSB) given in equation 2 and 3.

Procedure:

If $BV(DF_{n+1}) < BV(DF_n)$

Then

$$\forall \sum F_n \in \text{LSB flies to NSB} \quad (1)$$

$$\text{Else if } BV(DF_n) < BV(DF_{n-1}) \quad (2)$$

Then

$$\forall \sum F_n = DF_n \text{ in NSB} \quad (3)$$

Fireflies searching for partners through an optimal path and the associated crowd balancing on dominant fireflies are optimized through shorter flying distance, thus saving energy from flying over longer distances. The Euclidean distance calculation for searching $\forall DF_n$ is applicable for the fireflies' flying distance path. As per equation 4, which is an Euclidean distance formula, if (p,q) are two coordinates, then the distance between the two coordinates is calculated by,

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (qi - pi)^2} \quad (4)$$

In the same way, according to our proposed biological model, the distance between two fireflies (i.e., $\forall DF_n$ and $\forall F_n$) will be calculated using equation 5.

$$d(\forall \sum [DF]_n, \forall \sum F_n) = \sqrt{\left(\sum_{i=1}^{n} [DF_n] - F_n\right)^2} \quad (5)$$

Through this, it is clearly understood that a firefly from a group of fireflies to the nearest border by spending very less energy in its tail. Based on this scenario, a bio-inspired computing model was created, and a dominant firefly search algorithm was implemented in the Cloud computing environment for testing improved response times among the Cloud data centers based on the concept of low-loaded VMs with less task migration time. The proposed dominant firefly search can also be mapped with adjacency matrix representation to understand the load balancing algorithm clearly in a matrix form (Figure 3).

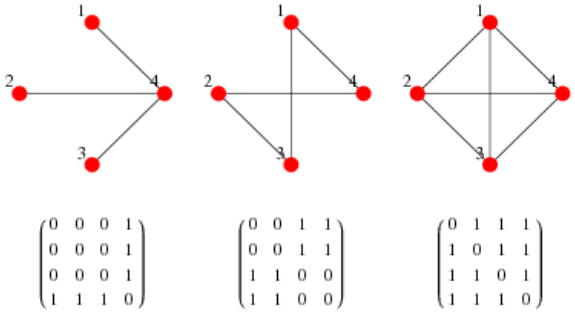

FIGURE 3. **Adjacency matrix representation of dominant firefly behavior**

Figure. 3 depicts an adjacency matrix representation of dominant firefly behavior. There are four fireflies, F_1, F_2, F_3, and F_4, with F_4 as the dominant firefly (i.e., the other fireflies are moving towards F_4). Hence, the adjacency matrix can be represented as 1->4, 2->4, 3->4.

3.3 Bio inspired computing model

Dominant firefly behavior can be applied to Cloud load balancing strategies, termed the dominant firefly algorithm. In a group of fireflies, there will be several dominant fireflies and many submissive fireflies. The method assumes that dominant fireflies represent Cloud servers and submissive fireflies represent Cloud users. Whenever the Cloud servers are occupied with a lot of load (user requests), this needs to be balanced in such a way that queries or requests are transferred to some other Cloud server to complete the task. Based on firefly behavior, it is understood that if dominant fireflies are already occupied with many other submissive fireflies during partner searching, then the load is balanced by passing on excess submissive fireflies to the next dominant firefly. According to this algorithm, when Cloud user requests are increased to a particular Cloud server, then users are automatically transferred to the next (dominant) Cloud server. Also, the flight path of submissive fireflies towards the dominant firefly represents nearby Cloud servers that provide the dynamicity of load balancing.

The design of dominant firefly algorithm is based on a dynamic load balancing strategy among VMs in the Cloud environment. Regarding larger-scale scenarios such as the Cloud load balancing strategy, this dominant firefly algorithm would be able to search for and find the optimal and nearest Cloud server in the pool of Cloud servers, thereby achieving optimal load balance among multiple Cloud server VMs. This finding was designed to work simpler and more efficiently to solve complex load balancing of tasks in Cloud computing environments. Path finding by Cloud servers can be optimized by applying this proposed dominant firefly algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2896253, IEEE Access

Cloud service providers provide the best cost reduction policies to end users for accessing their IaaS services in the Cloud with a valid Cloud SLA (Service Level Agreement). Each Cloud user should have an SLA while communicating with neighboring Cloud servers to ensure flexible and trustworthy sharing of files. Figure. 4 represents task migration among multiple Cloud server VMs.

