RESEARCH
www.commprac.com
ISSN 1462 2815

PERFORMANCE ASSESSMENT OF THROTTLED ADAPTED LOAD BALANCING ALGORITHM WITH EFFICIENT CRITICALITY ORIENTED BROKER POLICY IN IaaS CLOUD

S. Shanmuga Priya [1]* and N. Priya [2]

[1] Assistant Professor, PG Department of IT and BCA, Dwaraka Doss Goverdhan Doss Vaishnav College, Arumbakkam, University of Madras, India.
*Corresponding Author Email: priyadgvc17@gmail.com
[2] PG Department of Computer Science, Shrimathi Devkunvar Nanalal Bhatt Vaishnav College for Women, Chrompet, University of Madras, India.
Email: drnpriya2015@gmail.com

DOI: 10.5281/zenodo.13166230

Abstract

Cloud Computing is a multitenant technology that offers infrastructure as a Service (IaaS), in which routing the user's workload to the appropriate machine is one of the key factors of a cloud environment. One of the important activities in an IaaS cloud environment to be taken into account is the distribution of a user's workload across many heterogeneous resources, such as Data Centers (DC) and Virtual Machines (VM). Service Brokering Policies (SBP) and Load Balancing (LB) are the significant strategies used for workload distribution on the IaaS platform to improve the overall efficiency of the cloud system. In this study, the Efficient Criticality-Oriented Service Broker Policy (ECO-SBP) and the Throttled Adapted Load Balancing (TALB) algorithms are combined to carry out service brokering and load balancing among DCs and VMs, respectively, for the purpose of minimizing the Response Time (RT) and Data Center Processing Time (DCPT), reducing cost, minimizing Searching Time (ST) and overcoming the issue of load unbalancing. These strategies are implemented in the CloudAnalyst simulation tool and examined against some of the significant existing service brokering policies with load balancing strategies available in the CloudAnalyst tool.

Keywords: Cloud Computing, Infrastructure as a Service, Load Balancing, Service Brokering Policy, Datacenter, Virtual Machines, Throttled Adapted Load Balancing, Efficient Criticality-Oriented Service Broker Policy.

1. INTRODUCTION

Cloud computing is a universal system that distributes computing resources, including servers, memory, storage, access to networks, database servers, execution platforms, and readily available software, to service consumers according to the service level agreements and charges them according to usage. Rendering to the resources it provides to the client, cloud computing services can be divided into three distinct service models: IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service) [1] [2].

The IaaS cloud implements resource scheduling, which divides user workloads among several resources to process their requirements. Resource scheduling is currently a major concern in IaaS cloud computing that requires a robust resource scheduling strategy due to the fast-growing demand and substantial increases in computing resources in cloud computing that impact the overall performance of the IaaS cloud [3].

IaaS primarily offers two important resources, namely VMs and DC. The DCs are physical locations spread across the world that houses several heterogeneous virtual machines (VMs) that are assigned to users for processing their workloads. IaaS includes two ubiquitous techniques for resource scheduling among various DCs and

VMs, such as Service Brokering Policies and Load Balancing techniques, respectively [4].

Resource scheduling can be static or dynamic in nature. Static allocation allows workloads to be scheduled ahead of time, whereas dynamic allocation allows scheduling to occur while the workload is being executed. The static method needs prior knowledge of the VM's performance and once the load has been assigned to a specific VM, it cannot be moved to any other VM, even in the event of overloading or resource failures [5].

The dynamic method is best suited for distributed systems and heterogeneous cloud environments because it dynamically allocates workloads among VMs while taking into the state changes of the VMs during execution, and it is also feasible to migrate workloads from one VM to another in the event of any VM failures or overloading [6].

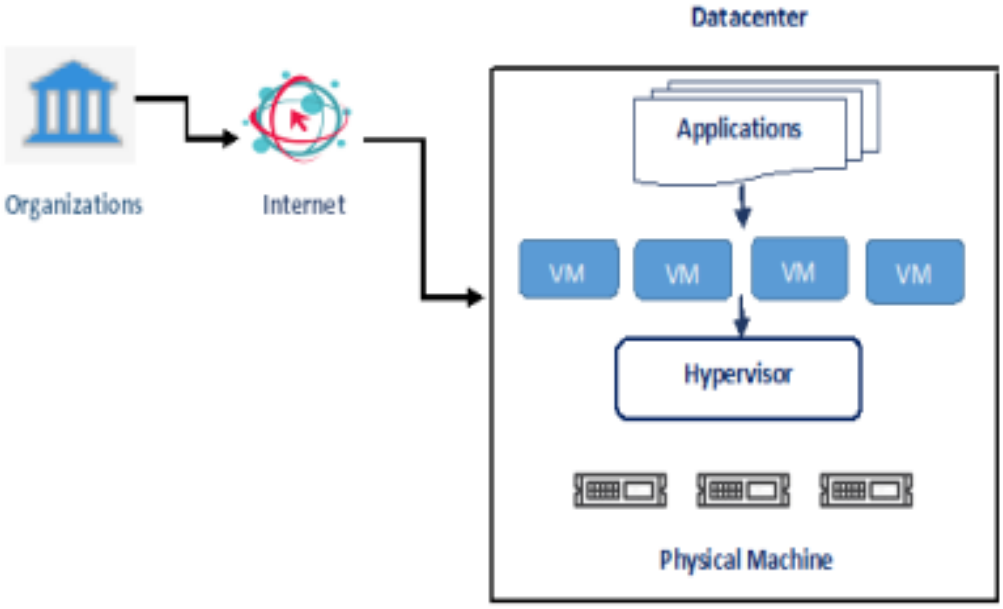

Figure 1: Resource Allocation in Load Balancing

The framework of IaaS load balancing is depicted in Figure 1. An organization sends the user's workloads, i.e., applications, to the DC over the Internet for processing [7]. Every DC contains many VMs that are installed on the top of each physical part of the hardware through virtualization. Hypervisors are software programs that construct VMs and distribute the hardware resources among them. VMs are logical machines that are capable of handling different applications [8].

