International Journal of Electrical and Computer Engineering (IJECE)
Vol. 12, No. 3, June 2022, pp. 3217~3225
ISSN: 2088-8708, DOI: 10.11591/ijece.v12i3.pp3217-3225

A load balancing strategy for reducing data loss risk on cloud using remodified throttled algorithm

Fatema Tuj Johora[1,2], Iftakher Ahmed[1,3], Md. Ashiqul Islam Shajal[1,3], Rony Chowdhory[1,3]
[1]Department of Computer Science and Engineering, Faculty of Science and Engineering, Daffodil International University, Dhaka, Bangladesh
[2]Institute of Information Technology, Jahangirnagar University, Dhaka, Bangladesh
[3]Department of Computer Science and Engineering, Faculty of Mathematical and Physical Sciences, Jahangirnagar University, Dhaka, Bangladesh

Article Info

Article history:

Received May 14, 2021
Revised Dec 17, 2021
Accepted Jan 10, 2022

Keywords:

Cloud computing
Load balancing
Resource utilization
Response time
Time utilization

ABSTRACT

Cloud computing always deals with new problems to fulfill the demand of the challenging organizations around the whole world. Reducing response time without the risk of data loss is a very critical issue for the user requests on cloud computing. Load balancing ensures quick response of virtual machine (VM), proper usage of VMs, throughput, and minimal cost of VMs. This paper introduces a re-modified throttled algorithm (RTMA) that reduces the risk of data hampering and data loss considering the availability of VM which increases system's performance. Response time of virtual machines have been considered in our work, so that when migration process is running, data will not be overflowed in the VMs. Thus, the data migration process becomes high and reliable. We have completed the overall simulation of our proposed algorithm on the cloud analyst tool and successfully reduced the risk of data loss as well as maintains the response time.

Corresponding Author:

Fatema Tuj Johora
Department of Computer Science and Engineering, Daffodil International University
102/1, Sukrabad Mirpur Rd, Dhaka 1207, Bangladesh
Email: fatema.cse@diu.edu.bd

1. INTRODUCTION

Cloud computing is the on-demand topic of modern science. A grate number of computing resources including physical and virtual resources are shared across the network and on demand applications are served according to the request of user. Cloud can be public, private and hybrid and each type have four layers for providing different services according to the needs of user on request by 24/7. In each layer a number of challenges are associated with each layer like security, cost management and containment, managing distributed clouds, performance, load balancing, resource allocation and so on.

Individual user, many small and large organization, important sectors are getting attached with the cloud for its overall performance, and load balancing is the most critical part for improving performance. Load balancing performs its execution by balancing the loads into different computing resources and datacenters. An important goal of load balancing is making quick response time, processing time, cost and maximize overall performance. Previous algorithms deal with the response time, and processing time. Without considering the present resources risk at the virtual machine. Resources risk while task migration, such as data loss, and data hampering, should need to be considered which was absent in the previous algorithms.

Our proposed algorithm is used to focus on improved performance of the cloud computing services and named as re-modified throttled algorithm (RMTA). RMTA is mainly concerned in reducing the risk of resources hampering or loss of the data that may occur in the migration process. Re-modified Throttled Algorithm executes its operation under a data center controller (DCC). In the previous throttled modified algorithm (TMA) [1] when a client wants to use a virtual machine (VM) for storing his information he sends a request to the controller of data center. Then DCC assign an idle VM to the user's request without considering the loads/available free space of that VM. As a result, when the migration process starts if the request load size is more than the available space of the VM then the data overflow. This can cause data loss or data hampering. At the same time when this situation occurs the response time also increases.

The RMTA algorithm deals with the above situation. It contains two index table where the VMs are being allocated according to their availability or unavailability and most importantly a present load controller which considers the present load of the VMs. Initially, all VMs are stored in "Available Index" chart. When a client sends a request to the DCC first check the "Available Index" chart. After then it also detects the available size of the VM and compare with the size of user's request so that the data might not overflow while the migration process is running. If the VM satisfies the request with availability, then load balancer (RMTA) forwards VM's Identity to the DCC. Then the DCC starts migration by sending a migration request to that particular VM. When the migration starts that particular VM's id is listed to the "Unavailable Index" table. After completing the migration that VMs id will again be listed in the available index table. By this process the risk of data overflow, data loss, data hampering will be reduced while migration as well the quick response time for the above situation. We have completed our simulation process in the cloudsim analyst tool. The overall performance of our proposed algorithm (RMTA) has been analyzed based on the simulation results.

2. RELATED WORKS

This section will give a clear concept about the Load Balancing Algorithms already exists and used recently. Vibhore Tyagi *et al.* [1] has proposed Throttled Algorithm that reduces the processing and response time of the server. They demand that service broker policy can have quick response. This algorithm identifies the work so that we can find the applicable and available virtual machine for performing individual job.

