The CSI Journal on
Computer Science and Engineering
Vol. 20, No. 1, 2026
Pages 113-127
doi.org/10.22034/jcse.2025.561324.1069
Regular Paper

Performance Benchmarking of Load Balancers and Service Brokers in Heterogeneous Clouds using CloudAnalyst

Ali Yousefi Choubini

Department of Computer Engineering, Faculty of Engineering and Technology, Shahid Ashrafi Esfahani University, Isfahan, Iran

Abstract

Cloud computing's foundational challenge lies in efficiently distributing user requests across geographically distributed data centers (DCs) and subsequently across heterogeneous virtualized resources. This review paper systematically analyzes two critical components of this process: Service Broker Policies (SBPs), which determine DC selection, and Load Balancing (LB) Algorithms, which manage task distribution within Virtual Machines (VMs) in a DC. The study provides a detailed taxonomy and analysis of prevalent policies and algorithms, evaluating their mechanisms, strengths, and weaknesses. Furthermore, it presents a rigorous empirical performance analysis using the CloudAnalyst simulation tool. Three representative algorithms Round Robin (RR), Equally Spread Current Execution (ESCE), and Throttled LB (TLB) are evaluated under two distinct SBP: Closest DC and Optimized Response Time. The simulation models a realistic, heterogeneous cloud infrastructure subjected to dynamically varied workloads. Empirical results demonstrate that the TLB algorithm provides the most robust performance, characterized by highly competitive average response times and exceptional stability, as evidenced by its consistently minimal maximum latency. This is achieved through a proactive, capacity-aware distribution strategy that strategically directs workloads toward high-performance VMs. In stark contrast, the static RR algorithm proves fundamentally unsuitable for heterogeneous environments, incurring severe performance degradation and unstable response times due to its inability to account for disparate VM capacities. Although the SBP exerts a measurable influence on performance, the analysis conclusively establishes that the selection of the load balancing algorithm is the paramount factor in optimizing overall system performance, quality of service (QoS), and resource utilization in heterogeneous cloud environments.

Keywords: Cloud computing, Load Balancing, Service broker Policy, CloudAnalyst, Heterogeneous Clouds, Resource Management, Quality of Service (QoS)

1. Introduction

Cloud computing (CC) has gained significant popularity as a technology that enables users to conveniently access a shared pool of computing resources. These resources, which include networks, servers, storage, applications, and services, are configurable and available on a pay-per-usage basis [1]. CC is a technology that enhances businesses worldwide as it aims to reduce hardware costs [2]. The primary aim of the CC is to use the distributed different resources effectively in order to attain high throughput, performance and efficiency [3]. In CC, resource delivery is accomplished with the help of services. while as the former comes under category of Infrastructure as a service (IaaS) cloud, the latter two comes under headings of Software as a service (SaaS) cloud and platform as a service (PaaS) cloud respectively [4]. In IaaS model, the cloud offers elementary IT resources such as networking features,

computers, more flexibility, and control over the computing resources [3]. The PaaS model is responsible for providing scalable and elastic runtime environments for the user on demand. It also lets the user to host the execution of applications at its end [5]. The SaaS is a model which allow the clients to use and rent the applications from the provider without install it on their own PC [6]. The greatest advantage of CC is that a single user physical machine is transformed into a multi-user VMs [4].

1.2. Need for Load Balancing

Increasing of service request will affect the QoS, it can only be managed efficiently by utilizing the workload in the DC resources. Resource allocation and balancing of workload in the DC are important aspects of efficient resource utilization [7]. As technology growing faster, there are huge amount of user on internet so managing and fulfil their requirement, load balancer come into the picture which essentially ensure that they get spread workload equally to the all-available server without any delay which help to accomplish a high user satisfaction, Maximum throughput with minimum response time [8]. LB also aims to provide scalability and flexibility for those applications whose size may increase in future and requires more resources as well as to provide priority to jobs that need instant execution as compared with another jobs [3].

1.3. Importance of Service Broker Policy (SBP)

It is extremely important to efficiently utilize CC's available resources. One such resource is DC, which is selected by a CSB. The CSB manages the routing between the DC and the user in the CC environment [1]. The SBP consists of some broker policies which helps to provide a DC for upcoming request. It also provides a common and unique interface through which users can provision and manage their management across multiple clouds [7]. Since the main goal of the SBPs is to direct the user requests to the best DC with optimal performance, the SBP has to efficiently select the best DC for the job considering many factors such as time, cost, and availability [9].

1.4. Motivation for Study

While substantial researchs has been dedicated to evaluating SBP and LB algorithms, a significant methodological gap persists in the prevailing evaluation paradigms. Foundational surveys, such as those by Al-E'mari et al. [1] and Amrita Jyoti et al. [7], have effectively taxonomized these strategies and highlighted their critical role in ensuring QoS. Furthermore, numerous empirical studies by Mahalle et al. [10], Meftah et al. [11], Shahid et al. [12], and Payaswini P. [13] have provided valuable comparative analyses, predominantly using the CloudAnalyst simulator. However, a common limitation across these influential works is their reliance on simplified and often homogeneous simulation models. These studies typically evaluate algorithms under controlled conditions with synthetic, uniformly distributed User Bases (UBs) and DCs that lack true heterogeneity in hardware performance, network capability and lack of heterogeneous workloads (comprising heterogeneous user requests like the real world). This approach fails to capture the complex and highly variable nature of real-world "cloud-of-clouds" environments, where workloads are inherently heterogeneous and infrastructure is disparate. Consequently, the performance rankings and conclusions drawn from these studies, while insightful, may not be robust or directly translatable to a more realistic, heterogeneous cloud ecosystem. Therefore, a critical need exists for comprehensive evaluation research that employs a real heterogeneous world workload and simulates a complete, heterogeneous cloud environment to rigorously assess the efficacy and resilience of different SBPs and LB algorithms under different conditions that accurately mirror operational reality.

1.5. Research Objectives

This research aims to conduct a comprehensive performance analysis of SBPs and LB algorithms within heterogeneous cloud environments. The primary objective is to empirically evaluate three LB algorithms, RR, ESCE, and the TLB algorithm, when deployed under two distinct SBPs: the Closest DC policy and the Optimized Response Time policy. The study will utilize the CloudAnalyst simulation tool to model a realistic, geographically distributed cloud infrastructure under heterogeneous workloads. The performance of these combinations will be measured and compared based on key metrics, including overall response time, DC processing time, and operational cost, to determine optimal configurations for efficient cloud service delivery.

2. Essential Concepts

This section begins by defining the concept of LB and surveying several of the most common LB algorithms. Subsequently, the role of the SBP is elucidated, emphasizing its critical function in cloud environments. Finally, a classification and discussion of different SBP types is provided.

2.1. Load Balancing

Balancing a scientific workflow in a cloud platform is an optimization process to share the running loads or split the loads over the available resources [14]. Improvement in CC can be achieved by the process of CC with the LB [15]. In the cloud environment whose main objective is to effectively manage the workload among different cloud nodes, thereby preventing any node from being overloaded or underutilized [16]. LB is an optimization technique in which task scheduling is an NP hard problem. There are a large number of LB approaches proposed by re- searchers where most of focus has been concerned on task scheduling, task allocation, resource scheduling, resource allocation, and resource management [4]. LB model is shown in Figure 1.

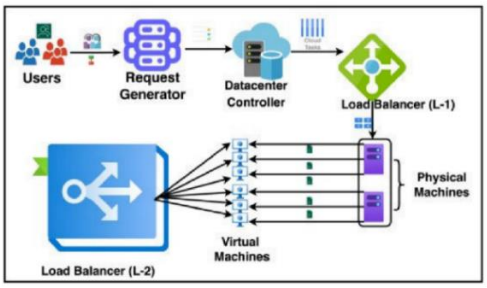

Figure 1. LB in CC [36].

2.1.1. Types of load balancing

Based on the system's state, LB algorithms initially are of three types viz Static, Dynamic and Hybrid [3,17, 18, 19, 20].

