ReT-ELBa: A Novel Algorithm for Efficient Load Balancing in Cloud Computing

-

ABSTRACT

This paper presents a novel load-balancing algorithm for balancing load on the cloud environment which we call Response Time Efficient Load Balancer (ReT-ELBa). Load balancing involves a dynamic and equal shifting of workloads among processors or Virtual Machines to achieve better resource utilisation. The study gives an insight into the design, implementation and evaluation of an enhanced load-balancing algorithm that allocates tasks across Virtual Machines on the cloud, which minimises Response Time and increases resource utilisation. The ReT-ELBa distributes tasks considering their size and requirements for their executionsas well as the state of state of Virtual Machines. The Cloud Analyst simulation tool was employed to simulate or evaluate various cases for the ReT-ELBa in comparison with the Throttled and Round Robin algorithms. The study reveals that the ReT-ELBa outperformed the two algorithms in relation to Response Time. The ReT-ELBa also outperformed the two algorithms in relation to Data Centre Processing Time for all simulated cases except one (Case 3), which recorded a small difference of 0.07ms. Even though the Response Time was the Primary focus of this research, the Data Centre Processing Time of the algorithm was also analysed to present a more elaborate representation of the performances of the algorithms.

Keywords: Load-Balancing; Cloud Computing; Algorithms; Distributed Systems.

1. INTRODUCTION

Cloud computing is a well-known technology that provides services (private and public) such as easy internet access to data, applications, and files; users of this technology only pay according to the resources theyuse.Cloud Computing enhances the running of businesses worldwide with computing technology byremoving the cost which organisations would have spent to establish on-premise distributed computing architecture, as well as the cost of personnel and maintenance. Users are able to access different computing services on demand where resources are allocated to them in a flexible way [1].

In the cloud environment, the optimisation of resources (Virtual Machines) can be achieved through a method termed load-balancing. This method involves the assignment of user requests, or tasks,to resources (Virtual Machines),to optimise resource utilisationwhile maintaining efficiency in terms of time for executing tasks and returning results to the user. Load balancing reduces delays in the transmission of data and also prevents overloading of nodes in the cloud environment which affects the quality of service in the cloud Data Centres [2].

Cloud user requests are usually analysed and transferred to a particular Data Centre, which is done by considering the current resources of that Data Centre. The transmission mechanism is to ensure that the Virtual Machines are not underloaded or overloaded; the goal is to keep them balanced in their handling of requests or loads. When a request is received, the distribution of this request to an appropriate Virtual Machine is achieved through load balancing; Cloud computing uses virtualization techniques to achieve this aim[3].

The traditional load-balancing techniques are categorised into three main types. These techniques are either implemented on the Data Centre end side, or, on the service broker policy (user end side)[4]. The first and second types are the static load balancing algorithms and the dynamic load balancing algorithms respectively. While the processes of the static algorithms requires prior knowledge of the cloud environment, thus, the processing power, the storage capacity and the memory, the dynamic algorithms are able to allocate resources to tasks based on real-time characteristics of the requests as well as the resources. Even though the dynamic algorithms are complex, they present flexibility benefits which make them perform better, and are preferable to the static algorithms[5]. The third category of algorithms are Nature-inspired, which are intelligent algorithms that perform better for dynamic and complex systems, by mimicking how bees locate honey which is used as a search method in genetic algorithmic processes[6].

Response Time is one of the most important metrics by which the effectiveness of a load-balancing algorithm is measured. The Response Time is the time it takes the algorithm to send a response to a user's request, which considers the waiting, service and transmission time. For optimising the performance and effectiveness of a load-balancing algorithm, a minimal Response Time must be achieved.

2. RELATED WORKS

The concept of load balancing has been an integral part of distributed systems which evolved from simple to more advanced techniques for managing load across several servers through various research efforts.The dynamic techniques used for load balancingare able to track and distribute load uninterruptedly to Virtual Machines, while the non-dynamic techniques use predefined rules which are unable to consider real-time load requirements to distribute load to the Virtual Machines. Client-side techniques are designed for delegating the workload balancing logic to the client side, enabling clients to choose the most suitable server based on factors like response time or server availability, while the server-side load balancers are usually positioned between the clients and the servers for assigning incoming requests to multiple servers based on server capacity or current load [7].

