Journal of Computer Science

Original Research Paper

FedLoBA-1: A Load Balancing Architecture for Mitigating Resource Overloading in Federated Cloud Infrastructures

[1]Damola Gideon Akinola, [1,2]Emmanuel Adetiba, [3]Abdultaofeek Abayomi, [4]Surendra Thakur, [1]Uche Nnaji and [5]Sibusiso Moyo

[1]Department of Electrical and Information Engineering and Covenant Applied Informatics and Communications African Center of Excellence, Covenant University Ota, Nigeria
[2]HRA, Institute for Systems Science, Durban University of Technology, Durban, South Africa
[3]HRA, Walter Sisulu University, East London 5200, South Africa & IASRG Summit University, PMB 4412, Offa, Nigeria
[4]Department of Information Technology and KZN e-Skills CoLab, Durban University of Technology, Durban, South Africa
[5]Department of Mathematical Sciences, School for Data Science and Computational Thinking, Stellenbosch University, Stellenbosch, South Africa

Article history
Received: 22-09-2023
Revised: 10-01-2024
Accepted: 28-02-2024

Corresponding Author:
Emmanuel Adetiba
Department of Electrical and Information Engineering and Covenant Applied Informatics and Communications African Center of Excellence, Covenant University Ota, Nigeria
Email: emmanuel.adetiba@covenantuniversity.edu.ng

Abstract: In a subscription-based service such as cloud computing, clients have scheduled access to shared resources such as data, software, storage, and other assets as needed. Despite several benefits, cloud computing still faces significant difficulties. Load balancing, which is the capacity of the cloud infrastructure to equally distribute tasks among the resources in the cloud environment has significant issues. Cloud federation is a novel concept in cloud deployment that was developed to overcome load imbalance and other drawbacks that come with standalone clouds. However, in a federated cloud system, effective workload sharing among participating Cloud Service Providers (CSP) is also challenging. Therefore, this study presents a Federated Load Balancing Architecture version 1 (FedLoBA-1) for optimal distribution of inter-cloud and intra-cloud loads within federated cloud infrastructures. The inter-cloud load balancing was realized using Ant Colony Optimization (ACO) whereas the intra-cloud component was realized with the Throttled algorithm. The implementation of the FedLoBA-1 and simulation of the federated cloud were carried out using the CloudAnalyst simulation toolkit. Experimental results show that FedLoBA-1 gave an average response time of 92.33 ms as compared with 328.4ms and 176.55 ms for Closest Datacenter (CDC) and Optimize Response Time (ORT) algorithms respectively. The minimum average processing time obtained for FedLoBA-1, CDC, and ORT were 1.49, 17.00, and 6.68 ms respectively. FedLoBA-1 is a valuable solution for effective resource utilization in federated cloud environments. It significantly improves load balancing in cloud federation by offering an optimized two-tiered approach for intra-cloud and inter-cloud load distribution. This approach results in significantly better performance than existing algorithms.

Keywords: ACO, Balancing, CSPs, Cloud, Federated, Load, Throttled

Introduction

The advent of cloud computing technology has made computing services such as application hosting, content storage, and other services to be in high demand at a reduced cost (Meenaskhi and Chhibber, 2016; Patrick *et al*., 2022). Cloud computing is a subscription-based service in which shared assets, data, software, and other resources are made accessible as needed to clients at scheduled times. In cloud computing, there is internetworking of computing resources, and on-demand configuration and accessibility of computing resources are made flexible, easy, and fast through the Internet (Bura *et al*., 2018).

However, despite the various advantages, there are still some challenges confronting cloud technology (Mrhaoaurh *et al*., 2018).

One of the major issues in cloud computing is the inability of the cloud infrastructure to evenly distribute

tasks among resources in the cloud environment leading to load imbalance (Fatemi Moghaddam *et al*., 2015; Balne, 2019). This is a consequence of the rapid increase in the number of cloud users requesting or accessing cloud services (Chamoli *et al*., 2016; Mohammadian *et al*., 2022; Prasadhu and Mehfooza, 2020). Thus, there is often a degradation in the Quality of Service (QoS) and a compromise of the Service Level Agreement (SLA) between the CSPs and the consumers (Afzal and Kavitha, 2019). Load balancing involves the dynamic and even allocation of workloads among all the nodes that are accessible in the cloud. This even distribution of traffic to various data centers or geographic regions provides geographic redundancy and enhances performance for users located in different areas (Bogdanov *et al*., 2018). Efficient load balancing ensures that each virtual machine within the cloud system can handle an equivalent workload. As a result, load balancing becomes crucial for optimizing throughput by reducing response times (Mishra *et al*., 2020).

In order to improve the load imbalance problem, a new paradigm of cloud deployment known as cloud federation was introduced (Bhuskute and Kadu, 2021). In cloud federation (also known as federated cloud), aggregation and interconnection of various CSPs are done to satisfy market requirements. The main elements of a federated cloud include a cloud broker, cloud exchange, and cloud coordinator (Bhuskute and Kadu, 2021).

One crucial advantage of federated cloud environments is the guaranteed availability, as users experience reduced response times thanks to a pool of virtualized resources from various Cloud Service Providers (CSPs) in the federation (Levin *et al*., 2018).

However, as opined in Ray *et al*. (2018), the federated cloud environments are still confronted with issues such as:

i) Intercloud load balancing between CSPs
ii) Dynamic allocation of computing resources in the data center (DC)

Thus, this study presents a federated load-balancing architecture based on metaheuristic optimization and throttle algorithms, to mitigate overloading in federated clouds. The proposed Federated Load Balancing Architecture (FedLoBA-1) presents a load-balancing solution at both the inter-cloud (i.e., Federated) and the intra-cloud levels. The inter-cloud level is load balancing among the data centers (DCs) of a federated cloud whereas the intra-cloud level is load balancing within the DC.

Background

Load Balancing

Load balancing is the process of distributing the entire workload across nodes that are available in a cloud computing infrastructure in order to have efficient task allocation and optimum resource utilization (Narale and Butey, 2018; Prasadhu and Mehfooza, 2020). It ensures equitable and dynamic workload distribution and better resource usage in the cloud environment. As the population of cloud users increases, the workloads on the cloud (i.e., memory capacity, network, and computing resources (e.g. Central Processing Unit (CPU) and Graphical Processing Unit (GPU)) become imbalanced due to overloading or underloading (Jadav and Gayatri Pandi, 2021). The consequence of load imbalance is a degradation in the Quality of Service (QoS) and Service Level Agreement (SLA) between the CSPs and the consumers (Afzal and Kavitha, 2019). An efficient task distribution contributes to better resource management and a high level of user satisfaction. Applying load balancing minimizes delays in data transmission and prevents overloaded node conditions in cloud data centers (Goyal and Bharti, 2014; Shafiq *et al*., 2021).