- **Static Load Balancing (SLB):** In SLB, the previous knowledge of the system is already known, which includes processing power, memory, performance and data on user requirements. These algorithms do not need current state information for process [7]. The static-based algorithms do not take into account dynamic changes to the load during runtime. Thus, the major drawback of these algorithms is low fault tolerance due to sudden changes in load [2]. Some static algorithms are Round Robin (RR), Min-Min and Max-Min algorithms, and Opportunistic Load Balancing (OLB) [21].
- **Dynamic Load Balancing (DLB):** The system current state is utilized by the DLB [15]. In these categories of algorithms, the VM information like capacity, memory size, performance, etc is known in advance and these details are considered for all the allocations. The change in this information during the run time do not affect the allotment of the task to VMs. Static algorithms are easy to incorporate but is not a good option for heterogeneous cloud environment [5]. Some of the dynamic algorithms include examples such as Ant Colony Optimization (ACO), Honey Bee, TLB and ESCE [21]. There are two types of algorithms, which are centralised (non-distributed) approach and distributed approach. n a centralised model, the sole node is in charge of the system's operation and dissemination. Additionally, not all nodes are accountable for this. In a distributed approach, each node independently constructs its own load vector by gathering load data from other nodes. Local load vectors are utilized to draw conclusions in close proximity. The distributed approach is suitable for widely distributed systems, such as CC [3, 18, 22].
- **Hybrid Load Balancing (HLB):** These are developed to overcome drawbacks of static and dynamic techniques by preserving their features and advantages [3], these algorithms are being used to aggregate the benefits and merits of static and dynamic algorithms in order to design a new one [23]. The qualities of hybrid approaches are inherited from static and DLB techniques, and efforts are made to overcome the drawbacks of both methods [17].

2.1.2. Common Parameters in Load Balancing

In this subsection, the metrics considered by most authors when reviewing the recent state of art are provided. These metrics are essential in designing and developing a LB algorithm. These metrics determine the algorithm's quality in terms of performance in cloud applications [2].

- **Throughput:** This metric is used to calculate the number of processes completed per unit time [21].
- **Makespan:** It denotes the overall duration needed to finish a specific set of tasks or jobs within a computing CC environment. Minimum makespan represents the efficiency and performance of the system in handling and processing tasks [24].
- **Response Time:** Amount of time taken by the algorithm to respond to a task. It takes into account the waiting time, trans- mission time, and service time. It is how much time is needed to respond to a user inquiry. Minimum RT is required in a good LB algorithm [2].
- **Scalability:** Sudden variations in the number of user requests and the computational load, should not affect the system's performance. The algorithm must be highly scalable for efficient LB [17].
- **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 [16].
- **Energy consumption:** It calculates the amount of energy consumed by all nodes. LB helps to avoid overheating and there- fore reducing energy usage by balancing the load across all the nodes [17].

2.1.3. Common load balancing algorithms

There is various common LB algorithms used to enhance the performance of CC. Some of the most common LB algorithm reviewed below:

Round Robin (RR): The RR is the most widely used and straightforward scheduling algorithm [16]. The RR LB technique is the simplest technique built and widely used for time-sharing systems [25]. RR Works in a circular and ordered procedure where each process is assigned a fixed time slot without any priority [2]. The procedure of VMs running in an RR model is depicted in Figure 2 [8].

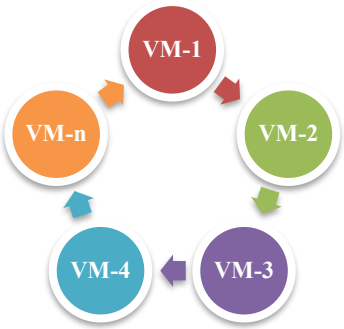

Figure 2. RR load balancer [8].

A common problem in LB is that after user requests, the allocation state of the VM is not saved and updated [2]. The RR without checking the capacity of sever, it's directly assigned the request (like whether it is over loaded or not) [8].

Weighted Round Robin (WRR): The WRR algorithm performs circular distribution based on the RR algorithm and relies on the capacity of each VM through its weight table to distribute the load to the VMs respectively [25]. When all weights reach the same level, servers will experience consistent traffic [19]. It is considered a good method and almost devoid of problems. But this depends on how you choose the right weight for the server, also, this technology requires the ability to guess processor engagement which is not possible to guess in networks due to the difference in packet sizes 'in press' [26]. In addition, this algorithm is great for estimating the waiting time, however, it does not consider different lengths of tasks to allocate for appropriate VM [20]. The WRR load balancer architecture is shown in Figure 3.

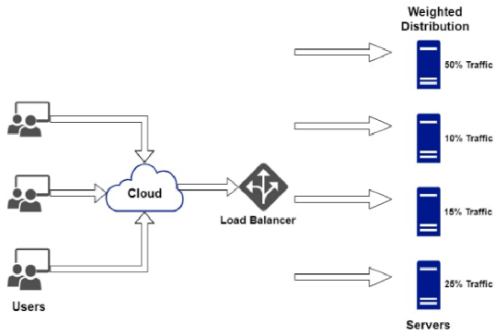

Figure 3. WRR load balancer [25].

Min-Min: This is an algorithm where it considers the minimum completion time for scheduling and balancing [2]. The job with the shortest anticipated completion time is then chosen and assigned to the node with the shortest execution time. The task has now been eliminated from the task list. This procedure is repeated until all task kinds are assigned to the same nodes. As a result, the algorithm improves if the smaller work is larger than the smaller task [22].

Max-Min: The max-min algorithm in question is identical to the min-min algorithm. The Max-Min method begins with a list of all the jobs that have been submitted but are not yet allocated to any nodes. The shortest completion time for each sort of work that is accessible is determined at the beginning. The resource with the shortest response time is then assigned the process with the longest execution duration. [22].

Equally Spread Current Execution Time (ESCE): This algorithm uses the concept of a distributed spectrum and operates in such a form that the number of available activities in each VM is equal at any moment [12]. It distributes the load randomly by first checking the size of the process and transferring the load to VMs that are only lightly loaded or which can handle the task easily in a small amount of time while maximizing throughput. This algorithm is a DLB algorithm that determines the priority by checking the size of the process. It requires a load balancer that monitors the jobs to be executed [11]. ESCE always find least loaded VM for assign new incoming request but it will not check whether it's previously utilized or not (so some VM over utilized and some is still ideal) [8]. The ESCE load balancer architecture is shown in Figure 4.

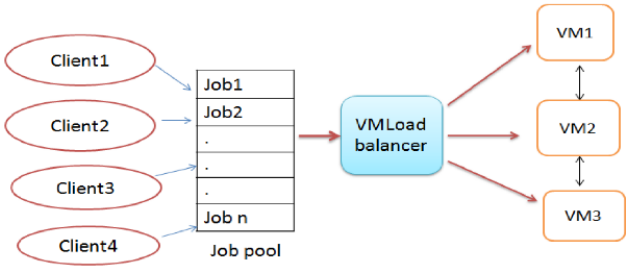

Figure 4. ESCE load balancer [16].

Throttled Load Balancing (TLB): The TLB is one of the approaches for balancing workflows in dynamic environments [14], which implements the throttled adjustable mechanisms [20]. The TLB algorithm balances the load by maintaining a configuration table of VMs and their status [25]. The algorithm uses two VM states based on which the algorithm is performed that are Available/Busy. This state is based on the allocation of VM to the request, is allocated or not. A hash table maintained at load balancer will contain this state information. ID and status (Available/Busy) of the VM are the two parameters of a hash table. At the beginning every VM is on available state [27].

In order for the DC 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 DC controller is contacted with the ID of any free VMs found so that it can be given requests [16].

Table 1. Allocation / Index Table [28]

VM	State
VM_1	Available
VM_2	Busy
VM_n	Available / Busy

Index table that includes all VMs state is shown in Table 1. In TLB algorithm, where the index table is resolve from the first index every time when the DC queries load balancer for allocation of VM. It does not take into account the advanced LB requirements such as processing times for each individual requests [8].

