

Efficient criticality oriented service brokering policy in cloud datacenters

Shanmugapriya Subramanian¹, Priya Natarajan²

¹PG Department of IT and BCA, Dwaraka Doss Goverdhan Doss Vaishnav College, University of Madras, Chennai, India

²Department of Computer Science, Shrimathi Devkunvar Nanalal Bhatt Vaishnav College for Women, University of Madras, Chennai, India

Article Info

Article history:

Received Jun 29, 2023

Revised Oct 25, 2023

Accepted Nov 29, 2023

Keywords:

Cloud computing

CloudAnalyst

Datacenter

Infrastructure as a service

Service broker policy

Virtual machine

ABSTRACT

Cloud service provider (CSP) offers a huge number of datacenters and virtual servers to the users for processing their workloads in an infrastructure as a service (IaaS) cloud computing environment. Due to the heterogeneous volume of these resources and the immense number of user workloads arriving simultaneously in the cloud, it is necessary to use an effective load distribution technique for scheduling the resources to achieve high performance and high user satisfaction. Service brokering policy and load balancing techniques are the two crucial areas to be focused on while selecting the datacenters and virtual machines, respectively. In this study, we have proposed a dynamic efficient criticality-oriented service brokering policy for load allocations among datacenters by considering task criticality, datacenter proximity, and traffic, the size of the datacenter, its present load and makespan value. The proposed methodology is examined against the current policies in the CloudAnalyst simulation tool and the analysis report confirms that our proposed policy gives priority to processing the urgent loads and chooses the optimum datacenter to diminish the load response time, datacenter processing time, minimizes the cost, achieves optimum resource utilization and workload balancing among resources.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Shanmugapriya Subramanian

Department of Computer Applications, Dwaraka Doss Goverdhan Doss Vaishnav College, University of Madras

E.V.R. Periyar High Road, Arumbakkam, Chennai – 600 106, Tamilnadu, India

Email: priyadgvc17@gmail.com

1. INTRODUCTION

Cloud service providers (CSP) build a cloud computing model such as infrastructure as a service (IaaS) to offer large-scale computing resources to the customers with greater flexibility at a low cost and in a highly elastic manner, i.e., provide resources according to their needs. IaaS offers resources like datacenters (DCs), virtual machines (VMs), random access memory (RAM), storage capacity, network connections and databases. Cloud users can scale up or down the resources via the internet on a charge-per-use basis for processing their workloads (applications) without worrying about infrastructure maintenance [1]. The IaaS cloud servicing provider maintains its computing resources in a large number of heterogeneous DCs in several locations. DCs are physical places that are geographically distributed, each containing several virtual servers with numerous computing resources. The resources include servers, processing powers, memory units, storage capacity, network equipment, and VMs to store and process the user's workload [2]. IaaS uses a feature called server virtualization that allows the sharing of a single physical resource among multiple users. Virtualization builds multiple VMs that are software-based machines instantiated on top of each

original piece of physical hardware. It avoids overloading or under loading of VMs and wastage of resources [3], [4]. Figure 1 depicts the model of the virtualization technique and the IaaS cloud services are deployed in various VMs. It offers resources to ensure efficient utilization of resources using fewer infrastructures and less cost. It also promotes rapid execution of the user's applications [5].

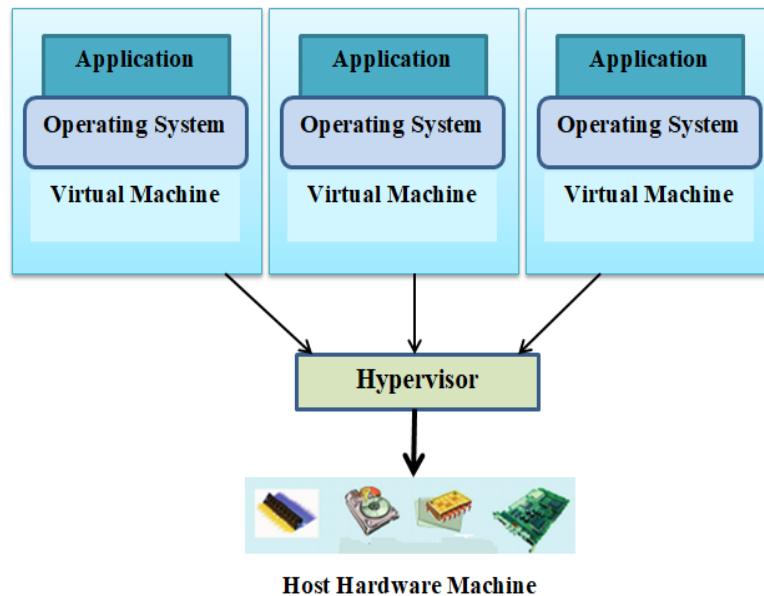


Figure 1. Architecture of virtualization

Service brokering and load balancing are the two key techniques used in IaaS cloud environments for selecting the DC and VM, respectively. Cloud service brokering is a DC selection technique that selects the appropriate DC by incorporating one of the service broker policies (SBP) for routing the consumer's loads to the appropriate DCs. Load balancing (LB) is the resource allocation technique used in IaaS cloud computing. LB uniformly allocates the user's workload over several resources according to their demands in an IaaS cloud environment [6], [7].

Due to the heterogeneous and dynamic nature of each user load, it is very important to dynamically know the current state of the available DC. It is necessary to examine the DC status, which includes its availability, capacity, and distance from the user's location and current load for efficient resource management. Dynamic scheduling provides efficient resource allocation and it is best suited for an IaaS heterogeneous cloud environment with a varied range of DCs and user applications [8].