A modified algorithm called TMA has been presented by Phi *et al.* [2]. The objectives of their algorithm are cutting down the latency and processing time based on the previous throttled algorithm. They have used use two different index tables for their works.

Taking job scheduling policy as a basis Li *et al.* [3] proposed ant colony optimization algorithm. The main function of this algorithm is ensuring proper resource allocation in a dynamic and complex network. Their proposed work can select acceptable resource allocation for performing job with the given size of cloud environment.

Soft computing methods with the help of genetic algorithm [4] has been proposed by another research group. They have used uniform resource alternative for their work. By using this technique, it is easy to control a vast space that can be applicable to complex objective function which removes the chances of being trapped into any certain solution for a specific neighborhood. This technique also assures quality of service (QoS) requirements for customer request. Agent based dynamic load balancing (ABDLB) has been proposed by Grover and Katiyar [5]. The pros of this algorithm are that the consumption time of the central processing unit (CPU) is one unit on the other hand the other known algorithm's consumption time of the CPU is ten units.

Chen *et al.* [6] has proposed a different scheduling algorithm named pro-active and re-active algorithm (PRS) that is designed for dynamic, real time, self-reliant and non-periodic task. The advantages of these algorithms are it can interdict propagation of un-certainties throughout the schedule. PRS algorithm also can lessen the task transfer between operating devices when assigned virtual machine needs migration. Liu *et al.* [7] presented a DeMS algorithm which is the combination of load balancing and task scheduling technique that contains three algorithms; scheduling algorithm based on user demand, migration of task based on query (QMT), and stage task migration (STM). Madni *et al.* [8] presented different allocation methods for cloud environment. In their paper they have found out the parameters by which cloud's activity can be improved. They have discussed about different Resource Allocation strategies also. This article tells us about different algorithms to provide best support for both suppliers and users by distributing and migrating resources among servers.

Sidana *et al.* [9] has proposed a midpoint algorithm named NBST for balancing load on the cloud. This algorithm performs execution by sorting the processing speed of VMs. For allocation at first the list of virtual machine and Cloudlet index are sent to the intermediary agent. By using the middle point algorithm then Broker allocates the VMs. The VM list and cloudlet list is divided by this algorithm until the cloudlet or

listed virtual machines reaches to the highest point. Then this algorithm performs re-source allocation. But there is no live migration mechanism in their proposed method. A number of Hybridization of meta-heuristic algorithm [10]–[13] has been proposed for independent task. They have shown load balancing just comparable with modified particle swarm optimization (MPSO) and Q-learning. Another work [14] that has proposed that firefly and improved multi-objective particle swarm optimization (FIMPSO) model exhibited effective performance but they took the average response time that is very high and got efficiency of 72%. Adaptive cat swarm optimization (ACSO) algorithm [15] has been proposed by another research group but they have not mentioned how it will work for dependent task.

Many researchers have worked to reduce the problem of load migration using heuristic and meta heuristic method [16], [17]. For non-pre-emptive independent tasks on VMs, load balancing based on honeybee behavior [18] has been proposed. Another work named weighted round robin (WRR) algorithm has been recommended [19] using non-primitive dependent task in this regard. They have not shown any results for primitive task. others scheduling algorithm like ipso [20], map reducing hybrid scheduling algorithm [21], and Bayes theorem [22], fuzzy logic [23], load balancing and rebalancing [24], randomized optimization [25], [26] also been proposed. Intensive task scheduling framework using deep reinforcement learning (DRL) with deep Q-network (DQN) [27] has been proposed for suitable VM selection for current situation. Another research team has proposed a method for improving load balancing using QoS, service-level agreement (SLA) violation and energy consumption [28].

Another group of researchers has come up with a framework for cloud/centralized radio access network (C-RAN) architecture concerning two points of recommended completion time, resource distribution and VM migration based on utilization factor. They basically focused on live migration. Self-adaptive load balancing strategy-based works [29], [30] have been proposed by research groups. Using their proposed work, they got quick response based on request arrival for windows media HTTP streaming protocol (WMSP) server thus increases the cost. Optimal user scheduling for multi-cloud (OSMC) [31] has been proposed based on minimum first derivative length (MFDL) of system load paths. Load balancing algorithm using fuzzy set and Q-Learning algorithm [32] has been proposed for multi-tier application in cloud computing. They actually focused on improving Round Robin algorithm. Although they improved their performance but sometimes VMs suffer from starving. Active monitoring load balancer with hill climbing algorithm (IAMLBHC) [33] has been proposed by another research team considering the response time, and processing amount to be asked compared with the existing similar algorithms. Another research group [34] tried to keep down the resources usage and improve load balancing in terms of task parameters such as QoS, priority of VMs, and resource distribution.

