RESEARCH ARTICLE OPEN ACCESS

Improving the Response Time of M-Learning and Cloud Computing Environment Using Superior Element Multitude Optimization

Ms. N. Zahira Jahan M.C.A.,M.Phil.,* Ms. V.Karthika**
*Associate Professor/MCA, Department of MCA, Nandha Engineering College, Erode, Tamilnadu, India.
Email: zahirajahan1977@gmail.com
** Final MCA, Department of MCA, Nandha Engineering College, Erode, Tamilnadu, India.
Email: karthikavenkadesan98@gmail.com

****

ABSTRACT

Mobile learning (M-Learning) may be a relatively new technology that helps students learn and gain knowledge using internet and Cloud computing technologies. Cloud computing is one among the recent advancements within the computing field that creates internet access easy to finish 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 required altogether the cloud servers. In 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, by improving the response efficiency of Cloud servers that concomitantly enhances the accuracy of m-learning systems. The methods and findings wont to solve load imbalance issues in Cloud servers, which can enhance the experiences of M-Learning users. Here findings such as Cloud SQL Structured Query Language, querying mechanism in mobile devices will ensure users receive their M-Learning content without delay. This method will demonstrate that by applying an effective load balancing technique would improve the throughput and the response time in mobile and cloud environments .This project proposes a resource provisioning and scheduling strategy for scientific workflows on Infrastructure as a Service (IaaS) and Platform as services clouds (PaaS). This project presents an algorithm supported the Superior Element Multitude Optimization (SEMO), which aims to attenuate the general workflow execution cost while meeting deadline constraints. The main scope of the project is employed to research best available resource within the cloud environment depend on the entire execution time and total execution cost which is compare between one process to another process. If the provider satisfies the time least time, then the method becomes to termination. Python 3.6 is employed because the front language to develop the appliance .

Index terms – **Cloud Computing, Dominant Firefly Approach, Superior Element Multitude Optimization**.

****

I . INTRODUCTION

Many computing methodologies are available within the computing field for maximizing automation. Among those, M-Learning and Cloud Computing are considered to be the simplest service oriented computing technologies to automate tasks in virtual machines also on enable users to access

information very efficiently. Also, M-Learning offers cost-effective solutions for an honest range of services. Mobile learning and Cloud computing are two essential domains to elucidate distributed data sharing. In M-Learning, mobile devices employed 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 kinds of knowledge and applications to mobile device easily and accurately.

However, load balancing issues in Cloud data centers should be addressed to enhance performance and efficiency. during this paper, we propose a meta-heuristic algorithm to beat this load balancing issue. M-Learning technologies are deployed in many M-Learning systems and applications to enhance the training sorts of current students. on the average , M-Learning technologies enhance the training capacity of people by 70% [2]. a number of the Cloud computing services that would 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 wont to distribute incoming traffic across multiple servers to attenuate the delay of the Cloud server response to the Cloud users. Cloud load balancing is taken under consideration adequate as long because the throughput within the Cloud server is high, delays are minimal, and jitter is minimal while addressing Cloud user requests. Sometimes failure of load balancing within the Cloud results in poor image resolution and poor video streaming for users [3]. Thus, load balancing in Cloud servers is vital to maximise throughput and to understand superior performance in both public and private Clouds.

II. LITERATURE REVIEW

In 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:

A. LOAD BALANCING IN CLOUD COMPUTING

The concept of load balancing was first recognized as a crucial issue for computing within the year 2001. In [4], 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 [5], described the deficit round-robin load balancing technique for scheduling of tasks that was supported an efficient fair queuing load balancing technique. In publications by [6], it had been proposed that the Cloud scheduler might be supported an ant colony optimization method. Several studies and reviews are useful in understanding scheduling of tasks within the Cloud load balancer. In [7], they mentioned configuring Cloud storage services as an Infrastructure as a Service model, which was the key point for the foremost of the researchers who works on Cloud load balancing issues. In [8], the authors have surveyed many patterns of Cloud computing and also proposed the knowledge level framework for the event and inclusion in Cloud computing technologies. Authors in [9] have proposed partition-based techniques for load balancing in Cloud computing supported the switch mode mechanism. Also, they discussed a theory of games model of Cloud computing. Authors in [10] have suggested a dynamic load balancing technique through mobile agents for Peer to see (P2P) networks.

B. LIMITATIONS OF LOAD BALANCING

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. 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.

QOS Metrics Comparison of the Load Balancing Algorithm with Other Related Algorithm

LBA	QoSM1	QoSM2	QoSM3	QoSM4	QoSM5	QoSM6	QoSM7
LBA 1	Y	N	Y	N	N	N	N
LBA 2	Y	Y	Y	Y	N	N	N
LBA 3	Y	Y	Y	Y	Y	Y	N
LBA 4	Y	N	N	N	Y	N	N
LBA 5	Y	Y	Y	Y	N	Y	N
LBA 6	**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: Dominant Firefly Load balancing algorithm.

