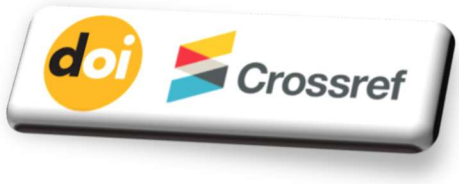
Journal, Advances in Mathematical & Computational Sciences
Vol. 9 No. 1, 2021
www.mathematics-computationaljournal.info



Journal of Advances in Mathematical & Computational Sciences
An International Pan-African Multidisciplinary Journal of the SMART Research Group
International Centre for IT & Development (ICITD) USA

Available online at https://www.isteams.net/ mathematics-computationaljournal.info
DOI: dx.doi.org/10.22624/AIMS/MATHS/V9N1P6
CrossREF Member Listing - https://www.crossref.org/06members/50go-live.html

A Throughput Maximization Based Load Balancing Technique for Big-data Cloud Systems

[1]Oduwole, O. A., [2]Akinboro S. A., [3]Lala, O. G., [4]Oladoye S. F. & [5]Olabiyisi S. O.
[1,3]Adeleke University Ede, Osun State. Nigeria.
[2]University of Lagos, Akoka Lagos, Nigeria.
[4] Osun State Polytechnic Iree, Osun State. Nigeria.
[5]Ladoke Akintola University of Technology, Ogbomoso, Oyo State. Nigeria.
E-mail: dayooduus@yahoo.com
Phone: +2348034408158

ABSTRACT

Cloud computing load balancing techniques need to be further improved. This is due to the immense and disperse nature of big-data cloud computing environments' tasks. This study developed an improved load balancing technique for big data cloud computing environments using the combination of a Throughput Maximization Model with Particle Swarm Optimization (PSO) and Firefly algorithms. The developed Throughput Maximization Based Balancer (TM-BAL) was designed to admit tasks at regional data centers using the throughput maximization model. The admitted tasks are either allocated servers within the regional data centers using PSO model or moved to the central load balancer (Firefly model) which handles task distribution among all regional data centers. The developed technique was simulated using MATLAB R2018 software and was benchmarked with PSO and Firefly models using throughput and response time as performance metrics. Results showed that the developed technique had the highest average throughput value and the least average response time of the three models. This implies that the developed technique outperformed PSO and Firefly algorithms in terms of throughput and response time, which is necessary for effective resource utilization as more tasks are processed within a short period of time. Hence, the development of the TM-BAL is justified by significant improvements in response time and network throughput

Keywords: Big-data, cloud computing, environment, load-balancing, maximization, technique, model

Oduwole, O. A., Akinboro S. A., , Lala, O. G, Oladoye, S.F. & Olabiyisi S. (2021): A Throughput Maximization Based Load Balancing Technique for Big-data Cloud Systems. Journal of Advances in Mathematical & Computational Science. Vol. 9, No. 1. Pp 57-70. DOI: dx.doi.org/10.22624/AIMS/MATHS/V9N1P6
Available online at www.isteams.net/mathematics-computationaljournal.

1. INTRODUCTION

Cloud computing is an internet based network technology that shares improvements in communication technology by providing online computing services to clients with a variety of needs. It provides pay-as-you-go hardware and software, as well as software development frameworks and tools for testing resources [1].

Such a delivery of resources is achieved with the aid of the three cloud service models which are Infrastructure as a service (IaaS) which provides hardware and software applications for customers, Software as a service (SaaS) which provides customers with software development frameworks, and Platform as a Service (PaaS) which makes available to customers, tools for developing and testing their applications [2].

The fundamental need for cloud computing is the sharing and availability of computational resources (demand-based virtual machines). The cloud offers numerous resource infrastructures, networks, and services that do not only save valuable time and expenses but are also secure, versatile, and scalable [3]. In cloud computing, a suitable virtual machine is allocated based on a request from the user. This allocation is accomplished using load balancing strategies involving task allocation and Virtual Machine (VM) / task migration [4]. However, despite the promising future of cloud computing, one of the key disadvantages requiring attention is the full realization of load balancing [5]. Load balancing problem is simply the distribution of loads to the processing elements. In an environment with multiple nodes, there is a high probability of having some nodes overloaded while others may be idle [6]. Load balancing is the process of redistributing workload in a distributed system, such as cloud computing, so that no computing machine is overloaded, under-loaded, or idle [7, 8]. Load balancing enhances performance of cloud by attempting to speed up various constrained parameters such as response time, execution time, system stability and task migration [9, 10]. Researchers have suggested a large range of load balancing techniques in cloud and network environments such as in [11 – 17] with emphasis mainly on task scheduling, task allocation, resource scheduling, resource allocation and resource management.

Big data is an immense amount of structured, unstructured and semi-structured data [18]. It continues to expand unaccountably with its data size increasing from exabytes to zettabytes, For instance, Facebook handles images of 40 billion users a day [19]. Each hour, Walmart processes one million customer transactions. All of these, with memory size of about 2.5 petabytes are stored in cloud storage. In addition, Instagram receives 65,000 billion images per minute, and YouTube receives 500 hours of video per minute. Every day, Google conducts one billion searches and sends 294 billion emails. Cell phone and Internet of Things (IOT) users are expected to reach 6.1 billion and 2.6 billion in number, respectively, by the year 2020 [20].