The Throttled load balancing technique is one of the known older techniques that has been useful and it is still being enhanced by researchers for more efficiency. It was developed to dynamically manage load on Virtual Machines to achieve improved resource utilisation and efficient response time[8]. Researchers have pointed out that, the Throttled algorithm does not effectively distribute load uniformly to Virtual Machines which sometimes results in overloading or underloading which in effect defeats its target of resource utilisation; a Virtual Machine is usually allocated based on its availability without considering its capacity and the requirements of the request being made for processing. The efficiency of the Throttled algorithm is also sometimes diminished due to its high overheads for checking for idle Virtual Machines[9]. On account of the weaknesses of the Throttled algorithm, several researchers have presented modifications to enhance its efficiency and effectiveness.

Ghosh and Banerjee proposed a modified Throttled algorithmwith a priority technique which uses a switching queue to hold requests that are temporarily removed from the Virtual Machine due to the arrival of requests with higher priority. This algorithm improved the execution timevia thedefiningof

priorities of requestson which basisthey are assigned toservers[10]. The drawback of this method is that, requests with lower priorities may queue for an undue longer time before they get executedif there are too many high-priority requests, or, if new arriving requests are often defined with high-priority.

Sachdeva and Kakkar also presented a hybrid algorithm which combines some advantages of both the Equally Spread Current Execution (ESCE) and the Throttled algorithm. It uses a HashMap to keep client requests. It scans for a Virtual Machine with the least load to assign the request to it (when the Virtual Machines are busy). This hybrid algorithm minimises the Response Time of the cloud environment [11].

Some research efforts have been made to target the problem of managing load for geographically dispersed data centres where a component-based Throttled algorithm uses closest data centre service brooker policy to dynamically handle load on resources to achieve better response time, data processing time and resource utilisation[8]. Other researchers have also modified the Throttle algorithm for avoiding data loss over the cloud [12] among many research efforts. Even though there are several research that have contributed to enhancing the efficiency of the Throttled algorithm, there are still gaps that are being explored to resolve the current unique issues to further improve the response and data centre processing time.

The Round Robin algorithm is another widely used algorithm for balancing load in Cloud Computing, where requests are evenly assigned to resources in a pool iteratively. Virtual Machines with similar characteristics such as hardware and software configuration are put in a pool to be allocated for user requests. The requests are distributed to the machines in a sequential order until the last one, and then loops back to the beginning to start another series of assigning tasks to the resources. This results in the resources being assigned an equal number ofrequests (averagely). The major challenge with this technique is that, a resource when overloaded, will continue to be allocated for requests along with less loaded resources in the pool. To this end, researchers have proposed various techniques to enhance the performance of the algorithm[13].

Some researchers proposed their enhancement techniques based on the use of the genetic algorithm which mimics the evolution of nature. The algorithm uses a HashMap to maintain the states (*available* or *busy*) of Virtual Machines, whereit scans through the HashMap from top to bottom for any available Virtual Machine to be allocated forhandlinga request. For every request, the Virtual Machine chosen at random is compared with the previously routed Virtual Machine before the allocation is finally done. The comparison is done to distribute the request to the Virtual Machine with the shortest request queue length among the available ones[14].

Other researchers have also made modification efforts for the Round Robin algorithm by introducing dynamic quantum allocation [15];dynamically computingweighting for resources and implementing them on software-defined networks using Pox controller [16];dynamically computing weights for resources based on their runtime performance for preventing continuous centralized scheduling and ensuring more efficient load balancing[17]; defining priorities and computing weights for tasks and then assigning them to resources based on the priorities and weights while maintaining a mechanism to migrate tasks on overloaded resources to underloaded ones[18]. These among many modifications have been proposed to resolve the general problem of the Round Robin algorithm of not being efficient for resource utilisation, in some cases inefficiency in terms of response time, imbalance loading, inability to adapt to the performance of Virtual Machines as well as the requirements of requests among other gaps. Even though these modifications have also presented some positive contributions to some targeted issues, there is still the need to enhance the response time of the Round Robin algorithm as well as the Data Centre processing time, to which end this research is

conducted, and provides a novel mechanism that considers the requirements of requests of users, as well as the availability of Virtual Machines for effective load balancing with less response time and Data Centre processing time.

