EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

Dynamic Load Balancing in Cloud Computing: A Review and a Novel Approach

Jasobanta Laha[1*], Sabyasachi Pattnaik[2] and Kumar Surjeet Chaudhury[3]

[1,2]PG Department of Computer Sc., Fakir Mohan University, Balasore, Odisha, India
[3]School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, Odisha, India

Abstract

In cloud computing, load balancing is essential since it guarantees effective resource utilisation and reduces response times. It effectively distributes incoming workload across available servers. Load balancing involves dividing tasks among multiple systems or resources over the internet. By distributing traffic and workloads, load balancing ensures that no server or computer is overloaded, underutilized, or idle in a cloud computing environment. To enhance overall performance in the cloud, it optimises a number of variables, including execution time, response time, and system stability. To increase the effectiveness and reliability of cloud services, load balancing evenly distributes traffic, workloads, and computing resources across the environment. The proposed method, Enhanced Dynamic Load Balancing for Cloud Computing, adjusts load distribution and maintains a balanced system by considering variables like server capacity, workload distribution, and system load. By incorporating these factors and employing adaptive threshold adjustment, this approach optimizes resource allocation and enhances system performance. Experimental research shows that the proposed new novel approach is more effective and efficient than current load balancing techniques. In this context of cloud computing, this ground-breaking method offers a workable substitute for dynamic load balancing.

Keywords: Cloud computing, load balancing, response time, execution time, and system stability

Received on 17 December 2023, accepted on 07 March 2024, published on 12 March 2024

doi: 10.4108/eetiot.5387

*Corresponding author. Email: yeshmcp@gmail.com

1. Introduction

The management and delivery of services by organisations over the internet has been completely transformed by cloud computing. Effective resource management and workload distribution are now essential for maintaining scalability, high availability, and top performance due to the growing rise and acceptance of cloud services. In order to accomplish these goals, dynamic load balancing strategically distributes incoming workload among the available servers in the cloud infrastructure [1].

Machine learning and neural networks, two types of artificial intelligence (AI), were mostly used in the past by load balancing algorithms to dynamically distribute workload among servers. AI-based technologies, which usually require huge computational resources, could make the system more complicated and costly.

In this paper, we present a non-AI solution for dynamic load balancing. To efficiently balance incoming workload among the available servers, our Enhanced Dynamic Load Balancing Algorithm takes into account critical factors such server capacity, workload distribution, and system load. This is a new strategy that offers an alternative solution which is efficient and resource-friendly. Our algorithm dynamically adjusts the load distribution to maintain a balanced system while ensuring optimal utilization of resources by considering these factors and employing adaptive threshold modification.

We present a novel, AI-free solution for dynamic load balancing in this research. Without depending on intricate AI models, the proposed Enhanced Dynamic Load Balancing

Algorithm for Cloud Computing seeks to accomplish effective work distribution and resource utilisation[2]. Our method continuously adjusts the load distribution to maintain a balanced system, considering important factors such as server capacity, workload distribution, and system load.

2. The term Load Balancing in Cloud Computing

In a cloud computing environment, load balancing refers to optimizing the distribution of virtual machine (VM) resources. To ensure equitable and dynamic task allocation while effectively utilizing resources, it is an essential technique used in the cloud environment [3]. Its main objective is to effectively manage the workload among different cloud nodes, thereby preventing any node from being overloaded or underutilized. Enhanced resource allocation and improved user satisfaction are the advantages of achieving a more efficient distribution of workloads [4]. In order to improve the performance of cloud-based applications and find solutions to load balancing problems, it is important. The load balancing in cloud computing is shown in Fig. 1.