An effective dynamic scheduling strategy is needed to improve IaaS performance by maximizing the use of DC and VMs, cutting down on load processing times and costs, and preventing servers from being overloaded or underutilized in an heterogeneous cloud platform.

2. RELATED WORK

In the past few years, many SBP and LB approaches have been proposed. This section describes some of the significant SBP and LB approaches related to this research study.

A fundamental static LB approach called round robin (RR) is applied circularly while distributing loads among several resources according to time quantum. Every load is given an equal amount of time to perform inside its designated time slots; if the time slice runs out before the load is finished, it must wait until the next time slice is available. Every VM receives a load under this policy to be processed. This approach has some shortcomings including its inability to function well in heterogeneous clouds due to the fact that each load varies in size and completion time, and the uneven distribution of workloads results in the overloading of certain VMs [9].

Equally Spread Current Execution (ESCE) is a dynamic approach; whenever a workload arrives, it locates the VM with the smallest current allocations and assigns the workload to that VM. In other words, ESCE distributes the workloads to the VM that has the fewest current loads; this cycle is resumed until all of the loads have been assigned, and each VM receives an equal amount of loads for execution in ESCE, which results in good resource utilization [10]. However, this approach is only effective in a homogeneous cloud; it is unable to balance workloads in heterogeneous clouds, or those with varying levels of loads. Because it doesn't examine the size of the VM, it simply checks the number of current loads in each VM [11].

A Throttled Load Balancing (TLB) strategy is an adaptive load balancing technique that distributes workloads between VMs according to their present capacity and accessibility. TLB uses a hashmap table and keeps track of all the available VMs and their current state, i.e., busy or available, in the hashmap. Whenever a workload arrives from the DC, the first VM that is available and checks whether its capacity fits the load size is chosen by TLB and assigned to a load for processing. The VM's status is then changed to busy; in this way, all the workloads are assigned to the proper VM. If no VM is currently available, the Data Centre Controller (DCC) queues the load and waits for the next available VM [12].

In contrast to the existing TLB method, which keeps only one component for keeping all the available and busy VMs, resulting in workload overhead during allocation, the new extended component-based TLB algorithm is composed of three VM components, which are 1. Reads all the available VMs 2. Keeps all the free VMs separately, and 3. Assigns the VM when the load arrives from the DCC. This algorithm has been examined in terms of DC processing time and response time compared to the current TLB algorithm using the CloudAnalyst simulation tool. The suggested approach reduces load overhead, but even though the VM has available space, only one load is ever assigned to it at any point in time, leaving the loads in the waiting state [13].

The novel Balanced Throttled LB Algorithm is demonstrated in this work, which routes all arriving workloads to all available VMs according to their status, i.e., busy or available, in a hash map table with the aim of balancing the workload among VMs. The suggested methods provide a faster load response time than alternative algorithms when compared to the current RR, active monitoring, and TLB strategies, according to an analysis conducted using the CloudAnalyst simulation tool [14].

The Throttled Modified Algorithm (TMA) modifies the current Throttled Load Balancing strategy by keeping both available and busy VMs separately using a different HashMap structure. Workload is assigned to specific VMs from the available HashMap table upon arrival; once the workload is allocated to a particular VM, the VM is shifted to the busy HashMap table. The VM is returned to an available VM after it has finished

processing the load. The TMA algorithm has been evaluated using the current TLB and RR techniques in the CLoudAnalyst simulation tool, and the results demonstrate that the TMA policy improves the load response time and DC processing time for each load. The restriction of TMA is that each VM can only have one load assigned to it at a time [15].

SBP is a method used by Service Broker for handling traffic between UserBase (UB) and DC. Service Brokering is a datacenter allocation policy that acts as a traffic router between UB and DC with the aim of routing numerous user workloads arriving from UB to DC, i.e., it decides which DC services the loads arriving from each user. SBP influences the overall performance of the IaaS cloud system [16].

The closest datacenter policy, also known as the service proximity-based service broker policy (SPB-SBP), assigns the incoming loads to the datacenter with the least network latency or proximity, i.e., selecting the DC with the least distance from the UB based on the region, since each UB is present in different regions and also each DC is present in different regions. If more DCs are present in the same region, then this policy performs the random selection of DCs. The load response time and overloading of some DCs will rise if the randomly chosen DC has more loads and a minimal hardware configuration [17].

Optimized Response Time service broker policy (ORT-SBP) assigns the incoming loads arriving from UB to the DC, which is likely to yield the shortest response time; ORT records the response time of the load that was previously serviced by the DC. ORT does not select the DCs that have not received any loads previously, which affects the balancing of loads among DCs. ORT is not suitable for a dynamic environment and also does not consider performance parameters like cost, resource utilization, and load processing time [18].

The dynamic reconfigure routing policy uses a closest datacenter policy for finding the nearest DC based on the distance from UB to DC for load distribution, and in addition to that, it allows dynamic scaling of VMs within each DC based on the demand of the users. It calculates the total load capacity requirements, and based on that, it dynamically extends or shrinks the number of VMs in each DC to give the best load processing time. When a DC fails or performs poorly, this policy dynamically migrates its loads to another DC [19].