3. METHODOLOGY

3.1 The Algorithm Design

Load balancers are usually designed to target CPU, Severs, Networks, Tasks, Virtual Machines, etc. The targeted domain of a balancer greatly dictates its key features and logic for decision-making as well as its requirements and constraints. In this study, the algorithmic efforts are targeted at balancing load on Virtual Machines, where requests are intended to be effectively distributed to servers in a manner that optimises the Response Time and the Data Centre Processing Time.

The ReT-ELBa algorithm was designed following a rule-based approach. This approach was chosen to establish novel defined rules for the balancer's intended efficiency. The algorithm employs task scheduling and resource allocation techniques, where tasks are assigned to computing resources, while the computing resources are monitored and maintained in three lists labelled as primary, secondary and engaged for the execution of the requests made. The level of activity in each Virtual Machine as well as the requirements of a request or task are also monitored. This monitoring effort feeds into the logic of the algorithm for the appropriate scheduling of tasks and allocation of resources.

3.2 The Key Parameters

Load balancers are usually not one-size-fits-all solutions for solving unbalanced load situations in Cloud Computing. They are mostly heuristic in nature for establishing the most optimised use cases or scenarios, based on what key parameters are made the focus for improvement. In this research, the optimisation of the Response Time and the Data Centre Processing Time of the ReT-ELBa are the parameters targeted for assessing its efficiency. Two key things (among others) affect the response of load balancers; the load on a server and the requirement or complexity of requests. Based on these two effects, the ReT-ELBa considers the availability and load of Virtual Machines, and the requirements of a request for allocations. The goal of the algorithm which is to ensure that requests are being responded to quickly in a very efficient way, is achieved via the monitoring and optimisation of the Response Time as well as the Data Centre Processing Time.

3.3 Decision-making Logic and Work Load Distribution Technique

The decision-making logic of the ReT-ELBa is based on the optimisation of Response Time and resource utilisation load balancing policy, where thresholds are set for the request made to be executed on the Virtual Machines. A trigger condition is defined to offset the loads that are assigned to the Virtual Machines which are maintained and monitored in the primary and secondary lists. This analysis takes a real-time dynamic approach where new requests are assessed together with the current state of the Virtual Machines to make quick decisions to maintain a short Response Time.

The technique employed for the workload distribution is content-based, where the request is assessed for its content as well as the characteristics of the content. This establishes whether the request requires more or less resources to be executed; this forms a crucial layer for the proposed balancer to make its decisions for assigning loads to Virtual Machines.

4. THE ReT-ELBa

The ReT-ELBa maintains available Virtual Machines v in a primary list pl, where client requests r are maintained in a list rl scheduled for some v. The function $get(r)$ retrieves requests to be assigned to, and processed by some v via $v \rightarrow process(r)$; v is retrieved by the function $get(v)$. The pl must be a non-empty list for the assignment and process or r to be possible. The Virtual Machines in the pl that get engaged are maintained in a separate list λ, where their percentage is computed with the function $p(\lambda)$; this is used to check if the defined threshold t_v for engaged Virtual machines in pl is reached or otherwise. The engaged Virtual Machines are further assessed to establish if the requests they are engaged with are bigger requests $b(r)$ or smaller requests $s(r)$. When the Virtual Machines that are engaged with bigger requests exceed the defined threshold t_r, the non-engaged Virtual Machines will be maintained in a secondary list sl and allocated for processing smaller requests only until the engaged Virtual Machines become less than the defined threshold.

Algorithm 1: The ReT-ELBa Algorithm

1. $pl = \{v: 1 \leq v \leq n\}, where v \in pl$
2. $sl = \{v: 0 \leq v \leq n\}, where v \in sl$
3. $rl = \{r: 0 \leq r \leq m\}, where r \in rl$
4. $tv = 0.7$
5. $tr = 0.5$
6. $get(v)$
7. $get(r)$
8. $while(pl \vee 1)$
9. $process(r) \rightarrow v, for v \in pl$
10. $endwhile$
11. $p(\lambda) = 100 - \left(\frac{sl}{pl} X100\right)$
12. $sl = pl - \lambda$
13. $If p(\lambda) \geq (tvxpl)$
14. $If b(r) \geq (trx\lambda)$
15. $sl = pl - \lambda$
16. $process\big(s(r)\big) \rightarrow v, for v \in sl$
17. $endif$
18. end

In other words, when 70% of the total Virtual Machines on the primary list are used and 50% of the used Virtual Machines are as a result of bigger requests, the remaining number of Virtual Machines in the primary list are then used to create a secondary list of Virtual Machines for the processing of only smaller client request. Until the designated threshold is reached, the primary list of Virtual Machines will keep processing client requests.

