International Journal of Electrical and Computer Engineering (IJECE)
Vol. 15, No. 3, June 2025, pp. 3171~3179
ISSN: 2088-8708, DOI: 10.11591/ijece.v15i3.pp3171-3179

Q-learning based active monitoring with weighted least connection round robin load balancing principle for serverless computing

Yashwanth Balan, Arokia Paul Rajan
Department of Computer Science, Christ University, Bangalore, India

Article Info

Article history:

Received Sep 2, 2024
Revised Jan 17, 2025
Accepted Mar 3, 2025

Keywords:

Artificial intelligence
Load balancing
Q-learning
Reinforcement learning
Serverless computing

ABSTRACT

Serverless computing is considered one of the most promising technologies for real-time applications, with function as a service (FaaS) managing service requests in serverless computing. Load balancing played a vital role in assigning tasks in serverless computing for customers; user requests were controlled by load balancing algorithms and managed using machine learning techniques to deliver results and performance metrics within specified time limits. All serverless computing applications aimed to achieve optimal performance based on the most effective load balancing techniques, which directed requests to the appropriate servers in a timely manner. This research focused on developing a novel Q-learning based active monitoring with least connection round robin load balancing principle (Q-LAMWLR LB) for serverless computing to address the aforementioned challenge. Also, aimed to intelligently assign requests to serverless computing based on the number of requests arriving at the load balancer and how intelligently they could be directed to the appropriate server. This work utilized standard techniques to calculate the average response time for each scheduling algorithm and develop a novel intelligent load-balancing technique in serverless computing. Required experiment were conducted and the results are giving the improvement as compared to other load balancing principles. The further research in this area also identified and presented.

Corresponding Author:

Yashwanth Balan
Department of Computer Science, Christ University
Bangalore, Karnataka 560029, India
Email: yashwanth.balan@res.christuniversity.in

1. INTRODUCTION

Artificial intelligence (AI) helps to make the software reliability more convenient with intelligent software to handle the situations and it works automatically. Aligning concepts from theory, practice and future perspective and consequently described the research area to provide an extra mile to understand the AI-based load balancing techniques. The systems distribute the workload according to the serverless schedulers based on the application characteristics. A schedular which can be used as centralized one to provide a parallel framework for the serverless computing using AWS lambda. Also, it is ideal to combine two approaches. The prime focus of this paper is to reduce the response time of the request sent by the user and to increase the performance of every request through machine learning load-balancing techniques [1]–[3]. Figure 1 presents the principle of intelligent load balancing in serverless. AI helps to make the software reliability more convenient with intelligent software to handle the situations and it works automatically. Serverless computing with load balancing is a challenging research problem mentioned in many research areas. Different load-balancing algorithms are proposed based on the load balancing in serverless computing. The problematic research areas are machine learning-based load-balancing techniques used in serverless computing. The research direction focuses on the advantage of strategic implementation to consider the managerial challenges. This research paper is structured as follows: section 2 comprises of related works and section 3 discusses the problem with underlying assumptions. A detailed introduction to the planned request scheduling principle is given in section 4. The experimental results and performance evaluation are presented in section 5, along with a comparison with a few current scheduling principles. Section 6 presents the conclusion and the future directions for this research work.

