International Journal of Basic and Applied Sciences, 14 (4) (2025) 144-148

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/kaejtm69
Research paper

Load Balancing in Distributed System

Nahla Flayyih Hasani *

University of Sumer, Rifai/Thi-Qar, Iraq
**Corresponding author E-mail: nahla.flayyih@gmail.com*

Received: June 12, 2025, Accepted: July 20, 2025, Published: August 7, 2025

Abstract

Scalability is a critical requirement for distributed systems, especially as the number of users and their demands grow. This paper reviews and compares various load balancing techniques, which are essential for ensuring efficient server resource utilization. We examine both static and dynamic load balancing methods, highlighting their properties, operational procedures, advantages, and limitations. A comparative analysis is provided to assist in selecting the appropriate technique for different application scenarios.

Keywords: *Load Balancing; Distributed System; Load Balancer; Static; Dynamic*

1. Introduction

Load balancing is the operation for allocating network signals across many servers. Load balancing develops the response of the application. It raises applications availability and websites to users. Recent applications are running with load balancers. Software load balancers of software are being added with more capabilities for the security of applications [1]. Load balancing is introduced via some systems, like Microsoft's Network Load Balancing [2]

2. Load balancing concept

Load balancing is the total load allocation between serving destinations [3]. In distributed systems, Load balancing is important for get the goodness of service by controlling client loads, which are changeable over time. Arriving requests are divided between resources of system, which are ready to evade suffocation of resources as well as to take full advantage of ready resources [4]. Load balancing produces additional resources in preparation for handling increased loads [5]. Anthropogenic alterations to the Earth established new habitats for mosquitoes and increased contact between humans and mosquitoes. Urban development, agricultural expansion, and deforestation are some of the significant contributors to emerging and reemerging zoonoses [24]. The principal reason for the establishment of new habitats is the alterations to the natural space previously occupied by the mosquitoes and the humans, which ultimately poses increasing contact and enables exposure to new pathogens. Travelers, including a group of people who increase risk by traveling to areas they have never been, have also been exposed in more recent history to dengue viruses, the most reported arboviral infections in traveling/groups of travelers [25]. For example, mass air travel and incidental recreational tourism, which provide the avenue for prospective hosts from different parts of the world, are fundamental to increasing the global incidence of arboviruses, particularly dengue [26],[27]. Resources and infrastructures for effective medical care, especially in impoverished countries, and appropriate and innovative mosquito management strategies are lacking in most regions of the world, contributing to the risk of arboviral infection [28].

2.1. Client-server architecture

In the architecture of client-server, client destinations send the demand for the servers, which in turn order processing then send a reply to the requested customer. Such a demand started from the entity of the customer is called "Work Load". The size of this workload transmitted between the customer and server is "Traffic". The main functions of servers are [5]:

- Receipt of the customer's request for validation.
- Customer order processing.
- Send the reply to the requested customer

2.2. Load balancer

Requests of customer requests for work are allocated via the load balancer [6]. Orders are transmitted to the servers depending on the different mechanisms of load balancing. The structure of the load balancer is shown in Fig. 1.

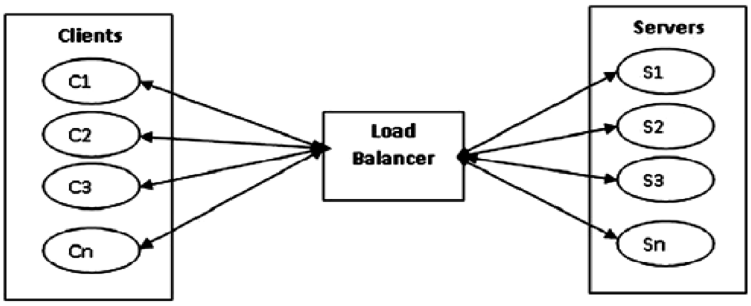

Fig. 1: Structure of Load Balancer (12).