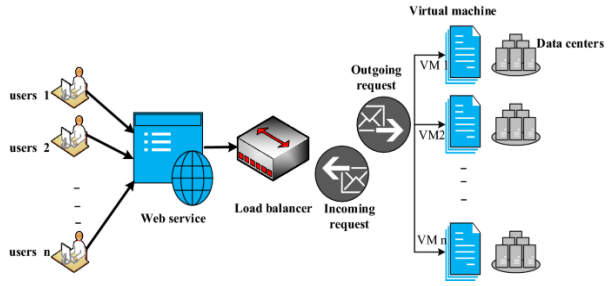

Figure 1. Working of Load Balancing in cloud servers

3. Various Parameters are employed for dynamic load balancing

In order to improve performance and resource utilization, dynamic load balancing in cloud computing effectively distributes workload among numerous servers or resources. A variety of factors are considered by dynamic load balancing algorithms to achieve successful load balancing. The list of frequently used parameters is as follows [5]:

1. CPU utilization: Monitoring the CPU usage of servers is a crucial parameter. Load balancers consider the current CPU utilization of each server to distribute the workload evenly. Servers with lower CPU utilization are preferred to handle additional requests.
2. Memory utilization: Memory (RAM) usage is another critical parameter. Load balancers monitor the memory consumption of servers and factor it into the load balancing decision. Servers with available memory can handle additional requests more efficiently.
3. Network utilization: The network bandwidth and congestion are considered parameters in load balancing. Servers with higher network capacity and lower traffic congestion are preferred for handling incoming requests.
4. Response time: The response time of servers to incoming requests is an important metric. Load balancers collect response time information from servers and direct new requests to servers with lower response times, ensuring faster service delivery.
5. Request queue length: The length of the request queue on each server helps determine the load on that server. Load balancers monitor the queue length and direct new requests to servers with shorter queues, reducing the waiting time for request processing.
6. Server health: Load balancers continuously monitor the health of servers to ensure that only healthy servers receive incoming requests. Parameters like server uptime, error rates, and resource availability are considered in assessing server health.
7. Geographical location: In some cases, load balancers consider the geographical location of servers and clients. By directing requests to servers closer to clients, latency and network congestion can be minimized, resulting in better performance.
8. Server capacity: Load balancers take into account the capacity or capability of each server. Factors like CPU power, memory capacity, and available resources are considered to distribute the workload optimally.
9. Dynamic thresholds: Load balancing algorithms often utilize dynamically adjustable thresholds to trigger load balancing decisions. These thresholds can be based on parameters like CPU utilization, network traffic, or response time to adapt to changing workload conditions.

It's important to note that different load balancing algorithms and techniques may use a combination of these parameters or additional ones based on specific requirements and constraints [6].

4. Algorithm for Dynamic Load Balancing

The current state of the system serves as the basis for all load balancing computations in this type of algorithm. The workload distribution will be based on the system's state at the moment. The primary advantage of this strategy, facilitated by dynamic load migration, is that load balancing decisions are made considering the system's current state, leading to improved overall performance [7].

5. Algorithm for Load Balancing Currently Used

The various load balancing strategies used in the cloud computing environment are thoroughly assessed and compared based on a predetermined set of factors. Throughput, response time, overhead, performance, fault tolerance, migration time, resource utilization, and scalability are some of these factors. Some commonly used load balancing algorithms are as follows [8]:

5.1 The Round Robin Load Balancing Algorithm

Round robin is the most widely used and straightforward scheduling algorithm. The algorithm's primary focus is time sharing. This algorithm executes each step in a cycle. A time quantum, or precise time slice, is defined by the system. In this load balancing, the datacenter controller rotates the VMs that are getting requests. The procedure will continue until the conclusion of the prior stage. The biggest problem is that if the old process took a long time and was very large, the new one will take just as long to complete [9]. The Round Robin technique does not take into consideration resource capabilities, priority, or assignment time. Slower response times are caused by longer jobs and higher priority.