CloudAnalyst is a modelling and simulation tool used in an IaaS cloud environment to assess the effectiveness of service broker policies and load balancing policies, and it has been built upon the CloudSim tool. CloudAnalyst has a graphical user interface (GUI) feature that shows the simulated results with charts and graphs. The cloud analyst divides the whole world into six regions, which contains several user bases (UB) and DCs. Each DC contains several virtual servers, and each UB consists of several user loads [9], [10]

On the grounds of increasing need for cloud services, the varied nature of cloud resources, and the exponential rise of workloads, it is challenging to select the optimum DC for a given workload. Improper allocation of workloads among DCs worsens the effectiveness of the entire IaaS cloud system. It also maximizes load response and processing time, increases cost, and results in inefficient resource utilization. [11], [12]. In this regard, service brokering in the IaaS cloud has been thoroughly explored over the past couple of decades. Numerous authors have put forth various service brokering policies for DC selection. This section highlights a few current research projects on cloud load balancing.

The service proximity based or closest DC service broker policy (SPB-SBP) services the user's load by selecting the DC in the closest region based on network proximity. It estimates the network proximity based on the distance between each DC from the user's locality and selects the DC with the lowest network latency. The problem arises when there is multiple DC present in the same closest region and the DC is chosen at random, it overloads the nearest DC, and its communication link leads to a high response time for user loads [13], [14].

The optimize response time service broker policy (ORT-SBP) first finds the closest DCs using SPB-SBP and estimates the response time of each DC based on its previous performance. This policy records the network delay and response time of each DC that was previously serviced by the DC and sends the traffic to the closest DC offering the fastest response time. If the closest DC does not have an optimum response time, it picks another DC with a good response time without considering the delay in workload allocation. But this policy does not assign any workload to the DC that has not received any loads previously [15], [16].

The reconfigure dynamically load (RDL-SBP) policy follows the SPB-SBP for choosing the appropriate DC. It also offers elasticity in expanding the number of VMs based on the user's demand, i.e., increasing and decreasing the number of VMs grounded on the workload arriving from the users. This policy incurs additional costs and also affects performance due to the increase of VMs [17], [18].

The main significant performance evaluation parameters used by the service broker policies are: resource utilization refers to how effectively the various resources are utilized by each DC. Makespan or completion time is the total time that the DC needs to process a set of loads for its whole execution. It should be minimized to improve the resource utilization ratio [19]. Response time is the total amount of transit period, waiting time, and processing interval that the DC spends responding to a specific task. For better system efficiency, the workload necessities to be distributed among suitable DCs in a way that minimizes response time task deadline is a very important parameter of a task that specifies the timeline of the task, which indicates when the task must be completed [20], [21].

Cost includes the price of the machine and the cost of processing the workload [22]. DC processing time is the exact period that the DC needs to process a user's load [23]. Throughput is the amount of workload that the DCs process per unit of time [24]. Network latency specifies the distance from the UB to the DC. i.e., the delay or amount of time taken to transport the load between the UB and the DC [25]. Network traffic represents the amount of workload moving across a DC at any given time [26].

The problem with existing policies is that, even though many service broker policies are incorporated, workload priority is not considered while selecting the DC itself. It also requires an efficient policy to cover almost all the important performance factors like network traffic, response and processing time, cost, workload balancing, resource utilization, and makespan while selecting the DC. In this research, we focus on choosing high-severity loads and identifying the optimum DC.

Our research proposes an efficient criticality-oriented service broker policy (ECO-SBP) in an IaaS cloud environment, it mainly focuses on selecting the optimum DC and balancing the user's workload among DCs that are scattered in different regions. Here, the original service proximity based (SPB-SBP) has been modified by eliminating the random selection of the DC. It also improves the method of choosing the DC by reducing the loads response time, DC processing time, and cost and achieving good workload balancing and incorporating the dynamic selection of the DC.

2. PROPOSED METHOD

In this research, we have enhanced the existing service proximity-based service brokering policy (SPB-SBP) also called as closest DC policy and proposed the ECO-SBP for finding the best DC and prioritizing high-severity loads. In the existing SPB-SBP, all the loads are treated equally and scheduled according to the order they arrive, i.e., it does not consider the load's completion time and always chooses the nearest DC concerning the DC's network latency, leads to high traffic and overloading of some DCs while others are idle. If multiple DCs arriving at the same geographical location then this policy randomly chooses any one DC. Moreover, this policy is static in nature, it does not consider the current performance of the DC, which leads to high response time for certain loads and high processing time taken by the DC, as well as increased costs due to the poor allotment of DC.

The proposed ECO-SBP eradicates the problem of SPB-SBP and focuses primarily on dynamically scheduling the user's loads across multiple DCs in an IaaS heterogeneous cloud environment. The main important factors to be considered are task criticality, loads response time, DC processing time, cost, and workload balancing. Figure 2 displays the overall architectural functioning of the proposed ECO-SBP policy, in which each user's workload is forwarded to the DC by a DC controller (DCC) using an appropriate service broker policy and forwards to VM load balancer to assign the load to the right VM. Once the load is processed, the result is sent back to the user.

The proposed ECO-SBP implements the existing SPB-SBP for locating the closest region DCs in terms of network proximity. The proposed ECO-SBP policy operates in two phases, as depicted in Figure 3. In the first phase, it chooses the load based on its severity value i.e., the loads to be completed within the stipulated time are processed first. In the second phase, it chooses the DC depending on a few factors such as network delay, network traffic, DC size, present load, and makespan value. After that, the load is allotted to the appropriate DC.

