AN OPTIMUM SERVICE BROKER POLICY FOR SELECTING DATA CENTER IN CLOUDANALYST

T. Menakadevi[1], N. Devakirubai[2]

[1]Professor, Department of Electronics and Communication, Adhiyaman College of Engineering, Hosur, India
[2]Asst. Professor, Department of Computer Science and Engineering, Jayam College of Engineering and Technology, Dharmapuri, India.

Abstract

Routing request to proper destination is a key consideration in distribution networks. Service broker policy is the essential logic, which functions with this intention. Choosing the right data center is the promising task for service broker policy. The simulation tool CloudAnalyst contains few service broker policies with which request is routed to the appropriate destination data center. Random selection of data center in these service broker policies deteriorates the efficiency of response time parameter. In this paper, a new service broker policy is proposed which optimally routes the request to a data center and overcomes the issues raised by random selection of a data center.

Keywords: *Cloud Computing; Cloudanalyst; Service Broker Policy; Data Center; Service Proximity Service Broker; Best Response Time Service Broker.*

----------***----------

1. INTRODUCTION

The Clouds are high configured infrastructure that delivers platform, infrastructure and software as a service. Cloud applications have two significant parameters for cogitation: response time and processing time. It is a tedious task for a researcher to scale the parameters in the real cloud platform due to the geographical distribution of cloud servers and its users. CloudAnalyst is a simulation application that performs the cloud platform functionalities like service brokering and load balancing [1].

CloudAnalyst is the extension of CloudSim. Since it is a GUI tool, it helps the user to focus more on simulation rather than on programming complexities. It generates reports on the parameters like, response time, processing time, cost of data transfer and Virtual Machine (VM) cost.

The traffic routing between the data centers and the user bases are accomplished by service brokers. Service brokering sends the request of users to the exact destination and balances the workload among the nodes in Cloud [2]. CloudAnalyst has three service broker policies, all with their own advantages and disadvantages. In all the service brokers, data centers are selected in a random manner.

Random selection of datacenters does not always identifies the right datacenter, it selects datacenters which is not cost effective. To overcome the issue raised by random selection of a data center, a new service broker policy is framed in this paper. Also, while selecting a destination data center, latency, and bandwidth are considered in the proposed algorithm. In all the other service broker policies, only latency is considered to select a destination data center.

Further, the paper is arranged as: Section 2 reviews about the related work. Section 3 details about the existing service broker policies. Section 4 illustrates the proposed service broker policy. Section 5 analyses the simulation results. Section 6 concludes and states the future work.

2. RELATED WORK

T-broker, a trust-aware service brokering system is presented in [3]. T-broker have to trust models: hybrid and adaptive, to calculate the overall trust degree of service resources. This work matched the multiple cloud services with the user requests. A service broker policy based on Round-Round algorithm with priority is proposed in [4]. They modified the closest data center service broker policy in a way, when there are two or more data centers in the proximity list, then based on the usage of each data center, a priority number is assigned to it. This priority list is used when selecting a data center with Round Robin scheduling.

A service broker policy named as dynamically reconfigure peak time policy is framed in [5]. This policy senses the availability of data centers in the nearest region. If the data center in the nearest region is busy with handling some other request, then the user request is redirected to another data center in the next nearest region which is in off peak time. The work attempted to reduce the data center processing time and cost by sharing the load to other data center in off peak time.

Table-1: A Comparison on the Existing Service Broker Policies.

S. No	Existing Service Broker Policy	Mechanism Used	Objective
1	Service proximity based service brokering [2]	Closer datacenter is selected using latency information.	Reducing response time, datacenter processing time, data transfer and VM cost.
2	Performance optimized based routing [2]	The closer and quickest datacenter is selected.	Reducing response time, datacenter processing time, data transfer and VM cost.
3	Dynamically reconfigurable routing [2]	The number of VMs are increased or decreased based on requirement.	Reducing response time, datacenter processing time, data transfer and VM cost.
4	Trust aware service brokering system [3]	Light-weight feedback mechanism.	Matching multiple cloud services to user requests.
5	Priority based Round Robin service brokering [4]	Round Robin algorithm	Efficient resource utilization.
6	Cost based datacenter selection policy [5]	User requests were redirected to next neighboring datacenter.	Reducing the datacenter workload.
7	Cost-effective datacenter selection [6]	Most cost effective datacenter was selected.	Reducing VM and datacenter cost.
8	Efficient datacenter selection [7]	Weighted round robin algorithm.	Reducing datacenter request service timing
9	Efficient service brokering [8]	Datacenter is selected based on its resource handling capacity	Reducing response time, datacenter processing time.
10	Future load aware service broker policy [11]	Datacenter load is calculated with genetically weight optimized Jordan neural networks.	Reducing datacenter processing time.

The service proximity based routing is modified in [6], by selecting the cost effective data center within the same region. The total virtual machine cost in CloudAnalyst is considered as the main parameter. Service proximity based routing is implemented in another way in [7]. When there is more than one data center within the earliest region, then selection of the data center is based on the processing capacity of each data center.

The service proximity based routing's random selection strategy is revised in [8]. Since the data center with many numbers of physical machines can handle many numbers of requests, instead of selecting random data center, the multiple data center in the earliest data region is selected based on the number of physical hardware in each data center.