The performance of big data cloud computing needs to be improved to cope with huge transmission volume, high transmission frequency, hard transmission deadline and high processing cost that characterize big-data computing systems [21, 20]. There is also the need to optimize the architecture of big-data cloud in order to achieve optimum load distribution [22]. While these challenges exist for big-data cloud load balancing, they can be overcome by implementing efficient strategies. A Central-Regional(Distributed) solution is proposed in this study. This is accomplished by first admitting tasks to regional data centers using a throughput maximization model. Admitted tasks with the same source and destination regions are distributed to servers within the region using PSO, while those with different source and destination regions, and those whose capacity cannot be accommodated by regional servers, are migrated to the central data control center for distribution to servers in available regions using the Firefly algorithm.

2. METHODOLOGY

Considering a cloud network having a set of data centres, $\mathcal{C} = \{C_1, C_2, \dots, C_n, \dots\}$, and each data centre C_i has a number of nodes. A node denotes a physical machine (or server), and the set of nodes belonging to a data centre C_i is given as $N_i = \{p_1, p_2, \dots, p_k, \dots\}$.

Assuming that the network receives tasks requests of different data rates and delay requirements at random, the objective function of the network's throughput maximization problem is formulated as:

$$\max \sum_{c_i \in C} \quad \sum_{p_k \in N_i} \quad \sum_{t_j \in T} (x_{jk} r_j) \tag{1}$$

subject to;

$$x_{jk} \in \{0,1\}, \qquad \forall\, c_i, t_j \in \tau, p_k \tag{1a}$$

$$\sum_{t_j \in \tau} (x_{jk} r_j) \leq r|N_i| \qquad \forall\, c_i, i \leq k \leq |N_i| \tag{1b}$$

$$x_{ji}\, d(t_j, C_i) \leq L_j \qquad \forall C_i, t_j \in \tau \tag{1c}$$

Where

The x_{jk} in Constraint (1a) is the indicator variable which takes either value 0 or 1. For instance, x_{01} implies that task j is not assigned to node k. Constraint (1b) implies that the sum packet rates, r, of all tasks to be processed does not exceed the capacity of all the available nodes N_i belonging to the data centre C_i. . Constraints (1c) implies that the latency experienced by task j being served by data centre C_i does not exceed the delay requirement L_j of task j. The set of tasks that the network is requested to serve is represented by: $T = \{t_1, t_2, \ldots, t_j, \ldots\}$. In this set, only the tasks that satisfy the objective function or constraints are admitted into the network. Every task t_j is assumed to originate from a given region, and each region represents a users' base. Also, every data centre C_i is located in only one region. In other words, each data centre (DC) is identified by the region it is located within the network, and a region can have one or more DCs.

A given task request can only be admitted into the network via its regional DC which consists of a number of PMs and VMs, and utilizes the proposed throughput maximization objective in the admission process. The implementation of this procedure is expressed in Algorithm 1. The architecture and flowchart of the proposed TM-BAL are illustrated in Figures 1 and 2 respectively. The model performs load balancing at two levels. The first level (Regional) load balancing

Algorithm 1: An algorithm for the throughput maximization procedure
Inputs: N_i //A set of nodes/servers available in a data centre C_i
T //A set of tasks requesting to be served by the network
Output: $newT$ //A set of admitted tasks that maximize throughput

BEGIN
Obtain the minimum packet rate in T as r_{min}// The least sending rate of task set T
Initialize the empty set $newT$
Admit the first arrived task t_1 into the new set $newT$//the first task in the set T is automatically admitted to newT
Initialize sum packet rate as $r = r_j$//Sum packet rate is the accumulated rate of the admitted tasks
For each task (t_j) in T Do //Start of the steps to be repeated for each of the subsequent tasks to be admitted or rejected
Get packet rate, r_j, of task t_j
Get delay requirement, L_j, of task t_j
Calculate current sum packet rate as $r = r + r_j$ // the packet rate of the current task is added to the total

packet rate of previously admitted tasks.

If difference of new r & previous r is greater than r_{min} **Then** // the difference must be greater than r_{min} for task t_j to increase the throughput significantly, and for the task to be admitted.

Calculate delay $d(t_j, C_i)$ for the task t_j to be processed by the DC C_i

If r_j satisfies Constraint (2) and $d(t_j, C_i)$ satisfies Constraint (3) **Then**

//If the packet rate of task j to be admitted is less or equal to the available capacity of the servers at the Data Center (constraint 2) and that the calculated delay does not exceed the delay requirement (constraint 3)

Admit t_j into newT // then task t_j is admitted having met the admission criteria

Else

Reject t_j // task t_j is not admitted for processing

End If

End If

End For

END

is performed in a distributed manner by each regional DC; while the second level (Central) load balancing is performed in a centralized manner over the entire DCs by a DC controller.