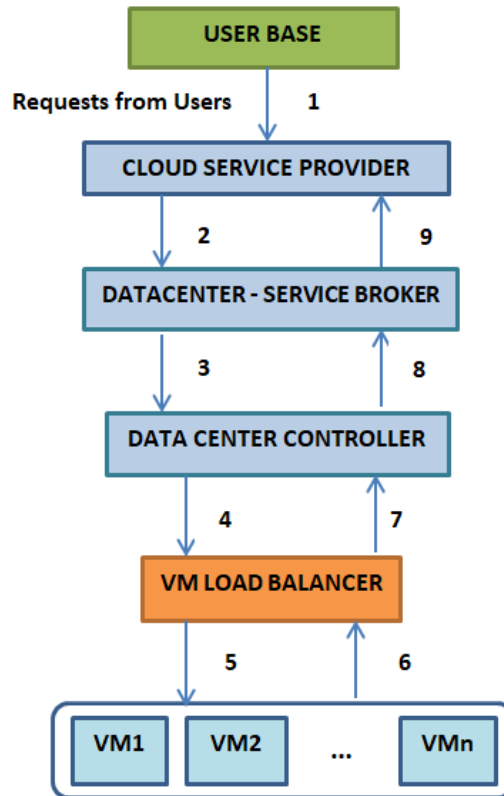


Figure 2. Generic framework of proposed ECO-SBP

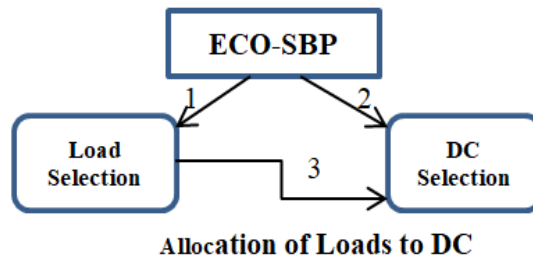


Figure 3. Phases of proposed ECO-SBP

2.1. Load selection

Here, the user's loads are divided into critical loads (CL) and non-critical loads (NCL) according to time-critical constraints such as severity values, and then it calculates the target time of each load based on the expected completion time. Table 1 shows the sample loads with their severity values, types, and target time, where UB1L1 represents load L1, which belongs to UB 1. Loads with low severity values are the most critical loads, which have a shorter target time and are executed as soon as they are received by the system. Non-critical loads have a larger target time that has been kept in the loads queue according to the severity value.

Table 1. Loads with severity and estimated processing time

Load number	Load severity	Load type	Load target time (milliseconds)	Load execution sequence
UB1L1	1	CL	5.66	1
UB1L2	2	CL	7	3
UB2L1	5	NCL	15	5
UB3L1	2	CL	6.87	2
UB3L2	4	NCL	10	4

2.2. DC selection

The principal objective of the proposed ECO-SBP approach is to select the entire closest region DCs based on network proximity, now, select all the DCs with low network traffic, which is determined by the network latency estimated by the number of loads moving across a DC. Now, choose the high-capacity DC by determining their size by evaluating the capacity of each VM present in that DC, and find the present load of each DC by adding the sizes of all the loads assigned to all the VMs of that DC. Now, find the remaining capacity of each DC by deducting the present load from the overall size to obtain the current size of the DC. The next step is to determine the makespan value for each DC in the HashMap table, makespan is estimated based on the length of time it takes the DC to process a given set of loads. Choose all the DCs whose size is greater than the load size and select the one with the lowest makespan value to process the load.

The ECO-SBP yields a good response and processing time. This approach also aims to reduce the overloading of the DC by allocating an appropriate DC based on the load size and also dynamically redirecting the non-critical loads to the next closest DC, which has a lower number of current loads. The DC network proximity is obtained in (1).

$$DC_{np} = Pd_l + Sd_l \quad (1)$$

where DC_{np} represents network proximity, Pd_l refers to propagation delay in length and Sd_l refers to serialization delay in length. $Pd_l = D/S$ where D refers distance i.e., length of the physical link between the UB and the DC and S refers to transmission speed. $Sd_l = L_s/T_r$, where L_s refers to load size (bits) and T_r refers to transmission rate (bps). The traffic intensity factor of a DC is the average server or resource utilisation over a given time period. The DC traffic intensity factor is formulated as (2).

$$DC_{tf} = (L_s * a)/R \quad (2)$$

where DC_{tf} refers to the traffic factor of a DC, L_s refers to a constant load size in that particular DC, a is the average rate of load per second, and R is the constant transmission rate. The DC threshold is given in (3).

$$DC_{th} = L(MaxCnfig(DC) + MinP_l(DC) + MinM_s(DC)) \quad (3)$$

where $MaxCnfig$ refers to high configuration DC, $MinP_l$ refers to minimum present load and $MinM_s$ refers to minimum makespan value DC. The DC response time is obtained in (4).

$$DC_{rt} = (LD_s/(BW + DC_l)) + DC_{pt} \quad (4)$$

where LD_s refers to load size, BW refers to bandwidth, DC_l refers to latency or network transfer delay, and DC_{pt} refers to DC processing time. The DC processing time is obtained from (5).

$$DC_{pt} = DC_{rt} - (2 * DC_{np}) \quad (5)$$

where DC_{np} is the network transfer delay and DC_{rt} refers the response time.

