MLMGAD Configuration to Expand Load Management for Cloud

Akhilesh Kumar Bhardwaj, Rajiv Mahajan, Surender Kumar

Abstract: ***Load management is a principle challenge in distributed cloud communication to circulate the load over various nodes to guarantee that no individual node is sitting idle, overburdened or under-burdened. In the cloud, load management is a key apprehension. The number of cloud clients is likewise expanded from 2.4 billion in 2013 to 3.6 billion in 2018. It implies 33% ascent in the number of clients in the last half decade. So, better load management is constantly required for a cloud client fulfillment to benefit the related services without any delay or information loss. Clearly, cloud organizations require better and reshaped workload engineering with the progression of time. This work of interest has persuaded us to pick this task and to create a modified load management genetic algorithm design (MLMGAD) is projected to compute and compare different performance measurements. The central goal of this work is clustered into five segments. (1) To test whether the implemented procedure is improving throughput and limiting packet delay. (2) To examine and reduce packet loss. (3) To examine and improve the response time by reason. (4) To examine and improve overall response time, overall data center processing time and average, minimum and maximum response time. (5) To examine and enhance execution time.***

Index Termss: Cloud, Load management, MLMGAD, Performance measurements

I. INTRODUCTION

The cloud outline is facilitated by different sorts of virtual systems on a physical server. Three major cloud model-public, private and hybrid are in contemplate with more sub-classifications developing. Public cloud is accessible to open for various application and capacity utilities. Private cloud helps in overseeing operations inside an explicit trade. Hybrid cloud is fitting for inward and outer services phenomenon. Public cloud framework incorporates load management, secrecy, security, organization investigation, and reliability. Load management is a method of separating the load between various nodes to guarantee that no node in the coordination is overburdened, under burdened or sits idle. The worry is to enhance response time with skilled asset use. Dealing with the tasks is the intense component since consistent forecast of the volume of issued tasks in cloud situation is extremely intense. Cloud changing characteristic is the key reason for this unconventionality. The numbers of cloud clients and their services prerequisites are expanding minutes by minutes. The cloud stage must furnish its enhancements to the clients with their most extreme fulfillment. Effective load management helps in engaging scalability, evades holdups and decreases responding time (P. P. G. Gopinath et al., 2015).

Load management is accomplished in a cloud environment in essentially two stages: in the prior one, the apportioning of the load occurs among the nodes. The later one spotlights on checking the virtual machines (VMs) and to actualize the load overseeing activities utilizing tasks movement or VM relocation technique. After handover the assignments to virtual machines, the cloud task scheduler begins playing out the load taking care of activity to migrate the tasks from overburdened to under-burdened VMs and all VMs need to persevere the adjusting condition (M. Kumara et al., 2017). Load planning intentionally strengthens different resources and systems by giving up in throughput and down response time (S.Srivastva et al., 2018).

It likewise parts the activity development among various servers while alluding and getting the data without deferring. It fairly disperses the resources with reasonable resource usage with most extreme client fulfillment. Static and dynamic improvements are utilized for the required working. Static algorithms bargain out the undertakings amid compiling time. They are non-preemptive in nature and work with the thought process of decreasing overall execution time. No unique changes are required at runtime. Dynamic algorithms bolster in reallocating the tasks or jobs.

II. RELATED LITERATURE SURVEY

In this division, a review of the previous strategies is presented before the proposed design. (Z. Zhang et al., 2010) have anticipated an ant colony based load management procedure for the distributed condition. The central object is to investigate the software and hardware resources of the cloud to deliver better throughput and resource usage (H. Ren et al., 2012).

(A. Jain et al., 2012) depicted a load assessing technique inside different processing units by actualizing dynamic programming with brute force strategy. Trials were accomplished utilizing Java.

