Weighted Service Broker Algorithm in Cloud Environment

Fatima Shannaq*[1], Areej Alshorman[2], Riziq Al- Sayyed[3], Mohammad Shehab[1], Walaa Al-Omari[1]
[1] College of Computer Science and Informatics, Amman Arab University, Amman, Jordan
[2] Faculty of Prince Al-Hussein Bin Abdullah II for IT, Al al-Bayt University, Mafraq, Jordan
[3] King Abdullah II School of Information Technology, University of Jordan, Amman, Jordan
Email: f.alshannaq@aau.edu.jo, areej2017shorman@aabu.edu.jo, r.alsayyed@ju.edu.jo, w.alomary@aau.edu.jo, moh.shehab12@gmail.com
*Corresponding author

Keywords: cloud computing, cloud service broker, cloud analyst, datacenter, VM

Received: December 30, 2023

Cloud Computing refers to the on-demand delivery of IT resources and applications via the Internet, utilizing pay-as-you-go pricing. Cloud service providers provide these services through high-configured servers housed in datacenters. Consequently, various factors are investigated, including response time, cost, the number of requests, and the selection of the optimal datacenter (DC) to fulfill customer demands. Acting as an intermediary between the customer and the service provider, the cloud service broker assumes responsibility for selecting the appropriate datacenter. Several policies exist to determine this selection, like Service Proximity-Based Routing, which prioritizes the nearest region with minimal communication delay and low network latency. However, this policy resorts to random datacenter selection when multiple datacenters exist within the closest region, neglecting factors like datacenter cost and processing time. To address this, a weighted-based approach-aware service brokering policy is proposed, grading datacenters based on factors such as virtual machine (VM) cost, data transfer cost, and VM quantity. The proposed algorithm has been tested and evaluated using Cloud Analyst simulation, yielding noteworthy reductions in total cost, processing time, and response time.

Povzetek: Predlagan je izboljšan način izbire podatkovnega centra za izvedbo oblačne storitve. Temelji na utežeh, ki ocenjujejo podatkovne centre glede na stroške prenosa podatkov in količino podatkov.

1 Introduction

As Internet technologies continue to evolve and the demands placed on computer applications grow, cloud computing has emerged as a versatile service provider that facilitates the sharing of information, software, and open resources within an Internet-based environment (Aljuhani et al., 2023).

Cloud computing presents many services such as servers, storage, and applications to the customers with pay-as-you-go pricing. The cloud service provider provides these services through datacenters with many hosted servers and switches connected with high-speed communication links (Rekha and Dakshayini (2018); Sheikhani et al. (2017); and Chen et al., (2014)).

Therefore, user satisfaction depends on the Quality of Service (QoS) provided by the service provider. Hence, the availability of datacenters and reliability of services are pivotal to ensuring superior quality of service. Unfortunately, datacenters can become congested, often due to uneven selection, load distribution, or a large increase in user numbers and requests. The consequences of overloaded datacenters are evident in the deteriorating quality of service. Thus, overloaded servers may reject new incoming requests when buffers reach saturation. Since response time estimates the duration between a user request being sent to the datacenter and the start of receiving results, a prolonged response time may indicate an overload on datacenter or cloud resources. Thus, to improve cloud performance, tasks or jobs should be distributed to the most appropriate datacenter by the service broker and virtual machines (VM) for execution with minimum response times. Minimum response time refers to the maximum number of tasks completed per unit time. Thus, the overall performance of the datacenter is improved without overloading (Manasrah et al., 2017; Aljuhani et al., 2023).

In addition, other factors affect the efficiency of service broker algorithms including cost, number of requests, and selection of datacenter. In such scenarios, an optimized cloud service broker is indispensable, acting as an intermediary between the consumer and multiple cloud service providers to select the appropriate data center that matches the user's requirements.

There are many policies to determine which datacenter (DC) should service the request for each customer, one of them is service proximity-based that select the earliest region, which has minimum communication delay and lowest network latency. In

contrast to meticulously orchestrated decisions driven by data analysis and cost-effectiveness, the selection of data centers within a given area is haphazard, devoid of any consideration for factors like virtual machine expenses, data transmission fees, the quantity of virtual machines, or computational duration. This arbitrary approach neglects optimization opportunities and undermines efficiency, potentially leading to suboptimal outcomes and unnecessary expenditures (Chen et al., 2014; and Ahmed et al., 2012)

Recently, researchers have delved into enhancing service broker algorithms, spurred by limitations in prevailing proximity-based approaches. This impetus has led to the conception of a novel algorithm tailored for data center (DC) selection within cloud environments. The algorithm, imbued with weighted mechanisms, seeks to optimize performance by minimizing costs linked to virtual machine (VM) deployment and data transmission, thus offering a fresh perspective on service provisioning dynamics. Furthermore, it takes into account the number of VMs within each DC as a contributing factor. These factors—VM cost, data transfer cost, and the number of VMs included—interact to influence the overall effectiveness of the datacenter selection process in the CloudAnalyst simulator. By considering these factors comprehensively, the proposed algorithm can make informed decisions regarding the optimal datacenter choice based on the application requirements, budget constraints, and performance expectations.

The remainder of the paper is structured as follows. Section 2 provides an overview of the Cloud Analyst simulation tool. Section 3 discusses related works, while Section 4 introduces the methodology and the proposed algorithms and provides their descriptions. The experimental setup and results are elaborated upon in Section 5. Furthermore, conclusions and avenues for future work are outlined in Section 6.

2 Introduction to cloud analyst

Cloud Analyst simulation tool built on cloudsim1 toolkit and it is a Java-based tool. It has an easy-to-use GUI that simulates the behavior of cloud computing environments and show the results in graphs and tables. It efficiently determines the best resource allocation technique and choosing datacenters to serve users' requests, and defines the cost related to these operations (Beloglazov et al., 2011 and Aazam and Huh, 2014).

The essential components in the cloud analyst tool are (Wickremasinghe et al., 2010)

- **User base (UB):** each user base relates to a group of users that is responsible for generating traffic in the simulation. In realistic, each user base represents a single user, but this cannot be reflected in the simulation because it will take a long time. This component is represented by UserBase.java class.

- **Cloudlet:** relates to the user requests. Each cloudlet represents a group of user request. It is represented by InternetCloudlet.java class.

  Datacenter (DC) and datacenter controller: DC represents the servers that includes the VMs and distributed in different geographical area. While DC controller is considered the main entity that is responsible for managing the activities of the datacenters, such as creation and destruction of virtual machines, and routing the cloudlet received from a user base to a VM through the Internet. The name of the class that represents this component is DataCenterController.java.

- **Broker policy:** that decides which DC should respond to the user base requests. CloudAppServiceBroker.java class represents this element.