The modified service broker policy is implemented in the CloudAnalyst simulation tool, which is used mainly for reducing response time and processing time. This policy follows the methods of laying eggs by the cuckoo and dumps them into an arbitrarily chosen nest and chooses the best nest with good quality eggs; likewise, this policy initially keeps loads in the randomly chosen DCs and finds the best DC that gives minimal response and processing time. It is examined against ORT, CDC, and RDL, and it gives the best overall response time than other policies [20].

The new service brokering policy removes the drawbacks of randomly selecting DCs by the Service proximity-based routing policy; instead, it chooses the DC by selecting the minimum makespan DC using the Ant Colony Optimization policy, where the ant visits all the DCs and chooses the efficient DC based on its capacity and makespan value and stores this information for feature prediction. This policy was experimented with using the CloudAnalyst simulation tool with the existing ORT, CDF, and RDWL policies, and the result shows that the proposed policy increases efficiency by reducing the response and average processing time of the load [21].

3. PROPOSED WORK

3.1 Method Overview

The main objective of the proposed study is to use application LB and service brokering techniques to dynamically route HTTP workloads to the appropriate DC and VM to handle workloads on the IaaS heterogeneous cloud computing platform. In this study, we have combined the proposed service brokering and LB approaches, such as the TALB algorithm and ECO-SBP, and evaluated their performance concerning RT, DCPT, cost, and ST.

3.2 Proposed Service Brokering Policy (ECO-SBP)

The proposed ECO-SBP is a dynamic service brokering policy that extends the existing closest DC policy, which always chooses the nearest DC based on its proximity or delay from the UserBase (UB), and overcomes the issues of random selection of DC by adding additional criteria to select the appropriate DC, such as its availability, current traffic, current capacity, current load, and makespan and also allocates the users loads in the order with respect to its severity value. For each load severity value assigned, ranging from 1 to 5, some loads should be completed within certain time limits; for such loads, low severity values are assigned, like one and two, and that load should be completed within a shorter time period, so it assigns a minimum target time and based on the target time, the load has been processed.

The ECO-SBP finds all the nearby DCs using the network delay value of each DC from the userbase, calculates the traffic by using its communication line, and chooses all the DCs with less traffic based on a threshold value. Now calculate the capacity and present load of the DC and choose the DC with a high capacity and a lower current number of allocations based on some threshold value. If more DCs are found, then choose the DC with a lower makespan value [22].

3.3 Throttled Adapted Load Balancing Strategy (TALB)

The proposed TALB policy extends the existing TLB policy by modifying the data structure, i.e., the way of storing the VM details in the system. TLB uses a HaspMap table for maintaining VMs, keeps all the available and busy VMs in a single HaspMap table, and stores the available and busy status of each VM in the VM status field. When the load arrives from the users, TLB searches the HaspMap table for available VMs by checking its status field. If the VMs are available and suitable for load based on their size, then the VM is allocated and moved on to the busy list. The limitations of this policy are that it can assign only one load to any VM at a particular point in time and also increases the searching time of the VM.

TALB overcomes the problem of TLB by using the TreeMap structure, which can be partitioned into many levels to maintain the busy VM and available VM separately; i.e., all the available VMs, i.e., VMs not having a single load for processing, are stored in the available partition list, and all the busy VMs, i.e., VMs having at least one load for execution, are stored in the busy list. All VMs are sorted according to their capacity and present number of allocations, i.e., the VMs with a high capacity and a low number of loads are stored at the top of both the available and busy partition lists. When a load arrives, the TALB policy selects the first available VM from the available partition list, and if that VM size is suitable for the load, then that VM is allocated and moved onto the busy VM list. If no VM is available, then TALB searches the busy VMs for load

allocation and assigns the load to the suitable VM; otherwise, the load has to wait till the appropriate VM is found [23].

3.4 Methodology

The capacity of each DC is the total capacity of all the VMs present in that DC is obtained by Eq. (1).

$$TVM_c = TVM\sum_{i=1}^{n} vCPU(i) + vST(i) + vRAM(i) + vBW(i) \quad (1)$$

In the above equation, TVMc represents the total capacity of all the VMs vCPU, vST, vRAM, vBW represents number of virtual processors, amount of storage capacity, amount of main memory capacity and available bandwidth configured for a particular VM. The total capacity of each DC is calculated in Eq. (2).

$$TDC_c = DC_{pm} * TotVM_c \quad (2)$$

Where, TDC_c represents capacity of the DC, represents the capacity of original physical machines and represents the total capacity of the all the VMs in that DC. Total utilization of DC is obtained in Eq. (3).

$$TDC_u = \sum_{i=1}^{n} TotVM_u \quad (3)$$

Where, TDC_u represents the capacity of the DC and n represents number of VMs and $TotVM_u$ represents utilization of each VM in a DC. The current capacity of each DC is attained in Eq. (4).

$$DC_{pl} = TDC_c - TDC_u \quad (4)$$

Where, DC_{pl} represents the present load of the DC and TDC_c represents the total capacity of the DC and TDC_u represents the represents utilization of a DC. Minimum makespan value is calculated by Eq. (5).

$$Min_{Makespan} = Min(Ext_{Time}(Li)), 1 \le i \le m \quad (5)$$

Where, $Min_{Makespan}$ is a lower bound of makespan that is the minimum time required by the DC to complete all loads. Li is the load and i represent load number. The response time of each DC is given in Eq. (6).

$$RTD_c = PTD_c + TD_l \quad (6)$$

Where, RTD_c refers to the response time of each DC, PTD_c refers to the processing time and TD_l refers to the transmission delay of the load. The DC processing time is computed in Eq. (7).

$$PTD_c = RTD_c - (NP_d + NS_d) \quad (7)$$

Where, PTD_c refers to processing time of DC, RTD_c refers to response time of the DC, and NP_d are network propagation delay and serialization delay respectively.