3. Load balancing types

Servers' Load balancing is executed in various techniques. These techniques are installed on the load balancer based on load balancing. Various methods are used for the load assignment between the ready servers [7] [5].

3.1. Methods of static load balancing

In Static load balancing methods, the performances of server nodes are selected at the first step of jobs. After that, depending on their performance, the load of work is assigned via the main node. The nodes of the slave evaluate their assigned work and, after that, send their result to the master node.

3.1.1. Random allocation

In this technique, the client requests are allocated to a server that is picked randomly between a group of ready servers. In this state, any server perhaps allocate many job demands. The random selection ensured that every server received its share of the load of customers.
While Random Allocation is easy to implement, it does not consider the current load or capacity of servers. This can lead to overloading certain nodes, especially in heterogeneous environments. It is generally unsuitable for systems requiring high reliability or predictable performance.

3.1.2. Round-robin method

The main node of the load balancer allocates the jobs for an available list of servers based on rotation. The initial job is allocated for a server that is randomly selected from the available; the main node follows the ring arrangement for forwarding the incoming job. When a server is allocated to the job, this server is transferred to the last position of the list. This makes the workload evenly shared by ready servers. Round-Robin is effective in environments where servers have roughly equal capabilities. However, in real-world systems with varying performance characteristics, this method can lead to inefficient resource utilization. It is commonly used in DNS load balancing and simple web services.

3.1.3. Weighted round robin method

In the weighted round robin algorithm, assign a weight for every server inside the menu, if any server can handle two treatments; the capable server earns a weight for two. In these statuses, the main node assigns two jobs for the capability server whereas individual job is assigned for the weak servers.
Weighted Round-Robin improves upon Round-Robin by assigning more tasks to more capable servers. It is suitable for systems where performance asymmetry exists among nodes. However, the method still lacks dynamic feedback on actual server health or runtime load.

3.1.4. Central manager method

In this method [8], a major node picks a host for new process. A node of server that is less process if contrast with every remained nodes in the group of servers is picked while job is initiated. For picking a server that has less load, the load director uses system load information of the server that is saved in the main node. The load information is edited via whole the distant nodes of the server that are in the collection by sending a message each time if changes in the load on them. Every node of the server can edit its load case for the central director.
This method enables better load decisions by continuously monitoring server states. It is effective for moderate-sized systems but may become a bottleneck in large-scale distributed environments due to centralized control. Failures at the central node can impact the whole system.

3.2. Techniques of dynamic load balancing

These methods are based on load information of the system and the choice of the process to allocate to the nodes of the server at execution time. In the method of dynamic load balancing, the decisions of load balancing depend on the system's current case, and so workloads are dynamically transitioned from a node of overloaded node to a node of under-loaded for obtain faster replies from the server nodes. In this system reply for differences is the major characteristic for the dynamic load balancing. The load balancer continuously monitors the load on all system nodes. An important opinion is made for executing of controlling process via the load balancer.

3.2.1. Strategies in dynamic load balancing techniques

Dynamic Load balancing techniques are shown via the execution of the following Strategies [9][8]:

3.2.1.1. Information strategy

This Strategy is accountable for the summation of state information systems. It is determined the assignments of load information type, from where it is to be gathered, and when. The local case information of neighboring nodes is gathered as information of global case; gathering all the nodes' case information in the system is better for scheduling decisions.
The success of this strategy depends on the accuracy and freshness of the collected load information. If the system is large, the overhead for gathering and disseminating global state data can affect performance. However, it enables more informed decision-making compared to static methods.

3.2.1.2. Triggering strategy

This Strategy determined a suitable time for the start of a load balancing operation.
This strategy helps reduce unnecessary load balancing operations by controlling when redistribution is triggered. If set incorrectly, it may delay balancing actions or cause frequent overhead. It is particularly useful in systems with fluctuating but predictable loads.