(F. Ramejani et al., 2014) proposed PSO to deal with the tasks course of action trouble in cloud load

Revised Manuscript Received on June 07, 2019

Akhilesh Kumar Bhardwaj, Research Scholar, Department of Computer Science and Engineering, IKG PTU, Punjab, India
Rajiv Mahajan, Professor, Department of Computer Science and Engineering, GCET, Gurdaspur, Punjab, India
Surender Kumar, Assistant Professor, Department of Computer Science, Guru Teg Bahadur College, Sangrur, Punjab, India

management. However, this type of algorithms traps in nearby optima. (P.Samal et al., 2013) anticipated round-robin method with measurements like resource test run and response time. Yet this calculation doesn't diminish the makespan or response time. (B.Sahoo et al., 2013) proposed the greedy strategy to decrease the makespan and execution time without using VM movement or undertaking task migration load adjusting approach. These engagements don't sit well in authentic circumstance. (S. Singh et al., 2014) put forward reformed genetic algorithm to resolve load planning issue by taking thought on max-min plan to create better outcomes for makespan time. (C. C. Lin et al., 2014) suggested an efficient methodology for the multimedia framework in the cloud with Balancing of multimedia load i.e. video, sound, and pictures between every one of the servers with lesser cost. (E. Pacini et al., 2015) proposed classification of dynamic processes, for example, ACO for load planning component. (A. V. Lakra et al., 2015) recommended a technique to decrease turn-around time, cost and increase throughput measurements. (A.Thomas et al., 2015) Implemented dynamic algorithm for makespan time, however, the endeavor was not able to decrease makespan time completely for a massive number of tasks. (S. A. Hamad et al., 2016) Suggested improved genetic algorithm situated planning strategy for use of resources, cost and completion time. (A. Tripathi et al., 2018) included an improved hybrid approach. (G. Kaur et al., 2017) prescribed firefly based scheduling strategy for load management. (Sonam et al., 2018) recommended incorporation of cloud partitioning and load adjusting component.

All things considered, the requirement for an updated task scheduling process is exceptionally required to allow the tasks in an effective mode so that less number of virtual machines face either overburden or under-under-burden condition.

III. THE MOTIVATION FOR THIS WORK

The number of cloud clients is likewise expanded from 2.4 billion in 2013 to 3.6 billion in 2018. It implies 33% ascent in the number of clients in the last half decade. The worldwide cloud traffic was 5636 Exabyte in 2017. It is 7712 Exabyte in 2018 and 11851 Exabyte in 2020 (Statista, 2018). So, better load management is constantly required for a cloud client fulfillment to benefit the related services without any delay or information loss. Clearly, cloud organizations require better and reshaped workload engineering with the progression of time. This work of interest has persuaded us to pick this task and to create MLMGAD with a dynamic standpoint.

IV. MLMGAD ARCHITECTURE

In the evaluation section, a hybrid approach deals with the movement to load. This methodology has made an overhead cost of the server and furthermore the response time is higher. Though in the ant colony scheme, it is considered that overall processing time gets slow and the said calculation is exceptionally tedious as well. In the existing algorithms, least and most threshold values are set. Presently before allotting the workload to the cloud server, if load < threshold, then the workload is assigned and finish one iteration. Here no inspection procedures are executed about the tasks handled are done productively or not by the assigned server.

To eliminate this issue, the proposed MLMGAD before assigning the workload checks the load. In the event that it is less than the threshold, the tasks are relegated and in parallel, the response time of the server is inspected. On the off chance that it is less than the average response time then all the tasks are migrated to the next cloud file server. In cloud partitioning based load balancing, the issue is that task migration starting with one segment then onto the next is slow since that procedure is done by partition load balancer which refreshes all VMs after some time. In the suggested algorithm, whatever the tasks are finished by virtual machines, the load balancer updates to every virtual node in parallel. Henceforth by fusing a modified dynamic load engagement design, the cloud system can be made proficient without overburdening or under burdening the nodes.

MLMGAD is an inquiry-based improvement method backings to get the optimal solutions for testing issues.

i. It supports parallel abilities in productive and quicker mode.

ii. In MLMGAD, we have a group of potential solutions. The procedure at that point run over and finished with recombination and transformation.

iii. To every child server, a fitness value is allocated followed by Darwin hypothesis.

iv. MLMGAD helps in furnishing a scale of good solutions in place of a single solution.

v. MLMGAD is helpful in complex search space and does not desirous of derivative data.