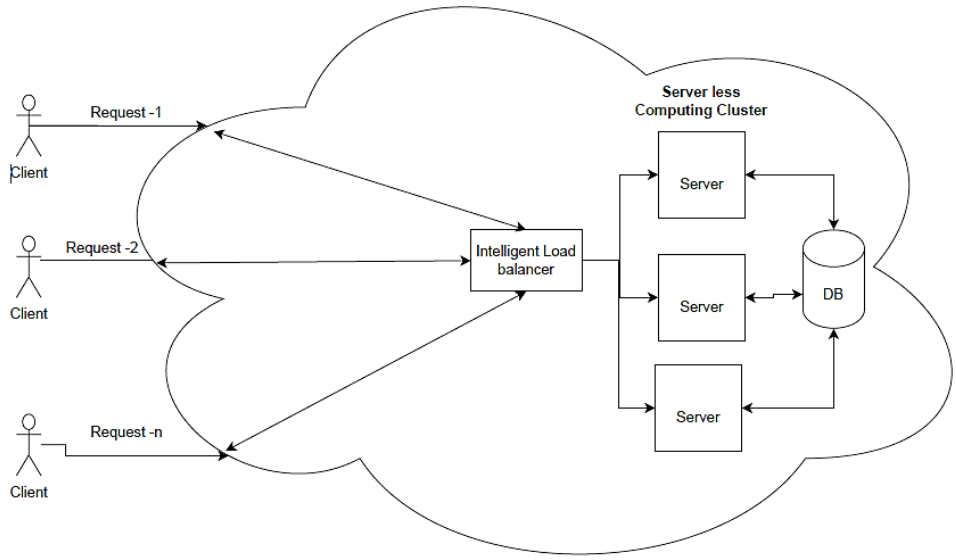

Figure 1. Intelligent load balancing techniques in serverless computing

2. METHOD

The modern world is entirely into internet, and users always avail information from internet. More users are helping the feature to achieve the desired technology in a better way. An appropriate communication system should be established to get a proper response to the request sent by the user. The intelligent load balancing techniques provide a platform for the above-mentioned problem. Smart load balancing techniques for the serverless computing method. In the serverless computing paradigm, the difficult part will be controlling the load received from the different users simultaneously. The response time accuracy of solving the request creates a high demand for intelligent load balancing [4]–[6]. There is a need for smart load balancing principles on serverless computing to ensure the load balancing serverless computing can reduce the response time and efficiently increase the availability of servers for the subsequent coming request. The serverless computing spins it up fresh and starts hosting the function through a cold start, and if the function is running successfully, it says warm start. If the function does not request anything again, it comes to the state of idle, and the following user will again be the cold start, which makes the time-space trade- off using serverless functions [7], [8]. Through that, the issue of load balancing can be solved to an extent. The load balancing is presented in Figure 2.

Q-learning active monitoring weighted least connection round robin load balancing (Q-LAMWLR LB) uses a reinforcement learning technique, and N number of requests arrives at the load balancer; it collects the server and function information at the earlier stage [9], [10]. The request from R1 to RN will be sent to the Q-LAMWLR LB methodology, and with the help of machine learning techniques associated with different load balancing algorithms, it will finalize the highest weight for the server T time an invocation needs to schedule and Q-LAMWLR LB will be having N clusters available in total. Q-LAMWLR LB receives a batch of the latest state Rt=(R1,. .., Rn,. .., RN) from the cluster, where n is the Nth available server. Once the data is received for the weight allotting procedure, Q- LAMWLR LB creates the index table with all the information hired, mainly the server and Function information [11], [12]. Allocating weight to the server is done with the help of reinforcement learning. The actor network and critic network are used to giving weight to the different servers. Figure 3 presents the placement of the designed load balancing principle specific to serverless computing.