Ant Colony Optimization (ACO): It is a meta-heuristic technique for finding optimized solutions to multimodal optimization problems [17]. This algorithm exhibits behaviours resembling that of actual ants. It is primarily based on ants' ability to find the best route from their shell to the source. When a demand is initiated in the ACO process, the ant begins to move. Ants start at the root node and move to adjacent nodes, checking each one to see if it is overloaded or underloaded as they go. Ants alert the pheromone table, which stores the information about each node's operation when they move towards the network [22].

Honey Bee Algorithm (HB): The idea of this is that a group of bees known as the foraging bees, spread to look for food sources and send location information to the other bees. This is known to solve decision making and classification problems with more robust and flexible ways. Similarly, in a cloud environment, there are various changes of demand on the servers, and services are allocated dynamically and VMs should be utilized to the maximum limit reducing the waiting time. Honey bee behaviour can be mapped to a cloud environment [2].

Table 2. LB methods comparison [7, 21]

	Static	Dynamic	Centralized	Distributed
RR	✓		✓	
WRR	✓		✓	
Min-Min	✓		✓	
Max-Min	✓		✓	
ESCE		✓	✓	
TLB		✓	✓	✓
ACO		✓		✓
HB		✓		✓

Table 2 shows the comparison between reviewed LB methods in static, dynamic, centralized and distributed parameters.

2.2. Service Broker Policy (SBP)

The SBP is used to determine which DC serve the requests from each user to control the routing traffic problem between the DC and UBs. The SBP consists of some broker policies which helps to provide a DC for upcoming request. It also provides a common and unique interface through which users can provision and manage their management across multiple clouds [7]. SBP objective is to ensure uninterrupted processing of user requests by distributing service instances across relevant DCs in the event of a failure in any specific DC. In essence, the SBP directs user requests to the most suitable DC situated in various regions worldwide [1].

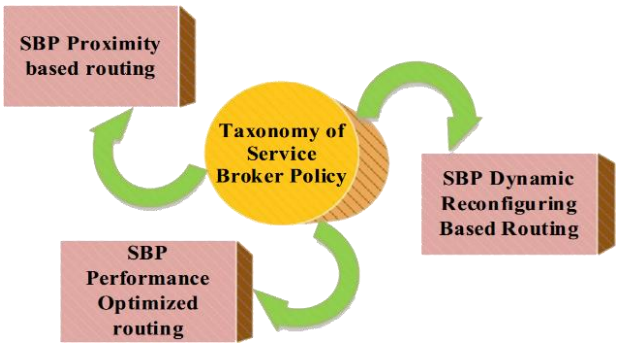

Figure 5. Types of SBP [7].

As shown in Figure 7 SBPs are categorized in three groups.

2.2.1 Type of Service Broker Policy

The layout of the VM to the physical machine in the DC is termed deployment and is a very critical part of the DC broker. Each could consider what kind of situation the SBP operates in [12].

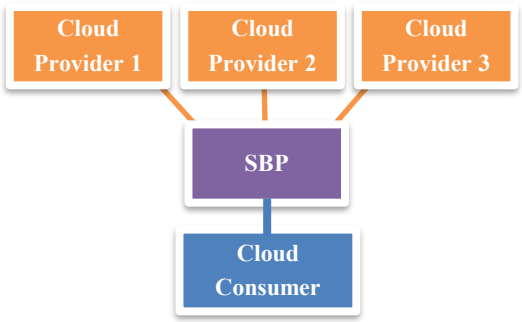

Figure 6. Role of SBP [29]

Closest DC Policy: The default routing policy routes traffic to the Closest DC in terms of network latency from the source UB [11]. The Closest DC Policy method can provide service only in shortest route between DC and user requests. It depends on the latency of the network selected to provide the path. Based on this route, the SBP routes traffic to the Closest DC considering trans- mission latency. It has mainly four phases: evaluation, filtering, selection and recommendation. These service stages are used to match the user request to the data providers. If there is no matching, results between the user's demands, it redirects the demands to other SBPs [7]. Closest DC Policy routing is shown in Figure 7 and following by its pseudocode in Figure 8.

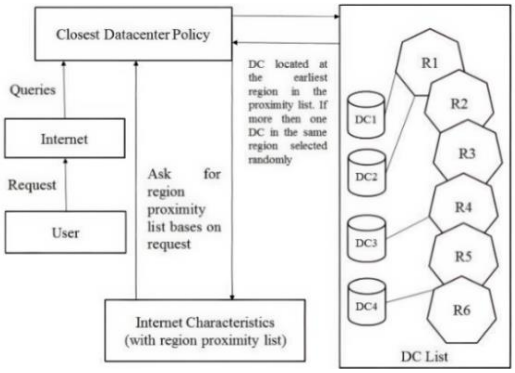

Figure 7. SBP Proximity-based routing [8].

Algorithm 1: SBP Closest DC Method

```
1.  Input: Incoming user requests with region information
2.  Output: Assign each request to a DC with minimum latency
3.  Begin
4.    while new requests are received do
5.      for each incoming request r do
6.        Find the set D of DCs closest to r (min latency)
7.        if |D| = 1 then
8.          Allocate r to the single closest DC d in D
9.        else
10.         Randomly select one DC d from D, Allocate r to d
11.       end if
12.     end for
13.   end while
14. End
```

Figure 8. SBP Closest DC pseudocode adopted from [30].

Optimized Response Time Policy: This SBP regularly tracks the efficiency of all DCs in this routing policy and is dependent on the direct bottleneck to the DC with the best reaction time [12]. It is routing performance is based on VM allocation mechanisms. This used directly by the requested user, the general performance of this broker routing is to determine the optimal solution of the VM service and the number of objective functions. This service policy is executed by the Optimized Response Time Policy, which has some steps for this policy [7]:

1) The best SBP response time keeps a main index of all available DCs.
2) When the message is received by a user, query the best processing time for the destination DC controller.
3) The Closest DC is identified by a brokering algorithm.
4) If the Closest DC has minimum response time, the best response time are selected from the Closest DC.

Dynamic-reconfiguring based routing: This policy works in terms of dynamic workload, as it is important to balance the load in CC environment [10]. This is an extension to Closest DC Policy where the routing logic is similar. But it has one more responsibility of scaling the application deployment based on the load it is facing [8]. In this algorithm, the routing logic is similarly based on proximity routing, but resizing the distribution of application is the main responsibility to face the overloading problem. The dynamic policy has one peculiar function to develop the number of VMs allocated in the DCs. This will be done by taking workloads into account the present processing times and achieve best solution in low processing time [7]. This approach helps maintain fast response times and efficient resource utilization [31].

3. Related Works

Al-E'mari et al. [1] in 2023 proposed a comprehensive survey on Cloud Service Broker (CSB) policies for DC selection in CC. The paper highlights the CSB's critical role in optimizing DC selection to meet QoS demands and prevent performance bottlenecks. It introduces a taxonomy classifying CSB policies into those based on service providers' interests, such as energy consumption and profit, and those focused on users' QoS concerns, including time, cost, and availability. The authors systematically analyze and compare various DC selection techniques, identifying their strengths and weaknesses, and outline significant research challenges and future directions, such as the need for real-time, adaptive, multi-objective, and large-scale DC selection strategies.

Mahalle et al. [10] in 2015 introduced an experimental study implementing SBPs for DC selection in a CC environment. Using the CloudAnalyst simulator, the paper evaluates three policies: Closest DC, Optimized Response Time, and Reconfigure Dynamically with LB. The results indicated that while all three policies performed similarly in terms of user response time, the Closest DC and Optimized Response Time policies were more efficient in request serving time compared to the dynamic load reconfiguration policy, which showed instability. The authors concluded by suggesting a future hybrid policy that combines proximity-based selection with LB to improve overall performance.

Meftah et al. [11] in 2018 conducted a performance analysis of SBPs and LB algorithms for large-scale internet applications in cloud DCs. Using the CloudAnalyst simulator to model a Facebook-like application, the study evaluated the Closest DC and Optimized Response Time broker policies alongside RR, ESCE, and TLB algorithms across six different DC configurations. The results demonstrated that the TLB algorithm consistently achieved the shortest overall average response and processing times. Furthermore, the research found that a geographically distributed DC configuration, when combined with the Optimized Response Time policy, yielded the best performance, providing valuable insights for cloud service providers and application designers on optimizing infrastructure.