So, most researchers have been going through this topic using scheduling algorithms [19]–[25], [31], [32] nowadays. They have shown the balancing but their response time was greater and throughput is less than 80%. Taking these issues such as less throughput, greater response time and problems during migration we have chosen throttled algorithm with some modifications called re-modified throttled algorithm (RMTA). In this algorithm, during migration only the availability of the servers is checked by the data center but in our proposed method we choose the availability of the server as well the capacity is checked with minimum response time to overcome the problem of data overflow and data loss.

Two basic constraints have been considered as significant addition of this proposed work. Loss of migration data while migrating from one server to another and Loss of energy and increased cost. For the solution of load balancing in cloud network, RMTA with consideration feature that checks the availability of the server as well the capacity of the server that reduces the data loss risk, data hampering risk, to keep the same response time of the TMA has been suggested. The output obtained through the proposed algorithm are compared with existing algorithms. Performance validation done through simulation result.

3. RESEARCH METHOD

Figure 1 represents the operation of TMA [2], they have developed a system where they tried to reduce the response time. To do so, they divided the whole available/busy table into two sections- available index and busy index. When DCC receives a new request it sends a query to the TMA load balancer for next allocation. TMA Load Balancer selects the VM's ID from the upper side of "Available Index" chart of DCC.

But limitation still appears in their algorithm which is this algorithm does not deal with the present load on virtual machine. As a result, when the migration process starts, if the request load size is more than the available space of the VM then the data overflow. This can cause data loss or data hampering. At the same time when this situation occurs the response time also increases. We solve this limitation through our proposed algorithm called re-modified throttled algorithm. Figure 2 shows the remodified throttled algorithm's (RMTA) operation. Re-modified Throttled methodology is shown by Algorithm 1.