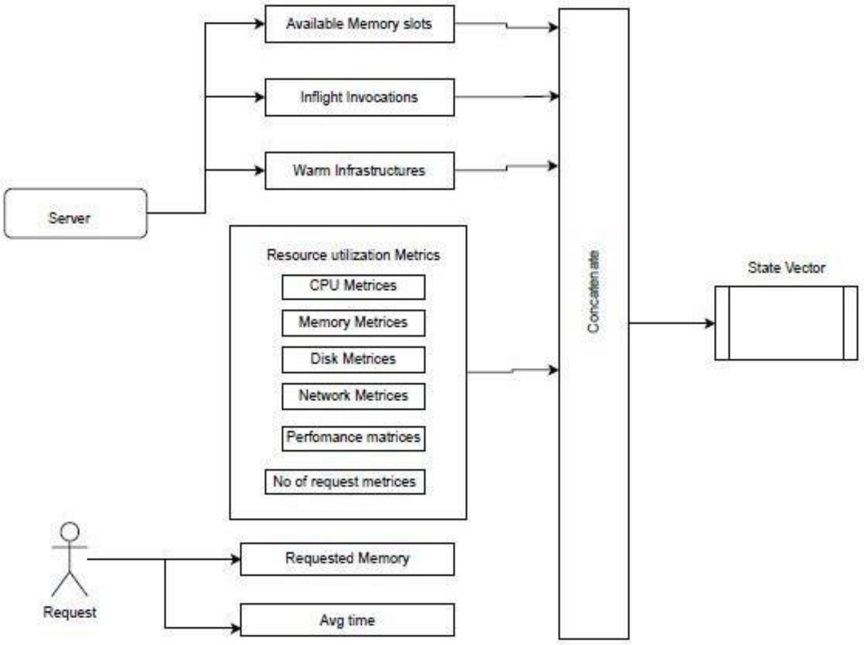

Figure 2. Identifying the right server using Q-LAMWLCRR LB

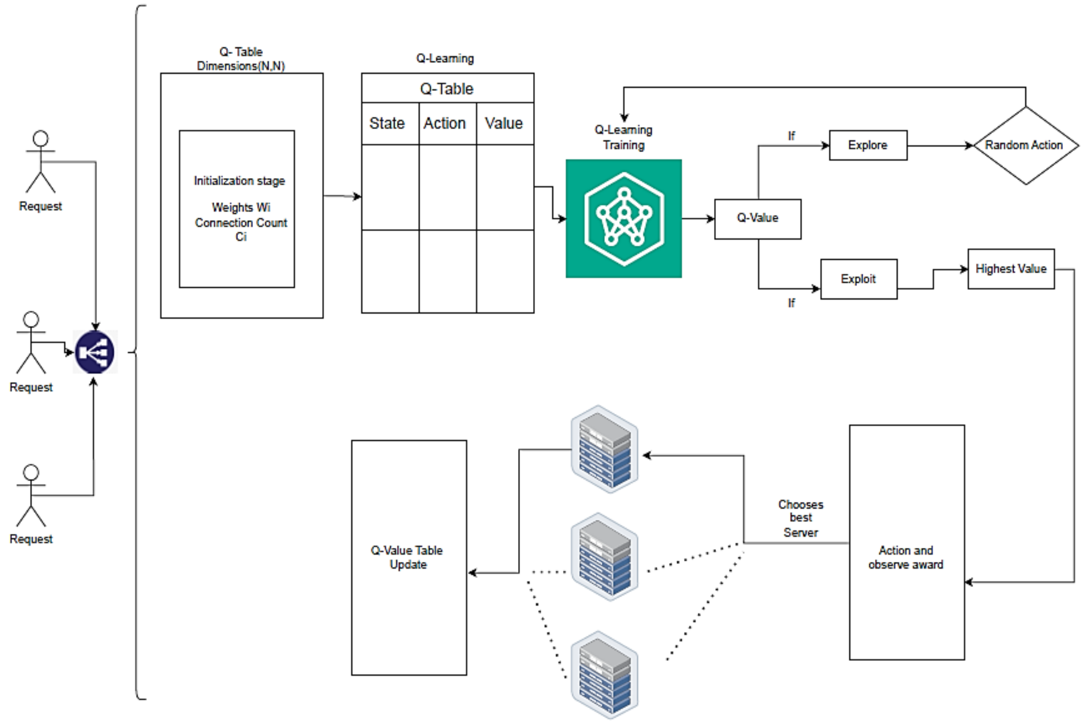

Figure 3. The Q-LAMWLR LB architecture

Algorithm 1 presents the algorithm represents the way of applying the Q-learning training with the best set of servers which is available and based on that the second algorithm will be provided and will find the best server for the request that is raised and an intelligent load balancer based on the Q-learning techniques and the other load balancing algorithms will be selecting the best server for the request [13]–[15]. The algorithm is about two conditions to get into the loop. One is if it explores the other is if its exploit. In the both condition the algorithms works and make a prompt attempt to get the result. The accurate values are to get the server in a average response time to choose a best server. If an agent chooses to explore (with probability ϵ), it evenly chooses a random action from among all options. If an agent chooses to exploit (with probability 1−ϵ), it determines the highest value. These are the condition which accepts with the algorithm. If the agent which relate with the Q-learning technique chooses to explore the probabiility of values and chooses a random action to exploit the same.

Algorithm 1. Finding the best server using Q-learning

Algorithm: Q-learning training based on the request

begin

Step 1: The load balancer receives a request.

Step 2: Determine which serverless Function is best for the request

S: List of serverless functions that are accessible.

N: The number of serverless features in S Wi: The weight of the serverless Function is i

Ci: The Connection count for serverless Function i

Q (i, j): The Q-value that indicates the acquired usefulness of changing from state i to state j.

Step 3: Create a Q-table of dimensions (N, N) and set all of its values to zero at first. Step 4: Initialize weights Wi for each serverless Function

Step 5: Initialize connection counts Ci for each serverless Function

Step 6 : Q-learning Training

Initialize state s and Choose action a
$$\begin{cases} \frac{\epsilon}{|\mathcal{A}|} + (1-\epsilon).\frac{\mathcal{Q}(s,a)}{\sum_{a`}\mathcal{Q}(s,a`)} & if\ explore \quad (1) \\ \frac{\mathcal{Q}(s,a)}{\sum_{a`}\mathcal{Q}(s,a`)} & if\ exploit \quad (2) \end{cases}$$

π(a|s): The possibility of picking a method of action a in state s.

|A|: The total amount of possible outcomes.