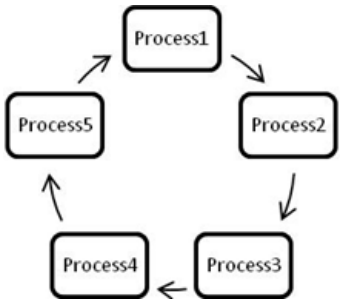

Figure 2. Round Robin Load Balancer [15]

5.2 The Equally Spread Current Execution (ESCE) algorithm

Process priority is taken into account by the dynamic load balancing method known as Equally Spread Current Execution (ESCE) [10]. By moving the burden to lightly populated virtual machines (VMs), it seeks to fairly distribute the workload by determining the priority based on the size of the process. By taking into account process size and choosing a VM with a lower load, this technique randomly distributes the load [11].

ESCE focuses on spreading the load across different nodes, earning it the name of a spread spectrum technique. By distributing the load evenly, the algorithm ensures that no specific VM becomes overloaded while others remain underutilized. This approach aims to optimize resource utilization and improve overall system performance.

Unlike Round Robin, which allocates resources in a cyclical manner without considering process size or priority, ESCE takes these factors into account. By prioritizing processes based on their size, the algorithm can make informed decisions about load distribution, ensuring that larger processes receive appropriate resources while preventing delays or bottlenecks.

ESCE offers an effective method for streamlining resource allocation in cloud computing environments through dynamic load balancing. ESCE contributes to increased system performance, shortened reaction times, and greater overall resource utilisation by allocating the workload across available VMs in a way that takes process size and current load into account [12].

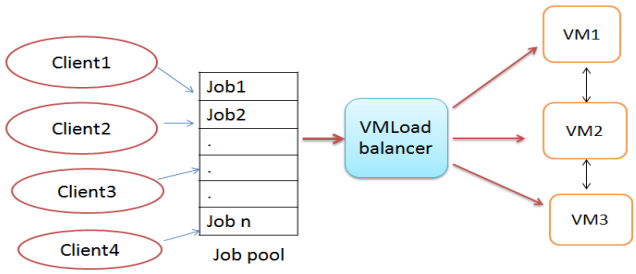

Figure 3. ESCE Load Balancer

5.3 The Throttled Load Balancer (TLB) algorithm

The Throttled load balancing method checks the status of virtual machines (VMs) to see if they are ready to handle incoming requests. Each VM's ID and state (either "Available" or "Busy") are stored in an index database that

the load balancer maintains. Every VM is created with the "Available" status [14] by default.

In order for the data centre to locate the appropriate VM, the load balancer receives client requests and forwards them there. Starting at the top of the index table, the load balancer searches for a VM that can handle the request. The data centre controller is contacted with the ID of any free VMs found so that it can be given requests.

The load balancer adjusts its index table as necessary after receiving notification that the data centre successfully assigned the requested VM ID to the request. The index table is not updated if the data centre runs into problems during the allocation process, but it does provide a negative feedback signal within a certain time frame.

The data centre controller receives a -1 notification from the load balancer when all VMs are in the "Busy" status. In these situations, the data centre begins internally queuing user requests until a VM becomes available [15].

A VM notifies the data centre controller when a task assigned to it is complete, and the controller then notifies the load balancer. The load balancer makes the necessary adjustments to the index table.

Three phases can be used to estimate how long the Throttled load balancing algorithm will take to complete. In the initial stage, virtual machines are constructed and left idle while they wait for the scheduler to distribute work from the queue. The second stage entails the actual job processing by the cloud-based VMs. The VMs are then either cleaned up or destroyed in the third phase.

It's important to keep in mind that this computing model's throughput is determined by the total number of tasks finished in a given period of time, disregarding the time required for VM construction and destruction.

In conclusion, the Throttled load balancing algorithm makes use of the VMs' status to ensure effective request distribution in cloud computing settings. The algorithm allows for efficient job assignment and boosts system performance by keeping an index table and taking VM availability into account.