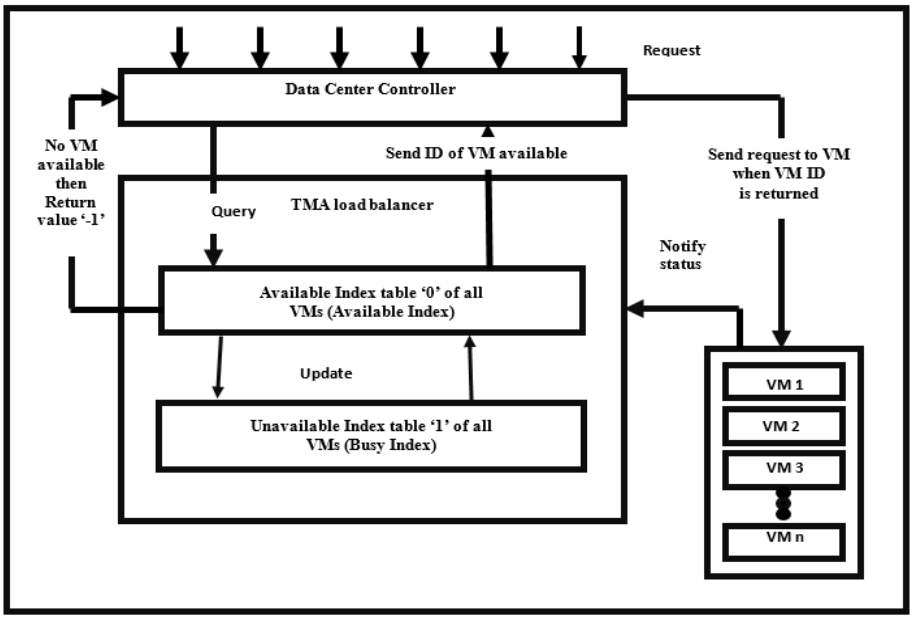

Figure 1. TMA algorithm operation diagram

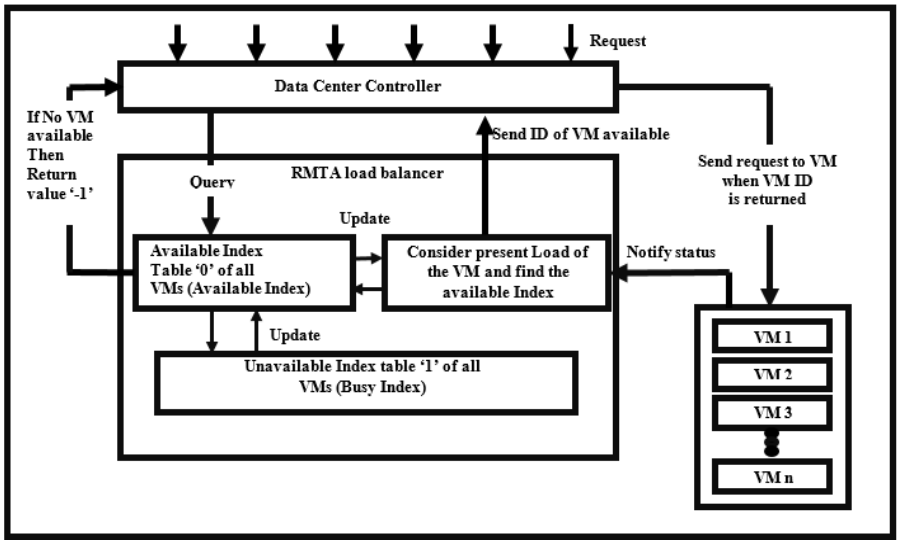

Figure 2. RMTA operation diagram

Algorithm 1: Re-modified Throttled Algorithm
Re-modified Throttled Algorithm (RMTA)
Notations:
R_{new} = *Request to Data Centre*
D_{cc} = *Data Centre*
L_B = *Load balancer*
VM_f = *First available VM*
$VM_{Storage}$ = *Storage of available VM*
$VM_{Requested\ Storage}$ = *Storage of Requested VM*
VM_{SB} = *Bandwidth of Storage VM*
VM_{RB} = *Bandwidth of Requested VM*
$VM_{Available}$ = *Available VM*
$VM_{Unavailable}$ = *Unavailable VM*

$L_{B\,(RMTA)}$ = *RMTA Load Balancer*
A_L = *Alert for New allocation to VM*

Initially $R_{new} = 0$
if $R_{new} \rightarrow D_{cc}$
Do $L_B \rightarrow VM_f$
If $VM_{Storage} \geq VM_{Requested\ Storage}$ && $VM_{SB} \geq VM_{RB}$
Then $VM_{Storage} = VM_{Available}$
$A_L \rightarrow VM_{Storage}$
$L_{B\,(RMTA)} \leftarrow A_L$
If $VM_{Storage} < VM_{Requested\ Storage}$ && $VM_{SB} < VM_{RB}$
Then $VM_{Storage} = VM_{Unavailable}$
Else
Do $L_B \rightarrow VM_f$
Else
$R_{new} \rightarrow D_{cc}$

The re-modified throttled algorithm (RMTA) operation designed by a data center controller, a RMTA load balancer that contains two index table where the VM are being allocated according to their availability or unavailability and most importantly a present load controller which considers the present load of the VMs. Steps for the operations:

Step 1: Re- modified throttled algorithm balances the loads through updating and maintenance of index table.
– Available Index Table: Available VM = '0'.
– Busy Index Table: Unavailable VM = '1'
Initially, "Available Index" reserves all VMs and the "Busy Index" is empty.

Step 2: New request from client is received by DCC.
Step 3: Then query to send to the RMTA load balancer for new task by the data center controller.
Step 4: RMTA Load Balancer selects the VM Id from the upper side of DCC's "Available Index".
Step 5: From the available VM, RMTA finally selects the VM which are available based on the comparison between the VM storage, VM bandwidth and request storage, request bandwidth.
– Allocation request is sent to the selected VM by that VM ID.
– TMA Load Balancer gets an alert from the data center controller for new allocation.
– Selected VM will be sent to the "Busy Index" Table through TMA-LB and wait until DCC sends another new request.
– TMA-LB is set to '-1' and this value will be sent to the DCC.
– The new requested are then arranged by the data center controller (DCC).

Step 6: When processing request is complete, data center controller (DCC) receives an acknowledgment from VM and then advise TMA Load Balancer to update the "Available Index" table.
Step 7: For handling upcoming requests, DCC goes back to Step 3 and continue the process until the "Available Index" is null.

Thus, the RMTA algorithm can detect available virtual machine (VM) with "Available Index" like TMA [1]. It also deals with the present load on VM which was the main limitation of the TMA. The risk of the data loss, data hampering has totally removed and provides higher accuracy than before.

4. RESULTS AND DISUSSSION

The performance RMTA algorithm has been analyzed based on the simulation results. This full experiment has been done through the CloudSim3.0.3 simulator and the whole cloud computing experiment runs on the machine which has a configuration of Intel core i7 processor, 4 GB RAM, 2.4 GHz CPU and Windows 10 platform. The simulation result of the experiments has been presented in Tables 1 and 2. Table 1 is showing the latency and processing time of DC using RMTA algorithm where 20 VMs have been considered initially. In Table 2 we have taken 50 VMs and perform the simulation. The results indicate that the RMTA algorithm generates same response time and also reduces the risk of data loss which was our main concern. The overall performance of our algorithm has been analyzed in terms of migrated task, latency of tasks, total time delayed in all tasks, idle time of tasks, make span before load balancing through Modified Throttled Algorithm [1] and after load balancing through modified re-modified throttled algorithm (RMTA). Table 1 is showing the RMTA-response time.