The three service brokers are measured up in [9] and [10]. The comparison is based on the parameters: average response time, average data center processing time and total cost. Optimize response time is concluded as the prime choice among all the three polices [9]. A near future load of a data center as a parameter is considered in [11] while routing a request to a data center. The future load of a data center is predicted using genetically weight optimized Jordan neural network.

Cloud simulators, Cloudsim and CloudAnalyst are compared and analysed in [12]. They reviewed the service broker policies, their issues and the available solutions.
In [13], the optimized response time service broker policy is evaluated with the Throttled load balancing algorithm. The analysis was done with different experimental setups and finally concluded that the Throttled load balancing algorithm produces required efficiency in results when compared with other load balancing algorithms. The various Service Broker policies discussed are compared in Table-1.

The predecessor of CloudAnalyst is CloudSim, which is poor in presentation and graphical outputs. CloudAnalyst differs from CloudSim, in delivering the graphical results, separating the simulation environment from programming environment. With this quality, the CloudAnalyst is able to produce output for different parameters or for the same set of parameters.

The introduction about CloudAnalyst is completely dealt in [1] and discussed about the features, design of the simulator and illustrated the optimal configurations and load balancing algorithms about service brokers. In [2], the main objectives, features and design of CloudAnalyst are dealt. The various case studies proved that this tool best suits for cloud computing environments.

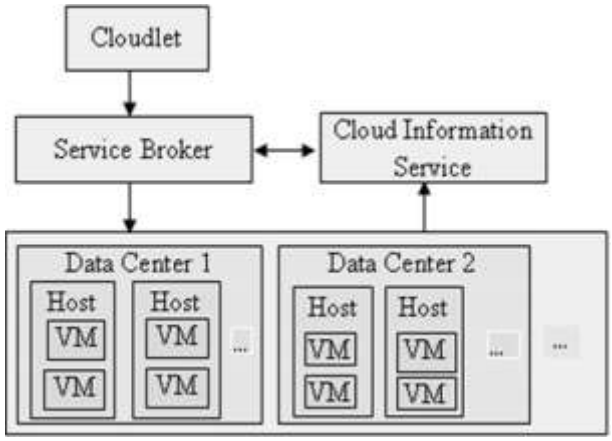

Fig-1: CloudAnalyst entities and their interaction

CloudAnalyst environment has three entities: Cloud information service, service broker and data center. The interactions between these entities are depicted in Fig-1. Cloud information service is a kind of registry that holds the information about data centers. There can be any number of data centers in the cloud environment and each data center need to be registered with a cloud information service.

As depicted in Fig.-2, Service broker communicates with cloud information service and retrieves information about data centers. The service broker policy routes the user base's request to the suitable data center. Data center controller and VM load balancer, involves when the user request reached a particular data center for processing. VM load balancer helps to distribute the workload among the available VMs.

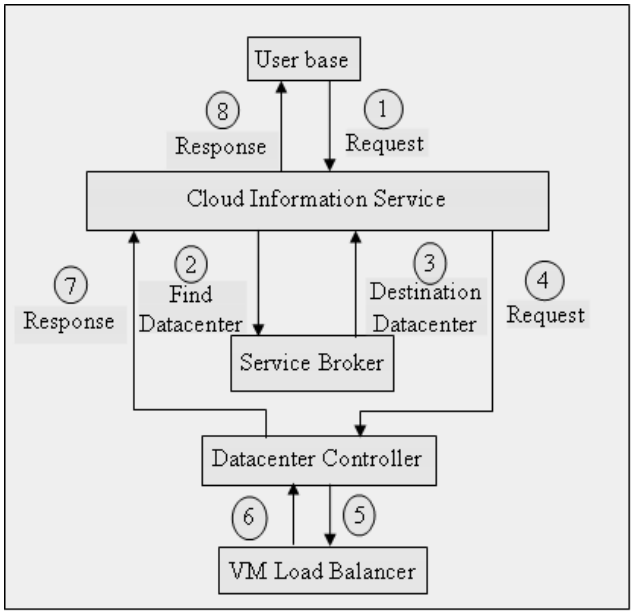

Fig-2: Request Routing Through Service Broker

Each data center comprises of physical machines, which in turn have processors, storage devices, memory and internal bandwidth. A data center will have a number of hosts and each host will be with different hardware configurations. Host contains many virtual machines. The users of cloud were grouped in user base. The user base can contain a single user or number of users [1].

3. SERVICE BROKER POLICY IN CLOUDANALYST

The six continents in the world are considered as six regions in CloudAnalyst. The user bases and data centers are geographically scattered over the six regions [1]. Request from a user base need to be routed to a data center, where it can get serviced. This process decides the efficiency in terms of response time, data center processing time and cost. Service broker policy plays an important role in achieving these parameters with efficient values. There are three service broker policies involved in CloudAnalyst: closest data center policy, optimize response time policy and dynamically reconfigurable routing with load balancing.

The first policy, Closest data center policy is service proximity based routing algorithm [2]. As its name adverts, the earliest data center is chosen for servicing the request. The proximity list of data centers is prepared in terms of least network latency. When there is more than one closest data center, then from the proximity list a data center is chosen randomly. The optimize response time policy is the performance optimized based routing and an extension of closest data center policy [2]. Initially, the closest data center is detected. If the response time of the closest data center starts degrading, then the data center with better response time at that particular time is searched and it is tagged as quickest data center.

