International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN (Online): 3139-6577
Volume: 01 | Issue: 2 | June 2026

Workload-Aware Dynamic Load Balancer for Cold-Start Reduction in Serverless Function-as-a-Service Environments

Swati Jain[1*], John Clarke[2]
[1] Department of Mechanical Engineering, Mahatma Gandhi Govt. Engineering College, Jeori, Himachal Pradesh, India.
[2] School of Mathematics, University of Birmingham, Birmingham, United Kingdom.
[*]Corresponding Author: swatijainmain@gmail.com

Article History:

***Received**: 02-03-2026*

***Revised:** 08-04-2026*

***Accepted**: 21-05-2026*

Abstract

Function as-a-Service (FaaS) is serverless cloud computing model that automatically manage servers based on user requirements. However, due to this advancement there is problem called "Cold Start Latency" occurs. The cold-start latency bottleneck arises because FaaS send request to server without knowing server state i.e., warm or cold. Previous Load Balancer (LB) research has not mention functional instance life cycle and resulted in inappropriate routing of functional instances, increased latency, and increased costs. This work provides a detailed comparison of previous static LB and traditional dynamic LB models with a new state-aware instance allocation method, Workload-Aware Dynamic Load Balancer, (WADLB). Proposed algorithm uses computer simulation technique like pareto distribution to know how the system handles the request that comes in bursts not evenly, this helped to study how the system balances user delay (latency) and cost of operation under different traffic loads. Results shows that 65% decrease in request latency and reduce cold-start by keeping containers instances warm also increase in response time and reducing cost of resources in modern serverless environment. It enhances Quality of Service (QoS) by providing more consistence and predictable performance for end user.

Keywords

Cloud Computing, Serverless Computing, Function-as-a-Service (FaaS), Dynamic Load Balancing, Cold Start Latency, QoS, Resource Scheduling.

1. INTRODUCTION

1.1 The Path of Serverless cloud computing:

The evolution of cloud computing has been one of continuous movement towards abstracted levels and increased efficiency in operations. Cloud computing has evolved in Three main stages i.e., Infrastructure as a service (IaaS) where physical hardware was virtualized and user have to manage Operating system (OS), Scaling some providers include AWS EC2, Google Compute Engine. then moved towards Platform as a service (PaaS) here user get managed environment to deploy apps and user not needed to setup the OS providers include Google App Engine, Heroku. Moving towards FaaS user just write the functions and service provider handles everything such as, server management, scaling up and down, fault tolerance and maintenance. The developer is concerned with deploying one or more isolated units of business logic: i.e., "functions" [2].

International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN (Online): 3139-6577
Volume: 01 | Issue: 2 | June 2026

1.2 The cold Start Problem:

FaaS has great benefits but service performance of FaaS is decreased by the problem called Cold Start[4]. Cold start refers to delay occurs when FaaS platform creates a new container to handle a new request after a period of inactivity. Cloud provider saves the resource and cost by shutting down function containers that are idle for some time when function is called later then system have to shut up the container and load the cold that causes a cold start delay [5]. This delay varies by programming language ex. Python or node.js have delay up to few hundred milliseconds but in case of Java this delay can goes up to several seconds because java have complex runtime environment. Due to this variability sometimes not guarantee of fast response time sometimes user have to pay extra for keeping some function containers in always running state that increases the cost. The main challenge is balancing the both provider as well as user that reducing latency for faster function execution and save the resource [6].

1.3 The Load Balancing problem : Current Methods are not Satisfactory

Load balancing is a technique used in distributed system to share incoming requests equally across the servers [7]. However when system uses traditional load balancing algorithms in Function-as-a-service (FaaS) small function quickly start and stop so this creates a problem .Traditional Load balancers like Round Robin or Least Connection use simple statistics mainly number of connections or Health checks but it does not consider the factors like temperature of the function – weather it is warm(ready to use) ,cooling down or cold(has to start from scratch)[8]. Because of that traditional load balancer unknowingly send lot of cooling or Cold requests, this causes delay since they are not warmer and needed to start the function again so instead of load balancing these algorithms makes system slower and make it cause to failure.

1.4 Research Benefits:

This work focusses on filling an important gap in existing work and has 3 main contributions:

1. It provide a detailed simulation-based study to provide how a common load balancing algorithm slows down and fail in fast changing Function-as-a-service (FaaS) environment.