Each DC accepts task requests that satisfy the throughput maximization target in their respective regions at the start of the scheme. The admitted tasks are then sorted into two groups: Group A and Group B. A task is assigned

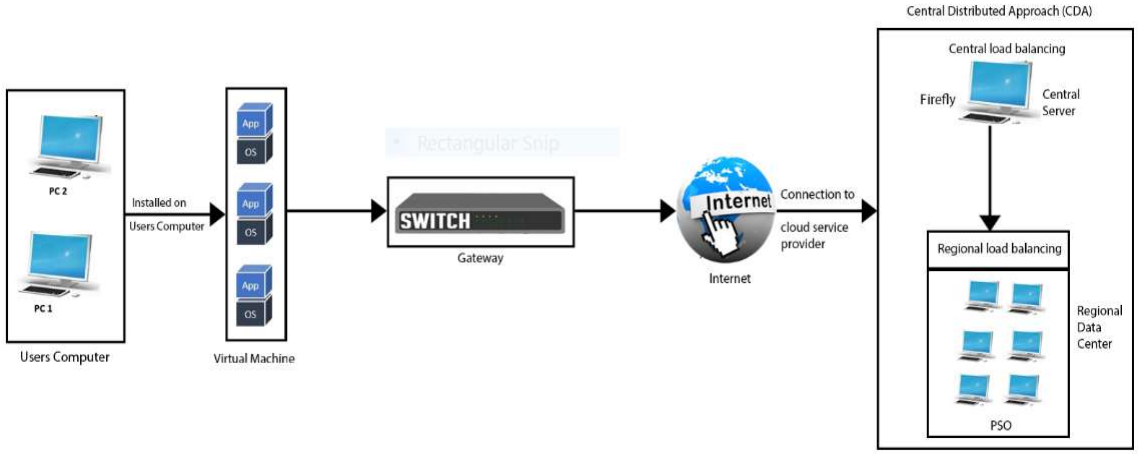


Figure 1: TM-BAL Architecture

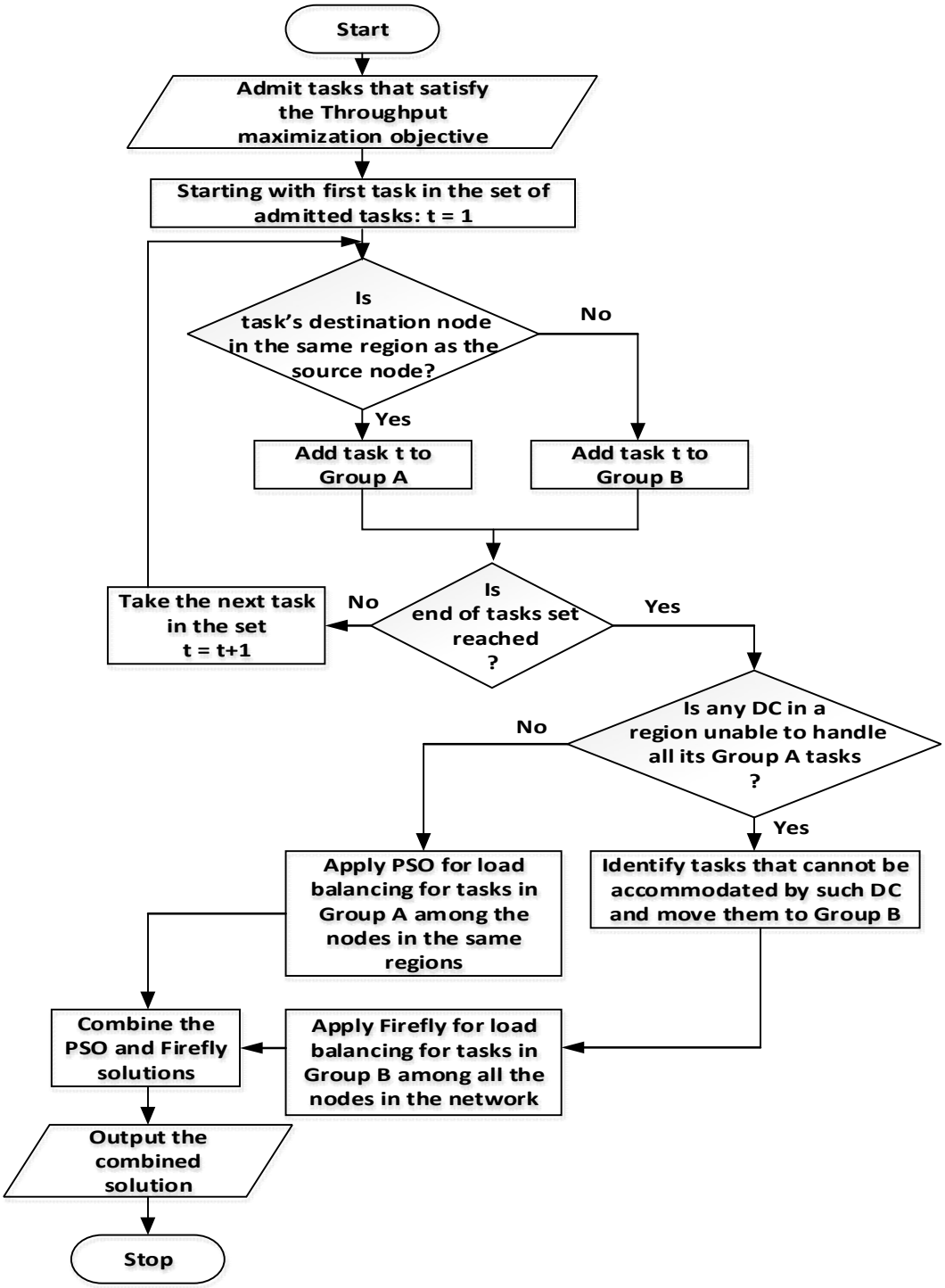


Figure 2: Flowchart of the TM-BAL

To Group A if the source and destination nodes are in the same region; otherwise, it is assigned to Group B. Algorithm 1 depicts the throughput maximization procedure. The Group A tasks are initially assigned to available nodes/servers at their respective DCs, and then the PSO algorithm is used to achieve regional load balancing across the nodes. All of the tasks in Group B, which are composed of tasks with different source and destination node regions, and those whose capacity cannot be accommodated by regional servers are forwarded to the DC controller for server allocation using Firefly algorithm.

The PSO algorithm was selected for regional level load balancing due to its ability to find optimal solution quickly by obtaining its global best solution from local best solutions [15, 23]. The firefly was applied to balance load at the central level due to its high rate of processing tasks [20, 24]. The flowcharts of the regional load balancer(PSO) and central load balancer(Firefly) are represented in Figures 3 and 4.

The proposed technique, the PSO and Firefly algorithms were implemented using MATLAB R2018 version software. The simulation parameters and specifications used were adopted from the Amazon's Elastic Compute Cloud EC2 [25], and were randomly drawn from the following categories:

CPU = [2500, 1000, 500]; % MIPS (million instructions per second)
RAM = [870, 1740, 613]; % MB;
VM price = [0.200, 0.047, 0.020]; % # currency

The performance evaluation of the proposed technique was performed by comparing the simulation results of its implementation with that of the PSO and Firefly techniques using throughput and response time as metrics. Throughput refers to the sending and receiving rates of data for the total number of completed tasks on a given input at a given time unit. A high throughput rate is required for improved cloud system performance [26].

$$\text{Throughput} = \frac{sum(SP)\times(P_{av})}{Total\ processing\ time} \quad (2)$$

where SP is the number of successful packets, P_{av} is the average packet size.

Response time is defined as the time elapsed between sending a request and receiving its response. It should be minimized in order to improve overall system performance [27].

$$\text{Response time} = \text{T}_{arrival} - \text{T}_{completion} \quad (3)$$

where $\text{T}_{arrival}$ denotes the time the task arrives the network, and $\text{T}_{completion}$ denotes the time the processing of the task is completed.