3. METHOD

3.1. Procedure of ECO-SBP

The design of the ECO-SBP policy is explained by employing a below procedure to determine every step of the process. This procedure explains the two phases of the ECO-SBP policy. In phase 1, it demonstrates the way loads are split into critical and non-critical and stored using a HashMap table. In phase 2, it shows the selection of an efficient data center based on various factors.

Procedure of ECO-SBP

Input: UB numbers ($UB1, UB2, \dots, UBn$), number of loads $L = L1, L2, L3 \dots Ln$, target time of the load (TT), severity value of the load (SV_i), region number of the load ($R0, R1, \dots, R6$), Load Size (LS_i), DC list ($DCIndexList$), DC status (network delay, network traffic, size, present load, and makespan values).

Output: DC name ($DCnme$), response time, processing time and cost.

Steps:

- Create a queue to maintain the user's loads. A cloudlet is a Gridlet created by the UBs for each user's load which is received from different UBs and each cloudlet contains a unique cloudlet id, load size, UB name, region name, severity value and adds each cloudlet.

- b. The DCC sends each cloudlet's details to the ECO-SBP by using *getGridletStatus()*.
- c. The ECO-SBP estimates the target time (TT) for each cloudlet based on each load's severity value and creates two HashMap tables, such as critical load HashMap (CLH) and non-critical load HashMap (NCLH), to keep the CL and NCLs respectively based on the target time, arranges the loads in both the HashMap tables from the minimum target time to maximum target time, chooses a load from CLH with the minimum TT value, and forwards to DCC.
- d. Using *DCList = GridSim.getGridResourceList*, the ECO-SBP policy obtains all the available DCs and stores the *DC_ID* and *DC_region* in the *DCHashMap* (DCHM) table.
- e. Now, estimate the network proximity of all DCs from the selected loads UB, select all the closest DCs from that UB, estimate the network traffic and assigns the threshold value based on the traffic and stores all the non-congested DCs based on the threshold in the *NewDCHashMap* table.
- f. Now, estimate the size and present load of each DC in the *NewDCHashMap* (NDCHM) table, find the current size and choose all the DCs with a current size greater than the load size, update the current size of each DC in the *NewDCHashMap* table and market it as 1 in the *DC_Selected* field.
- g. Retrieves all the DCs with the *DC_selected* field is set to 1 via *getDCCharacteristics*, estimates the makespan value and sorts the DCs according to their makespan in the *NewDCHashMap* table.
- h. The ECO-SBP chooses the DC with a low makespan value and allocates the load. The ECO-SBP first allocates all the CLH loads to the appropriate DCs and then directs all the NCLH loads to the remaining DCs and updates the DC status.
- i. Repeat the previous procedures until all of the loads have been assigned if further loads come onto the UB.
- j. Stop

3.2. The ECO-SBP pseudo code

The Pseudo code of the ECO-SBP policy describes the selection of the datacenter for load allocations in the CloudAnalyst tool. It shows the calculation of each load's target time using its severity value. It also mentions the creation of the HashMap table and how its partitions were made. Finally, the selection of the appropriate datacenters according to their proximity, traffic from the userbase, size, current allocations, and makespan value is outlined in the Pseudo code.

Pseudo code of the ECO-SBP

```
Create Queue Q, CLH, NCLH, DCHM, NDCHM //Create All Hash Map Tables
For each new load from each UB
  DCC Do
    Li_TT <- TT (Li_SV) //Fix Loads Target Time
    If (Li_SV <= 2) then CLH<-Li_ID
    Else
      NCLH<-Li_ID
    End If End For
For each load from CLH Do
  L <- Min (Li_TT)//select load with minimum TT
End For Return L
For each DC do
  DCHM <- Avialble (DC_id, DC_region) //keep all the available DCs into hash map table
  If (DC_NTdelay < DC_NTdelaythreshold) AND //select all the Closest DC
  If (DC_NTtraffic < DC_NTtrafficthreshold) then //select all the DCs with less traffic
    NDCHM = DC_ID
  End If End For.
For each DC in NDCHM Do
  DC_CurrentSize = DC_ActualSize - DC_UtilizationSize
  Update NDCHM
  If (L_Size < DC_CurrentSize) then DC_Status = 1
    Find DCMakespan for DC_Status = 1
  End If End For
DCselect <- min (DCMakespan)
Allocate (DCselect, L)
Return DC_ID.
```

3.3. Diagrammatic representation of routing loads of ECO-SBP

Figure 4 illustrates the step-by-step process of routing the loads to the appropriate DC by the proposed ECO-SBP policy. All incoming user loads arriving at the UBs are forwarded to the DCC, which in turn assigns a unique load id, and arranges them in the load queue with load size, region name, UB name, and severity value. The DCC forwards the loads to the ECO-SBP policy, which in turn calculates the target time of each load based on the severity value, chooses the minimum target time load, and allocates it to the efficient DC concerning the criteria given in the DC proficiency list.

3.4. Performance comparison of proposed ECO-SBP with SPB, ORT and RDL

The key features of the ECO-SBP policy in comparison to the current SPB-SBP, ORT-SBP, and RDL-SBP are shown in Table 2. The proposed ECO-SBP policy gives good performance since it also considers additional important factors such as load size, load severity, DC proximity, traffic, size, present load, and makespan. This policy also achieves optimum resource utilization and evenly balances the workload among the available DCs.