3.2.1.3 Resource type strategy

This Strategy is the resource determining as a receiver of workloads or a server node, depending on the readiness case.
This strategy enhances flexibility by evaluating the suitability of each node before assigning workloads. It works well in dynamic systems with diverse resource capabilities but adds decision-making complexity at runtime.

3.2.1.4. Location strategy

This Strategy is picking the best node in the server from all the ready nodes in the system. Resource ready and service ready are several factors that are required for choosing a node of a server to execute the load execution.
Selecting the optimal node based on proximity and availability reduces latency and enhances performance in distributed networks. However, it may suffer from network topology changes and communication overhead if not updated dynamically.

3.2.2. Techniques of dynamic load balancing: centralized and distributed

This section shows techniques of dynamic load balancing that depend on the location of decision making, information that is used for the purpose of the scalability factor, the reciprocity, the profile information overhead, and the operation of decision making [9].

3.2.2.1. Technique of centralized dynamic load balancing

In Technique of Centralized Dynamic Load Balancing, the main node makes the information and the decision of load balancing for the purpose of load balancing, which is acquired from the slave nodes, which are remaining either after a Preset fixed period or demand basis. If the current state of the system is changing, then the load information is collected. This technique reduces the communication of operations is reduced for avoid the overhead of the network. Limited scalability is a disadvantage of this technique.
Centralized control simplifies the decision process and can lead to more balanced allocations initially. However, it introduces a single point of failure and limits scalability. This method is best suited for smaller, well-controlled networks.

3.2.2.2. Technique of distributed non-cooperative dynamic load balancing

In the technique of Distributed Non-cooperative Dynamic Load Balancing, the responsibility of distributing the load is shared among whole working server nodes, replacing a master node. The present load information is gathered based on the criteria of demand. if any node of the server changes its state from working to overloaded case, then this node must allocate its information about the present load to every remaining node so the present load can be reallocated to arbitrate the load system. This technique produces temperate scalability compression with the centralized technique. When any overloaded node is then, this node has allocated its information of present load to every remaining node before rearrangement of the system load, it may raise the traffic of the network for intercommunication of operation.
This technique distributes decision-making and improves fault tolerance. It is more scalable than centralized approaches but can generate higher network traffic due to frequent state exchanges. Performance varies depending on communication efficiency.

3.2.3. Technique of queue-based dynamic load balancing

In this technique, queues are predominantly preserved in the system of the load balancer to save the incoming loads.

3.2.3.1. Central queue technique

In the Technique of Central Queue, the new load jobs and the unrealized demands are saved on the master node in a cyclic FIFO queue. Every new load demand that comes to the manager of the queue is added to the queue. After that, whenever a demand of load demand is receipted via the manager of the queue, it the first workload of the queue and transmits it to the demanding node. If the queue doesn't have ready loads, then the job is stored until a ready of new ready load. When a new load comes in, the manager of the queue and at the same time, there are no serviced jobs in the queue, then this job is deleted from the queue and the new load is assigned to it [5].
Central queuing provides a clear and organized way to manage jobs, ensuring fairness via FIFO scheduling. However, the central queue can become a bottleneck as system size grows, reducing the benefit of parallelism in distributed systems.

3.2.3.2. Local queue technique

The technique of Local Queue is characterized by the dynamic operation migration support. The primary job for the local Queue technique is to statically allocate for every new job with migration of the job started via a host if its load falls below the limit of threshold. This is a user-known technique parameter. The parameter determines the minimal number of available jobs the load manager attempts to assign to every processor [5].

Local queues support better scalability and reduce the central point of failure. However, job migration between nodes may result in performance degradation if not properly managed, especially in systems with high communication latency.