If the closest data center is the quickest data center, it will be selected as the destination data center. If the closest data center and the quickest data center is not the same, then the selection of a data center among quickest and closest will be done randomly by balanced chance to both of the datacenters. The third service broker policy is the dynamically reconfigurable routing with load balancing, which is also the extension of closest data center policy and works with same routing logic [2]. Additionally, this service broker policy is assigned with the job of balancing the workload by increasing or decreasing the number of virtual machines in the data center.

4. PROPOSED OPTIMUM ROUTING SERVICE BROKER POLICY

The proposed optimum routing service broker policy adapts the optimize response time policy and revise the practice of selecting the data center randomly. The random selection of datacenter leads to the threat of selecting a data center with higher cost, higher response time, and higher processing time [4]. Also, there is a possibility of unbalancing the load distribution among data centers. In CloudAnalyst, the latency and bandwidth between regions are given as a parameter set, which can be represented in two dimensional form, latency (L) and bandwidth (B). Delay in time between the request from the user and the response by a service provider is latency, which is denoted in milliseconds. The amount of data transmitted during a second is bandwidth, which is usually denoted in bits per second.

Regions (Destination)

Region (Source)	0	1	2	3	4	5
0	$d_{0,0}$	$d_{0,1}$	$d_{0,2}$	$d_{0,}$	$d_{0,}$	$d_{0,}$
1	$d_{1,0}$	$d_{1,1}$	$d_{1,2}$	$d_{1,}$	$d_{1,}$	$d_{1,}$
2	$d_{2,0}$	$d_{2,1}$	$d_{2,2}$	$d_{2,}$	$d_{2,}$	$d_{2,}$
3	$d_{3,0}$	$d_{3,1}$	$d_{3,2}$	$d_{3,}$	$d_{3,}$	$d_{3,}$
4	$d_{4,0}$	$d_{4,1}$	$d_{4,2}$	$d_{4,}$	$d_{4,}$	$d_{4,}$
5	$d_{5,0}$	$d_{5,1}$	$d_{5,2}$	$d_{5,}$	$d_{5,}$	$d_{5,}$

Latency (L)

Regions (Destination)

Region (Source)	0	1	2	3	4	5
0	$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,}$	$b_{0,}$	$b_{0,}$
1	$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,}$	$b_{1,}$	$b_{1,}$
2	$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,}$	$b_{2,}$	$b_{2,}$
3	$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,}$	$b_{3,}$	$b_{3,}$
4	$b_{4,0}$	$b_{4,1}$	$b_{4,2}$	$b_{4,}$	$b_{4,}$	$b_{4,}$
5	$b_{5,0}$	$b_{5,1}$	$b_{5,2}$	$b_{5,}$	$b_{5,}$	$b_{5,}$

Bandwidth (B)

Algorithm 1
Proposed optimum routing policy

Input: Requesting user base
Output: Destination data center
Procedure getDestination(requesting user base)

```
DCList ← DataCenterIndex.get(region)
if DCList is not Null then
closestDC ← closestDataCenterPolicy(DCList)
endif
LeastEstResTime ← maximum value
for all DataCenters → DC do
LastRecResTime ← InternetCharacteristics.
NWdelay ←InternetCharacteristics.getTotalDelay()
if (LastRecResTime=Null) then
currEstResTime ← NWdelay
else
currEstResTime ← LastRecResTime+NWdelay
endif
if (currEstResTime< LeastEstResTime) then
LeastEstResTime ← currEstResTime
quickDC ← DC
endif
endfor
if (closestDC=quickDC) then
destination ← closestDC
else
min_lt_index←internetCharacteristics.getLowLt(
requestorRegion, latencyMatrix)
max_bw_index←internetCharacteristics.getHighB
w( requestorRegion, bwMatrix)
if (min_lt_index=max_bw_index) then
destination ←min_lt_index
else
resTimeLt←estResponseTime(requestorRegion,
min_lt_index)
resTimeBw←estResponseTime(requestorRegion,
max_bw_index)
if (resTimeLt< resTimeBw) then
destination ←min_lt_index
else
destination ←max_bw_index
endif
endif
endif
```

CloudAnalyst considers latency as the parameter to choose a destination datacenter, whereas the proposed optimum routing service broker policy choose the destination data center, by considering both latency and bandwidth. Algorithm 1 is used to implement the proposed policy.

Each source, destination pair of regions have the latency and bandwidth values in any of the four combinations like, low latency and high bandwidth (Fig-3), high latency and high bandwidth (Fig-4), high latency and low bandwidth (Fig-5) and low latency and low bandwidth (Fig-6).