- **Load balancing algorithm:** that used by the DataCenterController to decide which VM should process the next cloudlet. Cloud analyst includes three load balancing algorithms, which are: round robin, throttled, and active algorithm. VmLoadBalancer.java class represents this component.

Figure 1 shows the Cloud Analyst GUI, in which user bases and datacenters are distributed over 6 regions that correspond with the 6 main continents in the World (North America, South America, Europe, Asia, Africa, and Oceania). Locations of all other main entities such as User Bases and Datacenters in the simulation are identified only by the region for simplicity.

Figure 2 shows the class diagram which includes the classes that are responsible for routing the user request (cloudlet) from the user base to a VM in Cloud Analyst. The routing process is performed as follows (Sahu et al., 2019):

1. User base generates traffic in the form of a Cloudlet that contains the application ID and the name of the user base as the originator of the request to get back the response.
2. User request (cloudlet) is sent to the Internet class.
3. Internet checks the service broker to decide which DC to select.
4. Service broker reply to the Internet about which datacenter controller to select.

[1] Download Link: http://www.cloudbus.org/cloudsim/

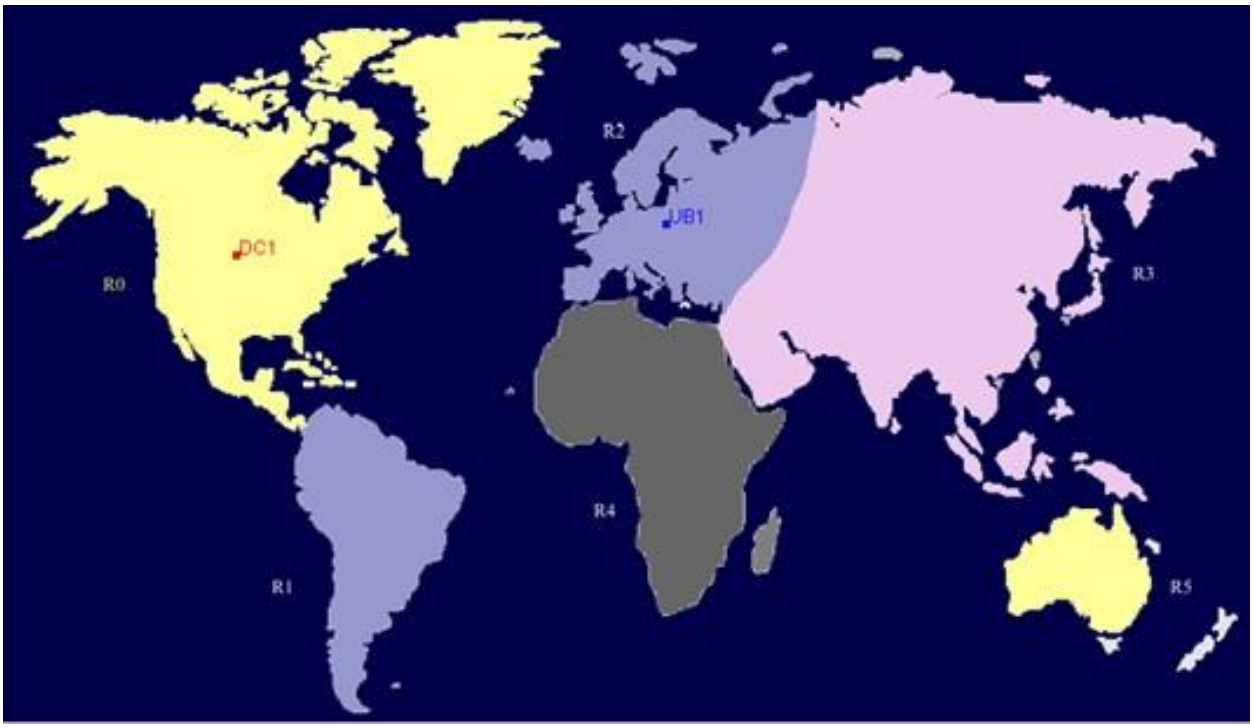

Figure 1: Cloud Analyst GUI

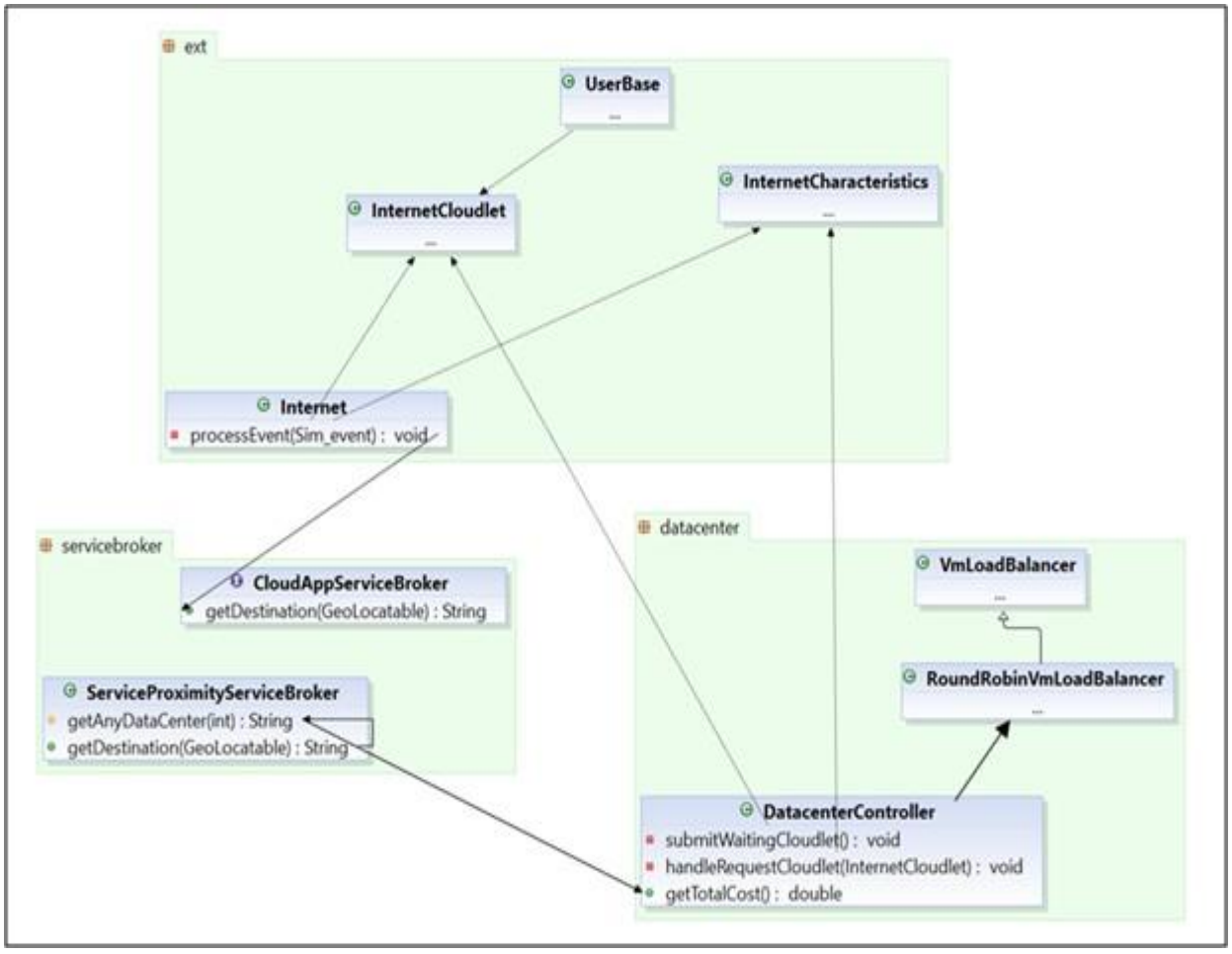

Figure 2: Class diagram of the main classes in CloudAnalyst

5. Internet adds the network delay (defined through the internet characteristics from the simulator GUI) with the cloudlet and sends to the selected DC controller.
6. DC controller uses the chosen VM load balancing policy that defined during the configuration.
7. VM load balancer starts assigning the virtual machine to the user request.
8. Selected DC sends the response to the Internet after finishing the processing of the request.
9. Internet checks the originator field in the cloudlet information then adds the response and sends it to the intended user base.

Choosing a datacenter when more than one datacenter in the same region is done by service broker. And thus, service broker controls the traffic routing between user bases and datacenters.

There are three central service broker policies currently included in the Cloud- Analyst (Sahu et al., 2019).

- **Service Proximity-Based Routing**: In this routing policy, the service broker chooses the path from user base to the nearest region, depending on the network delay and lowest network latency, this policy may suffer from bottleneck if huge requests are from the same region. However, the process of choosing a DC within a region is performed randomly.

- **Performance Optimized Routing:** In this case, the service broker actively selects the best available path based on datacenter workloads and network delay, using previous response times as a reference. However, this policy may decrease resource utilization if any datacenter experiencing high load is not considered.

- **Dynamically re-configuring router:** It is an extension to proximity-based routing. In addition, it balances application based on load by increases and decreases the number of virtual machines allocated in the datacenters.

3 Related work

The main goal of service broker policy is to select the best datacenter to serve the user request. Many factors affect choosing DC, such as processing capabilities, cost, performance, latency, and others. Cloud analyst, as mentioned earlier includes three broker algorithms, the proximity-based routing is the base algorithm that chooses the closest region according to the network latency. If that region contains more than one DC, then it chooses one randomly. However, this algorithm has many problems that affect the Quality of Service (QoS) and user's satisfaction.

Many kinds of research intended to solve this problem and improve the policy of selecting data- centers or to develop a new approach. For instance, many researchers use the Round Robin (R-R) algorithm to choose DC's in a circular manner to distribute the load between them and to avoid the problem of sending all user requests to only one DC and remains the others idle. Examples on this kind of research are the ones introduced by Kapgate (2014), Nishad et al. (2016), and Radi et al., (2015). Although these studies improved the resource utilization of the DC's, but they do not consider the characteristics of the available datacenters such as their processing capabilities and costs, which may lead to select a datacenter with a higher cost or slower response.