III. SYSTEM MODEL

In propose system along with 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. In this technique, that demonstrates the load balancing improvement on throughput and the response time of mobile devices and also using new technique Superior Element Multitude Optimization.

A. 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 [11] 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 [2], stated that M-Learning is a kind of E-learning which combines mobile technology and Cloud Computing wireless technology for a better learning experience.

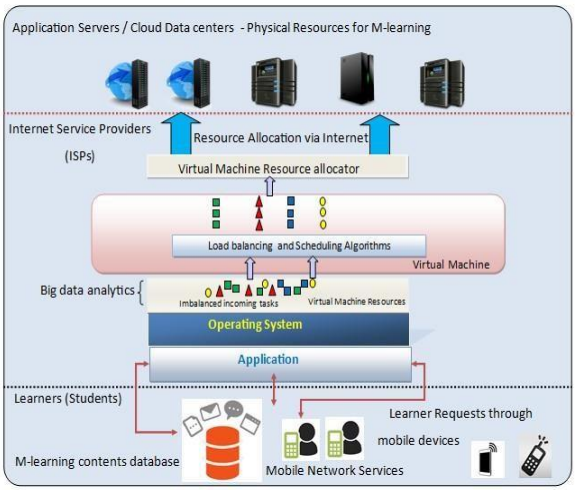


Fig l. M-Learning with Cloud Computing Architecture

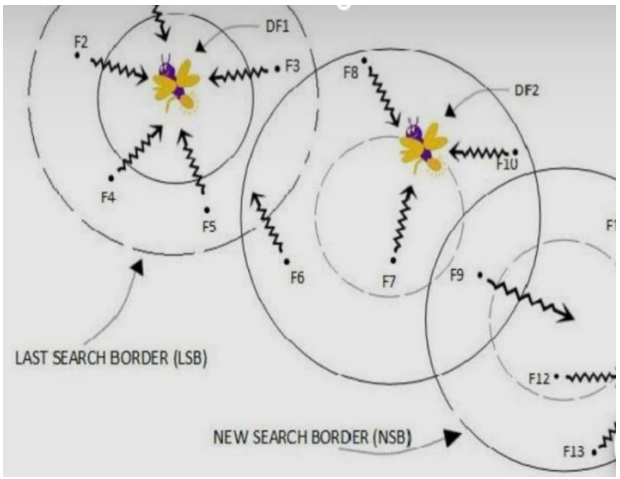


Fig 2. Dominant Firefly Approach

B. BIOLOGICAL MODEL OF DOMINANT FIREFLY BEHAVIOR

The firefly and its behavior for locating food sources and checking out partners are quite interesting. Fireflies that produce the most intense brightness are called dominant fireflies and other 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 F1, F2, F3……Fn be the submissive fireflies in a group of fireflies F.

Let DF1, DF2….DFn 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.

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

$\forall \sum F_n \in$ LSB flies to NSB (1)

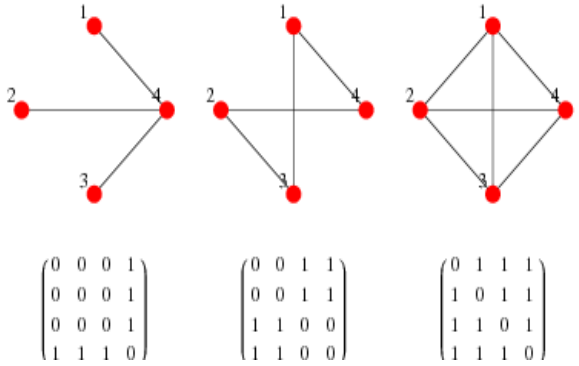
Else if $BV(DF_n) < BV(DF_{n-1})$ (2)

Then $\forall \sum F_n = DF_n$ in NSB (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)

$$=\sqrt{(q1-p1)2+(q2-p2)2+\cdots+(qn-pn)2}$$

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



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 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.

C. 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 tons of load (user requests), this must be balanced in such how 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 Cloud server. 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. represents task migration among multiple Cloud VMs**.**

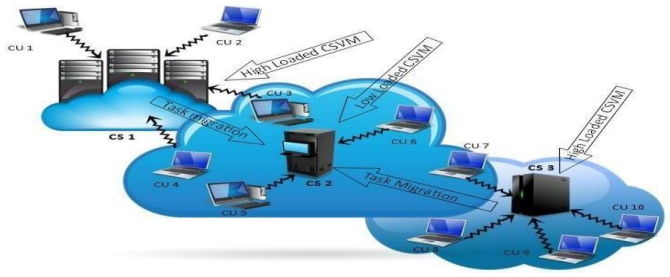


Fig 3. Task Migration among Multiple Cloud Server VMs Scenerio of Mobile Learning with Cloud Computing

IV. PROPOSED SYSTEM

To validate the dominant firefly search behavior strategy for balancing loads, here developed and tested our methods both in the Cloud and mobile environments. The results demonstrated were better when compared with load balancing methods. In the proposed Algorithm, along with 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. Here using a Superior Element Multitude Optimization.