Amrita Jyoti et al. [7] in 2020 presented a systematic survey and taxonomy of LB and SBPs in CC. Their work delivered a comparative and comprehensive study of newer methods developed between 2015 and 2018. They classified various LB algorithms into static and dynamic categories and detailed predominant techniques like RR, Weighted Least Connection, and metaheuristic methods including Genetic Algorithm and ACO. For cloud service brokering, they analyzed policies such as service proximity, performance-optimized, and dynamic reconfiguring-based routing. The survey also included a performance comparison based on metrics like response time, throughput, and makespan, discussed simulation tools, addressed security concerns, and outlined open challenges and future research directions for both domains.

Shahid et al. [12] in 2023 presented a performance evaluation of LB algorithms such as Particle Swarm Optimization (PSO), RR, ESCE, and TLB for CC using the CloudAnalyst simulator. They investigated these algorithms under three different SBPs, which are Closest DC, Optimized Response Time, and Reconfigure Dynamically with Load, by measuring metrics like overall response time, DC processing time, and cost. The findings indicated that PSO provided superior performance for optimized response time in certain configurations, while RR and ESCE delivered similar results in other cases, and the TLB algorithm demonstrated higher overhead during dynamic reconfiguration. The study underscores the importance of selecting appropriate algorithm and policy combinations to enhance cloud service efficiency and proposes the integration of intelligent techniques like machine learning for future SBP development.

Payaswini P. [13] in 2022 conducted a comparative study of various LB and SBP algorithms in CC using the CloudAnalyst simulation tool. The investigation evaluated traditional algorithms like RR, TLB, and ESCE alongside nature inspired methods such as HB, PSO, and ACO. These were tested under three SBPs, which are Closest DC, Optimized Response Time, and Reconfigure Dynamically with Load, across different scenarios with varying system loads and DC configurations. The findings indicated that the TLB and ESCE algorithms generally delivered the best response time when paired with the Optimized Response Time or Closest DC broker policies. In contrast, the PSO algorithm consistently achieved the lowest service cost across all tested scenarios. The study concluded that the optimal combination of LB algorithm and SBP is highly dependent on the specific workload and infrastructure configuration.

El Karadawy et al. [32] in 2020 conducted an empirical analysis of LB and SBPs in CC using the CloudAnalyst simulator. They evaluated three LB algorithms, which are RR, TLB, and Active Monitoring, under three SBPs, namely Closest DC, Optimized Response Time, and Dynamic Reconfigured. The analysis revealed that the TLB algorithm achieved the lowest average response time, particularly when combined with the Closest DC SBP. The study highlighted the importance of selecting appropriate algorithms and policies to enhance cloud system performance and provided insights for future work on dynamic LB.

Rawat et al. [33] in 2016 examined LB in cloud-based applications using various SBP through the CloudAnalyst simulator. They modeled an international bank's Internet banking scenario with a RR LB algorithm and tested it with different numbers of VMs under three SBPs: Closest DC, Optimized Response Time, and Reconfigure Dynamically. The analysis revealed that the Closest DC policy, when used with RR LB, achieved the lowest average response time and DC processing time, offering practical guidance for efficient cloud resource configuration and deployment in real-world applications.

4. Experimental Setup and Simulation

Researchers in CC need to perform actual experiments in their studies, setup, implementation, and experiments in the cloud; they often have a massive cost of creating a real cloud environment. Using models and simulation software to replicate cloud environments and perform prerequisite testing is one viable alternative [34]. There are a multitude of open-source cloud environment simulator tools such as CloudSim, CloudAnalyst, CloudReport, CloudExp, and GreenCloud, available for simulating the LB strategies in an IaaS model [35].

4.1. CloudAnalyst Simulator Tool

CloudAnalyst is a graphical user interface-based simulator that provides support for the evaluation of social network tools based on the geographic distribution of users and DCs [36]. Some of the CloudAnalyst features are: (1) Integrating cloud application analytics units into CloudSim; (2) geographically dispersing computer servers and client workloads for assessment using simulation in a large-scale cloud framework; (3) SBs maximizing the efficiency of software and service suppliers; and (4) the main units being composed of the DC, server, server broker, host, VM, and cloud coordinator [34].

The primary parts of CloudAnalyst are the Internet, UB, Regions, DC Controller, VM Load Balancer, and SBPs, as shown in Figure 9. There are seven core components present in the CloudAnalyst tool as follows [35]:

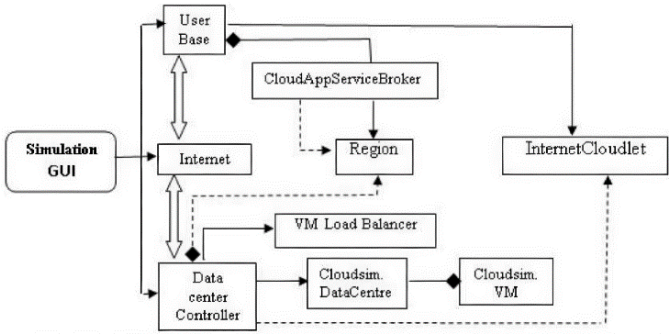

Figure 9. High-level architecture of CloudAnalyst [8].

- **Region:** In CloudAnalyst, the world is geographically distributed into six regions (continents). All the UBs and DCs associated with any one of these regions.
- **UB:** Each UB consists of a collection of users who need to execute loads (HTTP requests).
- **Internet:** It represents the real-world internet, which routes web traffic by the UB to the DC Controller. The Internet also specifies the traffic delay among the UBs and the DCs.
- **DC Controller:** It initiates the DC creation process, obtains the application given by the UB through the Internet, and routes the user's workload to the DC and the load balancer based on the criteria given in the LB algorithms.
- **Service Brokering:** Service Brokering selects the DC to which the user's requests should be forwarded. On the basis of the SBP, it selects the appropriate DC.
- **Internet Cloudlet:** Each user's load has been grouped into a single Cloudlet.
- **VM Load Balancer:** It assigns each Cloudlet to the proper VM for processing the Cloudlet

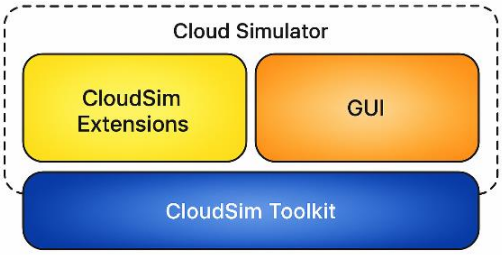

Figure 10. CloudAnalyst architecture [37].

The architectural design of CloudAnalyst is demonstrated in Figure 10 which expands over CloudSim toolbox [37].

4.2. Simulation Setup

To configure a simulation in CloudAnalyst, you must define five core elements: the cloud DCs and their hardware, the UBs that create the workload, the SBP, and the LB Policies (LBPs) [35].

Two scenarios were designed to simulate real-world heterogeneous cloud environments. In both, the physical servers and VMs were configured heterogeneously. The UBs and DCs configurations are detailed in Tables 3 and 4. Also, DC hardware is shown in Table 5. The VM instances were categorized by performance (Low, Medium, or High) and RR manner creation, with their specifications provided in Table 6.

The following parameters are configurable in CloudAnalyst's simulator advanced settings:

- Simulation duration: 24 hours.
- SBP: Optimized Response Time / Closest DC.
- User grouping factor in UBs: 100
- Request grouping factor in DCs: 100
- LBPs: the TLB, RR, and ESCE as predefined in the CloudAnalyst simulator tool.

The executable instruction length was varied in a RR manner between 50 and 300 bytes across different experimental scenarios (see Table 7) to simulate real-world workloads.