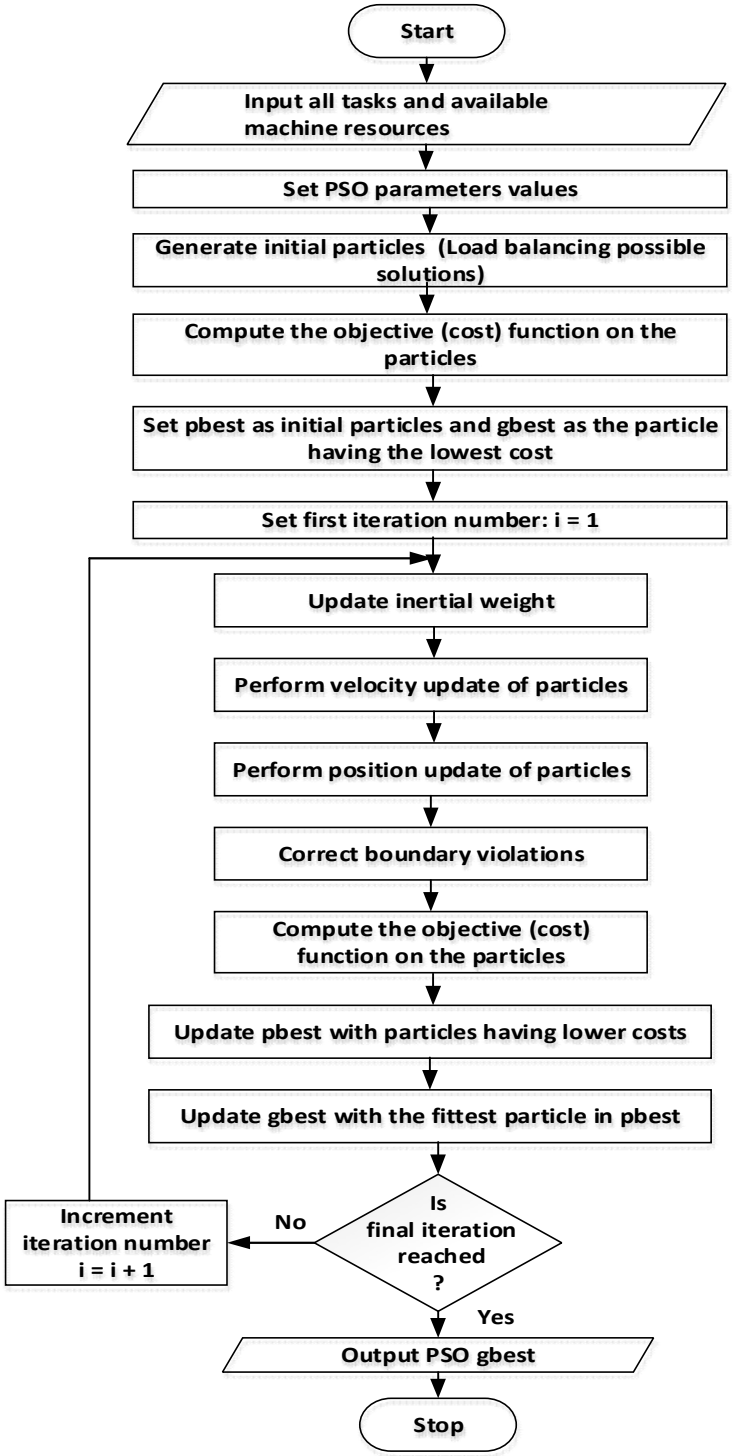


Figure 3: Flowchart of the PSO Load Balancing Model

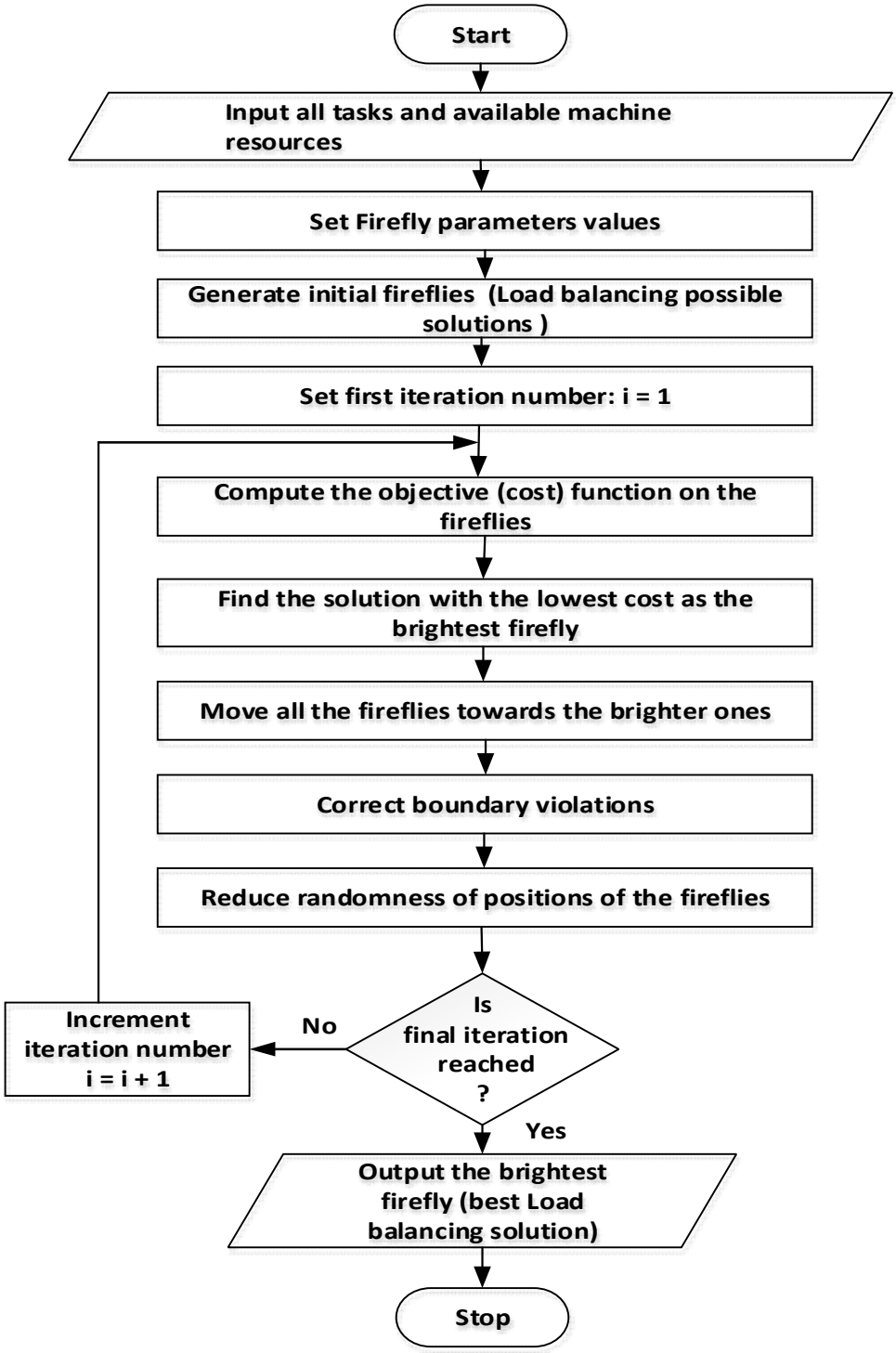


Figure 4: Flowchart of the Firefly Load Balancing Model

3. RESULTS AND DISCUSSION

The simulation results of the performance of the developed TM-BAL in comparison with PSO and Firefly algorithms using the metrics; throughput and response time, are presented in Tables 1, Figure 5, and Table 2, Figures 6 respectively.