MLMGAD Mechanism

1. Analyze user workload of Running clouds based on a formula CTN=(NPR+CRU+TFS+NWBW) …........(Eq. 1)
2. Main cloud server receives a request from users and store in AFQ and then check CTN of each running file server
3. If CTN< T of CS1,
Then assign jobs to CS1 Else Assign job to CS2
4. Then check the RT of assigned CS1 and if RT is higher then threshold ARTE then assign the job to another cloud
5. If Task type = Ci for each VM, Then migrate files to another cloud.

MLMGAD Working Steps

1. [Start]:
In this initial step, run n number of child servers.
2. [Fitness]
Compute wellness estimation of every child server with condition $\sum$ K=0 ∞ 1 (k+1) (k+3)…………(k+n) ……………………………………………….........(Eq. 2)
3. [New population]: Generate a fresh random population by rehashing following advances.
3.1 [Selection]: Get fitness estimation of every child server and afterward contrast and each other child server lastly sort of view of mounting request and store in the exhibit. Cluster [fitness 1, fitness 2, fitness 3…fitness N]
3.2 [Crossover]: Assign task to minimum fitness value child server and furthermore compute the energy of each running child server beneath.

3.3 [Mutation]: With a transformation likelihood, compute the response of each running child server
3.4 [Replace]: If the current child server response is not as much as the threshold at that point, replace child server with another higher fitness value child server

Table 1. MLMGAD Configuration

Sr. no.	Notation	Meaning
1	CTN	Cost of network
2	NPR	Numbers of the process running
3	CRU	Current ram usage
4	TFS	Total file size
5	NWBW	Network bandwidth
6	AFQ	An array of file queue
7	T	Threshold value
8	CS1	Cloud server 1
9	CS2	Cloud server 2
10	RT	Response time
11	ARTE	Actual response time
12	Task type	Length of all transactions
13	VM	Virtual machine

V. EXPERIMENTAL RESULTS AND ANALYSIS

The tests were led continuously condition utilizing JDK 1.8, Net Beans 8.1, My SQL 5.5, Java Swing and Network Programming with RAM 2 GB and HDD 50 GB.

Table 2. Throughput (Mbps)

CASE	Existing	MLMGAD
CASE 1	788	900
CASE 2	756	800
CASE 3	748	790
CASE 4	748	790

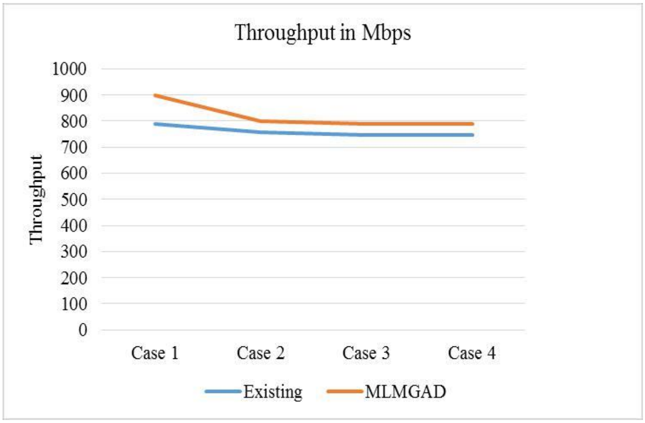

Figure 1. Throughput (Mbps)

Table 3. Packet delay (ms)

CASE	Existing	MLMGAD
CASE 1	3	1.5
CASE 2	42	30
CASE 3	10	7
CASE 4	9	8

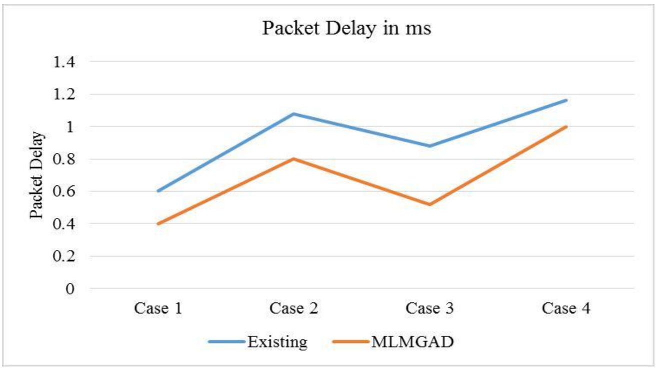

Figure 2. Packet delay (ms)