While Mishra et al. (2014b), Mishra and Bhukya (2014) propose a solution that based on R-R to distribute user's cloudlets equally among the available DC's in the selected region. What distinguishes these studies is that they prioritize DCs according to some rating factor. The first study uses the processing speed to order DCs, in which the DC with higher speed should be selected more times than the DCs with low processing speed. The other study prioritizes DCs based on both the processing speed and the DCs cost.

Other research dealt also with the cost factor and its effect in choosing the best DC to serve the user request. For instance, Limbani and Oza (2012b) conduct an experiment in a cloud environment using Cloud analyst simulator that includes two DCs and only one user base, in which all are located in the same region. The DC's have the same configuration except for their VM prices. Their experiment showed how the random behavior of the proximity service-based algorithm may lead to select the DC that has a higher cost. The authors suggested producing a cost-effective service broker algorithm, but they did not introduce any algorithm or implementation for their model.

Meanwhile, Aruna et al. (2017) produce an implementation of a service broker policy that selects the most cost-effective datacenter in the region. The paper also discusses some issues related to mapping of services to resources, and the studies on these issues. Although this work reduces the overall cost, but it increases the response time.

The work by Limbani and Oza (2012a) applies the min-min scheduling algorithm instead of the R-R algorithm to improve the proximity-based approach. The proposed algorithm counts the number of tasks needed to be assigned to some DC, and the number of available datacenters. Then, MxN Execute Time Matrix (ETM) will be generated that represents the execution time of M tasks when running on N types of DCs. The algorithm finds the Task-Datacenter pair with the minimum value (earliest finish time). And finally, assign the task to the DC. The results show improvements in response time and in the load balance over the available

datacenters.

Whereas Most of the previously discussed researches choose DC based on its location (low latency) or apply a random selection or based on DC's costs, the research of Kishor and Thapar (2014) takes the decision for choosing the DC based on their hardware configuration (number of hardware machines), because DC with a greater number of virtual machines can handle a larger number of cloudlets.

Cloud technologies improve rapidly and require a need for simulation tools like Cloud Analyst that offer the basic information and configuration about cloud resources. Cloud analyst includes many algorithms that determine the best resource allocation technique and choosing datacenters to serve users' requests. However, Mahalle et al. (2015) analyze these algorithms and compare them through different scenarios and experiments. The survey introduced by Patel et al., (2015) investigates the functionalities of two cloud simulators, which are Cloudsim and Cloud Analyst. Then it reviews the service broker policies of the Cloud Analyst tool, tests them under different scenarios, and shows their related challenges and issues. In addition, it also presents the different solutions and papers in this field.

In (Aljuhani et al., 2023), the authors proposed a new strategy for optimizing service provider selection, which integrates consumer and provider preferences. Their approach initiates with the utilization of the best-worst method (BWM) to prioritize tasks based on the preferences articulated by both stakeholders. Subsequently, the model evaluates and contrasts task similarities between consumers and providers through the application of two novel algorithms. The computational complexity of these algorithms has been assessed to be $O(n^3)$, signifying their efficiency and scalability.

Alwada'n et al. (2023) introduced a new approach to congestion management within cloud systems. The Dynamic Congestion Management (DCM) system they proposed was specifically tailored to efficiently handle a large influx of cloud requests while simultaneously ensuring compliance with client quality requirements as stipulated in the service-level policy. One notable aspect of their system was the introduction of a forwarding policy designed to prioritize high-priority calls originating from cloud service requesters. These calls were routed through the broker to appropriate cloud resources, with the policy drawing inspiration from congestion management mechanisms utilized by Cisco.