Q (s, a): The Q-value for state-action pair (s, a). ϵ: The exploration parameter (0 < ϵ ≤ 1)

If an agent chooses to explore (with probability ϵ), it evenly chooses a random action from among all options. If an agent chooses to exploit (with probability 1−ϵ), it determines the highest value.

/ End of Q-learning Training */*

Output: Q-learning table updated

The second algorithm presented in algorithm 2 gives a complete explanation of how Q-LAMWLR LB works [16], [17]. Based on the training of q-learning, the load balancer understands each request that arrives at the same or different interval of time and can be assigned to the server, which is available at a particular time based on the Q-values calculated using the algorithm [18], [19]. The algorithm provides more clarity to understand the working procedure in a detailed method. The algorithms give a complete explanation of how Q-LAMWLR LB works [20]–[22]. Based on the training of q-learning, the load balancer understands each request that arrives at the same or different interval of time and can be assigned to the server, which is available at a particular time based on the Q-values calculated using the algorithm. The algorithm checks the possible way of assigning the request to the proper server with the arrival of proper request to the algorithm.

Algorithm 2. Update Q-Value: For the current state-action pair, update the Q-value

Algorithm: Allocating the request to the server

begin

Step 1: The Q-value update for a state-action pair (s, a)

$$Q(s,a) \leftarrow Q(s,a)\cdot(1-\alpha) + (\gamma\cdot \max_{a'} Q(s',a') + R(s,a)))\cdot\alpha \quad (3)$$

Q (s, a): state action pair Q-value(s,a).

α: (0 < α ≤ 1) rate of learning

R (s, a): Action a reward point

γ: (0 < γ ≤ 1) the discounted rate

maxa′Q (s′, a′): Maximum Q-value for the next state’s′

Step2: Next function index=(i+1) mod N

Step 3: Selected function index=arg mini ($\frac{Ci}{wi}$)

Step 4: New Weight=Function of Current Weight, Performance Metrics

Step 5: New Q-value=Function of Current Q-value, Performance Metrics

Step 6: Transition to the next state: Using load balancing strategy

Step 7: Based on weights, determine the subsequent serverless Function.

Step 8: Considering the weights, choose the serverless Function with the fewest active connections.

Step 9: Track the performance indicators of serverless functions.

Step 10: To respond to changing conditions, adjust weights and Q-values based on the results of active monitoring

Step 11: Repeat from (d.ii) to (f)

Step 12: Proceed with the procedure for the predetermined number of times or until convergence.

end

/ End of Server allocation algorithm */*

Output: Request allocated to the server

Table 1 shows the performance of how Q-LAMWLR LB with different sets of processes and assigns the method based on the arrival, burst, and exit time to the proper server. Figure 3 already presented how Q-LAMWLR LB principle and shows the working procedure of the load balancer when a request reaches the load balancer: i) based on the request reaches the server assigns and ii) with the help of different algorithm the comparison happening. The comparison gives the idea to understand the algorithm which is provided has an impact based on the request receives.

The proposed methodology combines different load balancing algorithm with the machine learning algorithm Q-learning methodology from reinforcement learning. This approach addresses the challenges of serverless computing based on the number of requests arrives and the algorithm treats all the request in the effective way and assign to the server based on the arrival and burst time. The proposed methodology identifies the average time of the request based on the arrival and burst time. It helps the load balancer to identify the server and allocate it. The methodology increases the adaptability leveraging the real time feed back monitoring to refine system server and utilize the resources [23]–[25]. Through the real world testing and comprehensive simulation through the proposed methodology it can increase the scalability and reduce the response time to an extent while the requests are allocated efficiently to the server which is available. This proposed methodology of using Q-learning method used by the load balancer to allocate the server in the serverless computing improves the performance, reliability and cost management in the serverless computing. The algorithm completely focuses on reducing the response time and increase the scalability based on the arrival time and the burst time of the request. The approach clearly focuses on the areas mentioned in the proposed methodology.