5. EXPERIMENTAL RESULTS AND DISCUSSION

The experimental results obtained from the simulation of the ReT-ELBa were compared with the experimental results of the Throttled and Round Robin algorithms, where their corresponding

average, minimum and maximum Response Times in milliseconds as well as Data Centre Processing Time were evaluated; this was done to assert the efficiency of ReT-ELBa.

5.1 Case 1: Using 50 Virtual Machines with one Data Centre

For all the simulations, the ReT-ELBa, the Round Robin algorithm and the Throttled algorithm were analysed. In this case, 50 Virtual Machines from one Data Centre were tested and their corresponding Response Time were recorded. It turned out that the ReT-ELBa presents less Response Time than the other two algorithms with an average time of 249.25ms. The Response Time for the Throttled recorded an average of 249.26ms, and 271.51ms for the Round Robin algorithm (see Table 1).

Even though the ReT-ELBa performed better, it did not present a significant optimised Response Time compared to the Throttled algorithm. The Throttle does quite well in Response Time because it balances the workload by using two index tables of Virtual Machines, which enables it to utilise resources effectively. However, the Throttled algorithm always has to scan through the entire table for available or idle Virtual Machines to process a request. The downside of this is that the scanning may increase the Response Time in cases the idle Virtual Machine needed to process the request is at the bottom of the table.

Table 1: Response Time for Case 1

Algorithm	Average (ms)	Minimum (ms)	Maximum (ms)
Throttled	249.26	40.30	642.62
Round Robin	271.51	40.46	642.62
ReT-ELBa	249.25	40.29	642.62

The simulations were also done to assess the Data Centre processing time, which is the time it takes a Data Centre to process a request. The simulation results for Case 1 indicate that it takes 44.65ms on average to process a request by the Round Robin algorithm, whereas both the Throttled and the ReT-ELBa algorithms each recorded 22.57ms on average to do the same. Again, the ReT-ELBa appears to have a similar performance to the Throttled algorithm, while the Round Robin performs worst.

Table 2: Data Centre Processing Time for Case 1

Algorithm	Average (ms)	Minimum (ms)	Maximum (ms)
Throttled	22.57	0.12	102.27
Round Robin	44.65	0.12	117.55
ReT-ELBa	22.57	0.12	102.27

5.2 Case 2: 50 Virtual Machines for 2 Data Centres (25 each)

In this case, we simulate for a smaller number of Virtual Machines in two Data Centres. It is observed that (Table 3) the ReT-ELBa maintains a little better performance than the Throttled algorithm, while the Round Robin still lags in terms of Response Time. A similar performance is also recorded for the Data Centre Processing Time (Table 4).

Table 3: Response Time for Case 2

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	234.89	38.33	643.12

Round Robin	256.82	38.33	643.12
ReT-ELBa	234.86	38.33	643.12

Table 4: Data Centre Processing Time for Case 2

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	22.40	0.13	102.75
Round Robin	44.17	0.13	111.09
ReT-ELBa	22.38	0.13	102.75

5.3 Case 3: 100 Virtual Machines for 2 Data Centres (50 each)

The number of Virtual Machines were increased for two Data Centres, for this simulation. The ReT-ELBa continues to record the least Response Time, with an average of 234.86ms having to balance load not only within a Data Center but across other Data Centres as compared to the Throttled and the Round Robin, even though the Round Robin continues to be the worst performing algorithm among the three. The Throttled algorithm however outperforms the ReT-ELBa with difference of 0.07ms for the Data Centre Processing Time for this case.