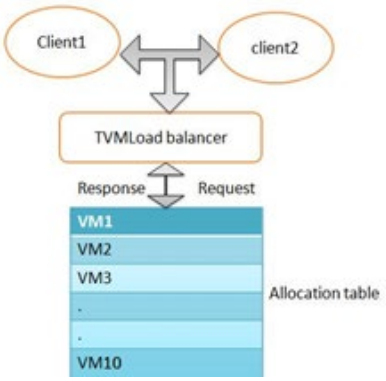

Figure 4. Throttled Load Balancer

5.4 The Active Monitoring Load Balancer (AMLB) algorithm

Active Monitoring Load Balancer (AMLB) is a load balancing technique utilized in cloud computing environments, specifically for dynamic load balancing. This strategy entails compiling data on each virtual machine (VM), including the quantity of requests that are currently being allotted to them [16]. Upon receiving a new request, the Data Centre Controller (DCC) searches for the least loaded or idle VM in the VM index table. This approach uses the first-come, first-served principle to distribute the load to the VM with the lowest index number when there have multiple servers [17].

After verifying that the requested VM is the one indicated by the VM ID returned from the AMLB method, the DCC transmits the request to that specific VM. The AMLB is notified of the new allocation by the DCC, which also delivers the cloudlet [18].

Upon completion of the task and lowering of the VM index table, the data is given to the DCC. The load balancer iterates the database once more before allocating a process to a new request. Figure 5 depicts an illustration of AMLB.

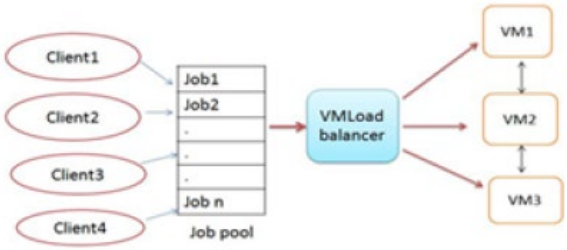

Figure 5. Active Monitoring Load Balancer

6. Problem Statement

The goal is to create a dynamic load balancing algorithm that evenly distributes the incoming workload among the available servers while taking into account server capacity, workload distribution, and system load. The method optimises resource utilisation and improves response and processing times by dynamically adjusting the workload distribution based on the availability and priority of jobs.

6.1 Proposed EMT-LB Algorithm: A Novel Approach

Based on the **Throttled Load Balancer (TLB)** algorithm, we will propose a modified version called ***Enhanced Model Priority Based Throttled Load Balancing (EMPBT-LB)*** algorithm. This updated algorithm incorporates task prioritization to further improve load

distribution and responsiveness. Here's the detail description of the modified algorithm:

1. Two lists should be initialised with the available and busy VMs, respectively: the "Available VMs list" and "Busy VMs list."
2. When a task arrives, the EMPBT-LoadBalancer receives the task and its priority from the DCCP.
3. EMPBT-LoadBalancer checks the priority of the task and selects the appropriate VM for execution based on the following steps:

a. If the "Available VMs list" is not empty:

- If there is a VM with a matching priority, select the first available VM with the highest matching priority.
- If there are multiple VMs with the same highest matching priority, select the VM with the lowest current workload.
- If no matching priority is found, select the first available VM with the lowest current workload.

b. If the "Available VMs list" is empty:

- If there is a VM with a matching priority in the "Busy VMs list", queue the task and wait for the first available VM with a matching priority.
- If no matching priority is found in the "Busy VMs list", queue the task and wait for the first available VM.

4. Once a VM is selected, mark it as busy, move it to the end of the "Busy VMs list," and execute the task on the VM.
5. After task execution, mark the VM as available, move it to the end of the "Available VMs list," and update its current workload.
6. If there are any queued tasks:
   - Check if there is an available VM with a matching priority in the "Available VMs list."
   - If found, assign the task to the VM, mark it as busy, and move it to the end of the "Busy VMs list."
   - If no matching priority is found, assign the task to the first available VM in the "Available VMs list," mark it as busy, and move it to the end of the "Busy VMs list."
7. Repeat steps 2-6 for new incoming tasks.

By introducing task prioritization, the EMPBT-LB algorithm aims to allocate tasks more effectively, ensuring that higher-priority tasks are executed promptly while maintaining load distribution among the available VMs. The algorithm dynamically adjusts the workload distribution based on the availability and priority of tasks, leading to improved response time, processing time and maximizing resource utilization compared to the original TLB algorithm and other load balancing algorithms like RR and AMLB.