In (Chauhan et al., 2021), the authors introduced the Brokering Service Selection (BSS) model for cloud service selection. Unlike previous approaches, the BSS model integrates subjective and objective weighting methods to evaluate Quality of Service (QoS) attributes. Subjective weights are obtained from user feedback, while objective weights are derived from benchmark-tested data of cloud services. By combining these weights, the model calculates an integrated total weight for each service. The BSS methodology is then utilized to rank the cloud services accordingly. The effectiveness of the BSS model was assessed through simulation and a real dataset case study. The results demonstrated the model's consistency in addressing rank reversal issues, a common challenge in service selection. Additionally, the BSS model exhibited superior execution time compared to other contemporary solutions, indicating its efficiency in practical applications.

Mehraj and Banday (2022) proposed a weighted averaging methodology specifically designed for Cloud computing. This new technique employs WMA-OWA functions to dynamically assign weights to different factors. Thus, it overcomes the constraints associated with traditional subjective weight assignment techniques. Unlike methods reliant on manual or subjective determinations by experts, such as random allocation, expert opinion, or average weight, the proposed approach offers superior flexibility, adaptability, and dynamic adjustment capabilities within Cloud computing environments. Empirical evidence from experiments validates the efficacy of this method, highlighting its potential to enhance performance in Cloud computing contexts.

Furthermore, various studies in the literature have adopted nature-inspired optimization algorithms to develop efficient load-balancing strategies for cloud environments. For example, Kaur et al. (2020) introduced a new framework named Deep Learning Deadline-constrained, Dynamic VM Provisioning and Load Balancing (DLD-PLB) to explore Deep Learning (DL) methodologies in cloud computing. This framework leveraged deep learning techniques to generate an optimal VM schedule using Genome workflow tasks as input. A comparative analysis of makespan and cost was performed against a previous load-balancing optimization framework, a hybrid approach called HDD-PLB. The earlier methods integrated a hybrid Predict-Earliest-Finish Time (PEFT) with Ant Colony Optimization (ACO) to optimize underutilized VMs, alongside a hybrid PEFT-Bat approach aimed at enhancing overflow VM utilization.

Abed-Alguni and Alawad (2021) introduced a discrete variant of the Grey Wolf Optimizer (DGWO) for scheduling interdependent tasks onto Virtual Machines (VMs). The scheduling task within DGWO is formulated as a dual-objective minimization challenge involving computation and data transmission expenses. To handle discrete candidate solutions, DGWO adopts the largest order value (LOV) technique, transforming continuous solutions into discrete ones. Empirical evaluations against established optimization-based scheduling algorithms (Particle Swarm Optimization (PSO), Grey Wolf Optimizer) demonstrate that DGWO achieves faster task distribution to VMs. Additionally, DGWO is contrasted with PSO and Binary PSO (BPSO) using Workflow-Sim and variously sized scientific workflows.

The simulation results indicate that DGWO outperforms the other algorithms in terms of makespan.

Table 1 shows a comparison between the different service broker models introduced in the literature and shows the improvement in each study, the factors adopted in brokering, and the simulation tool used to display the model. It shows that Cloud Analyst is used almost in all studies, which proves its eligibility in imitating the process of resource provisioning and allocation in cloud environments.

Our proposed approach distinguishes itself from prior research by meticulously examining numerous essential elements to enhance the efficiency of the service broker algorithm in data center selection. While earlier investigations tended to concentrate on singular aspects, our methodology encompasses a broader spectrum, amalgamating factors such as cost reduction, processing time, and response time concurrently. By acknowledging the intricate interactions among these variables, our algorithm not only enhances each facet independently but also guarantees equitable enhancements across the entire system. This holistic optimization strategy differentiates our work, showcasing a profound comprehension of the intricacies inherent in data center selection and advocating for more streamlined and responsive service delivery in cloud computing ecosystems.

Compared with nature-inspired optimization algorithms such as Genetic Algorithms (GA), the Weighted Service Broker algorithm operates on the foundation of deterministic optimization principles. Rather than relying on random variations and genetic operators for decision-making, this algorithm employs predefined weights and criteria to allocate resources efficiently. The deterministic approach fosters predictability and control, offering advantages in scenarios prioritizing stability and reliability. Moreover, the algorithm's streamlined computational complexity minimizes overhead compared to the resource-intensive exploration typical of nature-inspired optimizers. Its deterministic nature simplifies implementation and diminishes the necessity for intricate parameter tuning, facilitating faster convergence and reduced computational costs in applicable contexts.

4 Methodology

Cloud Analyst is a simulation tool designed to assist in the selection of an appropriate datacenter in a cloud computing environment. In this context, factors such as virtual machine (VM) cost, data transfer cost, and the number of VMs included are crucial in evaluating the effectiveness of different datacenter options.

In Cloud Analyst simulation tool, datacenter's cost is represented by virtual machine cost included in the DC, and the data transfer cost.

VM cost refers to the pricing associated with deploying and running virtual machines in a cloud datacenter. This cost can vary based on factors, like the type of VM instance (e.g., CPU, RAM, storage capacity), the duration of usage, and any additional services or features included. In the Cloud Analyst simulator, the VM cost factor helps evaluate the economic feasibility of deploying applications on different VM instances offered by various datacenters, in which lower VM costs can lead to reduced overall expenses for running applications, making a datacenter more attractive for selection.

Data transfer cost refers to the charges incurred for transferring data into and out of the cloud datacenter. This cost is influenced by factors like the volume of data transferred, the distance between the user and the datacenter, and the network bandwidth. In the context of the Cloud Analyst simulator, data transfer cost is essential in assessing the expenses associated with moving data between the cloud datacenter and the user, in which higher data transfer costs can significantly impact the overall operational expenses, especially for applications that require frequent data exchanges.

However, the number of VMs included refers to the capacity or scalability of the datacenter, represented by the maximum number of virtual machines that can be provisioned and managed within the infrastructure. This factor influences the ability of the datacenter to accommodate varying workloads and demands from applications. In the Cloud Analyst simulator, the number of VMs included plays a crucial role in determining the scalability and flexibility of each datacenter option, in which datacenters with a higher number of VMs included can handle more concurrent tasks and accommodate growing resource requirements, enhancing their suitability for diverse application workloads.

The algorithms proposed in this study have considered these factors with the objective of refining the proximity-based service broker policy. This policy utilizes a random selection of a datacenter to fulfill user requests when multiple datacenters are present within the same region. Consequently, the proposed algorithms aim to mitigate this random selection by incorporating these factors into the decision-making process.