Table 5: Response Time for Case 3

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	237.30	39.32	930.13
Round Robin	259.21	38.43	930.13
ReT-ELBa	237.24	39.32	930.13

Table 6: Data Centre Processing Time for Case 3

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	22.50	0.12	103.08
Round Robin	44.13	0.12	112.77
ReT-ELBa	22.57	0.12	103.88

5.4 Case 4: 150 Virtual Machines for 3 Data Centres (50 in each Data Centre)

The number of Virtual Machines were further increased for three Data Centres where we note the consistent performance differences among the three algorithms for Response Time and the Data Centre Processing Time as seen for Case 1 and Case 2. The ReT-ELBa outperforms the Throttled and the Round Robin algorithms in this case as well.

Table 7: Response Time for Case 4

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	136.66	37.85	390.25
Round Robin	163.34	38.21	388.62
ReT-ELBa	136.64	37.85	386.62

Table 8: Data Centre Processing Time for Case 3

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	27.14	0.12	103.18
Round Robin	53.55	0.12	112.78
ReT-ELBa	27.13	0.12	103.18

5.5 Performance Evaluation of ReT-ELBa

Generally, the ReT-ELBa showed similar performance with the Throttled algorithm for Response Time, even though the ReT-ELBa performed better in all four cases. The ReT-ELBa in one instance (Case 3) performed a little less than the Throttled algorithm for Data Centre Processing Time as indicated earlier. The three algorithms did not show any significant differences in the minimum and maximum recordings for both Response Time and Data Processing Time.

Figure 1 presents an interesting pattern for the Response Time in Case 4. It is expected that as the Data Centres increase the Response Time should also increase in direct proportionality. However, it is observed that, Case 4 records the lowest Response Time for the three algorithms. The feature accounting for this phenomenon is the fact that, the high number of Virtual Machines offers proximity and more availability for requests to be scheduled, it also lessens the number of engaged machines which makes it reach the threshold slowly. Availability translate into quick finding and assigning of requests and hence the pattern we see in the results.

As illustrated in Figure 2, as the number of Data Centres increases, the Data Centre Processing Time also increases from one case to the other. So, algorithms with better Data Centre Processing Time are expected to have a better Response Time and vice versa.

In Case 1, both the ReT-ELBa and the Throttled algorithms recorded the same Data Centre processing values of 22.57ms as compared to the Round Robin algorithm's 44.65ms. The Round Robin algorithm appear to use about twice as much time as used by the ReT-ELBa and the Throttled algorithm. A similar observation can be made from the experimental results for Case 2, Case 3 and Case 4.