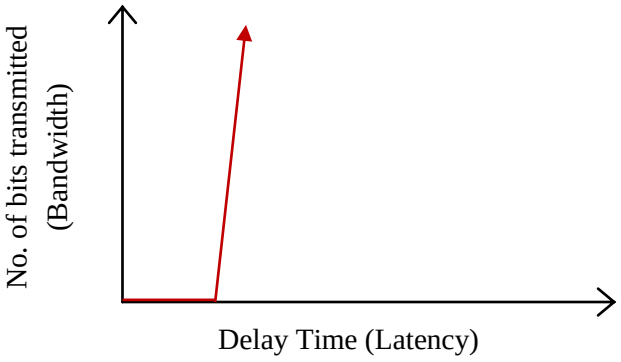

Fig-3: Low latency and high bandwidth

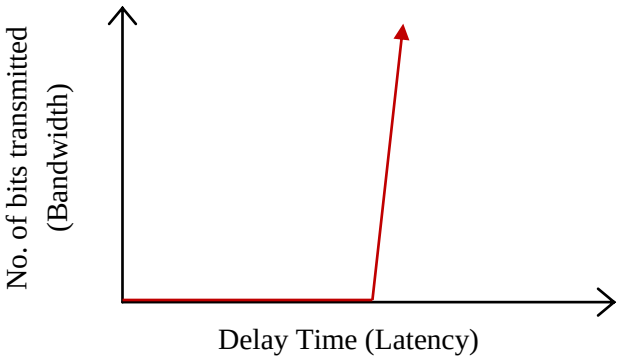

Fig-4: High latency and high bandwidth

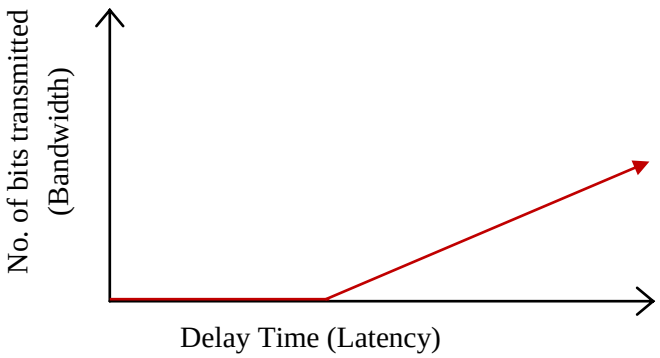

Fig-5: High latency and low bandwidth

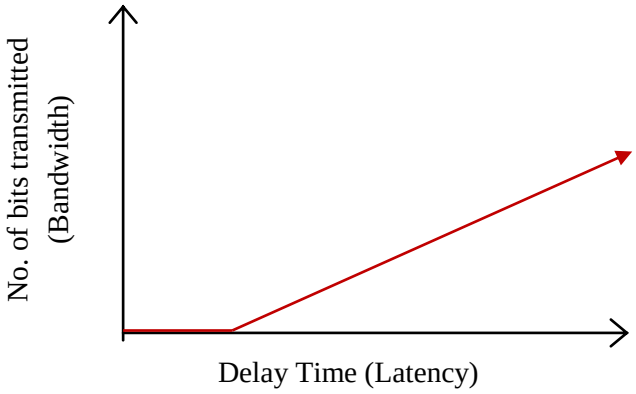

Fig-6: Low latency and low bandwidth

By observing Fig-3 to Fig-6, the source and destination pair of regions with low latency and high bandwidth is highly desirable (Fig-3). Our proposed policy finds the minimum latency node from the requestor region s,

$$\boldsymbol{d_{s,j}} = \min\{\boldsymbol{D_j}\}$$

Where j ranges from 0 to 6 and finds maximum bandwidth node from requestor region s,

$$\boldsymbol{b_{s,k}} = \max\{\boldsymbol{B_k}\}$$

Where k ranges from 0 to 6. If j=k then, j is choosen as the destination region as it has the low latency and high bandwidth. When j≠k, then estimate the response time of requestor region s to destination region j and requestor region s to destination region k. If response time of the pair (s,j) is less than (s,k), then j is choosen as destination region or else k is choosen as destination. The response time is estimated as

$$latency + (\boldsymbol{cloudlet\ data\ size/Bandwidth}).$$

5. RESULTS AND DISCUSSION

CloudAnalyst is used to assess the proposed optimum routing policy. It is implemented with a graphical user interface, which helps users to experiment quickly. Fig-7 depicts the visualized output of the simulation, through which the data center and user base distribution among the six regions can be seen. Also, it depicts the communication between a particular user base and data center, along with the response time. The 'configure simulation' screen helps to set the parameters related to user base, data center and grouping factors.

The 'define internet characteristics' screen is used to set the parameters related to latencies and bandwidths. These parameters can be executed with three different service broker policies and with the proposed service broker policy. Three load balancing policies are incorporated in this tool: Round robin, equally spread current execution load and Throttled.

The configuration screens of this tool hold the complete list of all required cloud environment parameters. The initial configuration parameter setup is listed in Table-2. The parameter setups can be saved and reloaded for further simulation. The proposed optimum routing service broker policy is analyzed with different set of parameters.

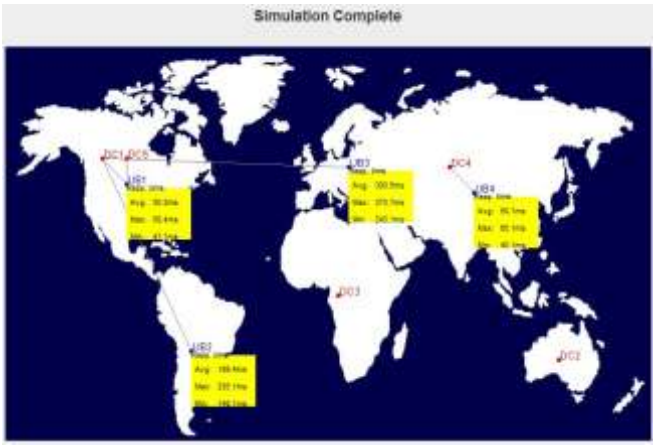

Fig-7: Simulation Screen

Table-2: Initial Configuration Parameters

SNo	Parameters	Values
User base Configuration		
1	Name	UB1
2	Region	2
3	Requests/User/Hr.	60 No.s
4	Data size/Request	100 bytes
5	Peak hours start	3 (GMT)
6	Peak hours end	9 (GMT)
7	Avg. peak users	1000 No.s
8	Avg. off peak users	100 No.s
Application Deployment Configuration		
1	Data Center	DC1
2	No. of VMs	5 No.s
3	Image Size	10000 bytes
4	Memory	512 MB
5	Bandwidth	1000 Mbps
Data Center Configuration		
1	Name	DC1
2	Region	0