In response to the challenge of load imbalance, several cloud providers offer Load-Balancing-as-a-Service (LBaaS) to customers who employ these services on an as-needed basis. Instead of distributing traffic among a group of servers within a single data center, LBaaS spreads workloads across servers and operates as a subscription or on-demand service (Ramya *et al*., 2014). Load balancing in cloud computing works exactly like a traffic controller directing traffic to avoid congestion. Figure (1) shows a generic architecture of a load balancing scheme within a stand-alone cloud environment.

Information Technology (IT) teams utilize load balancing to ensure that each node operates at maximum efficiency (Chamoli *et al*., 2016; Sajjan and Yashwantrao, 2017). As a result of the essentiality of load balancing in cloud computing, different algorithms are being used depending on the QoS demands between the CSPs and the customers (Shafiq *et al*., 2022; Thakur and Goraya, 2017). There are three broad categories of load-balancing algorithms, namely; static, dynamic, and nature-inspired/metaheuristic algorithms.

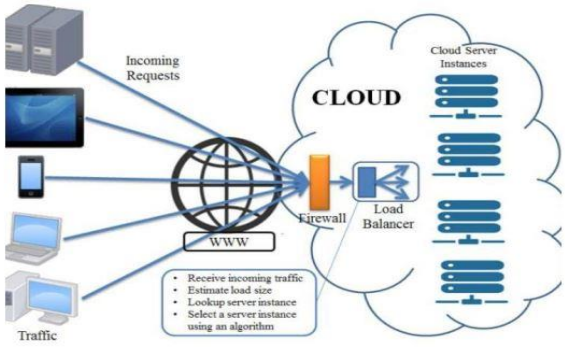

Fig. 1: Generic architecture of cloud load balancing (Singh *et al*., 2017)

Static Algorithm

In static load balancing methods, nodes are given jobs depending on their capacity to handle new requests after past knowledge of their capabilities and attributes has been taken into account (Ghutke and Shrawankar, 2014; Sajjan and Yashwantrao, 2017). These load-balancing algorithms are employed when the initial configuration, network topologies, and already established computational variables are all designated (Yadav and Prasad, 2018). As a result of a lack of consideration for the current state of the cloud infrastructure, the algorithm usually encounters a lack of fault tolerance as a major setback (Shafiq *et al.*, 2021). Instantaneous migration of tasks can also be a challenge in static algorithms (Shah and Farik, 2015). Some of the common static load balancing algorithms in cloud load balancing include round-robin, weighted round-robin, opportunistic or random, min-min, and max-min (Bura *et al.*, 2018).

Dynamic Algorithm

Dynamic load balancing algorithms are developed to proffer solutions to some of the challenges encountered in static load balancing algorithm. They search for the network's lightest server and then place the proper load on it (Kumar and Singh, 2015; Shah and Farik, 2015). The selection and distribution of tasks are based on the current state of the nodes, making them more flexible and complex (Fatima *et al.*, 2019). These algorithms are more suitable for heterogeneous environments. Examples include throttled, Equally Spaced Current Execution (ESCE), and least connection algorithms (Agarwal and Singh, 2019; Ramadhan *et al.*, 2018), etc.

Nature Inspired Algorithm

These load-balancing algorithms involve the development of optimization techniques with the aim of leveraging natural processes to solve the problem encountered during resource allocation and task scheduling in cloud computing (Thakur and Goraya, 2017). They are motivated by the behaviors of organisms such as ants, honey bees, lions, etc., or biological processes such as evolution and genetics (Shafiq *et al.*, 2021). Examples of nature-inspired algorithms that have been employed for load-balancing tasks include genetic algorithms, particle swarm optimization, honey bee foraging, ant colony optimization, etc., (Hashem *et al.*, 2017; Jyoti *et al.*, 2020).

Federated Cloud

Cloud federation, also known as federated cloud, is the merging and coordinating of different cloud computing services to meet business goals and customers' demands (Vaghela *et al.*, 2018). It is a global cloud system that combines community, private, and public clouds into High-Performance Computing (HPC) platforms. One of the fundamental goals is to meet high clients' demand by harnessing a large pool of computing resources from different CSPs. Consumers may not always be able to access high-quality services if they rely entirely on one cloud provider (Molo *et al.*, 2021). Thus, cloud federation helps CSPs render optimal services as the workload grows by renting resources from other providers. The cloud federation architecture comprises the major components hereafter described (Fig. 2):

i) Cloud broker: The federated cloud entity known as the cloud broker communicates with the cloud exchange on behalf of the client to learn about premium pricing models, SLA guidelines, resource availability, and cloud service providers. It is in charge of allocating resources in accordance with user needs (Molo *et al.*, 2021; Zangara *et al.*, 2015)
ii) Cloud exchange: The cloud exchange component serves as a mediator between the cloud coordinator and broker. It matches up the cloud coordinator's available resources with the cloud broker's requests. The cloud exchange keeps track of cloud service providers (who are actively offering their services), typical customer requests, and the current cost of facilities (Assis and Bittencourt, 2016)
iii) Cloud coordinator: The cloud coordinator readily updates all the data kept in the cloud exchange database storage. All the computing resources from different CSPs are pooled together by the cloud coordinator. The customer's budget and the Quality of Service (QoS) they require are taken into consideration when the cloud coordinator distributes the customer's access to the cloud's resources (Bhuskute and Kadu, 2021)

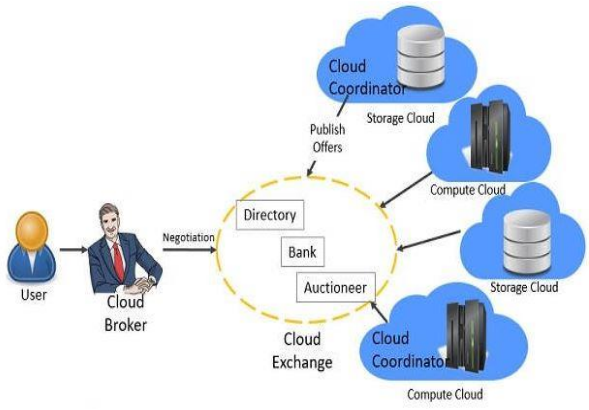

Fig. 2: Federated cloud architecture (Neha, 2020)

Both CSPs and cloud users can benefit from the cloud federation framework. Customers primarily gain from low prices and high performance, while CSPs can give their entire customer base increased cloud capabilities (Bhuskute and Kadu, 2021). However, federated cloud often encounters the problem of load imbalance due to the complexities involved in setting them up and in their operation (Molo *et al*., 2021; Zangara *et al*., 2015). In a federated cloud, more than one provider typically processes a user request. In such circumstances, dividing user requests equally between CSPs (using existing load-balancing techniques) becomes challenging for transparent workload sharing.