7. Simulation Specifiers

- 30 users are in the user base.
- 4 data centres total.
- Simulation: It lasts for 60 minutes.
- Count of regions visited: separated into six regions globally
- OS: LINUX
- Virtual Machine Manager: Xen (Paravirtualization)

8. Comparison, analysis and overall response time of cloudlet

Calculating the minimum, maximum, and average times for both scenarios allowed for a comparison. The throttled algorithm is the most effective among the three [19]. As a result, the proposed technique is evaluated using the same parameter that was previously used. Here is further information on the suggested algorithm's findings and how they compare to the throttled method.

When comparing the proposed algorithm to the throttled load balancing algorithm used in cloud computing, it can be shown from the findings in table 1 that the suggested technique takes much less time (minimum, maximum, and average). The average, minimum, and maximum values of the two algorithms' overall response times (ms) are illustrated in Table 1. Using the information from table 1 as a starting point, separate graphs for the average and minimum overall response times are shown in figures 6 and 7, respectively.

Table 1.

ALGORITHM↓	OVERALL RESPONSE TIME		
	AVERAGE (ms)	MINIMUM (ms)	MAXIMUM (ms)
Throttled	220.28	34.36	16285.43
EMPBT-LB*	**218.86**	**33.12**	**16285.03**

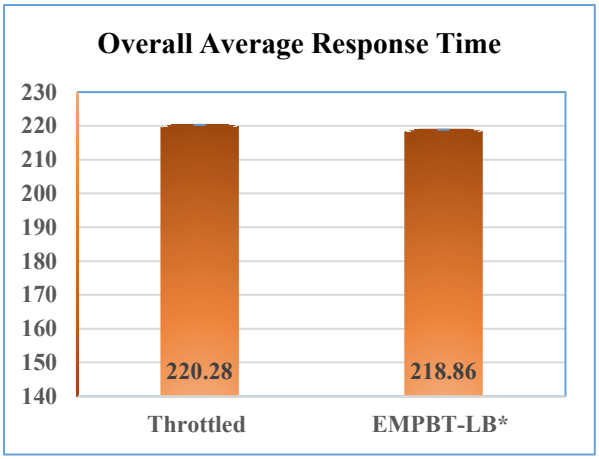

Figure 6. Illustrates an overall response time comparison for an average situation

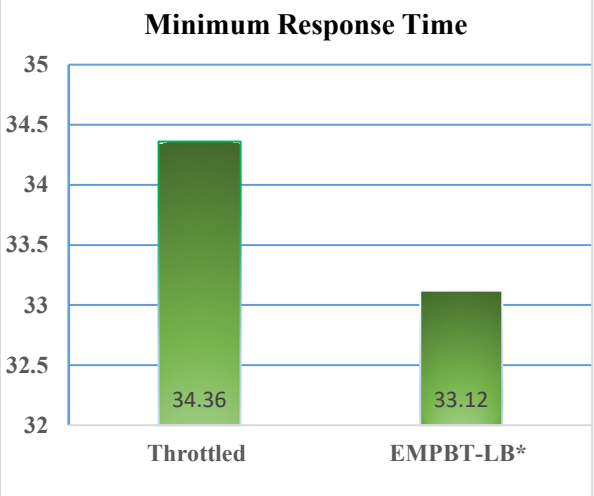

Figure 7. Illustrates an overall response time comparison for an average situation

9. Conclusion

The proposed Enhanced Model Priority Based Throttled Load Balancing (EMPBT-LB) algorithm, which is the best of the existing dynamic load balancing algorithms mentioned in the study, performs better than the Throttled load balancing algorithm, as can be seen from the paper. Regarding overall response time and datacenter processing time (for all circumstances, i.e. average, lowest, and highest), the suggested method obviously excels over the throttled load balancing algorithm.