A. SUPERIOR ELEMENT MULTITUDE OPTIMIZATION

The dissertation presented the algorithm named SEMO (Superior Element Multitude Optimization) which is compare the entire execution time and total execution cost between one processes to a different process. In addition, it extends the resource model to consider the data transfer cost between data in cloud environment so that nodes can be deployed on different regions. Also, it assigns different options for the choice of the initial resource pool. For example, for the given task, the various set of initial resource requirements is assigned. In addition, data transfer cost between data environment also are calculated so on minimize the value of execution in multi-cloud service provider environment.

B. EXECUTION TIME MATRIX GENERATION

This module generates the execution time matrix during which number of resources is taken as columns and tasks are taken as rows and therefore the time the tasks taken to finish in those resources are stored as values.

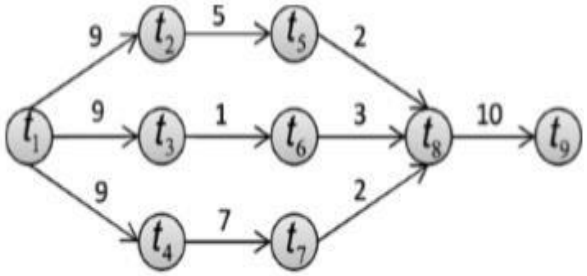


Fig 4. Execution Time Graph

C. TRANSFER TIME MATRIX GENERATION

This module generates the transfer time matrix during which number of taken are taken as columns and rows (square matrix is prepared) and therefore the time a task transfers the info to other task is stored as values. Therefore the diagonal elements are always zero since same task has no data transfer operation.

$$TEC = \sum_{i=1}^{|R|} C_{VM_{r_i}} * \left\lfloor \frac{(LET_{r_i} - LST_{r_i})}{\tau} \right\rfloor,$$

D. SCHEDULE GENERATION

Initially, the set of resources to lease R and the set of task to resource mappings M are empty and the total execution cost TEC and time TET are set to zero. After this, the algorithm estimates the execution time of every workflow task on every resource ri Rinitial. This is expressed as a matrix in which the rows represent the tasks, the columns represent the resources and the entry Exe Timei, j represent the time it takes to run task ti on resource rj. This time is calculated using Fig a. The next step is that the calculation of the info transfer time matrix. Such matrix is represented as a weighted adjacency matrix of the workflow DAG (Directed acyclic graph) where the entry Transfer Timei, j contains the time it takes to transfer the output data of task ti to task tj. This value is taken from database and is zero whenever ij or there's no directed edge connecting ti and tj. An example of those matrices is shown in Figure a) and b) below.

$$exeTime = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \end{array} \begin{array}{c} \begin{array}{ccc} r_1 & r_2 & r_3 \end{array} \\ \begin{bmatrix} 2 & 1 & 4 \\ 4 & 3 & 6 \\ 10 & 6 & 15 \\ 7 & 4 & 12 \\ 8 & 4 & 10 \\ 3 & 2 & 7 \\ 12 & 7 & 18 \\ 9 & 5 & 20 \\ 13 & 8 & 19 \end{bmatrix} \end{array}$$

(a)

Matrix Representation of Execution Time

$$transferTime = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 \end{array} \\ \begin{bmatrix} 0 & 9 & 9 & 9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

(b)

Matrix Representation of Transfer Time

V. CONCLUSION

In this work, the proposed SEMO technique behavior execution model was applied and simulated in CSVM instances to enhance load balancing of tasks within the Cloud computing environment. This approach helps to balance the load in multiple CSVMs by increasing QoS metrics like throughput and reaction time. Also, in our methods, the load of job requests from Cloud end-users submitted to CSVMs is optimally balanced to extend the efficiency of the Cloud server. The proposed algorithm was compared with other load balancing algorithms in dominant firefly approach. The results demonstrated an improvement in energy consumption among Cloud servers. These findings might be extended to cost computational methods to utilize maximum CPU that might increase server efficiency. The most objective of the proposed model is to reinforce M-Learning environments by finding many relational models to avoid the very best energy consuming server throughout the planet within the current real-time Cloud environment scenario, the throughput efficiency at the Cloud data center is an important factor and lots of researchers are showing keen interest in developing various algorithms for it. Also, there are many opportunities within the field of M-Learning, Cloud computing, and in Cloud-based organizations. This work reveals many challenges in M-Learning using Cloud computing technologies.

REFERENCES

[1] Kaushik Sekaran, Mohammed S.Khan, Rizwan Patan, Amir H.Gandomi, Venkata Krishna Parimala, Suresh Kailam (2019). Improving the Response Time of M-Learning and Cloud Computing Environment Using a Dominant Firefly Approach, DOI IEEE Access- 2896253.

[2] 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.

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] TsvetozarGeorgiev, Evgenia Georgieva, Angel Smrikarov, "MLearning-a New Stage of E-Learning", International Conference on Computer Systems and Technologies-CompSysTech'04, 2004.