Table 1: Comparison of service brokers models

Paper	Improvement	Factors	Simulation tool
Kapgate (2014)	Resource utilization	Response time observed by user & DC Request Service Times	Cloud Analyst
Nishad et al. (2016)	Resource utilization	total cost is same for all data centers by using round-robin selection of DC	CloudSim & Cloud Analyst
Radi (2015)	Resource utilization	Infrastructure of datacenters	Cloud Analyst simulation toolkit
Mishra et al. (2014b)	Resource utilization Processing time Response time	Processing speed	Cloud Analyst
Mishra and Bhukya (2014)	Resource utilization Processing time Response time	Rating factor that depends on (cost + speed MIPS)	Cloud-Analyst
Limbani and Oza (2012a)	Cost	VM cost + Data transfer cost	Cloud Analyst
Limbani and Oza (2012b)	Processing time	VM cost	Cloud-Analyst
Sheikhani et al. (2017)	Response time + average processing time	Speed MIPS Bandwidth	Cloud Analyst
Kishor and Thapar (2014)	Resource utilization overall response time (the regional request servicing time& the data centre processing time)	Hardware configuration	Cloud Analyst
Aljuhani et al. (2023)	Improve task scheduling Resource utilization Enhance consumers' trust	VM cost Speed MIPS	Cloud Analyst
Alwada'n et al. (2023)	Resource utilization Improve the functionality Enhance the performance	Improve the quality for the clients' requirements	CloudSim & Cloud Analyst
Chauhan et al. (2021)	Resource utilization Achieve the integrated total weight of QoS parameters	Execution time metric	Cloud Analyst, CloudSim & toolkit
Mehraj and Banday (2022)	Resource utilization Assign weights dynamically	flexibility, adaptability, and dynamic adjustment capability	Cloud Analyst & CloudSim
Kaur et al. (2020)	Resource provisioning, Load balancing	VM cost Number of VMs	Cloud workflow simulator
Abed-Alguni and Alawad (2021)	Scheduling dependent tasks to VMs	Computation and data transmission costs	WorkflowSim
Proposed model	Resource utilization Cost DC processing time DC response time	VM cost Data Transfer cost Number of VMs	Cloud Analyst

```
//Get index of all the datacenters in regions
01: regionalist =  regionalDataCenterIndex.get (region)  // store regionlist of selected datacentre
02: if regionaList is not NULL then
03:       listSize = size (regionalist)
04:     if  listSize is 1 then
05:       dcName = regionalist.get(0)
06:     else
07:       allInternetEntities InternetCharacteristics.getInstance().getAllEntities()
08:       leastVMCost = 1000000
09:       for all entities in allInternetEntities
10:                 if entity is DatacenterController and entity.getRegion()==region
11:                           if entity . getVmCost () < leastVMCost
12:                                     leastVMCost = entity . getVmCost ()
13:                                     dcName = entity.get_name()
14:       return dcName
```

Figure 3: Algorithm 1: VmCost

```
//Get index of all the datacenters in regions
01: regionalist =  regionalDataCenterIndex.get (region)  // store regionlist of selected datacentre
02: if regionaList is not NULL then
03:       listSize = size (regionalist)
04:     if  listSize is 1 then
05:       dcName = regionalist.get(0)
06:     else
07:       allInternetEntities InternetCharacteristics.getInstance().getAllEntities()
08:       leastTotalCost = 1000000
09:       for all entities in allInternetEntities
10:                 if entity is DatacenterController and entity.getRegion()==region
11:                           if entity . getTotalCost () < leastTotalCost
12:                                     leastTotalCost = entity . getTotalCost ()
13:                                     dcName = entity.get_name()
14:       return dcName
```

Figure 4:Algorithm 2: TotalCost

Algorithm 1 (Figure 3) prioritizes the DCs based on their virtual machine's cost (vmCost) only. The datacenter with the least cost should be selected first. However, Algorithm 2 (Figure 4) orders datacenters based on the total cost (vmCost and transferCost).

Algorithm 1 and algorithm 2 consider only the cost factor and neglected the physical characteristics of the DCs and their VMs, such as the processor speed, number of VM on each DC, and many others features.

Therefore, we have proposed Algorithm 3 (See Figure 5) that gives weights to datacenters according to their total cost and the number of VM on each datacenter, as we have considered that the DC that has a lower price and a larger number of VM should take a higher weight. The following equation demonstrates how weight (W_i) is calculated for DC*i*:

$$\mathbf{Wi} = \frac{\mathbf{R \times Max(C1, C2, \ldots, Cn)}}{\mathbf{Ci}} + \frac{\mathbf{((1-R) \times VMi)}}{\mathbf{Min(VM1, VM2, \ldots VMn)}}$$

R represents the rating factor and its value range between 0.0 and 1.0, and different values of *R* give different behavior of the algorithm. While, C_i and VM_i relates to the total cost and the number of VM created in DC*i*. However, $Max(C_1, .. C_n)$ returns the highest price among DCs, and $min(VM_1, \ldots VM_n)$ returns the lowest number of VMs among different DCs.

5 Experimental setup and results

In order to implement and evaluate the proposed algorithms in a cloud environment, we have used the Cloud Analyst tool. We have conducted three experiments that include 5 datacenters (DCs) and 1 user base. Then we displayed how the algorithm distributes

```
//Get index of all the datacenters in regions
01: regionalist =  regionalDataCenterIndex.get (region)  // store regionlist of selected datacentre
02: if regionaList is not NULL then
03:       listSize = size (regionalist)
04:     if  listSize is 1 then
05:       dcName = regionalist.get(0)
06:     else
07:       allInternetEntities InternetCharacteristics.getInstance().getAllEntities()
08:       leastNumberOfVM = 1000000;
09:       maxDCCost = 0;
10:       weightDC = Double.MIN_VALUE;
11:       for all entities in allInternetEntities
12:                 if entity is DatacenterController and entity.getRegion()==region
13:                           if entity . getVmStatesList().size() < leastNumberOfVM
14:                                     leastNumberOfVM = entity . getVmStatesList().size()
15:                           If entity. .getTotalCost() > maxDCCost
16:                                     maxDCCost =      entity. .getTotalCost()
17:       for all entities in allInternetEntities
18:                 if entity is DatacenterController and entity.getRegion()==region
19:                           R =0.7;
20:                           Tempest = (R* maxDCCost/entity. getTotalCost ()) +
21:                                     entity.getVmStatesList().size()/((1-R)*leastNumberOfVM));
22:                           if tempCost > weightDC
23:                                     weightDC = tempCost
24:                                     weightDC = tempCost
25:       return dcName
```

Figure 2: Algorithm 3: TotalCost and No. of VM

the user requests among these DCs. Then, we compared the obtained results with the serviceProximityService algorithm in the cloud analyst simulator.