Table 4. Packet loss (%)

CASE	Existing	MLMGAD
CASE 1	3	1.5
CASE 2	42	30
CASE 3	10	7
CASE 4	9	8

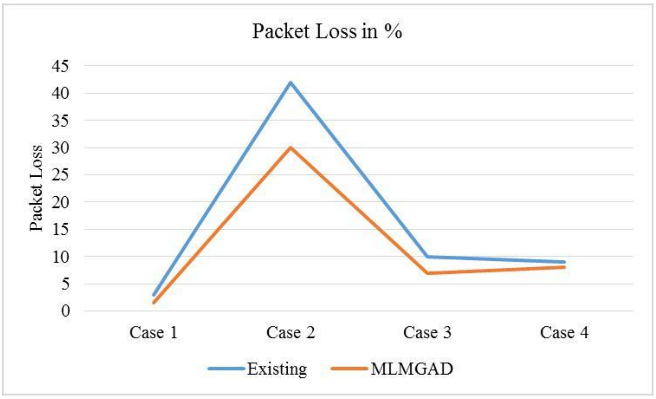

Figure 3. Packet loss (%)

Table 5. Response time by reason (ms)

User base	Existing	MLMGAD
UB 1	68.37	67.23
UB 2	75.68	74.56
UB 3	77.49	76.46
UB 4	342.19	341.28
UB 5	374.04	373.87
UB 6	283.46	282.54

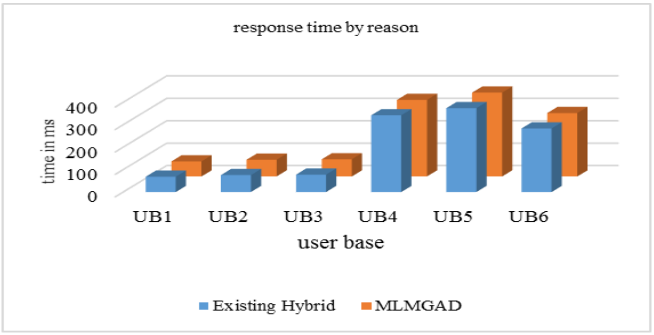

Figure 4. Response time by reason (ms)

*Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication*

Table 6. Overall response time (ms)

Existing	MLMGAD
256.94	254.03

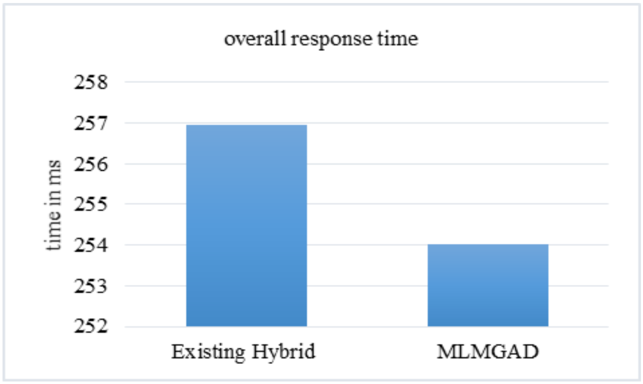

Figure 5. Overall response time (ms)

Table 7. Data center servicing time (ms)