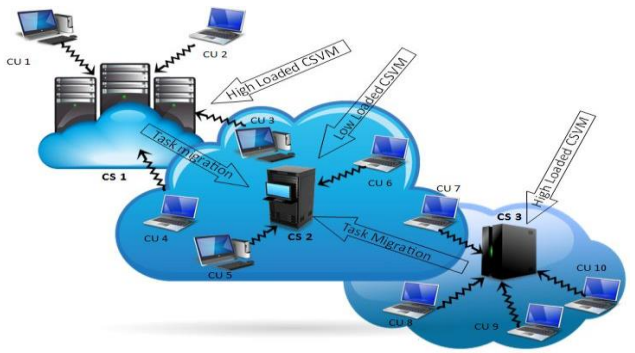

FIGURE 4. Task migration among multiple Cloud server VMs Scenario of mobile learning with Cloud computing

In this scenario, it is assumed that CS1, CS2, CS3,…CSn are Cloud servers and CU1, CU2, CU3…CU10 are Cloud users. In Cloud computing, Cloud users continuously send requests to the available Cloud server resources. Request mapping to the expected resource is a tedious process. In the proposed technique, a concept of a Cloud server pool of VMs is introduced to map every user request to the Cloud server. Cloud user requests may be given to any Cloud server randomly in the Cloud. Cloud server choice in the method relies on basic Cloud management policies that are dependent on the actual load on every Cloud server.

Scheduling policies have been adopted for any non-preemptive system such as Round Robin and First In First Out policies. By placing these kinds of strategies into a queue-based operation, the loads on every Cloud server virtual machine (CSVM) are found to be different. To make the loads balanced across every CSVM, a dynamic load balancing strategy is needed.

Load balancing is also going to be beneficial for the overall Cloud by reducing response time of the Cloud server as well as job shop scheduling problems (makespan), which is discussed in similar meta-heuristic techniques such as honey bee behavior inspired load balancing issues [9]. Our methods and equations are similarly derived in such brief. Hence performance of every individual Cloud server has been improved. From equation 6, makespan is the total length of the schedule of the overall tasks. In the proposed algorithm, the response time of every Cloud server with respect to number of tasks is represented as T_a on VM_b as RT_{ab}. Hence, the makespan is given as the following function:

$$\text{Makespan}=\min\{RT_{ab} \mid a\in T, a=1, 2, \ldots n \text{ and } b\in VM, b=1, 2, \ldots m\} \quad (6)$$

By reducing the response time of CSVMs, efficiency is highly improved.

Let CSVM = $\{CS_1, CS_2, CS_p\}$ be the set of CSVMs which process "n" tasks, represented by the set T = $\{T_1, T_2 \ldots T_n\}$. All Cloud servers in the Cloud are represented by S. Generally, all Cloud servers are fault tolerant servers. Non-preemptive tasks are scheduled on these CSVMs. The non-preemptive tasks are represented as T_{np}.

To reduce makespan, the proposed model is given as $S| T_{np} |RT_{min}$.

The response time of every CSVM is represented as RT_x on $CSVM_y$ -> R_{xy}.

Hence, the response time of all tasks on the CSVMs is defined in equation 7 and, by further maximizing the RT_{min} value, every CSVM response time is calculated by equation 8 and equation 9.

$$R_y \sum_{i=1}^{p} Rxy \qquad y=1, 2, \ldots q \quad (7)$$

By maximizing RT_{min} , we have,

$$\sum_{i=1} Rxy \geq RT_{min} \qquad y=1, 2, \ldots q \quad (8)$$

$$\Rightarrow R_y \geq RT_{min} \qquad y=1,2,\ldots q \quad (9)$$

During load balancing of multiple VMs, the migration of tasks occurs between one VM to the next. Equation 10 minimizes the overall response time and efficiently balances the load.

$$RT_{min} = \{[min]_{(i=1)}^{q}[RT]_i, [min]_{(j=1)}^{q}\sum_{(i=1)}^{q} \equiv R_y\} \quad (10)$$

The response time to the incoming tasks in all the CSVMs automatically have been minimized, which is denoted in equation 10.

3.4 Design of algorithms

To validate the dominant firefly search behavior strategy for balancing loads, we have developed and tested our methods both in the Cloud and mobile environments. The results demonstrated were better when compared with existing load balancing methods.