Table 1. The performance of Q-LAMWLR LB

Process	Server	Arrival	Burst	Exit	Turn Around	Wait
P1	S1	3	4	7	4	0
P3	S2	3	5	8	5	0
P5	S1	4	3	7	3	0
P4	S3	6	7	13	7	0
P2	S3	6	8	21	15	7

3. RESULTS AND DISCUSSION

This research conducted experiments deploying Q-LAMWLR LB on two different CloudSim clusters: one on the Compute Canada Cloud and the other on an Amazon elastic compute cloud (AWS EC2) cluster. Eachcluster comprised 13 virtual machines (VMs) with specific roles: one VM for hosting the controllers like application program interface (API) gateway and Redis services for the back end, messages are distributed and database as well, one for the Q-LAMWLR LB agent, and the function invokers are 10. In the Canada cloud compute, each 32 GBs of memory for VM and vCPU cores are 8. 2 GBs of random access memory (RAM) for invokers for each function execution, based on function memory requirements CPU power is allocated proportionally. On the EC2 AWS cluster, each VM was of type c5d.2xlarge with 8 vCPU cores and 16 GBs of memory, launched as spot instances. Similar to the Compute Canada Cloud. Q-LAMWLR LB independently on both clusters and then evaluated its. Figure 4 mentions about the request distribution in the serverless computing using the Q-LAMWLCRR LB.

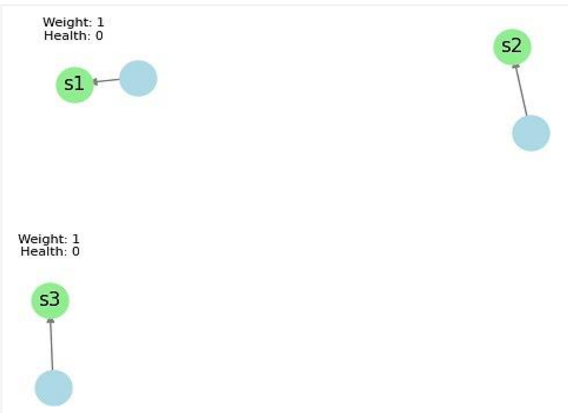

Figure 4. Request distribution to the servers using the Q-LAMWLCRR LB

3.1. Average response time

The Q-learning table will be updated for the Q-LAMWLR LB algorithm using the following algorithm based on the algorithm which is provided the average response time of accepting the request will be considered [26]–[28].

$$\begin{array}{l} \frac{\epsilon}{|A|} + (1-\epsilon)\cdot\frac{Q(S,a)}{\Sigma_{a`}Q(s,a`)}\ \ if\,explore \\ \frac{Q(s,a)}{\Sigma_{a`}Q(s,a`)}\ \ if\,exploit \end{array} \tag{4}$$

The equation (4) provides the updating of q-learning techniques and through the Q-learning update the server will be allotted and the request will be considered based on it. Once the request is treated to the server the average response time will be identified [29], [30]. The mathematical equation is to provide the final output of average responses time based on the request receives. Figure 5 showcases the result of comparison between the scheduling algorithms with the proposed algorithm in the research. Q-LAMWLR LB algorithm takes the less time as compared to the other algorithms. The comparison gives a clear picture of the proposed algorithm is working efficiently as compared to the other algorithms. The result is purely based on the request receives on the algorithm.

Figure 6 represents two parameters time interval on the X- axis and data interval on the Y axis. The figure compares the Canada cloud and AWS EC2 based on the algorithm the result produces in this format. The blue line makes the Canada cloud and which show cases the difference among the AWS EC2 in the different interval of time and how data is travelling according the time changes. The variation show cases the difference based on the time changes and the data interval.