3.2.4. Techniques of primary and centralized node-based dynamic load balancing

3.2.4.1. Technique of primary node-based load balancing

In the technique of Primary Node Based Load Balancing [10], either at the start phase, jobs are saved in a queue, or the jobs are assigned to the nodes as they come. When jobs are saved in a queue, then every job is assigned one via one for the basis nodes of the load balancing. Throughout the state, if the system becomes imbalanced in load, then jobs are moved from nodes of heavy load to nodes of light load. Migration of jobs has a significant impact on the bandwidth of the network and the load.
This approach allows for preemptive queue management and job migration to less-loaded nodes. While it enhances responsiveness, it requires constant monitoring, which can increase resource consumption and add to system complexity.

3.2.4.2. Technique of centralized node-based load balancing

There are many cases, a node of heavily loaded node can't detect the node of a lightly loaded node in its cluster, and because of network traffic, the node fails to find a node that is ready in a distant cluster. It would be suitable if a node of a heavily loaded cluster detects a temporary node on its cluster for processing the overload.
It provides a fallback mechanism when remote resources are unreachable, improving reliability in clustered systems. However, it still depends on efficient local resource detection and may not respond quickly in rapidly changing conditions.

4. Comparative study of techniques for load balancing in distributed systems

In this section, we show the comparison between dynamic load balancing techniques and static load balancing techniques [5].

4.1. Comparisons between techniques of dynamic load balancing and static load balancing

Different comparisons between dynamic load balancing techniques and static load balancing techniques are shown in Table 1and Table 2 [5] [11].

Table 1: Dynamic Load Balancing Techniques vs Static Load Balancing Techniques

Parameters	Dynamic	Static
Stability	Less	More
Complexity	More	Less
Implementation	Difficult	Easy
Flexibility	Less	More
Reliability	More	Less
Communication Overhead	More	Less
Adaptability	More	Less
Performance	More	Less
Resource Utilization	More	Less

Table 1 compares key parameters of dynamic and static load balancing techniques. As shown, dynamic load balancing outperforms static techniques in terms of adaptability, performance, and resource utilization. However, this comes at the cost of increased complexity and communication overhead. In contrast, static techniques are easier to implement and maintain due to their simplicity and predictability, but may underperform in dynamic or large-scale environments where workloads vary significantly over time. This trade-off is crucial when selecting a method for a particular application, especially in environments with real-time responsiveness or scalability needs.

Table 2: Comparative Analysis of Key Techniques Between Static Load Balancing Techniques vs Dynamic Load Balancing Techniques

Technique Type	Algorithm	Architecture	Key Mechanism	Features	Disadvantages
Static	Randomized	Decentralize method	Assigns jobs randomly	Easy of running	Inefficient for heterogeneous workloads
	Round Robin	Central method	Jobs are assigned periodically.	Low operating costs, Simple	override capabilities of the node
Dynamic	Least Connection	Central method	Assigns jobs to nodes that have the least effective connections	Suitable for web applications	Bottleneck at decision-making node
	Threshold-Based	Decentralize method	When a predefined threshold is reached, nodes stop accepting other jobs	Prevents overloading	Demands permanent monitoring
	Machine Learning-Based	Decentralize method	Uses AI to optimize distribution and predict load	High self-optimization, adaptability	Computationally expensive
	Agent-Based	Decentralize method	Intelligent agents balance loads and tracking autonomously	Fault-tolerant, scalable	Demand high inter-node communication

Table 2 presents a comparative analysis of individual load balancing techniques, highlighting their architecture types, operational mechanisms, and notable features. For instance, Randomized static methods are easy to run but inefficient for heterogeneous workloads, while Round Robin is simple and cost-effective yet can cause bottlenecks at the decision-making node. Dynamic techniques like Threshold-Based and Machine Learning-Based methods offer greater adaptability and fault tolerance but are computationally expensive and require continuous monitoring. These insights underline the importance of aligning technique selection with system requirements, such as node uniformity, network traffic tolerance, and the desired level of automation.