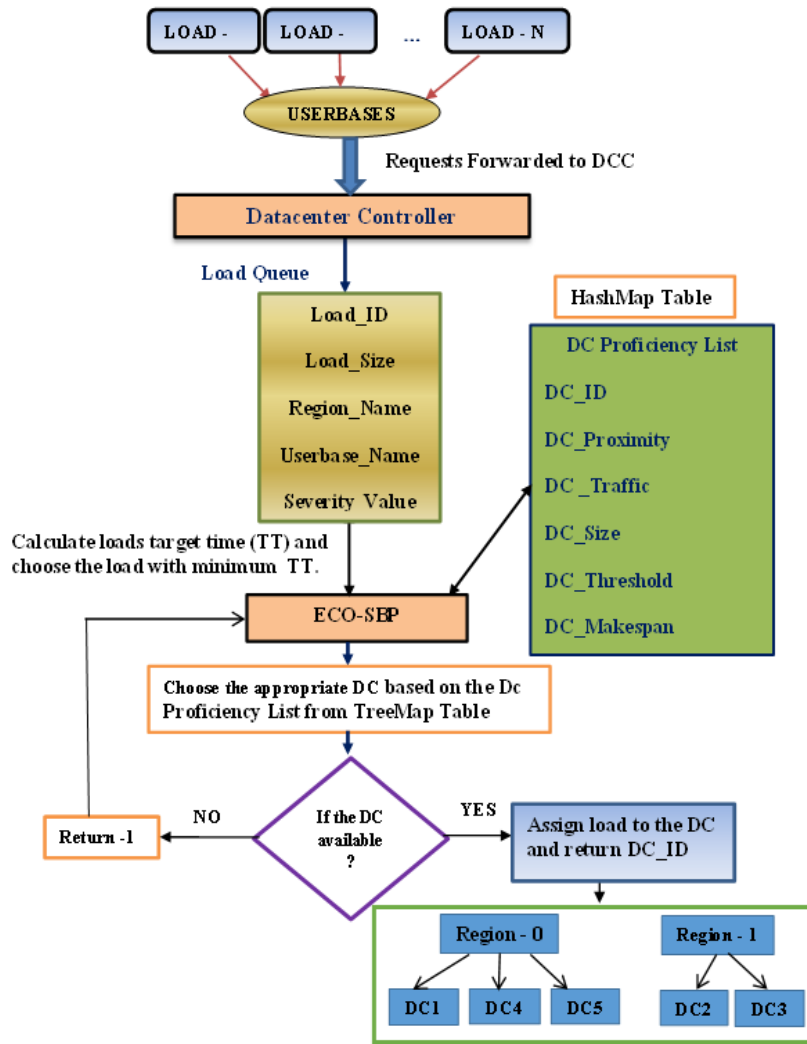


Figure 4. Load routing of proposed ECO-SBP

Table 2. The performance comparison of proposed ECO-SBP

Policy name	Load size	Load severity	DC latency	DC traffic	DC capacity	Present load DC	Resource utilization	Workload balancing
SPB	NO	NO	YES	YES	NO	NO	NO	NO
ORT	NO	NO	YES	YES	NO	NO	NO	NO
RDL	NO	NO	YES	YES	NO	NO	NO	NO
ECO	YES	YES	YES	YES	YES	YES	YES	YES

4. RESULTS AND DISCUSSION

4.1. Simulation parameters

In this experiment, the simulation parameter mainly includes the characteristics of DCs, physical machines, UBs and VMs. The experiments employed six regions (R0 to R5), twelve UBs (UB1 to UB12), five DCs (DC1 to DC5), two physical hosts in each DC and six VMs in each physical host. VM allocation policy used is time-shared policy and load balancing algorithms used is round Robin algorithm for VM selection. Figure 5 depicts the structure of DCs and physical hardware details of the DC in the CloudAnalyst tool and Figure 6 depicts the characteristics of the UBs which include the region number, number of requests, and size of the request, peak hours and average peak users.

4.2. Simulation environment

The proposed ECO-SBP is implemented in the CloudAnalyst simulation tool, and the experiment is done to assess the proposed ECO-SBP performance concerning the DC response and processing time, VM

cost, and processing cost with the current policies, namely service proximity-based service broker policy (SPB-SBP), optimized response time service brokering policy (ORT-SBP), and reconfigure dynamically with the load (RDL-SBP). The new ECO-SBP is implemented in CloudAnalyst service broker policies, as shown in Figure 6.

Configure Simulation

Main Configuration | **Data Center Configuration** | Advanced

Data Centers:

Name	Region	Arch	OS	VMM	Cost per VM \$/Hr	Memory Cost \$/s	Storage Cost \$/s	Data Transfer Cost \$/Gb	Physical HW Units
DC1		0 x86	Linux	Xen	0.1	0.05	0.1	0.1	2
DC2		1 x86	Linux	Xen	0.1	0.05	0.1	0.1	2
DC3		1 x86	Linux	Xen	0.1	0.05	0.1	0.1	2
DC4		0 x86	Linux	Xen	0.1	0.05	0.1	0.1	2
DC5		0 x86	Linux	Xen	0.1	0.05	0.1	0.1	2

Add New
Remove

Physical Hardware Details of Data Center : DC1

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	102480	100000000	1000000	20	10000	TIME_SHARED
1	204800	100000000	1000000	30	20000	TIME_SHARED

Add New
Copy
Remove

Figure 5. DC configuration

Configure Simulation

Main Configuration | **Data Center Configuration** | Advanced

Simulation Duration: ▾

User bases:

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	0	60	100	8	12	100000	10000
UB2	1	60	1000	12	15	500000	50000
UB3	1	60	3000	15	18	350000	35000
UB4	0	60	2000	18	22	750000	75000
UB5	3	60	500	6	8	200000	20000