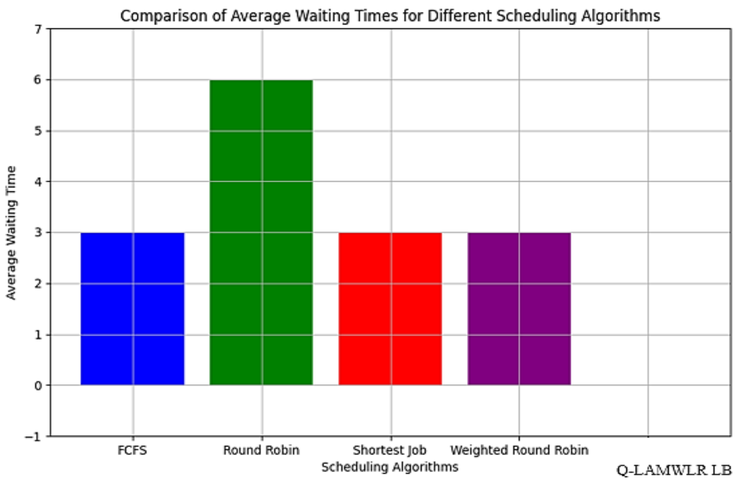

Figure 5. Comparison of the different algorithms with Q-LAMWLR LB

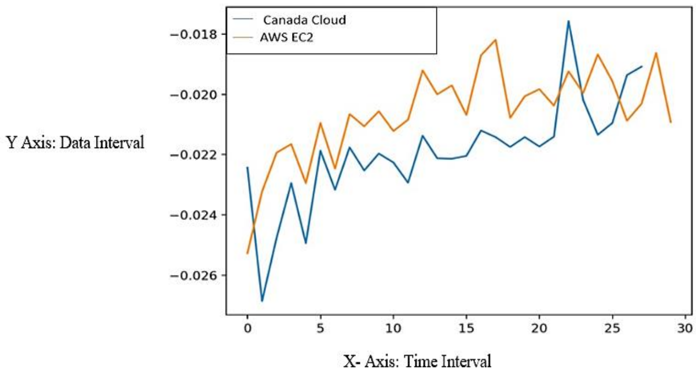

Figure 6. Comparison of the different algorithms with Q-LAMWLR LB

3.2. Average turnaround time

The turnaround time of the request is based on the arrival time and burst time of the algorithm. Based on these values the turnaround time will be calculated. The Algorithm 1 provides the data clearly. The equation provides the selection best server once the turnaround time is calculated [29].

$$Q\ (s, a)\ \leftarrow Q\ (s, a)\ \cdot\ (1 - \alpha)\ +\ (\gamma \cdot maxa'Q\ (s', a') + R\ (s, a)))\ \cdot\ \alpha \tag{5}$$

The Q-LAMWLR LB algorithm will be selecting the best server once the equation identies the server which is available at the moment based on the Q-learning table updating. Considering the Q-learning table data which is available the request will be considered and allots the server for the request [29], [30]. Figure 7 represents the comparison of different algorithm with the proposed algorithm to showcase the difference.

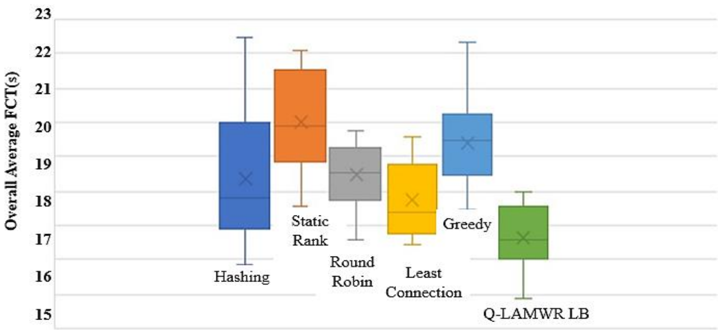

Figure 7. Comparison among different load balancing algorithms based on the Serverless environment

This work mainly focuses on the serverless computing platform to make an intelligent load balancer. The methodology describes the algorithm and which showcases the working pattern of the algorithm when the request arrives. The load balancer manages the request and turns into a Q-learning table to make the next request work promptly. The different instruments and the methods are used in the methodology are explained in the proposed methodology and the working procedure is in the experimental and discussion part. There is different type of algorithms compared to make a statement that Q-LAMWLR LB can be some kind of changes in the serverless. computing platform while using the algorithm. Based on the request arrives the servers are allotted and the next request will be perfectly allotted to the server based on the reinforcement learning.

4. CONCLUSION

The reinforcement active monitoring round-robin weighted least connection algorithm described in the introduction stage properly reduces the response time once the request arrives at the load balancer. The Q-LAMWLR LB principle finds the best server available through machine learning techniques using reinforcement learning in detail using the Q-learning technique. Q-learning techniques applied to the three different load balancing algorithms provide less response time to the request received at the load balancer-value table, which will be updated all the time after selecting the server for the request received. The algorithm's primary objective is satisfied through the result and discussion area as mentioned in the introduction. Every request's primary goal is to reduce the response time. Response time can be reduced through the proposed algorithms. The research challenges are to finalize the test bud to apply the algorithms in the real world. The future enhancement can be using machine learning techniques for serverless computing to make the server scalable through machine learning techniques. The research proposes a novel algorithm using Q-learning techniques of reinforcement learning applied to the load balancing concept using active monitoring load balancing, weighted round robin, and weighted least connection load balancing algorithm to satisfy the request which arrives at the load balancer to appropriately transfer to the serverless computing paradigm in a less response time.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Yashwanth Balan	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	
Arokia Paul Rajan		✓		✓				✓	✓	✓		✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, Y B, upon reasonable request.