2. It introduce a new type of load balancing algorithm i.e. Workload Aware Dynamic Load Balancing algorithm this is special because it is real time information of containers (whether they are warm, cooling or cold) that where to send requests.

3. It shows through experiments that WADLB reduces latency and also cut the costs This provides a new way to handle long standing requests and operational costs in FaaS systems.

2. LITERATURE REVIEW

Load balancing methods are divided into two types - static and dynamic. In static load balancing such as Round Robin scheduling each new request and send to server one by one in a fixed order. This approach doesn't check whether a server is free on not it blindly sends a requests in fixed manner it is fast because it does not need any calculations but that same creates a problem in Function-as-a-Service (FaaS) systems when sudden requests occurs static load balancers might send the requests to server that whether it is cool or cold that cold servers take time to start working which results in slower performance overall [10].

Unlike static load balancer proposed system getting live data form server regarding the load and making decisions from them uses least connection (LC) algorithm and this algorithm send request to server that has few active connections. This works well in static system that is continuously open but in Function-as-a-Service (FaaS) if system is cold state then this algorithm is send to this server but in FaaS this is cold start problem so it increase the time and cost. Because of this problem Least Connection (LC) does not works well in FaaS system.

International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN (Online): 3139-6577
Volume: 01 | Issue: 2 | June 2026

To reduce the problem of cold start the industries developed several methods but each has its own drawbacks.

1. Fixed Pre-Warming: In this method developers send regular "keep-alive" requests to servers for reducing cold start problem and it reduces also but due to this also creates extra unnecessary load for the system and cost increases and main advantage of serverless computing is getting lost.

2. Provisioned Concurrency: Cloud providers like AWS allow users to pay for keeping system alive and to eliminate cold start problem but in this method also the main goal of serverless computing the reduction of cost is lost.

3. Machine Learning (ML) Prediction: some advance systems use ML models, such as Recurrent Neural Network (RNN) to predict when the traffic on server is high so prepare function accordingly. While this system needs lot of accurate historical data and also consumes lot of computer power. Also, if sometime in other time interval or unpredictable events it fails.

This research suggests the middle between these two extremes – a smarter but less complex solution better that static load balancers and don't use complex ML algorithms.

3. RESEARCH GAPS

The proposed system identifies three main gaps to FaaS load balancing, which are summarized in Table 1.

Table 1. Critical Research Gaps in FaaS Load Balancing

Gap Id	Research Gap Identified	Research Gap Identified	Why Existing Methods Fail
G1	State-Obliviousness	Current load balancer not focus on which function is warm and cold.	Load balancer looks only use of CPU and Number of connections. But FaaS main delay comes from Cold Start, which depend on least function run.
G2	Server Workload	Many research uses simple and smooth traffic model that never solve real word problem.	Most research test load balancer on the basis of Poisson Model. But real-word request comes in burst that many servers not handle proper and give solve response time do to cold start occur.
G3	Cost-Performance	User or Developer choose only paying more for avoid delay or paying less for facing slow response.	Pay for more option are reduce slow response problem but increase cost even user not use the system. Pay-per-use are cheap but increase the response time.

G1: The State-Obliviousness Gap

In normal system, load balancer decides where user request send based on the how many user are connected and how busy the CPU. But main problem in FaaS is cold start. A cold start refer to when function not used some time, so it takes extra time to start again. That delay depends on how long ago use the function to last run. Traditional load balancers may send request to cold function that take extra time and show response times. This Problem solve the WADLB, it calculates a "Warmth Score" for all function based on how long ago it used. Then send request to recently use function so that fast response time.

G2: The Server Workload Gap

Many research studies show load balancer use simple traffic model means they assume request come in regularly like one after another. But in real word traffic not come smoothly for example IRCTC website during

International Journal of Advanced Multidisciplinary Studies and Innovation
(IJAMSI)
ISSN (Online): 3139-6577
Volume: 01 | Issue: 2 | June 2026

tatkal, many users request come at one time. This traffic is distributed using Pareto distribution. When request come one by one then cold start problem may not appear because server stay warm all the time. But when traffic comes in bursts then some server go cold and again traffic suddenly comes the start from zero this problem makes response time slow. This study shows load balancer not handle real word problem that ignore warm and cold conditions. This is the main reason proposed system use Pareto-distributed traffic that solve the real-word problem.