Related Works

The ultimate goal of load balancing in a cloud environment (both standalone and federated) is to provide seamless allocation of resources to cloud users. Researchers have developed different algorithms to handle load balancing in cloud computing infrastructure.

In (Ramadhan *et al*., 2018), an experimental simulation of the throttled algorithm was implemented on Cloud analysis. Key parameters such as response time, latency, servicing times, and cost were used in the evaluation. The authors reported that the response time was directly proportional to the number of UserBase.

Sharma and Jain (2018) proposed a load-balancing algorithm based on clients' QoS demand. Priority was based on Cost Based QoS Request (CBQR), which satisfies SLA between CSPs and clients. Three UserBases with different CBQRs were used for simulation on Cloud Analyst. The results showed an increase in response time for UserBase and datacenter in different locations while a decrease in response time occurred when the UserBase and datacenter were in the same location.

In (Jaikar *et al*., 2014), the authors proposed workload balancing in a federated cloud environment formed by different CSPs. The algorithm involves the development of two models (A and B) that study the access probability and resource utilization of federated cloud resources. The performance evaluation of the two models showed that Model A has a higher overall access probability of 2.67% than Model B.

Ray *et al*. (2018), developed the Overloaded Cloud Provider Detection Algorithm (OCPDA) to identify CSPs in a federation that are overloaded. It utilized Multiple Linear Regression (MLR) to estimate current load values for all the partner CSPs thereby detecting the overloaded ones. Experimental results showed that the algorithm successfully determined overloaded CSPs with an error between 0.9-8% for the estimated and actual overloading. However, the algorithm did not implement the balancing of loads among the CSPs.

The authors (Rajarajeswari and Aramudhan, 2016) proposed two load-balancing algorithms for improving the performance of federated cloud broker architecture. The first algorithm named Agent-based Round Robin Scheduling (ARRS) distributes service requests among the selected brokers by considering the workload and queue size of brokers. It was however reported that with ARRS, load imbalance still persists in the architecture. The second algorithm named the Decentralized Agent-based Load Balancing (DALB) algorithm operates at the broker's level and balances workload by migrating requests to underloaded brokers. It utilized stationary agents, decision-making agents, and migration agents to provide high flexibility in the request migration process. Experimental results showed that DALB achieved effective distribution of workload within a federated cloud environment.

Rajeshwari *et al*. (2021) proposed a two-fold hierarchical scheduling approach both at the federated and the cloud levels. The Queue Partitioned based Fair Load Distribution System (QPFS) was used for load distribution among the CSPs at the federation level. At the cloud level, it uses the Modified Activity Selection-based Task Scheduling by Greedy (MASG) technique to distribute tasks to the most appropriate Virtual Machines (VMs). Simulation results in CloudSim showed that QPFS-MASG achieved fairness in load distribution among multiple CSPs. Furthermore, 90% of the tasks were completed prior to their deadline with between 31 and 40% improvement in average response time.

An inter-cloud load balancer named Closest Datacenter (CDC) is used in the CloudAnalyst simulation toolkit for managing the routing of traffic between the cloud users and the data centers (Menakadevi and Devakirubai, 2016; Rani *et al*., 2015; Shahid *et al*., 2023). The closest data center (CDC) takes the delay in transmission into consideration while distributing traffic to the closest data center (Sankla, 2015). Optimize Response Time (ORT) is another scheme used in CloudAnalyst to achieve load balancing. The ORT scheme assigns a center to user requests by considering the performance of the closest data center in terms of response time. It assesses the current response time for each datacenter and then looks for the datacenter with the shortest estimated response time (Radi, 2015).

Materials and Methods

The system architecture of the proposed FedLoBA-1 is shown in Fig. (3). The architecture comprises Cloud Service Consumers, Internet, Cloudlets, Cloud Broker, Cloud Exchange (containing FedLoBA-1, the inter-cloud load balancer), and the Data Centers (DCs) within the federated cloud environment (with the intra-cloud load balancer). The Cloud Service Consumers make requests through the Internet, the Cloud Broker accommodates all the various requests of the cloud service users in the form of Cloudlets. In the Cloud Broker, the requests are

submitted in a Job Queue and each request is processed by the Cloud Information Services (CISs) to identify specifically the type of cloud service of the submitted requests. The Cloud Exchange serves as a link between the Cloud Broker and the DCs that form the federated cloud resources. The information of the interconnected data centers is registered in the Cloud Exchange. The inter-cloud or federated load balancing in this study was implemented with Ant Colony Optimization (ACO), which is a meta-heuristic algorithm. It was incorporated into the Cloud Exchange to select the best DCs with respect to cloud users' requests and to perform load balancing between the DCs. In order to evenly map cloud users' requests with the available resources within the DC resources, the intra-load balancer scheme was realized with the Throttled algorithm (Ramadhan *et al*., 2018). It was incorporated in each of the data centers as shown in Fig. (3).

Ant Colony Optimization for Inter-Cloud (Federated) Load Balancing

The inter-cloud (federated) load balancer leverages the ability of ACO to find the optimum shortest path to food source (s). In terms of the federated cloud environment, the mapping of the ACO algorithm is done such that the Ant, Food, and Pheromone represent the Load, DCs, and Communication Link respectively. The balancing of loads across the DCs via the shortest path using ACO entails three major steps. These include pheromone initialization, selection of DCs, and pheromone updating, which are hereafter described.

Pheromone Initialization

In ACO, the cloud users' requests search for nodes, which are contained in DCs, and create a communication path known as a pheromone trail. The pheromone trails activate an indirect communication behavior between requests. This communication path creates distances between the users' requests and nodes within the DCs. In implementing the ACO algorithm, choosing an optimum value as the start or initial value for the pheromone contributes either positively or negatively to the overall efficiency and convergence process of the algorithm (Bellaachia and Alathel, 2014; Tamura *et al*., 2021).

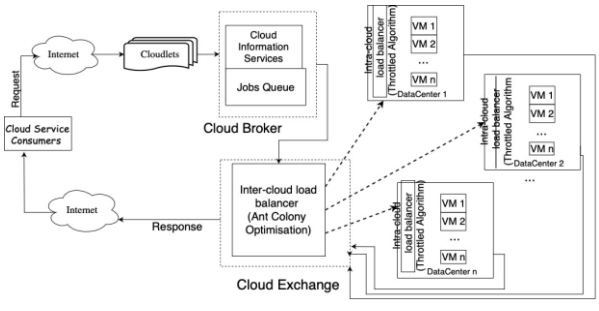

Fig. 3: FedLoBA-1 system architecture

In order to ensure the effectiveness of the initial pheromone value of the connection between DCs, it is ideal to set it as close as possible to the average expected pheromone value that a task would deposit on that edge during a single iteration (Nishant *et al*., 2012). The initial pheromone level in ACO algorithms is often initialized using either a fixed value or a value that was pre-determined by executing a fast incomplete route design technique (Nilesh and Patel, 2017). In order to initialize the pheromone in this study, the local pheromone initialization (Bellaachia and Alathel, 2014) method was adopted, with the initial pheromone value represented as Eq. (1):

$$\tau_{xy} = \frac{1}{\sum_{z \epsilon N_x^k} d_{xz}} \quad (1)$$

The term τ_{xy} represents the initial pheromone value and d_{xz} is the distance between the connecting datacenter *x* and datacenter *z* that is associated with the neighborhood N_x^k.