REFERENCES

[1] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, “Resource management with deep reinforcement learning,” in *HotNets 2016 - Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, Nov. 2016, pp. 50–56, doi: 10.1145/3005745.3005750.

[2] S. Hong, A. Srivastava, W. Shambrook, and T. Dumitras, “Go serverless: securing cloud via serverless design patterns,” *10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18)*, Jul. 2018.

[3] S. Sinha, A. Hazarika, and S. Johari, “Performance evaluation of ai based load balancing algorithm (reinforcement learning) with other load balancing algorithms in a JPPF Grid: E.coli genome sequence alignment problem,” in *International Conference on Bioinformatics and Systems Biology, BSB 2018*, Oct. 2018, pp. 64–66, doi: 10.1109/BSB.2018.8770657.

[4] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, “The rise of serverless computing,” *Communications of the ACM*, vol. 62, no. 12, pp. 44–54, Nov. 2019, doi: 10.1145/3368454.

[5] D. Khatri, S. K. Khatri, and D. Mishra, “Potential Bottleneck and measuring performance of serverless computing: a literature study,” in *ICRITO 2020 - IEEE 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, Jun. 2020, pp. 161–164, doi: 10.1109/ICRITO48877.2020.9197837.

[6] H. B. Hassan, S. A. Barakat, and Q. I. Sarhan, “Survey on serverless computing,” *Journal of Cloud Computing*, vol. 10, no. 1, Jul. 2021, doi: 10.1186/s13677-021-00253-7.

[7] Y. Wu, T. T. A. Dinh, G. Hu, M. Zhang, Y. M. Chee, and B. C. Ooi, “Serverless data science-are we there yet? a case study of model serving,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Jun. 2022, pp. 1866–1875, doi: 10.1145/3514221.3517905.

[8] J. Scheuner and P. Leitner, “Function-as-a-service performance evaluation: a multivocal literature review,” *Journal of Systems and Software*, vol. 170, p. 110708, Dec. 2020, doi: 10.1016/j.jss.2020.110708.

[9] M. Sadaqat, R. Colomo-Palacios, L. Emil, and S. Knudsen, “Serverless computing: a multivocal literature review,” *NOKOBIT - Norsk Konferanse Organisasjoners Bruk AV Informasjonsteknologi*, 2018.

[10] D. Taibi, N. El Ioini, C. Pahl, and J. R. S. Niederkofler, “Patterns for serverless functions (function-as-a-service): a multivocal literature review,” in *CLOSER 2020 - Proceedings of the 10th International Conference on Cloud Computing and Services Science*, 2020, pp. 181–192, doi: 10.5220/0009578501810192.

[11] C. P. Filho *et al.*, “A systematic literature review on distributed machine learning in edge computing,” *Sensors*, vol. 22, no. 7, p. 2665, Mar. 2022, doi: 10.3390/s22072665.

[12] K. Sivaraman, R. M. V. Krishnan, B. Sundarraj, and S. Sri Gowthem, “Network failure detection and diagnosis by analyzing syslog and SNS data: Applying big data analysis to network operations,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 9 Special Issue 3, pp. 883–887, 2019, doi: 10.35940/ijitee.I3187.0789S319.

[13] L. Schuler, S. Jamil, and N. Kuhl, “AI-based resource allocation: Reinforcement learning for adaptive auto-scaling in serverless environments,” in *Proceedings - 21st IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2021*, May 2021, pp. 804–811, doi: 10.1109/CCGrid51090.2021.00098.

[14] R. Cordingly, W. Shu, and W. J. Lloyd, “Predicting performance and cost of serverless computing functions with SAAF,” in *Proceedings - IEEE 18th International Conference on Dependable, Autonomic and Secure Computing, IEEE 18th International Conference on Pervasive Intelligence and Computing, IEEE 6th International Conference on Cloud and Big Data Computing and IEEE 5th Cybe*, Aug. 2020, pp. 640–649, doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00111.