Table 3. UB configurations

UB	Region	Request Per User Per hour	Data Size Per Request (byte)	Peak Hour Start	Peak Hour End	Average Peak Users	Average Off-Peak Users	Scenario
1	R3	70	100	01	03	200,000	20,000	1
2	R2	70	100	20	22	200,000	20,000	
3	R4	70	100	21	23	75,000	7,500	
1	R0	75	100	13	15	350,000	35,000	2
2	R1	75	100	15	17	450,000	45,000	

3	R2	75	100	20	22	300,000	30,000	
4	R3	75	100	01	03	200,000	20,000	
5	R4	75	100	21	23	150,000	15,000	
6	R5	75	100	09	11	85,000	8,500	

Table 4. DC configurations

Name	Region	Arch	OS	VMM	No. VMs	Scenario
DC1	R3	x86	Linux	Xen	15	1
DC2	R2	x86	Linux	Xen	13	
DC3	R4	x86	Linux	Xen	9	
DC1	R0	x86	Linux	Xen	50	2
DC2	R1	x86	Linux	Xen	21	
DC3	R2	x86	Linux	Xen	24	
DC4	R3	x86	Linux	Xen	31	
DC5	R5	x86	Linux	Xen	16	
DC6	R4	x86	Linux	Xen	14	

Table 5. Physical hardware for each DC

Memory (Mb)	Storage (Mb)	Available Bandwidth	Number of Processors	Processors Speed	VM Policy
204,800	100,000,000	1Gb/s	16 CPU	40,000 MIPS	Time-Shared
204,800	100,000,000	1Gb/s	8 CPU	20,000 MIPS	Time-Shared

Table 6. Configuration of heterogeneous types

Name	Ram	Image Size	No. PEs	Bandwidth	MIPS
Low-Performance	512	10,000	1	1,000	1,000
Medium-Performance	1,024	20,000	2	1,000	2,000
High-Performance	2,048	40,000	4	1,000	4,000

Table 7. Executable Instruction Lengths and output size in two scenarios.

	Scenario 1			Scenario 2		
Executable Instruction Lengths (byte)	75	150	225	100	200	300
Output size (byte)	2,000					

4.3 Performance Metrics

The efficiency of LB algorithms and SBPs is evaluated using key metrics, including processing time, response time, load distribution [17, 18, 38].

4.3.1. Processing Time

In CC, processing time is a fundamental performance metric. It measures the interval from when a DC receives a task from the load balancer to when it completes the processing. This metric is a direct indicator of a DC's computational efficiency and performance [16].

4.3.2. Response Time

Response time is time the system takes when it responds to the end-user after receiving the user request. It integrates transmission time, waiting time, and service time. It is computed as shown in Eq. (1) [17, 18]:

$$R^{Time} = T + W + S \quad (1)$$

Where R^{Time}: Response Time, T: Transmission Time, W: Waiting Time, S: Service Time (execution time).

4.3.3. Load Distribution

Load distribution is calculated using Eq. (2), quantifies the percentage of the total system tasks allocated to each VM [38].

$$Load\ Distribution_{(vm_i)} = \frac{No.allocated\ tasks_i}{Total\ No.Tasks} \times 100 \quad (2)$$

As seen in Eq. (2), Load Distributing parameter shows the percentage of tasks allocations to *i*-th VM in DC.

4.4. Result and Findings

This section evaluates the RR, ESCE, and TLB methods under two SBPs (Optimized Response Time and Closest DC), assessing their average response time, processing time, and load distribution across diverse workloads in CloudAnalyst simulator tool. The performance for the two test scenarios is summarized below, enabling a direct, statistically-backed comparison of the evaluated schemes.

Performance Metrics: Average processing times and response times are detailed in Tables 8 and 9, with corresponding visualizations in Figures 13 and 14.

Load Distribution: The resulting workload allocation across heterogeneous VMs is presented in Table 10.

Also Figures 11 and 12 shows the response time of the TLB method under the second test scenario under two different SBPs (optimized response time and closest DC).

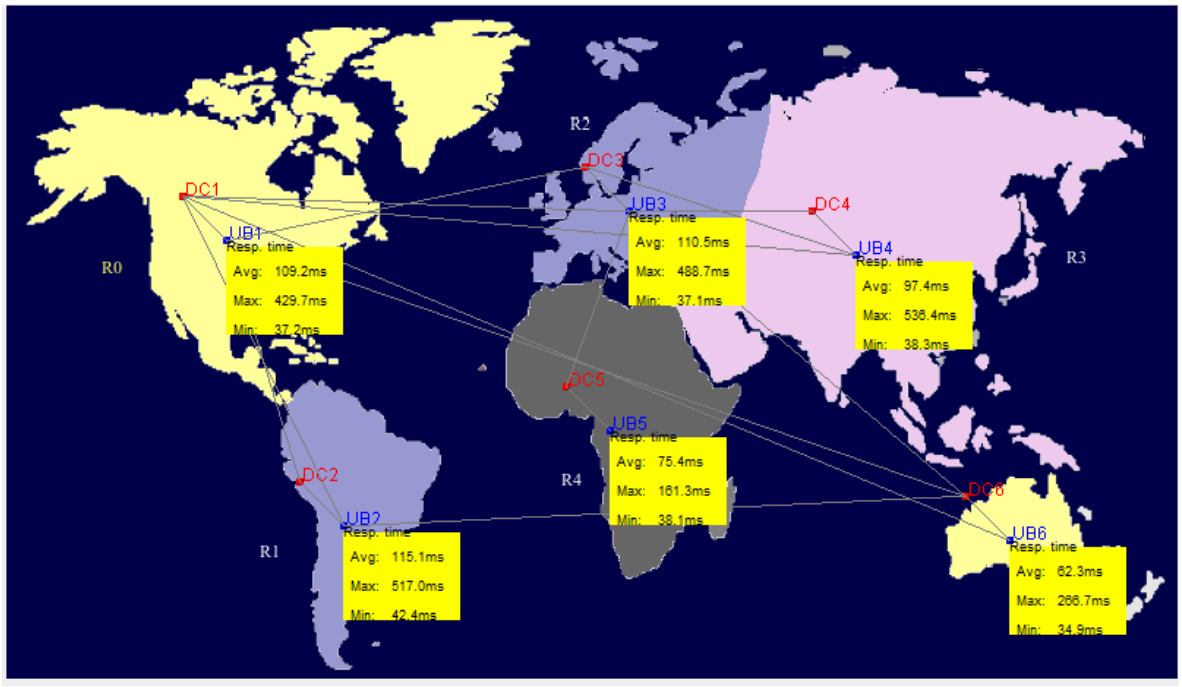

Figure 11. Detailed view of UBs response times for the Optimized Response Time Policy (scenario 2)

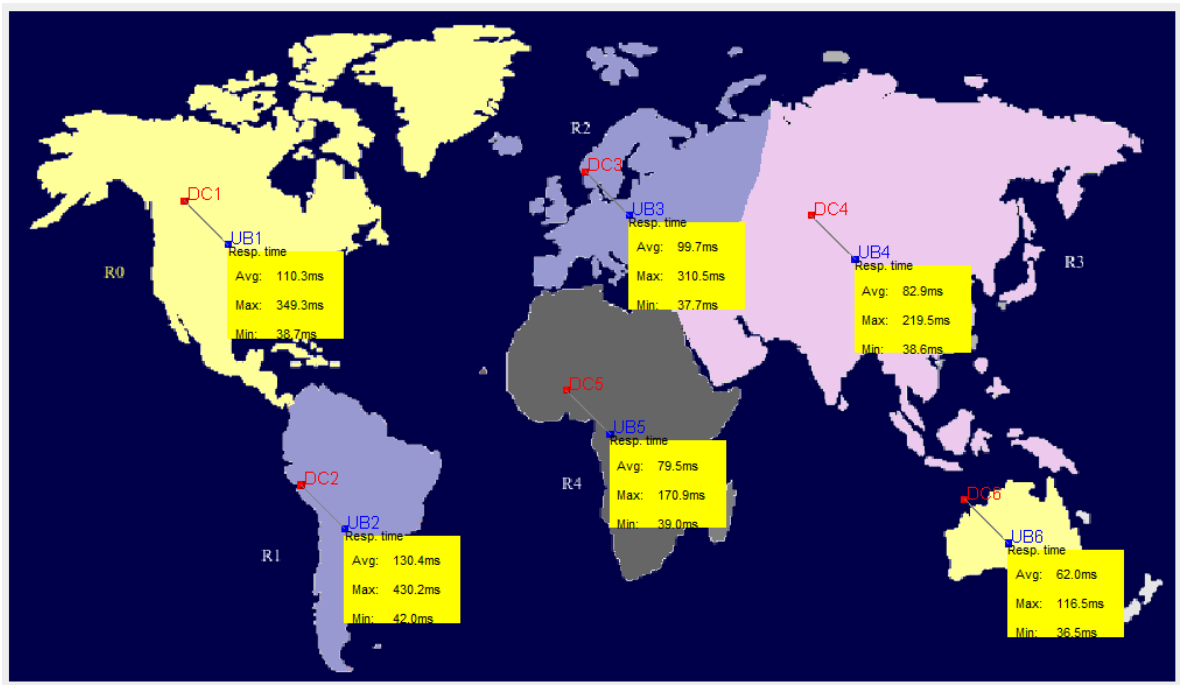

Figure 12. Detailed view of UBs response times for the Closest DC Policy (scenario 2)