The throughput values of the developed big-data cloud load balancing technique, as well as the PSO and Firefly algorithms, are shown in Table 1 and Figure 5. According to the results, the average network throughput for the TM-BAL, PSO, and Firefly algorithms are 1665651.988, 1011958.298, and 1057459.365, respectively.

The proposed technique had the highest average throughput, followed by the Firefly algorithm, while PSO had the lowest. The throughput of a communication link is defined as the average data rate of successful message delivery. This implies that of the three algorithms, the proposed technique had the highest task processing rate.

The response times of the proposed load balancing technique, as well as the PSO and Firefly algorithms, under the same condition as the previous experiment are shown in Table 2 and Figure 6. The average response times for the proposed technique, PSO, and Firefly algorithms are 0.022547465s, 0.3845324s, and 0.39847615s, respectively. The results show that the proposed technique experienced the shortest delay when processing tasks.

4. CONCLUSION

In this study, a Throughput Maximization-based Cloud Computing Load Balancer named TM-BAL for big-data cloud environments was developed using a throughput maximization model combined with Particle Swarm Optimization (PSO) and Firefly optimization algorithms. Response time and throughput were used to assess the performance of the developed technique and compare it to two state-of-the-art optimization algorithms, PSO and Firefly, which were analyzed and simulated.

The TM-BAL algorithm was discovered to outperform the two existing algorithms. The results indicate that the development of TM-BAL was justified due to significant improvements in response time and network throughput.

Table 1: Comparison of Throughput of the TM-BAL with PSO and Firefly Algorithms

No. of Requests	PSO	Firefly	Proposed
200	2327620.372	3421667.479	3770636.016
400	2031317.375	2708055.683	3817528.864
600	1724831.666	1675117.344	1742996.646
800	1510751.301	1578430.874	2357280.985
1000	1603787.855	1631969.594	2523890.751
1200	1319016.496	1156192.677	1876247.032
1400	1209129.331	1131134.312	2067169.728
1600	1036491.791	896167.1558	1690761.907
1800	985081.3961	988210.7485	1676983.611
2000	858512.5897	756002.5289	1562005.019
2200	823478.7738	743933.8985	1382555.644
2400	704135.4053	726816.9131	1324711.72
2600	586286.3891	568767.5294	1147788.871
2800	526859.0639	485427.3223	1024640.379
3000	647063.3185	583344.0135	1084183.827
3200	511990.0657	471281.4265	982999.9268
3400	447551.0485	380978.0821	742479.4549
3600	495520.5902	421183.0703	828133.9925
3800	478881.3365	429951.3385	920111.2212
4000	410858.7889	394555.3175	789934.1739
AVERAGE	1011958.248	1057459.365	1665651.988

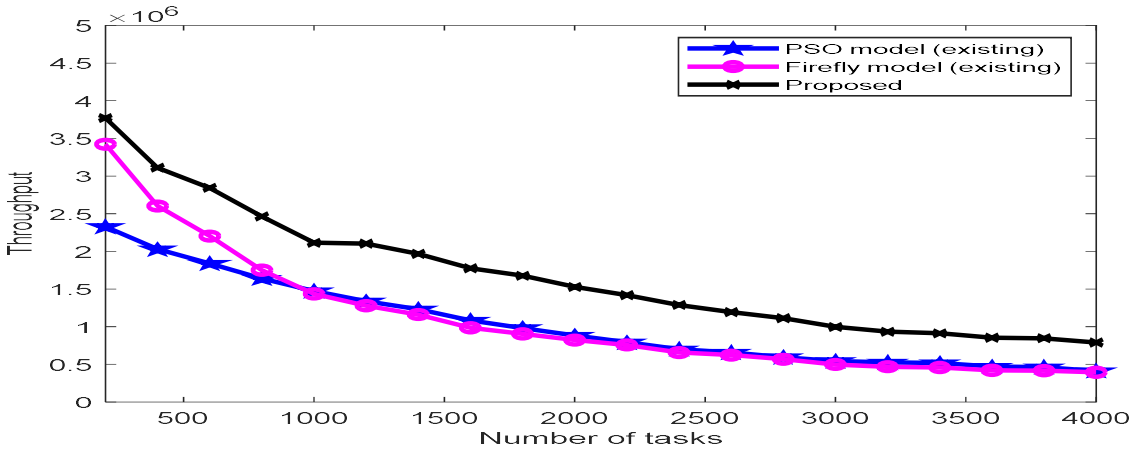


Figure 5.: Comparison of TM-BAL with PSO and Firefly Algorithms

Table 2: Comparison of Response Time of TM-BAL with PSO and Firefly Algorithms

No. of Requests	PSO	Firefly	Proposed
200	0.066738	0.076503	0.069279
400	0.10129	0.10584	0.078304
600	0.13605	0.15843	0.13787
800	0.17661	0.19103	0.11089
1000	0.19374	0.17935	0.11937
1200	0.22275	0.21948	0.13836
1400	0.25503	0.25461	0.14948
1600	0.28855	0.29015	0.20789
1800	0.33081	0.32006	0.18961
2000	0.36107	0.34913	0.19525
2200	0.38064	0.38299	0.21151
2400	0.41723	0.40441	0.23365
2600	0.4434	0.44691	0.24282
2800	0.52318	0.49288	0.27673
3000	0.49979	0.52902	0.29808
3200	0.59333	0.64799	0.30783
3400	0.6094	0.69214	0.34672
3600	0.64082	0.70971	0.37809
3800	0.70808	0.74845	0.41226
4000	0.74214	0.77044	0.4055
AVERAGE	0.3845324	0.39847615	0.22547465