We have simulated our work for 50 VM after the first simulation. Table 2. Is representing the RMTA-overall response time. The response time of virtual machines using RMTA algorithm. Here we have

shown our results for VM1, VM2, VM3, and VM4 by Figures 3 to 6. Table 3. Shows the data center request servicing time of our algorithm. Here we have shown for 10 VMs among 50 VMs.

Table 1. RMTA-response time summary

Response Time	Average(millisecond)	Minimum(millisecond)	Maximum(millisecond)
1	251.32	202.57	300.07
2	25.57	20.44	30.69
3	95.32	70.57	120.07
4	152.32	121.57	183.07
5	154.195	117.82	190.57
6	150.08	120.83	179.32
7	101.07	79.57	122.57
8	247.55	196.28	298.82
9	102.57	79.07	126.07
10	148.2	117.07	179.32
11	24.94	20.19	29.69
12	151.21	118.54	183.89
13	99.32	75.57	123.07
14	154.57	118.57	190.57
15	259.43	195.07	323.78
16	248.82	193.82	303.82
17	100.57	76.07	125.07
18	24.44	18.94	29.95
19	99.83	71.08	128.57
20	154.55	124.57	184.53

Table 2. RMTA-overall response time summary

Serial	Time	Average (millisecond)	Minimum (millisecond)	Maximum (ms)
1.	Overall Response Time	273.23	37.88	647.56
2.	Overall Data Center Processing Time	0.32	0.02	0.89

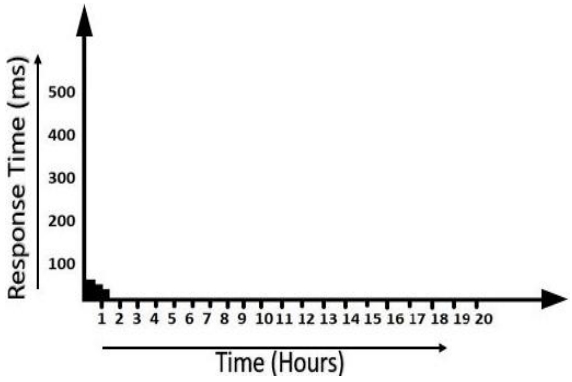

Figure 3. RMTA-response time of VM 1

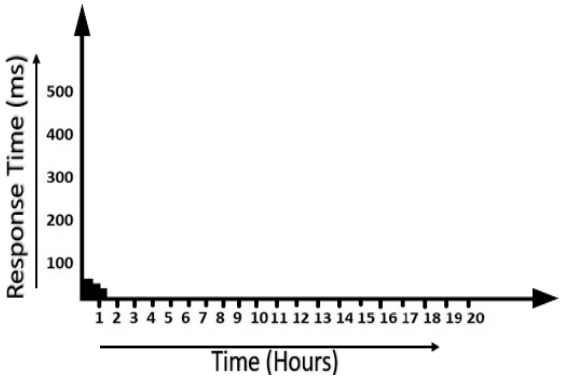

Figure 4. RMTA-response time of VM 2

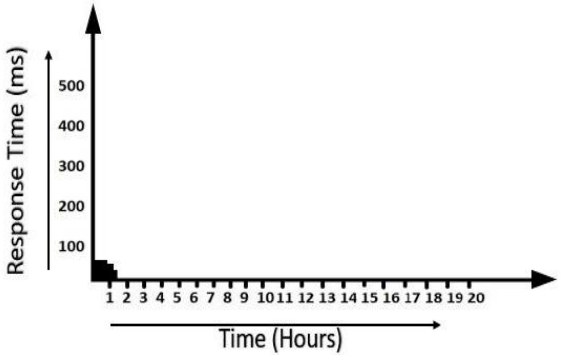

Figure 5. RMTA-response time of VM 3

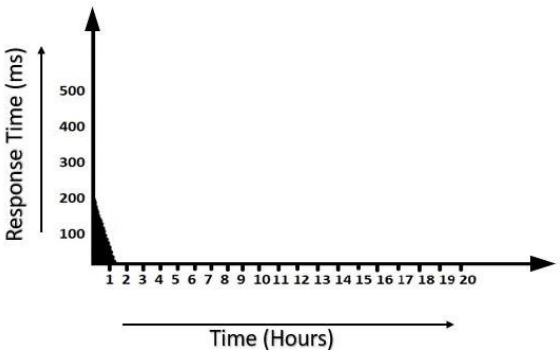

Figure 6. RMTA-response time of VM 4

Table 3. Data center request servicing time summary

Data Center	Average(ms)	Minimum(ms)	Maximum(ms)
1	0.23	0.01	0.44
2	0.22	0.01	0.43
3	0.23	0.01	0.44
4	0.23	0.01	0.45
5	0.23	0.01	0.44
6	0.23	0.01	0.45
7	0.23	0.01	0.44
8	0.23	0.01	0.45
9	0.23	0.01	0.45
10	0.23	0.01	0.44