In the proposed Algorithm-1, each firefly that flies toward the other firefly represents a VM, which can be mapped to the Cloud server accordingly. When a VM is formed, a firefly is initialized. An index provider that contains load information on each Cloud server is initialized through initialize Load Table method. Then, if a firefly is mapped to the VM by the given method, then the firefly is obtained from a group of fireflies through the Firefly VM method. If the VM does not exist in the Firefly Group, then a new Firefly is created along with the VM.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2896253, IEEE Access

Algorithm 1. Dominant firefly-based required Cloud server mapping algorithm for different VM.

Procedure Dominant Firefly Search and Make Strategy for VM, Cloud Server Index

Start
Initialize Load Table
Firefly is equals to Firefly method mapping in VM
if Firefly is equal to null **then**
required Cloud Server is equals to Required Cloud Server for VM in Cloud Server Index
Firefly is equals to new Firefly in VM for required Cloud Server
Add VM group in the Firefly method
End if condition
repeat
Call the Firefly and dominant Firefly Algorithm
until Firefly is Completed
Required Cloud Server is equals to Cloud Server Index for mapping Firefly method in Cloud server
if required is not Cloud Server Make VM map to get Firefly method
repeat
Firefly Search and Make Strategy for getting Firefly VM method in Cloud Server Index
Number of Repeated Fly Over Group
until Success or Number of Repeated Fly Over Group is equals to NULL
End

Algorithm 2. Dominant Firefly Algorithm

Procedure dominant Firefly Algorithm

Start Process

Brightness value is 1
Cloud user requests are 0
Initialize
While the brightness is extreme brightness then do
Current Load is equal to Cloud Server Load Information
Add Firefly Group Total in the current Load
Update the local Load Table
if the current Load is equal to 0 break
else if random less than dominant Firefly then
Next Cloud Server is equal to Randomly select Dominant Cloud Server
Else next Cloud Server is equal to select Dominant Cloud Server
end if
Dominant Cloud Server is equal to Decreasing task eventually in the Over Loaded
Cloud Servers from Dominant Cloud Server
Cloud user requests are equal to Cloud user requests increment by 1
Brightness is equal to Brightness increment by 1
Fly or map VM to next Cloud Server
end while condition
send VM to Cloud Server
End

A Cloud server with larger QoS metrics such as higher throughput and large bandwidth is required, as compared to the requirements of a Loaded VM. The firefly which is flying toward the dominant firefly is mapped with the adding Firefly Group into VM method.

Figure 5 and Figure 6 represents flow-charts for better understanding of the above algorithms.

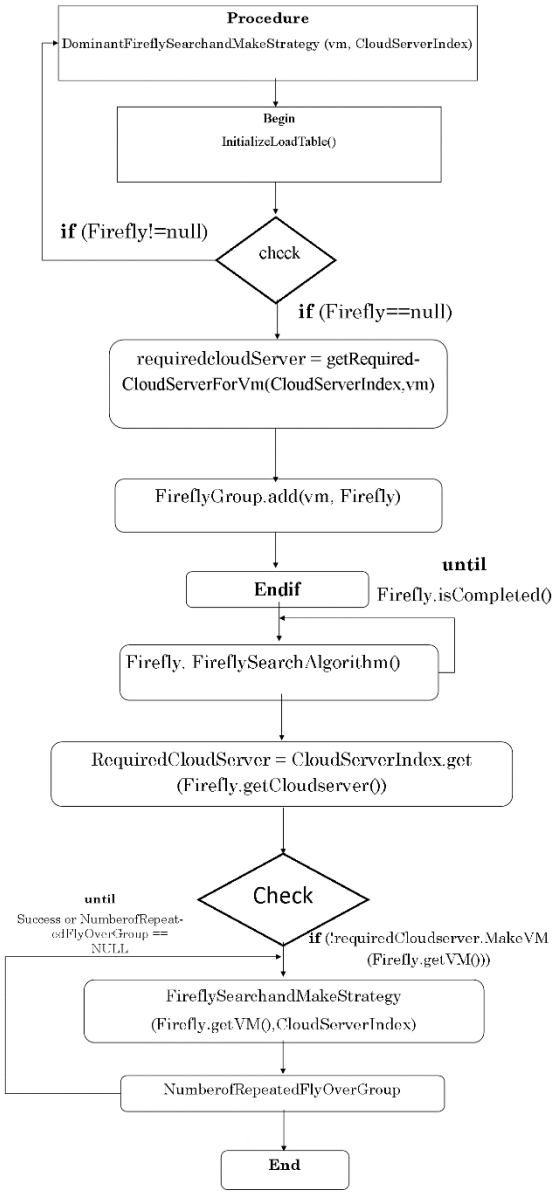

FIGURE 5. **Flow chart for DFA (Dominant Firefly) search method**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2896253, IEEE Access