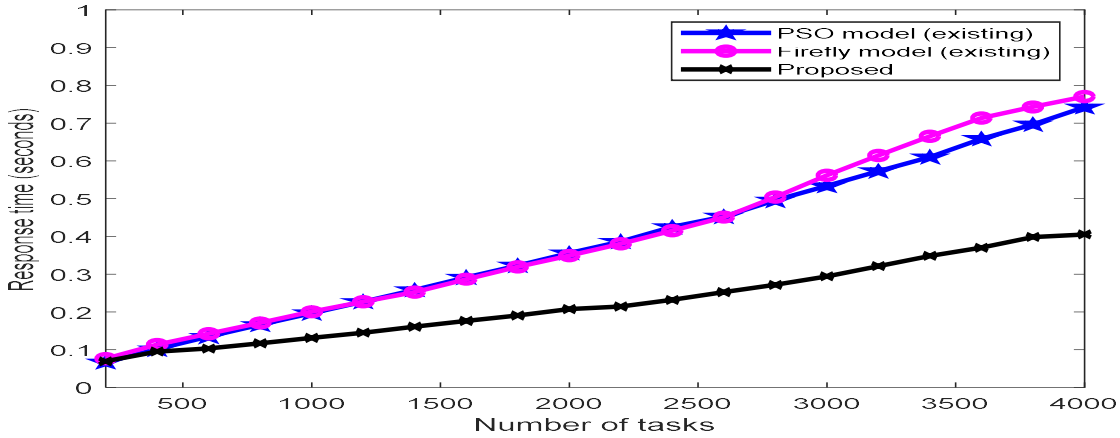


Figure 6.: Comparison of Response Time of TM-BAL with PSO and Firefly Algorithms

5. DIRECTION FOR FUTURE WORK

More research can be done to allocate cloud computing resources based on predictions of the types of services that will be provided at a given time. The study could be expanded to look at how to improve other performance metrics like task rejection ratio, algorithm complexity, and CPU utilization rate.

REFERENCES

[1] Birje, M. N., Challagidad, P. S. Goudar, R. H. Tapale, M. T. (2017). Cloud computing review: concepts, technology, challenges and security. *Int. J. Cloud Computing,* Vol. 6, No. 1, pp. 32 – 53.

[2] Dar, A. & Ravindran, A. (2018). A Comprehensive Study on Cloud Computing Paradigm. *International Journal of Advance Research in Science and Engineering,* 7(4), 235-242.

[3] *Soni, S. & Sangwa, S.* (2018). Load Balancing in Cloud Computing: A Review. *International Journal of Advanced Research in Computer Science,* 9 (2),760-763.

[4] Mishra, S. K., Sahoo, B., & Parida, P. P. (2018). Load balancing in cloud computing: a big picture. *J King Saud Univ Comp Infor Sci:* pp. 1–32.

[5] Ejimogua, O. H. & Basaran, S. (2017). A systematic mapping study on soft computing techniques to cloud environment. *Science Direct,* 120(45), 31-38.

[6] Afzal, S. & Kavitha, G. (2019). Load Balancing in Cloud Computing – A Hierarchical Taxonomical Classification. *Journal of Cloud Computing: Advances, Systems and Applications.* 8(22), pp. 1-24.

[7] Achar R., Thilagam, P. S., Soans, N. Vikyath, P. V., Rao, S., Vijeth, A. M. (2013). Load balancing in cloud based on live migration of virtual machines. In: 2013 Annual IEEE India Conference (INDICON), pp. 1–5.

[8] Magalhaes, D., Calheiros, R. N., Buyya, R., & Gomes, D. G. (2015). Workload modelling for resource usage analysis and simulation in cloud computing. *Comp Elect Eng*, 47:69–81; https:// DOI:10.1016/j.compeleceng.2015.08.016

[9] Dam, S., Mandal, G., Dasgupta, K,, & Dutta P. (2015). Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing. *In;* Proceedings of the third international conference on computer, communication, control and information technology (C3IT), pp 1–7.

[10] Dave, A, Patel, B., & Bhatt, G. (2016). Load balancing in cloud computing using optimization techniques: a study. *In: International Conference on Communication and Electronics Systems (ICCES),* pp 1–6.

[11] Akinboro, S. A. & Ayobami, T. O. (2019). A Model for Self-Adaptive Routing Optimization in Mobile Ad-hoc Network. *International Journal of Swarm Intelligence Research*, 10(1), pp. 58-74.

[12] Mohammed, A. & MdAsri, N. (2016). Hybrid Symbolic Organisms Search Optimization Algorithm for Scheduling of Tasks on Cloud Computing Environment. *Plos One,* 10(13), pp. 1-29.