We could propose that the suggested EMPBT-LB dynamic Load Balancing Algorithm is the best algorithm among those stated in this research because it performs better in both the scenarios of total response time.

References

[1] Abhay Kumar Agarwal, Atul Raj, A New Static Load Balancing Algorithm in Cloud Computing. International Journal of Computer Applications (0975 – 8887), Volume 132 – No.2, December2022.

[2] Sangeeta, Suman, Load Balancing in Cloud Computing: A Review, International Journal of Advanced Research in Computer Science, Volume 9, No. 2, March – April 2018.

[3] Ahmad AA Alkhatiba), Abeer Alsabbagh, Randa Maraqa, Shadi Alzubi, Load Balancing Techniques in Cloud Computing: Extensive Review, Advances in Science, Technology and Engineering Systems Journal (ASTES Journal), Vol. 6, No. 2, 860 – 870 (2021).

[4] S. Suguna and R. Barani, Simulation of Dynamic Load Balancing Algorithms, Bonfring International Journal of Software Engineering and Soft Computing, Vol. 5, No.1, July 2015.

[5] Soumya Ray and Ajanta De Sarkar, "Execution analysis of load balancing algorithms in cloud computing environment," International Journal on Cloud Computing: Services and Architecture (IJCCSA),Vol.2, No.5, October 2021.

[6] N. R. Tadapaneni, "A Survey Of Various Load Balancing Algorithms In Cloud Computing," 2020.

[7] Manjula K., S. Meenakshi Sundaram, Improved and Efficient Dynamic Load Balancing Algorithm in Cloud Based Distributed System, International Journal of Recent

Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-5, January 2020

[8] Amrutanshu Panigrahi, Bibhuprasad Sahu, Saroj Kumar Rout, and Amiya Kumar Rath, M-Throttled: Dynamic Load Balancing Algorithm for Cloud Computing, © Springer Nature Singapore Pte Ltd. 2021 D. Mishra et al. (eds.), Intelligent and Cloud Computing, Smart Innovation, Systems and Technologies 194,

[9] Soumen Swarnakar, Ritik Kumar, Saurabh Krishn, Improved Dynamic Load Balancing Approach in Cloud Computing. 2020 IEEE International Conference for Convergence in Engineering.

[10] R. Sajjan, B. R. Yashwantrao, "Load balancing and its algorithms in cloud computing: a survey," International Journal of Computer Sciences and Engineering, 5(1), 95–100, 2017.

[11] Ren et al., "The load balancing algorithm in cloud computing environment," in International Conference on Computer Science and Network Technology, Changchun, China, 2012.

[12] J. Bhatia et al., "HTV Dynamic Load Balancing Algorithm for Virtual Machine Instances in Cloud," in International Symposium on Cloud and Services Computing, Mangalore, KA, 2012.

[13] Dharmesh Kashyap, Jaydeep Viradiya, "A Survey of Various Load Balancing Algorithms in Cloud Computing", International Journal of Scientific & Technology Research, Vol. 3, Issue 11, November 2014.

[14] D. C. Devi and V. R. Uthariaraj, "Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Nonpreemptive Dependent Tasks," The Scientific World Journal, vol. 2016, pp. 1–14, Feb. 2016

[15] A. Khiyati, M. Zbakh, H. El Bakkali, D. El Kettani "Load Balancing Cloud Computing: State Of Art"IEEE, 2012.

[16] Martin Randles, David Lamb, A. Taleb-Bendiab, A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing, 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops

[17] A. Singh, P. Goyal, S. Batra: Anoptimized round robin scheduling Algorithm for CPU scheduling, International journal of computer and Electrical engineering (IJCEE), vol. 2, No. 7, Pp 2383- 2385, December, 2010.

[18] S. K. Upadhyay, A. Bhattacharya, S. Arya, T. Singh, "Load optimization in cloud computing using clustering: a survey," Int. Res. J. Eng. Technol, 5(4), 2455–2459, 2018.

[19] N. R. Tadapaneni, "A Survey Of Various Load Balancing Algorithms In Cloud Computing," 2020.