3	Architecture	X86
4	OS	Linux
5	VMM	Xen
6	Cost/VM	$0.1
7	Memory Cost	$0.05
8	Storage Cost	$0.1
9	Data Transfer Cost	$0.1
10	Physical HW units	2 (No.s)
Other Configurations		
1	User grouping factor	10 No.s
2	Request grouping factor	10 No.s
3	Executable instruction length / Request	100 bytes

5.1 Case 1: Multiple User Base and Data centers

Table-3 shows the parameter settings for the multiple user base and data centers. In this case, there are four user bases from different regions. Five data centers are configured in different regions and with different number of virtual machines. In CloudAnalyst, user base is a term coined to represent the group of users.

Table-3: Case 1 Configuration Parameters

SNo	Parameters	Values
User base Configuration		
1	Name	UB1, UB2, UB3,UB4
2	Region	0,1,2,3
Application Deployment Configuration		
1	Data Center	DC1,DC2,DC3,DC4,DC5
2	No. of VMs	1,2,3,4,5 (No.s)
Data Center Configuration		
1	Name	DC1,DC2,DC3,DC4,DC5
2	Region	0,5,4,3,0
3	Physical HW units	1,1,1,1,1 (No.s)

Table-4: Case 1 Results Comparison Table

Service Broker Policy	Overall Response Time (ms)
Closest datacenter	379.94
Reconfigure dynamically with load balancing	379.70
Optimize response time	378.06
Proposed optimum routing policy	149.78