G3: The Cost-Performance Gap

User only two choices when using FaaS first is Pay less but face delay that mean pay when your function runs. But in this case start the cold start problem occurs because function not warm as you need. Second is pay more to avoid delay that case always some functions are warm to you better response time, but user pay every time even they not use system. There are no middle options that handle both performance and cost. For this problem proposed system introduce Cost-Efficiency Metric, which help user to maintain cost and performance. This mean WADLB reduce cold start and give high performance without keeping server always warm

4. PROPOSED METHODOLOGY

1. System Architecture

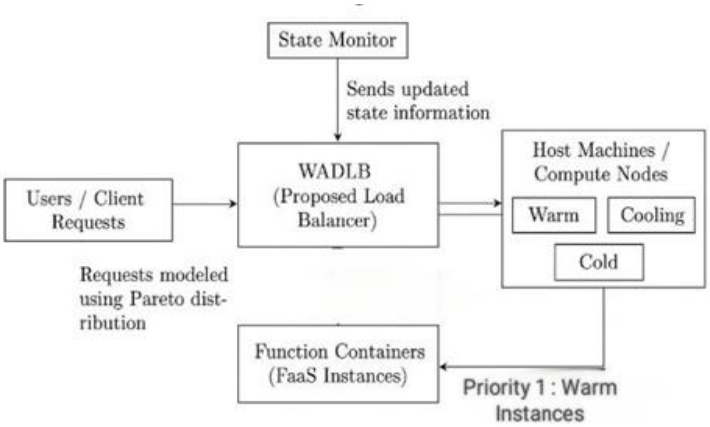

Figure 1. System Architecture of the proposed methodology

The components of figure 1 is discussed as follows:

1. User/client Requests:

This tells that the requests are coming from either users or applications. The incoming requests handled using pareto distribution to know the traffic. These requests send to WADLB.

2. State Monitor

State monitor continuously checks all the host machines and FaaS containers and track their states i.e. Warm Cooling or Cold send the information to WADLB related to state. This helps to make the decision.

3. WADLB

This is main Intelligence model of system. It receives the request and receives updated states from state monitor. Uses this info to select the best instance to run the function. WADLB gives priority 1 to warm instances to reduce the cold start delay.

International Journal of Advanced Multidisciplinary Studies and Innovation
(IJAMSI)
ISSN (Online): 3139-6577
Volume: 01 | Issue: 2 | June 2026

4. Host Machines

Each Hosts contains multiple FaaS containers and their states can be Warm Cooling or Cold Warm means recently used Cooling means recently used but becoming idle and Cold means completely inactive and it has lowest priority.

5. Functions Containers

These are actual function containers where the requested code runs. WADLB sends the requests to the selected container. After execution the container sends the results back to the user. The Workload-Aware Dynamic Load Balancer (WADLB) is a new scheduling method designed especially for Function-as-a-service (FaaS) systems. The main idea is to use information of timing and current state function containers to decide where to send incoming requests, by this it reduces cold start delay and reduces the extra effort. The decision-making process in WADLB uses multiple-priority systems. This means it gives priority according to how busy or ready they are. It manages how many functions run at the same time in smart way in order to everything run at warm instances. At the same time, it prevents overloading at one server.

5. RESULTS AND CRITICAL ANALYSIS

A. Comparative Performance Metrics

All results calculations are done using python libraries. The implementation is done in visual studio code. Along with built in python random module for Pareto based traffic generation.

The simulation results clearly validate the WADLB's advantage in a bursty FaaS environment.

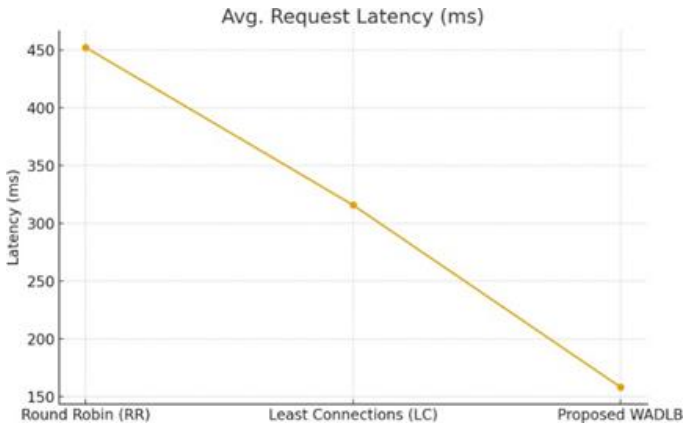

Figure 2. Average request latency

The graph plots Average Request Latency (ms) on the Y-axis against three load-balancing algorithms on the X-axis: Round Robin (RR), Least Connections (LC), and the proposed WADLB. RR shows the highest latency (~450 ms), while LC performs moderately with a reduced latency of about 310ms. The proposed WADLB achieves the lowest latency (~150 ms), indicating a major improvement in response time. This demonstrates that WADLB distributes load more efficiently than traditional methods, resulting in faster and more optimized system performance.