Add New
Remove

Application Deployment Configuration:

Service Broker Policy: ▾

Data Center	# VMs	Image Size	Memory	BW
DC1	60	10000	4096	1000
DC2	60	10000	4096	1000
DC3	60	10000	2048	1000
DC4	60	10000	2048	1000
DC5	60	10000	4096	1000

Add New
Remove

Figure 6. UB properties and proposed ECO-SBP

4.3. Obtained results

Four testing scenarios were set up by considering various conditions such as varying loads, varying number of UBs and number of requests in each UBs and five number of DC and sixty number of VMs as shown in Table 3. The thorough outcome of the experiments of each scenario consisting of various service brokering policies is summarized in Tables 4, 5, 6 and 7. Each experiment shows the load response time in milliseconds, processing time in milliseconds and total cost in dollar value which includes VM cost and also the DCs processing cost with round Robin (RR) load balancing strategy.

Table 3. Scenario description

Scenario	Number of UBs	Number of DC	Total VMs	Number of loads (Per UB)
Scenario – 1	6	5	60	5000
Scenario – 2	8	5	60	12000
Scenario – 3	10	5	60	17000
Scenario – 4	12	5	60	20000

Table 4. Experimental results of scenario – 1

Service brokering policy	Response time (milliseconds)			DC processing time (milliseconds)			Cost		
	Avg	Min	Max	Avg	Min	Max	VM cost	Data transfer cost	Total cost
SPB-SBP	203.21	40.56	757.85	130.56	22.10	223.10	1.93	29.59	31.52
ORT-SBP	210.48	41.34	757.44	135.65	24.89	224.22	2.10	25.89	27.99
RDL-SBP	213.48	43.56	870.67	140.56	35.43	258.56	3.70	30.43	34.13
ECO-SBP	196.48	35.43	700.32	125.65	20.43	214.22	1.50	20.43	21.93

Table 5. Experimental results of scenario – 2

Service brokering policy	Response time (milliseconds)			DC processing time (milliseconds)			Cost		
	Avg	Min	Max	Avg	Min	Max	VM cost (\$)	Data transfer cost (\$)	Total cost (\$)
SPB-SBP	310.34	60.56	854.58	235.10	38.15	332.01	4.17	38.33	42.5
ORT-SBP	310.10	61.34	859.44	235.50	38.89	324.22	4.40	38.98	43.38
RDL-SBP	318.10	71.34	887.44	255.50	47.89	374.12	5.10	55.78	60.88
ECO-SBP	270.89	55.43	810.32	190.89	28.43	318.21	3.60	24.55	28.15

Table 6. Experimental results of scenario – 3

Service brokering policy	Response time (milliseconds)			DC processing time (milliseconds)			Cost		
	Avg	Min	Max	Avg	Min	Max	VM cost (\$)	Data transfer cost (\$)	Total cost (\$)
SPB-SBP	361.43	82.65	1072.3	375.01	51.10	483.10	6.95	49.95	56.9
ORT-SBP	360.56	83.34	1571.1	375.87	57.00	463.23	5.30	43.87	49.17
RDL-SBP	428.32	89.45	1695.2	415.50	59.89	494.22	7.10	55.89	62.99
ECO-SBP	309.90	65.40	932.5	222.32	38.00	418.22	4.80	29.43	34.73

Table 7. Experimental results of scenario – 4

Service brokering policy	Response time (milliseconds)			DC processing time (milliseconds)			Cost		
	Avg	Min	Max	Avg	Min	Max	VM cost (\$)	Data transfer cost (\$)	Total cost (\$)
SPB-SBP	523.78	140.6	1123.6	452.11	68.66	443.00	9.73	50.59	60.32
ORT-SBP	524.96	141.4	1104.9	463.76	69.90	443.03	10.10	50.89	60.99
RDL-SBP	548.32	199.5	1465.1	500.50	75.89	494.66	12.10	65.89	77.99
ECO-SBP	454.00	100.0	992.3	322.32	38.02	388.11	7.90	38.43	46.33

The charts depicted in Figures 7(a) to 7(c) indicates the visualized form of comparing the results of the loads response time, DC Processing time along with the cost of each service brokering policies. It is perceived that the proposed ECO-SBP policy shows an improvement in load response time, DC processing time and also the cost of executing the load, especially in the case of an increase in the number of loads. The ECO-SBP policy chooses the best DC with minimum distance, less traffic, less loaded and minimum makespan value for each load arrived in the UB in such a way that it always yields the optimum response time, DC processing time, and cost than other policies.