Cost of the VM for each load is calculated based on the usage of the VM per hour shown in Eq. (8).

$$TVM_c = VML_{hrs} * VM_c \quad (8)$$

Where, TVM_c refers to the total cost of the VM used by a particular load, VML_{hrs} refers to the number of utilization hours of a VM by the load and VM_c refers to the cost of the VM per hour. Overall average cost of the VM is calculated in Eq. (9).

$$VMCost_{avg} = 1/n * \sum_{i=1}^{n} VMCost(i) \quad (9)$$

Where, $VMCost_{avg}$ refers to the cost of all the loads in that VM, n refers to the number of loads in each VM. The total cost of executing loads can be obtained in Eq. (10).

$$Load_{tc} = TVM_c + VM_{pc} \quad (10)$$

Where, $Load_{tc}$ refers to the cost of each load, TVM_c refers to the cost of the VM of each load and VM_{pc} refers to the processing cost of the load by the VM.

3.5 Flow Diagram

Fig. 2 illustrates the Workflow of allocating the load to the suitable DC and VM by the ECO-SBP and TALB approaches, respectively. The arriving workloads at the nearby UB from various regions are forwarded to the DCC, which calculates the load severity value for prioritizing it based on the completion time and forwards it to the ECO-SBP for finding the best DC. Once the loads are sent to the proper DC, the DC sends the loads to the TALB load balancer for VM allocations.

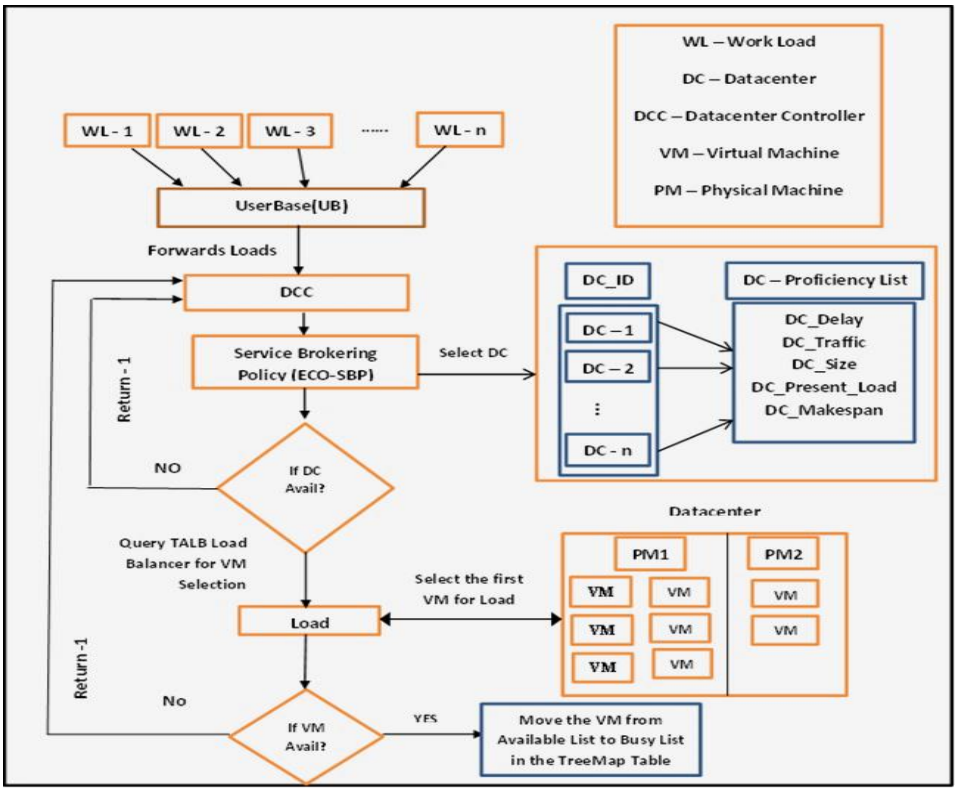

Figure 2: Block Diagram of ECO-SBP and TALB

3.6 ECO-SBP and TALB Algorithms

3.6.1 Algorithm - 1 //ECO-SBP

Input: Load Identifier (Load_ID), UserBase (UB), Region, DC identifier (DC_ID)

Output: DC Proficiency List (DC_PL), DC_ID

For all DC Do

Find DC_ND (Network Delay), DC_NT (Network Traffic), DC_Size (Size), DC_CL (Current Load), MakeSpan (DC_MS) of each DC

Create DC_PL (DC Proficiency List) using HashMap table

DC_Threshold<-"NULL", DC_PL<-DC_ID, DC_S, DC_CL, DC_NT, DC_ND, DC_MS, DC_Threshold

Update the DC_PL for each DC

Assign the threshold values for each DC based on the proficiency List

If (DC_CL > = 80) then

DC_Threshold<-"Max"// Overloaded DC

DC_S<-Busy

Else If (DC_CL >=50 && DC_CL <=79) then

DC_Threshold<-"Norm"

DC_S<-Busy //assign normal threshold End If

Else If (DC_CL<50) then DC_Threshold<-"Min"

DC_S<-busy

Else If (DC_CL ==0) then DC_S<-Available

// set all the DCs status value as available

End IF

Sort the DC_PL for each DC based on the threshold

Update DC_PL

Return DC_PL

End for

For each load Do

Assign DC_SV//Assign severity value for each Load according to its completion time

Calculate Target Time (Load_TT) and load Size (Load_S)