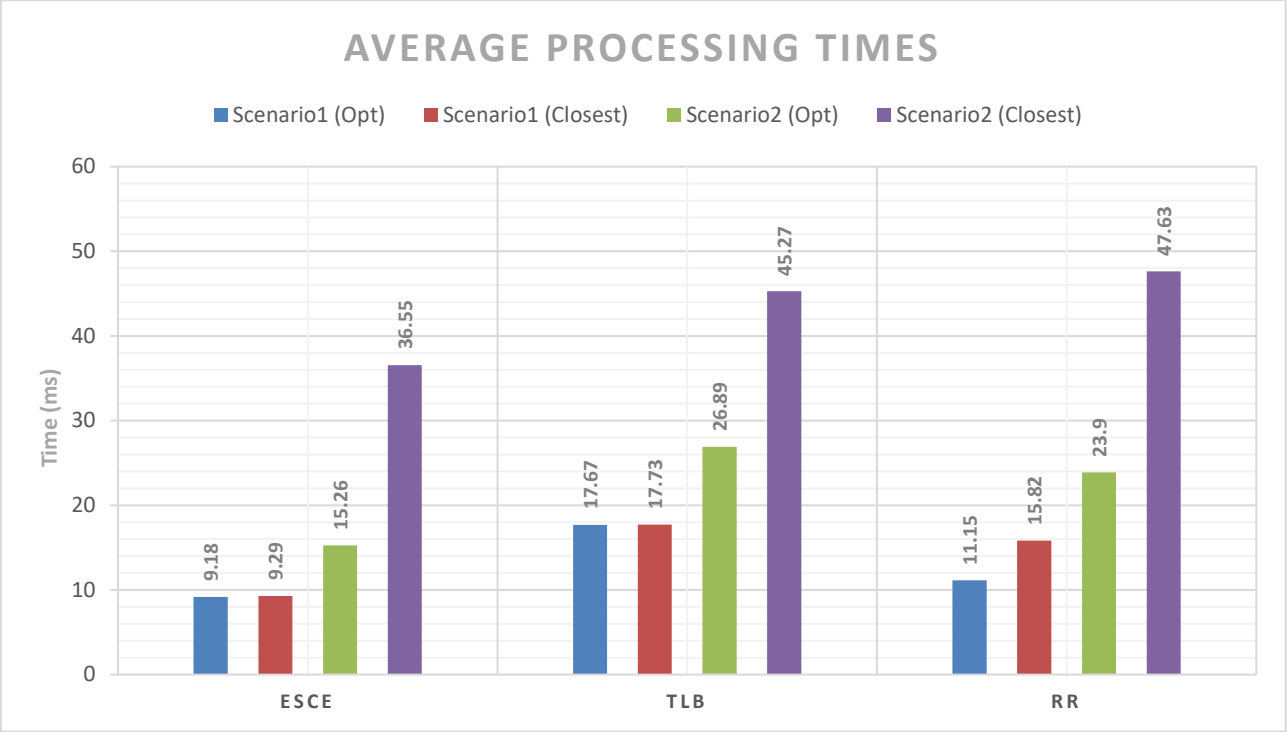

Figure 13. Average processing time in experimental scenarios

Table 8. Comparison of average, minimum, and maximum processing times

	Average (ms)	Min (ms)	Max (ms)	Scenario	SBP
RR	11.15	0.06	55537.92	1	Opt
ESCE	9.18	0.06	227.44		
TLB	17.67	0.06	101.75		
RR	15.82	0.07	50393.99		Closest
ESCE	9.29	0.06	234.00		
TLB	17.73	0.07	101.87		
RR	23.90	0.03	57113.27	2	Opt
ESCE	15.26	0.03	61434.60		
TLB	26.89	0.03	239.69		
RR	47.63	0.06	55194.89		Closest
ESCE	36.55	0.04	54713.70		
TLB	45.27	0.04	296.38		

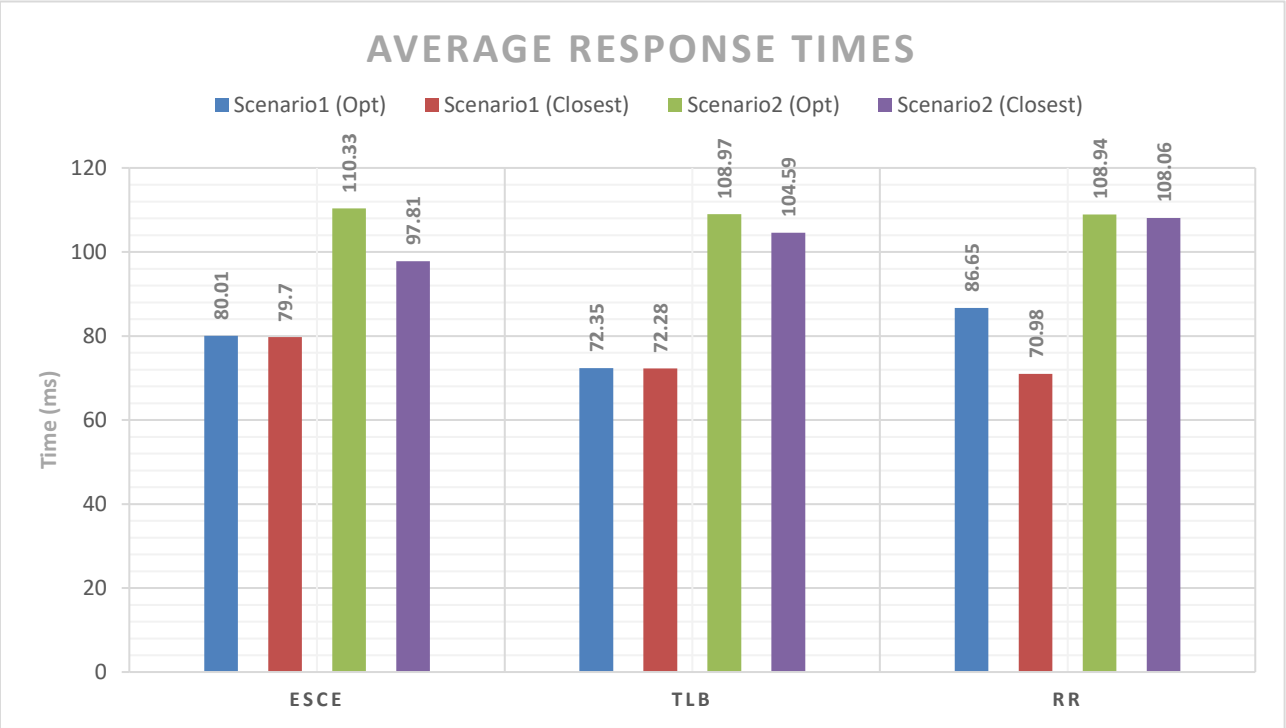

Figure 14. Average response time in experimental scenarios

Table 9. Comparison of average, minimum, and maximum response times

	Average (ms)	Min (ms)	Max (ms)	Scenario	SBP
RR	86.65	35.73	55593.60	1	Opt
ESCE	80.01	35.38	407.47		
TLB	72.35	35.38	364.15		
RR	70.98	34.70	50446.11		Closest
ESCE	79.70	35.38	405.19		
TLB	72.28	34.70	168.91		
RR	121.37	34.92	57168.12	2	Opt
ESCE	110.33	34.92	61481.24		
TLB	108.97	34.92	508.14		
RR	108.06	34.48	55248.06		Closest
ESCE	97.81	36.48	54764.51		
TLB	104.89	36.48	430.18		