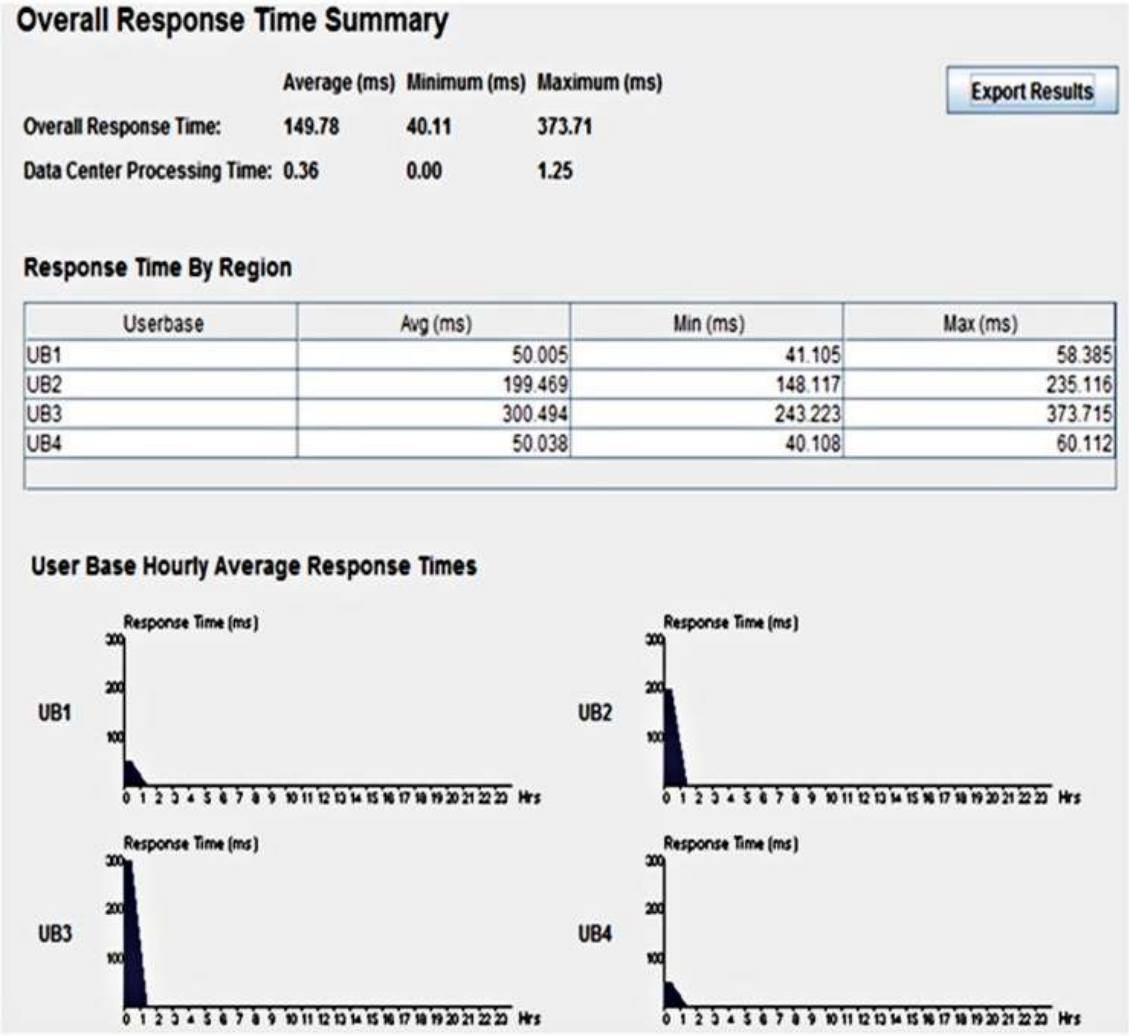

Fig-8: Case 1 Simulation Results

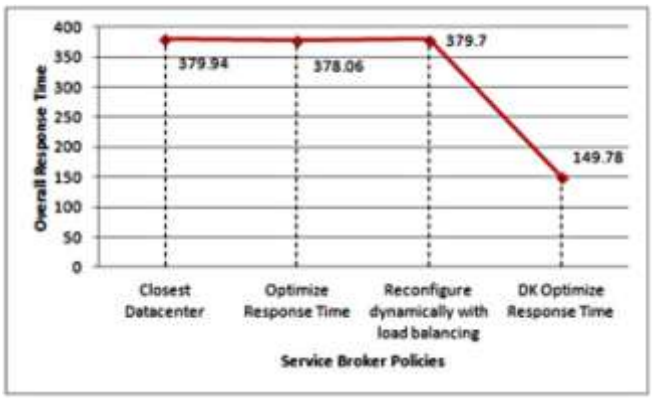

Fig-9: Case 1 Overall response time comparison

Fig-8 is the simulation results for Table-3 parameters. The value found by our proposed policy is compared in Table-4 and the same was visualized in Fig-9. The proposed service broker policy shows 60% of betterment in response time compared to other three service broker policies.

5.2 Case 2: Modified Grouping Factors and Instruction Length

The user base is the group of users. The traffic generated by this group of users is treated as a single cloudlet. The number specified in this parameter, user grouping factor in user base, is the number of users to be grouped. Request grouping factor in data centers, specifies the number of requests to be grouped for processing in a single VM. Executable instruction length per request, denotes the execution length of the instruction. Table-5 lists the modified grouping factors and instruction length per request.

Table-5: Case 2 Configuration Parameters

SNo	Parameters	Values
Other Configurations		
1	User grouping factor	1000 No.s
2	Request grouping factor	100 No.s
3	Executable instruction length / Request	250 bytes

Table-6: Case 2 Results Comparison Table

Service Broker Policy	Overall Response Time (ms)
Closest datacenter	382.52
Reconfigure dynamically with load balancing	382.52
Optimize response time	380.5
Proposed optimum routing policy	151.33