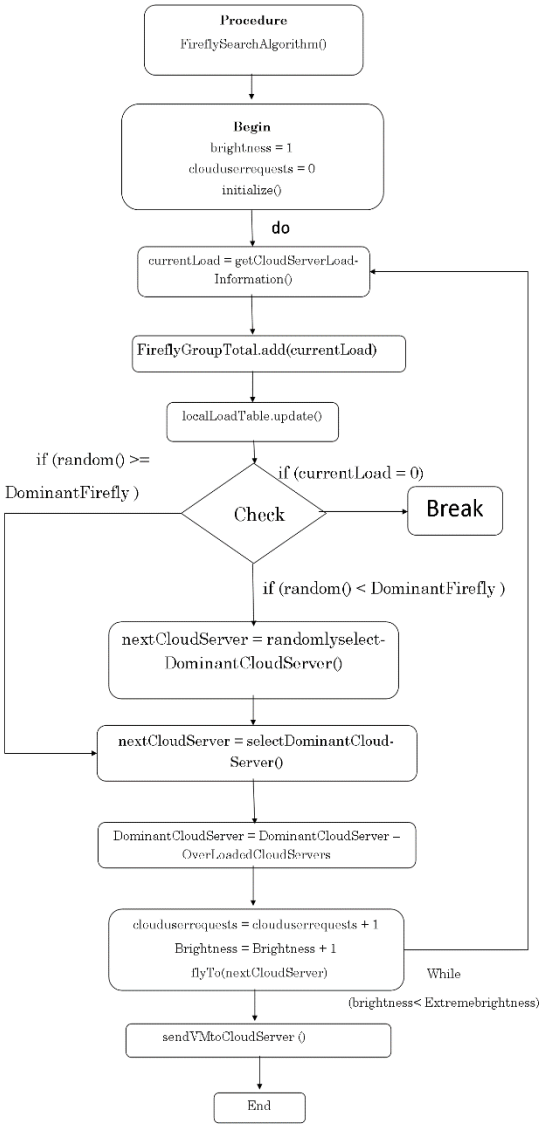

FIGURE 6. Flow chart for BV (Brightness Value) and VM mapping method for selecting cloud servers

IV. EXPERIMENTAL SETUP

Based on the number of tasks coming into the Cloud server, the load is analyzed in all Cloud servers and its response time is calculated using CloudSim simulation. For evaluation of receiving results from the Cloud server, m-learning systems were tested using the mobile test bed in java based mobile phones. We used Cloud SQL querying mechanism to improve analysis of the result and evaluation. As shown in Table 2, the average response time increased whenever the number of queries generated by m-learning users also increased.

TABLE II
BEFORE LOAD BALANCING IN CLOUD DATA CENTER: M-LEARNING USERS VS AVERAGE RESPONSE TIME (MS) BY GENERATING SIMPLE QUERIES(SQ) IN THE CLOUD DATA CENTER

Number of m-learning users (n)	No. of (SQ)Queries generated (n*4)	Avg. Response time(ms)
30	120	2400
40	160	3200
50	200	6000
60	240	7200

The response time is calculated using CloudSim simulation.
Steps in Cloudsim Simulation environment:

Step-1
Firstly, Eclipse IDE can be downloaded from the website www.eclipse.org/downloads

Step-2
We have extracted the eclipse software package to a specific directory. eg: C:\Program files\eclipse

Step-3
Then we have extracted the Cloudsim package to a specific directory.eg: C:\Program files\Cloudsim and we have included the jar files in the cloudsim in the location C:\Program files\cloudsim-3.0.2\jars\

Step-4:
We have created java project in eclipse and coded for the cross region based load balancing from algorithm 1 and algorithm 2 in the eclipse workspace.

Step-5
We have extracted the Cloudsim-3.0.2 folder in it and ran the DFA search load balancing code.