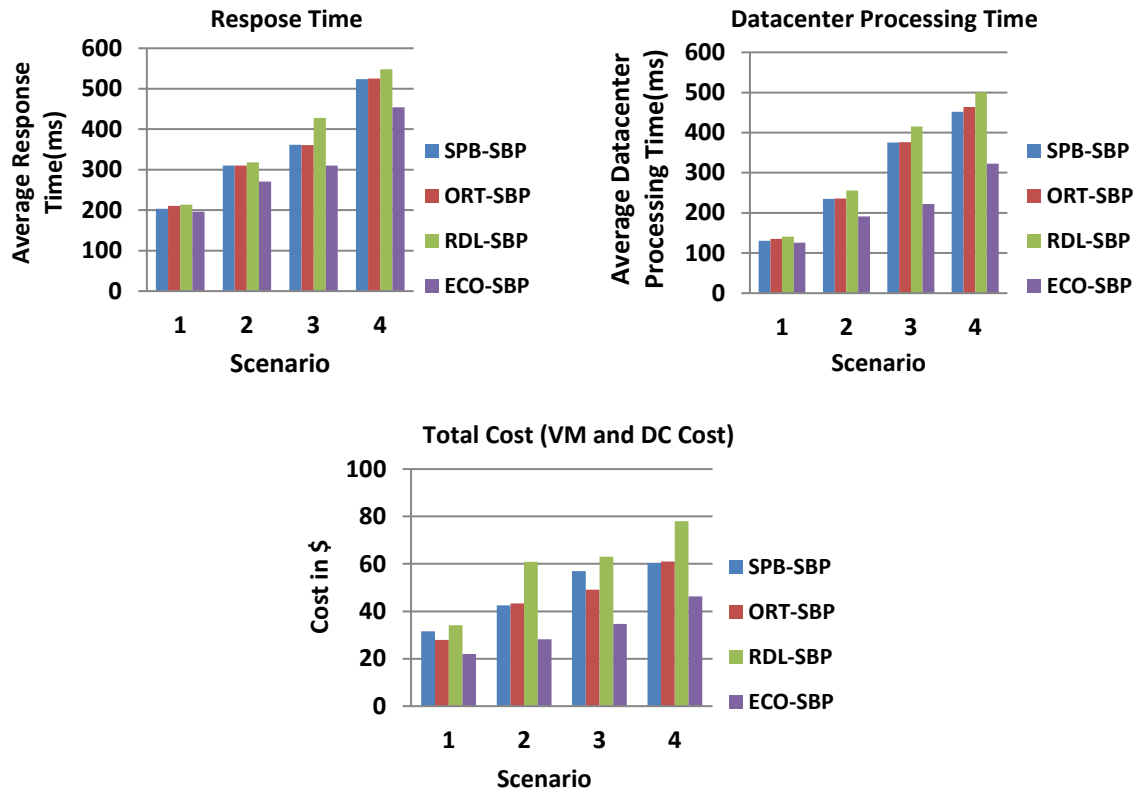


Figure 7. Comparison of ECO-SBP policy implemented in CloudAnalyst (a) average response time comparison, (b) average datacenter processing time comparison, and (c) cost comparison

5. CONCLUSION

In this research, we proposed and evaluated an ECO-SBP for efficient scheduling of users loads to suitable datacenters. In contrast to existing policies, the proposed policy is appropriate for a dynamic, large-scale heterogeneous IaaS environment. The proposed ECO-SBP is thoroughly assessed in the IaaS cloud environment using the CloudAnalyst simulation tool with variable userbase size and datacenter characteristics under different simulation scenarios. ECO-SBP is examined with the existing policies and the result indicates that the proposed policy improves cloud system performance by picking the appropriate DC in the first instance itself, and provides the best response and processing time with a reasonable cost range. It ensures workload balancing, maximizes resource utilization, reduces makespan, and prevents DCs from overloading or underloading even in case of heavy workloads.




REFERENCES

- [1] R. Tasneem and M. A. Jabbar, "An insight into load balancing in cloud computing," in *International Conference on Wireless Communications, Networking and Applications*, 2022, pp. 1125–1140, doi: 10.1007/978-981-19-2456-9_113.
- [2] A. Khodar, V. E. Mager, I. Alkhayat, F. A. Jebur Al-Soudani, and E. N. Desyatirikova, "Evaluation and analysis of service broker algorithms in cloud-analyst," in *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIconRus)*, Jan. 2020, pp. 351–355, doi: 10.1109/eiconrus49466.2020.9039187.
- [3] S. Talwani *et al.*, "Machine-learning-based approach for virtual machine allocation and migration," *Electronics*, vol. 11, no. 19, Oct. 2022, doi: 10.3390/electronics11193249.
- [4] S. Wiriya, W. Wongthai, and T. Phoka, "The enhancement of logging system accuracy for infrastructure as a service cloud," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 9, no. 4, pp. 1558–1568, Aug. 2020, doi: 10.11591/eei.v9i4.2011.
- [5] H. Shukur, S. Zeebaree, R. Zebari, D. Zeebaree, O. Ahmed, and A. Salih, "Cloud computing virtualization of resources allocation for distributed systems," *Journal of Applied Science and Technology Trends*, vol. 1, no. 3, pp. 98–105, Jun. 2020, doi: 10.38094/jastt1331.
- [6] M. A. Shahid, M. M. Alam, and M. M. Su'ud, "Performance evaluation of load-balancing algorithms with different service broker policies for cloud computing," *Applied Sciences*, vol. 13, no. 3, Jan. 2023, doi: 10.3390/app13031586.
- [7] M. A. Elmagzoub, D. Syed, A. Shaikh, N. Islam, A. Alghamdi, and S. Rizwan, "A survey of swarm intelligence based load balancing techniques in cloud computing environment," *Electronics*, vol. 10, no. 21, Nov. 2021, doi: 10.3390/electronics10212718.
- [8] S. Raghuvanshi and S. Kapoor, "The new service brokering policy for cloud computing based on optimization techniques," *International Journal of Engineering and Techniques*, vol. 4, no. 3, pp. 481–488, 2018.