The metrics that will be used to evaluate the performance of the proposed algorithms and compare them with the other algorithms encapsulated in the cloud analyst tool are:

- **Response time:** It is the difference between the time for sending a request and the time for receiving its response. The aim is to minimize the response time so we can improve the system's performance.
- **Processing time:** which is the time needed to process a task.
- **The total cost of operation** (Virtual machine cost + data transfer cost)

5.1. Experiment 1: Select datacenter with the least cost without considering DCs physical characteristics

5.1.1. Simulation configuration

Table 2 shows the simulation configuration for the first two experiments, in which 5 DCs and 1 UB are located in the same region, then test how the proposed algorithm avoids the random selection of datacenter and chooses the one with the minimum cost.

As shown in the above table, we assume that users use the system in the evenings after they finished their work for approximately two hours. Average peak users point to the number of registered users who are jointly online through the peak time. While the average off-peak users determine the number of simultaneous users during the off-peak hours, which we assume 10% of the number of average peak users. And also suppose that each online user makes 60 requests per hour. According to DC's configuration, we have assumed that each datacenter has 5 virtual machines (VMs), running Linux OS, and has Xen VMM architecture. Also, the cost per VM per hour and the cost per 1Gb of data

transfer from and to the Internet for DC1 to DC5 are $0.04, $0.07, $0.1, $0.15, $0.2 respectively. We have considered the datacenters have the same physical characteristics.

Table 2: Experimental Parameters

Parameter	Value
Simulation duration	5 hours
User base Region	2
Request Per User Per Hour	60
Data Size Per Request	100
Peak hour start (GMT)	13
Peak hour end (GMT)	15
Avg Peak Users	400000
Avg Off Peak Users	40000
Number of DC	5
DCs Region	0
DCs – Cost per VM $/H	0.04, 0.07, 0.1, 0.15, 0.2
DCs – Data Transfer Cost $/GB	0.04, 0.07, 0.1, 0.15, 0.2
Number of VMs	5
User Grouping Factor	10000
Request Grouping Factor	1000
Executable Instruction Length	250
Load Balancing Policy	Round Robin

5.1.2. Results discussion

Table 3 displays the results obtained from the first experiment and compares Algorithm 1 and 2 with the broker policies implemented in Cloud Analyst. Our model shows a significant reduction in total cost. While the processing time and the response time were better in Cloud Analyst policies because it adopts the best effort approach for selecting DC which depends on randomization.

Figure 6 analyses the performance of the first experiment. It shows that proximity-based policy has the least response time, which was 399.17. While Algorithm 1 was the best in terms of cost.

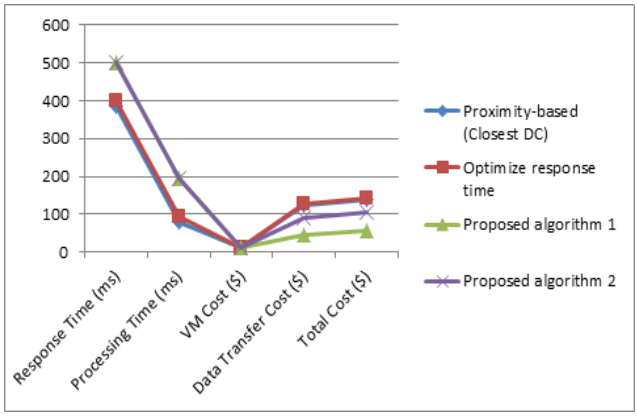

Figure 5: Performance Analysis – Experiment 1

Table 2: Experiment 1 results

Server broker policy	Response Time (ms)	Processing Time (ms)	VM Cost ($)	Data Transfer Cost ($)	Total Cost ($)
Proximity-based (Closest DC)	383.70	78.58	14.00	125.19	139.19
Optimize response time	399.17	95.38	14.00	127.09	141.09
Proposed algorithm 1	**500.79**	**195.78**	**11.40**	**45.27**	**56.67**
Proposed algorithm 2	**500.79**	**195.81**	**14.00**	**89.62**	**103.62**

5.2. Experiment 2: Select datacenter with the least cost when considering DCs physical characteristics

5.2.1. Simulation configuration

This experiment follows the same configuration as in the previous experiment, but the number of VMs on each DC changed according to Table 4 in order to test and evaluate the third proposed Algorithm.

Table 3: No. of VM on each DC

Datacenter name	Number of VMs
DC1	10
DC2	5
DC3	8
DC4	8
DC5	5

5.2.2. Results discussion

Table 5 shows the results of the weighted algorithms when considering different rating values (R). The table reveals that the algorithm gives approximately the same result for Average response time, Average processing time, and Data transfer cost for any value of *R*. However, it shows a significant reduction in VM cost at rating factors 0.0, 0.1, 0.9, and 1.0. Table 6 displays the comparison between the results obtained from the proposed algorithm (R=0.1) and the proximity-based algorithm, which apparently shows the improvement in the processing time and the total cost which was $146.13 and became approximately $24. However, the response time in our algorithm increased because of the extra computational time needed to calculate the weight for each DC. Figure 7 shows the performance analysis of this experiment.

Table 4: Weighted algorithm results with different rating factor

Rating factor	Average response time (ms)	Average processing time (ms)	VM cost ($)	Data transfer cost ($)	Total cost ($)
0.0	654.20	42.79	16.15	7.54	23.69
0.1	654.20	42.79	16.15	7.54	23.69
0.2	654.10	42.65	16.50	7.57	24.07
0.3	654.05	42.68	17.00	7.59	24.59
0.4	654.19	42.85	17.00	7.63	24.63
0.5	655.79	42.92	17.00	7.72	24.72
0.6	654.19	42.85	17.00	7.63	24.63
0.7	654.05	42.68	17.00	7.59	24.59
0.8	654.10	42.65	16.50	7.57	24.07
0.9	654.20	42.79	16.15	7.54	23.69
1.0	654.20	42.79	16.15	7.54	23.69

Table 5: Comparison between weighted algorithm and proximity based in Experiment 2

Service Broker Algorithm	Avg response time (ms)	Avg processing time (ms)	VM cost ($)	Data transfer cost ($)	Total cost ($)
Proximity-based (Closest DC)	379.38	74.25	18.75	127.38	146.13
Weighted Algorithm (R=0.1)	**654.20**	**42.79**	**16.15**	**7.54**	**23.69**

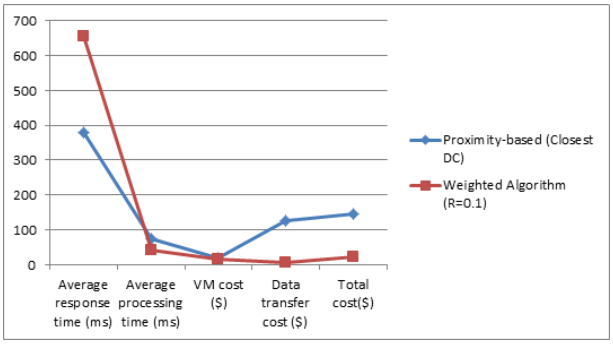

Figure 6: Performance Analysis - Experiment 2

5.3. Experiment 3: Select datacenter with the highest weight - DCs located in different regions

5.3.1. Simulation Configuration

This experiment follows the same configuration as in experiment 2, but the datacenters are located in different regions as shown in Table 7.