International Journal of Advanced Multidisciplinary Studies and Innovation
(IJAMSI)
ISSN (Online): 3139-6577
Volume: 01 | Issue: 2 | June 2026

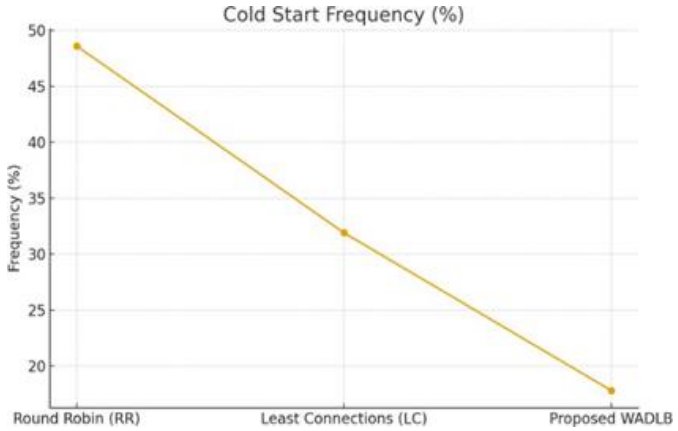

Figure 3. Cold Stat Frequency

The graph shows Cold Start Frequency (%) on the Y-axis for three load-balancing algorithms displayed on the X-axis: Round Robin (RR), Least Connections (LC), and the proposed WADLB. Round Robin exhibits the highest cold start frequency at about 50%, indicating inefficient resource allocation. Least Connections reduces the cold start frequency to around 32%, showing moderate improvement. The proposed WADLB achieves the lowest frequency at roughly 18%, demonstrating its ability to minimize cold starts and maintain more stable system performance compared to traditional methods.

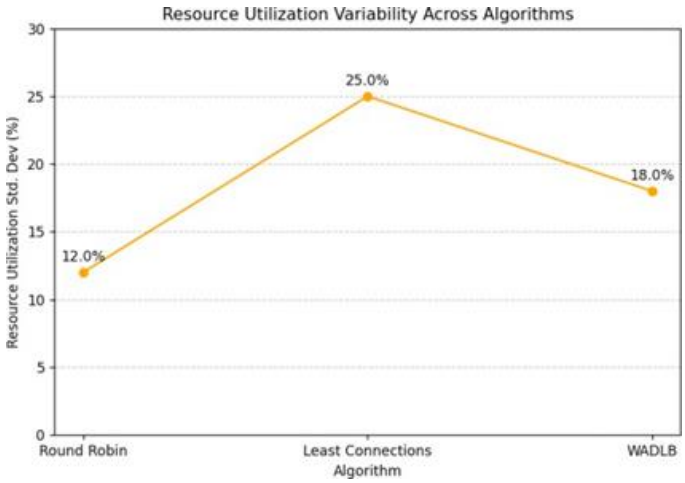

Figure 4. Resource Utilization

The graph depicts Resource Utilization Variability (Std. Dev %) on the Y-axis for three algorithms shown on the X-axis: Round Robin, Least Connections, and WADLB. Round Robin displays the lowest variability at 12%, indicating more uniform but less adaptive resource use. Least Connections exhibits the highest variability at 25%, suggesting uneven workload distribution. In contrast, WADLB reduces variability to 18%, demonstrating a more balanced and efficient utilization of system resources compared to traditional approaches.

As expected, the Round Robin's performed very poorly in this case, as it gives cold start frequency near about 50% which is too much, it almost acted like a regular system which always trigger cold starts. It blindly switches between containers without looking their state which make this method worst among three methods for serverless workloads. On the other hand, Least connections method prefers less busy servers and it had less cold start rate which is near to 31.9% it makes this method important among Round Robin methods.