Selection of Datacenters

Redistributing requests or loads among the DCs is the ACO algorithm's major function. The ACO technique redistributes user requests by computing the probability of the optimal DC to respond to the requests (Nilesh and Patel, 2017). The algorithm moves through the federated cloud network, selecting DCs for their subsequent step using the probabilistic function formula in Eq. (2):

$$P_K(c,n) = \frac{[\tau(c,n)][\eta(c,n)]^b}{\sum[\tau(c,u)][\eta(c,u)]^b} \quad (2)$$

where:

P_k = Probability of the ACO choosing the nearby DC *n* for the shortest path to the current DC *c*
τ = Pheromone intensity of the edge
η = The desirability of the move by the ant; and
b = Is a factor that represents the relationship between pheromone concentration and communication path

Pheromone Table Updating

The ant (i.e., load or user request) will move by using two different types of pheromones. The type of pheromone that the ant is updating would indicate the type of movements it makes and would reveal the type of DC it is looking for. There are two different kinds of pheromones known as foraging and trailing pheromones.

Foraging pheromone: Foraging pheromone involves the forward movement of requests in searching for food sources. Foraging pheromones would be laid down once the requests found the overloaded DCs to search under loaded DCs. An ant will therefore use a foraging pheromone to try to determine the next path after approaching a DC that is not fully filled. The foraging pheromone is computed via Eq. (3) (Nilesh and Patel, 2017; Nishant *et al*., 2012):

$$\tau_{fp(t)=(1-p)}\,\tau_{fp(t)} + \sum_{k=1}^{n} \Delta\tau_{fp(t)}^{k} \quad (3)$$

Given that:

$\tau_{fp(t)}$ = Foraging pheromone before the move
$\Delta\tau_{fp(t)}$ = Change in foraging pheromone; and
p = Pheromone evaporation rate

Trailing pheromone: Trailing pheromone is used to initiate backward movement of the request upon meeting an overloaded DC, the user request finds its way back to the DC via trailing pheromone. The requests use this pheromone in this algorithm to determine their route to the underloaded DC after coming across the overloaded DC. The trailing pheromone is computed via Eq. (4) (Nilesh and Patel, 2017; Nishant *et al*., 2012):

$$\tau_{tp(t)=(1-p)}\,\tau_{tp(t)} + \sum_{k=1}^{n} \Delta\tau_{tp(t)} \quad (4)$$

Given that:
$\tau_{tp(t)}$ = Trailing pheromone before the move
$\Delta\tau_{tp(t)}$ = Change in trailing pheromone
p = Pheromone evaporation rate

Table (1) presents the summary of the parameters used for the implementation of ACO in this study. These parameters determine which decision to make in order to find the optimum path for load balancing among federated data centers.

Intra-Cloud Load Balancing

The intra-cloud load balancing within the federated cloud infrastructure involves an even balancing of loads or requests across the Virtual Machines (VMs). The FedLoBA-1 uses existing Throttled algorithms (Panchal and Parida, 2018) to create an even distribution of loads in the VMs. By establishing a table of VMs and their status, the Throttled algorithm distributes the load equally. The intra-cloud load balancer picks the first VM it finds in the list of available VMs when a request to allocate VMs from the data center is made. The request is assigned to a VM if one is discovered. The datacenter will receive a return value of -1 if no VM is discovered. It will then add this request or task to the queue and wait for the discovered VM. The benefits of this dynamic intra-cloud load balancing approach include good speed and efficient resource consumption. The list of VMs is kept up to date with their state.

Figure (4) presents the process flowchart of the ACO-based inter-cloud (federated) load balancer and the Throttled algorithm-based intra-cloud load balancer in FedLoBA-1.

Table 1: Description of ant colony optimization parameters for federated load balancing

ACO Parameters	Description
Number of Ants	The number of users 'requests.
Pheromone initialization	The initial value or strength of the communication link
Threshold	The maximum value of the pheromone level to discover an overloaded data center
Pheromone update rate (Alpha)	The rate at which pheromone values are deposited or increased.
Pheromone decay rate (Beta)	The rate at which pheromone values are reduced
Iteration	The movement of users' requests around the data centers

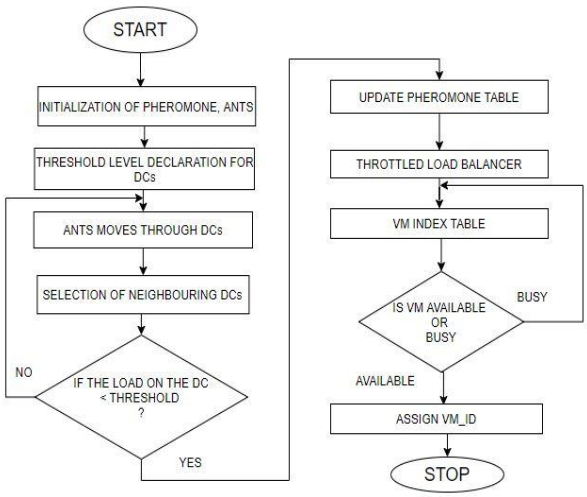

Fig. 4: FedLoBA-1 process flowchart

Implementation and Simulation of FedLoBA-1

In a typical federated cloud environment, the essential elements that must be created include the Federal Cloud Broker, Cloudlets, Cloud Exchange, Virtual Machines, and data centers (CSPs). The CloudAnalyst simulation toolkit (developed with Java) was adopted to simulate FedLoBA-1. It expands on the core functionality of the CloudSim framework to model and analyze the behavior of large-scale Internet applications in cloud settings (Jena *et al*., 2020). It contains Java classes that were used to simulate the various elements in FedLoBA-1 (Fig. 4) as well as the designed inter-cloud and intra-cloud load balancers in this study.

The configurations of the simulation parameters are in two parts; (i) The datacenter parameters and (ii) the user base parameters. The data center parameters were configured based on the specifications of the FEDGEN Testbed as shown in Table (2) (Nzanzu *et al*., 2022). The FEDGEN Testbed is a prototype federated cloud infrastructure at the Covenant Applied Informatics and Communications African Center of Excellence (CApIC-ACE) (Adetiba *et al*., 2022).