Data center	Existing	MLMGAD
UB 1	55.63	54.78
UB 2	24.10	23.98
UB 3	49.90	48.54

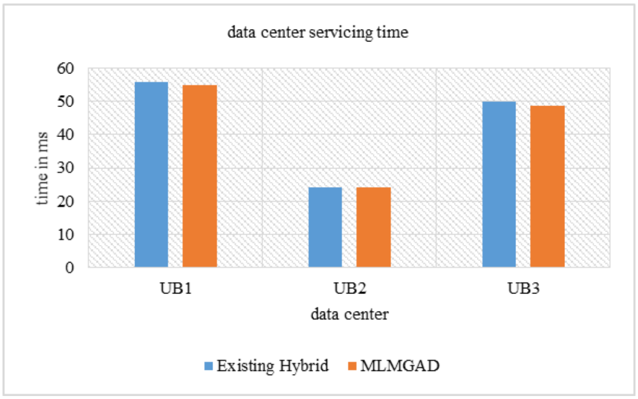

Figure 6. Data center servicing time (ms)

Table 8. Overall data center processing time (ms)

Existing	MLMGAD
47.09	45.86

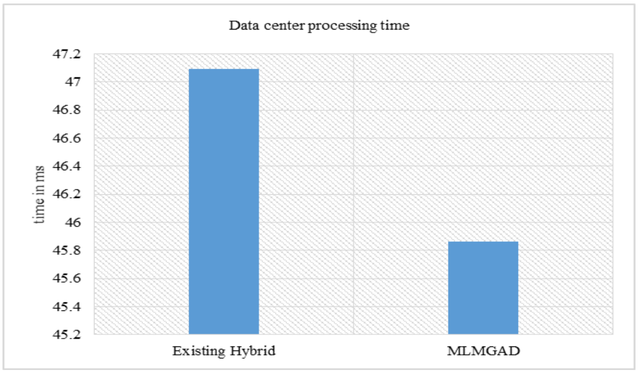

Figure 7. Overall data center processing time (ms)

Table 9. Response time (ms)

Response time	Existing	MLMGAD
Average	0.25	0.17
Minimum	0.10	0.05
Maximum	0.48	0.36

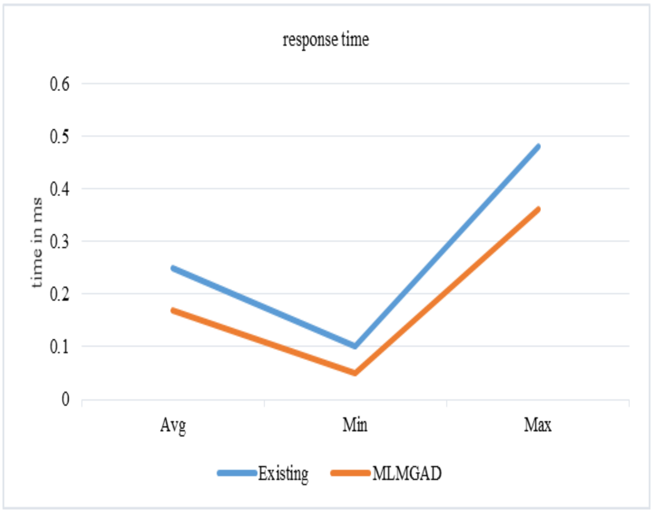

Figure 8. Response time (ms)

Table 10. Execution time (ms)

Config.	Existing	MLMGAD
C1	900	840
C2	850	760
C3	580	400

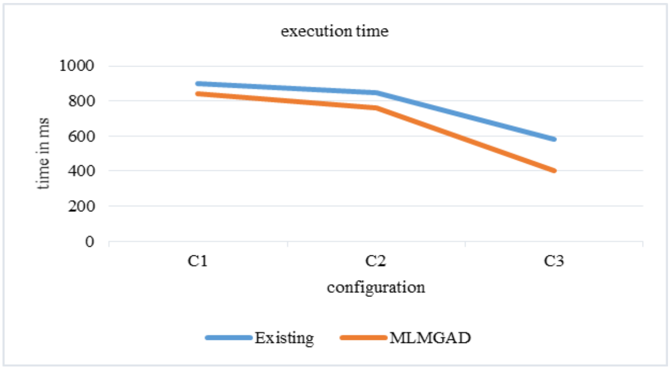

Figure 9. Execution time (ms)

From the outcomes derived, cloud network performance is improved fundamentally. The reason is that file server are various, so if response time gets an increment, the activity can be effortlessly moved to another file server.