Table 10. Ratio of Requests Sent to VM Types

	Scenario 1			Scenario 2			
	HC [a]	MC [b]	LC [c]	HC	MC	LC	SBP
TLB	28.28%	34.35%	37.37%	28.35%	34.78%	36.87%	Opt
ESCE	35.25%	35.02%	29.73%	40.43%	34.40%	25.17%	
RR	33.33%	33.33%	33.33%	32.73%	33.29%	33.98%	
TLB	24.05%	34.66%	41.29%	26.60%	34.78%	38.62%	Closest
ESCE	35.09%	34.75%	30.16%	41.46%	33.80%	24.74%	
RR	32.27%	32.27%	35.46%	32.31%	33.46%	34.23%	

[a] High-Performance, [b] Medium-Performance, [c] Low-Performance VMs

5. Analysis and Discussion

The results demonstrate a clear performance hierarchy among the LB algorithms and reveal the secondary, yet notable, influence of the SBP.

The TLB algorithm delivered the most robust and stable performance across all experimental scenarios. While it did not consistently achieve the absolute lowest average response time, it maintained highly competitive averages while exhibiting exceptional stability, with maximum response times consistently remaining under 508 ms. This performance consistency is directly linked to its intelligent workload distribution strategy, which proactively skews traffic toward high-performance VMs (directing over 40% of requests to them) while minimizing load on low-performance VMs. This capacity-aware approach effectively prevents any single VM from becoming a critical bottleneck.

The ESCE algorithm demonstrated competent performance, achieving the best average response time in Scenario 2 under the Closest DC policy (97.81 ms). However, its significantly higher maximum response times (exceeding 61 seconds in some cases) indicate substantially less consistent performance under peak load conditions compared to TLB.

In contrast, the RR algorithm proved fundamentally unsuitable for heterogeneous environments. Its static load distribution strategy consistently overloaded low-performance VMs, resulting in catastrophic maximum response times exceeding 55 seconds. This occurred despite RR occasionally achieving competitive average response times, as seen in Scenario 1 under the Closest DC (70.98 ms).

The load distribution patterns provide the fundamental explanation for these performance characteristics. RR's static, distribution ignores VM capacity, inevitably causing critical bottlenecks on low-performance VMs. ESCE dynamically adjusts load distribution, spreading workload more effectively than RR but still allowing for potential imbalances that lead to sporadic high latencies. TLB excels by implementing a proactive, capacity-aware distribution that actively prevents overloading any single VM type, thereby ensuring consistent performance and minimizing response time variance.

Regarding SBPs, the Optimized Response Time policy generally provided a slight performance advantage over the Closest DC policy, particularly under heavier loads in Scenario 2. However, the empirical evidence conclusively shows that the choice of LB algorithm had a far greater impact on system performance and stability than the choice of SBP.

In conclusion, for optimal QoS in a heterogeneous cloud environment, an intelligent, dynamic load balancer like TLB is critical due to its stability and consistent performance, and should be paired with a performance-oriented SBP like Optimized Response Time for the best results.

6. Conclusion and Future Works

This study conducted a comprehensive performance evaluation of three LB algorithms (RR, ESCE, and TLB) under two distinct SBPs (Optimized Response Time and Closest DC) in a simulated heterogeneous cloud environment. The results lead to the definitive conclusion that the TLB algorithm delivers the most robust and stable performance for heterogeneous environments, characterized by highly competitive average response times and exceptional stability, as evidenced by its consistently minimal maximum response times. This performance advantage is attributed to its proactive, capacity-aware load distribution strategy that effectively prevents VM overloading and ensures consistent QoS. The ESCE algorithm also demonstrated competent performance, achieving the best average response time in certain configurations, though with less consistency than TLB as indicated by its higher maximum response times. Conversely, the static nature of the RR algorithm rendered it unsuitable for heterogeneous infrastructures, resulting in severe performance bottlenecks despite occasionally competitive average response times. Furthermore, while the Optimized Response Time SBP provided a marginal advantage, the choice of the LB algorithm proved to be the dominant factor in determining overall system performance.

For future research, the following directions are proposed:

(1) **Investigation under Highly Dynamic Workloads**: This study utilized heterogeneous workloads, but future work could focus on extreme scenarios with bursty, non-stationary traffic patterns (e.g., flash crowds). Evaluating the algorithms' adaptability, convergence speed, and stability in such rapidly changing conditions would be highly valuable.

(2) **Integration of AI-Based Predictive Scaling:** A promising avenue is the design of a hybrid LB framework that integrates the proven efficiency of algorithms like TLB with predictive machine learning models. Such a system could forecast traffic spikes and pre-emptively adjust VM resource allocation, potentially surpassing the reactive performance of current state-of-the-art algorithms.

References

[1] S. Al E'mari, Y. Sanjalawe, A. Al Daraiseh, M. Bany Taha, and M. Aladaileh, "Cloud datacenter selection using service broker policies: A survey," *Comput. Model. Eng. Sci.*, vol. 139, no. 1, pp. 1–41, 2024, doi: 10.32604/cmes.2023.043627.

[2] D. A. Shafiq, N. Z. Jhanjhi, and A. Abdullah, "Load balancing techniques in cloud computing environment: A review," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 7, pp. 3910–3933, Jul. 2022, doi: 10.1016/j.jksuci.2021.02.007.

[3] P. Kumar and R. Kumar, "Issues and challenges of load balancing techniques in cloud computing: A survey," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–35, Art. no. 120, Feb. 2019, doi: 10.1145/3281010.

[4] S. Afzal and G. Kavitha, "Load balancing in cloud computing – A hierarchical taxonomical classification," *J. Cloud Comput.*, vol. 8, no. 22, pp. 1–24, Dec. 2019, doi: 10.1186/s13677-019-0146-7.

[5] S. M. Shetty and S. Shetty, "Analysis of load balancing in cloud data centers," *J. Ambient Intell. Humaniz. Comput.*, vol. 15, no. 2, pp. 973–981, Feb. 2024, doi: 10.1007/s12652-018-1106-7.

[6] M. U. Bokhari, Q. M. Shallal and Y. K. Tamandani, "Cloud computing service models: A comparative study," *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2016, pp. 890-895.

[7] A. Jyoti, M. Shrimali, S. Tiwari et al., "Cloud computing using load balancing and service broker policy for IT service: A taxonomy and survey," *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 11, pp. 4785–4814, Nov. 2020, doi: 10.1007/s12652-020-01747-z.

[8] S. Patel, R. Patel, H. Patel, and S. Vahora, "CloudAnalyst: A survey of load balancing policies," *Int. J. Comput. Appl.*, vol. 117, no. 21, pp. 1–5, May 2015, doi: 10.5120/20679-3525.

[9] A. M. Manasrah, T. Smadi, and A. Almomani, "A variable service broker routing policy for data center selection in cloud analyst," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 29, no. 3, pp. 365–377, Jul. 2017, doi: 10.1016/j.jksuci.2015.12.006.

[10] H. S. Mahalle, S. Tayde and P. R. Kaveri, "Implementing service broker policies in cloud computing environment," *2015 International Conference on Communication Networks (ICCN)*, Gwalior, India, 2015, pp. 186-190, doi: 10.1109/ICCN.2015.37.

[11] A. Meftah, A. E. Ahmed, and M. Zakariah, "Effect of service broker policies and load balancing algorithms on the performance of large scale Internet applications in cloud datacenters," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 5, pp. 370–378, 2018, doi: 10.14569/IJACSA.2018.090529.