Table 2: Datacenter configuration parameters

Region ID	No. of Server	Operating system	Architecture	RAM (GB)	Storage (TB)	Processor Speed (GHz)
1	6	Ubuntu	X86	8	0.1	2
2	6	Ubuntu	X86	16	3	2.10
3	6	Ubuntu	X86	16	3	2.10

Table 3: Number of Facebook users in April 2023 (Simon, 2023)

Region	CloudAnalyst Region ID	Users (millions)	Percentage (%)
North America	0	208.6	9.87
South America	1	261.2	12.35
Europe	2	275.4	13.02
Asia	3	268.4	12.69
Africa	4	1079.1	51.02
Australia	5	22.2	1.05

The UserBase is a representation of cloud users and requests from the UserBase are processed in a particular DC based on FedLoBA-1. In order to have a real-life scenario of the cloud users, we carried out the simulation using Facebook users' reports, which emulate the percentage distribution of users across the regions. This is similar to the approach used by the authors (Wickremasinghe, 2009). Table (3) shows the summary of the total number of Facebook users with respect to different regions of the world as reported by DataReportal in April 2023 (Simon, 2023). Our simulation in CloudAnalyst used a percentage ratio of the Facebook user's report to specify the UserBase distribution. The assumption for the simulation is that most cloud users use the cloud services in the morning between the hours of 8 and 11 and also that 10% of the registered users will be online during the peak time simultaneously and only one-tenth of that number during the off-peak hours.

Results and Discussion

Inter-Cloud and Intra-Cloud Load Balancing Simulation Results

In order to evaluate the inter-cloud and intra-cloud load balancing algorithms within FedLoBA-1, we simulated a federated cloud environment in CloudAnalyst with three data centers based on the FEDGEN Testbed configurations in Table (2). The simulation at the initial stage considered 6 UserBases spread across 6 continents and 3 datacenters named DC1, DC2, and DC3 located in North America (Region 0), Europe (Region 2), and Africa (Region 4) respectively (Fig. 5). The users' requests from each UserBase are evenly distributed with the cloud resources from the three datacenters as shown in Fig. (5). Other assumptions for the simulation include 5 requests per user, 10 bytes of data size per request and that all the cloud users utilize the cloud between the hour of 8-11 in the morning. The simulation runtime was set to 10 min. The ACO-based inter-cloud load balancing component of FedLoBa-1 was implemented as a Java class to select the best data center for the UserBases, whereas the Throttled algorithm was implemented as a VM load balancer within each of the data centers.

The simulations in this study were carried out under different scenarios. These scenarios were created by varying the number of available VMs in the data centers. Table (4) gives a summary of the number of allocated VMs for each scenario with their corresponding overall response time and processing time. For instance, in Scenario 1, 2 VMs, 4 VMs, and 4 VMs were allocated for DC1, DC2 and DC3 respectively. As shown in the table, the simulation of Scenario 1 shows the overall response time and processing time of 92.33 and 42.05 ms respectively. Furthermore, in Scenario 2 the allocated VMs for the DC1, DC2, and DC3 are 5, 10, and 10 respectively with overall response time and processing time of 69.07 and 19.44 ms respectively. The results for scenarios 3-6 are shown in Table (4). Based on the results, it can be deduced that an increase in the allocated number of VMs to the data centers resulted in a progressive decrease in both the overall average response time and processing time of the system.

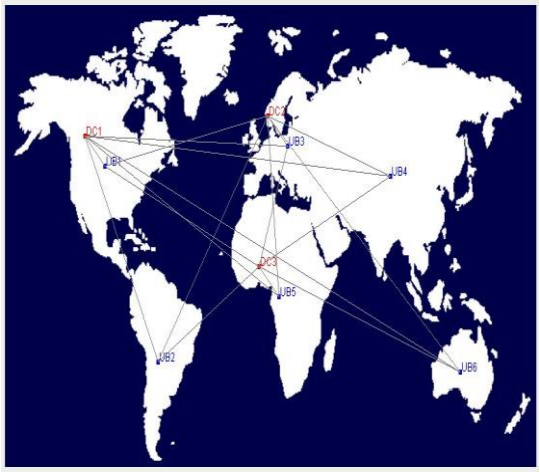

Fig. 5: Geographical distribution of data centers and user bases in a simulated federated cloud environment

Table 4: FedLoBA-1 simulation results for different configuration scenarios

Scenario ID	Scenario detail	Overall average response time (ms)	Overall average processing time (ms)
Scenario 1	2VMs in DC 1, 4VMs in each DC 2 and DC 3	92.33	42.05
Scenario 2	5VMs in DC 1, 10 VMs in each DC 2 and DC 3	69.07	19.44
Scenario 3	10VMs in DC 1, 15VMs in each DC 2 and DC 3	59.07	9.43
Scenario 4	10VMs in DC 1, 20VMs in each DC 2 and DC 3	57.25	7.62
Scenario 5	20VMs in DC 1, 40VMs in each DC 2 and DC 3	52.00	2.36
Scenario 6	25VMs in DC 1, 50VMs in each DC 2 and DC 3	51.13	1.49

Performance Benchmarking

The existing inter-cloud load balancers in CloudAnalyst include Closest Datacenter (CDC) and the Optimize Response Time (ORT) algorithms. The two inter-cloud load balancers leverage network latency to direct user requests to the data center. In this section, we report the performance benchmarking of the CDC and ORT algorithms against the proposed FedLoBA-1 in terms of response time and processing time.

Response Time

The amount of time it takes for a user or application to obtain a response after making a request is referred to as the response time in cloud computing environments. Cloud load balancer routes requests to data centers that can process them more effectively by using response time as a metric.

The CloudAnalyst CDC and ORT were evaluated against the load balancing algorithms in FedLoBA-1 by varying the number of allocated VMs from 10-125 for each of the 3 data centers. All the simulation parameters were also the same for these algorithms. Figure (6) shows the plot of the average response time against the number of VMs. As shown in the figure, FedLoBA-1 gave the lowest response time as compared with CDC and ORT. FedLoBA-1 has a maximum overall average response time of 92.13 ms while CDC has a maximum overall average response time of 328.40 ms and ORT has a maximum overall average response time of 176.55 ms.