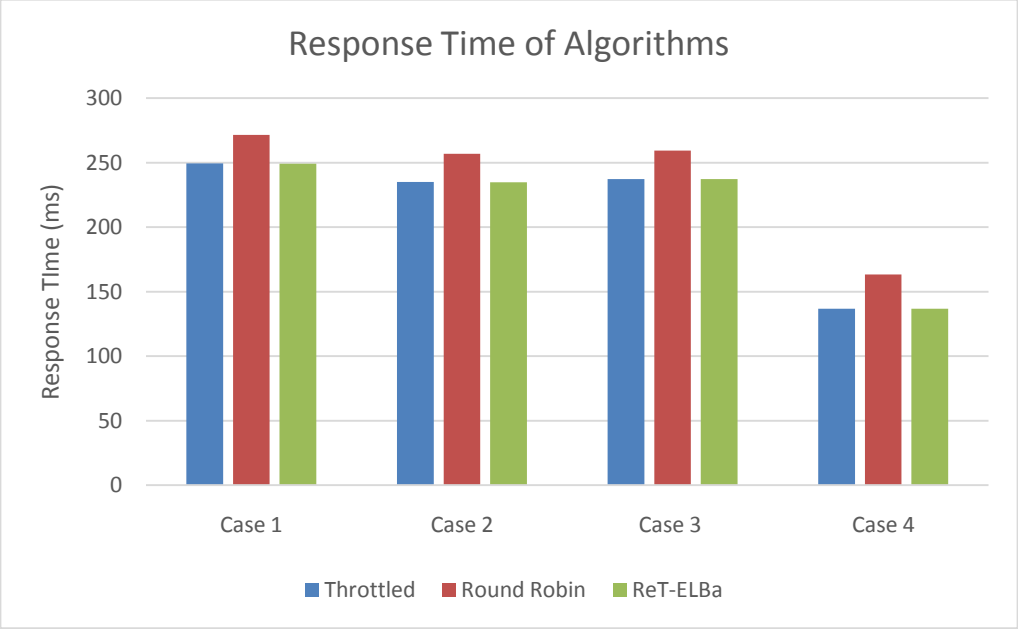

Figure 1: Performance Analysis of Response Time

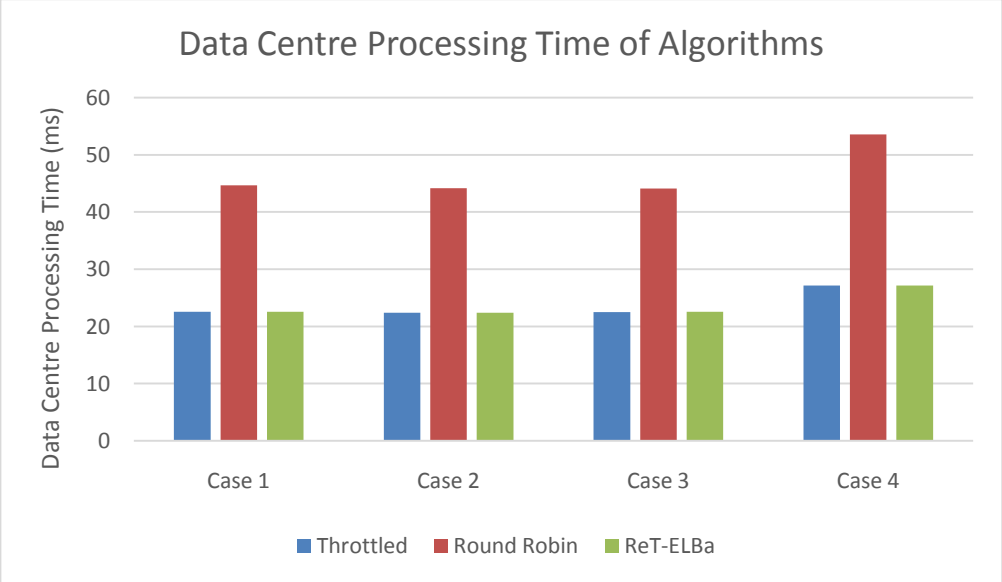

Figure 2: Performance Analysis of Data Centre Processing Time

6. CONCLUSIONS

This research presents a novel algorithm for optimising the Response and Data Centre Processing Time in comparison with the Throttled and Round Robin algorithms. The ReT-ELBa consistently outperformed the two algorithms with a not-too-significant difference from the performance of the Throttled algorithm. It however significantly outperforms the round Round Robin algorithm for both Response Time and Data Centre Processing Time.

Four simulation cases were completed to assess the performance of the algorithms for Virtual Machines in one Data Centre, as well as in multiple Data Centres with increasing number of Virtual machines, to assert if the performance of the algorithms would be affected by the variations. It turned

out that such variations had no significant impact of how the algorithms performed in terms of their Response Time and Data Centre Processing Time.

Accounting for the ReT-ELBa's performance via its logics is its ability to track the requirements of requests (whether it is big or small, or, needing more resources for execution), the current load and type of load on Virtual machines, in order to decide how to balance new load for existing Virtual Machines. The mechanism of maintaining a list of primary,secondary(idle) and engaged machines, makes it easy for the ReT-ELBa to switch loading to each category differently according to its state.

The future works of this research may consider advanced load-balancing techniques by incorporating Machine Learning methods to predict future load and likely resourcesto be allocated given the overall characteristics of request, policies and Virtual Machines. This will be done to further optimise the resources utilisation and response time for balancing load in Cloud Computing.

References

[1] A. S. N and M. Hemalatha, 'An Approach on Semi-Distributed Load Balancing Algorithm for Cloud Computing System', *International Journal of Computer Applications*, vol. 56, no. 12, pp. 5–10, Oct. 2012.

[2] R. Kaur, 'Load Balancing in Cloud System using Max Min and Min Min Algorithm', *International Journal of Computer Applications*.

[3] S. Hiranwal and D. K. C. Roy, 'Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice', vol. 2, no. 2, 2011.

[4] R. Choudhary and D. A. Kothari, 'A Novel Technique for Load Balancing in Cloud Computing Environment', vol. 6, no. 6, 2018.