- [9] C. Jittawiriyankoon, "Evaluation of load balancing approaches for Erlang concurrent application in cloud systems," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 4, pp. 1795–1801, Aug. 2020, doi: 10.12928/telkomnika.v18i4.13150.
- [10] A. Y. Ahmad and A. Y. Hammo, "A comparative study of the performance of load balancing algorithms using cloud analyst," *Webology*, vol. 19, no. 1, pp. 4898–4911, Jan. 2022, doi: 10.14704/web/v19i1/web19328.
- [11] P. M. Rekha and M. Dakshayini, "Dynamic cost-load aware service broker load balancing in virtualization environment," *Procedia Computer Science*, vol. 132, pp. 744–751, 2018, doi: 10.1016/j.procs.2018.05.086.
- [12] M. A. Khan, "Optimized hybrid service brokering for multi-cloud architectures," *The Journal of Supercomputing*, vol. 76, no. 1, pp. 666–687, Oct. 2019, doi: 10.1007/s11227-019-03048-5.
- [13] B. Nayak, B. Bisoyi, and P. K. Pattnaik, "Data center selection through service broker policy in cloud computing environment," *Materials Today: Proceedings*, vol. 80, pp. 2218–2223, 2023, doi: 10.1016/j.matpr.2021.06.185.
- [14] M. Al-Tarawneh and A. Al-Mousa, "Adaptive user-oriented fuzzy-based service broker for cloud services," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 2, pp. 354–364, Feb. 2022, doi: 10.1016/j.jksuci.2019.11.004.
- [15] A. Sankla, "Analysis of service broker policies in cloud analyst framework," *Journal of Advance Research in Computer Science & Engineering*, vol. 2, no. 4, pp. 32–37, Apr. 2015, doi: 10.53555/nncse.v2i4.456.
- [16] A. I. El Karadawy, A. A. Mawgoud, and H. M. Rady, "An empirical analysis on load balancing and service broker techniques using cloud analyst simulator," *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, Feb. 2020, doi: 10.1109/itce48509.2020.9047753.
- [17] T. Menakadevi, "An optimum service broker policy for selecting data center in cloud analyst," *International Journal of Research in Engineering and Technology*, vol. 05, no. 09, pp. 76–84, Sep. 2016, doi: 10.15623/ijret.2016.0509011.
- [18] P. Rani, R. Chauhan, and R. Chauhan, "An enhancement in service broker policy for cloud-analyst," *International Journal of Computer Applications*, vol. 115, no. 12, pp. 5–8, Apr. 2015, doi: 10.5120/20201-2450.
- [19] A. A. A. Alkhatib, A. Alsabbagh, R. Maraqa, and S. Alzubi, "Load balancing techniques in cloud computing: Extensive review," *Advances in Science, Technology and Engineering Systems Journal*, vol. 6, no. 2, pp. 860–870, Apr. 2021, doi: 10.25046/aj060299.
- [20] K. Balaji, "Load balancing in cloud computing: Issues and challenges," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 2, pp. 3077–3084, Apr. 2021, doi: 10.17762/turcomat.v12i2.2350.
- [21] A. Abuhamdah and M. Al-Shabi, "Hybrid load balancing algorithm for fog computing environment," *International Journal of Software Engineering and Computer Systems*, vol. 8, no. 1, pp. 11–21, Jan. 2022, doi: 10.15282/ijsecs.8.1.2022.2.0092.
- [22] A. Kazeem Moses, A. Joseph Bamidele, O. Roseline Oluwaseun, S. Misra, and A. Abidemi Emmanuel, "Applicability of MMRR load balancing algorithm in cloud computing," *International Journal of Computer Mathematics: Computer Systems Theory*, vol. 6, no. 1, pp. 7–20, Dec. 2020, doi: 10.1080/23799927.2020.1854864.
- [23] R. B. Reddy and M. Indiramma, "Efficient throttled load balancing algorithm to improve the response time and processing time in data center," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 23, Jul. 2022, doi: 10.1002/cpe.7208.
- [24] J. M. Shah, K. Kotecha, S. Pandya, D. B. Choksi, and N. Joshi, "Load balancing in cloud computing: Methodological survey on different types of algorithm," *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, May 2017, doi: 10.1109/icoei.2017.8300865.
- [25] M. Vinoth Kumar, K. Venkatachalam, M. Masud, and M. Abouhawahash, "Novel dynamic scaling algorithm for energy efficient cloud computing," *Intelligent Automation & Soft Computing*, vol. 33, no. 3, pp. 1547–1559, 2022, doi: 10.32604/iasc.2022.023961.
- [26] X. Fu, C. Zhang, J. Chen, L. Zhang, and L. Qiao, "Network traffic based virtual machine migration in cloud computing environment," *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Mar. 2019, doi: 10.1109/itnec.2019.8729184.

BIOGRAPHIES OF AUTHORS



Shanmugapriya Subramanian    received the MCA degree and M.Phil. in computer science from Madurai Kamaraj University, Madurai. Currently, she is an assistant professor at the PG Department of Computer Applications, DDGDVC, Chennai, India. She is currently pursuing her PhD in Shrimathi Devkunvar Nanalal Bhatt Vaishnav College for Women, Chennai. Her research interests include networking, artificial intelligence, machine learning, and internet of things in addition to cloud computing. She can be contacted at email: priyadgvc17@gmail.com.



Priya Natarajan    received is a technically skillful academician, currently she is working as an associate professor at the research Department of Computer Science in Shrimathi Devkunvar Nanalal Bhatt Vaishnav College for Women, Tamil Nadu, India for more than 15 years. She holds a PhD from Bharathiar University, Coimbatore. She qualified the NET examination. She is an expert in teaching artificial intelligence, machine learning, data mining, and soft computing. She can be contacted at email: dmpriya2015@gmail.com.