Processing Time

The amount of time it takes for a data center to process and complete a user's request is referred to as the data center processing time. The processing time can be used as a metric by load balancers to redirect requests to data centers with shorter processing times. Simulations were carried out in this study to compare the processing times among CDC, ORT, and FedLoBA-1 using VMs from 10-125. Figure (7) shows that FedLoBA-1 produced the lowest average processing time as compared with CDC and ORT. The maximum average processing time for FedLoBA-1, CDC, and ORT are 42.05, 278.80 and 126.94 ms respectively whereas the minimum average processing time for FedLoBA-1, CDC, and ORT are 1.49, 17.00, and 6.68 ms respectively. These evidently show the superior performance of FedLoBA-1 (within the simulated federated cloud infrastructure) over the other two CloudAnalyst load-balancing algorithms.

The FedLoBA-1 achieving a faster response time (and lower processing time) across all the experimented VMs implies that it effectively distributes the workload across the DCs in the simulated federated cloud environment.

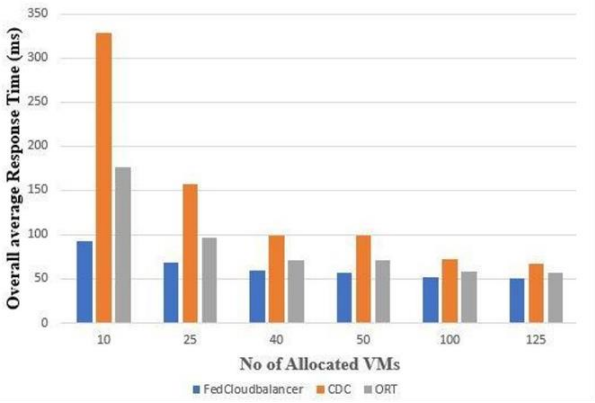

Fig. 6: Relationship between the average response time of FedLoBA-1, CDC, and ORT

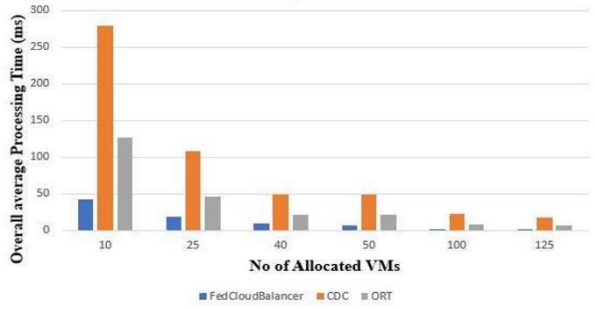

Fig. 7: Relationship between overall average processing time of FedLoBA-1, CDC, and, ORT

The results also show that FedLoBA-1 handles peak loads more efficiently thereby providing better mitigation of overloading (than CDC and ORT) during periods of high demands. The efficacy of the FedLoBA-1 is hinged on the capability of ACO to find the optimum shortest path between users' requests and underloaded data centers as well as dynamically distribute loads among the data centers. The capability of the Throttled algorithm to effectively distribute traffic among VMs within a data center is also demonstrated in the results of the response time and processing time.

Conclusion

In this study, we have presented the design and simulation of Federated Load Balancing Architecture version 1 (FedLoBA-1). We adopted the Ant Colony Optimization (ACO) algorithm to realize inter-cloud (federated) load balancing for optimal traffic distribution among federated data centers. The Throttled algorithm was further adopted to realize intra-cloud load balancing in order to evenly distribute users' requests among VMs within a data center. CloudAnalyst, which is a Java-based simulation toolkit was employed to simulate a federated cloud infrastructure and implement all the components of

FedLoBA-1. Furthermore, various simulations were carried out within CloudAnalyst for performance evaluation. The simulation results showed that both the response time and the processing time decreased as the number of allocated VMs per data center increased. This is proof that FedLoBA-1 is capable of mitigating resource overloading in federated cloud infrastructures. Moreover, benchmarking results also showed that FedLoBA-1 gave improved performance over CDC and ORT, which are the existing load-balancing algorithms in CloudAnalyst. In order to establish the performance of FedLoBA-1 in real-life scenarios, it will be deployed on live cloud testbeds such as the FEDGEN Testbed and similar platforms. In the future, we hope to improve the performance of FedLoBA-1 by exploring a hybrid of metaheuristic algorithms for inter-cloud load balancing. We also hope to develop a cloud-native web application for FedLoBA-1 so that CSPs can carry out real-time monitoring of workloads (across data centers and VMs) in a federated cloud environment.

Acknowledgment

The authors are grateful to Covenant University's Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE) for financing this research and the publication through a World Bank ACE Impact grant administered by the Nigerian National University Commission. The NEMISA eSkills Co-Lab, Durban University of Technology, Durban, South Africa is also acknowledged for supporting EA as a Visiting Professor/Research Associate during the writing of this article. Also, SM would like to thank the Department of Mathematical Sciences at Stellenbosch University, Stellenbosch, South Africa.

Funding Information

Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE), Covenant University, Nigeria received funding for this study and its publication through the World Bank ACE Impact grant administered by the Nigerian National University Commission.

Author's Contributions

Damola Gideon Akinola: Algorithm design and implementation, experimentation, and simulation, writing of manuscript.

Emmanuel Adetiba: Conception of the idea, the source for funding as the study's PI, supervision, manuscript correction, editing, and approval.

Abdultaofeek Abayomi: Contribution to manuscript writing, correction, and editing.

Surendra Thakur: Source for funding, manuscript correction, editing, and approval.

Uche Nnaji: Contribution to implementation, experimentation, and simulation, writing of the manuscript.

Sibusiso Moyo: Source for funding, manuscript correction, editing, and approval.

Ethics

The authors confirm that this article is original and has not been published in any other journal. All authors have read and approved the manuscript. No ethical review or approval process was undertaken for this study since no human or animal subjects were involved.

Reference

Adetiba, E., Akanle, M., Akande, V., Badejo, J., Nzanzu, V. P., Molo, M. J., Oguntosin, V., Oshin, O., & Adebiyi, E. (2022). FEDGEN Testbed: A Federated Genomics Private Cloud Infrastructure for Precision Medicine and Artificial Intelligence Research. *Informatics and Intelligent Applications*, 78–91. https://doi.org/10.1007/978-3-030-95630-1_6

Afzal, S., & Kavitha, G. (2019). Load balancing in cloud computing – A hierarchical taxonomical classification. *Journal of Cloud Computing*, *8*(1), 22. https://doi.org/10.1186/s13677-019-0146-7

Agarwal, S., & Singh, J. (2019). Performance analysis of load balancing algorithms for cloud computing. *International Journal of Recent Technology and Engineering*, *7*(6S4), 374–379.

Assis, M. R. M., & Bittencourt, L. F. (2016). A survey on cloud federation architectures: Identifying functional and non-functional properties. *Journal of Network and Computer Applications*, *72*, 51–71. https://doi.org/10.1016/j.jnca.2016.06.014

Balne, S. (2019). Review on challenges in SAAS model in cloud computing. *Journal for Innovative Development in Pharmaceutical and Technical Science*, *2*(3), 8–11.