As presented in table 2 and figure 1, the throughput was observed for case 1,2,3,4 with the existing. Throughput for case 1 is enhanced to 1.12% and 0.44%, 0.52% and 0.52% for the cases 2, 3 and 4 respectively.

As presented in table 3 and figure 2, the packet delay was examined for case 1,2,3,4 with the existing. Delay for case 1 is reduced to 33.33% and 25.92%, 40.90% and 13.79% for the cases 2, 3 and 4 respectively.

As presented in table 4 and figure 3, the packet loss was observed for case 1,2,3,4 with the existing. The loss for case 1 is minimized to 50% and 28.97%, 30% and 11.11% for the cases 2, 3 and 4 respectively.

As presented in table 5 and figure 4, the response time by reason was examined for user base 1, 2, 3, 4, 5 and 6 with the existing. It is improved over the existing as 1.67%, 1.48%, 1.33%, 0.27%, 0.05% and 0.32% respectively. As presented in table 6 and figure 5, the overall response time was observed with the existing hybrid approach. It is improved over the existing with 1.13%. As presented in table 7 and figure 6, the data center servicing time was examined for user base 1, 2, and 3 with the existing. It is improved over the existing as 1.53%, 0.49%, and 2.73% respectively. As presented in table and figure 7, the Overall data center processing time was observed with the existing hybrid approach. It is improved over the existing with 2.61%. As presented in table 9 and figure 8, the average, minimum and maximum response time was examined with the existing. It is found improved over the existing as 32%, 50%, and 25% respectively.

As presented in table 10 and figure 9, the execution time was observed for user configuration 1, 2, and 3 with the existing. It is improved over the existing as 6.67%, 10.58%, and 31.03% respectively.

Table 11. Performance Measurement Overview

Parameter	For	Improved over existing (in %)
Throughput	Case 1	1.12
	Case 2	0.44
	Case 3	0.52
	Case 4	0.52
Packet delay	Case 1	33.33
	Case 2	25.92
	Case 3	40.90
	Case 4	13.79
Packet loss	Case 1	50
	Case 2	28.97
	Case 3	30
	Case 4	11.11
Response time	User base 1	1.67
	User base 2	1.48
	User base 3	1.33
	User base 4	0.27
	User base 5	0.05
	User base 6	0.32
Overall Response time		1.13
Data center servicing time	User base 1	1.53
	User base 2	0.49
	User base 3	2.73
Overall data center processing time		2.61
Response time	Average	32
	Minimum	50
	Maximum	25
Execution time	Config. C1	6.67
	Config. C2	10.58
	Config. C3	31.03

The results presented in table 11 show that MLMGAD scheduling essentially helps in acquiring better and resource usage in cloud innovation.

VI. CONCLUSION

The concluding effort is to enhance load management process in a cloud situation. System execution and assets utility measurements were recognized utilizing hereditary based modified genetic application.

The measurements looked at were throughput, packet delay, packet loss, response time by reason, overall response time, data center servicing time, overall data center processing time, average-minimum-maximum response time, and execution time. The investigation gives the advanced system execution to the overall resource management.