For best-case response time and processing time remains same but for the worst-case they differ from TMA. The difference occurs because for the worst case RMTA checks each VM until the VM fulfill the user requirements. Figures 7 and 8 shows the response time analysis and processing time analysis of TMA and RMTA, respectively. We have compared our algorithm with the other related algorithm. Our algorithm focuses on the reduction of the data loss risk as well as consider the response time and processing time. Table 4 is showing the efficiency of our algorithm comparing with some other related algorithms.

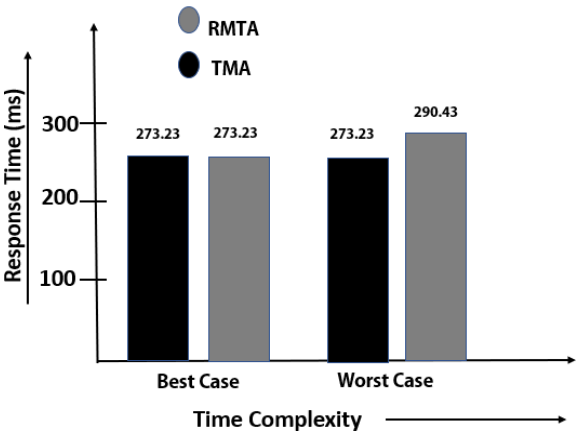

Figure 7. Response time analysis of TMA and RMTA

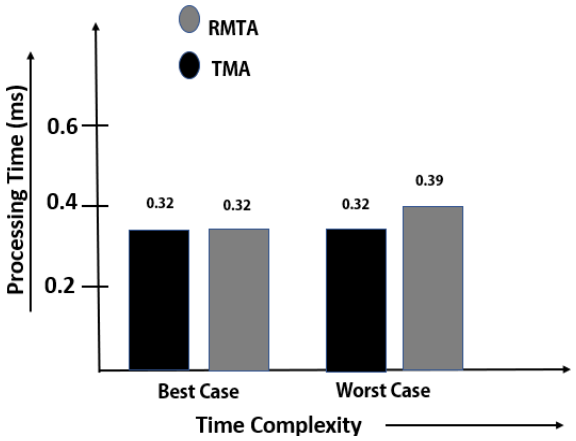

Figure 8. Processing time analysis of TMA and RMTA

Table 4. Comparison of our algorithm with the related algorithm

Serial	Algorithms	Reduce Response Time	Reduce VM Cost	Reduce Transfer Cost	Reduce Data Loss Cost
1.	Round Robin	Yes	Yes	Yes	No
2.	ESCE	Yes	Yes	Yes	No
3.	Throttled	Yes	Yes	Yes	No
4.	TMA	Yes	Yes	Yes	No
5.	RMTA	Yes	Yes	Yes	Yes

5. CONCLUSION

In this work, we have analyzed many load balancing algorithms and suggested RMTA algorithm with the concept of throttled modified algorithm that has accomplished the following goals. The output that we gained from the proposed algorithm is capable of reducing the data loss and maintain minimum response time compared to the old TMA algorithm. When the number of VMs increases RMTA algorithm has shown efficiencies such as reduced response and processing time of cloud data centers. In future, we will try to solve the problems of worst cases and increase the efficiency of RMTA algorithm.

REFERENCES

[1] S. G. Domanal and G. R. M. Reddy, "Load Balancing in cloud computingusing modified throttled algorithm," in *2013 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, Oct. 2013, pp. 1–5, doi: 10.1109/CCEM.2013.6684434.

[2] N. Xuan Phi, C. T. Tin, L. N. Ky Thu, and T. C. Hung, "Proposed load balancing algorithm to reduce response time and processing time on cloud computing," *International journal of Computer Networks & Communications*, vol. 10, no. 3, pp. 87–98, May 2018, doi: 10.5121/ijcnc.2018.10307.

[3] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *2011 Sixth Annual Chinagrid Conference*, Aug. 2011, pp. 3–9, doi: 10.1109/ChinaGrid.2011.17.

[4] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A genetic algorithm (GA) based load balancing strategy for cloud computing," *Procedia Technology*, vol. 10, pp. 340–347, 2013, doi: 10.1016/j.protcy.2013.12.369.