Step-6
Then we have created the required number of Datacenters, Brokers, virtual machines and cloudlets.

Step-7
Then by starting the simulation by testing it with the number of datacenters applicable to the respective dominant firefly search scenario and ran the simulation for multiple
times with different MIPS requirements for the cloudlets execution time.

Step-8
Finally we have analyzed the VMs performance by calculating the output and drawing the graph according to the statistical data we have during the execution.

Sample, Simple Queries (SQ) generated by m-learning users:
SQ-Query 1: Confirmation SMS sent to the server by the mobile device. [Approx ~ 20 KB]
SQ-Query 2: Generating the request_id for the m-learner from the Cloud server. [Approx ~ 20 KB]
SQ-Query 3: Testing the Interactivity parameters such as m-learning application GUI (Graphical User Interface). [Approx ~ 30 KB]
SQ-Query 4: M-learner (students) online feedback to the server by the mobile device. [Approx ~ 30 KB]

From Table II, SQ (queries with simple workloads in Cloud data center) from query1, query2, query3, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2896253, IEEE Access

query4 used in m-learning systems such as SMS (file size approximately 20KB) and generation of request_id require only minimal time for analyzing inside the server, although time is needed for the information to return to the m-learning clients. This total time is calculated as the average response time. Queries sent to the Cloud data center prior to implementation of the load balancing algorithm in and after load balancing were analyzed. Our m-learning evaluation through Cloud simulation clearly revealed better performance of m-learning systems after applying our load balancing algorithms. Also, in Table III, heavy queries represented as (HQ), such as LOAD csv file (file size approximately 36KB), typically require more response time to get back to m-learning clients. However, after applying our load balancing algorithm, response time was drastically decreased, which improved the m-learning system's overall performance.

TABLE III

AFTER LOAD BALANCING IN CLOUD DATA CENTER: M-LEARNING USERS VS. AVERAGE RESPONSE TIME (MS) BY GENERATING HEAVY QUERIES (HQ) IN THE CLOUD DATA CENTER

Number of m-learning users (n)	No. of (HQ)Queries generated (n*4)	Avg. Response time(ms)
30	120	4320
40	160	8000
50	200	8000
60	240	9600

Sample Heavy Queries (HQ) generated by m-learning users after load balancing:

(HQ)-Query 1: LOAD SMS_DATA INPATH '/user/sandbox/student.csv' OVERWRITE INTO TABLE temp_student;

[Approx ~ 36 KB]

(HQ)-Query 2: insert overwrite table student
SELECT
regexp_extract(col_value,'', 1) student_id,
regexp_extract(col_value,',1) student_name,
regexp_extract(col_value, '', 1) content
from temp_student;

[Approx ~ 50 KB]

(HQ)-Query 3: SELECT a.student_name, a.student_id, a.content from student a
JOIN (SELECT name, max(content) content FROM student GROUP BY name) b
ON (a.name = b.name AND a.content = b.content) ;

[Approx ~ 40 KB]

(HQ)-Query 4: DELETE from student where id=1;
ROLLBACK to student; Commit;[Approx ~ 40 KB]

Figure. 7 illustrates the response time of VMs in seconds for the dominant firefly algorithm, HBB-LB, ant colony optimization load balancing algorithm, and WRR algorithms. The X-axis represents the number of tasks and the Y-axis represents response time in seconds. Our proposed algorithm showed the optimal response time.