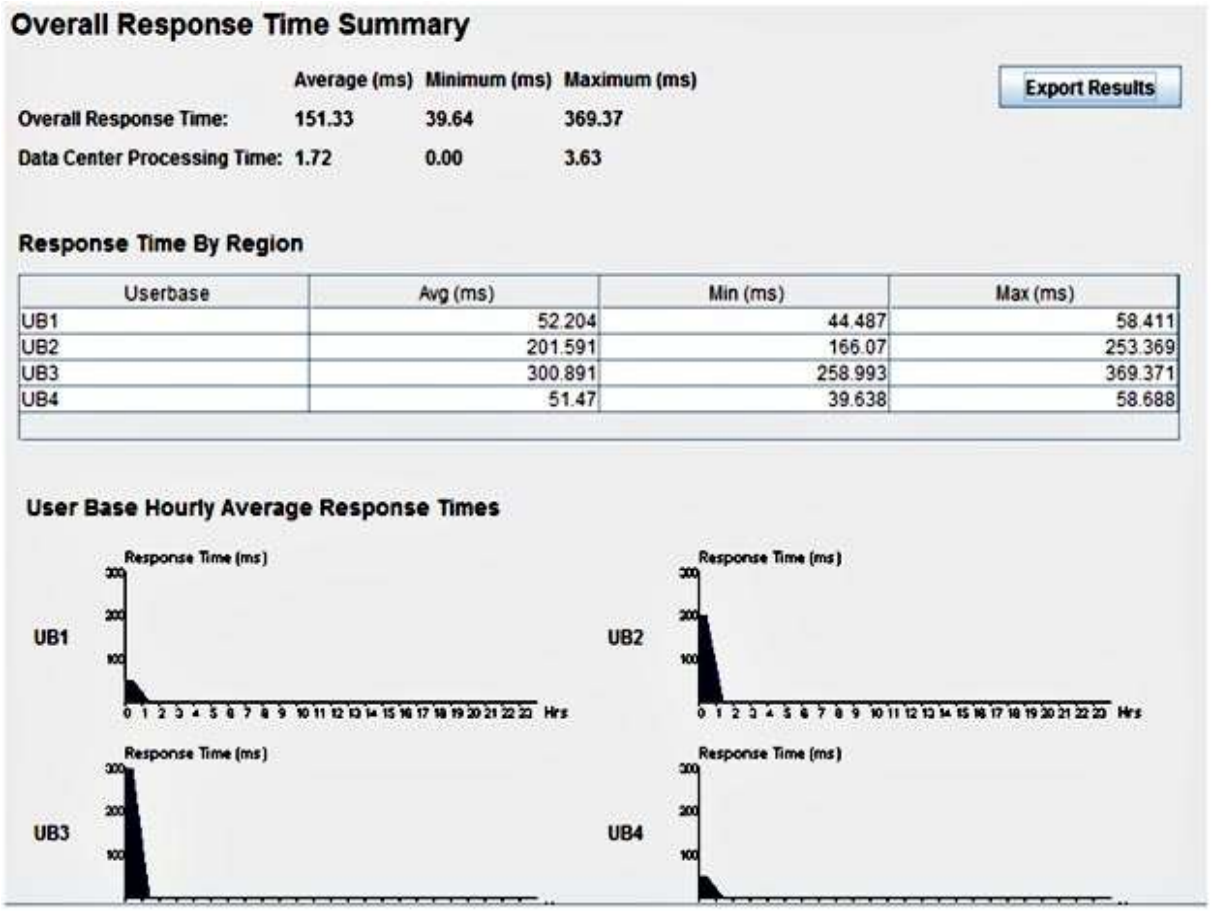

Fig-10: Case 2 Simulation Results

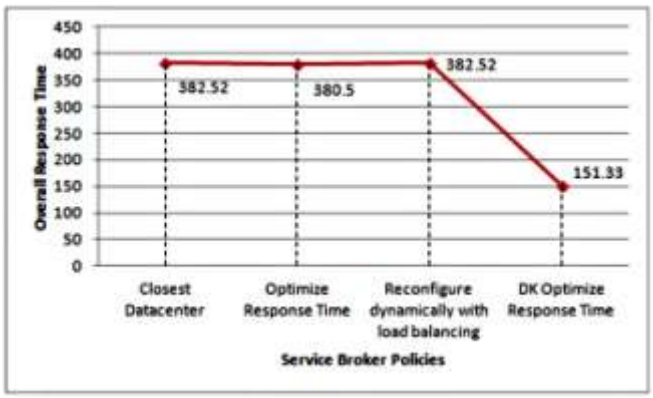

Fig-11: Case 2 Overall response time comparison

The simulation results for Table-5 parameters is put on view in Fig-10. The overall response time value attained through the proposed policy is stated in Table-6 and graphically compared in Fig-11. When measured up with other three service broker policies, the overall response time is reduced about 60% by the proposed policy.

5.3 Case 3: Real Time Value of Grouping Factors

In real cloud applications, each request from each user is executed individually, CloudAnalyst groups the requests as the individual traffic might slack the performance of the simulation. Table-7 have the simulation parameter values along with the real time value for grouping factors.

Table-7: Case 3 Configuration Parameters

SNo	Parameters	Values
Other Configurations		
1	User grouping factor	1 No.s
2	Request grouping factor	1 No.s
3	Executable instruction length / Request	1000 bytes

Table-8: Case 3 Results Comparison Table

Service Broker Policy	Overall Response Time (ms)
Closest Datacenter	378.91
Optimize Response Time	378.90
Reconfigure dynamically with load balancing	378.88
Proposed optimum routing policy	150.14

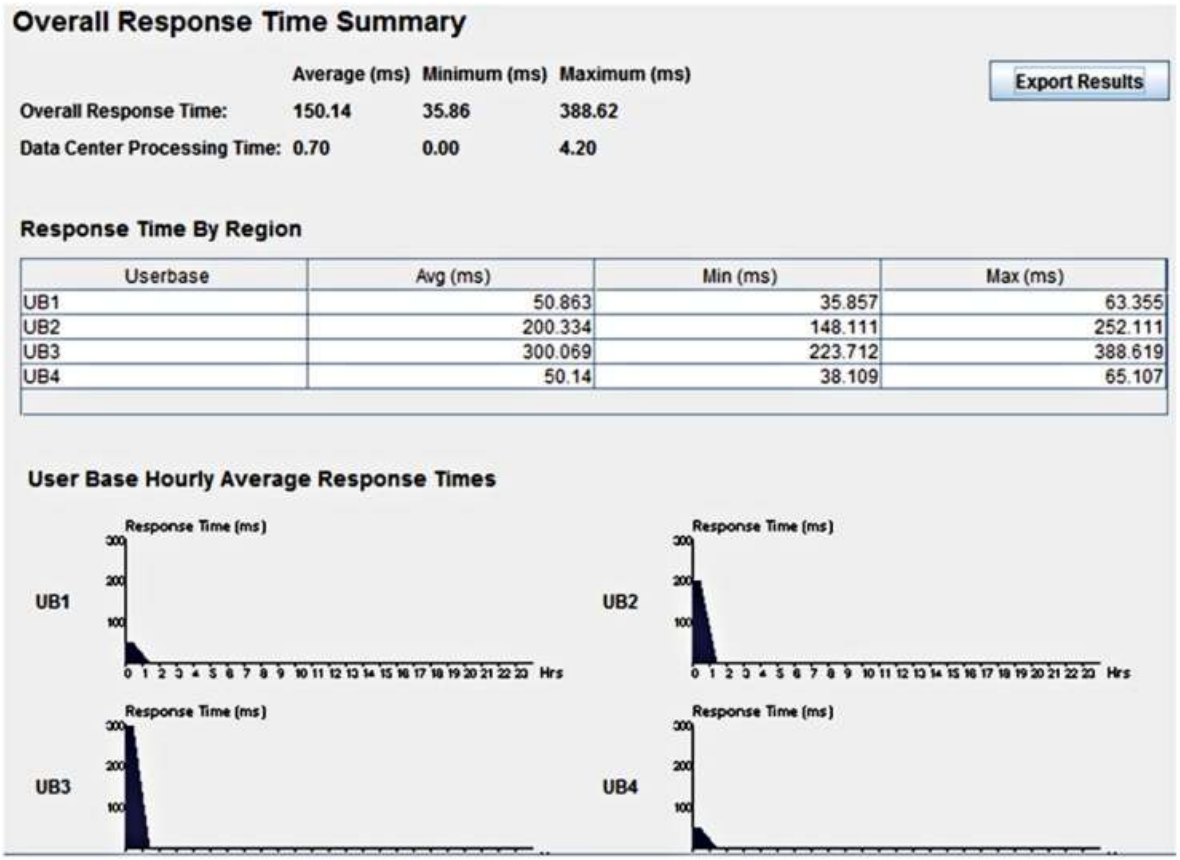

Fig-12: Case 3 Simulation Results

Table-7 lists the parameters with the real time grouping factors. The simulation results are displayed in Fig-12. The overall response time value obtained through the proposed policy is stated in Table-8 and compared graphically in Fig-13. 60% of betterment are evident by our proposed policy, while comparing with other three service broker policies.