5. Conclusion

This study presented a comprehensive review and comparison of static and dynamic load balancing techniques used in distributed systems. We examined various algorithms—including Round Robin, Weighted Round Robin, Centralized Dynamic, and Machine Learning-Based approaches—highlighting their architectures, mechanisms, advantages, and limitations. The comparative analysis shows that while static methods are simpler and easier to implement, dynamic methods offer superior adaptability and performance in modern, scalable computing environments.

Despite these advancements, several research gaps remain unaddressed. First, most existing techniques are not optimized for resource-constrained environments, such as IoT and edge computing systems, where bandwidth and power limitations significantly affect performance. Second, dynamic approaches involving machine learning or agent-based strategies often suffer from high computational overhead, limiting their applicability in lightweight or latency-sensitive systems. Third, the current methods generally lack context-awareness, meaning they do not adapt well to user behavior, task criticality, or workload type in real time.

Future research directions should focus on:

- Developing lightweight, energy-efficient load balancing techniques for edge and IoT systems.
- Creating adaptive hybrid models that combine the predictability of static methods with the flexibility of dynamic ones.
- Integrating context-aware and user-behavior-driven mechanisms for real-time task allocation in distributed architectures.
- Investigating the use of federated learning or decentralized AI models to improve scalability and privacy in load balancing without increasing network overhead.
- Studying load balancing in emerging technologies, such as 6G, serverless platforms, and containerized microservices under Kubernetes or Docker Swarm.

References

[1] https://avinetworks.com/what-is-load-balancing/.

[2] Yixin Diao, Chai Wah Wu, Joseph L. Hellerstein, Adam J.Storm, Maheswaran Surendra, Sam Lightstone, Sujay.

[3] Raman a Ku m ar K., Mah e sh V. Ghatage, “Load Balancing of Services with Server Initiated Connections”, ICPWC, 2005.

[4] Branko Radojevic, Mario Žagar, “Analysis of Issues with Load Balancing Algorithms in Ho sted Cloud Environments”, Opatija, Croatia, MIPRO 2011, May 23-27, 2011.

[5] P. Beaulah Soundarabai* 1 , Sandhya Rani A. 1 , Ritesh Kumar Sahai 1 , Thriveni J. 2 ,K.R. Venugopal 2 and L.M. Patnaik, " Comparative Study on Load Balancing Techniques in Distributed SYSTEMS " , International Journal of Information Technology and Knowledge Management,December 2012, Volume 6, No. 1, pp. 53-60.

[6] Network Load Balancing in Microsoft Windows 2000 Advanced Server and Datacenter Server Operating Systems, http://technet.microsoft.com/en-us/library/bb742455.aspx.

[7] Md. Firoj Ali1, Rafiqul Zaman Khan2, "The Study on Load Balancing Strategies in Distributed Computing System" ,International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.2, April 2012https://doi.org/10.5121/ijcses.2012.3203.

[8] P.L. McEntire, J.G. O’Reilly, and R.E. Larson, “Distributed Computing: Concepts and Implementations”. New York: IEEE Press, 1984.

[9] Ardhendu Mandal and Subhas Chandra Pal, “An Empirical Study and Analysis of the Dynamic Load Balancing Techniques Used in Parallel Computing Systems”, ICCS 2010.

[10] Parveen Jain and Daya Gupta, “An Algorithm for Dynamic Load Balancing in Distributed Systems with Multiple Supporting Nodes by Exploiting the Interrupt Service”, International Journal of Recent Trends in Engineering, 1(1), May 2009.

[11] Mamta Kumari and Rakesh Kumar Katare, “A Comparative Study of Various Load Balancing Algorithms in Parallel and Distributed Multiprocessor Systems“, International Journal of Computer Applications, Volume 169, Number 10, 2017https://doi.org/10.5120/ijca2017914901.

[12] Soundarabai, Paulsingh & Sandhya, Rani & Sahai, Ritesh & K R, Venugopal & Patnaik, Lalit. (2012). Comparative Study on Load Balancing Techniques in Distributed Systems.