Table 6: Location of DCs

Datacenter name	Region
DC1	0
DC2	1
DC3	1
DC4	2
DC5	2

5.3.2. Results discussion

Table 8 displays the results of both the proximity-based policy and the weighted algorithm when the DCs are located in different regions. As shown, the performance of our algorithm exceeds the other algorithm since it shows a reduction in response time, processing time as well as the total cost.

Table 7: Comparison between weighted algorithm and proximity-based in Experiment 3

Service Broker Algorithm	Average response time (ms)	Average processing time (ms)	VM cost ($)	Data transfer cost ($)	Total cost ($)
Proximity-based (Closest DC)	262.27	210.83	17.70	194.27	211.97
Weighted Algorithm (R=0.1)	**252.96**	**38.44**	**17.70**	**26.57**	**44.27**

The explanation behind these results is that in Experiment 2 the algorithm calculates the weight for 5 DCs, while in Experiment 3 it calculates the weight for only 2 DCs so it took less time and gave a faster response. Figure 8 shows the performance of the two algorithms.

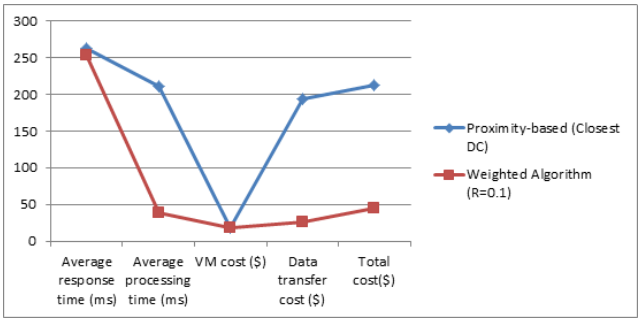

Figure 7: Performance Analysis– Experiment 3

5.4. Experiment 4: Select datacenter with the highest weight — DCs and UBs located in different regions

In this experiment, we aimed to test our algorithm over a different scenario, in which there are 6 user bases worldwide.

5.4.1. Simulation configuration

We have determined 6 user bases, each one in a different region as shown in Table 9, where each region represents a different continent, also we assume that all user bases located in the same time zone for simplicity reasons. We have also assumed that the number of off-peak users equal to 10% of the number of average peak users, and each online user makes 15 requests per hour.

Table 10 shows the configuration of datacenters. It shows 5 DCs in different regions. The cost for each DC is the same as the previous experiment.

Table 8: User bases configuration

User bases	Region	Continent	User request/hour	Peak hours start (GMT)	Peak hours end (GMT)	Avg. peak users	Avg. off-peak users
UB1	0	North America	15	13	15	400000	40000
UB2	1	South America	15	15	17	100000	10000
UB3	2	Europe	15	20	22	300000	30000
UB4	3	Asia	15	1	3	150000	15000
UB5	4	Africa	15	21	23	50000	5000
UB6	5	Oceania	15	9	11	80000	8000

Table 9: DCs configuration

Datacenter name	Region	Cost per VM $/HR	Data transfer cost $/GB	Number of VMs
DC1	0	0.04	0.04	10
DC2	1	0.07	0.07	5
DC3	4	0.1	0.1	8
DC4	5	0.15	0.15	8
DC5	2	0.2	0.2	5

5.4.2. Results discussion

Experiment 4 simulates the cloud environment when it encounters many users spread around the world. The results in Table 11 shows a significant improvement in the cloud services, since the response time decreased almost to the quarter of the random policy. Also, the processing time of the proximity-based policy is 10 times larger than in weighted policy.

Figure 9 shows the significant reduction in both response time and processing time in the weighted service broker algorithm. VM's cost was the same in both algorithms, but the data transfer cost was reduced from $168.39 to $18.87.

Table 10: Comparison between weighted algorithm and proximity-based in Experiment 4

Service Broker Algorithm	Average response time (ms)	Average processing time (ms)	VM cost ($)	Data transfer cost ($)	Total cost ($)
Proximity-based (Closest DC)	1924.48	1763.61	12.21	168.39	180.60
Weighted Algorithm (R=0.1)	**466.20**	**169.42**	**12.21**	**18.87**	**31.08**

6 Conclusion and future work

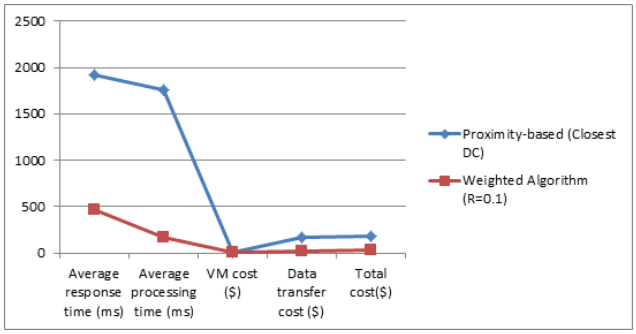

Figure 9: Performance Analysis – Experimet 4

Cloud Computing refers to the on-demand delivery of IT resources and applications via the Internet, utilizing pay-as-you-go pricing. Cloud service providers provide these services through high-configured servers housed in datacenters by adopting service broker algorithm like proximity-based routing that chooses the shortest path from the user base (UB) to the data center (DC) based on the network latency only. However, this algorithm suffers from various drawbacks due to the random selection of DC when there are multiple datacenters in the same region. In this study, we proposed a weighted approach algorithm to enhance proximity-based routing algorithm by prioritizing DCs based on their cost and number of their VMs. The algorithm tested over Cloud Analyst simulator and the results show improvements in the processing time and the total cost in all scenarios. And it also shows a reduction in response time when the number of DCs located in the same region is small since it calculates the weight for only a small number of DCs.

However, cloud computing environments often experience variations in workload demand, with periods of peak usage followed by periods of quiescence. Ensuring optimal resource allocation and user satisfaction in dynamic workloads necessitates the adaptation of service broker algorithms in real-time. The scalability of these algorithms becomes crucial to meet changing requirements efficiently; however, implementing vendor-specific algorithms in multi-cloud or hybrid environments can present significant challenges due to variations in APIs, pricing models, and performance characteristics across platforms. Addressing these issues requires an enhanced approach, which can be considered in future work, such as incorporating additional datacenter characteristics like processor speed and hardware units, alongside leveraging machine learning and artificial intelligence techniques. Furthermore, implementing standardized interfaces and protocols can also be considered in future research to foster interoperability and portability, facilitating improved decision-making and resource allocation across diverse cloud environments.