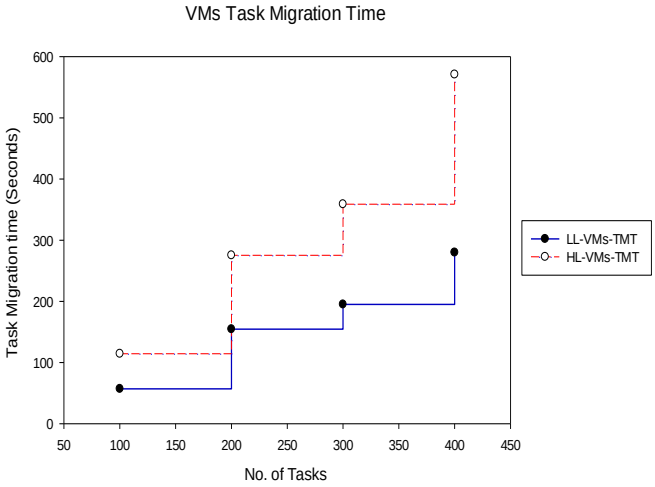

FIGURE 7. VMs task migration time Vs No. of tasks

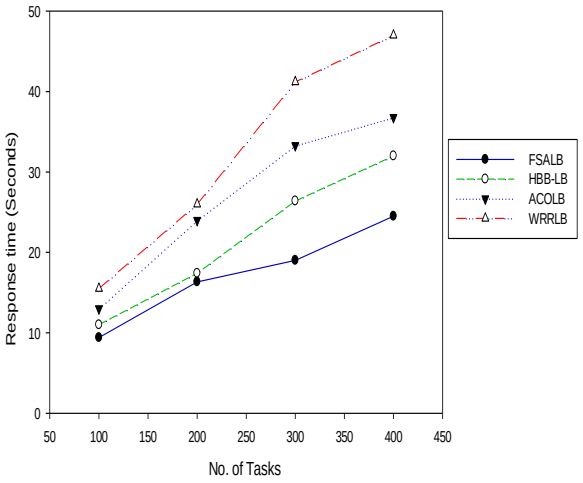

FIGURE 8. Comparison of the number of tasks with response time in seconds using the dominant firefly algorithm, HBB-LB, Ant colony optimization load balancing algorithm, WRR algorithms

Notes for Figure 8:
FSALB: Firefly Search Algorithm Load Balancing
HBB-LB: Honey Bee Behavior Load Balancing algorithm
ACOLB: Ant Colony Optimization Load Balancing algorithm
WRR-LB: Weighted Round Robin load balancing algorithm

According to the task migration in the Cloud servers VMs, performance various QoS metrics is analyzed. Figure. 8 represents the comparison of number of tasks with the high loaded VMs task migration time and low loaded VMs task migration time. The X-axis represents the number of tasks and the Y-axis represents the migration time in seconds from the proposed dominant firefly algorithm for load balancing. Also, by comparing different load balancing algorithms, we were able to find the response time of each task in the Cloud server VMs.

V. CONCLUSION AND FUTURE WORK

In this work, the proposed dominant firefly behavior search model was applied and simulated in CSVM instances to improve load balancing of tasks in the Cloud computing environment. This approach helps to balance the load in multiple CSVMs by increasing QoS metrics such as throughput and response time. Also, in our methods, the load of job requests from Cloud end-users submitted to CSVMs is optimally balanced to increase the efficiency of the Cloud server.

The proposed algorithm was compared with other load balancing algorithms. The results demonstrated an improvement in energy consumption among Cloud servers. These findings could be extended to cost computational methods to utilize maximum CPU that would increase server efficiency. The main objective of the proposed model is to enhance m-learning environments by finding many relational models to avoid the highest energy consuming server throughout the world. In the current real-time Cloud environment scenario, the throughput efficiency at the Cloud data center is an essential factor and many researchers are showing keen interest in developing various algorithms for it. Also, there are many opportunities in the field of m-learning, green computing, and in Cloud-based organizations. Our work reveals many challenges in m-learning using Cloud computing technologies.

REFERENCES

[1] Zomaya, A. Y., and Teh, Y. H. (2001). Observations on using genetic algorithms for dynamic load-balancing. Parallel and Distributed Systems, IEEE Transactions on, 12(9), 899-911.

[2] Shreedhar, M., and Varghese, G. (1996). Efficient fair queuing using deficit round-robin. Networking, IEEE/ACM Transactions on, 4(3), 375-385.

[3] Pacinia, E., Mateosb, C., and Garinoa, C. G. (2014). Balancing Throughput and Response Time in Online Scientific Clouds via Ant Colony Optimization. Advances in Engineering Software. In press. Elsevier.

[4] Sekaran, K., & Krishna, P. V. (2017). Cross region load balancing of tasks using region-based rerouting of loads in Cloud computing environment. International Journal of Advanced Intelligence Paradigms, Vol. 9, Issue No. 5-6, pp. 589-603.

[5] D.M. Eyers, R. Routray, R. Zhang, D. Willcocks, and P. Pietzuch. (2009). Towards a middleware for configuring large-scale storage infrastructures. In Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science, page 3.

[6] Fehling, C., Ewald, T., Leymann, F., Pauly, M., Rütschlin, J., & Schumm, D. (2012). Capturing Cloud computing knowledge and experience in patterns. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on (pp. 726-733). IEEE.