[5] J. Grover and S. Katiyar, "Agent based dynamic load balancing in cloud computing," in *2013 International Conference on Human Computer Interactions (ICHCI)*, Aug. 2013, pp. 1–6, doi: 10.1109/ICHCI-IEEE.2013.6887799.

[6] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, and J. Wu, "Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment," *Journal of Systems and Software*, vol. 99, pp. 20–35, Jan. 2015, doi: 10.1016/j.jss.2014.08.065.

[7] L. Liu *et al.*, "GreenCloud: a new architecture for green data center," in *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session - ICAC-INDST '09*, 2009, pp. 29–38, doi: 10.1145/1555312.1555319.

[8] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Recent advancements in resource allocation techniques for cloud computing environment: a systematic review," *Cluster Computing*, vol. 20, no. 3, pp. 2489–2533, Sep. 2017, doi: 10.1007/s10586-016-0684-4.

[9] S. Sidana, N. Tiwari, A. Gupta, and I. S. Kushwaha, "NBST algorithm: A load balancing algorithm in cloud computing," in *2016 International Conference on Computing, Communication and Automation (ICCCA)*, Apr. 2016, pp. 1178–1181, doi: 10.1109/CCAA.2016.7813914.

[10] F. Tang, L. T. Yang, C. Tang, J. Li, and M. Guo, "A dynamical and load-balanced flow scheduling approach for big data centers in clouds," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 915–928, Oct. 2018, doi: 10.1109/TCC.2016.2543722.

[11] S. Nabi and M. Ahmed, "OG-RADL: overall performance-based resource-aware dynamic load-balancer for deadline constrained Cloud tasks," *The Journal of Supercomputing*, vol. 77, no. 7, pp. 7476–7508, Jul. 2021, doi: 10.1007/s11227-020-03544-z.

[12] S. Omranian-Khorasani and M. Naghibzadeh, "Deadline constrained load balancing level based workflow scheduling for cost optimization," in *2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA)*, Sep. 2017, pp. 113–118, doi: 10.1109/CIAPP.2017.8167191.

[13] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *Journal of King Saud University - Computer and Information Sciences*, Feb. 2020, doi: 10.1016/j.jksuci.2020.01.012.

[14] A. F. S. Devaraj, M. Elhoseny, S. Dhanasekaran, E. L. Lydia, and K. Shankar, "Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments," *Journal of Parallel and Distributed Computing*, vol. 142, pp. 36–45, Aug. 2020, doi: 10.1016/j.jpdc.2020.03.022.

[15] K. Balaji, P. Sai Kiran, and M. Sunil Kumar, "An energy efficient load balancing on cloud computing using adaptive cat swarm optimization," *Materials Today: Proceedings*, Jan. 2021, doi: 10.1016/j.matpr.2020.11.106.

[16] R. Ezumalai, G. Aghila, and R. Rajalakshmi, "Design and architecture for efficient load balancing with security using mobile agents," *International Journal of Engineering and Technology*, vol. 2, no. 1, pp. 57–60, 2010, doi: 10.7763/IJET.2010.V2.100.

[17] N. Malarvizhi and V. Rhymend Uthariaraj, "Hierarchical load balancing scheme for computational intensive jobs in Grid computing environment," in *2009 First International Conference on Advanced Computing*, Dec. 2009, pp. 97–104, doi: 10.1109/ICADVC.2009.5378268.

[18] D. B. L.D. and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol. 13, no. 5, pp. 2292–2303, May 2013, doi: 10.1016/j.asoc.2013.01.025.

[19] D. C. Devi and V. R. Uthariaraj, "Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks," *The Scientific World Journal*, vol. 2016, pp. 1–14, 2016, doi: 10.1155/2016/3896065.

[20] H. Saleh, H. Nashaat, W. Saber, and H. M. Harb, "IPSO task scheduling algorithm for large scale data in cloud computing environment," *IEEE Access*, vol. 7, pp. 5412–5420, 2019, doi: 10.1109/ACCESS.2018.2890067.

[21] E. Jafarnejad Ghomi, A. Masoud Rahmani, and N. Nasih Qader, "Load-balancing algorithms in cloud computing: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 50–71, Jun. 2017, doi: 10.1016/j.jnca.2017.04.007.

[22] J. Zhao, K. Yang, X. Wei, Y. Ding, L. Hu, and G. Xu, "A heuristic clustering-based task deployment approach for load balancing using bayes theorem in cloud environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 305–316, Feb. 2016, doi: 10.1109/TPDS.2015.2402655.

[23] A. N. Toosi and R. Buyya, "A fuzzy logic-based controller for cost and energy efficient load balancing in geo-distributed data centers," 2015, doi: 10.1109/UCC.2015.35.