[5] M. A. and Z. A. Khan, 'Issues and Challenges of Load Balancing Algorithm in Cloud Computing Environment', *INDJST*, vol. 10, no. 25, pp. 1–12, Jul. 2017, doi: 10.17485/ijst/2017/v10i25/105688.

[6] A. Thakur and M. S. Goraya, 'A taxonomic survey on load balancing in cloud', *Journal of Network and Computer Applications*, vol. 98, pp. 43–57, Nov. 2017, doi: 10.1016/j.jnca.2017.08.020.

[7] Majd Mutaher Hasan and Nabel Alsohaibi, 'A Review on Load Balancing Algorithms in Cloud Computing', *IJCSMC*, vol. 11, no. 11, pp. 219–236, Nov. 2022, doi: : https://doi.org/10.47760/ijcsmc.2022.v11i11.019.

[8] D. Mekonnen, A. Megersa, R. K. Sharma, and D. P. Sharma, 'Designing a Component-Based Throttled Load Balancing Algorithm for Cloud Data Centers', *Mathematical Problems in Engineering*, vol. 2022, pp. 1–12, Oct. 2022, doi: 10.1155/2022/4640443.

[9] N. G. Elnagar, G. F. Elkabbany, A. A. Al-Awamry, and M. B. Abdelhalim, 'Simulation and performance assessment of a modified throttled load balancing algorithm in cloud computing environment', *IJECE*, vol. 12, no. 2, p. 2087, Apr. 2022, doi: 10.11591/ijece.v12i2.pp2087-2096.

[10] S. Ghosh and C. Banerjee, 'Priority based Modified Throttled Algorithm in Cloud Computing', in *2016 International Conference on Inventive Computation Technologies (ICICT)*, Aug. 2016, pp. 1–6. doi: 10.1109/INVENTIVE.2016.7830175.

[11] P.G Deptt of Computer Science, Dev Samaj College for Women, Ferozepur, Punjab, India, R. Sachdeva, S. Kakkar, and P.G Deptt of Computer Science Dev Samaj College of Women, Ferozepur, Punjab, India, 'A Novel Approach in Cloud Computing for Load Balancing Using Composite Algorithms', *IJARCSSE*, vol. 7, no. 2, pp. 51–56, Feb. 2017, doi: 10.23956/ijarcsse/V7I2/0119.
[12] F. T. Johora, I. Ahmed, M. A. I. Shajal, and R. Chowdhory, 'A load balancing strategy for reducing data loss risk on cloud using remodified throttled algorithm', *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 3, Art. no. 3, Jun. 2022, doi: 10.11591/ijece.v12i3.pp3217-3225.
[13] 'Load Balancing in Cloud Computing using Round Robin Algorithm', *International Journal of Engineering Research*.
[14] D. A. Shafiq, N. Z. Jhanjhi, and A. Abdullah, 'Load balancing techniques in cloud computing environment: A review', *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 3910–3933, Jul. 2022, doi: 10.1016/j.jksuci.2021.02.007.
[15] K. Surendhar, L. S. Chamarthy, D. Priyadharshini, G. Keerthana, G. Sinha, and A. V. Kumar, 'Enhancement of Round Robin Algorithm with Dynamic Quantum in Cloud Computing', in *2023 International Conference on Inventive Computation Technologies (ICICT)*, Apr. 2023, pp. 799–803. doi: 10.1109/ICICT57646.2023.10133995.
[16] B. Manasa and A. R. Babu, 'Dynamic Weighted Round Robin Approach in Software-Defined Networks Using Pox Controller', *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, no. 5, Art. no. 5, May 2023, doi: 10.17762/ijritcc.v11i5.6618.
[17] C. Gao and H. Wu, 'An Improved Dynamic Smooth Weighted Round-robin Load-balancing Algorithm', *J. Phys.: Conf. Ser.*, vol. 2404, no. 1, p. 012047, Dec. 2022, doi: 10.1088/1742-6596/2404/1/012047.
[18] A. Katangur, S. Akkaladevi, and S. Vivekanandhan, 'Priority Weighted Round Robin Algorithm for Load Balancing in the Cloud', in *2022 IEEE 7th International Conference on Smart Cloud (SmartCloud)*, Oct. 2022, pp. 230–235. doi: 10.1109/SmartCloud55982.2022.00044.