Choose the load with low severity value

For each DC in DC_PL DO

If (DC_Status==" Available") then

If (Load_S<DC_S) then

Allocate DC_ID<-Load_ID

```
            Return DC_ID // assign loads to that DC
DC_S<-“Busy”, Update DC_CL, Update     DC_Threshold, Update DC_PL
          End IF
       Else // no DC is idle or No DC is matched          then check all the busy DC
       If (DC_Threshold ==”Min”)   then
          If (Load_S<DC_S) then
Allocate DC_ID<-Load_ID, Update     DC_CL, Update DC_Threshold, Update DC_PL
            Return DC_ID // assign loads to DC
           End If
       If (DC_Threshold ==”Norm”)   then
          If (Load_S<DC_S) then
Allocate DC_ID<-Load_ID, Update   C_CL, Update DC_Threshold
Return DC_ID // assign loads to that DC Update DC_PL
          End If
        Else
Return -1 // Load has to wait until the suitable DC is found
       End If End If
End for End For
```

3.6.2 Algorithm - 2 //TALB

```
For each VM in the selected DC
Assign VM<-Status = 1 // set 1 to all the available VMs
Create TreeMap Struct//create TreeMap structure for keeping the VMs
TreeMap Part_1, Part_2 //Partition TreeMap in to 2 levels
 If (VM_Status==1) then
    TreeMap.Part_1<-VM_ID
 Else
     TreeMap.Part_2<-VM_ID
 End If
//Load Allocation
 If (UB_Load == “YES”)
Search Treemaps_Part_1//find available VMs
       If (VM_Capacity >Load_S) then
          Assign VM_ID<-Load_ID
       Else
```

Search Treemaps_ Part_2//find busy VMs

If (VM_Capacity >Load_S) then

Assign VM_ID<-Load_ID

Else Wait (Load)//Load has to wait

End If End If End If

End For

4. RESULTS AND DISCUSSION

4.1 Simulation Setup

The experiment for this study was conducted using a simulation tool called CloudAnalyst for assessing the performance of ECO-SBP and TALB and compared them with three service brokering policies such as SPB-SBP, ORT-SBP, and RDL-SBP, and three Load balancing strategies such as RR, ESCE, and TLB, which are currently used by the CloudAnalyst tool, with qualitative metrics such as RT, DCPT, ST, and Cost.

CloudAnalyst is a widely used simulation tool written in the Java programming language for IaaS cloud computing that inspects large-scale cloud applications that are distributed geographically in all locations using DCs and VMs. It provides Graphical User Interface suitable for executing both static and dynamic LB and SBP policies and also supports the scheduling of several heterogeneous resources for workload processing [10].

As illustrated in Fig. 3, we have used 5 DCs with 2 physical machines in each DC to demonstrate the tests. Under these DCs, 60 VMs with various physical specifications have been built, and 12 UBs with varied loads of different sizes have been tested.

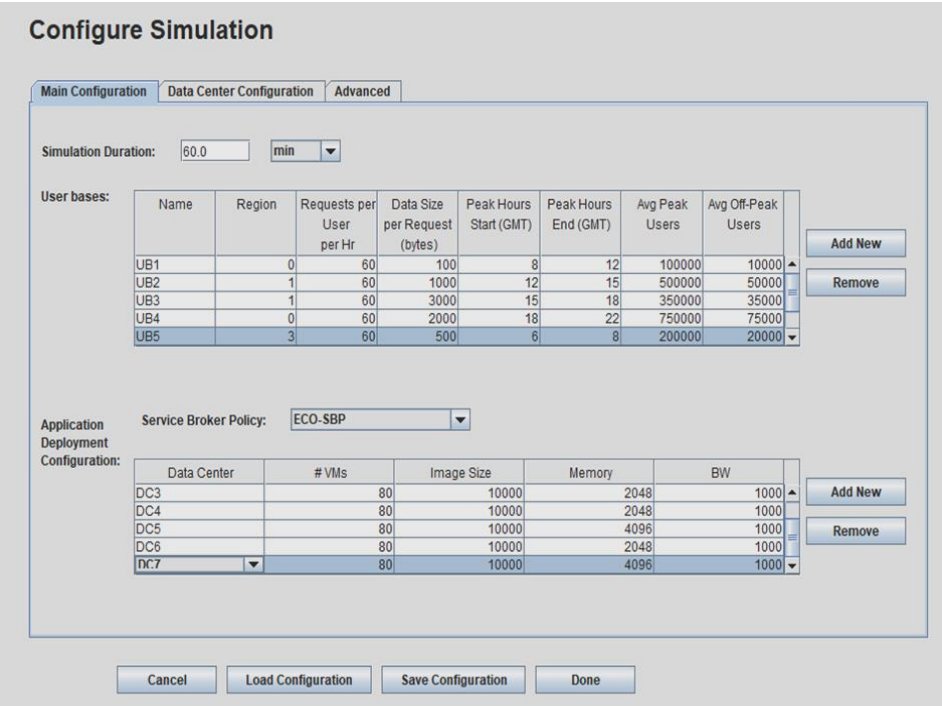

Figure 3: UserBase and Datacenter Configurations

Fig. 4 shows the inclusion of the proposed ECO-SBP and TALB policies in the CloudAnalyst simulation tool, and they are simulated in a large-scale heterogeneous IaaS cloud environment.