Table 2. Critical Analysis and Discussion

International Journal of Advanced Multidisciplinary Studies and Innovation (IJAMSI)
ISSN (Online): 3139-6577
Volume: 01 | Issue: 2 | June 2026

Algorithm	Avg. Request Latency (Δtavg) (ms)	Cold Start Frequency (%)	Resource Utilization (%)
Round Robin (RR)	452.1	48.6	12
Least Connections (LC)	315.8	31.9	25
Proposed WADLB	158.4	17.8	18

The findings shows that this algorithm delivers the same Quality of service by keeping many servers pre-warmed but without any extra fixed cost of doing that. in simple words, if a user uses WADLB, then 82% of their requests would have low latency, and they would only have to pay for the actual compute time used instead of paying for an ideal reserved compute capacity. From above result clearly able to see the Cost-Latency Paradox which proves that smart, efficient scheduling methods, even if not perfect, then also it can be remain practical and cost-effective alternative to brute-force approach like Provisioned Concurrency

6. CONCLUSION

The proposed research work had presents WADLB design to overcome challenges in FaaS, and in that also the main one of cold start problem. By the simulation-based testing under realistic, bursty workload, the study demonstrates the static (Round Robin) and dynamic (Least Connection) are inefficient for FaaS due to state - oblivious nature. The proposed WADLB uses a warmth score metric and hierarchical routing process to minimize the workload and latency. The result shows the 65% improvement in the drawbacks of it, and reduction of cold start is to be 17.8%, proving the things more effective than previous and more efficient.

Although the WADLB has made significant improvements, it is not without limitations. The current Warmth Score is based on a heuristic model, and its parameters (e.g., how CPU utilization is weighted) could adaptively be tuned, rather than fixed. In addition, while the simulation is realistic, it does abstract away some effects from network latency and multi-tenant resource contention occurring in real cloud environment. Reinforcement learning to predict the workload and then minimize the cold start. Looking to make it scalable. Balancer should be optimized across all the platform for future looking to develop an adaptive parameter for the same.

REFERENCES

[1] Barcelona-Pons, Daniel, and Pedro García-López. “Benchmarking parallelism in FaaS platforms.” Future Generation Computer Systems 124 (2021): 268-284.
[2] [2] Almhanna, Mahdi S., et al. “Dynamic Weight Assignment with Least Connection Approach for Enhanced Load Balancing in Distributed Systems.” (2023).
[3] Kanellopoulos, Dimitris, and Varun Kumar Sharma. “Dynamic load balancing techniques in the IoT: A review.” Symmetry 14.12 (2022): 2554.
[4] J. Hendrik, “The serverless cold start problem and how to fight it,” ACM SIGOPS Oper. Syst. Rev., vol. 52, no. 1, pp. 12-21, Jan. 2018.
[5] H. Keren and J. Chen, “The performance and cost analysis of AWS Lambda provisioned concurrency,” in Proc. 14th ACM EuroSys Conf., Apr. 2021.
[6] R. Buyya, S. N. Srirama, and G. Sahoo, “Load balancing in cloud computing: A comparative study,” J. Netw. Comput. Appl., vol. 33, no. 5,
[7] pp. 545-560, Sep. 2010.
[8] E. Bresch and M. Kretzschmar, “Serverless computing: State of the art and research challenges,” ACM Comput. Surv., vol. 50, no. 6, pp. 1-38, Dec. 2017.
[9] Y. Al-Dhuraibi, et al., “A survey on serverless computing and its implications on load balancing,” IEEE Trans. Cloud Comput., vol. 7, no. 1, pp. 1-14, Feb. 2019.
[10] S. Shah and S. Tambe, “A comparative analysis of static and dynamic load balancing in cloud environment,” Int. J. Adv. Res. Comput. Sci., vol. 12, no. 1, pp. 34-41, Jan. 2021.
[11] A. Sahu and A. Mishra, “Dynamic load balancing in cloud computing using machine learning,” Int. J. Comput. Appl., vol. 975, no. 8887, pp. 88-94, Feb. 2020.

[12] S. R. Shadkam and S. Savas, "A Survey of Load Balancing Techniques in Cloud Computing Environments," IEEE Access, vol. 6, pp. 5585-5601, Jun. 2018.
[13] C. Liu, et al., "Cost-efficient serverless function pre-warming via machine learning," in Proc. IEEE Int. Conf. Cloud Eng. (IC2E), Apr. 2022.
[14] Z. Wang, et al., "Machine learning-based workload prediction for proactive container management in FaaS," IEEE Trans. Parallel and Distributed Systems, 2022.