[24] X. Deng, D. Wu, J. Shen, and J. He, "Eco-aware online power management and load scheduling for green cloud datacenters," *IEEE Systems Journal*, vol. 10, no. 1, pp. 78–87, Mar. 2016, doi: 10.1109/JSYST.2014.2344028.

[25] H.-C. Hsiao, H.-Y. Chung, H. Shen, and Y.-C. Chao, "Load rebalancing for distributed file systems in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 5, pp. 951–962, May 2013, doi: 10.1109/TPDS.2012.196.

[26] A. V. Papadopoulos *et al.*, "Control-based load-balancing techniques: Analysis and performance evaluation via a randomized optimization approach," *Control Engineering Practice*, vol. 52, pp. 24–34, Jul. 2016, doi: 10.1016/j.conengprac.2016.03.020.

[27] Z. Tong, X. Deng, H. Chen, and J. Mei, "DDMTS: A novel dynamic load balancing scheduling scheme under SLA constraints in cloud computing," *Journal of Parallel and Distributed Computing*, vol. 149, pp. 138–148, Mar. 2021, doi: 10.1016/j.jpdc.2020.11.007.

[28] S. M. Moghaddam, M. O'Sullivan, C. P. Unsworth, S. F. Piraghaj, and C. Walker, "Metrics for improving the management of Cloud environments — Load balancing using measures of quality of service, service level agreement violations and energy consumption," *Future Generation Computer Systems*, vol. 123, pp. 142–155, Oct. 2021, doi: 10.1016/j.future.2021.04.010.

[29] B. Mahapatra, A. K. Turuk, S. K. Panda, and S. K. Patra, "Utilization-aware VB migration strategy for inter-BBU load balancing in 5G cloud radio access networks," *Computer Networks*, vol. 181, Nov. 2020, doi: 10.1016/j.comnet.2020.107507.

[30] R. Li, G. Dong, J. Jiang, H. Wu, N. Yang, and W. Chen, "Self-adaptive load-balancing strategy based on a time series pattern for concurrent user access on Web map service," *Computers & Geosciences*, vol. 131, pp. 60–69, Oct. 2019, doi: 10.1016/j.cageo.2019.06.015.

[31] B. Zhang, Z. Zeng, X. Shi, J. Yang, B. Veeravalli, and K. Li, "A novel cooperative resource provisioning strategy for multi-cloud load balancing," *Journal of Parallel and Distributed Computing*, vol. 152, pp. 98–107, Jun. 2021, doi: 10.1016/j.jpdc.2021.02.003.

[32] Y. Zhao, C. Liu, H. Wang, X. Fu, Q. Shao, and J. Zhang, "Load balancing-based multi-controller coordinated deployment strategy in software defined optical networks," *Optical Fiber Technology*, vol. 46, pp. 198–204, Dec. 2018, doi: 10.1016/j.yofte.2018.10.012.
[33] M. Zedan, G. Attiya, and N. El-Fishawy, "Load balancing based active monitoring load balancer in cloud computing," in *2021 International Conference on Electronic Engineering (ICEEM)*, Jul. 2021, pp. 1–6, doi: 10.1109/ICEEM52022.2021.9480611.
[34] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, "A load balancing algorithm for the data centres to optimize cloud computing applications," *IEEE Access*, vol. 9, pp. 41731–41744, 2021, doi: 10.1109/ACCESS.2021.3065308.

BIOGRAPHIES OF AUTHORS

Fatema Tuj Johora is serving as a lecturer at Daffodil International University since 2017. She has completed her MSc (IT) from Jahangirnagar University. Her research interests are Image Processing, Cloud Computing and IOT. She can be contacted at email: fatema.cse@diu.edu.bd.

Iftakher Ahmed is currently pursuing his M.Sc. from Jahangirnagar University, Dhaka, Bangladesh. He obtained his B.Sc (CSE) from Daffodil International University, Dhaka, Bangladesh. His research interests are Cloud Computing, Internet of Things (IoT). He can be contacted at email: iftakher15-960@diu.edu.bd.

Md. Ashiqul Islam Shajal is presently continuing his M.Sc. from Jahangirnagar University, Dhaka, Bangladesh. He has completed his B.Sc. (CSE) from Daffodil International University, Dhaka, Bangladesh. His research interests are Cloud Computing, Database. He can be contacted at email: islam15-987@diu.edu.bd.

Rony Chowdhury is studying at Jahangirnagar University, Dhaka, Bangladesh to complete his M.Sc. He has completed his B.Sc. (CSE) from Daffodil International University, Dhaka, Bangladesh. His research interests are cloud computing, and machine learning. He can be contacted at email: roany15-968@diu.edu.bd.