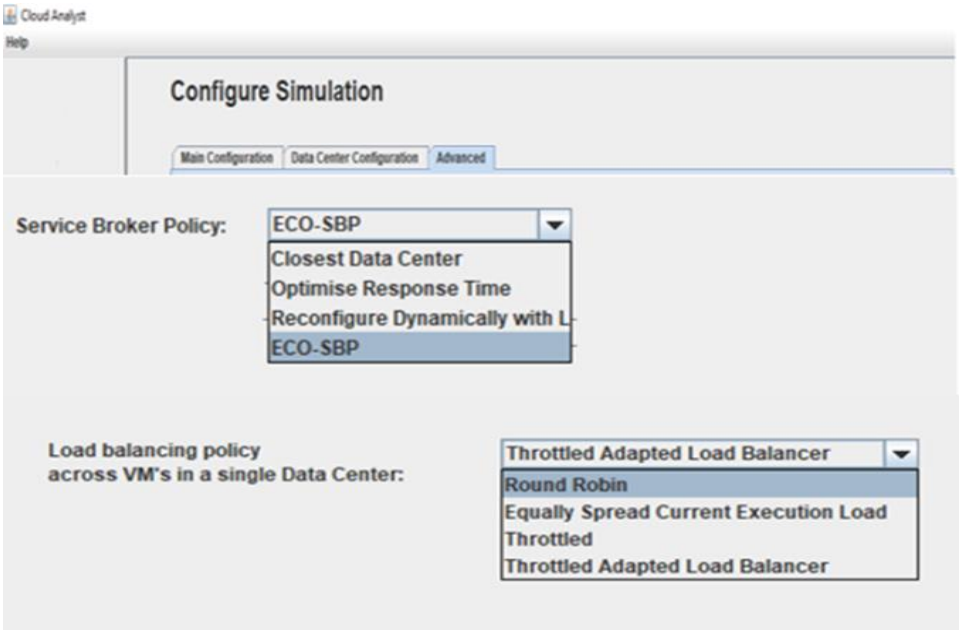

Figure 4: ECO-SBP and TALB in CloudAnalyst Tool

4.2 Experimental Results and Analysis

Fig. 5 shows the results after simulating the ECO-SBP and TALB in CloudAnalyst. The line from the UB to the DC shows that the loads of the UB were allocated to that DC when the loads arrived at each UB, and the RT of each UB was measured with the average, minimum and maximum RT of each load from various UBs displayed within the boxes in each UB.

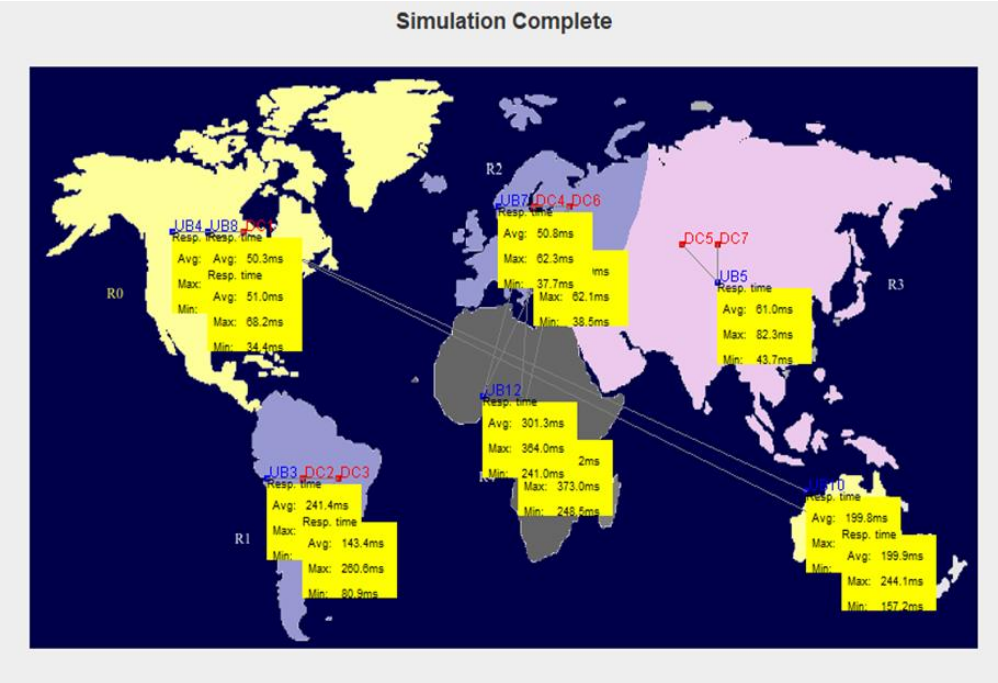

Figure 5: Response Time of Each UB in CloudAnalyst Tool

The RT, DCPT, Cost, and ST for each brokering policy including SPB-SBP, ORT-SBP, RDL-SBP, and ECO-SBP as well as each load balancing strategy including RR, ESCE, TLB, and TALB are shown in Figs. 7, 8, 9, and 10. It shows that SPB-SBP with all load balancing strategies yields a longer response time than the other three policies.

In contrast to other service brokering policies with other load balancing policies, the proposed ECO-SBP policy with the TALB load balancing strategy yields a minimum response time, minimal datacenter processing time , reduction in cost given in dollar($) and takes less searching time for finding the best VM than the other three policies.