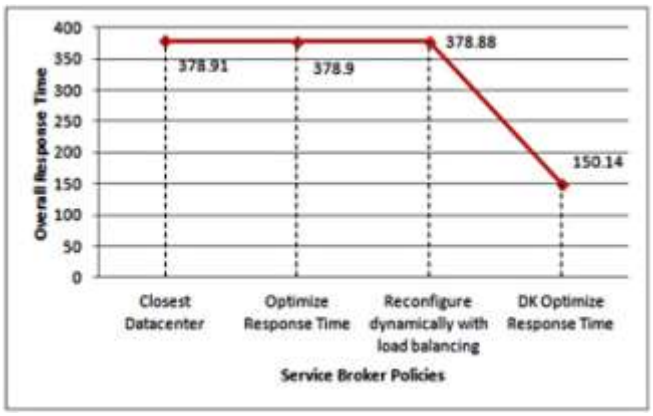

Fig-13: Case 3 Overall response time comparison

It is observed that the proposed policy shows an improvement in the overall response time. Thus the issue raised due to random selection of data center has been solved by the proposed algorithm in which random selection is not involved in any part of the algorithm.

6. CONCLUSION

The proposed optimum routing policy works better in terms of response time, by avoiding the random selection of a data center and by considering the bandwidth and latency parameter, to find the destination. The proposed policy is analyzed, first, with data centers and user bases at different region, secondly, with modified values in user grouping factor in user bases and request grouping factor in data center.

Finally, with the real time value of both, user grouping factor in user bases and request grouping factor in data center. In all the cases, the proposed optimum routing policy shows 60% of improvement in response time, when compared with other three service broker policies. The future scope of this work is to minimize the data center processing time, load balancing the requests among all the data centers and to reduce the virtual machine cost and data transfer cost.

REFERENCES

[1] Bhathiya Wickremasinghe and Rajkumar Buyya. "CloudAnalyst: A cloudsim-based tool for modelling and analysis of large scale cloud computing environments,", 2009, MEDC Project Report.

[2] Bhathiya Wickremasinghe, Rodrigo N Calheiros, and Rajkumar Buyya "CloudAnalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," *In Advanced Information Networking and Applications (AINA) 2010 24th IEEE International Conference on*, pp 446-452, 2010, IEEE.

[3] Xiaoyong Li; Huadong Ma; Feng Zhou and Wenbin Yao, "T-Broker: A Trust-Aware Service Brokering Scheme for Multiple Cloud Collaborative Services", *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 07 , Jul 2015, pp. 1402 – 1415.

[4] Rakesh Kumar Mishra, Sandeep Kumar, Sreenu Naik B, "Priority based Round Robin Service Broker Algorithm for CloudAnalyst", *Advance Computing Conference (IACC)*, Pages 878-881, Feb 2014 IEEE International.

[5] Rekha PM, Dakshayini M. "Cost based data center selection policy for large scale networks," *Proceedings of 2014 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*; Chennai.2014. pp. 18–23.

[6] Dhaval Limbani and Bhavesh Oza, "A Proposed Service Broker Strategy in CloudAnalyst for Cost Effective Data Center Selection," *International Journal of Engineering Research and Applications, India*, Vol. 2, Issue 1, Jan-Feb 2012, pp.793-797.

[7] Deepak Kapgate, "Efficient Service Broker Algorithm for Data Center Selection in Cloud Computing," International Journal of Computer Science and Mobile Computing, Vol. 3, Issue 1, Jan 2014, pp. 355-365.

[8] Kunal Kishore and Vivek Thapar, "An Efficient Service Broker Policy for Cloud Computing Environment," International Journal of Computer Science Trends and Technology, Vol. 2, Issue 4, Jul-Aug 2014, pp. 104-109.

[9] Mandeep Kaur and Verender Singh Madra, "Performance Evaluation of Virtual Machines using Service Broker Policies in Cloud Computing," International Journal of Science and Research, Vol 4, Issue 7, Jul 2015, pp. 344-347.

[10] Mandeep Kaur and Verender Singh Madra, "A review on Cloud Service Broker Policies," International Journal of Computer Science and Mobile Computing, Vol 4, Issue 5, May 2015, pp. 1077-1081.

[11] Radhakrishnan. A and Kavitha. V, "Future Load Aware Service Broker Policy for Infrastructure Requests in Cloud," Journal of Theoretical and Applied Informatio n Technology, Vol. 67, Issue. 2, Sep 2014, 345-352.

[12] Hetal V. Patel and Ritesh Patel, "CloudAnalyst: An Insight of Service Broker Policy," International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 1, Jan 2015, pp. 122-127.