[13] Li, J., Ma, T., Tang, M., Sheu, W., & Jin, Y. (2017). Improved FIFO Scheduling Algorithm Based on Fuzzy Clustering in Cloud Computing. *Information,* 8(5), 422 – 447.

[14] Hussain, D., Omar, M., Ahmad, K., Jain, V. & Wason, R. (2019): Load status evaluation for Load Balancing in Distributed Database Servers. *3C Tecnologia. Glosas de innovacion aplicadas a la pyme,* pp. 422 – 447.

[15] Miglani, N. & Sharma, G. (2019). Modified Particle Swarm Optimization based upon Task categorization in Cloud Environment. *International Journal of Engineering and Advanced Technology (IJEAT)*, 8(4C), 67 – 72.

[16] Aswini, J., Malarvizhi, N. & Kumanan, T. (2019). A Novel Firefly algorithm based Load Balancing approach for Cloud Computing. *International Journal of Innovative Technology and Exploring Engineering*(IJITEE). ISSN-2278-3075. 8(2), pp. 91-96.

[17] Manaseer, S., Alzghoul, M. & Mohmad, M. (2019). An Advanced Algorithm for Load Balancing in Cloud Computing using MEMA Technique. *International Journal of Innovative Technology and Exploring Engineering*(IJITEE). ISSN-2278-3075. 8(3), pp. 36-41.

[18] Sekar, K. & Padmavathamma, M. (2016). Comparative study of encryption algorithm over big data in cloud systems. *In Proceeding: 3rd International Conference on Computing for Sustainable Global Development (INDIACom),* New Delhi, pp. 1571-1574.

[19] Sivarajah, U., Kamal, M. M., Irani, Z. & Weerakkody, V. (2017). Critical analysis of Big Data challenges and analytical methods. *Journal of Business Research*. 70; 263-286.

[20] Jayaraj, T. & Abdul Samath, J. (2019). Process Optimization of Big-Data Cloud Centre Using Nature Inspired Firefly Algorithm and K-Means Clustering. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 8(12), 48-52.

[21] Jelassi, M., Ghazel, C., & Saïdane, L. A. (2017). A survey on quality of service in cloud computing. *3rd International Conference on Frontiers of Signal Processing (ICFSP)*, *Paris, 2017*, pp. 63-67; https://doi: 10.1109/ICFSP.2017.8097142.

[22] Castelino, C., Gandhi, D., Narula, H. G., & Chokshi, N. H. (2014). Integration of Big Data and Cloud Computing. International Journal of Engineering Trends and Technology (IJETT), 100-102.

[23] Devi, M. & Ryhmend, U. (2014). Particle swarm optimization based node and link lifetime prediction algorithm for route recovery in MANET. *Journal of Wireless Communication and Tech,*107(1),1–10.

[24] Kumar, D. S., Dharma, E. G., & Raj, P. (2018). A literature review on load balancing mechanisms in cloud computing *International Journal of Advanced Research in Computer Science*, 9(1), 432-435.

[25] Alkayal, E. S., Jennings, N. R. & Abulkhair, M. F. (2016). Efficient Task scheduling Multi- Objective Particle Swarm Optimization in Cloud Computing. IEEE 41st Conference on Local Computer Networks Workshops.

[26] Patel, N. & Chauhan, S. (2015). A survey on load balancing and scheduling in cloud computing. *Int J Innov Res Sci Technol.* 1(7), 185–189.

[27] Roy, S. Hossain, M. A. Sen, S. K. Hossain, N. & Al Asif, M. R (2019). Measuring the Performance on Load Balancing Algorithms. *Global Journal of Computer Science & Technology,* Volume 19 Issue 1 Version 1.0.

APPENDIX

Definition of Terms & Acronyms

1. **Algorithm:** An algorithm is a set of instructions that tells you how to solve a specific problem.
2. **Cloud:** This is a pool of computing resources that can be accessed by clients over the internet on pay as you go basis.
3. **Cloud computing:** This refers to computing operations that take place on computers that are connected to the internet.
4. **Complexity of algorithm:** is the amount of time (number of steps) and memory required to perform a computation specified by the algorithm.
5. **Computer network:** is set of computers connected together for the purpose of sharing resources such as internet, intranet, Local Area Network, Wide Area Network.
6. **CPU:** Central Processing Unit
7. **Data Center (DC):** A data center is a physical facility where enterprises store their critical applications and data.
8. **Distributed:** Spread across multiple computers.
9. **Flowchart:** a flowchart is a diagram made up of symbols and words that completely describes an algorithm (i.e., how to solve a problem).
10. **IaaS:** Infrastructure as a Service
11. **Model**: is a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process.
12. **PaaS:** Platform as a Service
13. **PSO:** Particle Swarm Optimization
14. **SaaS:** Software as a service
15. **Server:** A server is a computer that offers services to another computer (referred to as the client).
16. **Technique:** a method of carrying out a specific task, particularly the execution or performance of a creative work or a scientific procedure.
17. **TM-BAL:** Throughput Maximization Based Load Balancer
18. **Virtual machine (VM):** is a computer that does not exist in the physical realm but is simulated by another computer.
19. **RAM:** Random Access Memory
20. **Simulation:** A computer simulation is a computer program that performs a step-by-step representation of the behavior of a system in real life.