[15] W. Lloyd, S. Pallickara, O. David, M. Arabi, and K. Rojas, “Mitigating resource contention and heterogeneity in public clouds for

scientific modeling services," in *Proceedings - 2017 IEEE International Conference on Cloud Engineering, IC2E 2017*, Apr. 2017, pp. 159–166, doi: 10.1109/IC2E.2017.29.
[16] W. Lloyd, S. Pallickara, O. David, J. Lyon, M. Arabi, and K. Rojas, "Performance modeling to support multi-tier application deployment to infrastructure-as-a-service clouds," in *Proceedings - 2012 IEEE/ACM 5th International Conference on Utility and Cloud Computing, UCC 2012*, Nov. 2012, pp. 73–80, doi: 10.1109/UCC.2012.20.
[17] "Cloud Controls Matrix - Cloud Security Alliance: Cloud Security Alliance." Accessed: Apr. 13, 2024. [Online]. Available: https://cloudsecurityalliance.org/research/cloud-controls-matrix.
[18] I. Baldini *et al.*, "Serverless computing: Current trends and open problems," in *Research Advances in Cloud Computing*, Springer Singapore, 2017, pp. 1–20.
[19] K. Alpernas, C. Flanagan, S. Fouladi, L. Ryzhyk, M. Sagiv, and K. Winstein, "Secure serverless computing using dynamic information flow control," in *Proceedings of the ACM on Programming Languages*, Oct. 2018, vol. 2, no. OOPSLA, pp. 1–26, doi: 10.1145/3276488.
[20] "Apache Flink: Scalable stream and batch data processing," *Apache Software Foundation*. Accessed: Jun. 03, 2024. [Online]. Available: https://flink.apache.org/.
[21] K. Djemame, M. Parker, and D. Datsev, "Open-source serverless architectures: an evaluation of apache OpenWhisk," in *Proceedings - 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing, UCC 2020*, Dec. 2020, pp. 329–335, doi: 10.1109/UCC48980.2020.00052.
[22] A. Agache *et al.*, "Firecracker: Lightweight virtualization for serverless applications," in *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020*, 2020, pp. 419–434, doi: 10.5555/3388242.3388273.
[23] D. Du *et al.*, "Catalyzer: Sub-millisecond startup for serverless computing with initialization-less booting," in *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*, Mar. 2020, pp. 467–481, doi: 10.1145/3373376.3378512.
[24] "Apache OpenWhisk is an open-source serverless cloud platform," *Apache OpenWhisk*. Accessed: Jul. 02, 2024. [Online]. Available: https://openwhisk.apache.org/.
[25] T. Lynn, J. G. Mooney, B. Lee, · Patricia, and T. Endo, "The cloud-to-thing continuum: opportunities and challenges in cloud, fog and edge computing," *Palgrave Macmillan*, p. 163, 2020.
[26] A. J. Chaves, C. Martín, and M. Díaz, "The orchestration of machine learning frameworks with data streams and GPU acceleration in Kafka-ML: a deep-learning performance comparative," *Expert Systems*, vol. 41, no. 2, Mar. 2024, doi: 10.1111/exsy.13287.
[27] S. K. Mohanty, G. Premsankar, and M. Di Francesco, "An evaluation of open source serverless computing frameworks," in *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, Dec. 2018, vol. 2018-Decem, pp. 115–120, doi: 10.1109/CloudCom2018.2018.00033.
[28] A. Palade, A. Kazmi, and S. Clarke, "An evaluation of open source serverless computing frameworks support at the Edge," in *Proceedings - 2019 IEEE World Congress on Services, SERVICES 2019*, Jul. 2019, pp. 206–211, doi: 10.1109/SERVICES.2019.00057.
[29] A. Jindal, M. Gerndt, M. Chadha, V. Podolskiy, and P. Chen, "Function delivery network: Extending serverless computing for heterogeneous platforms," *Software - Practice and Experience*, vol. 51, no. 9, pp. 1936–1963, Mar. 2021, doi: 10.1002/spe.2966.
[30] D. Pinto, J. P. Dias, and H. S. Ferreira, "Dynamic allocation of serverless functions in IoT environments," in *Proceedings - 16th International Conference on Embedded and Ubiquitous Computing, EUC 2018*, Oct. 2018, pp. 1–8, doi: 10.1109/EUC.2018.00008.

BIOGRAPHIES OF AUTHORS

Yashwanth Balan received the MCA degree in computer application from Bangalore University, India, in 2014 and the BCA in computer applications from PES Institute Bangalore, India, in 2011. He is a lecture at the Department of Computer Science, Christ Academy. His research interests include serverless computing, cloud computing, robotic process automation, reinforcement learning, and artificial intelligence. Email-Yashwanth.balan@res.christuniversity.in.

Arokia Paul Rajan is currently working as associate professor, Department of Computer Science, Christ University, Bengaluru, India. He holds Ph.D. in Computer Science and Engineering from Pondicherry University, India. His research area is data management in cloud architectures. He published 11 research papers in international journals and conferences. Email-arokia.rajan@christuniversity.in.