REFERENCES

1. P. P. G. Gopinath and S. K. Vasudevan, "An in-depth analysis and study of Load balancing techniques in the cloud computing environment", *Procedia Computer Science,* 50: 427–432, 2015
2. M. Kumara and S. C. Sharma. "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing", *Communications.* 1877: 509, 2017
3. S. Srivastava and S. Singh, "Performance optimization in cloud computing through cloud partitioning-based load balancing", *Advances in Computer and Computational Sciences. Springer,* 301–311, 2018
4. Z. Zhang and X. Zhang, "A load balancing mechanism based on ant colony and complex network theory in open cloud computing", *Proc. Of 2nd International Conference on Industrial Mechatronics and Automation,* 2: 240-243, 2010
5. H. Ren, Y. Lan, and C. Yin, "The load balancing algorithm in a cloud computing environment," *Proceedings of the 2nd International Conference in Computer Science and Network Technology*, 925– 928, 2012
6. A. Jain and N. S. Chaudhari, "Genetic algorithm based concept design to optimize network load balance", *Journal of Soft Computing,* 2(4): 357–360, 2012
7. F. Ramezani, J. Lu, and F. K. Hussain; "Task-based system load balancing in cloud computing using particle swarm optimization", *International Journal of Parallel Program,* 42(5): 739–754, 2014
8. P. Samal and P. Mishra, "Analysis of variants in round-robin algorithms for load balancing in cloud computing," *International Journal of Computer Science and Information Technology,* 4(3): 416– 419, 2013
9. B. Sahoo, D. Kumar, and S. K. Jena, "Analysing the impact of heterogeneity on greedy resource allocation algorithms for dynamic load balancing in a heterogeneous distributed computing system", *International Journal Computer Applications,* 62(19), 2013
10. S. Singh and M. Kalra, "Scheduling of independent tasks in cloud computing using the modified genetic algorithm", *International Conference on Computational Intelligence and Communication Networks,* 565–569, 2014
11. C.-C. Lin, H.-H. Chin, and D.-J. Deng, "Dynamic multiservice load balancing in the cloud-based multimedia system", *IEEE System Journal,* 8(1): 225–234, 2014
12. E. Pacini, C. Mateos, and C. G. Garino, "Balancing throughput and response time in online scientific clouds via ant colony optimization", Advance *Engineering Software,* 84: 31–47, 2015
13. A. V. Lakra and D. K. Yadav, "A multi-objective tasks scheduling algorithm for cloud computing throughput optimization", *Procedia Computer Science*, 48: 107–113, 2015
14. A. Thomas, G. Krishnalal, and V. P. J. Raj, "Credit-based scheduling algorithm in a cloud computing environment", *Procedia Computer Science* 46: 913–920, 2015
15. S. A. Hamad and F. A. Omara, "Genetic-based task scheduling algorithm in a cloud computing environment", *International Journal of Advanced Computer Science* 7(4): 550–556, 2016
16. A. Tripathi, S. Shukla, and D. Arora, "A hybrid optimization approach for load balancing in cloud computing", *Advances in Computer and Computational Sciences*, 197–206, 2018
17. G. Kaur and A. Kaur, "An adaptive firefly algorithm for load balancing in cloud computing", Proc. *of the 6th international conference on soft computing for problem-solving* 63-72, 2017

18. Sonam Srivastava and S Singh, "Performance optimization in cloud computing through cloud partitioning based load balancing", *Advances in Intelligent Systems and Computing,* 301–311, 2018
19. www.statista.com, "Global cloud traffic by region worldwide", 2016-2021.

AUTHORS PROFILE

Akhilesh Kumar Bhardwaj got his B.Tech Degree in Computer Science and Engineering from Kurukshetra University and M.Tech degree in Computer Science from Rajasthan Technical University. At present, he is a Ph.D. research scholar in the Department of Computer Engineering at IKG Punjab Technical University, Punjab.

Rajiv Mahajan got his Ph.D. degree in Computer Engineering with Specialization in Wireless Communication and Security. His exploration zone is security in the ad-hoc system. As of now, he is currently working as Director-Principal at Golden College of Engineering and Technology, Gurdaspur, Punjab, India. He has 15 Years of Teaching and Research Experience. He has supervised more than 20 Ph.D. and M.Tech. Scholars. He has conducted different sessions at National and International level Conferences. Rajiv Mahajan has published more than 130 research papers in reputed Journals and Conferences. He is board member and reviewer for different International Journals.

Surender Kumar has done an M.Tech degree in Computer Science and Engineering from CDLU and M.Phil. in Computer Science from Alagappa University. He has received Ph.D. in Computer Science and Application from Kurukshetra University. He has more than 10 years of teaching experience. He is currently working as an Assistant Professor, Department of Computer Science, Guru Teg Bahadur College, Bhawanigarh (Sangrur), Punjab, India. He has published over 60 publications in reputed Journals and Conferences. His research interests include Fault Tolerance in Mobile Distributed Systems, Adhoc N/W, System Security, and Cryptography.