[7] Yang, X. S. (2009, October). Firefly algorithms for multimodal optimization. In International symposium on stochastic algorithms (pp. 169-178). Springer, Berlin, Heidelberg.

[8] Shen, X. J., Liu, L., Zha, Z. J., Gu, P. Y., Jiang, Z. Q., Chen, J. M., and Panneerselvam, J. (2014). Achieving dynamic load balancing through mobile agents in small world P2P networks. Computer Networks, 75, 134-148.

[9] Krishna, P. V. (2013). Honey bee behavior inspired load balancing of tasks in Cloud computing environments. Applied Soft Computing, 13(5), 2292-2303.

[10] Mondal, B., Dasgupta, K., and Dutta, P. (2012). Load balancing in Cloud computing using stochastic hill climbing-a soft computing approach. Procedia Technology, 4, 783-789.

[11] Mohamed, N., Al-Jaroodi, J., and Eid, A. (2013). A dual-direction technique for fast file downloads with dynamic load balancing in the Cloud. Journal of Network and Computer Applications, 36(4), 1116-1130.

[12] Civicioglu, P., & Besdok, E. (2018). A+ Evolutionary Search Algorithm and QR Decomposition Based Rotation Invariant Crossover Operator. Expert Systems with Applications.

[13] Andreas Holzinger, Alexander Nischelwitzer, Matthias Meisenberger, "Mobile Phones as a Challenge for MLearning: Examples for Mobile Interactive Learning Objects (MILOs)", *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications Workshops*, 2005.

[14] N. Susila, S. Chandramathi and Rohit Kishore, A Fuzzy-based Firefly Algorithm for Dynamic Load Balancing in Cloud Computing Environment, JOURNAL OF EMERGING TECHNOLOGIES IN WEB INTELLIGENCE, VOL. 6, NO. 4, NOVEMBER 2014

[15] Kaur, R., & Luthra, P. (2012, December). Load balancing in cloud computing. In Proceedings of International Conference on Recent Trends in Information, Telecommunication and Computing, ITC.

[16] Yu-Liang Chou, Shaoshan Liu, Eui-Young Chung, and Jean-Luc Gaudiot, Fellow, IEEE, An Energy and Performance Efficient DVFS Scheme for Irregular Parallel Divide-and-Conquer Algorithms on the Intel SCC, IEEE COMPUTER ARCHITECTURE LETTERS, VOL. 13, NO. 1, JANUARY- 2014

[17] Yang, X. S., Hosseini, S. S. S., & Gandomi, A. H. (2012). Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. Applied soft computing, 12(3), 1180-1186.

[18] Li, Kun, et al. "Cloud task scheduling based on load balancing ant colony optimization." Chinagrid Conference (ChinaGrid), 2011 Sixth Annual. IEEE, 2011.

[19] KousikDasgupta, BrototiMandal, ParamarthaDutta, JyotsnaKumarMondald and SantanuDam, AGenetic Algorithm(GA)basedLoad Balancing Strategy for CloudComputing, Elsevier Procedia Technology 10 (2013) 340–347

[20] Sekaran, K. and Krishna, P.V. (2016) 'Big Cloud: a hybrid Cloud model for secure data storage through Cloud space', International Journal of Advanced Intelligence Paradigms, Vol. 8, No. 2, pp.229–241.

[21] Tsvetozar Georgiev, Evgenia Georgieva, Angel Smrikarov, "M-Learning-a New Stage of E-Learning", International Conference on Computer Systems and Technologies-CompSysTech'04, 2004.

[22] Mitchell Keith, J. Nicholas, P. Race, Duncan McCaffery, Mark Bryson, Zhen Cai, "Unified and Personalized Messaging to Support E-Learning", 4th IEEE International Workshop on Wireless Mobile and Ubiquitous Technology in Education (ICHIT'06), 2006.

[23] Li, P., Li, J., Huang, Z., Li, T., Gao, C. Z., Yiu, S. M., & Chen, K. (2017). Multi-key privacy-preserving deep learning in cloud computing. Future Generation Computer Systems, 74, 76-85.

[24] Zhang, Q., Yang, L. T., Chen, Z., Li, P., & Deen, M. J. (2018). Privacy-preserving double-projection deep computation model with crowdsourcing on cloud for big data feature learning. IEEE Internet of Things Journal, 5(4), 2896-2903.