List of abbreviations

The key abbreviations used in this paper are:
VM: virtual machine
DC: datacenter
UB: User base
QoS: Quality of Service

References

[1] Aazam, M., & Huh, E.-N. (2014). Fog Computing and Smart Gateway Based Communication for Cloud of Things. *2014 International Conference on Future Internet of Things and Cloud*. https://doi.org/10.1109/ficloud.2014.83.

[2] Abed-alguni, B. H., & Alawad, N. A. (2021). Distributed Grey Wolf Optimizer for scheduling of workflow applications in *cloud environments. Applied Soft Computing*, 102, 107113. https://doi.org/10.1016/j.asoc.2021.107113.

[3] Ahmed, A.S., 2012. Proximity-Based Routing Policy for Service Brokering in Cloud Computing. *International Journal of Engineering Research and Applications*. 2(2) 453, 12, DOI={10.5220/0003908000760081}.

[4] Aljuhani, A., Alhubaishy, A., Khalid Imam Rahmani, M., & A. Alzahrani, A. (2023). Light-Weighted Decision Support Framework for Selecting Cloud Service providers. *Computers, Materials & Continua*, 74(2), 4293–4317. https://doi.org/10.32604/cmc.2023.033893.

[5] Alwada'n, T., Al-Tamimi, A.-K., Mohammad, A. H., Salem, M., & Muhammad, Y. (2023). Dynamic congestion management system for cloud service broker. *International Journal of Electrical and Computer Engineering (IJECE)*, 13(1), 872. https://doi.org/10.11591/ijece.v13i1.pp872-883.

[6] Aruna, M., Bhanu, D., & Karthik, S. (2017). Allocating resources in cloud using CloudAnalyst. 2017 *International Conference on Intelligent Computing and Control (I2C2)*. https://doi.org/10.1109/i2c2.2017.8321912.

[7] Beloglazov, A., & Buyya, R. (2011). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in *Cloud data centers. Concurrency and Computation: Practice and Experience*, 24(13), 1397–1420. Portico. https://doi.org/10.1002/cpe.1867

[8] Chauhan, S. S., Pilli, E. S., & Joshi, R. C. (2021). BSS: a brokering model for service selection using integrated weighting approach in *cloud environment. Journal of Cloud Computing*, 10(1). https://doi.org/10.1186/s13677-021-00239-5.

[9] Sahu, S., & Pandey, M. (2019). Efficient load Balancing algorithm analysis in *Cloud Computing*. 2019 International Conference on Communication and Electronics Systems (ICCES). https://doi.org/10.1109/icces45898.2019.9002248.

[10] Kapgate, D., (2014). Improved round robin algorithm for data center selection in *cloud computing. International Journal of Engineering Sciences &amp*; Research Technology 3, 686–691.

[11] Kaur, A., Kaur, B., Singh, P., Devgan, M.S. and Toor, H.K., (2020). Load balancing optimization based on deep learning approach in *cloud environment. International Journal of Information Technology and Computer Science*, 12(3), pp.8-18. https://doi.org/10.5815/ijitcs.2020.03.02.

[12] Kishor, K., & Thapar, V. (2014). An Efficient Service Broker Policy for Cloud Computing Environment. https://api.semanticscholar.org/CorpusID:16760843

[13] Limbani, D., Oza, B., (2012a). A proposed service broker policy for data center selection in cloud environment with implementation In *International Journal of Computer Technology*

[14] Limbani, D., Oza, B., (2012b). A proposed

service broker strategy in cloudanalyst for cost-effective data center selection In *International Journal of Engineering Research and Applications*, India 2,793–797. https://api.semanticscholar.org/CorpusID:61845591

[15] Mahalle, H. S., Tayde, S., & Kaveri, P. R. (2015). Implementing service broker policies in cloud computing environment. *2015 International Conference on Communication Networks (ICCN)*. https://doi.org/10.1109/iccn.2015.37.

[16] Manasrah, A. M., Smadi, T., & ALmomani, A. (2017). A Variable Service Broker Routing Policy for data center selection in cloud analyst. *Journal of King Saud University* - Computer and Information Sciences, 29(3), 365–377. https://doi.org/10.1016/j.jksuci.2015.12.006.

[17] Mehraj, S., & Banday, M. T. (2022). A Dynamic Weighted Averaging Technique for Trust Assessment in *Cloud Computing. International Journal of Cloud Applications and Computing*, 12(1), 1–21. https://doi.org/10.4018/ijcac.297099.

[18] Mishra, R.K., Bhukya, S.N., 2014. Service broker algorithm for cloud-analyst. *International Journal of Computer Science and Information Technologies 5*, 3957–3962. https://api.semanticscholar.org/CorpusID:15390326

[19] Mishra, R. K., Kumar, S., & Sreenu Naik, B. (2014). Priority based Round-Robin service broker algorithm for Cloud-Analyst. 2014 IEEE International Advance Computing Conference (IACC). https://doi.org/10.1109/iadcc.2014.6779438

[20] Nishad, L.S., Kumar, S., Bola, S.K., Beniwal, S., Pareek, A., (2016). Round robin selection of datacenter simulation technique cloudsim and cloud analsyt architecture and making it efficient by using load balancing technique In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 2901–2905.

[21] Patel, H. V., & Patel, R. (2015). Cloud Analyst: An Insight of Service Broker Policy. *IJARCCE*, 122–127. https://doi.org/10.17148/ijarcce.2015.4125.

[22] Radi, M., (2015). Efficient service broker policy for large-scale cloud environments. *arXiv preprint* arXiv:1503.03460. https://api.semanticscholar.org/CorpusID:15422365

[23] Rekha, P. M., & Dakshayini, M. (2018). Dynamic Cost-Load Aware Service Broker Load Balancing in *Virtualization Environment. Procedia Computer Science*, 132, 744–751. https://doi.org/10.1016/j.procs.2018.05.086.

[24] Sheikhani, L., Chang, Y., Gu, C., & Luo, F. (2017). Modifying broker policy for better response time in *datacenters. 2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. https://doi.org/10.1109/compcomm.2017.8322977.

[25] Chen, L., Shen, H., & Sapra, K. (2014). RIAL: Resource Intensity Aware Load balancing in clouds. *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. https://doi.org/10.1109/infocom.2014.6848062

[26] Wickremasinghe, B., Calheiros, R. N., & Buyya, R. (2010). CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications. *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. https://doi.org/10.1109/aina.2010.32.