Bellaachia, A., & Alathel, D. (2014). A local pheromone initialization approach for ant colony optimization algorithm. *2014 IEEE International Conference on Progress in Informatics and Computing*, 133–138. https://doi.org/10.1109/pic.2014.6972311

Bhuskute, S. S., & Kadu, S. (2021). A Study on Federated Cloud Computing Environment. *International Journal of Recent Technology and Engineering (IJRTE)*, *10*(2), 187–193. https://doi.org/10.35940/ijrte.b6311.0710221

Bogdanov, K. L., Reda, W., Maguire, G. Q., Kostić, D., & Canini, M. (2018). Fast and Accurate Load Balancing for Geo-Distributed Storage Systems. *Proceedings of the ACM Symposium on Cloud Computing*, 386–400. https://doi.org/10.1145/3267809.3267820

Bura, D., Singh, M., & Nandal, P. (2018). Analysis and Development of Load Balancing Algorithms in Cloud Computing. *International Journal of Information Technology and Web Engineering*, *13*(3), 35–53. https://doi.org/10.4018/ijitwe.2018070103

Chamoli, N., Suyal, H., Panwar, A., & Chauhan, R. (2016). Load Balancing Technique in Cloud Computing : A Review. *International Journal of Computer Applications*, *145*(15), 6–10. https://doi.org/10.5120/ijca2016910625

Fatemi Moghaddam, F., Ahmadi, M., Sarvari, S., Eslami, M., & Golkar, A. (2015). Cloud computing challenges and opportunities: A survey. *2015 1st International Conference on Telematics and Future Generation Networks (TAFGEN)*, 34–38. https://doi.org/10.1109/tafgen.2015.7289571

Fatima, S. G., Fatima, S. K., Sattar, S. A., Khan, N. A., & Adil, S. (2019). Cloud Computing and Load Balancing. *International Journal of Advanced Research in Engineering and Technology (IJARET)*, *10*(2), 189–209. https://doi.org/10.34218/ijaret.10.2.2019.019

Ghutke, B., & Shrawankar, U. (2014). Pros and cons of load balancing algorithms for cloud computing. *2014 International Conference on Information Systems and Computer Networks (ISCON)*, 123–127. https://doi.org/10.1109/iciscon.2014.6965231

Goyal, A., & Bharti, B. (2014). A Study of Load Balancing in Cloud Computing using Soft Computing Techniques. *International Journal of Computer Applications*, *92*(9), 33–39. https://doi.org/10.5120/16041-5257

Hashem, W., Nashaat, Heba, & Rizk, Rawya. (2017). Honey Bee Based Load Balancing in Cloud Computing. *KSII Transactions on Internet and Information Systems*, *11*(12), 5694–5711. https://doi.org/10.3837/tiis.2017.12.001

Jadav, P., & Pandi, G. S. (2021). Priority Based Algorithm for Load Balancing and Scalability in Distributed Environment of Cloud. *International Research Journal of Engineering and Technology*, *8*(1), 923–927.

Jaikar, A., Kim, G.-R., & Noh, S.-Y. (2014). Matrix-based Data Center Selection Algorithm for a Federated Cloud. *International Journal of Multimedia and Ubiquitous Engineering*, *9*(6), 153–158. https://doi.org/10.14257/ijmue.2014.9.6.15

Jena, S. R., Shanmugam, R., Saini, K., & Kumar, S. (2020). Cloud Computing Tools: Inside Views and Analysis. *Procedia Computer Science*, *173*, 382–391. https://doi.org/10.1016/j.procs.2020.06.045

Jyoti, A., Shrimali, M., Tiwari, S., & Singh, H. P. (2020). Cloud computing using load balancing and service broker policy for IT service: a taxonomy and survey. *Journal of Ambient Intelligence and Humanized Computing*, *11*(11), 4785–4814. https://doi.org/10.1007/s12652-020-01747-z

Kumar, S., & Singh, D. (2015). Various Dynamic Load Balancing Algorithms in Cloud Environment: A Survey. *International Journal of Computer Applications*, *129*(6), 14–19. https://doi.org/10.5120/ijca2015906927

Levin, A., Lorenz, D., Merlino, G., Panarello, A., Puliafito, A., & Tricomi, G. (2018). Hierarchical load balancing as a service for federated cloud networks. *Computer Communications*, *129*, 125–137. https://doi.org/10.1016/j.comcom.2018.07.031

Mrhaoaurh, I., Okar, C., Namir, A., & Chafiq, N. (2018). Challenges of cloud computing use: A systematic literature review. *MATEC Web of Conferences*, *200*, 00007. https://doi.org/10.1051/matecconf/201820000007

Meenaskhi, M., & Chhibber, A. (2016). An overview on cloud computing technology. *International Journal of Advances in Computing and Information Technology*, *1*(2), 219–223. https://doi.org/10.6088/ijacit.12.10028

Menakadevi, T., & Devakirubai, N. (2016). An Optimum Service Broker Policy for Selecting Data Center in Cloudanalyst. *International Journal of Research in Engineering and Technology*, *05*(09), 76–84. https://doi.org/10.15623/ijret.2016.0509011

Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: A big picture. *Journal of King Saud University - Computer and Information Sciences*, *32*(2), 149–158. https://doi.org/10.1016/j.jksuci.2018.01.003

Mohammadian, V., Navimipour, N. J., Hosseinzadeh, M., & Darwesh, A. (2022). Fault-Tolerant Load Balancing in Cloud Computing: A Systematic Literature Review. *IEEE Access*, *10*, 12714–12731. https://doi.org/10.1109/access.2021.3139730

Molo, M. J., Badejo, J. A., Adetiba, E., Nzanzu, V. P., Noma-Osaghae, E., Oguntosin, V., Baraka, M. O., Takenga, C., Suraju, S., & Adebiyi, E. F. (2021). A Review of Evolutionary Trends in Cloud Computing and Applications to the Healthcare Ecosystem. *Applied Computational Intelligence and Soft Computing*, *2021*, 1–16. https://doi.org/10.1155/2021/1843671

Narale, S., & Butey, P. (2018). Implementation of Load Balancing Algorithms in Cloud Computing Environment using Cloud Analyst Simulator. *International Journal of Recent Trends in Engineering and Research*, *4*(7), 22–27.

Neha, T. (2020). Federated Cloud. *Binary Terms*. https://binaryterms.com/federated-cloud.html

Nilesh, A. M., & Patel, C. A. (2017). Load Balancing in Cloud Computing using Ant Colony Optimization. *International Journal of Computer Engineering & Technology (IJCET)*, *8*(6), 54–59.