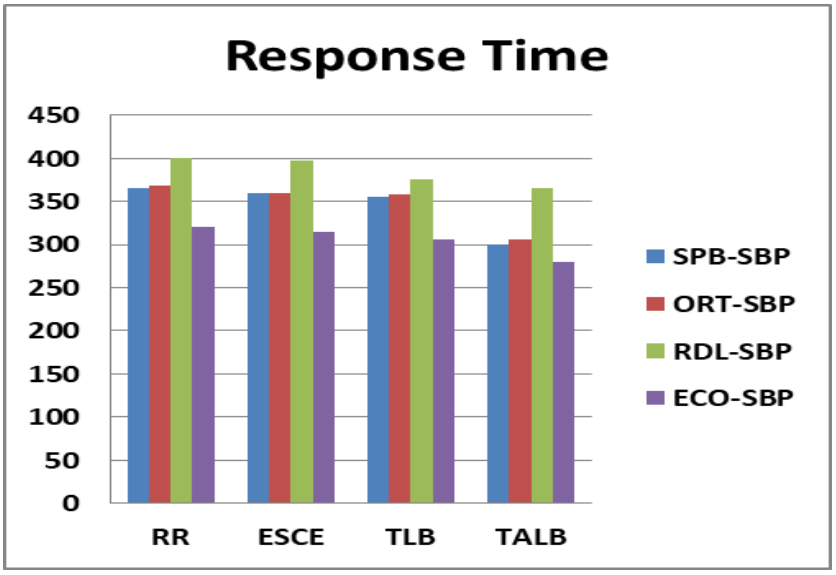

Figure 6: Response Time of SBP with LB

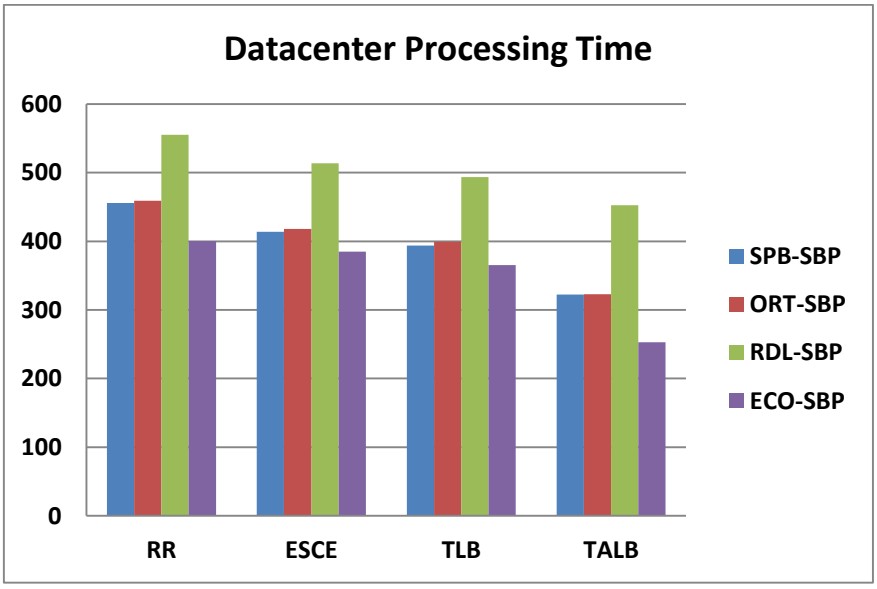

Figure 7: Datacenter Processing Time of SBP with LB

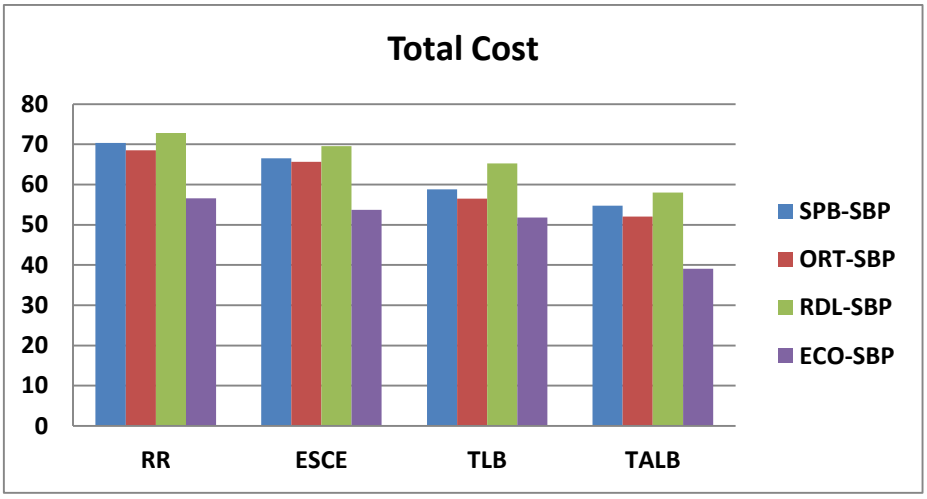

Figure 8: Total VM and DC Processing Cost of SBP with LB

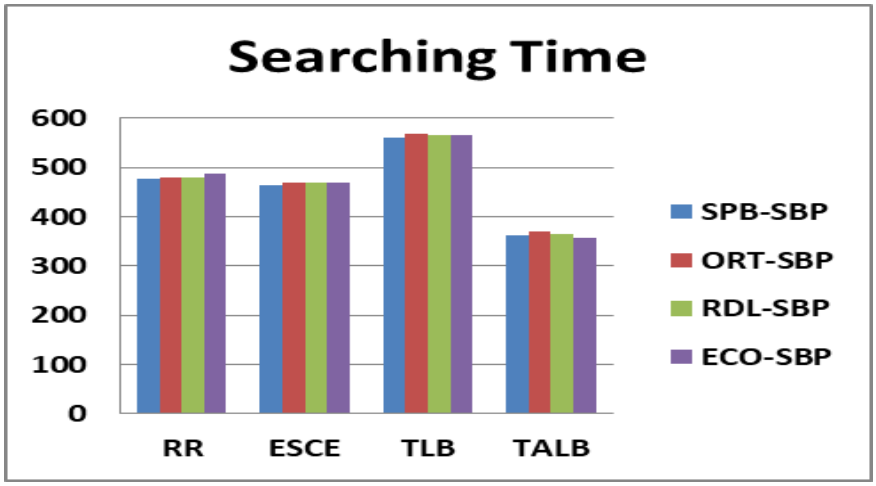

Figure 9: Searching Time of SBP with LB

The main aim of ECO-SBP and TALB is to select the best DC and VM for each load in order to reduce each load's ST, RT, and DCPT, save costs, and achieve efficient resource utilization with the given available resources.