[12] M. A. Shahid, M. M. Alam, and M. M. Su'ud, "Performance evaluation of load balancing algorithms with different service broker policies for cloud computing," *Appl. Sci.*, vol. 13, no. 3, p. 1586, 2023, doi: 10.3390/app13031586.

[13] P. Payaswini, "Comparative study on load balancing and service broker algorithms in cloud computing using cloud analyst tool," *Int. J. Next-Gener. Comput.*, vol. 12, no. 1, pp. 49–61, Mar. 2021.

[14] V. D. Kumar et al., "Efficient cloud resource scheduling with an optimized throttled load balancing approach," *Comput. Mater. Continua*, vol. 77, no. 2, pp. 2179–2188, 2023, doi: 10.32604/cmc.2023.034764.

[15] S. Syed Abuthahir, J. Subash Chandra Bose, and R. Saha, "An efficient enhanced dynamic load balancing weighted round robin algorithm for virtual machine in cloud computing," *J. Algebraic Statist.*, vol. 13, no. 3, pp. 3056–3065, 2022.

[16] J. Laha, S. Pattnaik, and K. Chaudhury, "Load balancing in cloud computing: A review and a novel approach," *EAI Endorsed Trans. Internet Things*, vol. 10, 2024, Art. no. e7, doi: 10.4108/eetiot.5387.

[17] Y. Lohumi, D. Gangodkar, P. Srivastava, M. Khan, A. Alahmadi, and A. Alahmadi, "Load balancing in cloud environment: A state of the art review," *IEEE Access*, vol. 11, pp. 141934–141956, 2023, doi: 10.1109/ACCESS.2023.3337146.

[18] R. Rajak, A. Choudhary, and M. Sajid, "Load balancing techniques in cloud platform: A systematic study," *Int. J. Exp. Res. Rev.*, vol. 30, pp. 15-24, 2023. doi: 10.52756/ijerr.2023.v30.002.

[19] S. S. Priya and T. Rajendran, "Enhanced weighted round robin: A new paradigm in cloud load balancing," Indian *J. Sci. Technol.*, vol. 18, no. 15, pp. 1220–1228, Apr. 2025, doi: 10.17485/IJST/v18i15.3976.

[20] D. Mekonnen, A. Megersa, R. K. Sharma, and D. P. Sharma, "Designing a component based throttled load balancing algorithm for cloud data centers," *Math. Probl. Eng.*, vol. 2022, Art. no. 4640443, 2022, doi: 10.1155/2022/4640443.

[21] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, "Load-balancing algorithms in cloud computing: A survey," *J. Netw. Comput. Appl.*, vol. 88, pp. 50–71, Jun. 2017, doi: 10.1016/j.jnca.2017.04.007.

[22] N. Chauhan, D. G. Thakur, D. A. Joshi, V. Kumar, and A. Kumar, "Review of techniques and algorithms of load balancing in cloud computing," *J. Recent Innov. Comput*. Sci. Technol., vol. 1, no. 1, pp. 15–26, 2024, doi: 10.70454/JRICST.2024.10103.

[23] A. Jain and R. Kumar, "Hybrid load balancing approach for cloud environment," *Int. J. Commun. Netw. Distrib. Syst.*, vol. 18, nos. 3–4, pp. 264–286, 2017.

[24] N. Devi, S. Dalal, K. Solanki et al., "A systematic literature review for load balancing and task scheduling techniques in cloud computing," *Artif. Intell. Rev.*, vol. 57, no. 10, p. 276, Oct. 2024, doi: 10.1007/s10462-024-10925-w.

[25] H. Le and H. Tran, "ITA: The improved throttled algorithm of load balancing on cloud computing," *Int. J. Comput. Netw. Commun.*, vol. 14, no. 1, pp. 25–39, Jan. 2022, doi: 10.5121/ijcnc.2022.14102.

[26] M. S. Almhanna, T. Murshedi, F. Al-Turaihi, R. Almuttairi, and R. Wankar, "Dynamic weight assignment with least connection approach for enhanced

load balancing in distributed systems," *Preprint*, 2023. doi: 10.21203/rs.3.rs-3216549/v1.

[27] M. Aswathi, N. N. Sharma, and A. S. Mahesh, "An enhancement of throttled load balancing algorithm in cloud using throughput," *Int. J. Eng. Res. Technol.*, vol. 5, no. 9, pp. 7603–7611, 2016.

[28] V. D. Kumar, J. Praveenchandar, M. Arif, A. Brezulianu, O. Geman, and A. Ikram, "Efficient cloud resource scheduling with an optimized throttled load balancing approach," *Comput. Mater. Continua*, vol. 77, no. 2, pp. 2179–2188, 2023, doi: 10.32604/cmc.2023.034764.

[29] K. Kishor and V. Thapar, "An efficient service broker policy for cloud computing environment," *Int. J. Comput. Sci. Trends Technol.*, vol. 2, no. 4, pp. 108–112, Jul./Aug. 2014.

[30] H. H. R. Panuganti and P. S. Rajakumar, "Enhanced throttled load balancing for virtual machine allocation in multiple data centers," *Scalable Comput.: Pract. Exper.*, vol. 25, no. 5, 2024, doi: 10.12694/scpe.v25i5.3165.

[31] M. Deep, A. Ghosh, and K. Acharjee, "Performance analysis of load balancing algorithms using Cloud Analyst," *YMER Digit.*, vol. 23, no. 5, pp. 934–945, May 2024, doi: 10.37896/YMER23.05/79.

[32] A. I. El Karadawy, A. A. Mawgoud and H. M. Rady, "An Empirical Analysis on Load Balancing and Service Broker Techniques using Cloud Analyst Simulator," *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, Aswan, Egypt, 2020, pp. 27-32, doi: 10.1109/ITCE48509.2020.9047753.

[33] P. S. Rawat, P. Dimri, G. P. Saroha, and V. Barthwal, "A load balancing analysis of cloud base application with different service broker policies," *Int. J. Comput. Appl.*, vol. 135, no. 10, pp. 11–15, Feb. 2016, doi: 10.5120/ijca2016908516.

[34] M. A. Shahid, M. M. Alam, and M. M. Su'ud, "A systematic parameter analysis of cloud simulation tools in cloud computing environments," *Appl. Sci.*, vol. 13, no. 15, p. 8785, Aug. 2023, doi: 10.3390/app13158785.

[35] S. Shanmugapriya and N. Priya, "Examination of cloud simulation platforms and implementation of load balancing in CloudAnalyst," *Indian J. Sci. Technol.*, vol. 17, no. 33, pp. 3424–3436, Aug. 2024, doi: 10.17485/IJST/v17i33.1751.

[36] O. Oladimeji, D. Oyeyiola, O. Oladimeji, and P. Oyeyiola, "A comprehensive survey on cloud computing simulators," *Sci. J. Inform.*, vol. 8, no. 1, pp. 51–59, May 2021, doi: 10.15294/sji.v8i1.28878.

[37] S. R. Jena, R. Shanmugam, K. Saini, and S. Kumar, "Cloud computing tools: Inside views and analysis," *Procedia Comput. Sci.*, vol. 173, pp. 382–391, 2020, doi: 10.1016/j.procs.2020.06.045.

[38] N. Elnagar, G. El Kabbany, A. Al Awamry, and M. B. Abdelhalim, "Simulation and performance assessment of a modified throttled load balancing algorithm in cloud computing environment," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 2, pp. 2087–2096, Apr. 2022, doi: 10.11591/ijece.v12i2.pp2087-2096.

Ali Yousefi Choubini teaches part-time in Computer Engineering at Sepahan Institute. He graduated first in his class for both his B.Sc. (Sepahan Institute) and M.Sc. (Shahid Ashrafi Esfahani University, GPA: 19.49/20). His research interests in cloud computing and load balancing which guide his plans to transition into a Ph.D. program.

Email: a.yousofi@ashrafi.ac.ir (alternative: alijosephy2019@gmail.com)

Paper Handling Data:

Corresponding author: a.yousofi@ashrafi.ac.ir

Affiliation: Department of Computer Engineering, Faculty of Engineering and Technology, Shahid Ashrafi Esfahani University, Isfahan, Iran