Nishant, K., Sharma, P., Krishna, V., Gupta, C., Singh, K. P., Nitin, & Rastogi, R. (2012). Load Balancing of Nodes in Cloud Using Ant Colony Optimization. *2012 UKSim 14th International Conference on Computer Modelling and Simulation*, 3–8. https://doi.org/10.1109/uksim.2012.11

Nzanzu, V. P., Adetiba, E., Badejo, J. A., Molo, M. J., Akanle, M. B., Mughole, K. D., Akande, V., Oshin, O., Oguntosin, V., Takenga, C., Mbaye, M., Diongue, D., & Adebiyi, E. F. (2022). FEDARGOS-V1: A Monitoring Architecture for Federated Cloud Computing Infrastructures. *IEEE Access*, *10*, 133557–133573. https://doi.org/10.1109/access.2022.3231622

Panchal, B., & Parida, S. (2018). Review Paper on Throttled Load Balancing Algorithm in Cloud Computing Environment. *International Journal of Scientific Research in Science, Engineering and Technology*, *2*(4), 201–204. https://doi.org/10.32628/IJSRSET184232

Patrick, N. V., Misra, S., Adetiba, E., & Agrawal, A. (2022). An Incremental Load Balancing Algorithm in Federated Cloud Environment. *Data, Engineering and Applications*, 395–408. https://doi.org/10.1007/978-981-19-4687-5_30

Prasadhu, M. N., & Mehfooza, M. (2020). An Efficient Hybrid Load Balancing Algorithm for Heterogeneous Data Centers in Cloud Computing. *International Journal of Advanced Trends in Computer Science and Engineering*, *9*(3), 3078–3085. https://doi.org/10.30534/ijatcse/2020/89932020

Radi, M. (2015). Efficient Service Broker Policy for Large-Scale Cloud Environments. *International Journal of Computer Science Issues*, *12*(1), 85–90. https://doi.org/10.48550/arXiv.1503.03460

Rajarajeswari, C. S., & Aramudhan, M. (2016). Agent Based Load Balancing Mechanisms in Federated Cloud. *Research Journal of Applied Sciences, Engineering and Technology*, *13*(8), 632–637. https://doi.org/10.19026/rjaset.13.3049

Rajeshwari, B. S., Dakshayini, M., & Guruprasad, H. S. (2021). Efficient Task Scheduling and Fair Load Distribution among Federated Clouds. *Journal of ICT Research and Applications*, *15*(3), 216–238. https://doi.org/10.5614/itbj.ict.res.appl.2021.15.3.2

Ramadhan, G., Purboyo, T. W., & Latuconsina, R. (2018). Experimental Model for Load Balancing in Cloud Computing Using Throttled Algorithm. *International Journal of Applied Engineering Research*, *13*(2), 1139–1143.

Ramya, R., Kriushanth, M., & Arockiam, L. (2014). A State-of-Art Load Balancing Algorithms in Cloud Computing. *International Journal of Computer Applications*, *95*(19), 10–14. https://doi.org/10.5120/16701-6834

Rani, P., Chauhan, R., & Chauhan, R. (2015). An Enhancement in Service Broker Policy for Cloud-Analyst. *International Journal of Computer Applications*, *115*(12), 5–8. https://doi.org/10.5120/20201-2450

Ray, B. K., Ghosh, O., Bhattacherjee, S., Roy, S., & Khatua, S. (2018). OCPDA: A novel approach towards detection of overloaded cloud providers in a federated environment. *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 1–6. https://doi.org/10.1109/ants.2018.8710148

Sajjan, R. S., & Yashwantrao, B. R. (2017). Load Balancing and its Algorithms in Cloud Computing: A Survey. *International Journal of Computer Sciences and Engineering*, *5*(1), 95–100.

Sankla, A. (2015). Analysis of Service Broker Policies in Cloud Analyst Framework. *Journal of Advance Research in Computer Science & Engineering (ISSN: 2456-3552)*, *2*(4), 32–37. https://doi.org/10.53555/nncse.v2i4.456

Shafiq, D. A., Jhanjhi, N. Z., & Abdullah, A. (2022). Load balancing techniques in cloud computing environment: A review. *Journal of King Saud University - Computer and Information Sciences*, *34*(7), 3910–3933. https://doi.org/10.1016/j.jksuci.2021.02.007

Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., & Alzain, M. A. (2021). A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications. *IEEE Access*, *9*, 41731–41744. https://doi.org/10.1109/access.2021.3065308

Shah, N., & Farik, M. (2015). Static Load Balancing Algorithms In Cloud Computing: Challenges & Solutions. *International Journal of Scientific & Technology Research*, *4*(10), 353–355.

Shahid, M. A., Alam, M. M., & Su'ud, M. M. (2023). Performance Evaluation of Load-Balancing Algorithms with Different Service Broker Policies for Cloud Computing. *Applied Sciences*, *13*(3), 1586. https://doi.org/10.3390/app13031586

Sharma, M., & Jain, V. K. (2018). Load balancing in cloud using prioritization based on Quality of Services (QoS) demand. *International Journal of Computer Sciences and Engineering*, *6*(11), 938–943. https://doi.org/10.26438/ijcse/v6i11.938943

Simon, K. (2023). The Latest Facebook Statistics: Everything You Need to Know DataReportal Global Digital Insights. *DataReportal*. https://datareportal.com/essential-facebook-stats

Singh, A. B., Bhat, S., Raju, R., & D'Souza, R. (2017). Survey on Various Load Balancing Techniques in Cloud Computing. *Advances in Computing*, *7*(2), 28–34. https://doi.org/10.5923/j.ac.20170702.04

Tamura, Y., Sakiyama, T., & Arizono, I. (2021). Ant Colony Optimization Using Common Social Information and Self-Memory. *Complexity*, *2021*(1), 6610670. https://doi.org/10.1155/2021/6610670

Thakur, A., & Goraya, M. S. (2017). A taxonomic survey on load balancing in cloud. *Journal of Network and Computer Applications*, *98*, 43–57. https://doi.org/10.1016/j.jnca.2017.08.020

Vaghela, A. M., Shah, N. Y., & Tulshyan, V. (2018). Cloud Federation : A Review. *Journal of Emerging Technologies and Innovative Research (JETIR)*, *5*(7), 1–6.

Wickremasinghe, B. (2009). *Cloudanalyst: A cloudsim-based tool for modelling and analysis of large scale cloud computing environments.*

Yadav, M., & Prasad, J. S. (2018). A Review on Load Balancing Algorithms in Cloud Computing Environment. *International Journal of Computer Sciences and Engineering*, *6*(8), 771–778. https://doi.org/10.26438/ijcse/v6i8.771778

Zangara, G., Terrana, D., Corso, P. P., Ughetti, M., & Montalbano, G. (2015). A Cloud Federation Architecture. *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, 493–503. https://doi.org/10.1109/3pgcic.2015.183