5. CONCLUSION

Service Brokering and Load Balancing techniques are the most vital resource scheduling techniques in IaaS cloud computing systems for the efficient allocation of datacenter and virtual machines respectively. This paper mainly focuses on examining the proposed dynamic ECO-SBP service brokering policy and TALB Load Balancing algorithm with some of the existing significant policies using CloudAnalyst simulation tools and the comparisons done with respect to the performance metrics such as loads response and processing time, cost of executing the loads and searching time of resource allocation. The ECO-SBP policy process the loads in the order of their severity values and chooses the DC with respect to its suitability of the load such as its capacity, present load, Markesan and TALB algorithm finds the best VM with less time for the purpose of minimizing the searching time of each VM, achieves minimum response time, loads processing time, and cost.

References

1) S. Sefati, M. Mousavinasab, and R. Zareh Farkhady, “Load balancing in cloud computing environment using the grey wolf optimization algorithm based on the reliability: Performance evaluation,” *The Journal of Supercomputing*, vol. 78, no. 1, pp. 18–42, May 2021.

2) S. Talwani *et al.*, “Machine-learning-based approach for virtual machine allocation and migration,” *Electronics*, vol. 11, no. 19, p. 3249, Oct. 2022. doi:10.3390/electronics11193249

3) 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*, vol. 9, no. 4, pp. 1558–1568, Aug. 2020. doi:10.11591/eei.v9i4.2011

4) S. R. Gundu, C. A. Panem, and A. Thimmapuram, “Real-time cloud-based load balance algorithms and an analysis,” *SN Computer Science*, vol. 1, no. 4, May 2020.

5) M. P. Yadav, N. Pal, D. K. Yadav, “Workload prediction over cloud server using time series data”, *Proceedings of the 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence),* Jan 28 2021, pp. 267-272, IEEE.

6) R. Rajak, A. Choudhary, and M. Sajid, “Load balancing techniques in Cloud platform: A systematic study,” *International Journal of Experimental Research and Review*, vol. 30, pp. 15–24, Apr. 2023.

7) Challa M, Sudha D. An efficient approach for minimization of energy and makespan in cloud computing. Annals of the Romanian Society for Cell Biology. 2021 Jun 3;25(6):7422-30.

8) D. A. Shafiq, N. Z. Jhanjhi, and A. Abdullah, “Load balancing techniques in cloud computing environment: A Review,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 3910–3933, Jul. 2022. doi:10.1016/j.jksuci.2021.02.007

9) Y. Lohumi et al., “Load balancing in cloud environment: A state-of-the-art review,” *IEEE Access,* vol. 11, pp. 134517–134530, 2023.

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.

11) A. Singh and R. Kumar, “Performance evaluation of load balancing algorithms using cloud analyst,” *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Jan. 2020.

12) El Karadawy, A.I., Mawgoud, A.A. and Rady, H.M., 2020, February. An empirical analysis on load balancing and service broker techniques using cloud analyst simulator. In 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE) (pp. 27-32). IEEE.

13) A. Belgacem, K. Beghdad-Bey, H. Nacer, and S. Bouznad, “Efficient Dynamic Resource Allocation Method for Cloud Computing Environment,” *Cluster Computing*, vol. 23, no. 4, pp. 2871–2889, Feb. 2020.

14) S. Y. Mohamed, M. H. Taha, H. N. Elmahdy, and H. Harb, “A proposed load balancing algorithm over cloud computing (balanced throttled),” *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 10, no. 2, pp. 28–33, Jul. 2021. doi:10.35940/ijrte.b6101.0710221

15) Elnagar, N.G., Elkabbany, G.F., Al-Awamry, A.A. and Abdelhalim, M.B., 2022. Simulation and performance assessment of a modified throttled load balancing algorithm in cloud computing environment. *International Journal of Electrical & Computer Engineering,* 12(2), pp.2087-2096.

16) Shahid MA, Alam MM, Su’ud MM. Performance evaluation of load-balancing algorithms with different service broker policies for cloud computing. *Applied Sciences,* 2023 Jan 26;13(3):1586.

17) J. Bisht and V. V. Subrahmanyam, “Improvising service broker policies in Fog Integrated Cloud Environment,” *International Journal of Communication Networks and Distributed Systems,* vol. 28, no. 5, p. 534, 2022. doi:10.1504/ijcnds.2022.125360

18) A. H. Zamri, N. S. Pakhrudin, S. Saaidin, and M. Kassim, “Equally spread current execution load modelling with optimize response time brokerage policy for cloud computing,” *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 2, 2023. doi:10.14569/ijacsa.2023.0140257

19) 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

20) Nazir, S. *et al.* (2018) ‘Cuckoo optimization algorithm based job scheduling using cloud and Fog computing in smart grid’, *Advances in Intelligent Networking and Collaborative Systems*, pp. 34–46. doi:10.1007/978-3-319-98557-2_4.

21) Raghuwanshi, S. and Kapoor, S., 2018. The new service brokering policy for cloud computing based on optimization techniques. *Int. J. Eng. Tech.,* 4, pp.481-488

22) Subramanian, S. and Natarajan, P. (2024) ‘Efficient criticality oriented service brokering policy in cloud datacenters’, *International Journal of Electrical and Computer Engineering (IJECE)*, 14(2), p. 2024. doi:10.11591/ijece.v14i2.pp2024-2034.

23) S. Shanmugapriya and N. Priya, “The proposed it-talb in infrastructure as a service cloud,” *Indian Journal Of Science And Technology*, vol. 17, no. 16, pp. 1654–1662, Apr. 2024. doi:10.17485/ijst/v17i16.3274.